Neural Network Tutorial

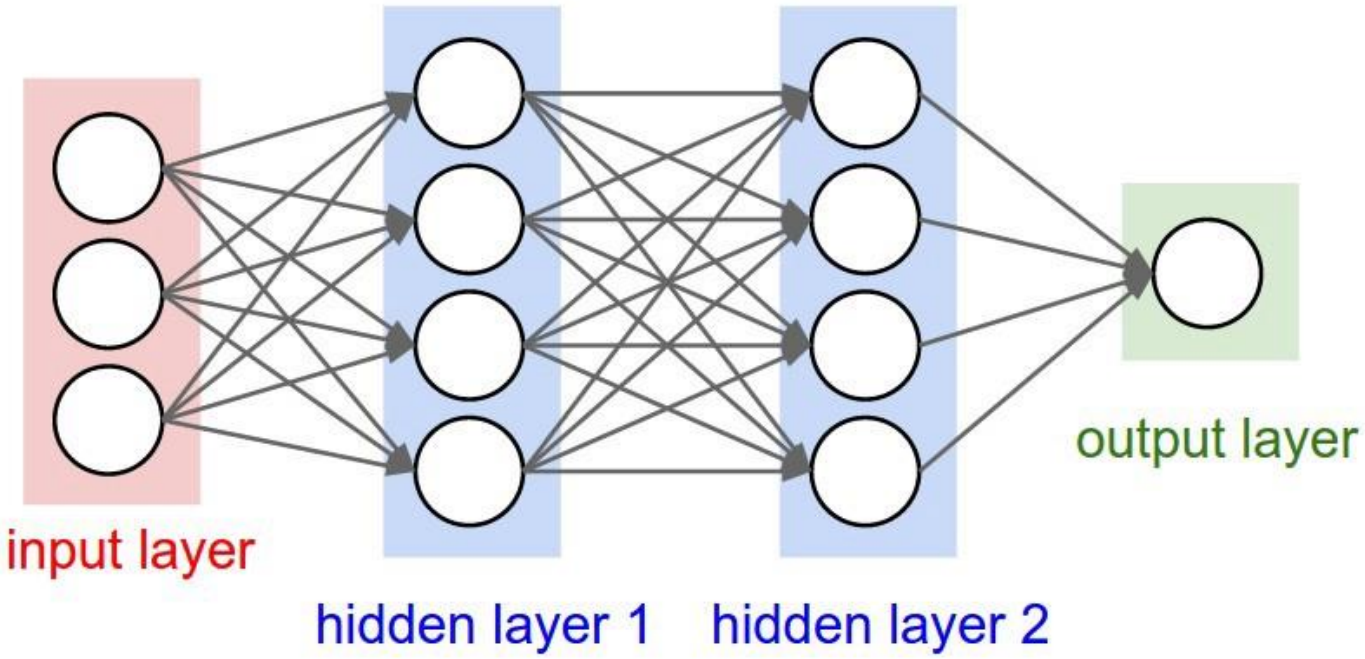
IKM Laboratory

Outline

- **Introduction to NN**
    - **Feedforward**
        - Fully Connect Layer
        - Activation Function
    - **Backpropagation**
        - Lost/Cost Function
        - Optimization
- **Simple text classification model**
    - **Data Preprocessing**
    - **Vectorization**
    - **Model building**

Neural Network

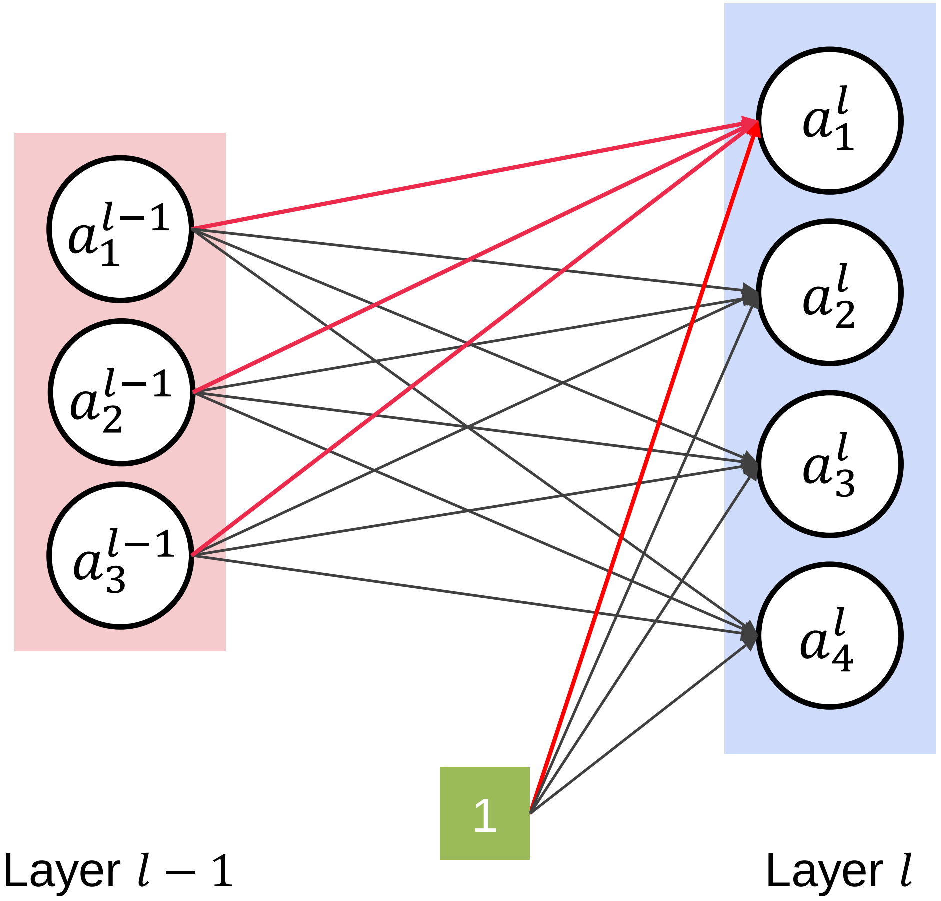
A neural network is an artificial neural network wherein connections between the nodes do not form a cycle. As such, it is different from recurrent neural networks. It is illustrated in the following figure



Neural Network

Fully connected layer

All neurons in this layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.



W_{ij}^l $\longrightarrow$ Layer $l-1$ to Layer l

$$a_1^l = f(W_{11}^l a_1^{l-1} + W_{12}^l a_2^{l-1} + W_{13}^l a_3^{l-1} + b_1^l)$$

We can define each neural in current layer by :

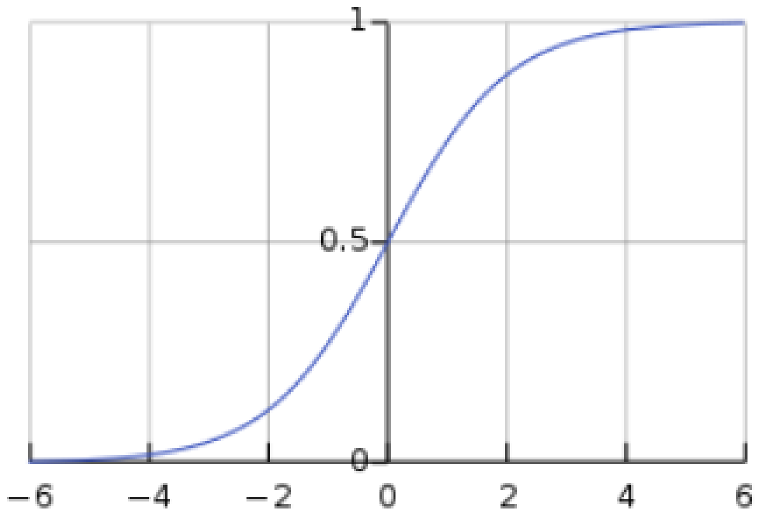
$$a_i^l = f(\sum_{j=1}^{N_{l-1}} W_{ij}^l a_j^{l-1} + b_i^l)$$

f is activation function

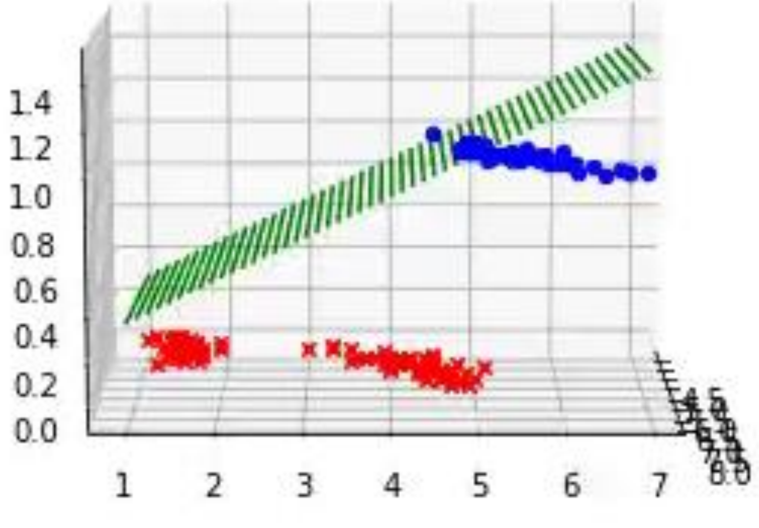
Neural Network

Activation Function

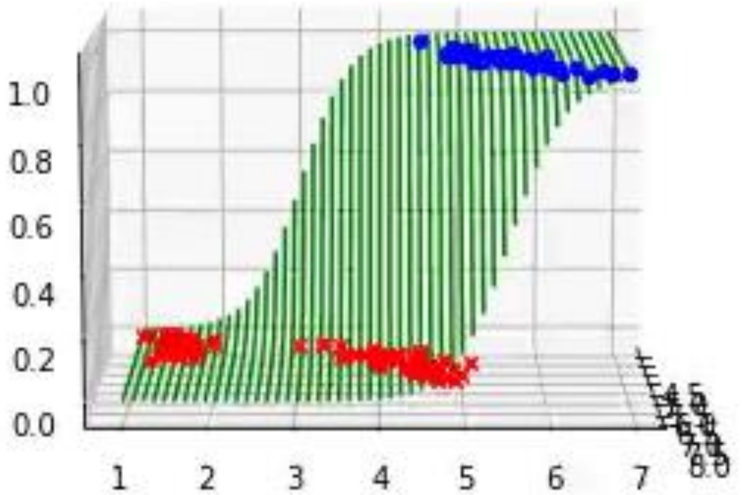
It’s just a thing that you add to the output end of any neural network. It is also known as **Transfer Function**. It can also be attached in between two Neural Networks. It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 (sigmoid) whatever the inputs are.



Sigmoid function



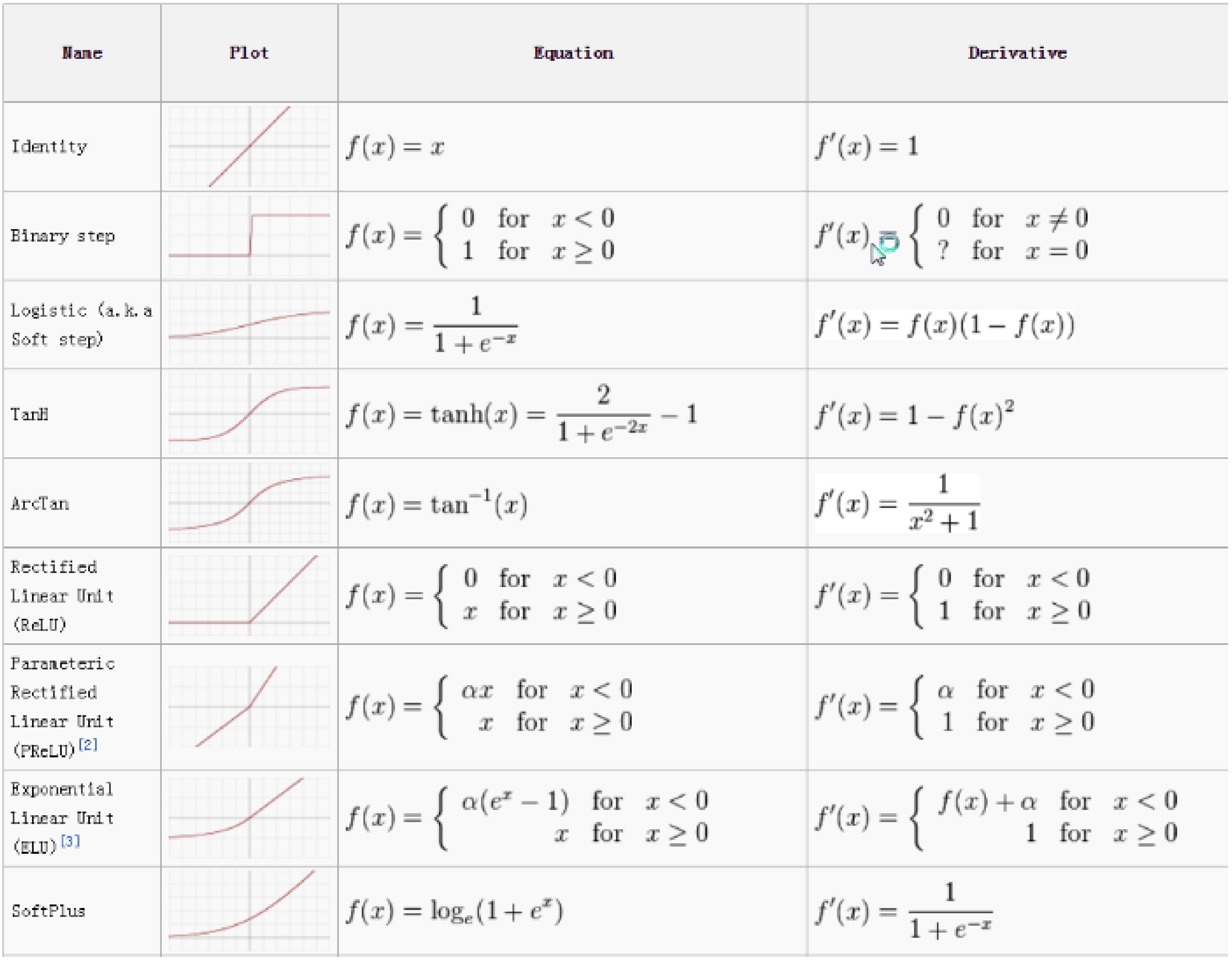
No activation



Activation

Neural Network

Different Activation Function

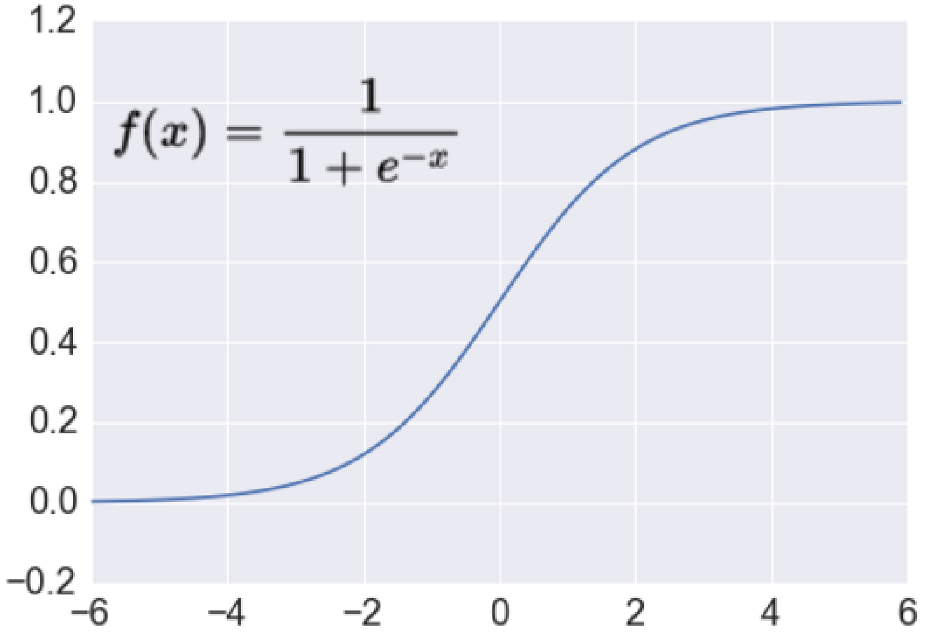


Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Neural Network

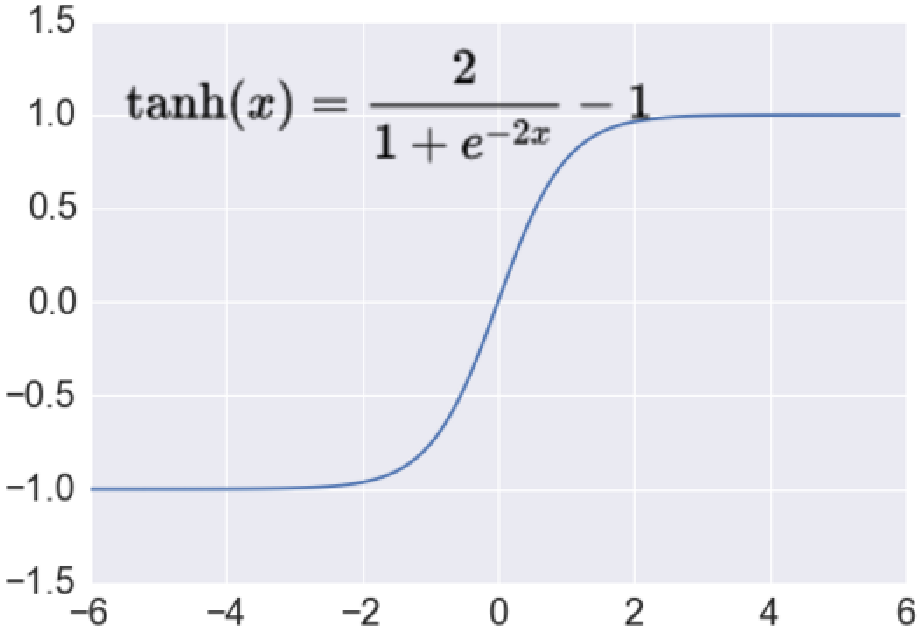
Commonly used activation function

Sigmoid



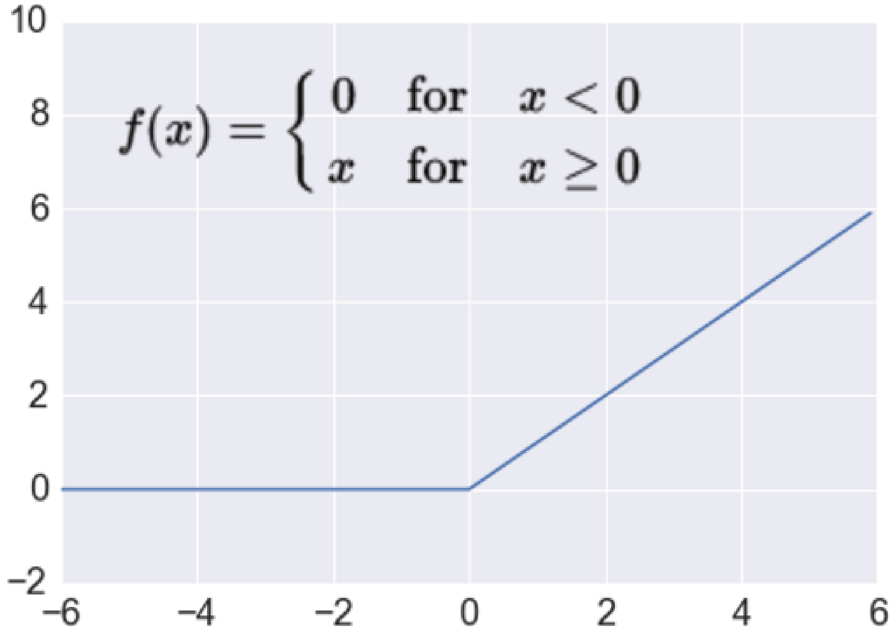
The main reason why we use sigmoid function is because it exists between (0 to 1). Therefore, it is especially used for models where we have to **predict the probability as an output**.

TanH



The range of the Tanh function is from (-1 to 1). Tanh is also sigmoidal (s - shaped). **The advantage is that the negative inputs will be mapped strongly negative** and the zero inputs will be mapped near zero in the tanh graph.

ReLU

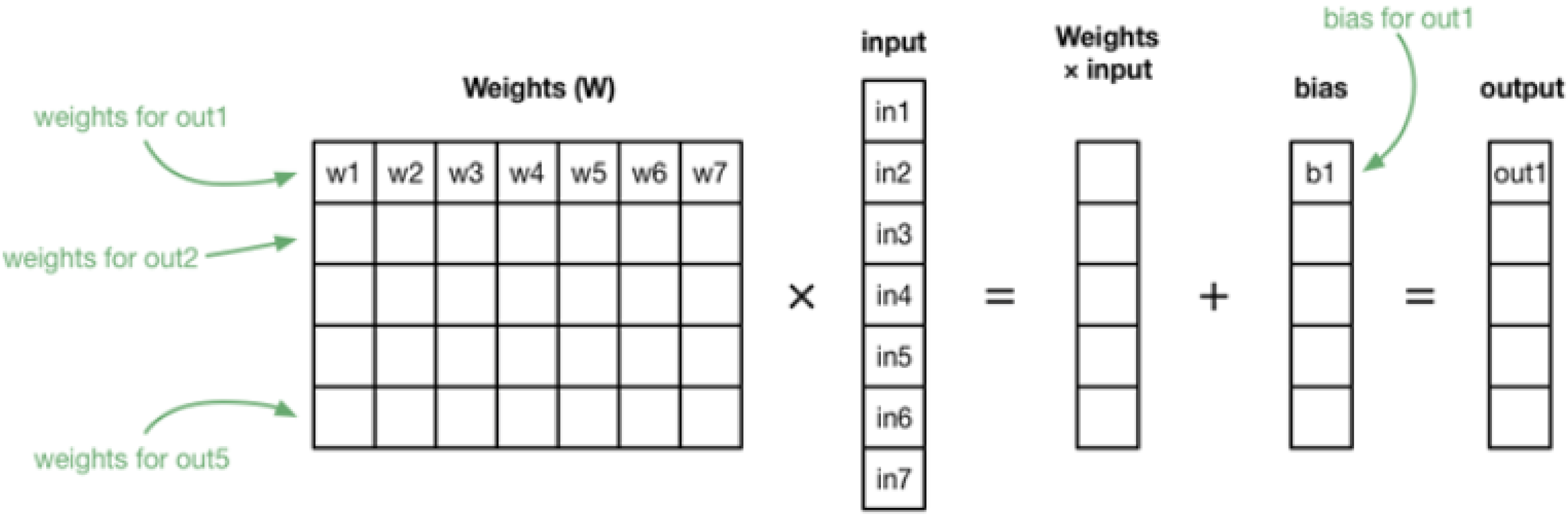


The ReLU is **the most used activation function** in the world right now. Since, it is used in almost all the convolutional neural networks or deep learning.

Neural Network

Feedforward

The formula can be implement by **two multiplied matrices** and **plus a bias vector** like this:



Neural Network

Backward pass

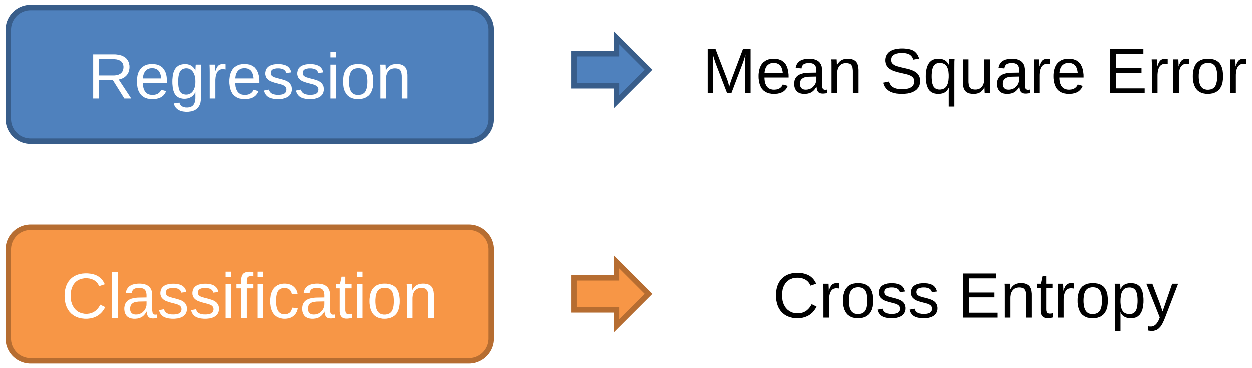
How did we set our weights in forward pass ? We must define a **loss function** to decide what weights can decrease difference of outputs and label.

Loss function

In most learning networks, error is calculated as the difference between the actual output and the predicted output.

$$J(w) = p - \widehat{p}$$

Different loss functions are used to deal with different type of tasks, for example



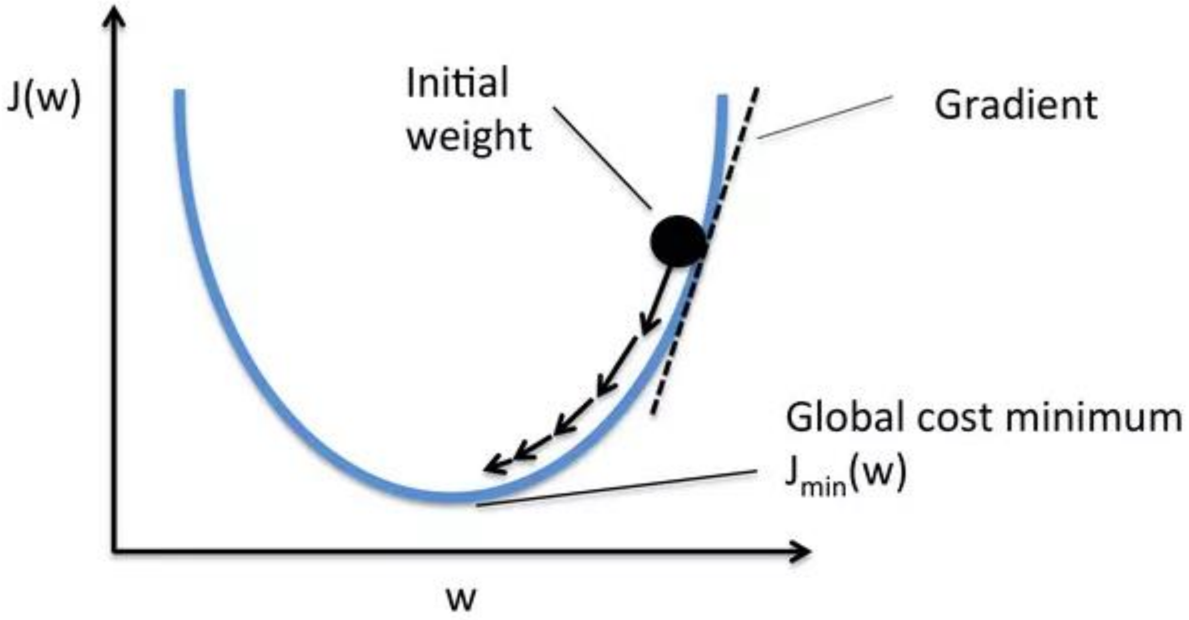
Regression → Mean Square Error

Classification → Cross Entropy

Neural Network

Optimisation

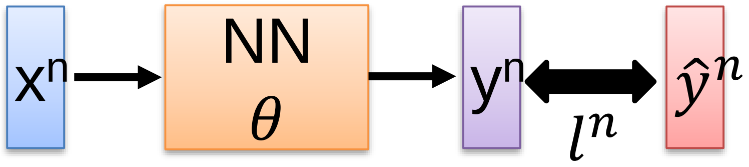
Error J(w) is a function of internal parameters of model i.e. weights and bias. For accurate predictions, one needs to minimize the calculated error.



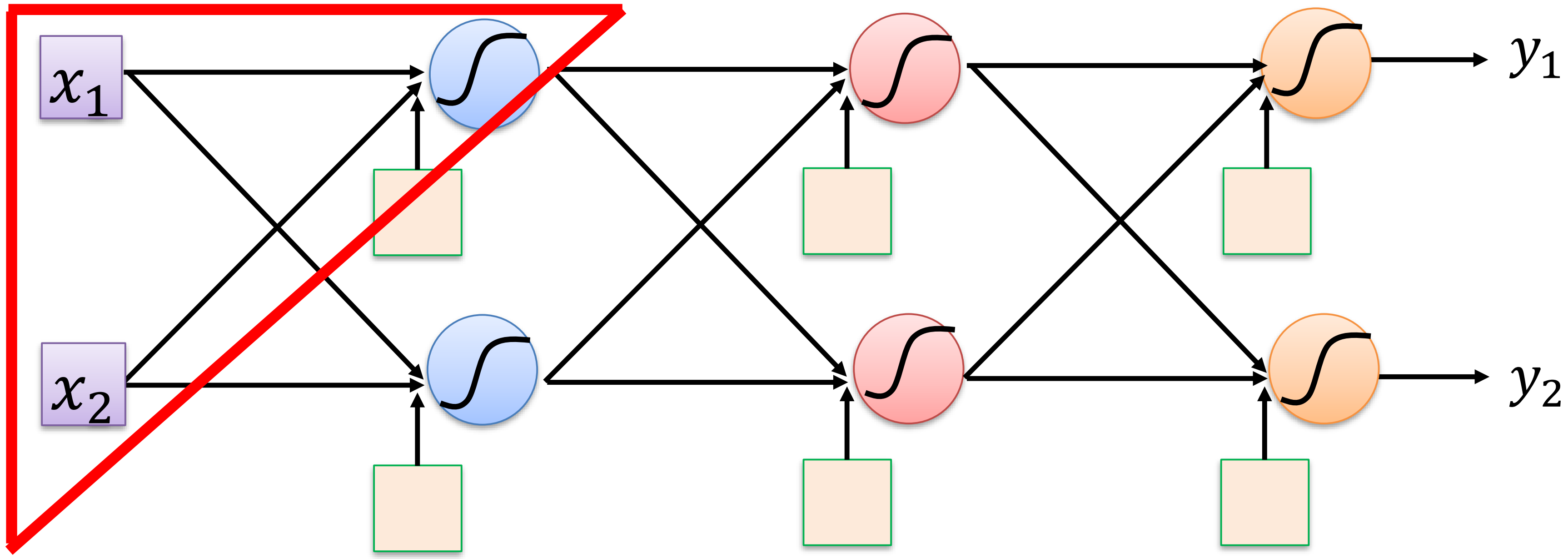
Optimisation usually calculate the gradient i.e. the partial derivative of loss function with respect to weights, and the weights are modified in the opposite direction of the calculated gradient. This cycle is repeated until we reach the minima of loss function.

$$w_{new} = w - \eta \Delta w$$

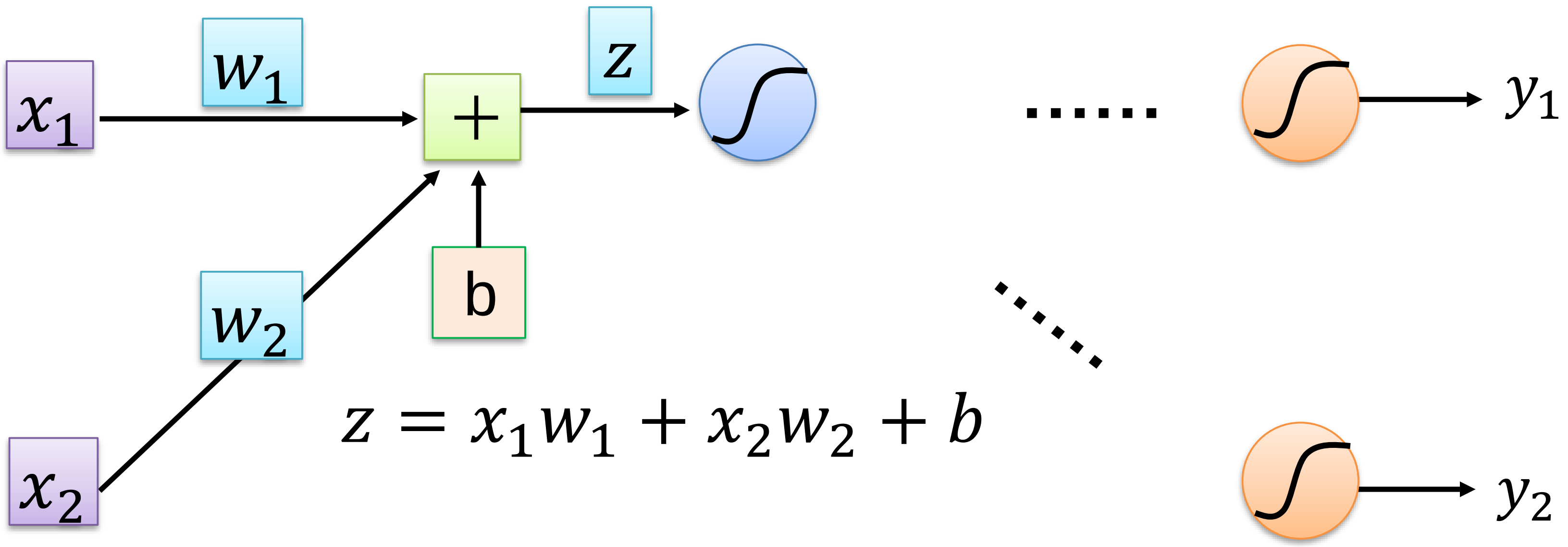
Neural Network



$$L(\theta) = \sum_{n=1}^{N} l^n(\theta) \quad \Rightarrow \quad \frac{\partial L(\theta)}{\partial w} = \sum_{n=1}^{N} \frac{\partial l^n(\theta)}{\partial w}$$



Neural Network



$$z = x_1 w_1 + x_2 w_2 + b$$

$$\frac{\partial l}{\partial w} =? \quad \frac{\partial z}{\partial w}\frac{\partial l}{\partial z}$$

(Chain rule)

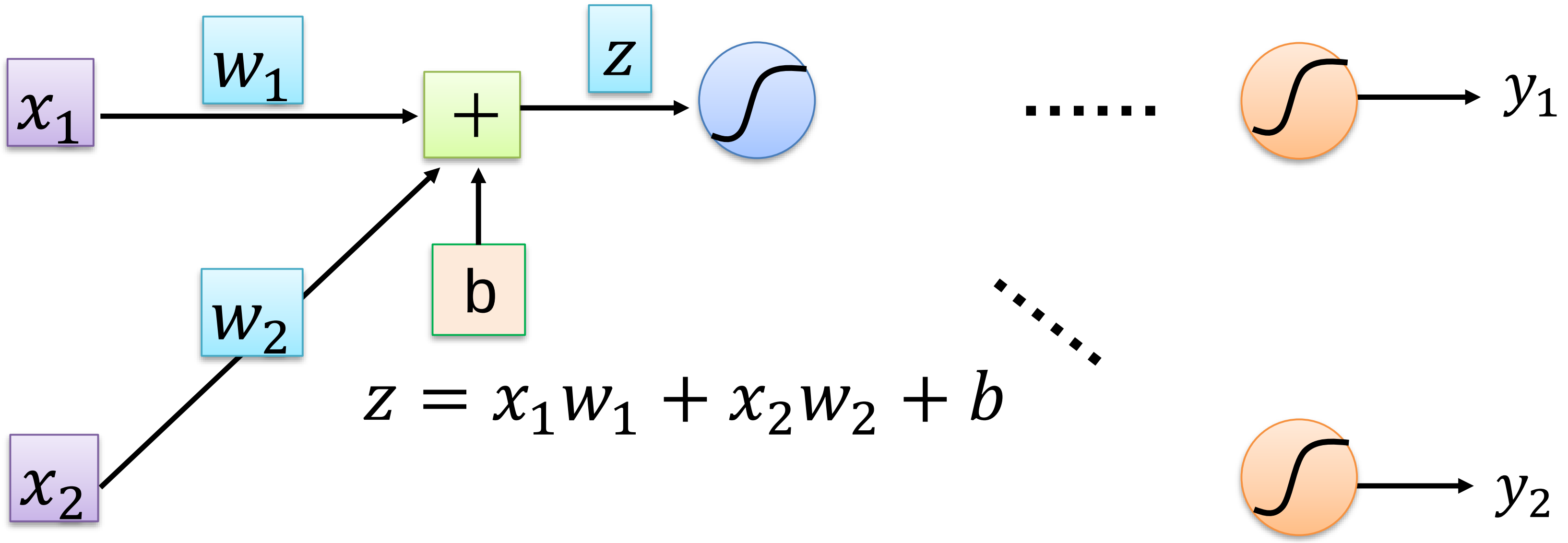
Forward pass:

Compute $\partial z / \partial w$ for all parameters

Backward pass:

Compute $\partial l / \partial z$ for all activation function inputs z

Neural Network



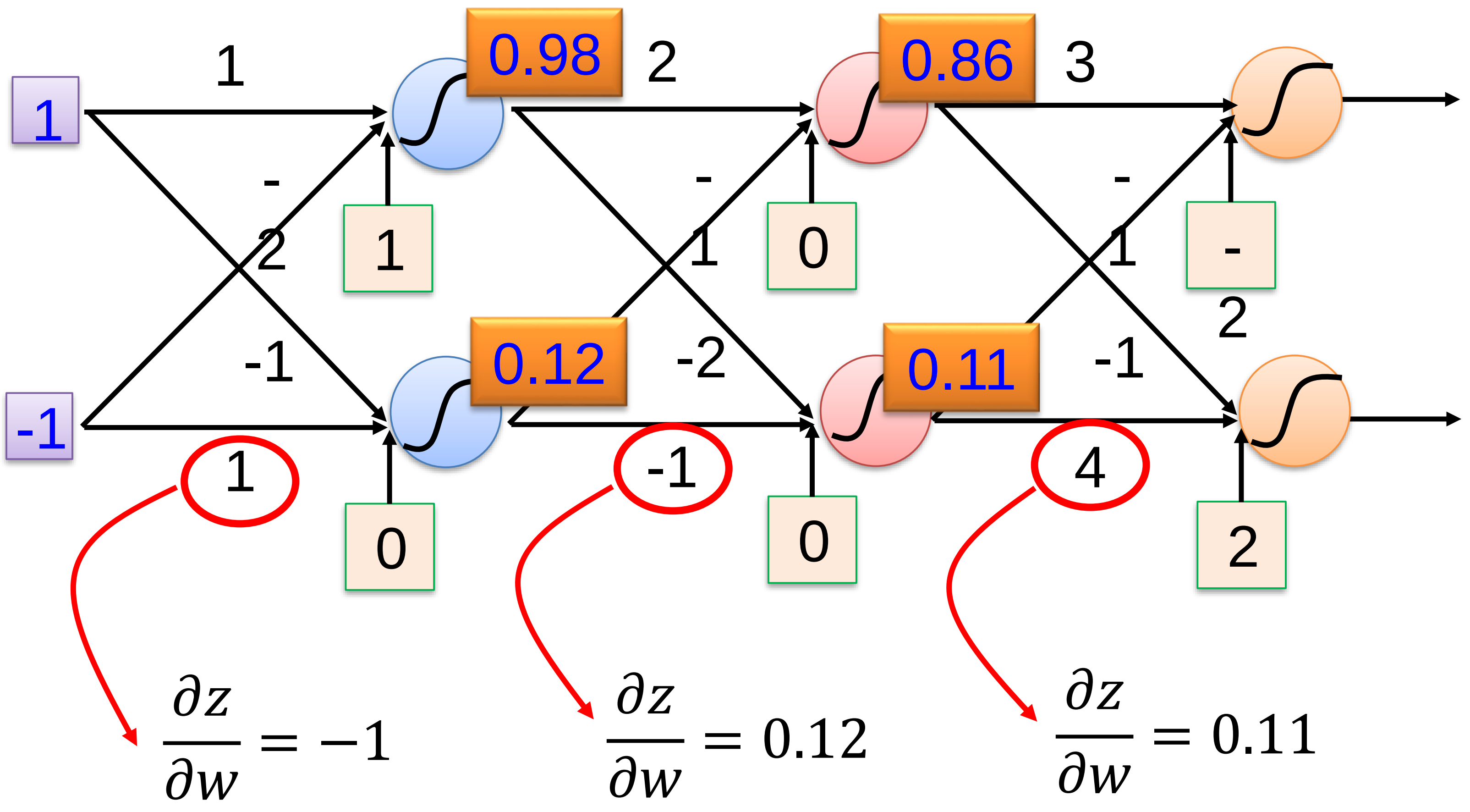
$$z = x_1 w_1 + x_2 w_2 + b$$

$$\left.\begin{array}{l}\partial z/\partial w_1 =?\ \ x_1\\ \partial z/\partial w_2 =?\ \ x_2\end{array}\right\}$$

The value of the input connected by the weight

Neural Network

Compute $\partial z / \partial w$ for all parameters



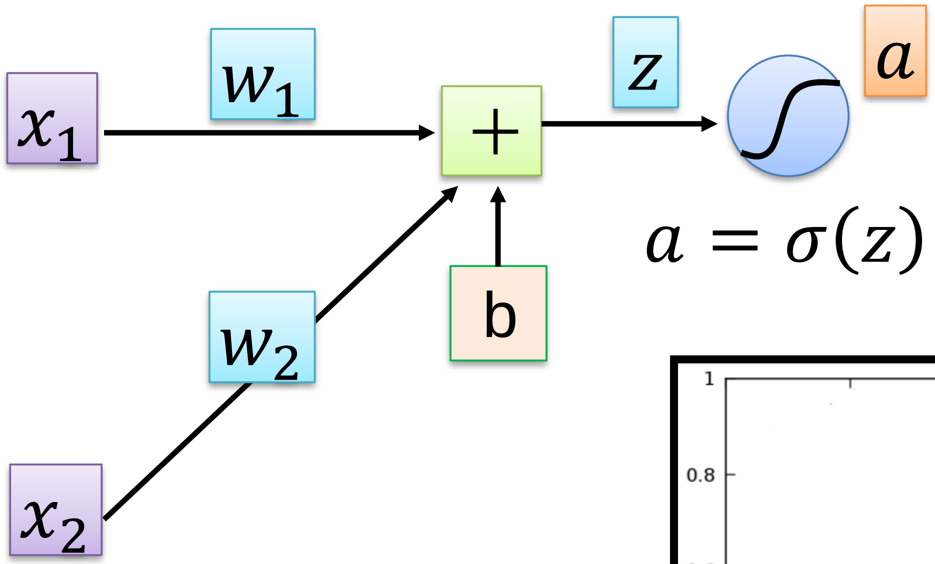
$$\frac{\partial z}{\partial w} = -1$$

$$\frac{\partial z}{\partial w} = 0.12$$

$$\frac{\partial z}{\partial w} = 0.11$$

Neural Network

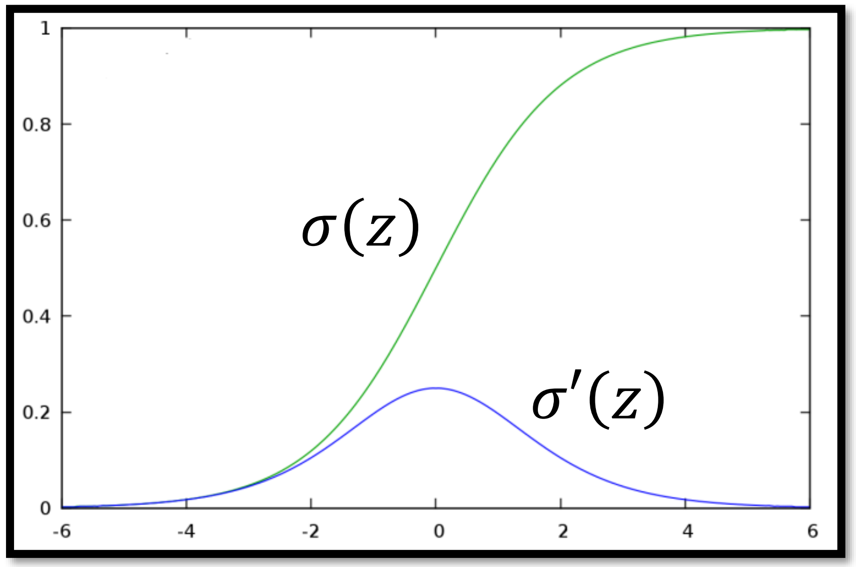
Compute $\partial l/\partial z$ for all activation function inputs z



$$a = \sigma(z)$$

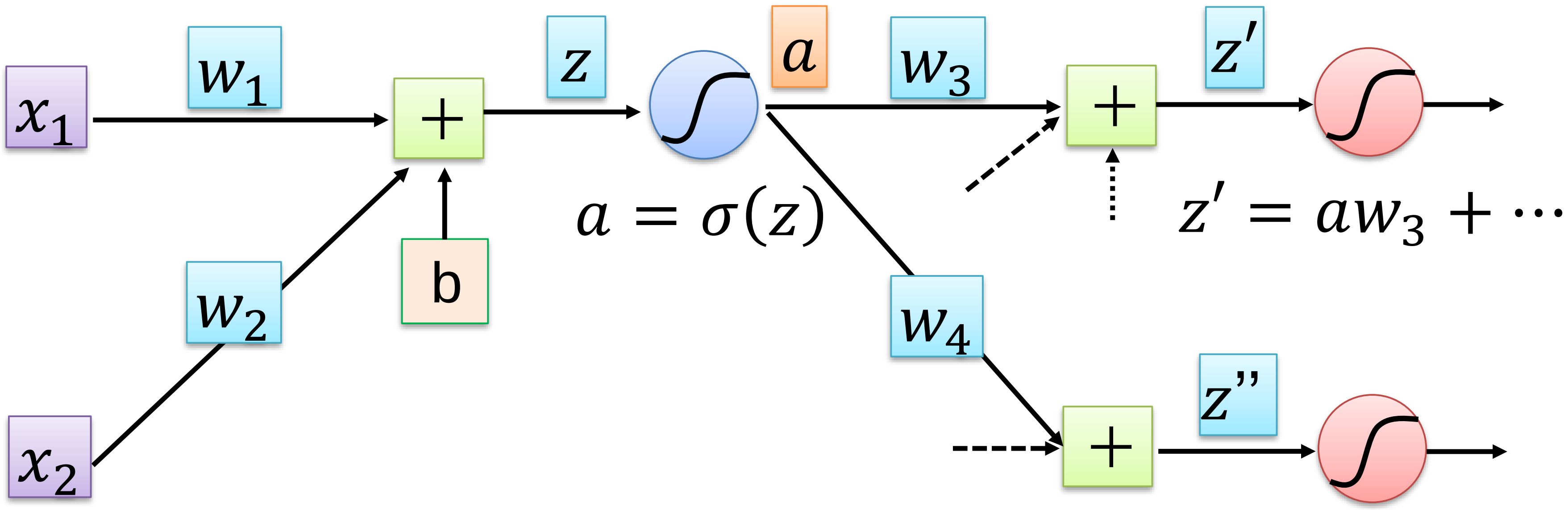
$$\frac{\partial l}{\partial z} = \underline{\frac{\partial a}{\partial z}}\,\frac{\partial l}{\partial a}$$

➡ $\sigma'(z)$



Neural Network

Compute $\partial l/\partial z$ for all activation function inputs z



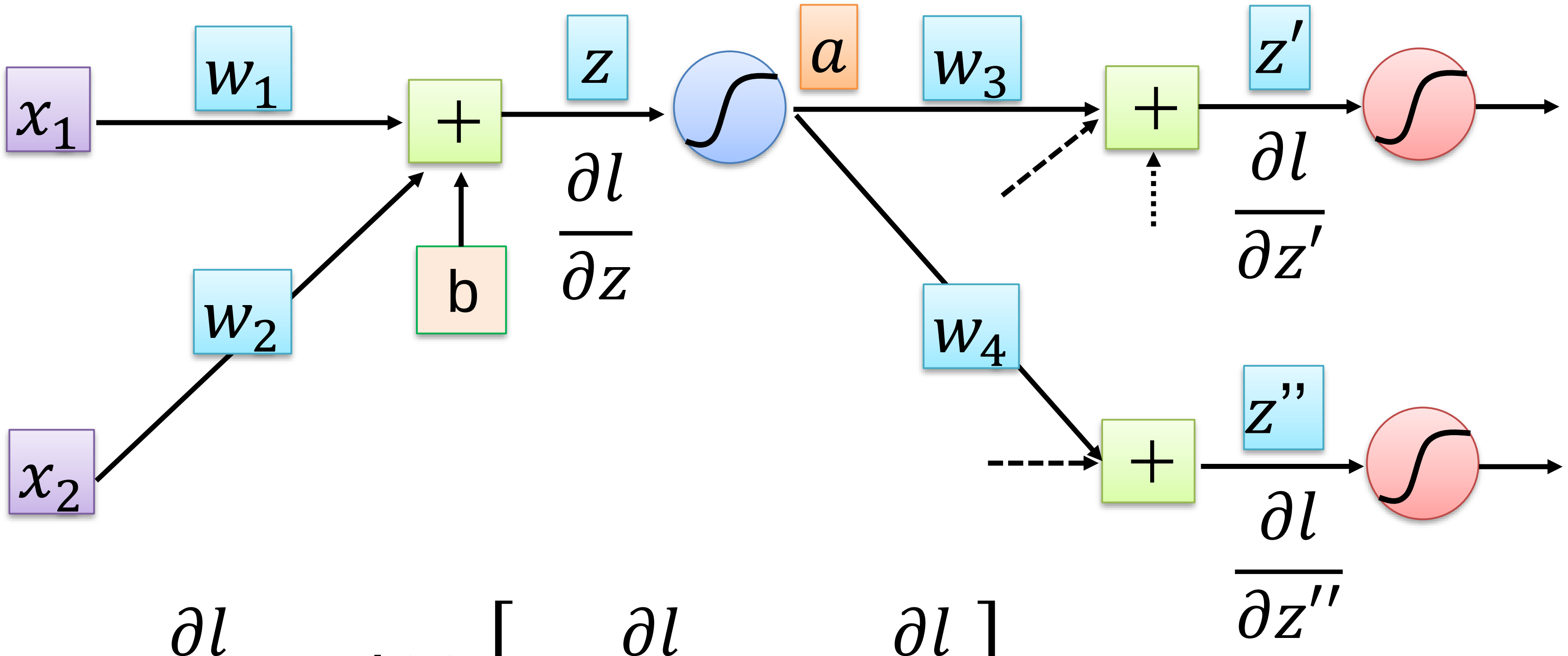
$$\frac{\partial l}{\partial z} = \frac{\partial a}{\partial z}\underline{\frac{\partial l}{\partial a}} \qquad \frac{\partial l}{\partial a} = \underbrace{\frac{\partial z'}{\partial a}}_{w_3}\underbrace{\frac{\partial l}{\partial z'}}_{?} + \underbrace{\frac{\partial z''}{\partial a}}_{w_4}\underbrace{\frac{\partial l}{\partial z''}}_{?}$$

(Chain rule)

Assumed it's known

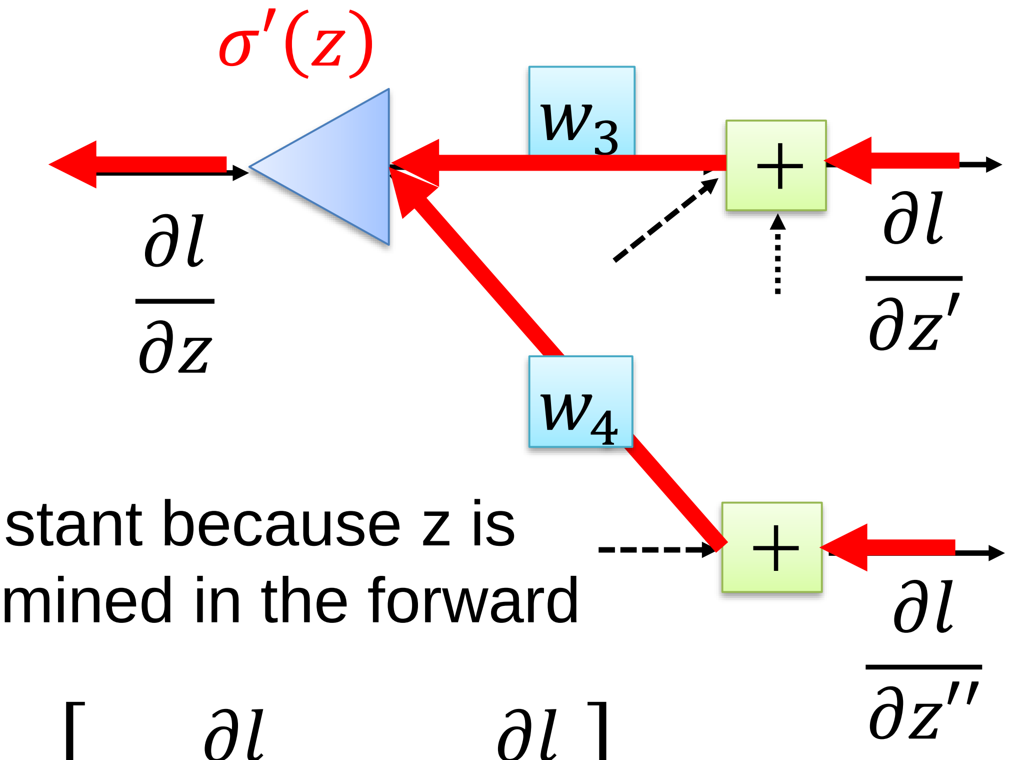
Neural Network

Compute $\partial l / \partial z$ for all activation function inputs z


x_1
w_1
$+$
z
a
w_3
$+$
z'
x_2
w_2
b
$\frac{\partial l}{\partial z}$
$\frac{\partial l}{\partial z'}$
w_4
$+$
z''
$\frac{\partial l}{\partial z''}$

$$\frac{\partial l}{\partial z} = \sigma'(z)\left[w_3 \frac{\partial l}{\partial z'} + w_4 \frac{\partial l}{\partial z''}\right]$$

Neural Network

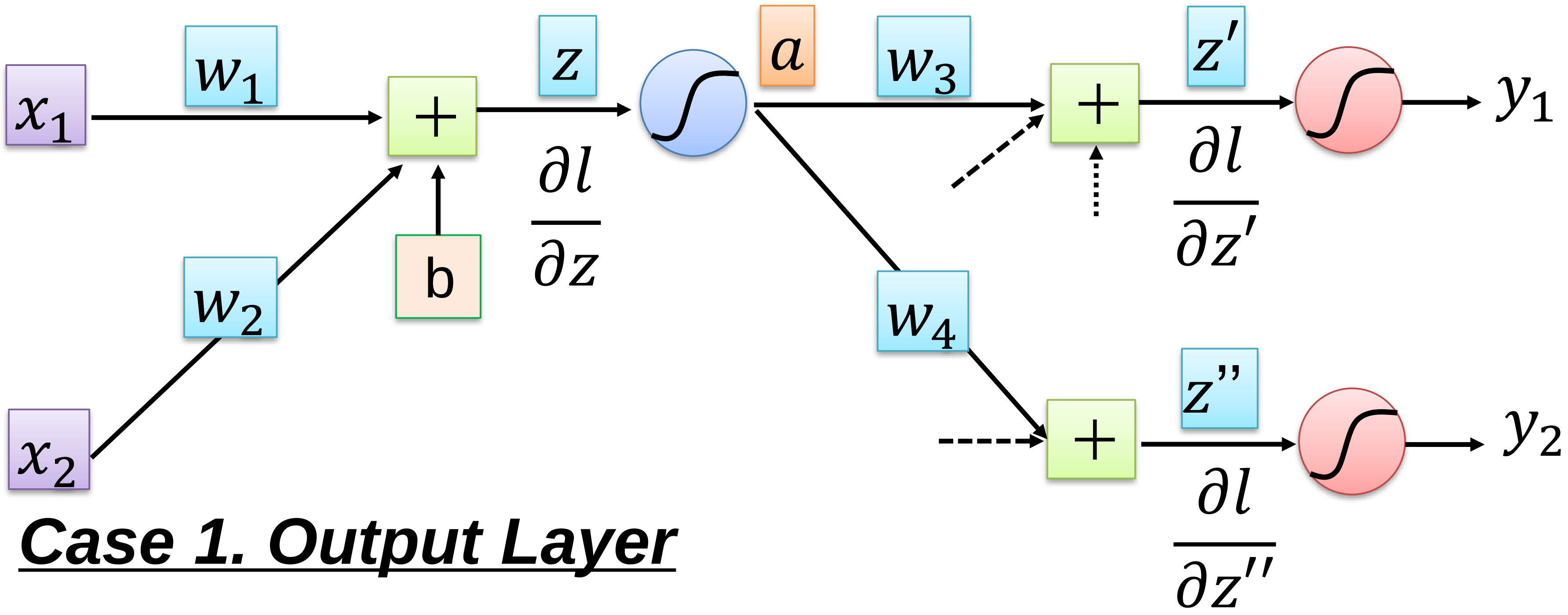

$\sigma'(z)$
w_3
$+$
$\frac{\partial l}{\partial z}$
$\frac{\partial l}{\partial z'}$
w_4
$+$
$\frac{\partial l}{\partial z''}$

$\sigma'(z)$ is a constant because z is already determined in the forward pass.

$$\frac{\partial l}{\partial z} = \sigma'(z)\left[w_3 \frac{\partial l}{\partial z'} + w_4 \frac{\partial l}{\partial z''}\right]$$

Neural Network

Compute $\partial l/\partial z$ for all activation function inputs z



x_1
w_1
+
z
a
w_3
+
z'
y_1
w_2
b
$\frac{\partial l}{\partial z}$
$\frac{\partial l}{\partial z'}$
w_4
z''
x_2
+
y_2
$\frac{\partial l}{\partial z''}$

Case 1. Output Layer

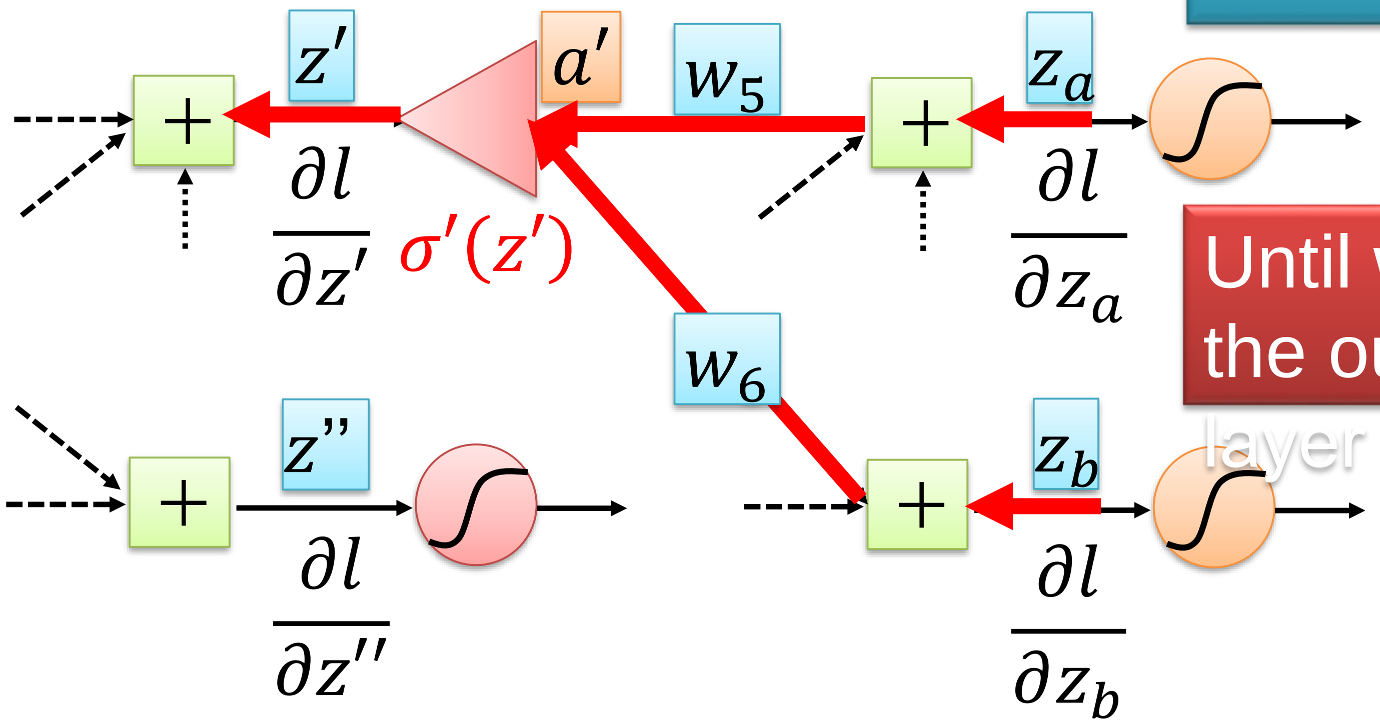
$$\frac{\partial l}{\partial z'} = \frac{\partial y_1}{\partial z'} \frac{\partial l}{\partial y_1} \qquad \frac{\partial l}{\partial z''} = \frac{\partial y_2}{\partial z''} \frac{\partial l}{\partial y_2}$$

Done!

Neural Network

Compute $\partial l / \partial z$ for all activation function inputs z

Case 2. Not Output Layer


z'
a'
w_5
z_a
$+$
$\frac{\partial l}{\partial z'}$
$\sigma'(z')$
$\frac{\partial l}{\partial z_a}$
w_6
z''
z_b
$\frac{\partial l}{\partial z''}$
$\frac{\partial l}{\partial z_b}$

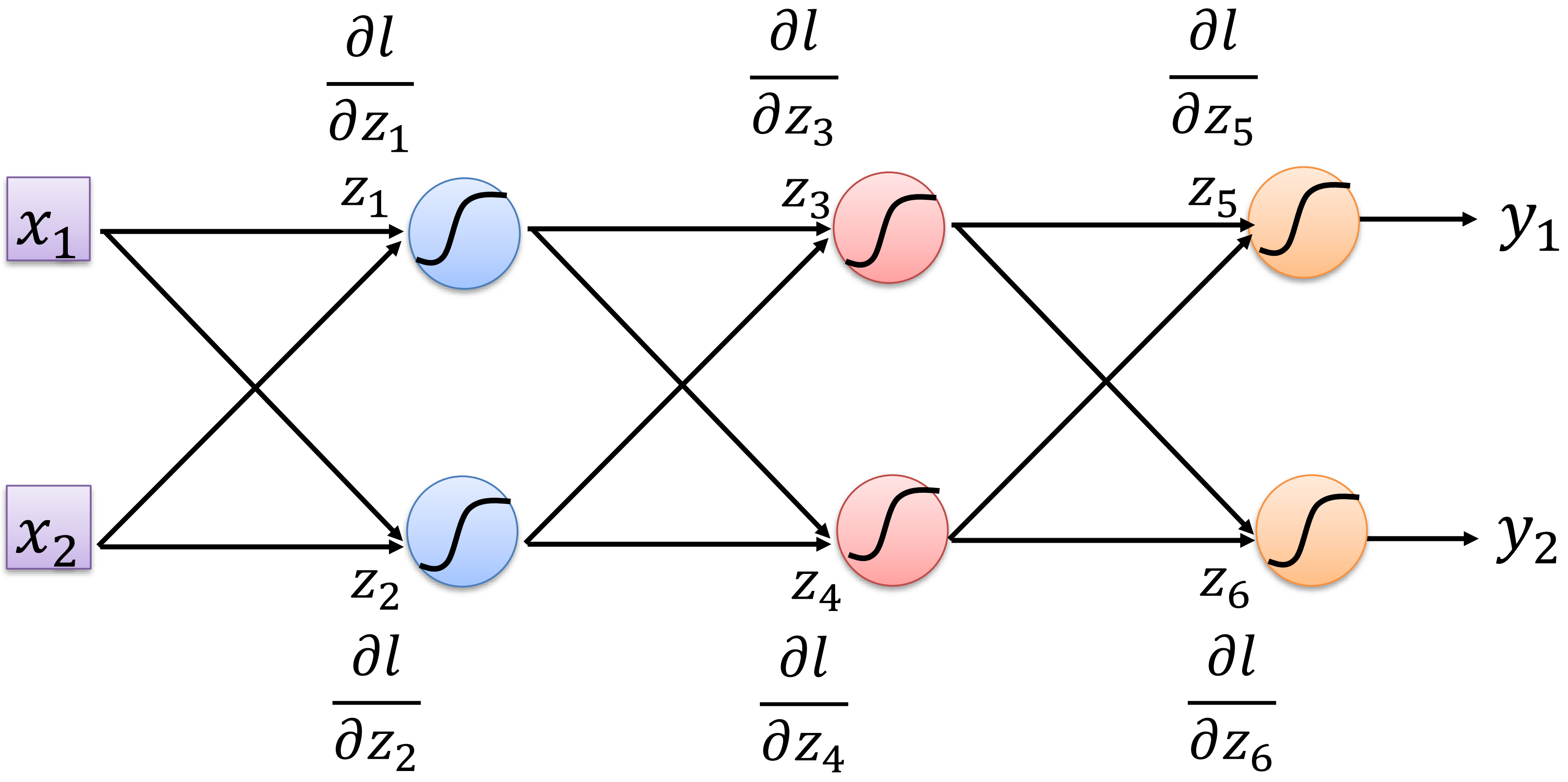
Compute $\partial l / \partial z$ recursively

Until we reach the output layer ……

Neural Network

Compute $\partial l / \partial z$ for all activation function inputs z

Compute $\partial l / \partial z$ from the output layer

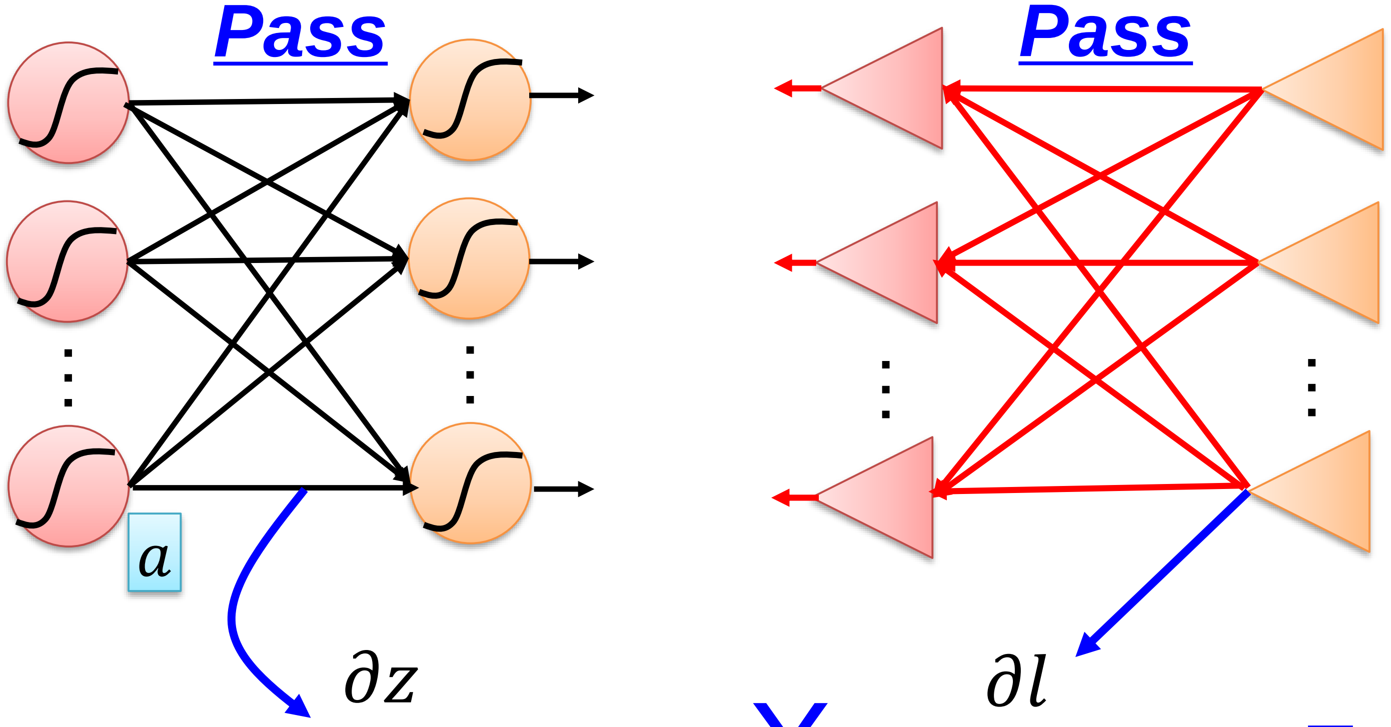


$\frac{\partial l}{\partial z_1}$
$\frac{\partial l}{\partial z_3}$
$\frac{\partial l}{\partial z_5}$
x_1
z_1
z_3
z_5
y_1
x_2
z_2
z_4
z_6
y_2
$\frac{\partial l}{\partial z_2}$
$\frac{\partial l}{\partial z_4}$
$\frac{\partial l}{\partial z_6}$

Neural Network

Forward Pass

Backward Pass

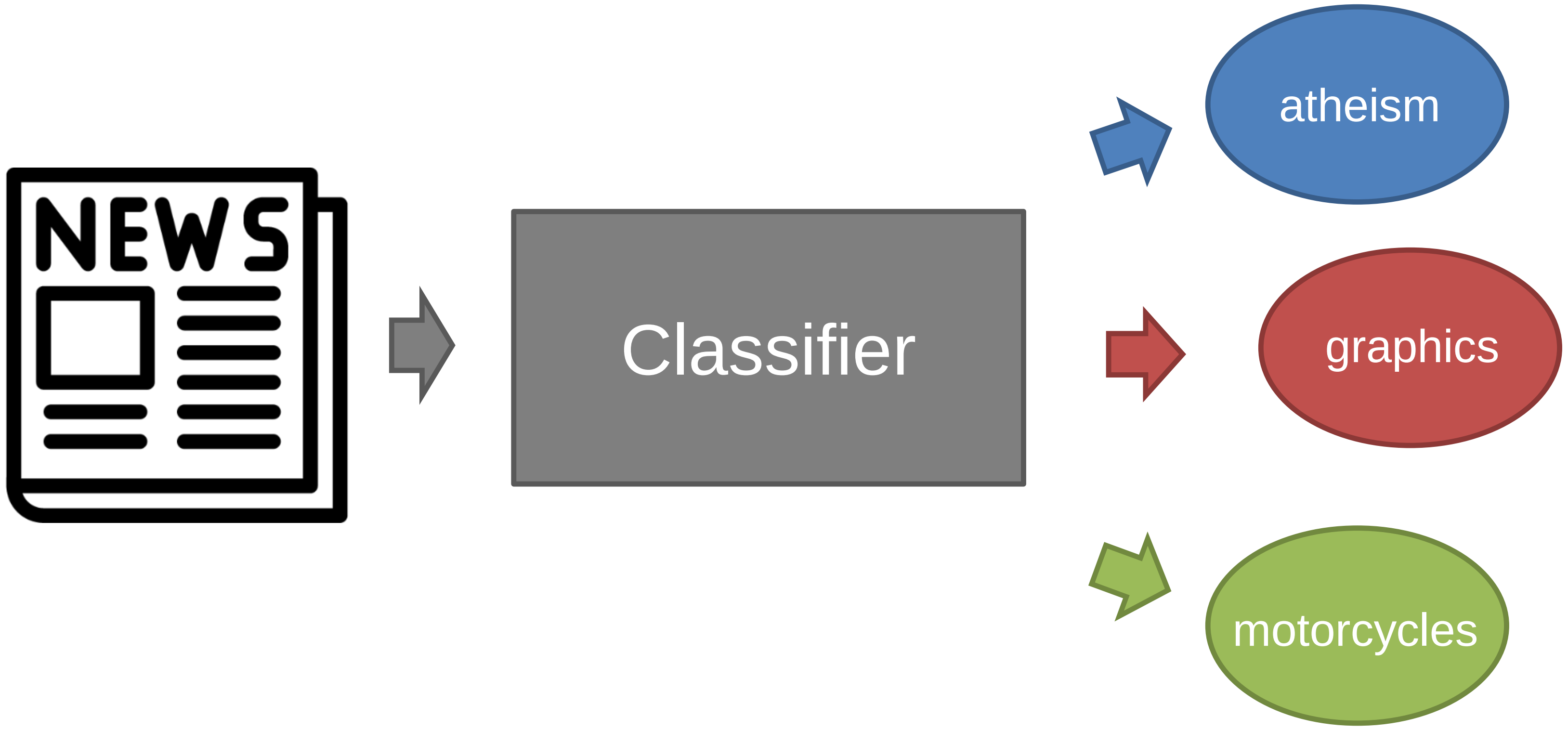


$$\frac{\partial z}{\partial w} = a \quad \text{X} \quad \frac{\partial l}{\partial z} \quad = \frac{\partial l}{\partial w}$$

for all w

Text Classification

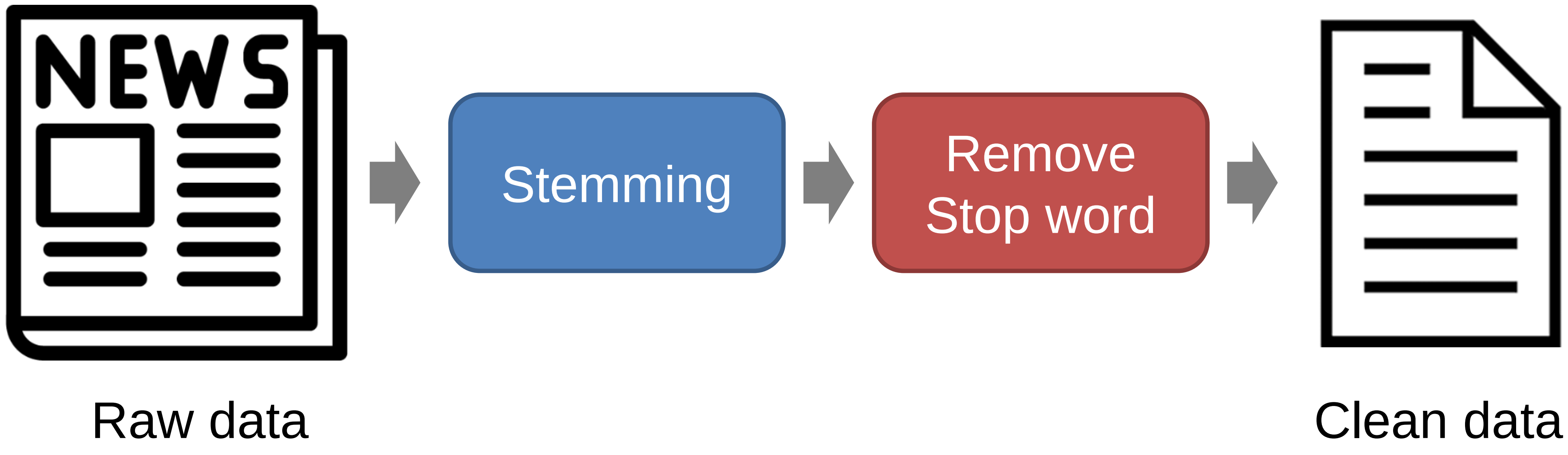
A neural network can help us to classify the category of each text. Let's define our problem



Text Classification

Data processing

The raw texts from 20 newsgroups dataset have many stop words and no-stemmed words. Therefore, we have to clean the data and preserve the significant information.



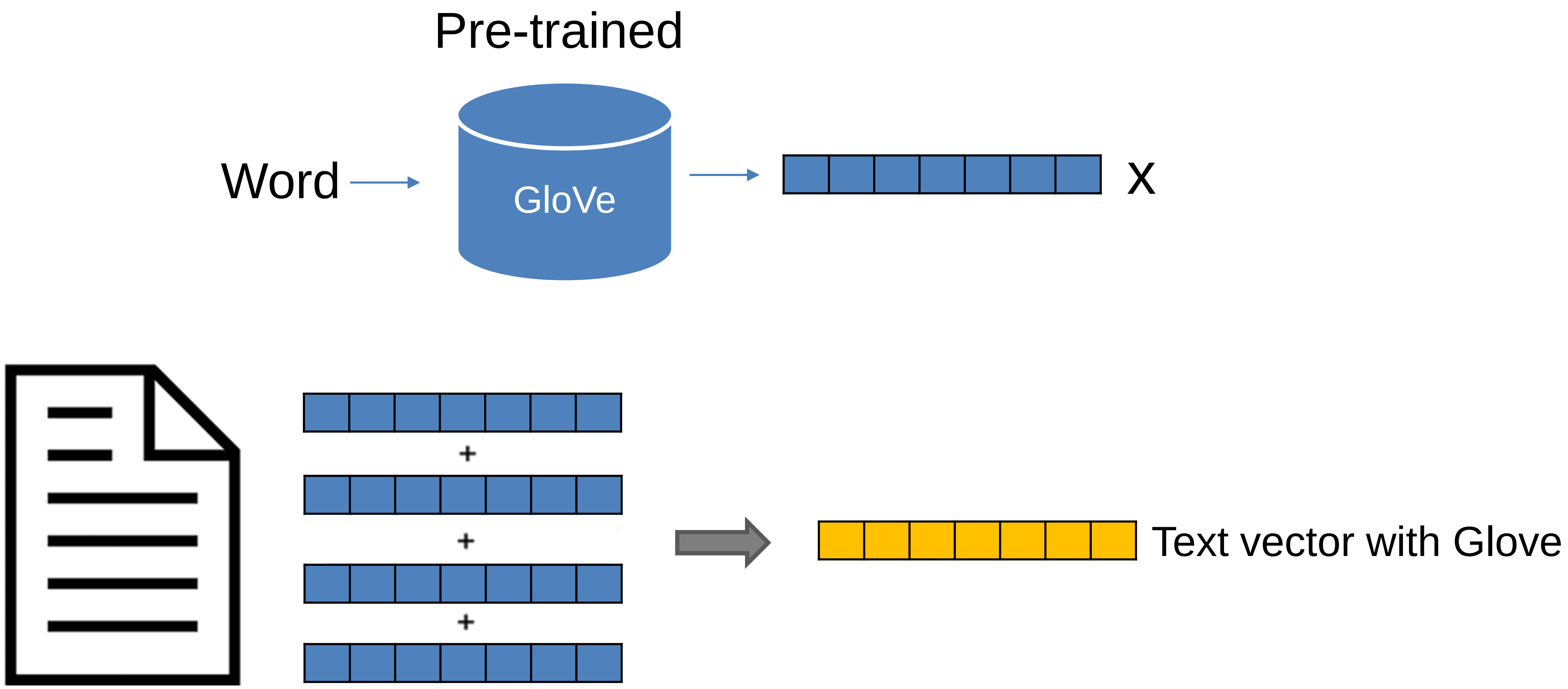
Stop word : Some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary entirely.

Stemming : For grammatical reasons, documents are going to use different forms of a word, such as organize, organizes, and organizing. If we don't stem the words in the raw text, we could not remove some stop words because of different forms not recorded in stop list.

Text Classification

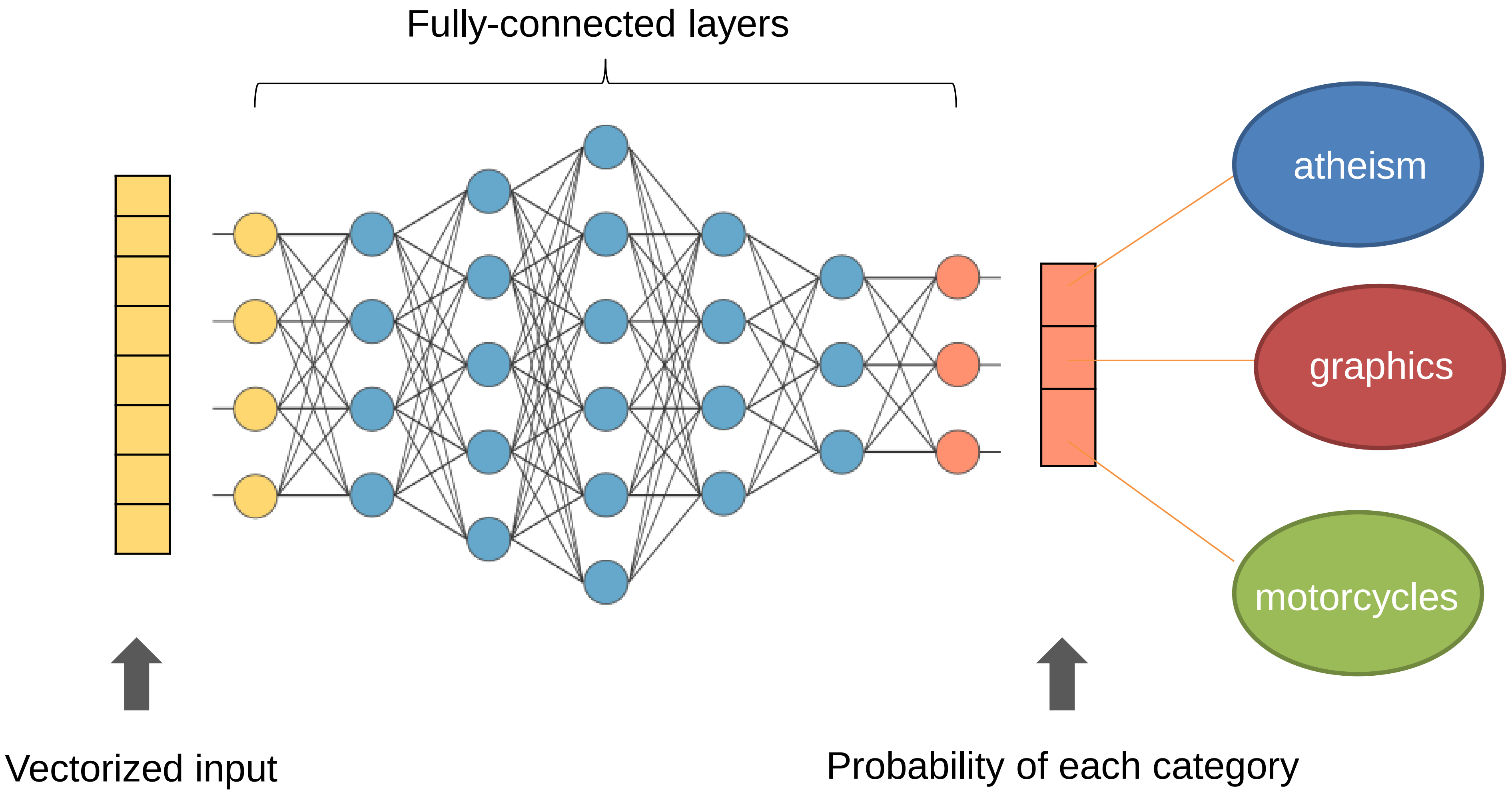
Vectorization

To use text data as our input in neural network, we must transfer texts of each news to a single vector. **GloVe** is an unsupervised learning algorithm for obtaining vector representations for words. We **sum all vectors** of all the words in the document.



Text Classification

Then, we use a neural network model to help us classify this news belong which category. Through multiple fully-connected layers , the model can give us **the probability of each category** for the news.



Text Classification

Let’s start to practice building a text classification model

https://github.com/IKMLab/Feedforward-Tutorial/blob/master/text-classfication/text-classfication.ipynb