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.

<table>
<thead>
<tr>
<th></th>
<th>(i)</th>
<th>(ii)</th>
<th>(iii)</th>
<th>(iv)</th>
<th>(v)</th>
<th>(vi)</th>
<th>(vii)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>*2</td>
<td>*3</td>
<td>*4</td>
<td>∑</td>
<td>max</td>
<td>min</td>
<td>max</td>
</tr>
</tbody>
</table>

(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 $\frac{\partial g}{\partial z} = g(z)(1-g(z))$.

<table>
<thead>
<tr>
<th></th>
<th>$\frac{\partial y}{\partial a}$</th>
<th>$\frac{\partial y}{\partial b}$</th>
<th>$\frac{\partial y}{\partial c}$</th>
<th>$\frac{\partial y}{\partial d}$</th>
<th>$\frac{\partial y}{\partial e}$</th>
<th>$\frac{\partial y}{\partial f}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(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”.

\[
f(x_1, x_2) = \max(\text{(i)} + \text{(ii)}, \text{(iii)} + \text{(iv)}) + \text{(v)}
\]

\[
\begin{array}{c|c|c|c}
\text{(i)} & x_1 & -x_1 & 0 \\
\text{(ii)} & x_2 & -x_2 & 0 \\
\text{(iii)} & x_1 & -x_1 & 0 \\
\text{(iv)} & x_2 & -x_2 & 0 \\
\text{(v)} & 1 & -1 & 0 \\
\end{array}
\]

\( \circ \) No valid combination

\[
f(x_1, x_2) = (\text{vi}) - \max(\text{(vii)} + \text{(viii)}, \text{(ix)} + \text{(x)})
\]

\[
\begin{array}{c|c|c|c}
\text{(vi)} & x_2 & -x_2 & 0 \\
\text{(vii)} & x_1 & -x_1 & 0 \\
\text{(viii)} & x_2 & -x_2 & 0 \\
\text{(ix)} & x_1 & -x_1 & 0 \\
\text{(x)} & x_2 & -x_2 & 0 \\
\end{array}
\]

\( \circ \) 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 \( y^\ast \) (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 \( y^\ast \).
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 $x, y^*, w_1, a_i, z_i$:

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