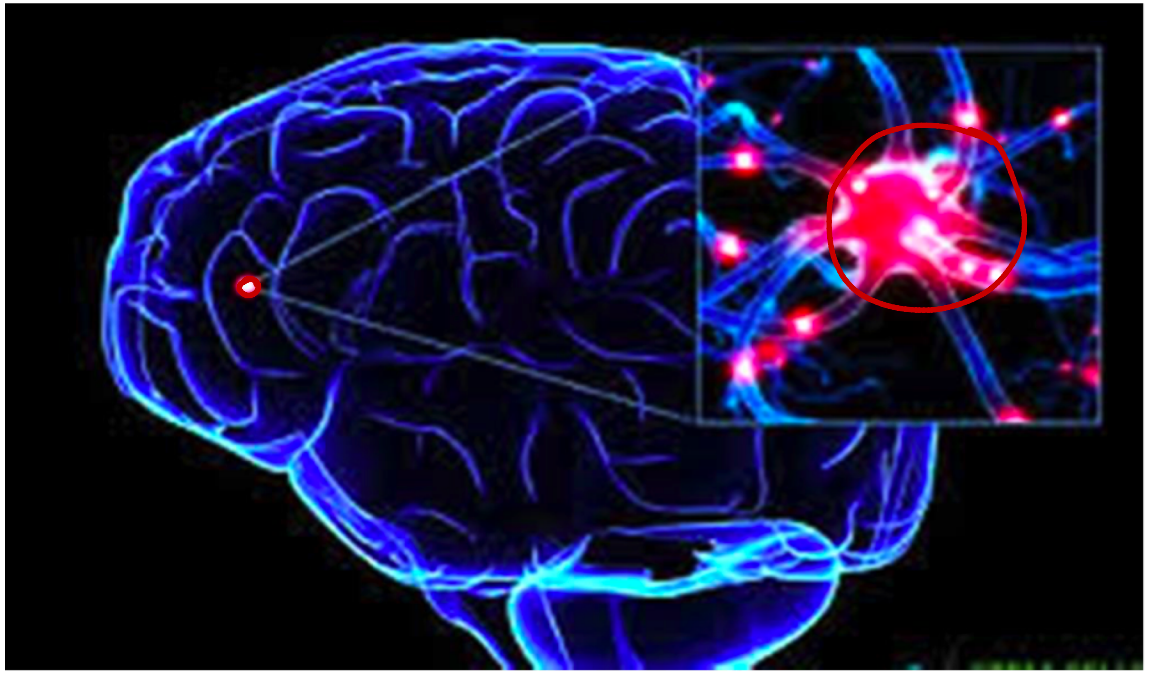
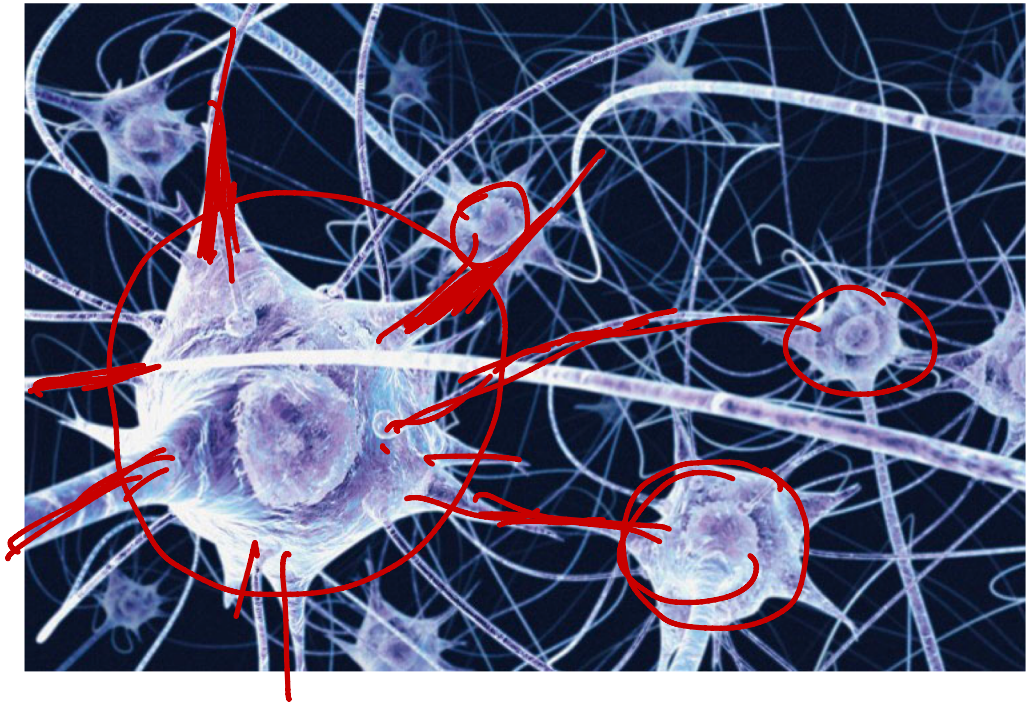
AI

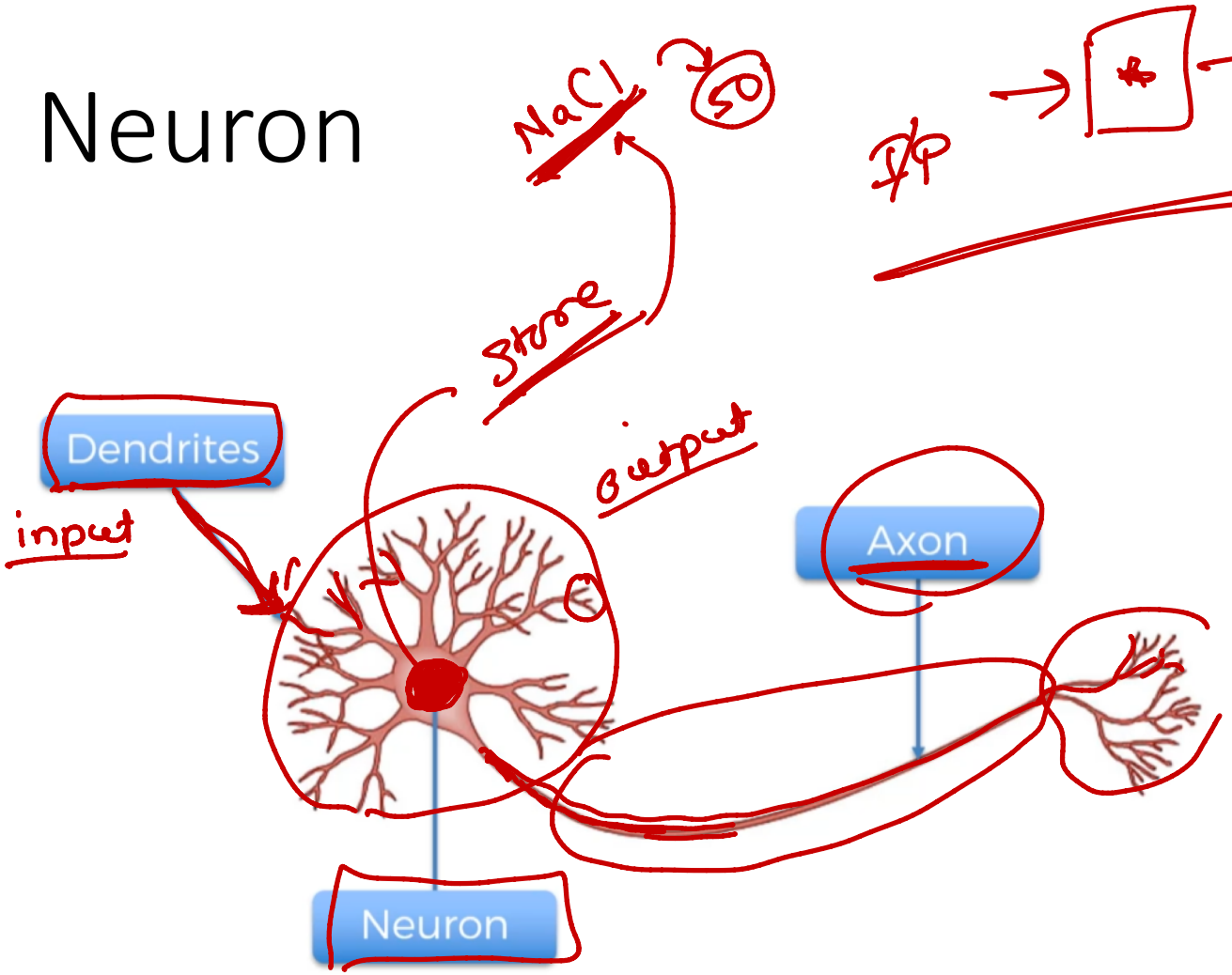
Artificial Intelligent

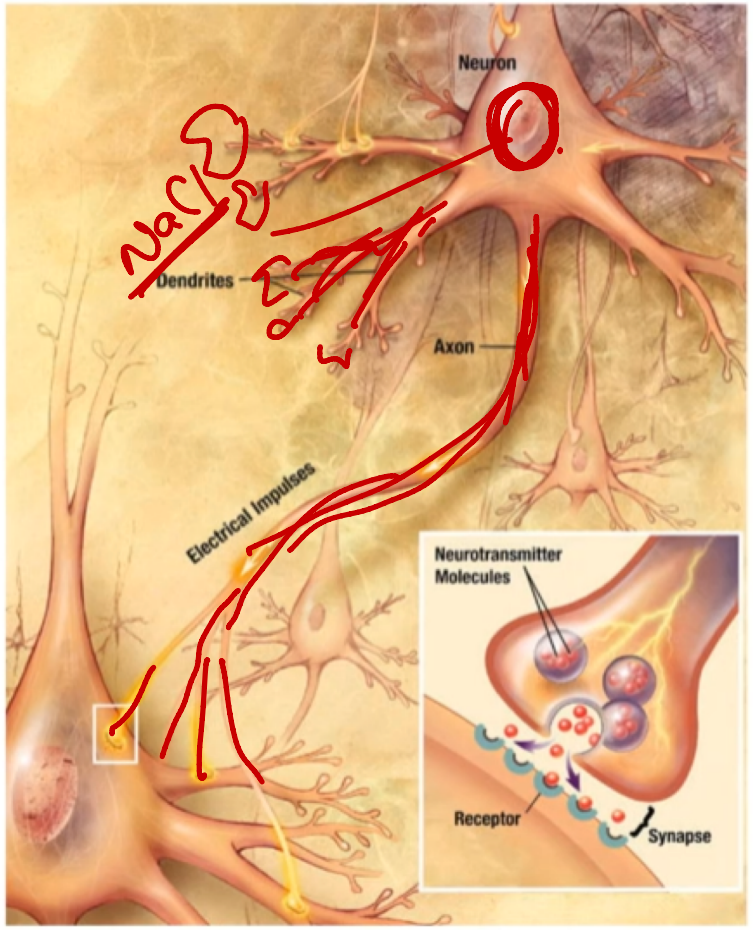
Neural Network



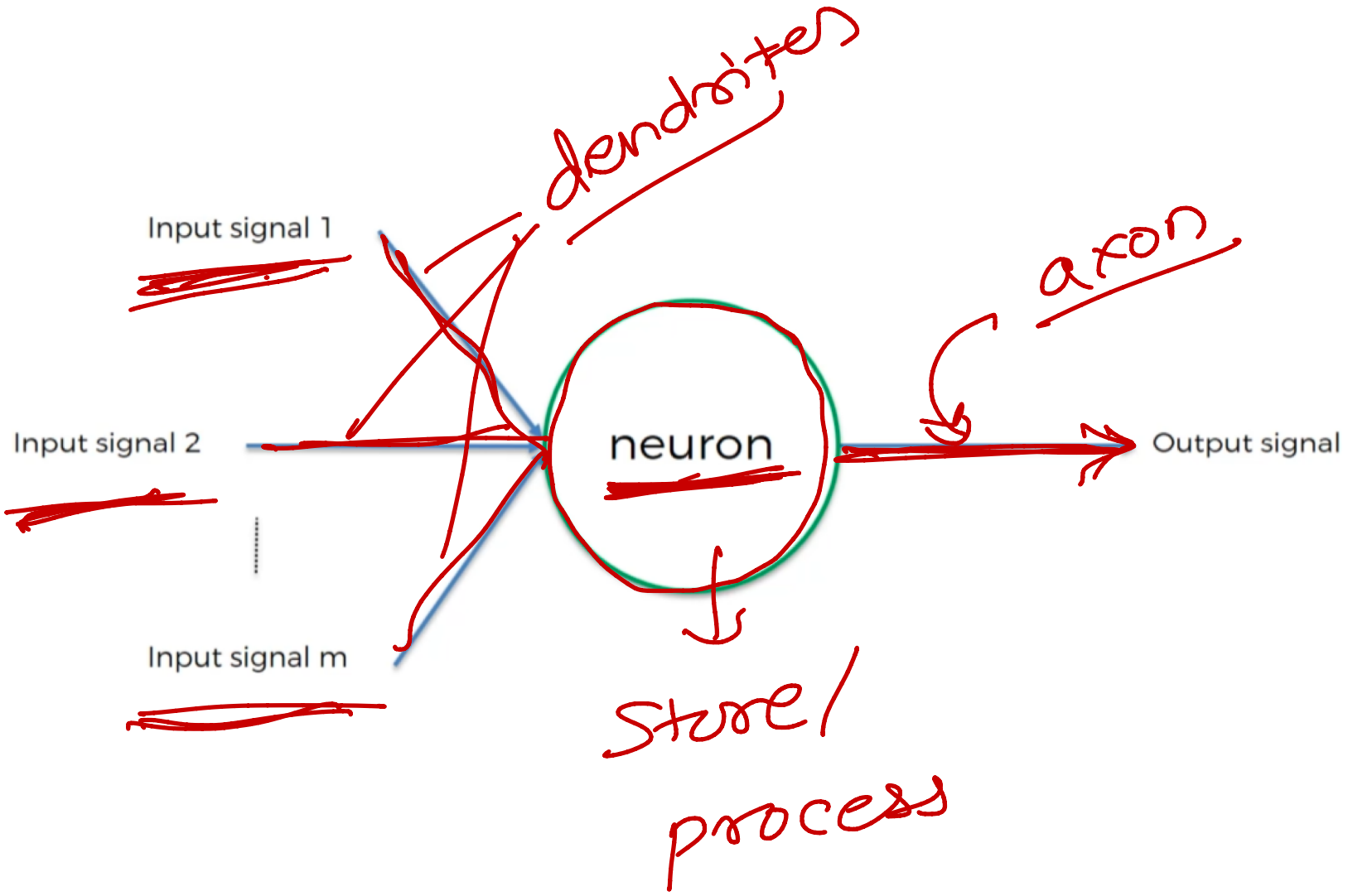


Neuron

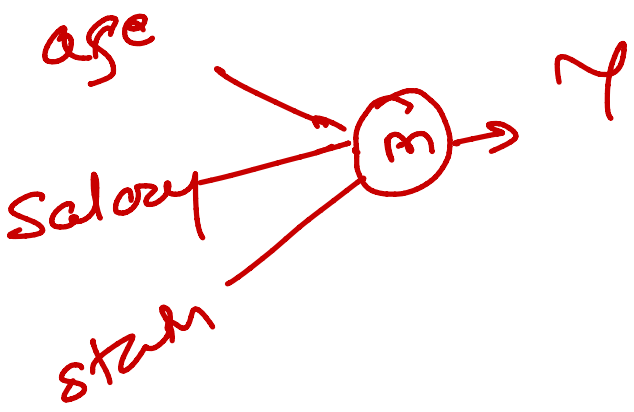

Dendrites
Axon
Neuron


Neuron
Dendrites
Axon
Electrical Impulses
Neurotransmitter Molecules
Receptor
Synapse

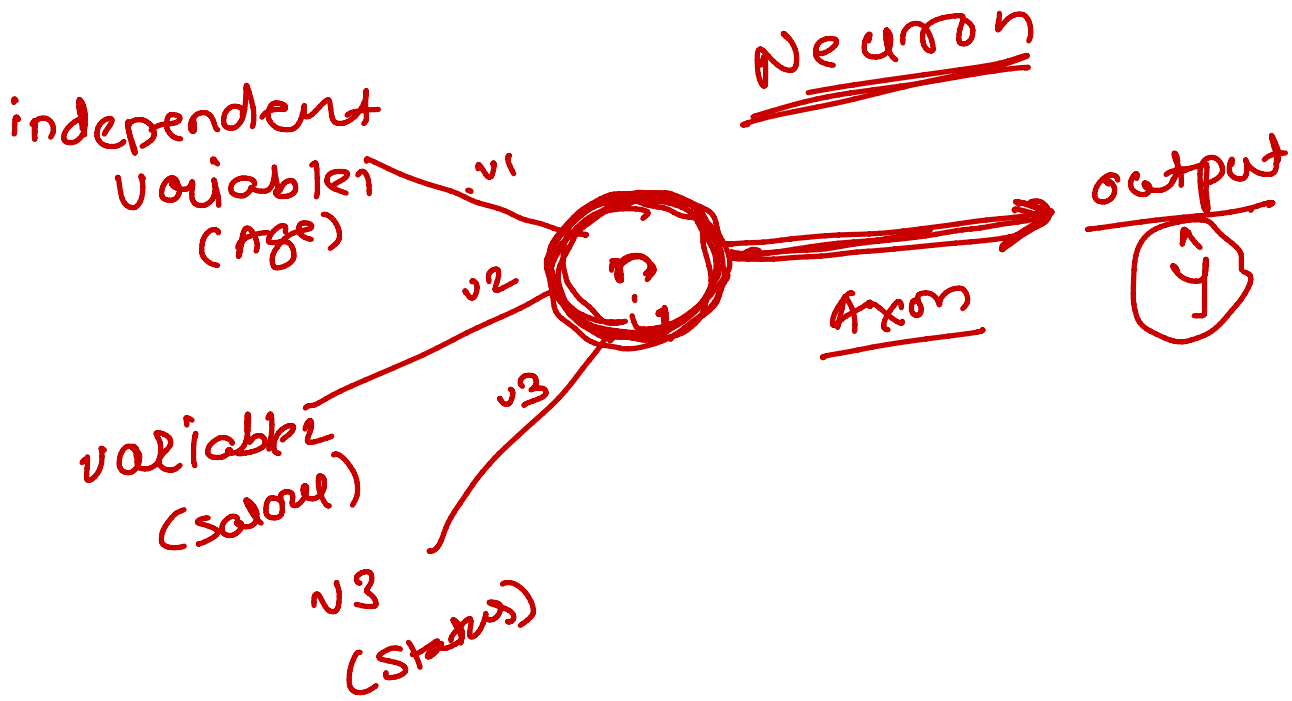
Neuron

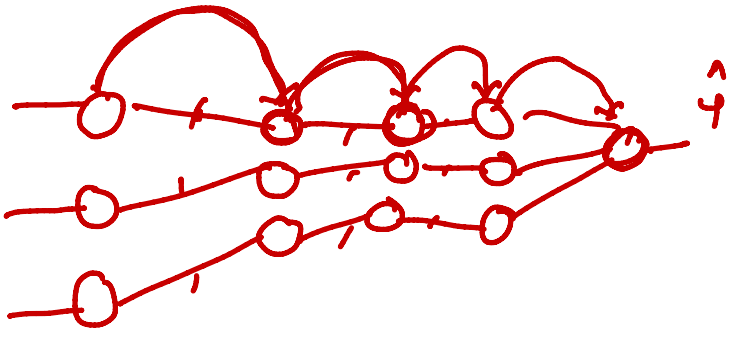

dendrites
Input signal 1
Input signal 2
Input signal m
neuron
axon
Output signal
store/
process


X
Y

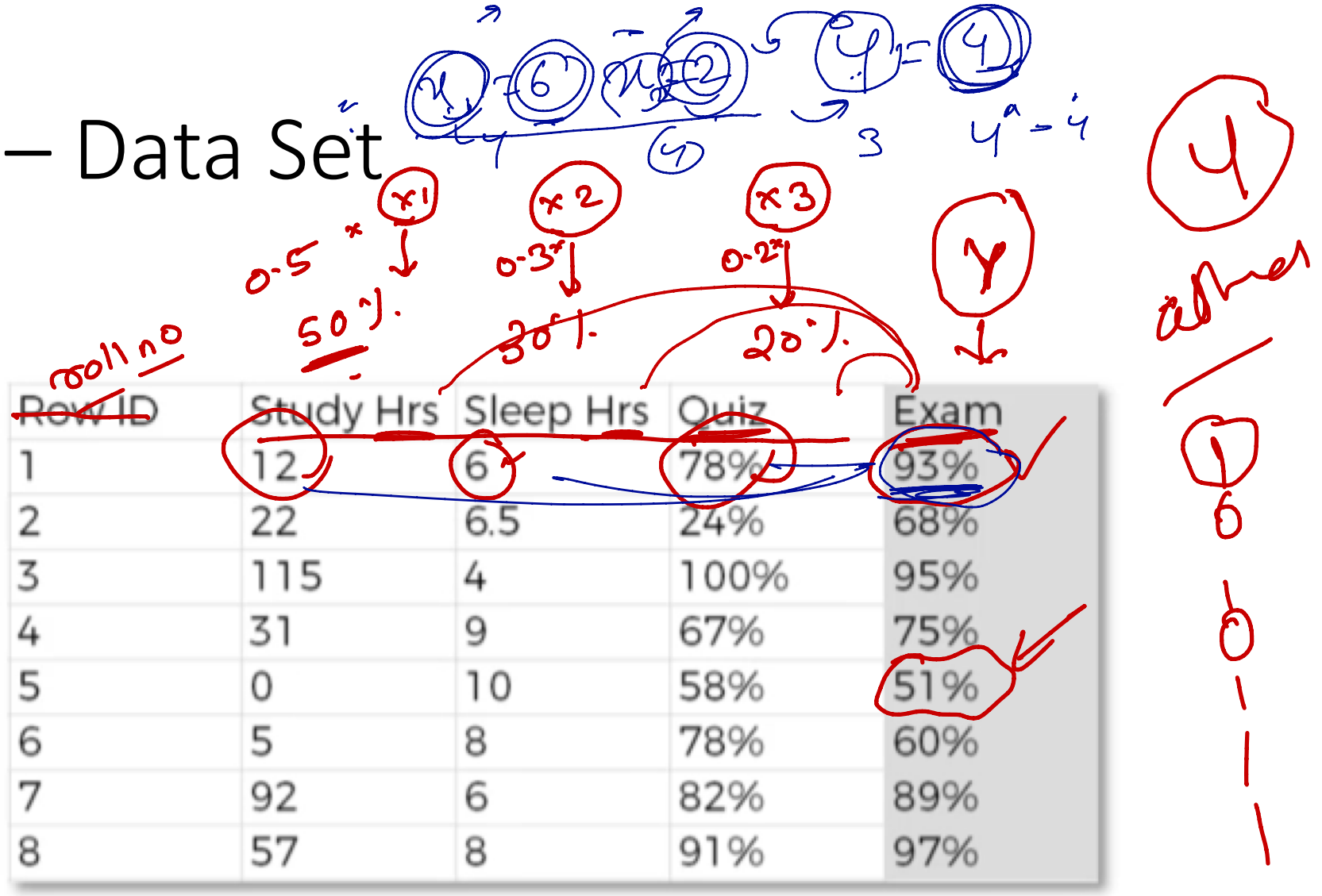

age
Salary
Status
Y


X1
X2


Neuron
independent
Variables
(Age)
v1
v2
v3
variable
(Salary)
v3
(Status)
output
Axon

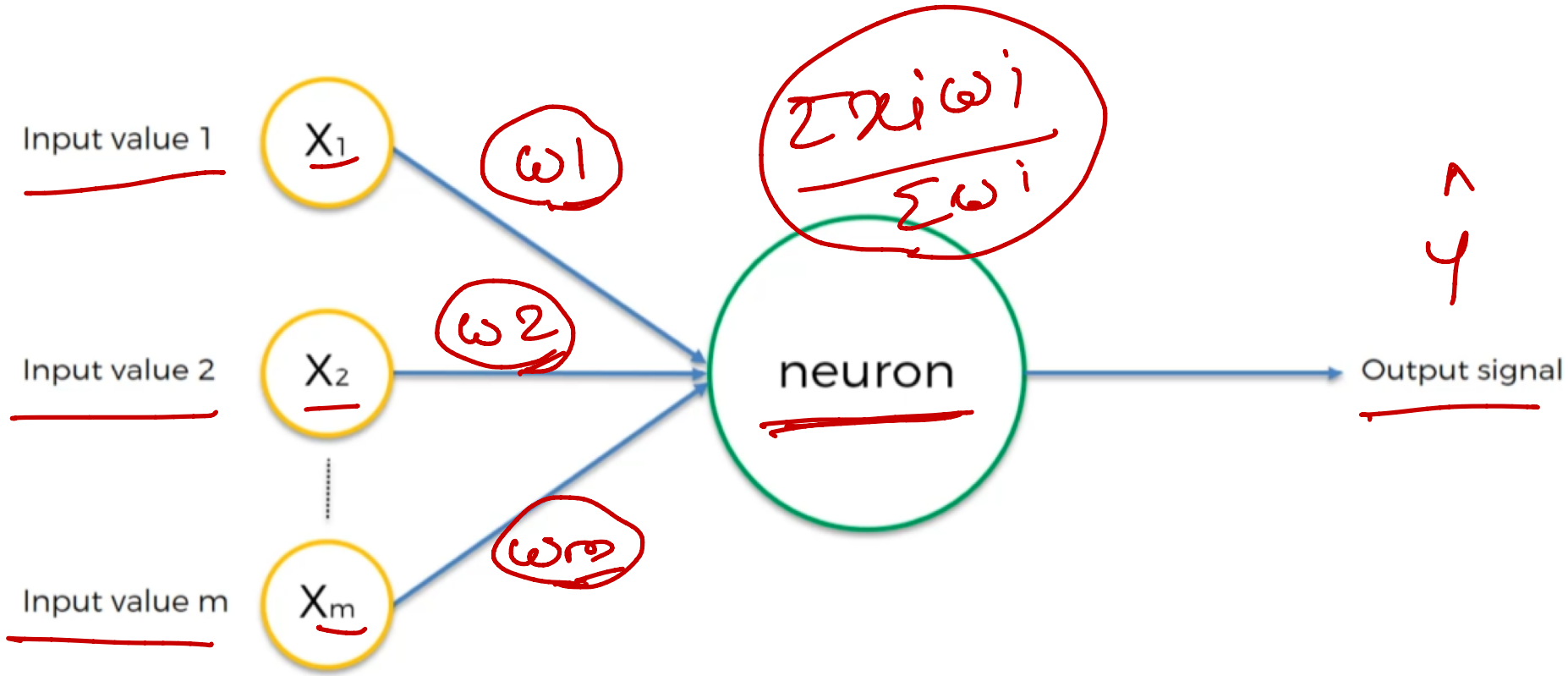


Neuron – Data Set

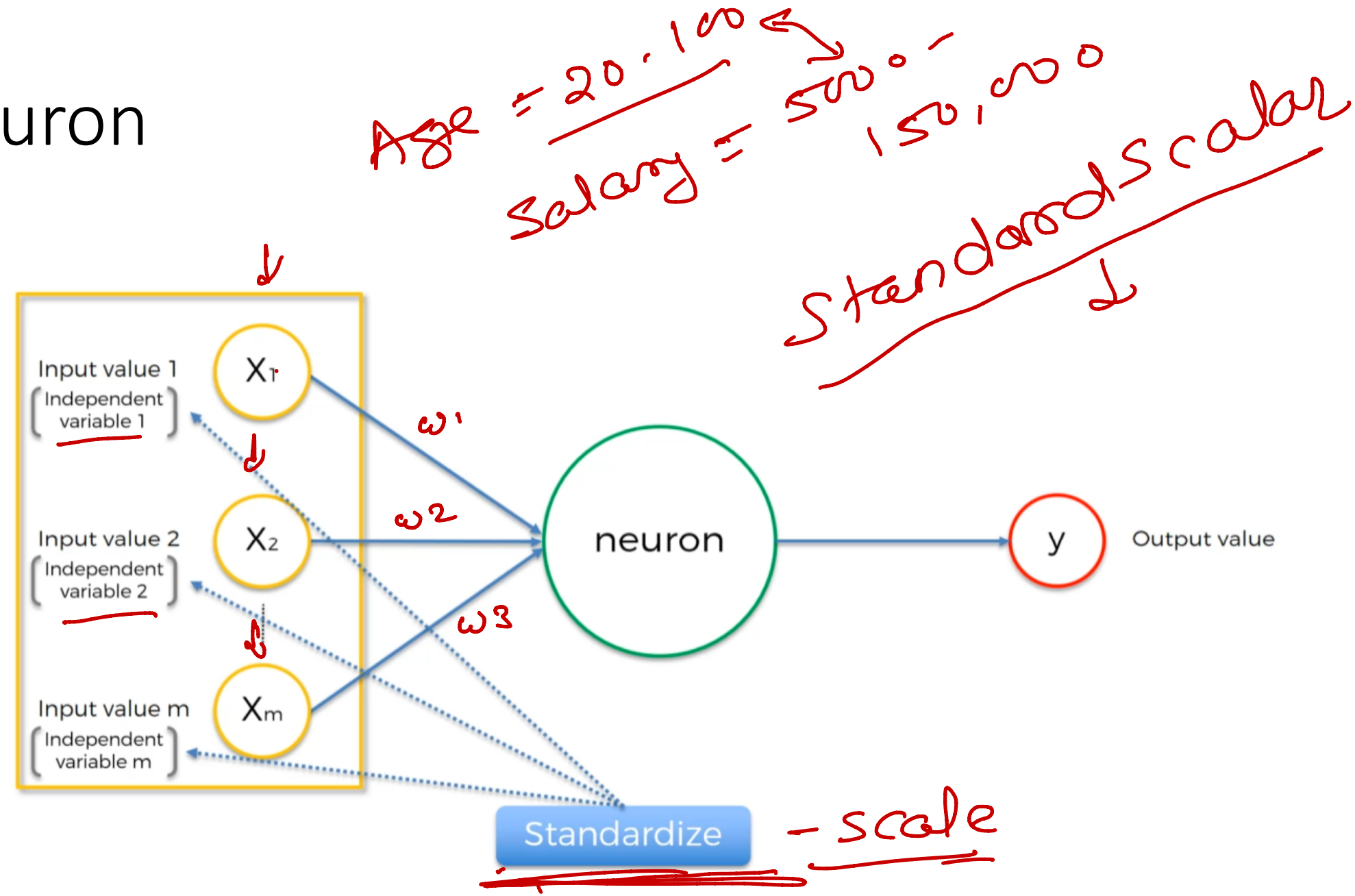


~~Row ID~~	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

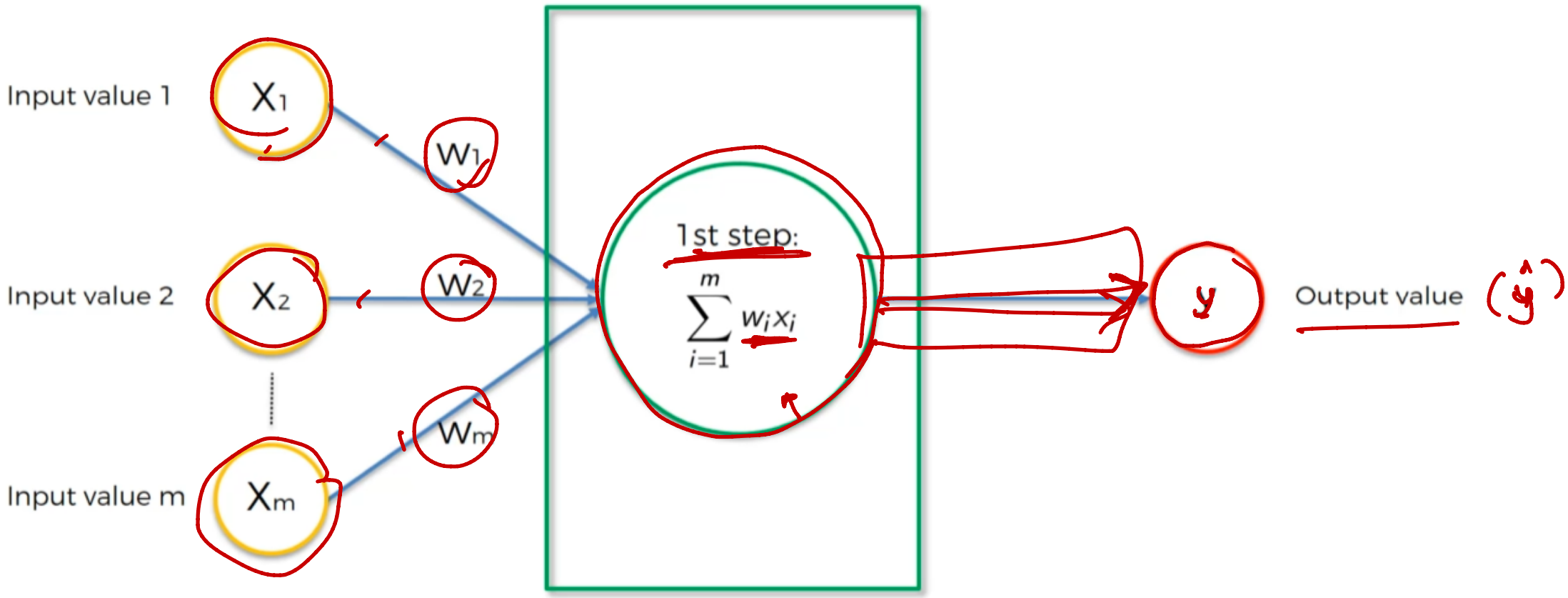
Neuron


Input value 1
X1
ω1
Input value 2
X2
ω2
Input value m
Xm
ωm
neuron
Output signal
ŷ

Neuron

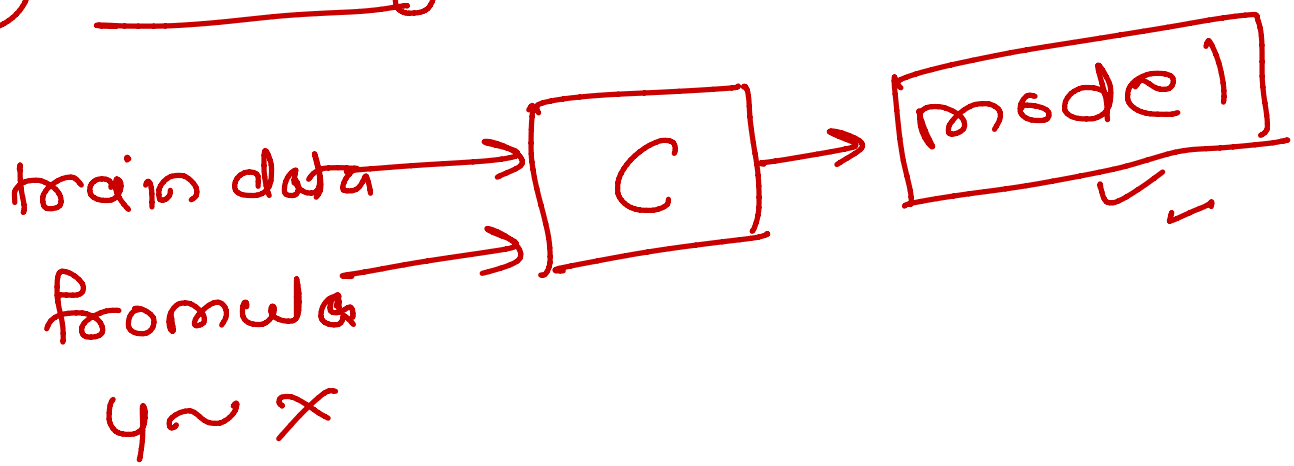

Age = 20 - 100
Salary = 5000 - 150,000
Standard scalar
Input value 1
Independent variable 1
Input value 2
Independent variable 2
Input value m
Independent variable m
X1
X2
Xm
w1
w2
w3
neuron
y
Output value
Standardize
- scale

Neuron

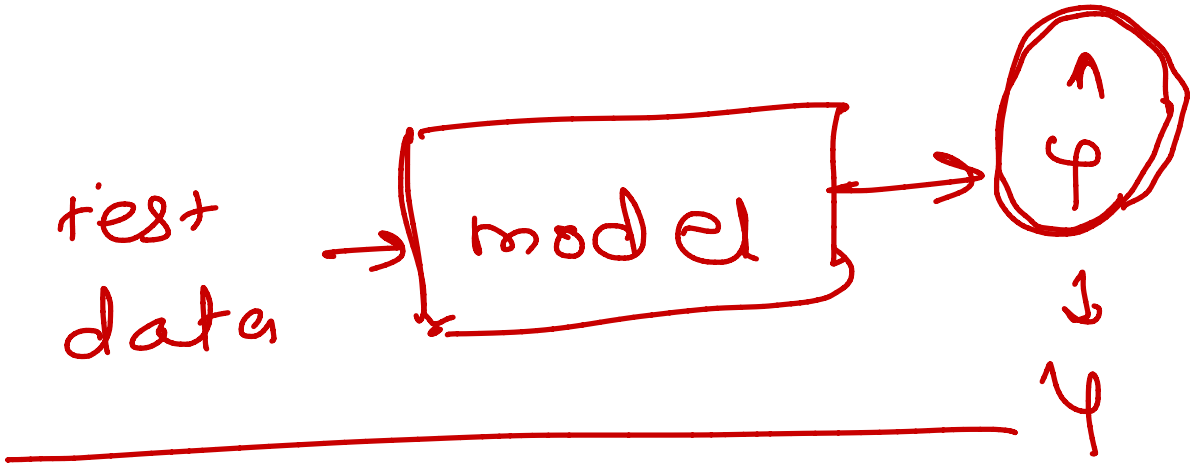

Input value 1
x_1
w_1
Input value 2
x_2
w_2
Input value m
x_m
w_m
1st step:
$\sum_{i=1}^{m} w_i x_i$
y
Output value
$\hat{y}$

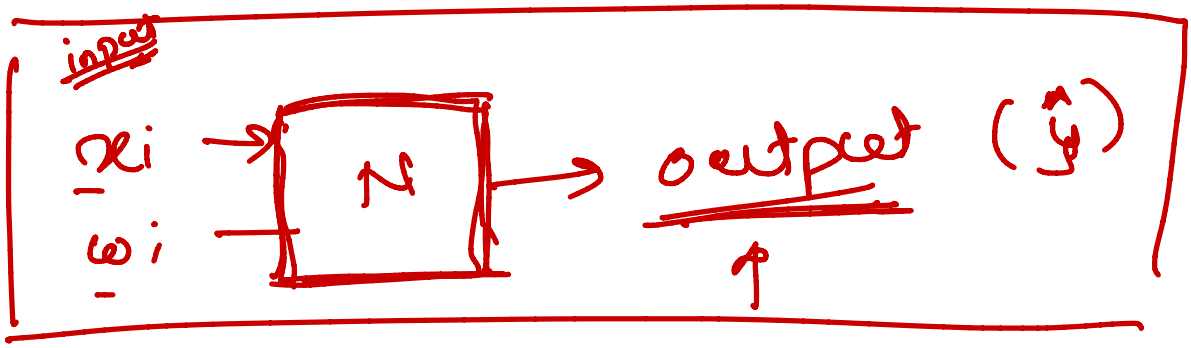
ML

① training



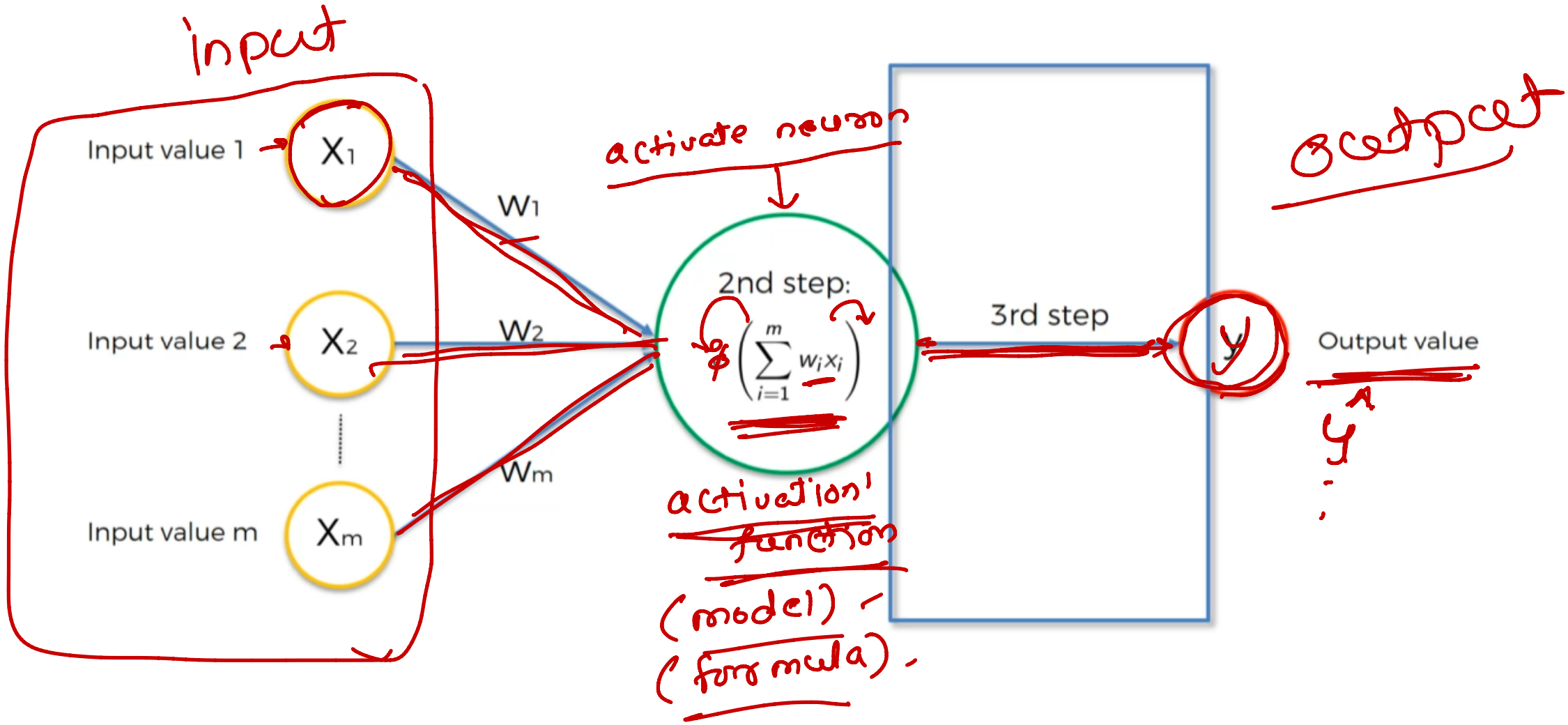
② testing / validate





Activation Function

Perceptron - 1 neuron

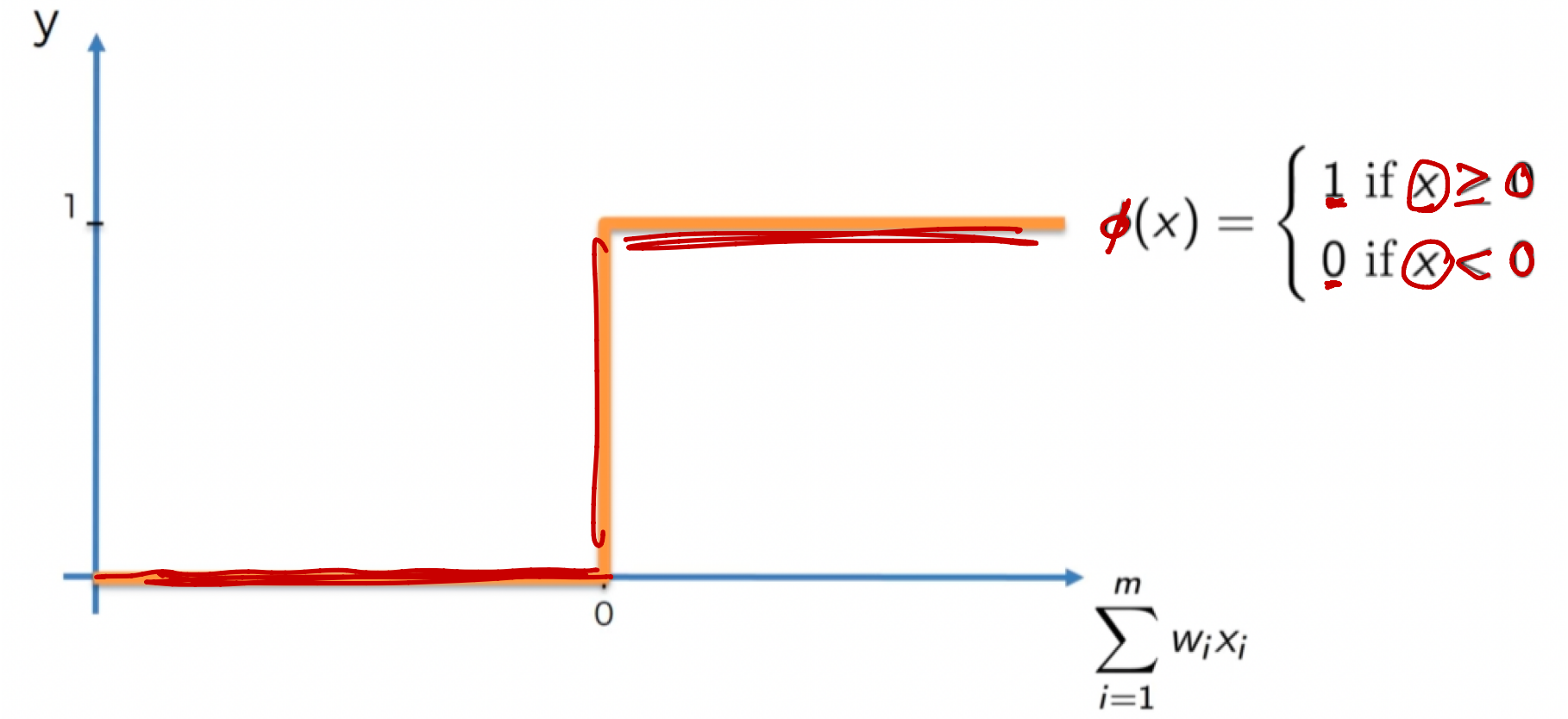


Activation Function

- 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

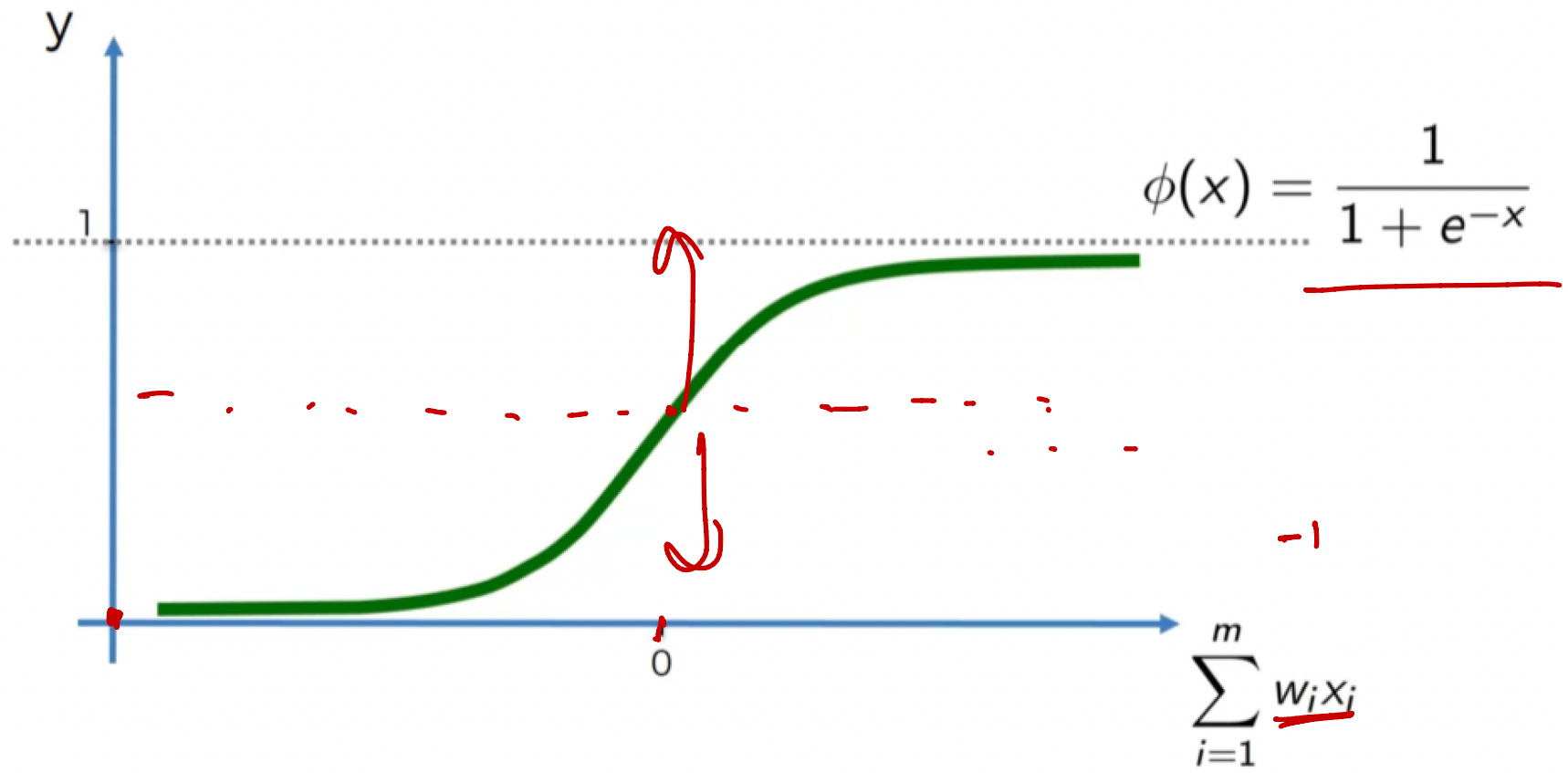
(classification)



$$\phi(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$

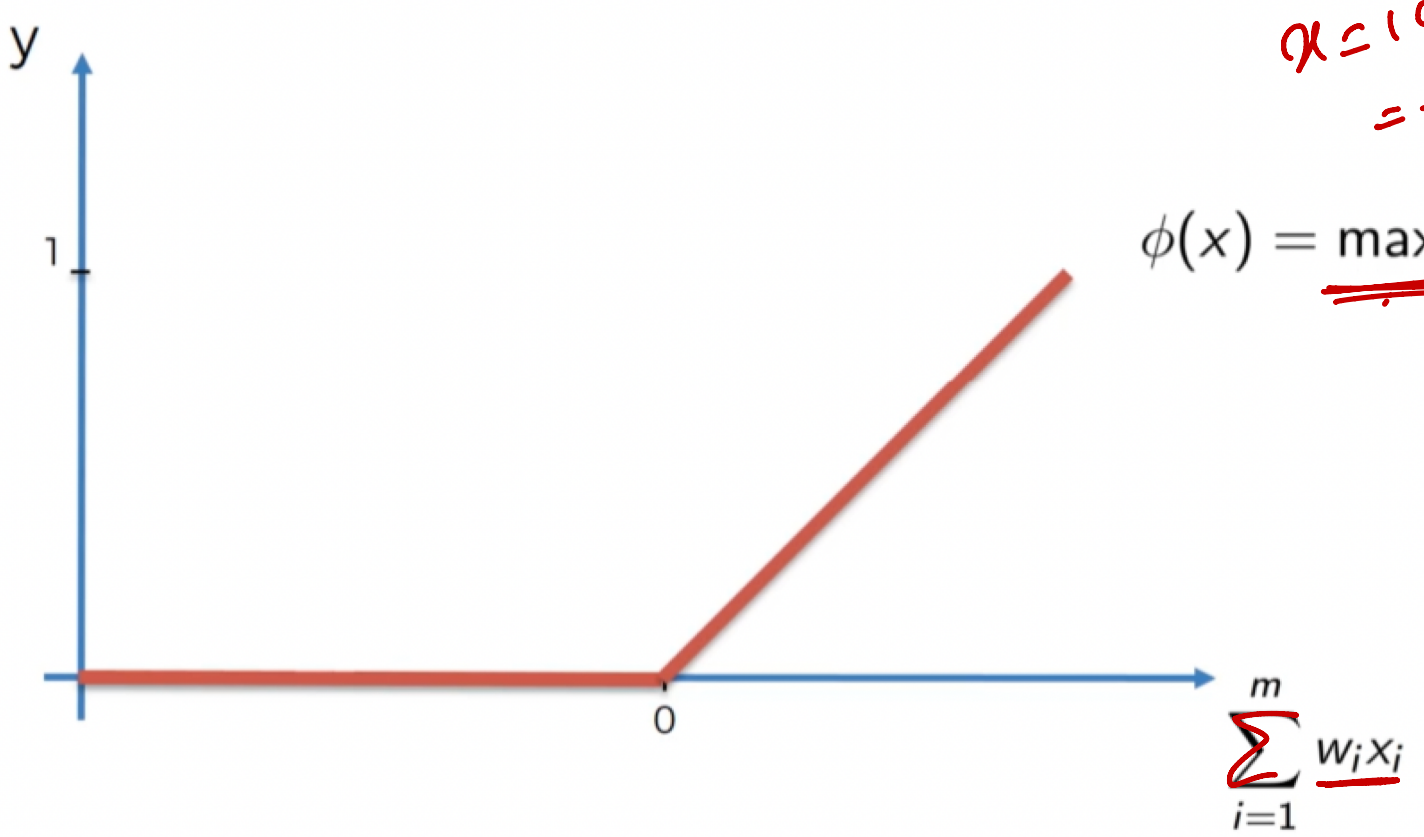
Sigmoid Function

(Regression)



Rectifier Function

ReLu

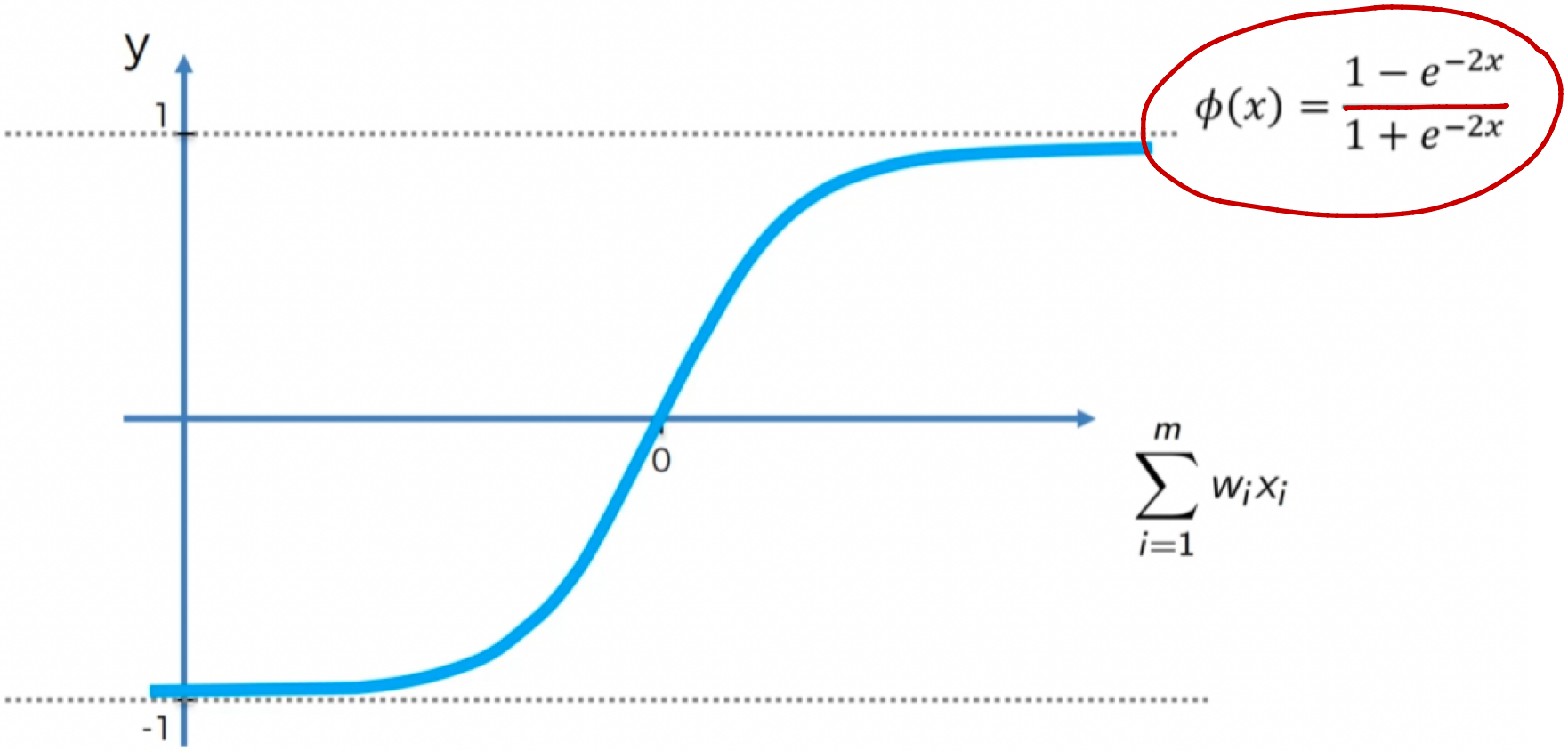


$\phi(x) = \max(x, 0)$

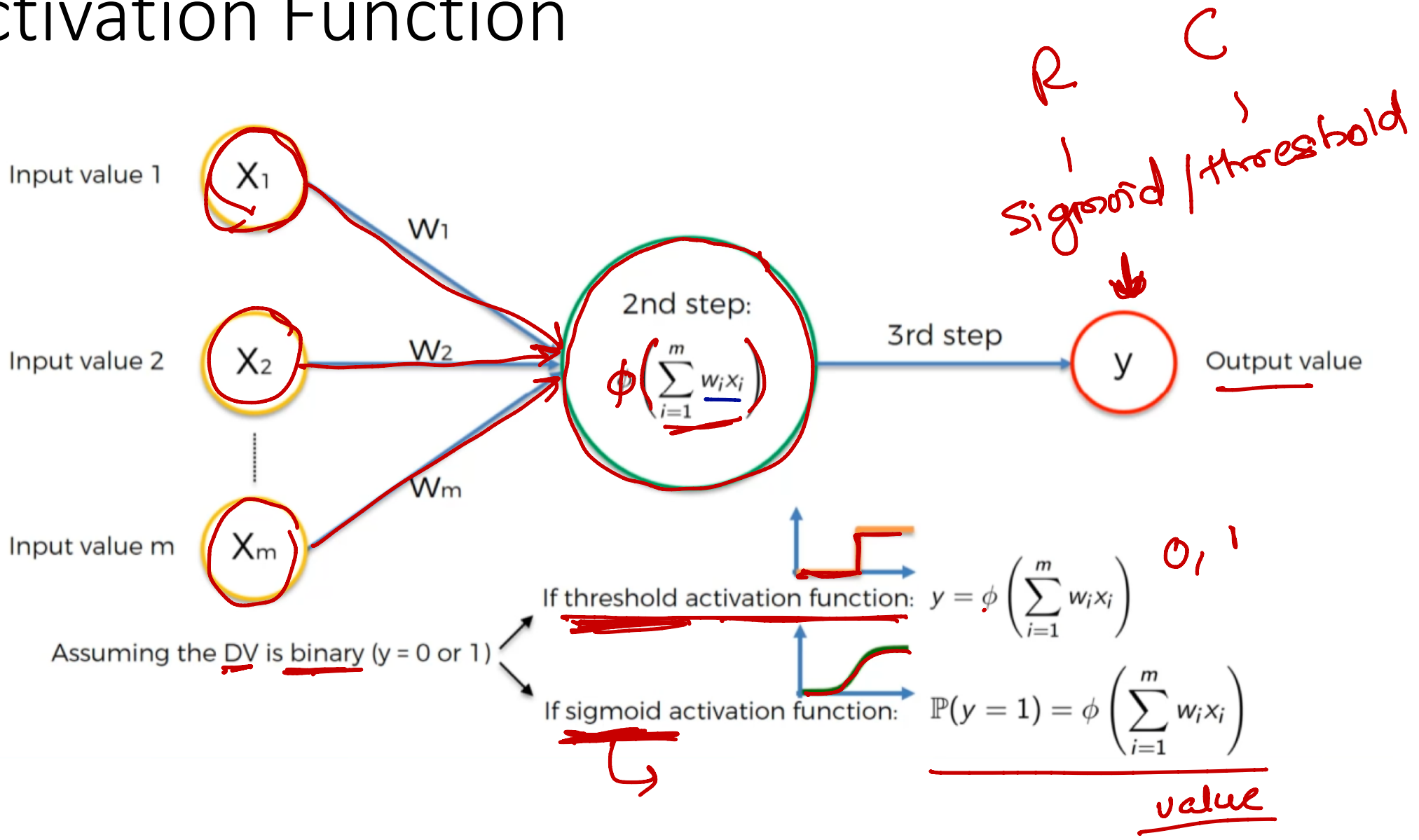
$\alpha = 10, \quad = 10$

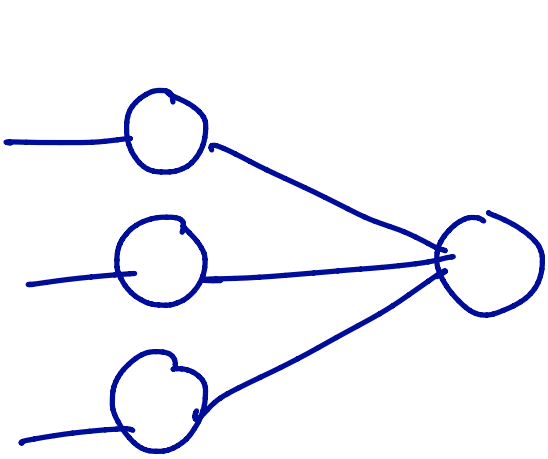
$= -1, \quad = 0$

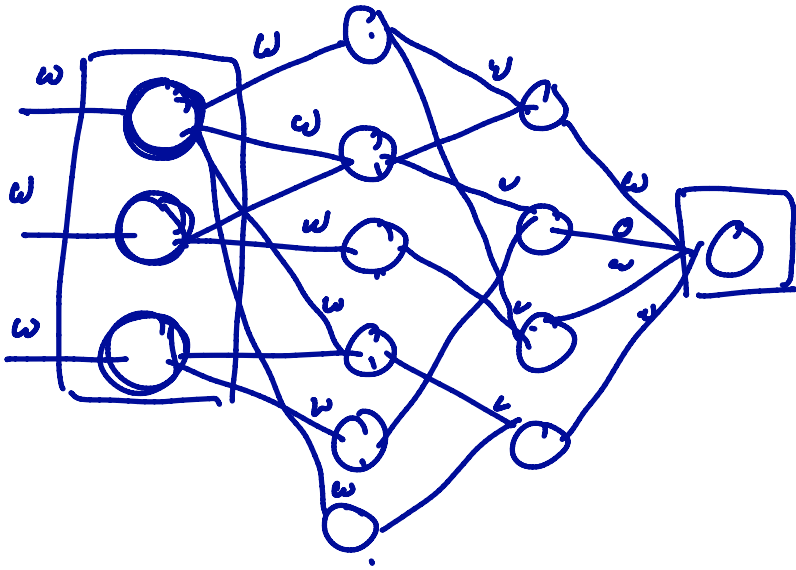
Hyperbolic Tangent Function

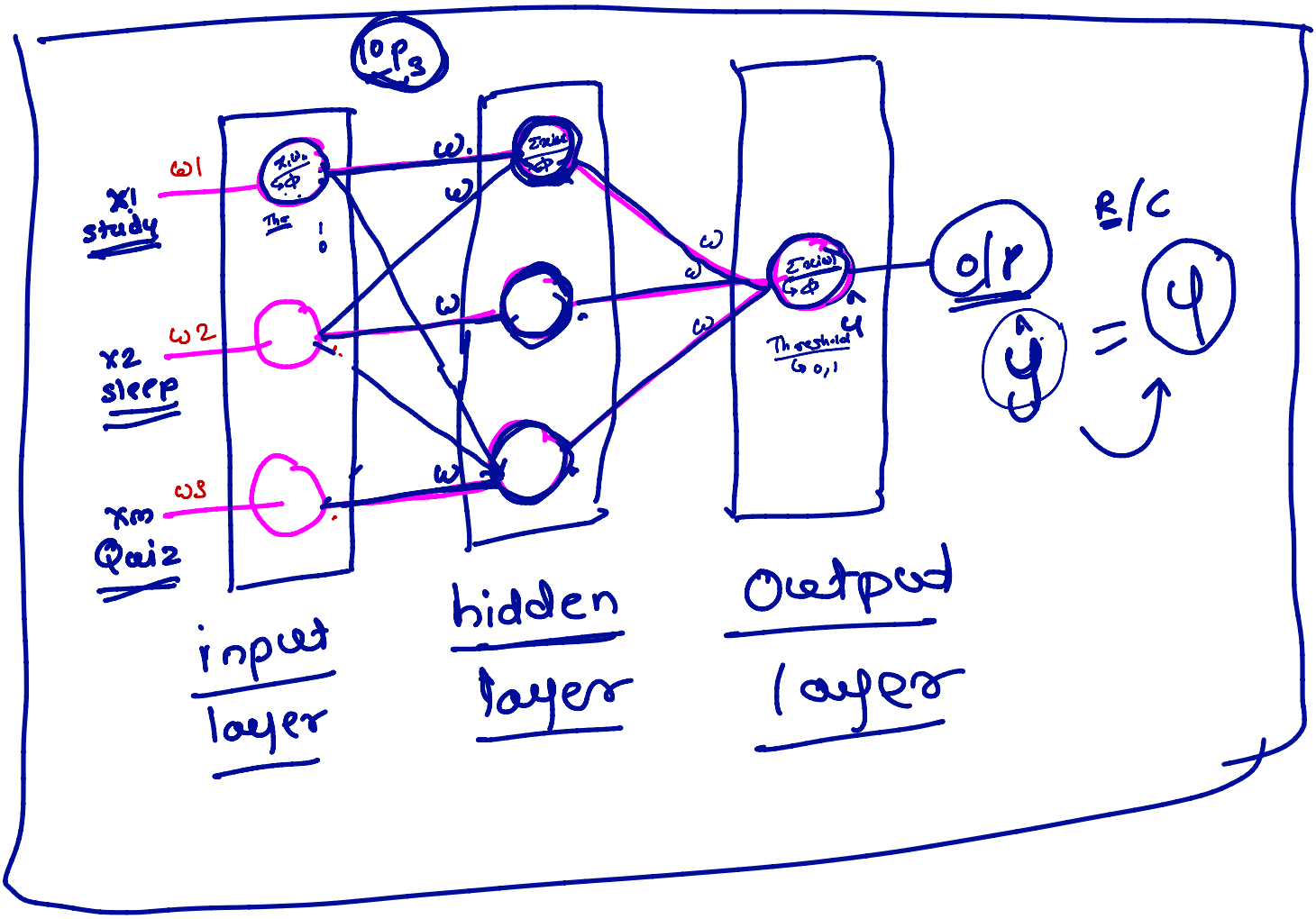


Activation Function

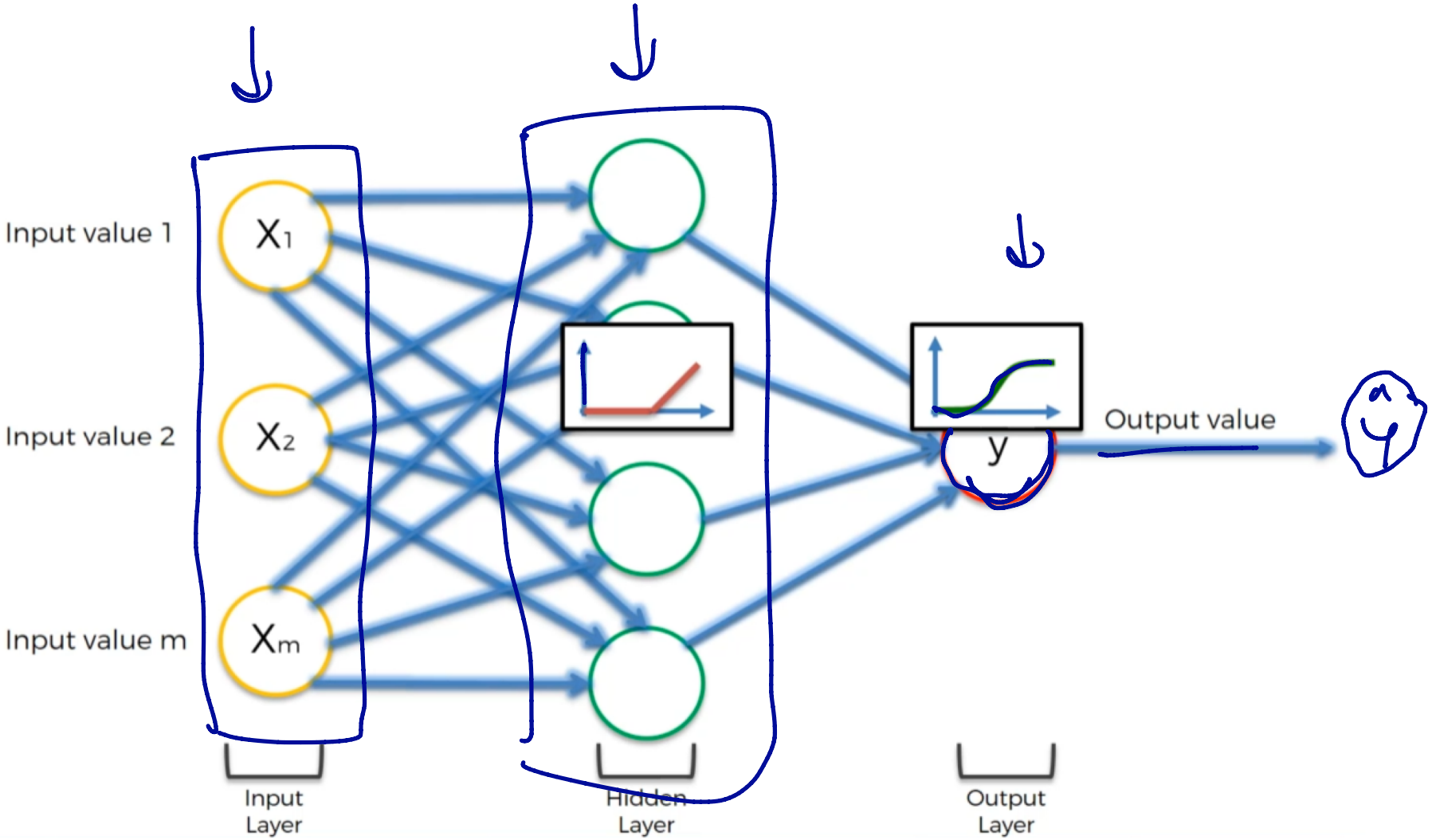


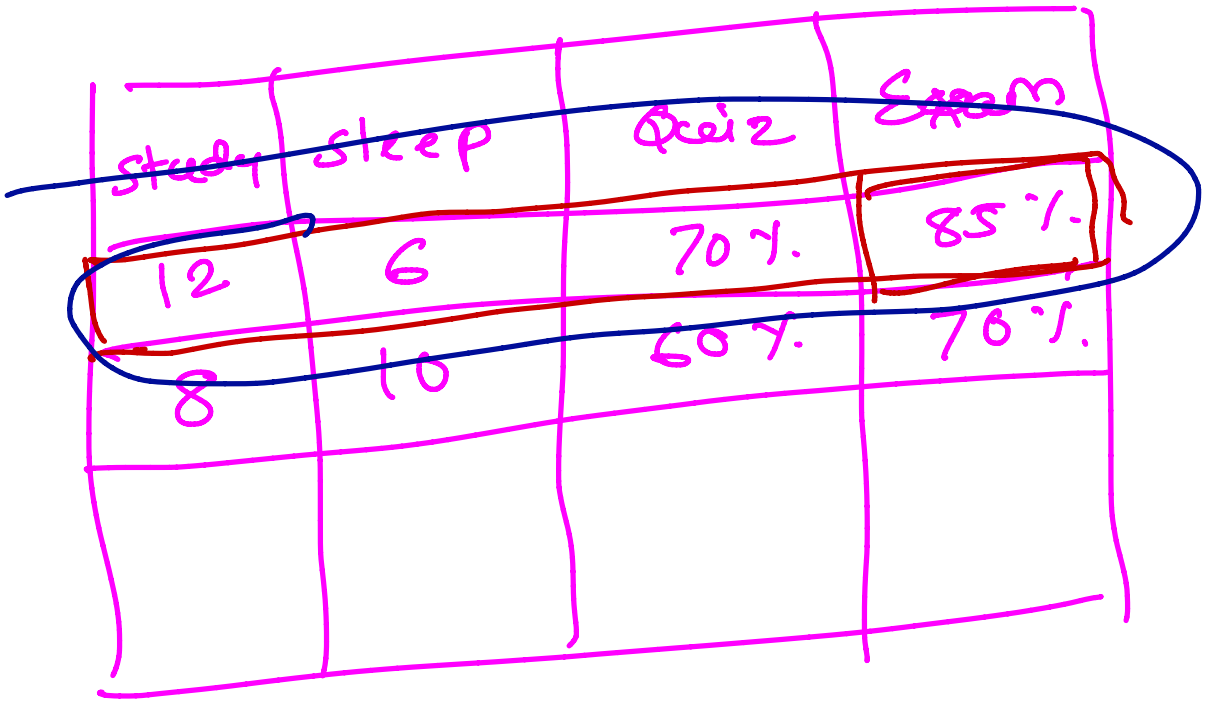





10 P3
x1
study
ω1
x2
sleep
ω2
x3
Quiz
ω3
ω
Threshold
0 or 1
o/p
ŷ
R/C
y
=
input
layer
hidden
layer
Output
layer

Activation Function



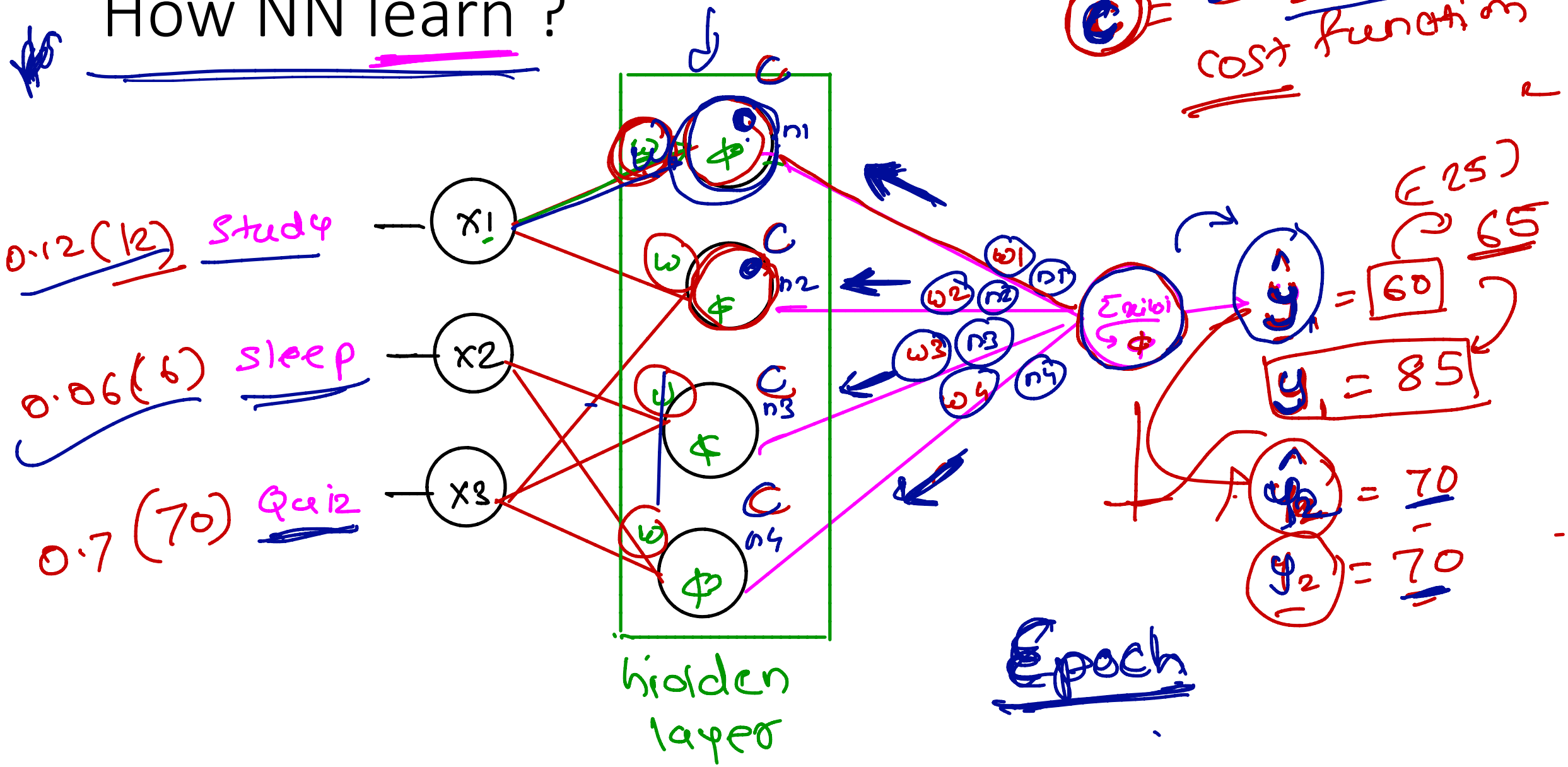

Study
sleep
Quiz
Exam
12
6
70 %.
85 %.
8
10
60 %.
76 %.

How NN learn ?

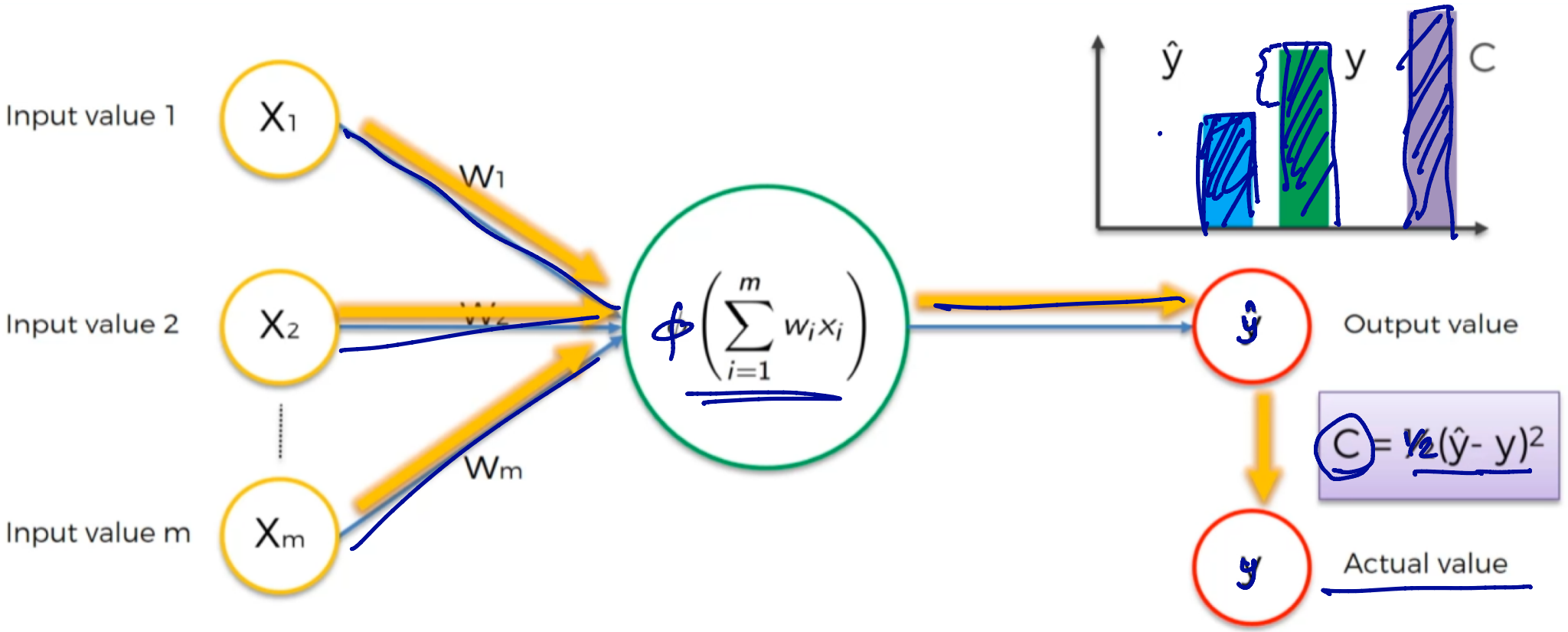
Back propagation

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

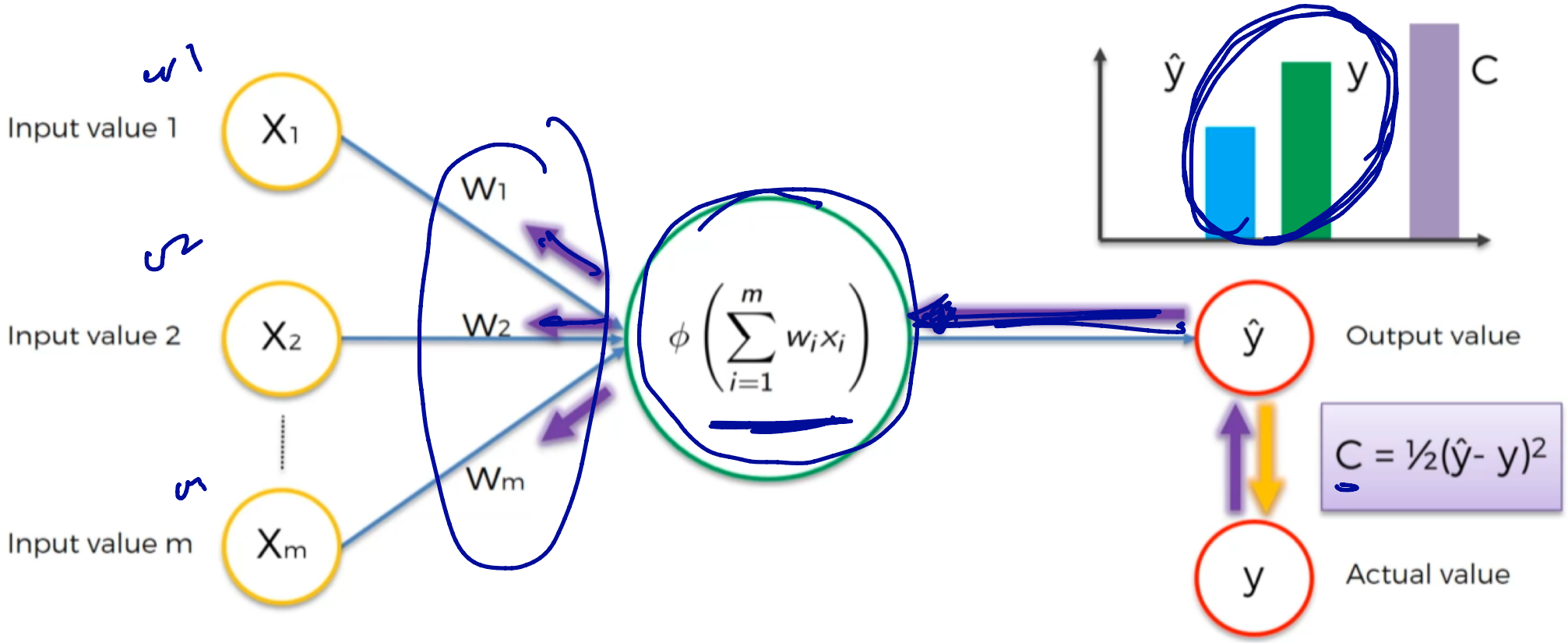
cost function



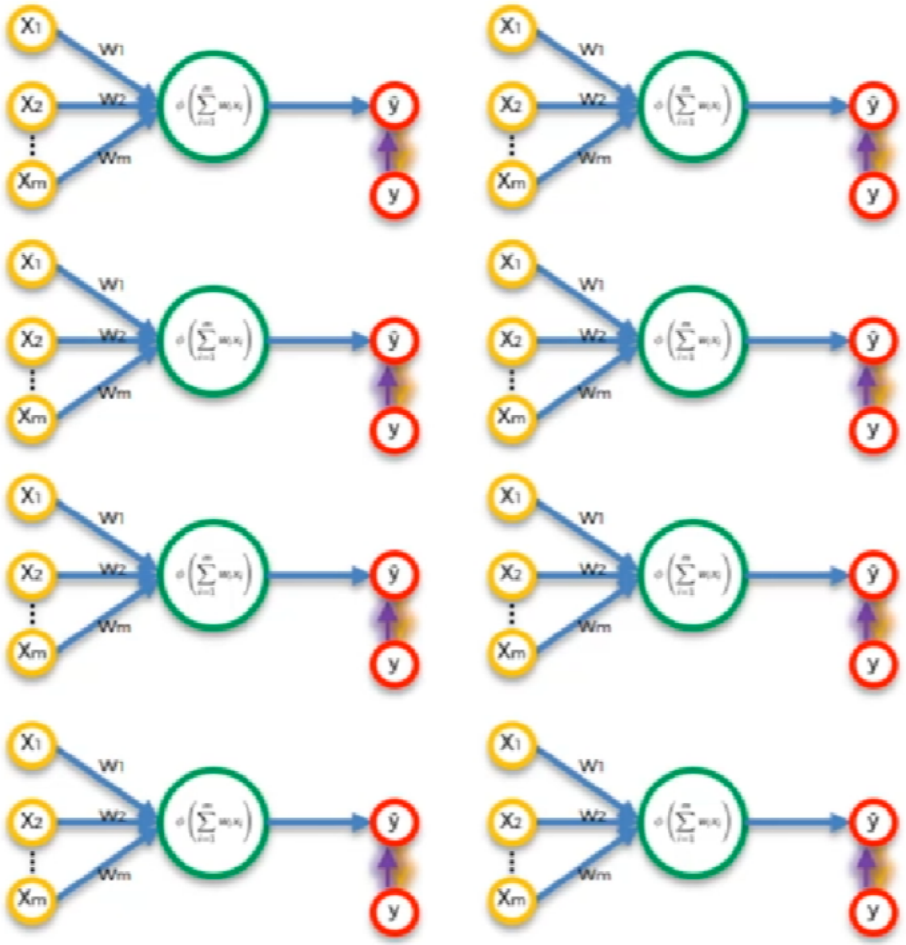
Neural Network - learn



Neural Network - learn



Neural Network - learn

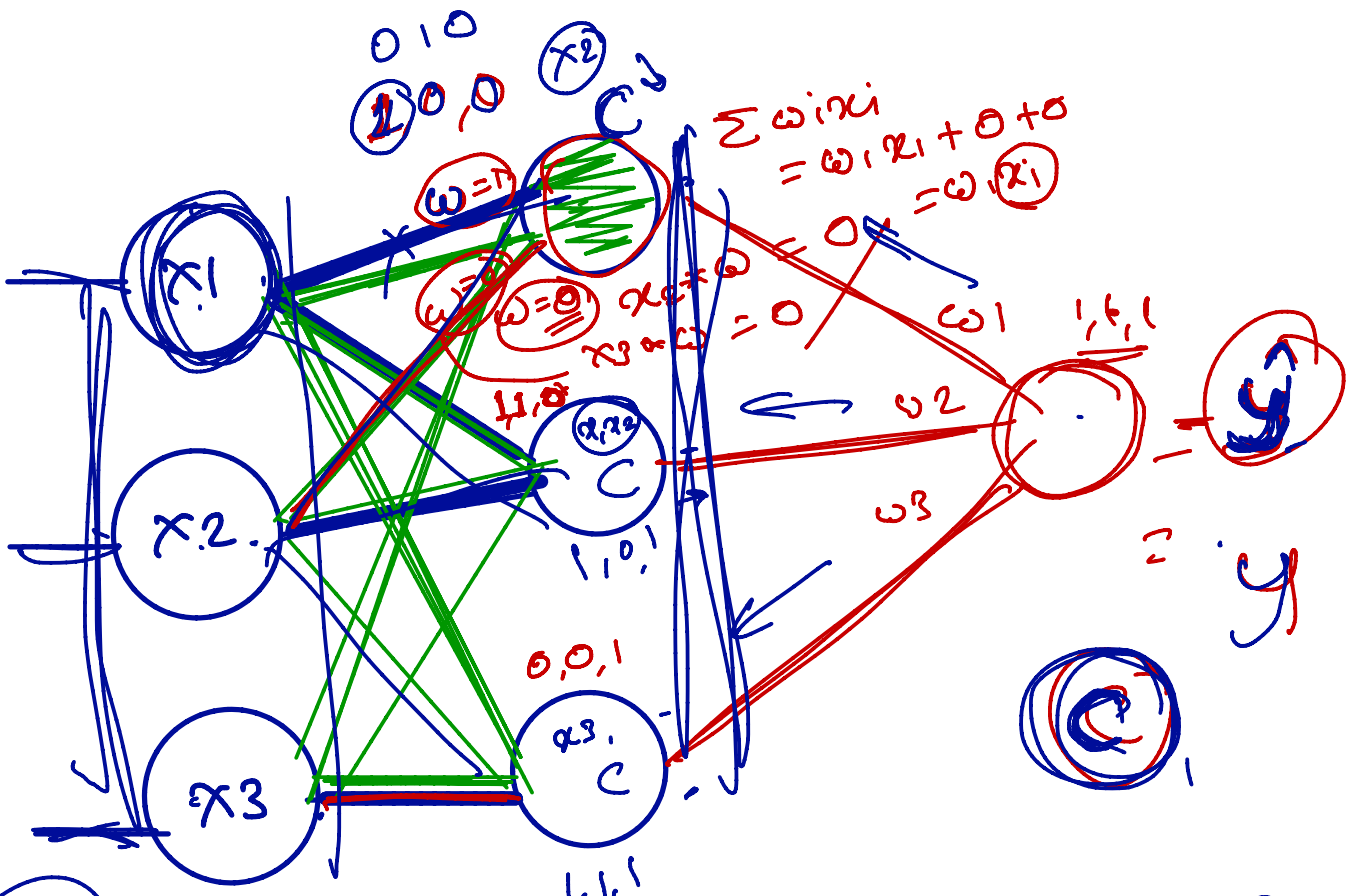


Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

epochs

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

Adjust w_1, w_2, w_3



$\mathcal{P}_3$

$\mathcal{P}_3 = 100$ pow

e1 - c1

e2 = c2

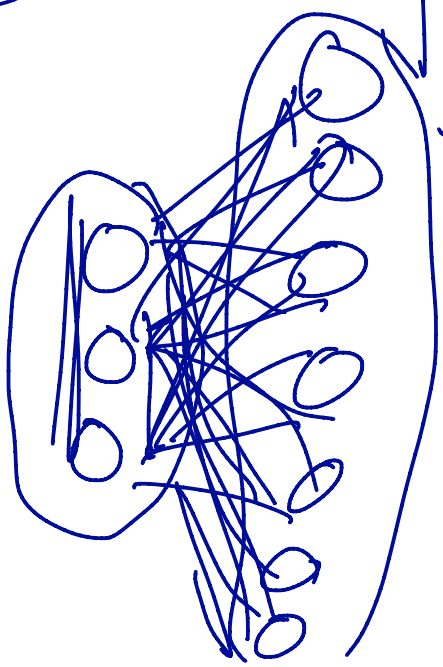
e3 = c3

100

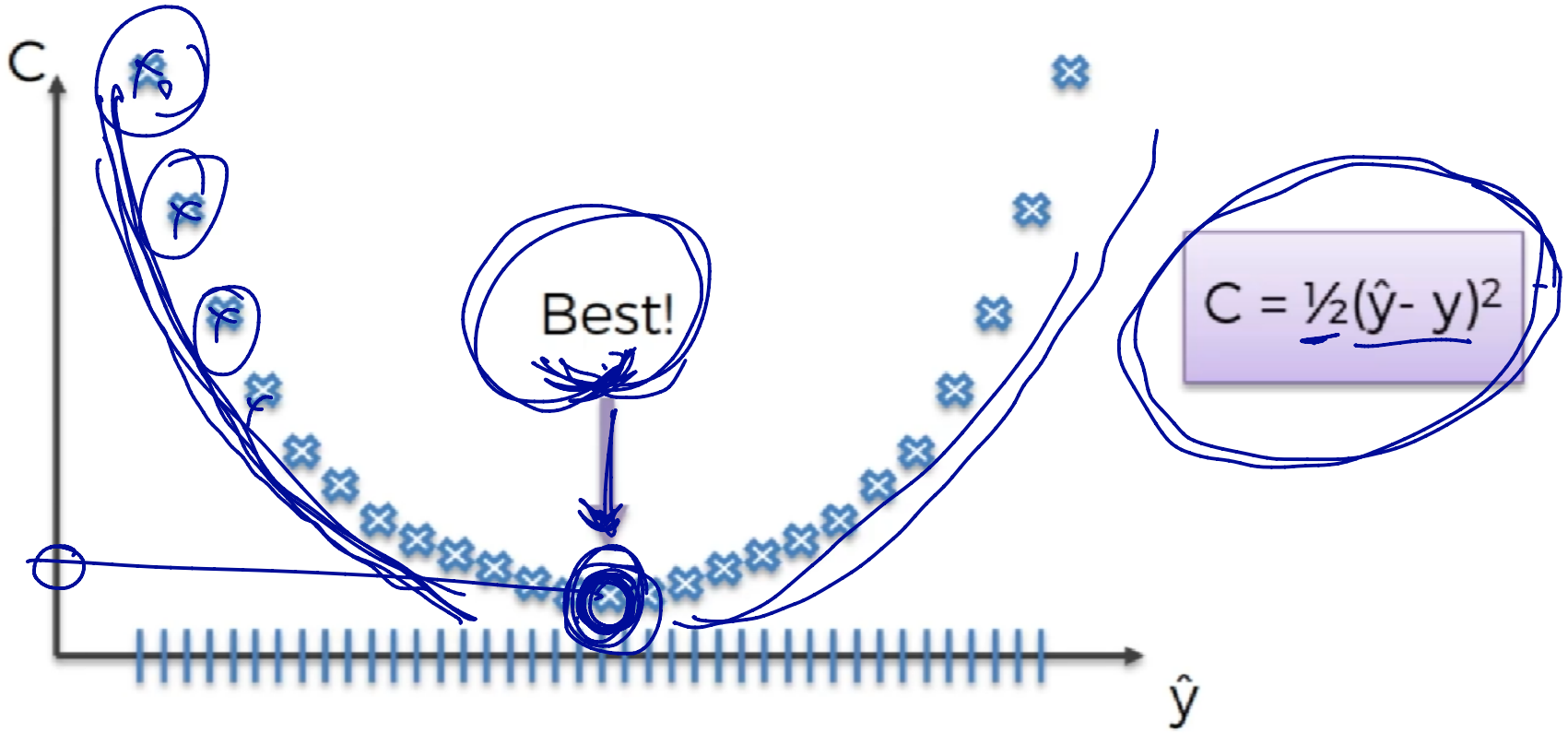
100

100

100 pow



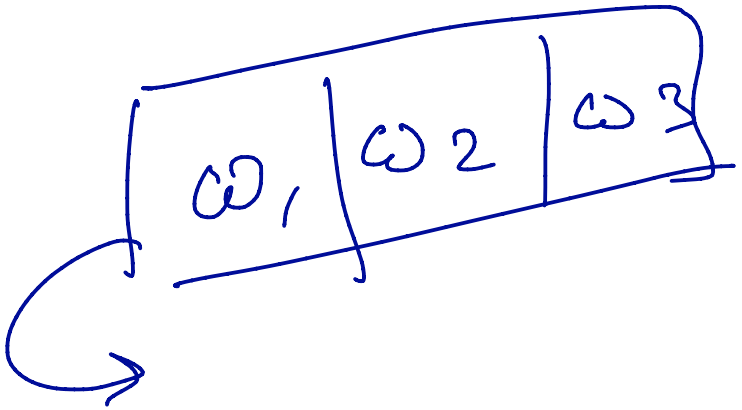
Gradient Descent

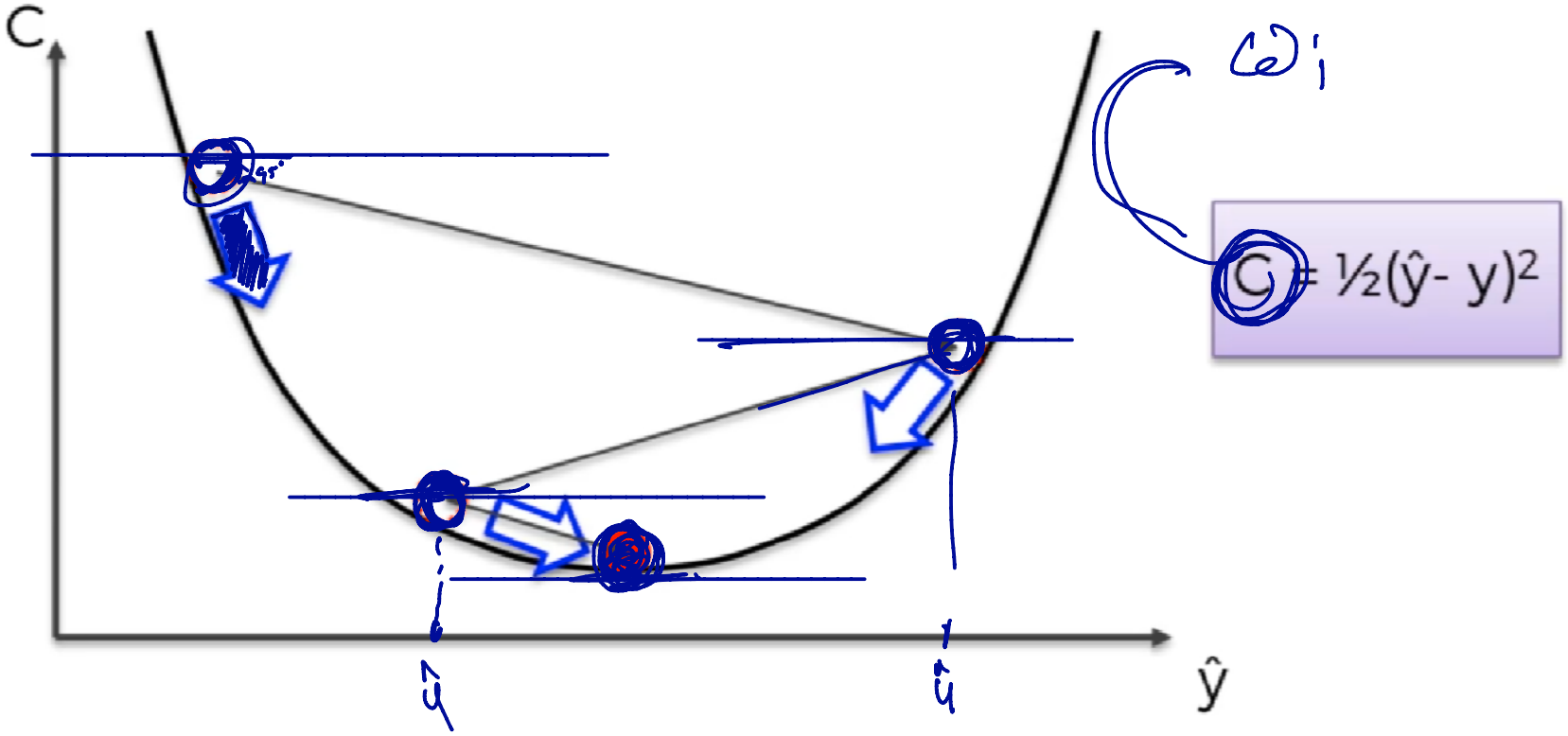


Curse of Dimensionality

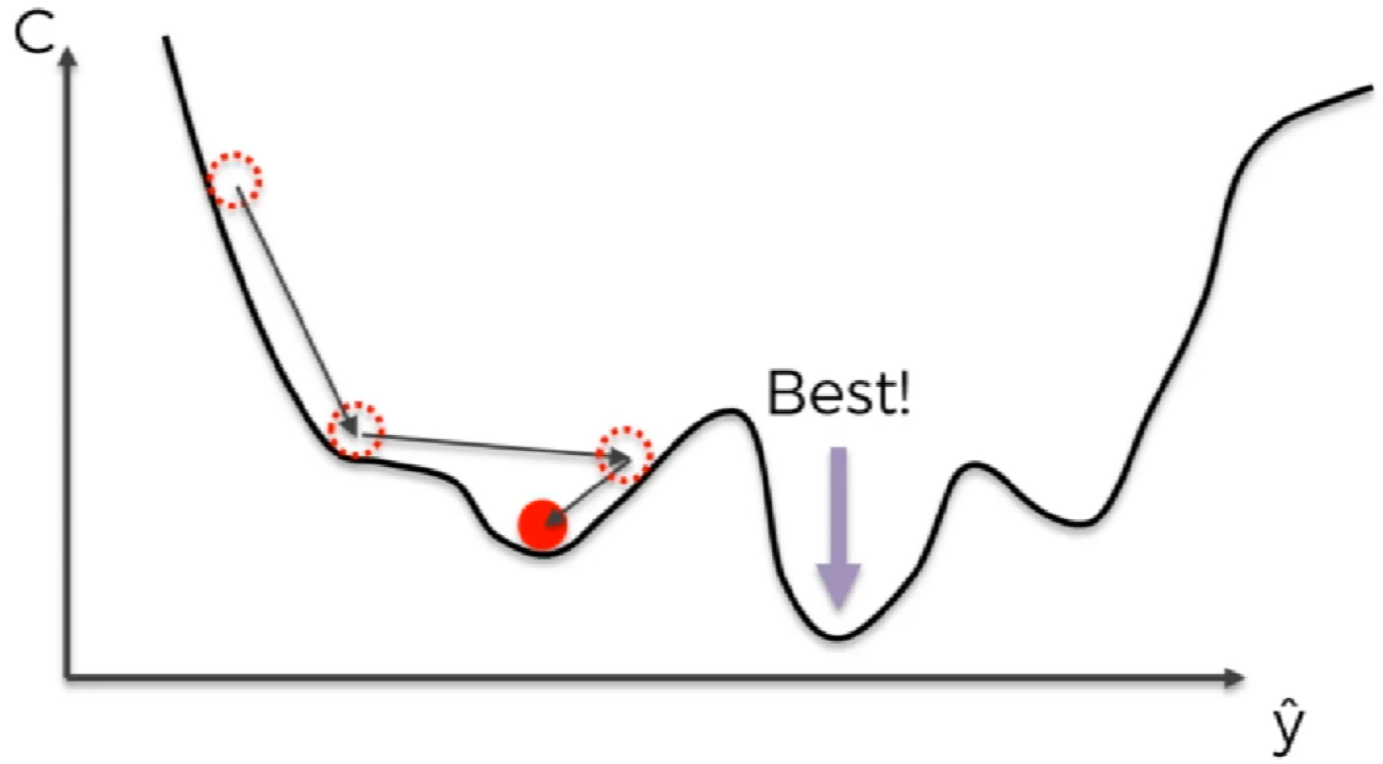
- As the dimensionality of features increases, the number of configurations grow exponentially and thus the number of configurations covered by observations decreases

Gradient Descent

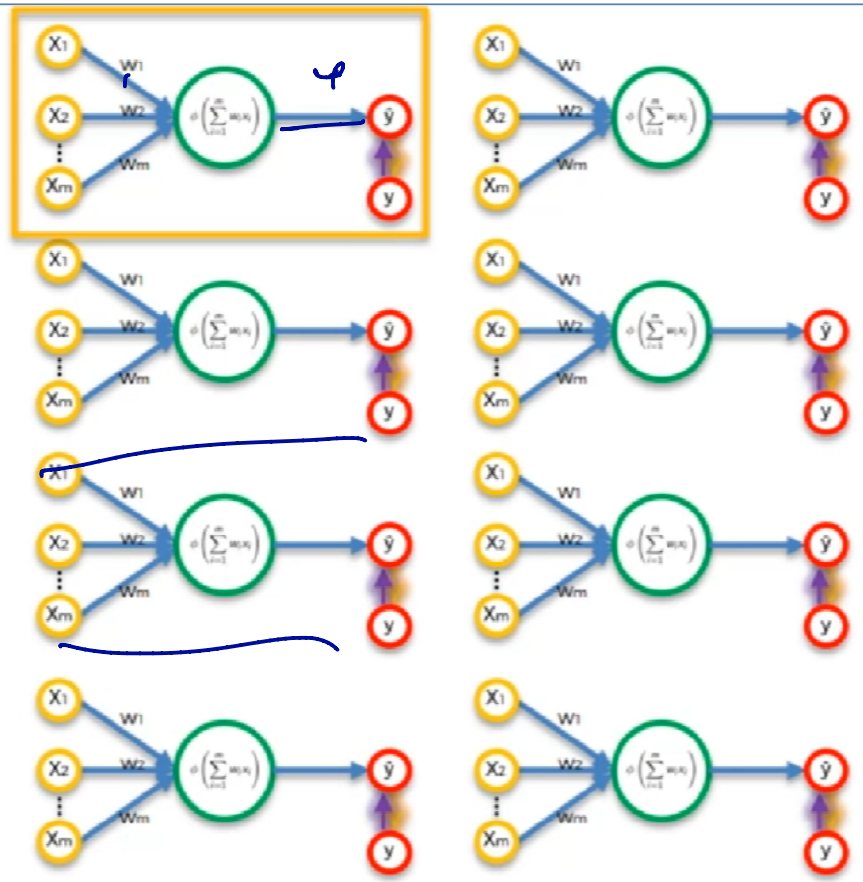




Stochastic Gradient Descent



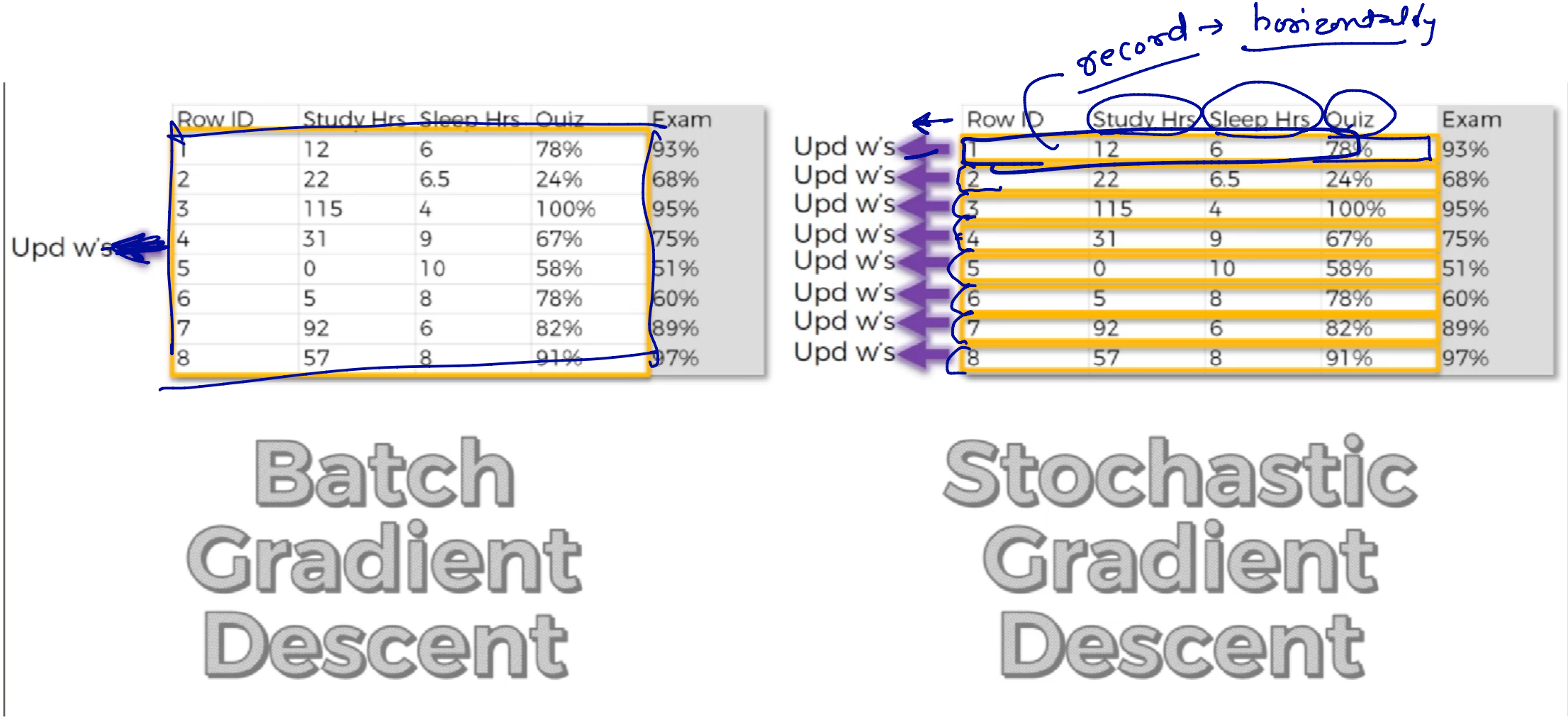
Stochastic Gradient Descent



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

$(\hat{y} - y)$

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

Stochastic Gradient Descent - Steps

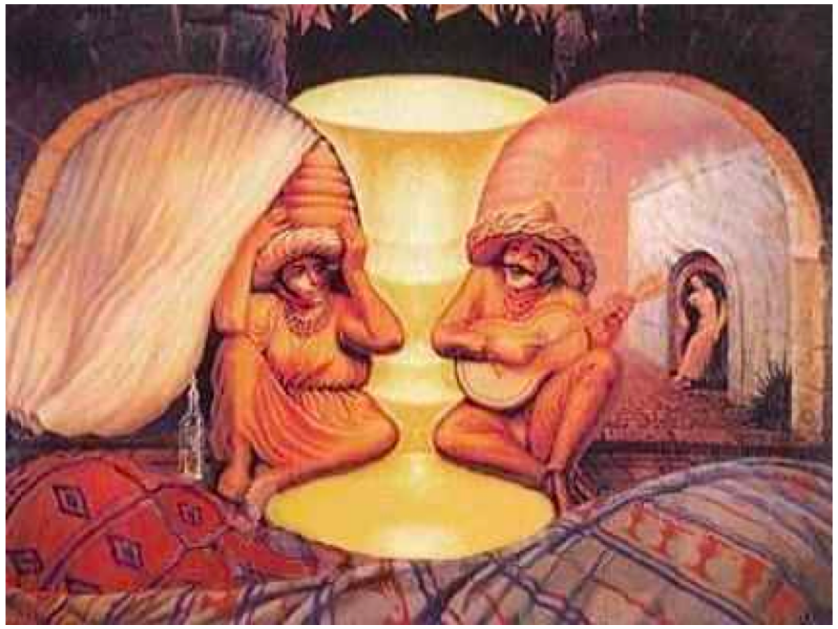
- Back Propagation
  - From right to left
  - The error is back propagated
  - Update the weights according to how much they are responsible for the error
  - The learning rate decides by how much we update the weights
- Repeat the steps and update the weights
  - after each observation (stochastic)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

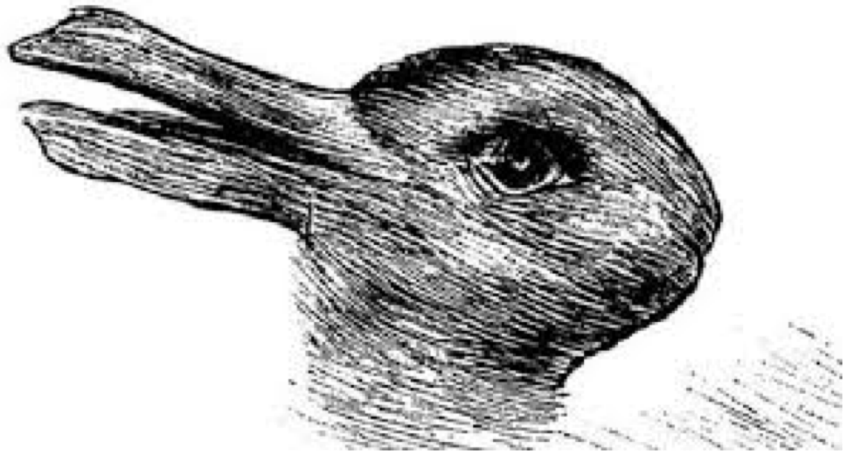
CNN

Convolutional Neural Network

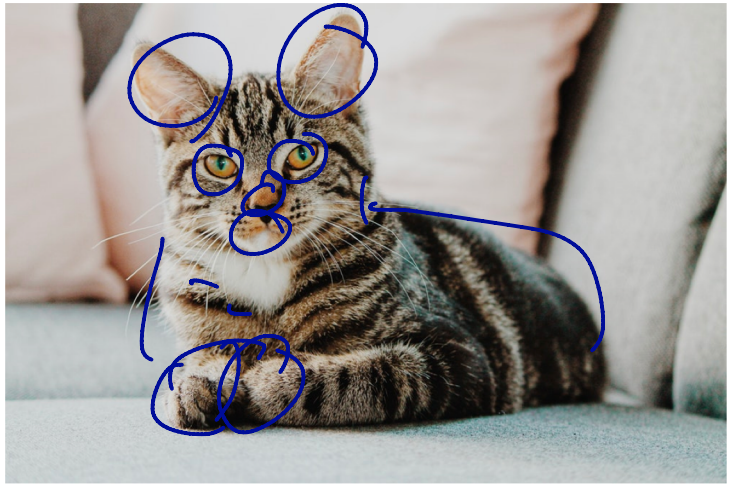
CNN

illusion

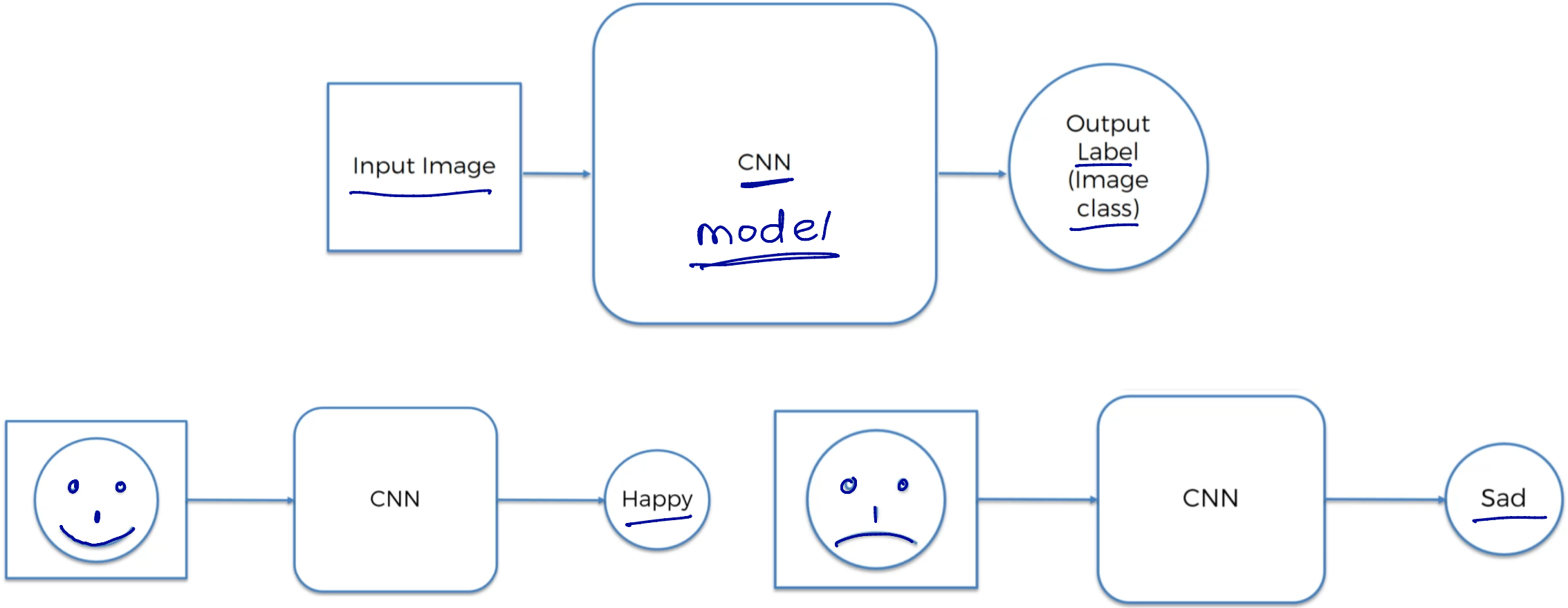




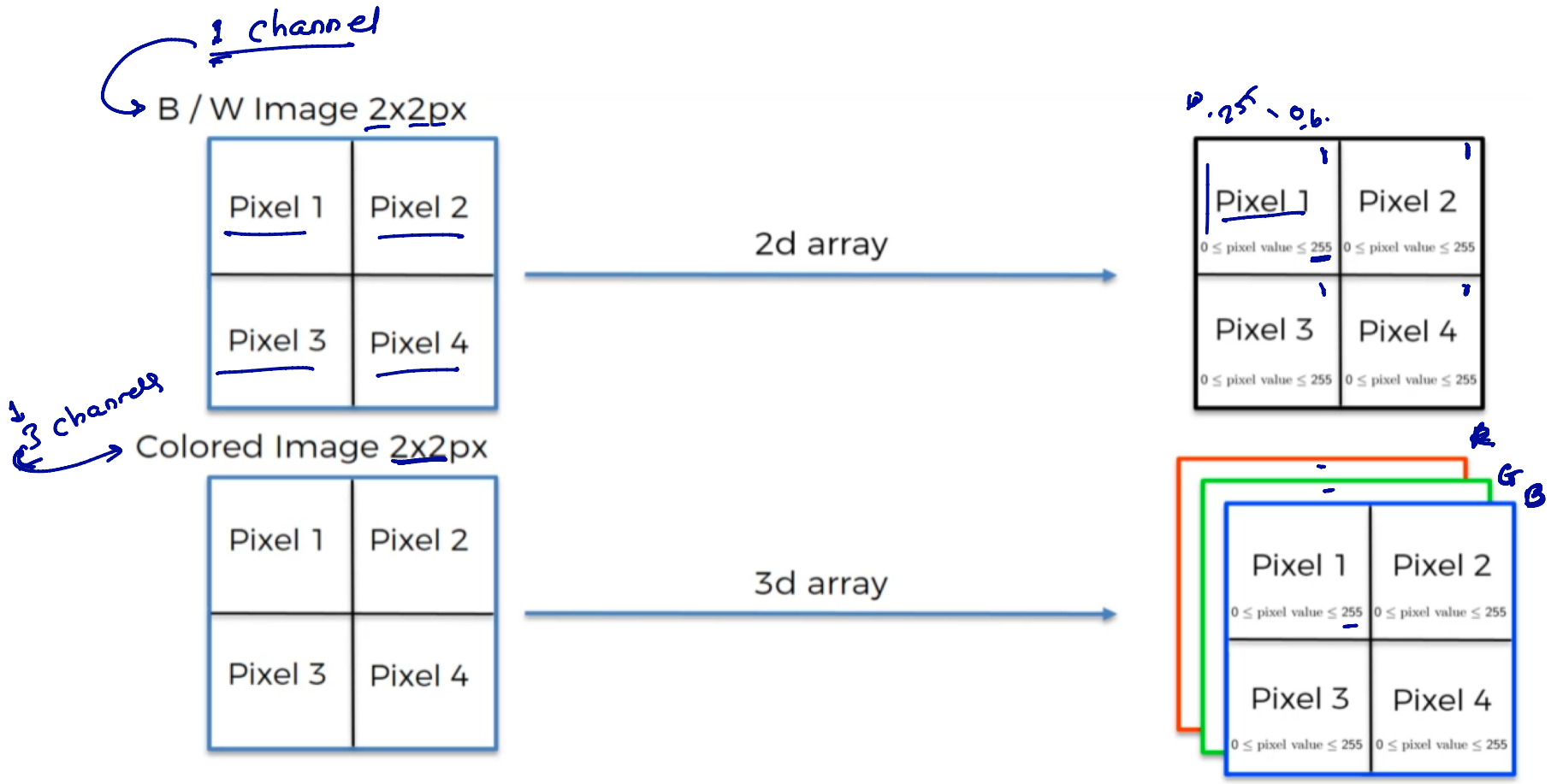
CNN

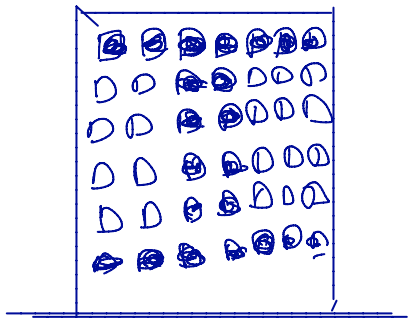


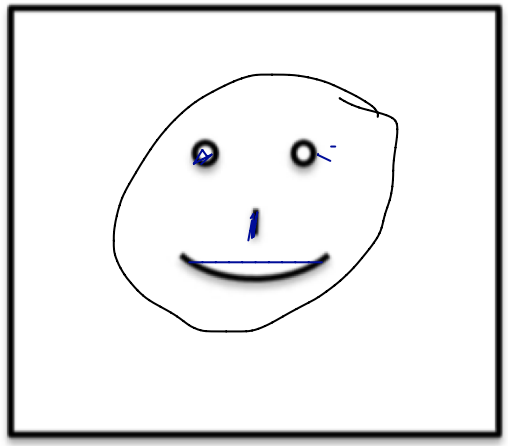
CNN

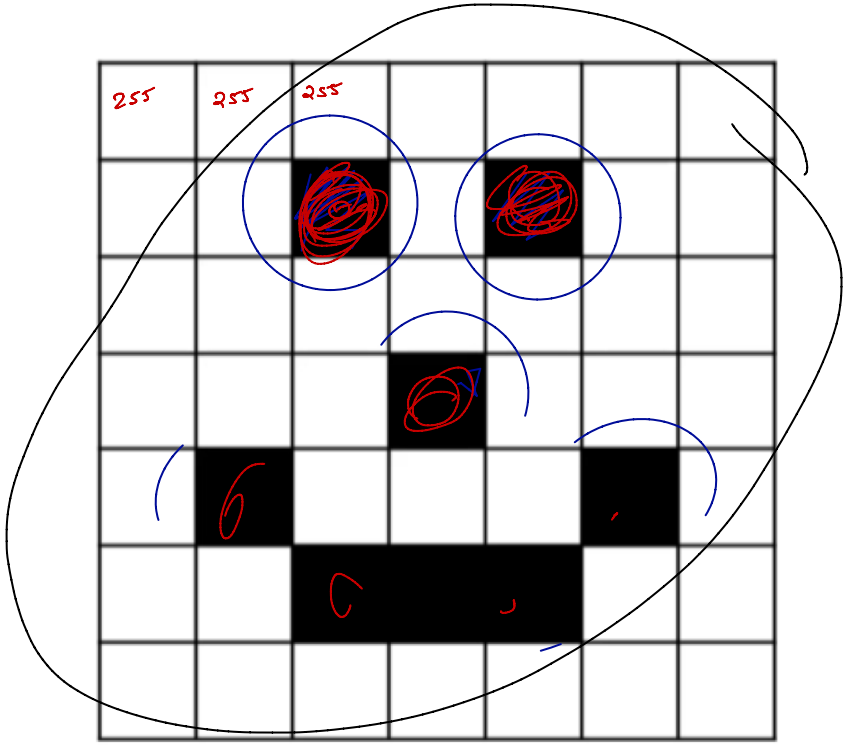

Input Image
CNN
model
Output
Label
(Image
class)
CNN
Happy
CNN
Sad

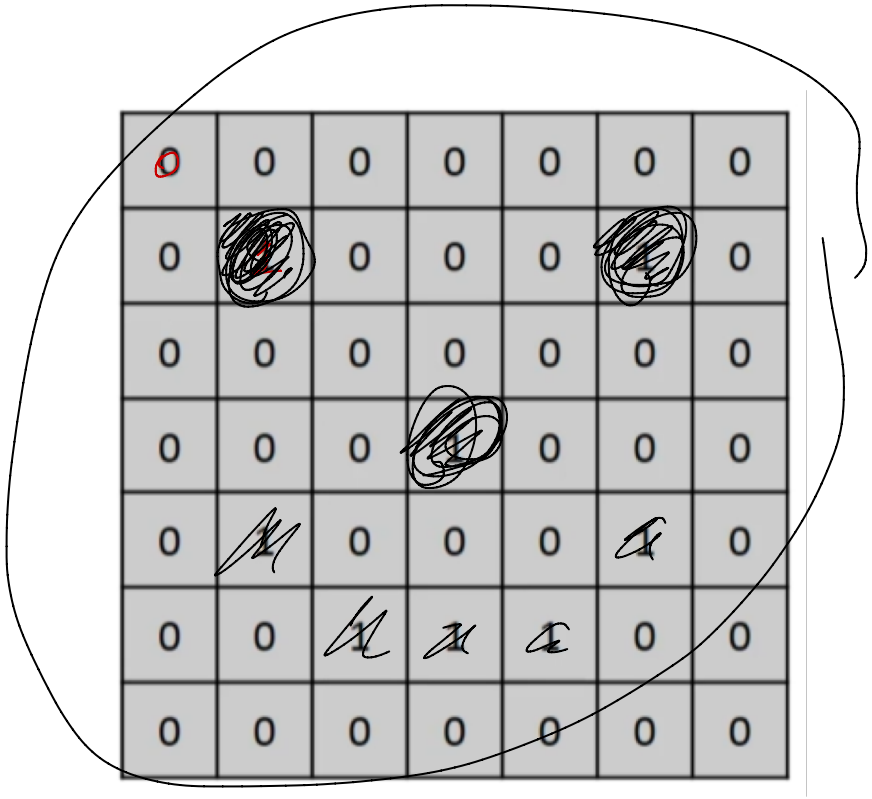
CNN


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






255
255
255


0 0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0

CNN - Steps

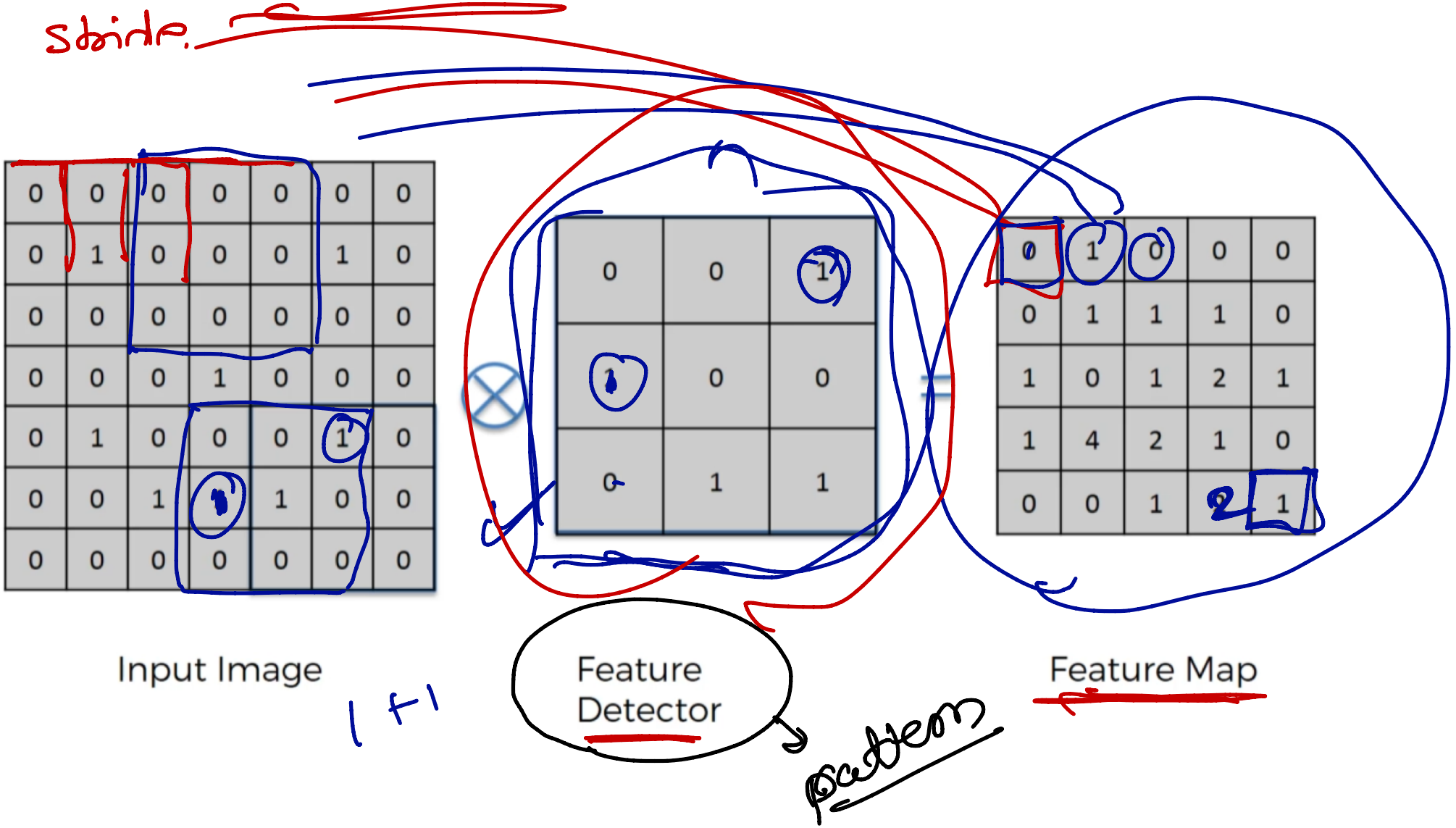
- Step 1: Convolutions
- Step 2: Max Pooling
- Step 3: Fattening
- Step 4: Full Connection

Step 1 – Convolution

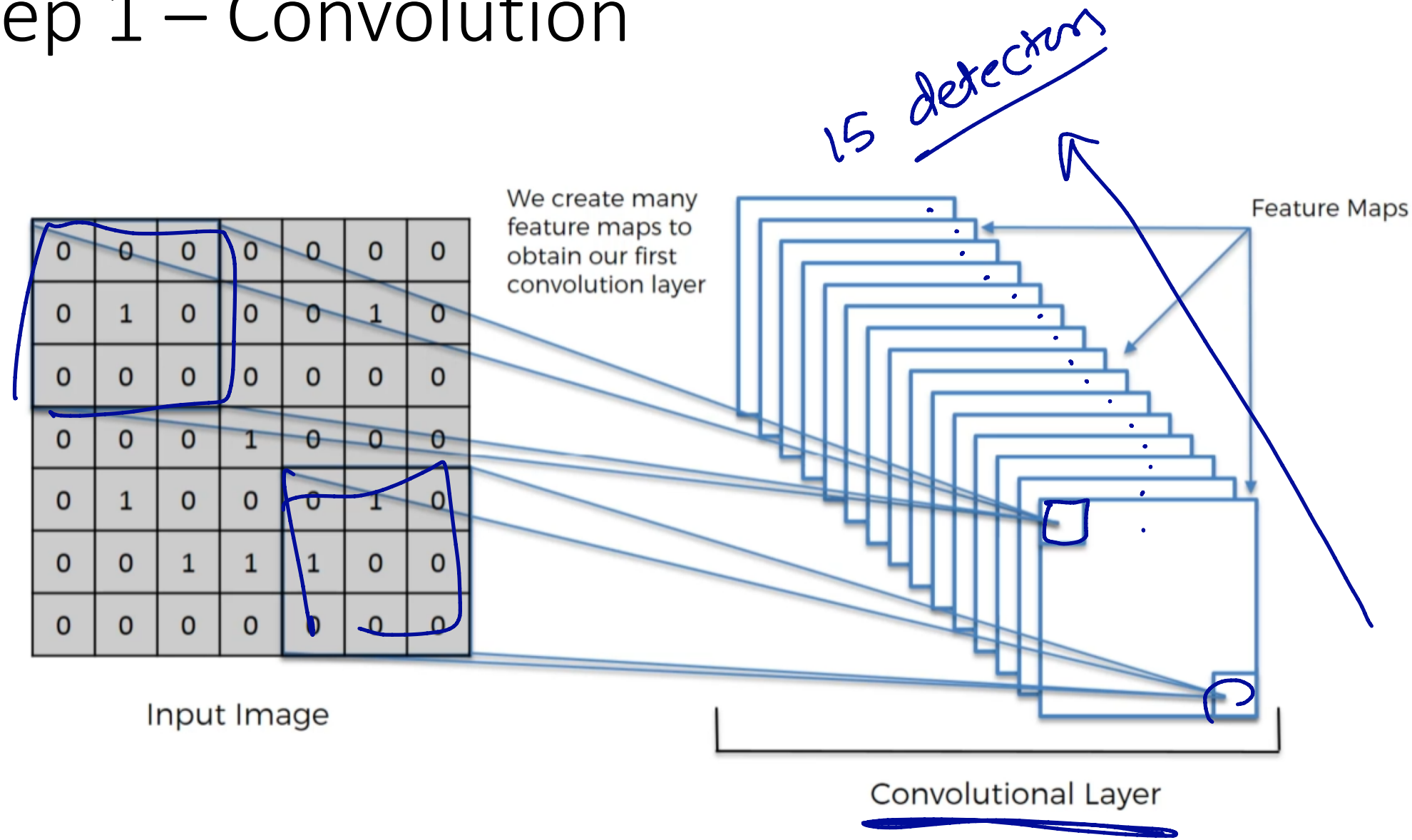
$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

- 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

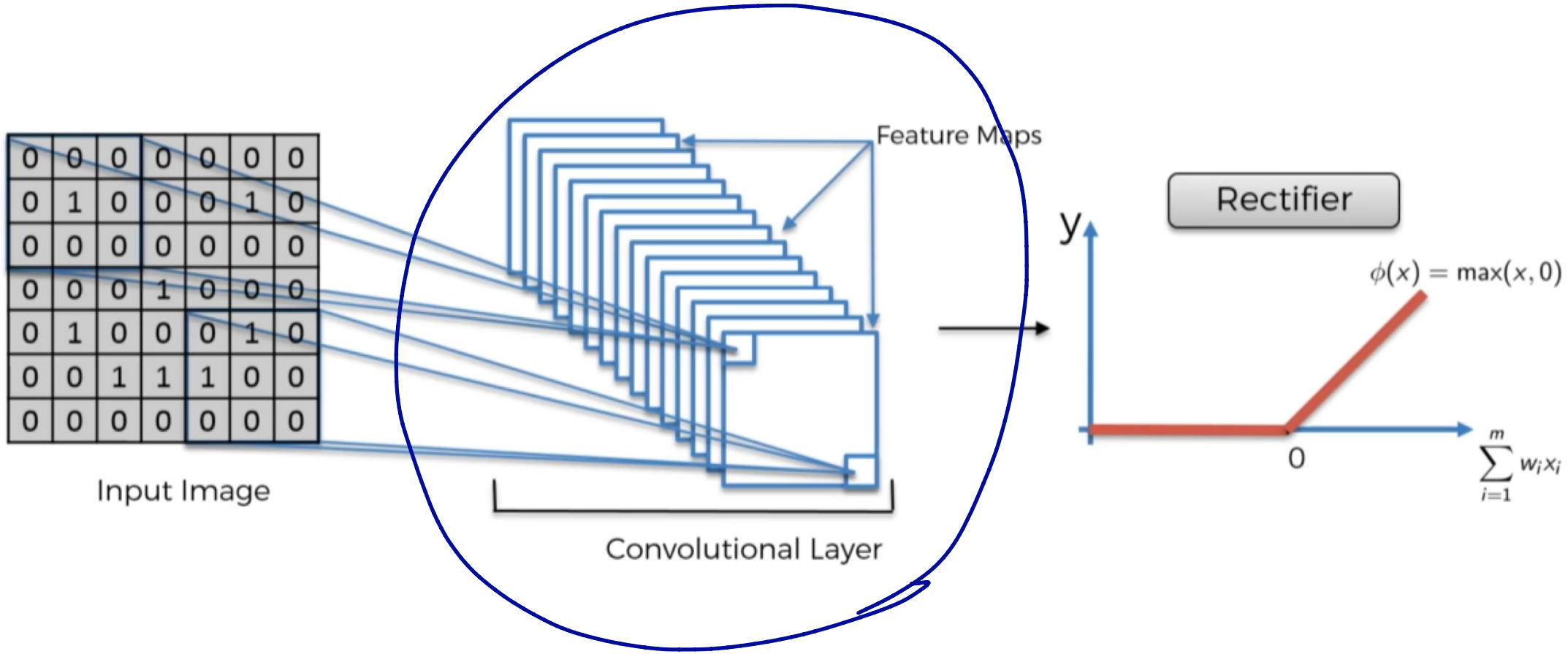
Step 1 – Convolution



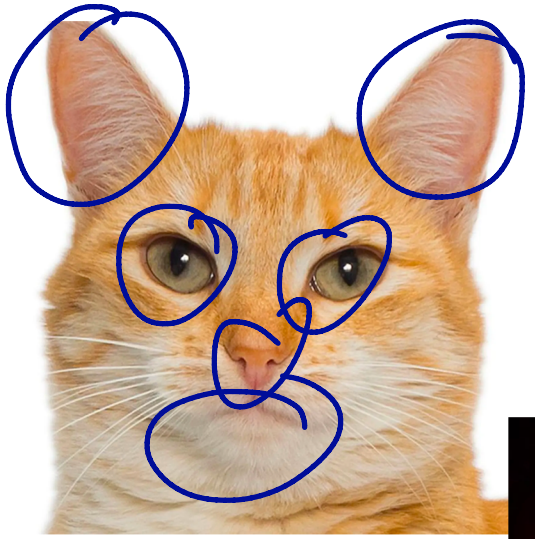
Step 1 – Convolution

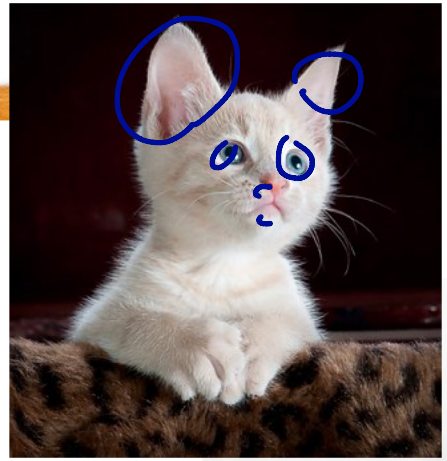


Step 1 (B) – ReLu Layer

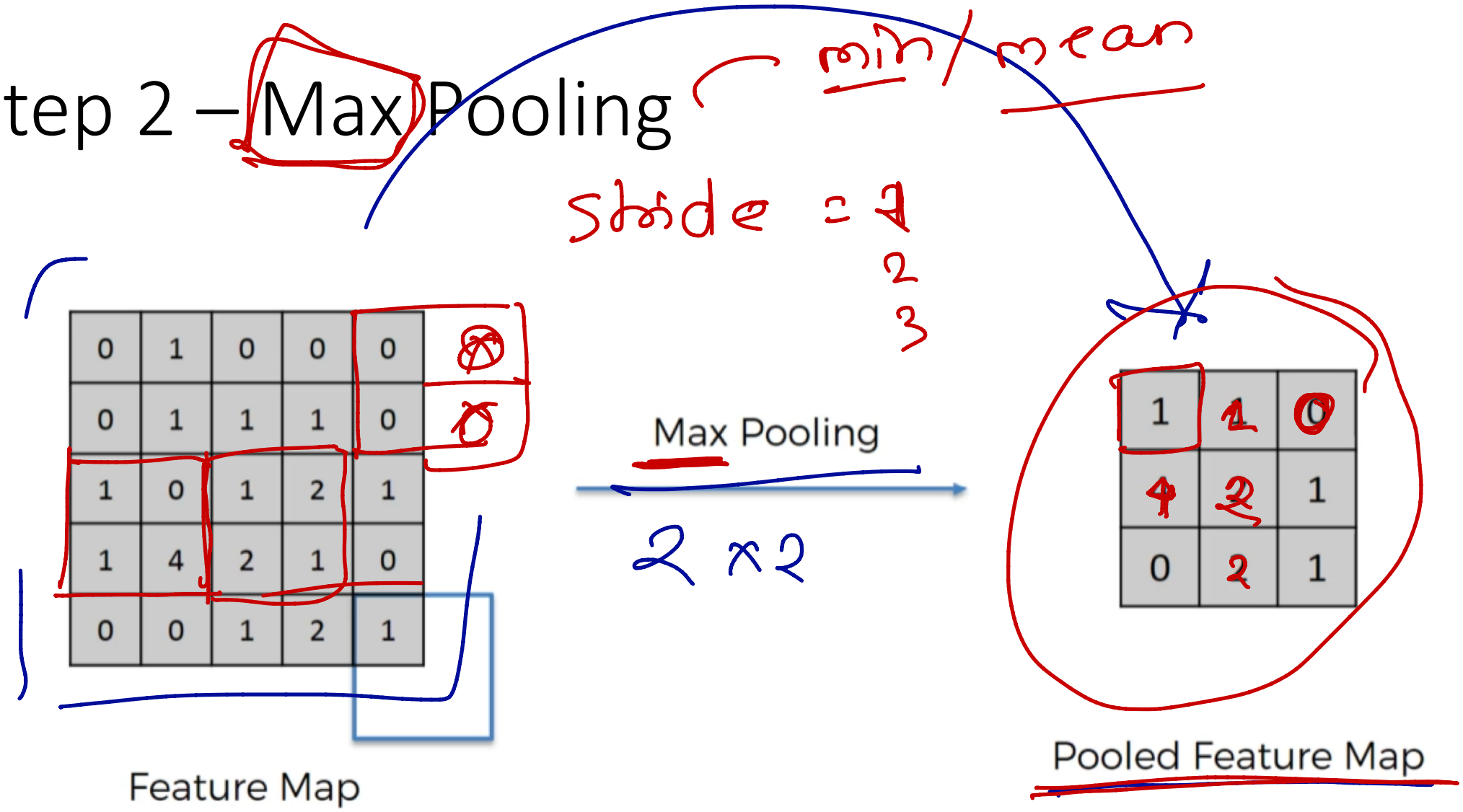


Step 2 – Max Pooling

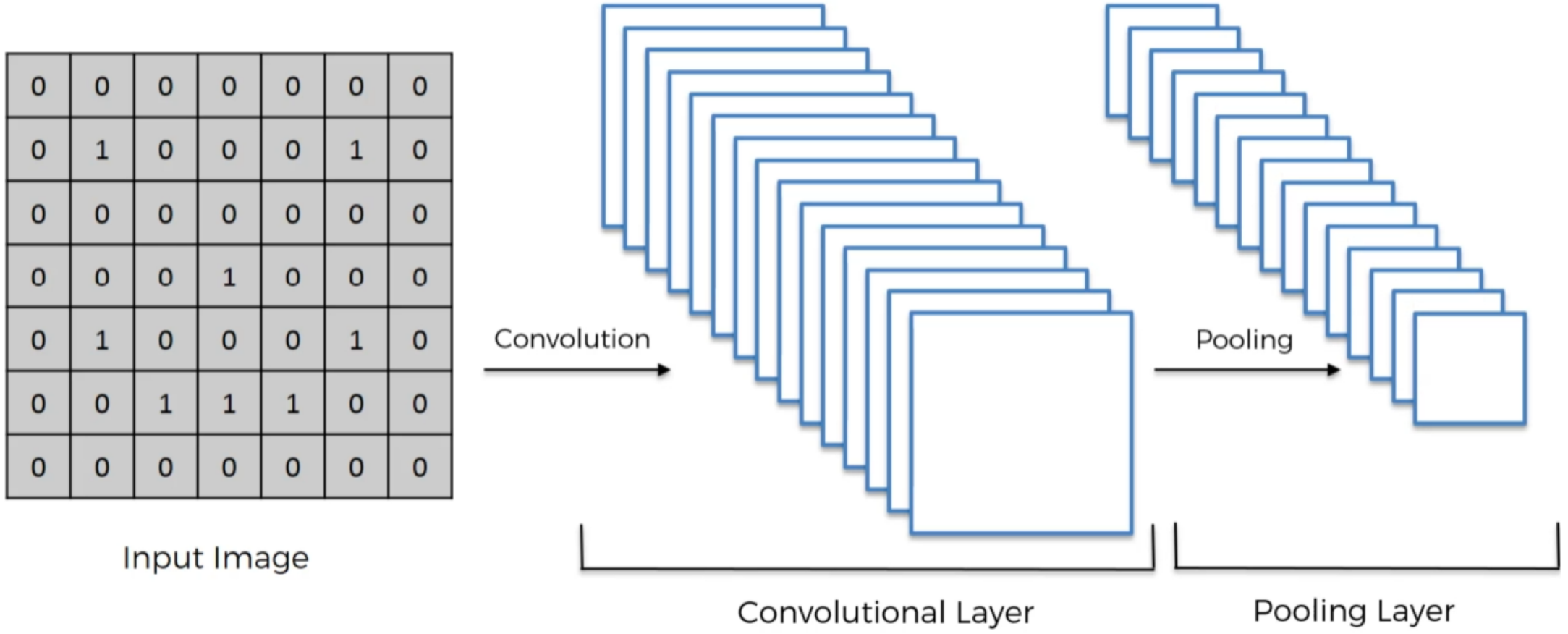




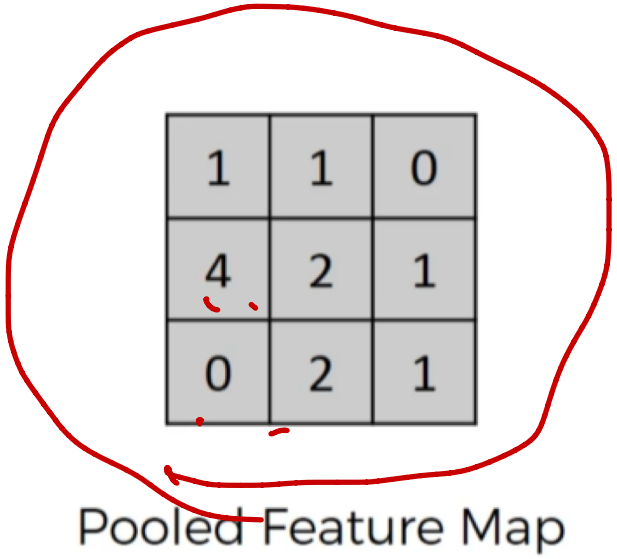
Step 2 – Max Pooling

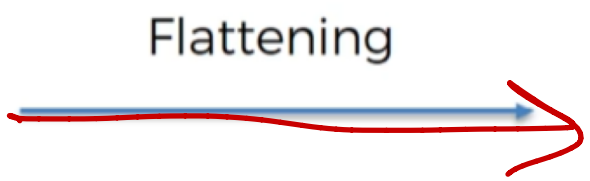


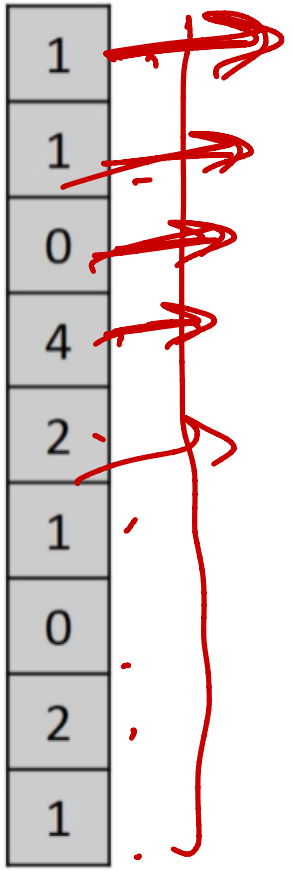
Step 2 – Max Pooling



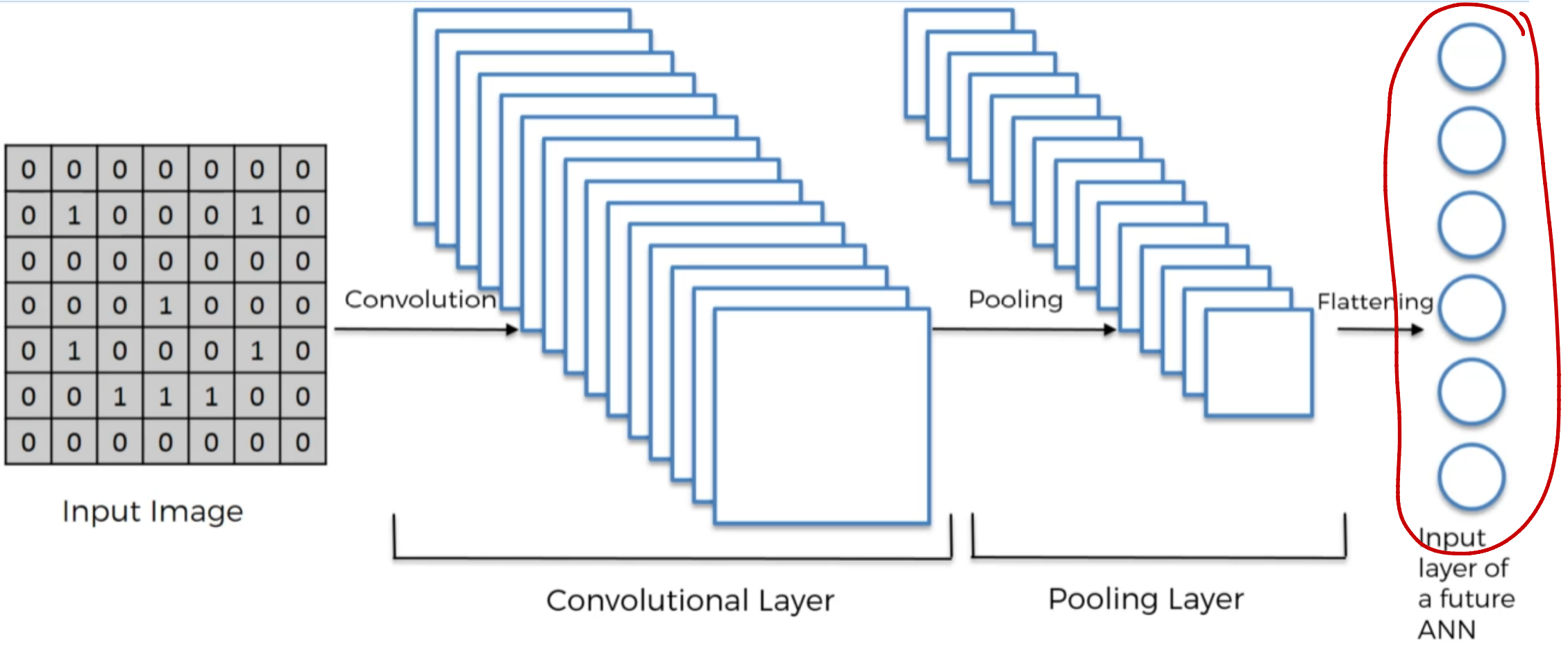
Step 3 – Flattening



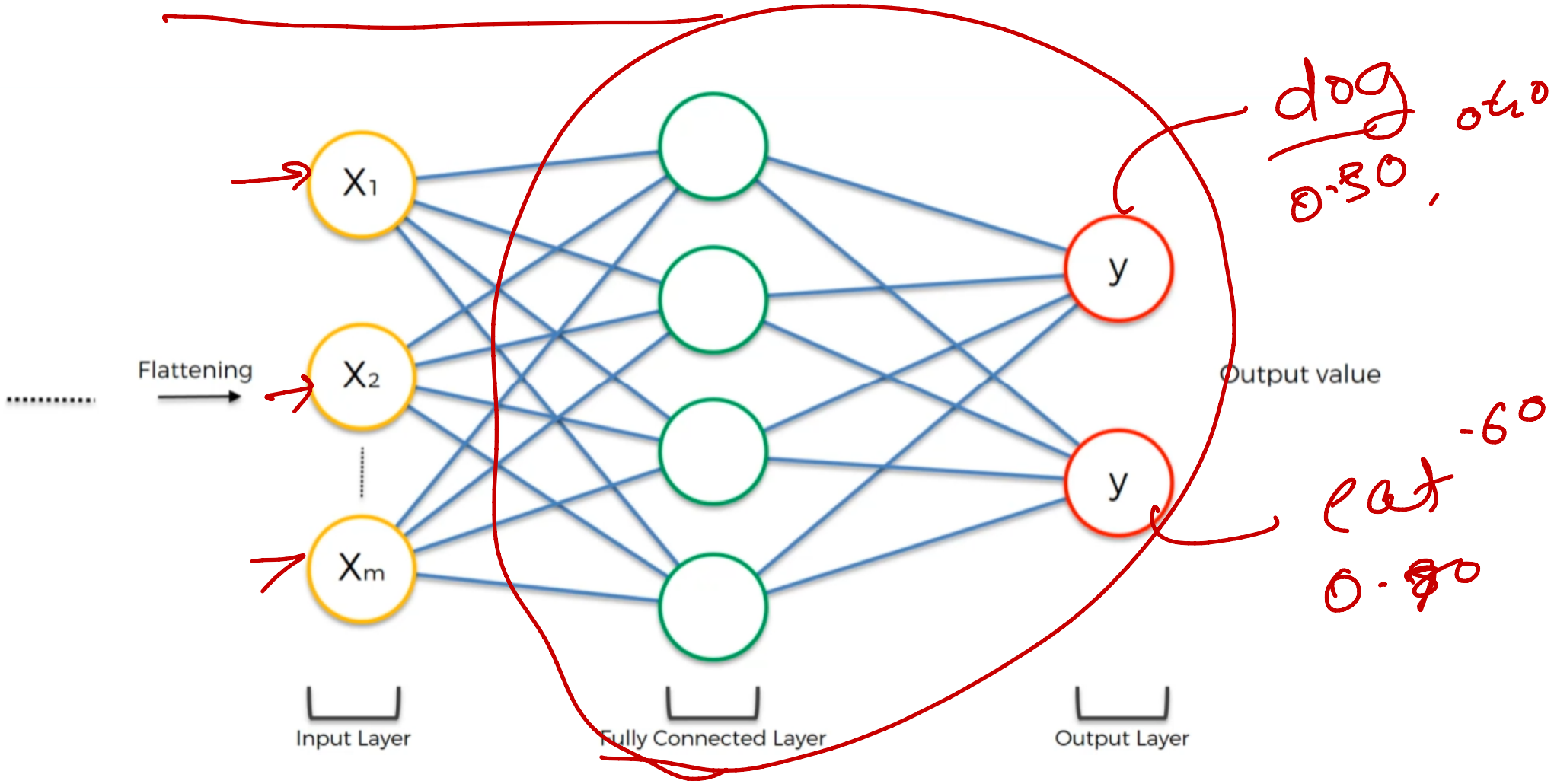




Step 3 – Flattening



Step 4 – Full Connection



Step 4 – Full Connection