CS-GY 9223 D: Lecture 6
Online and Stochastic Gradient Decent

NYU Tandon School of Engineering, Prof. Christopher Musco

PROJECT

- If you don't have a project partner by the end of today, please email me.
- Take home midterm **week of October 26th**.
    - 2 hours, self-proctored. Design for 1.25 hours.
    - Can take anytime during that week.
    - Administered either via email or another option.
    - Solutions can be hand-written and scanned.
    - I will post some review questions.
- Need volunteers to present at **10/26 reading group** (in 2 weeks). Sign-up sheet on course webpage.

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

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

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

Projected gradient descent:

- Select starting point $\mathbf{x}^{(0)}$, learning rate η.
- For $i = 0, \ldots, T$:
  - $\mathbf{z} = \mathbf{x}^{(i)} - \eta \nabla f(\mathbf{x}^{(i)})$
  - $\mathbf{x}^{(i+1)} = P_{\mathcal{S}}(\mathbf{z})$
- Return $\hat{\mathbf{x}} = \arg\min_i f(\mathbf{x}^{(i)})$.

Conditions for convergence:

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

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

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

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

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

$$T =$$

iterations.

Today:

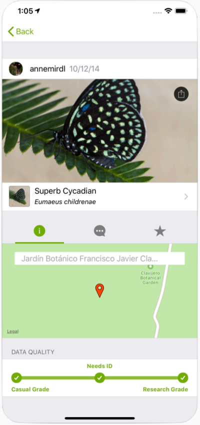
- Basics of Online Learning + Optimization.
- Introduction to Regret Analysis.
- Application to analyzing Stochastic Gradient Descent.

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

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

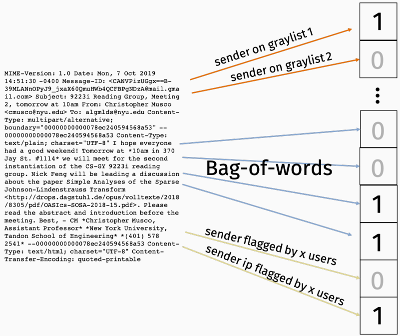
Plant identification via iNaturalist app.

(California Academy of Science + National Geographic)



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

ML based email spam/scam filtering.



Markers for spam change overtime, so model might change.

EXAMPLE

ML based email spam/scam 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

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

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

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

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

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

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

Abstraction as optimization problem: Instead of a single objective function f, we have a single (initially unknown) function $f_i, \ldots, f_T : \mathbb{R}^d \to \mathbb{R}$ 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 <u>no assumptions</u> that $f_1, \ldots, f_T$ are related to each other at all!

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

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

If $f_1, \ldots, f_T$ are very different it might seem like nonsense right now...

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

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

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

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

This guarantee might seem a bit unfair. Why?

Regret compares to the best fixed solution in hindsight.

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

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

Exercise: Argue that the following is impossible to achieve:

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

HARD EXAMPLE FOR ONLINE OPTIMIZATION

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

Beautiful balance:

- Either $f_1, \ldots, f_T$ are similar, so an method like Online Gradient Descent will effectively minimize $\sum_{i=1}^{T} f_i(\mathbf{x}^{(i)})$.
- 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.

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

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

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

Theorem (OGD Regret Bound)

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

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

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

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.

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 $i = 1, \ldots, T$,

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

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

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 $i = 1, \ldots, T$,

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

Telescoping Sum:

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

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

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

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

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:

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

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

Example: least squares linear regression.

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

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

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

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

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

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

Trade slower convergence for cheaper iterations.

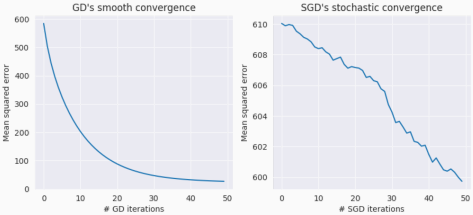
Stochastic first-order oracle for $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x})$.

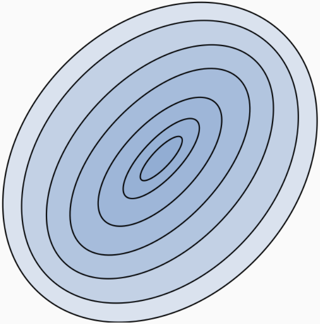
- **Function Query:** For any chosen $j, \mathbf{x}$, return $f_j(\mathbf{x})$
- **Gradient Query:** For any chosen $j, \mathbf{x}$, return $\nabla f_j(\mathbf{x})$

Computing $f(\mathbf{x})$ would take n separate function queries.

Stochastic Gradient descent:

- Choose starting vector $\mathbf{x}^{(1)}$, learning rate η
- 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 $\hat{\mathbf{x}} = \frac{1}{T} \sum_{i=1}^{T} \mathbf{x}^{(i)}$


GD's smooth convergence
Mean squared error
600
500
400
300
200
100
0
0
10
20
30
40
50
GD iterations
SGD's stochastic convergence
Mean squared error
610
608
606
604
602
600
0
10
20
30
40
50
SGD iterations



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

Stochastic Gradient descent:

- Choose $\mathbf{x}^{(1)}$, steps T, learning rate $\eta = \frac{D}{G'\sqrt{T}}$.
- 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 $f_{j_1}, \ldots, f_{j_T}$.

Claim (SGD Convergence)

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

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

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

Claim 1:

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

Claim (SGD Convergence)

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

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

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

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

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 $G \leq G'$:

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

So GD converges strictly faster than *SGD*.

But for a fair comparison:

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

We always have $\|\nabla f(\mathbf{x})\|_2 \leq G'$. 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 $\|\nabla f(\mathbf{x})\|_2 = G'$?

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?

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

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