ECE/CS 559: Neural Networks **Fall 2024**

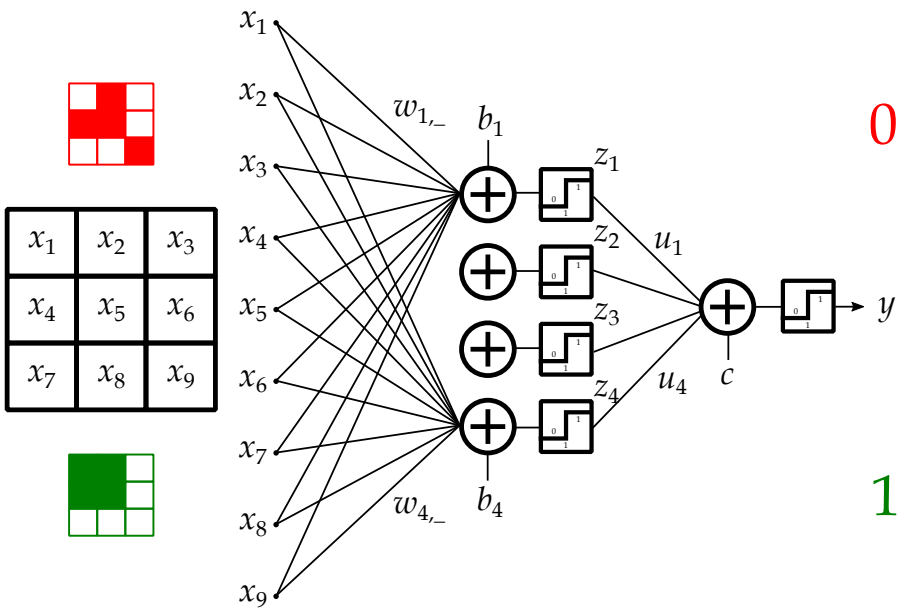
Homework 1

Due: **Tuesday** September 3, 2024 (by 9pm, on Gradescope)

Note: *You may discuss these problems in groups. However, you must write up your own solutions and mention the names of the people in your group. Also, please do mention any solution manuals, online material, or other sources you used in a major way.*

1. **[Logic and Neural Networks]**

   You have a 3×3 binary image (each pixel can be either 0 or 1). You want to build a detector that will output 1 if and only if there's a 2×2 square of 1's in this image. Consider the following *architecture*.



   Let the *inputs* of this neural network be the pixels of the image, labeled $x_1, \cdots, x_9$. These inputs are fed into 4 linear units, which are followed by a step nonlinearity. More precisely, for $j = 1, 2, 3, 4$ we get:

$$z_j = \text{step}_1\left(b_j + \sum_{i=1}^{9} w_{j,i} x_i\right), \text{ where } \text{step}_1(t) := \begin{cases} 0 & ; \quad t < 1 \\ 1 & ; \quad t \geq 1 \end{cases}$$

   These values, z_1, z_2, z_3 and z_4 are not output directly, which is why they are called *hidden*. The equations producing them is the first *layer* of this neural network.

Instead, these hidden values are fed into a similar linear unit followed by a step nonlinearity, to produce the *output*, as follows:

$$y = \mathsf{step}_1\left(c + \sum_{j=1}^{4} u_j z_j\right).$$

This is the second (and final) *layer*. Our goal is to adjust the *weights* $w_{j,i}$ (there are $4 \times 9 = 36$ of these numbers) and u_j (there are 4 of them), as well as the *biases* b_j (there are 4 of them) and c (just 1 number), such that the output detects 2×2 squares correctly ($y = 1$ if there's such a square, 0 if not.)

(a) Show that if, for some j, $w_{j,i}$ are all 1 and $b_j = 0$, then that z_j is the *logical OR* of all the x_i.

(b) Show that in (a), if for some i's you change $w_{j,i}$ to -1 and set b_j to the number of such changed i, then z_j is the logical OR of all the x_i, but *inverted* wherever i is changed.

(c) Use this analogy with logic to figure out how to adjust all 36 $w_{j,i}$, 4 b_j, 4 u_j, and 1 c, to produce the desired output correctly. Explain how you obtained these numbers. (Express $w_{j,i}$ in a 4×9 *matrix*.) [Hint: Break down the detection into four cases, then use DeMorgan's law to specialize each z_j in the first layer to one case, and then use the second layer to combine all four cases.]

2. **[Calculus and Neural Networks]** Let's say we have a complicated function $g(x)$. We would like to approximate it using another function from a family of functions $f(x, w)$, where by changing w we change which element in this family we pick. w is called a *parameter*. To judge how good the approximation is, we need to compare $g(x)$ and $f(x, w)$, using a loss function $\ell(f, g)$. Instead of making this comparison at every x, we typically summarize it by looking at the average loss across all x's, $L = \int \ell(f(x, w), g(x))\mathrm{d}x$. How do we find w to make L as small as possible? Doing this by hand (see Q1) could be tedious. As long as the functions are nice and differentiable, we can instead use the machinery of calculus and optimization to turn the 'knob' of w, until f nicely matches g.

Let's make this concrete. Let $g(x) = x^3$ and let $f(x, w) = wx$ be the linear family. That is, we are trying to approximate a cubic function with a linear function. Let's use the squared loss $\ell(f, g) = (f - g)^2$ to measure the loss. Let's say we only care about the approximation for $x \in [-1, 1]$, which we can take as the limits of L's integral:

$$L(w) = \int_{-1}^{1} \ell(f(x, w), g(x))\mathrm{d}x = \int_{-1}^{1} (xw - x^3)^2 \mathrm{d}x$$

By optimality, we want to find the value of w where $\frac{\mathrm{d}L}{\mathrm{d}w} = 0$. We need to calculate this derivative. By Fubini's theorem, we have:

$$\frac{\mathrm{d}L}{\mathrm{d}w} = \int_{-1}^{1} \frac{\partial}{\partial w} \ell(f(x, w), g(x)) \mathrm{d}x$$

All throughout, note that the inputs x and and target outputs $g(x)$ are just fixed constants that are not part of the differentiation.

(a) Write the chain rule, to see that to calculate $\frac{\partial}{\partial w}\ell$ we need to go *backward*: first calculate the derivative of $\ell(f, g)$ with respect to f, and then combine it with the derivative of $f(x, w)$ with respect to w. (The reason this is called *backward*, is because if the chain were longer, we would complete calculating the gradients of the parameters in each function from the last to the first.)

(b) Calculate those derivatives explicitly, combine them, and integrate them to get $\frac{\mathrm{d}L}{\mathrm{d}w}$. Set this to 0, to find the optimal parameter w.

(c) Plot $g(x)$ and $f(x, w)$ with the w from (b) on the same plot over the range $x \in [-1, 1]$, to see how well the w that you found works. (You can try other w's too, to qualitatively check that yours is better.)