# Homework 1

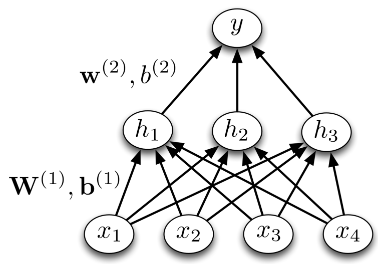
**Deadline:** Thursday, Jan. 24, at 11:59pm.

**Submission:** You must submit your solutions as a PDF file through MarkUs[1]. You can produce the file however you like (e.g. LaTeX, Microsoft Word, scanner), as long as it is readable.

**Late Submission:** MarkUs will remain open until 3 days after the deadline, after which no late submissions will be accepted.

Weekly homeworks are individual work. See the Course Information handout[2] for detailed policies.

1. **Hard-Coding a Network.** **[2pts]** In this problem, you need to find a set of weights and biases for a multilayer perceptron which determines if a list of length 4 is in sorted order. More specifically, you receive four inputs $x_1, \ldots, x_4$, where $x_i \in \mathbb{R}$, and the network must output 1 if $x_1 < x_2 < x_3 < x_4$, and 0 otherwise. You will use the following architecture:




All of the hidden units and the output unit use a hard threshold activation function:

$$\phi(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Please give a set of weights and biases for the network which correctly implements this function (including cases where some of the inputs are equal). Your answer should include:

- A $3 \times 4$ weight matrix $\mathbf{W}^{(1)}$ for the hidden layer
- A 3-dimensional vector of biases $\mathbf{b}^{(1)}$ for the hidden layer
- A 3-dimensional weight vector $\mathbf{w}^{(2)}$ for the output layer
- A scalar bias $b^{(2)}$ for the output layer

You do not need to show your work.

2. **Backprop.** Consider a neural network with $N$ input units, $N$ output units, and $K$ hidden units. The activations are computed as follows:

$$\begin{aligned} \mathbf{z} &= \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \\ \mathbf{h} &= \sigma(\mathbf{z}) \\ \mathbf{y} &= \mathbf{x} + \mathbf{W}^{(2)}\mathbf{h} + \mathbf{b}^{(2)}, \end{aligned}$$

[1] https://markus.teach.cs.toronto.edu/csc421-2019-01

[2] http://www.cs.toronto.edu/~rgrosse/courses/csc421_2019/syllabus.pdf

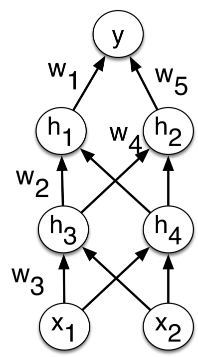
where $\sigma$ denotes the logistic function, applied elementwise. The cost will involve both $\mathbf{h}$ and $\mathbf{y}$:

$$\mathcal{J} = \mathcal{R} + \mathcal{S}$$
$$\mathcal{R} = \mathbf{r}^\top \mathbf{h}$$
$$\mathcal{S} = \frac{1}{2}\|\mathbf{y} - \mathbf{s}\|^2$$

for given vectors $\mathbf{r}$ and $\mathbf{s}$.

- **[1pt]** Draw the computation graph relating $\mathbf{x}$, $\mathbf{z}$, $\mathbf{h}$, $\mathbf{y}$, $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{J}$.
- **[3pts]** Derive the backprop equations for computing $\overline{\mathbf{x}} = \partial\mathcal{J}/\partial\mathbf{x}$. You may use $\sigma'$ to denote the derivative of the logistic function (so you don't need to write it out explicitly).

3. **Sparsifying Activation Function.** **[4pts]** One of the interesting features of the ReLU activation function is that it sparsifies the activations and the derivatives, i.e. sets a large fraction of the values to zero for any given input vector. Consider the following network:




Note that each $w_i$ refers to the weight on a *single* connection, not the whole layer. Suppose we are trying to minimize a loss function $\mathcal{L}$ which depends only on the activation of the output unit $y$. (For instance, $\mathcal{L}$ could be the squared error loss $\frac{1}{2}(y-t)^2$.) Suppose the unit $h_1$ receives an input of -1 on a particular training case, so the ReLU evaluates to 0. Based only on this information, which of the weight derivatives

$$\frac{\partial\mathcal{L}}{\partial w_1}, \frac{\partial\mathcal{L}}{\partial w_2}, \frac{\partial\mathcal{L}}{\partial w_3}$$

are **guaranteed** to be 0 for this training case? Write YES or NO for each. Justify your answers.