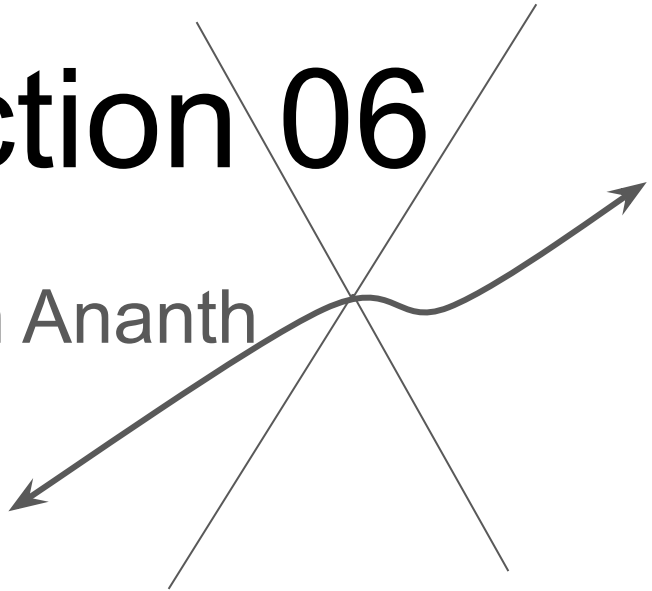
446 Section 06

TA: Varun Ananth



Plans for today!

1. This
2. Reminders
3. PyTorch Tutorial (No Demo)
4. GD, SGD, Mini-Batch GD
5. Gradient Descent Proofs

Reminders

- HW2 due November 5th
  - Keep track of your late days!
- How was the midterm?

Gradient Descent

vs.

Stochastic Gradient Descent

vs.

Mini-Batch Gradient Descent

Gradient Descent Variants (for this class)

Gradient Descent (GD)

- Take a step after looking at **every** single datapoint in the training set

Stochastic Gradient Descent (SGD)

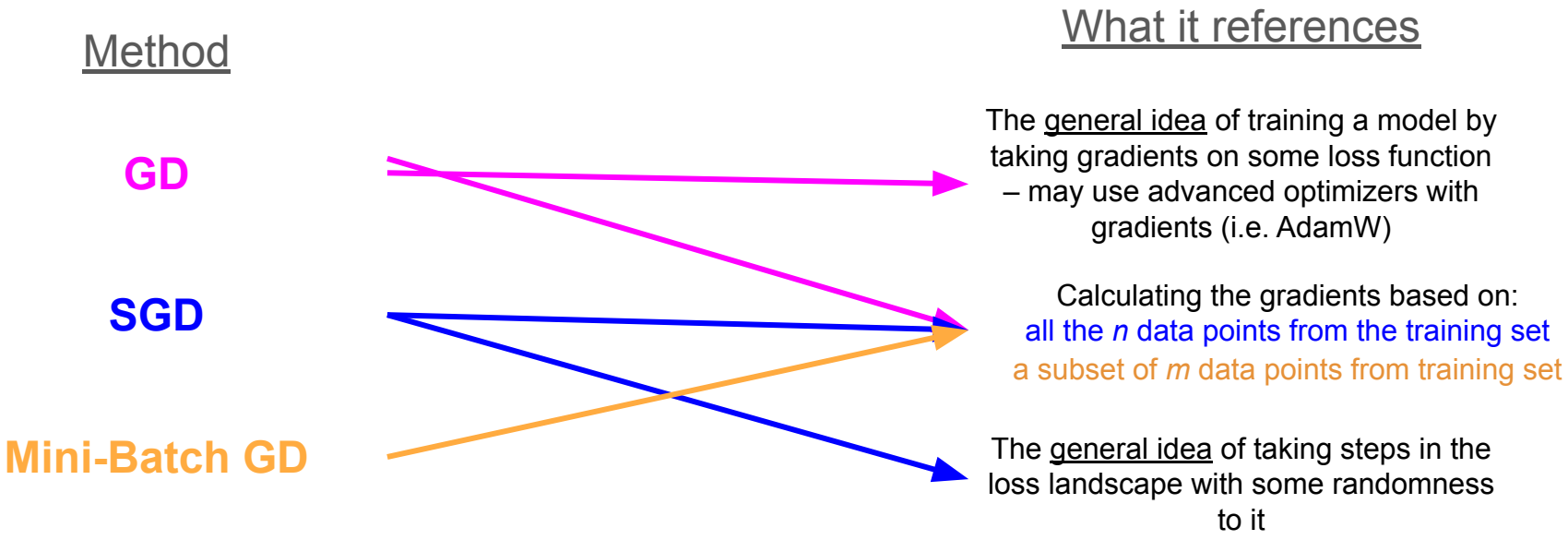
- Take a step after looking at **1** randomly sampled point from the training set

Mini-Batch Gradient Descent (Mini-Batch GD)

- Take a step after looking at ***n*** sampled datapoints from the training set

These definitions are specific and important for this class, but…

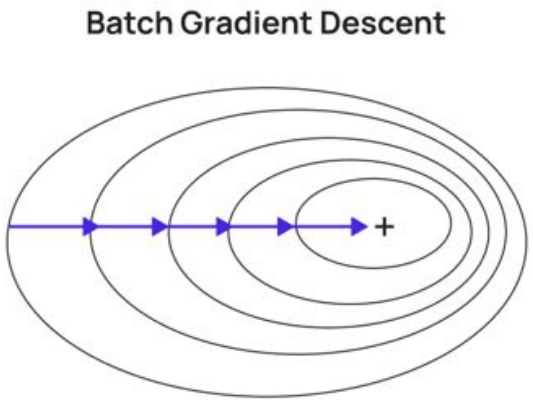
Gradient Descent Variants (as discussed in industry)

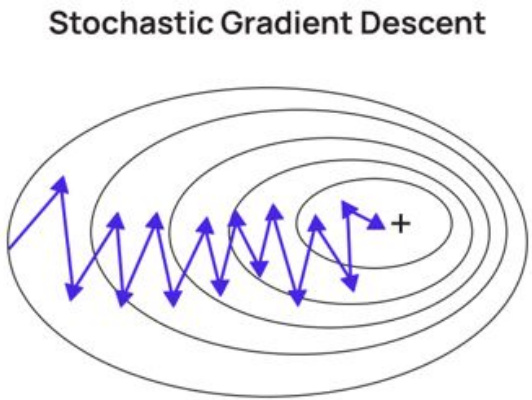


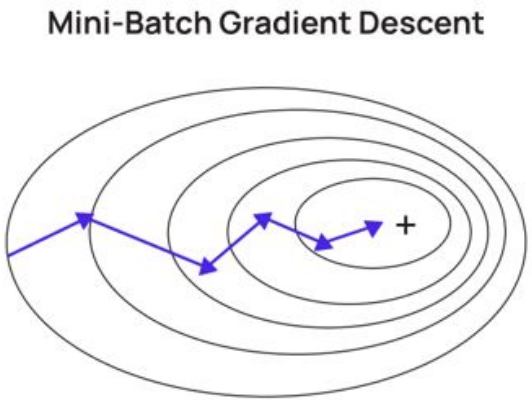
This is how people talk about them IRL. Note that SGB & Minibatch are used interchangeably.

Note: From here on out in this class, the definitions of these terms are **these ones →**

Gradient Descent (GD)
Take a step after looking at **every** single datapoint in the training set
Stochastic Gradient Descent (SGD)
Take a step after looking at **1** randomly sampled point from the training set
Mini-Batch Gradient Descent (Mini-Batch GD)
Take a step after looking at ***n*** sampled datapoints from the training set







Another way of looking at things!

Stochastic Gradient Descent

$$w^{(t+1)} = w^{(t)} - \eta \cdot \frac{1}{n} \sum_{i=1}^{n} \nabla \ell_i \left(w^{(t)} \right).$$

The computational cost of a single step here is $\mathcal{O}(dn)$. To reduce cost, one idea is to just use a subset of all samples to approximate the full gradient. Specifically, consider revising the gradient descent step as follows:

$$w^{(t+1)} = w^{(t)} - \eta \cdot \nabla \ell_{I_t}(w^{(t)}),$$

where I_t is chosen randomly within $\{1, 2, ..., n\}$ with equal probability. This is called **stochastic gradient descent (SGD)**, and the computational cost of a single step now reduces to $\mathcal{O}(d)$.

Is this estimator unbiased?

Yes!

- $\mathbb{E}_{I_t} \left[\nabla \ell_{I_t}(w^{(t)}) \right] = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell_i \left(w^{(t)} \right)$, which is the full gradient. Hence the estimate of gradient is unbiased.

Issues with SGD → MiniBatch GD

- Although the estimator is unbiased overall, a single point gradient estimate is very noisy
  - **Pros:**
    - Noise within the optimization process is not inherently a bad thing. It can help you escape sub-optimal local minima and “wander/explore” the loss landscape more.
    - For giant datasets SGD is also more computationally feasible
  - **Cons:**
    - Noisy updates mean you could also wander off the optimal path if the loss landscape is simpler

Gradient descent is theoretically nice but computationally expensive…

SGD is noisy (maybe too noisy) but computationally efficient…

Meet in the middle with MiniBatch GD!

3b. MiniBatch Gradient Descent

Short form: **Take gradient of loss with *n* datapoints, take a step, and repeat** (as opposed to taking the loss over **all** datapoints)

Choice of *n* can have impacts… Can you think of some?

The choice of the optimal batch size is not an easy question, and there is no standard answer to it. However, we still try to provide some important intuitions regarding the choice of batch size. Firstly, when the objective function (to be minimized) behaves "better" (e.g., Lipschitz continuous, strong convex) than convex functions, the difference in the convergence rates between GD and SGD becomes significant, suggesting a nontrivial gain of having a faster convergence rate and hence we should consider relatively larger batch size. Secondly, a smaller batch size yields less stable gradient estimates, suggesting that we shall employ a fairly small step size/learning rate. An increase in the batch size can be paired with an increase in the step size/learning rate.

Pytorch Tutorial

This Colab notebook walks you through a basic PyTorch tutorial. Very helpful for future homeworks!

We suggest doing this notebook if you don’t know PyTorch or need a refresher. It is optional!

Gradient Descent Proofs

Additional Definitions: Lipschitz Continuity

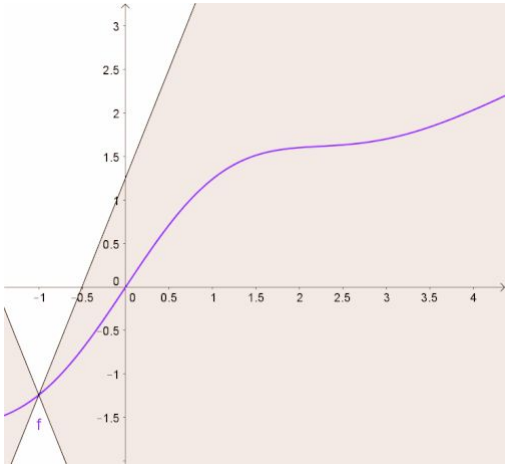
Functions can be shown to be **K-Lipschitz**

Definition 1 (Lipschitz continuous function). A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is K-Lipschitz for some $K > 0$, and some p-norm iff:

$$\forall x, y : \|f(x) - f(y)\|_p \leq K\|x - y\|_p$$

(Just think of absolute value for simplicity)

Wikipedia: there **exists a real number [K]** such that, for every pair of points on the graph of this function **[x,y]**, the absolute value of the slope of the line connecting them **[|f(x) - f(y)|]** is not greater than this real number



Geometric intuition: The function's derivative must be bounded by K

Proving convergence in # of steps given conditions…

Assume that $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable, and additionally,

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \text{ for any } x, y,$$

i.e., ∇f is Lipschitz continuous with constant $L \geq 0$.

- The **derivative of *f*** never changes by a rate more than L
- L >= 0

$$\eta \leq \frac{1}{L}$$

- Step size is < 1/L

Let's see what the proof says we can guarantee:

Think about what this means:

The gradient can't change by a large number (bounded by L)

When we estimate the gradient, our approximation won't degrade as quickly

Proving convergence in # of steps given conditions…

- *f* is convex and differentiable everywhere
  - Good properties for gradient descent!
- Gradient of *f* is L-Lipschitz
  - The **derivative of *f*** never changes by a rate more than L
  - L > 0
- Step size is < 1/L

Means that:

$$f(x_k) - f(x^*) \leq \frac{\|x_0 - x^*\|^2}{2\eta k}, \qquad k \in \mathbb{N}.$$

What do the variables mean? (Check the section notes)

After we take *k* steps…

The difference between the lowest possible loss and our current loss

Will be less than

Initial error of initialization point vs best parameter(s)...

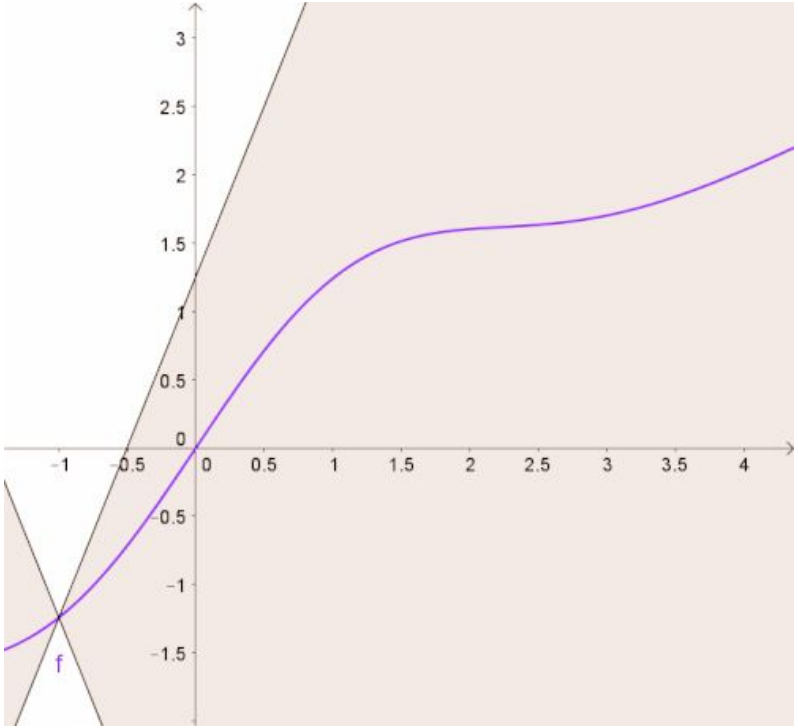
cut by a factor of *k*, and is cut further the higher the step size is

$$f(x_k) - f(x^*) \leq \frac{\|x_0 - x^*\|^2}{2\eta k}, \qquad k \in \mathbb{N}.$$

We will always get closer and closer, but now we can have this guarantee with a non-infinitesimal step size thanks to Lipschitz continuity!

Takeaways

- **Lipschitz continuity is a really good property to have** on the gradient of your loss function
  - Lower L/K-Lipschitz is better
- If you know this about your loss function, **you can make step size larger which leads to faster convergence!**



Questions/Chat Time!