CS188 Fall 2018 Section 12: Neural Networks and Decision Trees

1 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 $z = w_1x_1 + w_2x_2$, 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. 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?

$$w_i \leftarrow w_i - \alpha \frac{\partial Loss}{\partial w_1}$$

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 $(x_1, x_2) = (-1, 2)$, $y^* = 1$?

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 $w_2 = -2$.

4. 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 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:

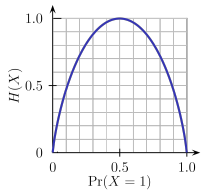
Sport	Position	Name	Height	Weight	Age	College
Basketball	Guard	Michael Jordan	6'06"	195	49	North Carolina
Basketball	Guard	Vince Carter	6'06"	215	35	North Carolina
Basketball	Guard	Muggsy Bogues	5'03"	135	47	Wake Forest
Basketball	Center	Tim Duncan	6'11"	260	35	Oklahoma
Football	Center	Vince Carter	6'02"	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 $P(X)$.
   $H(X) = \sum_x P(x) log_2 (1/P(x))$
   You can see this as the *expected* information content of the distribution.

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*?

   To calculate the entropy for a random variable, we estimate the probability distribution and use the formula from the part above.

$$P(S = \text{football}) = 1/2, P(S = \text{basketball}) = 1/2 \qquad H(S) = \frac{log_2(2)}{2} + \frac{log_2(2)}{2} = 1$$

$$P(P = \text{guard}) = 1/2, P(P = \text{kicker}) = 1/4, P(P = \text{center}) = 1/4 \qquad H(P) = \frac{log_2(2)}{2} + \frac{log_2(4)}{4} + \frac{log_2(4)}{4} = 3/2$$

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.

   Since the variable that we want to predict is *Sport*, we want to be caluclating the entropy with respect to the variable *Sport*.

   (a) i. Distribution before: 8 examples with (1/2, 1/2). (here the first number in the tuple is P(basketball), and the second number is P(football)).

   ii. Entropy before: $\frac{8}{8}\left(\frac{log(2)}{2}+\frac{log(2)}{2}\right)$

   (b) i. Distribution after: 4 examples with (1, 0), 4 examples with (0, 1)

   ii. Entropy after: $\frac{4}{8}\left(\frac{log(1)}{1}\right)+\frac{4}{8}\left(\frac{log(1)}{1}\right)=0$

   So, the information gain is $(1-0)=1$, which is the greatest possible.

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?

   Note that here we will be splitting on different variables but still need to look at the entropy of the distribution of the variable we need to predict which is *sport*. So, the before case remains same as before.

   (a) **Position**

   i. Distribution after: 4 examples with (3/4, 1/4), 2 examples with (1/2, 1/2), 2 examples with (0, 1).

   ii. Entropy after: $\frac{4}{8}\left(\frac{log(4/3)}{4/3}+\frac{log(4)}{4}\right)+\frac{2}{8}\left(\frac{log(2)}{2}+\frac{log(2)}{2}\right)+\frac{2}{8}\left(\frac{log(1)}{1}\right)=0.66$

   (b) **Name**

   i. Distribution after: 1 examples with (1, 0), 2 examples with (1/2, 1/2), 1 examples with (0, 1), 2 examples with (1/2, 1/2), 1 example with (0,1), 1 example with (0,1).

   ii. Entropy after: 0.5

   (c) **College**

   i. Distribution after: 2 examples with (1, 0), 1 examples with (1, 0), 3 examples with (1/3, 2/3), 1 examples with (0, 1), 1 example with (0,1).

   ii. Entropy after: 0.34

   Note that none of these variables completely classifies the data.

   Regarding using the **Name** as a feature to use in classifying data: since we expect people's names to be unique, using them as a feature in learning is akin to using the unique ID of each data point. That is to say, it's quite a bad idea—you will overfit to the training data.

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?

   Use an inequality relation, Attribute $> a$, where a is a split point chosen to give the highest information gain. E.g., an initial split on Age > 34 will perfectly classify the training data.

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? We use the algorithm as given in the slides, and for each split use the variable that gives us the maximum information gain. In this given problem, as we observed above, the variable Age correctly classifies all of the training data, so that is the first variable that gets picked up, and the algorithm stops at that.

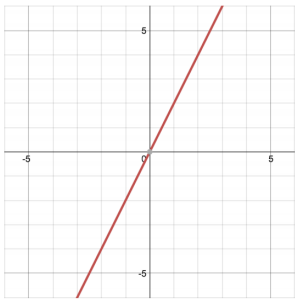
   This decision tree would predict test example 1 to be Basketball and test example 2 to be Football.

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?

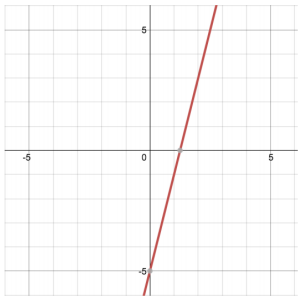
   When we come to a split on a variable whose value for the test subject is missing in the tree, we could just choose the most likely branch of the split (the branch that leads to the node with the greatest number of items).

3 Neural Network Representations

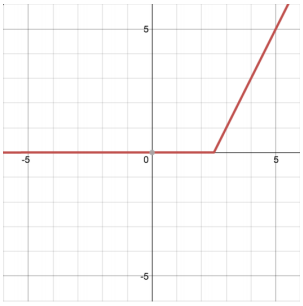
You are given a number of functions (a-h) of a single variable, x, which are graphed below. The computation graphs on the following pages will start off simple and get more complex, building up to neural networks. For each computation graph, indicate which of the functions below they are able to represent.



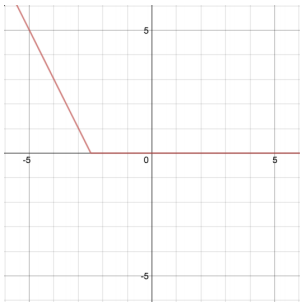
(a) $2x$



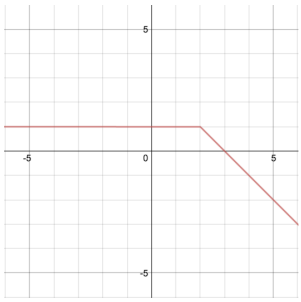
(b) $4x - 5$



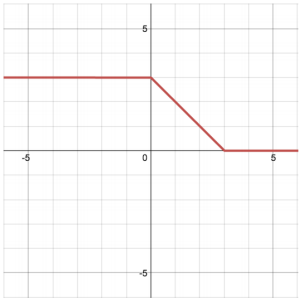
(c) $\begin{cases} 2x - 5 & x \geq 2.5 \\ 0 & x < 2.5 \end{cases}$



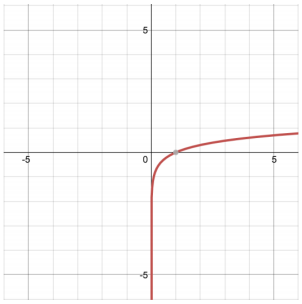
(d) $\begin{cases} -2x - 5 & x \leq -2.5 \\ 0 & x > -2.5 \end{cases}$



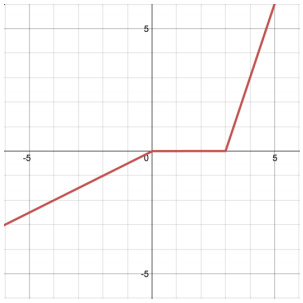
(e) $\begin{cases} -x + 3 & x \geq 2 \\ 1 & x < 2 \end{cases}$



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

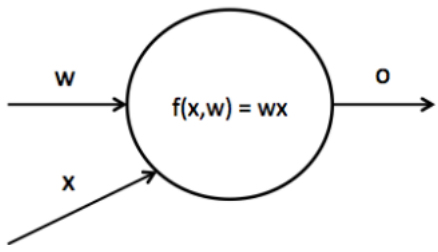


(g) $\log(x)$



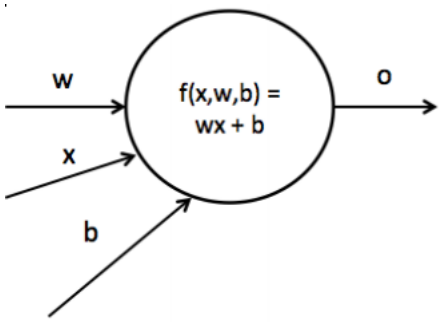
(h) $\begin{cases} 0.5x & x \leq 0 \\ 0 & 0 < x \leq 3 \\ 3x - 9 & x > 3 \end{cases}$

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.



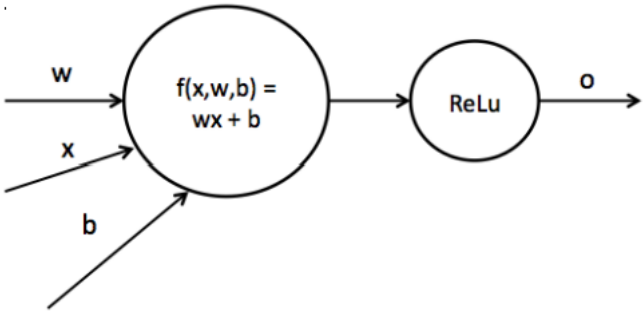
This graph can only represent (a), with $w = 2$. Since there is no bias term, the line must pass through the origin.

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.



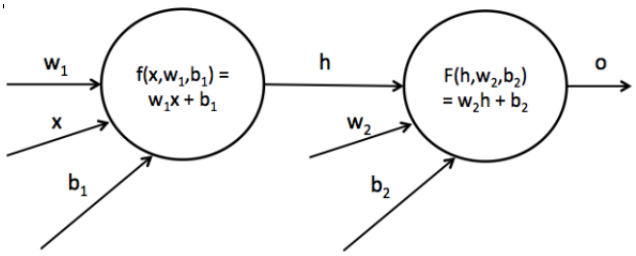
(a) with $w = 2$ and $b = 0$, and (b) with $w = 4$ and $b = -5$

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.



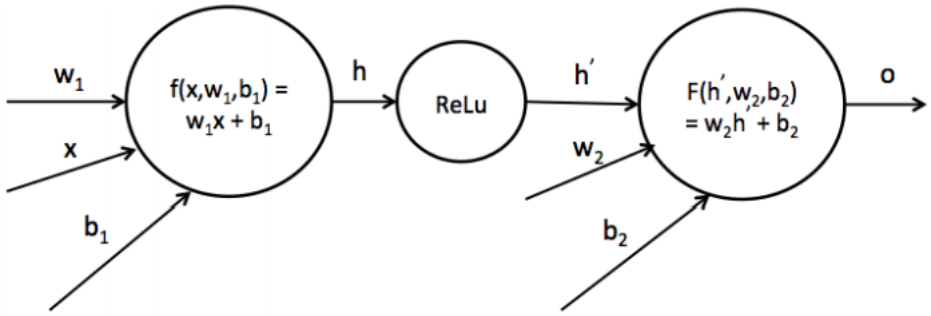
With the output coming directly from the ReLU, this cannot produce any values less than zero. It can produce (c) with $w = 2$ and $b = -5$, and (d) with $w = -2$ and $b = -5$

4. 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.



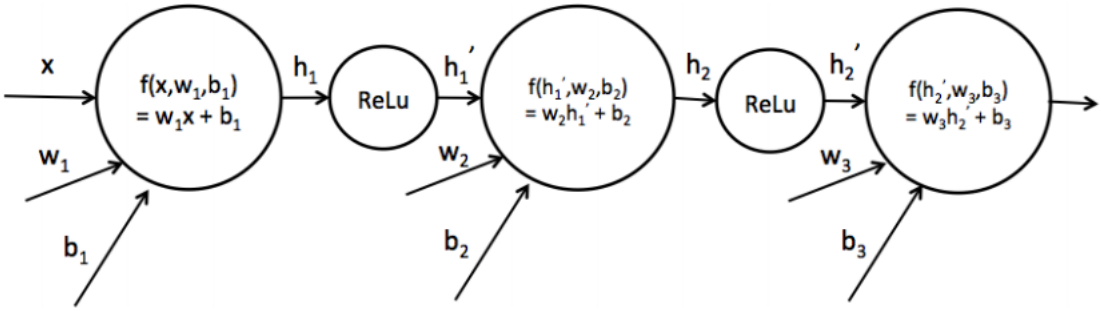
Applying multiple affine transformations (with no non-linearity in between) is not any more powerful than a single affine function: $w_2(w_1 x + b_1) + b_2 = w_2 w_1 x + w_2 b_1 + b_2$, so this is just a affine function with different coefficients. The functions we can represent are the same as in 1,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



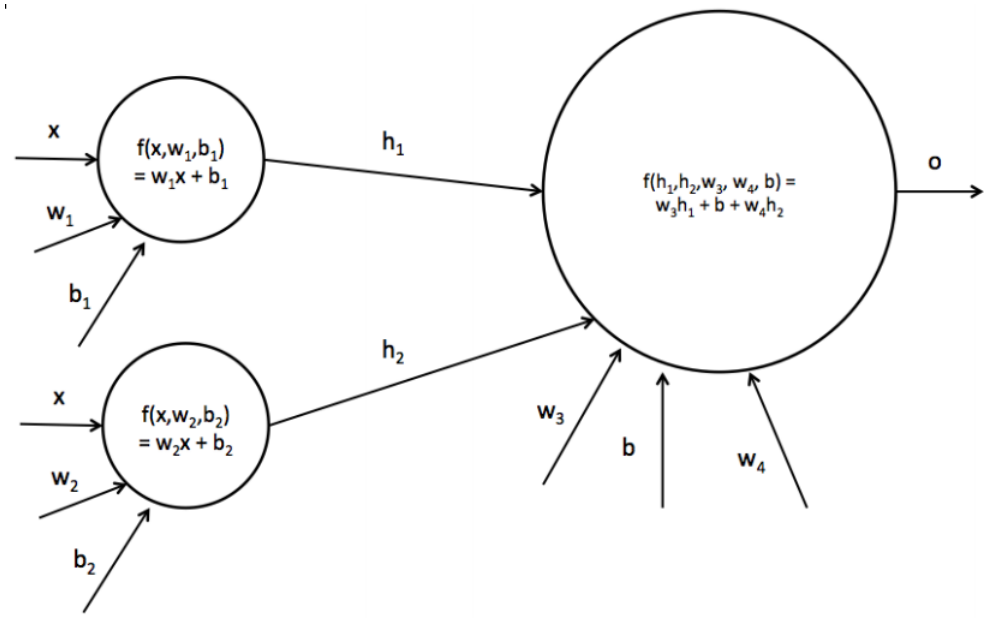
(c), (d), and (e). The affine transformation after the ReLU is capable of stretching (or flipping) and shifting the ReLU output in the vertical dimension. The parameters to produce these are:
(c) with $w_1 = 2, b_1 = -5, w_2 = 1, b_2 = 0$, (d) with $w_1 = -2, b_1 = -5, w_2 = 1, b_2 = 0$, and (e) with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

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



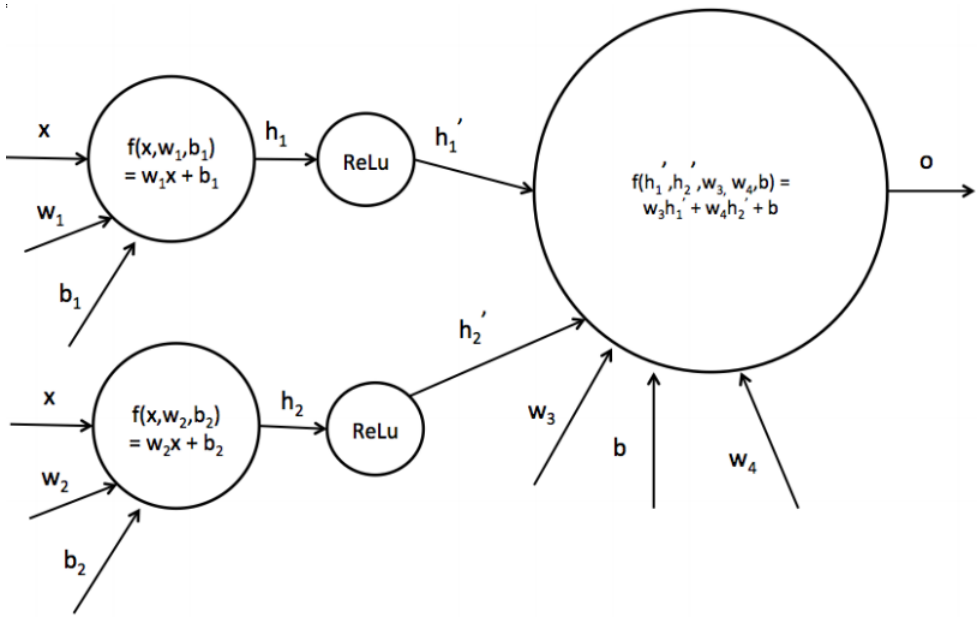
(c), (d), (e), and (f). The network can represent all the same functions as Q5 (because note that we could have $w_2 = 1$ and $b_2 = 0$). In addition it can represent (f): the first ReLU can produce the first flat segment, the affine transformation can flip and shift the resulting curve, and then the 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. 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?



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

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?



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.