Deep learning using Keras

Ch1. Intro to Deep learning

By Eng. Mohammed Marwan Shahin

Training Outlines

- Neural Network Theories & Fundamentals.
- TensorFlow & Keras Libraries
- ANN Algorithm for Classification & Regression.
- Images Processing using OpenCV and CNN Algorithm.
- Creating Sequential modeling using RNN Algorithm.
- Text Processing using NLP Vectorization Techniques.
- Model Deployment Using Flask to Create Web Service (API).

Training Outlines – Day1

- Deep Learning History
- Artificial Neural Network
- Perceptron
- Forward propagation
- Activation function
- Cost/loss function
- Loss optimization & Gradient Decent (Back propagation)
- Project Discussion


CLICK
BBC
NEWS
WH
.GOV

What is Deep Learning?

Artificial Intelligence

Any technique that enables computers to mimic human behavior

Machine Learning

Ability to learn without explicitly being programmed

Deep Learning

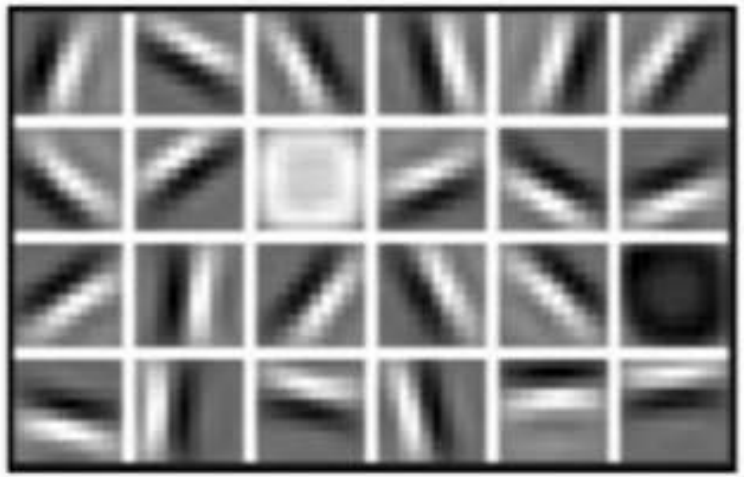
Extract patterns from data using neural networks

Why Deep Learning?

Hand engineered features are time consuming, brittle, and not scalable in practice

Can we learn the **underlying features** directly from data?

Low Level Features



Lines & Edges

Mid Level Features

Eyes & Nose & Ears

High Level Features

Facial Structure

Why Now?

1956

1980

2017

SanDisk Ultra 256GB MicroSDXC UHS-I Card with Adapter (SDSQUNI-256G-GN6MA).

by SanDisk

$149.99 ~~$199.99~~ Prime

Get it by **Wednesday, Mar 22**

FREE Shipping on eligible orders

More Buying Choices

$147.99 (10 new offers)

4.5 stars

21,224

Product Features

256GB capacity breakthrough

Why Now?

1956

2x

1980

25,600x

2017

SanDisk Ultra 256GB MicroSDXC UHS-I Card with Adapter (SDSQUNI-256G-GN6MA).

by SanDisk

$149.99 ~~$199.99~~ Prime

Wednesday, Mar 22

hipping on eligible

21,224

Product Features

256GB capacity breakthrough

hoices

$147.99 (10 new offers)

Why Now?

Neural Networks date back decades, so why the resurgence?

1. Big Data

- Larger Datasets
- Easier Collection & Storage

IMAGENET

2. Hardware

- Graphics Processing Units (GPUs)
- Massively Parallelizable

3. Software

- Improved Techniques
- New Models
- Toolboxes

TensorFlow

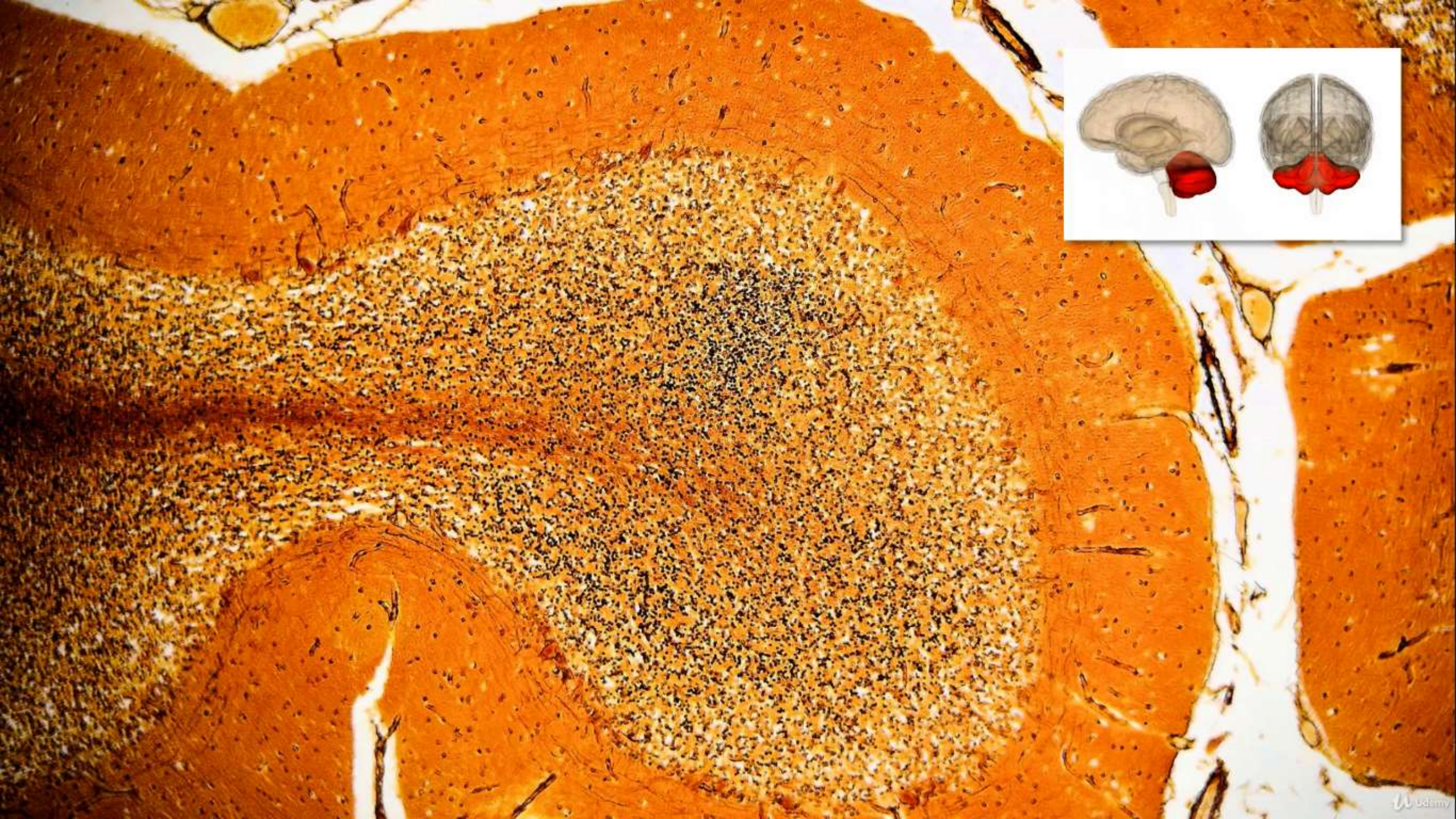
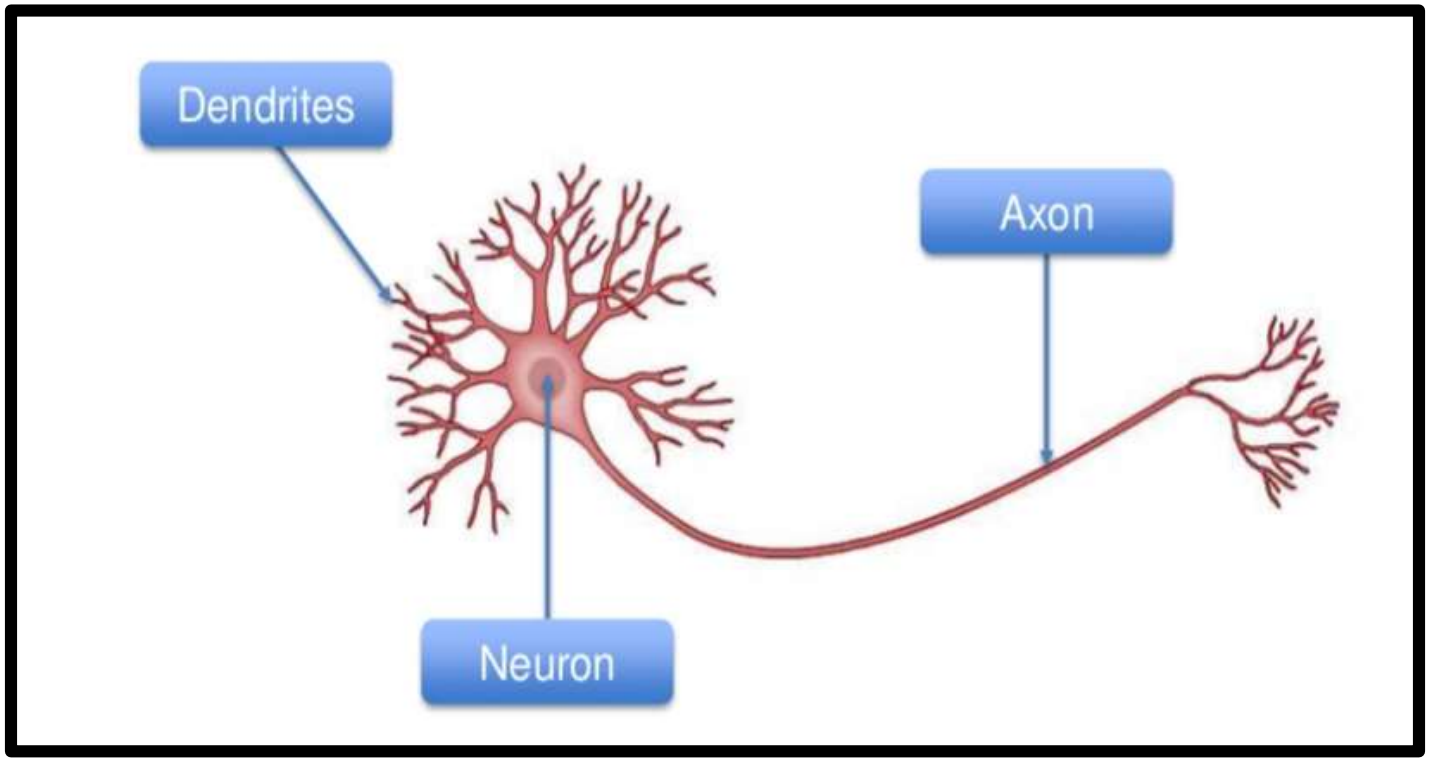


Image Source: www.austincc.edu

The Neuron

- The neuron that forms the basis of all Neural Networks is an imitation of what has been observed in the human brain.



Deep learning Application

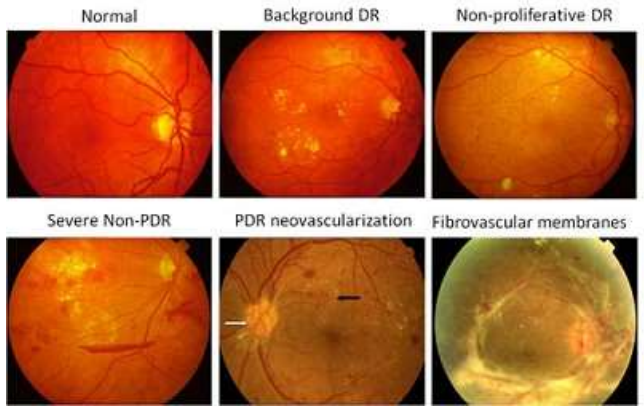
Medical Applications in Imaging

Tumor Detection/Medical Interpretation

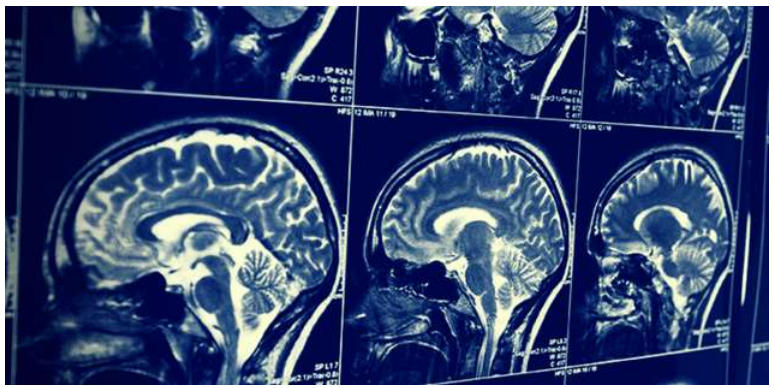
Over 5 million cases are diagnosed with skin cancer each year in the United States. The most commonly diagnosed cancer in the nation, skin cancer treatments cost the U.S. healthcare system over $8 billion annually.

Melanoma (the deadliest form of skin cancer) is highly curable if diagnosed early and treated properly, with survival rates varying between 15 percent and 65 percent from early to terminal stages respectively. Proper treatment can even produce a 5-year survival rate of over 98 percent.

To detect the tumor, the DL algorithm learns important features related to the disease from a group of medical images and then makes predictions (i.e. detection) based on that learning.



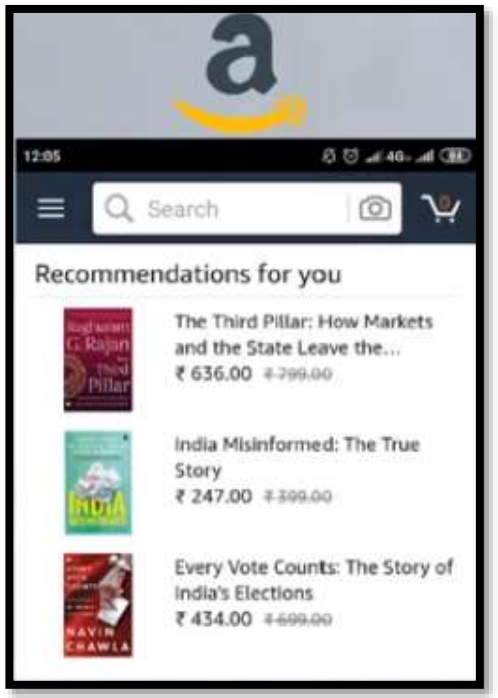
Diabetic Retinopathy



https://emerj.com/ai-sector-overviews/deep-learning-applications-in-medical-imaging/

Entertainment

- Netflix , Amazon

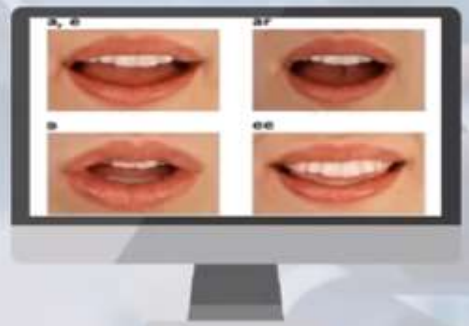


Music Generation & Adding Sounds To Silent Movies

Voice and audio recognition technology can be used to train a Deep Learning network to produce music compositions

Learns the patterns

Generates new music



Reads lip movements

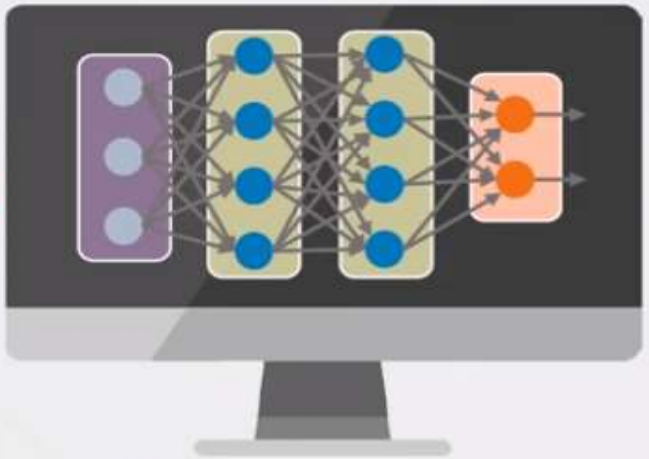
Google's Wavenet and Baidu's Deep Speech can train a computer to learn the patterns and the statistics that are unique to the music. It can then generate a completely new composition

Oxford and Google scientists created a neural network called LipNet that could read people's lips with 93% success. This can be used to add sounds to silent movies

Coloring Images and Videos

Deep Learning can be used to color images and videos by taking the objects and their context within the photograph

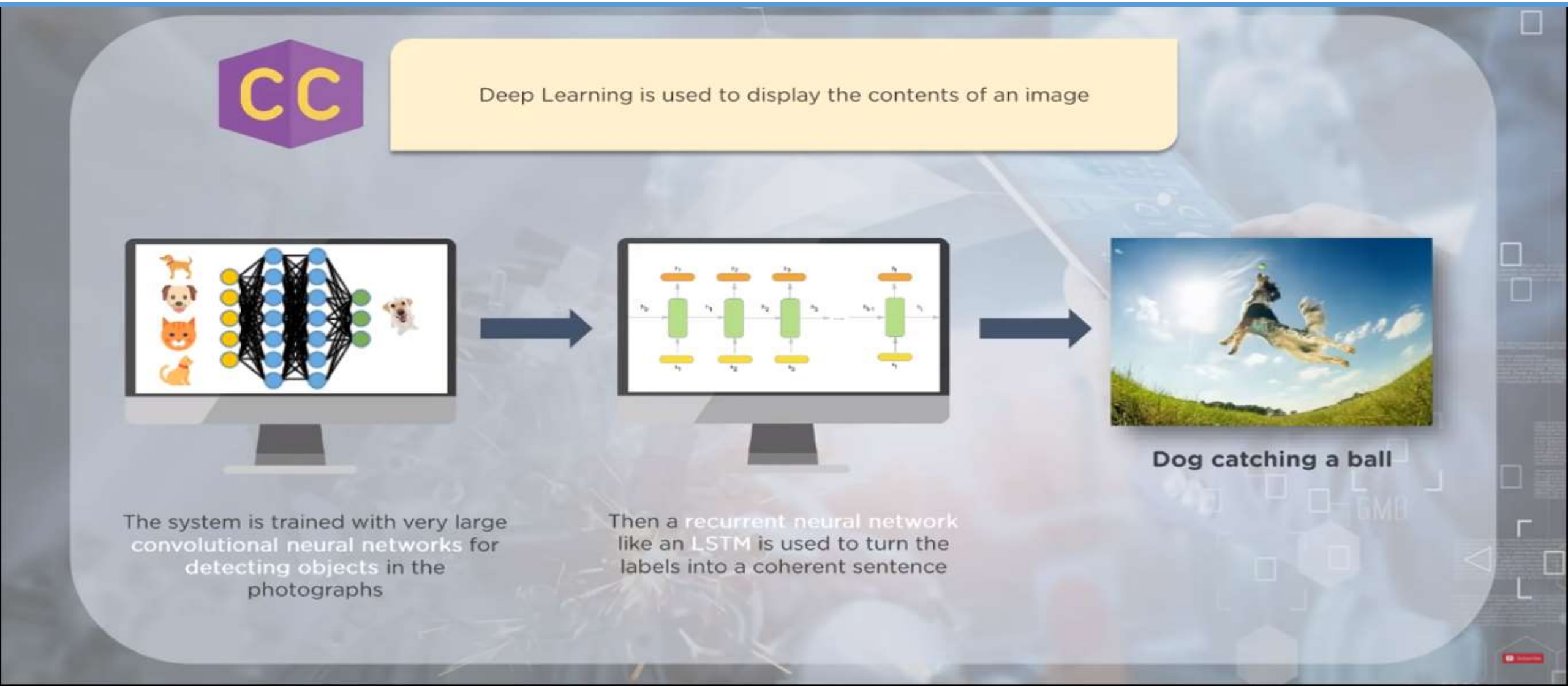
Black and white image is fed



Convolutional Neural Network is used to learn the patterns

Recreates the image by adding the colors

Image Caption Generation



Earthquake Prediction

Deep Learning model can be used to predict earthquakes by considering a factor called von Mises yield criterion

Scientists at Harvard used Deep Learning to teach a computer to perform viscoelastic computations, which are used in the prediction of earthquakes

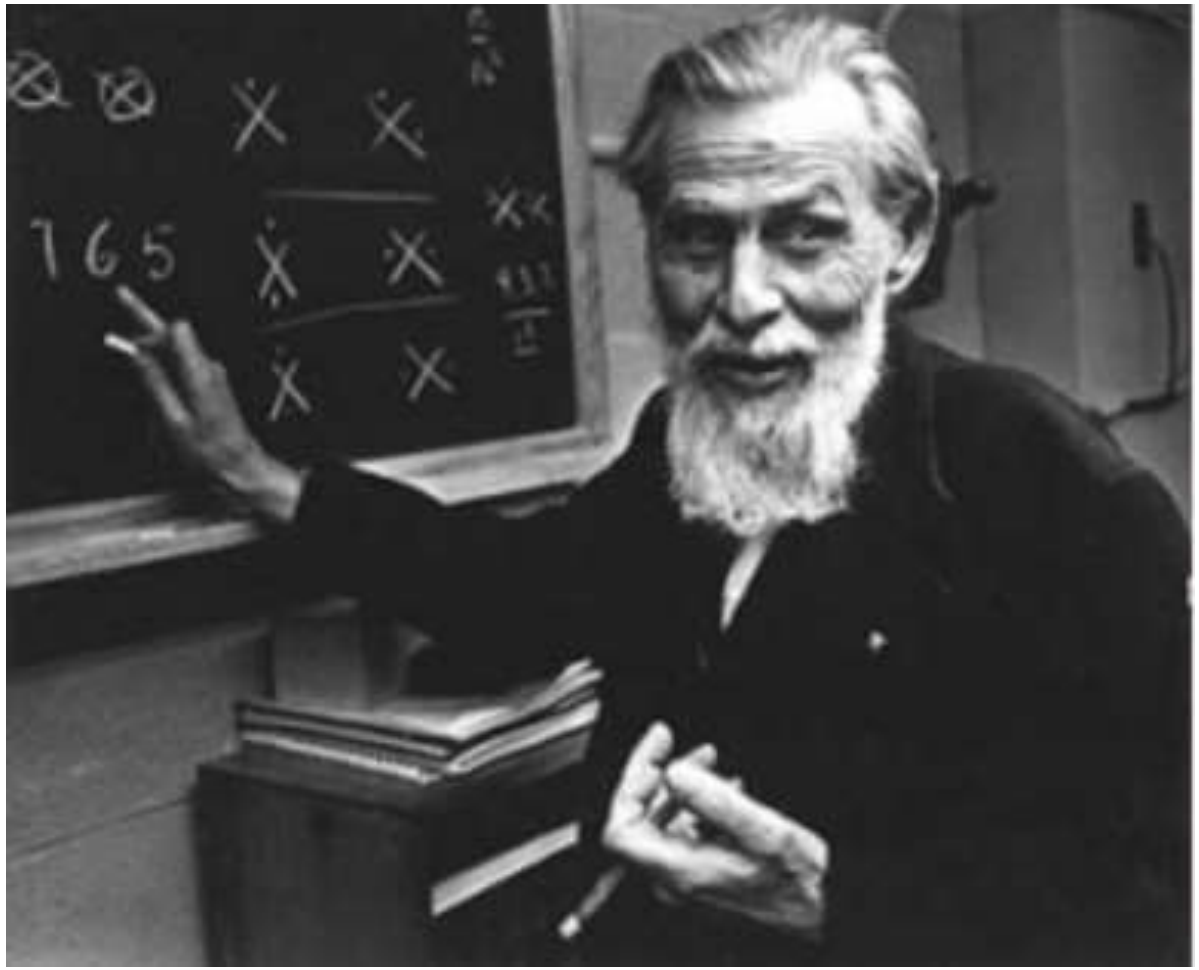
This application helped to improve the earthquake calculation time by **50,000%**

Neural Network History

The McCulloch-Pitts model of Neuron (1943 model)

He described how the human neuron could be represented mathematically in the characteristic of the biological neural network.

When a neuron receives a signal it does not automatically start continuing the signal downstream. It will hold onto the signal until a threshold is met and then it sends the message along its axon to be received by its neighboring cells.



The McCulloch-Pitts model of Neuron (1943 model)

This model is made up of a basic unit called Neuron.

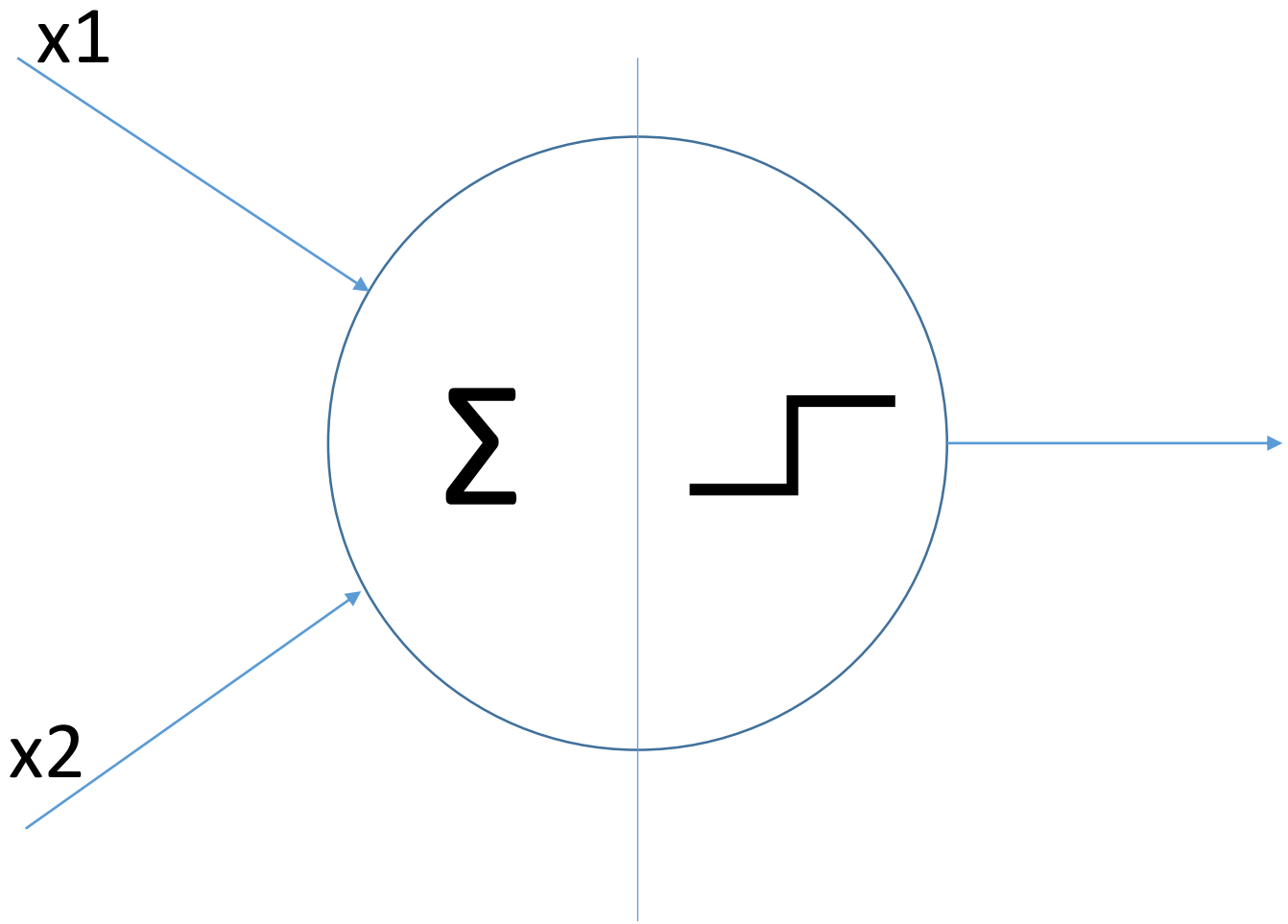
The main feature of their Neuron model is that a weighted sum of input signals is compared against a threshold to determine the neuron output. When the sum is greater than or equal to the threshold, the output is 1.

When the sum is less than the threshold, the output is 0. It can be put into the equations as such:

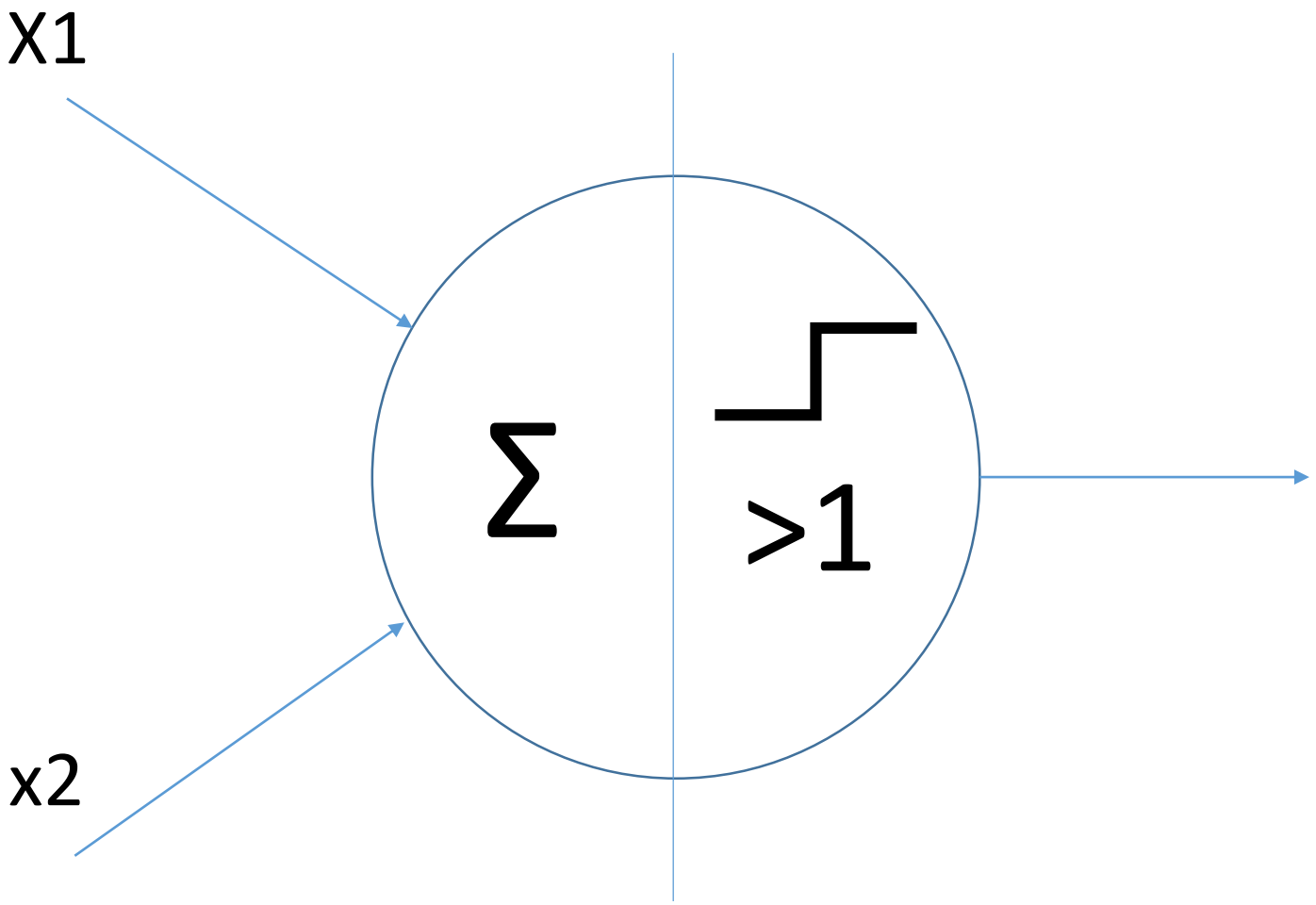
$$\text{Sum} = \sum_{i=1}^{N} I_i W_i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

$$\text{Output (y)} = f(Sum) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

McColloch

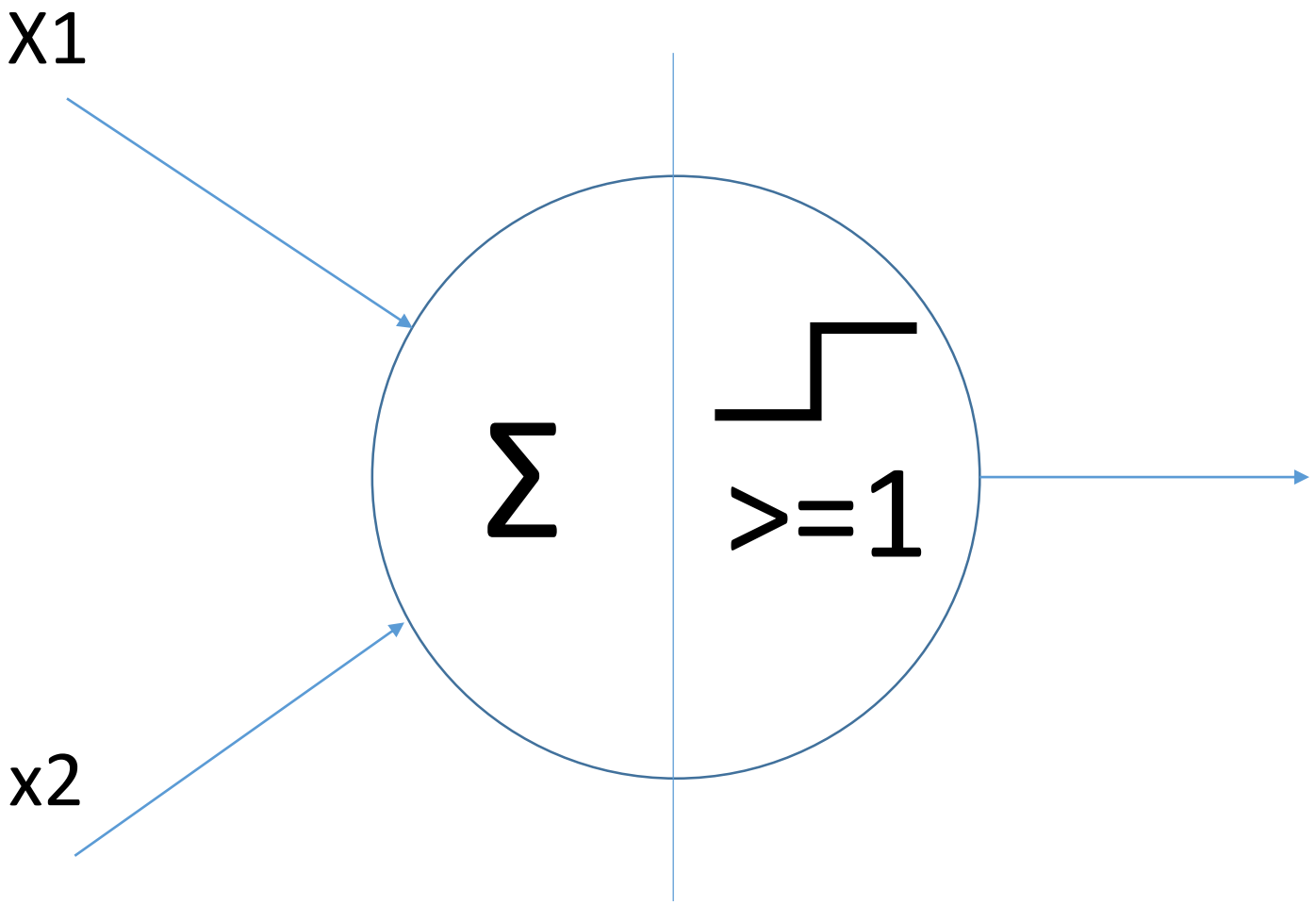


McColloch to Implement AND gate



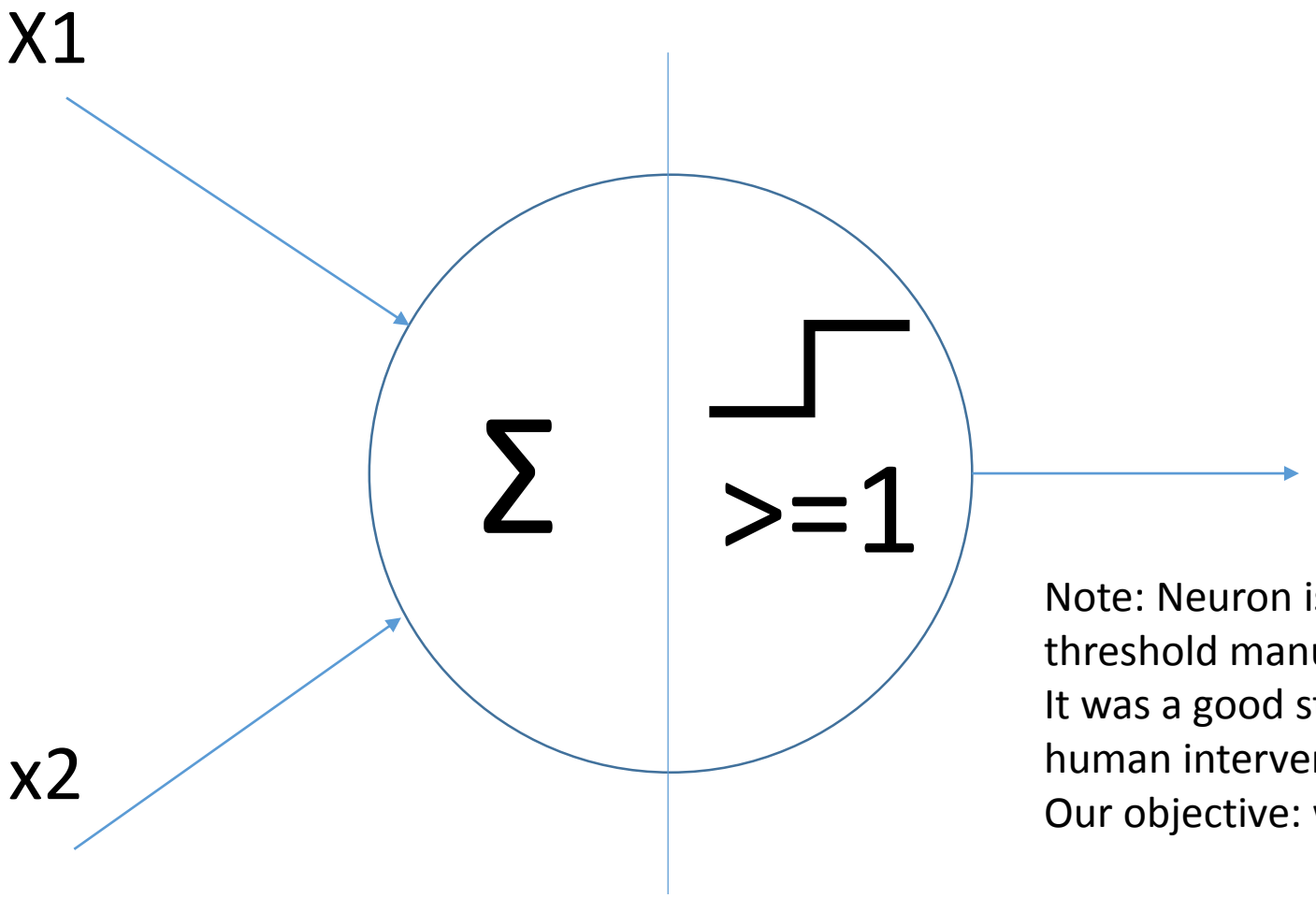
Input1	Input2	Output
0	0	0
0	1	0
1	0	0
1	1	1

McColloch to Implement OR gate



Input1	Input2	Output
0	0	0
0	1	1
1	0	1
1	1	1

McColloch to Implement OR gate



Input1	Input2	Output
0	0	0
0	1	1
1	0	1
1	1	1

Note: Neuron is not learning anything as we need to change the threshold manually based on the behavior we need.
It was a good start but not good enough , because it require human intervention.
Our objective: without human intervention can we make it learn.

Rosenblatt - 1957

He introduced the building blocks of neural networks, perceptron. He describes how these artificial neurons could learn from data.

He is credited with creating supervised learning which allowed for the neuron to alter its own weights based on its accuracy.

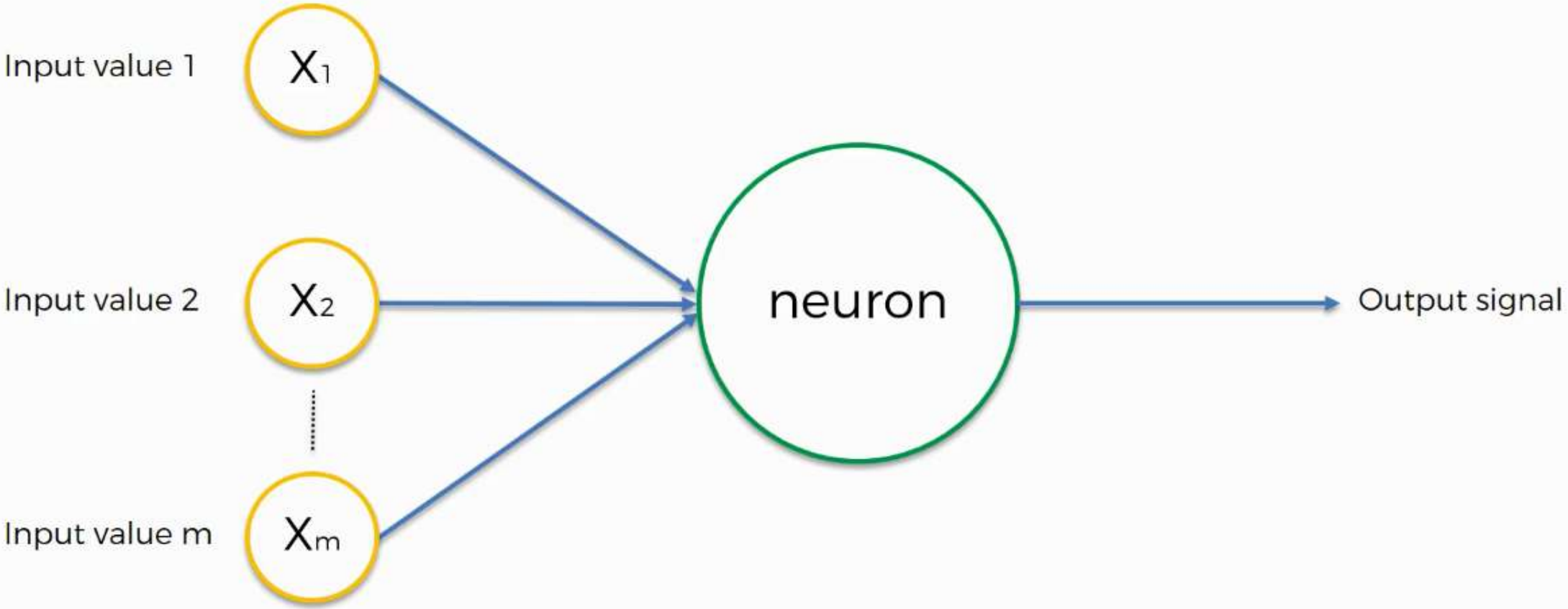
What is Perceptron?

The **perceptron** is a mathematical **model** of a biological **neuron**. While in actual **neurons** the dendrite receives electrical signals from the axons of other **neurons**, in the **perceptron** these electrical signals are represented as numerical values. ... As in biological neural networks, this output is fed to other **perceptrons**.

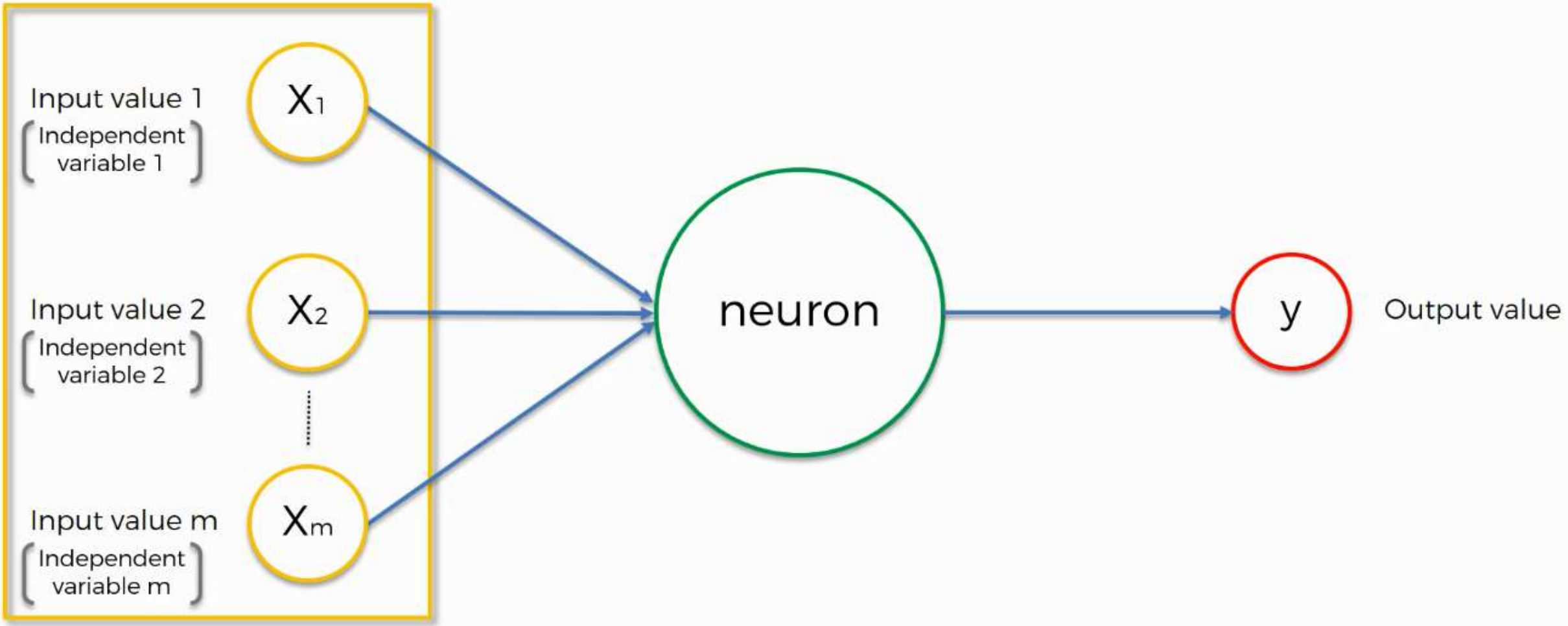
Consist of 4 parts:

- Input values
- Weights and bias
- Summation function
- Activation function

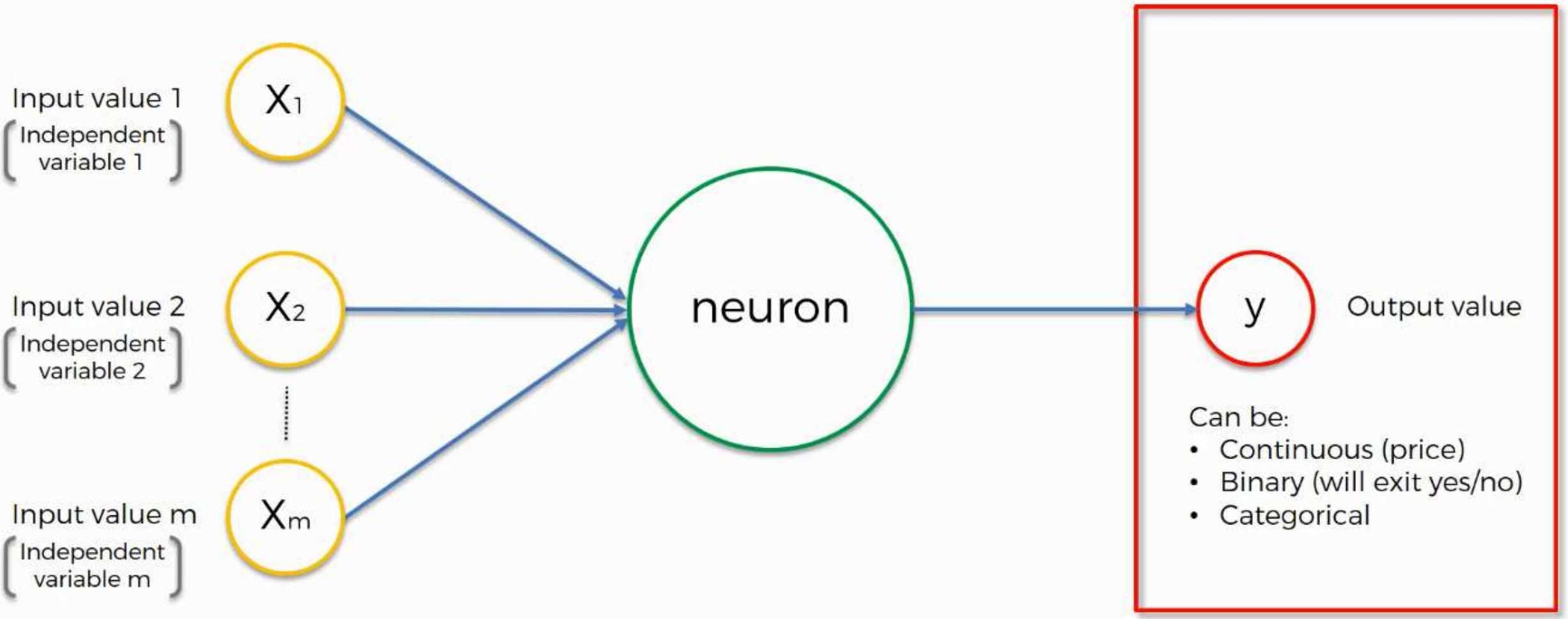
The Neuron


Input value 1
X_1
Input value 2
X_2
Input value m
X_m
neuron
Output signal

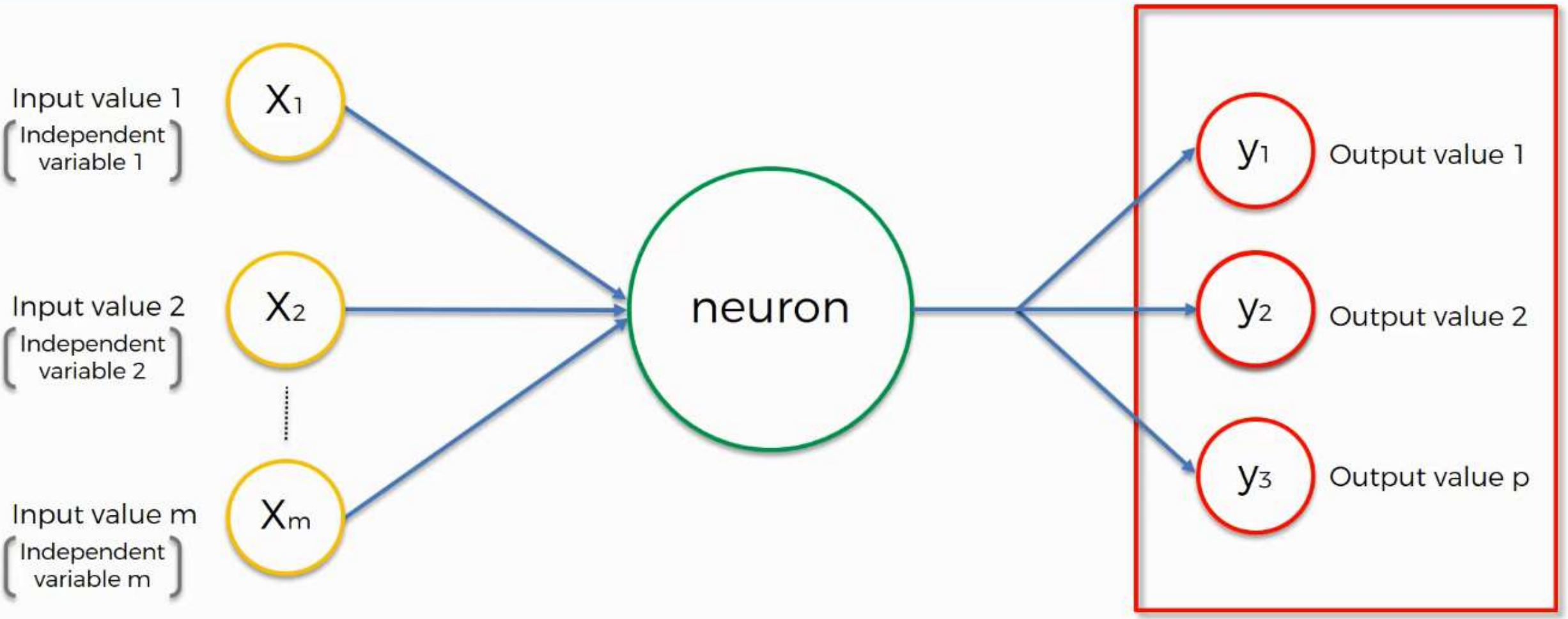
The Neuron


Input value 1
Independent variable 1
X_1
Input value 2
Independent variable 2
X_2
Input value m
Independent variable m
X_m
neuron
y
Output value

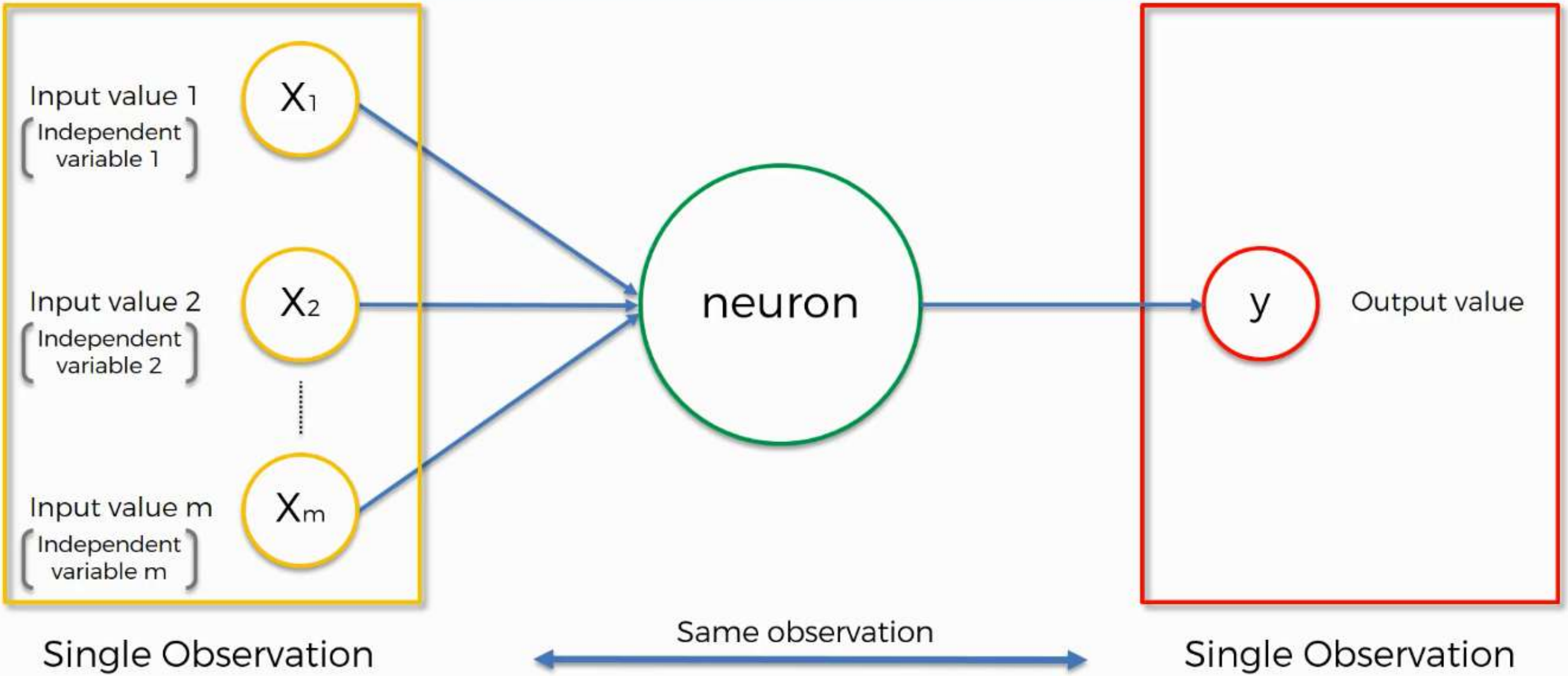
The Neuron


Input value 1
Independent variable 1
X_1
Input value 2
Independent variable 2
X_2
Input value m
Independent variable m
X_m
neuron
y
Output value
Can be:
• Continuous (price)
• Binary (will exit yes/no)
• Categorical

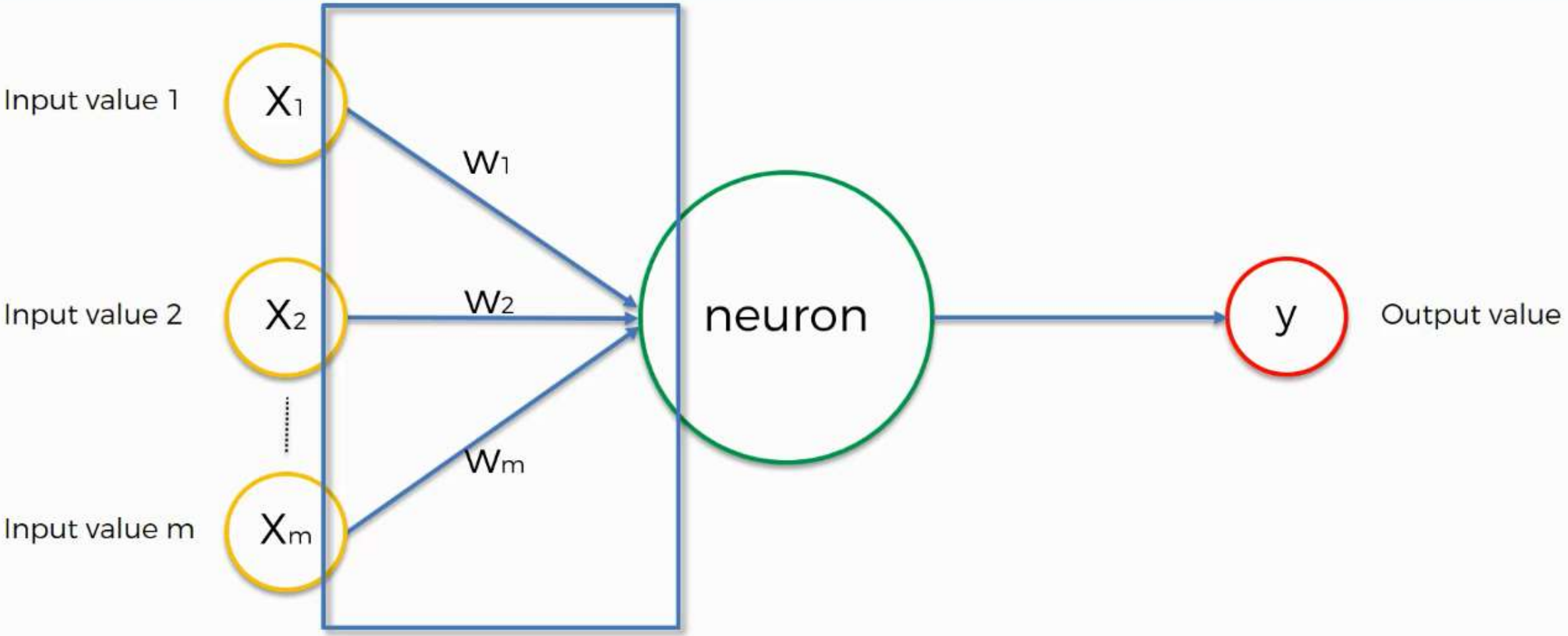
The Neuron


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

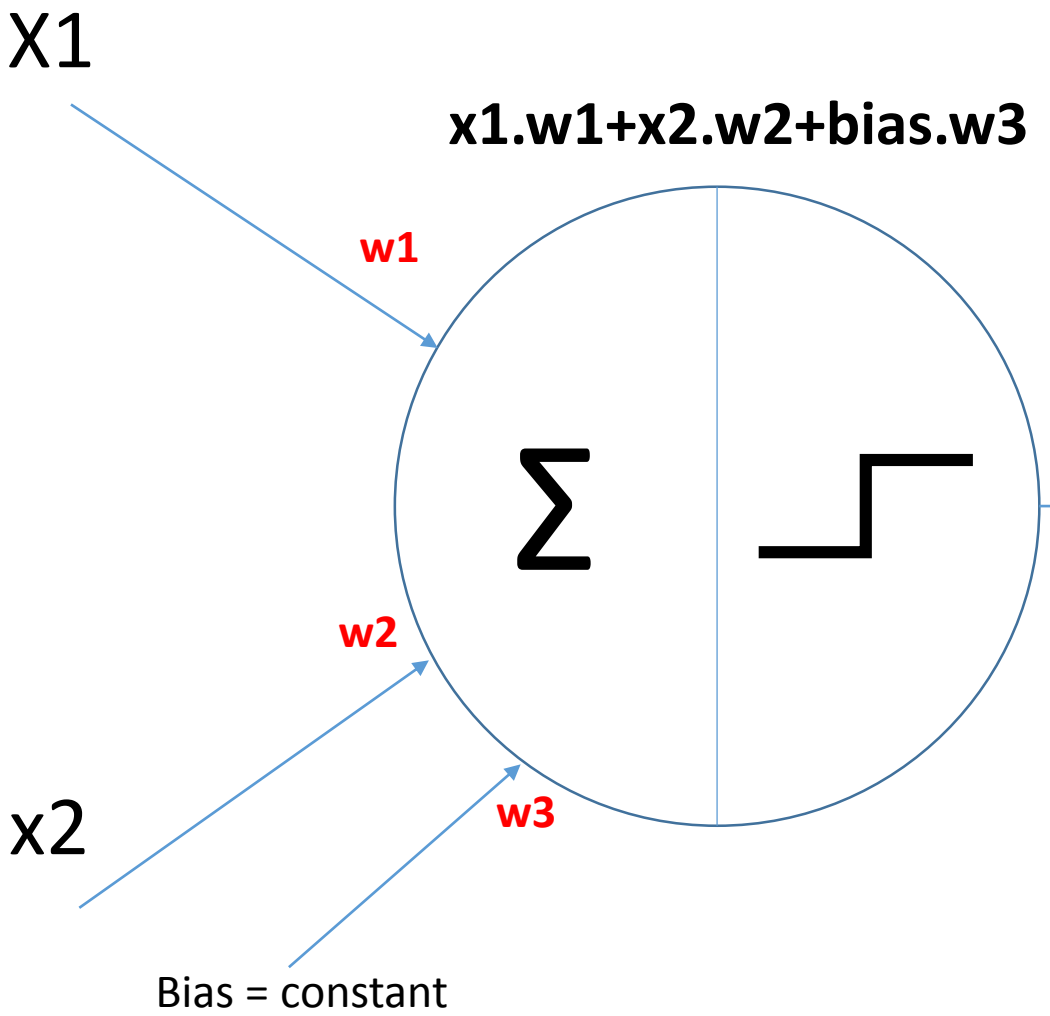
The Neuron


Input value 1
Independent variable 1
X_1
Input value 2
Independent variable 2
X_2
Input value m
Independent variable m
X_m
neuron
y
Output value
Single Observation
Same observation
Single Observation

The Neuron


Input value 1
X1
w1
Input value 2
X2
w2
neuron
y
Output value
wm
Input value m
Xm

Rosenblatt Neuron

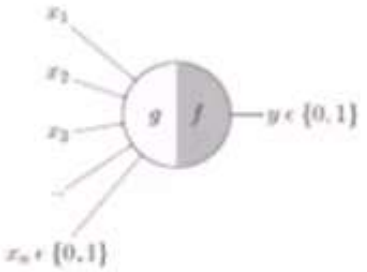


$$\hat{Y} - Y = E$$

Predicted actual error

Note: He have introduced the concept of weights and adjusting the weights to minimize the error, so no need to re-fix the threshold.
This was the first version of artificial neuron. By just giving the data and adjusting the neuron to minimize the error.
Rosenblatt called perceptron

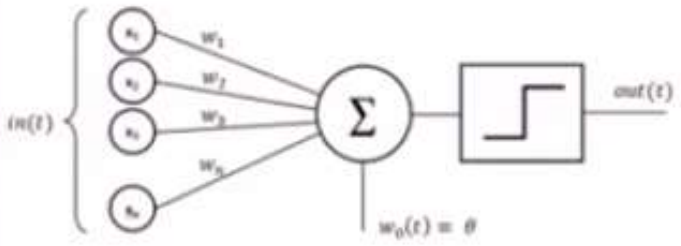
McColloch Vs. Rosenblatt Neuron



$$g(x_1, x_2, x_3, ..., x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$

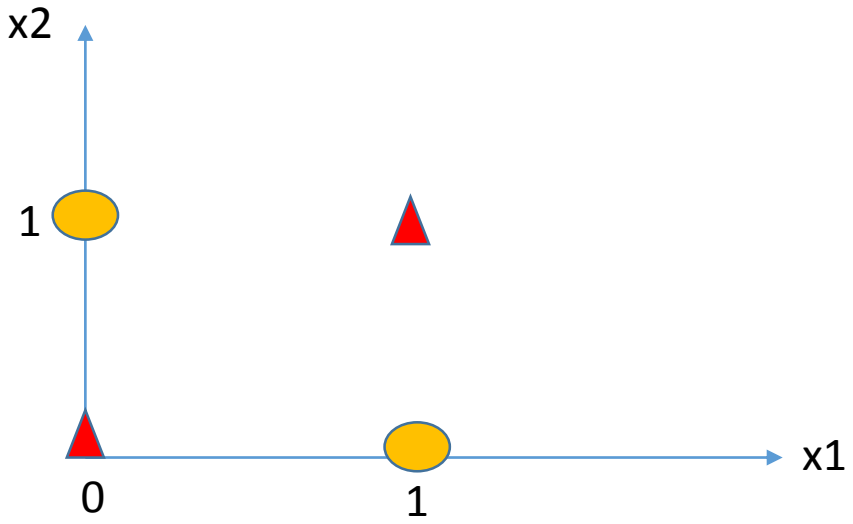
$$y = f(g(\mathbf{x})) = 1 \quad if \quad g(\mathbf{x}) \geq \theta$$
$$= 0 \quad if \quad g(\mathbf{x}) < \theta$$

1. It has an input layer that acts like dendrites
2. It has two parts, the first part, ***g*** is weighted addition of inputs. The weights are manually initialized and all have same weight
3. The weighted sum is passed through an activation function ***f*** which yields a 1 if threshold is crossed, else 0



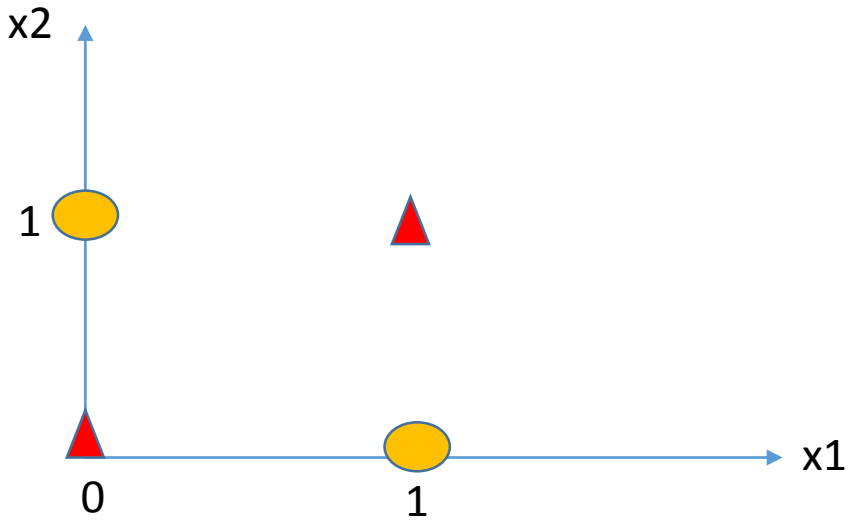
1. It has an Input layer that acts as dendrites
2. It has two parts, first part is weighted addition Each input is multiplied with a weight (which is typically initialized with some random value)
3. The sum is then passed through an activation function which yields a 1 if threshold is crossed
4. The step function can be defined in such a way that output can range from -1 to +1

Rosenblatt Neuron problem with XOR



Input1	Input2	Output
0	0	
0	1	
1	0	
1	1	

Rosenblatt Neuron problem with XOR

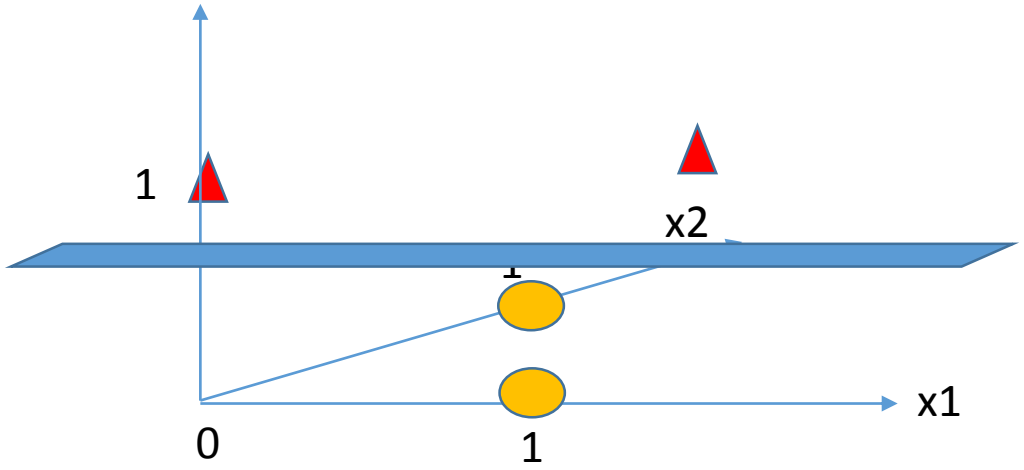


Input1	Input2	Output
0	0	0
0	1	1
1	0	1
1	1	0

XOR can not be divided and classified linearly. So Artificial Neuron could not able to implement XOR and it was rejected and Artificial neuron entered the winter season.

Solution for this was to use multiple Artificial neurons to implement any gate and they called it Artificial Neural Network (ANN).

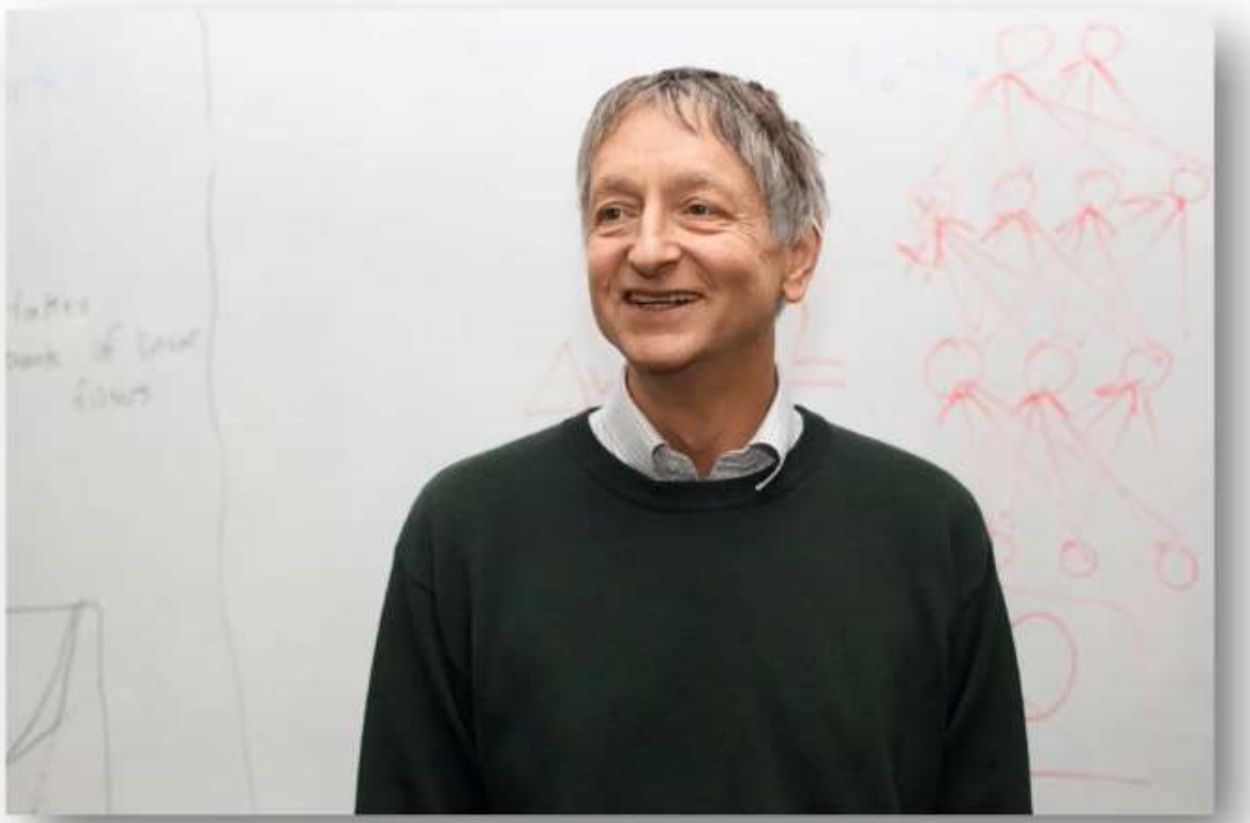
Rosenblatt Neuron problem with XOR



Input1	Input2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Covers Theorem : If we have distributions and we can not separate them then project them in higher dimension you will get more chance to separate them.

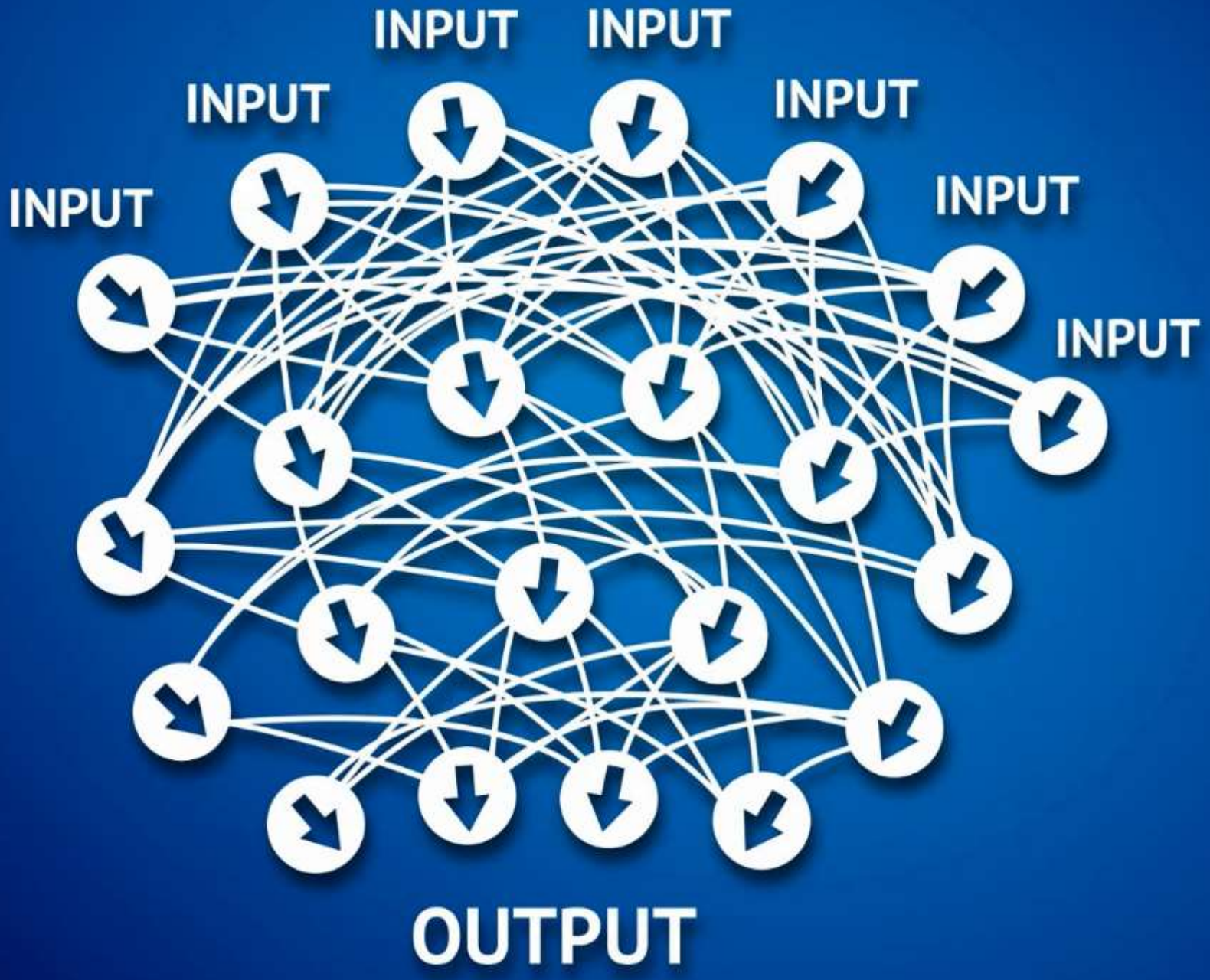
Geoffrey Everest Hinton is a British-Canadian cognitive psychologist and computer scientist, most noted for his work on artificial neural networks. Since 2013, he has divided his time working for Google and the University of Toronto.



Geoffrey Hinton

Yann LeCun is a French computer scientist working primarily in the fields of machine learning, computer vision, mobile robotics, and computational neuroscience.

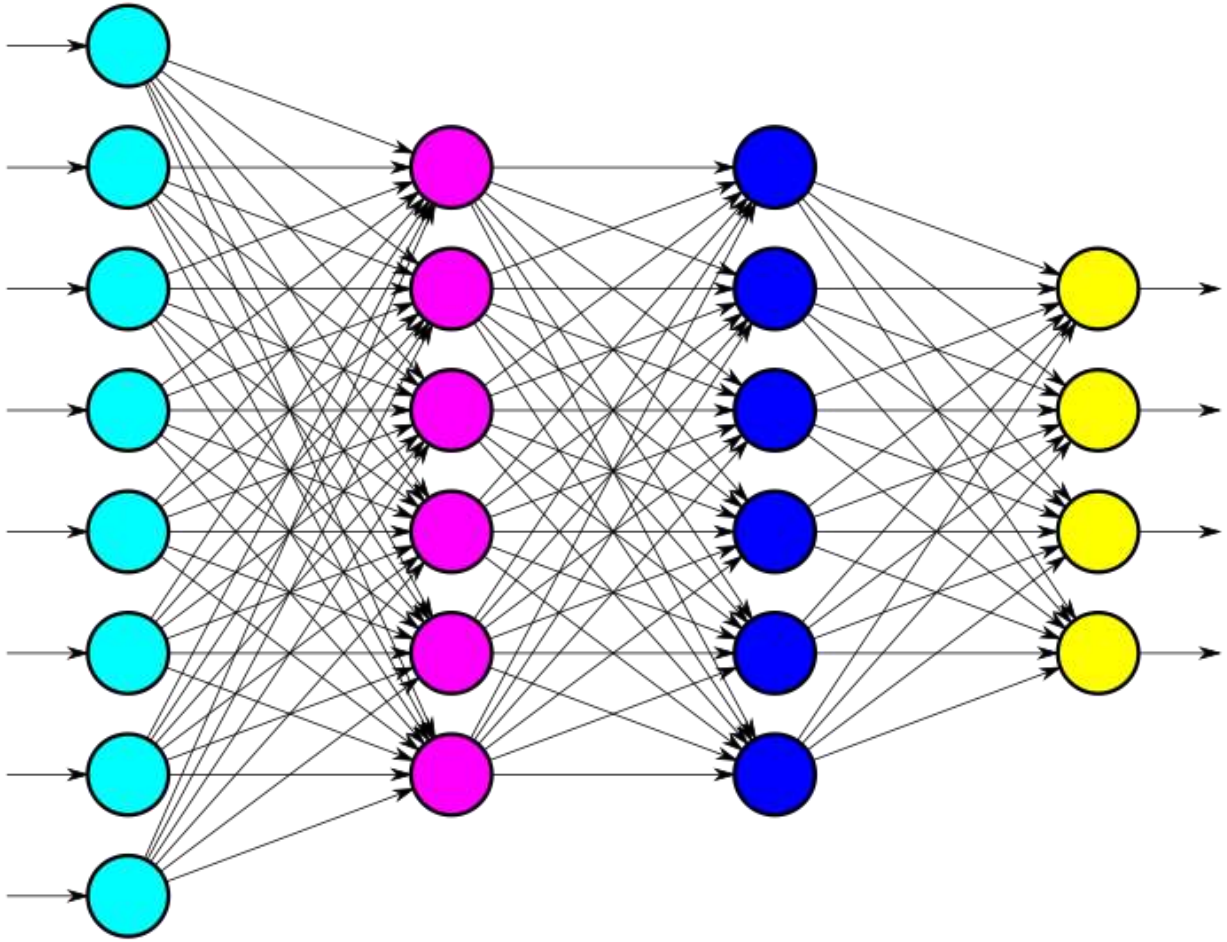
Yann LeCun - CNN



What Does Deep Mean In Deep Learning?

Deep Learning structure component:

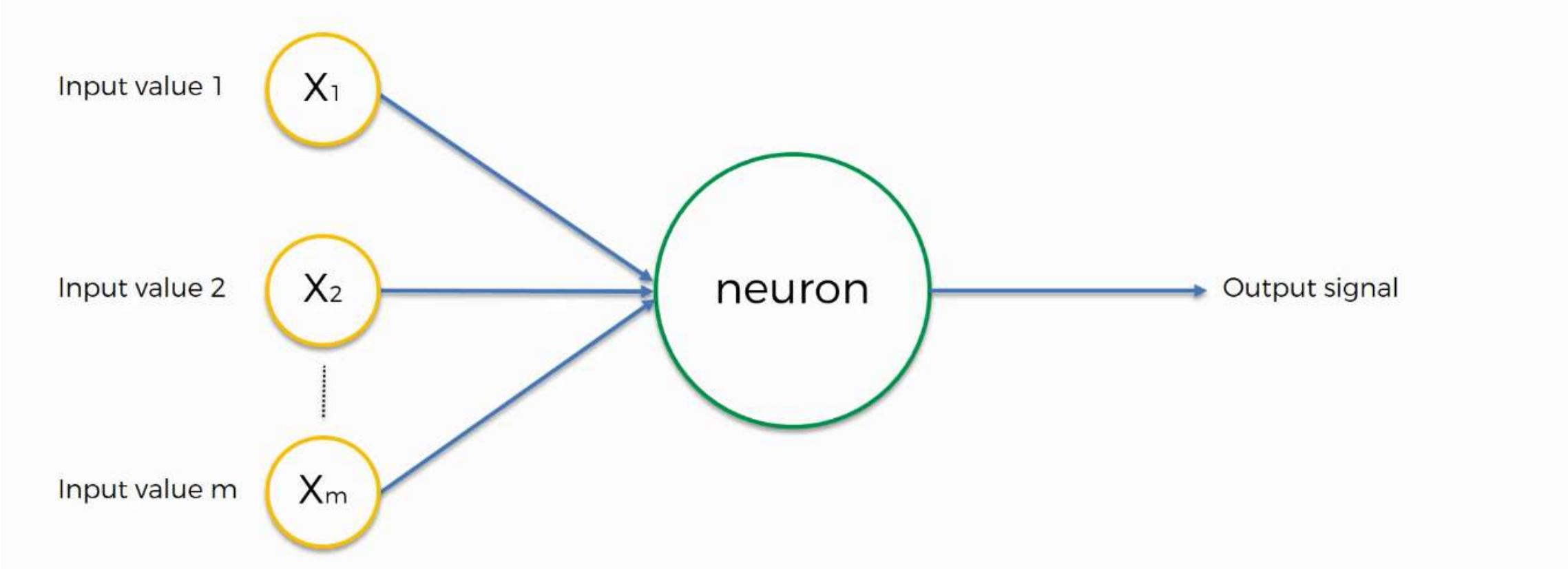
- ANNs are built using what we call neurons.
- Neurons in an ANN are organized into what we call layers.
- Layers *within* an ANN (all but the input and output layers) are called hidden layers.
- If an ANN has more than one hidden layer, the ANN is said to be a deep ANN.



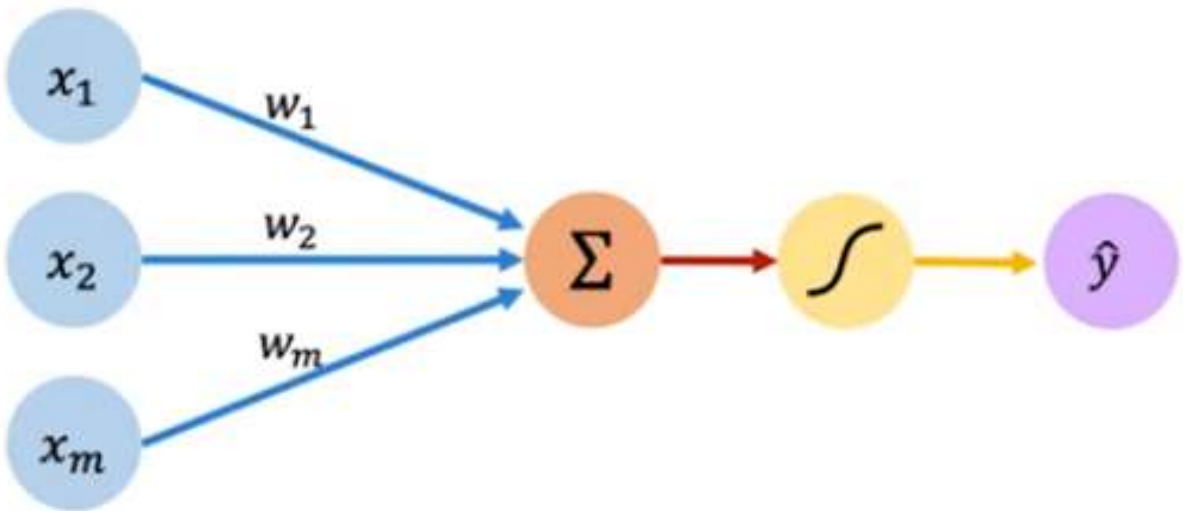
Forward Propagation

The Perceptron: Forward Propagation

As the name suggests, the input data is fed in the forward direction through the network. Each hidden layer accepts the input data, processes it as per the activation function and passes to the successive layer.



The Perceptron: Forward Propagation



Inputs Weights Sum Non-Linearity Output

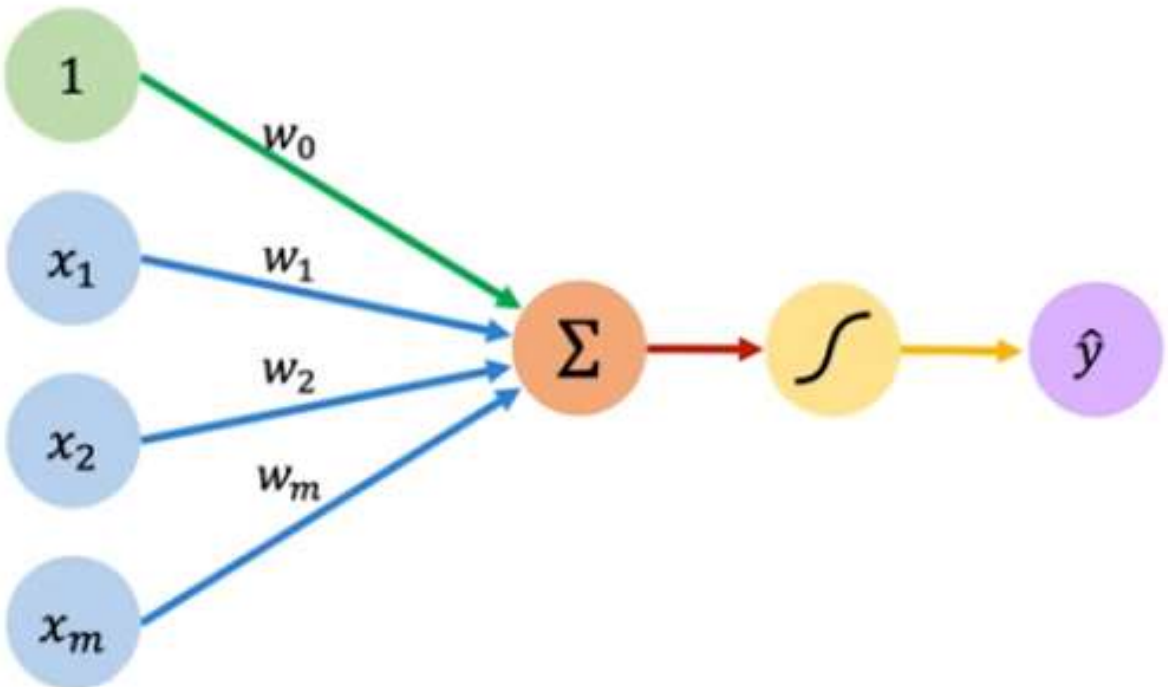
Output

Linear combination of inputs

$$\hat{y} = g\left(\sum_{i=1}^{m} x_i \, w_i\right)$$

Non-linear activation function

The Perceptron: Forward Propagation



Inputs Weights Sum Non-Linearity Output

Output

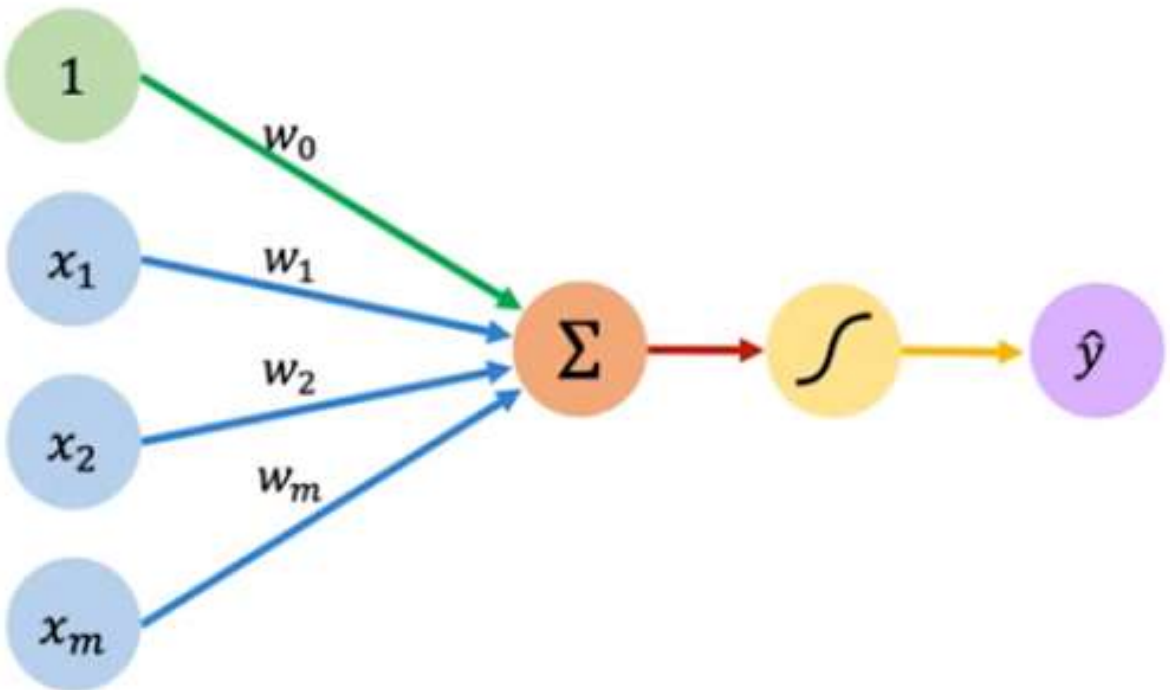
Linear combination of inputs

$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i\, w_i\right)$$

Non-linear activation function

Bias

The Perceptron: Forward Propagation



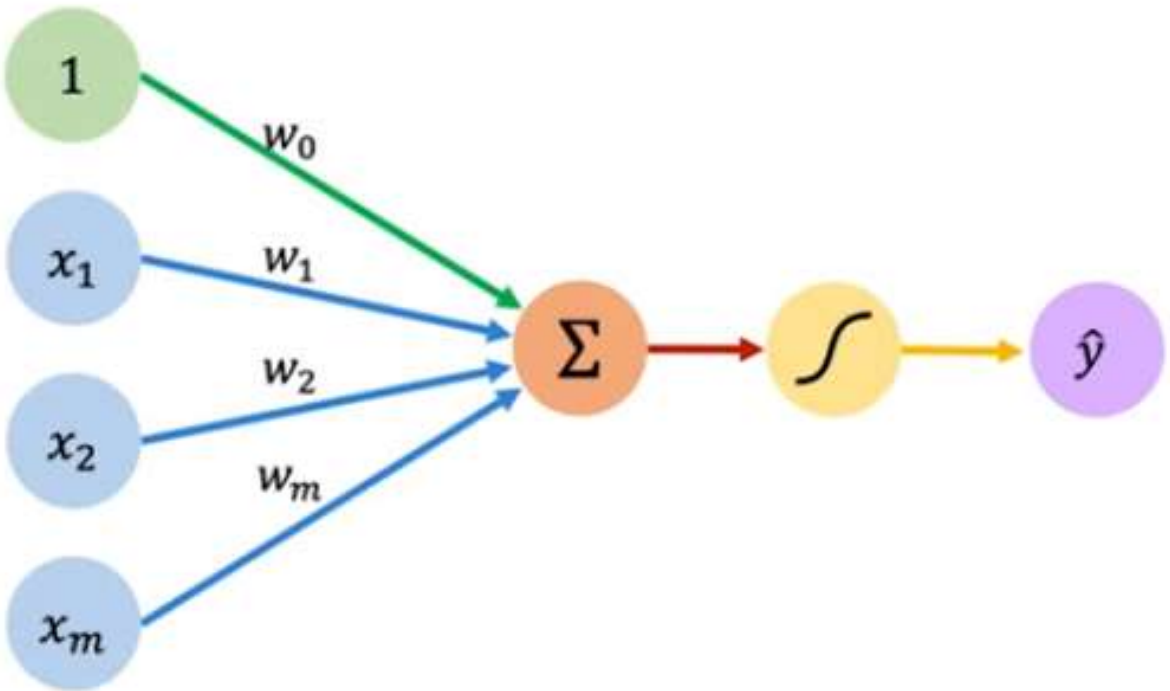
Inputs Weights Sum Non-Linearity Output

$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i\, w_i\right)$$

$$\hat{y} = g\left(w_0 + \boldsymbol{X}^T \boldsymbol{W}\right)$$

where: $\boldsymbol{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$ and $\boldsymbol{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$

The Perceptron: Forward Propagation



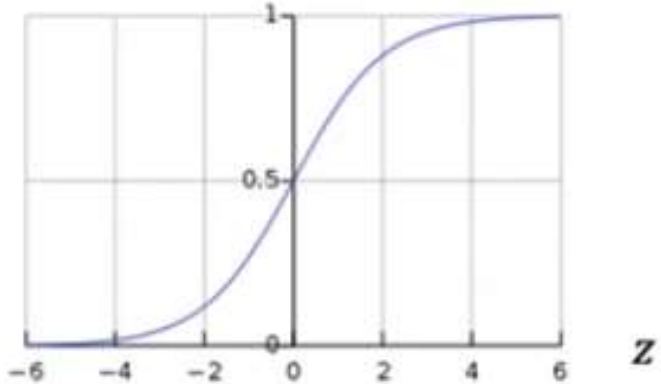
Inputs Weights Sum Non-Linearity Output

Activation Functions

$$\hat{y} = g\left(w_0 + \boldsymbol{X}^T \boldsymbol{W} \right)$$

- Example: sigmoid function

$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$



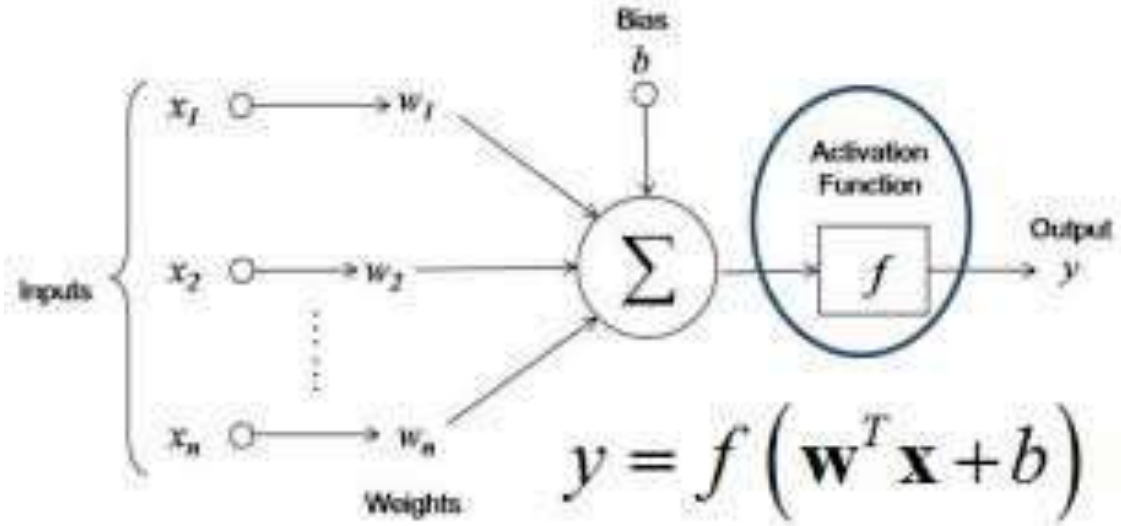
Activation Function

What is Activation Function

An activation function is a very important feature of an artificial neural network , they basically decide whether the neuron should be activated or not.

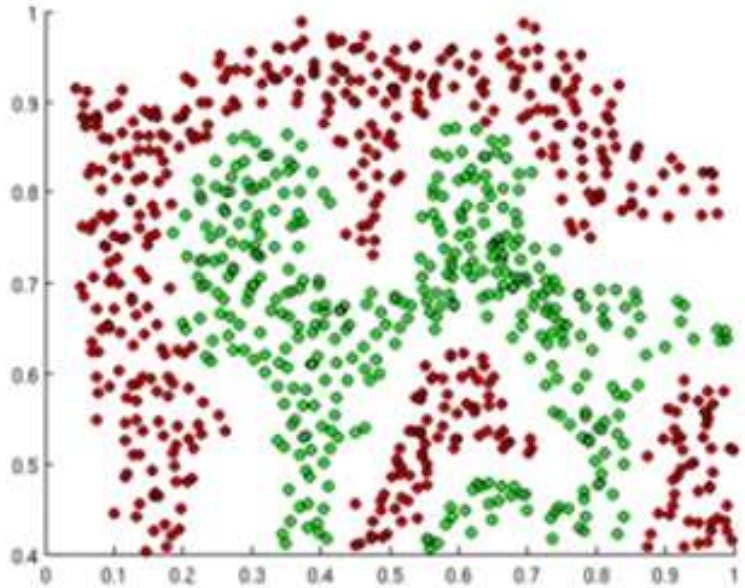
In artificial neural networks, the activation function defines the output of that node given an input or set of inputs. Important use of any activation function is to introduce non-linear properties to our Network.

Activation functions help in normalizing the output between 0 to 1 or -1 to 1.



Importance of Activation Functions

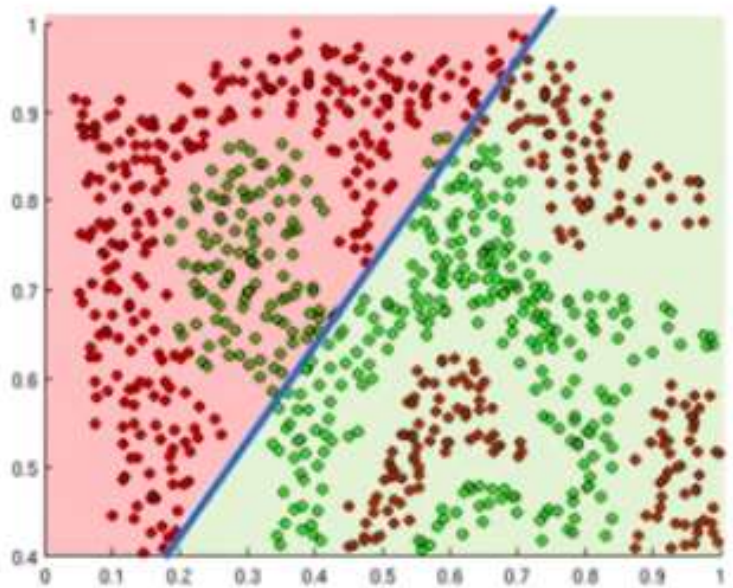
The purpose of activation functions is to ***introduce non-linearities*** *into the network*



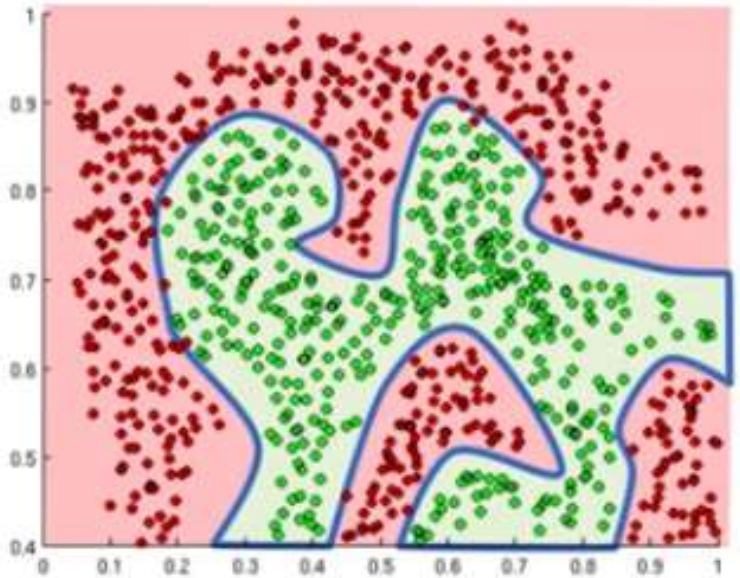
What if we wanted to build a neural network to distinguish green vs red points?

Importance of Activation Functions

*The purpose of activation functions is to **introduce non-linearities** into the network*



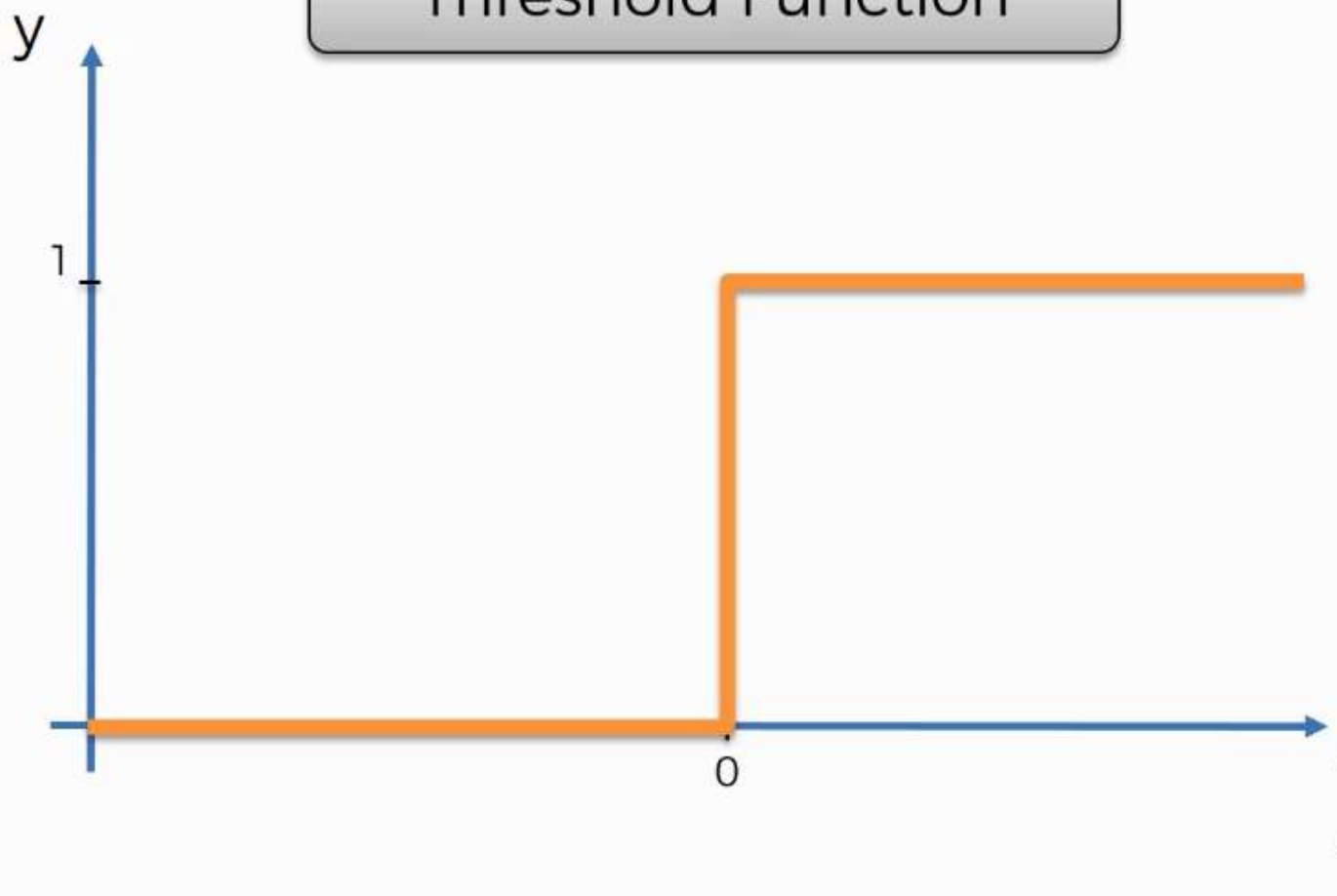
Linear activation functions produce linear decisions no matter the network size



Non-linearities allow us to approximate arbitrarily complex functions

The Activation Function

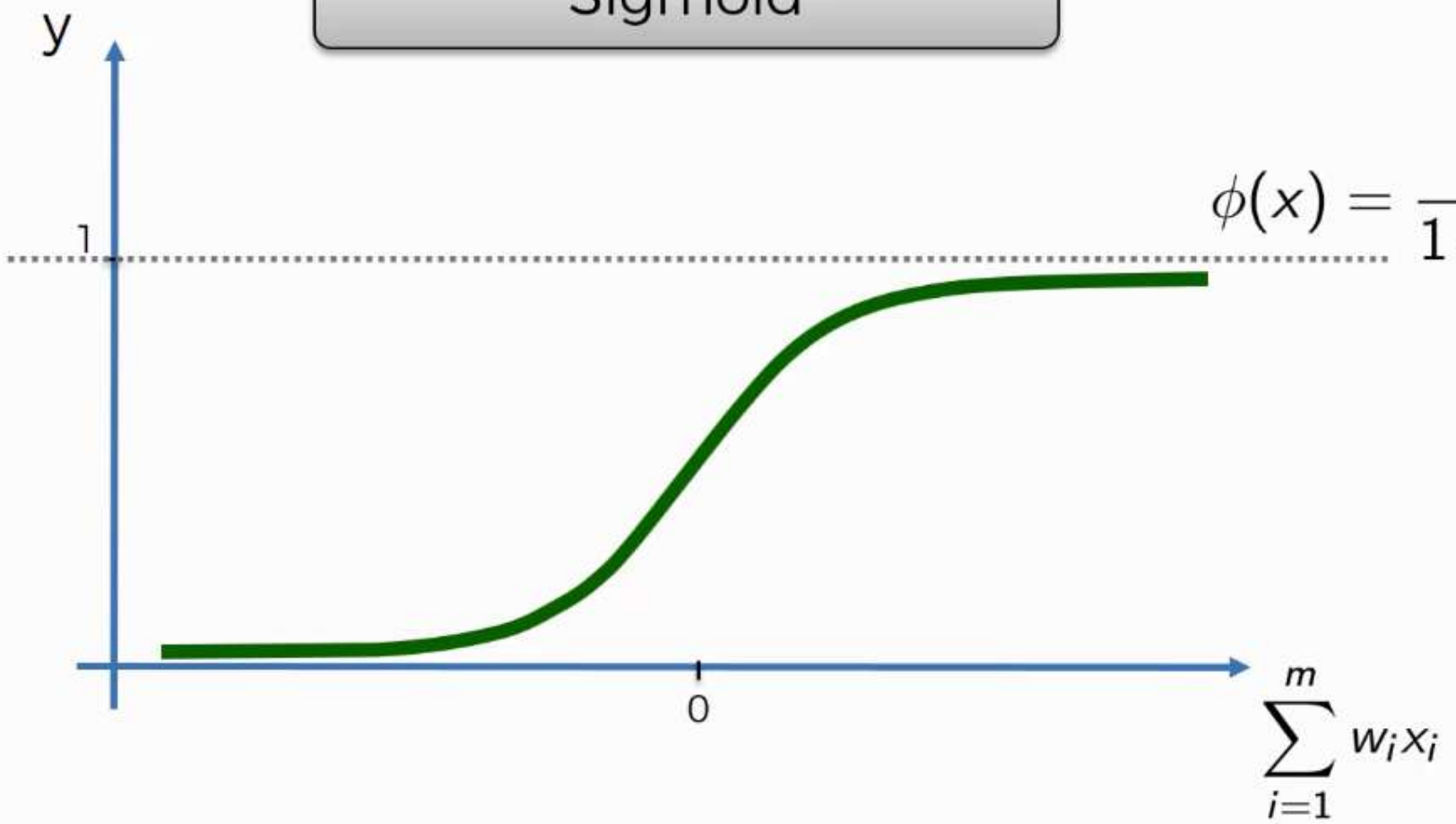
Threshold Function



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

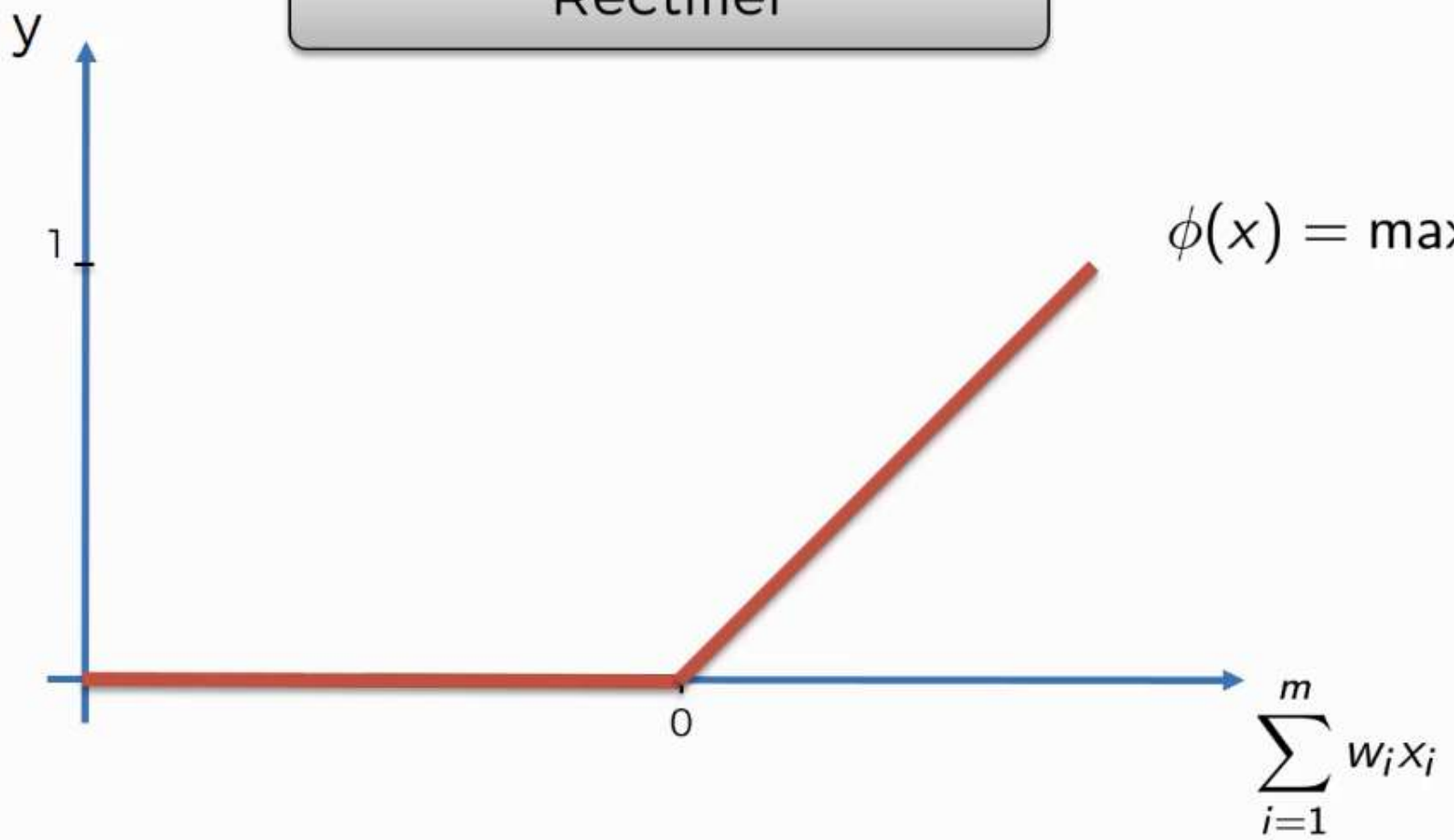
The Activation Function

Sigmoid



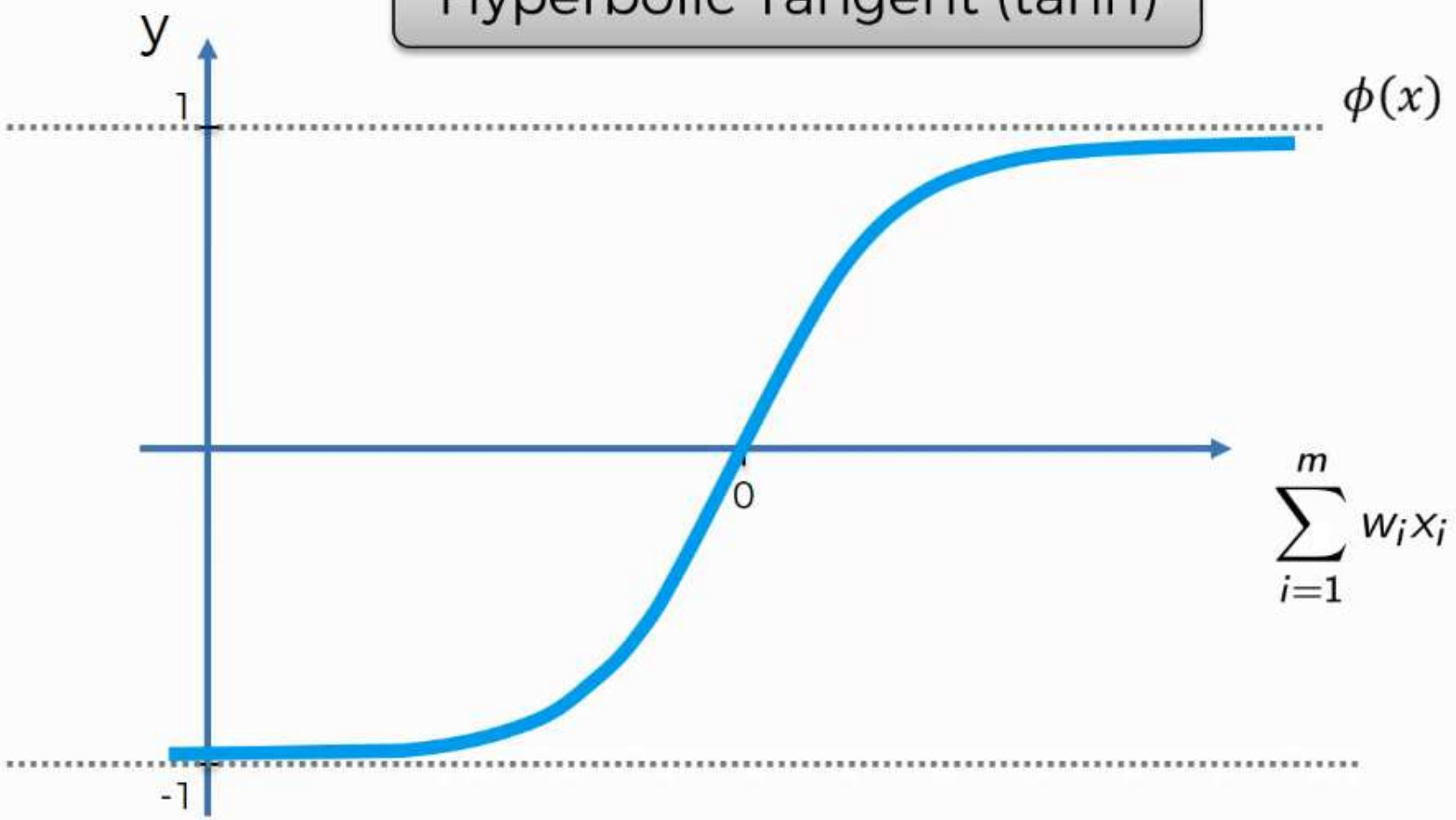
The Activation Function

Rectifier



The Activation Function

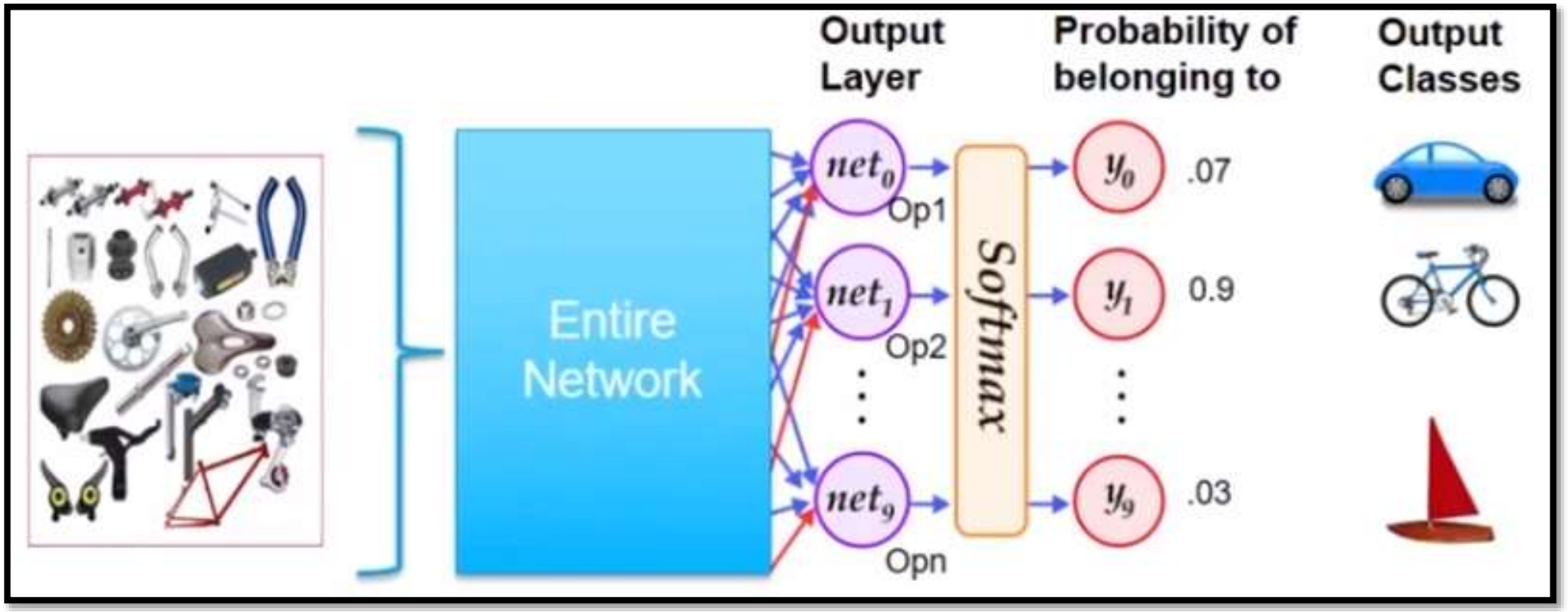
Hyperbolic Tangent (tanh)



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

SoftMax functions

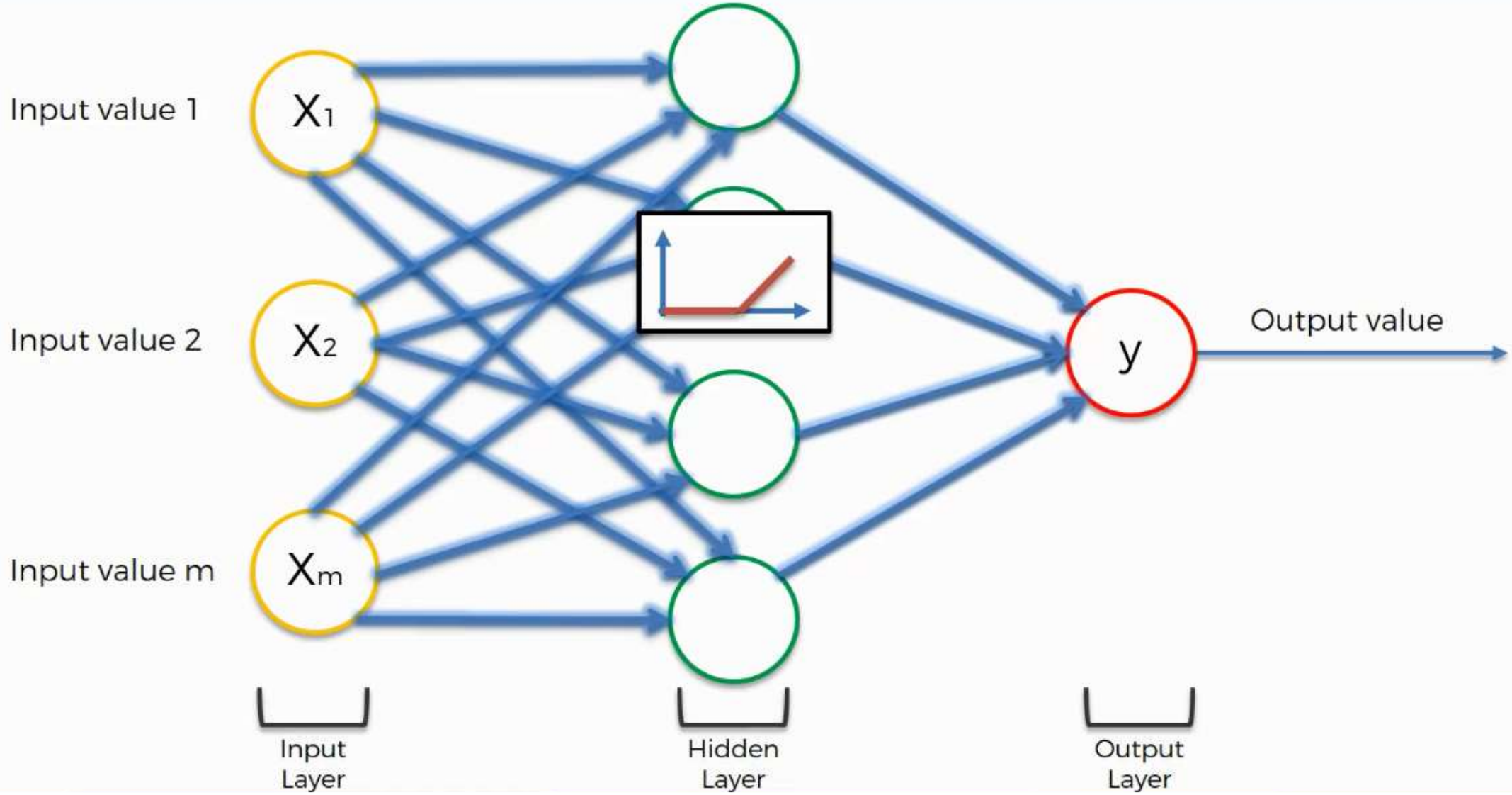
- Is to convert the output values of multiclass classification neural networks into probability values.



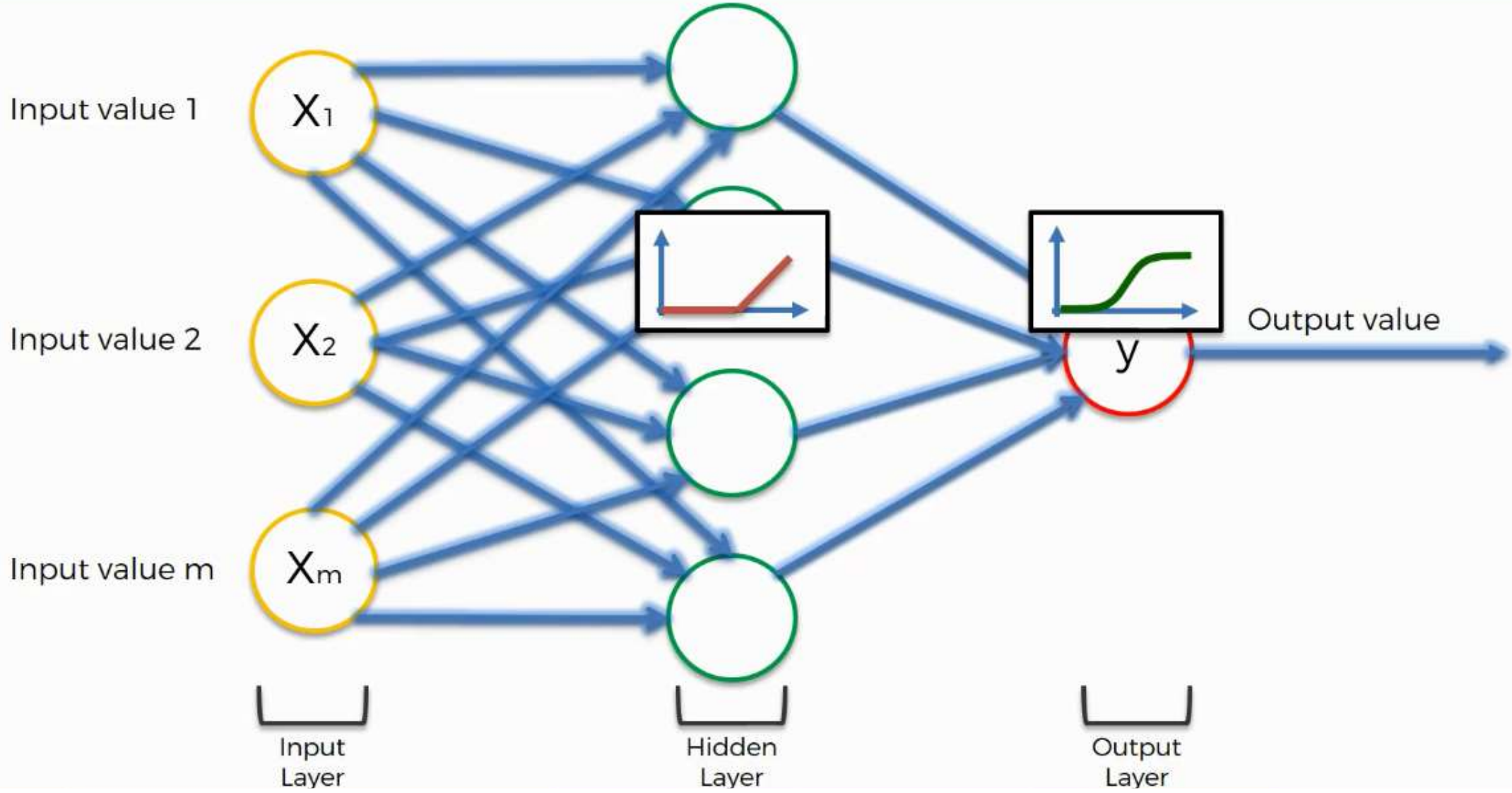
SoftMax

1. A kind of operation applied at the output neurons of a classifier network
2. Used only when we have two or more output neurons and is applied simultaneously to all the output neurons
3. Turns raw numbers coming out of the pen-ultimate layer into probability values in the output layer
4. Suppose output layer neurons emit (Op1, Op2, Opn). The raw numbers may not make much sense. We convert that into probabilities using Softmax which becomes more meaningful. For e.g. input belongs to cycle is 30 times more likely than sailboat, 13 times more probable than car

The Activation Function



The Activation Function



Summary

Note:

- We use relu activation function usually in the hidden layer.
- We use relu activation function in the output layer in the output is a Continuous number.
- We use sigmoid activation at the output layer if it is going to be either 1 or 0 (Two class classification)
- We use Softmax activation function at the output layer if it’s going to be (Multi-class classification)

Loss Function

Loss function

- Imagine the scenario, Once you developed your machine learning model that you believe, successfully identifying the cats and dogs but how do you know this is the best result?
- Here, we are looking for the metrics or a function that we can use to optimize our model performance. The loss function tells how good your model is in predictions. If the model predictions are closer to the actual values the Loss will be minimum and if the predictions are totally away from the original values the loss value will be the maximum.

$$Loss = abs(Y\_pred - Y\_actual)$$

On the basis of the Loss value, you can update your model until you get the best result.

Example Problem

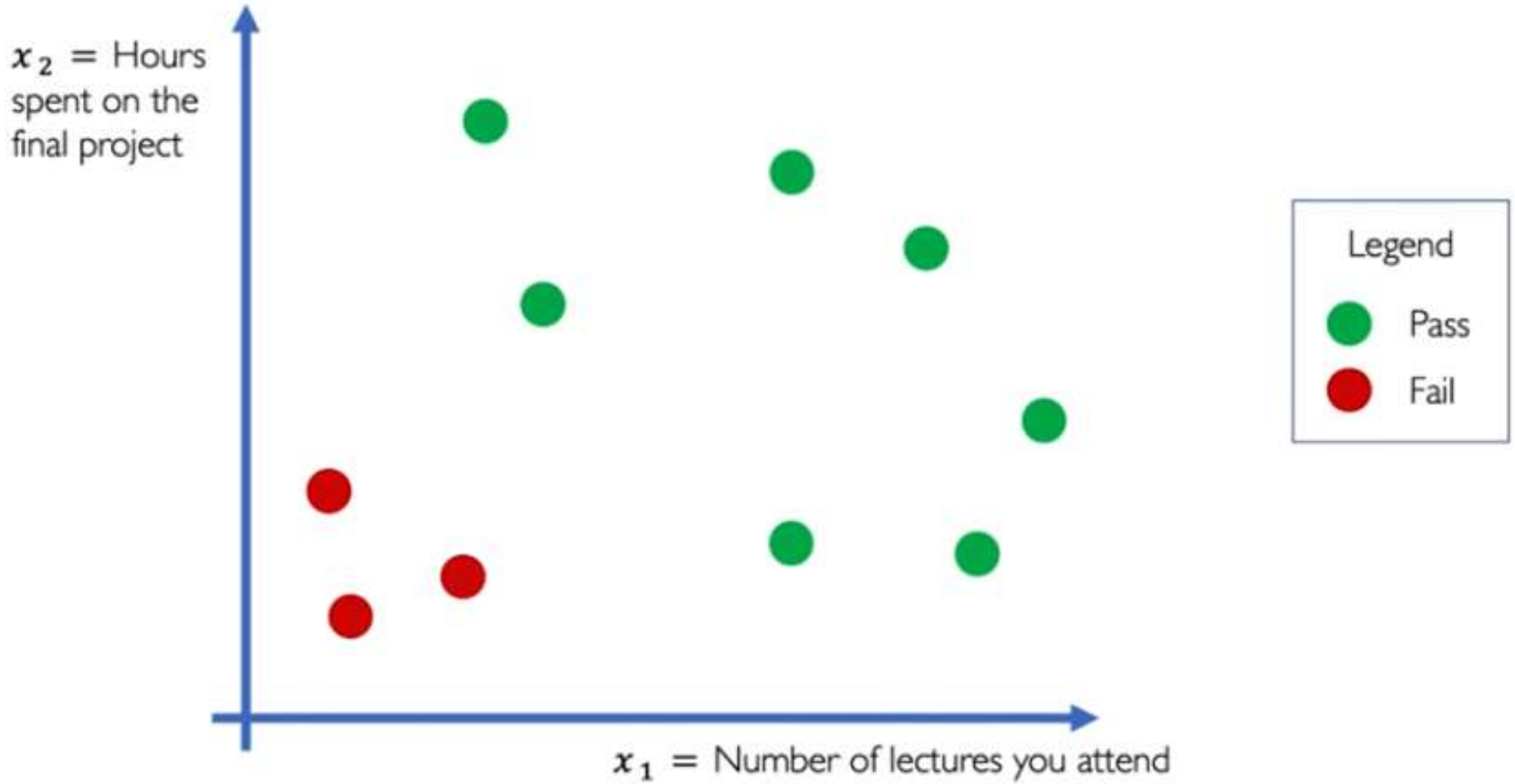
Will I pass this class?

Let's start with a simple two feature model

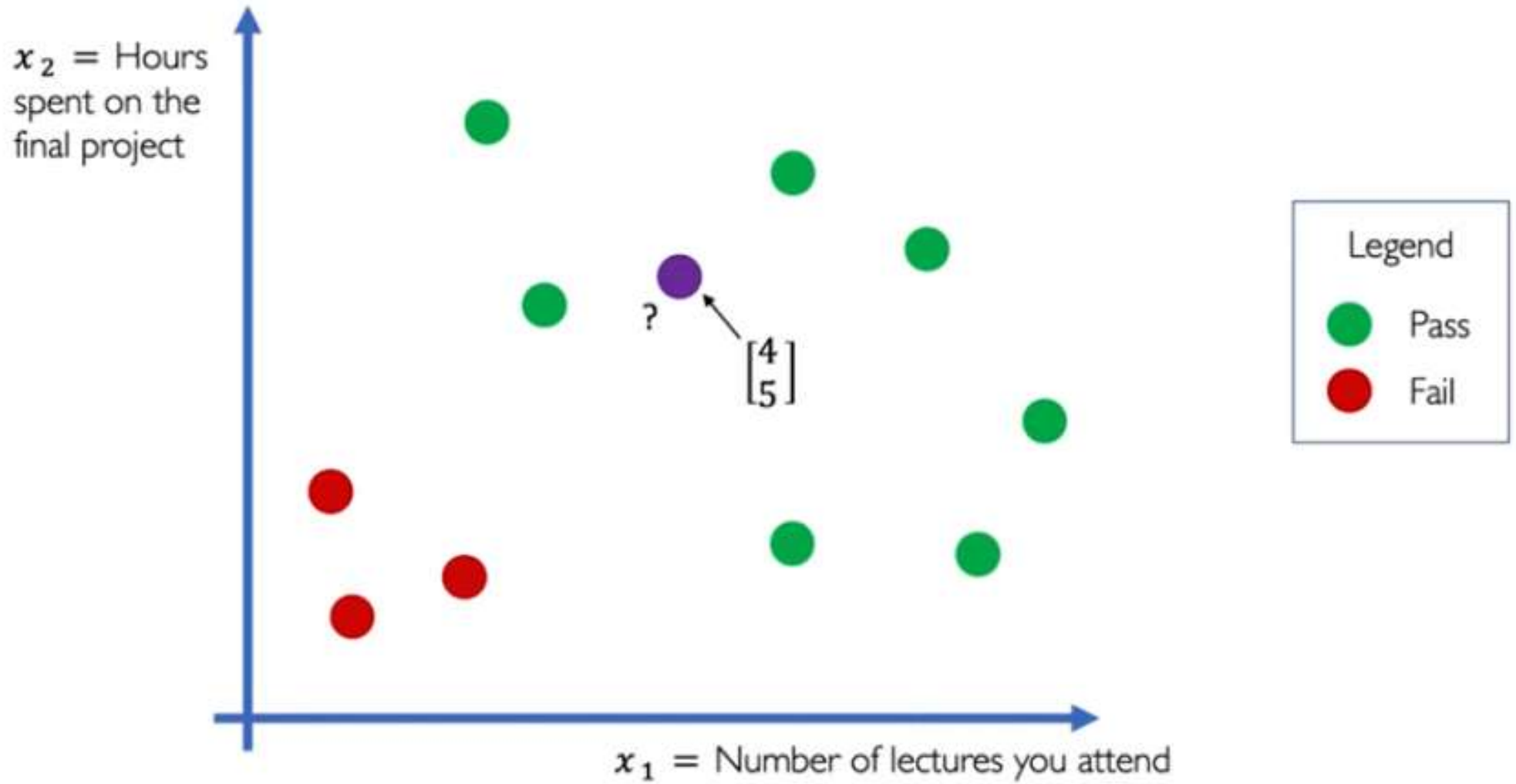
x_1 = Number of lectures you attend

x_2 = Hours spent on the final project

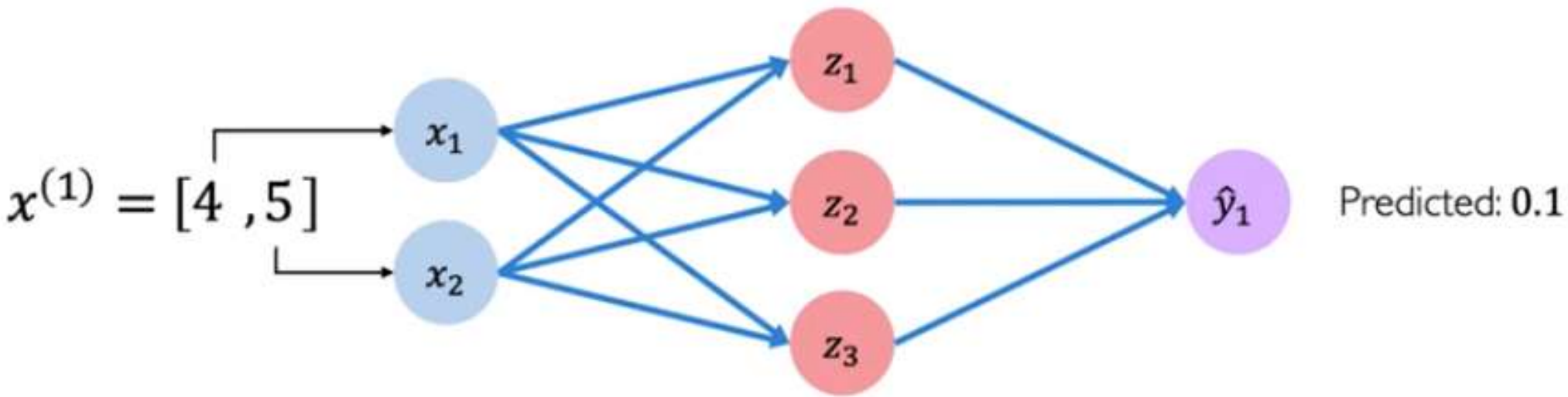
Example Problem: Will I pass this class?



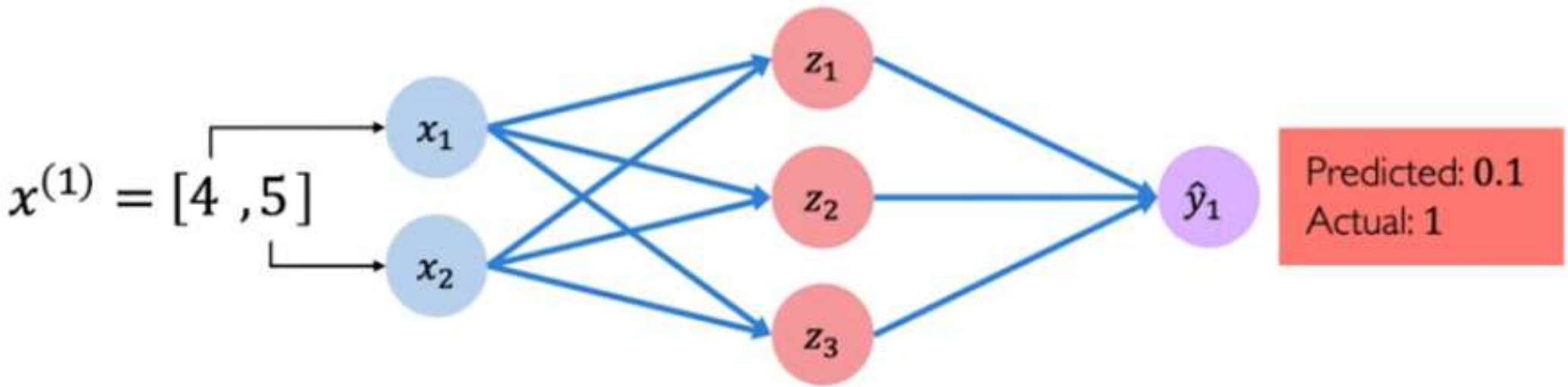
Example Problem: Will I pass this class?



Example Problem: Will I pass this class?



Example Problem: Will I pass this class?



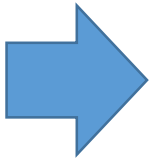
Binary Cross Entropy /Logs Loss

- Binary cross entropy compares each of the predicted probabilities to actual class output which can be either 0 or 1. It then calculates the score that penalizes the probabilities based on the distance from the expected value. That means how close or far from the actual value.
- Let’s first get a formal definition of binary cross-entropy

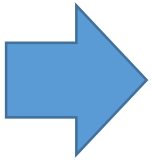
Definition: Binary Cross Entropy is the negative average of the log of corrected predicted probabilities.

Binary Cross Entropy /Logs Loss

ID	Actual	Predicted probabilities
ID6	1	0.94
ID1	1	0.90
ID7	1	0.78
ID8	0	**0.56**
ID2	0	**0.51**
ID3	1	0.47
ID4	1	0.32
ID5	0	**0.10**



ID	Actual	Predicted probabilities	Corrected Probabilities
ID6	1	0.94	0.94
ID1	1	0.90	0.90
ID7	1	0.78	0.78
ID8	0	**0.56**	**0.44**
ID2	0	**0.51**	**0.49**
ID3	1	0.47	0.47
ID4	1	0.32	0.32
ID5	0	**0.10**	**0.90**



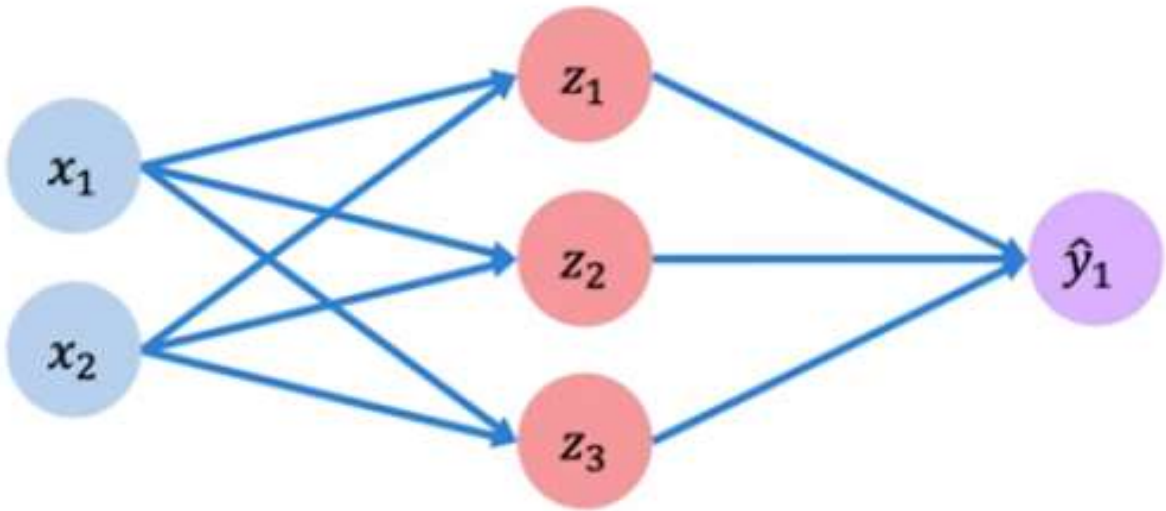
ID	Actual	Predicted probabilities	Corrected Probabilities	Log
ID6	1	0.94	0.94	-0.0268721464
ID1	1	0.90	0.90	-0.0457574906
ID7	1	0.78	0.78	-0.1079053973
ID8	0	0.56	0.44	-0.3565473235
ID2	0	0.51	0.49	-0.30980392
ID3	1	0.47	0.47	-0.3279021421
ID4	1	0.32	0.32	-0.4948500217
ID5	0	0.10	0.90	-0.0457574906

$$-\frac{1}{N}\sum_{i=1}^{N}\left(\log\left(p_i\right)\right)$$

Binary Cross Entropy Loss

Cross entropy loss *can be used with models that output a probability between 0 and 1*

$$\boldsymbol{X} = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$



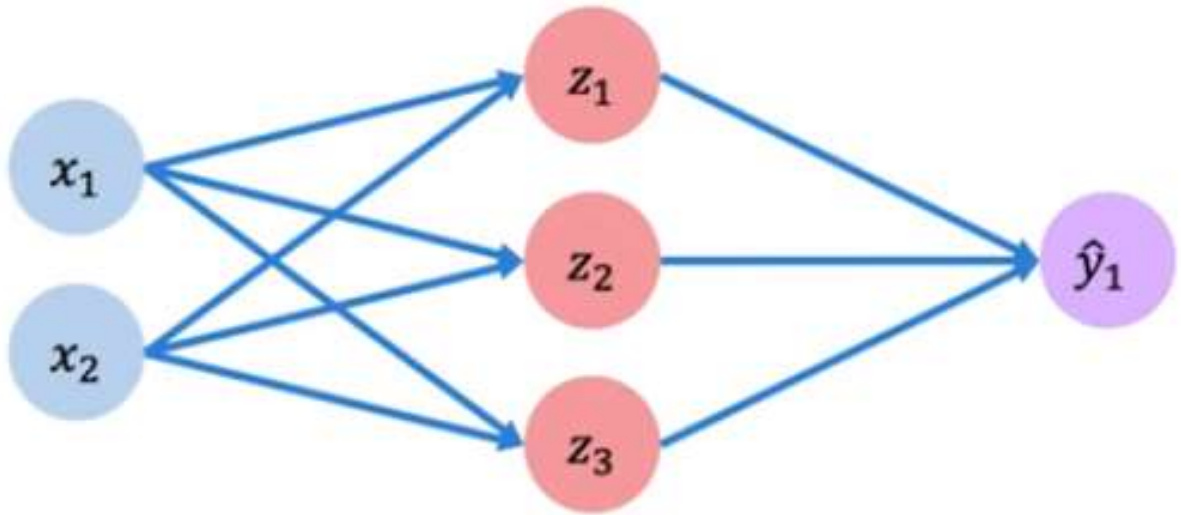
$f(x)$ y

$$\begin{bmatrix} 0.1 \\ 0.8 \\ 0.6 \\ \vdots \end{bmatrix} \begin{matrix} \times \\ \times \\ \checkmark \\ \end{matrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$

Mean Squared Error Loss

Mean squared error loss *can be used with regression models that output continuous real numbers*

$$\boldsymbol{X} = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$



$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - f\left(x^{(i)}; \boldsymbol{W}\right)\right)^2$$

Actual Predicted

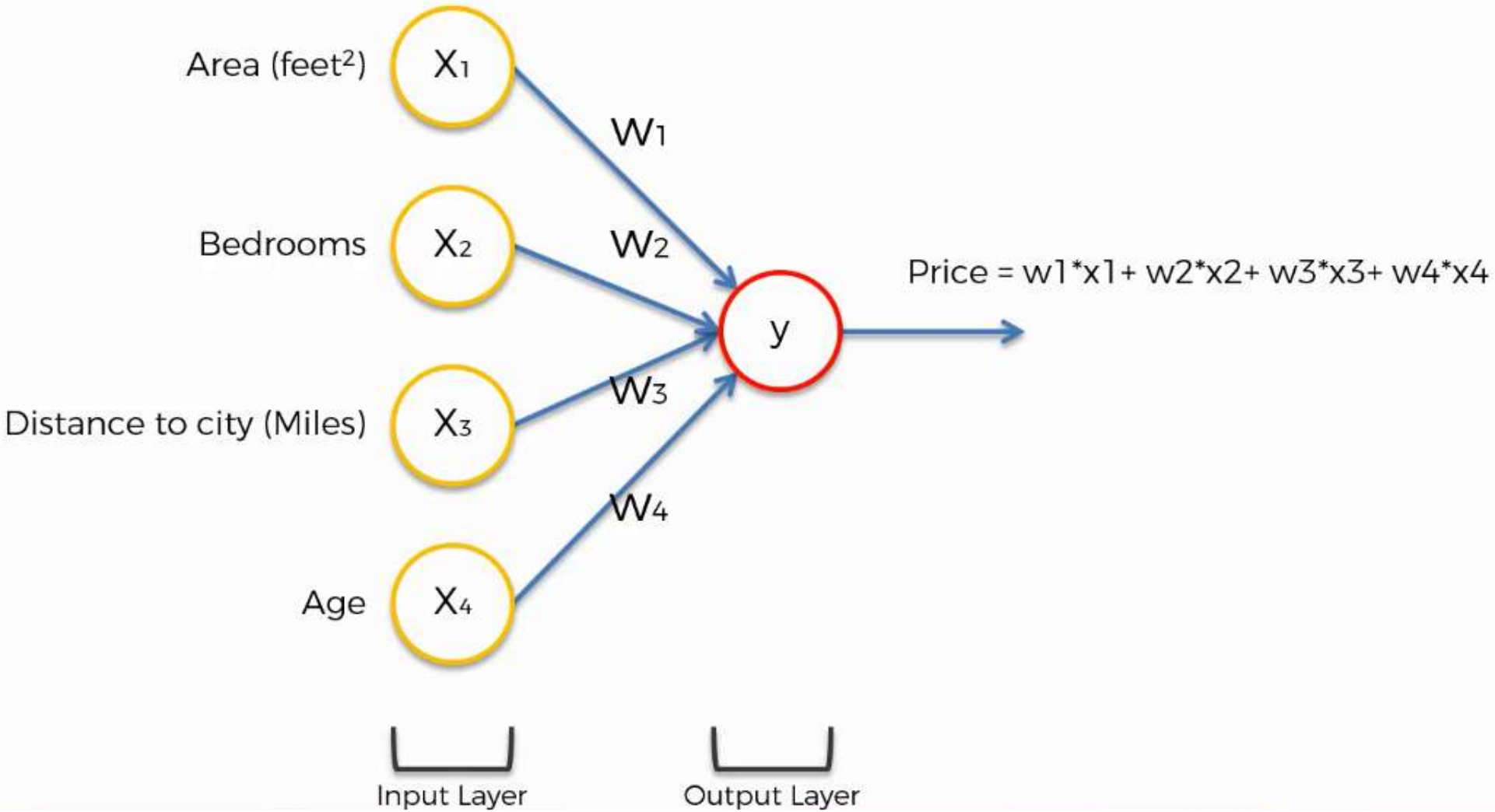
$f(x)$ y

$$\begin{bmatrix} 30 \\ 80 \\ 85 \\ \vdots \end{bmatrix} \begin{matrix} \times \\ \times \\ \checkmark \\ \end{matrix} \begin{bmatrix} 90 \\ 20 \\ 95 \\ \vdots \end{bmatrix}$$

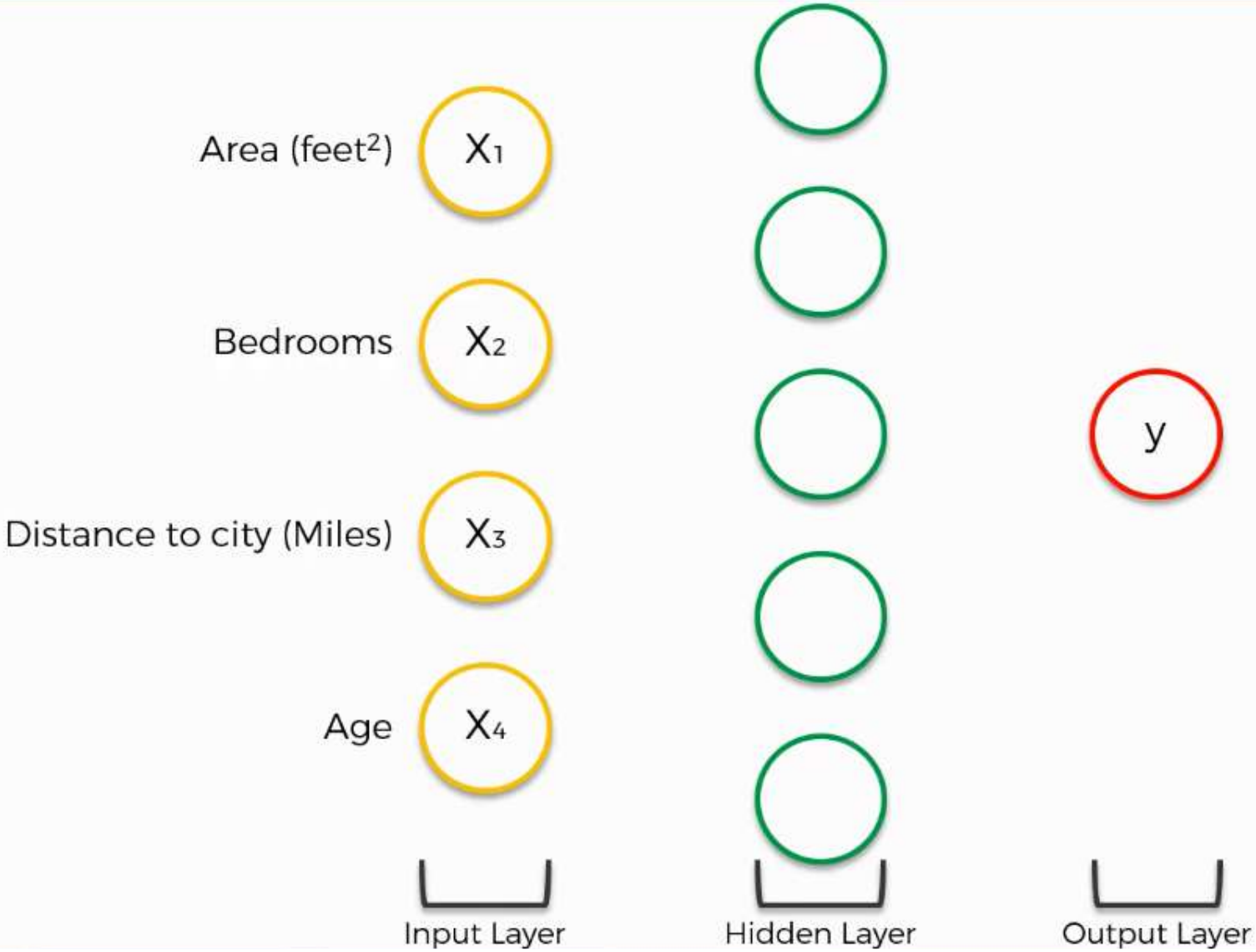
Final Grades (percentage)

Example :
Let’s see how do ANN works?

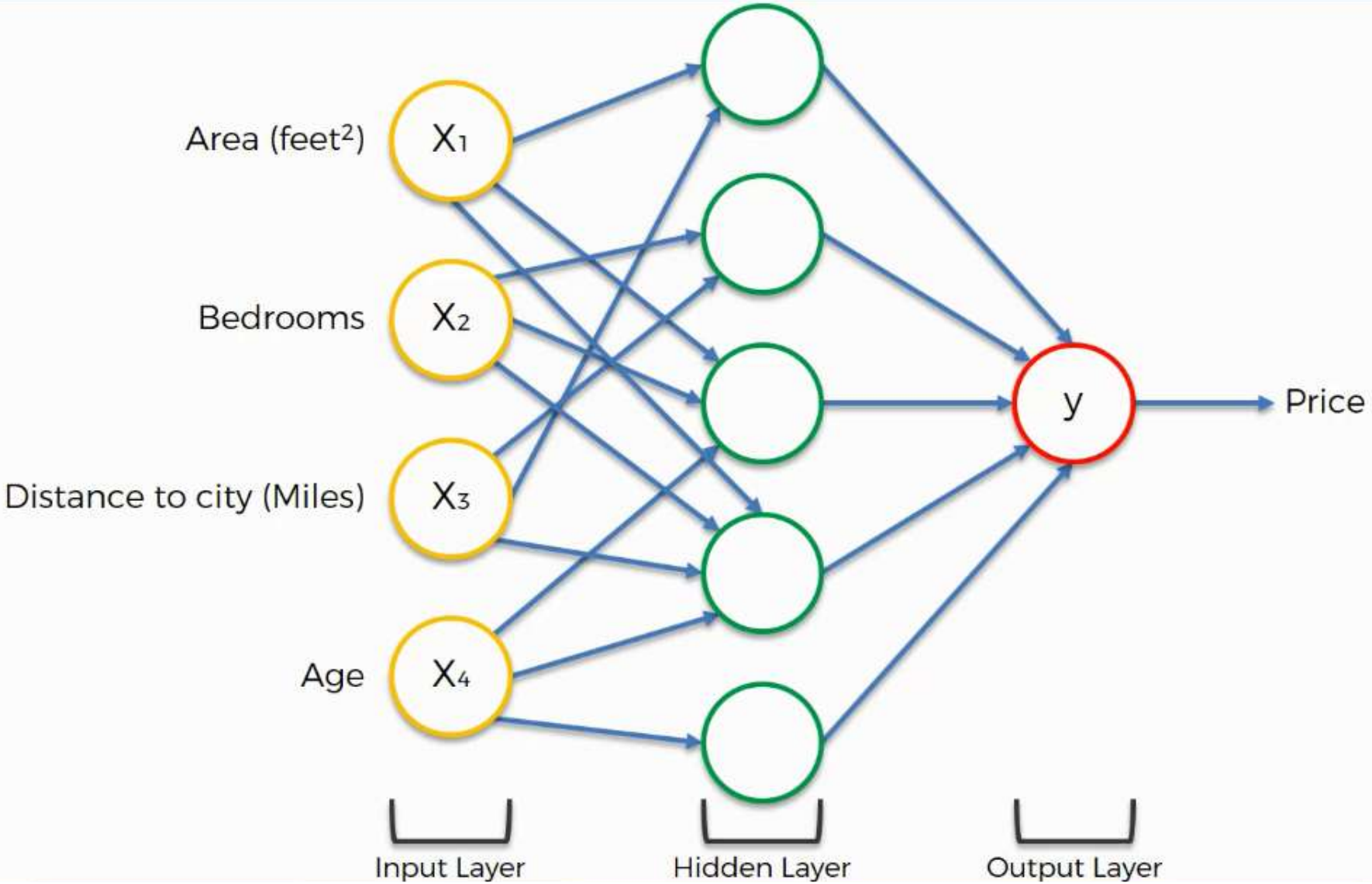
How Do Neural Networks Work?



How Do Neural Networks Work?

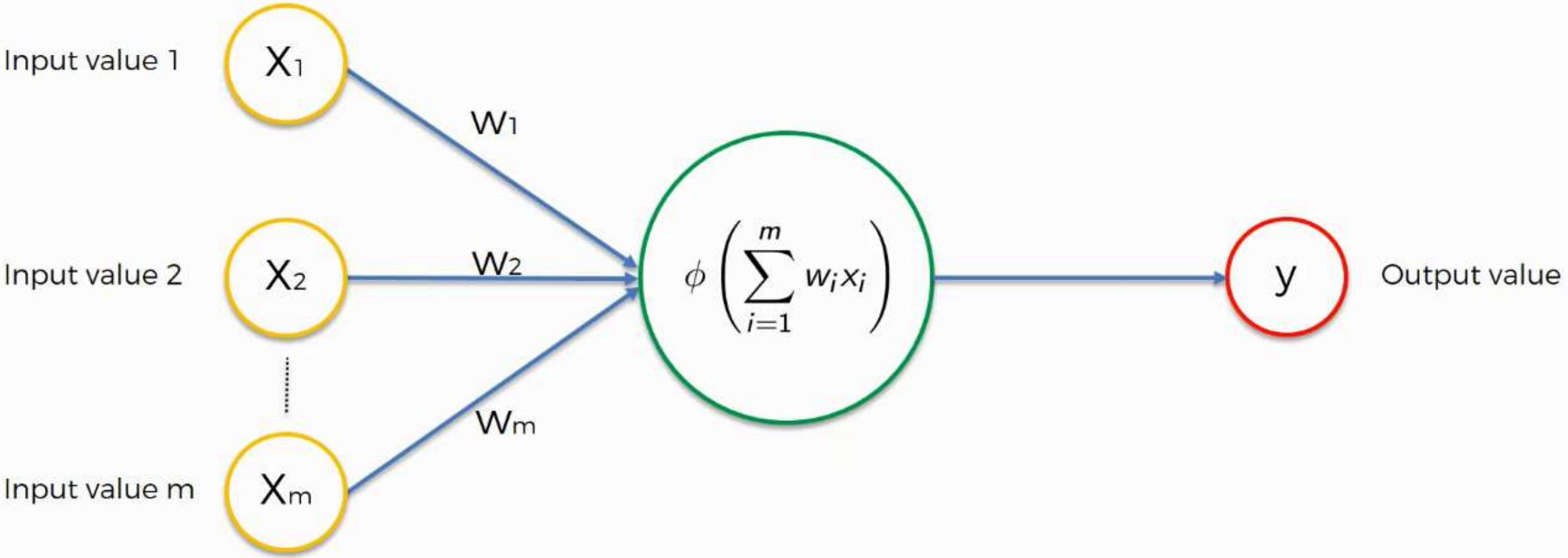


How Do Neural Networks Work?

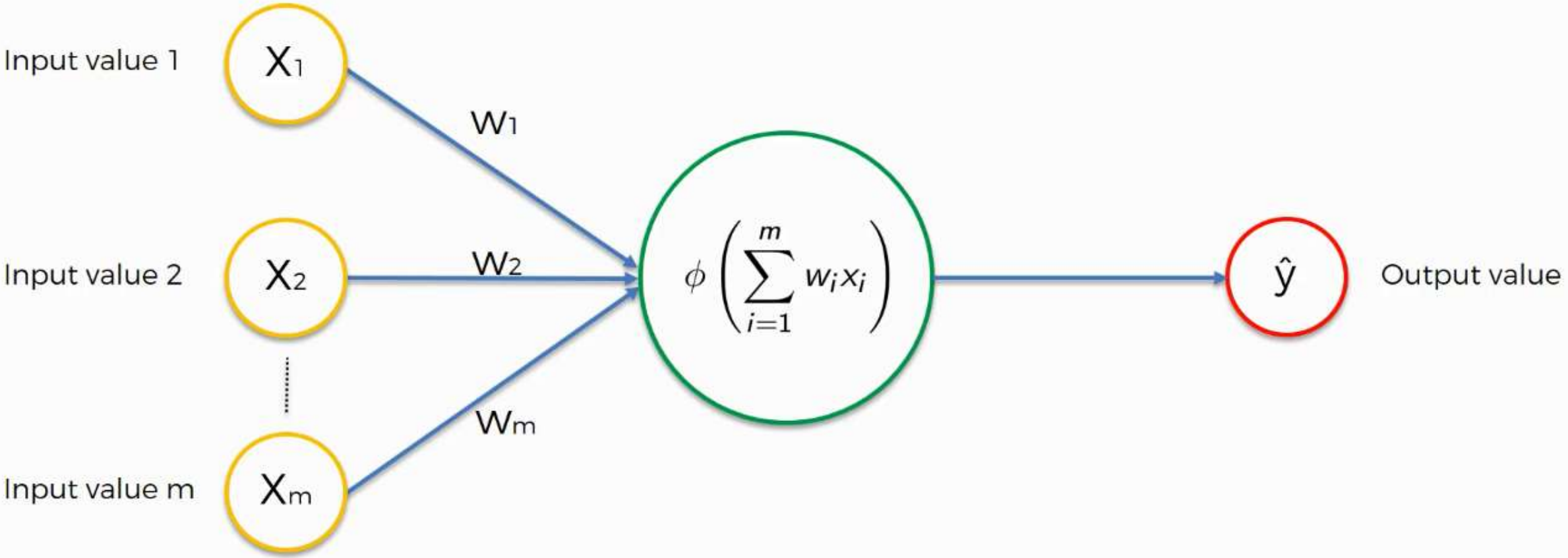


How do NNs Learn?

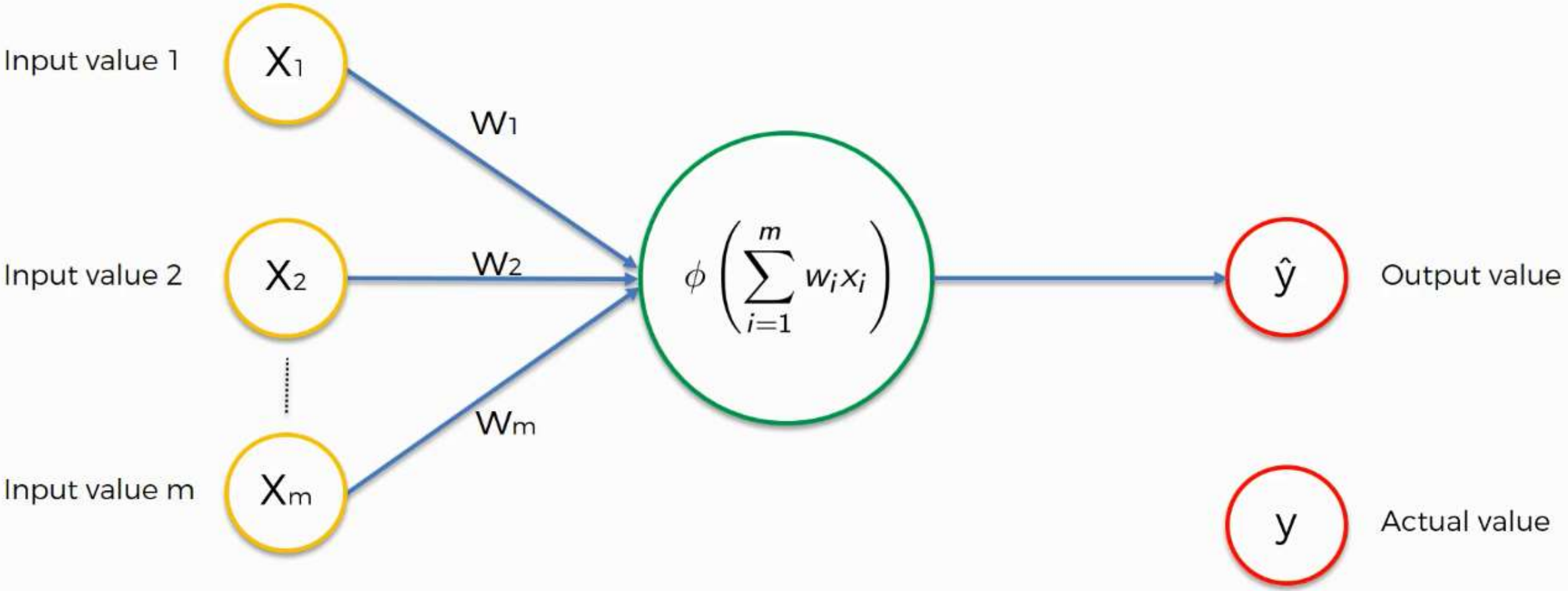
How do Neural Networks learn?



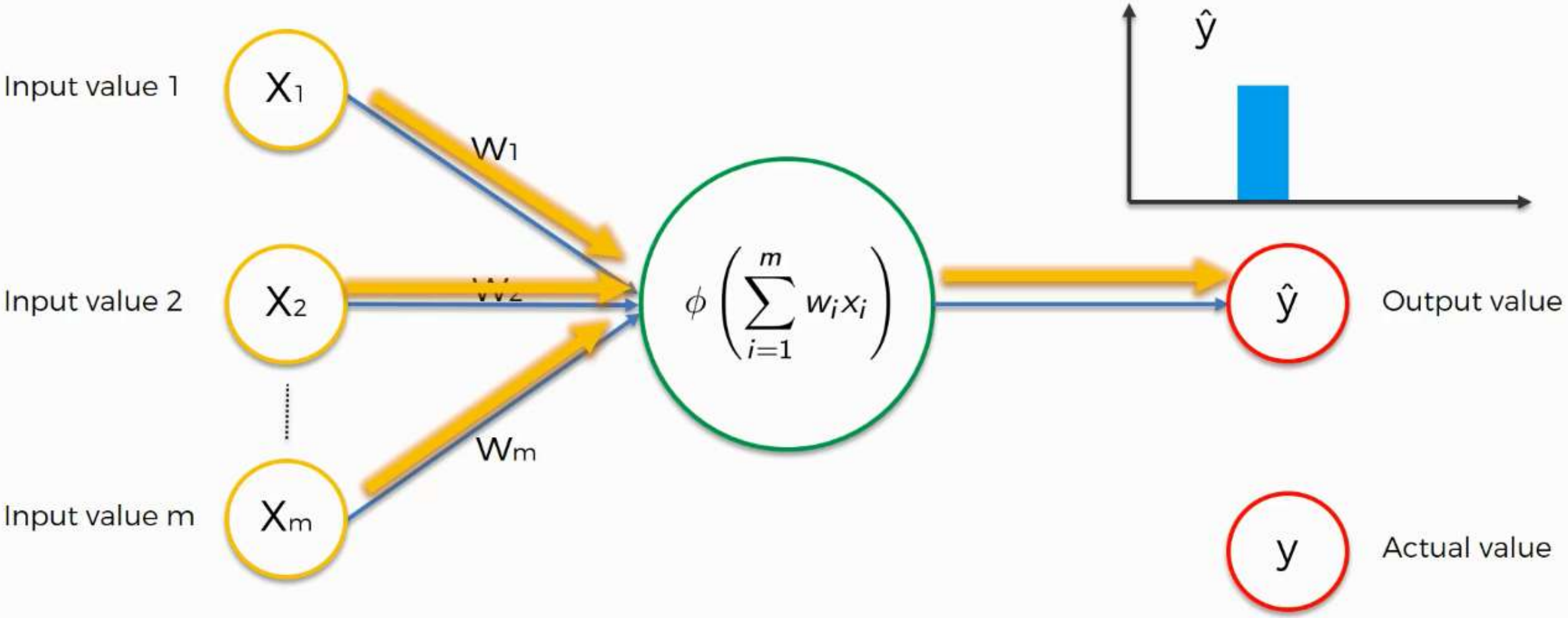
How do Neural Networks learn?



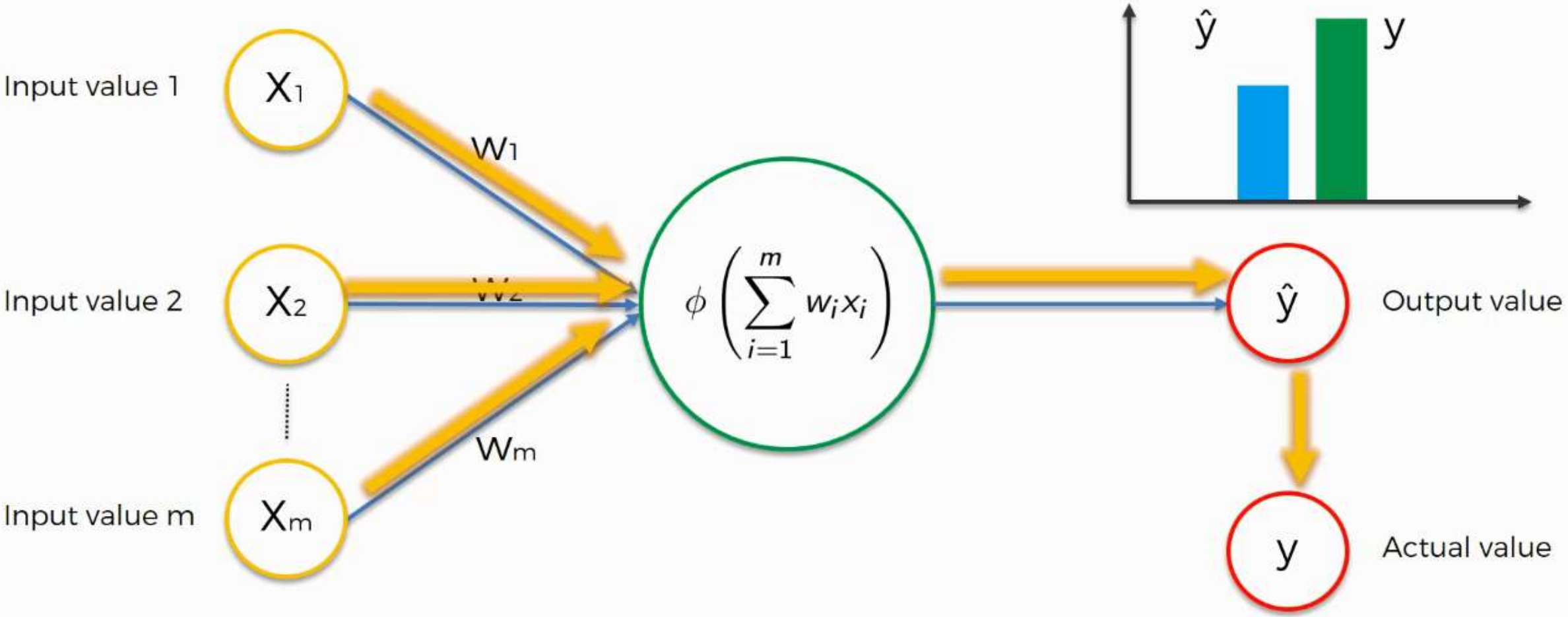
How do Neural Networks learn?



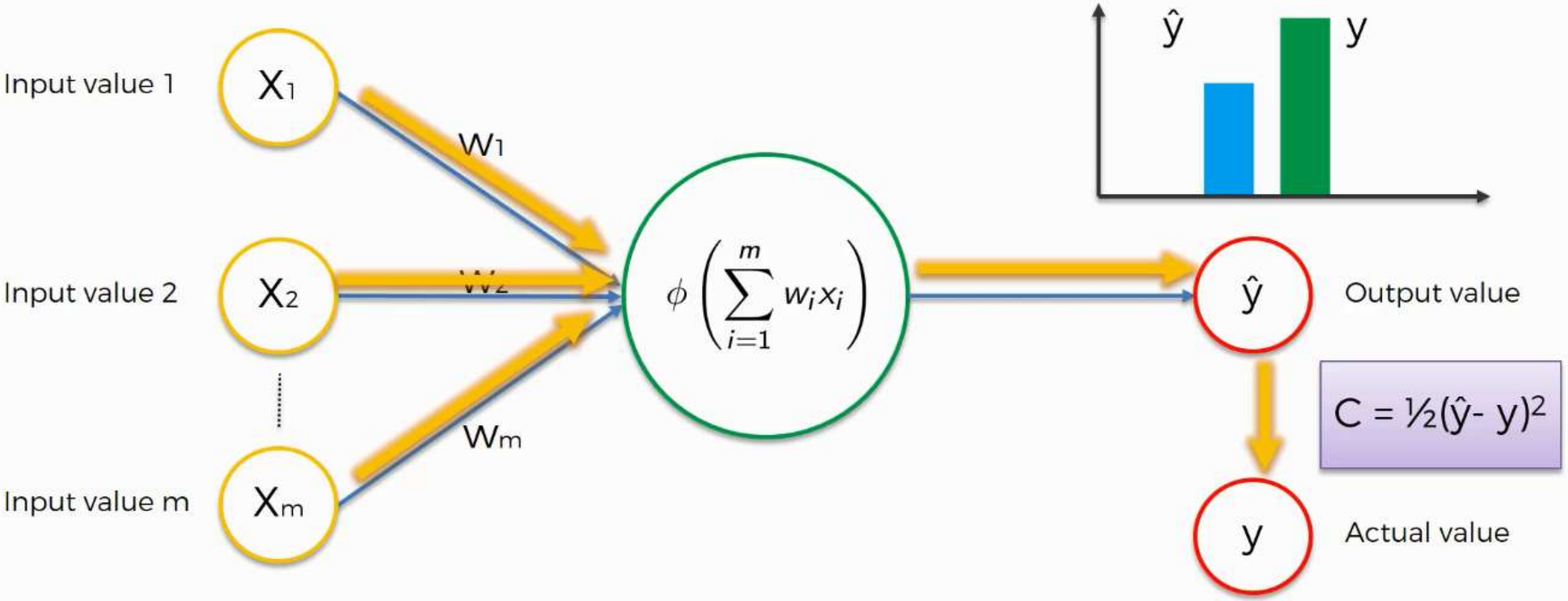
How do Neural Networks learn?



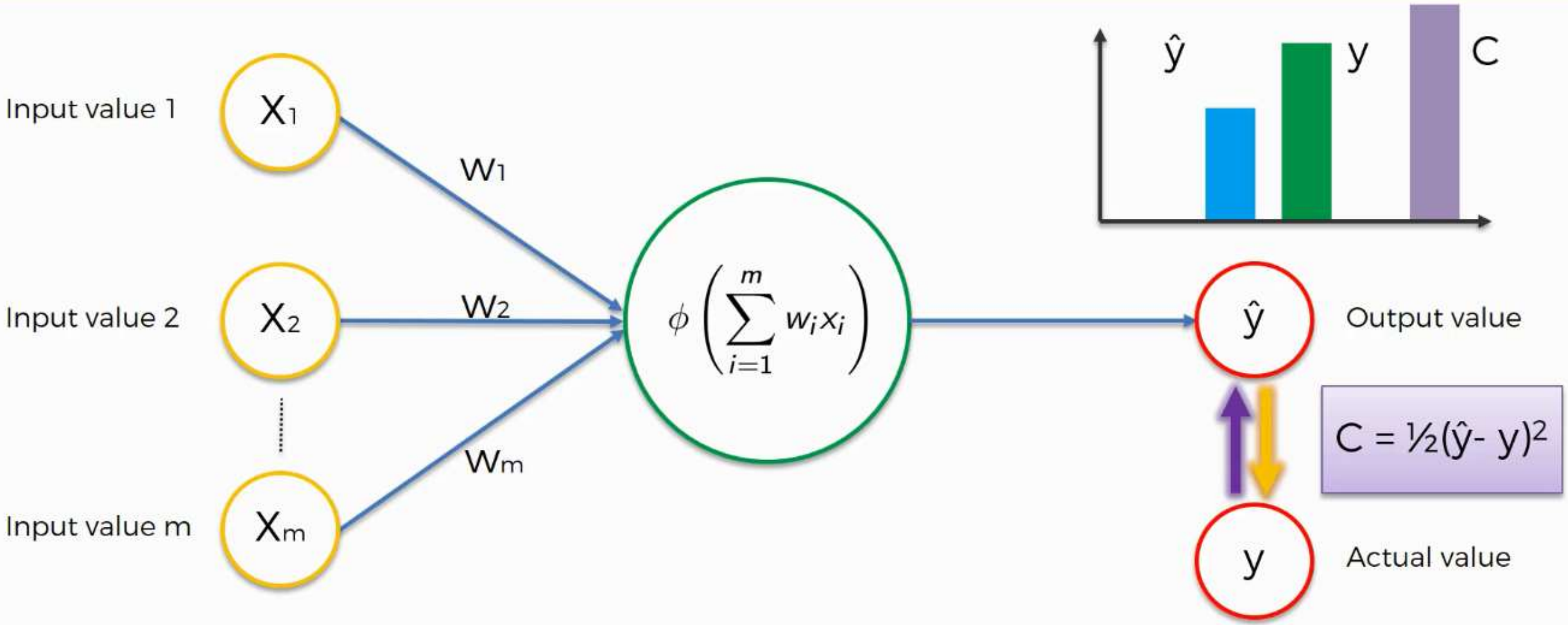
How do Neural Networks learn?



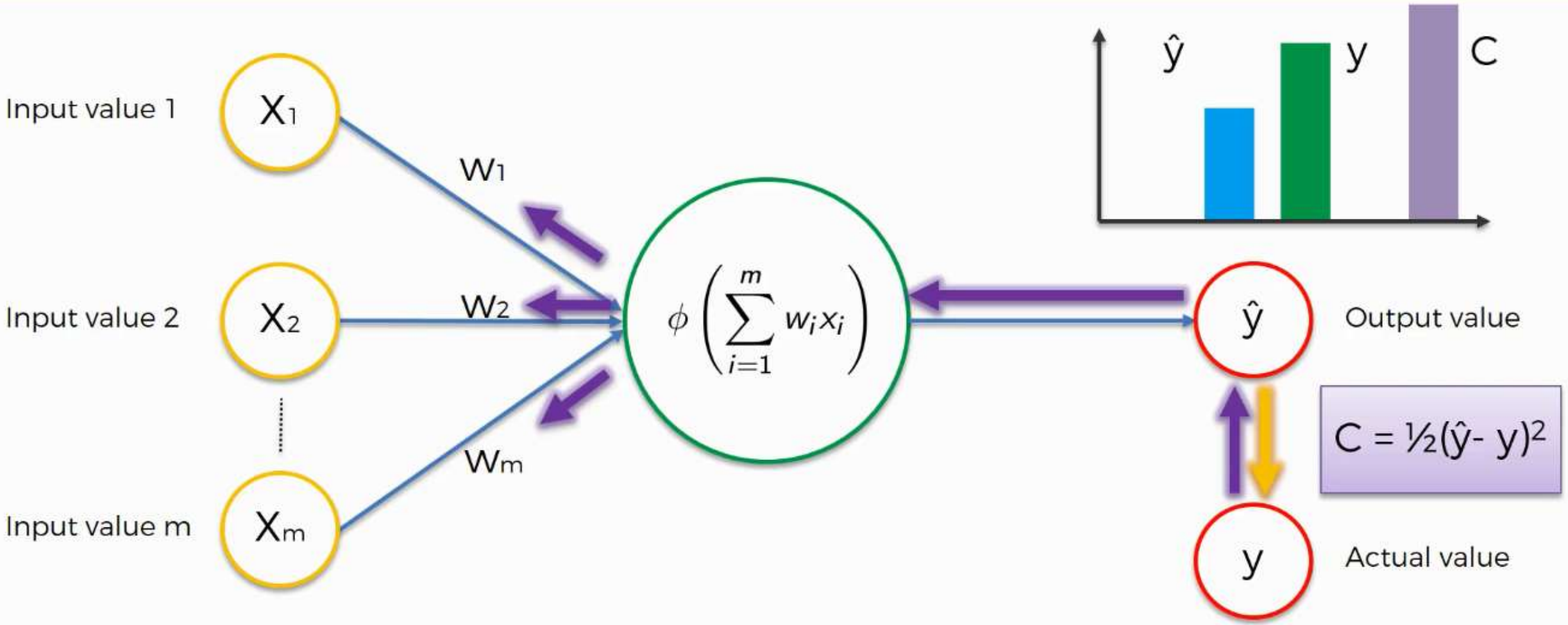
How do Neural Networks learn?



How do Neural Networks learn?



How do Neural Networks learn?



How do Neural Networks learn?

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%

Input value 1 X_1

Input value 2 X_2

Input value m X_m

W_1

W_2

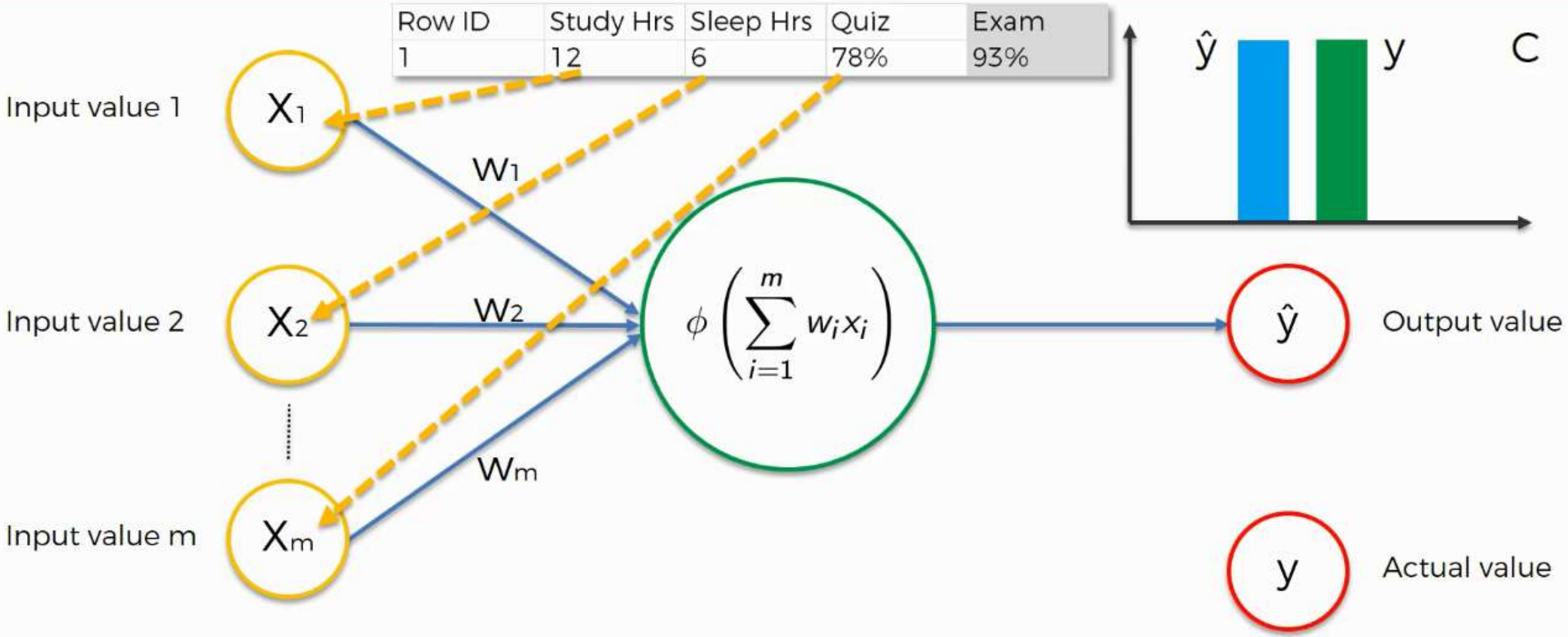
W_m

$$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$$

$\hat{y}$ Output value

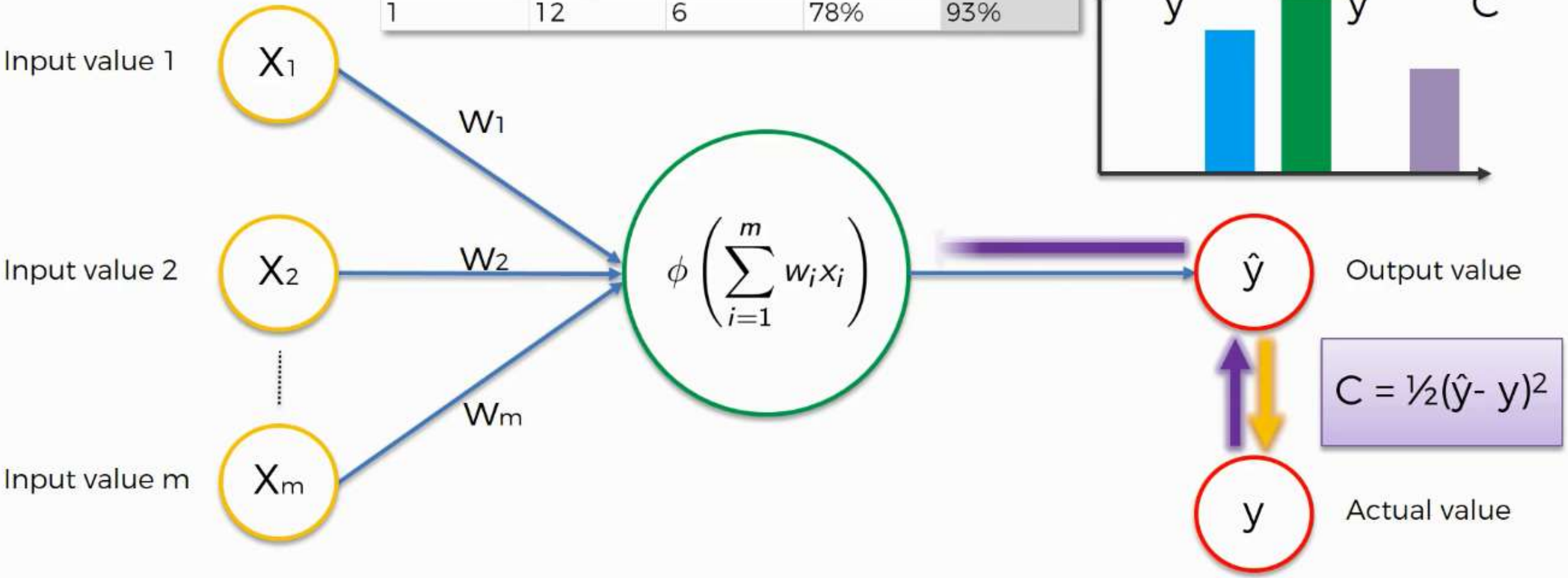
y Actual value

How do Neural Networks learn?



How do Neural Networks learn?

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%



How do Neural Networks learn?

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%

Input value 1 X_1

Input value 2 X_2

Input value m X_m

W_1

W_2

W_m

$$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$$

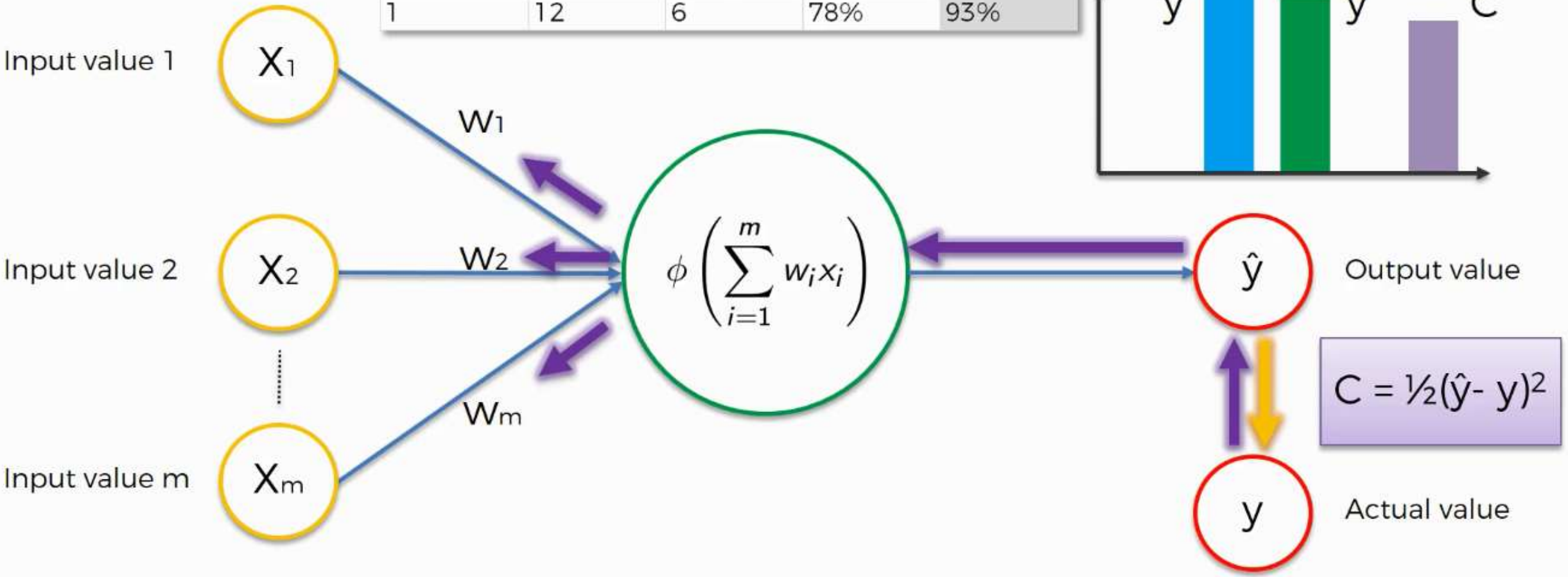
$\hat{y}$ Output value

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

y Actual value

How do Neural Networks learn?

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%



How do Neural Networks learn?

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%

Input value 1 X_1

Input value 2 X_2

Input value m X_m

W_1

W_m

$$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$$

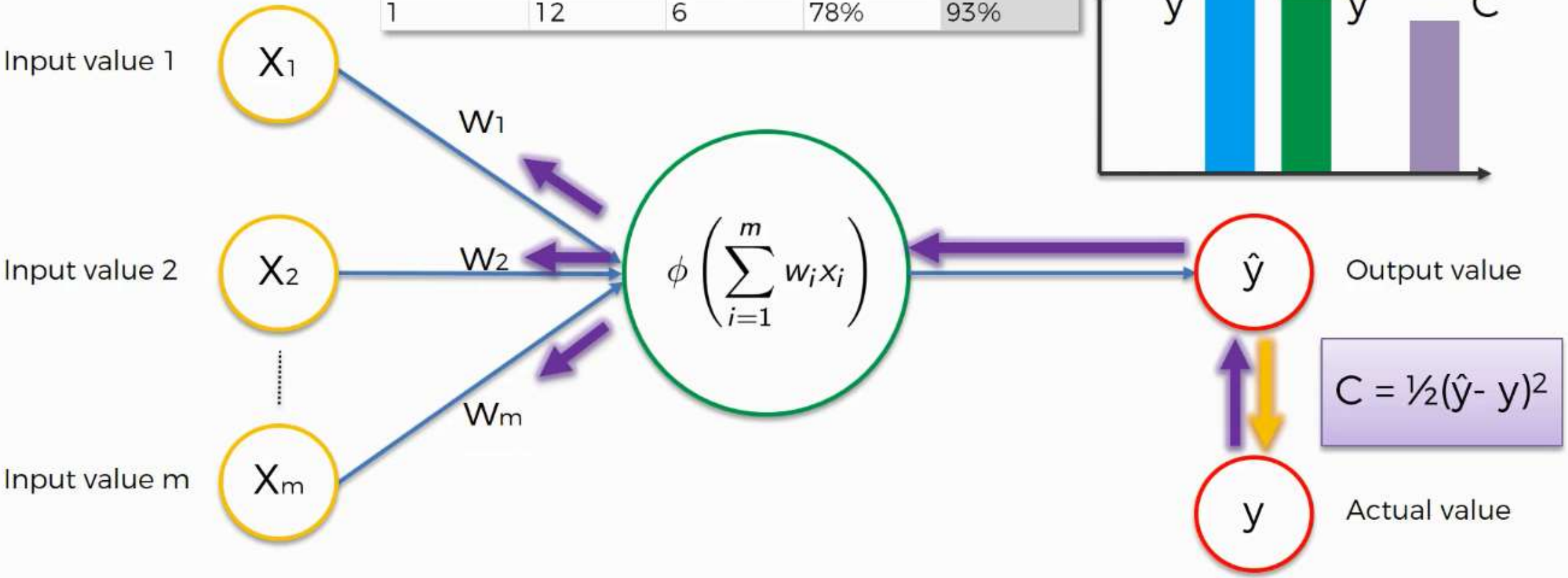
$\hat{y}$ Output value

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

y Actual value

How do Neural Networks learn?

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%

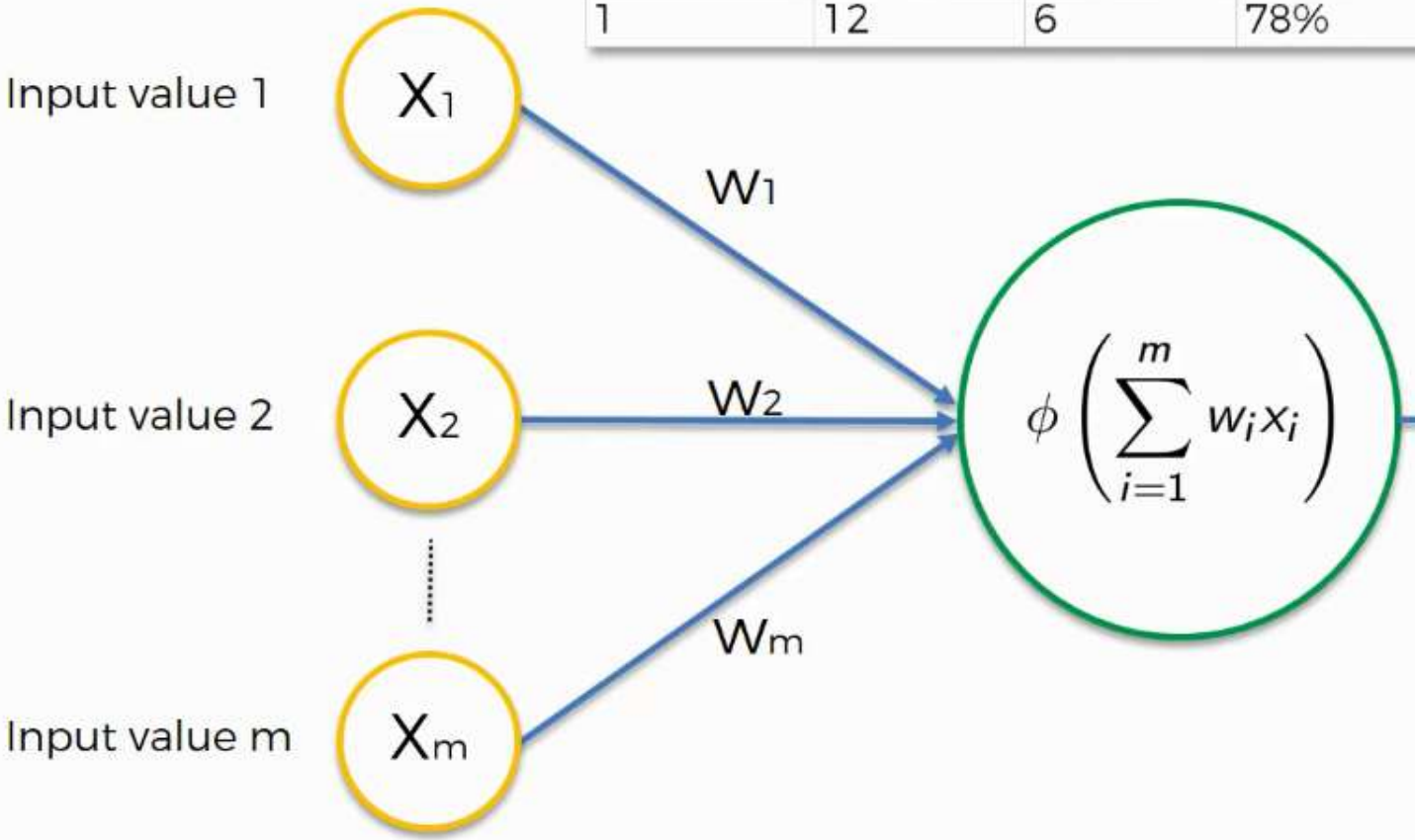


How do Neural Networks learn?

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%

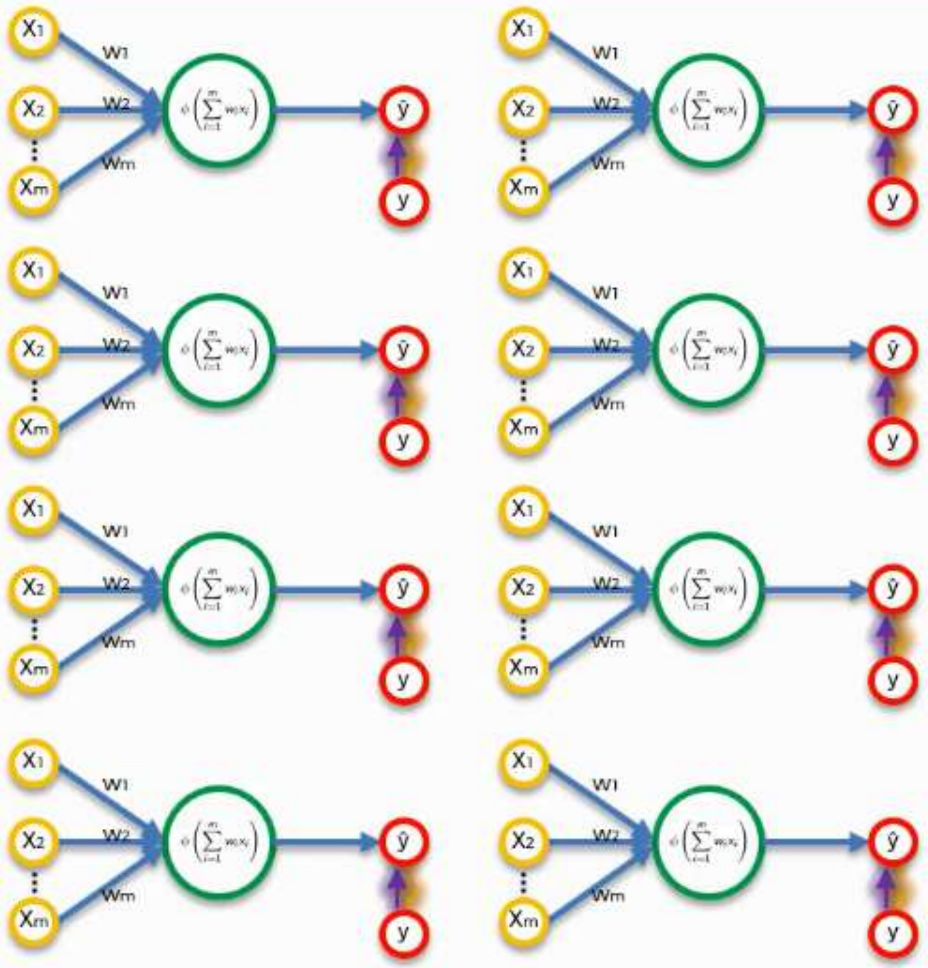
How do Neural Networks learn?

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%



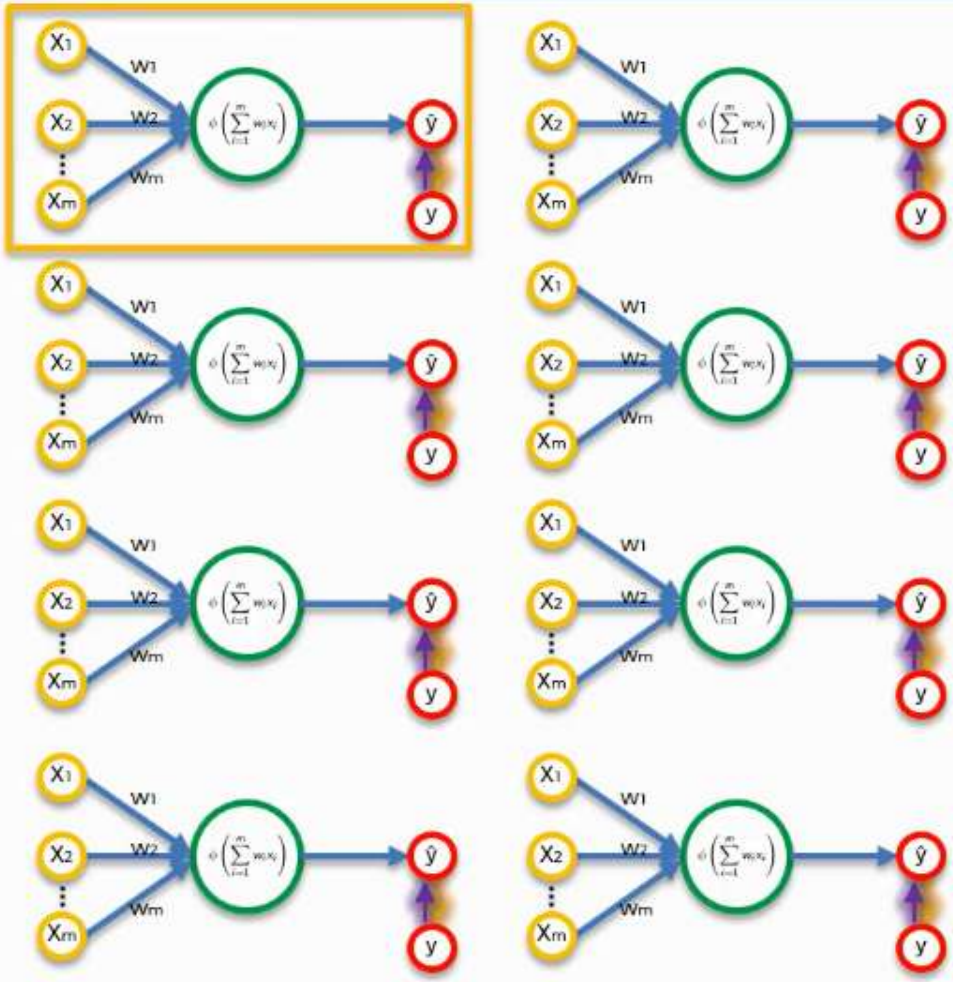
$C = \frac{1}{2}(\hat{y} - y)^2$

How do Neural Networks learn?



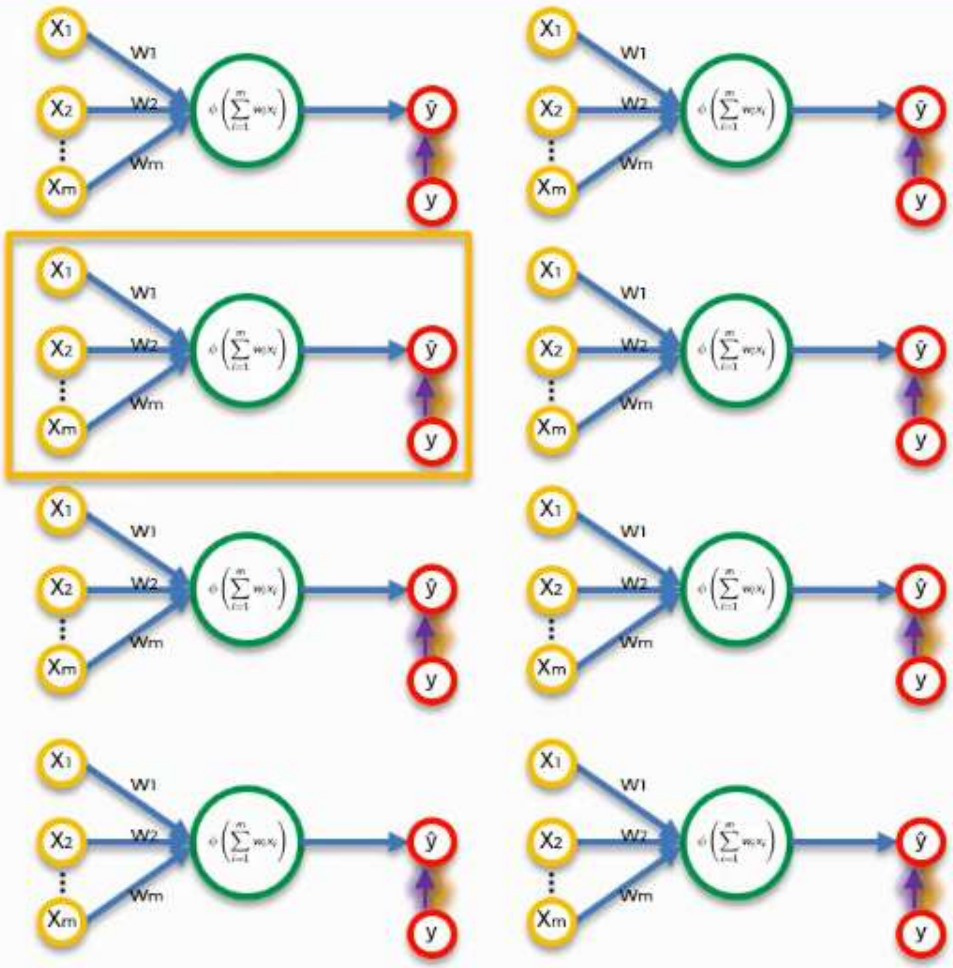
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

How do Neural Networks learn?



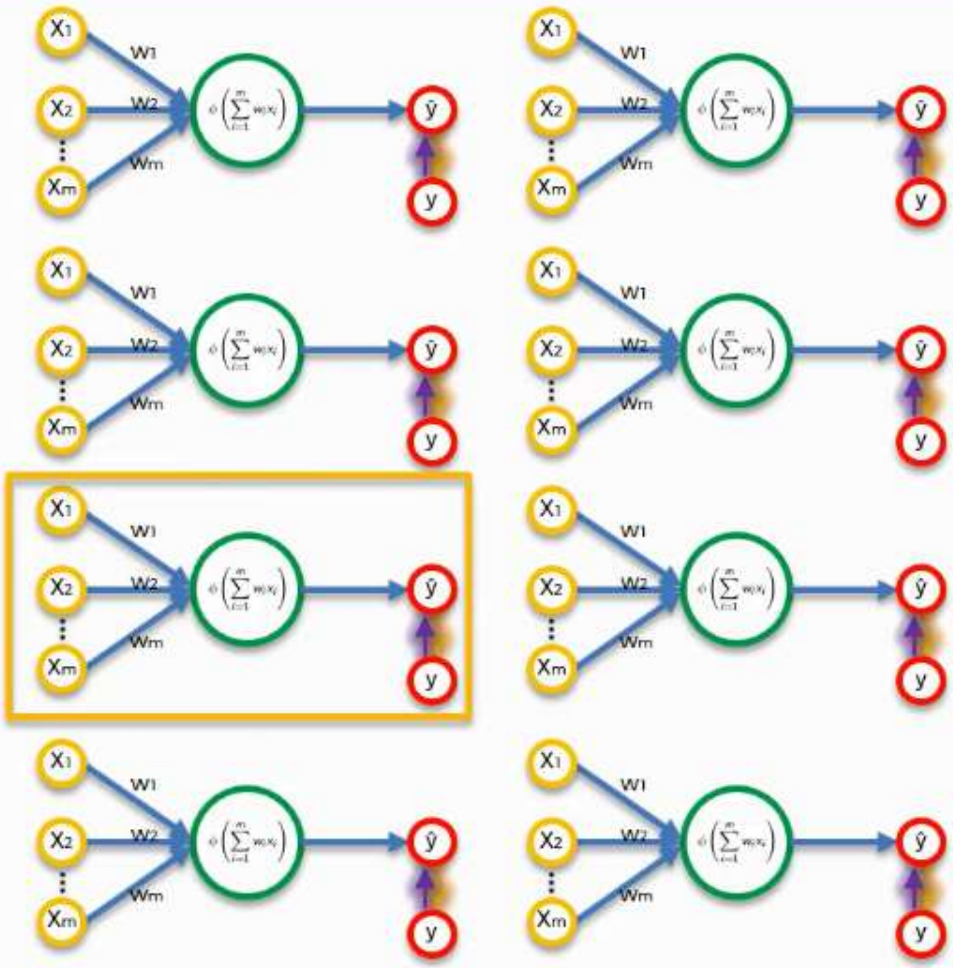
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

How do Neural Networks learn?



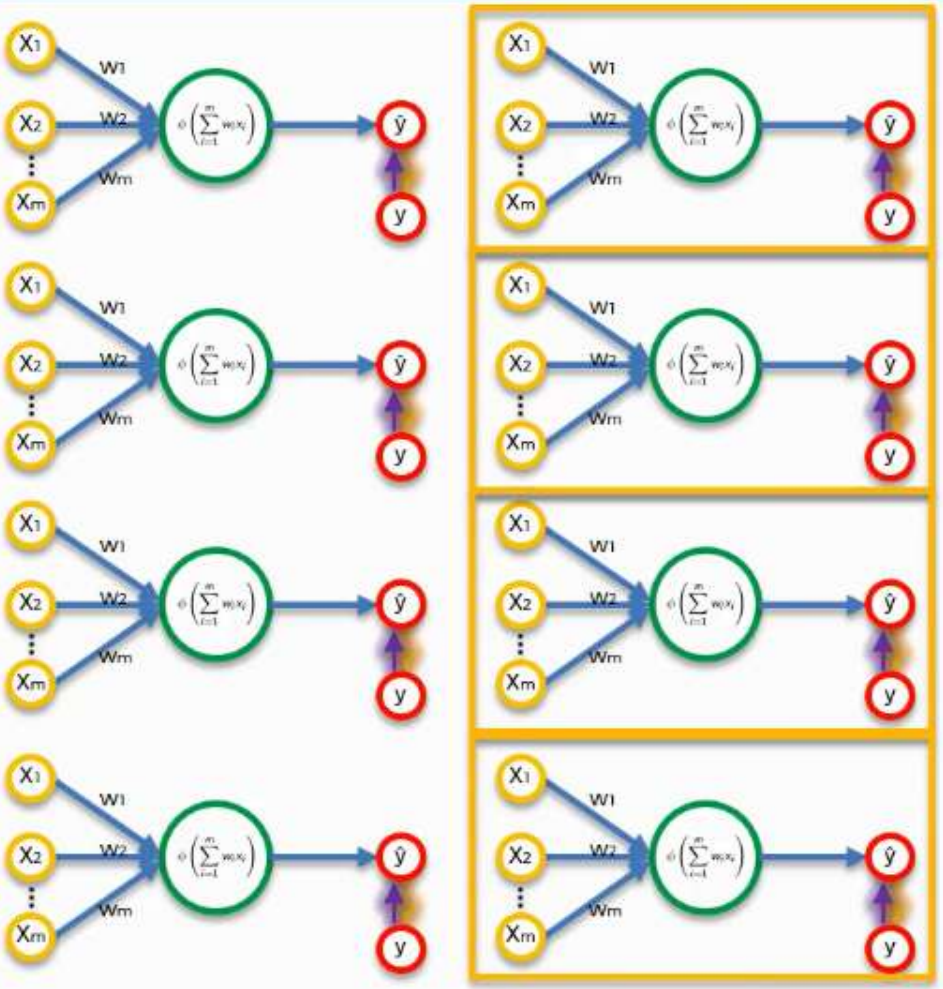
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

How do Neural Networks learn?

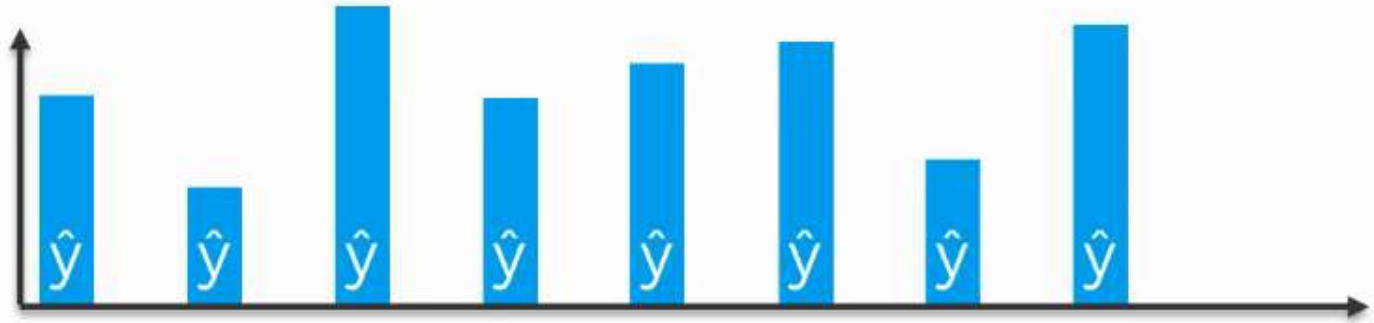


Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

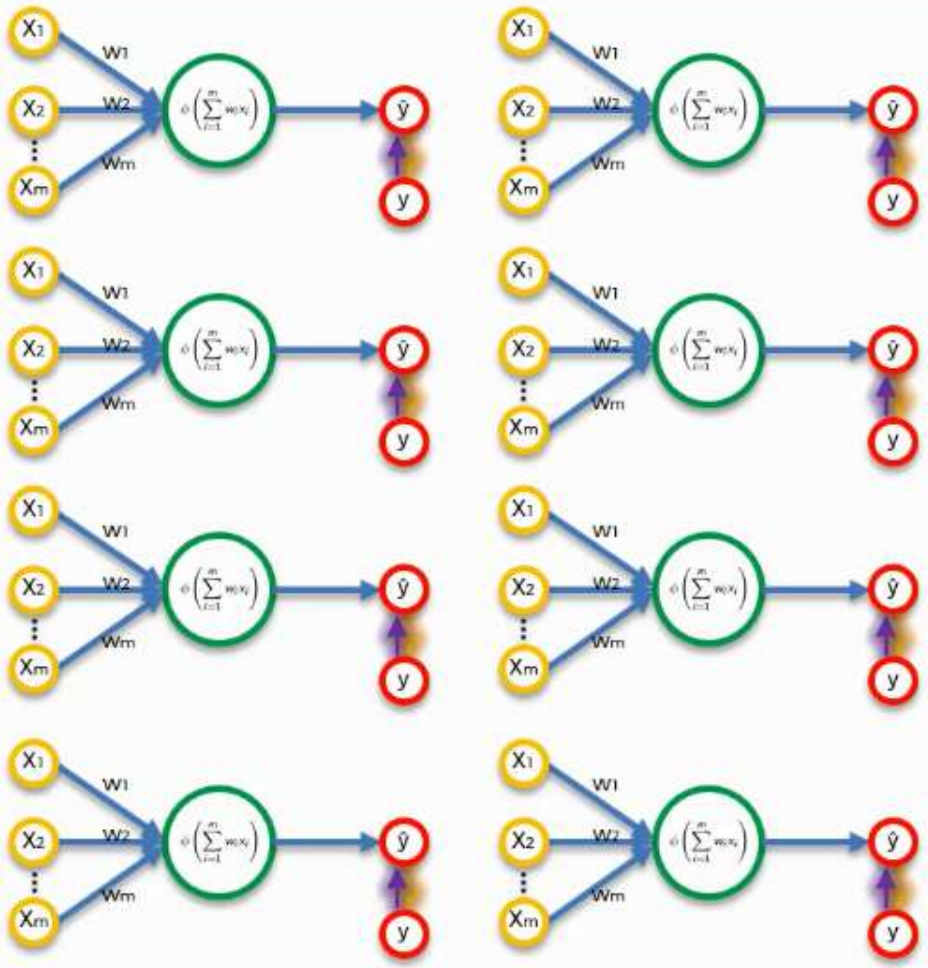
How do Neural Networks learn?



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

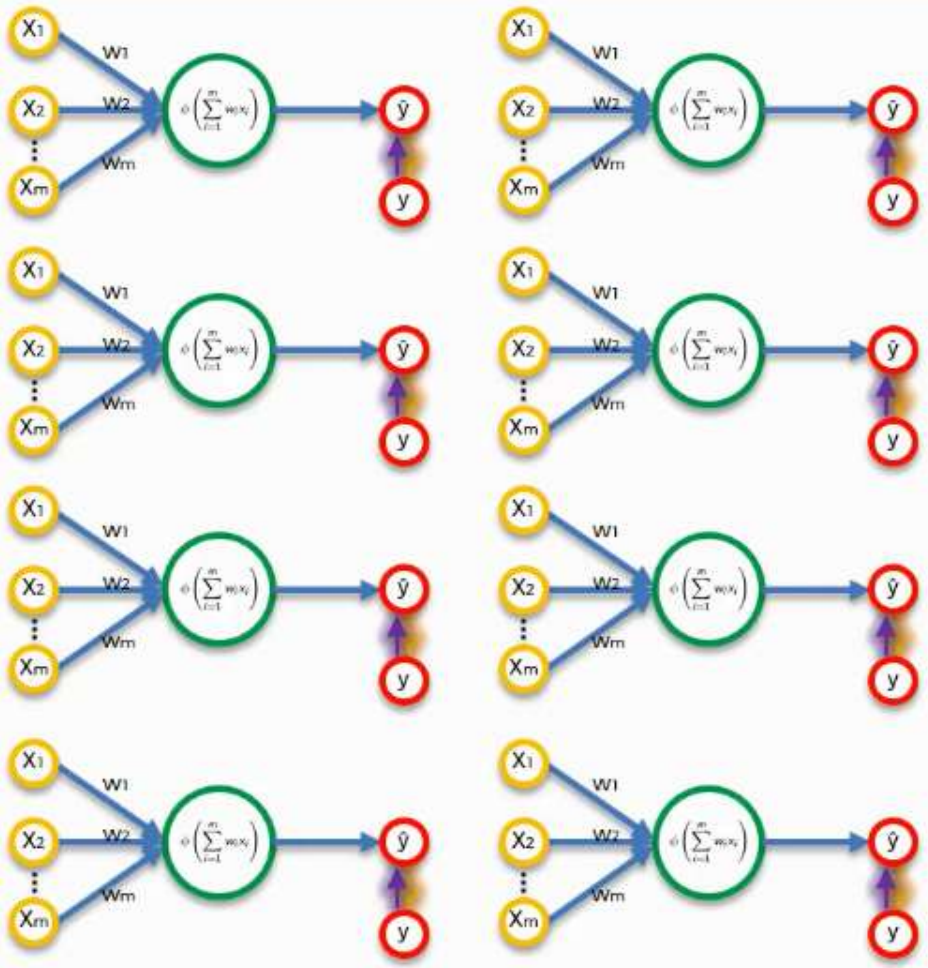


How do Neural Networks learn?



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

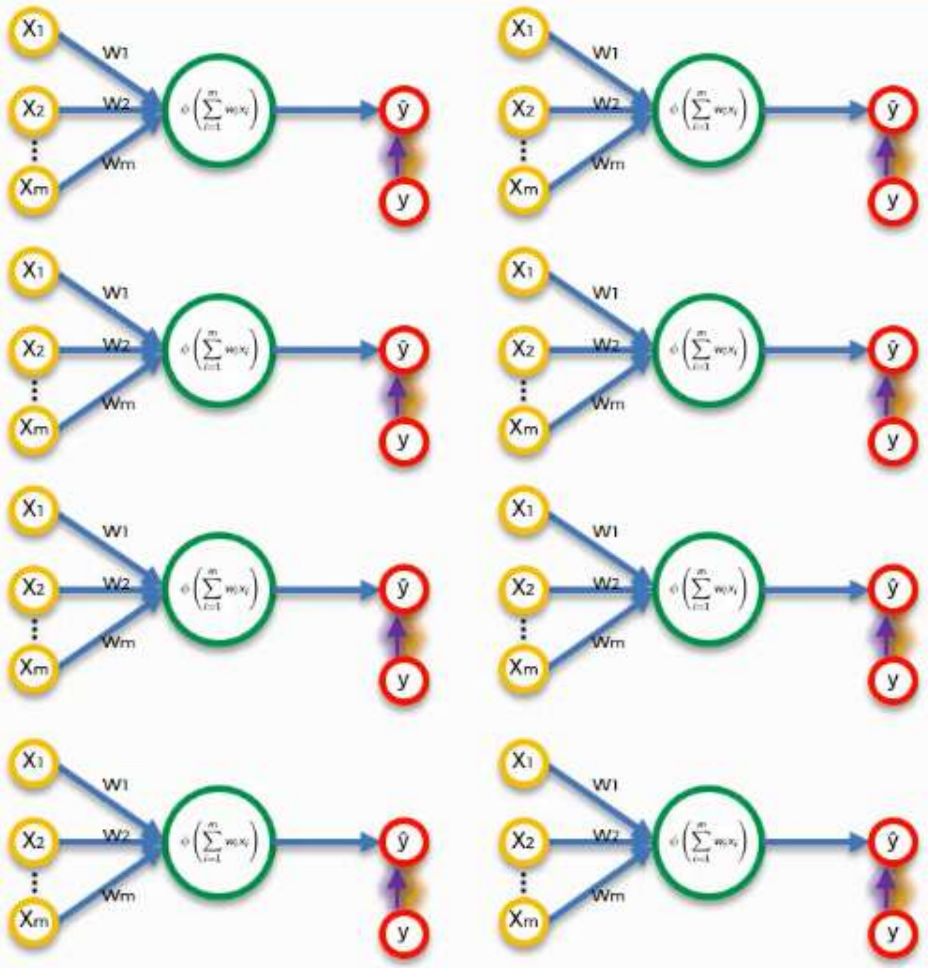
How do Neural Networks learn?



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

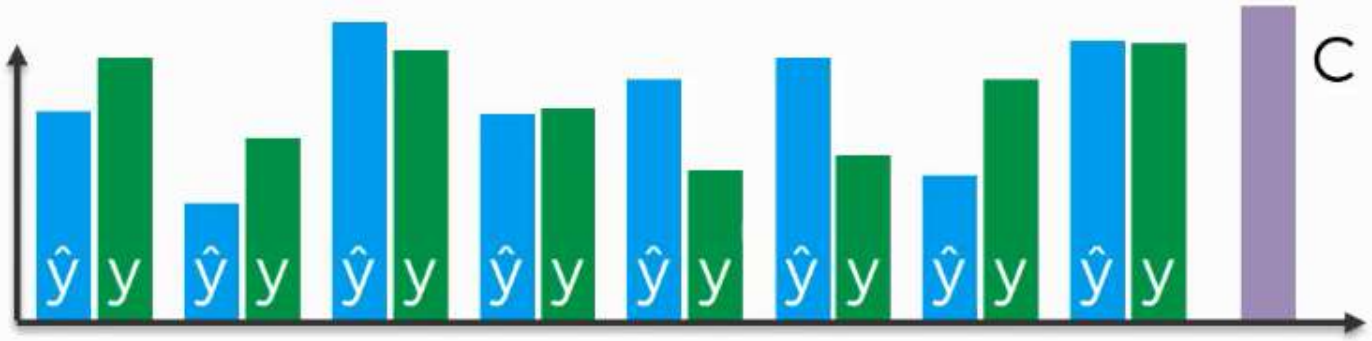
$$C = \sum \tfrac{1}{2}(\hat{y} - y)^2$$

How do Neural Networks learn?

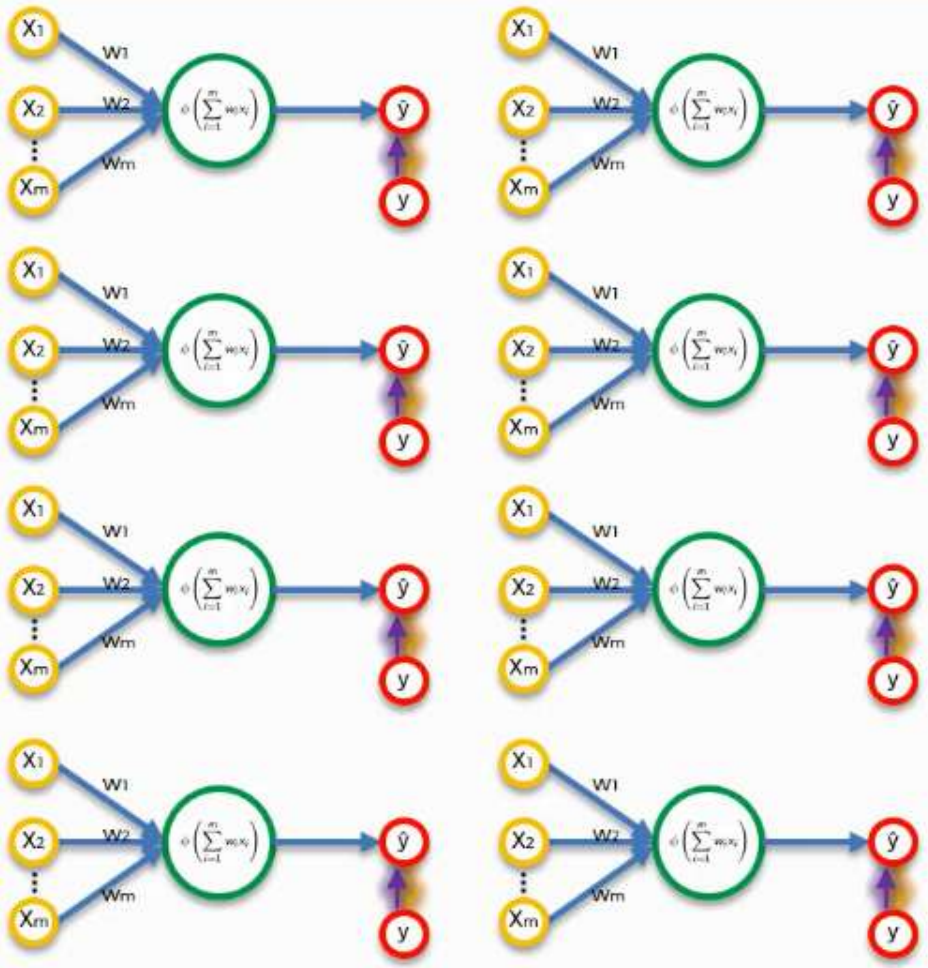


Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

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

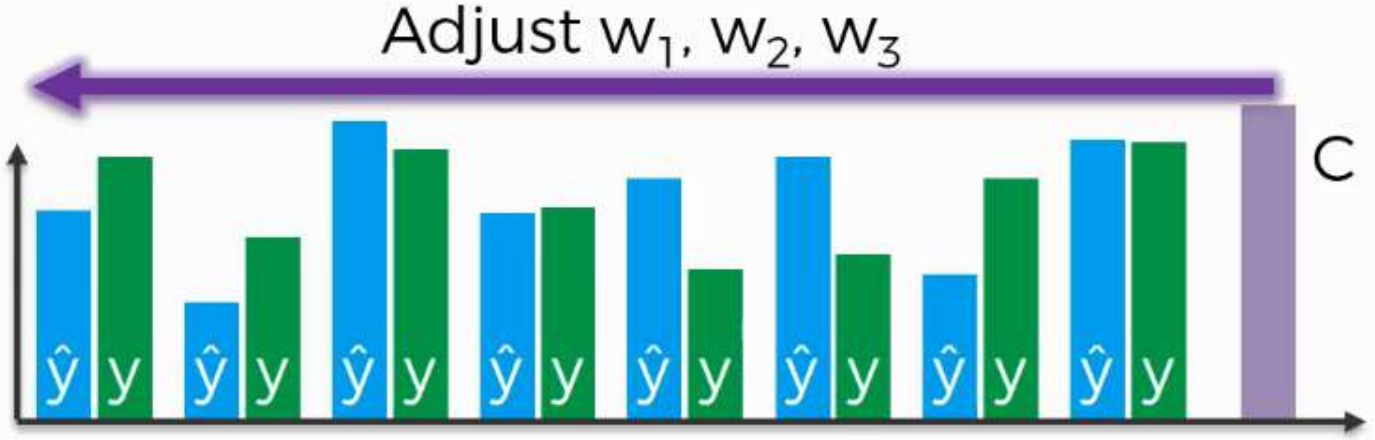


How do Neural Networks learn?

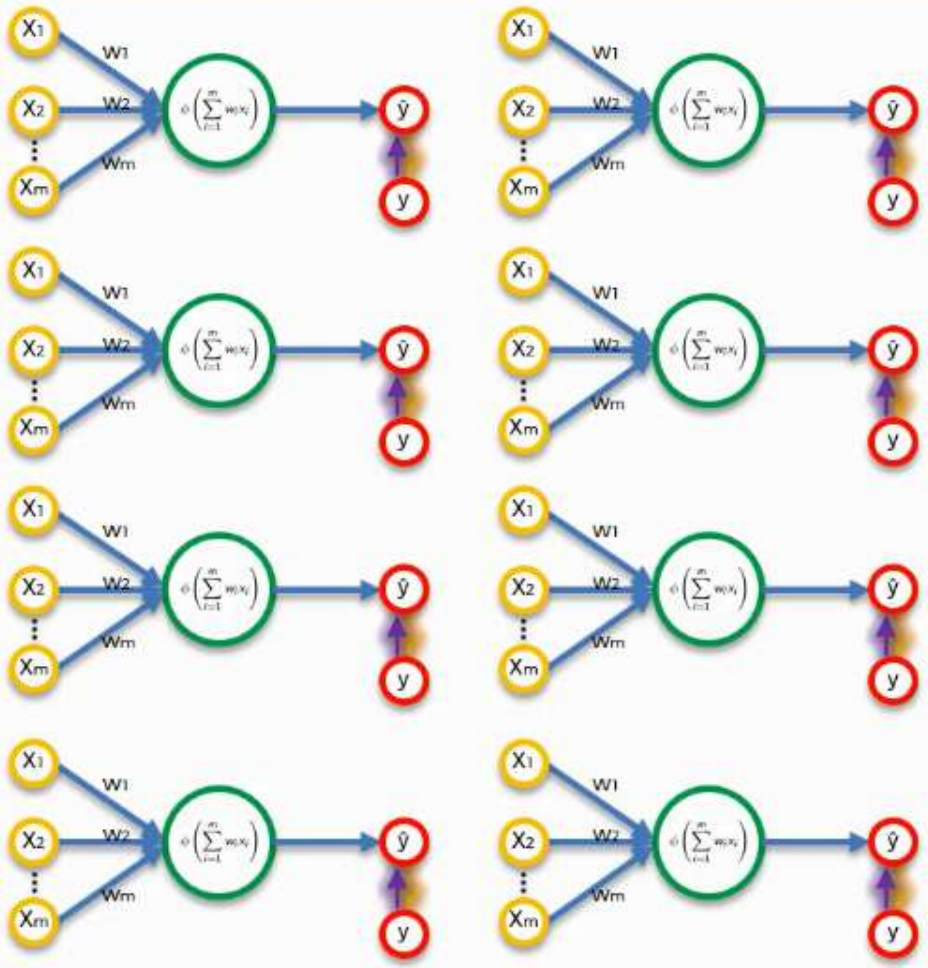


Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

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

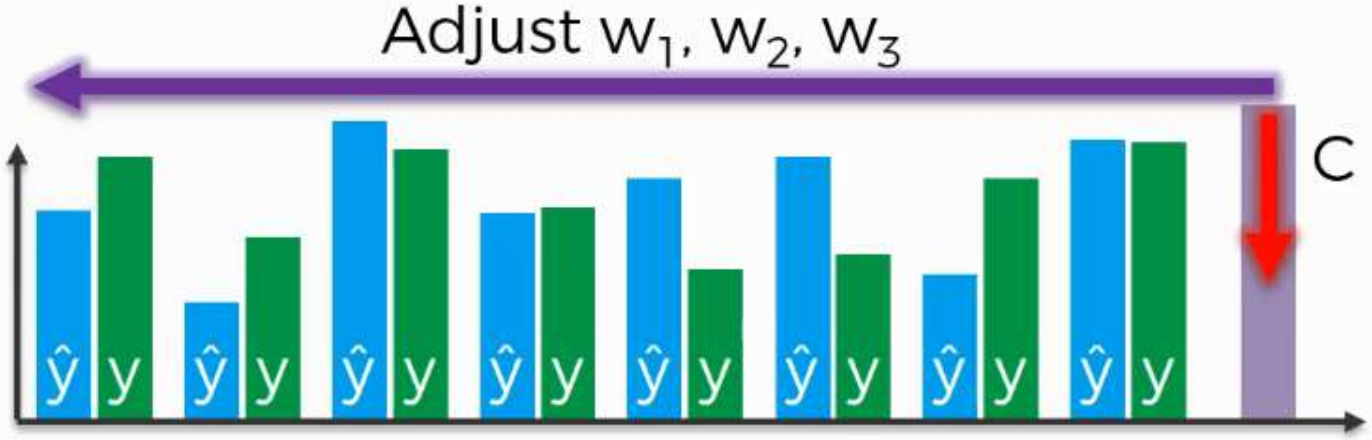


How do Neural Networks learn?



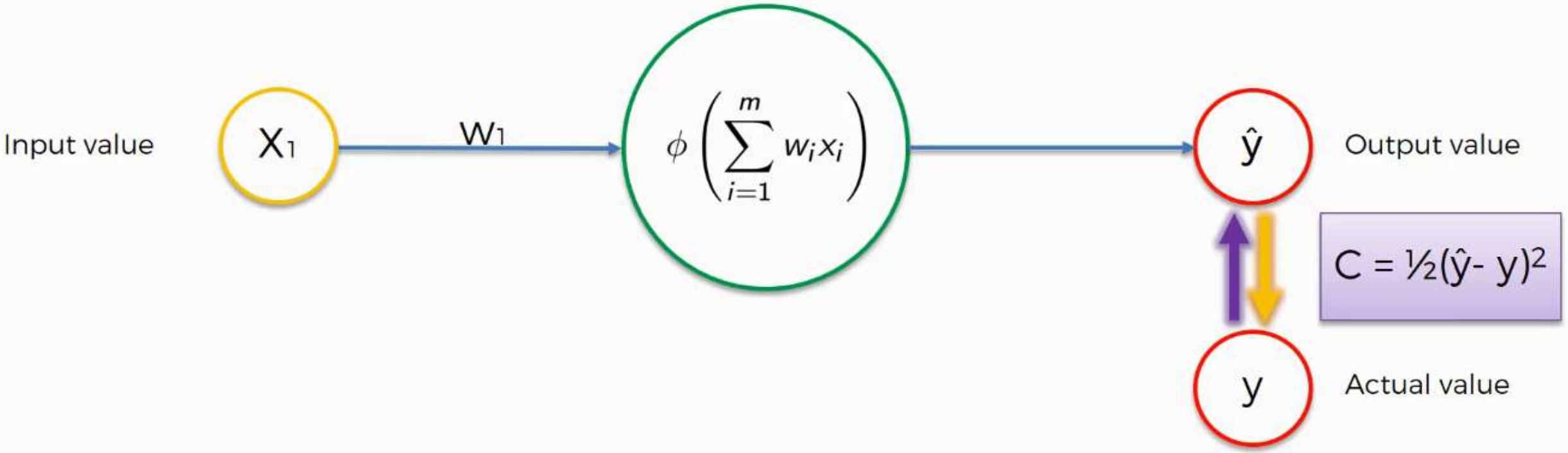
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

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

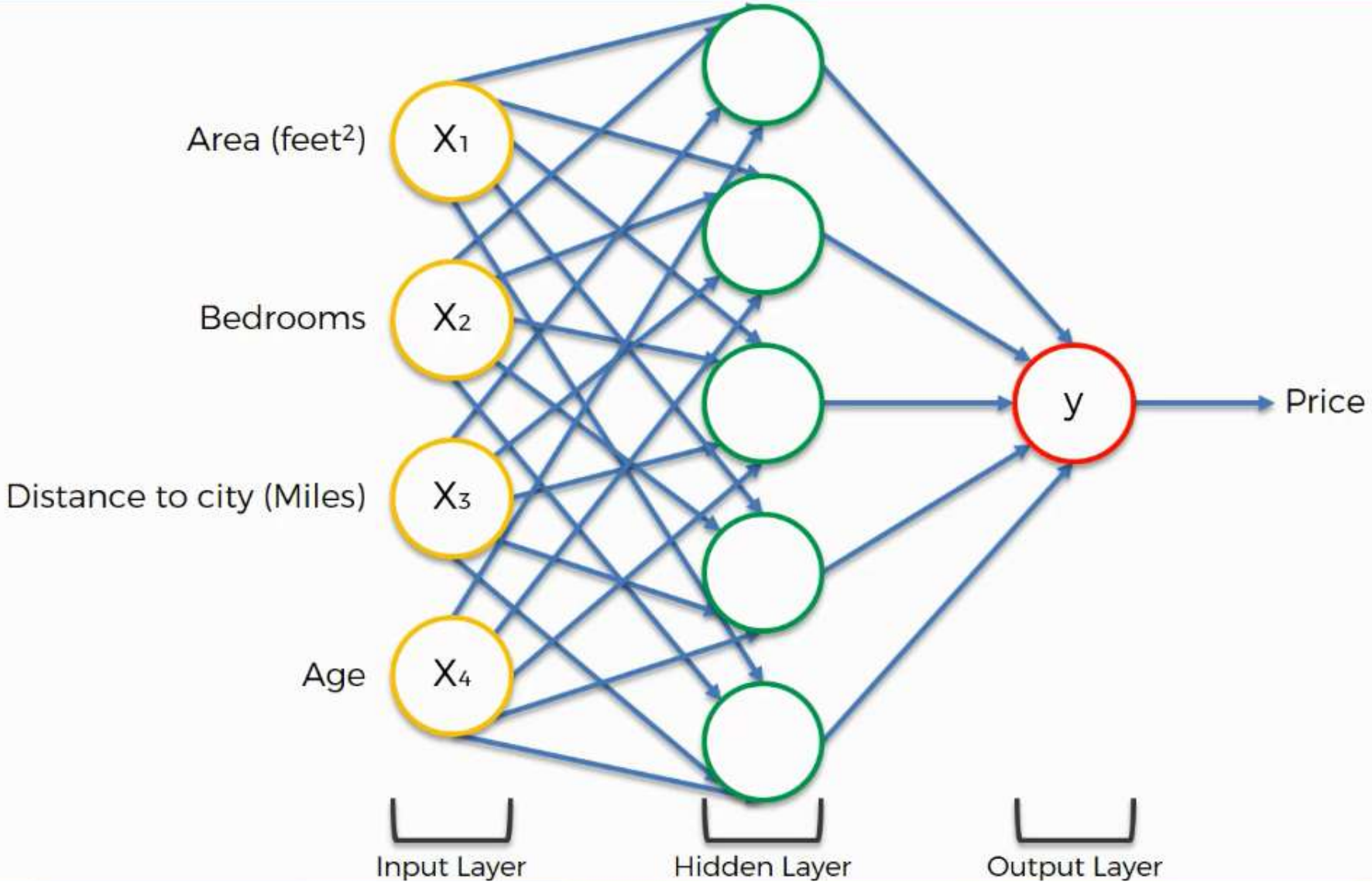


Gradient Descent

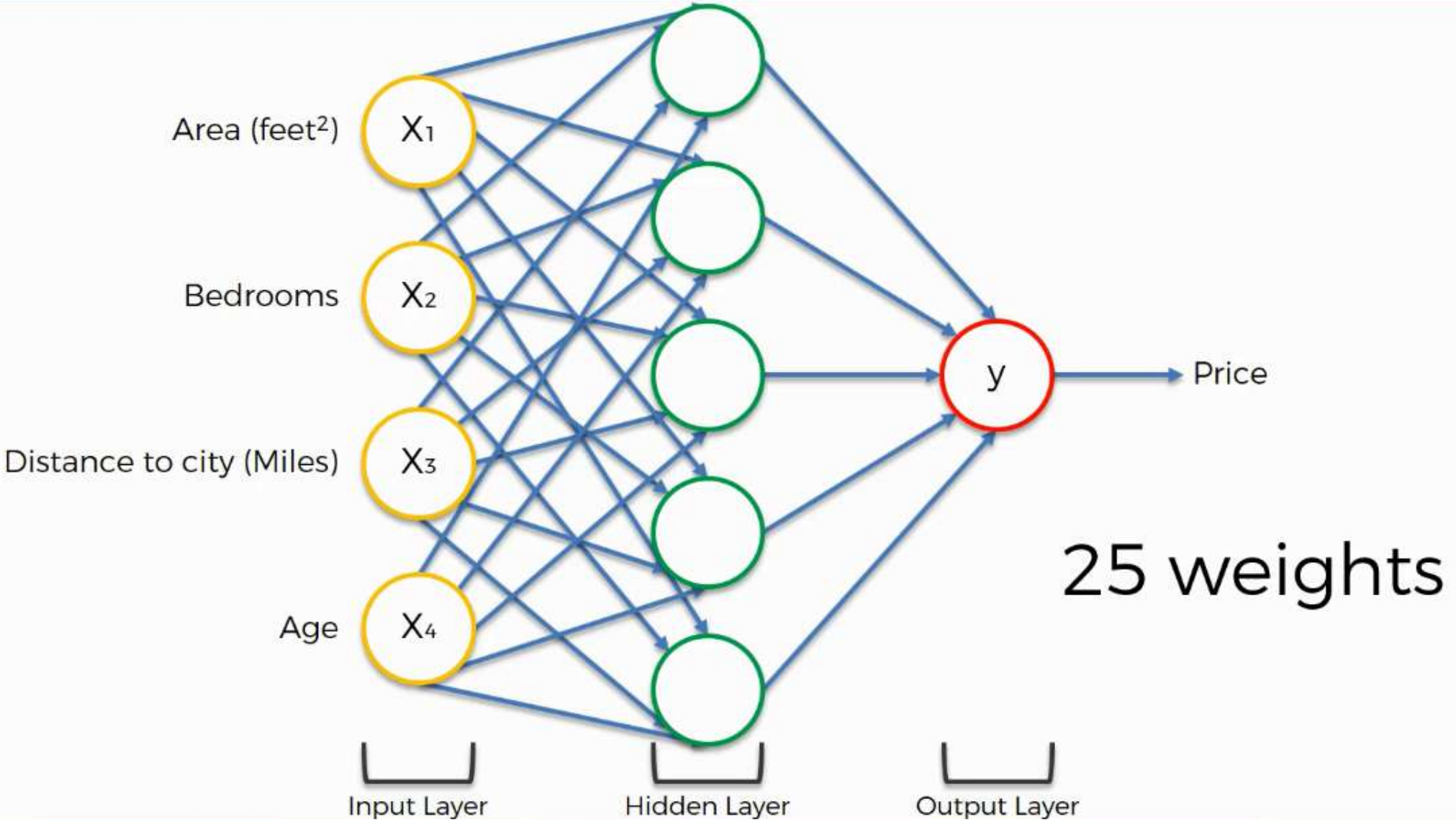
Gradient Descent



Gradient Descent



Gradient Descent



Gradient Descent

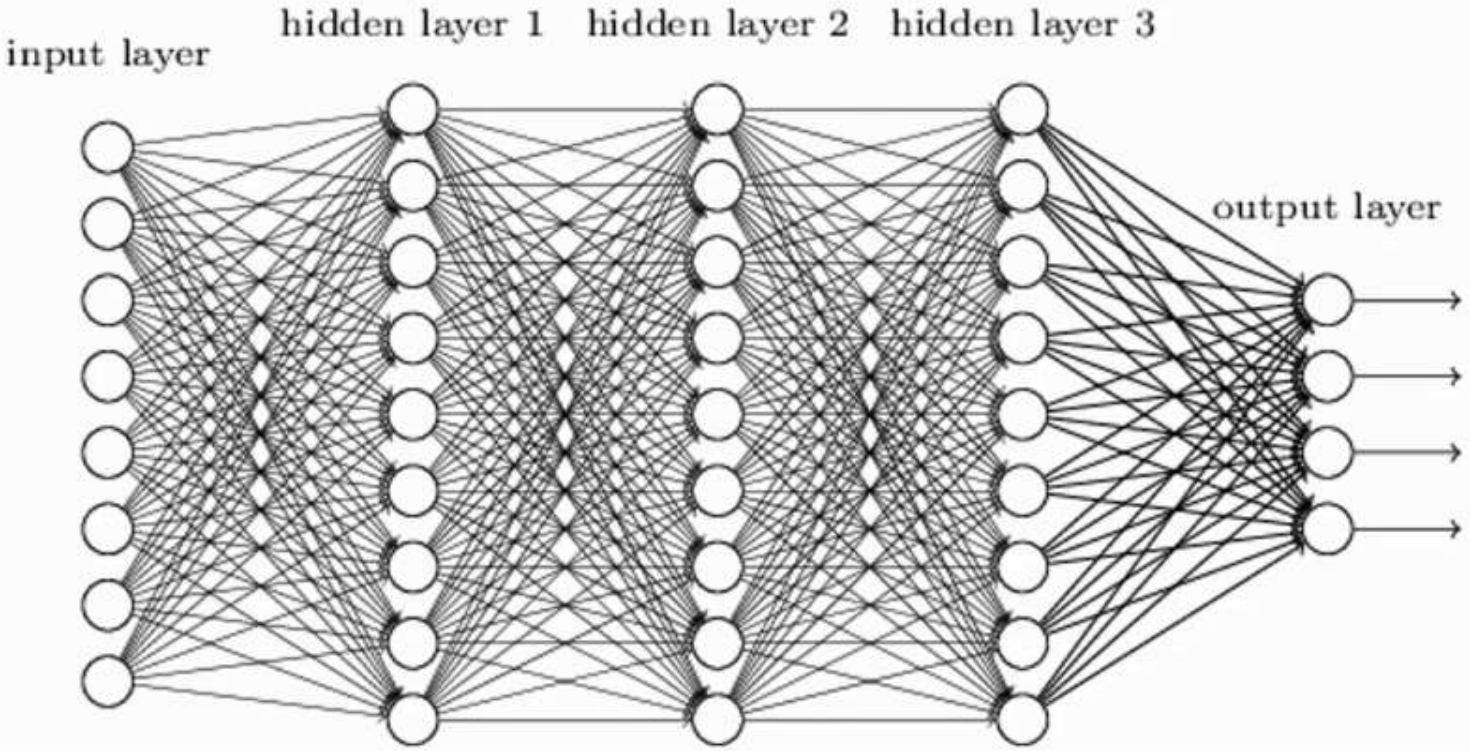


Image Source: neuralnetworksanddeeplearning.com

Gradient Descent

Gradient descent is the most popular optimization strategy used in machine learning and deep learning at the moment. It is used when training data models, can be combined with every algorithm.

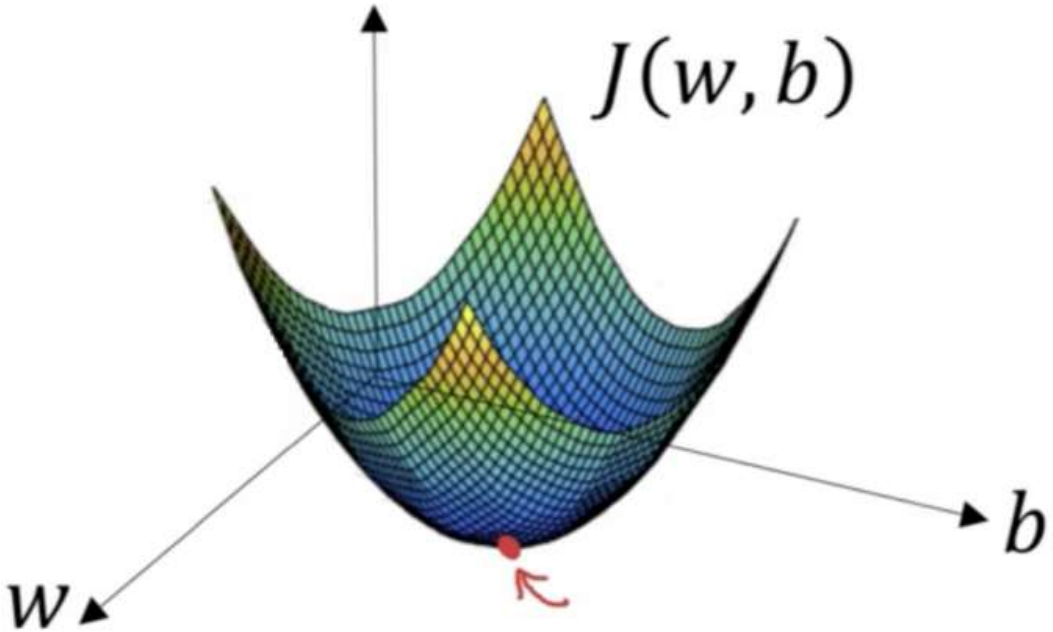
Gradient descent is an optimization algorithm that's used when training a machine learning model. It's based on a convex function and tweaks its parameters iteratively to minimize a given function to its local minimum.

Gradient descent is simply used to find the values of a function's parameters (coefficients) that minimize a cost function as far as possible.

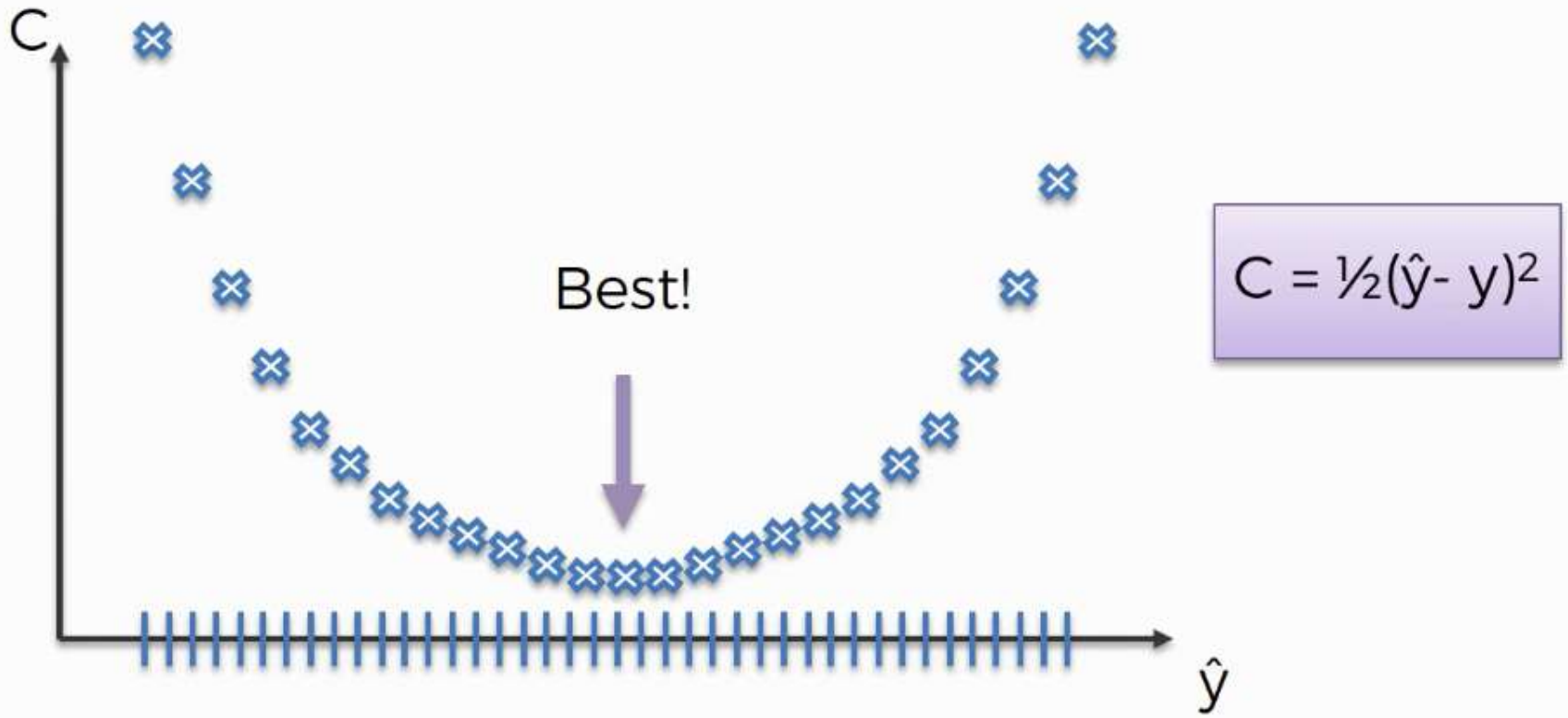
Although this function does not always guarantee to find a global minimum and can get stuck at a local minimum.

Gradient Descent

We want to find the values of *w* and *b* that correspond to the minimum of the cost function (marked with the red arrow). To start finding the right values we initialize *w* and *b* with some random numbers. Gradient descent then starts at that point (somewhere around the top of our illustration), and it takes one step after another in the steepest downside direction (i.e., from the top to the bottom of the illustration) until it reaches the point where the cost function is as small as possible.



Gradient Descent



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

TYPES OF GRADIENT DESCENT

There are three popular types of gradient descent that mainly differ in the amount of data they use:

- BATCH GRADIENT DESCENT
- STOCHASTIC GRADIENT DESCENT
- MINI-BATCH GRADIENT DESCENT

BATCH GRADIENT DESCENT

- Batch gradient descent, also called vanilla gradient descent, calculates the error for each example within the training dataset, but only after all training examples have been evaluated does the model get updated. This whole process is like a cycle and it's called a training epoch.

- Some disadvantages are the stable error gradient can sometimes result in a state of convergence that isn’t the best the model can achieve. It also requires the entire training dataset be in memory and available to the algorithm.

STOCHASTIC GRADIENT DESCENT

- Stochastic gradient descent (SGD) does this for each training example within the dataset, meaning it updates the parameters for each training example one by one. Depending on the problem, this can make SGD faster than batch gradient descent.

MINI-BATCH GRADIENT DESCENT

- Mini-batch gradient descent is the go-to method since it’s a combination of the concepts of SGD and batch gradient descent. It simply splits the training dataset into small batches and performs an update for each of those batches.

- This is the go-to algorithm when training a neural network and it is the most common type of gradient descent within deep learning.

Stochastic Gradient Descent

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

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

Gradient Descent

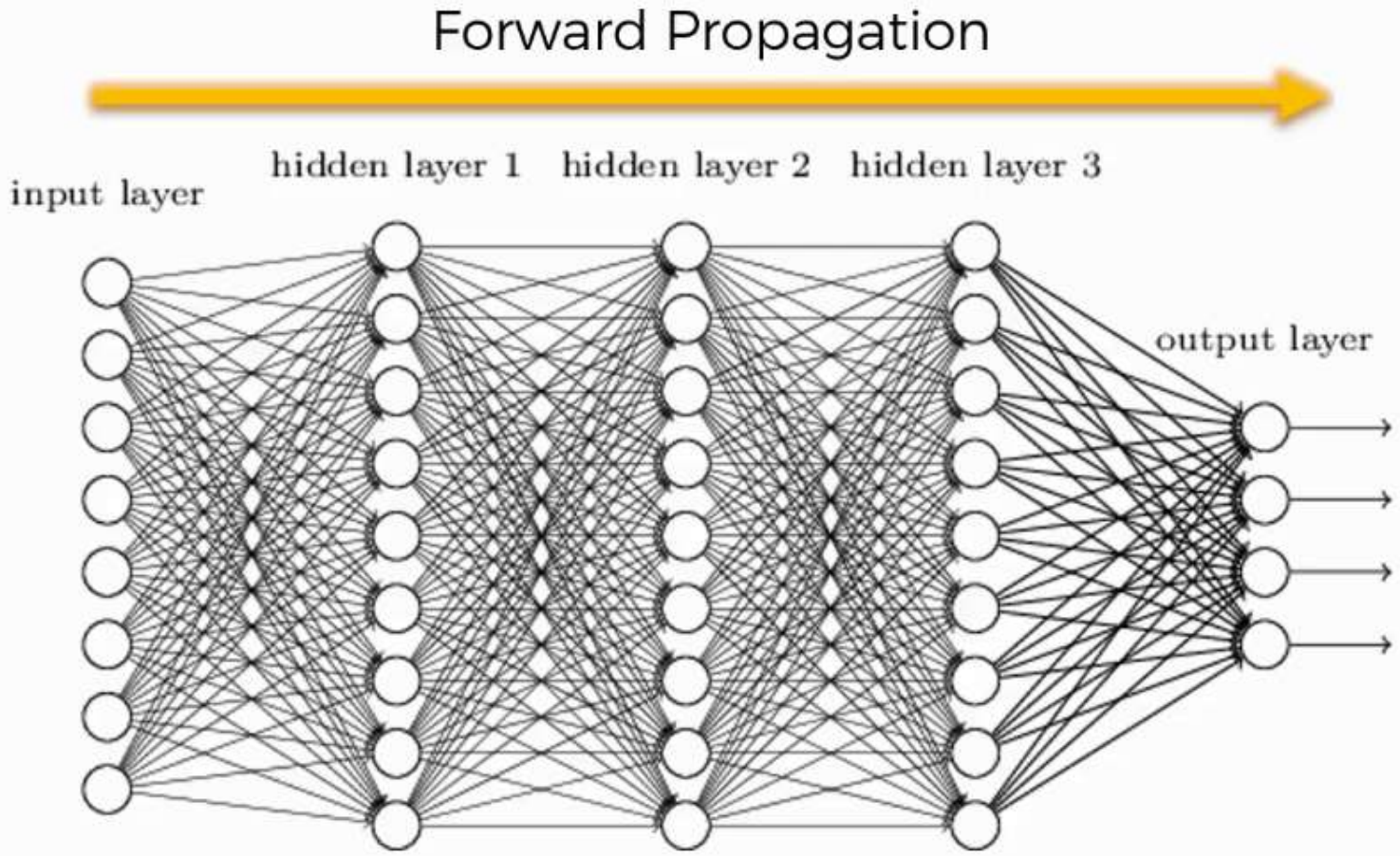


Image Source: neuralnetworksanddeeplearning.com

Gradient Descent

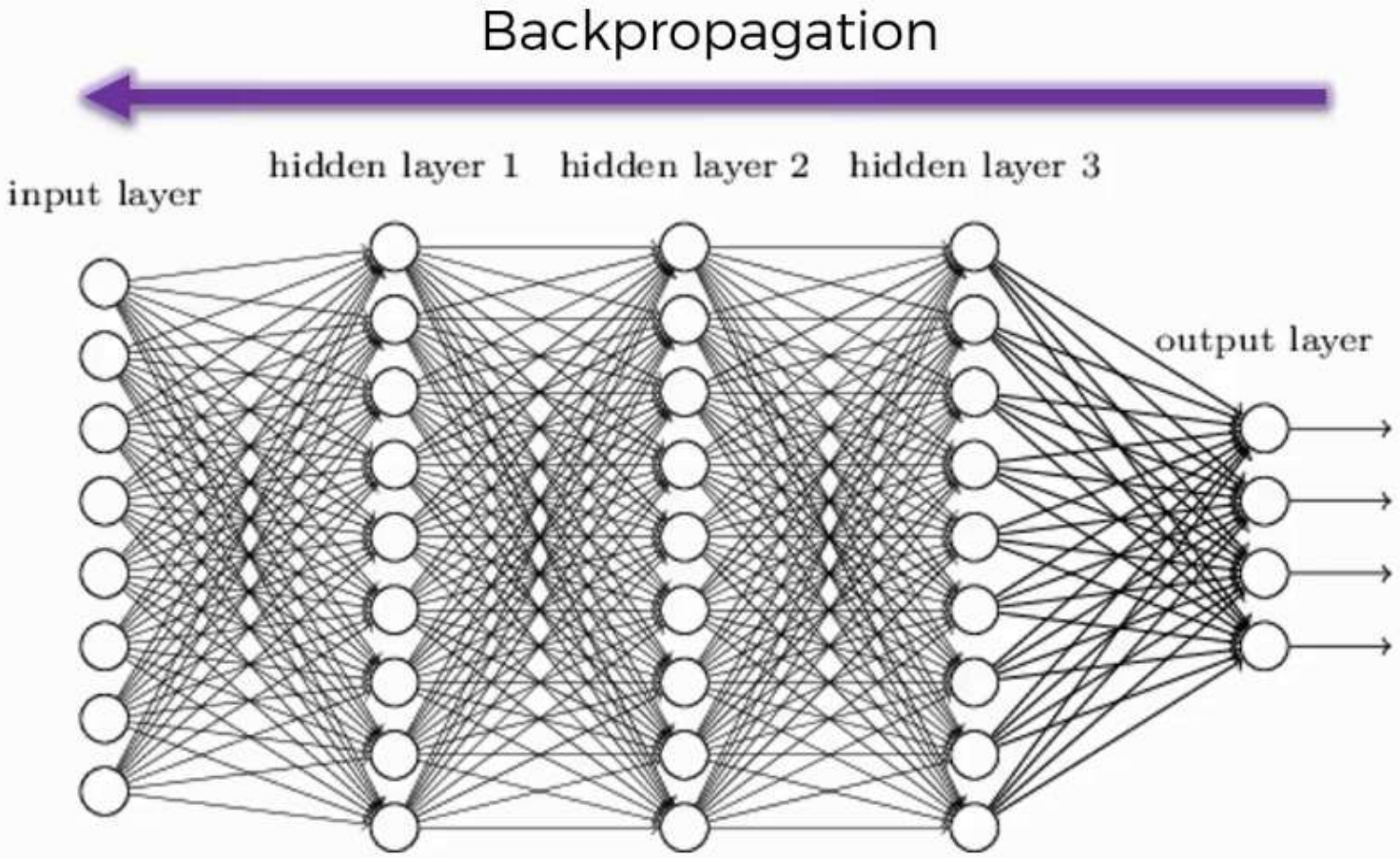


Image Source: neuralnetworksanddeeplearning.com

Training the ANN with Stochastic Gradient Descent

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.

Thank you !!