

# How to Teach *Neuron Sandbox* In Middle School

David S. Touretzky  
Carnegie Mellon University

November 28, 2023

# Accessing Neuron Sandbox

Neuron Sandbox runs in any web browser. Access it for free at:

<https://www.cs.cmu.edu/~dst/NeuronSandbox>

Neuron Sandbox is an open source project published on GitHub:

<https://github.com/touretzkyds/NeuronSandbox>

Authors: Angela Chen, Neel Pawar, and David S. Touretzky, Carnegie Mellon

# Motivation: Why Are We Learning This?

Many of the AI-powered applications we use every day are built from **artificial neural networks**.

These systems that understand speech, recognize faces, converse with us (as in ChatGPT), drive cars, and more, use thousands or even millions of artificial neurons.

Even one artificial neuron can do quite a lot.

Neuron Sandbox teaches you how one artificial neuron works.

# Learning Progression

Stage 1 (steps 1-8) teaches students about Boolean functions, truth tables, the AND and OR functions, and how a linear threshold neuron works.

Completing stage 1 is a good goal for middle school students if there is limited time available.

Stage 2 (step 9) adds logical negation (NOT) and negative weights/thresholds.

Stage 3 (steps 10-14) examines NOT-AND (“NAND”) and NOT-OR (“NOR”).

Stage 4 (step 15) examines three-input functions.

Stage 5 (steps 16-18) examines asymmetric functions “AND NOT” and “OR NOT”.

Stage 6 (steps 19-20) examines the EQUAL and XOR functions, which cannot be realized by a single linear threshold unit.

The “Enrichment” section at the end introduces more advanced concepts such as De Morgan’s Laws and the space of all Boolean functions.

# Pedagogy

Students experience what it “feels like” to be a neuron and develop an effective mental model of how simulated neurons work.

Learning outcome: students should be able to describe how a neuron works.

## Key Skills:

- Mapping an English statement to a logical formula.
- Drawing a truth table.
- Calculating the activation value of a neuron given its weights.
- Calculating the output value of a neuron given its weights and threshold.
- Finding the weights and threshold that allow a neuron to compute the desired output for each possible input.

# Vocabulary

- Neuron
- Weight
- Threshold
- Activation value
- Boolean function
- Truth table
- Logical negation (NOT)
- AND, OR, NAND, NOR, XOR functions

# How To Use The Worksheets

We have provided written worksheets to let students practice skills such as writing down a truth table or calculating activation values. Here is how to use them:

1. Let students experiment with the Neuron Sandbox tool so they can take advantage of the high degree of scaffolding. Click on the “How does this work?” button. Do Problem 1 (peanut butter and jelly) and Problem 3 (rain or snow) to let them experience both AND problems and OR problems.
2. Do the worksheets to let them reproduce by hand what they encountered in the Neuron Sandbox tool. Worksheet 1 is an AND problem; worksheet 2 is an OR problem.

# Step 1

Discuss the difference between “AND” an “OR”, and clarify that “OR” means *inclusive or*, i.e., “a or b or both”.

- A PBJ sandwich requires peanut butter and jelly
- I should wear my boots if it's raining or snowing
- Luisa wants to get a cat or a dog
- Joe likes ketchup and mustard on his hamburger
- Meat pizzas have sausage or pepperoni
- A car needs both headlights and brake lights



## Step 2

Understand that “AND” and “OR” can be viewed as mathematical functions that take inputs and produce an output.

The inputs are either “true” or “false”. The output is also either “true” or “false”.

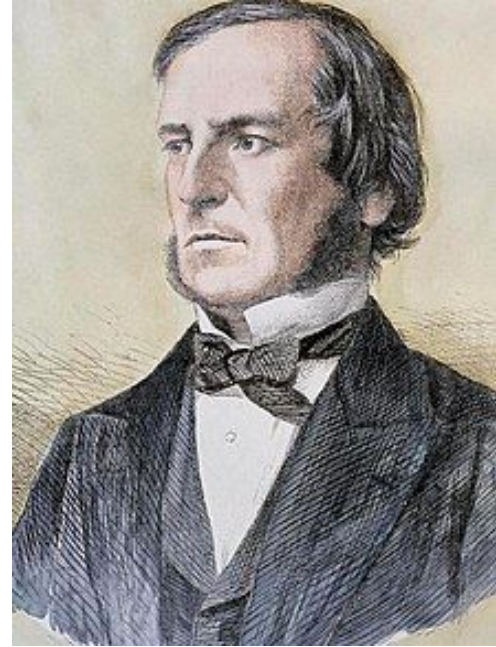
Example:

- a is “have peanut butter” (can be true or false)
- b is “have jelly” (can be true or false)
- “a AND b” is true if both a is true and b is true (can make PBJ sandwich)
- “a AND b” is false if either a is false or b is false (no sandwich)

Functions whose inputs and output are restricted to the values “true” or “false” are called Boolean functions.

# Boolean Functions

- A Boolean function takes inputs that are either true or false.
  - True can be represented by 1.
  - False can be represented by 0.
- The output of a Boolean function is also either true or false (1 or 0).
- Boolean functions are the building blocks of computer circuitry.



English mathematician  
George Boole:  
1815 - 1864.

# Step 3





Understand that a truth table shows the value of a Boolean function like “AND” or “OR” for every possible combination of inputs. Each row is a different combination.

“AND” Function

Inputs		
Have Peanut Butter	Have Jelly	Desired Output
0 	0 	0 
0 	1 	0 
1 	0 	0 
1 	1 	1 

PBJ sandwich

“OR” Function

Inputs		
Rain	Snow	Desired Output
0 	0 	0 
0 	1 	1 
1 	0 	1 
1 	1 	1 

should wear my boots













## Step 4

Turn an English description of a Boolean expression into a truth table, row by row. This will either be an “AND” truth table or an “OR” truth table. **Try This:** Use the “Solve for Outputs” mode in Neuron Sandbox to practice this.

Have Peanut Butter	Have Jelly	Predicted Output (0=No, 1=Yes)
0 	0 	 <input type="radio"/> 0 <input type="radio"/> 1
0 	1 	 <input checked="" type="radio"/> 0 <input type="radio"/> 1
1 	0 	 <input checked="" type="radio"/> 0 <input type="radio"/> 1
1 	1 	<input type="radio"/> 0 <input checked="" type="radio"/> 1 

# Writing Down the Truth Table for “OR”

I should wear my boots if it's raining or snowing.

Rain	Snow	Predicted Output (0=No, 1=Yes)
0 	0 	 0 0 1
0 	1 	0 0 1 
1 	0 	0 0 1 
1 	1 	0 0 1 

# Ask Students To Draw A Truth Table

- The ordering of 1's and 0's doesn't have to exactly match what Neuron Sandbox does.
- But they must list all four cases correctly.

—— Inputs ——

Have Bread	Have Butter

## Step 5

Understand the kind of neuron we're studying: the linear threshold unit.

- It multiplies each of its inputs (0 or 1) by a **weight** (a number) and adds up the results. The sum is called the **activation** of the unit.

$$\text{activation} = \text{input}_1 \times \text{weight}_1 + \text{input}_2 \times \text{weight}_2$$


- The neuron compares its activation to a **threshold** (another number) and outputs 1 if the activation is greater than the threshold. Otherwise it outputs 0.

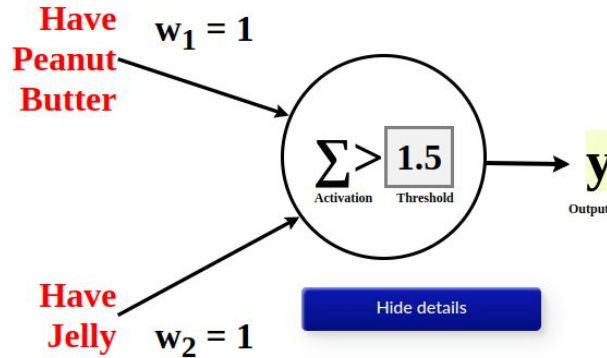
**TRY THIS:** run the PBJ problem in Neuron Sandbox, click on “How does this work?”, and hover over the rows of the truth table to see how the computation is performed. Then do the same for the boots problem.

# The “AND” Function









The input weights  $w_1$  and  $w_2$  are both 1.

The threshold is 1.5.

Inputs	
Have Peanut Butter	Have Jelly
0 	0 
0 	1 
1 	0 
1 	1 

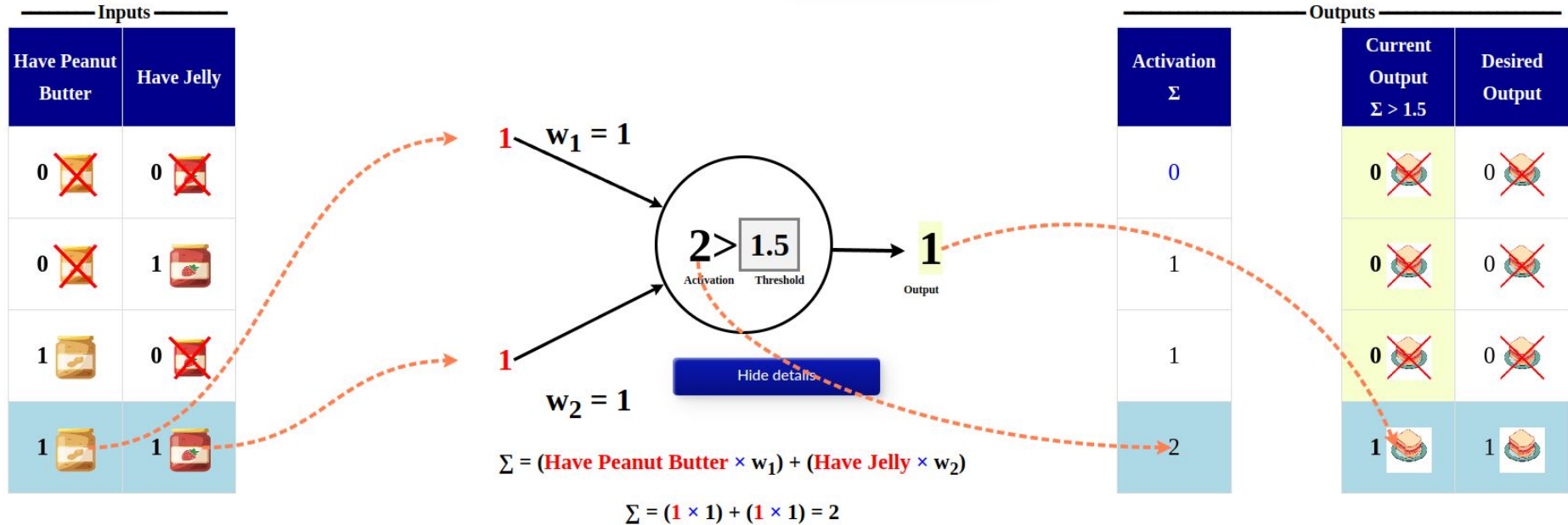


$$\Sigma = (\text{Have Peanut Butter} \times w_1) + (\text{Have Jelly} \times w_2)$$

Activation $\Sigma$	Outputs	
	Current Output $\Sigma > 1.5$	Desired Output
0	0 	0 
1	0 	0 
1	0 	0 
2	1 	1 



# Hovering Over a Row of the “AND” Truth Table



# Reflection on the AND Threshold

Our “AND” function uses a threshold of 1.5.

Would a threshold of 1.4 work? Try it!

Would a threshold of 1.6 work? Try it!

**INVESTIGATE:** what is the range of threshold values we can use for “AND”, given that the weights are both 1?

(Answer: the threshold must be greater than or equal to 1, and less than 2.)

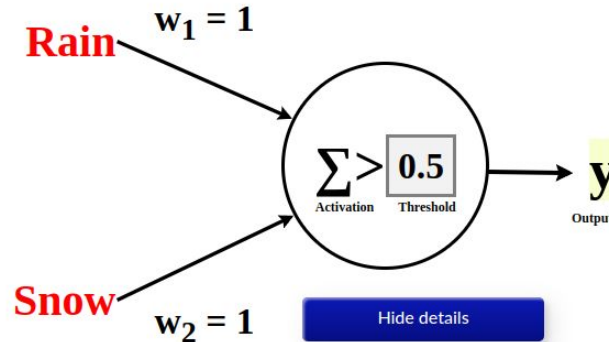
# The “OR” Function

“I should wear my boots if it is raining or snowing.”









Input weights are 1. Threshold is 0.5.

Activations are the same as for “AND”; only the threshold is different.

Inputs	
Rain	Snow
0 	0 
0 	1 
1 	0 
1 	1 



$$\Sigma = (\text{Rain} \times w_1) + (\text{Snow} \times w_2)$$

Activation $\Sigma$	Outputs	
	Current Output $\Sigma > 0.5$	Desired Output
0	0 	0 
1	1 	1 
1	1 	1 
2	1 	1 

# Reflection on the OR Threshold

Our “OR” function uses a threshold of 0.5.

Would a threshold of 0.4 work? Try it!

Would a threshold of 0.6 work? Try it!

**INVESTIGATE:** what is the range of threshold values we can use for “OR”, given that the weights are both 1?

(Answer: the threshold must be greater than or equal to 0, and less than 1.)

# Step 6

Compare the “AND” function to the “OR” function in Neuron Sandbox.

- The weights  $w_1$  and  $w_2$  are the same: 1 and 1.
- Therefore the activations are the same for all four rows of the truth table.
- The thresholds are different:
  - “AND” uses a threshold of 1.5
  - “OR” uses a threshold of 0.5
- The outputs are both 0 for the first row (both inputs 0).
- The outputs are different for the middle two rows: 0 for “AND” vs. 1 for “OR”.
- The outputs are both 1 for the last row (both inputs 1).

**REFLECTION:** How would the neuron behave with a threshold of 2.5? (Output always 0.)  
How would the neuron behave with a threshold of -0.5? (Output always 1.)

## Step 7

Given a neuron's weights, show how to compute its activation value  $\Sigma$  (sigma) for each row of the truth table.

**TRY THIS:** Click on the “How does this work?” button to display the calculation when you hover over a row of the table.

$$\Sigma = (\text{Have Peanut Butter} \times w_1) + (\text{Have Jelly} \times w_2)$$

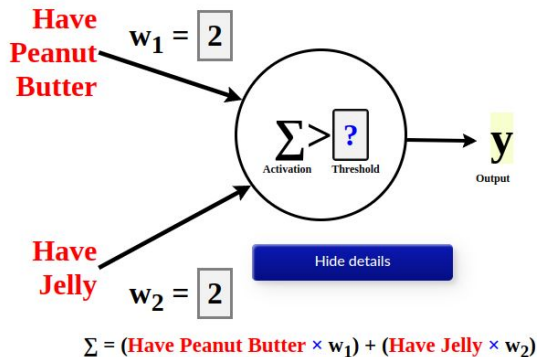
$$\Sigma = (1 \times 1) + (1 \times 1) = 2$$

Students should be able to reproduce this calculation on their own, given a worksheet showing the weights and truth table.

## Step 8

Understand how the threshold value depends on the scale of the weights. Given a neuron whose weights are both 2 instead of 1, what should the threshold be for an “AND” function or an “OR” function?

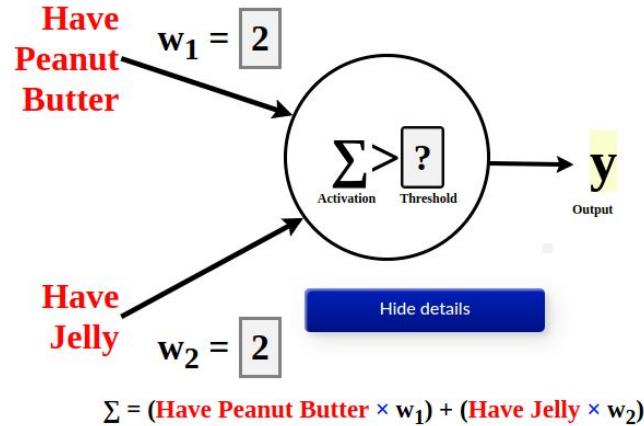
Inputs	
Have Peanut Butter	Have Jelly
0 	0 
0 	1 
1 	0 
1 	1 



**HINT:** compute the activation values first.

# Re-Scaled Weights

Inputs	
Have Peanut Butter	Have Jelly
0 	0 
0 	1 
1 	0 
1 	1 






Activation $\Sigma$
0
2
2
4

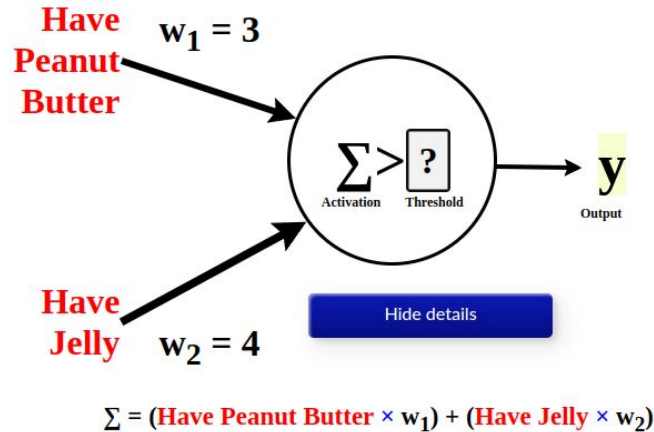
Answer: when the weights are both 2, the threshold could be 3.5 for “AND” and 0.5 for “OR”.



# Re-Scaled Weights

What if  $w_1$  is 3 and  $w_2$  is 4? What are the thresholds for “AND” and “OR” then?

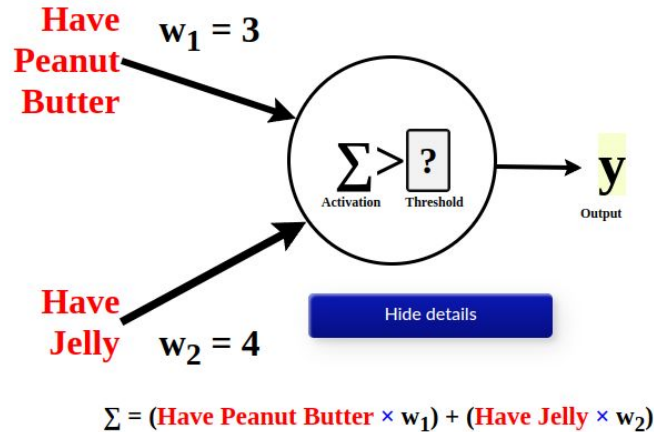
Inputs	
Have Peanut Butter	Have Jelly
0 	0 
0 	1 
1 	0 
1 	1 



# Re-Scaled Weights

What if  $w_1$  is 3 and  $w_2$  is 4? What are the thresholds for “AND” and “OR” then?

Inputs	
Have Peanut Butter	Have Jelly
0 	0 
0 	1 
1 	0 
1 	1 



For “AND” the threshold must be greater than or equal to 3 and less than 7.

For “OR” the threshold must be greater than or equal to 0 and less than 3.

## Step 9

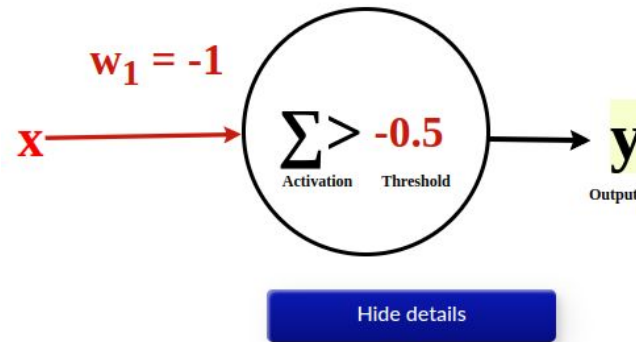
Understand the “NOT” function, which only takes 1 input.

- “NOT” of 0 is 1.
- “NOT” of 1 is 0.

**PROBLEM:** what weight and threshold values will produce the “NOT” function?

**HINT:** consider using negative values.

# The “NOT” Function



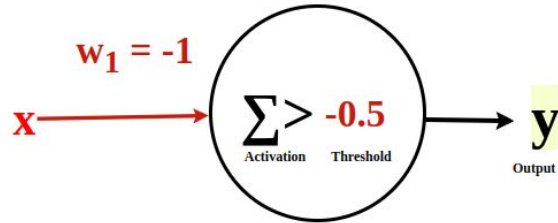
$$\Sigma = (x \times w_1)$$

**PROBLEM:** work through the activation and output calculations for “NOT” given an input of “false” (0) and an input of “true” (1).

# The “NOT” Function

—Inputs—

x
0
1



Hide details

$$\Sigma = (x \times w_1)$$

—Outputs—

Activation $\Sigma$	Current Output $\Sigma > -0.5$	Desired Output
0	1	1
-1	0	0

## Step 10

Now that we have negation (the “NOT” function), we can apply it to “AND” to get the “NOT-AND” function, “NOT (a AND b)”, also called the “NAND” function.

Example: a baby will be distraught if it hasn’t both been fed and had its diaper changed.

- a is “baby has been fed”
- b is “baby had its diaper changed”
- distraught = NOT (a AND b) = a NAND b

**PROBLEM:** write down the truth table for “NAND”.

**REFLECTION:** Compare the truth table for “NAND” to the one for “AND”.

# AND vs. NAND (NOT-AND)

a	b
0	0
0	1
1	0
1	1

a AND b	a NAND b
0	1
0	1
0	1
1	0

# Language Check

NAND is short for NOT-AND.

“a NAND b” is short for

- $\text{NOT}(a \text{ AND } b)$


“a NAND b” is not the same as either of these:

- $(\text{NOT } a) \text{ AND } (\text{NOT } b)$
- $a \text{ AND NOT } b$

**PROBLEM:** write out the truth tables for all three of these expressions to prove that they really are different.



# NAND: Language Matters



a	b
0	0
0	1
1	0
1	1

NOT (a AND b)	(NOT a) AND (NOT b)	a AND NOT b
1	1	0
1	0	0
1	0	1
0	0	0

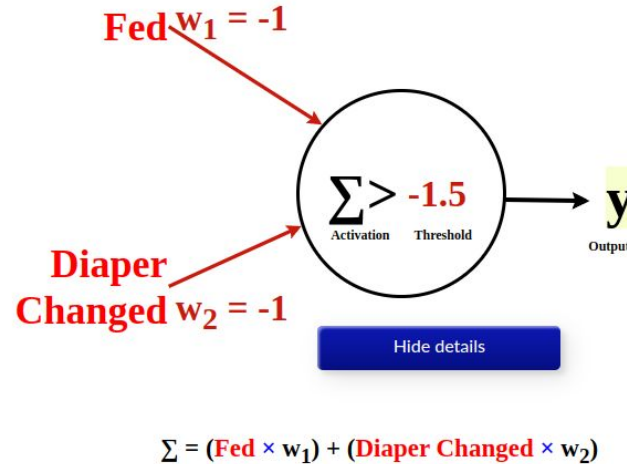
## Step 11

Find weight and threshold values to implement the NAND function.

**HINT:** Consider using negative values.

# Weights and Threshold for NAND

Inputs	
Fed	Diaper Changed
0	0
0	1
1	0
1	1



Outputs		
Activation $\Sigma$	Current Output $\Sigma > -1.5$	Desired Output
0	1	1
-1	1	1
-1	1	1
-2	0	0

## Step 12

The “NOR” function (“NOT-OR”) is the opposite of “OR”.

“a NOR b” means “NOT (a OR b)”.

Example: a pizza is good to eat if it's not undercooked or overcooked.

- a is “pizza is undercooked”
- b is “pizza is overcooked”
- pizza is good if NOT (a OR b)

**PROBLEM:** compare the truth tables for OR and NOR.

# OR vs. NOR

a	b
0	0
0	1
1	0
1	1

a OR b	a NOR b
0	1
1	0
1	0
1	0

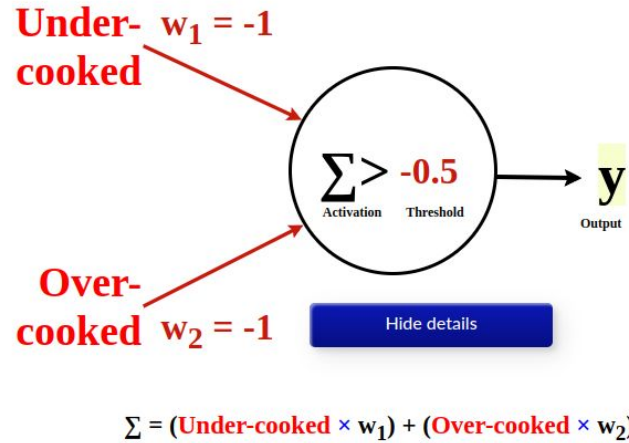
## Step 13

Find weight and threshold values to implement the NOR function.

**HINT:** Consider using negative values.

# Weights and Threshold for NOR

Inputs	
Under-cooked	Over-cooked
0	0
0	1
1	0
1	1



Outputs		
Activation $\Sigma$	Current Output $\Sigma > -0.5$	Desired Output
0	1	1
-1	0	0
-1	0	0
-2	0	0

## Step 14

Consider a neuron that takes three inputs and outputs “true” if at least 2 of its inputs are true.

How many rows does the truth table have? (Answer:  $2^3 = 8$  rows.)

Draw the truth table for for this function.



# Truth Table for “At Least 2 True Inputs”

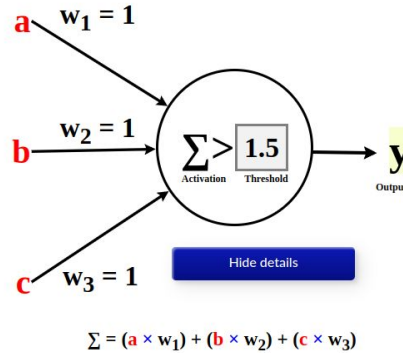
a	b	c
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Desired Output
0
0
0
1
0
1
1
1

# Step 15

Find the weights and thresholds to implement the “at least 2 true inputs” function.

Inputs		
a	b	c
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



Outputs		
Activation $\Sigma$	Current Output $\Sigma > 1.5$	Desired Output
0	0	0
1	0	0
1	0	0
2	1	1
1	0	0
2	1	1
2	1	1
3	1	1

## Step 16

Until now we've considered only symmetric functions, where all the inputs have the same weights.

Now we consider the asymmetric function “a AND NOT b”. Note that this is different than the NAND function, “NOT (a AND b)”. which is symmetric.

**PROBLEM:** write out the truth table for “a AND NOT b”.

## Truth Table for “a AND NOT b”

a	b
0	0
0	1
1	0
1	1

a AND NOT b
0
0
1
0

# Step 17

What are the correct weights and threshold for “a and NOT b”?

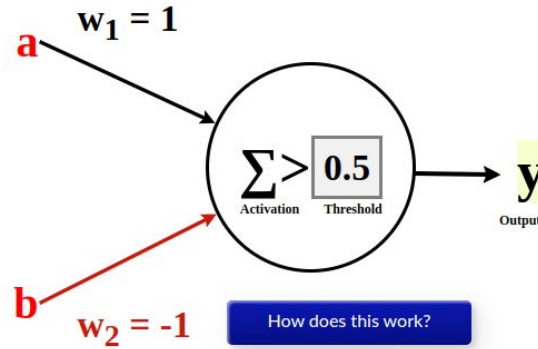
Here is how to reason it out:

1. Consider the case where both inputs are 0, so the activation is zero. Since the desired output is zero, the threshold should be positive.
2. Consider the case where  $a=1$  and  $b=0$ . In this case the desired output is one, so the weight on input a must be positive and greater than the threshold.
3. Now consider the case where  $a=1$  and  $b=1$ . The desired output is zero. So turning on the b input must put the activation below threshold. This means the b input needs a negative weight.

**CHALLENGE:** find weight and threshold values that solve the problem.

# a AND NOT b

Inputs	
a	b
0	0
0	1
1	0
1	1



Outputs		
Activation $\Sigma$	Current Output $\Sigma > 0.5$	Desired Output
0	0	0
-1	0	0
1	1	1
0	0	0

## Step 18

We saw how to get “a AND NOT b” using asymmetric weights.

**CHALLENGE:** how can we get “b AND NOT a”?

What does the truth table look like?

a	b
0	0
0	1
1	0
1	1

b AND NOT a
0
1
0
0

## b AND NOT a

To get the weights for “b AND NOT a”, take the solution for “a AND NOT b” and swap the weights for a and b.

Also note that “b AND NOT a” is the same as “(NOT a) AND b”: we just change the order of the two inputs to the “AND”.



## Step 18

Another asymmetric function is “a OR NOT b”. Write down its truth table.

a	b
0	0
0	1
1	0
1	1

a OR NOT b
1
0
1
1

## a OR NOT b

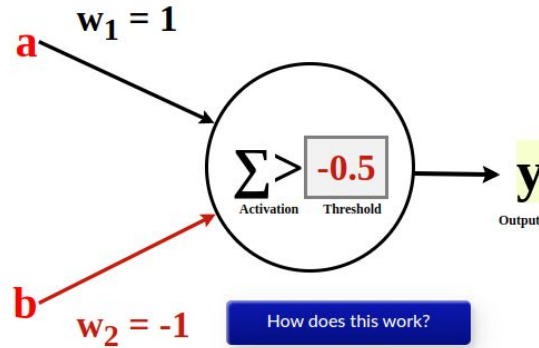
What weights and threshold produce “a OR NOT b”?

1. When both inputs are zero, the activation is zero but the desired output is one. So the threshold must be negative.
2. When input a is one and b is zero, the desired output is one, so a’s weight must be greater than the threshold.
3. When input b is one and a is zero, the desired output is zero, so b’s weight must be less than the threshold.

**PROBLEM:** find weight and threshold values to implement “a OR NOT b”.

# a OR NOT b

Inputs	
a	b
0	0
0	1
1	0
1	1



Outputs		
Activation $\Sigma$	Current Output $\Sigma > -0.5$	Desired Output
0	1	1
-1	0	0
1	1	1
0	1	1

## Step 19

The EQUAL function outputs 1 if its inputs match: either both 0 or both 1.

The NOT-EQUAL function is the opposite: it outputs 1 if its inputs don't match.

Another name for NOT-EQUAL is “exclusive or”, abbreviated XOR.

**PROBLEM:** write down the truth tables for regular OR (“inclusive or”) and for XOR (“exclusive or”).

# OR vs. XOR

a	b
0	0
0	1
1	0
1	1

a OR b	a XOR b
0	0
1	1
1	1
1	0

OR and XOR only differ in the case where both inputs are true.

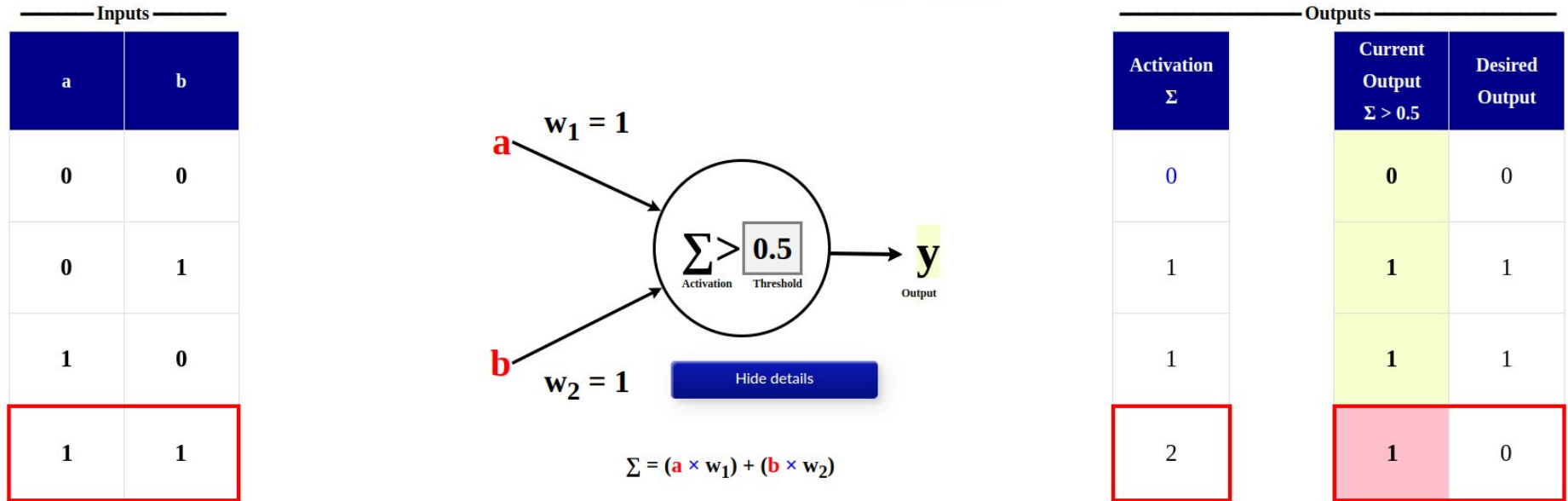
## Step 20

**PROBLEM:** Find weight and threshold values to implement the XOR function.

Is this possible?

# Can't Implement XOR or EQUAL

Surprise! Neither XOR nor EQUAL can be implemented by one linear threshold unit. No combination of weights and threshold can get all four cases correct.



We prove this fact in the “Enrichment” section (see later slides).

## “At Least 2 Of 3” vs. “Exactly 2 Of 3”

Earlier we saw how to get a three-input neuron to output “true” if at least 2 of its 3 inputs were “true”: make all the weights be 1 and the threshold be 1.5.

What if we want the neuron to output “true” if exactly 2 of its 3 inputs are “true”? In other words, it should output “false” if all 3 inputs are “true”.

**PROBLEM:** is it possible to get a single linear threshold unit to do this? Explain your reasoning.

Answer: no, a single linear threshold unit cannot do this. Since two “true” inputs must put the activation above threshold, three “true” inputs will put it even further above threshold, so it will still have to output “true”; it cannot output “false” in this case. The problem is similar to XOR.



# ENRICHMENT

# De Morgan's Laws

NOT (a AND b) equals (NOT a) OR (NOT b)

NOT (a OR b) equals (NOT a) AND (NOT b)

**PROBLEM:** Write out the truth tables to verify that De Morgan's Laws are true.



British mathematician  
Augustus De Morgan:  
1806 - 1871

# Applying De Morgan's Laws

“I can get new sneakers if my sneakers don't fit me or they have no tread left.”

- a is “sneakers fit me”
- b is “sneakers have some tread left”
- can get new sneakers =  $(\text{NOT } a) \text{ OR } (\text{NOT } b)$

What Boolean function does this correspond to?

De Morgan: “ $(\text{NOT } a) \text{ OR } (\text{NOT } b)$ ” is the same as “ $\text{NOT } (a \text{ AND } b)$ ”, or “NAND”.

So use the NAND function to see if you can get new sneakers.

# Applying De Morgan's Laws

What is  $\text{NOT}(a \text{ AND } \text{NOT } b)$ ?

De Morgan:  $\text{NOT}(x \text{ AND } y)$  is the same as  $(\text{NOT } x) \text{ OR } (\text{NOT } y)$ .

Here we have  $x = a$  and  $y = (\text{NOT } b)$ .

So  $\text{NOT}(a \text{ AND } \text{NOT } b)$  is the same as  $(\text{NOT } a) \text{ OR } (\text{NOT } (\text{NOT } b))$ .

But  $\text{NOT}(\text{NOT } b) = b$ , so...

$\text{NOT}(a \text{ AND } \text{NOT } b)$  is the same as  $(\text{NOT } a) \text{ OR } b$ , which is also “ $b \text{ OR } \text{NOT } a$ ”.

# Size of the Truth Table

A truth table lists all the possible combinations of inputs a Boolean function could have, and the desired output for each input.

For one input the truth table has  $2^1 = 2$  rows.

For two inputs the truth table has  $2^2 = 4$  rows.

For three inputs the truth table has  $2^3 = 8$  rows.

For four inputs the truth table has  $2^4 = 16$  rows.

# How Many One-Input Boolean Functions Are There?

For a one-input Boolean function the truth table has  $2^1 = 2$  rows.

There are  $2^2$  distinct one-input Boolean functions:

Input	Outputs (Four Functions)			
x	Always Zero	Identity	NOT	Always One
0	0	0	1	1
1	0	1	0	1


# How Many Two-Input Boolean Functions Are There?

For a two-input Boolean function the truth table has  $2^2 = 4$  rows.

There are  $2^4 = 16$  distinct two-input Boolean functions.

**PROBLEM:** Can you list them all?

Answer:



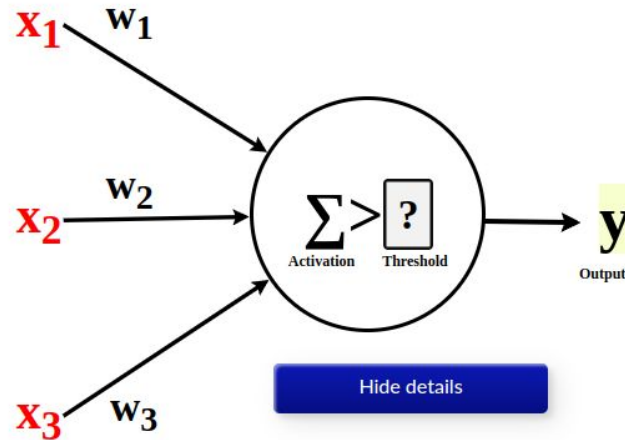
Cannot be computed by a single linear threshold unit

Always Zero	Always a	Always b	AND	OR	a AND NOT b	a OR NOT b	a EQUALS b
Always One	Always NOT a	Always NOT b	NAND	NOR	b OR NOT a	b AND NOT a	a XOR b

# How Many Three-Input Boolean Functions Are There?

The truth table contains  $2^3 = 8$  rows.

So there are  $2^8 = 64$  distinct three-input Boolean functions.



$$\Sigma = (x_1 \times w_1) + (x_2 \times w_2) + (x_3 \times w_3)$$



# Proof That One Neuron Can't Compute XOR

Assume there are weights  $w_1$  and  $w_2$  and threshold  $\theta$  that solve the XOR problem.

a	b
0	0
0	1
1	0
1	1

Activation	a XOR b
0	0
$w_2$	1
$w_1$	1
$w_1 + w_2$	0

Row 1 shows that

$$0 \leq \theta, \text{ so } \theta \geq 0$$

Row 2 shows that

$$w_2 > \theta$$

Row 3 shows that

$$w_1 > \theta$$

Add rows 2 and 3:

$$w_1 + w_2 > 2\theta$$

Since  $\theta \geq 0$ , we know that  $2\theta \geq \theta$ , so:

$$w_1 + w_2 > \theta$$

But row 4 shows that

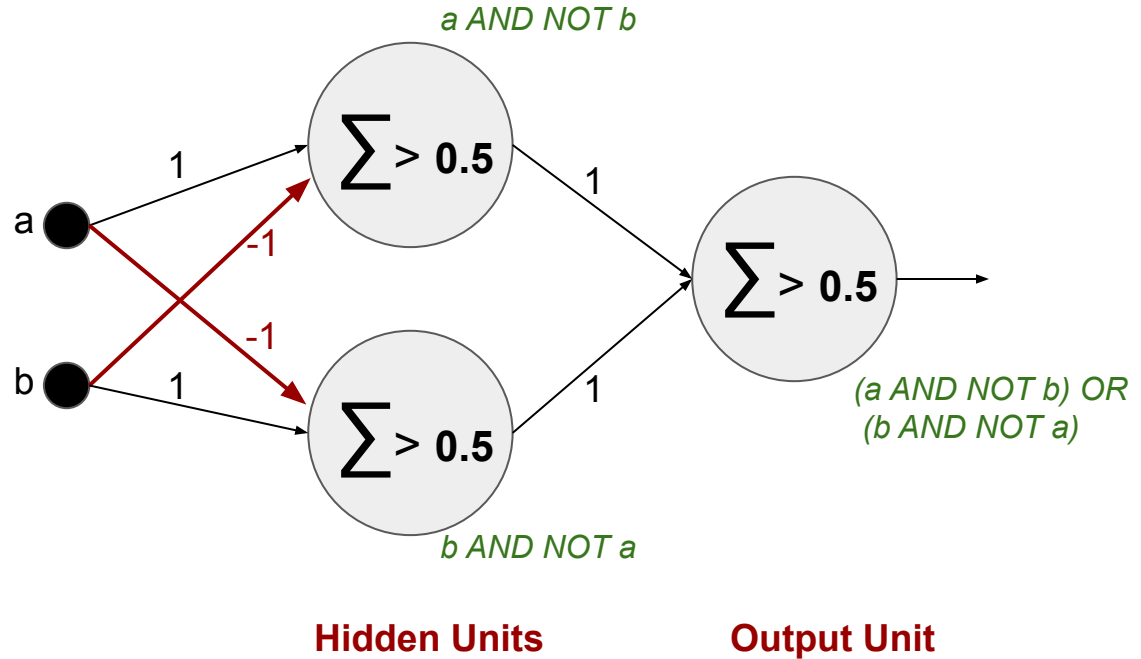
$$w_1 + w_2 \leq \theta$$

Contradiction!

We show this assumption leads to a logical contradiction, so there can be no such solution.

# XOR With Three Neurons: Solution 1

a	b	a XOR b
0	0	0
1	0	1
0	1	1
1	1	0



# XOR With Three Neurons: Solution 2

a	b	a XOR b
0	0	0
1	0	1
0	1	1
1	1	0

