Combining Computational Units

Natalie Parde
UIC CS 421

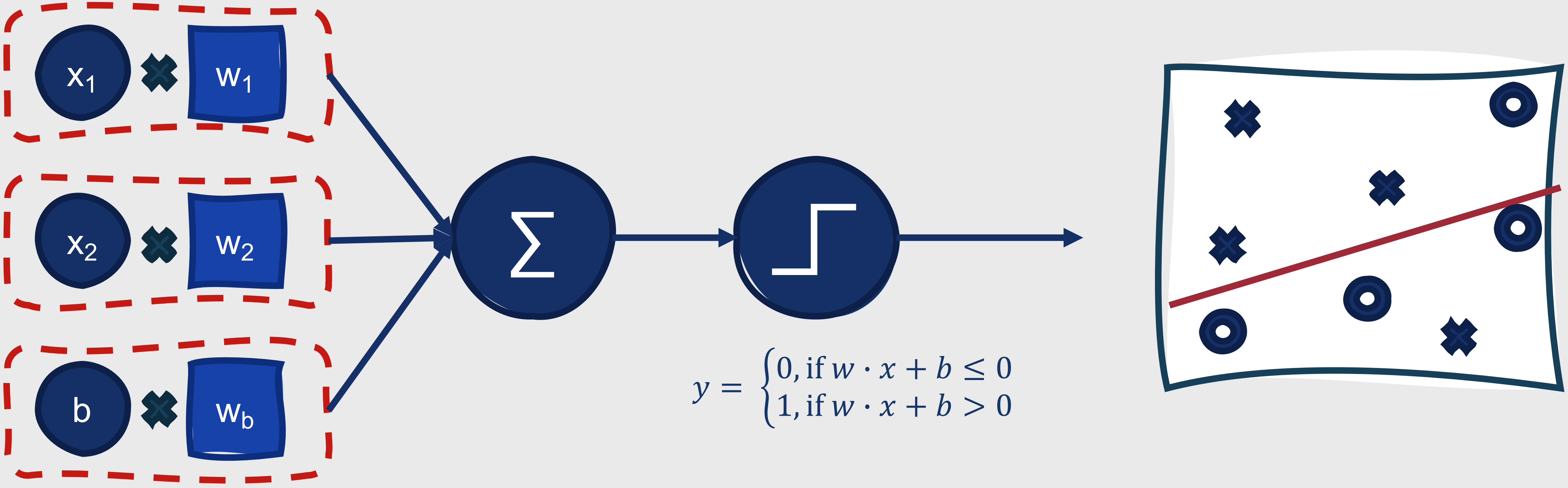
Combining Computational Units

Neural networks are powerful primarily because they are able to **combine multiple computational units into larger networks**

Many problems cannot be solved using a single computational unit

Early example of this: The XOR problem

AND			OR			XOR		
x1	x2	y	x1	x2	y	x1	x2	y
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

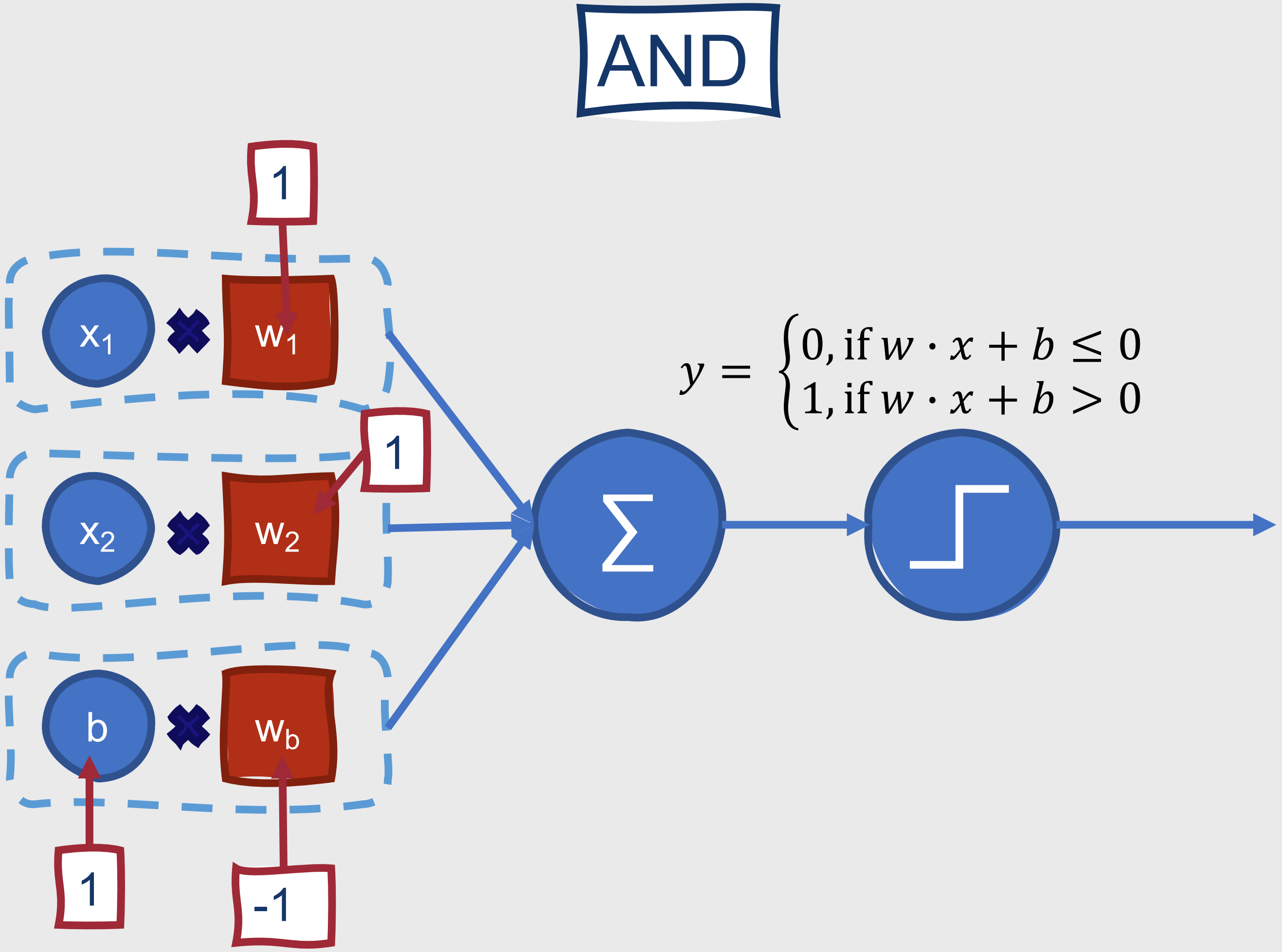


$$y = \begin{cases} 0, \text{if } w \cdot x + b \leq 0 \\ 1, \text{if } w \cdot x + b > 0 \end{cases}$$

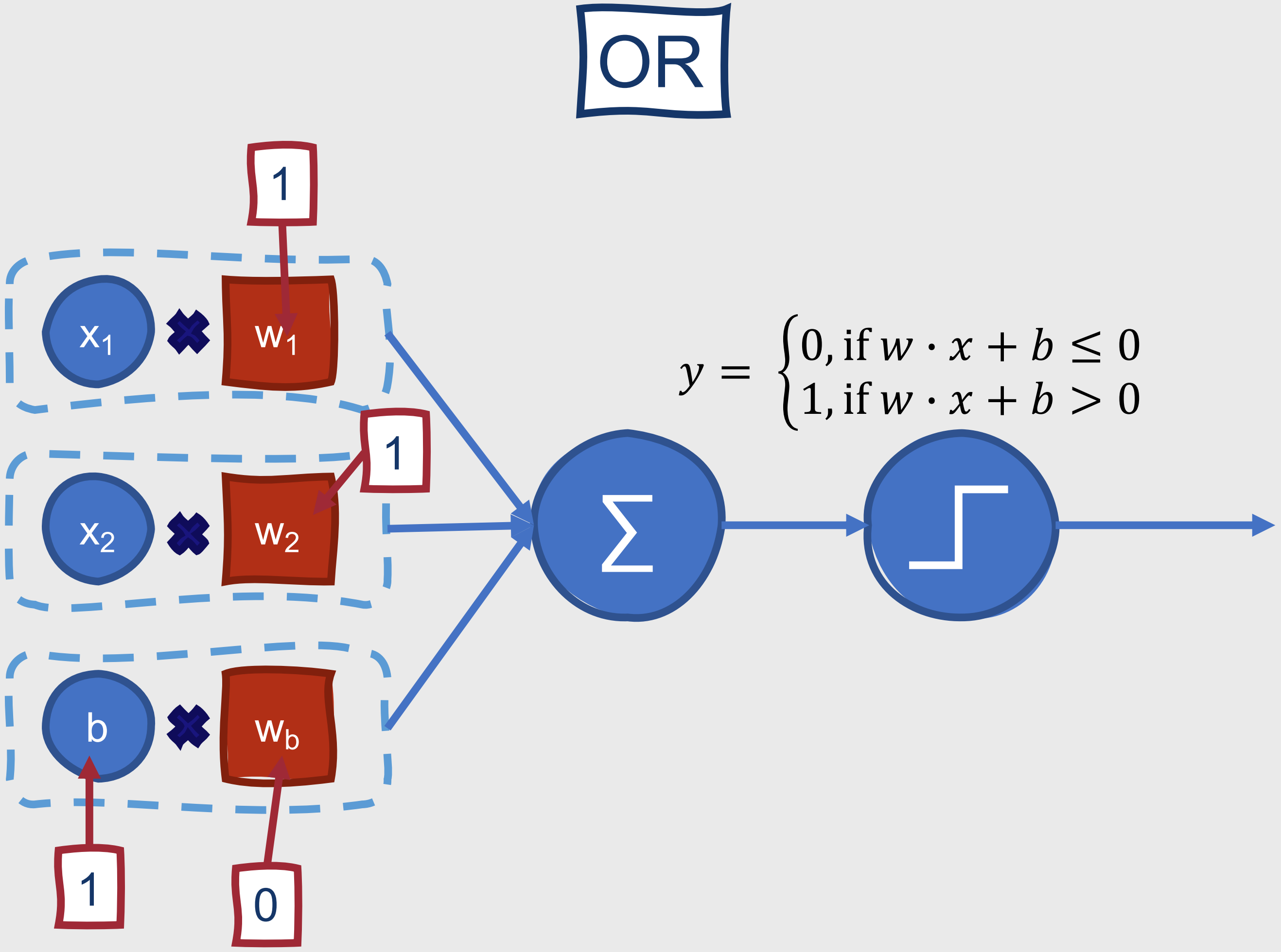
AND and OR can both be solved using a single perceptron.

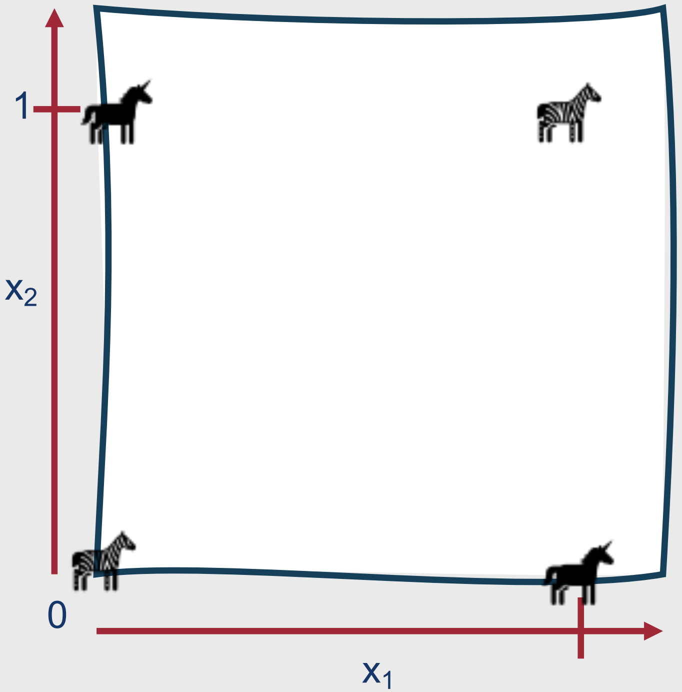
- **Perceptron:** A function that outputs a binary value based on whether the product of its inputs and associated weights surpasses a threshold
  - Learns this threshold iteratively by trying to find the boundary that is best able to distinguish between data of different categories

It's easy to compute AND and OR using perceptrons.



It's easy to compute AND and OR using perceptrons.



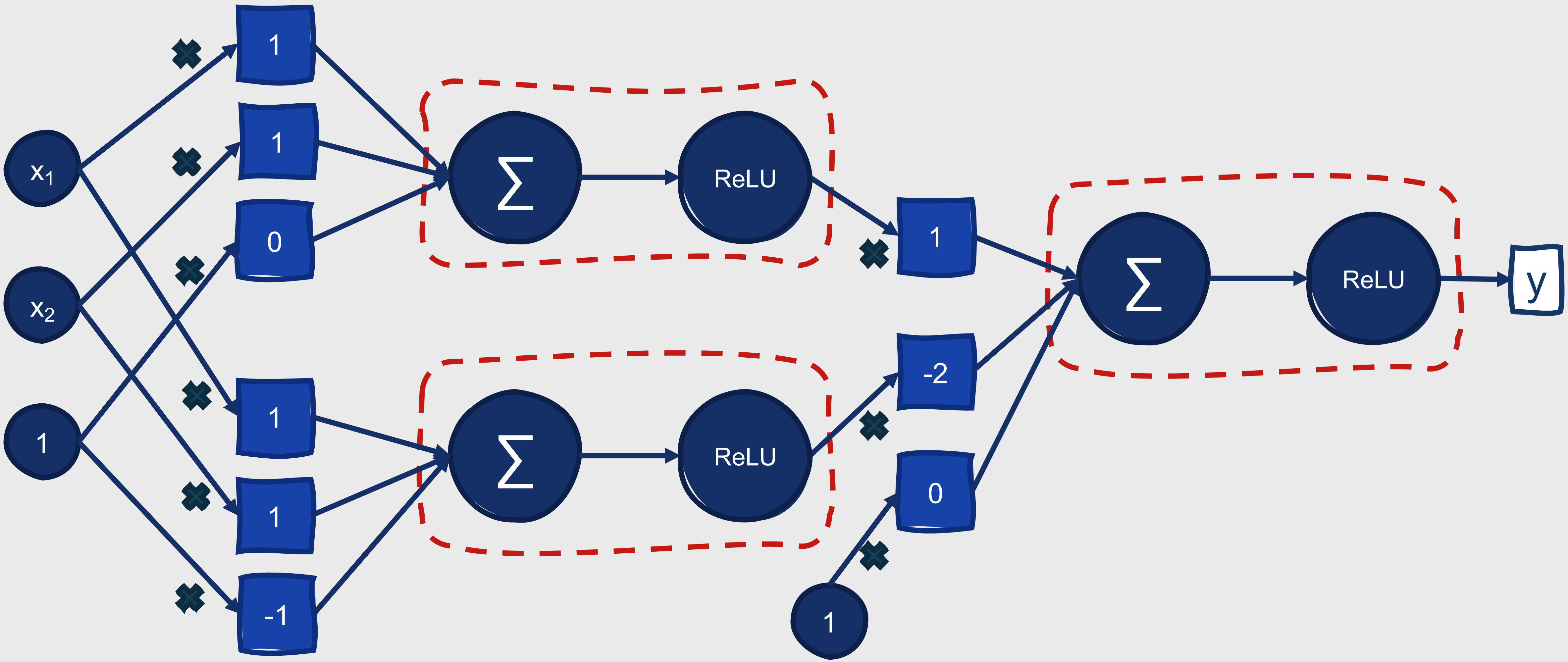


AND			OR			XOR		
x1	x2	y	x1	x2	y	x1	x2	y
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

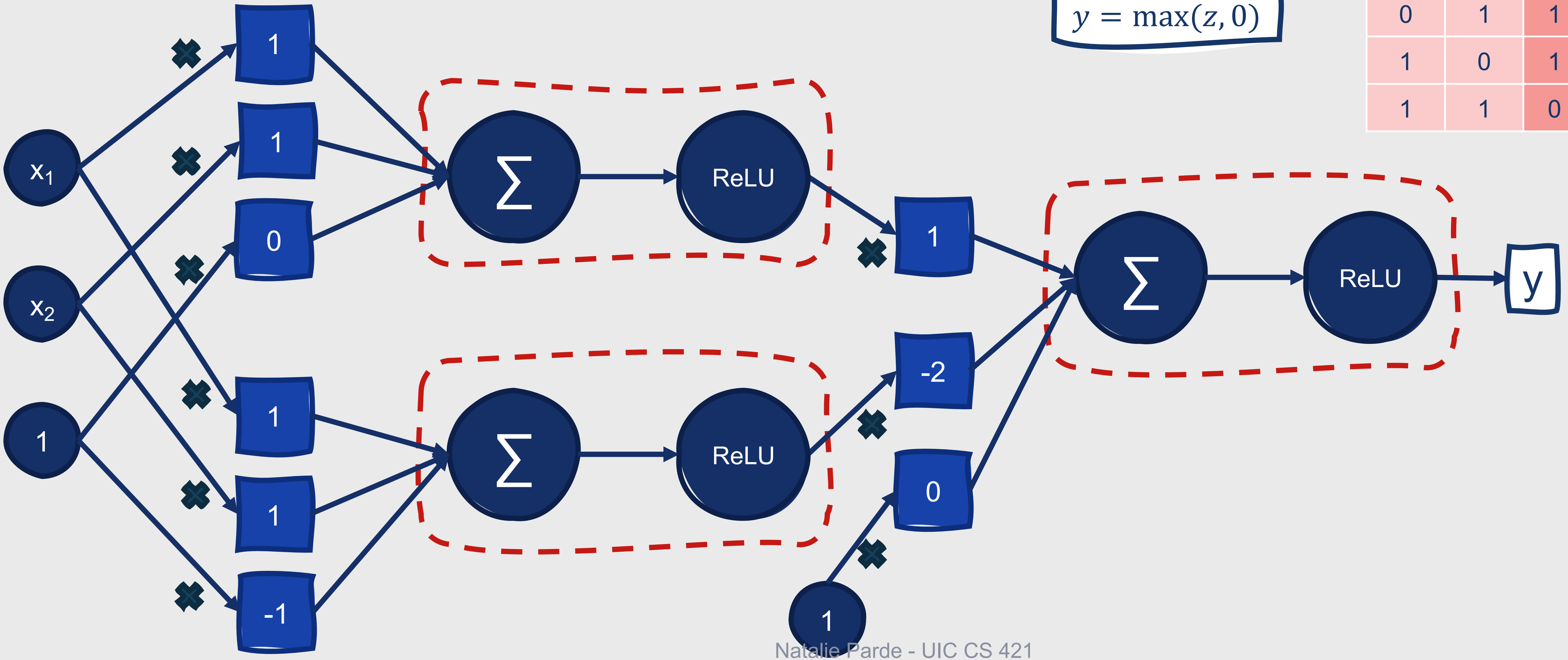
However, it's impossible to compute XOR using a single perceptron.

- Why?
  - Perceptrons are **linear classifiers**
  - XOR is not a **linearly separable function**

The only successful way to compute XOR is by combining these smaller units into a larger network.



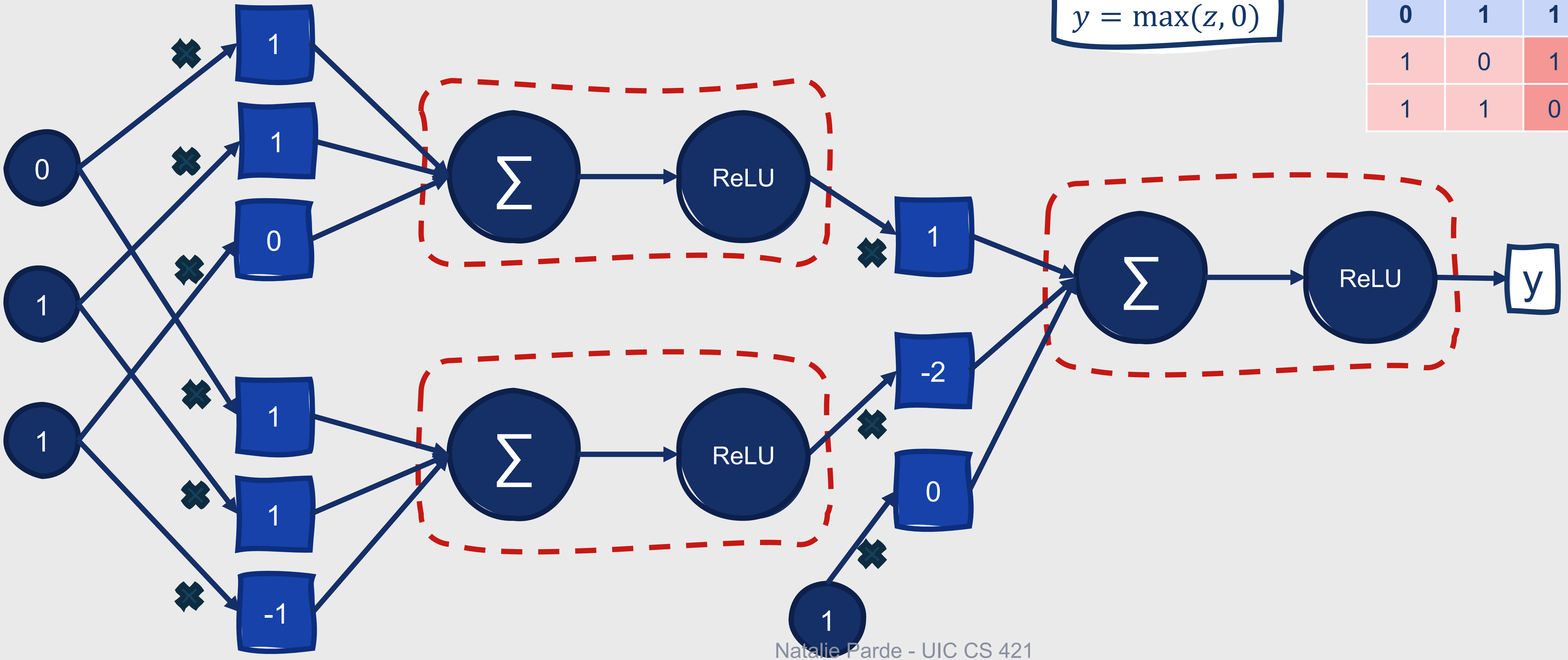
Truth Table Examples: XOR



$y = \max(z, 0)$

XOR		
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table Examples: XOR



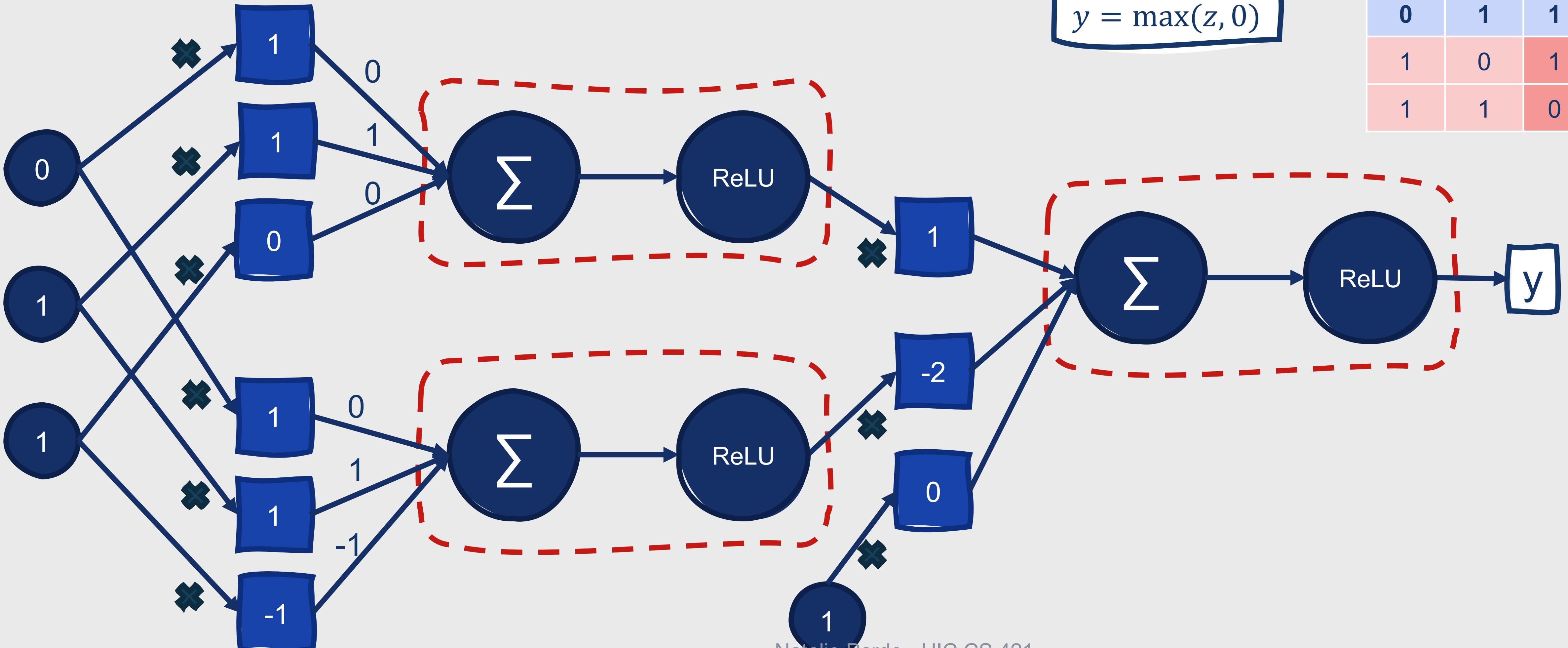
$$y = \max(z, 0)$$

XOR		
x1	**x2**	**y**
0	0	0
0	**1**	**1**
1	0	1
1	1	0

Truth Table Examples: XOR

XOR		
x1	**x2**	**y**
0	0	0
0	**1**	**1**
1	0	1
1	1	0

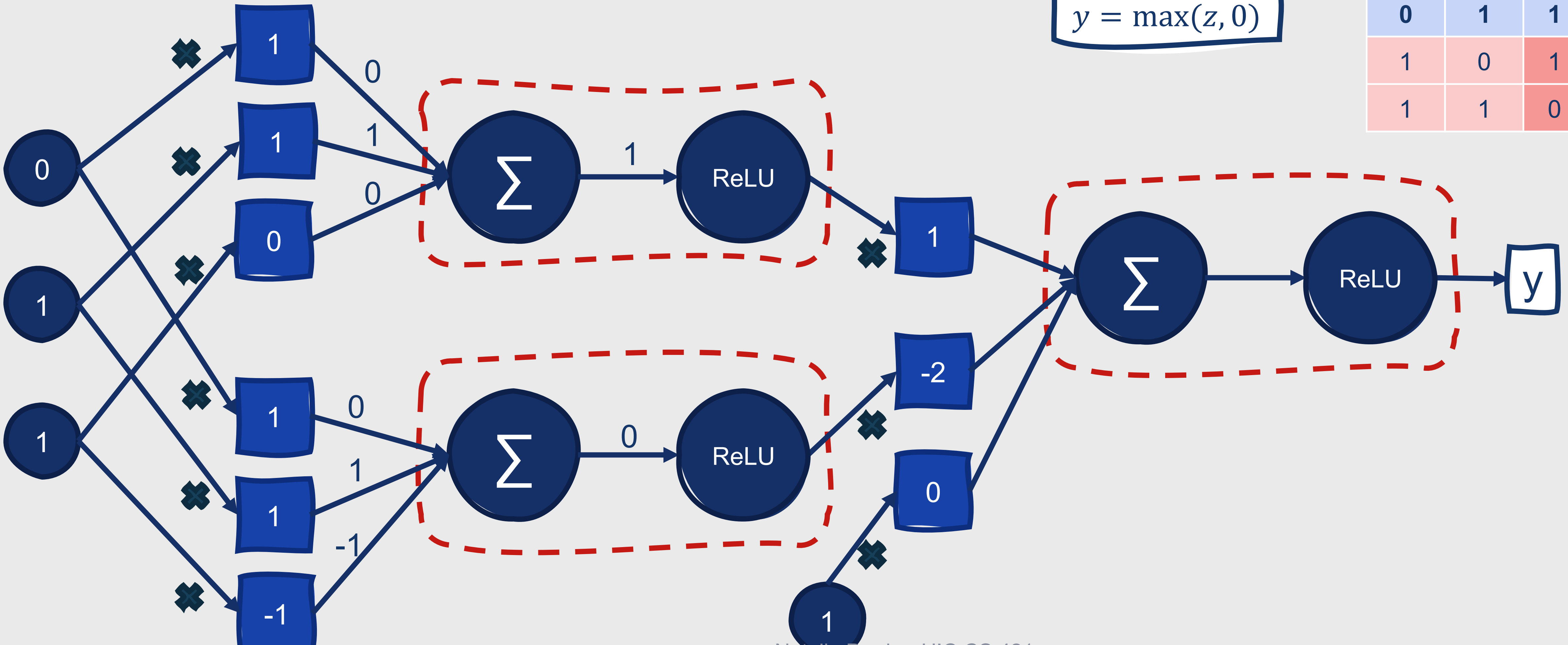
$y = \max(z, 0)$



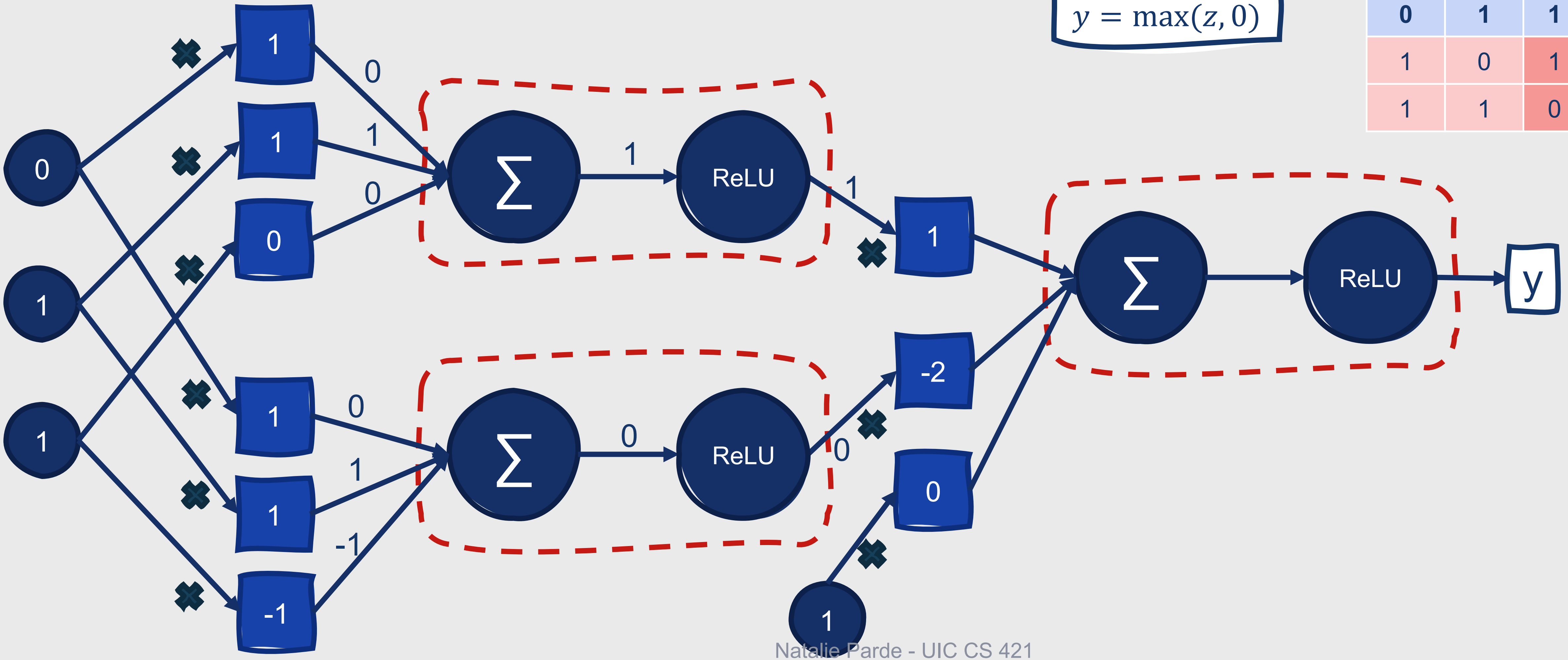
Truth Table Examples: XOR

XOR		
x1	**x2**	**y**
0	0	0
0	**1**	**1**
1	0	1
1	1	0

$$y = \max(z, 0)$$



Truth Table Examples: XOR



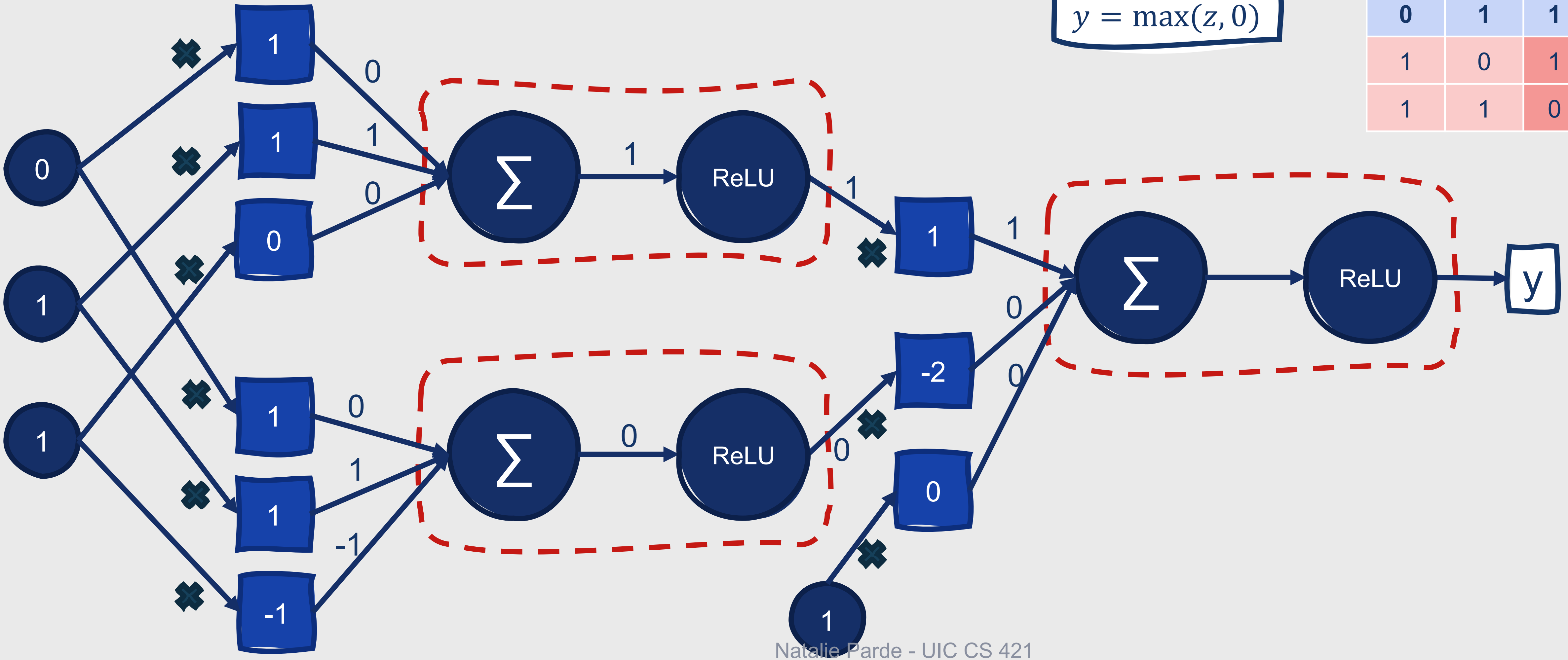
$y = \max(z, 0)$

XOR		
x1	**x2**	**y**
0	0	0
0	**1**	**1**
1	0	1
1	1	0

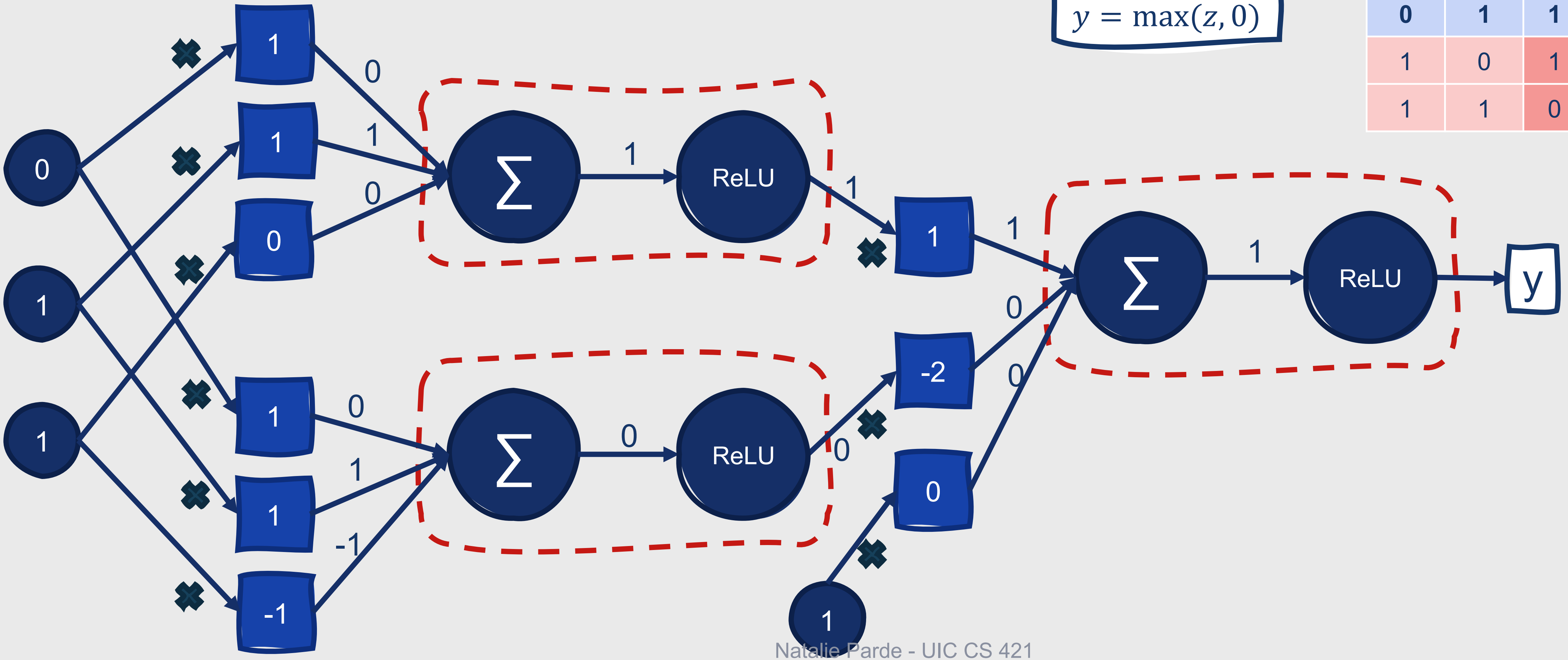
Truth Table Examples: XOR

XOR		
x1	**x2**	**y**
0	0	0
0	**1**	**1**
1	0	1
1	1	0

$$y = \max(z, 0)$$



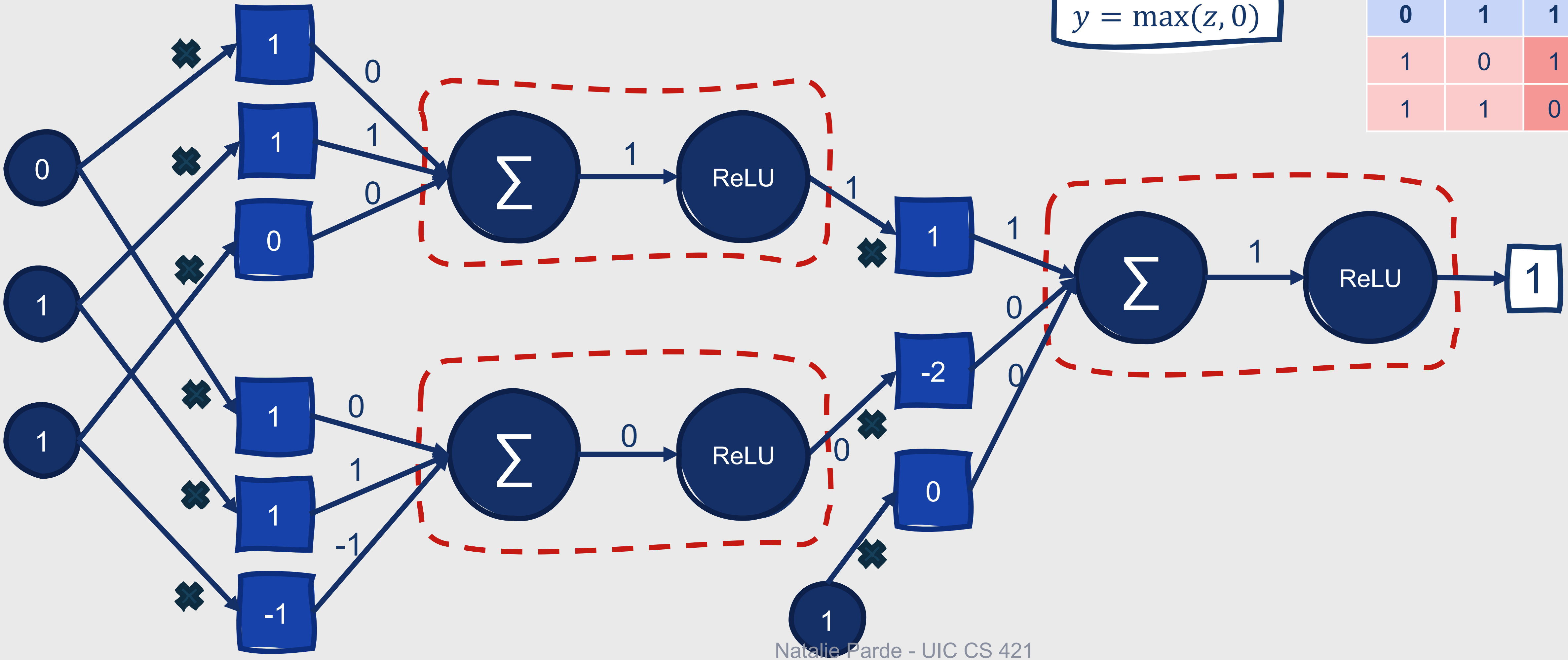
Truth Table Examples: XOR



$y = \max(z, 0)$

XOR		
x1	x2	y
0	0	0
0	**1**	**1**
1	0	1
1	1	0

Truth Table Examples: XOR



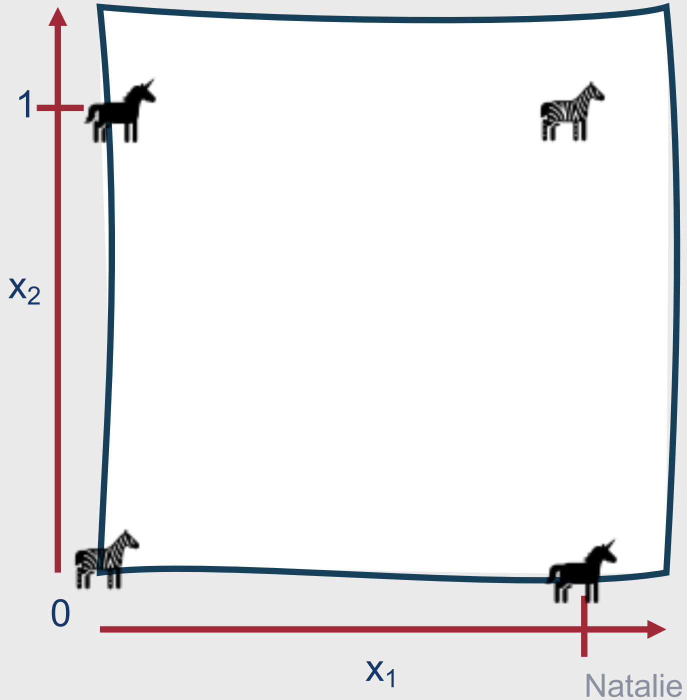
$$y = \max(z, 0)$$

XOR		
x1	**x2**	**y**
0	0	0
0	**1**	**1**
1	0	1
1	1	0

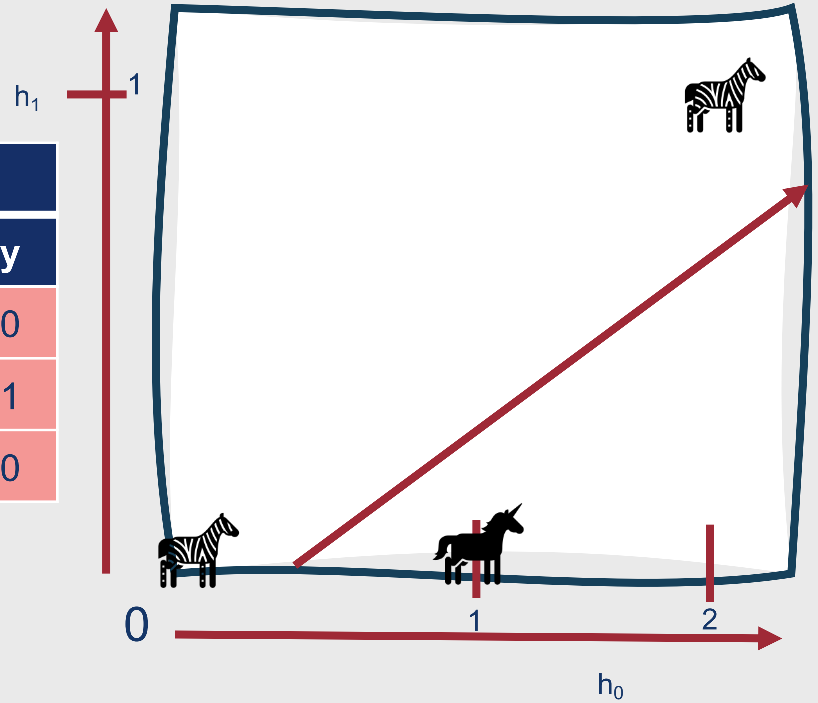
Why does this work?

- When computational units are combined, the outputs from each successive layer provide **new representations** for the input
- These new representations are **linearly separable**

XOR		
x1	**x2**	**y**
0	0	0
0	1	1
1	0	1
1	1	0



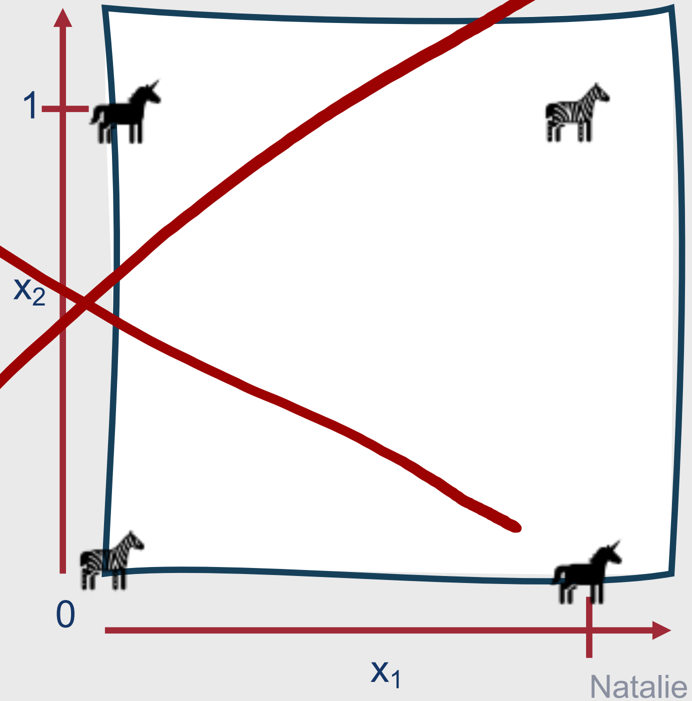
XOR		
h0	**h1**	**y**
0	0	0
1	0	1
2	1	0



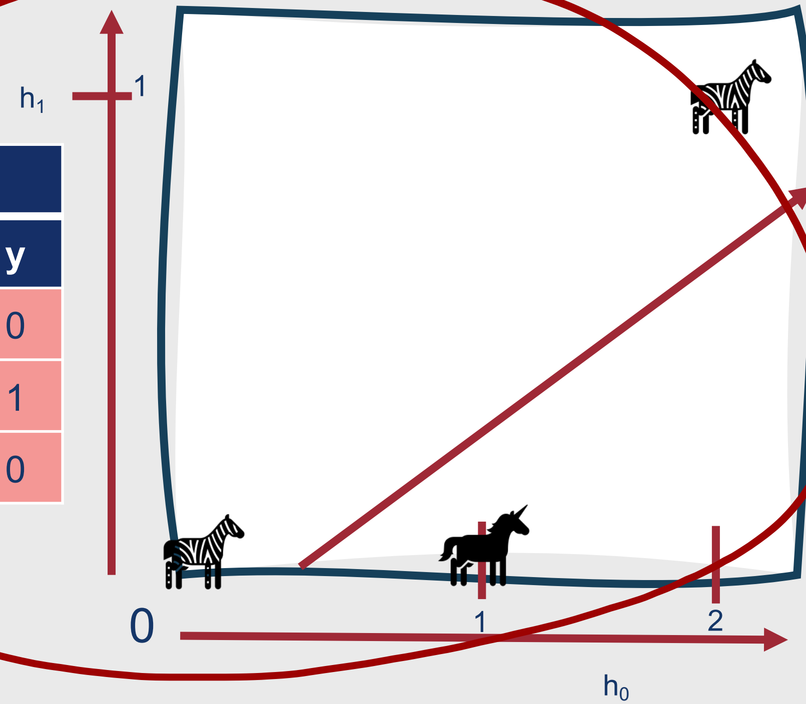
Why does this work?

- When computational units are combined, the outputs from each successive layer provide **new representations** for the input
- These new representations are **linearly separable**

XOR		
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0



XOR		
h0	h1	y
0	0	0
1	0	1
2	1	0



Combining Computational Units

- In our XOR example, we manually assigned weights to each unit
- In real-world examples, these weights are learned automatically using a **backpropagation** algorithm
- Thus, the network is able to learn a useful representation of the input training data on its own
  - Key advantage of neural networks