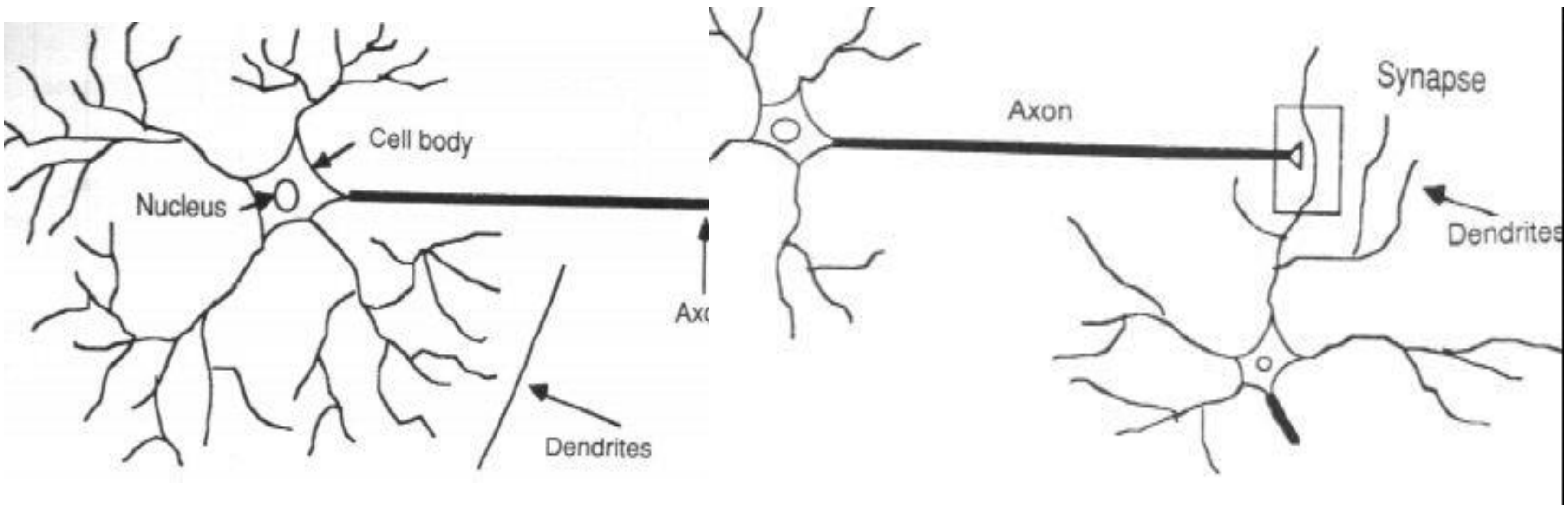
Artificial Neural Network (ANN)

Artificial Neutral Network (ANN)

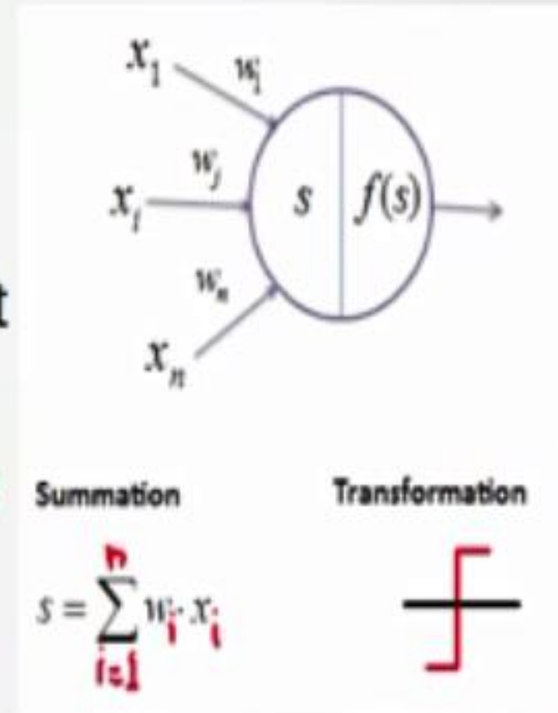
- **ANN** is a system that is based on the biological neural network, such as the brain.
- The brain has approximately 100 billion neurons, which communicate through **electro-chemical signals**.
- The neurons are connected through junctions called synapses.
- Each neuron receives thousands of connections with other neurons, constantly receiving incoming signals to reach the cell body.
- If the resulting sum of the signals above a certain threshold, a response is sent through the axon.
- The ANN attempts to recreate the computational mirror of the biological neural network.



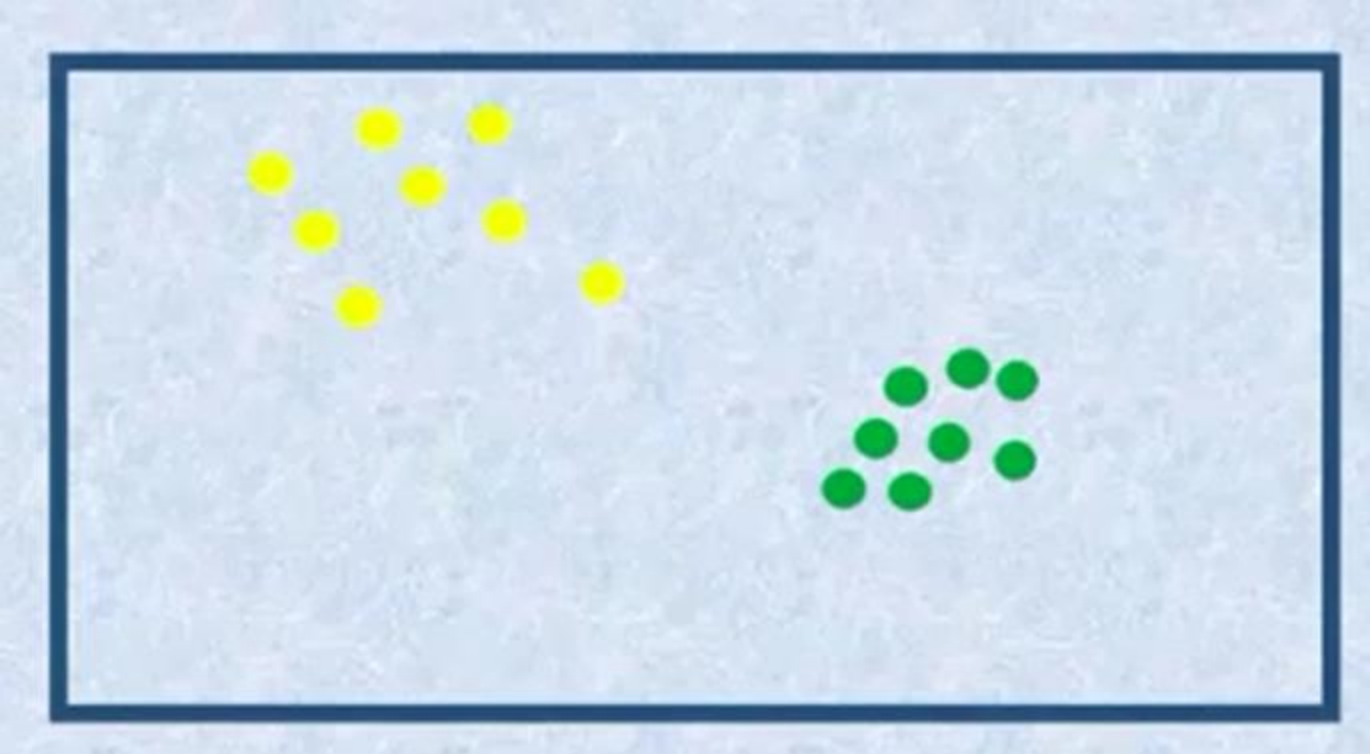
Cell body
Nucleus
Dendrites
Axon
Synapse
Dendrites

Perceptron (An Artificial Neuron)

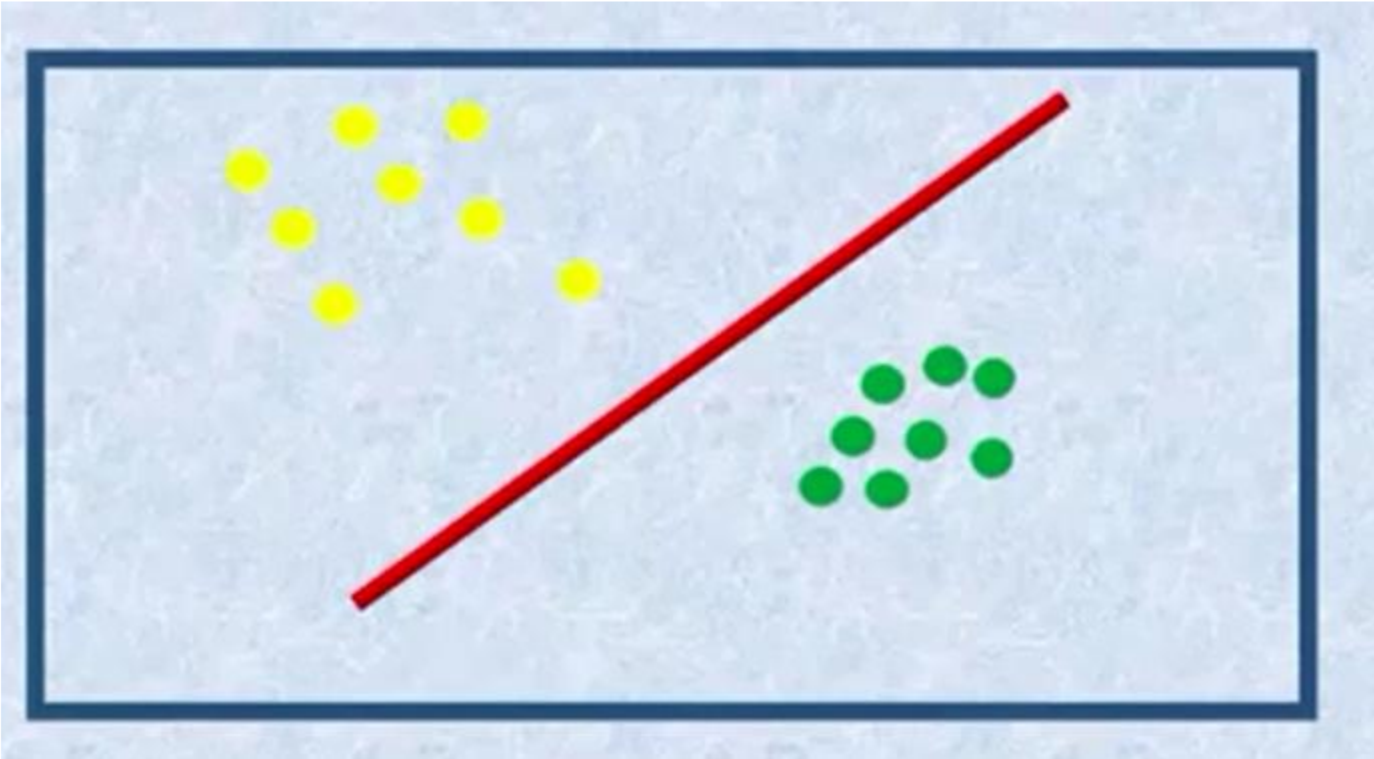
- A perceptron models a neuron
- It receives n inputs (corresponding to features)
- It sums those inputs, checks the result and produces an output
- It is used to classify linearly separable classes
- Often for binary classification



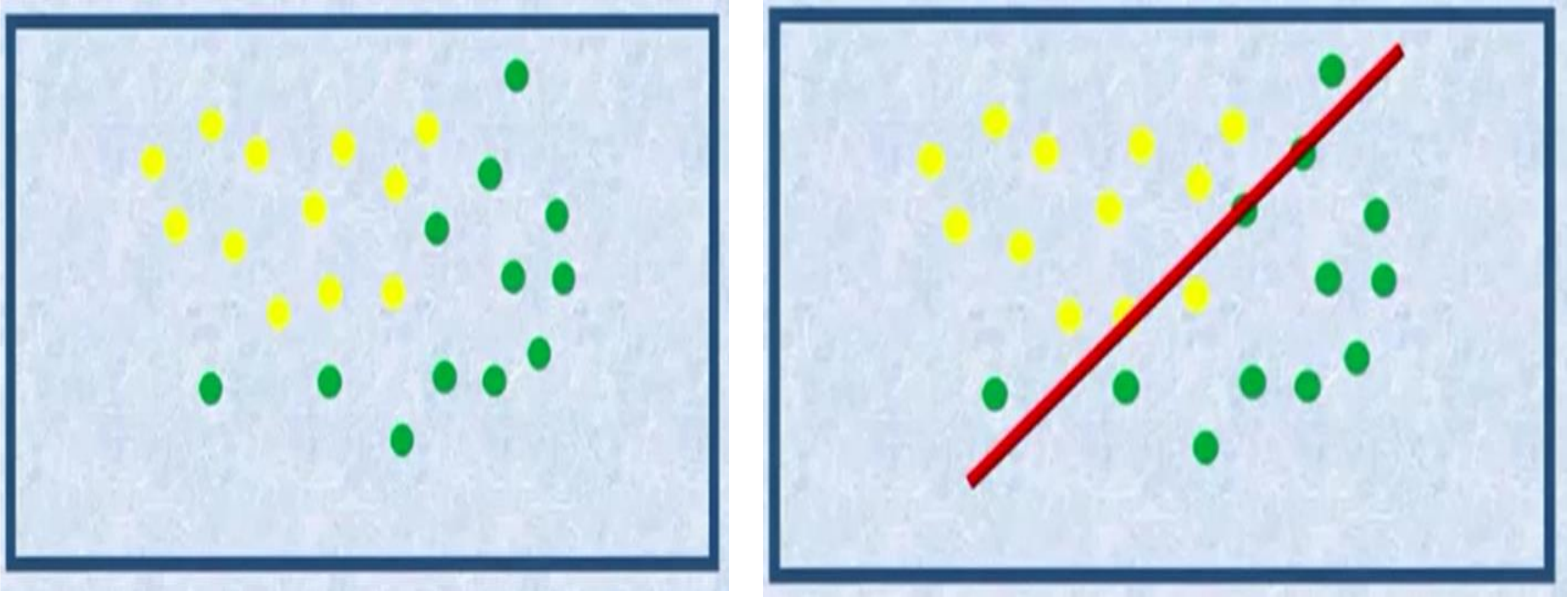
Neural Network and Classification



Linear Classifier

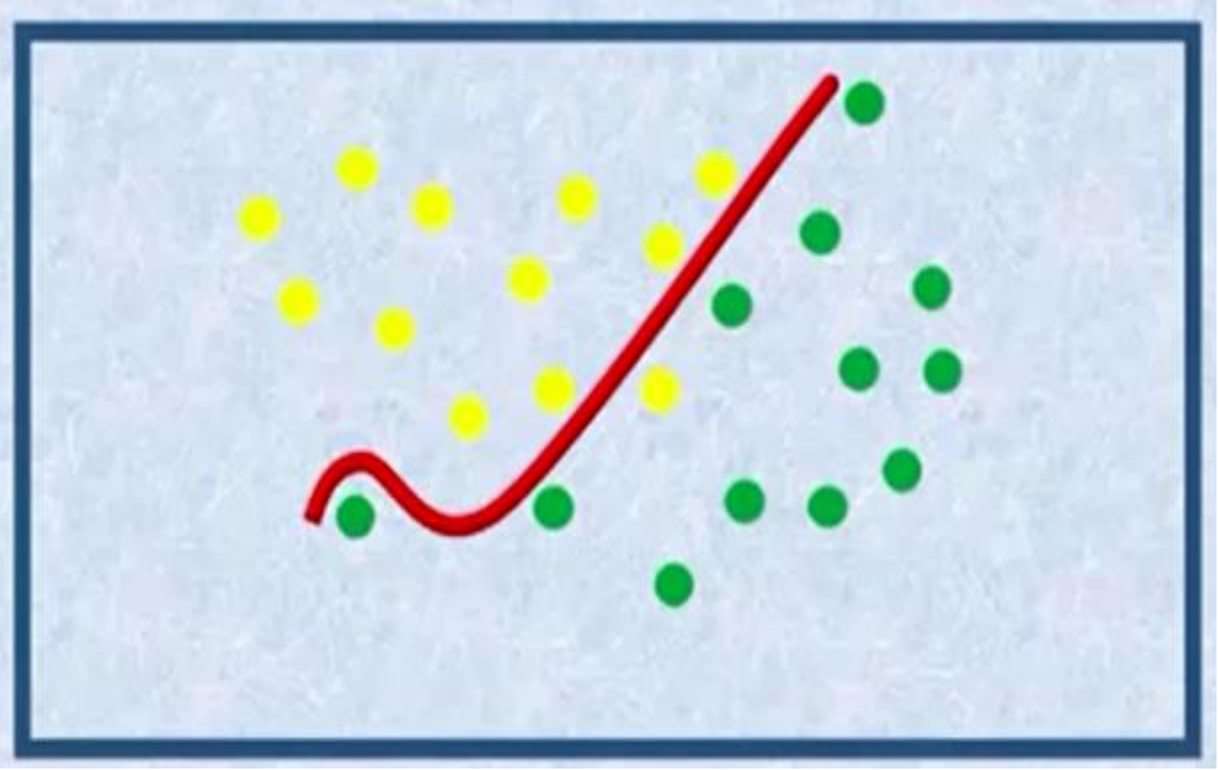


Linear Classifier - Complex Data (Non-Linear)

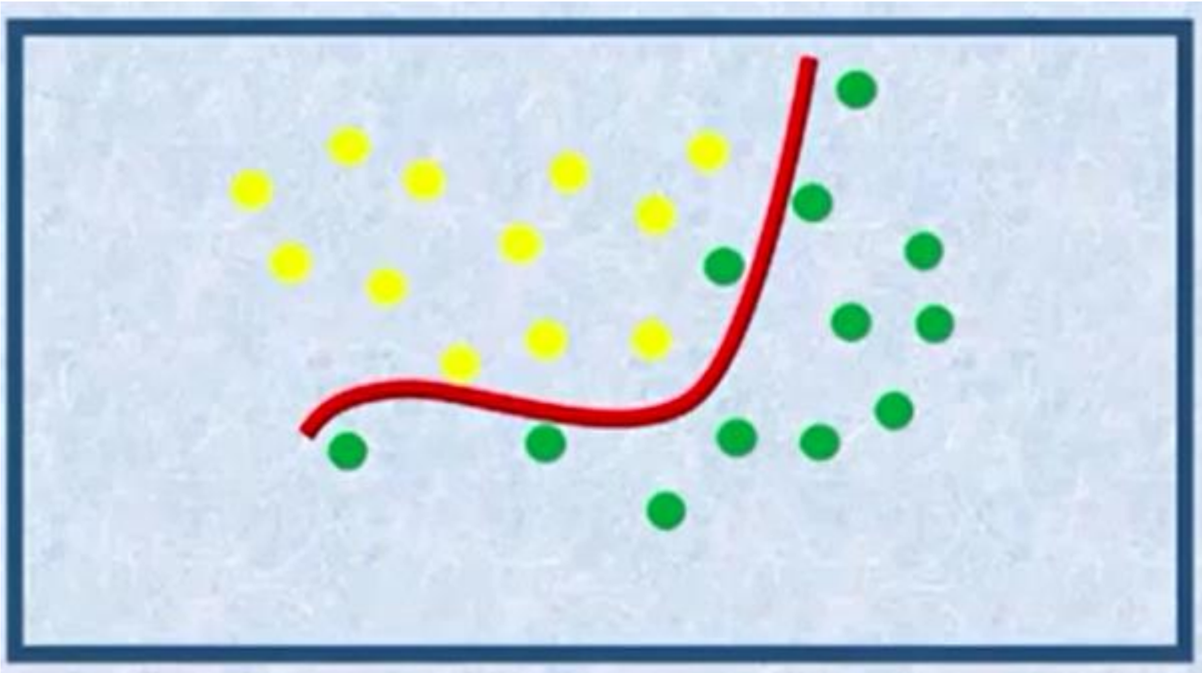


Non-Linear Classifier
Neural Network Training Data

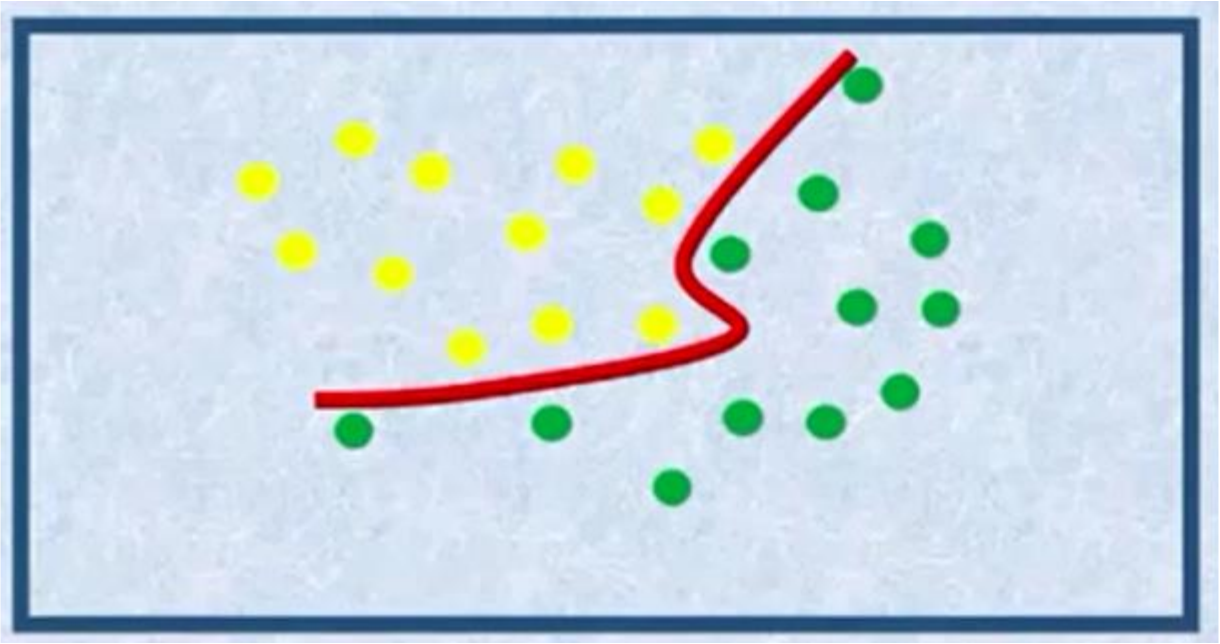
- Epoch 1



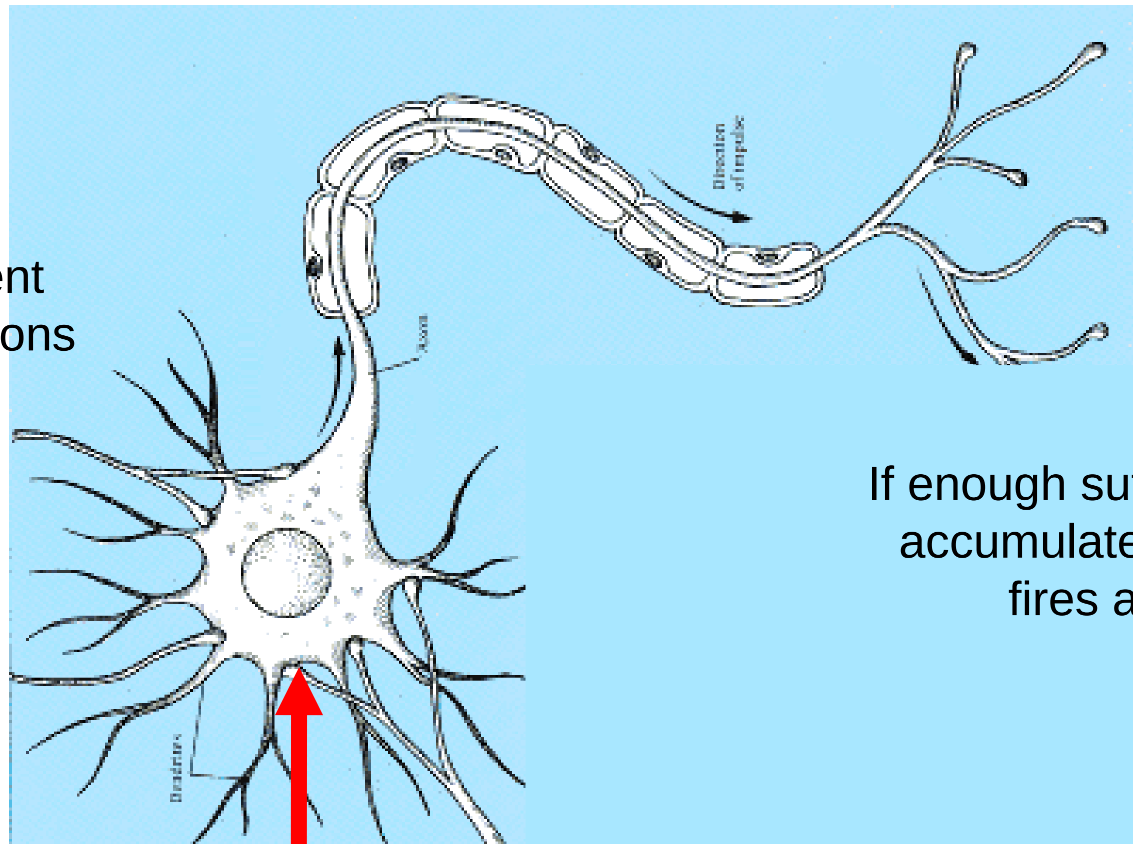
- Epoch2



- Epoch 3



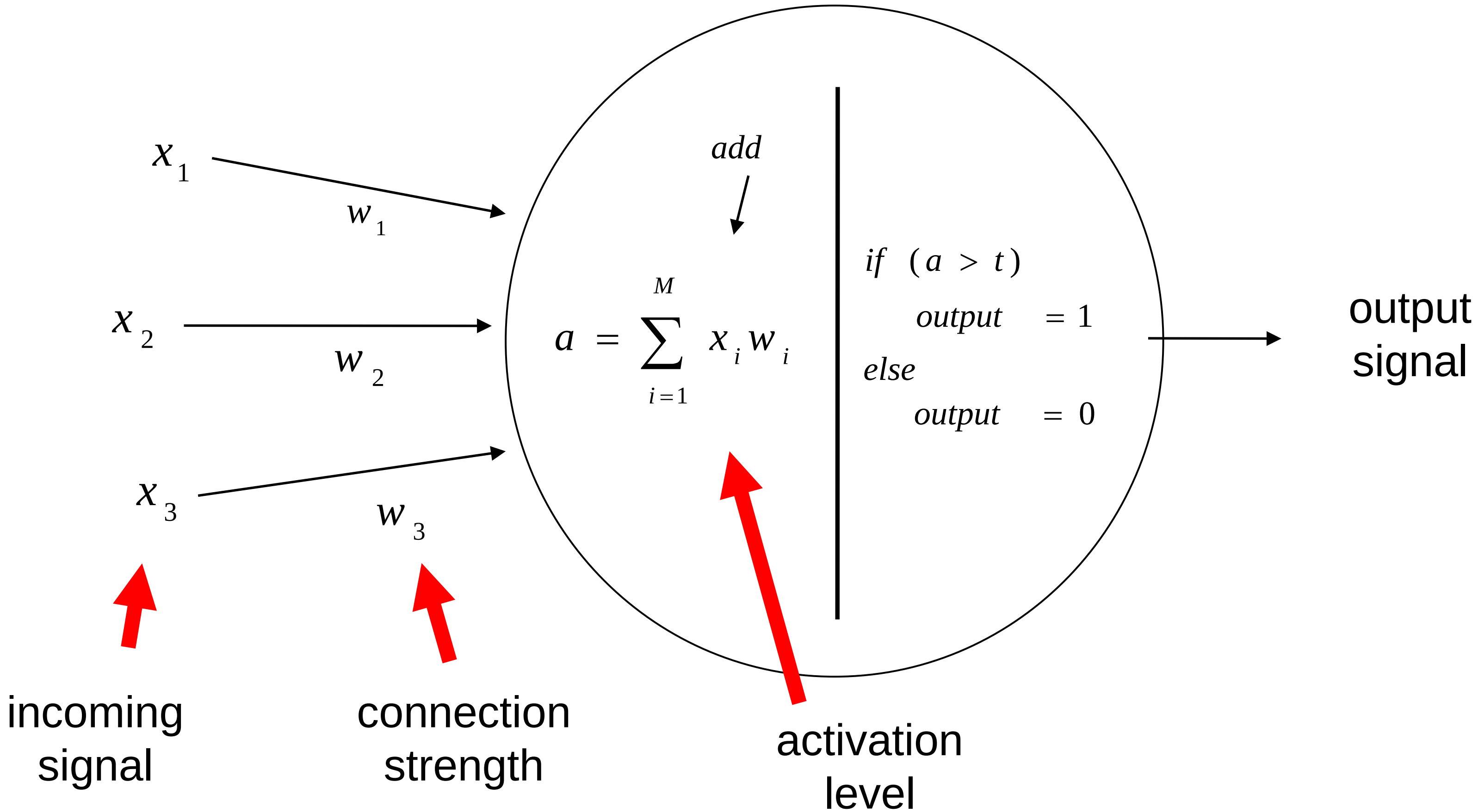
Input signals sent
from other neurons



If enough sufficient signals
accumulate, the neuron
fires a signal.

Connection strengths determine how
the signals are accumulated

- input signals ‘x’ and coefficients ‘w’ are multiplied
- weights correspond to connection strengths
- signals are added up – if they are enough, FIRE!



Calculation...

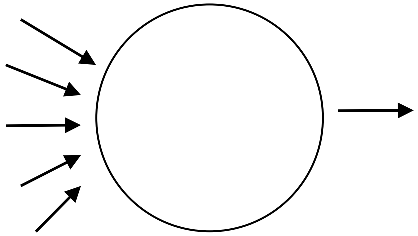
$$a = \sum_{i=1}^{M} x_i w_i$$

Sum notation
(just like a loop from 1 to M)

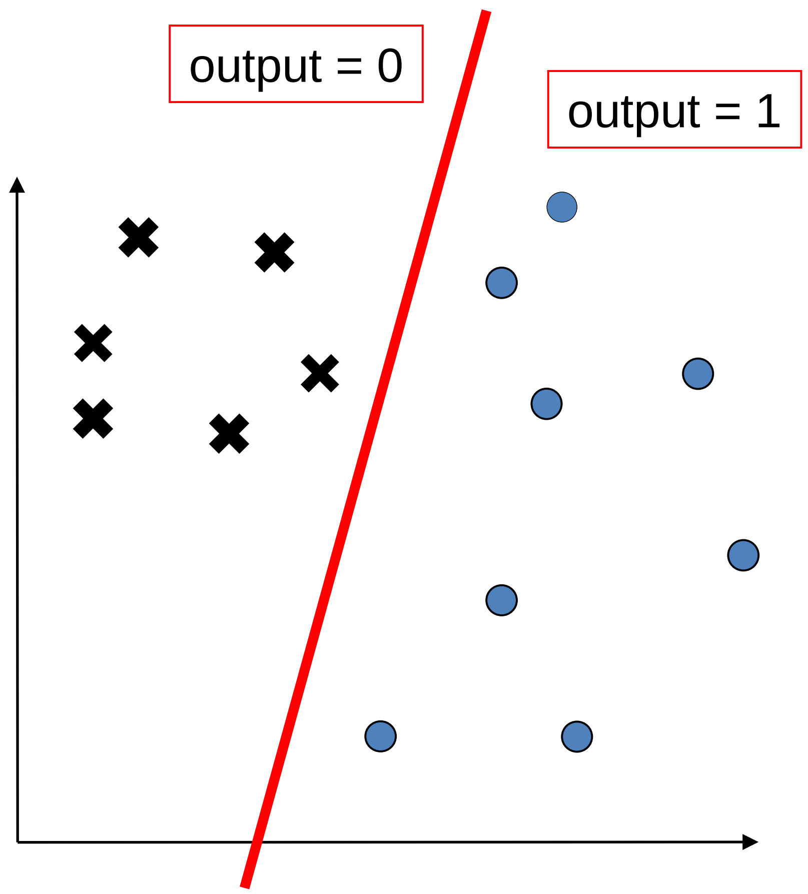
if (activation > threshold) FIRE !

The Perceptron Decision Rule

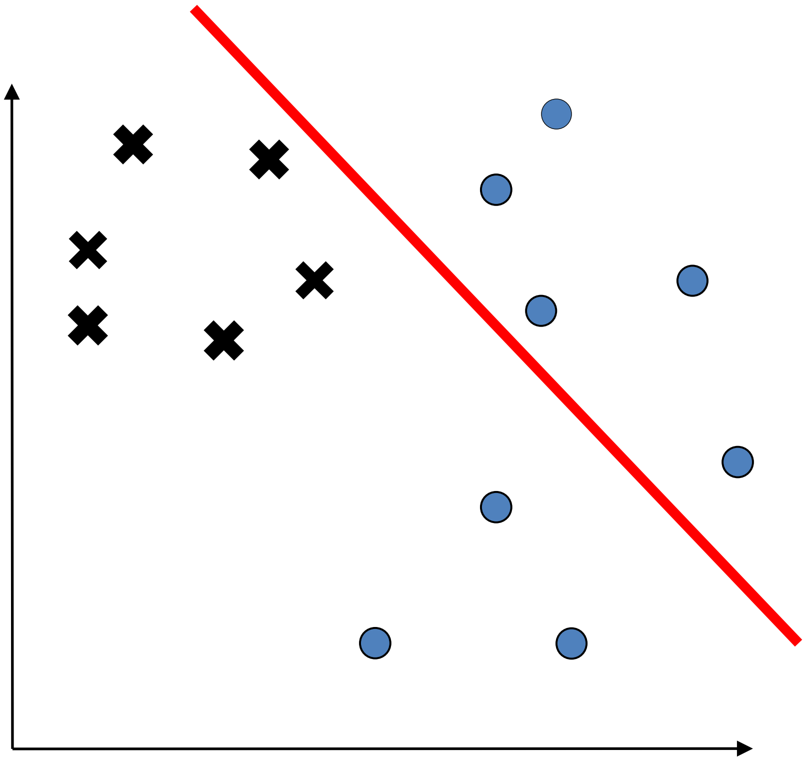
$$\textit{if} \left(\sum_{i=1}^{M} x_i w_i \right) > t \quad \text{then} \quad \textit{output} = 1, \ \text{else} \quad \textit{output} = 0$$



$$\textit{if} \left(\sum_{i=1}^{M} x_i w_i \right) > t \quad \text{then} \quad output = 1, \ \text{else} \quad output = 0$$

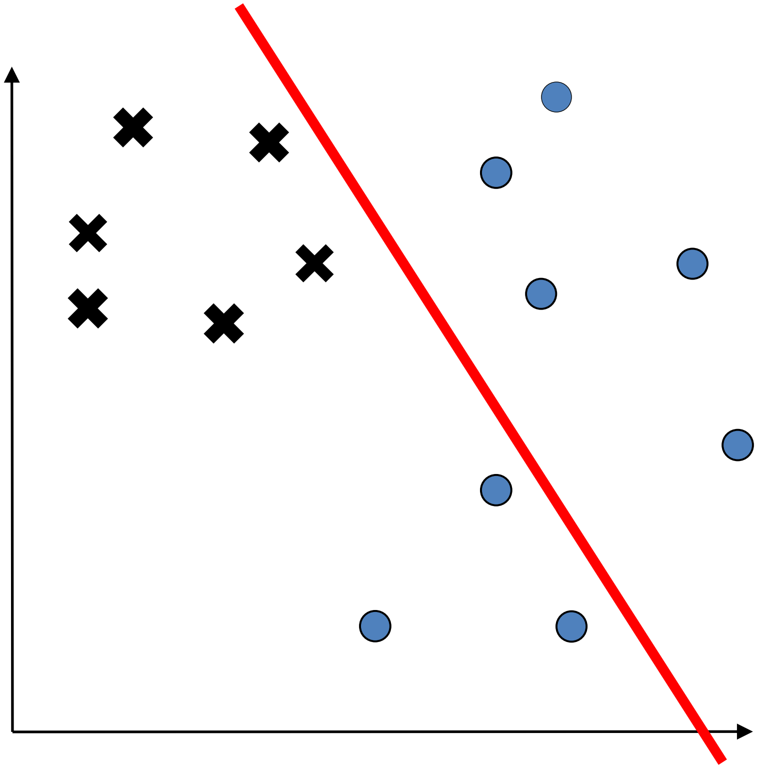


Rugby player = 1
Ballet dancer = 0



Is this a good decision boundary?

$$\textit{if} \quad \left(\sum_{i=1}^{M} x_i w_i \right) > t \quad \text{then} \quad output = 1, \ \text{else} \quad output = 0$$

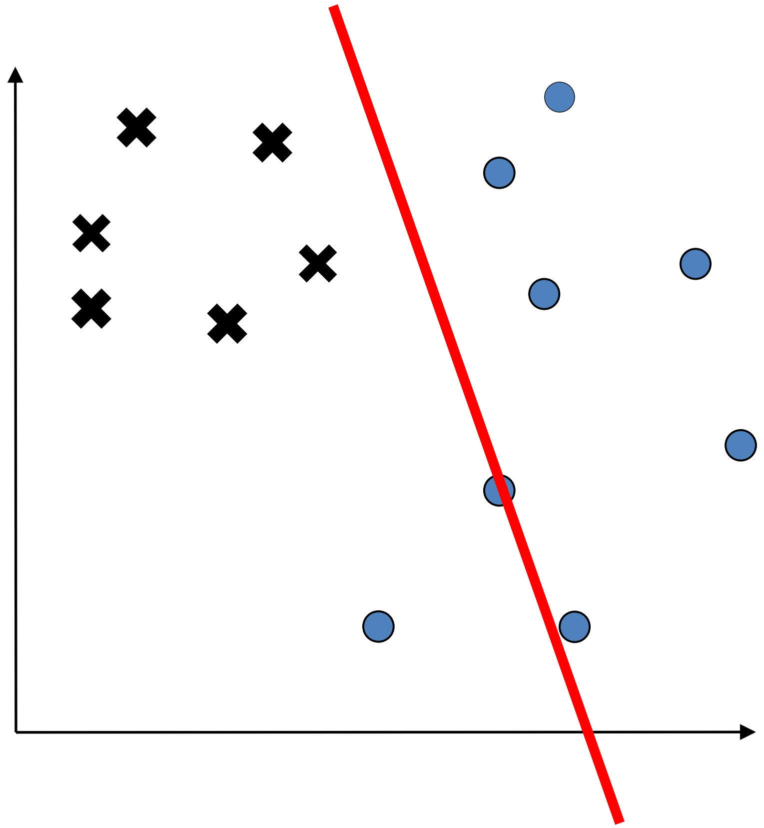


$w_1 = 1.0$

$w_2 = 0.2$

$t = 0.05$

$$\textit{if} \left(\sum_{i=1}^{M} x_i w_i \right) > t \quad \text{then} \quad output = 1, \ \text{else} \quad output = 0$$

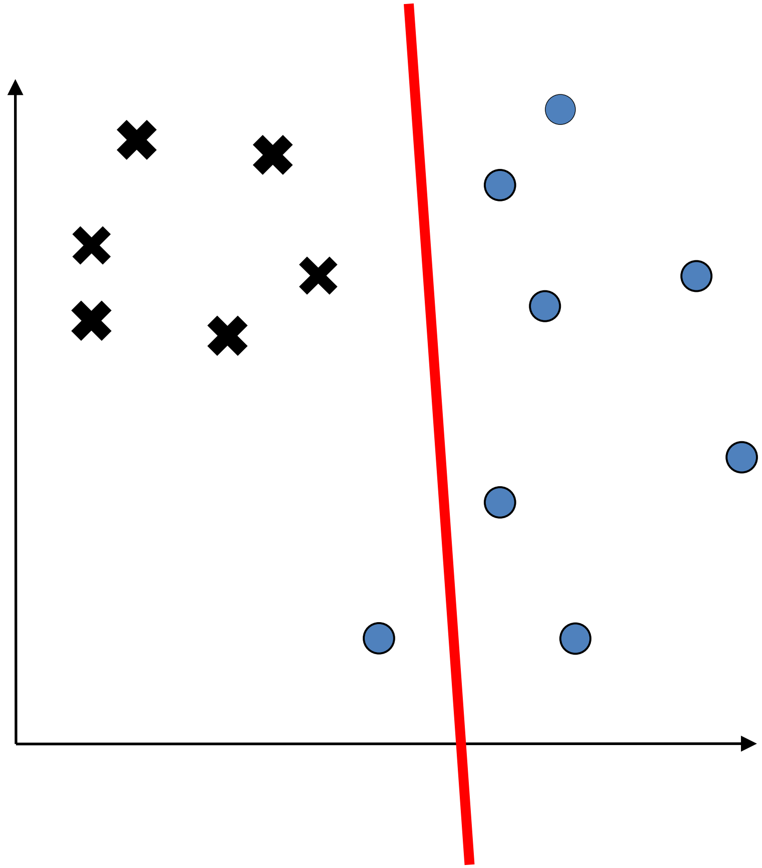


$w_1 = 2.1$

$w_2 = 0.2$

$t = 0.05$

$$\textit{if} \left(\sum_{i=1}^{M} x_i w_i \right) > t \quad \text{then} \quad output = 1, \text{ else } output = 0$$

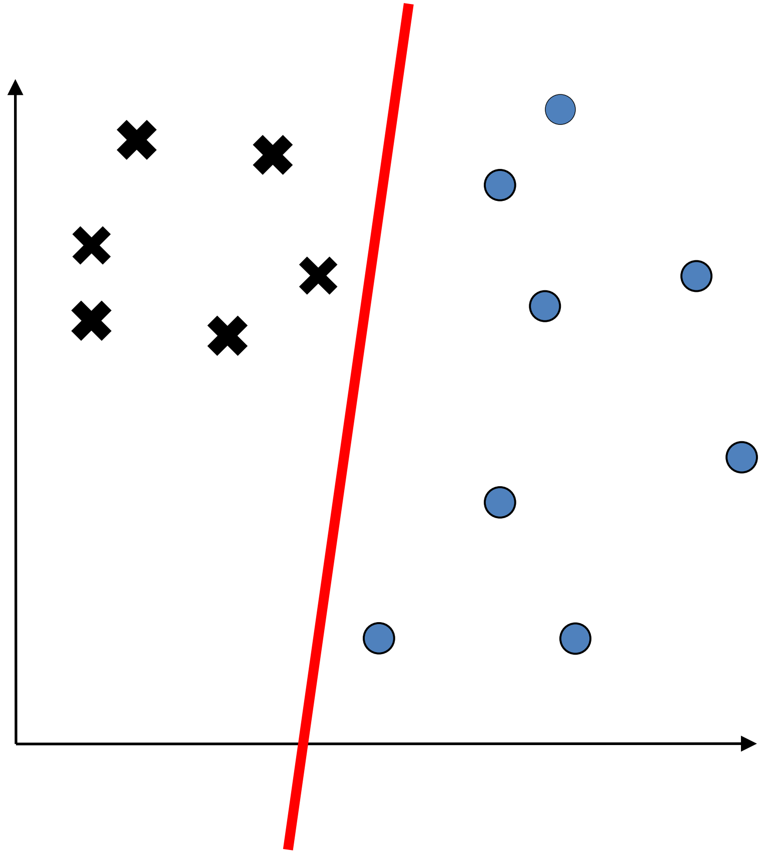


$w_1 = 1.9$

$w_2 = 0.02$

$t = 0.05$

$$\textit{if} \left(\sum_{i=1}^{M} x_i w_i \right) > t \quad \text{then} \quad output = 1, \; \text{else} \quad output = 0$$



$w_1 = -0.8$

$w_2 = 0.03$

$t = 0.05$

Changing the weights/threshold makes the decision boundary move.

Pointless / impossible to do it by hand – only ok for simple 2-D case.

We need an algorithm….

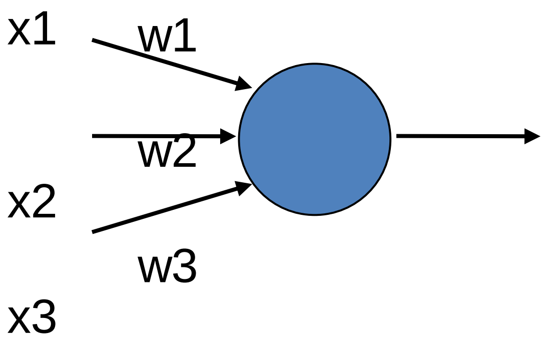
Example

$$x = [\ 1.0,\ 0.5,\ 2.0\ ]$$

$$w = [\ 0.2,\ 0.5,\ 0.5\ ]$$

$$t = 1.0$$

$$a = \sum_{i=1}^{M} x_i w_i$$



Q1. What is the activation, *a*, of the neuron?

Q2. Does the neuron fire?

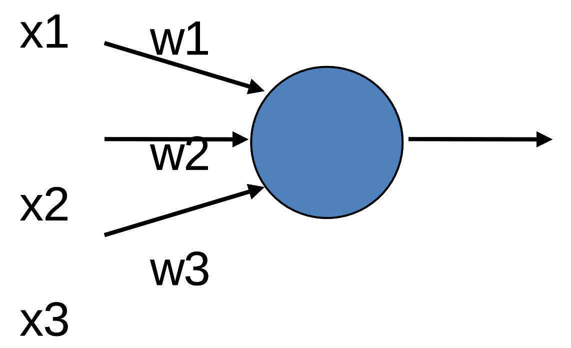
Q3. What if we set threshold at 0.5 and weight #3 to zero?

$$x = [\ 1.0,\ 0.5,\ 2.0\ ]$$

$$w = [\ 0.2,\ 0.5,\ 0.5\ ]$$

$$t = 1.0$$

$$a = \sum_{i=1}^{M} x_i w_i$$



Q1. What is the activation, *a*, of the neuron?

$$a = \sum_{i=1}^{M} x_i w_i = (1.0 \times 0.2) + (0.5 \times 0.5) + (2.0 \times 0.5) = 1.45$$

Q2. Does the neuron fire?

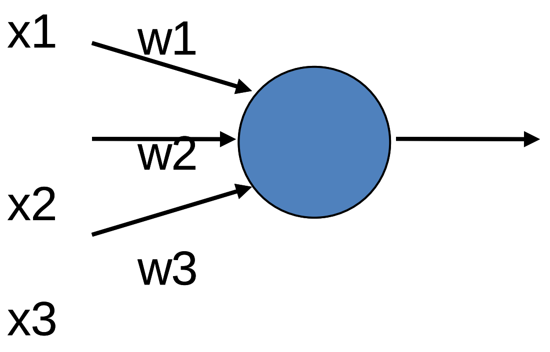
if (activation > threshold) output=1 else output=0
…. So yes, it fires.

$$x = [\ 1.0,\ 0.5,\ 2.0\ ]$$

$$w = [\ 0.2,\ 0.5,\ 0.5\ ]$$

$$t = 1.0$$

$$a = \sum_{i=1}^{M} x_i w_i$$



Q3. What if we set threshold at 0.5 and weight #3 to zero?

$$a = \sum_{i=1}^{M} x_i w_i = (1.0 \times 0.2) + (0.5 \times 0.5) + (2.0 \times 0.0) = 0.45$$

if (activation > threshold) output=1 else output=0

…. So no, it does not fire..

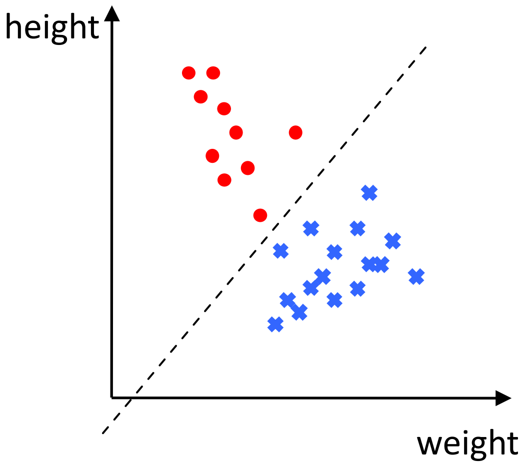
We need a more sophisticated model…

$$\text{if } (weight > t) \text{ then } \text{"player"} \text{ else } \text{"dancer"}$$

$$\text{if } (f(\vec{x}) > t) \text{ then } \text{"player"} \text{ else } \text{"dancer"}$$

$$x_1 = height \ (cm)$$

$$x_2 = weight \ (kg)$$



The Perceptron

$$f(\vec{x}) = (w_1 * x_1) + (w_2 * x_2)$$

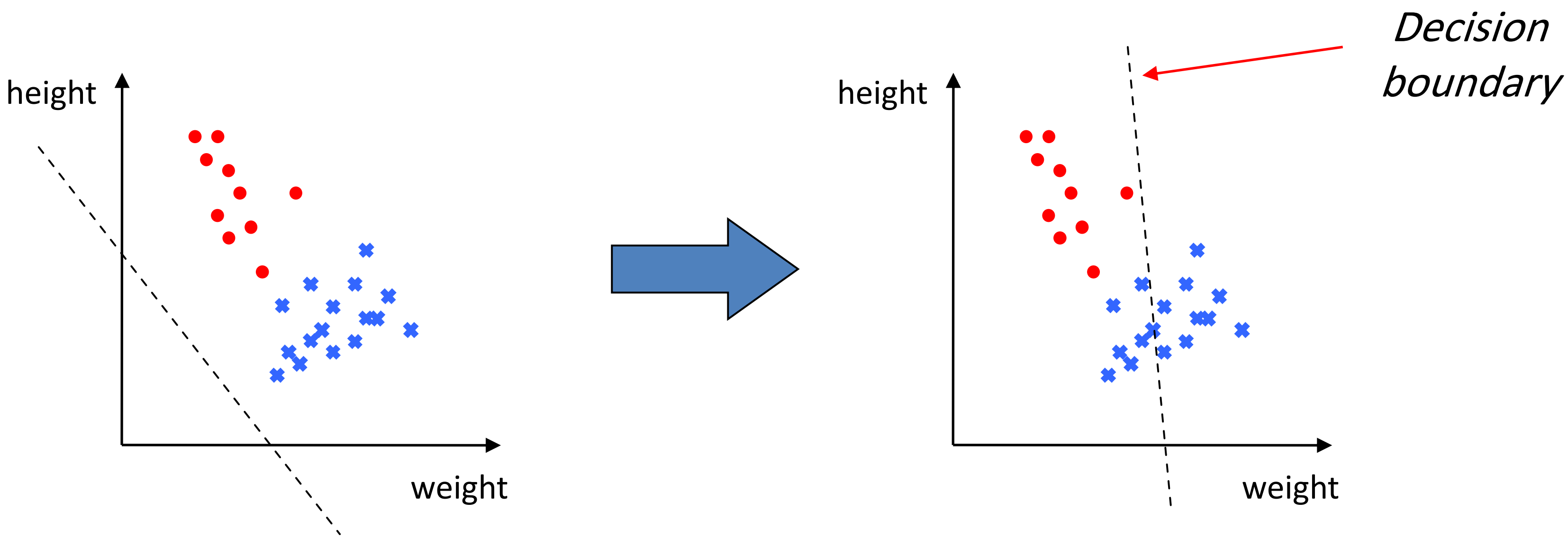
$$= \sum_{i=1}^{d} w_i x_i$$

The Perceptron

$$\text{if} \quad f(\vec{x}) > t \ \text{then} \quad \text{"player"} \quad \text{else} \quad \text{"dancer"}$$

$$f(\vec{x}) = (w_1 * x_1) + (w_2 * x_2)$$

$$= \sum_{i=1}^{d} w_i x_i$$



w_1 and w_2 change the position of the DECISION BOUNDARY

The Perceptron

Model

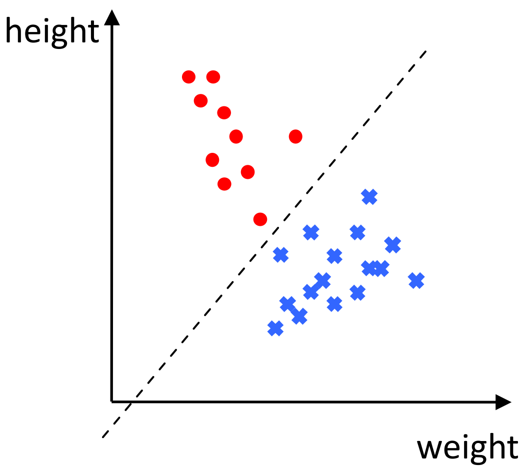
$$\text{if } \sum_{i=1}^{d} w_i x_i > t \text{ then } \hat{y} = 1 \text{ else } \hat{y} = 0 \quad \begin{cases} \textit{"player"} & = 1 \\ \textit{"dancer"} & = 0 \end{cases}$$

Error function

Number of mistakes (a.k.a. classification error)

Learning algo.

??? .... need to optimise the w and t values...



Perceptron Learning Rule

$$\boxed{\text{new weight} = \text{old weight} + \underbrace{0.1 \times (\text{trueLabel} - \text{output}) \times \text{input}}_{\textit{update}}}$$

What weight updates do these cases produce?

if… (`target = 0, output = 0`) …. then update = ?
if… (`target = 0, output = 1`) …. then update = ?
if… (`target = 1, output = 0`) …. then update = ?
if… (`target = 1, output = 1`) …. then update = ?

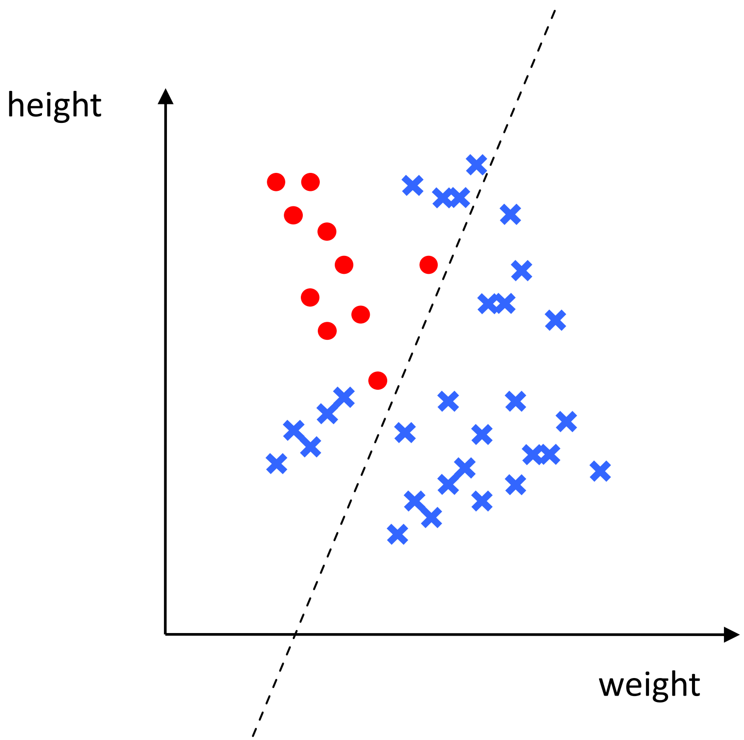
Learning algorithm for the Perceptron

```
initialise weights to random numbers in range -1 to +1
for n = 1 to NUM_ITERATIONS
        for each training example (x,y)
                calculate activation
                for each weight
                        update weight by learning rule
                end
        end
end
```

Perceptron convergence theorem:

If the data is linearly separable, then application of the Perceptron learning rule will find a separating decision boundary, within a finite number of iterations

New data…. *“non-linearly separable”*



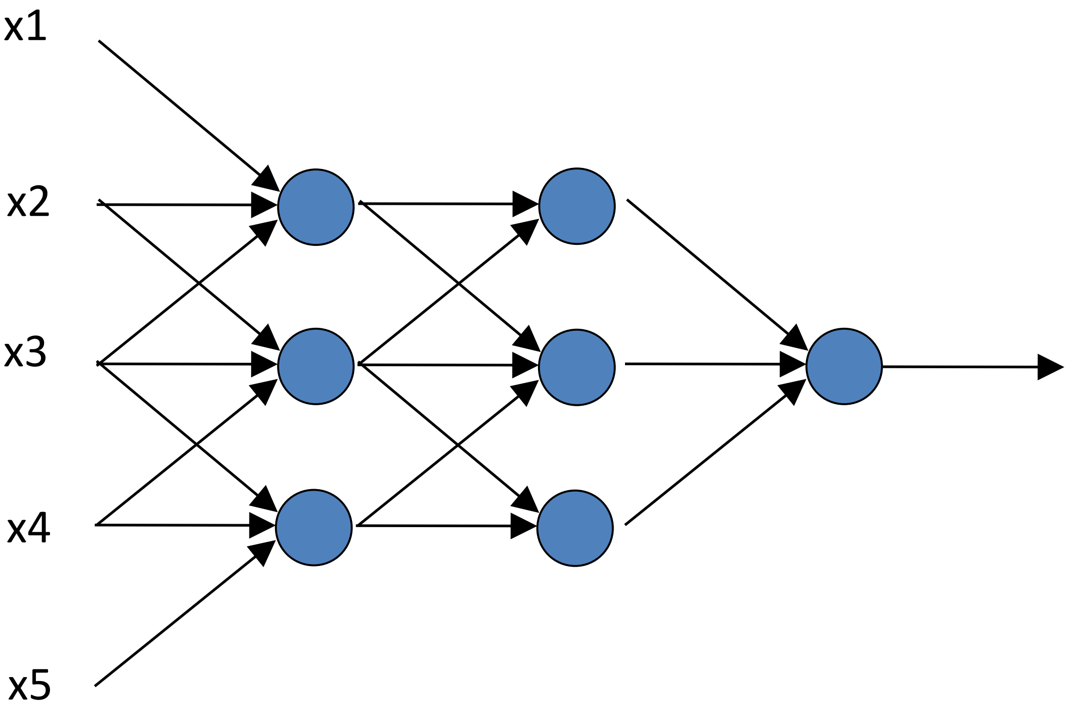
Our model does not match the problem!

(AGAIN!)

$$\text{if } \sum_{i=1}^{d} w_i x_i > t \text{ then } \text{"player"} \quad \text{else } \text{"dancer"}$$

Many mistakes!

Multilayer Perceptron

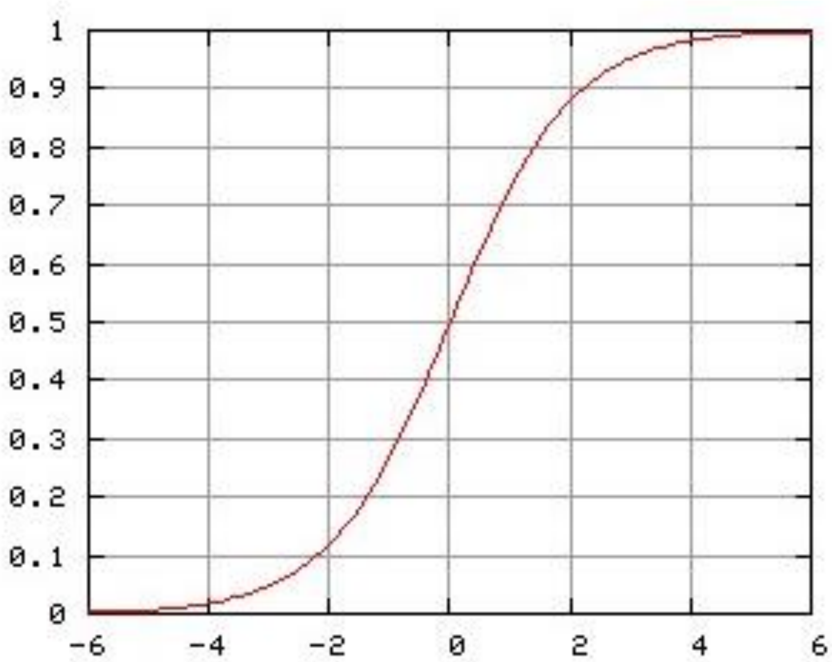


Sigmoid activation – no more thresholds needed ☺

$$\cancel{\text{if } \sum_{i=1}^{d} w_i x_i > t \text{ then } \hat{y} = 1 \text{ else } \hat{y} = 0}$$

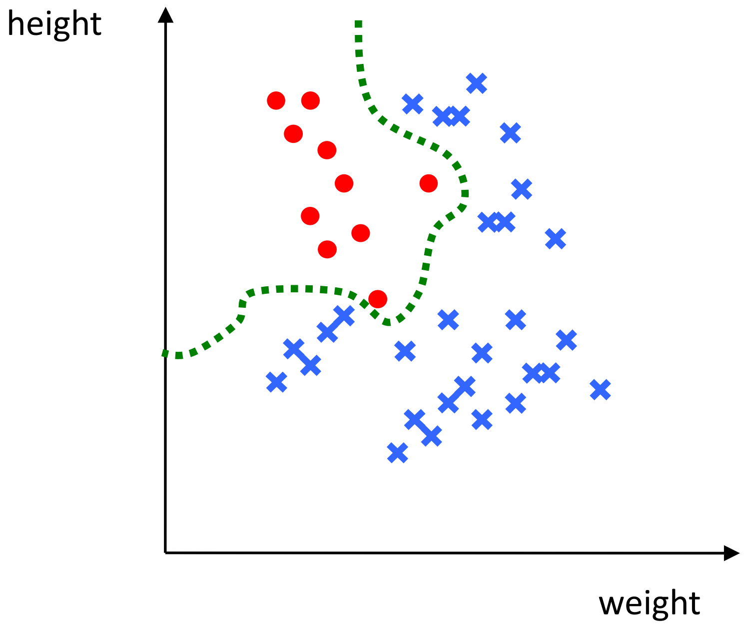
$$a = \frac{1}{1 + \exp(-\sum_{i=1}^{d} w_i x_i)}$$

activation level



$$\sum_{i=1}^{d} w_i x_i$$

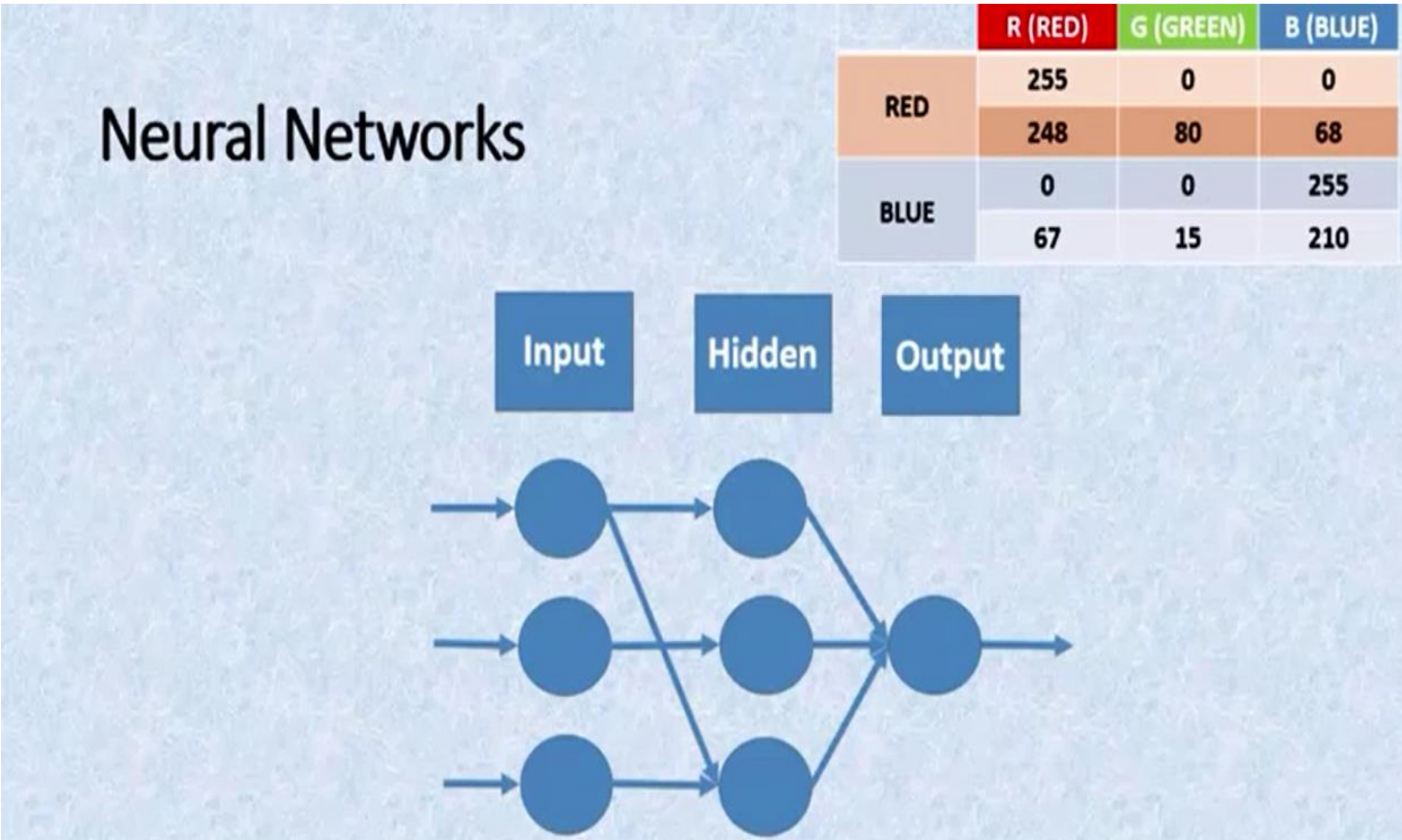
MLP decision boundary – nonlinear problems, solved!



Simple Neural network Classification Example

	R (RED)	G (GREEN)	B (BLUE)
RED	255	0	0
	248	80	68
BLUE	0	0	255
	67	15	210

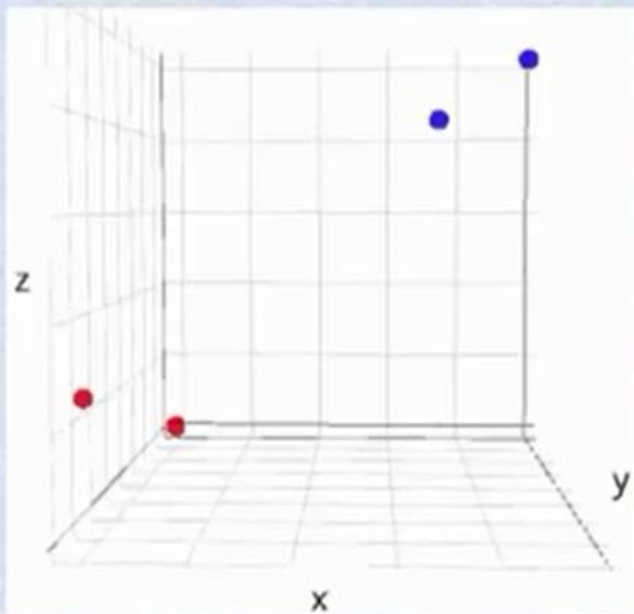
Network Architecture



		R (RED)	G (GREEN)	B (BLUE)
RED		255	0	0
		248	80	68
BLUE		0	0	255
		67	15	210

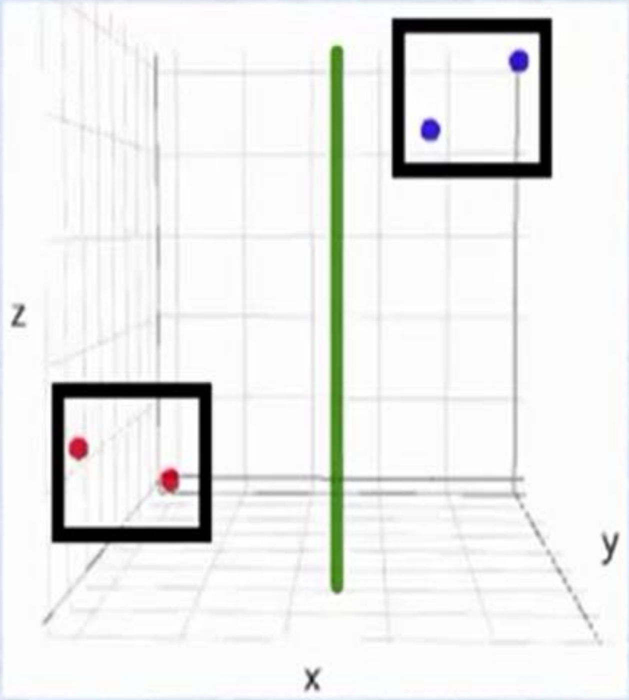
Neural Networks

	R (RED)	G (GREEN)	B (BLUE)
RED	255	0	0
	248	80	68
BLUE	0	0	255
	67	15	210



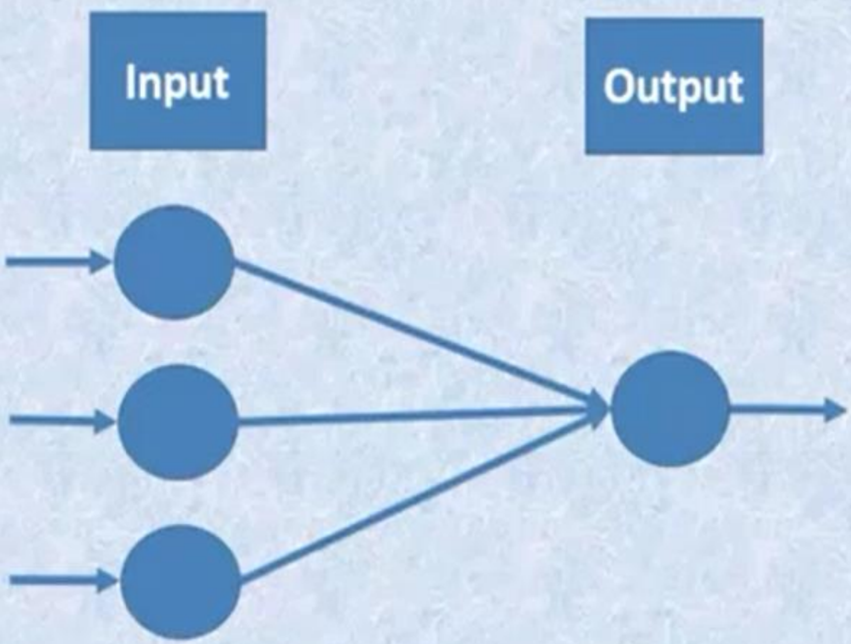
Neural Networks

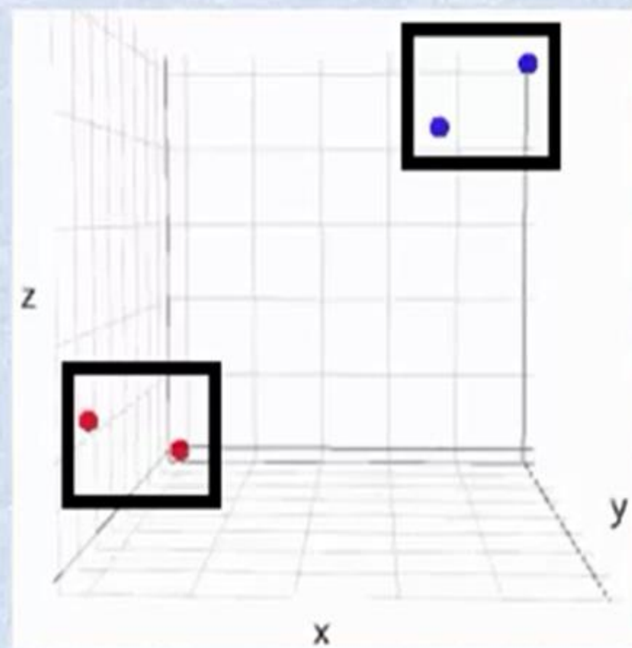
	R (RED)	G (GREEN)	B (BLUE)
RED	255	0	0
	248	80	68
BLUE	0	0	255
	67	15	210



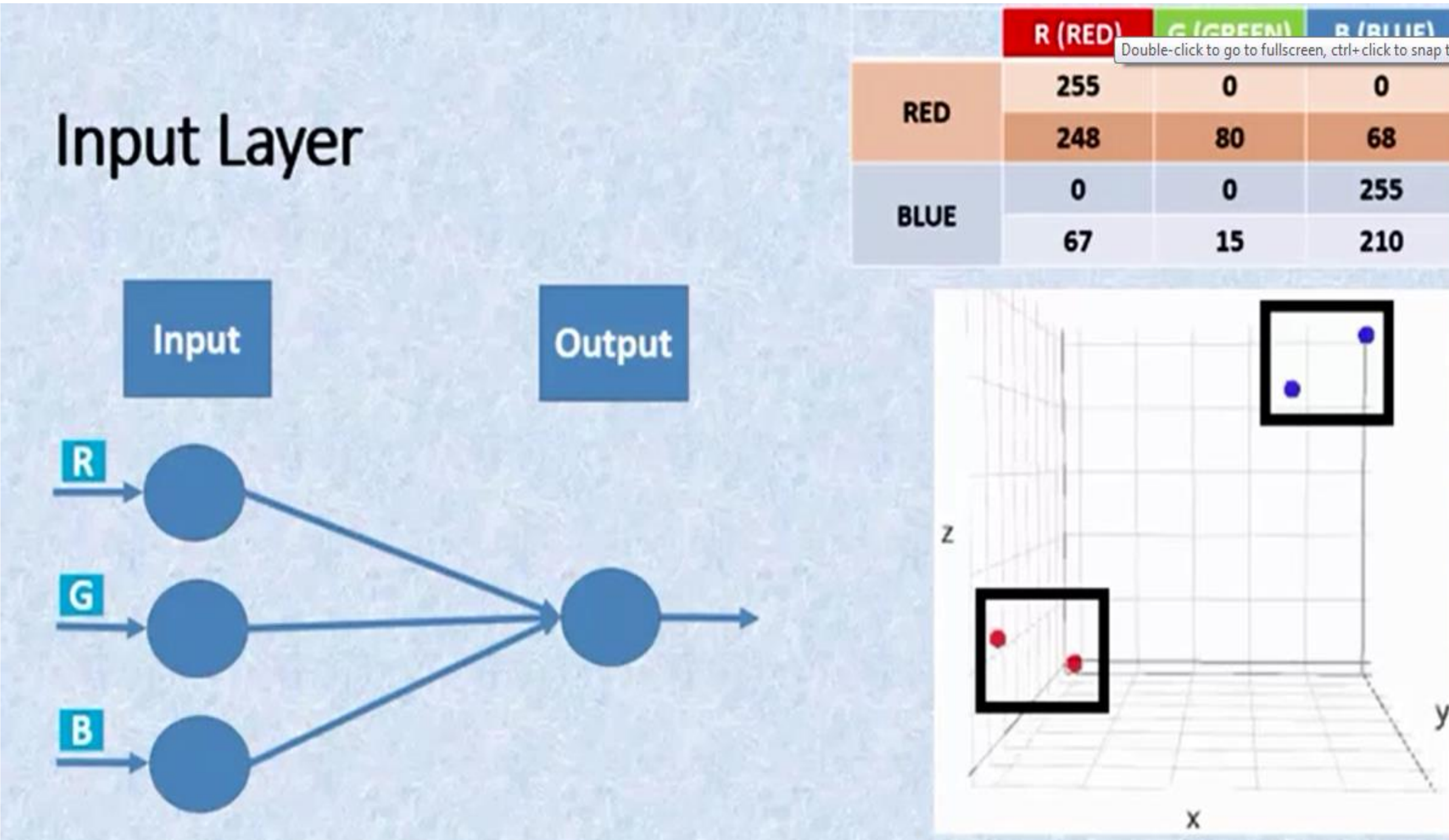
Input Layer

	R (RED)	G (GREEN)	B (BLUE)
RED	255	0	0
	248	80	68
BLUE	0	0	255
	67	15	210





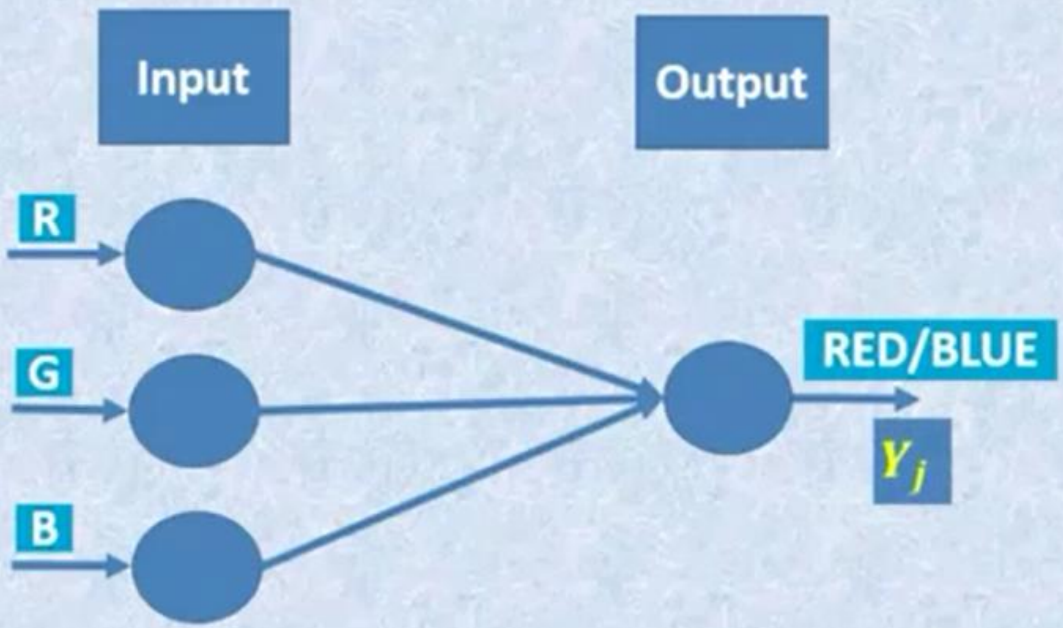
Input Layer

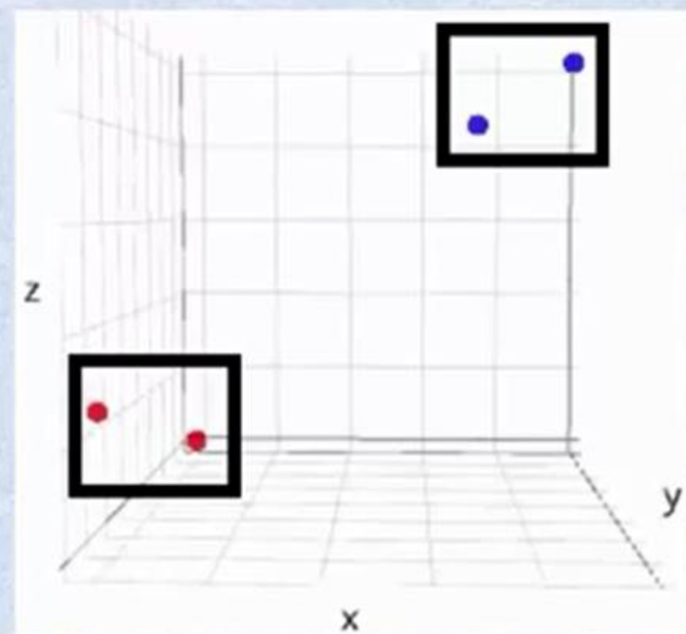

Input Layer
R (RED)
Double-click to go to fullscreen, ctrl+click to snap to
RED
255
0
0
248
80
68
BLUE
0
0
255
67
15
210
Input
Output
R
G
B
z
y
x

Output Layer

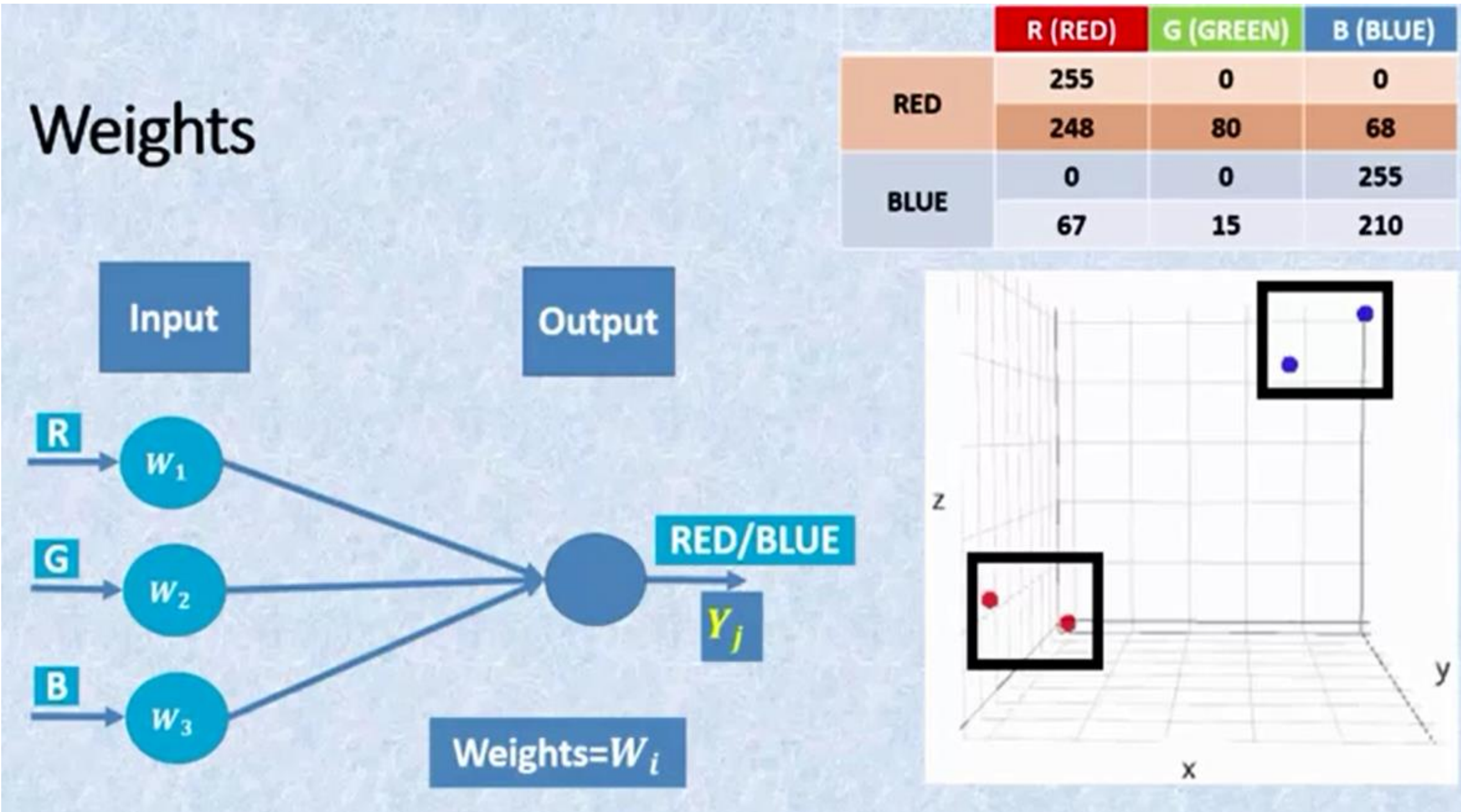
Output Layer

	R (RED)	G (GREEN)	B (BLUE)
RED	255	0	0
	248	80	68
BLUE	0	0	255
	67	15	210



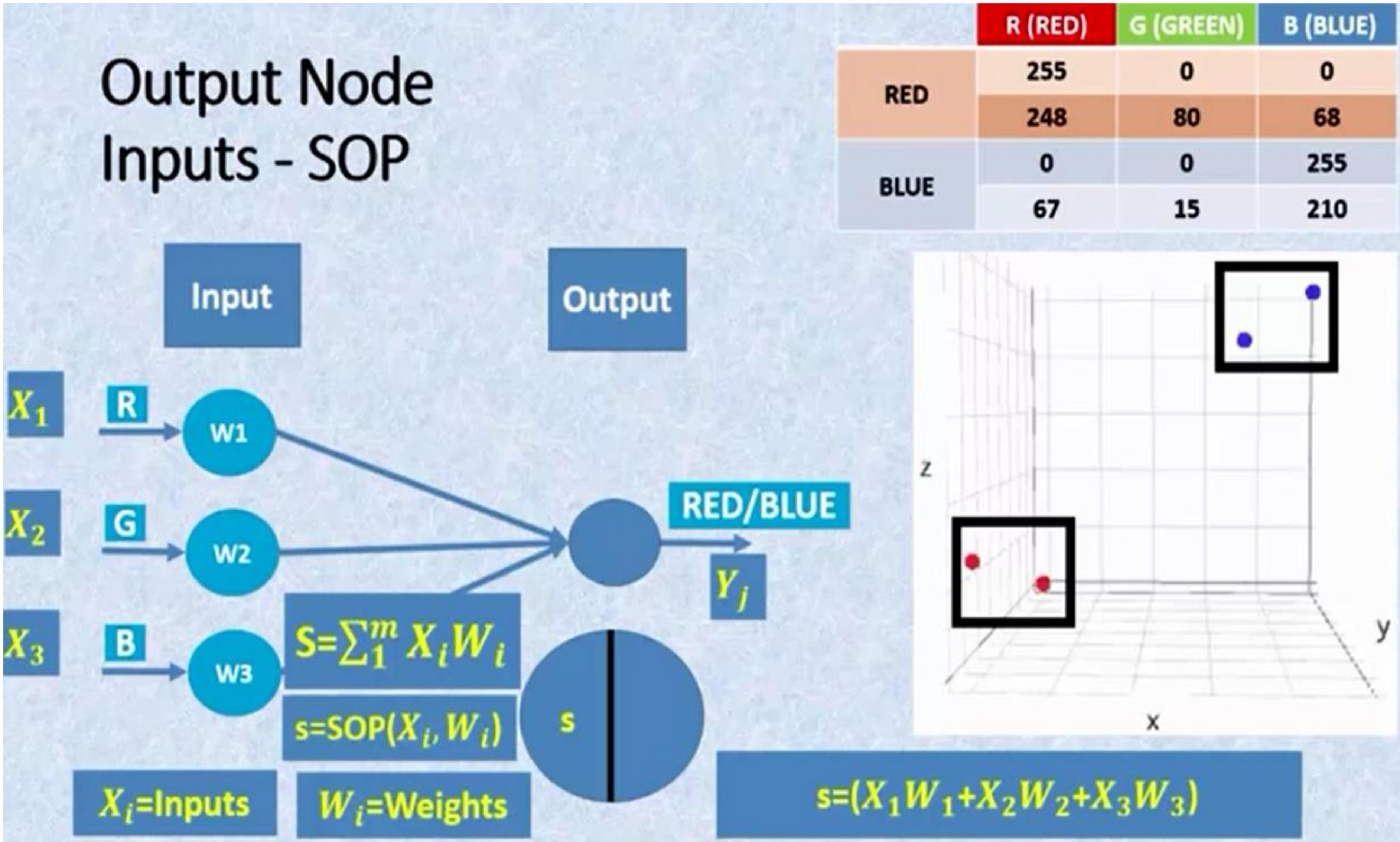


Weights



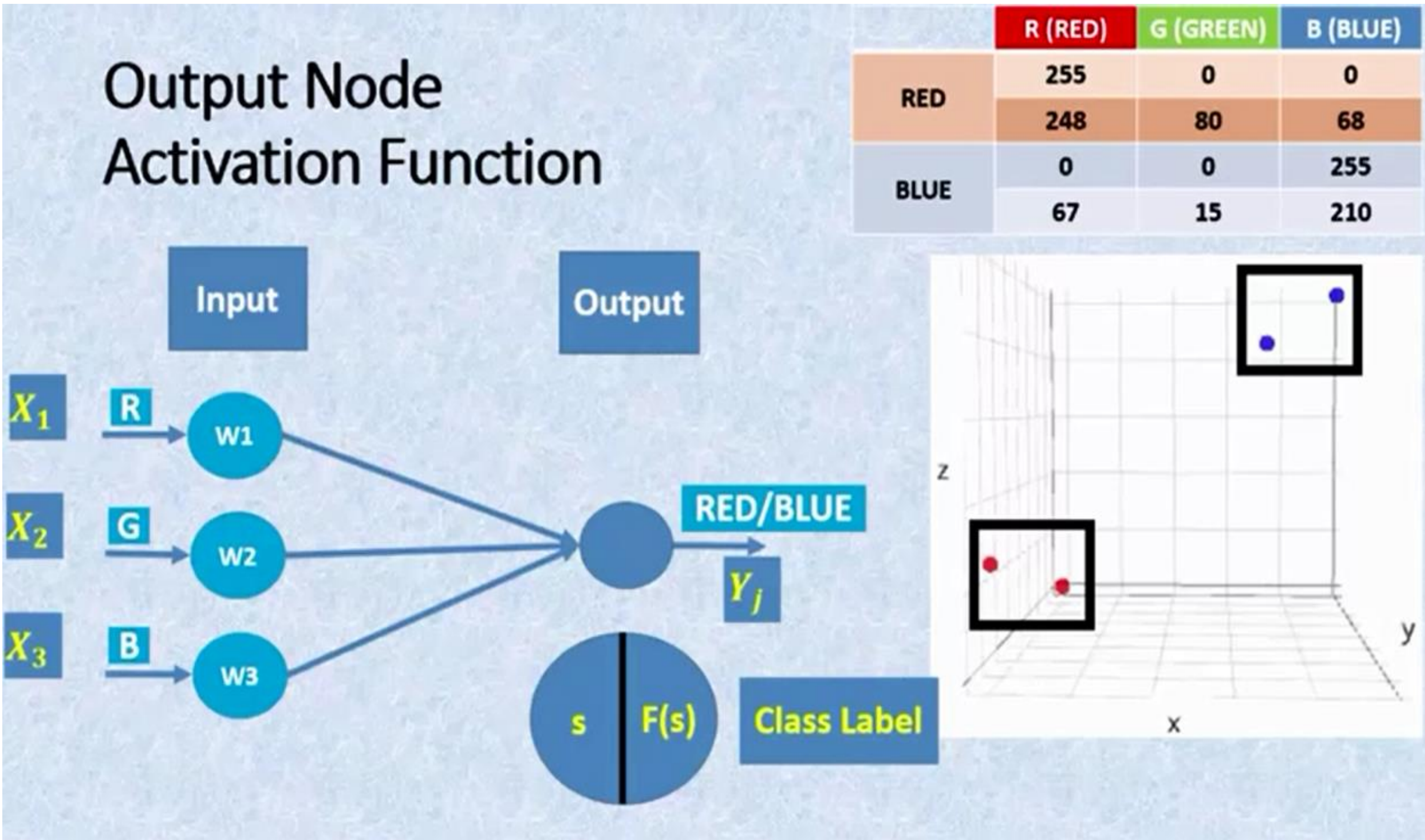
	R (RED)	G (GREEN)	B (BLUE)
RED	255	0	0
	248	80	68
BLUE	0	0	255
	67	15	210

Mapping between I/O

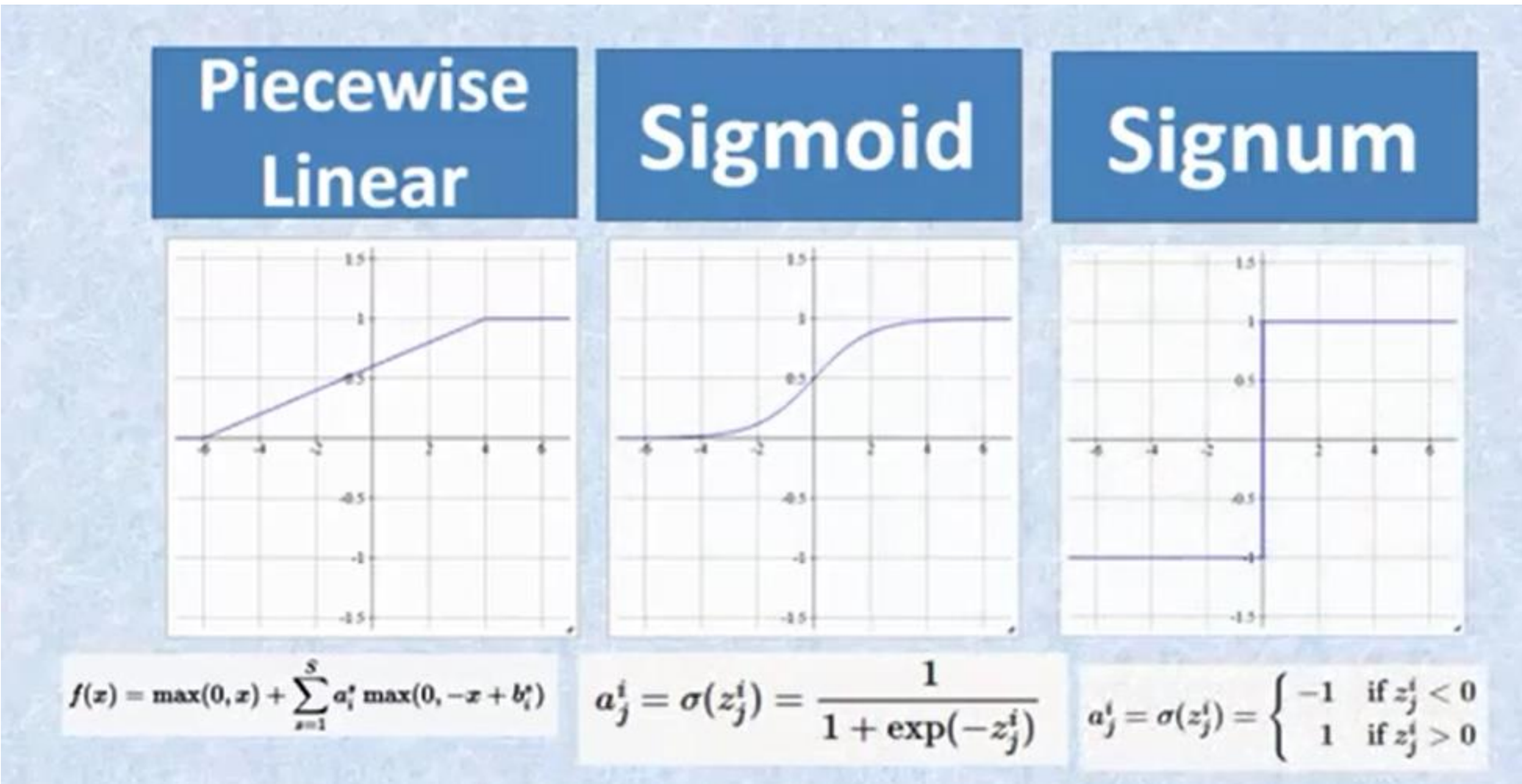


	R (RED)	G (GREEN)	B (BLUE)
RED	255	0	0
	248	80	68
BLUE	0	0	255
	67	15	210

Activation Function



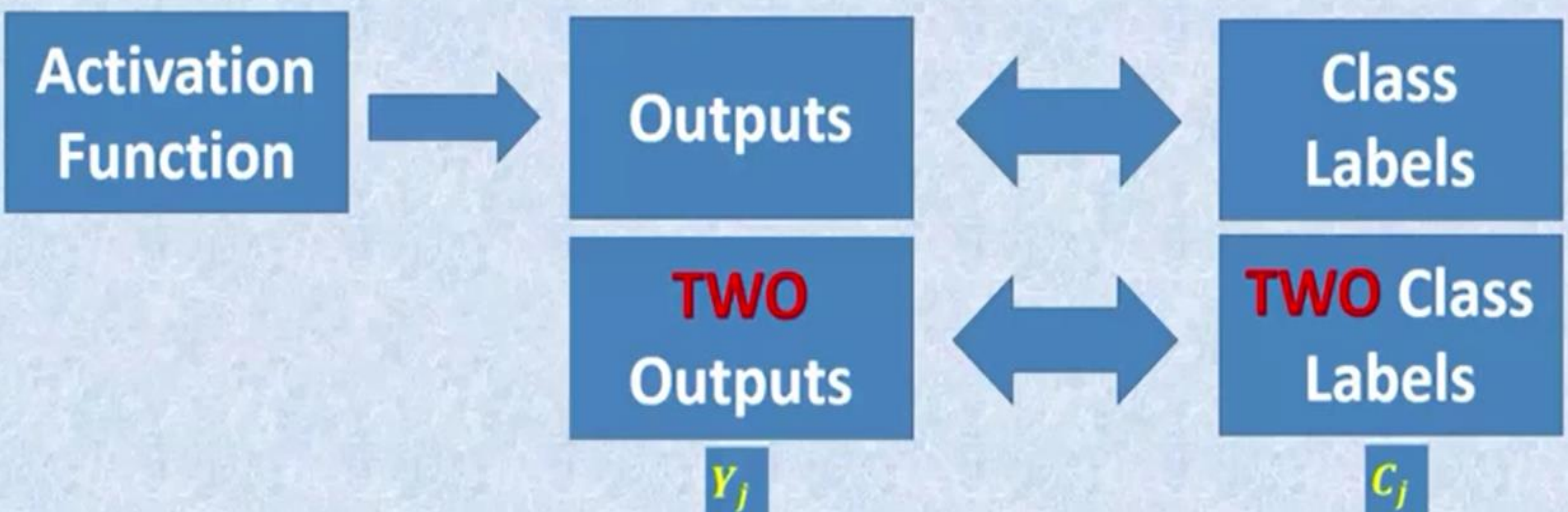
Activation Function Types



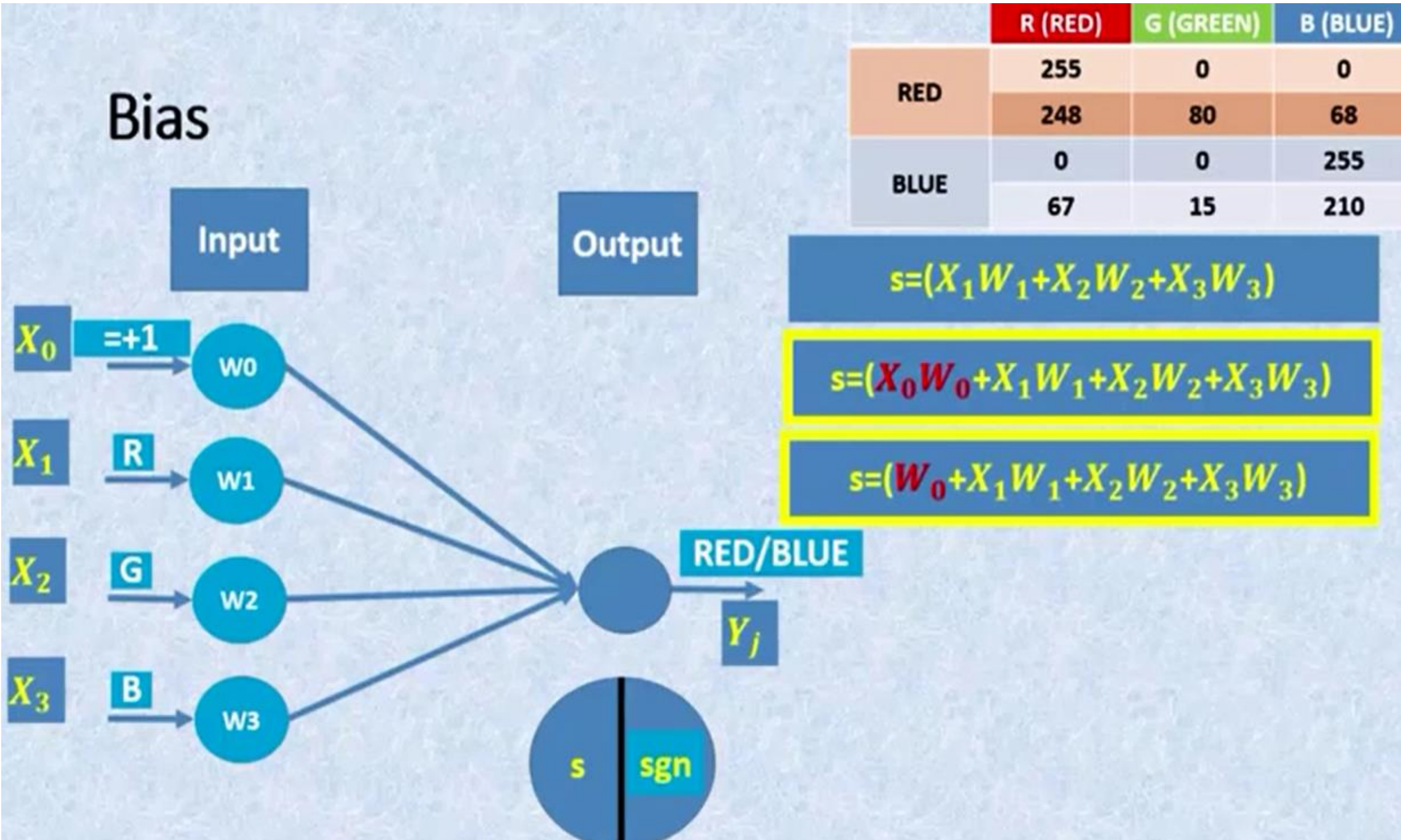
Activation Functions

	R (RED)	G (GREEN)	B (BLUE)
RED	255	0	0
	248	80	68
BLUE	0	0	255
	67	15	210

Which activation function to use?

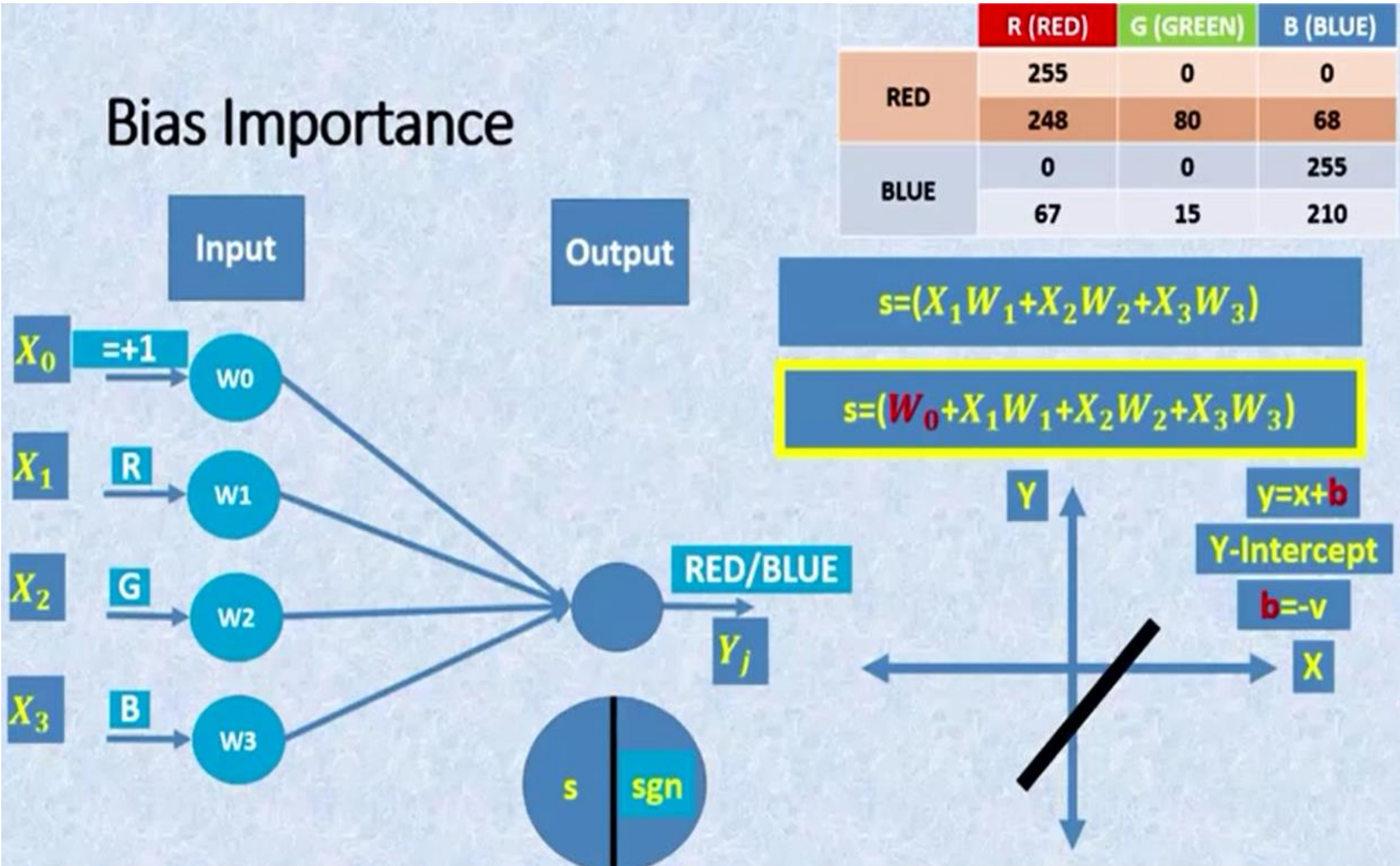


Bias

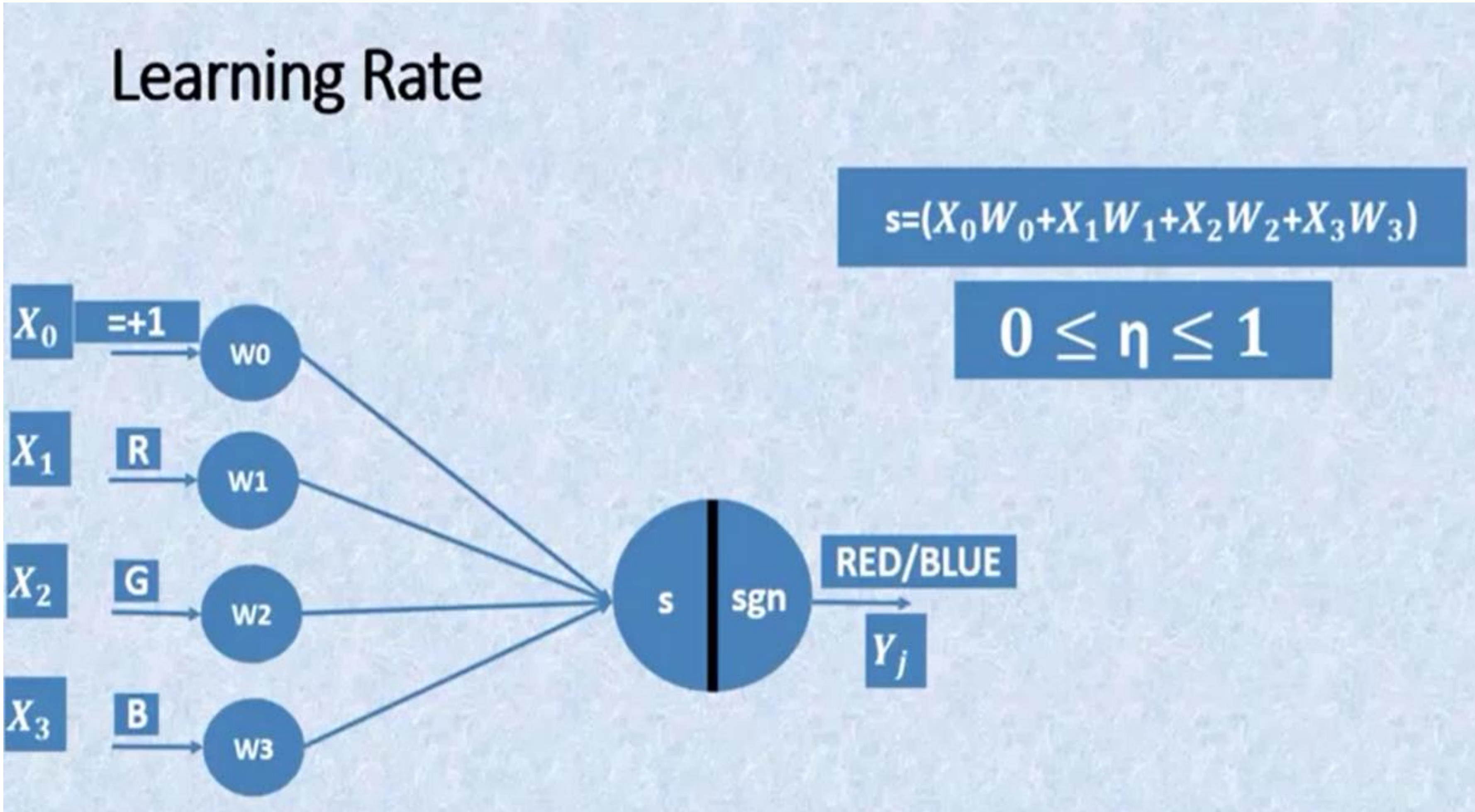


Bias
R (RED)
G (GREEN)
B (BLUE)
RED
255 0 0
248 80 68
BLUE
0 0 255
67 15 210
Input
Output
$s=(X_1W_1+X_2W_2+X_3W_3)$
$s=(X_0W_0+X_1W_1+X_2W_2+X_3W_3)$
$s=(W_0+X_1W_1+X_2W_2+X_3W_3)$
X_0
=+1
W0
X_1
R
W1
X_2
G
W2
X_3
B
W3
RED/BLUE
Y_j
s
sgn

Bias Importance



Learning Rate



Neural Network Parameters

- Input Neurons + Bias
- Weights + Bias Weight
- Sum of product (SOP)..s
- Activation Function (sgn)
- Output (Y_j).....>Actual class
- Learning Rate
- Step n [0,1,2,3,....etc]
- Desired Output (d_j)

Desired Output

	R (RED)	G (GREEN)	B (BLUE)
RED = -1	255	0	0
	248	80	68
BLUE = +1	0	0	255
	67	15	210

$$d(n) = \begin{cases} -1, x(n) \text{ belongs to } C1\ (RED) \\ +1, x(n) \text{ belongs to } C2\ (BLUE) \end{cases}$$

NN training steps

1. Weights Initialization
2. Inputs Application
3. Sum of Inputs-Weights Products
4. Activation Function Response Calculation
5. Weights Adaptation
6. Back to Step 2

Regarding 5th Step: Weights Adaptation

- If the predicted output Y is not the same as the desired output d, then weights are to be adapted according to the following equation:

$$W(n+1) = W(n) + \eta[d(n) - Y(n)]X(n)$$

Where

$$W(n) = [b(n), W_1(n), W_2(n), W_3(n), \ldots, W_m(n)]$$

Neural Networks Training Example Step n=0

	R (RED)	G (GREEN)	B (BLUE)
RED = -1	255	0	0
	248	80	68
BLUE = +1	0	0	255
	67	15	210

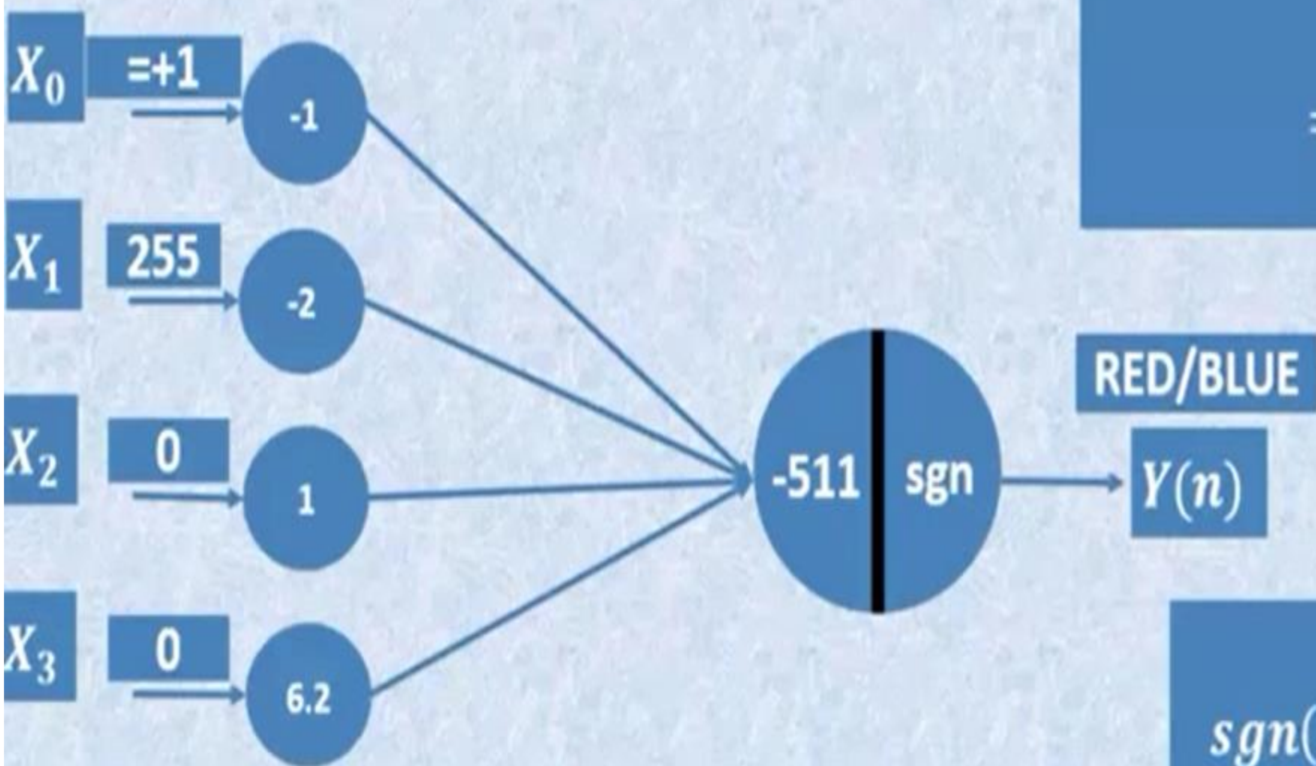
- In each step in the solution, the parameters of the neural network must be known.
- Parameters of step n=0:

$$\eta = .001$$
$$X(n) = X(0) = [X_0, X_1, X_2, X_3] = [+1, 255, 0, 0]$$
$$W(n) = W(0) = [w_0, w_1, w_2, w_3] = [-1, -2, 1, 6.2]$$
$$d(n) = d(0) = -1$$

Neural Networks Training Example Step n=0 - Output

	R (RED)	G (GREEN)	B (BLUE)
RED = -1	255	0	0
	248	80	68
BLUE = +1	0	0	255
	67	15	210

$$\begin{aligned} Y(n) &= Y(0) \\ &= SGN(s) \\ &= SGN(-511) \\ &= -1 \end{aligned}$$



$$sgn(s) = \begin{cases} +1, s \geq 0 \\ -1, s < 0 \end{cases}$$

Neural Networks Training Example Step n=1

	R (RED)	G (GREEN)	B (BLUE)
RED = -1	255	0	0
	248	80	68
BLUE = +1	0	0	255
	67	15	210

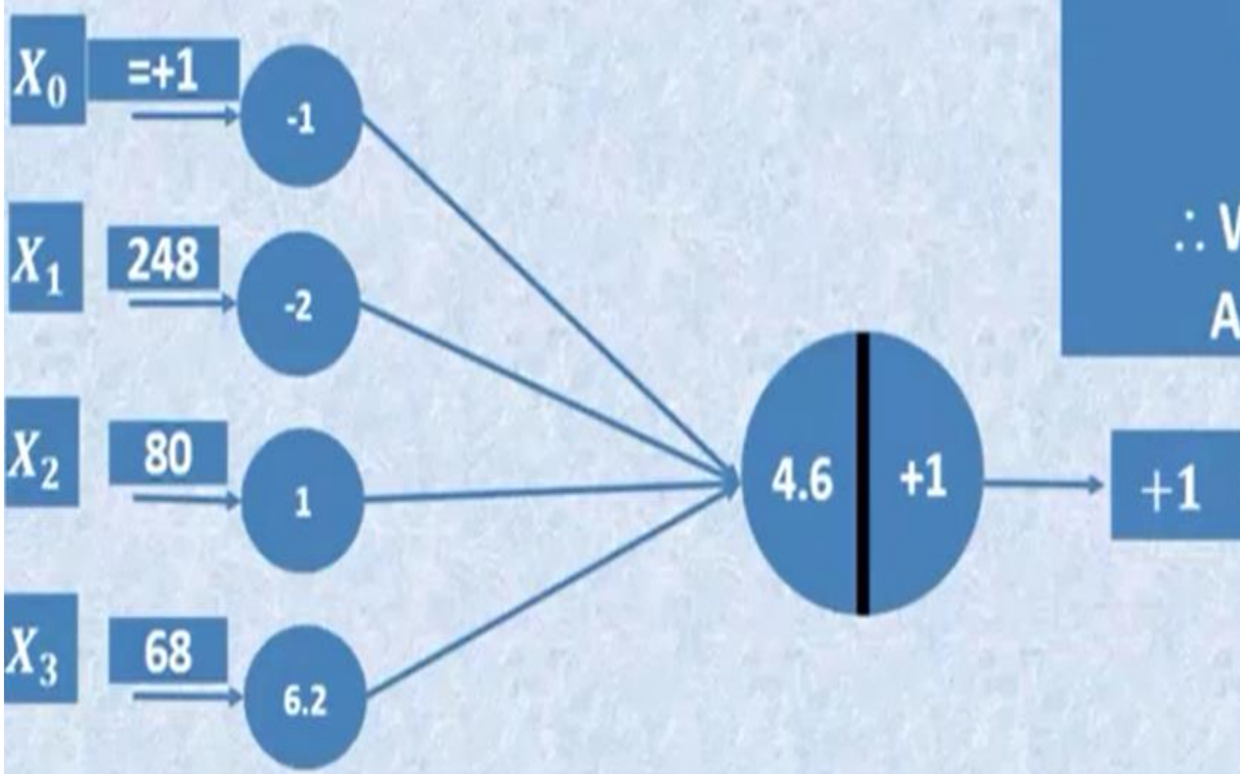
- In each step in the solution, the parameters of the neural network must be known.
- Parameters of step n=1:

$$\eta = .001$$
$$X(n) = X(1) = [+1, 248, 80, 68]$$
$$W(n) = W(1) = W(0) = [-1, -2, 1, 6.2]$$
$$d(n) = d(1) = -1$$

Neural Networks
Training Example
Step n=1
Predicted Vs. Desired

	R (RED)	G (GREEN)	B (BLUE)
RED = -1	255	0	0
	248	80	68
BLUE = +1	0	0	255
	67	15	210

$$Y(n) = Y(1) = +1$$
$$\mathrm{d}(n) = d(1) = -1$$
$$\because Y(n) \neq d(n)$$
$\therefore$ Weights are Incorrect.
Adaptation Required



Weights Adaptation

- According to

$$W(n+1) = W(n) + \eta[d(n) - Y(n)]X(n)$$

- Where n = 1

$$W(1+1) = W(1) + \eta[d(1) - Y(1)]X(1)$$
$$W(2) = [-1, -2, 1, 6.2] + .001[-1 - (+1)][+1, 248, 80, 68]$$
$$W(2) = [-1, -2, 1, 6.2] + .001[-2][+1, 248, 80, 68]$$
$$W(2) = [-1, -2, 1, 6.2] + [-.002][+1, 248, 80, 68]$$
$$W(2) = [-1, -2, 1, 6.2] + [-.002, -.496, -.16, -.136]$$
$$W(2) = [-1.002, -2.496, .84, 6.064]$$

Neural Networks Training Example Step n=2

	R (RED)	G (GREEN)	B (BLUE)
RED = -1	255	0	0
	248	80	68
BLUE = +1	0	0	255
	67	15	210

- In each step in the solution, the parameters of the neural network must be known.
- Parameters of step n=2:

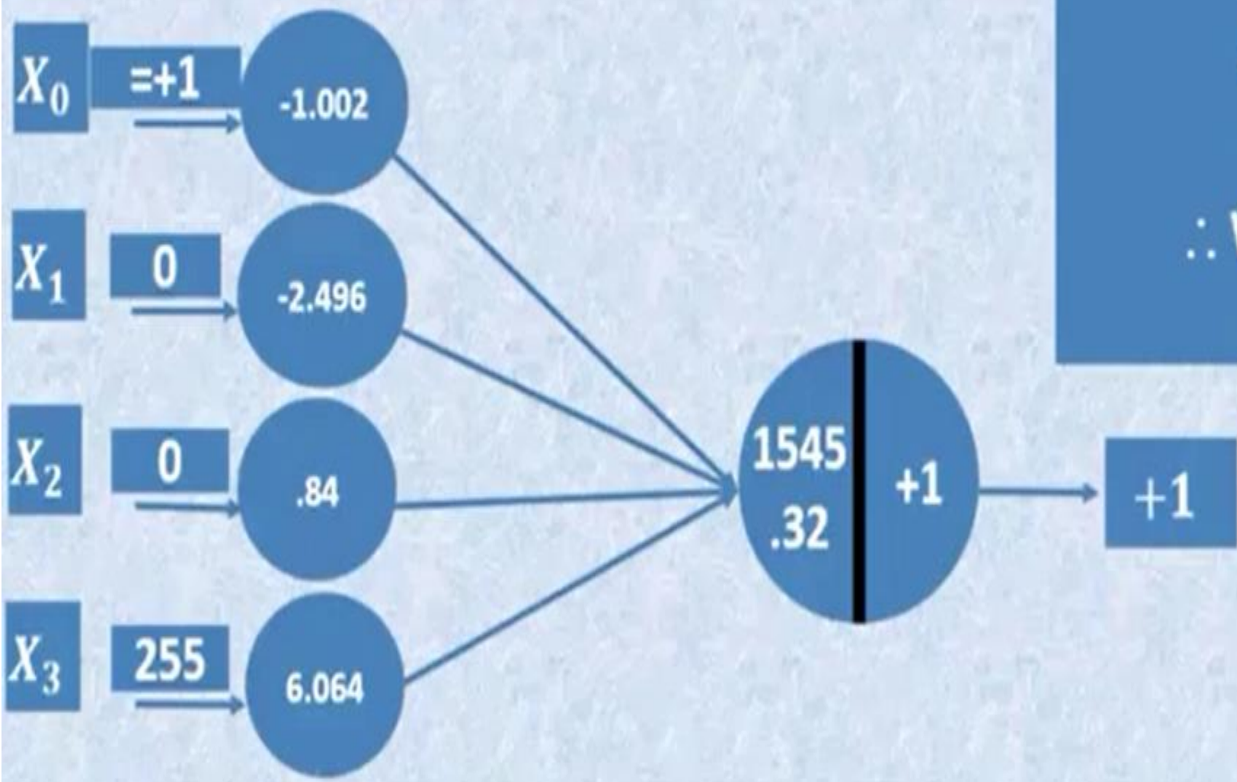
$$\eta = .001$$
$$X(n) = X(2) = [+1, 0, 0, 255]$$
$$W(n) = W(2) = [-1.002, -2.496, .84, 6.064]$$
$$d(n) = d(2) = +1$$

Neural Networks
Training Example
Step n=2
Predicted Vs. Desired

	R (RED)	G (GREEN)	B (BLUE)
RED = -1	255	0	0
	248	80	68
BLUE = +1	0	0	255
	67	15	210

$$Y(n) = Y(2) = +1$$
$$\mathrm{d}(n) = d(2) = +1$$
$$\because Y(n) = d(n)$$
$\therefore$ Weights are Correct.

No Adaptation



Neural Networks Training Example Step n=3

	R (RED)	G (GREEN)	B (BLUE)
RED = -1	255	0	0
	248	80	68
BLUE = +1	0	0	255
	67	15	210

- In each step in the solution, the parameters of the neural network must be known.
- Parameters of step n=3:

$$\eta = .001$$
$$X(n) = X(3) = [+1, 67, 15, 210]$$
$$W(n) = W(3) = W(2) = [-1.002, -2.496, .84, 6.064]$$
$$d(n) = d(3) = +1$$

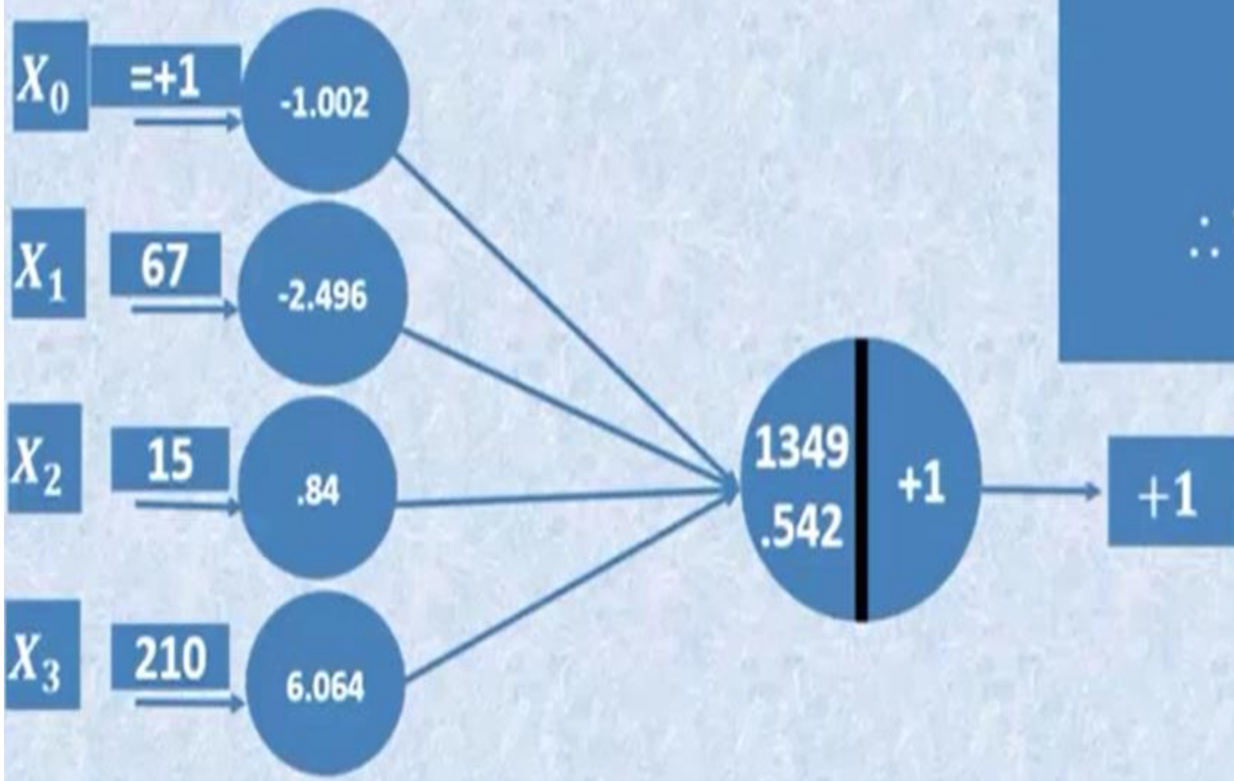
Neural Networks
Training Example
Step n=3
Predicted Vs. Desired

	R (RED)	G (GREEN)	B (BLUE)
RED = -1	255	0	0
	248	80	68
BLUE = +1	0	0	255
	67	15	210

$$Y(n) = Y(3) = +1$$
$$\mathrm{d}(n) = d(3) = +1$$
$$\because Y(n) = d(n)$$

$\therefore$ Weights are Correct.

No Adaptation



Neural Networks
Training Example
Step n=4

RED = -1	255	0	0
	248	80	68
BLUE = +1	0	0	255
	67	15	210

- In each step in the solution, the parameters of the neural network must be known.
- Parameters of step n=4:

$$\eta = .001$$
$$X(n) = X(4) = [+1, 255, 0, 0]$$
$$W(n) = W(4) = W(3) = [-1.002, -2.496, .84, 6.064]$$
$$d(n) = d(4) = -1$$

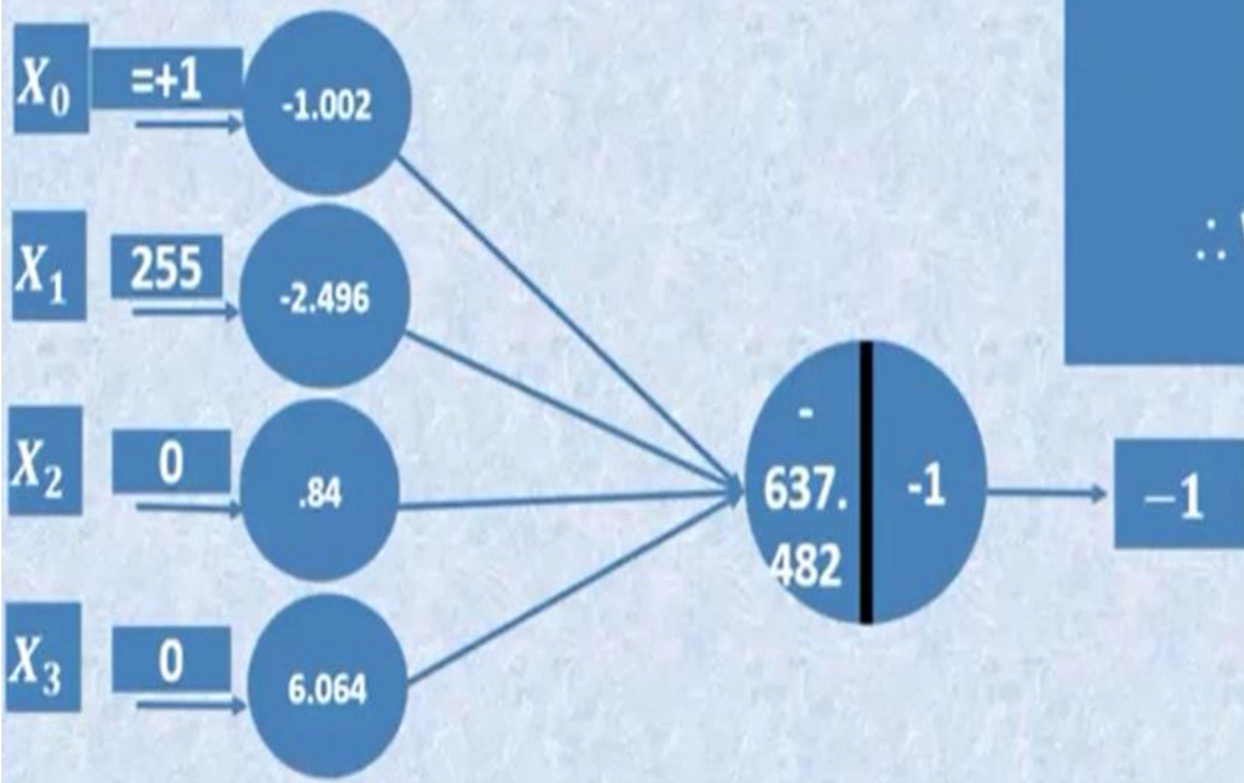
Neural Networks
Training Example
Step n=4
Predicted Vs. Desired

	R (RED)	G (GREEN)	B (BLUE)
RED = -1	255	0	0
	248	80	68
BLUE = +1	0	0	255
	67	15	210

$$Y(n) = Y(4) = -1$$
$$d(n) = d(4) = -1$$
$$\because Y(n) = d(n)$$

$\therefore$ Weights are Correct.

No Adaptation



Neural Networks Training Example Step n=5

	R (RED)	G (GREEN)	B (BLUE)
RED = -1	255	0	0
	248	80	68
BLUE = +1	0	0	255
	67	15	210

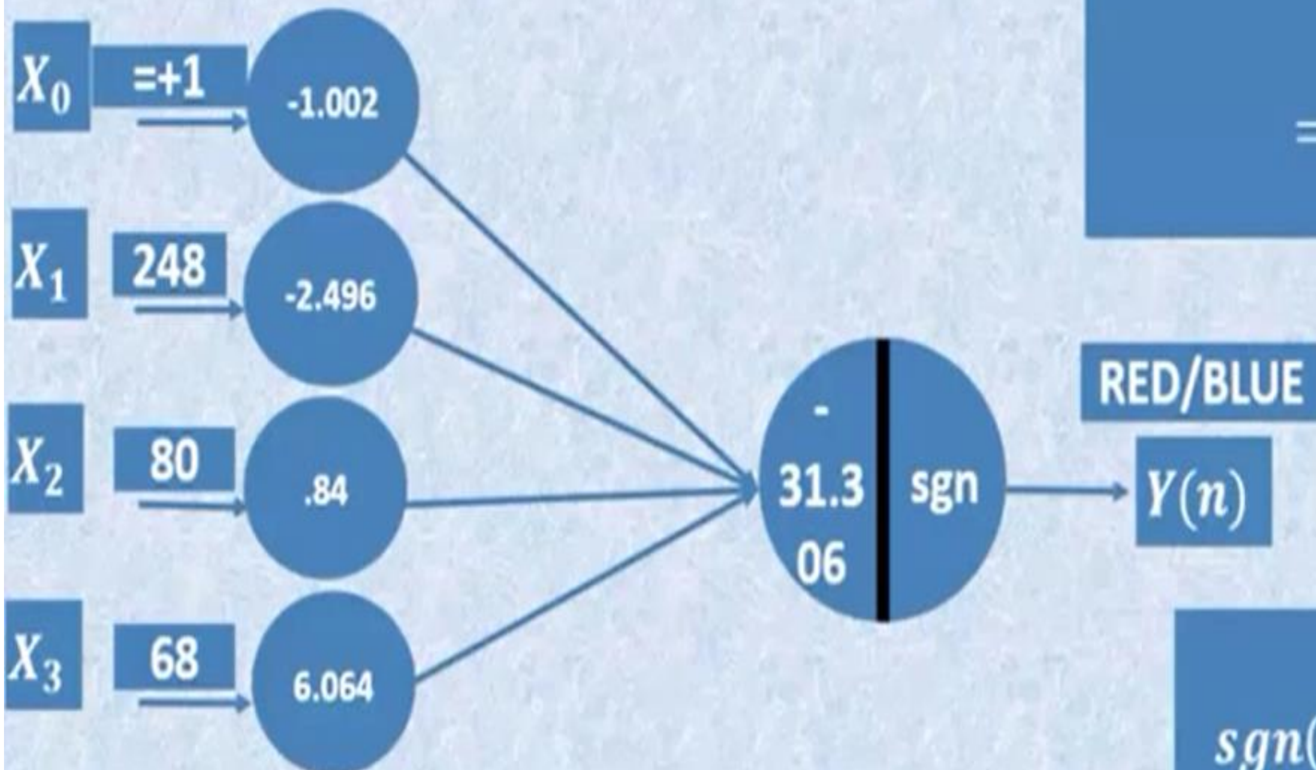
- In each step in the solution, the parameters of the neural network must be known.
- Parameters of step n=5:

$$\eta = .001$$
$$X(n) = X(5) = [+1, 248, 80, 68]$$
$$W(n) = W(5) = W(4) = [-1.002, -2.496, .84, 6.064]$$
$$d(n) = d(5) = -1$$

Neural Networks Training Example Step n=5 - Output

	R (RED)	G (GREEN)	B (BLUE)
RED = -1	255	0	0
	248	80	68
BLUE = +1	0	0	255
	67	15	210

$$Y(n) = Y(5)$$
$$= SGN(s)$$
$$= SGN(-31.306)$$
$$= -1$$



$$sgn(s) = \begin{cases} +1, s \geq 0 \\ -1, s < 0 \end{cases}$$

Generalization
Test case

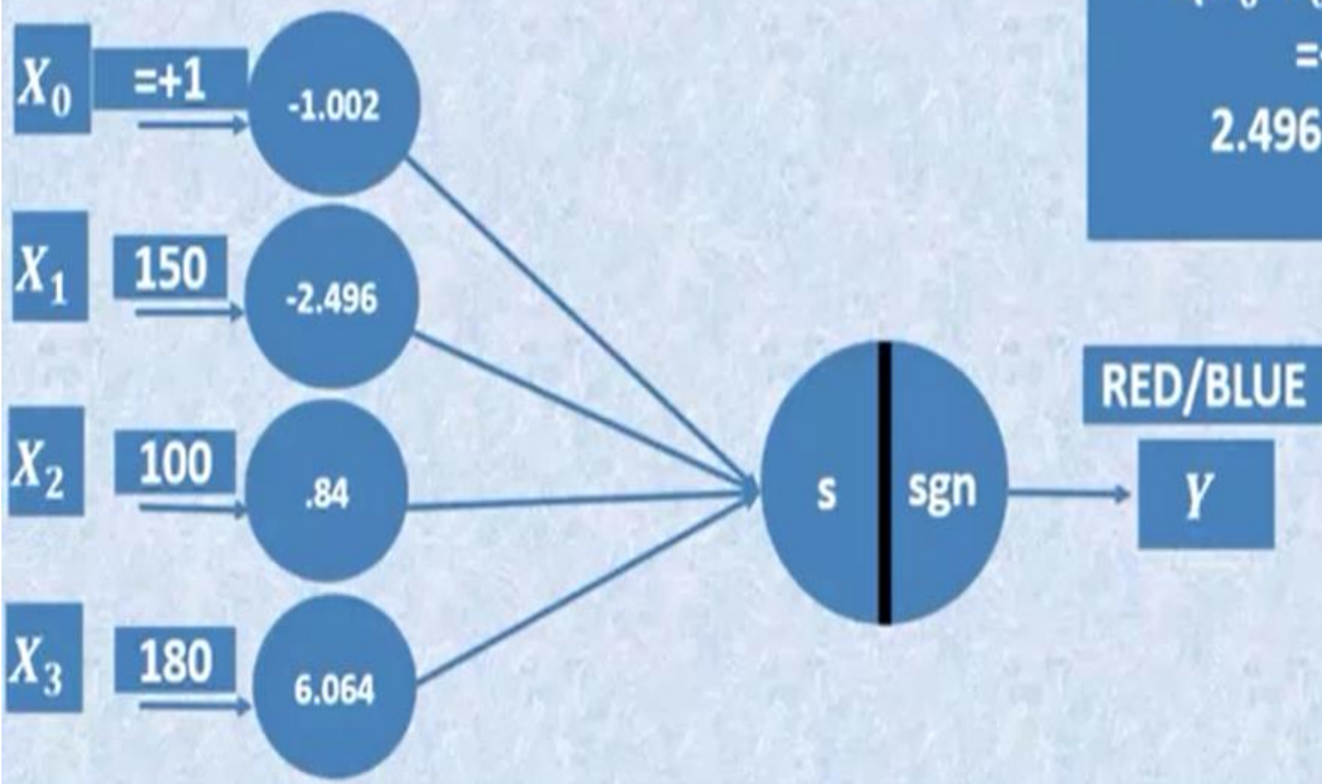
Correct Weights

- After testing the weights across all samples and results were correct then we can conclude that current weights are correct ones for training the neural network.
- After training phase, we come to predicting the class label of an unknown sample.
- What is the class of the unknown color of values of **R=150, G=100, B=180**?

Trained Neural Network

(R, G, B) = (150, 100, 180)

SOP



$s=(X_0W_0+X_1W_1+X_2W_2+X_3W_3)$

=+1*-1.002+150*-2.496+100*.84+180*6.064

=800.118