

CS-GY 6763: Lecture 6

Online and Stochastic Gradient Descent

NYU Tandon School of Engineering, Prof. Christopher Musco

- Midterm in class next Tuesday. .
 - See Ed post.
 - If you have permission to take remotely, please email asap so I know who you are.
 - List of topics covered and practice problems are on the course webpage.
- Tomorrow in the reading group **Hayden Edelson** will present **Estimating Sizes of Social Networks via Biased Sampling**. See you there!
- Thanks to **Robert Ronan** for the presentation last week.

First Order Optimization: Given a function f and a constraint set \mathcal{S} , assume we have:

- **Function oracle:** Evaluate $f(\mathbf{x})$ for any \mathbf{x} .
- **Gradient oracle:** Evaluate $\nabla f(\mathbf{x})$ for any \mathbf{x} .
- **Projection oracle:** Evaluate $P_{\mathcal{S}}(\mathbf{x})$ for any \mathbf{x} .

Goal: Find $\hat{\mathbf{x}} \in \mathcal{S}$ such that $f(\hat{\mathbf{x}}) \leq \min_{\mathbf{x} \in \mathcal{S}} f(\mathbf{x}) + \epsilon$.

Projected gradient descent:

- Select starting point $\mathbf{x}^{(0)}$, learning rate η .

- For $i = 0, \dots, T$:

$$\chi^{(i+1)} \Leftarrow \mathbf{z} = \mathbf{x}^{(i)} - \underline{\eta \nabla f(\mathbf{x}^{(i)})}$$

- $\mathbf{x}^{(i+1)} = P_{\mathcal{S}}(\mathbf{z})$

- Return $\hat{\mathbf{x}} = \arg \min_i f(\mathbf{x}^{(i)})$.

Conditions for convergence:

- **Convexity:** f is a convex function, \mathcal{S} is a convex set.
- **Bounded initial distance:**

$$\|\underline{\mathbf{x}}^{(0)} - \underline{\mathbf{x}}^*\|_2 \leq R$$

- **Bounded gradients (Lipschitz function):**

$$\|\underline{\nabla f(\mathbf{x})}\|_2 \leq G \text{ for all } \mathbf{x} \in \mathcal{S}.$$

Theorem: Projected Gradient Descent returns $\hat{\mathbf{x}}$ with $f(\hat{\mathbf{x}}) \leq \min_{\mathbf{x} \in \mathcal{S}} f(\mathbf{x}) + \epsilon$ after

$$T = \frac{R^2 G^2}{\epsilon^2}$$

iterations.

OTHER CONVERGENCE GUARANTEES

Convexity:

$$0 \leq [f(y) - f(x)] - \nabla f(x)^T(y - x) \leq$$

α -strong-convexity and β -smoothness: $\alpha \leq f''(x) \leq \beta$

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

Number of iterations for ϵ error:

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

CONVERGENCE GUARANTEE

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:

$$\|x^{(T)} - x^*\|_2^2 \leq e^{-(T\eta)\frac{\alpha}{\beta}} \|x^{(0)} - x^*\|_2^2$$

$T = \frac{\beta}{\alpha} \cdot \log(1/\epsilon)$

$$\leq \epsilon \|x^{(0)} - x^*\|_2^2$$

Corollary: If $T = O\left(\frac{\beta}{\alpha} \log(R\beta/\epsilon)\right)$ we have:

$$f(x^{(T)}) - f(x^*) \leq \epsilon$$

We will prove this in the special case of

$$f(x) = \|Ax - b\|_2^2$$

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

THE LINEAR ALGEBRA OF CONDITIONING

Let f be a twice differentiable function from $\mathbb{R}^d \rightarrow \mathbb{R}$. Let the **Hessian** $\mathbf{H} = \nabla^2 f(\mathbf{x})$ contain all of its second derivatives at a point \mathbf{x} . So $\mathbf{H} \in \mathbb{R}^{d \times d}$. We have:

$$H_{i,j} = [\nabla^2 f(\mathbf{x})]_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

For vector \mathbf{x}, \mathbf{v} :

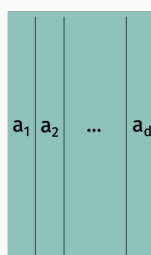
$$\nabla f(\mathbf{x} + t\mathbf{v}) \approx \nabla f(\mathbf{x}) + t [\nabla^2 f(\mathbf{x})] \mathbf{v}.$$

THE LINEAR ALGEBRA OF CONDITIONING

Let f be a twice differentiable function from $\mathbb{R}^d \rightarrow \mathbb{R}$. Let the **Hessian** $\mathbf{H} = \nabla^2 f(\mathbf{x})$ contain all of its second derivatives at a point \mathbf{x} . So $\mathbf{H} \in \mathbb{R}^{d \times d}$. We have:

$$H_{i,j} = [\nabla^2 f(\mathbf{x})]_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

Example: Let $f(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|_2^2$. Recall that $\nabla f(\mathbf{x}) = 2\mathbf{A}^T(\mathbf{Ax} - \mathbf{b})$.



A



x



b

$$\mathbf{H} = 2\mathbf{A}^T\mathbf{A}$$

HESSIAN MATRICES AND POSITIVE SEMIDEFINITENESS

Claim: If f is twice differentiable, then it is convex if and only if the matrix $\mathbf{H} = \nabla^2 f(\mathbf{x})$ is positive semidefinite for all \mathbf{x} .

Definition (Positive Semidefinite (PSD))

A square, symmetric matrix $\mathbf{H} \in \mathbb{R}^{d \times d}$ is positive semidefinite (PSD) for any vector $\mathbf{y} \in \mathbb{R}^d$, $\mathbf{y}^T \mathbf{H} \mathbf{y} \geq 0$.

$$\begin{bmatrix} 1 & 0 \\ 0 & -10 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = -10$$

This is a natural notion of “positivity” for symmetric matrices.

To denote that \mathbf{H} is PSD we will typically use “Loewner order” notation (`\succeq` in LaTeX):

$$\mathbf{y}^T \mathbf{H} \mathbf{y} = \mathbf{y}^T \mathbf{H}^{1/2} \mathbf{H}^{1/2} \mathbf{y} = \|\mathbf{H}^{1/2} \mathbf{y}\|^2 \quad \mathbf{H} \succeq 0. \quad \mathbf{y}^T \begin{bmatrix} 1 & 0 \\ 0 & 5 \\ 0 & 0 & 100 \end{bmatrix} \mathbf{y} = \begin{bmatrix} \sqrt{1} & & \\ & \sqrt{5} & \\ & & \sqrt{100} \end{bmatrix} \begin{bmatrix} \sqrt{1} & & \\ & \sqrt{5} & \\ & & \sqrt{100} \end{bmatrix}$$

We write $\mathbf{B} \succeq \mathbf{A}$ or equivalently $\mathbf{A} \preceq \mathbf{B}$ to denote that $(\mathbf{B} - \mathbf{A})$ is positive semidefinite. This gives a partial ordering on matrices.

$$\mathbf{B} - \mathbf{A} \succeq 0$$

HESSIAN MATRICES AND POSITIVE SEMIDEFINITENESS

Claim: If f is twice differentiable, then it is convex if and only if the matrix $\mathbf{H} = \nabla^2 f(\mathbf{x})$ is positive semidefinite for all \mathbf{x} .

Definition (Positive Semidefinite (PSD))

A square, symmetric matrix $\mathbf{H} \in \mathbb{R}^{d \times d}$ is positive semidefinite (PSD) for any vector $\mathbf{y} \in \mathbb{R}^d$, $\mathbf{y}^T \mathbf{H} \mathbf{y} \geq 0$.

For the least squares regression loss function: $f(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|_2^2$, $\mathbf{H} = \nabla^2 f(\mathbf{x}) = \mathbf{2A}^T \mathbf{A}$ for all \mathbf{x} .

We know that H is PSD because:

$$\mathbf{x}^T \mathbf{H} \mathbf{x} = \mathbf{x}^T \mathbf{2A}^T \mathbf{A} \mathbf{x} = 2 \|\mathbf{Ax}\|_2^2 \geq 0.$$

$$\mathbf{x}^T \mathbf{A}^T = (\mathbf{Ax})^T \quad \text{and} \quad (\mathbf{Ax})^T (\mathbf{Ax}) = \|\mathbf{Ax}\|_2^2$$

THE LINEAR ALGEBRA OF CONDITIONING

If f is β -smooth and α -strongly convex then at any point \mathbf{x} , $\mathbf{H} = \nabla^2 f(\mathbf{x})$ satisfies:

$$\alpha \mathbf{I}_{d \times d} \preceq \mathbf{H} \preceq \beta \mathbf{I}_{d \times d},$$

where $\mathbf{I}_{d \times d}$ is a $d \times d$ identity matrix.

This is the natural matrix generalization of the statement for scalar valued functions:

$$\alpha \leq f''(x) \leq \beta.$$

$$\alpha \leq f''(x) \leq \beta$$

$$\beta \begin{bmatrix} 1 & & \\ & 1 & \\ 0 & & 1 \end{bmatrix}$$

$$\alpha \begin{bmatrix} 1 & & \\ & 1 & \\ 0 & & 1 \end{bmatrix}$$

SMOOTH AND STRONGLY CONVEX HESSIAN

$$\underline{\underline{H - \alpha I \succcurlyeq 0}}$$

$$\begin{bmatrix} \lambda_{d_1} - \alpha & 0 \\ 0 & \lambda_{d_2} - \alpha \end{bmatrix}$$

$$\underline{\underline{\alpha I_{d \times d} \preceq H \preceq \beta I_{d \times d}}}.$$

Equivalently for any z ,

$$\alpha \|z\|_2^2 \leq \underbrace{z^T H z}_{\text{}} \leq \beta \|z\|_2^2.$$

$$\underline{\underline{B \cdot I - H \succcurlyeq 0}}$$

$$\text{for all } z, \quad z^T (B \cdot I - H) z \geq 0$$

$$B z^T I z - z^T H z \geq 0$$

$$B \|z\|_2^2 - z^T H z \geq 0$$

SIMPLE EXAMPLE

Let $f(\mathbf{x}) = \|\mathbf{D}\mathbf{x} - \mathbf{b}\|_2^2$ where \mathbf{D} is a diagonal matrix. For now imagine we're in two dimensions: $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $\mathbf{D} = \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix}$.

What are α, β for this problem?

$$\underline{\alpha \|\mathbf{z}\|_2^2 \leq \mathbf{z}^T \mathbf{H} \mathbf{z} \leq \beta \|\mathbf{z}\|_2^2} \quad \text{for } \underline{\|\mathbf{z}\|_2^2}$$

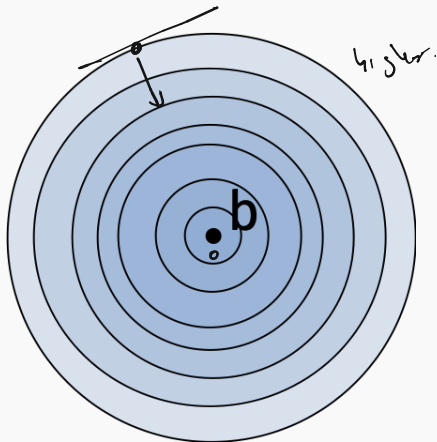
$$\mathbf{H} = 2\mathbf{D}^T \mathbf{D} = \begin{bmatrix} 2d_1^2 & \\ & 2d_2^2 \end{bmatrix}$$

$$\frac{2d_1^2 \cdot z_1^2 + 2d_2^2 \cdot z_2^2}{z_1^2 + z_2^2}$$

$$\alpha = \min(2d_1^2, 2d_2^2) \quad \beta = \max(2d_1^2, 2d_2^2)$$

GEOMETRIC VIEW

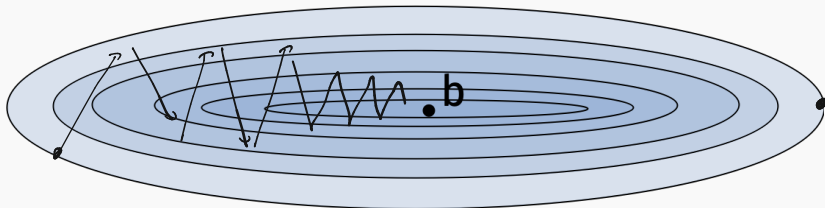
$$\alpha = b = 1$$



Level sets of $\|\mathbf{D}\mathbf{x} - \mathbf{b}\|_2^2$ when $d_1^2 = 1, d_2^2 = 1$.

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

GEOMETRIC VIEW



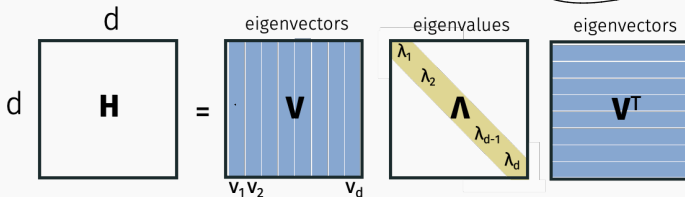
Level sets of $\|\underline{D}\mathbf{x} - \mathbf{b}\|_2^2$ when $d_1^2 = \frac{1}{3}, d_2^2 = 2$.

EIGENDECOMPOSITION VIEW

Any symmetric matrix \mathbf{H} has an orthogonal, real valued eigendecomposition.

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$$

$$\mathbf{H}\mathbf{v} = \lambda\mathbf{v}$$



Here \mathbf{V} is square and orthogonal, so $\mathbf{V}^T\mathbf{V} = \mathbf{V}\mathbf{V}^T = \mathbf{I}$. And for each \mathbf{v}_i , we have:

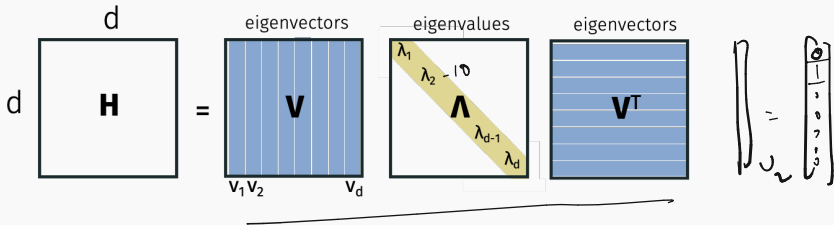
$$\mathbf{H}\mathbf{v}_i = \lambda_i\mathbf{v}_i.$$

By definition, that's what makes $\mathbf{v}_1, \dots, \mathbf{v}_d$ eigenvectors.

EIGENDECOMPOSITION VIEW

Recall $\mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}$.

$$\mathbf{v}_i^T \mathbf{H} \mathbf{v}_i = \mathbf{e}_i^T \mathbf{\Lambda} \mathbf{e}_i = \lambda_i$$



Claim: \mathbf{H} is PSD $\Leftrightarrow \lambda_1, \dots, \lambda_d \geq 0$.

$$\mathbf{H} = \mathbf{A}^* \mathbf{A}$$

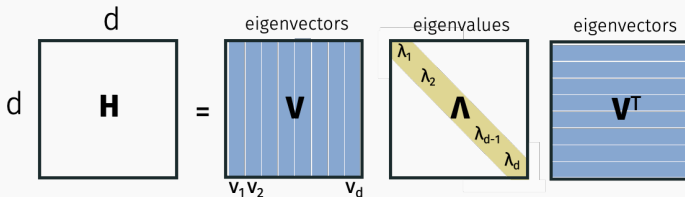
$$\mathbf{\Lambda}^{1/2} = \begin{bmatrix} \lambda_1^{1/2} & & 0 \\ & \lambda_2^{1/2} & \\ 0 & & \lambda_d^{1/2} \end{bmatrix}$$

$$\mathbf{A} = \mathbf{\Lambda}^{1/2} \mathbf{V}^T$$

$$\mathbf{V} \mathbf{\Lambda}^{1/2} \mathbf{\Lambda}^{1/2} \mathbf{V}^T = \mathbf{H}$$

EIGENDECOMPOSITION VIEW

Recall $\mathbf{V}\mathbf{V}^T$ = $\mathbf{V}^T\mathbf{V} = \mathbf{I}$.



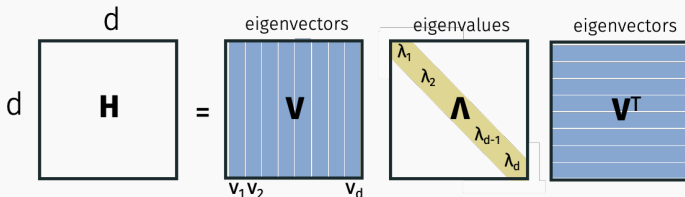
Claim: $\alpha \mathbf{I} \preceq \mathbf{H} \preceq \beta \mathbf{I}$ $\Leftrightarrow \alpha \leq \lambda_d \leq \dots \leq \lambda_1 \leq \beta$.

$$\mathbf{H} - \alpha \mathbf{I} \succeq 0$$

$$\mathbf{V} \mathbf{\Lambda} \mathbf{V}^T - \alpha \mathbf{V} \mathbf{V}^T = \mathbf{V} \underbrace{(\mathbf{\Lambda} - \alpha \mathbf{I})}_{\succeq 0} \mathbf{V}^T$$

EIGENDECOMPOSITION VIEW

Recall $\mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}$.



In other words, if we let $\lambda_{\max}(\mathbf{H})$ and $\lambda_{\min}(\mathbf{H})$ be the smallest and largest eigenvalues of \mathbf{H} , then for all \mathbf{z} we have:

$$\mathbf{z}^T \mathbf{H} \mathbf{z} \leq \lambda_{\max}(\mathbf{H}) \cdot \|\mathbf{z}\|^2$$

$$\mathbf{z}^T \mathbf{H} \mathbf{z} \geq \lambda_{\min}(\mathbf{H}) \cdot \|\mathbf{z}\|^2$$

If the maximum eigenvalue of $\mathbf{H} = \nabla^2 f(\mathbf{x}) = \beta$ and the minimum eigenvalue of $\mathbf{H} = \nabla^2 f(\mathbf{x}) = \alpha$ then $f(\mathbf{x})$ is β -smooth and α -strongly convex.

$$\begin{aligned}\underline{\lambda_{\max}(\mathbf{H})} &= \underline{\beta} \\ \underline{\lambda_{\min}(\mathbf{H})} &= \underline{\alpha}\end{aligned}$$

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:

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

$$H = 2A^T A$$

Goal: Prove for $f(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|_2^2$.

Let $\lambda_{\max} = \lambda_{\max}(\mathbf{A}^T \mathbf{A})$. Gradient descent update is:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \frac{1}{2\lambda_{\max}} 2\mathbf{A}^T (\mathbf{Ax}^{(t)} - \mathbf{b})$$

ALTERNATIVE VIEW OF GRADIENT DESCENT

$$A^T(Ax^* - b) = 0 \quad A^T A x^* = A^T b$$

$$A^T b = A^T A x^*$$

Richardson Iteration view:

$$\underline{(x^{(t+1)} - x^*)} = \left(I - \frac{1}{\lambda_{\max}} A^T A \right) \underline{(x^{(t)} - x^*)}$$

$x^T H x$

$$x^{(t+1)} = x^{(t)} - \frac{1}{\lambda_{\max}} A^T (A x^{(t)} - b)$$

$$x^{(t+1)} - x^* = x^{(t)} - \frac{1}{\lambda_{\max}} A^T (A x^{(t)} - b) - x^*$$

$$x^{(t)} - \frac{1}{\lambda_{\max}} A^T A x^{(t)} - \frac{1}{\lambda_{\max}} A^T A x^* - x^*$$

$$A^T A = V \Lambda V^T$$

$$= \left(I - \frac{1}{\lambda_{\max}} A^T A \right) (x^{(t)} - x^*)$$

$$\rightarrow 1 - \frac{\lambda_{\min}}{\lambda_{\max}}$$

What is the maximum eigenvalue of the symmetric matrix

$$\left(I - \frac{1}{\lambda_{\max}} A^T A \right)$$

in terms of the eigenvalues $V V^T - \frac{1}{\lambda_{\max}} V \Lambda V^T$

$$\lambda_{\max} = \underline{\lambda_1} \geq \dots \geq \underline{\lambda_d} = \underline{\lambda_{\min}} \text{ of } A^T A$$

$$V \left(I - \frac{1}{\lambda_{\max}} \Lambda \right) V^T$$

UNROLLED GRADIENT DESCENT

$$\left(I - \frac{1}{\lambda_{\max}} A^T A\right) \left(I - \frac{1}{\lambda_{\max}} A^T A\right) \dots \left(I - \frac{1}{\lambda_{\max}} A^T A\right)$$

$$\max_z \frac{z^T M z}{\|z\|^2} = 1$$

$$(x^{(T+1)} - x^*) = \left(I - \frac{1}{\lambda_{\max}} A^T A \right)^T (x^{(1)} - x^*)$$

$$\min_z \frac{z^T M z}{\|z\|^2}$$

$$= \lambda_{\min}(M)$$

$$\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}$$

Approach: Show that the maximum eigenvalue of $\left(I - \frac{1}{\lambda_{\max}} A^T A\right)^T$ is small – i.e., bounded by $e^{-T/\kappa}$.

$$z^T M z \leq \lambda_{\max}(M) \cdot \|z\|^2$$

Conclusion:

$$\|x^{(T+1)} - x^*\|_2^2 = (x^{(1)} - x^*)^T \left(I - \frac{1}{\lambda_{\max}} A^T A \right)^{2T} (x^{(1)} - x^*)$$

$$\text{Since } \lambda_{\max}(M) = \max_z \frac{z^T M z}{\|z\|_2^2}, \text{ we have:}$$

$$\|x^{(T+1)} - x^*\|_2^2 \leq \lambda_{\max} \left(\left(I - \frac{1}{\lambda_{\max}} A^T A \right)^{2T} \right) \|x^{(1)} - x^*\|_2^2$$

So we have $\|x^{(T+1)} - x^*\|_2 \leq$

UNROLLED GRADIENT DESCENT

$$(x^{(T+1)} - x^*) = \left(I - \frac{1}{\lambda_{\max}} A^T A \right)^T (x^{(1)} - x^*)$$

$$W = \left(I - \frac{1}{\lambda_{\max}} A^T A \right) = U C U^T$$

$$W \cdot W \cdot \dots \cdot W = U \cancel{C} \cancel{U^T} U \cancel{C} \cancel{U^T} \dots U \cancel{C} \cancel{U^T} = U \underline{C^{2T}} U^T$$

What is the maximum eigenvalue of the symmetric matrix

$$\left(I - \frac{1}{\lambda_{\max}} A^T A \right)^{2T} ? = \left(\left(1 - \frac{\lambda_{\min}}{\lambda_{\max}} \right)^k \right)^{2T/k} = \left(\frac{1}{e} \right)^{2T/k} \leq e^{-2T/k}$$

$$\lambda_{\max} \left(\left(I - \frac{1}{\lambda_{\max}} A^T A \right) \right) = \underline{\underline{1 - \frac{\lambda_{\min}}{\lambda_{\max}}}}$$

$$\text{where } \lambda_{\min} = \lambda_{\min}(A^T A)$$

$$\lambda_{\max} = \lambda_{\max}(A^T A)$$

How large does T need to be so $\left(1 - \frac{\lambda_{\min}}{\lambda_{\max}} \right)^{2T} \leq \epsilon$.

$$T = O\left(\frac{\lambda_{\max}}{\lambda_{\min}} \log(1/\epsilon) \right)$$

ACCELERATION

ACCELERATED GRADIENT DESCENT

Nesterov's accelerated gradient descent:

- $\mathbf{x}^{(1)} = \mathbf{y}^{(1)} = \mathbf{z}^{(1)}$

- For $t = 1, \dots, T$

- $\mathbf{y}^{(t+1)} = \mathbf{x}^{(t)} - \frac{1}{\beta} \nabla f(\mathbf{x}^{(t)})$

- $\mathbf{x}^{(t+1)} = \left(1 + \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right) \mathbf{y}^{(t+1)} + \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} (\mathbf{y}^{(t+1)} - \mathbf{y}^{(t)})$

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

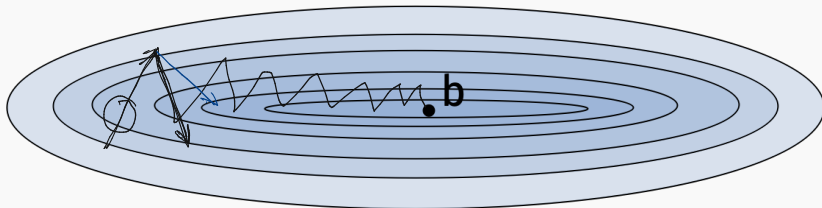
Theorem (AGD for β -smooth, α -strongly convex.)

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 $T = O(\sqrt{\kappa} \log(\kappa/\epsilon))$ achieve error ϵ .

INTUITION BEHIND ACCELERATION



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

Other terms for similar ideas:

- Momentum
- Heavy-ball methods

What if we look back beyond two iterates?

BREAK

Second part of class:

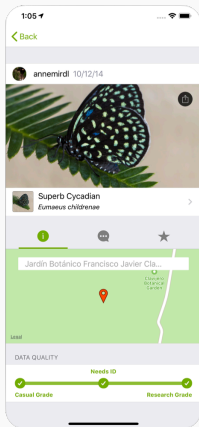
- Basics of Online Learning + Optimization.
- Introduction to Regret Analysis.
- Application to analyzing Stochastic Gradient Descent.

Many machine learning problems are solved in an online setting with constantly changing data.

- Spam filters are incrementally updated and adapt as they see more examples of spam over time.
- Image classification systems learn from mistakes over time (often based on user feedback).
- Content recommendation systems adapt to user behavior and clicks (which may not be a good thing...)

Plant identification via iNaturalist app.

(California Academy of Science + National Geographic)



- When the app fails, image is classified via crowdsourcing (backed by huge network of amateurs and experts).
- Single model that is updated constantly, not retrained in batches.

EXAMPLE

ML based email spam/scam filtering.

```
MIME-Version: 1.0 Date: Mon, 7 Oct 2019
14:51:30 -0400 Message-ID: <CANV7LzUgXw==B-
39MLAN0Py19_jaaK60gmR04QCFB9y@uadmail-spa
11.com> Subject: 92231 Reading Group, Meeting
2, tomorrow at 10am From: Christopher Musco
<cmusco@nyu.edu> To: algalids@nyu.edu Content-
Type: multipart/alternative;
boundary="0000000000078ec240594568a53" --
0000000000078ec240594568a53 Content-Type:
text/plain; charset="UTF-8" I hope everyone
had a good weekend! Tomorrow at *10am in 370
Jay St. #1114* we will meet for the second
instantiation of the CS-CV 92231 reading
group. Nick Feng will be leading a discussion
about the paper Simple Analyses of the Sparse
Johnson-Lindenstrauss Transform
<http://drops.dagstuhl.de/opus/volltexte/2018
/8305/pdf/OASiCS-SCMA-2018-15.pdf>. Please
read the abstract and introduction before the
meeting. Best, - CM *Christopher Musco,
Assistant Professor* *New York University,
Tandon School of Engineering* *4011 576
2541* --0000000000078ec240594568a53 Content-
Type: text/html; charset="UTF-8" Content-
Transfer-Encoding: quoted-printable
```

sender on graylist 1

sender on graylist 2

Bag-of-words

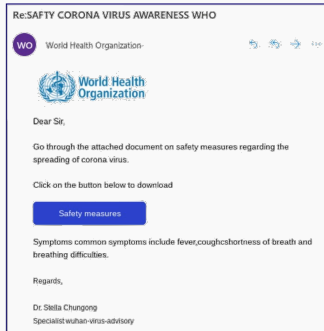
sender flagged by x users

sender ip flagged by x users

1
0
⋮
0
0
1
1
0
1

Markers for spam change overtime, so model might change.

ML based email spam/scam filtering.



Markers for spam change overtime, so model might change.

Choose some model $M_{\mathbf{x}}$ parameterized by parameters \mathbf{x} and some loss function ℓ . At time steps $1, \dots, T$, receive data vectors $\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(T)}$.

- At each time step, we pick (“play”) a parameter vector $\mathbf{x}^{(i)}$.
- Make prediction $\tilde{y}^{(i)} = M_{\mathbf{x}^{(i)}}(\mathbf{a}^{(i)})$.
- Then told true value or label $y^{(i)}$.
- Goal is to minimize cumulative loss:

$$y^{(i)} - M_{\mathbf{x}^{(i)}}(\mathbf{a}^{(i)})$$

$$L = \sum_{i=1}^T \ell(\mathbf{x}^{(i)}, \mathbf{a}^{(i)}, y^{(i)})$$

For example, for a regression problem we might use the ℓ_2 loss:

$$\ell(\mathbf{x}^{(i)}, \mathbf{a}^{(i)}, y^{(i)}) = \left| \langle \mathbf{x}^{(i)}, \mathbf{a}^{(i)} \rangle - y^{(i)} \right|^2.$$

For classification, we could use logistic/cross-entropy loss.

Abstraction as optimization problem: Instead of a single objective function f , we have a single (initially unknown) function $\underline{f_1}, \dots, \underline{f_T} : \mathbb{R}^d \rightarrow \mathbb{R}$ for each time step.

- For time step $i \in 1, \dots, T$, select vector $\mathbf{x}^{(i)}$.
- Observe f_i and pay cost $f_i(\mathbf{x}^{(i)})$
- Goal is to minimize $\sum_{i=1}^T f_i(\mathbf{x}^{(i)})$.

We make no assumptions that f_1, \dots, f_T are related to each other at all!

REGRET BOUND

In offline optimization, we wanted to find $\hat{\mathbf{x}}$ satisfying $f(\hat{\mathbf{x}}) \leq \min_{\mathbf{x}} f(\mathbf{x})$. Ask for a similar thing here.

Objective: Choose $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}$ so that:

$$\sum_{i=1}^T f_i(\mathbf{x}^{(i)}) \leq \left[\min_{\mathbf{x}} \sum_{i=1}^T f_i(\mathbf{x}) \right] + \epsilon.$$

Here ϵ is called the regret of our solution sequence $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}$.

Regret compares to the best fixed solution in hindsight.

$$\sum_{i=1}^T f_i(\mathbf{x}^{(i)}) \leq \left[\min_{\mathbf{x}} \sum_{i=1}^T f_i(\mathbf{x}) \right] + \epsilon.$$

It's very possible that $\sum_{i=1}^T f_i(\mathbf{x}^{(i)}) < \left[\min_{\mathbf{x}} \sum_{i=1}^T f_i(\mathbf{x}) \right]$. Could we hope for something stronger?

Exercise: Argue that the following is impossible to achieve:

$$\sum_{i=1}^T f_i(\mathbf{x}^{(i)}) \leq \left[\sum_{i=1}^T \min_{\mathbf{x}} f_i(\mathbf{x}) \right] + \epsilon.$$

HARD EXAMPLE FOR ONLINE OPTIMIZATION

Convex functions:

$$\begin{aligned} f_1(x) &= |x - h_1| &= |x| \\ &\vdots &= |x - 1| \\ f_T(x) &= |x - h_T| &\begin{matrix} |x| \\ |x| \\ |x-1| \\ \vdots \\ \vdots \end{matrix} \end{aligned}$$

where h_1, \dots, h_T are i.i.d. uniform $\{0, 1\}$.

$$\sum_{i=1}^T f_i(x_i) \approx T/2$$

$$\min_x \sum_{i=1}^T f_i(x) \approx T/2$$

... 2)

$$\sum_{i=1}^T f_i(\mathbf{x}^{(i)}) \leq \left[\min_{\mathbf{x}} \sum_{i=1}^T f_i(\mathbf{x}) \right] + \epsilon.$$

Beautiful balance:

- Either f_1, \dots, f_T are similar, so we can learn predict f_i from earlier functions.
- Or f_1, \dots, f_T are very different, in which case $\min_{\mathbf{x}} \sum_{i=1}^T f_i(\mathbf{x})$ is large, so regret bound is easy to achieve.
- Or we live somewhere in the middle.

Online Gradient descent:

- Choose $\mathbf{x}^{(1)}$ and $\eta = \frac{R}{G\sqrt{T}}$.
- For $i = 1, \dots, T$:
 - Play $\mathbf{x}^{(i)}$.
 - Observe f_i and incur cost $f_i(\mathbf{x}^{(i)})$.
 - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla f_i(\mathbf{x}^{(i)})$

If $f_1, \dots, f_T = f$ are all the same, this looks a lot like regular gradient descent. We update parameters using the gradient ∇f at each step.

ONLINE GRADIENT DESCENT (OGD)

$$\underline{\mathbf{x}}^* = \arg \min_{\mathbf{x}} \sum_{i=1}^T \underline{f_i(\mathbf{x})} \text{ (the offline optimum)}$$

Assume:

- f_1, \dots, f_T are all convex.
- Each is G -Lipschitz: for all \mathbf{x}, i , $\|\underline{\nabla f_i(\mathbf{x})}\|_2 \leq G$.
- Starting radius: $\|\underline{\mathbf{x}}^* - \underline{\mathbf{x}}^{(1)}\|_2 \leq R$.

Online Gradient descent:

- Choose $\mathbf{x}^{(1)}$ and $\eta = \frac{R}{G\sqrt{T}}$.
- For $i = 1, \dots, T$:
 - Play $\mathbf{x}^{(i)}$.
 - Observe f_i and incur cost $f_i(\mathbf{x}^{(i)})$.
 - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla f_i(\mathbf{x}^{(i)})$

Let $\mathbf{x}^* = \arg \min_{\mathbf{x}} \sum_{i=1}^T f_i(\mathbf{x})$ (the offline optimum)

Theorem (OGD Regret Bound)

After T steps, $\epsilon = \left[\sum_{i=1}^T f_i(\mathbf{x}^{(i)}) \right] - \left[\sum_{i=1}^T f_i(\mathbf{x}^*) \right] \leq \underline{RG\sqrt{T}}$.

Average regret overtime is bounded by $\frac{\epsilon}{T} \leq \frac{RG}{\sqrt{T}}$.

Goes $\rightarrow 0$ as $T \rightarrow \infty$.

All this with no assumptions on how f_1, \dots, f_T relate to each other! They could have even been chosen **adversarially** – e.g. with f_i depending on our choice of \mathbf{x}_i and all previous choices.

Theorem (OGD Regret Bound)

After T steps, $\epsilon = \left[\sum_{i=1}^T f_i(\mathbf{x}^{(i)}) \right] - \left[\sum_{i=1}^T f_i(\mathbf{x}^*) \right] \leq RG\sqrt{T}$.

Claim 1: For all $i = 1, \dots, T$,

$$f_i(\mathbf{x}^{(i)}) - f_i(\mathbf{x}^*) \leq \frac{\|\mathbf{x}^{(i)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(i+1)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

(Same proof as last class. Only uses convexity of f_i .)

$$\begin{aligned} \|\mathbf{x}^{(i+1)} - \mathbf{x}^*\|_2^2 &= \|\mathbf{x}^{(i)} - \eta \nabla f_i(\mathbf{x}^{(i)}) - \mathbf{x}^*\|_2^2 \\ &= \|\mathbf{x}^{(i)} - \mathbf{x}^*\|_2^2 + \|\eta \nabla f_i(\mathbf{x}^{(i)})\|_2^2 - \underbrace{2\eta \nabla f_i(\mathbf{x}^{(i)})^\top (\mathbf{x}^{(i)} - \mathbf{x}^*)}_{\leq \eta^2 G^2} \\ &\leq \eta^2 G^2 \end{aligned}$$

Theorem (OGD Regret Bound)

After T steps, $\epsilon = \left[\sum_{i=1}^T f_i(\mathbf{x}^{(i)}) \right] - \left[\sum_{i=1}^T f_i(\mathbf{x}^*) \right] \leq RG\sqrt{T}$.

Claim 1: For all $i = 1, \dots, T$,

$$f_i(\mathbf{x}^{(i)}) - f_i(\mathbf{x}^*) \leq \frac{\|\mathbf{x}^{(i)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(i+1)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$$

Telescoping Sum: $\left(\frac{\|\mathbf{x}^{(1)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(2)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2} \right) + \left(\frac{\|\mathbf{x}^{(2)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(3)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{\eta G^2}{2} \right) + \dots$

$$\sum_{i=1}^T [f_i(\mathbf{x}^{(i)}) - f_i(\mathbf{x}^*)] \leq \frac{\|\mathbf{x}^{(1)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(T)} - \mathbf{x}^*\|_2^2}{2\eta} + \frac{T\eta G^2}{2}$$

$$\leq \frac{R^2}{2\eta} + \frac{T\eta G^2}{2}$$

$$\eta : \frac{RG}{\sqrt{T}}$$

$$\|\mathbf{x}^{(1)} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}^{(T)} - \mathbf{x}^*\|_2^2 \leq \|\mathbf{x}^{(1)} - \mathbf{x}^*\|_2^2 \leq R^2$$

STOCHASTIC GRADIENT DESCENT (SGD)

Efficient offline optimization method for functions f with finite sum structure:

$$\underline{f(\mathbf{x})} = \sum_{i=1}^n \underline{f_i(\mathbf{x})}.$$

Goal is to find $\hat{\mathbf{x}}$ such that $f(\hat{\mathbf{x}}) \leq f(\mathbf{x}^*) + \epsilon$.

- The most widely use optimization algorithm in modern machine learning.
- Easily analyzed as a special case of online gradient descent!

Recall the machine learning setup. In empirical risk minimization, we can typically write:

$$f(\mathbf{x}) = \sum_{i=1}^n f_i(\mathbf{x})$$

where f_i is the loss function for a particular data example $(\mathbf{a}^{(i)}, y^{(i)})$.

$$\nabla f(\mathbf{x}) = \sum_{i=1}^n \nabla f_i(\mathbf{x})$$

Example: least squares linear regression.

$$f(\mathbf{x}) = \sum_{i=1}^n (\mathbf{x}^T \mathbf{a}^{(i)} - y^{(i)})^2$$

Note that by linearity, $\nabla f(\mathbf{x}) = \sum_{i=1}^n \nabla f_i(\mathbf{x})$.

Main idea: Use random approximate gradient in place of actual gradient.

Pick random $j \in 1, \dots, n$ and update \mathbf{x} using $\nabla f_j(\mathbf{x})$.

$$\mathbb{E} [\nabla f_j(\mathbf{x})] = \frac{1}{n} \nabla f(\mathbf{x}).$$

$n\nabla f_j(\mathbf{x})$ is an unbiased estimate for the true gradient $\nabla f(\mathbf{x})$, but can often be computed in a $1/n$ fraction of the time!

Trade slower convergence for cheaper iterations.

STOCHASTIC GRADIENT DESCENT

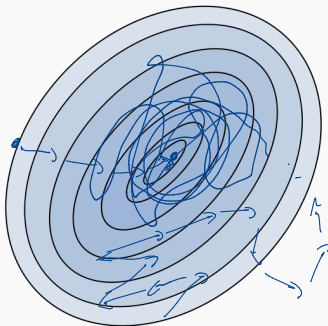
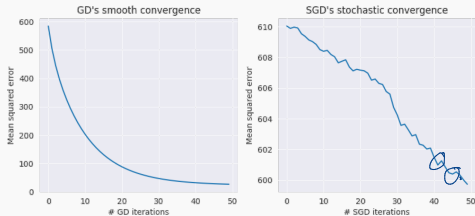
Stochastic first-order oracle for $\underline{f(\mathbf{x})} = \sum_{i=1}^n \underline{f_i(\mathbf{x})}$.

- **Function Query:** For any chosen j, \mathbf{x} , return $f_j(\mathbf{x})$
- **Gradient Query:** For any chosen j, \mathbf{x} , return $\underline{\nabla f_j(\mathbf{x})}$

Stochastic Gradient descent:


- Choose starting vector $\mathbf{x}^{(1)}$, learning rate η
- For $i = 1, \dots, T$:
 - Pick random $j_i \in 1, \dots, n$.
 - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla f_{j_i}(\mathbf{x}^{(i)})$
- Return $\hat{\mathbf{x}} = \frac{1}{T} \sum_{i=1}^T \mathbf{x}^{(i)}$

VISUALIZING SGD



STOCHASTIC GRADIENT DESCENT

Assume:

- Finite sum structure: $f(\mathbf{x}) = \sum_{i=1}^n f_i(\mathbf{x})$, with f_1, \dots, f_n all convex.
- Lipschitz functions: for all \mathbf{x}, j , $\|\nabla f_j(\mathbf{x})\|_2 \leq \frac{G'}{n}$. 
 - What does this imply about Lipschitz constant of f ?
- Starting radius: $\|\mathbf{x}^* - \mathbf{x}^{(1)}\|_2 \leq R$.

Stochastic Gradient descent:

- Choose $\mathbf{x}^{(1)}$, steps T , learning rate $\eta = \frac{D}{G'\sqrt{T}}$.
- For $i = 1, \dots, T$:
 - Pick random $j_i \in 1, \dots, n$.
 - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla f_{j_i}(\mathbf{x}^{(i)})$
- Return $\hat{\mathbf{x}} = \frac{1}{T} \sum_{i=1}^T \mathbf{x}^{(i)}$

Approach: View as online gradient descent run on function sequence f_{j_1}, \dots, f_{j_T} .

Only use the fact that step equals gradient in expectation.

Claim (SGD Convergence)

After $T = \frac{R^2 G^2}{\epsilon^2}$ iterations:

$$\hat{\mathbf{x}} = \frac{1}{T} \sum_{i=1}^T \mathbf{x}^{(i)}$$

$$\mathbb{E}[f(\hat{\mathbf{x}}) - f(\mathbf{x}^*)] \leq \epsilon.$$

Claim 1:

$$f\left(\frac{\mathbf{x}}{2} + \frac{\mathbf{y}}{2}\right) \leq \frac{1}{2} f(\mathbf{x}) + \frac{1}{2} f(\mathbf{y})$$

$$\underbrace{f(\hat{\mathbf{x}}) - f(\mathbf{x}^*)}_{\text{Claim 1}} \leq \frac{1}{T} \sum_{i=1}^T [f(\mathbf{x}^{(i)}) - f(\mathbf{x}^*)] = \left[\frac{1}{T} \sum_{i=1}^T f(\mathbf{x}^{(i)}) \right] - f(\mathbf{x}^*)$$

Prove using Jensen's Inequality:

$$f\left(\frac{1}{T} \sum_{i=1}^T \mathbf{x}^{(i)}\right) \leq \frac{1}{T} \sum_{i=1}^T f(\mathbf{x}^{(i)})$$

STOCHASTIC GRADIENT DESCENT ANALYSIS

Claim (SGD Convergence)

$j_i \sim \text{unif}(1, \dots, n)$

After $T = \frac{R^2 G'^2}{\epsilon^2}$ iterations:

$$\mathbb{E}[f(\hat{\mathbf{x}}) - f(\mathbf{x}^*)] \leq \epsilon.$$

$f(\mathbf{x}^{(i)})$

$$\mathbb{E}[f(\hat{\mathbf{x}}) - f(\mathbf{x}^*)] \leq \frac{1}{T} \sum_{i=1}^T \mathbb{E}[f(\mathbf{x}^{(i)}) - f(\mathbf{x}^*)]$$

$$= \sum_{j=1}^n f_j(\mathbf{x}^{(i)})$$

$$= \frac{1}{T} \sum_{i=1}^T n \mathbb{E}[f_{j_i}(\mathbf{x}^{(i)}) - f_{j_i}(\mathbf{x}^*)]$$

$$= \frac{1}{T} \sum_{i=1}^T n \mathbb{E}[f_{j_i}(\mathbf{x}^{(i)}) - f_{j_i}(\mathbf{x}^*)]$$

$\mathbf{x}^{\text{offline}} = \min_{\mathbf{x}} \sum_{i=1}^T f_{j_i}(\mathbf{x})$

$$\leq \frac{n}{T} \cdot \mathbb{E} \left[\sum_{i=1}^T f_{j_i}(\mathbf{x}^{(i)}) - f_{j_i}(\mathbf{x}^*) \right]$$

$$= \frac{R G'}{\sqrt{T}}$$

$$\leq \frac{n}{T} \cdot \left(R \cdot \frac{G'}{n} \cdot \sqrt{T} \right) \quad (\text{by OGD guarantee.})$$

STOCHASTIC VS. FULL BATCH GRADIENT DESCENT

Number of iterations for error ϵ :

- Gradient Descent: $T = \frac{R^2 G^2}{\epsilon^2}$

- Stochastic Gradient Descent: $T = \frac{R^2 G'^2}{\epsilon^2}$

Always have $\underline{G} \leq \underline{G'}$:

$$\max_x \|\nabla f(x)\|_2 \leq \max_x (\|\nabla f_1(x)\|_2 + \dots + \|\nabla f_n(x)\|_2) \leq n \cdot \frac{G'}{n} = G'$$

So GD converges strictly faster than SGD.

But for a fair comparison:

- SGD cost = (# of iterations) $\cdot \underline{O(1)}$

- GD cost = (# of iterations) $\cdot O(n)$

$$f(x) = \sum_{i=1}^n h_i(x)$$

$$\|\nabla f(x)\|_2 = \eta \cdot \|\nabla h(x)\|_2$$

$$\underline{G'} > \underline{G}$$

$$\begin{aligned} & \leq \max_x \|\nabla f_1(x)\|_2 \\ & + \max_x \|\nabla f_2(x)\|_2 \\ & + \dots \\ & + \max_x \|\nabla f_n(x)\|_2 \end{aligned}$$

STOCHASTIC VS. FULL BATCH GRADIENT DESCENT

We always have $G \leq G'$. When it is much smaller then GD will perform better. When it is closer to this upper bound, SGD will perform better.

What is an extreme case where $G = G'$?

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

$$O(\underbrace{nd \cdot K \log(1/\epsilon)} + nd)$$

STOCHASTIC VS. FULL BATCH GRADIENT DESCENT

What if each gradient $\nabla f_i(\mathbf{x})$ looks like random vectors in \mathbb{R}^d ?
E.g. with $\mathcal{N}(0, 1)$ entries?

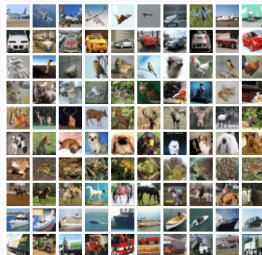
$$\mathbb{E} [\|\nabla f_i(\mathbf{x})\|_2^2] =$$

$$\mathbb{E} [\|\nabla f(\mathbf{x})\|_2^2] = \mathbb{E} \left[\left\| \sum_{i=1}^n \nabla f_i(\mathbf{x}) \right\|_2^2 \right] =$$

STOCHASTIC VS. FULL BATCH GRADIENT DESCENT

$$| \langle y, x_i \rangle - z_i |^2 \quad \alpha_i$$

Takeaway: SGD performs better when there is more structure or repetition in the data set.



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:

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

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

Many, many problem specific preconditioners are used in practice. Their 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!

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

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

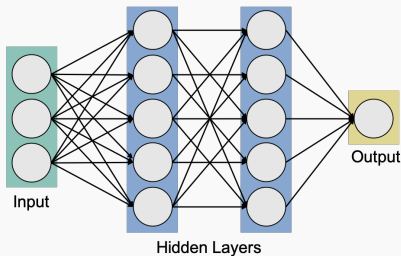
Gradient descent on g :

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

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:

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

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

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

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

Stochastic Coordinate Descent:

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

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

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

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