CS 188
Summer 2023

Discussion 7A Solutions

1 Perceptron

We would like to use a perceptron to train a classifier with 2 features per point and labels +1 or −1. Consider the following labeled training data:

Features (x_1, x_2)	Label y^*
$(-1, 2)$	1
$(3, -1)$	-1
$(1, 2)$	-1
$(3, 1)$	1

1. Our two perceptron weights have been initialized to $w_1 = 2$ and $w_2 = -2$. After processing the first point with the perceptron algorithm, what will be the updated values for these weights?

   For the first point, $y = g(w_1x_1 + w_2x_2) = g(2 \cdot -1 + -2 \cdot 2) = g(-5) = -1$, which is incorrectly classified. To updated the weights, we add the first data point: $w_1 = 2 + (-1) = 1$ and $w_2 = -2 + 2 = 0$.

2. After how many steps will the perceptron algorithm converge? Write "never" if it will never converge.

   Note: one step means processing one point. Points are processed in order and then repeated, until convergence.

   The data is not separable, so it will never converge.

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 descent on the loss function.

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 $z = w_1x_1 + w_2x_2$, 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?

$$
\begin{aligned}
\frac{\partial Loss}{\partial w_1} &= \frac{\partial}{\partial w_1}\frac{1}{2}(g(w_1x_1 + w_2x_2) - y^*)^2 \\
&= (g(w_1x_1 + w_2x_2) - y^*) * \frac{\partial}{\partial w_1} g(w_1x_1 + w_2x_2) \\
&= (g(w_1x_1 + w_2x_2) - y^*) * g'(w_1x_1 + w_2x_2) * \frac{\partial}{\partial w_1}(w_1x_1 + w_2x_2) \\
&= (g(w_1x_1 + w_2x_2) - y^*) * g'(w_1x_1 + w_2x_2) * x_1
\end{aligned}
$$

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

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

Given the gradient descent equation $w_i \leftarrow w_i - \alpha \frac{\partial Loss}{\partial w_1}$, 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 first point?

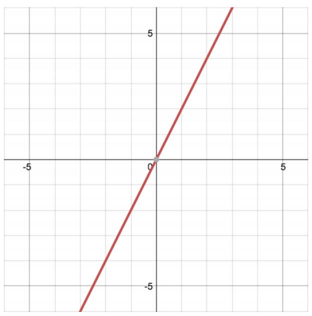
Because the derivative of g is always zero, $g'(z) = 0$ (although it has two pieces, both pieces are constant and so have no slope), $\frac{\partial Loss}{\partial w_1}$ will be zero, and so the weights will stay $w_1 = 2$ and $w_2 = -2$.

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

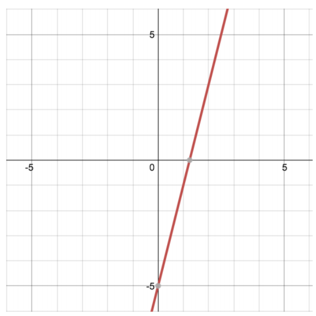
   The gradient of that activation function is zero, so the weights will not update.

2 Neural Network Representations

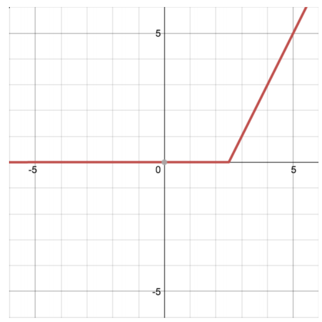
You are given a number of functions (a-h) of a single variable, x, which are graphed below. The computation graphs on the following pages will start off simple and get more complex, building up to neural networks. For each computation graph, indicate which of the functions below they are able to represent.



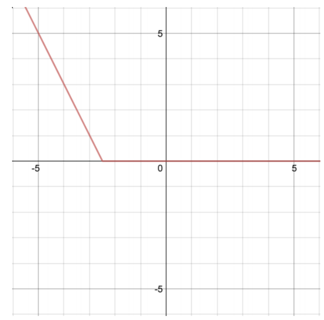
(a) $2x$



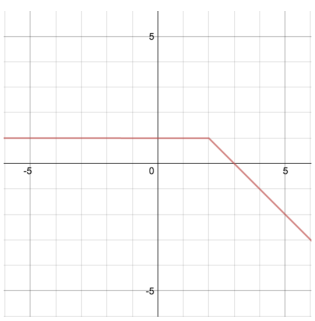
(b) $4x - 5$



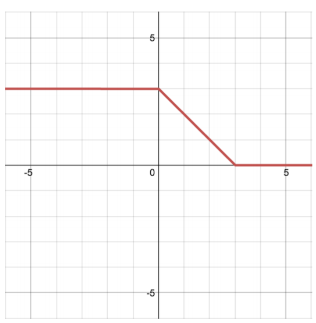
(c) $\begin{cases} 2x - 5 & x \geq 2.5 \\ 0 & x < 2.5 \end{cases}$



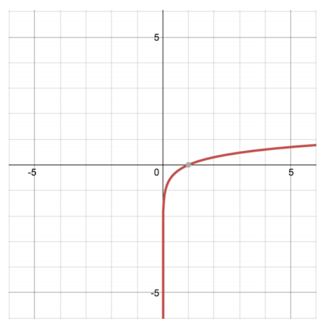
(d) $\begin{cases} -2x - 5 & x \leq -2.5 \\ 0 & x > -2.5 \end{cases}$



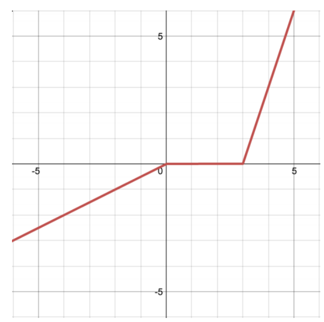
(e) $\begin{cases} -x + 3 & x \geq 2 \\ 1 & x < 2 \end{cases}$



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



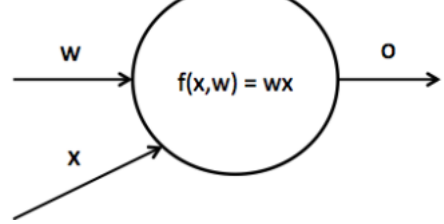
(g) $\log(x)$



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

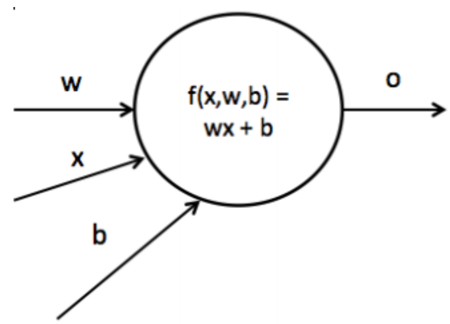
For each of the following computation graphs, determine which functions can be represented by the graph. In parts 1-5, write out the appropriate values of all w's and b's for each function that can be represented.

1. Linear Transformation



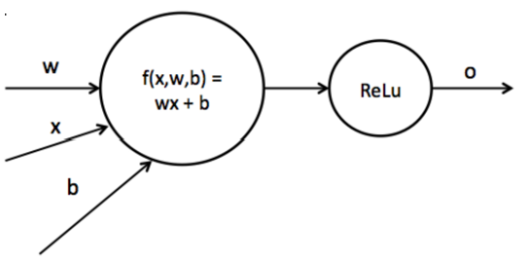
This graph can only represent (a), with $w = 2$. Since there is no bias term, the line must pass through the origin.

2. Linear plus Bias (aka affine transformation)



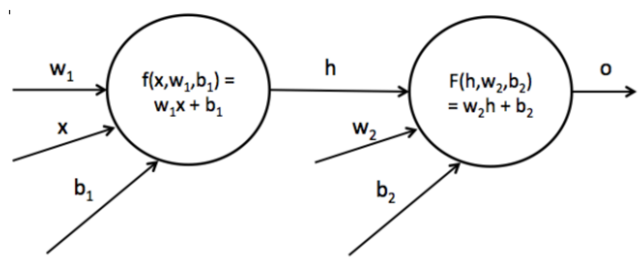
(a) with $w = 2$ and $b = 0$, and (b) with $w = 4$ and $b = -5$

3. Nonlinearity after Linear layer



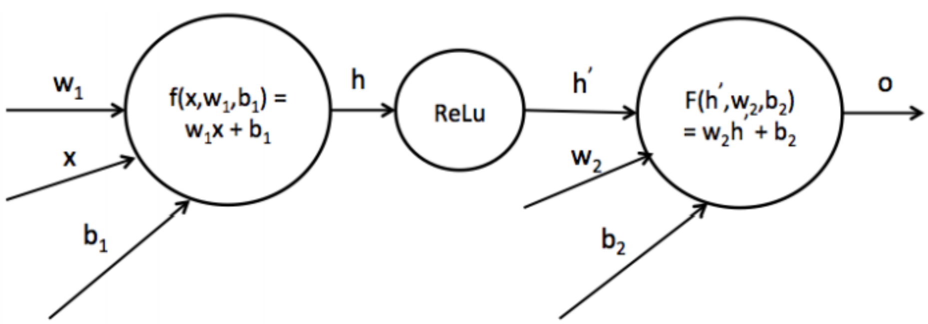
With the output coming directly from the ReLU, this cannot produce any values less than zero. It can produce (c) with $w = 2$ and $b = -5$, and (d) with $w = -2$ and $b = -5$

4. Composition of Affine layers



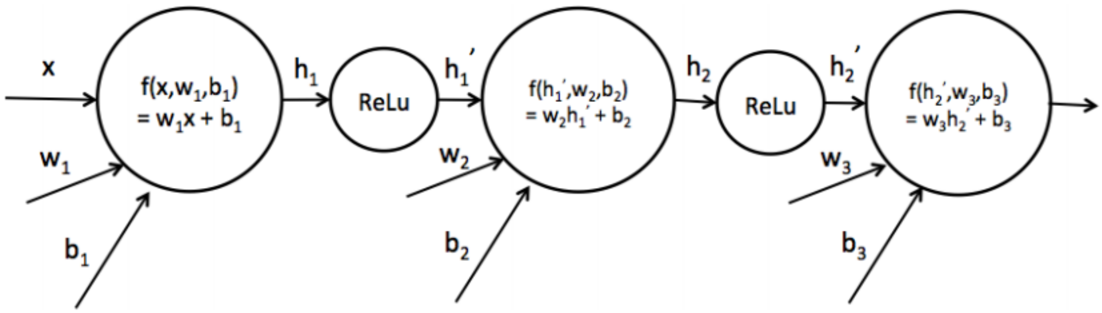
Applying multiple affine transformations (with no non-linearity in between) is not any more powerful than a single affine function: $w_2(w_1x + b_1) + b_2 = w_2w_1x + w_2b_1 + b_2$, so this is just a affine function with different coefficients. The functions we can represent are the same as in 1, if we choose $w_1 = w, w_2 = 1, b_1 = 0, b_2 = b$: (a) with $w_1 = 2, w_2 = 1, b_1 = 0, b_2 = 0$, and (b) with $w_1 = 4, w_2 = 1, b_1 = 0, b_2 = -5$.

5. Two Affine layers with nonlinearity in between (hidden layer)



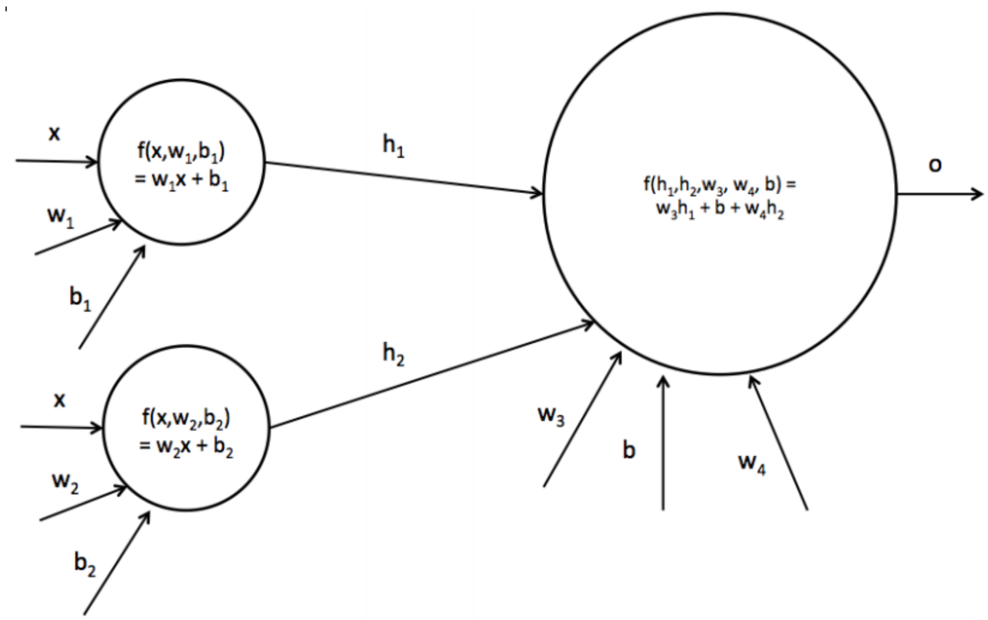
(c), (d), and (e). The affine transformation after the ReLU is capable of stretching (or flipping) and shifting the ReLU output in the vertical dimension. The parameters to produce these are:
(c) with $w_1 = 2, b_1 = -5, w_2 = 1, b_2 = 0$, (d) with $w_1 = -2, b_1 = -5, w_2 = 1, b_2 = 0$, and (e) with $w_1 = 1, b_1 = -2, w_2 = -1, b_2 = 1$

6. Add another hidden layer



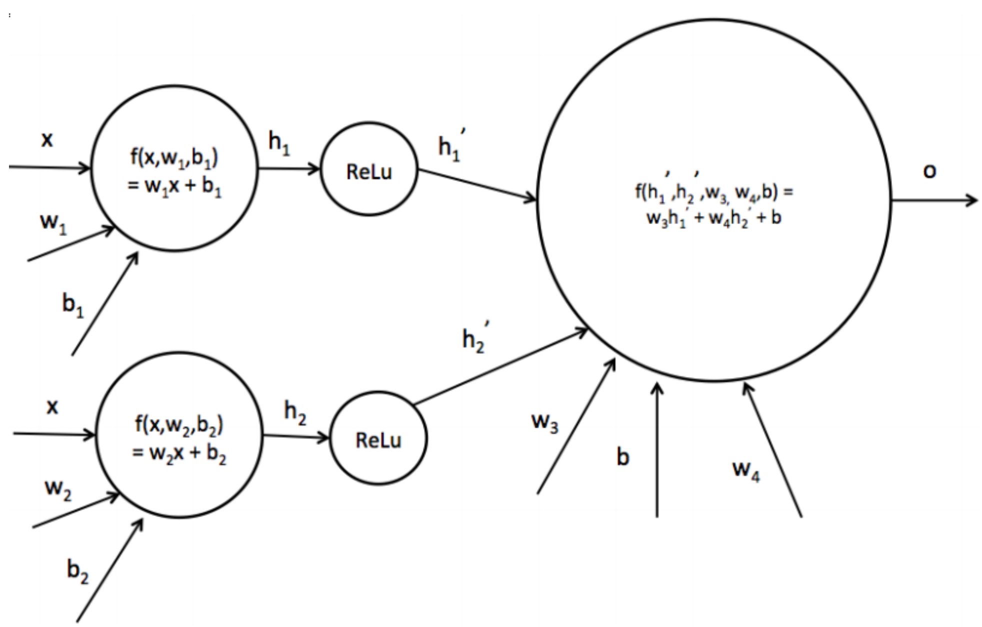
(c), (d), (e), and (f). The network can represent all the same functions as Q5 (because note that we could have $w_2 = 1$ and $b_2 = 0$). In addition it can represent (f): the first ReLU can produce the first flat segment, the affine transformation can flip and shift the resulting curve, and then the second ReLU can produce the second flat segment (with the final affine layer not doing anything). Note that (h) cannot be produced since its line has only one flat segment (and the affine layers can only scale, shift, and flip the graph in the vertical dimension; they can't rotate the graph).

7. Hidden layer of size 2, no nonlinearities



(a) and (b). With no non-linearity, this reduces to a single affine function (in the same way as Q4)

8. Add nonlinearities between layers



All functions except for (g). Note that we can recreate any network from (5) by setting w_4 to 0, so this allows us to produce (c), (d) and (e). To produce the rest of the functions, note that h_1' and h_2' will be two independent functions with a flat part lying on the x-axis, and a portion with positive slope. The final layer takes a weighted sum of these two functions. To produce (a) and (b), the flat portion of one ReLU should start at the point where the other ends ($x = 0$ for (a), or $x = 1$ for (b). The final layer then vertically flips the ReLU sloping down and adds it to the one sloping up, producing a single sloped line. To produce (h), the ReLU sloping down should have its flat portion end (at $x = 0$ before the other's flat portion begins (at $x = 3$). The down-sloping one is again flipped and added to the up-sloping. To produce (f), both ReLUs should have equal slope, which will cancel to produce the first flat portion above the x-axis.