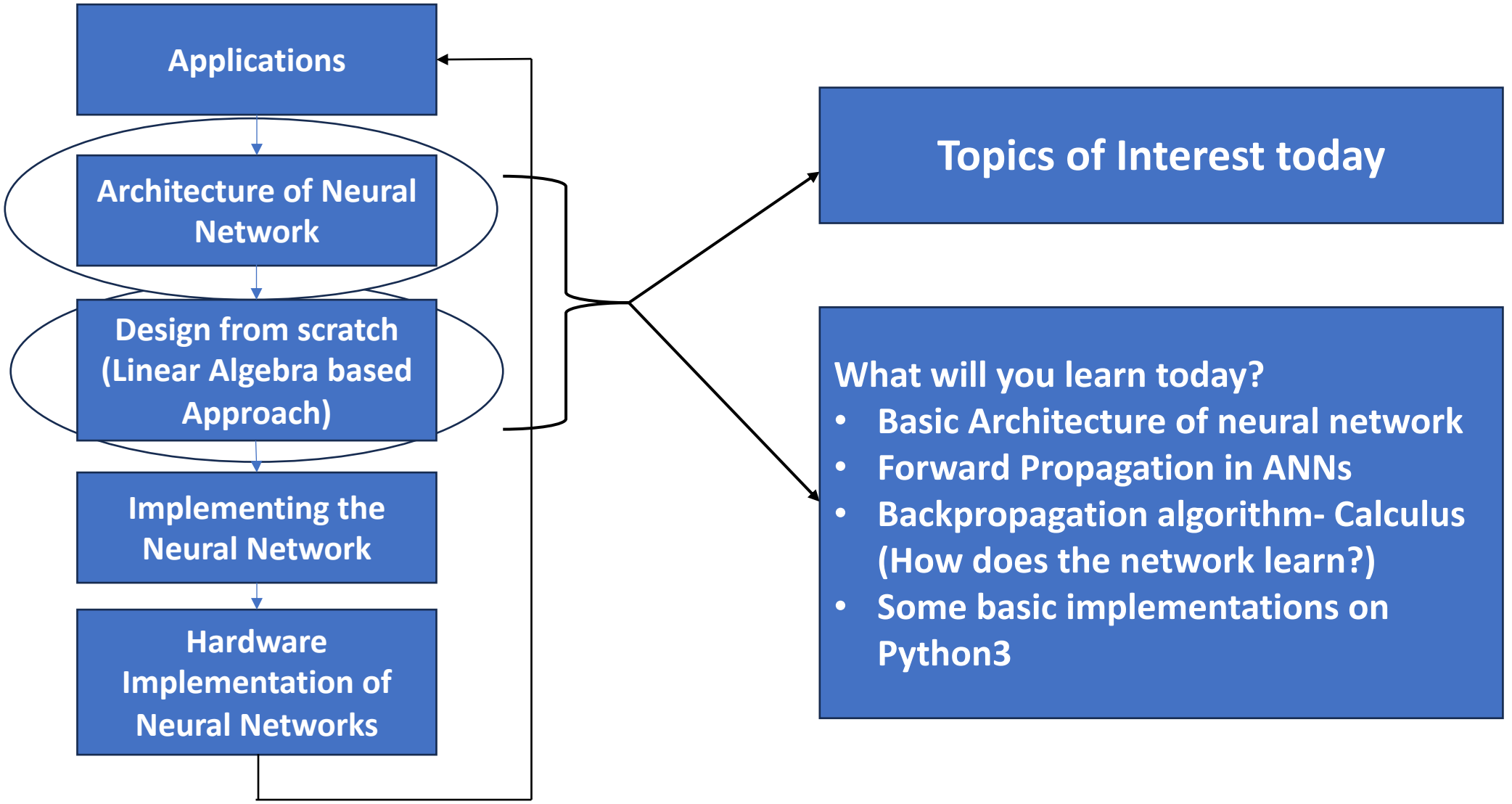
Artificial Neural Networks
AHP Slot-2
Neural Network: Design From Scratch

Dheemanth Joshi , Aniket Gangotri, CH Sai Pranay, Shwetha R*

Teaching Assistant, Department of Electronics and Communication Engineering, PES University

*** Faculty In-charge, Department of Electronics and Communication Engineering PES University**

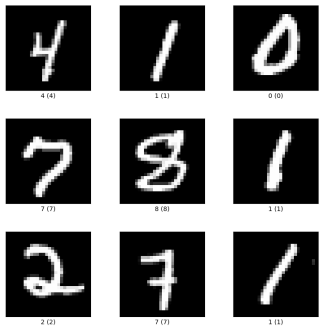
Plan of today's session

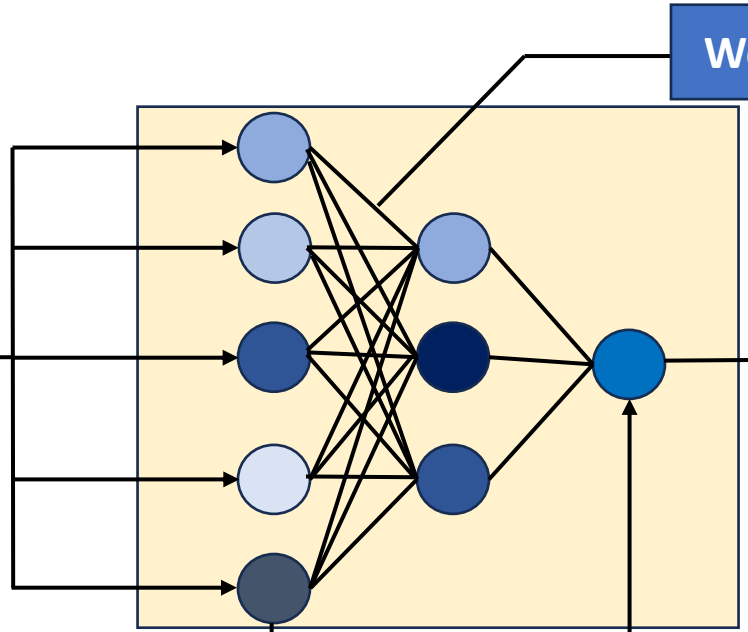


Architecture of ANN

Neural Network Building blocks: weights, biases, activation functions

Input Vector





Weight

Cat

Output

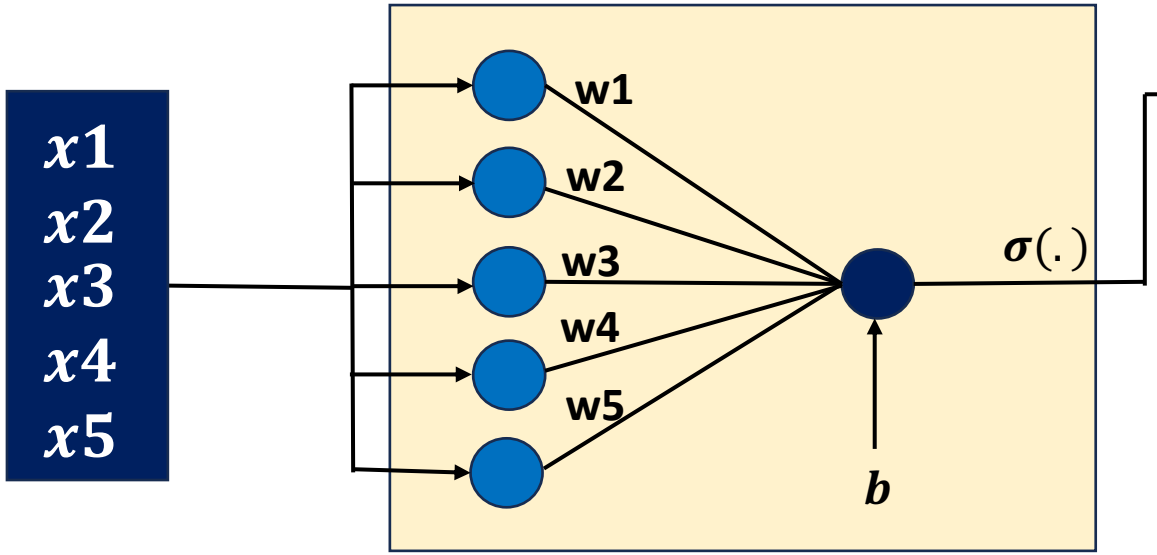
Neuron
1) Weighted influence
2) Activation

Bias

4	1	0
7	8	1
2	7	1

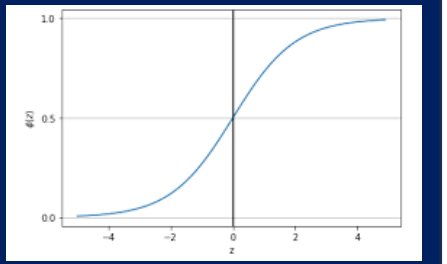
Design From Scratch: Forward Propagation

We start off the design by implementing the forward propagation, for this simple 5x1 Neural Network



$$\sigma(w1x1 + w2x2 + w3x3 + w4x4 + w5x5 + b)$$

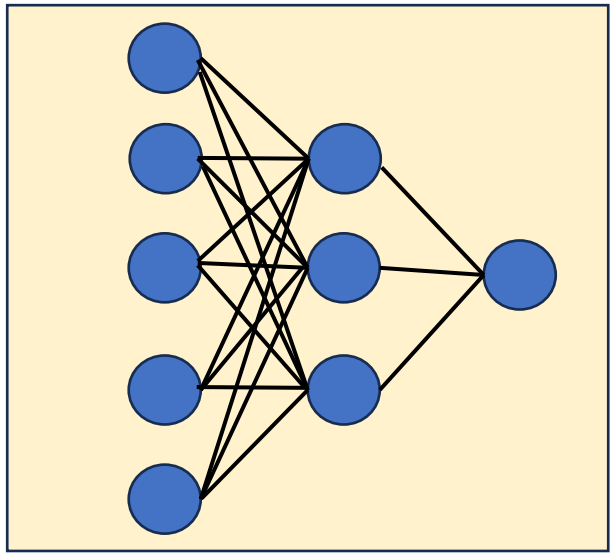
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Task-1: Design This neural network from scratch on Python-3.

Design From Scratch: Forward Propagation

Design This Neural Network from scratch on Python-3. Assume all the weights and biases are randomly assigned



$\mathbb{R}^5 \rightarrow \mathbb{R}^3 \rightarrow \mathbb{R}$

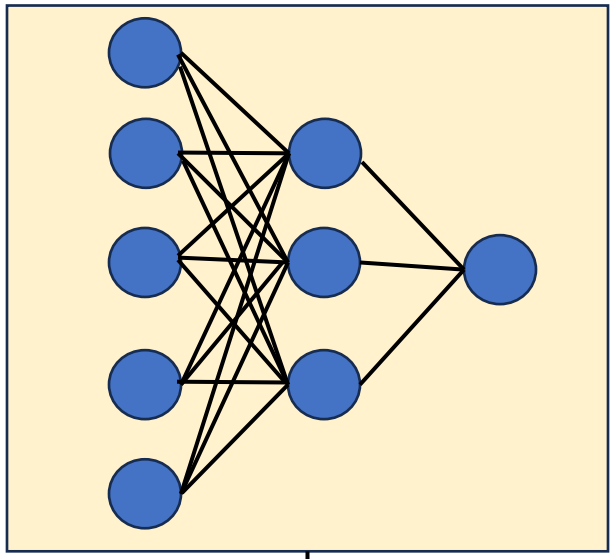
What is $y = W^T X + b$?

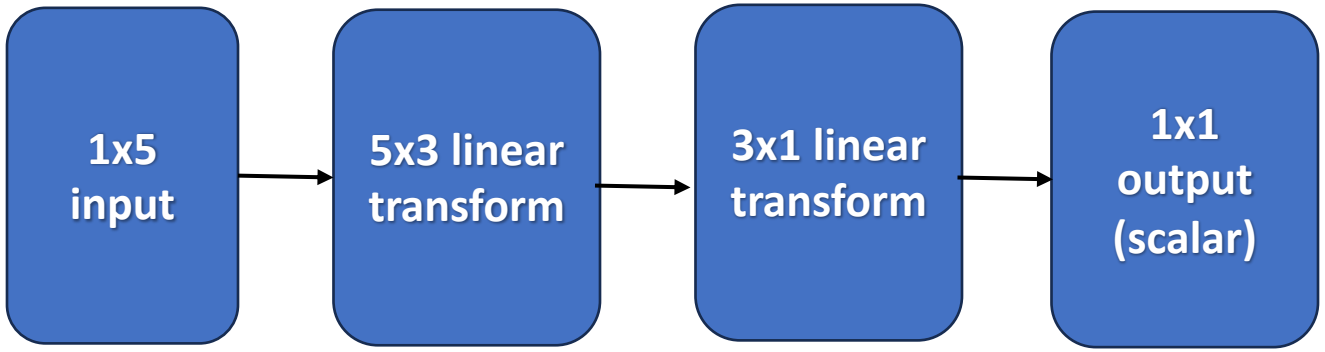
Hint : It is a linear transform

Sounds Familiar?

Forward Pass: what are we actually doing?

The Layers of the neural network (weights and biases) can be expressed in terms of matrices and can be forwarded by weight matrix multiplication and addition of bias vector

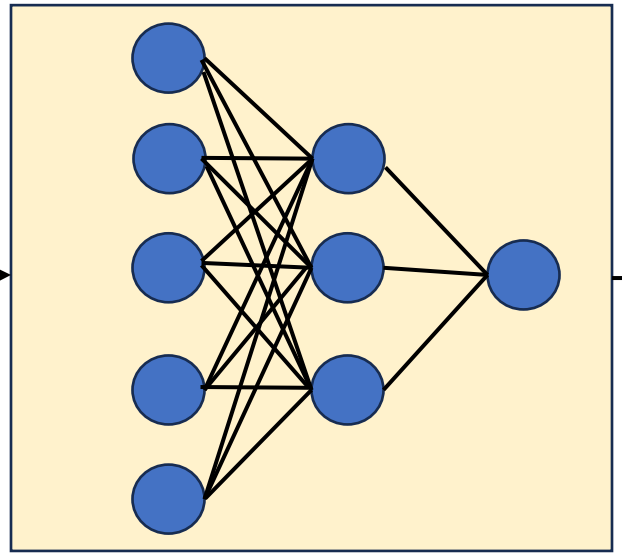




Essence of $y = W^T X + b$

Backward Pass: Cost

Loss, Error, Cost etc… metrics indicate how much the model is far/ not optimal to perform a certain application. There are many functions to calculate cost of a given neural network.



? ? ?

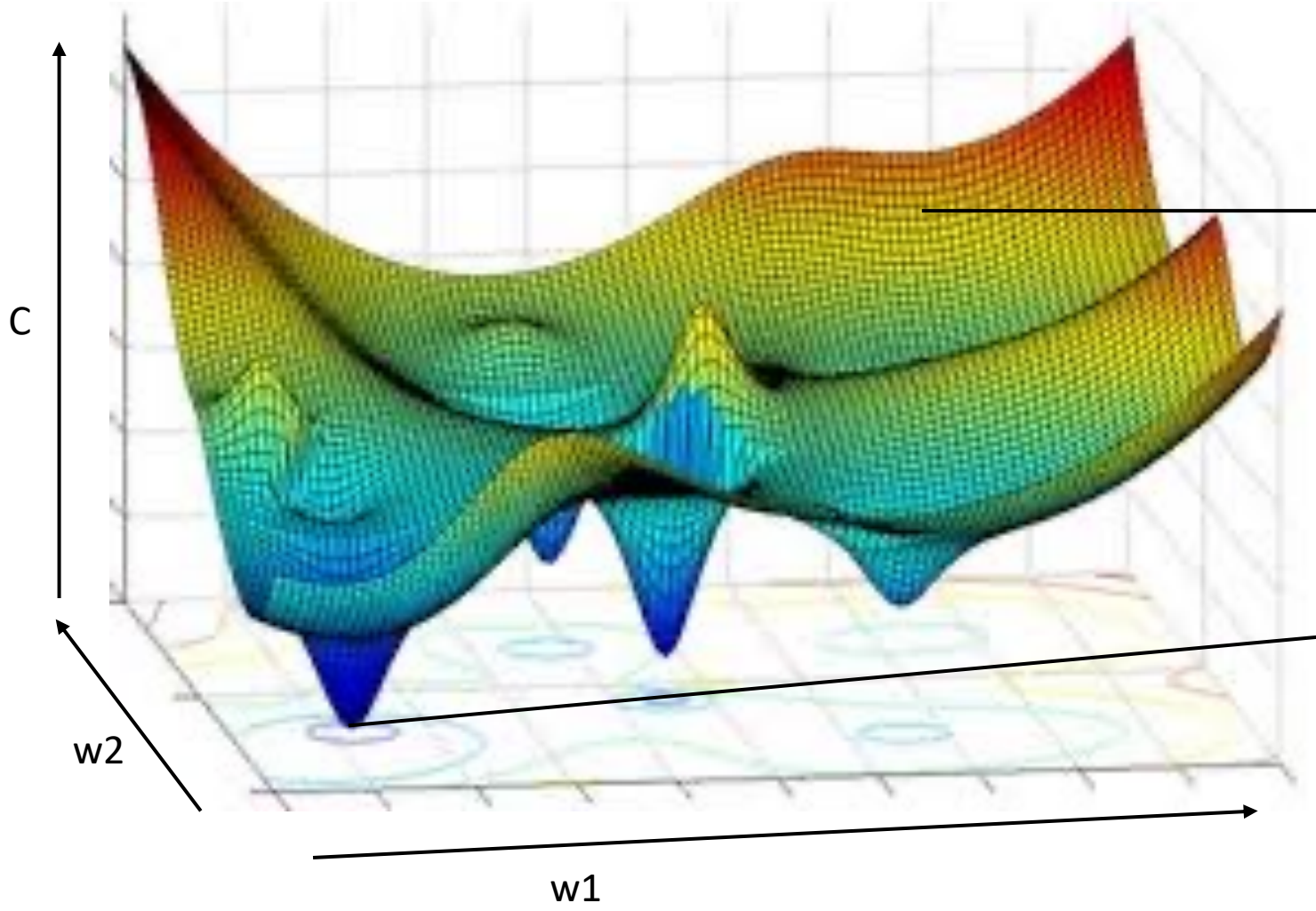
Dog

$$\mathbb{C} = \frac{\sum_{n=1}^{N}(y^t - y)^2}{N}$$

Inference: Model is not trained to detect and classify features of a cat as "cat"

Note that neural network algorithms are supervised, i.e. they need data with valid targets to learn the features of the applications they are being trained on.

Backward Pass: What are we trying to achieve?

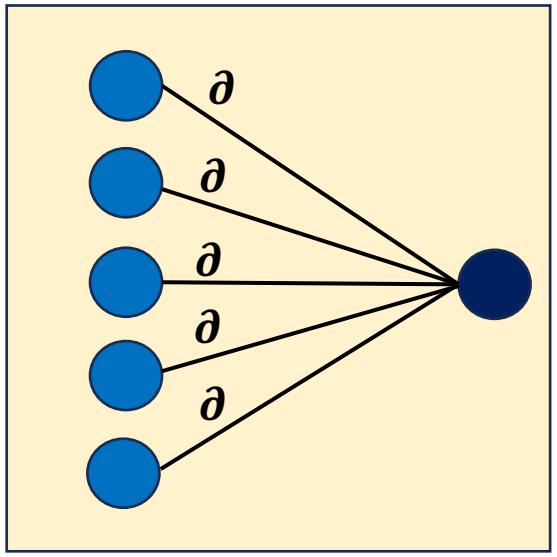


Our current position (random configuration)

Where we want to be.

Backward Pass: The Backpropagation Algorithm

Backward pass essentially provides us with gradients for each parameters defined in the neural network. These gradients point to the global minimum of the cost function defined wrt to the parameters.

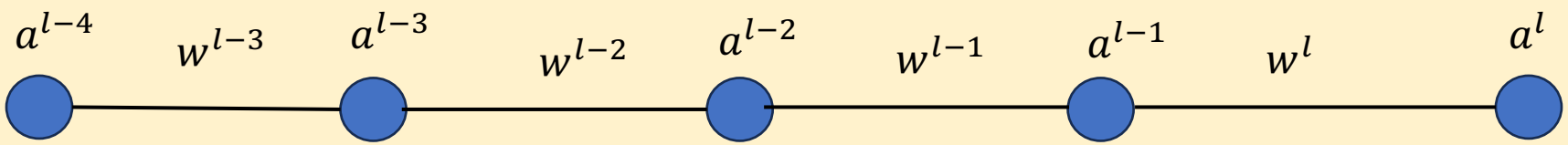


Gradients are obtained through the algorithm. It computes the gradients through the cost metric given the state of parameters

Once we obtain the gradients for each parameter, we update our weights by using any of the available updating techniques. For example, gradient descent.

Backpropagation algorithm: Simplified

Consider the following neural network, lets define all the parameters for this configuration.



$$C = \frac{(a^l - y^t)^2}{2}$$

$$define:$$

$$z^l = w^l a^{l-1} + b^l$$

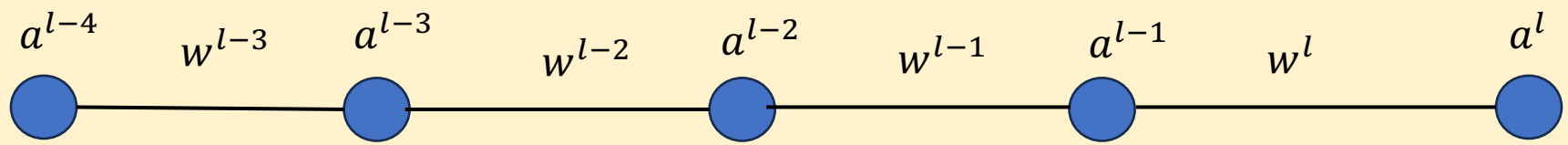
$$a^l = \sigma(z^l)$$

Our Aim is to Find:

$$\frac{\partial C}{\partial w^l} \quad \frac{\partial C}{\partial b^l}$$

Backpropagation algorithm: Equations

The required expressions are obtained by chain rule. We apply chain rule to the functions defined earlier.



$$\frac{\partial C}{\partial w^l} = \frac{\partial C}{\partial a^l}\ \frac{\partial a^l}{\partial z^l}\ \frac{\partial z^l}{\partial w^l}$$

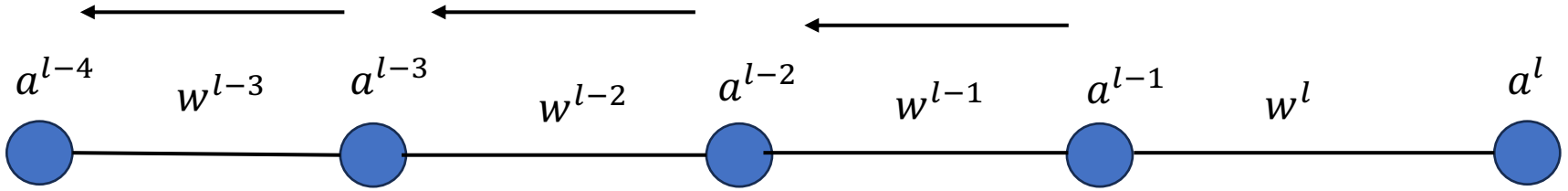
$$\frac{\partial C}{\partial b^l} = \frac{\partial C}{\partial a^l}\ \frac{\partial a^l}{\partial z^l}\ \frac{\partial z^l}{\partial b^l}$$

$$\frac{\partial C}{\partial a^{l-1}} = \frac{\partial C}{\partial a^l}\ \frac{\partial a^l}{\partial z^l}\ \frac{\partial z^l}{\partial a^{l-1}}$$

Backpropagation algorithm: Recursive Property

The same set of equations can recursively be propagated to all the set of layers of neuron hence the name ”Back Propagation”

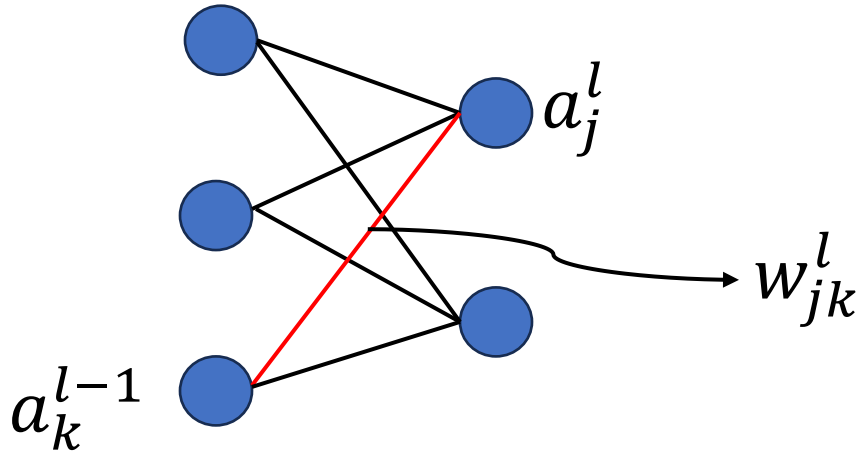
$$\left\{\begin{array}{l} \frac{\partial C}{\partial w^l} = \frac{\partial C}{\partial a^l}\frac{\partial a^l}{\partial z^l}\frac{\partial z^l}{\partial w^l} \\ \frac{\partial C}{\partial b^l} = \frac{\partial C}{\partial a^l}\frac{\partial a^l}{\partial z^l}\frac{\partial z^l}{\partial b^l} \\ \frac{\partial C}{\partial a^{l-1}} = \frac{\partial C}{\partial a^l}\frac{\partial a^l}{\partial z^l}\frac{\partial z^l}{\partial a^{l-1}} \end{array}\right.$$



But isn’t this model over simplified? It may get really get complex if you add more neurons each layer?
What do you say?

Backpropagation algorithm: Back to our original network

We add a few more indices to keep track of the weight locations



What actually has changed?

$$\frac{\partial C}{a_k^{l-1}} = \sum_{j=0}^{N-1} \frac{\partial C}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l} \frac{\partial z_j^l}{\partial a_k^{l-1}}$$

Updated algorithm equations

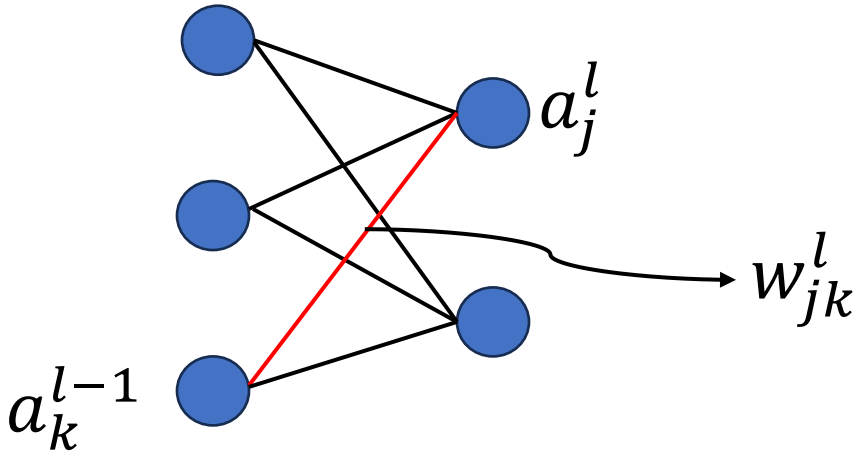
$$z_j^l = \cdots + w_{jk}^l a_k^{l-1} + \ldots$$

$$C = \sum_{j=0}^{N-1} (a_j^l - y^j)^2$$

$$\frac{\partial C}{\partial w^l} = \frac{\partial C}{\partial a^l} \frac{\partial a^l}{\partial z^l} \frac{\partial z^l}{\partial w^l}$$

$$\frac{\partial C}{\partial b^l} = \frac{\partial C}{\partial a^l} \frac{\partial a^l}{\partial z^l} \frac{\partial z^l}{\partial b^l}$$

Backpropagation algorithm: Back to our original network

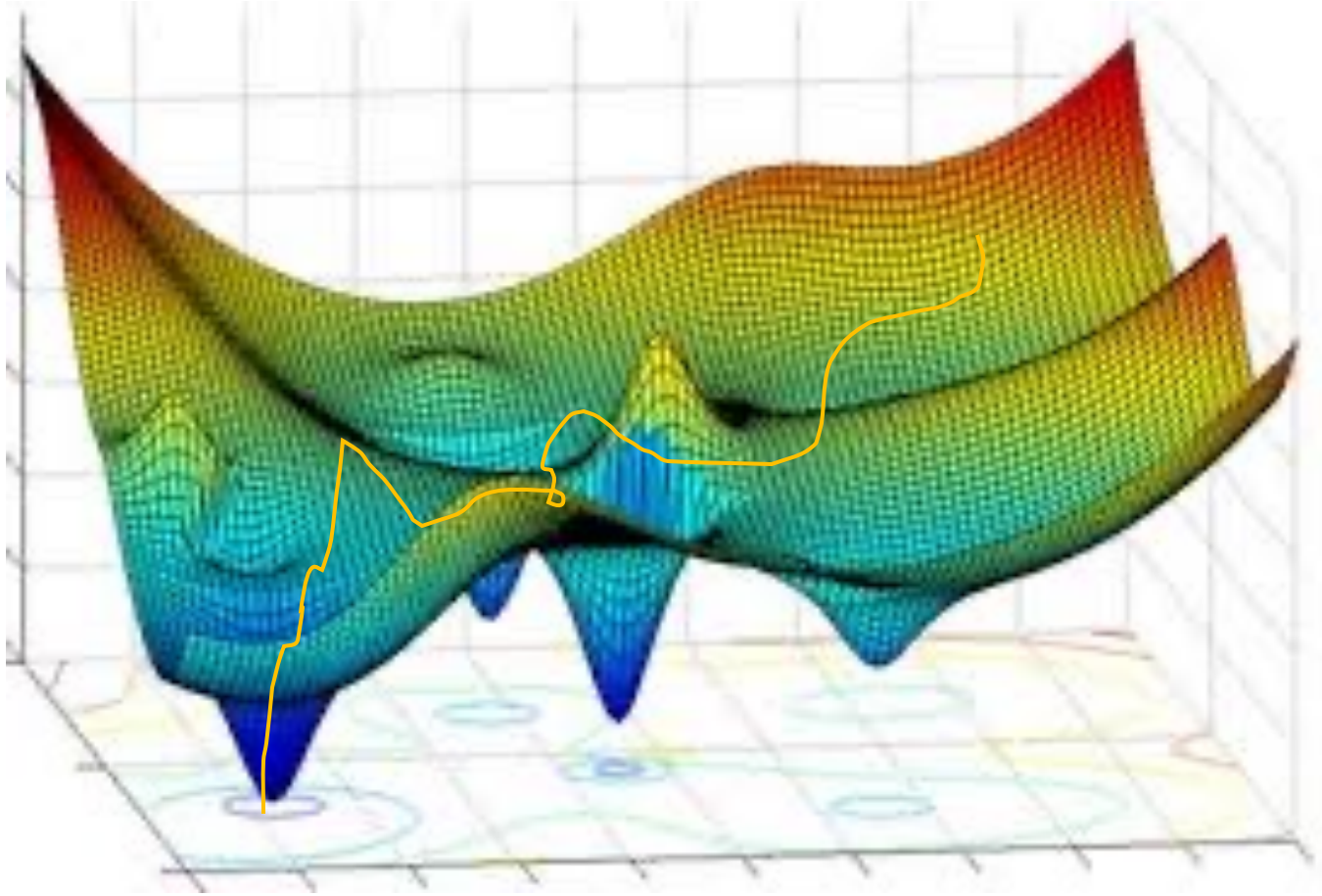


$$\frac{\partial C}{a_k^{l-1}} = \sum_{j=0}^{N-1} \frac{\partial C}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l} \frac{\partial z_j^l}{\partial a_k^{l-1}}$$

Weight Updating: Gradient Descent

$$w = w - \eta \frac{\partial C}{\partial w}$$

$$b = b - \eta \frac{\partial C}{\partial b}$$



QNA

Thanks