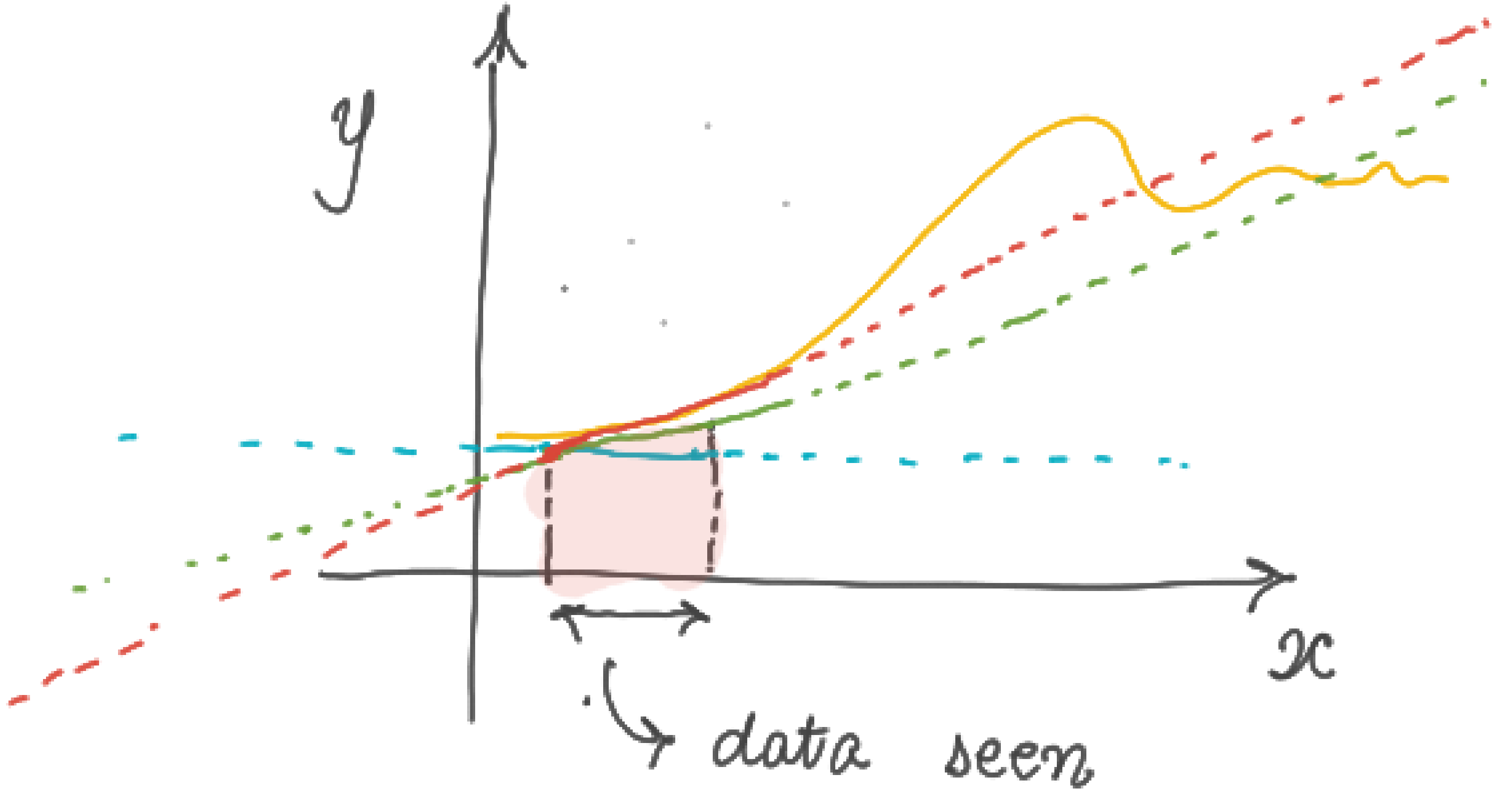
Problems with Classical ML

- make assumptions / impose inductive biases on the data
- makes dealing with unstructured data a pain
- provides no guarantees to finding the true hypothesis function

$y \rightarrow f(X)$

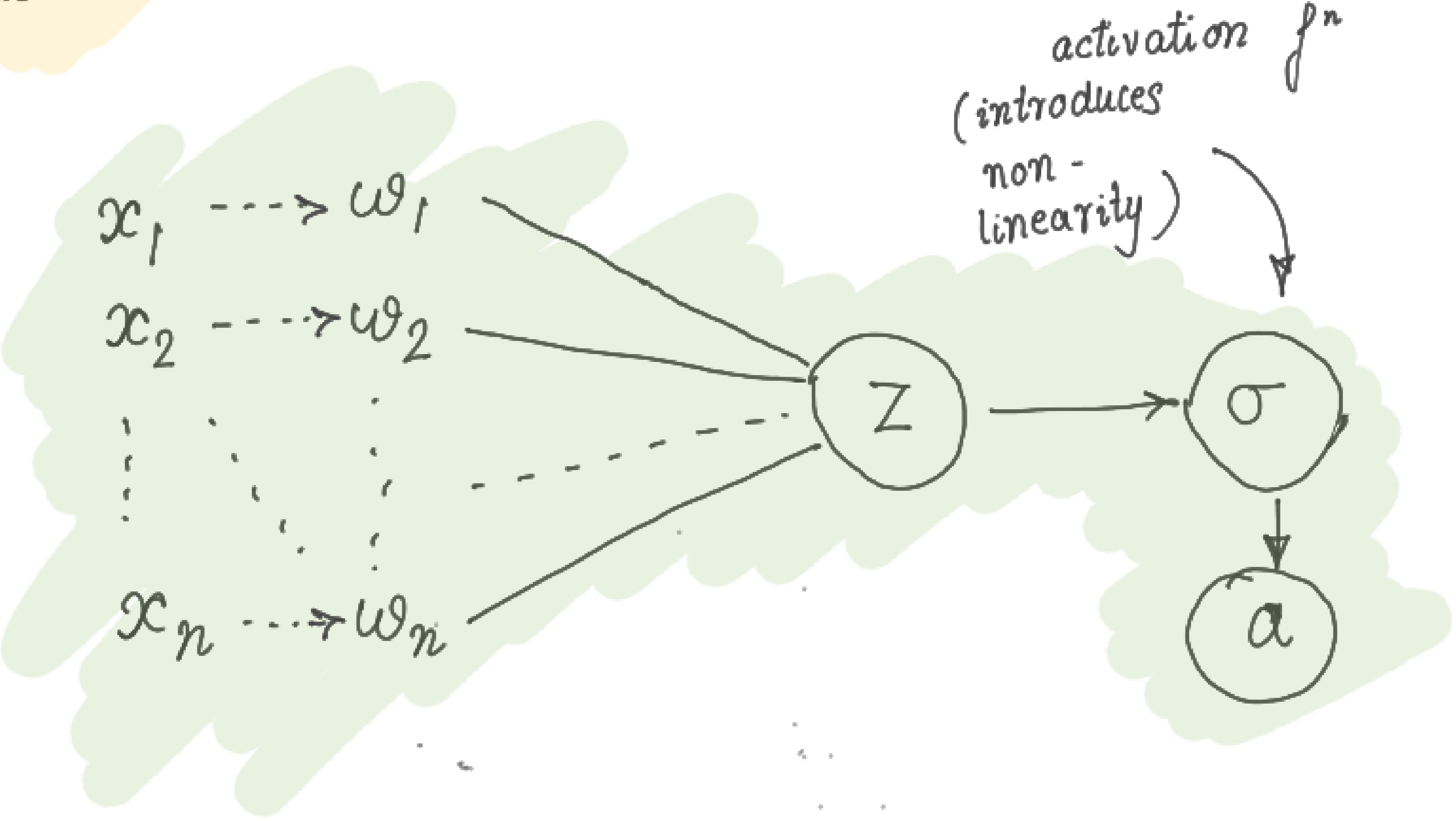
find f

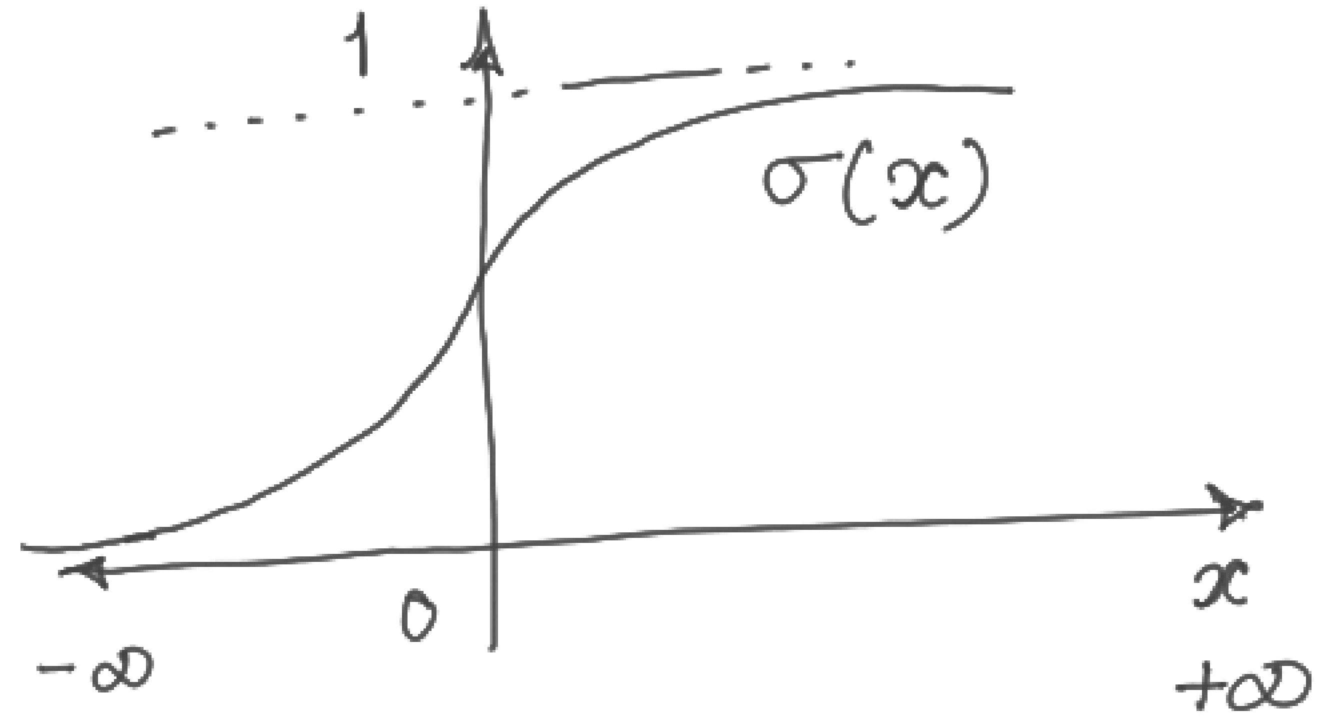


Enter Deep Learning

- no assumptions / inductive biases on data
- unstructured data can be used.
- by UAT (universal approximation theorem):
  all possible functions can be approximated by a neural network architechture.

Perceptron

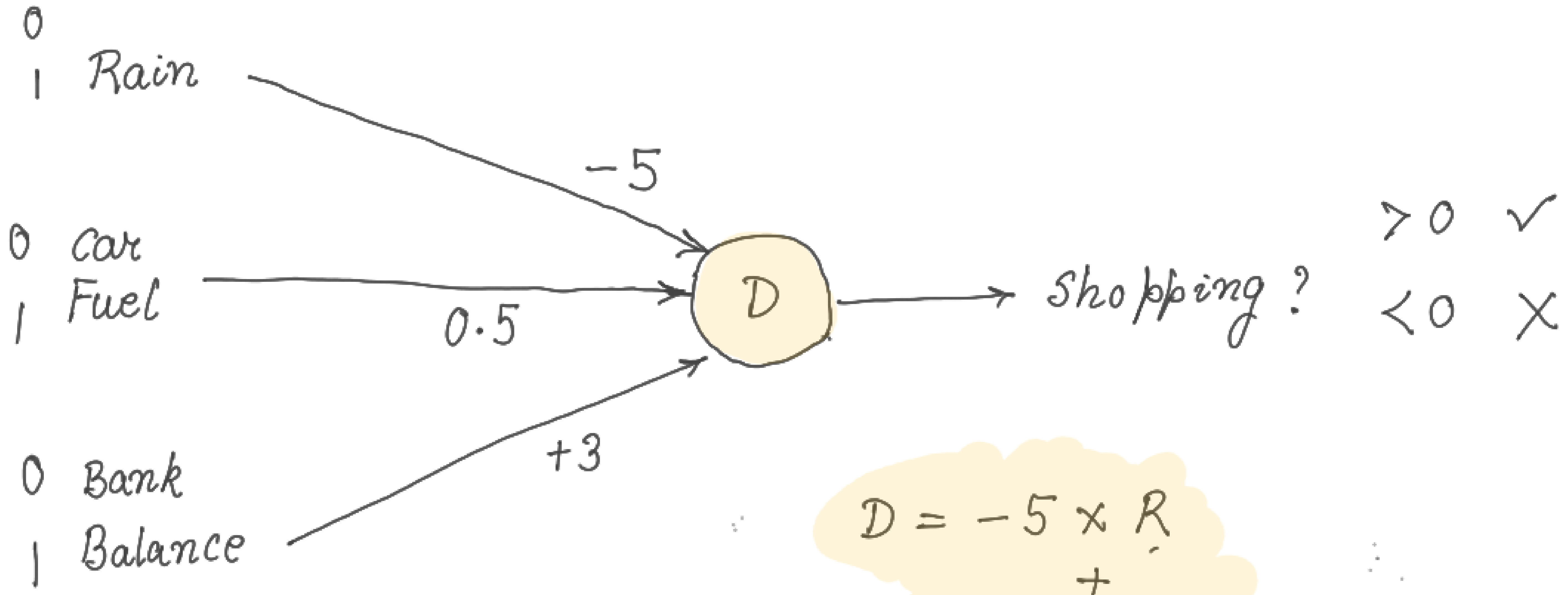




$$Z = w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots w_n x_n$$

$$a = \sigma(Z) = \frac{1}{1 + e^{-Z}}$$

Real Life Example of Perceptron

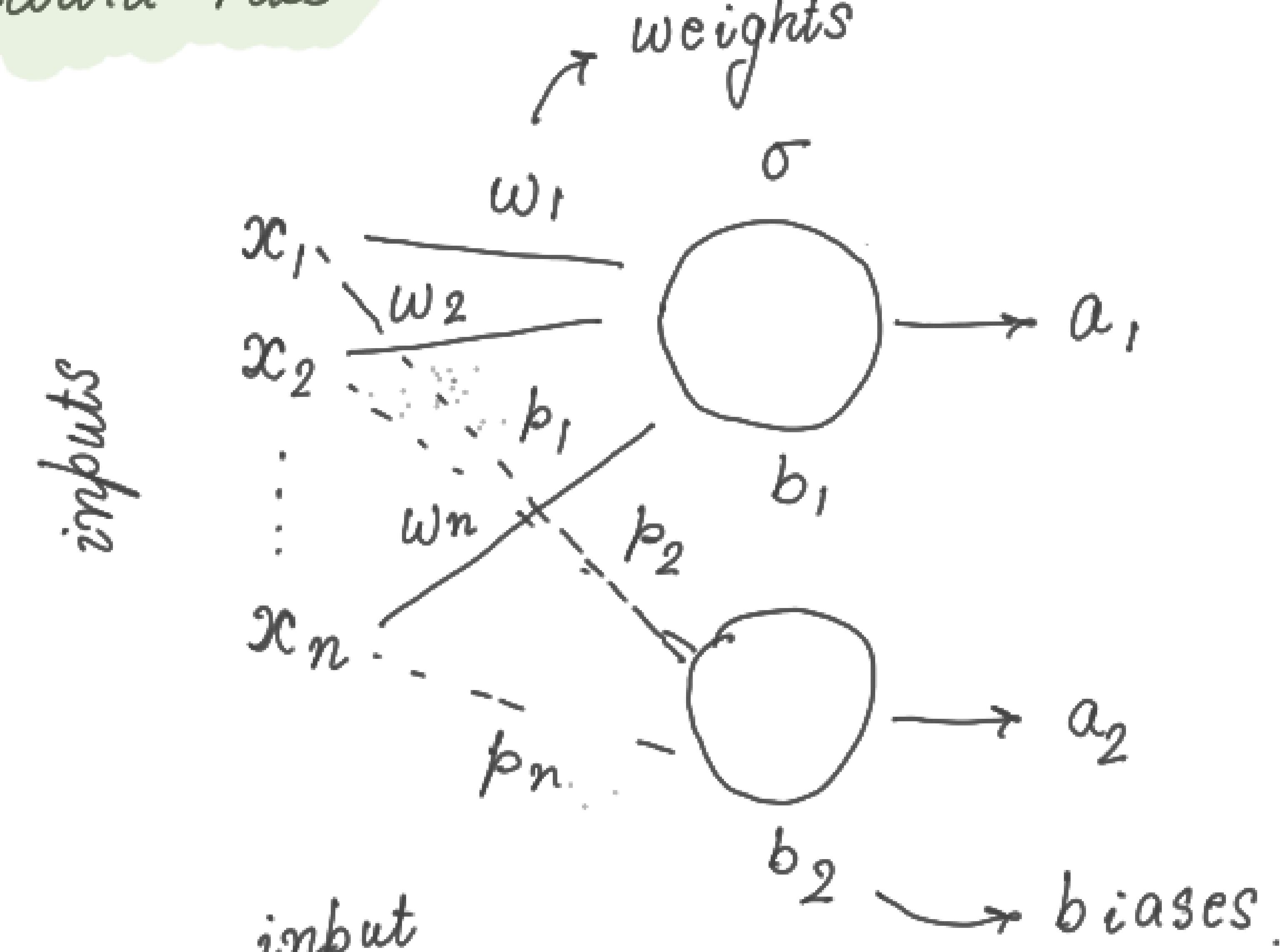


$$D = -5 \times R + 0.5 \times CF + 3 \times BB$$

R 1 : CF 1 : BB 1 → X

R 0 : CF 1 : BB 0 → ✓

Forward Pass

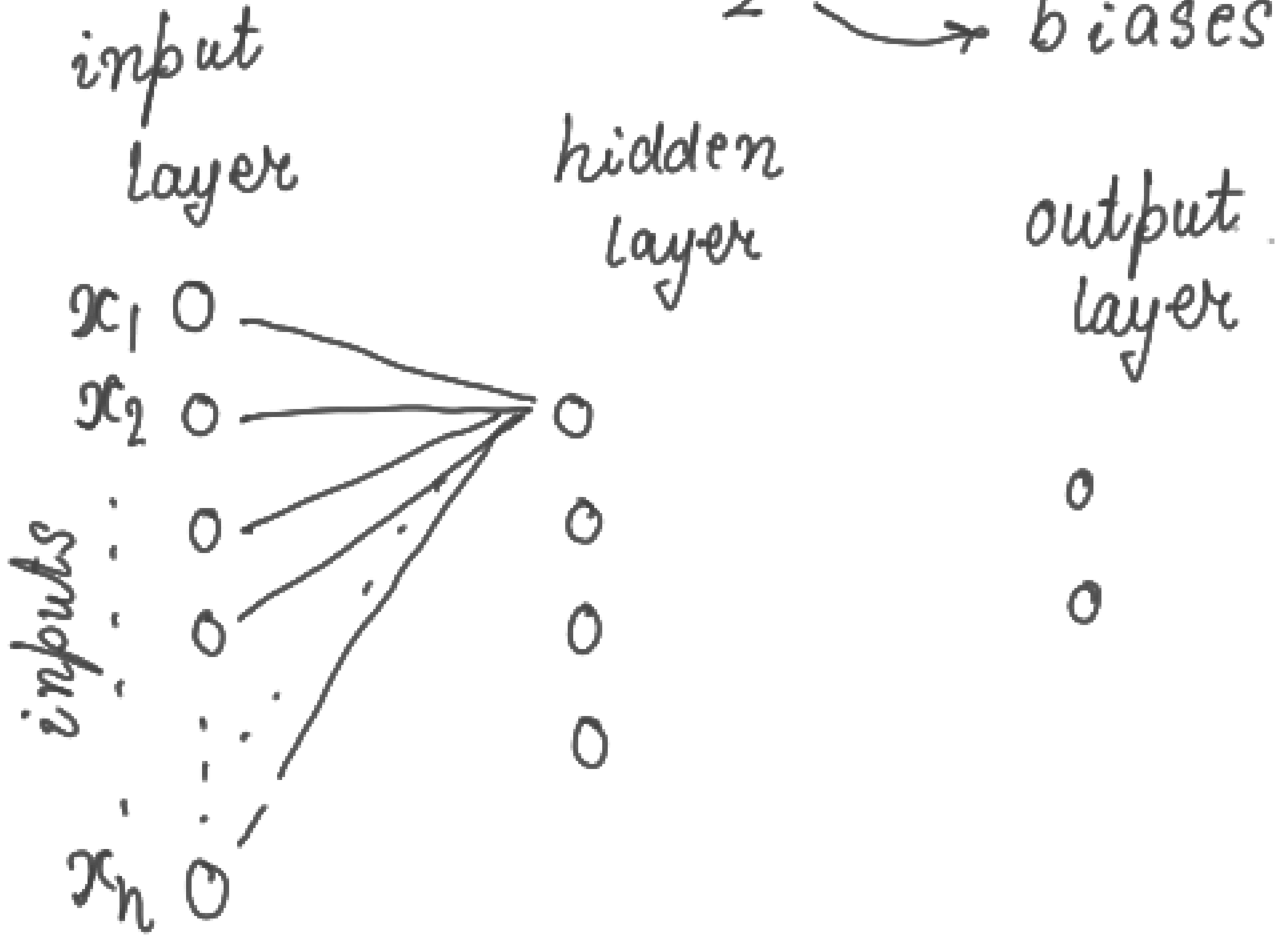


$$a = \sigma(z)$$

$$z = \sum w_i x_i + b_1$$

$$a_2 = \sigma(z_2)$$

$$z_2 = \sum p_i x_i + b_2$$



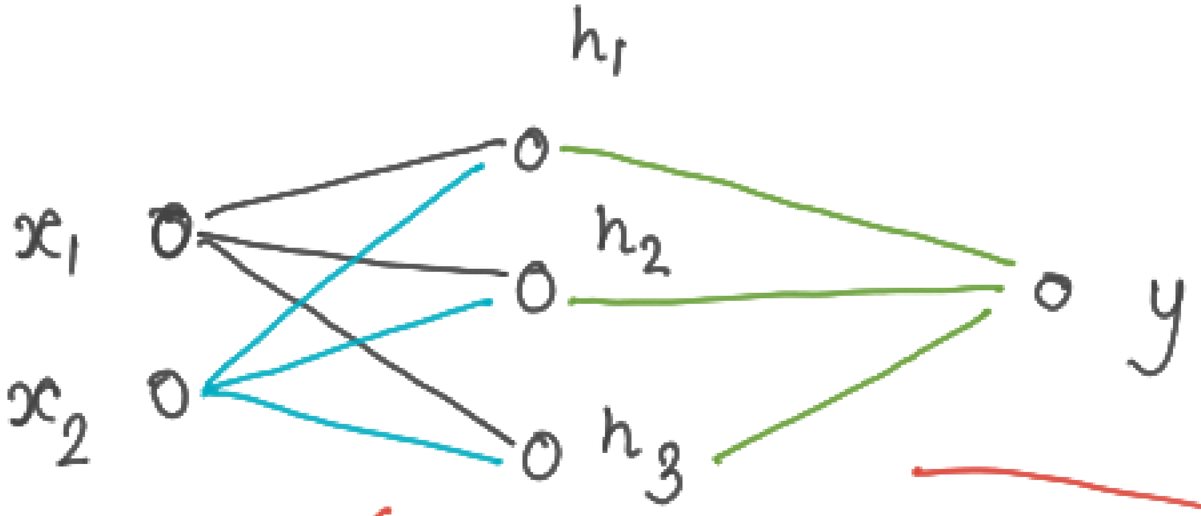
⇒ Information flows forward for prediction

– Forward - pass

⇒ all weights are known

Back propagation

- way to estimate weights for a neural network



w^i_{jk}

→ $i \rightarrow i+1$ th layer

→ k th node of $i+1$ th layer

→ j th node of ith layer

$$w^{1^T} = \begin{bmatrix} w^1_{11} & w^1_{21} \\ w^1_{12} & w^1_{22} \\ w^1_{13} & w^1_{23} \end{bmatrix}$$

$$\begin{bmatrix} w^2_1 & w^2_2 & w^2_3 \end{bmatrix} = w^{2^T}$$

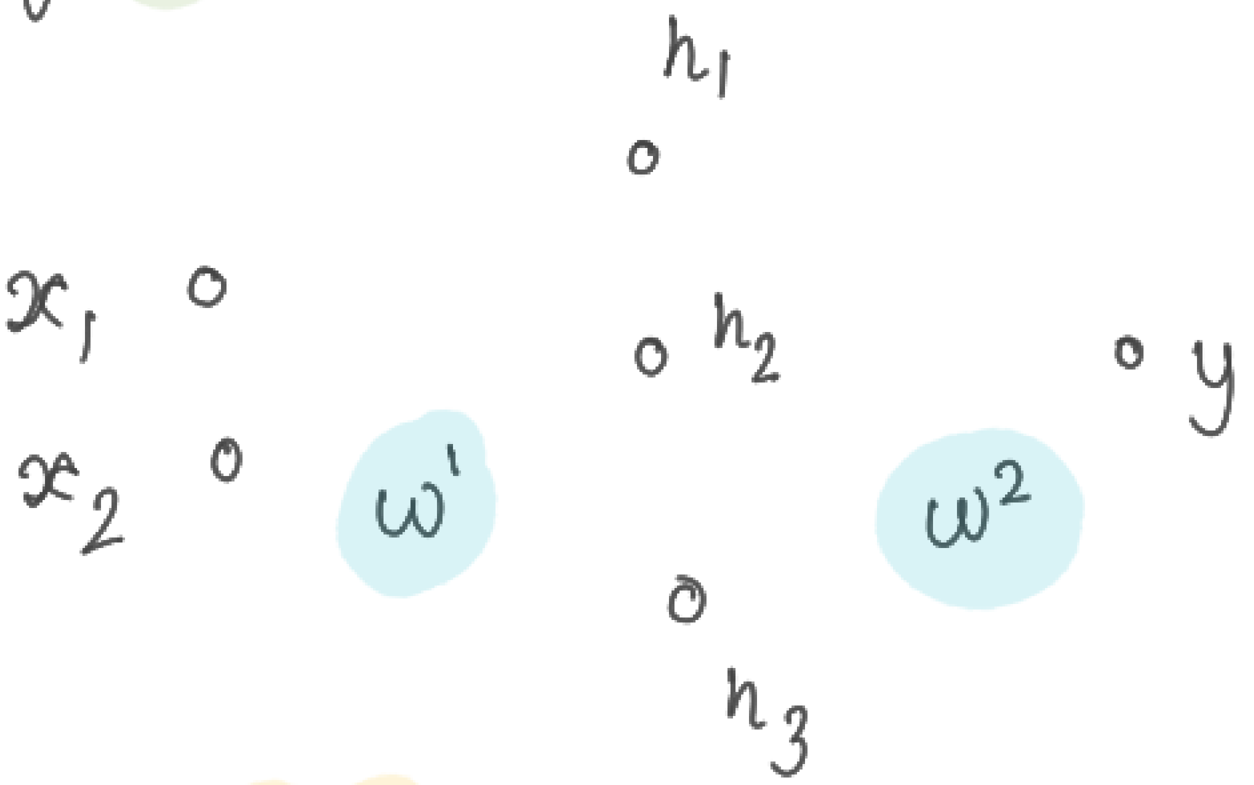
$$\hat{y} = w^{2^T} \cdot \underline{h} \rightarrow 1$$

$$h = w^{1^T} \cdot \underline{x} \rightarrow 2$$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow 3$$

Define a loss function $\mathcal{L} : \sum (y - \hat{y})$

Backpropagation



w^{1T}, w^{2T}

$$\mathcal{L} : \Sigma (y - \hat{y})^2$$

$$\hat{y} = w^{2T} \cdot h$$

$$\frac{\partial \mathcal{L}}{\partial w^{2T}} \propto -2(y - \hat{y}) \underline{h} \rightarrow \text{easy}$$

$$\frac{\partial \mathcal{L}}{\partial w^{1T}} = \underbrace{\frac{\partial \mathcal{L}}{\partial w^{2T}}}_{\text{reused from earlier}} \cdot \underbrace{\frac{\partial w^{2T}}{\partial w^{1T}}}_{\text{new component}} \rightarrow \text{repeat computation}$$

Technique to update model wts by calculating partial. dv. of loss. fn wrt. model wts; but also reusing earlier partial dvs. to speed up computation.

Backpropogation

weight updating :

$$w'^T \rightarrow w'^T - \frac{\partial \mathcal{L}}{\partial w'^T} \cdot \gamma \rightarrow \text{learning rate}$$

$$\phi : \phi(x, y, z)$$

$$\nabla \phi = \frac{\partial \phi}{\partial x}\hat{i} + \frac{\partial \phi}{\partial y}\hat{j} + \frac{\partial \phi}{\partial z}\hat{k} \quad : \text{Gradient}$$