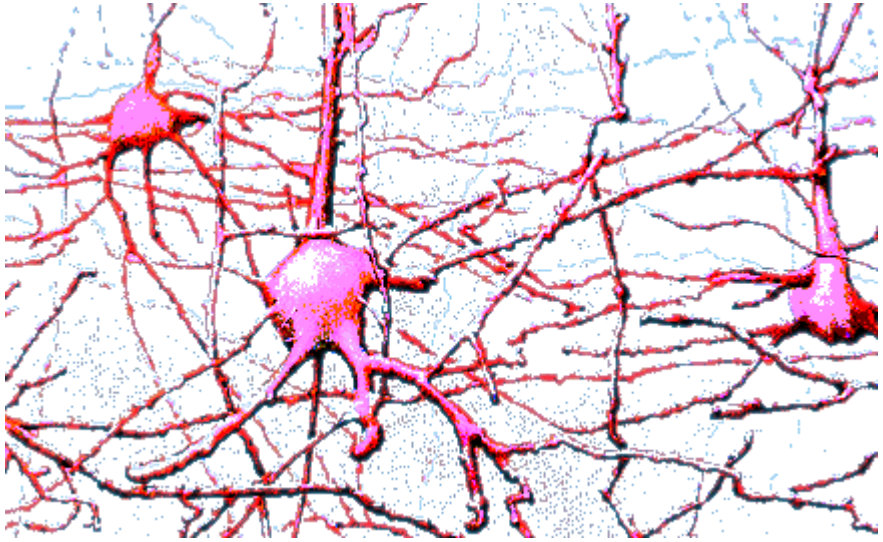
Neural Network System

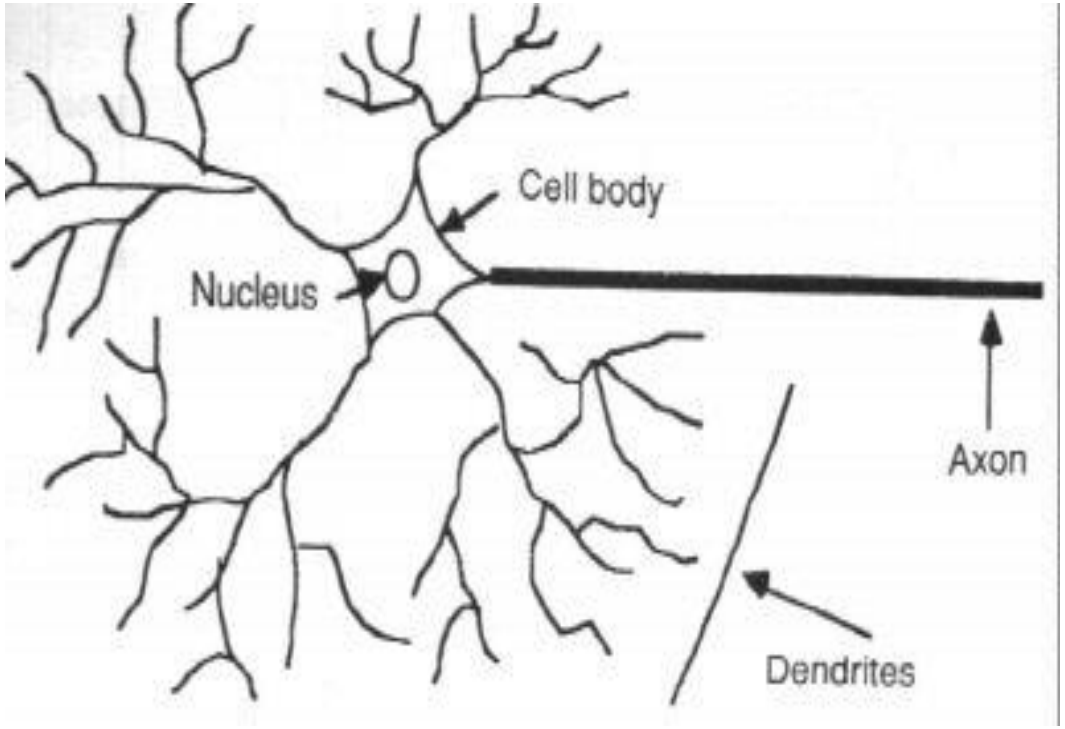
Artificial Intelligence: Biological Neuron

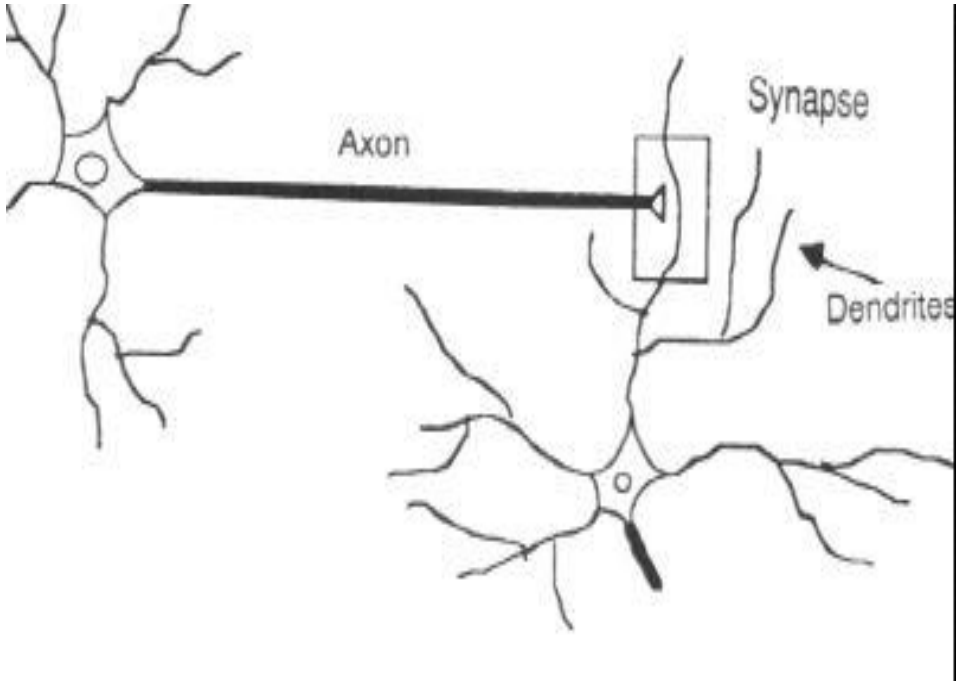
- Animals are able to react adaptively to changes in their external and internal environment, and they use their nervous system to perform these behaviours.
- An appropriate model/simulation of the nervous system should be able to produce similar responses and behaviours in artificial systems.

John McCarthy “1956”



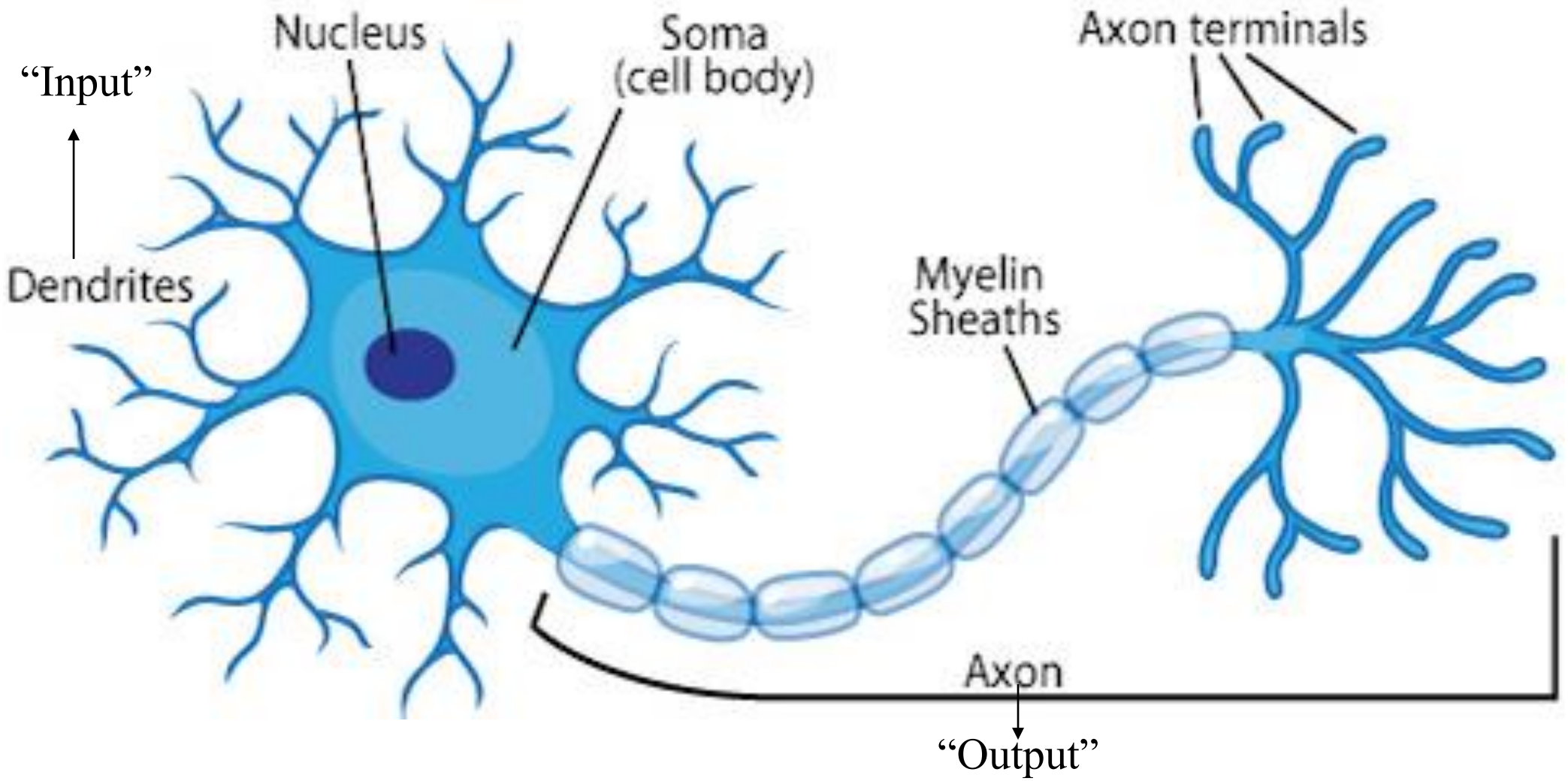
Biological Neuron





The information transmission happens at the synapses

Biological Neuron



Biological Neuron

- A **human brain has billions of neurons** that are interconnected and they are **involved in processing and transmitting** chemical and electrical signals.
- **Dendrites receive information** from other neurons.
- **Cell nucleus processes the information** received from dendrites.
- **Axon** is a cable that is **used by neurons to send information**.
- **Multiple signals arrive at the dendrites** which, are integrated into the cell body.
- If **the accumulated signal exceeds a certain threshold**, **an output signal is generated (i.e. neuron is fired)** that will be passed on by the axon.

Artificial Neuron

- The **Warren McCullock and Walter Pitts** created a computational model for neural networks in 1943 based on **threshold logic**. And it was called as **McCullock-Pitts (MCP) neuron**.
- An **artificial neuron is a mathematical function** that takes inputs along with the parameters or weights, sums them up and **passes this sum through a nonlinear function** (e.g. logistic) to produce output.
- Every **neuron holds an internal state called activation** signal.
- Each **connection carries information** about the input signal.
- Every **neuron is connected to another neuron via connection** link.

Artificial Intelligence

- Ability to learn and take decision using past experience
- Machine can perform repetitive task accurately and efficiently

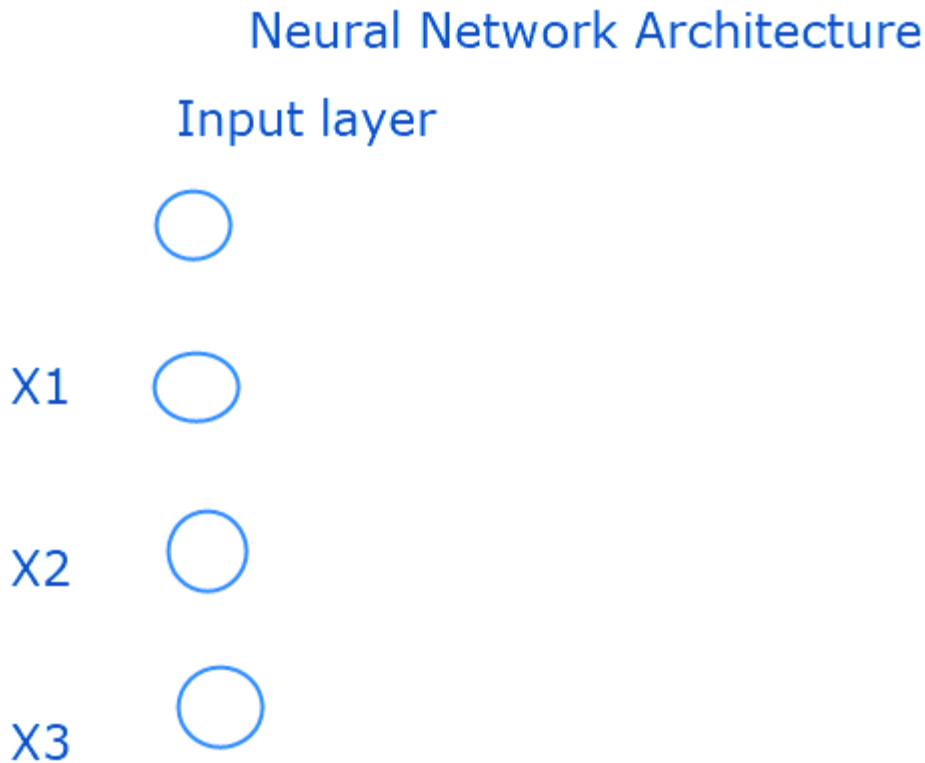
Brain  Machines

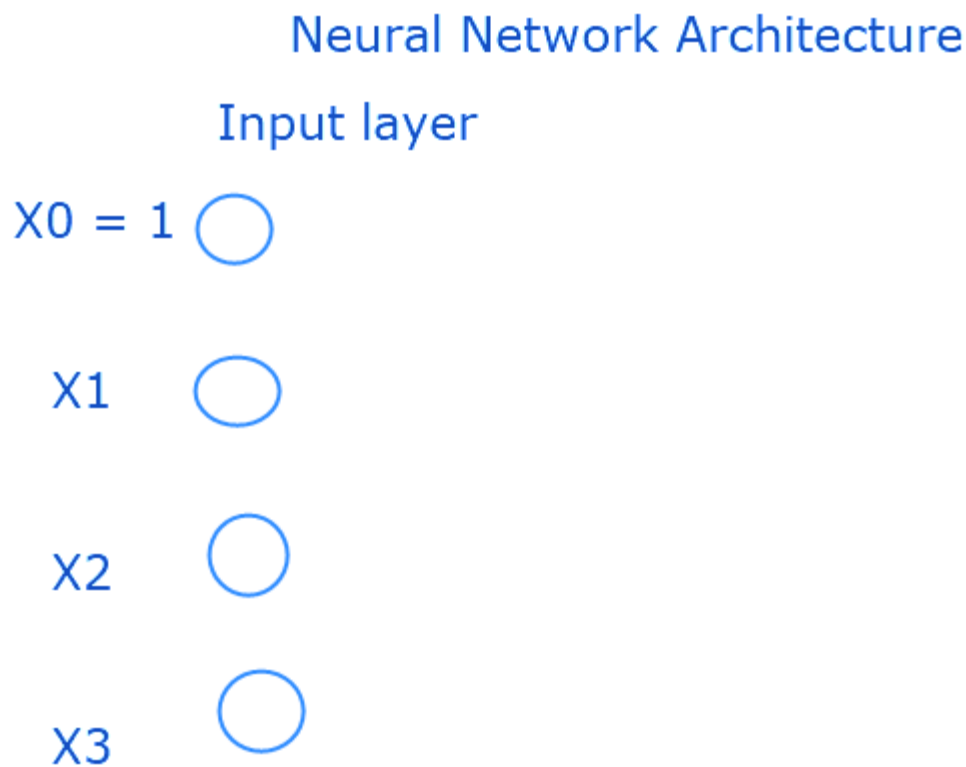
Neural Network Learn by day to day experience “Human Intelligence”

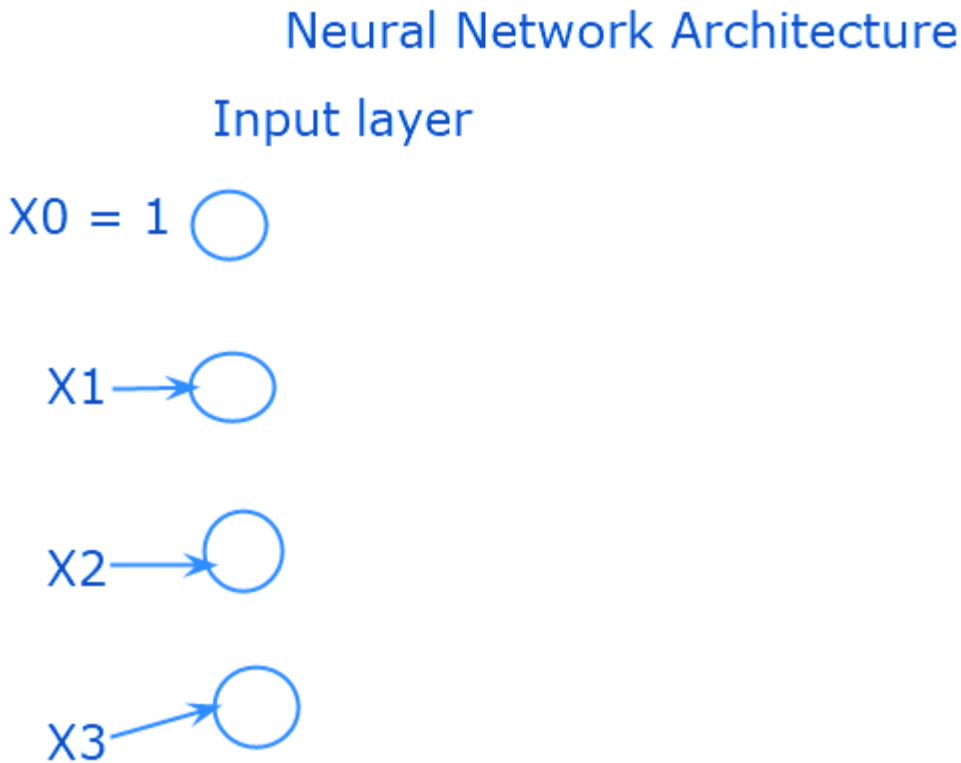
Fixed set of instruction

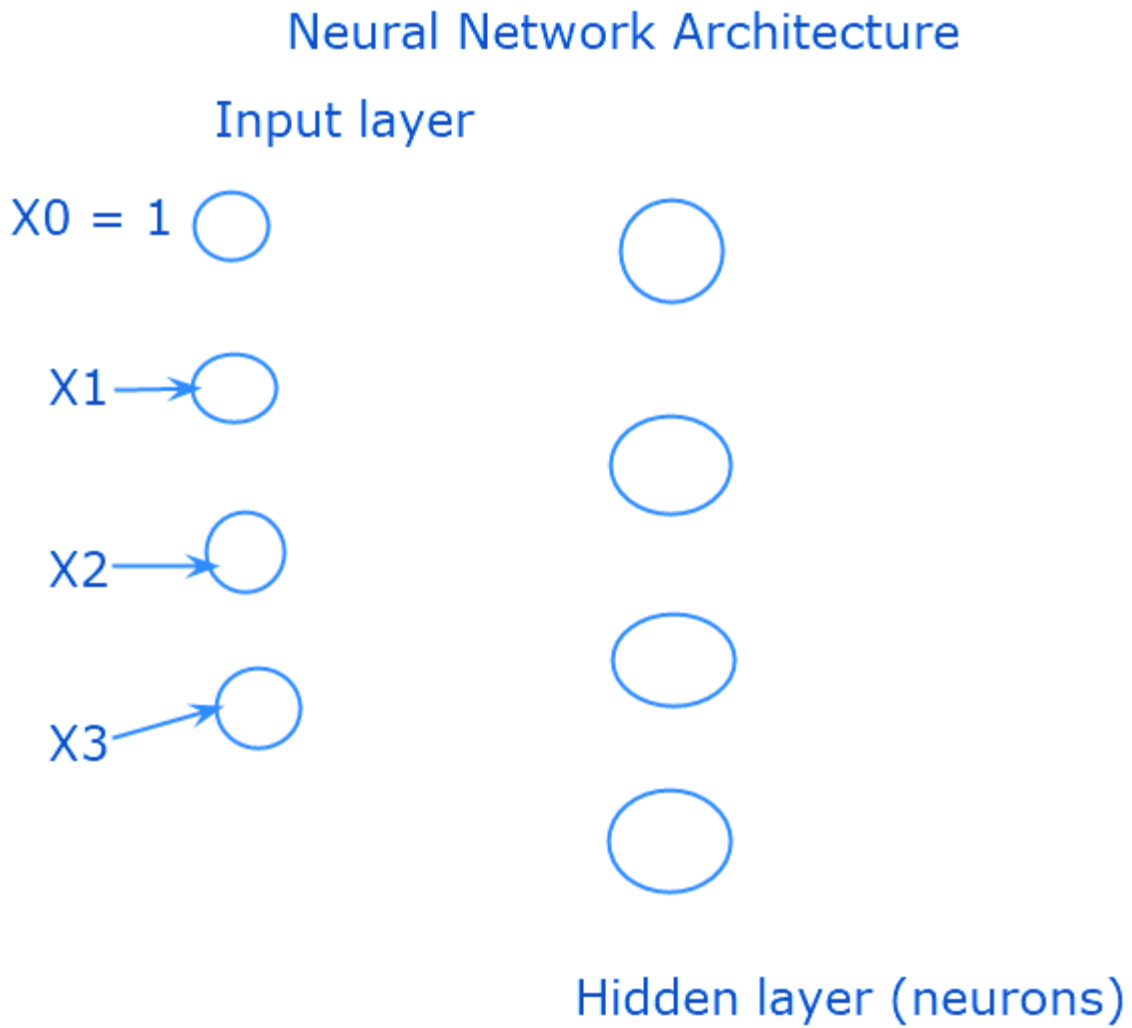
Neural Network Architecture

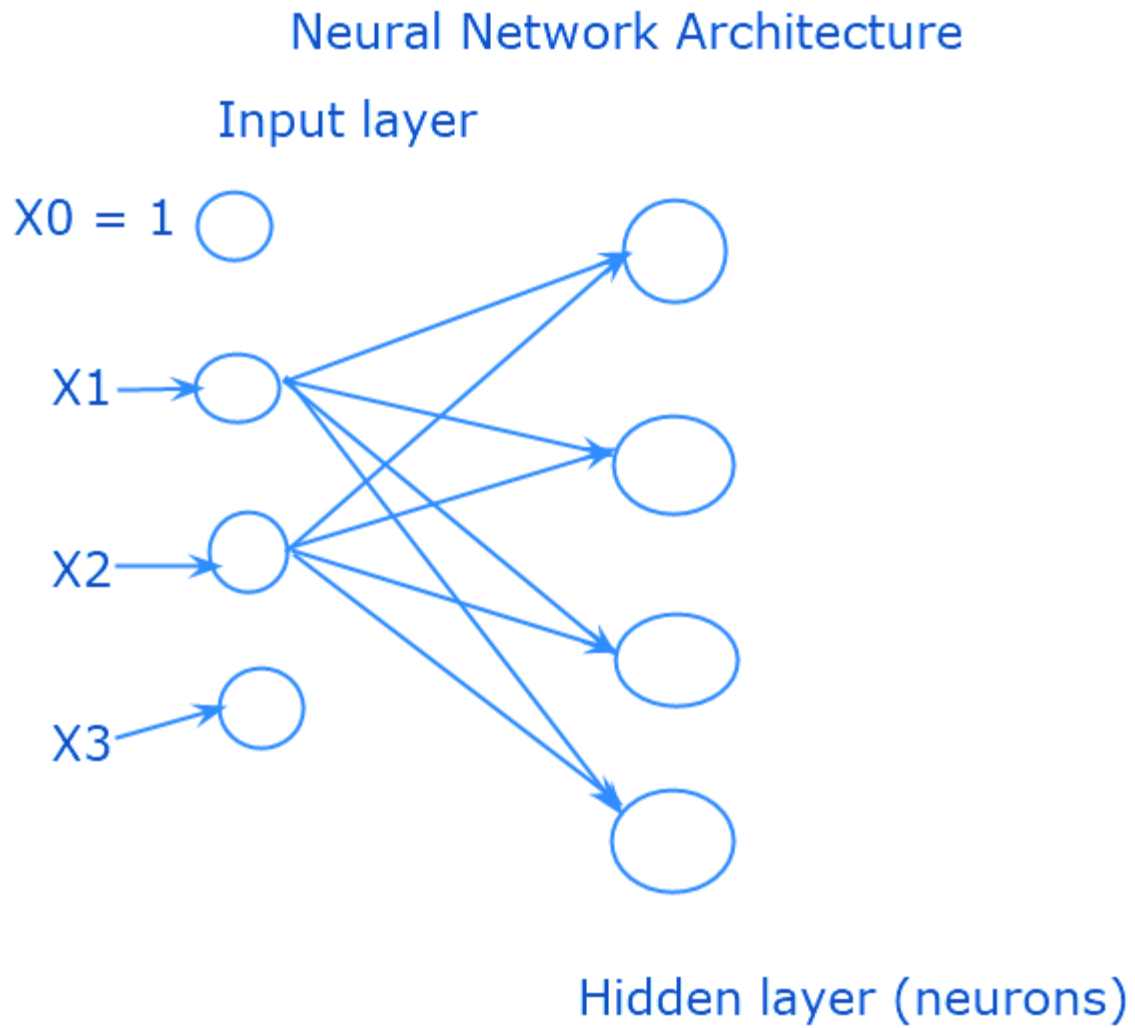
Neural Network Architecture

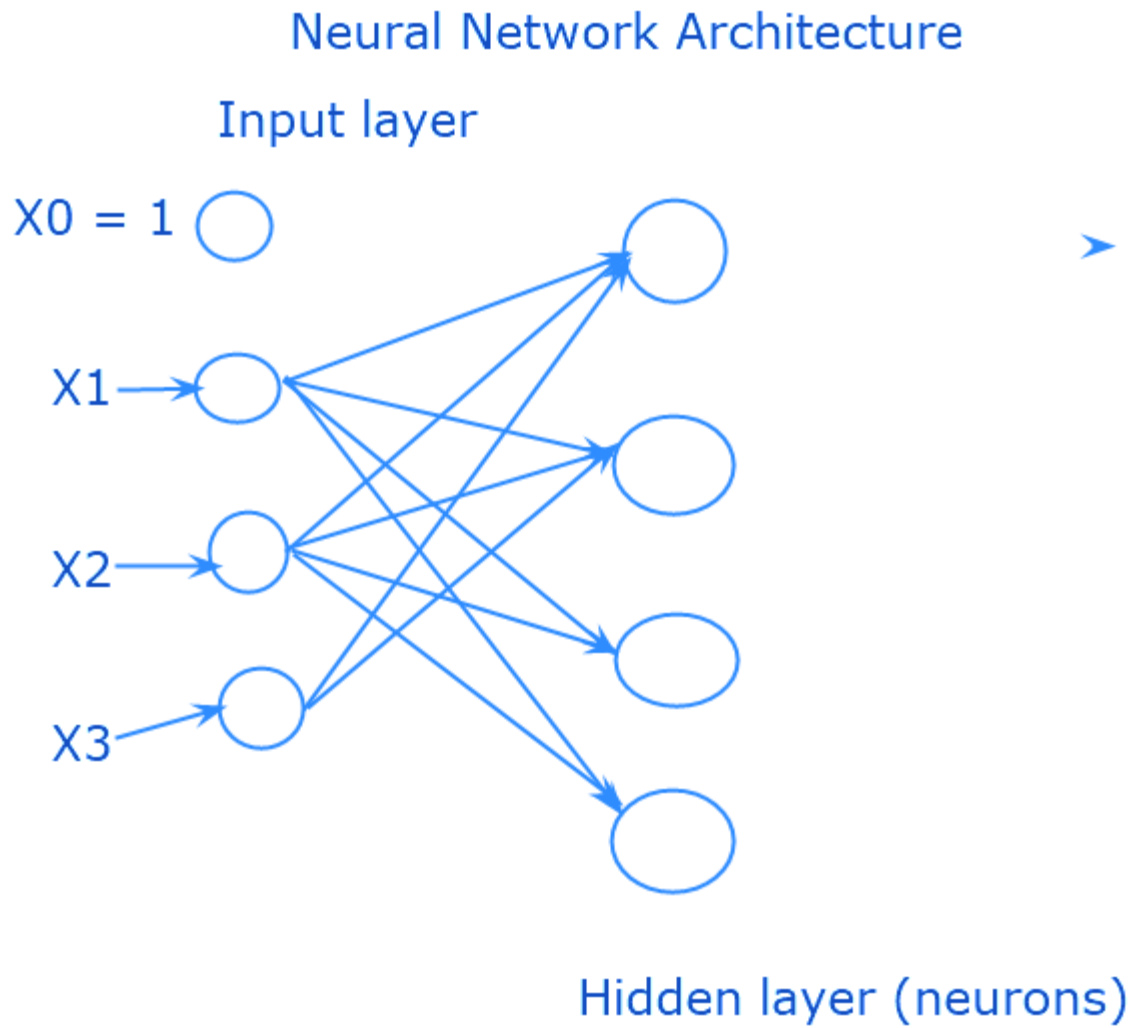

Neural Network Architecture
Input layer
X1
X2
X3

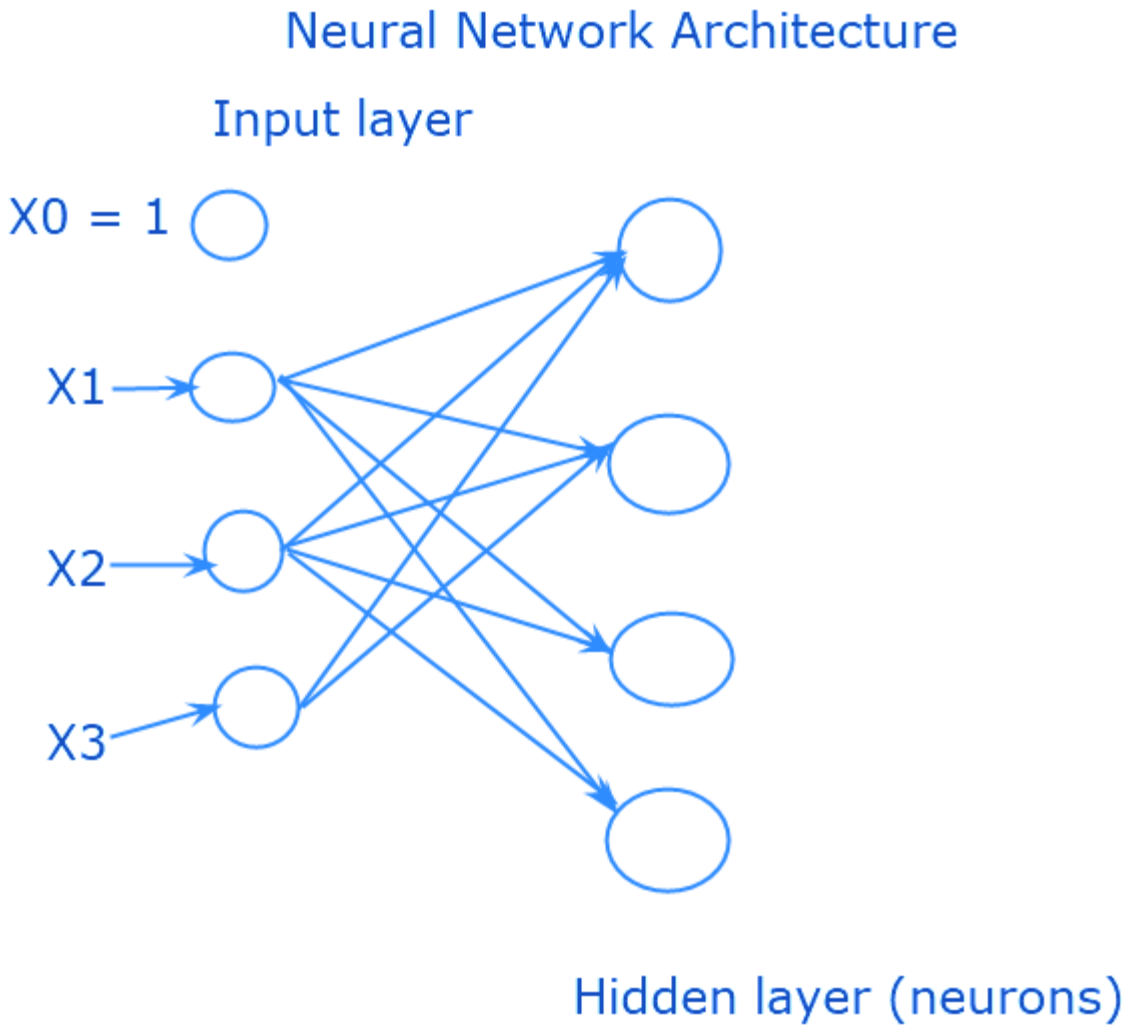

Neural Network Architecture
Input layer
X0 = 1
X1
X2
X3

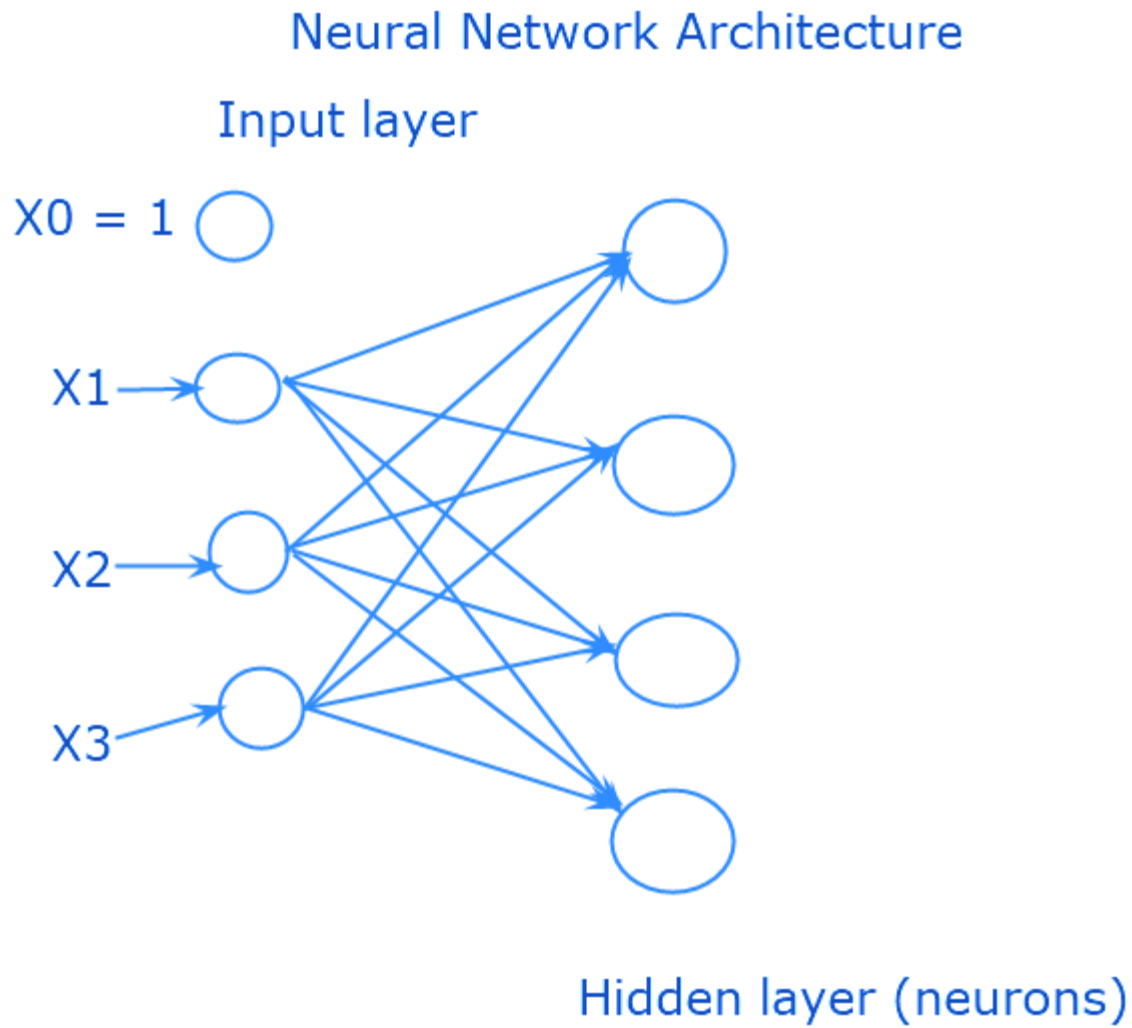

Neural Network Architecture
Input layer
X0 = 1
X1
X2
X3

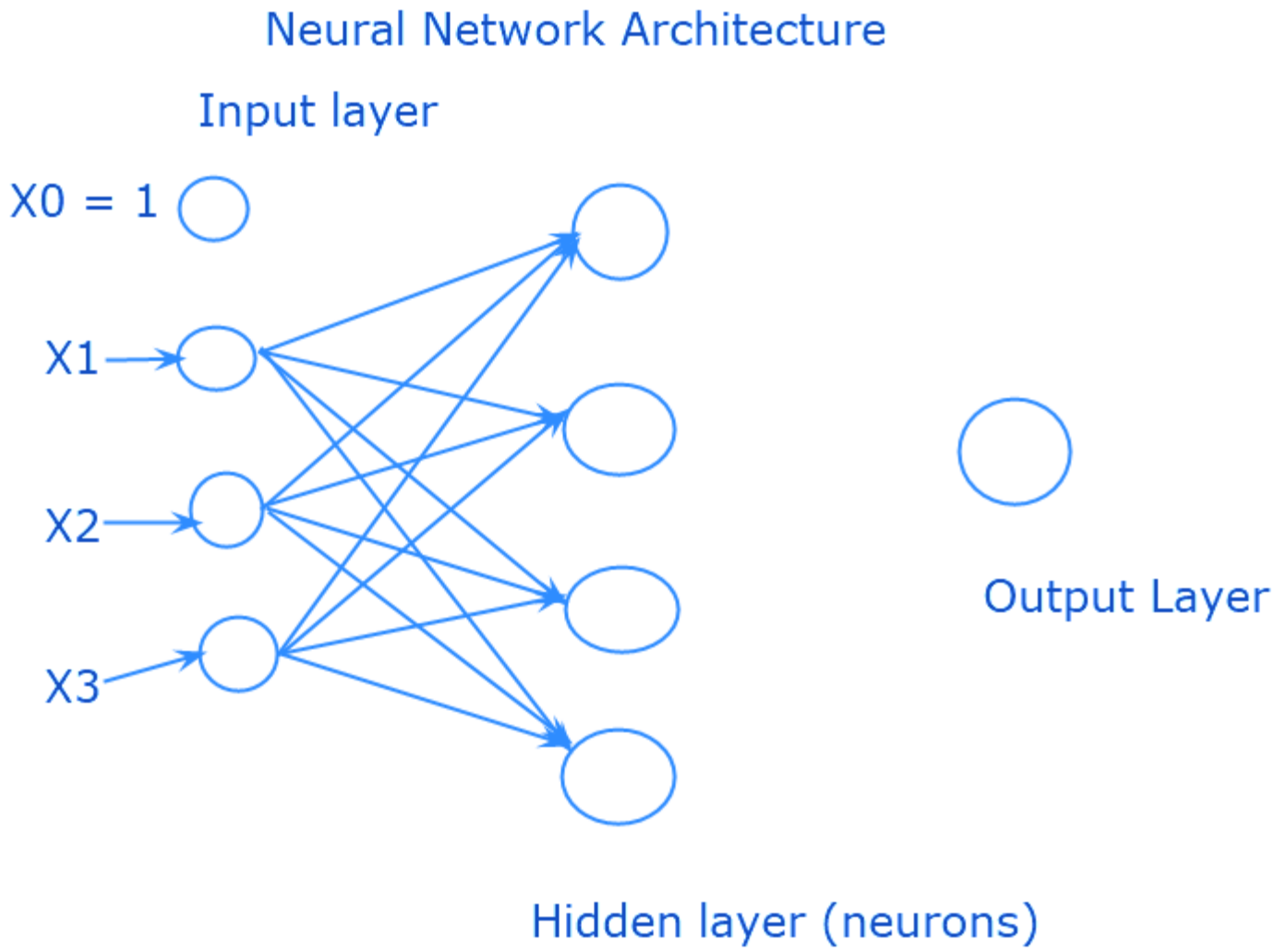

Neural Network Architecture
Input layer
X0 = 1
X1
X2
X3
Hidden layer (neurons)

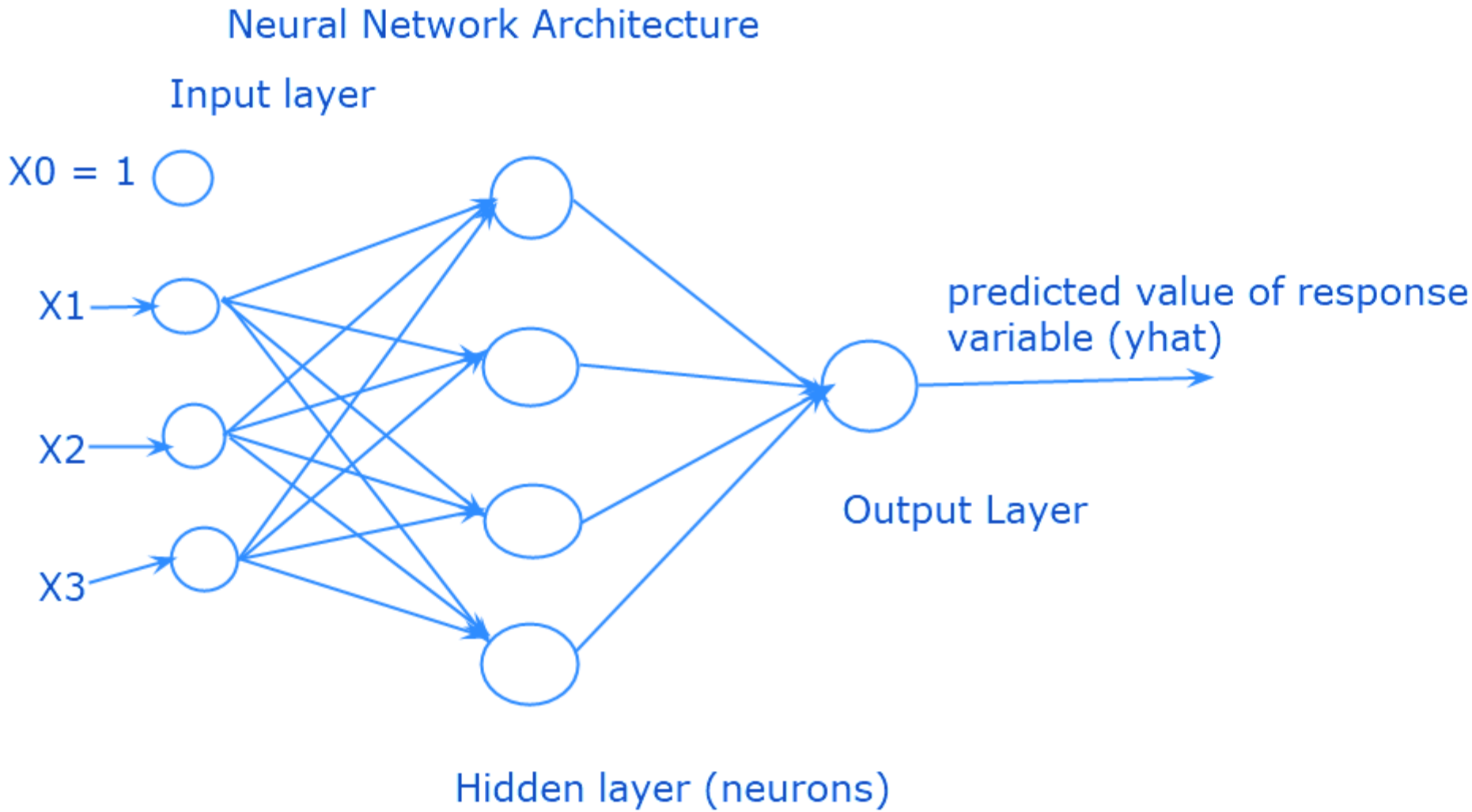

Neural Network Architecture
Input layer
X0 = 1
X1
X2
X3
Hidden layer (neurons)

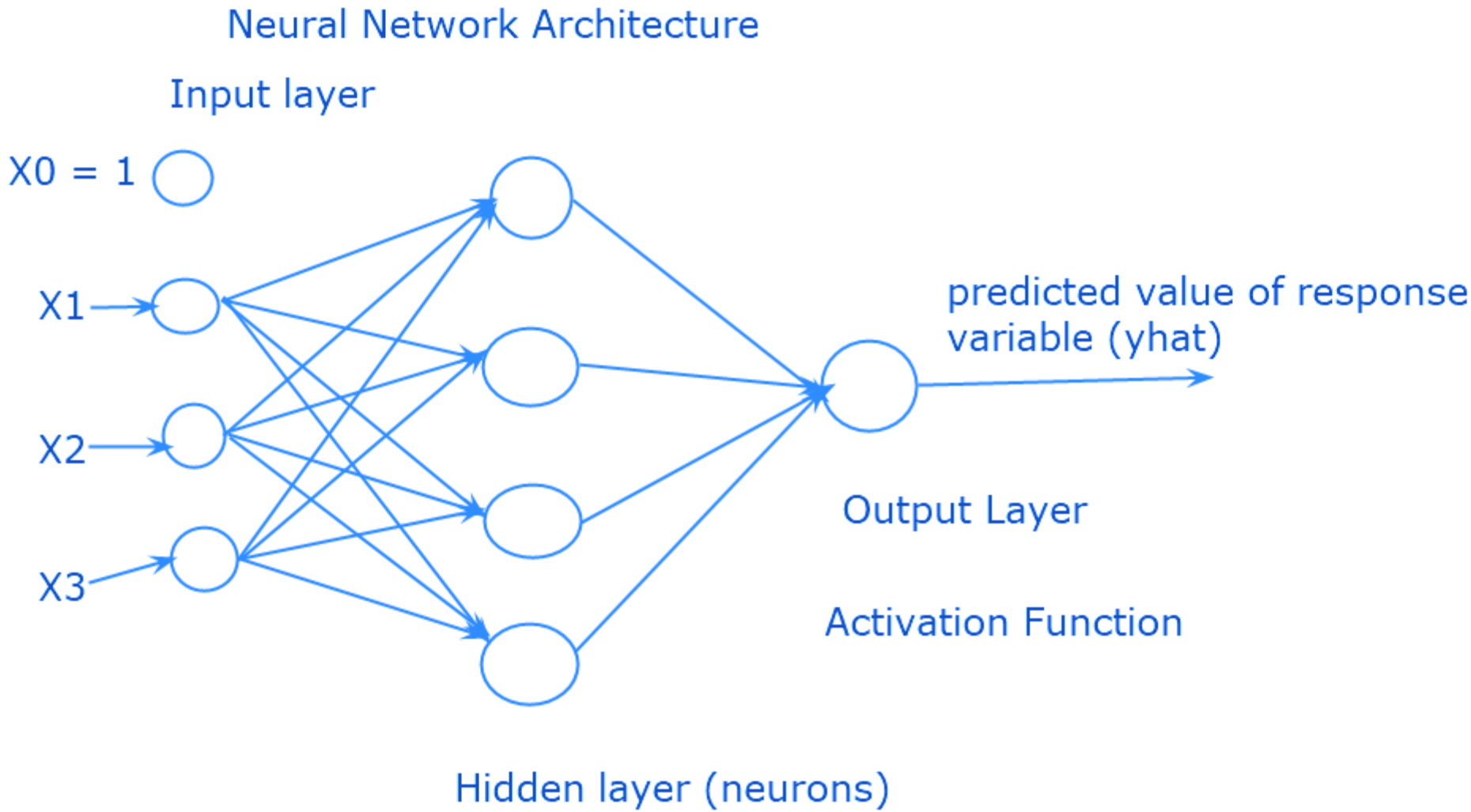

Neural Network Architecture
Input layer
X0 = 1
X1
X2
X3
Hidden layer (neurons)

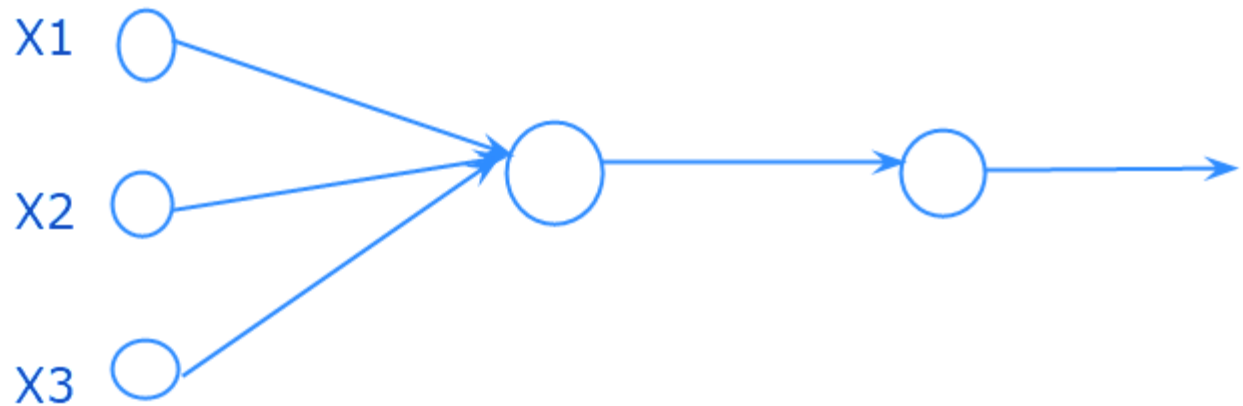

Neural Network Architecture
Input layer
X0 = 1
X1
X2
X3
Hidden layer (neurons)

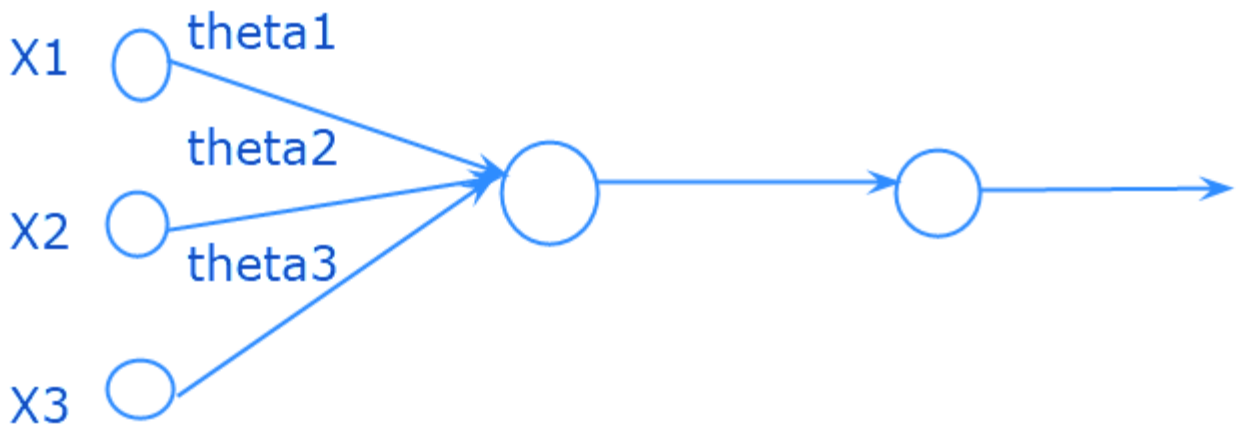

Neural Network Architecture
Input layer
X0 = 1
X1
X2
X3
Hidden layer (neurons)

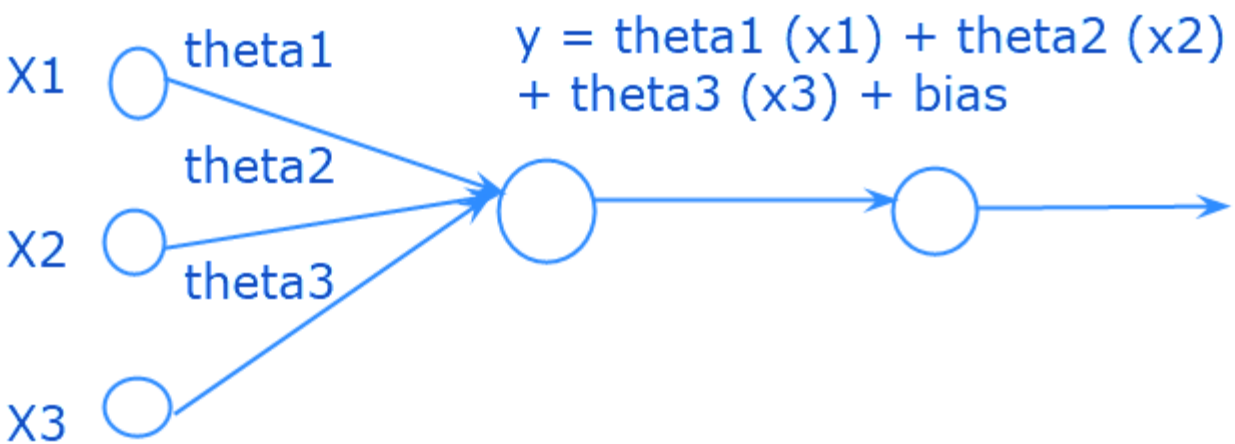

Neural Network Architecture
Input layer
X0 = 1
X1
X2
X3
Output Layer
Hidden layer (neurons)

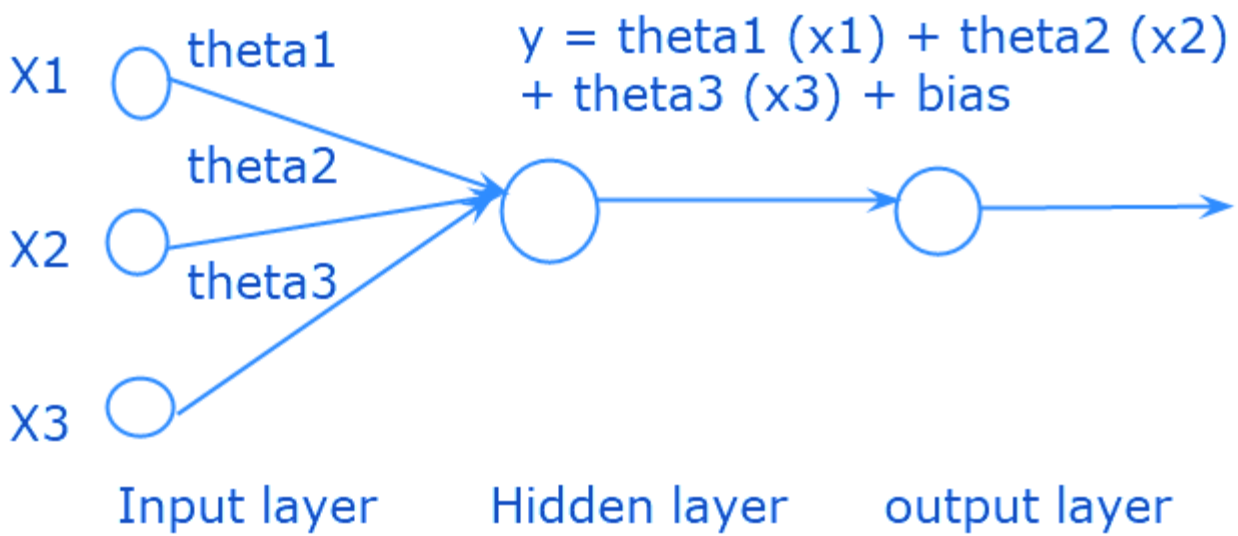

Neural Network Architecture
Input layer
X0 = 1
X1
X2
X3
predicted value of response variable (yhat)
Output Layer
Hidden layer (neurons)

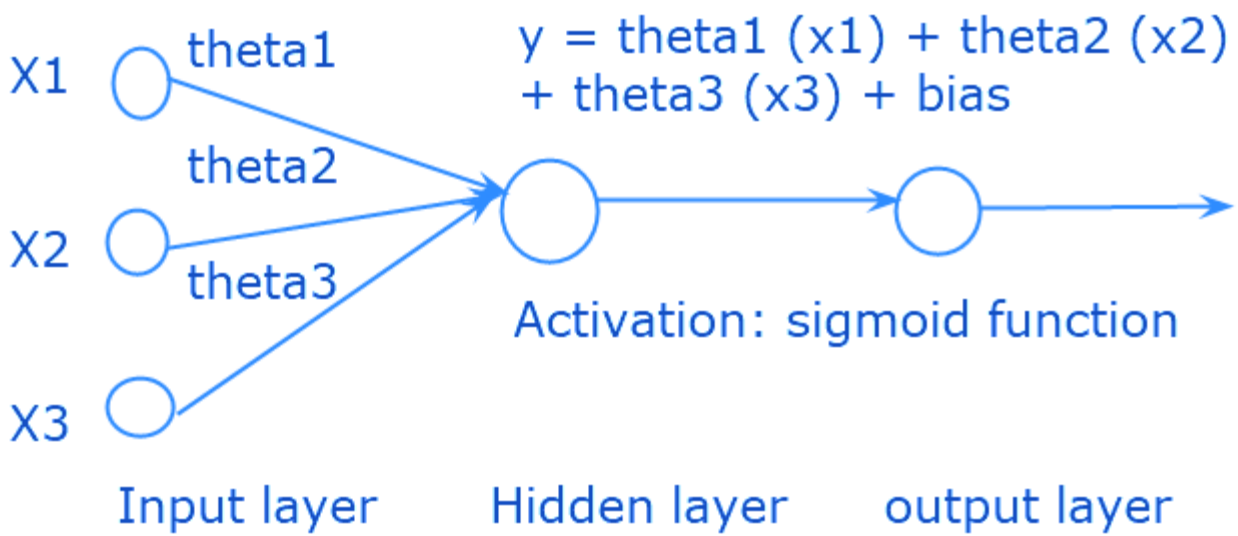

Neural Network Architecture
Input layer
X0 = 1
X1
X2
X3
predicted value of response variable (yhat)
Output Layer
Activation Function
Hidden layer (neurons)

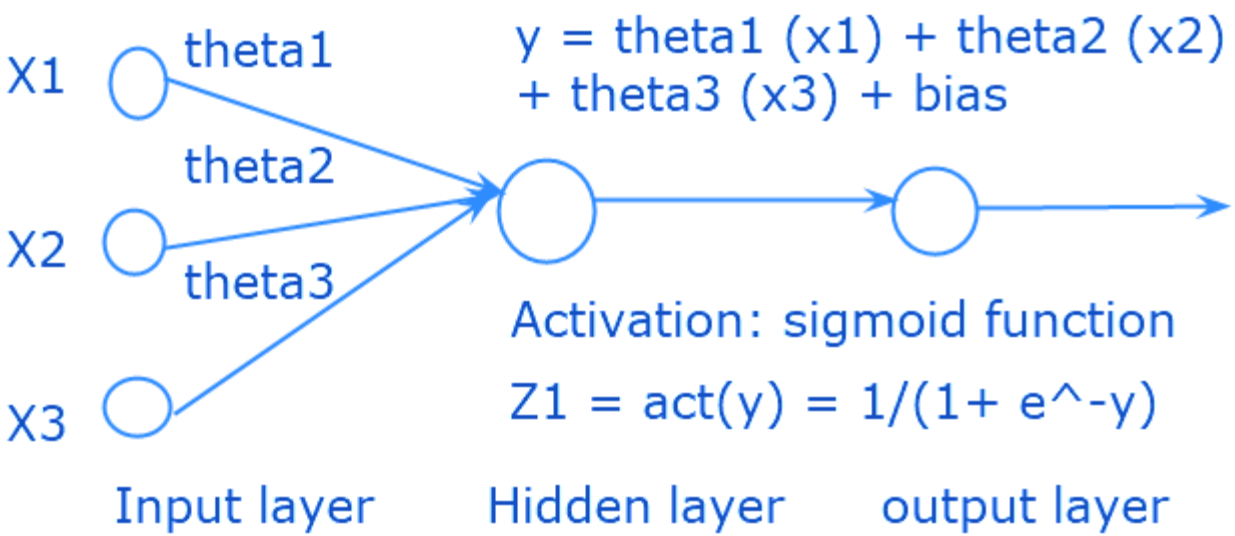

X1
X2
X3

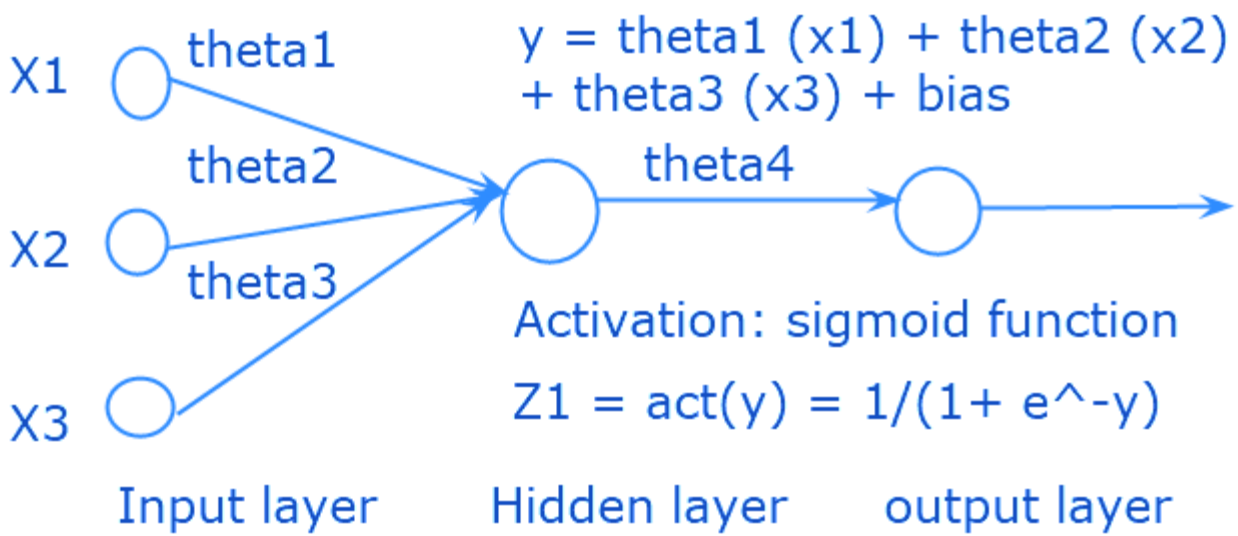

X1
X2
X3
theta1
theta2
theta3

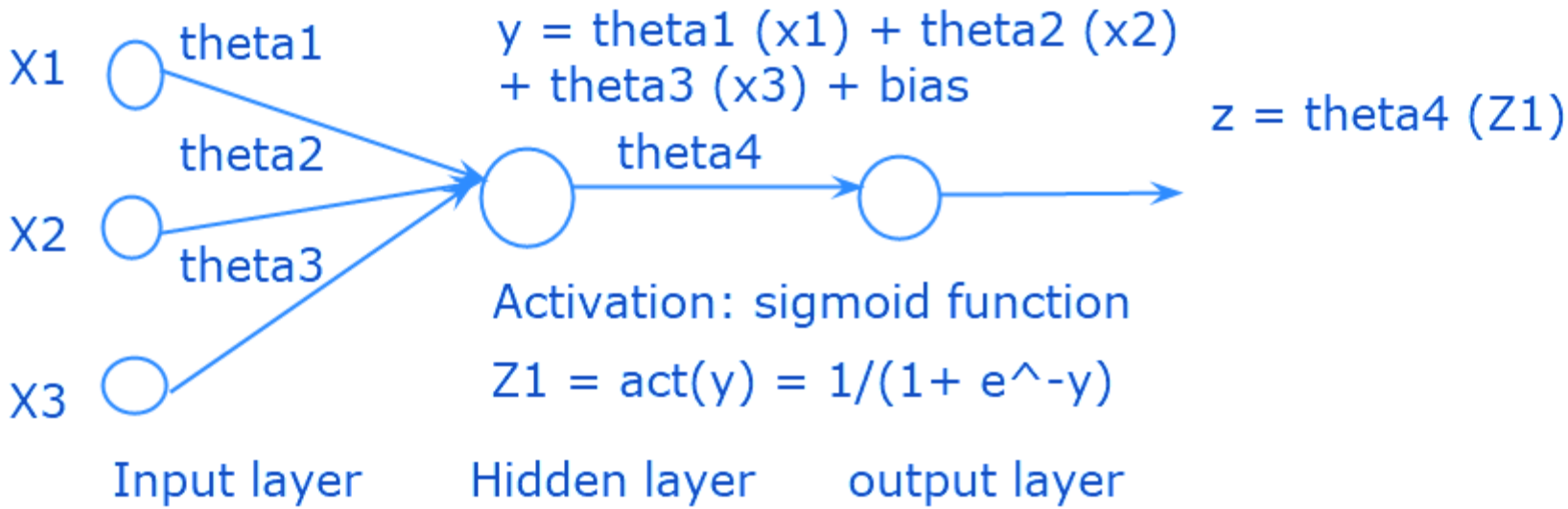

X1
X2
X3
theta1
theta2
theta3
y = theta1 (x1) + theta2 (x2)
+ theta3 (x3) + bias

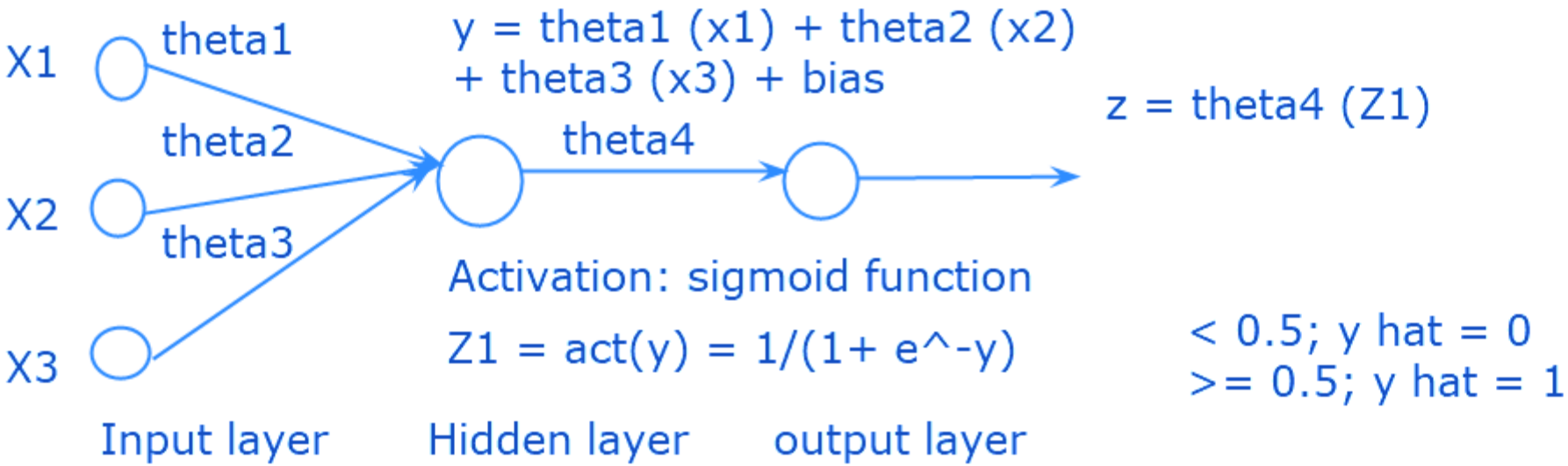

y = theta1 (x1) + theta2 (x2)
+ theta3 (x3) + bias
X1
X2
X3
theta1
theta2
theta3
Input layer
Hidden layer
output layer


X1
X2
X3
theta1
theta2
theta3
y = theta1 (x1) + theta2 (x2)
+ theta3 (x3) + bias
Activation: sigmoid function
Input layer
Hidden layer
output layer


X1
X2
X3
theta1
theta2
theta3
y = theta1 (x1) + theta2 (x2)
+ theta3 (x3) + bias
Activation: sigmoid function
Z1 = act(y) = 1/(1+ e^-y)
Input layer
Hidden layer
output layer


X1
X2
X3
theta1
theta2
theta3
theta4
y = theta1 (x1) + theta2 (x2)
+ theta3 (x3) + bias
Activation: sigmoid function
Z1 = act(y) = 1/(1+ e^-y)
Input layer
Hidden layer
output layer


X1
X2
X3
theta1
theta2
theta3
y = theta1 (x1) + theta2 (x2)
+ theta3 (x3) + bias
theta4
z = theta4 (Z1)
Activation: sigmoid function
Z1 = act(y) = 1/(1+ e^-y)
Input layer
Hidden layer
output layer


X1
X2
X3
theta1
theta2
theta3
y = theta1 (x1) + theta2 (x2)
+ theta3 (x3) + bias
theta4
z = theta4 (Z1)
Activation: sigmoid function
Z1 = act(y) = 1/(1+ e^-y)
< 0.5; y hat = 0
>= 0.5; y hat = 1
Input layer
Hidden layer
output layer

step 2: Act (

step 2: Act (summation for i = 1 to m (thetai) (xi)

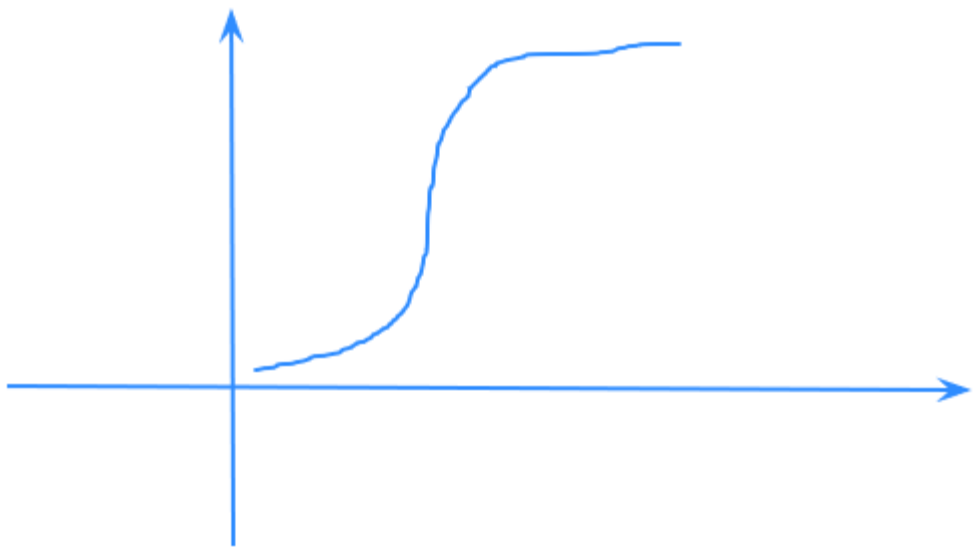
step 2: Act (summation for i = 1 to m (thetai) (xi))

step 2: Act (summation for i = 1 to m (thetai) (xi))

Sigmoid Function

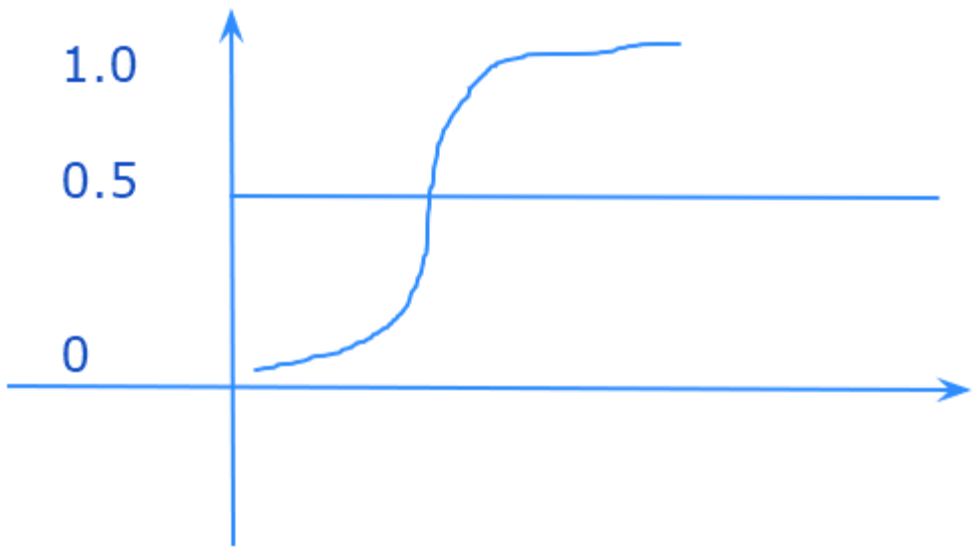
step 2: Act (summation for i = 1 to m (thetai) (xi))

Sigmoid Function



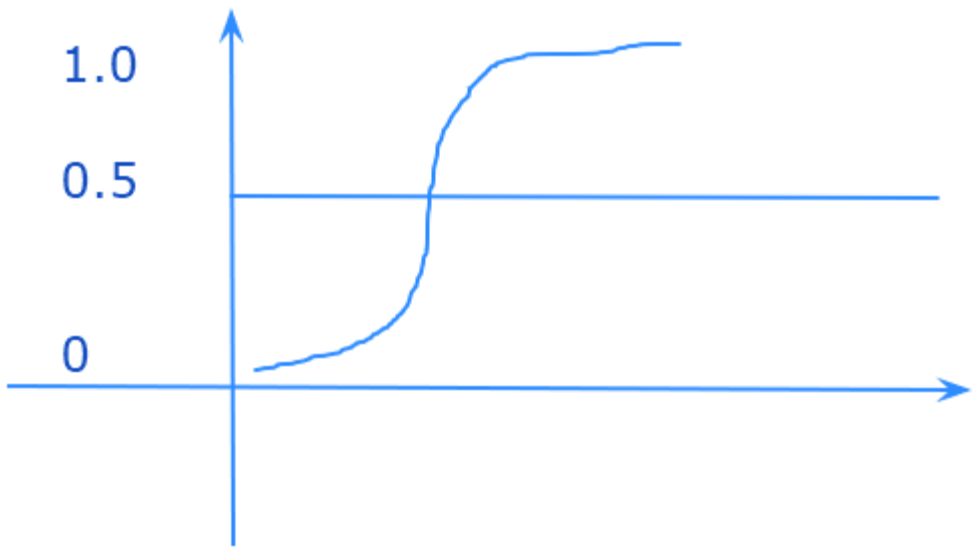
step 2: Act (summation for i = 1 to m (thetai) (xi))

Sigmoid Function



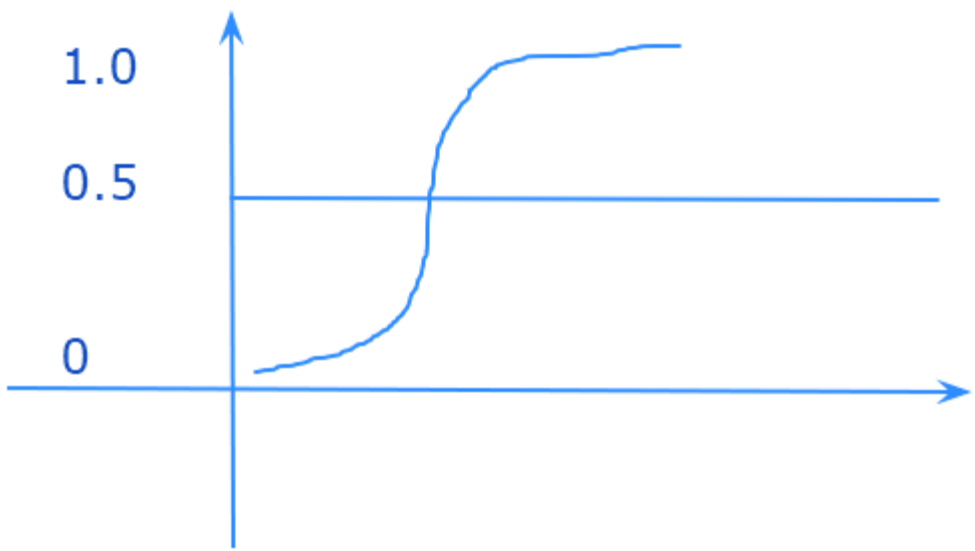
step 2: Act (summation for i = 1 to m (thetai) (xi))

Sigmoid Function

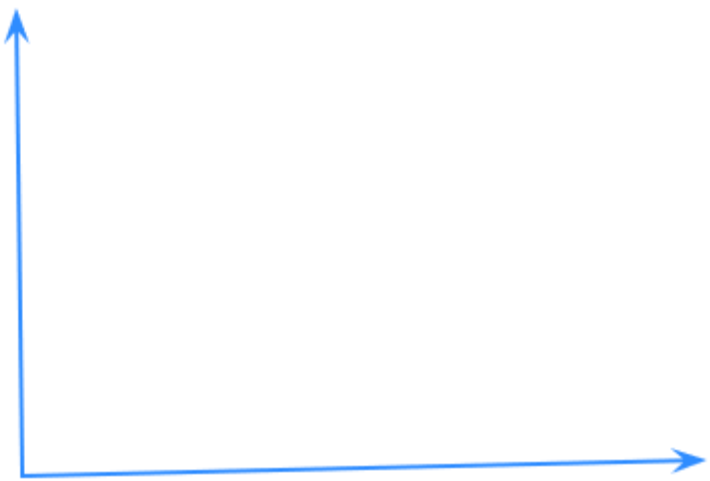


step 2: Act (summation for i = 1 to m (thetai) (xi))

Sigmoid Function

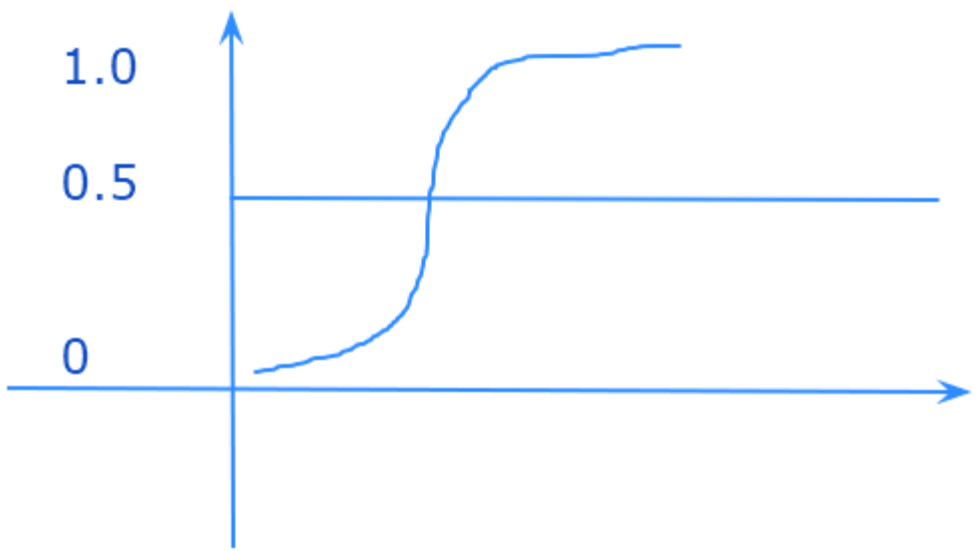


Relu AF

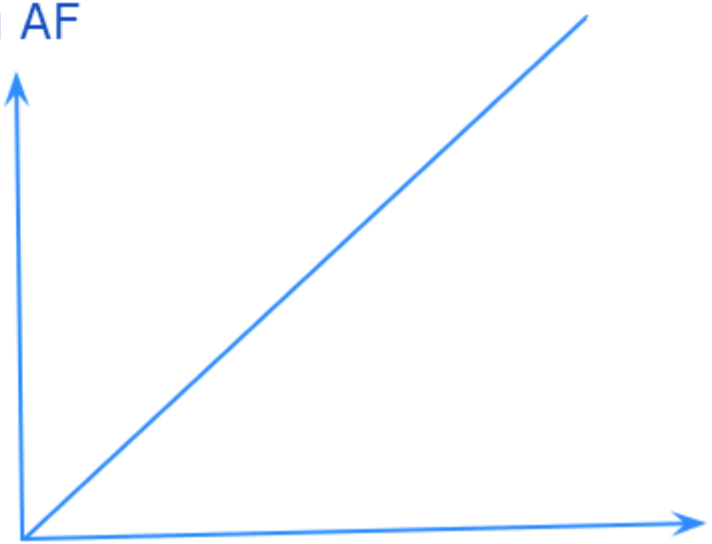


step 2: Act (summation for i = 1 to m (thetai) (xi))

Sigmoid Function

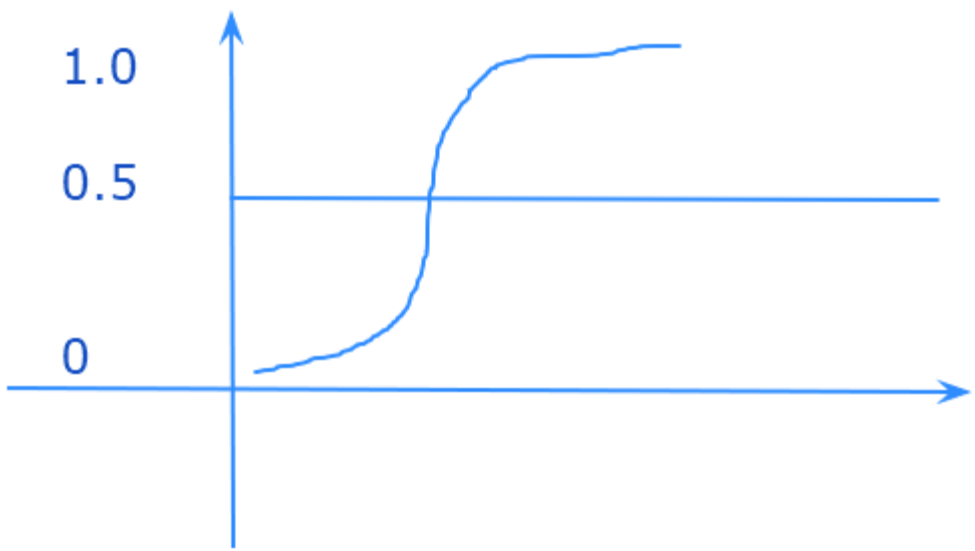


Relu AF

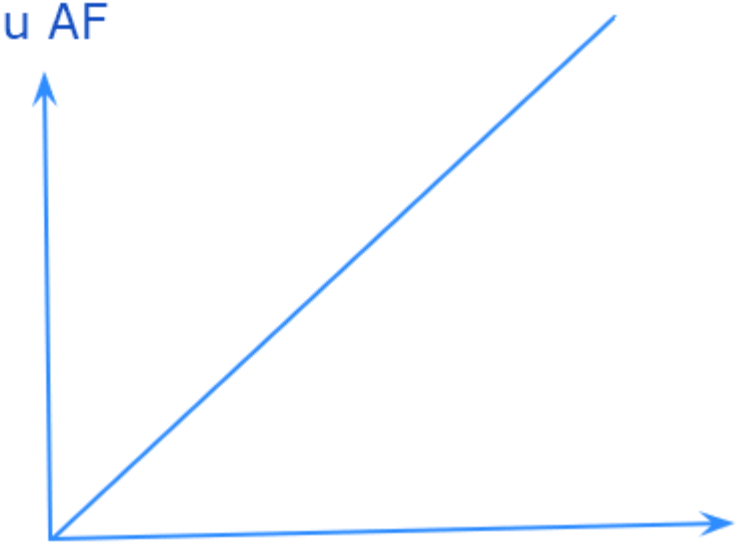


step 2: Act (summation for i = 1 to m (thetai) (xi))

Sigmoid Function



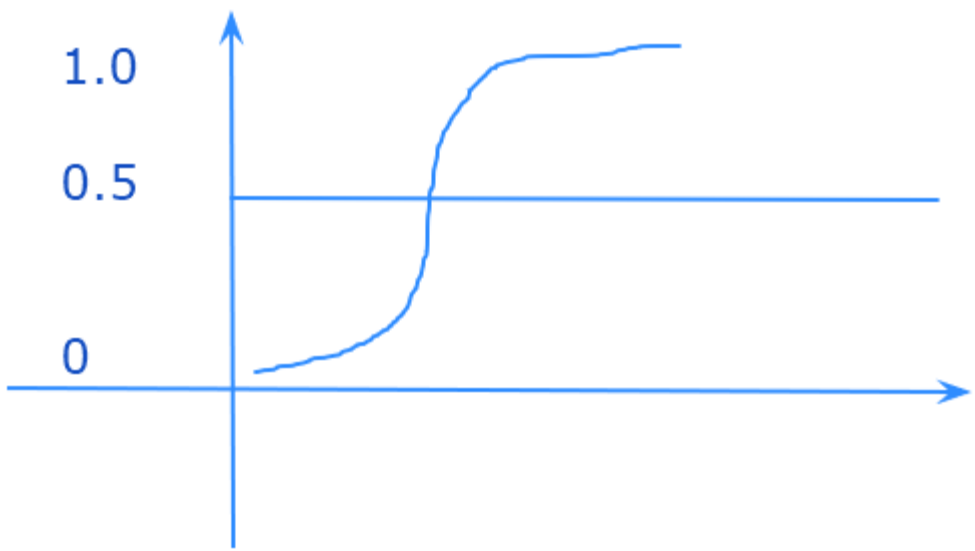
Relu AF



Y hat = max(y hat, 0)

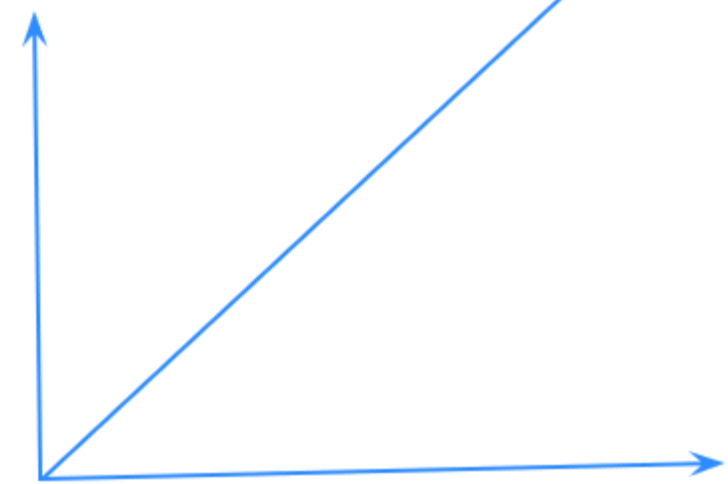
step 2: Act (summation for i = 1 to m (thetai) (xi))

Sigmoid Function



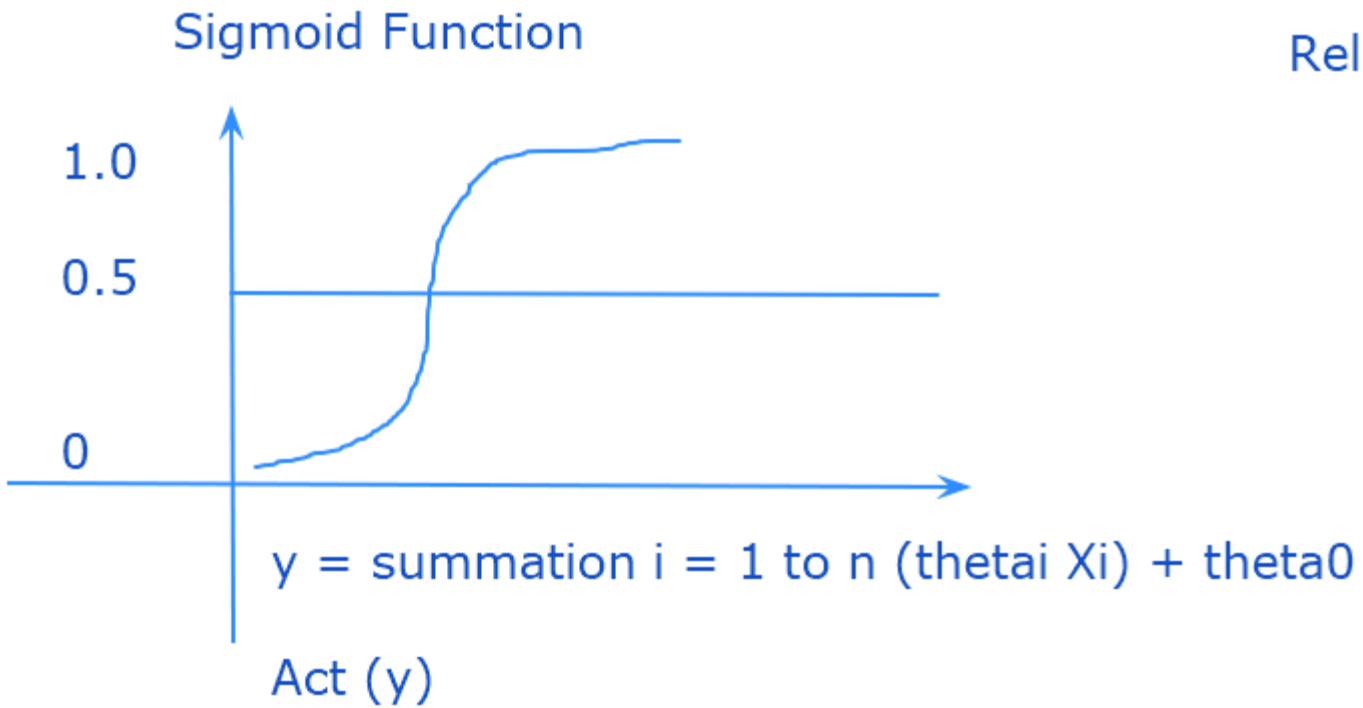
y hat = -ve value
output = 0
y hat = +ve
output = +ve value

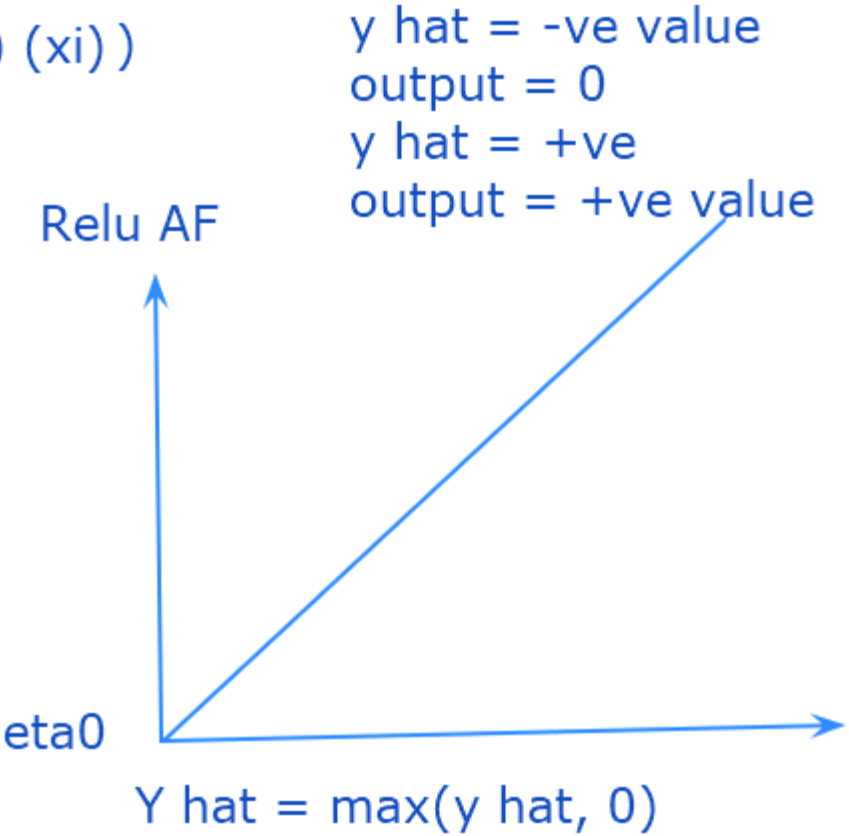
Relu AF



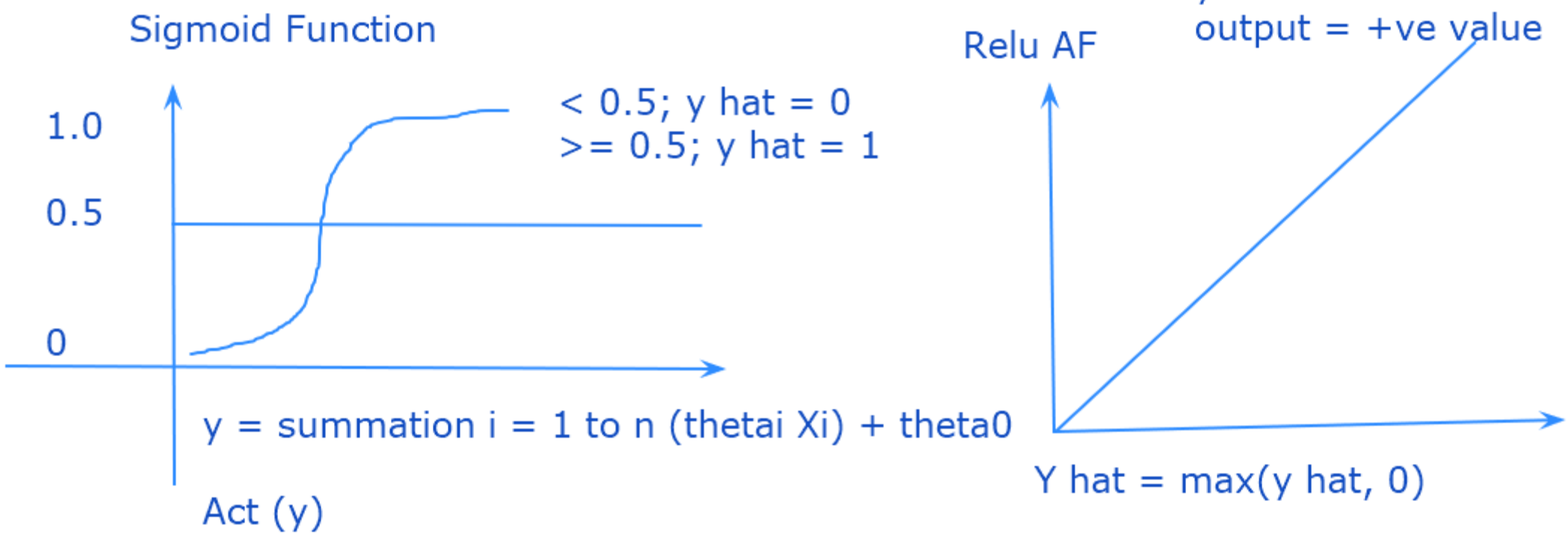
Y hat = max(y hat, 0)

step 2: Act (summation for i = 1 to m (thetai) (xi))

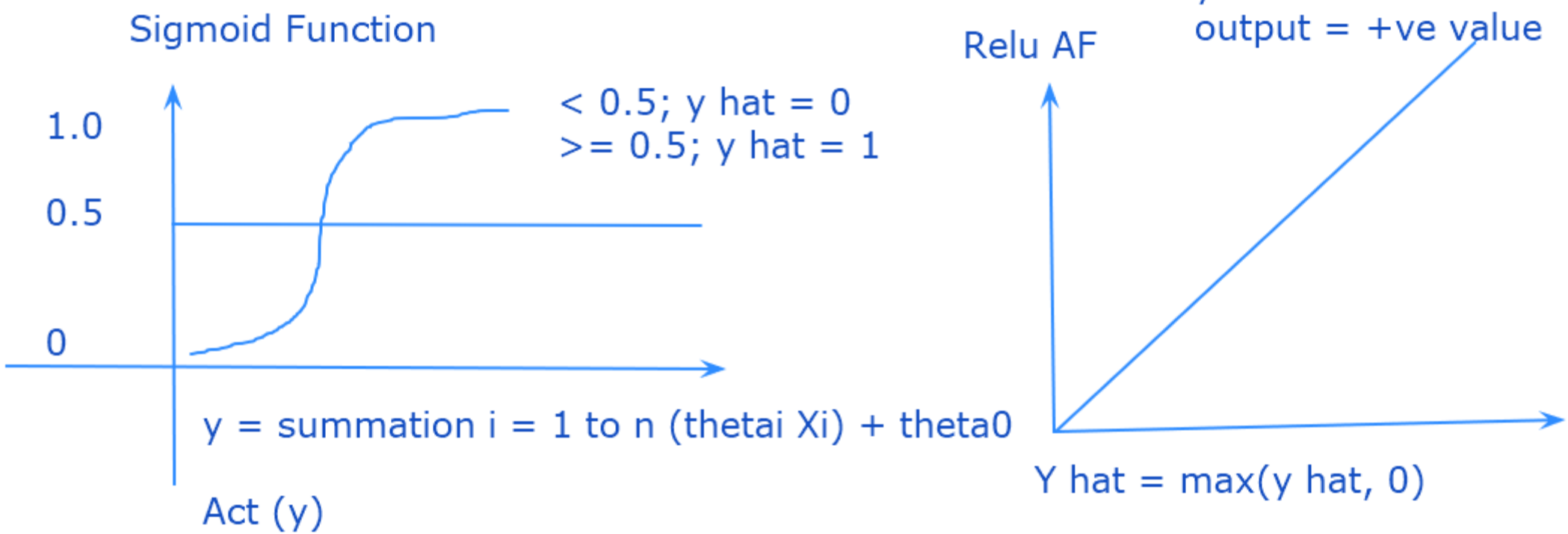




step 2: Act (summation for i = 1 to m (thetai) (xi))

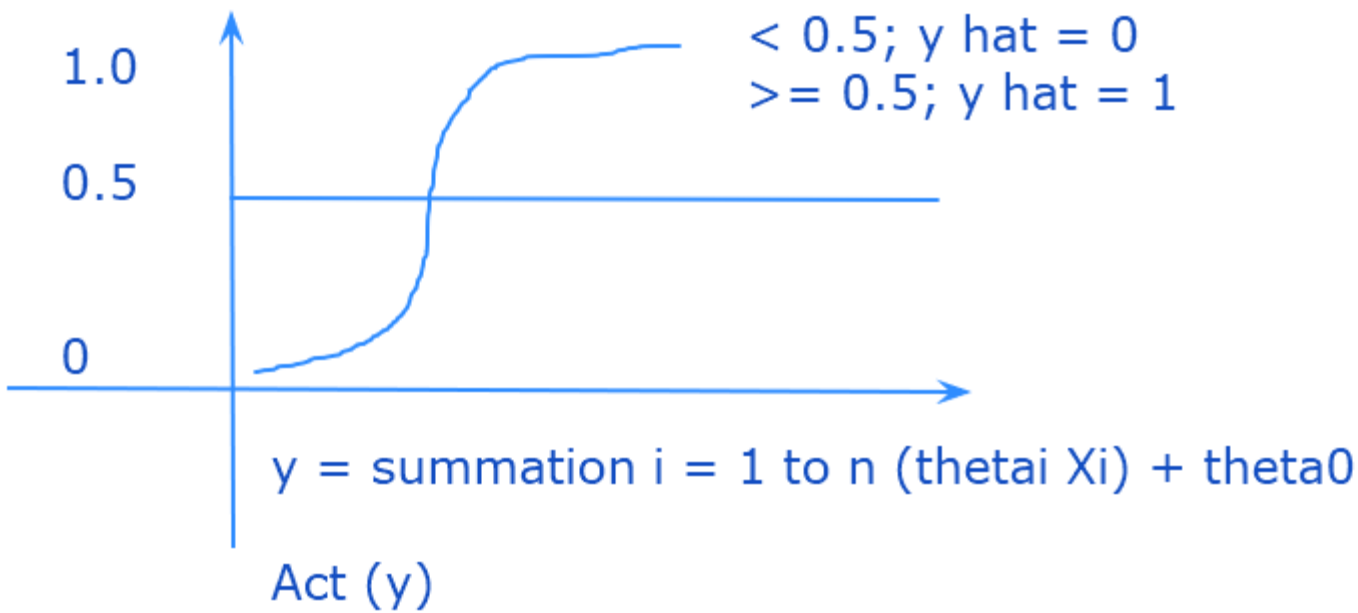


step 2: Act (summation for i = 1 to m (thetai) (xi))



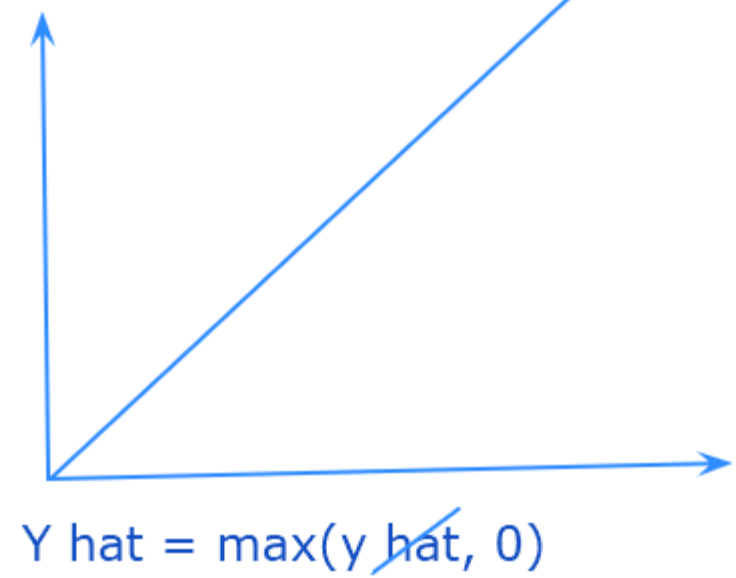
step 2: Act (summation for i = 1 to m (thetai) (xi))

Sigmoid Function

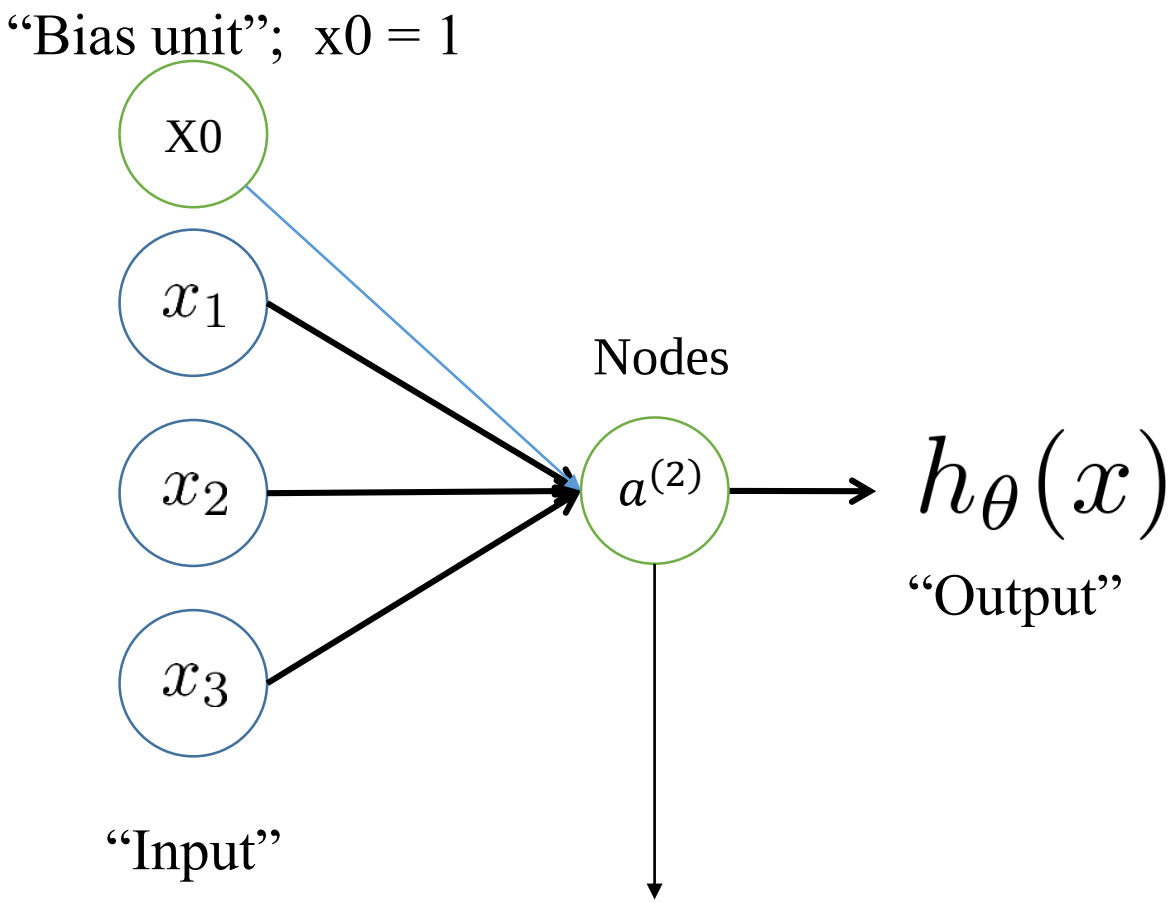


Relu AF

y hat = -ve value
output = 0
y hat = +ve
output = +ve value



Artificial Neuron model: Logistic unit



$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

$$h_\theta(x) = g(z)$$

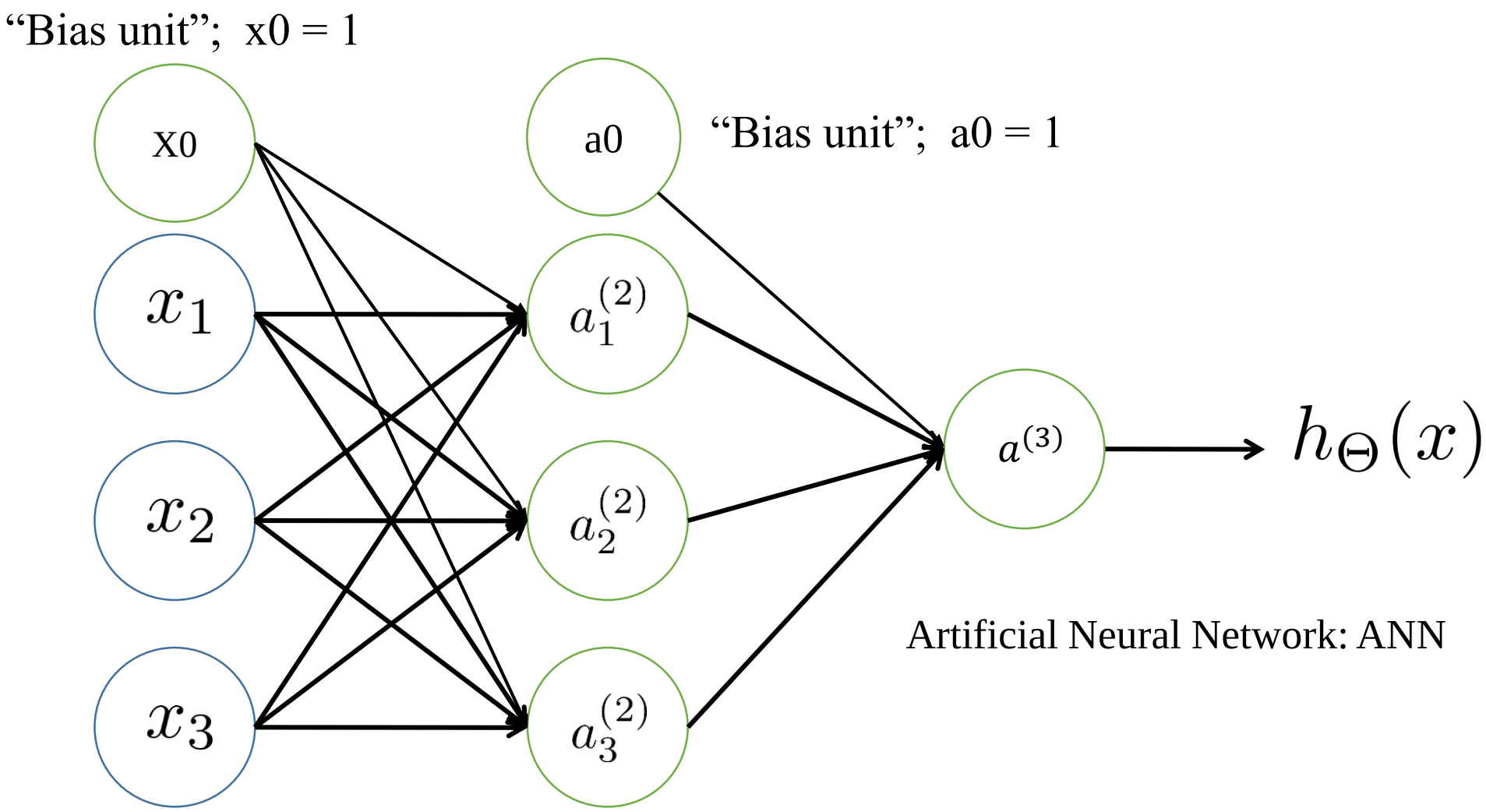
$$z = \theta^T x$$

- Sigmoid (logistic) activation function; neuron i.e. Analogous to neurons body; called as **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.

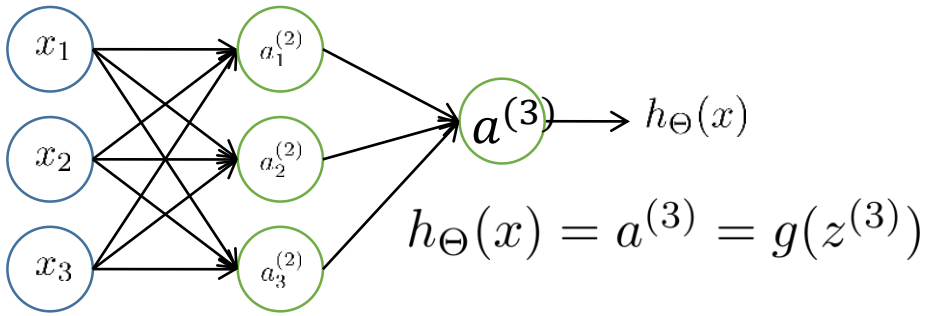
Perceptron

- **Perceptron** was first introduced by **Frank Rosenblatt in 1957** and he also discussed a Perceptron learning rule based on the original MCP neuron.
- **Perceptron Learning Rule:** states that the **algorithm would automatically learn** the optimal parameters or weight coefficients. The input features are then **multiplied with these parameters or weights to determine if a neuron fires or not**.
- There are **two types of Perceptron's**: Single layer and Multilayer.
- **Single layer Perceptron's** can learn only linearly separable patterns.
- **Multilayer Perceptron's** or **feedforward neural networks** with **two or more layers** have the greater processing power.
- If the sum of the input signals exceeds a certain threshold, it outputs a signal **(i.e. neuron is fired)**; otherwise, there is no output.

Neural Network System



Neural Network: Forward Propagation



$a_i^{(j)}$ = "activation" of unit i in layer j

$\Theta^{(j)}$ = matrix of weights controlling function mapping from layer j to layer $j+1$

$$a^{(2)} = g(z^{(2)}) \quad z^{(2)} = \Theta^{(1)}x$$

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$

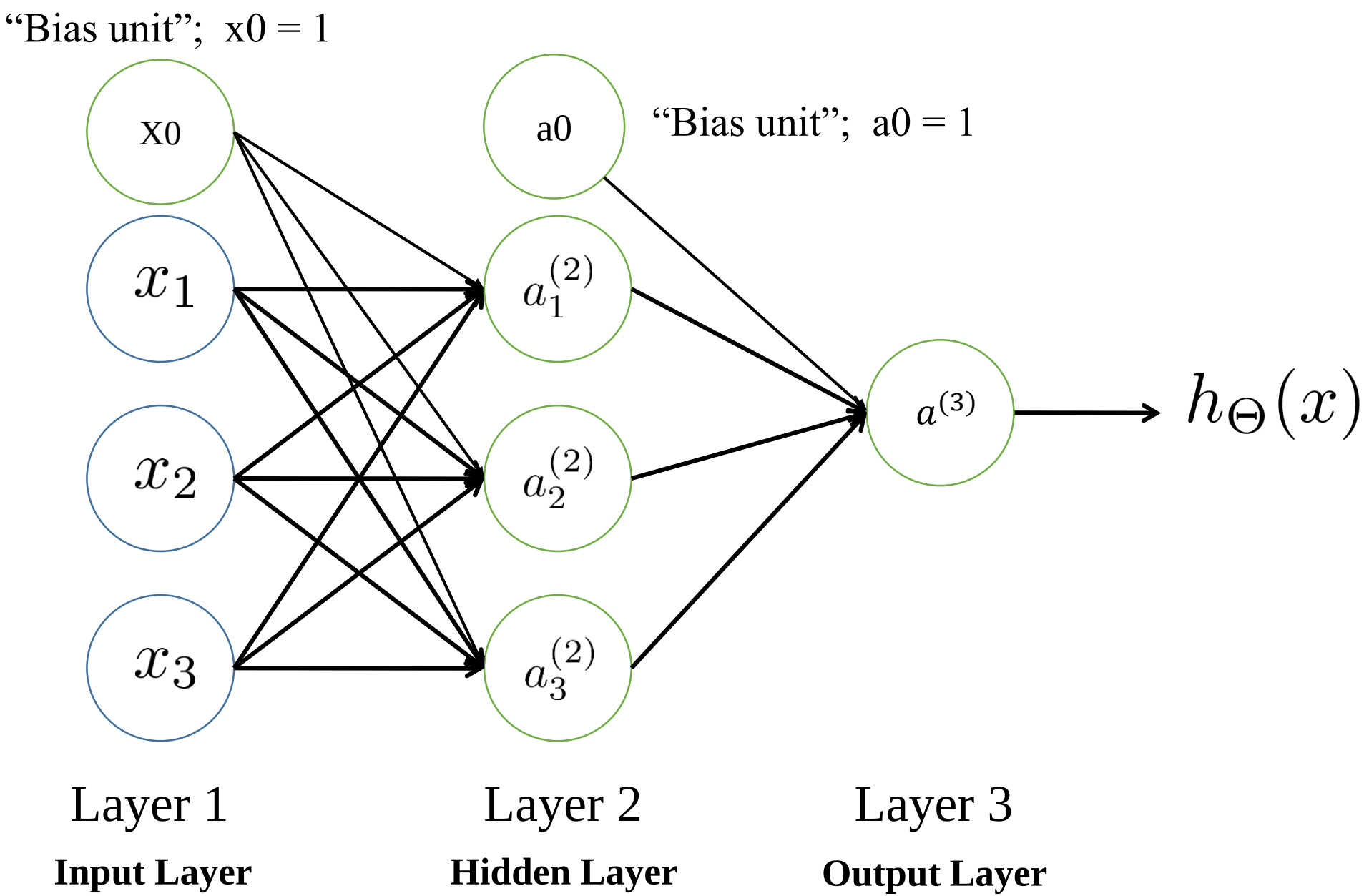
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

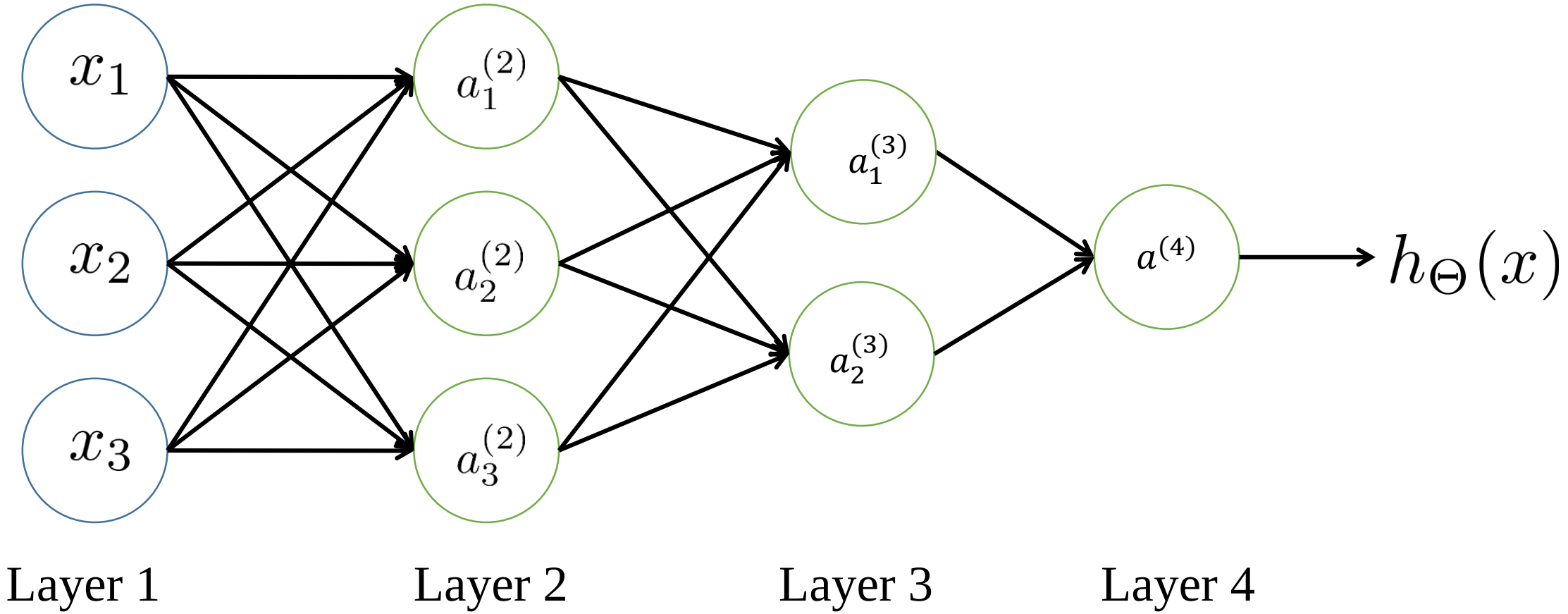
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

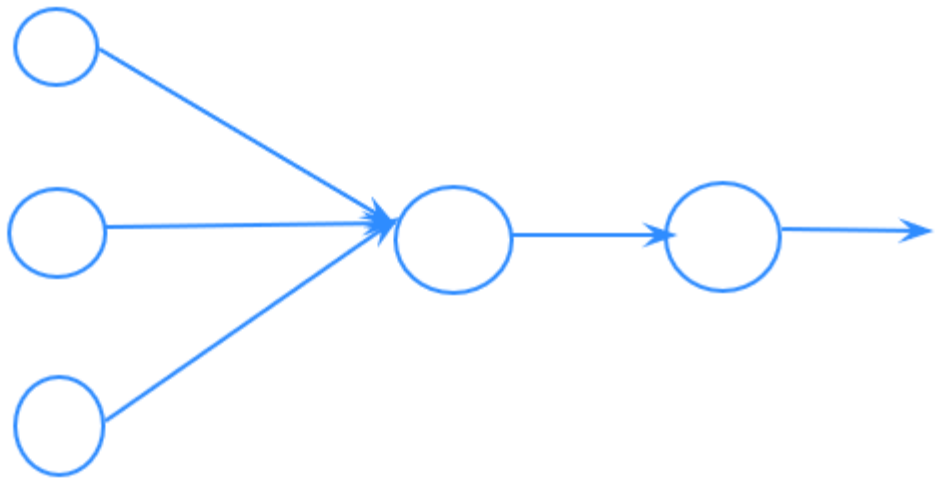
If network has s_j units in layer j, s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$.

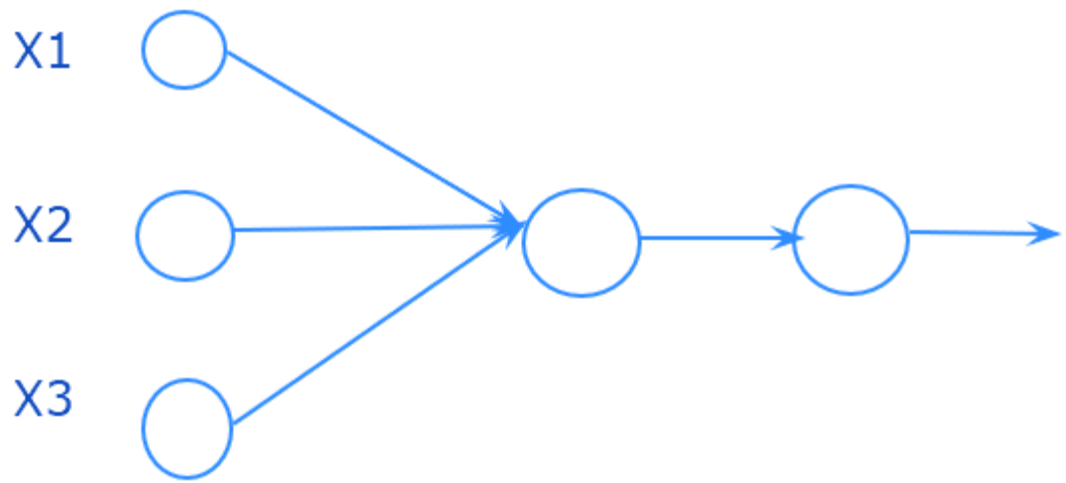
Neural Network learning its own features

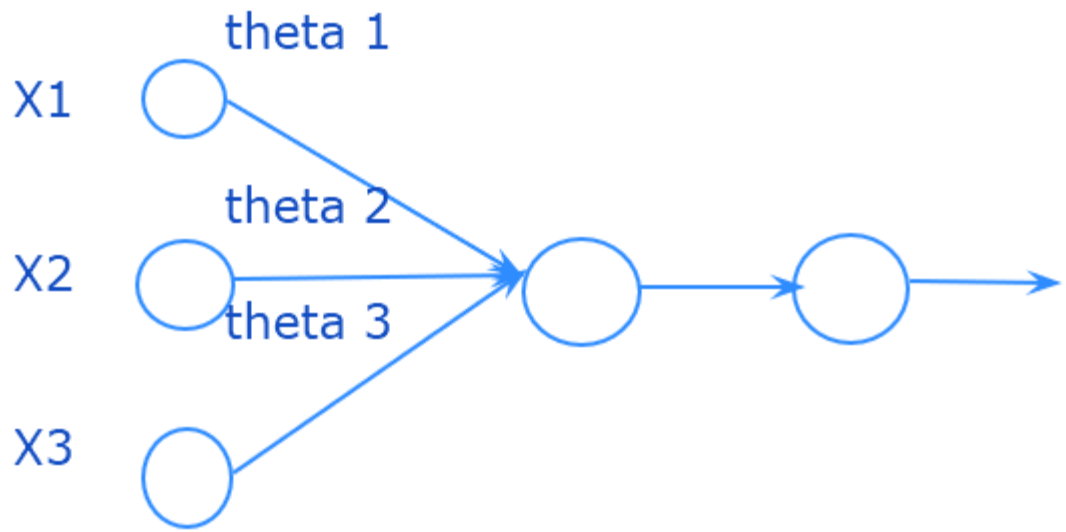


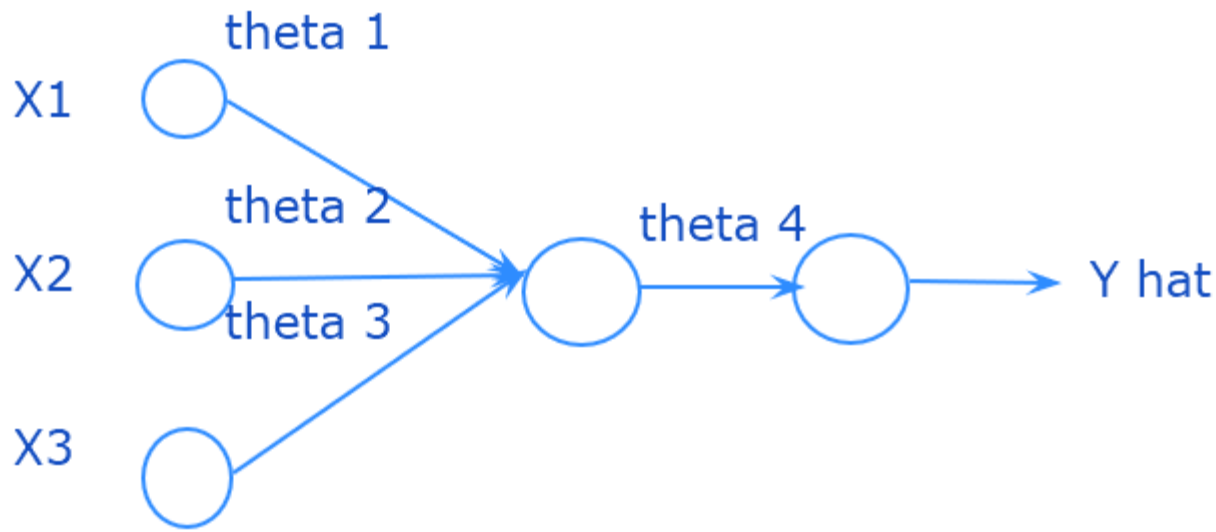
Other network architectures

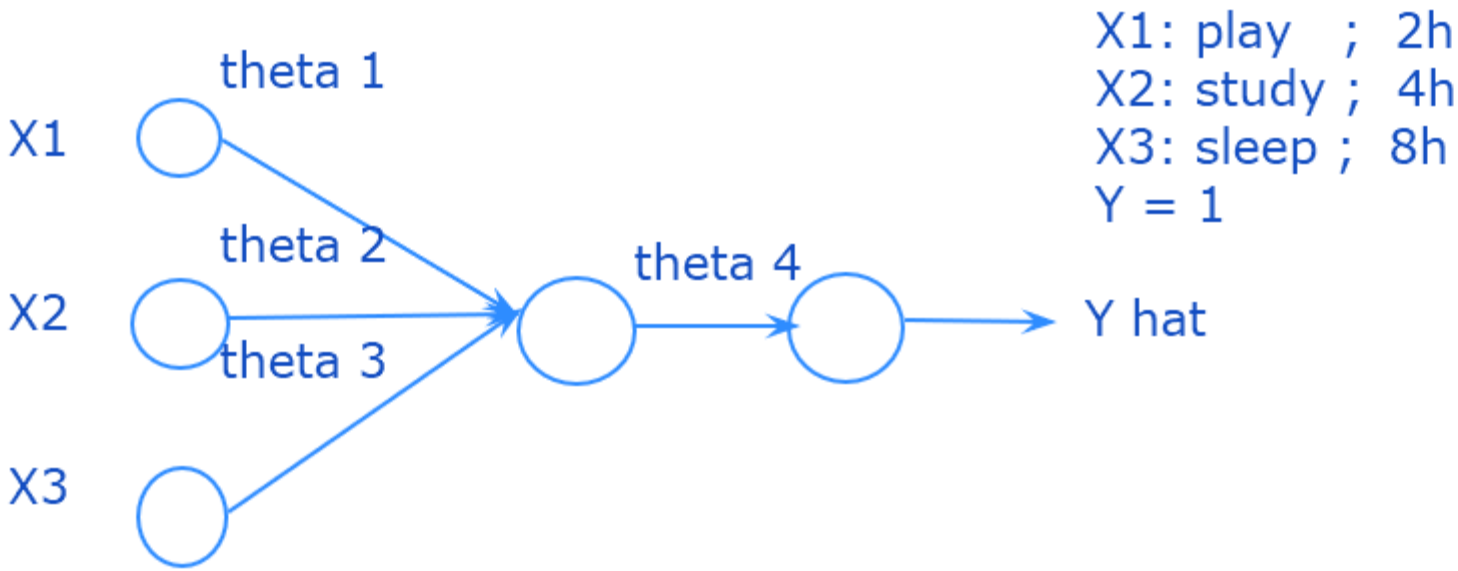




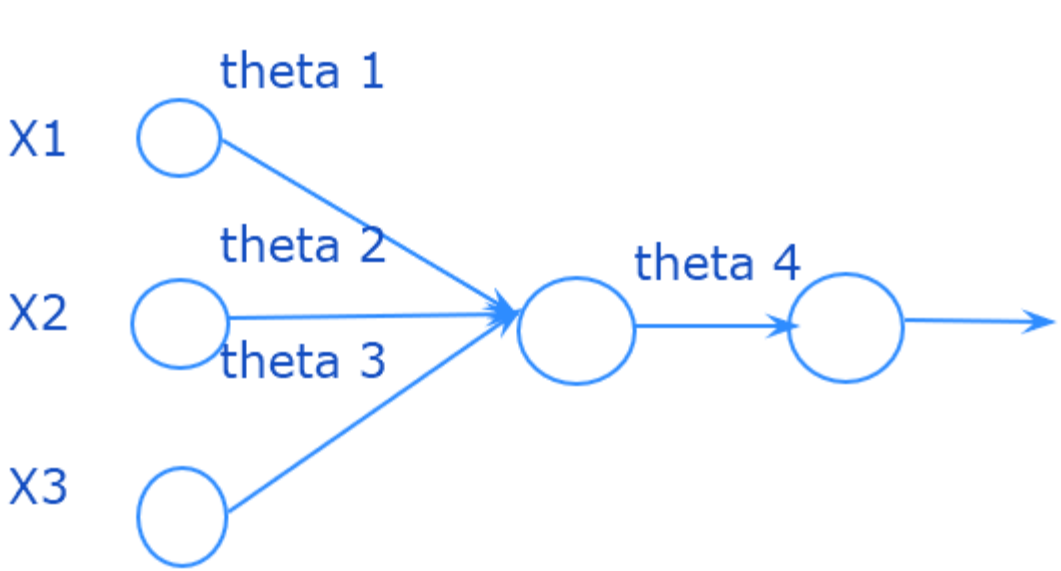

X1
X2
X3


theta 1
X1
theta 2
X2
theta 3
X3


theta 1
X1
theta 2
X2
theta 4
Y hat
theta 3
X3

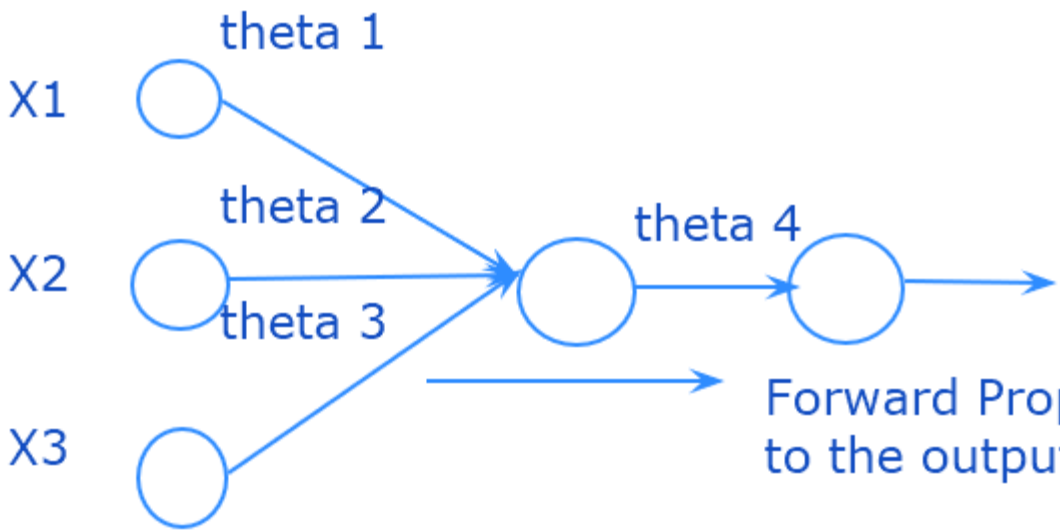


X1: play ; 2h
X2: study ; 4h
X3: sleep ; 8h
Y = 1



X1: play ; 2h
X2: study ; 4h
X3: sleep ; 8h
Y = 1

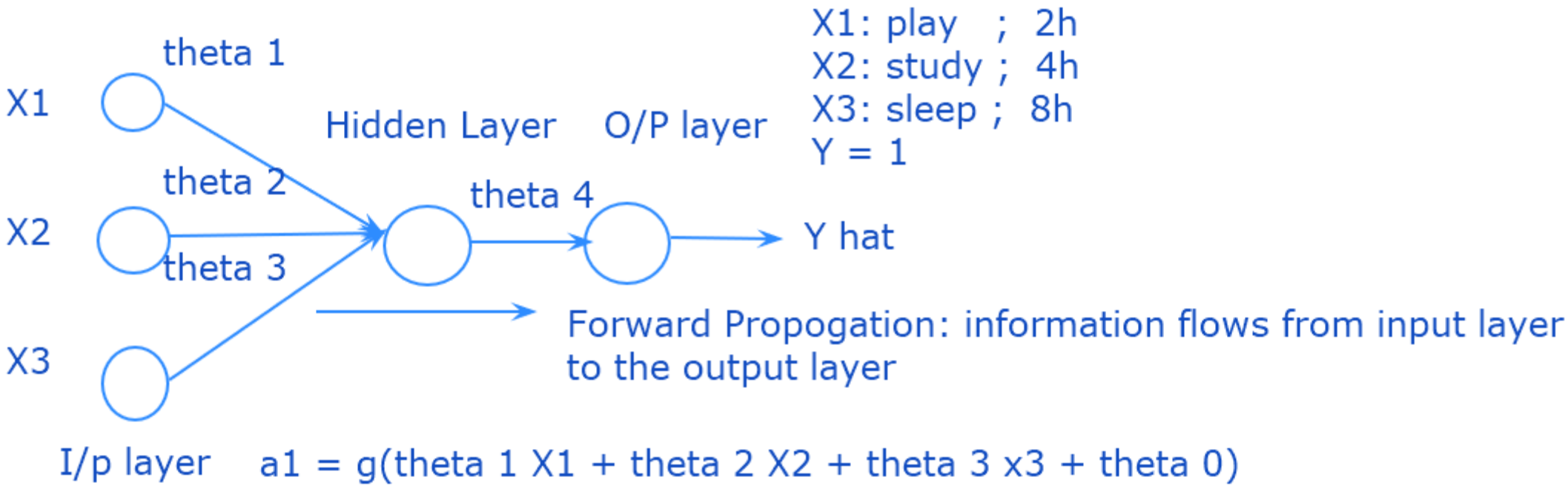
$$a1 = g(\text{theta 1 } X1 + \text{theta 2 } X2 + \text{theta 3 } x3 + \text{theta 0})$$

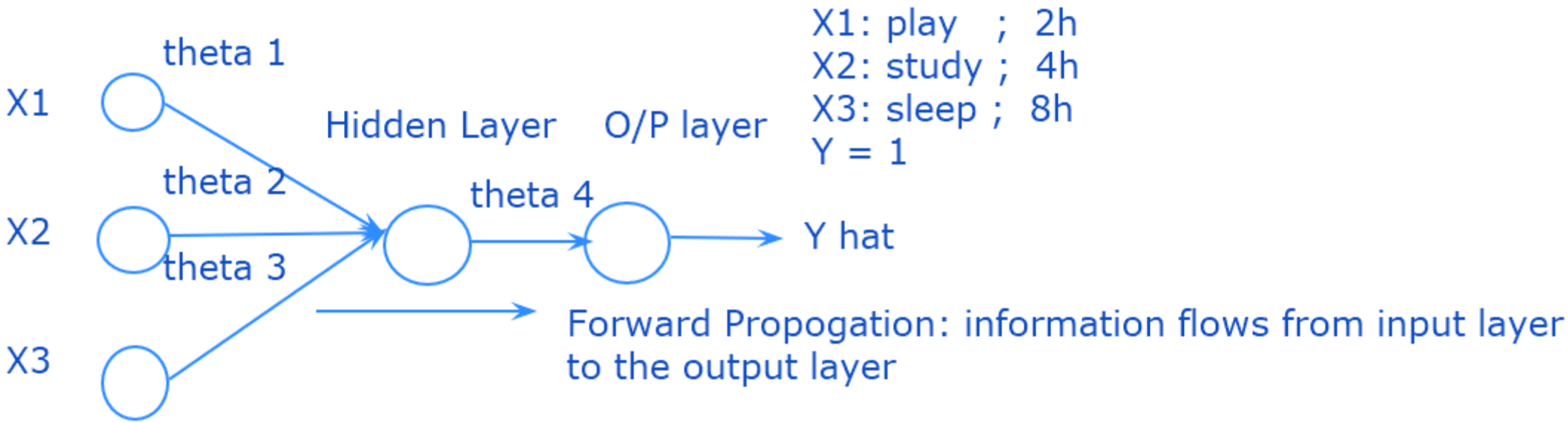


X1: play ; 2h
X2: study ; 4h
X3: sleep ; 8h
Y = 1

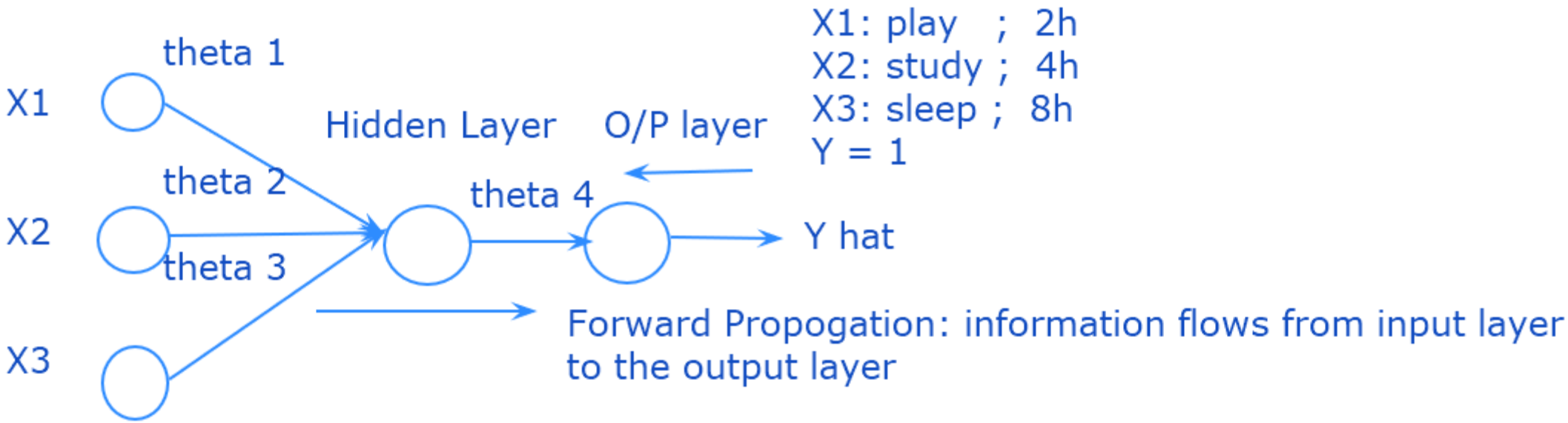
Forward Propogation: information flows from input layer to the output layer

$$a1 = g(\text{theta } 1\ X1 + \text{theta } 2\ X2 + \text{theta } 3\ x3 + \text{theta } 0)$$

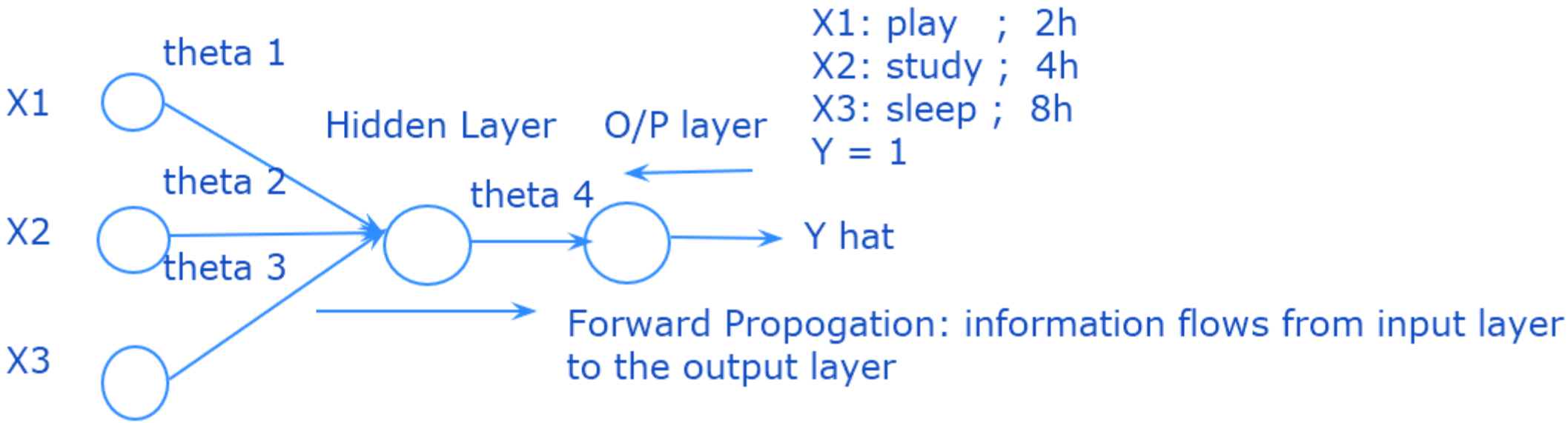

X1: play ; 2h
X2: study ; 4h
X3: sleep ; 8h
Y = 1
theta 1
X1
Hidden Layer
O/P layer
theta 2
theta 4
X2
Y hat
theta 3
Forward Propogation: information flows from input layer to the output layer
X3
I/p layer
a1 = g(theta 1 X1 + theta 2 X2 + theta 3 x3 + theta 0)



Loss function= (y- yhat)^2

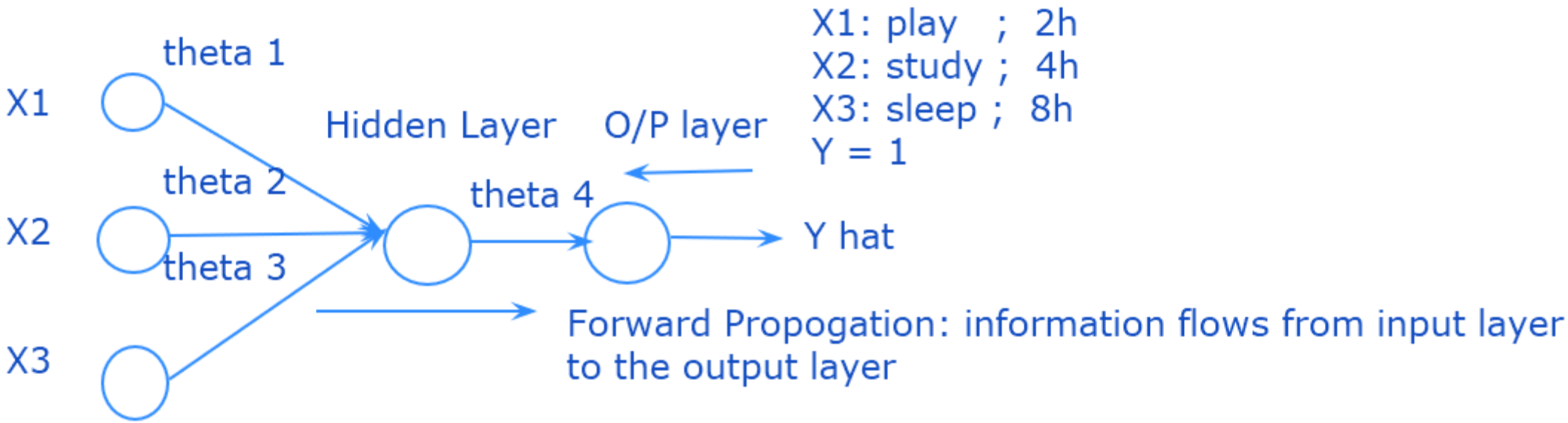


Loss function= (y- yhat)^2



Loss function= (y- yhat)^2

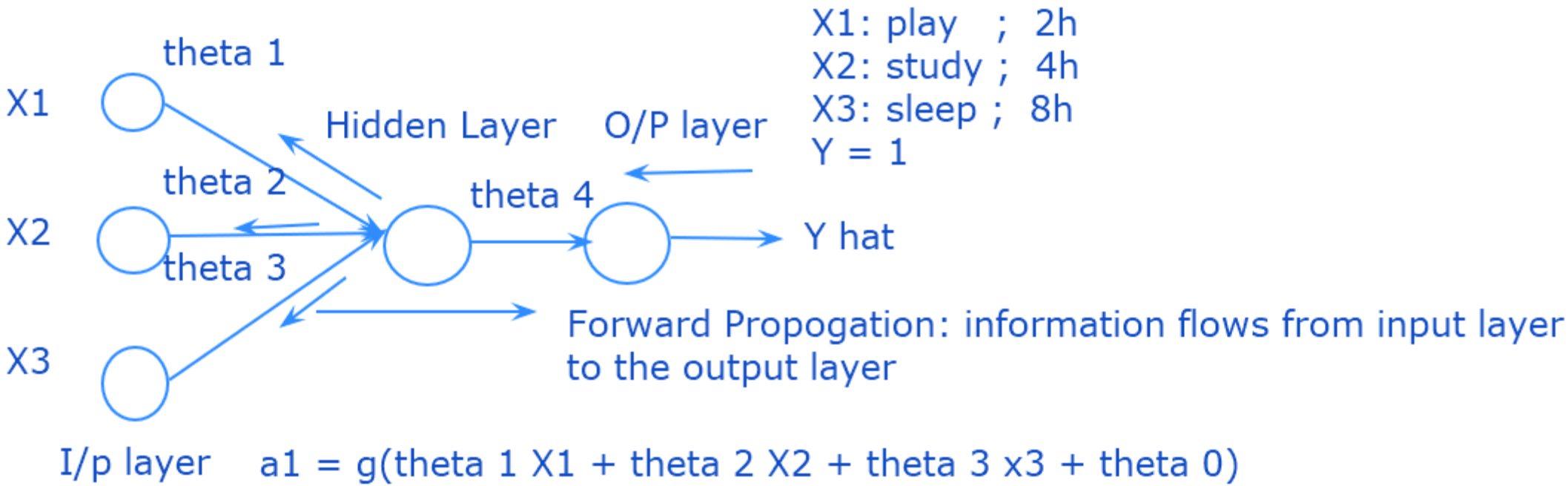
Backward propogation: used to find appropriate values thetas



Loss function= (y- yhat)^2

Backward propogation: used to find appropriate values thetas

theta 4 (new):= theta 4 (old) - alpha partial derivative of cost function w.r.t parameter value itself



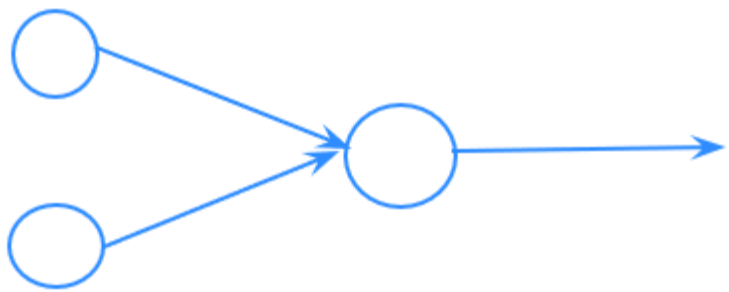
Loss function= (y- yhat)^2

Backward propogation: used to find appropriate values thetas

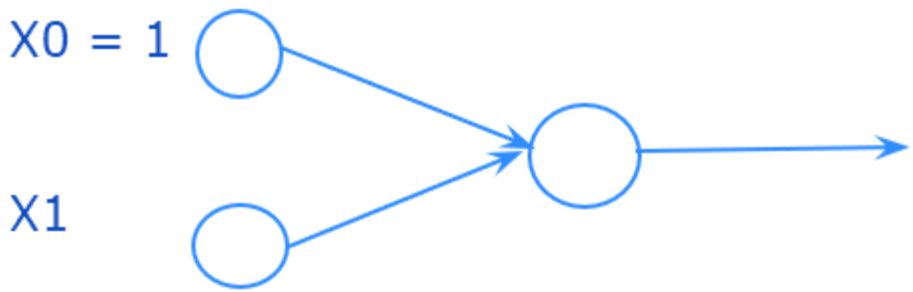
theta 4 (new):= theta 4 (old) - alpha partial derivative of cost function w.r.t parameter value itself

Example:

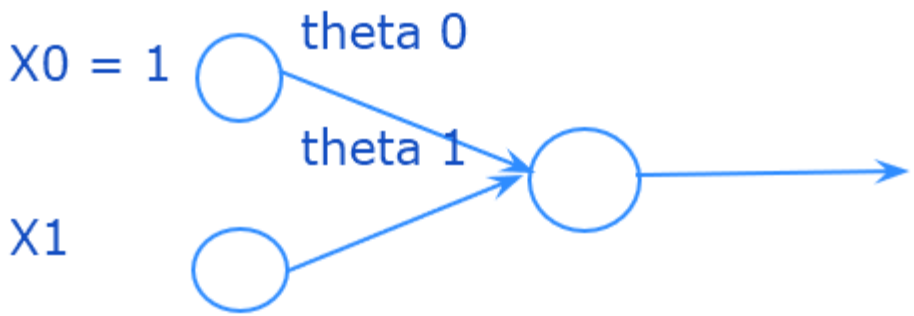
Example:



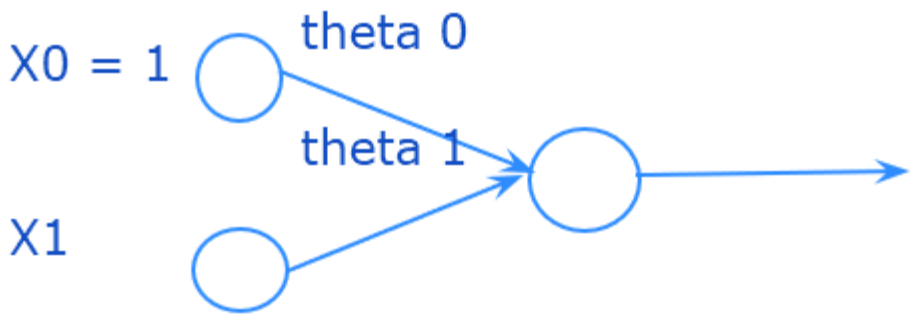
Example:



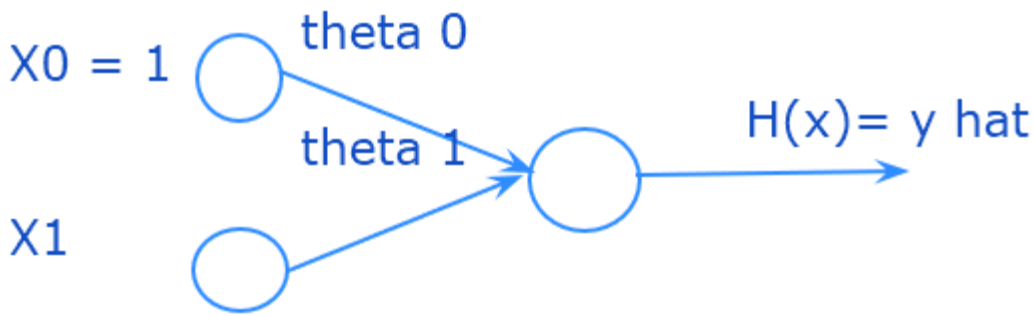
Example:



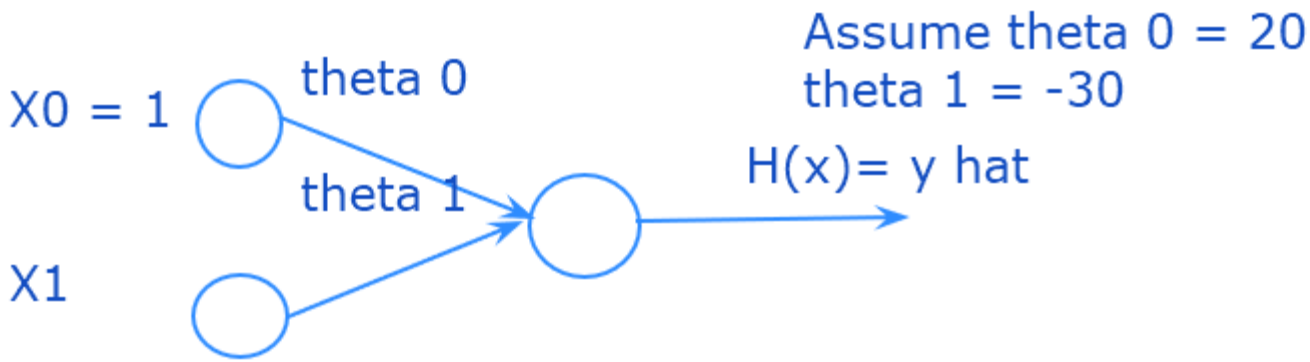
Example:



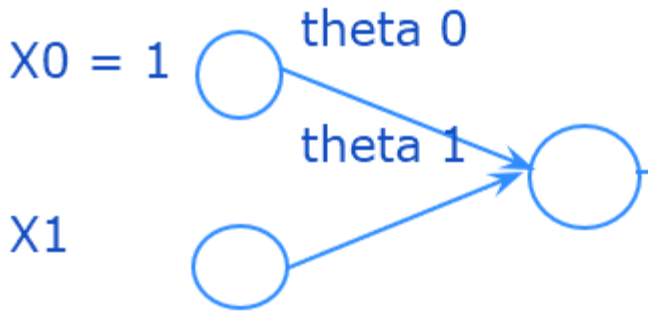
Example:



Example:



Example:

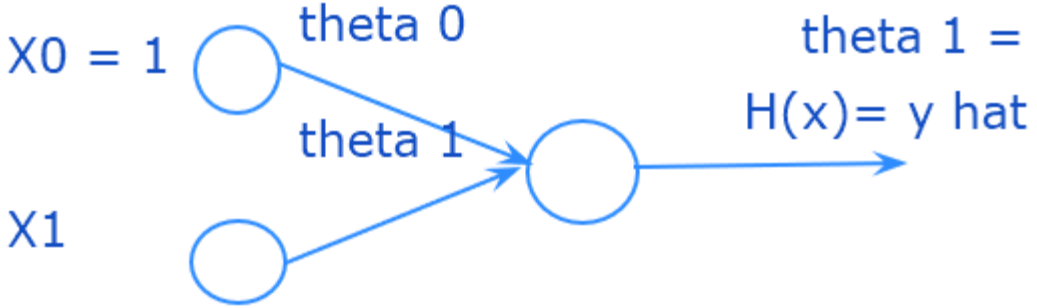


Assume theta 0 = 20
theta 1 = -30

H(x)= y hat

X1 h(X)

Example:

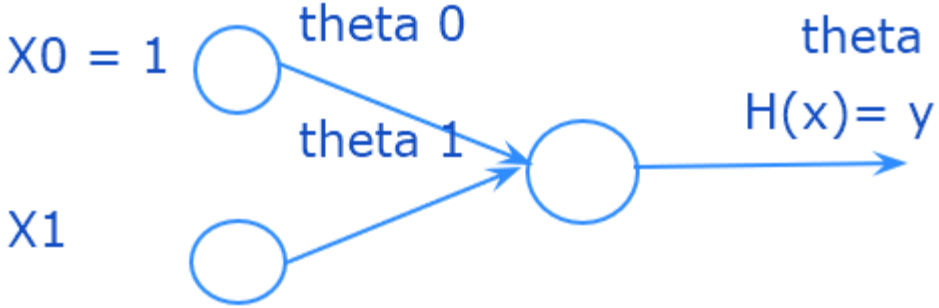


Assume theta 0 = 20
theta 1 = -30

X1 h(X)

h(x) = g(20 -30X1) = y hat
if x1 = 0; h(x) = g(20) = y hat = 1
if x1 = 1; h(x) = g(20-30) = g(-10) = y hat = 0

Example:



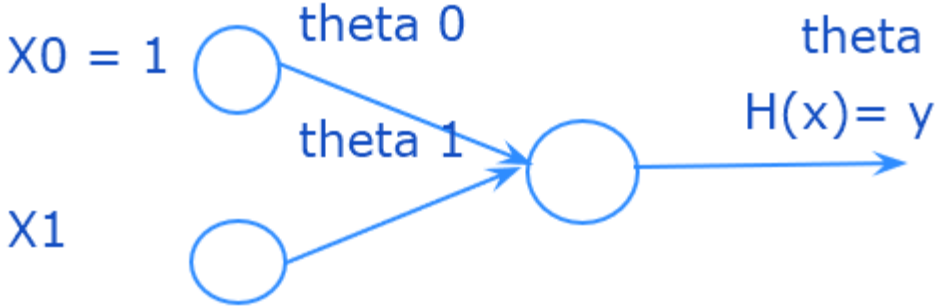
Assume theta 0 = 20
theta 1 = -30

H(x)= y hat

X1	h(X)
0	1
1	0

h(x) = g(20 -30X1) = y hat
if x1 = 0; h(x) = g(20) = y hat = 1
if x1 = 1; h(x) = g(20-30) = g(-10) = y hat = 0

Example: Logic Gate (NOT)



Assume theta 0 = 20
theta 1 = -30

H(x)= y hat

X1	h(X)
0	1
1	0

h(x) = g(20 -30X1) = y hat
if x1 = 0; h(x) = g(20) = y hat = 1
if x1 = 1; h(x) = g(20-30) = g(-10) = y hat = 0