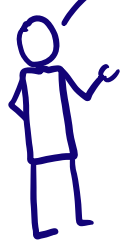
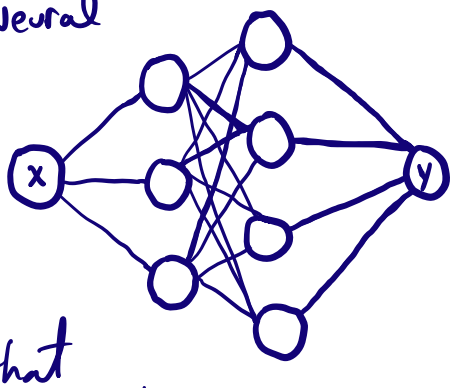
A graphical interpretation of neural networks



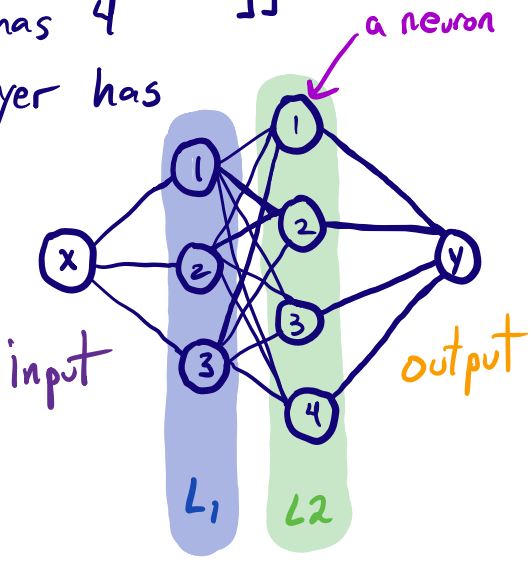
Here is a pretty typical depiction of a neural network.



Its not actually that useful I think, at least until you know what it means. Let's break it down.

We would say this NN has two hidden layers.

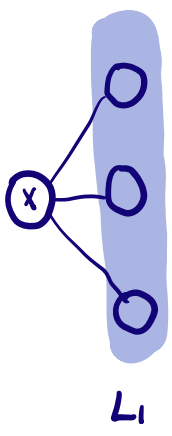
The first layer has 3 neurons and the second layer has 4 neurons. The input layer has on



um. Professor, isn't there some math? How are we supposed to use that?

Yes there is! one step at a time. You can think about each layer as a nonlinear, dimensional transform.

Let's start with the first layer.



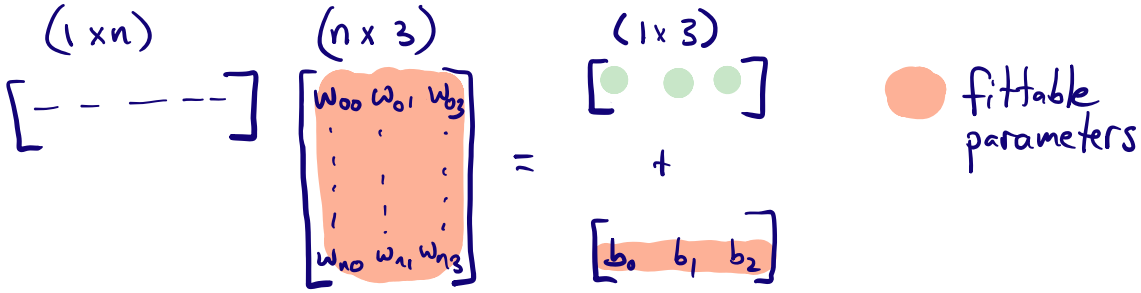
X will usually be a row vector with a shape (1 x n)

This figure means we matrix multiply X by an array with a shape (n x 3) known as the weights for this layer. Then add an array of shape (1, 3) known as the biases. And finally do an elementwise "activation" with a nonlinear function.

so, we have an input of shape $(1 \times n)$ times an array of weights with shape $(n \times 3)$.

$$(1 \times n) @ (n \times 3) = (1 \times 3)$$

$$+ \ (1 \times 3) = (1 \times 3)$$



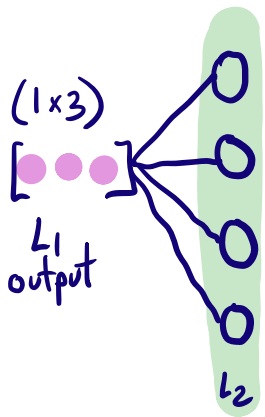
$$= \begin{bmatrix} \bullet & \bullet & \bullet \end{bmatrix}$$

see how we made a nonlinear, dimensional change? The input was $(1 \times n)$ and now it is 1×3. so if $n=1$, we have increased the dimensionality, and if $n=5$, we decreased the dimensionality

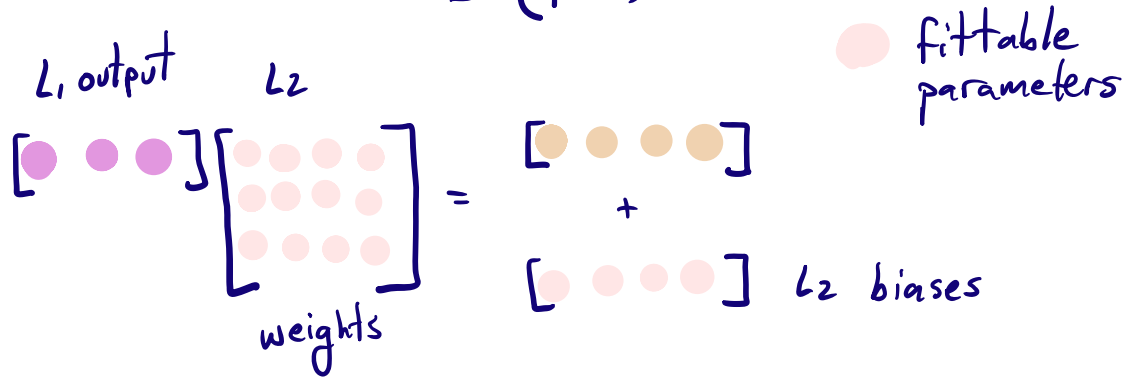
$$\text{act}\left(\begin{bmatrix} \bullet & \bullet & \bullet \end{bmatrix}\right) = \begin{bmatrix} \bullet & \bullet & \bullet \end{bmatrix}$$

output of Layer 1

- Next, the output of Layer 1 is the input to layer 2. So, this means to multiply the L_1 output by an array of weights with shape $(3, 4)$, then add a bias array with shape $(1, 4)$, and activate the result elementwise with a nonlinear function.



$$(1\times3) \mathbin{@} (3\times4) = (1\times4) + (1\times4) = (1\times4)$$



$$= \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \end{bmatrix}$$

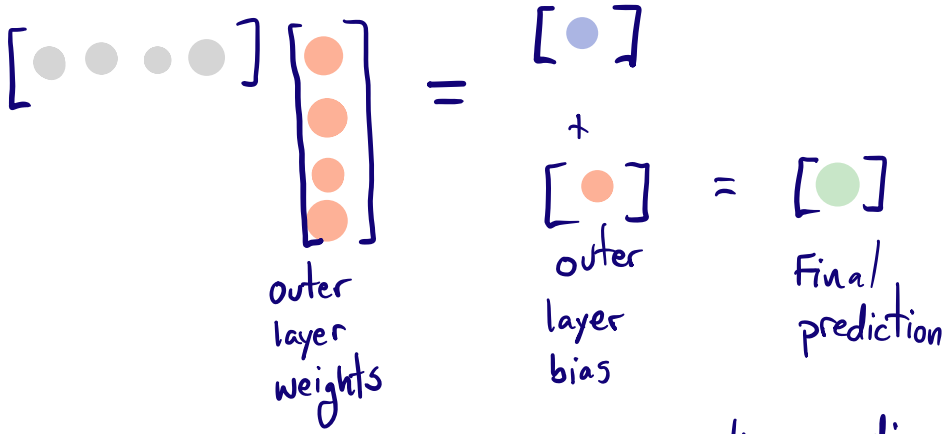
$$\text{act}\left(\begin{bmatrix} \bullet & \bullet & \bullet & \bullet \end{bmatrix}\right) = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \end{bmatrix}$$

L_2 output

See, L_2 does a nonlinear, dimensional transform on the output of L_1! We are almost to the end.

If the neural network is used for regression (as opposed to say classification) the output layer is usually just a linear combination of the output from the last hidden layer, so that we predict the right number of things.

So if we are predicting one thing, and the output of the last layer is (1×8) we simply multiply by an array of weights with shape (8×1) and add a single bias to it.

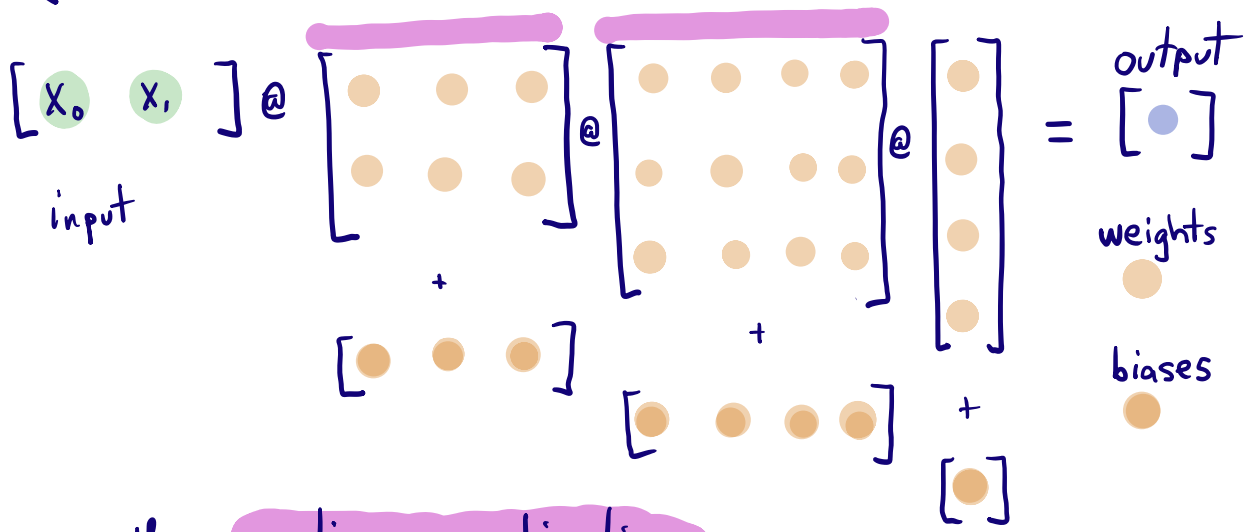


The final layers serves as a linear dimensional transform to the final result

So, all together, say the input has 2 elements and the output is one number. In other words, $y = f(x_0, x_1)$ and we have 2 hidden layers, one with 3 neurons, and one with 4 neurons. That looks like this

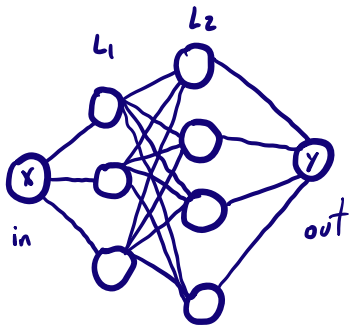
L_1 L_2 output layer

$(1 \times 2) \; @ \; (2 \times 3) \; @ \; (3 \times 4) \; @ \; (4 \times 1) = (1 \times 1)$



the nonlinear activations are done on L_1 and L_2, and not on the output layer. This model has 30 fittable parameters (the weights and biases).

In graph form this model looks like:



To use this model, you have to fit it to data. That means use an optimization algorithm to find the set of weights and biases that minimize the errors between the data and model predictions.

AI Jockeys like to call this "training", and say things like the models learn things about the data. Fff...

You have to choose how many neurons, how many layers, and what activation function. These are called hyperparameters. They all affect the flexibility of the model to fit your data.