CS-GY 6763: Lecture 9
Finish Second Order Conditions, Online and Stochastic Gradient Descent

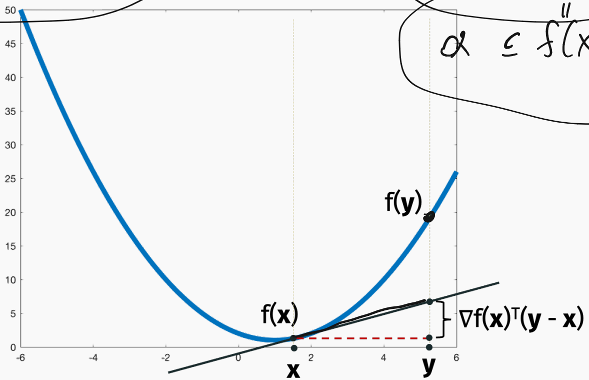
NYU TandonSchool of Engineering, Prof. Christopher Musco

SECOND ORDER CONDITIONS

A function is α-strongly convex and β-smooth if for all $\mathbf{x}, \mathbf{y}$:

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

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



Definition (β-smoothness)

A function f is β smooth if and only if, for all $\mathbf{x}, \mathbf{y}$

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

I.e., the gradient function is a β-Lipschitz function.

We won't use this definition directly, but it's good to know. Easy to prove equivalency to previous definition (see Lem. 3.4 in Bubeck's book).

$y = x^*$

$$\|\nabla f(x) - \nabla f(x^*)\|_2 \leq \beta \| x - x^*\|_2$$

$$\|\nabla f(x)\|_2 \leq \beta \|x - x^*\|_2$$

$f(\hat{x}) \leq f(x^*) + \epsilon$

Having either an upper and lower bound on the second derivative helps convergence. Having both helps a lot.

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	$O\left(\frac{G^2}{\alpha\epsilon}\right)$	$O\left(\frac{\beta}{\alpha}\log(1/\epsilon)\right)$

Gradient descent for β-smooth functions:

$\|\nabla f(x)\|_2 \leq G$

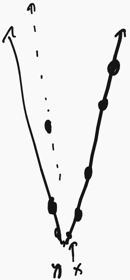
- Select starting point $\mathbf{x}^{(0)}$, $\eta = 1/\beta$.
- For $i = 0, \ldots, T$:
  - $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \eta \nabla f(\mathbf{x}^{(i)})$
- Return $\hat{\mathbf{x}} = \arg\min_i f(\mathbf{x}^{(i)})$.

β

G T

$-\eta \nabla f(x^{(i)})$

Why do you think gradient descent might be faster when a function is β-smooth?



Previously learning rate/step size η depended on G. Now choose it based on β:

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

Progress per step of gradient descent:

$+1/\beta\ \nabla f(x^{(t)})^T \nabla f(x^{(t)})$

1. $[f(\mathbf{x}^{(t+1)}) - f(\mathbf{x}^{(t)})] - \nabla f(\mathbf{x}^{(t)})^T(\underbrace{\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}}_{-1/\beta\ \nabla f(x^{(t)})}) \le \frac{\beta}{2}\|\mathbf{x}^{(t)} - \mathbf{x}^{(t+1)}\|_2^2.$

2. $[f(\mathbf{x}^{(t+1)}) - f(\mathbf{x}^{(t)})] + \frac{1}{\beta}\|\nabla f(\mathbf{x}^{(t)})\|_2^2 \le \frac{\beta}{2}\|\frac{1}{\beta}\nabla f(\mathbf{x}^{(t)})\|_2^2. = \frac{1}{2\beta}\|\nabla f(x^{(t)})\|_2^2$

3. ($f(\mathbf{x}^{(t)}) - f(\mathbf{x}^{(t+1)}) \ge \frac{1}{2\beta}\|\nabla f(\mathbf{x}^{(t)})\|_2^2.$)

$f(x^{(t+1)}) < f(x^{(t)})$

Where did we use convexity in this proof?

CONVERGENCE TO STATIONARY POINT

Theorem (Convergence to Near-Stationary Point)

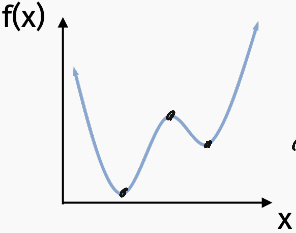
For any β-smooth differentiable function f (convex or not), if we run GD for T steps, we can find a point $\hat{\mathbf{x}}$ such that:

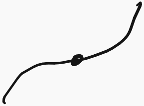
$$\|\nabla f(\hat{\mathbf{x}})\|_2^2 \leq \frac{2\beta}{T}\left(f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)\right) \leq \epsilon$$

?

$$T = \frac{\beta f(x^0) - f(x^*)}{\epsilon}$$

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





We have that $\frac{1}{2\beta}\|\nabla f(\mathbf{x}^{(t)})\|_2^2 \leq f(\mathbf{x}^{(t)}) - f(\mathbf{x}^{(t+1)})$. So:

$$\underbrace{\sum_{t=0}^{T-1} \frac{1}{2\beta}\|\nabla f(\mathbf{x}^{(t)})\|_2^2} \leq f(\mathbf{x}^{(0)}) - f(\mathbf{x}^{(t)})$$

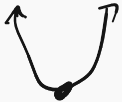
$f(x^0) - \cancel{f(x^1)} + \cancel{f(x^1)} - \cancel{f(x^2)} + \cancel{f(x^2)} \ldots$

$$\frac{1}{T}\sum_{t=0}^{T-1} \|\nabla f(\mathbf{x}^{(t)})\|_2^2 \leq \frac{2\beta}{T}\left(f(\mathbf{x}^{(0)}) - f(\mathbf{x}^{(t)})\right) \leq \frac{2\beta}{T}\left(f(x^0) - f(x^*)\right)$$

$$\min_t \|\nabla f(\mathbf{x}^{(t)})\|_2^2 \leq \frac{2\beta}{T}\left(f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)\right)$$

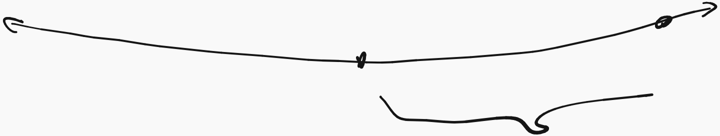
$\hat{x} = \arg\min_t \|\nabla f(x^{(t)})\|_2^2$

For convex functions, we want to further prove that:



$$f(\hat{\mathbf{x}}) - f(\mathbf{x}^*) \leq \frac{2\beta R^2}{T}$$

Convex functions only have one stationary point: the global minimum $\mathbf{x}^*$. Ideally, we would argue that a <u>near-stationary point</u> is a <u>near-minimizer</u>. However, this isn't always the case!



Nevertheless, not to hard to obtain a proof from the progress condition. A concise version can be found on Page 15 in (Garrigos and Gower's notes.)

$T = O\left(\frac{G^2 B^2}{\epsilon^2}\right)$

Theorem (GD convergence for β-smooth functions.)

Let f be a β smooth convex function and assume we have $\|\mathbf{x}^* - \mathbf{x}^{(0)}\|_2 \leq R$. If we run GD for T steps with $\eta = \frac{1}{\beta}$ we have:

$$f(\mathbf{x}^{(T)}) - f(\mathbf{x}^*) \leq \frac{2\beta R^2}{T}$$

Corollary: If $T = O\left(\frac{\beta R^2}{\epsilon}\right)$ we have $f(\mathbf{x}^{(T)}) - f(\mathbf{x}^*) \leq \epsilon$.

Note: This is not optimal! Can be improved to depend on $O(1/T^2)$ using a technique called acceleration.

$f(x^{(T)}) \le f(x^*) + \varepsilon$

What if f is both β-smooth and α-strongly convex?

Theorem (GD for β-smooth, α-strongly convex.) $f(x^{(T)}) - f(x^*)$

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:

$$\frac{2}{\beta}\left[f(x^{(T)}) - f(x^*)\right] \le \|\mathbf{x}^{(T)} - \mathbf{x}^*\|_2^2 \le e^{-T\frac{\alpha}{\beta}} \underbrace{\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2^2}_{\le R} \le$$

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

Is it better if κ is large or small?

$\beta \ge \alpha$ $\qquad \frac{\beta}{\alpha} \ge 1$ $\qquad e^{-T\alpha/\beta} = \varepsilon$

$T\alpha/\beta = \log(1/\varepsilon)$

$\boxed{T = \frac{\beta}{\alpha}\log(1/\varepsilon)}$

Converting to more familiar form: Using that fact the $x = x^*$
$\nabla f(\mathbf{x}^*) = \mathbf{0}$ along with

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

$$\frac{\alpha}{2}\|x^*-y\|_2^2 \leq f(y)-f(x^*) \leq \frac{\beta}{2}\|x^*-y\|_2^2$$

we have:

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

$y = x^{(T)}$

We also assume

$$\|\mathbf{x}^{(0)}-\mathbf{x}^*\|_2^2 \leq R^2.$$

Corollary (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:

$$f(\mathbf{x}^{(T)}) - f(\mathbf{x}^*) \leq \frac{\beta}{2} e^{-T\frac{\alpha}{\beta}} \cdot R^2$$

$\log(1/\epsilon)$

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

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

$T = O\left(\sqrt{\frac{\beta}{\alpha}} \cdot \log(R\beta/\epsilon)\right)$

Only depend on $\log(1/\epsilon)$ instead of on $1/\epsilon$ or $1/\epsilon^2$!

Note: Can be further improved with acceleration!

After break we will prove the guarantee for the special case of:

$$f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$$

Goal: Get some of the key ideas across, introduces important concepts like the Hessian, and show the connection between conditioning and linear algebra.

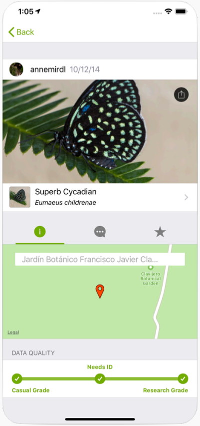
But first we will talk about online gradient descent and stochastic gradient descent.

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

Original motivation for online learning: Often need to train machine learning models on constantly updating/changing data. Do not want to restart from scratch.

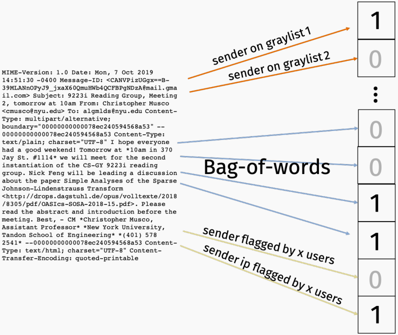
Plant identification via iNaturalist app.

(California Academy of Science + National Geographic)



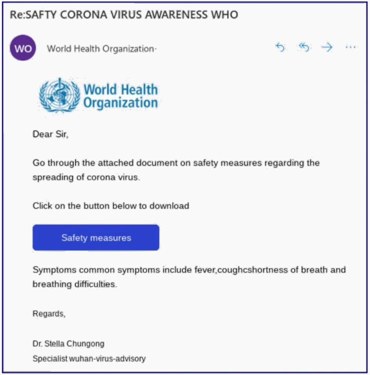
- When the app fails, image is classified via crowdsourcing (backed by huge network of amateurs and experts).
- Single model that is updated constantly, not retrained in batches.

Machine learning based email spam filtering.



Markers for spam change overtime, so model might change.

Machine learning based email spam filtering.



Re:SAFTY CORONA VIRUS AWARENESS WHO

WO World Health Organization·

World Health Organization

Dear Sir,

Go through the attached document on safety measures regarding the spreading of corona virus.

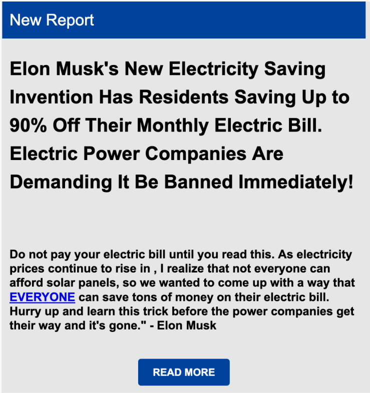
Click on the button below to download

Safety measures

Symptoms common symptoms include fever,coughcshortness of breath and breathing difficulties.

Regards,

Dr. Stella Chungong
Specialist wuhan-virus-advisory



New Report

Elon Musk's New Electricity Saving Invention Has Residents Saving Up to 90% Off Their Monthly Electric Bill. Electric Power Companies Are Demanding It Be Banned Immediately!

Do not pay your electric bill until you read this. As electricity prices continue to rise in , I realize that not everyone can afford solar panels, so we wanted to come up with a way that EVERYONE can save tons of money on their electric bill. Hurry up and learn this trick before the power companies get their way and it's gone." - Elon Musk

READ MORE

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 $1, \ldots, T$, receive data vectors $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(T)}$.

$M_x(a)$

- 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 $y^{(i)}$. Possibly use this information to choose a new $\mathbf{x}^{(i+1)}$.
- Goal is to minimize cumulative loss:

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

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

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

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

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

$f(x)$

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

Regret

Abstraction as optimization problem: Instead of a single objective function f, we have a unknown function $f_1, \ldots, f_T : \mathbb{R}^d \to \mathbb{R}$ for each time step.

- For time step $i \in 1, \ldots, T$, 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 $f_1, \ldots, f_T$ are related to each other at all!

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

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

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

regret.

$\epsilon \leq \sqrt{T}$

$\leq \log(T)$

Here ϵ is called the regret of our solution sequence $\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(T)}$.

We typically ϵ to be growing sublinearly in T.

$x^* = \arg\min_x \sum_{i=1}^{T} f_i(x)$

x^* = "best solution in hindsight"

= "best fixed solution".

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 is very possible that $\left[\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)})\right] < \left[\min_{\mathbf{x}} \sum_{i=1}^{T} f_i(\mathbf{x})\right]$. Could we hope for something stronger?

Exercise: Argue that the following is impossible to achieve:

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

x_i^* $\quad \sum_{i=1}^{T} f_i(x_i^*)$

Convex functions:

If $h_i = 0$
choose $x_i^* = 0$

$T/2$

$$f_1(x) = |x - h_1|$$
$$\vdots$$
$$f_T(x) = |x - h_T|$$

If $h_i = 1$
choose $x_i^* = 1$

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

$x^{(i)}$

$$\sum_{i=1}^{T} \min_{x} f_i(x) = 0$$

$$\mathbb{E}\, f_i(x^{(i)}) = \frac{1}{2}|0 - x^{(i)}| + \frac{1}{2}|1 - x^{(i)}|$$
$$= \frac{1}{2}|x^{(i)}| + \frac{1}{2}|1 - x^{(i)}|$$
$$\geq \frac{1}{2}$$

$$\mathbb{E}\left[\sum_{i=1}^{T} f_i(x^{(i)})\right] \geq \quad T/2$$

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

Beautiful balance:

- Either $f_1, \ldots, f_T$ are similar or changing slowly, so we can learn/predict f_i from earlier functions.
- Or $f_1, \ldots, f_T$ 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.

Follow-the-leader algorithm:

$x^{(i)}$

$f_1, \ldots f_{i-1}$

- Choose $\mathbf{x}^{(0)}$.
- For $i = 1, \ldots, T$:
  - Let $\mathbf{x}^{(i)} = \arg\min_{\mathbf{x}} \sum_{j=1}^{i-1} f_j(\mathbf{x})$.
  - Play $\mathbf{x}^{(i)}$.
  - Observe f_i and incur cost $f_i(\mathbf{x}^{(i)})$.

$\arg\min_x \sum_{j=1}^{i-1} \alpha^j f_j(x)$

Simple and intuitive, but there are two issues with this approach. One is computational, one is related to the accuracy.

Hard case with regret $O(T)$:

$f_1 = f_a$

$f_2 = f_b$

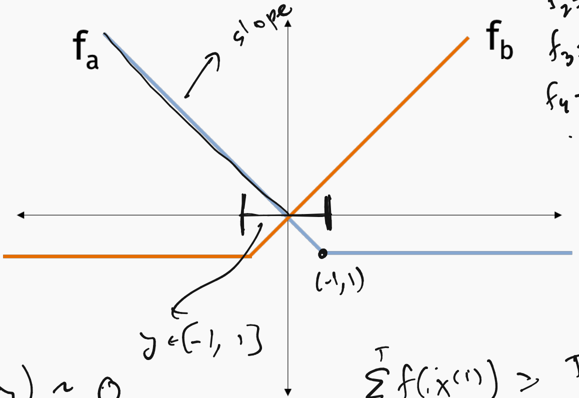
$f_3 = f_b$

$f_4 = f_a$

f_a

f_b

f_b



$\sum_{i=1}^{T} f_i(y) \simeq 0$

$\sum_{i=1}^{T} f(x^{(i)}) > T/2$

https://www.desmos.com/calculator/3t8bfowo3j

Online Gradient descent:

return 3:35

- Choose $\mathbf{x}^{(1)}$ and η.
- For $i = 1, \ldots, T$:
  - 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 $f_1, \ldots, f_T = f$ are all the same, this is the same as regular gradient descent. We update parameters using the gradient ∇f at each step.

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

"best hindsight solution" $\|\nabla f(x)\|_2 \leq G$

Assume:

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

Online Gradient descent:

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

Theorem (OGD Regret Bound)

If the conditions of the previous slide hold, then after T steps,

$$\frac{\epsilon}{T} = \frac{1}{T}\left[\underline{\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)})}\right] - \frac{1}{T}\left[\underline{\sum_{i=1}^{T} f_i(\mathbf{x}^{*})}\right] \leq RG\cancel{\sqrt{T}}. \; / \sqrt{T}$$

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 $f_1, \ldots, f_T$) 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.

$f_i \qquad x^{(i-1)}$

ONLINE GRADIENT DESCENT ANALYSIS

Theorem (OGD Regret Bound)

If the conditions of the previous slide hold, then after T steps,
$\epsilon = \left[\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)})\right] - \left[\sum_{i=1}^{T} f_i(\mathbf{x}^{*})\right] \leq RG\sqrt{T}.$

$x^{(i+1)} = x^{(i)} - \eta \nabla f_i(x^{(i)})$

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

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

(Same proof for standard GD. Only uses convexity of f_i.)

$\bar{x}$

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 $i = 1, \ldots, T$,

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

Telescoping Sum:

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

$$\leq \frac{R^2}{2\eta} + \frac{T\eta G^2}{2} = \frac{RG\sqrt{T}}{2} + \frac{RG\sqrt{T}}{2} = \underline{RG\sqrt{T}}$$

$\eta = \frac{R}{G\sqrt{T}}$

That's it!

→ # of training examples

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

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

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

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:

$[\ a^{(1)}\ ]\quad y^{(1)}$

$\vdots \qquad \vdots$

$[\ a^{(n)}\ ]\quad y^{(n)}$

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

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

Example: least squares linear regression.

$= \|Ax - y\|_2^2$

$O(nd)$

$2A^T(Ax - y)$

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

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

$2(x^T a^{(i)} - y^{(i)}) \cdot a^{(i)} \qquad O(d)$

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

$\nabla f(x)$

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

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

$$\sum_{j=1}^{n} \frac{1}{n} \nabla f_j(x) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x) = \frac{1}{n} \nabla \left[\sum_{i=1}^{n} f_i(x) \right] = \frac{1}{n} \nabla f(x)$$

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

$1/n$

cost

Trade slower convergence for cheaper iterations.

STOCHASTIC GRADIENT DESCENT

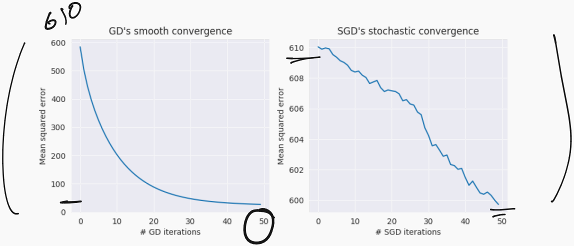
Stochastic first-order oracle for $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x})$.

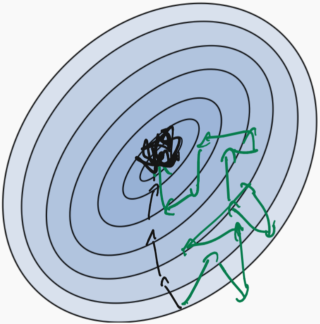
- **Function Query:** For any chosen $j, \mathbf{x}$, return $f_j(\mathbf{x})$
- **Gradient Query:** For any chosen $j, \mathbf{x}$, return $\underline{\nabla f_j(\mathbf{x})}$

Stochastic Gradient descent:

- Choose starting vector $\mathbf{x}^{(1)}$, step size η $\quad j_i \sim 1, \ldots, n$
- For $\underline{i} = \underline{1, \ldots, T}$:
  - Pick random $j_i \in 1, \ldots, n$.
  - $\underline{\mathbf{x}^{(i+1)}} = \underline{\mathbf{x}^{(i)}} - \underline{\eta \nabla f_{j_i}}(\mathbf{x}^{(i)})$
- Return $\underline{\hat{\mathbf{x}}} = \underline{\frac{1}{T} \sum_{i=1}^{T} \mathbf{x}^{(i)}}$

VISUALIZING SGD





Assume:

- Finite sum structure: $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x})$, with $f_1, \ldots, f_n$ 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: $\|\mathbf{x}^* - \mathbf{x}^{(1)}\|_2 \leq R$.

$\|\nabla f(x)\|_2 = \|\sum_{j=1}^{n} \nabla f_j(x)\|_2$

$\leq \sum_{j=1}^{n} \|\nabla f_j(x)\|_2$

$\leq n \cdot \frac{G'}{n} = G'$

Stochastic Gradient descent:

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

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

uniform $1, \ldots, n$

STOCHASTIC GRADIENT DESCENT BOUND

Claim (SGD Convergence)

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

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

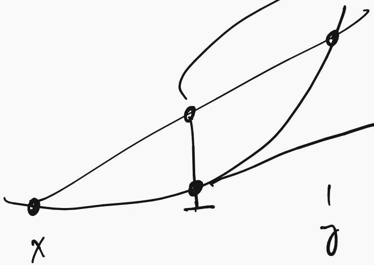
Will prove using:

1. Black-box result for online gradient descent (already proven).
2. The fact that $n \cdot \mathbb{E}[f_{j_i}(\mathbf{x}^{(i)})] = f(\mathbf{x}^{(i)})$.
3. Jensen's inequality.

$$n \cdot \mathbb{E} f_{j_i}(y) = n \cdot \sum_{i=1}^{n} \frac{1}{n} f_i(y) = \sum_{i=1}^{n} f_i(y) = f(y)$$

For a convex function f and points $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t)}$

$$f\left(\frac{1}{t}\cdot \mathbf{x}^{(1)} + \ldots + \frac{1}{t}\cdot \mathbf{x}^{(t)}\right) \leq \frac{1}{t}\cdot f(\mathbf{x}^{(1)}) + \ldots + \frac{1}{t}\cdot f(\mathbf{x}^{(t)})$$



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

Claim (SGD Convergence)

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

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

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

Claim 1:

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

Prove using Jensen's Inequality:

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

$$= \frac{1}{T}\sum_{i=1}^{T}\left[f(x^{(i)}) - f(x^*)\right]$$

$\mathbb{E}[n \cdot f_{j_i}(x^{(i)})] = \mathbb{E}[f(x^{(i)})]$

Claim (SGD Convergence)

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

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

$\to n \cdot f_{j_i}$

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

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

STOCHASTIC GRADIENT DESCENT ANALYSIS

Claim (SGD Convergence)

$f_{j_1} \ldots f_{j_T}$

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

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

$$\begin{aligned}
\mathbb{E}[f(\hat{\mathbf{x}}) - f(\mathbf{x}^*)] &\leq \frac{1}{T}\sum_{i=1}^{T} \mathbb{E}\left[f(\mathbf{x}^{(i)}) - f(\mathbf{x}^*)\right] \\
&= \frac{1}{T}\sum_{i=1}^{T} n\mathbb{E}\left[f_{j_i}(\mathbf{x}^{(i)}) - f_{j_i}(\mathbf{x}^*)\right] = \frac{n}{T}\, \mathbb{E} \sum_{i=1}^{T}\left(f_{j_i}(x^{(i)}) - f_{j_i}(x^*)\right) \\
&\leq \frac{n}{T} \cdot \mathbb{E}\left[\sum_{i=1}^{T} f_{j_i}(\mathbf{x}^{(i)}) - f_{j_i}(\mathbf{x}^{offline})\right],
\end{aligned}$$

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

$$\sum_{i=1}^{T} f_{j_i}(x^*) \geq \sum_{i=1}^{T} f_{j_i}(x^{offline})$$

Claim (SGD Convergence)

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

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

$BG\sqrt{T}$

$$\begin{aligned}
\mathbb{E}[f(\hat{\mathbf{x}}) - f(\mathbf{x}^*)] &\leq \frac{1}{T}\sum_{i=1}^{T}\mathbb{E}\left[f(\mathbf{x}^{(i)}) - f(\mathbf{x}^*)\right] \\
&= \frac{1}{T}\sum_{i=1}^{T} n\mathbb{E}\left[f_{j_i}(\mathbf{x}^{(i)}) - f_{j_i}(\mathbf{x}^*)\right] \\
&\leq \frac{n}{T}\cdot\mathbb{E}\left[\sum_{i=1}^{T} f_{j_i}(\mathbf{x}^{(i)}) - f_{j_i}(\mathbf{x}^{offline})\right] \\
&\leq \frac{n}{T}\cdot\left(R\cdot\frac{G'}{n}\cdot\sqrt{T}\right) \qquad \text{(by OGD guarantee.)}
\end{aligned}$$

$\to B \cdot \frac{G'}{n} \cdot \sqrt{T}$

$= \frac{BG'}{\sqrt{T}}$

$T = \frac{B^2 G'^2}{\epsilon^2}$

$\|\nabla f(x)\|_2 \leq G$

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 $G \leq G'$:

$\max_x \|\nabla f_1(x) + \ldots \nabla f_n(x)\|_2 \leq$

$$G = \max_{\mathbf{x}} \|\nabla f(\mathbf{x})\|_2 \leq \max_{\mathbf{x}} (\|\nabla f_1(\mathbf{x})\|_2 + \ldots + \|\nabla f_n(\mathbf{x})\|_2)$$
$$\leq \max_{\mathbf{x}} (\|\nabla f_1(\mathbf{x})\|_2) + \ldots + \max_{\mathbf{x}} (\|\nabla f_n(\mathbf{x})\|_2)$$
$$\leq n \cdot \frac{G'}{n} = G'.$$

$f_1 = f_2 = \ldots f_n$

$G = G'$

So GD converges strictly faster than *SGD*.

But for a fair comparison:

- SGD cost = (# of iterations) $\cdot$ $O(1)$
- GD cost = (# of iterations) $\cdot$ $O(n)$

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 $G = G'$?

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?

of dimension d.

$\nabla f_i(x)$ is random Gaussian

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

$\|\nabla f_i(x)\|_2 \approx \sqrt{d}$

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

$\|\nabla f(x)\|_2 \approx \sqrt{nd}$

$\sum_{i=1}^{n} N(0,1) = N(0,n)$

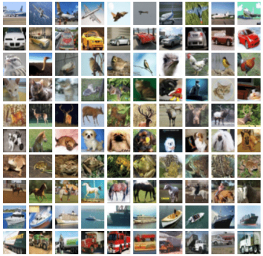
$\frac{G'}{n} \approx \sqrt{d}$

$G' \approx n\sqrt{d}$

$G \approx \sqrt{nd}$

$G \leq \frac{1}{\sqrt{n}} G'$

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., is smoother, or 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

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

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:

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

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!

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

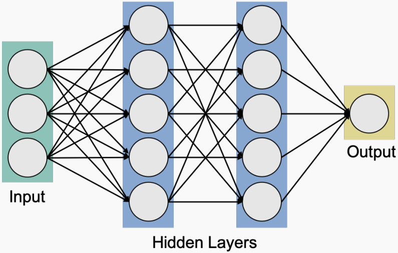
Gradient descent on g:

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