Explanatory Notes for 6.390

Shaunticlair Ruiz (Current TA)

Fall 2022

Stochastic Gradient Descent

Another problem with gradient descent

One **advantage** we mentioned for gradient descent at the beginning of the chapter is that we can **stop early** if we think we're done, saving on **time**.

This is helpful for really **large** datasets, and it's also more **computationally manageable** than inverting a gigantic XX^T matrix.

But, we **haven't** taken **full advantage** of it: above, we used a **sum** to get our gradient - we're assuming we'll use all of our **data points**.

But, what if we don't want to have to use all of them at once? That might be **expensive**.

And in fact, there are **other** reasons not to: maybe the gradient will change directions after only a **small** distance.

Right now, we're getting **lots of data** before even taking a **single step**: if we start moving **immediately**, our program could **adapt** to the "terrain" more quickly!

A better way: stochastic GD

Instead, why wait until we have **added** up over all the data? We could just **compute** the gradient over **one** data point **at a time**. In fact, to be fair, we'll do it **randomly**.

To compensate, our steps will have to be **smaller**!

But wait, this **seems** like it would be **less** effective - after all, how much does **one** data point tell you?

Well, even if it isn't much, this isn't very **different** from adding them up all at **once**: in **theory**, taking lots of **little** steps should average out to the **same** information as if we do it all at once.

Definition 1

Stochastic Gradient Descent (SGD) is the process of applying **gradient descent** on **randomly** selected data points.

This should **average** out to being **similar** to regular (batch) gradient descent, but the **randomness** often lets it improve **faster** and **avoid** some common problems.

Stochastic is just a very mathematically precise word for "random".

There are more possible benefits, too: **randomly** choosing data points adds some **noise**, and random movement might be able to pull us out of local minima we don't want.

We mean "noise" in the signals sense: random **variation** in our data that **isn't** part of what we're trying to pay attention to: in this case, the **distribution**.

This sort of **noise** and **randomness** can make it hard for our model to **perfectly** fit the training data: this can reduce **overfitting**, too!

For example, it's hard to focus on the details of someone's eyelashes (unimportant details) if your vision is blurry.

Concept 2

There are many **benefits** to **SGD** (Stochastic Gradient Descent) over regular BGD (Batch Gradient Descent).

- SGD can sometimes **learn** a good model **without** using all of our **data**, which can **save us time** when data sets are **too large**.
  - It can also let us address problems **early** if the model **isn't** improving.
- The noise produced by the random sampling in SGD can sometimes help it **avoid local minima** that aren't very good models,
  - This is because the model might be moved in a **random direction** in **parameter space**, and randomly **pulled out** of that minimum, even if BGD would have gotten **stuck**.
- The noise also **reduces overfitting**, because it's **harder** for the model to **memorize** the exact details of the **distribution**.

Ensuring Convergence

How do we make sure that our SGD method converges? We need some kind of termination criteria. Thankfully, there's a useful theorem on the matter:

Theorem 3

SGD **converges** with *probability one* to the **optimal** Θ if

- f is convex

And our step size(learning rule) $\eta(t)$ follows these rules:

$$\sum_{t=1}^{\infty} \eta(t) = \infty \qquad \text{and} \qquad \sum_{t=1}^{\infty} \eta(t)^2 < \infty$$

Why these rules? Let's see:

- The **first** rule is for the **same** reason as for regular BGD: if it isn't **convex**, we can get stuck in **local minima**, or if it's **concave**, decrease **forever**.
- The **second** rule means that your steps need to add up to an **infinite distance**: this allows you to reach **any** possible point in your **parameter space**.
- The **third** one is a bit **trickier**, but basically means the steps need to get **smaller**, so we can approach the **minimum** (otherwise we might **diverge**!)

One option is $\eta(t) = 1/t$. But often, we use rules that **decrease** more **slowly**, so that it doesn't take as **long**.

But technically, we're no longer guaranteed convergence!