Neural Network

Lecture 4

Introduction

Superstar Researchers

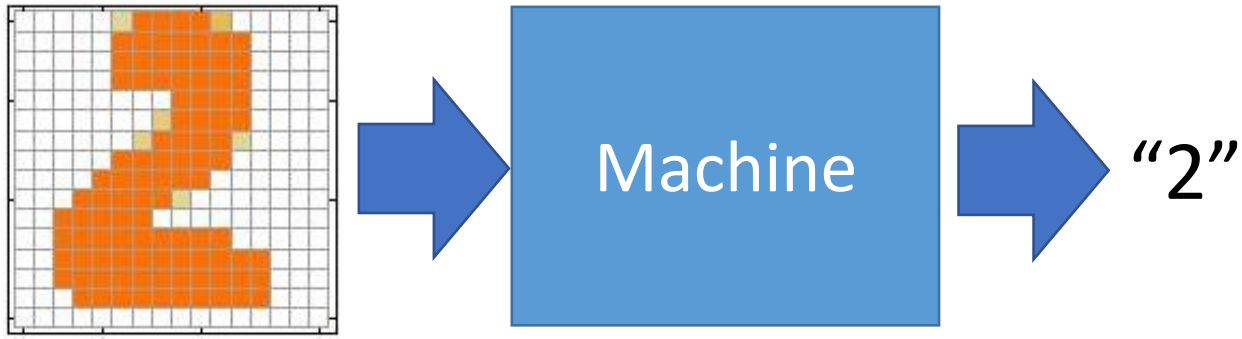
I have worked all my life in machine learning, and **I've never seen one algorithm knock over benchmarks like deep learning .**
Andrew Ng(Stanford & Baidu)

Deep learning is an algorithm which has no theoretical limitations of what it can learn ;**the more data you give and the more computational time you provide ;the better it** is-Geoffrey Hilton(Google)

Human-level artificial intelligence has the potential to help humanity thrive more than any invention that has come before it –
Dileep George(Co-Founder Vicarious)

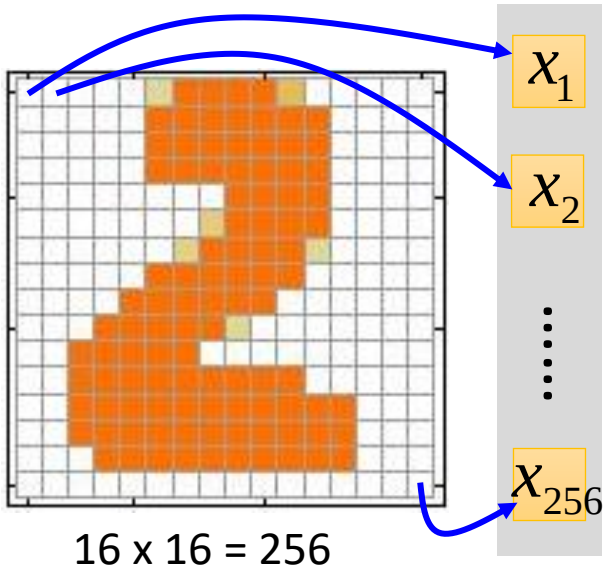
Example Application

- Handwriting Digit Recognition



Handwriting Digit Recognition

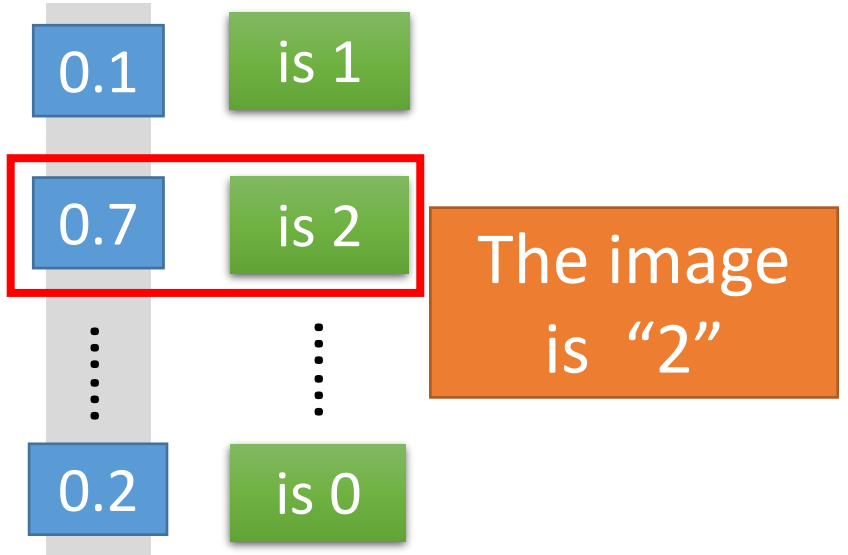
Input



Ink → 1

No ink → 0

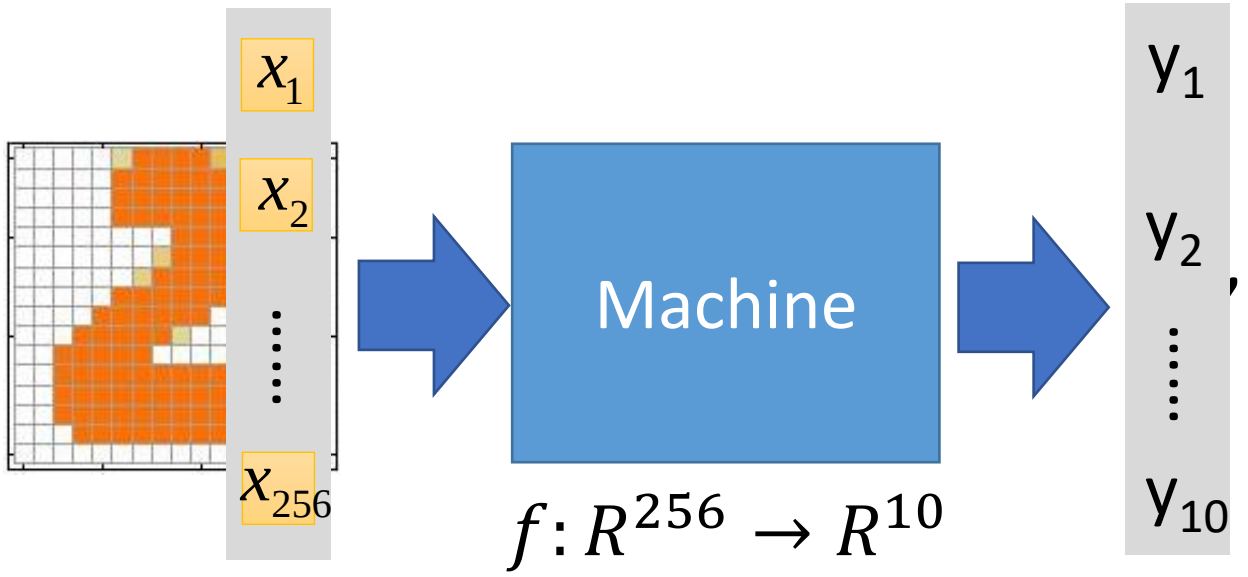
Output



Each dimension represents the confidence of a digit.

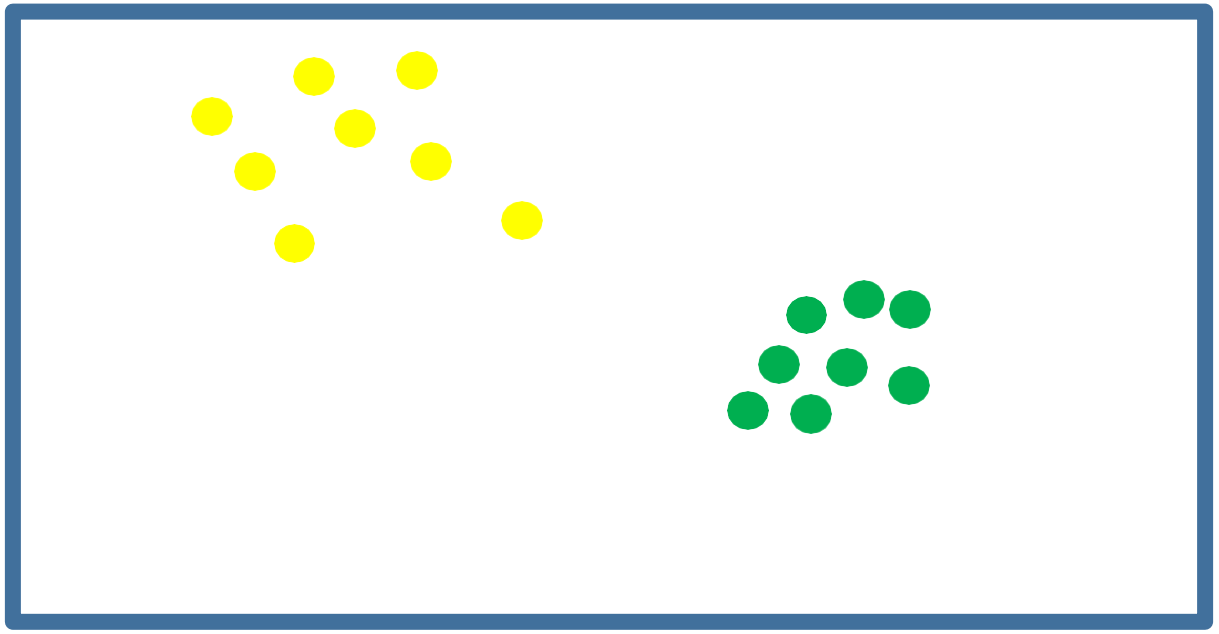
Example Application

- Handwriting Digit Recognition

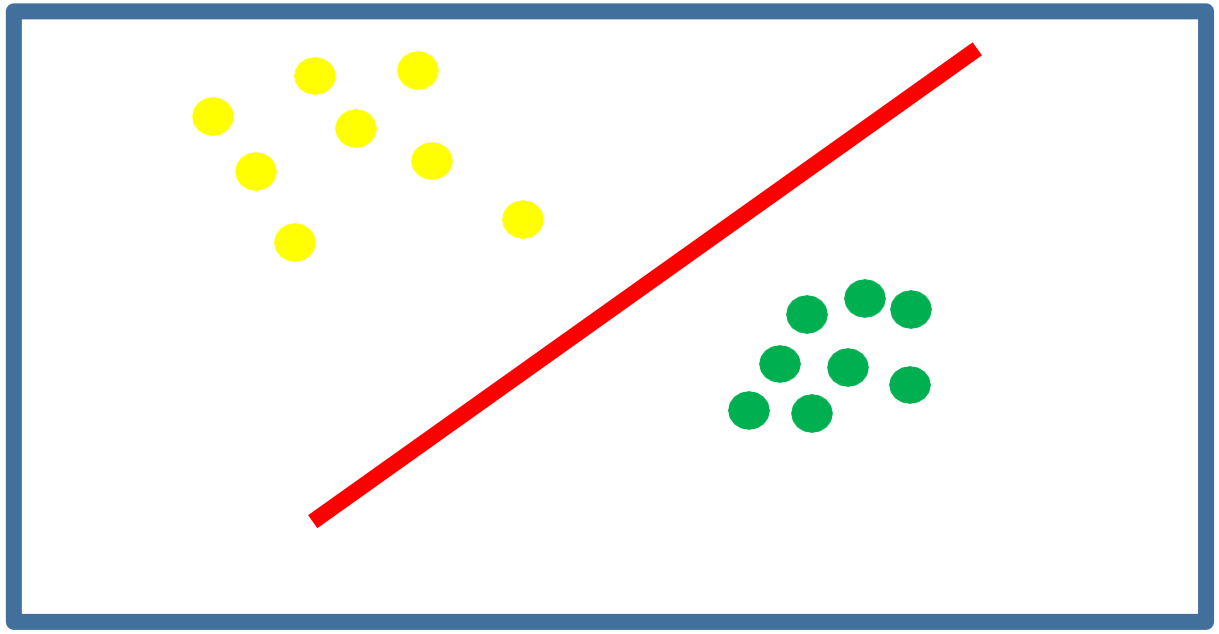


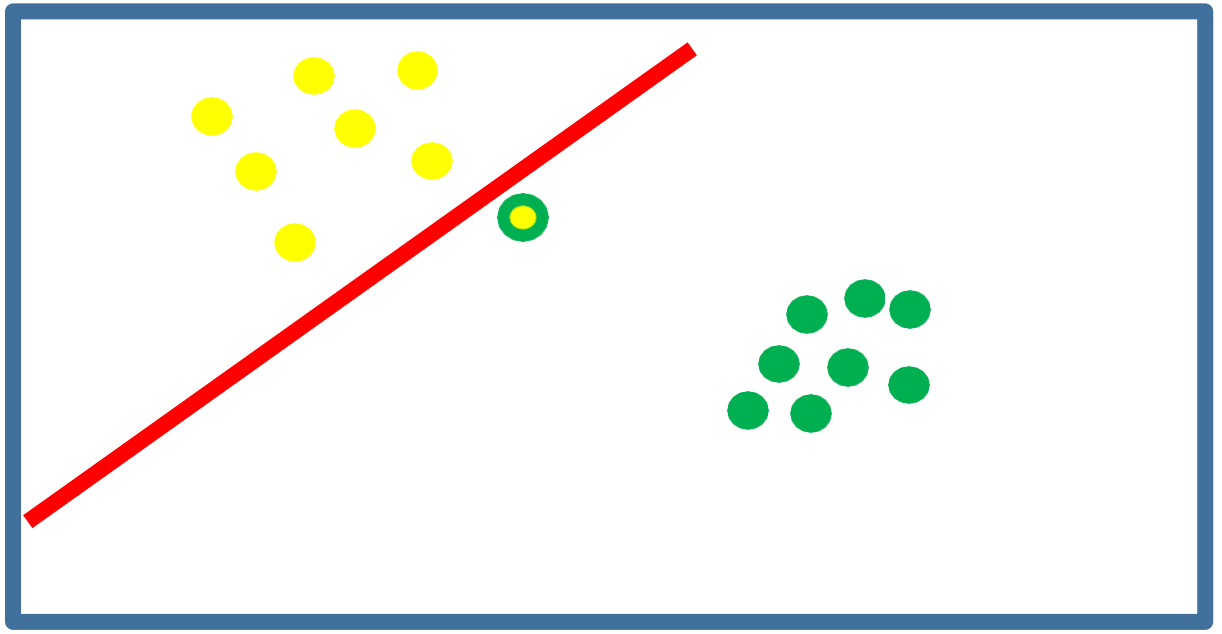
In deep learning, the function f is represented by neural network

Classification problem

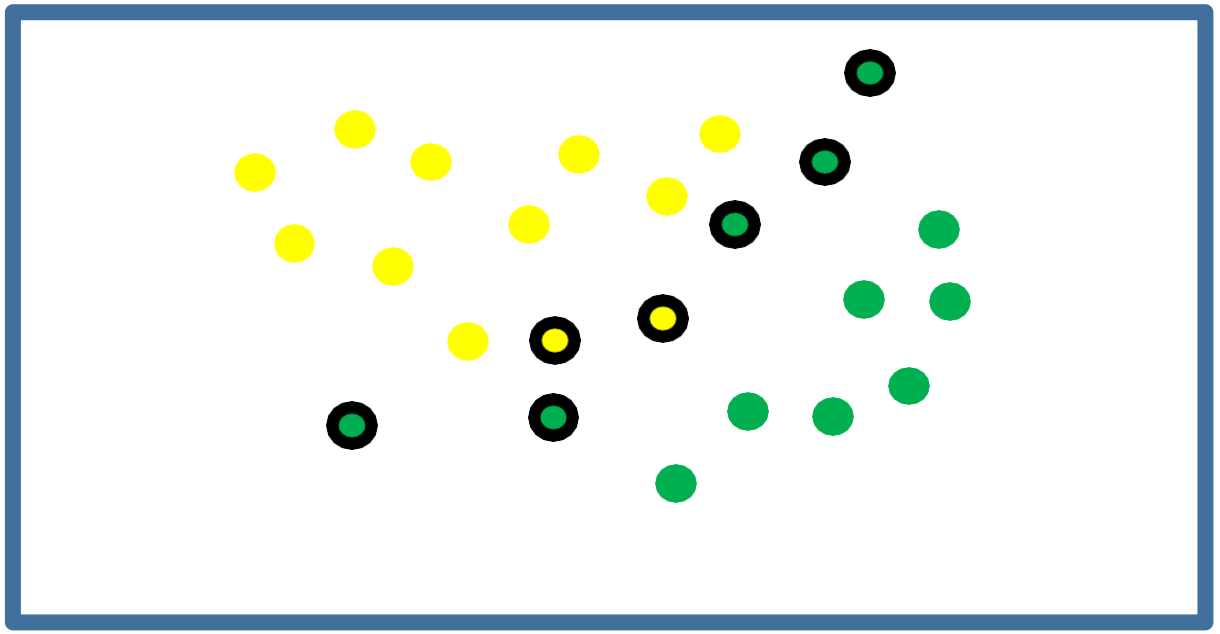


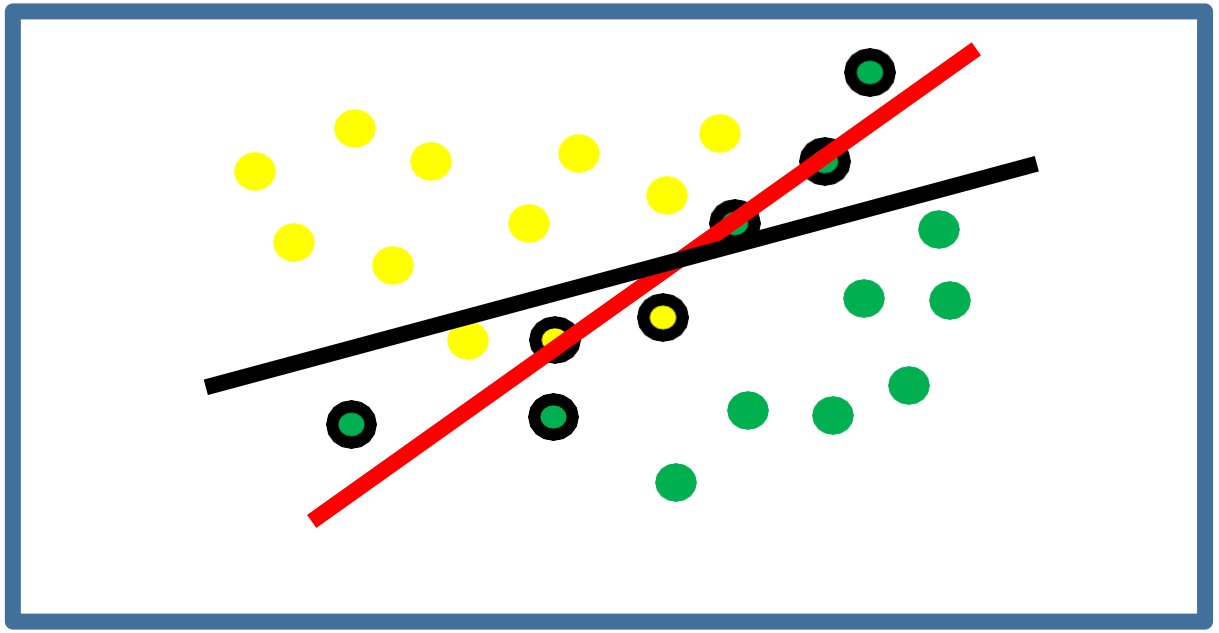
Linear Classifiers



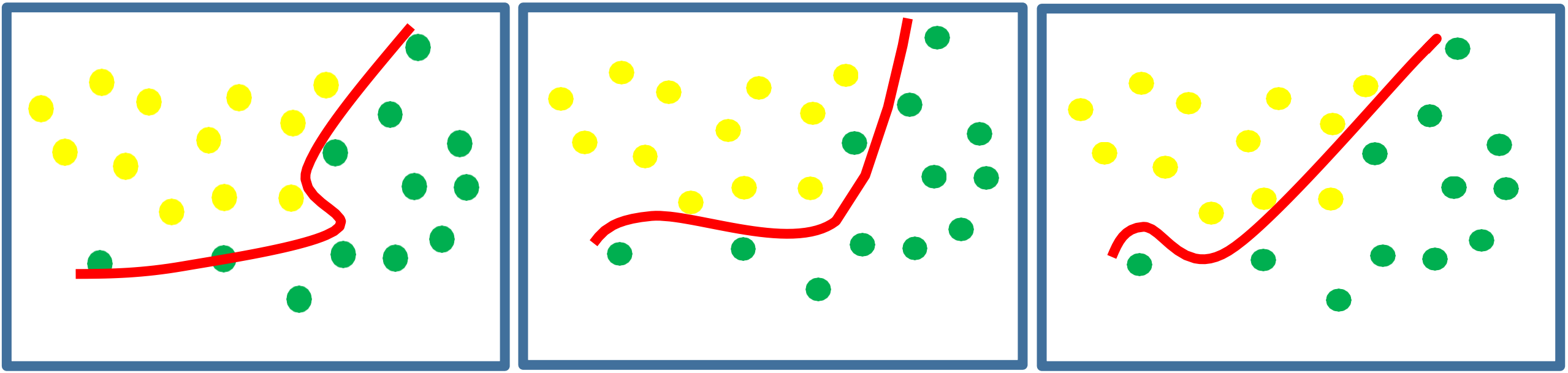


Not Solved Linearly





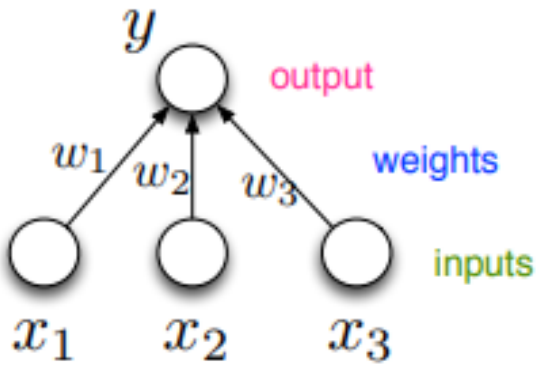
Nonlinear Classifiers

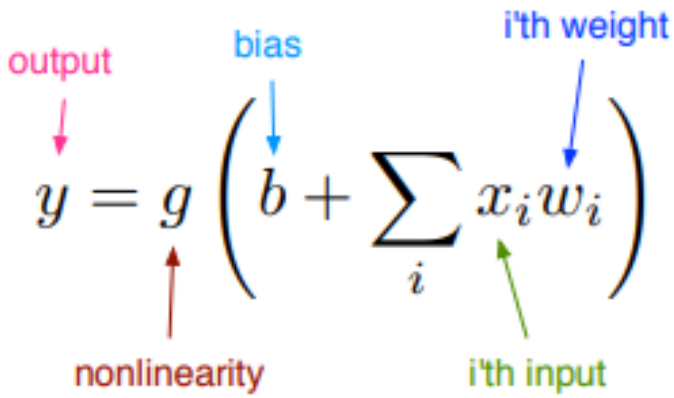


Neural Network

Neural Network: A Neuron

- Neural networks define functions of the inputs (hidden features), computed by neurons
- A neuron (called unit) is a computational unit in the neural network that exchanges messages with each other.





Neural Network Architecture

- Network with one layer of four hidden units:.

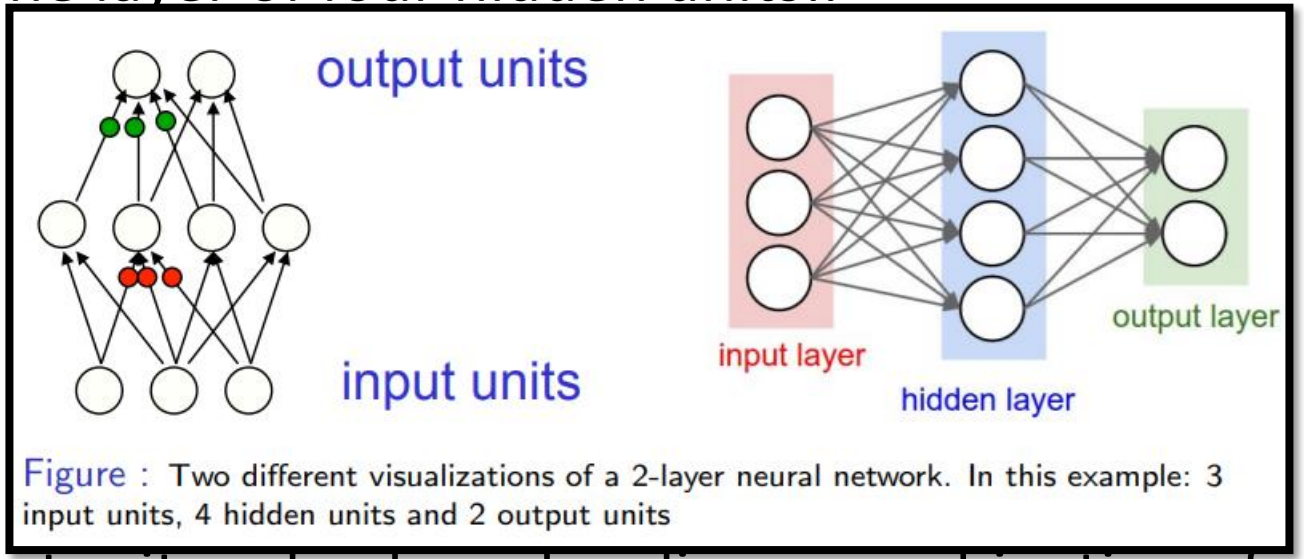
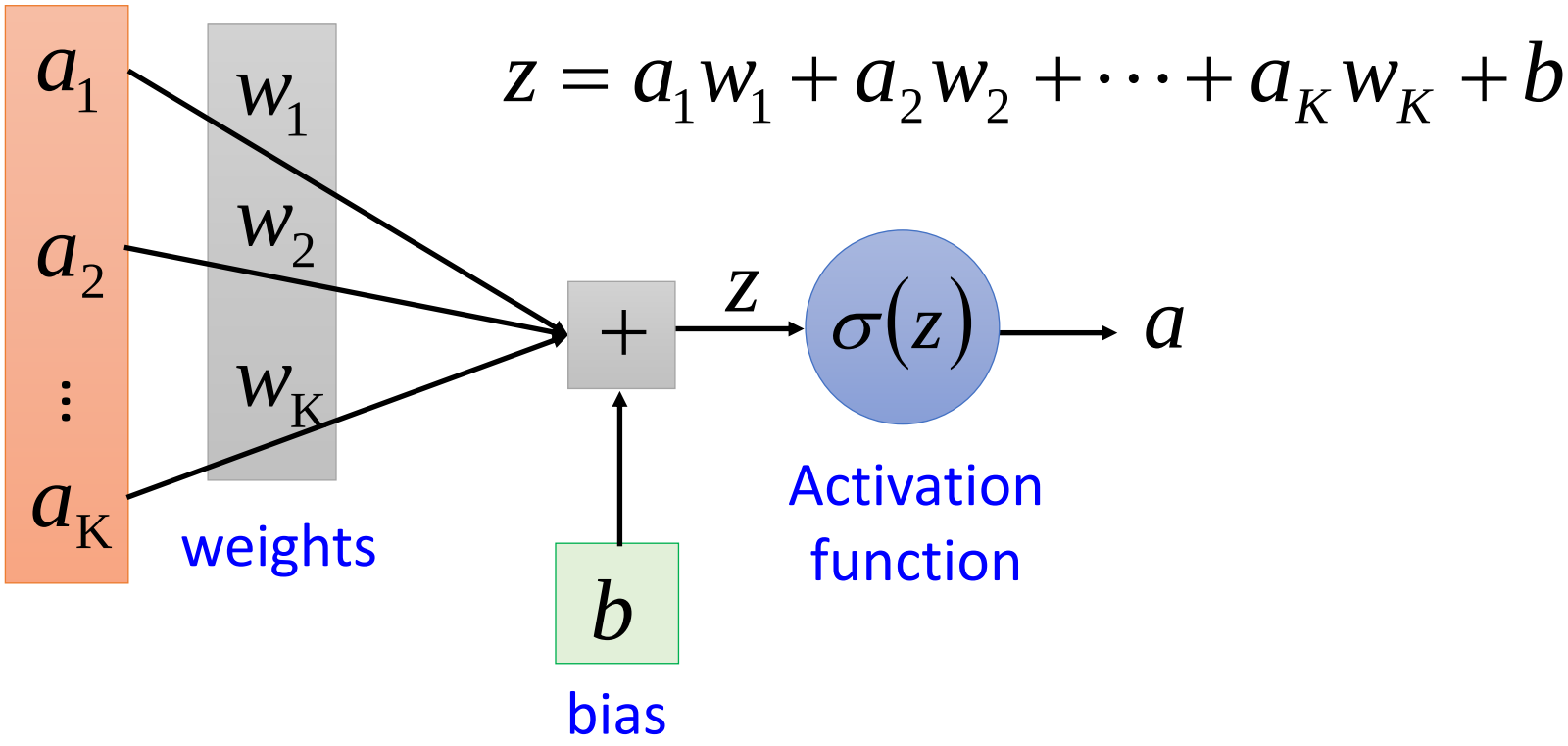


Figure : Two different visualizations of a 2-layer neural network. In this example: 3 input units, 4 hidden units and 2 output units

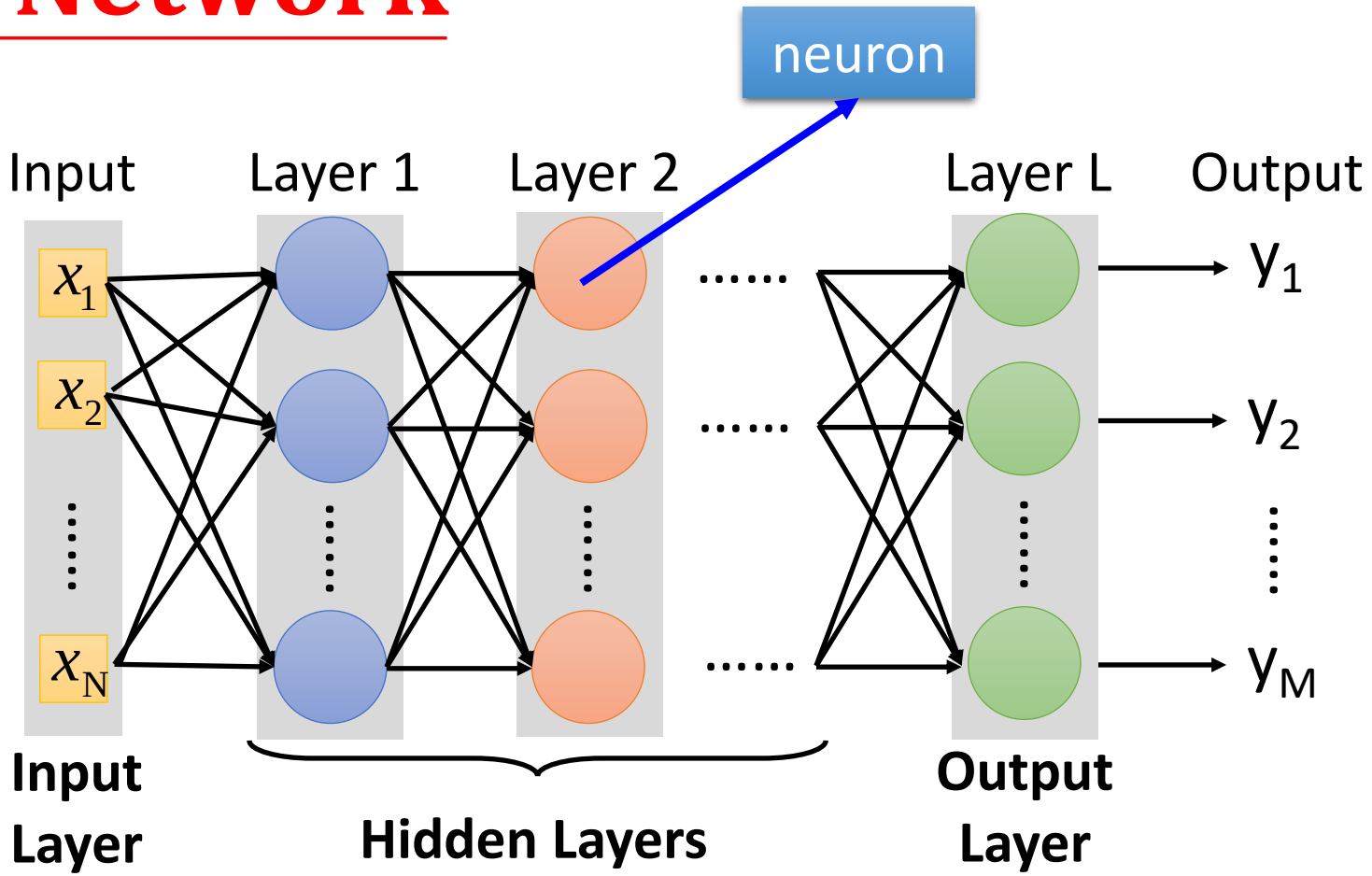
- Each unit computes its value based on linear combination of values of units that point into it, and an activation function.
- Naming conventions; a N-layer neural network:
  - N-1 layer of hidden units
  - One output layer (we do not count the inputs as a layer).
- In this figure: 2-layer neural network

Element of Neural Network

Neuron $f: R^K \rightarrow R$



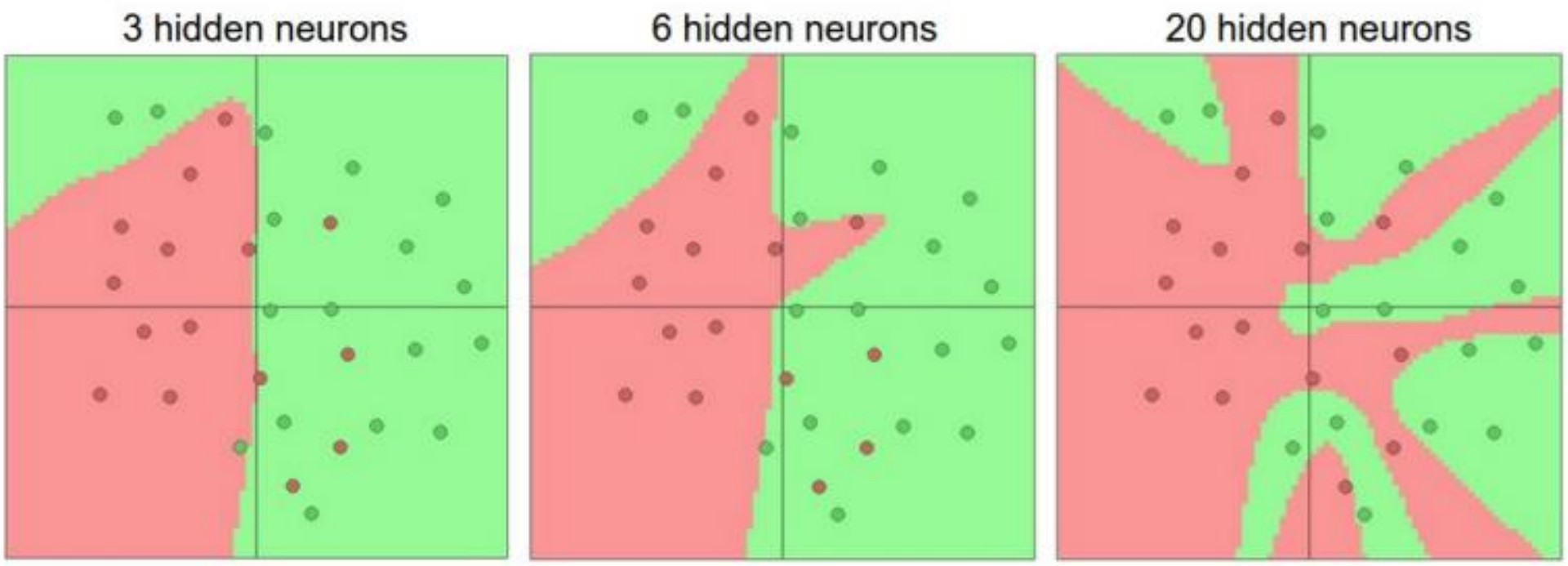
Neural Network

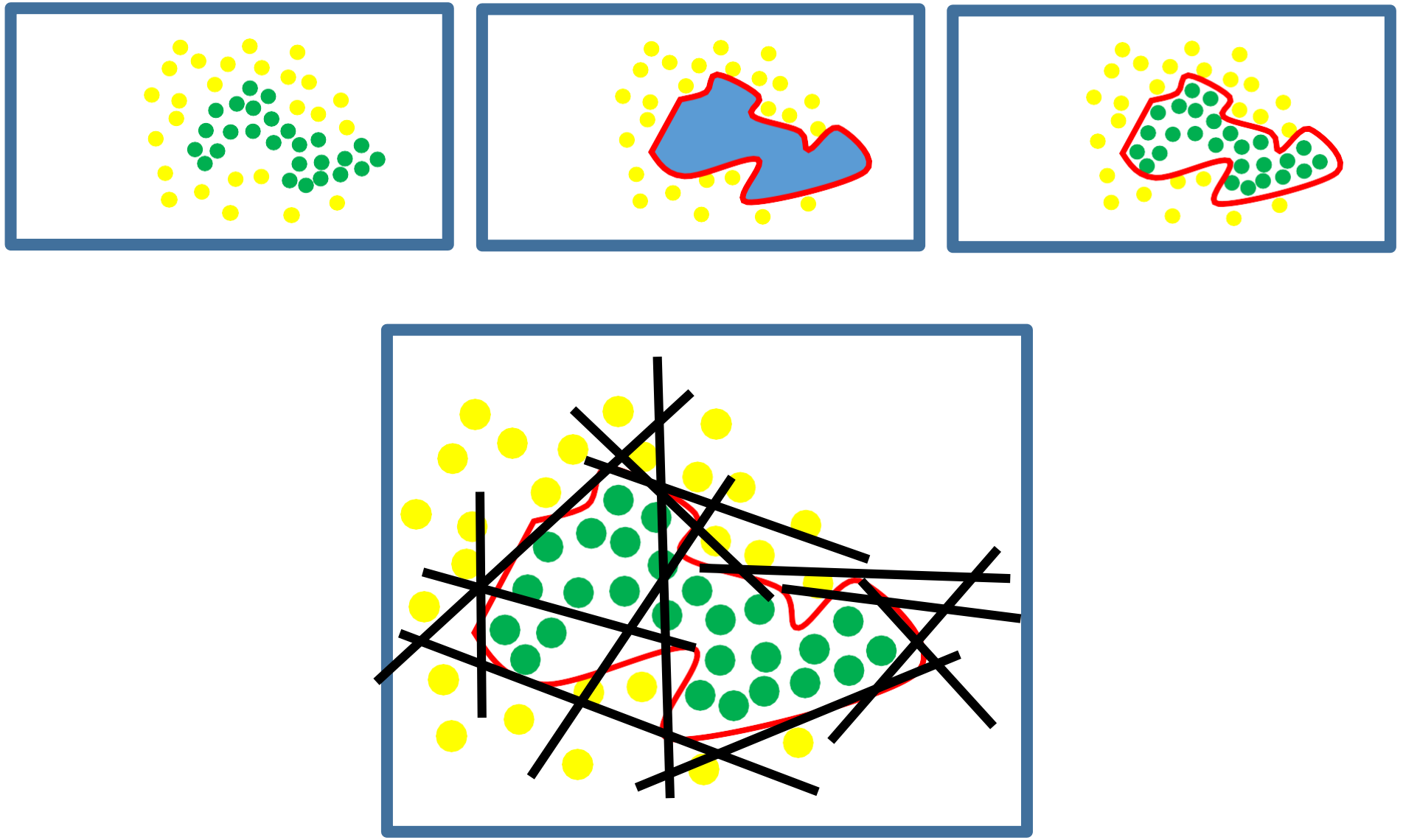


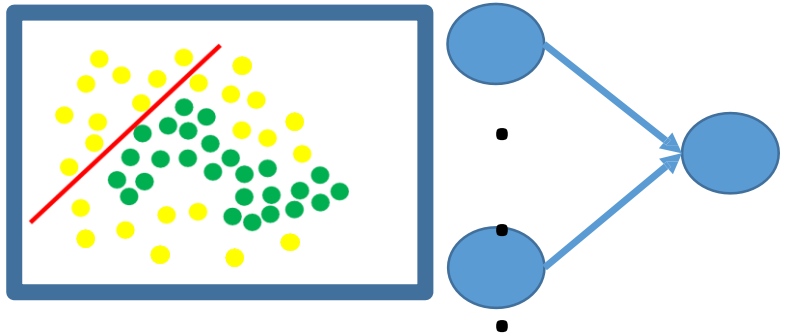
Deep means many hidden layers

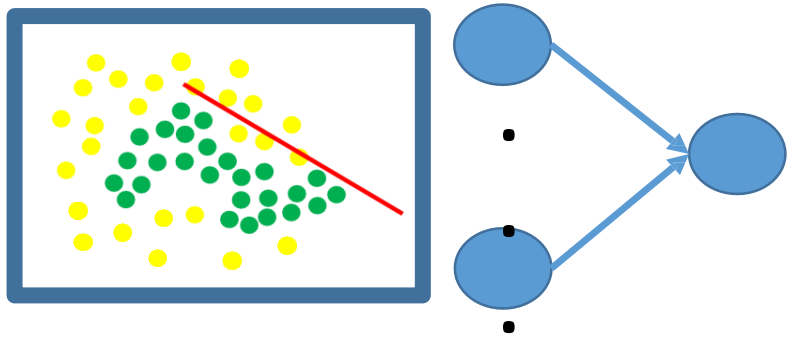
Network capacity

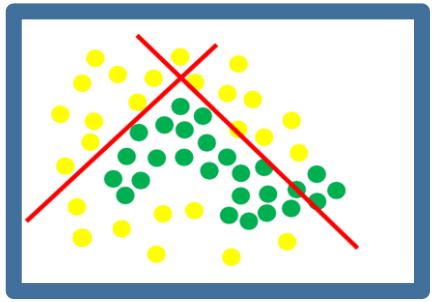
The capacity of the network increases with more hidden units and more hidden layers

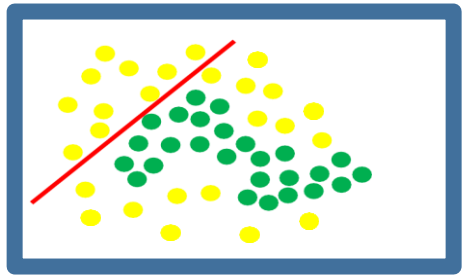


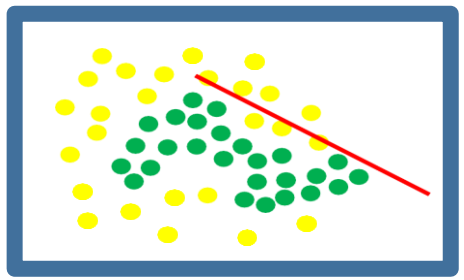


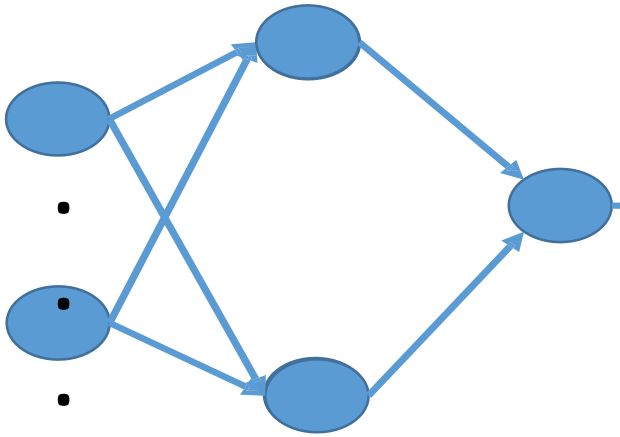


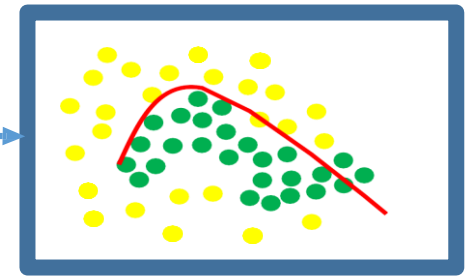


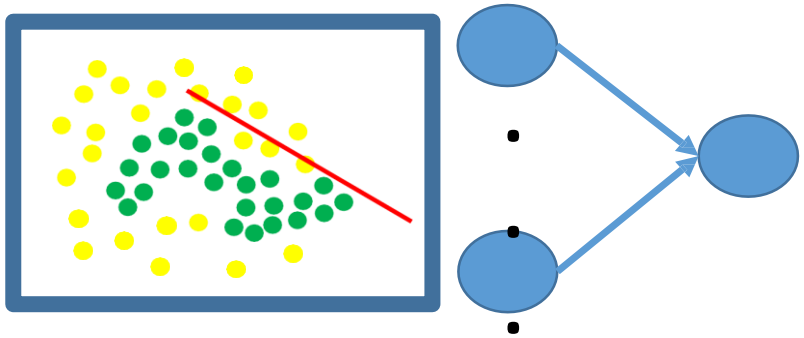


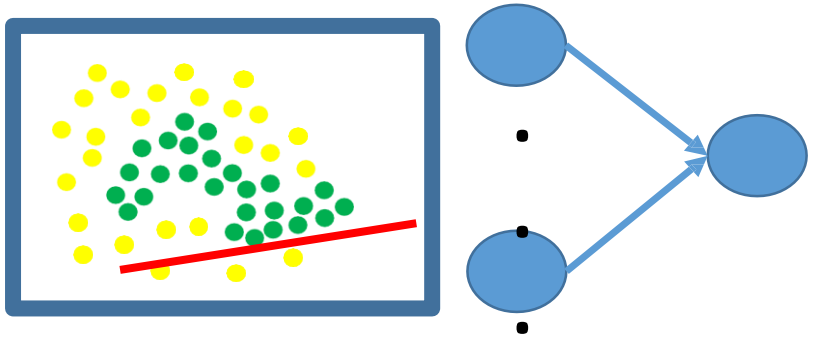


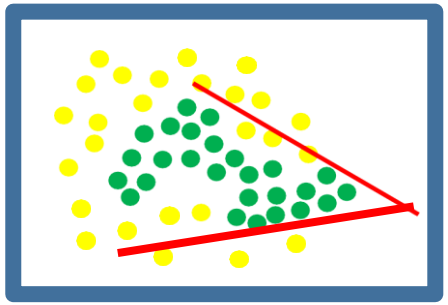


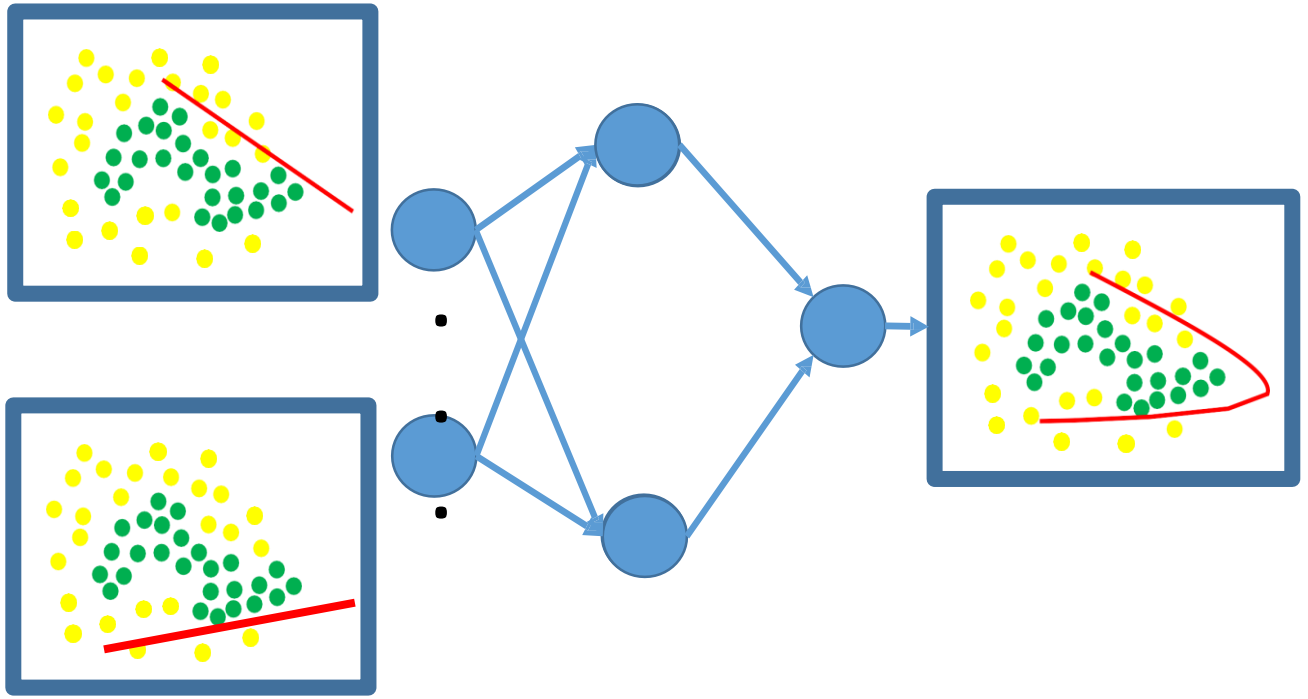


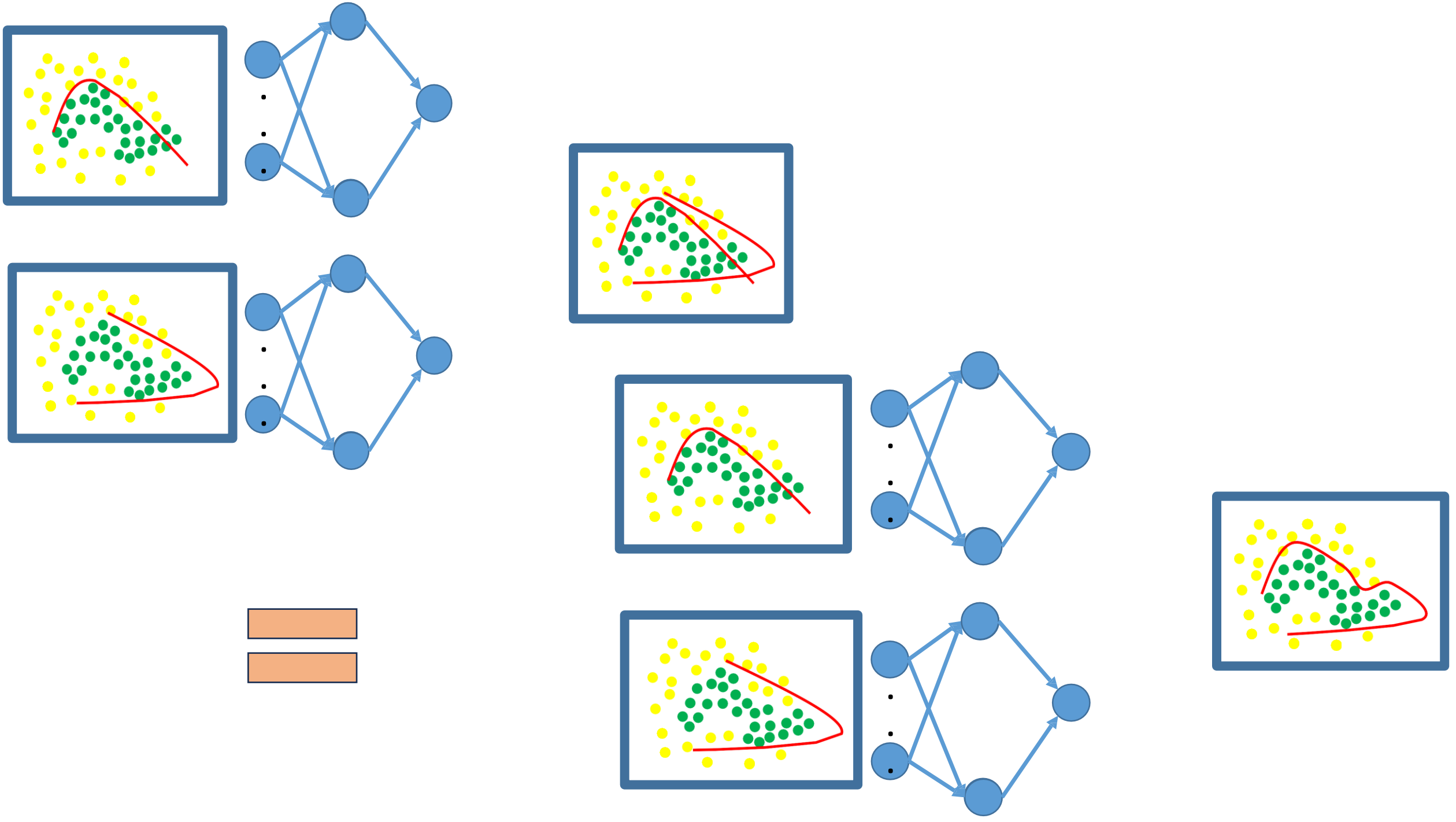


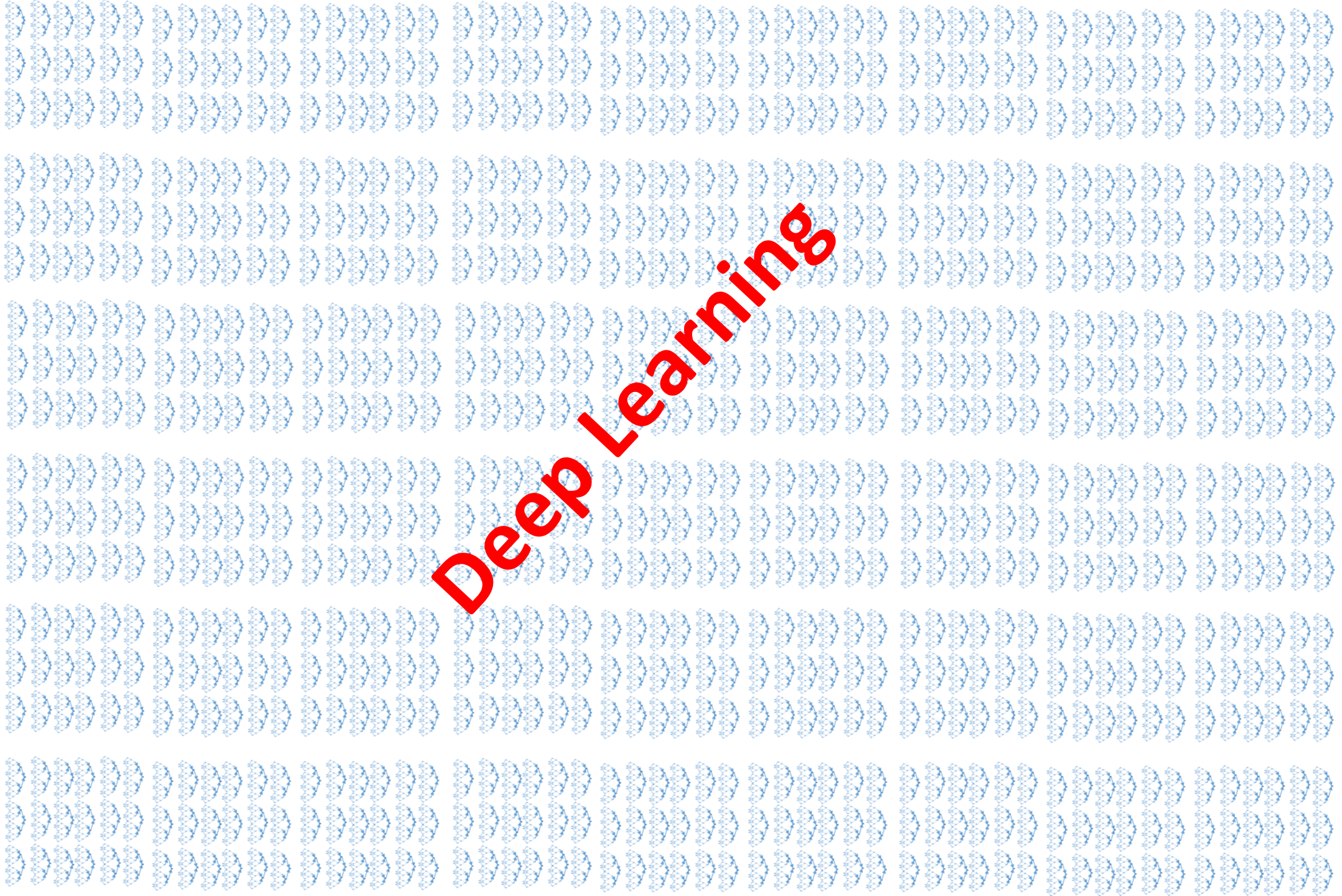











Deep Learning

Perceptron

A perceptron is a neural network unit (an artificial neuron) that does certain computations to detect features or business intelligence in the input data.

Each perceptron comprises different parts:

1. **Input**: A set of values or a dataset for predicting the output value. They are also described as a dataset's features and dataset.
2. **Weights:** The real value of each feature is known as weight. It tells the importance of that feature in predicting the final value.
3. **Bias:** The activation function is shifted towards the left or right using bias. You may understand it simply as the y-intercept in the line equation.
4. **Summation Function:** The summation function binds the weights and inputs together. It is a function to find their sum.
5. **Activation Function:** The activation function determines the output of the perceptron based on the weighted sum of the inputs and the bias term.
6. **Learning Algorithm (Weight Update Rule):** During training, the perceptron learns by adjusting its weights and bias based on a learning algorithm. A common approach is the perceptron learning algorithm, which updates weights based on the difference between the predicted output and the true output
7. **Output:** The output of the perceptron is a single binary value, either 0 or 1, which indicates the class or category to which the input data belongs.

Types of Perceptron

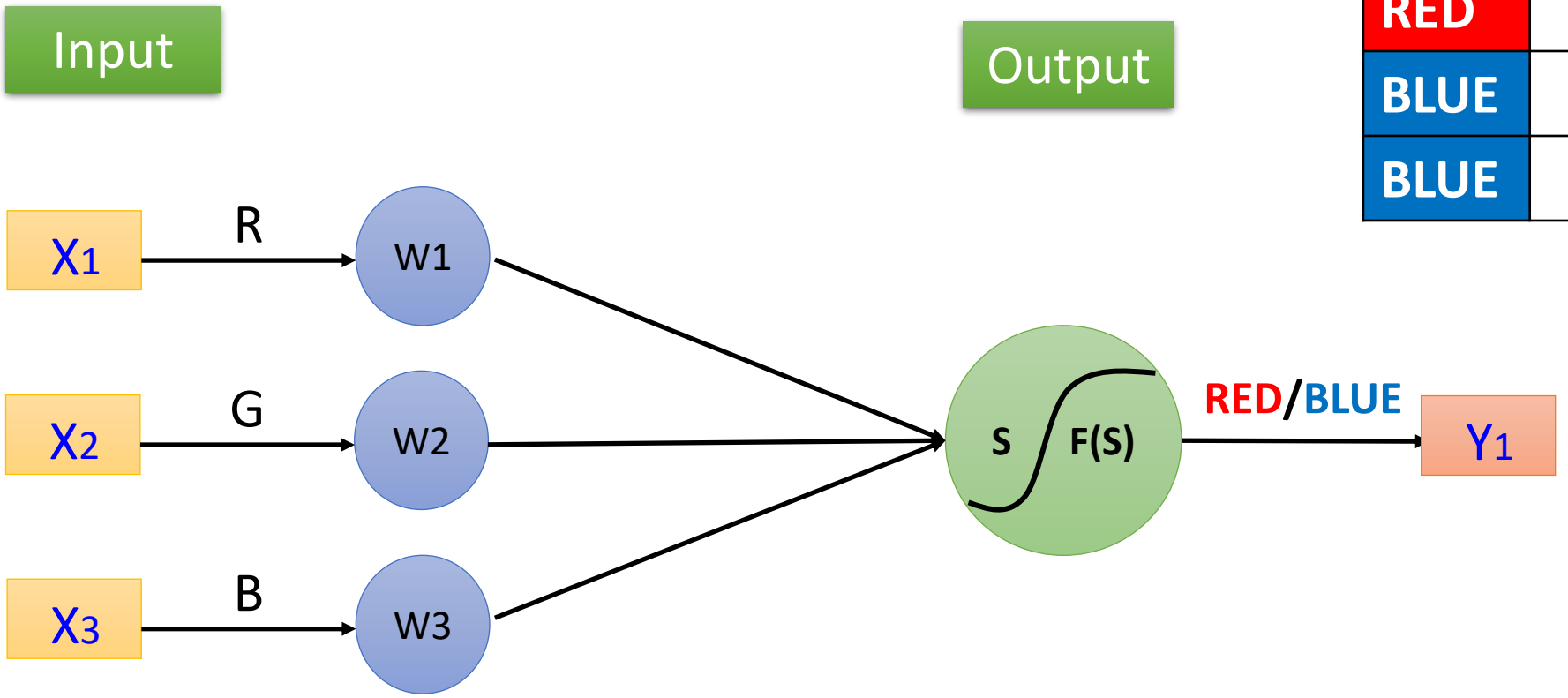
- **Types of Perceptron:**
  - **Single-Layer Perceptron:** This type of perceptron is limited to learning linearly separable patterns.
  - **Multilayer Perceptron:** Multilayer perceptrons possess enhanced processing capabilities as they consist of three (input, output, and at least one hidden layer) or more layers, adept at handling more complex patterns and relationships within the data.

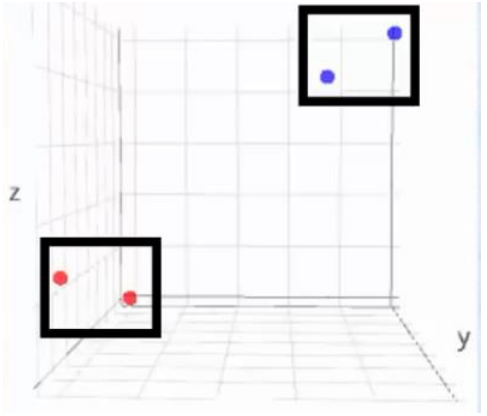
- The multi-layer perceptron model has two stages as follows:
  - **Forward Stage:** Activation functions start from the input layer in the forward stage and terminate on the output layer.
  - **Backward Stage:** In the backward stage, weight and bias values are modified as per the model's requirement. In this stage, the error between actual output and demanded originated backward on the output layer and ended on the input layer.

Example of Single-layer

Example: Color

	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210





x_i=Inputs w_i=Weights $S = \sum_1^m x_i\, w_i = (x_1 w_1 + x_2 w_2 + x_3 w_3)$

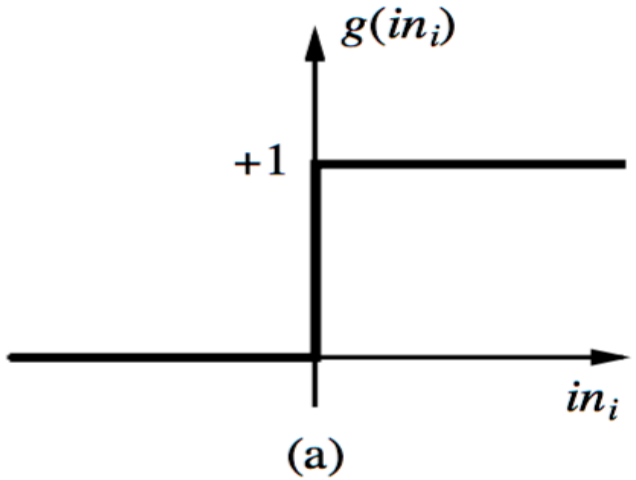
Note: Using hidden layer depends on linearity, if data classified with non-linear model so we use hidden layer, otherwise we don't use hidden layer.

Activation Function

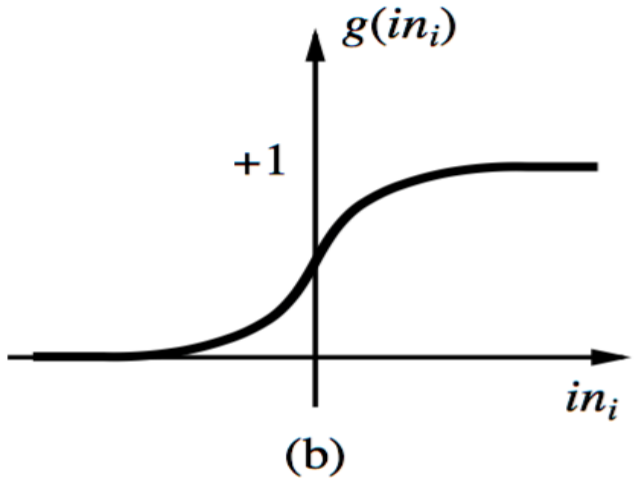
Input

Output	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

Possible activation functions:



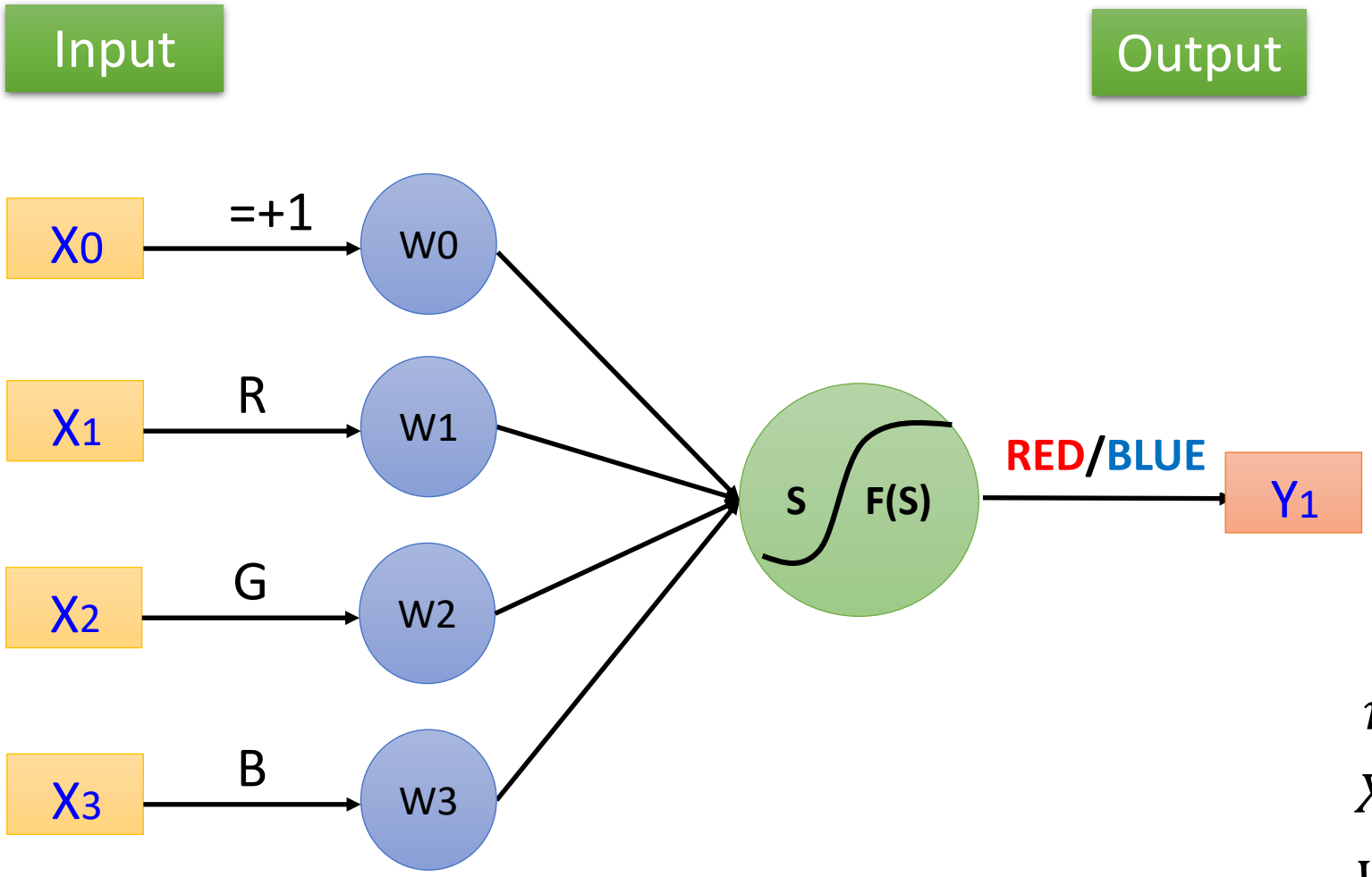
Step function/threshold function



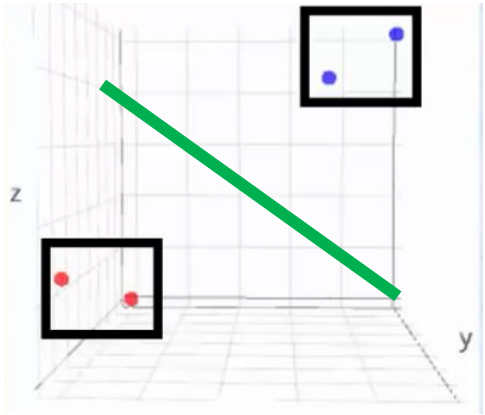
Sigmoid function (logistic function)

2 Output

Bias



	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210



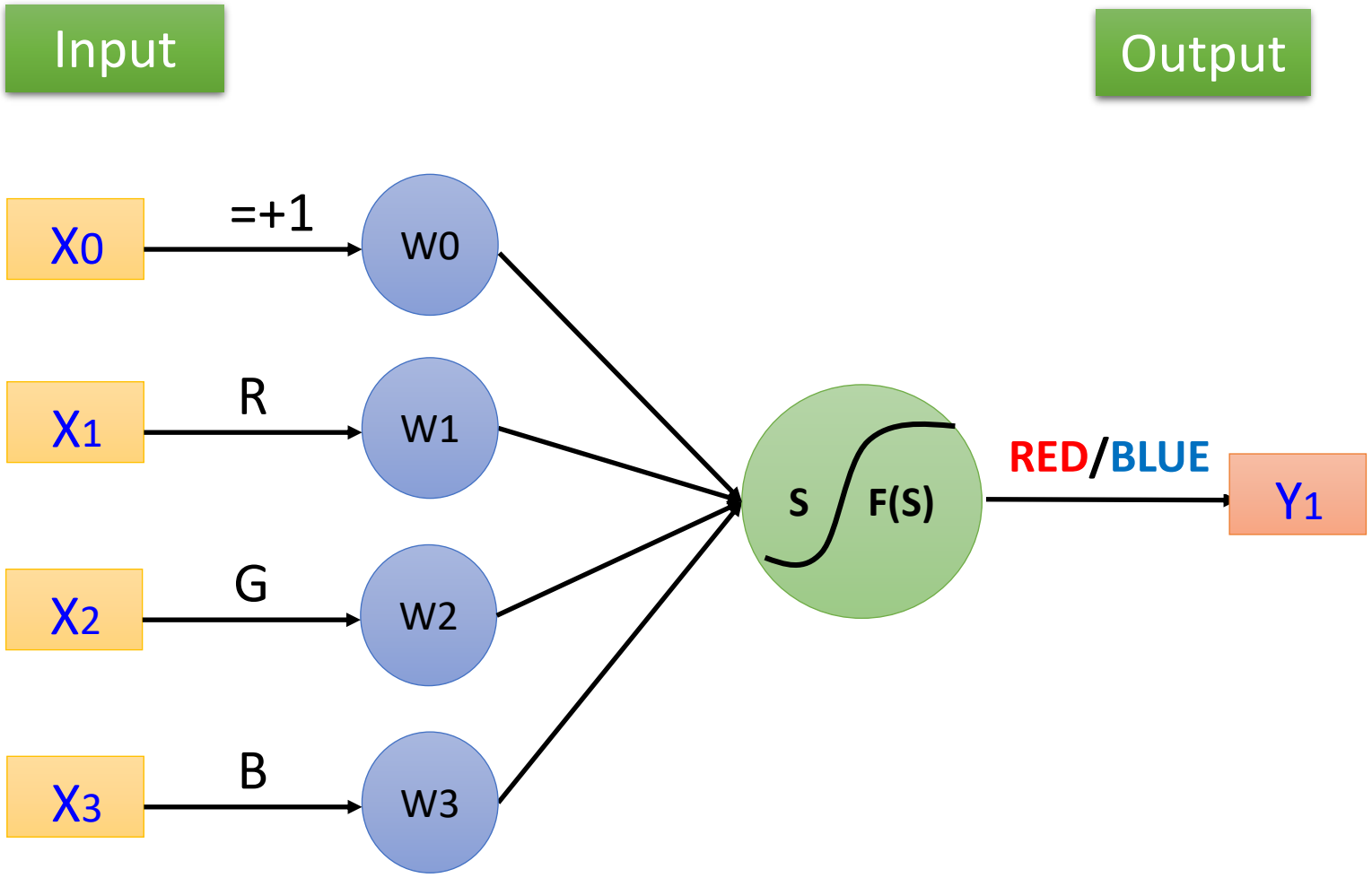
$$n = 0, 1, 2, 3$$

$$X(n) = (x_0, x_1, x_2, x_3)$$

$$W(n) = (w_0, w_1, w_2, w_3)$$

$$S = (w_0 + x_i w_i + x_i w_i + x_i w_i)$$

Learning Rate



	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

$$n = 0, 1, 2, 3$$

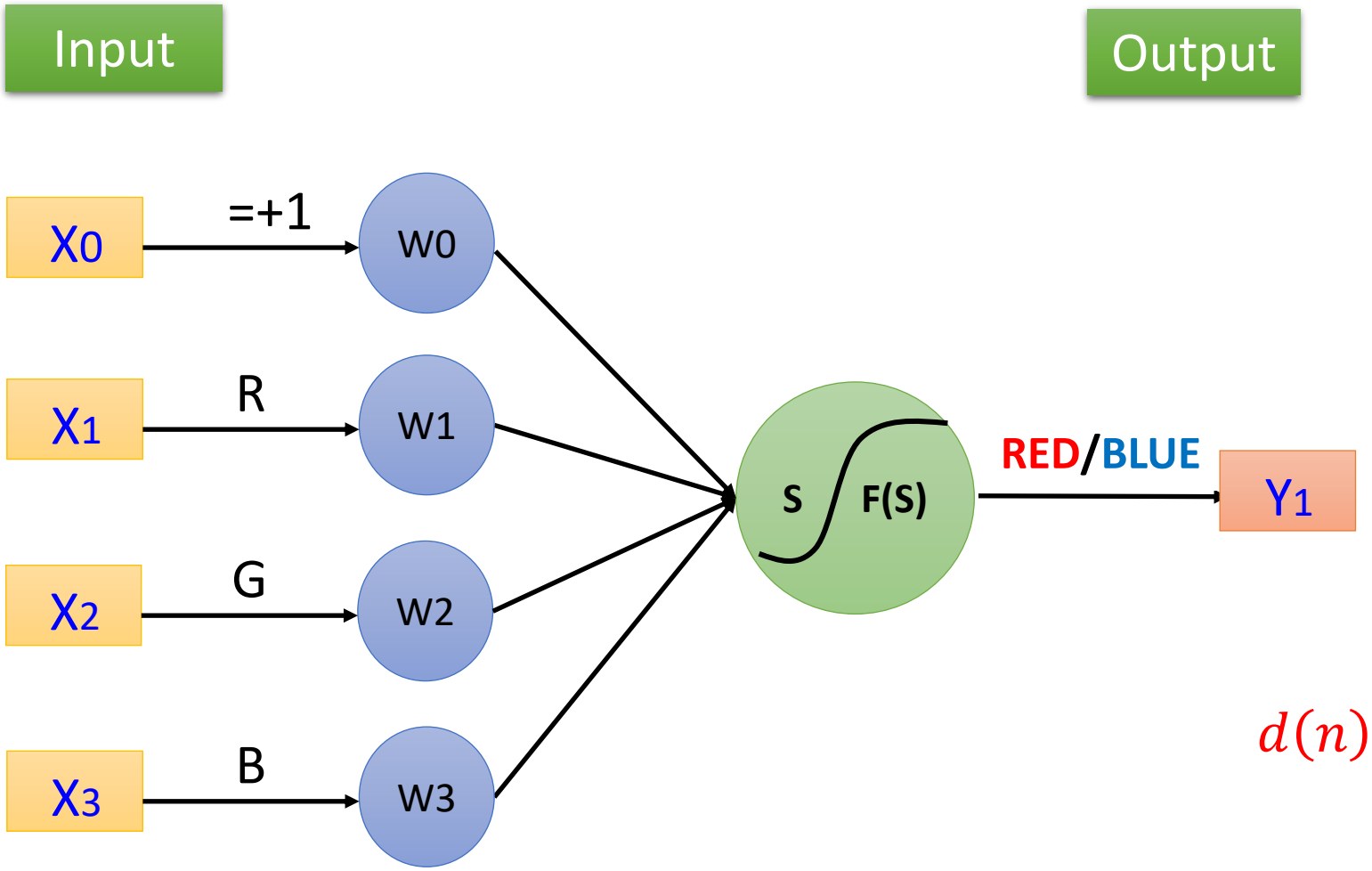
$$X(n) = (x_0, x_1, x_2, x_3)$$

$$W(n) = (w_0, w_1, w_2, w_3)$$

$$S = (w_0 + x_i w_i + x_i w_i + x_i w_i)$$

$$0 \leq \eta \leq 1$$

Desired output dj



	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

+1

$$n = 0, 1, 2, 3$$

$$X(n) = (x_0, x_1, x_2, x_3)$$

$$W(n) = (w_0, w_1, w_2, w_3)$$

$$S = (w_0 + x_i w_i + x_i w_i + x_i w_i)$$

$$0 \leq \eta \leq 1$$

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

Neural Networks 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

Step 5: 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,

$$W(n) = [b(n), W_1(n), W_2(n), \dots, W_m(n)$$

Step n=0

	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

+1

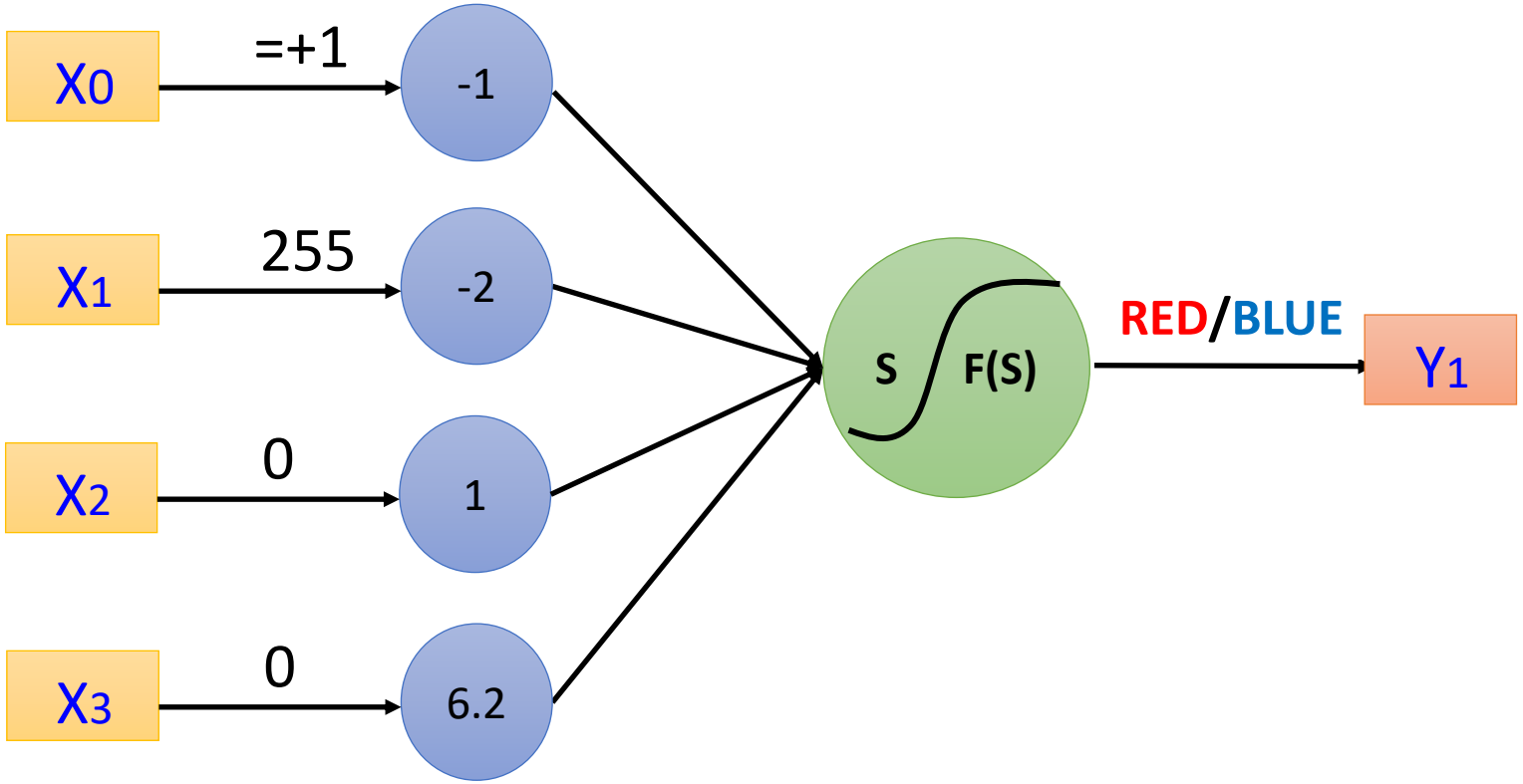
- $n = 0$
- $\eta = .001$
- $X(n) = X(0) = [x_0, x_1, x_2, x_3] = [+1,255,0,0]$
- $W(n) = W(0) = [w_0, w_1, w_2, w_3] = [-1, -2,1,6.2]$
- $d(n) = d(0) = -1$

Step n=0 -SOP

	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

+1



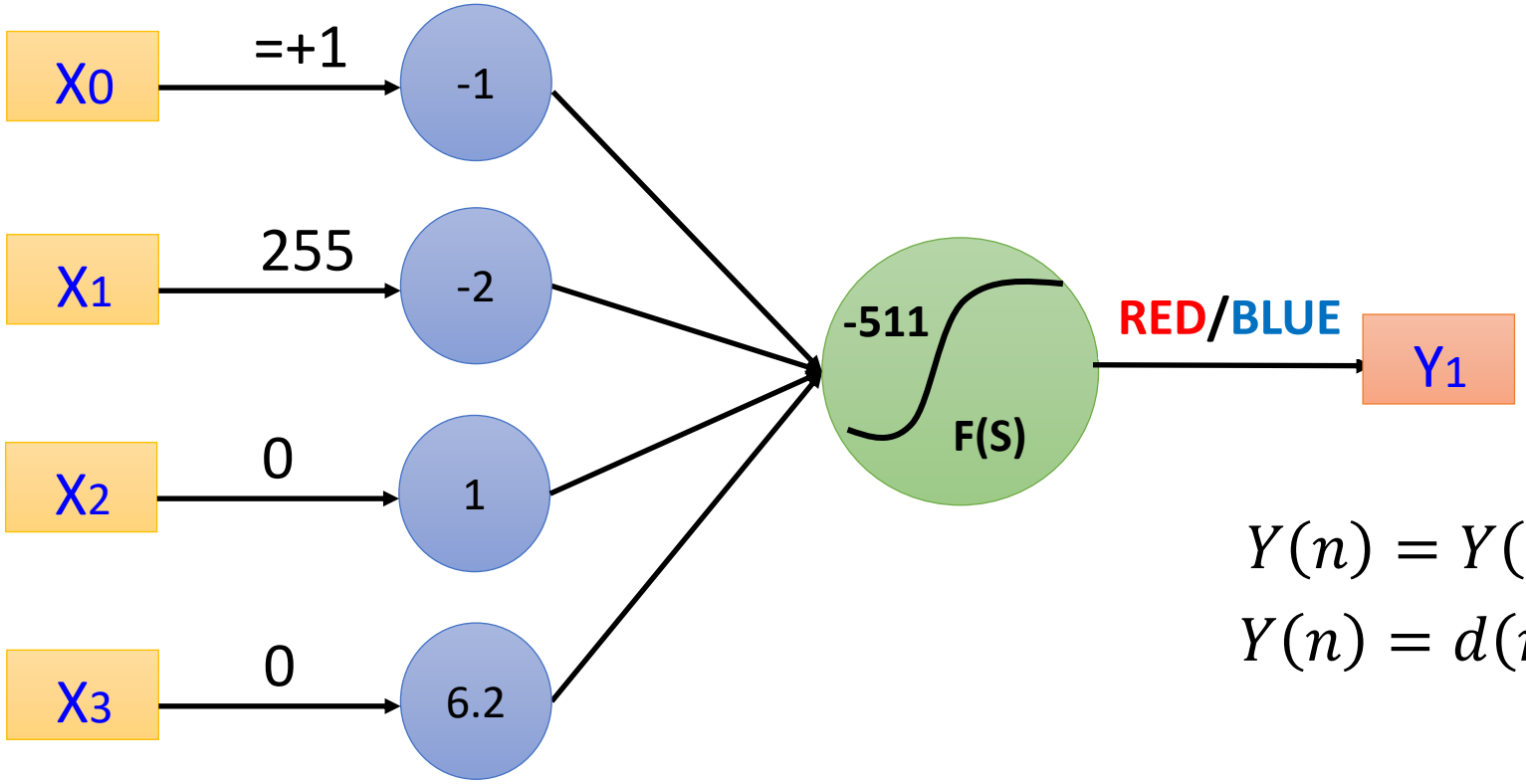
$$S = (x_0 w_0 + x_1 w_1 + x_2 w_2 + x_3 w_3) = +1 * -1 + 255 * -2 + 0 * 1 + 0 * 6.2 = -511$$

Step n=0 -Output

	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

+1



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

$$Y(n) = Y(0) = SGN(s) = SGN(-511) = -1$$

$$Y(n) = d(n) = -1$$ ***x(n) belong to C1 (RED)***

Weights are correct.

No Adaptation Required

Step n=1

	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

+1

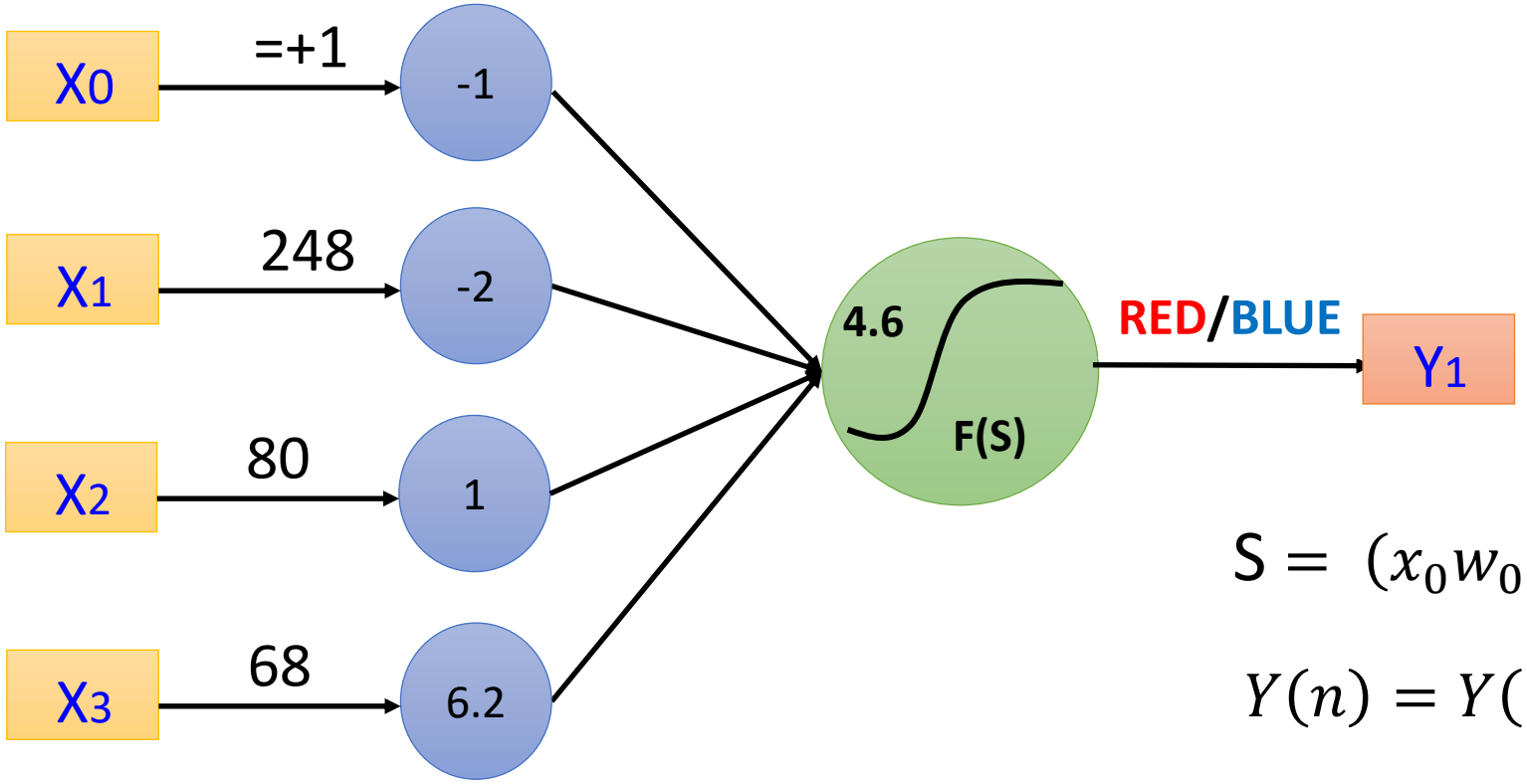
- $n = 1$
- $\eta = .001$
- $X(n) = X(1) = [x_0, x_1, x_2, x_3] = [+1,248,80,68]$
- $W(n) = W(1) = [w_0, w_1, w_2, w_3] = [-1, -2,1,6.2]$
- $d(n) = d(1) = -1$

Step n=1 -Output

	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

+1



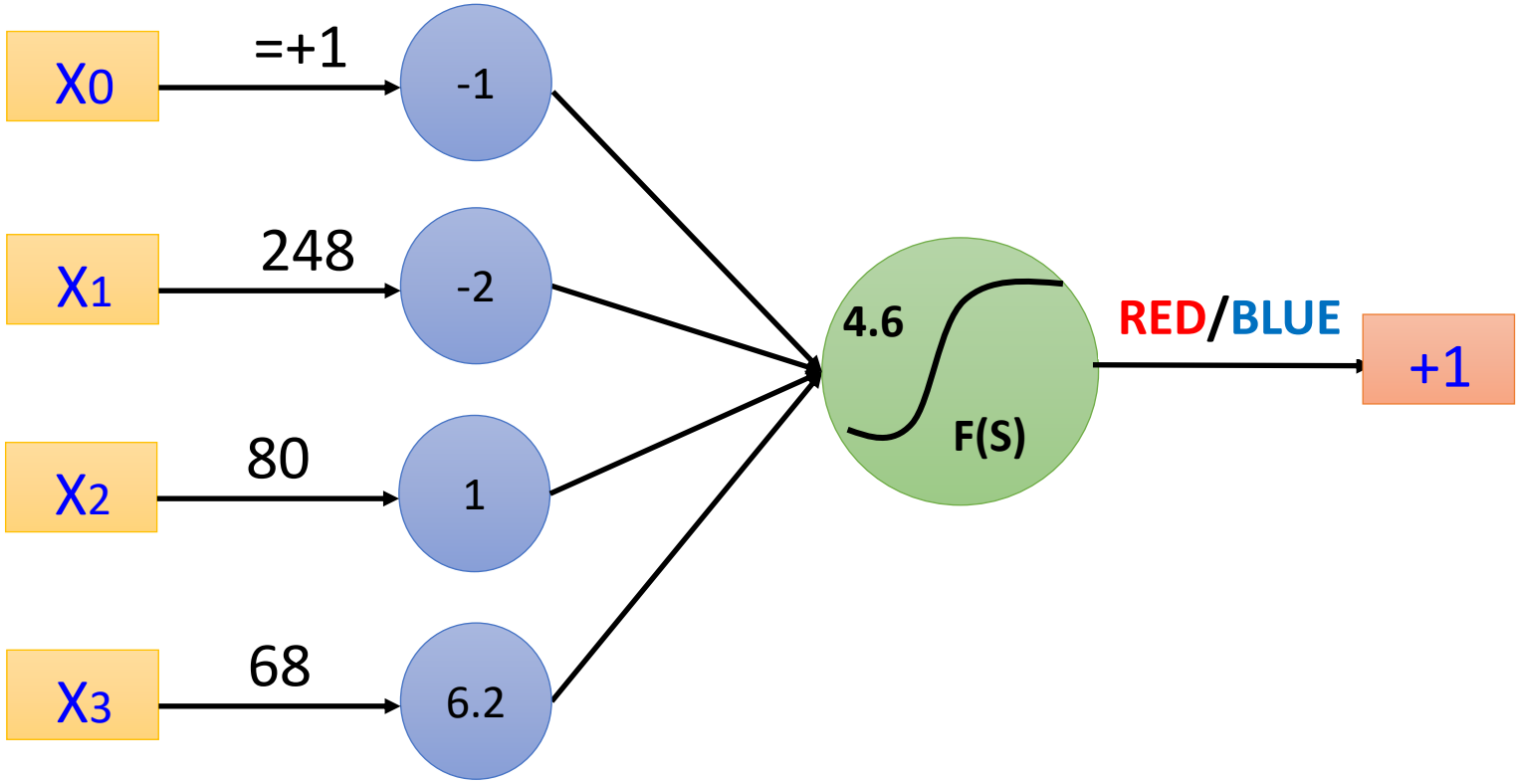
$$SGN(s) = \begin{cases} +1, s \geq 0 \\ -1, s \leq 0 \end{cases}$$

$$S = (x_0 w_0 + x_1 w_1 + x_2 w_2 + x_3 w_3) = 4.6$$

$$Y(n) = Y(1) = SGN(s) = SGN(4.6) = +1$$

x(n) belong to C2 (BLUE)

Step n=1 –Output Predicted vs Desired



	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

+1

$$Y(n) = Y(1) = +1$$
$$\mathrm{d}(n) = d(1) = -1$$
$$Y(n) \neq d(n)$$

Weights are incorrect.
Adaptation Required

Weights Adaptation

- According to

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

- Where n=1

$$W(1+1) = W(1) + \eta[d(1) - Y(1)]X(1)$$
$$W(2) = [-1, -2, 1, 6.2] + .001[-1 - (+1)][+1, 248, 80, 68]$$
$$W(2) = [-1, -2, 1, 6.2] + .001[-2][+1, 248, 80, 68]$$
$$W(2) = [-1, -2, 1, 6.2] + [-0.002, -0.496, -0.16, -0.136]$$
$$W(2) = [-1.002, -2.496, 0.84, 6.064]$$
$$W(2) = [-1.002, -2.496, 0.84, 6.064]$$

Step n=2

	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

+1

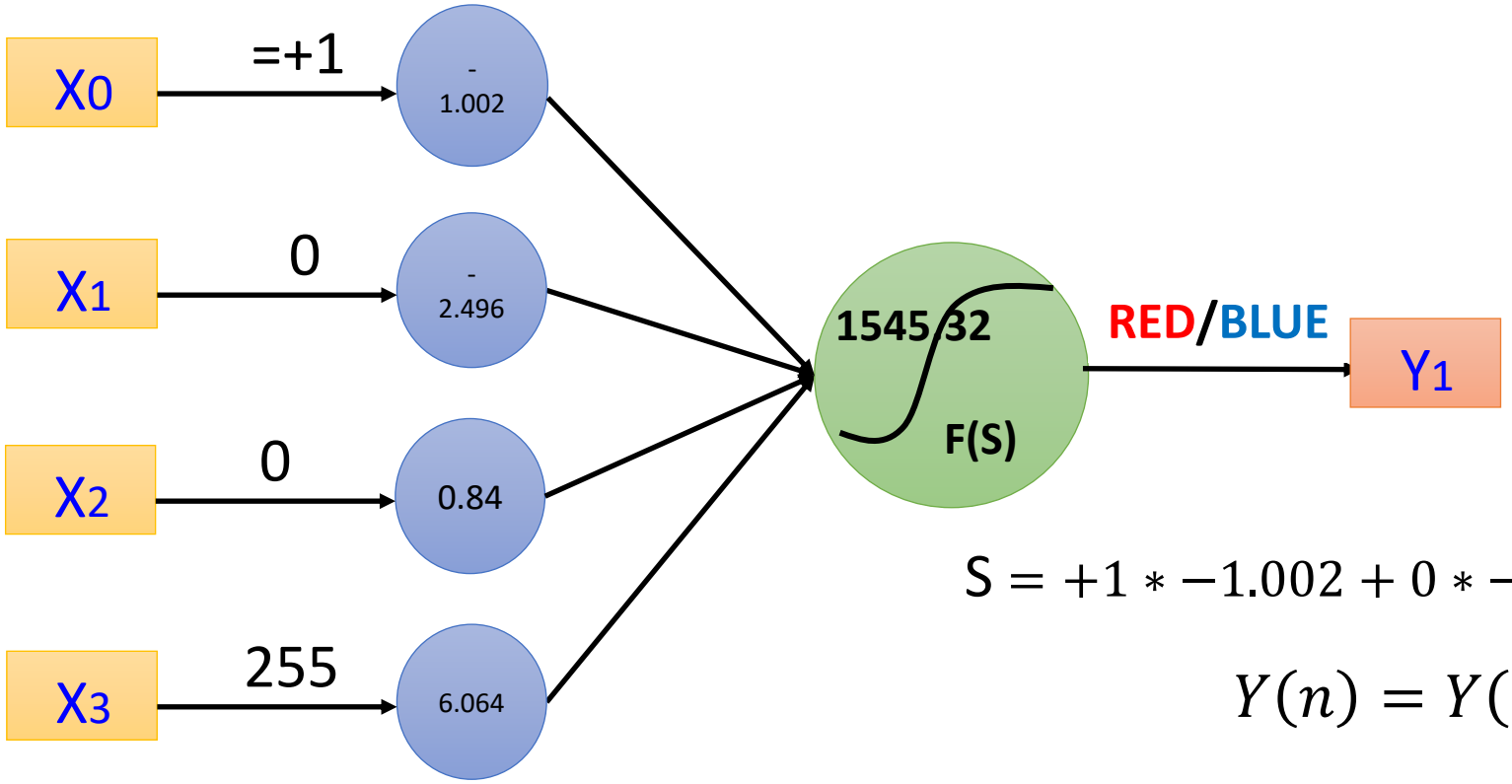
- $n = 2$
- $\eta = .001$
- $X(n) = X(2) = [x_0, x_1, x_2, x_3] = [+1,0,0,255]$
- $W(n) = W(2) = [w_0, w_1, w_2, w_3] = [-1.002, -2.496, (+)0.84,6.064]$
- $d(n) = d(2) = +1$

Step n=2 -Output

	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

+1



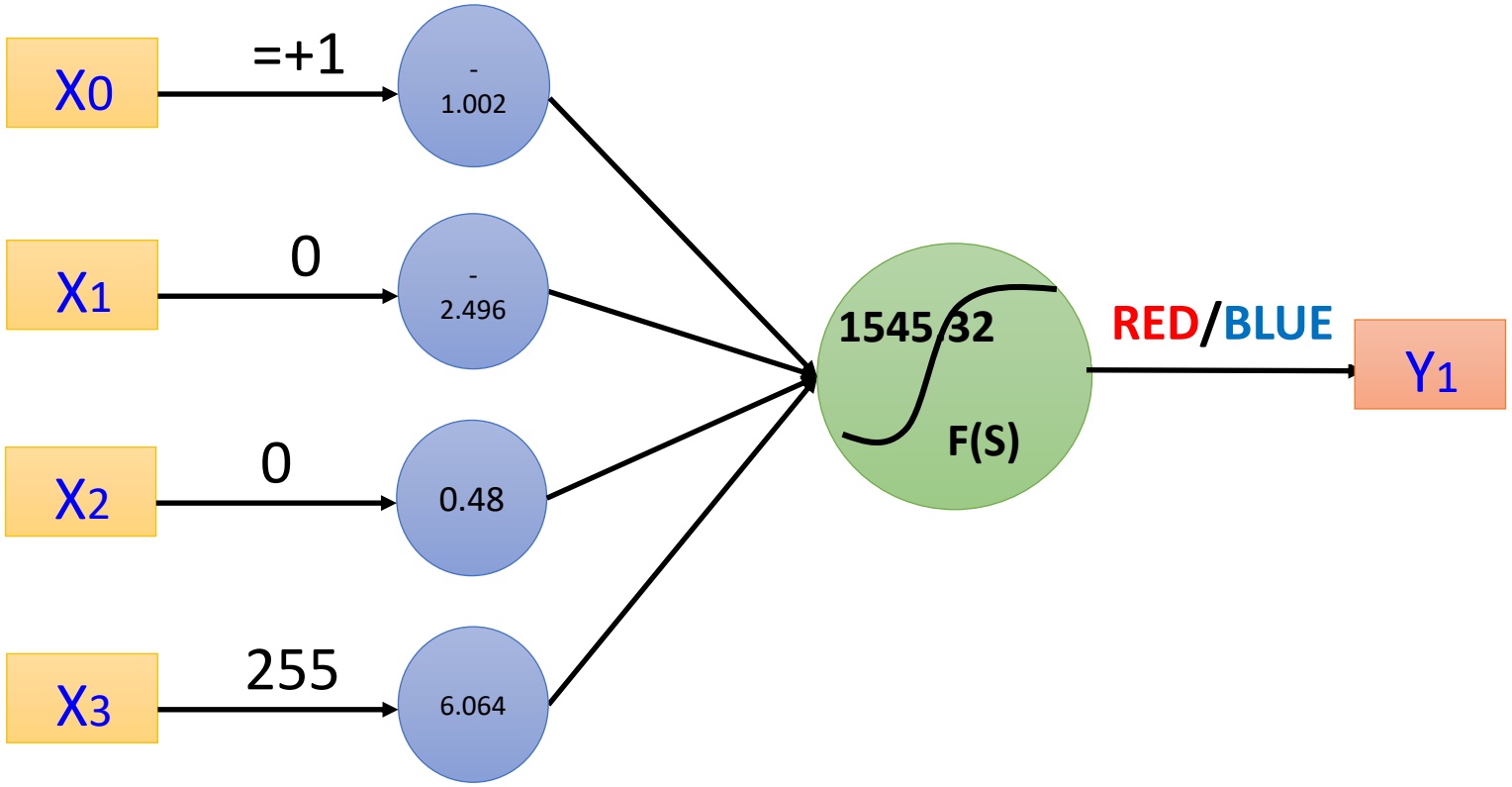
$$SGN(s) = \begin{cases} +1, s \geq 0 \\ -1, s \leq 0 \end{cases}$$

$$S = +1 * -1.002 + 0 * -2.496 + 0 * 0.84 + 255 * 6.064 = 1545.32$$

$$Y(n) = Y(2) = SGN(s) = SGN(1545.32) = +1$$

x(n) belong to C2 (BLUE)

Step n=2
Predicted vs. Desired



	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

+1

$$Y(n) = Y(2) = +1$$
$$\mathrm{d}(n) = d(2) = +1$$
$$Y(n) = d(2)$$

Weights are correct

No Adaptation

Step n=3

	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

+1

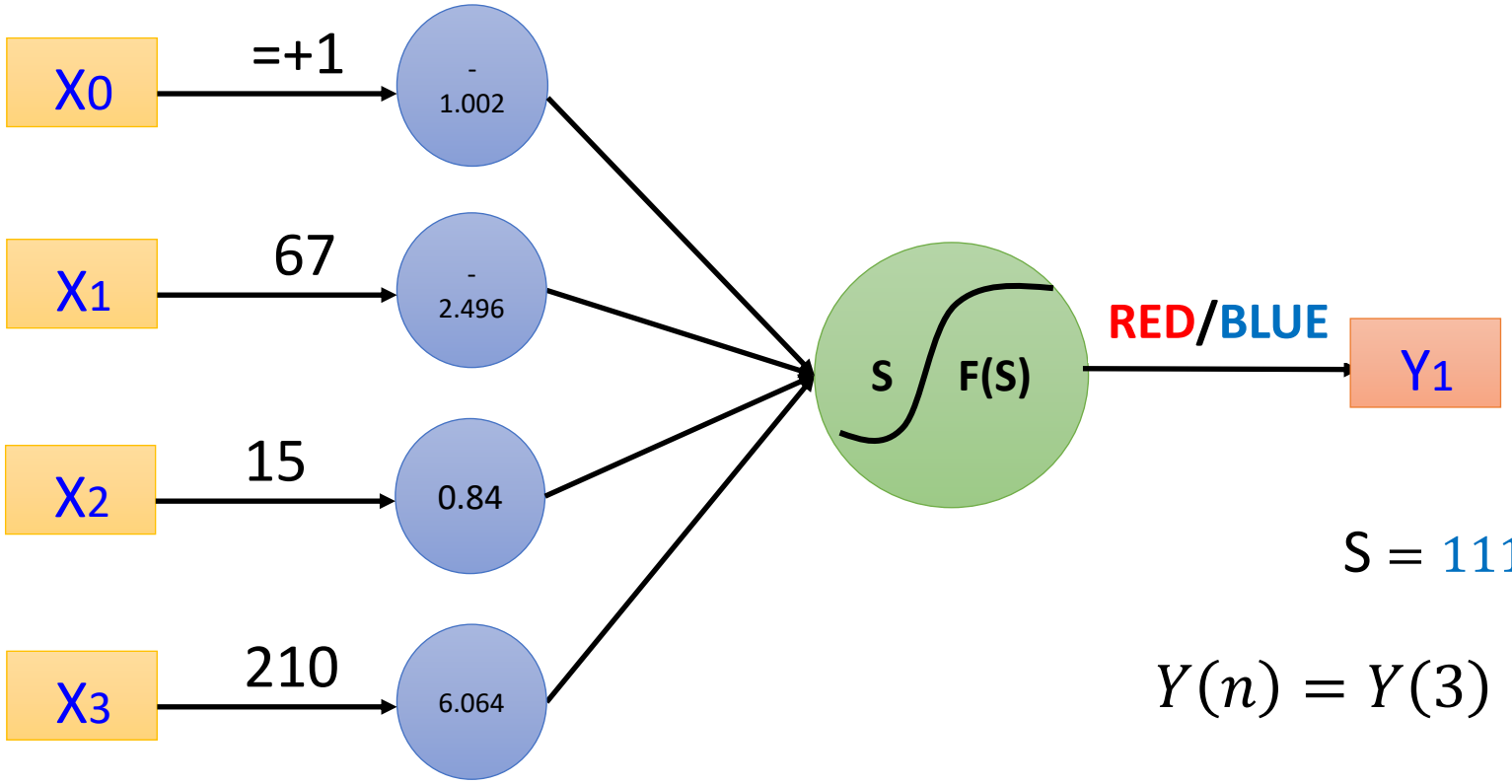
- $n = 3$
- $\eta = .001$
- $X(n) = X(3) = [x_0, x_1, x_2, x_3] = [+1,67,15,210]$
- $W(n) = W(3) = W(2) = [-1.002, -2.496, (+)0.84,6.064]$
- $d(n) = d(3) = +1$

Step n=3 -Output

	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

+1



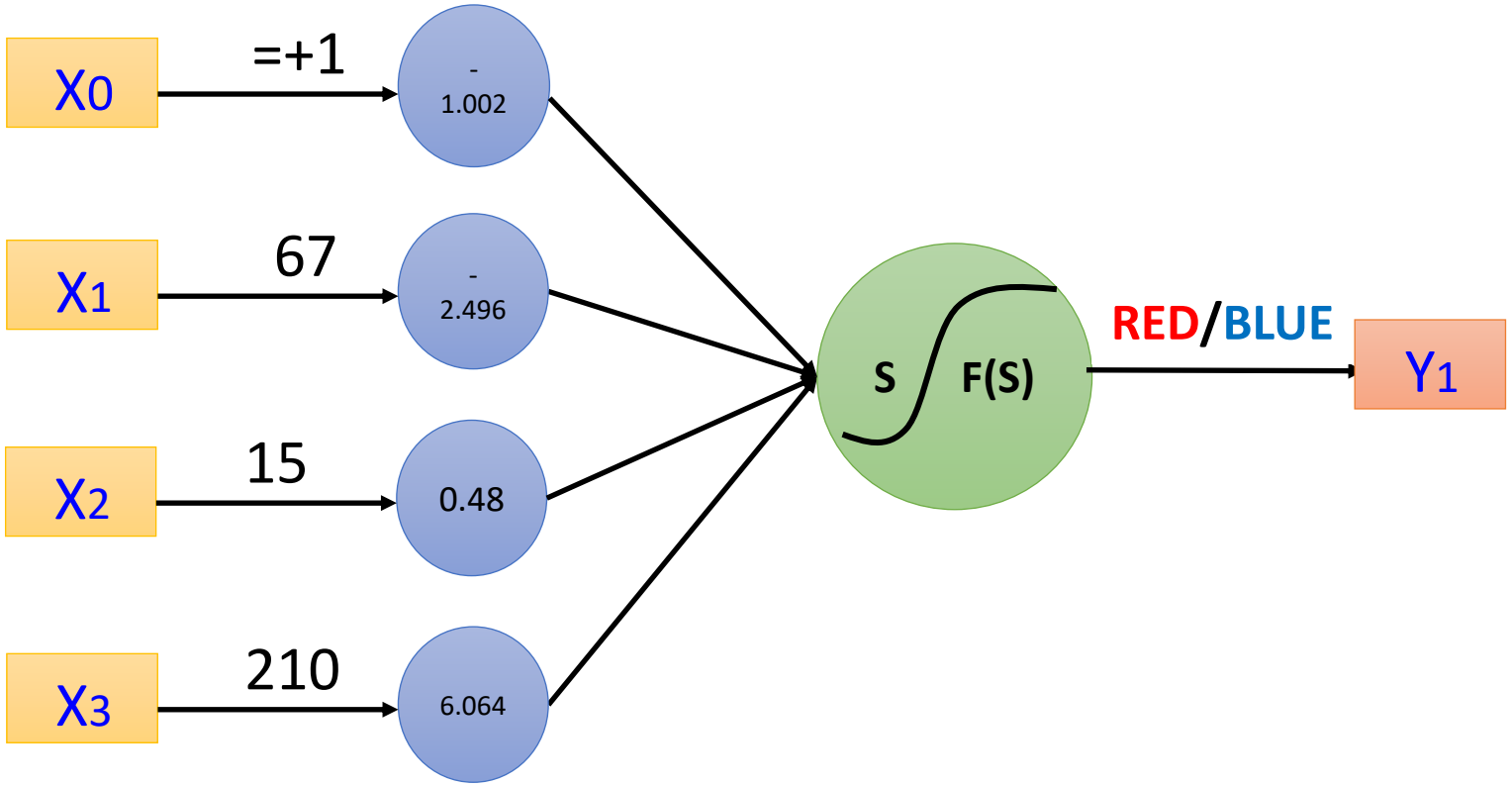
$$SGN(s) = \begin{cases} +1, s \geq 0 \\ -1, s \leq 0 \end{cases}$$

$$S = 1117.806$$

$$Y(n) = Y(3) = SGN(s) = SGN(1117.806) = +1$$

x(n) belong to C2 (BLUE)

Step n=3
Predicted vs. Desired



	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

+1

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

Weights are correct

No Adaptation

Step n=4

	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

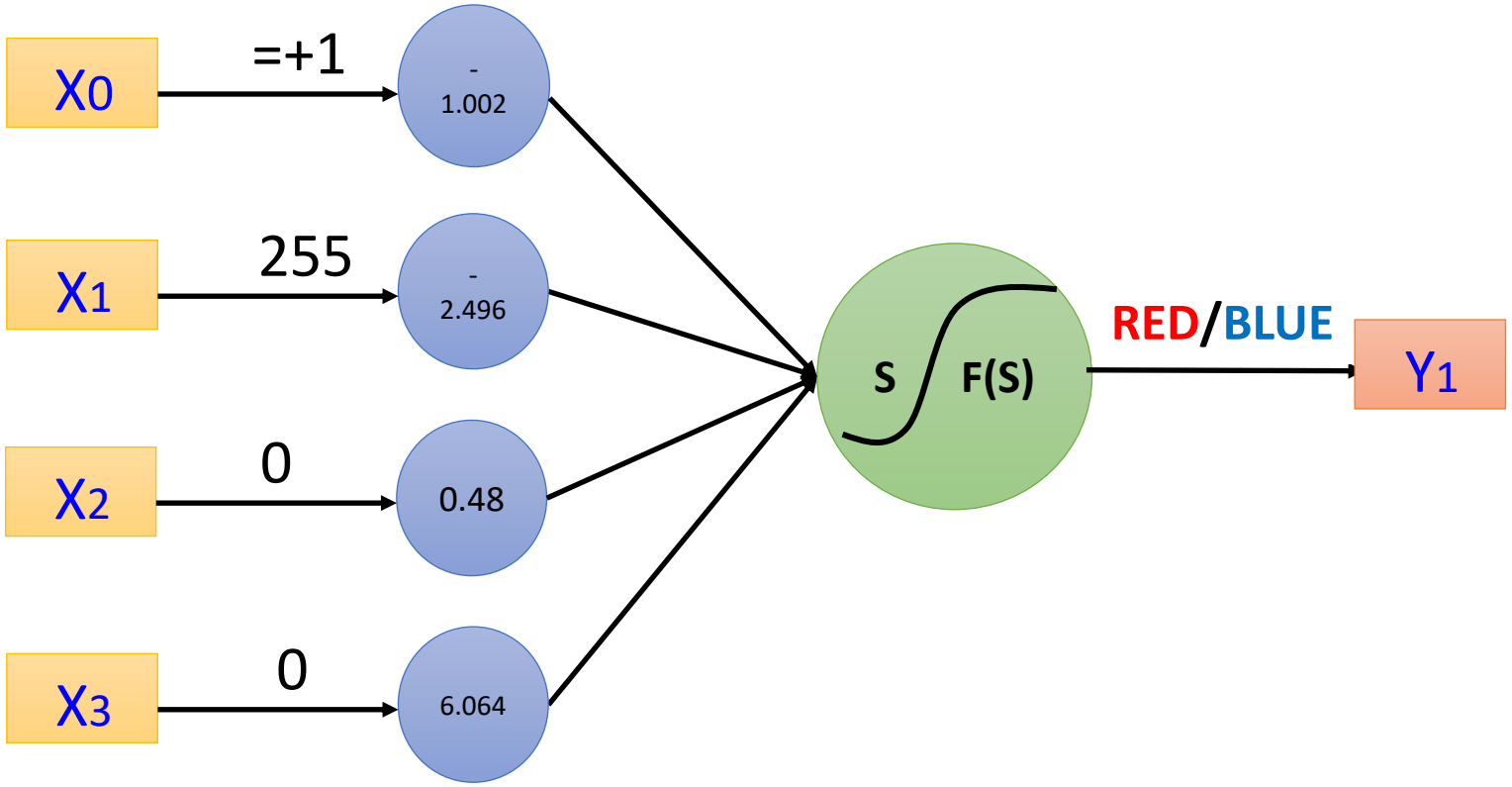
-1

+1

- $n = 4$
- $\eta = .001$
- $X(n) = X(4) = [x_0, x_1, x_2, x_3] = [+1,255,0,0]$
- $W(n) = W(4) = W(2) = [-1.002, -2.496, +0.84,6.064]$
- $d(n) = d(4) = -1$

Note: Step n=4 because first row didn't calculated by these weights

Step n=4
Predicted vs. Desired



	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

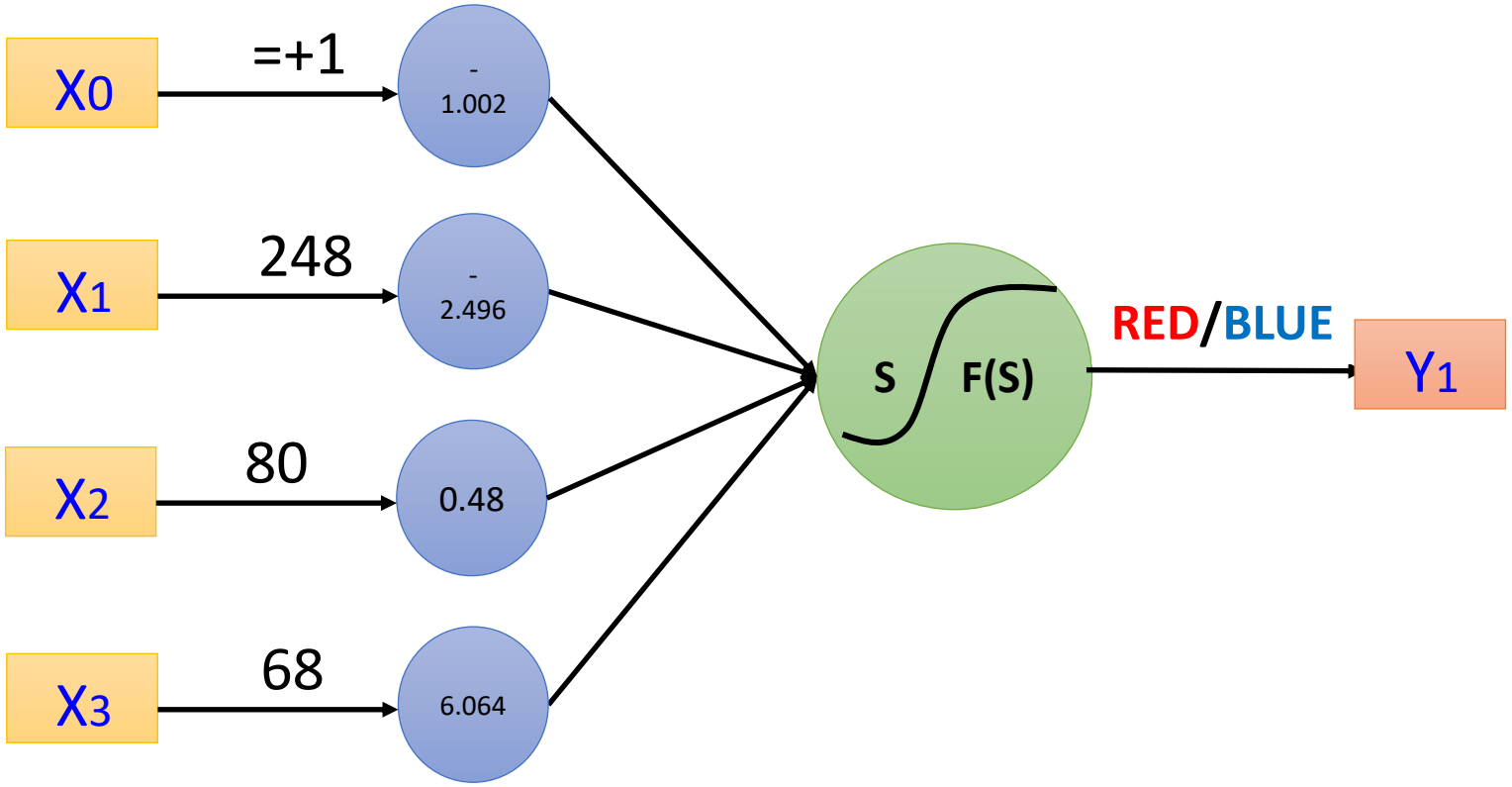
+1

$$Y(n) = Y(4) = -1$$
$$\mathrm{d}(n) = d(4) = -1$$
$$Y(n) = d(4)$$

Weights are correct

No Adaptation

Step n=5
Predicted vs. Desired



	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

+1

$$Y(n) = Y(5) = -1$$
$$\mathrm{d}(n) = d(5) = -1$$
$$Y(n) = d(5)$$

Weights are correct

No Adaptation

Correct Weights

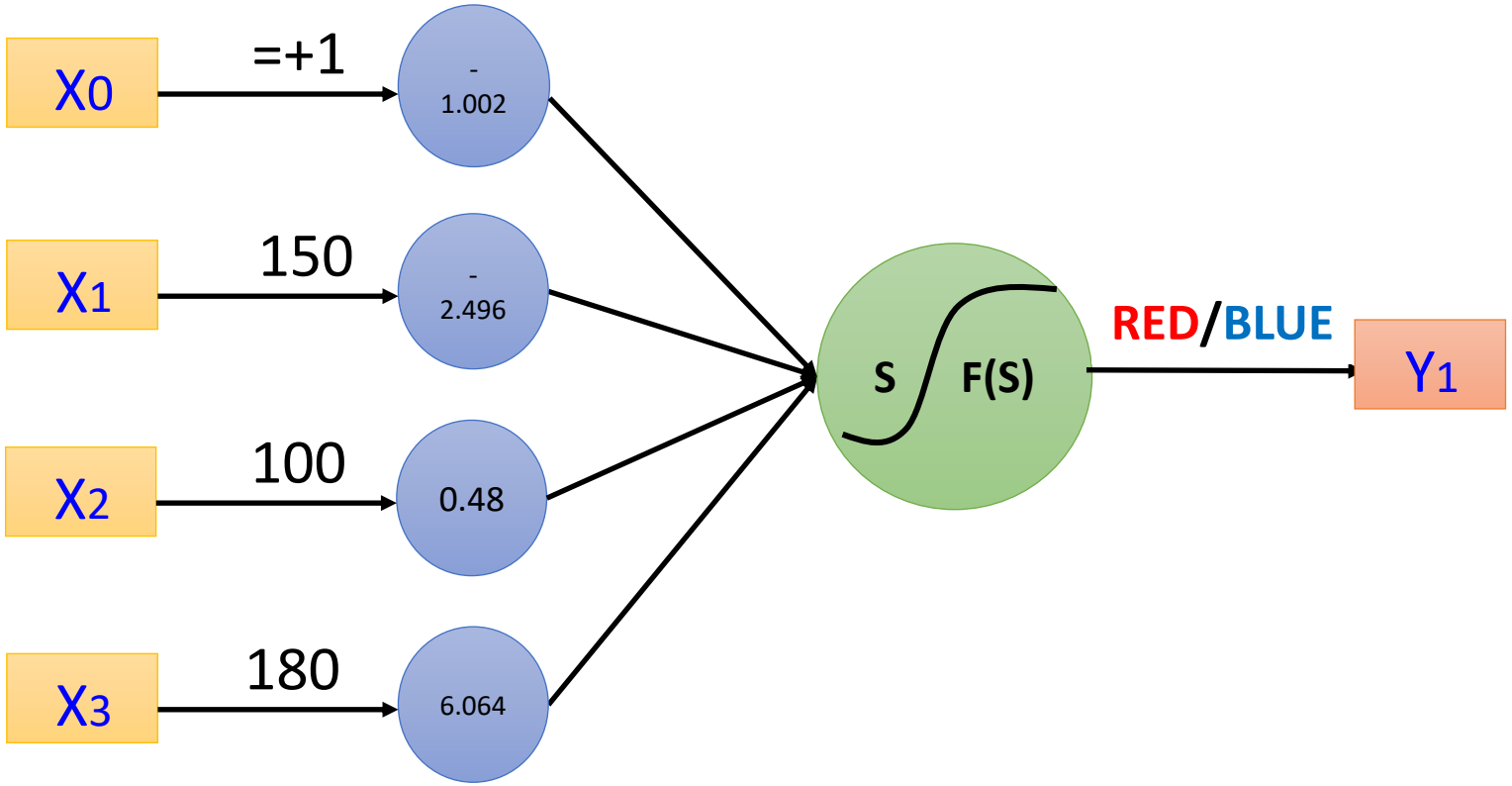
- All results are correct with the current weight,
- Now, **Training phase is finished.**
- **Test phase:** unknown color of values **R=150, G=100, B=180**

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

	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

+1



$$S = (x_0 w_0 + x_1 w_1 + x_2 w_2 + x_3 w_3)$$

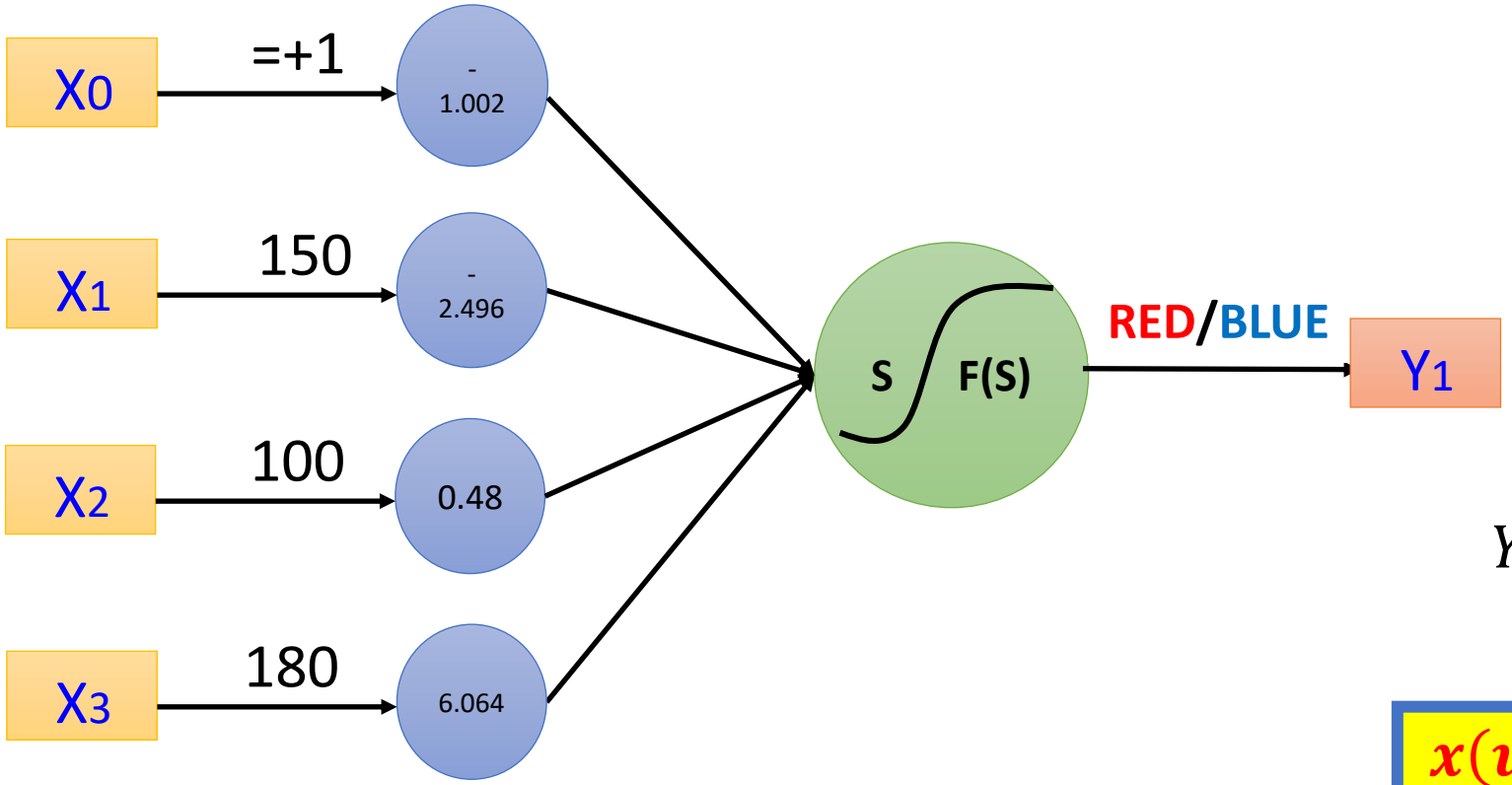
$$= +1 * -1.002 + 150 * -2.496 + 100 * 0.48 + 180 * 6.064 = 800.118$$

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

	RED	GREEN	BLUE
RED	255	0	0
RED	248	80	68
BLUE	0	0	255
BLUE	67	15	210

-1

+1



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

$$Y = SGN(s) = SGN(800.118) = +1$$

x(unknown) belong to C2 (BLUE)

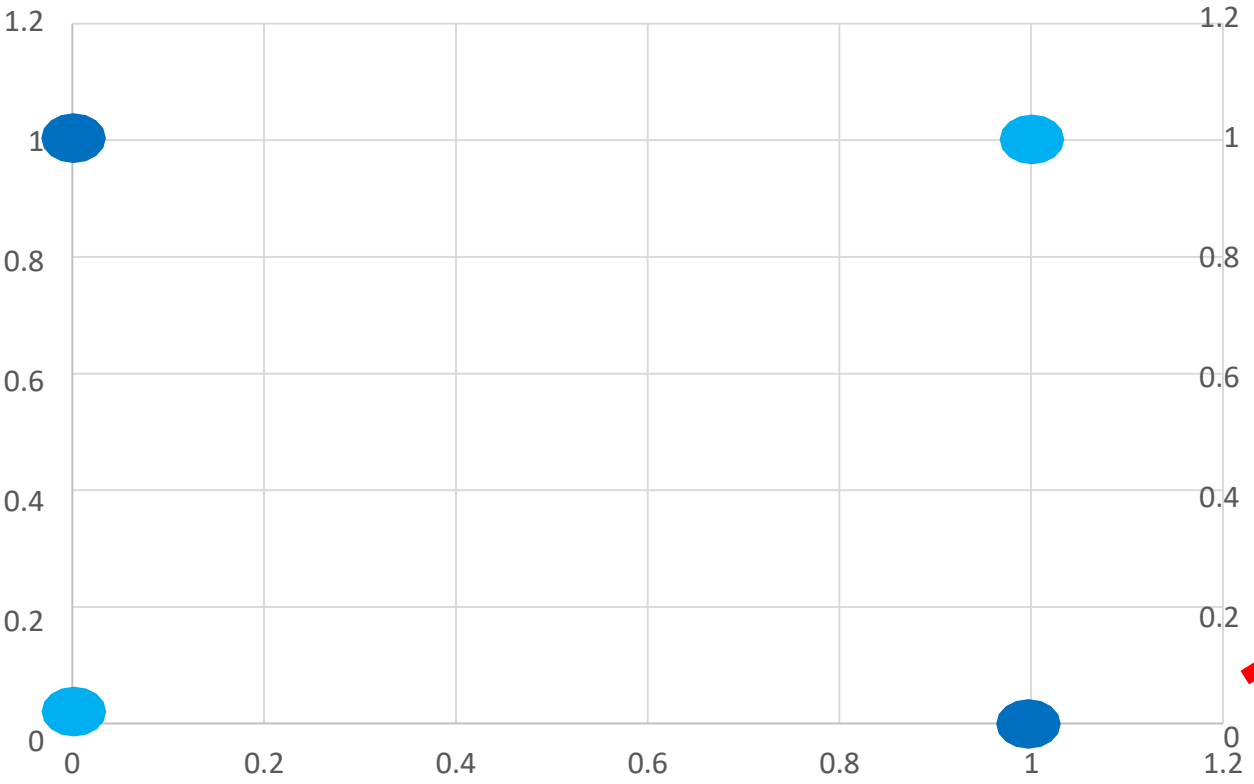
Example of Multi-layer

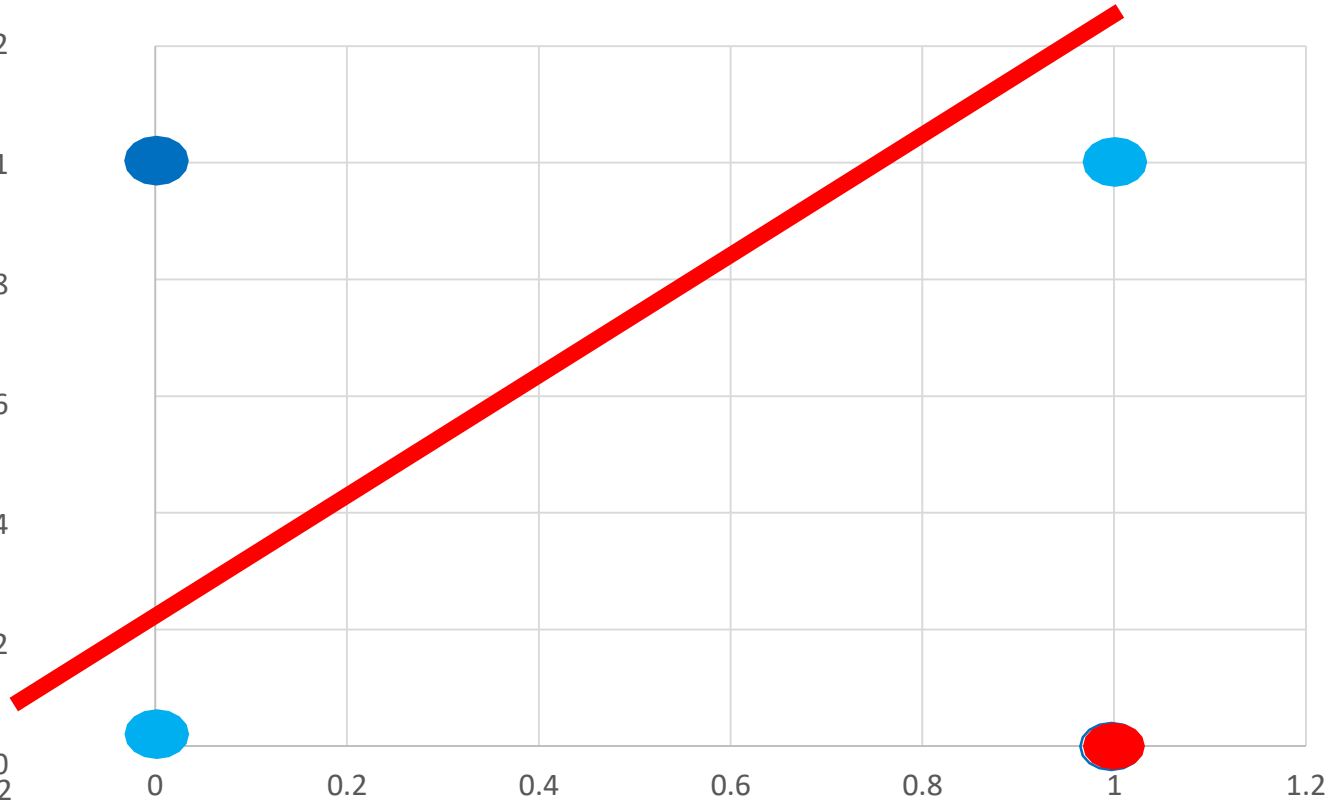
Classification Example

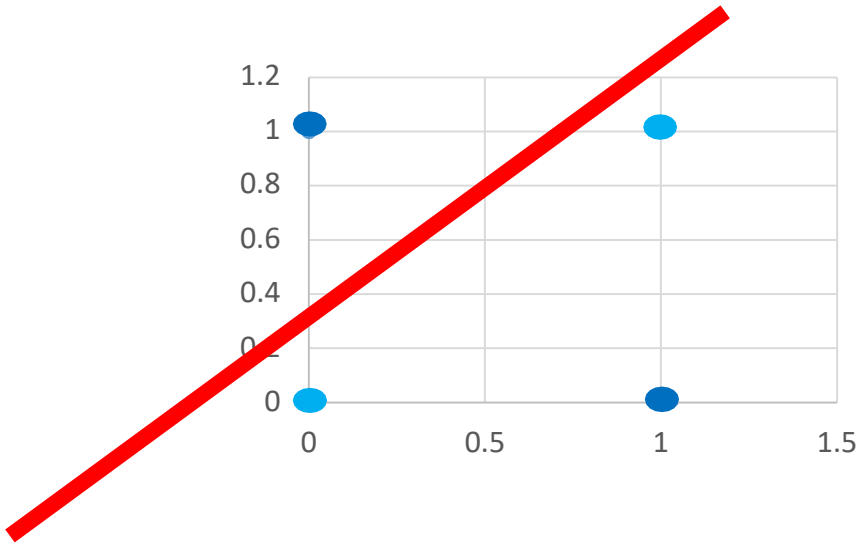
	A	B
1	1	0
	0	1
0	0	0
	1	1

Neural Networks

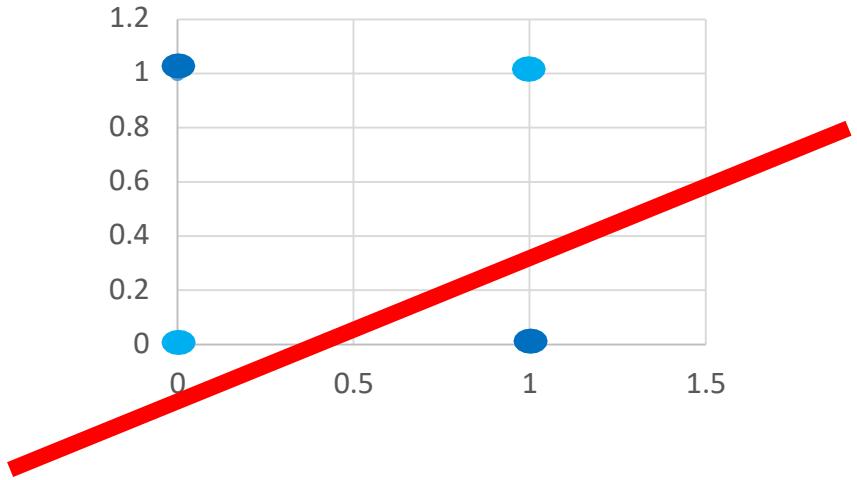
	A	B
1	1	0
	0	1
0	0	0
	1	1

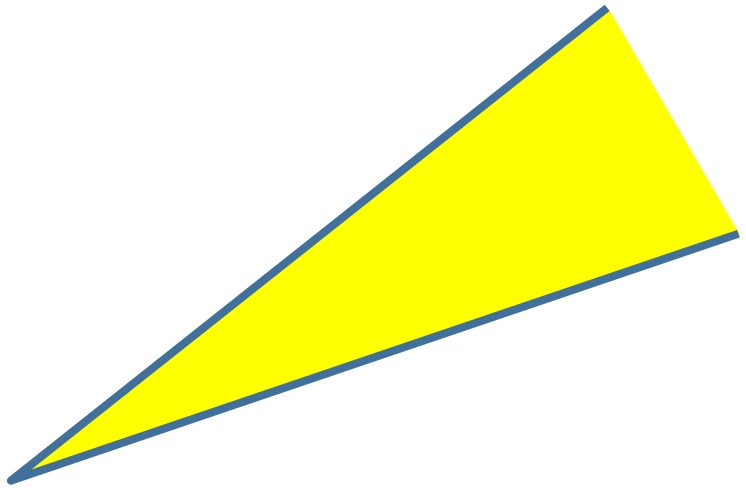




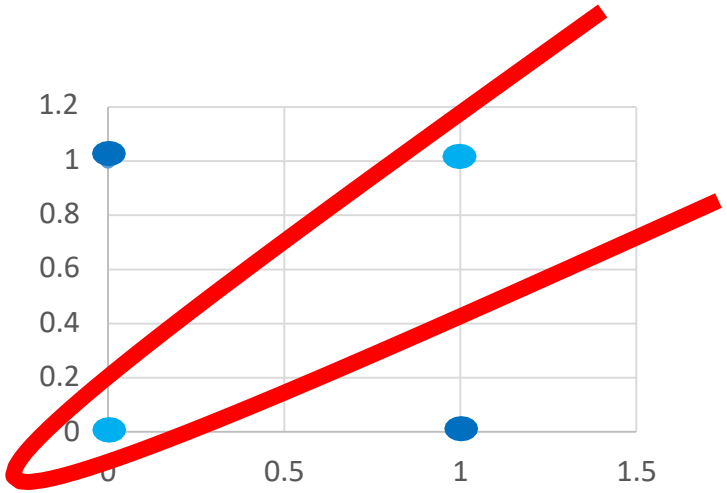

1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5

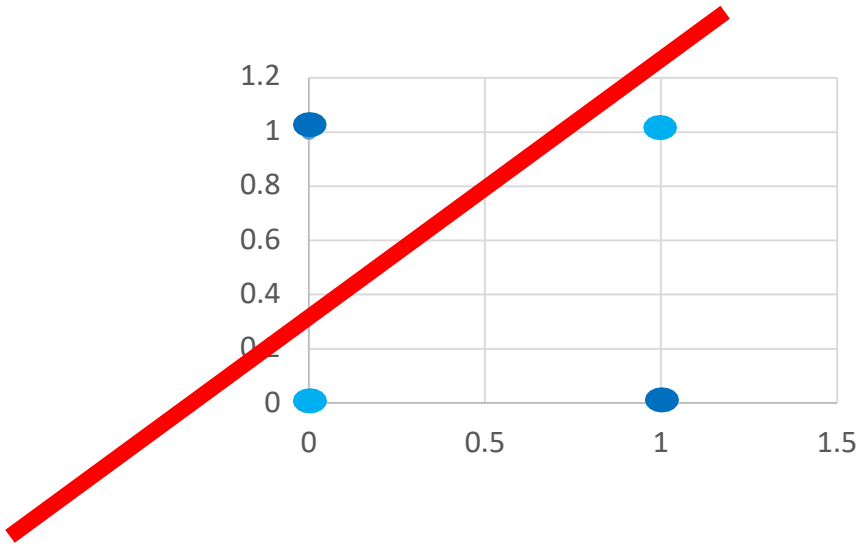
+

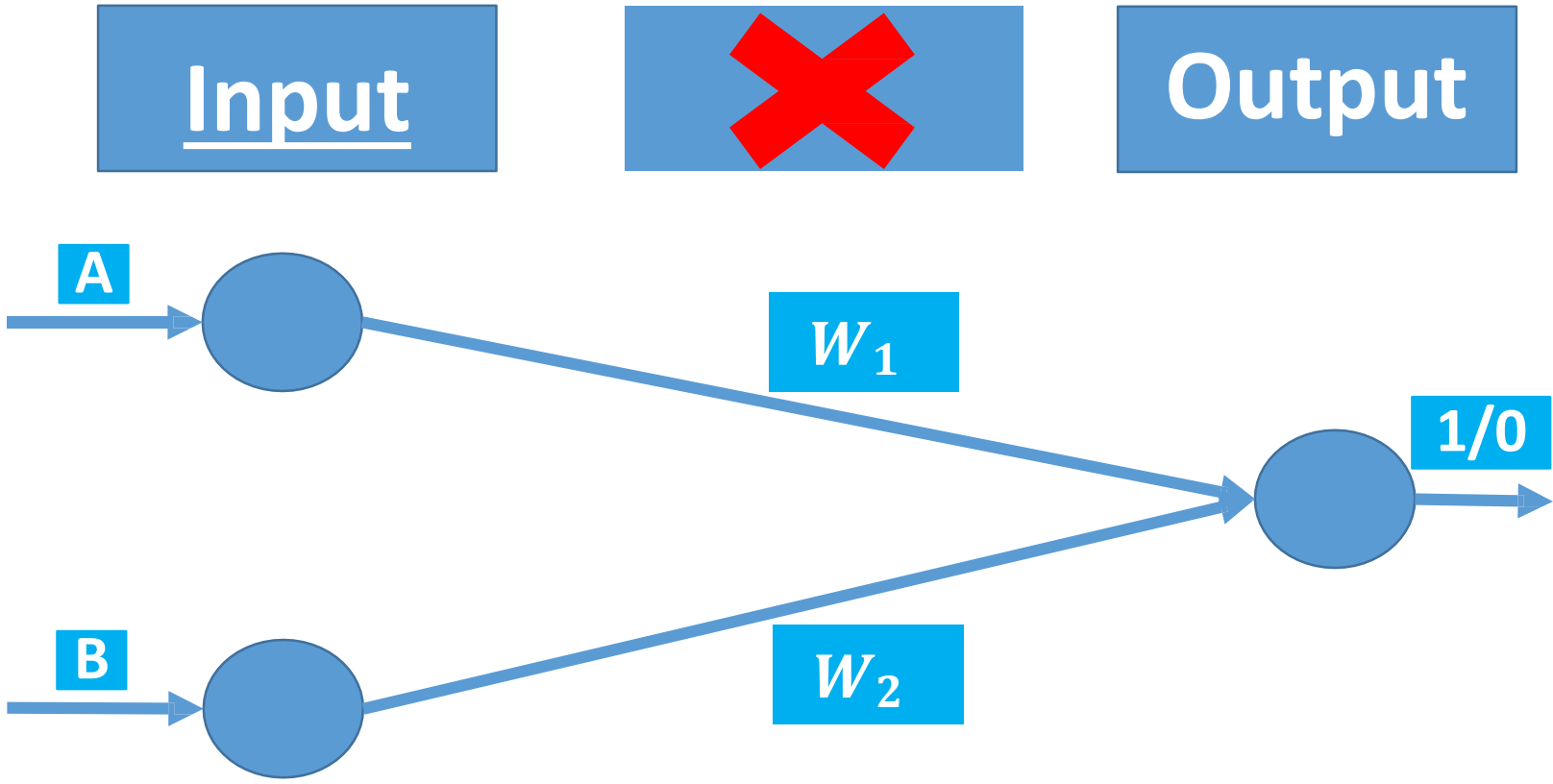

1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5

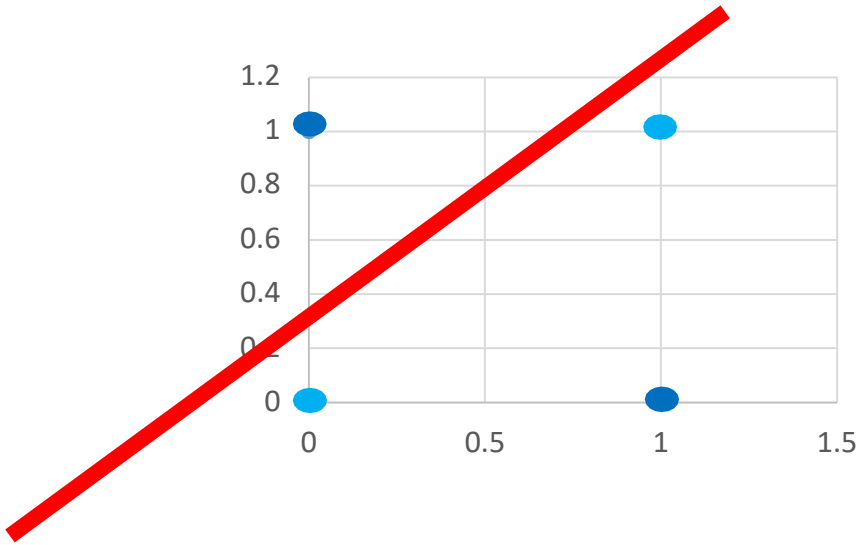


=

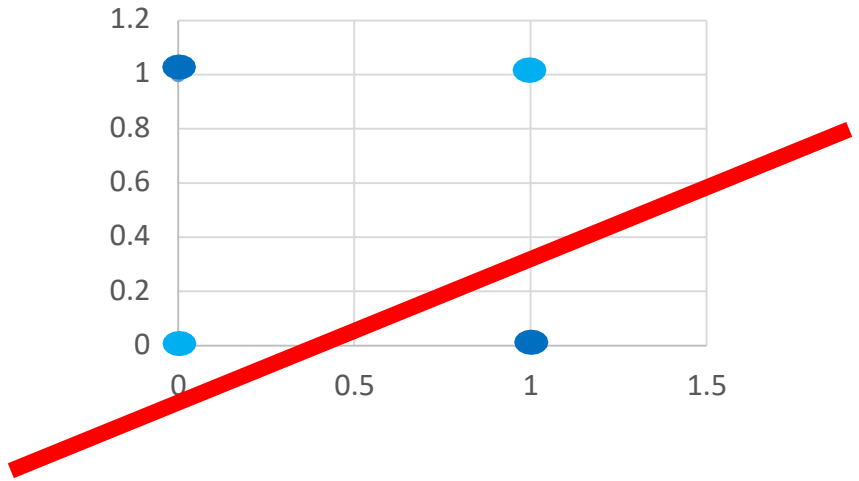

1.2
1
0.8
0.6
0.4
0.2
0
0.5
1
1.5


1.2
1
0.8
0.6
0.4
0
0
0.5
1
1.5

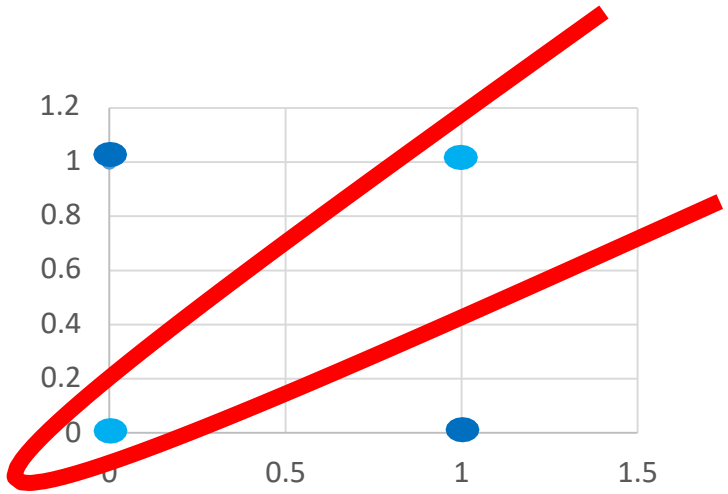

Input
Output
A
B
W_1
W_2
1/0

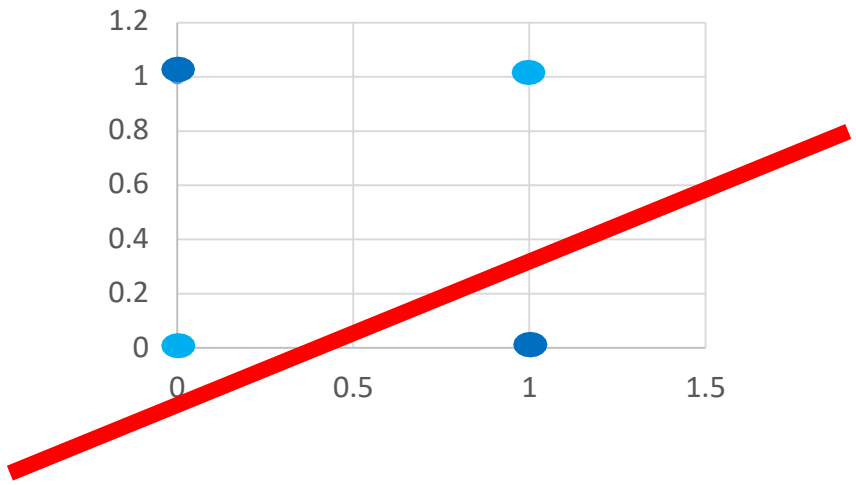

1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5

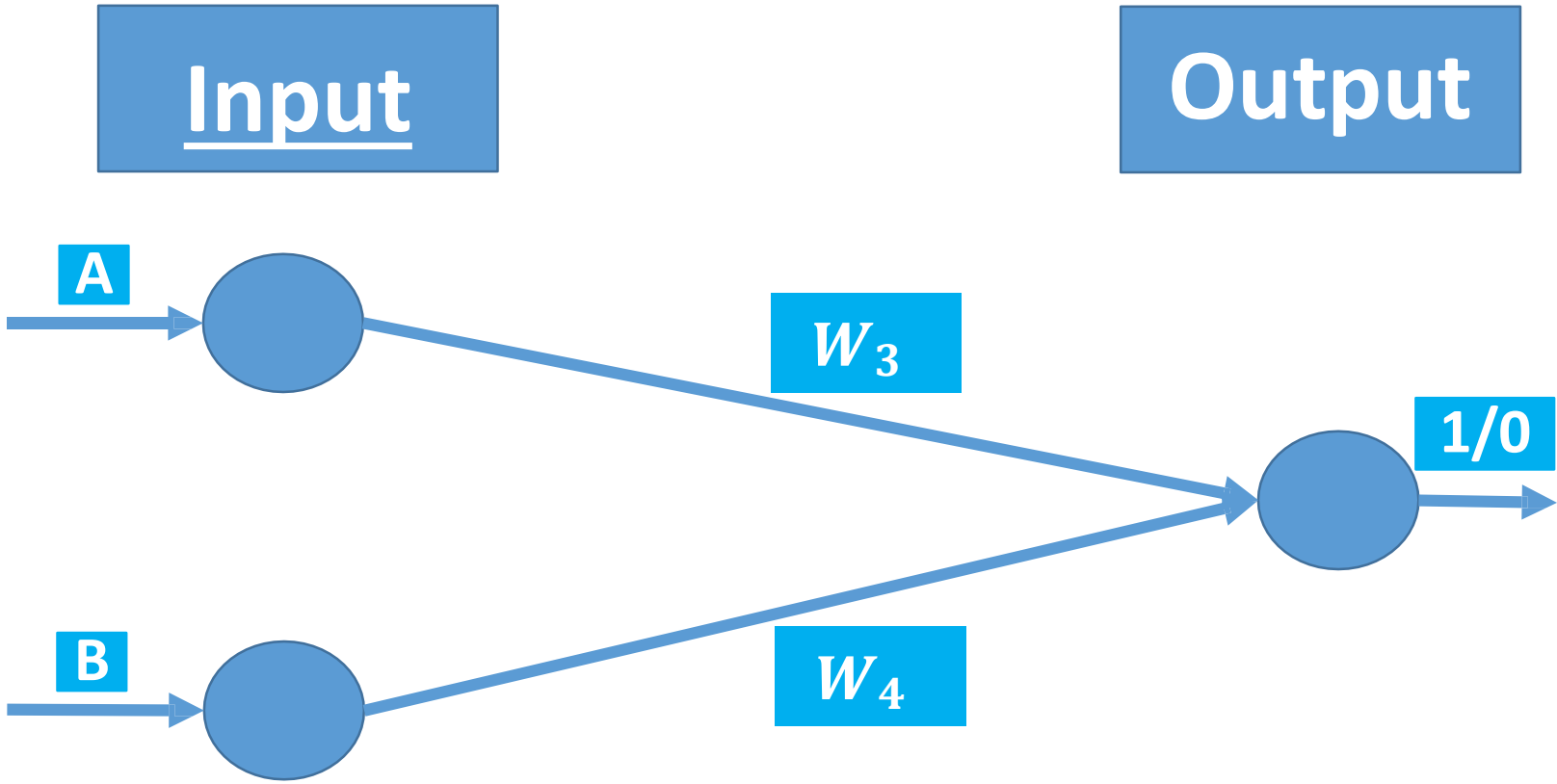
+

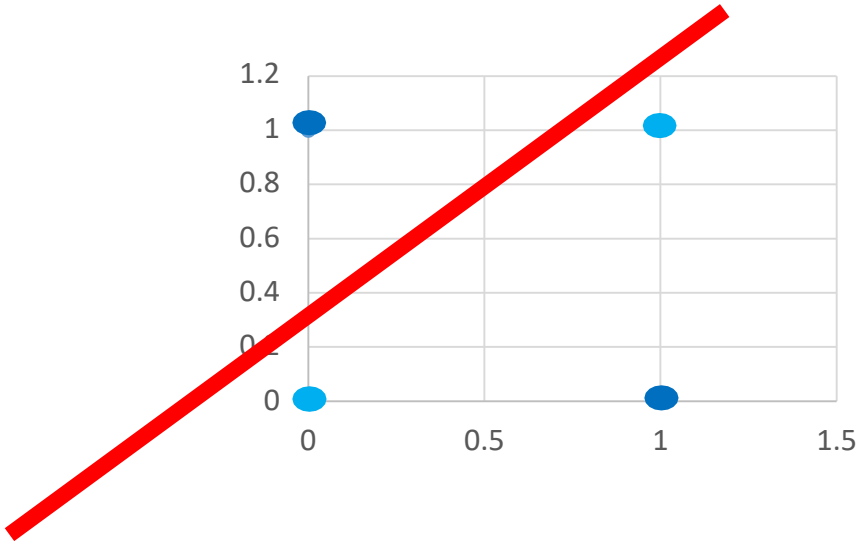

1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5

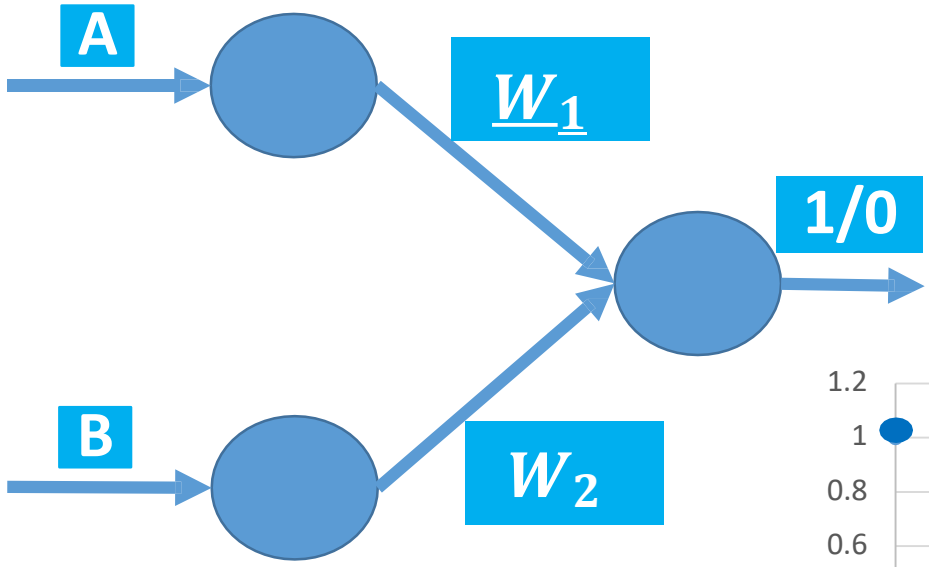
=

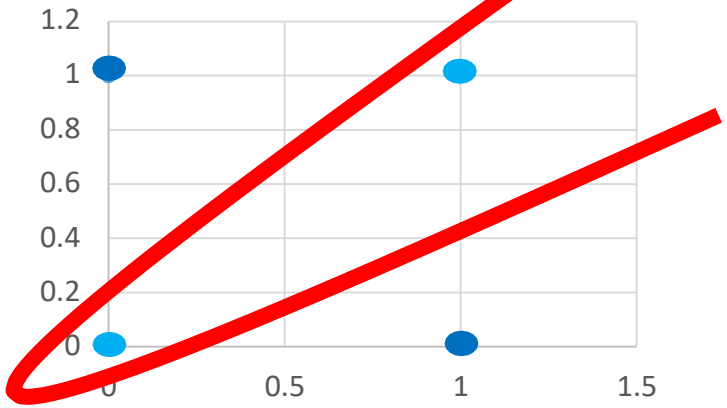

1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5

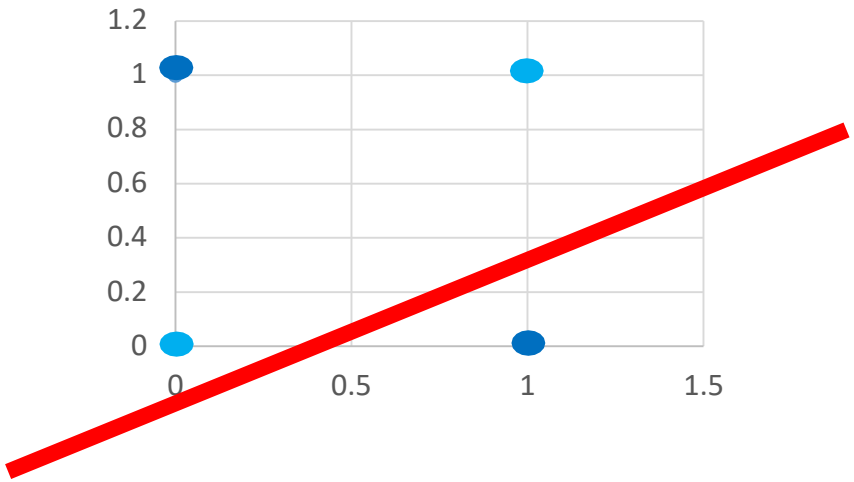

1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5

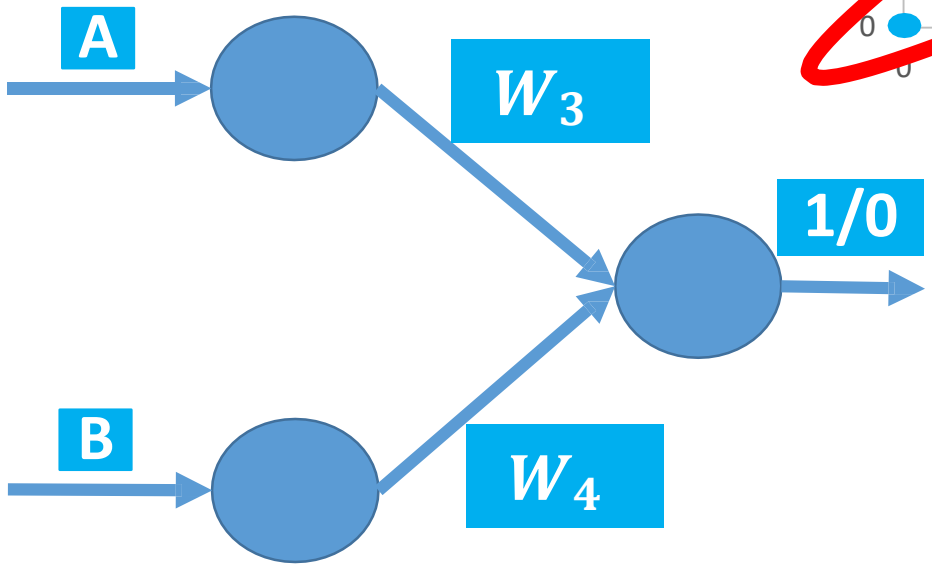

Input
Output
A
B
W_3
W_4
1/0

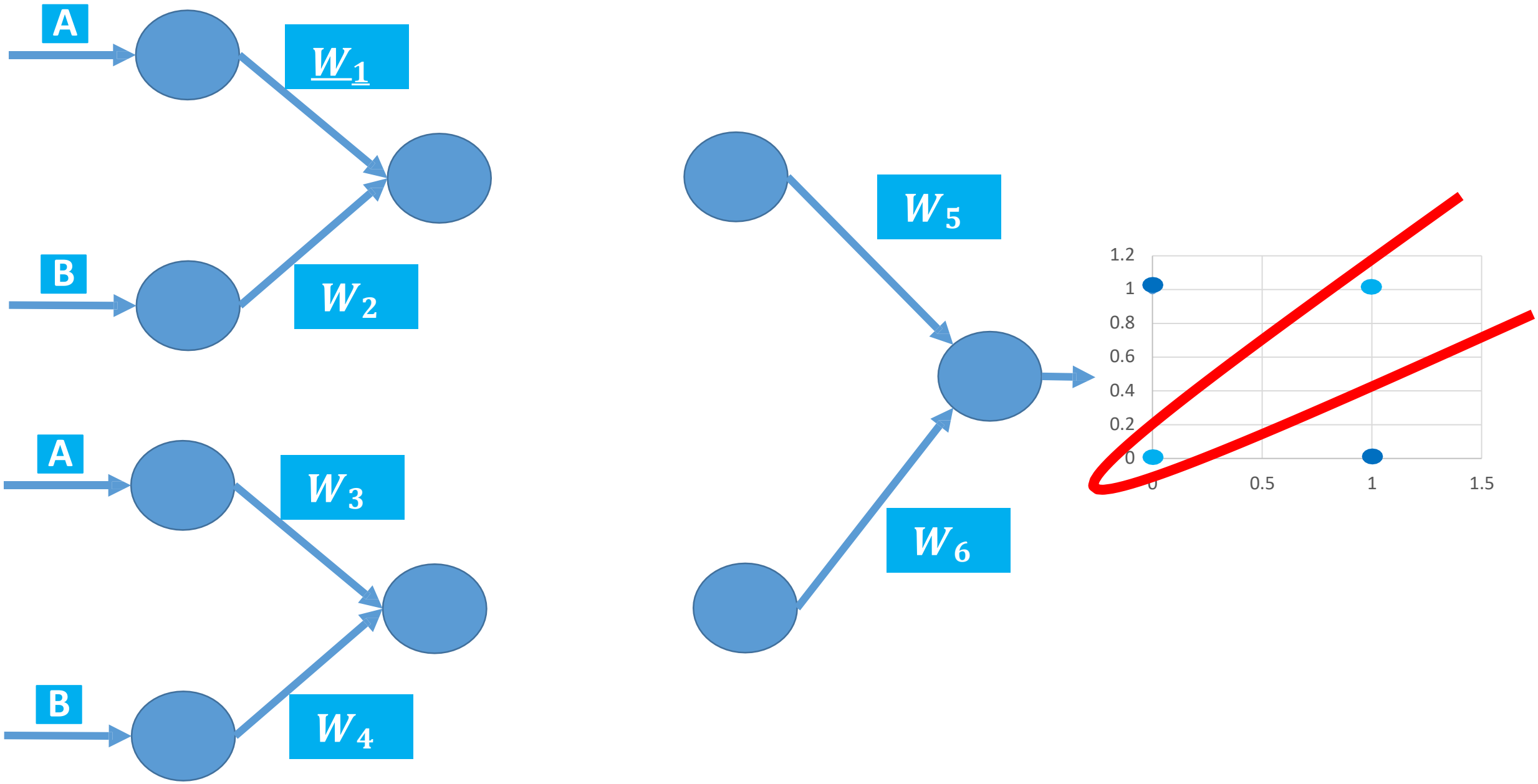

1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5

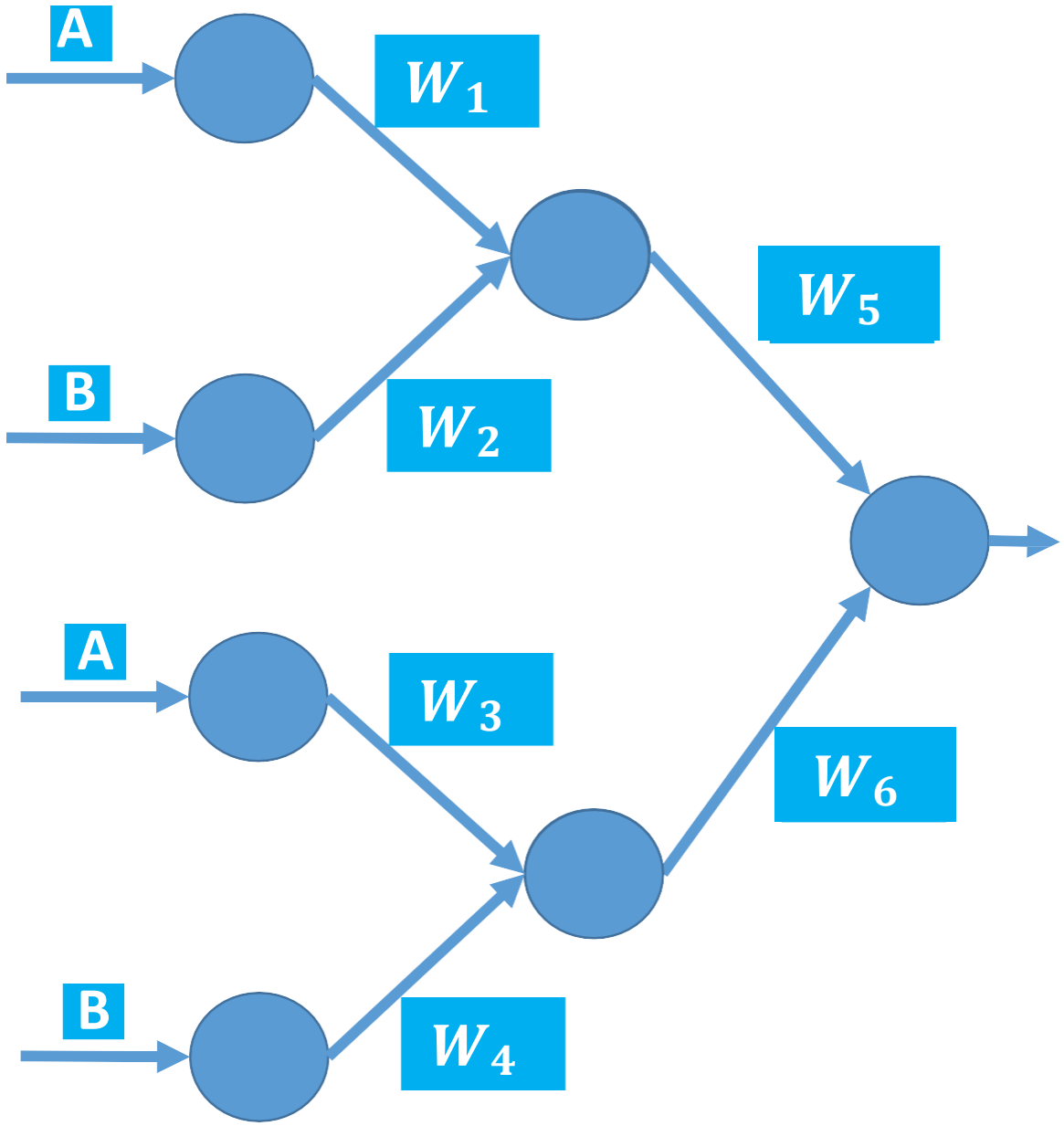

A
B
W_1
W_2
1/0

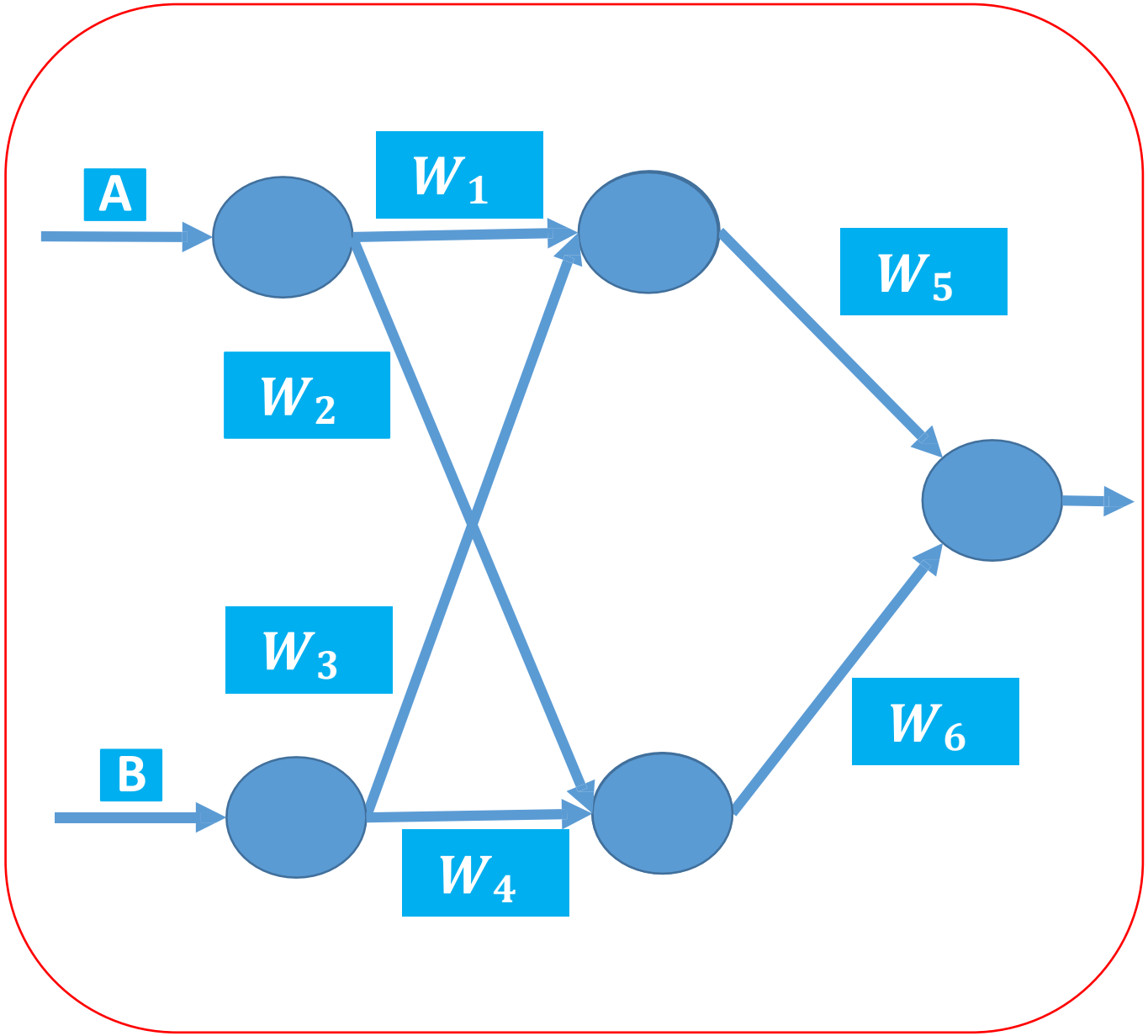

1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5


1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5

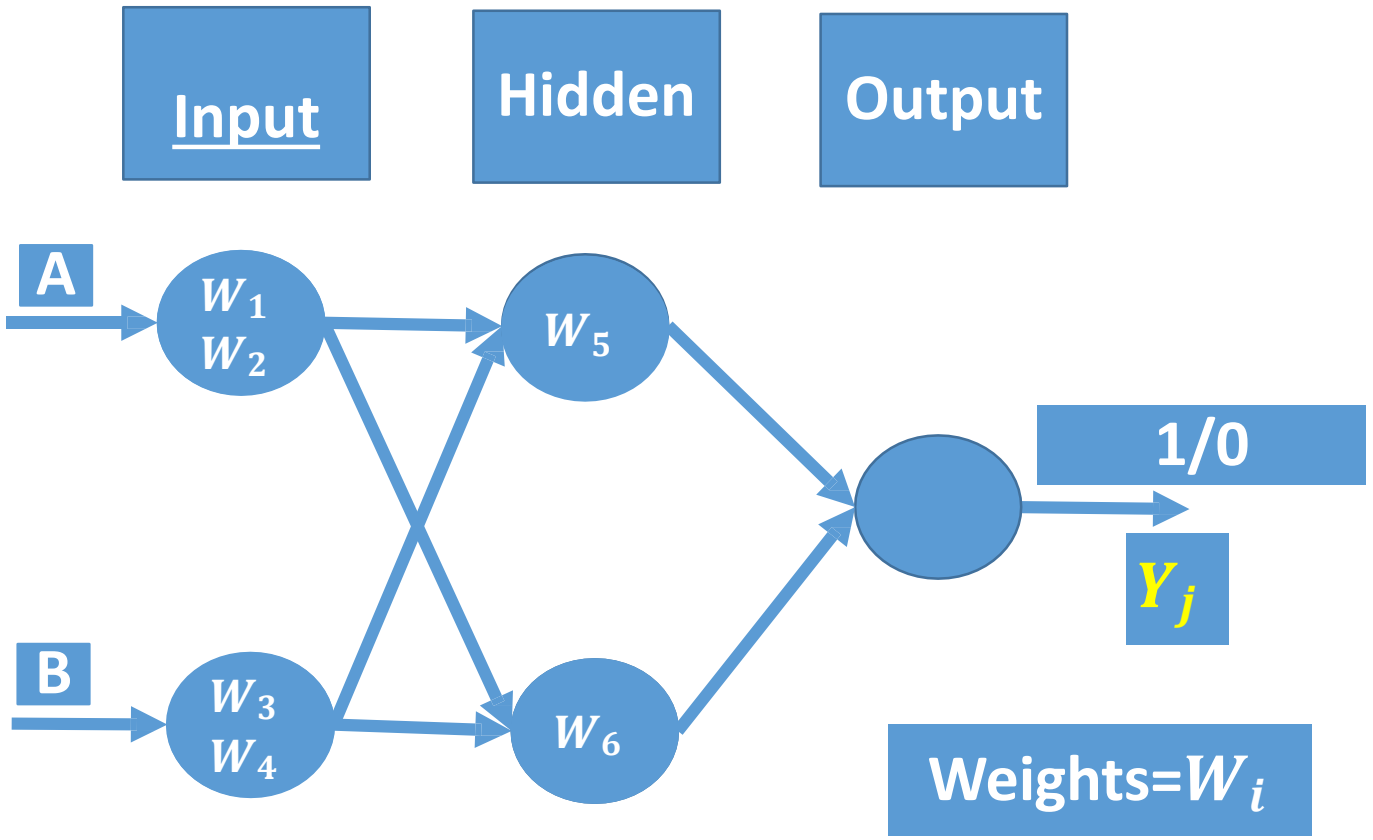

A
B
W_3
W_4
1/0

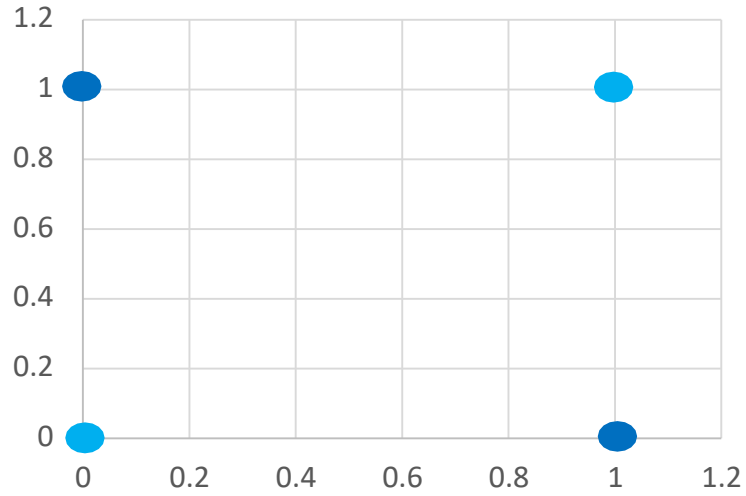

A
B
W_1
W_2
A
B
W_3
W_4
W_5
W_6
1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5


A
B
A
B
W_1
W_2
W_3
W_4
W_5
W_6


A
B
W_1
W_2
W_3
W_4
W_5
W_6

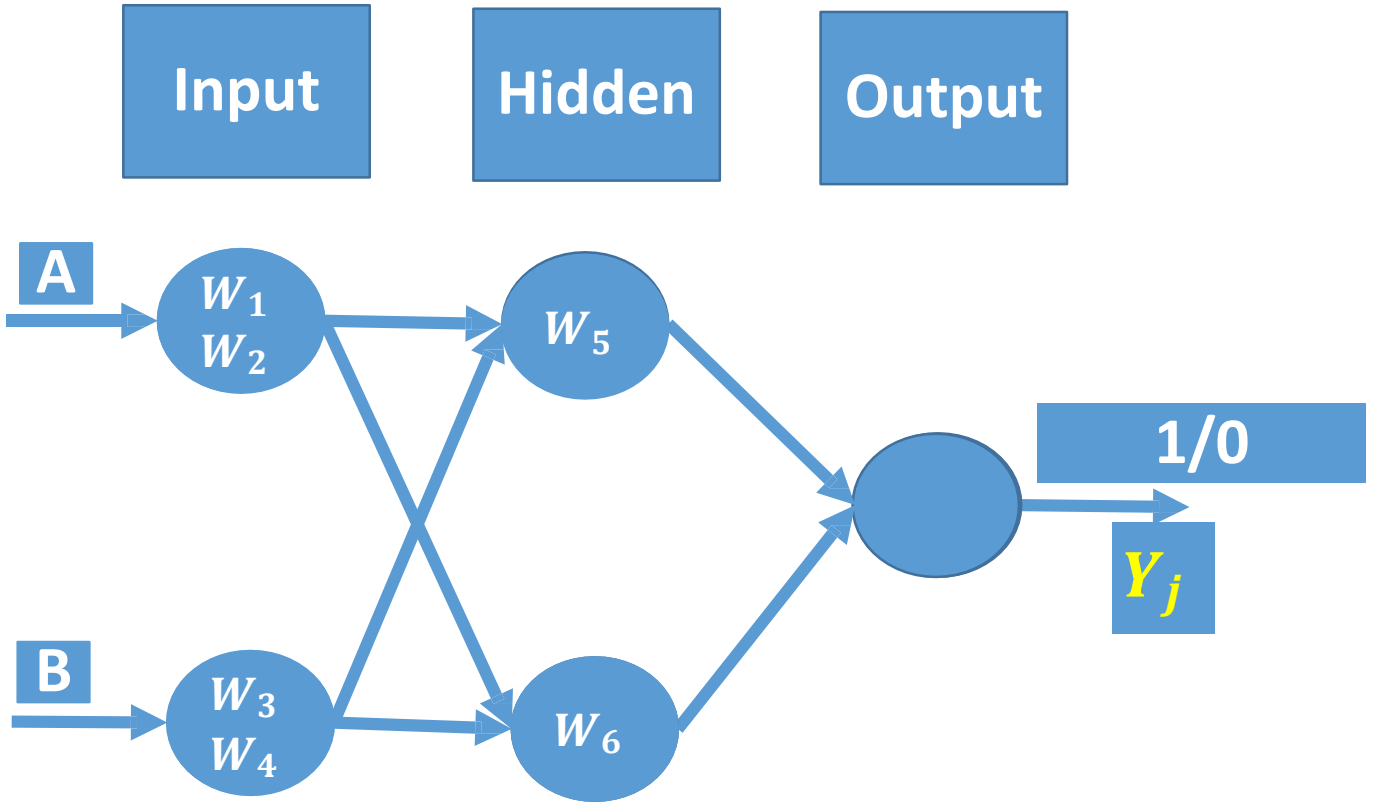
	A	B
1	1	0
	0	1
0	0	0
	1	1

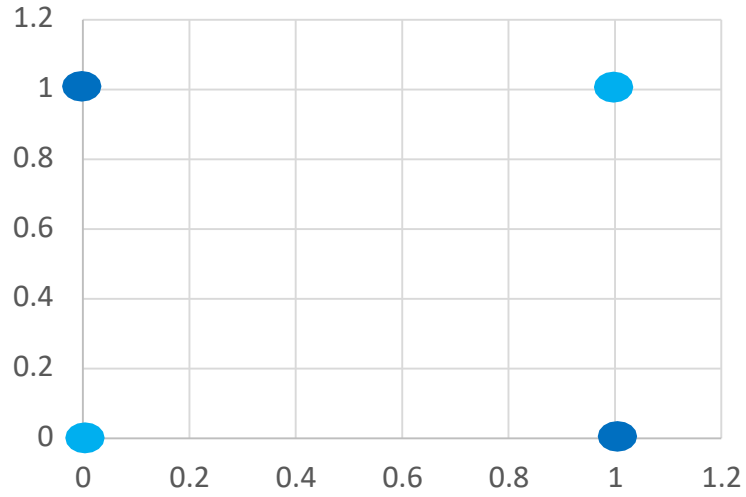




Activation Function

	A	B
1	1	0
	0	1
0	0	0
	1	1

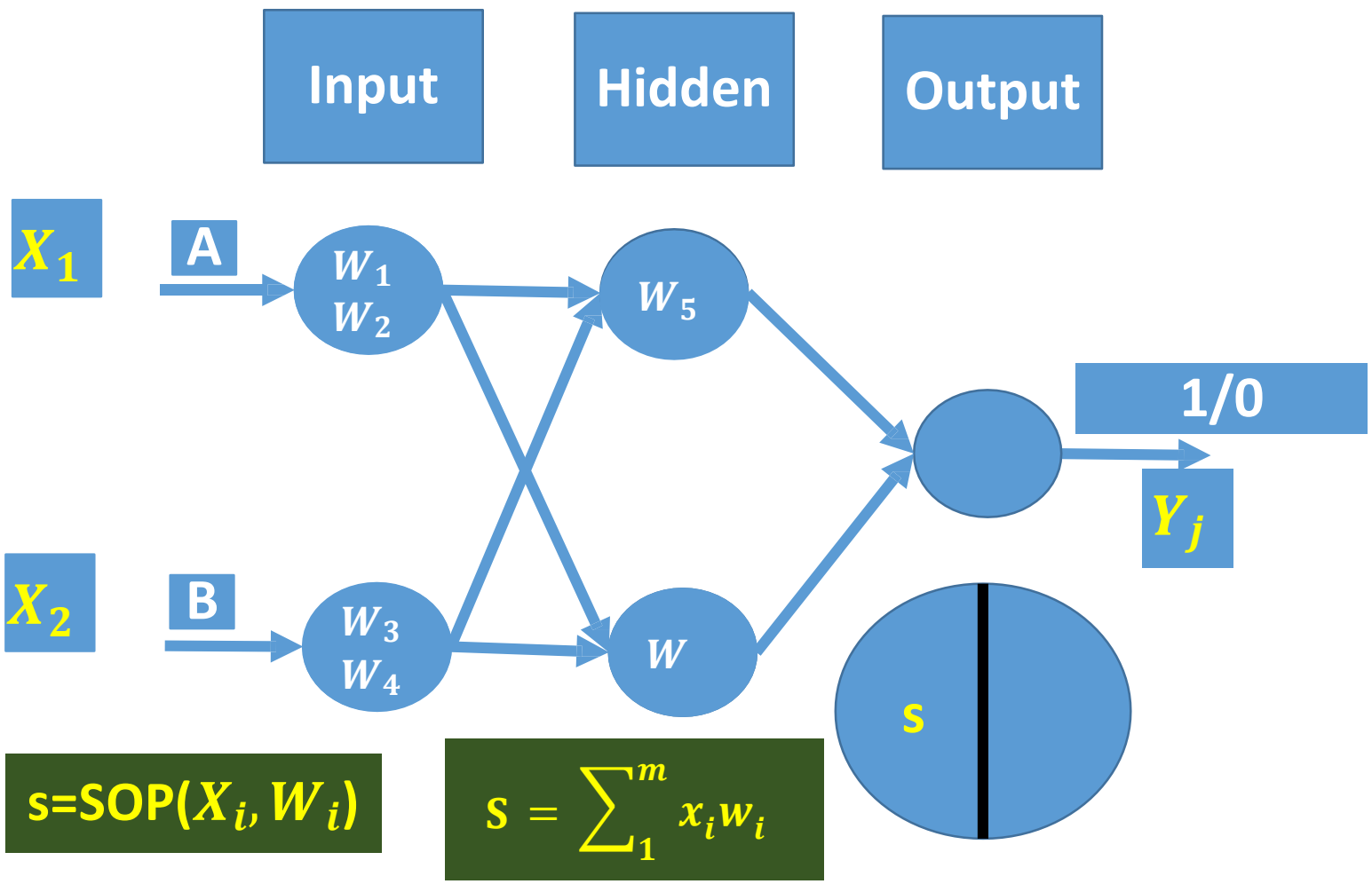




Activation Function Inputs

X_i=Inputs

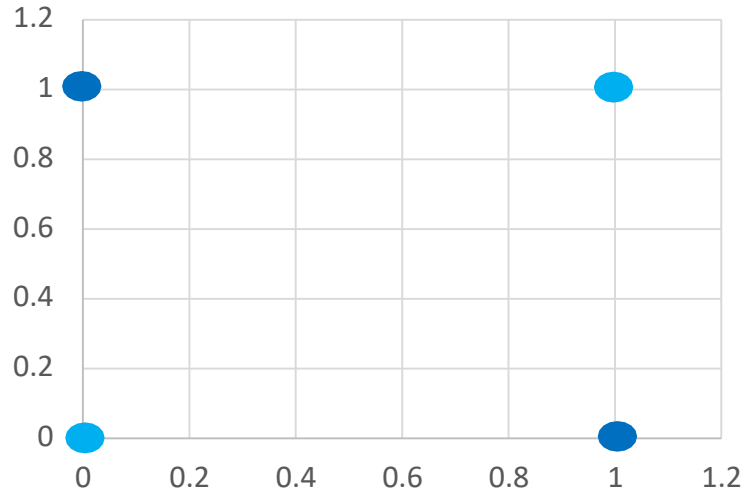
W_i=Weights



s=SOP(X_i, W_i)

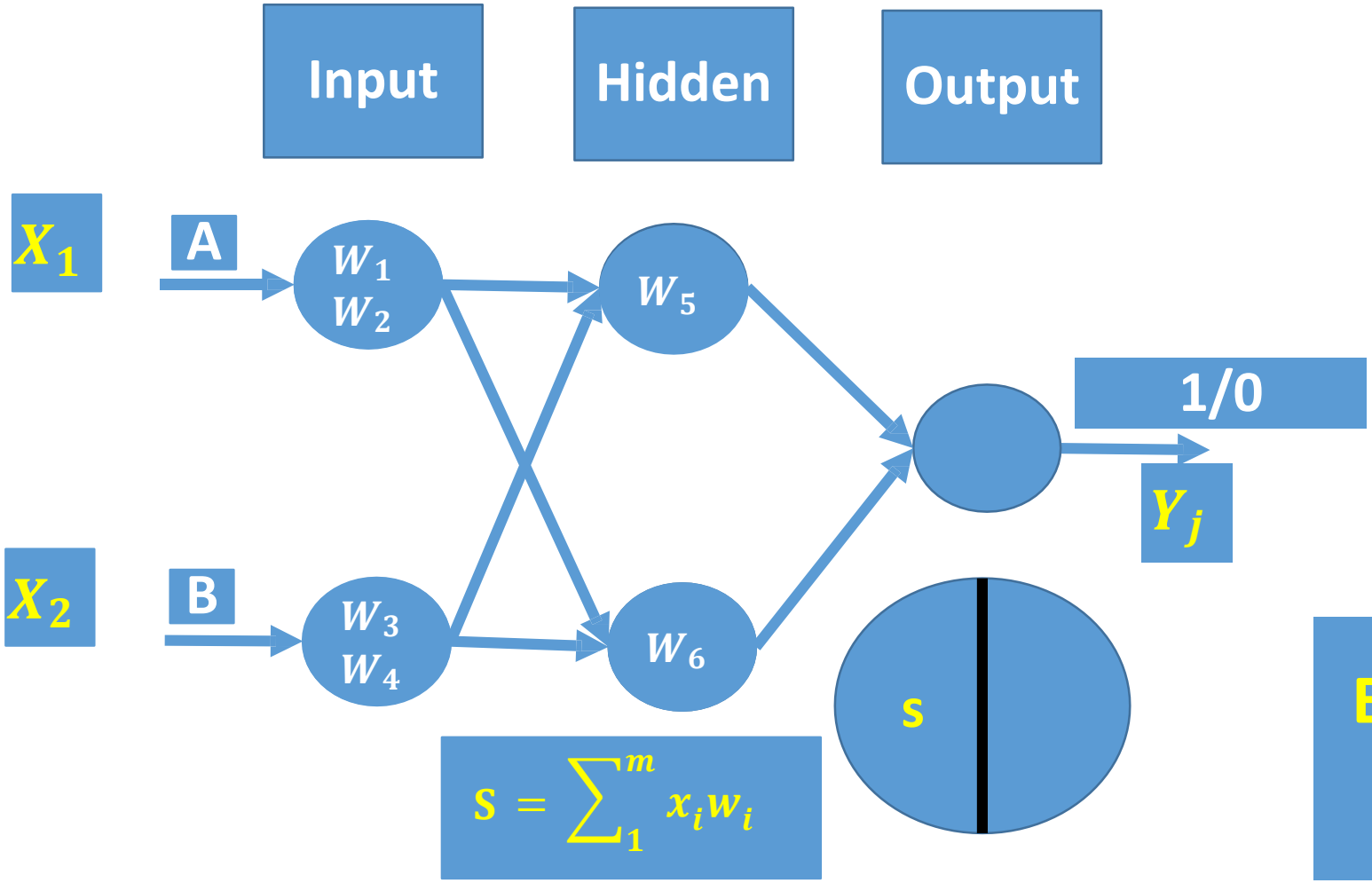
$$S = \sum_1^m x_i w_i$$

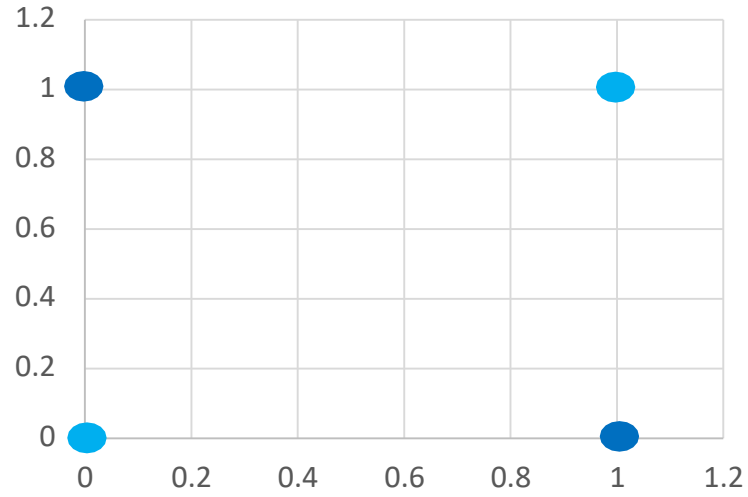
	A	B
1	1	0
	0	1
0	0	0
	1	1



Activation Function Inputs

	A	B
1	1	0
	0	1
0	0	0
	1	1

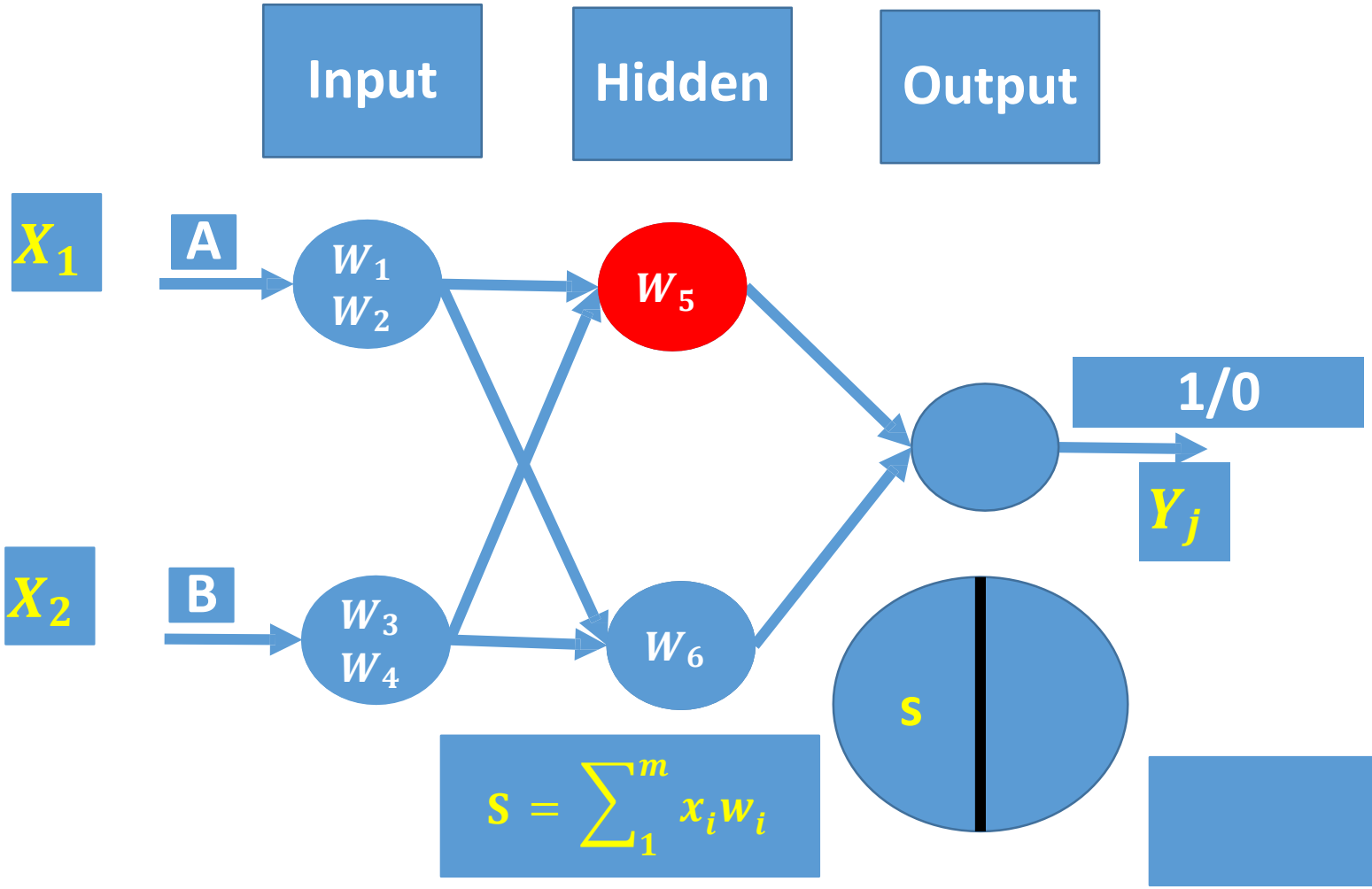


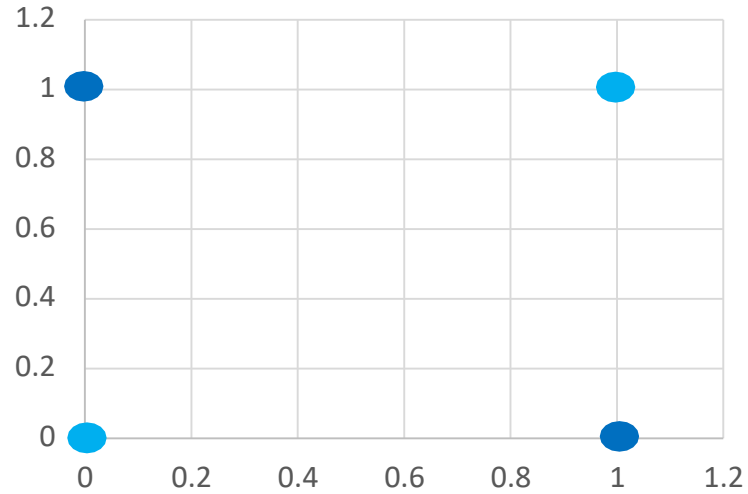


Each Hidden/Output Layer Neuron has its SOP.

Activation Function Inputs

	A	B
1	1	0
	0	1
0	0	0
	1	1

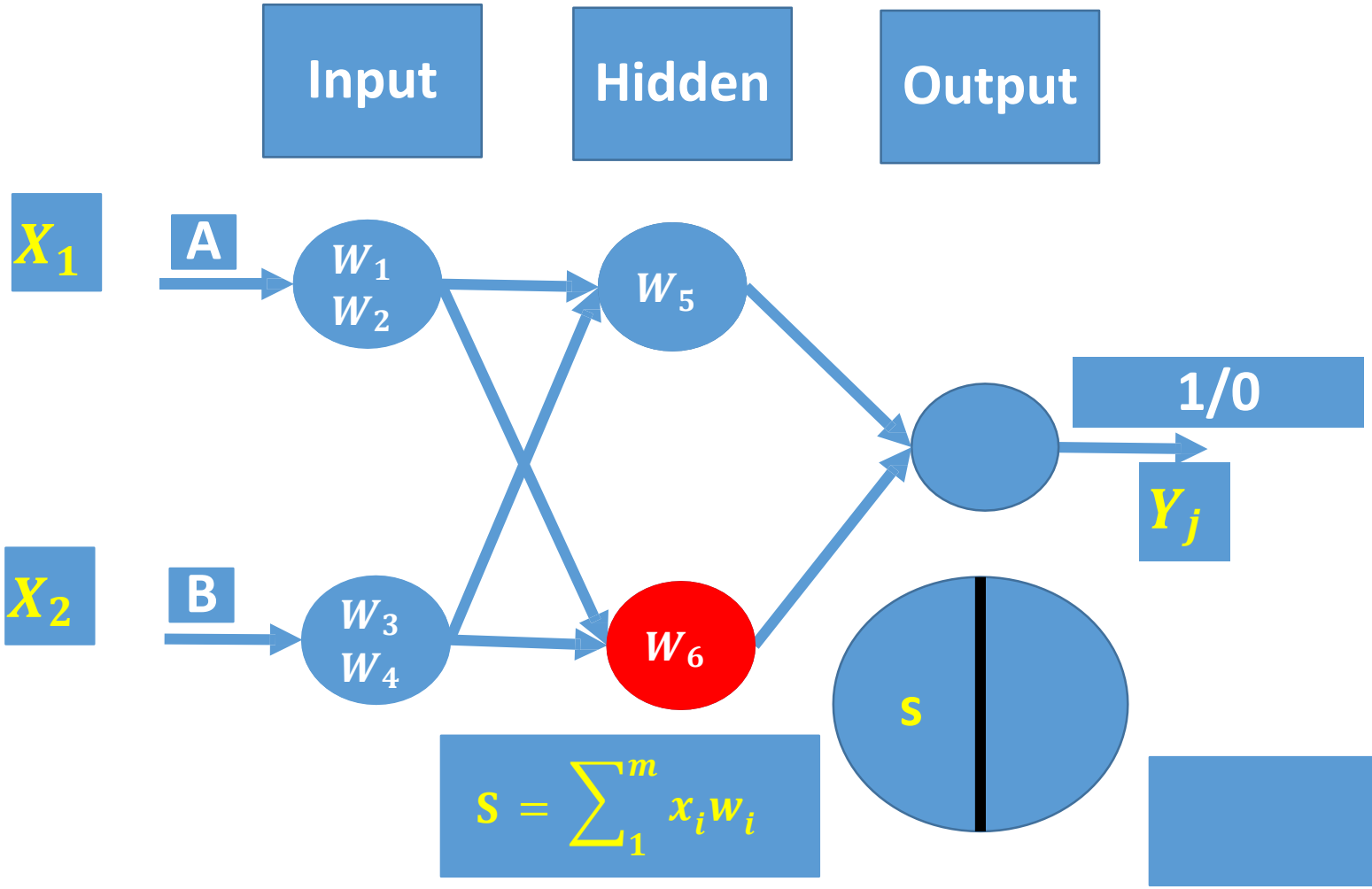


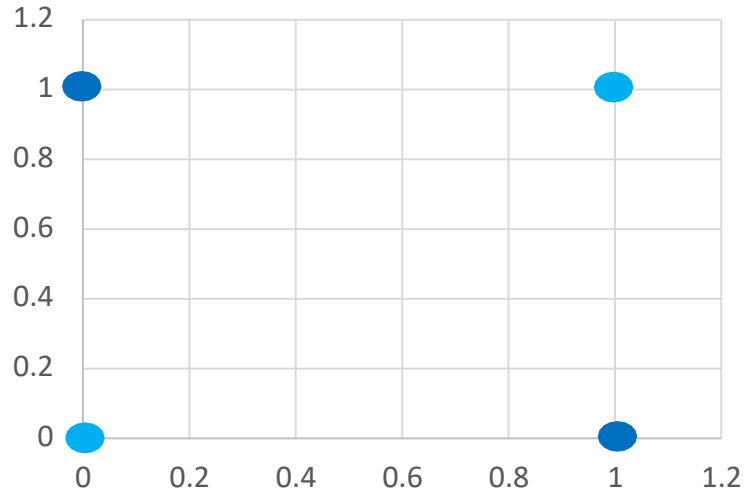


$$S_1=(X_1W_1+X_2W_3)$$

Activation Function Inputs

	A	B
1	1	0
	0	1
0	0	0
	1	1

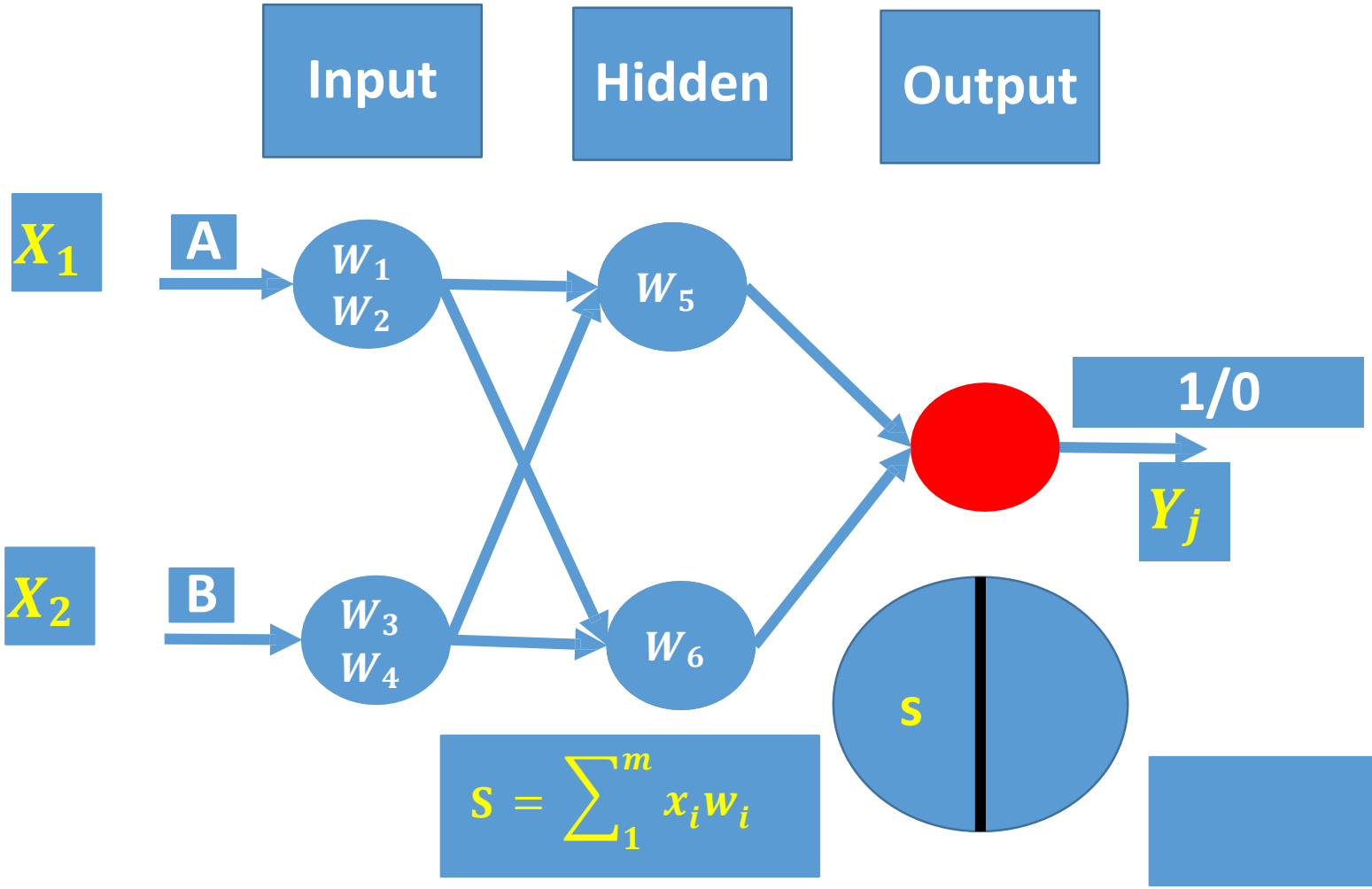


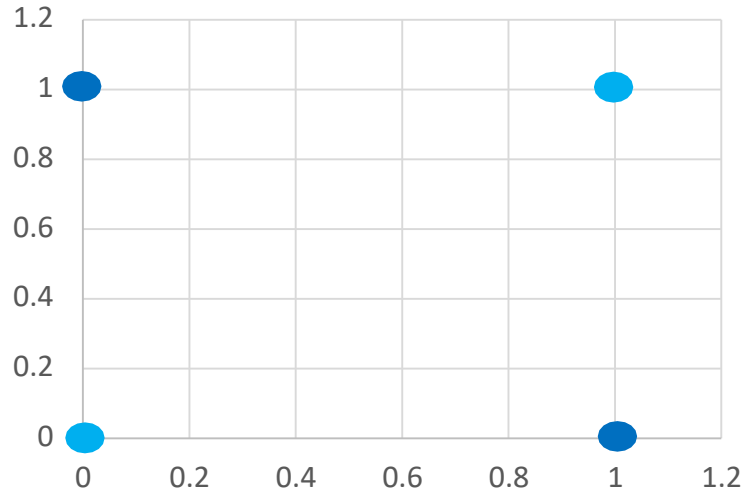


$$S_2=(X_1W_2+X_2W_4)$$

Activation Function Inputs

	A	B
1	1	0
	0	1
0	0	0
	1	1

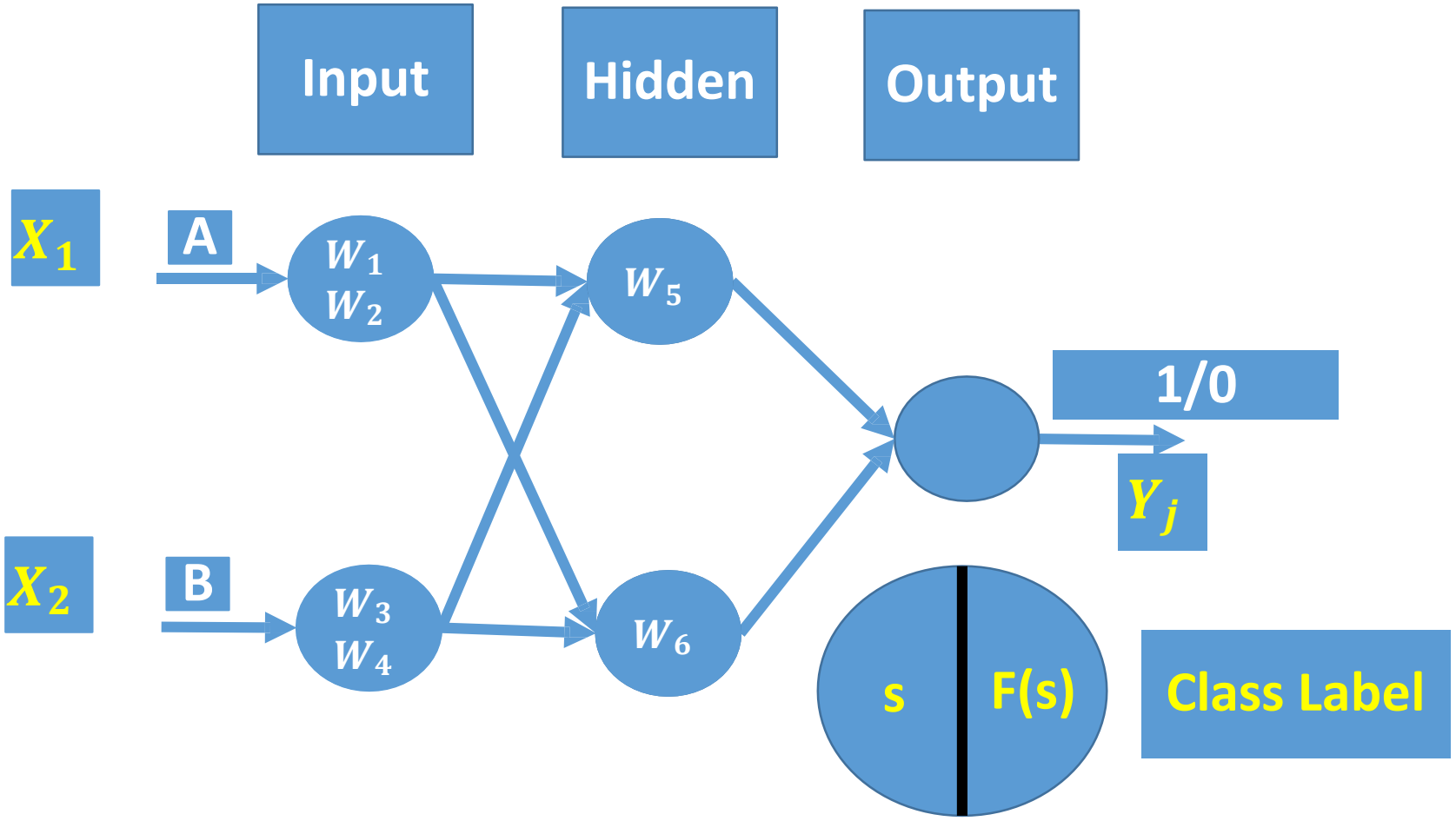


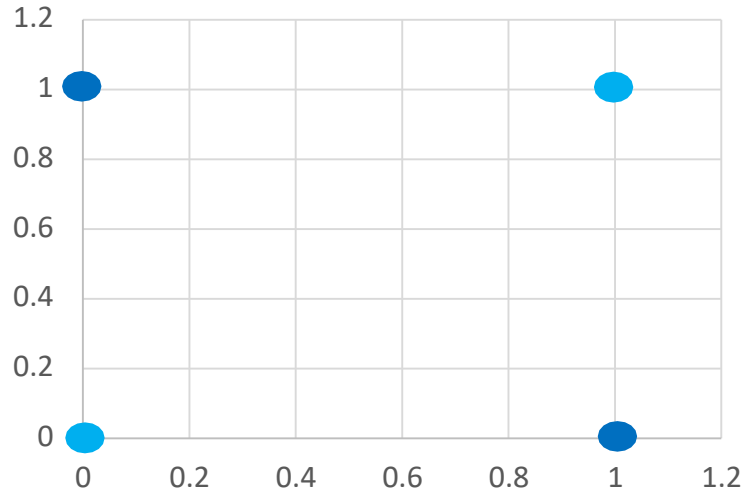


$$S_3=(S_1W_5+S_2W_6)$$

Activation Function Outputs

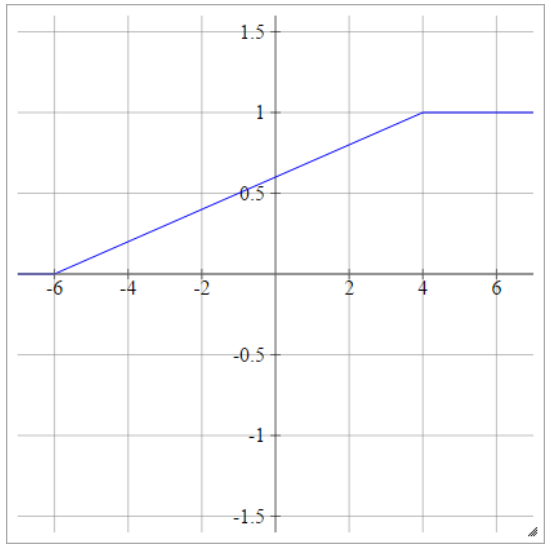
	A	B
1	1	0
	0	1
0	0	0
	1	1





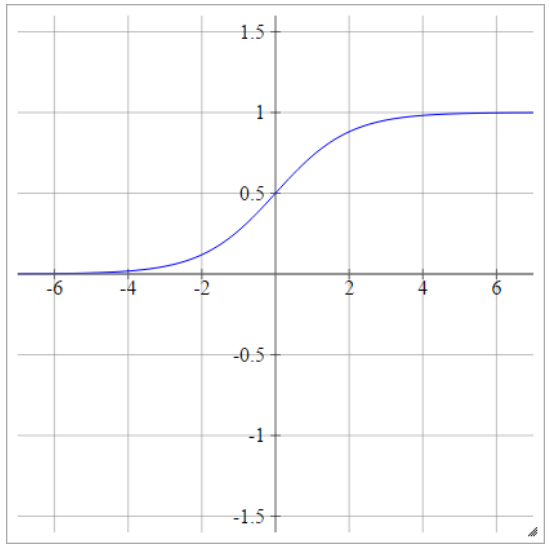
Activation Functions

Piecewise Linear



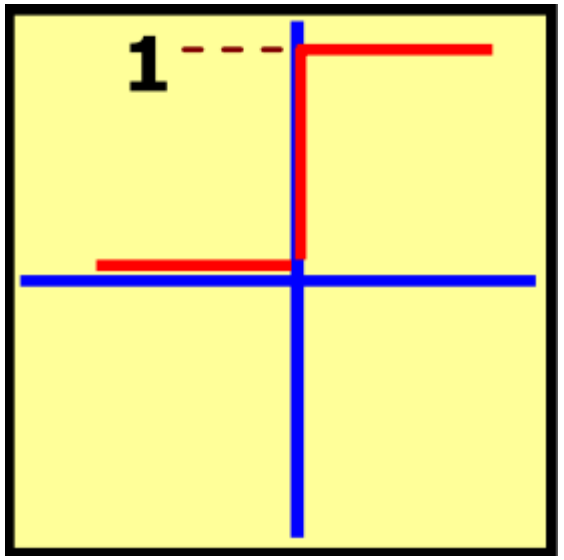
$$f(x) = \max(0, x) + \sum_{s=1}^{S} a_i^s \max(0, -x + b_i^s)$$

Sigmoid



$$a_j^i = \sigma(z_j^i) = \frac{1}{1 + \exp(-z_j^i)}$$

Binary



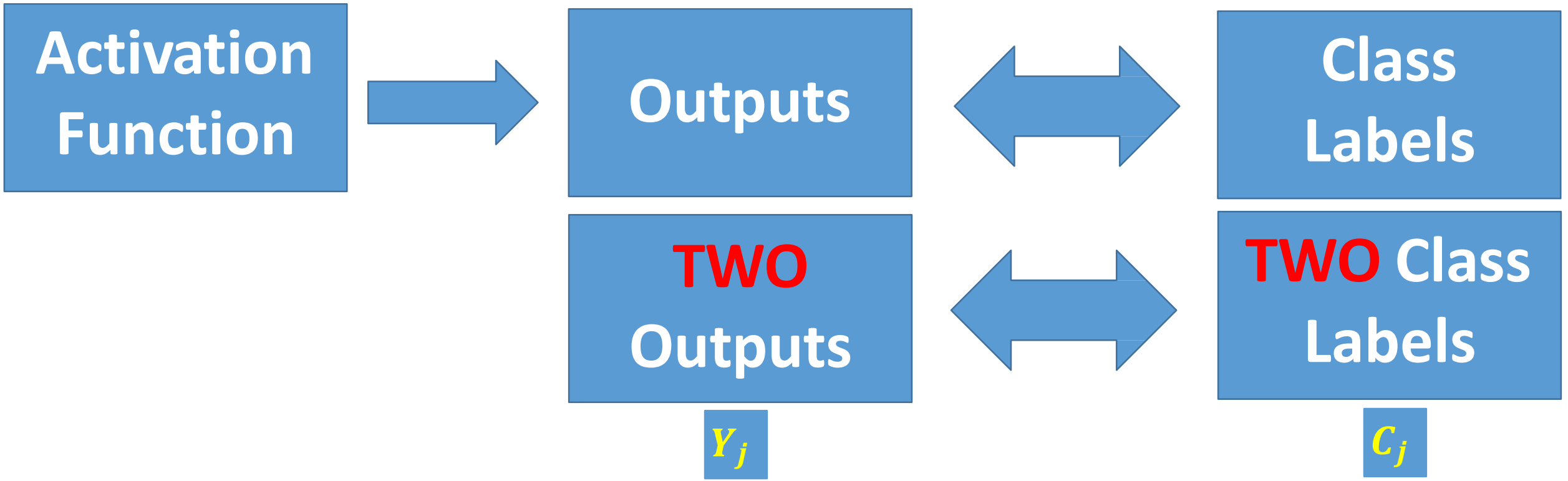
$$Y = f(I) = \begin{cases} 1 & \text{if } I \geq 0 \\ 0 & \text{if } I < 0 \end{cases}$$

Activation Functions

	A	B
1	1	0
	0	1
0	0	0
	1	1

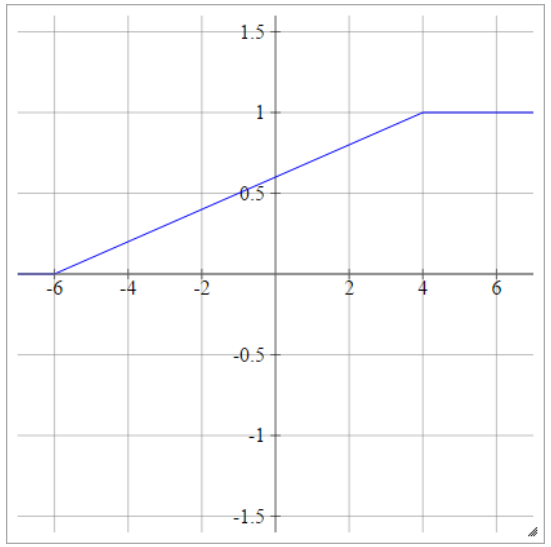
Which activation function to use?

One that gives two outputs.



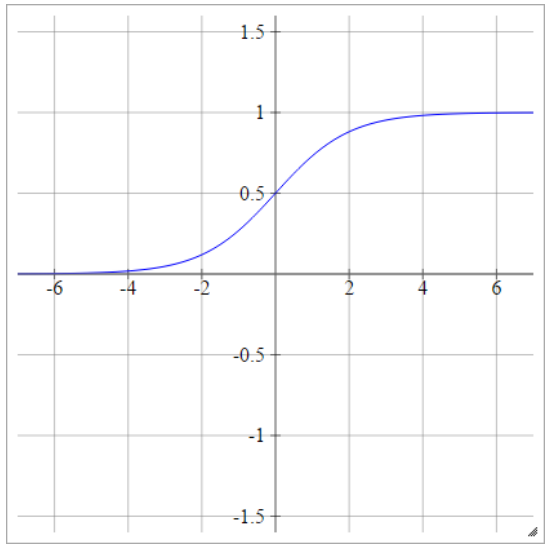
Activation Functions

Piecewise Linear



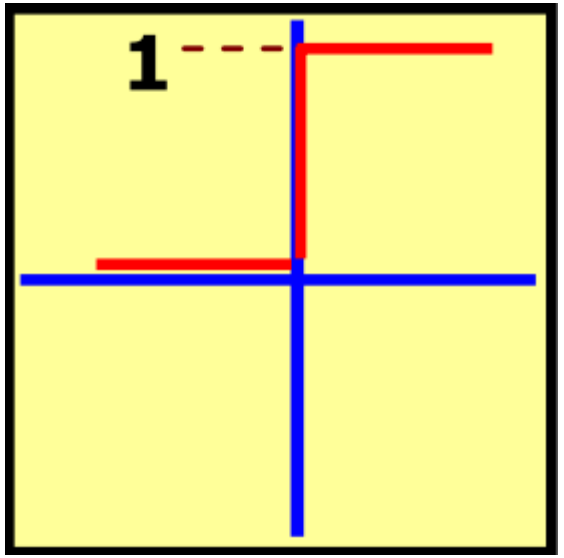
$$f(x) = \max(0, x) + \sum_{s=1}^{S} a_i^s \max(0, -x + b_i^s)$$

Sigmoid



$$a_j^i = \sigma(z_j^i) = \frac{1}{1 + \exp(-z_j^i)}$$

Binary

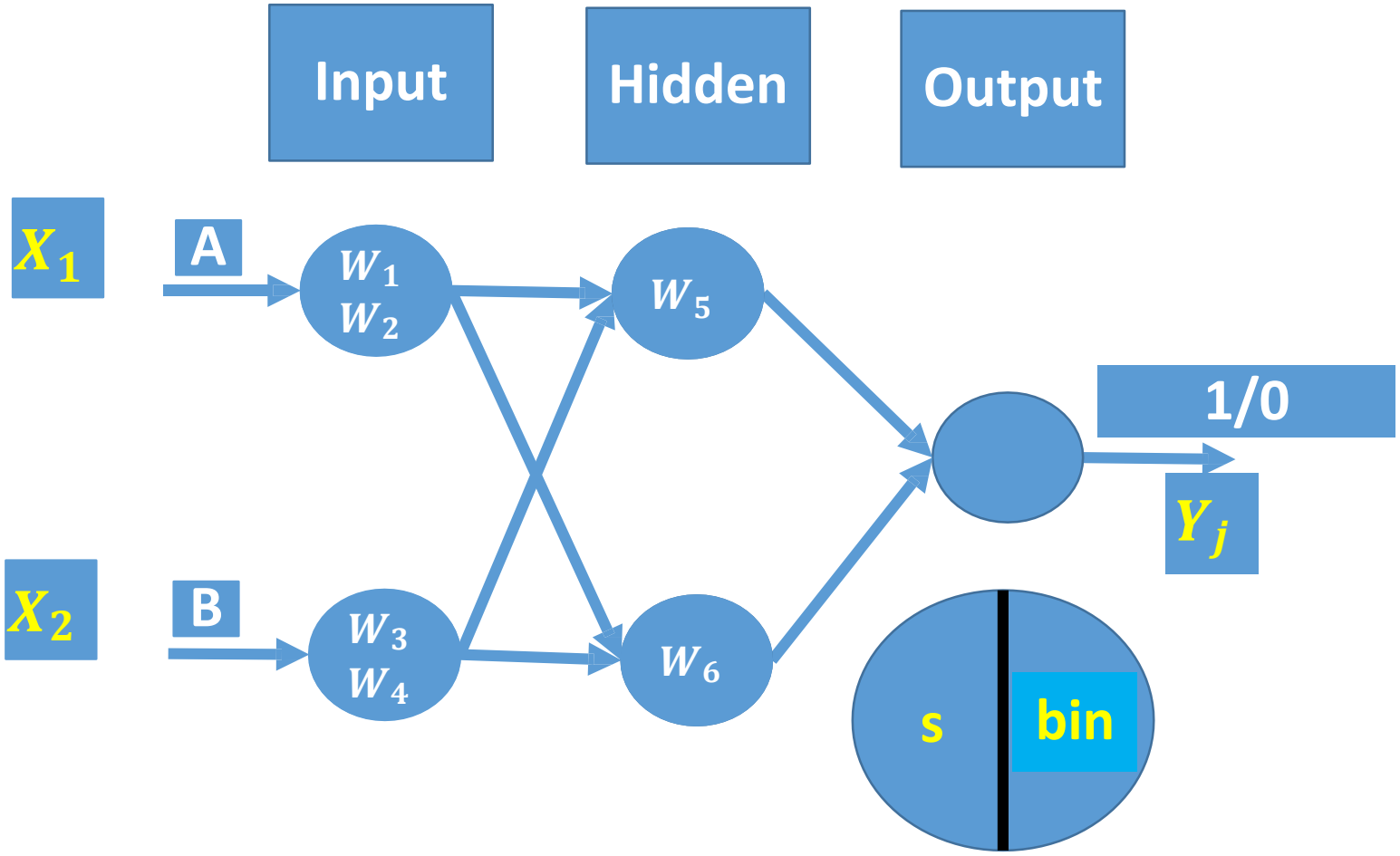


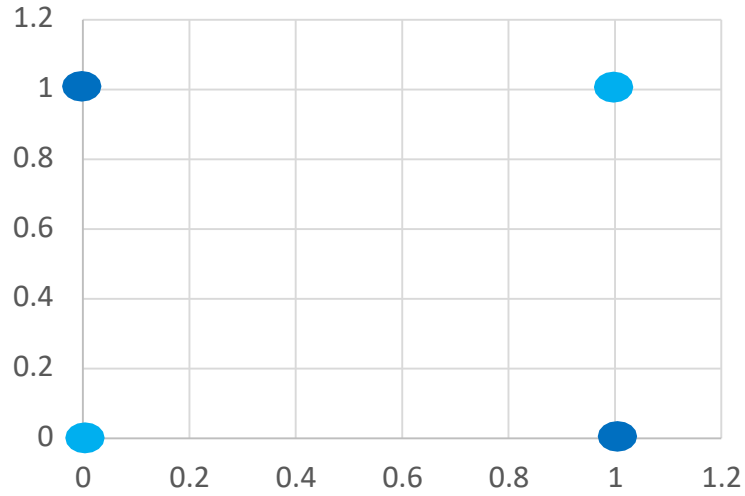
Y = f (I) = 1 if I ≥ 0

0 if I < 0

Activation Function

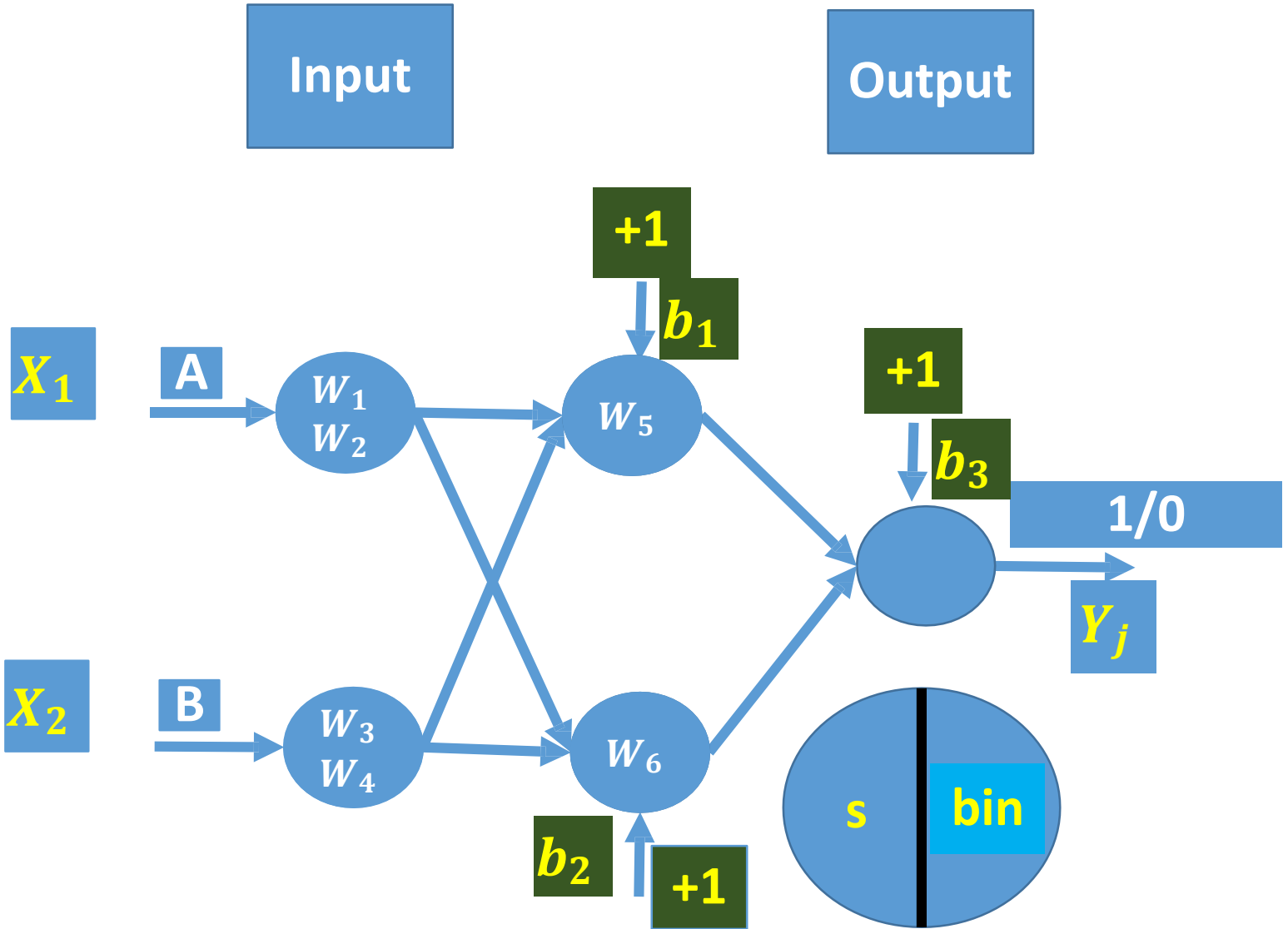
	A	B
1	1	0
	0	1
0	0	0
	1	1





All Bias Values

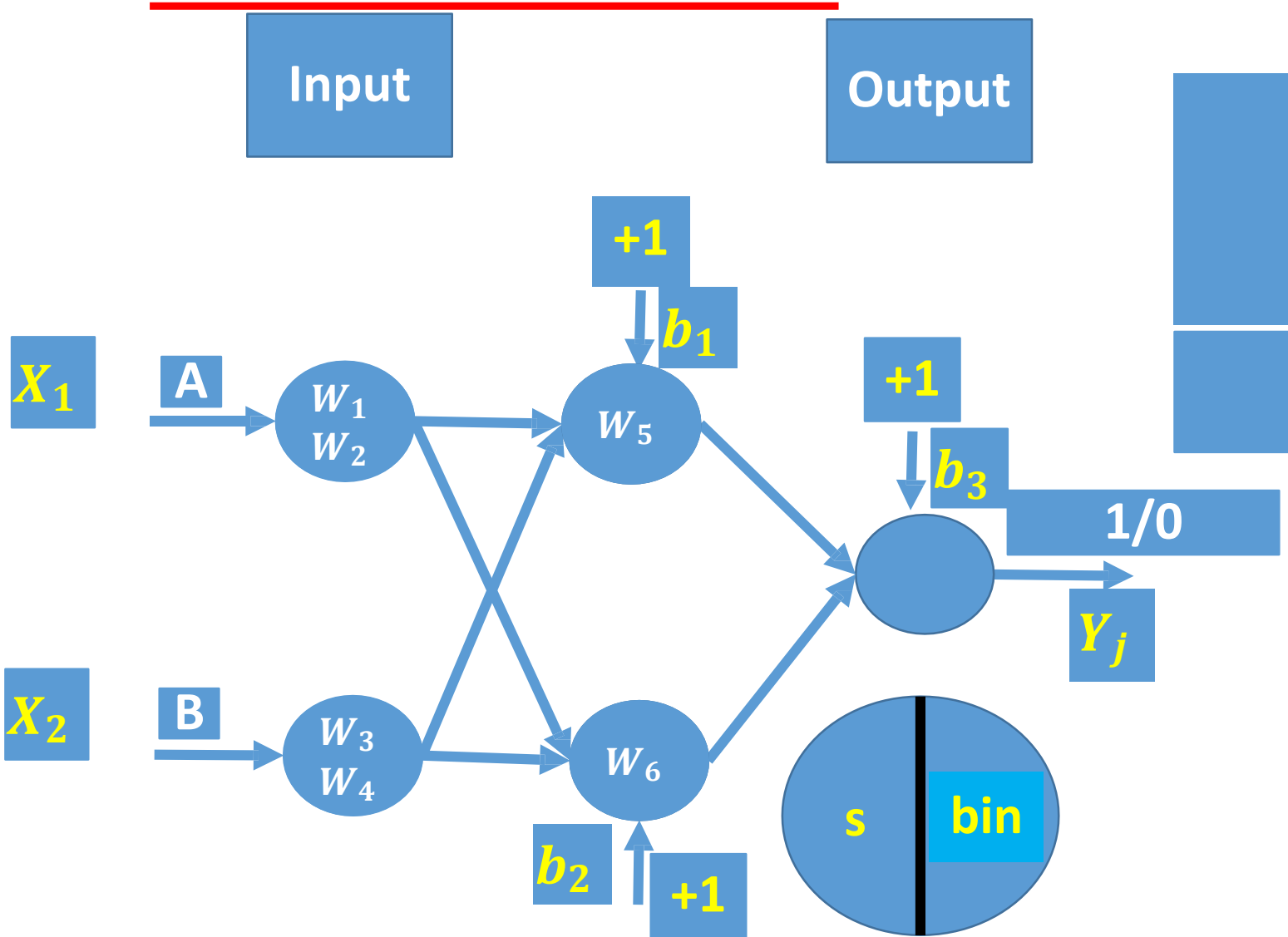
	A	B
1	1	0
	0	1
0	0	0
	1	1



Bias

Add Bias to SOP

	A	B
1	1	0
	0	1
0	0	0
	1	1



$$S_1=(X_1W_1+X_2W_3)$$

$$S_2=(X_1W_2+X_2W_4)$$

$$S_3=(S_1W_5+S_2W_6)$$

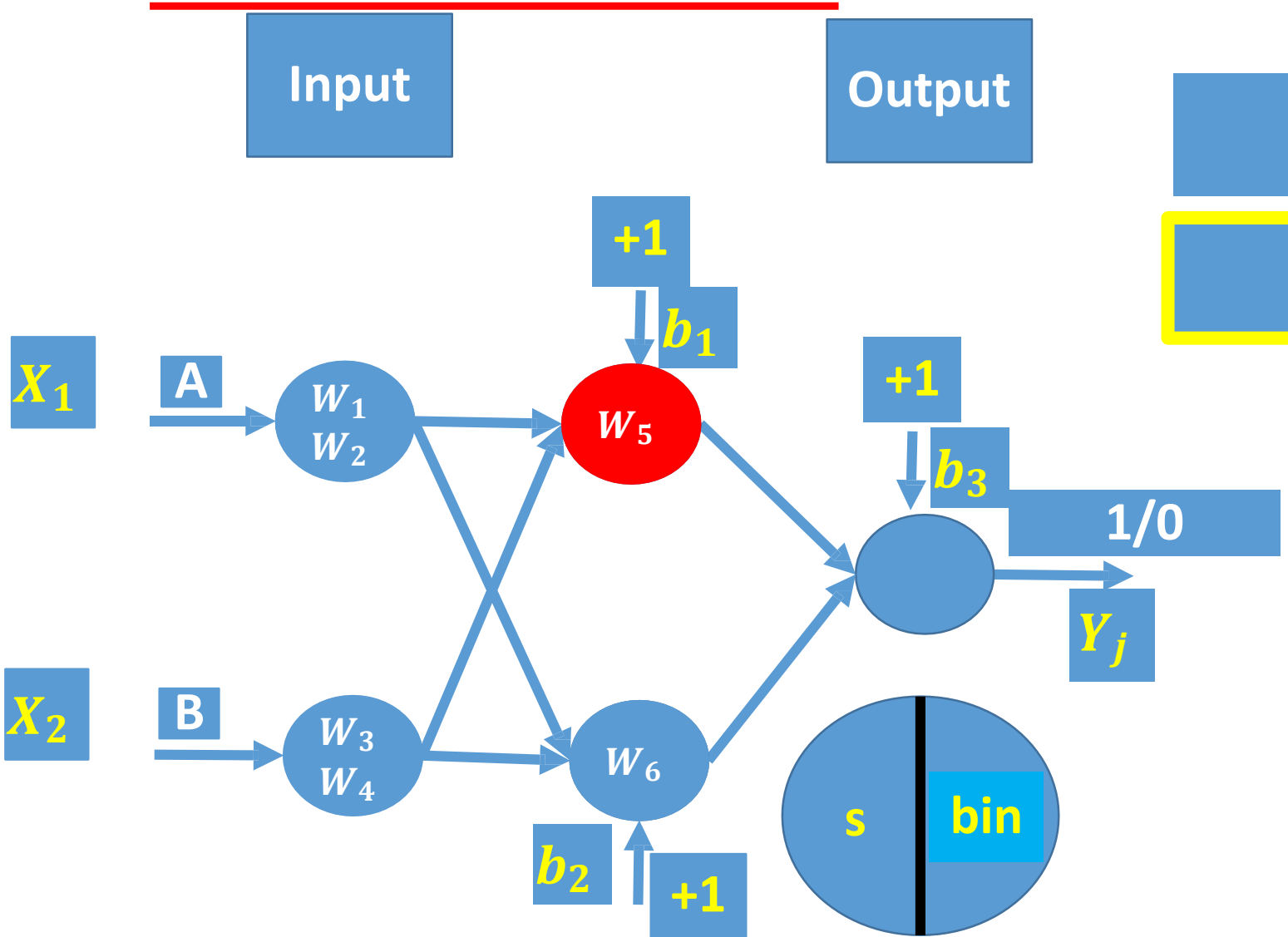
Bias

Add Bias to SOP

	A	B
1	1	0
	0	1
0	0	0
	1	1

$$S_1=(X_1W_1+X_2W_3)$$

$$S_1=(+1b_1+X_1W_1+X_2W_3)$$



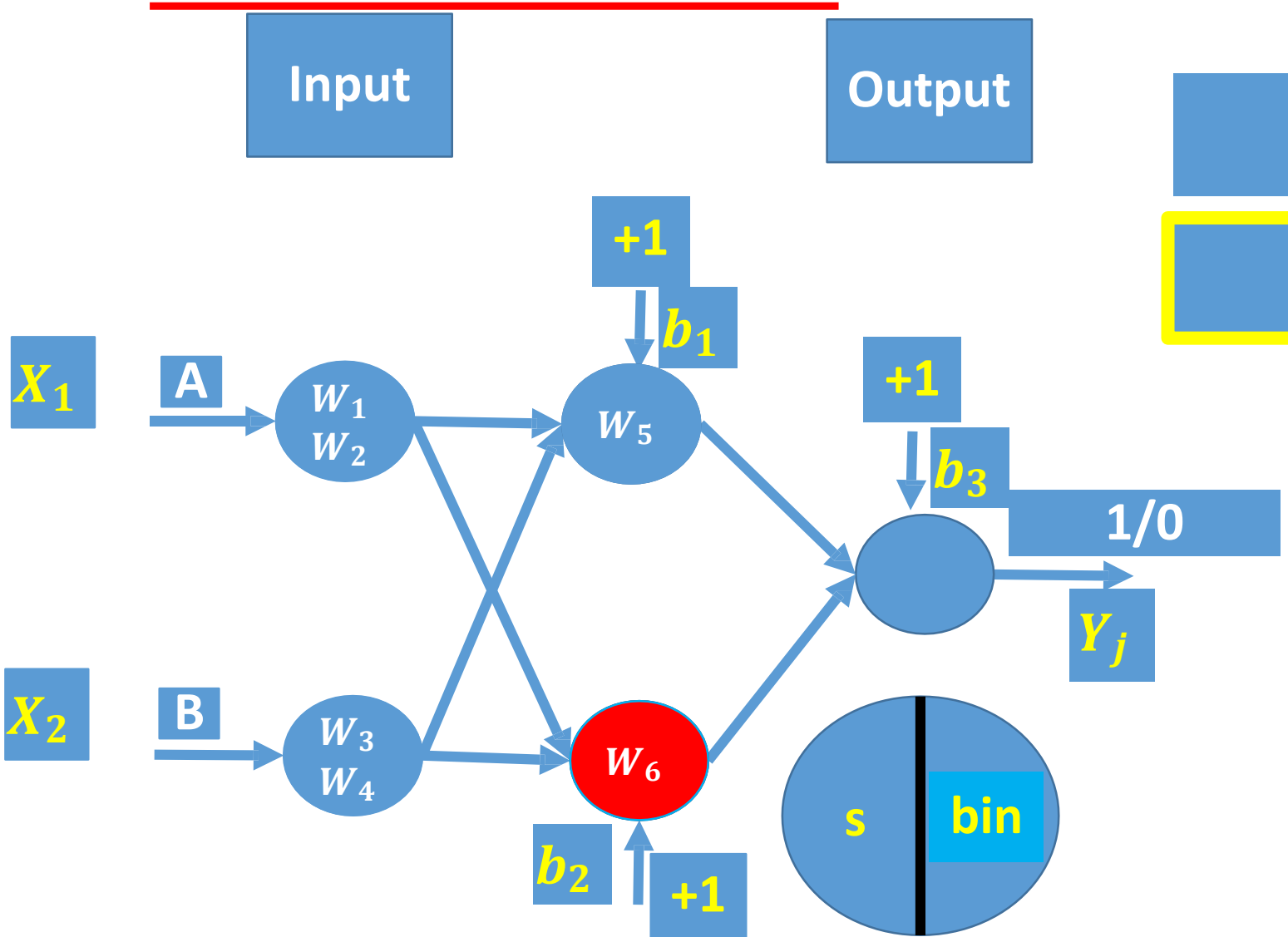
Bias

Add Bias to SOP

	A	B
1	1	0
	0	1
0	0	0
	1	1

$S_2=(X_1W_2+X_2W_4)$

$S_2=(+1b_2+X_1W_2+X_2W_4)$



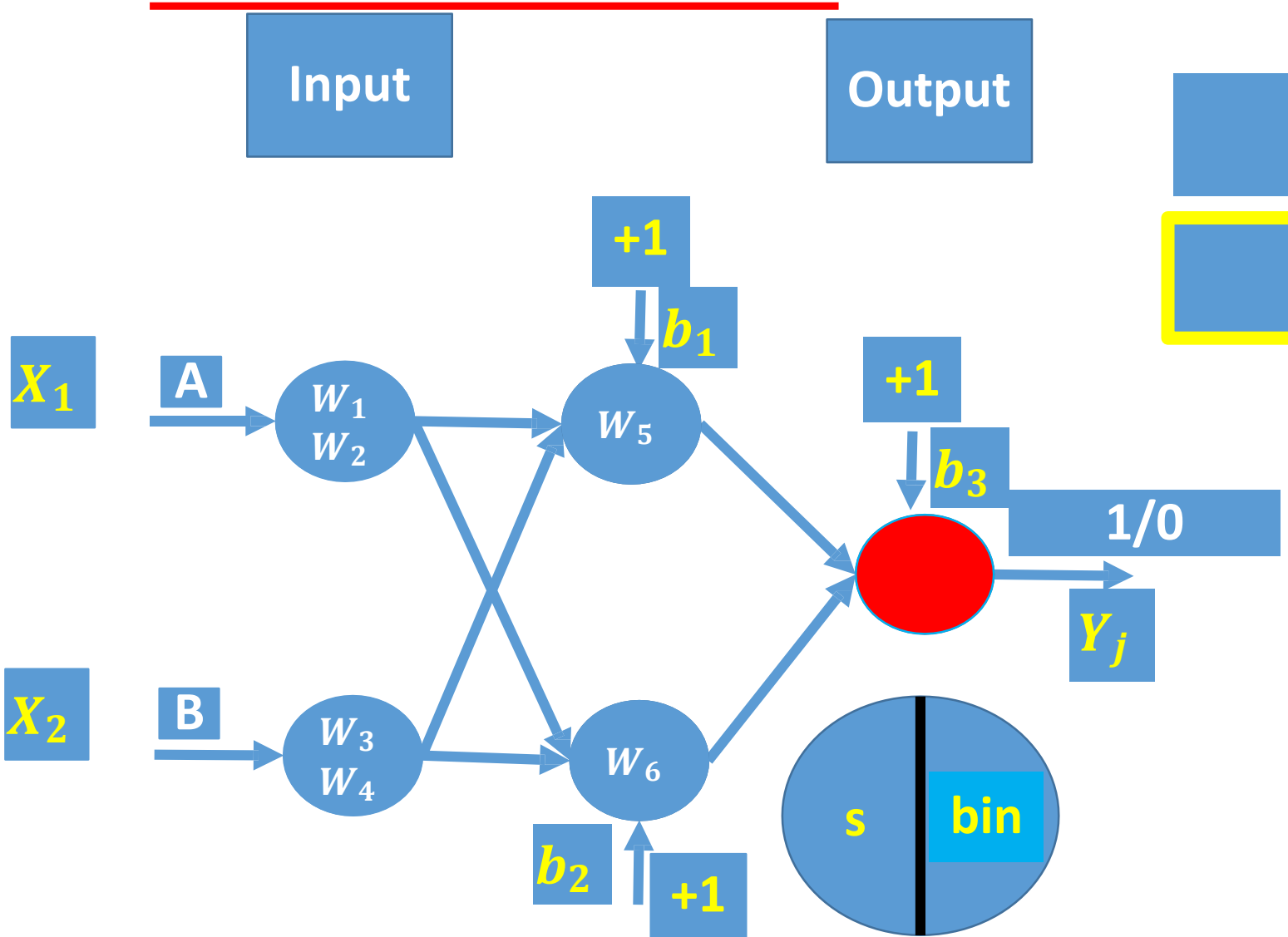
Bias

Add Bias to SOP

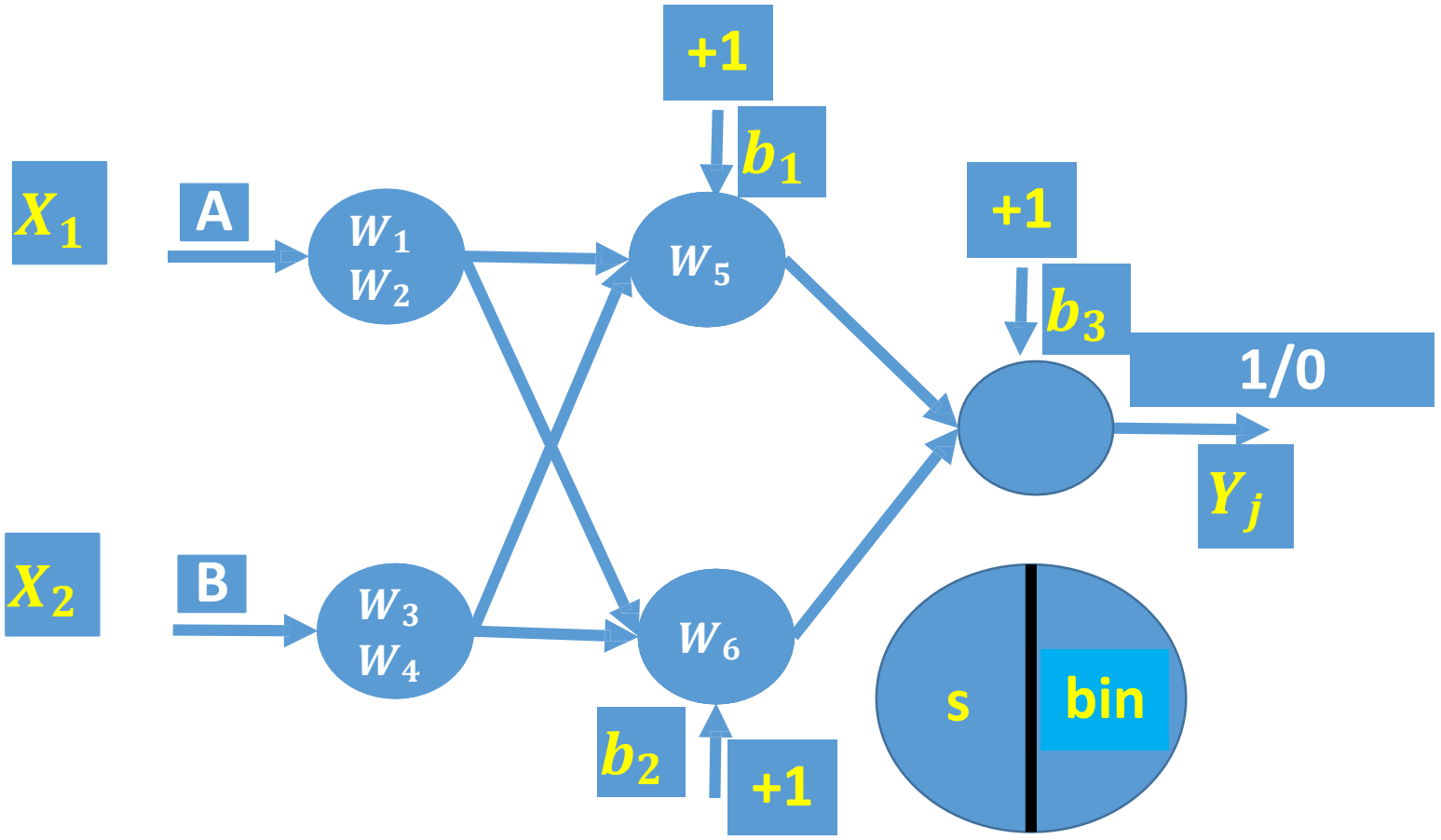
	A	B
1	1	0
	0	1
0	0	0
	1	1

$S_3=(S_1W_5+S_2W_6)$

$S_3=(+1b_3+S_1W_5+S_2W_6)$



Learning Rate



$$0 \leq \eta \leq 1$$

Other Parameters

Step n

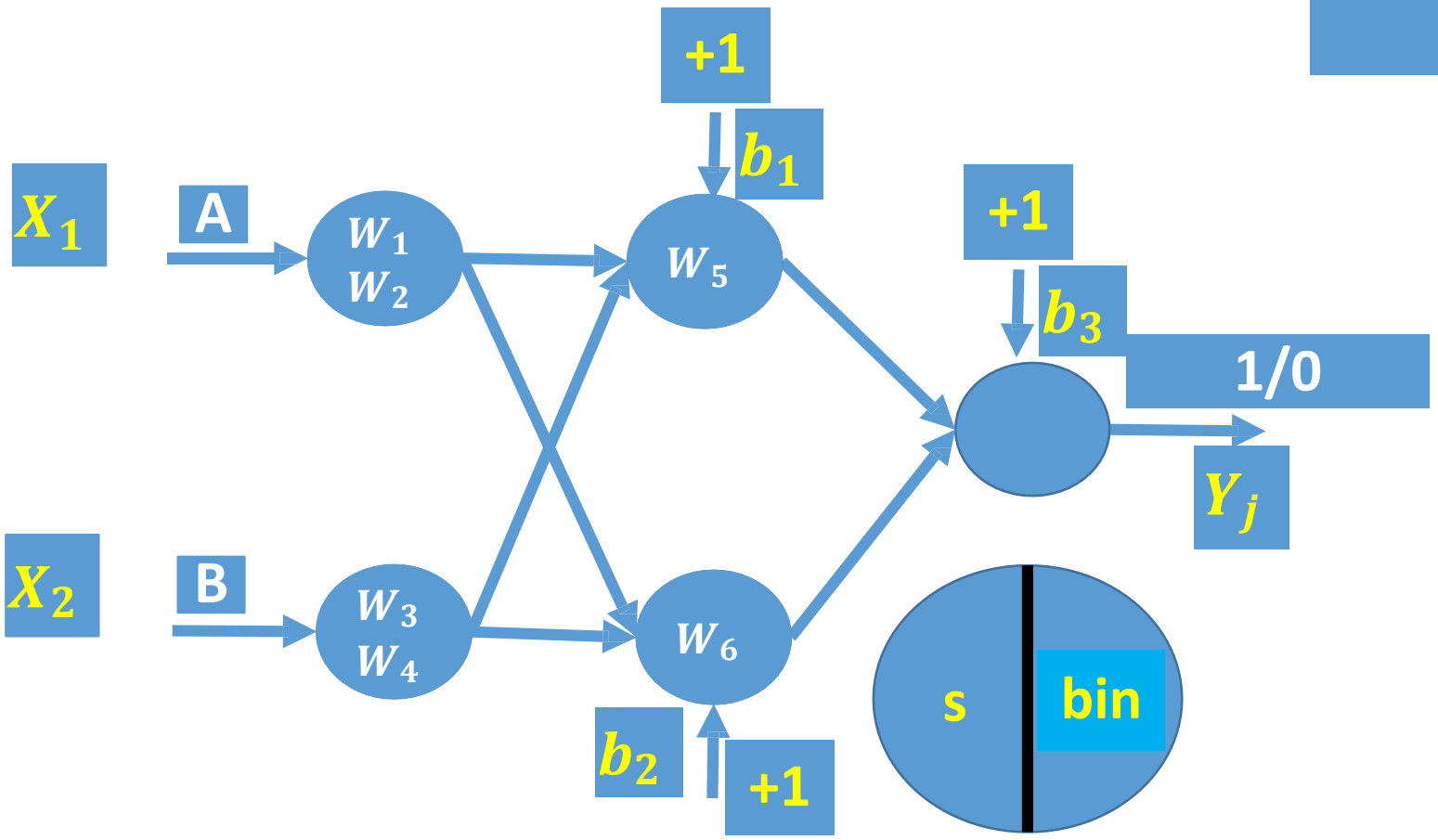
$$n = 0, 1, 2, \ldots$$

$$s=(X_0W_0+X_1W_1+X_2W_2+\ldots)$$

$$0 \leq \eta \leq 1$$

$$X(n)=(X_0, X_1, X_2, \ldots)$$

$$W(n)=(W_0, W_1, W_2, \ldots)$$

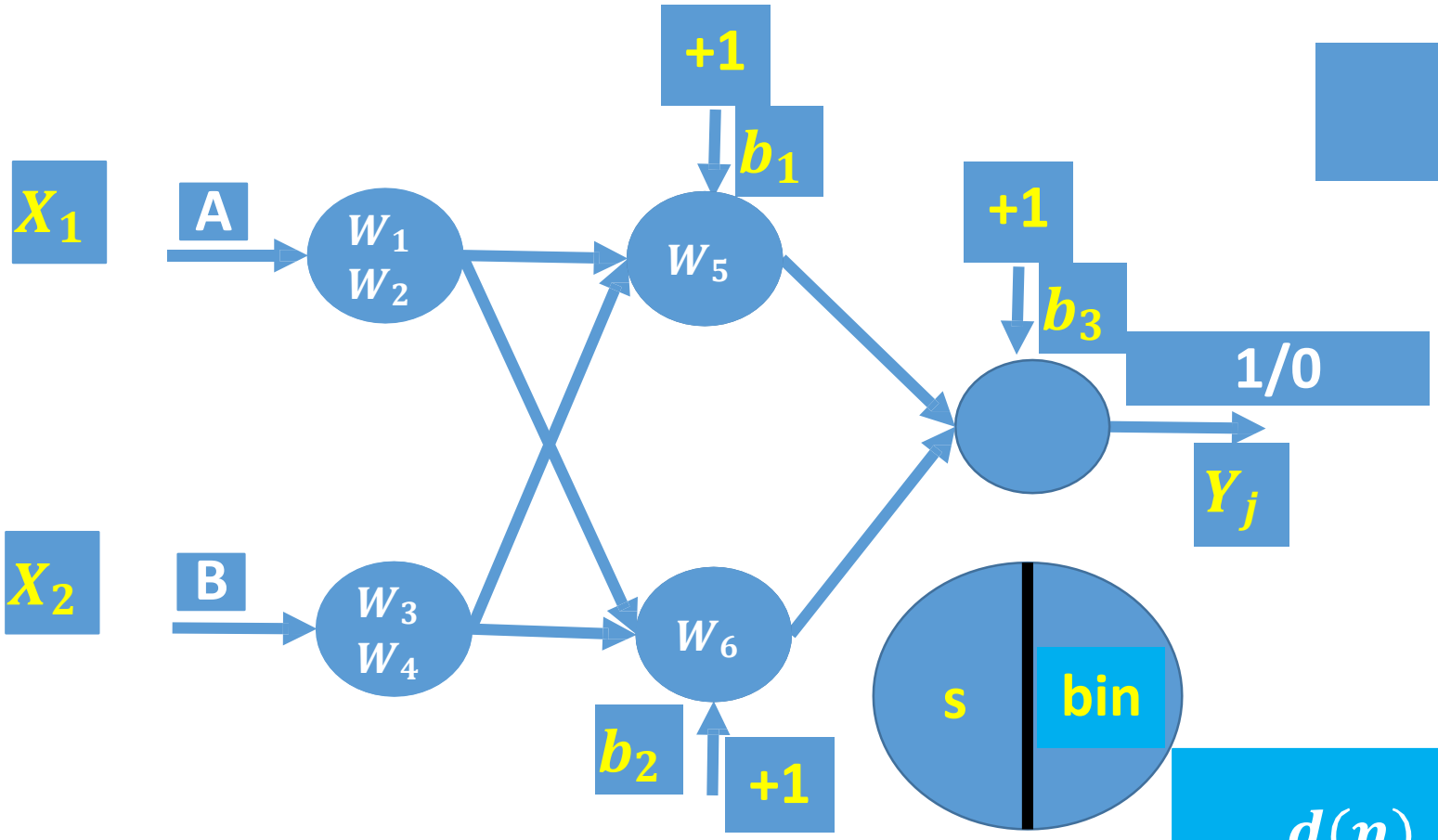


Other Parameters

Desired Output d_j

	A	B
1	1	0
	0	1
0	0	0
	1	1

$$n = 0, 1, 2, \ldots$$



$$s=(X_0W_0+X_1W_1+X_2W_2+\ldots)$$

$$0 \leq \eta \leq 1$$

$$X(n)=(X_0, X_1, X_2, \ldots)$$

$$W(n)=(W_0, W_1, W_2, \ldots)$$

$$d(n) = \begin{cases} 1, x(n)\ belongs\ to\ C1\ (1) \\ 0, x(n)\ belongs\ to\ C2\ (0) \end{cases}$$

Neural Networks 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:

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

Where

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

Neural Networks Training Example

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

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

$$n = 0$$

$$\eta = .001$$

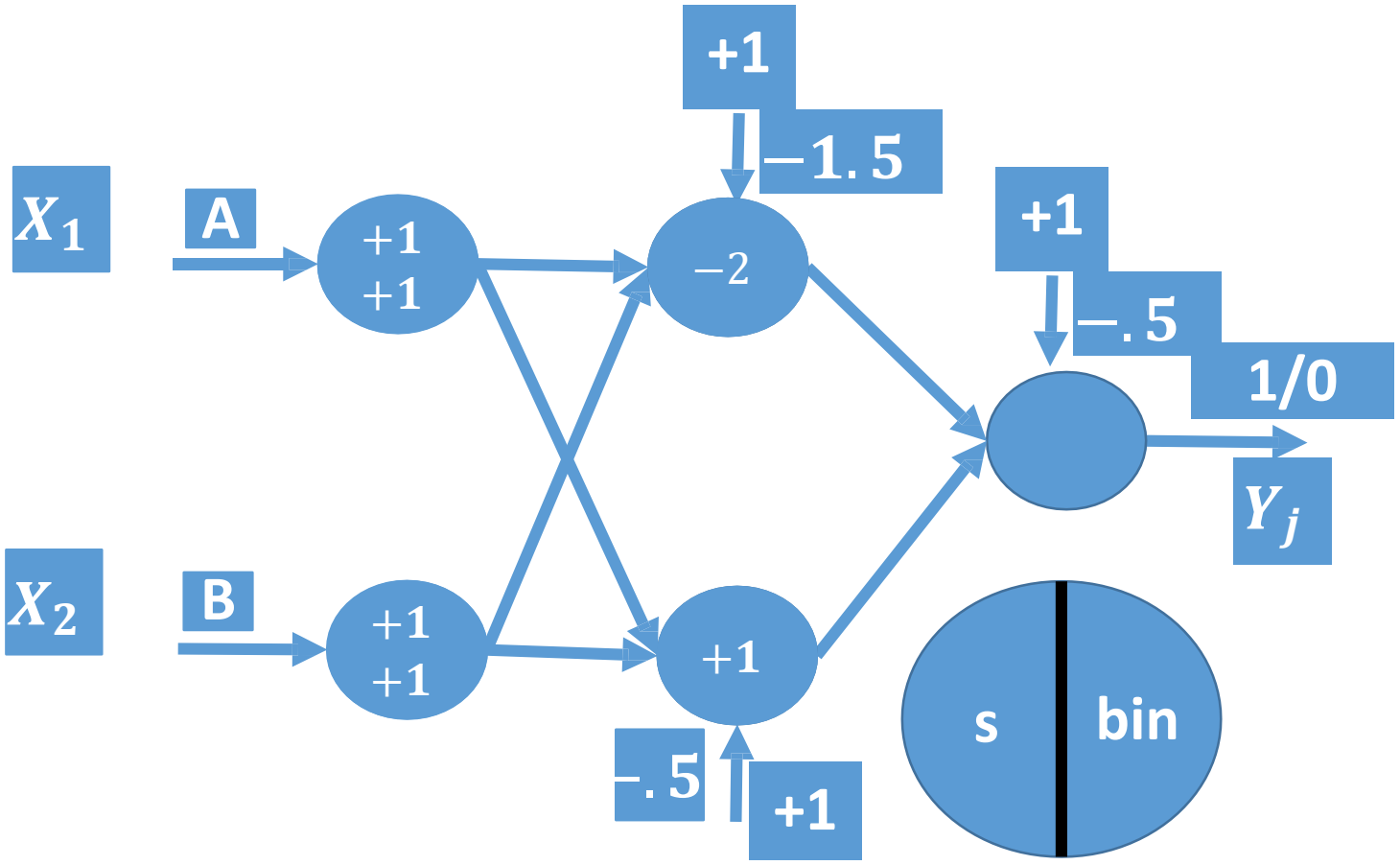
$$X(n) = X(0) = [x_0, x_1, x_2, x_3, x_4] = [+1, +1, +1, 1, 0]$$

$$W(n) = W(0) = [\, b_1, b_2, b_3, w_1, w_2, w_3\, , w_4, \, w_5, w_6] = [-1.5, -0.5, -0.5, 1, 1, 1, 1, -2, 1]$$

$$d(n) = d(0) = 1$$

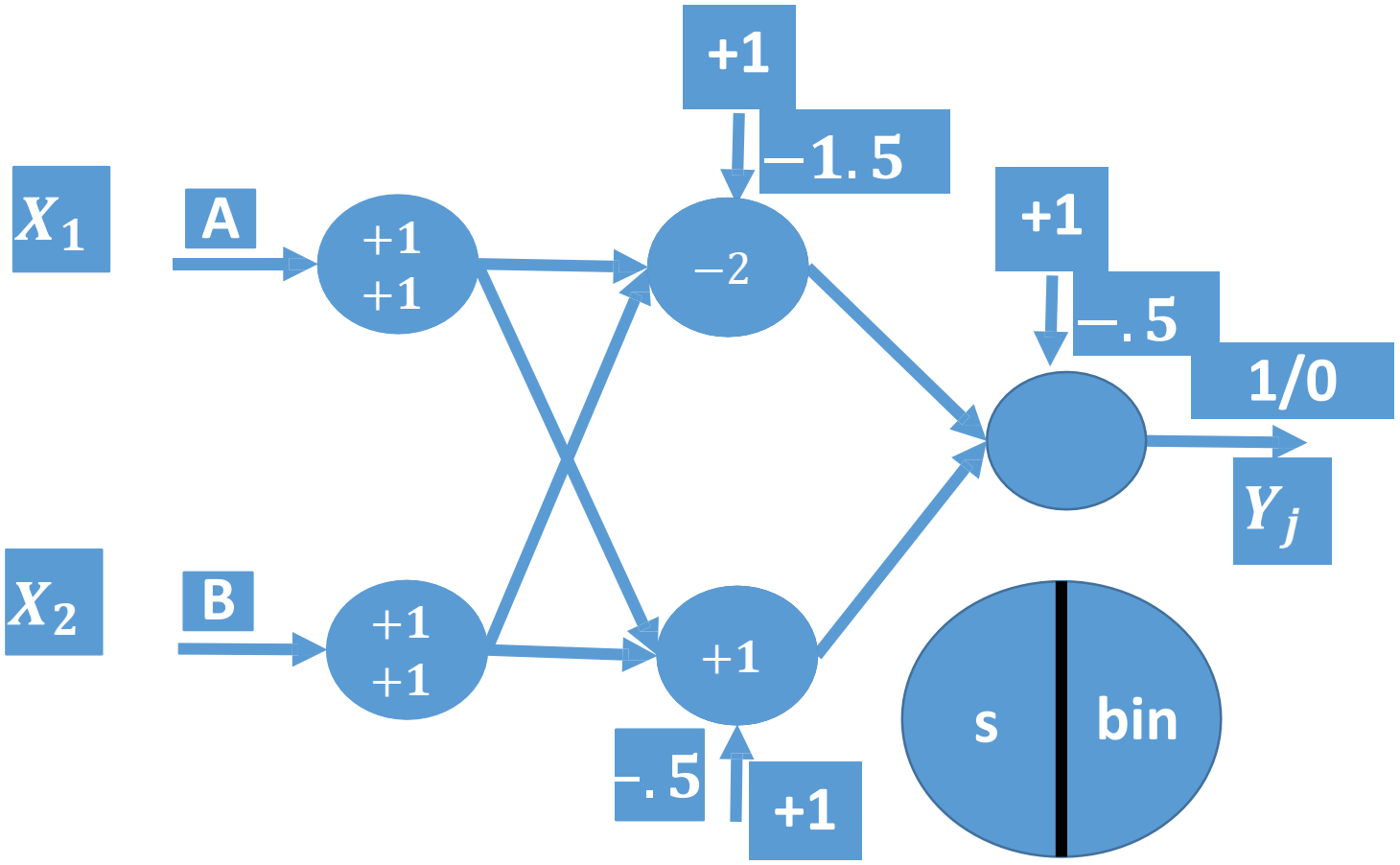
Neural Networks Training Example Step n=0

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



Neural Networks Training Example Step n=0 – SOP – S_1

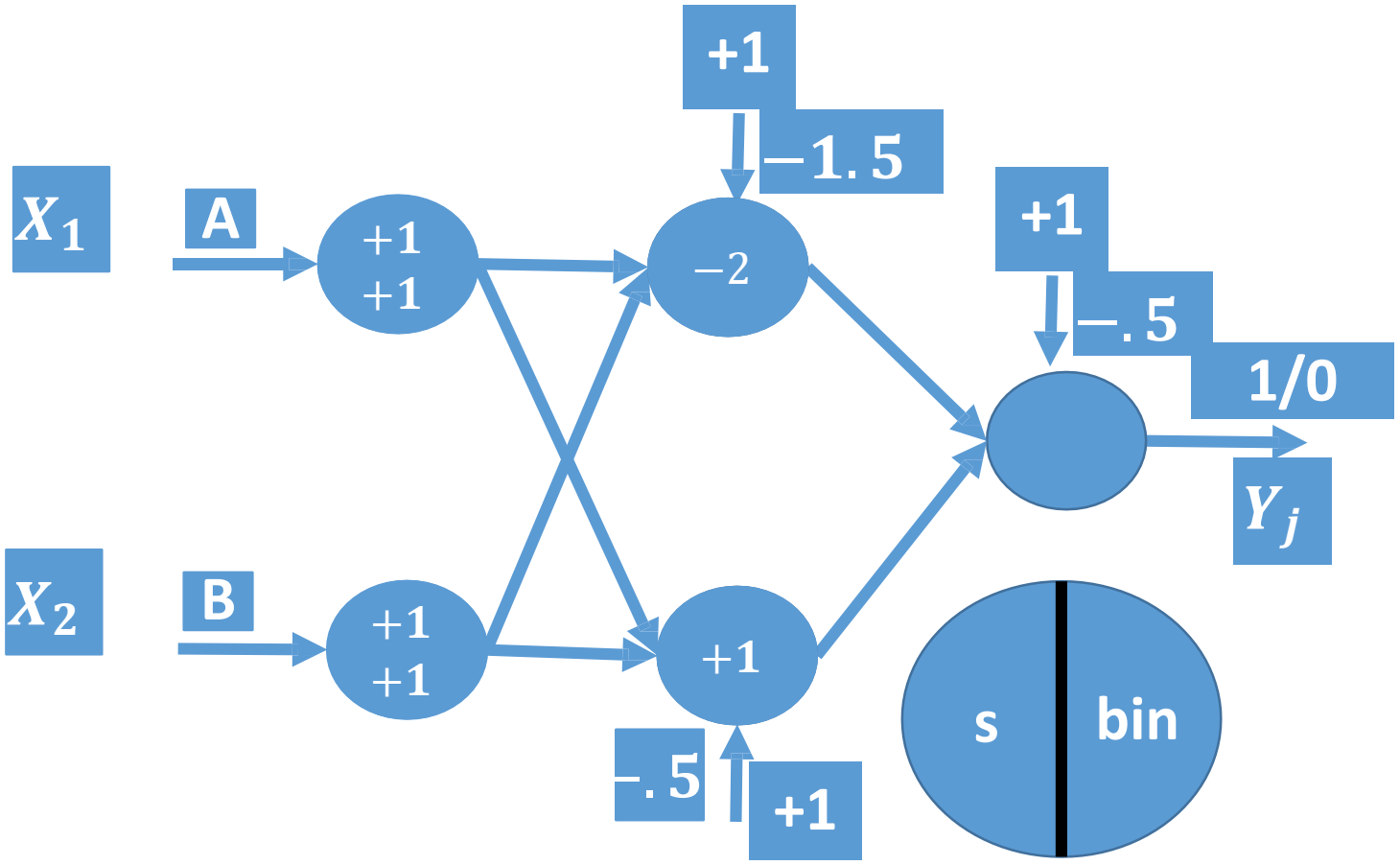
	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



$S_1=(+1b_1+X_1W_1+X_2W_3)$
=+1*-1.5+1*1+0*1
=-.5

Neural Networks Training
Example Step n=0 – Output – S_1

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

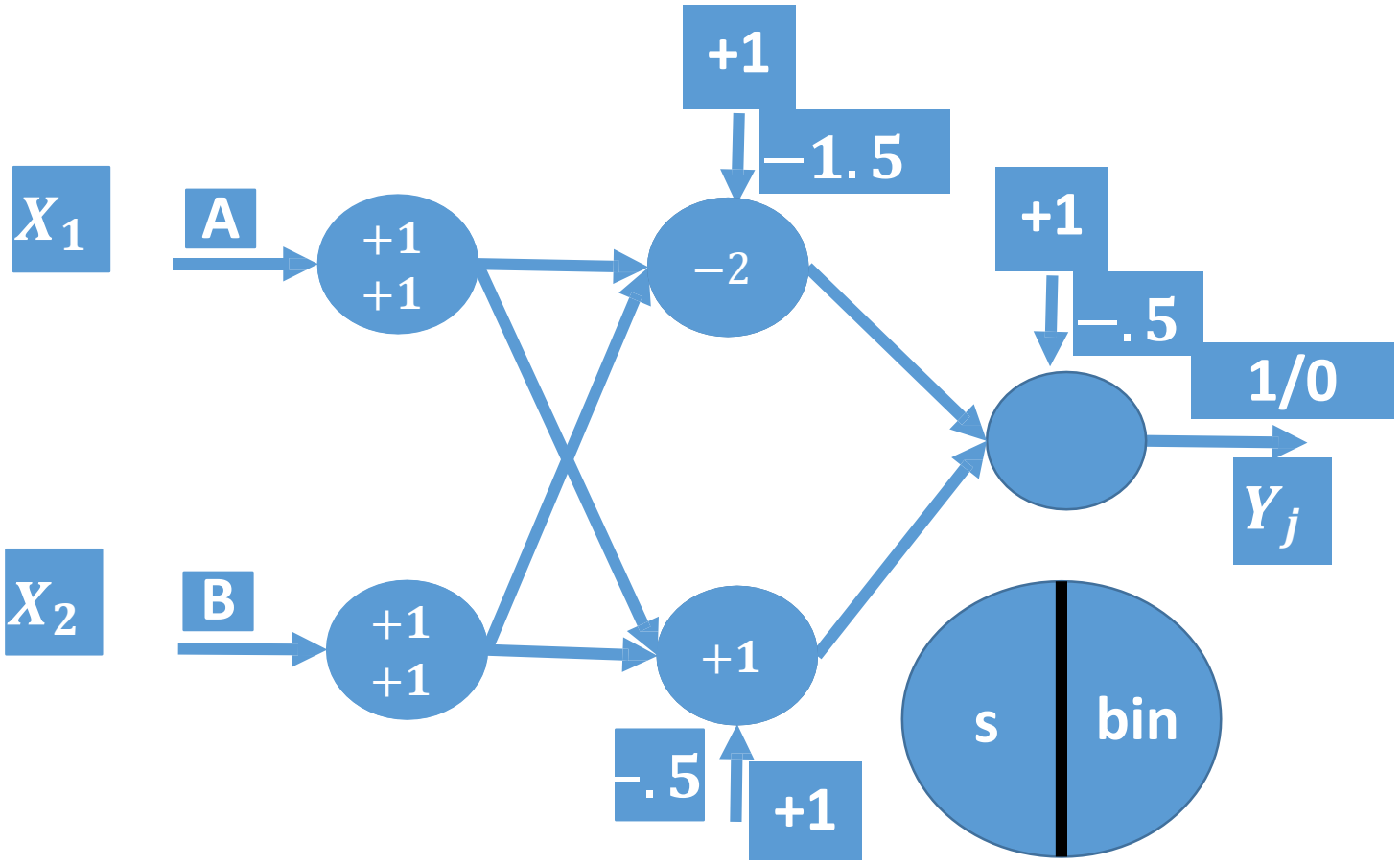


$$Y(S_1) = \\ = BIN(S_1) \\ = BIN(-.5) \\ = 0$$

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

Neural Networks Training Example Step n=0 – SOP – S_2

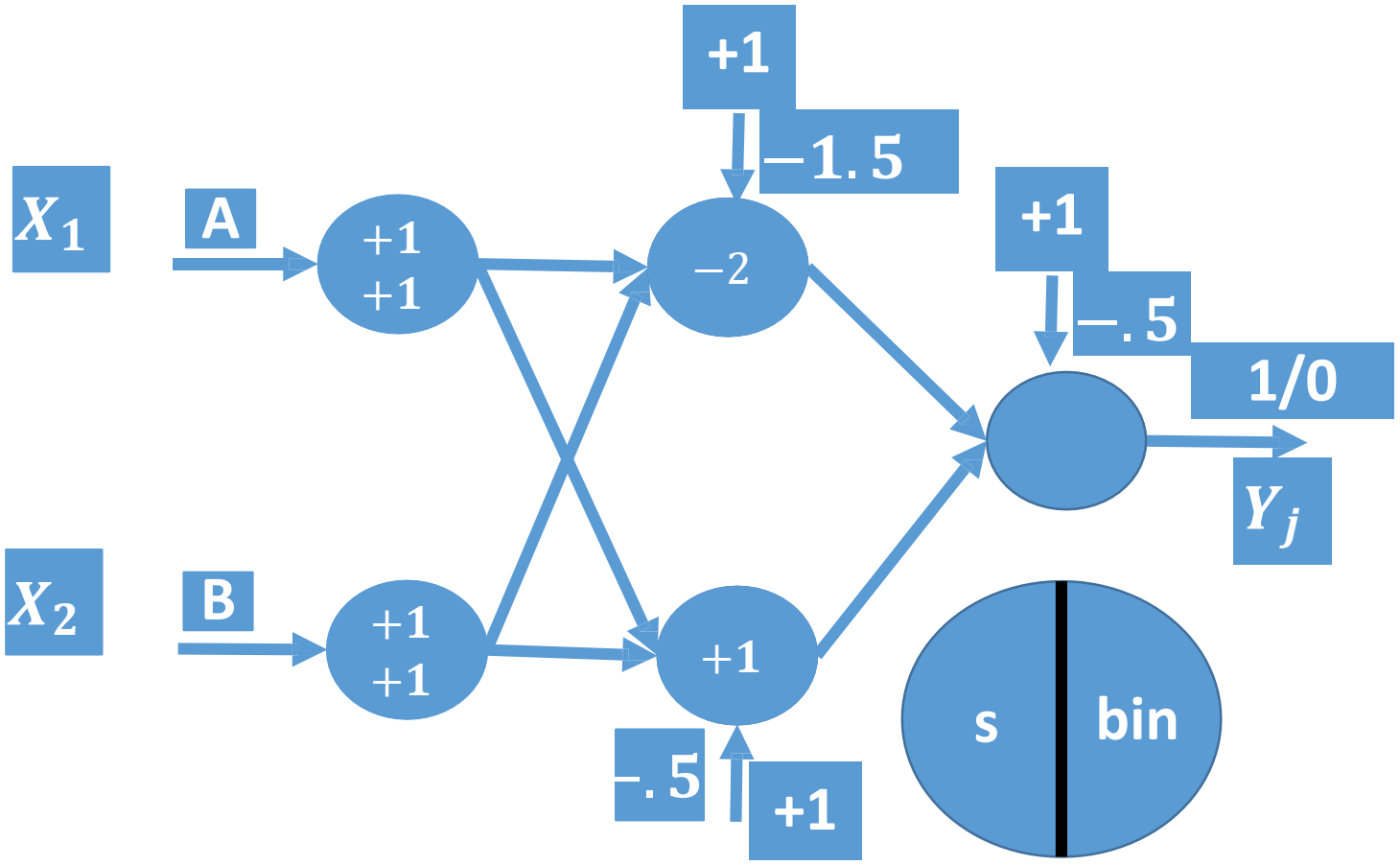
	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



$S_2=(+1b_2+X_1W_2+X_2W_4)$
=+1*-.5+1*1+0*1
=.5

Neural Networks Training
Example Step n=0 – Output – S_2

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

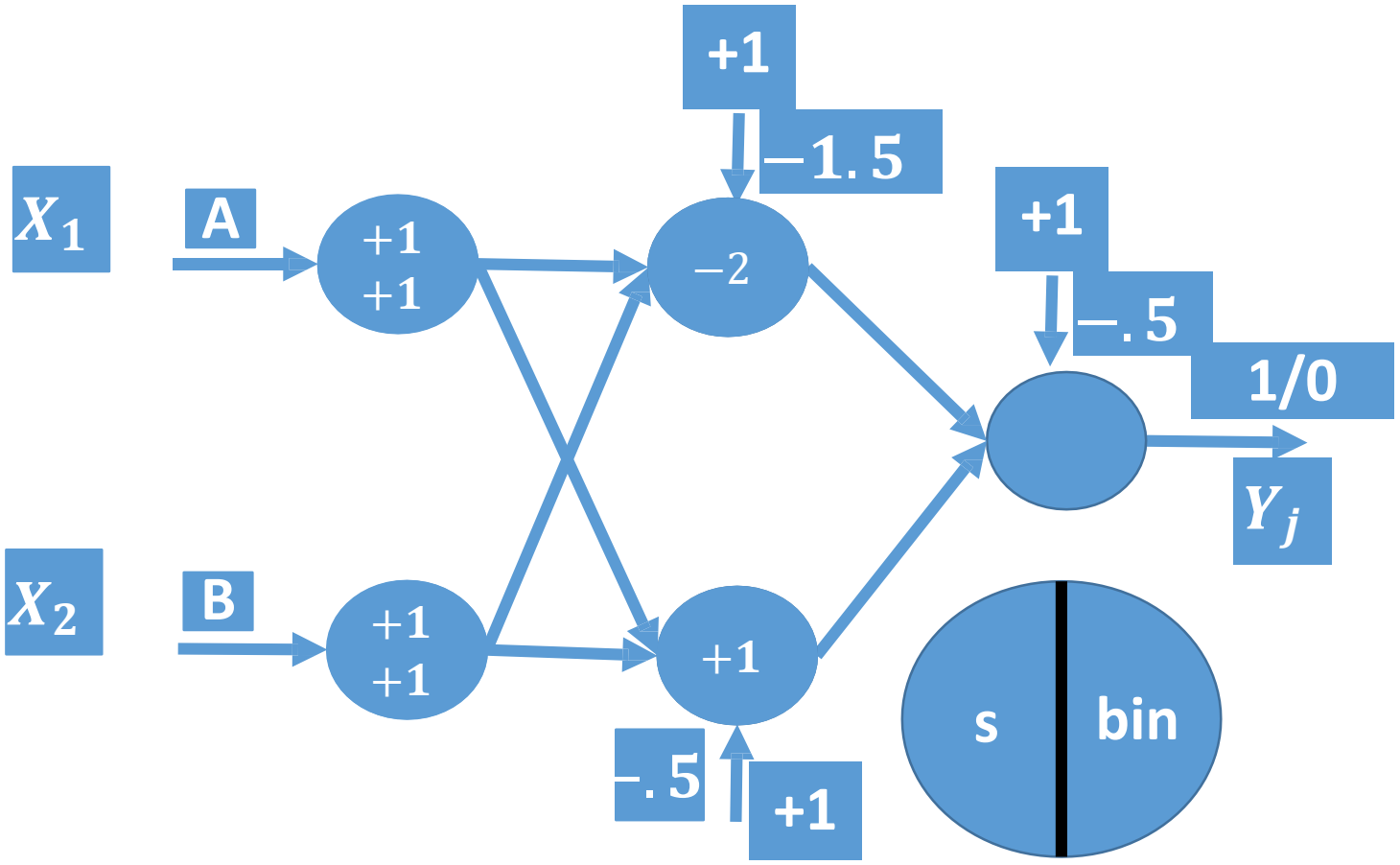


$$Y(S_2) = \\ = BIN(S_2) \\ = BIN(.5) \\ = 1$$

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

Neural Networks Training Example Step n=0 – SOP – S_3

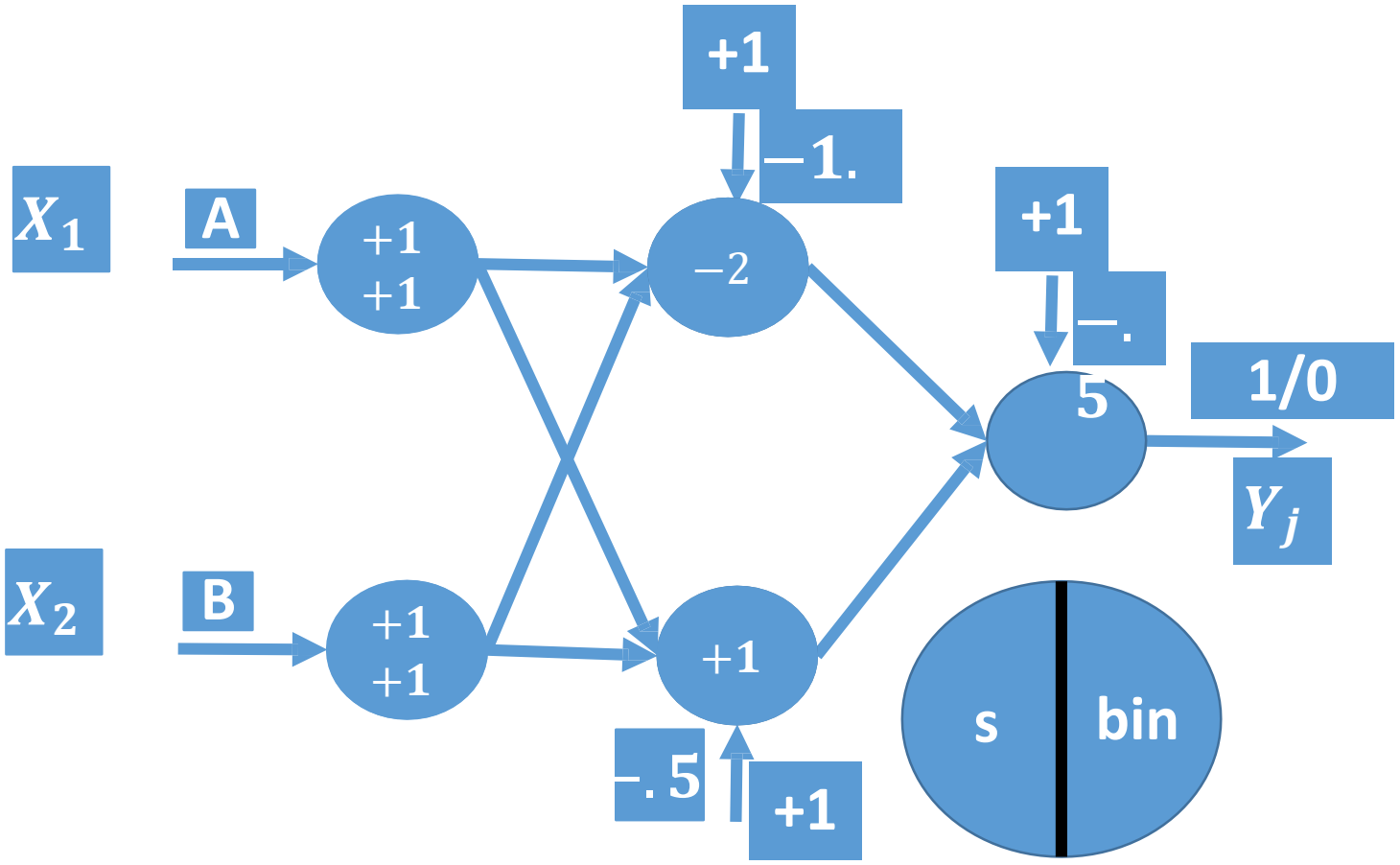
	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



$$S_3=(+1b_3+S_1W_5+S_2W_6)$$
$$=+1*-.5+0*-2+1*1$$
$$=.5$$

Neural Networks Training
Example Step n=0 – Output – S_3

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

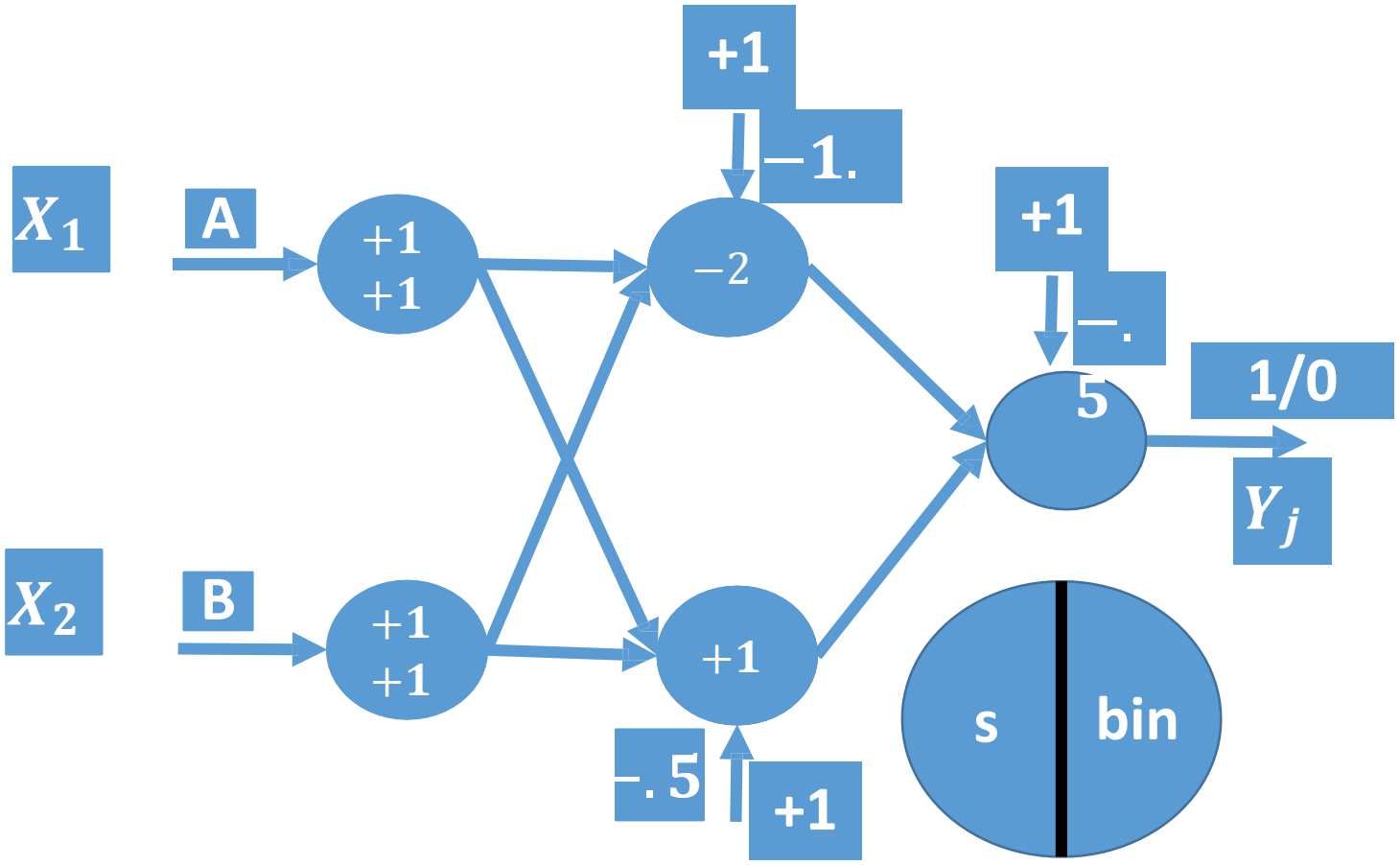


$$Y(S_3) = \\ = BIN(S_3) \\ = BIN(.5) \\ = 1$$

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

Neural Networks Training Example Step n=0 - Output

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

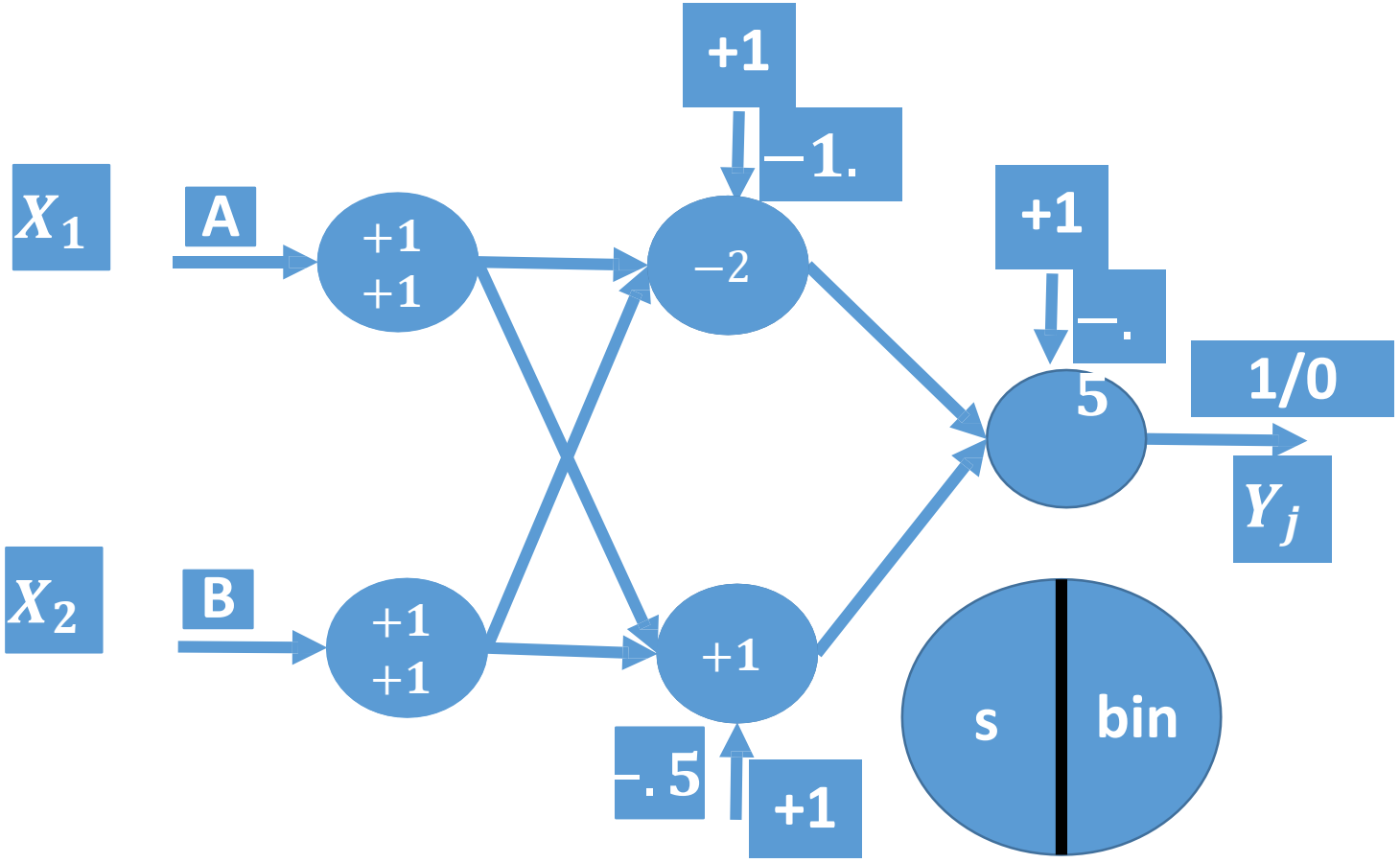


$$Y(n) = Y(0) = Y(S_3) = 1$$

Neural Networks Training Example Step n=0

Predicted Vs. Desired

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



$$Y(n) = Y(0) = 1$$
$$\mathrm{d}(n) = d(0) = 1$$
$$\because Y(n) = d(n)$$
$$\therefore \text{Weights are Correct.}$$
$$\text{No Adaptation}$$

Neural Networks Training Example

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

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

$$n = 1$$

$$\eta = .001$$

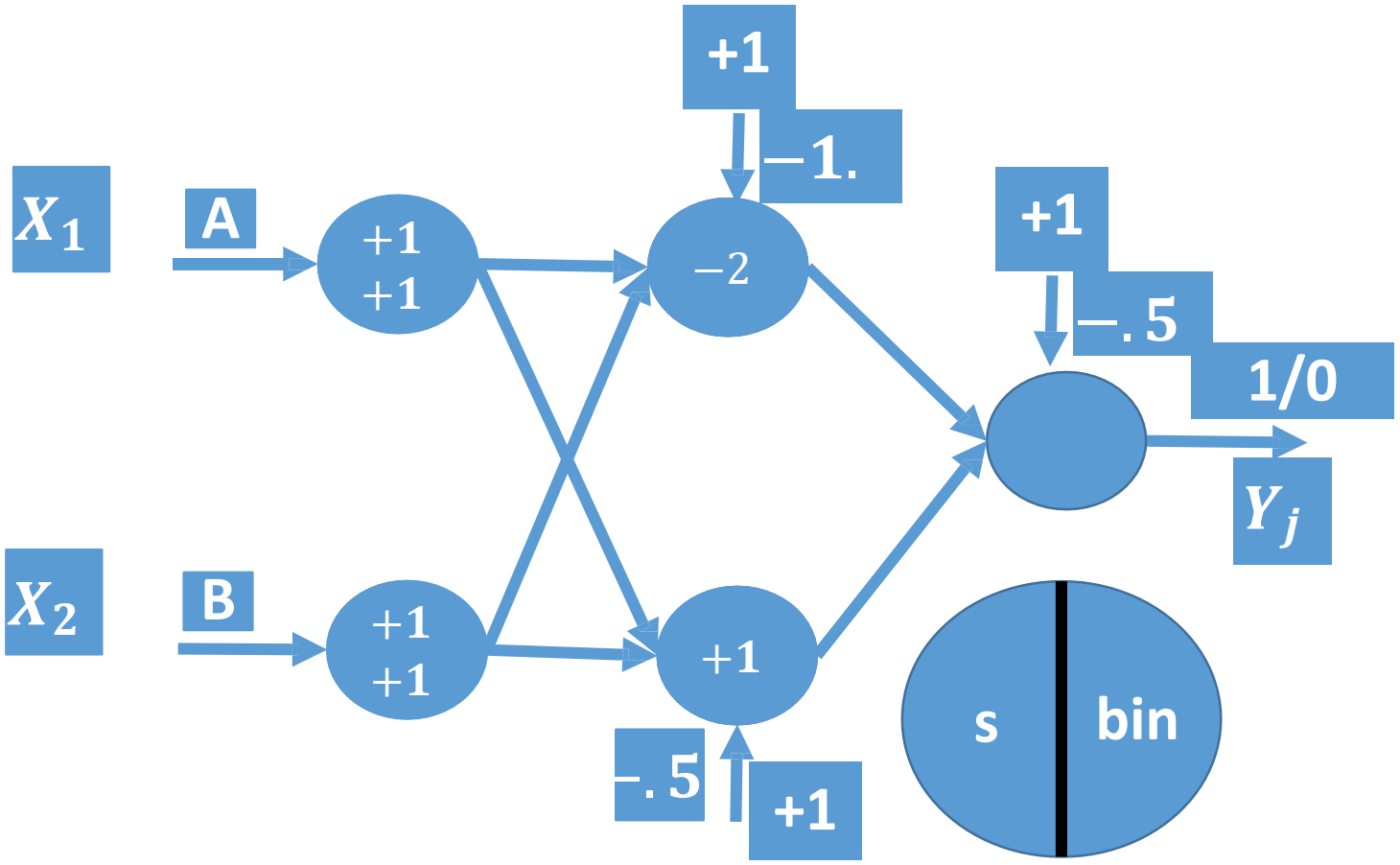
$$X(n) = X(1) = [x_0, x_1, x_2, x_3, x_4] = [+1, +1, +1, 0, 1]$$

$$W(n) = W(0) = [\, b_1, b_2, b_3, w_1, w_2, w_3\, , w_4, w_5, w_6] = [-1.5, -0.5, -0.5, 1, 1, 1, 1, -2, 1]$$

$$d(n) = d(1) = 1$$

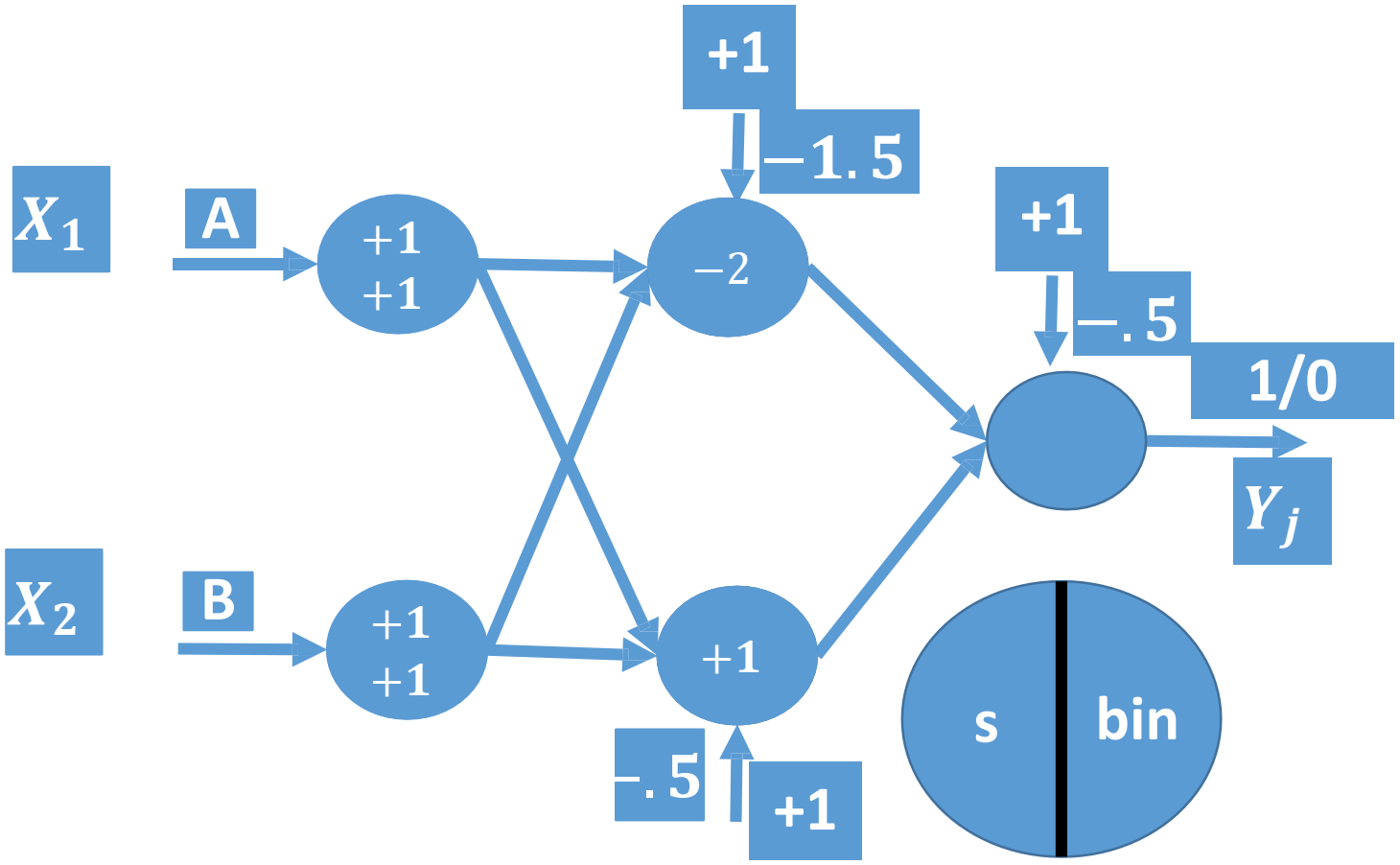
Neural Networks Training Example Step n=1

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



Neural Networks Training Example Step n=1 – SOP – S_1

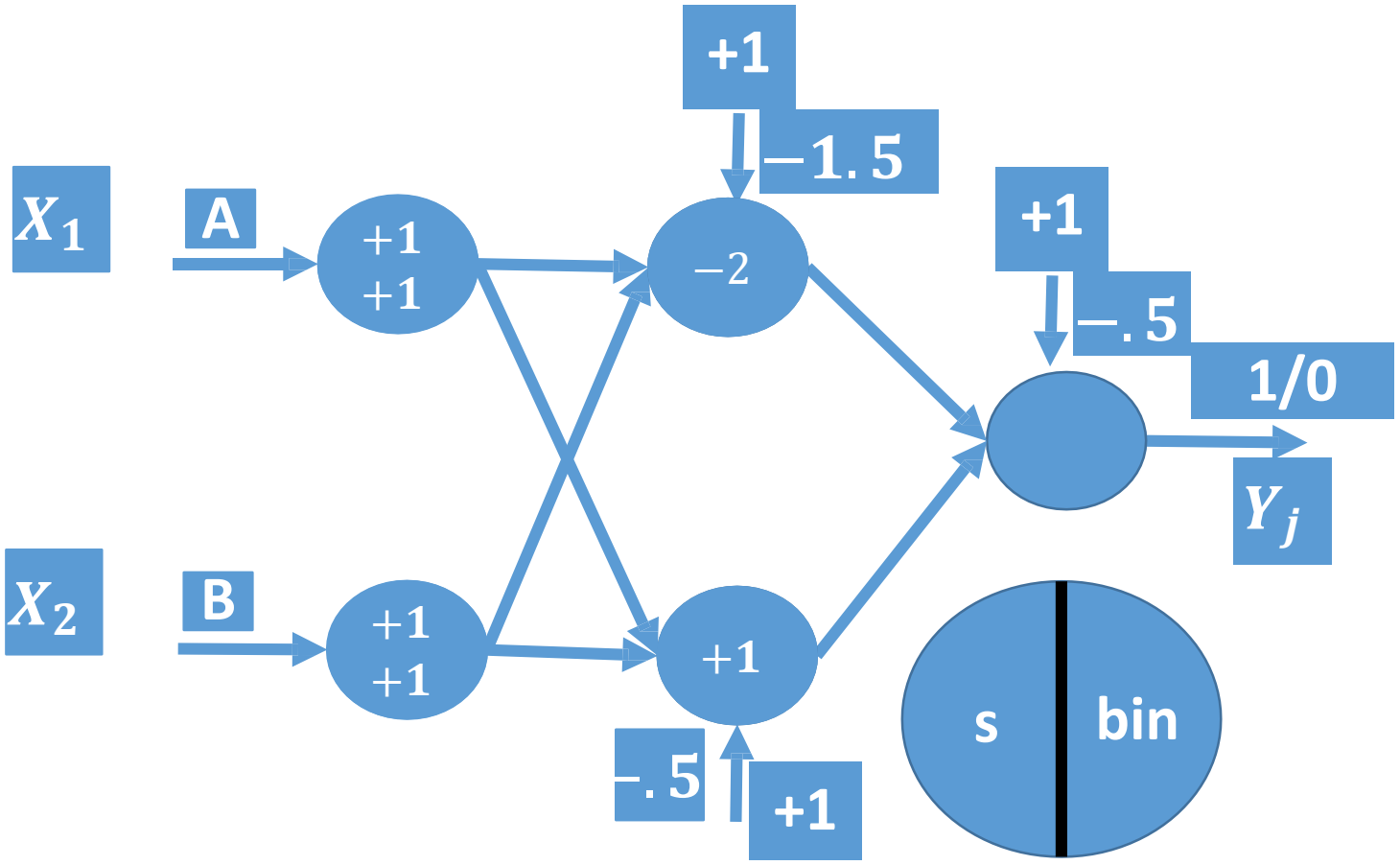
	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



$S_1=(+1b_1+X_1W_1+X_2W_3)$
=+1*-1.5+0*1+1*1
=-.5

Neural Networks Training
Example Step n=1 – Output – S_1

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

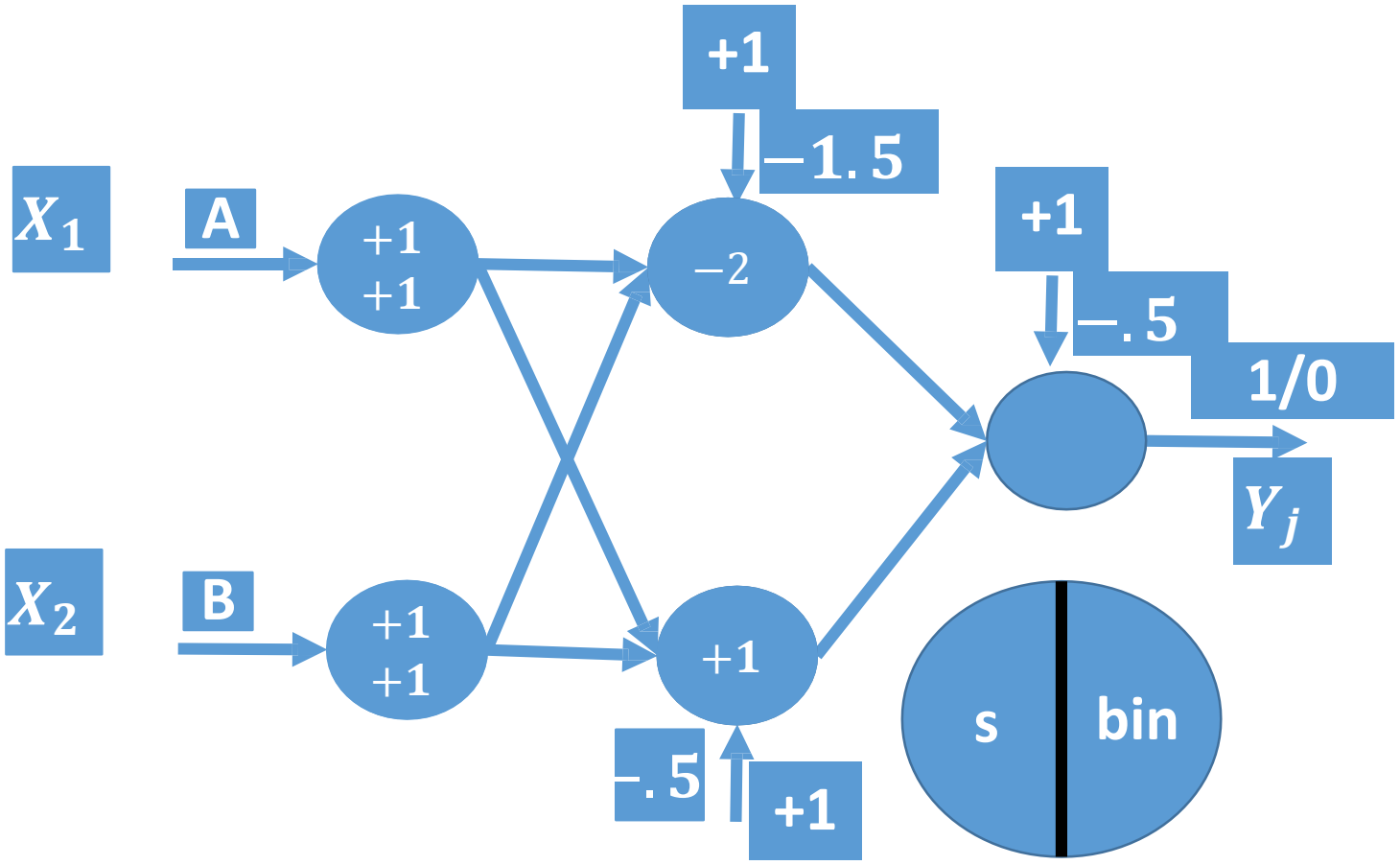


$$Y(S_1) = \\ = BIN(S_1) \\ = BIN(-.5) \\ = 0$$

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

Neural Networks Training Example Step n=1 – SOP – S_2

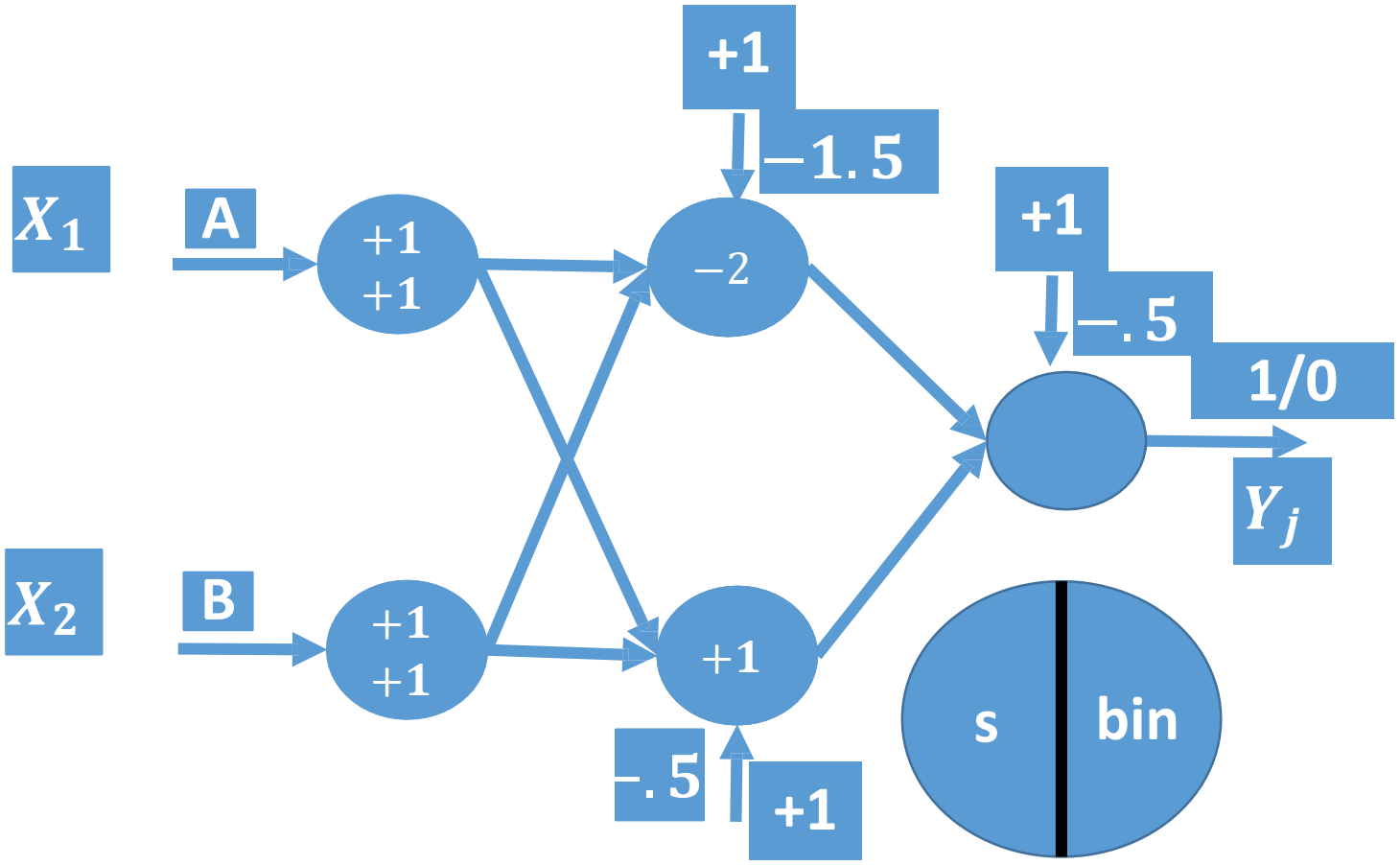
	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



$$S_2=(+1b_2+X_1W_2+X_2W_4)$$
$$=+1*-.5+0*1+1*1$$
$$=.5$$

Neural Networks Training
Example Step n=1 – Output – S_2

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

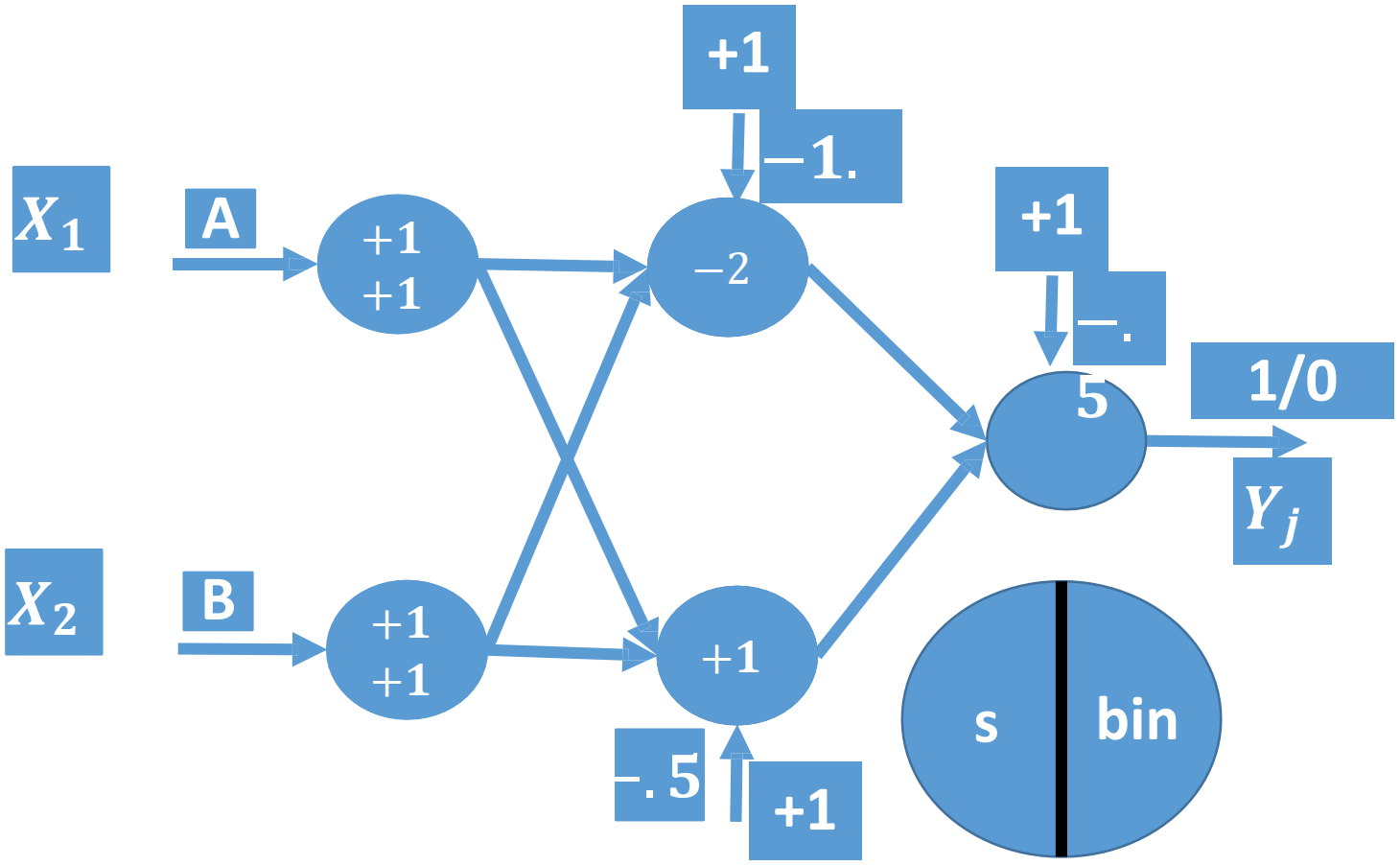


$$Y(S_2) = \\ = BIN(S_2) \\ = BIN(.5) \\ = 1$$

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

Neural Networks Training Example Step n=1 – SOP – S_3

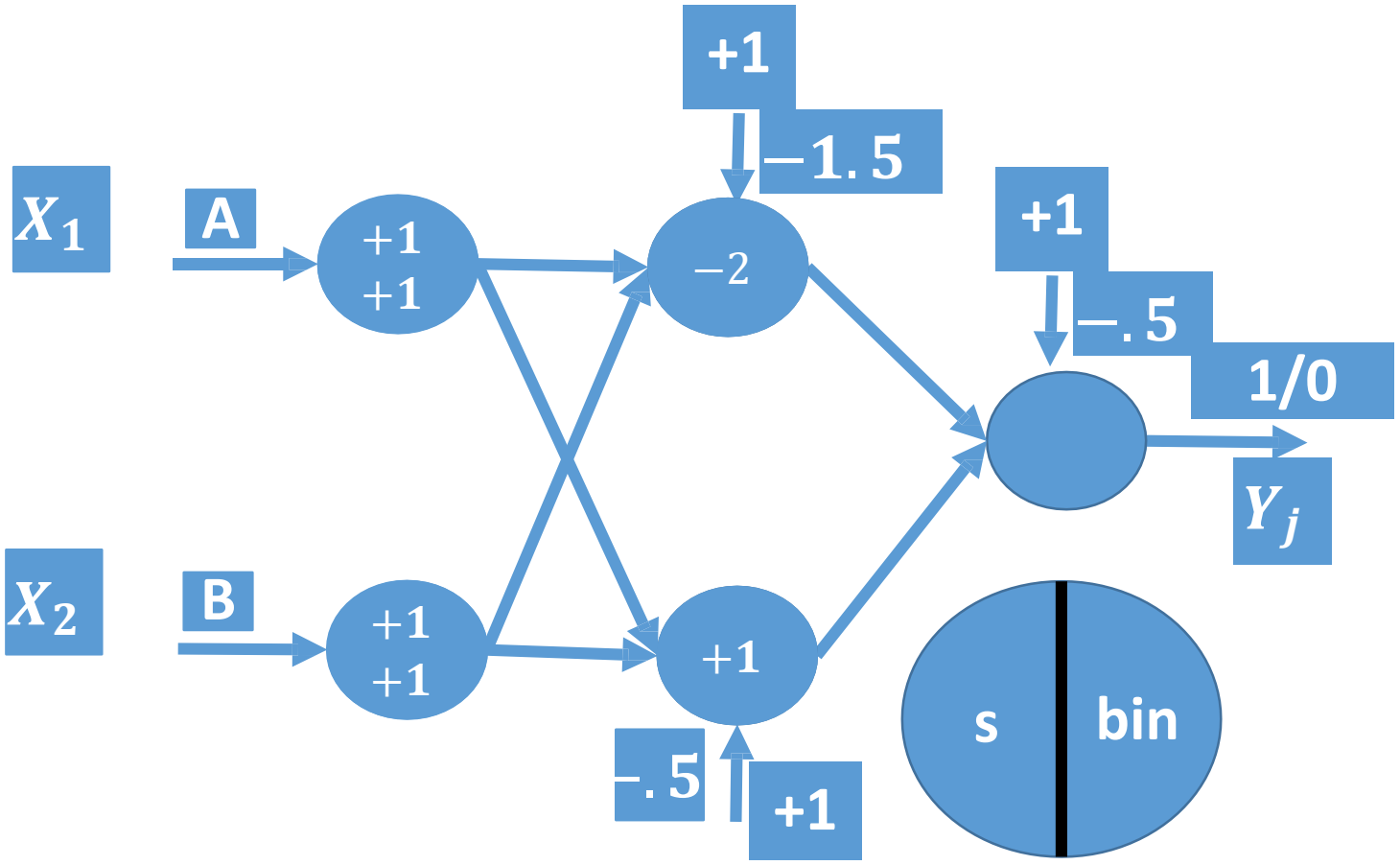
	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



$$S_3=(+1b_3+S_1W_5+S_2W_6)$$
$$=+1*-.5+0*-2+1*1$$
$$=.5$$

Neural Networks Training
Example Step n=1 – Output – S_3

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

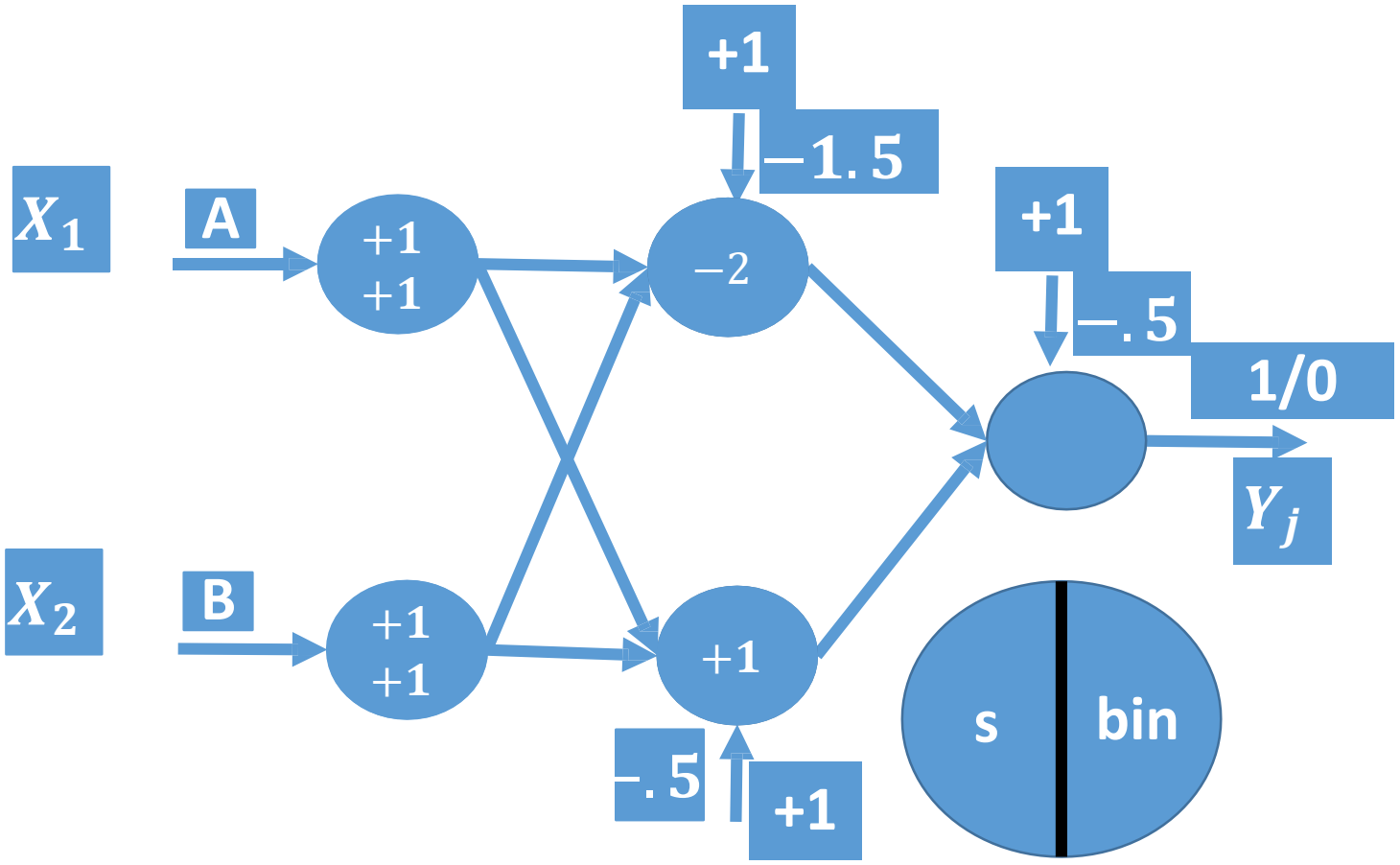


$$Y(S_3) = \\ = BIN(S_3) \\ = BIN(.5) \\ = 1$$

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

Neural Networks Training Example Step n=1 - Output

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

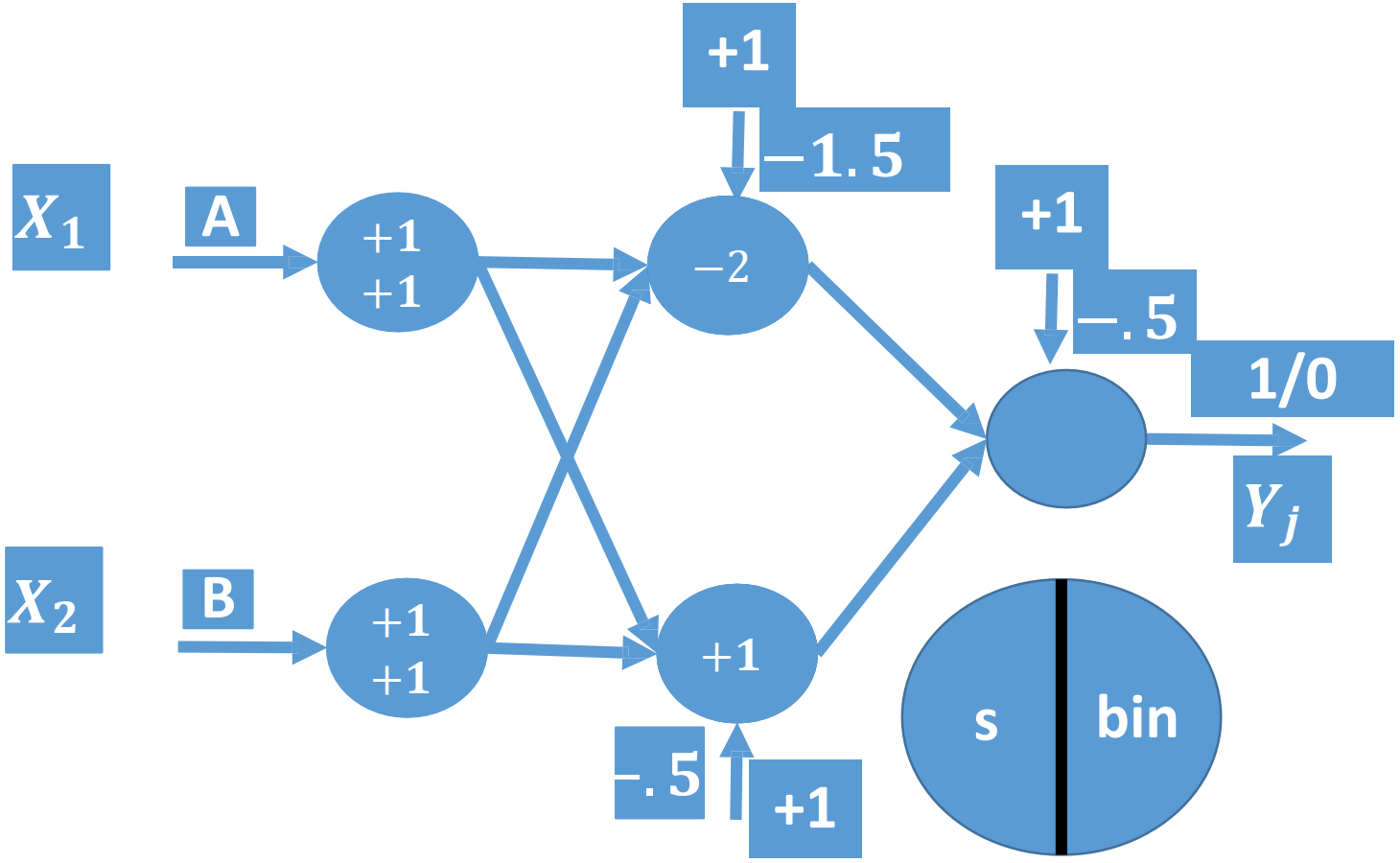


$$Y(n) = Y(1) = Y(S_3) = 1$$

Neural Networks Training Example Step n=1

Predicted Vs. Desired

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



$$Y(n) = Y(1) = 1$$
$$\mathrm{d}(n) = d(1) = 1$$
$$\because Y(n) = d(n)$$
$$\therefore \text{Weights are Correct.}$$
$$\text{No Adaptation}$$

Neural Networks Training Example

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

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

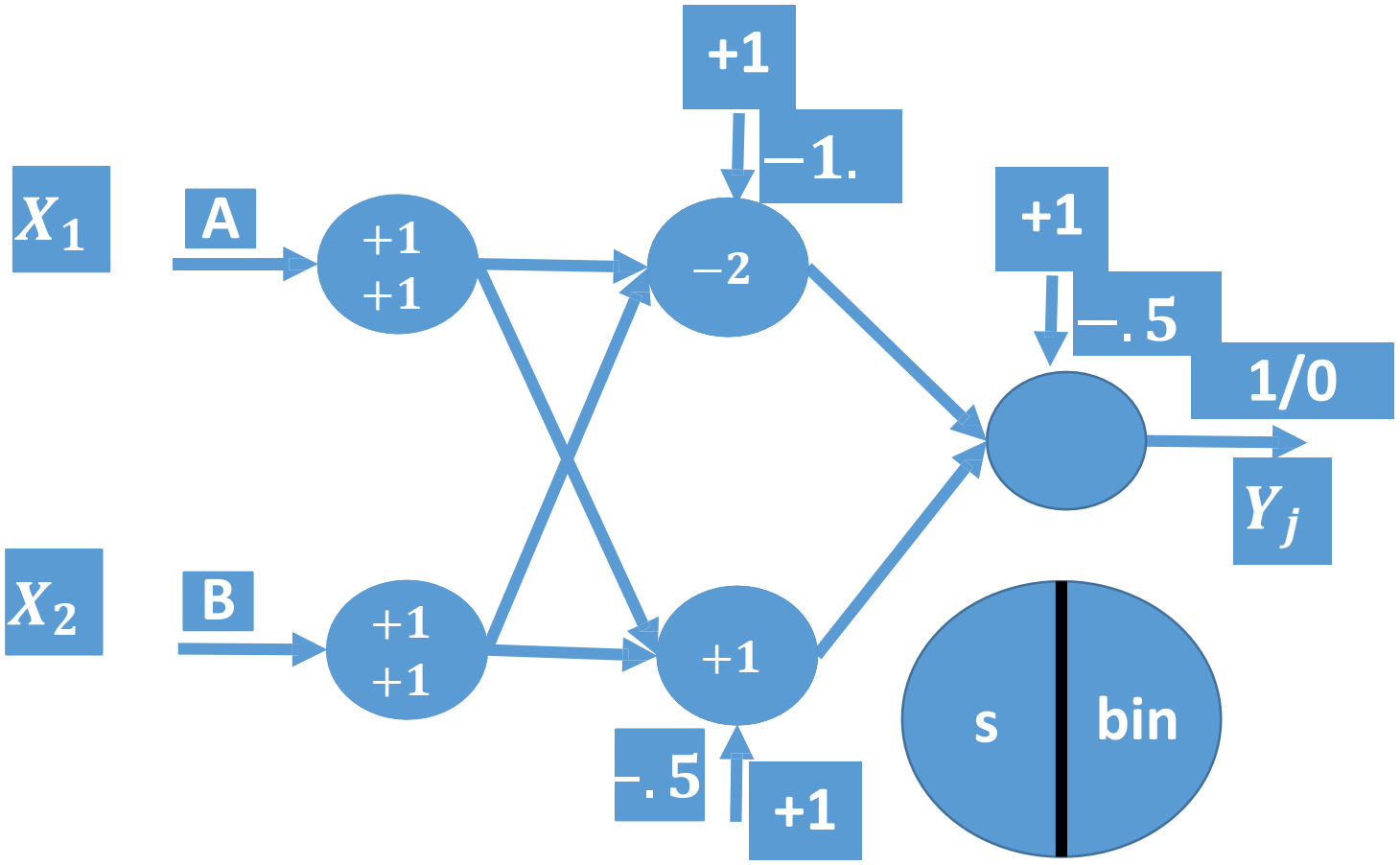
$$n = 2$$
$$\eta = .001$$

$$X(n) = X(2) = [x_0, x_1, x_2, x_3, x_4] = [+1, +1, +1, 0, 0]$$
$$W(n) = W(2) = [\, b_1, b_2, b_3, w_1, w_2, w_3\, , w_4, \, w_5, w_6] = [-1.5, -0.5, -0.5, 1, 1, 1, 1, -2, 1]$$

$$d(n) = d(2) = 0$$

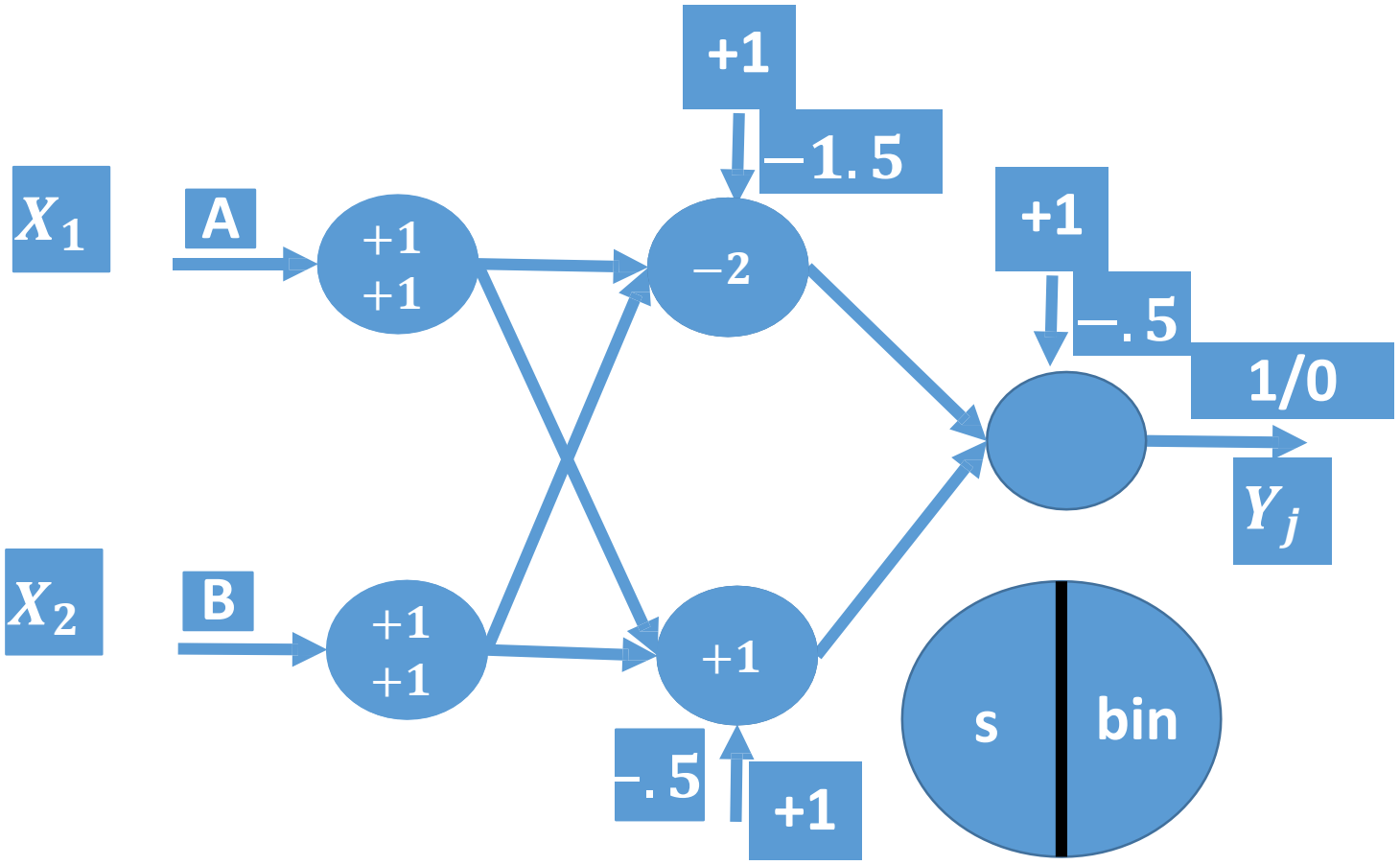
Neural Networks Training Example Step n=2

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



Neural Networks Training Example Step n=2 – SOP – S_1

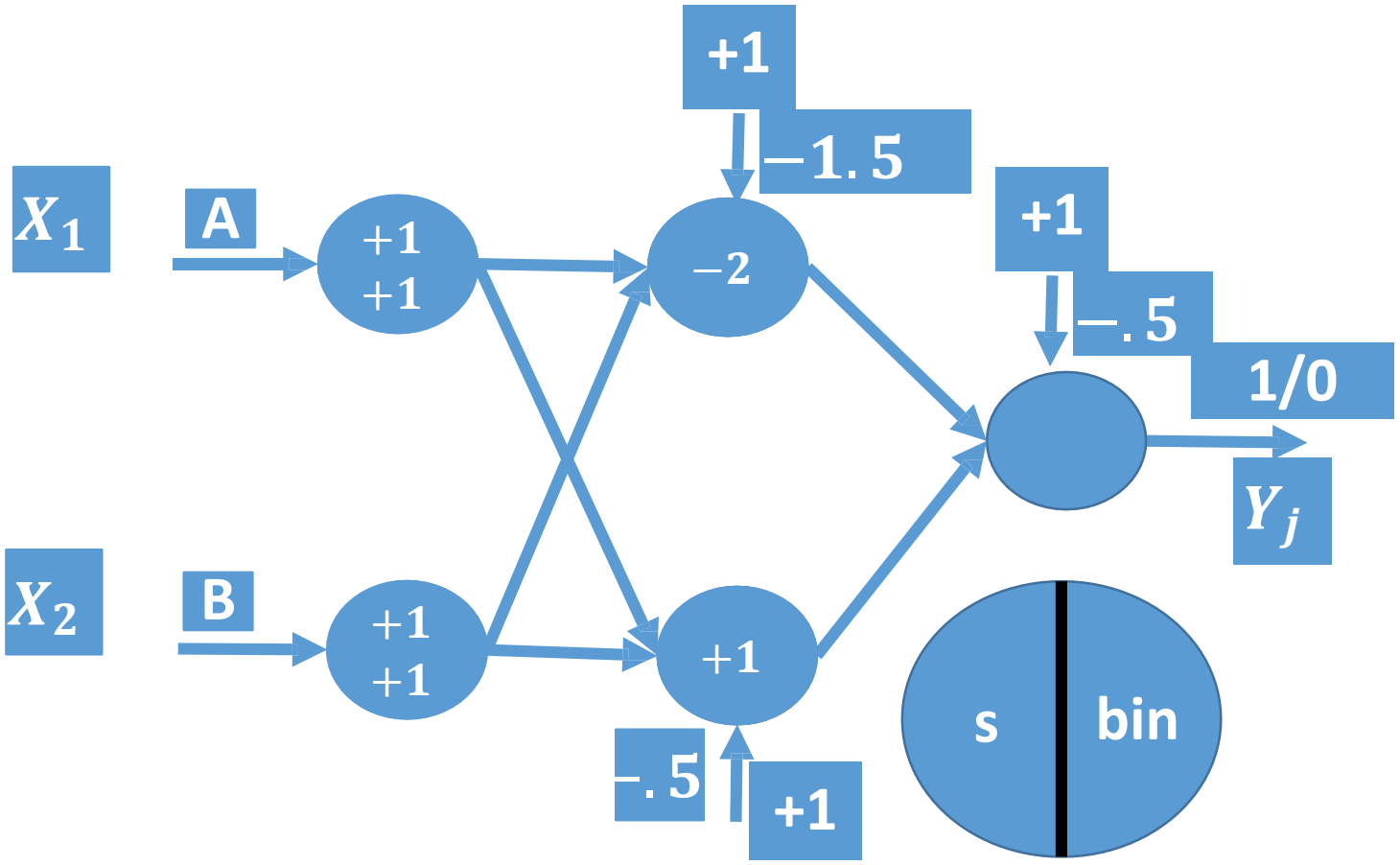
	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



$S_1=(+1b_1+X_1W_1+X_2W_3)$
=+1*-1.5+0*1+0*1
=-1.5

Neural Networks Training
Example Step n=2 – Output – S_1

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

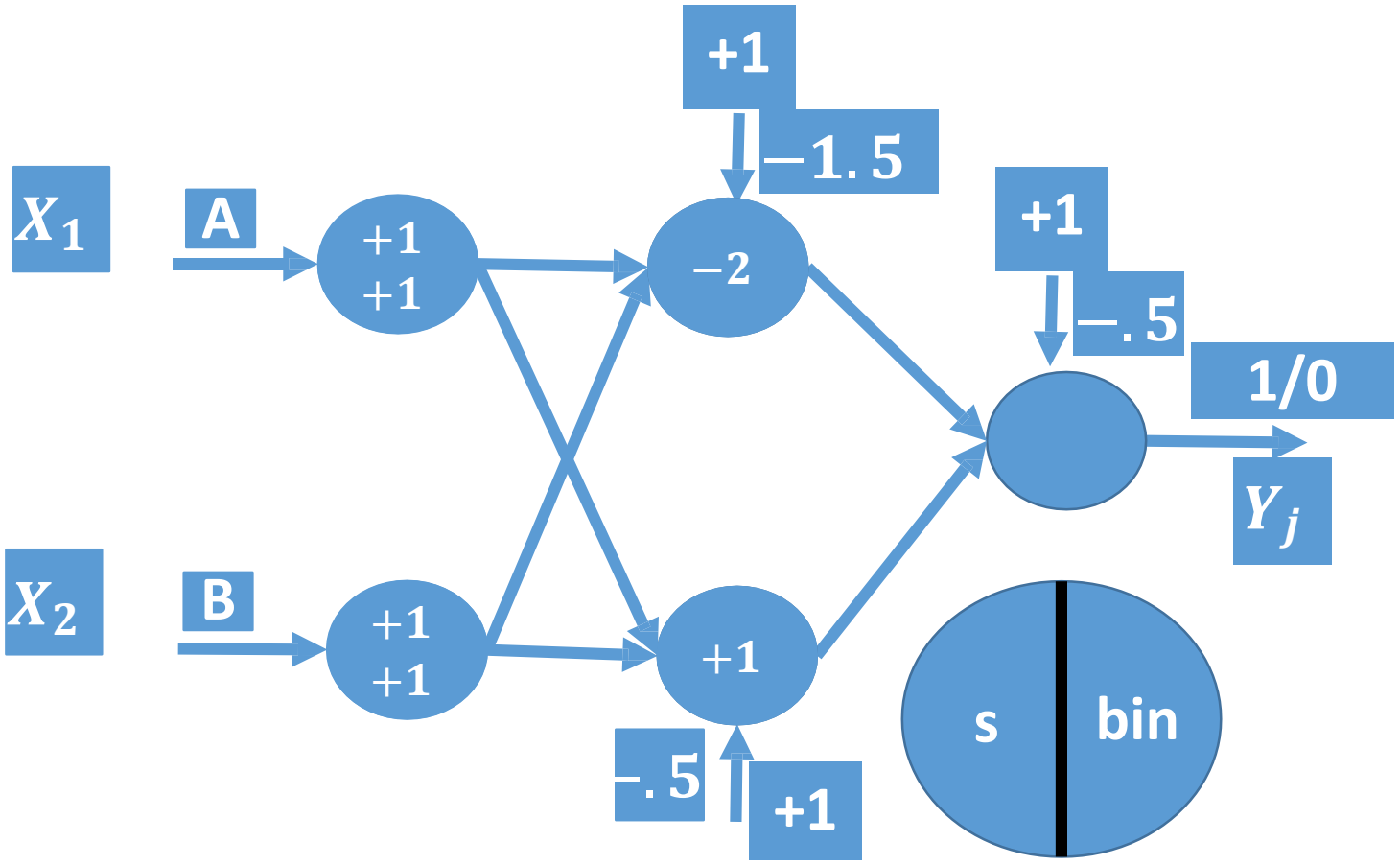


$$Y(S_1) = \\ = BIN(S_1) \\ = BIN(-1.5) \\ = 0$$

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

Neural Networks Training Example Step n=2 – SOP – S_2

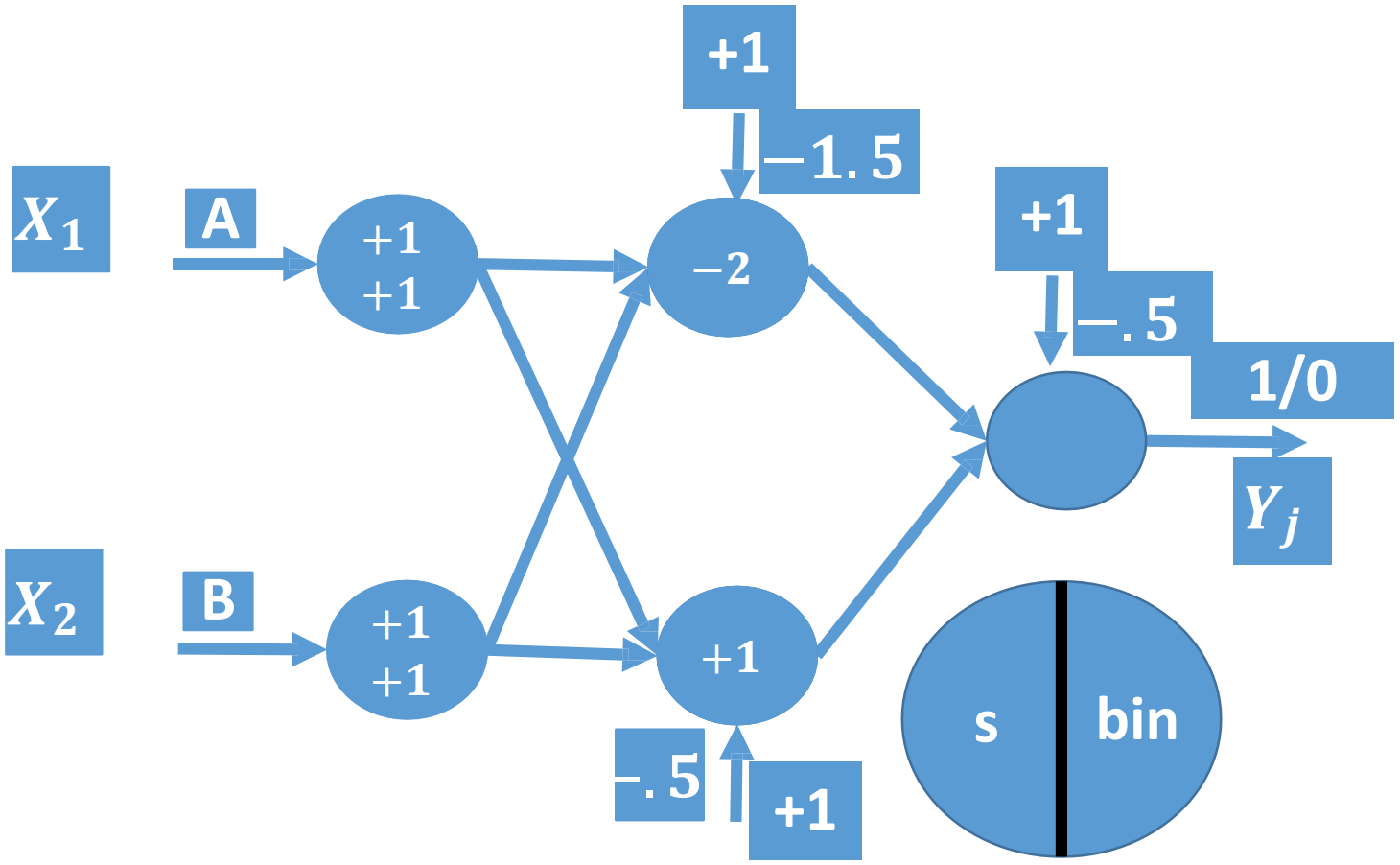
	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



$$S_2=(+1b_2+X_1W_2+X_2W_4)$$
$$=+1*-.5+0*1+0*1$$
$$=-.5$$

Neural Networks Training
Example Step n=2 – Output – S_2

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

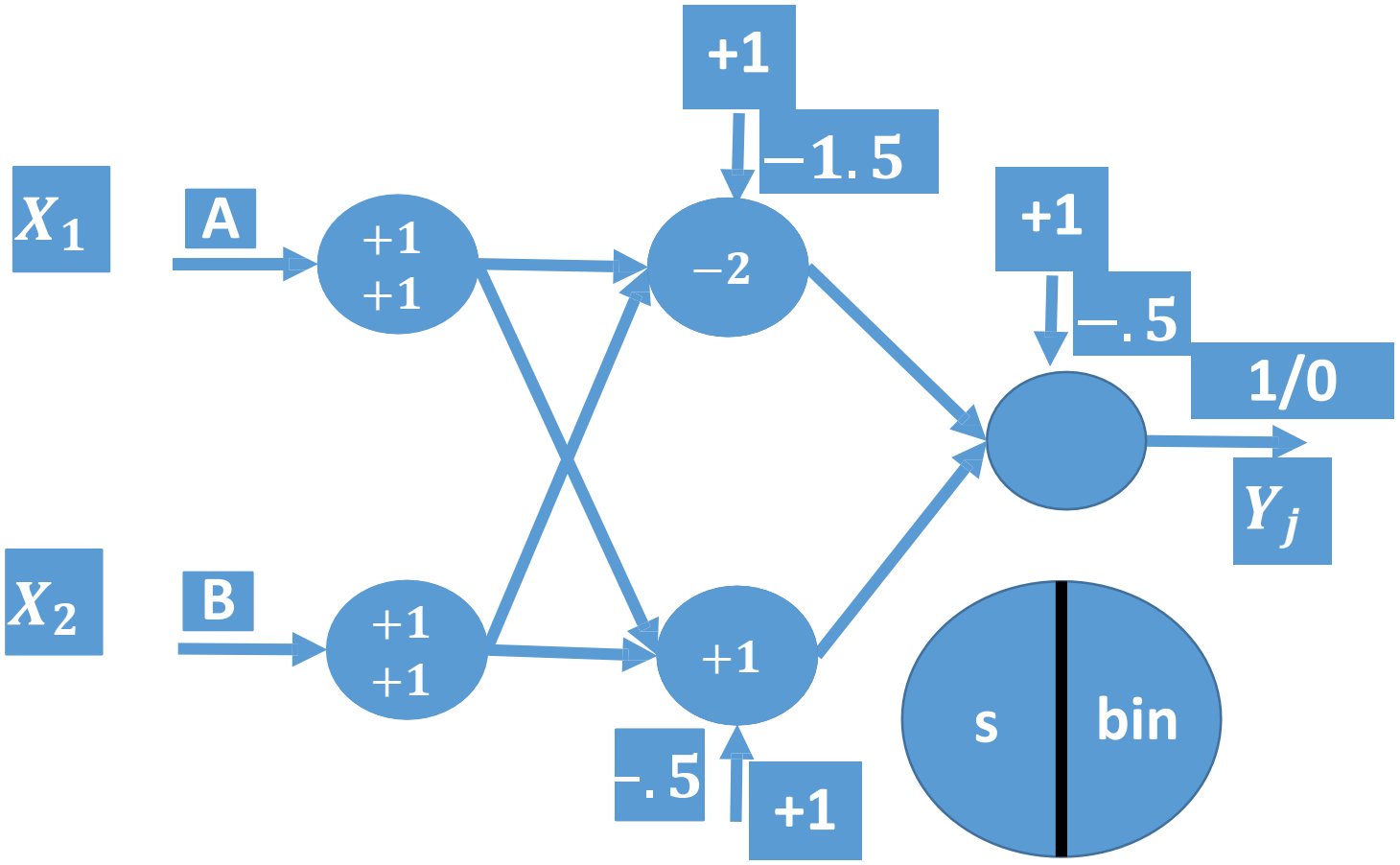


$$Y(S_2) = \\ = SGN(S_2) \\ = SGN(-.5) \\ = 0$$

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

Neural Networks Training Example Step n=2 – SOP – S_3

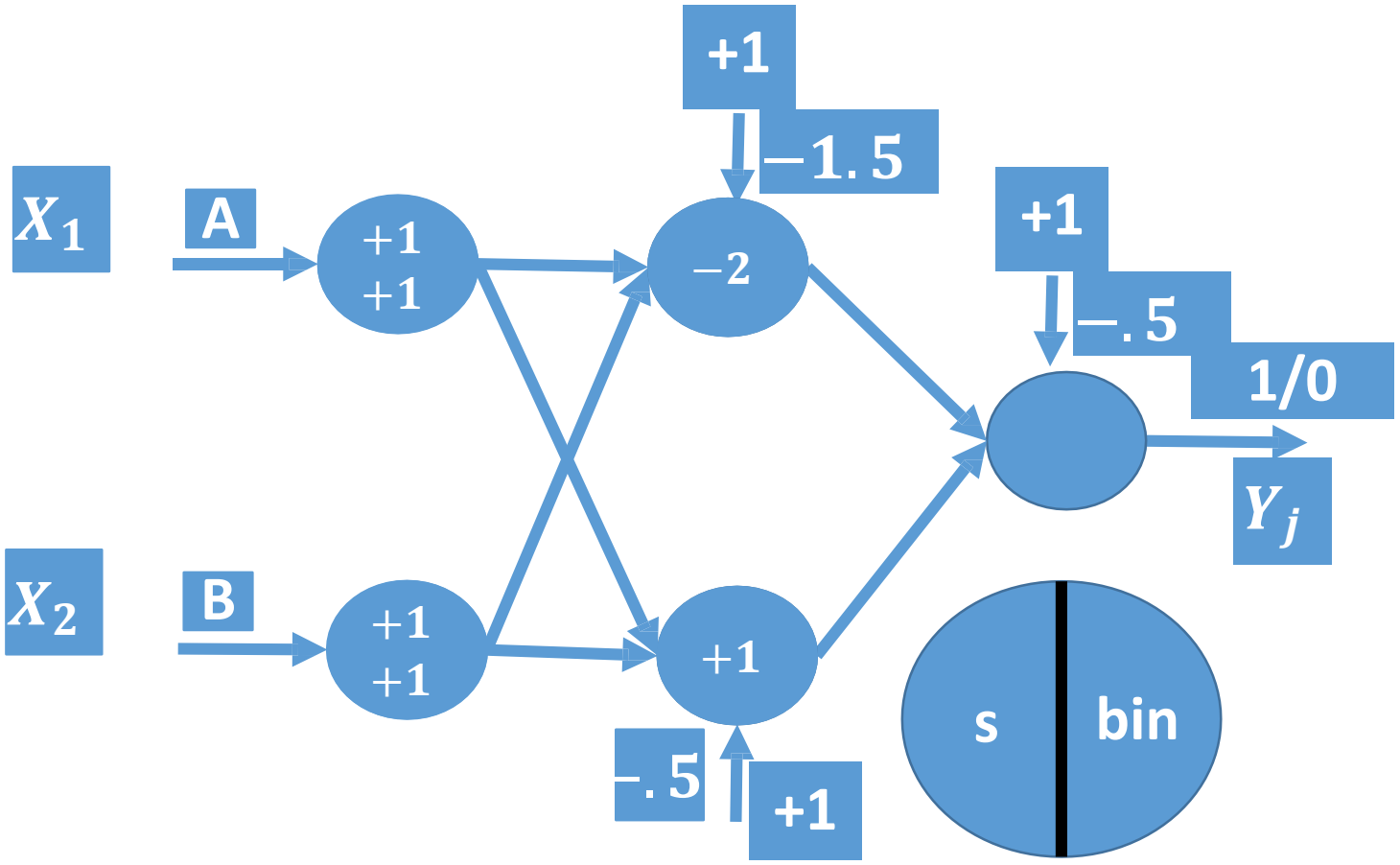
	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



$$S_3=(+1b_3+S_1W_5+S_2W_6)$$
$$=+1*-.5+0*-2+0*1$$
$$=-.5$$

Neural Networks Training
Example Step n=2 – Output – S_3

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

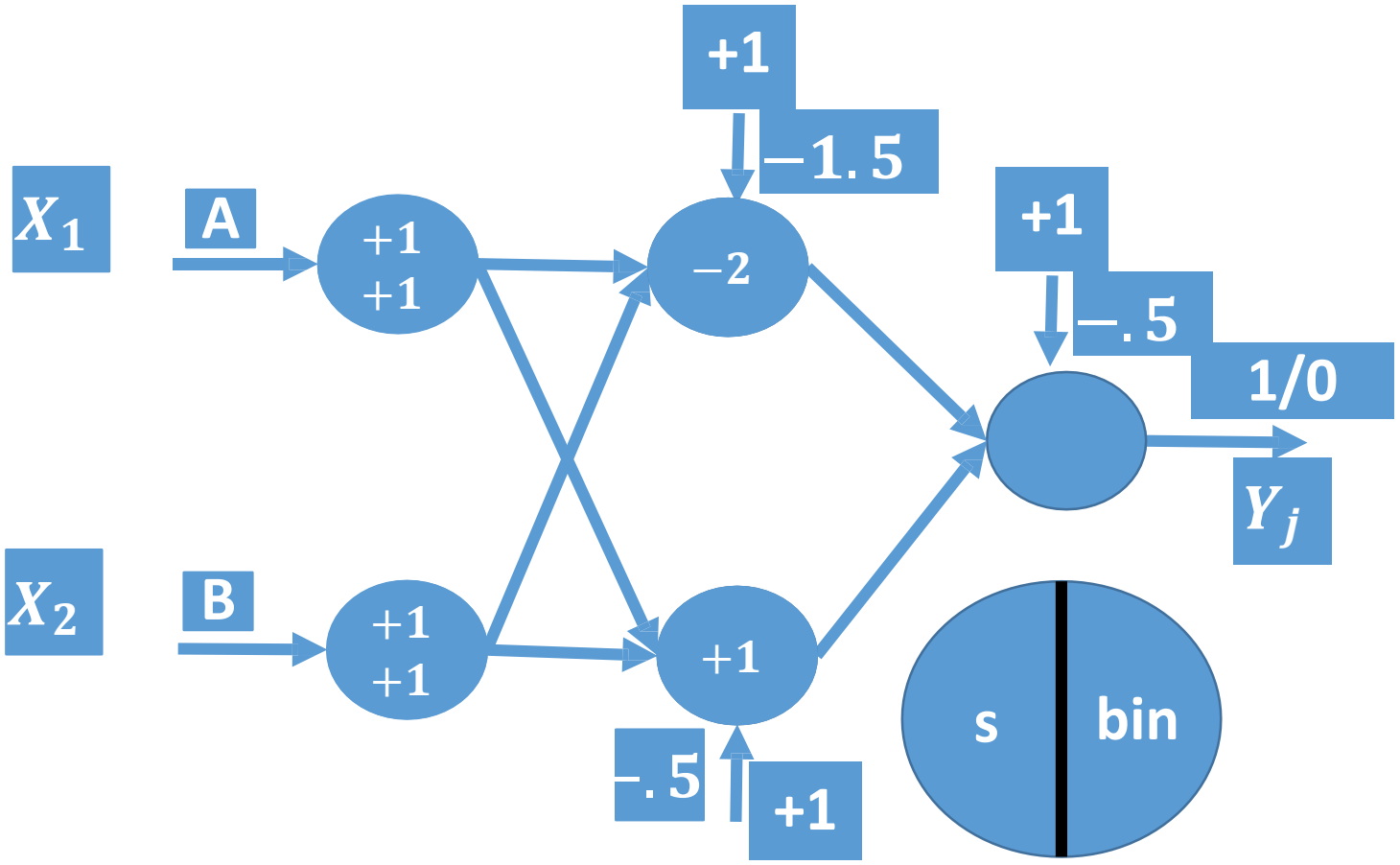


$$Y(S_3) = \\ = BIN(S_3) \\ = BIN(-.5) \\ = 0$$

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

Neural Networks Training Example Step n=2 - Output

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

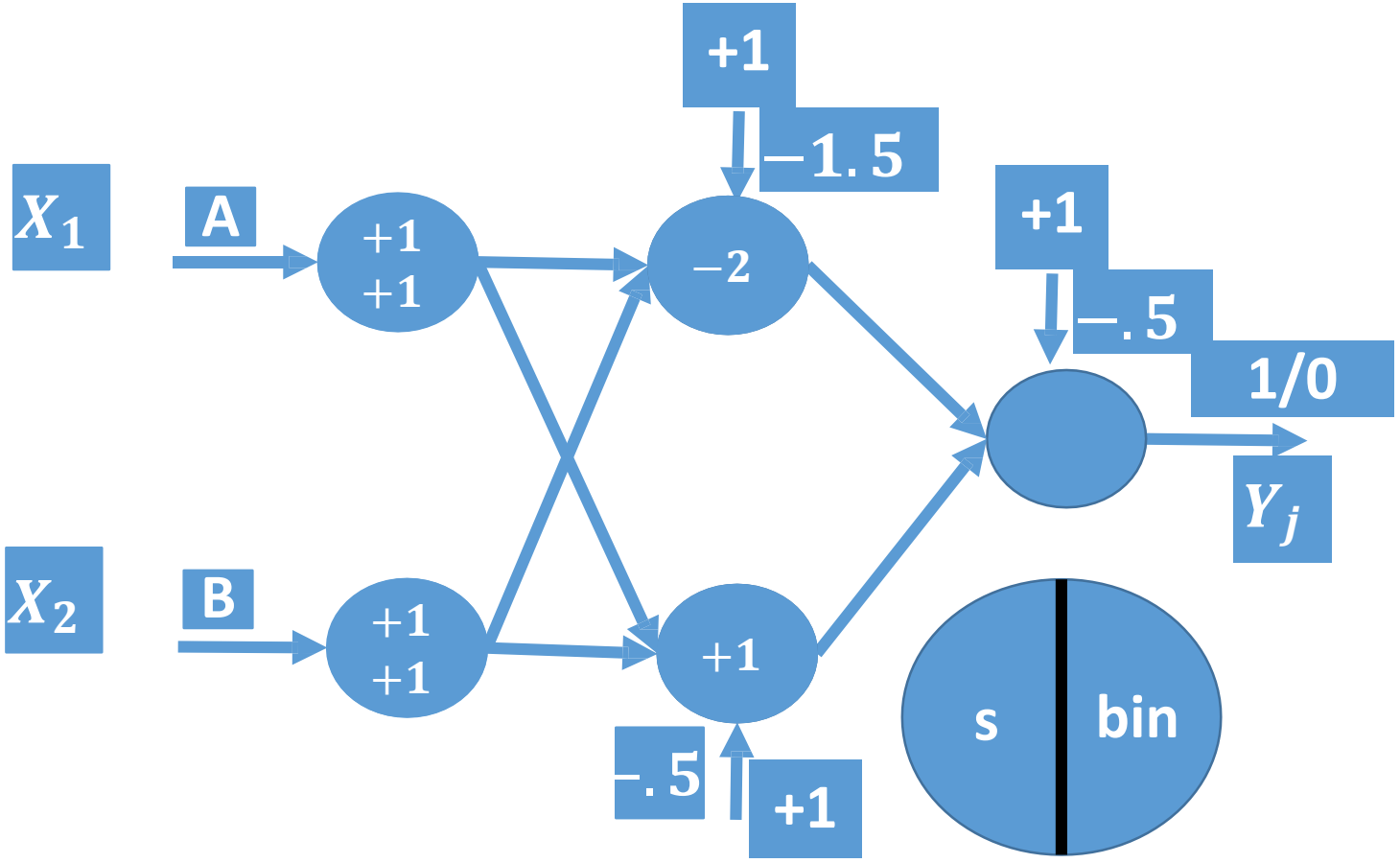


$$Y(n) = Y(2) = Y(S_3) = 0$$

Neural Networks Training Example Step n=2

Predicted Vs. Desired

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



$$Y(n) = Y(2) = 0$$
$$\mathrm{d}(n) = d(2) = 0$$
$$\because Y(n) = d(n)$$
$$\therefore \text{Weights are Correct.}$$
$$\text{No Adaptation}$$

Neural Networks Training Example

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

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

$$n = 3$$

$$\eta = .001$$

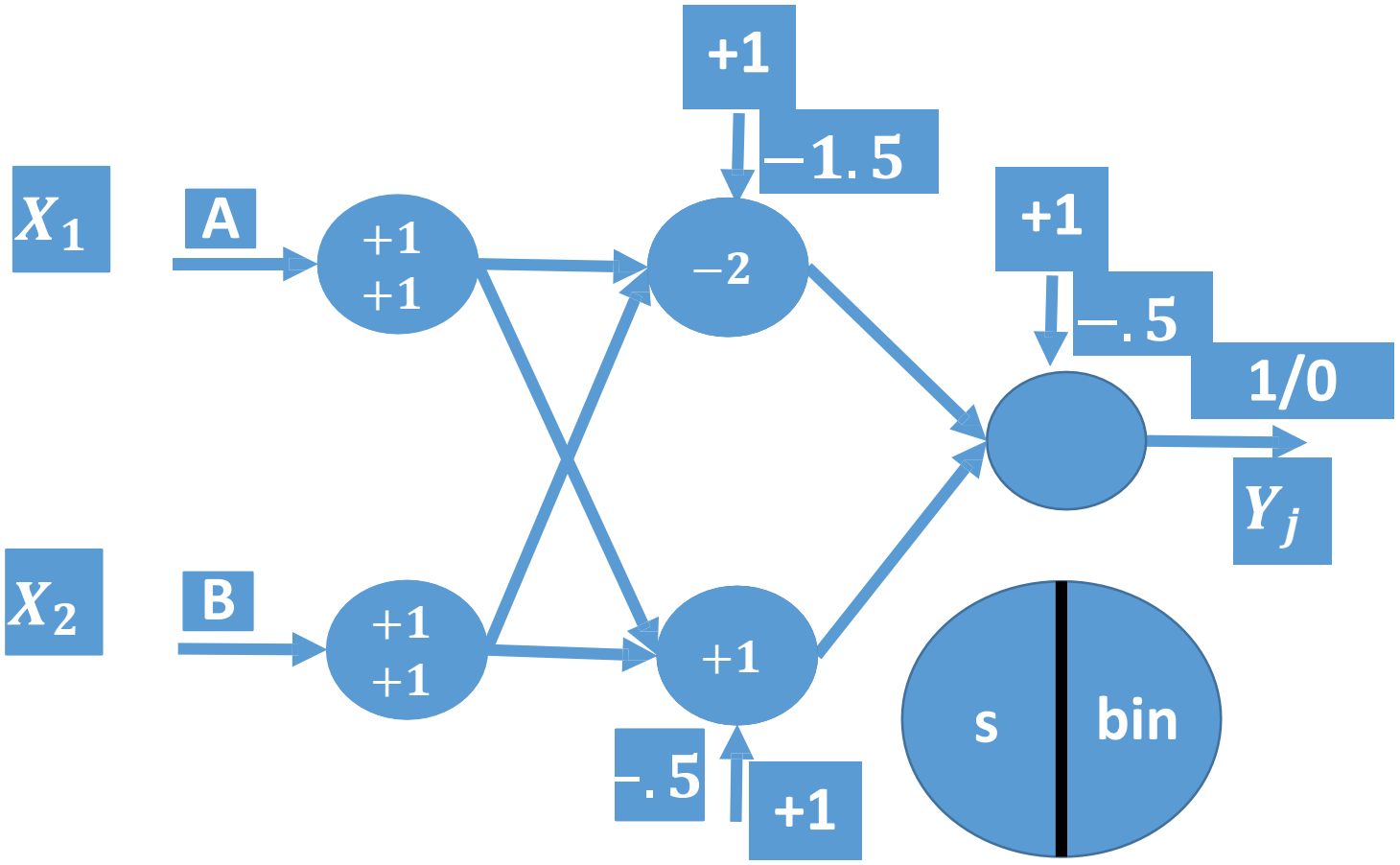
$$X(n) = X(0) = [x_0, x_1, x_2, x_3, x_4] = [+1, +1, +1, 1, 1]$$

$$W(n) = W(3) = W(2) = [\, b_1, b_2, b_3, w_1, w_2, w_3\,, w_4, w_5, w_6] = [-1.5, -0.5, -0.5, 1, 1, 1, 1, -2, 1]$$

$$d(n) = d(3) = 0$$

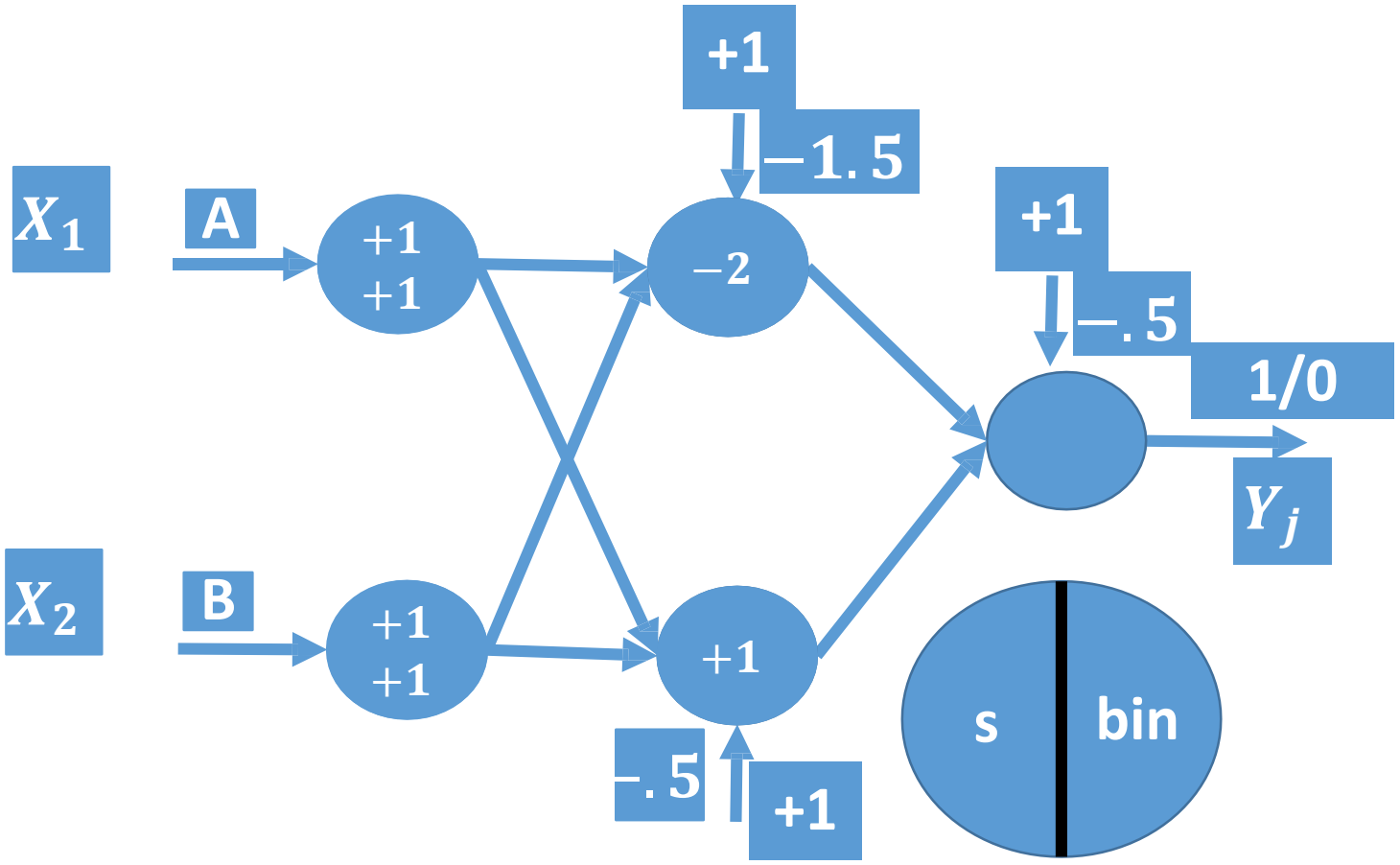
Neural Networks Training Example Step n=3

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



Neural Networks Training Example Step n=3 – SOP – S_1

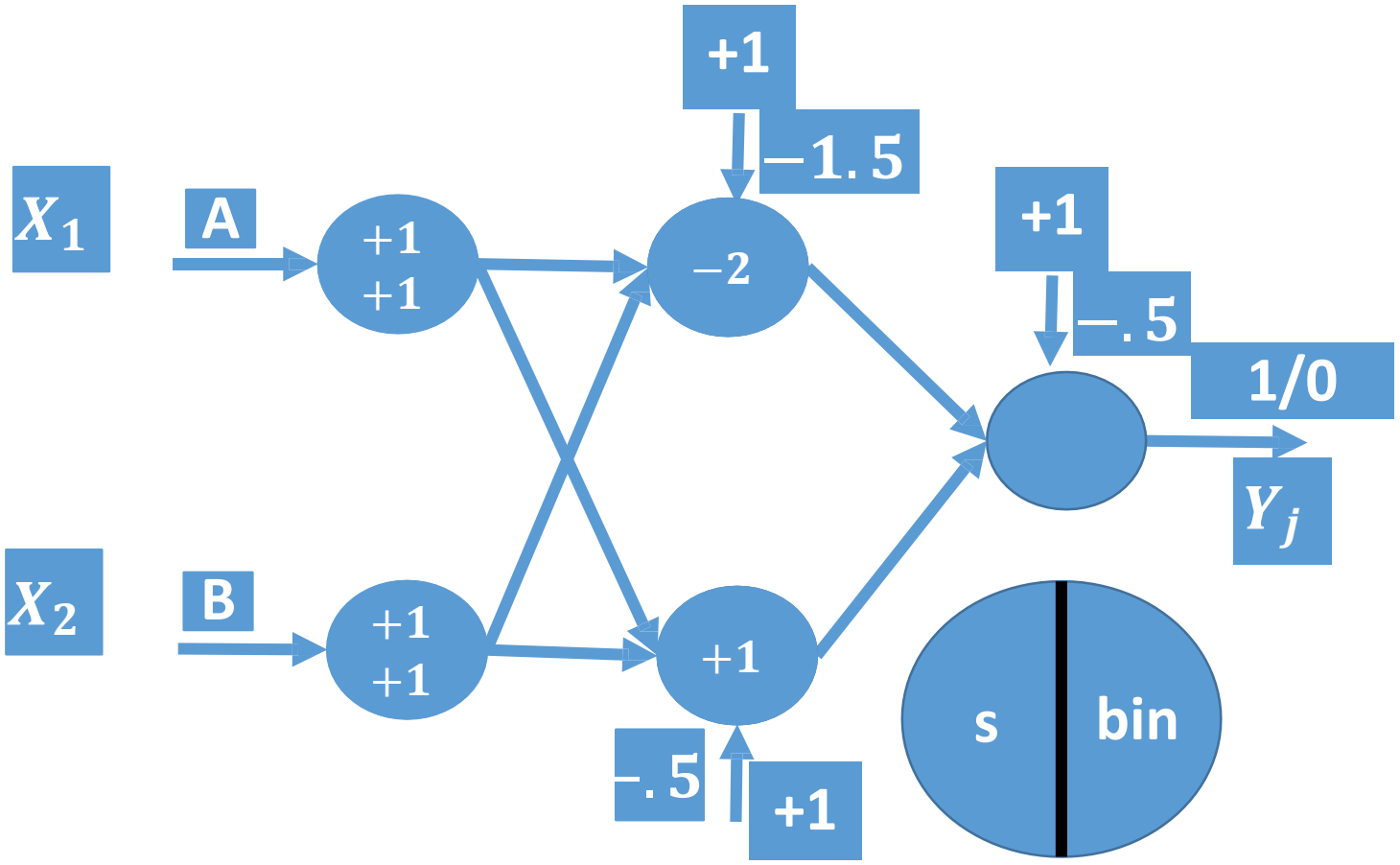
	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



$S_1=(+1b_1+X_1W_1+X_2W_3)$
=+1*-1.5+1*1+1*1
=.5

Neural Networks Training
Example Step n=3 – Output – S_1

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

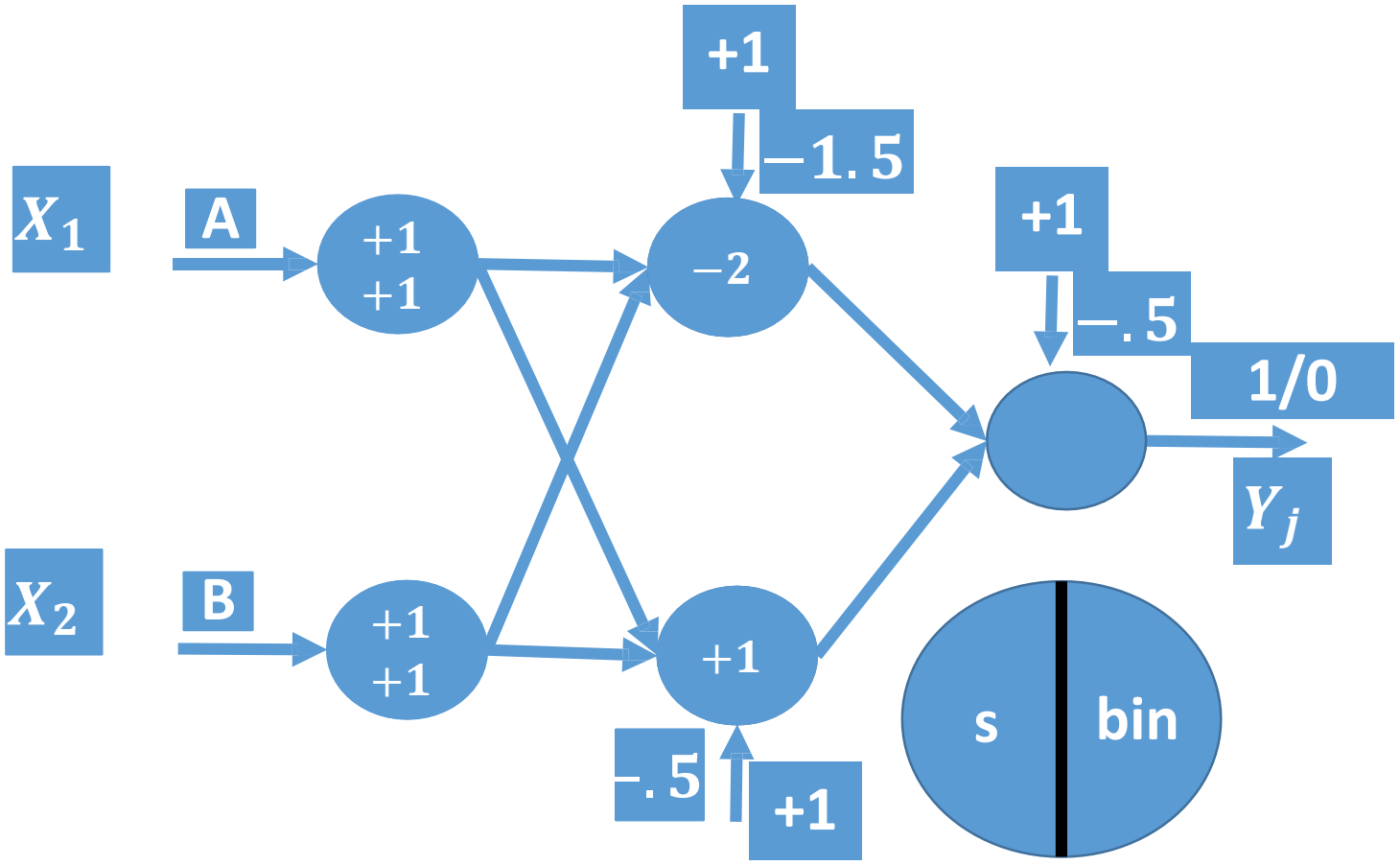


$$Y(S_1) = \\ = BIN(S_1) \\ = BIN(.5) \\ = 1$$

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

Neural Networks Training Example Step n=3 – SOP – S_2

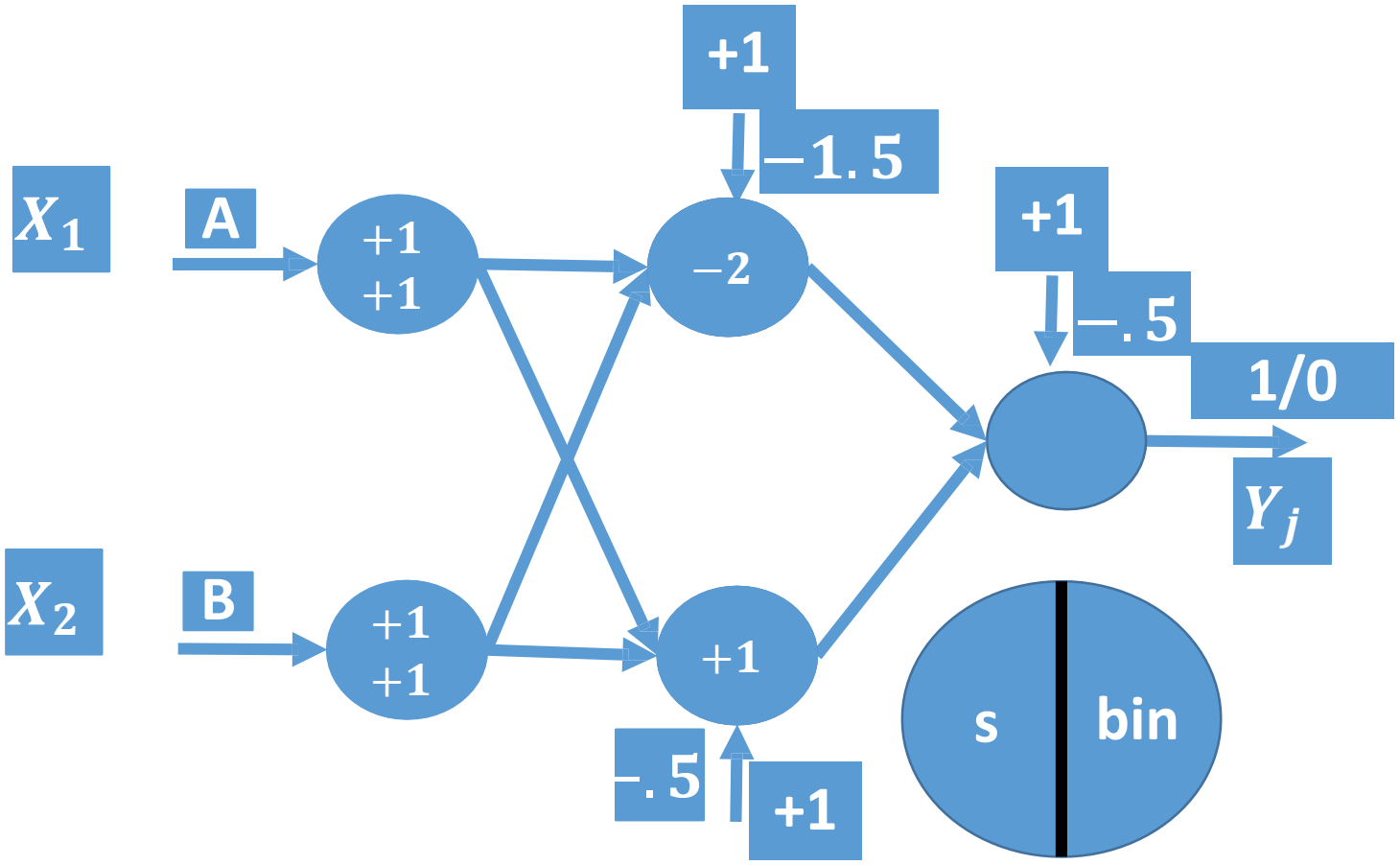
	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



$$S_2=(+1b_2+X_1W_2+X_2W_4)$$
$$=+1*-.5+1*1+1*1$$
$$=1.5$$

Neural Networks Training
Example Step n=3 – Output – S_2

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

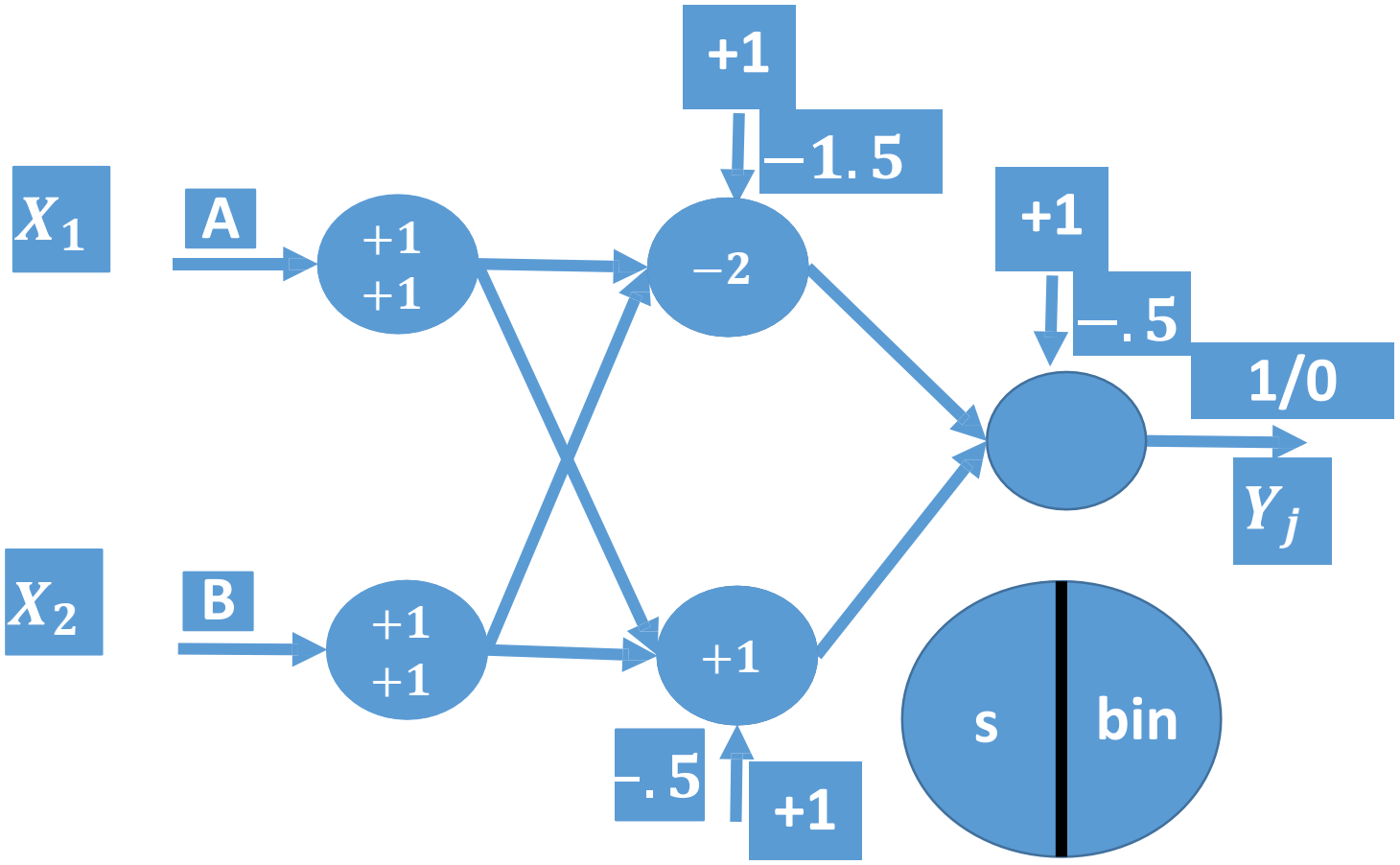


$$Y(S_2) = \\ = BIN(S_2) \\ = BIN(1.5) \\ = 1$$

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

Neural Networks Training Example Step n=3 – SOP – S_3

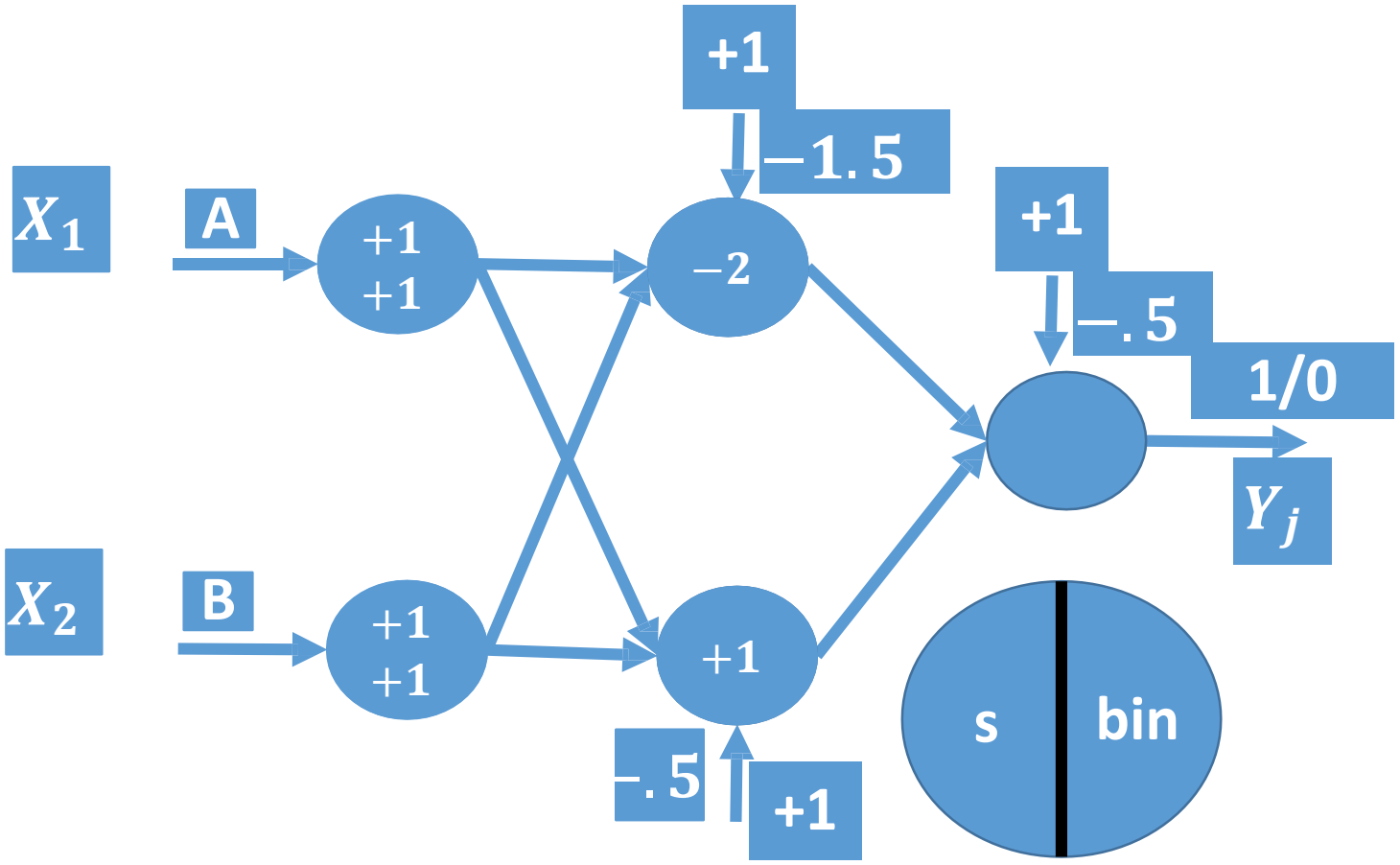
	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



$S_3=(+1b_3+S_1W_5+S_2W_6)$
=+1*-.5+1*-2+1*1
=-1.5

Neural Networks Training Example
Step n=3 – Output – S_3

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

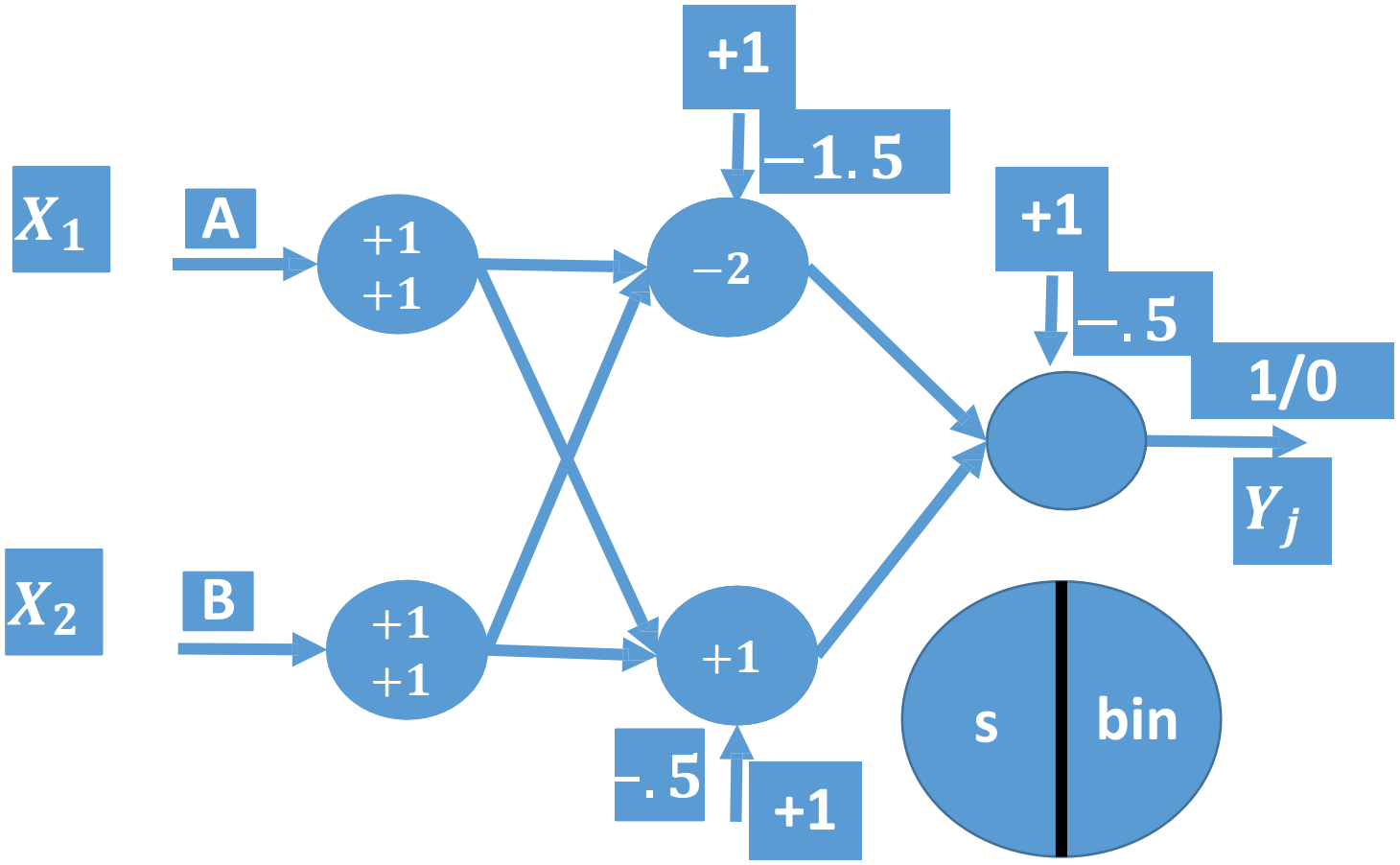


$$Y(S_3) = \\ = BIN(S_3) \\ = BIN(-1.5) \\ = 0$$

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

Neural Networks Training Example Step n=3 - Output

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1

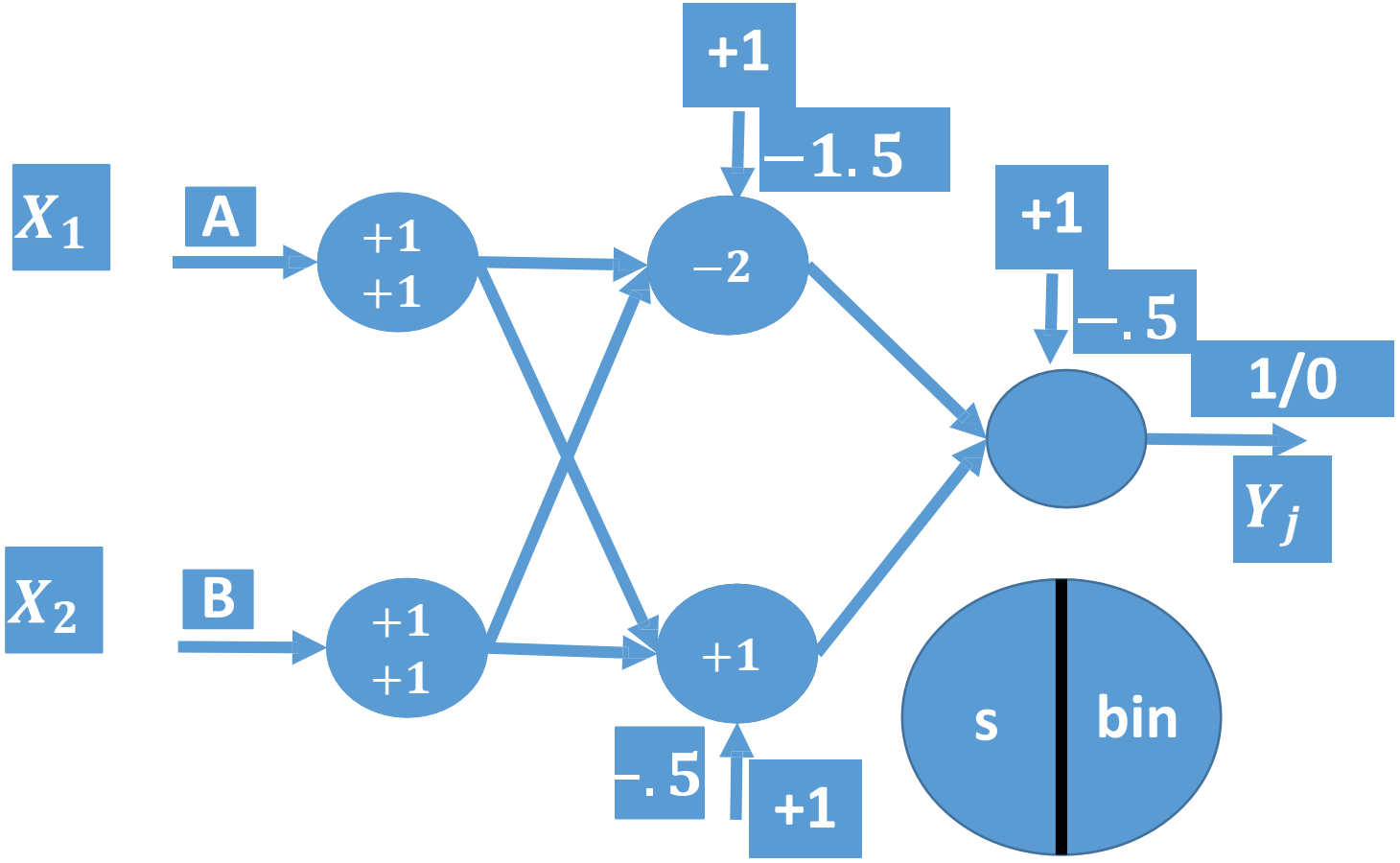


$$Y(n) = Y(3) = Y(S_3) = 0$$

Neural Networks Training Example Step n=3

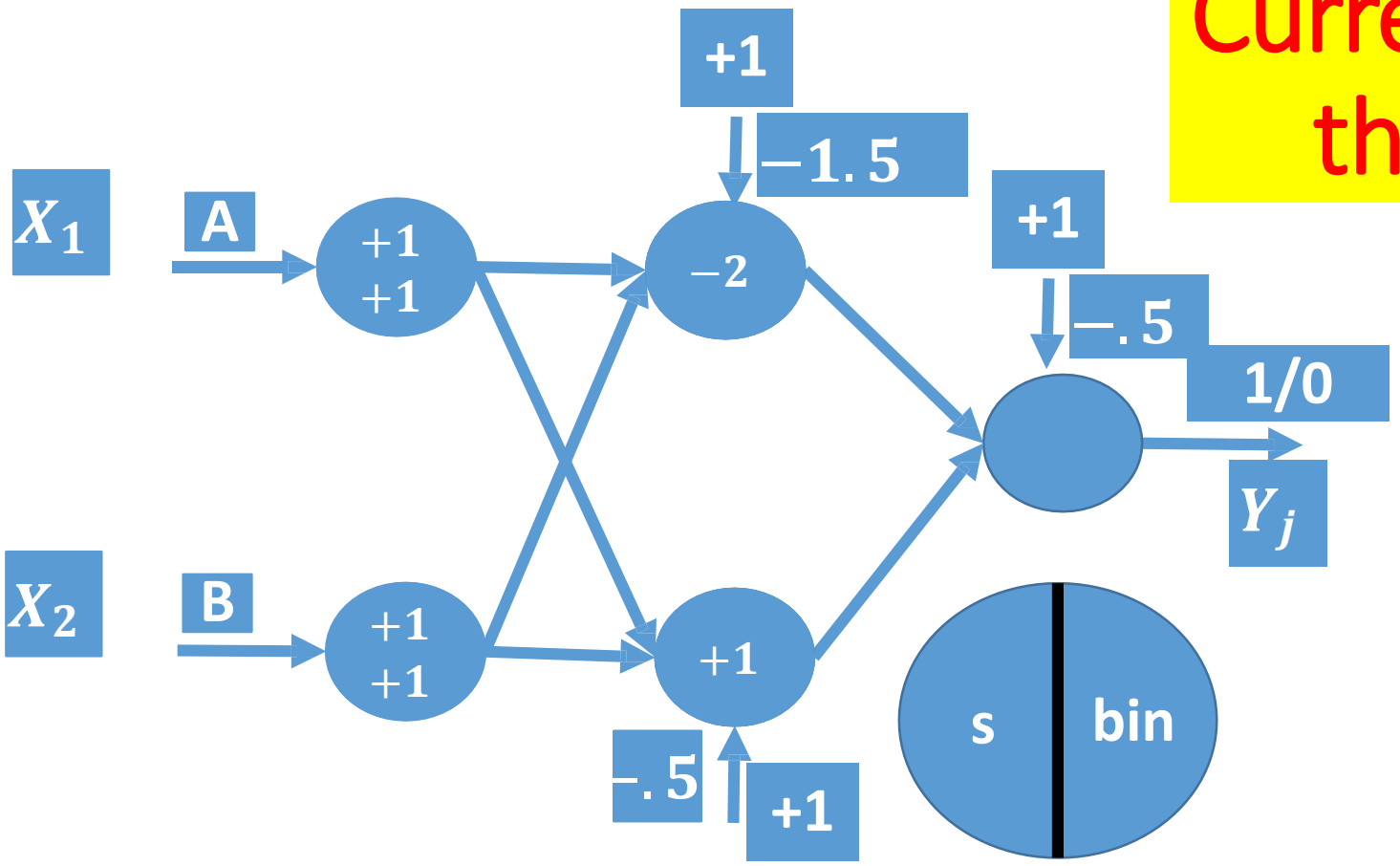
Predicted Vs. Desired

	A	B
1 => 1	1	0
	0	1
0 => 0	0	0
	1	1



$$Y(n) = Y(3) = 0$$
$$\mathrm{d}(n) = d(3) = 0$$
$$\because Y(n) = d(n)$$
∴Weights are Correct.
No Adaptation

Final Weights



Current weights predicted the desired outputs.