CS-GY 9223 D: Lecture 8
Acceleration, preconditioning, coordinate methods

NYU Tandon School of Engineering, Prof. Christopher Musco

IMPROVING GRADIENT DESCENT

We now have a 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

Nesterov's accelerated gradient descent:

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

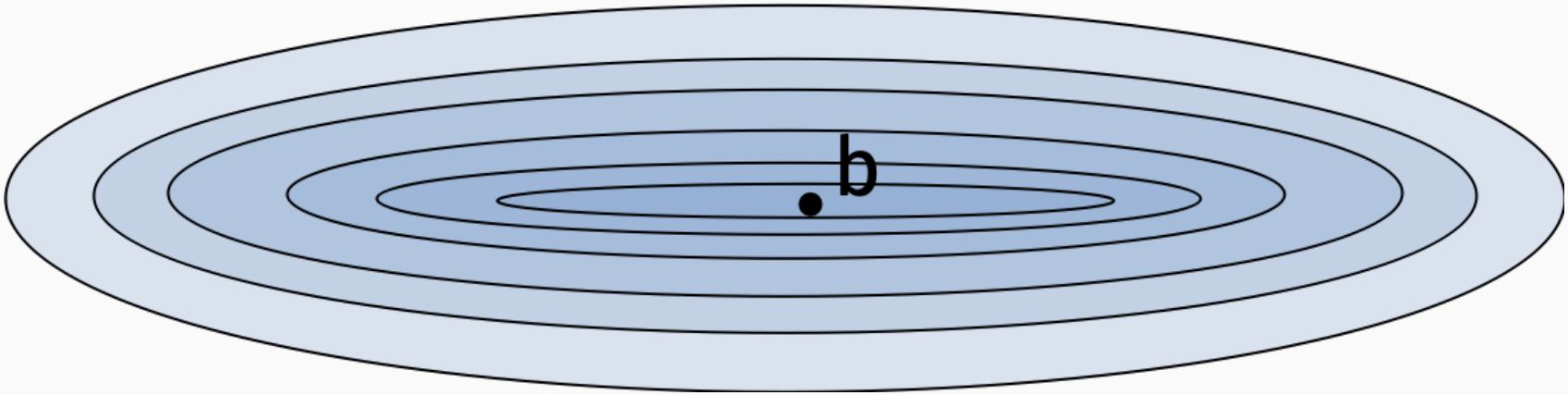
Theorem (AGD for β-smooth, α-strongly convex.)

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.

INTUITION BEHIND ACCELERATION



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 two iterates?

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 $h(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$ 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.

If $\mathbf{y}^* = \arg\min g(\mathbf{y})$, then $\mathbf{x}^* = \mathbf{P}\mathbf{y}^*$ minimizes $f(\mathbf{x})$.

Second Goal:

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

High dimensional chain rule:

$$\text{If } g(\mathbf{x}) = f(\mathbf{P}\mathbf{x}), \qquad \nabla^2 g(\mathbf{x}) = \nabla^2 \mathbf{P}^T f(\mathbf{P}\mathbf{x})\mathbf{P}.$$

Recall that the condition number is equal to:

$$\max_{\mathbf{x}} \frac{\lambda_{\max}\left(\nabla^2 g(\mathbf{x})\right)}{\lambda_{\min}\left(\nabla^2 g(\mathbf{x})\right)}$$

Example:

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

Ideal preconditioner: Choose P so that $\mathbf{P}^T\mathbf{A}^T\mathbf{A}\mathbf{P} = \mathbf{I}$. For example, could set $P = \sqrt{(\mathbf{A}^T\mathbf{A})^{-1}}$. But obviously this is too expensive to compute.

DIAGONAL PRECONDITIONER

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 for least squares problems.

- Let $\mathbf{D} = \mathbf{diag}(\mathbf{A}^T\mathbf{A})$
- Want $\mathbf{P}\mathbf{A}^T\mathbf{A}\mathbf{P}$ to be close to identity $\mathbf{I}$.
- Let $\mathbf{P} = \sqrt{\mathbf{D}^{-1}}$

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

DIAGONAL PRECONDITIONER

```
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
```

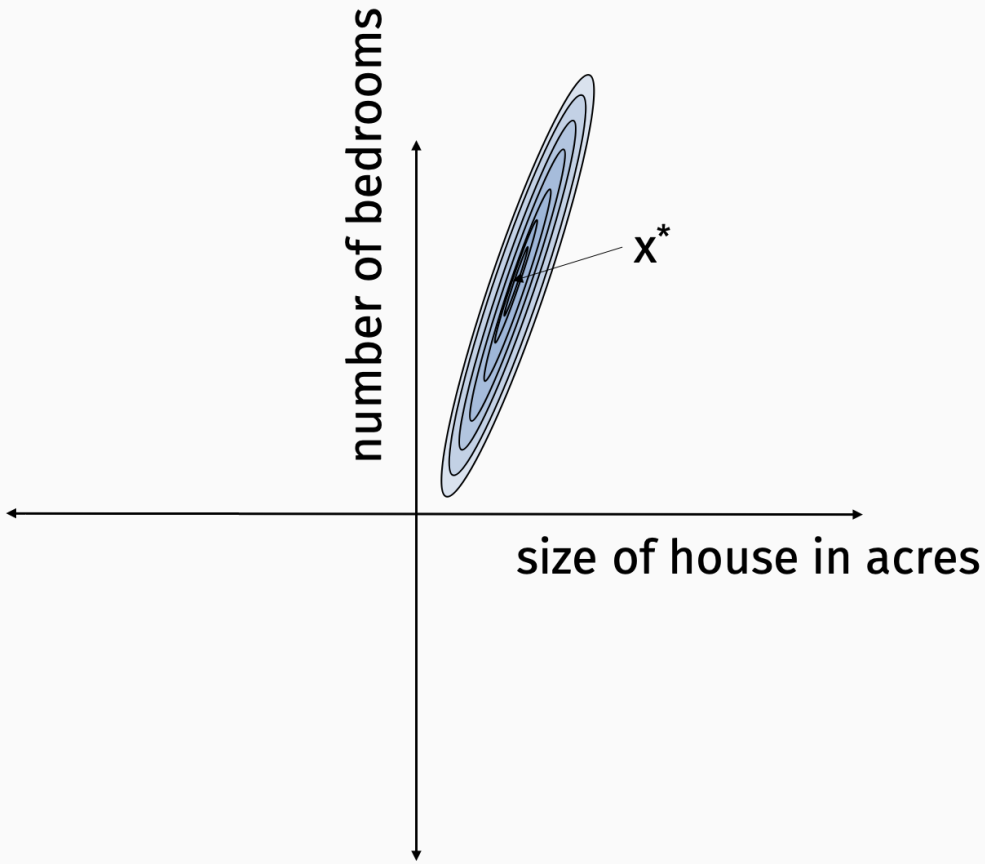
```
>> P = sqrt(inv(diag(diag(A'*A))));
>> cond(P*A'*A*P)

ans =

   10.3878
```

DIAGONAL PRECONDITIONER INTUITION

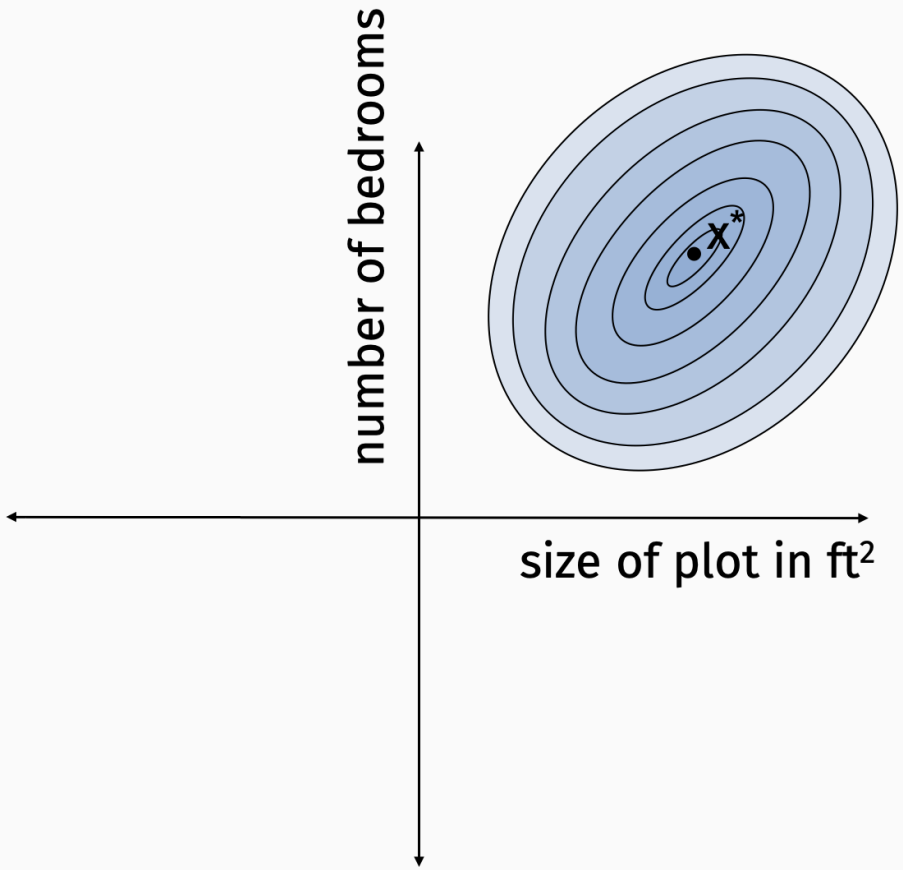
$g(\mathbf{x}) = f(\|\mathbf{APx} - \mathbf{b}\|_2^2)$ is the same least squares problem as $f(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|_2^2$, but with each feature (column of $\mathbf{A}$) scaled differently. The i^{th} column is scaled by P_{ii}.



Feature scaling can have a huge impact on conditioning.

DIAGONAL PRECONDITIONER INTUITION

$g(\mathbf{x}) = f(\|\mathbf{APx} - \mathbf{b}\|_2^2)$ is the same least squares problem as $f(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|_2^2$, but with each feature (column of $\mathbf{A}$) scaled differently. The i^{th} column is scaled by P_{ii}.



Feature scaling can have a huge impact on conditioning.

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

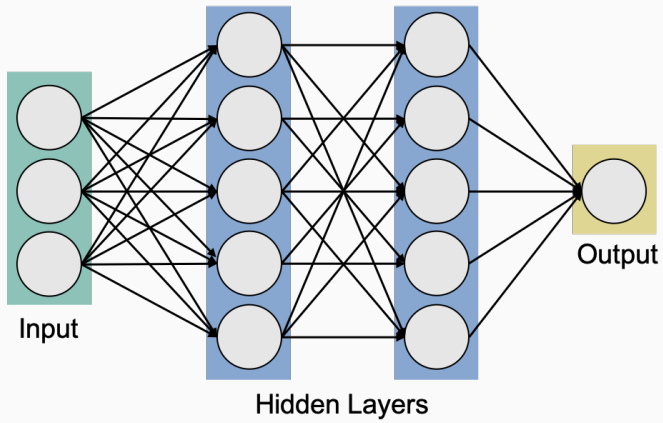
When $\mathbf{P}$ is diagonal, this is just gradient descent with a different step size for each parameter!

ADAPTIVE STEPSIZES

Less clear how to set P for general optimization problems where the Hessian is changing, but lots of heuristic algorithms based on this idea:

- AdaGrad, AdaDelta
- 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 $\nabla_i f(\mathbf{x})$ sometimes 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 $\mathbf{A} \in \mathbb{R}^{n \times d}, \mathbf{x} \in \mathbb{R}^d, \mathbf{b} \in \mathbb{R}^n$.

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

Computing full gradient takes $O(nd)$ time. Can we do better here?

When $\mathbf{x}$ has d parameters, computing $\nabla_i f(\mathbf{x})$ sometimes 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 $\mathbf{A} \in \mathbb{R}^{n \times d}, \mathbf{x} \in \mathbb{R}^d, \mathbf{b} \in \mathbb{R}^n$.

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

- $\mathbf{A}\mathbf{x}^{(t+1)} = \mathbf{A}\left(\mathbf{x}^{(t)} + c \cdot \mathbf{e}_i\right)$ — $O(n)$ time
- $2\left[\mathbf{A}^T\left(\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{b}\right)\right]_i$ — $O(n)$ time

Stochastic Coordinate Descent:

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

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 $\|\mathbf{a}_i\|_2^2$:

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

Let's analyze this approach.

Specialization of SCD to $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$:

Randomized Coordinate Descent (Strohmer, Vershynin 2007 / Leventhal, Lewis 2018)

- For iterate $\mathbf{x}^{(t)}$, let $\mathbf{r}^{(t)}$ be the residual:

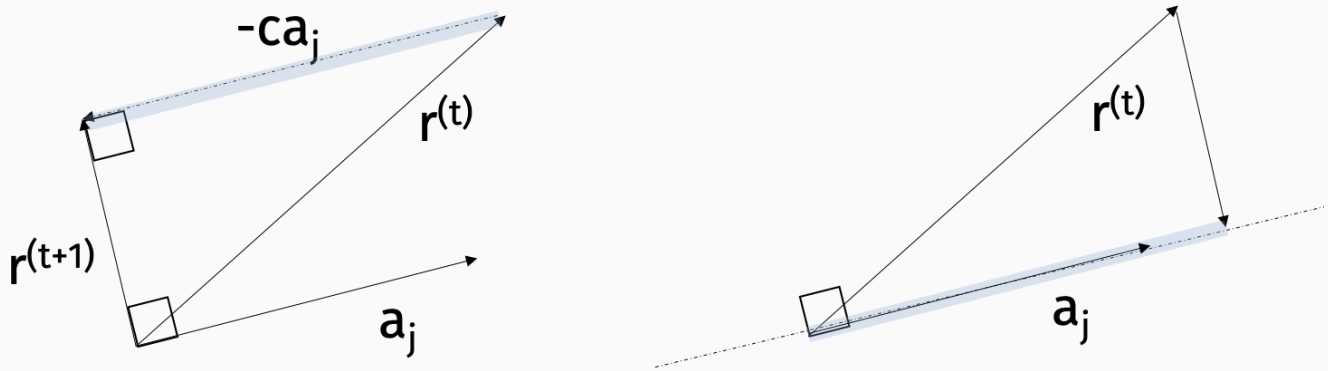
$$\mathbf{r}^{(t)} = \mathbf{A}\mathbf{x}^{(t)} - \mathbf{b}$$

- $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - c\mathbf{e}_j$.
- $\mathbf{r}^{(t+1)} = \mathbf{r}^{(t)} - c\mathbf{a}_j$. Here $\mathbf{a}_j$ is the i^{th} column of $\mathbf{A}$.

Typically c depends on fixed learning rate. Here we will choose it optimally – similar idea to gradient descent with line search.

What choice for c minimizes $\|\mathbf{r}^{(t+1)}\|_2^2$?

- $\|\mathbf{r}^{(t+1)}\|_2^2 = \|\mathbf{r}^{(t)} - c\mathbf{a}_j\|_2^2$
- Requires projecting $\mathbf{r}^{(t)}$ onto perpendicular of $\mathbf{a}_j$.



- $c = \dfrac{\mathbf{a}_j^T \mathbf{r}^{(t)}}{\|\mathbf{a}_j\|_2^2}$

Note that $\|\mathbf{r}^{(t+1)}\|_2^2 = \|\mathbf{r}^{(t)}\|_2^2 - \|c\mathbf{a}_j\|_2^2 = \|\mathbf{r}^{(t)}\|_2^2 - \dfrac{(\mathbf{a}_j^T \mathbf{r}^{(t)})^2}{\|\mathbf{a}_j\|_2^2}$

Specialization of SCD to $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$:

Randomized Coordinate Descent

- Choose number of steps T.
- Let $\mathbf{x}^{(1)} = \mathbf{0}$ and $\mathbf{r}^{(1)} = \mathbf{b}$.
- For $t = 1, \ldots, T$:
  - Pick random $j \in 1, \ldots, d$. Index j is selected with probability proportional to $\|\mathbf{a}_j\|_2^2 / \|\mathbf{A}\|_F^2$.
  - Set $c = \mathbf{a}_j^T \mathbf{r}^{(t)} / \|\mathbf{a}_j\|_2^2$
  - $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - c\mathbf{e}_j$
  - $\mathbf{r}^{(t+1)} = \mathbf{r}^{(t)} - c\mathbf{a}_j$
- Return $\mathbf{x}^{(T)}$.

Claim

$$\mathbb{E}\|\mathbf{r}^{(t+1)}\|_2^2 = \|\mathbf{r}^{(t)}\|_2^2 - \frac{1}{\|\mathbf{A}\|_F^2}\|\mathbf{A}^T\mathbf{r}^{(t)}\|_2^2$$

Any residual $\mathbf{r}$ can be written as $\mathbf{r} = \mathbf{r}^* + \bar{\mathbf{r}}$ where $\mathbf{r}^* = \mathbf{A}\mathbf{x}^* - \mathbf{b}$ and $\bar{\mathbf{r}} = \mathbf{A}(\mathbf{x}^t - \mathbf{x}^*)$. Note that $\mathbf{A}^T\mathbf{r}^* = 0$ and $\bar{\mathbf{r}} \perp \mathbf{r}^*$.

Claim

$$\mathbb{E}\|\bar{\mathbf{r}}^{(t+1)}\|_2^2 \leq \|\bar{\mathbf{r}}^{(t)}\|_2^2 - \frac{\lambda_{\min}(\mathbf{A}^T\mathbf{A})}{\|\mathbf{A}\|_F^2}$$

$$\mathbb{E}\|\bar{\mathbf{r}}^{(t+1)}\|_2^2 + \|\mathbf{r}^*\|_2^2 \leq \|\bar{\mathbf{r}}^{(t)}\|_2^2 + \|\mathbf{r}^*\|_2^2 - \frac{1}{\|\mathbf{A}\|_F^2}\|\mathbf{A}^T\bar{\mathbf{r}}^{(t)}\|_2^2\|\bar{\mathbf{r}}^{(t)}\|_2^2$$

Exercise: Because $\bar{\mathbf{r}}$ is in the column span of $\mathbf{A}$,

$$\|\mathbf{A}^T\bar{\mathbf{r}}^{(t)}\|_2^2 \geq \lambda_{\min}(\mathbf{A}^T\mathbf{A})\|\bar{\mathbf{r}}^{(t)}\|_2^2$$

Theorem (Randomized Coordinate Descent convergence)

After T steps of RCD with importance sampling run on $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$, we have:

$$\mathbb{E}[f(\mathbf{x}^{(t)}) - f(\mathbf{x}^*)] \leq \left(1 - \frac{\lambda_{\min}(\mathbf{A}^T\mathbf{A})}{\|\mathbf{A}\|_F^2}\right)^t [f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)]$$

Corollary: After $T = O(\frac{\|\mathbf{A}\|_F^2}{\lambda_{\min}(\mathbf{A}^T\mathbf{A})} \log \frac{1}{\epsilon})$ we obtain error $\epsilon\|\mathbf{b}\|_2^2$.

Is this more or less iterations than the $T = O(\frac{\lambda_{\max}(\mathbf{A}^T\mathbf{A})}{\lambda_{\min}(\mathbf{A}^T\mathbf{A})} \log \frac{1}{\epsilon})$ required for gradient descent to converge?

Recall useful linear algebraic fact:

$$\|\mathbf{A}\|_F^2 = \mathrm{tr}(\mathbf{A}^T\mathbf{A}) = \sum_{i=1}^{d} \lambda_i(\mathbf{A}^T\mathbf{A})$$

$$\lambda_{\max}(\mathbf{A}^T\mathbf{A}) \leq \|\mathbf{A}\|_F^2 \leq d \cdot \lambda_{\max}(\mathbf{A}^T\mathbf{A})$$

For solving $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$,

$$(\#\text{ GD Iterations}) \leq (\#\text{ RCD Iterations}) \leq d \cdot (\#\text{ GD Iterations})$$

But RCD iterations are cheaper by a factor of d.

When does $\|A\|_F^2 = \text{tr}(A^T A) = d \cdot \lambda_{\max}(A^T A)$?

When does $\|A\|_F^2 = \text{tr}(A^T A) = 1 \cdot \lambda_{\max}(A^T A)$?

Roughly:

Stochastic Gradient Descent performs well when data points (rows) are repetitive.

Stochastic Coordinate Descent performs well when data features (columns) are repetitive.

NON-CONVEX OPTIMIZATION

STATIONARY POINTS

We understand much less about optimizing non-convex functions in comparison to convex functions, but not nothing. In many cases, we're still figuring out the right questions to ask

Definition (Stationary point)

For a differentiable function f, a stationary point is any $\mathbf{x}$ with:

$$\nabla f(\mathbf{x}) = \mathbf{0}$$

local/global minima - local/global maxima - saddle points

Reasonable goal: Find an approximate stationary point $\hat{\mathbf{x}}$ with

$$\|\nabla f(\hat{\mathbf{x}})\|_2^2 \leq \epsilon.$$

SMOOTHNESS FOR NON-CONVEX FUNTIONS

Definition

A differentiable (potentially non-convex) function f is β smooth if for all $\mathbf{x}, \mathbf{y}$,

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq \beta \|\mathbf{x} - \mathbf{y}\|_2$$

Corollary: For all $\mathbf{x}, \mathbf{y}$

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

GRADIENT DESCENT FINDS APPROXIMATE STATIONARY POINTS

Theorem

If GD is run with step size $\eta = \frac{1}{\beta}$ on a differentiable function f with global minimum $\mathbf{x}^*$ then after $T = O(\frac{\beta[f(\mathbf{x}^{(1)}) - f(\mathbf{x}^*)]}{\epsilon})$ we will find an ϵ-approximate stationary point $\hat{\mathbf{x}}$.

- $\nabla f(\mathbf{x}^{(t)})^T(\mathbf{x}^{(t)} - \mathbf{x}^{(t+1)}) - f(\mathbf{x}^{(t)}) + f(\mathbf{x}^{(t+1)}) \leq \frac{\beta}{2}\|\mathbf{x}^{(t)} - \mathbf{x}^{(t+1)}\|_2^2$.
- $f(\mathbf{x}^{(t+1])}) - f(\mathbf{x}^{(t)}) \leq \frac{\beta}{2}\eta^2\|\nabla f(\mathbf{x}^{(t)})\|_2^2 - \eta\|\nabla f(\mathbf{x}^{(t)})\|_2^2$
- $f(\mathbf{x}^{(t+1])}) - f(\mathbf{x}^{(t)}) \leq \frac{-\eta}{2}\|\nabla f(\mathbf{x}^{(t)})\|_2^2$
- $\frac{1}{T}\sum_{t=1}^{T}\frac{\eta}{2}\|f(\mathbf{x}^{(t)})\|_2^2 \leq \frac{1}{T}\sum_{t=1}^{T} f(\mathbf{x}^{(t)}) - f(\mathbf{x}^{(t+1)})$
- $\frac{\eta}{2}\min_t \|f(\mathbf{x}^{(t)})\|_2^2 \leq \frac{1}{T}\left[f(\mathbf{x})^{(1)} - f(\mathbf{x})^{(T)}\right]$

Questions in non-convex optimization

If GD can find a stationary point, are there algorithms which find a stationary point faster using preconditioning, acceleration, stocastic methods, etc.?

What if my function only has global minima and stationary points? Randomized methods (SGD, perturbed gradient methods, etc.) can "escape" stationary points under some minor assumptions.

Example: $\min_x \frac{-x^T A^T A x}{x^T x}$

- **Global minimum**: Top eigenvector of $A^T A$ (i.e., top principal component of A).
- **Stationary points:** All other eigenvectors of A.

Useful for lots of other matrix factorization problems beyond vanilla PCA.