Neural Nets
The Perceptron

Rao Vemuri

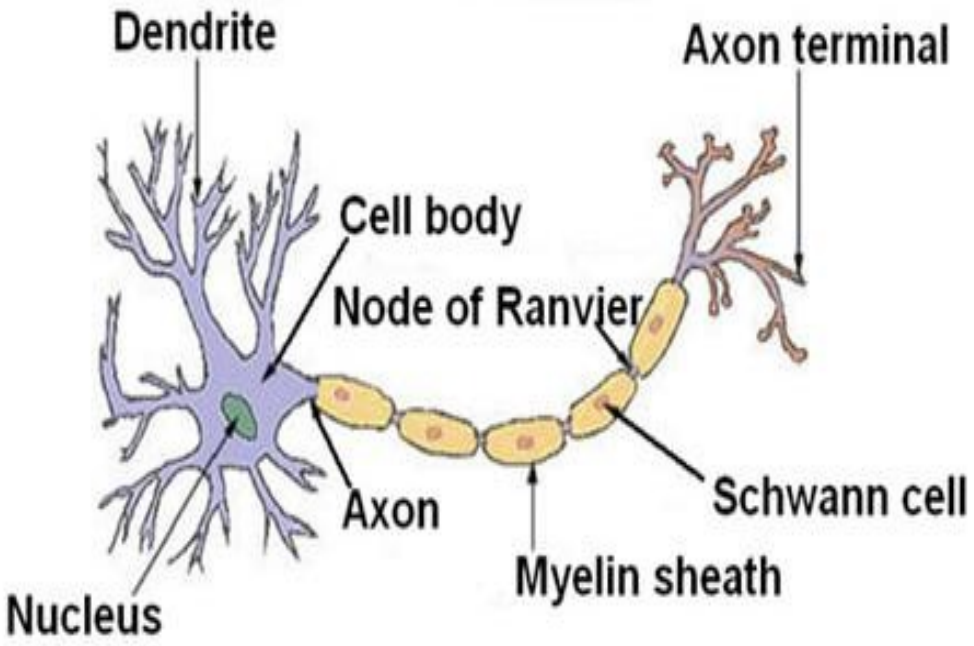
What is the Perceptron?

- The perceptron was invented in 1958 by Frank Rosenblatt at Cornell U
- Based on Rosenblatt's statements, *The New York Times* reported the perceptron to be "the embryo of an electronic computer that will be able to walk, talk, see, write, reproduce itself and be conscious of its existence."

How?

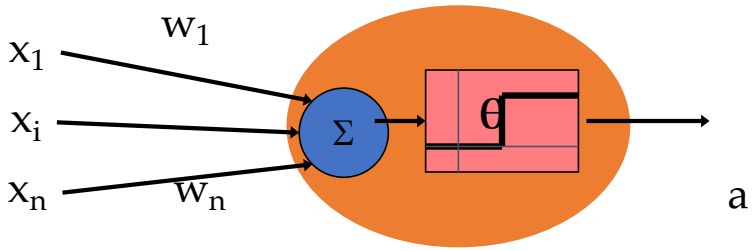
- The Perceptron was supposed to have represented a model of brain cells called neurons.
- That was a tall claim!!
- It caused a lot of excitement

Neuron in the Brain



Perceptron Model

- A perceptron (a neuron model) has several inputs
- a unit's inputs, x_i, are weighted, w_i, and **linearly combined**
- This weighted sum goes through a **linear threshold unit** (LTU), usually a step function with a "step" at θ
- Output of the threshold unit**,** called **activation** is binary (say, 0 or 1)



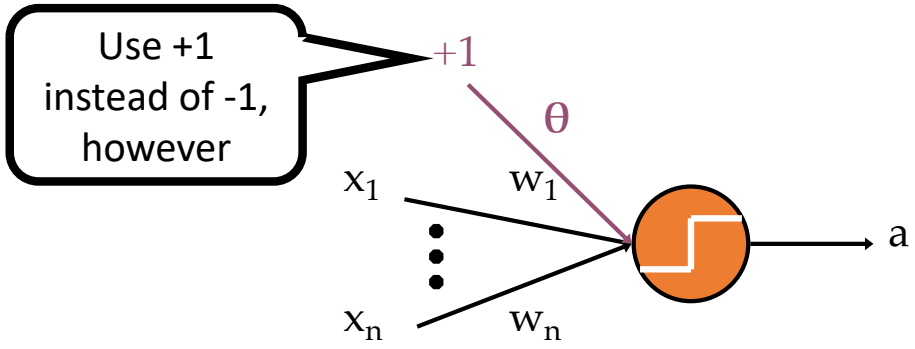
Linear Threshold Units (LTU)

Threshold can be thought of as just another weight (called the **bias**):

If $(w_1\, x_1) + (w_2\, x_2) + \ldots + (w_n\, x_n) \geq \theta$, *there is an output*

which is equivalent to

$(w_1\, x_1) + (w_2\, x_2) + \ldots + (w_n\, x_n) + (\theta(-1)) \geq 0$



Every unit uses a fixed threshold = 0

Perceptron Learning

How are the weights "learned" by a Perceptron?

- Programmer specifies:
  - A training data set

    (ie a table of inputs and corresponding desired outputs)
- Unknown: set of weights associated with inputs
- Learning (Calculation) of weights is *supervised*
  - each training example is comprised of a list of input values and the corresponding correct (True) output (T). Our job is to find the weights that satisfy the input-output relations

Perceptron Learning Algorithm

1. Initialize the weights in the network
   (usually with random values)
2. Repeat until all training examples correctly classified
   (or some other stopping criterion is met)

   for each example, ***e***, in the training set **do**

   Calculate Perceptron_output = O

   Compare with target output = *T*

   Calculate error = T – O

   Update weights

- Each pass through *all* the training examples is called an **epoch**
- Step 2 requires multiple epochs

Perceptron Learning Rule

How should the weights be updated?

Perceptron Learning Rule:

$$new\ w_i = old\ w_i + \Delta w_i$$

where $\Delta w_i = \alpha\ x_i\ (T - O)$

= **(learning rate) (input) (error)**

x_i is the input value associated with i^{th} input line

α = **learning rate** (a value chosen between 0.0 and 1.0)

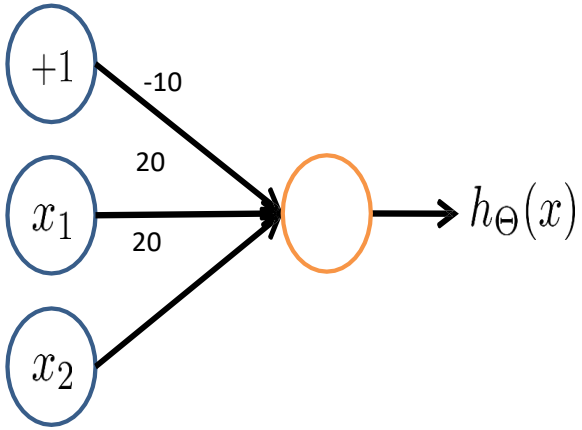
Perceptron Learning Rule Properties

- $\Delta w_i = \alpha\ x_i\ (T - O)$ doesn't depend on w_i
- No change in weight (i.e., $\Delta w_i = 0$) if:
  - **correct output**, i.e., $T = O$ gives $\Delta w_i = \alpha \times x_i \times 0 = 0$
  - **zero input**, i.e., $x_i = 0$ gives $\Delta w_i = \alpha \times 0 \times (T - O) = 0$
- If $T=1$ and $O = 0$, *increase* the weight

  so that maybe next time the result will exceed the output unit's threshold, causing it to be 1
- If $T=0$ and $O = 1$, *decrease* the weight

  so that maybe next time the result won't exceed the output unit's threshold, causing it to be 0

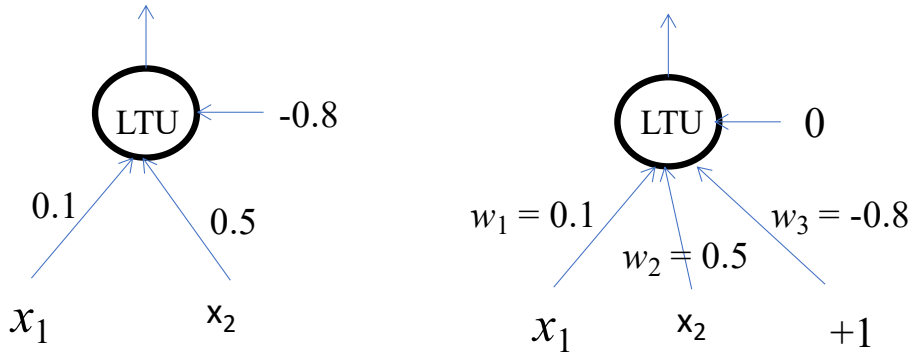
Example: Boolean OR function



x_1	x_2	$h_\Theta(x)$
0	0	
0	1	
1	0	
1	1	

Example: Learning OR

- $\Delta w_i = \alpha(T - O)x_i = 0.2(T - O)x_i$
- Initial network:



First input: Net input = $(0.1)x(x_1=0) + (0.5)x(x_2=0) + (-0.8)x(x_3=1) = -0.8$
This is NOT greater than 0, so output of TLU is 0,
Because T = 0, no weight adjustment. Apply next input

Ex: Training Steps for Learning OR

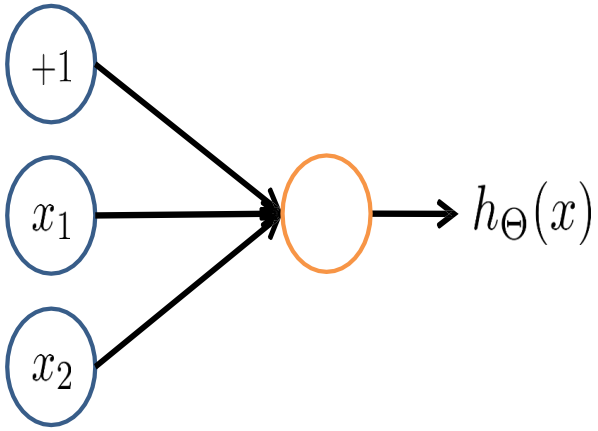
bias

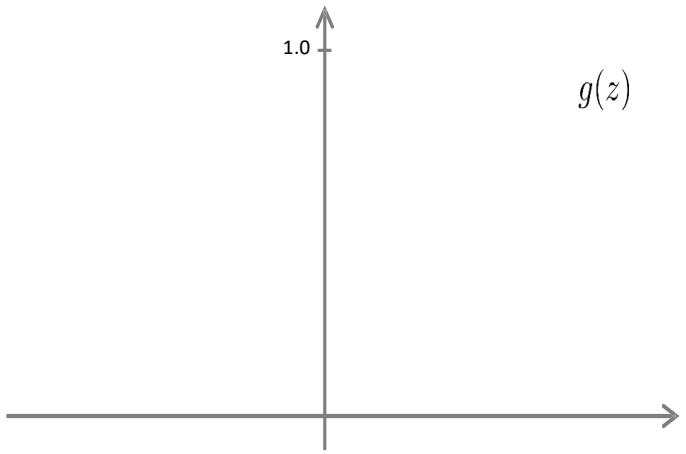
x1	x2	T	O	Δw1	w1	Δw2	w2	Δw3	w3
					0.1		0.5		-0.8
0	0	0	0	0	0.1	0	0.5	0	-0.8
0	1	1	0	0	0.1	0.2	0.7	0.2	-0.6
1	0	1	0	0.2	0.3	0	0.7	0.2	-0.4
1	1	1	1	0	0.3	0	0.7	0	-0.4
0	0	0	0	0	0.3	0	0.7	0	-0.4
0	1	1	1	0	0.3	0	0.7	0	-0.4
1	0	1	0	0.2	0.5	0	0.7	0.2	-0.2
1	1	1	1	0	0.5	0	0.7	0	-0.2
0	0	0	0	0	0.5	0	0.7	0	-0.2
0	1	1	1	0	0.5	0	0.7	0	-0.2
1	0	1	1	0	0.5	0	0.7	0	-0.2
1	1	1	1	0	0.5	0	0.7	0	-0.2

Homework: AND

$x_1, x_2 \in \{0, 1\}$

$y = x_1 \text{ AND } x_2$





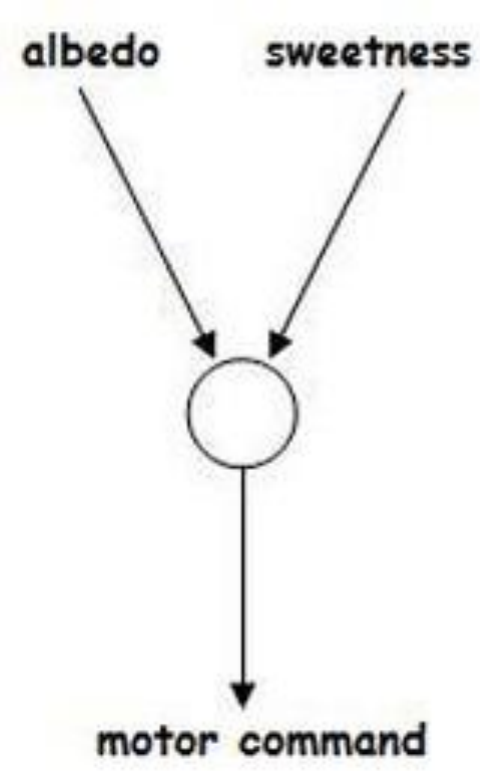
x_1	x_2	$h_\Theta(x)$
0	0	
0	1	
1	0	
1	1	

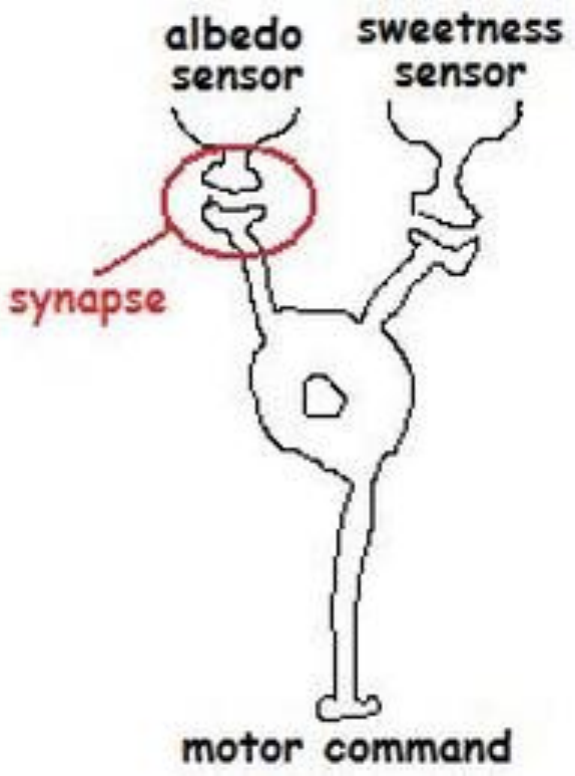
Do this as HW: For what weights does this behave like AND?
Start with the same initial weights as we did for OR and develop the table

Simulating a Neuron in Excel

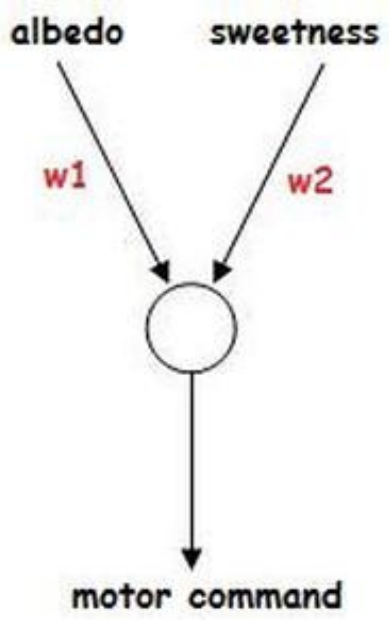
- Consider a neuron in our brain that decides whether we eat or don't eat a piece of food that is in front of us based upon the food's albedo (whiteness) and how sweet it smells.
- Input 1 is connected to an “albedo sensor” which is active (1) if the food is white, and (0) if the food is dark.
- Input 2 is connected to a “sweetness sensor” which is active with a value of 1 when the food smells sweet, and with a value of 0 when it doesn't.
- The output is a motor command to eat the food or not, taking a value of 1 or 0 respectively.

Model of a Neuron





Definition of a Neuron & its Weights



- Activation

  a = w1* input 1 + w2 * input 2

Output = 0; if a < threshold

Output = 1; if a > threshold

- There are 2 inputs. Let us assume that
- Albedo sensor can detect if the food is white (1) or dark (0)
- Sweetness sensor can detect if the food smells sweet (1) or not (0)
- There are four possibilities:

0 (Dark)	0 (Not sweet)	0 (Do not eat)
0 (Dark)	1 (sweet)	0 (Do not eat)
1 (White)	0 (Not sweet)	0 (Do not eat)
1 (White)	1 (sweet)	1 (Eat!)

Let us assume we eat the food if it is white and sweet

Training Data Set

	albedo	sweetness	eat?
nothing	0	0	0
marmite	0	1	0
salt	1	0	0
ice cream	1	1	1

Creating an Excel Spread Sheet

	A	B	C	D	E
1	inputs				
2	weights	0.2		0.3	
3	activation				
4	threshold		1		
5	output				
6					

Initializing the Neuron

- In Excel, type in what I've done in the picture here. Note that the inputs (cells B1 & D1) are empty. ie, the neuron is not presented with any food.
- I've chosen the weights arbitrarily. You'll have a chance to explore what happens when different weights are used
- The threshold has also been chosen arbitrarily = 1. You'll see later that it's not so important what value you use, as long as you select the weights appropriately.

Defining Activation

- Here, the activation is the weighted sum of the inputs
- The activation will be calculated in cell C3, so click on this cell and type the following:

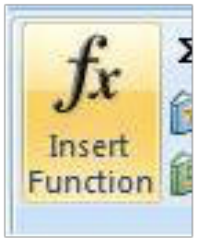
  =B1*B2 + D1*D2

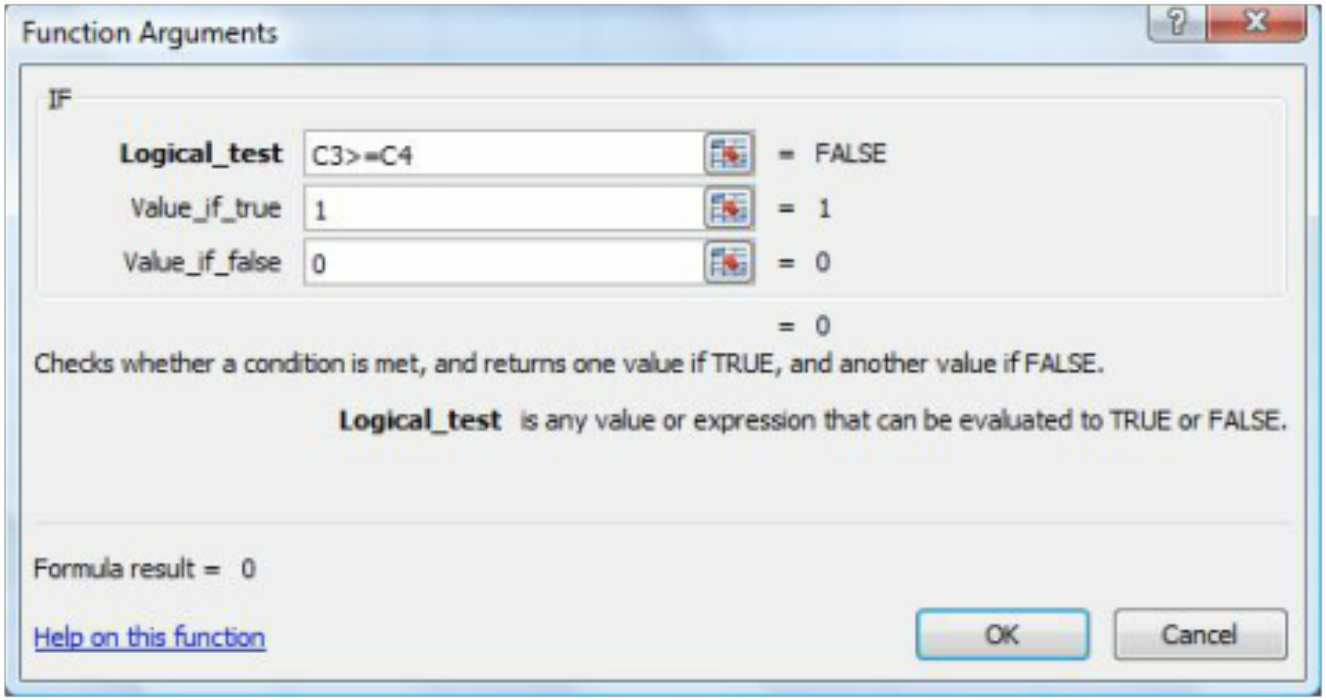
- Hit return, then check that it's worked by double clicking on C3. You should see a color-coded sum of the cells that the equation is using. Be sure you know what each element of this equation means: B1 is the albedo value of the food that the neuron is looking at, D1 is the associated weight, etc.

Comparing the Activation with Threshold

- Finally, we need to compare the activation to the threshold to see if the neuron should fire or not. This will happen in cell C5, so click on it.
- This time let's use a built-in Excel function. Go to the "Formulas" tabbed menu, then find the "Insert Function" button on the left. When you click this, a floating box will come up like the one shown below.

Floating Formula Box





Firing the Neuron

- Our logical test: is the activation greater than the threshold?
- Since we've put the activation in cell C3 and the threshold in cell C4, our formula for this is C3>=C4, meaning "if C3 is greater than or equal to C4". If this is the case, we want the neuron to fire, so we put 1 (fire) in the "Value_if_true" field and 0 (don't fire) in the "Value_if_false" field. Click ok.

Testing

	albedo	sweetness	eat?
nothing	0	0	
marmite	0	1	
salt	1	0	
ice cream	1	1	

Testing the Neuron

- Show the neuron "nothing" by putting 0 in cell B1 (albedo sensor) and 0 in cell D1 (sweetness sensor). What does the neuron output? (remember that the desired behavior is 0 - not to eat)
- Show the neuron "marmite" by putting 0 in B1 and 1 in D1. What does it output?
- Show the neuron "salt": 1 in B1 and 0 in D1.
- Show the neuron "ice cream": 1 in both input sensors. This is the only situation in which the neuron should decide to "eat". Does it output 1, as desired?

Trial and Error Method

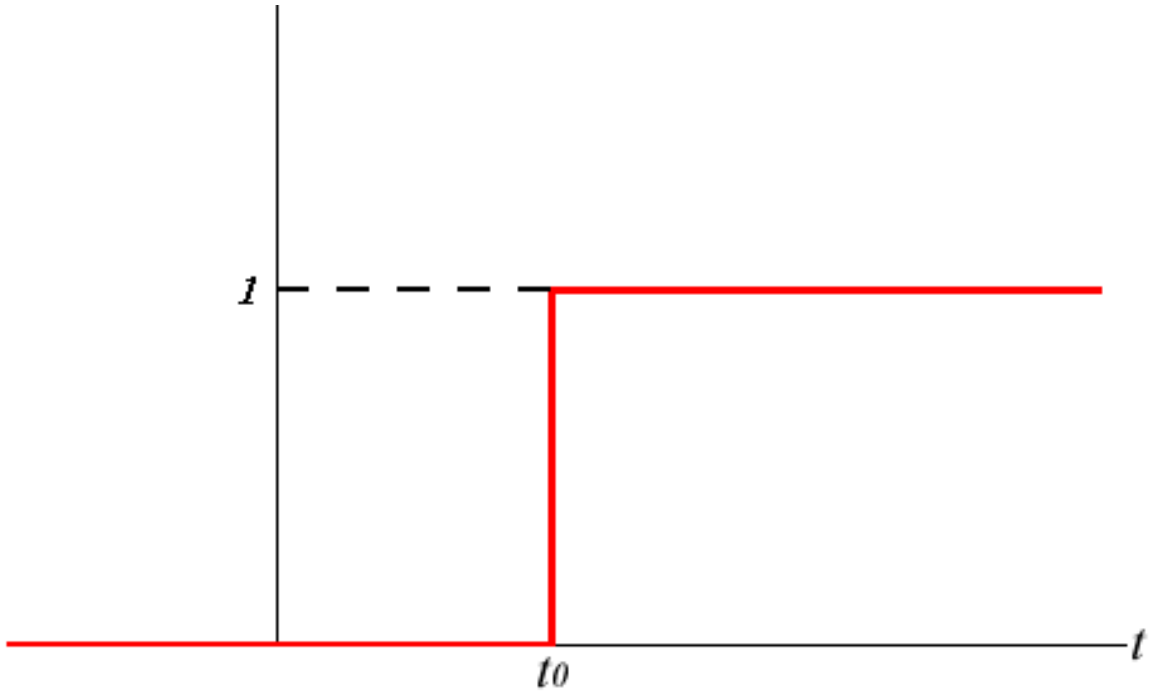
- Did the neuron behave as expected?

  If it did not (and if you've used the same weights as I did, it won't!), then we need to change the weights. (that is, train the neuron)

- Let's try weights of 1.5 for albedo and 1 for sweetness. That is, in cell B2 put 1.5 and in cell D2 put 1. (this is called trial and error method)

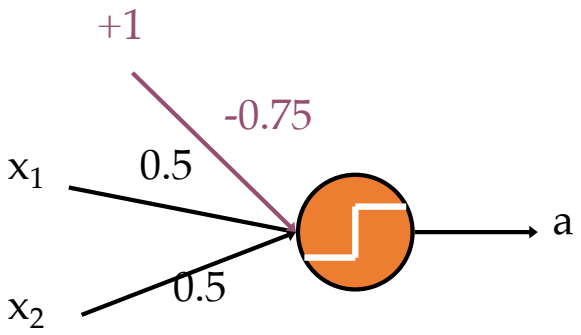
- Ok, you must be getting sick of entering 1s and 0s for the inputs now, so this time I'll give you some weights that will solve the problem: albedo = 0.6, sweetness = 0.7.
- A bit later we'll look at why these weights in particular work while the others didn't. You may be able to work it out for yourself. It's not so difficult.

f() is a Step Function (LTU)



Separating Hyperplane Example

- "AND" Perceptron:
  - inputs are 0 or 1
  - output is 1 when both x_1 and x_2 are 1

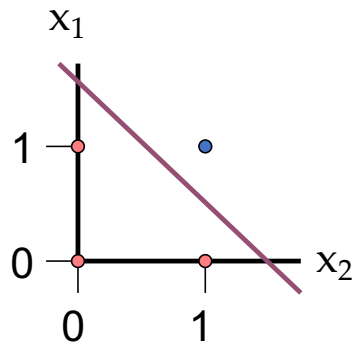


- 2D "input space"

  decision line equation:

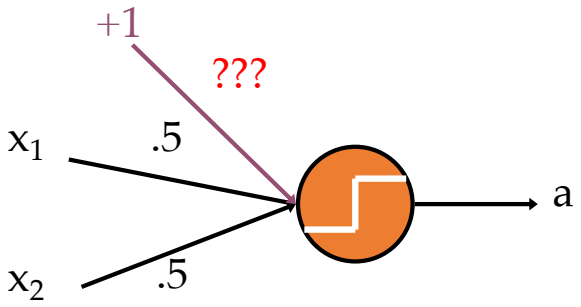
  $0.5x_1 + 0.5x_2 - 0.75 = 0$

  or, in slope-intercept form: $x_1 = -x_2 + 1.5$



Limits of Perceptron Learning: XOR

- "XOR" Perceptron?
  - inputs are 0 or 1
  - output is 1 when
    x_1 is 1 and x_2 is 0 or
    x_1 is 0 and x_2 is 1



- 2D input space with
  4 possible data points

- How do you separate
  + from – using a straight line?