# EE270
# Large scale matrix computation, optimization and learning

Instructor : Mert Pilanci

Stanford University

Tuesday, Feb 18 2020

# Randomized Linear Algebra and Optimization
# Lecture 14: Second-Order Optimization Algorithms, Strong Convexity and Randomized Preconditioners

# Recap: Gradient Descent with 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**

# Computational complexity

- Gradient Descent ($\beta = 0$) total computational cost $\kappa nd \log(\frac{1}{\epsilon})$ for $\epsilon$ accuracy
- Gradient Descent with Momentum total computational cost $\sqrt{\kappa} nd \log(\frac{1}{\epsilon})$ for $\epsilon$ 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 $\epsilon$ accuracy
- Gradient Descent with Momentum total computational cost $\sqrt{\kappa} nd \log(\frac{1}{\epsilon})$ for $\epsilon$ 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

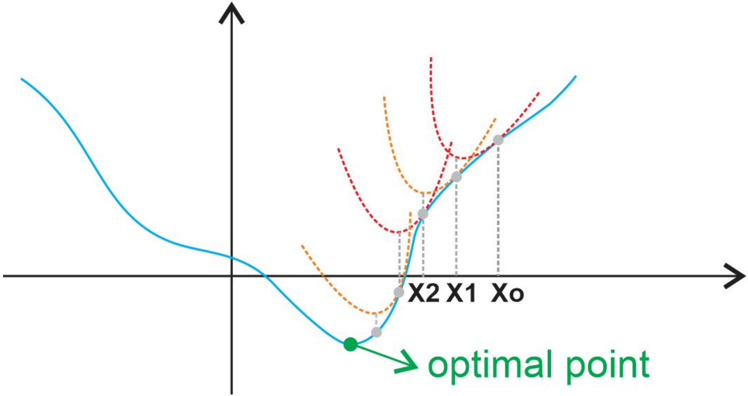
# Newton's Method

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

# Newton's Method in one dimension



# 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 $\mu_t = \mu$

# Newton's Method with Random Projection

- Randomized Newton's Method:

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

- fixed step size $\mu_t = \mu$
- computational cost:
- $O(nd \log n)$ to form $SA$ using Fast Johnson Lindenstrauss Transform and $O(d^3)$ to invert $(A^T S^T SA)^{-1}$
- alternatively $O(md^2)$ to factorize $SA$

# Analyzing Newton's Method with Random Projection

- Randomized Newton's Method:

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

- Define $\Delta_t = A(x_t - x^*)$

# Analyzing Newton's Method with Random Projection

- Randomized Newton's Method:

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

- Define $\Delta_t = A(x_t - x^*)$
  $\Delta_{t+1} = \Delta_t - \mu A(A^T S^T SA)^{-1} A^T \Delta_t$
- after $M$ iterations
  $\Delta_M = (I - \mu A(A^T S^T SA)^{-1} A^T)^M \Delta_0$

# Analyzing Newton's Method with Random Projection

- Let $A = U\Sigma V^T$ be the Singular Value Decomposition of A
- $A(A^T S^T S A)^{-1} A^T = U(U^T S^T S U) U^T$

  $\Delta_M = (I - \mu U(U^T S^T S U)^{-1} U^T)^M \Delta_0$
- $\Delta_t \in \text{Range}(A)$ implies $UU^T \Delta_t = \Delta_t$ and $\|U^T \Delta_t\|_2 = \|\Delta_t\|_2$

# Analyzing Newton's Method with Random Projection

- Let $A = U\Sigma V^T$ be the Singular Value Decomposition of A
- $A(A^T S^T S A)^{-1} A^T = U(U^T S^T S U) U^T$
  $\Delta_M = (I - \mu U(U^T S^T S U)^{-1} U^T)^M \Delta_0$
- $\Delta_t \in \text{Range}(A)$ implies $UU^T \Delta_t = \Delta_t$ and $\|U^T \Delta_t\|_2 = \|\Delta_t\|_2$
  $U^T \Delta_M = U^T (I - \mu U(U^T S^T S U)^{-1} U^T)^M UU^T \Delta_0$
- Note that
  $U^T (I - \mu U(U^T S^T S U)^{-1} U^T) = (I - \mu (U^T S^T S U)^{-1}) U^T$
  $U^T \Delta_M = (I - \mu (U^T S^T S U)^{-1})^M U^T \Delta_0$

# Analyzing Newton's Method with Random Projection

- Let $A = U\Sigma V^T$ be the Singular Value Decomposition of A
- $A(A^T S^T S A)^{-1} A^T = U(U^T S^T S U) U^T$
  $\Delta_M = (I - \mu U(U^T S^T S U)^{-1} U^T)^M \Delta_0$
- $\Delta_t \in \text{Range}(A)$ implies $UU^T \Delta_t = \Delta_t$ and $\|U^T \Delta_t\|_2 = \|\Delta_t\|_2$
  $U^T \Delta_M = U^T (I - \mu U(U^T S^T S U)^{-1} U^T)^M U U^T \Delta_0$
- Note that
  $U^T (I - \mu U(U^T S^T S U)^{-1} U^T) = (I - \mu (U^T S^T S U)^{-1}) U^T$
  $U^T \Delta_M = (I - \mu (U^T S^T S U)^{-1})^M U^T \Delta_0$
- $\|\Delta_M\|_2 \leq \sigma_{\max}\left(I - \mu (U^T S^T S U)^{-1})^M\right) \|\Delta_0\|_2$
- $\|\Delta_M\|_2 \leq \max_{i=1,\ldots,d} \left|1 - \mu \lambda_i((U^T S^T S U)^{-1})\right|^M \|\Delta_0\|_2$

# Eigenvalues of randomly projected matrices

- $\lambda_i((U^T S^T SU)^{-1}) = \lambda_i(U^T S^T SU)^{-1}$
- Recall that Approximate Matrix Multiplication for $U^T U = I$
  $\|\underbrace{U^T U}_{I} - U^T S^T SU\|_F \leq \epsilon$ implies
  $\sigma_{\max}\left(I - U^T S^T SU\right) \leq \epsilon$
- which is identical to $\left|1 - \lambda_i(U^T S^T SU)\right| \leq \epsilon \quad \forall i = 1, ..., d$
- All eigenvalues of $U^T S^T SU$ are in the range $[1 - \epsilon, 1 + \epsilon]$

# Optimal step-size

- All eigenvalues of $U^T S^T S U$ are in the range $[1-\epsilon, 1+\epsilon]$
- All eigenvalues of $(U^T S^T S U)^{-1}$ are in the range $[\frac{1}{1-\epsilon}, \frac{1}{1+\epsilon}]$

$$\|\Delta_M\|_2 \leq \max_{i=1,\ldots,d} \left|1-\mu\lambda_i((U^T S^T S U)^{-1})\right|^M \|\Delta_0\|_2 \tag{1}$$

$$= \max\left(\left|1-\mu\frac{1}{1-\epsilon}\right|, \left|1-\mu\frac{1}{1+\epsilon}\right|\right)^M \|\Delta_0\|_2 \tag{2}$$

- optimal step-size that minimizes the upper-bound satisfies

$$\left|1-\mu^*\frac{1}{1-\epsilon}\right| = \left|1-\mu^*\frac{1}{1+\epsilon}\right|$$

- $\mu^* = \frac{2}{\frac{1}{1-\epsilon}+\frac{1}{1+\epsilon}} = (1-\epsilon)(1+\epsilon)$

# Convergence rate

- $\mu^* = \frac{2}{\frac{1}{1-\epsilon}+\frac{1}{1+\epsilon}} = (1-\epsilon)(1+\epsilon)$

$$\begin{aligned}
\|\Delta_M\|_2 &\leq \max\left(\left|1-\mu\frac{1}{1-\epsilon}\right|, \left|1-\mu\frac{1}{1+\epsilon}\right|\right)^M \|\Delta_0\|_2 && (3)\\
&= \max\left(|1-(1+\epsilon)|, |1-(1-\epsilon)|\right)^M \|\Delta_0\|_2 && (4)\\
&= \epsilon^M \|\Delta_0\|_2 && (5)
\end{aligned}$$

# Row Sampling Setch

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

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

- $A^T S^T SA$ is a subsampled Hessian

# How to choose the sketch

- According to the convergence analysis we need $\|U^T S^T S U - U^T U\|_2 \leq \epsilon$ for some $\epsilon > 0$ since

$$\|\Delta_M\|_2 \leq \sigma_{\max}\left(I - \mu(U^T S^T S U)^{-1})^M\right)\|\Delta_0\|_2$$

  - Row sampling
    - Nonuniform row sampling. Probabilities $p_i = \frac{\|u_i\|_2^2}{\sum_{j=1}^n \|u_j\|_2^2}$ (leverage scores, or optimal AMM for $U^T U = I$)
    - Uniform row sampling
  - Johnson Lindenstrauss Embeddings:
    - i.i.d. Gaussian, Rademacher
    - Sparse JL Transform (one/few non-zeros per column)
    - Fast JL Transform (*PHD* based on Randomized Hadamard)

# Number of samples/sketches required

- In order to obtain the approximation

$$\mathbb{E}\|U^T S^T S U - U^T U\|_2 \leq \epsilon$$

- Row sampling
  - Nonuniform row sampling with $p_i = \frac{\|u_i\|_2^2}{\sum_{j=1}^n \|u_j\|_2^2}$
    $m = \frac{d \log d}{\epsilon^2}$ samples are needed
  - Uniform row sampling
    $m = \frac{\mu n \log(\mu n)}{\epsilon^2}$ samples are needed where
    $\mu := \mu(U) := \max_i \|u_i\|_2^2$
- Johnson Lindenstrauss Embeddings:
  - i.i.d. Gaussian, Rademacher $m = \frac{d}{\epsilon^2}$
  - Sparse JL Transform (one non-zeros per column) $m = \frac{d^2}{\epsilon^2}$
  - Sparse JL Transform ($O(\frac{\log d}{\epsilon})$ non-zeros per column) $m = \frac{d}{\epsilon^2}$
  - Fast JL Transform (Randomized Hadamard) $m = \frac{d \log d}{\epsilon^2}$

# Coherence of a matrix

- Coherence parameter is defined as
  $\mu := \mu(U) = \max_{i=1,\ldots,n} \|u_i\|_2^2$
- Note that ${u_i}^\top u_i = e_i^\top U U^\top e_i = e_i^\top P e_i = P_{ii}$ and $\mathbf{tr} P = d$ therefore $\frac{d}{n} \leq \mu_U \leq 1$
- Uniform row sampling
  $m = \frac{\mu n \log(\mu n)}{\epsilon^2}$ samples are required to obtain the subspace embedding

  $$\|U^T S^T S U - U^T U\|_2 \leq \epsilon$$

  $m$ can be between $\frac{d \log d}{\epsilon^2}$ (best case) and $\frac{n \log d}{\epsilon^2}$ (worst case) depending on the distribution of $\|u_i\|_2^2$
- Non-uniform (leverage score) sampling, or JL embeddings does not have the $\mu(U)$ coherence factor

# How to prove sampling results: Matrix Concentration

- Suppose that we sample the rows of $U$ non-uniformly wrt a distribution $p_i$, $i = 1, ., , n$. How large is the spectral norm error $\|U^T S^T S U - U^T U\|_2$? In AMM, we considered Frobenius norm error.
- Concentration of sums of matrices

  **Theorem:**[1] Let $\tilde{u}_1, ..., \tilde{u}_m$ be i.i.d. vectors such that $\|\tilde{u}_i\|_2 \leq B$, $\forall i$, then

$$\mathbb{E}\left\|\frac{1}{m}\sum_{j=1}^{m}\tilde{u}_j\tilde{u}_j^T - \mathbb{E}\tilde{u}_1\tilde{u}_1^T\right\|_2 \leq \epsilon := \text{constant} \times B\sqrt{\frac{\log m}{m}}$$

[1] Can be improved to a high probability result: Sampling from Large Matrices: An Approach through Geometric Functional Analysis, Rudelson and Vershynin, 2007

# How to prove sampling results: Matrix Concentration

- Suppose that we sample the rows of $U$ non-uniformly wrt a distribution $p_i$, $i = 1, ., , n$. How large is the spectral norm error $\|U^T S^T S U - U^T U\|_2$? In AMM, we considered Frobenius norm error.
- Concentration of sums of matrices

  **Theorem:**[1] Let $\tilde{u}_1, ..., \tilde{u}_m$ be i.i.d. vectors such that $\|\tilde{u}_i\|_2 \leq B$, $\forall i$, then

$$\mathbb{E}\left\|\frac{1}{m}\sum_{j=1}^{m}\tilde{u}_j\tilde{u}_j^T - \mathbb{E}\tilde{u}_1\tilde{u}_1^T\right\|_2 \leq \epsilon := \text{constant} \times B\sqrt{\frac{\log m}{m}}$$

- non-uniform row sampling $\tilde{u}_1 = u_i/\sqrt{p_i}$ with probability $p_i\,\forall i$. Note that $\mathbb{E}u_1u_1^T = \sum_{i=1}^n \frac{u_i}{\sqrt{p_i}}\frac{u_i}{\sqrt{p_i}}^T p_i = \sum_{i=1}^n u_i u_i^T = U^T U = I$. $B = \max_i \|u_i\|_2/\sqrt{p_i}$, ideally needs to be small.

[1] Can be improved to a high probability result: Sampling from Large Matrices: An Approach through Geometric Functional Analysis, Rudelson and Vershynin, 2007

# How to prove sampling results: Matrix Concentration

**Theorem:**[2] Let $\tilde{u}_1, \ldots, \tilde{u}_m$ be i.i.d. vectors such that $\|\tilde{u}_i\|_2 \le B$, $\forall i$, then

$$\mathbb{E}\left\| \frac{1}{m} \sum_{j=1}^{m} \tilde{u}_j \tilde{u}_j^T - \mathbb{E} \tilde{u}_1 \tilde{u}_1^T \right\|_2 \le \epsilon := \text{constant} \times B \sqrt{\frac{\log m}{m}}$$

- non-uniform row sampling $\tilde{u}_1 = u_i / \sqrt{p_i}$ with probability $p_i \, \forall i$.
  - Using leverage score distribution $p_i = \frac{\|u_i\|_2^2}{\sum_{j=1}^n \|u_j\|_2^2}$ we have $B = \max_i \|u_i\|_2 / \|u_i\|_2 \sum_{j=1}^n \|u_j\|_2^2 = \mathbf{tr}\, U^T U = d$
  - Using uniform distribution $p_i = \frac{1}{n}$, we have $B = \max_i \|u_i\|_2 / \sqrt{1/n} = n\mu(U)$ where $\mu(U) := \max_i \|u_i\|_2$ is the coherence parameter of $U$.
  - Picking $m = c \frac{B^2}{\epsilon^2} \log(\frac{B^2}{\epsilon^2})$ we obtain the sampling results $m = \frac{d \log d}{\epsilon^2}$ and $m = \frac{\mu n \log(\mu n)}{\epsilon^2}$ respectively.

[2] Can be improved to a high probability result: Sampling from Large Matrices: An Approach through Geometric Functional Analysis, Rudelson and Vershynin, 2007

# Computational complexity

- For $\epsilon$ accuracy in the objective value, i.e., $\|A\hat{x} - Ax^*\|_2 \leq \epsilon$
- Gradient Descent (GD) total computational cost $\kappa nd \log(\frac{1}{\epsilon})$
- Gradient Descent with Momentum (GD-M) total computational cost $\sqrt{\kappa} nd \log(\frac{1}{\epsilon})$
- Note that we need to know eigenvalues of $A^T A$ to find optimal step-sizes for GD and GD-M. Conjugate Gradient (CG) doesn't require the eigenvalues explicitly and results in $\sqrt{\kappa} nd \log(\frac{1}{\epsilon})$ operations
- Randomized Newton Method (using randomized Hadamard based fast JL, $m = \text{constant} \times d \log d$) total computational cost $nd \log n + d^3 \log d + ndlog(\frac{1}{\epsilon})$ for $n \gg d$, the complexity is $O(nd \log(1/\epsilon))$

  uniform row sampling, leverage score sampling and other sketching matrices also work with different sketch sizes.

# Preconditioning Least Squares Problems

$$\min_x \|Ax - b\|_2^2$$

- Convergence of GD, GD-M or CG depend on the condition number $\kappa := \frac{\lambda_{\max}(A^T A)}{\lambda_{\min}(A^T A)}$.
- We can precondition the problem by a variable change $x = Rx'$ where $R$ is an invertible matrix. Then, we form the problem

$$\min_{x'} \|ARx' - b\|_2^2$$

  whose solution is $(AR)^\dagger b = (R^T A^T AR)^{-1} R^T A^T b = R^{-1}(A^T A)^{-1} A^T b = R^{-1} A^\dagger b$.
  Then we can recover $x^* = Rx' = RR^{-1} A^\dagger b = A^\dagger b$
- Condition number of $AR$ can be better than $A$ for carefully chosen preconditioners $R$, and hence GD, GD-M or CG can converge faster. Ideally, eigenvalues of $R^T A^T AR$ should be all near 1.

# Preconditioning Trade-off

- original problem

$$\min_{x} \|Ax - b\|_2^2$$

- preconditioned problem

$$\min_{x'} \|ARx' - b\|_2^2$$

  - $R = I$ is the original problem $R^T A^T A R = A^T A$. Condition number is the same.
  - $R = (A^T A)^{-1}$ perfectly preconditions since $(A^T A)^{-1/2} A^T A (A^T A)^{-1/2} = I$. Condition number is 1.
    - Recovering the solution requires solving $A^T A x = x'$!

  we need a cheaply invertible matrix that preconditions the eigenvalues
  - example: diagonal preconditioner $R = \mathrm{diag}(A)^{-1}$

# Randomized Preconditioners

- original problem

$$\min_{x} \|Ax - b\|_2^2$$

- preconditioned problem

$$\min_{x'} \|ARx' - b\|_2^2$$

Condition number of $R^T A^T AR$ should be small.
exploring different options

- $R$ i.i.d random, e.g., Gaussian?

# Randomized Preconditioners

- original problem

$$\min_{x} \|Ax - b\|_2^2$$

- preconditioned problem

$$\min_{x'} \|ARx' - b\|_2^2$$

Condition number of $R^T A^T AR$ should be small.
exploring different options

- $R$ i.i.d random, e.g., Gaussian?
- $R = A^T S^T SA$?

# Randomized Preconditioners

- original problem

$$\min_x \|Ax - b\|_2^2$$

- preconditioned problem

$$\min_{x'} \|ARx' - b\|_2^2$$

Condition number of $R^T A^T A R$ should be small.
exploring different options

- $R$ i.i.d random, e.g., Gaussian?
- $R = A^T S^T S A$?
- Let $R = (A^T S^T S A)^{-1/2}$. Then we have

$$R^T A^T A R = (A^T S^T S A)^{-1/2} A^T A (A^T S^T S A)^{-1/2}$$

# Hessian Square Root $(A^T S^T SA)^{-1/2}$ Preconditioner

- Let $R = (A^T S^T SA)^{-1/2}$. Then we have
- Note that $R^T A^T AR$ and $ARR^T A^T$ have the same non-zero eigenvalues
- $ARR^T A^T = A(A^T S^T SA)^{-1/2}(A^T S^T SA)^{-1/2} A^T = A(A^T S^T SA)^{-1} A^T$

# Hessian Square Root $(A^T S^T SA)^{-1/2}$ Preconditioner

- Let $R = (A^T S^T SA)^{-1/2}$. Then we have
- Note that $R^T A^T AR$ and $ARR^T A^T$ have the same non-zero eigenvalues
- $ARR^T A^T = A(A^T S^T SA)^{-1/2}(A^T S^T SA)^{-1/2} A^T = A(A^T S^T SA)^{-1} A^T$
- Let $A = U\Sigma V^T$ the Singular Value Decomposition

  Then we have $A(A^T S^T SA)^{-1} A^T = U(U^T S^T SU)^{-1} U^T$, whose eigenvalues are the eigenvalues of $(U^T S^T SU)^{-1}$
- Therefore, subspace approximation $\|U^T S^T SU - I\|_2 \leq \epsilon$ implies that eigenvalues of $U^T S^T SU$ are in $(1-\epsilon, 1+\epsilon)$.
- Consequently, eigenvalues of $R^T A^T AR$ are also in $(1-\epsilon, 1+\epsilon)$, which improves the condition number to $\kappa(AR) = \frac{1+\epsilon}{1-\epsilon}$

  Non-uniform row sampling, uniform row sampling (with extra coherence dependence), JL embeddings will work

# Implementing Randomized Preconditioning

- Generate a sketching matrix $S$. Recall $R = (A^T S^T SA)^{-1/2}$
- Apply QR factorization to $SA$ to obtain $SA = Q_{SA} R_{SA}$ where $R_{SA}$ is upper triangular and $Q_{SA}$ is orthonormal.

  Observe that
  $R = (A^T S^T SA)^{-1/2} = (R_{SA}^T Q_{SA}^T Q_{SA} R_{SA})^{-1} = (R_{SA}^T R_{SA})^{-1/2}$
  and an inverse square root is given by $R_{SA}$

  Since $R_{SA}$ is upper triangular, we can apply it to vectors in linear time using back-substitution.

# Implementing Randomized Preconditioning

- Generate a sketching matrix $S$. Recall $R = (A^T S^T SA)^{-1/2}$
- Apply QR factorization to $SA$ to obtain $SA = Q_{SA} R_{SA}$ where $R_{SA}$ is upper triangular and $Q_{SA}$ is orthonormal.

  Observe that
  $R = (A^T S^T SA)^{-1/2} = (R_{SA}^T Q_{SA}^T Q_{SA} R_{SA})^{-1} = (R_{SA}^T R_{SA})^{-1/2}$
  and an inverse square root is given by $R_{SA}$

  Since $R_{SA}$ is upper triangular, we can apply it to vectors in linear time using back-substitution.
- Solve

$$\min_{x'} \|ARx' - b\|_2^2$$

  using Conjugate Gradient method or Gradient Descent with Momentum (since we know about the eigenvalues). Note that each steps requires gradient calculation $R^T A^T (A(Rx) - b)$, which can be done with back-substitution and matrix vector products

# Randomized Newton vs Preconditioning

- Both approaches remove the condition number dependence
- Randomized Preconditioning requires QR decomposition and back-substitution steps
- Randomized Newton (also called Iterative Hessian Sketch) is more flexible since QR decomposition is not required. We can use approximate sub-solvers

$$
\begin{aligned}
x^{t+1} &= x_t - (A^T S^T SA)^{-1} A^T (Ax_t - b) \\
&= x_t + \arg\min_z \frac{1}{2} \|SAz\|_2^2 + z^T (A^T (Ax_t - b))
\end{aligned}
$$

- e.g., CG to approximately solve the system $(A^T S^T SA) z = A^T (Ax_t - b)$
- Furthermore, Randomized Newton generalizes to arbitrary functions: $\textbf{HessianSketch}^{-1}\textbf{gradient}$

# 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 $\beta_-, \beta_+$

  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 $\mu = \frac{1}{\beta_+}$
- $f(x_M) - f^* \le (1 - \frac{\beta_-}{\beta_+})^M (f(x_0) - f^*)$
- For optimizing functions $f(Ax)$

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

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

Questions?

# References


- Improved analysis of the subsampled randomized Hadamard transform JA Tropp - Advances in Adaptive Data Analysis, 2011 - World Scientific
- Sampling from large matrices: An approach through geometric functional analysis M Rudelson, R Vershynin - Journal of the ACM (JACM), 2007
- A fast randomized algorithm for overdetermined linear least-squares regression V Rokhlin, M Tygert. Proceedings of the National Academy of Sciences, 2008
- OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings Jelani Nelson, Huy L. Nguyen, 2012
- Iterative Hessian sketch: Fast and accurate solution approximation for constrained least-squares M Pilanci, MJ Wainwright - The Journal of Machine Learning Research, 2016