CSC321H5 Homework 3.

Deadline: Thursday, Feb. 6, by 9pm

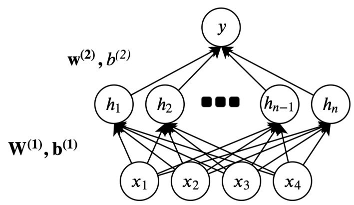
Submission: You must submit your solutions as a PDF file through MarkUs. You can produce the file however you like (e.g. LaTeX, Microsoft Word, scanner), as long as it is readable.

Late Submission: Please see the syllabus for the late submission criteria.

Question 1. Hand-Coding a Neural Network

In this problem, we'll find parameters for a multilayer perceptron to check whether four inputs x_1, x_2, x_3, x_4, where $x_i \in \mathbb{R}$, are unique. That is, our network should output 1 if no two input are equal with $x_i \neq x_j$ for $i \neq j$.

We will use a two-layer neural network like this one:



All of the hidden units will use the **impulse activation function**, defined as follows:

$$\phi(x) = \begin{cases} 1, & \text{if } x = 0 \\ 0, & \text{otherwise} \end{cases}$$

Part (a) – 1 pts

We will use $n = 6$ hidden units in our hand-coded neural network. Explain why we will not require more than 6 hidden units.

Part (b) – 2 pts

What are the *shapes* of each of the following quantities?

- $\mathbf{W}^{(1)}$ – the weight matrix containing the weights of the first layer of the neural network
- $\mathbf{b}^{(1)}$ – the bias vector containing the baises to the hidden layers
- $\mathbf{W}^{(2)}$ – the weights containing the weights of the second layer of the neural network
- $\mathbf{b}^{(2)}$ – the bias containing the baises to the output layer

Part (c) – 4 pts

Hand-pick a set of weights and biases so that the network correctly implements the desired functionality. Your answer should include the values of $\mathbf{W}^{(1)}$, $\mathbf{b}^{(1)}$, $\mathbf{w}^{(2)}$, and $b^{(2)}$.

Part (d) – 3 pts

Show that your network correctly classifies the below three sets of inputs:

- Input 1: $x_1 = 2$, $x_2 = 2$, $x_3 = 1$, $x_4 = 1$
- Input 2: $x_1 = 1$, $x_2 = 2$, $x_3 = 3$, $x_4 = 1$
- Input 3: $x_1 = 1$, $x_2 = 0$, $x_3 = 0$, $x_4 = -1$

Question 2. Softmax Activation.

In lecture 3, we defined the softmax activation as follows:

$$y_k = \text{softmax}(z_1, ..., z_K)_k = \frac{e^{z_k}}{\sum_{m=1}^{K} e^{z_m}}$$

Part (a) – 4 pts

Compute the derivative $\frac{\partial y_k}{\partial z_k}$

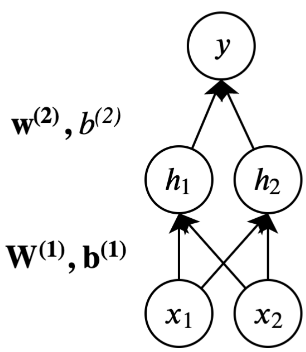
Part (b) – 4 pts

Compute the derivative $\frac{\partial y_k}{\partial z_j}$, where $k \neq j$.

Question 3. XOR Problem

In this question, we'll train a neural network using backpropagation **by hand**. Don't worry, we will only train the network for a single iteration.

Our neural network will have two input features x_1 and x_2, and two hidden units h_1 and h_2. The hidden units h_1 and h_2 will use the ReLU activation, and for final prediction y we will use the sigmoid activation to obtain a prediction between 0 and 1.



Our training example will look like this:

- $\mathbf{x^{(1)}} = [\mathbf{0}, \mathbf{0}]^\mathbf{T}$, $t^{(1)} = 0$
- $\mathbf{x^{(2)}} = [\mathbf{0}, \mathbf{1}]^\mathbf{T}$, $t^{(2)} = 1$
- $\mathbf{x^{(3)}} = [\mathbf{1}, \mathbf{0}]^\mathbf{T}$, $t^{(3)} = 1$
- $\mathbf{x^{(4)}} = [\mathbf{1}, \mathbf{1}]^\mathbf{T}$, $t^{(4)} = 0$

Part (a) – 4 pts

Our initial weights and biases will look like this:

$$\mathbf{W}^{(1)} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, \mathbf{b}^{(1)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mathbf{W}^{(2)} = \begin{bmatrix} 1 & 1 \end{bmatrix}, \mathbf{b}^{(2)} = \begin{bmatrix} 0 \end{bmatrix}$$

Compute the forward pass for each of the four examples. That is, for each $\mathbf{x}^{(n)}$, compute h_1, h_2 and the prediction y. Use the notation m_1 and m_2 to denote the pre-activations values of the hidden states, so that $h_1 = \text{relu}(m_1)$ and $h_2 = \text{relu}(m_2)$. Use z to represent the logit of the prediction, so that $y = \sigma(z)$.

You may also use the vector representations $\mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$, $\mathbf{m} = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix}$, but always start with the scalar representations first.

Part (b) – 4 pts

Suppose that we are using a mini-batch size of 1, and that we are using $\mathcal{L}(\mathbf{x}^{(4)}, t^{(4)})$ to estimate the cost function. Use backpropagation to compute the below terms. For the vector and matrix derivatives, **always start with the scalar derivatives of its elements** before putting them together in vector or matrix.

Update (Feb 1): Please use the cross-entropy loss.

$$
\begin{aligned}
\overline{\mathcal{L}} &= 1 \\
\overline{y} &= \\
\overline{z} &= \\
\overline{w_k^{(2)}} &= \\
\overline{b^{(2)}} &= \\
\overline{h_k} &= \\
\overline{m_k} &= \\
\overline{W_{jk}^{(1)}} &= \\
\overline{\mathbf{b}_k^{(1)}} &=
\end{aligned}
$$

Part (c) – 2 pts

Assuming a learning rate of 0.5, perform one iteration of weight update based on the values you obtained from part (b).