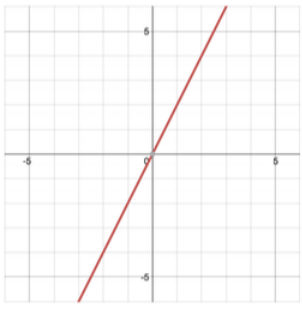
CS 188
Summer 2020

Introduction to
Artificial Intelligence

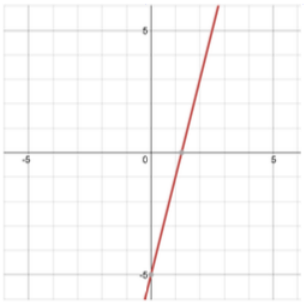
Section 12
Neural Networks

1 Neural Network Representations

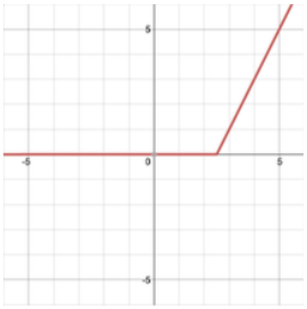
You are given a number of functions (a-h) of a single variable, x, which are graphed below. The computation graphs on the following pages will start off simple and get more complex, building up to neural networks. For each computation graph, indicate which of the functions below they are able to represent.



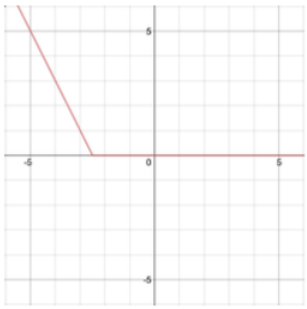
(a) $2x$



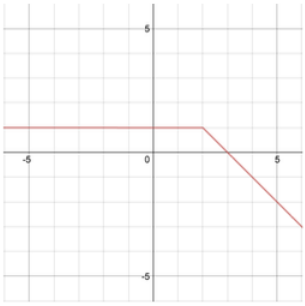
(b) $4x - 5$



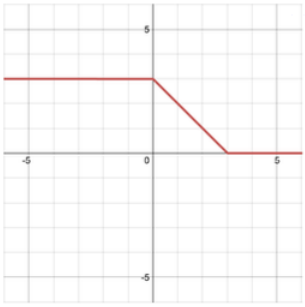
(c) $\begin{cases} 2x - 5 & x \geq 2.5 \\ 0 & x < 2.5 \end{cases}$



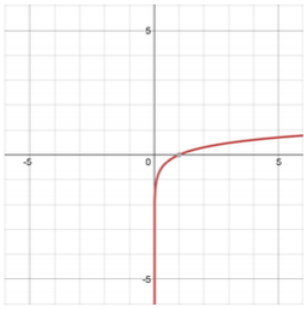
(d) $\begin{cases} -2x - 5 & x \leq -2.5 \\ 0 & x > -2.5 \end{cases}$



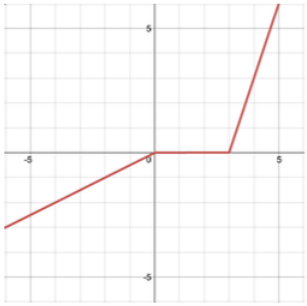
(e) $\begin{cases} -x + 3 & x \geq 2 \\ 1 & x < 2 \end{cases}$



(f) $\begin{cases} 3 & x \leq 0 \\ 3 - x & 0 < x \leq 3 \\ 0 & x > 3 \end{cases}$

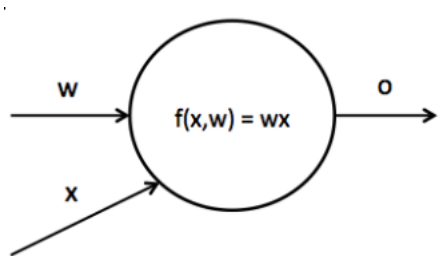


(g) $\log(x)$

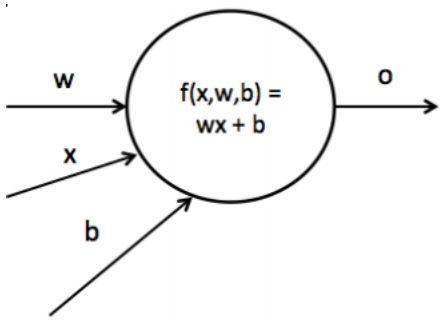


(h) $\begin{cases} 0.5x & x \leq 0 \\ 0 & 0 < x \leq 3 \\ 3x - 9 & x > 3 \end{cases}$

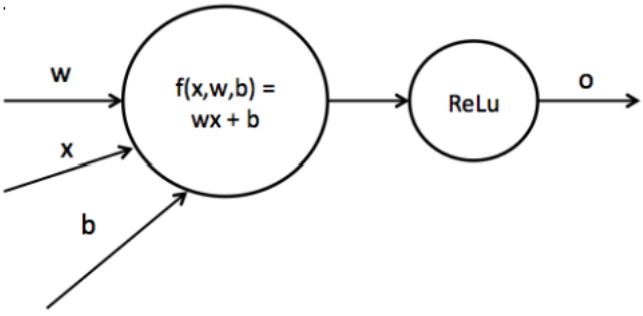
1. Consider the following computation graph, computing a linear transformation with scalar input x, weight w, and output o, such that $o = wx$. Which of the funcions can be represented by this graph? For the options which can, write out the appropriate value of w.



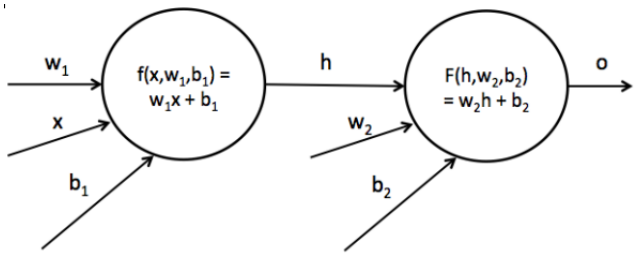
2. Now we introduce a bias term b into the graph, such that $o = wx + b$ (this is known as an *affine* function). Which of the functions can be represented by this network? For the options which can, write out an appropriate value of w, b.



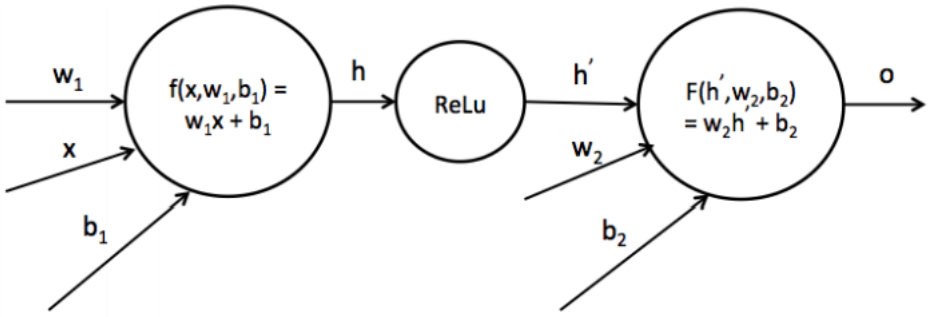
3. We can introduce a non-linearity into the network as indicated below. We use the ReLU non-linearity, which has the form $ReLU(x) = \max(0, x)$. Now which of the functions can be represented by this neural network with weight w and bias b? For the options which can, write out an appropriate value of w, b.



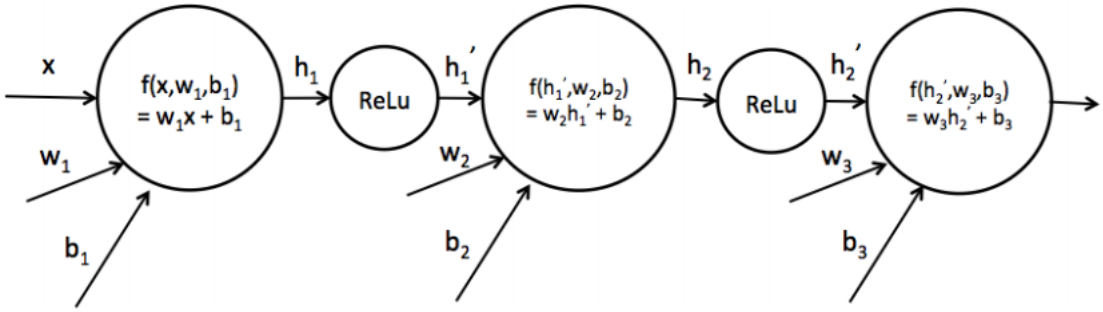
4. Now we consider neural networks with multiple affine transformations, as indicated below. We now have two sets of weights and biases w_1, b_1 and w_2, b_2. We denote the result of the first transformation h such that $h = w_1 x + b_1$, and $o = w_2 h + b_2$. Which of the functions can be represented by this network? For the options which can, write out appropriate values of w_1, w_2, b_1, b_2.



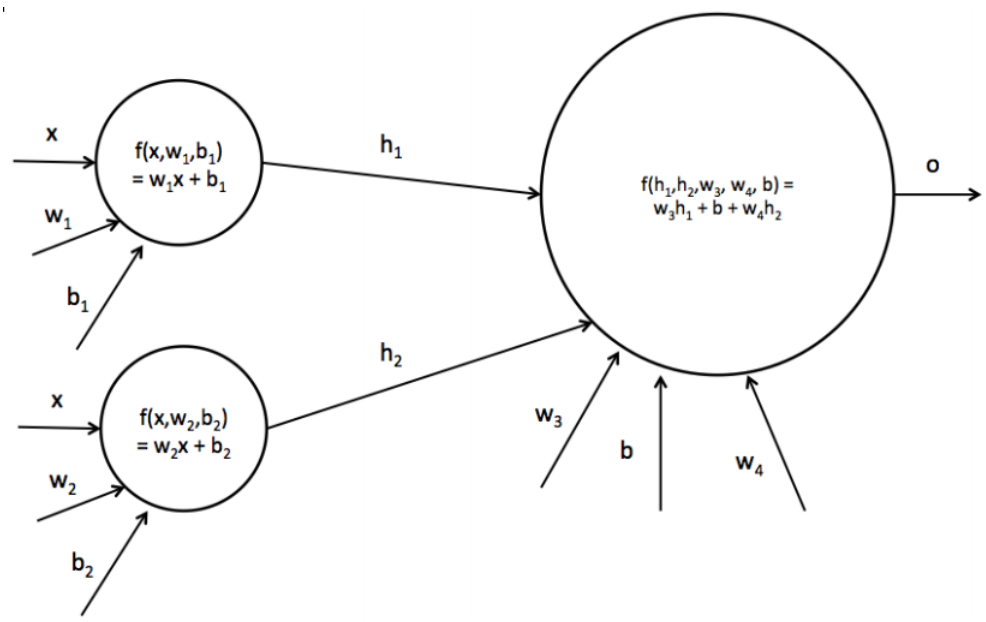
5. Next we add a ReLU non-linearity to the network after the first affine transformation, creating a hidden layer. Which of the functions can be represented by this network? For the options which can, write out appropriate values of w_1, w_2, b_1, b_2.



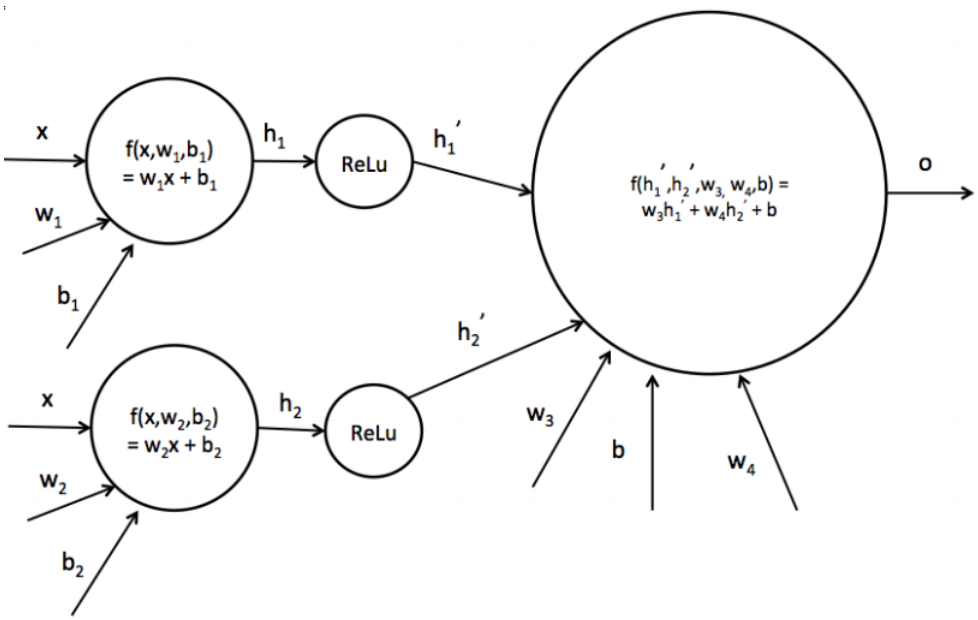
6. Now we add another hidden layer to the network, as indicated below. Which of the functions can be represented by this network?



7. We'd like to consider using a neural net with just one hidden layer, but have it be larger – a hidden layer of size 2. Let's first consider using just two affine functions, with no nonlinearity in between. Which of the functions can be represented by this network?



8. Now we'll add a non-linearity between the two affine layers, to produce the neural network below with a hidden layer of size 2. Which of the functions can be represented by this network?



2 Perceptron → Neural Nets

Instead of the standard perceptron algorithm, we decide to treat the perceptron as a single node neural network and update the weights using gradient-based optimization.

In lecture, we covered maximizing likelihood using gradient ascent. We can also choose to **minimize** a loss function that calculates the distance between a prediction and the correct label. The loss function for one data point is $Loss(y, y^*) = \frac{1}{2}(y - y^*)^2$, where y^* is the training label for a given point and y is the output of our single node network for that point.

We will compute a score $z = w_1 x_1 + w_2 x_2$, and then predict the output using an activation function g: $y = g(z)$.

1. Given a general activation function $g(z)$ and its derivative $g'(z)$, what is the derivative of the loss function with respect to w_1 in terms of $g, g', y^*, x_1, x_2, w_1$, and w_2?

   $$\frac{\partial Loss}{\partial w_1} =$$

2. We wish to *minimize* the loss, so we will use gradient *descent* (not gradient ascent). What is the update equation for weight w_i given $\frac{\partial Loss}{\partial w_i}$ and learning rate α?

   $$w_i \leftarrow$$

3. For this question, the specific activation function that we will use is

   $$g(z) = 1 \text{ if } z \geq 0 \text{ , or } -1 \text{ if } z < 0$$

   Use gradient descent to update the weights for a single data point. With initial weights of $w_1 = 2$ and $w_2 = -2$, what are the updated weights after processing the data point $(x_1, x_2) = (-1, 2)$, $y^* = 1$?

4. What is the most critical problem with this gradient descent training process with that activation function?