CS7643: Deep Learning
Spring 2022
Computational Graph Practice Set

1 Automatic Differentiation

1. In practice, writing the closed-form expression of the derivative of a loss function f w.r.t. the parameters of a deep neural network is hard (and mostly unnecessary) as f becomes complex. Instead, we define computation graphs and use the automatic differentiation algorithms (typically backpropagation) to compute gradients using the chain rule. For example, consider the expression

$$f(x, y) = (x + y)(y + 1) \tag{1}$$

Let's define intermediate variables a and b such that

$$a = x + y \tag{2}$$
$$b = y + 1 \tag{3}$$
$$f = a \times b \tag{4}$$

A computation graph for the "forward pass" through f is shown in Fig. 1.

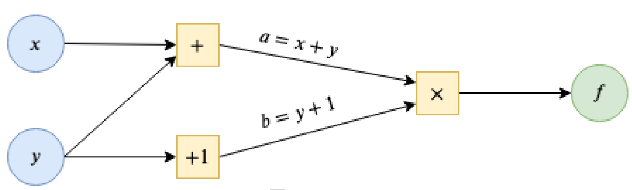

Figure 1

We can then work backwards and compute the derivative of f w.r.t. each intermediate variable ($\frac{\partial f}{\partial a}$ and $\frac{\partial f}{\partial b}$) and chain them together to get $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$.

Let $\sigma(\cdot)$ denote the standard sigmoid function. Now, for the following vector function:

$$f_1(w_1, w_2) = e^{e^{w_1} + e^{2w_2}} + \sigma(e^{w_1} + e^{2w_2}) \tag{5}$$
$$f_2(w_1, w_2) = w_1 w_2 + \max(w_1, w_2) \tag{6}$$

(a) Draw the computation graph. Compute the value of f at $\vec{w} = (1, -1)$.

(b) At this $\vec{w}$, compute the Jacobian $\frac{\partial \vec{f}}{\partial \vec{w}}$ using numerical differentiation (using $\Delta w = 0.01$).

(c) At this $\vec{w}$, compute the Jacobian using forward mode auto-differentiation.

(d) At this $\vec{w}$, compute the Jacobian using backward mode auto-differentiation.

(e) Don't you love that software exists to do this for us?