EE270
Large scale matrix computation, optimization and learning

Instructor : Mert Pilanci

Stanford University

Tuesday, Feb 18 2020

Randomized Linear Algebra and Optimization
Lecture 13: Gradient Descent with Momentum and Preconditioning

Optimizing convex least squares cost

- Consider

$$\min_x \underbrace{\frac{1}{2}\|Ax - b\|_2^2}_{f(x)}$$

- gradient $\nabla f(x) = A^T(Ax - b)$
- Gradient Descent:

$$x_{t+1} = x_t - \mu A^T(Ax_t - b)$$

- fixed step size $\mu_t = \mu$

Optimizing convex least squares cost

- Basic (in)equality method

  (1) x^* minimizes $f(x)$, hence $\nabla f(x^*) = A^T(Ax^* - b) = 0$

  (2) $x_{t+1} = x_t - \mu A^T(Ax_t - b)$

  (3) define error $\Delta_t = x_t - x^*$

Optimizing convex least squares cost

- Basic (in)equality method

  (1) x^* minimizes $f(x)$, hence $\nabla f(x^*) = A^T(Ax^* - b) = 0$

  (2) $x_{t+1} = x_t - \mu A^T(Ax_t - b)$

  (3) define error $\Delta_t = x_t - x^*$

- $\Delta_{t+1} = \Delta_t - \mu A^T A \Delta_t$

Optimizing convex least squares cost

- run gradient descent M iterations, i.e., $t = 1, \dots, M$
- $\Delta_M = (I - \mu A^T A)^M \Delta_0$
- $\|\Delta_M\|_2 \leq \sigma_{\max}\left((I - \mu A^T A)^M\right) \|\Delta_0\|_2$
  $\sigma_{\max}\left(I - \mu A^T A\right)^M = \max_{i=1,..,d} \left|1 - \mu\lambda_i(A^T A)\right|^M$
  where λ_i is the i-th eigenvalue in decreasing order
- Define
  λ_- as the smallest eigenvalue of $A^T A$
  λ_+ as the largest eigenvalue of $A^T A$
- $\max_{i=1,..,d} \left|1 - \mu\lambda_i(A^T A)\right| = \max\left(\left|1 - \mu\lambda_-\right|, \left|1 - \mu\lambda_+\right|\right)$
- optimal step size that minimizes above
- $\min_{\mu \geq 0} \max\left(\left|1 - \mu\lambda_-\right|, \left|1 - \mu\lambda_+\right|\right)$
- optimal $\mu = \mu^*$ satisfies $\left|1 - \mu^*\lambda_-\right| = \left|1 - \mu^*\lambda_+\right|$
  which implies $\mu^* = \frac{2}{\lambda_+ + \lambda_-}$

Optimizing convex least squares cost

- Convergence rate using $\mu^* = \frac{2}{\lambda_+ + \lambda_-}$
- $\max\left(\left|1 - \mu^*\lambda_-\right|, \left|1 - \mu^*\lambda_+\right|\right) = \frac{\lambda_+ - \lambda_-}{\lambda_+ + \lambda_-}$
- $\|x_M - x^*\|_2 \leq \left(\frac{\lambda_+ - \lambda_-}{\lambda_+ + \lambda_-}\right)^M \|x_0 - x^*\|_2$

  convergence depends on the eigenvalues of $A^T A$

  Two extremes:
- Identical eigenvalues (extremely well conditioned) $\lambda_- = \lambda_+$, i.e., $\lambda_1 = \lambda_2 = \cdots = \lambda_d \implies$ convergence in one iteration
- Distant eigenvalues (poorly conditioned) $\lambda_+ \gg \lambda_-$ $\implies \frac{\lambda_+ - \lambda_-}{\lambda_+ + \lambda_-} \approx 1$ leads to slow convergence
- Condition number $\kappa := \frac{\lambda_+}{\lambda_-}$
- $\|x_M - x^*\|_2 \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^M \|x_0 - x^*\|_2$

Computational complexity

$$\|x_M - x^*\|_2 \leq \left(\frac{\kappa-1}{\kappa+1}\right)^M \|x_0 - x^*\|_2$$

- Initialize at $x_0 = 0$
- For ϵ accuracy, i.e., $\|x_M - x^*\|_2 \leq \epsilon$
- We need to set the number of iterations M to

$$M \log\left(\frac{\kappa - 1}{\kappa + 1}\right) + \log \|x^*\|_2 \leq \log(\epsilon)$$

- $M = O\left(\frac{\log(\frac{1}{\epsilon})}{\log(\frac{\kappa+1}{\kappa-1})}\right)$
- $\log\left(\frac{\kappa+1}{\kappa-1}\right) \approx \frac{2}{\kappa-1}$ for large κ
- $M = O\left(\frac{\log(\frac{1}{\epsilon})}{\log(\frac{\kappa+1}{\kappa-1})}\right) = O\big(\kappa \log(\frac{1}{\epsilon})\big)$ for large κ
- Total computational cost $\kappa n d \log(\frac{1}{\epsilon})$ for ϵ accuracy

Improving condition number dependence: momentum

- $\min_x f(x)$
- Gradient Descent with Momentum

$$x_{t+1} = x_t - \mu_t \nabla f(x_t) + \beta_t (x_t - x_{t-1})$$

- the term $\beta_t (x_t - x_{t-1})$ is referred to as **momentum**

Momentum

- Gradient Descent with Momentum

$$x_{t+1} = x_t - \mu_t \nabla f(x_t) + \beta_t (x_t - x_{t-1})$$

- related to a discretization of the second order ordinary differential equation

$$\ddot{x} + a\dot{x} + b\nabla f(x)$$

- which models the motion of a body in a potential field given by f

Momentum

- also called accelerated gradient descent, or heavy-ball method
- can be re-written as

$$\begin{aligned} p_t &= \beta_t p_{t-1} - \nabla f(x_t) \\ x_{t+1} &= x_t + \alpha_t p_t \end{aligned}$$

- p_t is the search direction
- there is a short-term memory
- typically we set $p_0 = 0$

Gradient Descent with Momentum for Least Squares Problems

- $\min_x f(x)$ where $f(x) = \|Ax - b\|_2^2$
- Gradient Descent with momentum (Heavy Ball Method)

$$x_{t+1} = x_t - \mu_t \nabla f(x_t) + \beta_t (x_t - x_{t-1})$$

- Recall that when $\beta = 0$ (Gradient Descent) we defined $\Delta_t := x_t - x^*$ where $x^* = A^\dagger b$ and established the recursion

$$\Delta_{t+1} = \left(I - \mu A^T A\right) \Delta_t$$

- Since there is one time step memory, consider $V_t := \|\Delta_{t+1}\|_2^2 + \|\Delta_t\|_2^2$ instead
- we can write V_t in terms of $V_{t-1} = \|\Delta_t\|_2^2 + \|\Delta_{t-1}\|_2^2$
- **Lyapunov analysis**

  V_t is an energy function that decays to zero and upper-bounds error, i.e., $\|\Delta_t\|_2^2 \leq V_t$

Convergence analysis

- $\min_x f(x)$ where $f(x) = \|Ax - b\|_2^2$
- Gradient Descent with momentum (Heavy Ball Method)

$$x_{t+1} = x_t - \mu_t \nabla f(x_t) + \beta_t (x_t - x_{t-1})$$

- let $\Delta_t := x_t - x^*$ where $x^* = A^\dagger b$
- note that $b = Ax^* + b^\perp$ and $\nabla f(x_t) = A^T A \Delta_t$

$$\begin{aligned} \begin{bmatrix} \Delta_{t+1} \\ \Delta_t \end{bmatrix} &= \begin{bmatrix} x_t - \alpha \nabla f(x_t) + \beta(x_t - x_{t-1}) - x^* \\ \Delta_t \end{bmatrix} \\ &= \begin{bmatrix} (1+\beta)I - \alpha A^T A & \beta I \\ I & 0 \end{bmatrix} \begin{bmatrix} \Delta_t \\ \Delta_{t-1} \end{bmatrix} \end{aligned}$$

Convergence analysis

- iterating for $t = 1, \dots, M$

$$\begin{bmatrix} \Delta_{M+1} \\ \Delta_M \end{bmatrix} = \begin{bmatrix} (1+\beta)I - \alpha A^T A & \beta I \\ I & 0 \end{bmatrix}^M \begin{bmatrix} \Delta_1 \\ \Delta_0 \end{bmatrix}$$

- taking norms

$$\begin{aligned} \left\| \begin{bmatrix} \Delta_{t+1} \\ \Delta_t \end{bmatrix} \right\|_2 &= \left\| \begin{bmatrix} (1+\beta)I - \alpha A^T A & \beta I \\ I & 0 \end{bmatrix}^M \begin{bmatrix} \Delta_t \\ \Delta_{t-1} \end{bmatrix} \right\|_2 \\ &\leq \sigma_{\max}\left(\begin{bmatrix} (1+\beta)I - \alpha A^T A & \beta I \\ I & 0 \end{bmatrix}^M \right) \left\| \begin{bmatrix} \Delta_t \\ \Delta_{t-1} \end{bmatrix} \right\|_2 \end{aligned}$$

Spectral Radius

- Let M be an $d \times d$ matrix with eigenvalues $\lambda_1, ..., \lambda_d$
- spectral radius is defined as

$$\rho(M) := \max_{i=1,..,d} |\lambda_i(M)|$$

Lemma (Gelfand's formula) $\lim_{k\to} \sigma_{\max}(M^k)^{\frac{1}{k}} = \rho(M)$

- Let λ_i denote the eigenvalues of $A^T A$ for $i = 1, ..., d$
- **Lemma** The eigenvalues of

$$\left[\begin{array}{cc} (1+\beta)I - \alpha A^T A & \beta I \\ I & 0 \end{array}\right]$$

are given by the eigenvalues of 2×2 matrices

$$\left[\begin{array}{cc} 1+\beta-\alpha\lambda_i & -\beta \\ 1 & 0 \end{array}\right]$$

- for $i = 1, ..., d$
- These are given by the roots of $u^2 - (1+\beta-\alpha\lambda_i)u + \beta = 0$
- setting $\alpha = \frac{4}{\sqrt{\lambda_+}+\sqrt{\lambda_-}}$ and $\beta = \frac{\sqrt{\lambda_+}-\sqrt{\lambda_-}}{\sqrt{\lambda_+}+\sqrt{\lambda_-}}$ yields
- spectral radius: $\rho\left(\left[\begin{array}{cc} (1+\beta)I - \alpha A^T A & \beta I \\ I & 0 \end{array}\right]\right) = \frac{\sqrt{\lambda_+}-\sqrt{\lambda_-}}{\sqrt{\lambda_+}+\sqrt{\lambda_-}}$

Convergence result

- setting $\alpha = \frac{4}{\sqrt{\lambda_+}+\sqrt{\lambda_-}}$ and $\beta = \frac{\sqrt{\lambda_+}-\sqrt{\lambda_-}}{\sqrt{\lambda_+}+\sqrt{\lambda_-}}$ yields

$$\left\| \begin{bmatrix} \Delta_{t+1} \\ \Delta_t \end{bmatrix} \right\|_2 \leq \sigma_{\max} \left(\frac{\sqrt{\lambda_+} - \sqrt{\lambda_-}}{\sqrt{\lambda_+} + \sqrt{\lambda_-}} \right)^M \left\| \begin{bmatrix} \Delta_t \\ \Delta_{t-1} \end{bmatrix} \right\|_2$$

Computational complexity

- Gradient Descent ($\beta = 0$) total computational cost $\kappa nd \log(\frac{1}{\epsilon})$ for ϵ accuracy
- Gradient Descent with Momentum total computational cost $\sqrt{\kappa} nd \log(\frac{1}{\epsilon})$ for ϵ accuracy
- we need to know eigenvalues of $A^T A$ to find optimal step-sizes

Computational complexity

- Gradient Descent ($\beta = 0$) total computational cost $\kappa nd \log(\frac{1}{\epsilon})$ for ϵ accuracy
- Gradient Descent with Momentum total computational cost $\sqrt{\kappa} nd \log(\frac{1}{\epsilon})$ for ϵ accuracy
- we need to know eigenvalues of $A^T A$ to find optimal step-sizes
- Conjugate Gradient doesn't require the eigenvalues explicitly and results in $\sqrt{\kappa} nd \log(\frac{1}{\epsilon})$ operations

Newton's Method

- Suppose f is twice differentiable, and consider a second order Taylor approximation at a point x_t

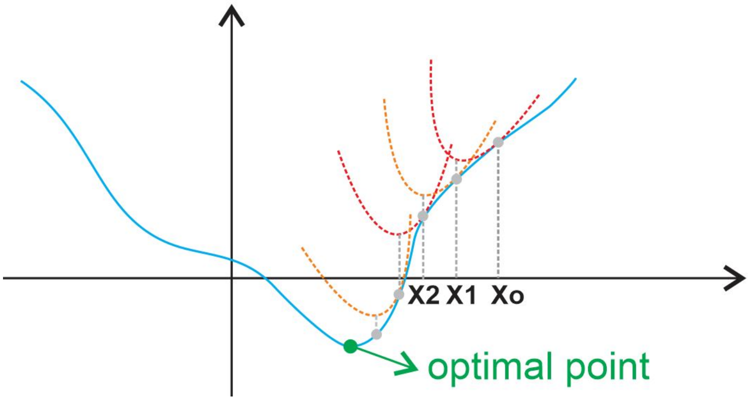
$$f(y) \approx f(x_t) + \nabla f(x_t)^T (y - x_t) + \frac{1}{2}(y - x^t)\nabla^2 f(x^t)(y - x^t)$$

- and minimize the approximation

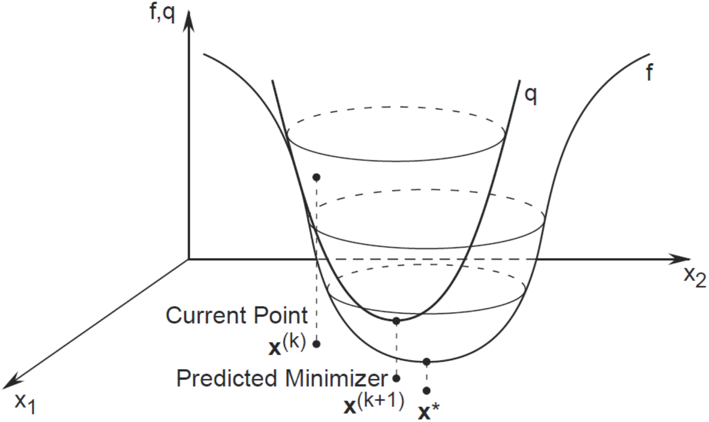
Newton's Method

- $x_{t+1} = x_t - \mu_t \left(\nabla^2 f(x)\right)^{-1} \nabla f(x)$
- complexity:

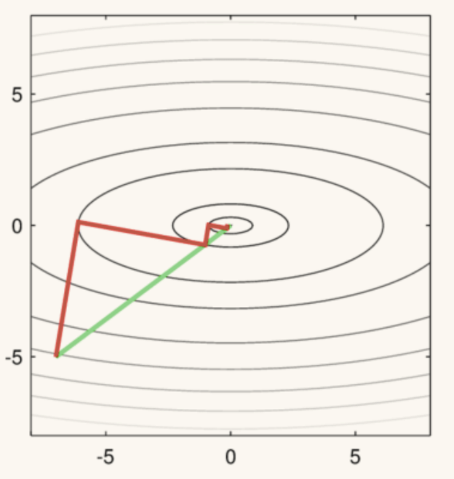
Newton's Method in one dimension



Newton's Method in higher dimensions



Newton's Method vs Gradient Descent



Newton's Method for least squares converges in one step

- Consider

$$\min_x \underbrace{\frac{1}{2}\|Ax-b\|_2^2}_{f(x)}$$

- gradient $\nabla f(x) = A^T(Ax-b)$
- Hessian $\nabla^2 f(x) = A^T A$
- Gradient Descent:

$$x_{t+1} = x_t - \mu A^T(Ax_t - b)$$

- Newton's Method:

$$x_{t+1} = x_t - \mu(A^T A)^{-1} A^T(Ax_t - b)$$

- fixed step size $\mu_t = \mu$

Questions?