Note this only contains major formulas and notes, the complete article could be found here - https://medium.com/@rishit.dagli/debugging-your-neural-nets-and-checking-your-gradients-f4d7f55da167

Debugging your Neural Nets and checking your Gradients

Rishit Dagli

April 2020

1 Introduction

This document contains a brief and all the major formulas discussed in my blog "Debugging your Neural Nets and checking your Gradients"

2 Double sided derivatives

A formula derived to compute double sided derivative

$$\frac{f(\theta+\epsilon)-f(\theta-\epsilon)}{2\cdot\epsilon}\approx g(\theta)$$

Working thee double sided derivative with example numbers

$$\frac{(1.01)^3-(0.99)^3}{2\cdot 0.01}=3.0001$$

Actual derivative for $f(\theta^3)$

$$g(\theta)=3\cdot\theta^2=3$$

One sided derivative

$$g(\theta)=3.0301$$

Comparing methods

1. One sided derivative
Approximation error = 0.0301

2. Double sided derivative
Approximation error = 0.0001

3 Going a bit deeper

$$f^{'} = \lim_{\epsilon \to 0} \frac{f(\theta + \epsilon) - f(\theta - \epsilon)}{2 \cdot \epsilon} \approx g(\theta)$$

iff $\lim_{\epsilon \to 0}$

if $\epsilon \neq 0$ then $error = O(\epsilon^2)$

Again ϵ is a very small number of course less than 1, so $\epsilon >> \epsilon^2$
The error function in case of double sided approach would be

$$error = O(\epsilon^2)$$

And for single sided approach it would be

$$error = O(\epsilon)$$

4 Gradient Checking

Concatenate all $w^{[i]}$ and $b^{[i]}$ into one single huge vector called θ
So, now our cost function

$$J(w^{[1]}, b^{[1]}, w^{[3]}...w^{[n]}, b^{[n]}) = J(\theta) \qquad \cdots\cdots(*)$$

Now you can also write

$$J(\theta) = J(\theta_1, \theta_2, \theta_3...\theta_n)$$

Example for single iteration of θ_5

$$d\theta^{[5]}_{approx.} = \frac{J(\theta_1, \theta_2...\theta_5 + \epsilon...\theta_n) - J(\theta_1, \theta_2...\theta_5 - \epsilon...\theta_n)}{2 \cdot \epsilon}$$

Considering the double sided discussion earlier

$$d\theta^{[5]}_{approx.} = \frac{J(\theta_1, \theta_2...\theta_5 + \epsilon...\theta_n) - J(\theta_1, \theta_2...\theta_5 - \epsilon...\theta_n)}{2 \cdot \epsilon} \approx d\theta^{[i]} = \frac{\partial J}{\partial \theta_5}$$

Having this clear to us we can now repeat the same process not just for but for all i such that $i \in (1, n)$. So a pseudo Python code for this would look something like this-

for i in range(1, n+1):

$$d\theta^{[i]}_{approx.} = \frac{J(\theta_1, \theta_2...\theta_i + \epsilon...\theta_n) - J(\theta_1, \theta_2...\theta_i - \epsilon...\theta_n)}{2 \cdot \epsilon}$$

5 Approaching an outcome

Computing similarity

$$\frac{||d\theta_{approx.} - d\theta||_2}{||d\theta_{approx.}||_2 + ||d\theta||_2}$$

Difference	Comments
<= 10^{-7}	Keep up the good work, you are doing just wonderful!
10^{-5}	Okay, Double check components of my vectors
	and check that none of the components are too large
=>10^{-3}	Something to worry about, there might be a bug

6 Implementing in practice