CS188 Fall 2018 Section 12: Neural Networks and Decision Trees

1 Perceptron $\rightarrow$ 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 $z = w_1x_1 + w_2x_2$, 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?

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

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?

2 Decision Trees

You are a geek who hates sports. Trying to look cool at a party, you join a discussion that you belive to be about football and basketball. You gather information about the two main subjects of discussion, but still cannot figure out what sports they play.

Sport	Position	Name	Height	Weight	Age	College
?	Guard	Charlie Ward	6'02"	185	41	Florida State
?	Defensive End	Julius Peppers	6'07"	283	32	North Carolina

Fortunately, you have brought your CS 188 notes along, and will build some classifiers to determine which sport is being discussed.

You come across a pamphlet from the Atlantic Coast Conference Basketball Hall of Fame, as well as an Oakland Raiders team roster, and create the following table:

Sport	Position	Name	Height	Weight	Age	College
Basketball	Guard	Michael Jordan	6'06"	195	49	North Carolina
Basketball	Guard	Vince Carter	6'06"	215	35	North Carolina
Basketball	Guard	Muggsy Bogues	5'03"	135	47	Wake Forest
Basketball	Center	Tim Duncan	6'11"	260	35	Oklahoma
Football	Center	Vince Carter	6'02"	295	29	Oklahoma
Football	Kicker	Tim Duncan	6'00"	215	33	Oklahoma
Football	Kicker	Sebastian Janikowski	6'02"	250	33	Florida State
Football	Guard	Langston Walker	6'08"	345	33	California

2.1 Entropy

Before we get started, let's review the concept of entropy.

1. Give the definition of entropy for an arbitrary probability distribution $P(X)$.

2. Draw a graph of entropy $H(X)$ vs. $P(X = 1)$ for a binary random variable X.

3. What is the entropy of the distribution of *Sport* in the training data? What about *Position*?

2.2 Decision Trees

Central to decision trees is the concept of "splitting" on a variable.

1. To review the concept of "information gain", calculate it for a split on the *Sport* variable.

2. Of course, in our situation this would not make sense, as *Sport* is the very variable we lack at test time. Now calculate the information gain for the decision "stumps" (one-split trees) created by first splitting on *Position*, *Name*, and *College*. Do any of these perfectly classify the training data? Does it make sense to use *Name* as a variable? Why or why not?

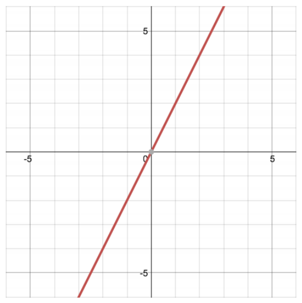
3. Decision trees can represent any function of discrete attribute variables. How can we *best* cast continuous variables (*Height*, *Weight*, and *Age*) into discrete variables?

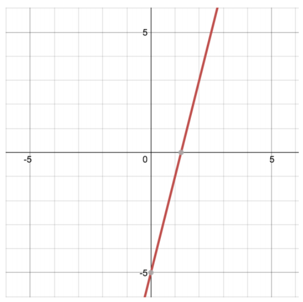
4. Draw a few decision trees that each correctly classify the training data, and show how their predictions vary on the test set. What algorithm are you following?

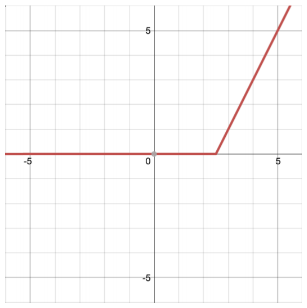
5. You may have noticed that the testing data has a value for *Position* that is missing in training data. What could we do in this case?

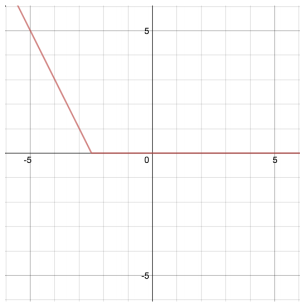
3 Neural Network Representations

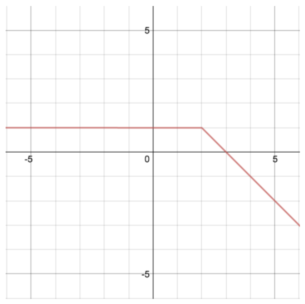
You are given a number of functions (a-h) of a single variable, x, which are graphed below. The computation graphs on the following pages will start off simple and get more complex, building up to neural networks. For each computation graph, indicate which of the functions below they are able to represent.

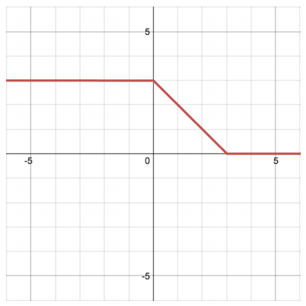

(a) $2x$

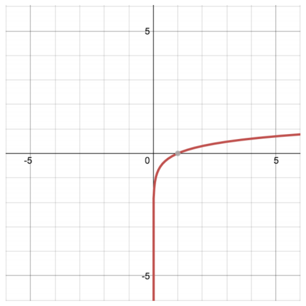

(b) $4x - 5$

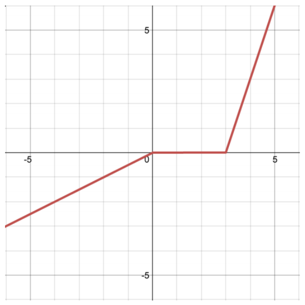

(c) $\begin{cases} 2x - 5 & x \geq 2.5 \\ 0 & x < 2.5 \end{cases}$


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

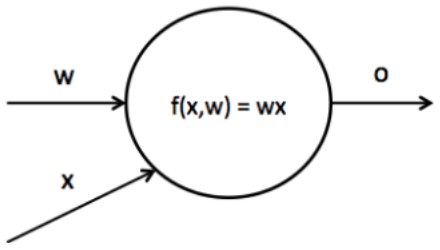

(e) $\begin{cases} -x + 3 & x \geq 2 \\ 1 & x < 2 \end{cases}$


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

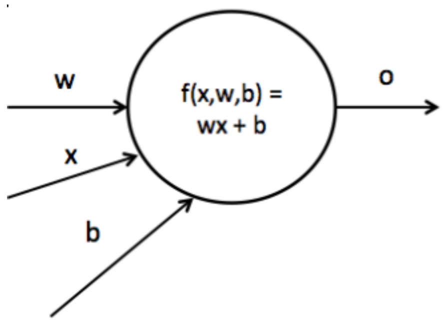

(g) $\log(x)$


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

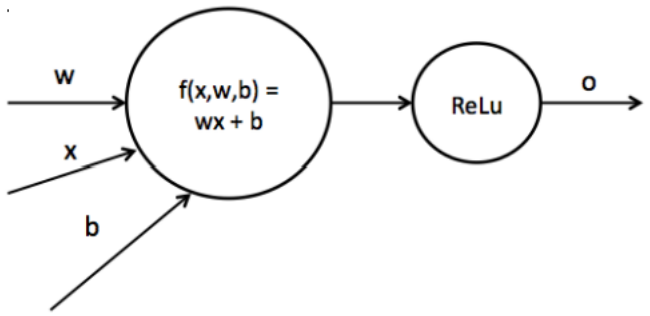
1. Consider the following computation graph, computing a linear transformation with scalar input x, weight w, and output o, such that $o = wx$. Which of the funcions can be represented by this graph? For the options which can, write out the appropriate value of w.



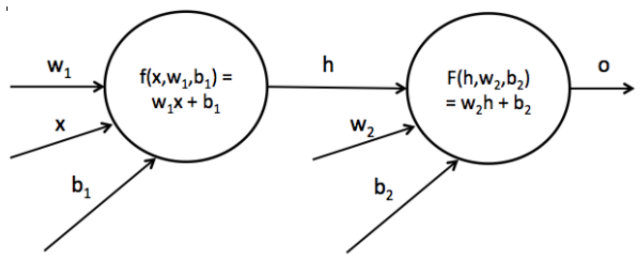
2. Now we introduce a bias term b into the graph, such that $o = wx + b$ (this is known as an *affine* function). Which of the functions can be represented by this network? For the options which can, write out an appropriate value of w, b.



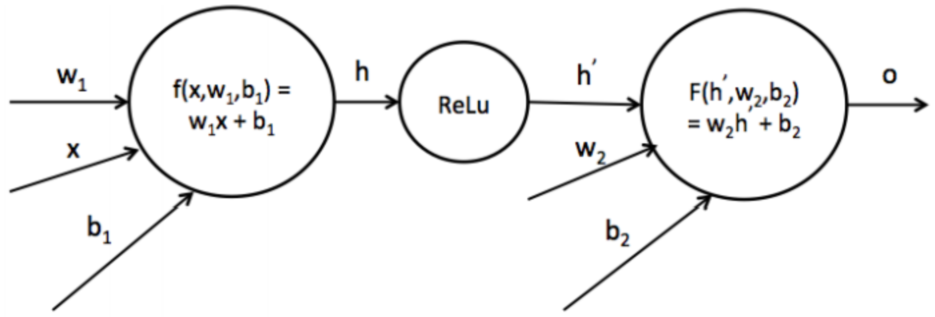
3. We can introduce a non-linearity into the network as indicated below. We use the ReLU non-linearity, which has the form $ReLU(x) = \max(0, x)$. Now which of the functions can be represented by this neural network with weight w and bias b? For the options which can, write out an appropriate value of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



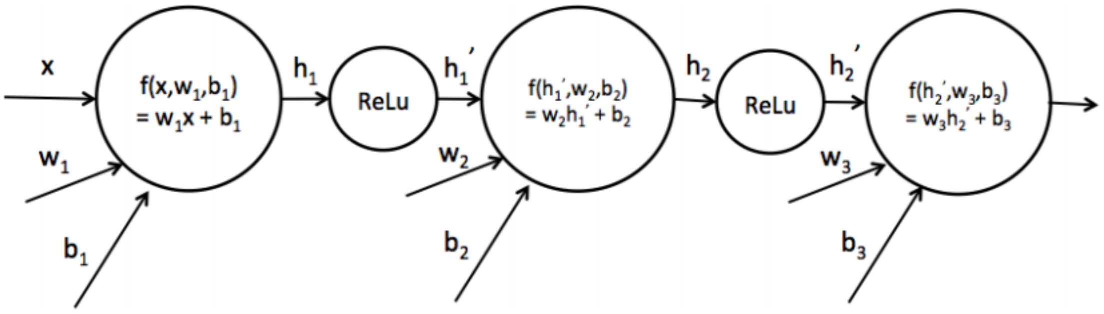
4. Now we consider neural networks with multiple affine transformations, as indicated below. We now have two sets of weights and biases w_1, b_1 and w_2, b_2. We denote the result of the first transformation h such that $h = w_1x + b_1$, and $o = w_2h + b_2$. Which of the functions can be represented by this network? For the options which can, write out appropriate values of w_1, w_2, b_1, b_2.



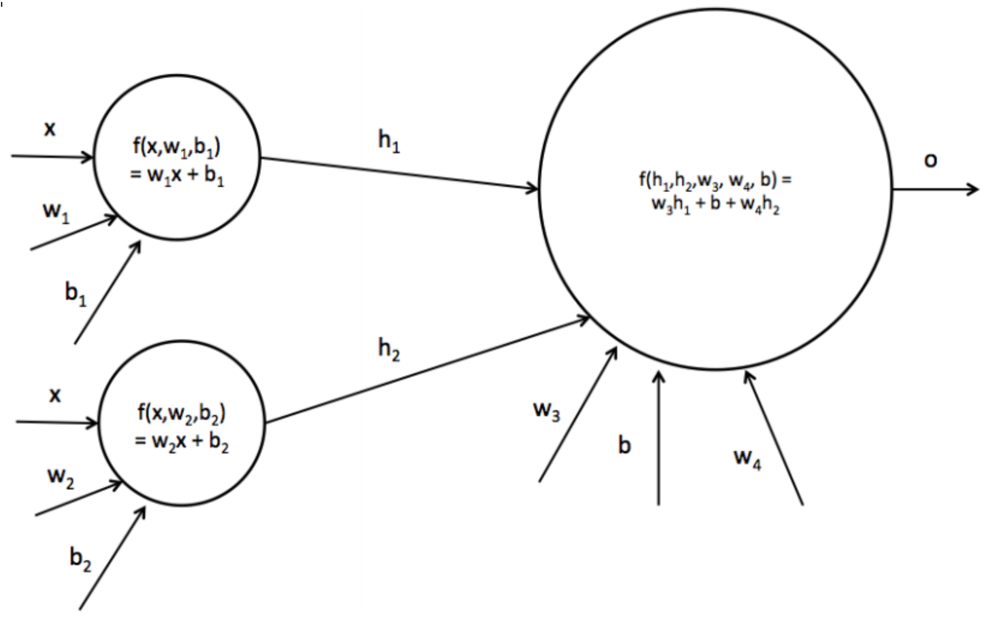
5. Next we add a ReLU non-linearity to the network after the first affine transformation, creating a hidden layer. Which of the functions can be represented by this network? For the options which can, write out appropriate values of w_1, w_2, b_1, b_2.



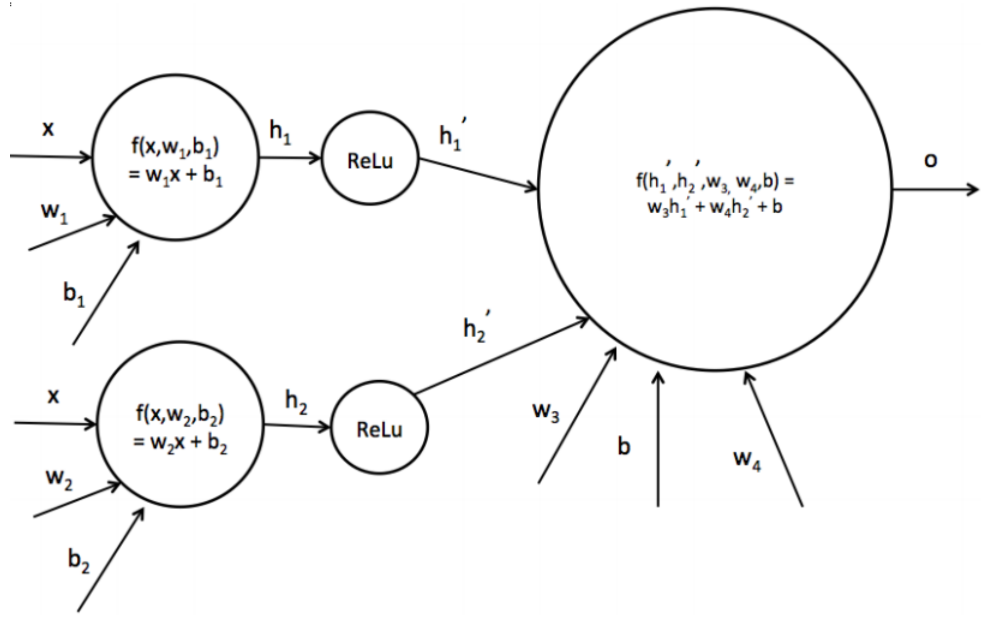
6. Now we add another hidden layer to the network, as indicated below. Which of the functions can be represented by this network?



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?



x
f(x,w₁,b₁) = w₁x + b₁
w₁
b₁
h₁
ReLu
h₁'
x
f(x,w₂,b₂) = w₂x + b₂
w₂
b₂
h₂
ReLu
h₂'
f(h₁',h₂',w₃, w₄,b) = w₃h₁' + w₄h₂' + b
o
w₃
b
w₄