CS-GY 9223 I: Lecture 7
Preconditioning, acceleration, coordinate decent, etc.

NYU Tandon School of Engineering, Prof. Christopher Musco

Recall from last lecture: a convex function f is β-smooth and α-strongly convex if, for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,

$$\frac{\alpha}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 \leq \nabla f(\mathbf{x})^T(\mathbf{x}-\mathbf{y}) - [f(\mathbf{x}) - f(\mathbf{y})] \leq \frac{\beta}{2}\|\mathbf{x}-\mathbf{y}\|_2^2.$$

Theorem (GD for β-smooth, α-strongly convex.)

Let f be a β-smooth and α-strongly convex function. If we run GD for T steps (with step size $\eta = \frac{1}{\beta}$) we have:

$$\|\mathbf{x}^{(t)} - \mathbf{x}^*\|_2^2 \leq e^{-(t-1)\frac{\alpha}{\beta}}\|\mathbf{x}^{(1)} - \mathbf{x}^*\|_2^2$$

$\kappa = \frac{\beta}{\alpha}$ is called the "condition number" of f.

Corollary: If $T = O\left(\kappa \log(1/\epsilon)\right)$ we have:

$$\|\mathbf{x}^{(T)} - \mathbf{x}^*\|_2 \leq \epsilon\|\mathbf{x}^{(1)} - \mathbf{x}^*\|_2.$$

Let $f(\mathbf{x}) = \|\mathbf{D}\mathbf{x} - \mathbf{b}\|_2^2$ where $\mathbf{D}$ is a diagonal matrix.

- $\beta = 2\max(\mathbf{D})^2$
- $\alpha = 2\min(\mathbf{D})^2$

Gradient descent on f:

- $\mathbf{x}^{(1)} = \mathbf{0}$
- For $t = 1, \ldots, T$
  - $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \frac{1}{\beta}(2\mathbf{D}(\mathbf{D}\mathbf{x} - \mathbf{b}))$

Theorem (GD for β-smooth, α-strongly convex.)

Let f be a β-smooth and α-strongly convex function. If we run GD for T steps (with step size $\eta = \frac{1}{\beta}$) we have:

$$\|\mathbf{x}^{(t)} - \mathbf{x}^*\|_2 \leq e^{-t/\kappa}\|\mathbf{x}^{(1)} - \mathbf{x}^*\|_2$$

Prove for $f(\mathbf{x}) = \|\mathbf{D}\mathbf{x} - \mathbf{b}\|_2^2$. You may assume that $\min(\mathbf{D})^2 > 0$.

Alternate view:

$$(\mathbf{x}^{(t+1)} - \mathbf{x}^*) = \left(\mathbf{I} - \frac{2}{\beta}\mathbf{D}^2\right)(\mathbf{x}^{(t)} - \mathbf{x}^*)$$

IN-CLASS EXERCISE

$$(\mathbf{x}^{(t+1)} - \mathbf{x}^*) = \left(\mathbf{I} - \frac{2}{\beta}\mathbf{D}^2\right)^t (\mathbf{x}^{(1)} - \mathbf{x}^*)$$

$$\|\mathbf{x}^{(t+1)} - \mathbf{x}^*\|_2 \leq \qquad ?$$

This same analysis holds for any linear system minimized via gradient descent:

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 = \min_{\mathbf{x}} \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{A}^T\mathbf{b}$$

Unrolling gradient decent updates leads to:

$$(\mathbf{x}^{(t+1)} - \mathbf{x}^*) = \left(\mathbf{I} - \eta\mathbf{A}^T\mathbf{A}\right)^t (\mathbf{x}^{(1)} - \mathbf{x}^*).$$

Quick linear algebra review:

- $\mathbf{A}^T\mathbf{A}$ is symmetric so has an orthogonal eigendecomposition: $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$.
  - $\mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{I}$.
  - $\mathbf{\Lambda}$ is diagonal with entries $\lambda_1 \geq \lambda_2 \geq \ldots, \lambda_d$.

Claim: $\lambda_d \geq 0$ (i.e., $\mathbf{A}^T\mathbf{A}$ is positive semidefinite).

Verify outside of class:

$f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ is $2\lambda_1$ smooth and $2\lambda_d$ strongly convex. So we have: $\kappa = \frac{\lambda_1}{\lambda_d}$

$$(\mathbf{x}^{(t+1)} - \mathbf{x}^*) = \left(\mathbf{I} - \eta \mathbf{A}^T\mathbf{A}\right)^t (\mathbf{x}^{(1)} - \mathbf{x}^*).$$

$$\left(\mathbf{I} - \eta \mathbf{A}^T\mathbf{A}\right)^t = \left(\mathbf{U}\left(\mathbf{I} - \eta \mathbf{\Lambda}\right)\mathbf{U}^T\right)^t = \mathbf{U}\left(\mathbf{I} - \eta \mathbf{\Lambda}\right)^t \mathbf{U}^T$$

$$\|\mathbf{x}^{(t+1)} - \mathbf{x}^*\|_2 =$$

We now have a really good understanding of gradient descent.

Number of iterations for ϵ error:

	G-Lipschitz	β-smooth
R bounded start	$O\left(\frac{G^2R^2}{\epsilon^2}\right)$	$O\left(\frac{\beta R^2}{\epsilon}\right)$
α-strong convex	$O\left(\frac{G^2}{\alpha\epsilon}\right)$	$O\left(\frac{\beta}{\alpha}\log(1/\epsilon)\right)$

How do we use this understanding to design faster algorithms?

ACCELERATION

Total runtime for solving linear regression via GD:

(time per iteration) x (number of iterations)

$$O(nd \cdot \kappa \log(1/\epsilon))$$

for $\mathbf{A} \in \mathbb{R}^{n \times d}$, $\mathbf{x} \in \mathbb{R}^{d}$, $\mathbf{b} \in \mathbb{R}^{n}$.

Theorem (Accelerated Iterative Regression)

Let $\mathbf{x}^* = \min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$. *There is an algorithm which finds* $\tilde{\mathbf{x}}$ *with* $\|\tilde{\mathbf{x}} - \mathbf{x}^*\|_2 \leq \epsilon \|\mathbf{x}^*\|_2$ *in time:*

$$O(nd \cdot \sqrt{\kappa \log(1/\epsilon)})$$

Claim: For any η, polynomial $p(z) = c_1 z + c_2 z^2 + \ldots + c_q z^q$ with $p(1) = \sum_{j=1}^{q} c_q = 1$, there is an algorithm running in $O(ndq)$ time which outputs $\tilde{\mathbf{x}}$ satisfying:

$$\tilde{\mathbf{x}} - \mathbf{x}^* = p(\mathbf{I} - \frac{1}{\eta}\mathbf{A}^T\mathbf{A})\mathbf{x}^*$$

For standard gradient descent, $p(z) = z^q$.

Claim: For any η, polynomial $p(z) = c_1 z + c_2 z^2 + \ldots + c_q z^q$ with $p(1) = \sum_{j=1}^{q} c_q = 1$, there is an algorithm running in $O(ndq)$ time which outputs $\tilde{\mathbf{x}}$ satisfying:

$$\tilde{\mathbf{x}} - \mathbf{x}^* = c_1 \cdot (\mathbf{I} - \eta \mathbf{A}^T\mathbf{A})\mathbf{x}^* + c_2 \cdot (\mathbf{I} - \eta \mathbf{A}^T\mathbf{A})^2\mathbf{x}^* + \ldots + c_q \cdot (\mathbf{I} - \eta \mathbf{A}^T\mathbf{A})^q\mathbf{x}^*$$

Claim: $c_j \cdot \mathbf{I} - \eta \mathbf{A}^T\mathbf{A})^j\mathbf{x}^* = c_j \cdot \mathbf{x}^* + p_j'(\mathbf{I} - \eta \mathbf{A}^T\mathbf{A})\mathbf{A}^T\mathbf{A}\mathbf{x}^*$ where p_j is a polynomial with degree $j - 1$.

Claim: For any η, polynomial $p(z) = c_1 z + c_2 z^2 + \ldots + c_q z^q$ with $p(1) = \sum_{j=1}^{q} c_q = 1$, there is an algorithm running in $O(ndq)$ time which outputs $\tilde{\mathbf{x}}$ satisfying:

$$\mathbf{x}^* - \tilde{\mathbf{x}} = (c_1 + c_2 + \ldots + c_q) \cdot \mathbf{x}^* + p'(\mathbf{I} - \eta \mathbf{A}^T\mathbf{A})\mathbf{A}^T\mathbf{A}\mathbf{x}^*$$

$\tilde{\mathbf{x}} = p'(\mathbf{I} - \eta \mathbf{A}^T\mathbf{A})\mathbf{A}^T\mathbf{b}$ where p' is a polynmial with degree $q - 1$.

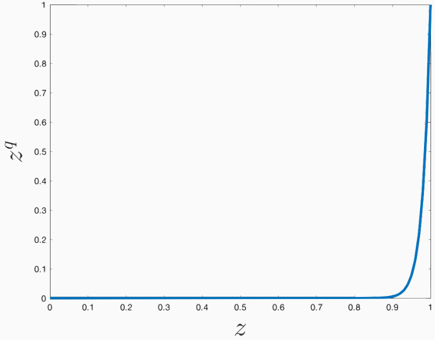
$$\tilde{x} - x^* = p(I - \eta A^T A)x^*$$
$$p(I - \eta A^T A) = U p(I - \eta \Lambda) U^T$$

$$\|\tilde{x} - x^*\| = \|U p(I - \eta \Lambda) U^T x^*\|_2$$
$$= \|p(I - \eta \Lambda) U^T x^*\|_2$$

As long as $\max\left[p(I - \eta \Lambda)\right] \leq \epsilon$,

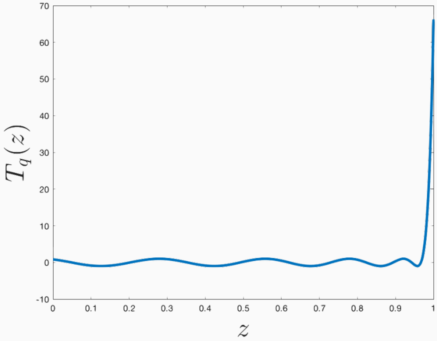
$$\|\tilde{x} - x^*\|_2 \leq \epsilon \|x^*\|_2$$

Goal: Find polynomial p such that $p(1) = 1$ and $p(z) \leq \epsilon$ for $z \in [0, 1 - \frac{1}{\kappa}]$.



Gradient descent uses $p(z) = z^{O(\kappa \log(1/\epsilon))}$.

Goal: Find polynomial p such that $p(1) = 1$ and $p(z) \leq \epsilon$ for $z \in [0, 1 - \frac{1}{\kappa}]$.

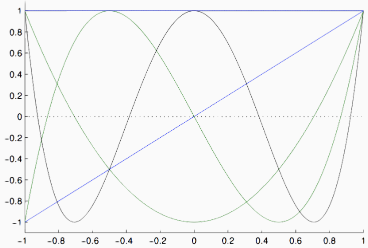


Can be done with degree $O(\sqrt{\kappa}\log(1/\epsilon))$ polynomial instead!

What are these polynomials?

Chebyshev polynomials of the first kind.

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_2(x) = 2x^2 - 1$$
$$\vdots$$
$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$$



"There's only one bullet in the gun. It's called the Chebyshev polynomial." – Prof. Rocco Servedio

Nesterov's accelerated gradient descent:

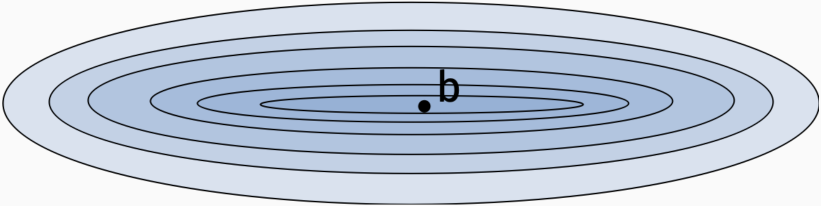
- $\mathbf{x}^{(1)} = \mathbf{y}^{(1)} = \mathbf{z}^{(1)}$
- For $t = 1, \ldots, T$
  - $\mathbf{y}^{(t+1)} = \mathbf{x}^{(t)} - \frac{1}{\beta}\nabla f(\mathbf{x}^{(t)})$
  - $\mathbf{x}^{(t+1)} = \left(1 + \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)\mathbf{y}^{(t+1)} - \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\mathbf{y}^{(t)}$

Theorem (AGD for β-smooth, α-strongly convex.)

Let f be a β-smooth and α-strongly convex function. If we run AGD for T steps we have:

$$f(\mathbf{x}^{(t)}) - f(\mathbf{x}^*) \leq \kappa e^{-(t-1)\sqrt{\kappa}} \left[f(\mathbf{x}^{(1)}) - f(\mathbf{x}^*)\right]$$

Corollary: If $T = O\left(\sqrt{\kappa}\log(\kappa/\epsilon)\right)$ achieve error ϵ.



Level sets of $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$.

Other terms for similar ideas:

- Momentum
- Heavy-ball methods

What if we look back beyond <u>two iterates</u>?

PRECONDITIONING

Main idea: Instead of minimizing $f(\mathbf{x})$, find another function $g(\mathbf{x})$ with the same minimum but which is better suited for first order optimization (e.g., has a smaller conditioner number).

Claim: Let $h(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ be an invertible function. Let $g(\mathbf{x}) = f(h(\mathbf{x}))$. Then

$$\min_{\mathbf{x}} f(\mathbf{x}) = \min_{\mathbf{x}} g(\mathbf{x}) \quad \text{and} \quad \arg\min_{\mathbf{x}} f(\mathbf{x}) = h\left(\arg\min_{\mathbf{x}} g(\mathbf{x})\right).$$

First Goal: We need $g(\mathbf{x})$ to still be convex.

Claim: Let $\mathbf{P}$ be an invertible $d \times d$ matrix and let $g(\mathbf{x}) = f(\mathbf{P}\mathbf{x})$.

$g(\mathbf{x})$ is always convex.

Second Goal:

$g(\mathbf{x})$ should have better condition number κ than $f(\mathbf{x})$.

Example:

- $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$. $\kappa_f = \frac{\lambda_1(\mathbf{A}^T\mathbf{A})}{\lambda_d(\mathbf{A}^T\mathbf{A})}$.
- $g(\mathbf{x}) = \|\mathbf{A}\mathbf{P}\mathbf{x} - \mathbf{b}\|_2^2$. $\kappa_g = \frac{\lambda_1(\mathbf{P}^T\mathbf{A}^T\mathbf{A}\mathbf{P})}{\lambda_d(\mathbf{P}^T\mathbf{A}^T\mathbf{A}\mathbf{P})}$.

Ideal preconditioner: Choose P so that $\mathbf{P}^T\mathbf{A}^T\mathbf{A}\mathbf{P} = \mathbf{I}$. For example, could set $P = \sqrt{(\mathbf{A}^T\mathbf{A})^{-1}}$.

What's the problem with this choice?

Third Goal: $\mathbf{P}$ should be easy to compute.

Many, many problem specific preconditioners are used in practice. There design is usually a heuristic process.

Example: Diagonal preconditioner.

- Let $\mathbf{D} = \text{diag}(\mathbf{A}^T\mathbf{A})$
- Intuitively, we roughly have that $\mathbf{D} \approx \mathbf{A}^T\mathbf{A}$.
- Let $\mathbf{P} = \sqrt{\mathbf{D}^{-1}}$

$\mathbf{P}$ is often called a **Jacobi preconditioner**. Often works very well in practice!

```
A =

    -734      1      33     9111      0
     -31     -2     108     5946    -19
     232     -1     101     3502     10
     426      0     -65    12503      9
    -373      0      26     9298      0
    -236     -2     -94     2398     -1
    2024      0    -132    -6904    -25
   -2258     -1      92    -6516      6
    2229      0       0    11921    -22
     338      1      -5   -16118    -23
```

```
>> cond(A'*A)

ans =

   8.4145e+07
```

```
>> P = sqrt(inv(diag(diag(A'*A))));
>> cond(P*A'*A*P)

ans =

   10.3878
```

Can you think of an example $\mathbf{A}$ where Jacobi preconditioning doesn't decrease a large κ?

Can Jacobi preconditioning increase κ?

Another view: If $g(\mathbf{x}) = f(\mathbf{P}\mathbf{x})$ then $\nabla g(\mathbf{x}) = \mathbf{P}^T \nabla f(\mathbf{P}\mathbf{x})$.

$\nabla g(\mathbf{x}) = \mathbf{P}\nabla f(\mathbf{P}\mathbf{x})$ when $\mathbf{P}$ is symmetric.

Gradient descent on g:

- For $t = 1, \ldots, T$,
  - $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta \mathbf{P}\left[\nabla f(\mathbf{P}\mathbf{x}^{(t)})\right]$

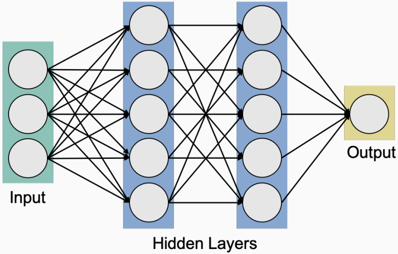
Gradient descent on g:

- For $t = 1, \ldots, T$,
  - $\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)} - \eta \mathbf{P}^2\left[\nabla f(\mathbf{y}^{(t)})\right]$

When $\mathbf{P}$ is diagonal, this is just gradient descent with a different step size for each parameter!

Algorithms based on this idea:

- AdaGrad
- RMSprop
- Adam optimizer



(Pretty much all of the most widely used optimization methods for training neural networks.)

COORDINATE DESCENT

Main idea: Trade slower convergence (more iterations) for cheaper iterations.

Stochastic Gradient Descent: When $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x})$, approximate $\nabla f(\mathbf{x})$ with $\nabla f_i(\mathbf{x})$ for randomly chosen i.

Main idea: Trade slower convergence (more iterations) for cheaper iterations.

Stochastic Coordinate Descent: Only compute a single random entry of $\nabla f(\mathbf{x})$ on each iteration:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ \frac{\partial f}{\partial x_2}(\mathbf{x}) \\ \vdots \\ \frac{\partial f}{\partial x_d}(\mathbf{x}) \end{bmatrix} \qquad \nabla_i f(\mathbf{x}) = \begin{bmatrix} 0 \\ \frac{\partial f}{\partial x_i}(\mathbf{x}) \\ \vdots \\ 0 \end{bmatrix}$$

Update: $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \eta \nabla_i f(\mathbf{x}^{(t)})$.

When $\mathbf{x}$ has d parameters, computing $\nabla_i f(\mathbf{x})$ often costs just a $1/d$ fraction of what it costs to compute $\nabla f(\mathbf{x})$

Example: $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ for $\mathbf{A} \in \mathbb{R}^{n \times d}, \mathbf{x} \in \mathbb{R}^d, \mathbf{b} \in \mathbb{R}^n$.

- $\nabla f(\mathbf{x}) = 2\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{A}^T\mathbf{b}$.
- $\nabla_i f(\mathbf{x}) = 2\left[\mathbf{A}^T\mathbf{A}\mathbf{x}\right]_i - 2\left[\mathbf{A}^T\mathbf{b}\right]$.

Stochastic Coordinate Descent:

- Choose number of steps T and step size η.
- For $i = 1, \ldots, T$:
    - Pick random $j_i \in 1, \ldots, d$.
    - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla_{j_i} f(\mathbf{x}^{(i)})$
- Return $\hat{\mathbf{x}} = \frac{1}{T} \sum_{i=1}^{T} \mathbf{x}^{(i)}$.

Theorem (Stochastic Coordinate Descent convergence)

Given a G-Lipschitz function f with minimizer $\mathbf{x}^*$ *and initial point* $\mathbf{x}^{(1)}$ *with* $\|\mathbf{x}^{(1)} - \mathbf{x}^*\|_2 \leq R$*, SCD with step size* $\eta = \frac{1}{Rd}$ *satisfies the guarantee:*

$$\mathbb{E}[f(\hat{\mathbf{x}}) - f(\mathbf{x}^*)] \leq \frac{2GR}{\sqrt{T/d}}$$

Often it doesn't make sense to sample i uniformly at random:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -.5 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 10 \\ 42 \\ -11 \\ -51 \\ 34 \\ -22 \end{bmatrix}$$

Select indices i proportional to $\|\mathbf{a}_i\|_2^2$:

$$\Pr[\text{select index } i \text{ to update}] = \frac{\|\mathbf{a}_i\|_2^2}{\sum_{j=1}^{d} \|\mathbf{a}_j\|_2^2} = \frac{\|\mathbf{a}_i\|_2^2}{\|\mathbf{A}\|_2^2}$$