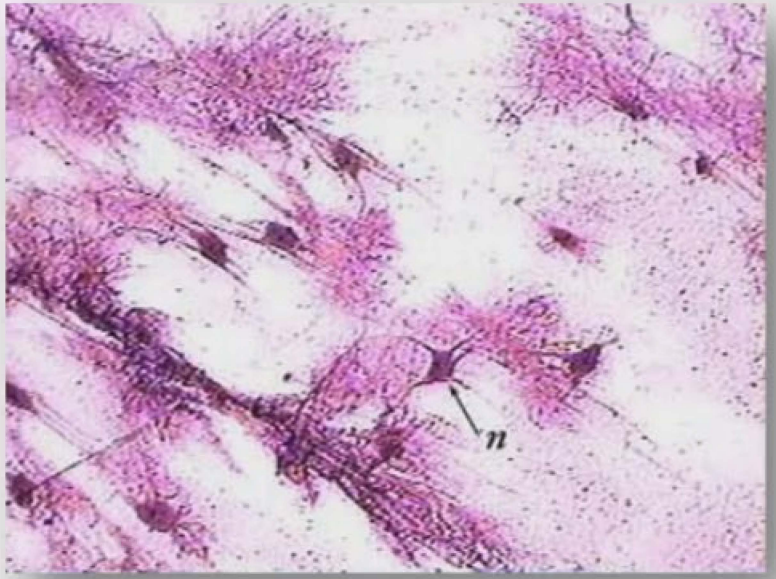
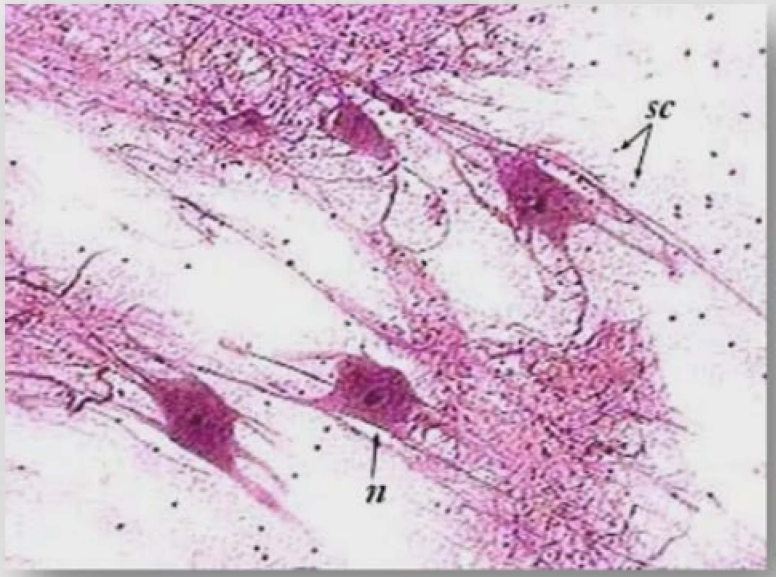
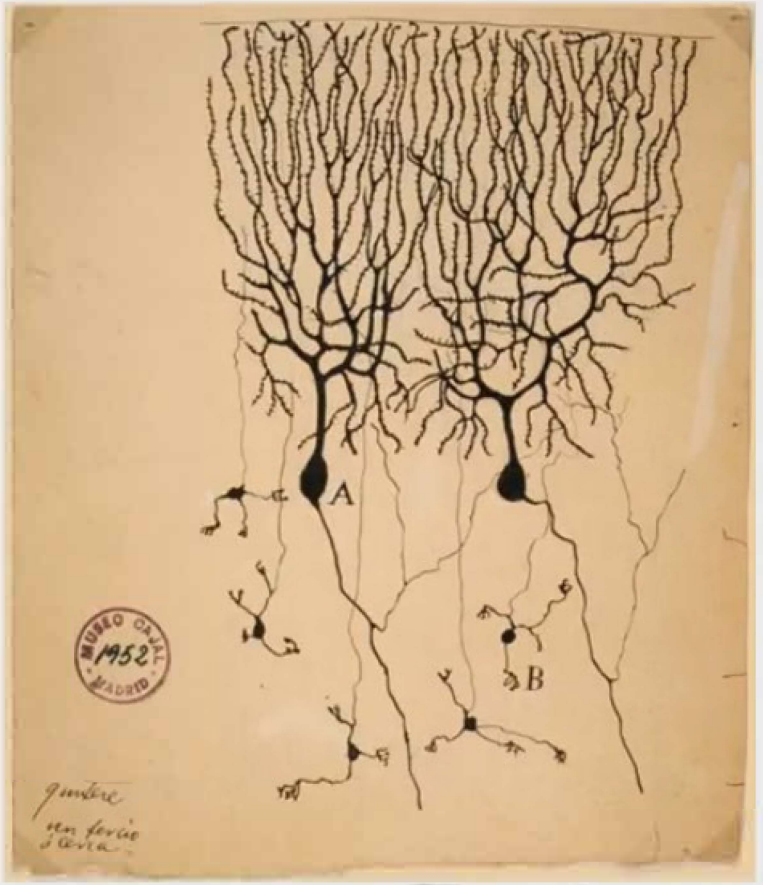
What we will learn in this section:

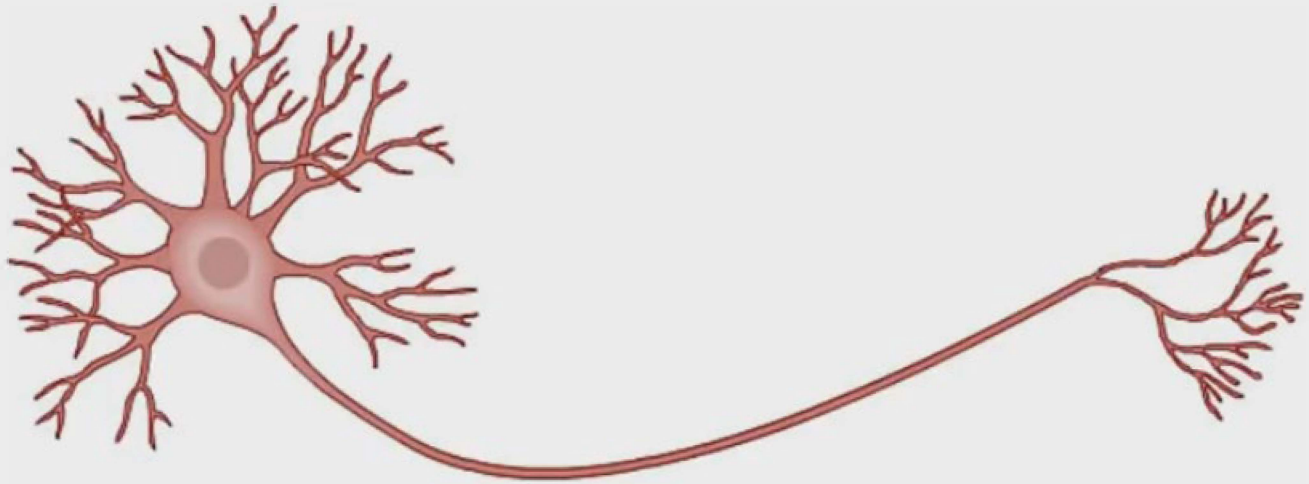
- The Neuron
- The Activation Function
- How do Neural Networks work? (example)
- How do Neural Networks learn?
- Gradient Descent
- Stochastic Gradient Descent
- Backpropagation

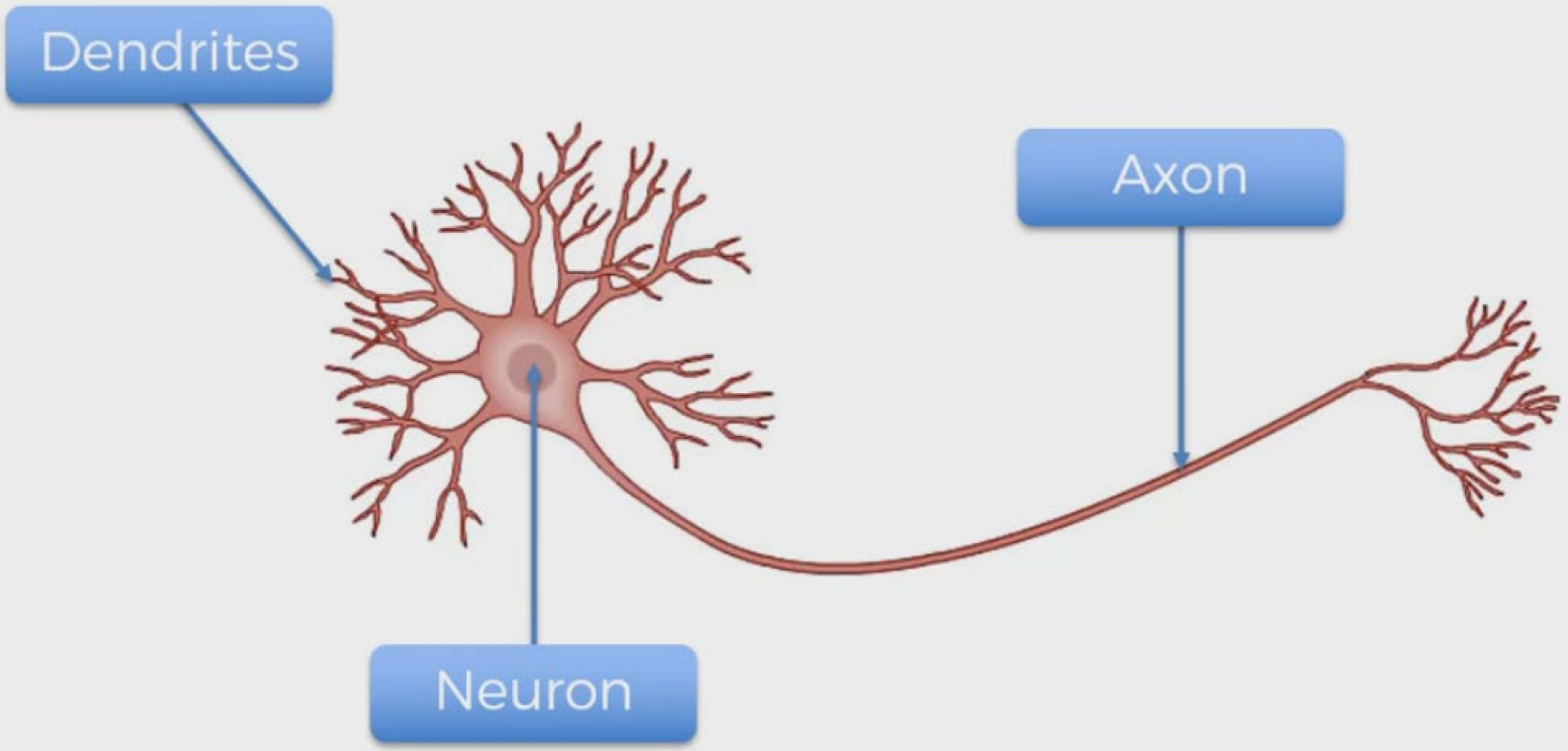
The Neuron

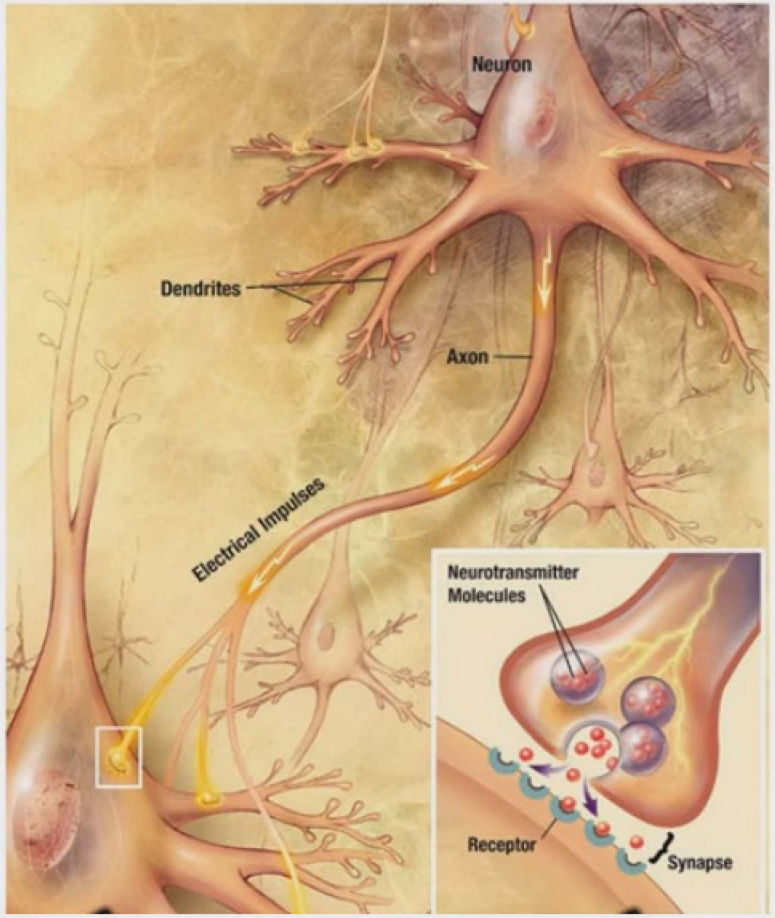

n

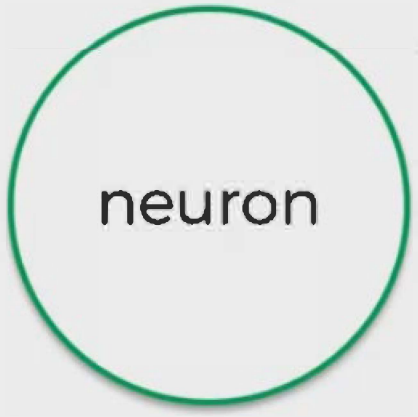

sc
n

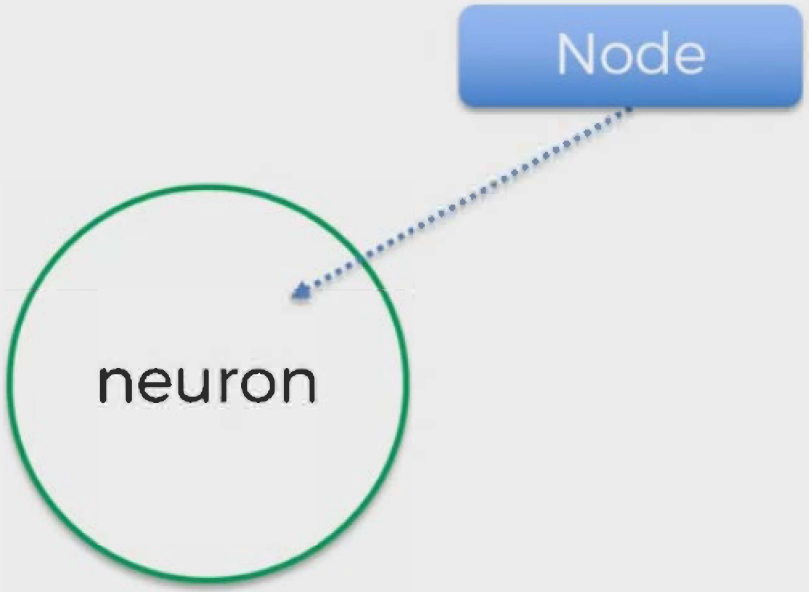

A
B
MUSEO CAJAL
1952
MADRID

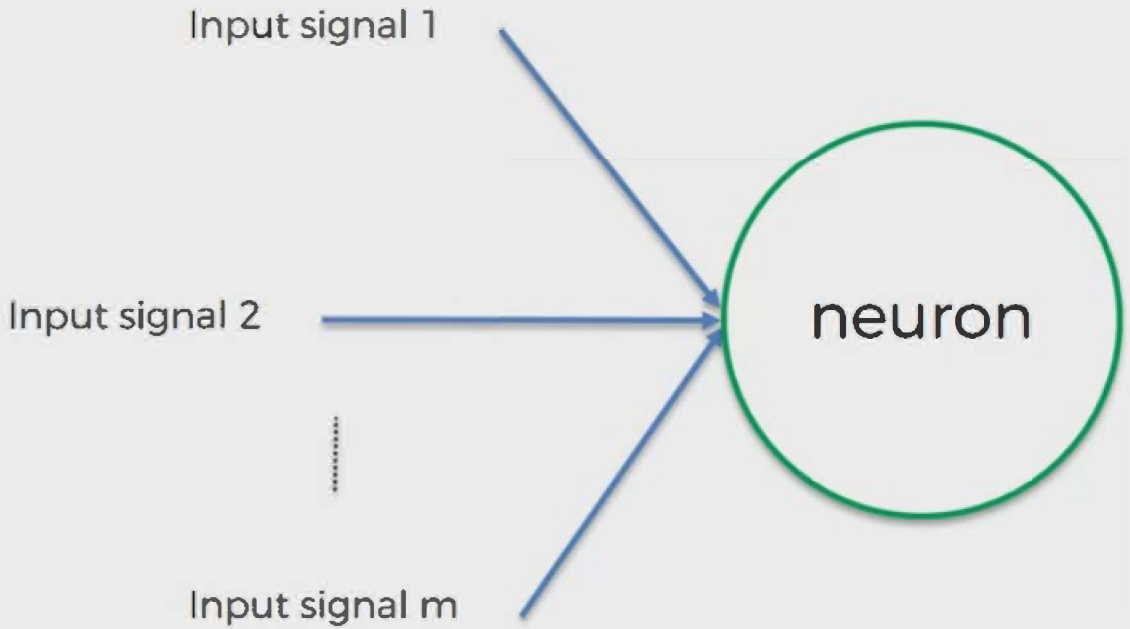


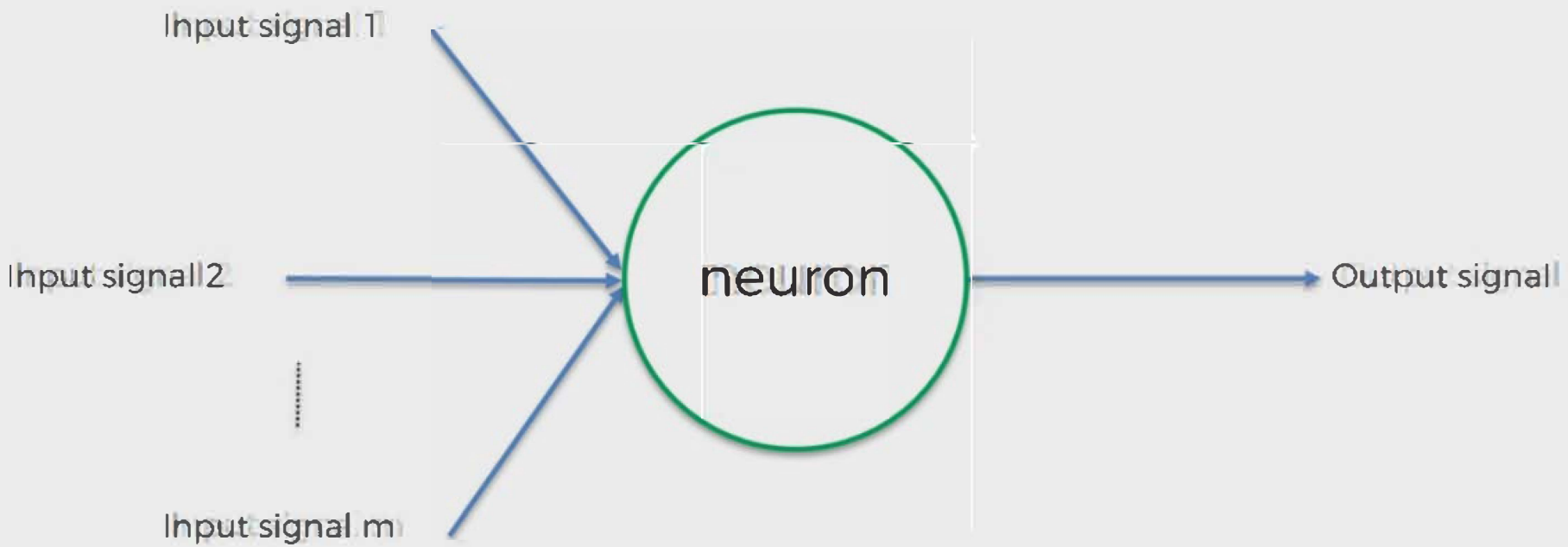

Dendrites
Axon
Neuron

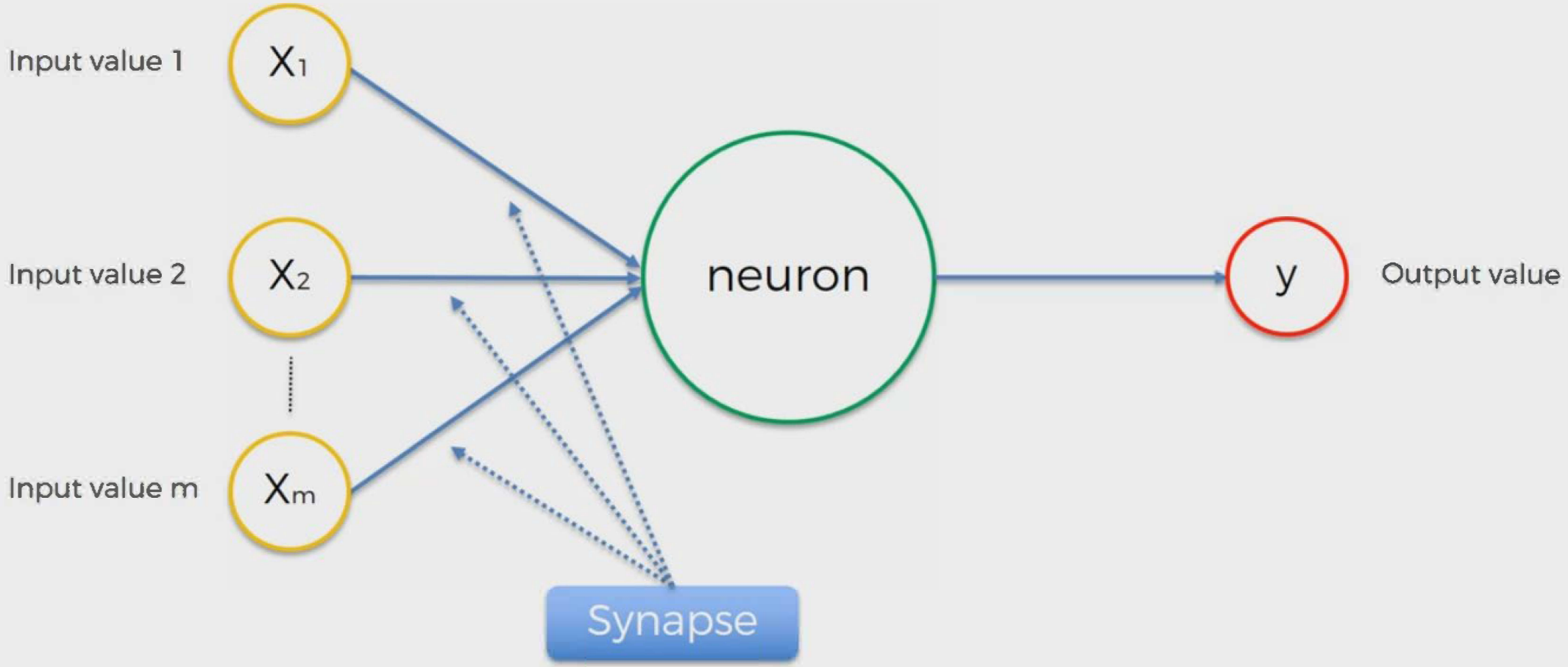

Neuron
Dendrites
Axon
Electrical Impulses
Neurotransmitter Molecules
Receptor
Synapse

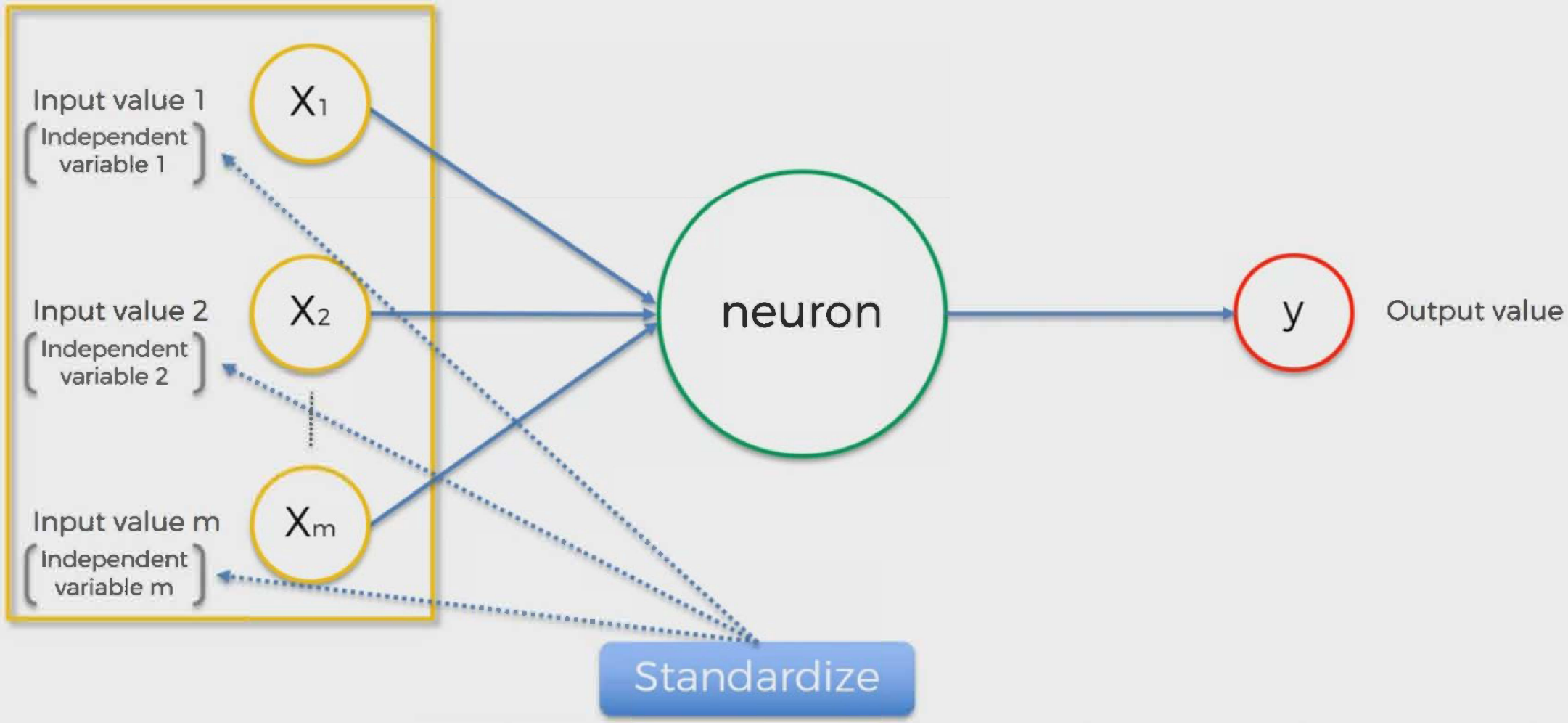

neuron


Node
neuron


Input signal 1
Input signal 2
Input signal m
neuron


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

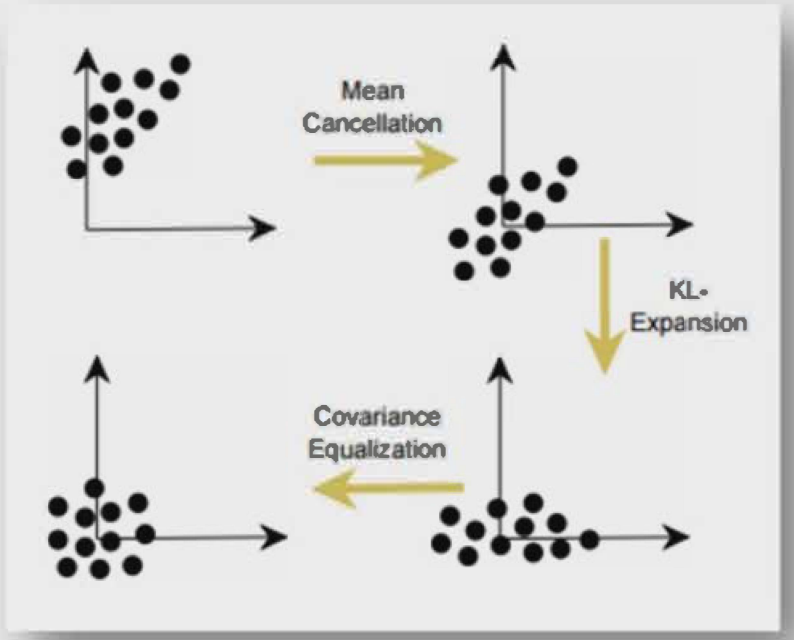

Input value 1
X_1
Input value 2
X_2
Input value m
X_m
neuron
y
Output value
Synapse


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

Additional Reading:

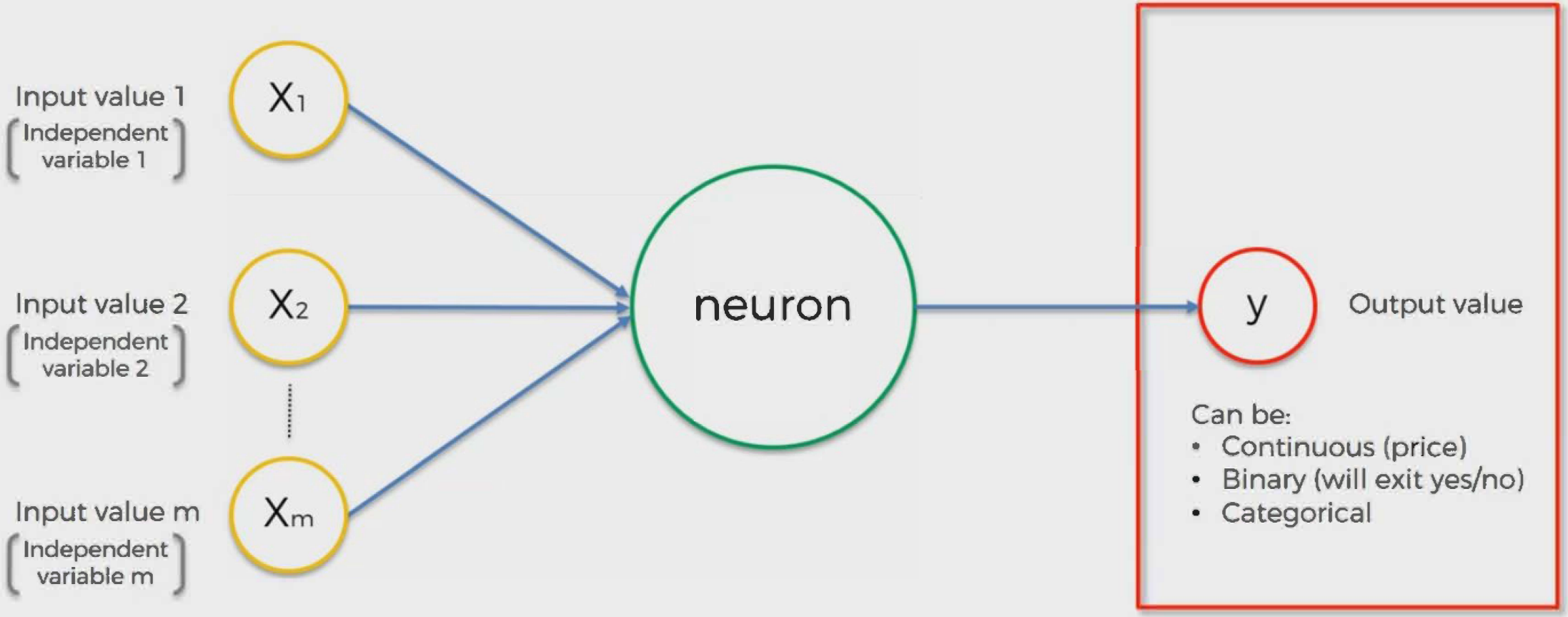
Efficient BackProp

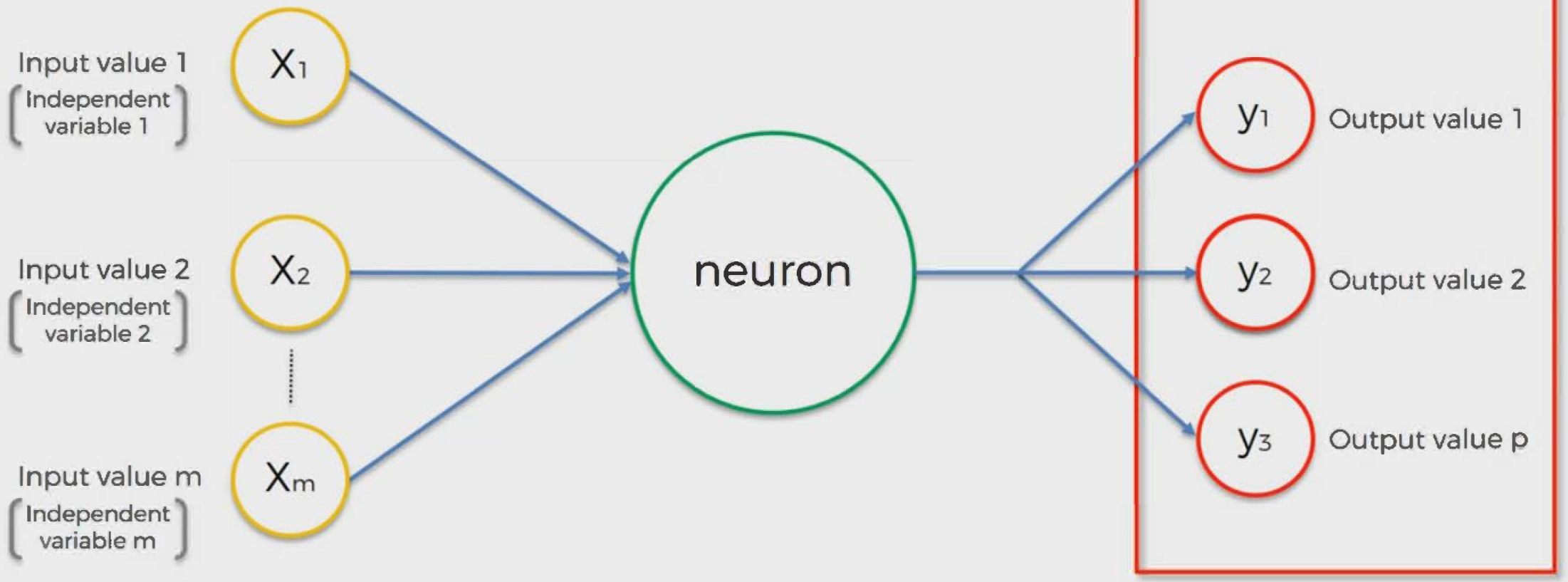
By Yann LeCun et al. (1998)

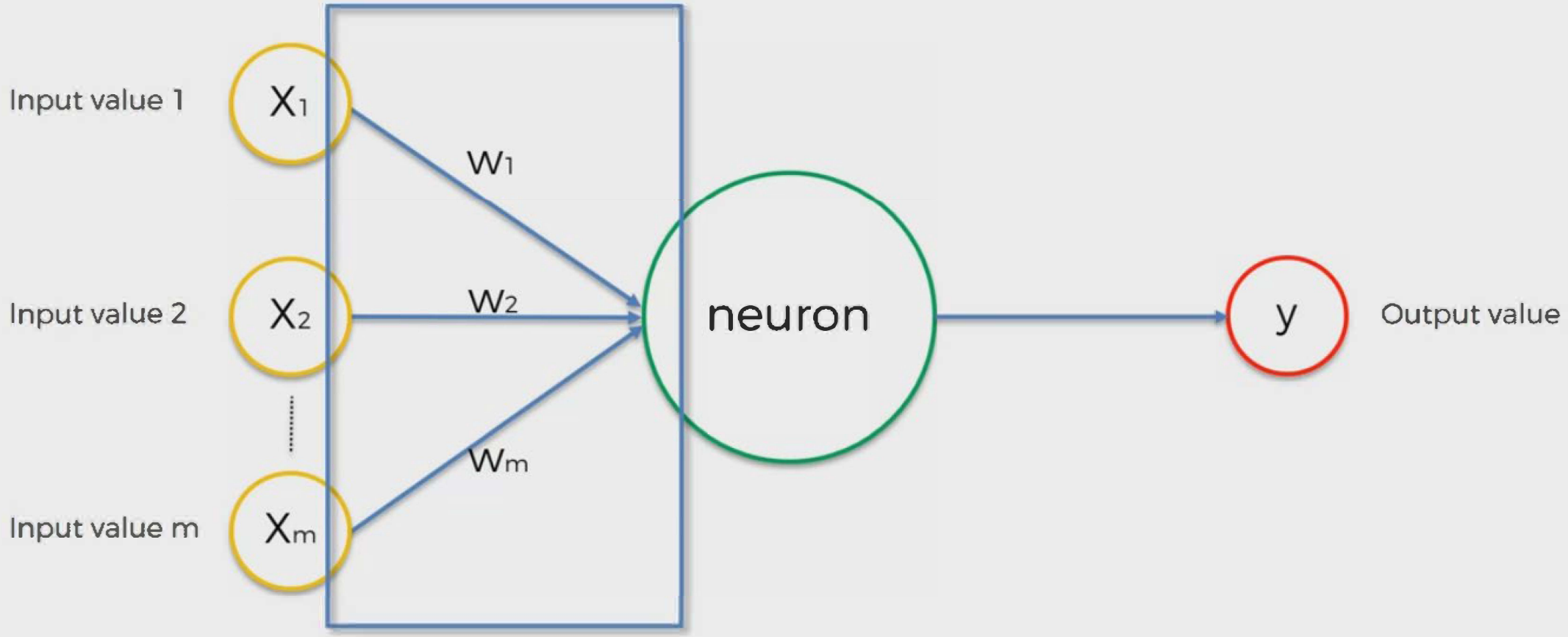


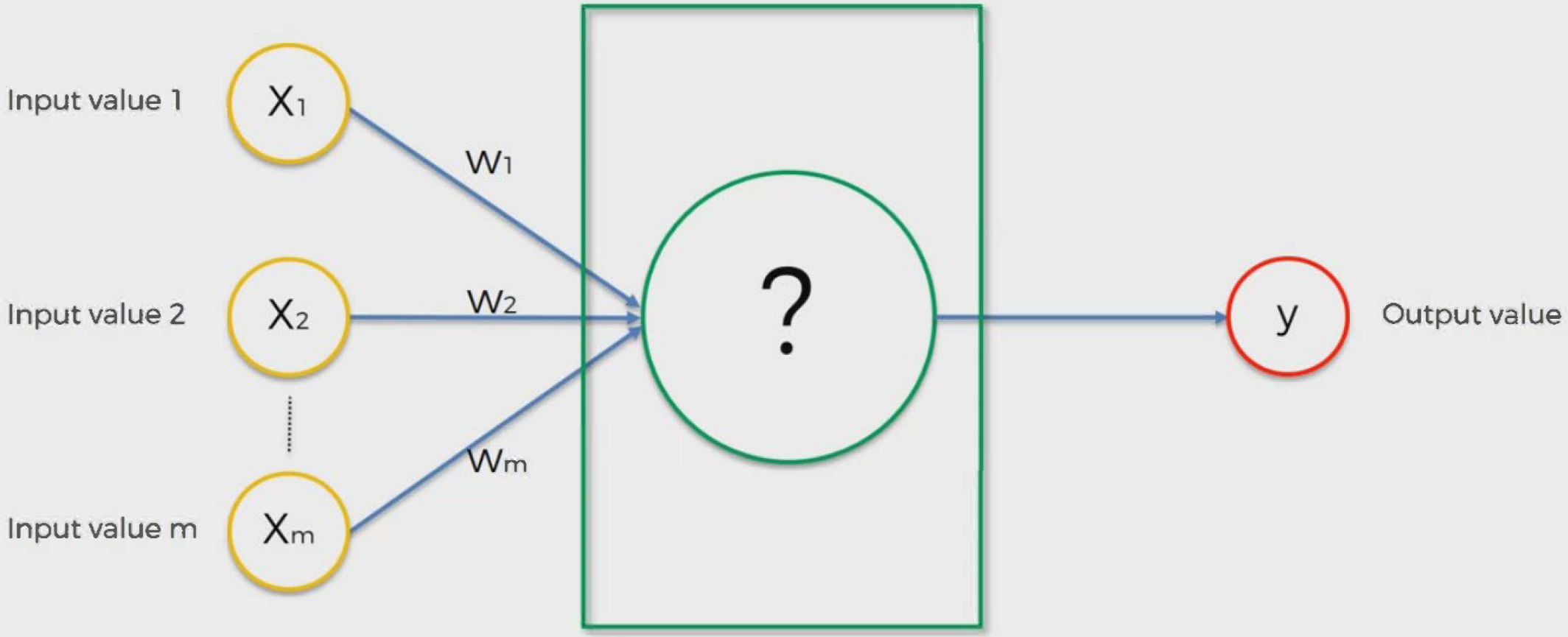
Link:

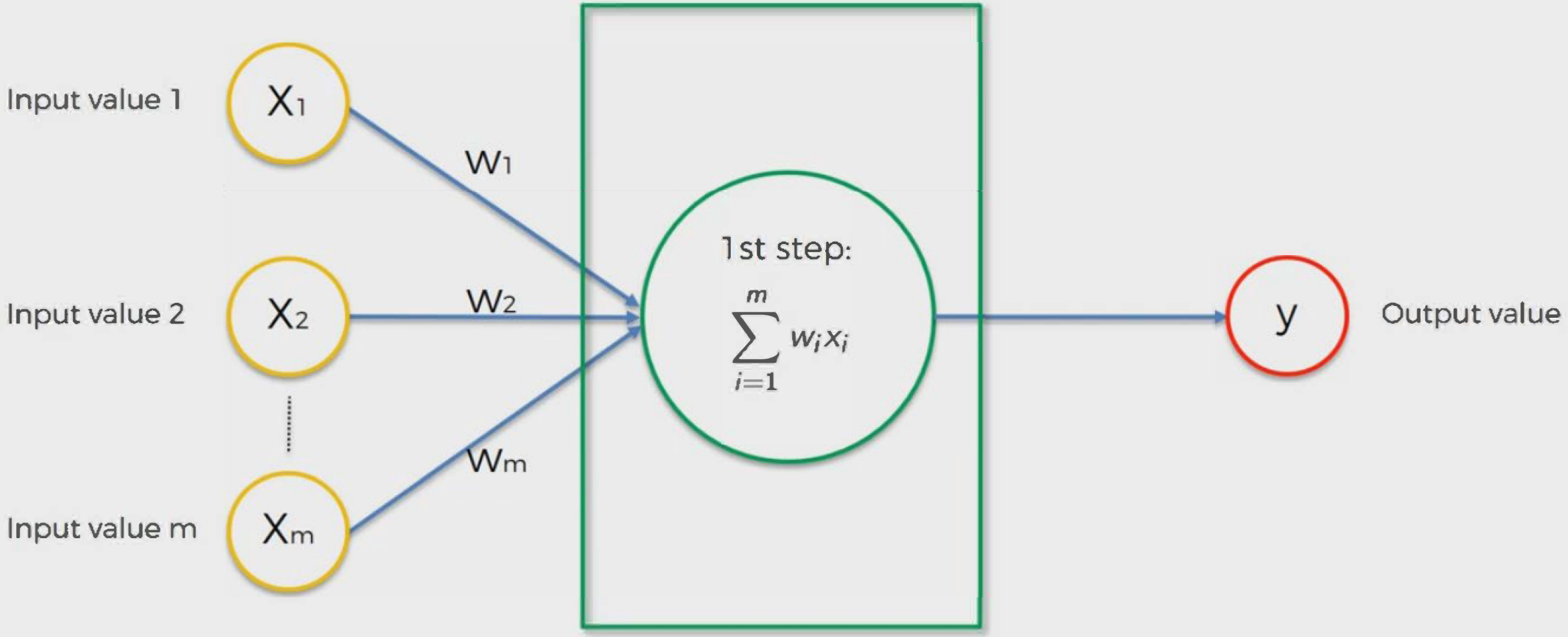
http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

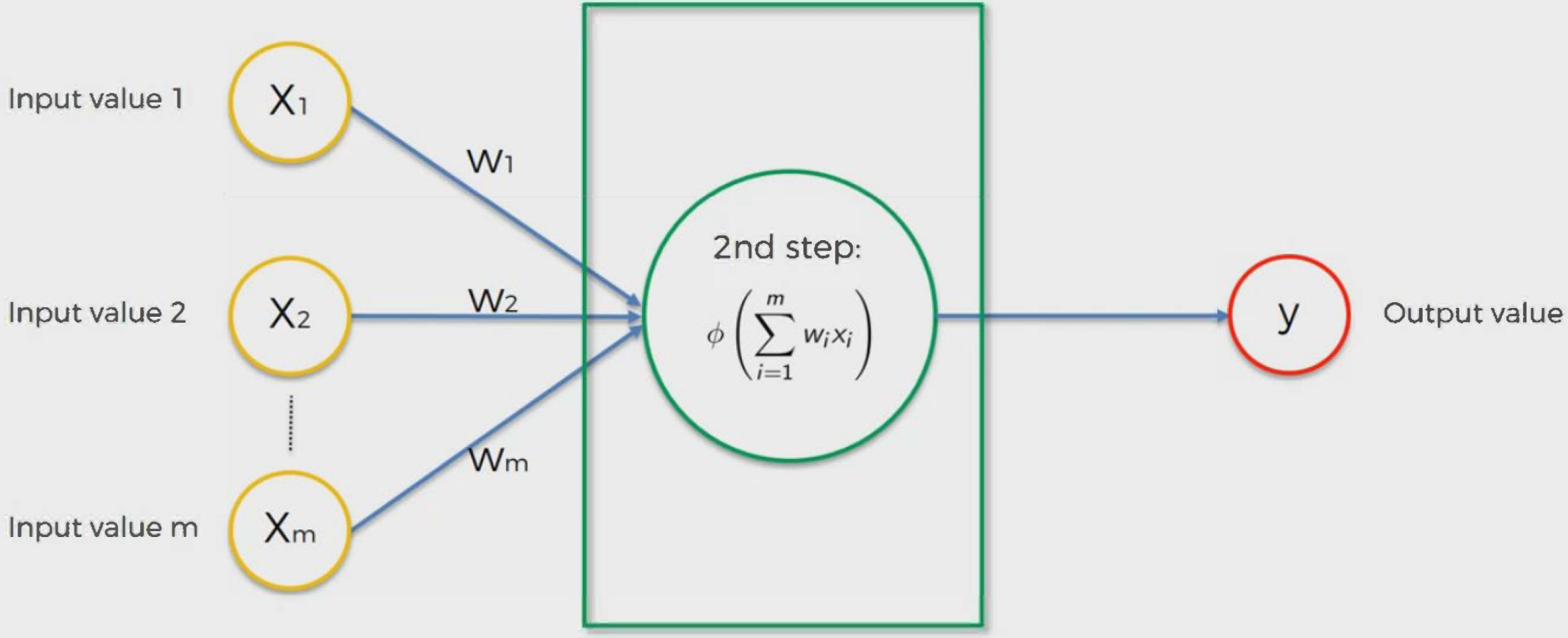

Input value 1
Independent variable 1
X1
Input value 2
Independent variable 2
X2
Input value m
Independent variable m
Xm
neuron
y
Output value
Can be:
• Continuous (price)
• Binary (will exit yes/no)
• Categorical

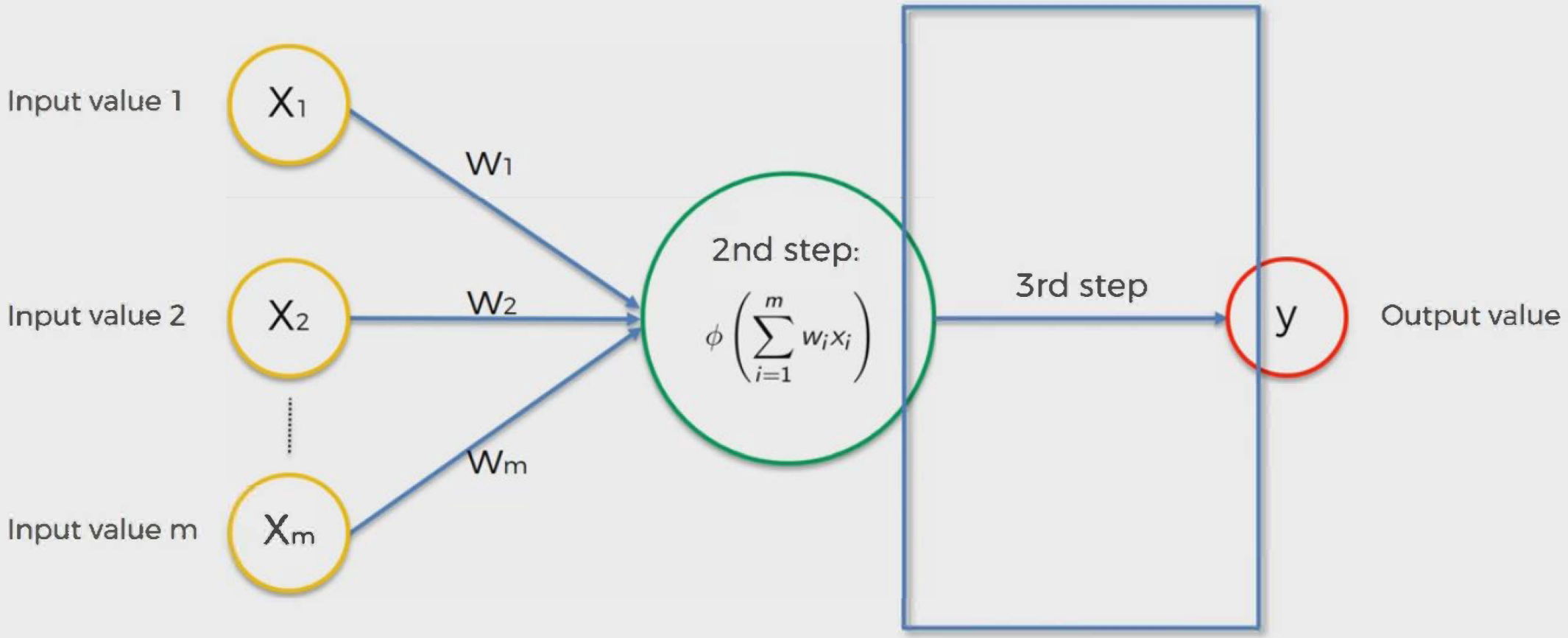

Input value 1
Independent variable 1
X1
Input value 2
Independent variable 2
X2
Input value m
Independent variable m
Xm
neuron
y1
Output value 1
y2
Output value 2
y3
Output value p


Input value 1
X1
W1
Input value 2
X2
W2
neuron
y
Output value
Wm
Input value m
Xm

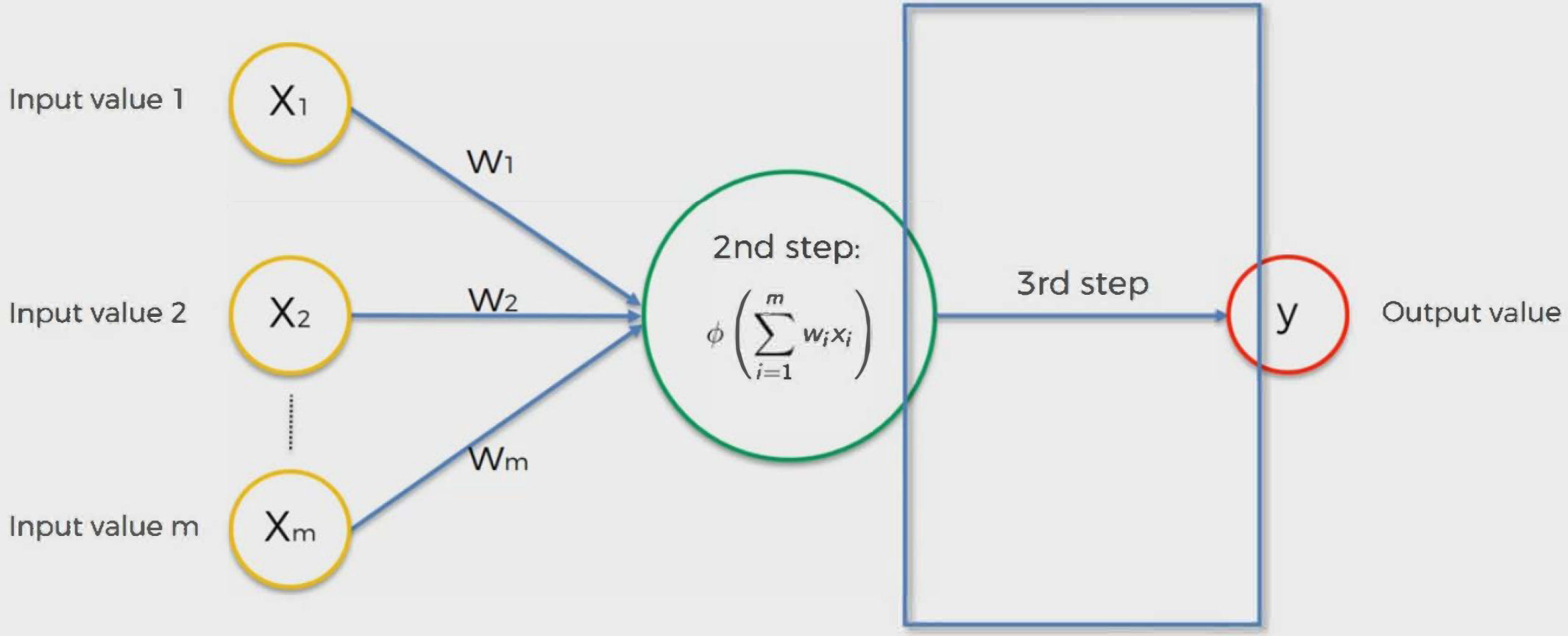

Input value 1
X1
W1
Input value 2
X2
W2
?
y
Output value
Wm
Input value m
Xm


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

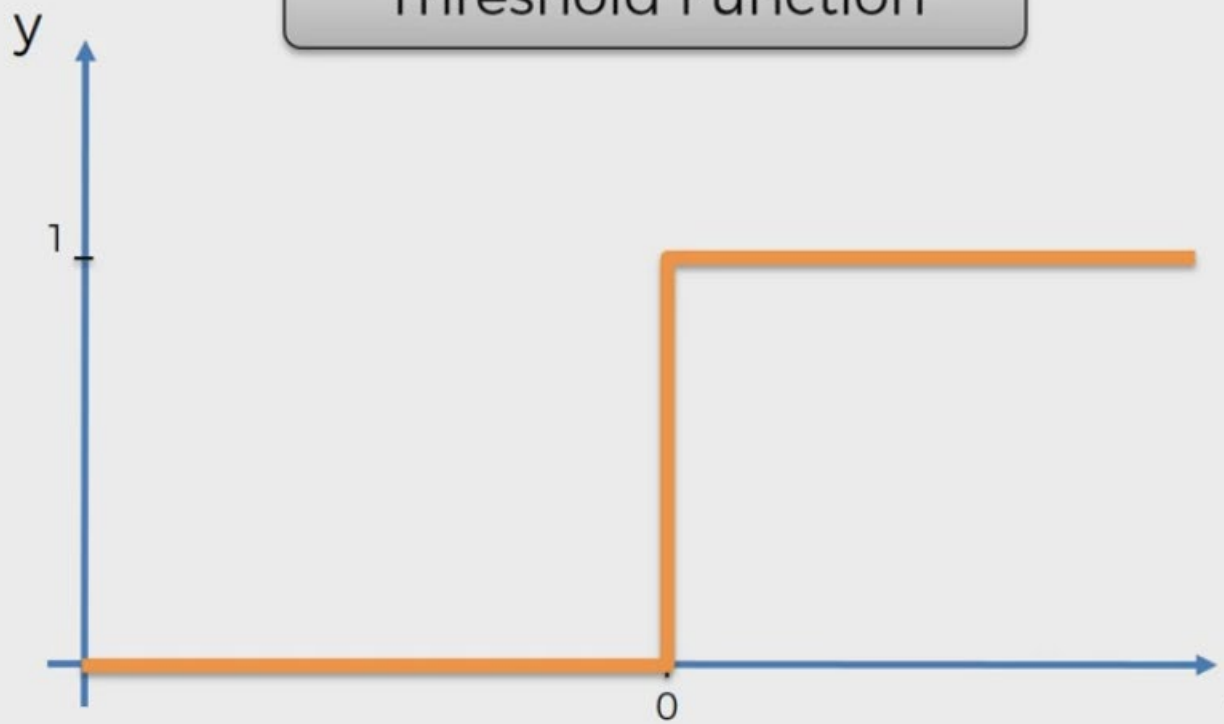

Input value 1
X1
W1
Input value 2
X2
W2
Input value m
Xm
Wm
2nd step:
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
y
Output value


Input value 1
X1
W1
Input value 2
X2
W2
Input value m
Xm
Wm
2nd step:
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
3rd step
y
Output value

The Activation Function

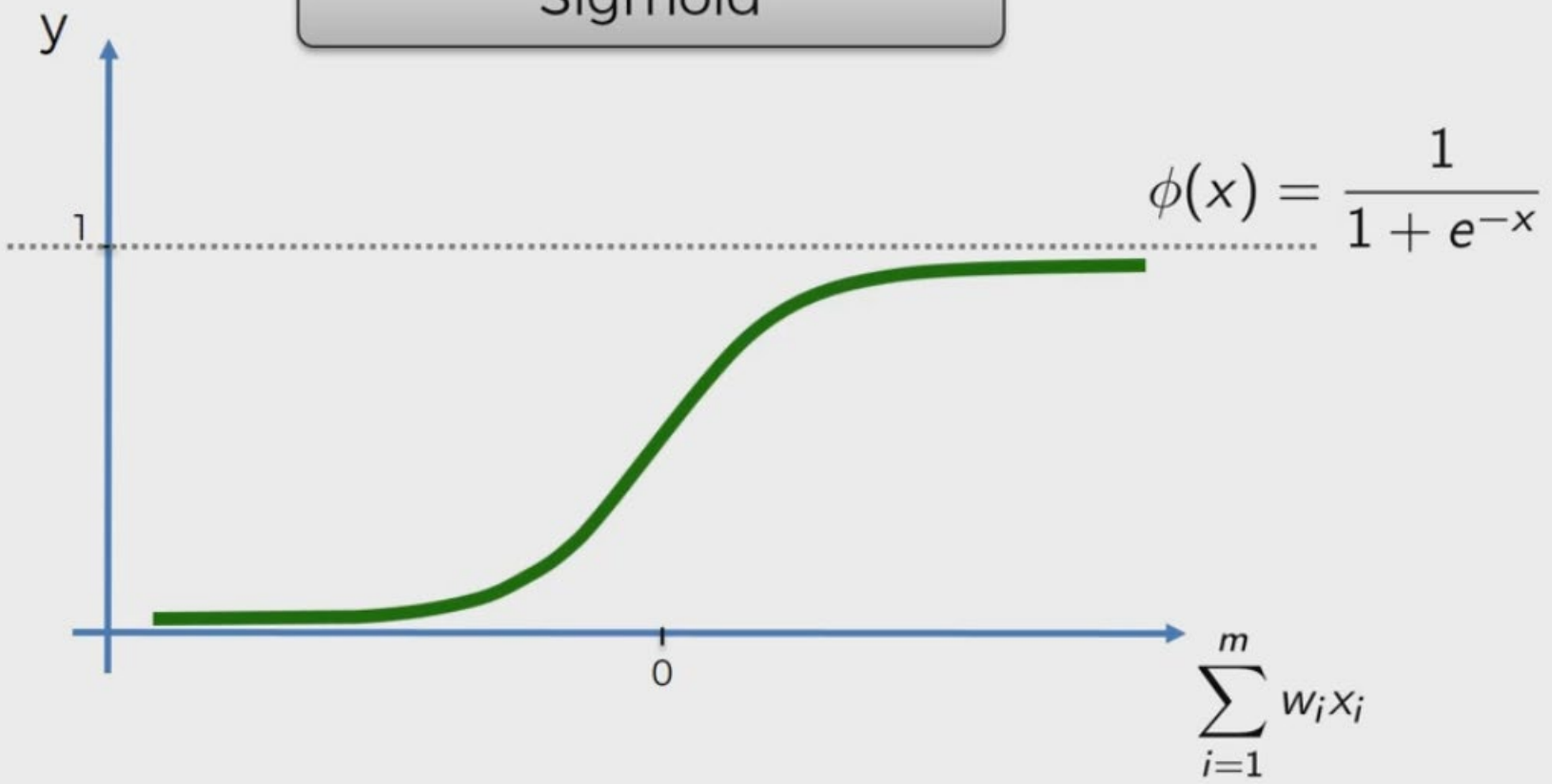

Input value 1
X1
W1
Input value 2
X2
W2
Input value m
Xm
Wm
2nd step:
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
3rd step
y
Output value

Threshold Function

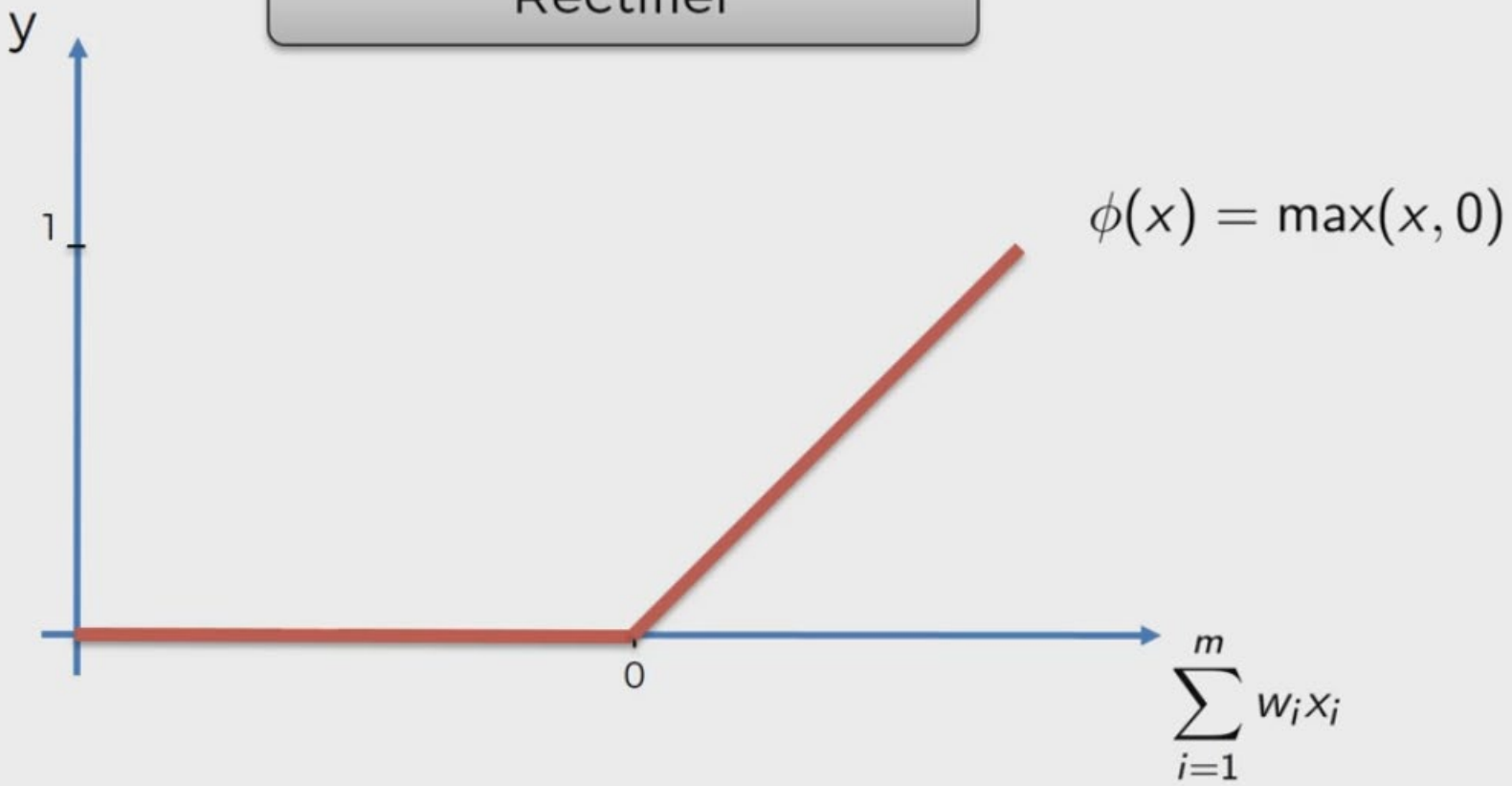


$$\phi(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$

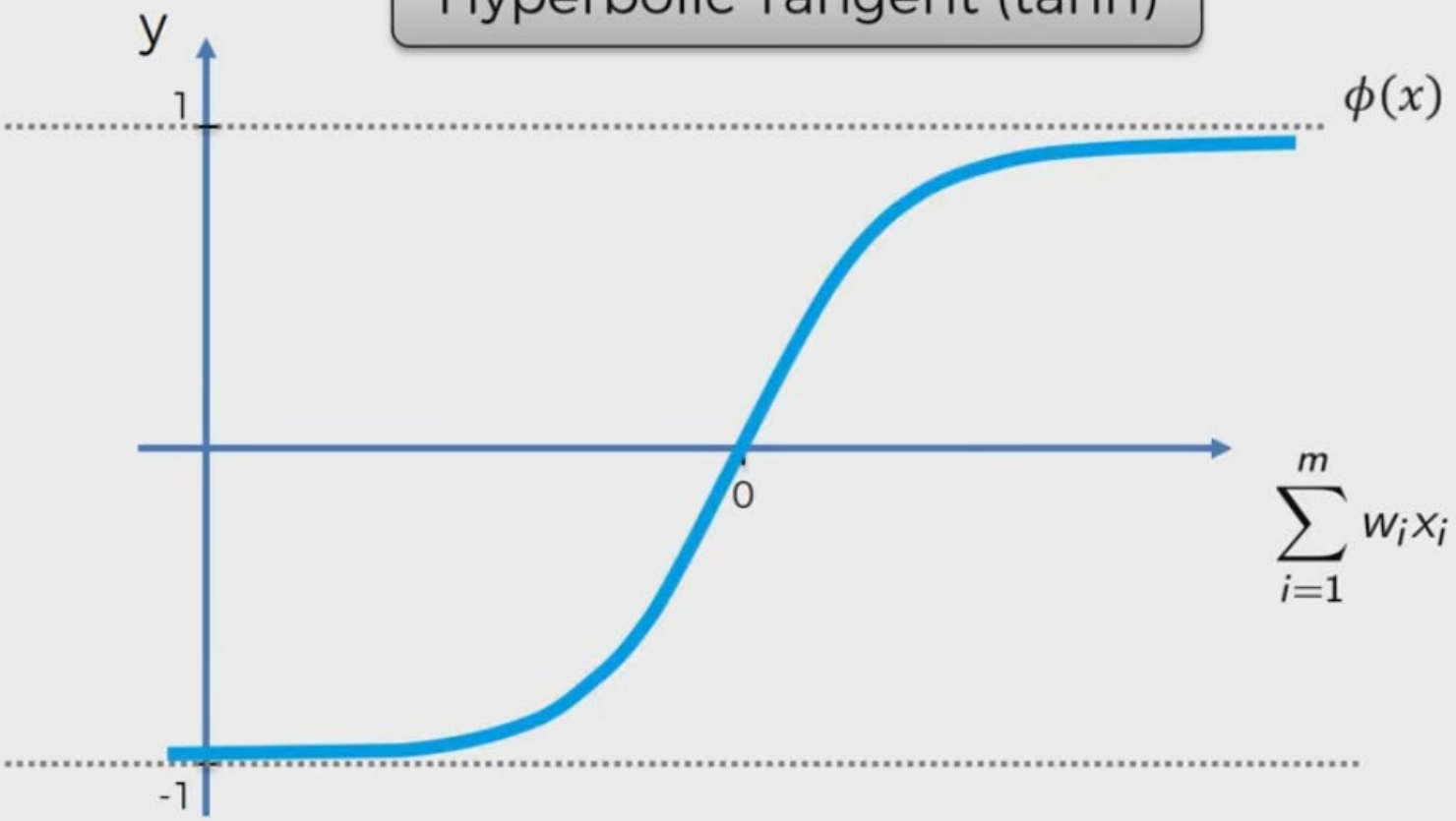
Sigmoid



Rectifier



Hyperbolic Tangent (tanh)

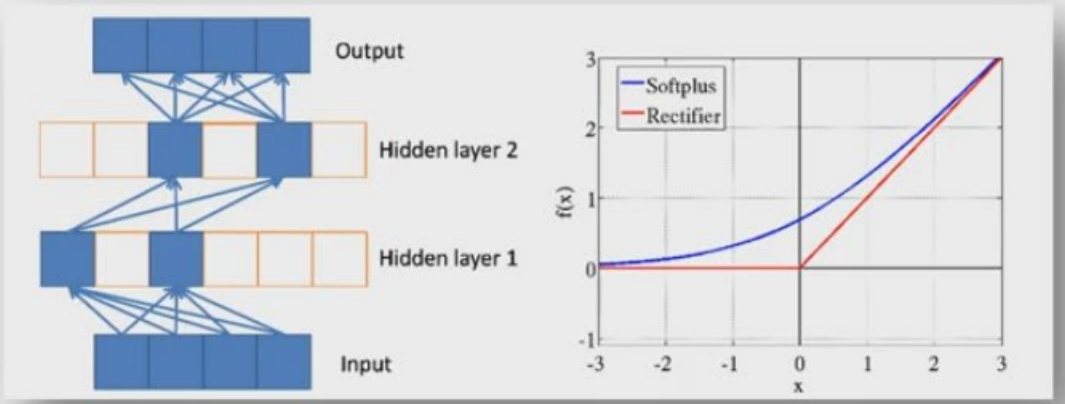


$$\phi(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

Additional Reading:

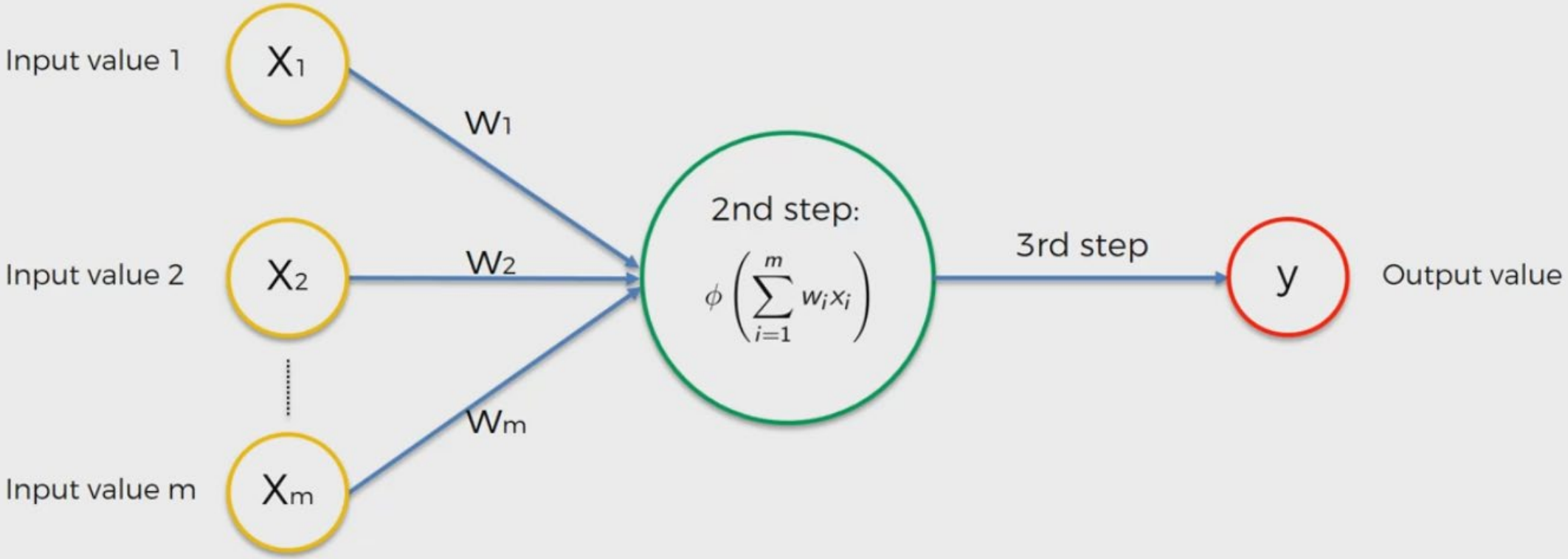
Deep sparse rectifier neural networks

By Xavier Glorot et al. (2011)

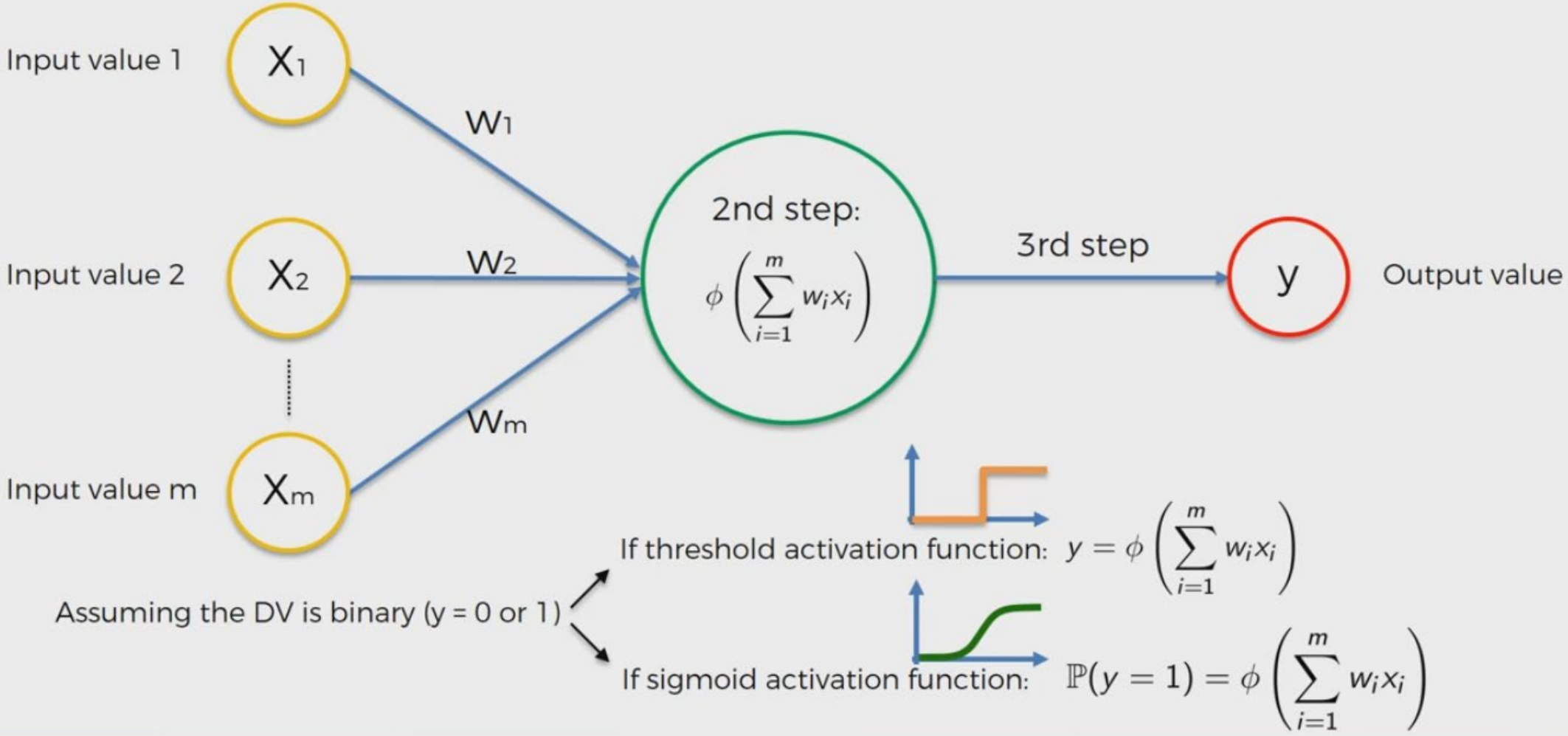


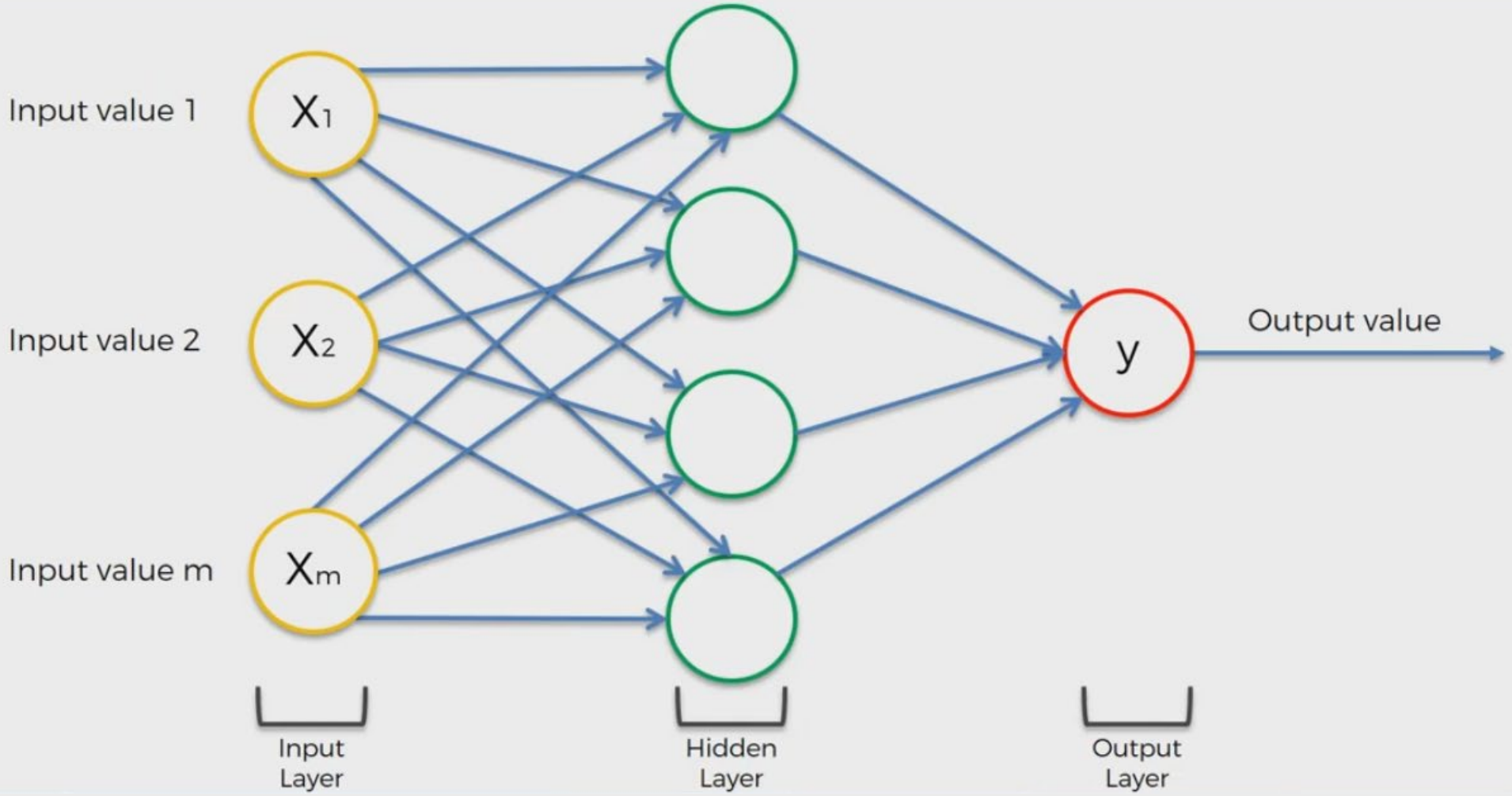
Link:

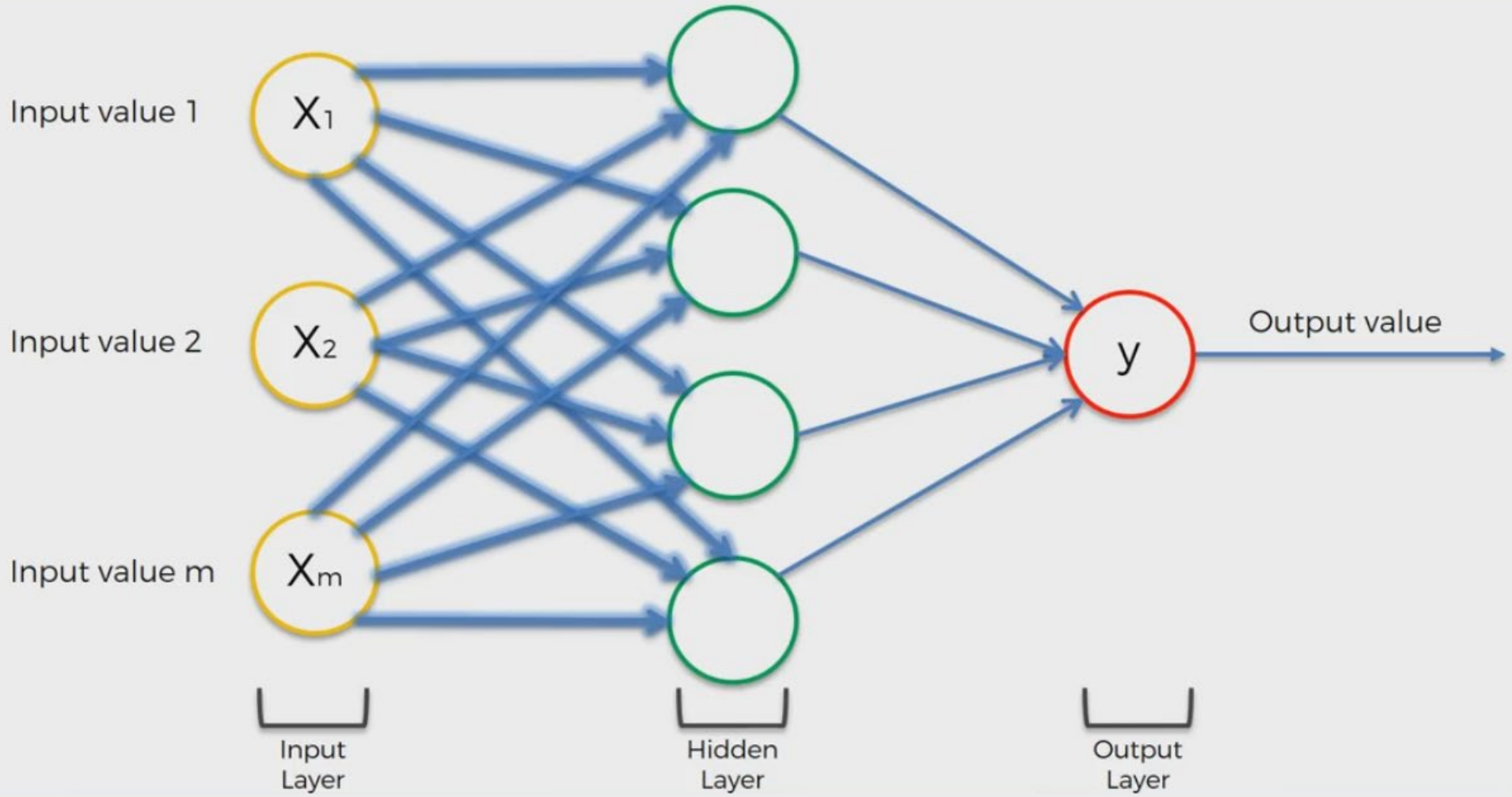
http://jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf

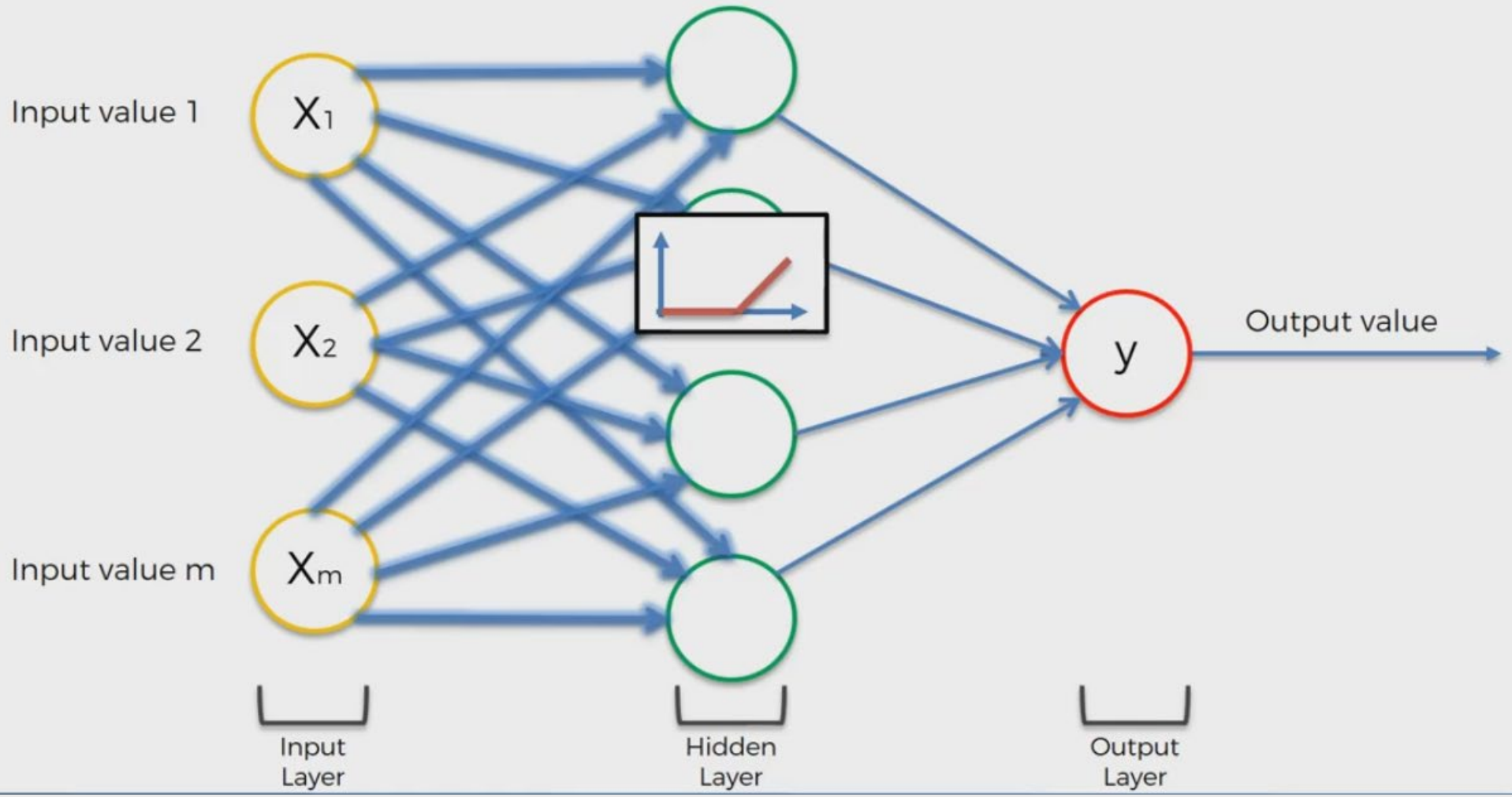


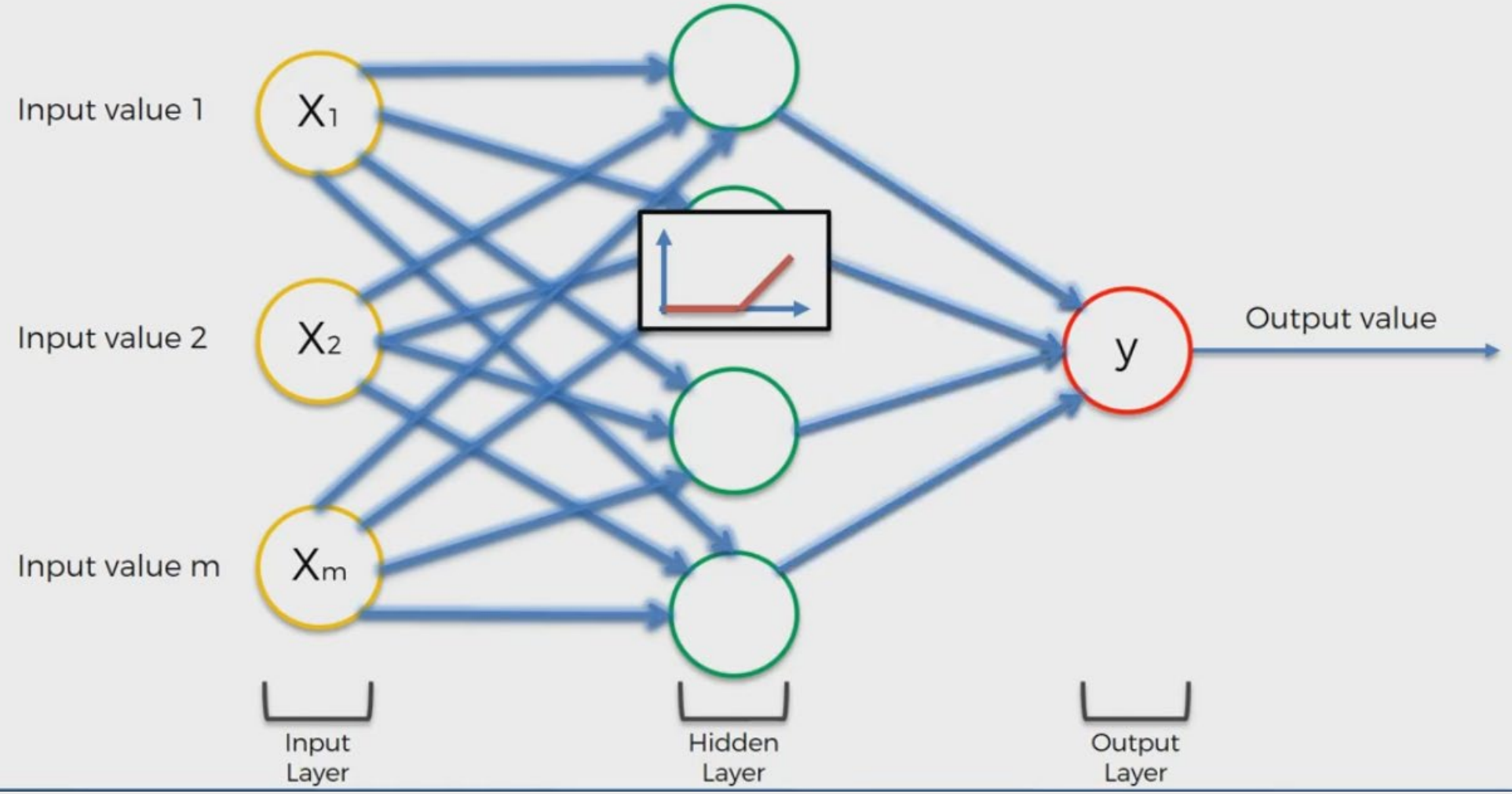
Assuming the DV is binary (y = 0 or 1)

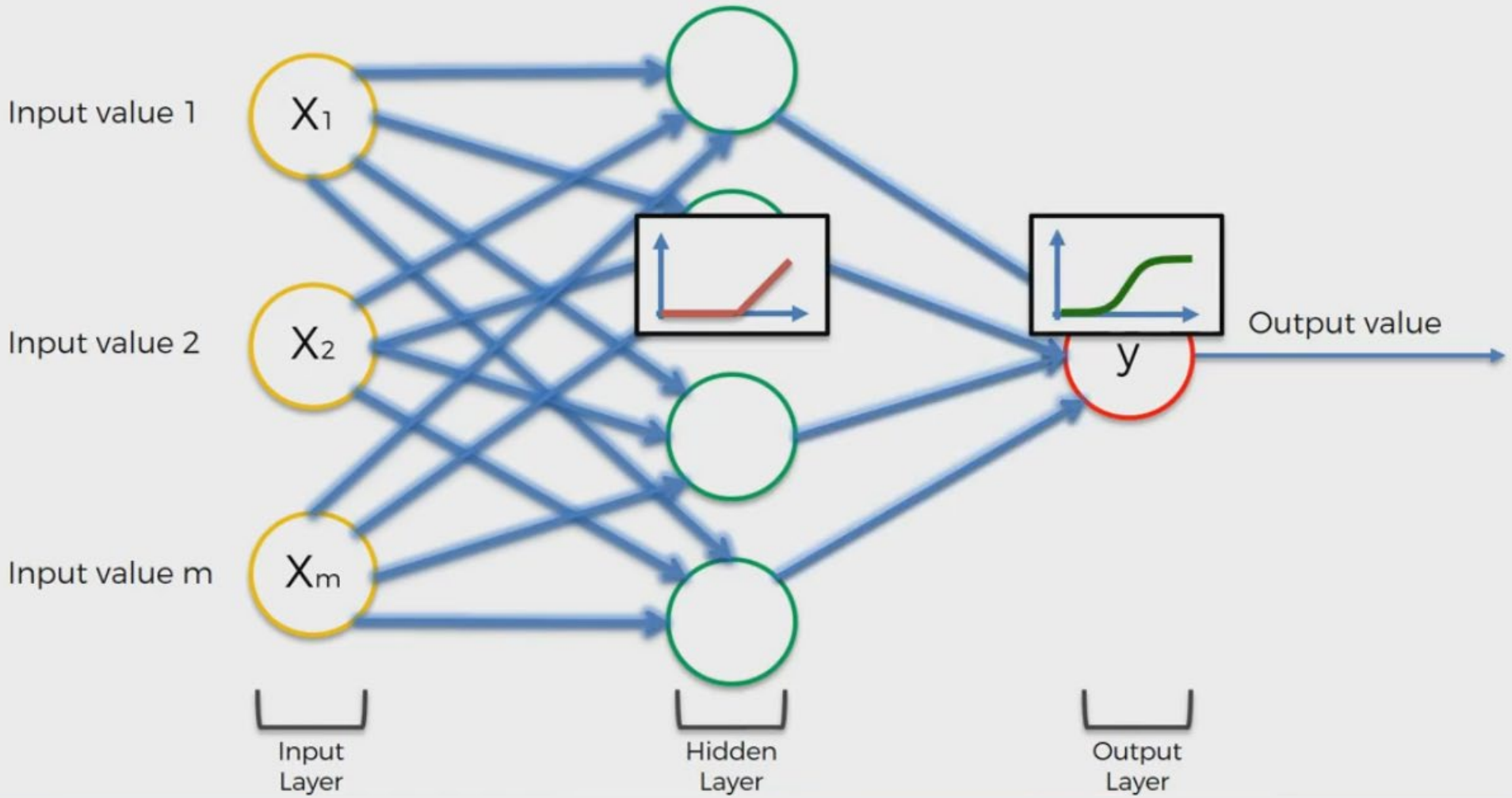

Input value 1
X1
W1
Input value 2
X2
W2
Wm
Input value m
Xm
2nd step:
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
3rd step
y
Output value
Assuming the DV is binary (y = 0 or 1)
If threshold activation function: $y = \phi\left(\sum_{i=1}^{m} w_i x_i\right)$
If sigmoid activation function: $\mathbb{P}(y = 1) = \phi\left(\sum_{i=1}^{m} w_i x_i\right)$

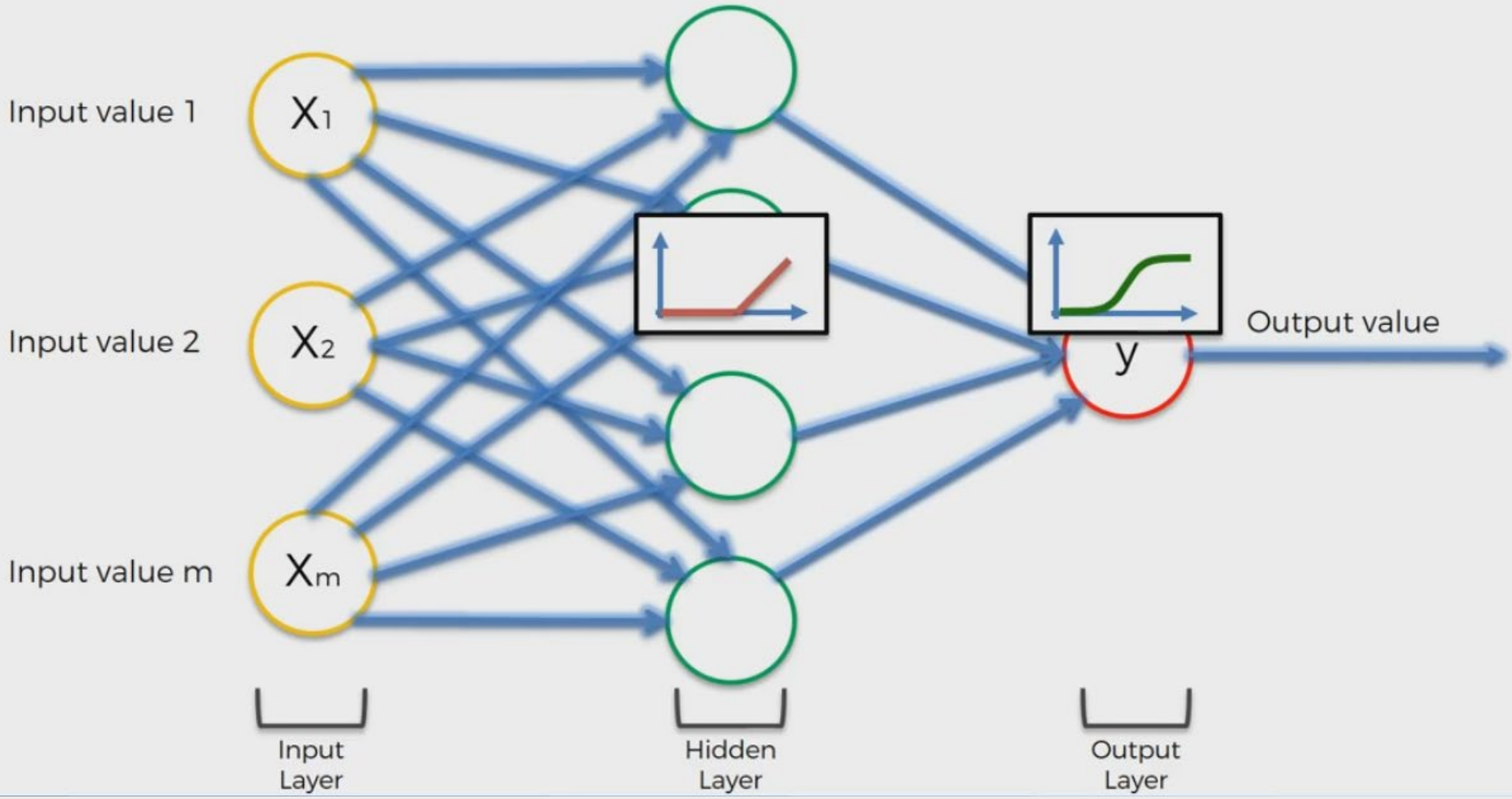

Input value 1
Input value 2
Input value m
X1
X2
Xm
y
Output value
Input Layer
Hidden Layer
Output Layer


Input value 1
Input value 2
Input value m
X1
X2
Xm
y
Output value
Input Layer
Hidden Layer
Output Layer


Input value 1
Input value 2
Input value m
X1
X2
Xm
y
Output value
Input Layer
Hidden Layer
Output Layer


Input value 1
Input value 2
Input value m
X1
X2
Xm
y
Output value
Input Layer
Hidden Layer
Output Layer


Input value 1
Input value 2
Input value m
X_1
X_2
X_m
y
Output value
Input Layer
Hidden Layer
Output Layer


Input value 1
Input value 2
Input value m
X_1
X_2
X_m
y
Output value
Input Layer
Hidden Layer
Output Layer

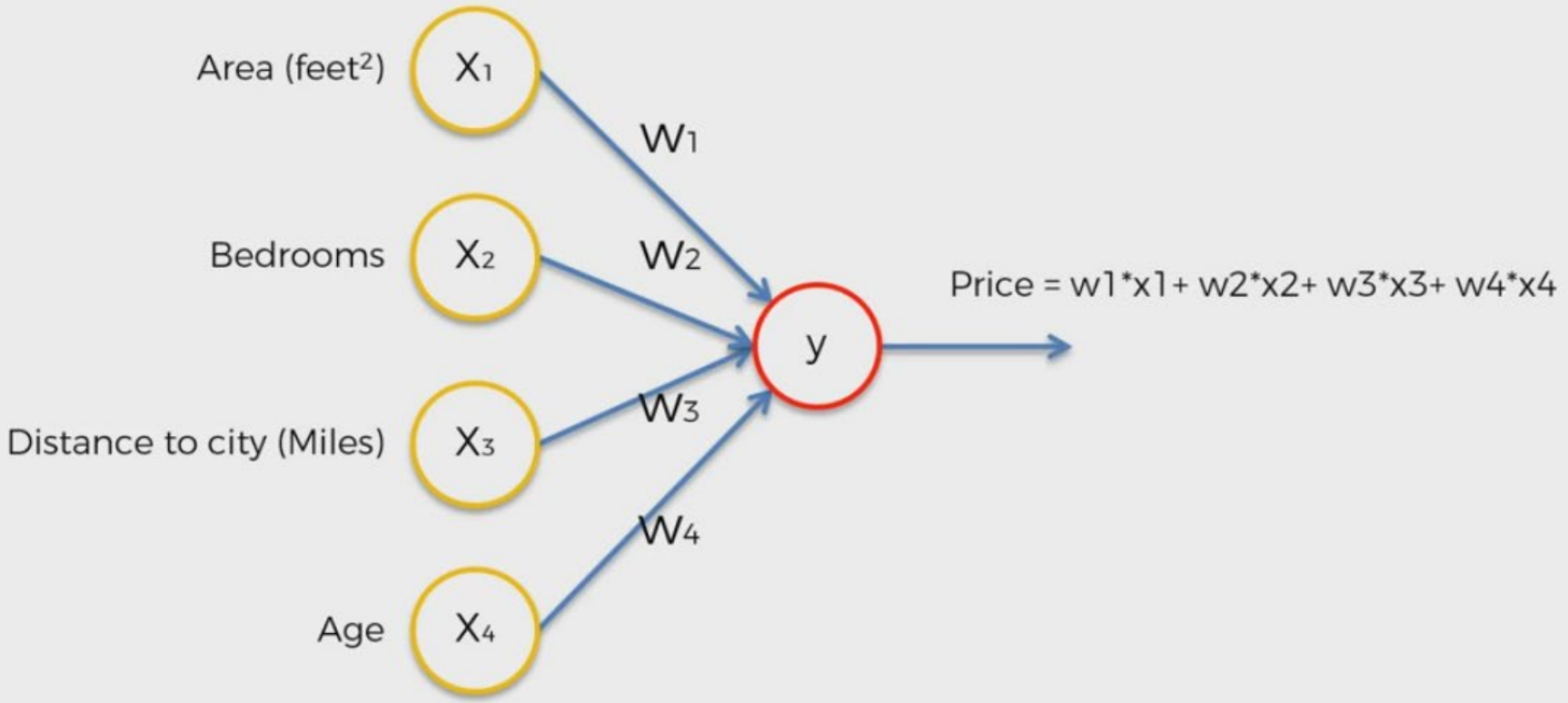
How do NNs Work?

Area (feet^2) X_1

Bedrooms X_2

Distance to city (Miles) X_3

Age X_4

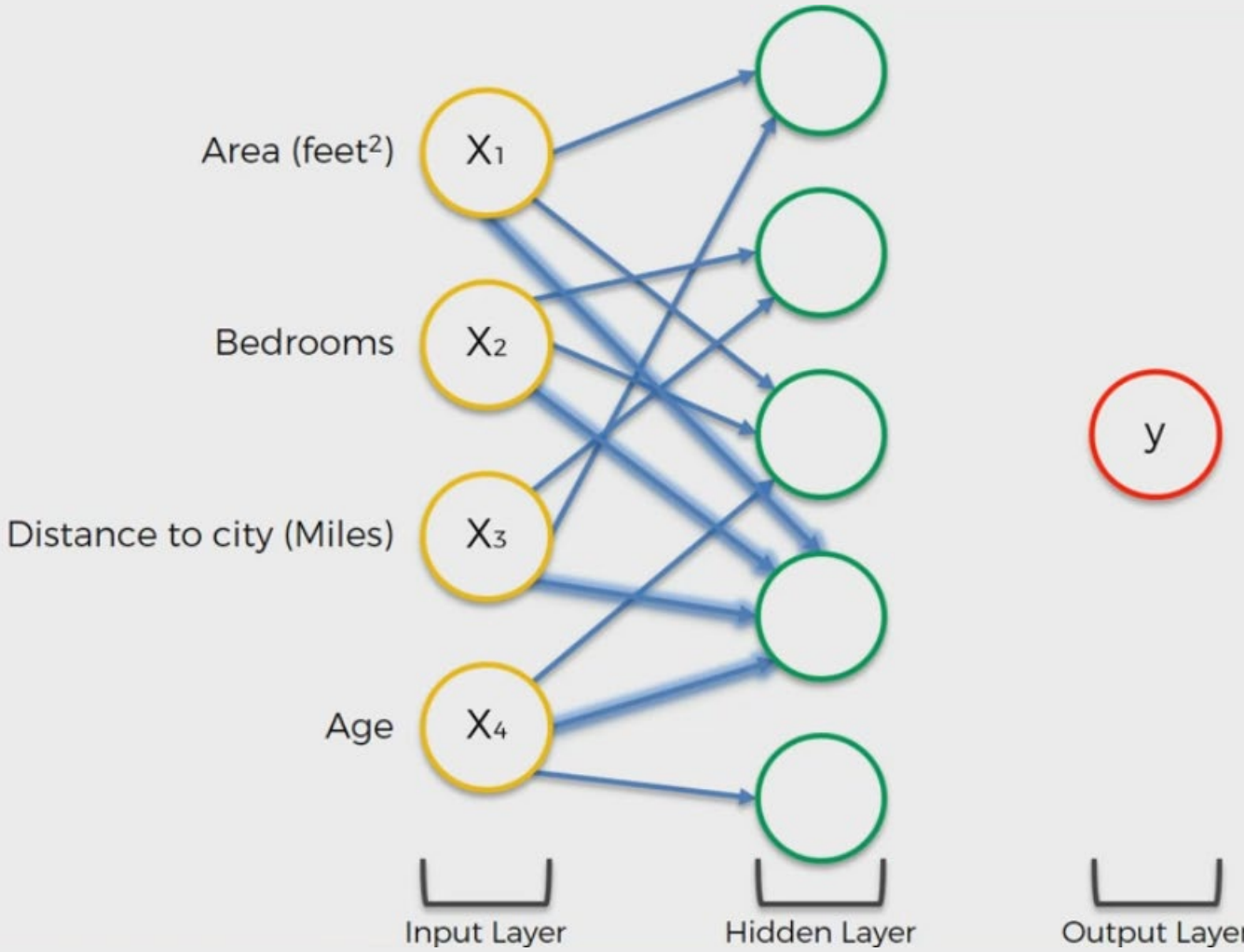

Area (feet²)
X1
W1
Bedrooms
X2
W2
Distance to city (Miles)
X3
W3
Age
X4
W4
y
Price = w1*x1+ w2*x2+ w3*x3+ w4*x4

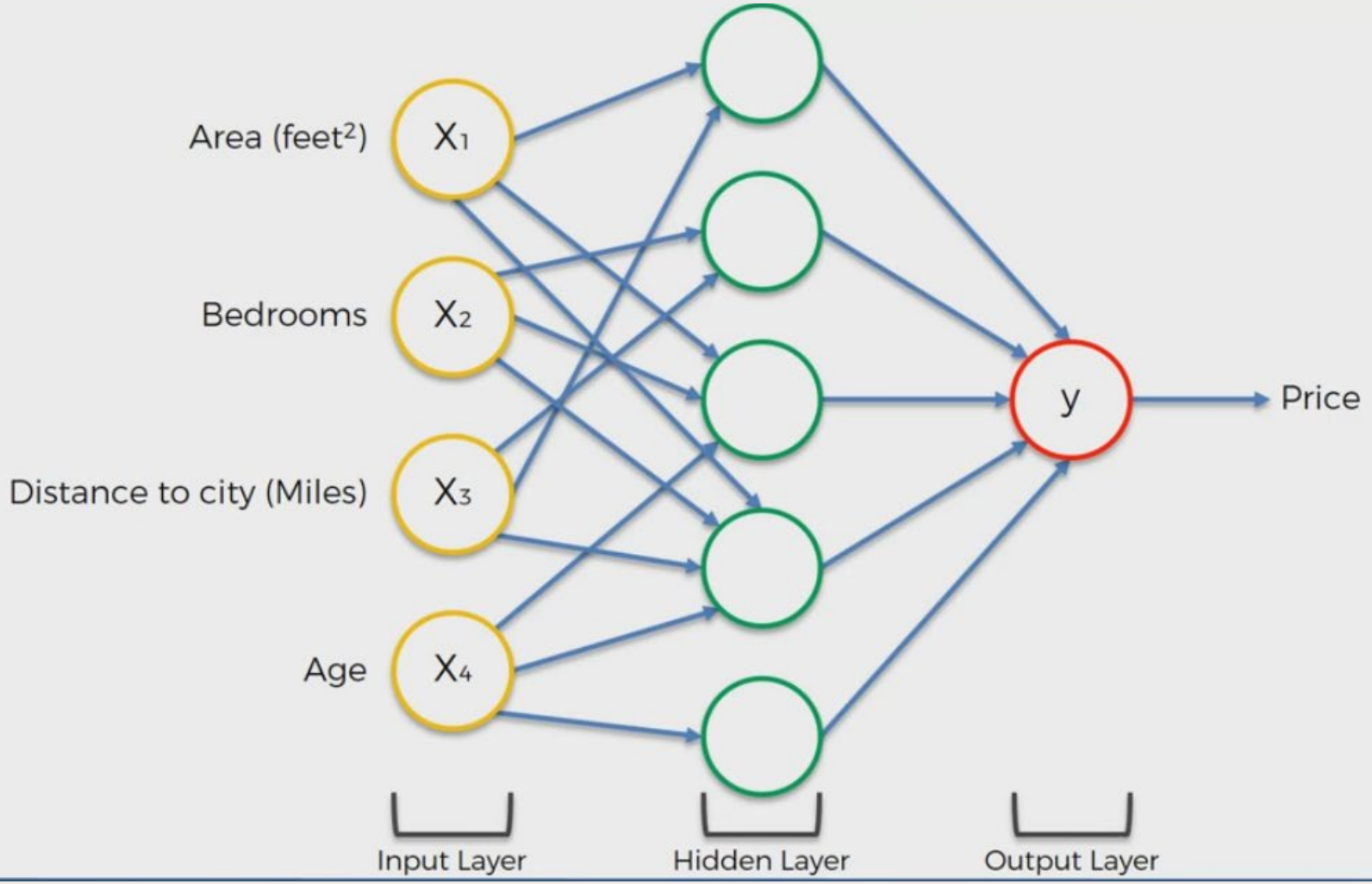

Area (feet²)
X1
Bedrooms
X2
Distance to city (Miles)
X3
Age
X4
y


Area (feet²)
X1
Bedrooms
X2
Distance to city (Miles)
X3
Age
X4
y


Area (feet²)
X1
Bedrooms
X2
Distance to city (Miles)
X3
Age
X4
y

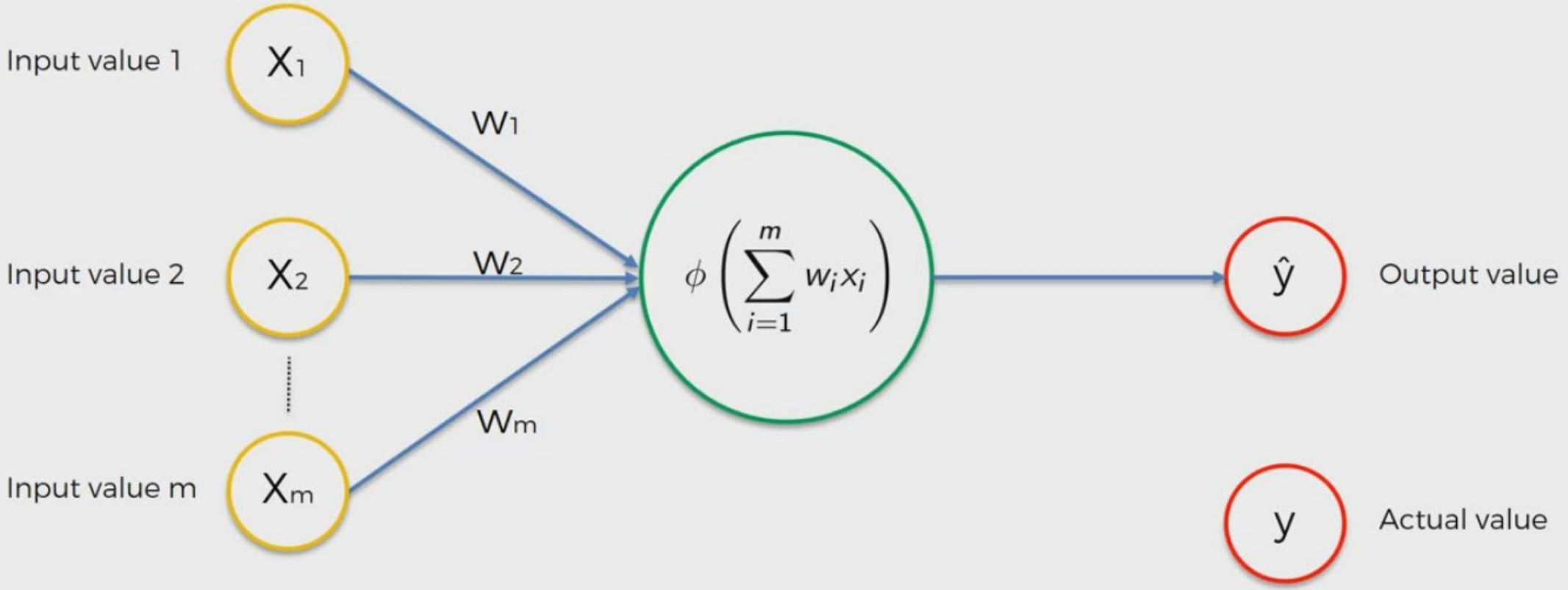

Area (feet²)
X1
Bedrooms
X2
Distance to city (Miles)
X3
Age
X4
y

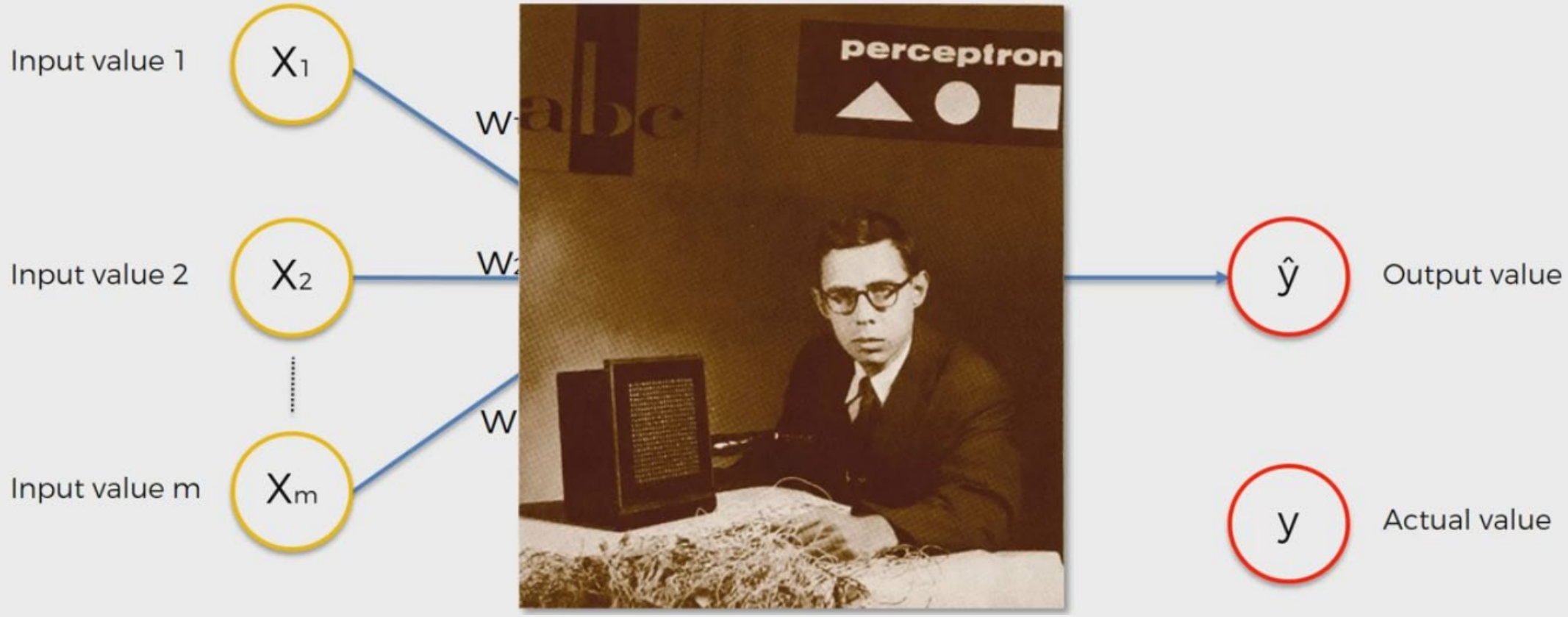

Area (feet²)
X1
Bedrooms
X2
Distance to city (Miles)
X3
Age
X4
y
Input Layer
Hidden Layer
Output Layer

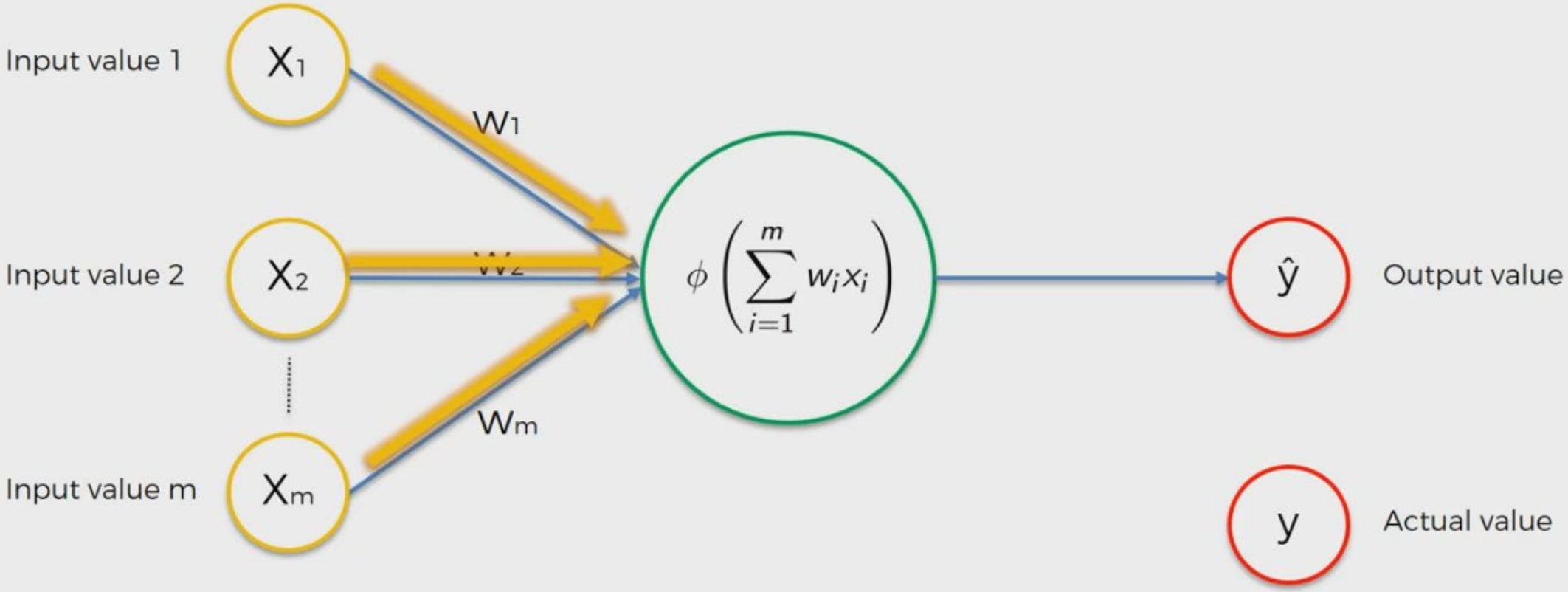

Area (feet²)
X1
Bedrooms
X2
Distance to city (Miles)
X3
Age
X4
y
Price
Input Layer
Hidden Layer
Output Layer

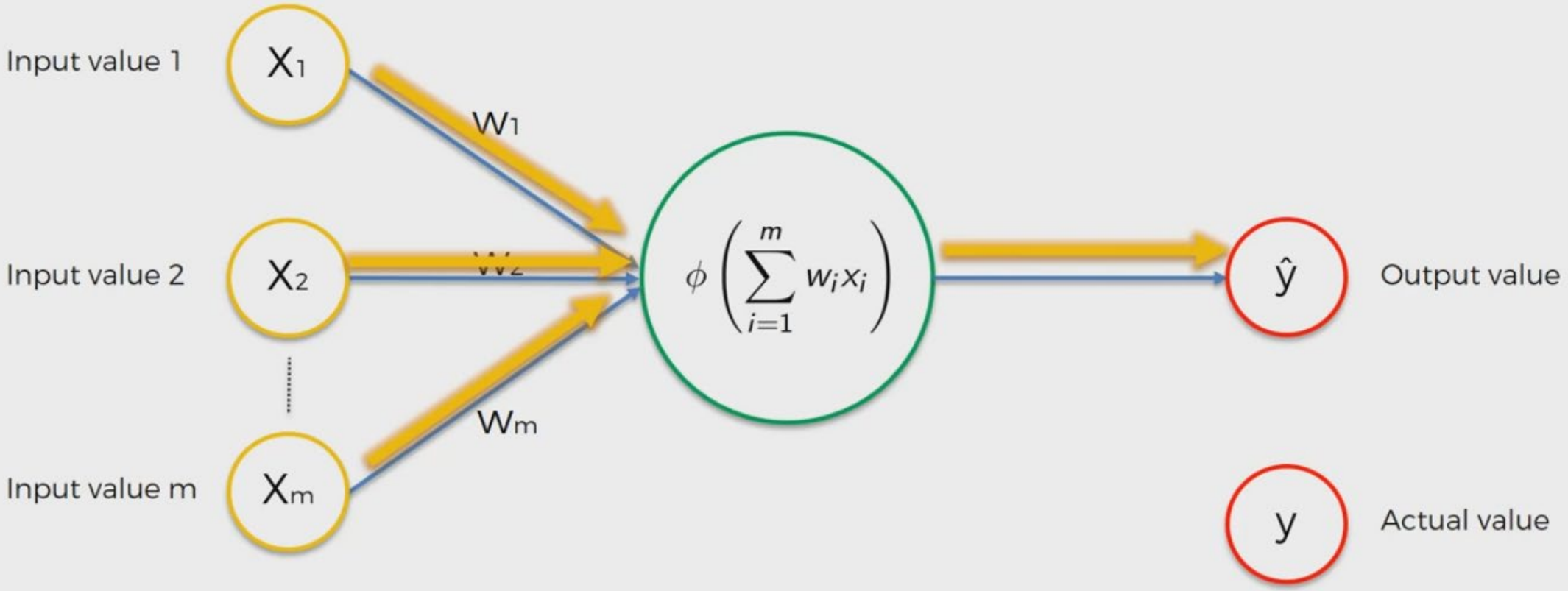
How do NNs Learn?

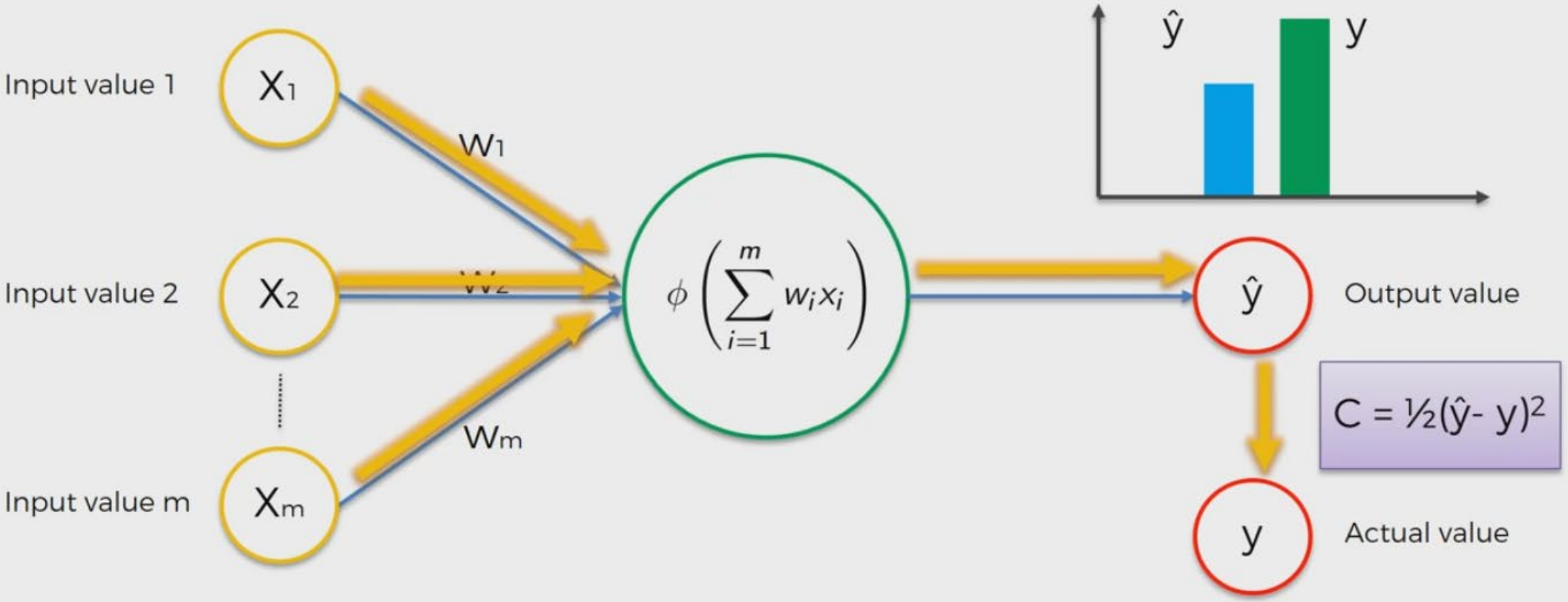

return c; }
(a[c]);
inp_array.length;
(c.push(inp_array[a]),
inp_array));
-1 < b
use_array(a, b)
for

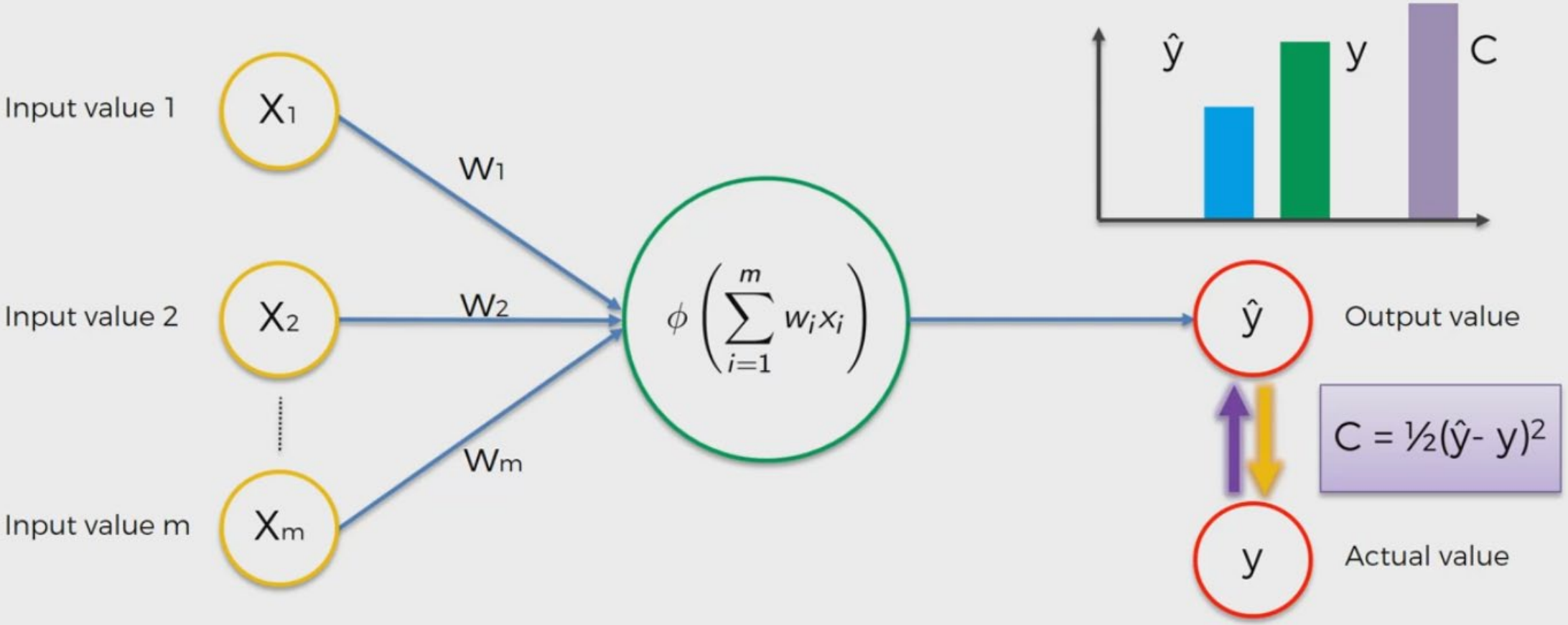

Input value 1
X1
W1
Input value 2
X2
W2
Wm
Input value m
Xm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
Output value
y
Actual value

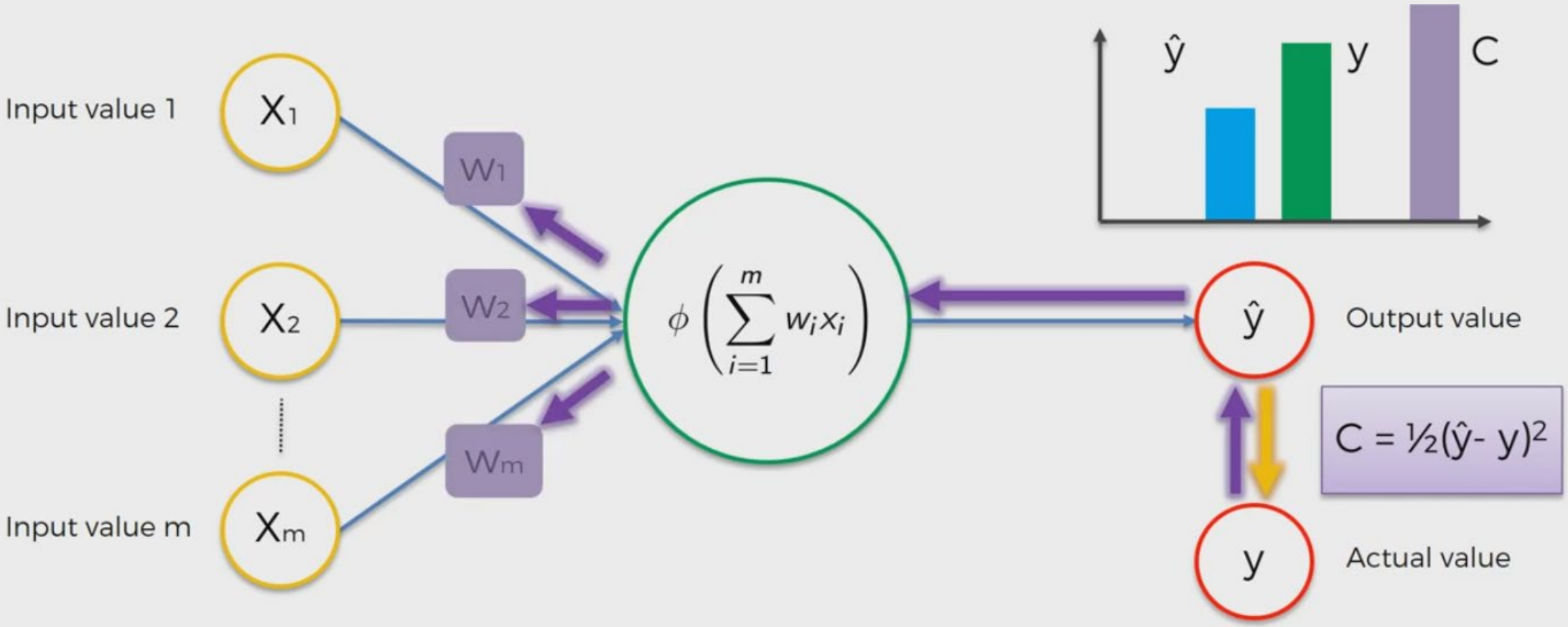

Input value 1
X_1
Input value 2
X_2
Input value m
X_m
perceptron
$\hat{y}$
Output value
y
Actual value

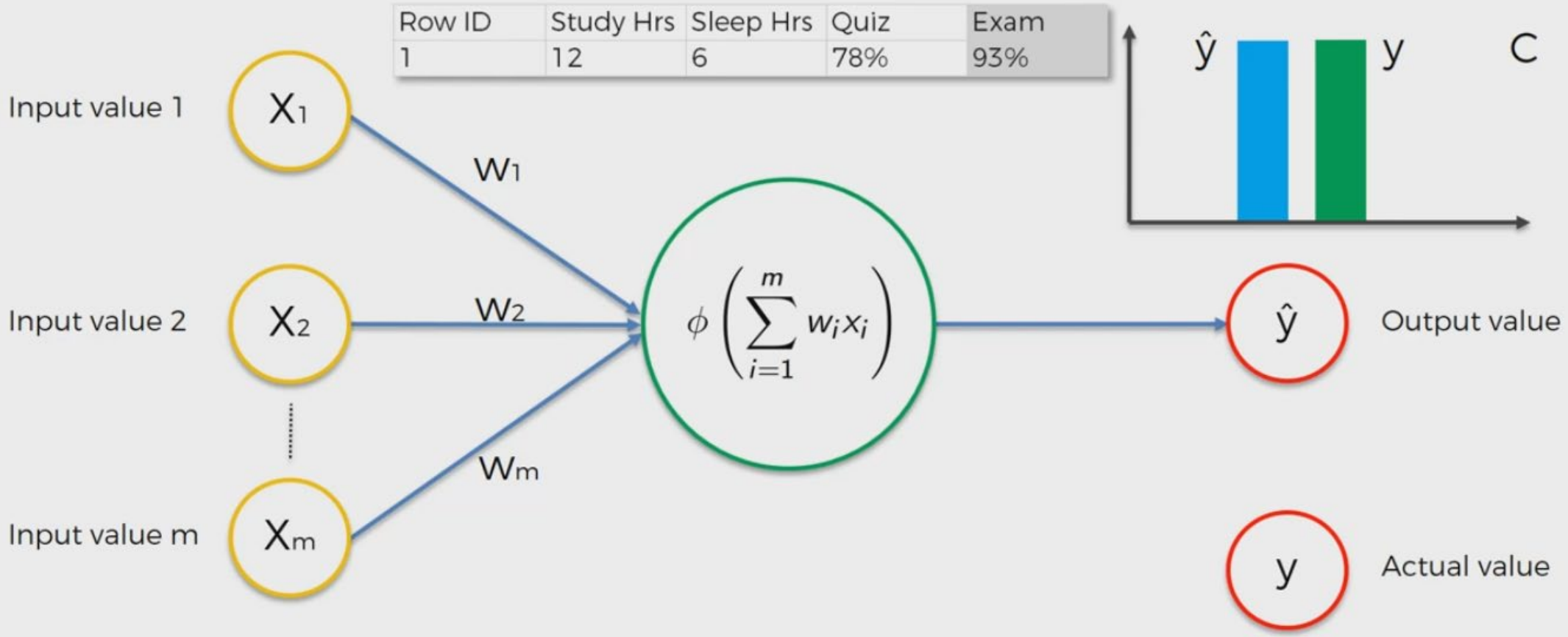

Input value 1
X1
W1
Input value 2
X2
Wm
Input value m
Xm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
Output value
y
Actual value

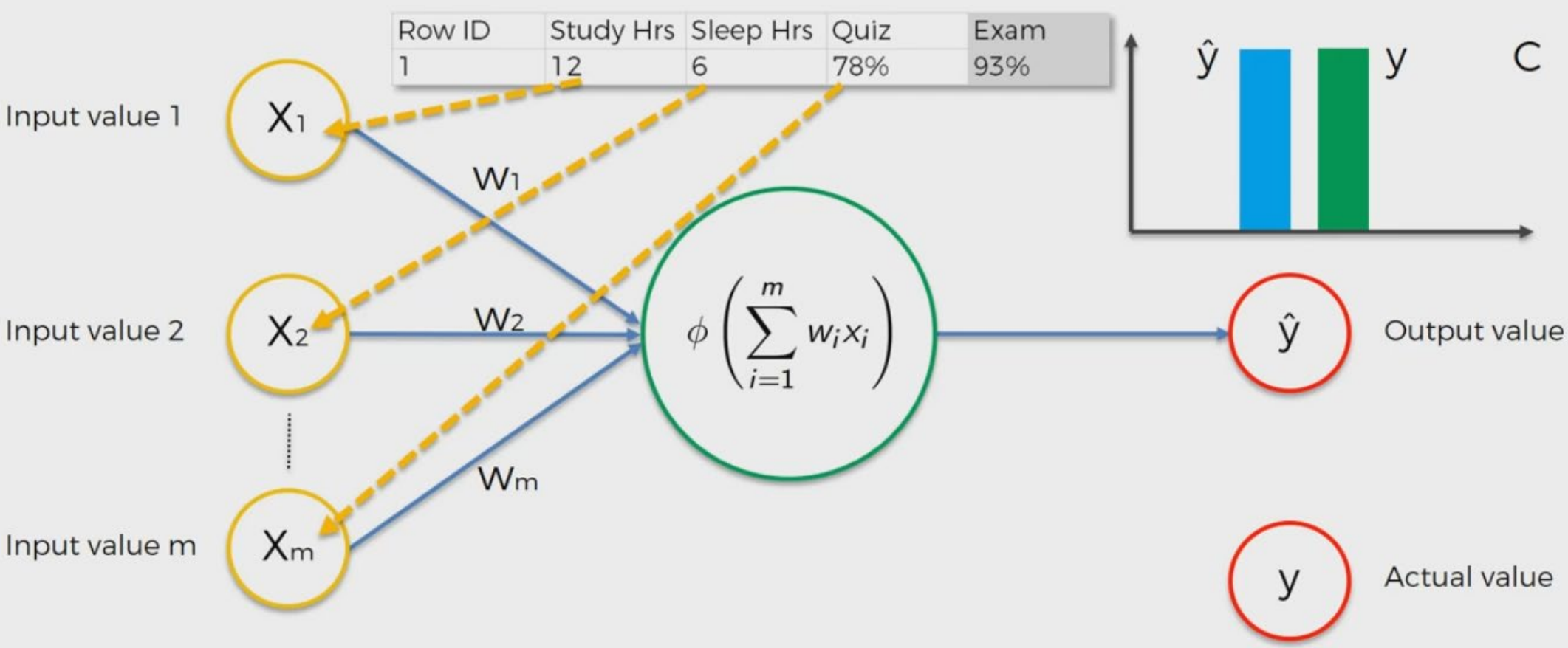

Input value 1
X1
W1
Input value 2
X2
Input value m
Xm
Wm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
Output value
y
Actual value

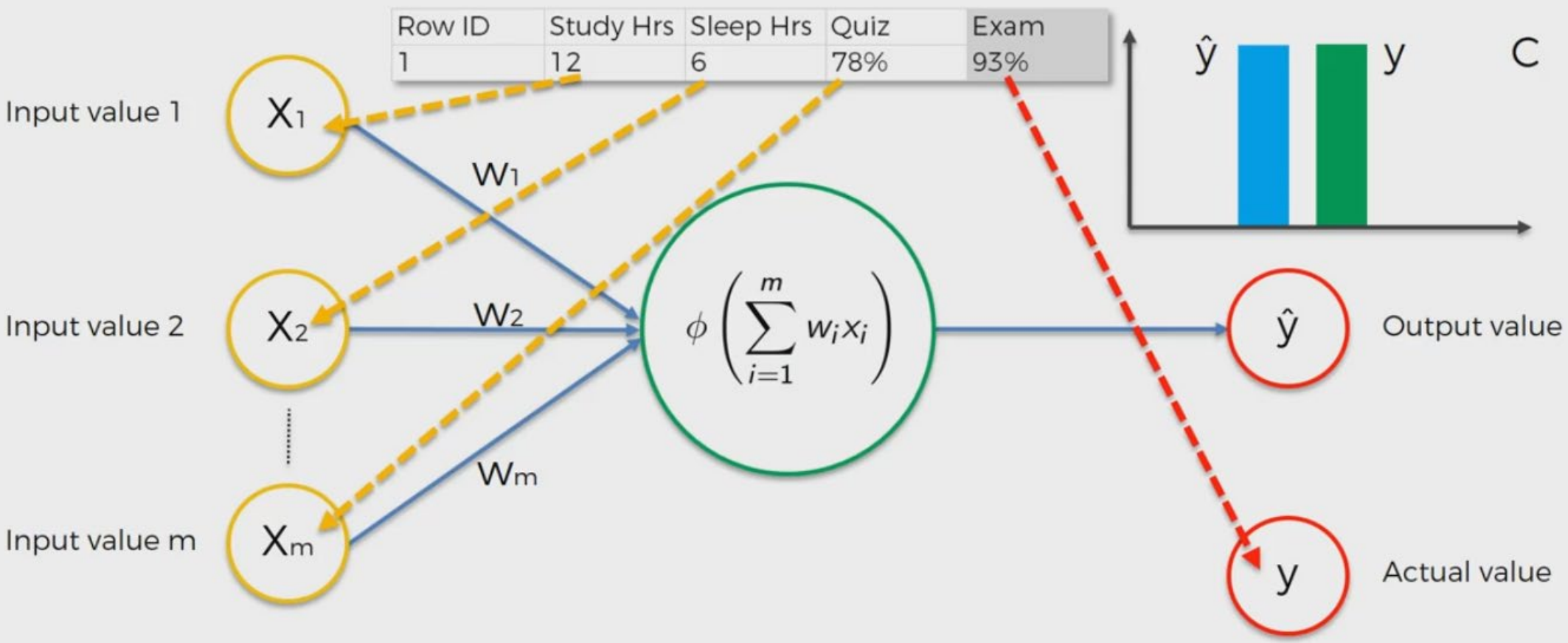

Input value 1
Input value 2
Input value m
X1
X2
Xm
W1
Wm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
y
ŷ
Output value
y
Actual value
C = ½(ŷ- y)²

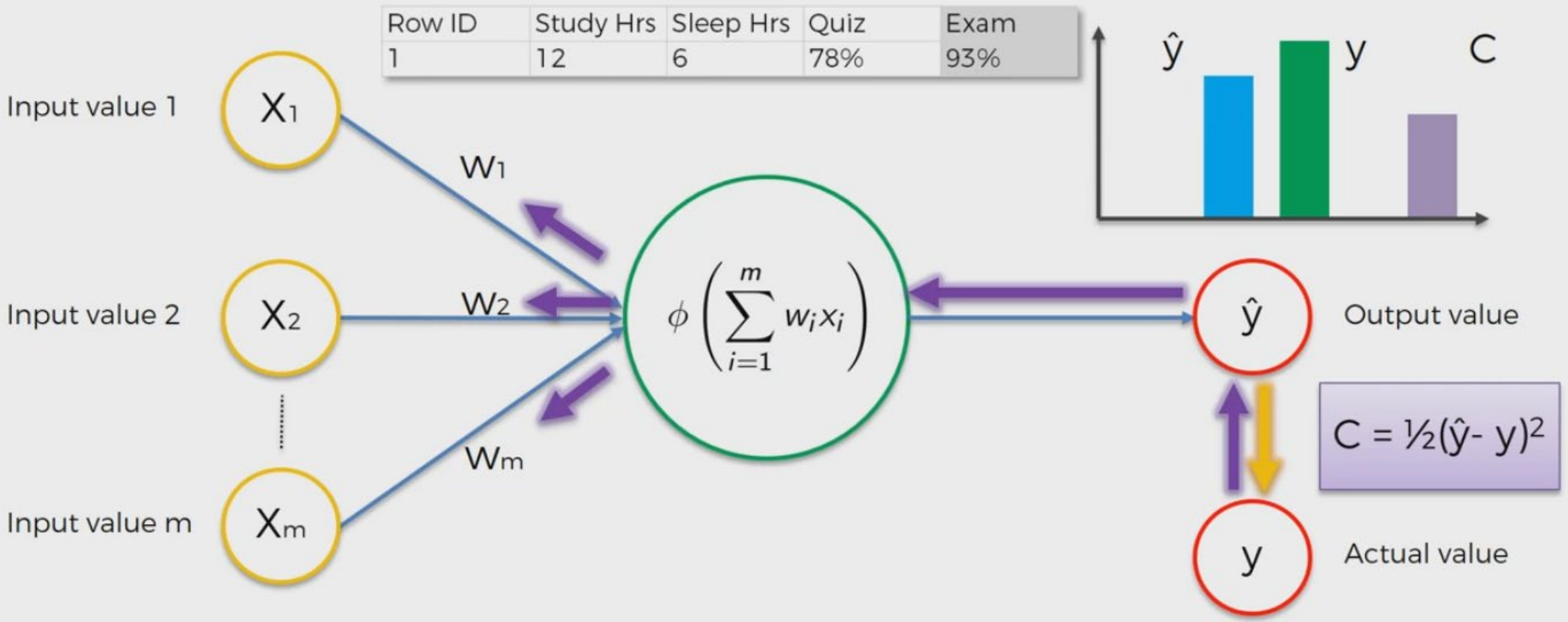

Input value 1
X1
W1
Input value 2
X2
W2
Input value m
Xm
Wm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
y
C
ŷ
Output value
C = ½(ŷ- y)²
y
Actual value

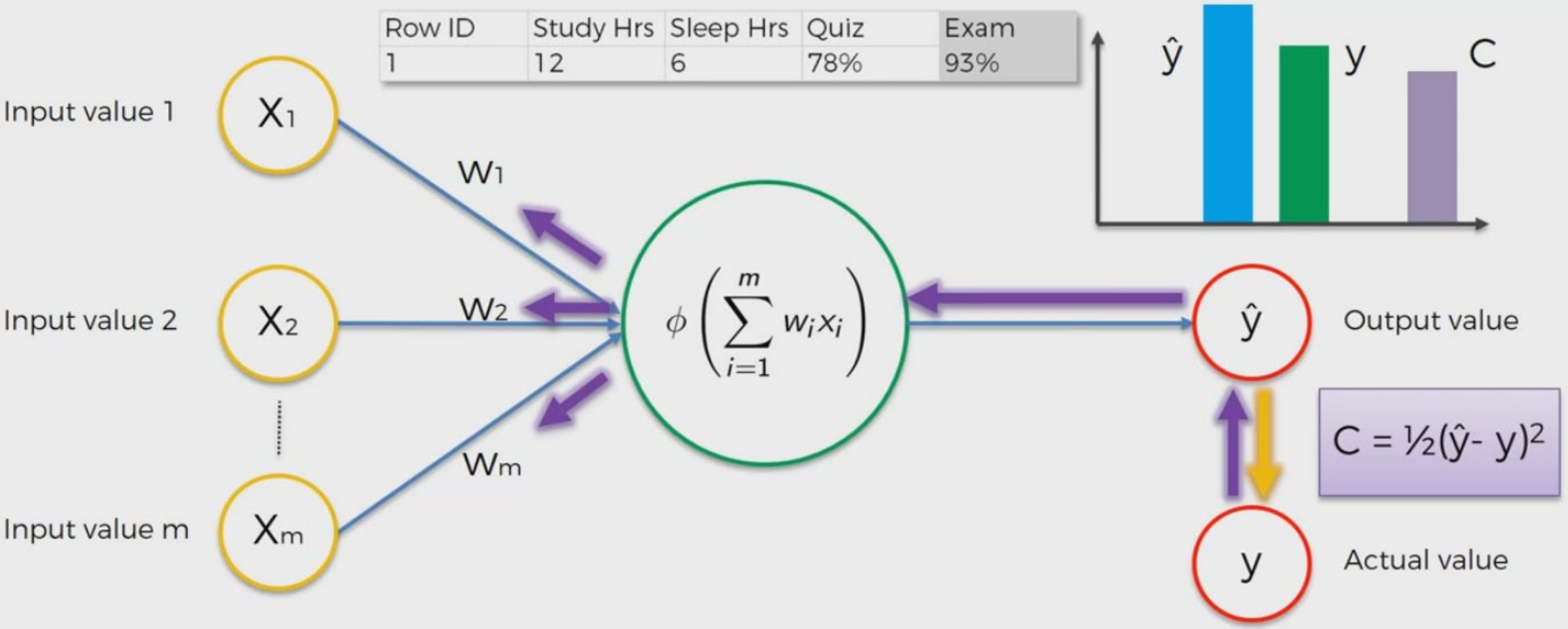

Input value 1
Input value 2
Input value m
X1
X2
Xm
W1
W2
Wm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
y
C
ŷ
Output value
y
Actual value
C = ½(ŷ- y)²

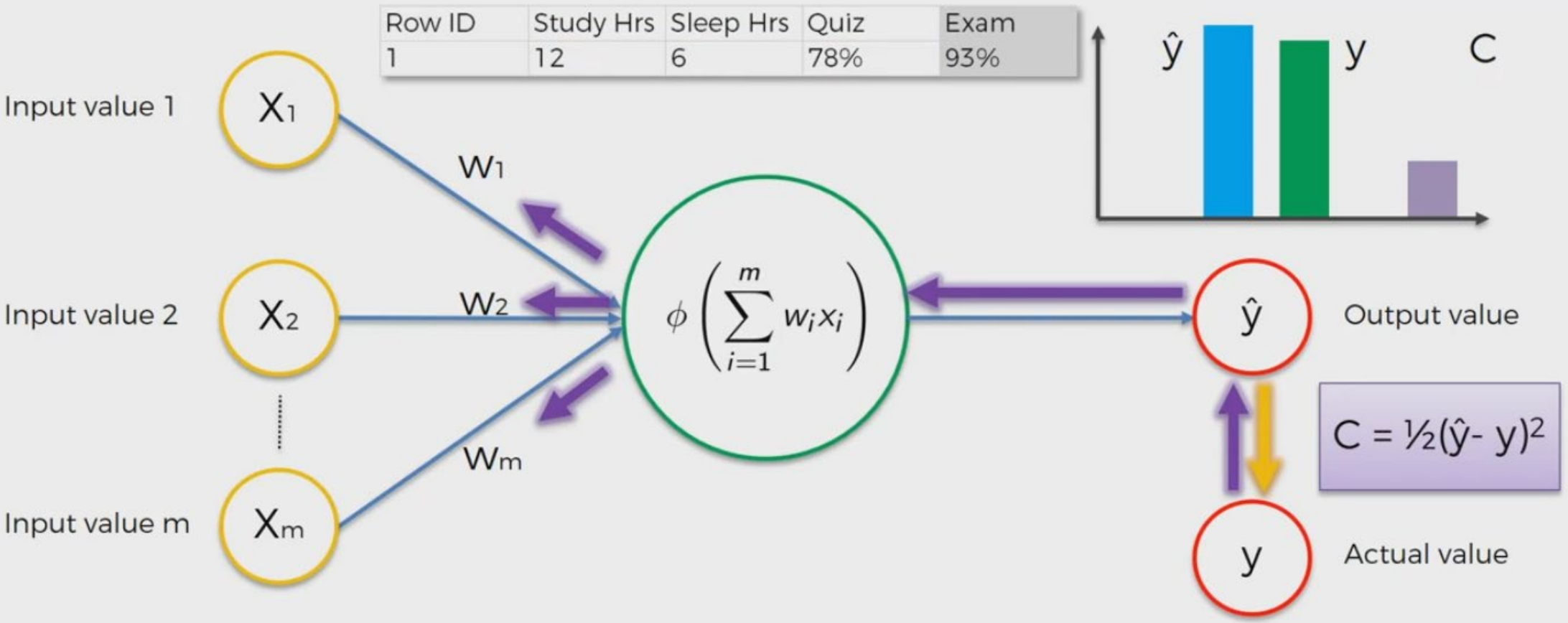

Row ID | Study Hrs | Sleep Hrs | Quiz | Exam
1 | 12 | 6 | 78% | 93%
ŷ
y
C
Input value 1
X1
W1
Input value 2
X2
W2
Input value m
Xm
Wm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
Output value
y
Actual value

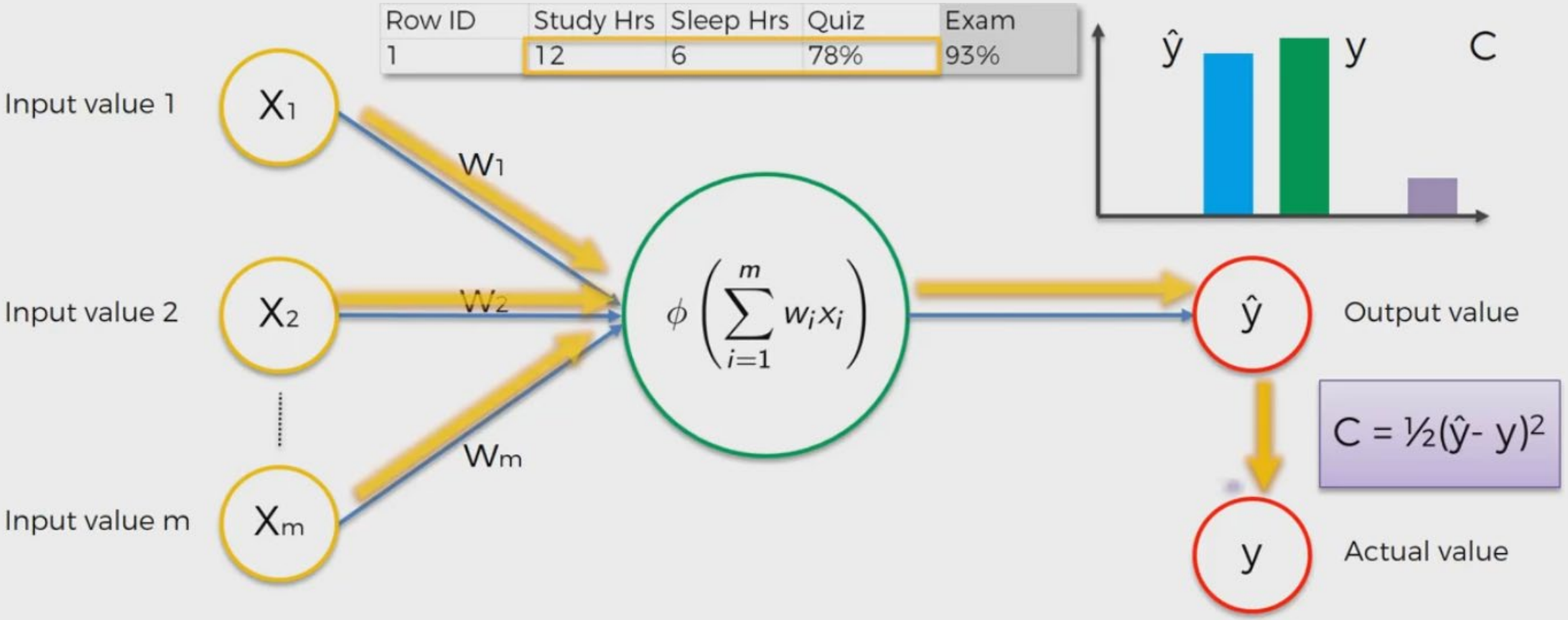

Row ID | Study Hrs | Sleep Hrs | Quiz | Exam
1 | 12 | 6 | 78% | 93%
ŷ
y
C
Input value 1
X1
W1
Input value 2
X2
W2
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
Output value
Wm
Input value m
Xm
y
Actual value

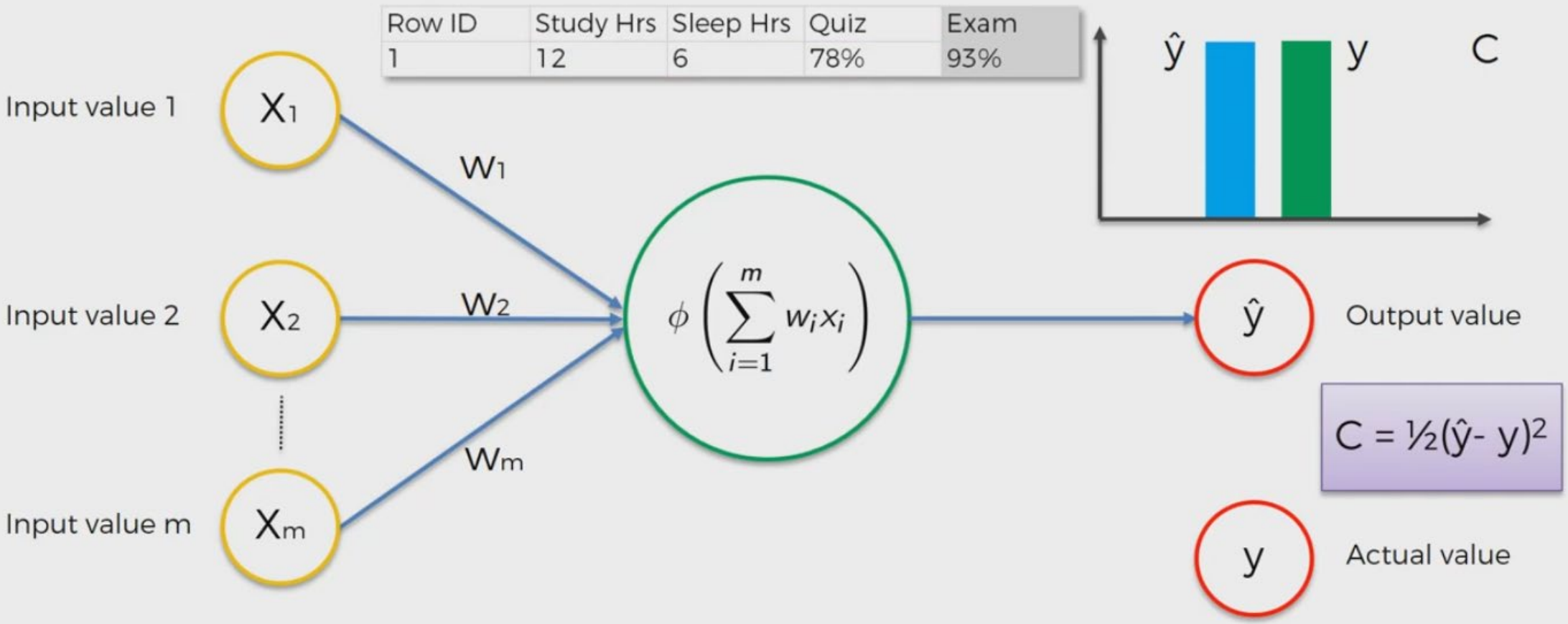

Row ID | Study Hrs | Sleep Hrs | Quiz | Exam
1 | 12 | 6 | 78% | 93%
ŷ
y
C
Input value 1
Input value 2
Input value m
X1
X2
Xm
W1
W2
Wm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
Output value
y
Actual value

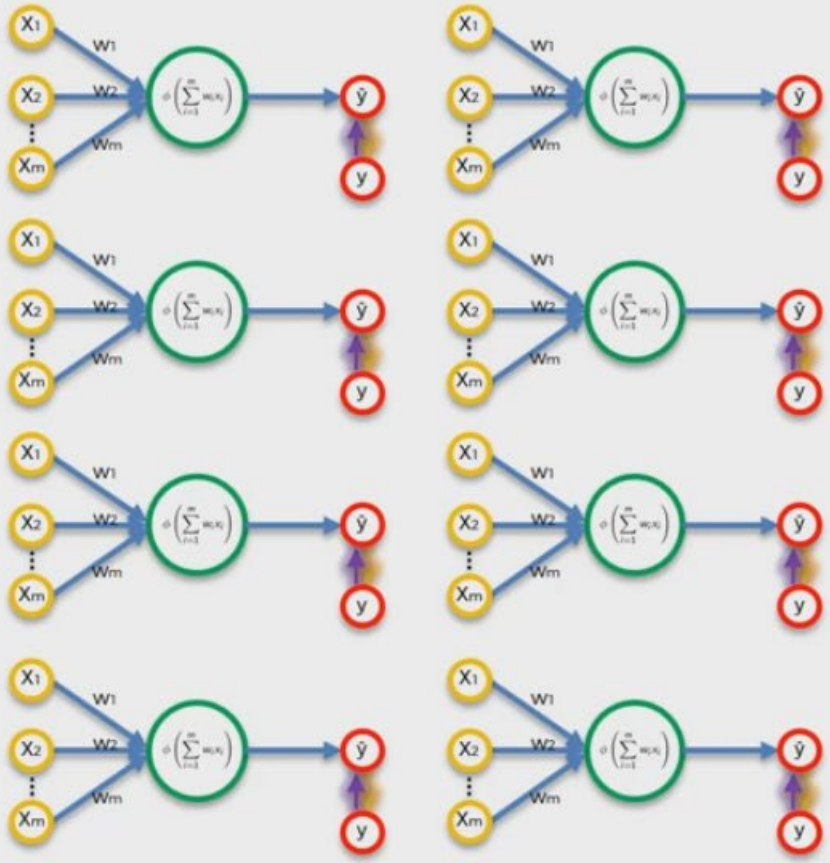

Row ID | Study Hrs | Sleep Hrs | Quiz | Exam
1 | 12 | 6 | 78% | 93%
ŷ
y
C
Input value 1
X1
W1
Input value 2
X2
W2
Input value m
Xm
Wm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
Output value
C = ½(ŷ- y)²
y
Actual value


Row ID | Study Hrs | Sleep Hrs | Quiz | Exam
1 | 12 | 6 | 78% | 93%
ŷ
y
C
Input value 1
X1
W1
Input value 2
X2
W2
Input value m
Xm
Wm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
Output value
C = ½(ŷ- y)²
y
Actual value

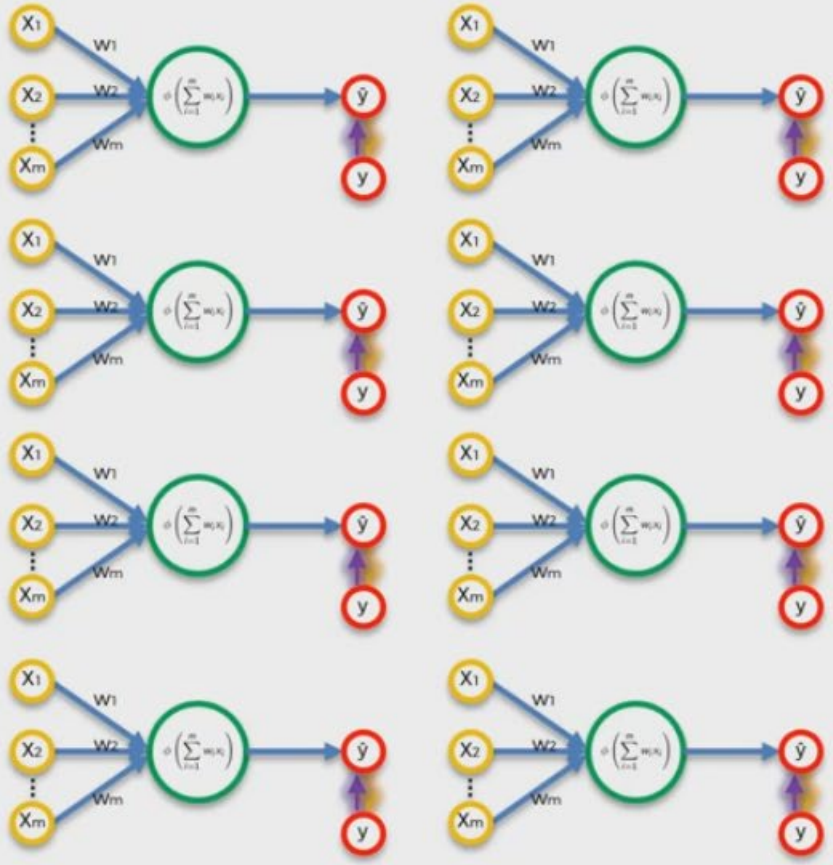

Row ID | Study Hrs | Sleep Hrs | Quiz | Exam
1 | 12 | 6 | 78% | 93%
ŷ
y
C
Input value 1
X1
W1
Input value 2
X2
W2
Input value m
Xm
Wm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
Output value
C = ½(ŷ- y)²
y
Actual value


Row ID | Study Hrs | Sleep Hrs | Quiz | Exam
1 | 12 | 6 | 78% | 93%
ŷ
y
C
Input value 1
X1
W1
Input value 2
X2
W2
Input value m
Xm
Wm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
Output value
C = ½(ŷ- y)²
y
Actual value


Row ID | Study Hrs | Sleep Hrs | Quiz | Exam
1 | 12 | 6 | 78% | 93%
ŷ
y
C
Input value 1
X1
W1
Input value 2
X2
W2
Input value m
Xm
Wm
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
ŷ
Output value
C = ½(ŷ- y)²
y
Actual value

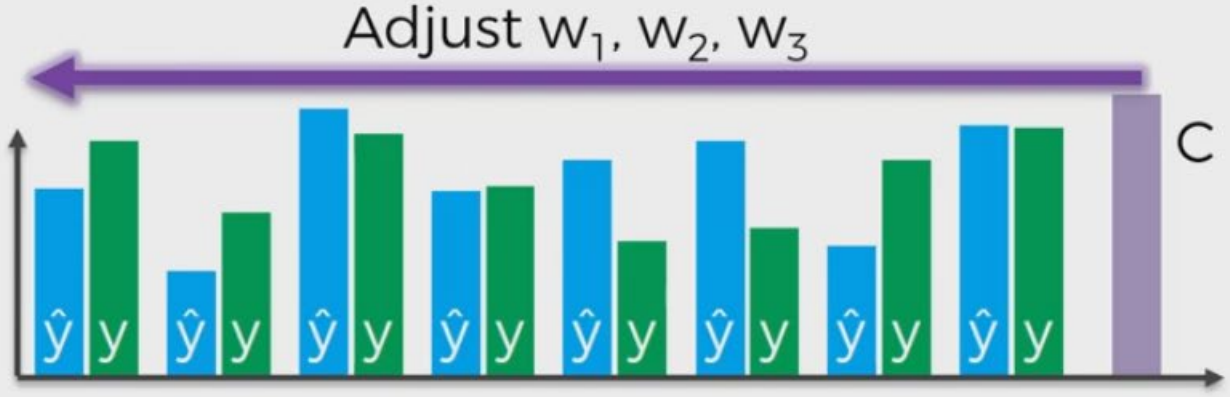


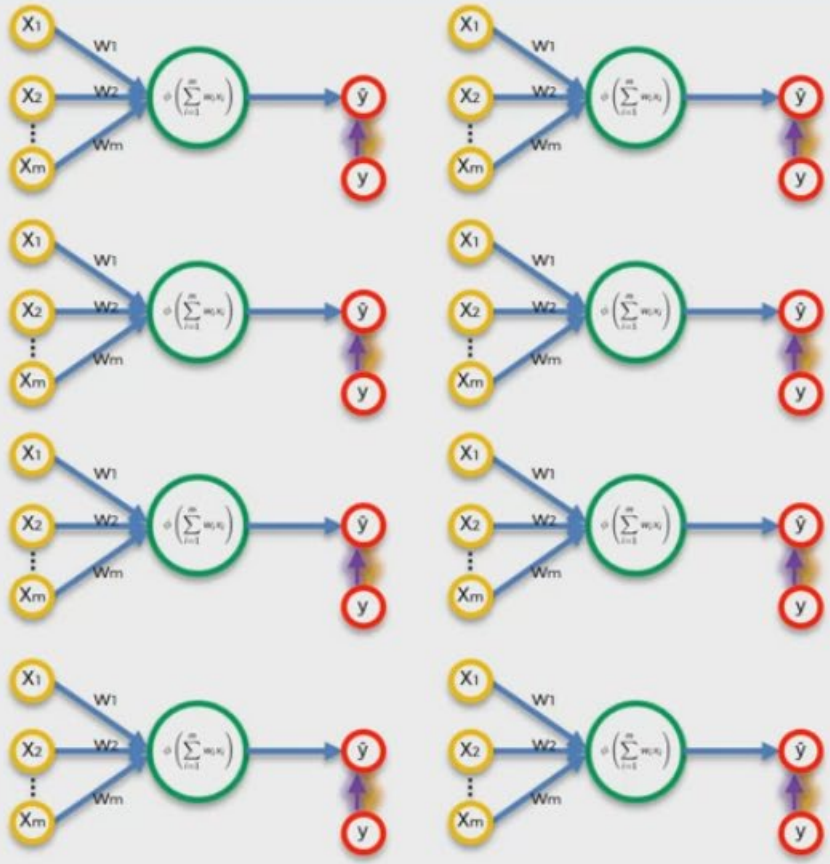
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%



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

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





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

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

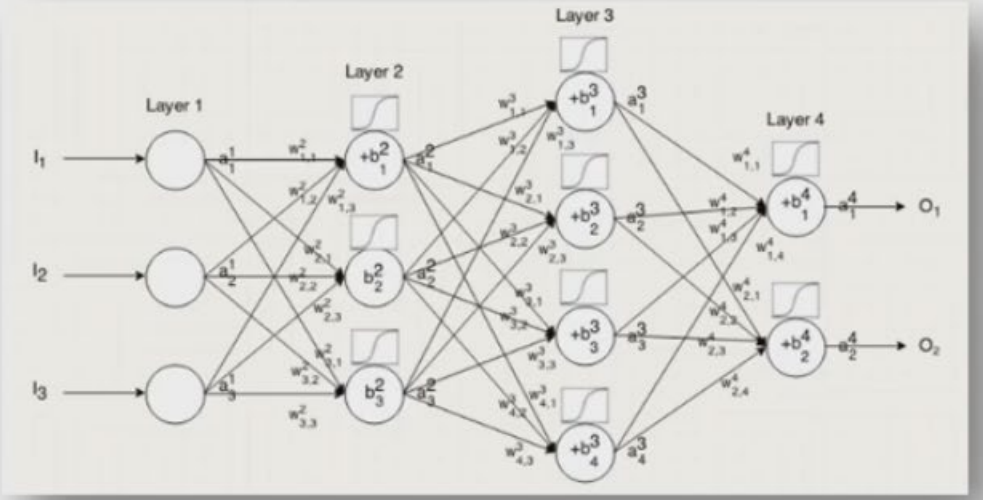
Additional Reading:

A list of cost functions used in neural networks, alongside applications

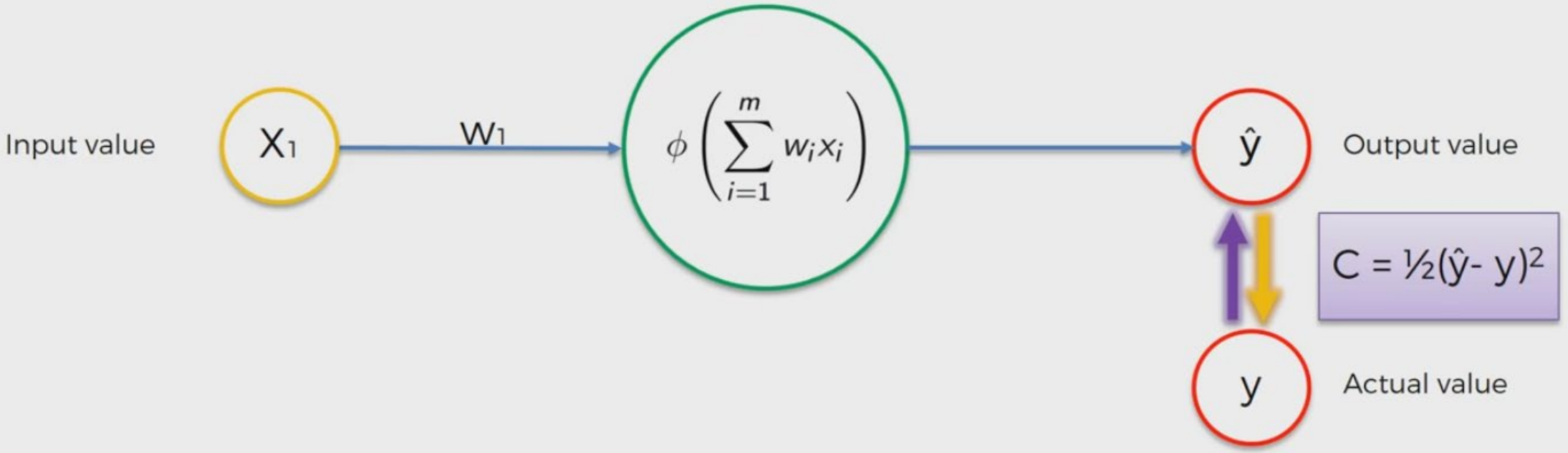
CrossValidated (2015)

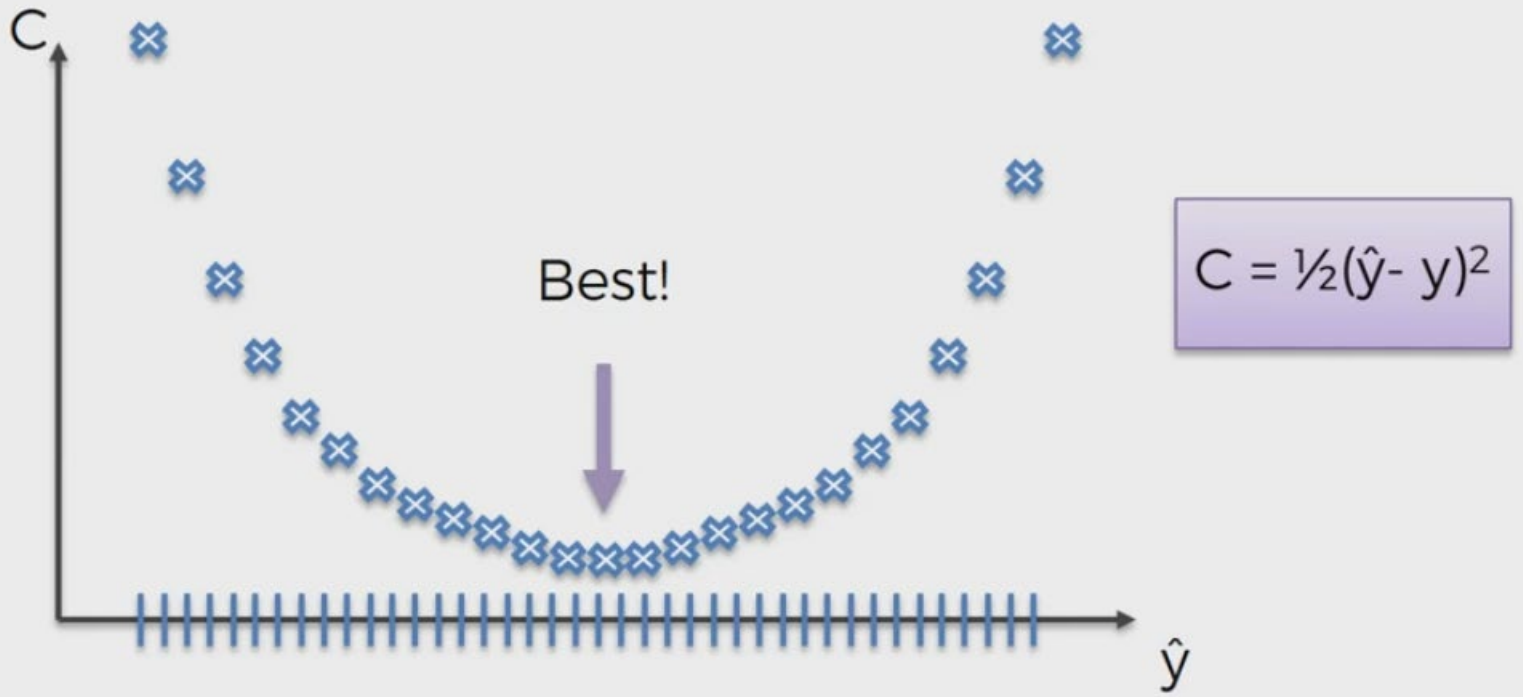
Link:

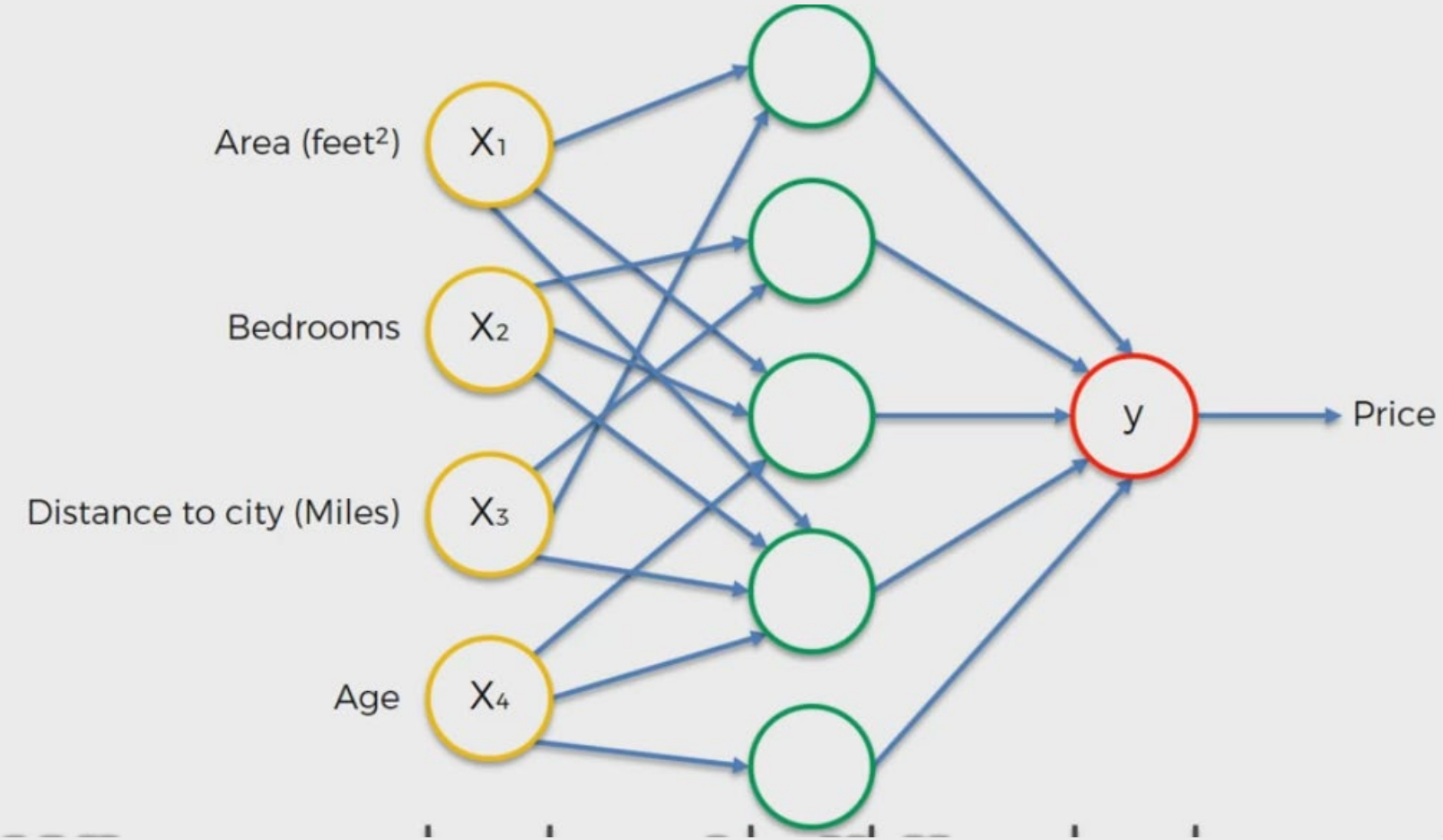
http://stats.stackexchange.com/questions/154879/a-list-of-cost-functions-used-in-neural-networks-alongside-applications

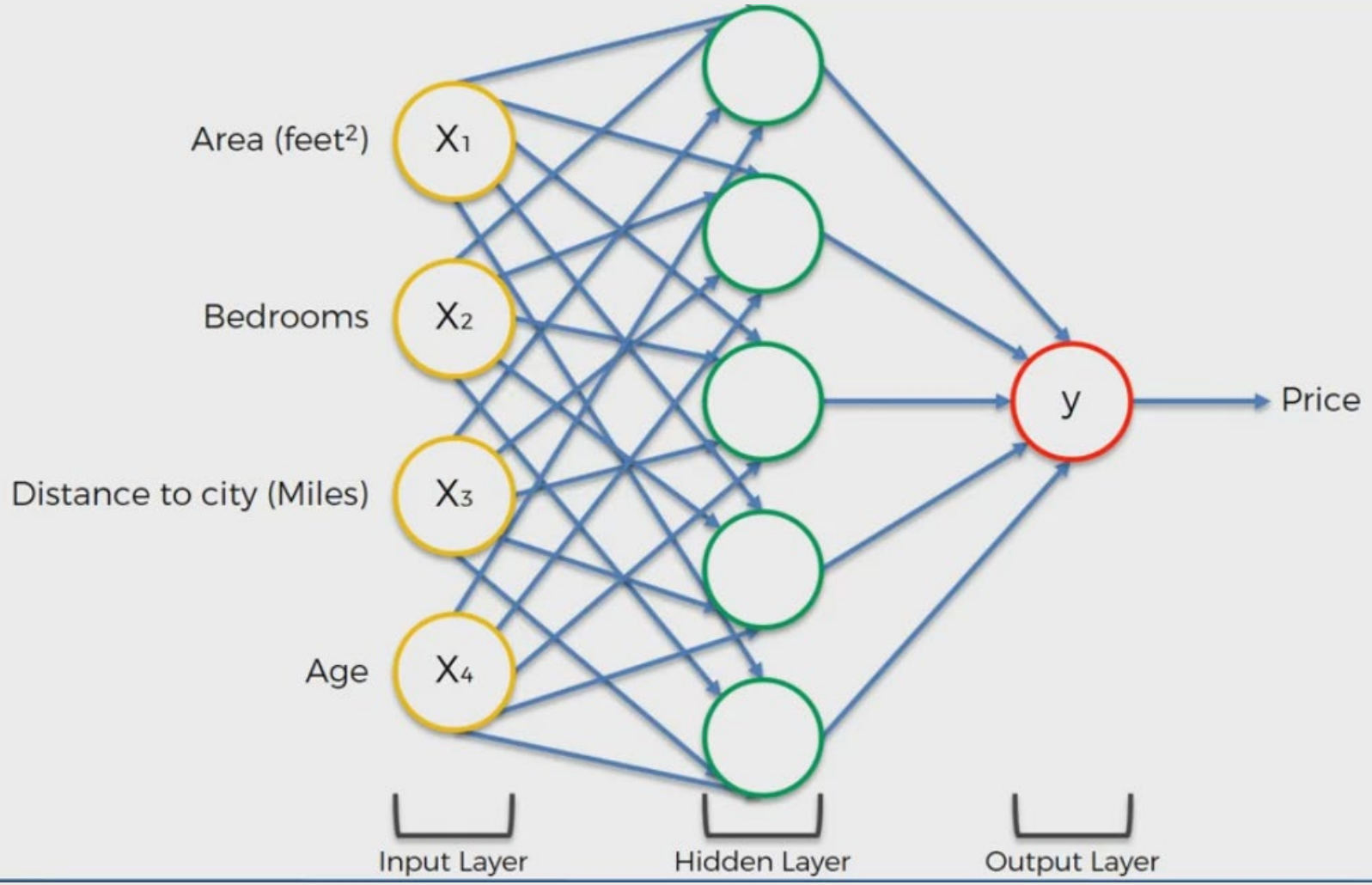


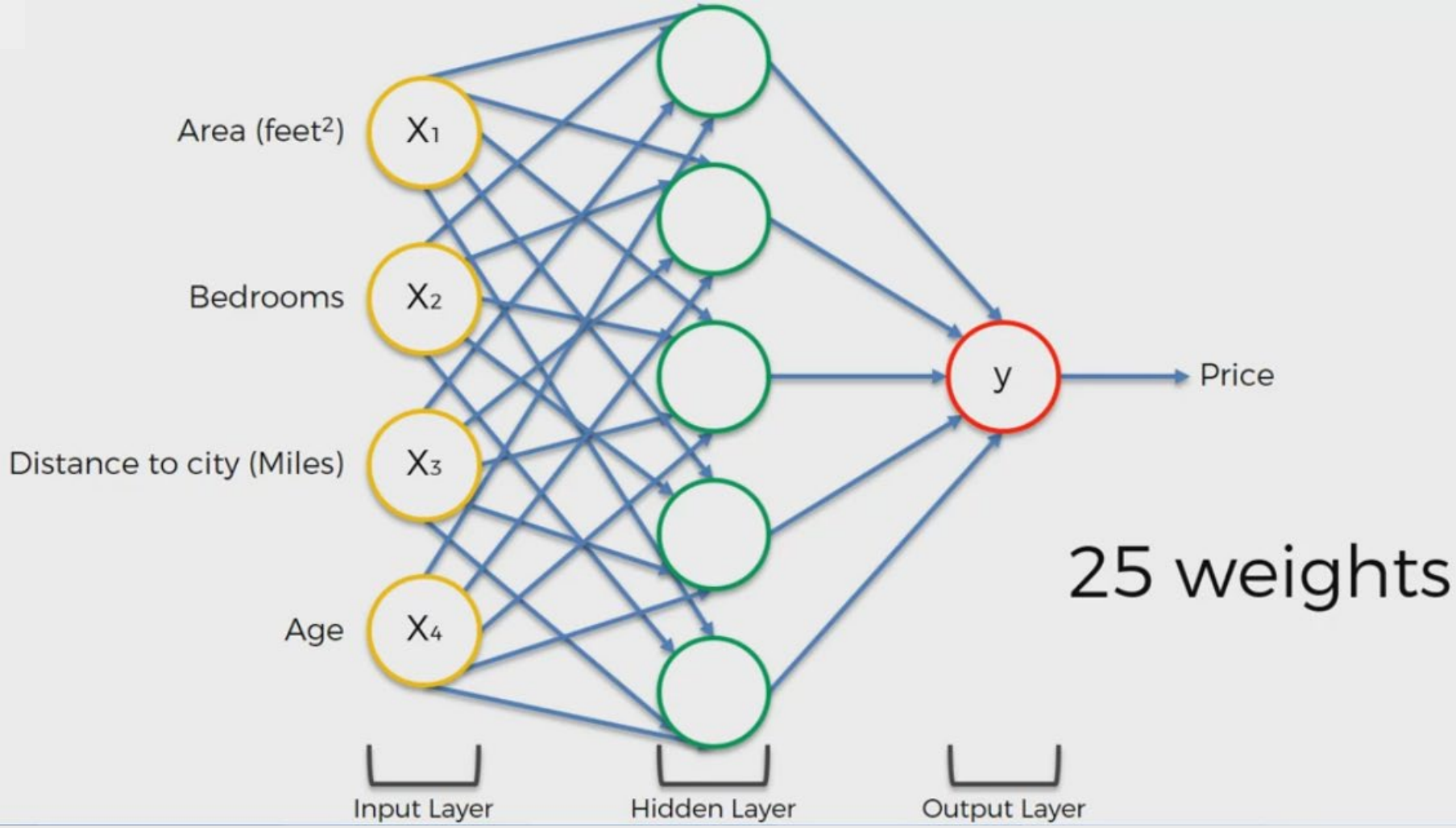
Gradient Descent


Input value
x_1
w_1
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
$\hat{y}$
Output value
$C = \frac{1}{2}(\hat{y} - y)^2$
y
Actual value


C
Best!
ŷ
C = ½(ŷ- y)²


Area (feet²)
X1
Bedrooms
X2
Distance to city (Miles)
X3
Age
X4
y
Price


Area (feet²)
X1
Bedrooms
X2
Distance to city (Miles)
X3
Age
X4
y
Price
Input Layer
Hidden Layer
Output Layer


Area (feet²)
X1
Bedrooms
X2
Distance to city (Miles)
X3
Age
X4
y
Price
25 weights
Input Layer
Hidden Layer
Output Layer

1,000 x 1,000 x … x 1,000 = $1{,}000^{25} = 10^{75}$ combinations

Sunway TaihuLight: World's fastest Super Computer

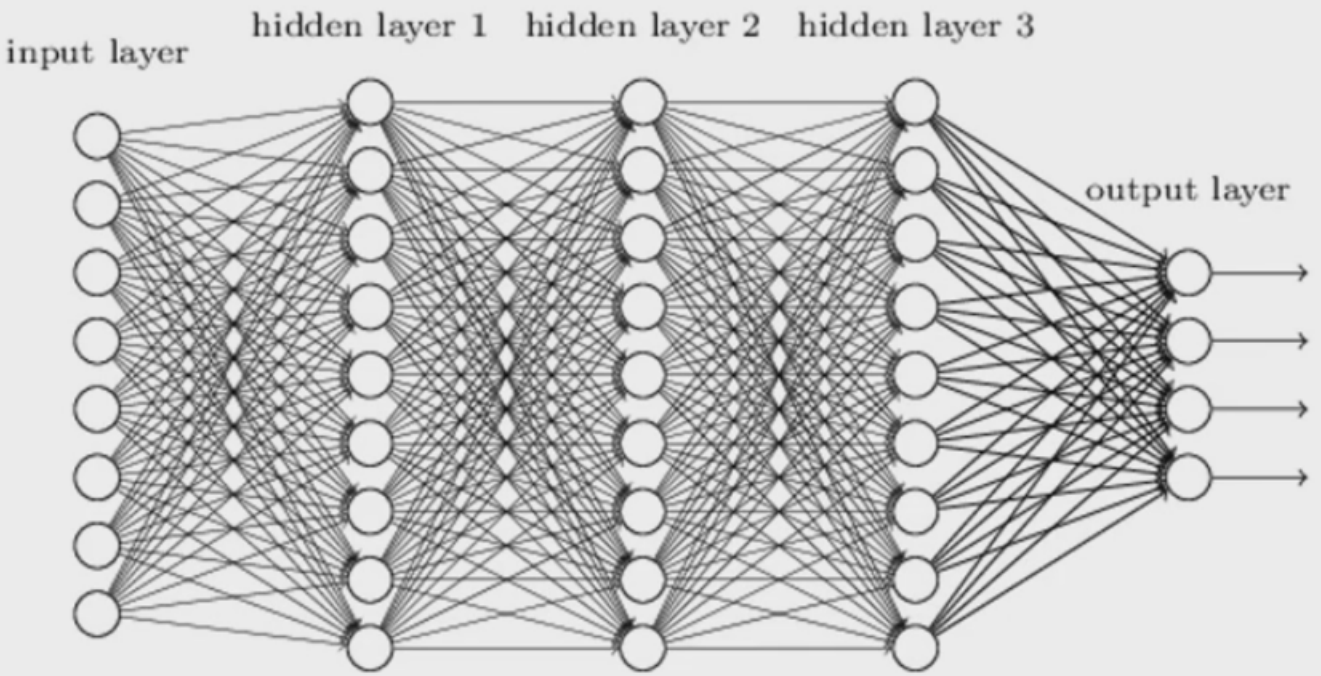
93 PFLOPS

93×10^{15}

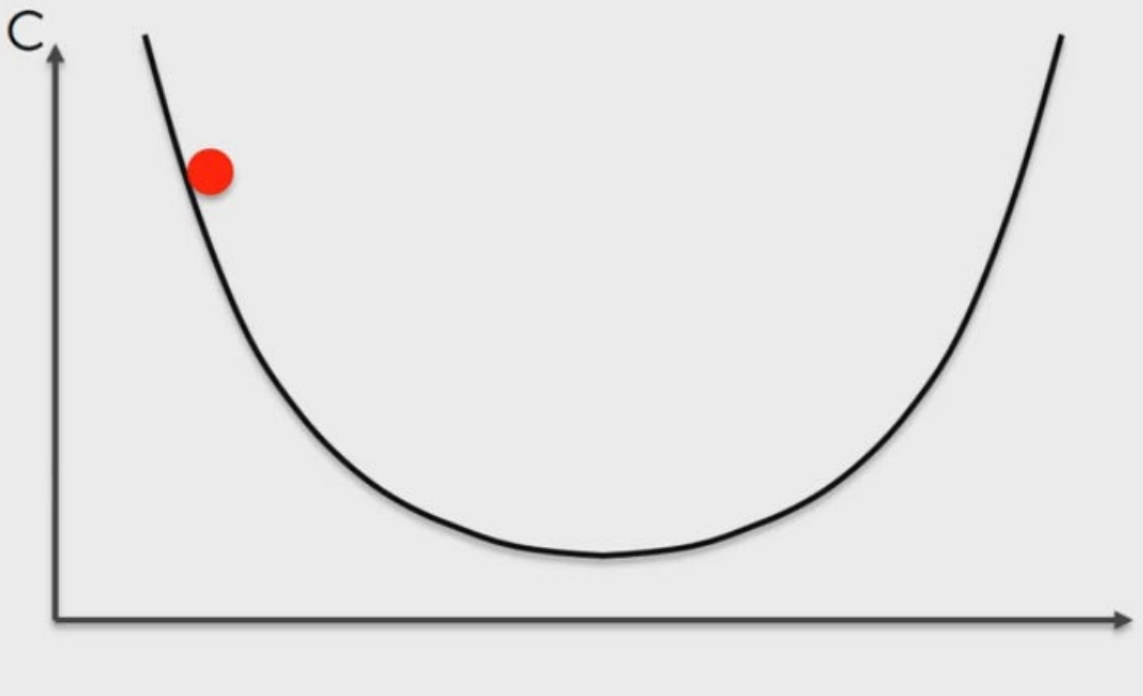
$10^{75} / (93 \times 10^{15})$

$= 1.08 \times 10^{58}$ seconds

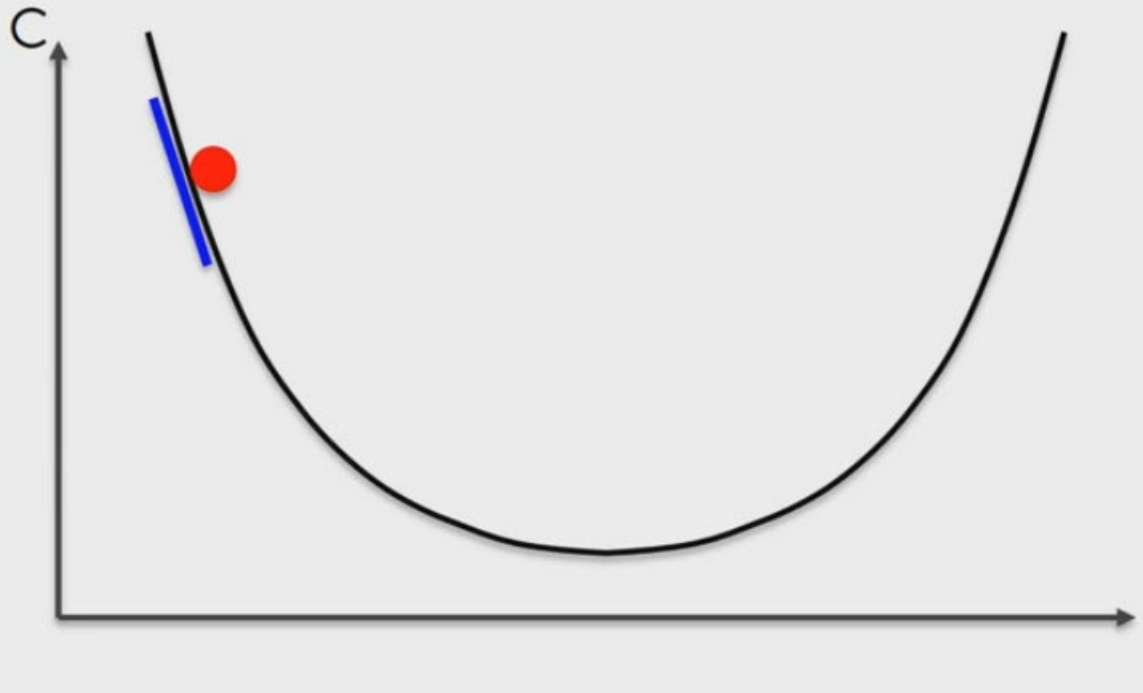
$= 3.42 \times 10^{50}$ years


input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer

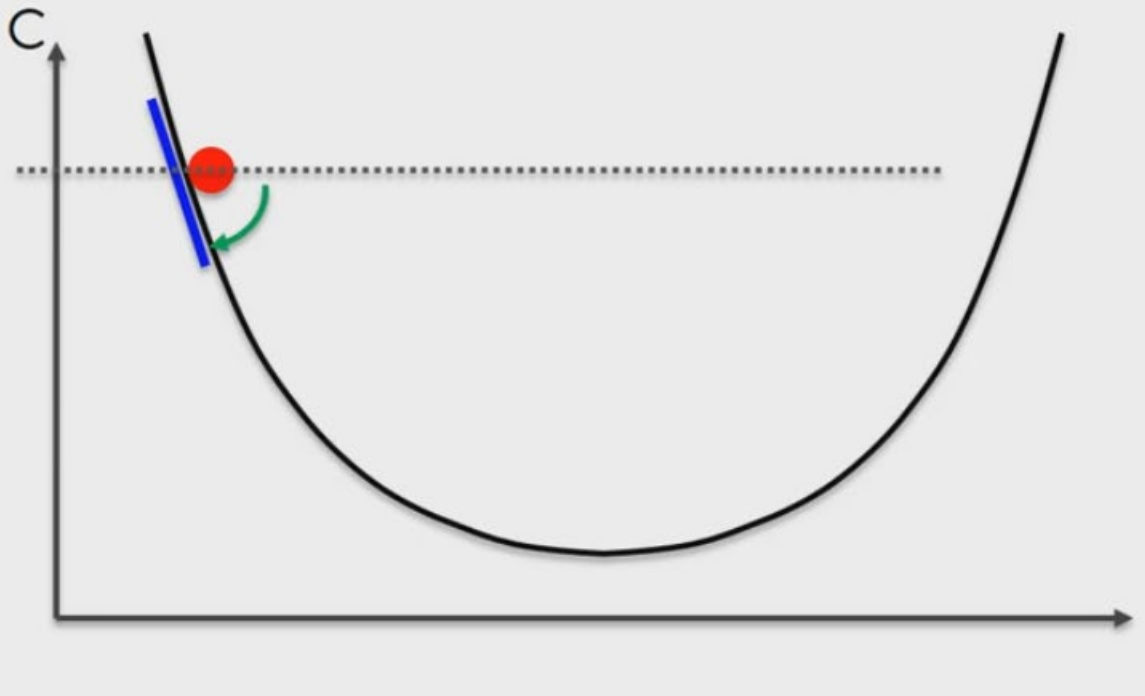
Gradient Descent



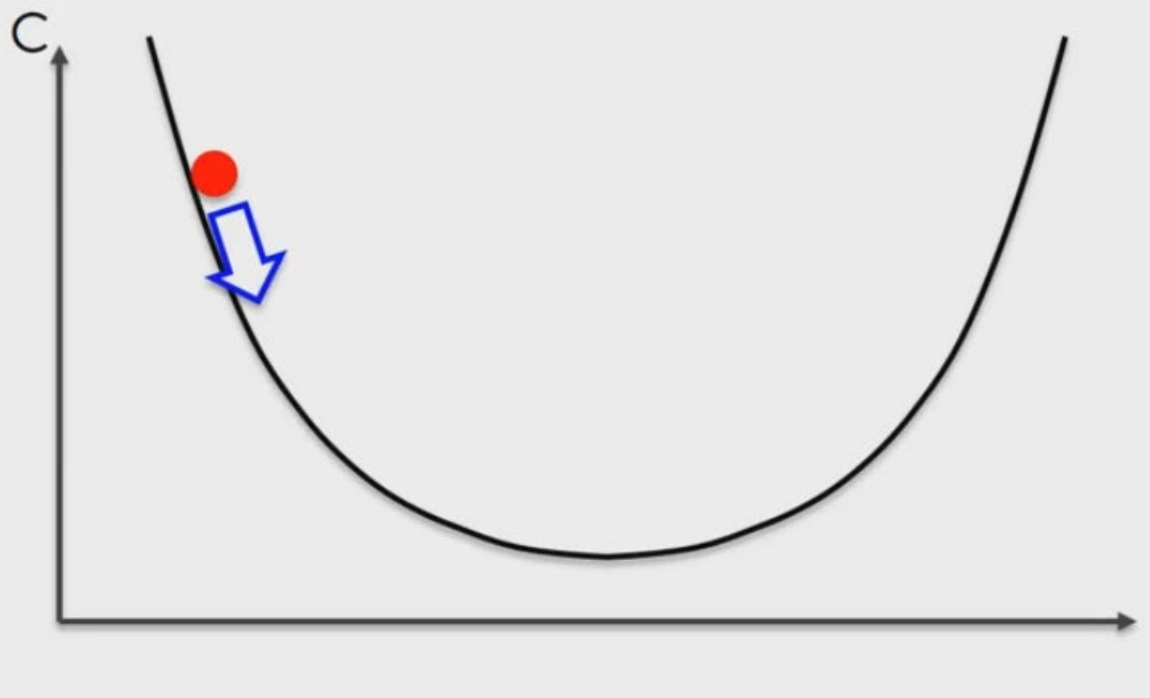
$$C = \tfrac{1}{2}(\hat{y} - y)^2$$



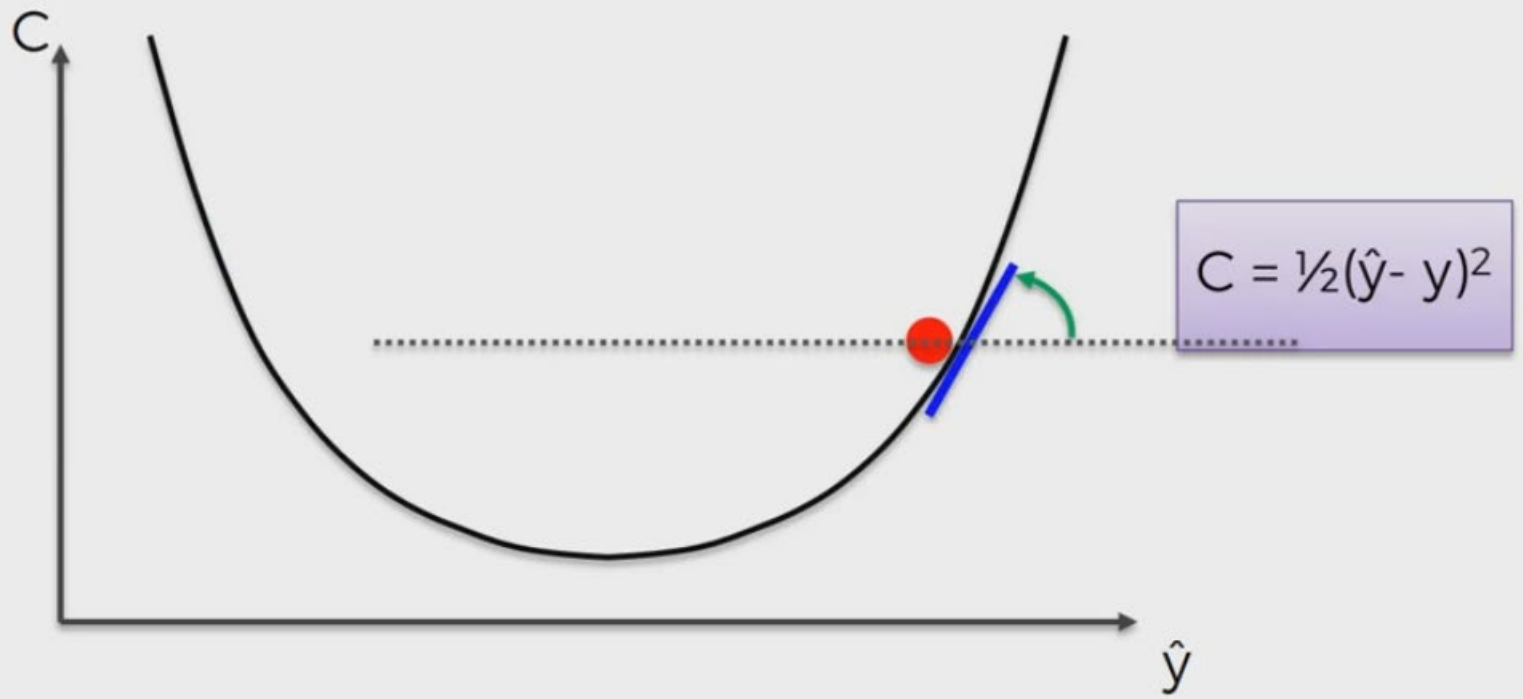
$$C = \frac{1}{2}(\hat{y} - y)^2$$

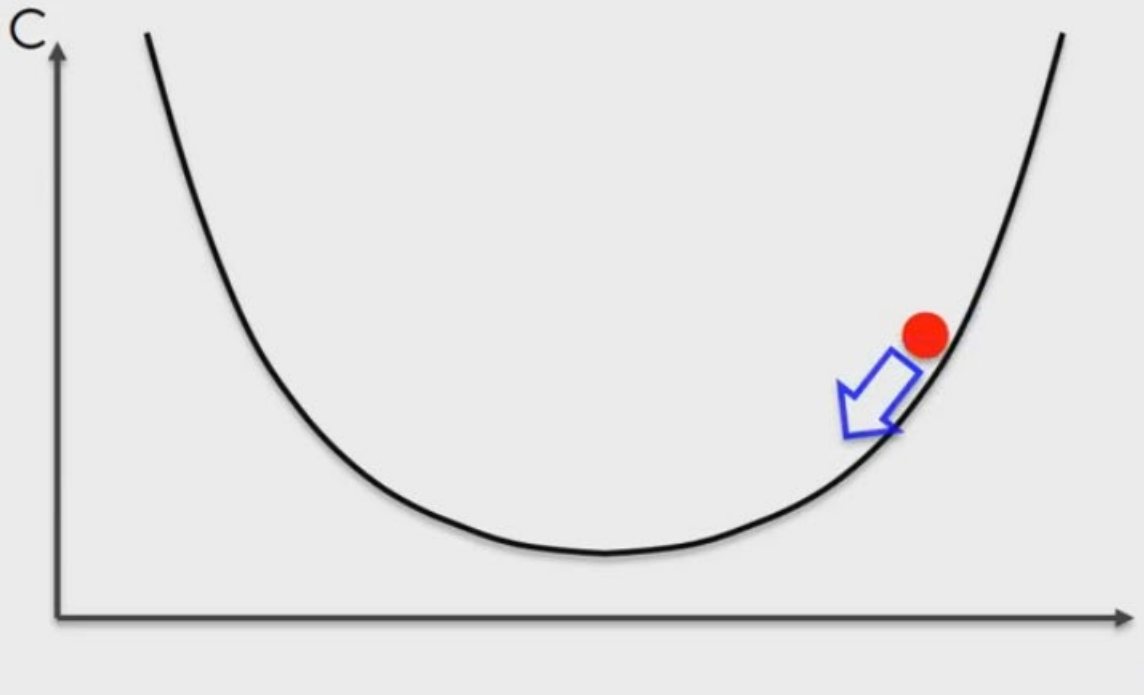


$$C = \tfrac{1}{2}(\hat{y} - y)^2$$

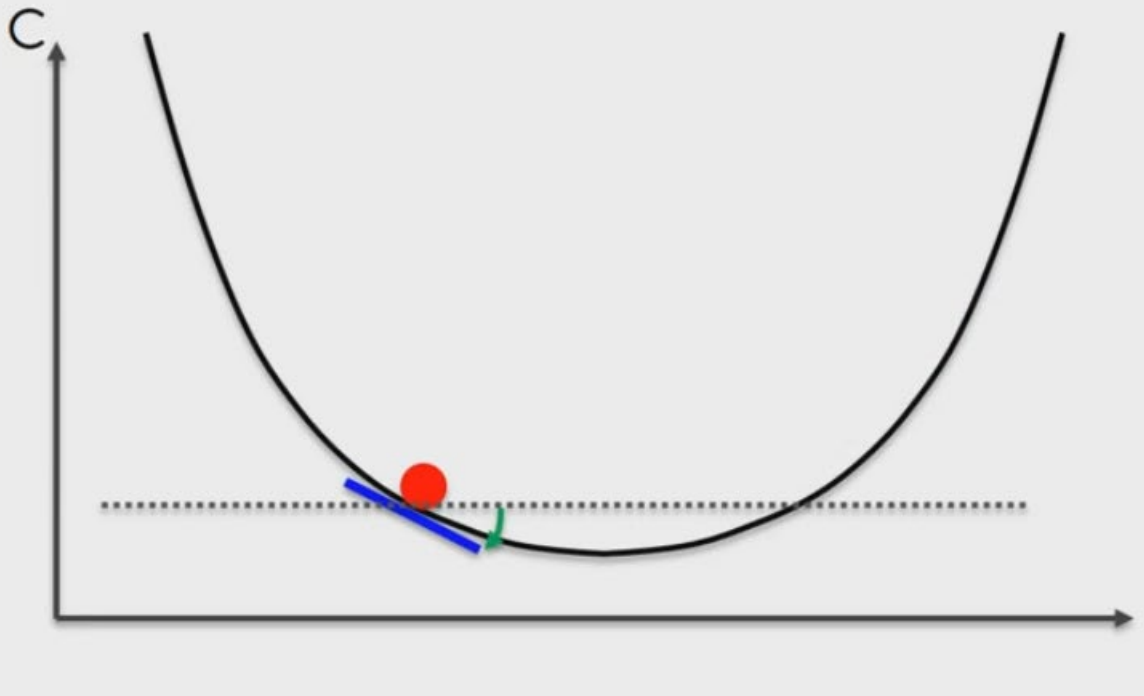


$$C = \frac{1}{2}(\hat{y} - y)^2$$

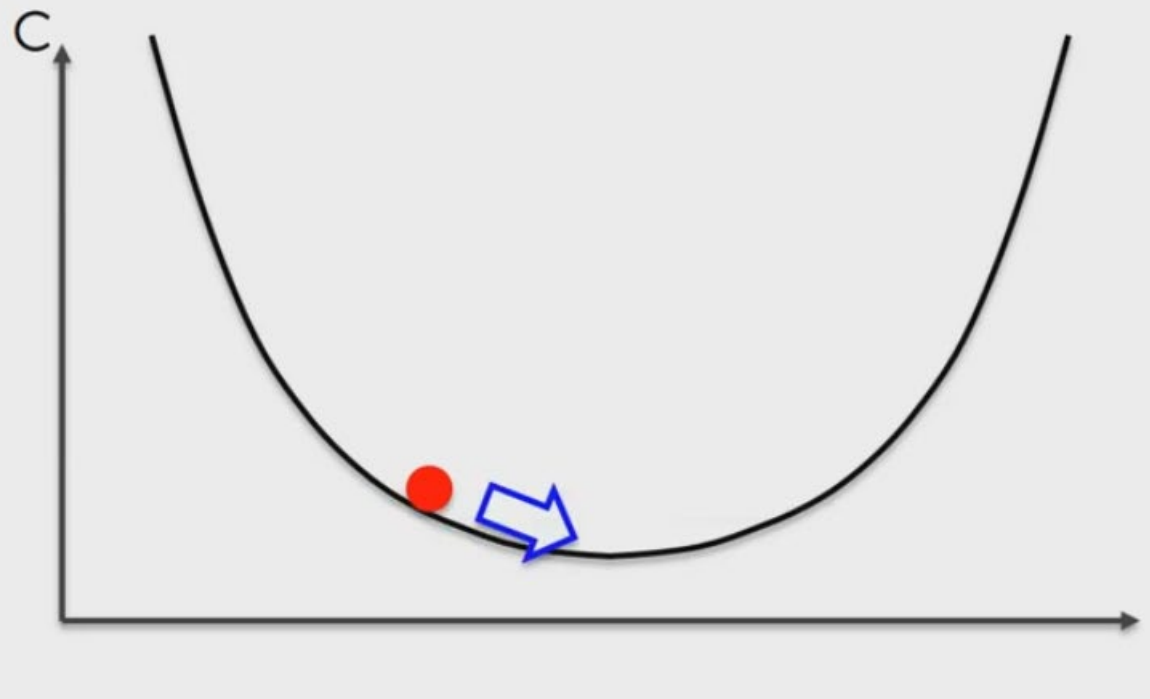

C
$C = \frac{1}{2}(\hat{y} - y)^2$
$\hat{y}$



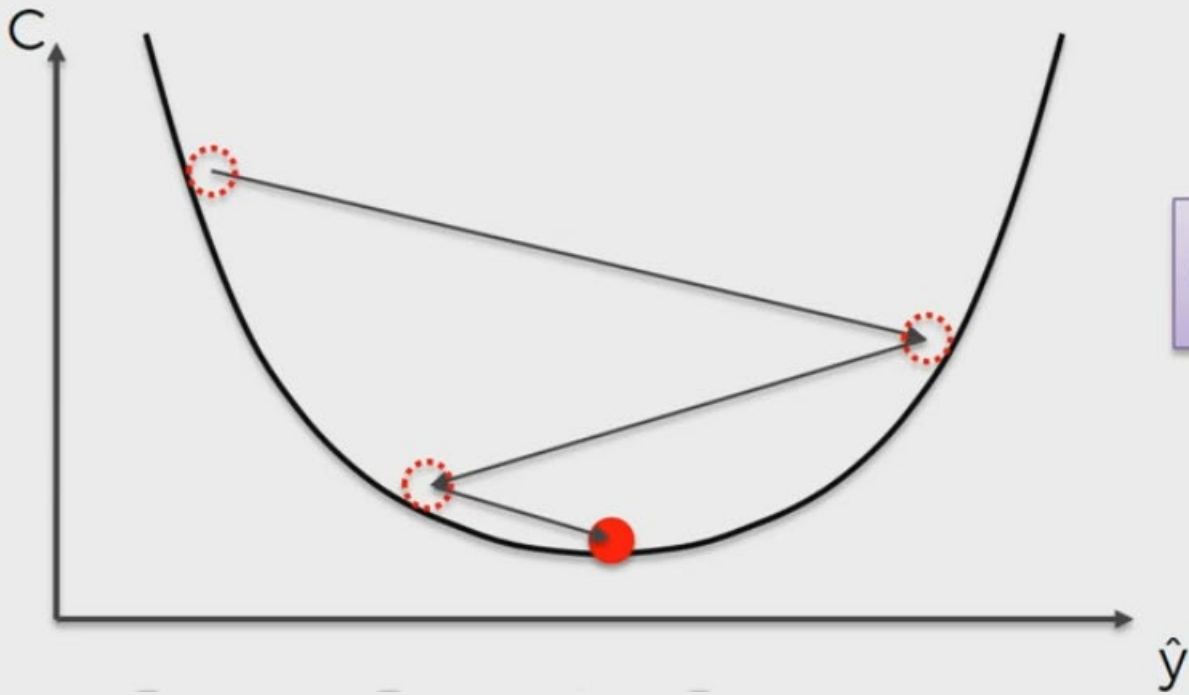
$$C = \frac{1}{2}(\hat{y} - y)^2$$



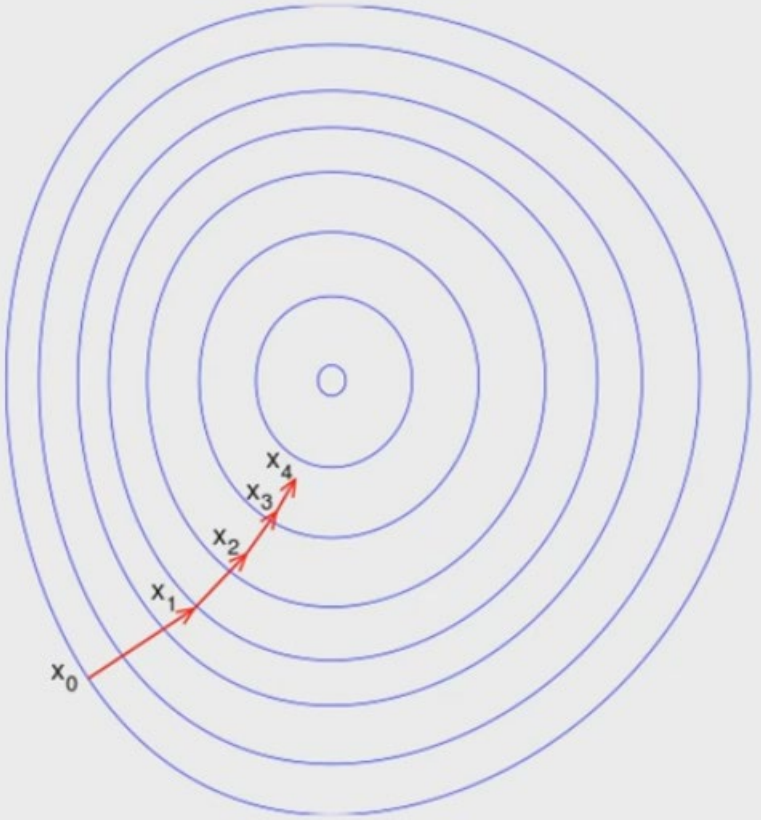
$$C = \frac{1}{2}(\hat{y} - y)^2$$

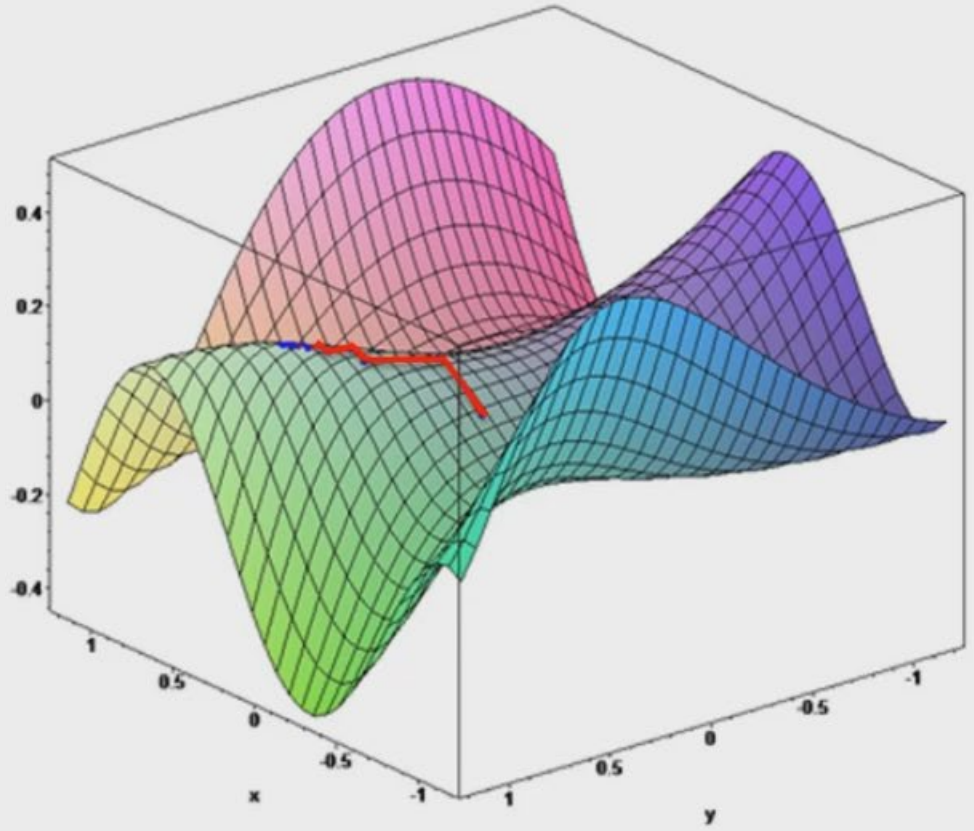


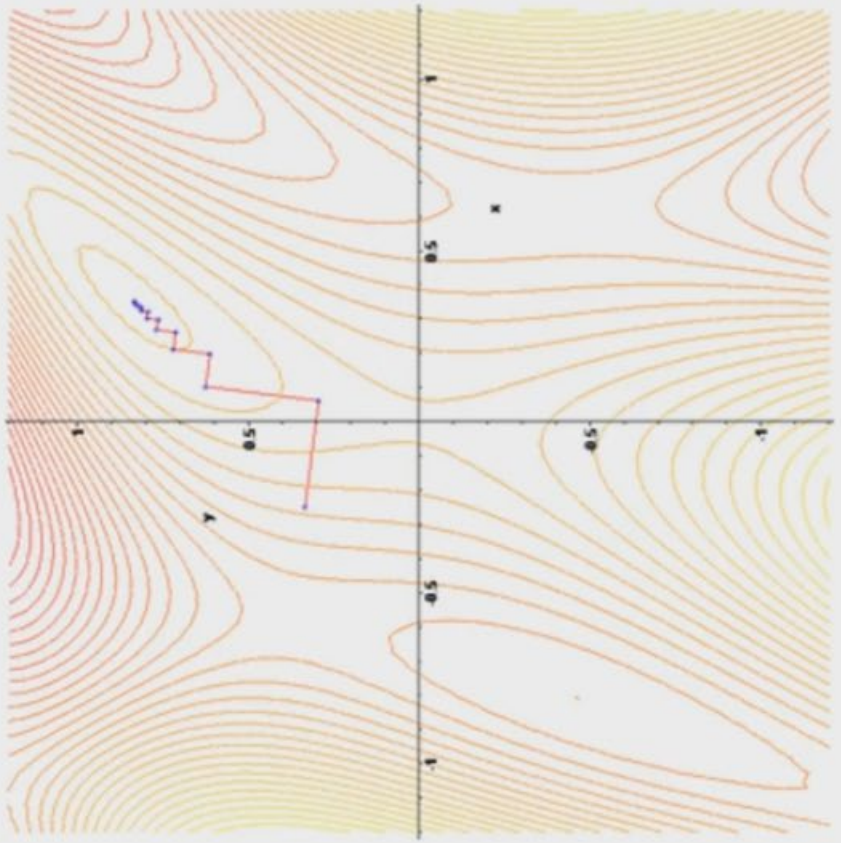
$$C = \frac{1}{2}(\hat{y} - y)^2$$



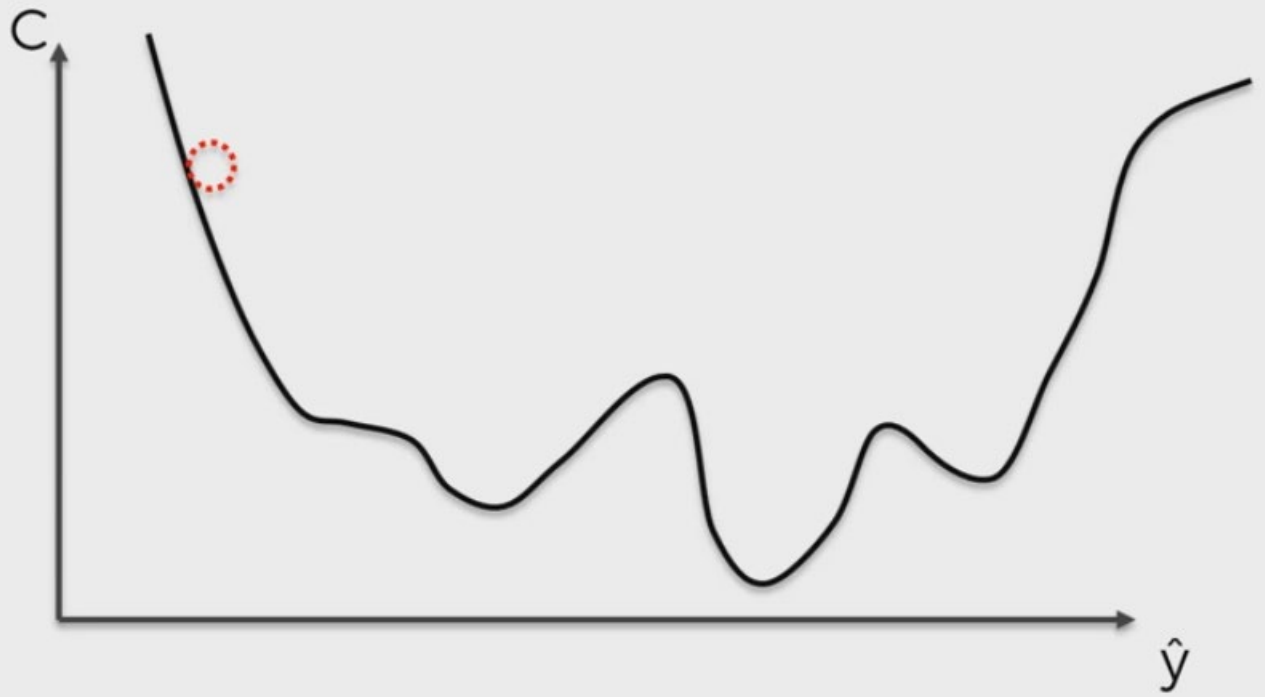
$$C = \frac{1}{2}(\hat{y} - y)^2$$

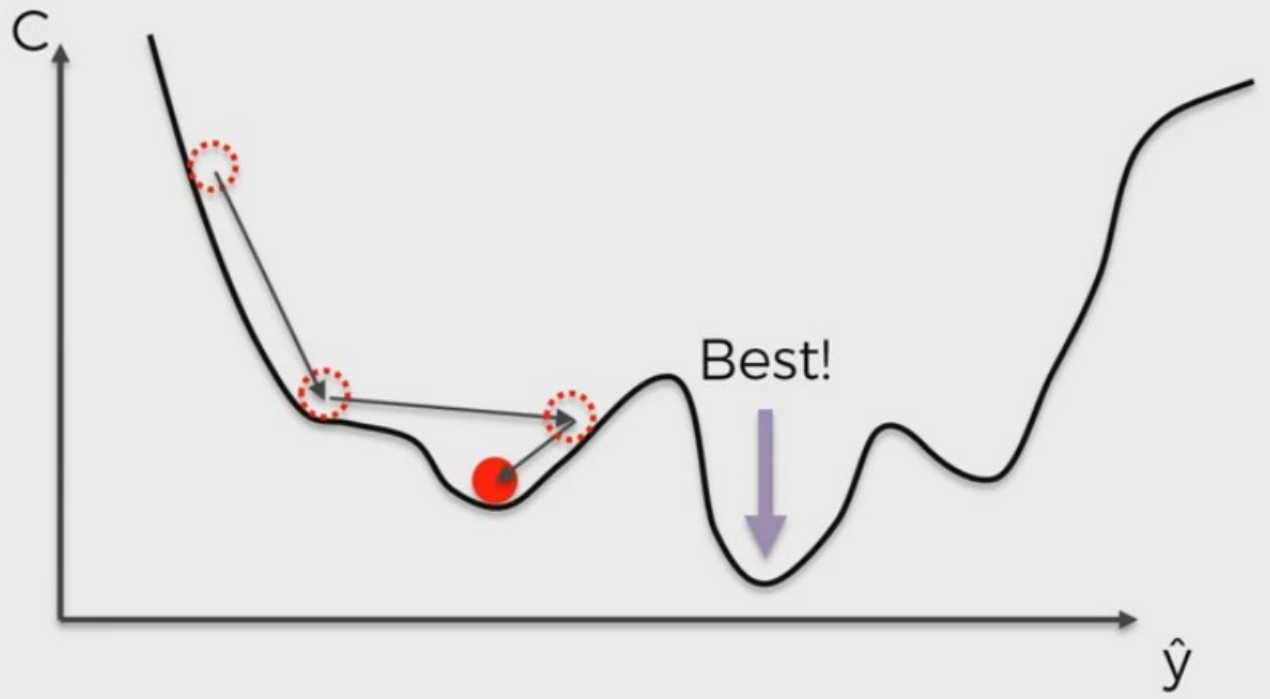

x_0
x_1
x_2
x_3
x_4

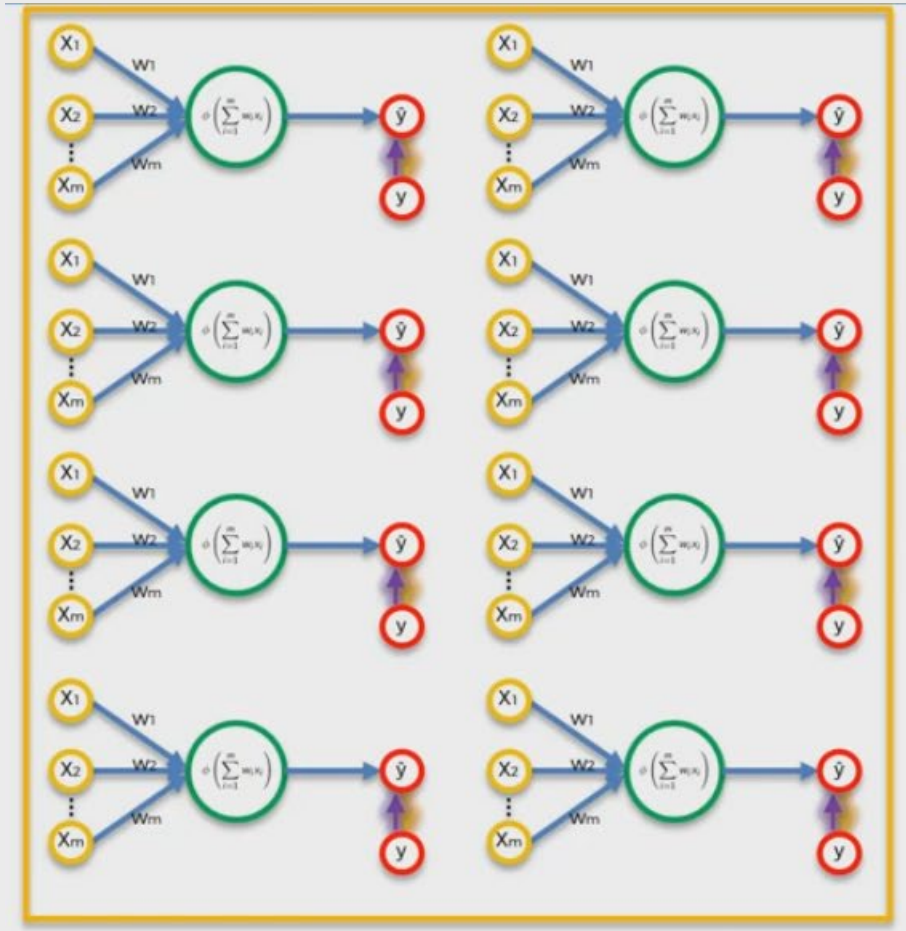

0.4
0.2
0
-0.2
-0.4
1
0.5
0
-0.5
-1
x
1
0.5
0
-0.5
-1
y


1
0.5
x
1
0.5
-0.5
-1
y
-0.5
-1

Stochastic Gradient Descent

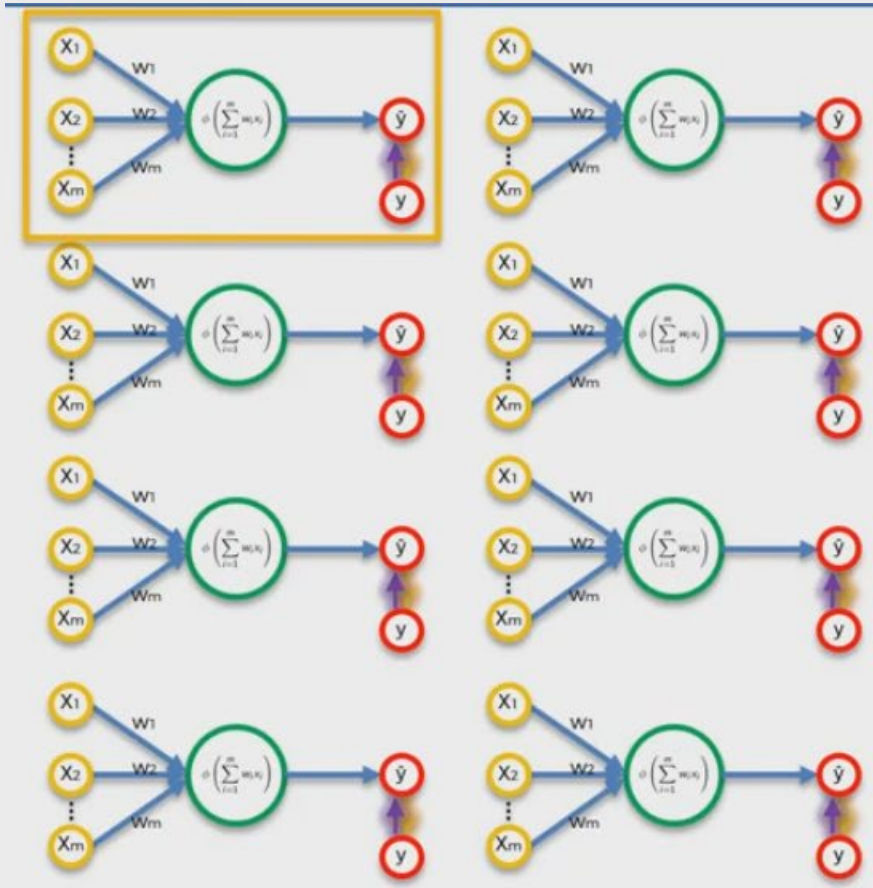

C
ŷ


C
Best!
ŷ



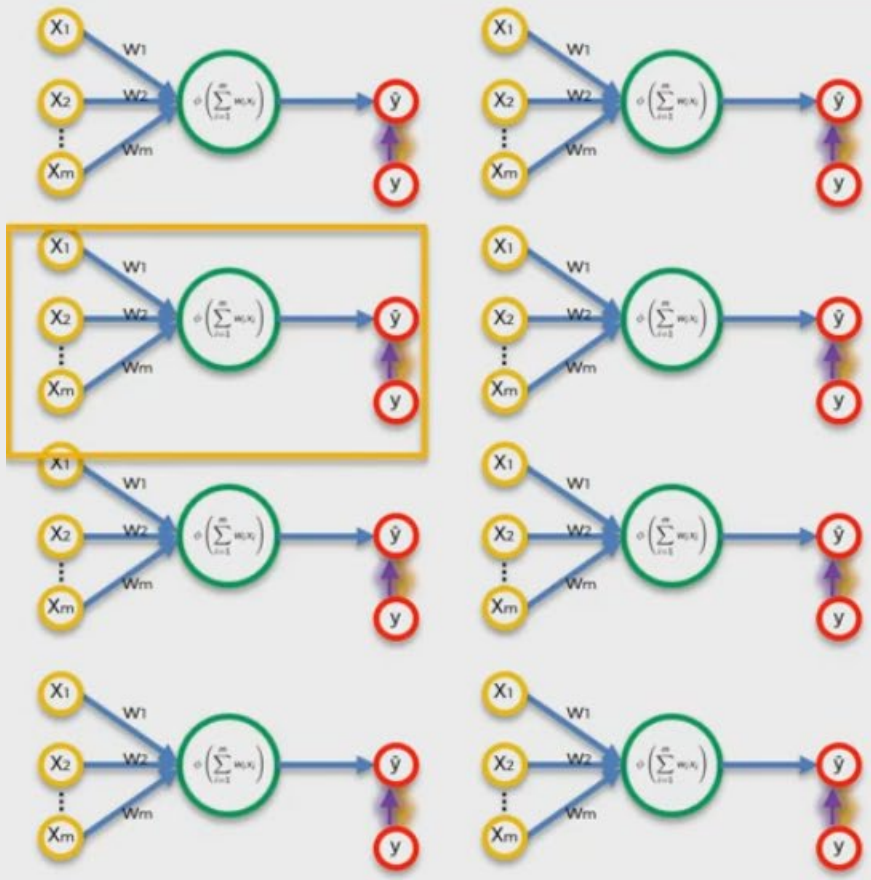
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

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



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

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



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

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

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

Upd w's

Batch Gradient Descent

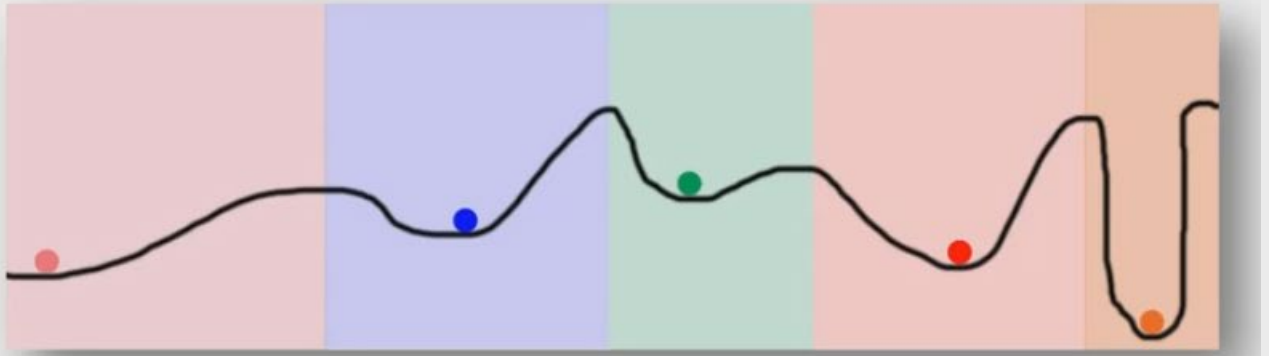
	Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
Upd w's	1	12	6	78%	93%
Upd w's	2	22	6.5	24%	68%
Upd w's	3	115	4	100%	95%
Upd w's	4	31	9	67%	75%
Upd w's	5	0	10	58%	51%
Upd w's	6	5	8	78%	60%
Upd w's	7	92	6	82%	89%
Upd w's	8	57	8	91%	97%

Stochastic Gradient Descent

Additional Reading:

A Neural Network in 13 lines of Python (Part 2 - Gradient Descent)

Andrew Trask (2015)



Link:

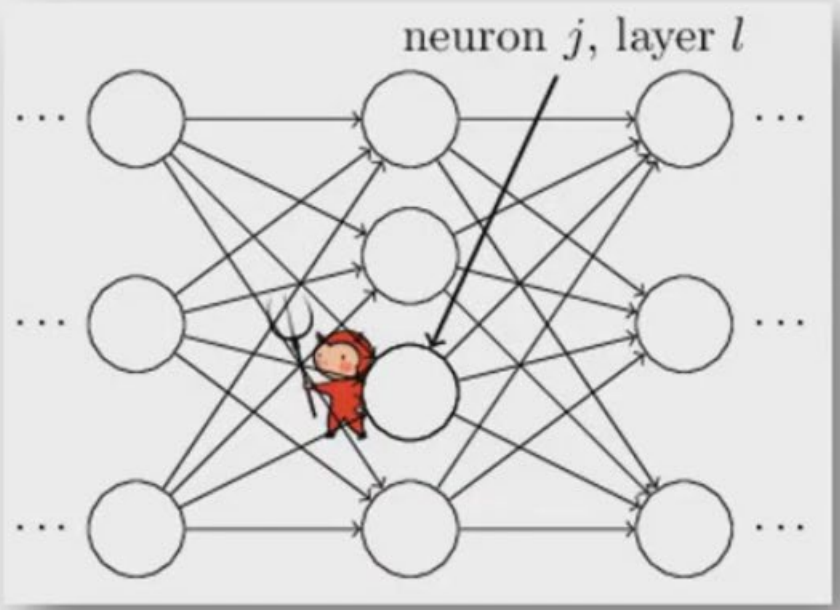
https://iamtrask.github.io/2015/07/27/python-network-part2/

Additional Reading:

Neural Networks and Deep Learning

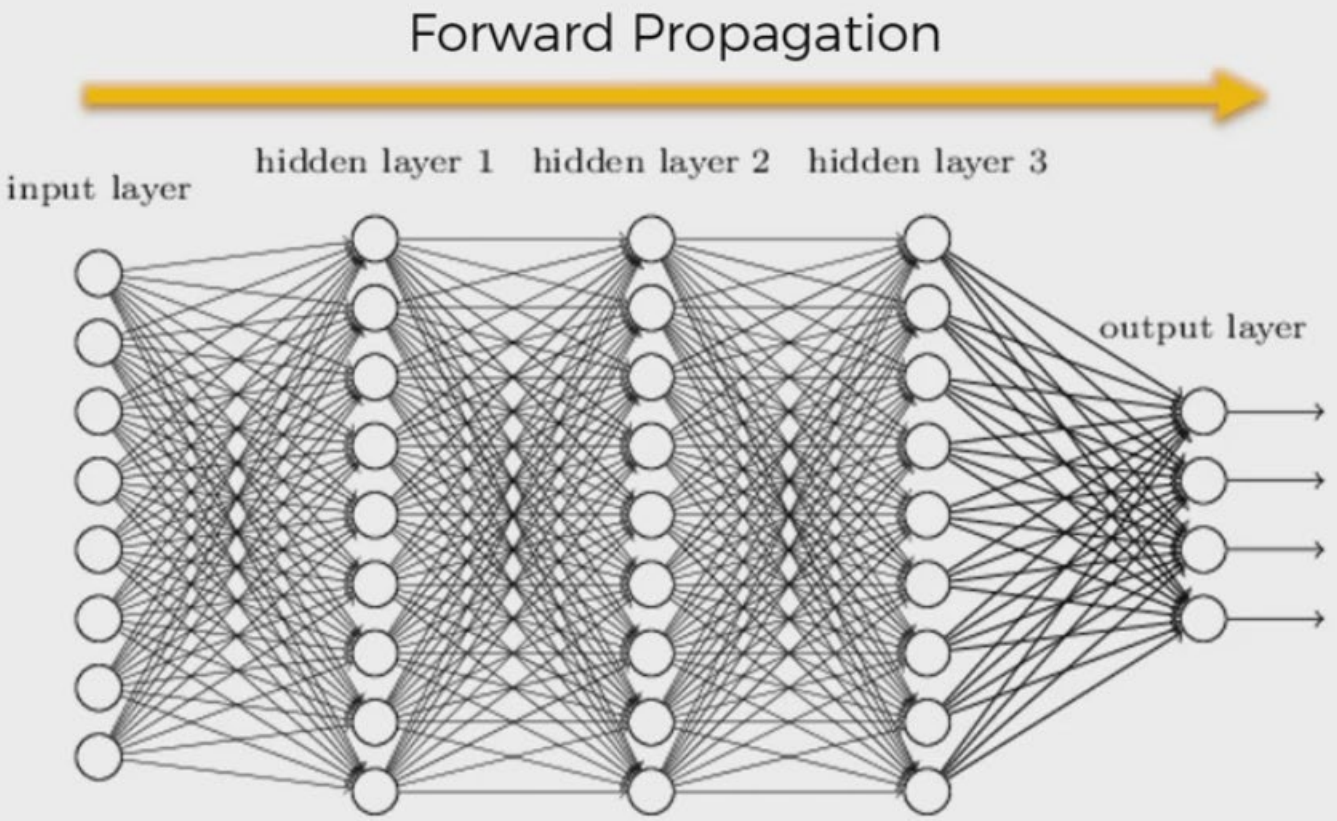
Michael Nielsen (2015)

Link:

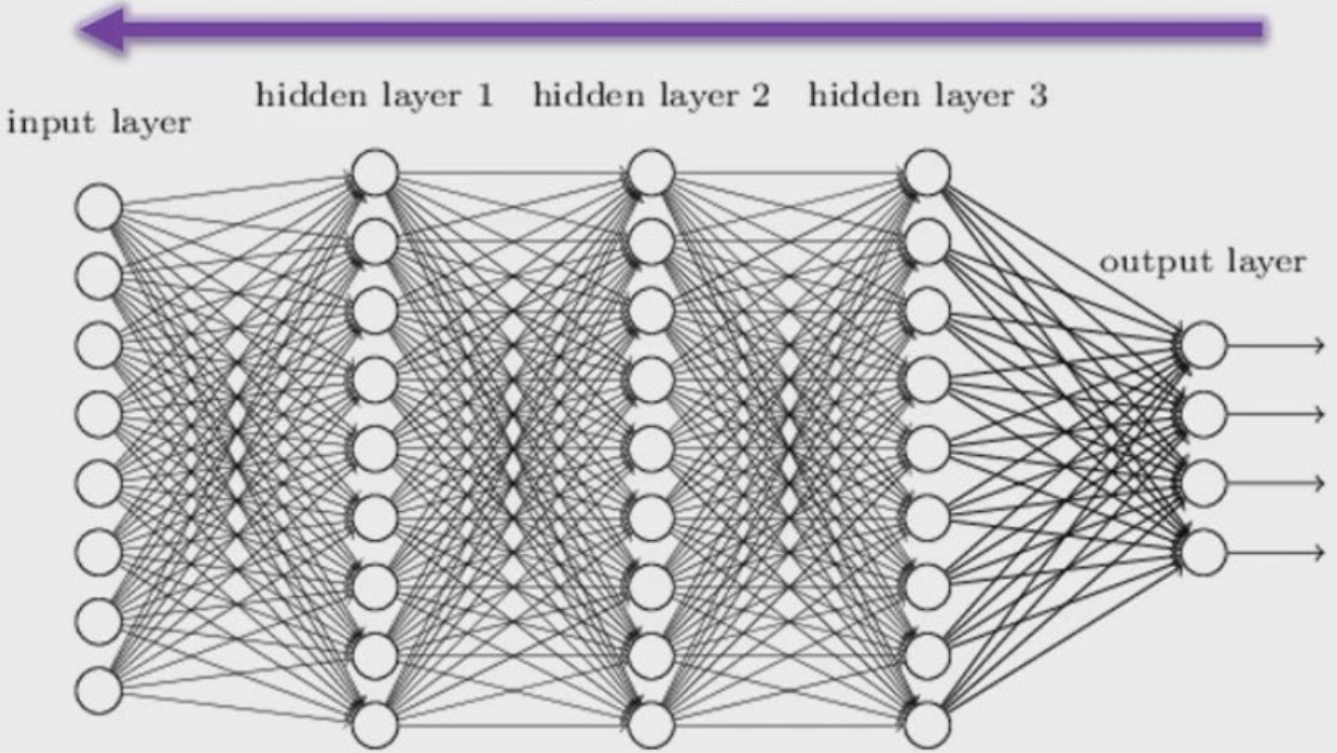


http://neuralnetworksanddeeplearning.com/chap2.html

Backpropagation


Forward Propagation
input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer

Backpropagation

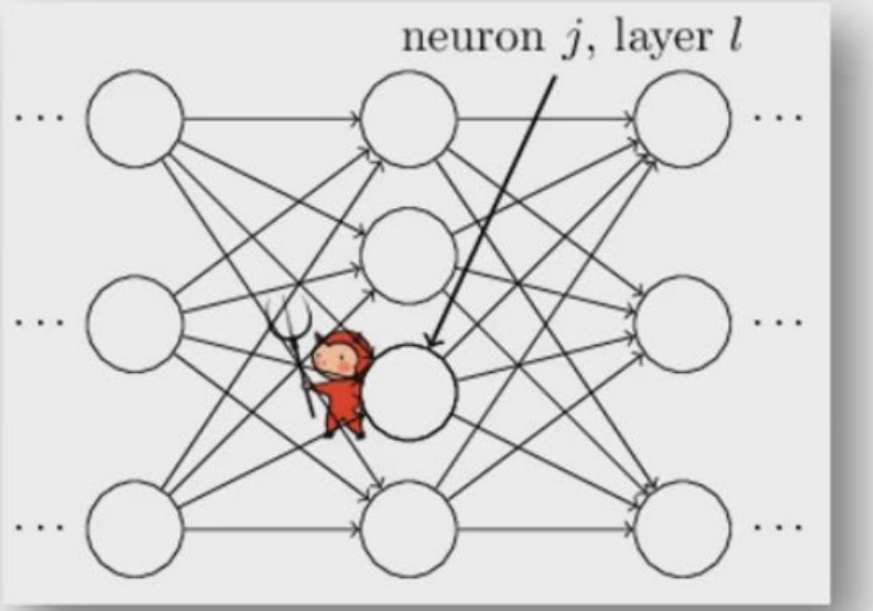


Additional Reading:

Neural Networks and Deep Learning

Michael Nielsen (2015)

Link:



http://neuralnetworksanddeeplearning.com/chap2.html

STEP 1: Randomly initialise the weights to small numbers close to 0 (but not 0).

STEP 2: Input the first observation of your dataset in the input layer, each feature in one input node.

STEP 3: Forward-Propagation: from left to right, the neurons are activated in a way that the impact of each neuron's activation is limited by the weights. Propagate the activations until getting the predicted result y.

STEP 4: Compare the predicted result to the actual result. Measure the generated error.

STEP 5: Back-Propagation: from right to left, the error is back-propagated. Update the weights according to how much they are responsible for the error. The learning rate decides by how much we update the weights.

STEP 6: Repeat Steps 1 to 5 and update the weights after each observation (Reinforcement Learning). Or:
Repeat Steps 1 to 5 but update the weights only after a batch of observations (Batch Learning).

STEP 7: When the whole training set passed through the ANN, that makes an epoch. Redo more epochs.