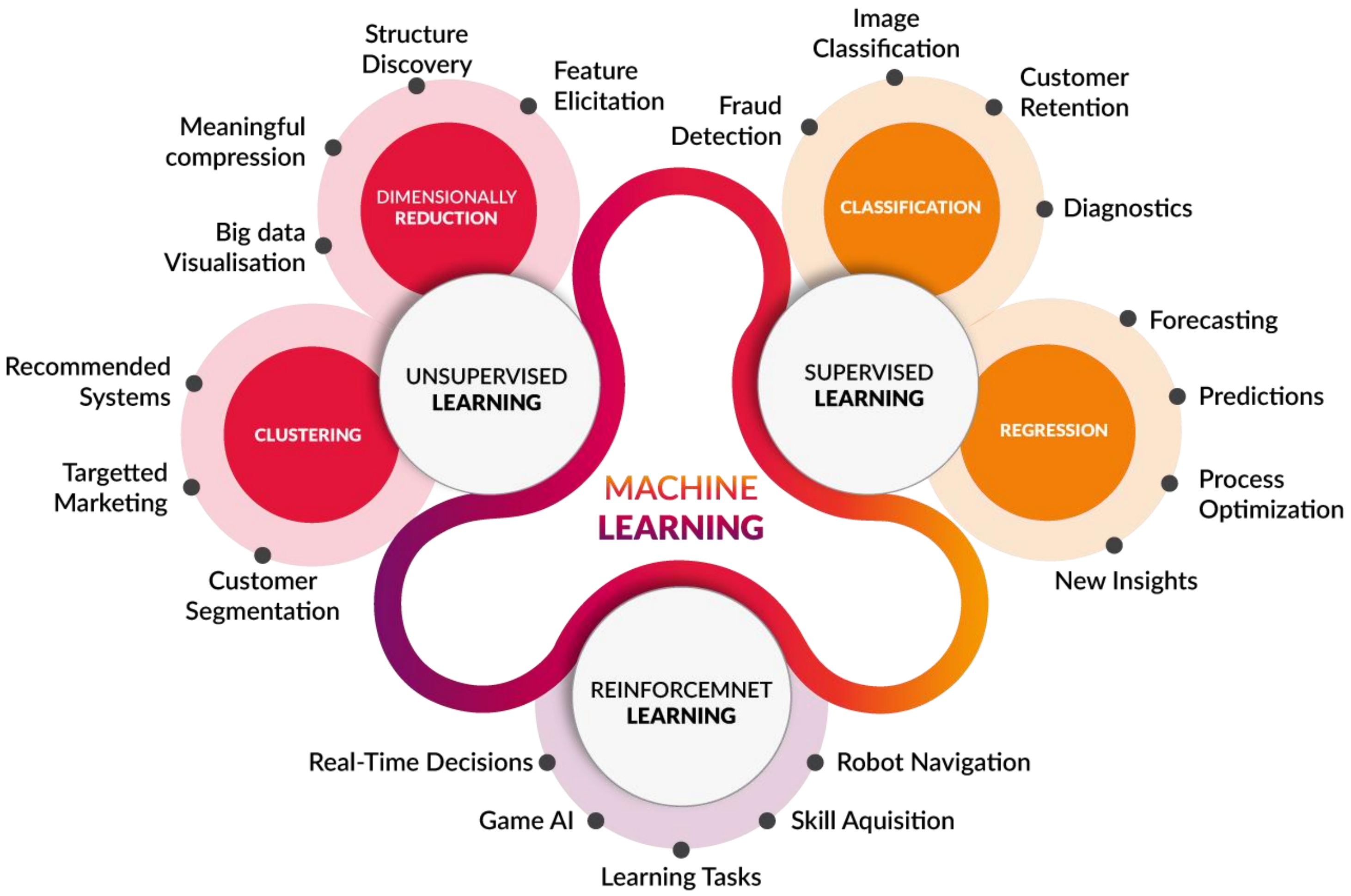
Deep Learning

Machine Learning

Dr. Adnan Abid

Courtesy SuperDataScience

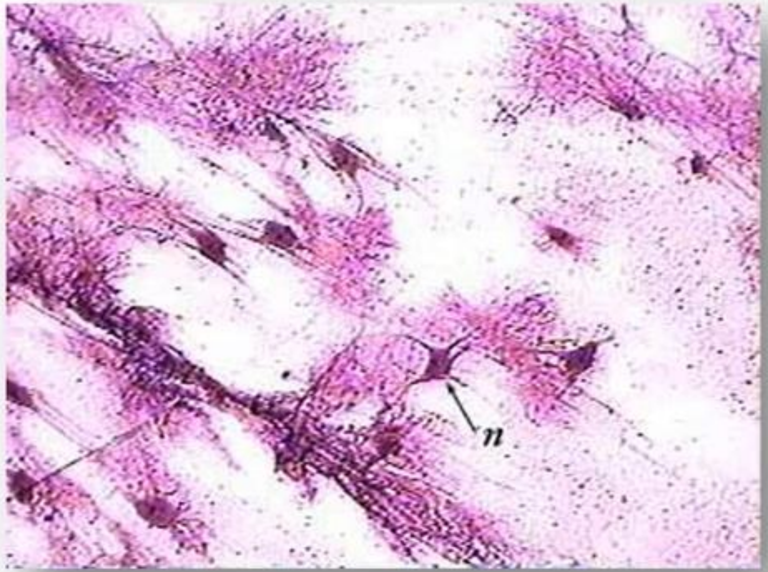

MACHINE LEARNING
UNSUPERVISED LEARNING
SUPERVISED LEARNING
REINFORCEMNET LEARNING
DIMENSIONALLY REDUCTION
Structure Discovery
Feature Elicitation
Meaningful compression
Big data Visualisation
CLUSTERING
Recommended Systems
Targetted Marketing
Customer Segmentation
CLASSIFICATION
Image Classification
Customer Retention
Fraud Detection
Diagnostics
REGRESSION
Forecasting
Predictions
Process Optimization
New Insights
Real-Time Decisions
Game AI
Learning Tasks
Skill Aquisition
Robot Navigation

Plan of Attack

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



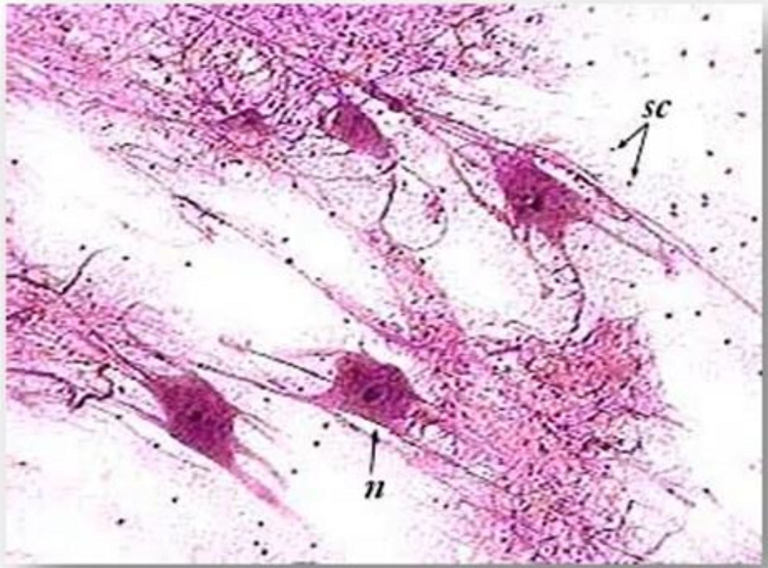


Image Source: www.austincc.edu

The Neuron

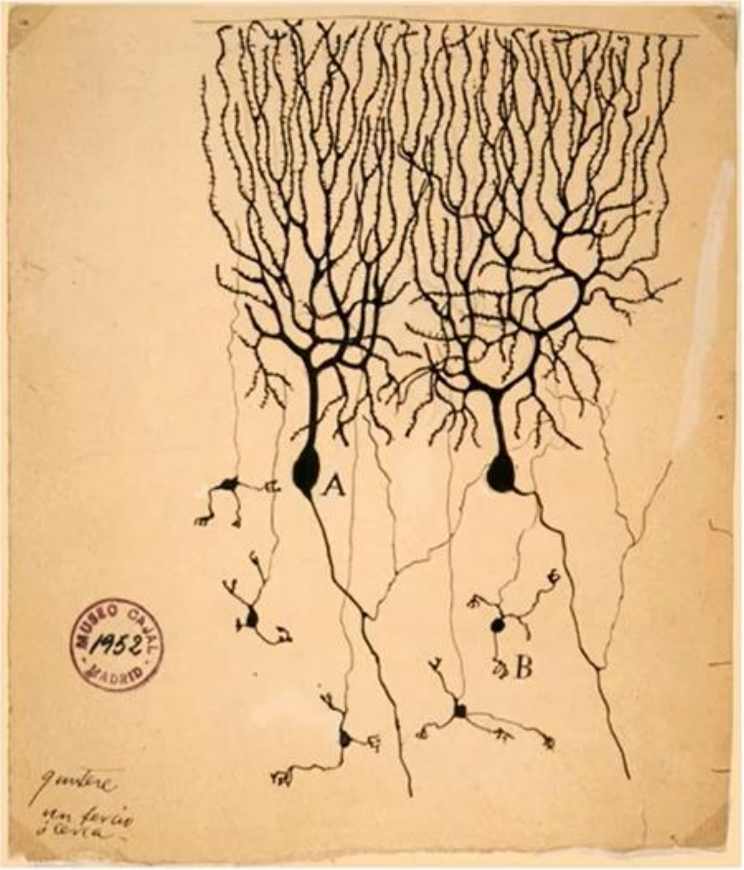


Image Source: Wikipedia

The Neuron

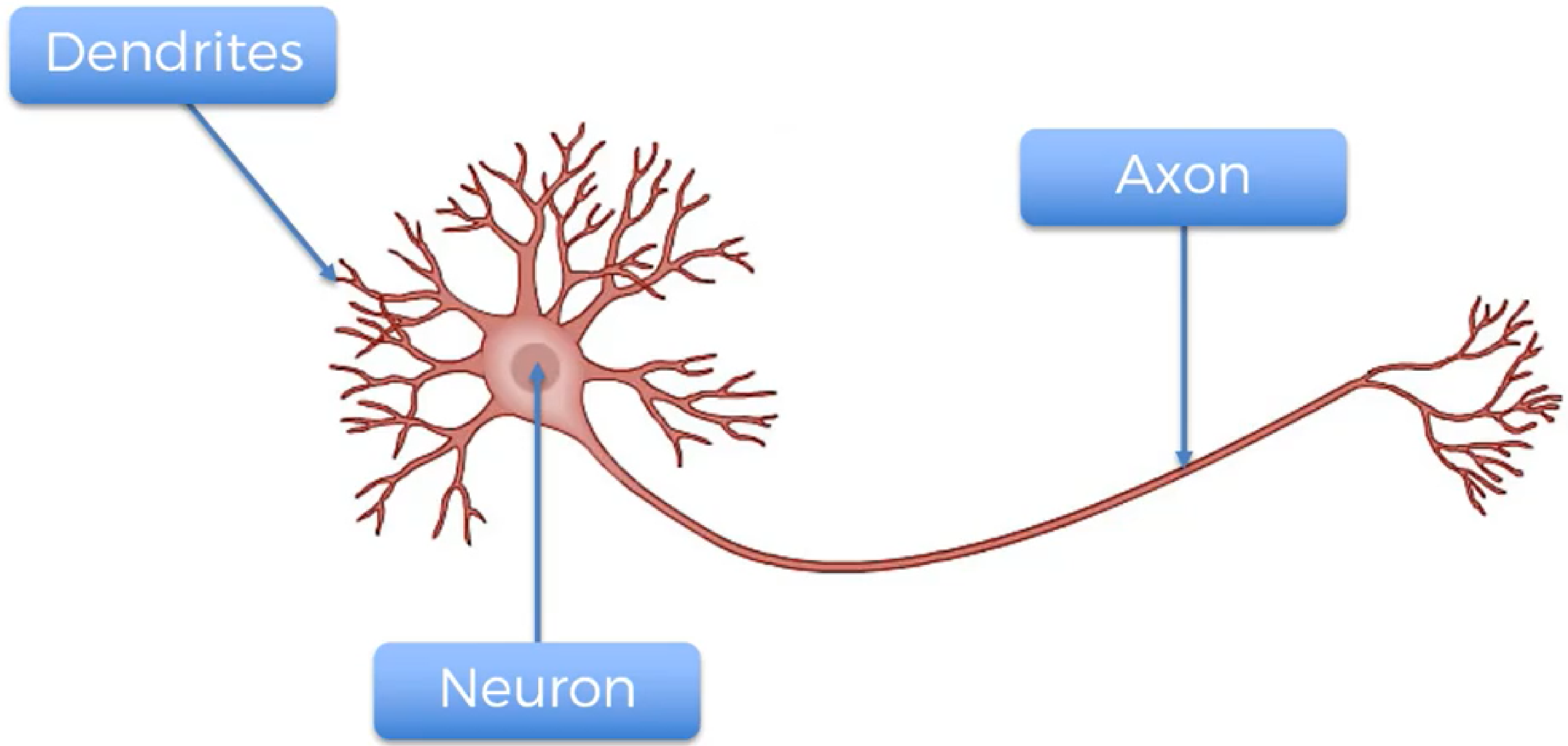


Image Source: Wikipedia

The Neuron

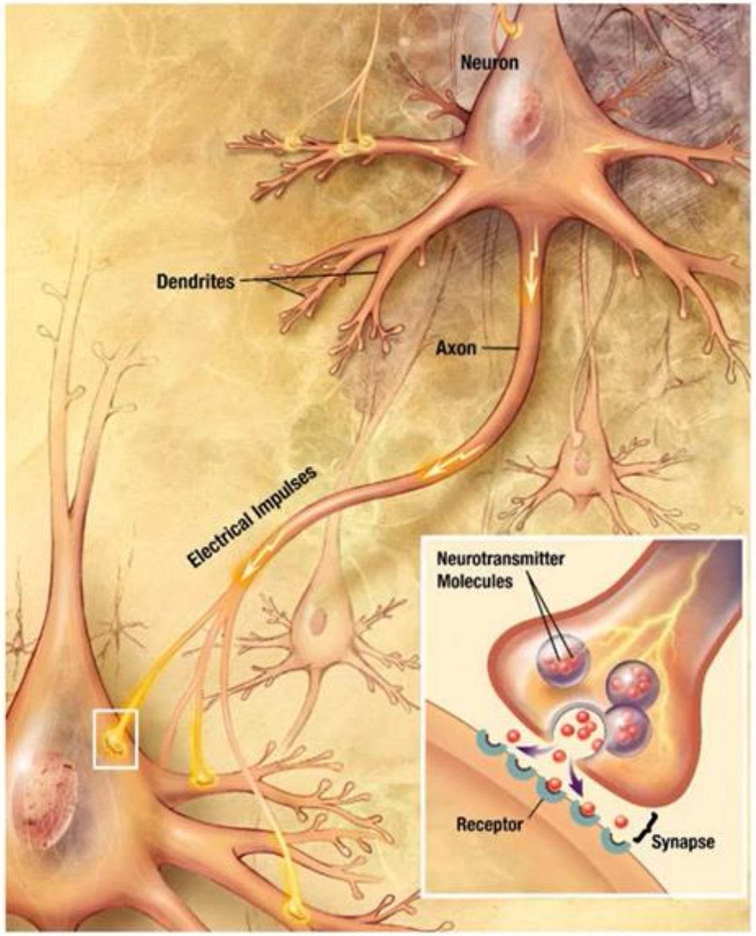
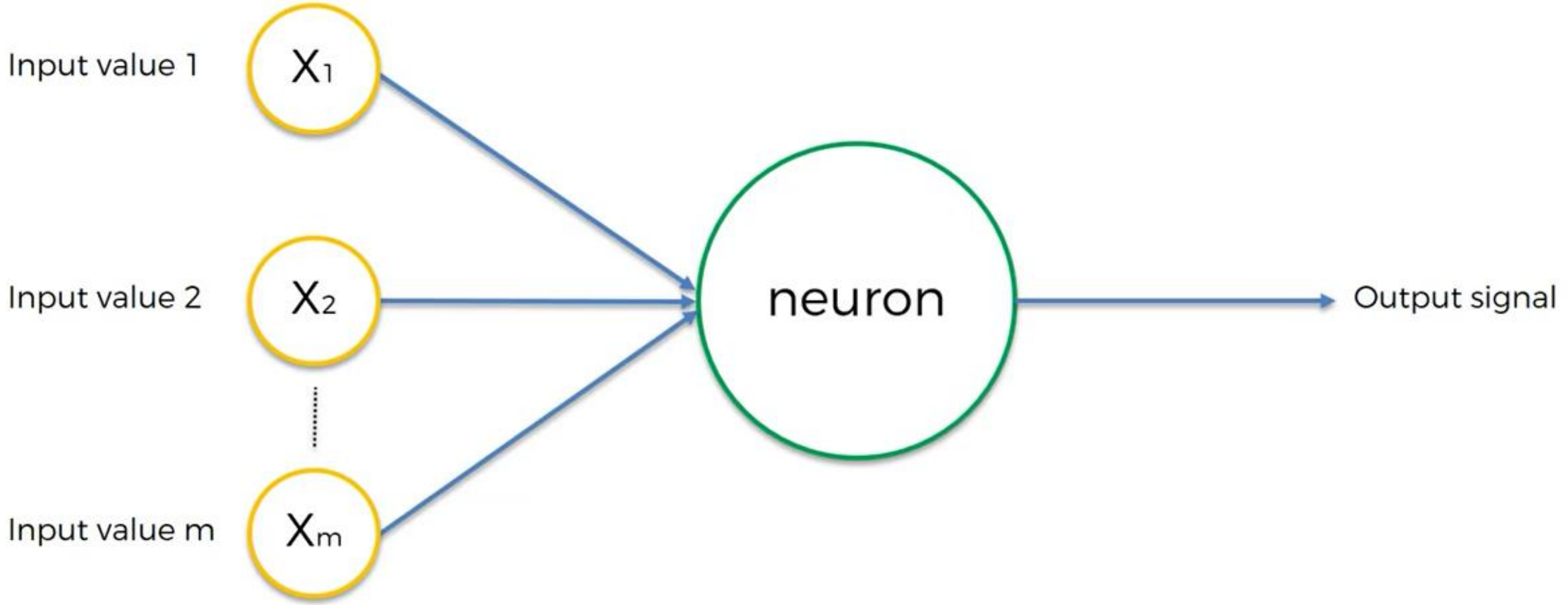


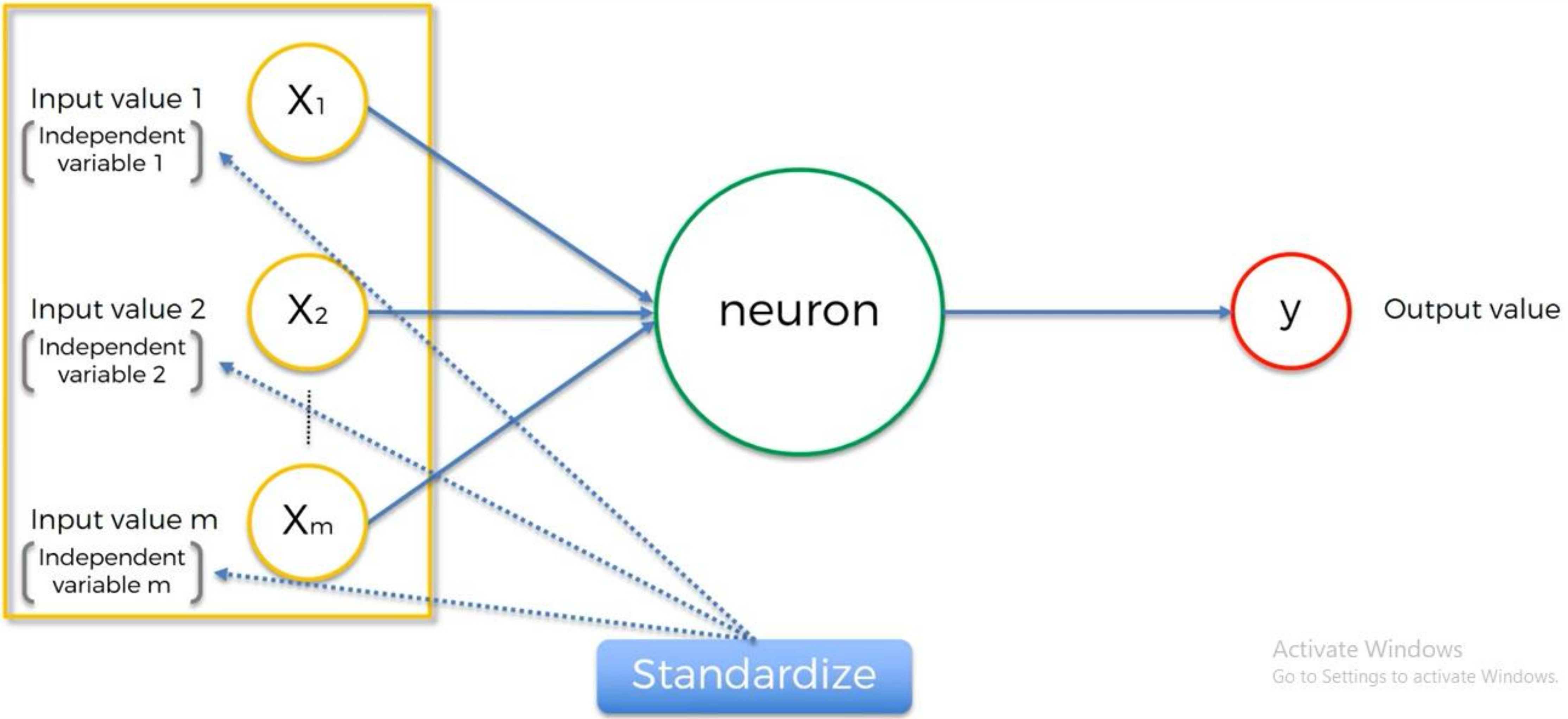
Image Source: Wikipedia

The Neuron



Input value 1
Input value 2
Input value m
X1
X2
Xm
neuron
Output signal

The Neuron

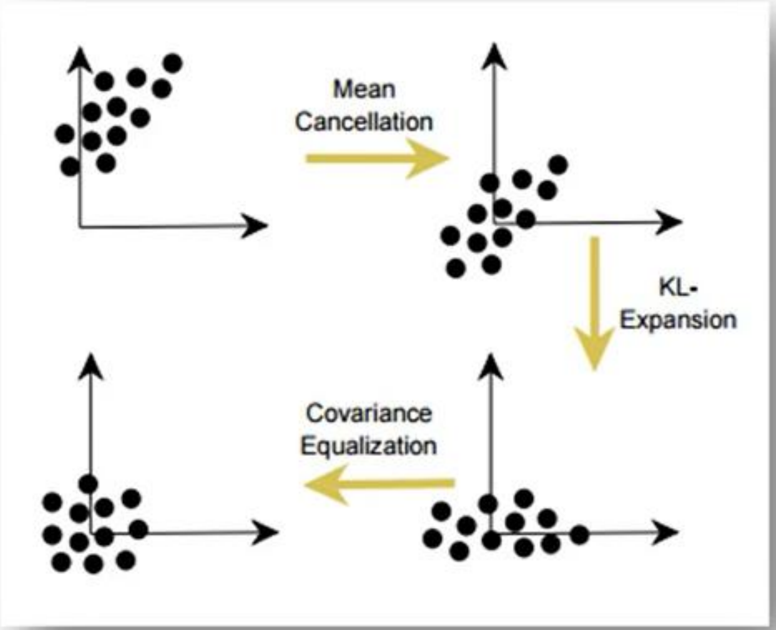

Input value 1
Independent variable 1
Input value 2
Independent variable 2
Input value m
Independent variable m
X_1
X_2
X_m
neuron
y
Output value
Standardize

The Neuron

Additional Reading:

Efficient BackProp

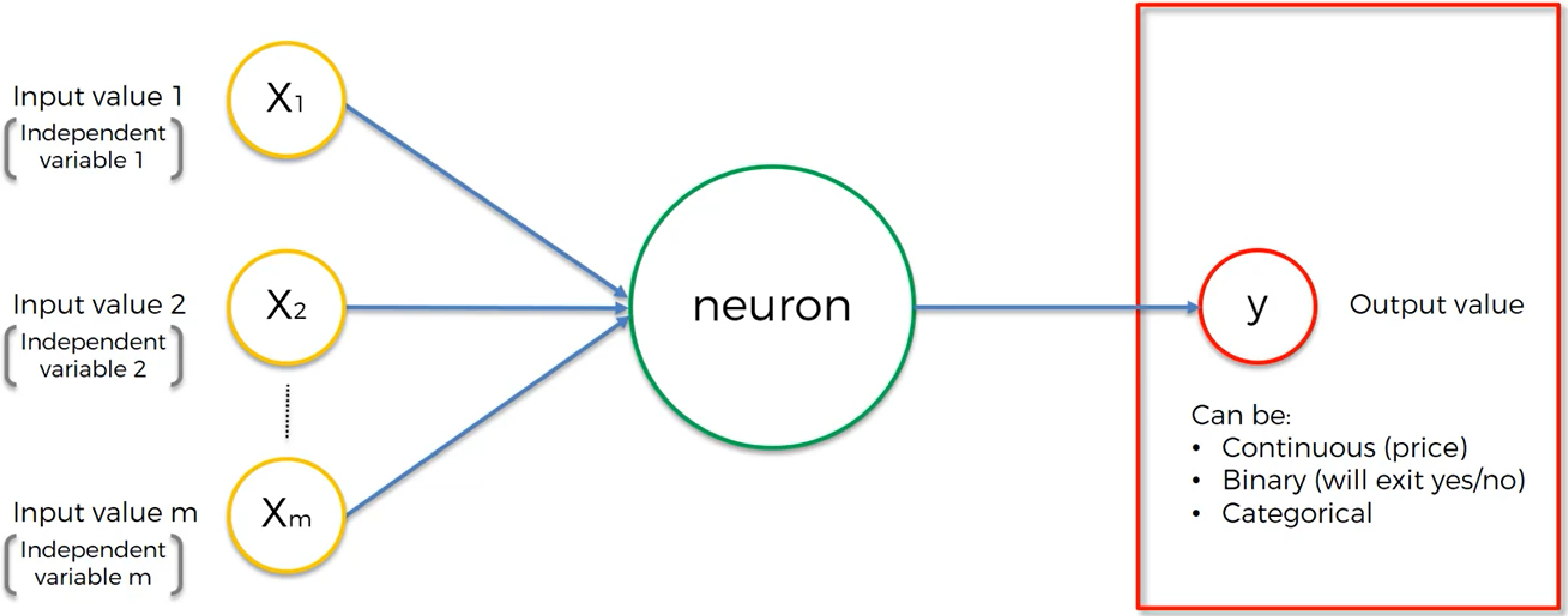
By Yann LeCun et al. (1998)



Link:

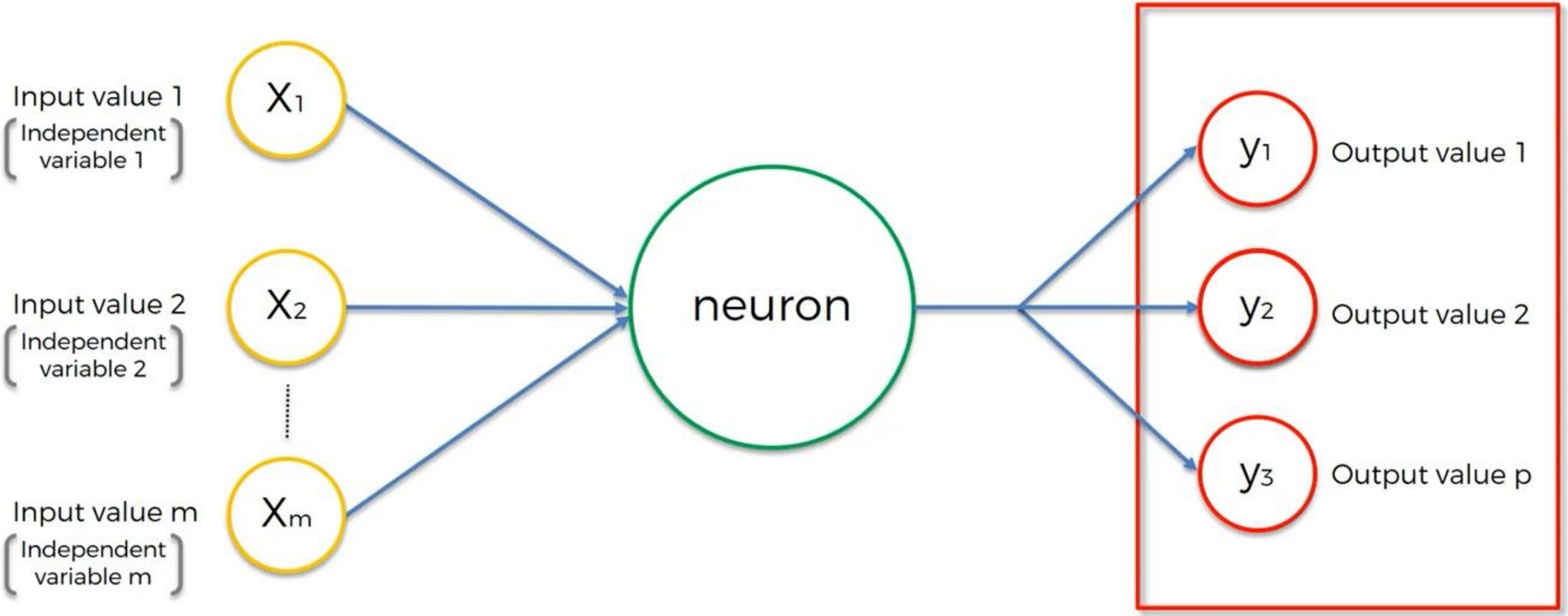
http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

The Neuron



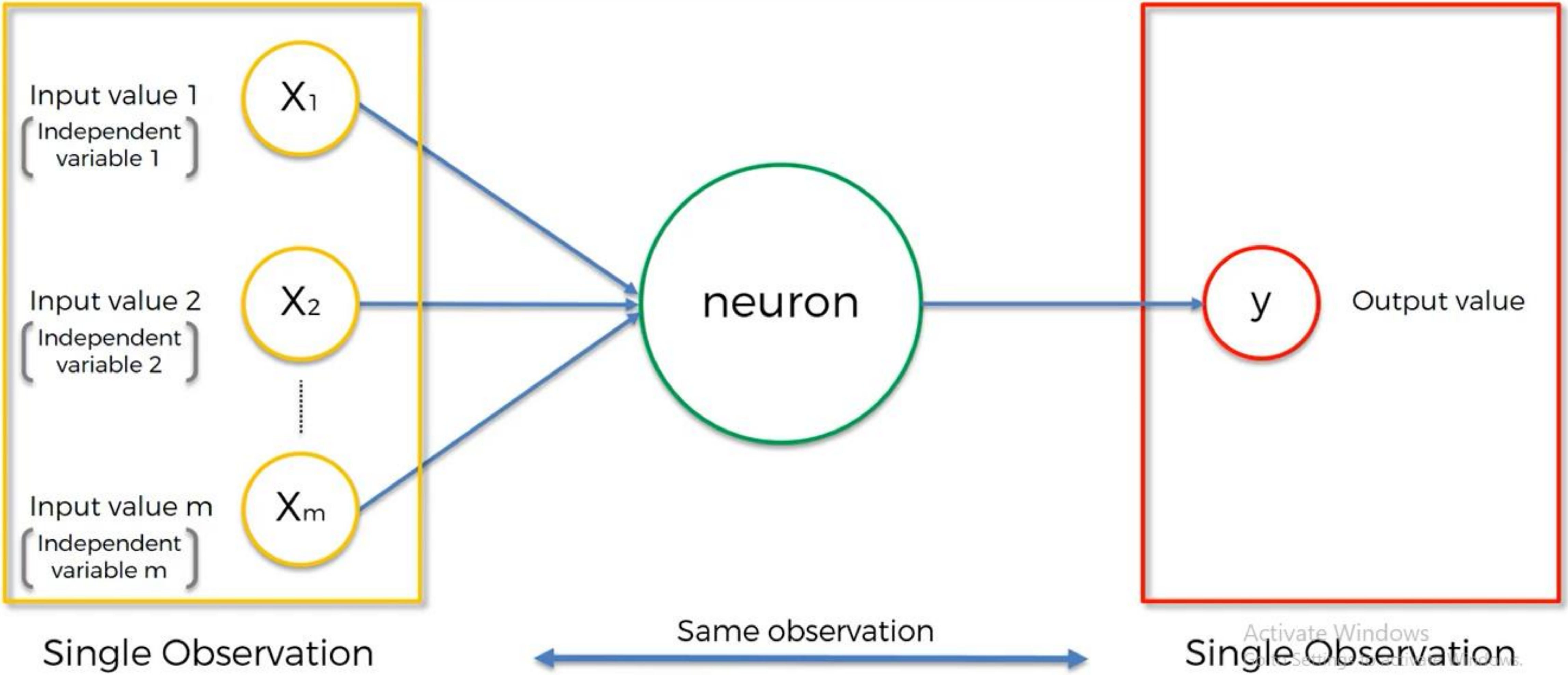
Input value 1
Independent variable 1
X_1
Input value 2
Independent variable 2
X_2
Input value m
Independent variable m
X_m
neuron
y
Output value
Can be:
• Continuous (price)
• Binary (will exit yes/no)
• Categorical

The Neuron

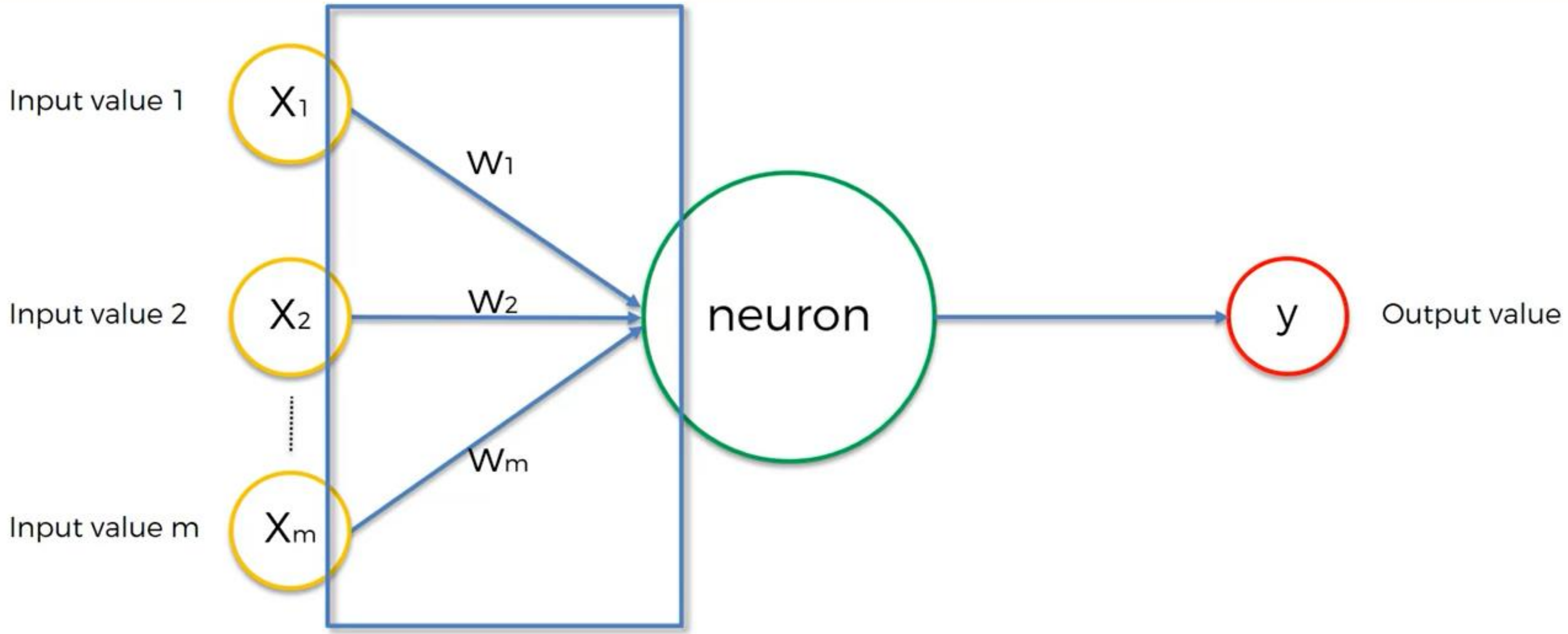


Input value 1
Independent variable 1
Input value 2
Independent variable 2
Input value m
Independent variable m
X1
X2
Xm
neuron
y1
y2
y3
Output value 1
Output value 2
Output value p

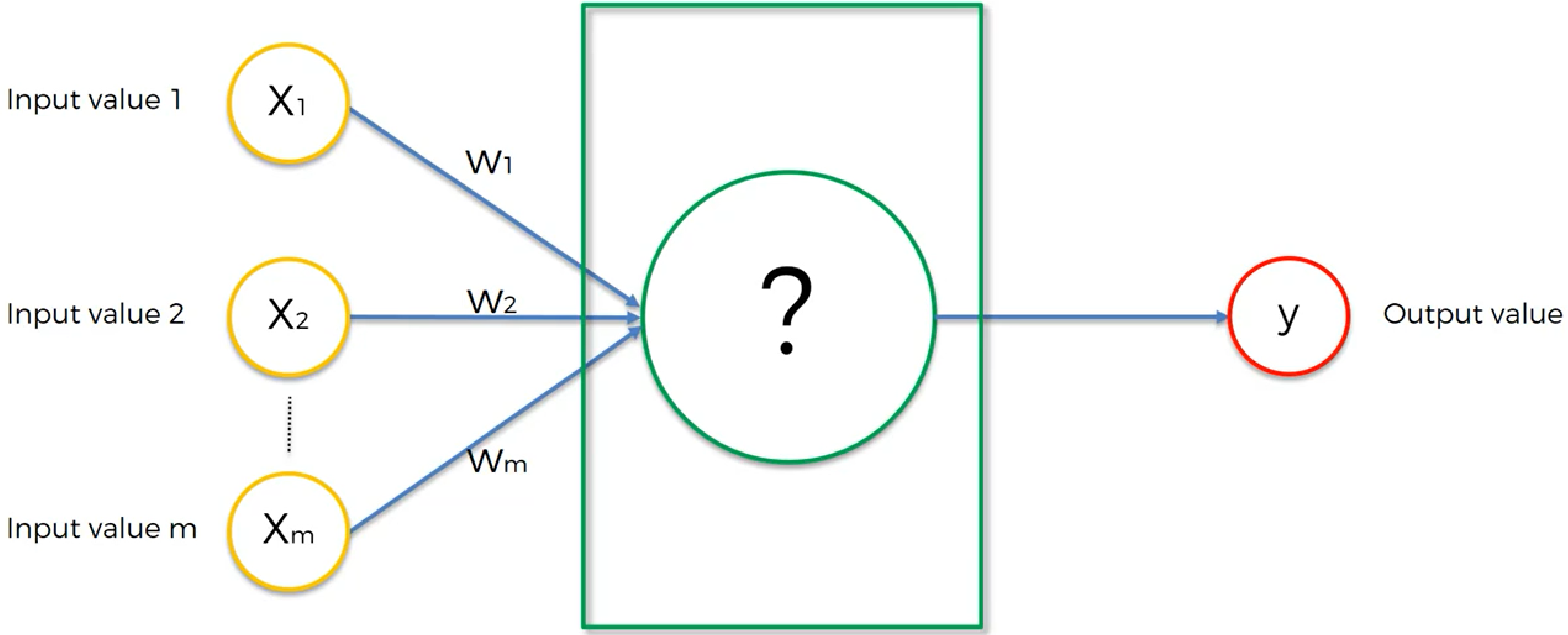
The Neuron


Input value 1
Independent variable 1
X1
Input value 2
Independent variable 2
X2
Input value m
Independent variable m
Xm
neuron
y
Output value
Single Observation
Same observation
Single Observation

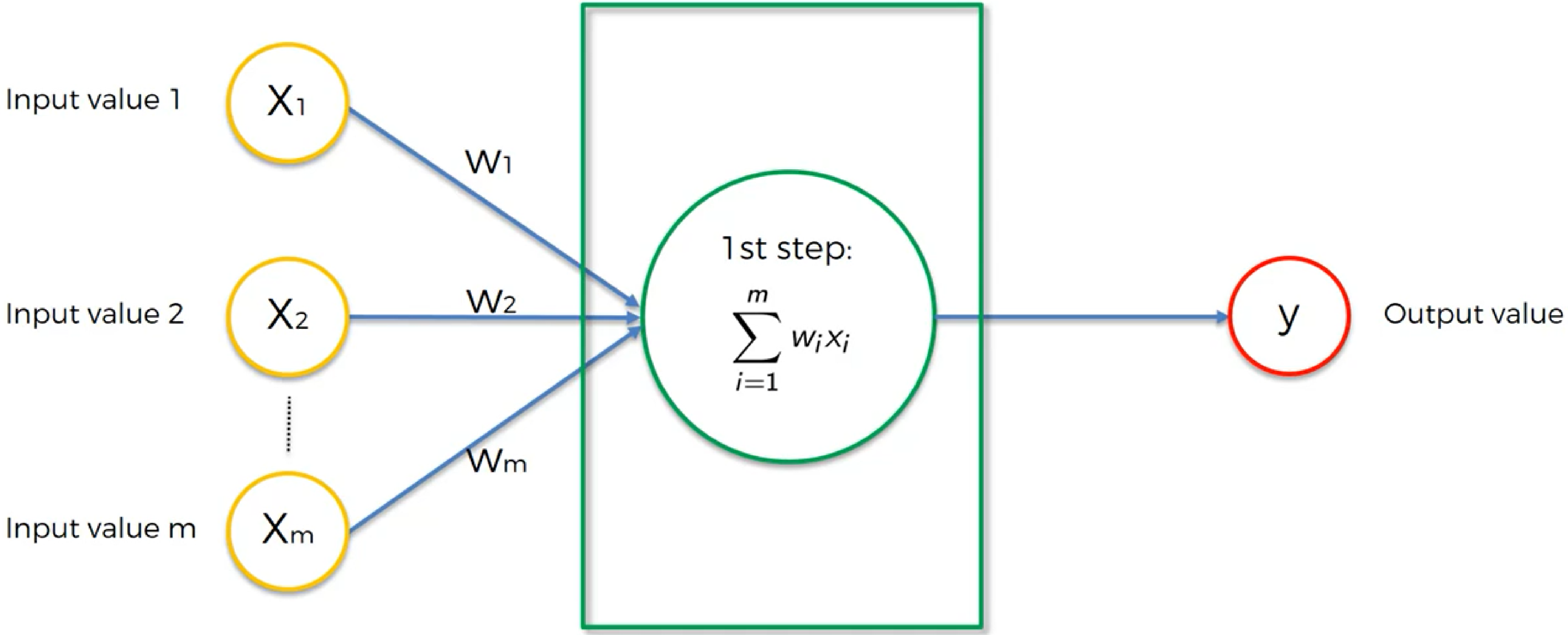
The Neuron


Input value 1
Input value 2
Input value m
X_1
X_2
X_m
w_1
w_2
w_m
neuron
y
Output value

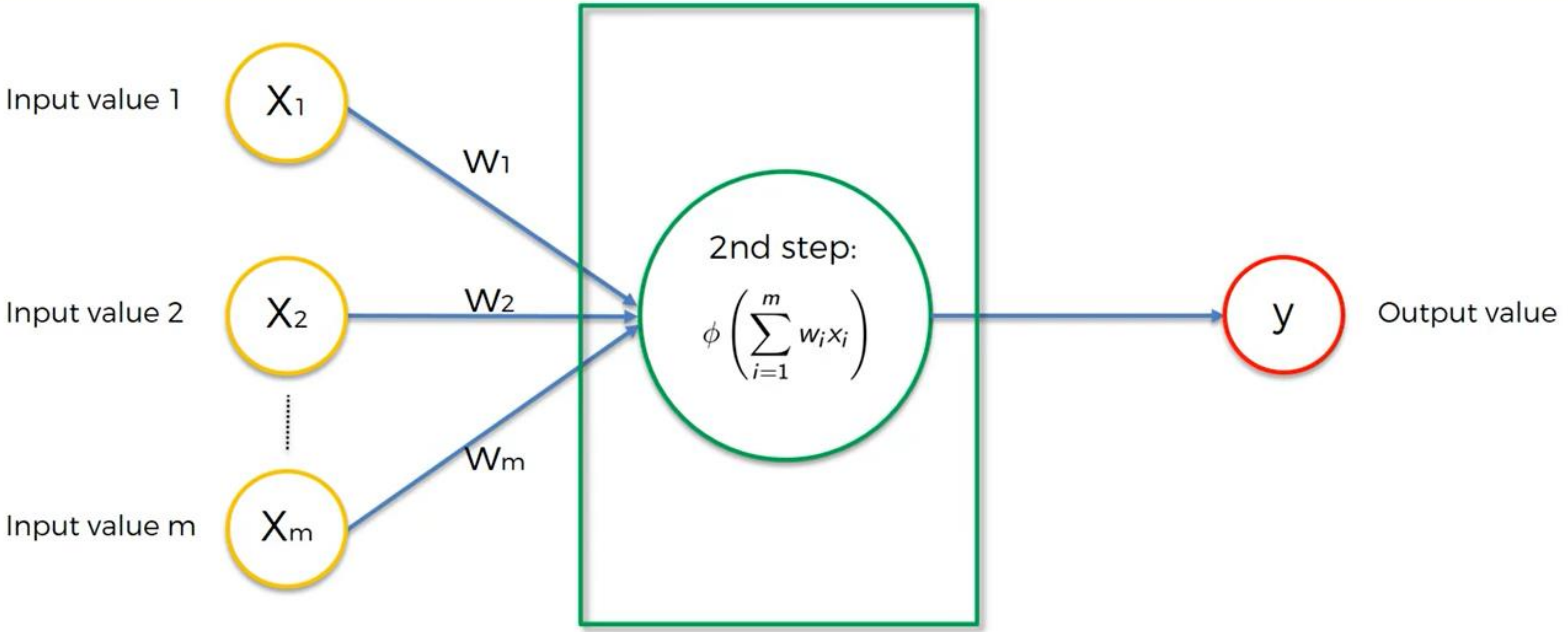
The Neuron


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

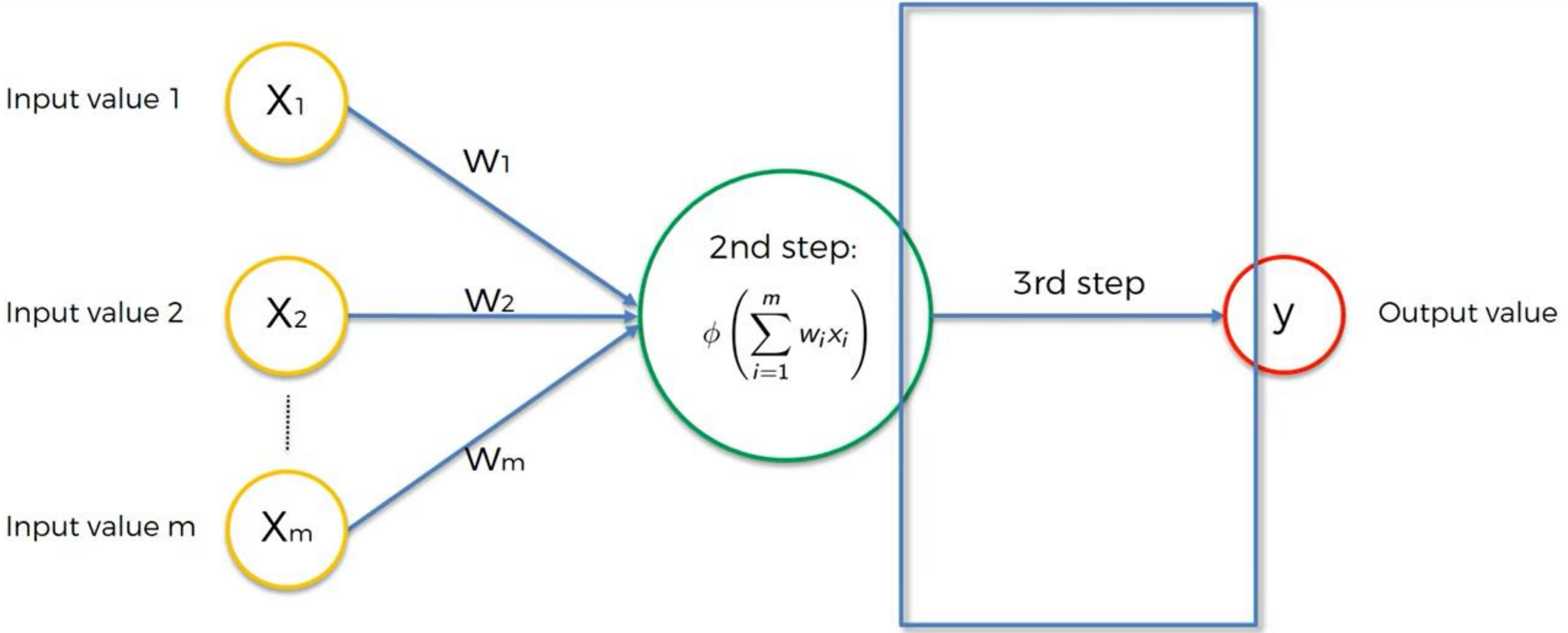
The Neuron


Input value 1
Input value 2
Input value m
X_1
X_2
X_m
W_1
W_2
W_m
1st step:
$\sum_{i=1}^{m} w_i x_i$
y
Output value

The Neuron


Input value 1
Input value 2
Input value m
X_1
X_2
X_m
W_1
W_2
W_m
2nd step:
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
y
Output value

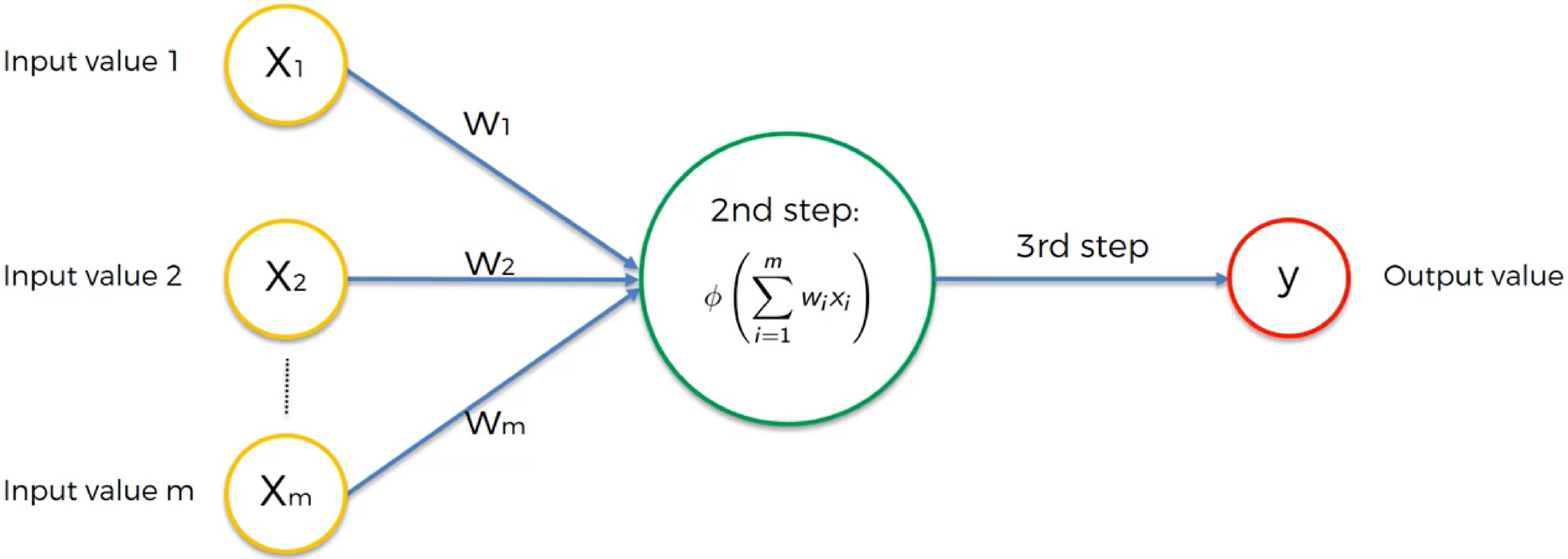
The Neuron



Input value 1
Input value 2
Input value m
X_1
X_2
X_m
w_1
w_2
w_m
2nd step:
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
3rd step
y
Output value

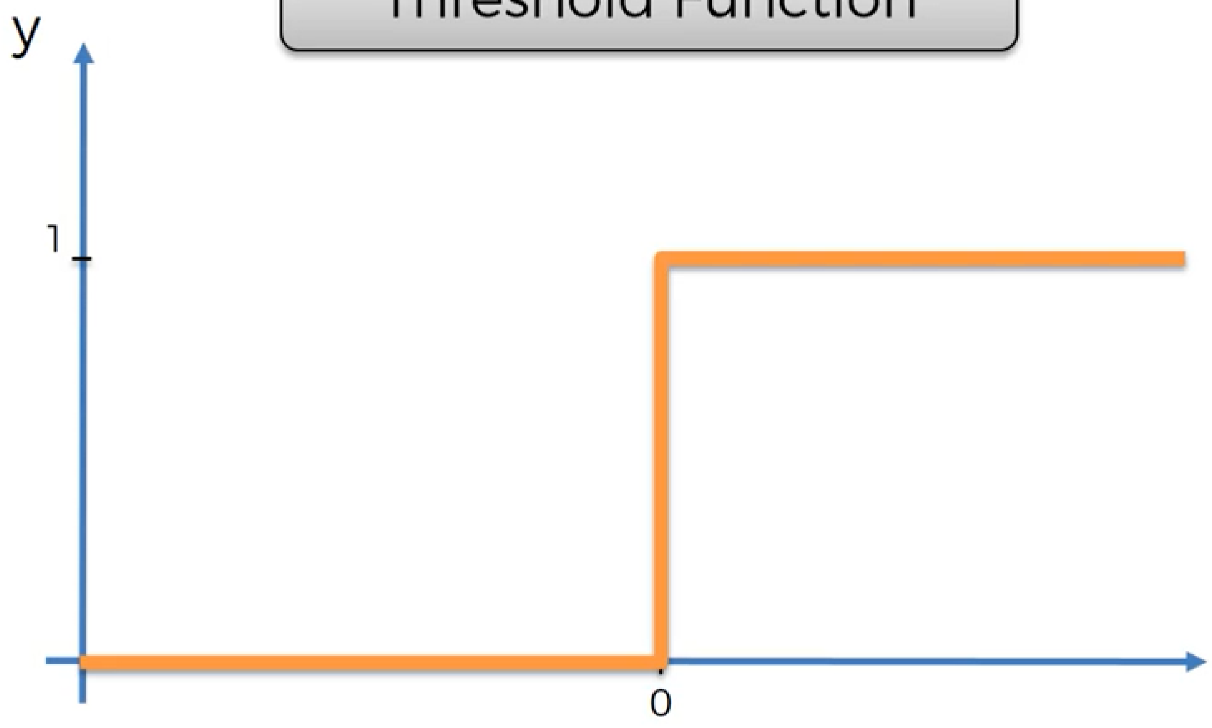
The Activation Function

The Activation Function



The Activation Function

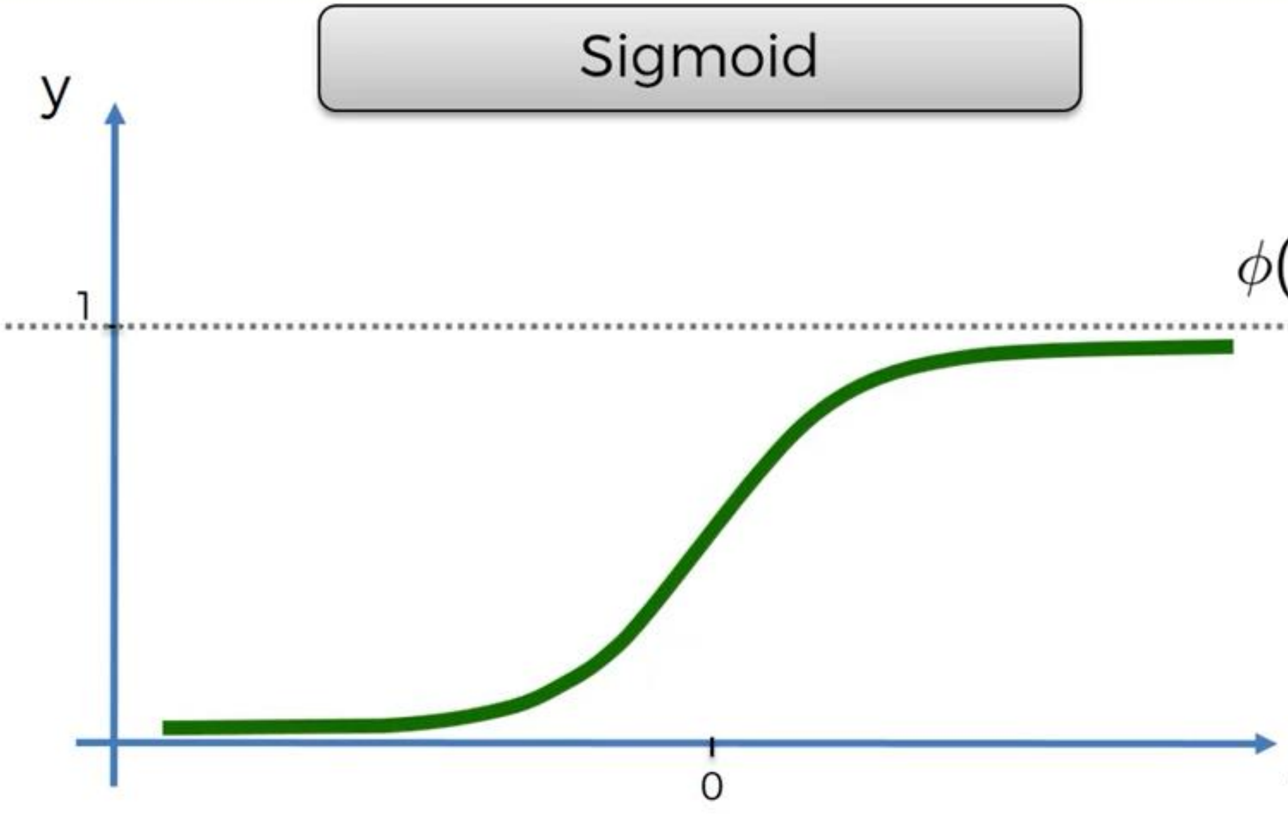
Threshold Function



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

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

The Activation Function

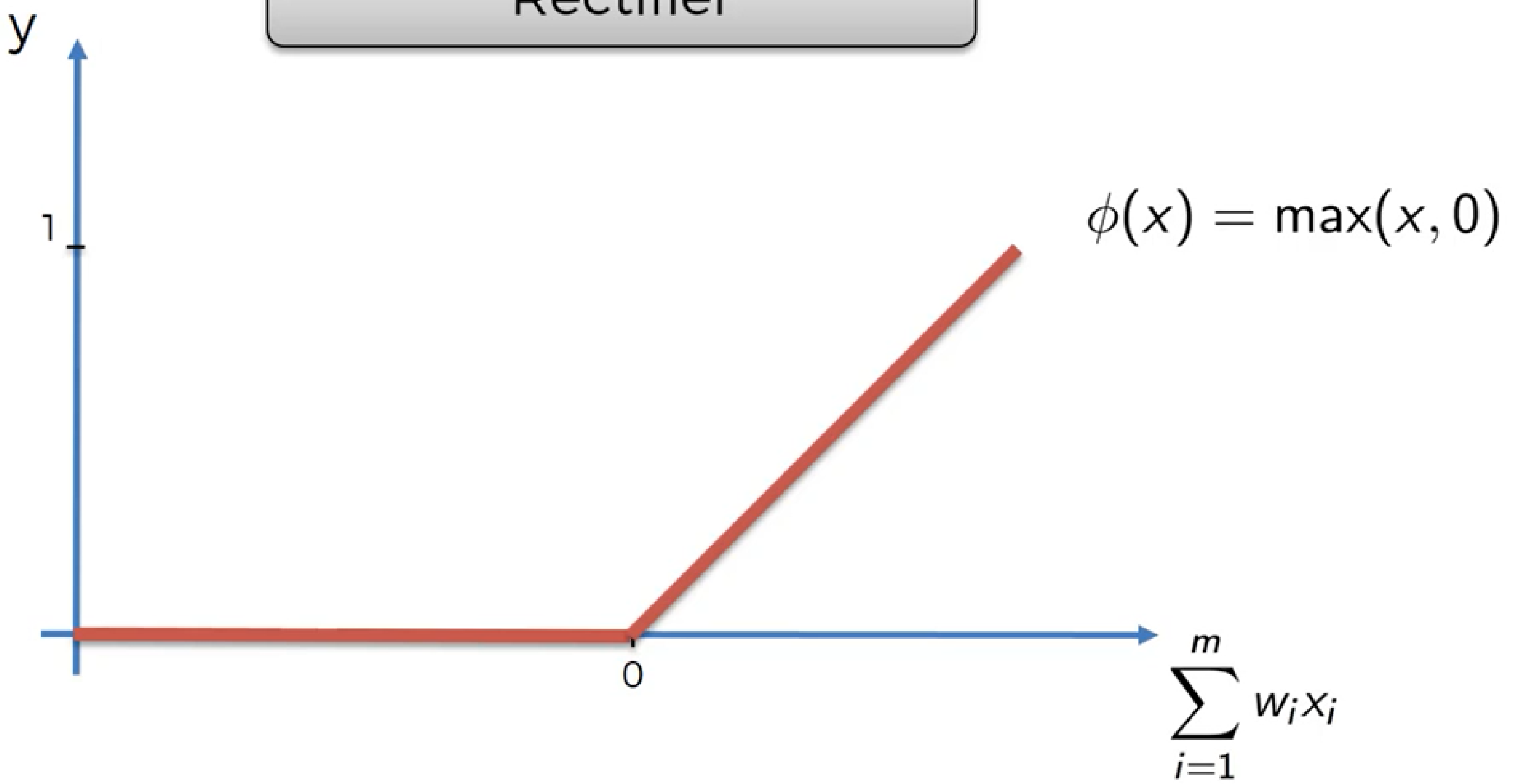


$$\phi(x) = \frac{1}{1+e^{-x}}$$

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

The Activation Function

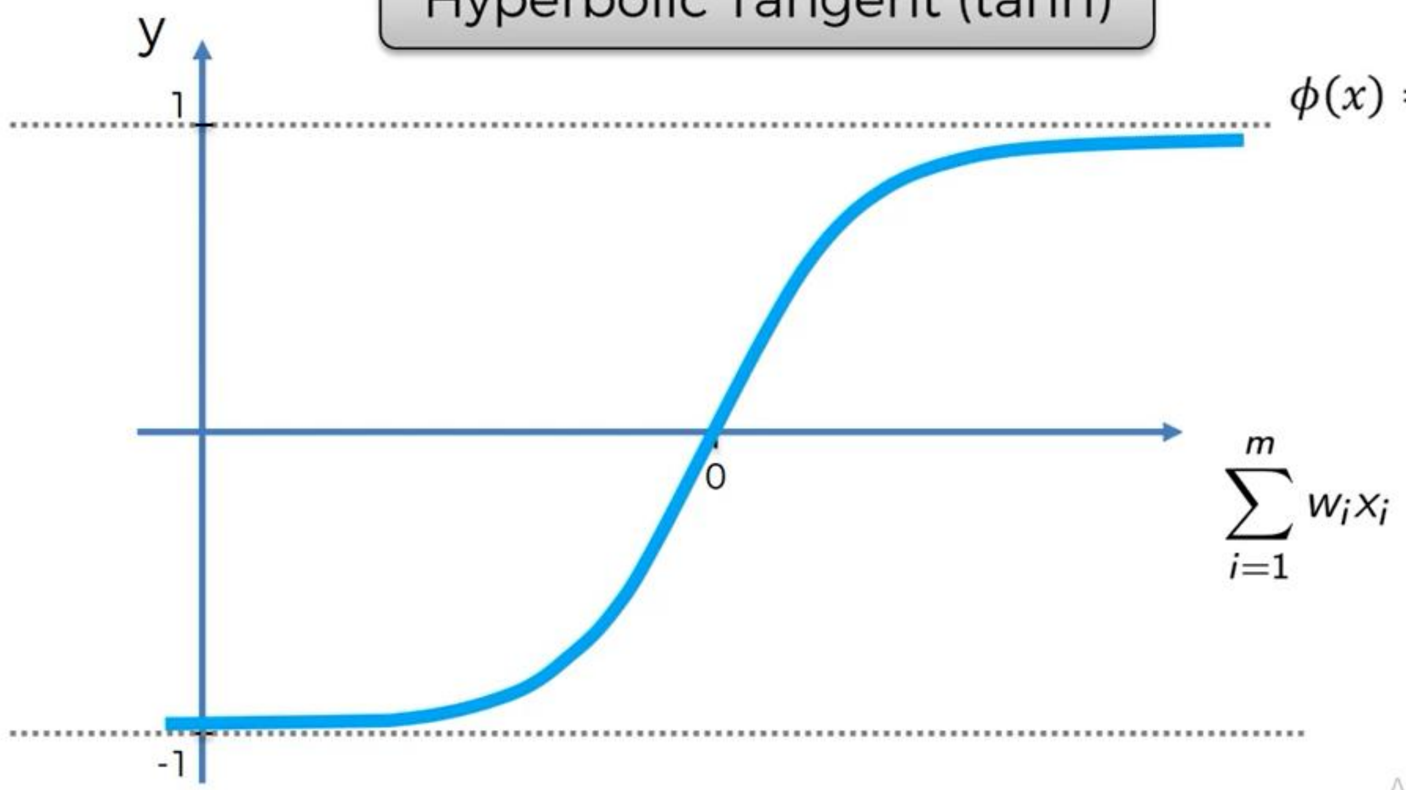
Rectifier



The Activation Function

Hyperbolic Tangent (tanh)

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

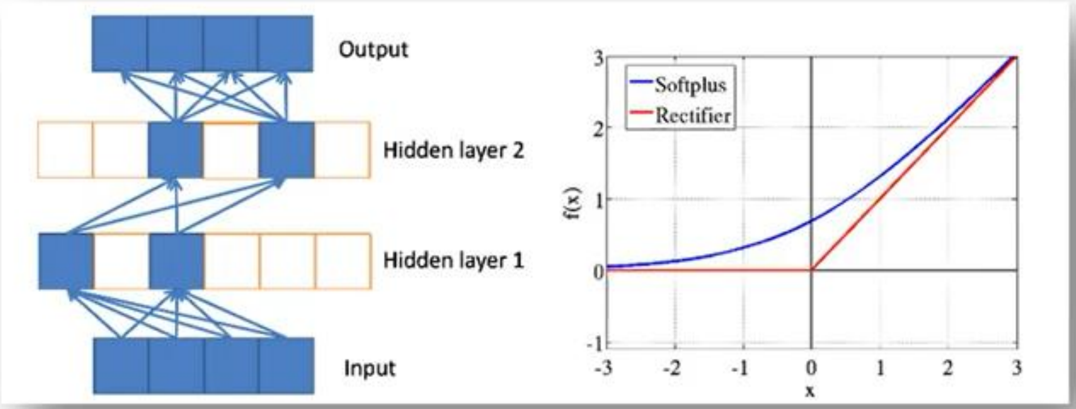


The Activation Function

Additional Reading:

Deep sparse rectifier neural networks

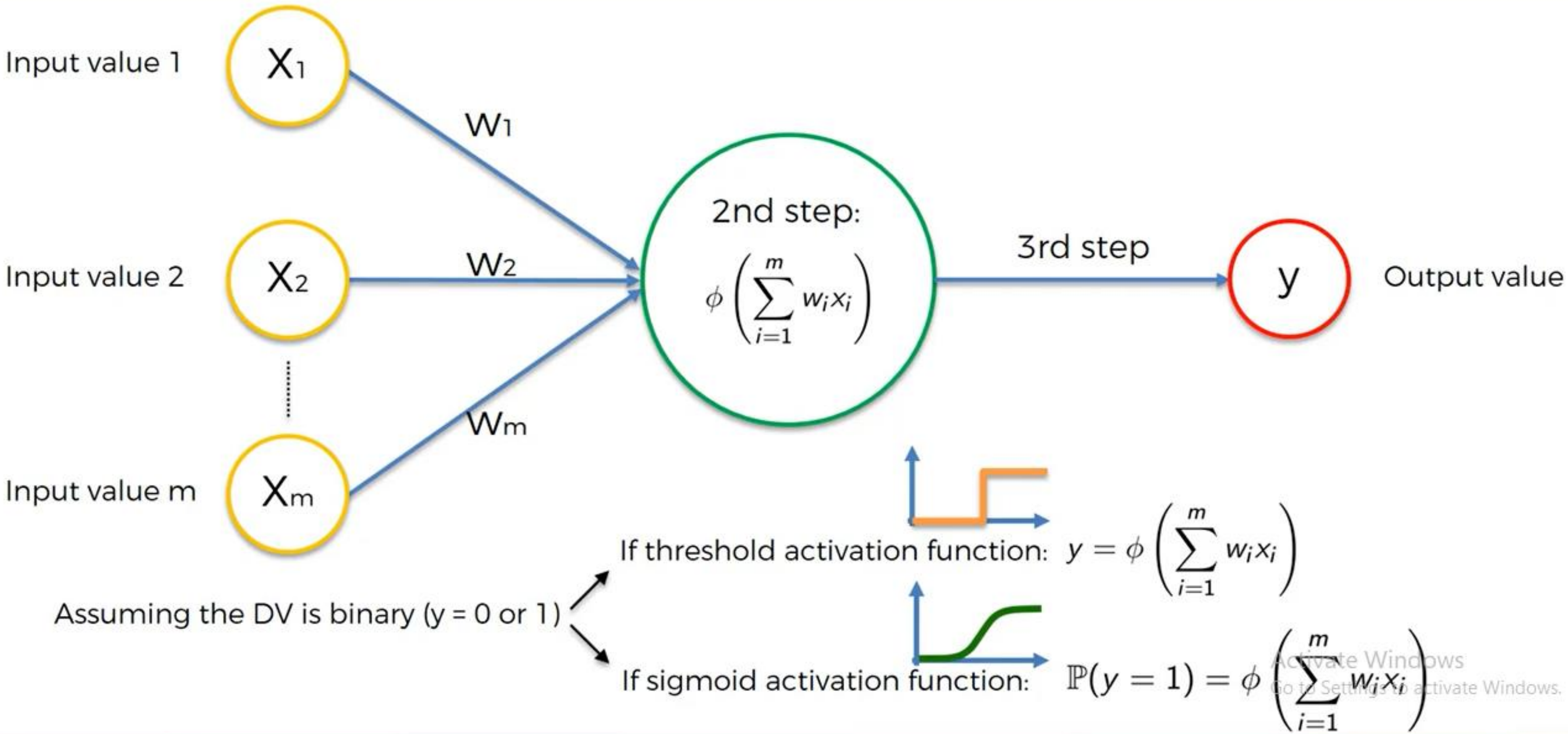
By Xavier Glorot et al. (2011)



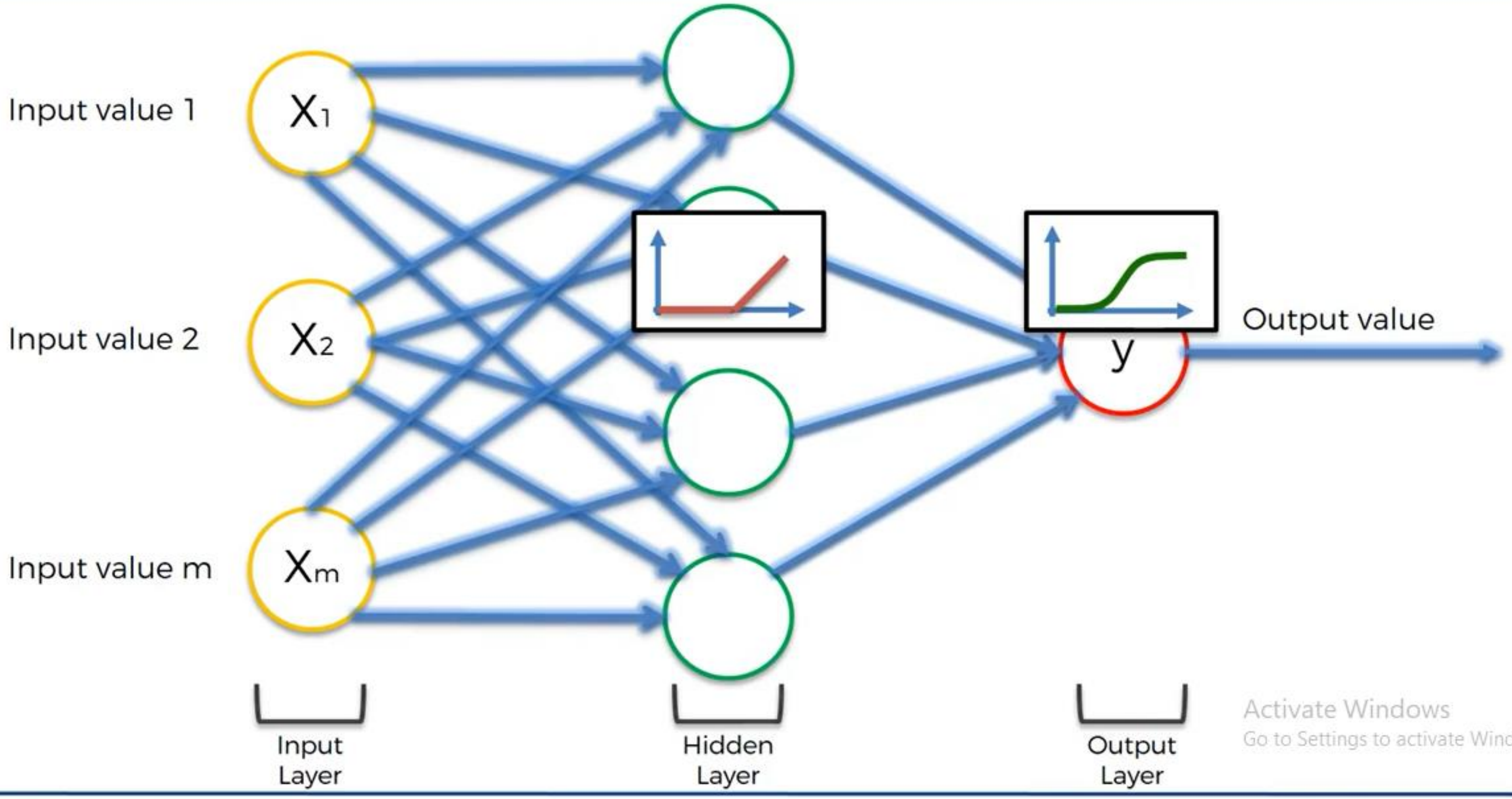
Link:

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

The Activation Function



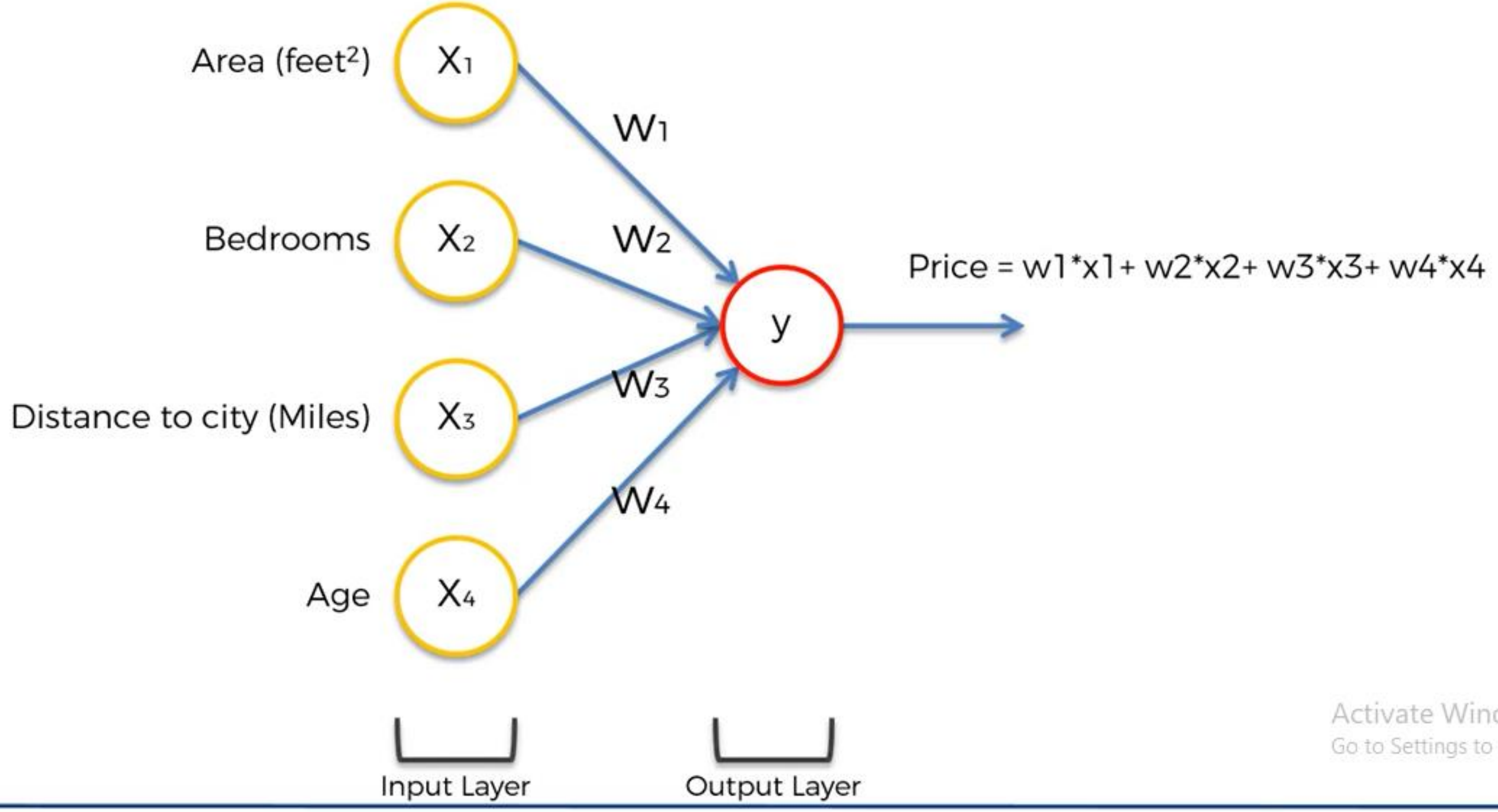
The Activation Function



How do NNs Work?

How Do Neural Networks Work?

How Do Neural Networks Work?

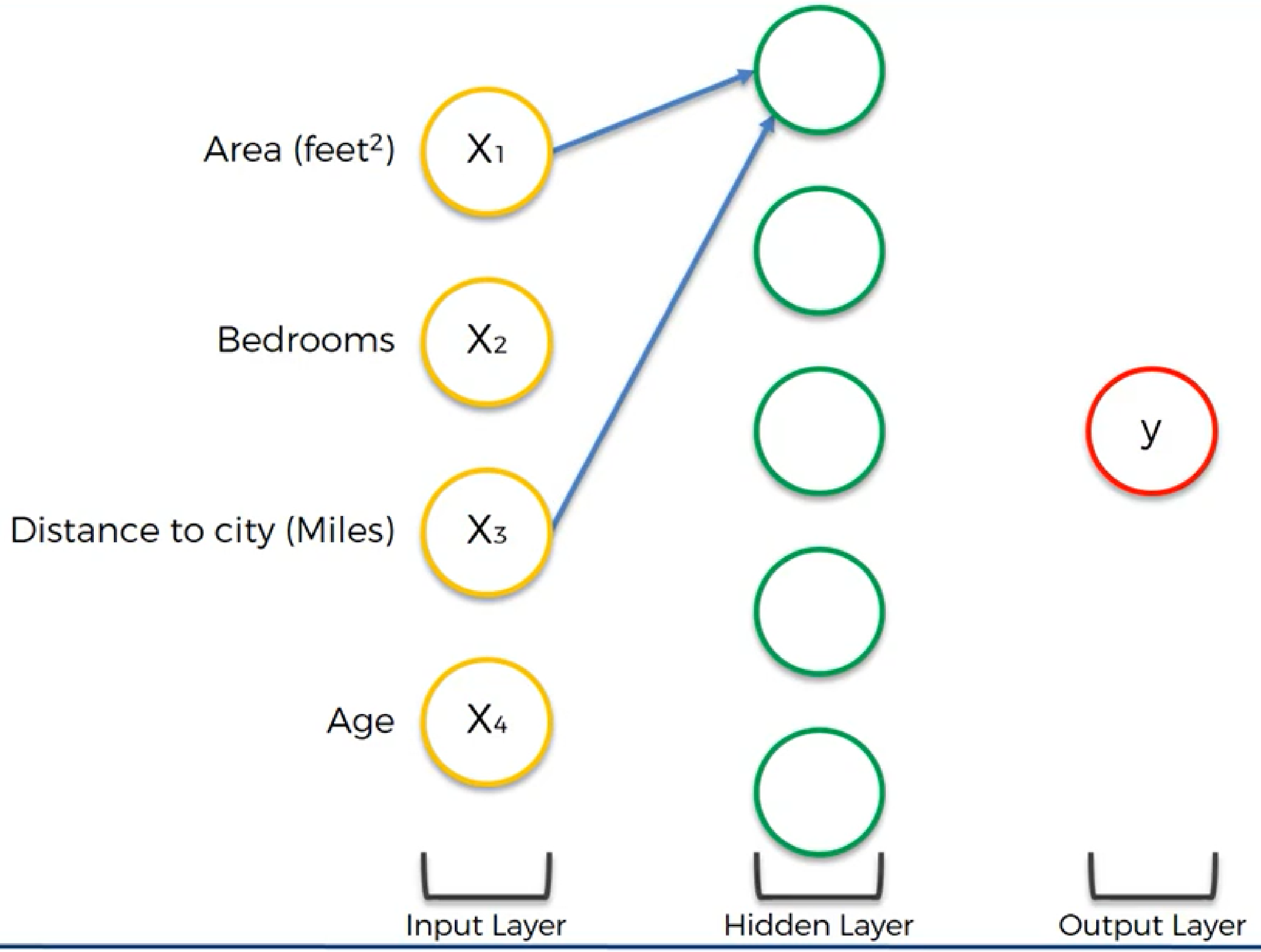


How Do Neural Networks Work?

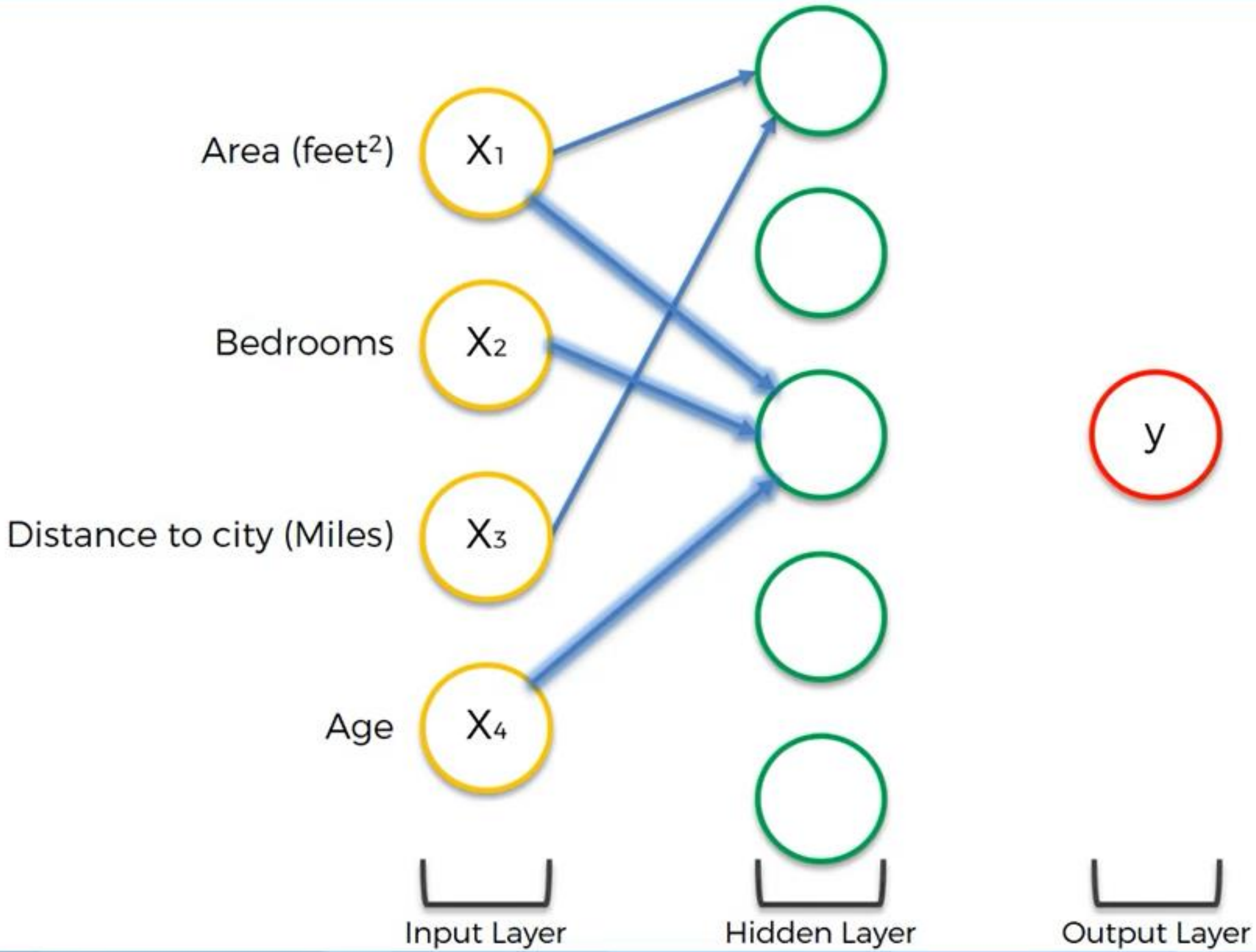
How Do Neural Networks Work?

How Do Neural Networks Work?

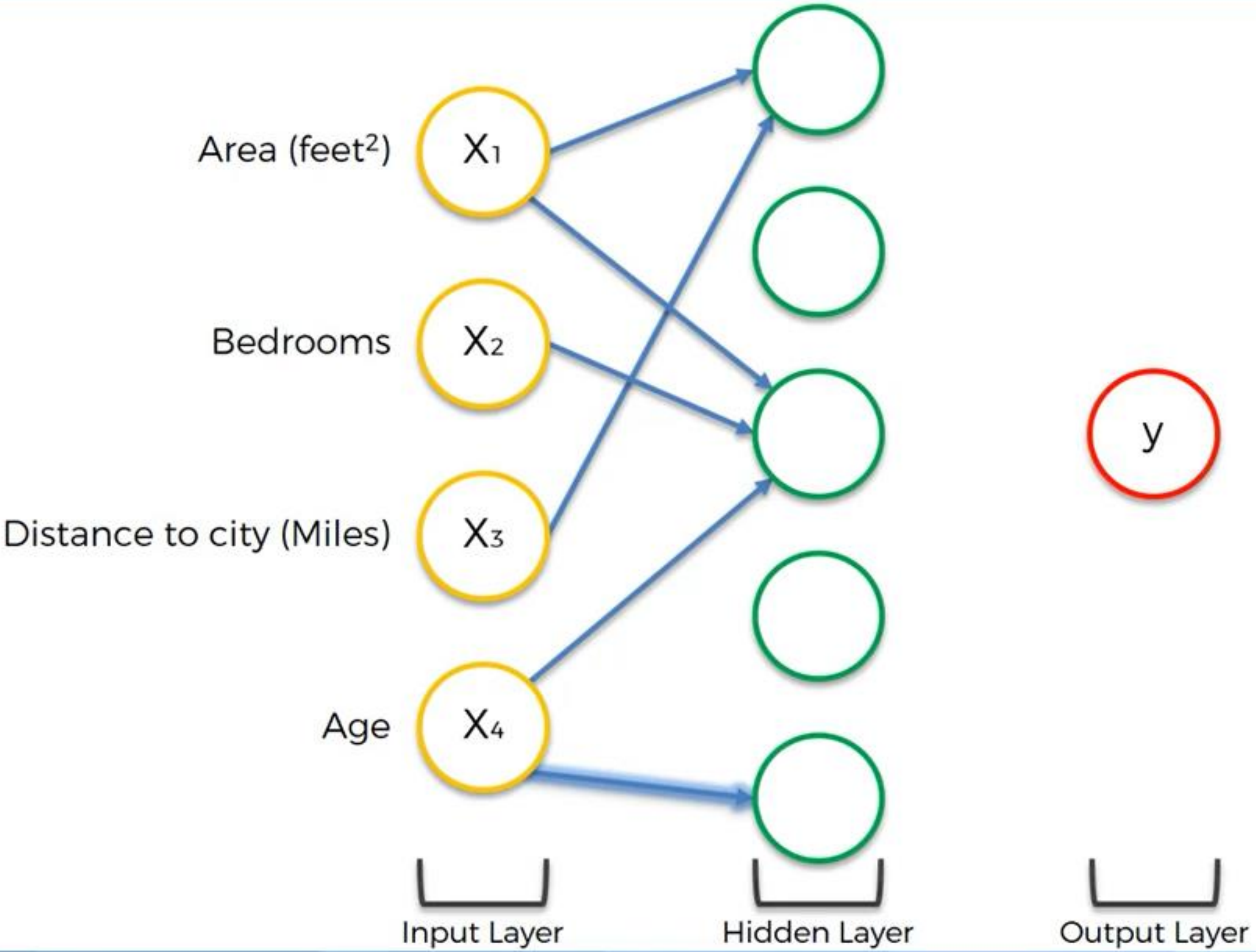
How Do Neural Networks Work?



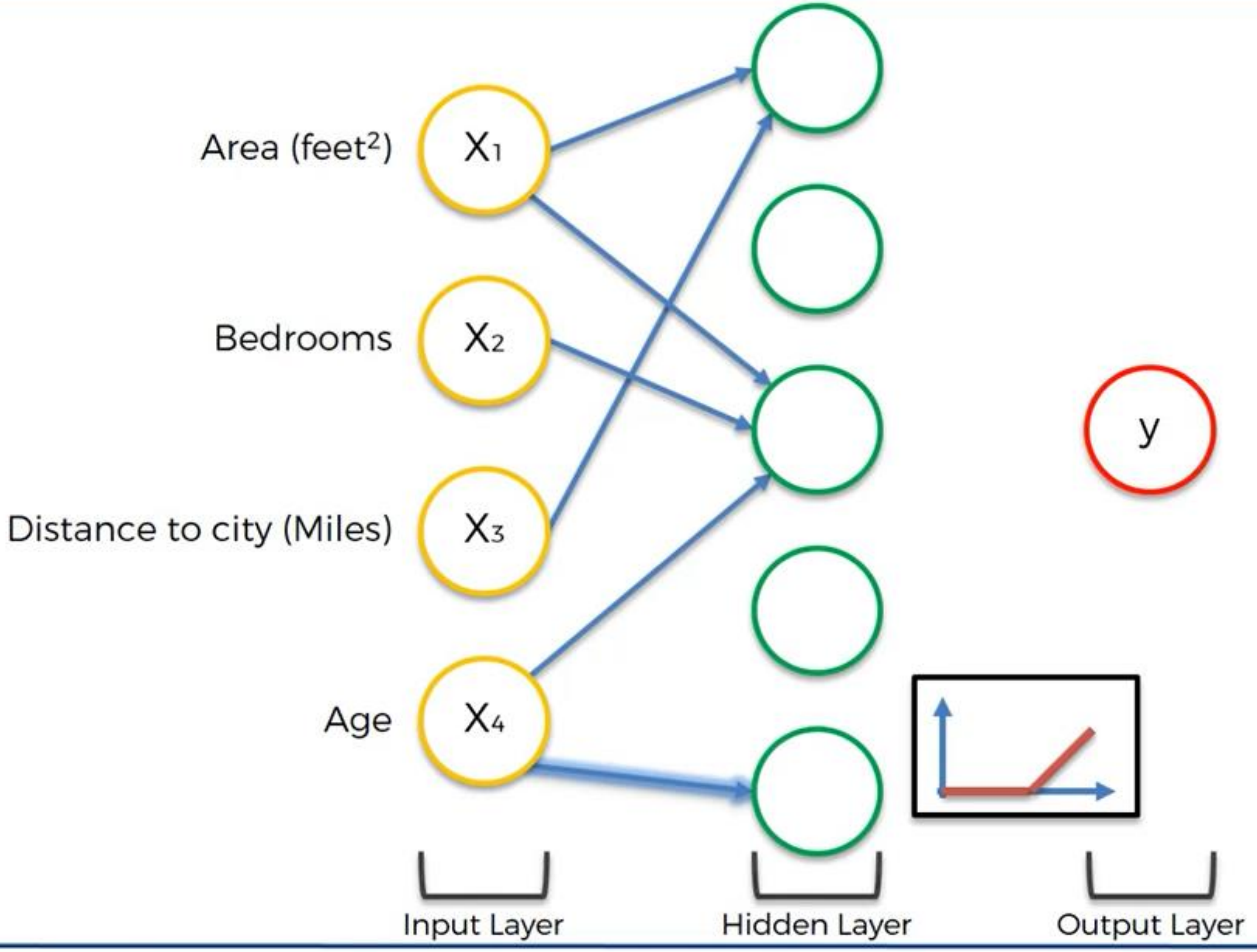
How Do Neural Networks Work?



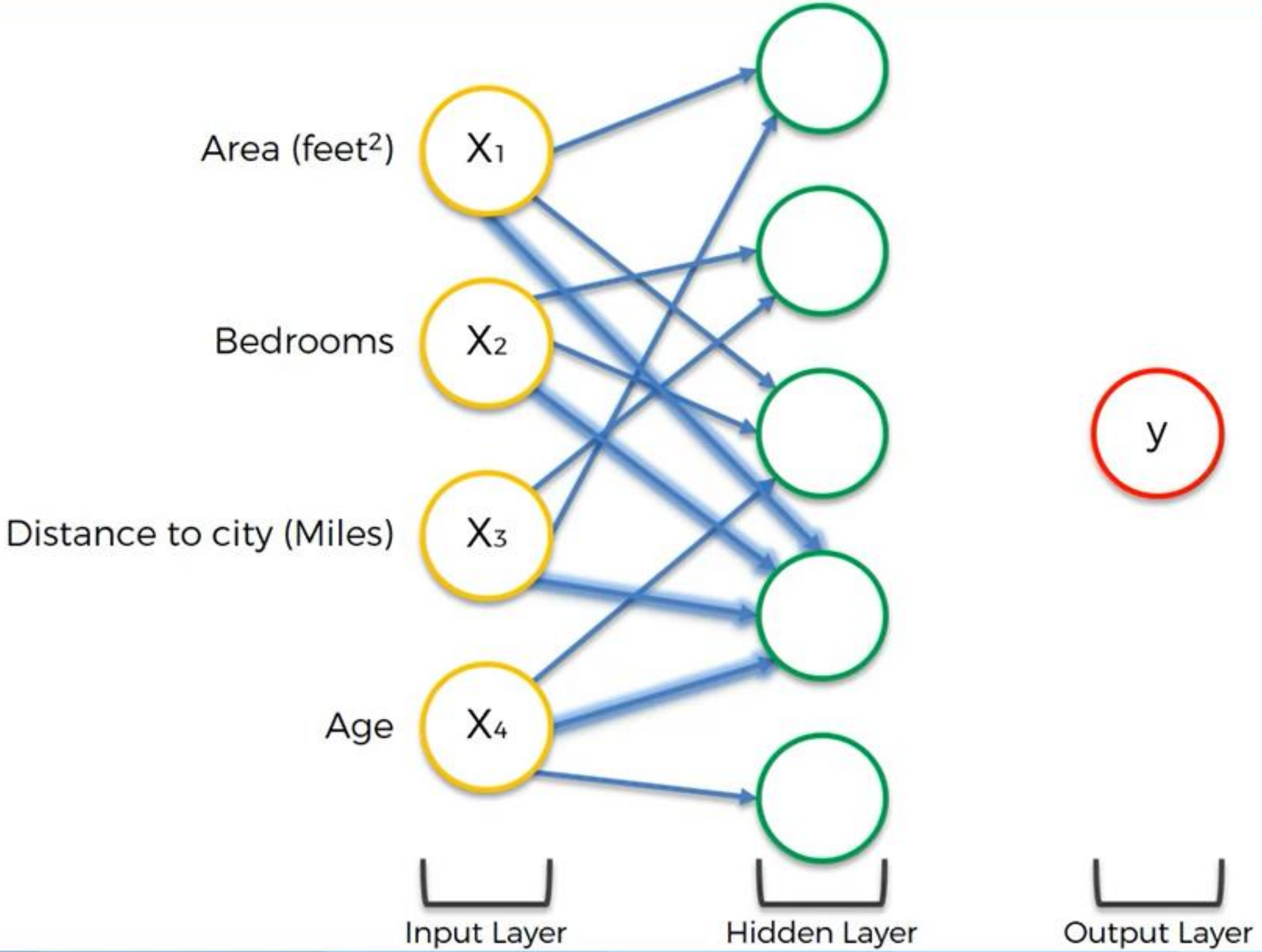
How Do Neural Networks Work?



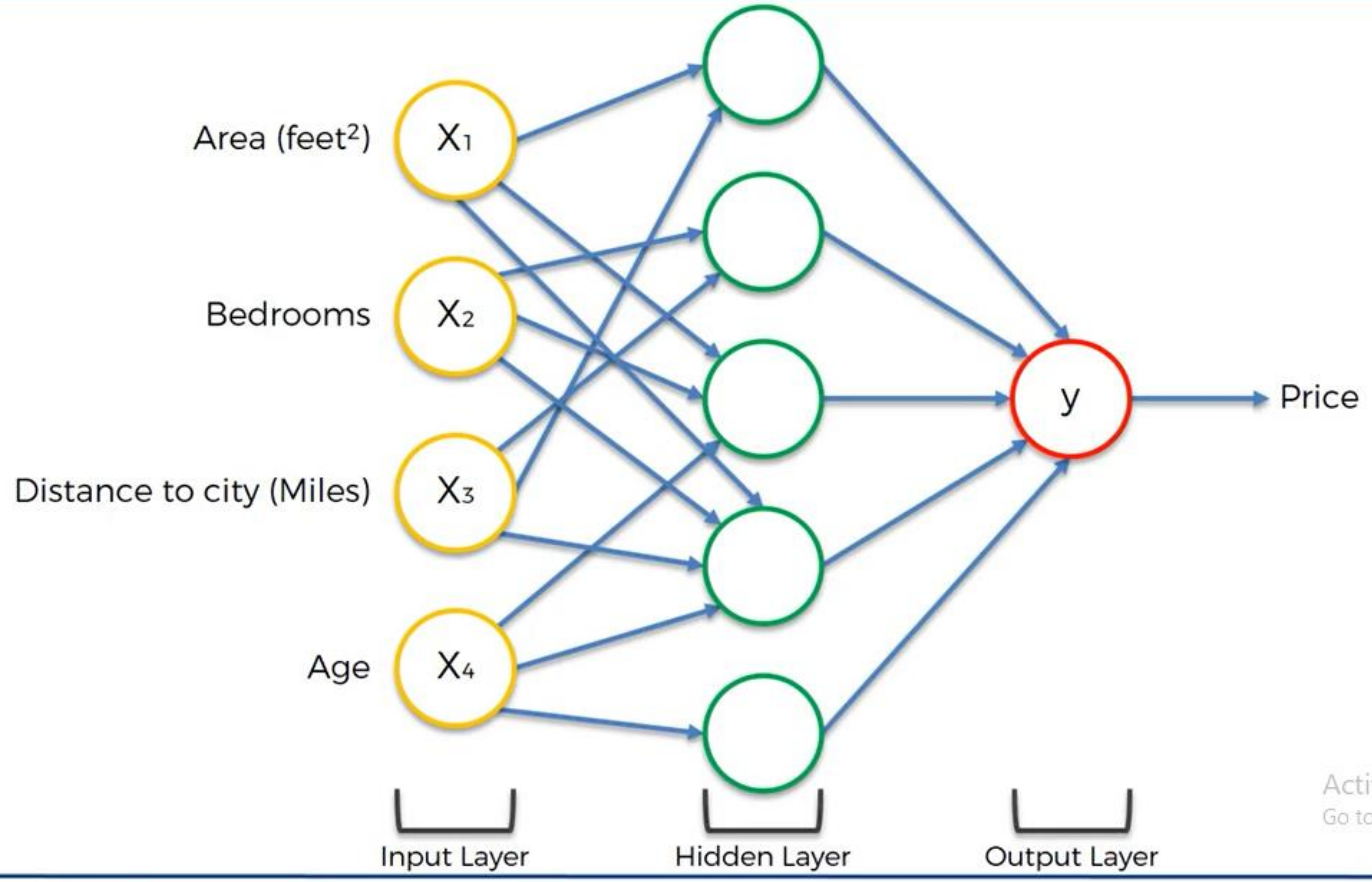
How Do Neural Networks Work?



How Do Neural Networks Work?



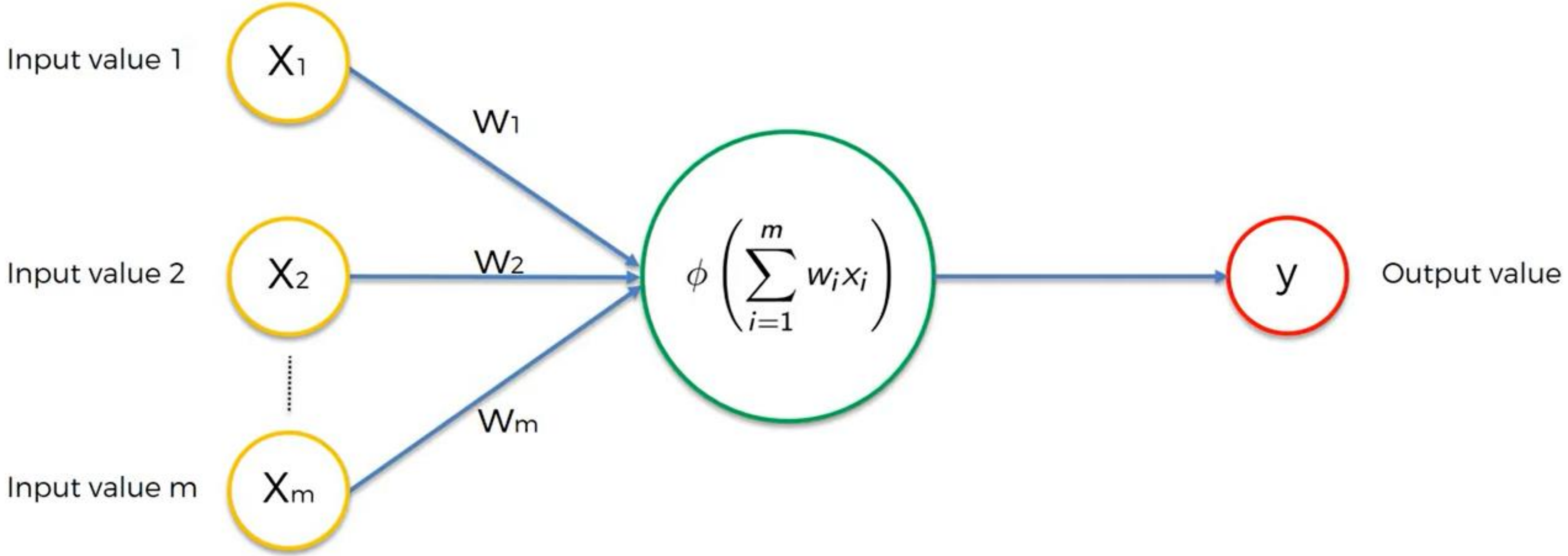
How Do Neural Networks Work?



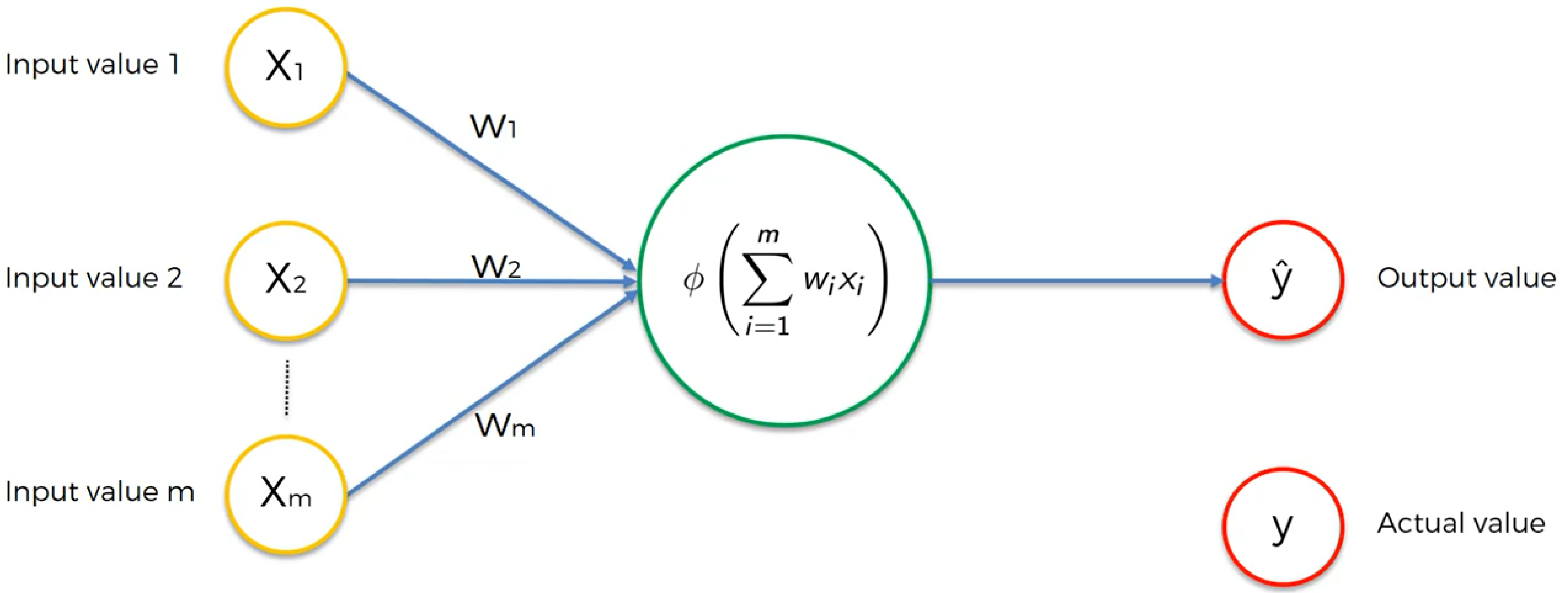
How do NNs Learn?

How do Neural Networks learn?

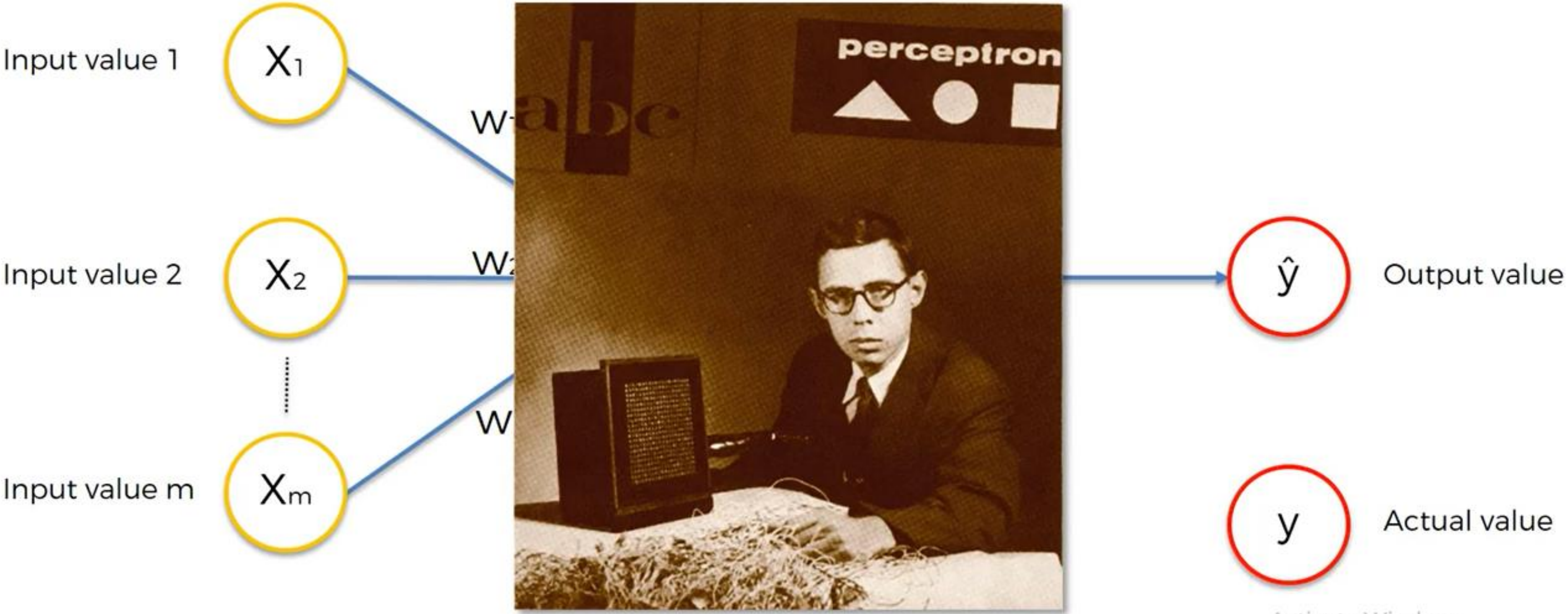
How do Neural Networks learn?



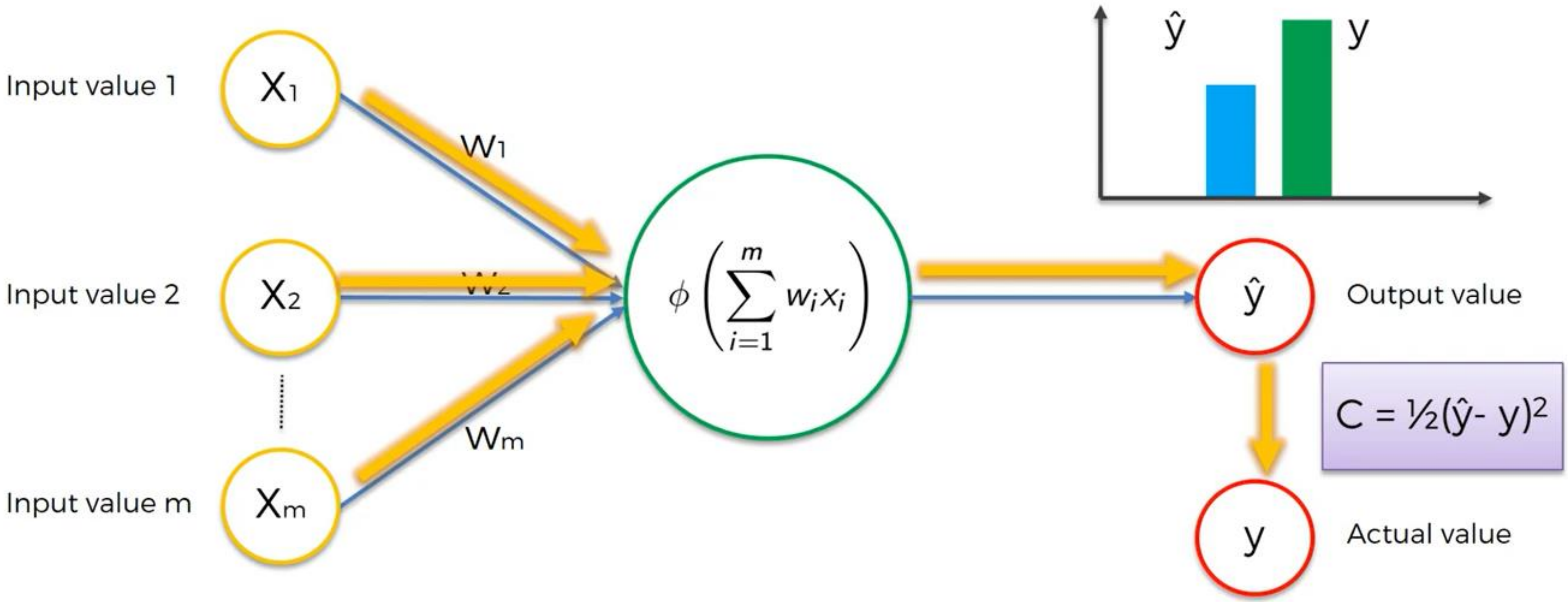
How do Neural Networks learn?



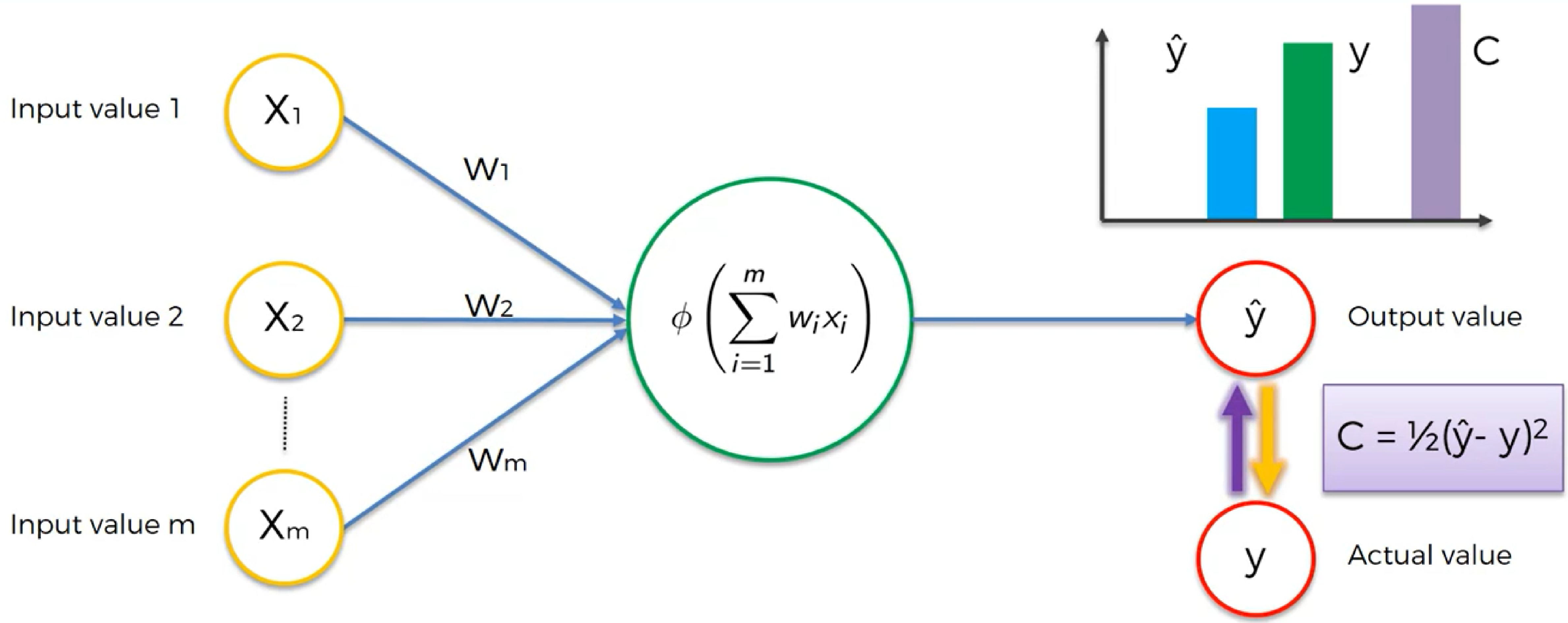
How do Neural Networks learn?



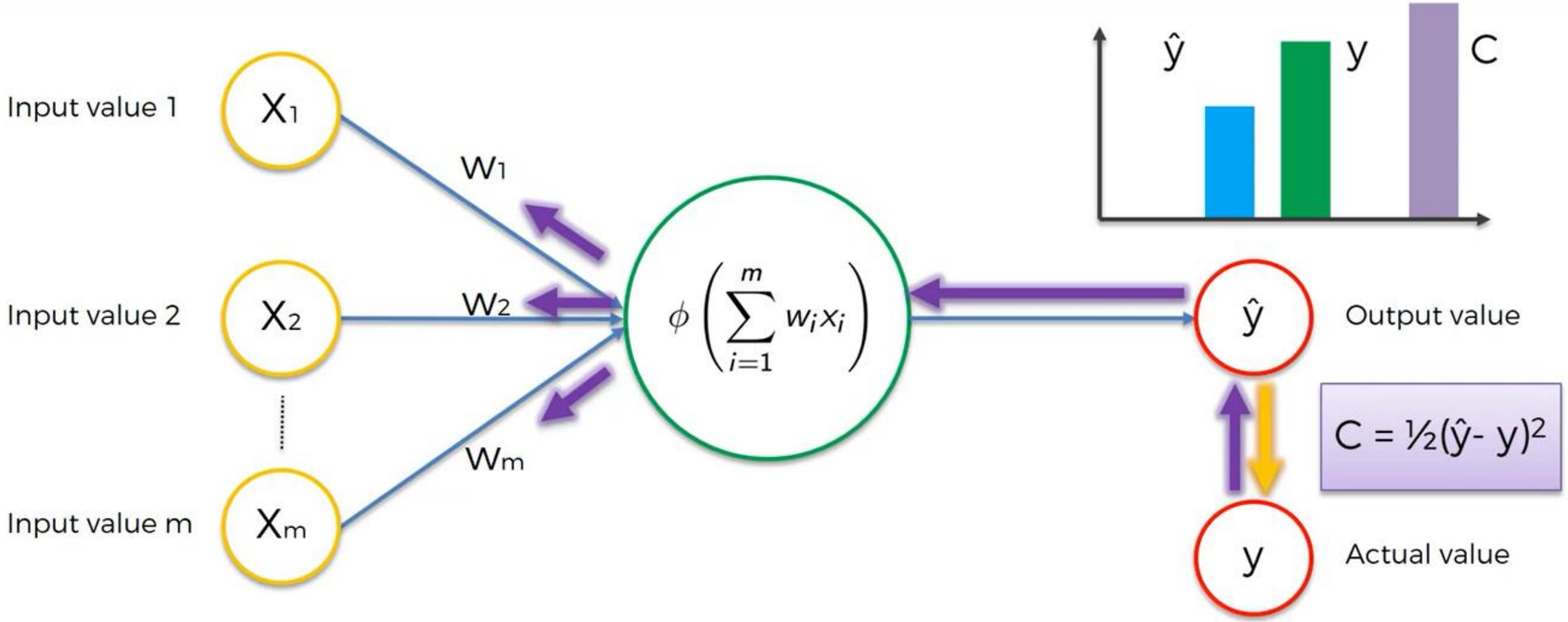
How do Neural Networks learn?



How do Neural Networks learn?

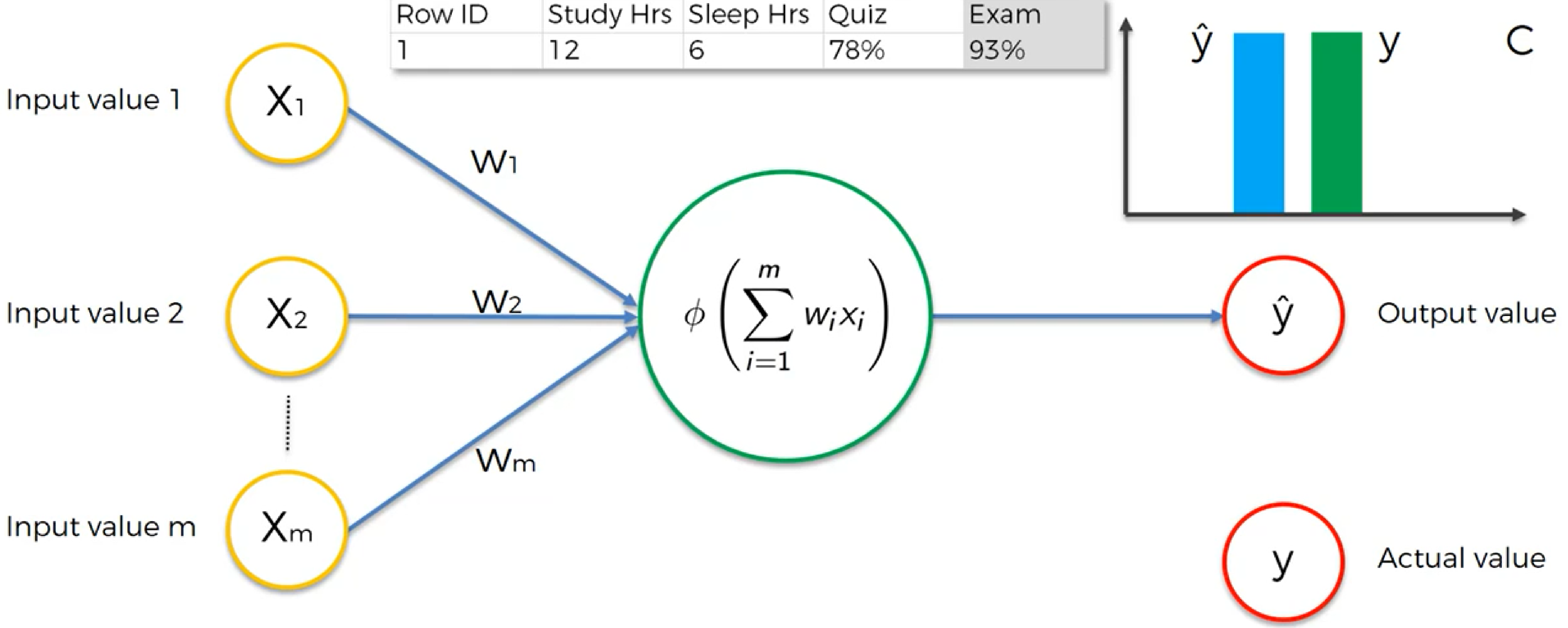


How do Neural Networks learn?



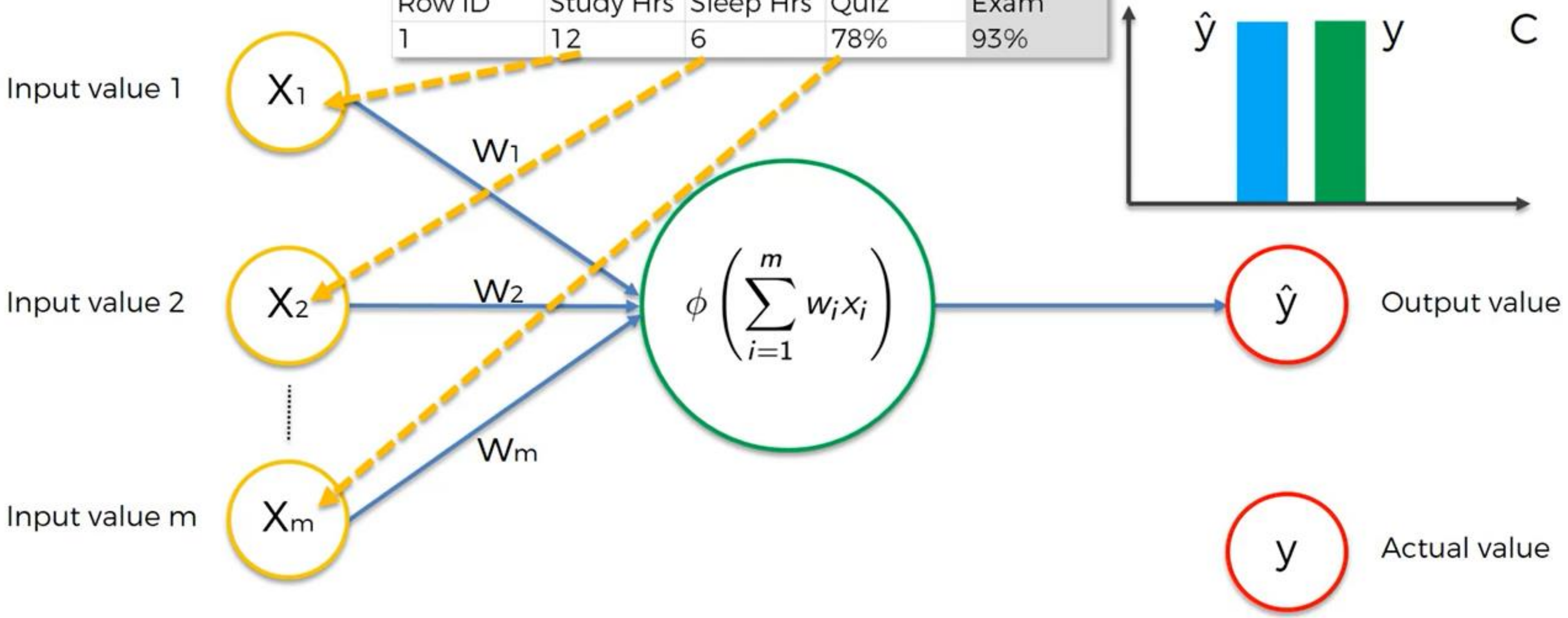
How do Neural Networks learn?

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



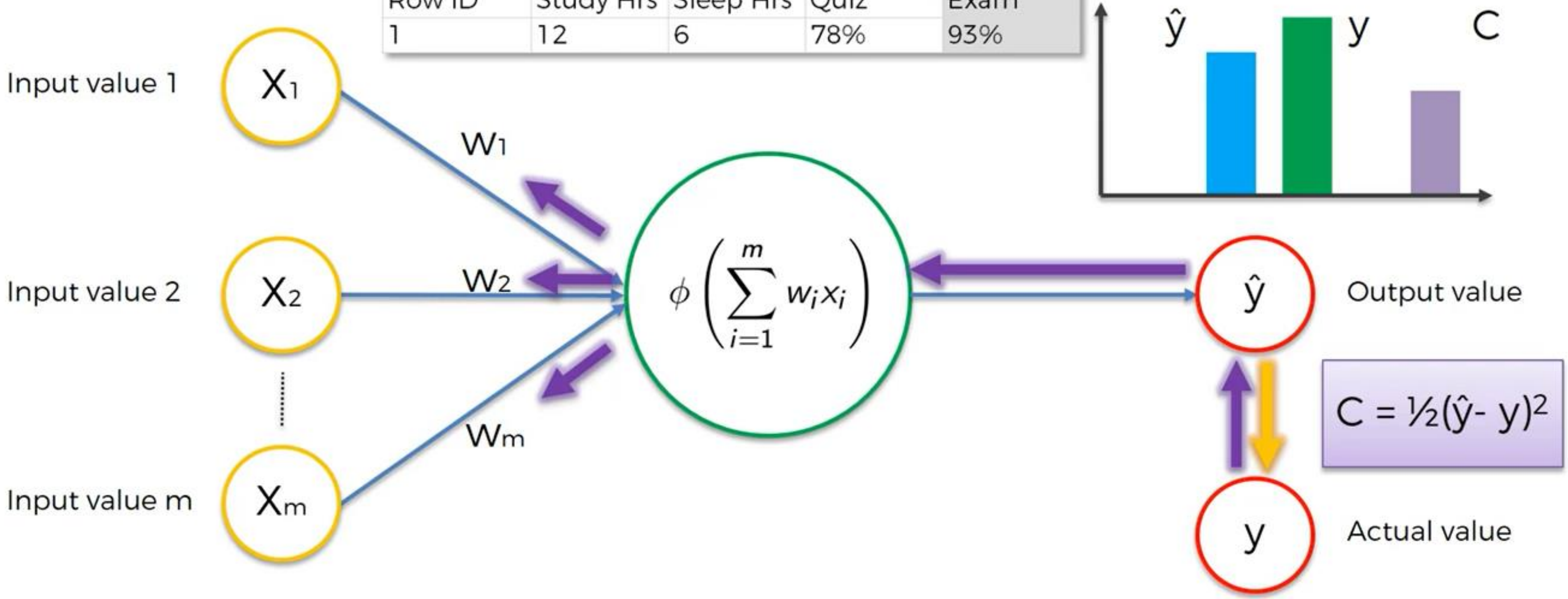
How do Neural Networks learn?

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



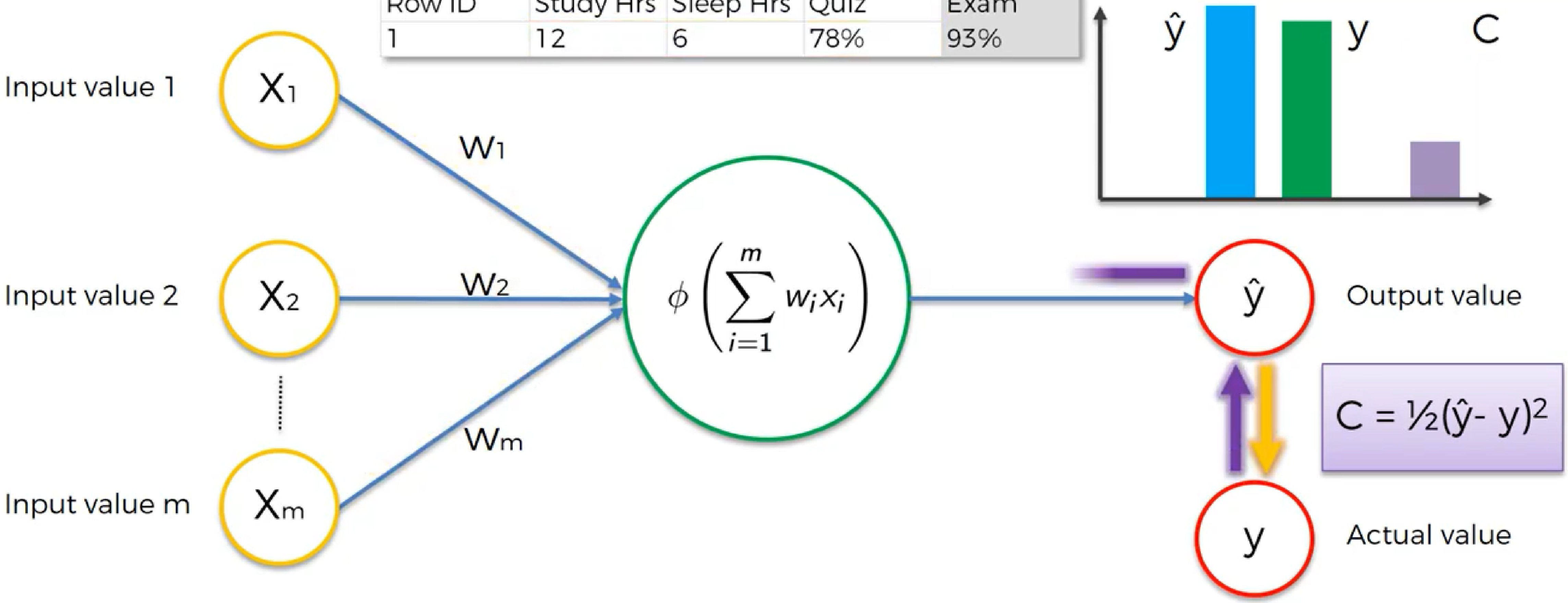
How do Neural Networks learn?

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



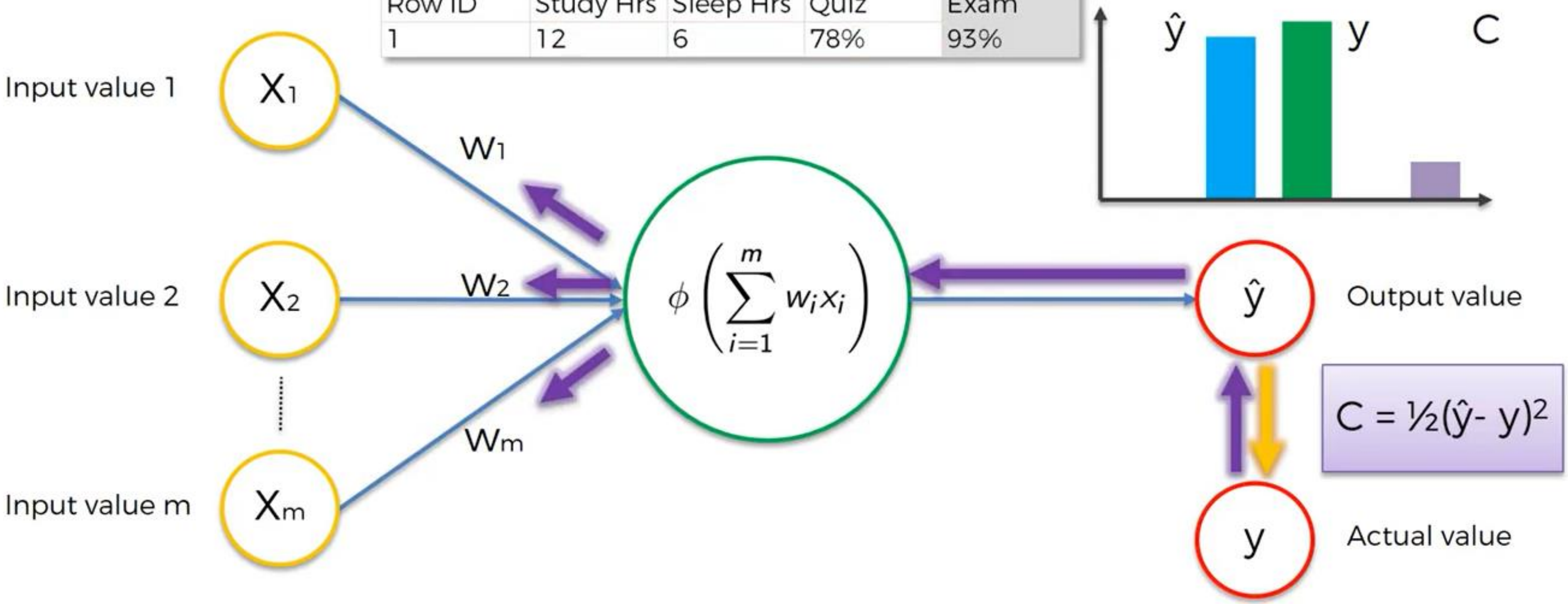
How do Neural Networks learn?

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

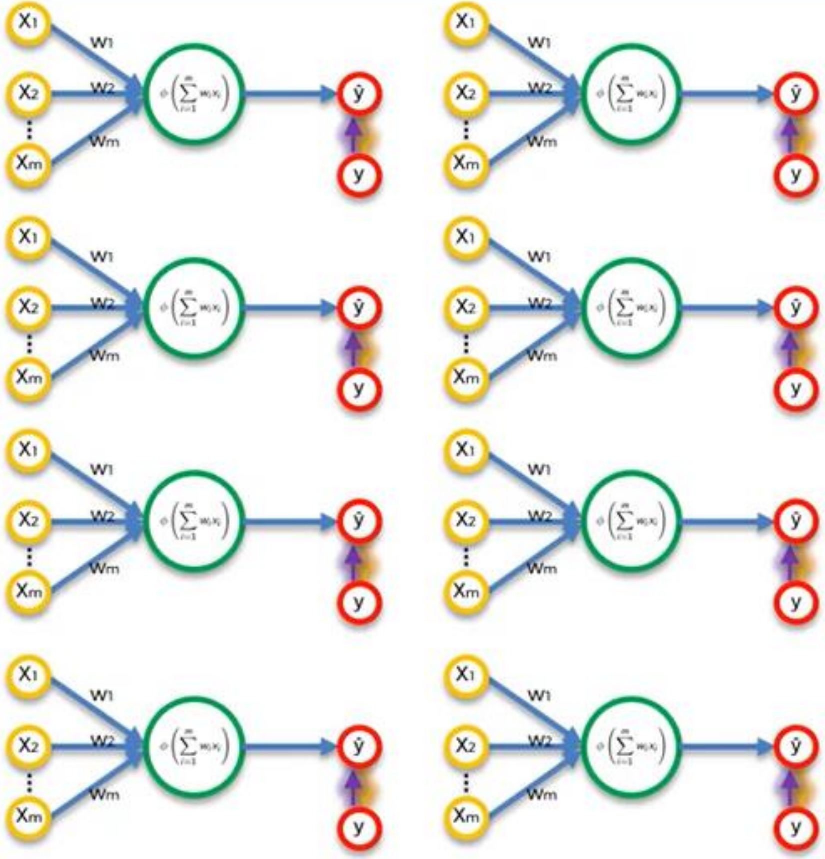


How do Neural Networks learn?

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

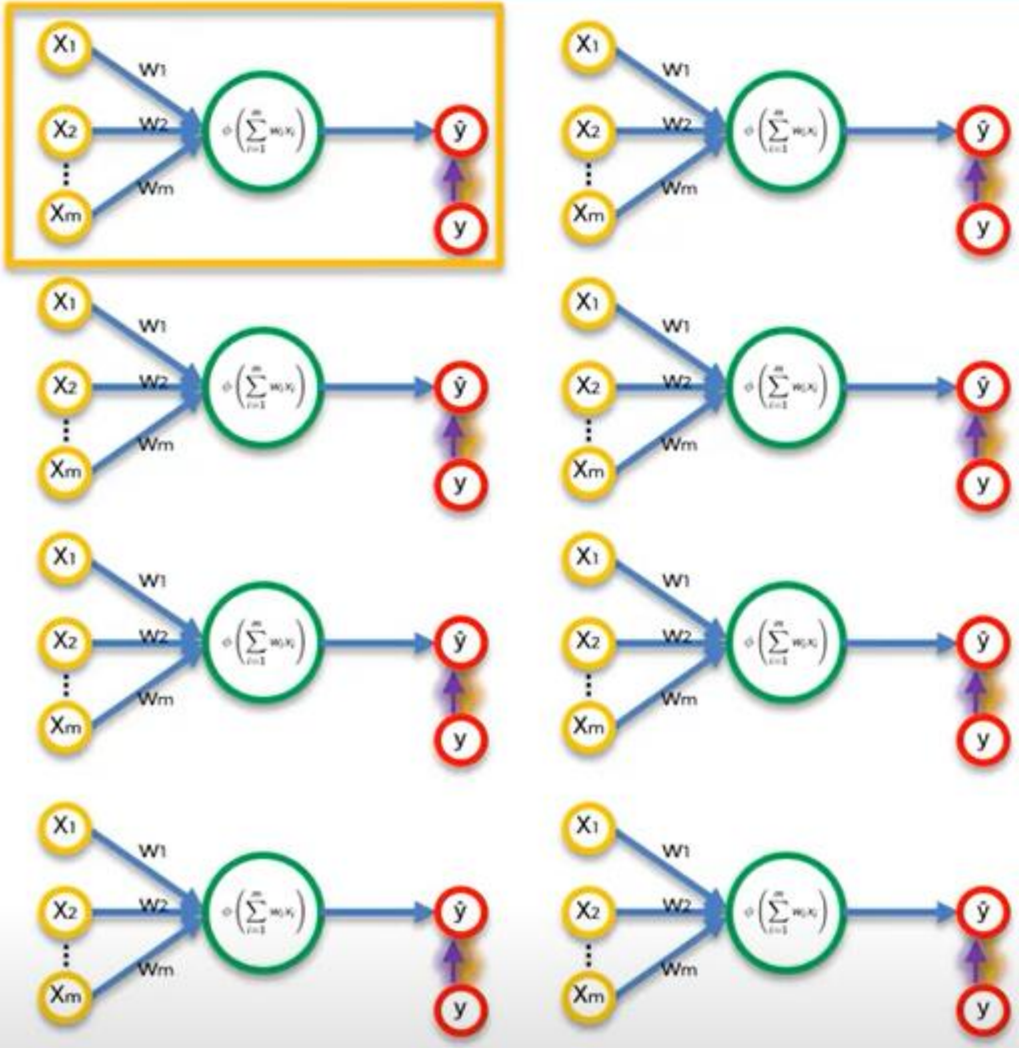


How do Neural Networks learn?

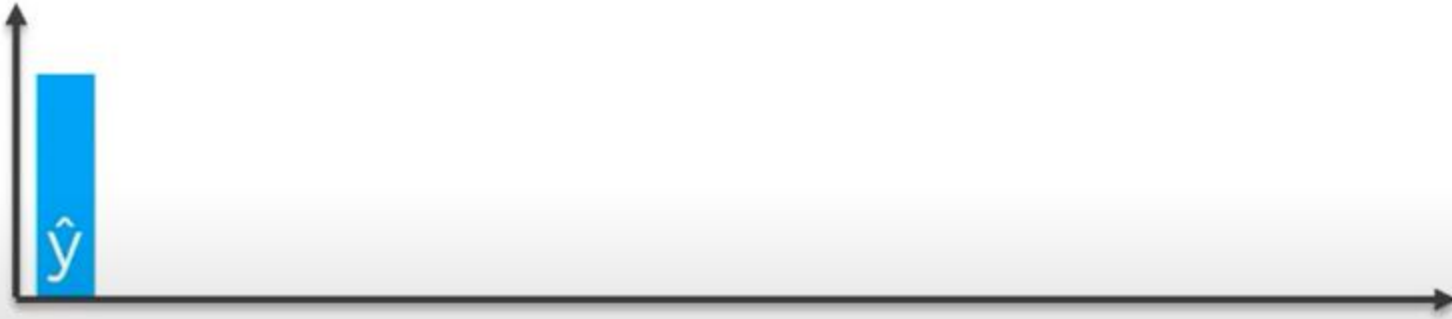


Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

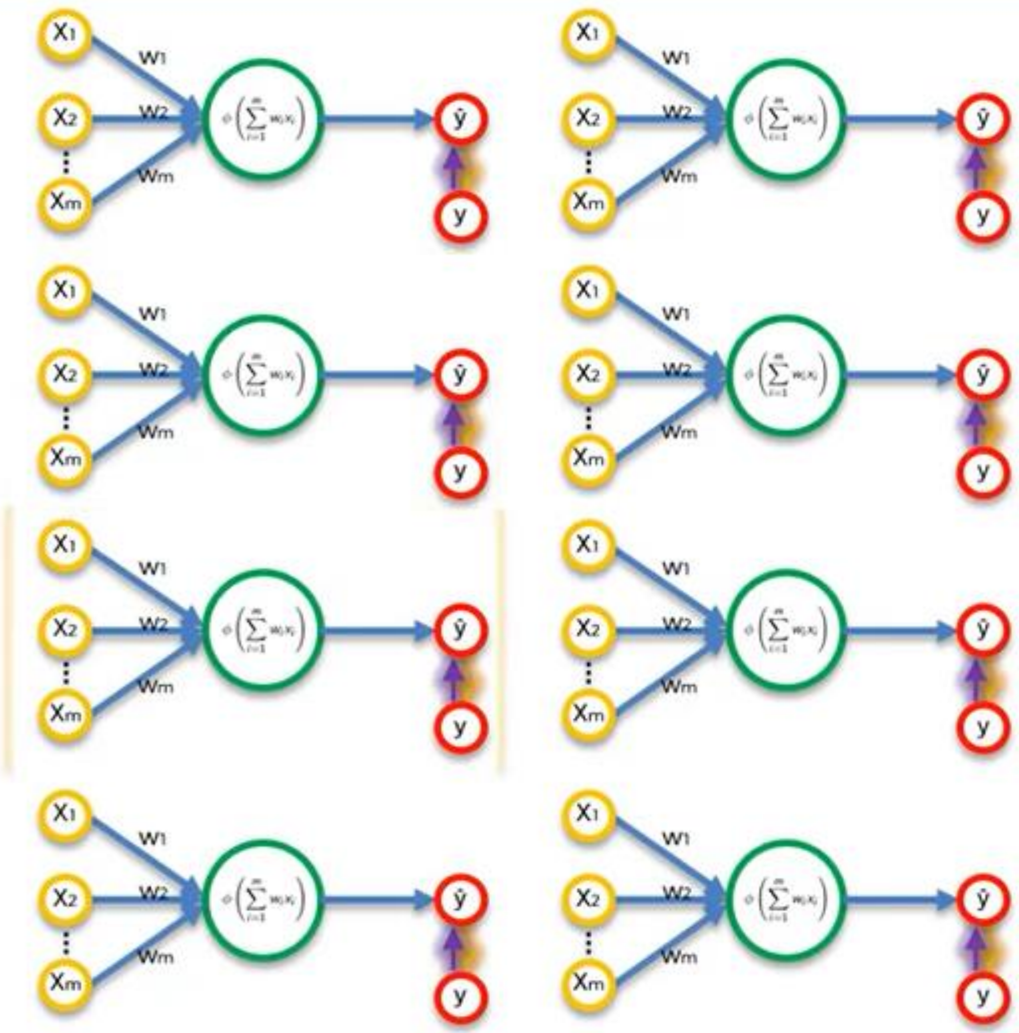
How do Neural Networks learn?



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%

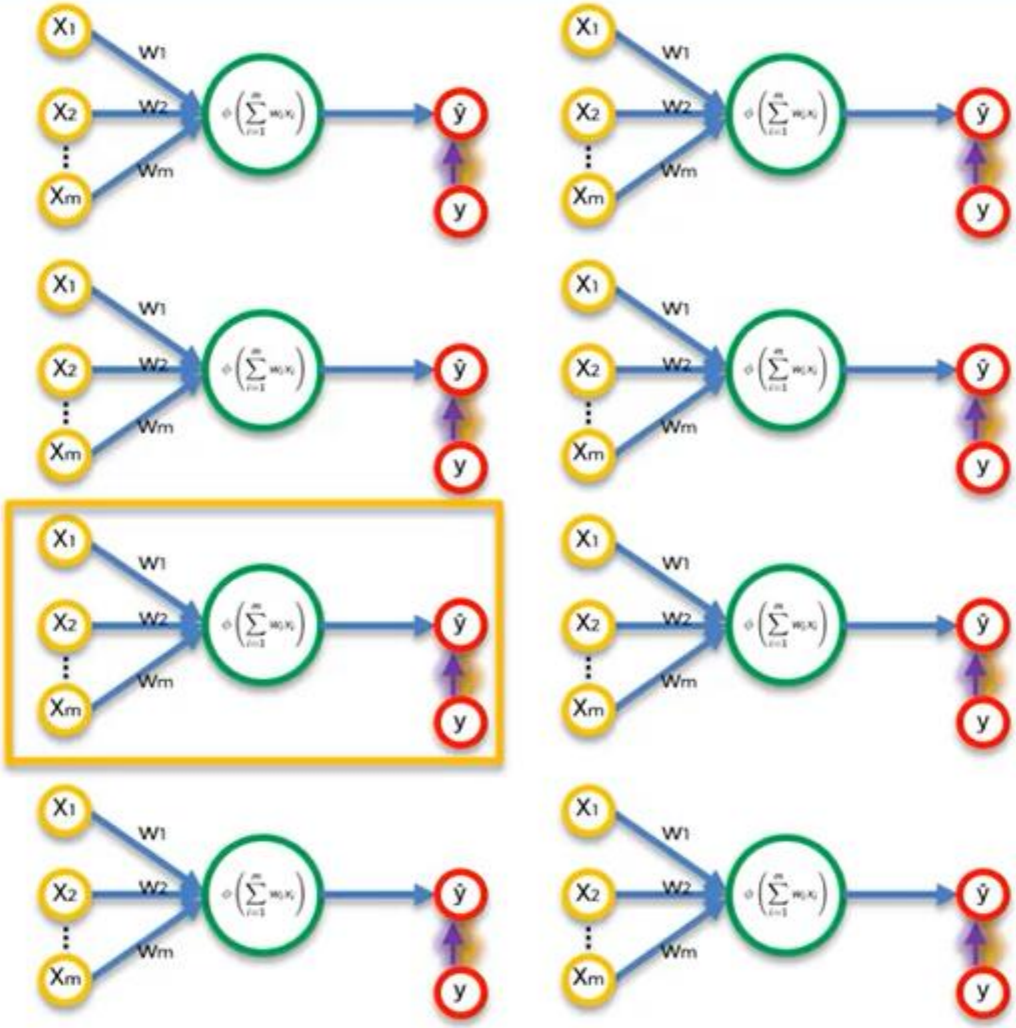


How do Neural Networks learn?



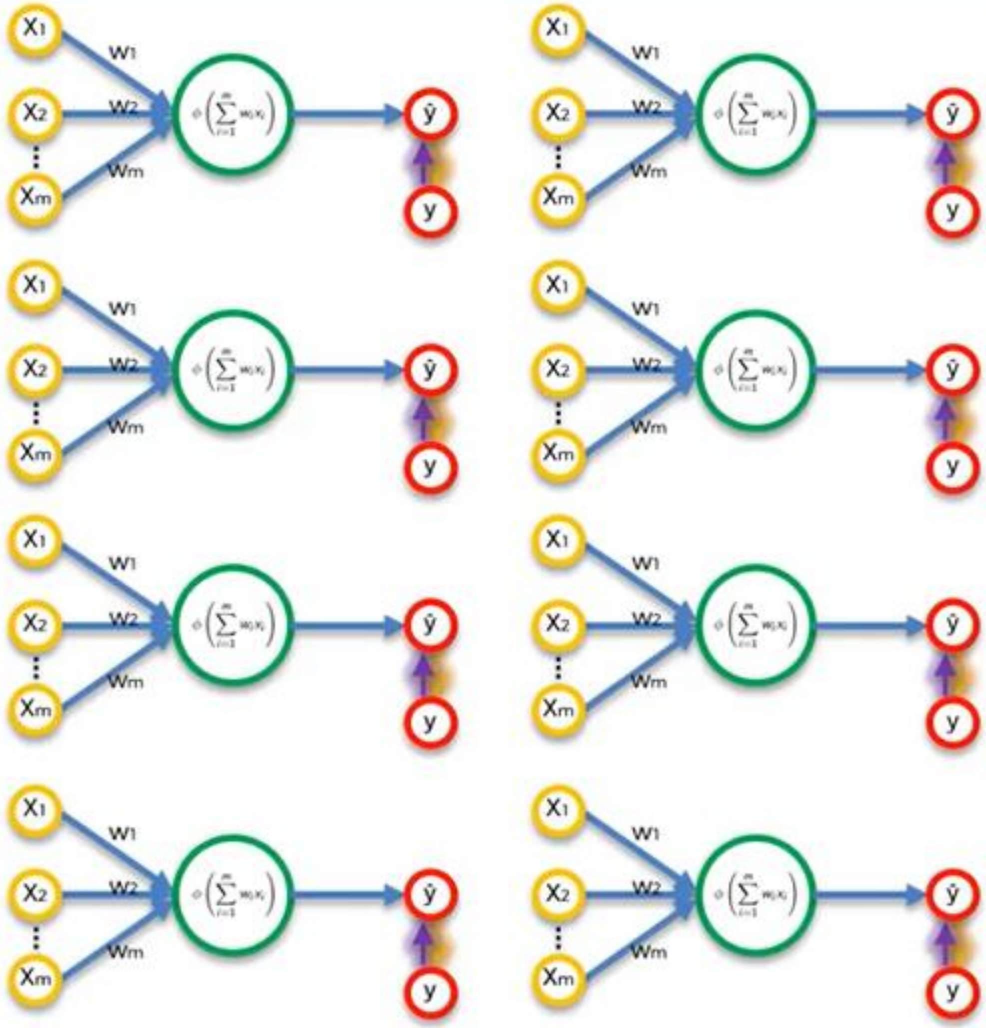
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

How do Neural Networks learn?



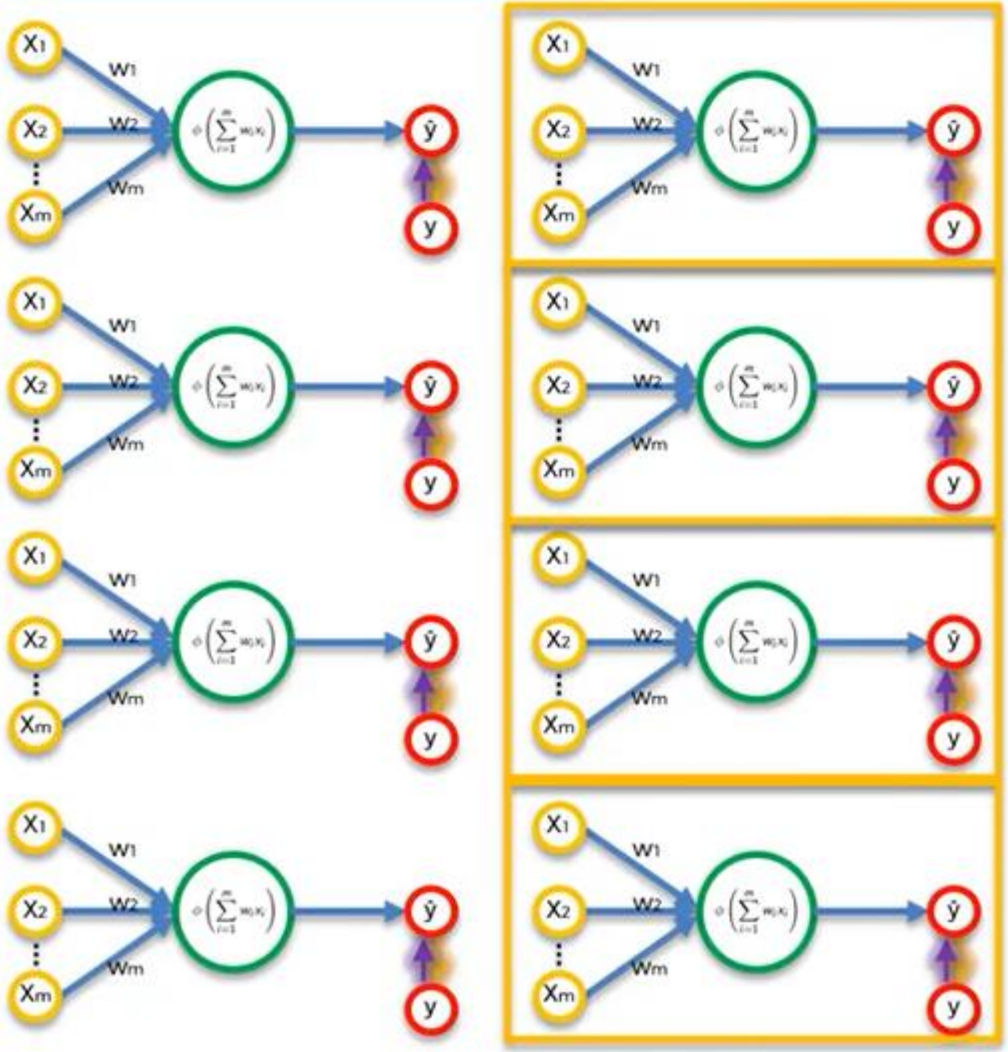
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

How do Neural Networks learn?

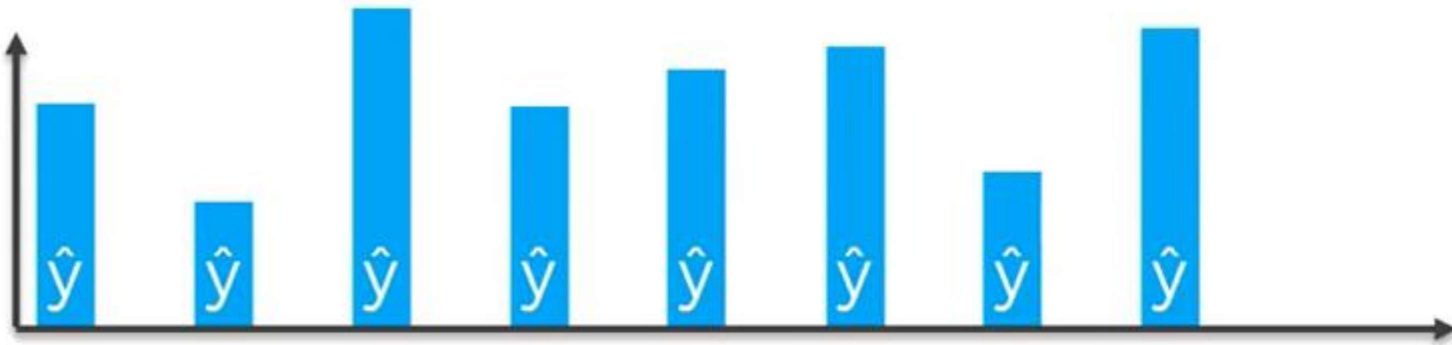


Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

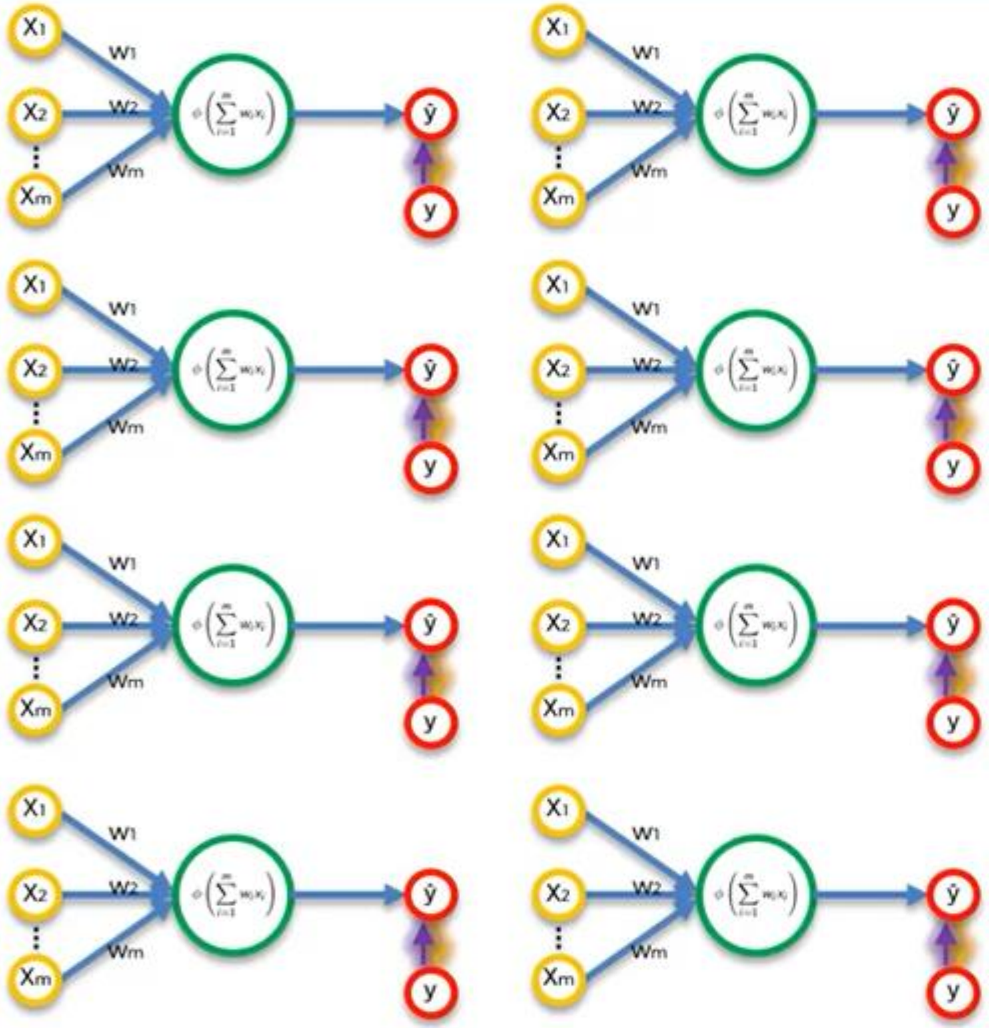
How do Neural Networks learn?



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%

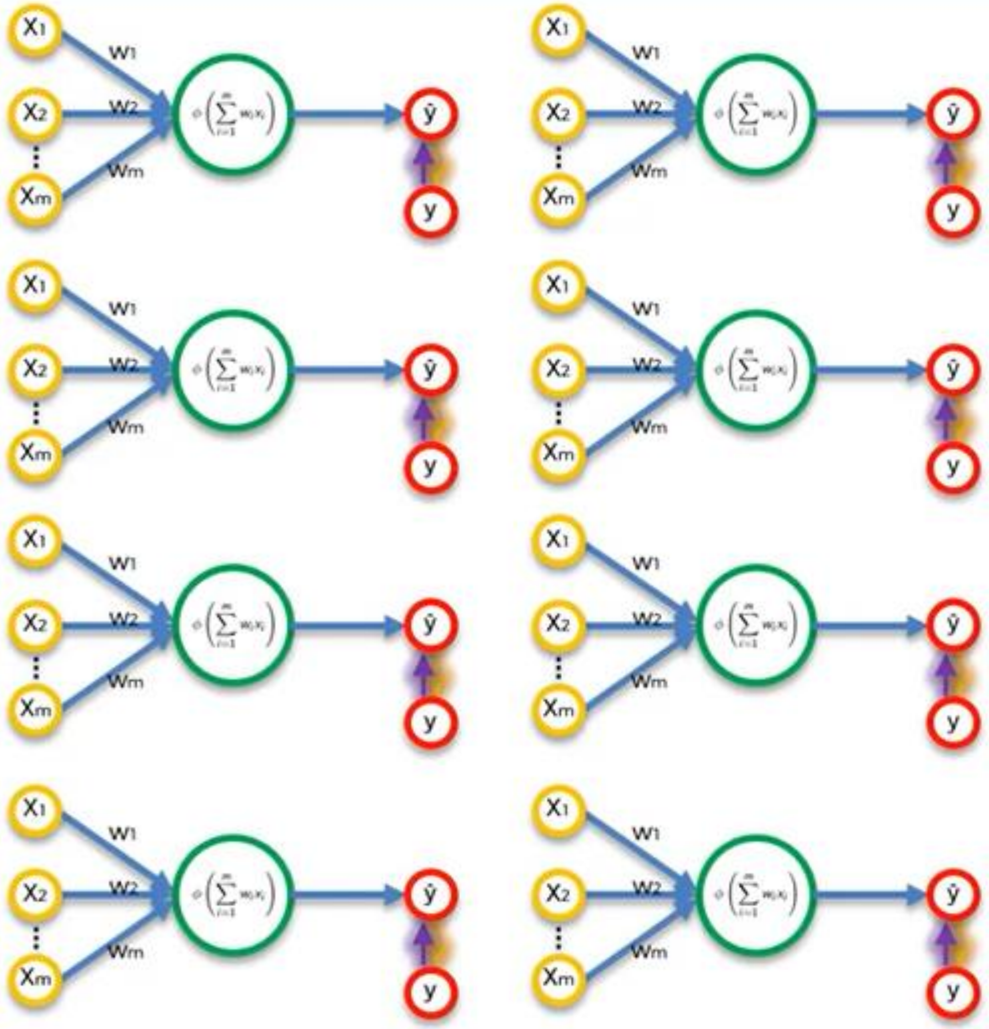


How do Neural Networks learn?



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%

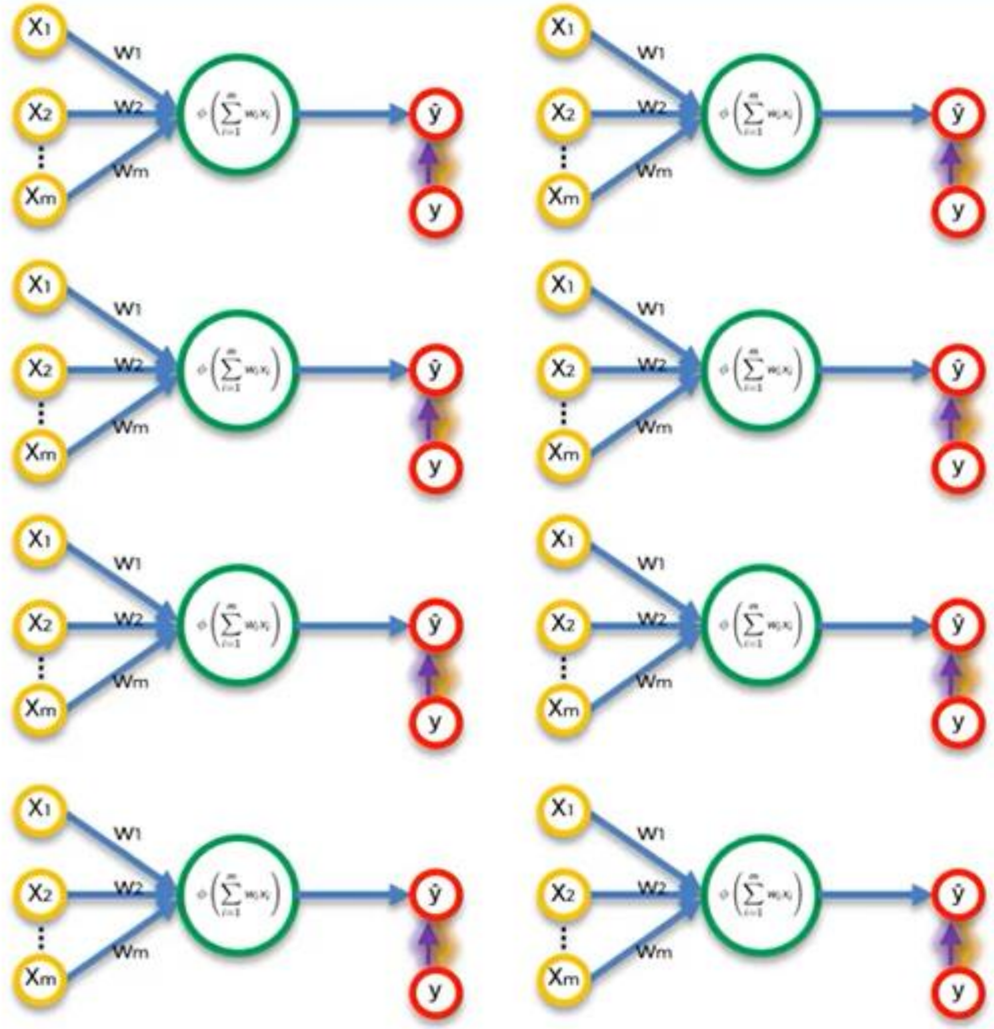
How do Neural Networks learn?



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%

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

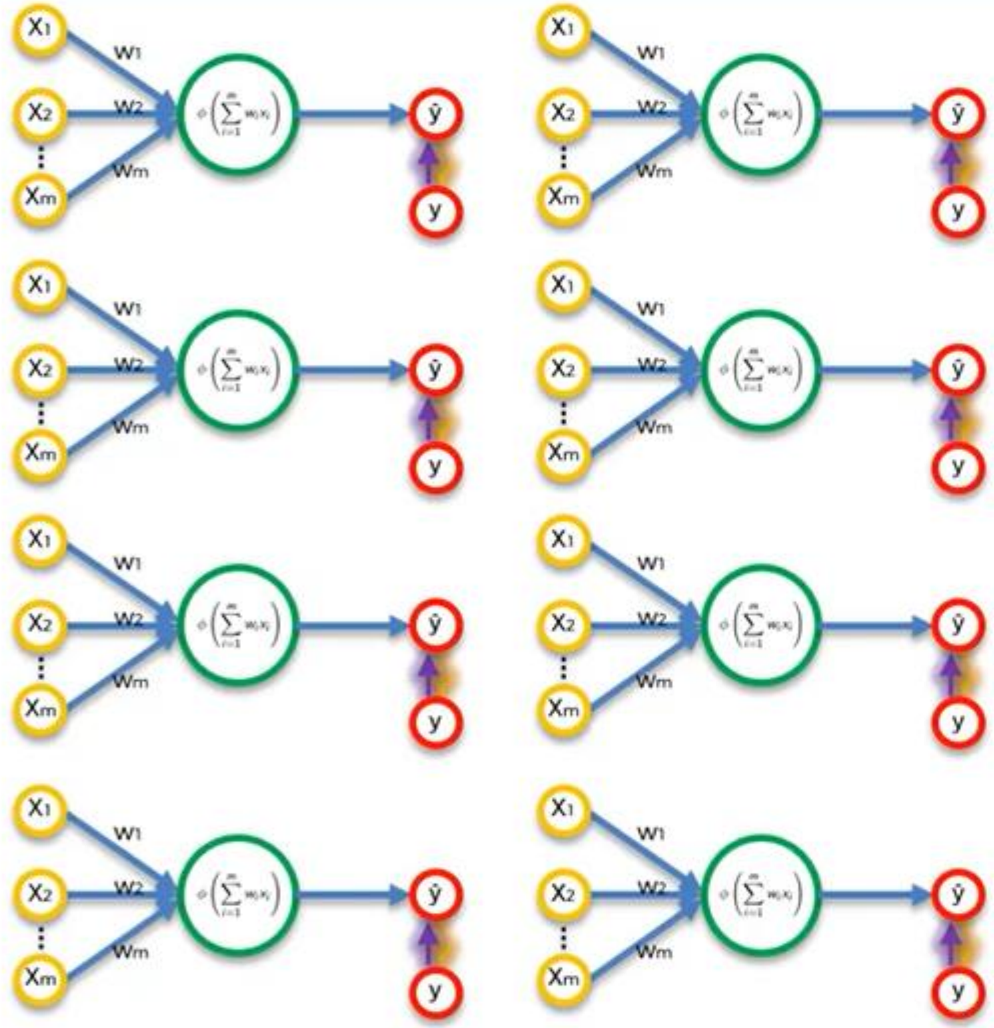
How do Neural Networks learn?



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%

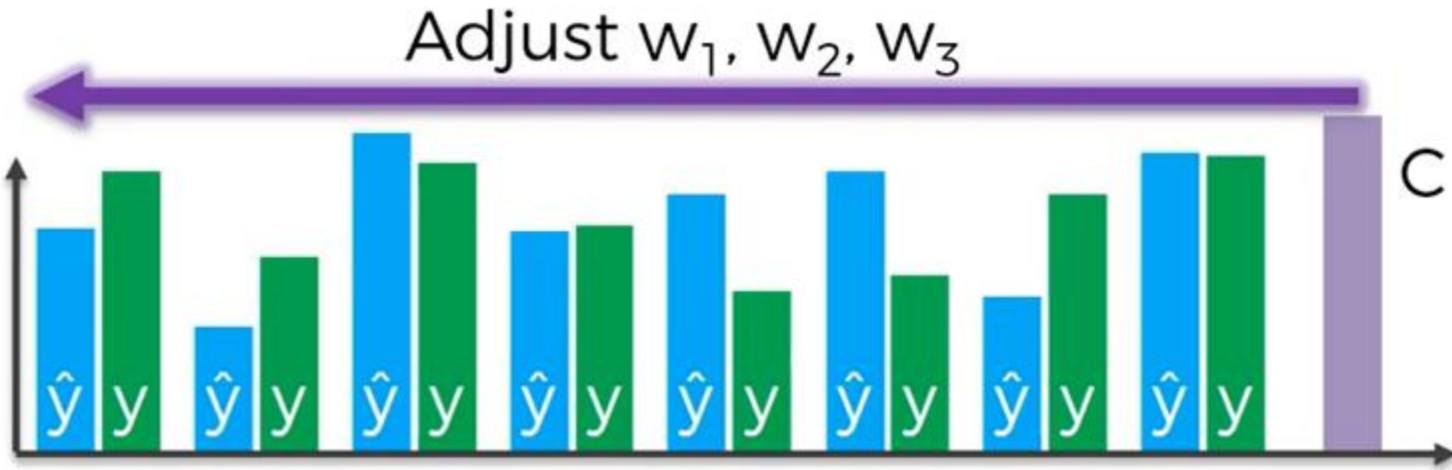
$C = \sum \frac{1}{2}(\hat{y} - y)$

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%

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

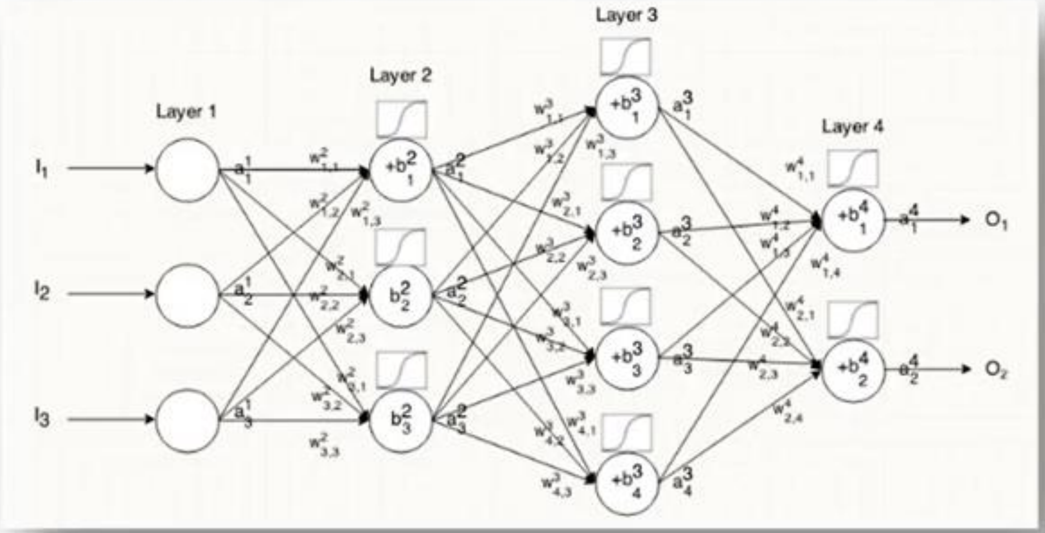


How do Neural Networks learn?

Additional Reading:

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

CrossValidated (2015)

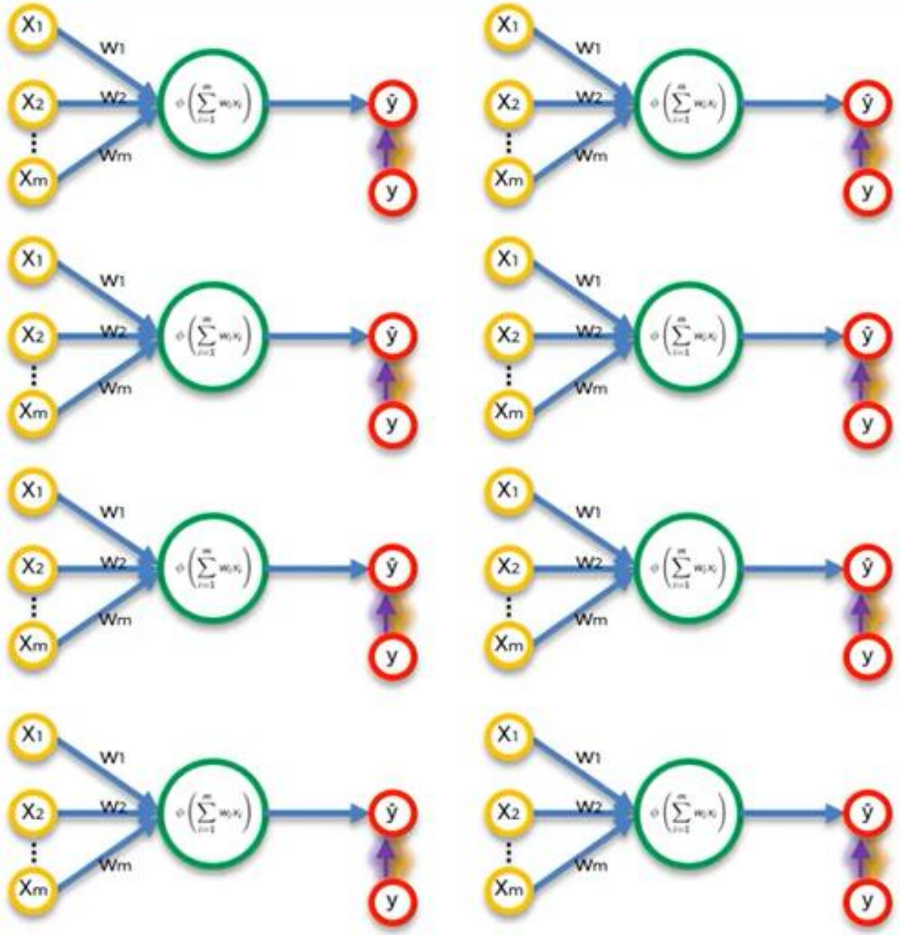


Link:

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

Gradient Descent

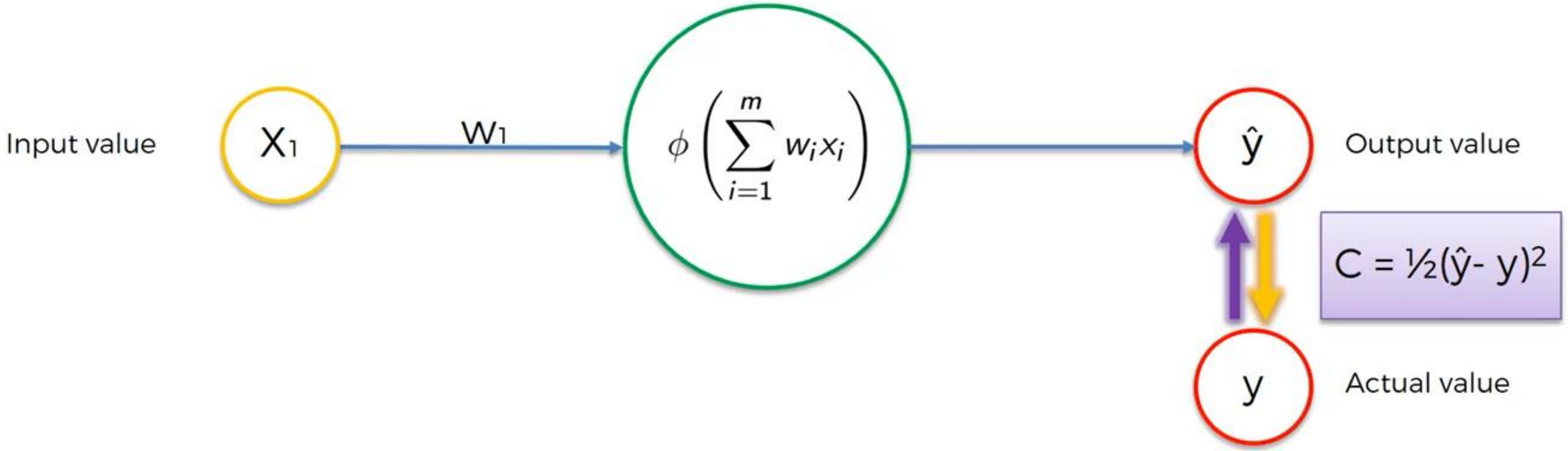
Gradient Descent



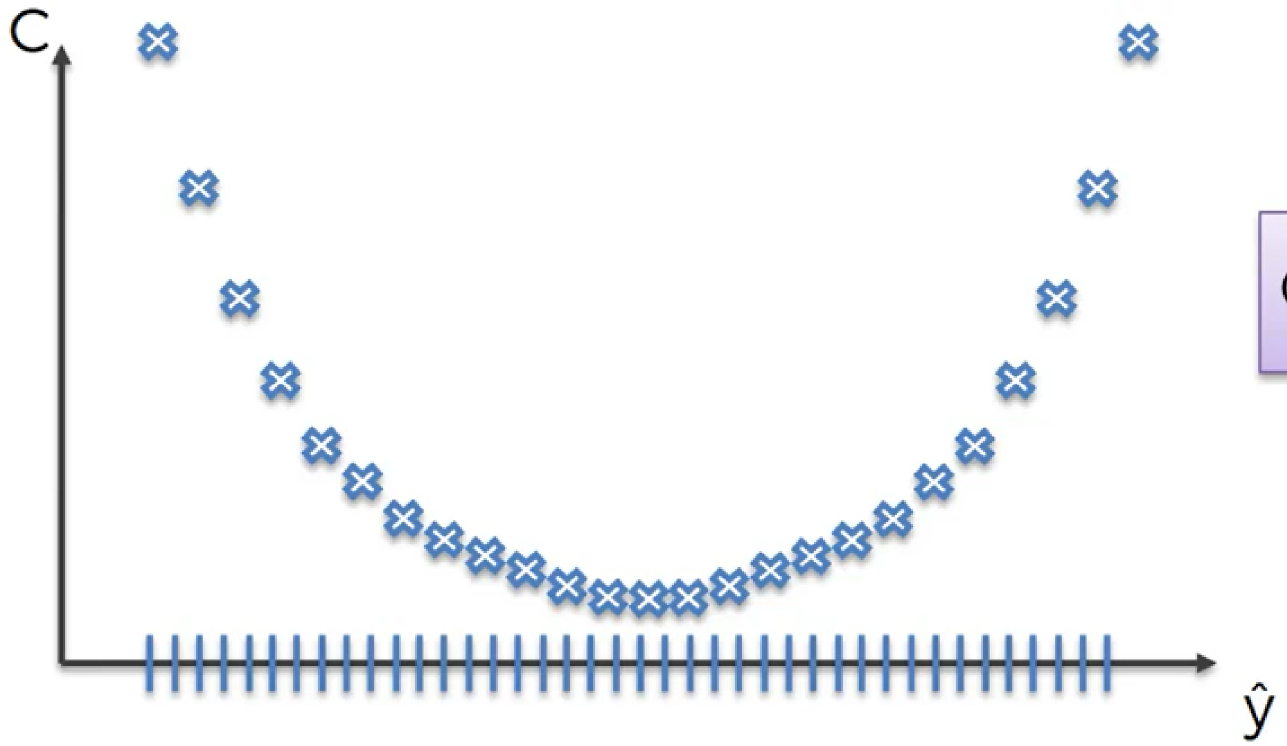
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

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

Gradient Descent

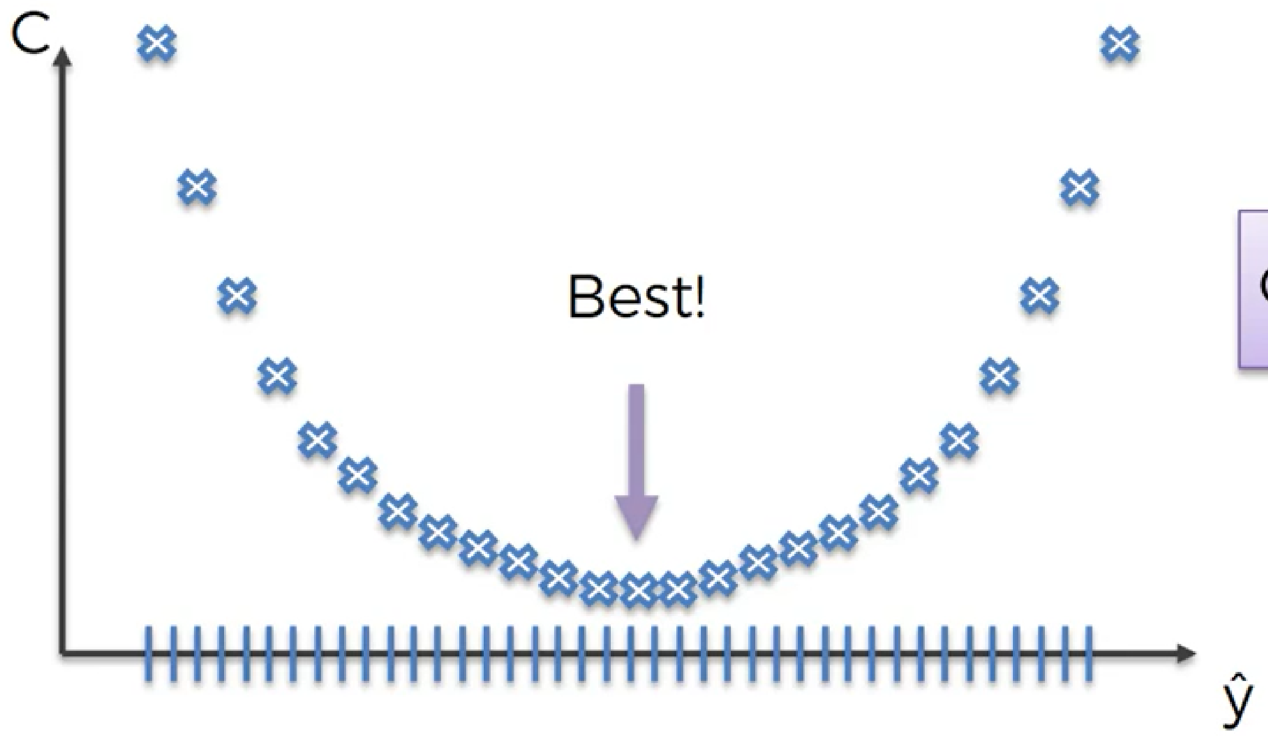


Gradient Descent



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

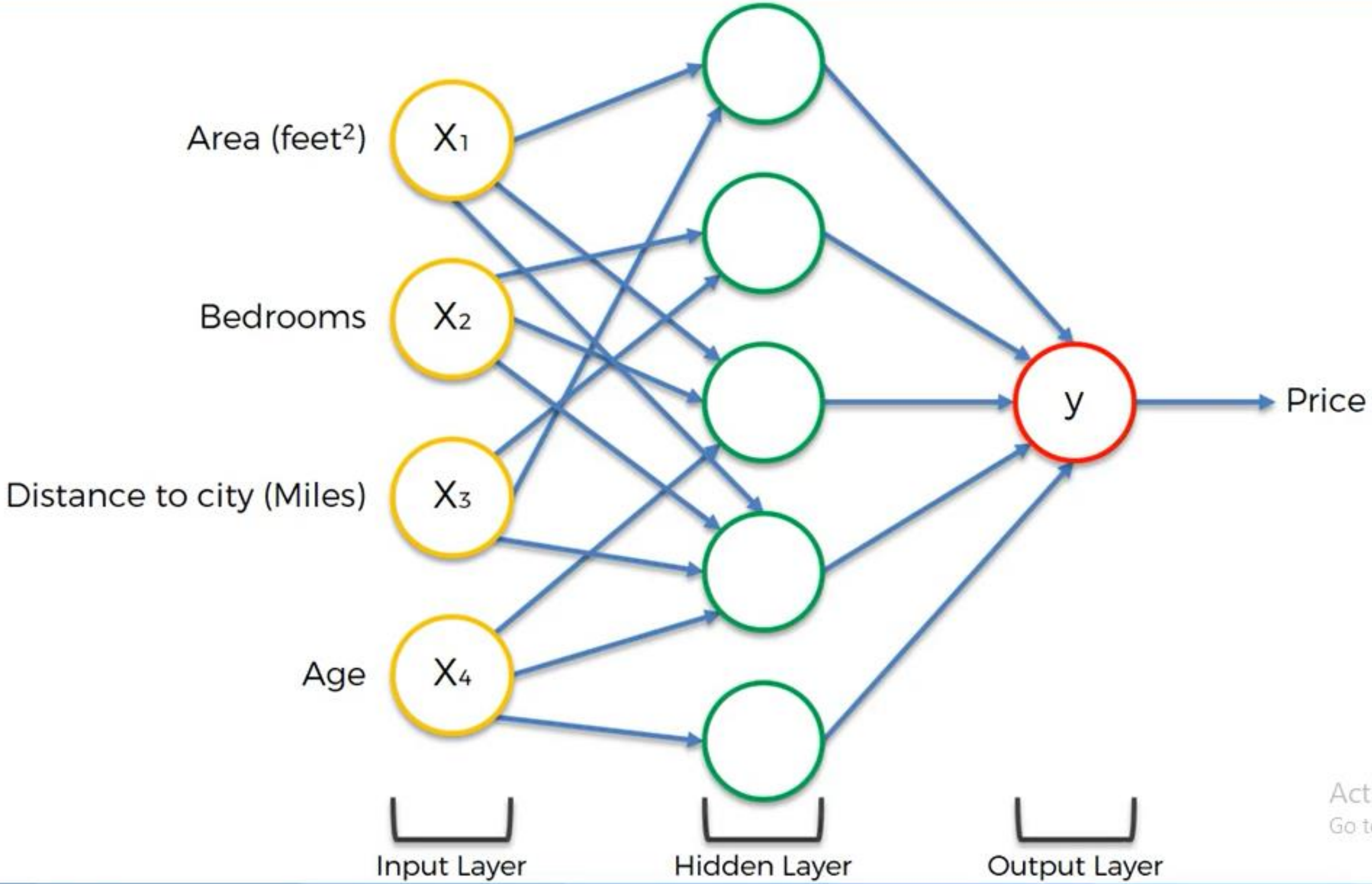
Gradient Descent



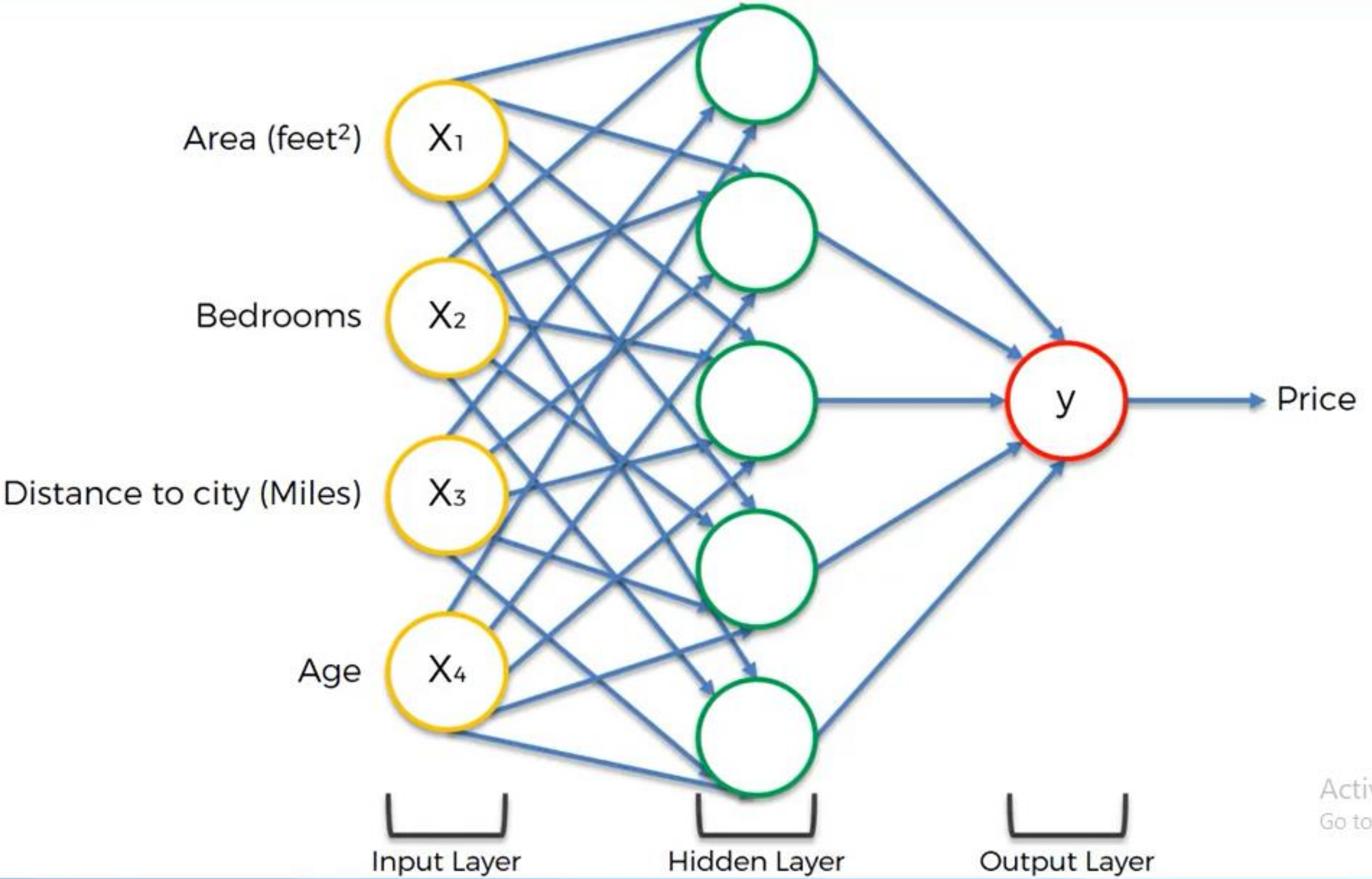
$C = \frac{1}{2}(\hat{y} - y)^2$

Curse of Dimensionality

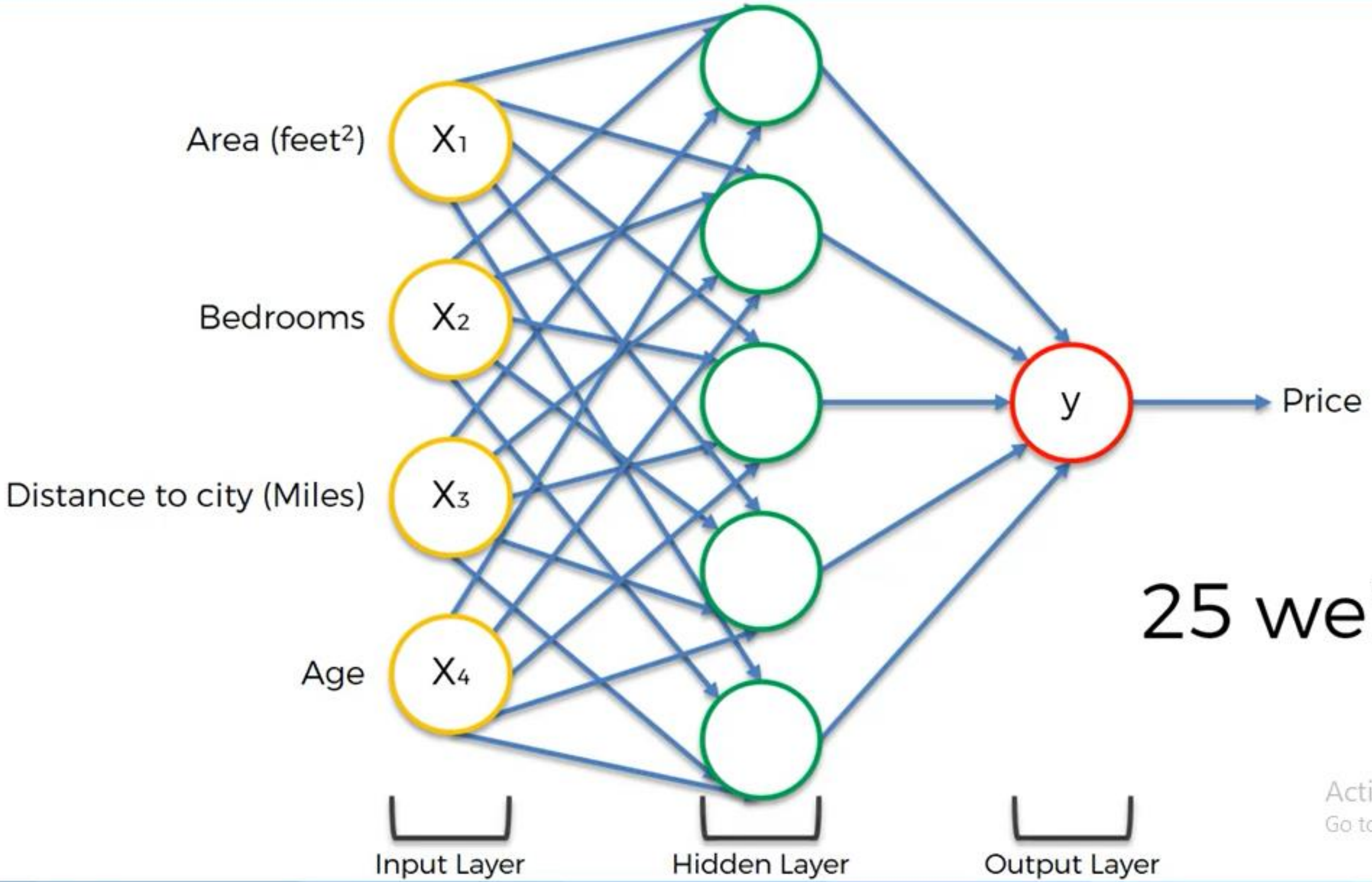
Gradient Descent



Gradient Descent



Gradient Descent



25 weights

Gradient Descent

$1{,}000 \times 1{,}000 \times \ldots \times 1{,}000 = 1{,}000^{25} = 10^{75}$ 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

Gradient Descent

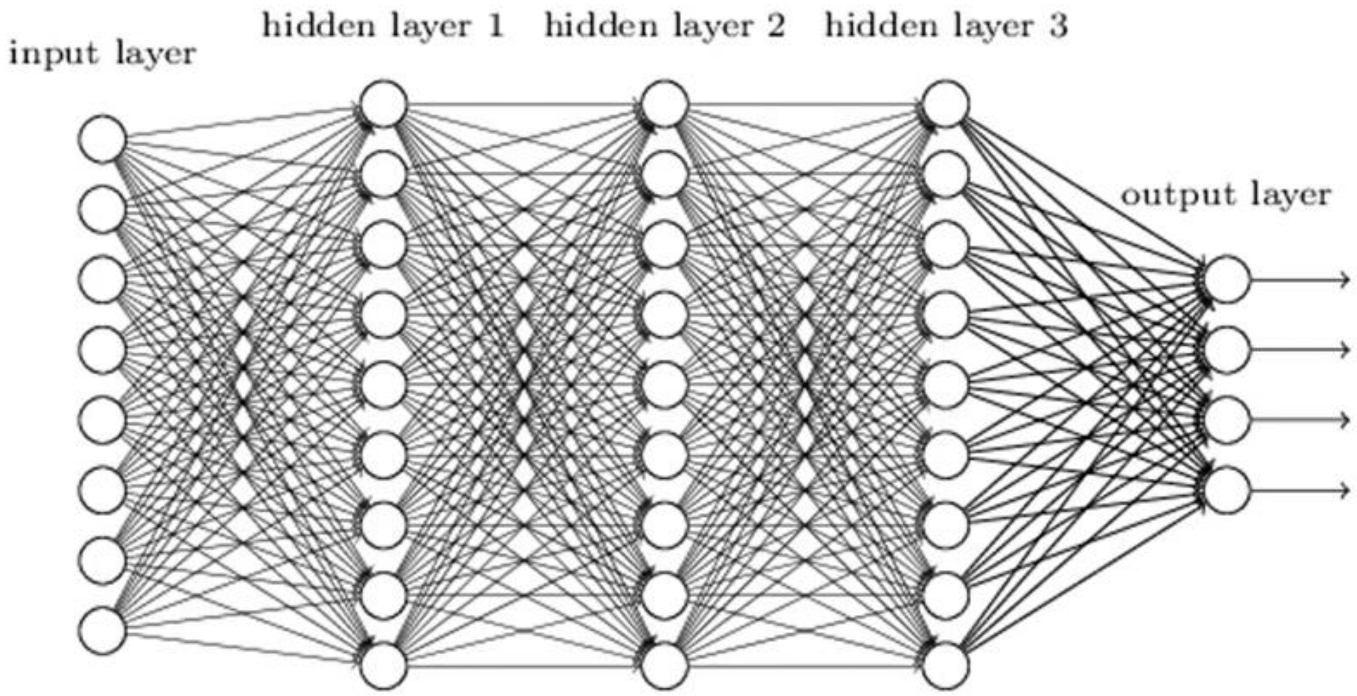
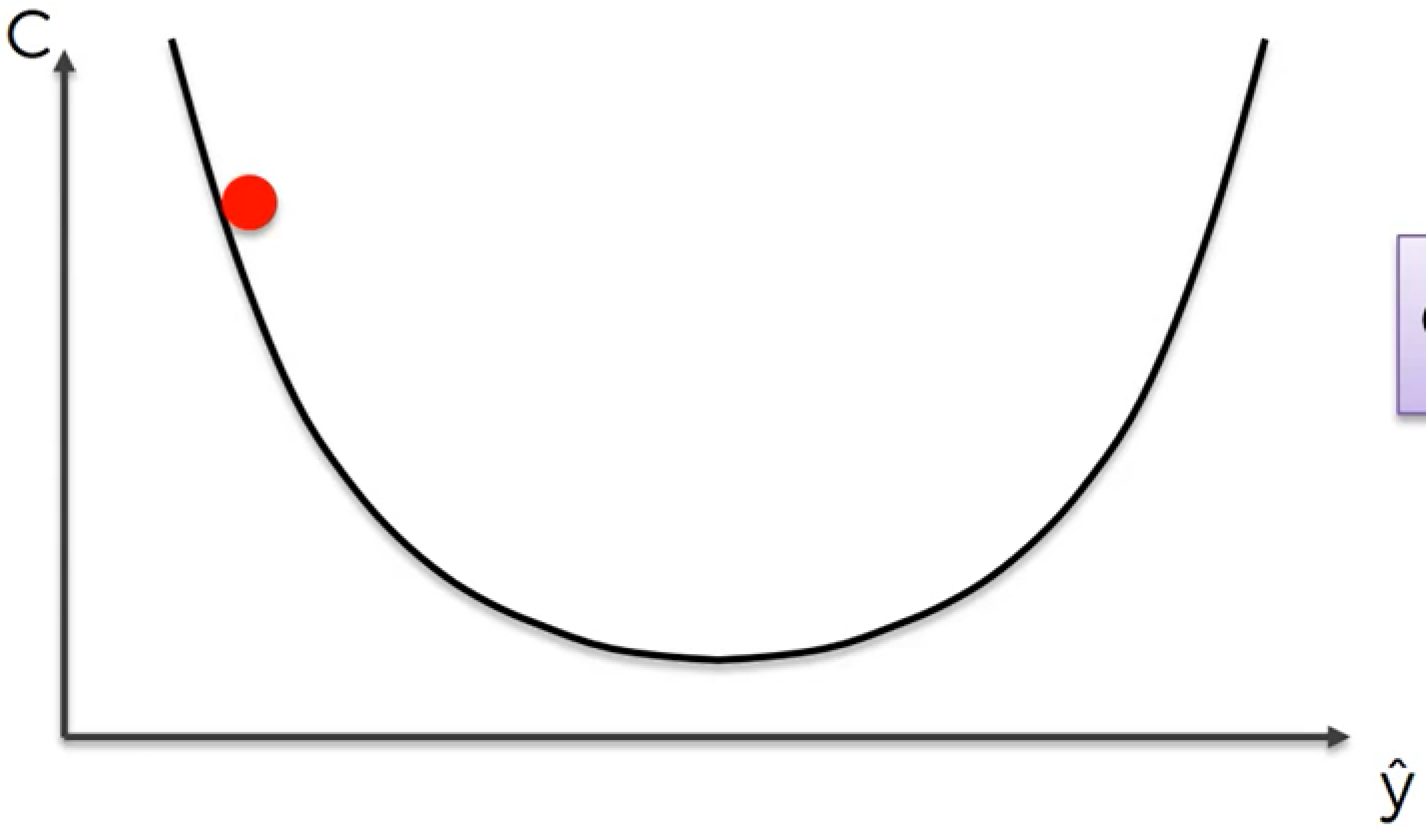


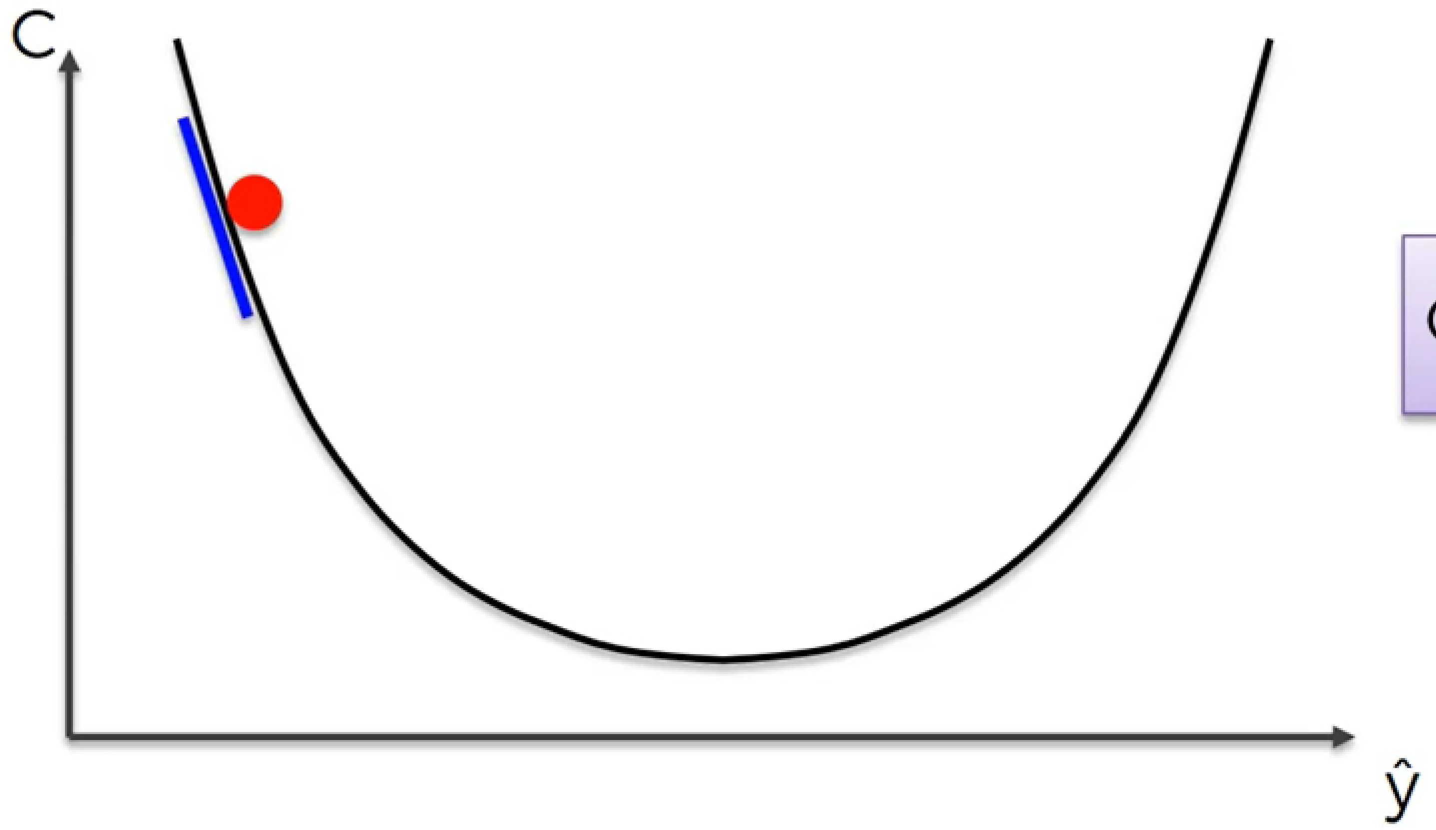
Image Source: neuralnetworksanddeeplearning.com

Gradient Descent



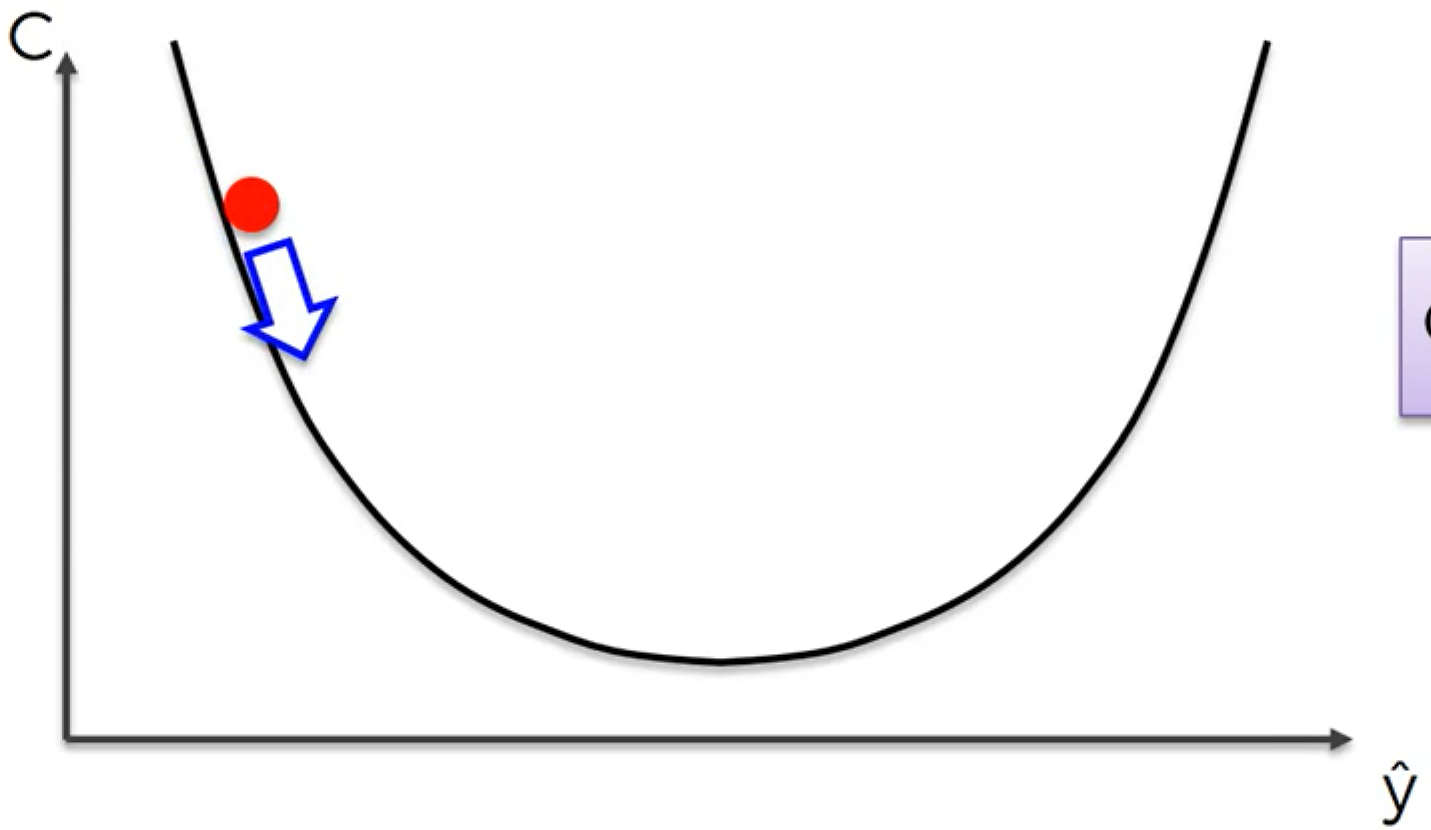
$C = \frac{1}{2}(\hat{y} - y)^2$

Gradient Descent



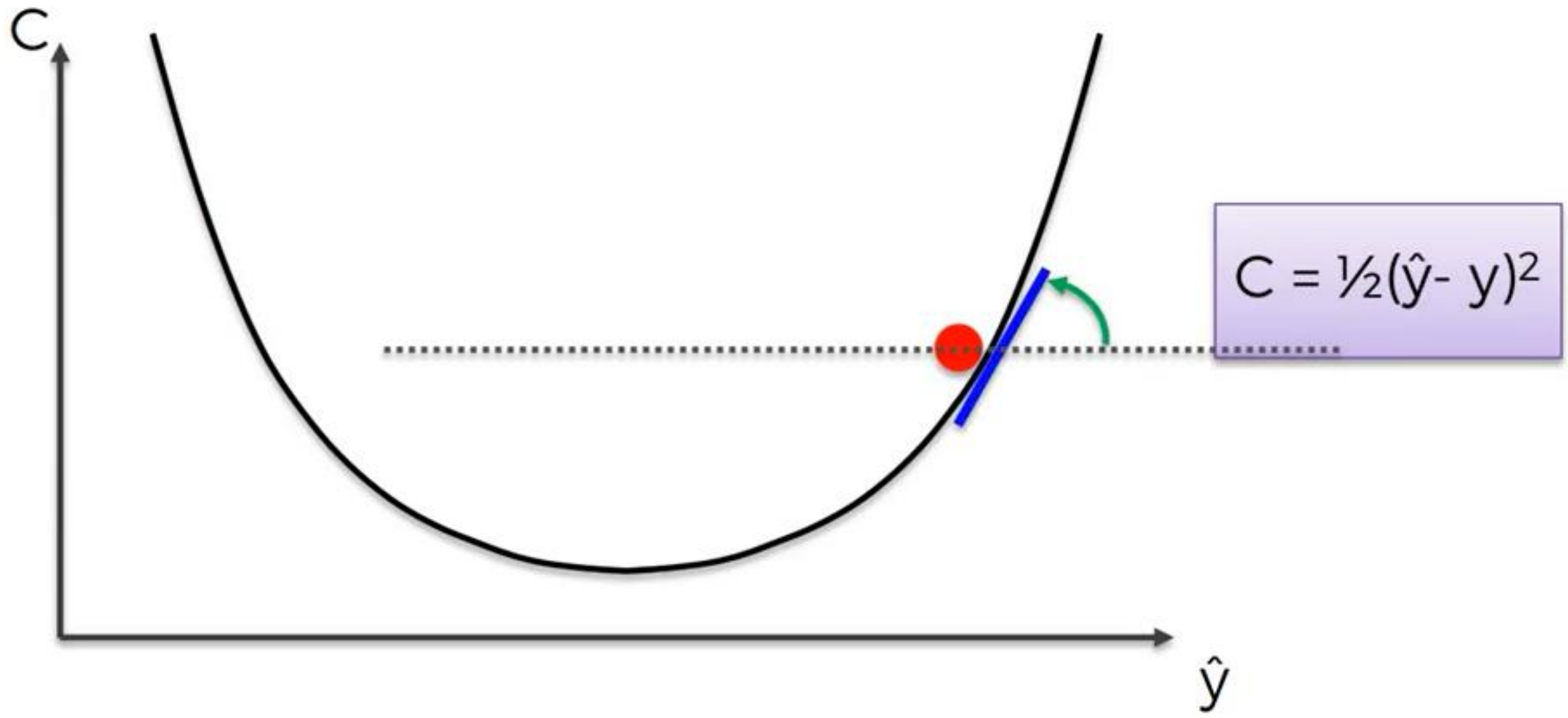
$$C = \frac{1}{2}(\hat{y} - y)^2$$

Gradient Descent

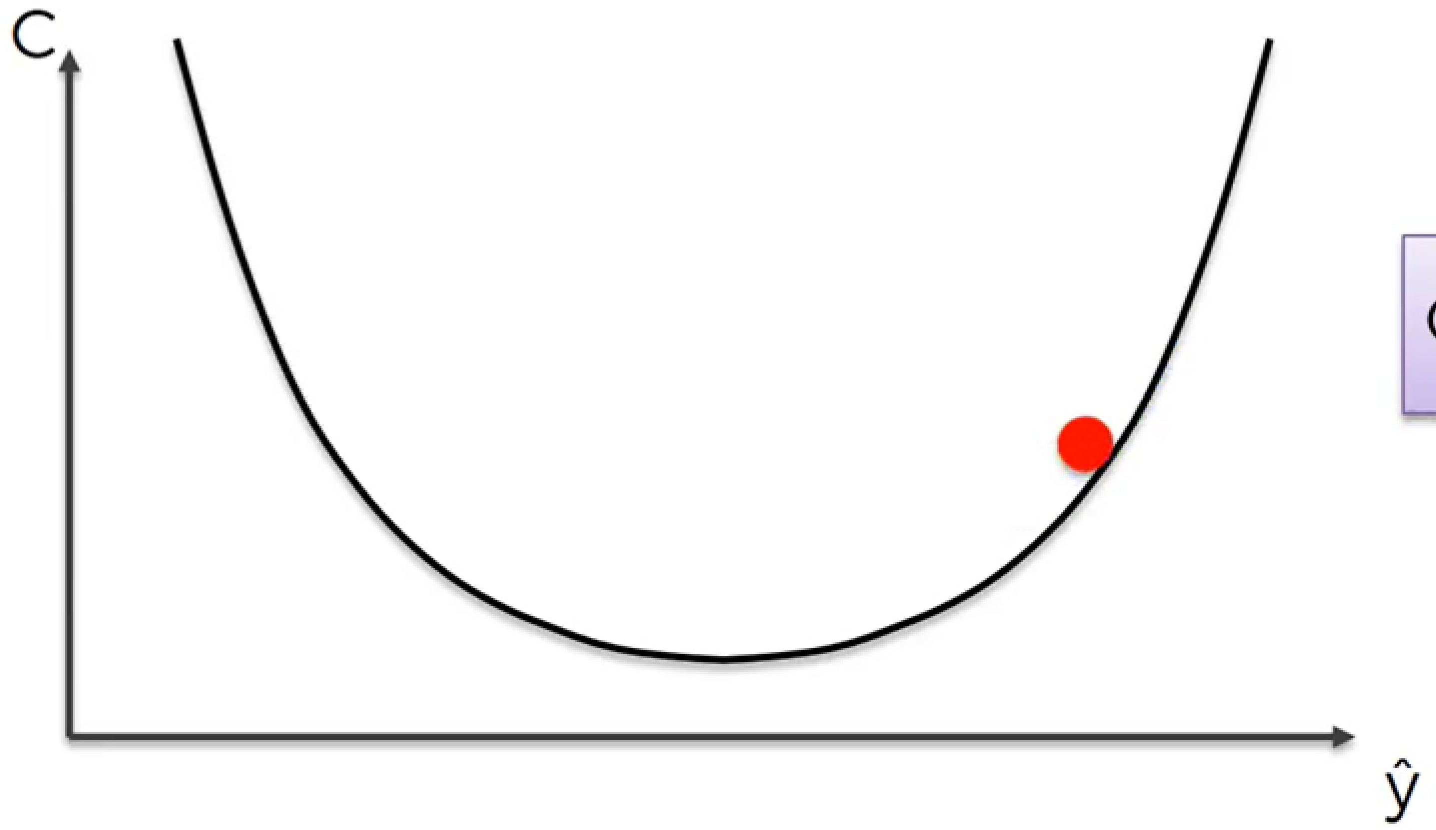


$$C = \frac{1}{2}(\hat{y} - y)^2$$

Gradient Descent

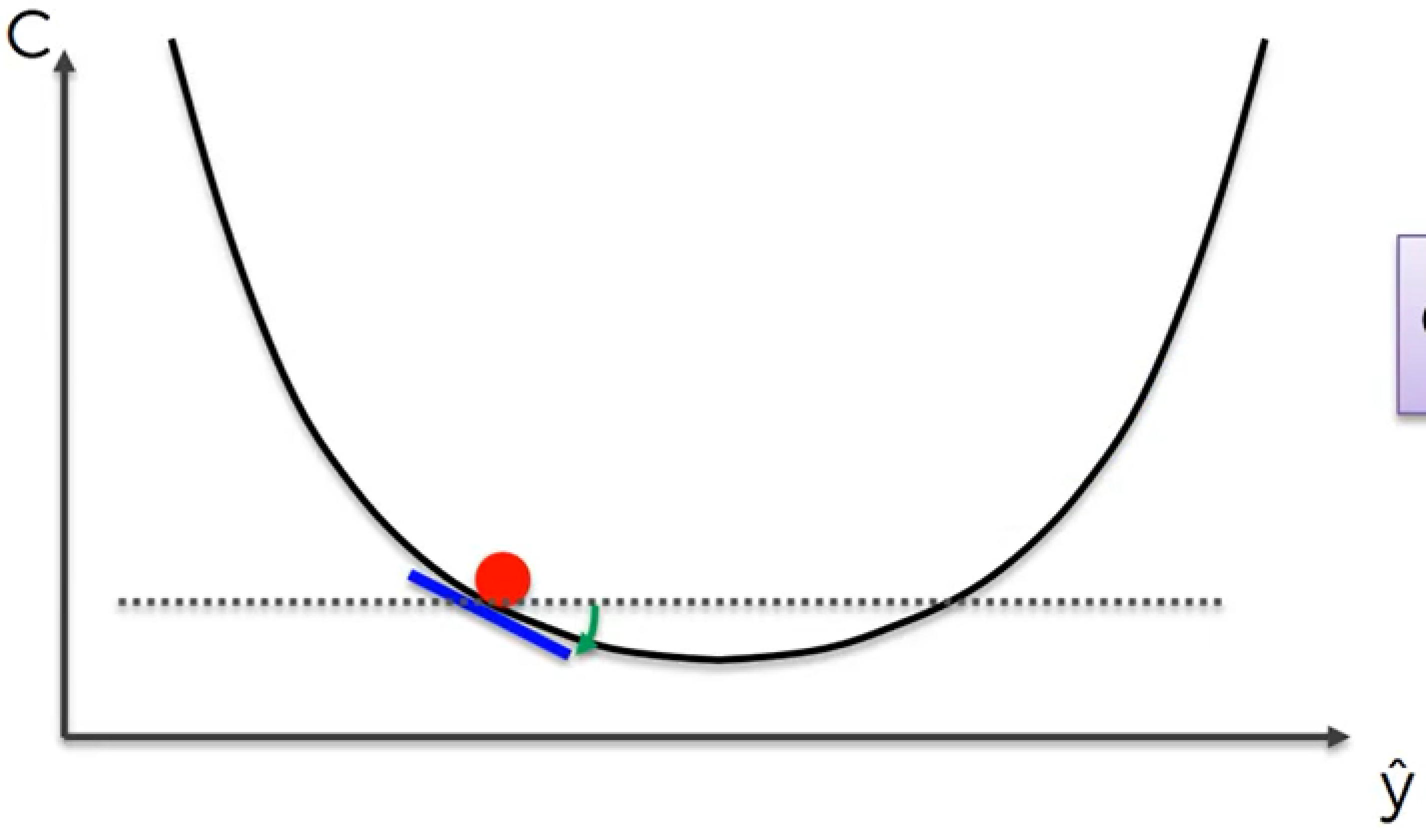


Gradient Descent



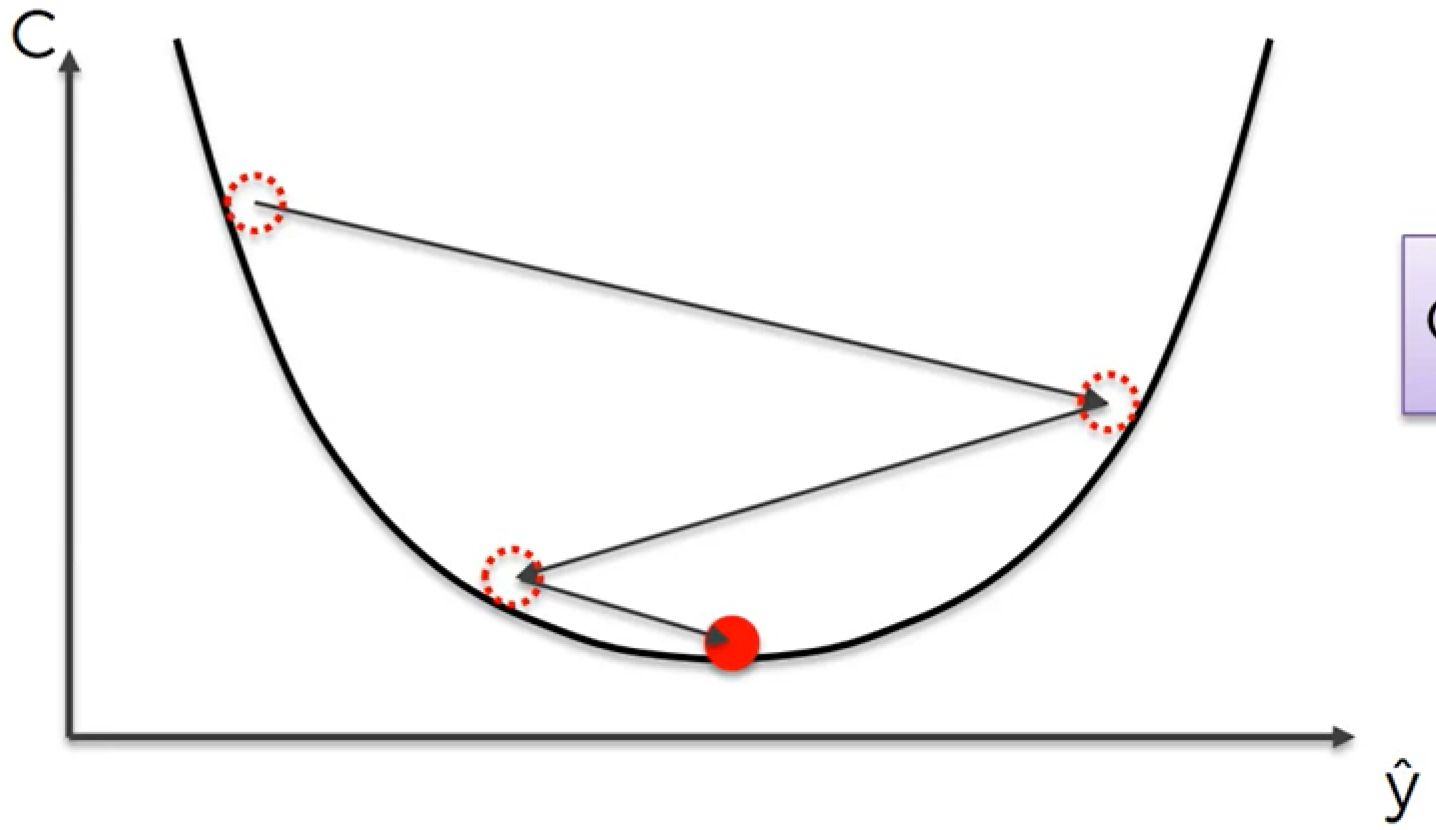
$C = \frac{1}{2}(\hat{y} - y)^2$

Gradient Descent



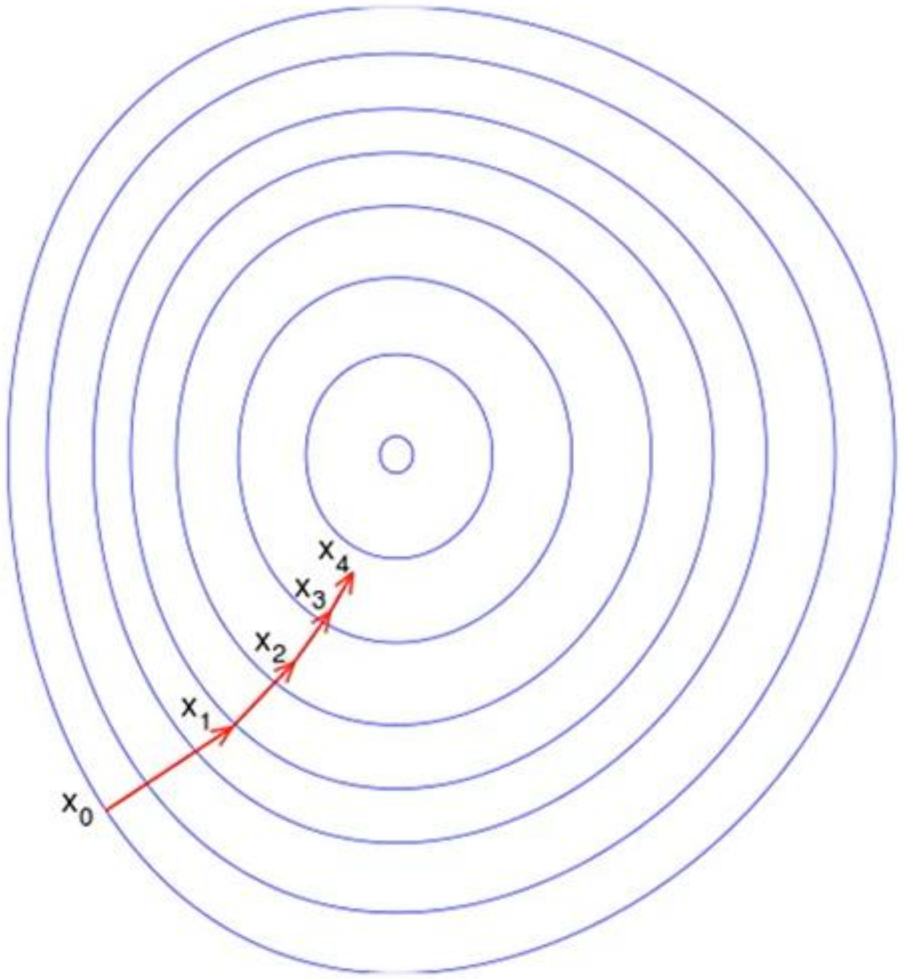
$$C = \frac{1}{2}(\hat{y} - y)^2$$

Gradient Descent

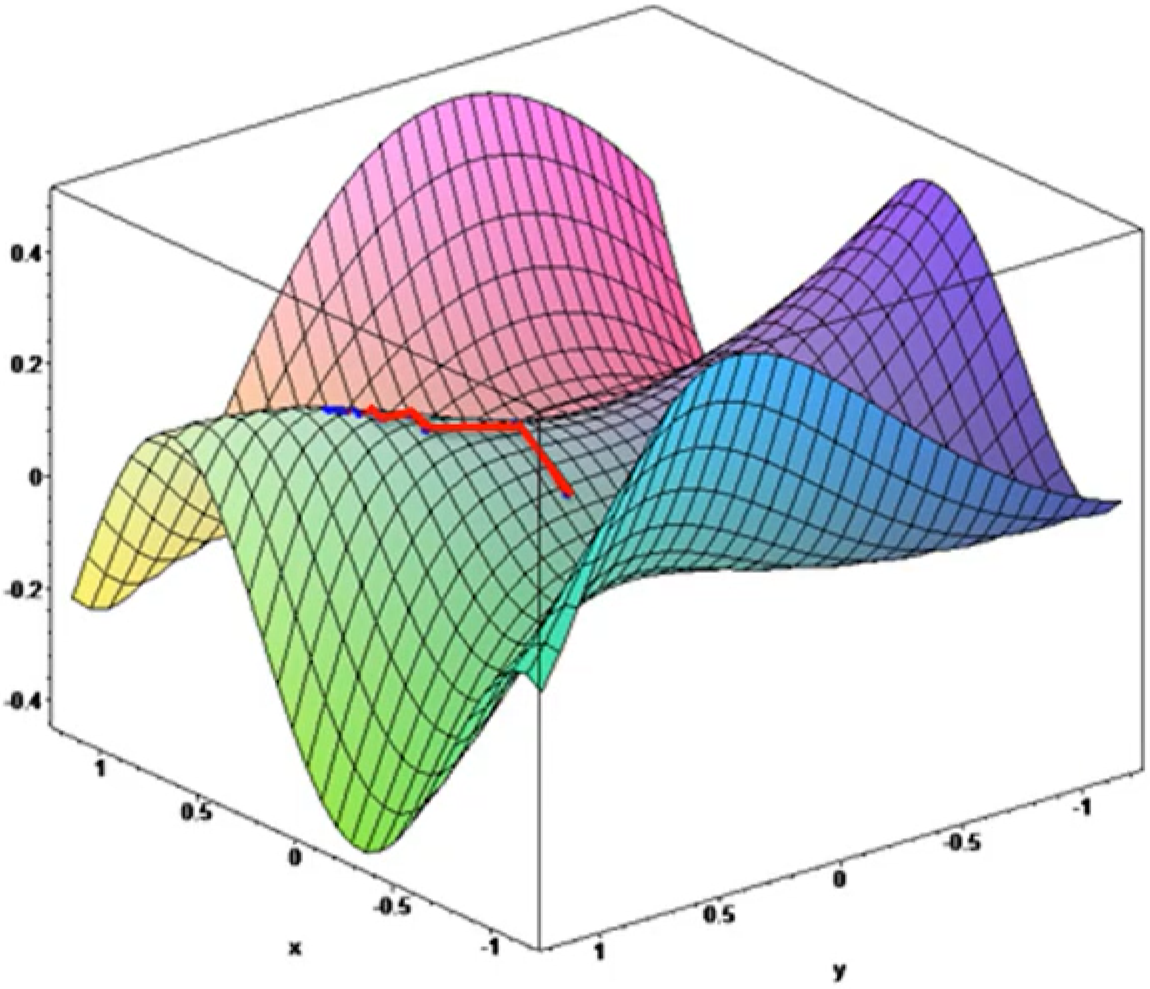


$$C = \frac{1}{2}(\hat{y} - y)^2$$

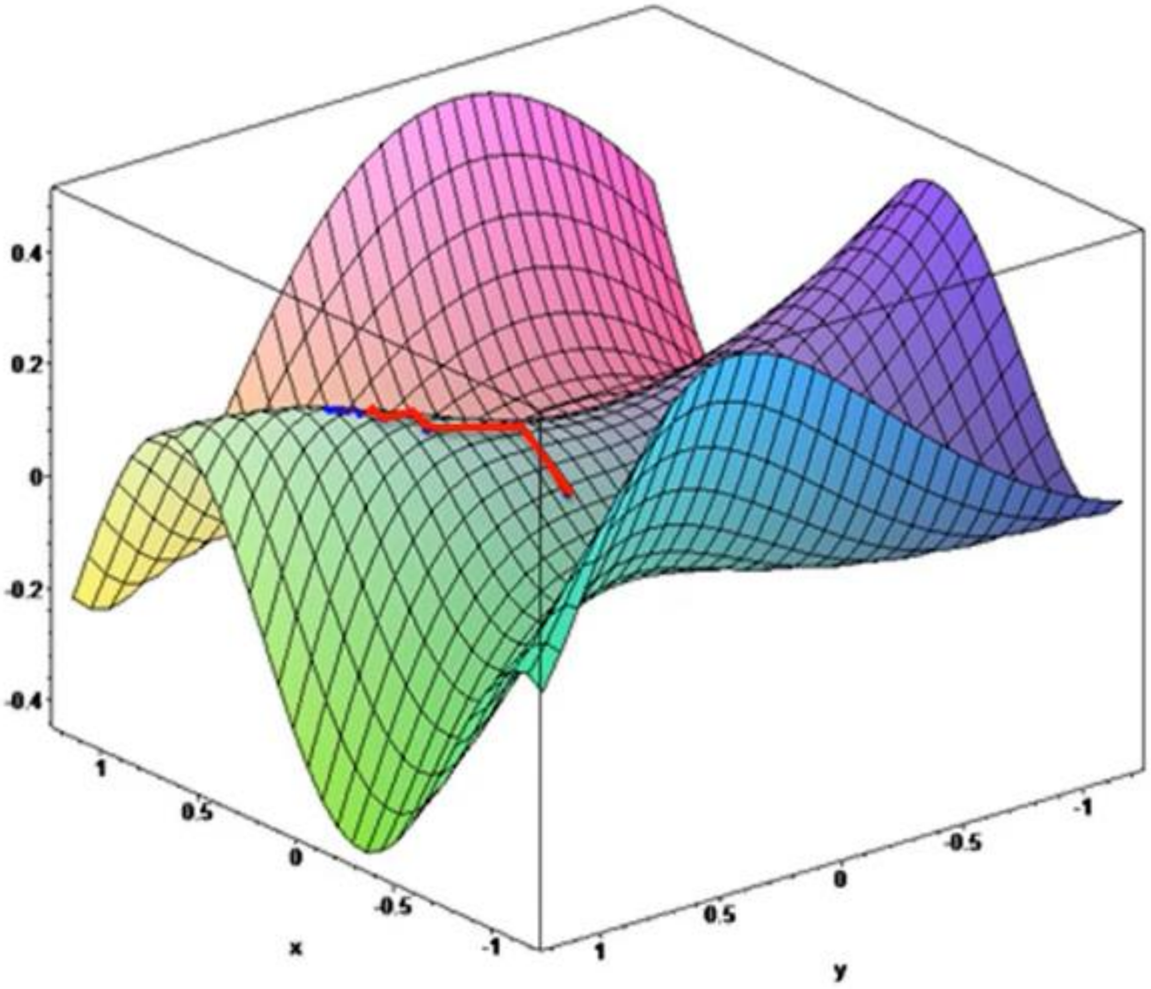
Gradient Descent

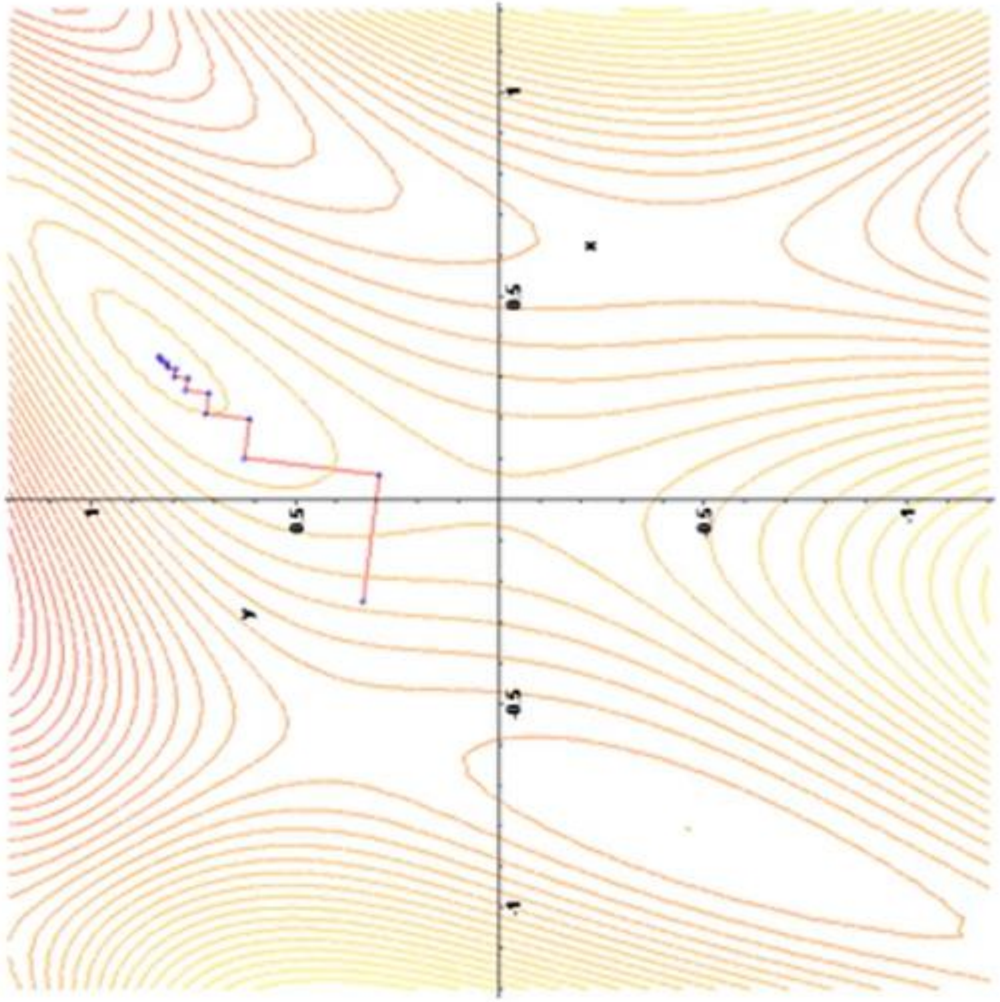


Gradient Descent



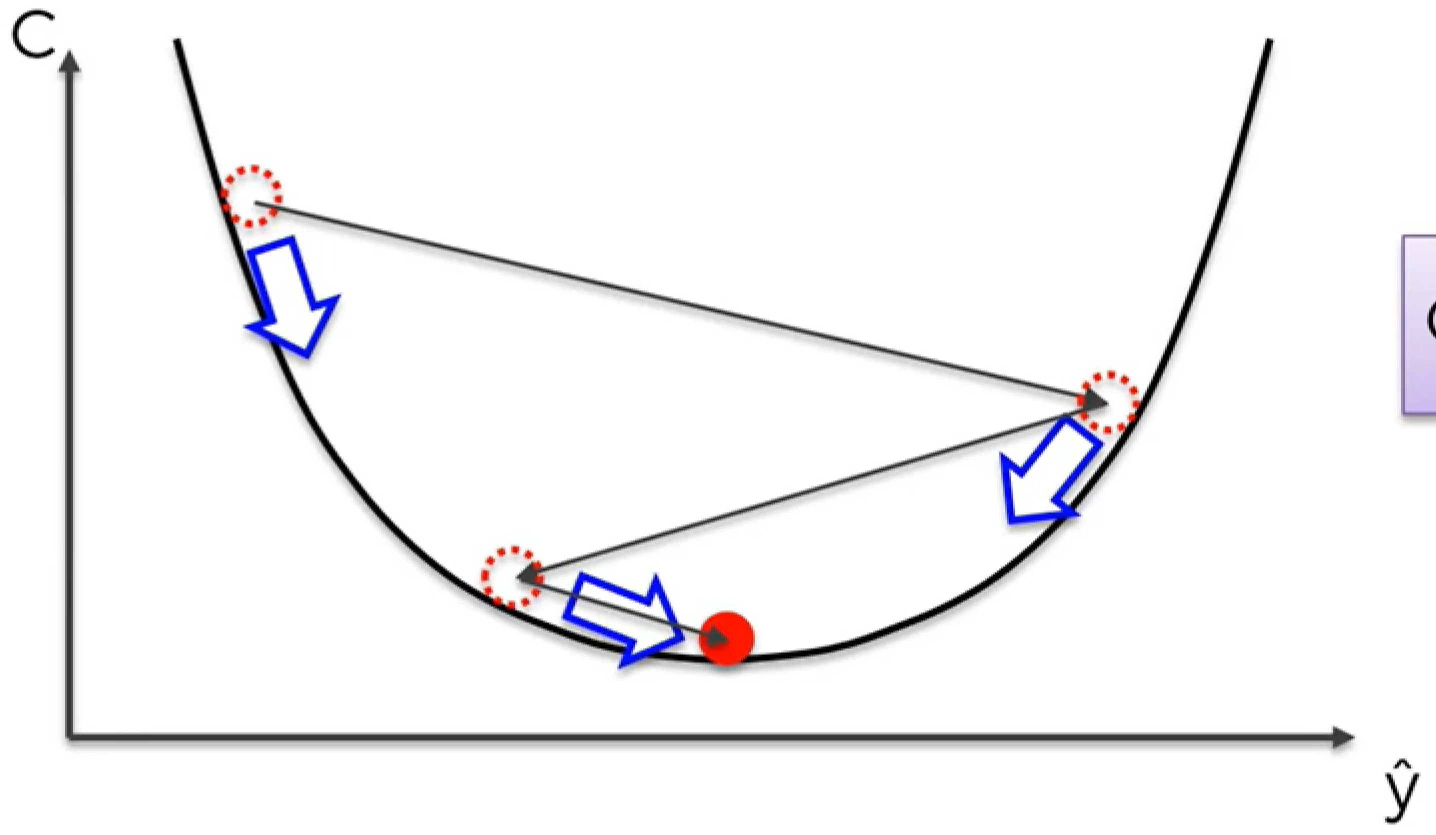
Gradient Descent





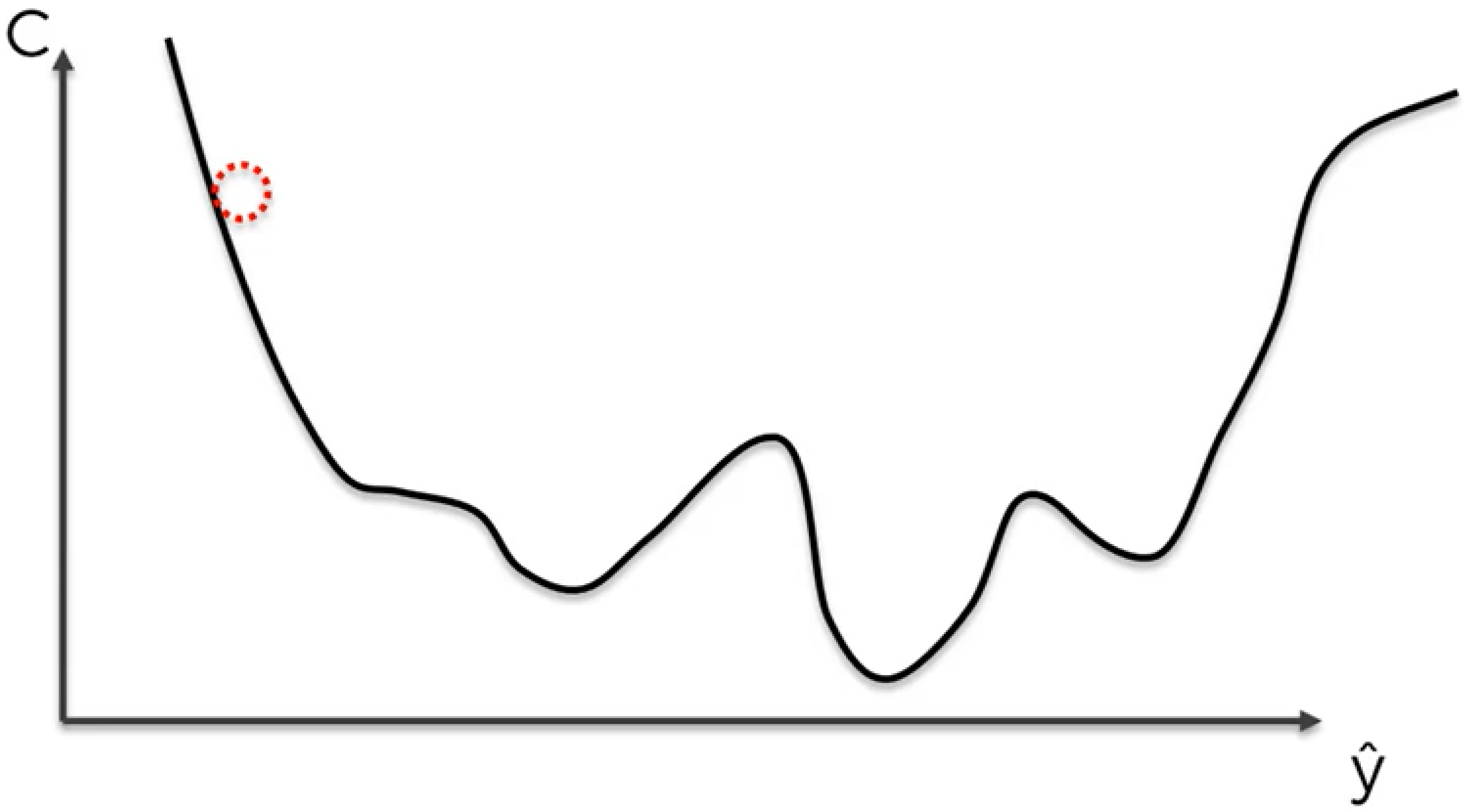
Stochastic Gradient Descent

Stochastic Gradient Descent

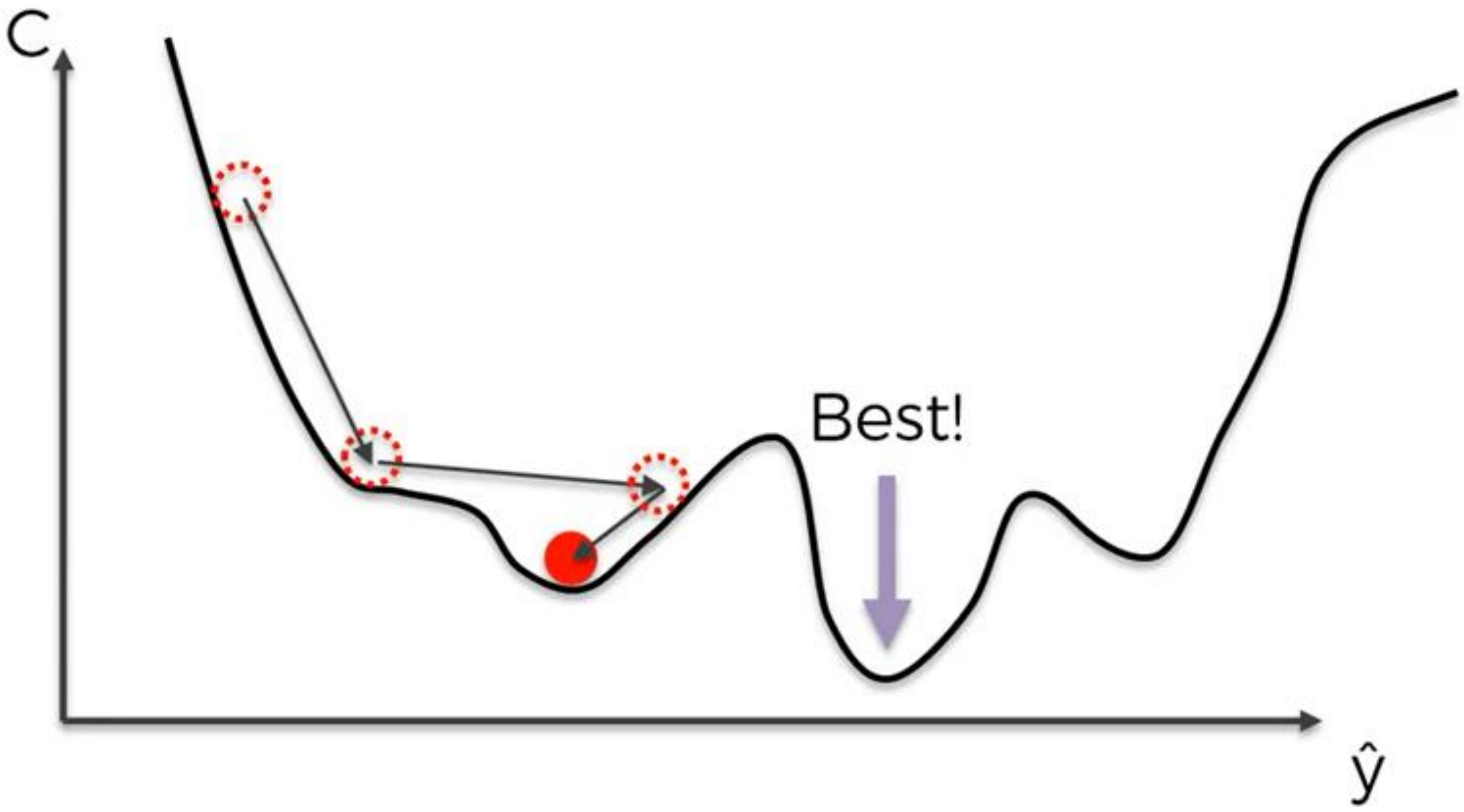


$$C = \frac{1}{2}(\hat{y} - y)^2$$

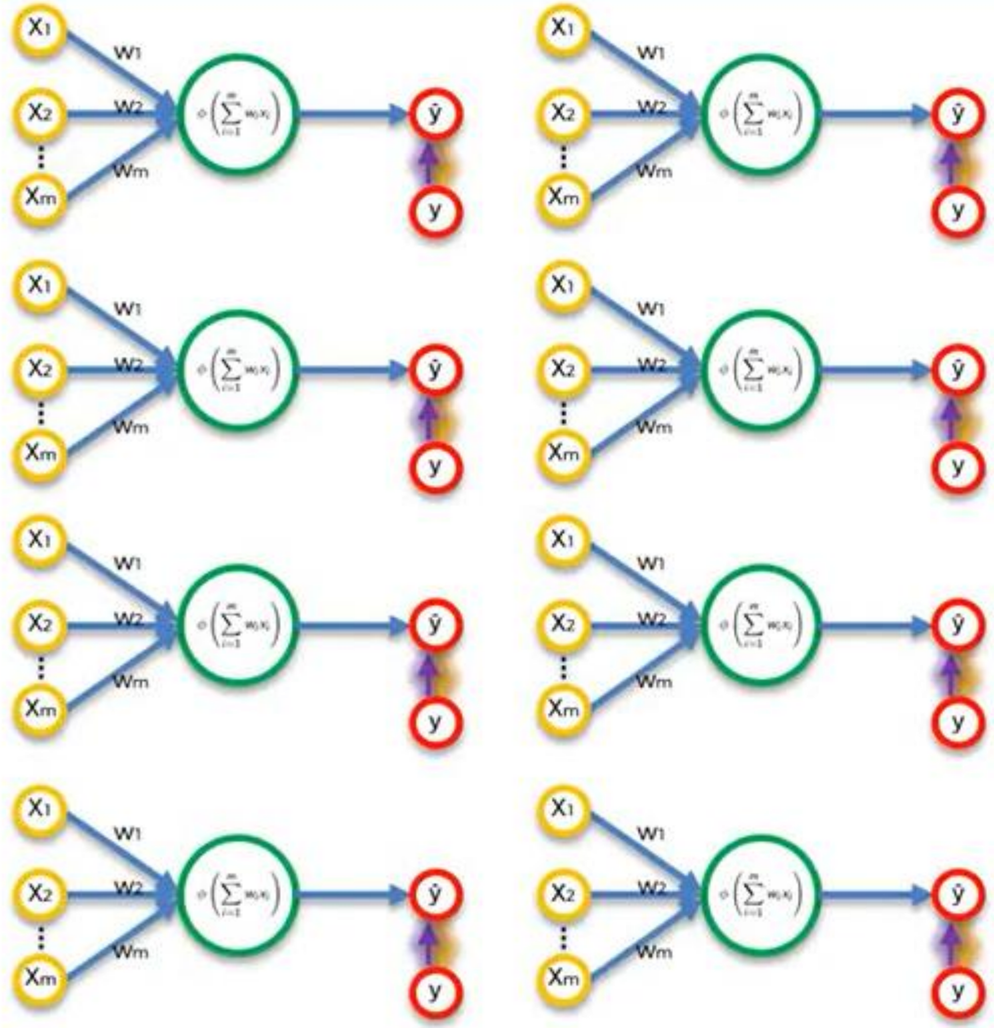
Stochastic Gradient Descent



Stochastic Gradient Descent



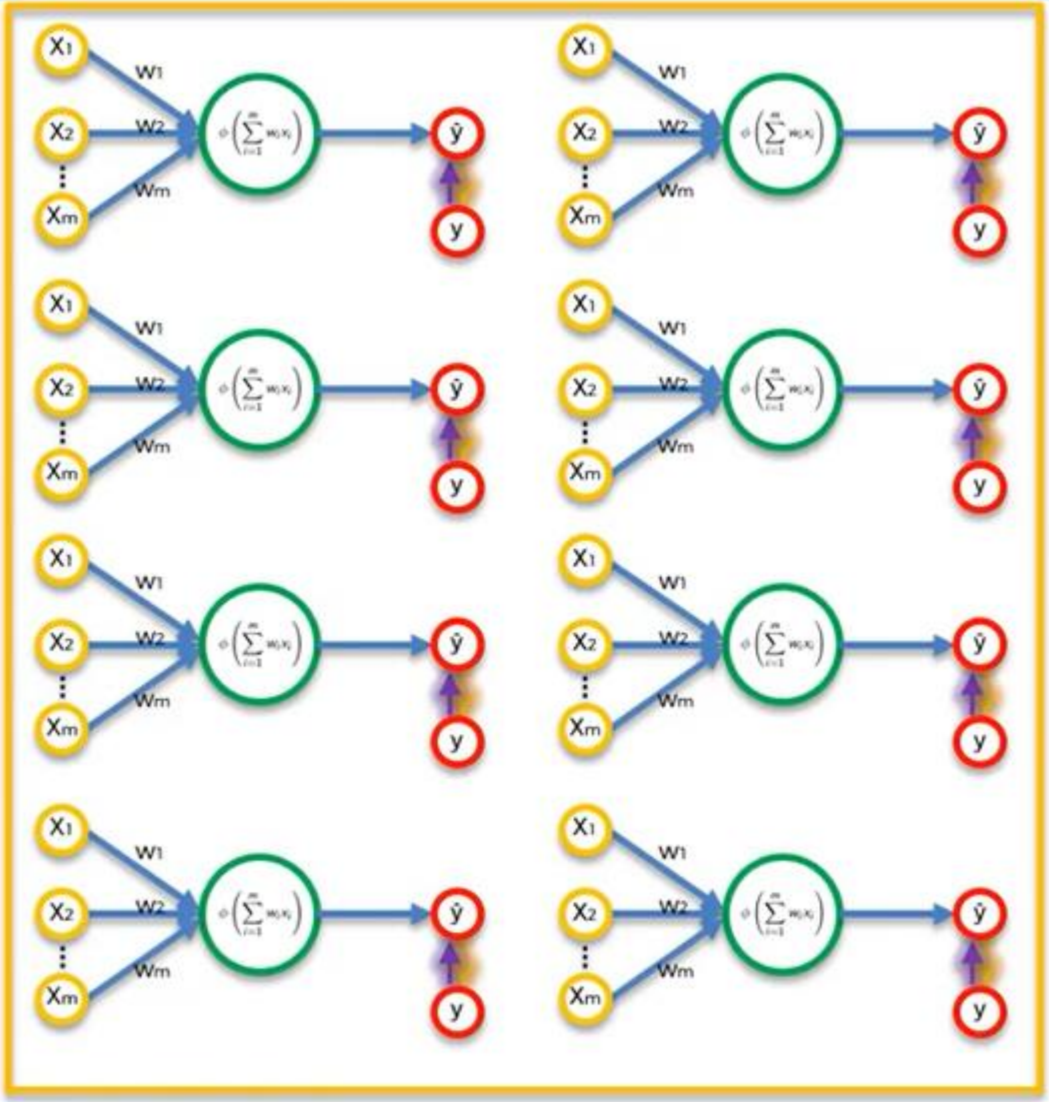
Stochastic Gradient Descent



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%

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

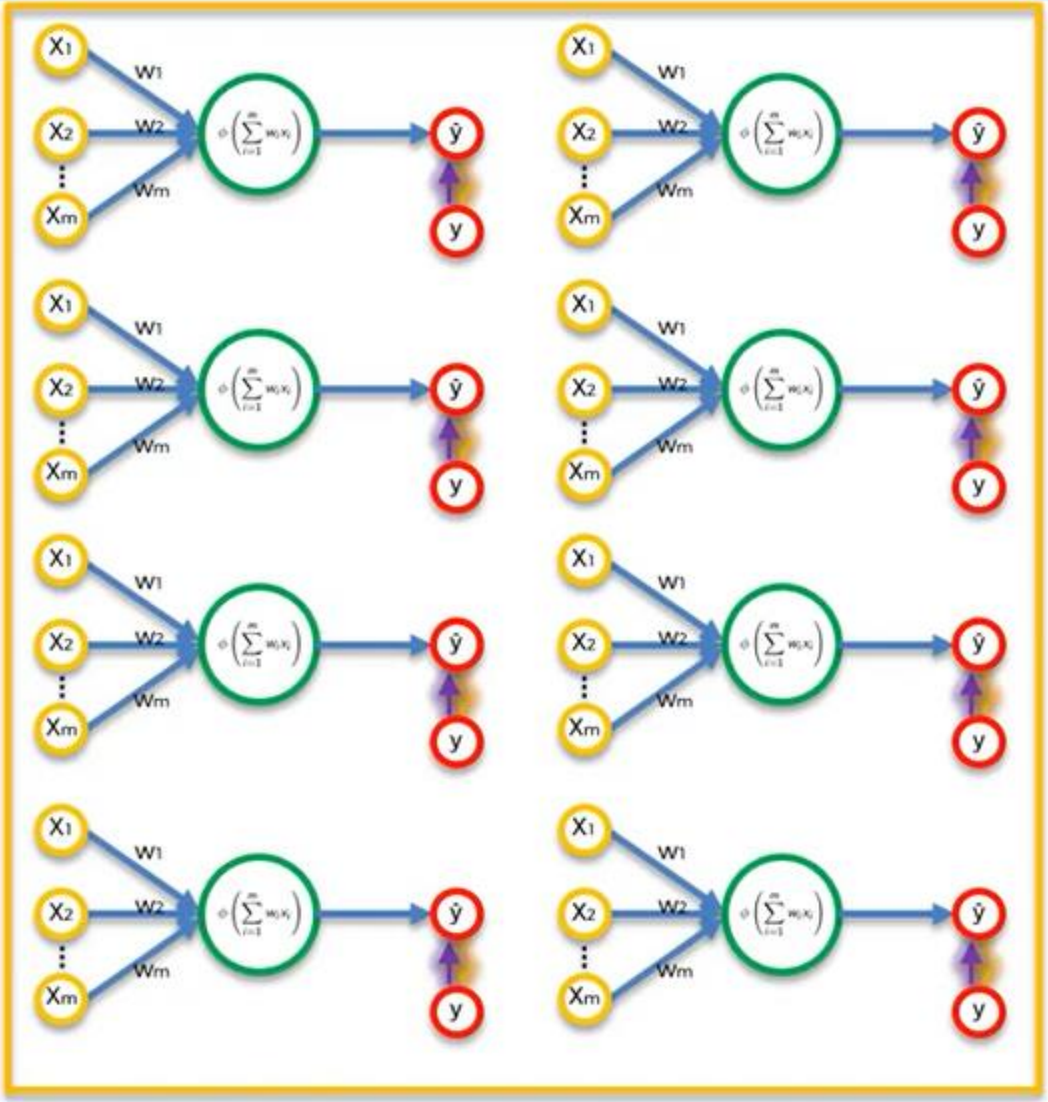
Stochastic Gradient Descent



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%

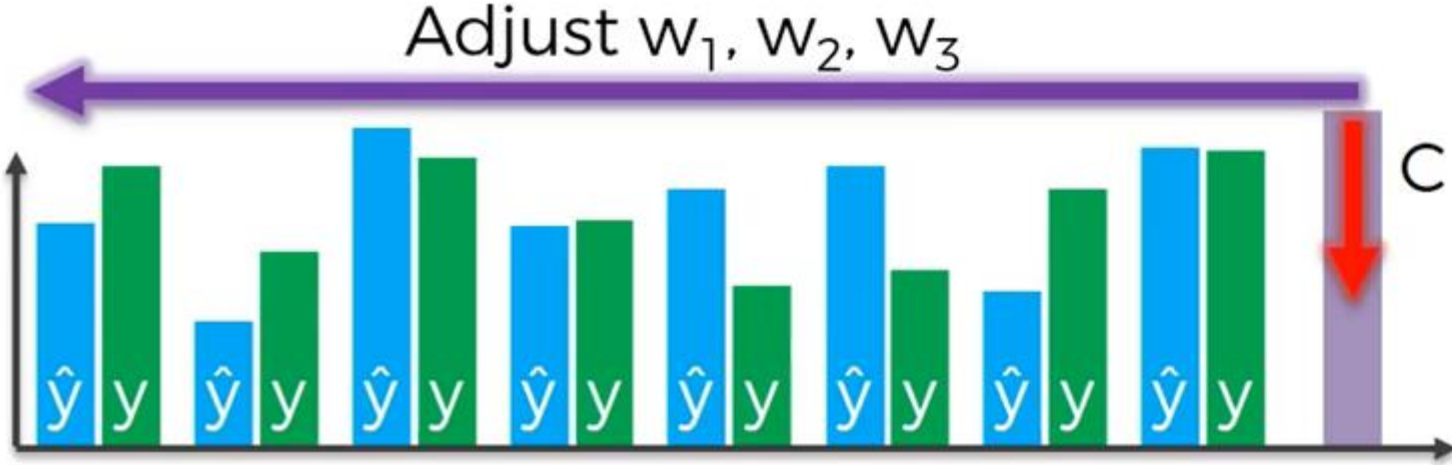
$C = \sum \frac{1}{2}(\hat{y}- y$

Stochastic Gradient Descent

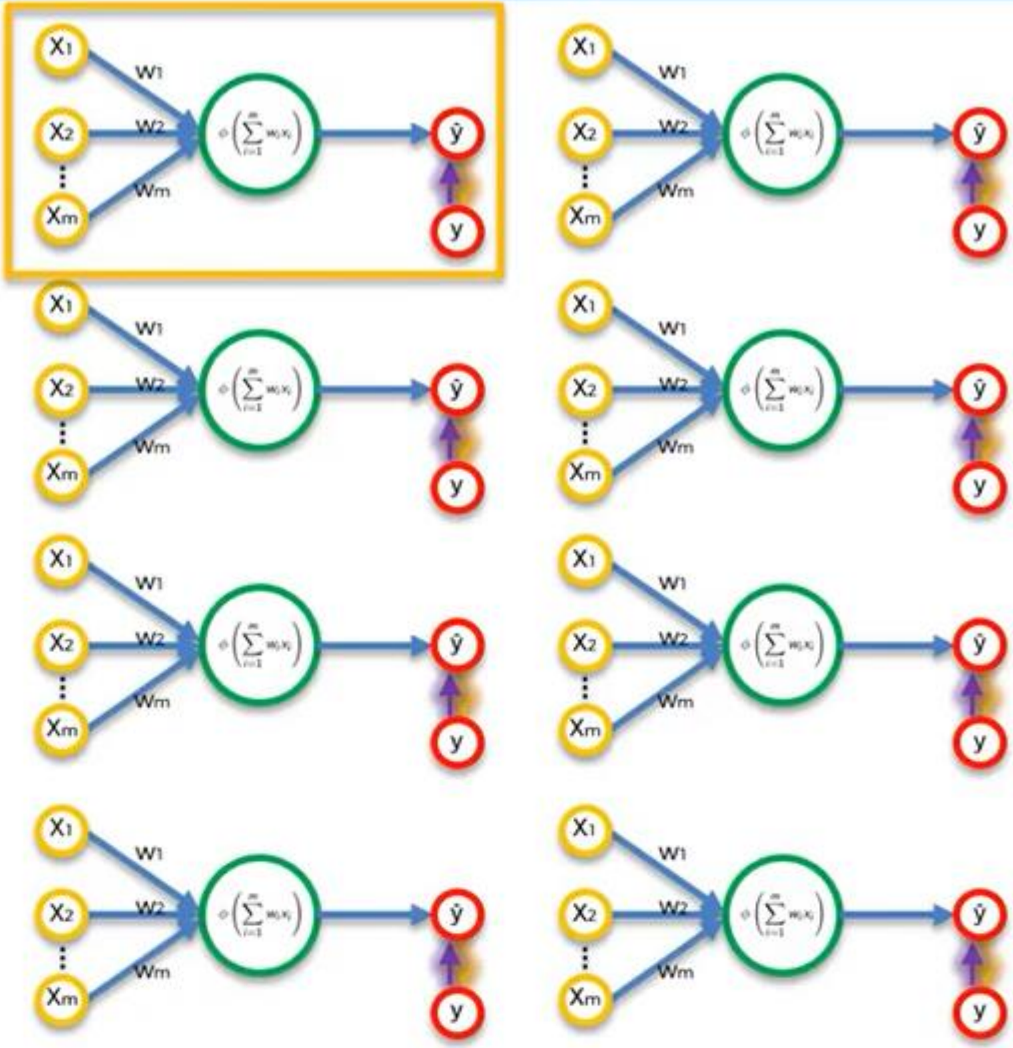


Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

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

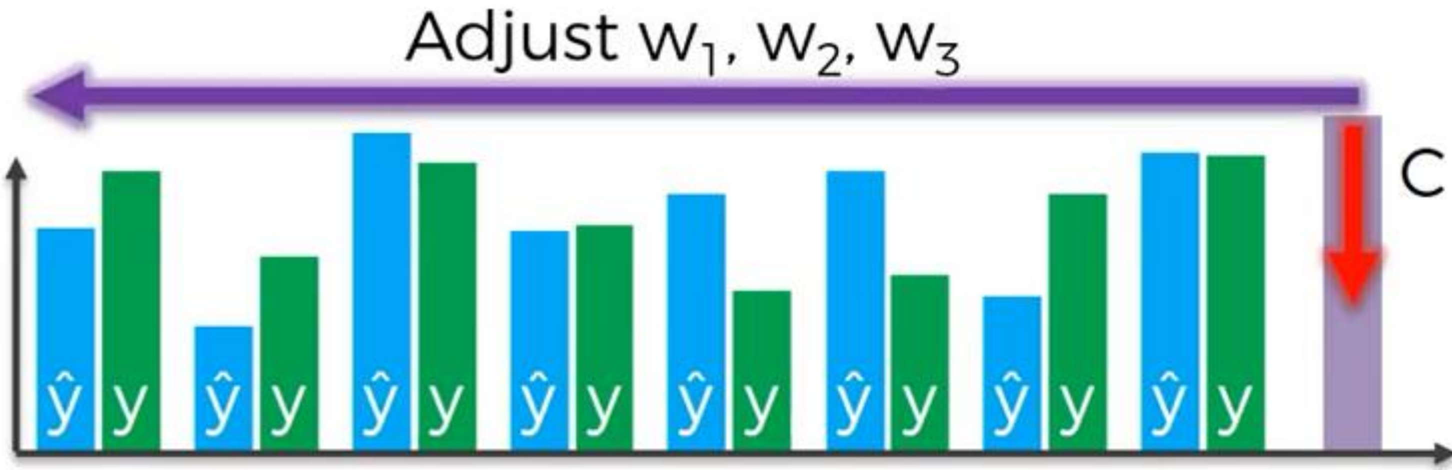


Stochastic Gradient Descent

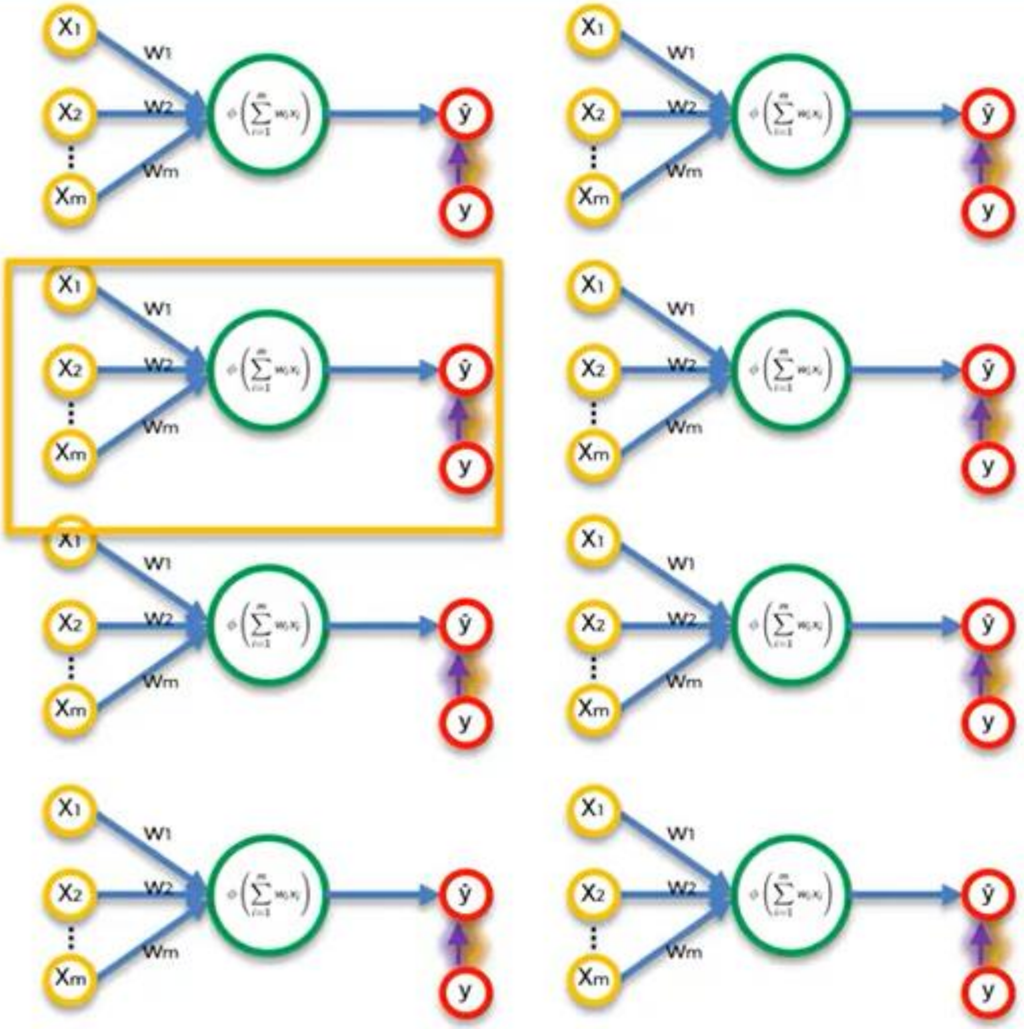


Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

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



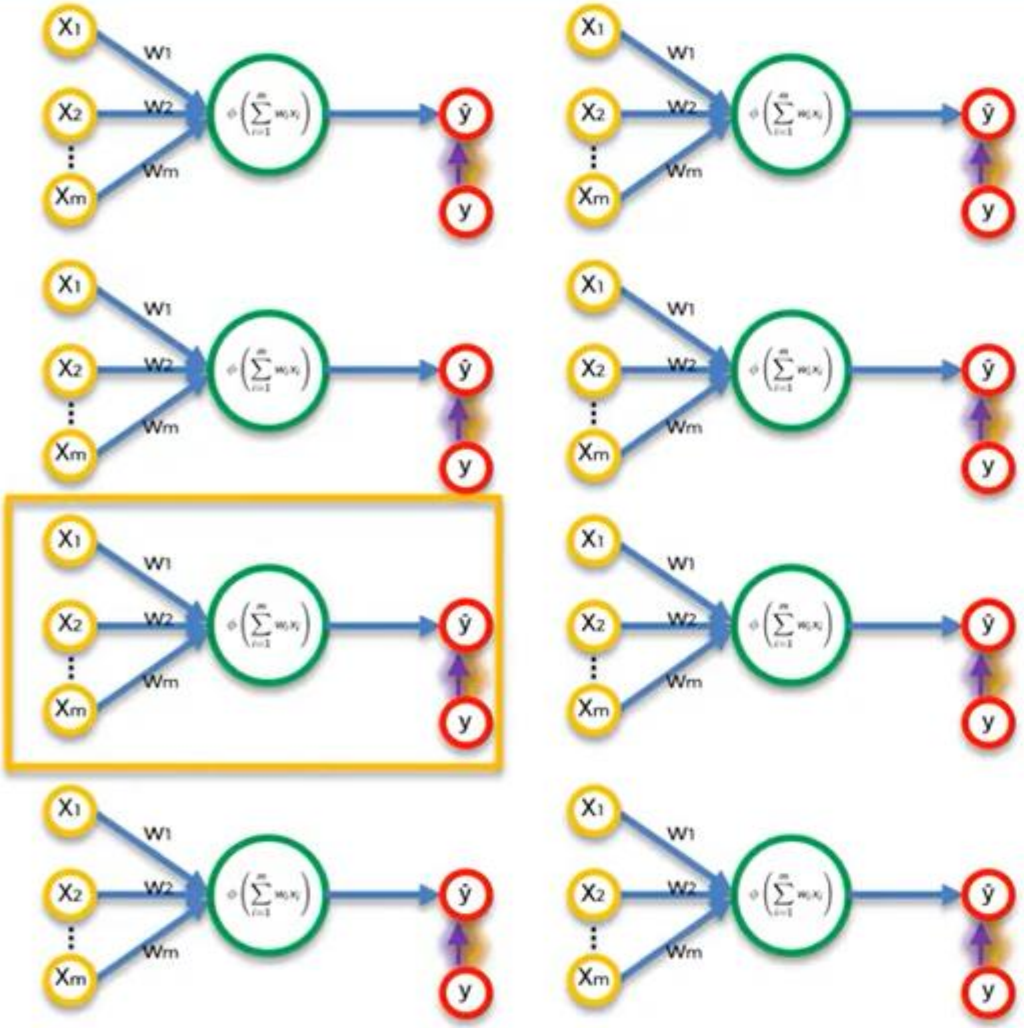
Stochastic Gradient Descent



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%

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

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%

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

Stochastic Gradient Descent

d w's

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%

Batch Gradient Descent

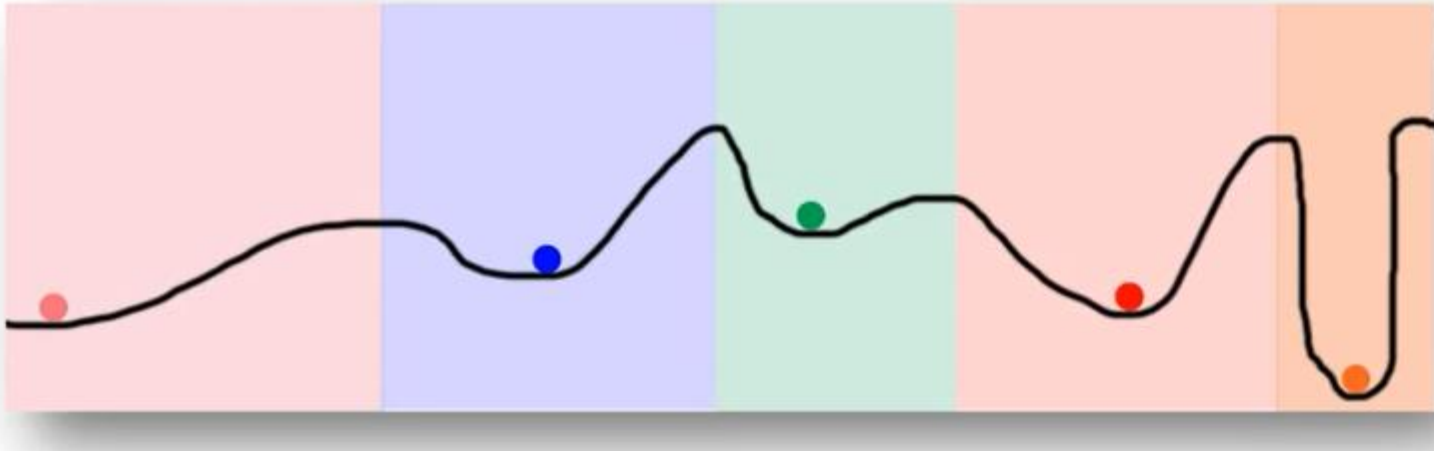
	Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
Upd w's	1	12	6	78%	93%
Upd w's	2	22	6.5	24%	68%
Upd w's	3	115	4	100%	95%
Upd w's	4	31	9	67%	75%
Upd w's	5	0	10	58%	51%
Upd w's	6	5	8	78%	60%
Upd w's	7	92	6	82%	89%
Upd w's	8	57	8	91%	97%

Stochastic Gradient Descent

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/

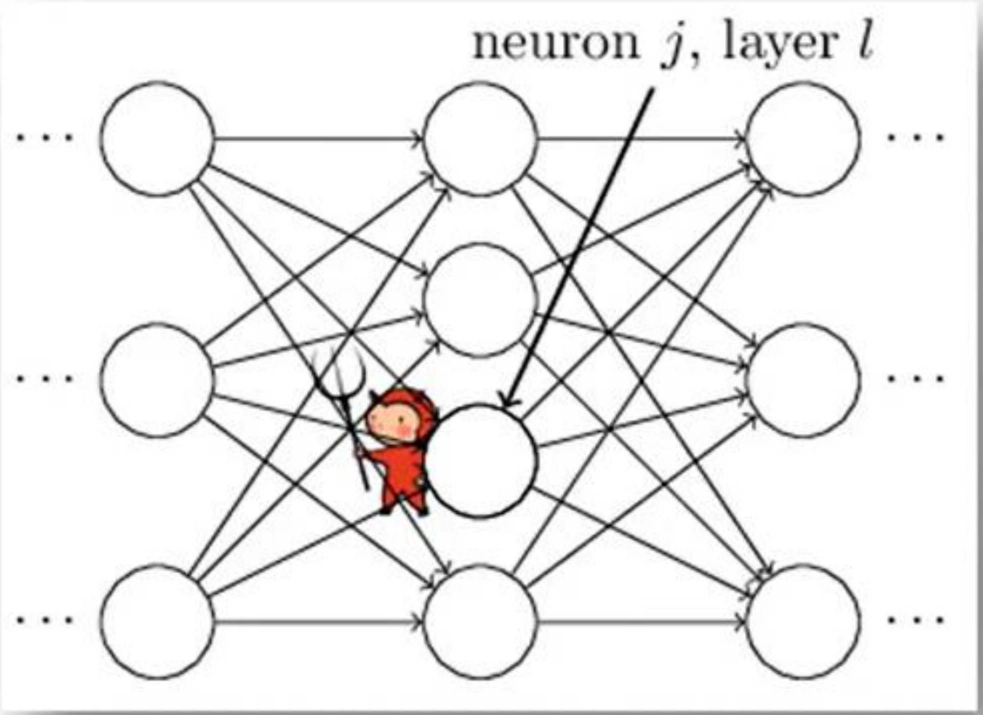
Stochastic Gradient Descent

Additional Reading:

Neural Networks and Deep Learning

Michael Nielsen (2015)

Link:



http://neuralnetworksanddeeplearning.com/chap2.html

Backpropagation

Gradient Descent

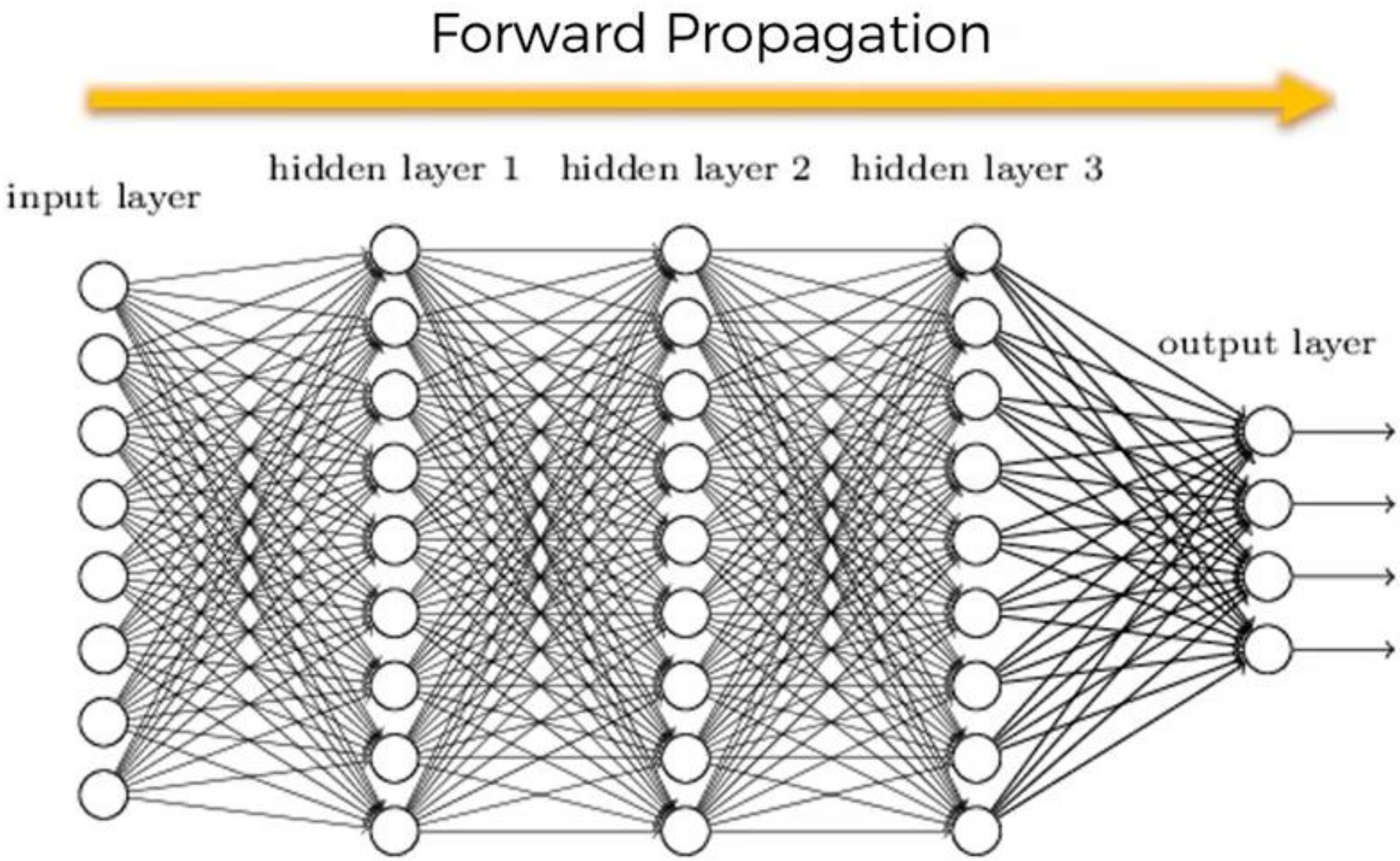


Image Source: neuralnetworksanddeeplearning.com

Gradient Descent

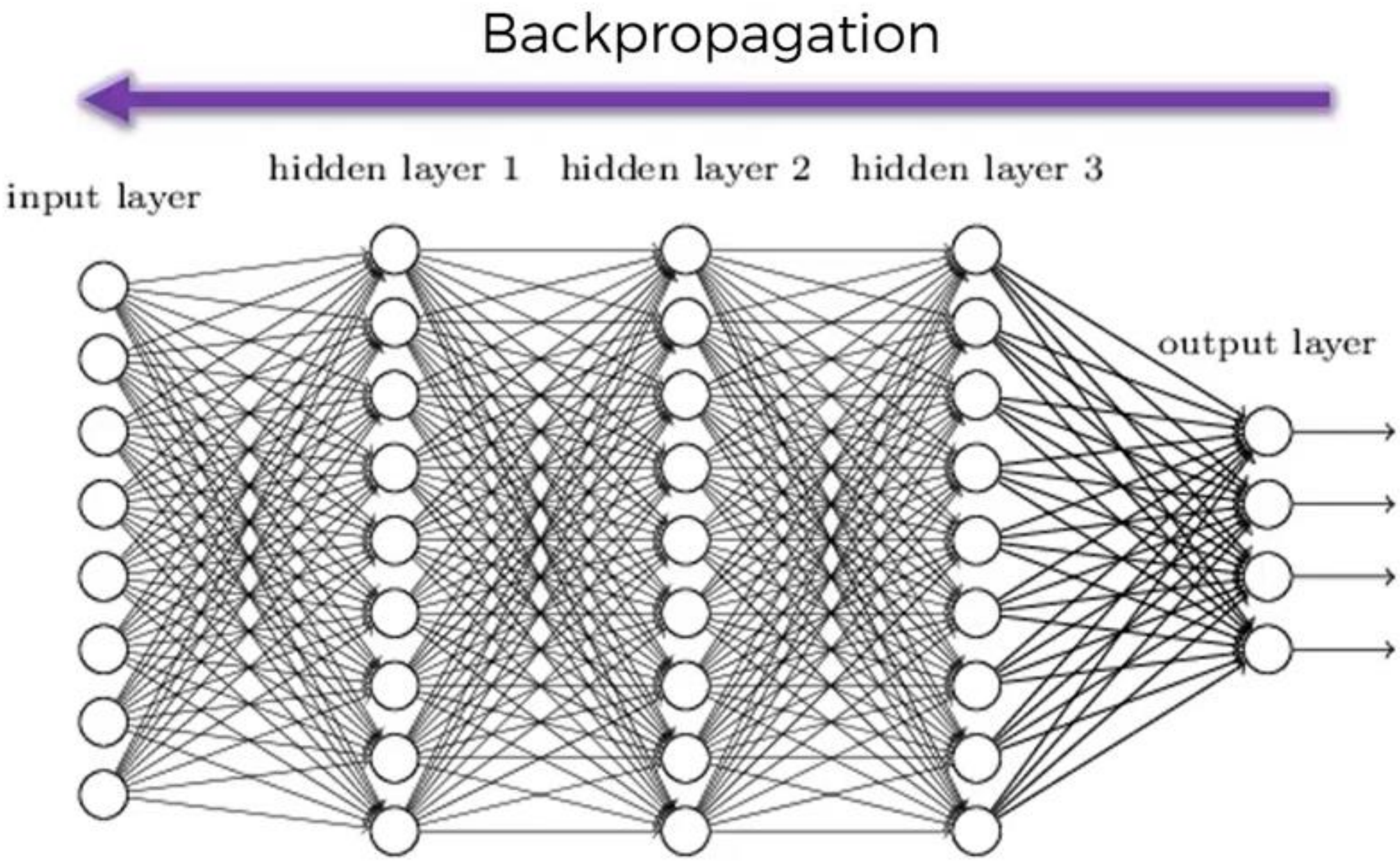


Image Source: neuralnetworksanddeeplearning.com

Training the ANN with Stochastic Gradient Descent

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

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

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

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

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

TEP 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).

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

Data Set – Churn Analysis

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.88	1
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
3	15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.57	1
4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.8	1	1	1	79084.1	0
6	15574012	Chu	645	Spain	Male	44	8	113755.8	2	1	0	149756.71	1
7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
8	15656148	Obinna	376	Germany	Female	29	4	115046.7	4	1	0	119346.88	1
9	15792365	He	501	France	Male	44	4	142051.1	2	0	1	74940.5	0
10	15592389	H?	684	France	Male	27	2	134603.9	1	1	1	71725.73	0
11	15767821	Bearce	528	France	Male	31	6	102016.7	2	0	0	80181.12	0
12	15737173	Andrews	497	Spain	Male	24	3	0	2	1	0	76390.01	0
13	15632264	Kay	476	France	Female	34	10	0	2	1	0	26260.98	0
14	15691483	Chin	549	France	Female	25	5	0	2	0	0	190857.79	0
15	15600882	Scott	635	Spain	Female	35	7	0	2	1	1	65951.65	0
16	15643966	Goforth	616	Germany	Male	45	3	143129.4	2	0	1	64327.26	0
17	15737452	Romeo	653	Germany	Male	58	1	132602.9	1	1	0	5097.67	1
18	15788218	Hendersor	549	Spain	Female	24	9	0	2	1	1	14406.41	0
19	15661507	Muldrow	587	Spain	Male	45	6	0	1	0	0	158684.81	0
20	15568982	Hao	726	France	Female	24	6	0	2	1	1	54724.03	0

Data Set – Churn Analysis

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.88	1
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
3	15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.57	1
4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.8	1	1	1	79084.1	0
6	15574012	Chu	645	Spain	Male	44	8	113755.8	2	1	0	149756.71	1
7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
8	15656148	Obinna	376	Germany	Female	29	4	115046.7	4	1	0	119346.88	1
9	15792365	He	501	France	Male	44	4	142051.1	2	0	1	74940.5	0
10	15592389	H?	684	France	Male	27	2	134603.9	1	1	1	71725.73	0
11	15767821	Bearce	528	France	Male	31	6	102016.7	2	0	0	80181.12	0
12	15737173	Andrews	497	Spain	Male	24	3	0	2	1	0	76390.01	0
13	15632264	Kay	476	France	Female	34	10	0	2	1	0	26260.98	0
14	15691483	Chin	549	France	Female	25	5	0	2	0	0	190857.79	0
15	15600882	Scott	635	Spain	Female	35	7	0	2	1	1	65951.65	0
16	15643966	Goforth	616	Germany	Male	45	3	143129.4	2	0	1	64327.26	0
17	15737452	Romeo	653	Germany	Male	58	1	132602.9	1	1	0	5097.67	1
18	15788218	Henderson	549	Spain	Female	24	9	0	2	1	1	14406.41	0
19	15661507	Muldrow	587	Spain	Male	45	6	0	1	0	0	158684.81	0
20	15568982	Hao	726	France	Female	24	6	0	2	1	1	54724.03	0

Data Set – Churn Analysis

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.88	1
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
3	15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.57	1
4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.8	1	1	1	79084.1	0
6	15574012	Chu	645	Spain	Male	44	8	113755.8	2	1	0	149756.71	1
7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
8	15656148	Obinna	376	Germany	Female	29	4	115046.7	4	1	0	119346.88	1
9	15792365	He	501	France	Male	44	4	142051.1	2	0	1	74940.5	0
10	15592389	H?	684	France	Male	27	2	134603.9	1	1	1	71725.73	0
11	15767821	Bearce	528	France	Male	31	6	102016.7	2	0	0	80181.12	0
12	15737173	Andrews	497	Spain	Male	24	3	0	2	1	0	76390.01	0
13	15632264	Kay	476	France	Female	34	10	0	2	1	0	26260.98	0
14	15691483	Chin	549	France	Female	25	5	0	2	0	0	190857.79	0
15	15600882	Scott	635	Spain	Female	35	7	0	2	1	1	65951.65	0
16	15643966	Goforth	616	Germany	Male	45	3	143129.4	2	0	1	64327.26	0
17	15737452	Romeo	653	Germany	Male	58	1	132602.9	1	1	0	5097.67	1
18	15788218	Henderson	549	Spain	Female	24	9	0	2	1	1	14406.41	0
19	15661507	Muldrow	587	Spain	Male	45	6	0	1	0	0	158684.81	0
20	15568982	Hao	726	France	Female	24	6	0	2	1	1	54724.03	0

Geography: OneHotEncoding
Gender: Label Encoding

Data Set – Churn Analysis

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.88	1
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
3	15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.57	1
4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.8	1	1	1	79084.1	0
6	15574012	Chu	645	Spain	Male	44	8	113755.8	2	1	0	149756.71	1
7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
8	15656148	Obinna	376	Germany	Female	29	4	115046.7	4	1	0	119346.88	1
9	15792365	He	501	France	Male	44	4	142051.1	2	0	1	74940.5	0
10	15592389	H?	684	France	Male	27	2	134603.9	1	1	1	71725.73	0
11	15767821	Bearce	528	France	Male	31	6	102016.7	2	0	0	80181.12	0
12	15737173	Andrews	497	Spain	Male	24	3	0	2	1	0	76390.01	0
13	15632264	Kay	476	France	Female	34	10	0	2	1	0	26260.98	0
14	15691483	Chin	549	France	Female	25	5	0	2	0	0	190857.79	0
15	15600882	Scott	635	Spain	Female	35	7	0	2	1	1	65951.65	0
16	15643966	Goforth	616	Germany	Male	45	3	143129.4	2	0	1	64327.26	0
17	15737452	Romeo	653	Germany	Male	58	1	132602.9	1	1	0	5097.67	1
18	15788218	Henderson	549	Spain	Female	24	9	0	2	1	1	14406.41	0
19	15661507	Muldrow	587	Spain	Male	45	6	0	1	0	0	158684.81	0
20	15568982	Hao	726	France	Female	24	6	0	2	1	1	54724.03	0

Geography: OneHotEncoding
Gender: Label Encoding

Feature Scaling

Data Set – Churn Analysis

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.88	1
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
3	15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.57	1
4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.8	1	1	1	79084.1	0
6	15574012	Chu	645	Spain	Male	44	8	113755.8	2	1	0	149756.71	1
7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
8	15656148	Obinna	376	Germany	Female	29	4	115046.7	4	1	0	119346.88	1
9	15792365	He	501	France	Male	44	4	142051.1	2	0	1	74940.5	0
10	15592389	H?	684	France	Male	27	2	134603.9	1	1	1	71725.73	0
11	15767821	Bearce	528	France	Male	31	6	102016.7	2	0	0	80181.12	0
12	15737173	Andrews	497	Spain	Male	24	3	0	2	1	0	76390.01	0
13	15632264	Kay	476	France	Female	34	10	0	2	1	0	26260.98	0
14	15691483	Chin	549	France	Female	25	5	0	2	0	0	190857.79	0
15	15600882	Scott	635	Spain	Female	35	7	0	2	1	1	65951.65	0
16	15643966	Goforth	616	Germany	Male	45	3	143129.4	2	0	1	64327.26	0
17	15737452	Romeo	653	Germany	Male	58	1	132602.9	1	1	0	5097.67	1
18	15788218	Hendersor	549	Spain	Female	24	9	0	2	1	1	14406.41	0
19	15661507	Muldrow	587	Spain	Male	45	6	0	1	0	0	158684.81	0
20	15568982	Hao	726	France	Female	24	6	0	2	1	1	54724.03	0

Python Code: Preprocessing

```python
# Importing the libraries
import numpy as np
import pandas as pd
import tensorflow as tf
tf.__version__

# Part 1 - Data Preprocessing

# Importing the dataset
dataset = pd.read_csv('Churn_Modelling.csv')
X = dataset.iloc[:, 3:-1].values
y = dataset.iloc[:, -1].values
print(X)
print(y)
```

Import Libraries and Load Data

dataset - DataFrame

Index	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	umOfProduc	HasCrCard	ActiveMemb	stimatedSalar	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101349	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.9	1	0	1	112543	0
2	3	15619304	Onio	502	France	Female	42	8	159661	3	1	0	113932	1
3	4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.6	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125511	1	1	1	79084.1	0
5	6	15574012	Chu	645	Spain	Male	44	8	113756	2	1	0	149757	1
6	7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
7	8	15656148	Obinna	376	Germany	Female	29	4	115047	4	1	0	119347	1
8	9	15792365	He	501	France	Male	44	4	142051	2	0	1	74940.5	0

Python Code: Preprocessing

Gender: Label Encoding
Geography: OneHotEncoding

dataset - DataFrame

Index	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	umOfProduc	HasCrCard	ActiveMembe	stimatedSalar	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101349	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.9	1	0	1	112543	0
2	3	15619304	Onio	502	France	Female	42	8	159661	3	1	0	113932	1
3	4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.6	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125511	1	1	1	79084.1	0
5	6	15574012	Chu	645	Spain	Male	44	8	113756	2	1	0	149757	1
6	7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
7	8	15656148	Obinna	376	Germany	Female	29	4	115047	4	1	0	119347	1
8	9	15792365	He	501	France	Male	44	4	142051	2	0	1	74940.5	0

```python
# Encoding categorical data
# Label Encoding the "Gender" column
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:, 2] = le.fit_transform(X[:, 2])
print(X)
# One Hot Encoding the "Geography" column
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
print(X)
```

Python Code: Preprocessing

Gender: Label Encoding
Geography: OneHotEncoding

dataset - DataFrame

Index	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	umOfProduc	HasCrCard	ActiveMembe	stimatedSalar	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101349	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.9	1	0	1	112543	0
2	3	15619304	Onio	502	France	Female	42	8	159661	3	1	0	113932	1
3	4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.6	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125511	1	1	1	79084.1	0
5	6	15574012	Chu	645	Spain	Male	44	8	113756	2	1	0	149757	1
6	7	155										1	10062.8	0
7	8	156										0	119347	1
8	9	157										1	74940.5	0

X - NumPy object array (read only)

	0	1	2	3	4
0	619	France	0	42	2
1	608	Spain	0	41	1
2	502	France	0	42	8
3	699	France	0	39	1
4	850	Spain	0	43	2
5	645	Spain	1	44	8
6	822	France	1	50	7
7	376	Germany	0	29	4
8	501	France	1	44	4

```
# Encoding categorical data
# Label Encoding the "Gender" column
from sklearn.preprocessing import LabelEnc
le = LabelEncoder()
X[:, 2] = le.fit_transform(X[:, 2])
print(X)
# One Hot Encoding the "Geography" column
from sklearn.compose import ColumnTransfor
from sklearn.preprocessing import OneHotEn
ct = ColumnTransformer(transformers=[('enc                    gh')
X = np.array(ct.fit_transform(X))
print(X)
```

Python Code: Preprocessing

X - NumPy object array (read only)

	0	1	2	3	4	5
0	1.0	0.0	0.0	619	0	42
1	0.0	0.0	1.0	608	0	41
2	1.0	0.0	0.0	502	0	42
3	1.0	0.0	0.0	699	0	39
4	0.0	0.0	1.0	850	0	43
5	0.0	0.0	1.0	645	1	44
6	1.0	0.0	0.0	822	1	50
7	0.0	1.0	0.0	376	0	29
8	1.0	0.0	0.0	501	1	44

Geography	Gender	Age	Tenure	Balance	umOfProduc	HasCrCard	ActiveMembe	stimatedSalar	Exited
France	Female	42	2	0	1	1	1	101349	1
Spain	Female	41	1	83807.9	1	0	1	112543	0
France	Female	42	8	159661	3	1	0	113932	1
France	Female	39	1	0	2	0	0	93826.6	0
Spain	Female	43	2	125511	1	1	1	79084.1	0
Spain	Male	44	8	113756	2	1	0	149757	1
France	Male	50	7	0	2	1	1	10062.8	0
Germany	Female	29	4	115047	4	1	0	119347	1
France	Male	44	4	142051	2	0	1	74940.5	0

Gender: L

Geograph

```python
# Encoding categorical data
# Label Encoding the "Gender" column
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:, 2] = le.fit_transform(X[:, 2])
print(X)
# One Hot Encoding the "Geography" column
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
print(X)
```

Python Code: Preprocessing Feature Scaling

```python
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
print(X)
```

Name	Type	Size
ct	compose._column_transformer.ColumnTransformer	1
dataset	DataFrame	(10000, 14)
le	preprocessing._label.LabelEncoder	1
sc	preprocessing._data.StandardScaler	1
X	Array of float64	(10000, 12)
y	Array of int64	(10000,)

Plots Help Variable Explorer Files

	3	4	5	6	7	8
0	619	0	42	2	0.0	1
1	608	0	41	1	83807.86	1
2	502	0	42	8	159660.8	3
3	699	0	39	1	0.0	2
4	850	0	43	2	125510.82	1
5	645	1	44	8	113755.78	2
6	822	1	50	7	0.0	2
7	376	0	29	4	115046.74	4
8	501	1	44	4	142051.07	2

	3	4	5	6	7	8
0	-0.326221	-1.09599	0.293517	-1.04176	-1.22585	-0.911583
1	-0.440036	-1.09599	0.198164	-1.38754	0.11735	-0.911583
2	-1.53679	-1.09599	0.293517	1.03291	1.33305	2.52706
3	0.501521	-1.09599	0.00745665	-1.38754	-1.22585	0.807737
4	2.06388	-1.09599	0.388871	-1.04176	0.785728	-0.911583
5	-0.0572053	0.912419	0.484225	1.03291	0.597329	0.807737
6	1.77417	0.912419	1.05635	0.68713	-1.22585	0.807737
7	-2.84049	-1.09599	-0.946079	-0.350204	0.618019	4.24638
8	-1.54714	0.912419	0.484225	-0.350204	1.05082	0.807737

Python Code: Preprocessing
Train Test Split

```python
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Name	Type	Size
ct	compose._column_transformer.ColumnTransformer	1
dataset	DataFrame	(10000, 14)
le	preprocessing._label.LabelEncoder	1
sc	preprocessing._data.StandardScaler	1
X	Array of float64	(10000, 12)
X_test	Array of float64	(2000, 12)
X_train	Array of float64	(8000, 12)
y	Array of int64	(10000,)
y_test	Array of int64	(2000,)
y_train	Array of int64	(8000,)

Python Code: Building the ANN

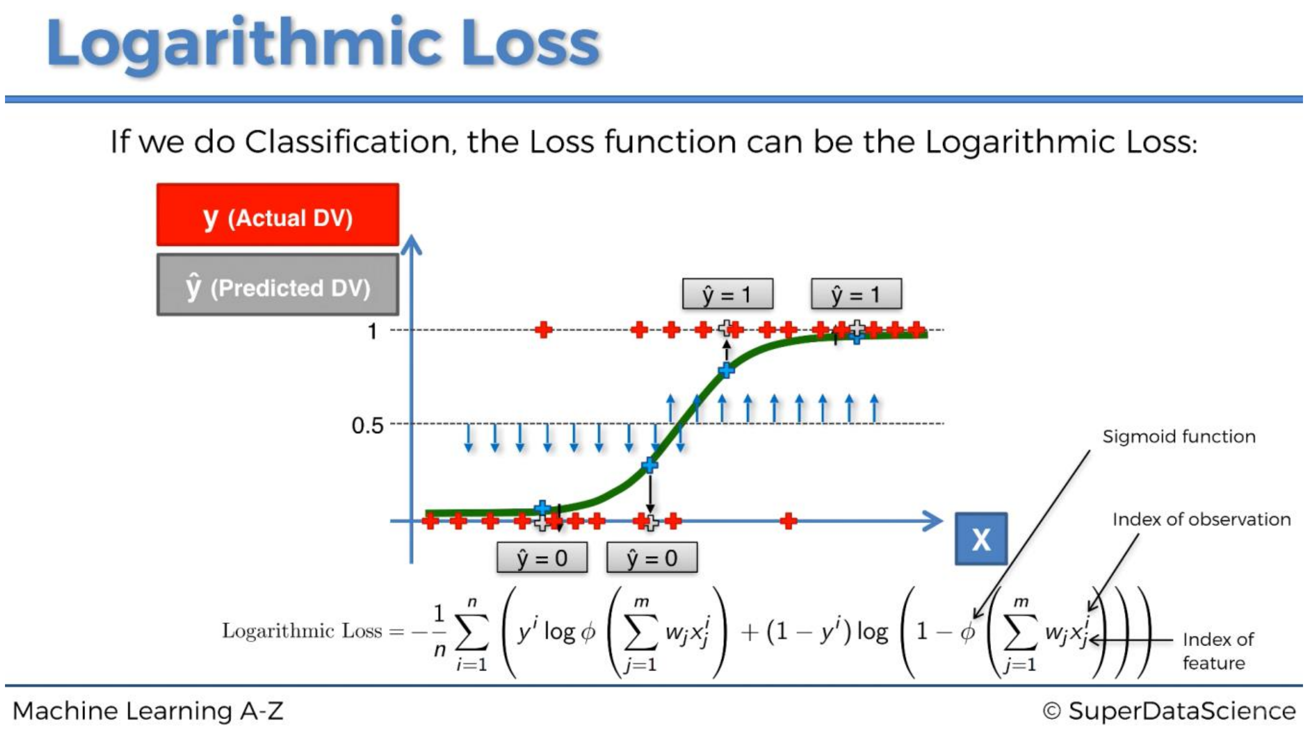
```python
# Part 2 - Building the ANN

# Initializing the ANN
ann = tf.keras.models.Sequential()

# Adding the input layer and the first hidden layer
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))

# Adding the second hidden layer
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))

# Adding the output layer
ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

Python Code: Building the ANN

```
# Part 2 - Now let's make the ANN!

# Importing the Keras libraries and packages
import keras
from keras.models import Sequential
from keras.layers import Dense

# Initialising the ANN
classifier = Sequential()

# Adding the input layer and the first hidden layer
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu', input_dim = 11))

# Adding the second hidden layer
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'))

# Adding the output layer
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))

# Compiling the ANN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

# Part 3 - Making the predictions and evaluating the model

# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

Python Code: Training the ANN

```python
# Part 3 - Training the ANN

# Compiling the ANN
ann.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

# Training the ANN on the Training set
ann.fit(X_train, y_train, batch_size = 32, epochs = 100)
```



Optmizer **adam** is an implementation of SGD

Binary_crossentropy is used in case of classification, where there is a binary output. While, **categorical_crossentropy** is used in case of more than 2 outputs.

Mertrics takes a list as input, that is why we are using square brackets

Python Code: Training the ANN

```python
# Part 3 - Training the ANN

# Compiling the ANN
ann.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

# Training the ANN on the Training set
ann.fit(X_train, y_train, batch_size = 32, epochs = 100)
```

```
loss: 0.4398 - accuracy: 0.7980
Epoch 5/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.4298 - accuracy: 0.8080
Epoch 6/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.4193 - accuracy: 0.8176
Epoch 7/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.4110 - accuracy: 0.8213
Epoch 8/100
250/250 [==============================] - 1s 4ms/step -
loss: 0.4063 - accuracy: 0.8232
Epoch 9/100
250/250 [==============================] - 1s 5ms/step -
loss: 0.4027 - accuracy: 0.8234
Epoch 10/100
250/250 [==============================] - 1s 4ms/step -
loss: 0.3990 - accuracy: 0.8271
Epoch 11/100
250/250 [==============================] - 1s 2ms/step -
loss: 0.3965 - accuracy: 0.8273
Epoch 12/100
 67/250 [=======>......................] - ETA: 0s - loss:
0.4071 - accuracy: 0.8204
```

```
Epoch 33/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.3552 - accuracy: 0.8553
Epoch 34/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.3541 - accuracy: 0.8559
Epoch 35/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.3531 - accuracy: 0.8576
Epoch 36/100
250/250 [==============================] - 1s 4ms/step -
loss: 0.3529 - accuracy: 0.8555
Epoch 37/100
250/250 [==============================] - 1s 5ms/step -
loss: 0.3521 - accuracy: 0.8572
Epoch 38/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.3514 - accuracy: 0.8565
Epoch 39/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.3503 - accuracy: 0.8569
Epoch 40/100
206/250 [=======================>......] - ETA: 0s - loss:
0.3478 - accuracy: 0.8563
```

Python Code: Training the ANN

```
Epoch 63/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.3421 - accuracy: 0.8596
Epoch 64/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.3422 - accuracy: 0.8624
Epoch 65/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.3422 - accuracy: 0.8612
Epoch 66/100
250/250 [==============================] - 1s 4ms/step -
loss: 0.3419 - accuracy: 0.8585
Epoch 67/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.3421 - accuracy: 0.8596
Epoch 68/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.3414 - accuracy: 0.8593
Epoch 69/100
250/250 [==============================] - 1s 2ms/step -
loss: 0.3417 - accuracy: 0.8614
Epoch 70/100
 67/250 [=======>......................] - ETA: 0s - loss:
0.3149 - accuracy: 0.8806
```

```
Epoch 94/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.3387 - accuracy: 0.8604
Epoch 95/100
250/250 [==============================] - 1s 4ms/step -
loss: 0.3384 - accuracy: 0.8619
Epoch 96/100
250/250 [==============================] - 1s 5ms/step -
loss: 0.3382 - accuracy: 0.8609
Epoch 97/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.3383 - accuracy: 0.8625
Epoch 98/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.3382 - accuracy: 0.8612
Epoch 99/100
250/250 [==============================] - 1s 4ms/step -
loss: 0.3381 - accuracy: 0.8614
Epoch 100/100
250/250 [==============================] - 1s 3ms/step -
loss: 0.3376 - accuracy: 0.8622
```

Python Code: Testing ANN

```python
# Part 4 - Making the predictions and evaluating the model

# Predicting the Test set results
y_pred = ann.predict(X_test)
y_pred = (y_pred > 0.5)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.re
```

y_pred - NumPy object array

	0
0	False
1	False
2	False
3	False
4	False
5	True
6	False
7	False
8	False
9	True
10	False
11	False
12	False

y_test - NumPy object array

	0
0	0
1	1
2	0
3	0
4	0
5	1
6	0
7	0
8	1
9	1
10	0
11	0
12	0

```python
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

cm - NumPy object array

	0	1
0	1511	84
1	205	200

Python Code: Confusion Matrix