COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco

University of Massachusetts Amherst. Spring 2020.
Lecture 24 (Final Lecture!)

LOGISTICS

- Problem Set 4 is due Sunday 5/3 at 8pm.
- Exam is at **2pm on May 6th**. Open note, similar to midterm.
- Exam review guide and practice problems have been posted under the schedule tab on the course page.
- I will hold usual office hours today and exam review office hours this Thursday and next Tuesday during the regular class time 11:30am-12:45pm
- Regular SRTI's are suspended this semester. But I am holding an optional SRTI for this class and would really appreciate your feedback.
- `http://owl.umass.edu/partners/courseEvalSurvey/uma/`.

Last Class:

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

This Class:

- Online learning, regret, and online gradient descent.
- Application to analysis of stochastic gradient descent (if time).
- Course summary/wrap-up

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 $L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \ell(\vec{\theta}, \vec{x}_i)$, when data points are presented in an online fashion $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n$ (like in streaming algorithms)

Stochastic gradient descent is a special case: when data points are considered a random order for computational reasons.

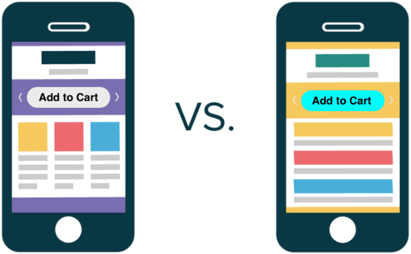
Online Optimization: In place of a single function f, we see a different objective function at each step:

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

- 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!

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.

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 $f_1, \ldots, f_t$: $f_i(\vec{\theta}^{(i)}) = (\langle \vec{x}_i, \vec{\theta}^{(i)} \rangle - price_i)^2$ revealed when $home_i$ is listed or sold.
- 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:

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

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?
- Comparing online solution to best fixed solution in hindsight. ϵ can be negative!

INTUITION CHECK

What if for $i = 1, \ldots, t$, $f_i(\theta) = |x - 1000|$ or $f_i(\theta) = |x + 1000|$ in an alternating pattern?

How small can the regret ϵ be? $\sum_{i=1}^{t} f_i(\vec{\theta}^{(i)}) \leq \sum_{i=1}^{t} f_i(\vec{\theta}^{off}) + \epsilon$.

What if for $i = 1, \ldots, t$, $f_i(\theta) = |x - 1000|$ or $f_i(\theta) = |x + 1000|$ in no particular pattern? How can any online learning algorithm hope to achieve small regret?

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 $f_i(\vec{\theta}^{(i)})$.
  - $\vec{\theta}^{(i+1)} = \vec{\theta}^{(i)} - \eta \cdot \vec{\nabla} f_i(\vec{\theta}^{(i)})$

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:

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

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

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

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

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

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

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

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

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

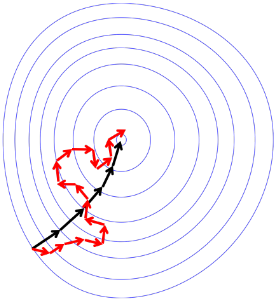
- The most popular optimization method in modern machine learning.
- Easily analyzed as a special case of online gradient descent!

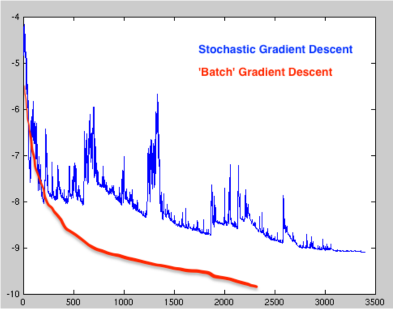
Assume that:

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





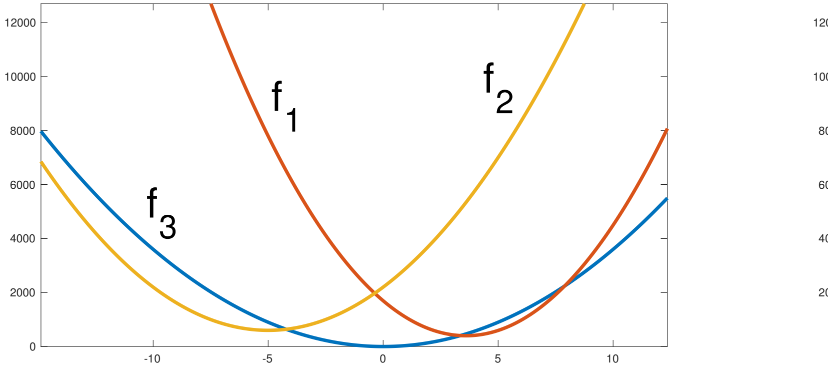
$$\vec{\theta}^{(i+1)} = \vec{\theta}^{(i)} - \eta \cdot \vec{\nabla} f_{j_i}(\vec{\theta}^{(i)}) \text{ vs. } \vec{\theta}^{(i+1)} = \vec{\theta}^{(i)} - \eta \cdot \vec{\nabla} f(\vec{\theta}^{(i)})$$

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 (batch GD, randomly quantized, measurement noise, differentially private, etc.)

TEST OF INTUITION

What does $f_1(\theta) + f_2(\theta) + f_3(\theta)$ look like?



A sum of convex functions is always convex (good exercise).

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

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

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

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:

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

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

When would this bound be tight?

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.
- 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.

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.
- Connections to the weird geometry of high-dimensional space.
- 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.

Foundations of continuous optimization and gradient descent.

- Motivation for continuous optimization as loss minimization in ML. 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 optimization and online gradient descent.
- Lots that we didn't cover: stochastic gradient descent, accelerated methods, adaptive methods, second order methods (quasi-Newton methods), practical considerations. Gave mathematical tools to understand these methods.

Thanks for a great semester!