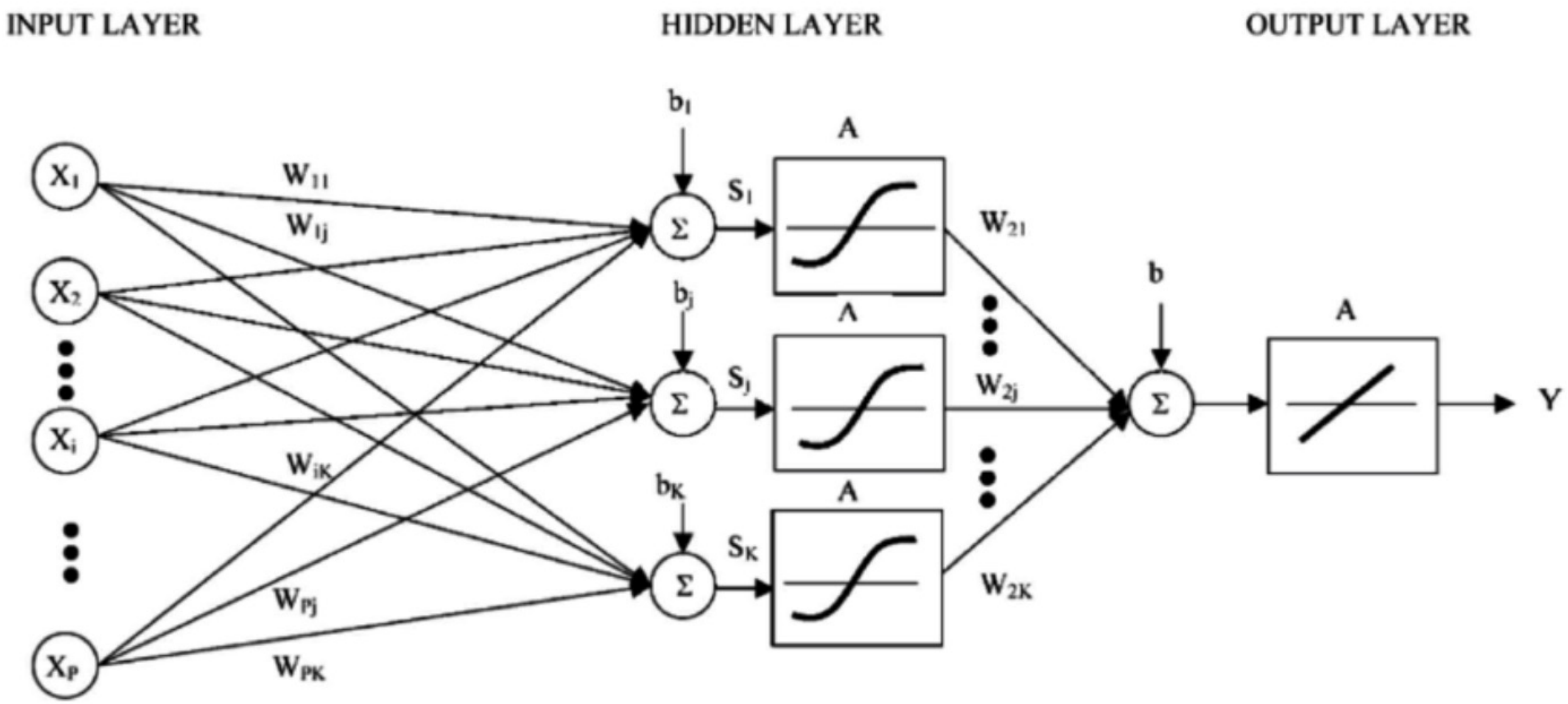
Deep Learning

Elizabeth Savochkina | 14th June

Neural Network Outlook

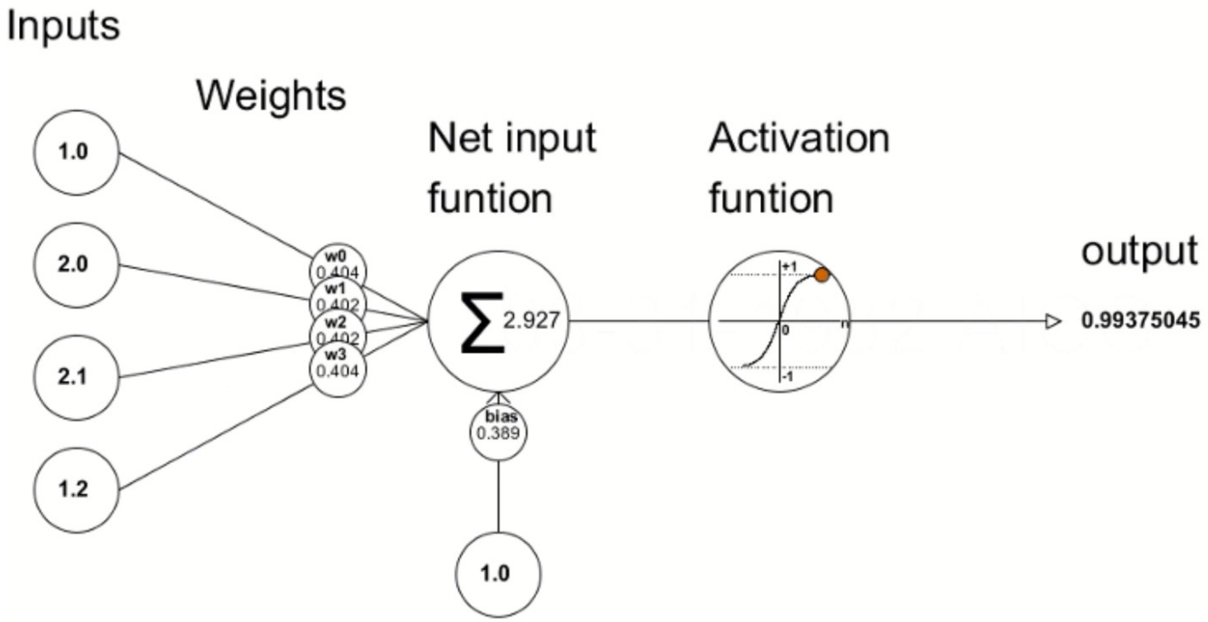


Bias - allows the **activation function** to be **shifted to the left or right,** to better fit data

- Only influences the output values, it doesn't interact with the actual input data.
- The role of bias isn't to act as a threshold, but to help ensure the output **best fits the incoming signal**.

Parts of a Single Neuron

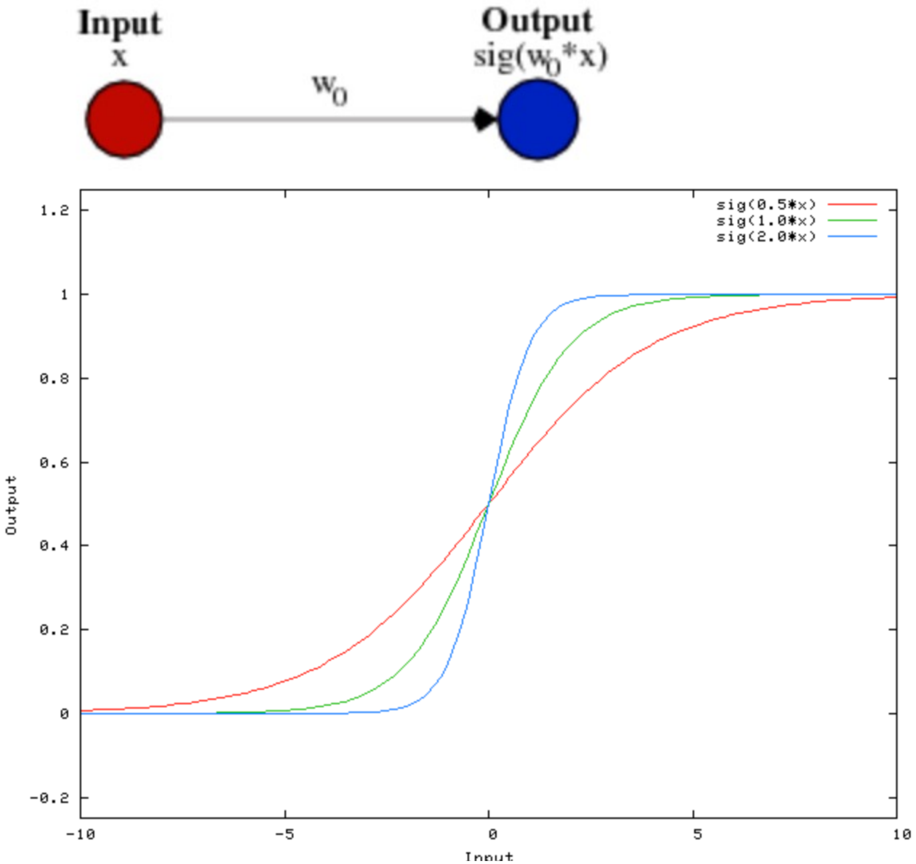
- Like our neurons, the artificial neuron is triggered when encountered sufficient stimuli.
- The neuron combines input from the ***data with a set of coefficients (weights),*** that either amplify or dampen that input, ***assigning significance*** to inputs for the task the algorithm is trying to learn.

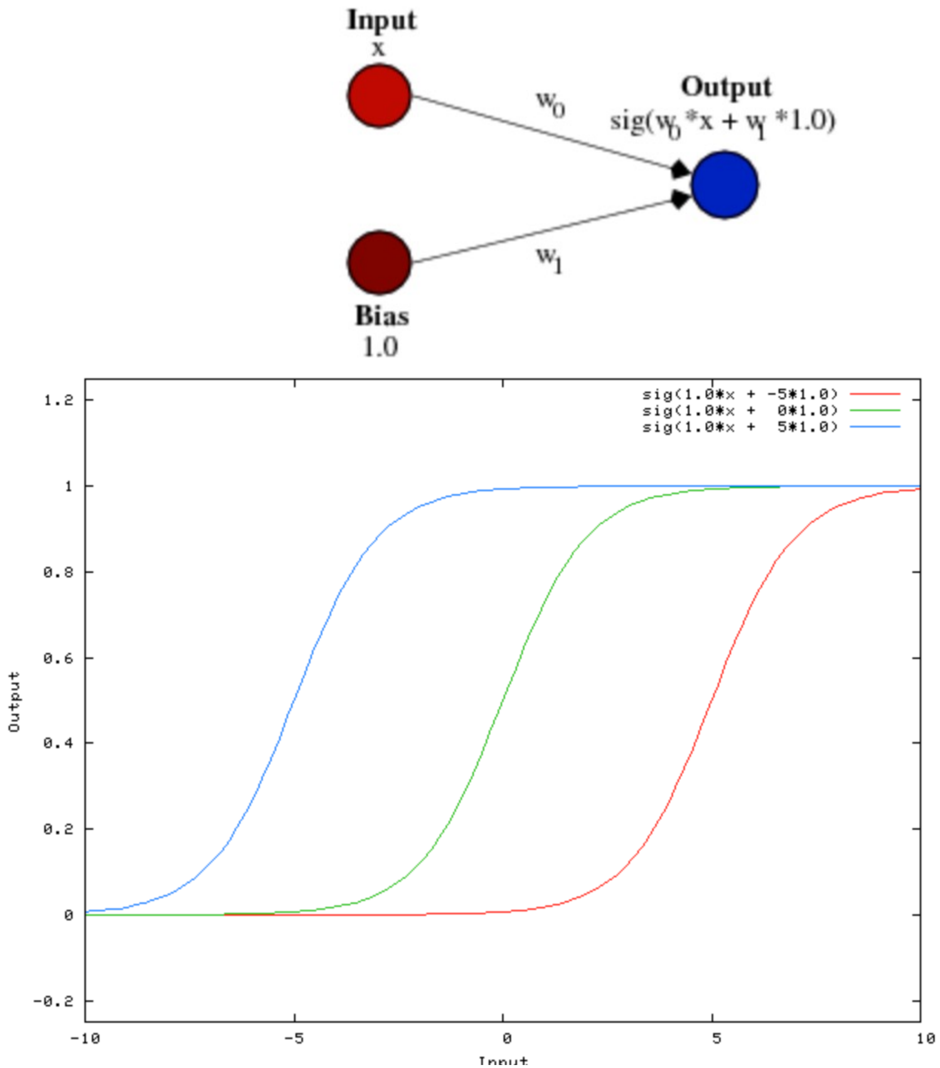


Weights

- Control the signal/strength of neuron fired (randomly initialized and fixed)
- Granting greater or lesser meaning to the input going inside the neuron ➔ to the output coming out
- The goal of neural network training algorithms is to determine the "best" possible set of weight values for the problem to resolve
- Weights are not being adjusted correctly when our model overfits

Bias





- Changing the weight w_0 essentially changes the "steepness" of the sigmoid
- BUT if you wanted output 0 when x is 2 – we need to shift the curve

Forward Pass/ Feedforward

$$x_j = \sum_i y_i w_{ji}$$

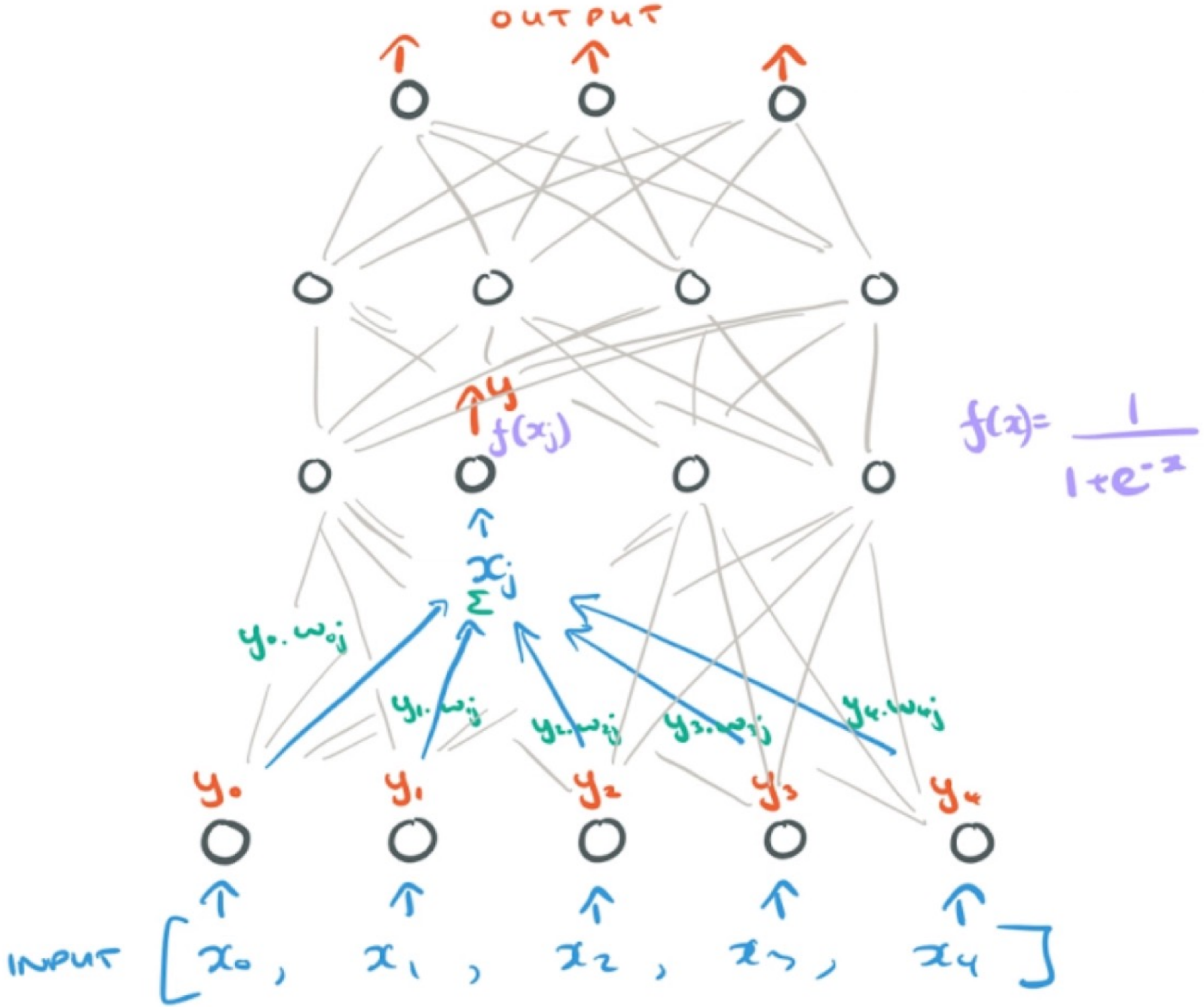
$$y_j = \frac{1}{1 + e^{-x_j}}$$

i,j: units (j is connected to i)

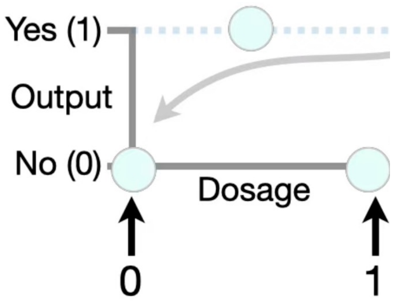
x_j: total input of unit j

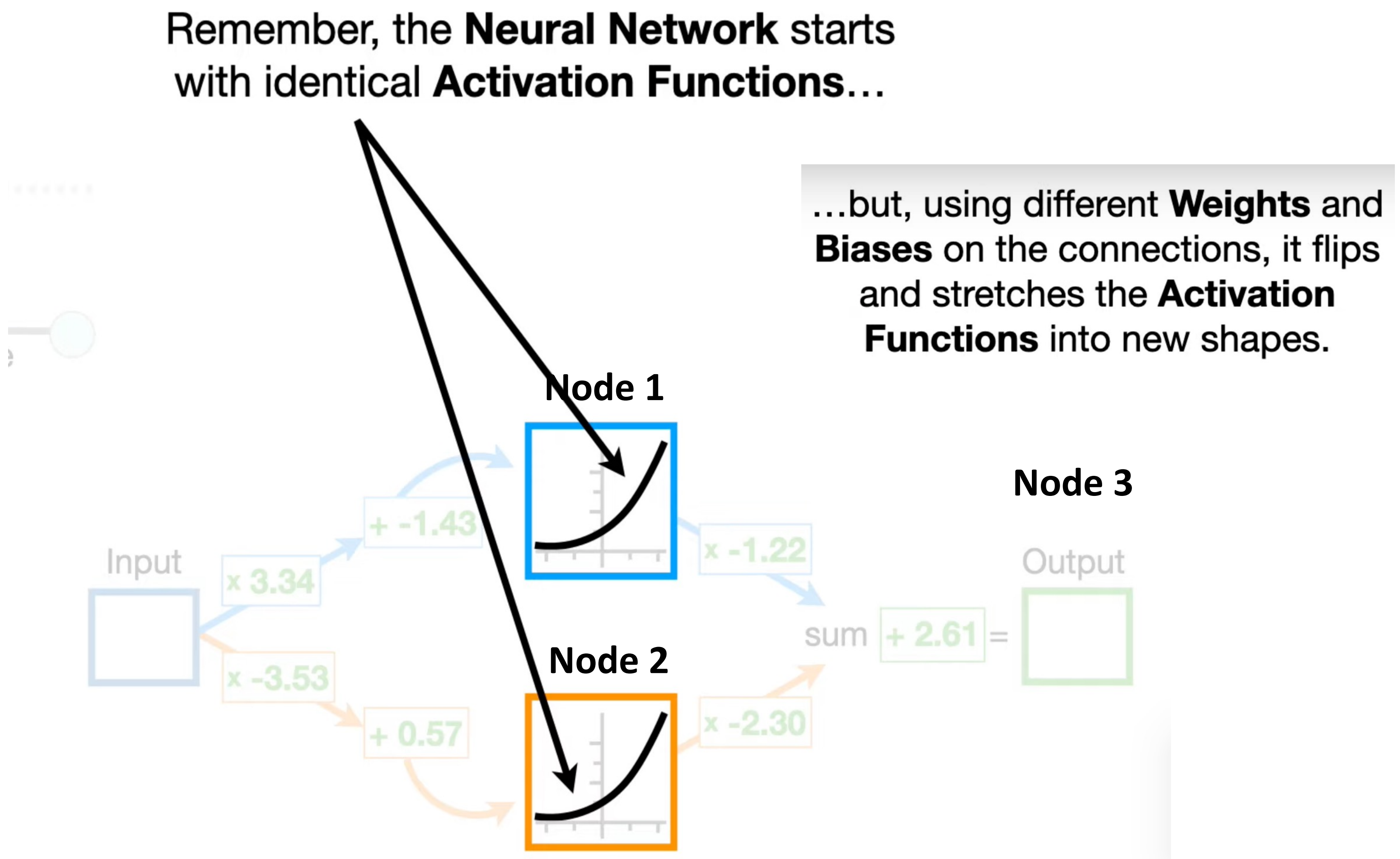
y_j: total output of unit j

w_{ji}: weight between two units

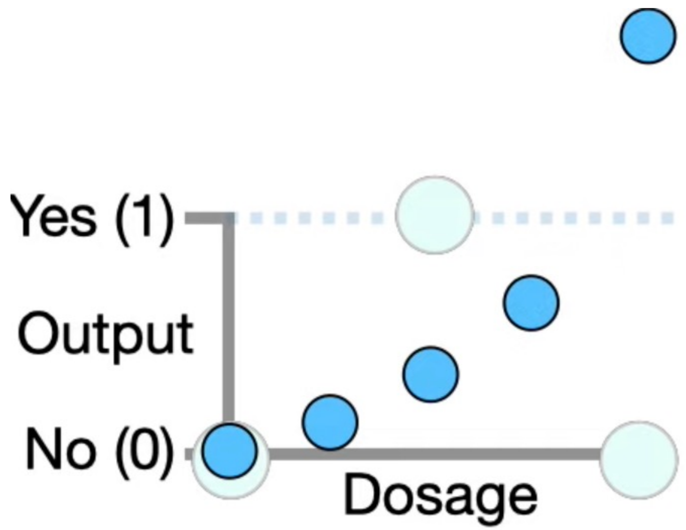


Forward Pass/ Feedforward

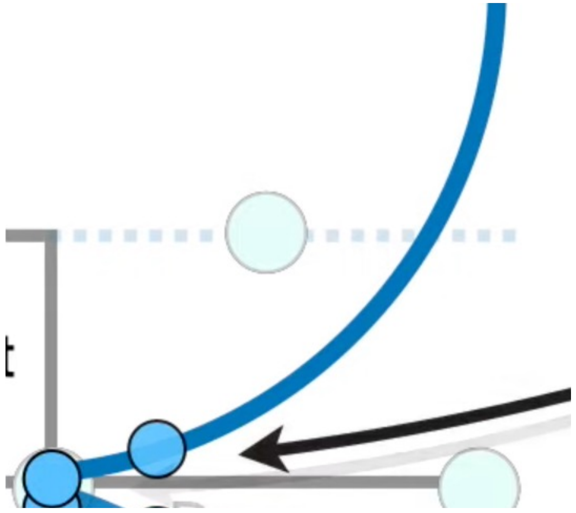




Forward Pass/ Feedforward

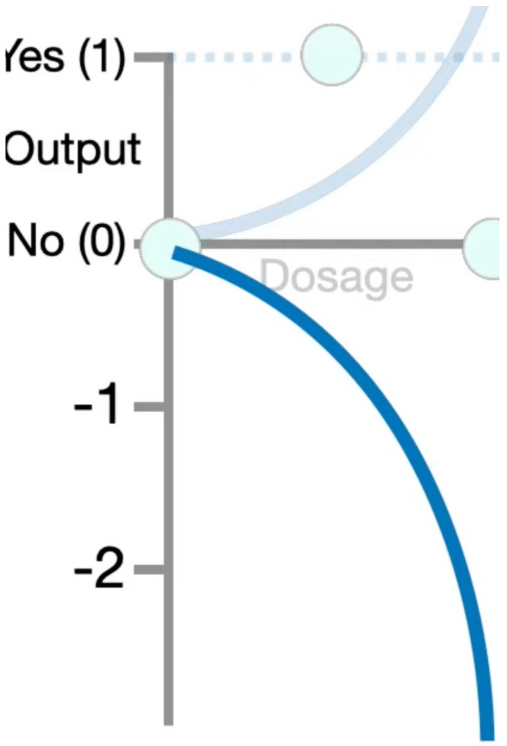


Use input x values and plot through the activation function



Get the values of y (output) of a hidden layer

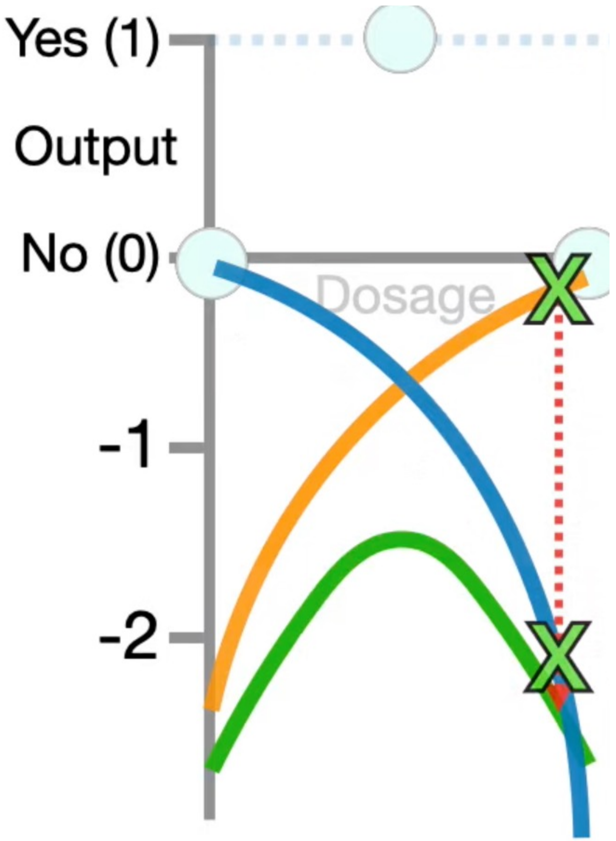
Multiply by the next weight corresponding to Node3 (-1.22)



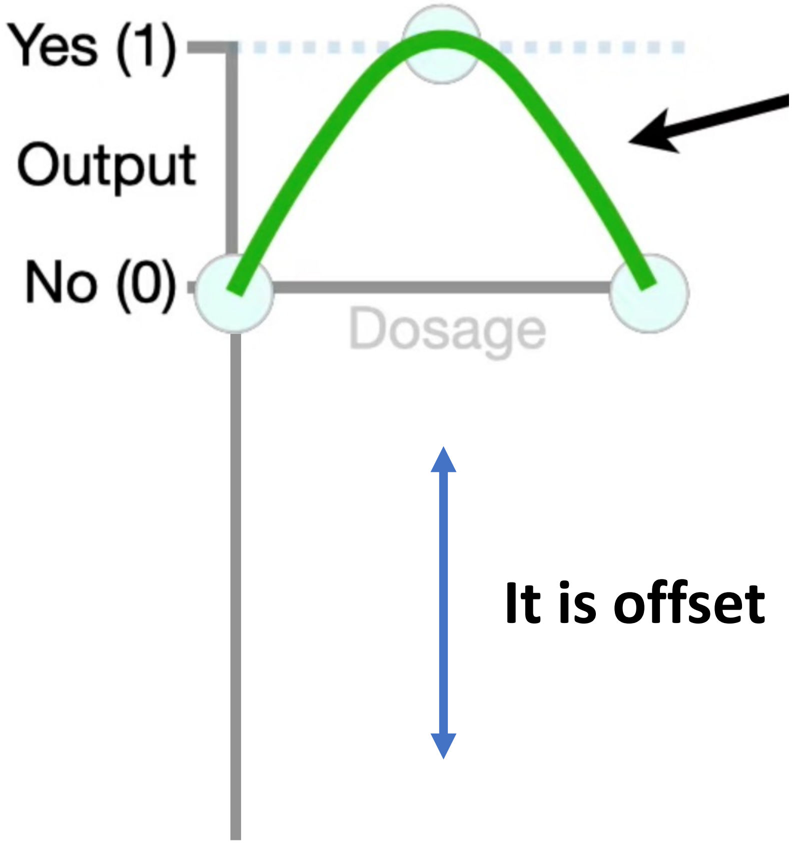
*Do the same for bottom Node 2

Forward Pass/ Feedforward

Add both Nodes and their weights + bias=0 to get Node3

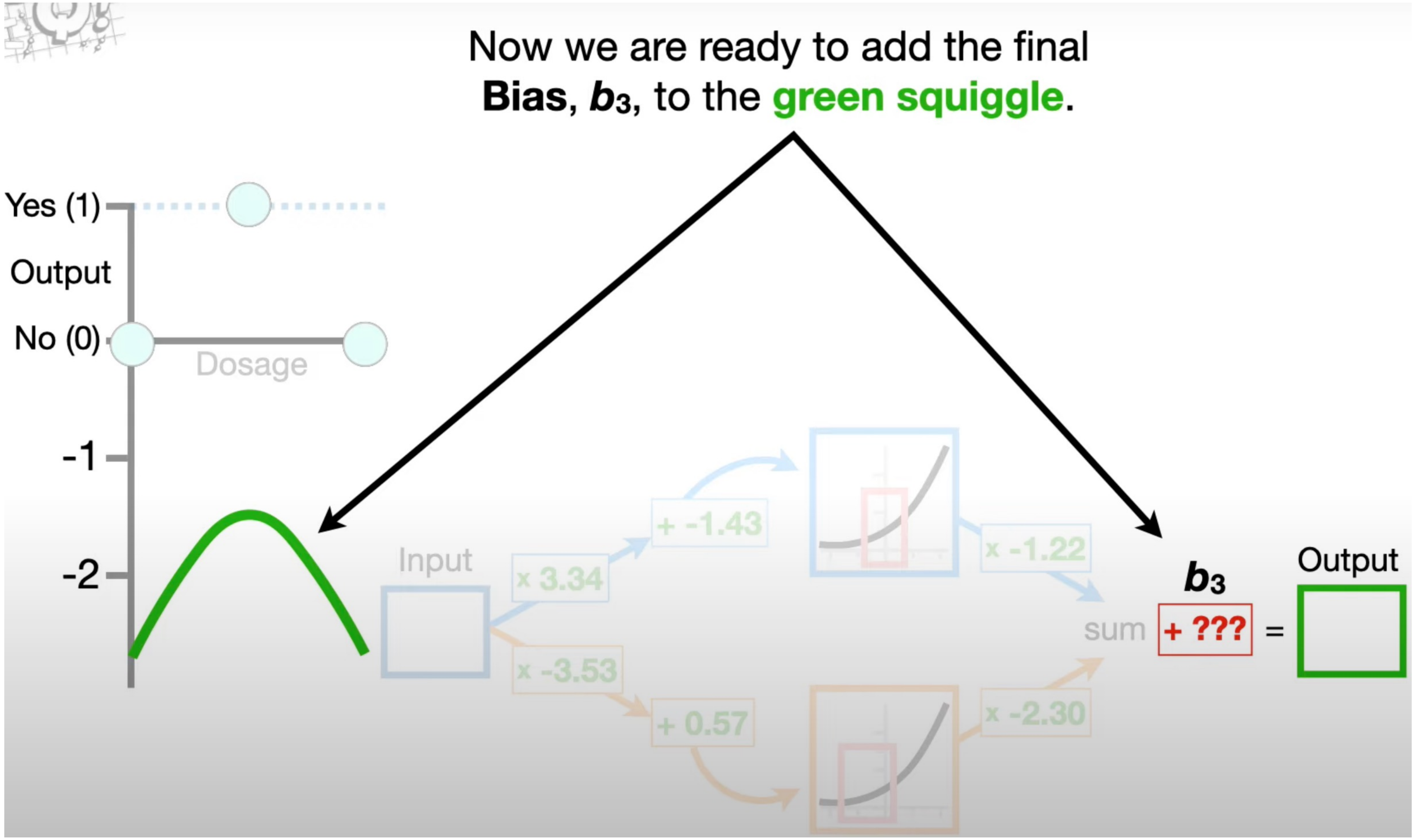


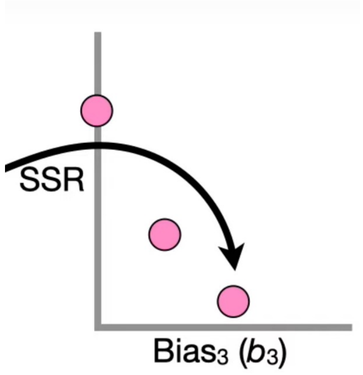
Predicted



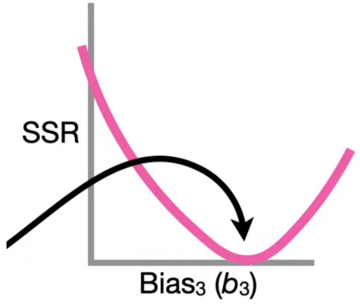
Actual

Backward Pass/ Backpropagation

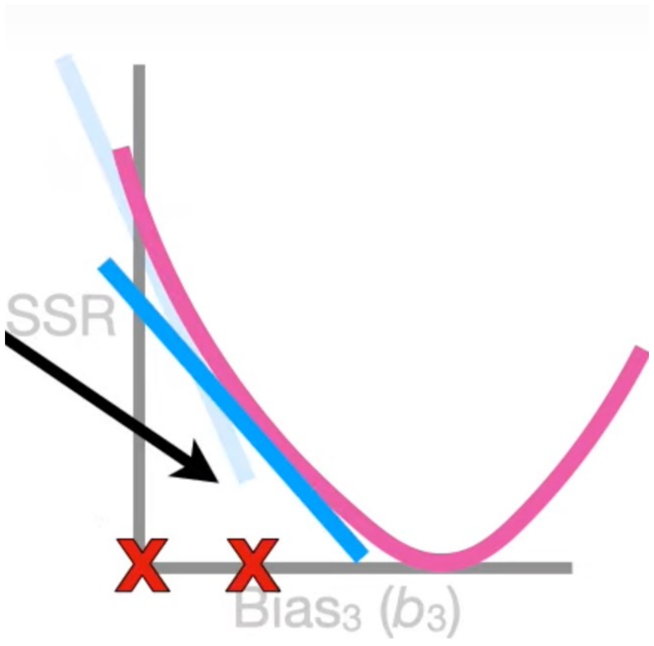




- In most cases the bias is set to 0 at the beginning of training
- The predicted curve is far from real point – the Sum of the Squared Residuals (SSR) is big
- Bias is added to shift the curve up and find the lowest SSR point = minimize the error
- The lowest point on the SSR curve is the bias used at Node3



Backward Pass/ Backpropagation



- **Instead of adding all the bias values to calculating the lowest point of the curve (minimum error)**
- **We use Gradient Descent to find it quickly**
- **That means we need to find the derivative of SSR (error/loss) with respect to Bias (b_3)**
- **Derivative = the direction of our slope**

$$\boxed{\begin{aligned}\text{SSR} = &(\text{Observed}_1 - \text{Predicted}_1)^2\\ &+ (\text{Observed}_2 - \text{Predicted}_2)^2\\ &+ (\text{Observed}_3 - \text{Predicted}_3)^2\end{aligned}} \rightarrow \text{SSR} = \sum_{i=1}^{\boxed{n=3}} (\text{Observed}_i - \text{Predicted}_i)^2$$

$$\text{SSR} = \sum_{i=1}^{n=3} (\text{Observed}_i - \boxed{\text{Predicted}_i})^2$$

$$\boxed{\text{Predicted}_i} = \text{green squiggle}_i = \text{blue} + \text{orange} + b_3$$

$$\frac{d\ \text{SSR}}{d\ b_3} = \frac{d\ \text{SSR}}{d\ \text{Predicted}} \times \frac{d\ \text{Predicted}}{d\ b_3}$$ ***(Chain rule)***

$$\frac{d\ \text{SSR}}{d\ b_3} = \sum_{i=1}^{n=3} -2 \times (\text{Observed}_i - \text{Predicted}_i) \times 1$$

Step Size = $\boxed{\text{Slope}}$ × Learning Rate

Backward Pass/ Backpropagation

$$\frac{d\,SSR}{d\,b_3} = \frac{d\,SSR}{d\,Predicted} \times \frac{d\,Predicted}{d\,b_3}$$ *(Chain rule)*

$$\frac{d\,SSR}{d\,b_3} = \sum_{i=1}^{n=3} -2 \times (\text{Observed}_i - \text{Predicted}_i) \times 1$$

-20

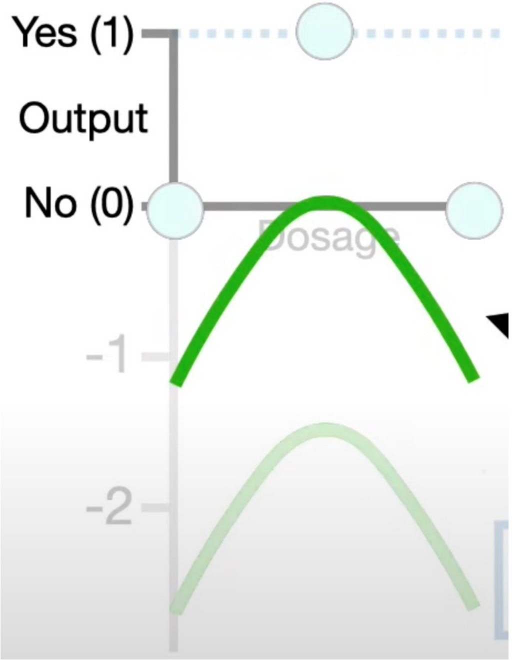
$$\text{Step Size} = \boxed{\text{Slope}} \times \text{Learning Rate}$$

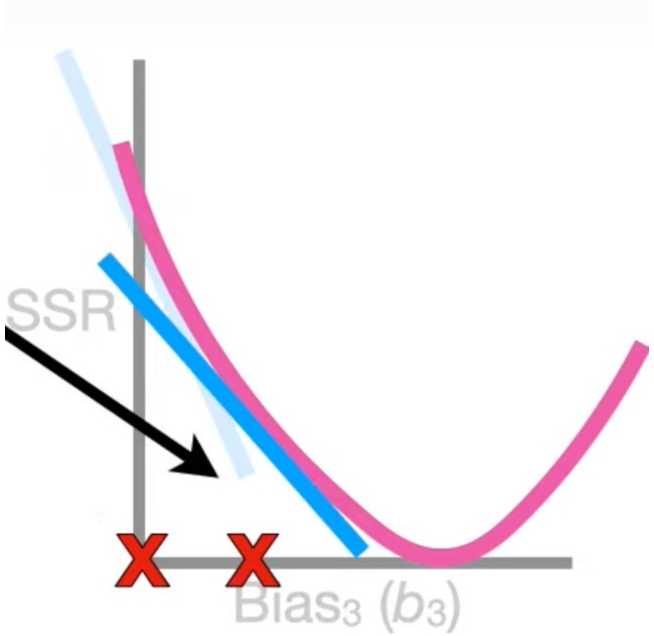
LR = 0.1

Step Size = -2

New bias:

$$\text{New } b_3 = \text{Old } b_3 - \text{Step Size}$$ **= 0 + 2 = 2**

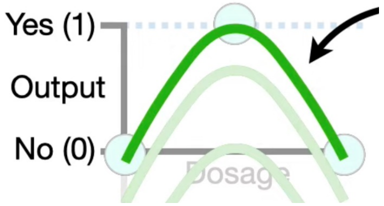


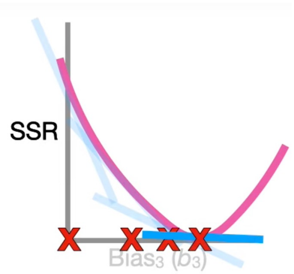


The predicted curve is now shifted up – lower error

Get the new gradient/direction of the slope:

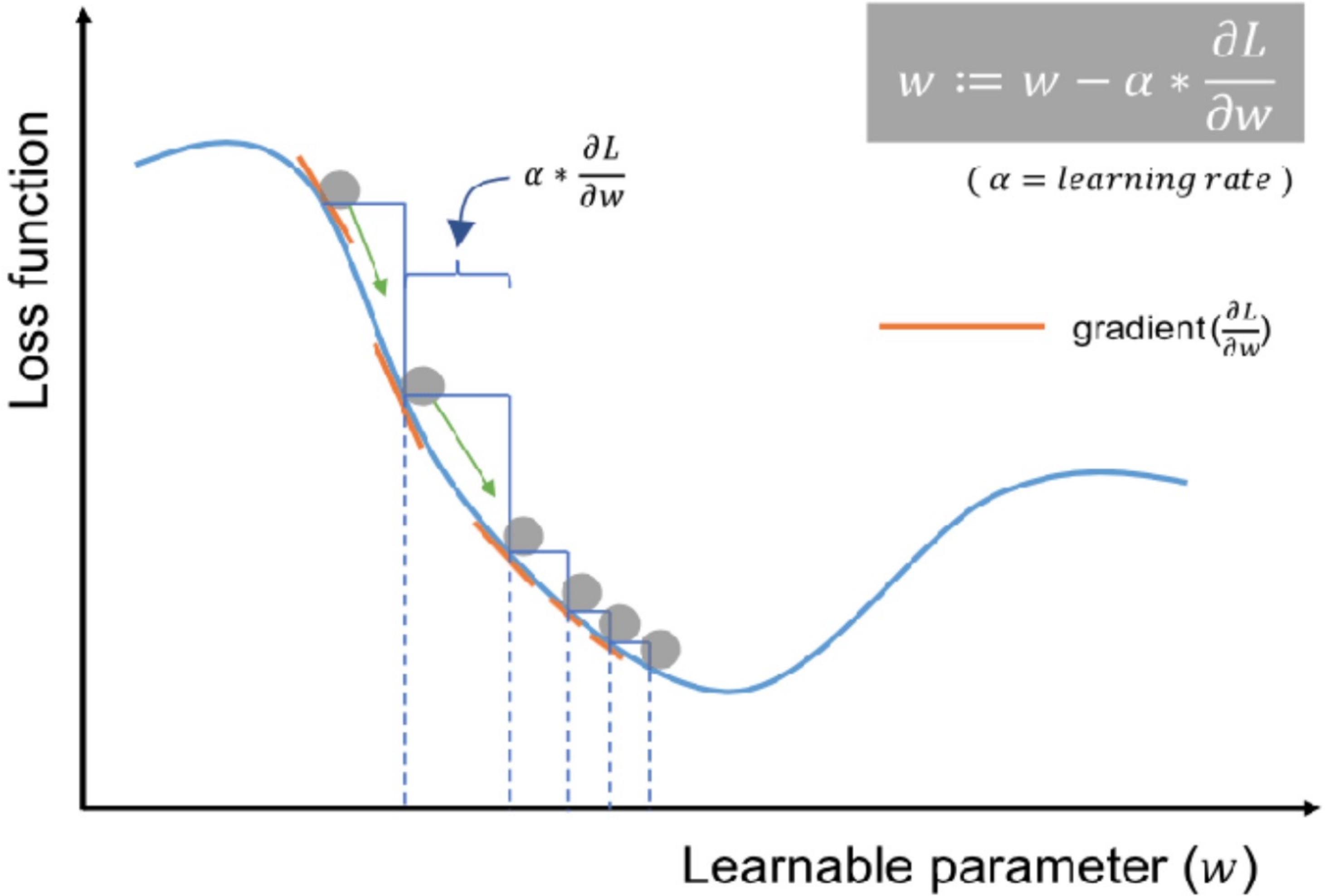
- *Plug in x values into the d SRR/ s b_3 = etc keep plugging until reached minimum loss*





Backward Pass/ Backpropagation

- Taking the ***first derivative*** of any function gives ***direction (slope)*** to which we move our position ***to achieve minimum function value (minimum loss).***



Activation Functions