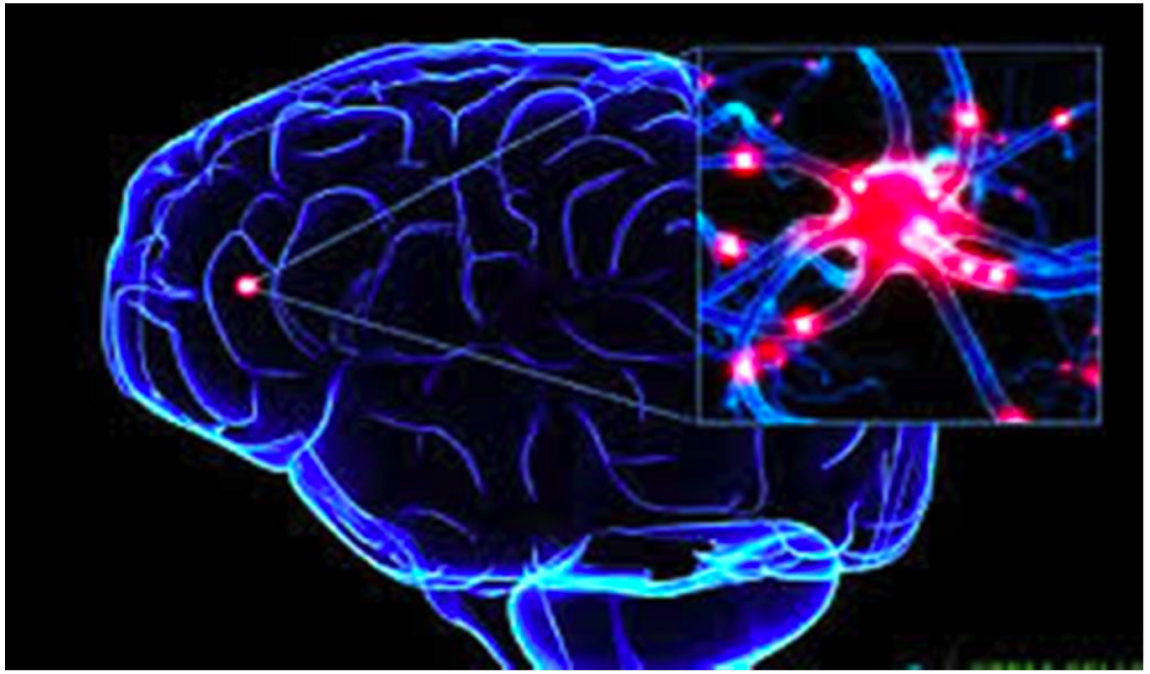
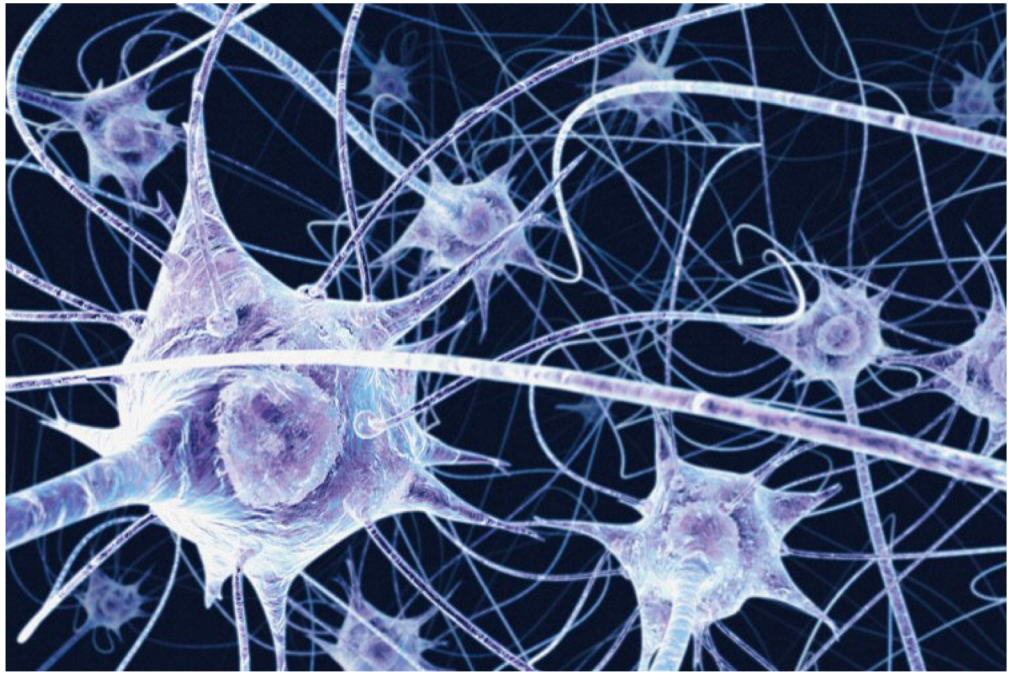
AI

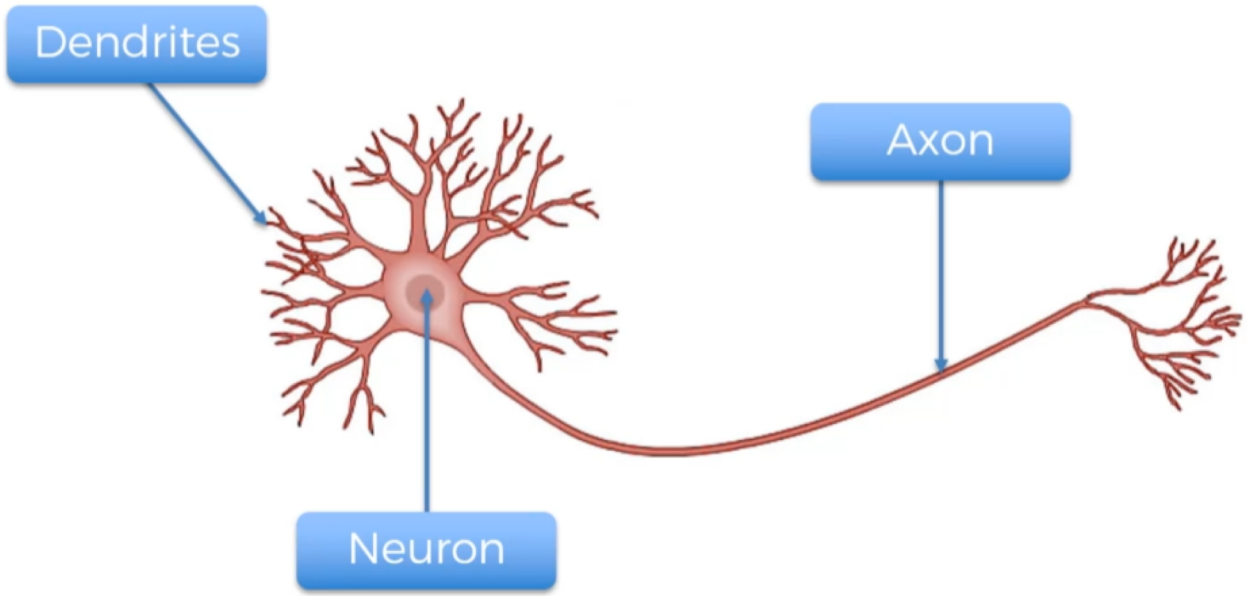
Artificial Intelligent

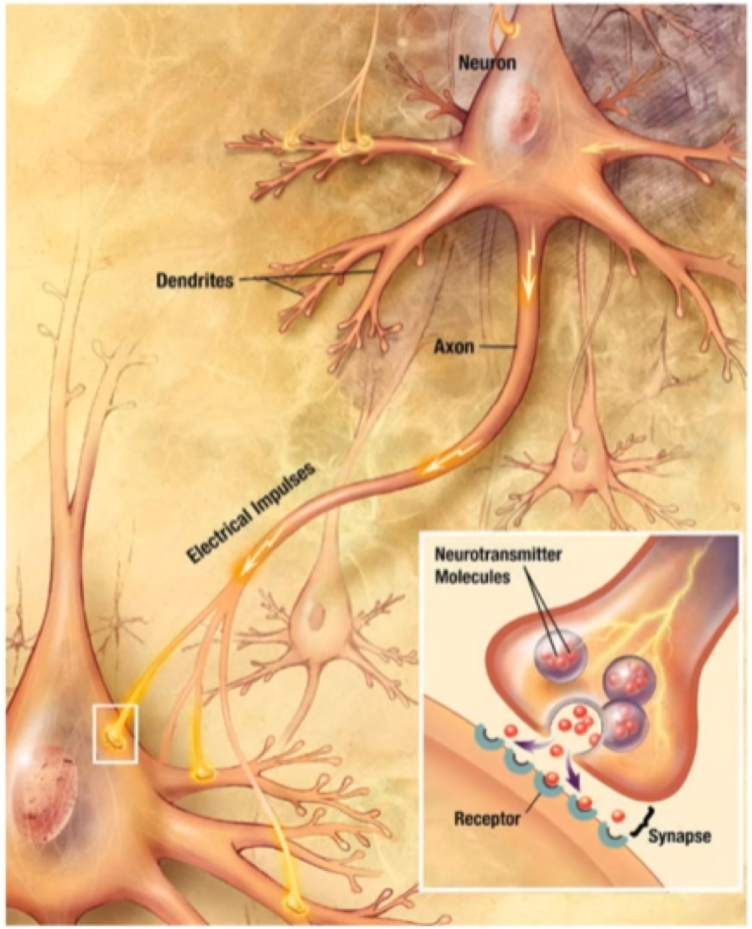
Neural Network



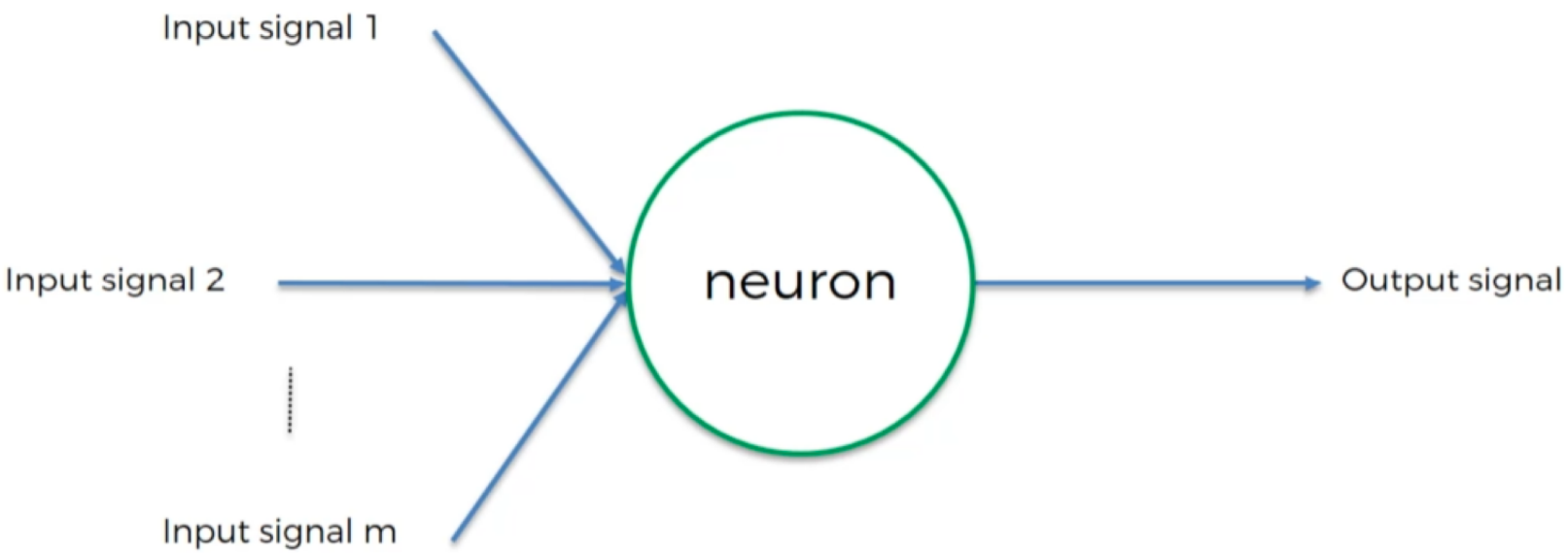


Neuron


Dendrites
Axon
Neuron


Neuron
Dendrites
Axon
Electrical Impulses
Neurotransmitter Molecules
Receptor
Synapse

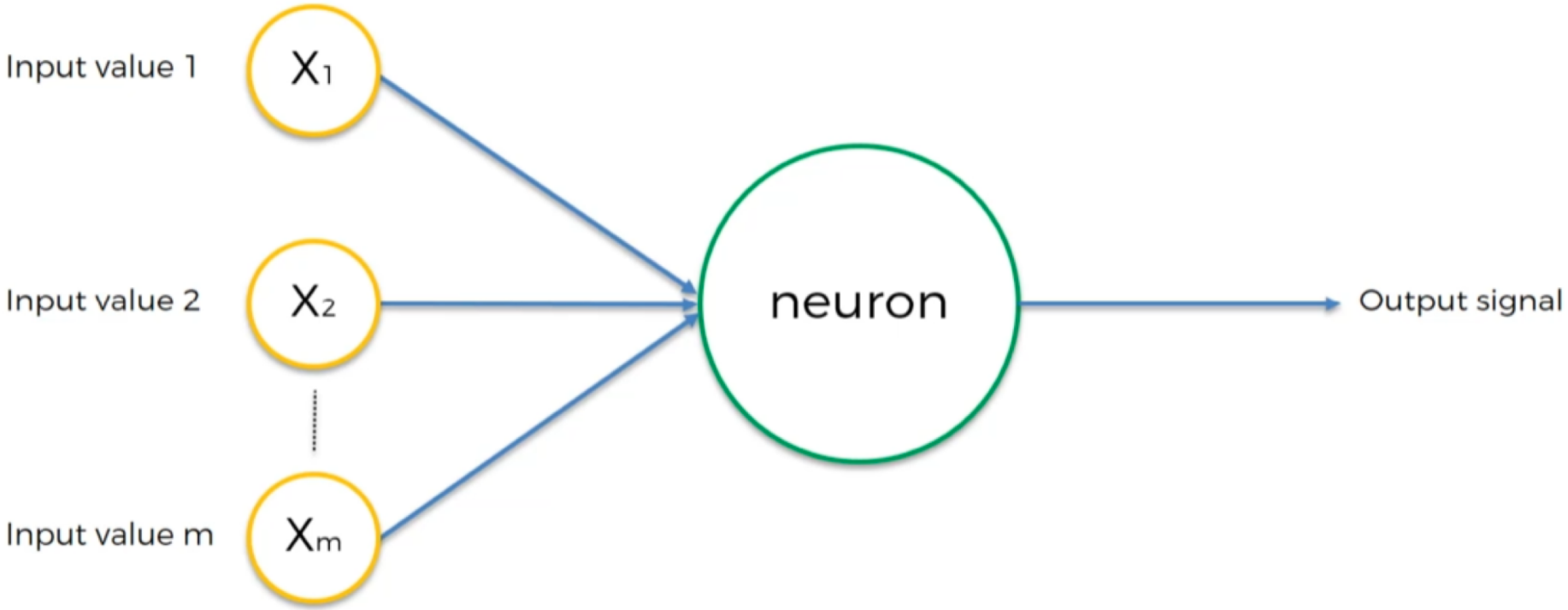
Neuron


Input signal 1
Input signal 2
Input signal m
neuron
Output signal

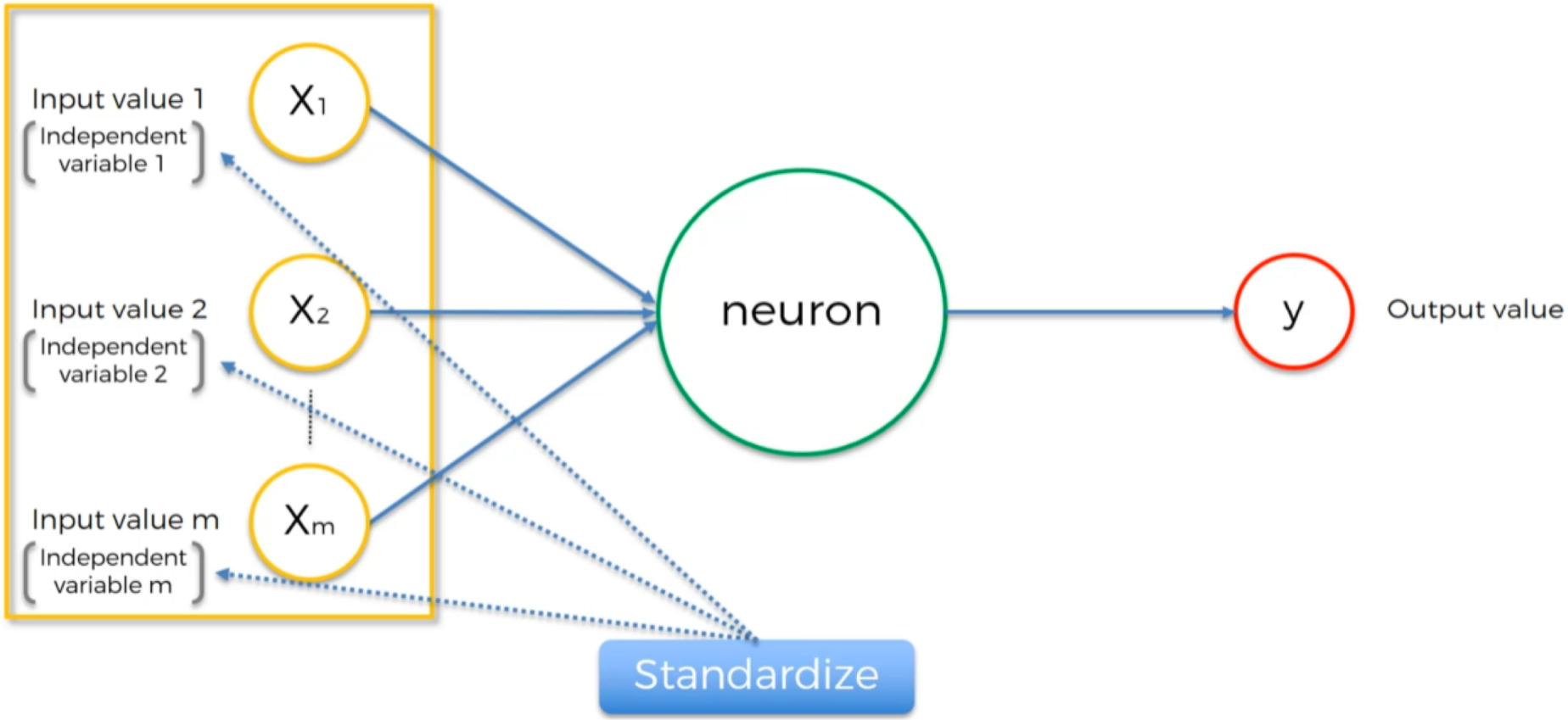
Neuron – Data Set

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

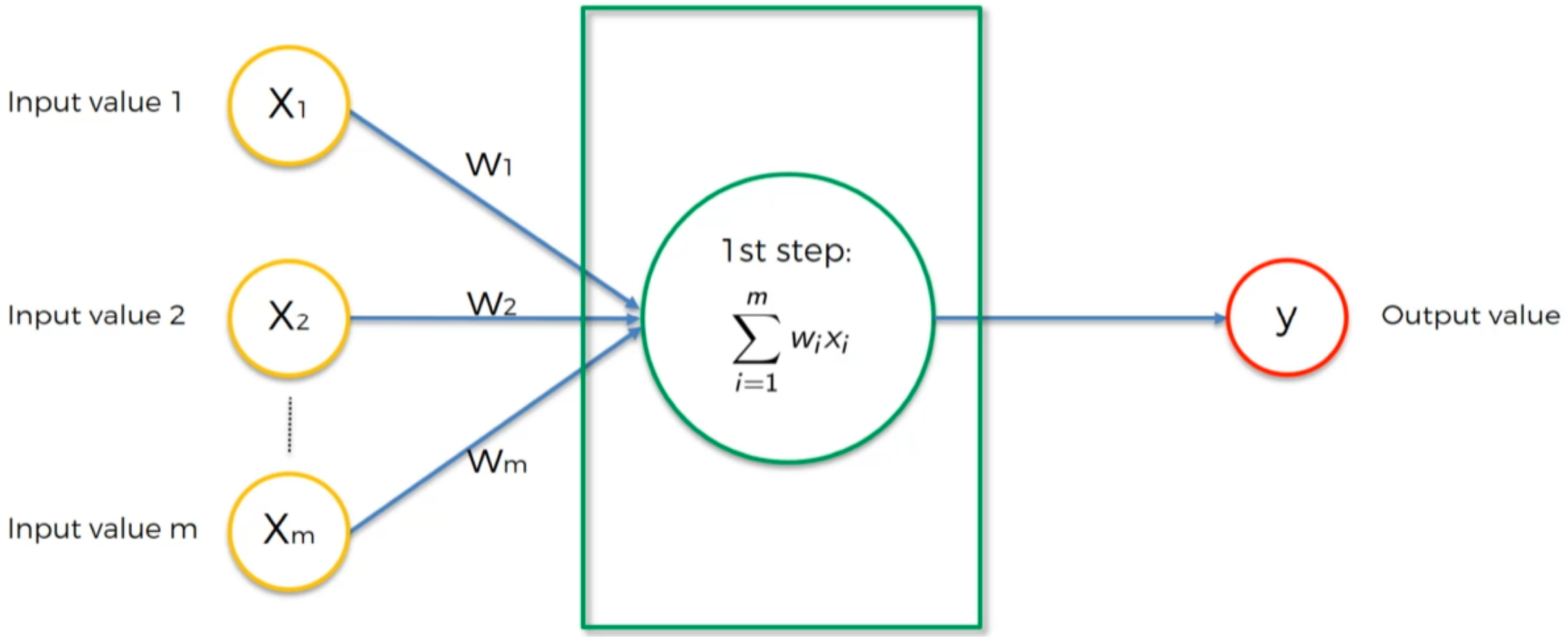
Neuron


Input value 1
X_1
Input value 2
X_2
Input value m
X_m
neuron
Output signal

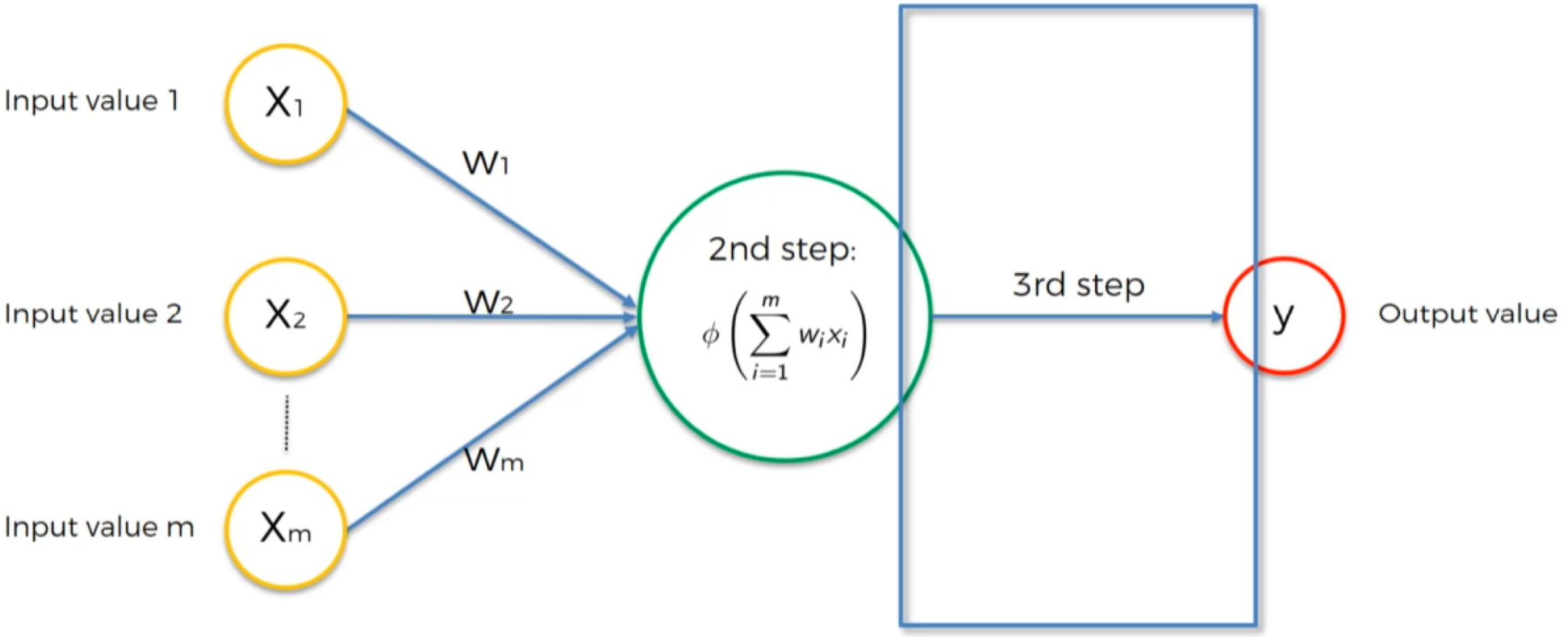
Neuron


Input value 1
Independent variable 1
X1
Input value 2
Independent variable 2
X2
Input value m
Independent variable m
Xm
neuron
y
Output value
Standardize

Neuron


Input value 1
X1
W1
Input value 2
X2
W2
Input value m
Xm
Wm
1st step:
$\sum_{i=1}^{m} w_i x_i$
y
Output value

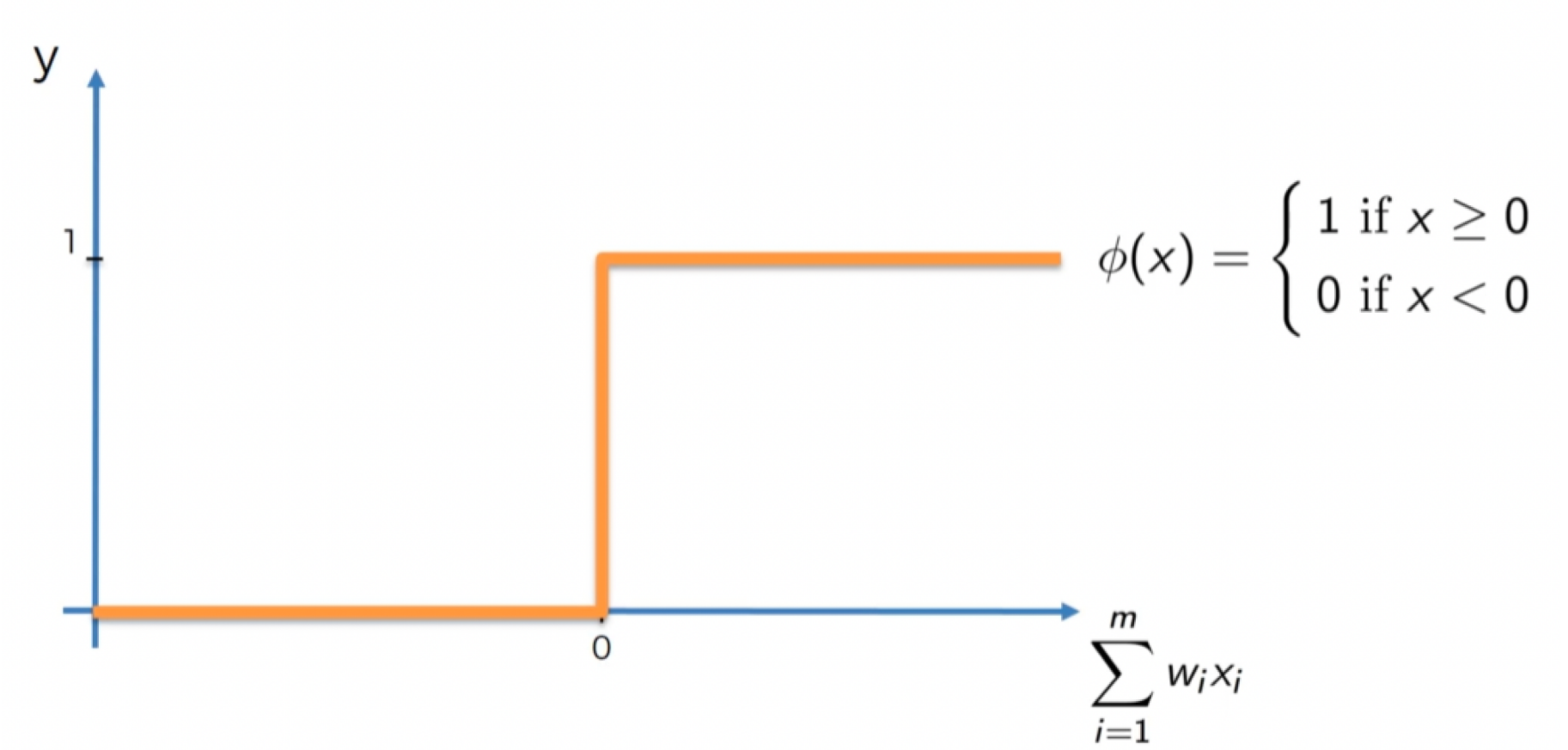
Activation Function



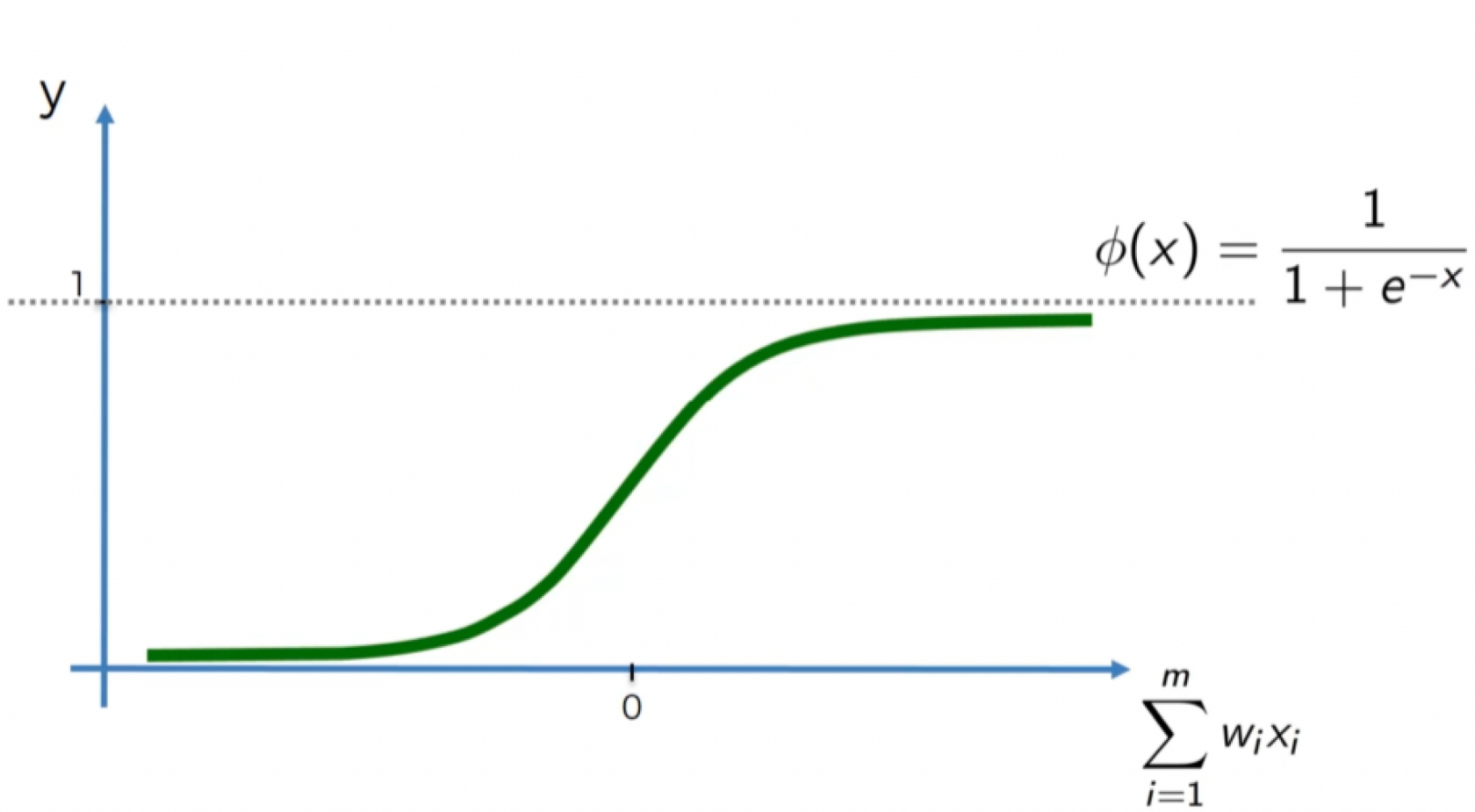
Activation Function

- Their main purpose is to convert a input signal of a node in a A-NN to an output signal
- That output signal can be used as a input in the next layer in the stack
- Specifically in A-NN
  - we do the sum of products of inputs(**X**) and their corresponding Weights(**W**)
  - apply a Activation function **f(x)** to it to get the output of that layer
  - feed it as an input to the next layer

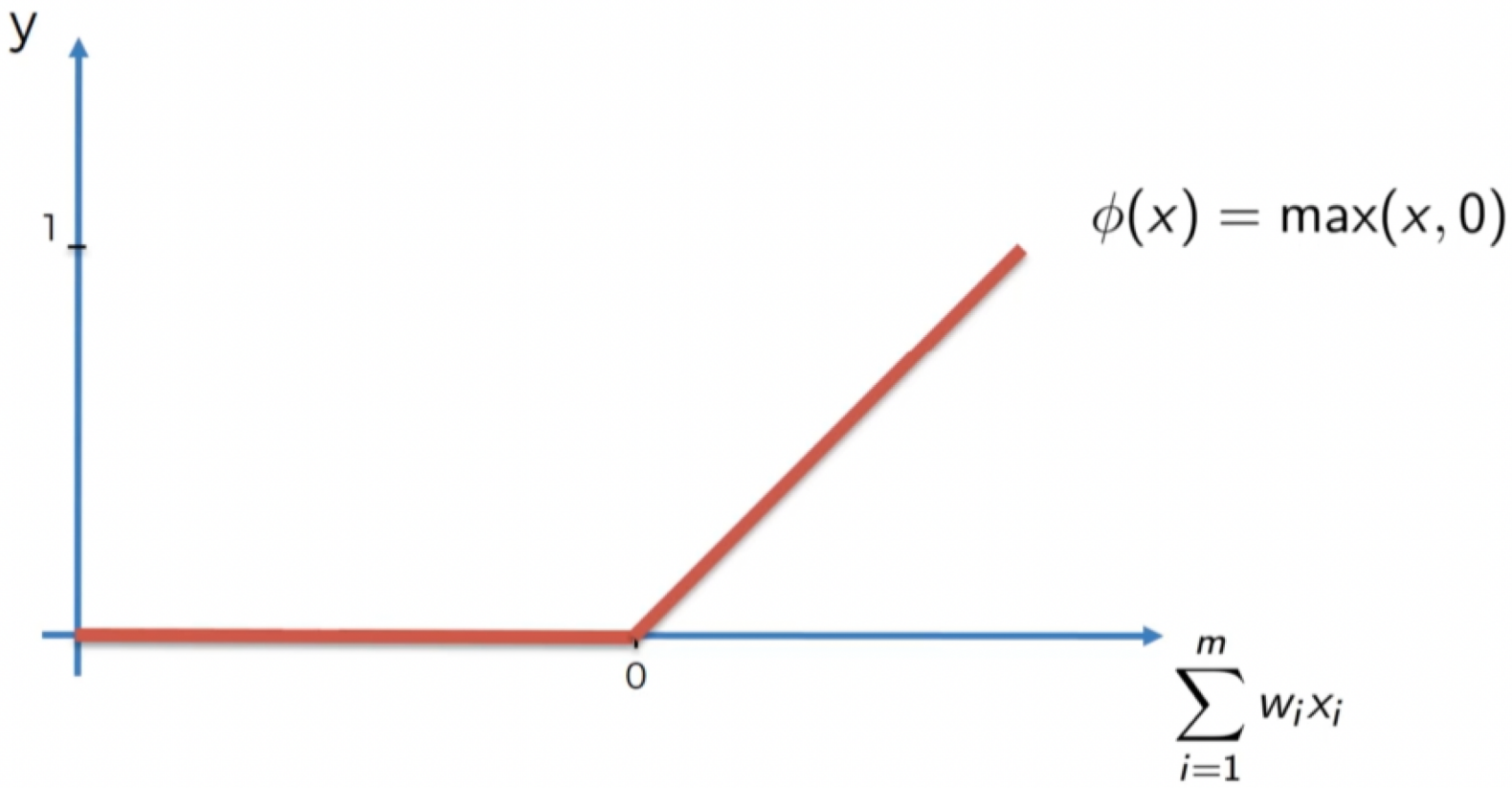
Threshold Function



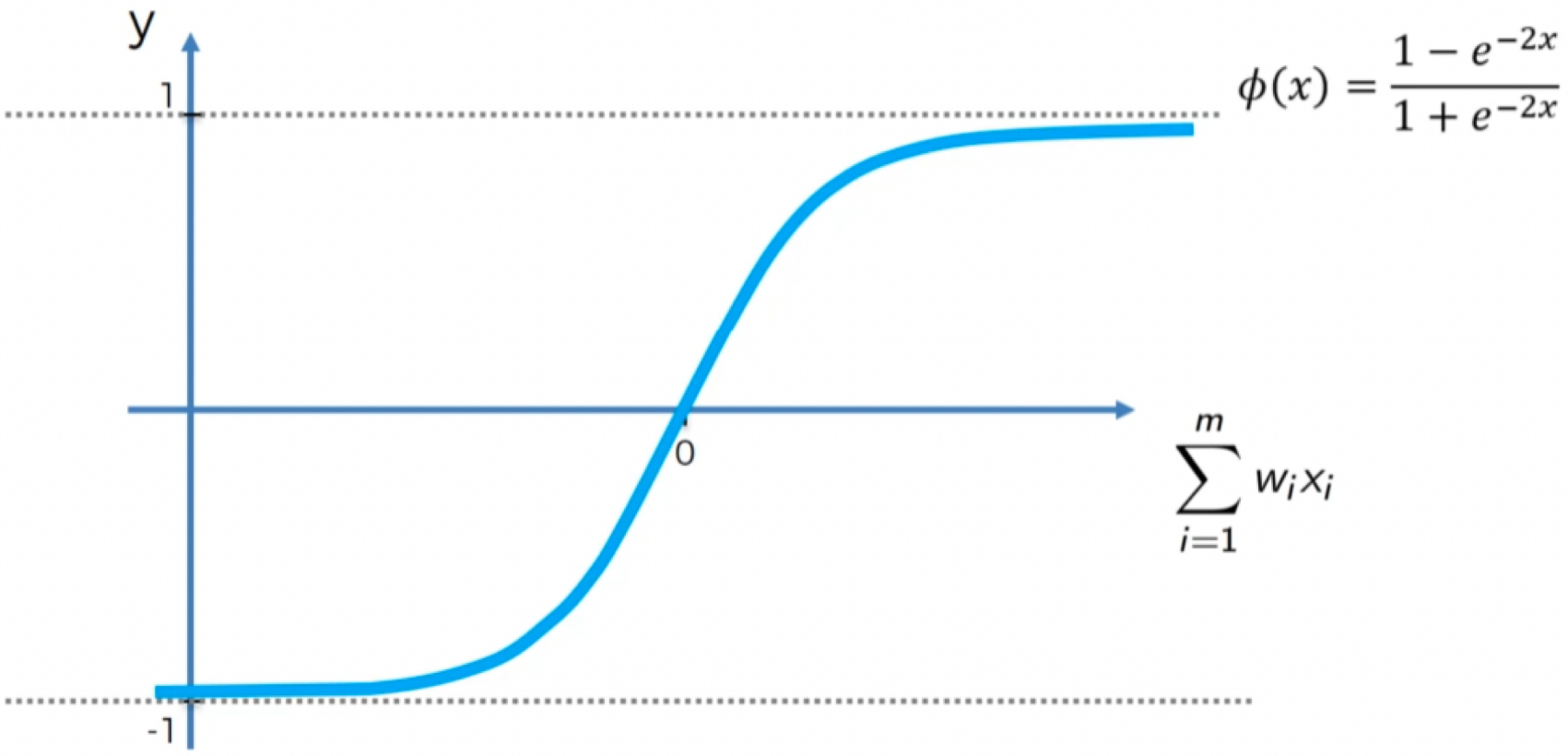
Sigmoid Function



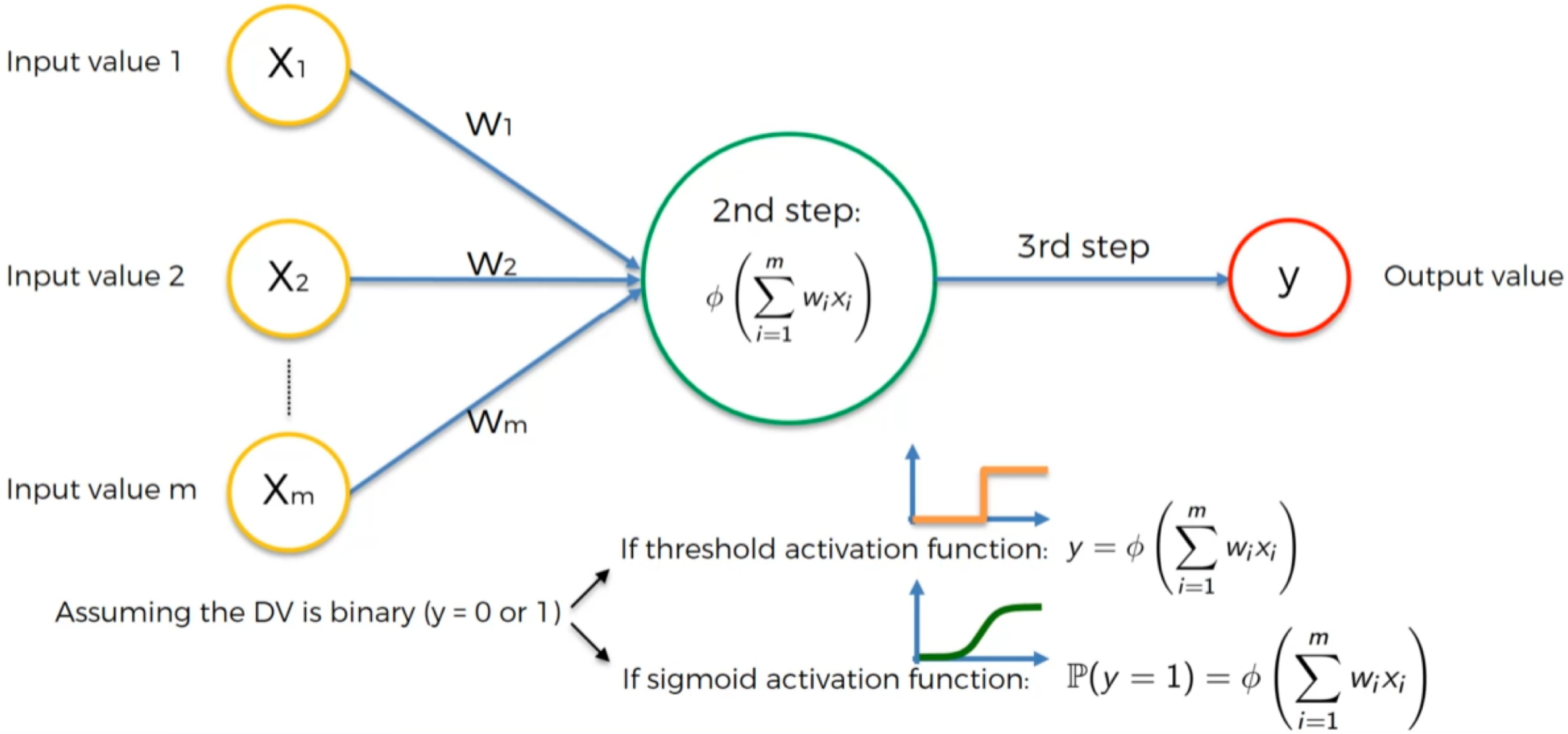
Rectifier Function



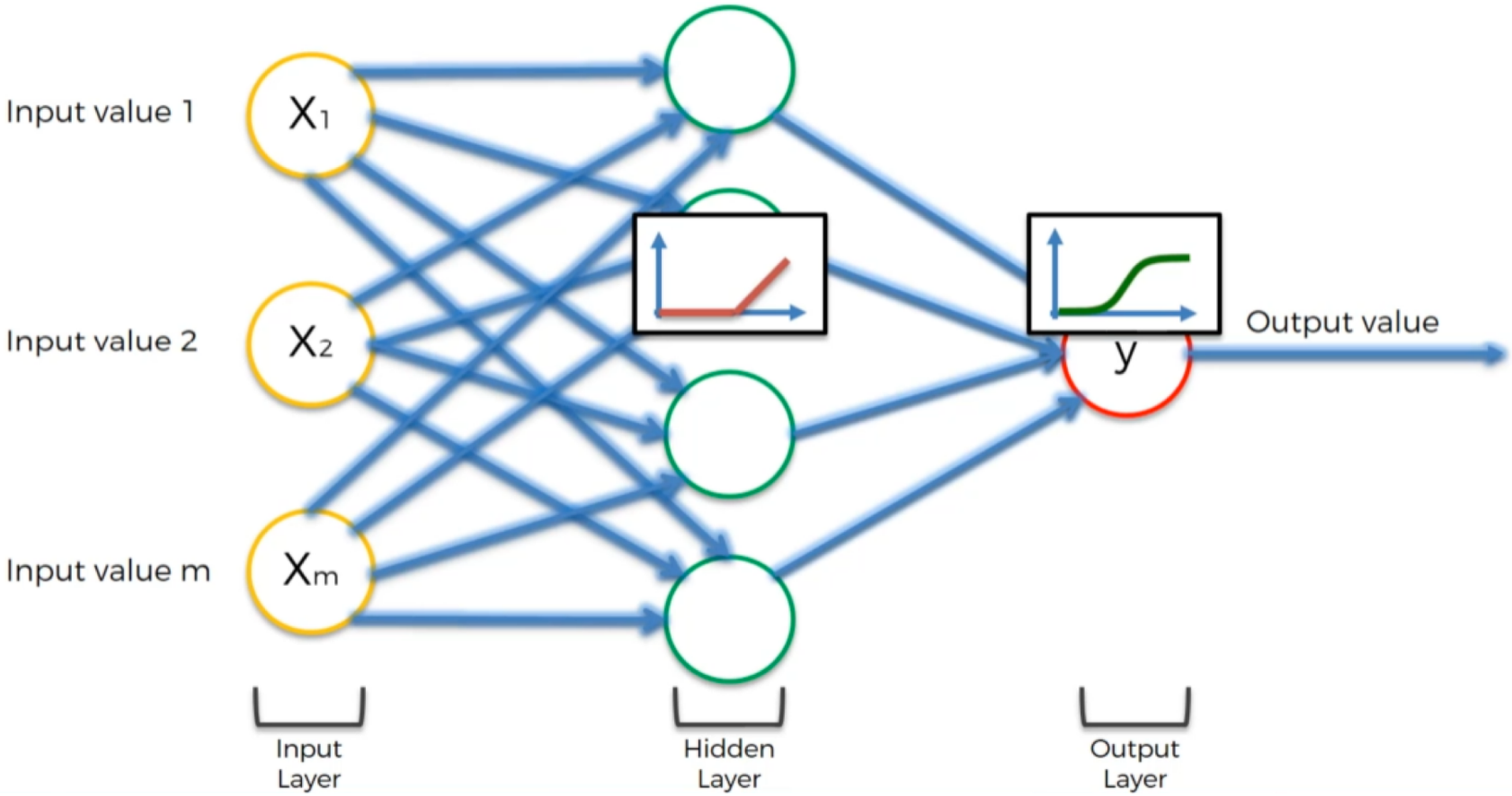
Hyperbolic Tangent Function



Activation Function

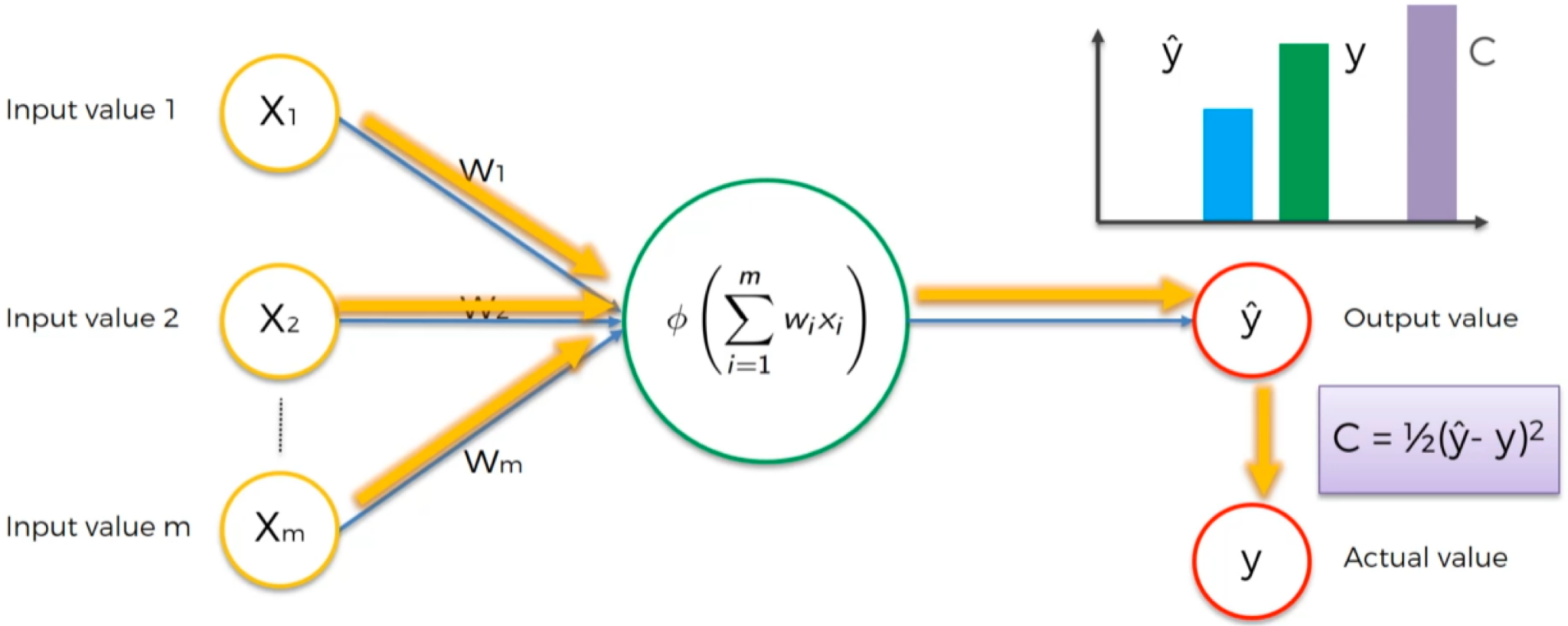


Activation Function

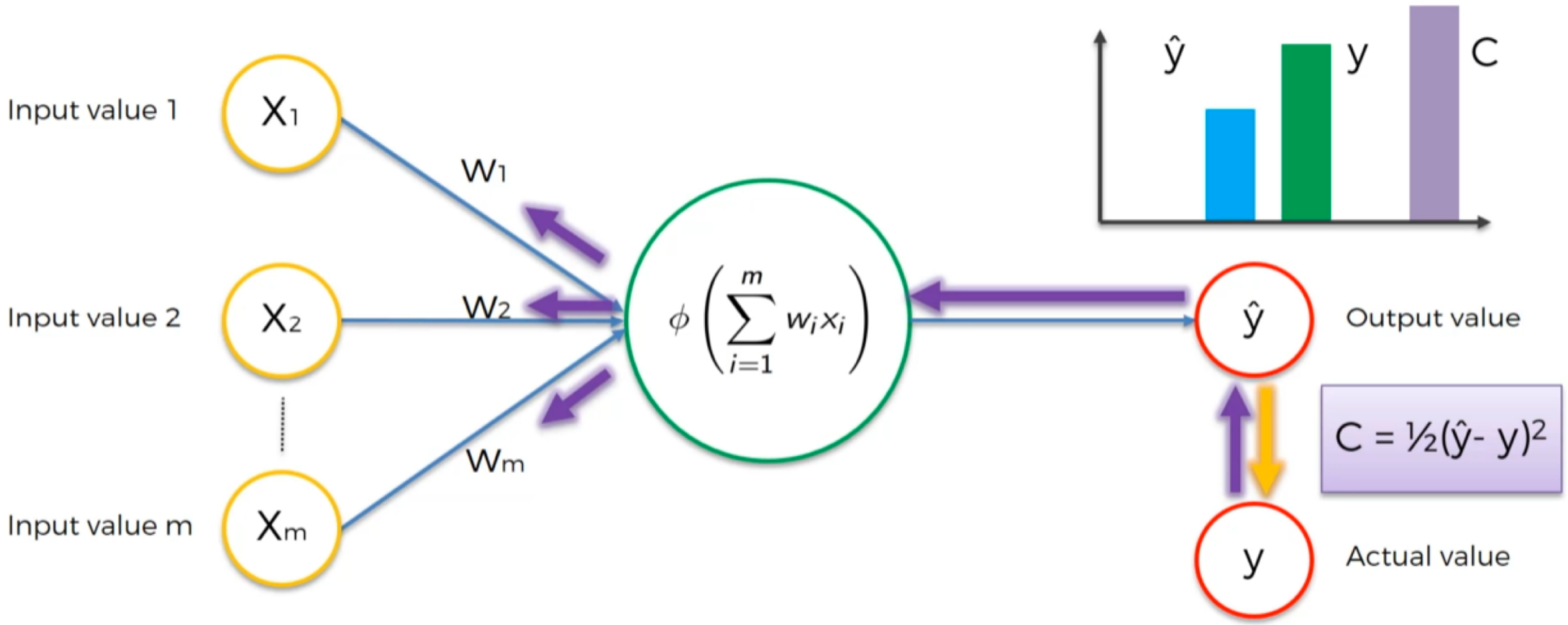


How NN learn ?

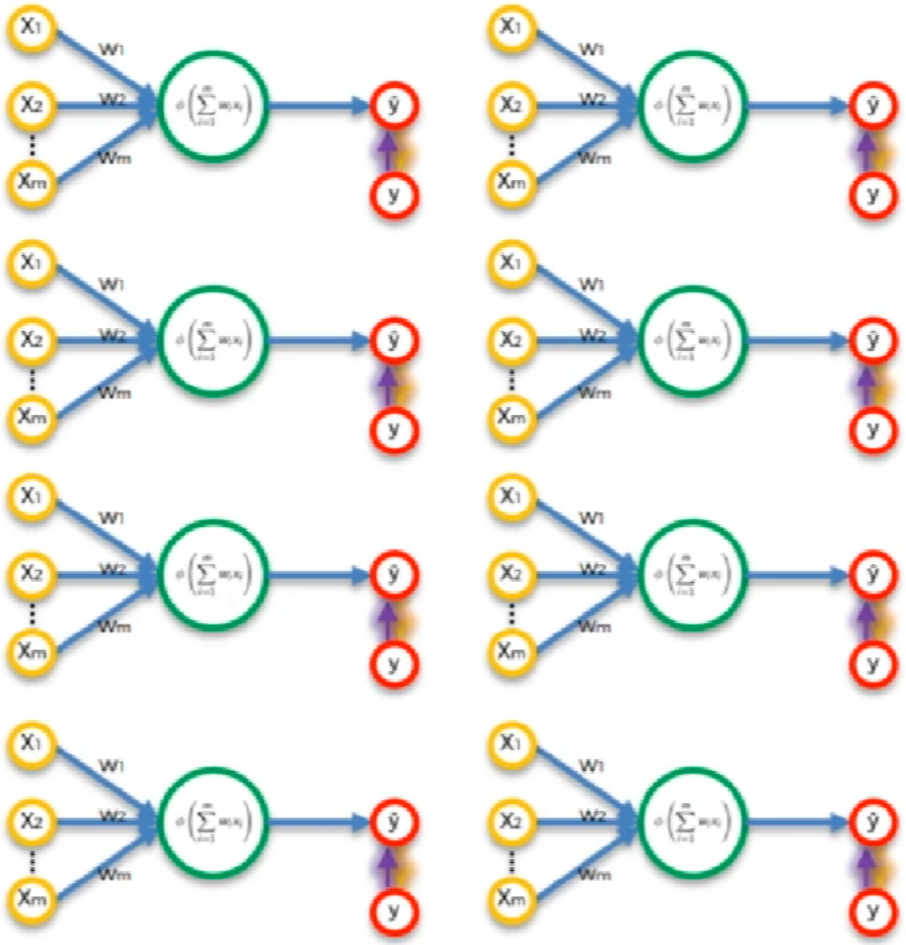
Neural Network - learn



Neural Network - learn



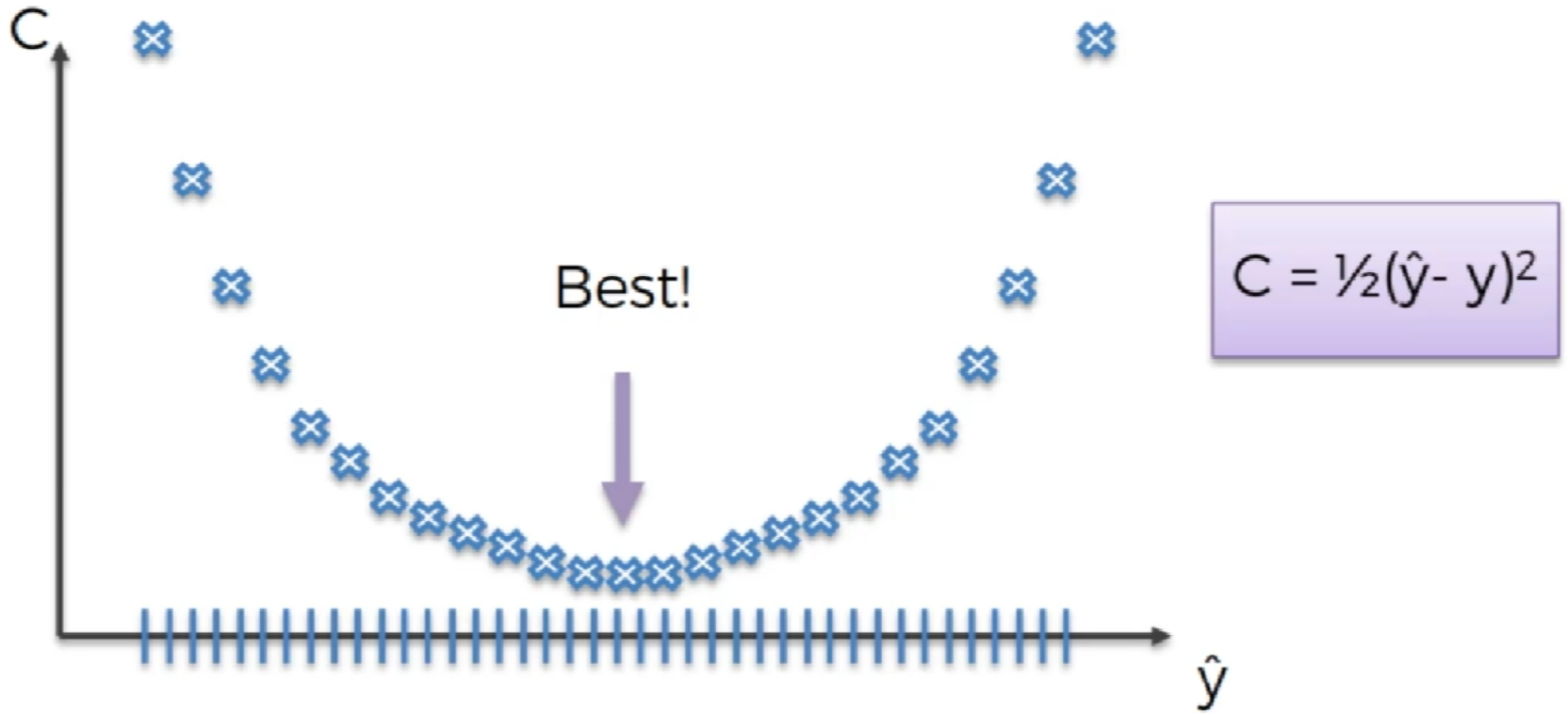
Neural Network - learn



Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

$$C = \sum \tfrac{1}{2}(\hat{y} - y)^2$$

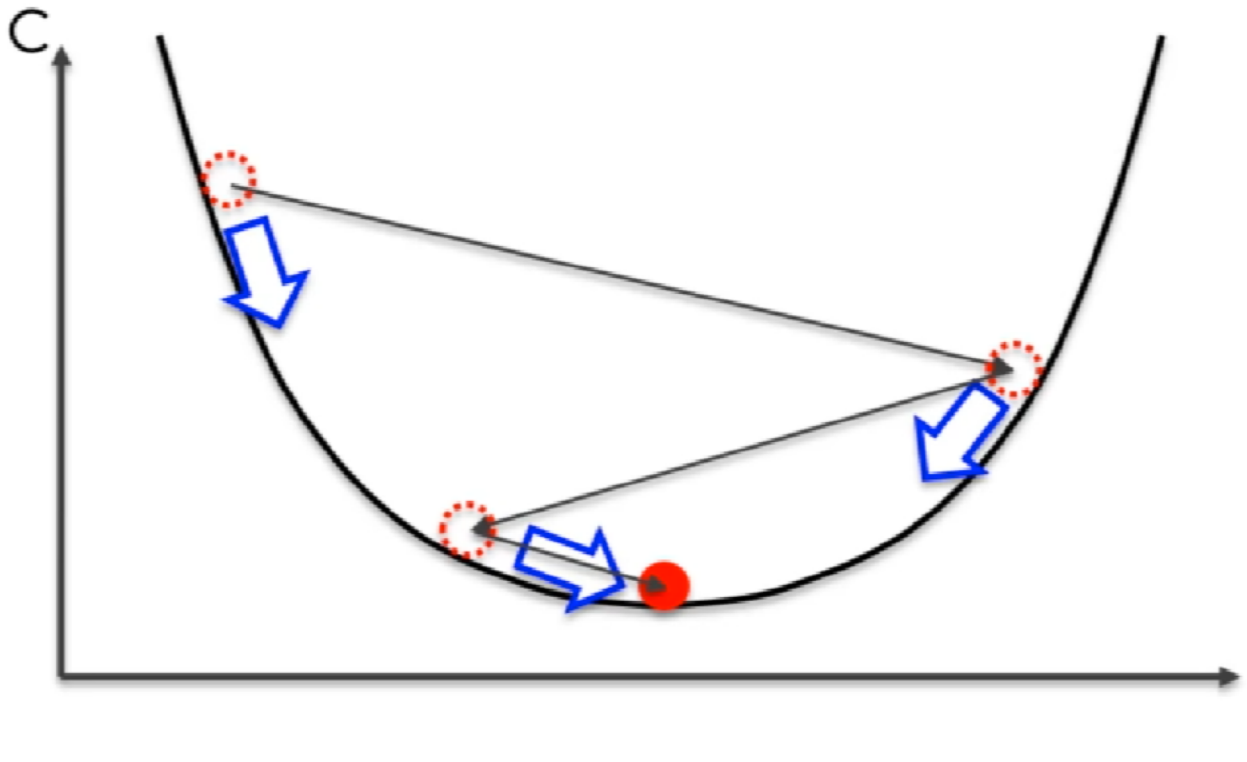
Gradient Descent



Curse of Dimensionality

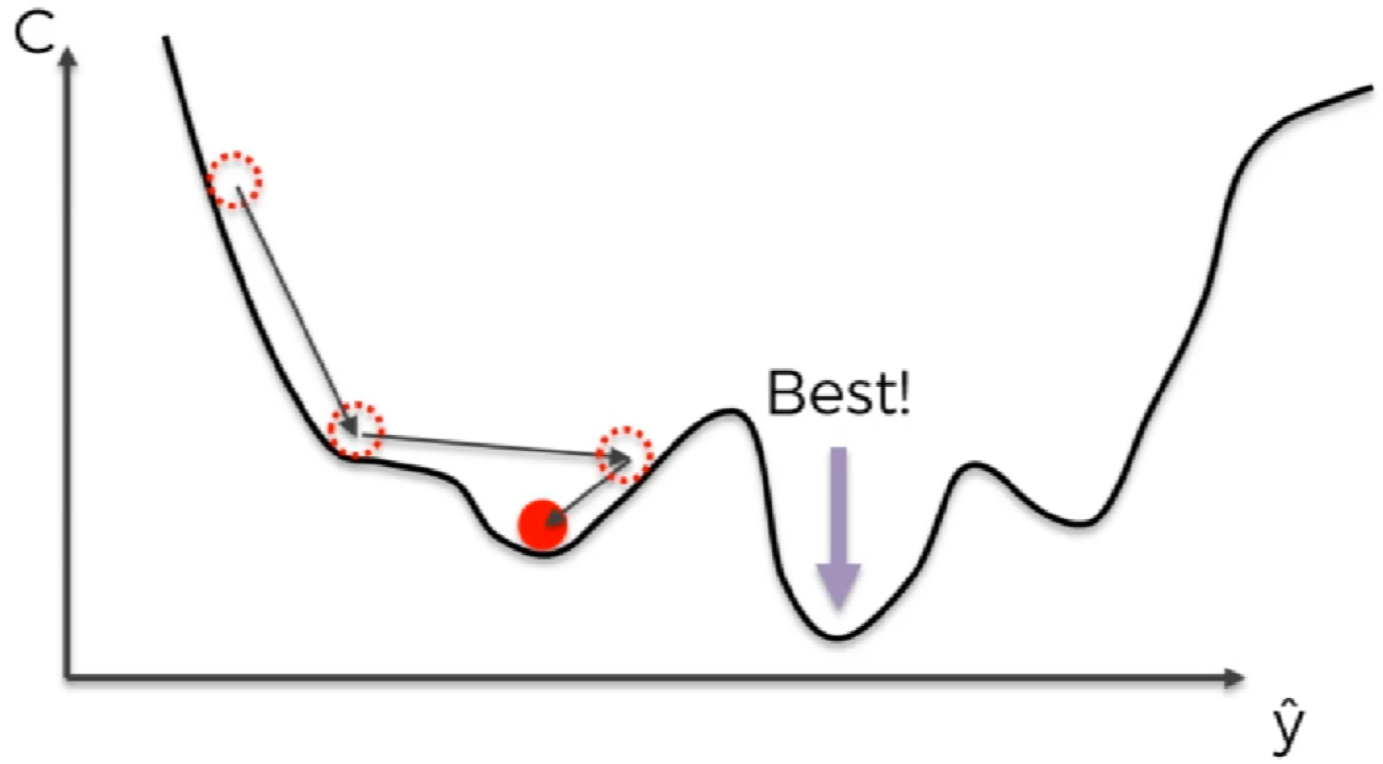
- As the dimensionality of features increases, the number of configurations grow exponentially and thus the number of configurations covered by observations decreases

Gradient Descent

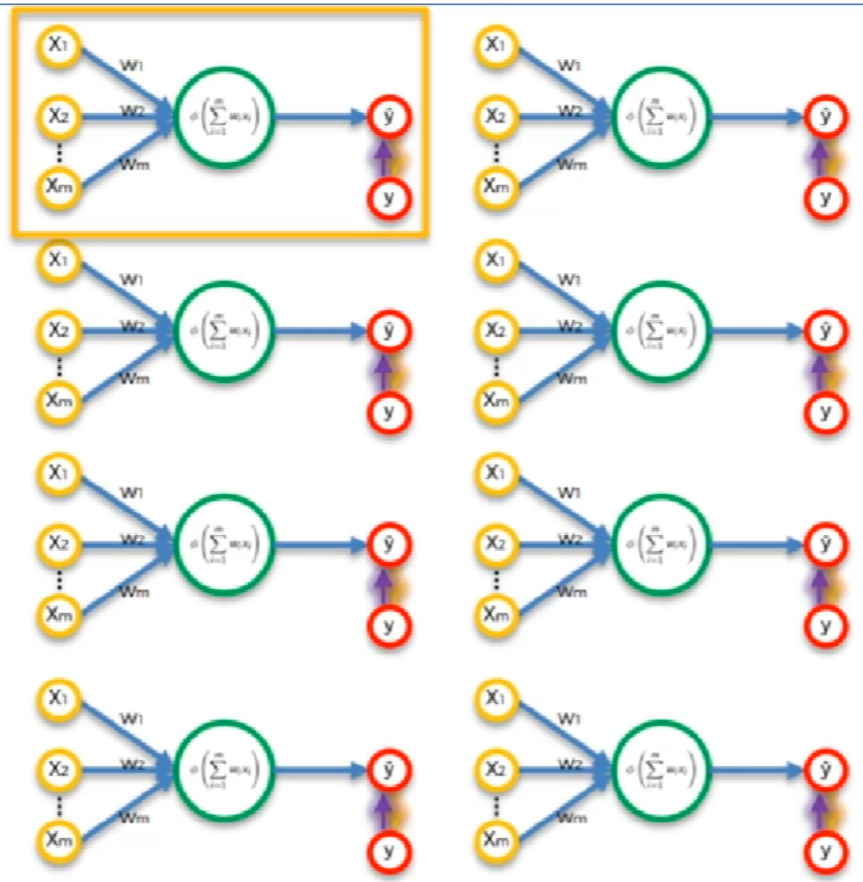


$$C = \frac{1}{2}(\hat{y} - y)^2$$

Stochastic Gradient Descent

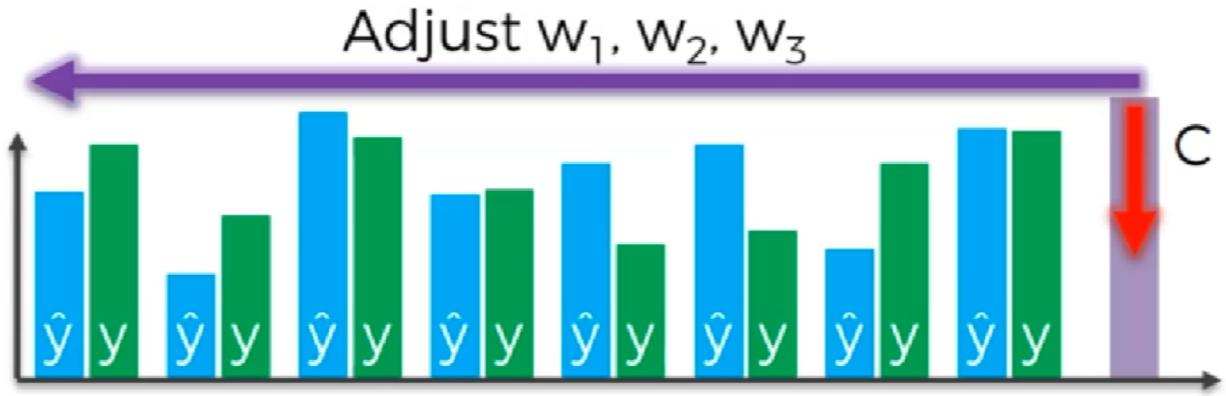


Stochastic Gradient Descent

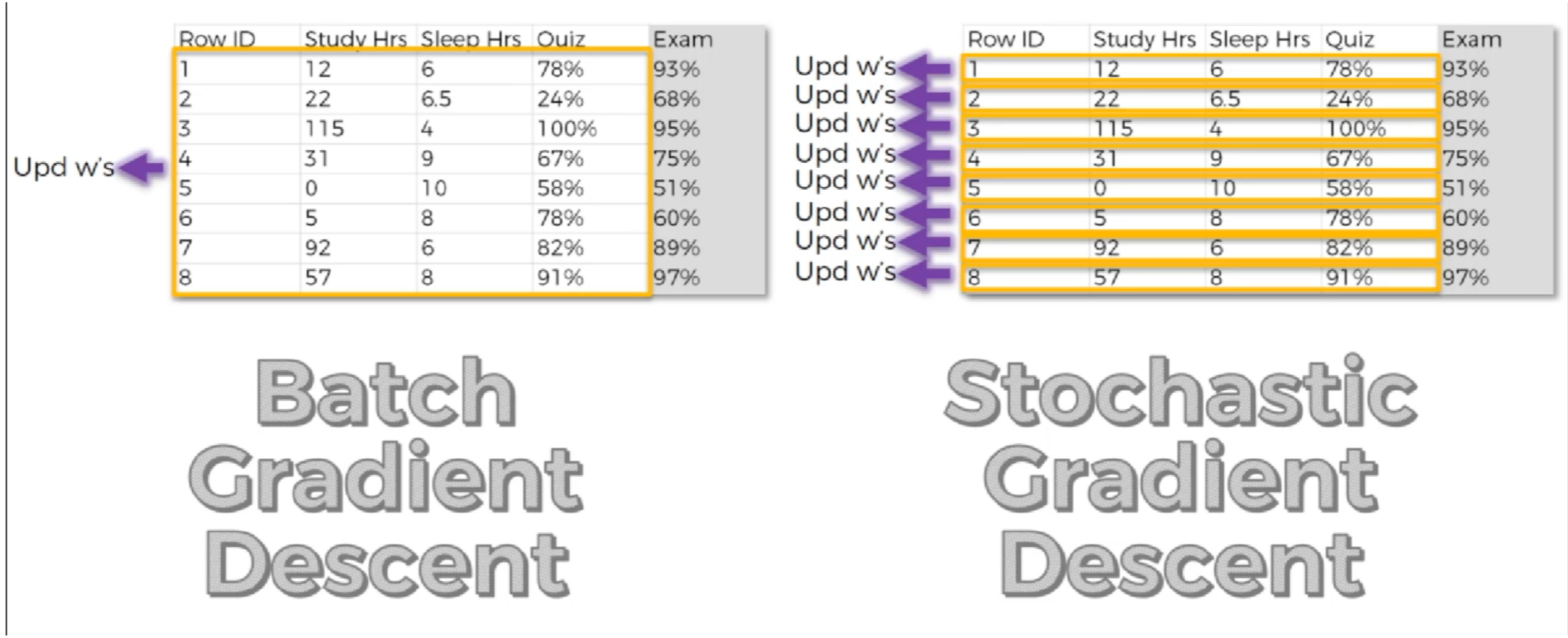


Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

$$C = \sum \tfrac{1}{2}(\hat{y} - y)^2$$



Stochastic Gradient Descent



Stochastic Gradient Descent - Steps

- Randomly initialize the weights to small numbers close to 0 (!= 0)
- Input the first observation in the input layer, each feature in one input node
- Forward Propagation:
  - From left to right
  - The neurons are activated in a way that the impact of each neuron's activation is limited by the weights
  - Propagate until getting the predicted result
- Compare the predicted result to the actual result
- Measure the generated error

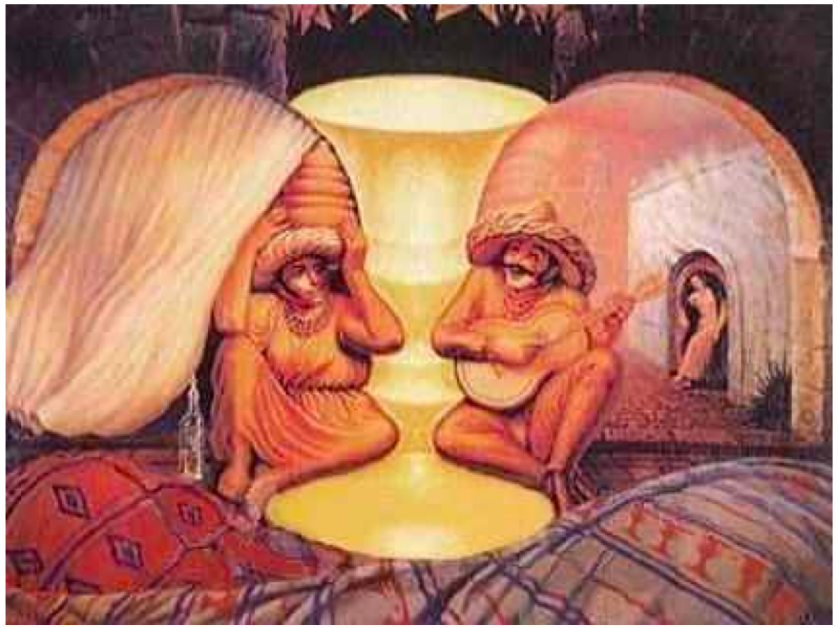
Stochastic Gradient Descent - Steps

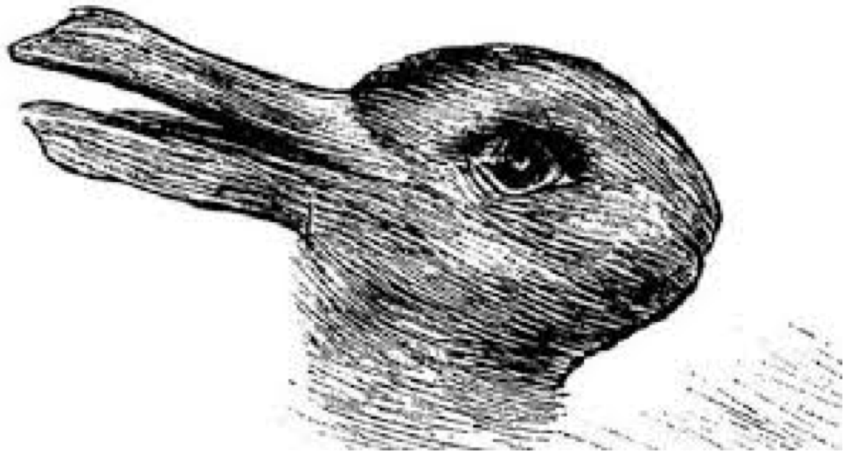
- Back Propagation
  - From right to left
  - The error is back propagated
  - Update the weights according to how much they are responsible for the error
  - The learning rate decides by how much we update the weights
- Repeat the steps and update the weights
  - after each observation
  - After a batch of observations
- When the whole training set passes through the ANN, that makes an epoch. Redo more epochs.

CNN

Convolutional Neural Network

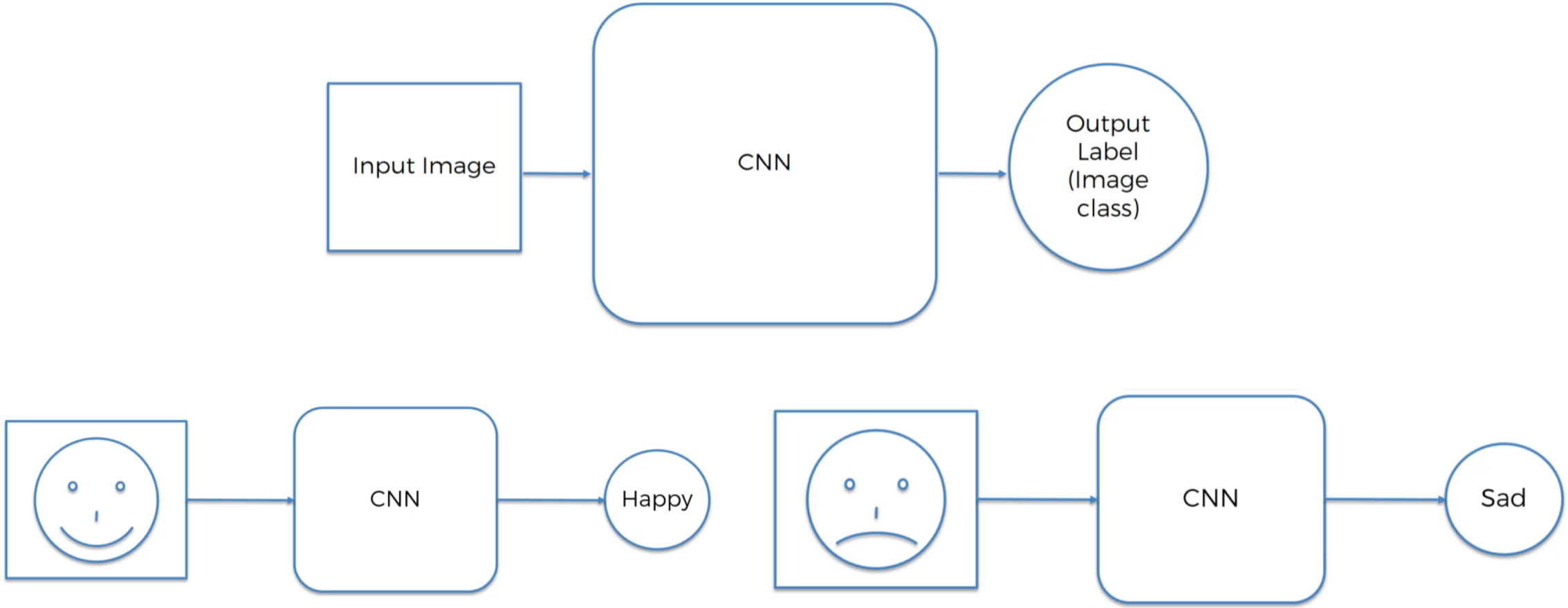
CNN



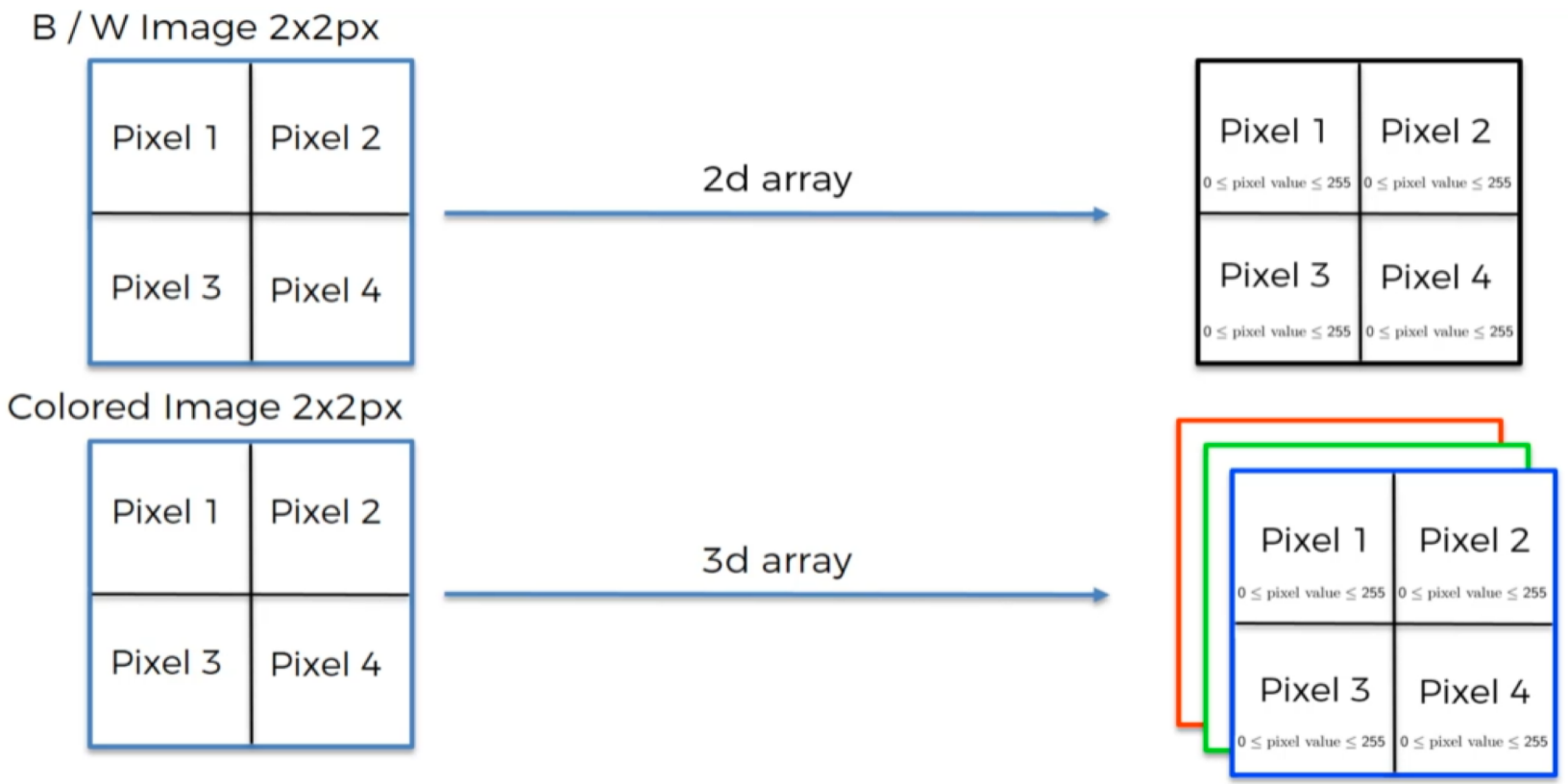


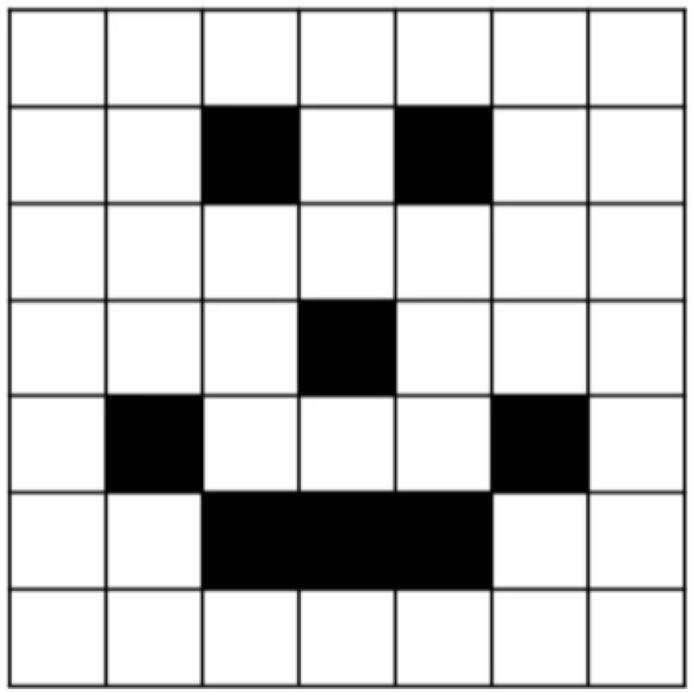
CNN

CNN



CNN


B / W Image 2x2px
Pixel 1
Pixel 2
Pixel 3
Pixel 4
2d array
Pixel 1
0 ≤ pixel value ≤ 255
Pixel 2
0 ≤ pixel value ≤ 255
Pixel 3
0 ≤ pixel value ≤ 255
Pixel 4
0 ≤ pixel value ≤ 255
Colored Image 2x2px
Pixel 1
Pixel 2
Pixel 3
Pixel 4
3d array
Pixel 1
0 ≤ pixel value ≤ 255
Pixel 2
0 ≤ pixel value ≤ 255
Pixel 3
0 ≤ pixel value ≤ 255
Pixel 4
0 ≤ pixel value ≤ 255



0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

CNN - Steps

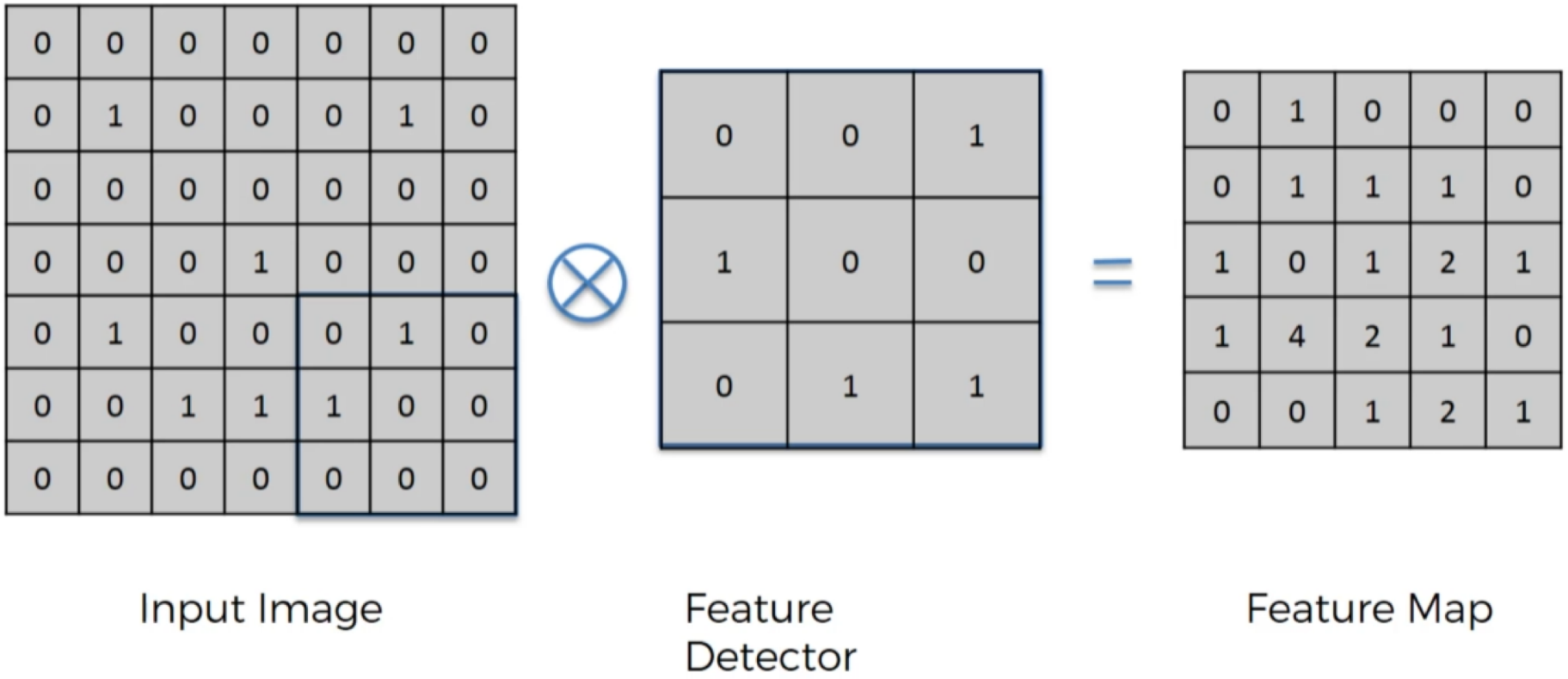
- Step 1: Convolutions
- Step 2: Max Pooling
- Step 3: Fattening
- Step 4: Full Connection

Step 1 – Convolution

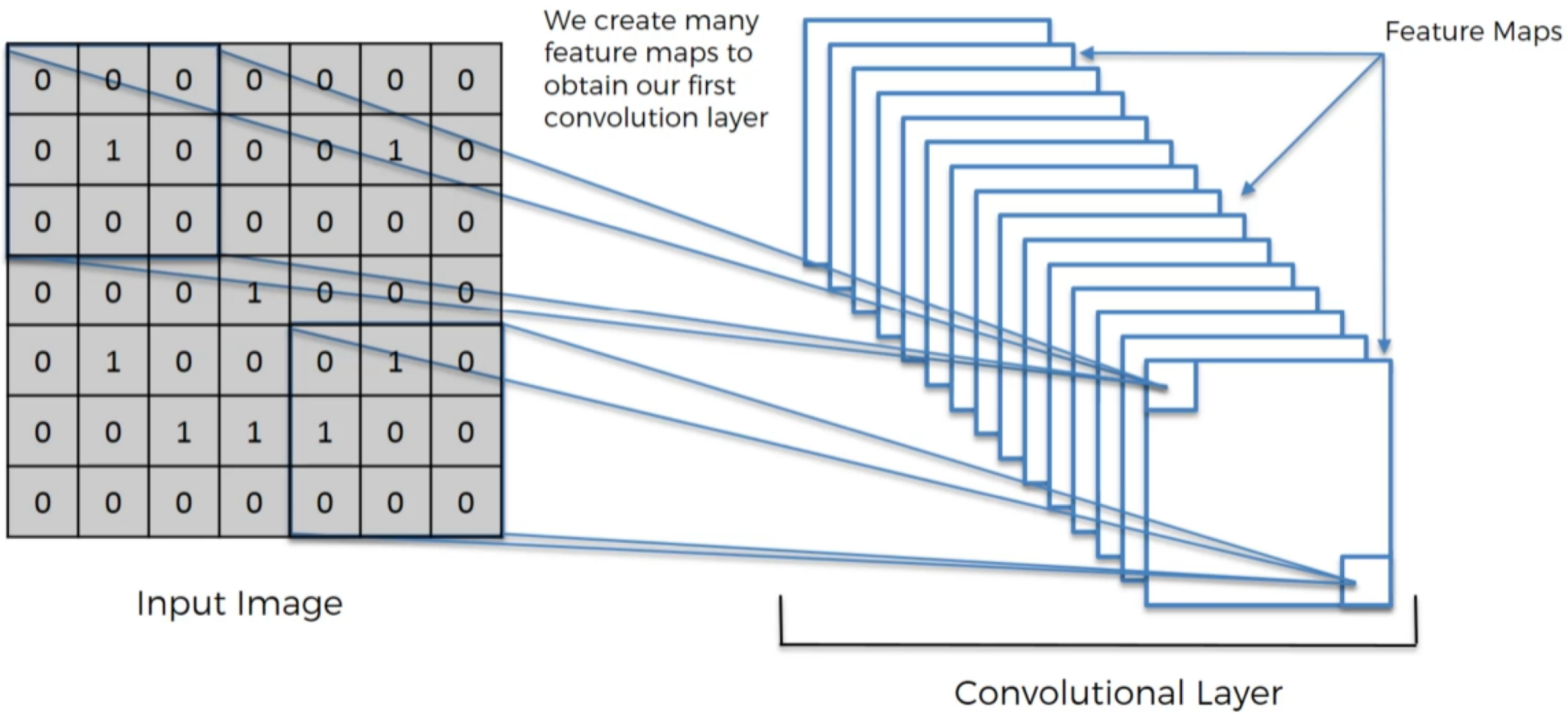
$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

- Convolution is a mathematical operation on two functions (f and g) to produce a third function that expresses how the shape of one modified by the other
- convolution refers to both the result function and to the process of computing it

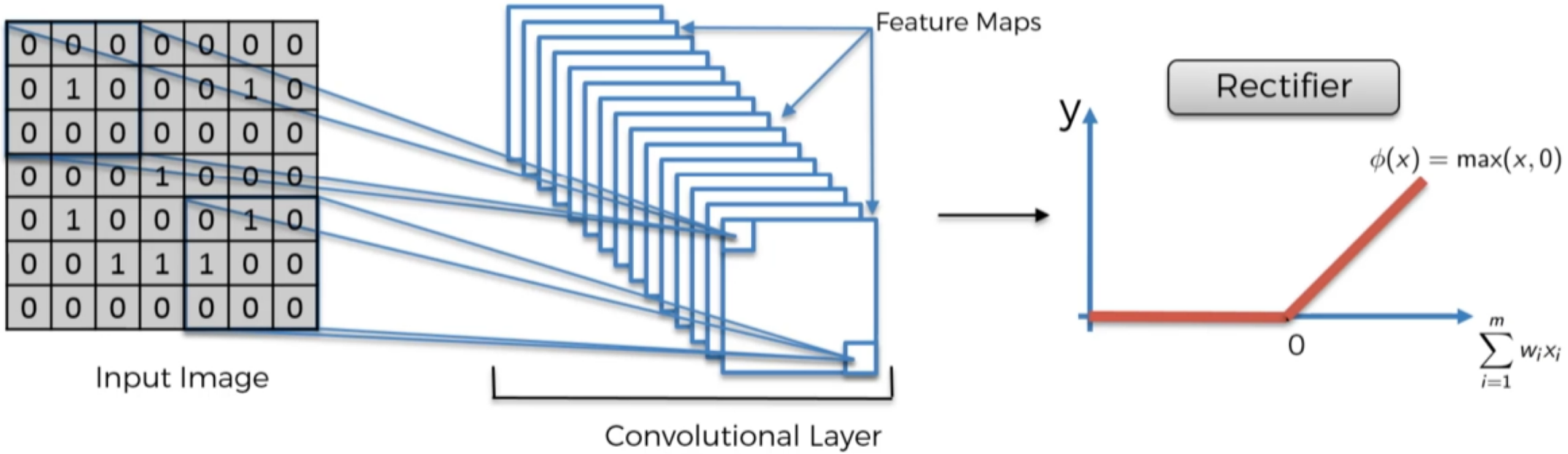
Step 1 – Convolution



Step 1 – Convolution



Step 1 (B) – ReLu Layer



Step 2 – Max Pooling

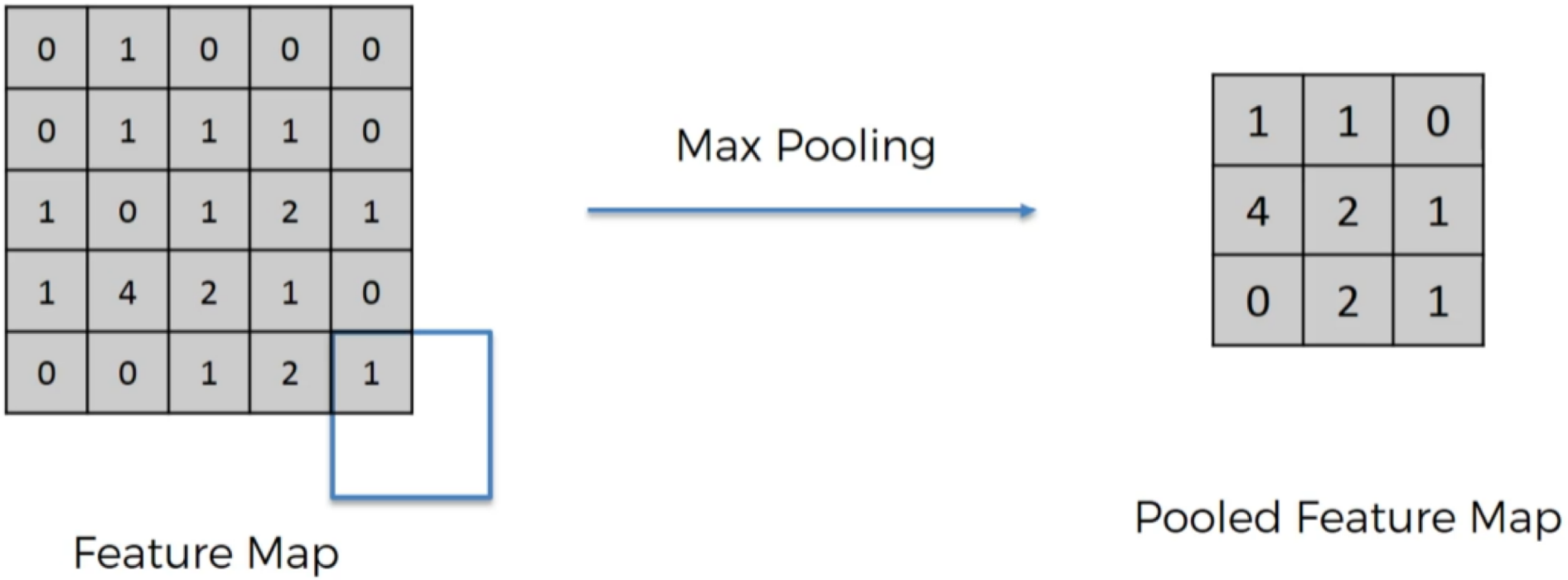




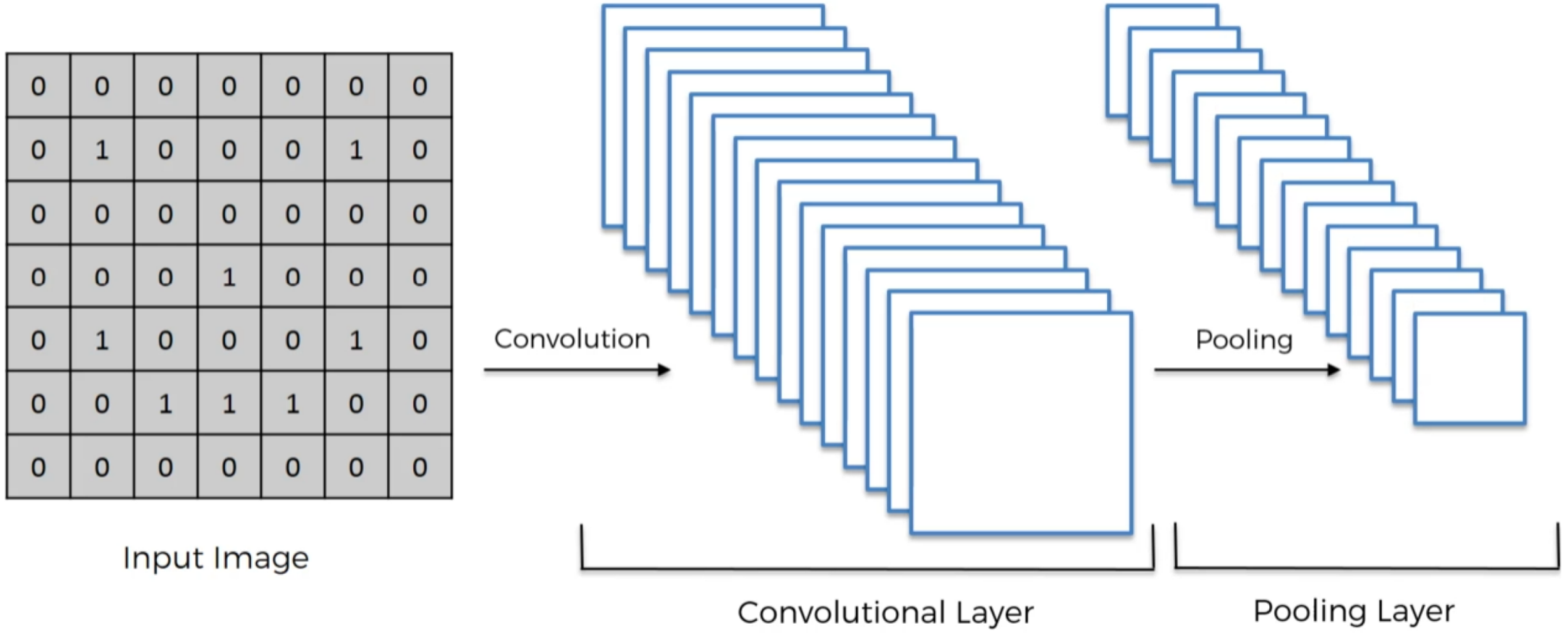




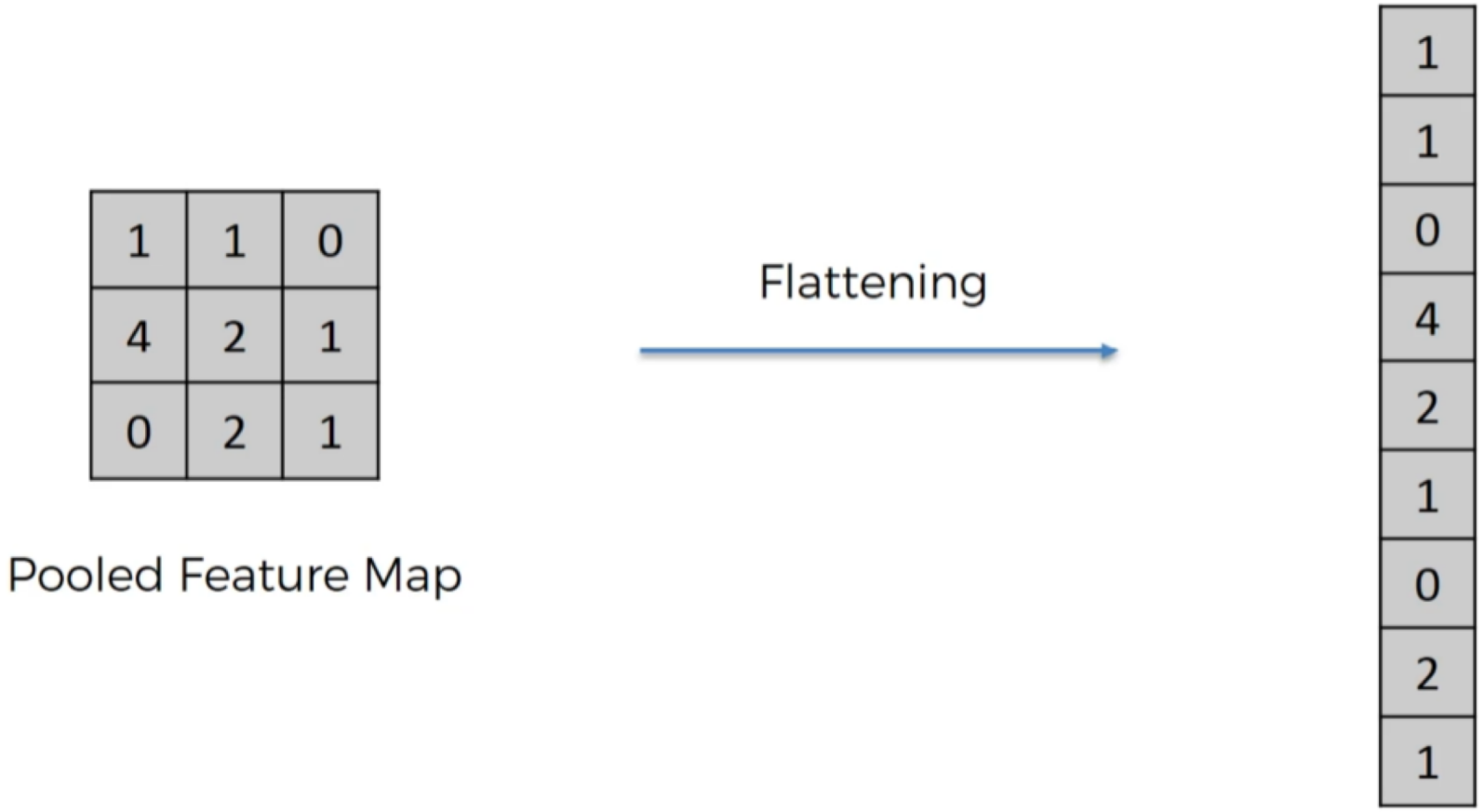
Step 2 – Max Pooling



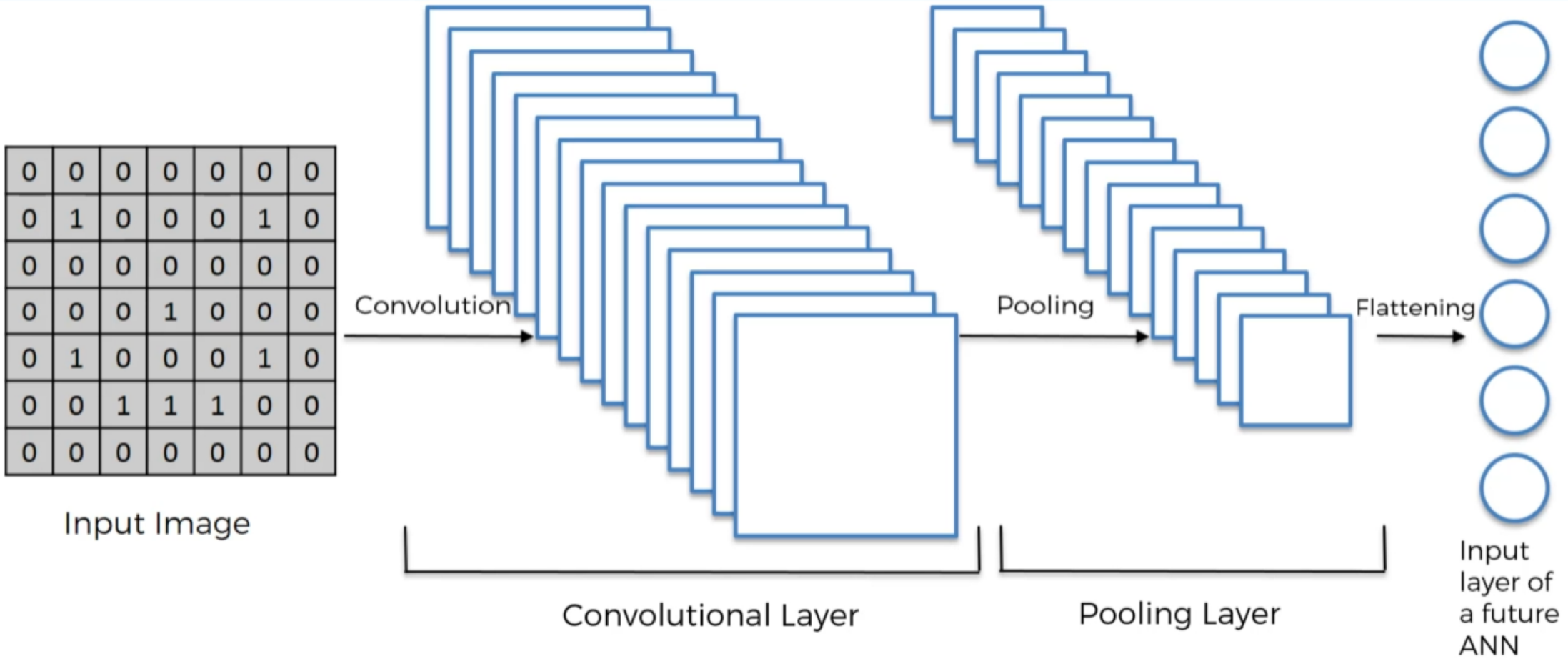
Step 2 – Max Pooling



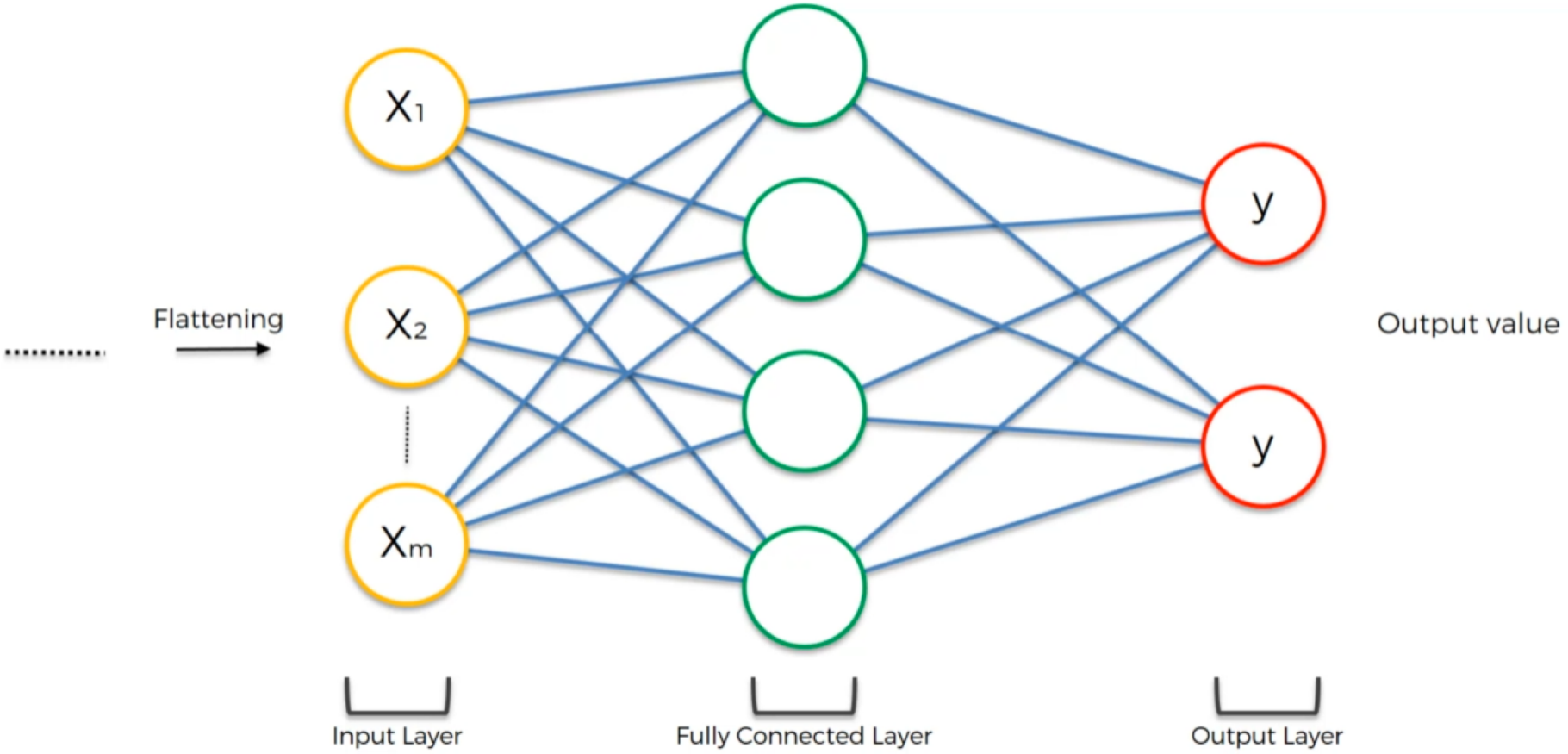
Step 3 – Flattening



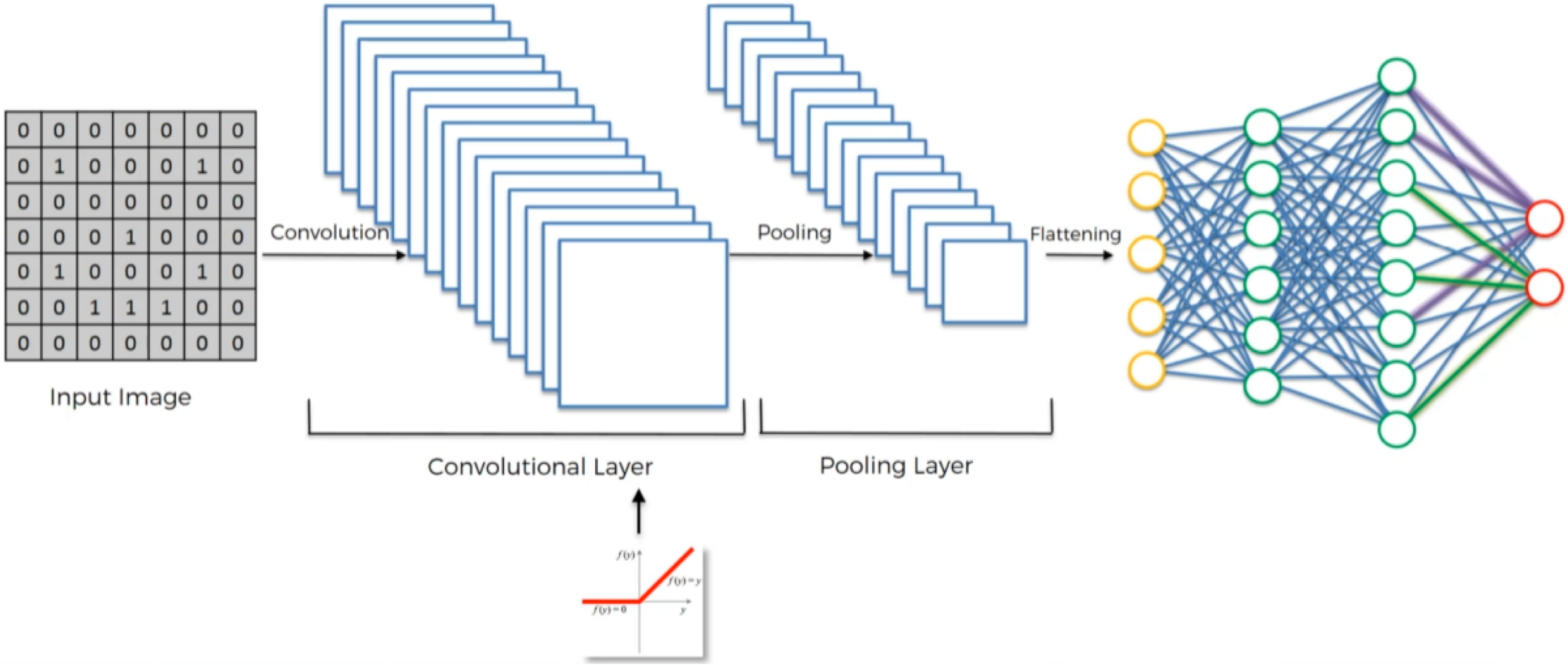
Step 3 – Flattening



Step 4 – Full Connection



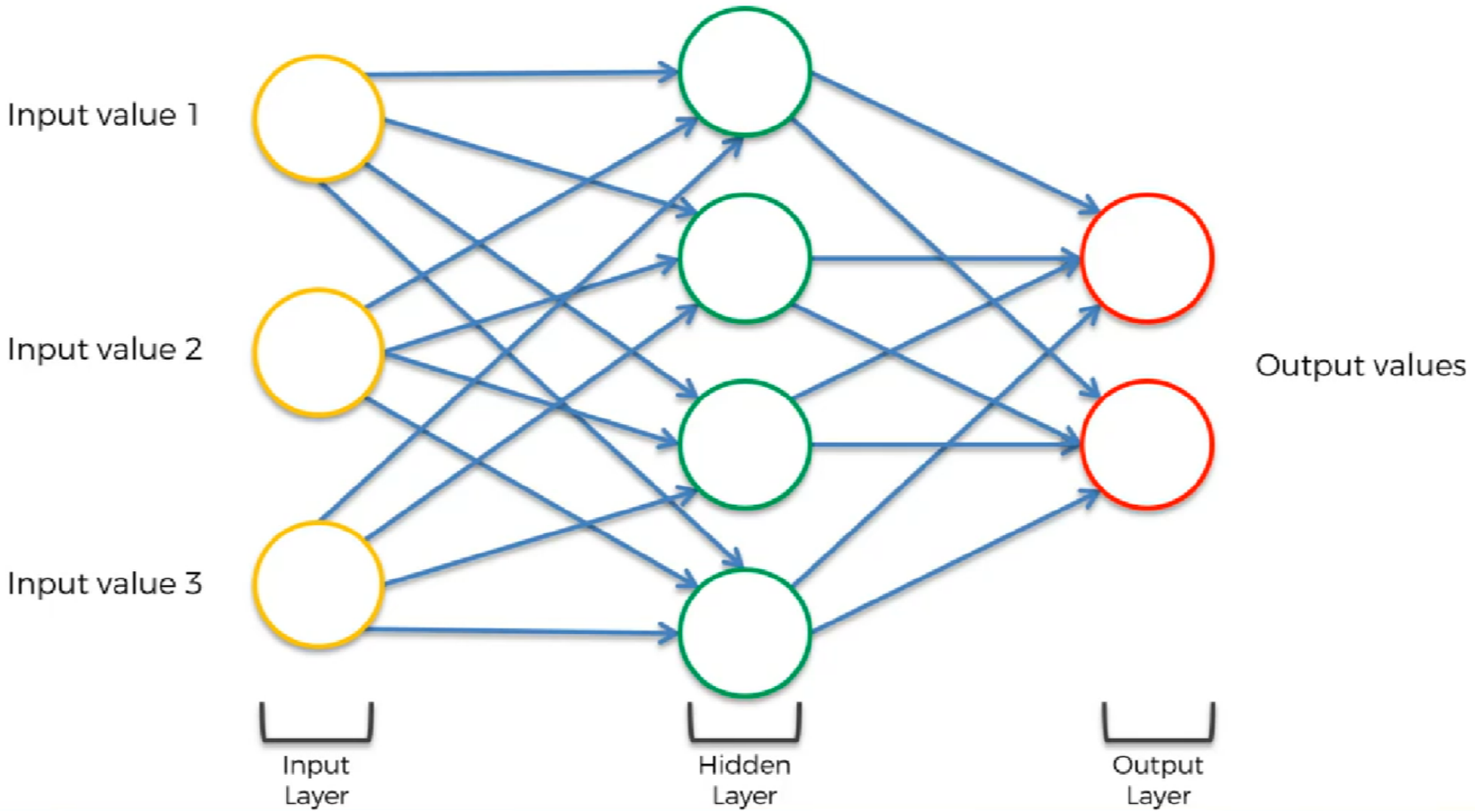
Step 4 – Full Connection



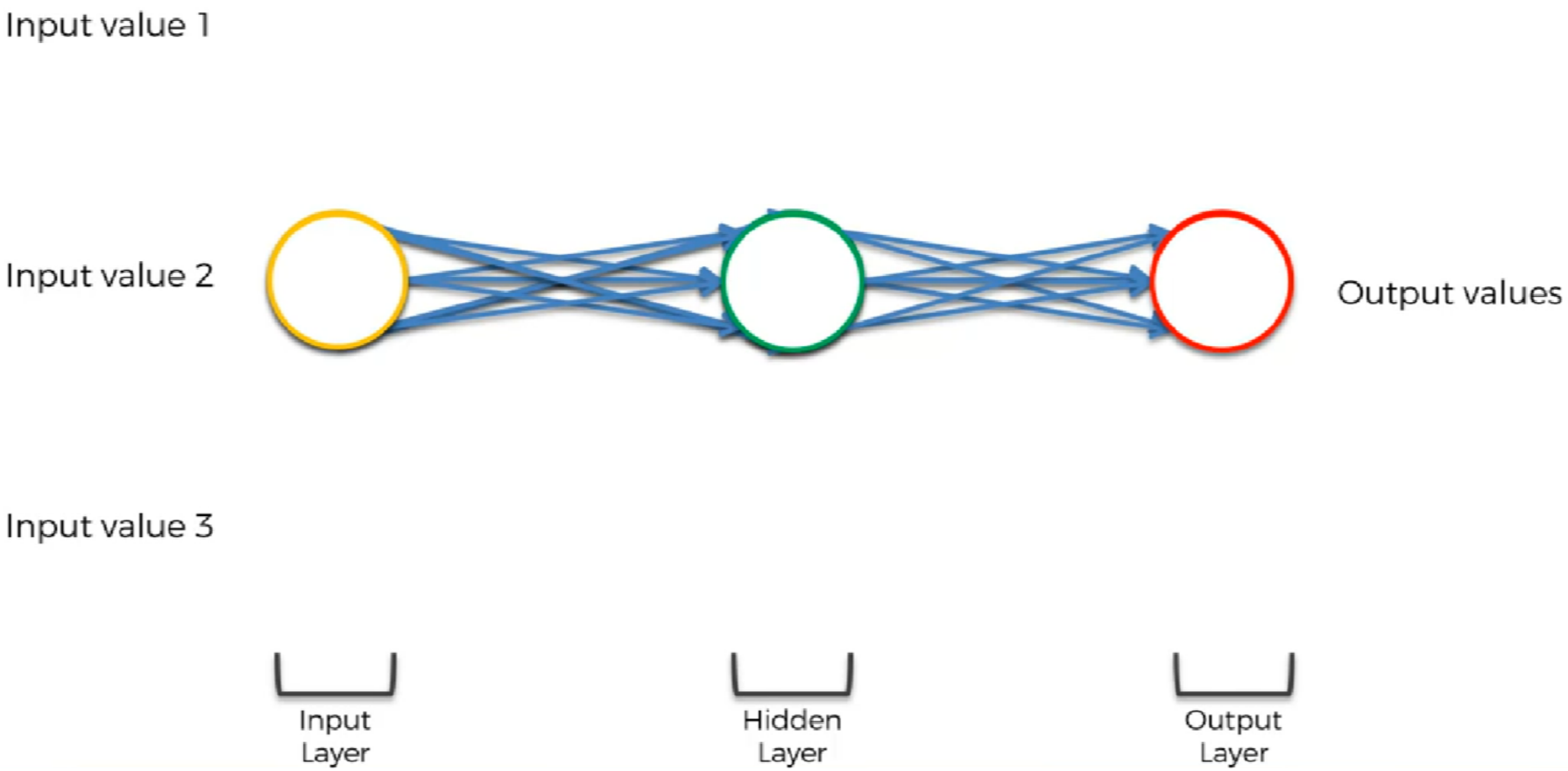
RNN

Recurrent Neural Network

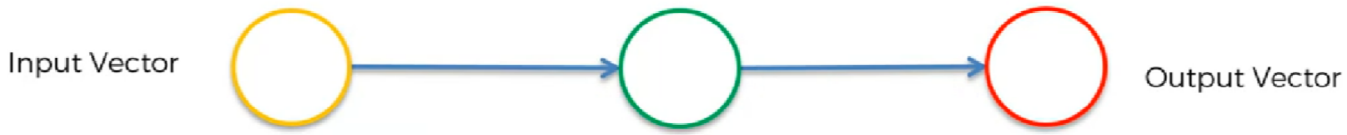
NN

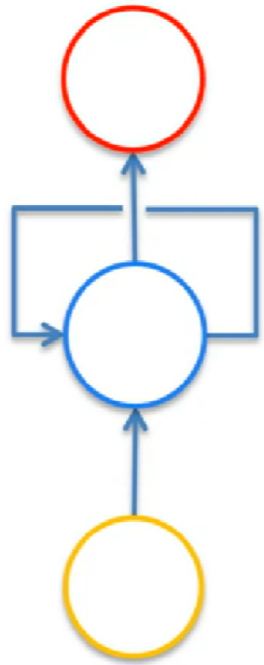

Input value 1
Input value 2
Input value 3
Output values
Input Layer
Hidden Layer
Output Layer

RNN

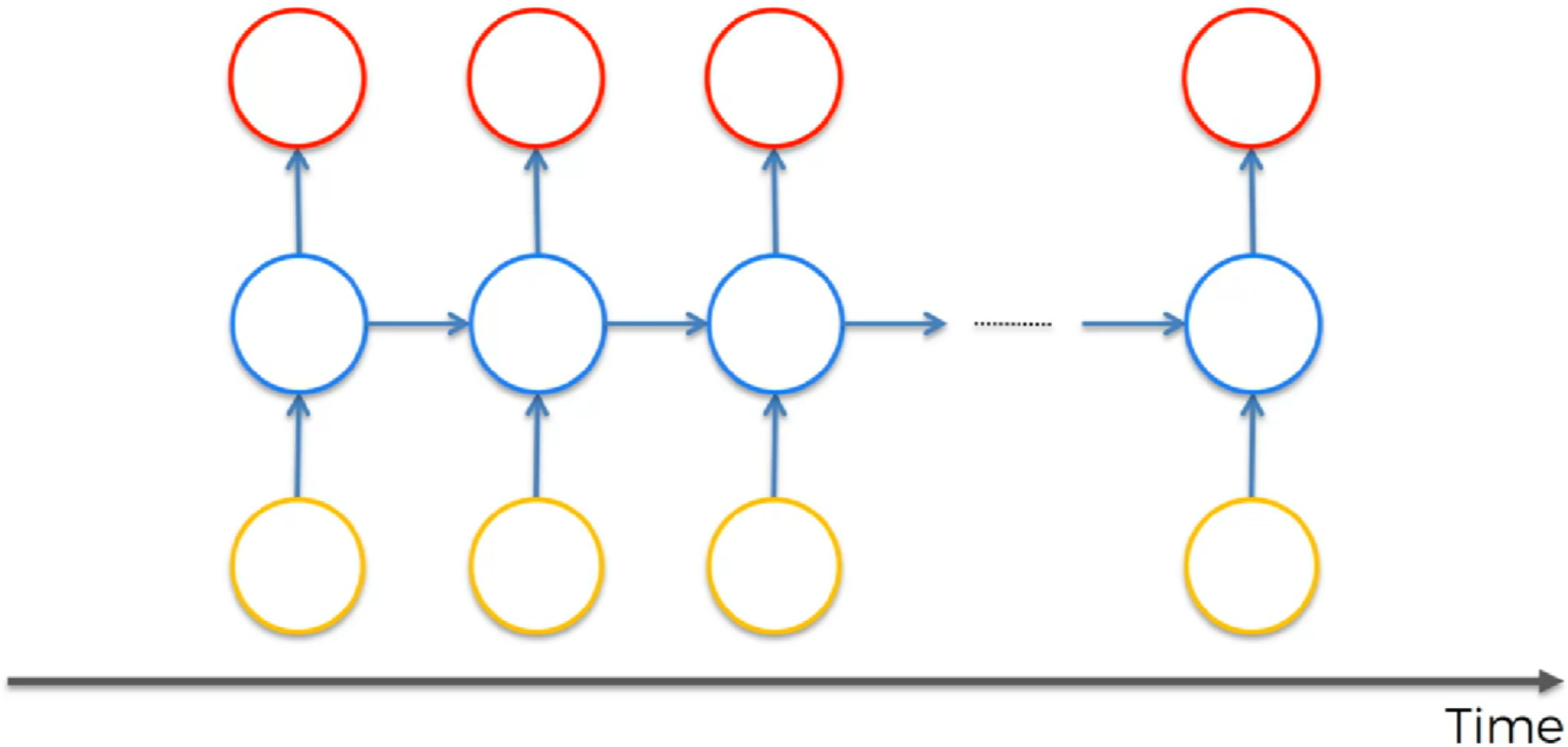

Input value 1
Input value 2
Input value 3
Output values
Input Layer
Hidden Layer
Output Layer

RNN

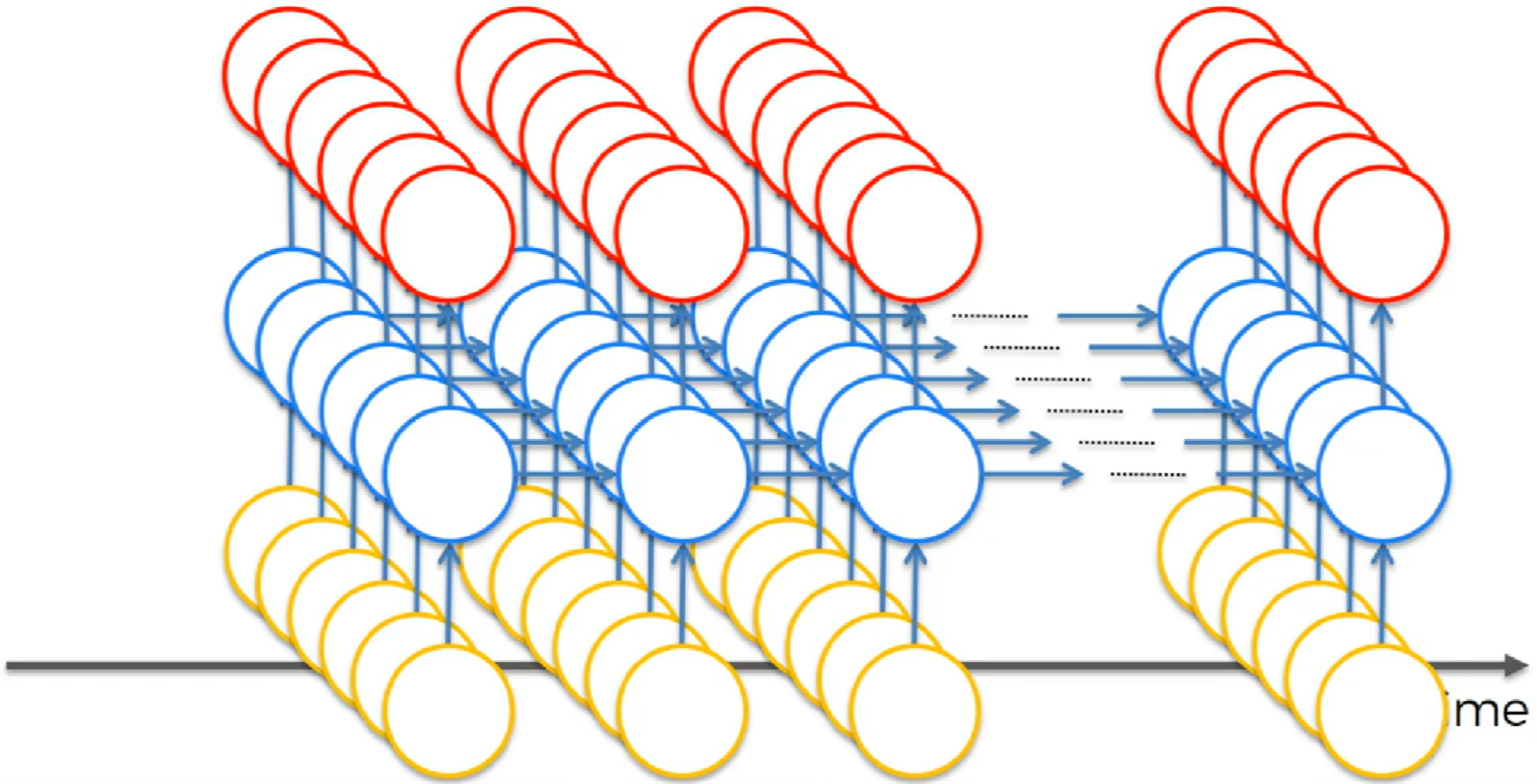




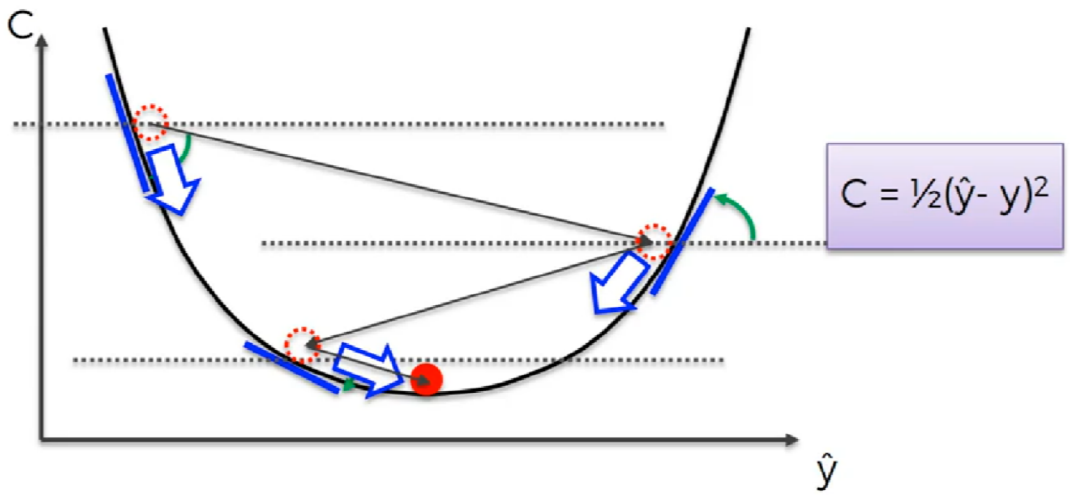
RNN

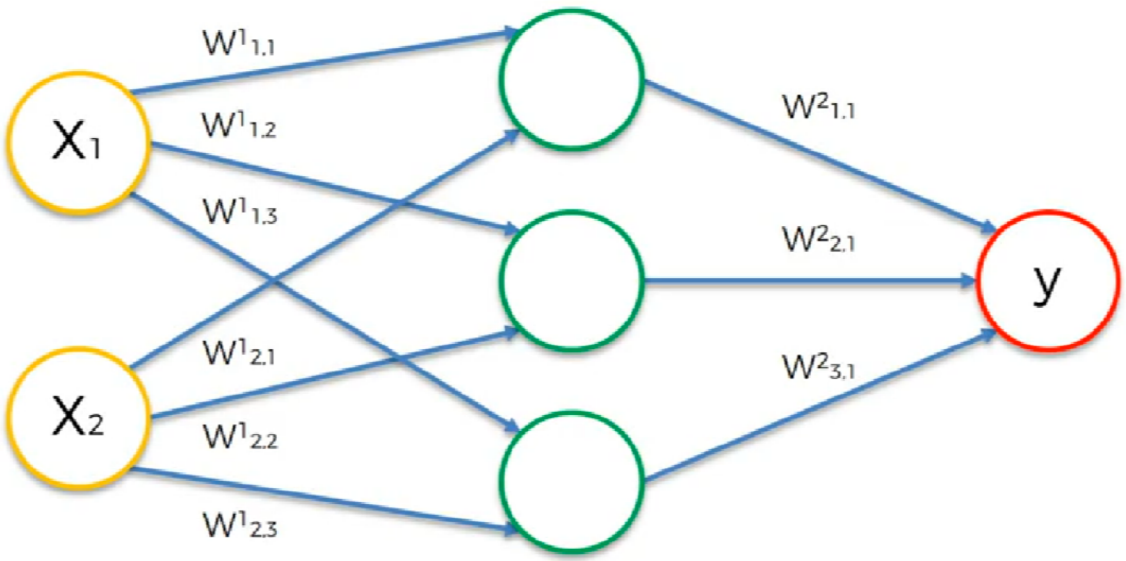


RNN

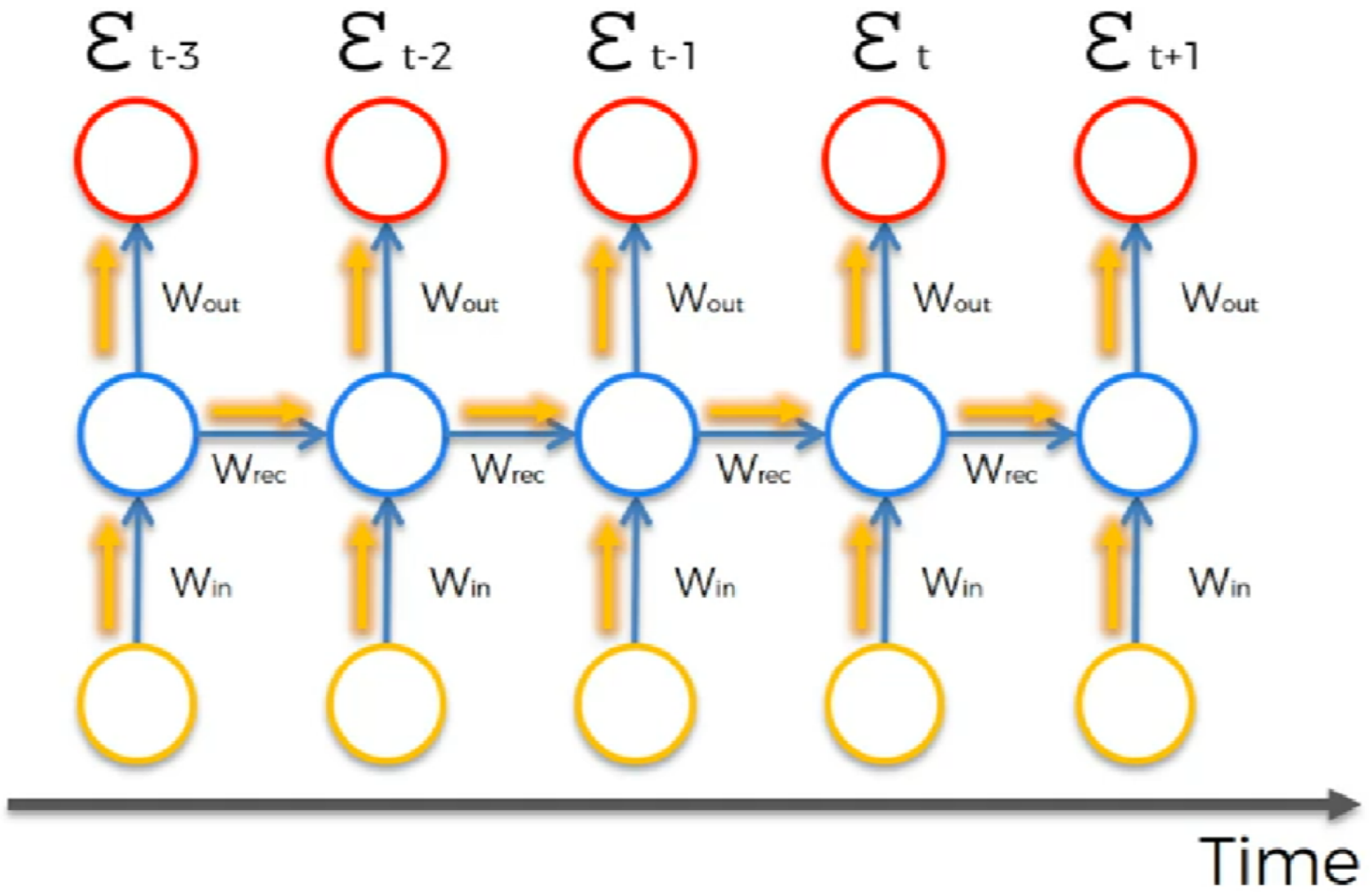


Vanishing Gradient Problem

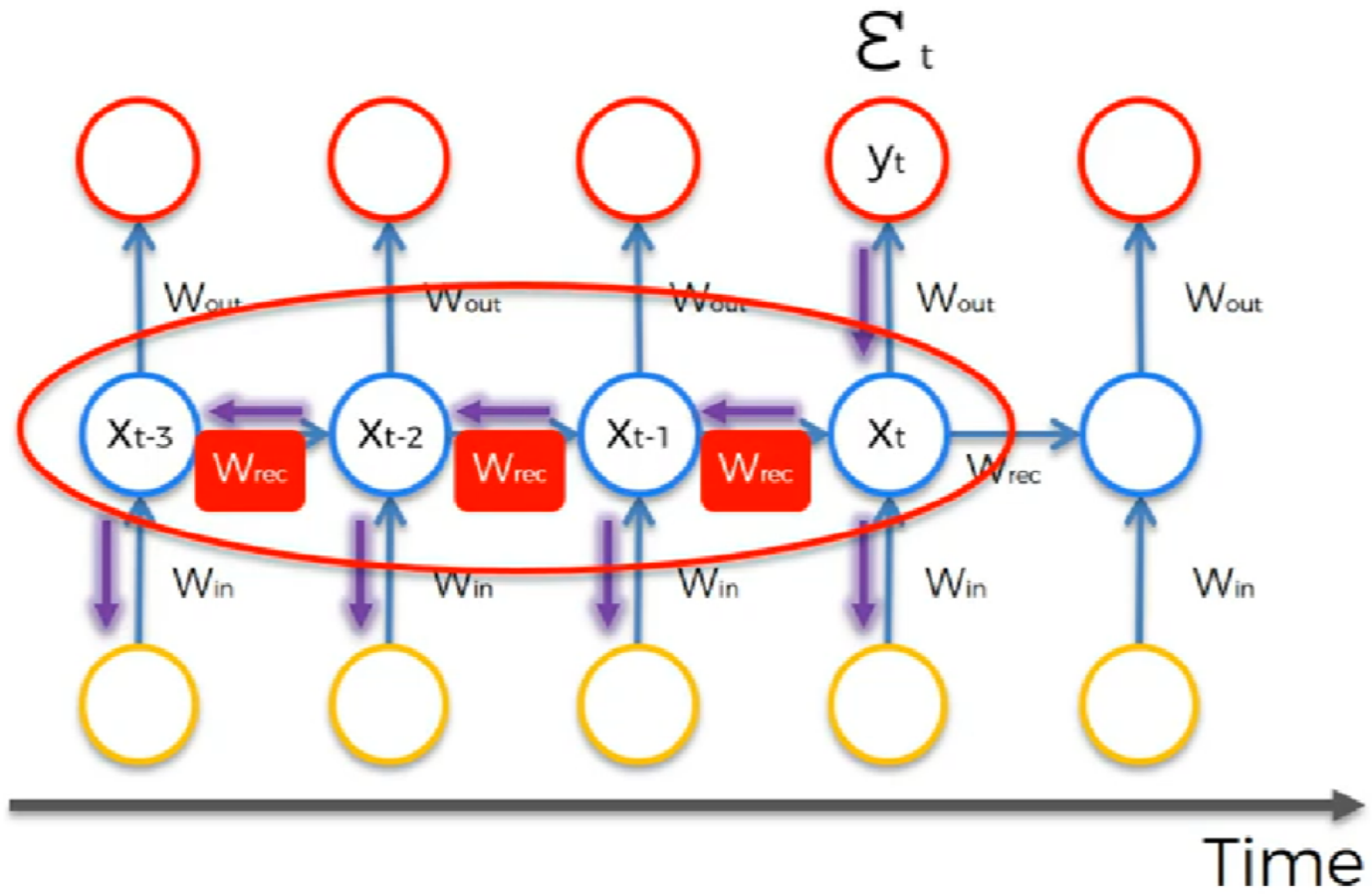




Vanishing Gradient Problem



Vanishing Gradient Problem



W_{rec} ~ small ⇨ Vanishing

W_{rec} ~ large ⇨ Exploding

Vanishing Gradient Problem - Solutions

- Exploding Gradient
  - Truncated Backpropagation
  - Penalties
  - Gradient Clipping
- Vanishing Gradient
  - Weight initialization
  - Echo State Network
  - Long Short-Term Memory Networks (LSTMs)