Explanatory Notes for 6.390

Shaunticlair Ruiz (Current TA)

Fall 2022

Computational Gradient

Sometimes, we **can't** easily find the **equation** for our gradient: maybe our loss isn't a simple **equation**, or we have some **other** kind of problem. So, rather than getting the **exact** gradient, we **approximate** it.

But how do we **approximate** the gradient? Well, first, we could **reference** how we approximate a **simple derivative**.

A derivative is just a 1-D gradient, after all!

The definition of the **derivative** can be gotten as

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1}$$

But, what if we can't take the **limit**? Or, we just don't **want** to?

We can **approximate** by taking h to be a small, **finite** number.

Instead of h, we'll call this δ.

Concept 1

When **approximating** the derivative, we can choose a **small** finite width to measure, called δ, so that

$$\frac{df}{dx} \approx \frac{f(x+\delta) - f(x)}{\delta}, \qquad \delta << 1 \tag{2}$$

So, let's **extend** that to the **gradient**:

$$\nabla_\theta J = \begin{bmatrix} \partial J / \partial \theta_1 \\ \partial J / \partial \theta_2 \\ \vdots \\ \partial J / \partial \theta_d \end{bmatrix} \tag{3}$$

Luckily, the **gradient** is just a bunch of derivatives **stacked** in a **vector**!

So, we can just **compute** each of them **separately**, and then put them together.

Let's show how we'd **write** that in **vector** form, for just one of them. We want something like

$$\underbrace{J'(\theta) \approx \frac{J(\theta+\delta) - f(\theta)}{\delta}}_{\text{Not correct, but closer}} \tag{4}$$

This isn't quite right, because a **scalar** δ would **add** to **every term**.

We **only** want to shift **one** variable at a time, so we can do a **simple** derivative.

Let's say we want $dJ/d\theta_1$. We would **only** want to add δ to θ_1: the other parameters are **unchanged**.

So, we **can't** add a **scalar**. Instead, we need a $(d \times 1)$ vector: one term to **separately** add to each θ_k term.

$$\Delta\theta = \begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \vdots \\ \Delta\theta_d \end{bmatrix} \tag{5}$$

We want most terms **unchanged**, so we'll **add 0** to each of them, and we'll add δ to the one term we want to **edit**.

$$\Delta\theta = \begin{bmatrix} \delta \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

We'll **create** one of these vectors for each **dimension**. We'll give them a special **name**: δ_k, for the k^{th} dimension.

$$\delta_1 = \begin{bmatrix} \delta \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \qquad \delta_2 = \begin{bmatrix} 0 \\ \delta \\ \vdots \\ 0 \\ 0 \end{bmatrix} \qquad \delta_{d-1} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \delta \\ 0 \end{bmatrix} \qquad \delta_d = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \delta \end{bmatrix} \tag{7}$$

Finally, we'll **divide** by δ. We have what we need for our full equation:

Key Equation 2

In order to **computationally find the gradient**, you need to find the **partial derivative** for each term θ_k.

$$\frac{dJ}{d\theta_k} \approx \frac{J(\theta + \delta_k) - f(\theta)}{\delta}$$

Where

- δ is a small positive number
- δ_k is the $(d \times 1)$ **column vector** with a δ in the k^{th} row, and a 0 in every other row.