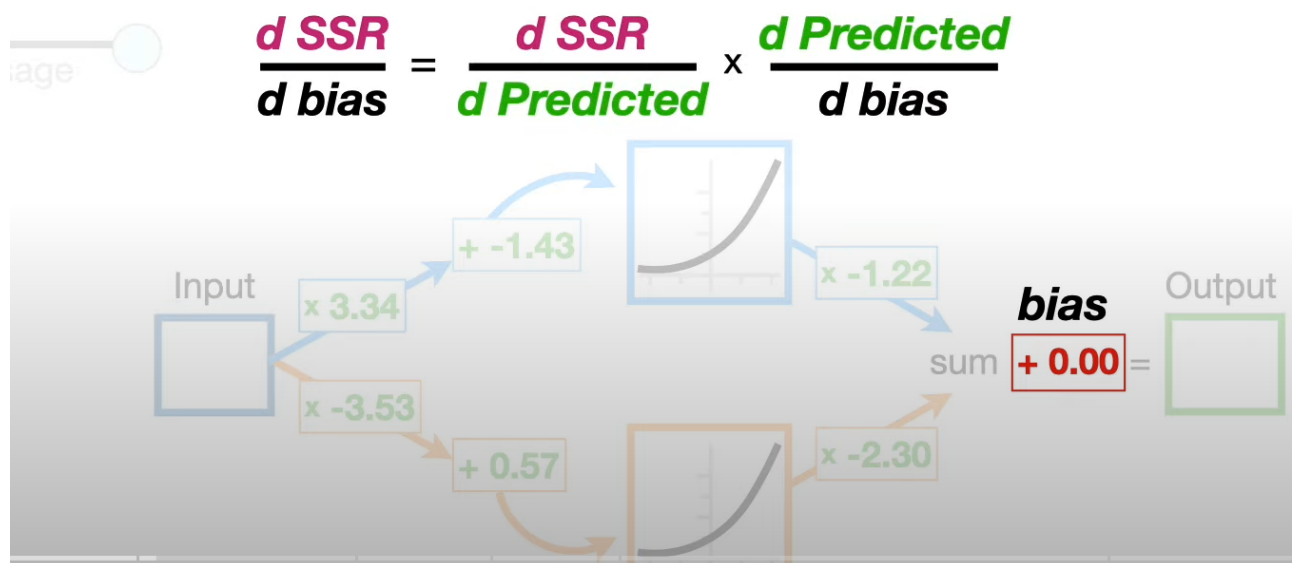


Backpropagation Main Ideas

- It optimizes the **Weights** and **Biases** in Neural Networks.

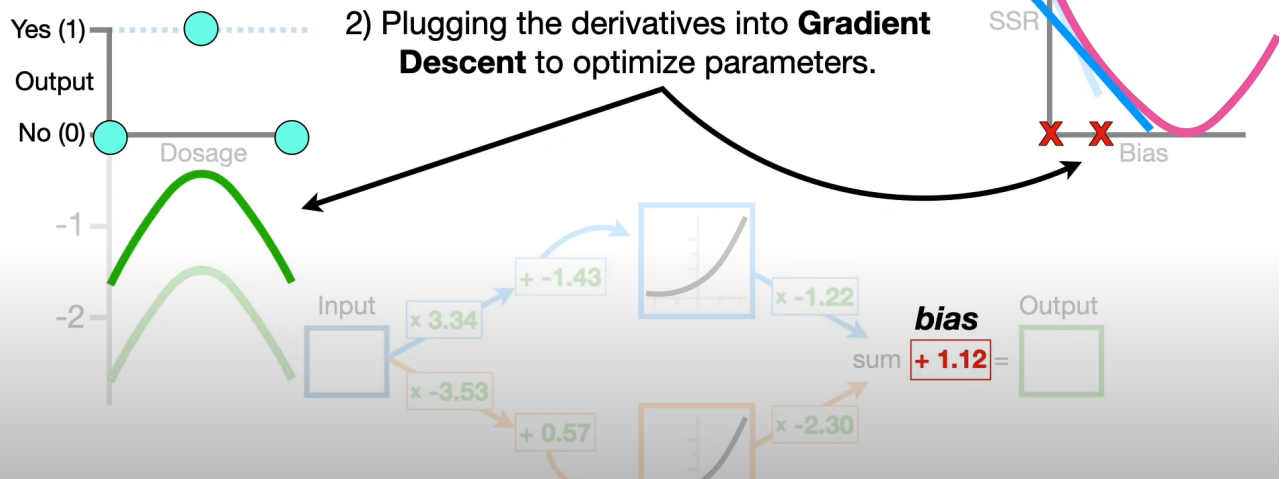
In this part, we talk about the **Main Ideas of Backpropagation:**

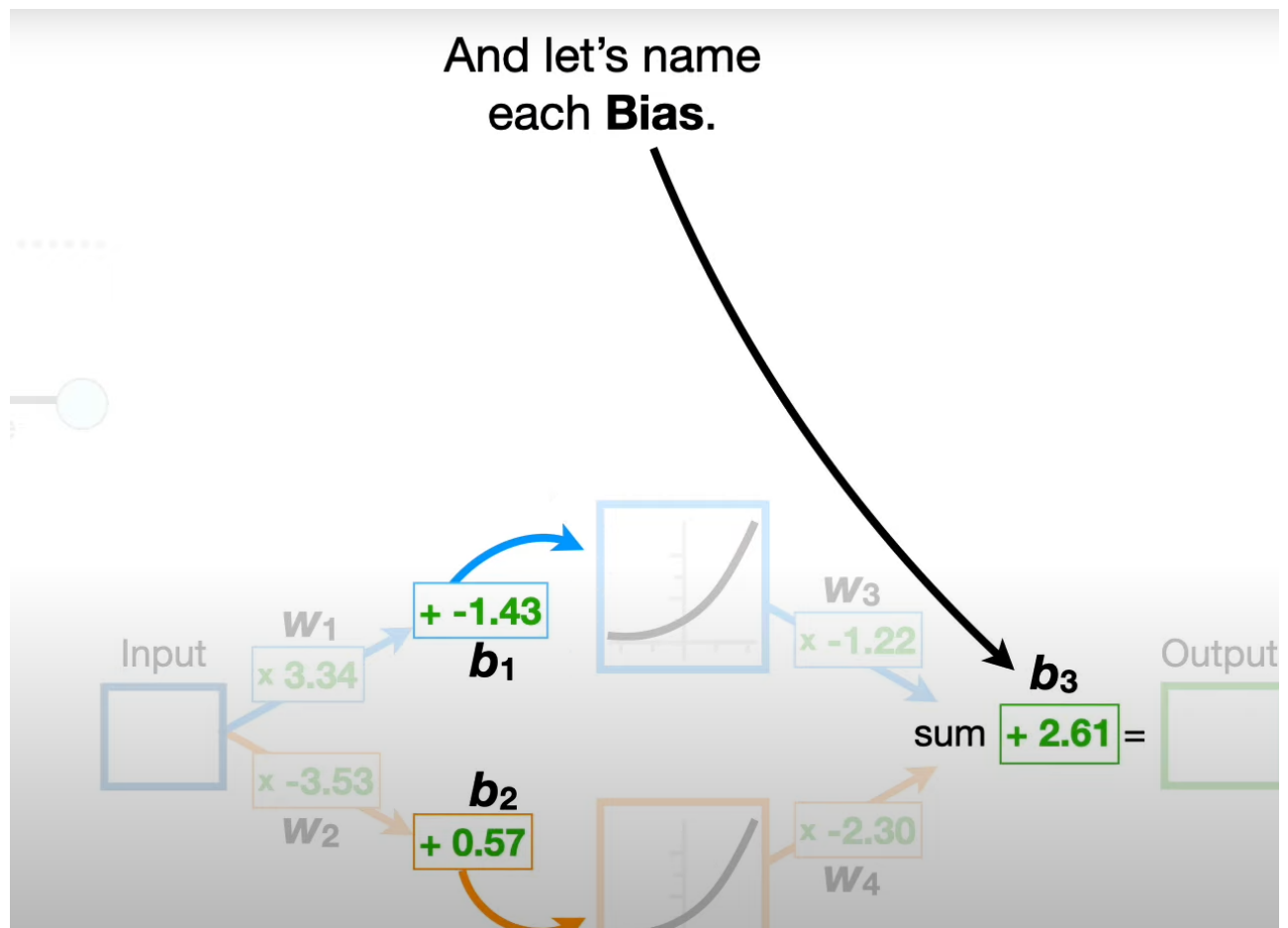
1) Using the **Chain Rule** to calculate derivatives...



In this part, we talk about the **Main Ideas of Backpropagation:**

2) Plugging the derivatives into **Gradient Descent** to optimize parameters.

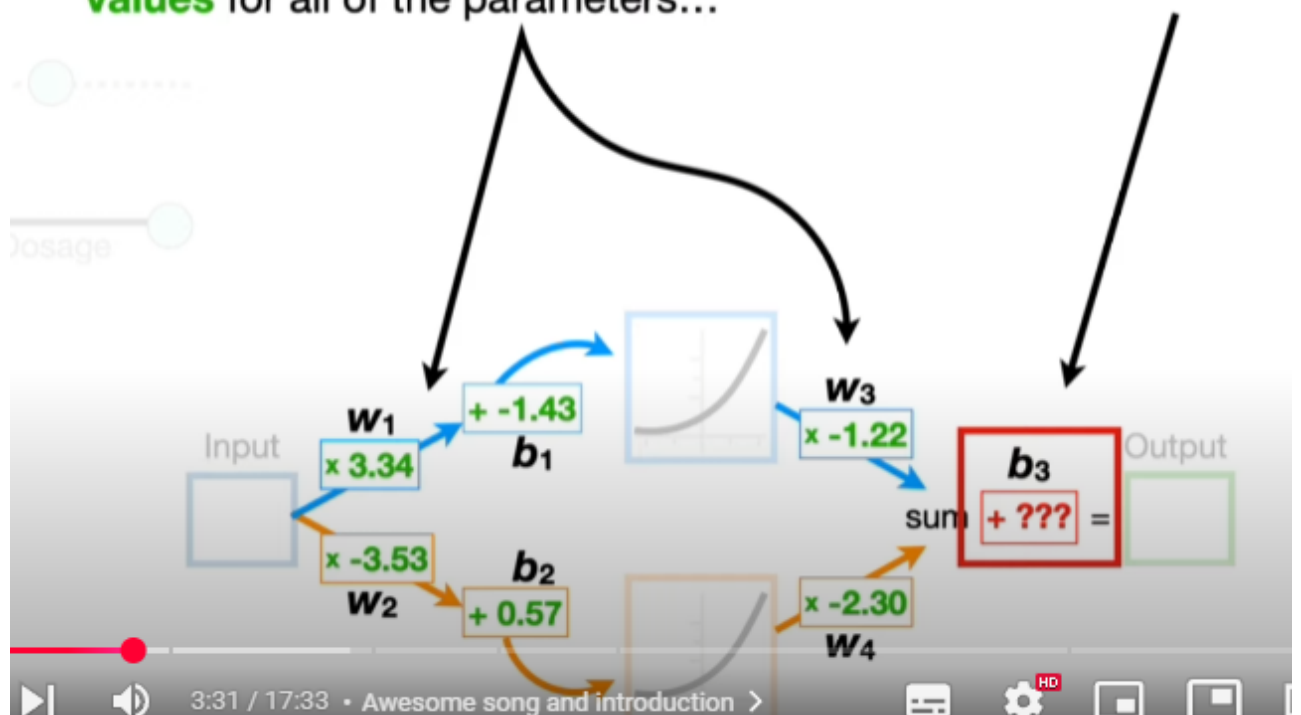




- Let's name weights (w_1 , w_2 , w_3 and w_4) and biases (b_1 , b_2 and b_3).
- Backpropagation works with the last parameter and works its way backwards to estimate all of the other parameters.
- However, we can discuss main idea by estimating last parameter, bias b_3 .

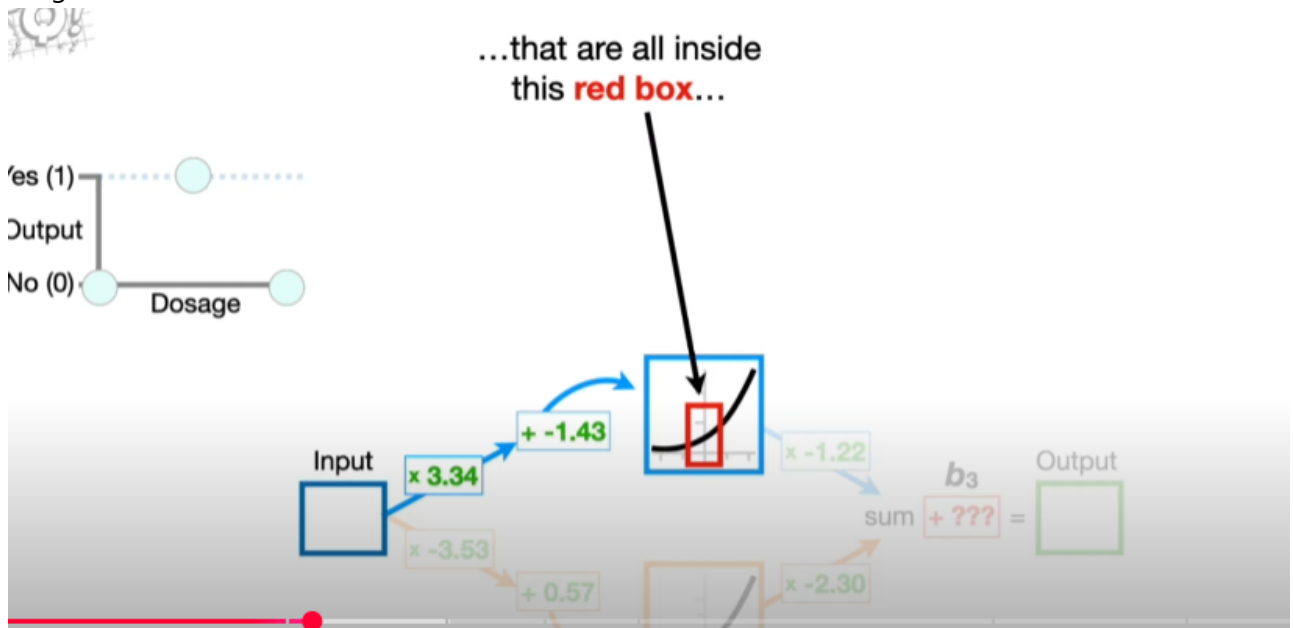
So, in order to start from the back, let's assume that we already have **optimal values** for all of the parameters...

...except for the last **Bias term, b_3** .

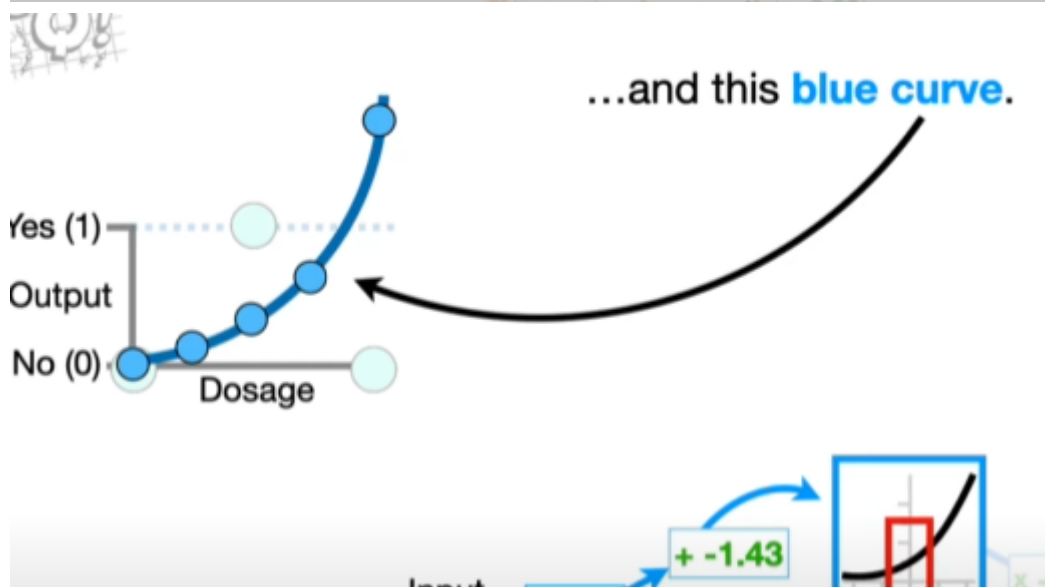
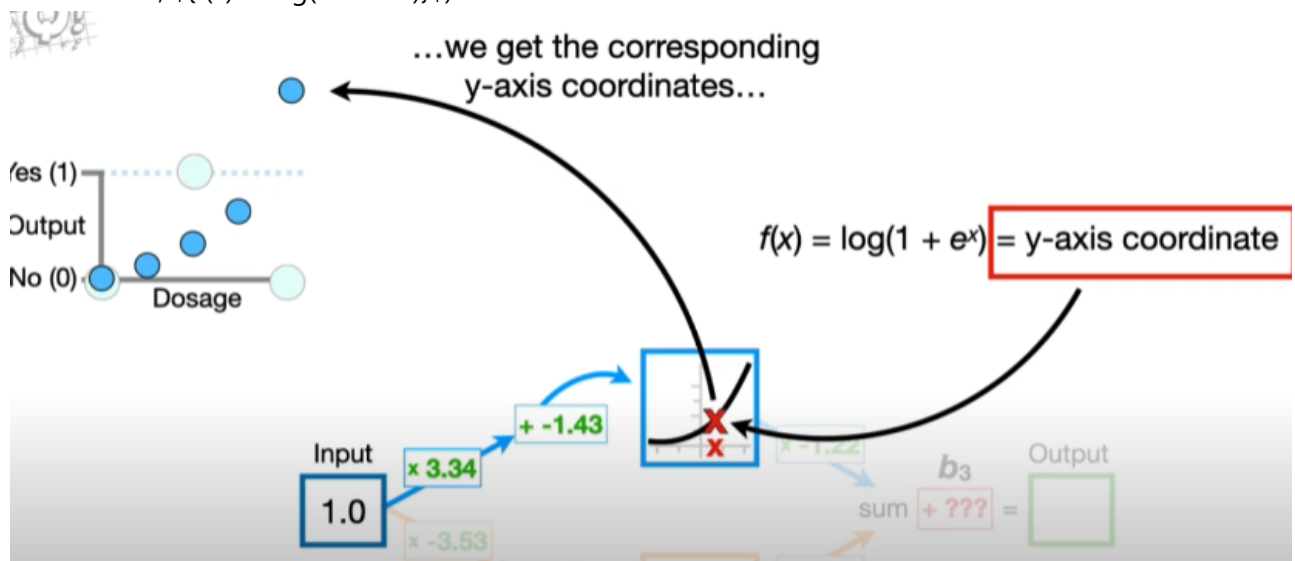


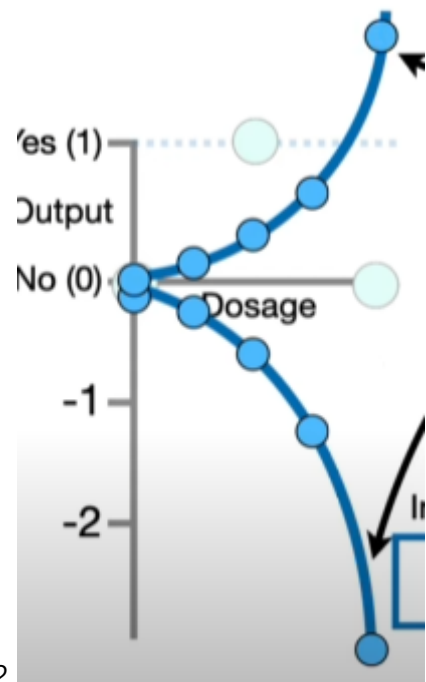
Note : **Unoptimized Doses** and **Optimized Doses**

- Now if we run Dosages from 0 to 1 through the connection to the top Node in the Hidden Layer, then we get the x-axis coordinates for the Activation Function

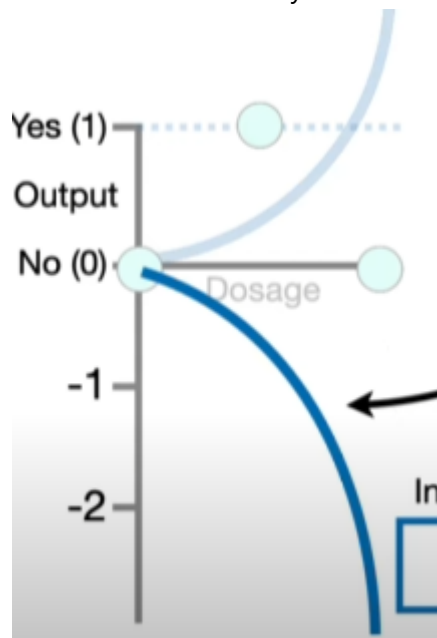


and when we plug the x-axis coordinates into the Activation Function (in this case is Softplus Activation Function i.e., $f(x) = \log(1 + e^x)$)



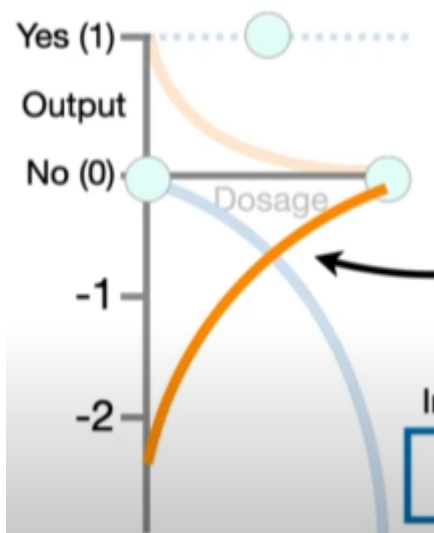


- Then we multiply the y-axis coordinates on the blue curve by -1.22

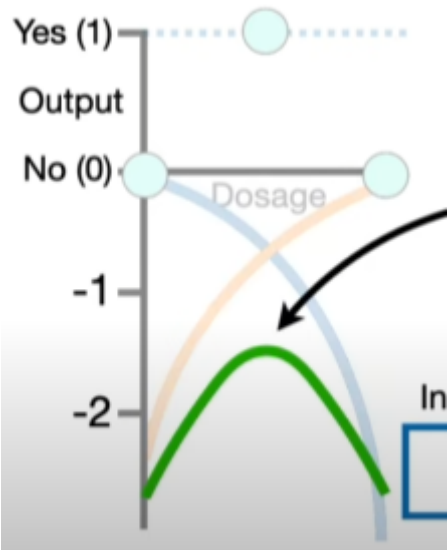


and then we get the final blue curve.

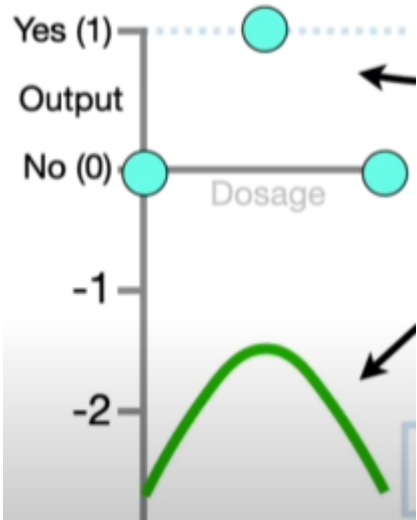
- Now if we run Dosages from 0 to 1 through the connection to the bottom Node in the Hidden Layer, ...



and then we get the final orange curve.



- Adding those blue and orange curves will get us a green squiggle
- Since we don't have optimal value of b_3 , we gotta give it some initial value like 0 .
- 0 means green squiggle doesn't move and it is far from observed values.

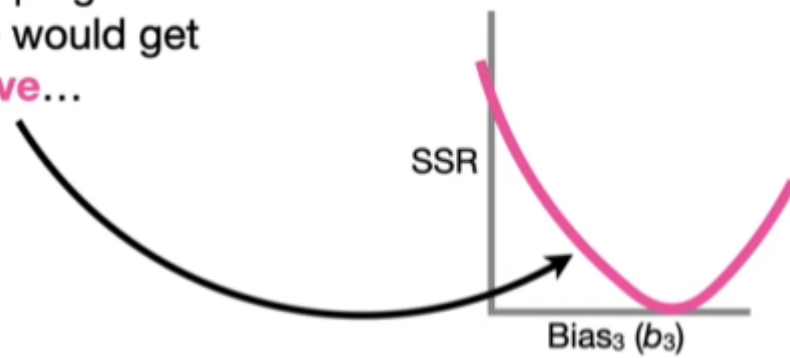


- It can be quantified how good it fits the data by calculating $\sum_{i=1}^n ((\text{Observed}_i - \text{Predicted}_i)^2)$.

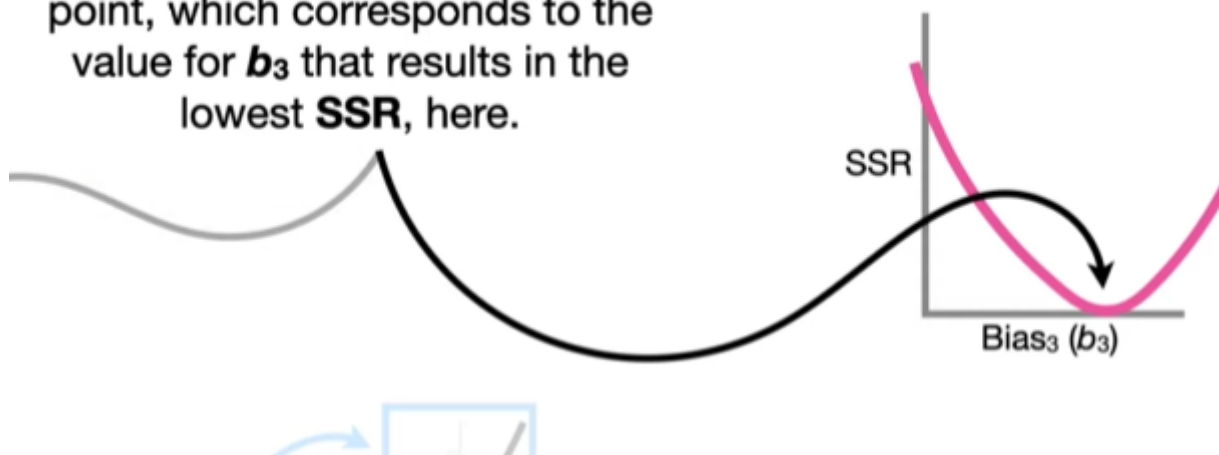
Note : Changing Bias means Squiggle translates

Bias	Sum of Squared Residuals
0	20.4
1	7.8
2	1.11
3	0.46

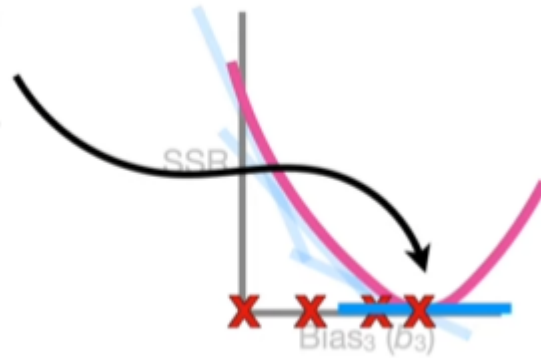
And if we had time to plug in tons of values for b_3 , we would get this **pink curve**...



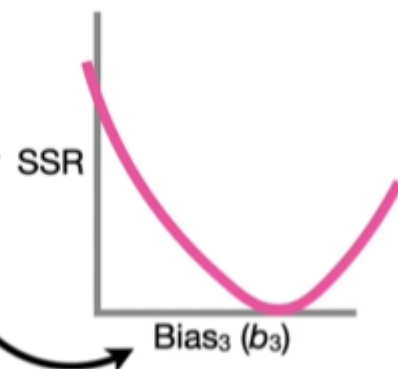
...and we could find the lowest point, which corresponds to the value for b_3 that results in the lowest **SSR**, here.



However, instead of plugging in tons of values to find the lowest point in the **pink curve**, we use **Gradient Descent** to find it relatively quickly.



And that means we need to find the derivative of the **Sum of the Squared Residuals** with respect to b_3 .



- Now, since $\text{Sum of Squared Residuals} = \sum_{i=1}^n ((\text{Observed}_i - \text{Predicted}_i)^2)$, and each Predicted_i value comes from green squiggle_i , which comes from last part of neural network.
- Therefore, $\text{green squiggle}_i = \text{blue} + \text{orange} + b_3$.
- For Gradient Descent to optimize b_3 , we need to take SSR's derivative w.r.t b_3 .

$$\frac{d, \text{SSR}}{d, b_3}$$

- Therefore by **The Chain Rule**,
- $\frac{d, \text{SSR}}{d, b_3} = \frac{d, \text{SSR}}{d, \text{Predicted}} \times \frac{d, \text{Predicted}}{d, b_3}$

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

$$2. \frac{d}{d b_3} \text{Predicted} = 1$$

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

Note : Let Learning Rate be 0.1

Bias	$\frac{d}{d b_3} \text{SSR}$	Step Size = $\frac{d}{d b_3} \text{SSR} \times \text{Learning Rate}$	$\text{New } b_3 = \text{Old } b_3 - \text{Step Size}$
0	-15.7	$-15.7 \times 0.1 = -1.57$	1.57
1.57	-6.26	$-6.26 \times 0.1 = -0.626$	2.19

Do it till Step Size close to 0.

- Most optimized $b_3 = 2.61$