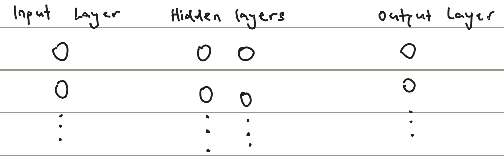
Structure of Neural Network

- Feedforward neural network ≡ Multilayer Perceptron
- Nodes ← Building blocks of network. (holds a number, typically b/w 0 and 1)

0.4 ← Node aka neuron

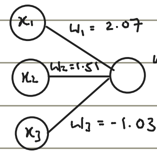
Activation

Highest Activation will be the output



- Activation (value) in 1 layer affects activation in subsequent layers.

Each connection to a node (neuron) is assigned a weight



Activation $= w_1 x_1 + w_2 x_2 + w_3 x_3$

∴ Resultant can be any value.

aka logistic curve

Sigmoid Fn $\sigma(x) = \frac{1}{1 + e^{-x}}$

└ squishes the Activation to between 0 and 1

Activation $= \sigma(\underbrace{w_1 x_1 + w_2 x_2 + w_3 x_3}_{\text{weighted sum}})$.

Bias

- Additional parameter to shift Activation up / down.
- Help model learn more complex problems.
- Specify threshold for weighted sum to be activated.

Activation $= \sigma(\text{weighted sum} + \beta)$ ← Bias.

Learning

- Finding the right weights & Biases

Notational Representation

$$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ \vdots & & & \\ w_{k,0} & \cdots & & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}\right) = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \end{bmatrix}$$

Activation in 1 layer

← Activation for next layer

← Bias for next layer nodes.

Activation fn for each component

└ Weights for each activation. corresponds to No. of nodes in next layer

* Deep learning has strong emphasis on linear algebra

Compact notation

$\sigma(W \cdot A + B)$ ← Function within each node.

W: $k \times n$, A: $n \times 1$, B: $n \times 1$

* Each node can be seen as a fn

* ReLU more favoured over sigmoid

↳ computationally faster & cheaper

Vanishing gradient problem for Sigmoid:

$\sigma'(x) = \sigma(x)(1 - \sigma(x))$

minimum when $x = 0$, ie $\sigma'(x=0) = 0.25 \to 0$, $x \to \pm\infty$

New weights $= \frac{\partial L}{\partial a} \cdot \sigma'(x) \cdot$ weights

↑ if small, ∴ new weights small.