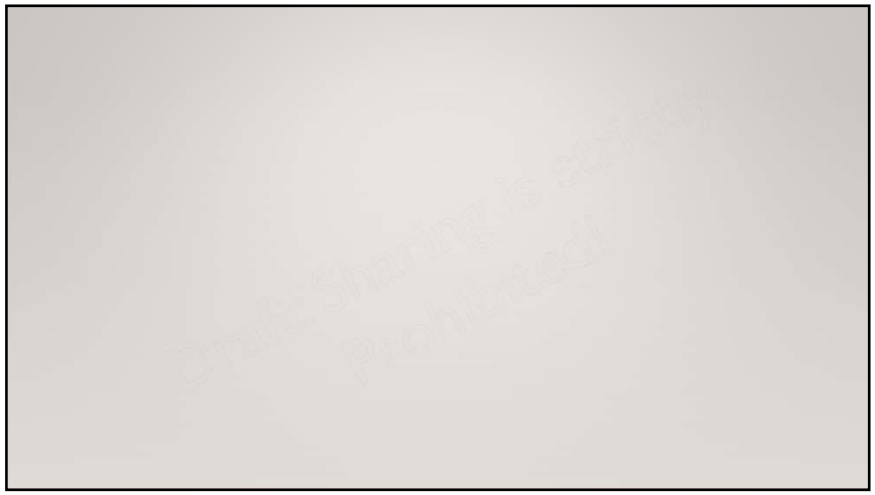
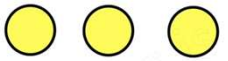
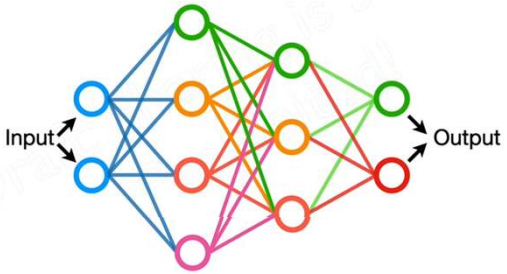


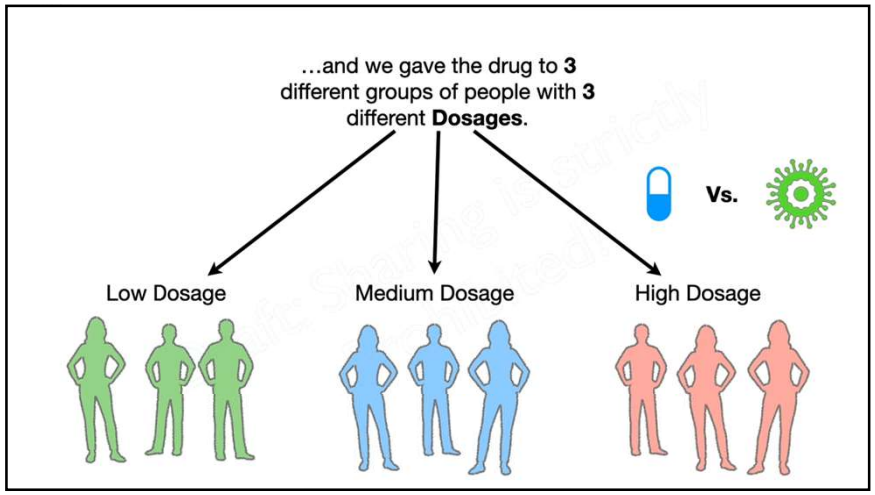


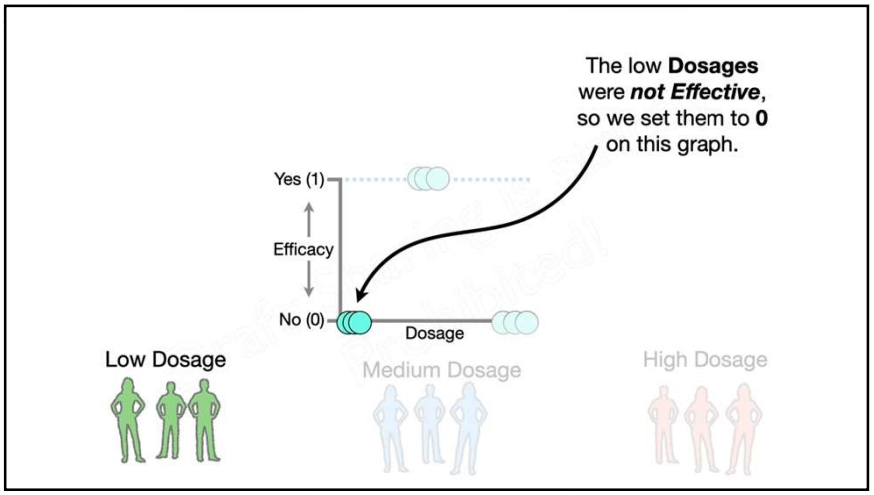
Neural Networks…

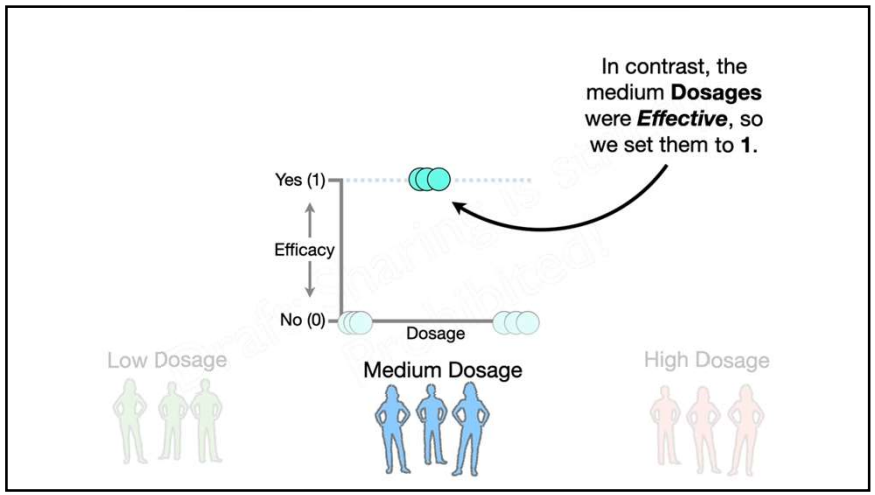
Neural Networks, one of the most popular algorithms in **Machine Learning**, cover a broad range of concepts and techniques.

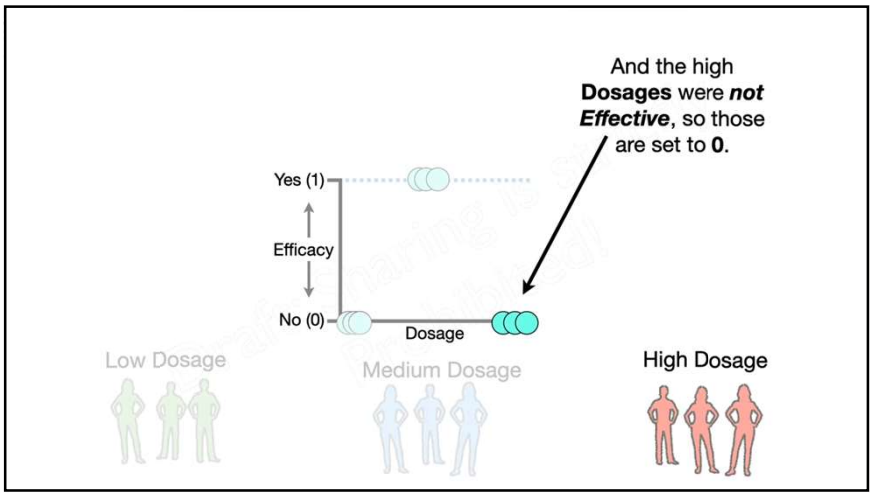


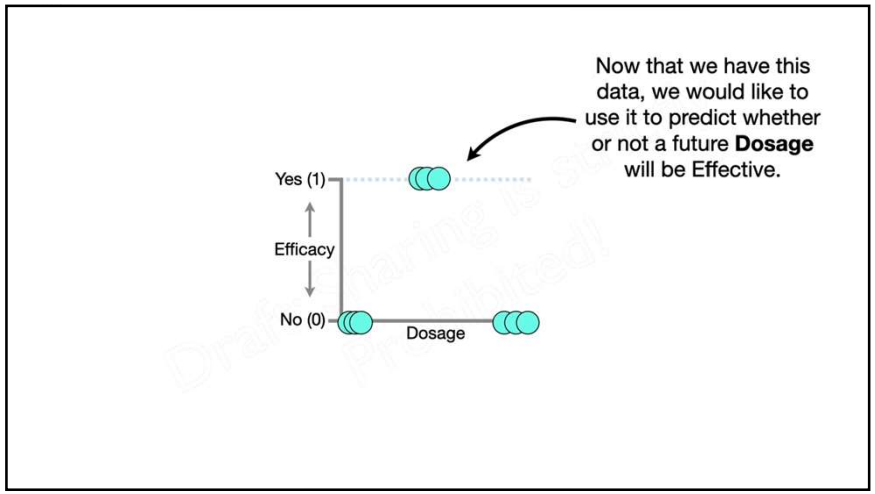
So, with that said, let's imagine we tested a drug that was designed to treat an illness…

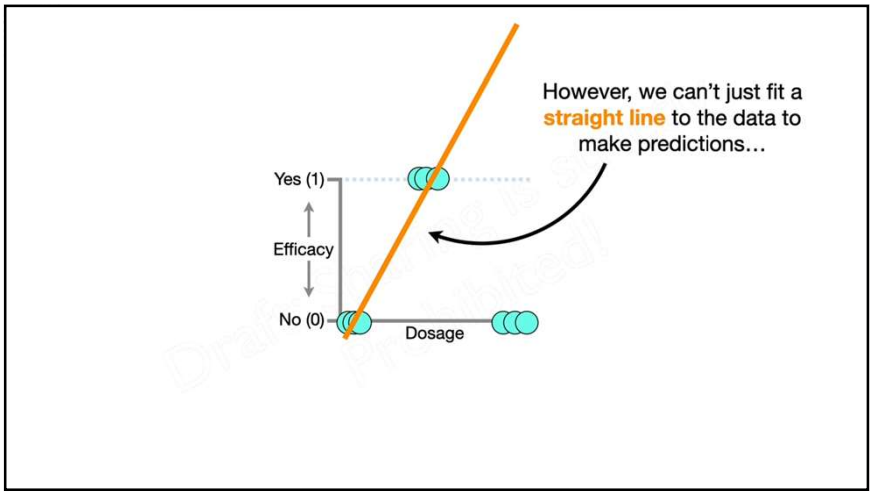

…and we gave the drug to **3** different groups of people with **3** different **Dosages**.
Vs.
Low Dosage
Medium Dosage
High Dosage

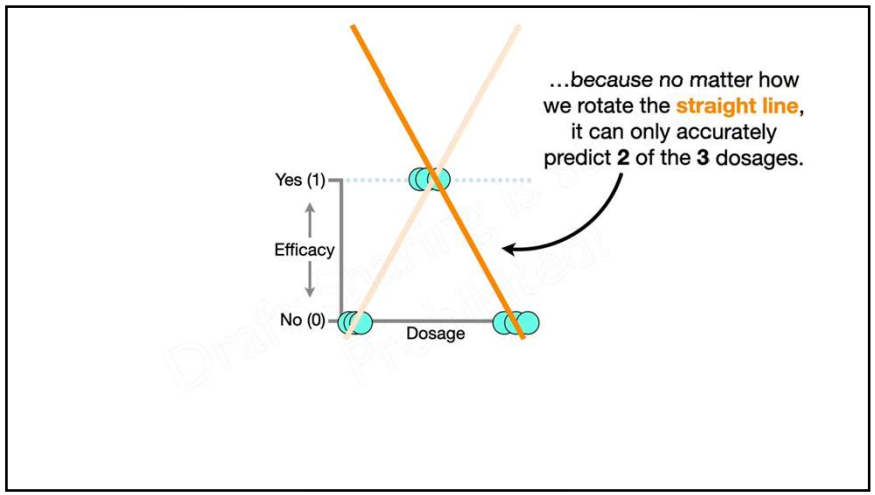

The low **Dosages** were ***not Effective***, so we set them to **0** on this graph.
Yes (1)
Efficacy
No (0)
Dosage
Low Dosage
Medium Dosage
High Dosage

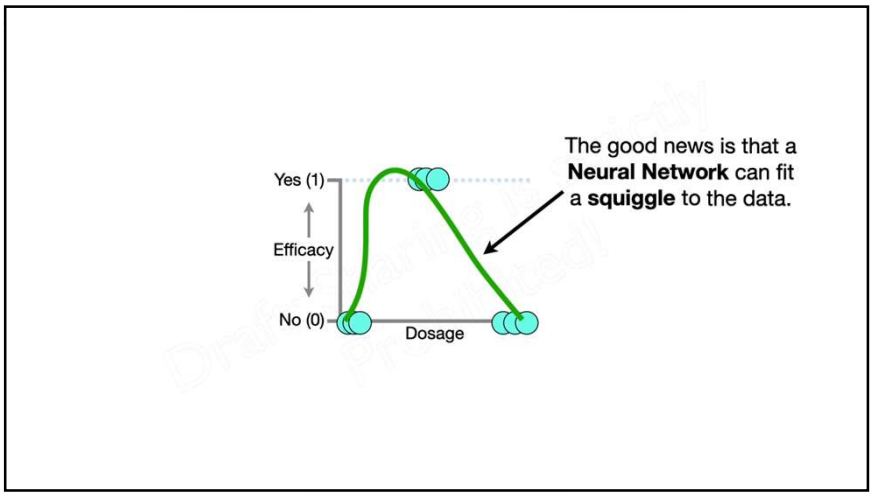

In contrast, the medium **Dosages** were ***Effective***, so we set them to **1**.
Yes (1)
Efficacy
No (0)
Dosage
Low Dosage
Medium Dosage
High Dosage

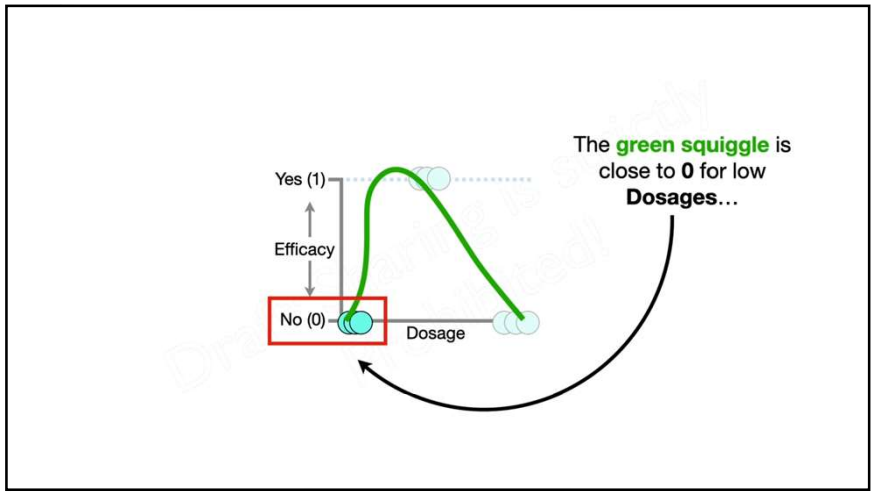

And the high **Dosages** were ***not Effective***, so those are set to **0**.
Yes (1)
Efficacy
No (0)
Dosage
Low Dosage
Medium Dosage
High Dosage

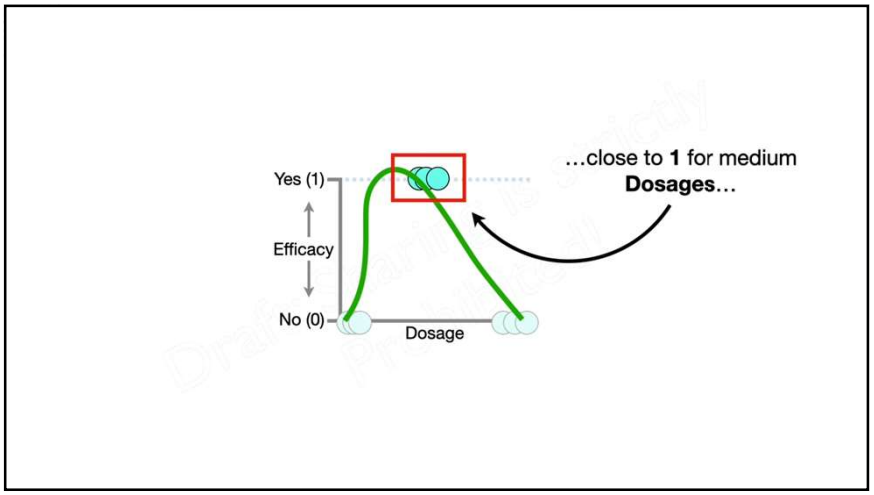

Now that we have this data, we would like to use it to predict whether or not a future **Dosage** will be Effective.
Yes (1)
Efficacy
No (0)
Dosage

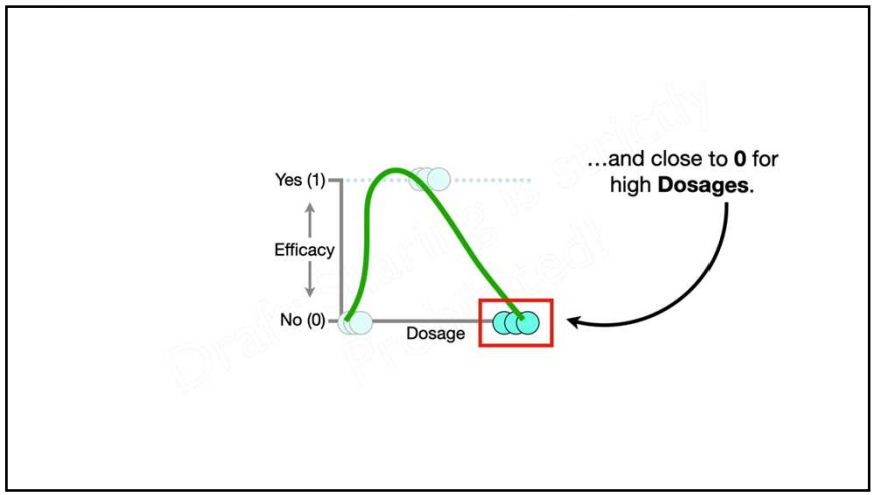

However, we can't just fit a straight line to the data to make predictions…
Yes (1)
Efficacy
No (0)
Dosage

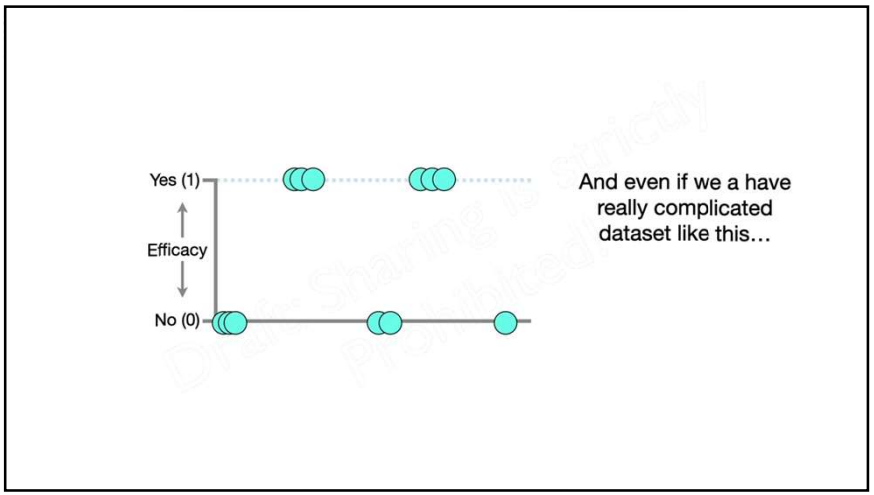

…*because no* matter how we rotate the straight line, it can only accurately predict **2** of the **3** dosages.
Yes (1)
Efficacy
No (0)
Dosage

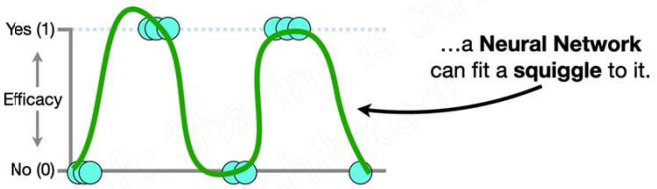

The good news is that a **Neural Network** can fit a **squiggle** to the data.
Yes (1)
Efficacy
No (0)
Dosage


Yes (1)
Efficacy
No (0)
Dosage
The green squiggle is close to 0 for low Dosages…


Yes (1)
Efficacy
No (0)
Dosage
…close to 1 for medium Dosages…

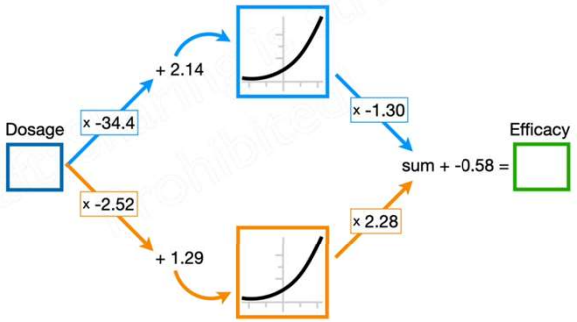

Yes (1)
Efficacy
No (0)
Dosage
…and close to 0 for high Dosages.


Yes (1)
Efficacy
No (0)
And even if we a have really complicated dataset like this…

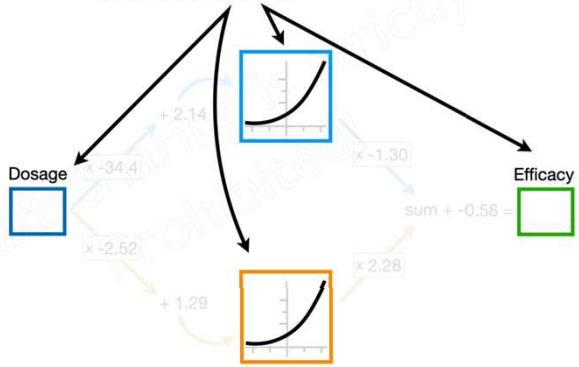


…a **Neural Network** can fit a **squiggle** to it.

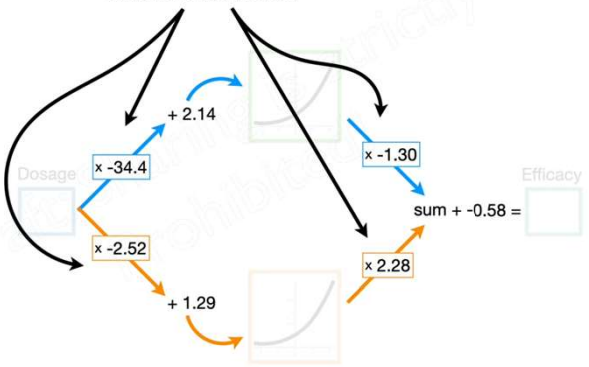
But first let's just talk about what a **Neural Network** is.

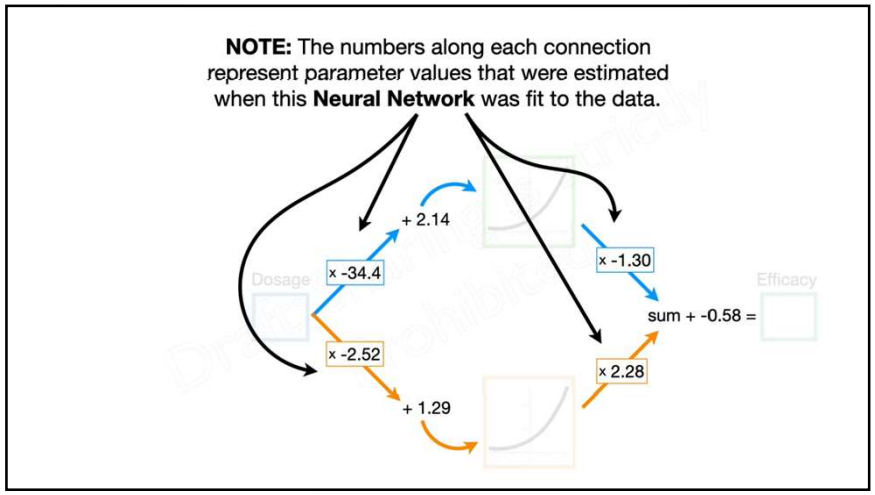


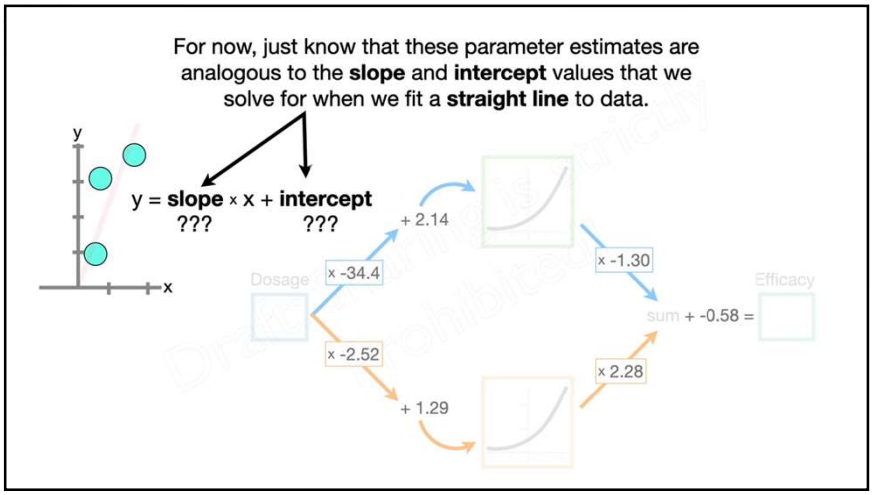
A **Neural Network** consists of **Nodes**…

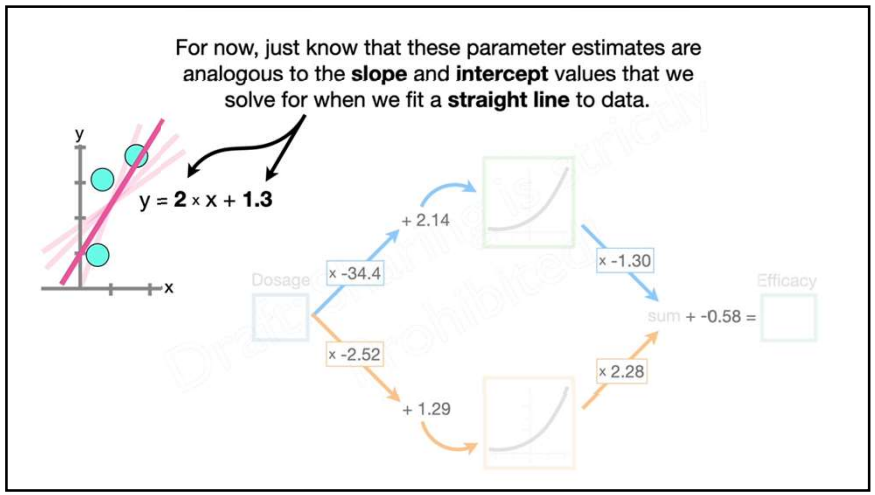


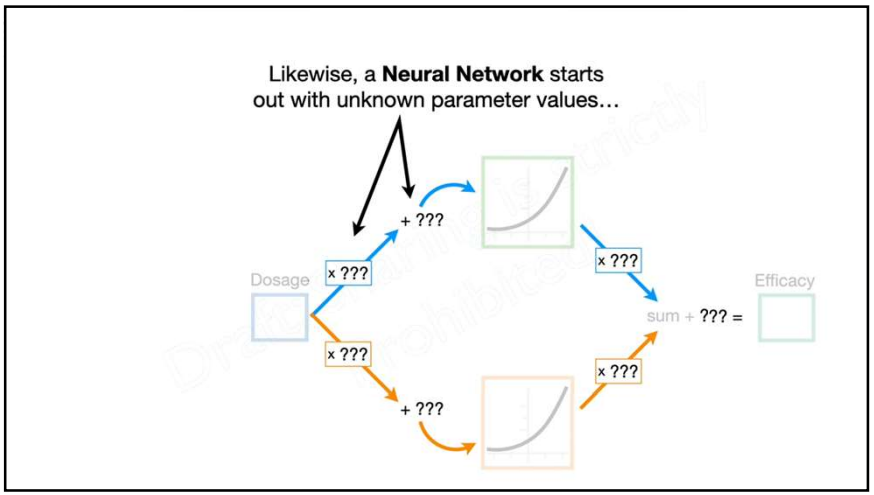
…and **connections** between the nodes.

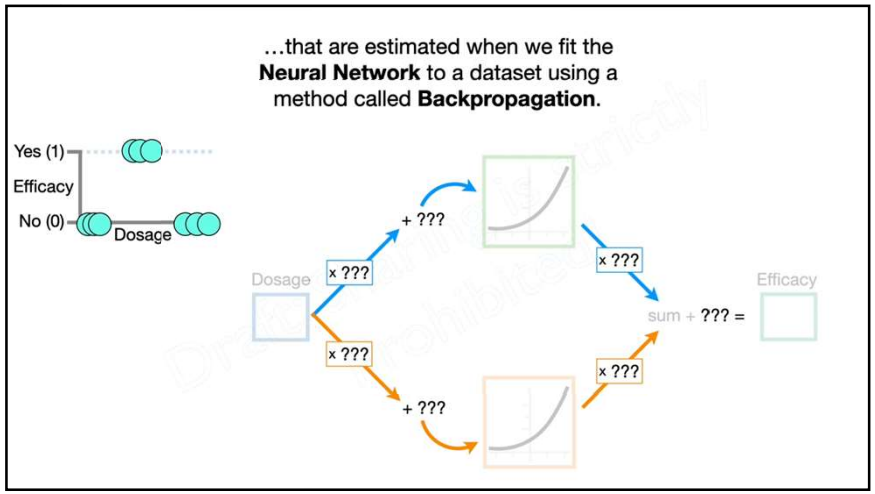


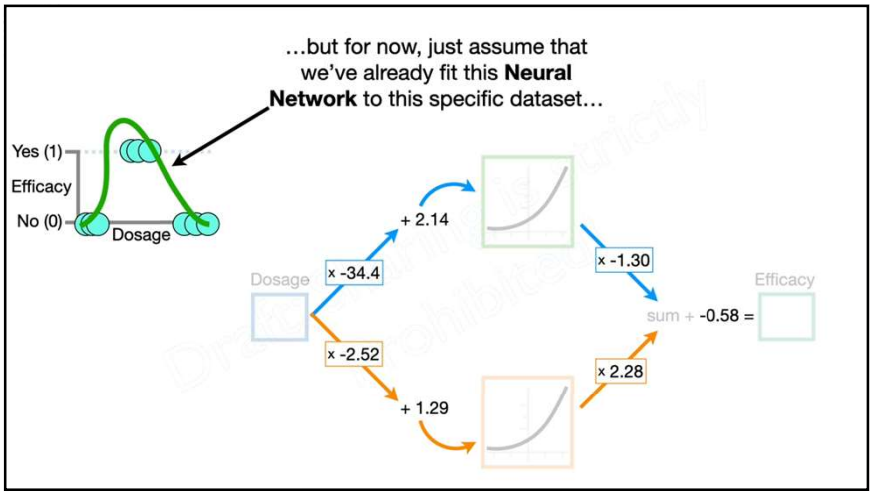

NOTE: The numbers along each connection represent parameter values that were estimated when this Neural Network was fit to the data.
Dosage
x -34.4
+ 2.14
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58 =
Efficacy

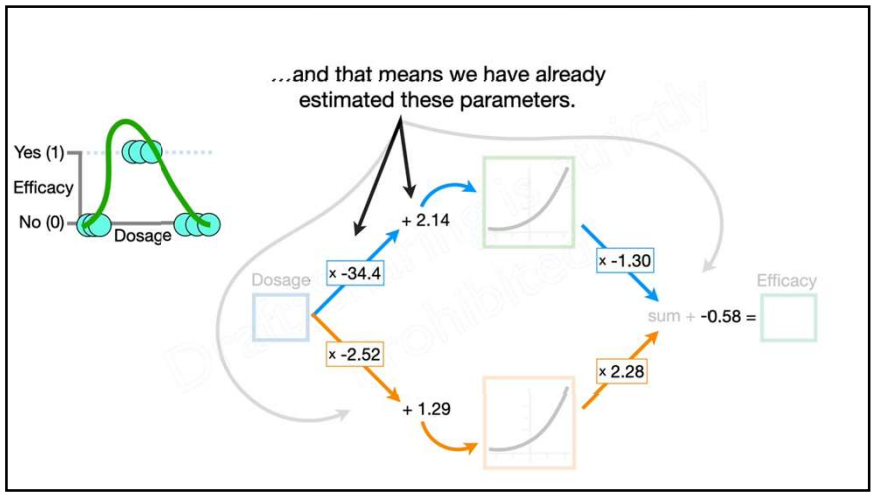

For now, just know that these parameter estimates are analogous to the slope and intercept values that we solve for when we fit a straight line to data.
y = slope x x + intercept
??? ???
y
x
Dosage
x -34.4
+ 2.14
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58 =
Efficacy

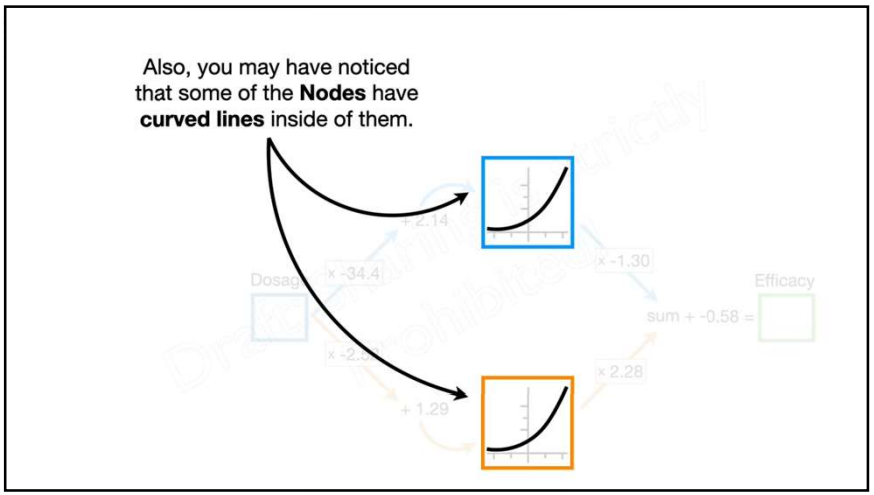

For now, just know that these parameter estimates are analogous to the slope and intercept values that we solve for when we fit a straight line to data.
y = 2 x x + 1.3
y
x
Dosage
x -34.4
+ 2.14
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58 =
Efficacy


Likewise, a Neural Network starts out with unknown parameter values…
Dosage
x ???
+ ???
x ???
+ ???
x ???
x ???
sum + ??? =
Efficacy

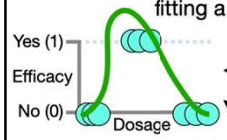

…that are estimated when we fit the **Neural Network** to a dataset using a method called **Backpropagation**.
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x ???
+ ???
x ???
+ ???
x ???
x ???
sum + ??? =
Efficacy


…but for now, just assume that we've already fit this **Neural Network** to this specific dataset…
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
+ 2.14
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58 =
Efficacy


…and that means we have already estimated these parameters.
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
+ 2.14
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58 =
Efficacy


Also, you may have noticed that some of the **Nodes** have **curved lines** inside of them.

These **bent** or **curved lines** are the building blocks for fitting a **squiggle** to data.

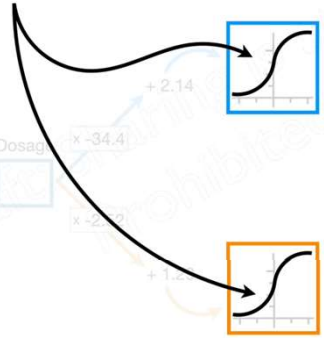


NOTE: There are many common **bent** or **curved lines** that we can choose for a **Neural Network**.

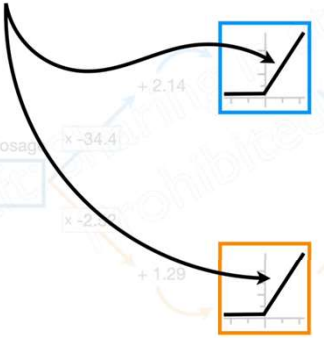
This specific **curved line** is called **softplus**, which sounds like a brand of toilet paper.

Alternatively, we could use this **bent line**, called, **ReLU**, which is short for **Rectified Linear Unit** and sounds like a robot.

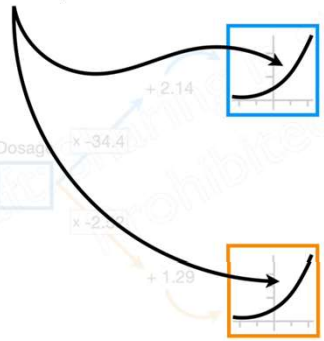
Or we could use a **sigmoid** shape or any other **bent** or **curved line**.



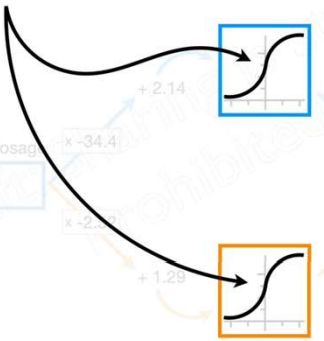
The **curved** or **bent lines** are called **Activation Functions**.



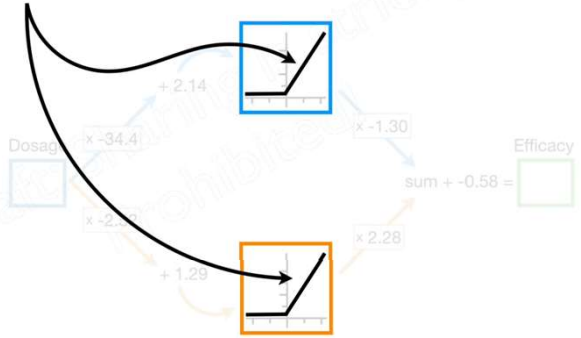
When you build a **Neural Network**, you have to decide which **Activation Function** or **Functions** you want to use.



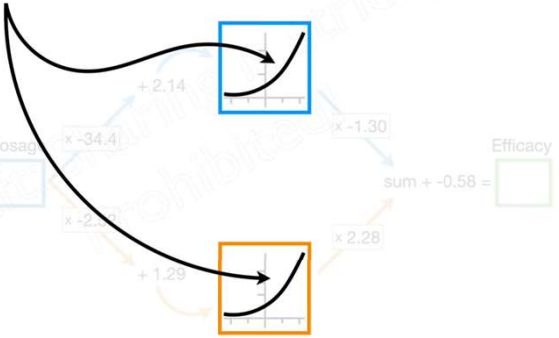
When most people *teach* **Neural Networks**, they use the **sigmoid Activation Function**.



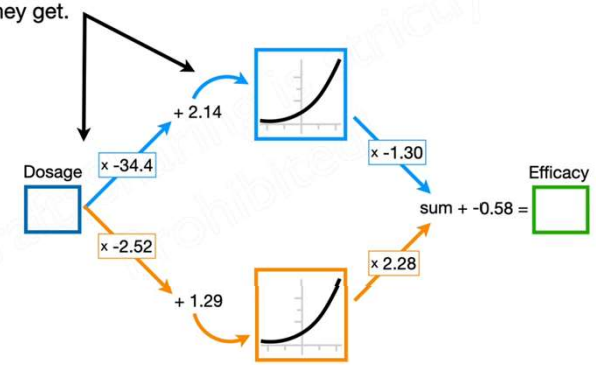
However, in *practice*, it is much more common to use the **ReLU Activation Function**…



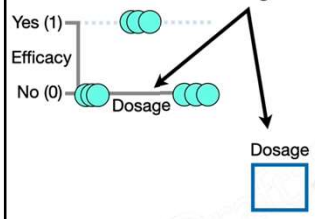
…or the **softplus Activation Function**.

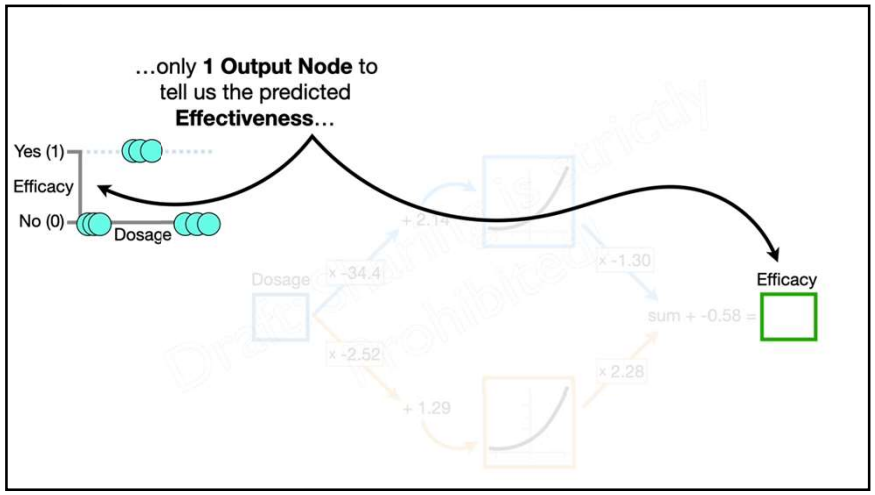


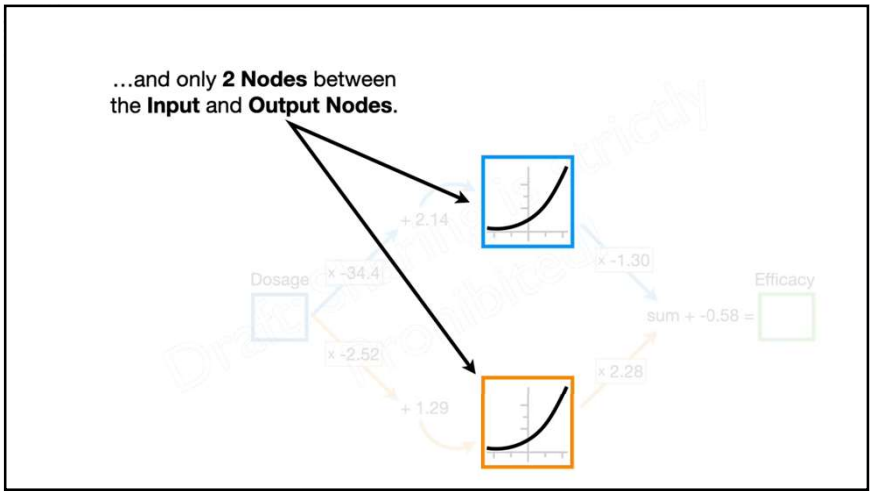
NOTE: This specific **Neural Network** is about as simple as they get.

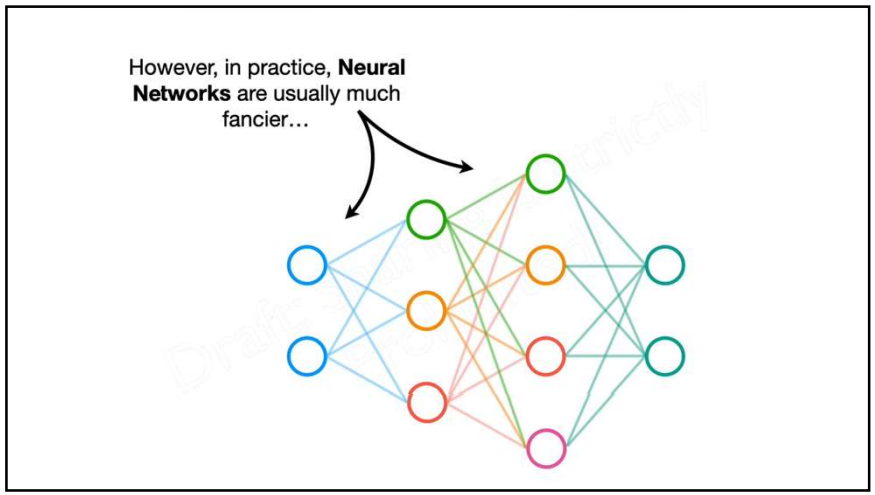


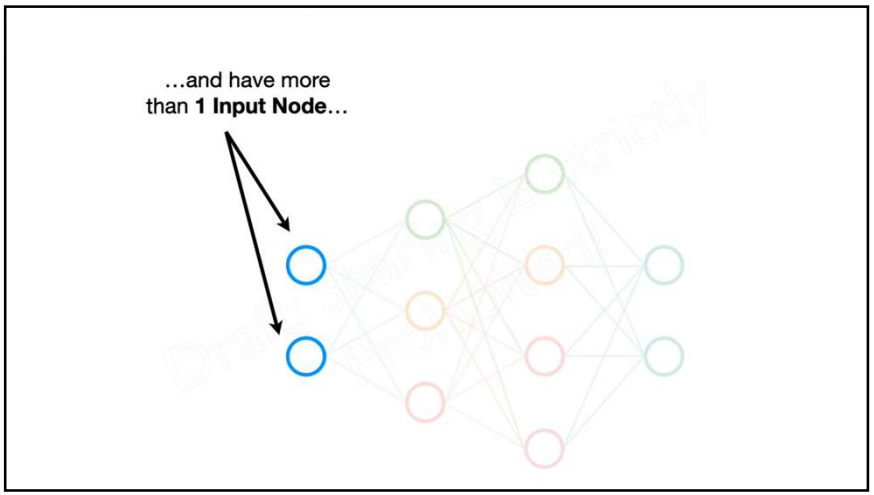
It only has **1 Input Node** where we plug in the **Dosage**…

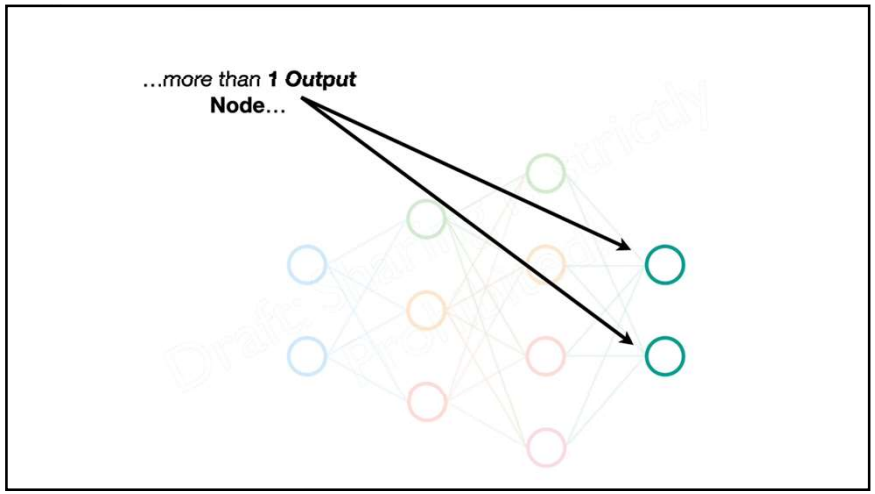


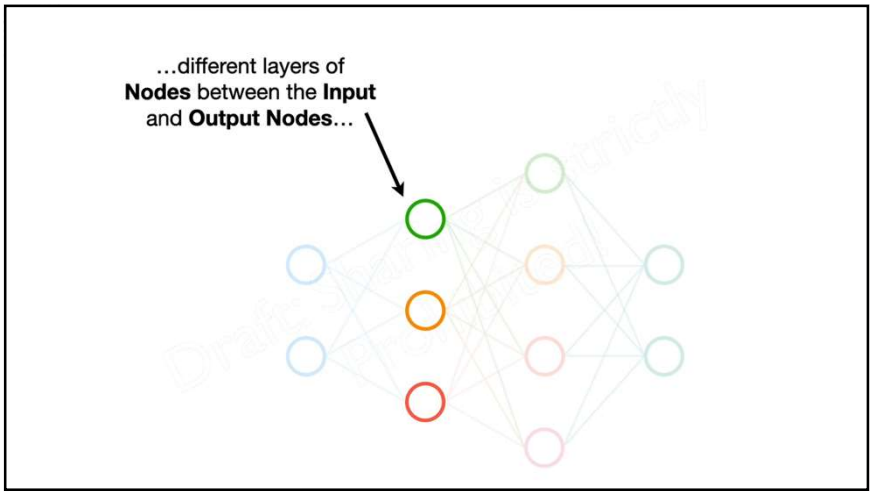

...only **1 Output Node** to tell us the predicted **Effectiveness**...
Yes (1)
Efficacy
No (0)
Dosage
Efficacy

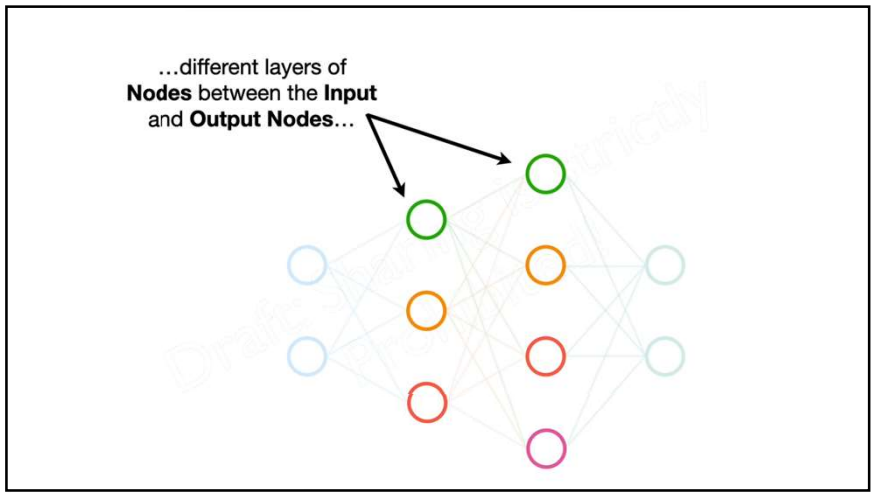

...and only **2 Nodes** between the **Input** and **Output Nodes**.

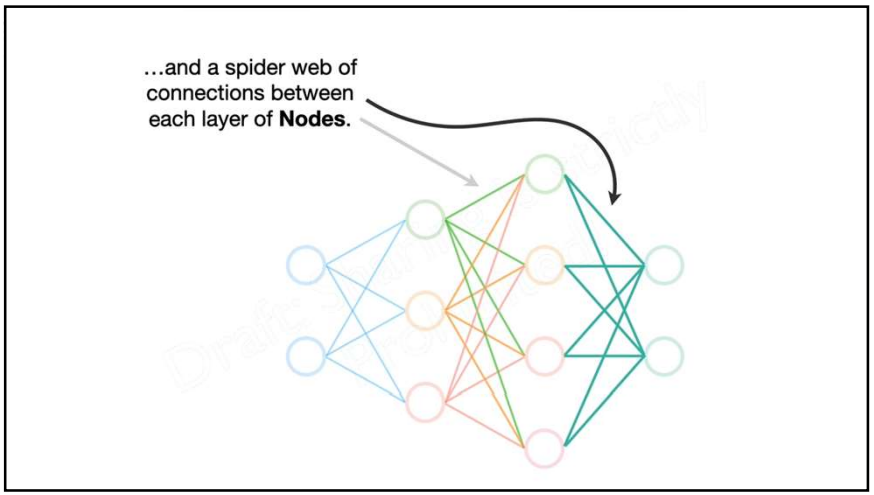

However, in practice, **Neural Networks** are usually much fancier...


...and have more than **1 Input Node**...

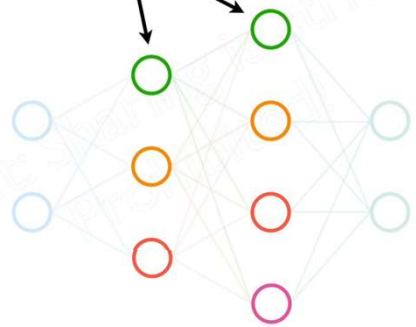

...more than 1 Output Node...


...different layers of Nodes between the Input and Output Nodes...

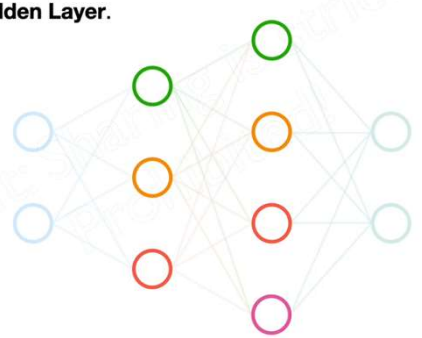

...different layers of Nodes between the Input and Output Nodes...


...and a spider web of connections between each layer of Nodes.

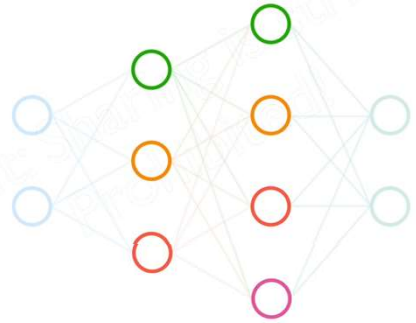
The layers of **Nodes** between the **Input** and **Output Nodes** are called **Hidden Layers**.



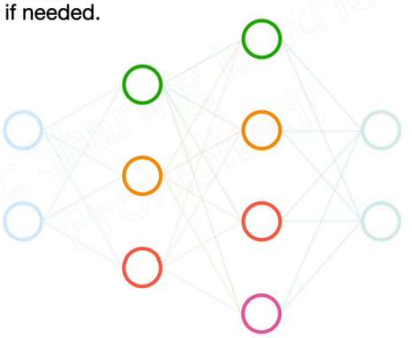
When you build a **Neural Network**, one of the first things you do is decide how many **Hidden Layers** you want, and how many **Nodes** go into each **Hidden Layer**.



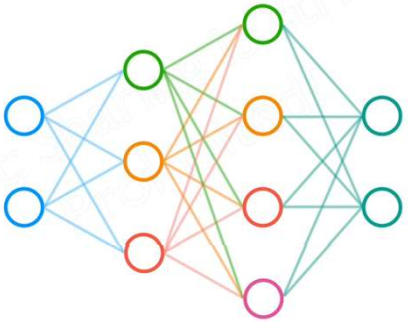
Although there are rules of thumb for making decisions about the **Hidden Layers**…



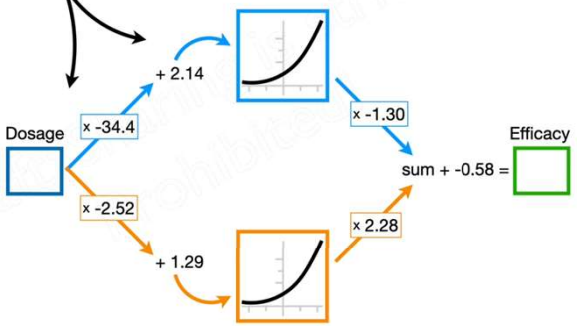
…you essentially make a guess and see how well the **Neural Network** performs, adding more layers and nodes if needed.



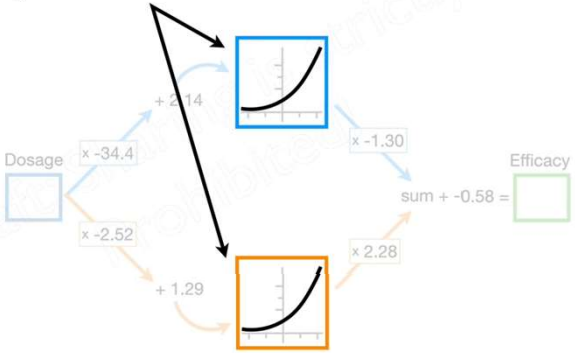
Now, even though this **Neural Network** looks fancy, it is still made from the same parts…



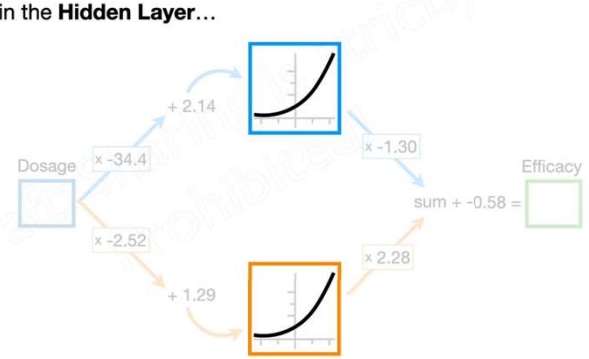
…used in this simple **Neural Network**…

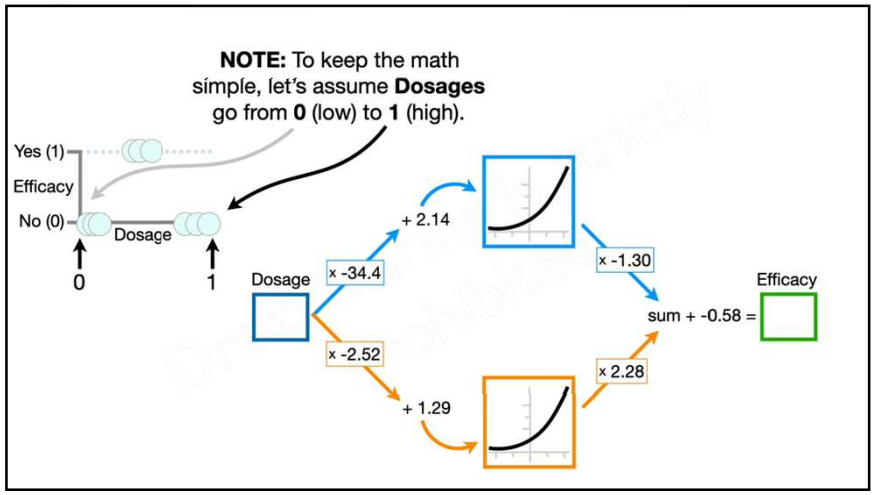


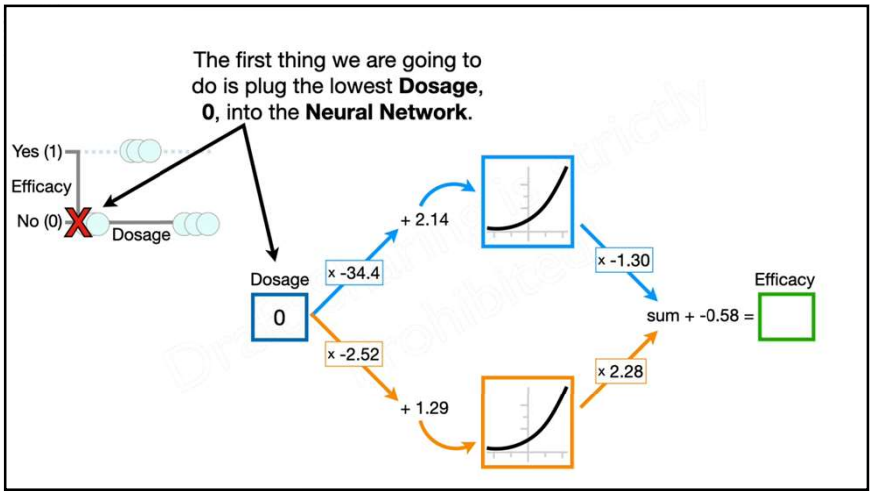
…*which only has* **1 Hidden Layer** with **2 Nodes**.

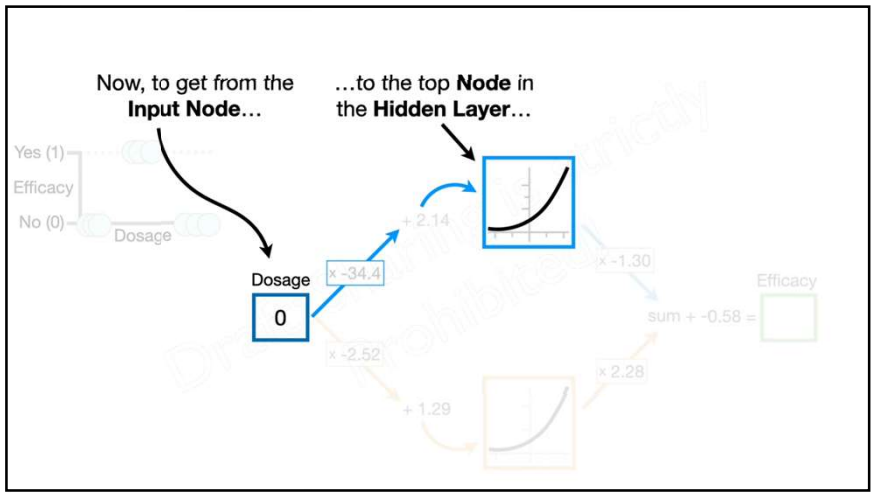


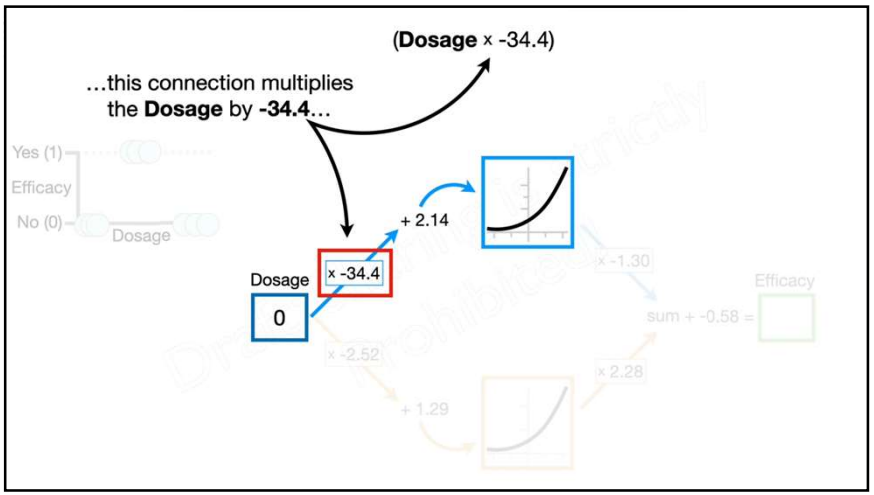
So let's learn how this **Neural Network** creates new shapes from the **curved** or **bent lines** in the **Hidden Layer**…

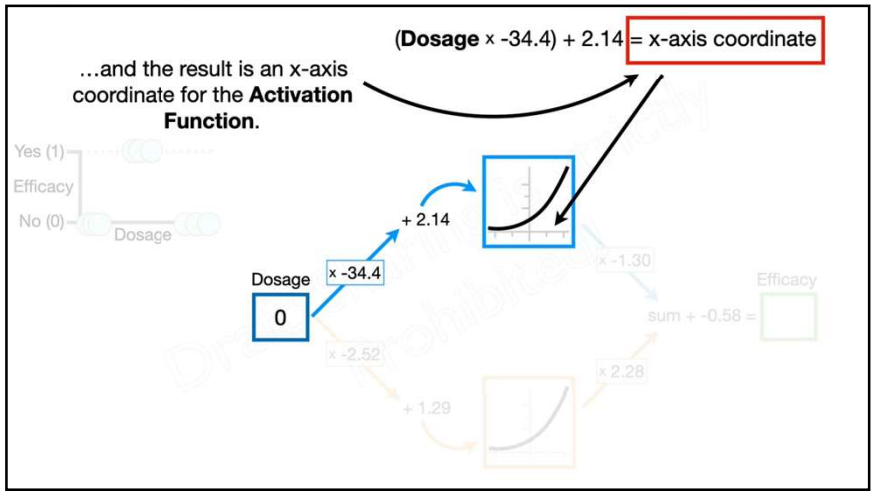


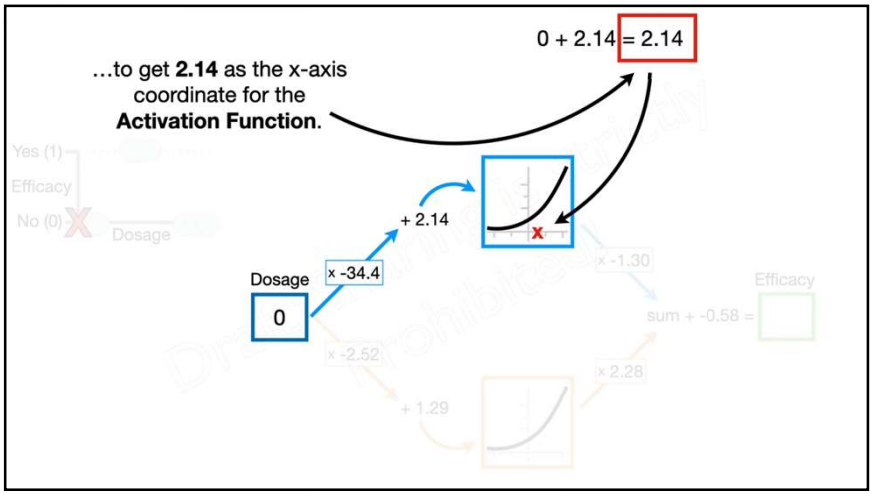

NOTE: To keep the math simple, let's assume Dosages go from 0 (low) to 1 (high).
Yes (1)
Efficacy
No (0)
Dosage
0
1
Dosage
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

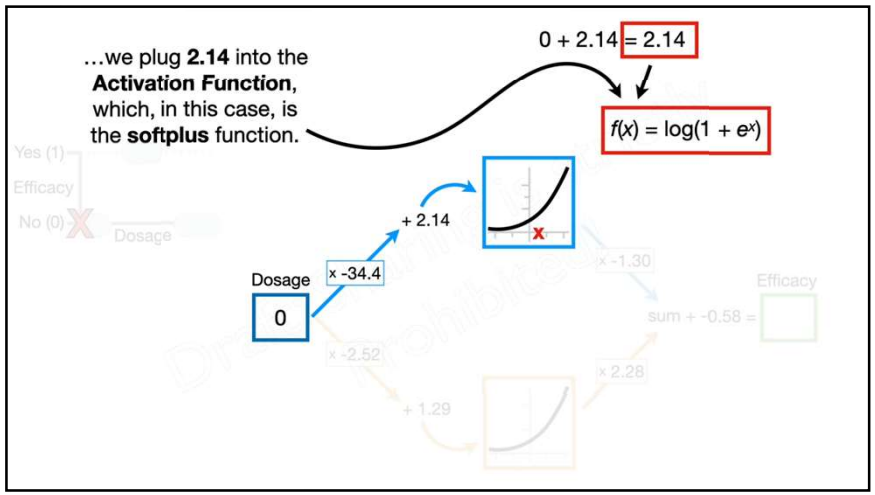

The first thing we are going to do is plug the lowest Dosage, 0, into the Neural Network.
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

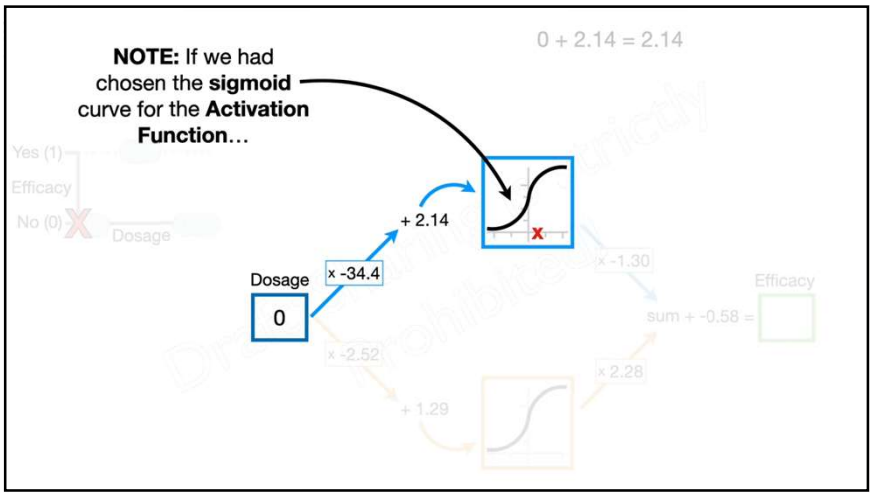

Now, to get from the Input Node…
…to the top Node in the Hidden Layer…
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

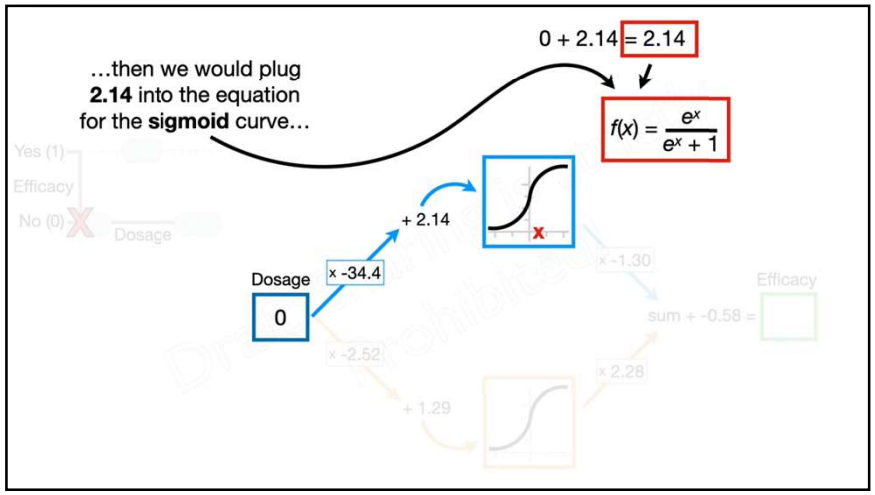

(Dosage x -34.4)
…this connection multiplies the Dosage by -34.4…
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

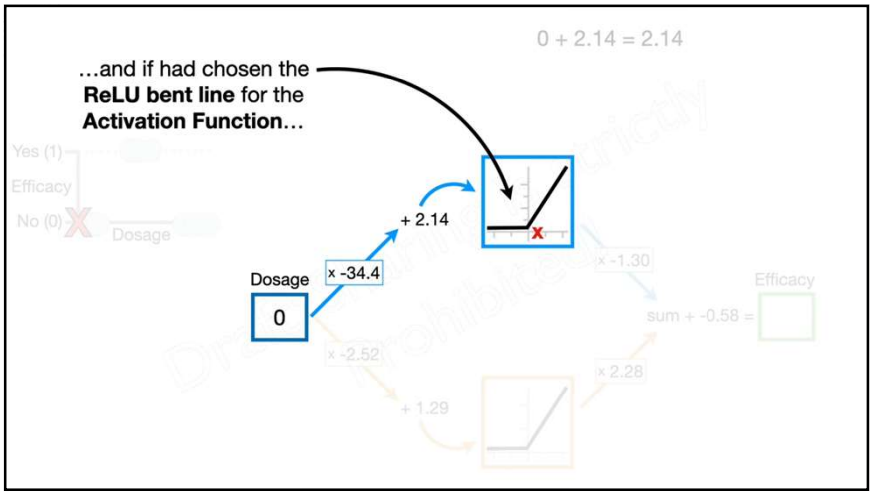

(Dosage x -34.4) + 2.14 = x-axis coordinate
...and the result is an x-axis coordinate for the Activation Function.
Dosage
0
x -34.4
+ 2.14

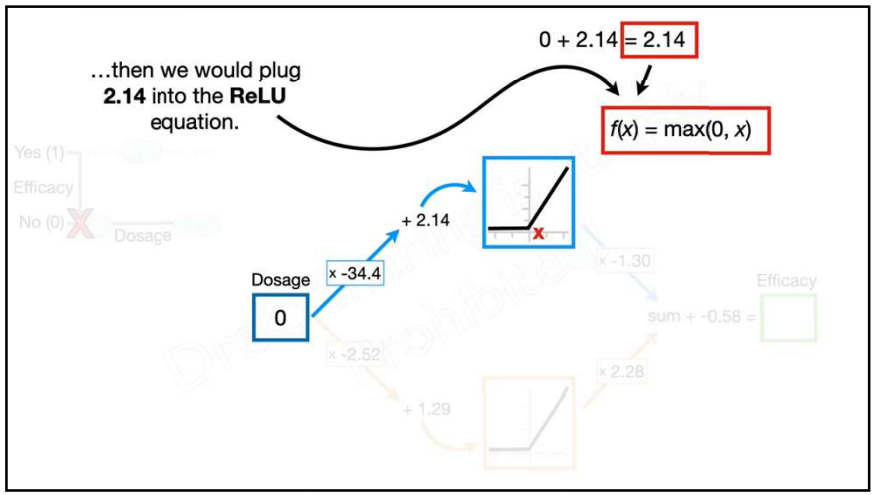

0 + 2.14 = 2.14
...to get 2.14 as the x-axis coordinate for the Activation Function.
Dosage
0
x -34.4
+ 2.14

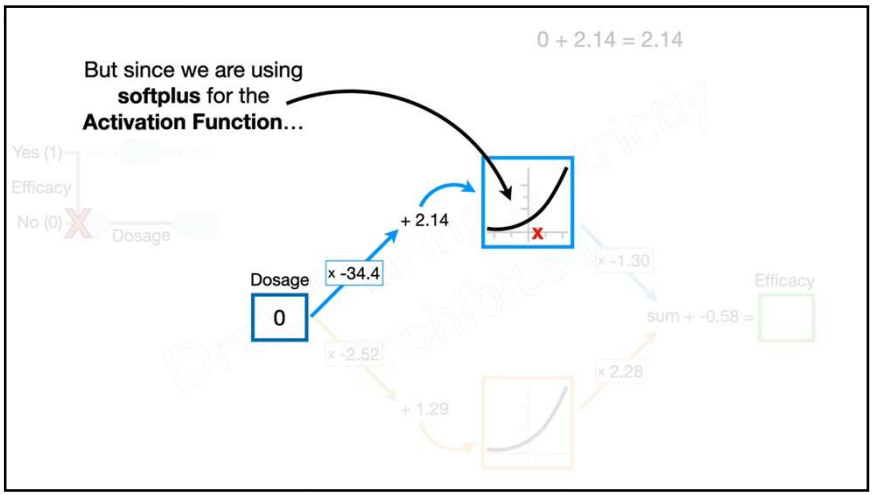

0 + 2.14 = 2.14
...we plug 2.14 into the Activation Function, which, in this case, is the softplus function.
f(x) = log(1 + e^x)
Dosage
0
x -34.4
+ 2.14

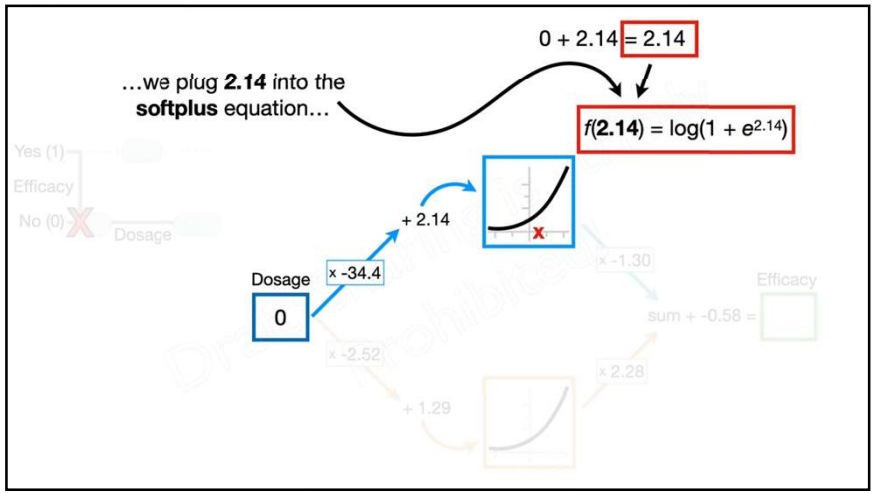

0 + 2.14 = 2.14
NOTE: If we had chosen the sigmoid curve for the Activation Function...
Dosage
0
x -34.4
+ 2.14

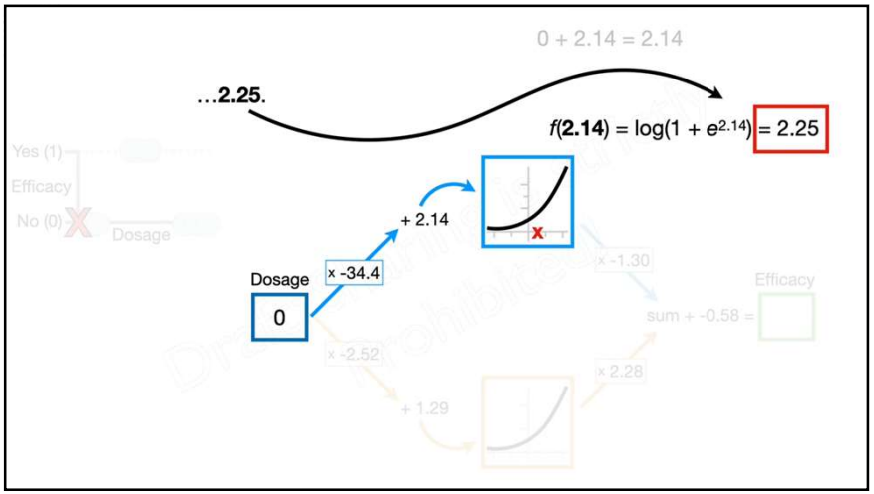

0 + 2.14 = 2.14
…then we would plug
2.14 into the equation
for the sigmoid curve…
$f(x) = \frac{e^x}{e^x + 1}$
Dosage
0
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

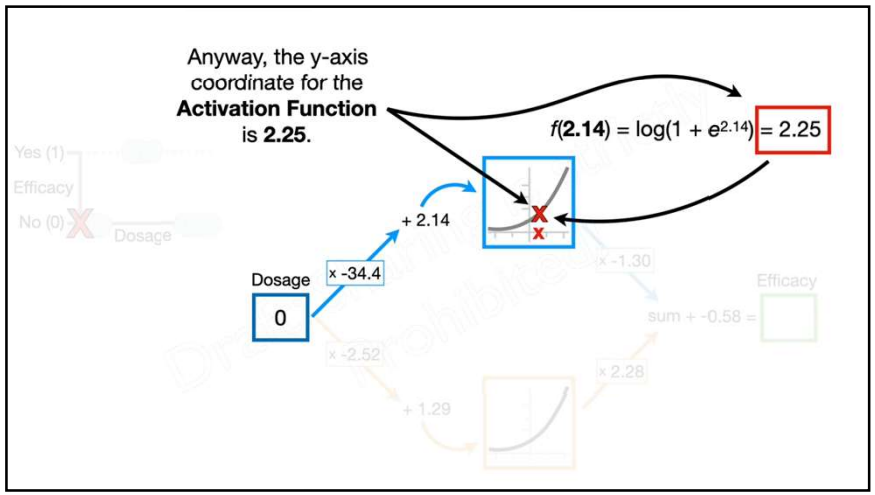

0 + 2.14 = 2.14
…and if had chosen the
ReLU bent line for the
Activation Function…
Dosage
0
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

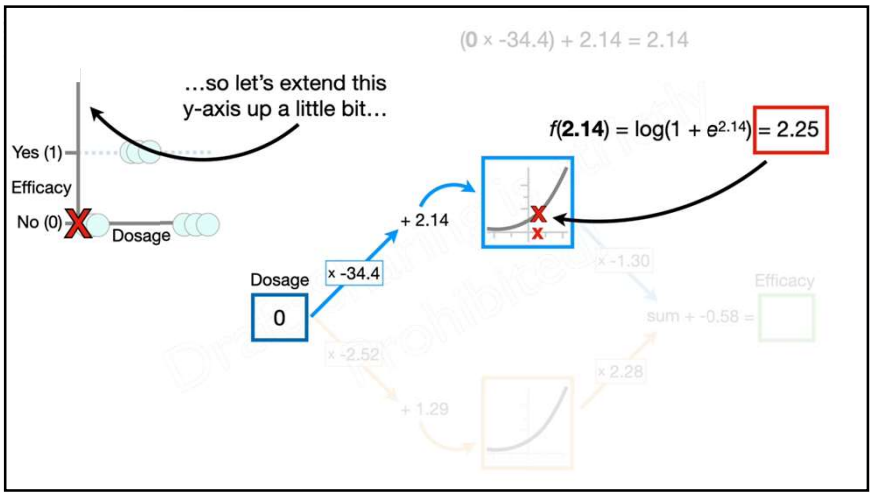

0 + 2.14 = 2.14
…then we would plug
2.14 into the ReLU
equation.
f(x) = max(0, x)
Dosage
0
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

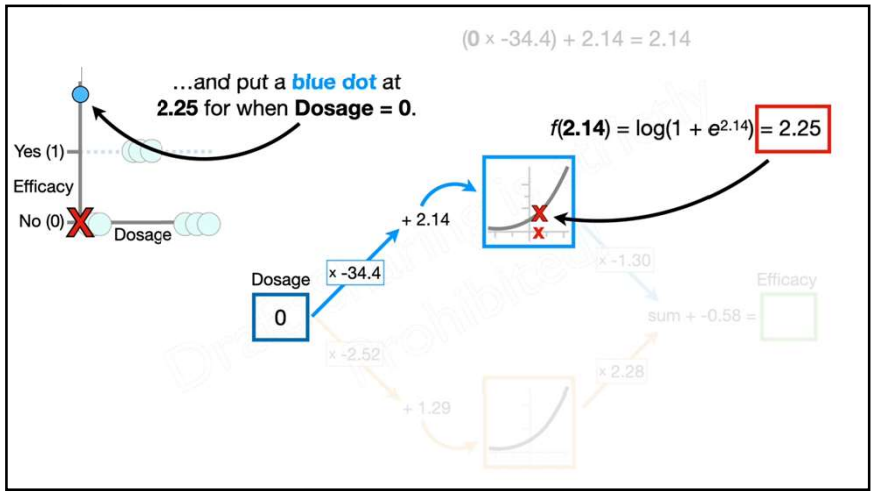

0 + 2.14 = 2.14
But since we are using
softplus for the
Activation Function…
Dosage
0
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

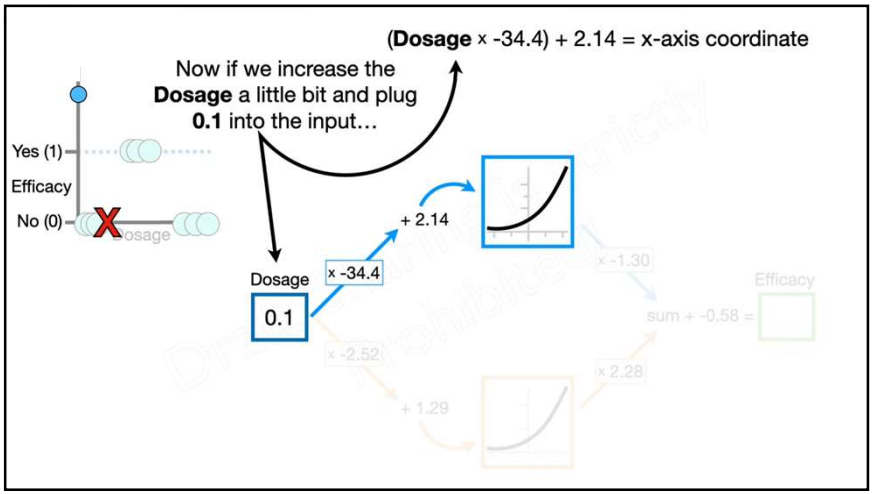

0 + 2.14 = 2.14
…we plug **2.14** into the **softplus** equation…
$f(\mathbf{2.14}) = \log(1 + e^{2.14})$
Dosage
0
x -34.4
+ 2.14
x -2.52
+ 1.29
x 2.28
x -1.30
sum + -0.58 =
Efficacy
Yes (1)
Efficacy
No (0)
Dosage

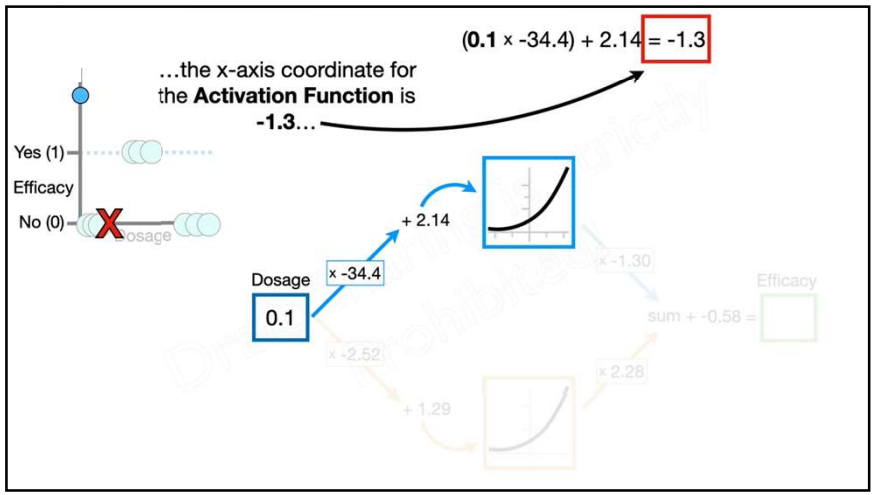

0 + 2.14 = 2.14
…**2.25**.
$f(\mathbf{2.14}) = \log(1 + e^{2.14}) = 2.25$
Dosage
0
x -34.4
+ 2.14
x -2.52
+ 1.29
x 2.28
x -1.30
sum + -0.58 =
Efficacy
Yes (1)
Efficacy
No (0)
Dosage

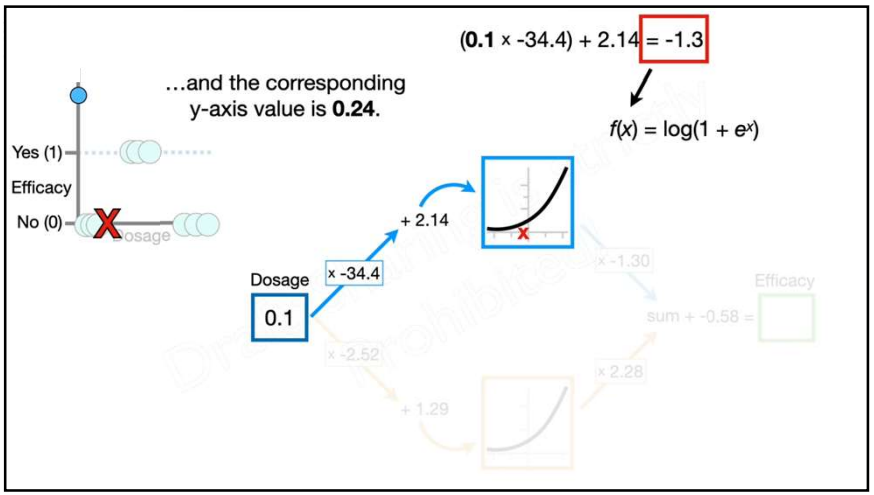

Anyway, the y-axis coordinate for the **Activation Function** is **2.25**.
$f(\mathbf{2.14}) = \log(1 + e^{2.14}) = 2.25$
Dosage
0
x -34.4
+ 2.14
x -2.52
+ 1.29
x 2.28
x -1.30
sum + -0.58 =
Efficacy
Yes (1)
Efficacy
No (0)
Dosage

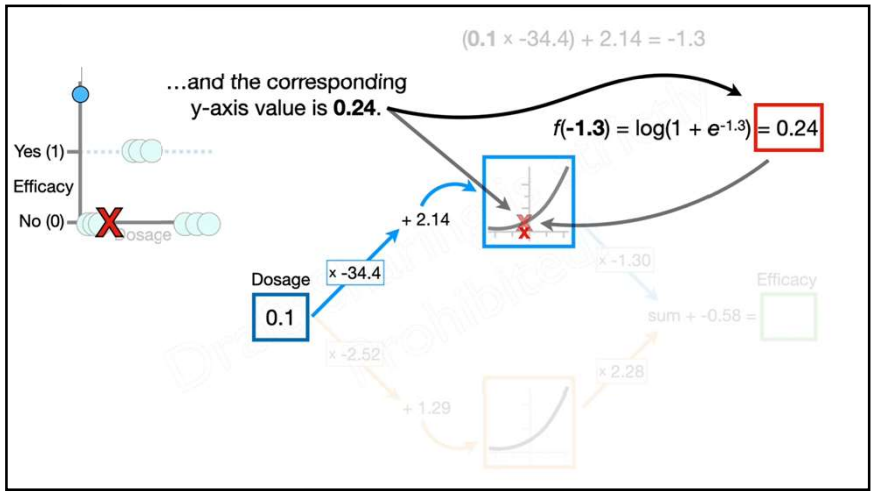

(**0** x -34.4) + 2.14 = 2.14
…so let's extend this y-axis up a little bit…
$f(\mathbf{2.14}) = \log(1 + e^{2.14}) = 2.25$
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0
x -34.4
+ 2.14
x -2.52
+ 1.29
x 2.28
x -1.30
sum + -0.58 =
Efficacy

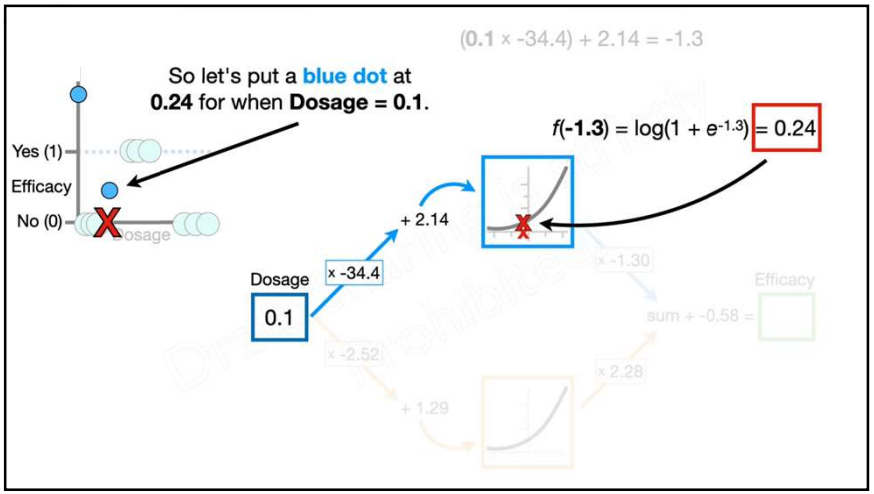

(0 x -34.4) + 2.14 = 2.14
…and put a blue dot at 2.25 for when Dosage = 0.
f(2.14) = log(1 + e^2.14) = 2.25
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0
x -34.4
+ 2.14
x -1.30
sum + -0.58 =
Efficacy
x -2.52
x 2.28
+ 1.29

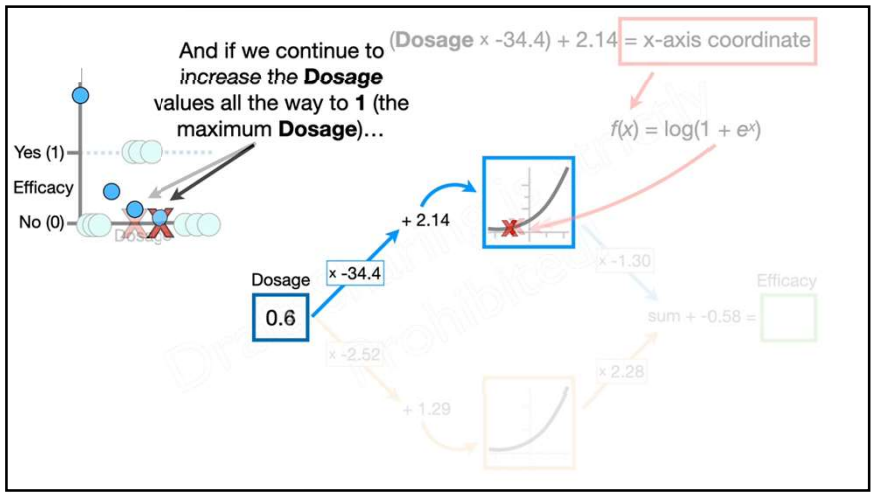

(Dosage x -34.4) + 2.14 = x-axis coordinate
Now if we increase the Dosage a little bit and plug 0.1 into the input…
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0.1
x -34.4
+ 2.14
x -1.30
sum + -0.58 =
Efficacy
x -2.52
x 2.28
+ 1.29

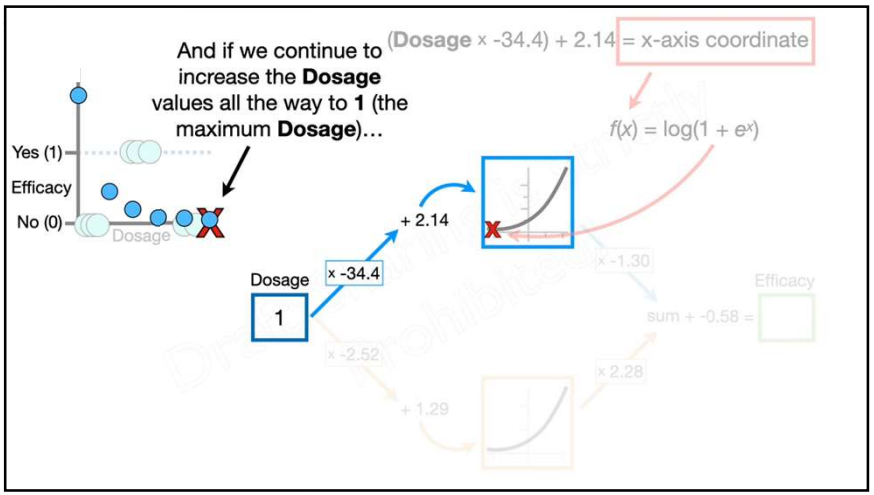

(0.1 x -34.4) + 2.14 = -1.3
…the x-axis coordinate for the Activation Function is -1.3…
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0.1
x -34.4
+ 2.14
x -1.30
sum + -0.58 =
Efficacy
x -2.52
x 2.28
+ 1.29

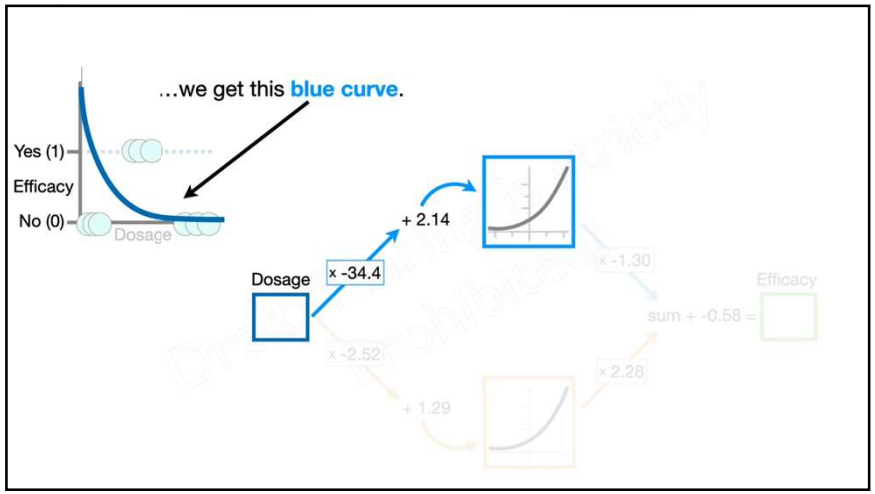

(0.1 x -34.4) + 2.14 = -1.3
…and the corresponding y-axis value is 0.24.
f(x) = log(1 + e^x)
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0.1
x -34.4
+ 2.14
x -1.30
sum + -0.58 =
Efficacy
x -2.52
x 2.28
+ 1.29

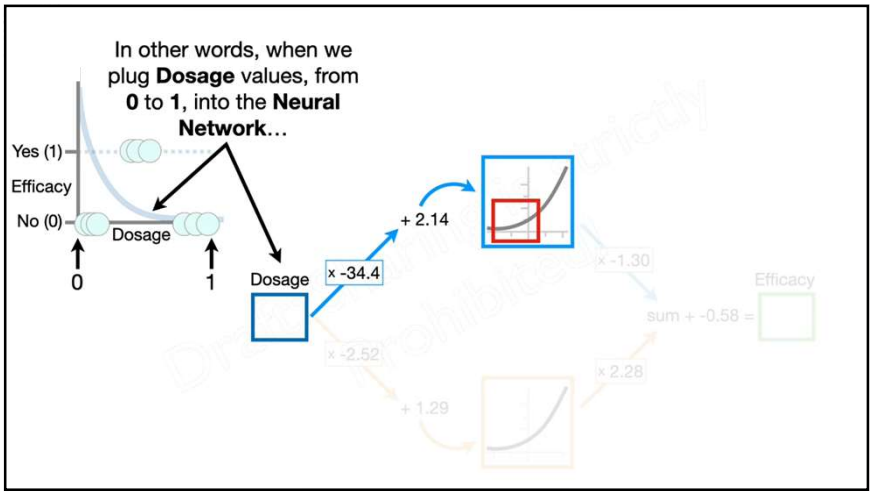

(0.1 x -34.4) + 2.14 = -1.3
...and the corresponding y-axis value is 0.24.
f(-1.3) = log(1 + e^-1.3) = 0.24
Yes (1)
Efficacy
No (0)
Dosage
+ 2.14
Dosage
x -34.4
0.1
x -1.30
Efficacy
sum + -0.58 =
x -2.52
x 2.28
+ 1.29

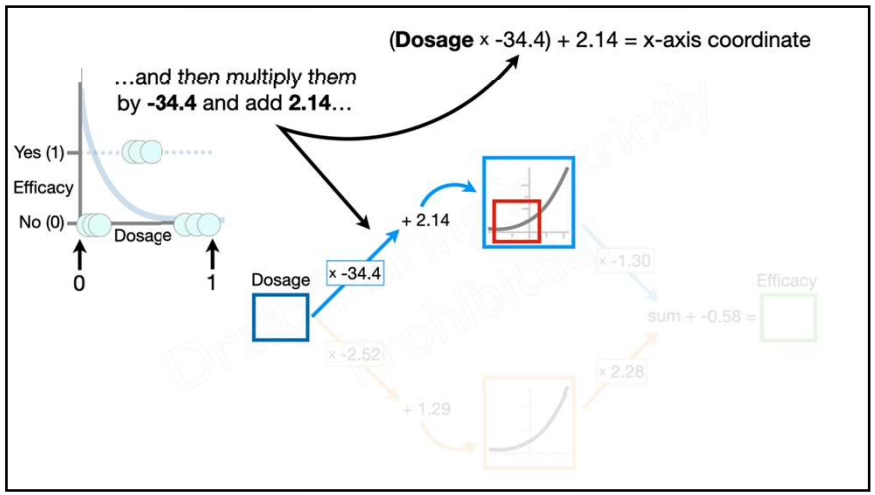

(0.1 x -34.4) + 2.14 = -1.3
So let's put a blue dot at 0.24 for when Dosage = 0.1.
f(-1.3) = log(1 + e^-1.3) = 0.24
Yes (1)
Efficacy
No (0)
Dosage
+ 2.14
Dosage
x -34.4
0.1
x -1.30
Efficacy
sum + -0.58 =
x -2.52
x 2.28
+ 1.29

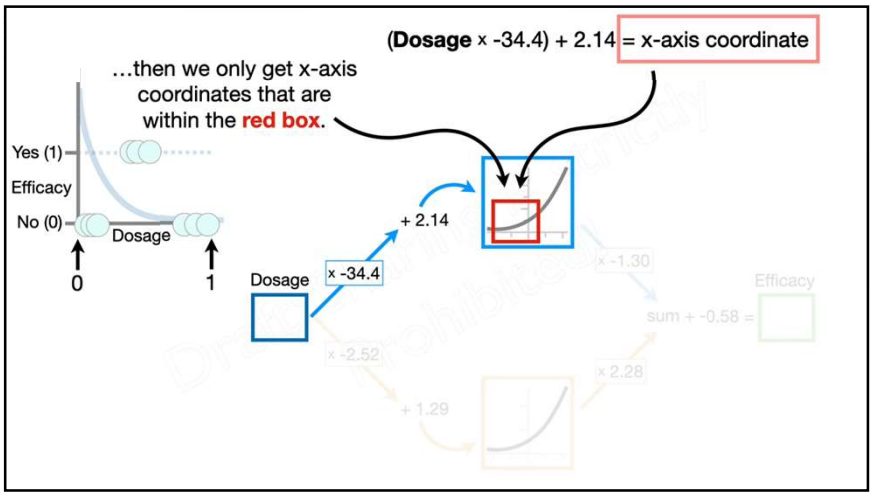

(Dosage x -34.4) + 2.14 = x-axis coordinate
And if we continue to increase the Dosage values all the way to 1 (the maximum Dosage)...
f(x) = log(1 + e^x)
Yes (1)
Efficacy
No (0)
Dosage
+ 2.14
Dosage
x -34.4
0.6
x -1.30
Efficacy
sum + -0.58 =
x -2.52
x 2.28
+ 1.29

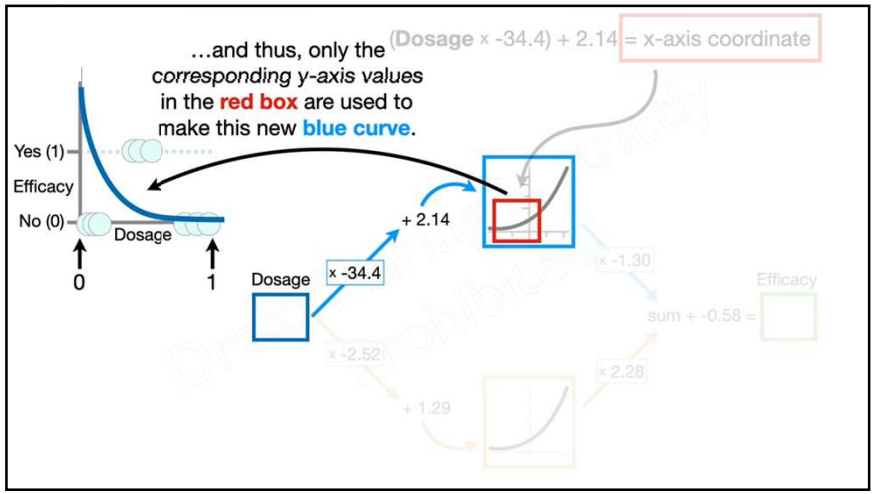

(Dosage x -34.4) + 2.14 = x-axis coordinate
And if we continue to increase the Dosage values all the way to 1 (the maximum Dosage)...
f(x) = log(1 + e^x)
Yes (1)
Efficacy
No (0)
Dosage
+ 2.14
Dosage
x -34.4
1
x -1.30
Efficacy
sum + -0.58 =
x -2.52
x 2.28
+ 1.29

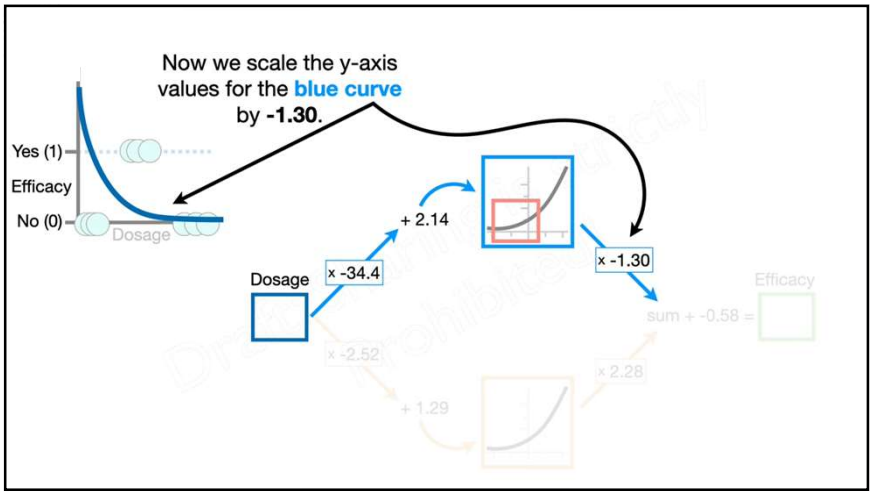

...we get this blue curve.
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
+ 2.14
x -1.30
x -2.52
x 2.28
+ 1.29
sum + -0.58 =
Efficacy

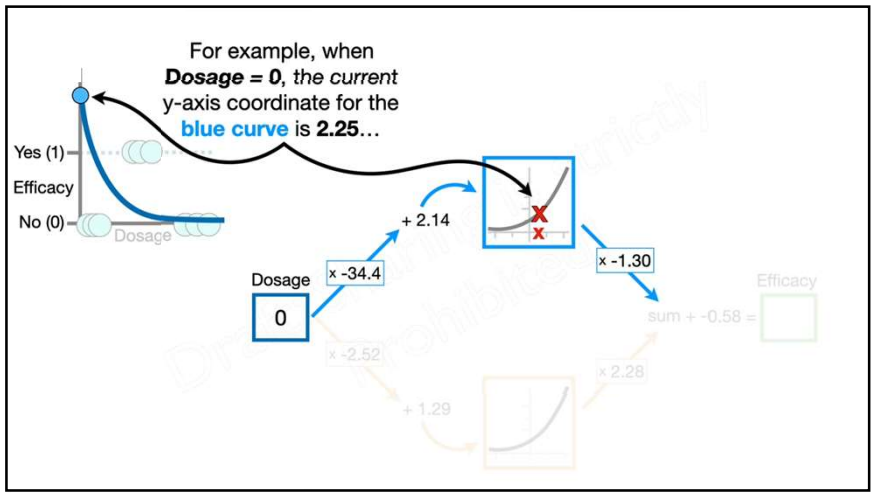

In other words, when we plug Dosage values, from 0 to 1, into the Neural Network...
Yes (1)
Efficacy
No (0)
Dosage
0
1
Dosage
x -34.4
+ 2.14
x -1.30
x -2.52
x 2.28
+ 1.29
sum + -0.58 =
Efficacy

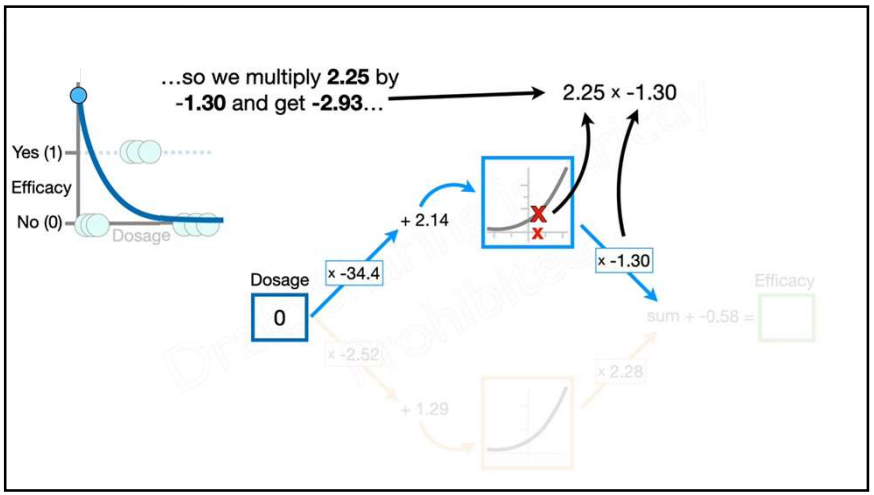

(Dosage x -34.4) + 2.14 = x-axis coordinate
...and then multiply them by -34.4 and add 2.14...
Yes (1)
Efficacy
No (0)
Dosage
0
1
Dosage
x -34.4
+ 2.14
x -1.30
x -2.52
x 2.28
+ 1.29
sum + -0.58 =
Efficacy

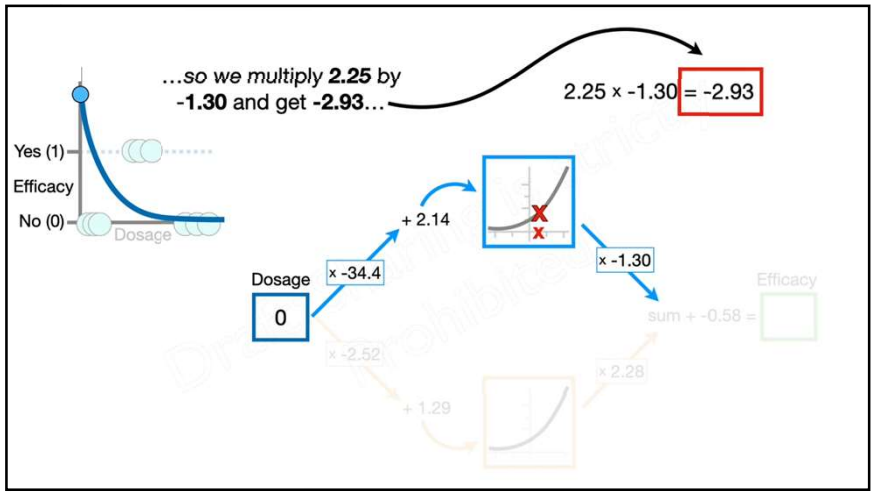

(Dosage x -34.4) + 2.14 = x-axis coordinate
...then we only get x-axis coordinates that are within the red box.
Yes (1)
Efficacy
No (0)
Dosage
0
1
Dosage
x -34.4
+ 2.14
x -1.30
x -2.52
x 2.28
+ 1.29
sum + -0.58 =
Efficacy

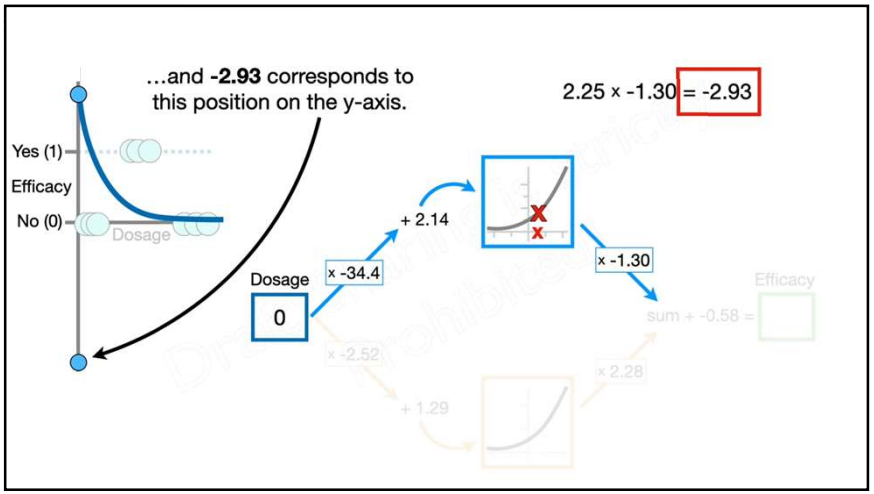

…and thus, only the corresponding y-axis values in the red box are used to make this new blue curve.
(Dosage x -34.4) + 2.14 = x-axis coordinate
Yes (1)
Efficacy
No (0)
Dosage
0
1
Dosage
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

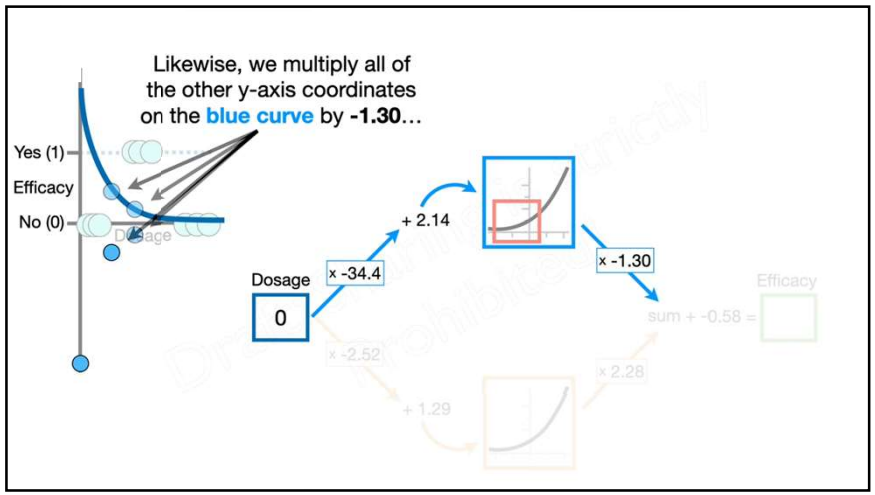

Now we scale the y-axis values for the blue curve by -1.30.
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

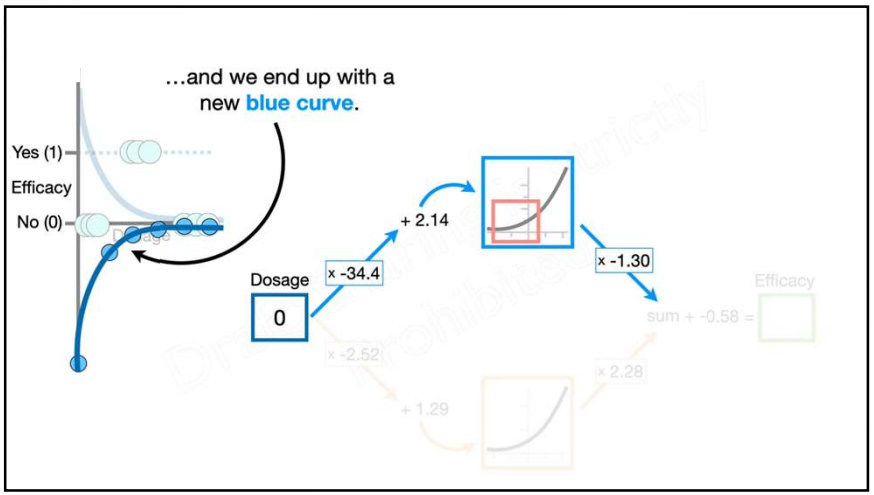

For example, when Dosage = 0, the current y-axis coordinate for the blue curve is 2.25…
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

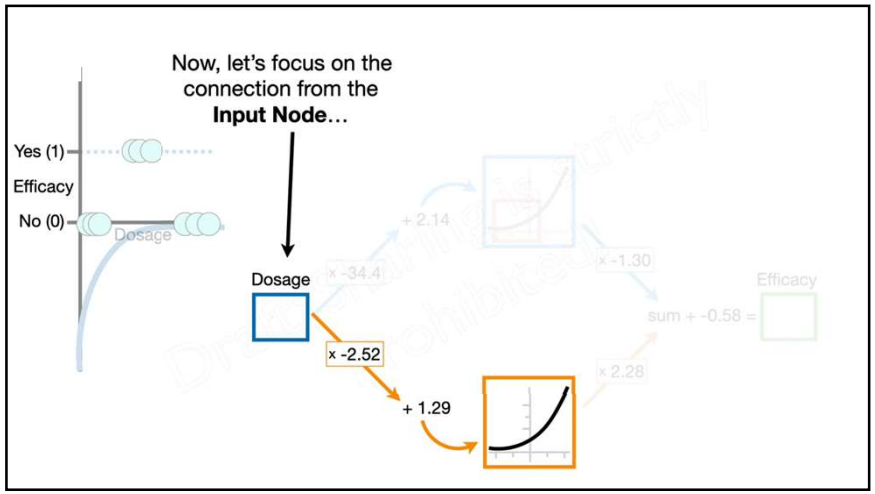

…so we multiply 2.25 by -1.30 and get -2.93…
2.25 x -1.30
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

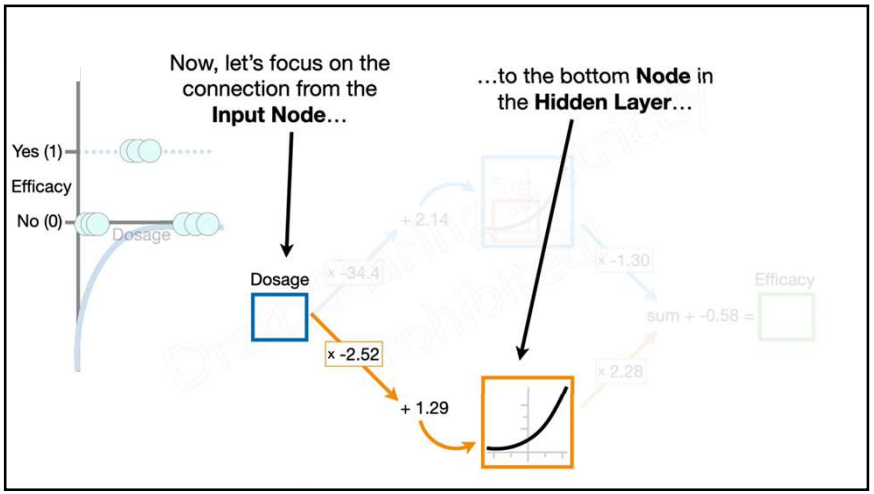

...so we multiply 2.25 by -1.30 and get -2.93...
2.25 x -1.30 = -2.93
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

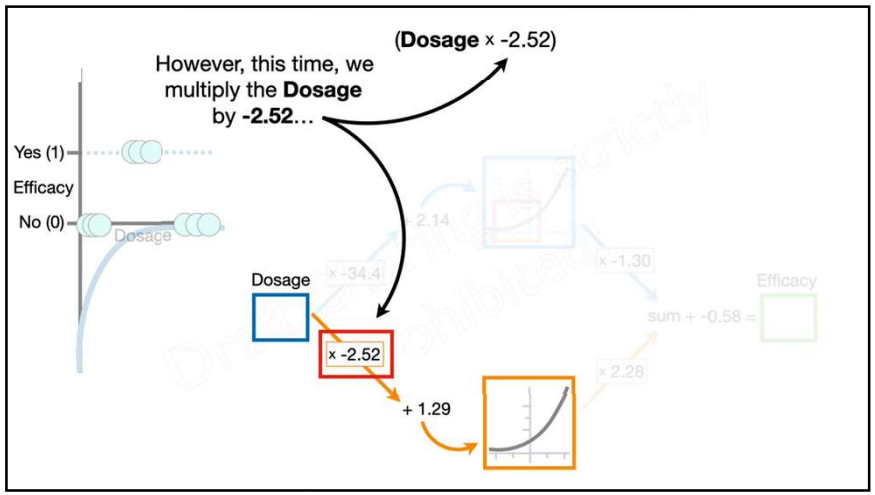

...and -2.93 corresponds to this position on the y-axis.
2.25 x -1.30 = -2.93
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

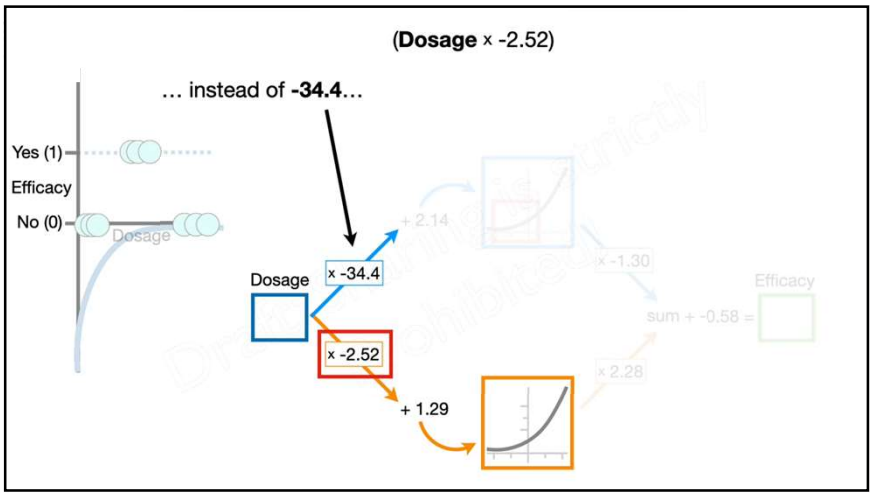

Likewise, we multiply all of the other y-axis coordinates on the blue curve by -1.30...
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

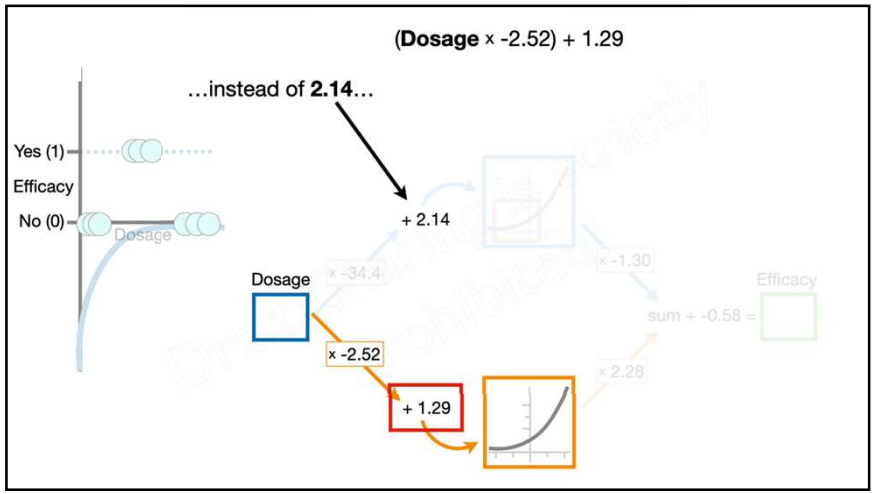

...and we end up with a new blue curve.
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

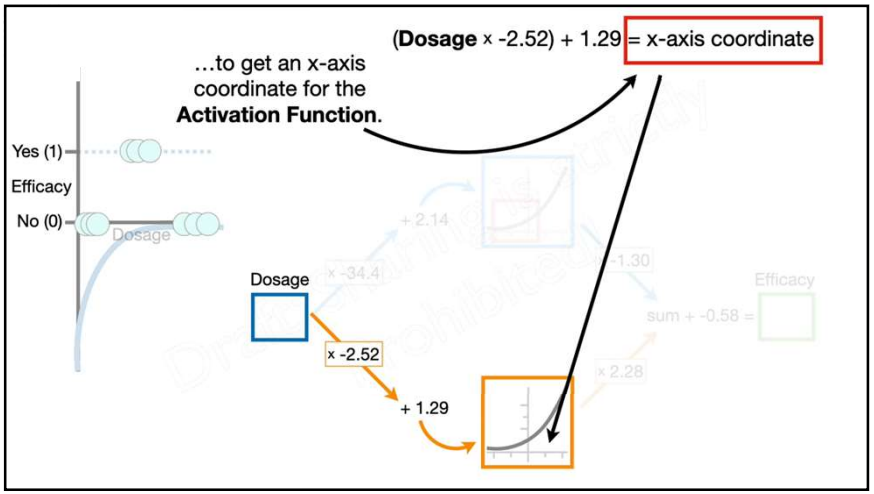

Now, let's focus on the connection from the **Input Node**...
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -2.52
+ 1.29
+ 2.14
x -34.4
x -1.30
sum + -0.58 =
Efficacy
x 2.28

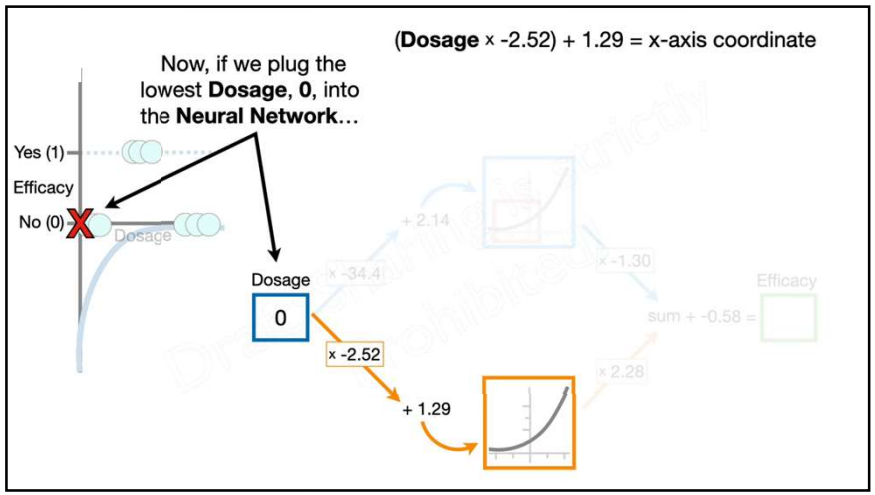

Now, let's focus on the connection from the **Input Node**...
...to the bottom **Node** in the **Hidden Layer**...
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -2.52
+ 1.29
+ 2.14
x -34.4
x -1.30
sum + -0.58 =
Efficacy
x 2.28

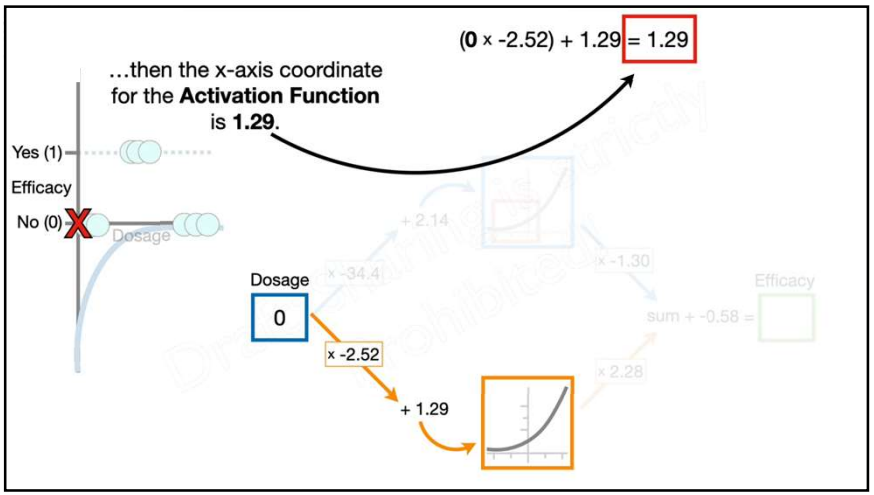

(**Dosage** x -2.52)
However, this time, we multiply the **Dosage** by **-2.52**...
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -2.52
+ 1.29
+ 2.14
x -34.4
x -1.30
sum + -0.58 =
Efficacy
x 2.28

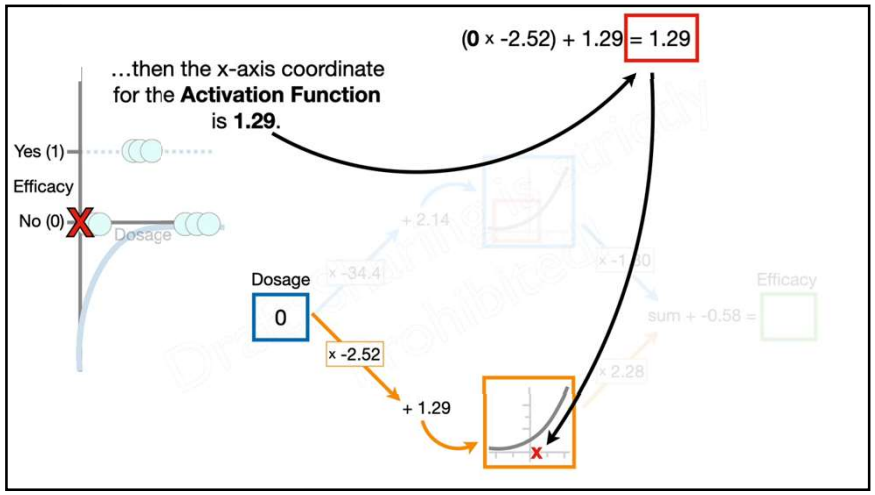

(**Dosage** x -2.52)
... instead of **-34.4**...
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
x -2.52
+ 1.29
+ 2.14
x -1.30
sum + -0.58 =
Efficacy
x 2.28

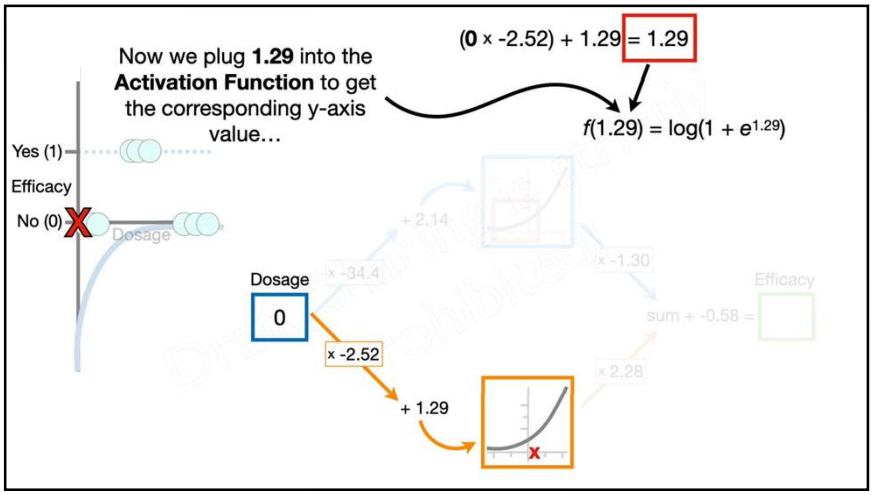

(Dosage x -2.52) + 1.29
...instead of 2.14...
Yes (1)
Efficacy
No (0)
Dosage
+ 2.14
Dosage
x -34.4
x -1.30
sum + -0.58 =
Efficacy
x -2.52
x 2.28
+ 1.29

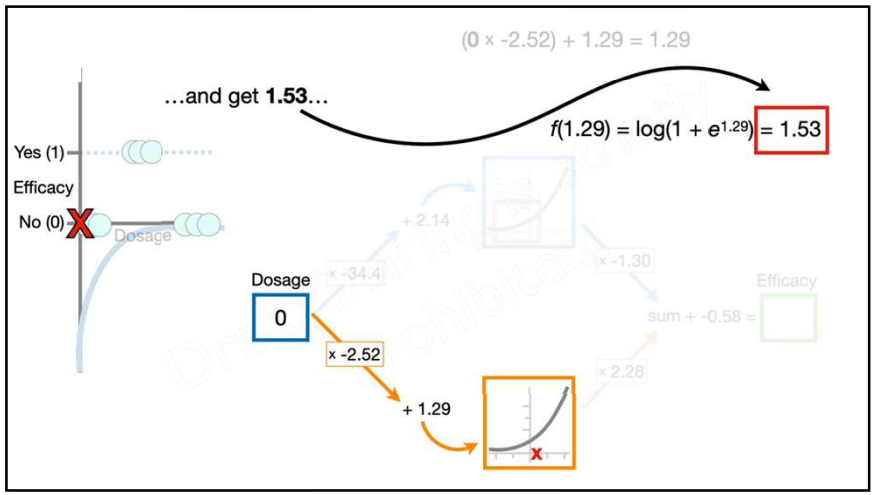

(Dosage x -2.52) + 1.29 = x-axis coordinate
...to get an x-axis coordinate for the Activation Function.
Yes (1)
Efficacy
No (0)
Dosage
+ 2.14
Dosage
x -34.4
x -1.30
sum + -0.58 =
Efficacy
x -2.52
x 2.28
+ 1.29

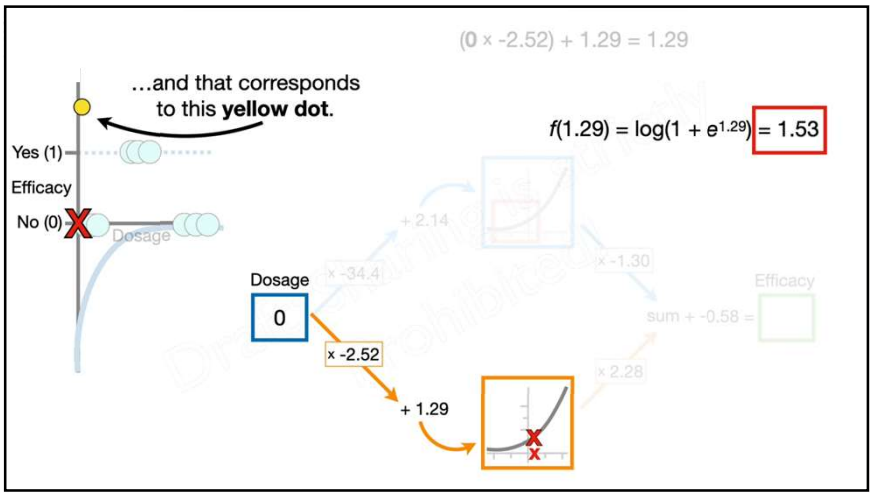

(Dosage x -2.52) + 1.29 = x-axis coordinate
Now, if we plug the lowest Dosage, 0, into the Neural Network...
Yes (1)
Efficacy
No (0)
Dosage
+ 2.14
Dosage
0
x -34.4
x -1.30
sum + -0.58 =
Efficacy
x -2.52
x 2.28
+ 1.29

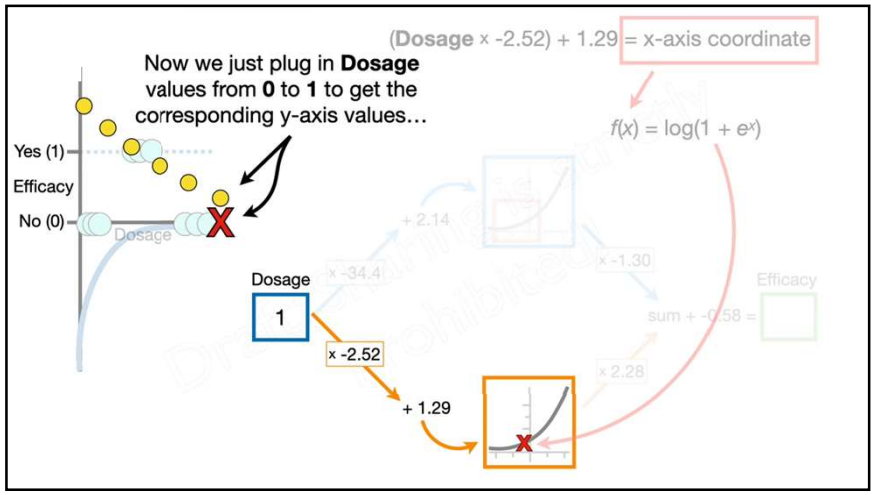

(0 x -2.52) + 1.29 = 1.29
...then the x-axis coordinate for the Activation Function is 1.29.
Yes (1)
Efficacy
No (0)
Dosage
+ 2.14
Dosage
0
x -34.4
x -1.30
sum + -0.58 =
Efficacy
x -2.52
x 2.28
+ 1.29

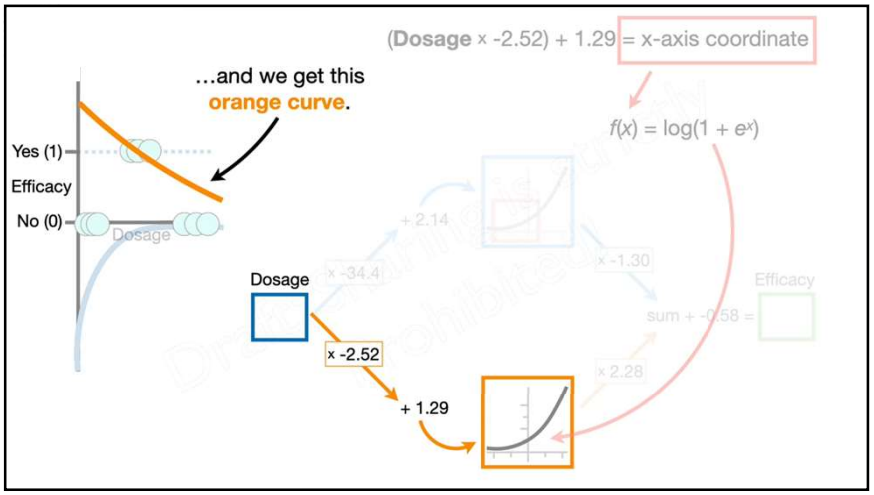

(0 x -2.52) + 1.29 = 1.29
...then the x-axis coordinate for the Activation Function is 1.29.
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0
x -2.52
+ 1.29
x
+ 2.14
x -34.4
x -1.30
sum + -0.58 =
Efficacy
x 2.28

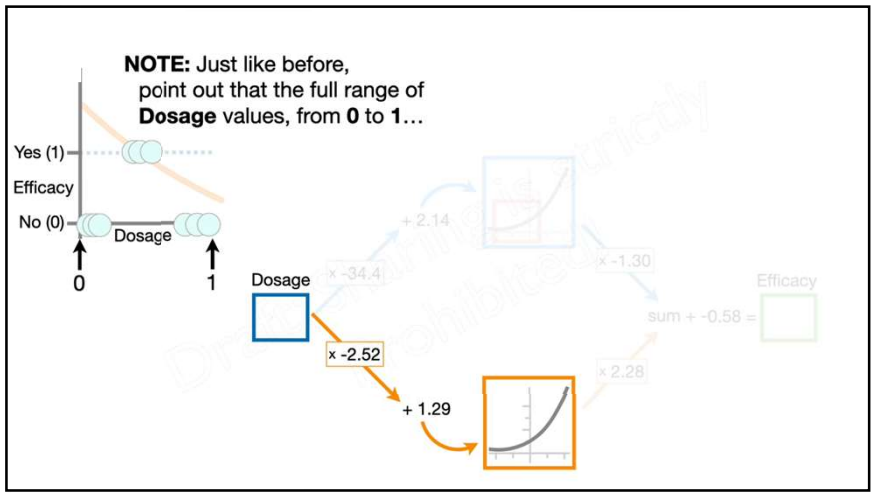

(0 x -2.52) + 1.29 = 1.29
Now we plug 1.29 into the Activation Function to get the corresponding y-axis value...
f(1.29) = log(1 + e^1.29)
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0
x -2.52
+ 1.29
x
+ 2.14
x -34.4
x -1.30
sum + -0.58 =
Efficacy
x 2.28

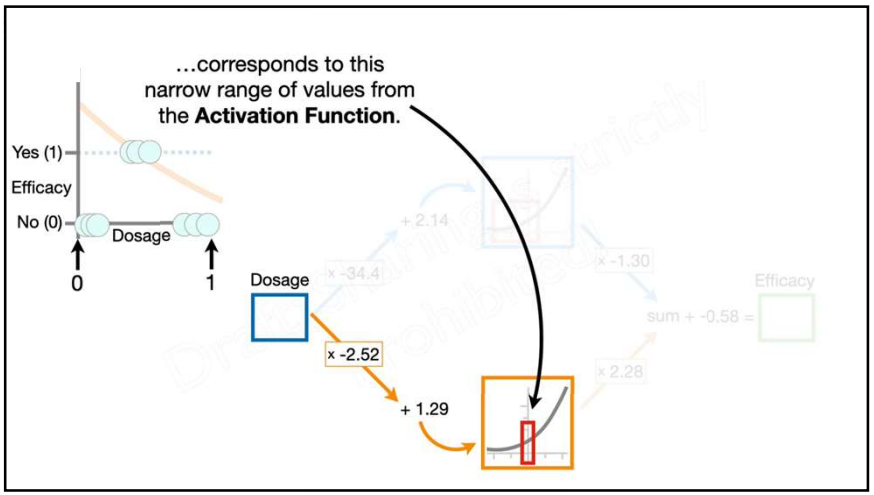

(0 x -2.52) + 1.29 = 1.29
...and get 1.53...
f(1.29) = log(1 + e^1.29) = 1.53
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0
x -2.52
+ 1.29
x
+ 2.14
x -34.4
x -1.30
sum + -0.58 =
Efficacy
x 2.28

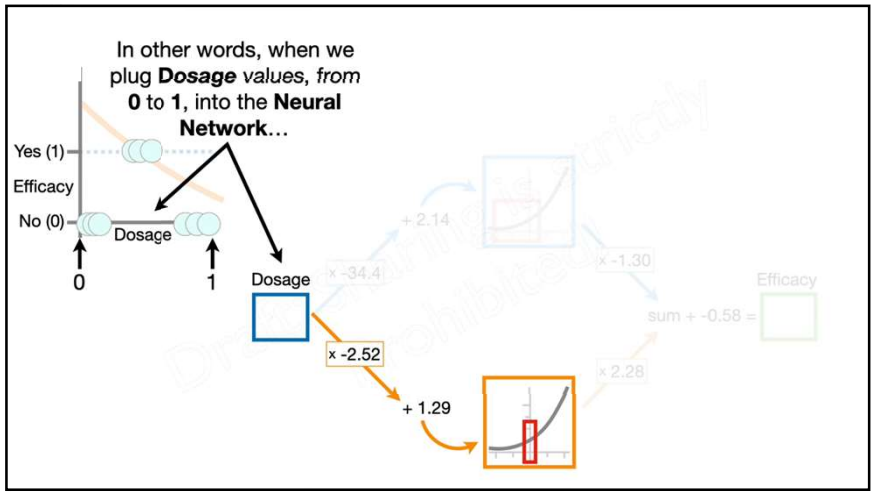

(0 x -2.52) + 1.29 = 1.29
...and that corresponds to this yellow dot.
f(1.29) = log(1 + e^1.29) = 1.53
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0
x -2.52
+ 1.29
x
+ 2.14
x -34.4
x -1.30
sum + -0.58 =
Efficacy
x 2.28

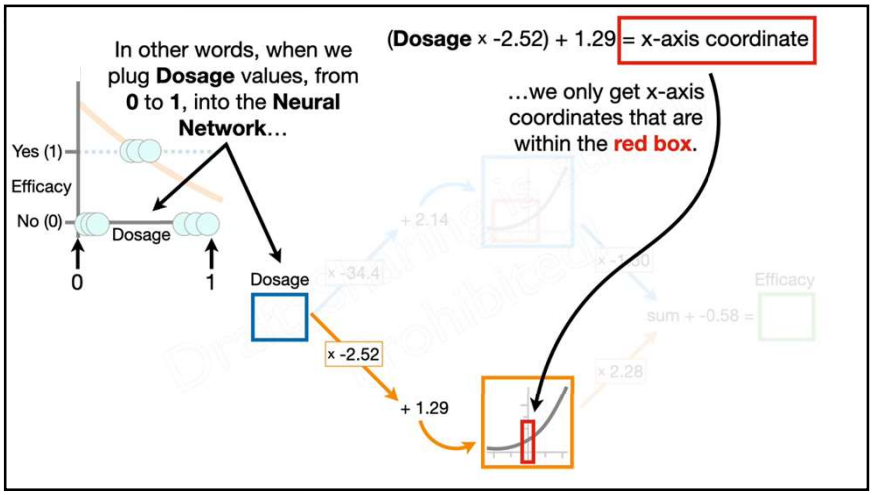

(Dosage x -2.52) + 1.29 = x-axis coordinate
Now we just plug in Dosage values from 0 to 1 to get the corresponding y-axis values…
f(x) = log(1 + e^x)
Yes (1)
Efficacy
No (0)
Dosage
Dosage
1
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

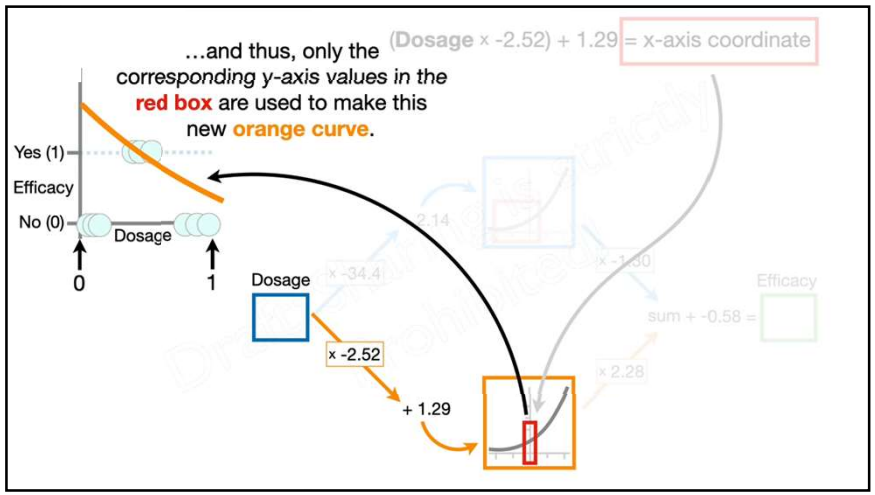

(Dosage x -2.52) + 1.29 = x-axis coordinate
…and we get this orange curve.
f(x) = log(1 + e^x)
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

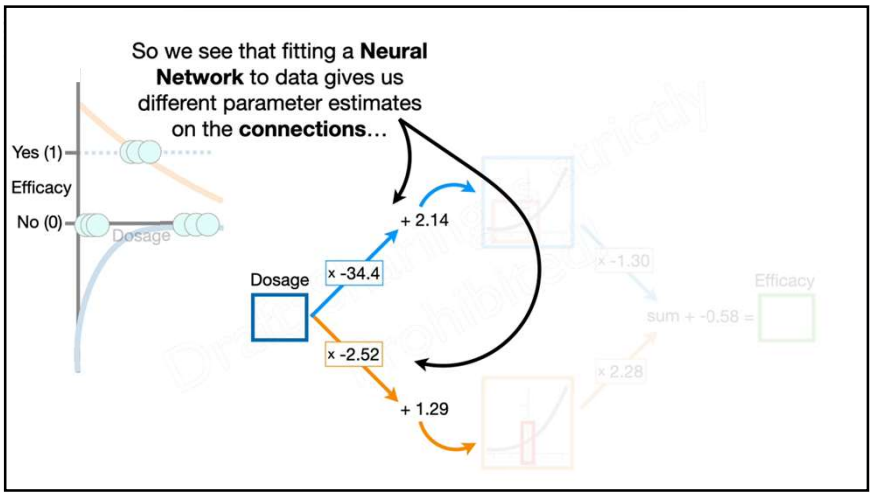

NOTE: Just like before, point out that the full range of Dosage values, from 0 to 1…
Yes (1)
Efficacy
No (0)
Dosage
0
1
Dosage
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

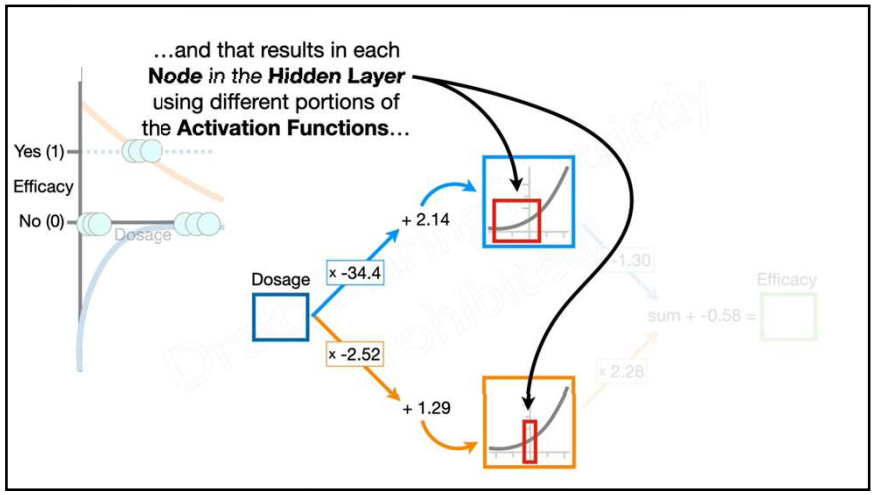

…corresponds to this narrow range of values from the Activation Function.
Yes (1)
Efficacy
No (0)
Dosage
0
1
Dosage
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

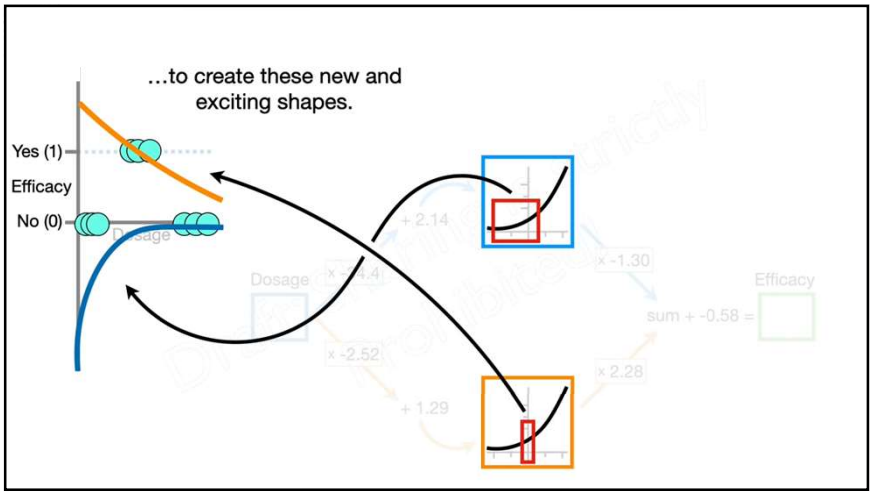

In other words, when we plug **Dosage** *values, from* **0** to **1**, into the **Neural Network**…
Yes (1)
Efficacy
No (0)
Dosage
0
1
Dosage
x -2.52
+ 1.29
x -34.4
+ 2.14
x -1.30
sum + -0.58 =
Efficacy
x 2.28

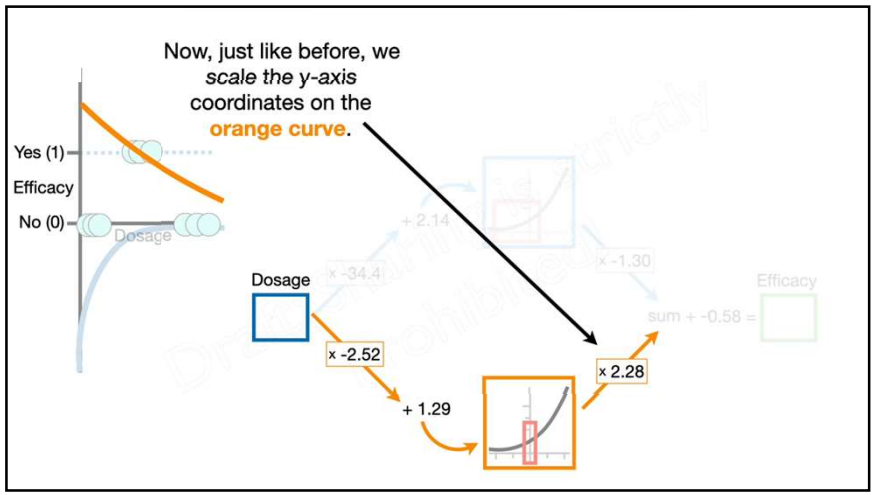

In other words, when we plug **Dosage** values, from **0** to **1**, into the **Neural Network**…
(**Dosage** x -2.52) + 1.29 = x-axis coordinate
…we only get x-axis coordinates that are within the **red box**.
Yes (1)
Efficacy
No (0)
Dosage
0
1
Dosage
x -2.52
+ 1.29
x -34.4
+ 2.14
x -1.30
sum + -0.58 =
Efficacy
x 2.28

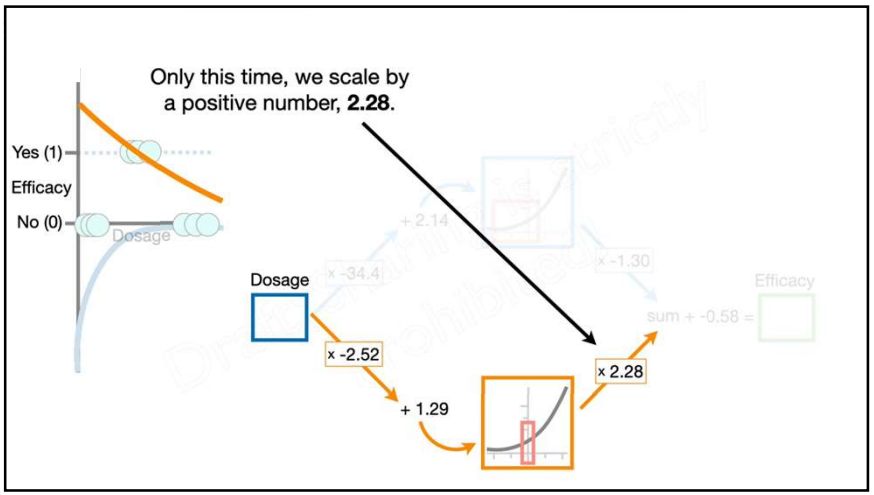

…and thus, only the *corresponding y-axis values in the* **red box** are used to make this new **orange curve**.
(**Dosage** x -2.52) + 1.29 = x-axis coordinate
Yes (1)
Efficacy
No (0)
Dosage
0
1
Dosage
x -2.52
+ 1.29
x -34.4
+ 2.14
x -1.30
sum + -0.58 =
Efficacy
x 2.28

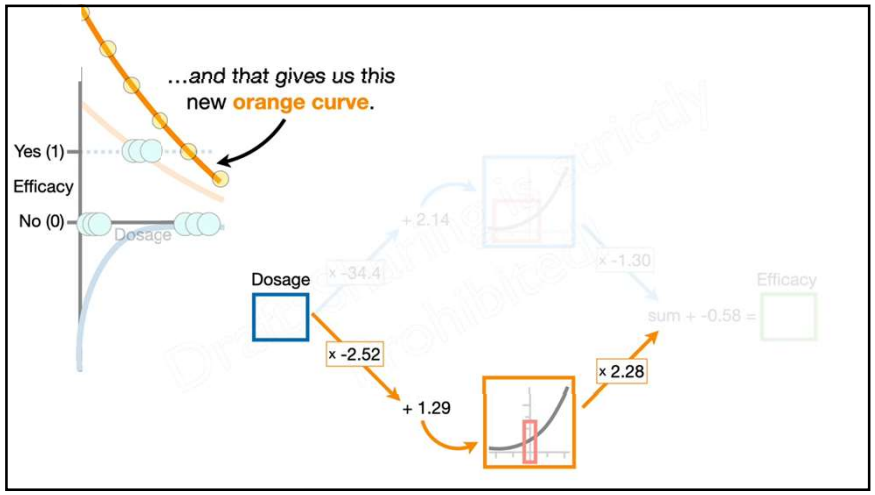

So we see that fitting a **Neural Network** to data gives us different parameter estimates on the **connections**…
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
x -2.52
+ 2.14
+ 1.29
x -1.30
sum + -0.58 =
Efficacy
x 2.28

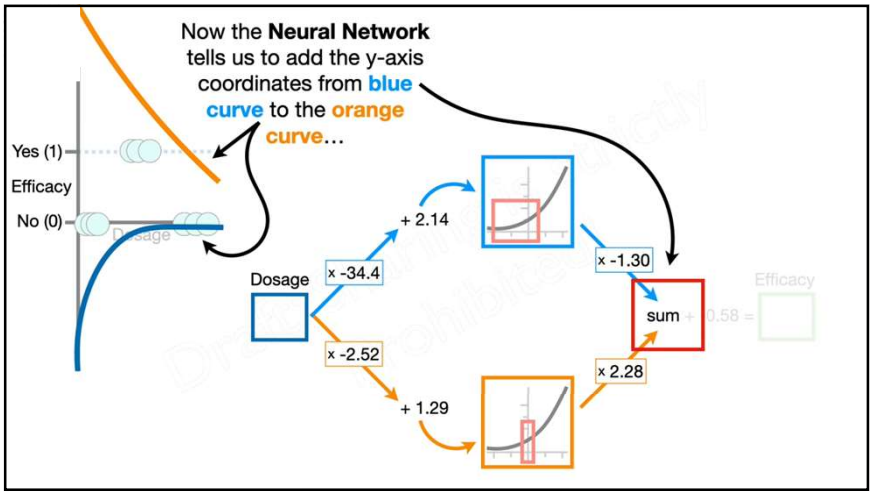

…and that results in each
Node *in the* ***Hidden Layer***
using different portions of
the **Activation Functions**…
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
+ 2.14
x -2.52
+ 1.29

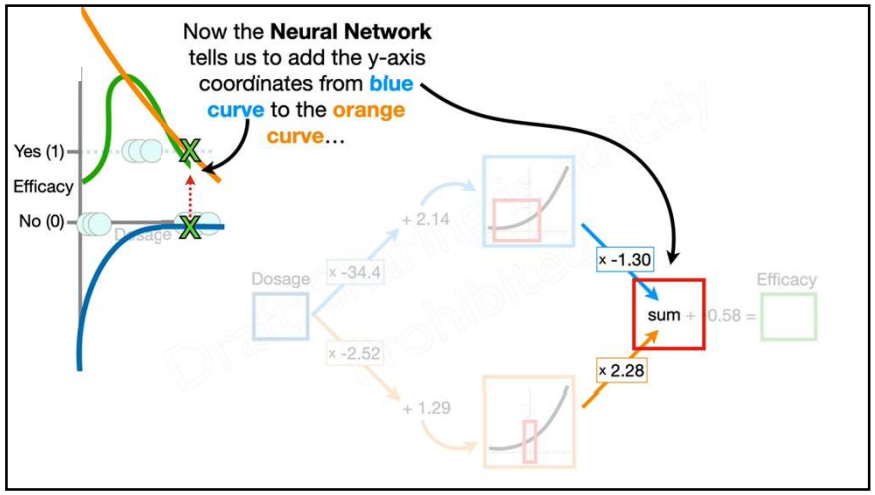

…to create these new and
exciting shapes.
Yes (1)
Efficacy
No (0)
Dosage

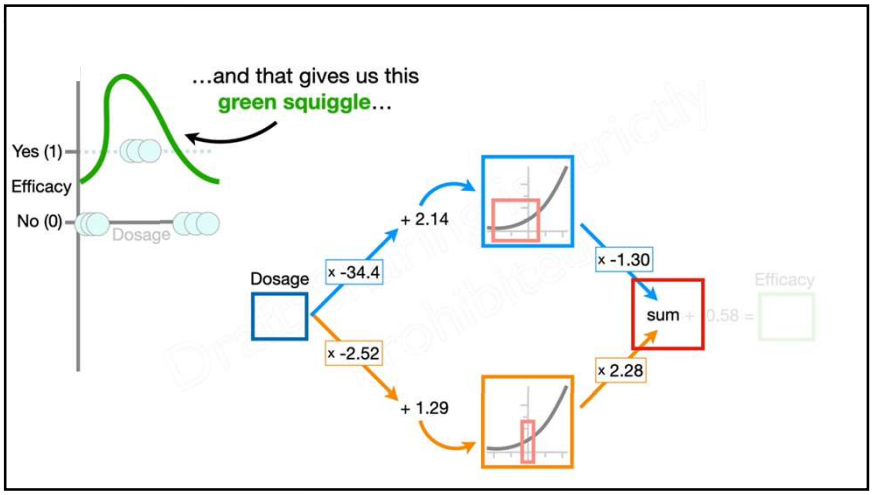

Now, just like before, we
scale the y-axis
coordinates on the
orange curve.
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -2.52
+ 1.29
x 2.28

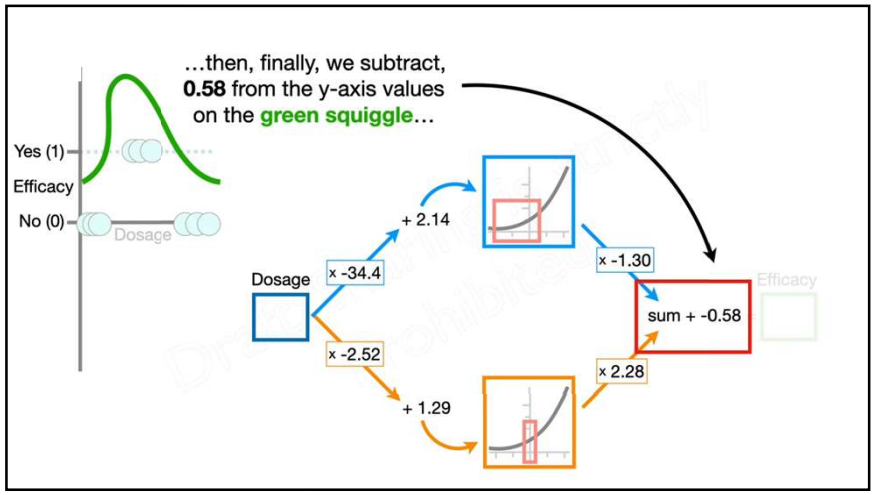

Only this time, we scale by
a positive number, **2.28**.
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -2.52
+ 1.29
x 2.28

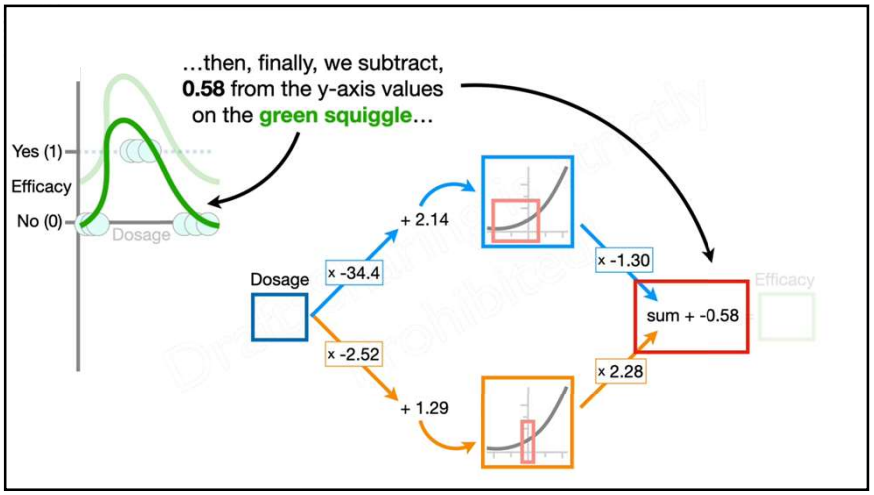

...and that gives us this
new orange curve.
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
+ 2.14
x -1.30
sum + -0.58 =
Efficacy
x -2.52
+ 1.29
x 2.28

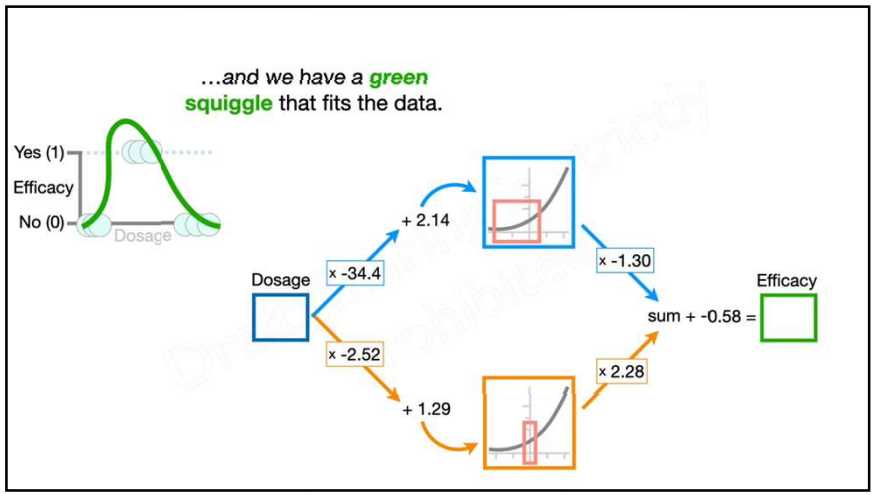

Now the Neural Network
tells us to add the y-axis
coordinates from blue
curve to the orange
curve...
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
+ 2.14
x -1.30
sum
Efficacy
x -2.52
+ 1.29
x 2.28

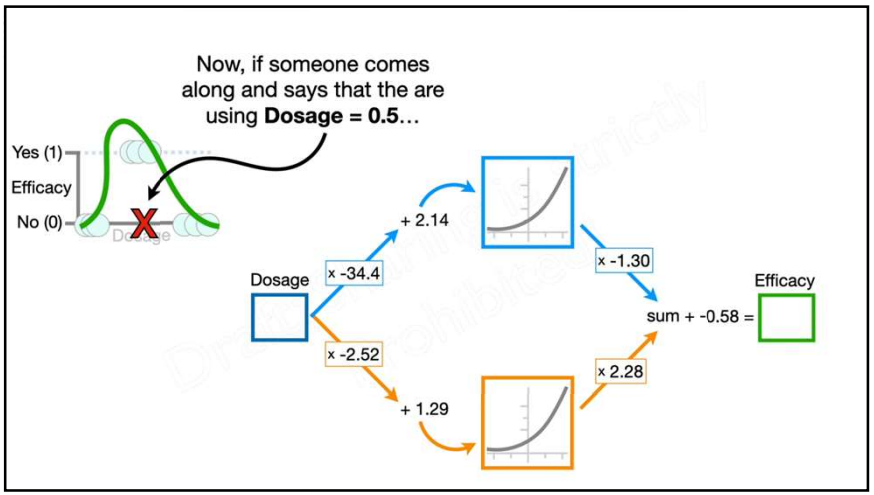

Now the Neural Network
tells us to add the y-axis
coordinates from blue
curve to the orange
curve...
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
+ 2.14
x -1.30
sum + -0.58 =
Efficacy
x -2.52
+ 1.29
x 2.28

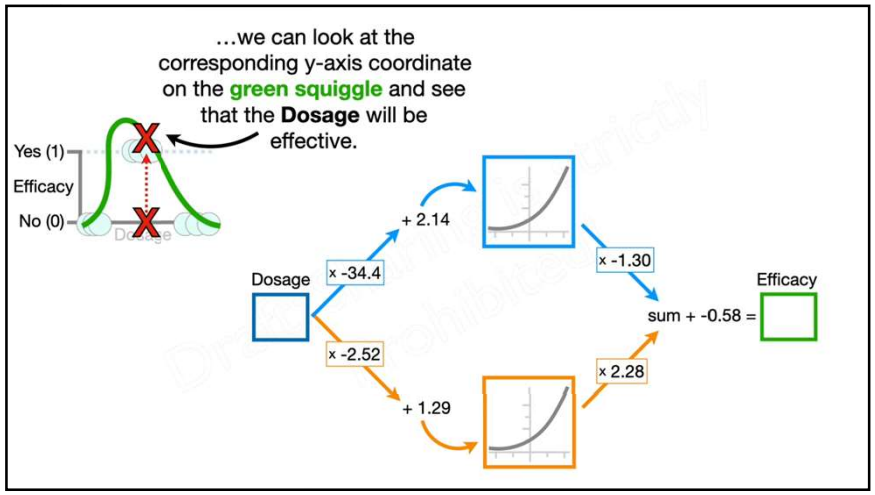

...and that gives us this
green squiggle...
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
+ 2.14
x -1.30
sum
Efficacy
x -2.52
+ 1.29
x 2.28

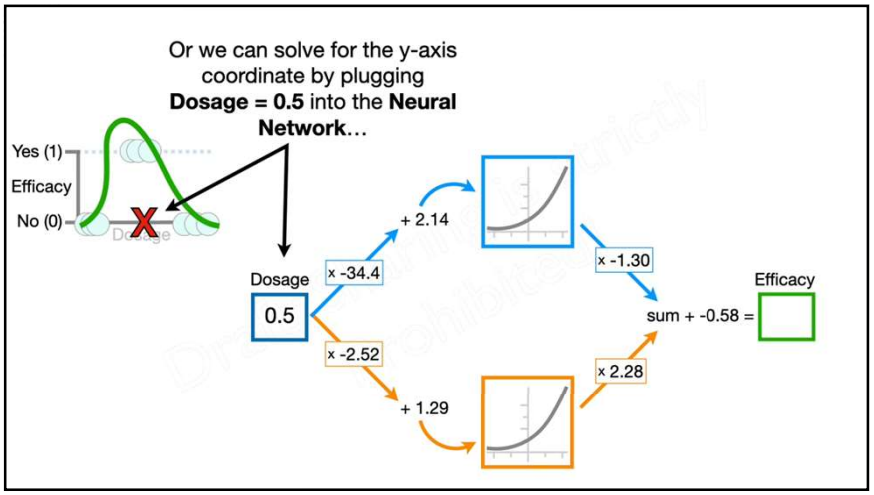

…then, finally, we subtract,
0.58 from the y-axis values
on the green squiggle…
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
+ 2.14
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58
Efficacy

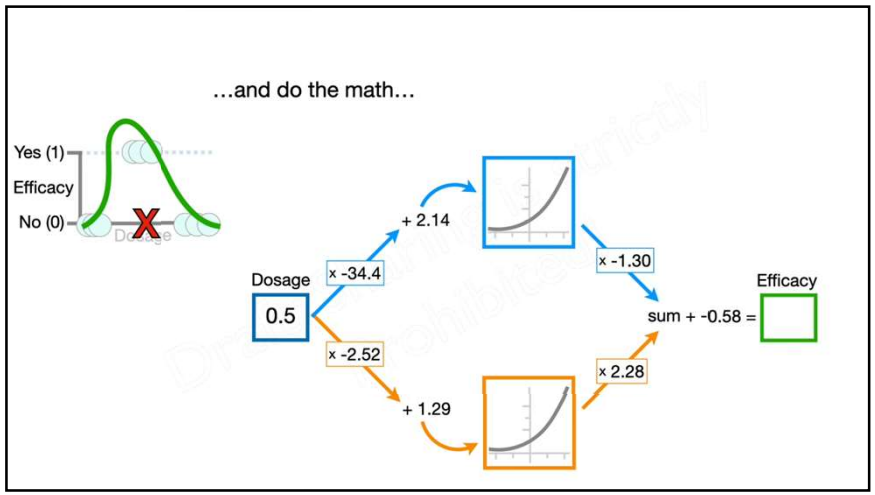

…then, finally, we subtract,
0.58 from the y-axis values
on the green squiggle…
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
+ 2.14
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58
Efficacy

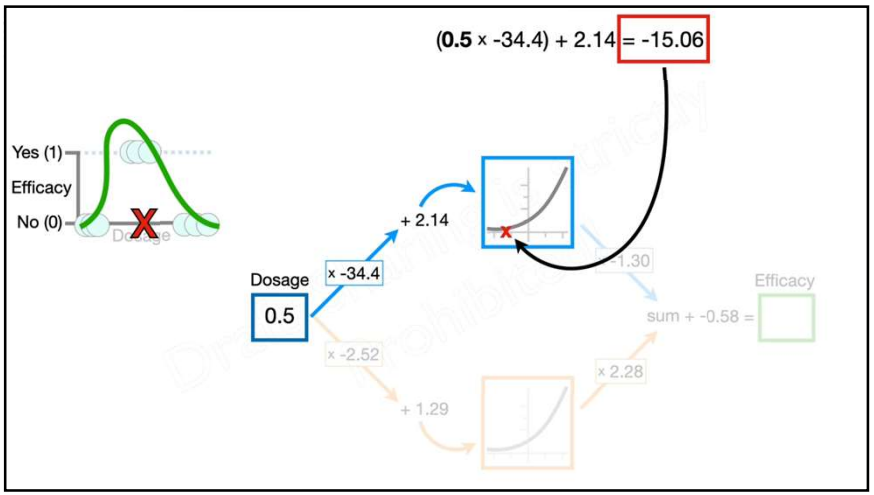

…and we have a green
squiggle that fits the data.
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
+ 2.14
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58 =
Efficacy

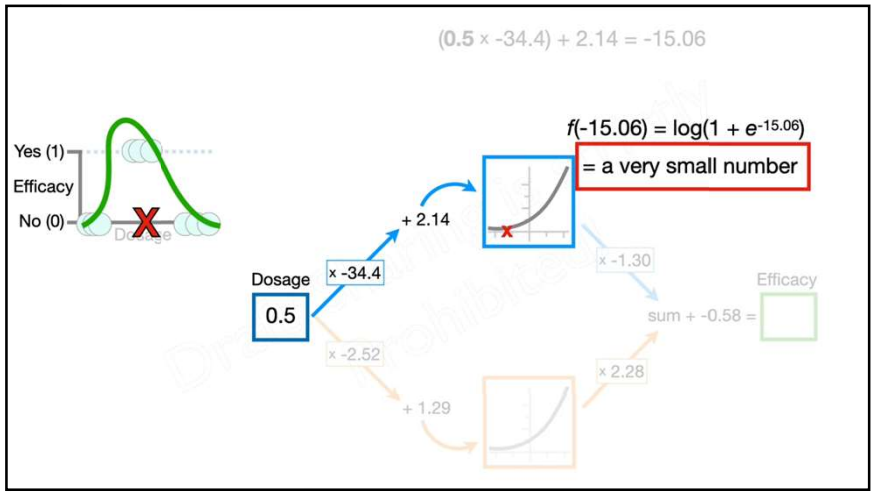

Now, if someone comes
along and says that the are
using Dosage = 0.5…
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
+ 2.14
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58 =
Efficacy

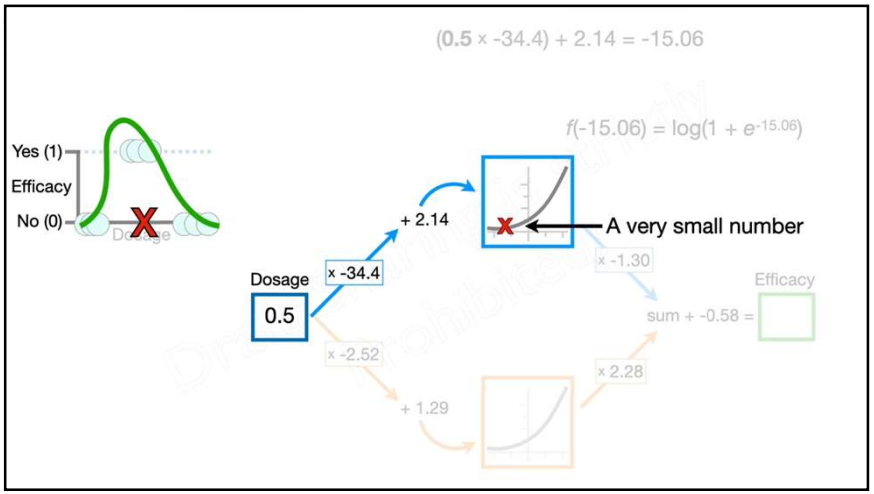

...we can look at the corresponding y-axis coordinate on the green squiggle and see that the Dosage will be effective.
Yes (1)
Efficacy
No (0)
Dosage
+ 2.14
x -34.4
x -1.30
Efficacy
sum + -0.58 =
x -2.52
x 2.28
+ 1.29

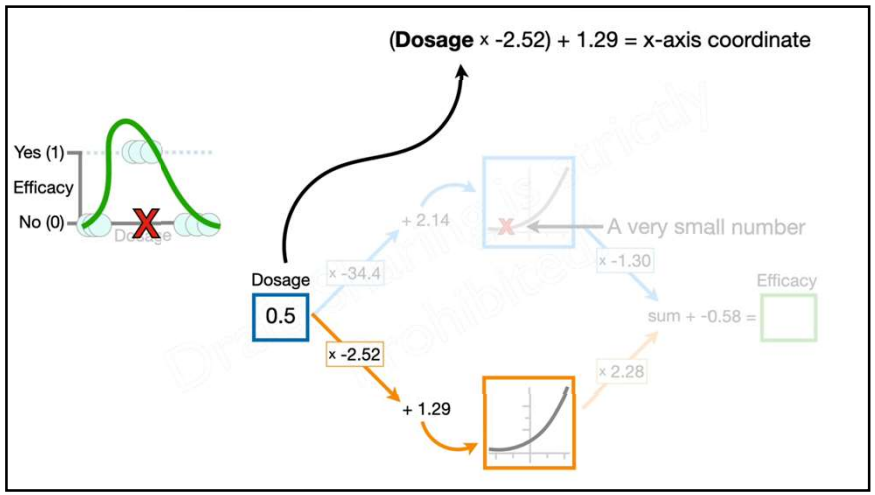

Or we can solve for the y-axis coordinate by plugging Dosage = 0.5 into the Neural Network...
Yes (1)
Efficacy
No (0)
Dosage
0.5
+ 2.14
x -34.4
x -1.30
Efficacy
sum + -0.58 =
x -2.52
x 2.28
+ 1.29

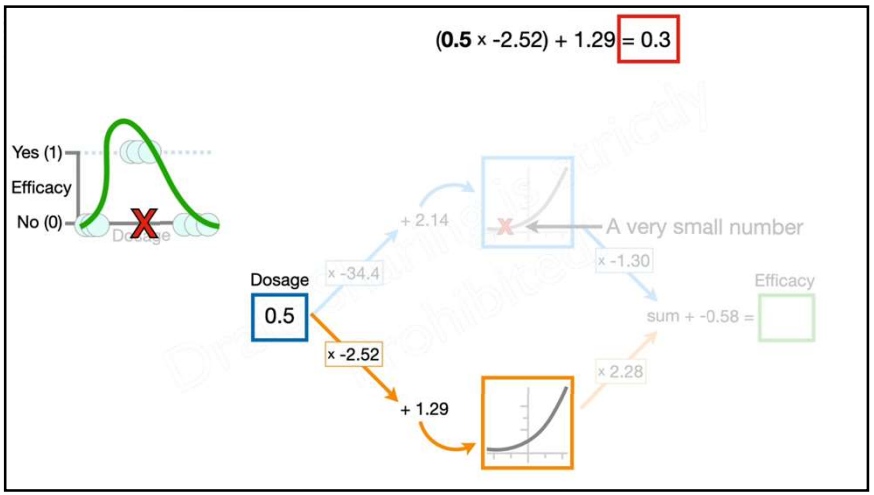

...and do the math...
Yes (1)
Efficacy
No (0)
Dosage
0.5
+ 2.14
x -34.4
x -1.30
Efficacy
sum + -0.58 =
x -2.52
x 2.28
+ 1.29

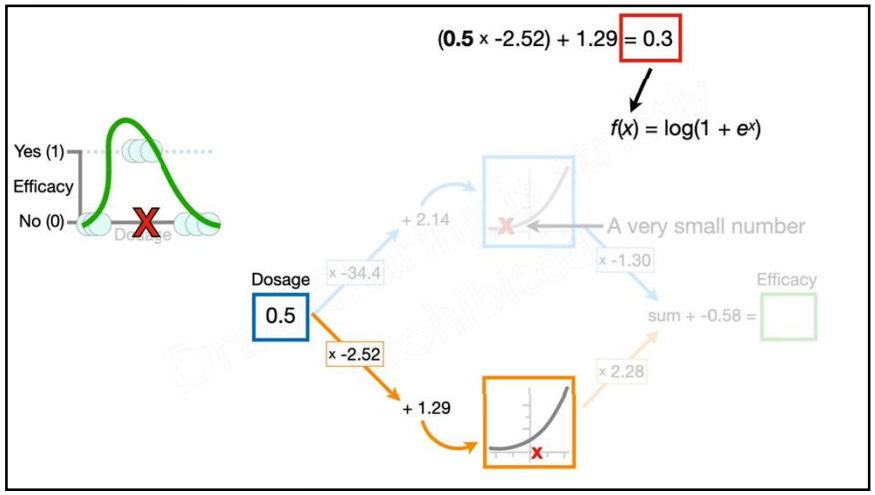

(0.5 x -34.4) + 2.14 = -15.06
Yes (1)
Efficacy
No (0)
Dosage
0.5
+ 2.14
x -34.4
Efficacy
sum + -0.58 =
x -2.52
x 2.28
+ 1.29

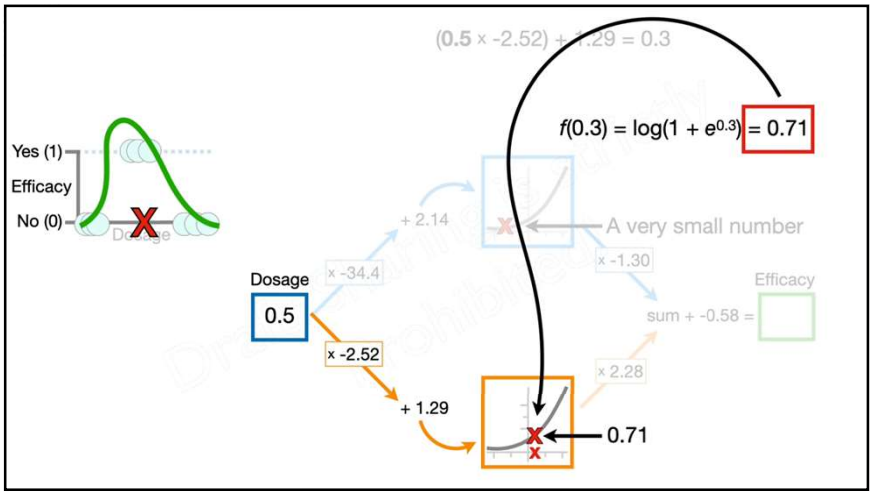

(0.5 x -34.4) + 2.14 = -15.06
Yes (1)
Efficacy
No (0)
Dosage
f(-15.06) = log(1 + e^-15.06)
= a very small number
+ 2.14
Dosage
0.5
x -34.4
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58 =
Efficacy

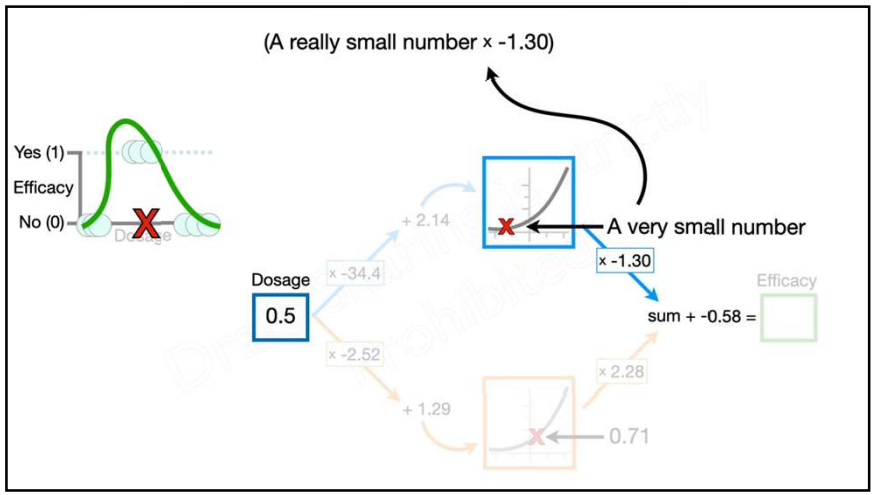

(0.5 x -34.4) + 2.14 = -15.06
Yes (1)
Efficacy
No (0)
Dosage
f(-15.06) = log(1 + e^-15.06)
A very small number
+ 2.14
Dosage
0.5
x -34.4
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58 =
Efficacy

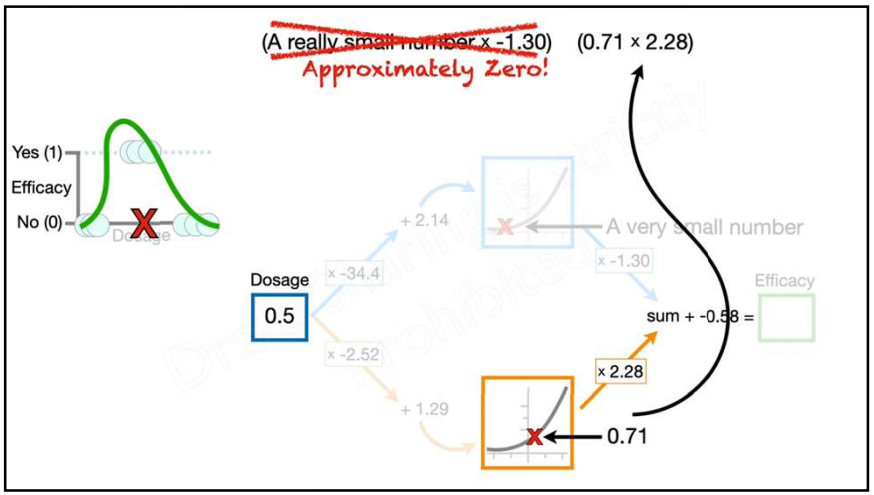

(Dosage x -2.52) + 1.29 = x-axis coordinate
Yes (1)
Efficacy
No (0)
Dosage
A very small number
+ 2.14
Dosage
0.5
x -34.4
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58 =
Efficacy

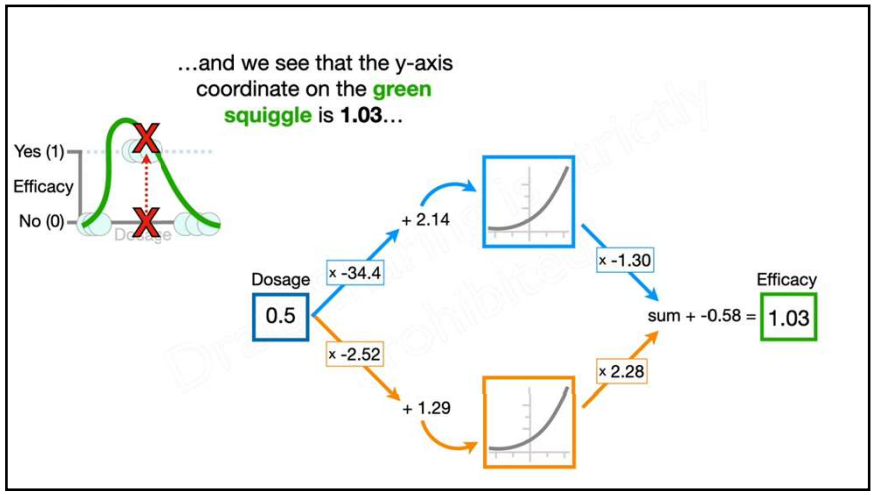

(0.5 x -2.52) + 1.29 = 0.3
Yes (1)
Efficacy
No (0)
Dosage
A very small number
+ 2.14
Dosage
0.5
x -34.4
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58 =
Efficacy

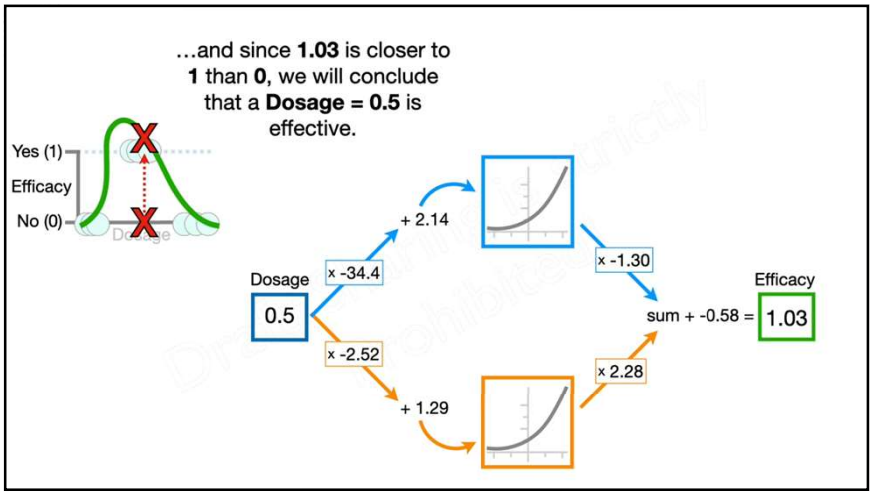

(0.5 x -2.52) + 1.29 = 0.3
f(x) = log(1 + e^x)
Yes (1)
Efficacy
No (0)
Dosage
A very small number
Dosage
0.5
x -34.4
+ 2.14
x -1.30
Efficacy
sum + -0.58 =
x -2.52
x 2.28
+ 1.29

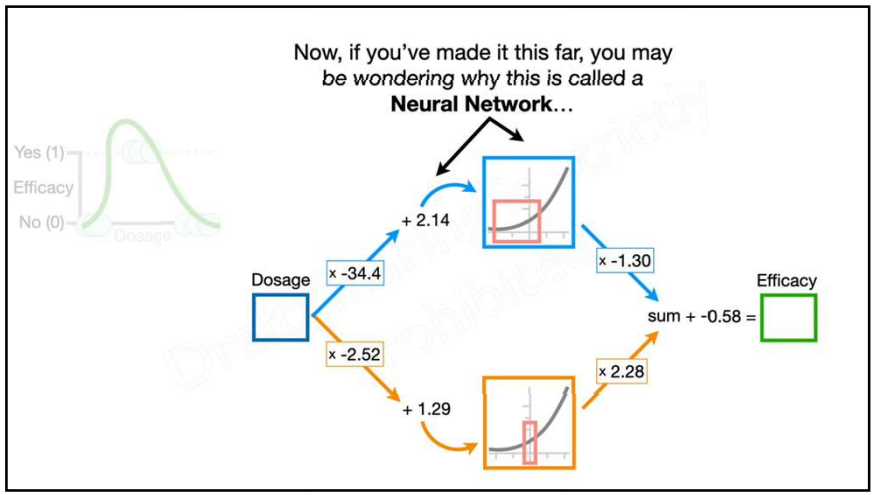

(0.5 x -2.52) + 1.29 = 0.3
f(0.3) = log(1 + e^0.3) = 0.71
Yes (1)
Efficacy
No (0)
Dosage
A very small number
Dosage
0.5
x -34.4
+ 2.14
x -1.30
Efficacy
sum + -0.58 =
x -2.52
x 2.28
+ 1.29
0.71

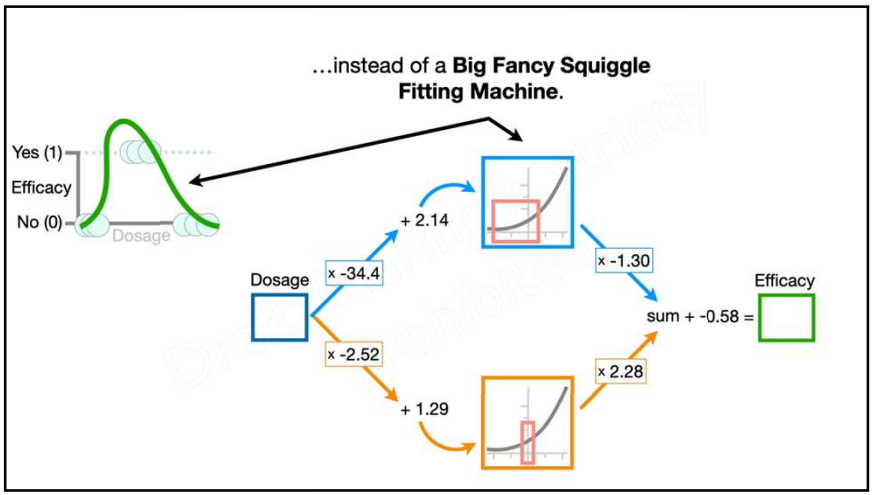

(A really small number x -1.30)
Yes (1)
Efficacy
No (0)
Dosage
A very small number
Dosage
0.5
x -34.4
+ 2.14
x -1.30
Efficacy
sum + -0.58 =
x -2.52
x 2.28
+ 1.29
0.71

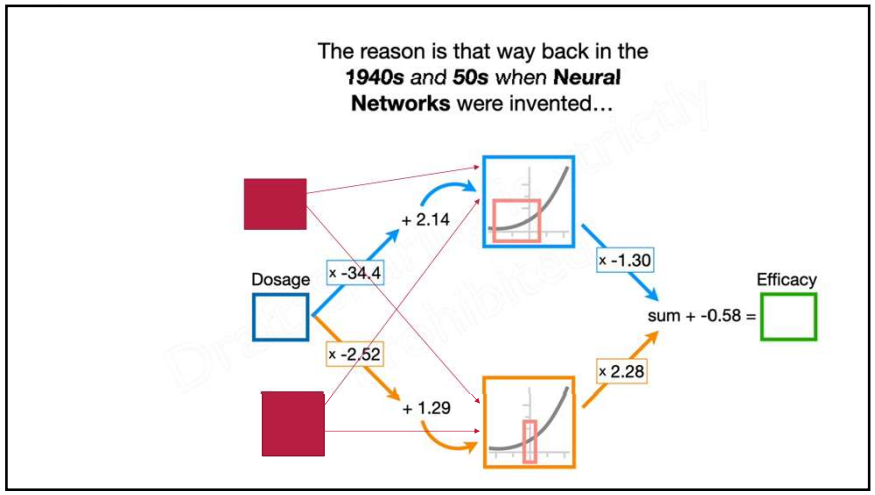

(A really small number x -1.30)
Approximately Zero!
(0.71 x 2.28)
Yes (1)
Efficacy
No (0)
Dosage
A very small number
Dosage
0.5
x -34.4
+ 2.14
x -1.30
Efficacy
sum + -0.58 =
x -2.52
x 2.28
+ 1.29
0.71

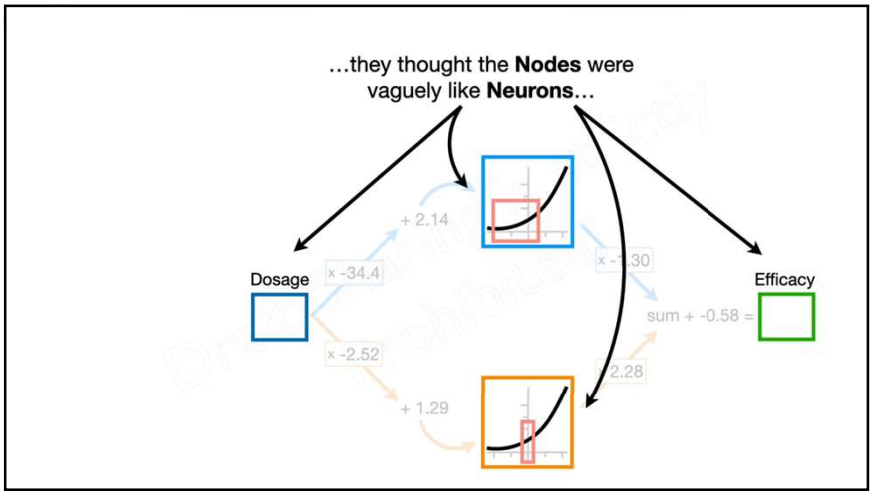

...and we see that the y-axis coordinate on the green squiggle is 1.03...
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0.5
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy
1.03

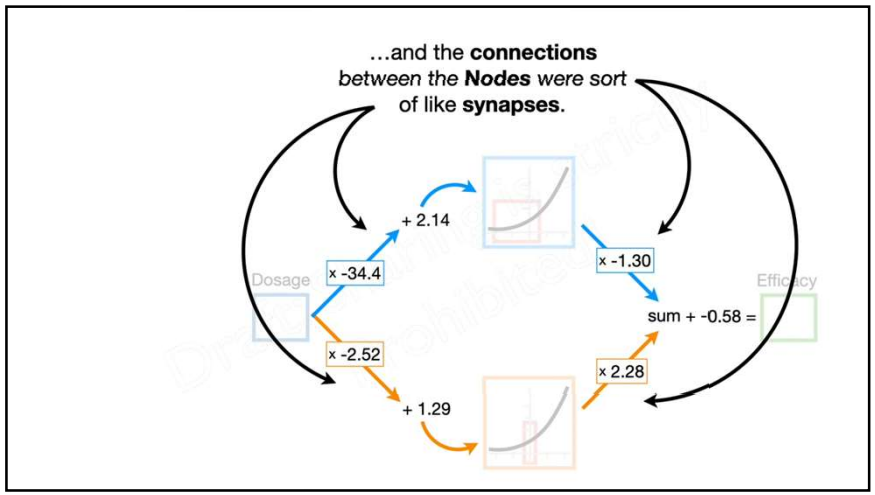

...and since 1.03 is closer to 1 than 0, we will conclude that a Dosage = 0.5 is effective.
Yes (1)
Efficacy
No (0)
Dosage
Dosage
0.5
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy
1.03

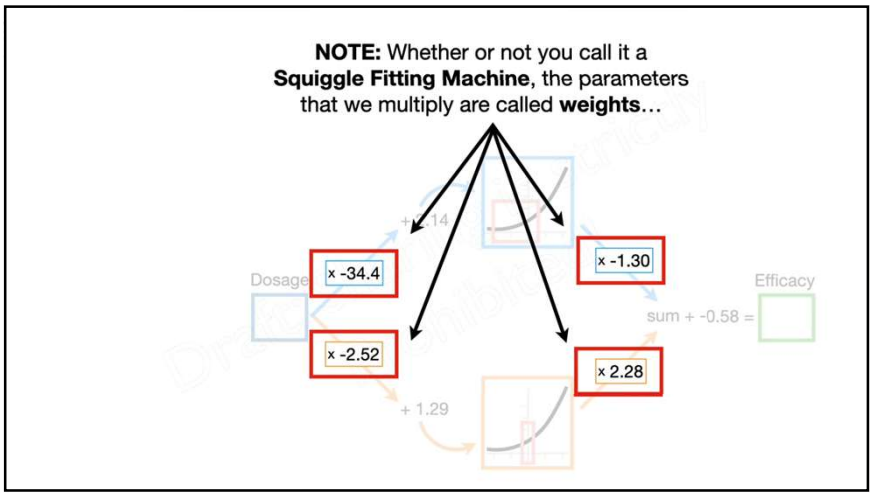

Now, if you've made it this far, you may be wondering why this is called a Neural Network...
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

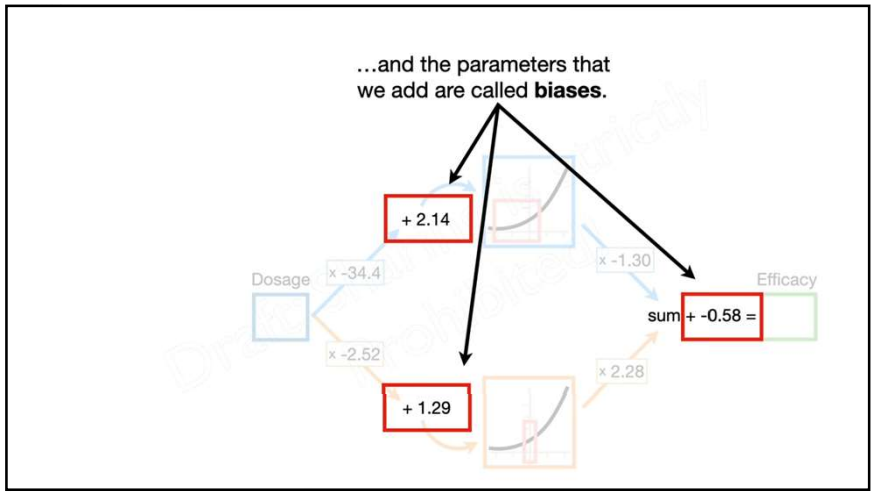

...instead of a Big Fancy Squiggle Fitting Machine.
Yes (1)
Efficacy
No (0)
Dosage
Dosage
x -34.4
+ 2.14
x -1.30
x -2.52
+ 1.29
x 2.28
sum + -0.58 =
Efficacy

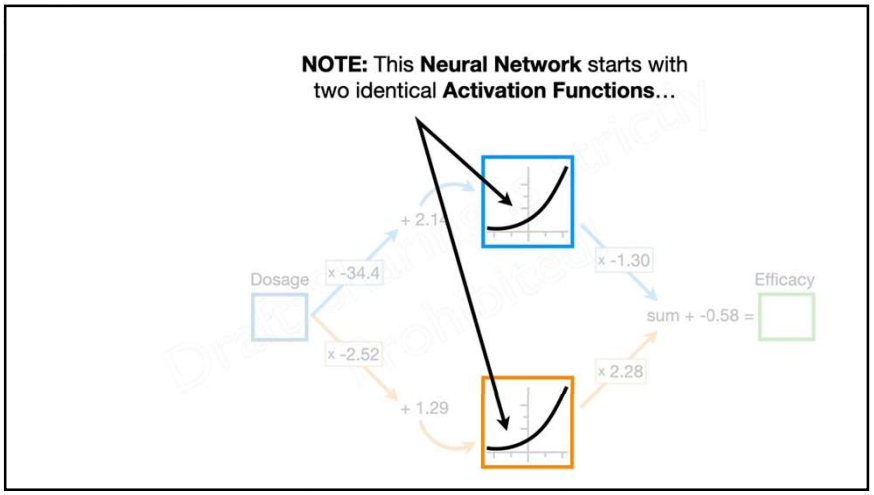

The reason is that way back in the
1940s and 50s when Neural
Networks were invented…
Dosage
x -34.4
+ 2.14
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58 =
Efficacy

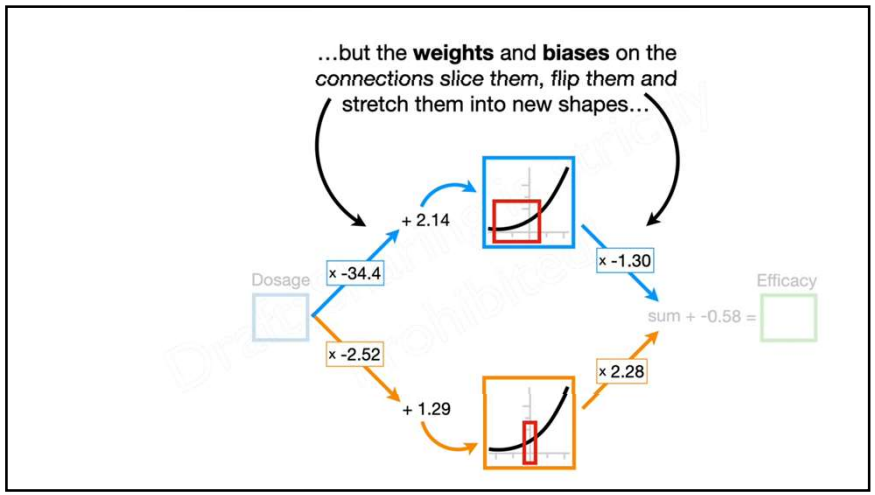

…they thought the Nodes were
vaguely like Neurons…
Dosage
x -34.4
+ 2.14
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58 =
Efficacy

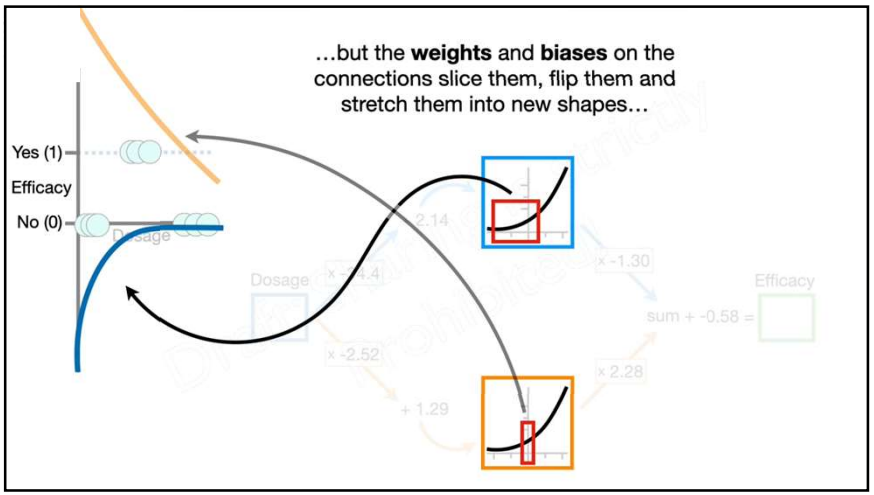

…and the connections
between the Nodes were sort
of like synapses.
Dosage
x -34.4
+ 2.14
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58 =
Efficacy

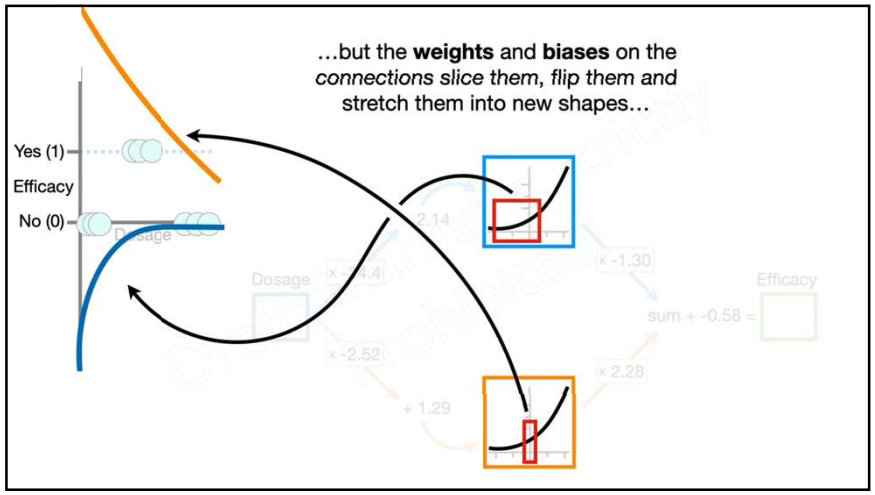

NOTE: Whether or not you call it a
Squiggle Fitting Machine, the parameters
that we multiply are called weights…
Dosage
x -34.4
+ 2.14
x -2.52
+ 1.29
x -1.30
x 2.28
sum + -0.58 =
Efficacy

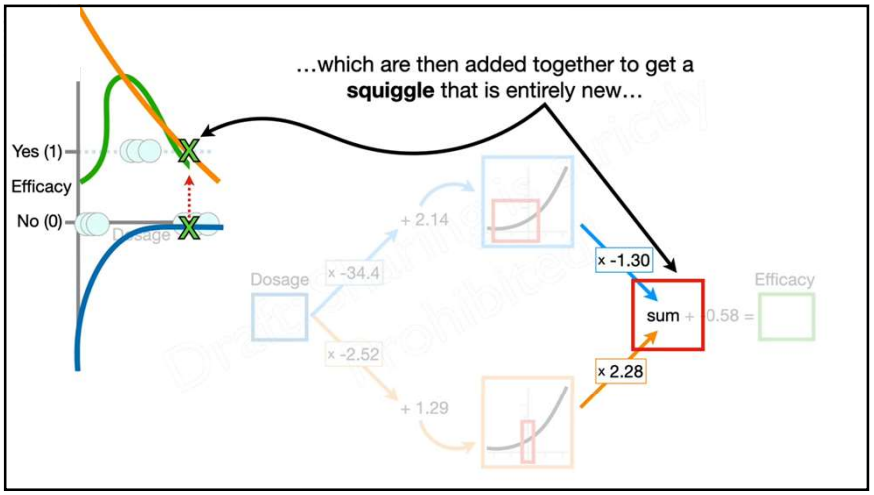

...and the parameters that
we add are called **biases**.
+ 2.14
Dosage
x -34.4
x -2.52
+ 1.29
x -1.30
x 2.28
Efficacy
sum + -0.58 =

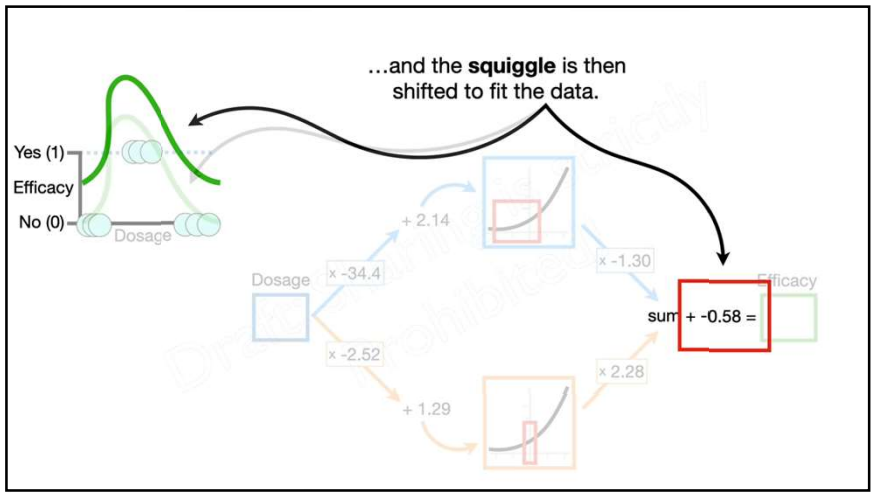

NOTE: This **Neural Network** starts with
two identical **Activation Functions**...
Dosage
x -34.4
x -2.52
+ 1.29
x -1.30
x 2.28
Efficacy
sum + -0.58 =

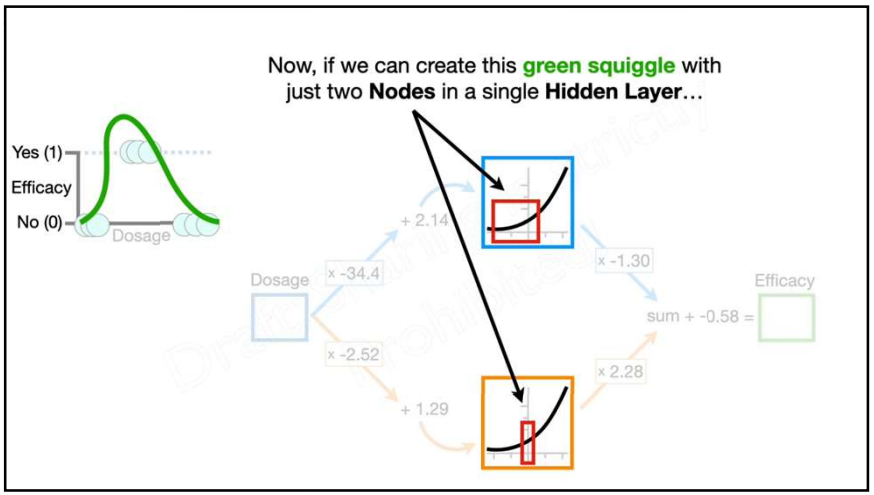

...but the **weights** and **biases** on the
connections slice them, flip them and
stretch them into new shapes...
+ 2.14
Dosage
x -34.4
x -2.52
+ 1.29
x -1.30
x 2.28
Efficacy
sum + -0.58 =


...but the **weights** and **biases** on the
connections slice them, flip them and
stretch them into new shapes...
Yes (1)
Efficacy
No (0)
Dosage

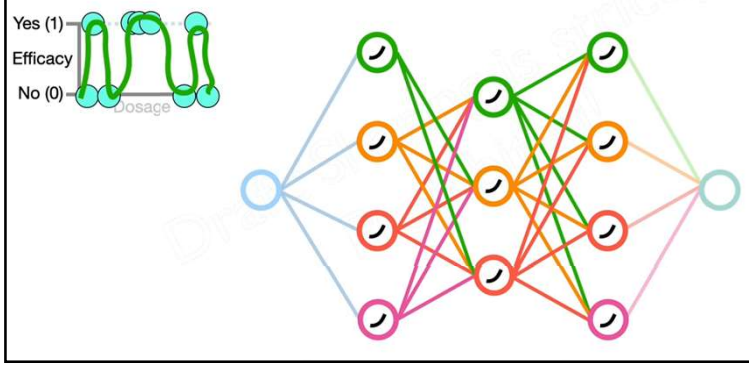

...but the **weights** and **biases** on the *connections slice them, flip them and* stretch them into new shapes...
Yes (1)
Efficacy
No (0)
Dosage
+ 2.14
x -34.4
x -1.30
Efficacy
sum + -0.58 =
x -2.52
x 2.28
+ 1.29


...which are then added together to get a **squiggle** that is entirely new...
Yes (1)
Efficacy
No (0)
Dosage
+ 2.14
x -34.4
x -1.30
Efficacy
sum + -0.58 =
x -2.52
x 2.28
+ 1.29

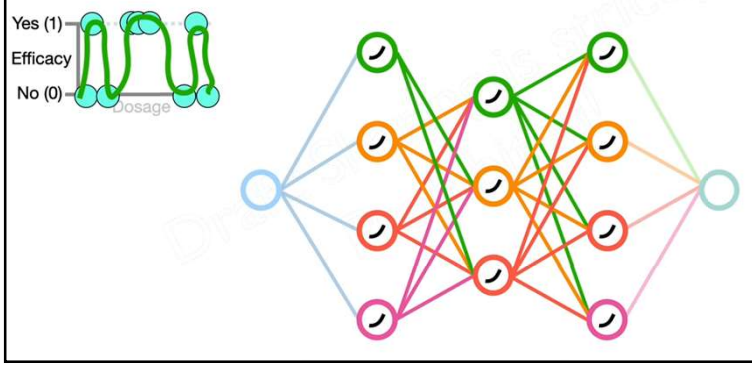

...and the **squiggle** is then shifted to fit the data.
Yes (1)
Efficacy
No (0)
Dosage
+ 2.14
x -34.4
x -1.30
Efficacy
sum + -0.58 =
x -2.52
x 2.28
+ 1.29


Now, if we can create this **green squiggle** with just two **Nodes** in a single **Hidden Layer**...
Yes (1)
Efficacy
No (0)
Dosage
+ 2.14
x -34.4
x -1.30
Efficacy
sum + -0.58 =
x -2.52
x 2.28
+ 1.29

…just imagine what types of **green squiggles** *we could fit with more* ***Hidden Layers*** *and* more **Nodes** in each **Hidden Layer**.



In theory, **Neural Networks** can fit a **green squiggle** to just about any dataset, no matter how complicated!



THANK YOU!