COMPSCI 514: Algorithms for Data Science

Cameron Musco

University of Massachusetts Amherst. Fall 2024.

Lecture 24 (Final Lecture!)

Logistics

- Problem Set 5 can be submitted up to Thursday at 11:59pm.
- Exam is next Wednesday 12/18, from 10:30am-12:30pm in the Totman Gym.
- Similar format to midterm. Closed book, no calculators.
- I am holding exam review office hours this Friday 12/13 10-11:30am in ELab 303 and next Tuesday 12/17 2:30pm-4pm in LGRC A112.
- It would be really helpful if you could fill out SRTIs for this class.

Summary

Last Class:

- Analysis of gradient descent for convex and Lipschitz functions.
- Direct extension to constrained optimization via projected gradient descent. Analysis for convex functions and convex constraint sets. + projection moves us closer to θ^*
- Motivation for stochastic gradient descent (SGD) for performing gradient descent at scale.

This Class:

- Online optimization and online gradient descent.
- Analysis of online gradient descent. (essentially same GD analysis)
- Application to analysis of SGD as a special case.

Gradient Descent At Scale

Typical Optimization Problem in Machine Learning: Given data points $\vec{x}_1, \ldots, \vec{x}_n$ and labels/observations $y_1, \ldots, y_n$ solve:

$$\vec{\theta}^* = \arg\min_{\vec{\theta} \in \mathbb{R}^d} \nabla L(\vec{\theta}, \mathrm{X}, y) = \sum_{j=1}^{n} \nabla \ell(M_{\vec{\theta}}(\vec{x}_j), y_j).$$

loss on training point i

The gradient of $L(\vec{\theta}, \mathrm{X})$ has one component per data point so can be very expensive to compute.

Gradient Descent At Scale

Typical Optimization Problem in Machine Learning: Given data points $\vec{x}_1, \ldots, \vec{x}_n$ and labels/observations $y_1, \ldots, y_n$ solve:

$$\vec{\theta}^* = \arg\min_{\vec{\theta} \in \mathbb{R}^d} L(\vec{\theta}, \mathbf{X}, y) = \sum_{j=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_j), y_j).$$

The gradient of $L(\vec{\theta}, \mathbf{X})$ has one component per data point so can be very expensive to compute.

Solution: Update using just a single data point, or a small batch of data points per iteration.

- If the data point is chosen uniformly at random, the sampled gradient is correct in expectation.

$$\vec{\nabla} L(\vec{\theta}, \mathbf{X}) = \sum_{i=j}^{n} \vec{\nabla} \ell(M_{\vec{\theta}}(\vec{x}_j), y_j) \to \boxed{\mathbb{E}_{j \sim [n]}[\vec{\nabla} \ell(M_{\vec{\theta}}(\vec{x}_j), y_j)]} = \frac{1}{n} \cdot \vec{\nabla} L(\vec{\theta}, \mathbf{X}).$$

- The key idea behind **stochastic gradient descent** (SGD).

Online Gradient Descent

SGD is closely related to online gradient descent.

Online Gradient Descent

SGD is closely related to online gradient descent.

In reality many learning problems are online.

- Websites optimize ads or recommendations to show users, given continuous feedback from these users.
- Spam filters are incrementally updated and adapt as they see more examples of spam over time.
- Face recognition systems, other classification systems, learn from mistakes over time.

Online Gradient Descent

SGD is closely related to online gradient descent.

In reality many learning problems are online.

- Websites optimize ads or recommendations to show users, given continuous feedback from these users.
- Spam filters are incrementally updated and adapt as they see more examples of spam over time.
- Face recognition systems, other classification systems, learn from mistakes over time.

Want to minimize some global loss $L(\vec{\theta}, \mathbf{X})$, when data points are presented in an online fashion $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n$ (like in streaming algorithms)

Online Gradient Descent

SGD is closely related to online gradient descent.

In reality many learning problems are online.

- Websites optimize ads or recommendations to show users, given continuous feedback from these users.
- Spam filters are incrementally updated and adapt as they see more examples of spam over time.
- Face recognition systems, other classification systems, learn from mistakes over time.

Want to minimize some global loss $L(\vec{\theta}, \mathbf{X})$, when data points are presented in an online fashion $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n$ (like in streaming algorithms)

Will view SGD as a special case: when data points are presented (by design) in a random order.

Online Optimization Formal Setup

Online Optimization: In place of a single function f, we see a different objective function at each step:

$$f_1, \ldots, f_t : \mathbb{R}^d \to \mathbb{R}$$

Online Optimization Formal Setup

Online Optimization: In place of a single function f, we see a different objective function at each step:

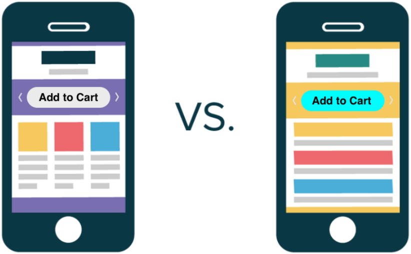
$$f_1, \ldots, f_t : \mathbb{R}^d \to \mathbb{R}$$

- At each step, first pick (play) a parameter vector $\vec{\theta}^{(i)}$.
- Then are told f_i and incur cost $f_i(\vec{\theta}^{(i)})$.
- **Goal:** Minimize total cost $\sum_{i=1}^{t} f_i(\vec{\theta}^{(i)})$.

No assumptions on how $f_1, \ldots, f_t$ are related to each other!

Online Optimization Example

UI design via online optimization.



- Parameter vector $\vec{\theta}^{(i)}$: some encoding of the layout at step i.
- Functions $f_1, \ldots, f_t$: $f_i(\vec{\theta}^{(i)}) = 1$ if user does not click 'add to cart' and $f_i(\vec{\theta}^{(i)}) = 0$ if they do click.
- Want to maximize number of purchases. I.e., minimize $\sum_{i=1}^{t} f_i(\vec{\theta}^{(i)})$

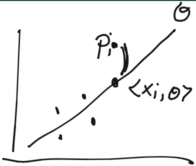
Online Optimization Example

Home pricing tools. f_i, θ_i

linear model

$\langle \vec{x}, \vec{\theta} \rangle$

$275,000



$\vec{x} = [\#baths, \#beds, \#floors \ldots]$

- Parameter vector $\vec{\theta}^{(i)}$: coefficients of linear model at step i.
- Functions $f_1, \ldots, f_t$: $f_i(\vec{\theta}^{(i)}) = (\vec{\theta}^{(i)} - price_i)^2$ revealed when $home_i$ is listed or sold.

  $\langle \vec{\theta}^{(i)}, x_i \rangle$ predicted price

- Want to minimize total squared error $\sum_{i=1}^{t} f_i(\vec{\theta}^{(i)})$ (same as classic least squares regression).

Regret

In normal optimization, we seek $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq \min_{\vec{\theta}} f(\vec{\theta}) + \epsilon.$$

$f(\hat{\theta}) - f(\theta^*) \leq \epsilon$

Regret

In normal optimization, we seek $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq \min_{\vec{\theta}} f(\vec{\theta}) + \epsilon.$$

In online optimization we will ask for the same.

optimal offline solution in hindsight

$$\sum_{i=1}^{t} f_i(\vec{\theta}^{(i)}) \leq \min_{\vec{\theta}} \sum_{i=1}^{t} f_i(\vec{\theta}) + \epsilon = \sum_{i=1}^{t} f_i(\vec{\theta}^{\text{off}}) + \epsilon$$

ϵ is called the regret.

Regret

In normal optimization, we seek $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq \min_{\vec{\theta}} f(\vec{\theta}) + \epsilon.$$

In online optimization we will ask for the same.

$$\sum_{i=1}^{t} f_i(\vec{\theta}^{(i)}) \leq \min_{\vec{\theta}} \sum_{i=1}^{t} f_i(\vec{\theta}) + \epsilon = \sum_{i=1}^{t} f_i(\vec{\theta}^{off}) + \epsilon$$

ϵ is called the regret.

- This error metric is a bit 'unfair'. Why?

Regret

In normal optimization, we seek $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq \min_{\vec{\theta}} f(\vec{\theta}) + \epsilon.$$

In online optimization we will ask for the same.

$$\sum_{i=1}^{t} f_i(\vec{\theta}^{(i)}) \leq \min_{\vec{\theta}} \sum_{i=1}^{t} f_i(\vec{\theta}) + \epsilon = \sum_{i=1}^{t} f_i(\vec{\theta}^{\text{off}}) + \epsilon$$

$f_1 = \theta^2 \quad \theta_1 = 0$

$f_2 = (\theta-1)^2 \quad \theta_2 = 1$

$f_3 = \theta^2 \quad \theta_3 = 0$

$f_4 = (\theta-1)^2 \quad \theta_4 = 1$

$\theta^{\text{off}} = \frac{1}{2}$

$\theta^{\text{off}} = \arg\min \sum_{i=1}^{t} f_i(\theta)$

$\sum f_i(\theta^i) < \sum f_i(\theta^{\text{off}})$

ϵ is called the regret.

- This error metric is a bit 'unfair'. Why?
- Comparing online solution to best fixed solution in hindsight. ϵ can be negative!

$\min_{\theta_1, \ldots, \theta_t} \sum_{i=1}^{t} f_i(\theta_i)$

online optimum "too strong"

Online Gradient Descent

Assume that:

- $f_1, \ldots, f_t$ are all convex.
- Each f_i is G-Lipschitz (i.e., $\|\vec{\nabla} f_i(\vec{\theta})\|_2 \leq G$ for all $\vec{\theta}$.)
- $\|\vec{\theta}^{(1)} - \vec{\theta}^{off}\|_2 \leq R$ where $\theta^{(1)}$ is the first vector chosen.

Same assumptions as GD.

$\theta^{(1)} = 0 \quad \|\theta^{off}\|_2 \leq R$

Assume that:

- $f_1, \ldots, f_t$ are all convex.
- Each f_i is G-Lipschitz (i.e., $\|\vec{\nabla} f_i(\vec{\theta})\|_2 \leq G$ for all $\vec{\theta}$.)
- $\|\vec{\theta}^{(1)} - \vec{\theta}^{off}\|_2 \leq R$ where $\theta^{(1)}$ is the first vector chosen.

Online Gradient Descent

- Set step size $\eta = \frac{R}{G\sqrt{t}}$.
- For $i = 1, \ldots, t$
  - Play $\vec{\theta}^{(i)}$ and incur cost $f_i(\vec{\theta}^{(i)})$.
  - $\vec{\theta}^{(i+1)} = \vec{\theta}^{(i)} - \eta \cdot \vec{\nabla} f_i(\vec{\theta}^{(i)})$

Online Gradient Descent Analysis

Theorem – OGD on Convex Lipschitz Functions: For convex G-Lipschitz $f_1, \ldots, f_t$, OGD initialized with starting point $\theta^{(1)}$ within radius R of θ^{off}, using step size $\eta = \frac{R}{G\sqrt{t}}$, has regret bounded by:

$$\left[\sum_{i=1}^{t} f_i(\theta^{(i)}) - \sum_{i=1}^{t} f_i(\theta^{\cancel{*}\,off})\right] \leq RG\sqrt{t}$$

Theorem – OGD on Convex Lipschitz Functions: For convex G-Lipschitz $f_1, \ldots, f_t$, OGD initialized with starting point $\theta^{(1)}$ within radius R of θ^{off}, using step size $\eta = \frac{R}{G\sqrt{t}}$, has regret bounded by:

$O(\log T)$

$$\left[\sum_{i=1}^{t} f_i(\theta^{(i)}) - \sum_{i=1}^{t} f_i(\theta^*)\right] \leq RG\sqrt{t}$$

$\sqrt{t}$ instead of t

Average regret goes to 0 and $t \to \infty$. 'Sublinear regret' or 'no regret' algorithm.

$$\frac{1}{t}\sum f_i(\theta^i) - \frac{1}{t}\sum_{i=1}^{t} f_i(\theta^{off}) \leq \frac{RG}{\sqrt{t}}$$

Online Gradient Descent Analysis

Theorem – OGD on Convex Lipschitz Functions: For convex G-Lipschitz $f_1, \ldots, f_t$, OGD initialized with starting point $\theta^{(1)}$ within radius R of θ^{off}, using step size $\eta = \frac{R}{G\sqrt{t}}$, has regret bounded by:

$$\left[\sum_{i=1}^{t} f_i(\theta^{(i)}) - \sum_{i=1}^{t} f_i(\theta^*)\right] \leq RG\sqrt{t}$$

Average regret goes to 0 and $t \to \infty$. 'Sublinear regret' or 'no regret' algorithm. No assumptions on $f_1, \ldots, f_t$!

Theorem – OGD on Convex Lipschitz Functions: For convex G-Lipschitz $f_1, \ldots, f_t$, OGD initialized with starting point $\theta^{(1)}$ within radius R of θ^{off}, using step size $\eta = \frac{R}{G\sqrt{t}}$, has regret bounded by:

$$\left[\sum_{i=1}^{t} f_i(\theta^{(i)}) - \sum_{i=1}^{t} f_i(\theta^*)\right] \leq RG\sqrt{t}$$

Average regret goes to 0 and $t \to \infty$. 'Sublinear regret' or 'no regret' algorithm. No assumptions on $f_1, \ldots, f_t$!

Step 1.1: For all i, $\nabla f_i(\theta^{(i)})^T(\theta^{(i)} - \theta^{off}) \leq \frac{\|\theta^{(i)}-\theta^{off}\|_2^2 - \|\theta^{(i+1)}-\theta^{off}\|_2^2}{2\eta} + \frac{\eta G^2}{2}$.

Convexity $\implies$ **Step 1:** For all i,

$$f_i(\theta^{(i)}) - f_i(\theta^{off}) \leq \frac{\|\theta^{(i)} - \theta^{off}\|_2^2 - \|\theta^{(i+1)} - \theta^{off}\|_2^2}{2\eta} + \frac{\eta G^2}{2}.$$

Online Gradient Descent Analysis

Theorem – OGD on Convex Lipschitz Functions: For convex G-Lipschitz $f_1, \ldots, f_t$, OGD initialized with starting point $\theta^{(1)}$ within radius R of θ^{off}, using step size $\eta = \frac{R}{G\sqrt{t}}$, has regret bounded by:

$$\left[\sum_{i=1}^{t} f_i(\theta^{(i)}) - \sum_{i=1}^{t} f_i(\theta^{off})\right] \leq RG\sqrt{t}$$

Step 1: For all i, $f_i(\theta^{(i)}) - f_i(\theta^{off}) \leq \frac{\|\theta^{(i)} - \theta^{off}\|_2^2 - \|\theta^{(i+1)} - \theta^{off}\|_2^2}{2\eta} + \frac{\eta G^2}{2}$

Theorem – OGD on Convex Lipschitz Functions: For convex G-Lipschitz $f_1, \ldots, f_t$, OGD initialized with starting point $\theta^{(1)}$ within radius R of θ^{off}, using step size $\eta = \frac{R}{G\sqrt{t}}$, has regret bounded by:

$$\left[\sum_{i=1}^{t} f_i(\theta^{(i)}) - \sum_{i=1}^{t} f_i(\theta^{off})\right] \leq RG\sqrt{t}$$

Step 1: For all i, $f_i(\theta^{(i)}) - f_i(\theta^{off}) \leq \frac{\|\theta^{(i)}-\theta^{off}\|_2^2 - \|\theta^{(i+1)}-\theta^{off}\|_2^2}{2\eta} + \frac{\eta G^2}{2} \implies$

$$\left[\sum_{i=1}^{t} f_i(\theta^{(i)}) - \sum_{i=1}^{t} f_i(\theta^{off})\right] \leq \sum_{i=1}^{t} \frac{\|\theta^{(i)} - \theta^{off}\|_2^2 - \|\theta^{(i+1)} - \theta^{off}\|_2^2}{2\eta} + \frac{\eta G^2}{2}.$$

$\|\theta^1 - \theta^{off}\| - \|\theta^2 - \theta^{off}\| + \|\theta^2 - \theta^{off}\| - \ldots$

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

$\leq \frac{\|\theta^{(1)} - \theta^{off}\|_2^2}{2\eta} + \frac{\eta G^2}{2} \leq \frac{R^2}{2\eta} + \frac{\eta G^2}{2}$

$\leq \frac{R^2}{2\frac{R}{G\sqrt{t}}} + \frac{\frac{R}{G\sqrt{t}} \cdot G^2}{2} = RG\sqrt{t}$

Stochastic Gradient Descent

Recall: Stochastic gradient descent is an efficient offline optimization method, seeking $\hat{\theta}$ with

$$f(\hat{\theta}) \leq \min_{\vec{\theta}} f(\vec{\theta}) + \epsilon = f(\vec{\theta}^*) + \epsilon.$$

Stochastic Gradient Descent

Recall: Stochastic gradient descent is an efficient offline optimization method, seeking $\hat{\theta}$ with

$$f(\hat{\theta}) \leq \min_{\vec{\theta}} f(\vec{\theta}) + \epsilon = f(\vec{\theta}^*) + \epsilon.$$

Easily analyzed as a special case of online gradient descent!

Stochastic Gradient Descent

Assume that:

training loss

- f is convex and decomposable as $f(\vec{\theta}) = \sum_{j=1}^{n} f_j(\vec{\theta})$.

loss on data point i

Stochastic Gradient Descent

Assume that:

- f is convex and decomposable as $f(\vec{\theta}) = \sum_{j=1}^{n} f_j(\vec{\theta})$.
  - E.g., $L(\vec{\theta}, \mathbf{X}) = \sum_{j=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_j), y_j)$.

Stochastic Gradient Descent

Assume that:

- f is convex and decomposable as $f(\vec{\theta}) = \sum_{j=1}^{n} f_j(\vec{\theta})$.
    - E.g., $L(\vec{\theta}, \mathbf{X}) = \sum_{j=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_j), y_j)$.
- Each f_j is $\frac{G}{n}$-Lipschitz (i.e., $\|\vec{\nabla} f_j(\vec{\theta})\|_2 \leq \frac{G}{n}$ for all $\vec{\theta}$.)

f_j is convex.

Assume that:

- f is convex and decomposable as $f(\vec{\theta}) = \sum_{j=1}^{n} f_j(\vec{\theta})$.
  - E.g., $L(\vec{\theta}, \mathbf{X}) = \sum_{j=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_j), y_j)$.
- Each f_j is $\frac{G}{n}$-Lipschitz (i.e., $\|\vec{\nabla} f_j(\vec{\theta})\|_2 \leq \frac{G}{n}$ for all $\vec{\theta}$.)
  - What does this imply about how Lipschitz f is?

triangle inequality

$$\|\nabla f(\theta)\|_2 = \left\|\sum_{j=1}^{n} \nabla f_j(\theta)\right\|_2 \leq \sum_{j=1}^{n} \|\nabla f_j(\theta)\|_2$$

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

Stochastic Gradient Descent

Assume that:

- f is convex and decomposable as $f(\vec{\theta}) = \sum_{j=1}^{n} f_j(\vec{\theta})$.
    - E.g., $L(\vec{\theta}, \mathbf{X}) = \sum_{j=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_j), y_j)$.
- Each f_j is $\frac{G}{n}$-Lipschitz (i.e., $\|\vec{\nabla} f_j(\vec{\theta})\|_2 \leq \frac{G}{n}$ for all $\vec{\theta}$.)
    - What does this imply about how Lipschitz f is?
- Initialize with $\theta^{(1)}$ satisfying $\|\vec{\theta}^{(1)} - \vec{\theta}^*\|_2 \leq R$.

Stochastic Gradient Descent

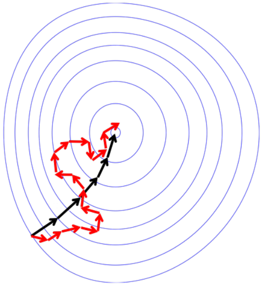
Assume that:

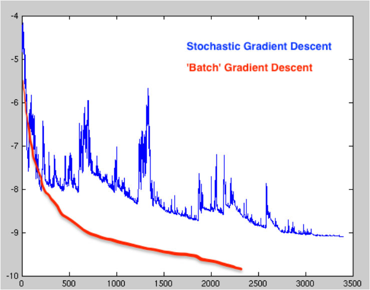
- f is convex and decomposable as $f(\vec{\theta}) = \sum_{j=1}^{n} f_j(\vec{\theta})$.
  - E.g., $L(\vec{\theta}, \mathbf{X}) = \sum_{j=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_j), y_j)$.
- Each f_j is $\frac{G}{n}$-Lipschitz (i.e., $\|\vec{\nabla} f_j(\vec{\theta})\|_2 \leq \frac{G}{n}$ for all $\vec{\theta}$.)
  - What does this imply about how Lipschitz f is?
- Initialize with $\theta^{(1)}$ satisfying $\|\vec{\theta}^{(1)} - \vec{\theta}^*\|_2 \leq R$.

Stochastic Gradient Descent

- Set step size $\eta = \frac{R}{G\sqrt{t}}$.
- For $i = 1, \ldots, t$
  - Pick random $j_i \in 1, \ldots, n$.
  - $\vec{\theta}^{(i+1)} = \vec{\theta}^{(i)} - \eta \cdot \vec{\nabla} f_{j_i}(\vec{\theta}^{(i)})$
- Return $\hat{\theta} = \frac{1}{t} \sum_{i=1}^{t} \vec{\theta}^{(i)}$.

Stochastic Gradient Descent





$$\vec{\theta}^{(i+1)} = \vec{\theta}^{(i)} - \eta \cdot \vec{\nabla} f_{j_i}(\vec{\theta}^{(i)}) \text{ vs. } \vec{\theta}^{(i+1)} = \vec{\theta}^{(i)} - \eta \cdot \vec{\nabla} f(\vec{\theta}^{(i)})$$

Note that: $\mathbb{E}[\vec{\nabla} f_{j_i}(\vec{\theta}^{(i)})] = \frac{1}{n} \vec{\nabla} f(\vec{\theta}^{(i)})$.

Analysis extends to any algorithm that takes the gradient step in expectation (mini-batch GD, randomly quantized, measurement noise, differentially private GD, etc.)

Stochastic Gradient Descent Analysis

Theorem – SGD on Convex Lipschitz Functions: SGD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius R of θ^*, outputs $\hat{\theta}$ satisfying: $\mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon$.

Stochastic Gradient Descent Analysis

Theorem – SGD on Convex Lipschitz Functions: SGD run with $t \geq \frac{R^2 G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius R of θ^*, outputs $\hat{\theta}$ satisfying: $\mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon$.

Step 1: $f(\hat{\theta}) - f(\theta^*) \leq \frac{1}{t}\sum_{i=1}^{t}[f(\theta^{(i)}) - f(\theta^*)]$ (you prove on Pset 5, 2.3)

average iterate

Theorem – SGD on Convex Lipschitz Functions: SGD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius R of θ^*, outputs $\hat{\theta}$ satisfying: $\mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon$.

Step 1: $f(\hat{\theta}) - f(\theta^*) \leq \frac{1}{t}\sum_{i=1}^t [f(\theta^{(i)}) - f(\theta^*)]$ (you prove on Pset 5, 2.3)

Step 2: $\mathbb{E}[f(\hat{\theta}) - f(\theta^*)] \leq \frac{n}{t} \cdot \mathbb{E}\left[\sum_{i=1}^t [f_{j_i}(\theta^{(i)}) - f_{j_i}(\theta^*)]\right]$.

$$\sum_{i=1}^{t} f(\theta^i) - f(\theta^*) = n \cdot \mathbb{E}\left[f_{j_i}(\theta^i) - f_{j_i}(\theta^*)\right]$$

Stochastic Gradient Descent Analysis

Theorem – SGD on Convex Lipschitz Functions: SGD run with $t \geq \frac{R^2 G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius R of θ^*, outputs $\hat{\theta}$ satisfying: $\mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon$.

Step 1: $f(\hat{\theta}) - f(\theta^*) \leq \frac{1}{t}\sum_{i=1}^{t}[f(\theta^{(i)}) - f(\theta^*)]$ (you prove on Pset 5, 2.3)

Step 2: $\mathbb{E}[f(\hat{\theta}) - f(\theta^*)] \leq \frac{n}{t} \cdot \mathbb{E}\left[\sum_{i=1}^{t}[f_{j_i}(\theta^{(i)}) - f_{j_i}(\theta^*)]\right]$.

Step 3: $\mathbb{E}[f(\hat{\theta}) - f(\theta^*)] \leq \frac{n}{t} \cdot \mathbb{E}\left[\sum_{i=1}^{t}[f_{j_i}(\theta^{(i)}) - f_{j_i}(\theta^{off})]\right]$.

Stochastic Gradient Descent Analysis

Theorem – SGD on Convex Lipschitz Functions: SGD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius R of θ^*, outputs $\hat{\theta}$ satisfying: $\mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon$.

Step 1: $f(\hat{\theta}) - f(\theta^*) \leq \frac{1}{t}\sum_{i=1}^{t}[f(\theta^{(i)}) - f(\theta^*)]$ (you prove on Pset 5, 2.3)

Step 2: $\mathbb{E}[f(\hat{\theta}) - f(\theta^*)] \leq \frac{n}{t} \cdot \mathbb{E}\left[\sum_{i=1}^{t}[f_{j_i}(\theta^{(i)}) - f_{j_i}(\theta^*)]\right]$.

Step 3: $\mathbb{E}[f(\hat{\theta}) - f(\theta^*)] \leq \frac{n}{t} \cdot \mathbb{E}\left[\underbrace{\sum_{i=1}^{t}[f_{j_i}(\theta^{(i)}) - f_{j_i}(\theta^{off})]}_{\text{bounded by OGD analysis}}\right]$.

Step 4: $\mathbb{E}[f(\hat{\theta}) - f(\theta^*)] \leq \frac{n}{t} \cdot \underbrace{R \cdot \frac{G}{n} \cdot \sqrt{t}}_{\text{OGD bound}} = \boxed{\frac{RG}{\sqrt{t}}}$.

$$\frac{RG}{\sqrt{t}} = \frac{RG}{\frac{RG}{\varepsilon}} = \varepsilon$$

SGD vs. GD

Stochastic gradient descent generally makes more iterations than gradient descent.

Each iteration is much cheaper (by a factor of n).

$$\vec{\nabla} \sum_{j=1}^{n} f_j(\vec{\theta}) \text{ vs. } \vec{\nabla} f_j(\vec{\theta})$$

SGD vs. GD

When $f(\vec{\theta}) = \sum_{j=1}^{n} f_j(\vec{\theta})$ and $\|\vec{\nabla} f_j(\vec{\theta})\|_2 \leq \frac{G}{n}$:

per data point Lipschitz on

Theorem – SGD: After $t \geq \frac{R^2 G^2}{\epsilon^2}$ iterations outputs $\hat{\theta}$ satisfying:

$$\mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon.$$

When $\|\vec{\nabla} f(\vec{\theta})\|_2 \leq \bar{G}$:

over Lipchitz

Theorem – GD: After $t \geq \frac{R^2 \bar{G}^2}{\epsilon^2}$ iterations outputs $\hat{\theta}$ satisfying:

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

SGD vs. GD

When $f(\vec{\theta}) = \sum_{j=1}^{n} f_j(\vec{\theta})$ and $\|\vec{\nabla} f_j(\vec{\theta})\|_2 \leq \frac{G}{n}$:

GD iterations cost n times more SGD iteration

Theorem – SGD: After $t \geq \frac{R^2 G^2}{\epsilon^2}$ iterations outputs $\hat{\theta}$ satisfying:

$$\mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon.$$

When $\|\vec{\nabla} f(\vec{\theta})\|_2 \leq \bar{G}$: $<< G$

Theorem – GD: After $t \geq \frac{R^2 \bar{G}^2}{\epsilon^2}$ iterations outputs $\hat{\theta}$ satisfying:

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

$$\underbrace{\|\vec{\nabla} f(\vec{\theta})\|_2}_{\bar{G}} = \|\vec{\nabla} f_1(\vec{\theta}) + \ldots + \vec{\nabla} f_n(\vec{\theta})\|_2 \leq \sum_{j=1}^{n} \|\vec{\nabla} f_j(\vec{\theta})\|_2 \leq n \cdot \frac{G}{n} \leq G.$$

SGD vs. GD

When $f(\vec{\theta}) = \sum_{j=1}^{n} f_j(\vec{\theta})$ and $\|\vec{\nabla} f_j(\vec{\theta})\|_2 \leq \frac{G}{n}$:

Theorem – SGD: After $t \geq \frac{R^2 G^2}{\epsilon^2}$ iterations outputs $\hat{\theta}$ satisfying:

$$\mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon.$$

When $\|\vec{\nabla} f(\vec{\theta})\|_2 \leq \bar{G}$:

Theorem – GD: After $t \geq \frac{R^2 \bar{G}^2}{\epsilon^2}$ iterations outputs $\hat{\theta}$ satisfying:

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

$$\|\vec{\nabla} f(\vec{\theta})\|_2 = \|\vec{\nabla} f_1(\vec{\theta}) + \ldots + \vec{\nabla} f_n(\vec{\theta})\|_2 \leq \sum_{j=1}^{n} \|\vec{\nabla} f_j(\vec{\theta})\|_2 \leq n \cdot \frac{G}{n} \leq G.$$

When would this bound be tight? – when $f_1 = f_2 = \ldots f_n$

Questions?

Course Review

Randomized Methods

Randomization as a computational resource for massive datasets.

Randomized Methods

Randomization as a computational resource for massive datasets.

- Focus on problems that are easy on small datasets but hard at massive scale – set size estimation, load balancing, distinct elements counting (MinHash), checking set membership (Bloom Filters), frequent items counting (Count-min sketch), near neighbor search (locality sensitive hashing).

Randomized Methods

Randomization as a computational resource for massive datasets.

- Focus on problems that are easy on small datasets but hard at massive scale – set size estimation, load balancing, distinct elements counting (MinHash), checking set membership (Bloom Filters), frequent items counting (Count-min sketch), near neighbor search (locality sensitive hashing).
- Just the tip of the iceberg on randomized streaming/sketching/hashing algorithms. Check out 614 if you want to learn more.

Randomized Methods

Randomization as a computational resource for massive datasets.

- Focus on problems that are easy on small datasets but hard at massive scale – set size estimation, load balancing, distinct elements counting (MinHash), checking set membership (Bloom Filters), frequent items counting (Count-min sketch), near neighbor search (locality sensitive hashing).
- Just the tip of the iceberg on randomized streaming/sketching/hashing algorithms. Check out 614 if you want to learn more.
- In the process covered probability/statistics tools that are very useful beyond algorithm design: concentration inequalities, higher moment bounds, law of large numbers, central limit theorem, linearity of expectation and variance, union bound, median as a robust estimator.

Dimensionality Reduction

Methods for working with (compressing) high-dimensional data

Dimensionality Reduction

Methods for working with (compressing) high-dimensional data

- Started with randomized dimensionality reduction and the JL lemma: compression from *any* d-dimensions to $O(\log n/\epsilon^2)$ dimensions while preserving pairwise distances.

Dimensionality Reduction

Methods for working with (compressing) high-dimensional data

- Started with randomized dimensionality reduction and the JL lemma: compression from *any* d-dimensions to $O(\log n/\epsilon^2)$ dimensions while preserving pairwise distances.
- Dimensionality reduction via low-rank approximation and optimal solution with PCA/eigendecomposition/SVD.

Dimensionality Reduction

Methods for working with (compressing) high-dimensional data

- Started with randomized dimensionality reduction and the JL lemma: compression from *any* d-dimensions to $O(\log n/\epsilon^2)$ dimensions while preserving pairwise distances.
- Dimensionality reduction via low-rank approximation and optimal solution with PCA/eigendecomposition/SVD.
- Low-rank approximation of similarity matrices and entity embeddings (e.g., LSA, word2vec, DeepWalk).

Dimensionality Reduction

Methods for working with (compressing) high-dimensional data

- Started with randomized dimensionality reduction and the JL lemma: compression from *any* d-dimensions to $O(\log n/\epsilon^2)$ dimensions while preserving pairwise distances.
- Dimensionality reduction via low-rank approximation and optimal solution with PCA/eigendecomposition/SVD.
- Low-rank approximation of similarity matrices and entity embeddings (e.g., LSA, word2vec, DeepWalk).
- Spectral graph theory – nonlinear dimension reduction and spectral clustering for community detection.

Dimensionality Reduction

Methods for working with (compressing) high-dimensional data

- Started with randomized dimensionality reduction and the JL lemma: compression from *any* d-dimensions to $O(\log n/\epsilon^2)$ dimensions while preserving pairwise distances.
- Dimensionality reduction via low-rank approximation and optimal solution with PCA/eigendecomposition/SVD.
- Low-rank approximation of similarity matrices and entity embeddings (e.g., LSA, word2vec, DeepWalk).
- Spectral graph theory – nonlinear dimension reduction and spectral clustering for community detection.
- In the process covered linear algebraic tools that are very broadly useful in ML and data science: eigendecomposition, singular value decomposition, projection, norm transformations.

Continuous Optimization

Foundations of continuous optimization and gradient descent.

Continuous Optimization

Foundations of continuous optimization and gradient descent.

- Foundational concepts like convexity, convex sets, Lipschitzness, directional derivative/gradient.

Continuous Optimization

Foundations of continuous optimization and gradient descent.

- Foundational concepts like convexity, convex sets, Lipschitzness, directional derivative/gradient.
- How to analyze gradient descent in a simple setting (convex Lipschitz functions).

Continuous Optimization

Foundations of continuous optimization and gradient descent.

- Foundational concepts like convexity, convex sets, Lipschitzness, directional derivative/gradient.
- How to analyze gradient descent in a simple setting (convex Lipschitz functions).
- Simple extension to projected gradient descent for optimization over a convex constraint set.

Continuous Optimization

Foundations of continuous optimization and gradient descent.

- Foundational concepts like convexity, convex sets, Lipschitzness, directional derivative/gradient.
- How to analyze gradient descent in a simple setting (convex Lipschitz functions).
- Simple extension to projected gradient descent for optimization over a convex constraint set. + online GD + SGD
- Lots that we didn't cover: online and stochastic gradient descent, accelerated methods, adaptive methods, second order methods (quasi-Newton methods), practical considerations. Gave mathematical tools to understand these methods. Check out CS 651 for more.