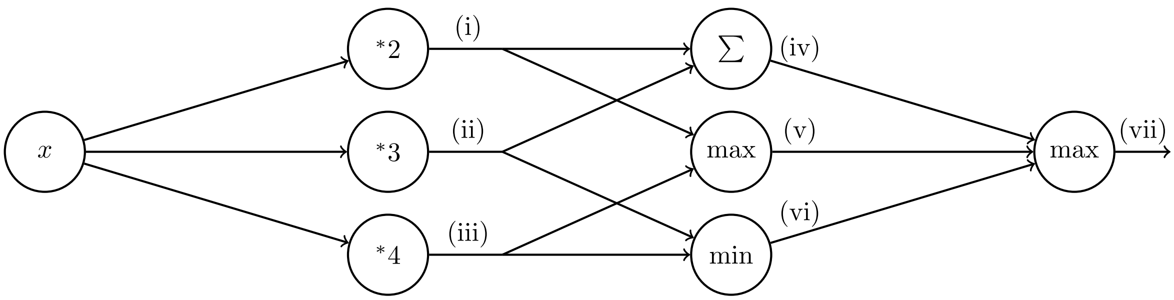
CS 188
Summer 2023

Discussion 7C

Q1. Deep Learning

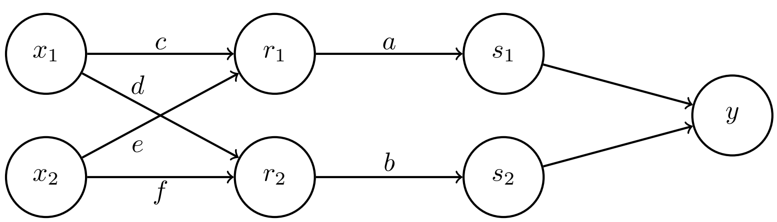
(a) Perform forward propagation on the neural network below for $x = 1$ by filling in the values in the table. Note that (i), . . . , (vii) are outputs after performing the appropriate operation as indicated in the node.

(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)



(b) **[Optional]** Below is a neural network with weights a, b, c, d, e, f. The inputs are x_1 and x_2.
The first hidden layer computes $r_1 = \max(c \cdot x_1 + e \cdot x_2, 0)$ and $r_2 = \max(d \cdot x_1 + f \cdot x_2, 0)$.
The second hidden layer computes $s_1 = \frac{1}{1+\exp(-a \cdot r_1)}$ and $s_2 = \frac{1}{1+\exp(-b \cdot r_2)}$.
The output layer computes $y = s_1 + s_2$. Note that the weights a, b, c, d, e, f are indicated along the edges of the neural network here.

Suppose the network has inputs $x_1 = 1, x_2 = -1$.
The weight values are $a = 1, b = 1, c = 4, d = 1, e = 2, f = 2$.
Forward propagation then computes $r_1 = 2, r_2 = 0, s_1 = 0.9, s_2 = 0.5, y = 1.4$. Note: some values are rounded.



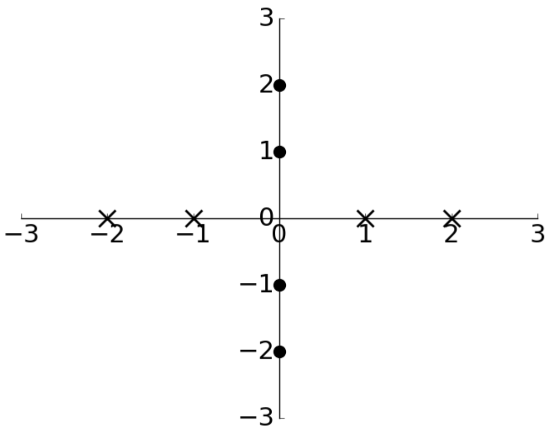
Using the values computed from forward propagation, use backpropagation to numerically calculate the following partial derivatives. Write your answers as a single number (not an expression). You do not need a calculator. Use scratch paper if needed.
Hint: For $g(z) = \frac{1}{1+\exp(-z)}$, the derivative is $\frac{\partial g}{\partial z} = g(z)(1 - g(z))$.

$\frac{\partial y}{\partial a}$	$\frac{\partial y}{\partial b}$	$\frac{\partial y}{\partial c}$	$\frac{\partial y}{\partial d}$	$\frac{\partial y}{\partial e}$	$\frac{\partial y}{\partial f}$

(c) Below are two plots with horizontal axis x_1 and vertical axis x_2 containing data labelled $\times$ and $\bullet$. For each plot, we wish to find a function $f(x_1, x_2)$ such that $f(x_1, x_2) \geq 0$ for all data labelled $\times$ and $f(x_1, x_2) < 0$ for all data labelled $\bullet$.

Below each plot is the function $f(x_1, x_2)$ for that specific plot. Complete the expressions such that all the data is labelled correctly. If not possible, mark "No valid combination".

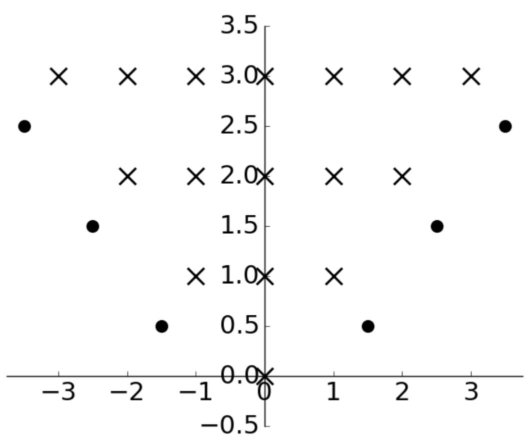
[subfigure]labelformat=empty



$f(x_1, x_2) = \max(\underline{\ \textbf{(i)}\ } + \underline{\textbf{(ii)}}, \underline{\textbf{(iii)}} + \underline{\textbf{(iv)}}) + \underline{\textbf{(v)}}$

(i) ○ x_1 ○ $-x_1$ ○ 0
(ii) ○ x_2 ○ $-x_2$ ○ 0
(iii) ○ x_1 ○ $-x_1$ ○ 0
(iv) ○ x_2 ○ $-x_2$ ○ 0
(v) ○ 1 ○ -1 ○ 0
○ No valid combination

[subfigure]labelformat=empty

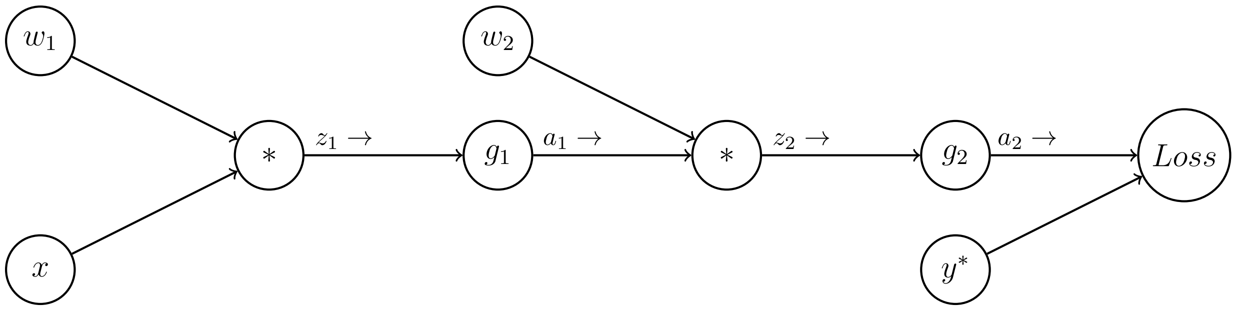


$f(x_1, x_2) = \underline{\textbf{(vi)}} - \max(\underline{\textbf{(vii)}} + \underline{\textbf{(viii)}}, \underline{\textbf{(ix)}} + \underline{\ \textbf{(x)}\ })$

(vi) ○ x_2 ○ $-x_2$ ○ 0
(vii) ○ x_1 ○ $-x_1$ ○ 0
(viii) ○ x_2 ○ $-x_2$ ○ 0
(ix) ○ x_1 ○ $-x_1$ ○ 0
(x) ○ x_2 ○ $-x_2$ ○ 0
○ No valid combination

2 Neural Nets

Consider the following computation graph for a simple neural network for binary classification. Here x is a single real-valued input feature with an associated class y^* (0 or 1). There are two weight parameters w_1 and w_2, and non-linearity functions g_1 and g_2 (to be defined later, below). The network will output a value a_2 between 0 and 1, representing the probability of being in class 1. We will be using a loss function *Loss* (to be defined later, below), to compare the prediction a_2 with the true class y^*.



1. Perform the forward pass on this network, writing the output values for each node z_1, a_1, z_2 and a_2 in terms of the node's input values:

2. Compute the loss $Loss(a_2, y^*)$ in terms of the input x, weights w_i, and activation functions g_i:

3. **[Optional]** Now we will work through parts of the backward pass, incrementally. Use the chain rule to derive $\frac{\partial Loss}{\partial w_2}$. Write your expression as a product of partial derivatives at each node: i.e. the partial derivative of the node's output with respect to its inputs. (Hint: the series of expressions you wrote in part 1 will be helpful; you may use any of those variables.)

4. **[Optional]** Suppose the loss function is quadratic, $Loss(a_2, y^*) = \frac{1}{2}(a_2 - y^*)^2$, and g_1 and g_2 are both sigmoid functions $g(z) = \frac{1}{1+e^{-z}}$ (note: it's typically better to use a different type of loss, *cross-entropy*, for classification problems, but we'll use this to make the math easier).

   Using the chain rule from Part 3, and the fact that $\frac{\partial g(z)}{\partial z} = g(z)(1 - g(z))$ for the sigmoid function, write $\frac{\partial Loss}{\partial w_2}$ in terms of the values from the forward pass, y^*, a_1, and a_2:

5. **[Optional]** Now use the chain rule to derive $\frac{\partial Loss}{\partial w_1}$ as a product of partial derivatives at each node used in the chain rule:

6. **[Optional]** Finally, write $\frac{\partial Loss}{\partial w_1}$ in terms of x, y^*, w_i, a_i, z_i:

7. **[Optional]** What is the gradient descent update for w_1 with step-size α in terms of the values computed above?