Artificial Neural Networks

Dr. Muhammad Aqib

University Institute of Information Technology

PMAS-Arid Agriculture University Rawalpindi

Activation Functions in Neural Networks

Contents

- Activation Functions
  - Linear Activation
  - Sigmoid Activation
  - Rectified Linear Activation
- Why use activation functions at all?
- Softmax Activation

Activation Function

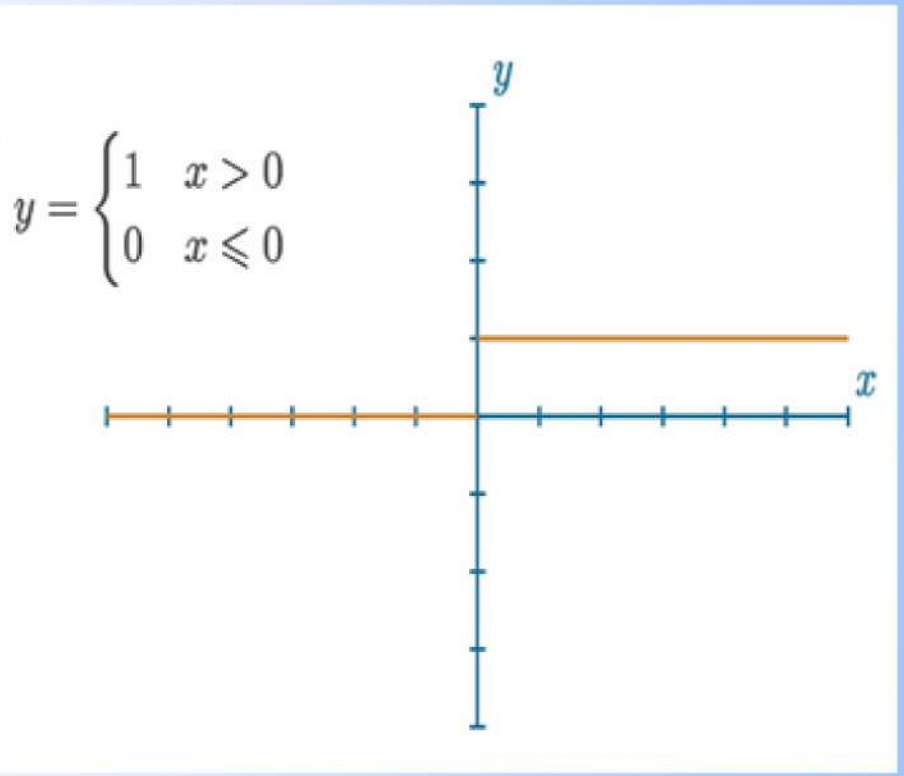
- Activation function is applied to the output of a neuron.
- As a result of this, output of layer(s) is modified.
- If the activation function is non-linear, it usually takes two or more hidden layers to map nonlinear functions.
- Neural Network will have two types of activation functions.
  - Activation Function used in hidden layer(s)
  - Activation function used in output layer(s)
- Usually, the activation function used for hidden neurons will be same for all of them (but it is not hard rule).

Step Function

- The purpose this activation function serves is to mimic a neuron "firing" or "not firing" based on input information.
- The simplest version of this is a step function.
- In a single neuron, if the

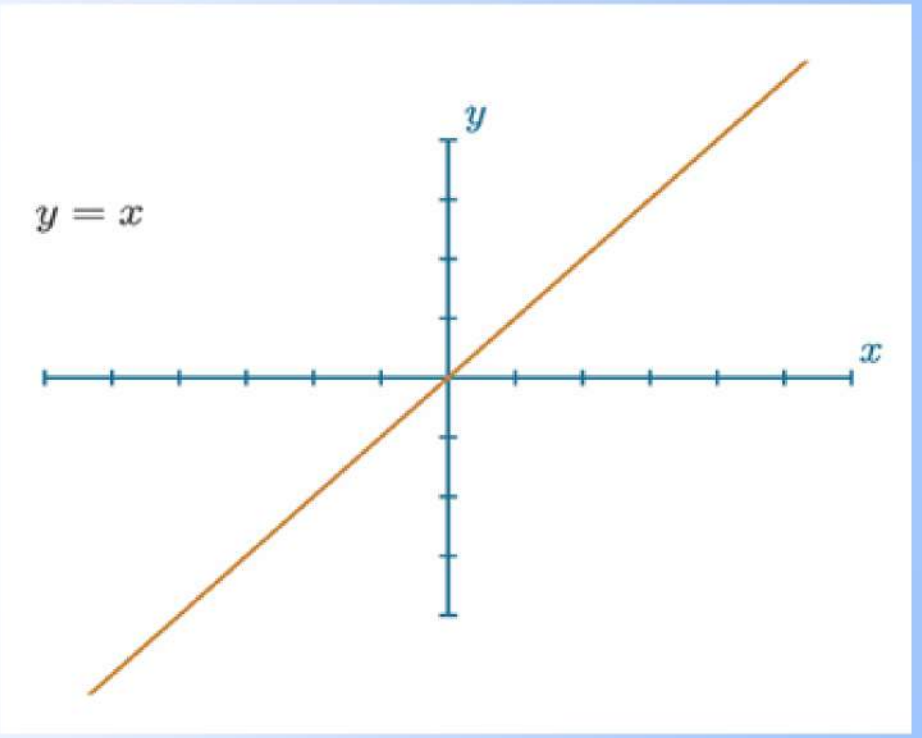
  weights × inputs + bias

  results in a value greater than 0, the neuron will fire and output a 1; otherwise, it will output a 0.



Linear Activation Function

- A linear function is simply the equation of a line. It will appear as a straight line when graphed.
- Here y = x and the output value equals the input.
- This activation function is usually applied to the last layer's output in the case of a regression model.
  - A model that outputs a scalar value instead of classification.



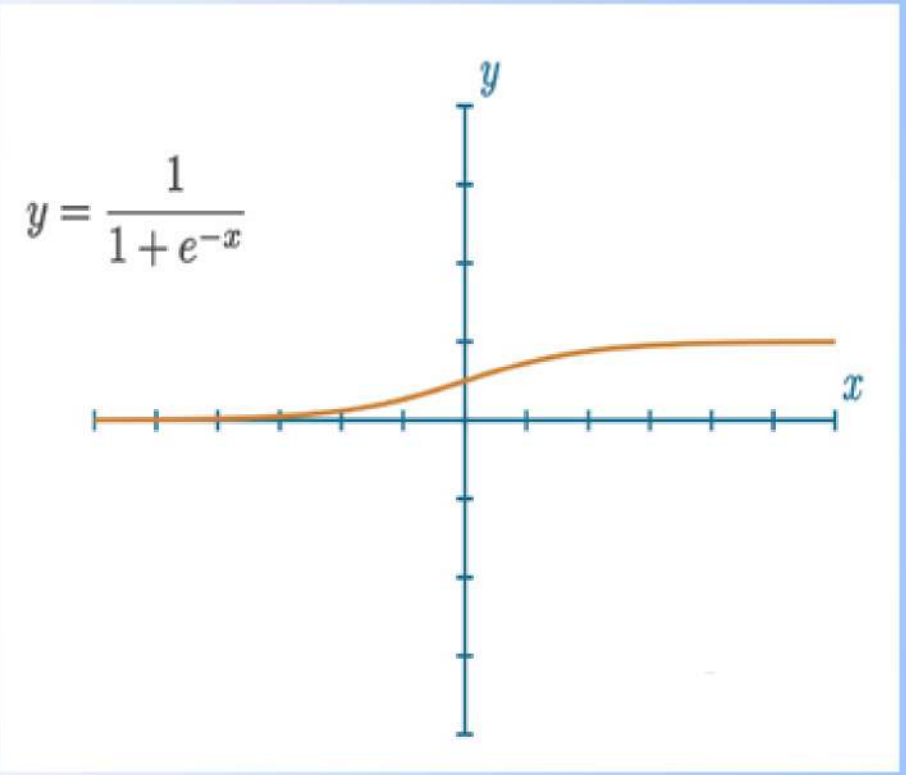
Sigmoid Activation Function

- The problem with the step function is that it is not very informative.
- While training and network optimization, the way an optimizer works is by assessing individual impacts that weights and biases have on a network's output.
- The problem with a step function is that it's less clear to the optimizer what these impacts are because there's very little information gathered from this function.
- It's either on (1) or off (0). It's hard to tell how "close" this step function was to activating or deactivating.
- Thus, when it comes time to optimize weights and biases, it's easier for the optimizer if we have activation functions that are more granular and informative.
- This is where Sigmoid serves it purpose.

Sigmoid Activation Function

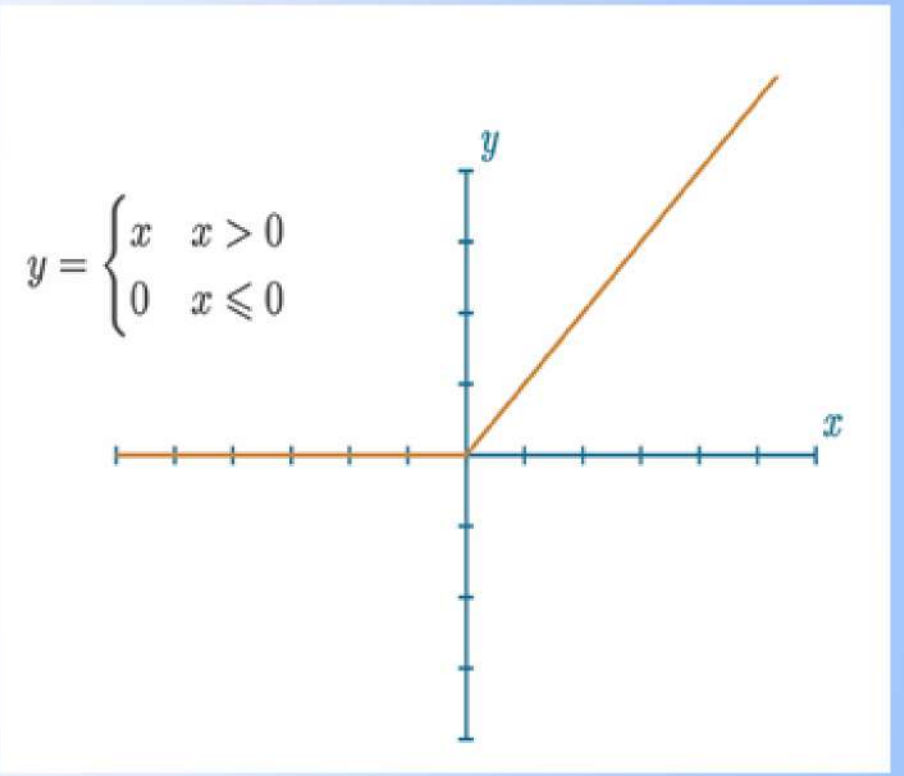
- Sigmoid function returns a value in the range of 0 for negative infinity, through 0.5 for the input of 0, and to 1 for positive infinity.

- The output from the Sigmoid function, being in the range of 0 to 1, also works better with neural networks, especially compared to the range of the negative to the positive infinity and adds nonlinearity.



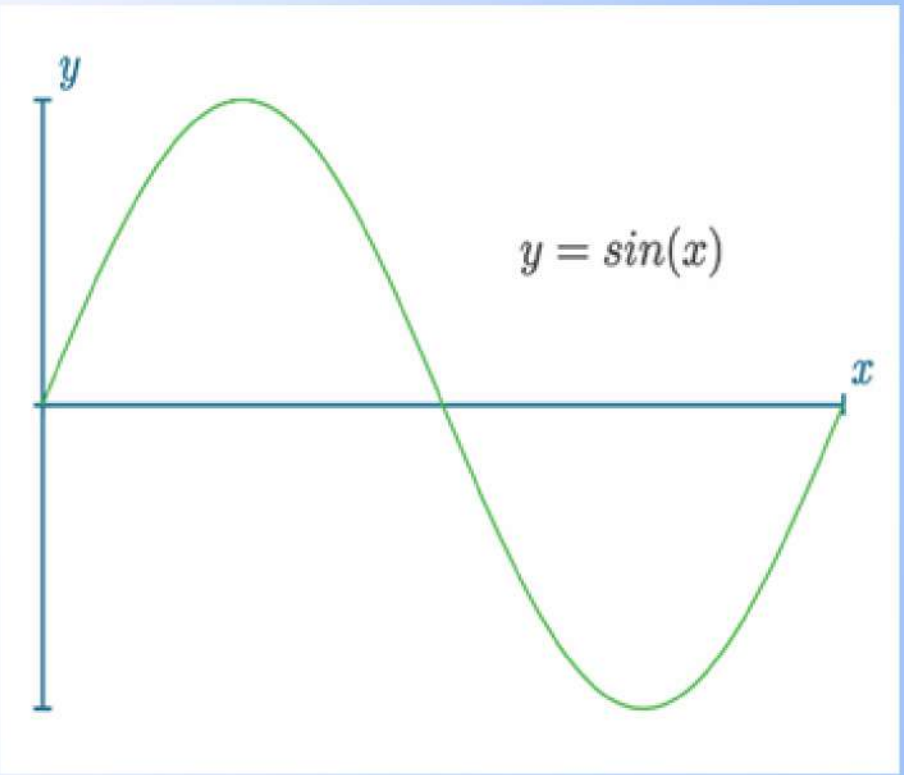
Rectified Linear Activation Function

- The rectified linear activation function is simpler than the sigmoid. It's quite literally y = x, clipped at 0 from the negative side. If x is less than or equal to 0, then y is 0, otherwise, y is equal to x.
- This simple yet powerful activation function is the most widely used activation function for reasons like speed and efficiency.
- Sigmoid is much more challenging to compute than the ReLU.
- The ReLU activation function is extremely close to being a linear activation.
- Functions while remaining nonlinear, due to that bend after 0.



Why We Use Activation Functions?

- In most cases, for a neural network to fit a nonlinear function, we need it to contain two or more hidden layers, and we need those hidden layers to use a nonlinear activation function.
- What's a nonlinear function?
- A nonlinear function cannot be represented well by a straight line, such as a sine function.



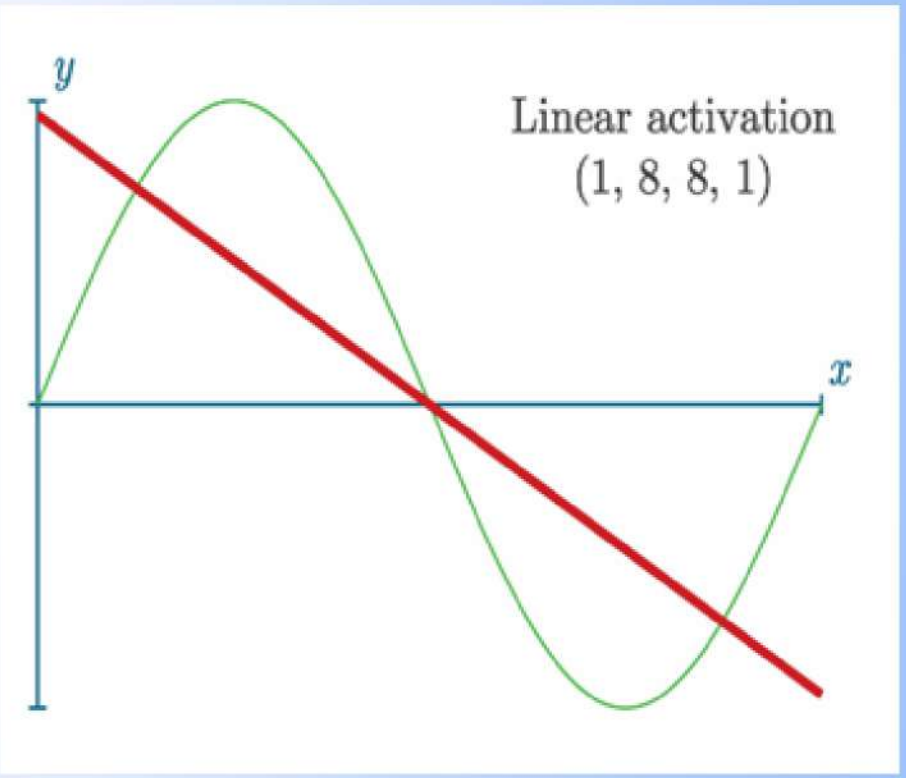
Why We Use Activation Functions? (cont.)

- While there are certainly problems in life that are linear in nature, for example, trying to figure out the cost of some number of shirts, and we know the cost of an individual shirt, and that there are no bulk discounts, then the equation to calculate the price of any number of those products is a linear equation.
- Other problems in life are not so simple, like the price of a home. The number of factors that come into play, such as size, location, time of year attempting to sell, number of rooms, yard, neighborhood, and so on, makes the pricing of a home a nonlinear equation.
- Many of the more interesting and hard problems of our time are nonlinear.
- The main attraction for neural networks has to do with their ability to solve nonlinear problems.

Why We Use Activation Functions? (cont.)

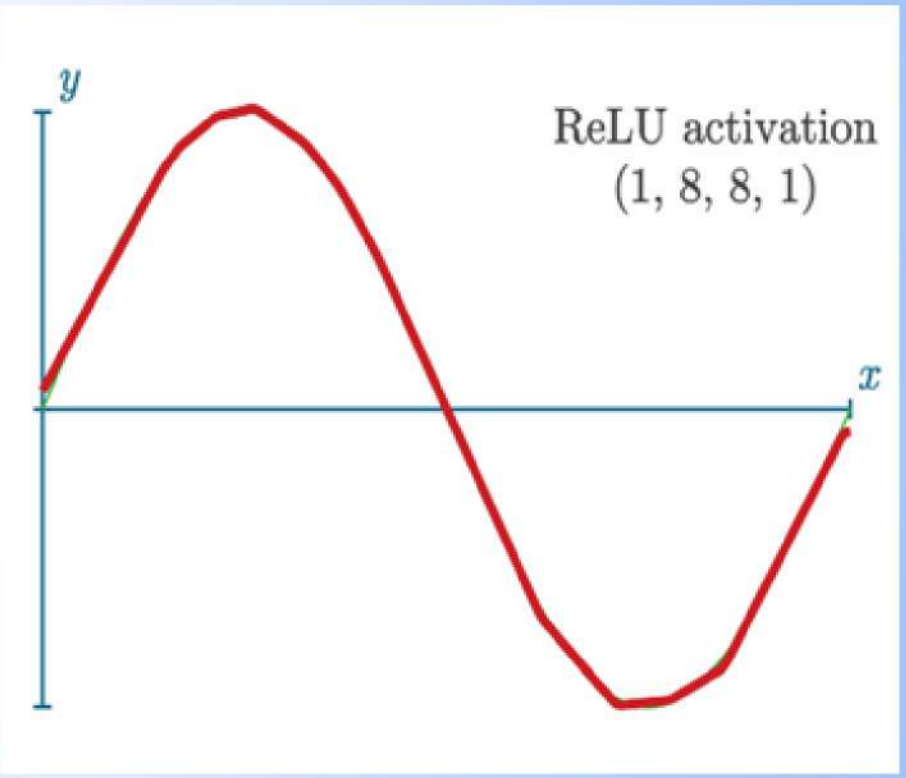
- Let's consider a situation where neurons have no activation function, which would be the same as having an activation function of y = x.

- With this linear activation function in a neural network with 2 hidden layers of 8 neurons each, the result of training this model will look like.



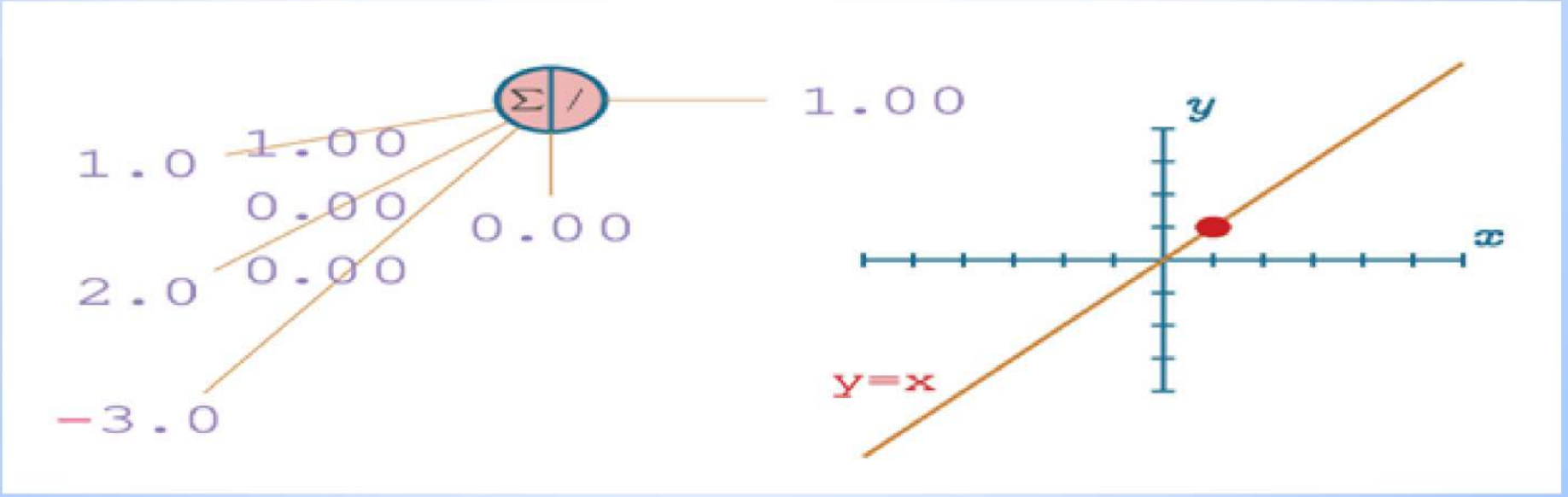
Why We Use Activation Functions? (cont.)

- When using the same 2 hidden layers of 8 neurons each with the rectified linear activation function, we see the following result after training.



Linear Activations in Hidden Layers (cont.)

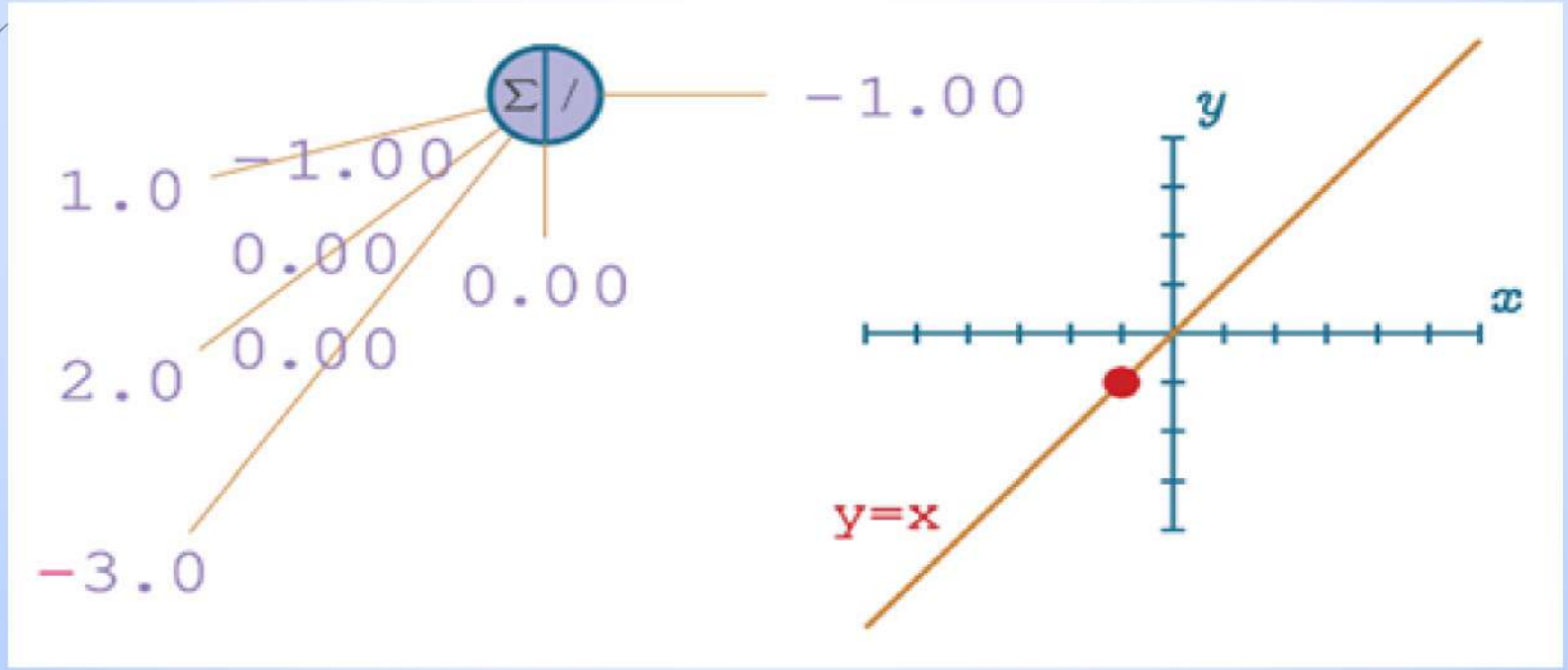
- Let's revisit the linear activation function of y = x, and let's consider this on a singular neuron level.
- Given values for weights and biases, what will the output be for a neuron with a y = x activation function?



Linear Activations in Hidden Layers (cont.)

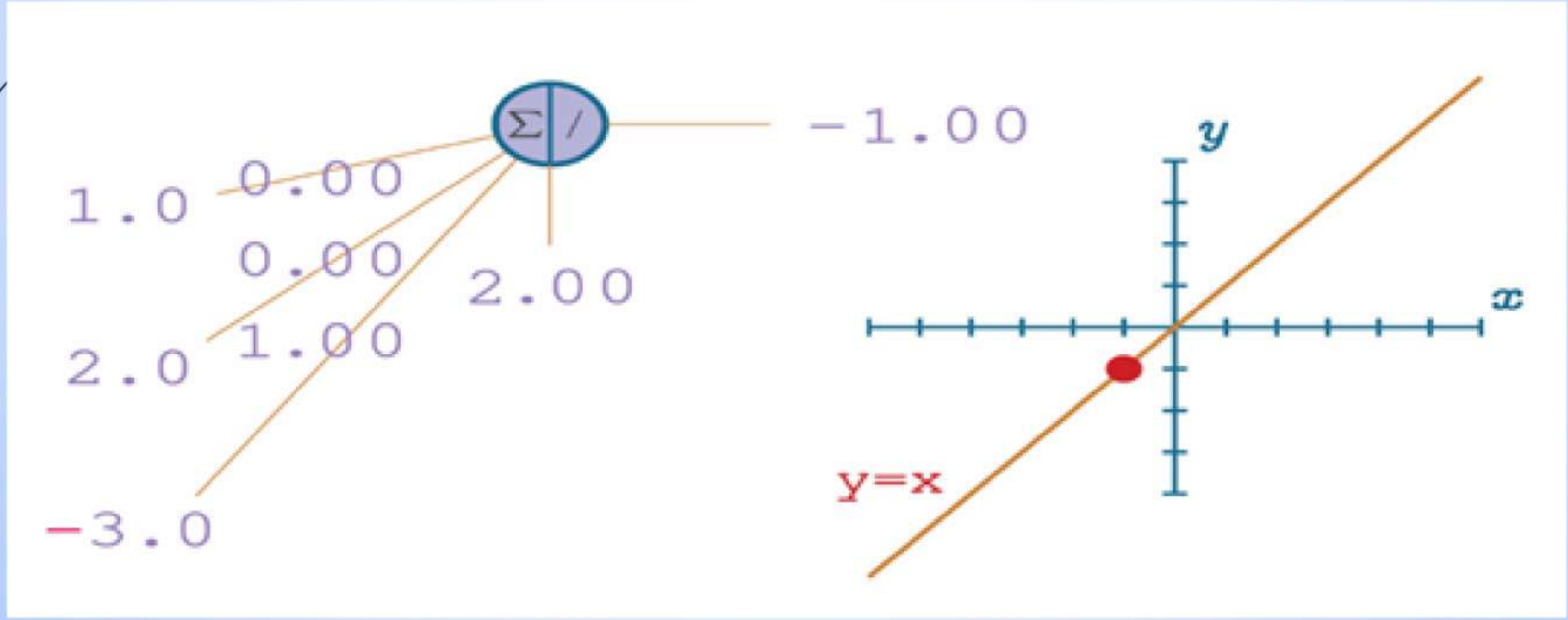
- As we continue to tweak with weights, updating with a negative number, we get a different output.

- New because of y = x, new calculated value will be the output.



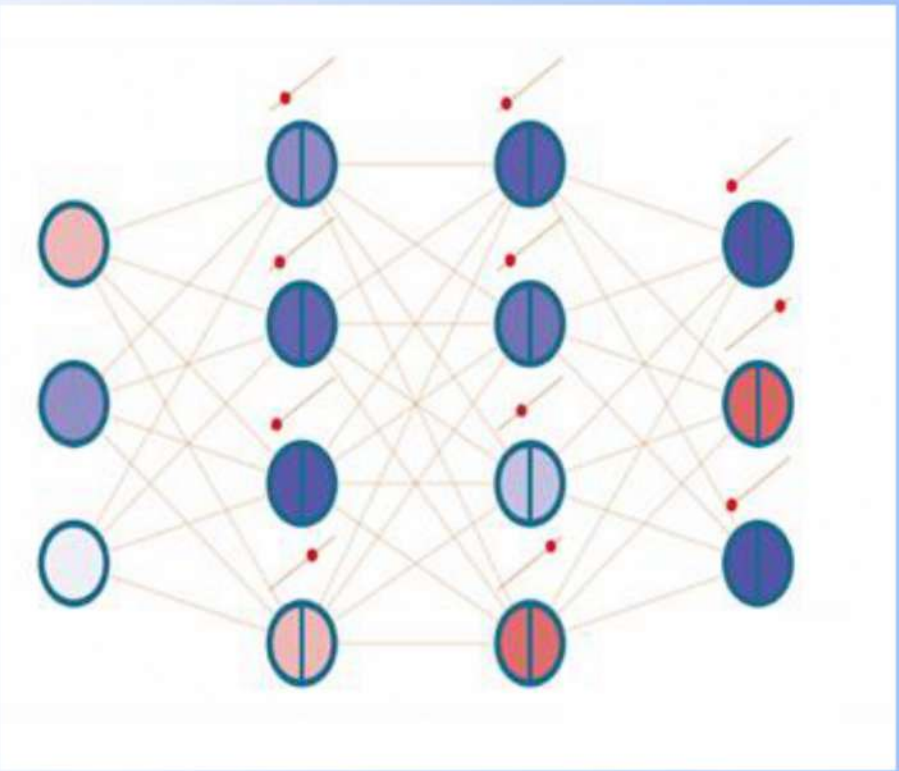
Linear Activations in Hidden Layers (cont.)

- Updating weights and additionally a bias, linear activation function produces new output.



Linear Activations in Hidden Layers (cont.)

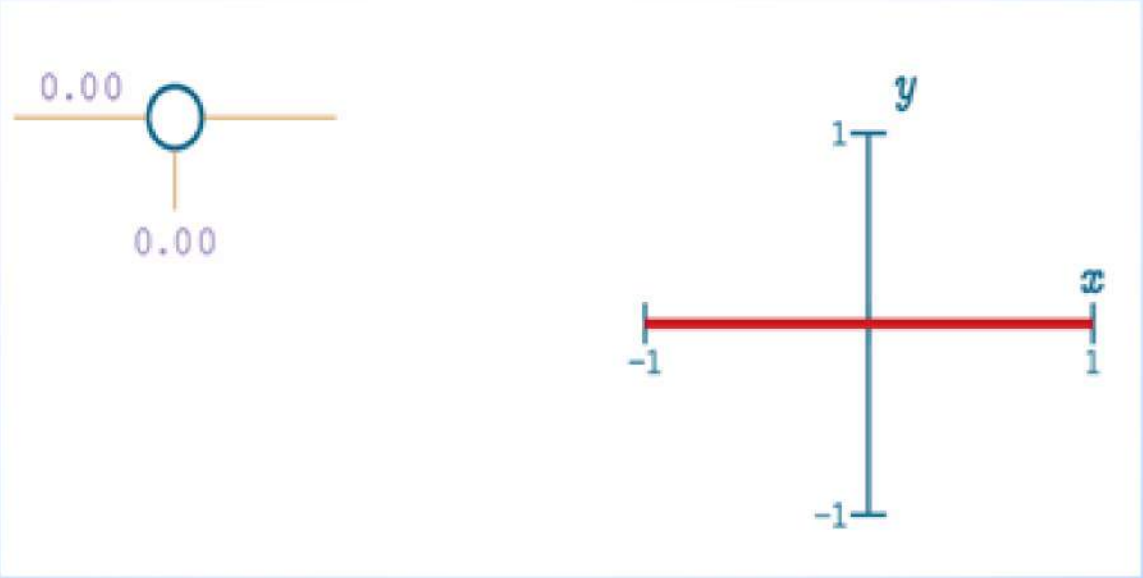
- No matter what we do with this neuron's weights and biases, the output of this neuron will be perfectly linear to y = x of the activation function. This linear nature will continue throughout the entire network.
- No matter what we do, this network can only depict linear relationships if we use linear activation functions.
- Each neuron in each layer acts linearly, so the entire network is a linear function as well.



ReLU Activation Function

- Let's start again with a single neuron. We begin with both, a weight of 0 and a bias of 0.

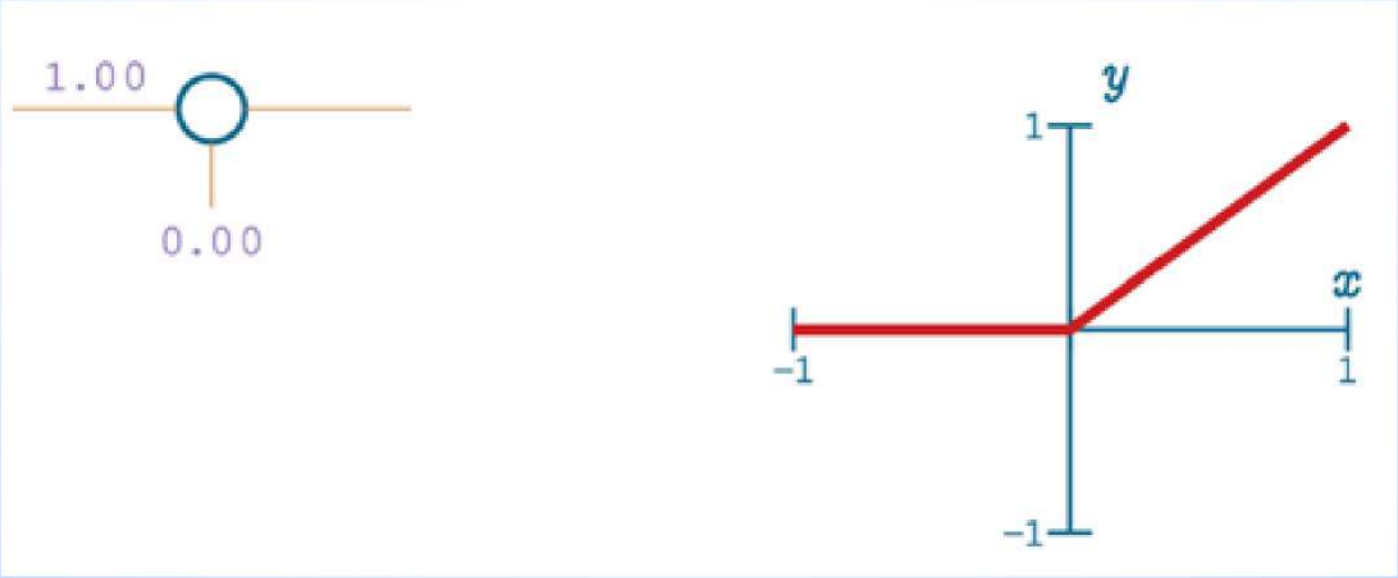
- In this case, no matter what input we pass, the output of this neuron will always be a 0, because the weight is 0, and there's no bias.



ReLU Activation Function (cont.)

- Let's set the weight to be 1:

- Now it looks just like the basic rectified linear function.



ReLU Activation Function (cont.)

- Now let's set the bias to 0.50:

- We can see that, in this case, with a single neuron, the bias offsets the overall function's activation point horizontally. By increasing bias, we're making this neuron activate earlier.