EE270
Large scale matrix computation, optimization and learning

Instructor : Mert Pilanci

Stanford University

Tuesday, Feb 18 2020

Randomized Linear Algebra and Optimization
Lecture 15: Randomized Newton's Method

Recap: Gradient Descent for Convex Optimization Problems

- Strong convexity

  A convex function f is called strongly convex if there exists two positive constants $\beta_- \leq \beta_+$ such that

  $$\beta_- \leq \lambda_i \left(\nabla^2 f(x)\right) \leq \beta_+$$

  for every x in the domain of f

- Equivalent to

  $\lambda_{\min}(\nabla^2 f(x)) \geq \beta_-$
  $\lambda_{\max}(\nabla^2 f(x)) \leq \beta_+$

Gradient Descent for Strongly Convex Functions

- $x_{t+1} = x_t - \mu_t \nabla f(x_t)$
- Suppose that f is strongly convex with parameters β_-, β_+

  let $f^* := \min_x f(x)$

  Theorem
- Set constant step-size $\mu_t = \frac{1}{\beta_+}$

  $f(x_{t+1}) - f^* \leq (1 - \frac{\beta_-}{\beta_+})(f(x_t) - f^*)$

  recursively applying we get
- $f(x_M) - f^* \leq (1 - \frac{\beta_-}{\beta_+})^M (f(x_0) - f^*)$

Gradient Descent for Strongly Convex Functions

- $x_{t+1} = x_t - \mu \nabla f(x_t)$
- step-size $\mu = \frac{1}{\beta_+}$
- $f(x_M) - f^* \leq (1 - \frac{\beta_-}{\beta_+})^M (f(x_0) - f^*)$
- For optimizing functions $f(Ax)$

  computational complexity $O(\kappa nd \log(\frac{1}{\epsilon}))$

  where $\kappa = \frac{\beta_+}{\beta_-}$

Gradient Descent with Momentum (Heavy Ball Method) for Strongly Convex Functions

- $x_{t+1} = x_t - \mu \nabla f(x_t) + \beta(x_t - x_{t-1})$
- step-size parameter $\mu = \frac{4}{(\sqrt{\beta_+} + \sqrt{\beta_-})^2}$
- momentum parameter $\beta = \max\left(|1 - \sqrt{\mu\beta_-}|, |1 - \sqrt{\mu\beta_+}|\right)^2$
- For optimizing functions $f(Ax)$

  computational complexity $O(\sqrt{\kappa} nd \log(\frac{1}{\epsilon}))$

  where $\kappa = \frac{\beta_+}{\beta_-}$

Newton's Method

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

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

- minimizing the approximation yields $x_{t+1} = x_t + \left(\nabla^2 f(x)\right)^{-1} \nabla f(x)$
- $x_{t+1} = x_t - t\Delta_t$ where $\Delta_t := \left(\nabla^2 f(x)\right)^{-1} \nabla f(x)$
- for functions $f(Ax)$ where $A \in \mathbb{R}^{n \times d}$

  complexity $O(nd^2)$ to form the Hessian and $O(d^3)$ to invert

  or alternatively $O(nd^2)$ for factorizing the Hessian

Choosing step-sizes: backtracking (Armijo) line search

given a descent direction Δx for f at $x \in \mathbf{dom}\, f$, $\alpha \in (0, 0.5)$, $\beta \in (0, 1)$.
$t := 1.$
while $f(x + t\Delta x) > f(x) + \alpha t \nabla f(x)^T \Delta x, \quad t := \beta t.$

Newton's Method with Line Search

given a starting point $x \in \mathbf{dom}\, f$, tolerance $\epsilon > 0$.

repeat

1. *Compute the Newton step and decrement.*
   $\Delta x_{\mathrm{nt}} := -\nabla^2 f(x)^{-1} \nabla f(x); \quad \lambda^2 := \nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x).$
2. *Stopping criterion.* **quit** if $\lambda^2/2 \le \epsilon$.
3. *Line search.* Choose step size t by backtracking line search.
4. *Update.* $x := x + t\Delta x_{\mathrm{nt}}$.

Newton's Method for Strongly Convex Functions

- Strong convexity with parameters β_-, β_+
- Additional condition: Lipschitz continuity of the Hessian

  $$\|\nabla^2 f(x) - \nabla^2 f(y)\|_2 \leq L\|x - y\|_2^2$$

  for some constant $L > 0$
- **Theorem** The number of iterations for ϵ approximate solution in objective value is bounded by

  $$T := \text{constant} \times \frac{f(x_0) - f^*}{\beta_-/\beta_+^2} + \log_2 \log_2 \left(\frac{\epsilon_0}{\epsilon}\right)$$

  where $\epsilon_0 = 2\beta_-^3/L^2$.
- Computational complexity: $O((nd^2 + nd)T)$

Self-concordant Functions in $\mathbb{R}$

- A function $f : \mathbb{R} \to \mathbb{R}$ is self-concordant when f is convex and

$$f'''(x) \leq 2f''(x)^{3/2}$$

  for all x in the domain of f.
- examples: linear and quadratic functions, negative logarithm
- One can use a constant k other than 2 in the definition

Self-concordant Functions in $\mathbb{R}^d$

- A function $f : \mathbb{R}^d \to \mathbb{R}$ is self-concordant when it is self-concordant along every line, i.e.,
  (i) f is convex
  (ii) $g(t) := f(x + tv)$ is self-concordant for all x in the domain of f and all v

Self-concordant Functions in $\mathbb{R}^d$

- Scaling with a positive factor of at least 1 preserves self-concordance:

$$f \text{ is self concordant} \implies \alpha f \text{ is self concordant} \quad \text{for } \alpha \geq 1$$

- Addition preserves self-concordance

$$f_1 \text{ and } f_2 \text{ is self concordant} \implies f_1 + f_2 \text{ is self concordant}$$

- if $f(x)$ is self-concordant, affine transformations $g(x) := f(Ax + b)$ are also self-concordant

Newton's Method for Self-concordant Functions

- Suppose f is a self-concordant function
- **Theorem**

  Newton's method with line search finds an ϵ approximate point in less than

  $$T := \text{constant} \times (f(x_0) - f^*) + \log_2 \log_2 \frac{1}{\epsilon}$$

  iterations.
- Computational complexity: $T \times$ (cost of Newton Step)

  (Nesterov and Nemirovski)

Interior Point Programming

- Logarithmic Barrier Method

  Goal:

$$\min_x f_0(x) \text{ s.t. } f_i(x) \leq 0, \; i = 1, ..., n$$

  Indicator penalized form

$$\min_x f_0(x) + \sum_{i=1}^{n} \mathbb{I}(f_i(x))$$

  where $\mathbb{I}$ is a $\{0, \infty\}$ valued indicator function

Interior Point Programming

- Logarithmic Barrier Method

  Goal:

  $$\min_x f_0(x) \text{ s.t. } f_i(x) \leq 0,\ i = 1, \dots, n$$

  Indicator penalized form

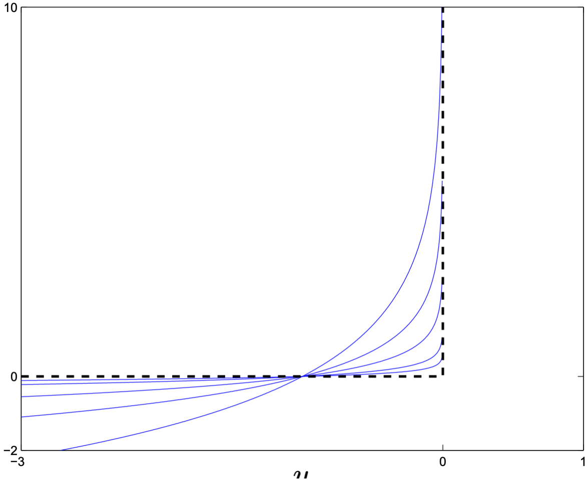
  $$\min_x f_0(x) + \sum_{i=1}^{n} \mathbb{I}(f_i(x))$$

  where $\mathbb{I}$ is a $\{0, \infty\}$ valued indicator function
- Approximation via $-t - \log(-\cdot)$

  $$\min_x f_0(x) - t \sum_{i=1}^{n} \log(-f_i(x))$$

- $t > 0$ is the barrier parameter

Interior Point Programming



Linear Programming

- LP in standard form where $A \in R^{n \times d}$

$$\min_{Ax \leq b} \; c^T x$$

- Logarithmic barrier approximation

$$\min_x c^T x - t \sum_{i=1}^{n} \log(b_i - a_i^T x)$$

- scaling with $\mu = \frac{1}{t}$

$$\min_x \mu c^T x - \sum_{i=1}^{n} \log(b_i - a_i^T x)$$

- self-concordant function

Linear Programming

- LP in standard form where $A \in R^{n \times d}$

$$\min_{Ax \leq b} \; c^T x$$

- Logarithmic barrier approximation

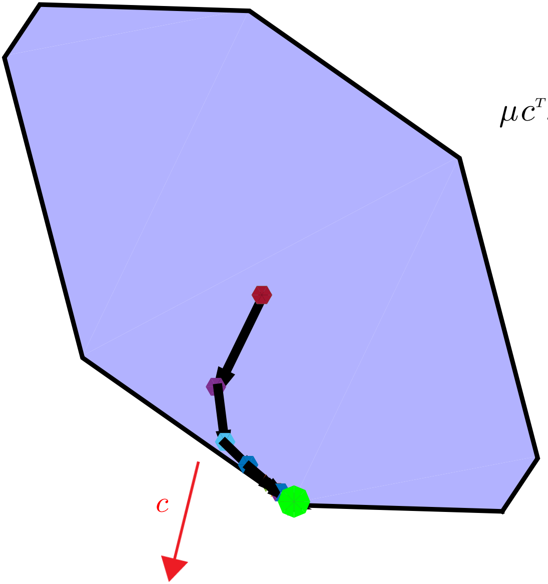
$$\min_x c^T x - t \sum_{i=1}^{n} \log(b_i - a_i^T x)$$

- scaling with $\mu = \frac{1}{t}$

$$\min_x \mu c^T x - \sum_{i=1}^{n} \log(b_i - a_i^T x)$$

- self-concordant function
- Hessian $\nabla^2 f(x) = A^T diag\left(\frac{1}{(b_i - a_i^T x)^2}\right) A$ takes $O(nd^2)$ operations

$$\min_{Ax \le b} \; c^T x$$



$$\mu c^T x - \sum_{i=1}^{n} \log(b_i - a_i^T x)$$

Randomized Newton's Method

- Suppose we want to find $\min_{x \in \mathcal{C}} g(x)$
- Randomized Newton's Method

$$x^{t+1} = \arg\min_{x \in \mathcal{C}} \; \langle \nabla g(x^t), x - x^t \rangle + \frac{1}{2}(x - x^t)^T \tilde{\nabla}^2 g(x^t)(x - x^t)$$

- $\tilde{\nabla}^2 g(x^t) \approx \nabla^2 g(x^t)$ is an approximate Hessian
- e.g., sketching $\tilde{\nabla}^2 g(x^t) = (\nabla^2 g(x^t))^{1/2} S^T S (\nabla^2 g(x^t))^{1/2}$

Randomized Newton's Method: Row Sampling Setch

- We may pick a row sampling matrix S
  as in Approximate Matrix Multiplication $A^T S^T S A \approx A^T A$

$$x^{t+1} = \arg\min_{x \in \mathcal{C}} \; \langle \nabla g(x^t), x - x^t \rangle + \frac{1}{2}(x - x^t)^T \tilde{\nabla}^2 g(x^t)(x - x^t)$$

- $\tilde{\nabla}^2 g(x^t) \approx \nabla^2 g(x^t)$ is a subsampled Hessian
- $\tilde{\nabla}^2 g(x^t) = (\nabla^2 g(x^t))^{1/2} S^T S (\nabla^2 g(x^t))^{1/2}$
- also called Subsampled Newton's Method[1]

[1] On the use of stochastic hessian information in optimization methods for machine learning, 2011, Byrd et al.

Interior Point Methods for Linear Programming

- Hessian of $f(x) = c^T x - \sum_{i=1}^{n} \log(b_i - a_i^T x)$

$$\nabla^2 f(x) = A^T diag\left(\frac{1}{(b_i - a_i^T x)^2}\right) A \; ,$$

Interior Point Methods for Linear Programming

- Hessian of $f(x) = c^T x - \sum_{i=1}^{n} \log(b_i - a_i^T x)$

$$\nabla^2 f(x) = A^T diag\left(\frac{1}{(b_i - a_i^T x)^2}\right) A \ ,$$

- Root of the Hessian

$$(\nabla^2 f(x))^{1/2} = diag\left(\frac{1}{|b_i - a_i^T x|}\right) A \ ,$$

Interior Point Methods for Linear Programming

- Hessian of $f(x) = c^T x - \sum_{i=1}^{n} \log(b_i - a_i^T x)$

$$\nabla^2 f(x) = A^T diag \left(\frac{1}{(b_i - a_i^T x)^2} \right) A \ ,$$

- Root of the Hessian

$$(\nabla^2 f(x))^{1/2} = diag \left(\frac{1}{|b_i - a_i^T x|} \right) A \ ,$$

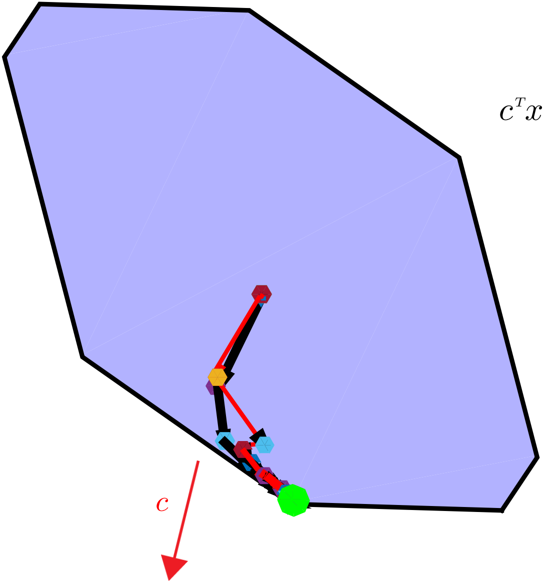
- Sketch of the Hessian

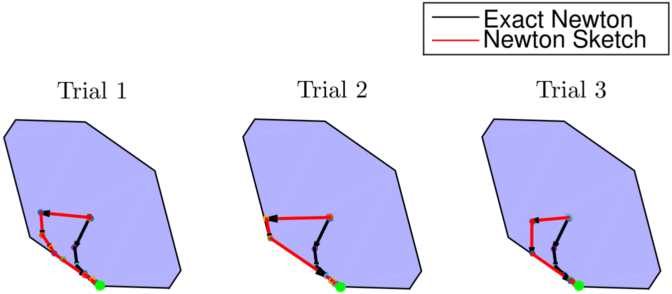
$$S^t (\nabla^2 f(x))^{1/2} = S^t diag \left(\frac{1}{|b_i - a_i^T x|} \right) A$$

takes $O(md^2)$ operations

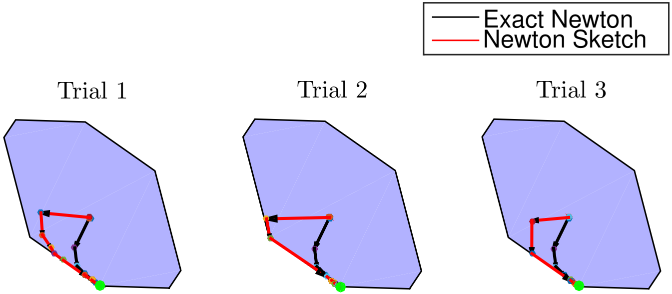
$$\min_{Ax \leq b} c^T x$$

$$c^T x - \mu \sum_{i=1}^{n} \log(b_i - a_i^T x)$$

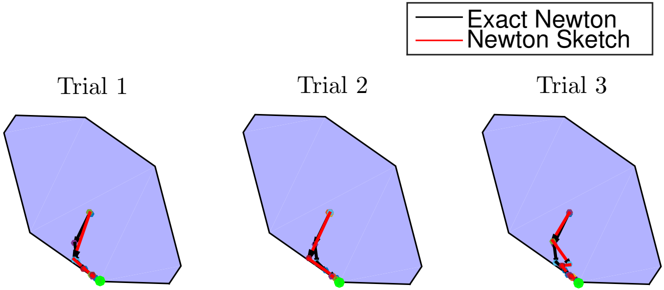




(a) sketch size $m = d$



(a) sketch size $m = d$



(b) sketch size $m = 4d$

Convergence of the Randomized Newton's Method

- Suppose f is a self-concordant function and S is a random projection matrix (e.g. Randomized Hadamard, Gaussian, CountSketch)
- **Theorem**

  Randomized Newton's method with line search finds an ϵ approximate point in less than

  $$T := \text{constant} \times (f(x_0) - f^*) + \log_2 \frac{1}{\epsilon}$$

  iterations.
- Computational Complexity: $nd \log n + nd \log_2 \frac{1}{\epsilon}$

References

- On the use of stochastic hessian information in optimization methods for machine learning, Byrd et al, SIAM Journal on Optimization, 2011
- Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence Pilanci and Wainwright - SIAM Journal on Optimization, 2017
- Sub-sampled Newton methods Roosta-Khorasani and Mahoney - Mathematical Programming, 2019 - Springer

Questions?