COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco

University of Massachusetts Amherst. Fall 2019.

Lecture 20

LOGISTICS

- Problem Set 3 was due Friday/Sunday.
- Problem Set 4 will be on optimization. Out before Thanksgiving and due sometime towards the end of classes.
- Final is on December 19th, 10:30am-12:30pm.

Last Class:

- Analysis of gradient descent for optimizing convex functions.
- (The same) analysis of projected gradient descent for optimizing under constraints.

Last Class:

- Analysis of gradient descent for optimizing convex functions.
- (The same) analysis of projected gradient descent for optimizing under constraints.

This Class:

- Stochastic and online gradient descent for computationally efficient and online learning.
- Unified analysis.

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:

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

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:

argmin

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

The gradient of $L(\vec{\theta}, \mathbf{X})$ has one component per data point:

$l_i(\vec{\theta})$

$$\vec{\nabla} L(\vec{\theta}, \mathbf{X}) = \sum_{j=1}^{n} \vec{\nabla} \ell(M_{\vec{\theta}}(\vec{x}_j), y_j).$$

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:

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

The gradient of $L(\vec{\theta}, \mathrm{X})$ has one component per data point:

$$\vec{\nabla} L(\vec{\theta}, \mathrm{X}) = \sum_{j=1}^{n} \vec{\nabla} \ell(M_{\vec{\theta}}(\vec{x}_j), y_j).$$

When n is large this is very expensive to compute!

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:

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

The gradient of $L(\vec{\theta}, \mathbf{X})$ has one component per data point:

$$\vec{\nabla} L(\vec{\theta}, \mathbf{X}) = \sum_{j=1}^{n} \vec{\nabla} \ell(M_{\vec{\theta}}(\vec{x}_j), y_j).$$

When n is large this is very expensive to compute!

Training a neural network on ImageNet would require $n = 14$ million back propagations!

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:

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

The gradient of $L(\vec{\theta}, \mathbf{X})$ has one component per data point:

$$\vec{\nabla} L(\vec{\theta}, \mathbf{X}) = \sum_{j=1}^{n} \vec{\nabla} \ell(M_{\vec{\theta}}(\vec{x}_j), y_j).$$

When n is large this is very expensive to compute!

Training a neural network on ImageNet would require $n = 14$ million back propagations! ... per iteration of GD.

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

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

- Looking at a single data point gives you a coarse, but still useful cue on how to improve your model.

GRADIENT DESCENT AT SCALE

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

- Looking at a single data point gives you a coarse, but still useful cue on how to improve your model.
- If the data point is chosen uniformly at random, the sampled gradient is correct in expectation.

$x_1 \; . \; . \; . \; x_n \qquad x_j$

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

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

- Looking at a single data point gives you a coarse, but still useful cue on how to improve your model.
- If the data point is chosen uniformly at random, the sampled gradient is correct in expectation.

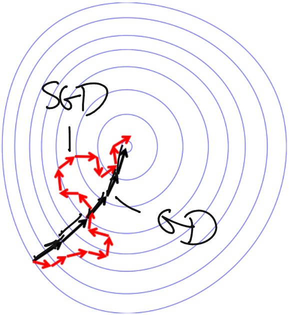
$$\vec{\nabla} L(\vec{\theta}, \mathbf{X}) = \sum_{j=1}^{n} \vec{\nabla} \ell(M_{\vec{\theta}}(\vec{x}_j), y_j) \rightarrow \mathbb{E}_{j \sim [n]}[\vec{\nabla} \ell(M_{\vec{\theta}}(\vec{x}_j), y_j)] = \frac{1}{n} \cdot \vec{\nabla} L(\vec{\theta}, \mathbf{X}).$$

$$= \sum_{j=1}^{n} \frac{1}{n} \cdot \nabla \ell_j$$

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

Stochastic gradient descent takes more, but much cheaper steps than gradient descent.

Stochastic gradient descent takes more, but much cheaper steps than gradient descent.



$$\underline{\vec{\theta}^{(i+1)} = \vec{\theta}^{(i)} - \eta \cdot \vec{\nabla} L(\vec{\theta}^{(i)}, \mathrm{X})} \text{ vs. } \vec{\theta}^{(i+1)} = \underline{\vec{\theta}^{(i)} - \eta \cdot \vec{\nabla} \ell(M_{\vec{\theta}^{(i)}}(\vec{x}_j)}, y_j)$$

ONLINE GRADIENT DESCENT

SGD is closely related to 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.

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

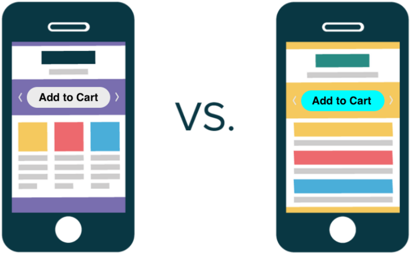
$$f_1, \ldots, f_t : \mathbb{R}^d \to \mathbb{R}$$

- 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 $\sum_{i=1}^{t} f_i(\vec{\theta}^{(i)})$.

$\theta^1 \ldots \theta^t$

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

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 $\sum_{i=1}^{t} f_i(\vec{\theta}^{(i)})$

Home pricing tools.

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

linear model

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

$275,000

- Parameter vector $\vec{\theta}^{(i)}$: coefficients of linear model at step i.
- Functions $f_1, \ldots, f_t$: $f_i(\vec{\theta}^{(i)}) = (\cancel{\vec{\theta}}^{(i)} - price_i)^2$ revealed when $home_i$ is listed or sold. $(\langle \vec{x}_i, \theta^i \rangle - price_i)^2$
- Want to minimize total squared error $\sum_{i=1}^{t} f_i(\vec{\theta}^{(i)})$ (same as classic least squares regression).

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

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

REGRET

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

θ^*

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

In online optimization we will ask for the same.

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

ϵ is called the 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.

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

ϵ is called the regret. regret can be negative.

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

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.

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

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!

Assume that: $(\langle x_i, \theta^i \rangle - y_i)^2$

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

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 $\vec{\theta}$.)
- $\|\vec{\theta}^{(1)} - \vec{\theta}^{ol}\|_2 \leq R$ where $\theta^{(1)}$ is the first vector chosen.

θ^{Ol} : $\arg\min_{\theta} \sum f_i(\theta)$

Online Gradient Descent

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

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 $\vec{\theta}$.)
- $\|\vec{\theta}^{(1)} - \vec{\theta}^{ol}\|_2 \leq R$ 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 $f_i(\vec{\theta}^{(i)})$.
  - $\vec{\theta}^{(i+1)} = \vec{\theta}^{(i)} - \eta \cdot \vec{\nabla} f_i(\vec{\theta}^{(i)})$

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^{ol}, using step size $\eta = \frac{R}{G\sqrt{t}}$, has regret bounded by:

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

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^{ol}, using step size $\eta = \frac{R}{G\sqrt{t}}$, has regret bounded by:

$$\frac{1}{t}\left[\sum_{i=1}^{t} f_i(\theta^{(i)}) - \sum_{i=1}^{t} f_i(\theta^{ol})\right] \le RG\sqrt{t} \qquad = \frac{RG}{\sqrt{t}}$$

Average regret goes to 0 and $t \to \infty$.

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^{ol}, 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 $t \to \infty$. 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^{ol}, 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 $t \to \infty$. No assumptions on $f_1, \ldots, f_t$!

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

$$\|\theta^{i+1} - \theta^{ol}\|^2 = \|\theta^i - \eta \nabla f_i(\theta^i) - \theta^{ol}\|^2$$

$$= \|\theta^i - \theta^{ol}\| - 2\eta \nabla f_i(\theta^i)^T(\theta^i - \theta^{ol}) + \|\cancel{\eta}\nabla f_i(\theta^i)\|_2^2 \quad \eta^2 G^2$$

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^{ol}, using step size $\eta = \frac{R}{G\sqrt{t}}$, has regret bounded by:

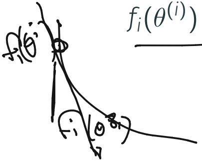
$$\left[\sum_{i=1}^{t} f_i(\theta^{(i)}) - \sum_{i=1}^{t} f_i(\theta^*)\right] \leq RG\sqrt{t}$$

Average regret goes to 0 and $t \to \infty$. No assumptions on $f_1, \ldots, f_t$!

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

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

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



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^{ol}, using step size $\eta = \frac{R}{G\sqrt{t}}$, has regret bounded by:

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

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

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^{ol}, using step size $\eta = \frac{R}{G\sqrt{t}}$, has regret bounded by:

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

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

$\|\theta^1 - \theta^{ol}\|_2^2 - \|\theta^{t+1} \theta^{ol}\|_2^2$

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

$$\leq \frac{\|\theta^1 - \theta^{ol}\|_2^2}{2\eta} + \frac{\eta G^2}{2} \leq \frac{R^2}{2\eta} + \frac{t\eta G^2}{2}$$

$$= \frac{RG\sqrt{t}}{2} + \frac{RG\sqrt{t}}{2}$$

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

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

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!

Assume that:

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

Assume that:

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

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 $\vec{\theta}$.)

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?

$$\nabla f(\theta) = \sum_{j=1}^{n} \nabla f_j(\theta)$$

$$\|\nabla f(\theta)\| = \left\| \sum \nabla f_j(\theta) \right\|$$

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

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

Assume that:

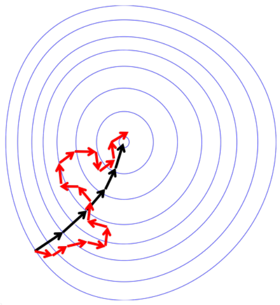
- f is convex and decomposable as $f(\vec{\theta}) = \sum_{j=1}^{n} f_j(\vec{\theta})$.
  - E.g., $L(\vec{\theta}, \mathbf{X}) = \sum_{j=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_j), y_j)$.
- Each f_j is $\frac{G}{n}$-Lipschitz (i.e., $\|\vec{\nabla} f_j(\vec{\theta})\|_2 \leq \frac{G}{n}$ for all $\vec{\theta}$.)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

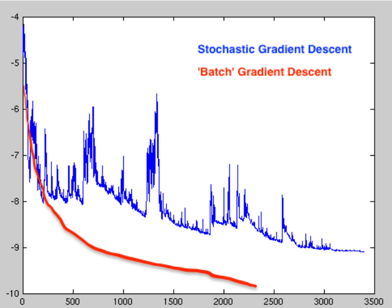
Assume that:

- 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 $j_i \in 1, \ldots, n$.
  - $\vec{\theta}^{(i+1)} = \vec{\theta}^{(i)} - \eta \cdot \underline{\vec{\nabla} f_{j_i}}(\vec{\theta}^{(i)})$
- Return $\hat{\theta} = \frac{1}{t} \sum_{i=1}^{t} \vec{\theta}^{(i)}$.





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

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, etc.)

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

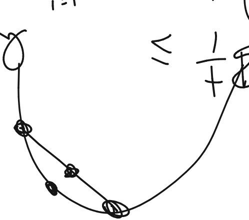
Theorem – SGD on Convex Lipschitz Functions: SGD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius R of θ^*, outputs $\hat{\theta}$ satisfying: $\mathbb{E}[f(\hat{\theta})] \leq f(\theta^*) + \epsilon$.

Step 1: $f(\hat{\theta}) - f(\theta^*) \leq \frac{1}{t}\sum_{i=1}^{t}[f(\theta^{(i)}) - f(\theta^*)]$

$$\hat{\theta} = \frac{1}{t}\sum_{i=1}^{t}\theta^i \Rightarrow f\left(\frac{1}{t}\sum\theta^i\right) - f(\theta^*)$$

Convexity

$$\leq \frac{1}{t}\sum\left[f(\theta^i) - f(\theta^*)\right]$$



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: $f(\hat{\theta}) - f(\theta^*) \leq \frac{1}{t}\sum_{i=1}^{t}[f(\theta^{(i)}) - f(\theta^*)]$

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

$j_1, j_2, \ldots, j_{t-1}$

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

$$= \frac{1}{t}\sum_{i=1}^{t}\mathbb{E}\, f_{j_i}(\theta^i)\cdot n - f_{j_i}(\theta^*)\cdot n]$$

$$f(\theta^i) = \sum_{j=1}^{n} f_j(\theta^i)$$

$$\mathbb{E}\, f_{j_i}(\theta^i) = \sum_{j=1} \Pr(j_i = j)\cdot f_j(\theta^i) = \frac{1}{n}\sum f_j(\theta^i)$$

$$= \frac{1}{n} f(\theta^i)$$

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: $f(\hat{\theta}) - f(\theta^*) \leq \frac{1}{t}\sum_{i=1}^{t}[f(\theta^{(i)}) - f(\theta^*)]$

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

$\theta^{ol} = \arg\min_{\theta} \sum f_{j_i}(\theta)$

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

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: $f(\hat{\theta}) - f(\theta^*) \leq \frac{1}{t}\sum_{i=1}^{t}[f(\theta^{(i)}) - f(\theta^*)]$

$RG\sqrt{t}$

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

$f_{j_1} \; f_{j_2} \ldots f_{j_t}$

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

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

$\frac{RG}{\sqrt{t}} = \frac{RG}{RG/\epsilon} = \epsilon$

$\sum_{i=1}^{t} f_{j_i}(\theta^*) \geq \sum_{i=1}^{t} f_{j_i}(\theta^{ol})$

Stochastic gradient descent generally makes more iterations than gradient descent.

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

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

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:

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

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 $f(\vec{\theta}) = \sum_{j=1}^{n} f_j(\vec{\theta})$ 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?

Questions?