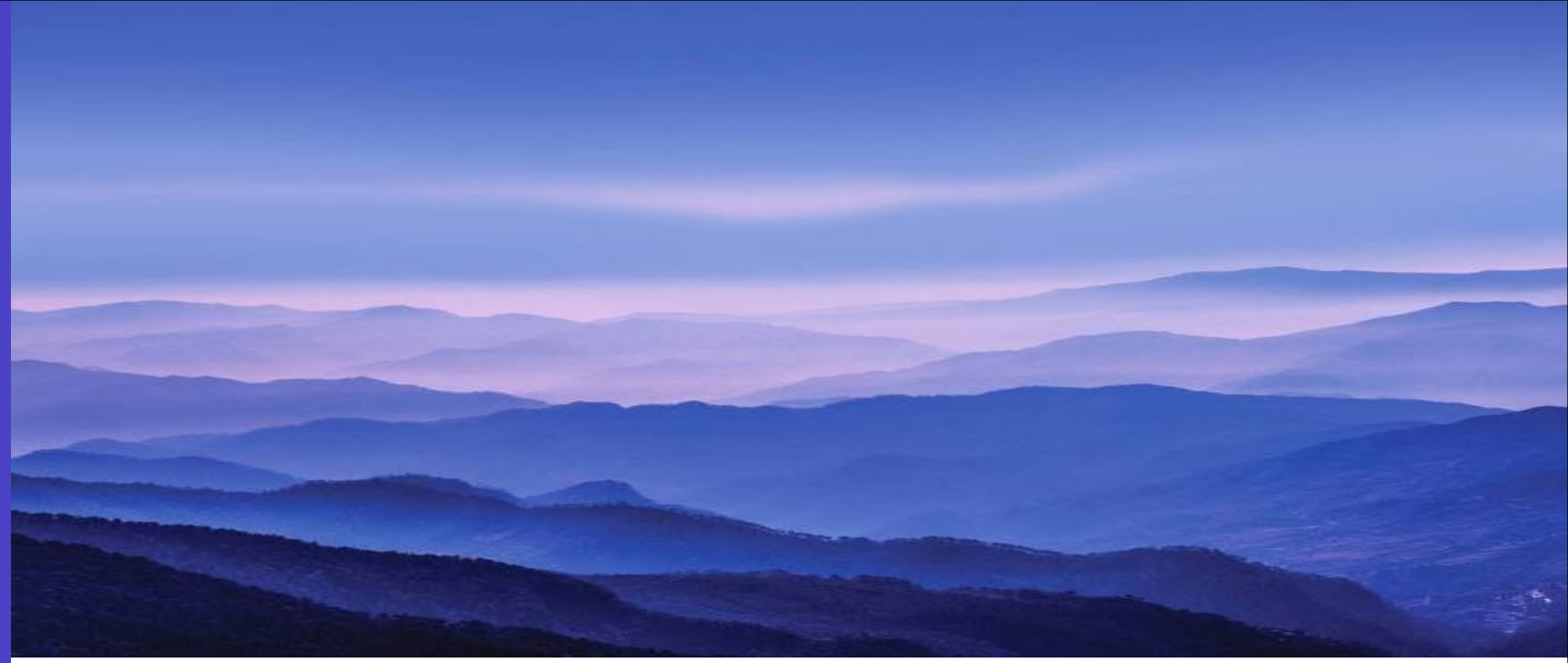
Introduction to Deep Learning

CFI summer school 2021

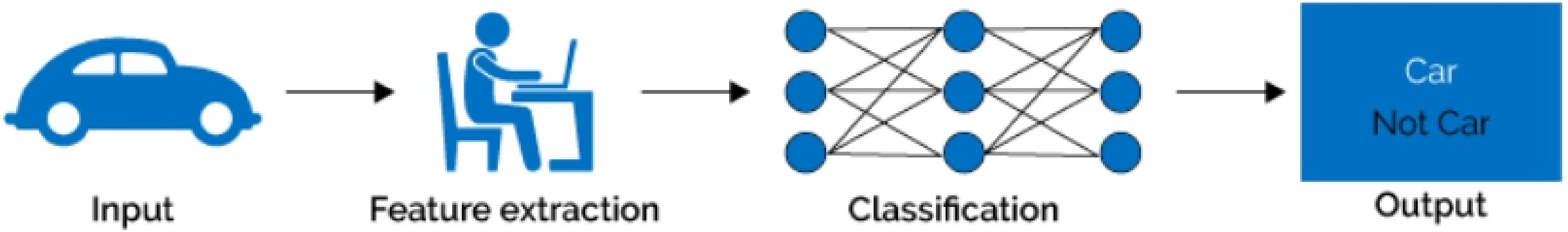
Agenda



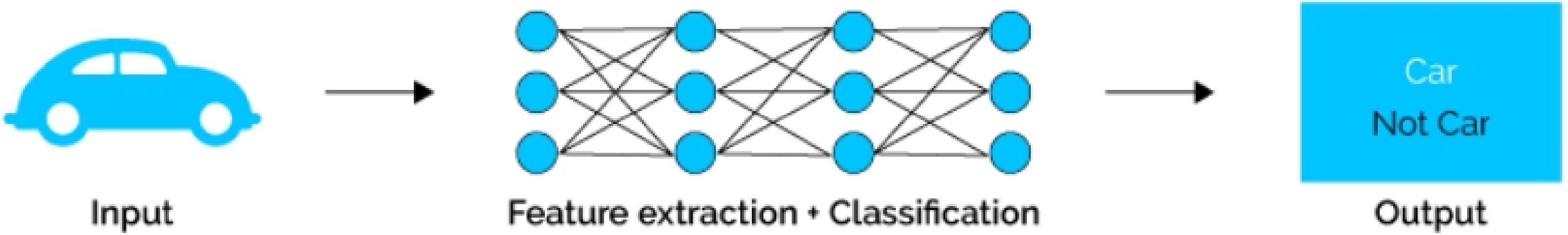
- What is deep learning? How is it different from ML?
- Neuro-inspired origins of DL
- What are perceptrons?
- What is a neural network?
- What is gradient descent?
- What is back propagation?

Why deep learning?

Machine Learning



Deep Learning

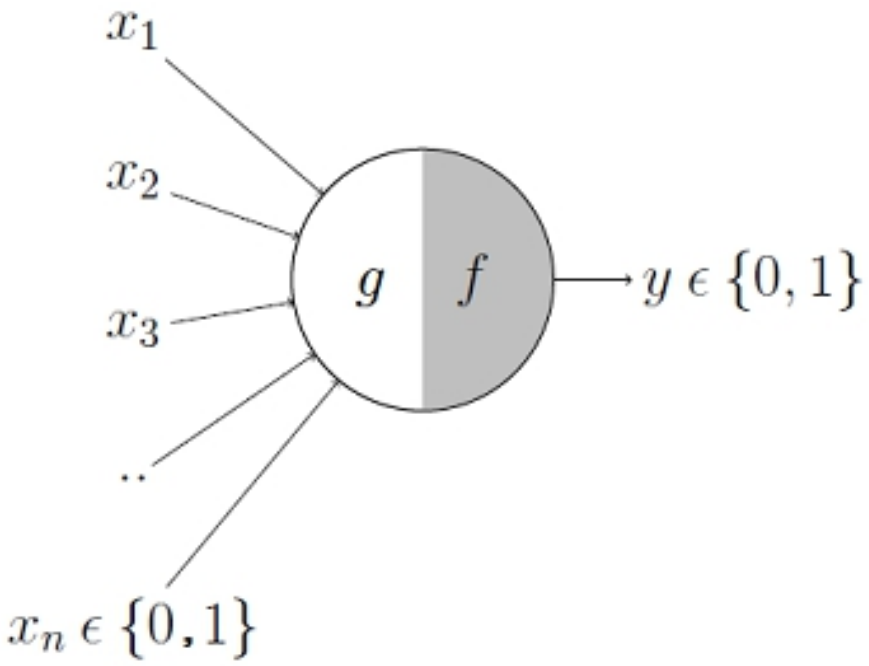



What was the biological inspiration
behind neural networks?

Biological Neurons

Hubel and Wiesel cat experiment

McCulloch Pitts (MP) Neuron



- The MP neuron is divided into two parts.
- The first part (g) takes an input and performs an aggregation
- The second part (f) then based on the aggregated value makes a decision.

Thresholding logic

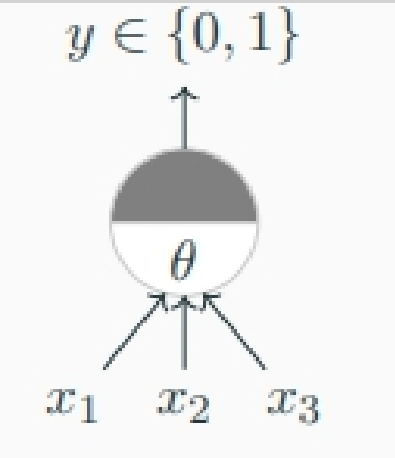
In mathematical terms, this is what an MP neuron does

$$g(x_1, x_2, x_3, \ldots, x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$

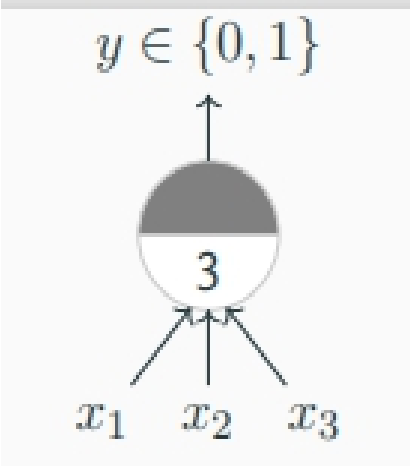
$$\begin{aligned} y = f(g(\mathbf{x})) &= 1 \quad if \quad g(\mathbf{x}) \geq \theta \\ &= 0 \quad if \quad g(\mathbf{x}) < \theta \end{aligned}$$

θ is called the thresholding parameter and this procedure is called thresholding logic

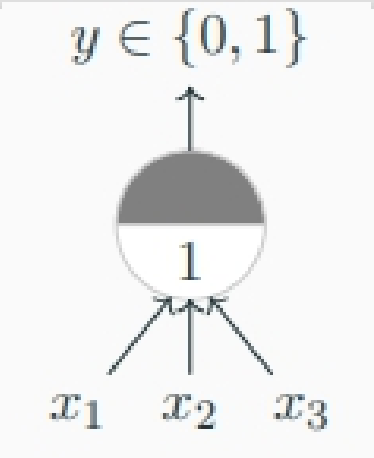
A few examples...



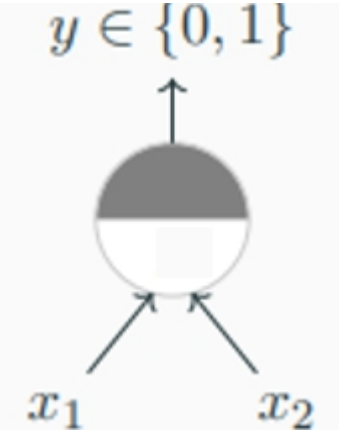
MP neuron
XOR GATE



AND GATE



OR GATE



XOR – Linearly inseparable

Limitation of MP neuron

- A single MP neuron splits the input points (4 points for 2 binary inputs) into two halves-

  $\sum_{i=1}^{n} x_i - \theta = 0$

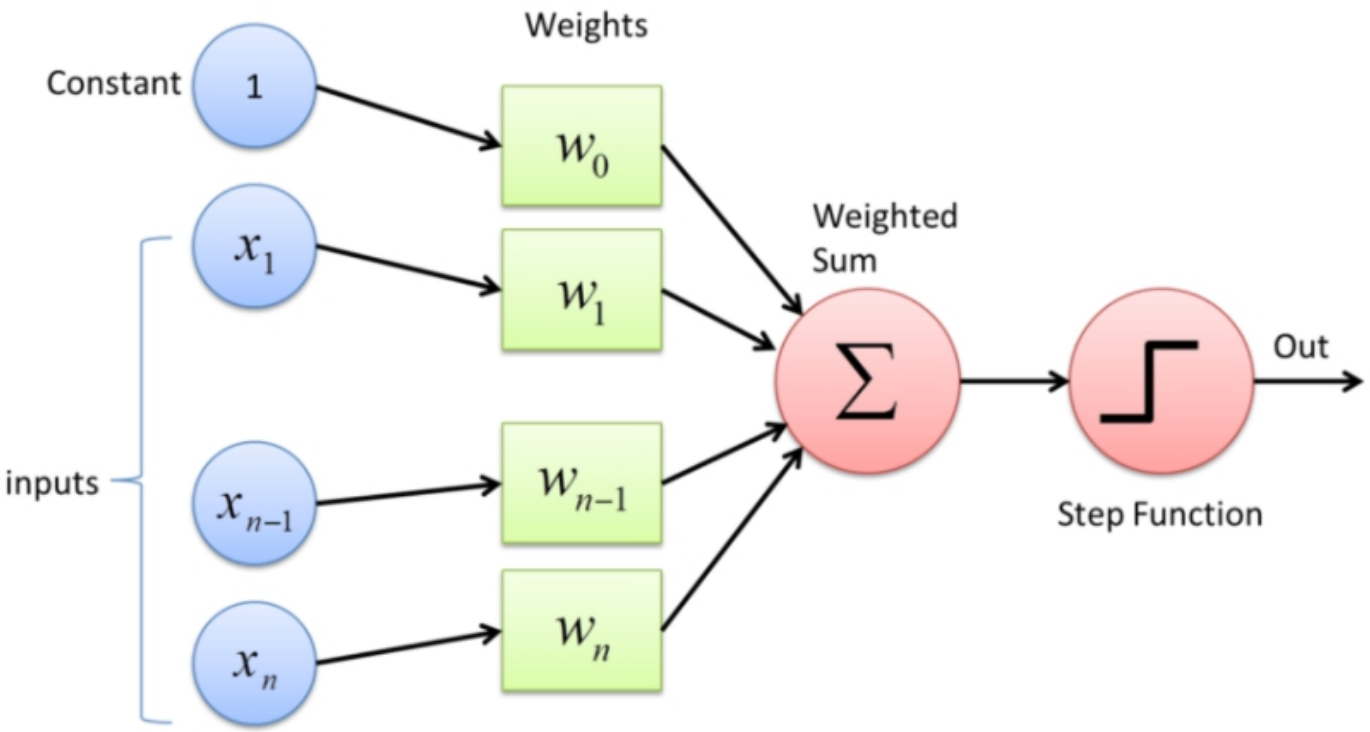
  Points lying on or above the line and points lying below this line

- In other words, all inputs which produce an output 0 will be on one side of the line and all inputs which produce an output 1 will lie on the other side of this line

- What about non-Boolean (real) inputs?
- What about functions which are not linearly separable?
- What if there are some inputs which are more important than others?


Perceptrons, PLA and
multilayer perceptrons

Perceptron



- The perceptron is the basic unit of a neural network.
- It is a more general computational model than MP neurons

So, how exactly is it different from a MP neuron?

1. It has weights associated with inputs, and has a mechanism to learn those weights.
2. The inputs need not be Boolean

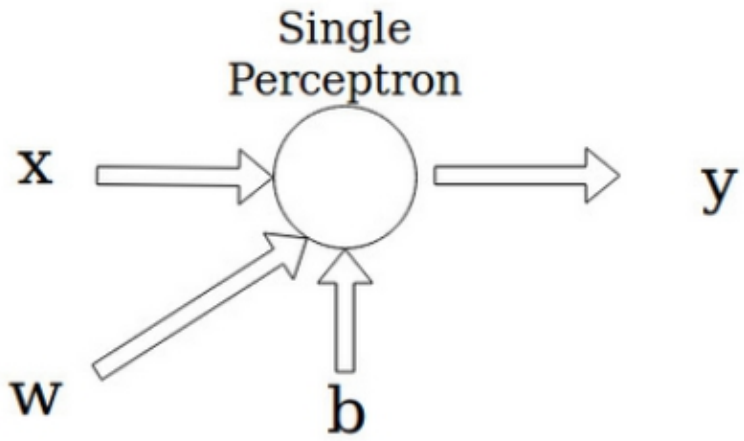
Maths of perceptrons and PLA

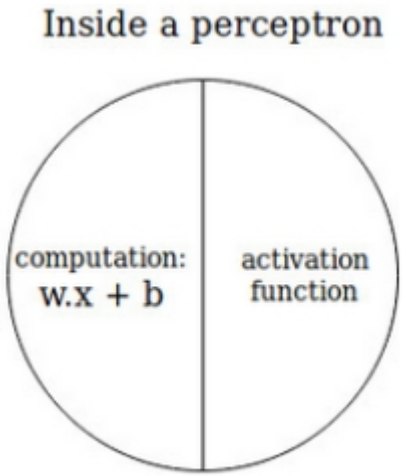
Consider

$$X = \{x_1, \quad x_2, \quad \ldots, \quad x_n\}$$

$$W = \{w_0, \quad w_1, \quad \ldots, \quad w_n\}$$

w_0 is the bias





$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i \geq \theta$$

$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i < \theta$$

$$y = 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0$$

Algorithm: Perceptron Learning Algorithm

$P \leftarrow inputs \quad with \quad label \quad 1;$

$N \leftarrow inputs \quad with \quad label \quad 0;$

Initialize $\mathbf{w}$ randomly;

while $!convergence$ **do**

  Pick random $\mathbf{x} \in P \cup N$;

  if $\mathbf{x} \in P \quad and \quad \sum_{i=0}^{n} w_i * x_i < 0$ **then**

    $\mathbf{w} = \mathbf{w} + \mathbf{x}$;

  end

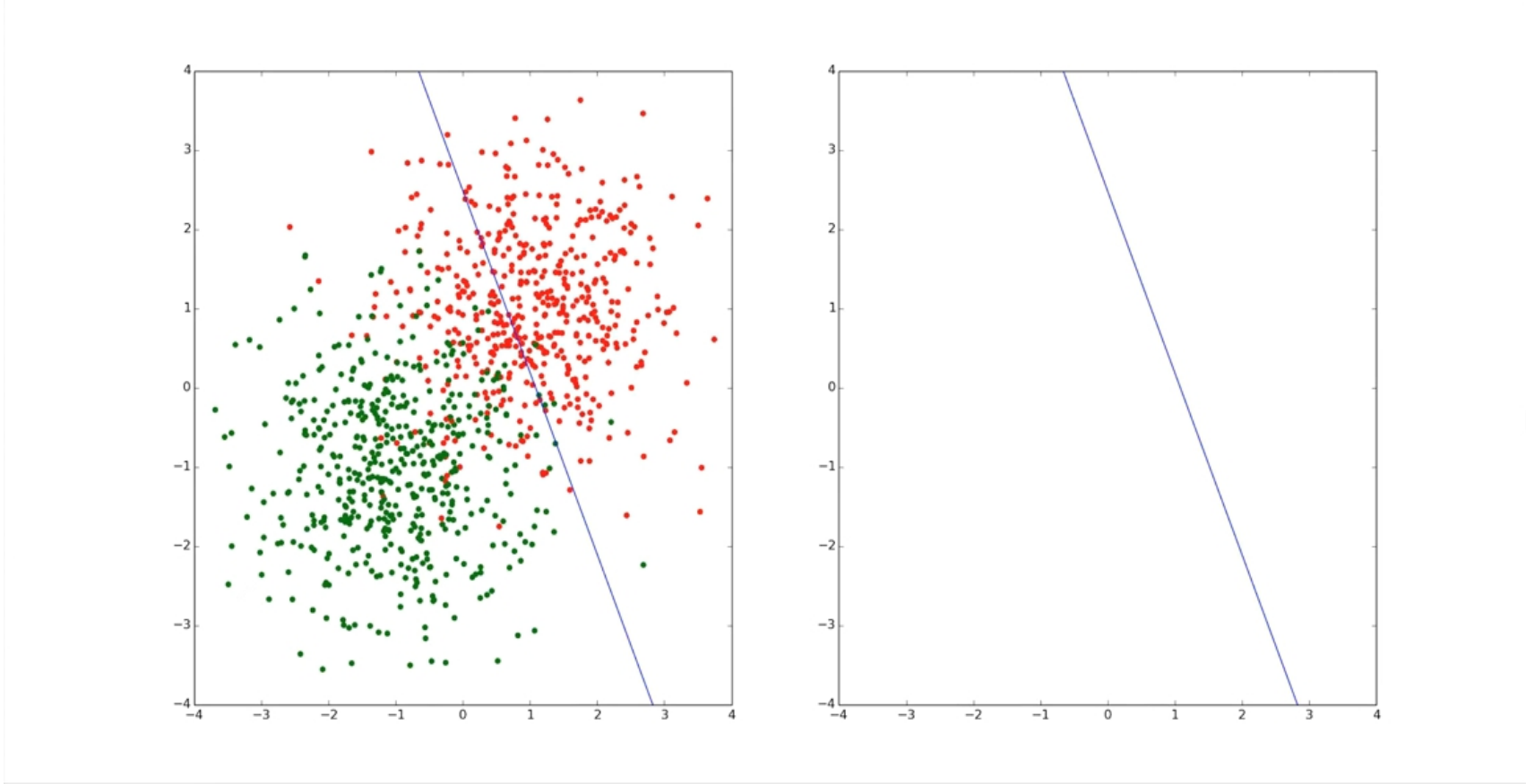
  if $\mathbf{x} \in N \quad and \quad \sum_{i=0}^{n} w_i * x_i \geq 0$ **then**

    $\mathbf{w} = \mathbf{w} - \mathbf{x}$;

  end

end

Animation of perceptron learning algorithm



Consider the XOR function again. Do you think the perceptron model will be able to deal with this?

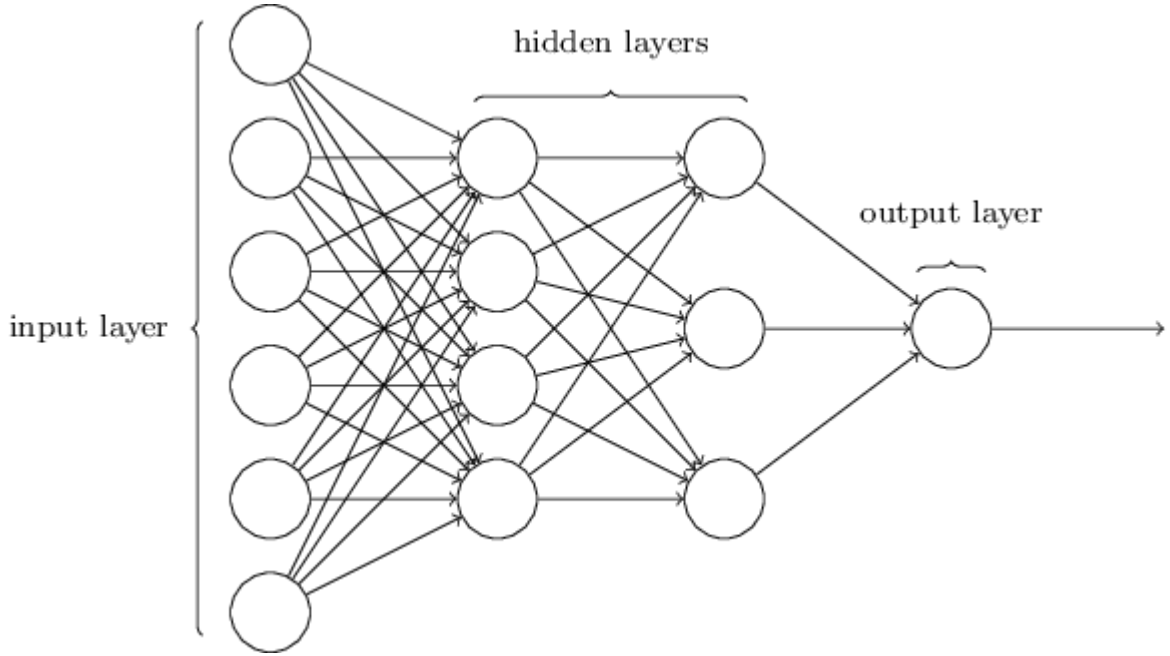
x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$

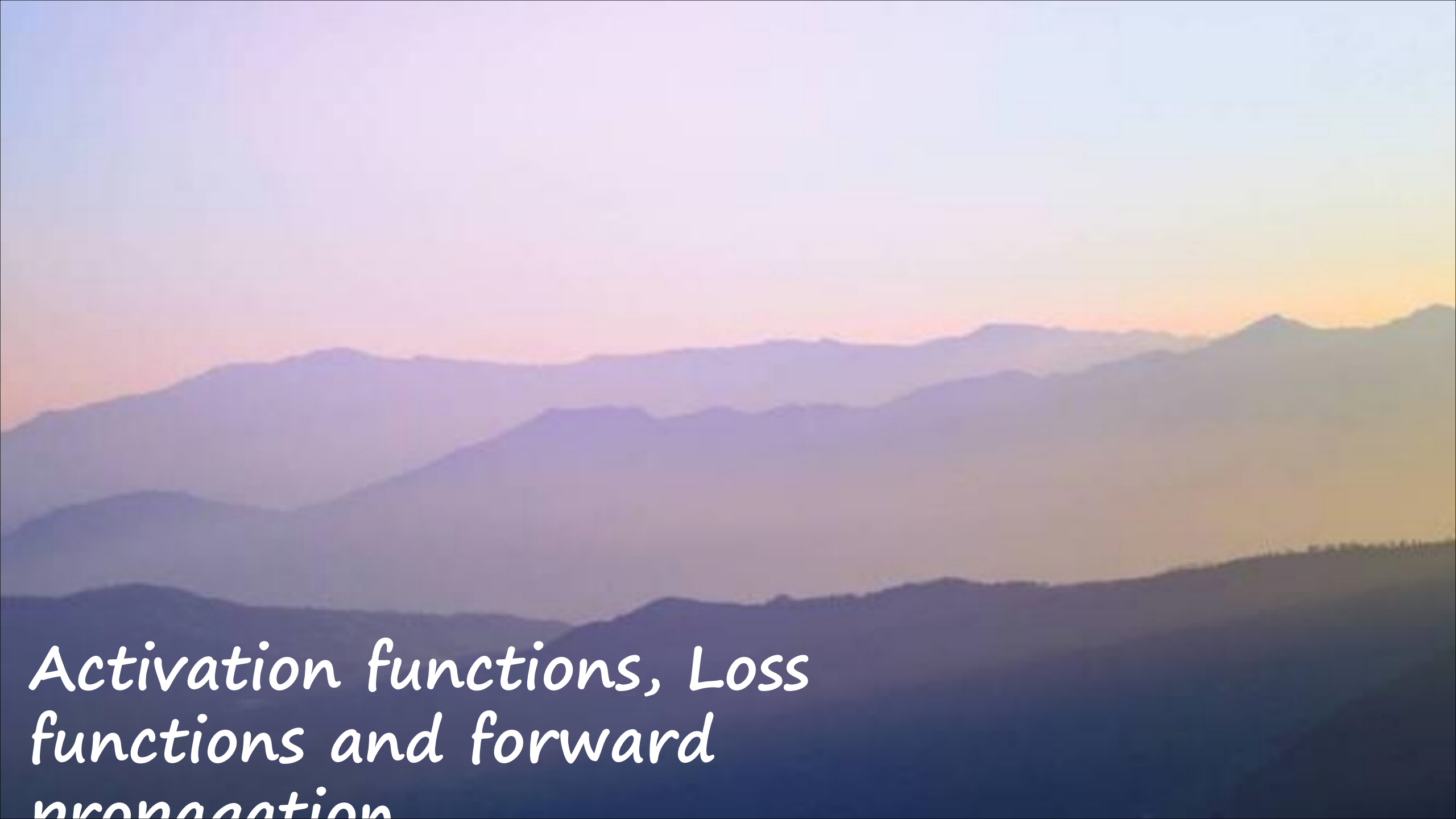
$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$
$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 \geq -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 \geq -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 < 0 \implies w_1 + w_2 < -w_0$$

Although a single perceptron cannot deal with this, a network of perceptrons can deal with a linear inseparable function

Multi-layer Perceptrons

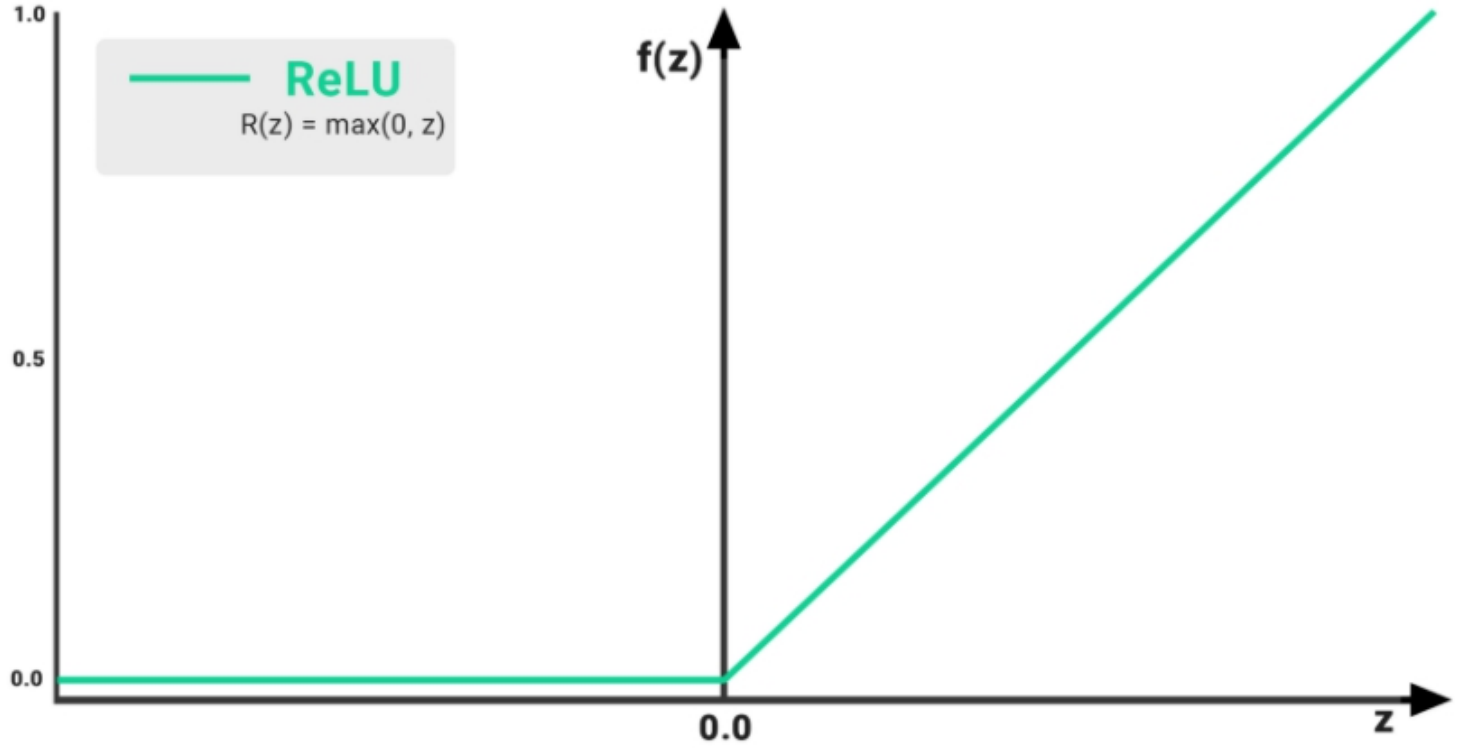
- Networks of perceptrons are which contain an input, output and one or more hidden layer are called Multilayer Perceptrons (MLP)



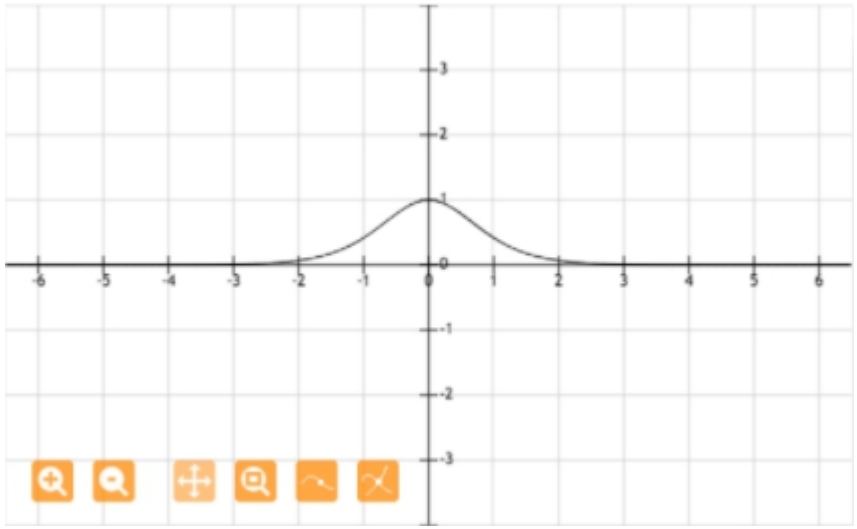

Activation functions, Loss
functions and forward
propagation

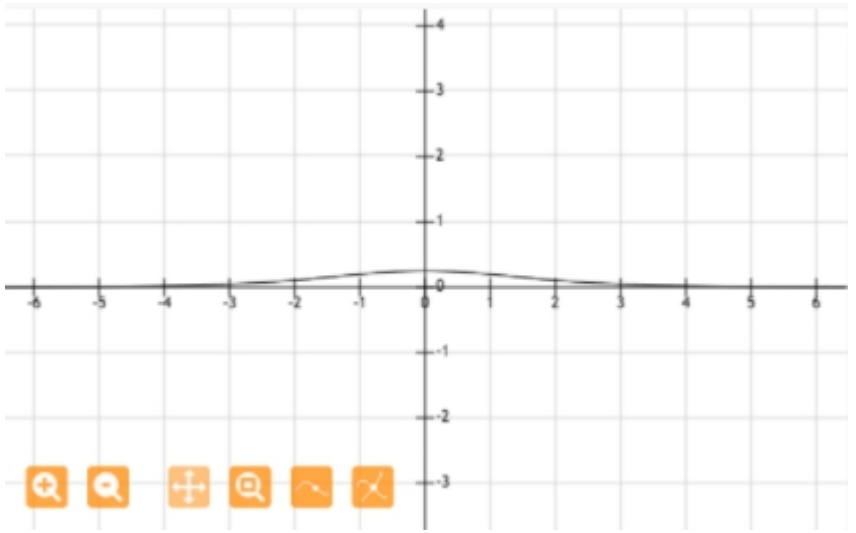
Activation function

- Activation function decides whether a neuron should be activated or not.
- An a unction.



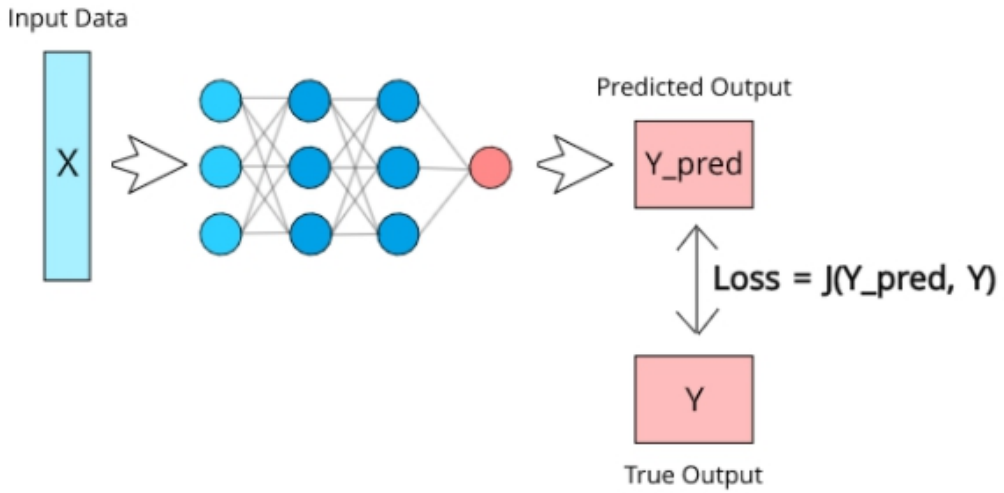
Tanh vs sigmoid derivatives





Loss Function

- The function we want to minimize or maximize is called the objective function or criterion. When we are minimizing it, we may also call it the cost function, loss function, or error function.
- It's a method of evaluating how well specific algorithm models the given datc



- Broadly loss functions can be classified into two major categories-

Regression and classification losses.

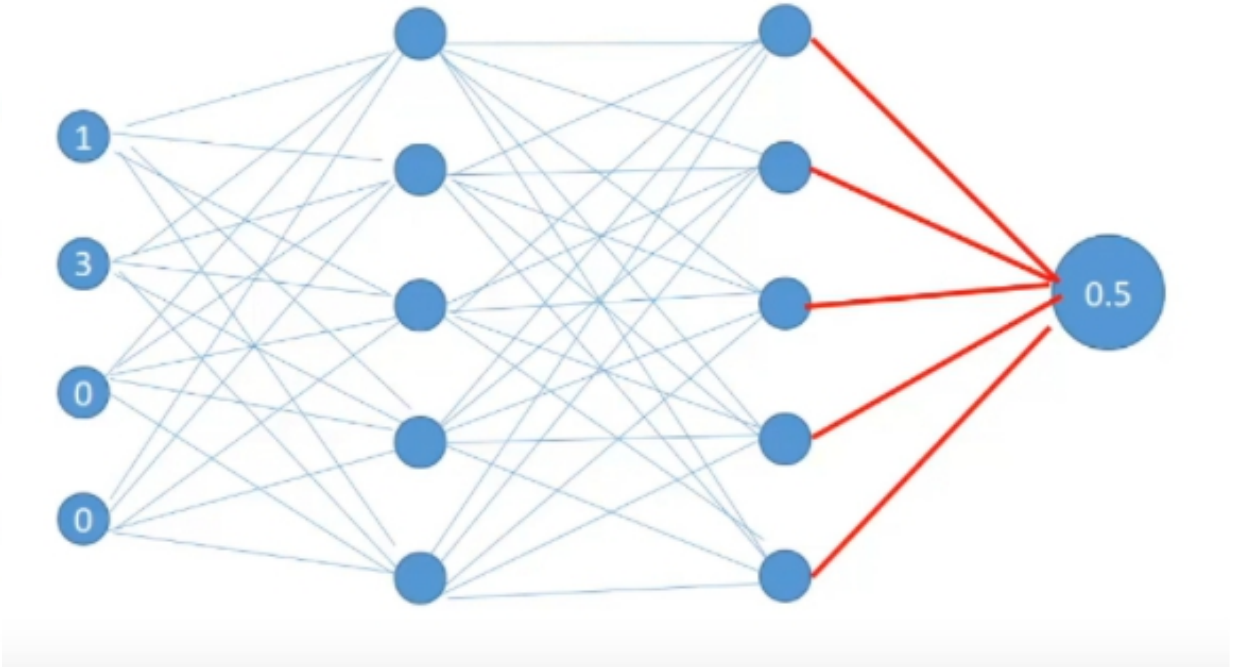
$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

Mean Squared Error

$$CrossEntropyLoss = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i))$$

Cross entropy loss

Forward propagation



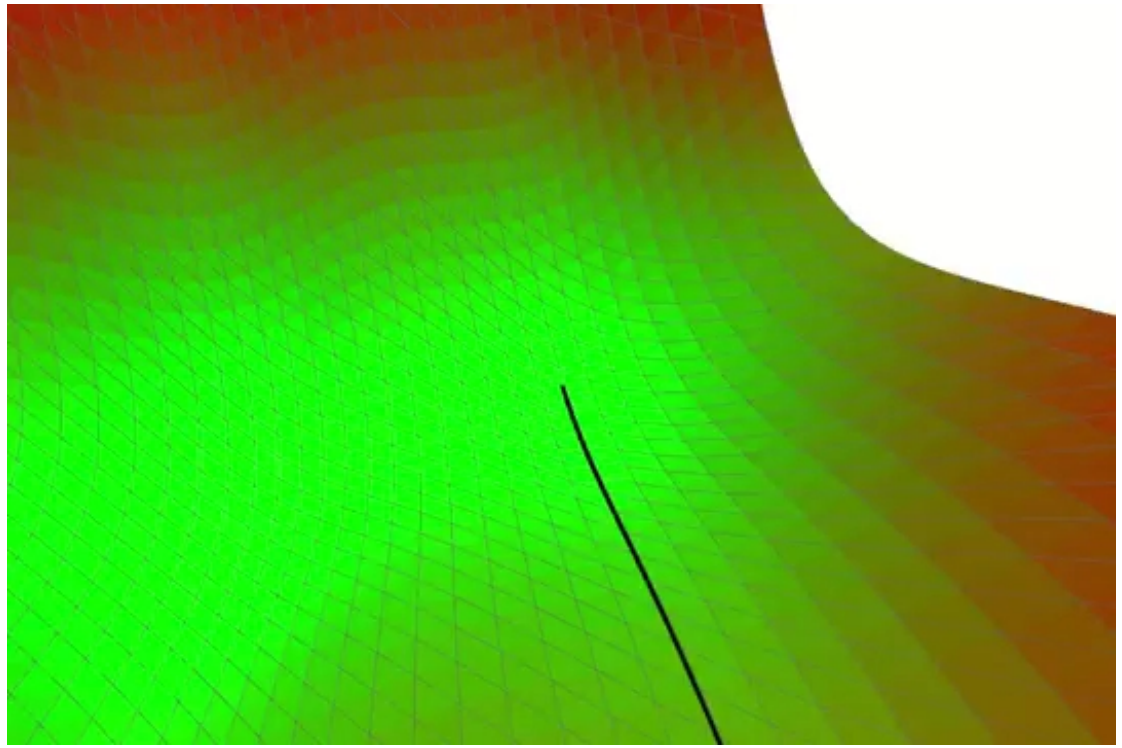
Gradient Descent

- Optimization algorithm capable of finding optimal solutions to a wide range of problems.

$$\theta = \theta - lr * (\partial(f(\theta)/\partial\theta)$$

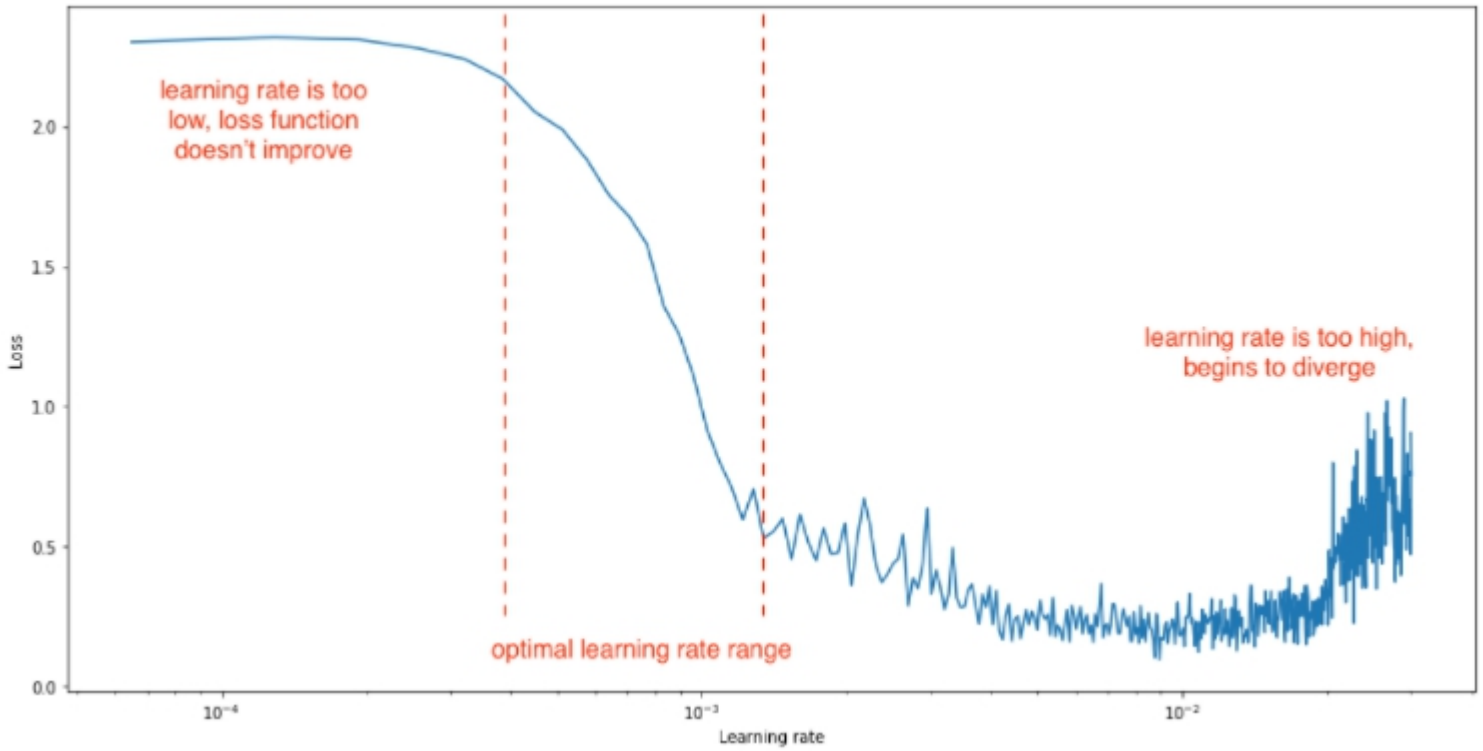
Gradient descent measures the local gradient of the error function with regards to the parameter vector θ, and it goes in the direction of descending gradient.

Once the gradient is zero, you have reached a minimum!



Learning Rate

- Learning rate is the tuning parameter that determines the step size at each step.



Sample, Epoch, Batch

- **Sample**- Single row of data. A sample may also be called an instance, an observation, an input vector, or a feature vector.
- **Batch**- Batch defines the number of samples to work through before updating the internal model parameters.
- **Epoch**- Epoch defines the number of times that the learning algorithm will work through the entire dataset.

Types of Gradient Descent

Batch Gradient Descent	Stochastic Gradient Descent (SGD)	Mini-Batch Gradient Descent
• Entire dataset for updation	• Single observation for updation	• Subset of data for updation
• Cost function reduces smoothly	• Lot of variations in cost function	• Smoother cost function as compared to SGD
• Computation cost is very high	• Computation time is more	• Computation time is lesser than SGD
		• Computation cost is lesser than Batch Gradient Descent

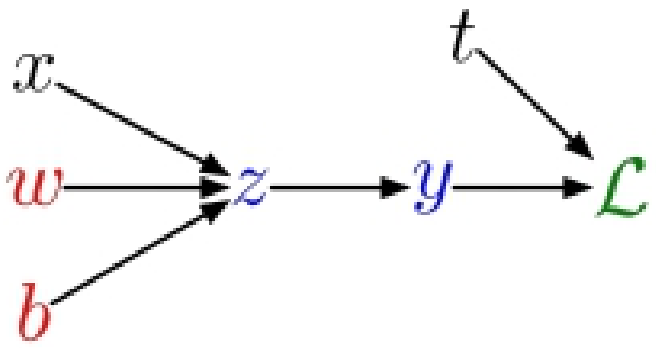
it

once!!!!

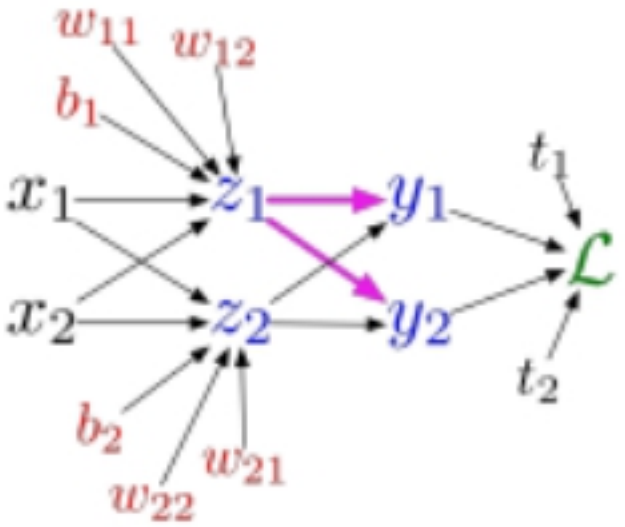

Back propagation

Chain rule

Univariate Chain Rule

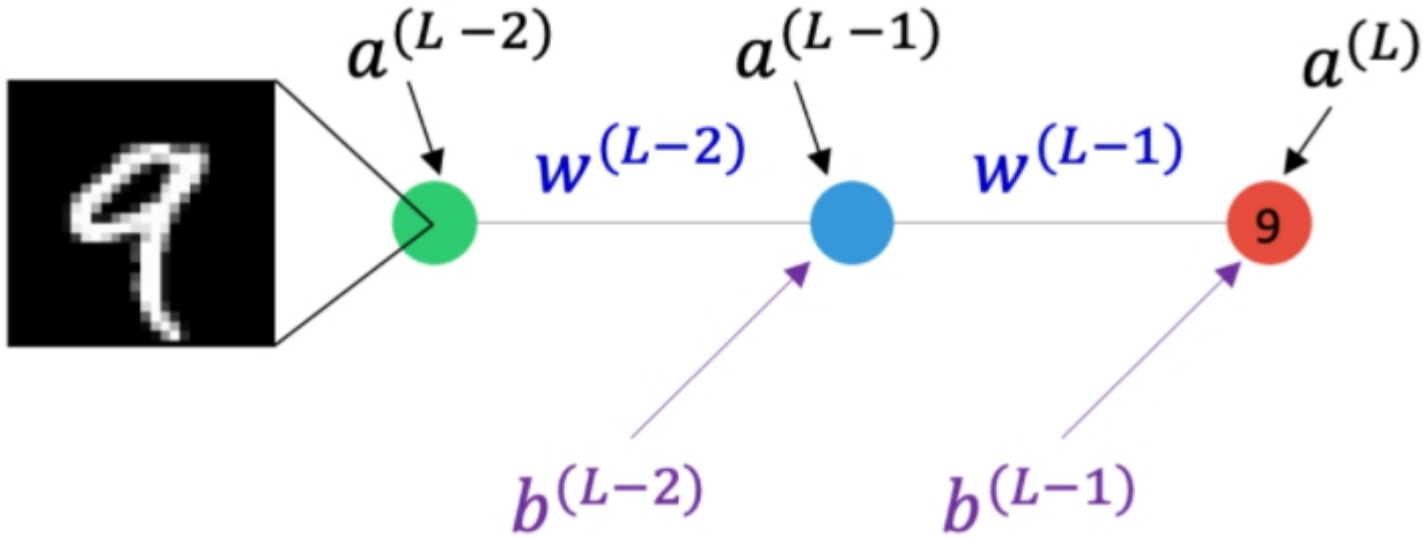


Multivariate Chain Rule



Back Propagation

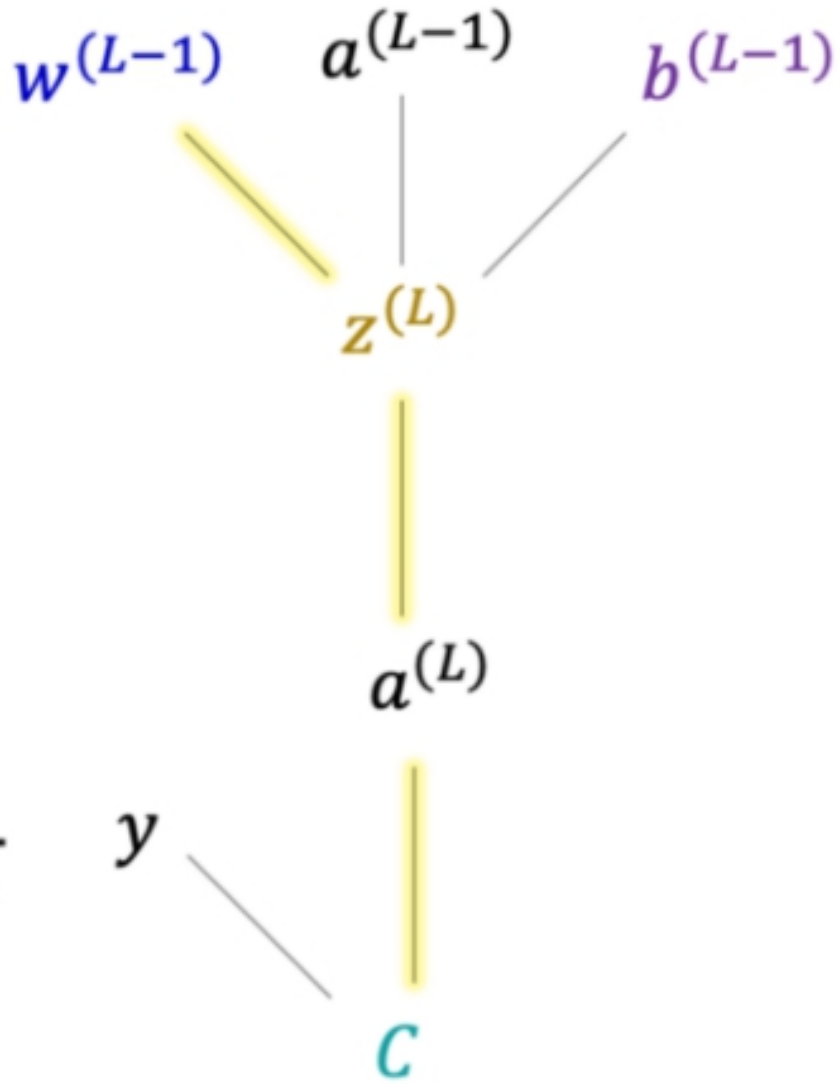
- Now let's look at the simplest case, with just a three layer neural network each having just one neuron.



What we want: $\frac{\partial C}{\partial w^{(L-1)}}$

Chain rule

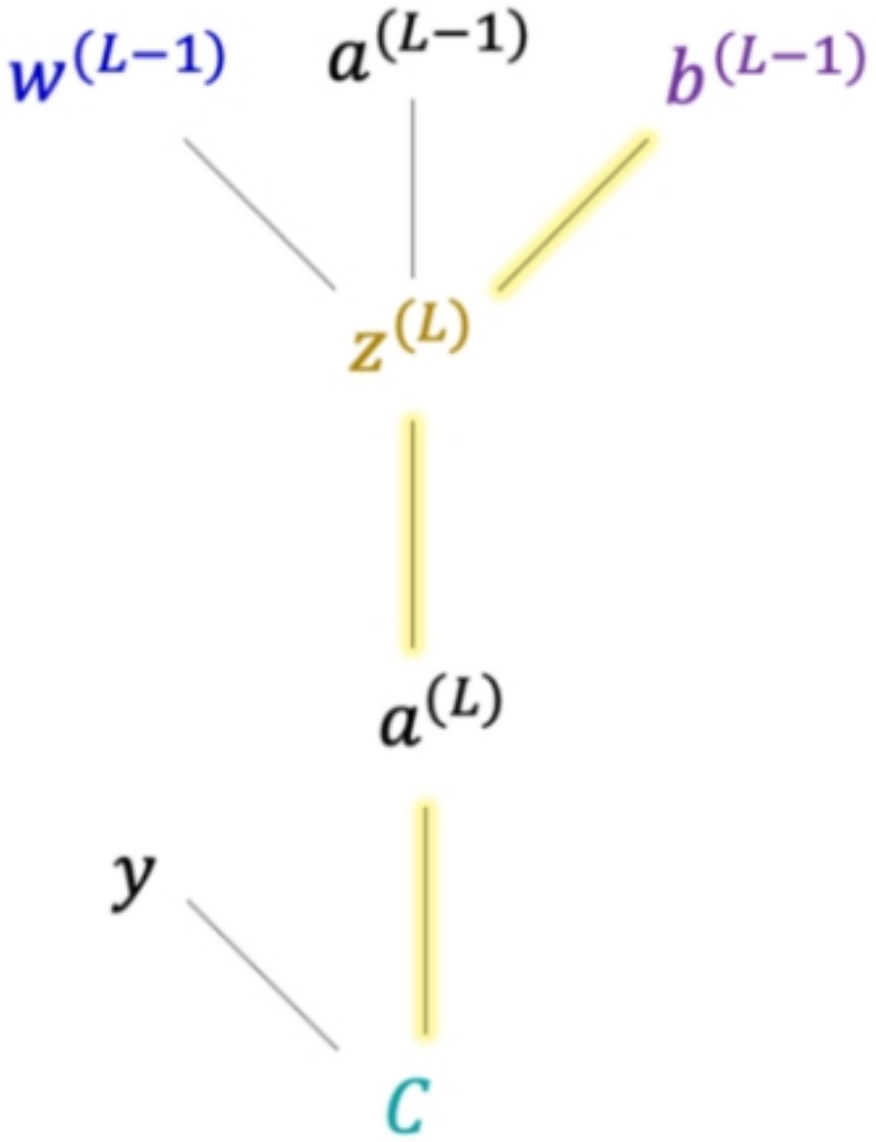
$$\frac{\partial C}{\partial w^{(L-1)}} = \frac{\partial z^{(L)}}{\partial w^{(L-1)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C}{\partial a^{(L)}}$$



What we want: $\frac{\partial C}{\partial b^{(L-1)}}$

Chain rule

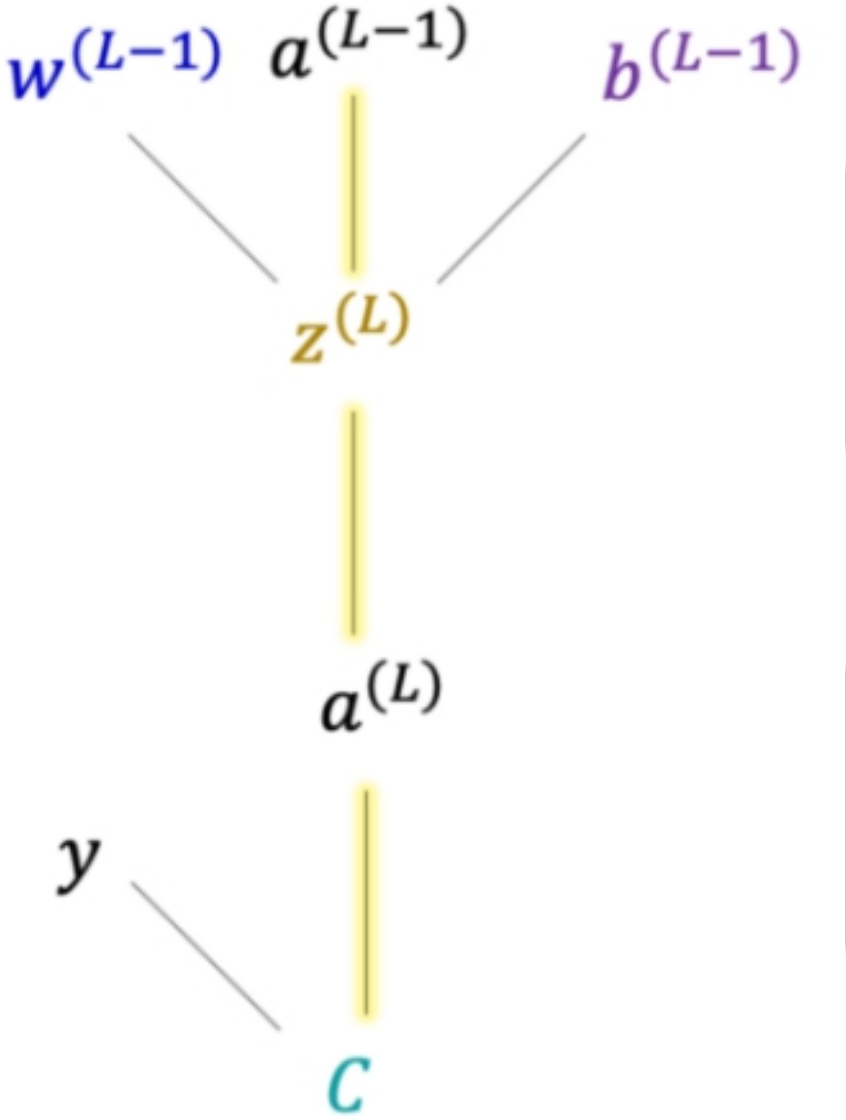
$$\frac{\partial C}{\partial b^{(L-1)}} = \frac{\partial z^{(L)}}{\partial b^{(L-1)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C}{\partial a^{(L)}}$$



What we want: $\frac{\partial C}{\partial a^{(L-1)}}$

Chain rule

$$\frac{\partial C}{\partial a^{(L-1)}} = \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C}{\partial a^{(L)}}$$

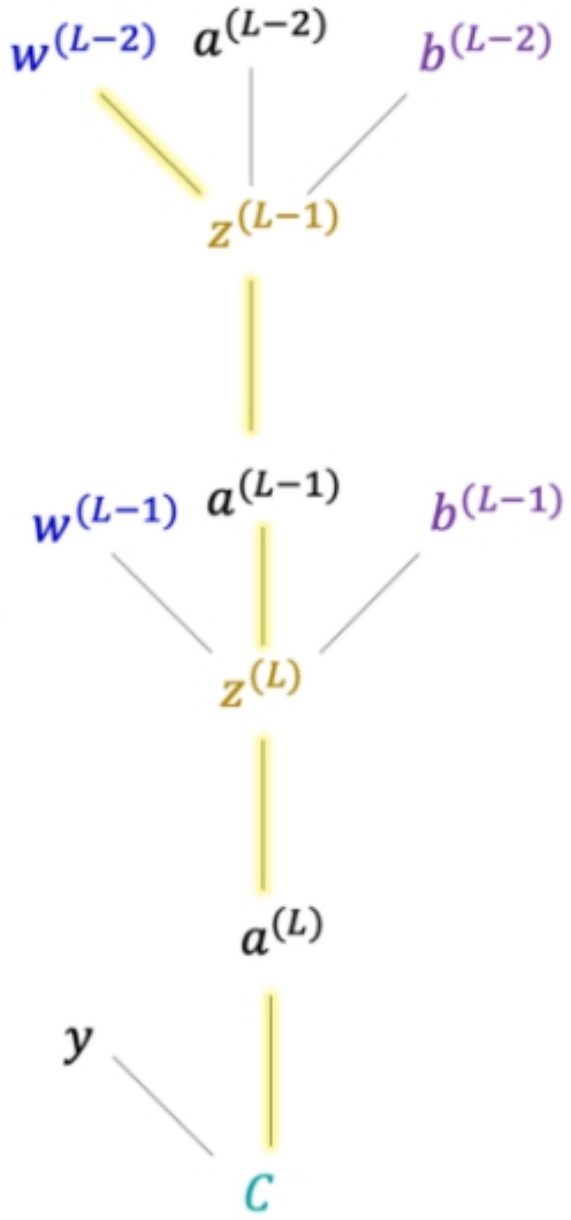


What we want: $\frac{\partial C}{\partial w^{(L-2)}}$

Chain rule

$$\frac{\partial C}{\partial w^{(L-2)}} = \frac{\partial z^{(L-1)}}{\partial w^{(L-2)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \underbrace{\frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C}{\partial a^{(L)}}}$$

$$\frac{\partial C}{\partial w^{(L-2)}} = \frac{\partial z^{(L-1)}}{\partial w^{(L-2)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial C}{\partial a^{(L-1)}}$$



$$\frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} = \sigma'\left(z^{(L-1)}\right)$$

$$\frac{\partial z^{(L-1)}}{\partial w^{(L-2)}} = a^{(L-2)}$$

$$\frac{\partial C}{\partial w^{(L-2)}} = \frac{\partial z^{(L-1)}}{\partial w^{(L-2)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial C}{\partial a^{(L-1)}} = a^{(L-2)} \sigma'\left(z^{(L-1)}\right) \frac{\partial C}{\partial a^{(L-1)}}$$

$$\frac{\partial C}{\partial w_{jk}^{(L-2)}} = \frac{\partial z_{jk}^{(L-1)}}{\partial w_{jk}^{(L-2)}} \frac{\partial a_j^{(L-1)}}{\partial z_j^{(L-1)}} \frac{\partial C}{\partial a_j^{(L-1)}}$$

$$\frac{\partial C}{\partial w_{jk}^{(L-2)}} = \frac{\partial z_{jk}^{(L-1)}}{\partial w_{jk}^{(L-2)}} \frac{\partial a_j^{(L-1)}}{\partial z_j^{(L-1)}} \sum_{j=1}^{N_L} \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial C}{\partial a_j^{(L)}}$$