

# Artificial Neural Networks

# Plan of Attack

# 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

# The Neuron

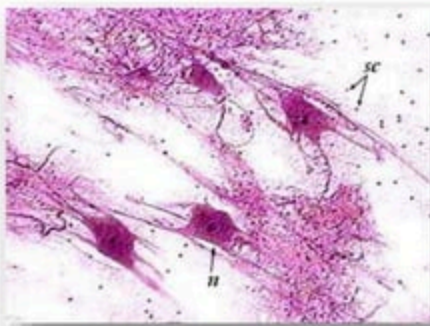
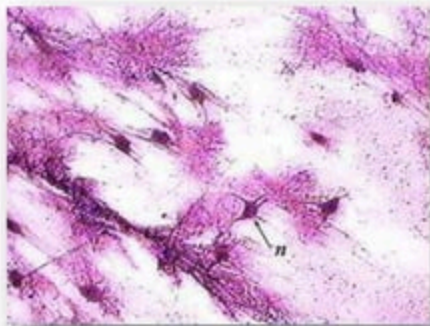


Image Source: [www.austincc.edu](http://www.austincc.edu)

# The Neuron

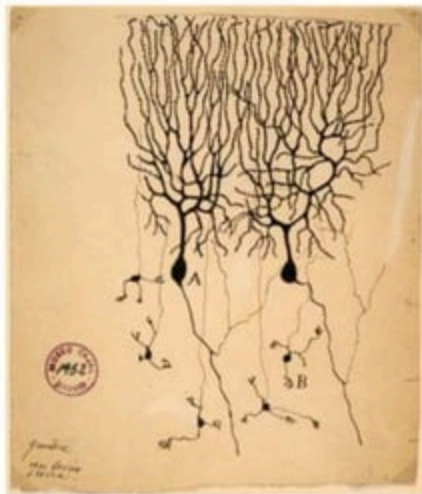
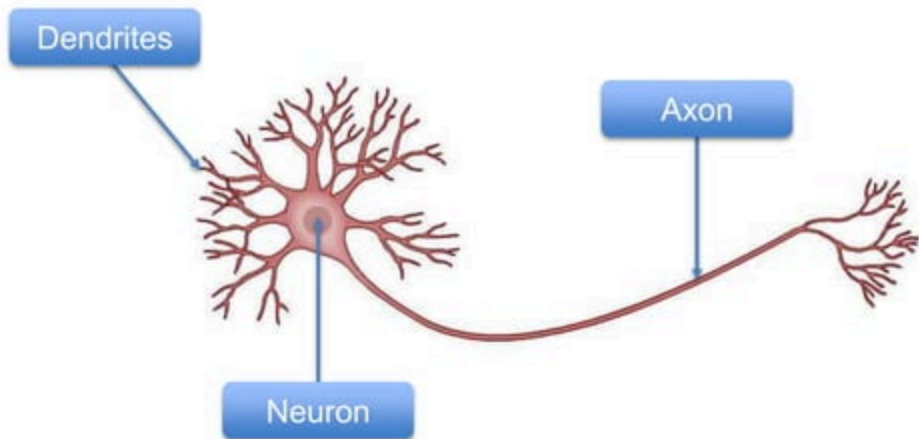


Image Source: Wikipedia

# The Neuron



*Image Source: Wikipedia*

# The Neuron

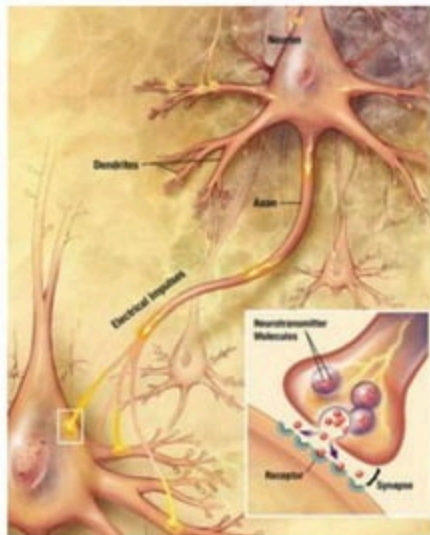
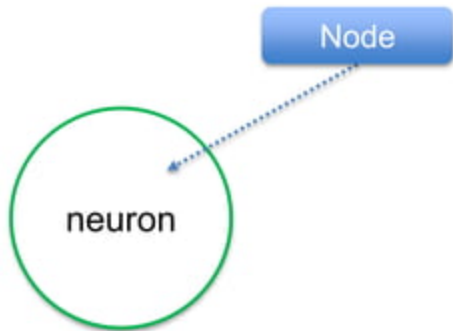


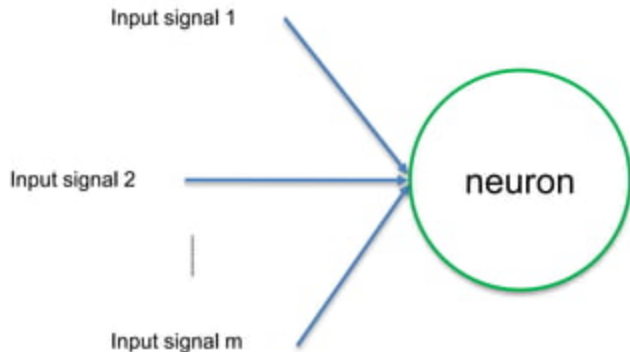
Image Source: Wikipedia



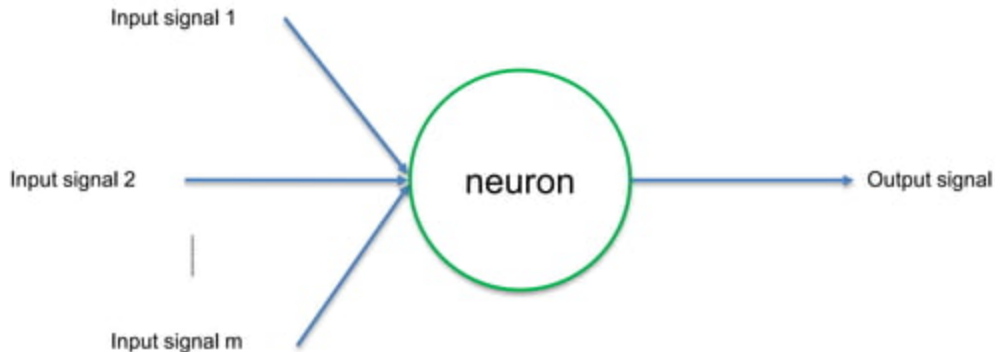
# The Neuron



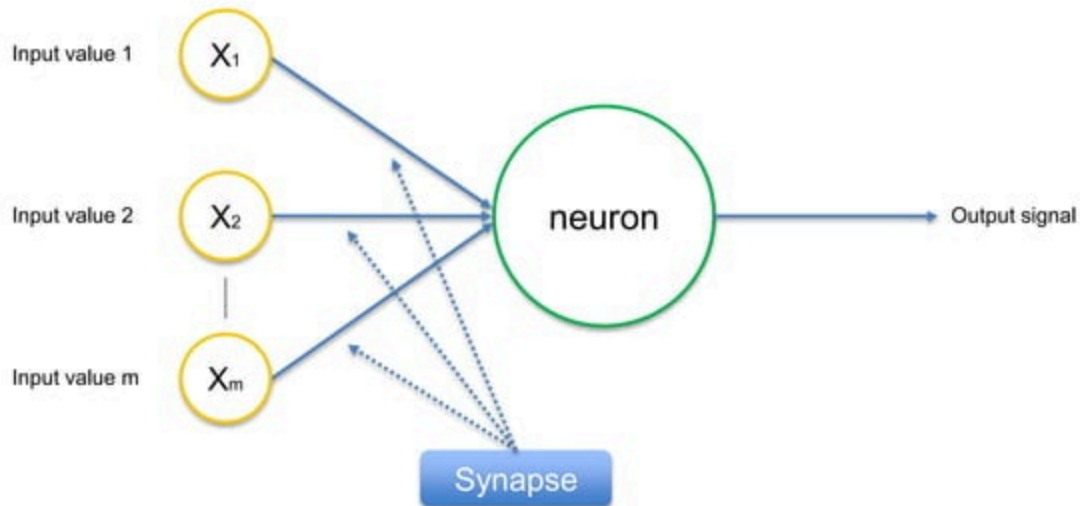
# The Neuron



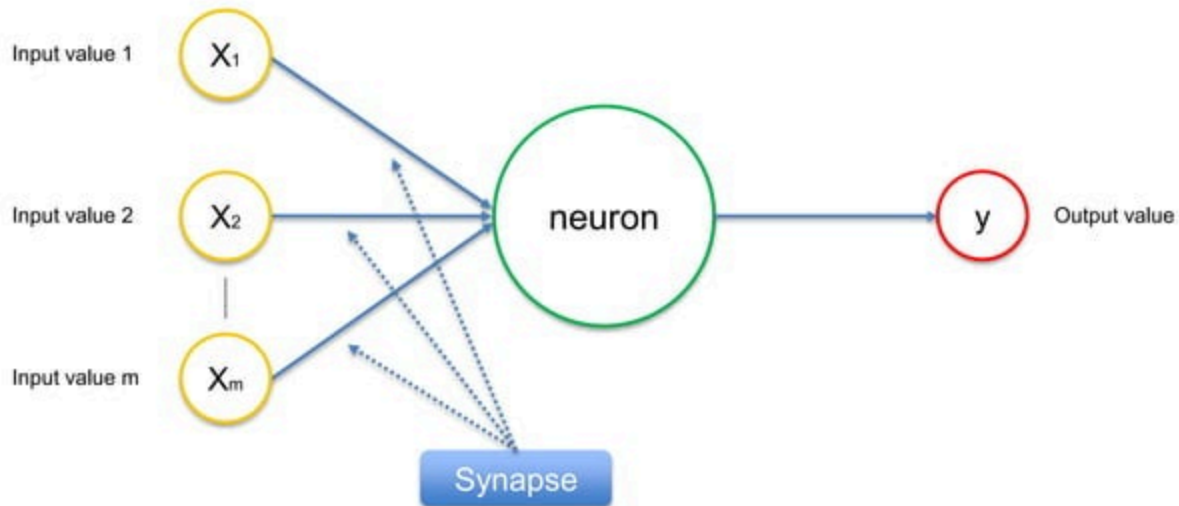
# The Neuron



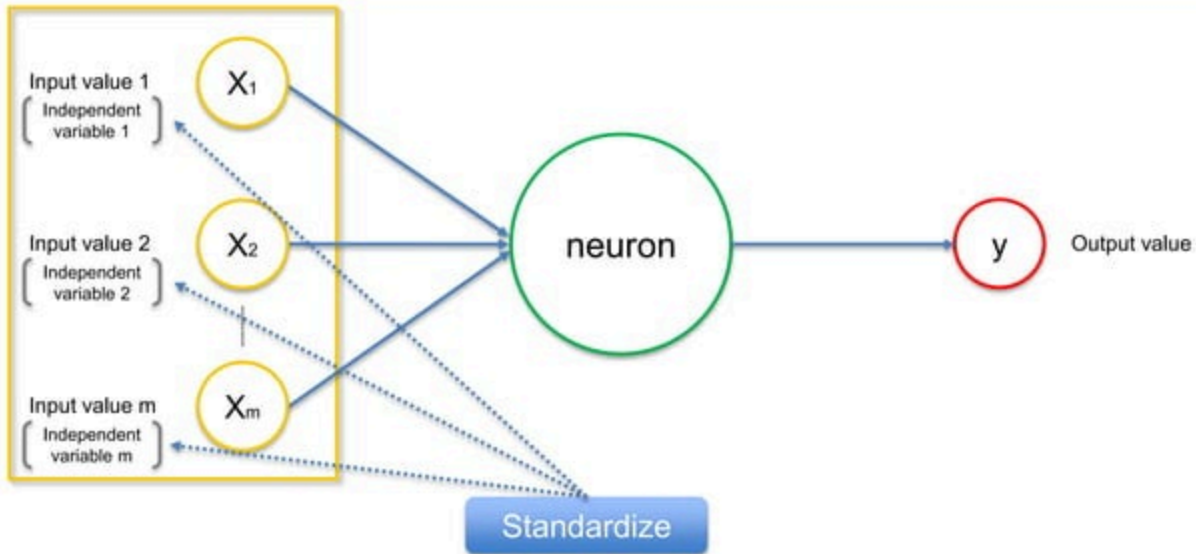
# The Neuron



# The Neuron



# The Neuron



# The Neuron

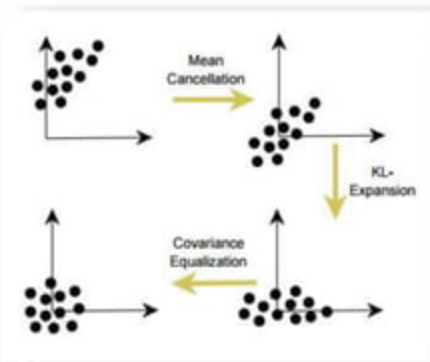
## Additional Reading:

*Efficient BackProp*

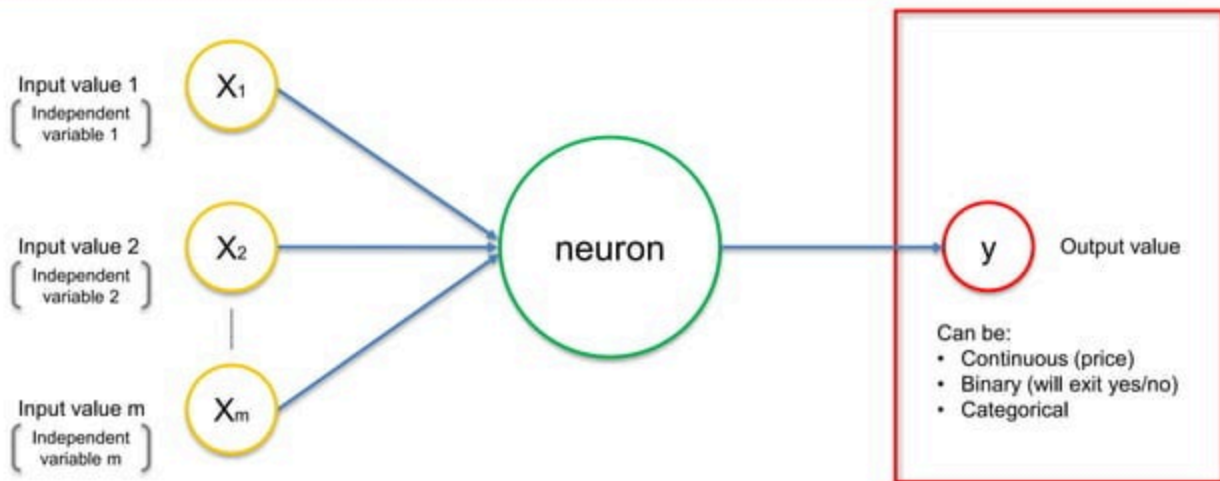
By Yann LeCun et al. (1998)

Link:

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

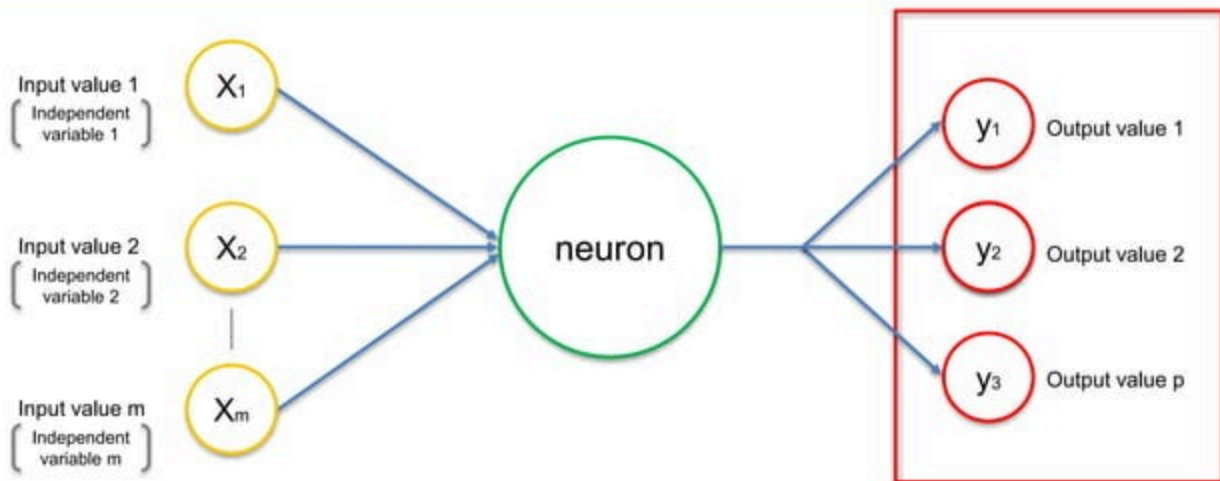


# The Neuron

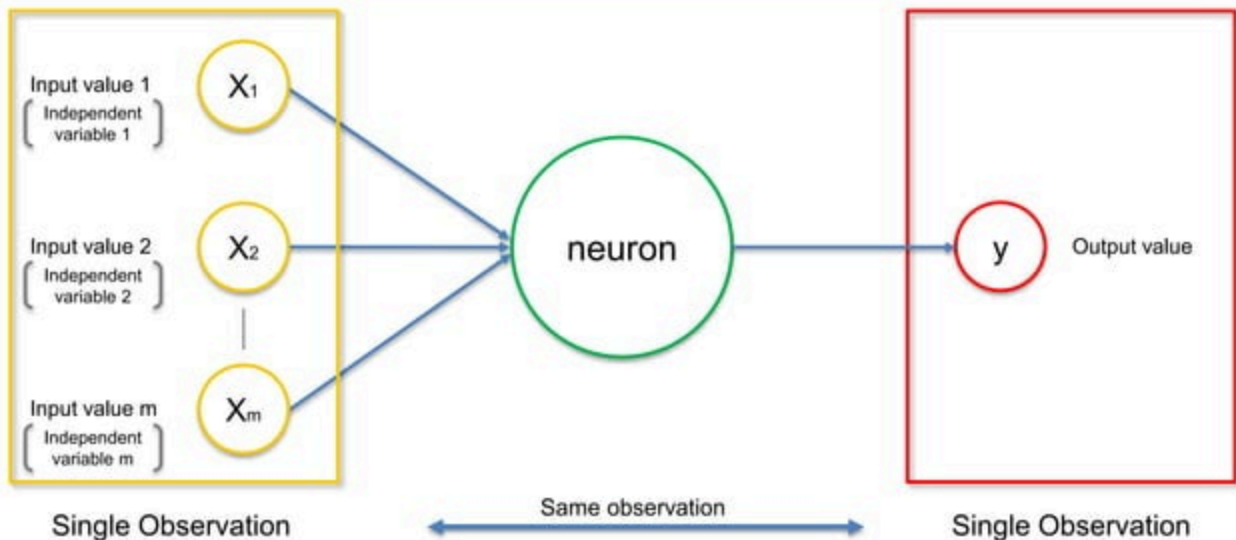




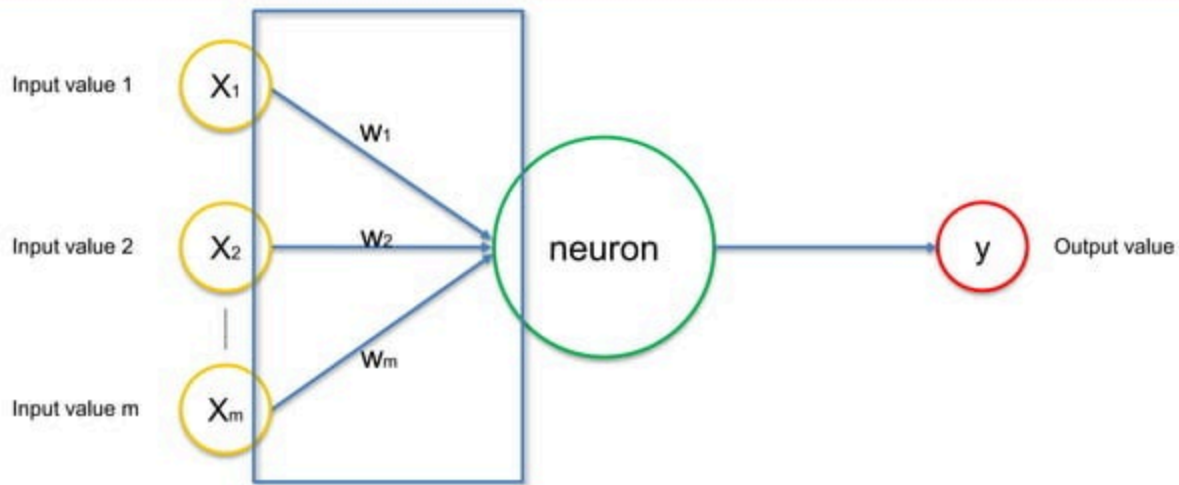
# The Neuron



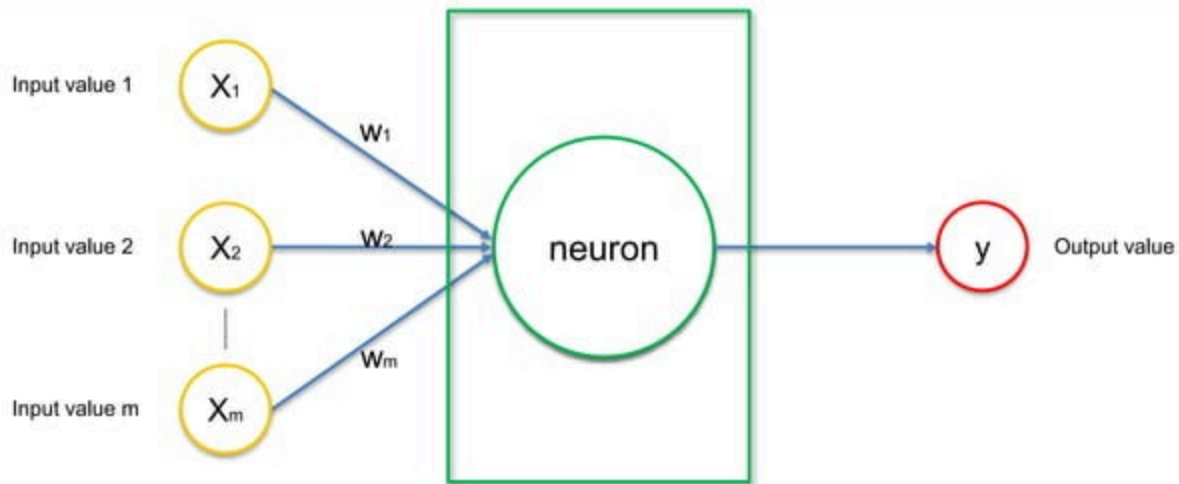
# The Neuron



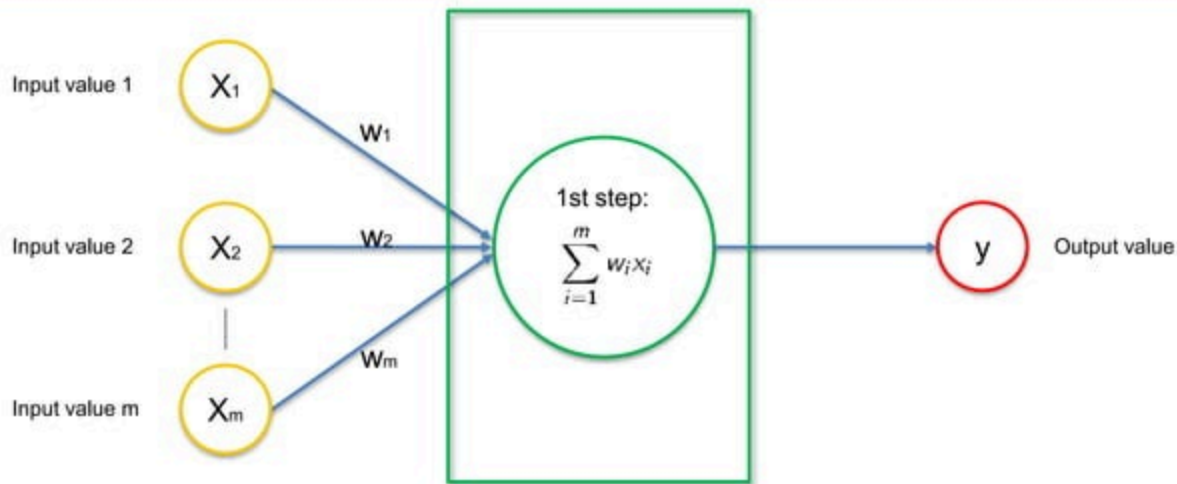
# The Neuron



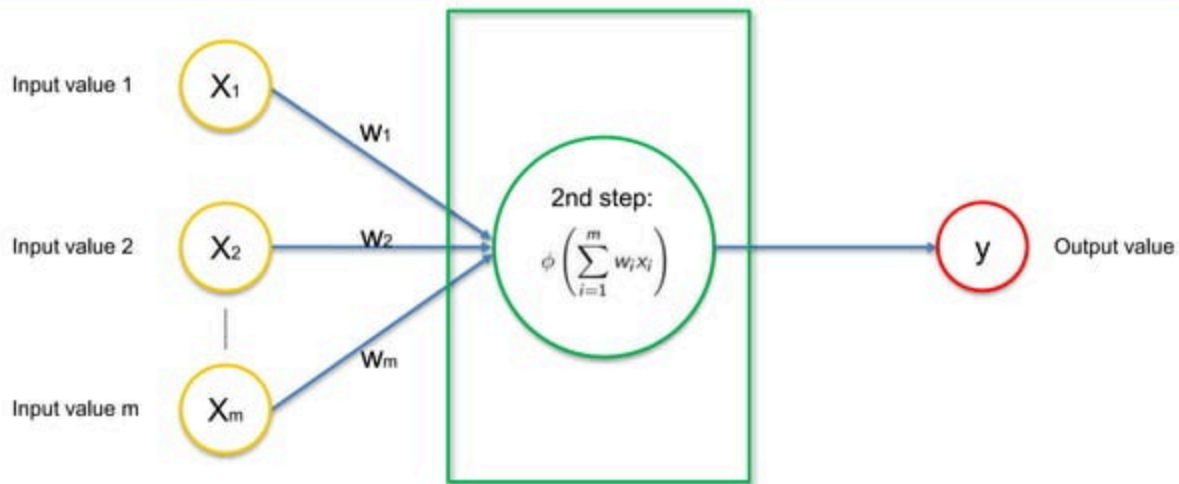
# The Neuron



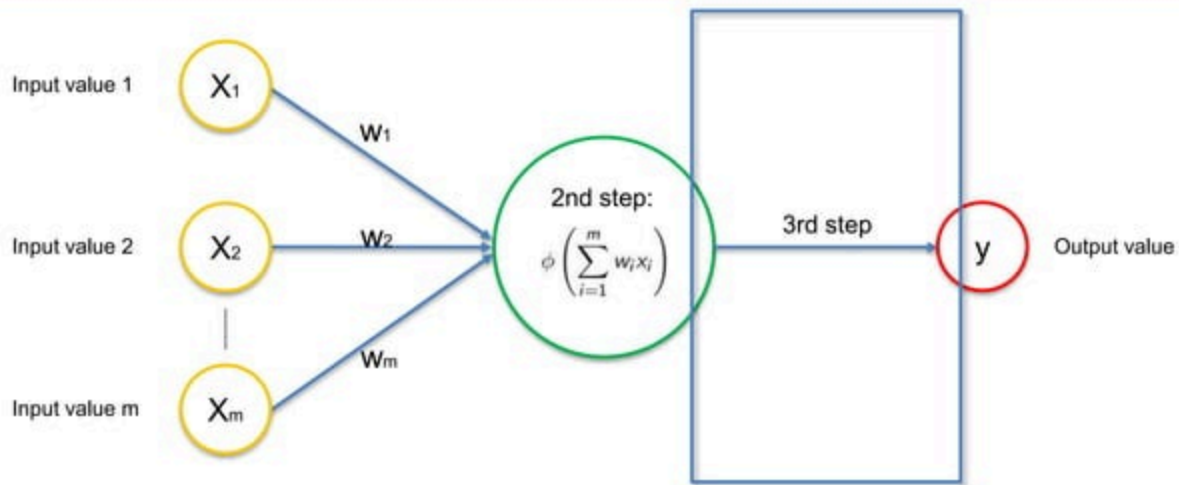
# The Neuron



# The Neuron



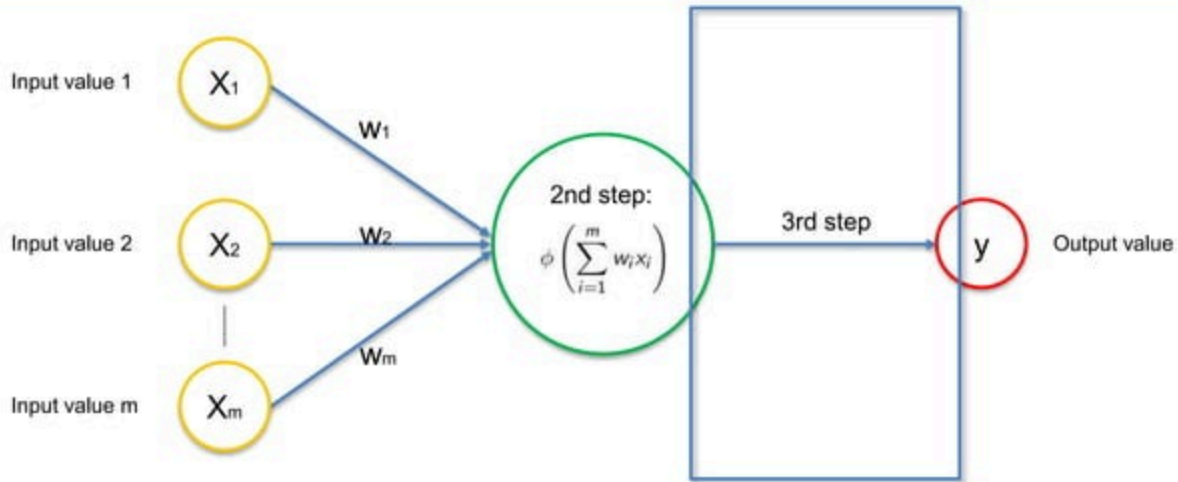
# The Neuron



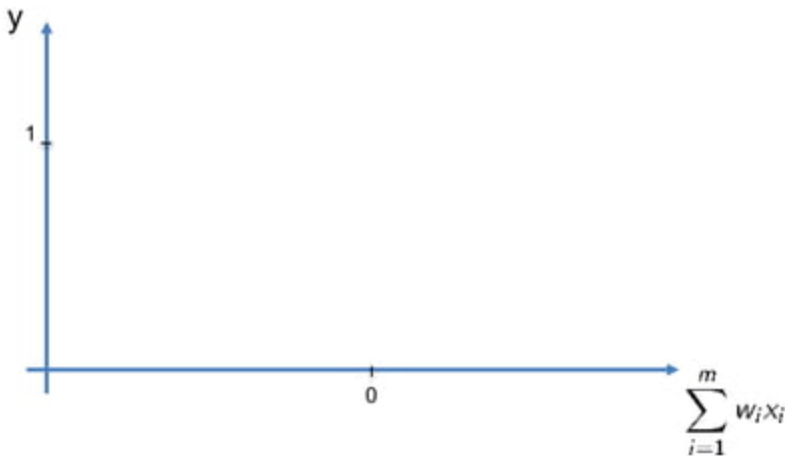
# The Activation Function



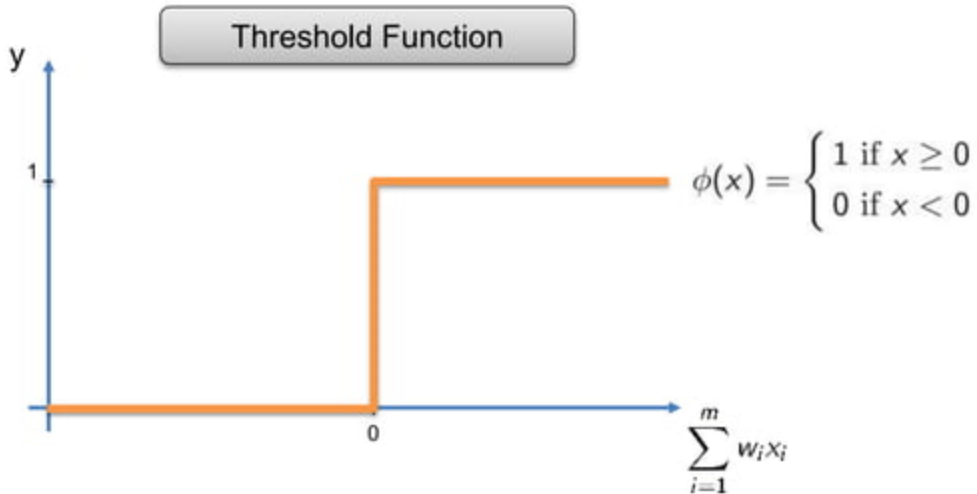
# The Activation Function



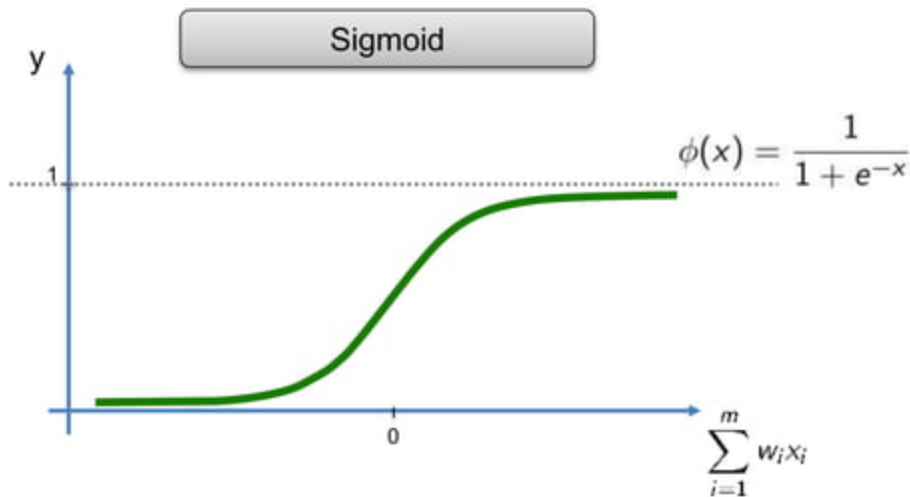
# The Activation Function



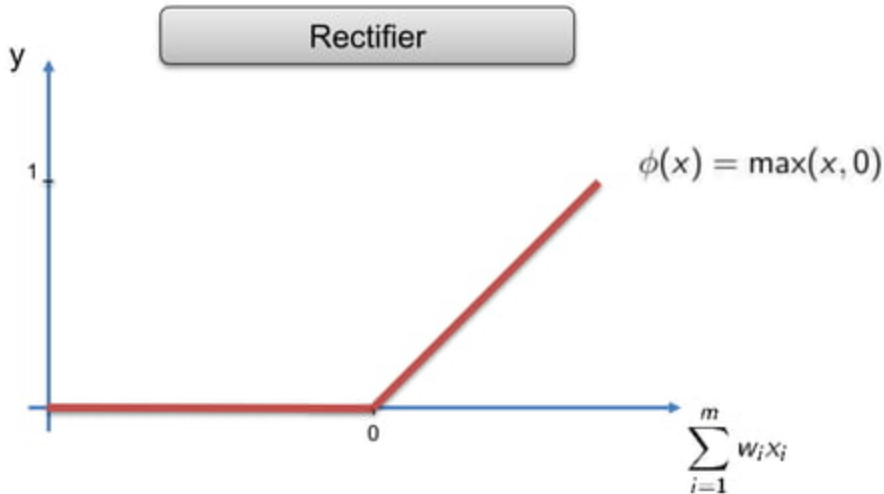
# The Activation Function



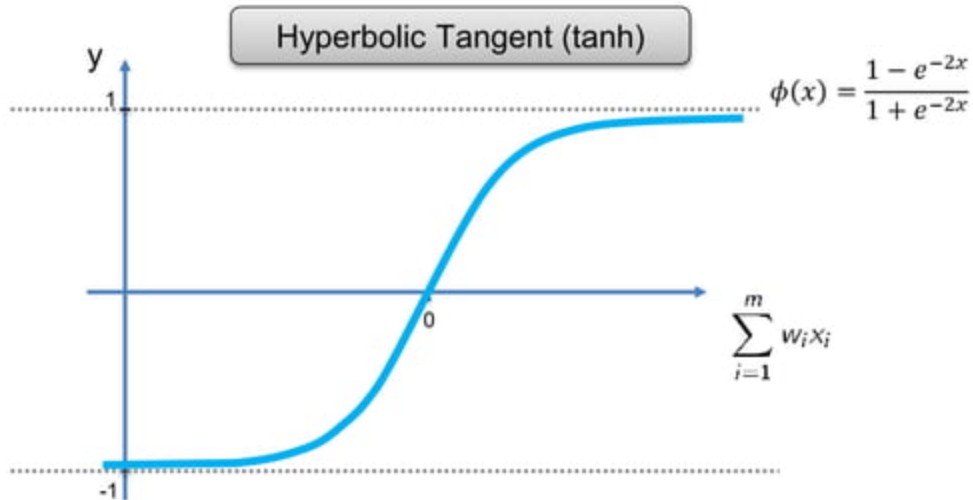
# The Activation Function



# The Activation Function



# The Activation Function

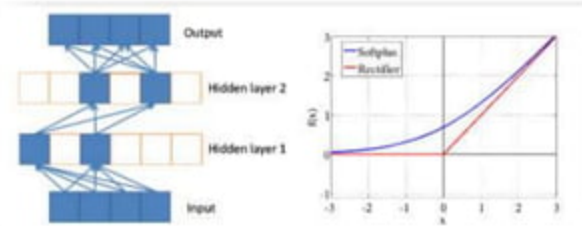


# The Activation Function

## Additional Reading:

*Deep sparse rectifier  
neural networks*

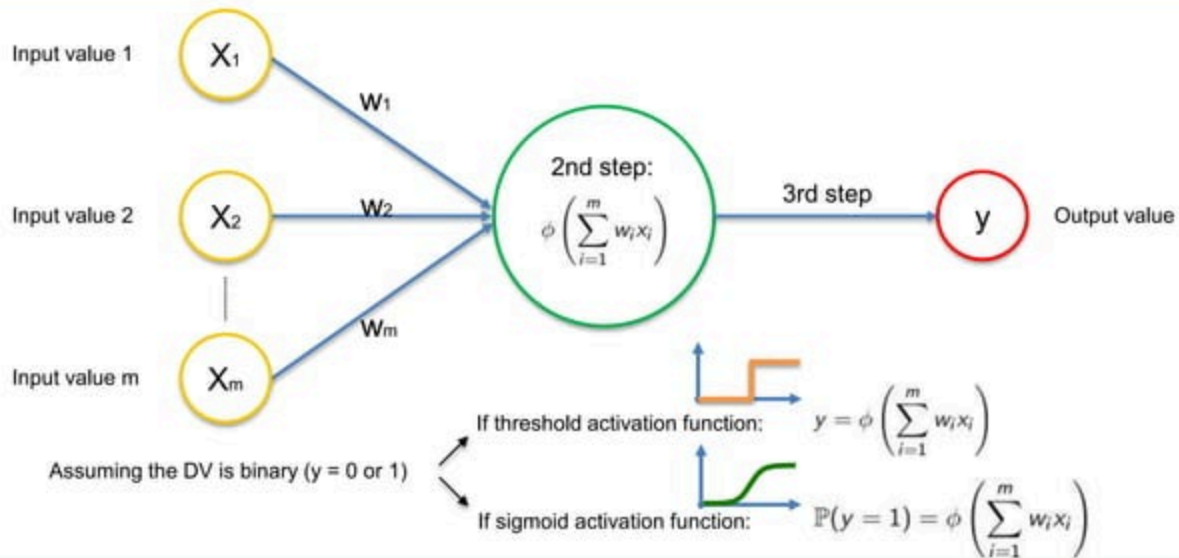
By Xavier Glorot et al. (2011)



Link:

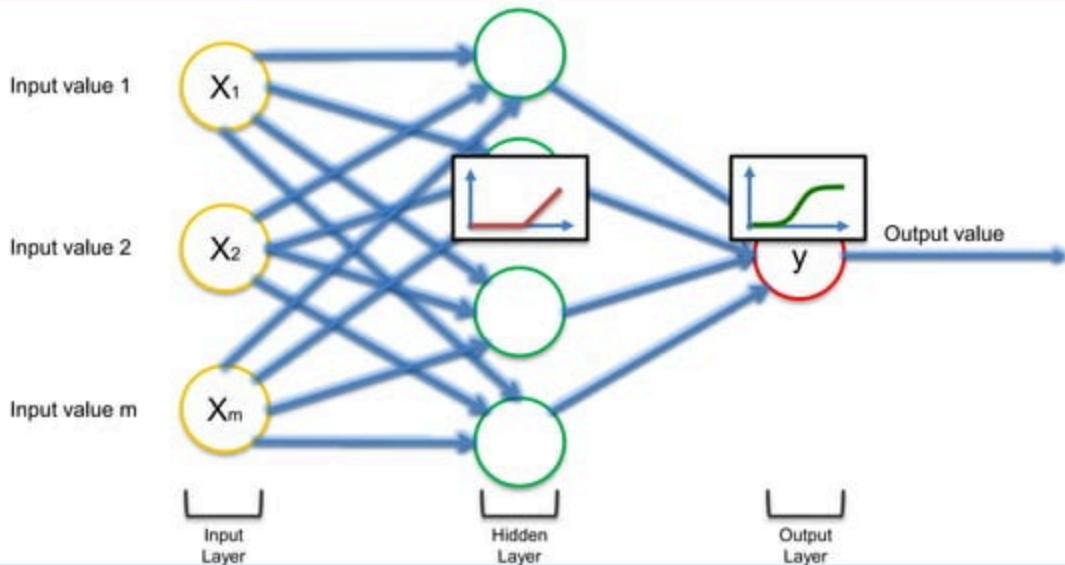
<http://jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf>

# The Activation Function





# The Activation Function



# How do NNs Work?



# How Do Neural Networks Work?



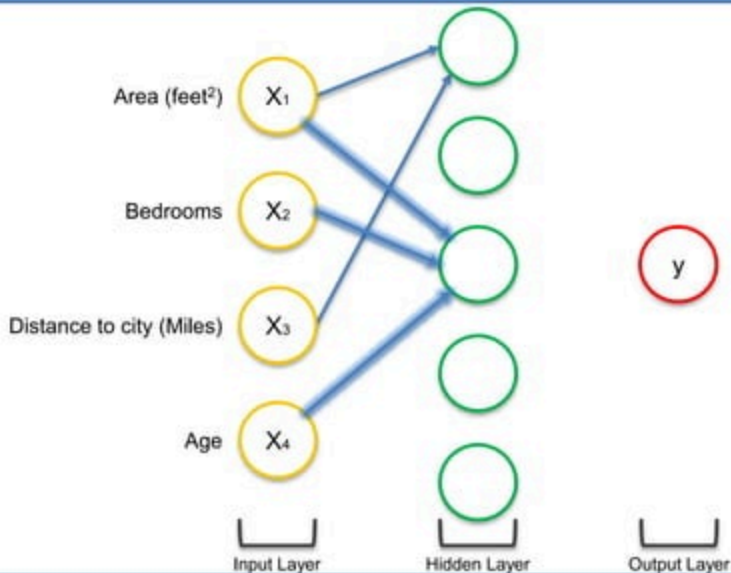
# How Do Neural Networks Work?



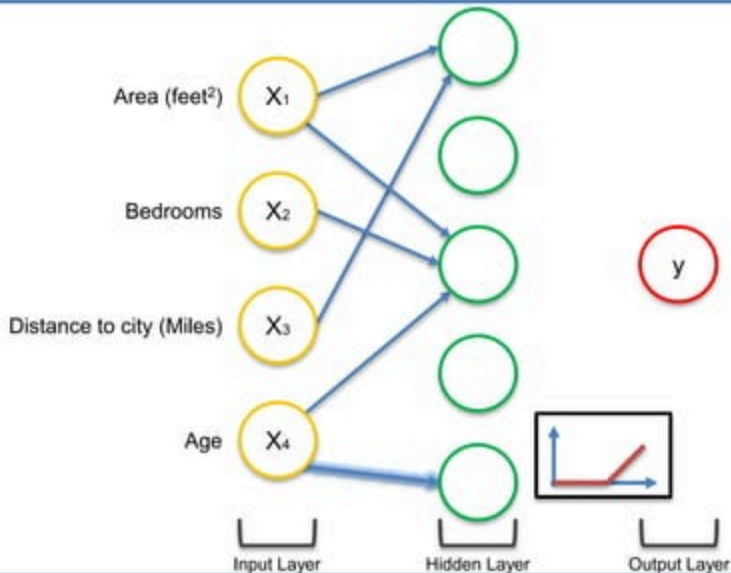
# How Do Neural Networks Work?



# How Do Neural Networks Work?

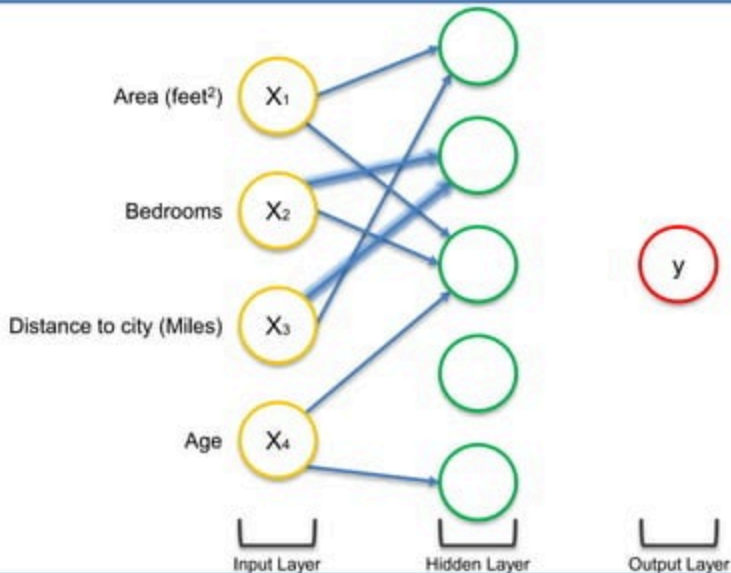


# How Do Neural Networks Work?

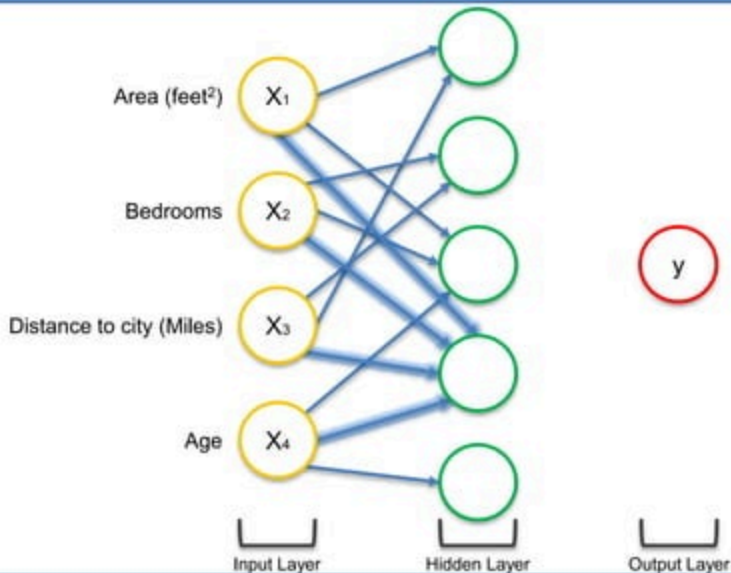




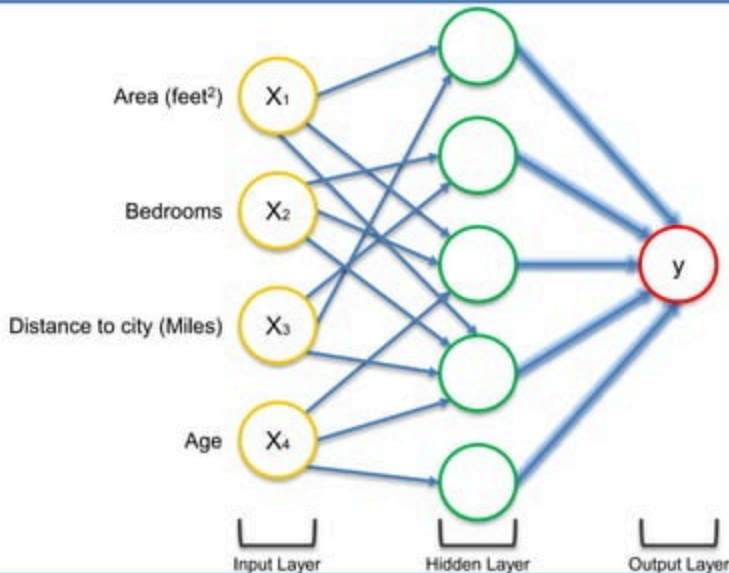
# How Do Neural Networks Work?



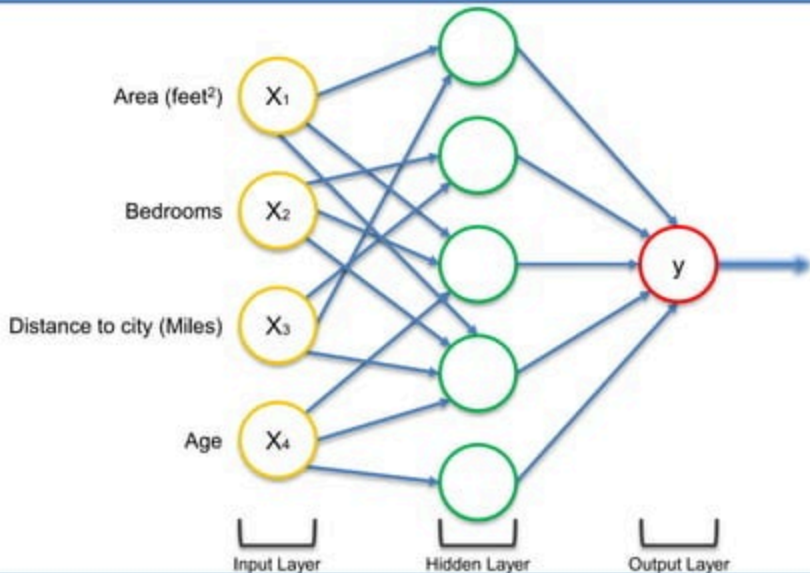
# How Do Neural Networks Work?



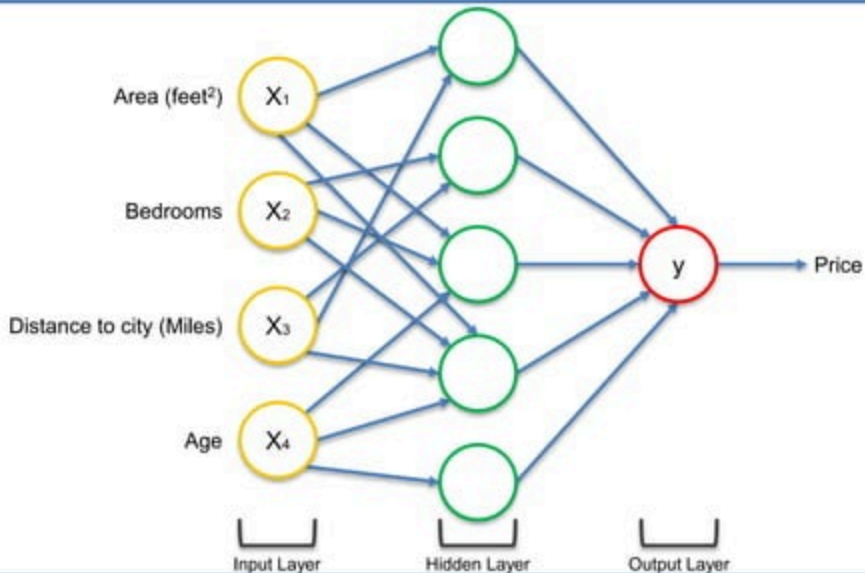
# How Do Neural Networks Work?



# How Do Neural Networks Work?



# How Do Neural Networks Work?

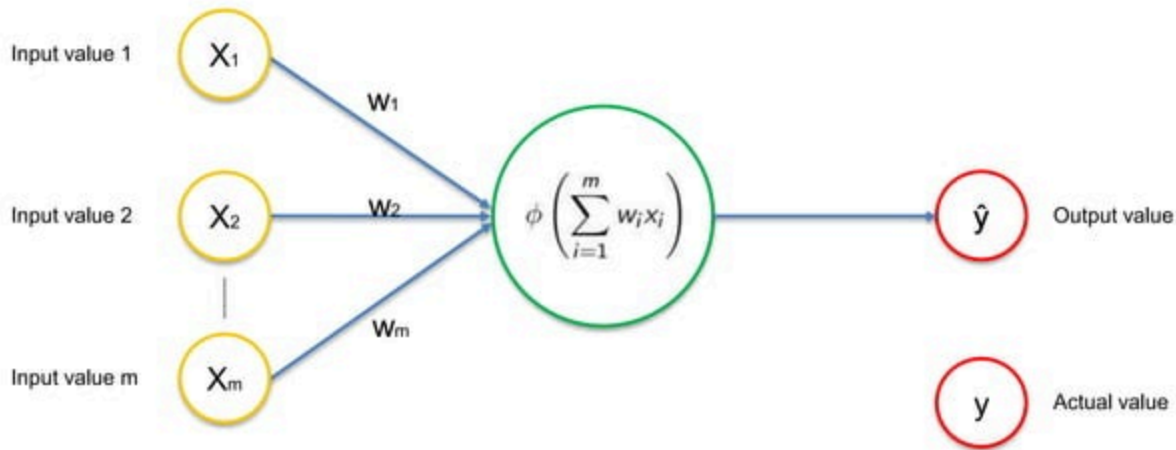


# How do NNs Learn?

# How do Neural Networks learn?

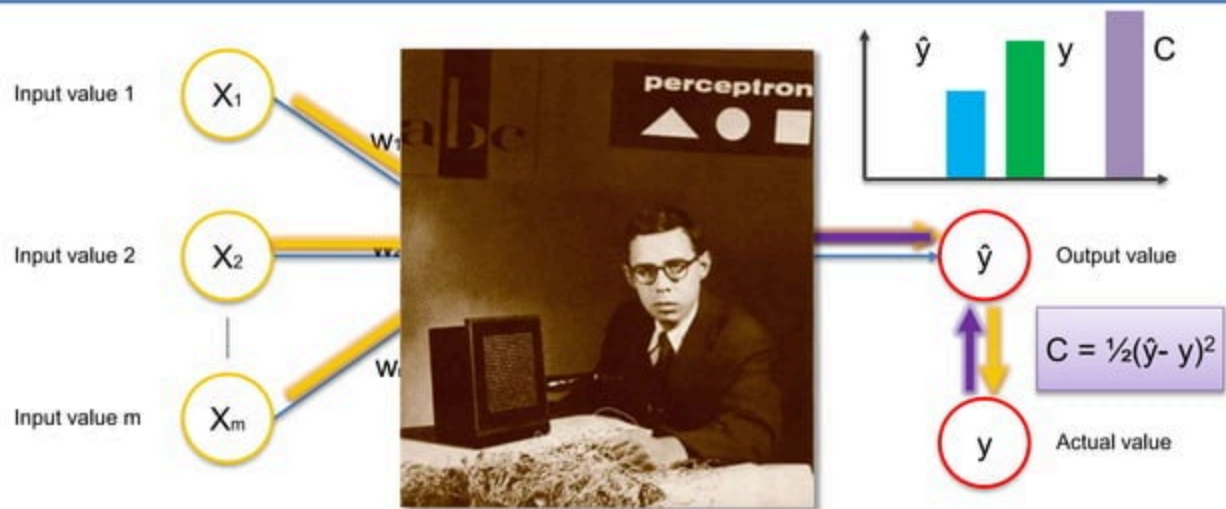


# How do Neural Networks learn?

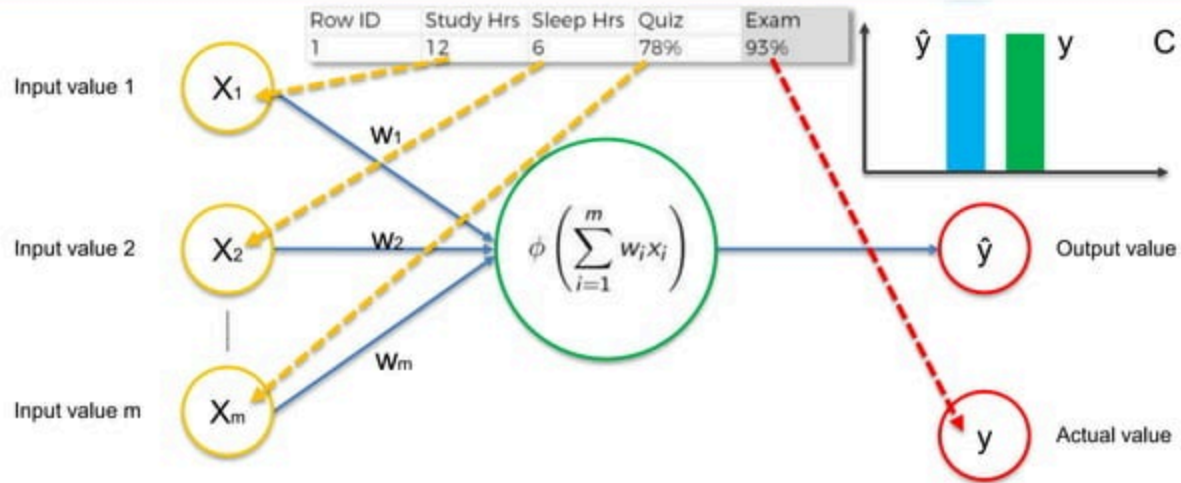




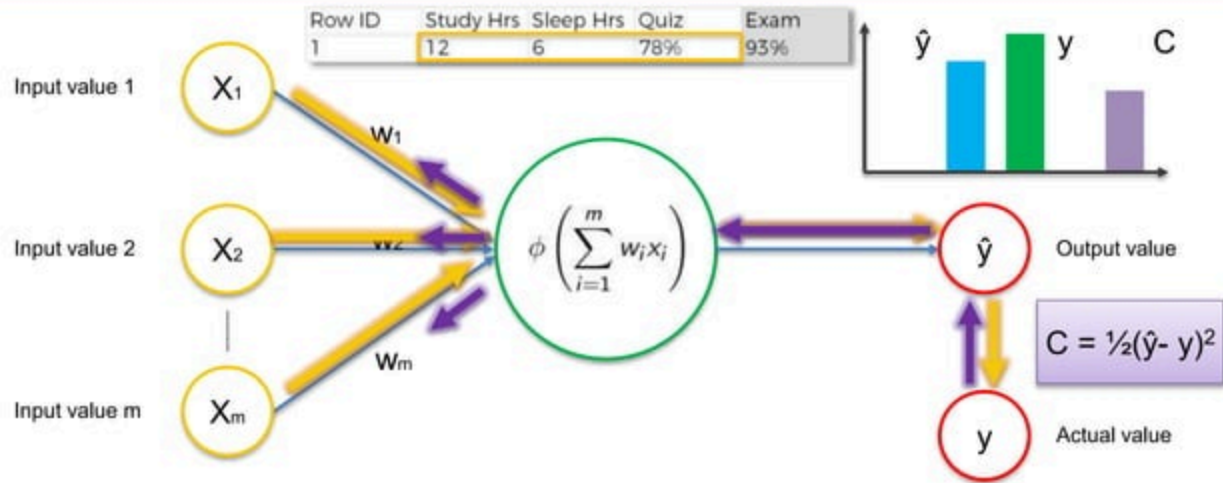
# How do Neural Networks learn?



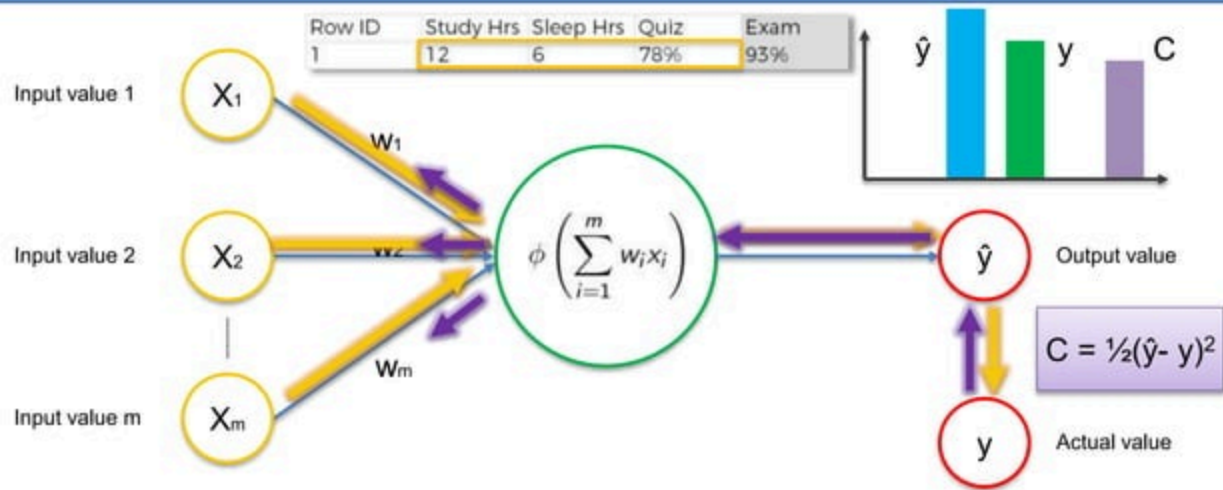
# How do Neural Networks learn?



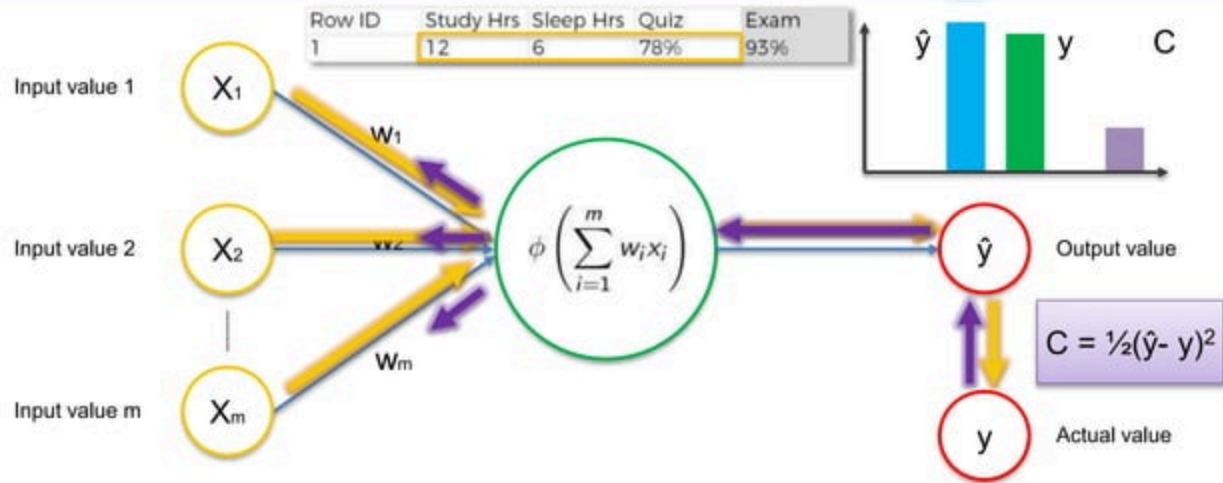
# How do Neural Networks learn?



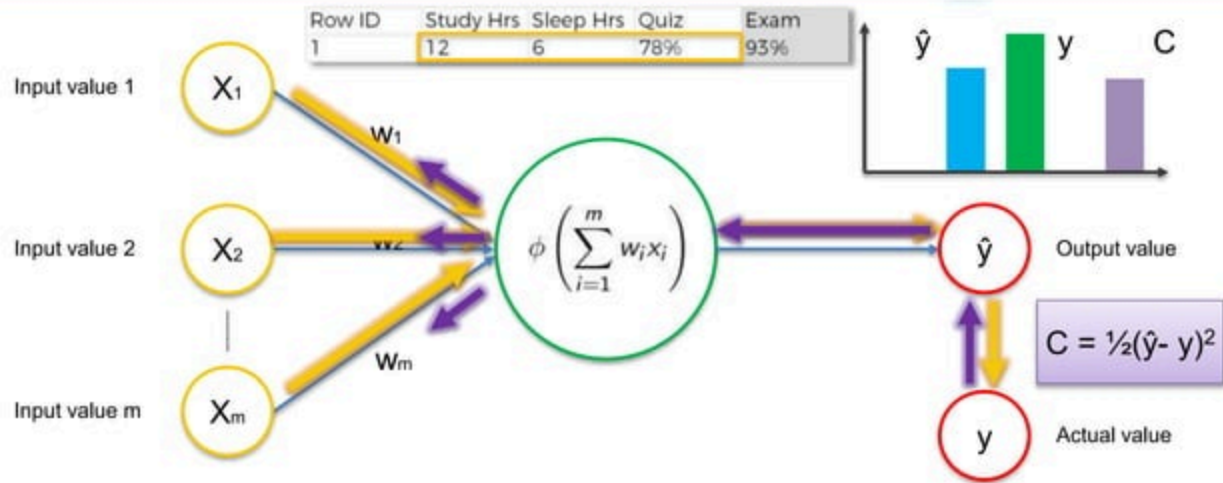
# How do Neural Networks learn?



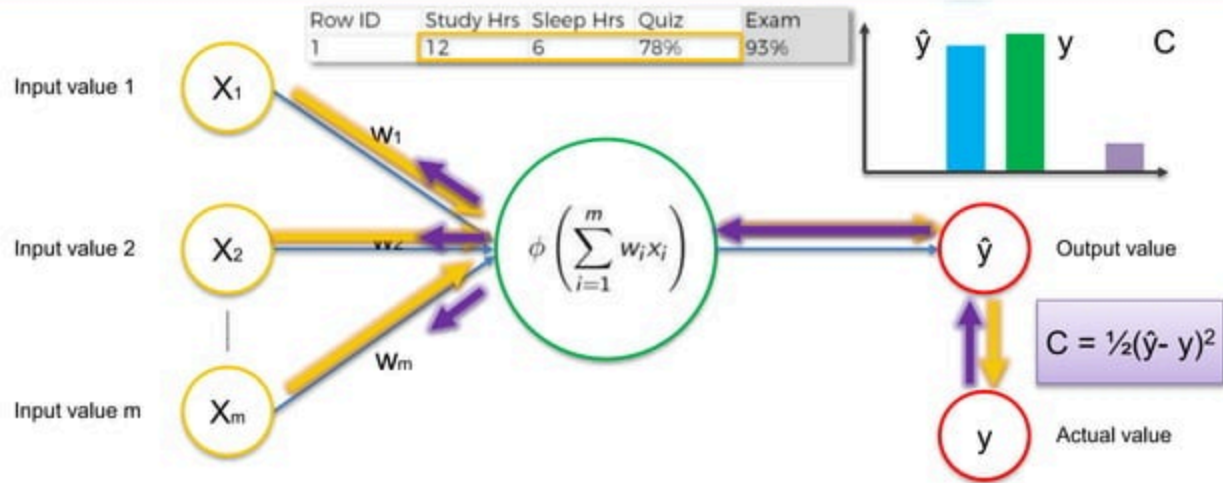
# How do Neural Networks learn?



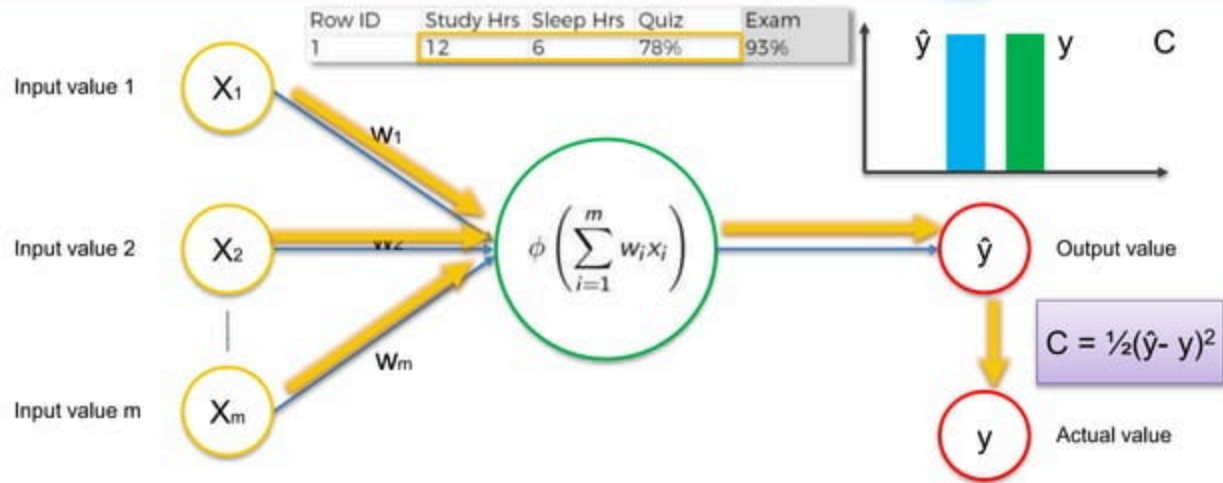
# How do Neural Networks learn?



# How do Neural Networks learn?

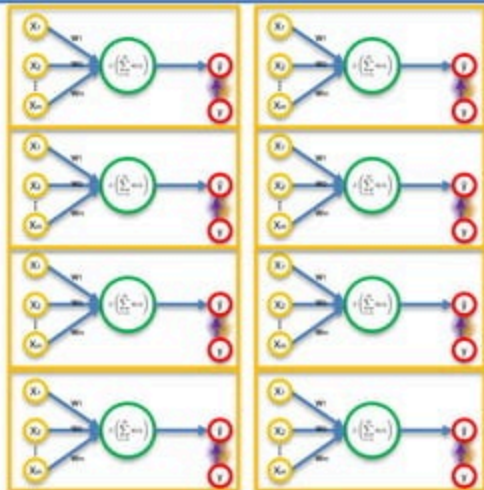


# How do Neural Networks learn?



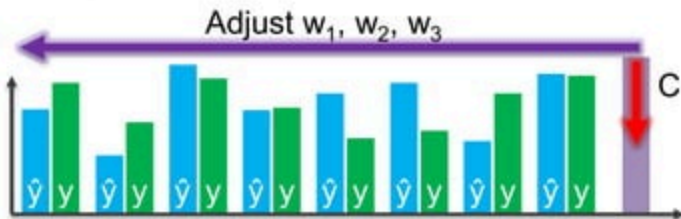


# How do Neural Networks learn?



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%

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



# How do Neural Networks learn?

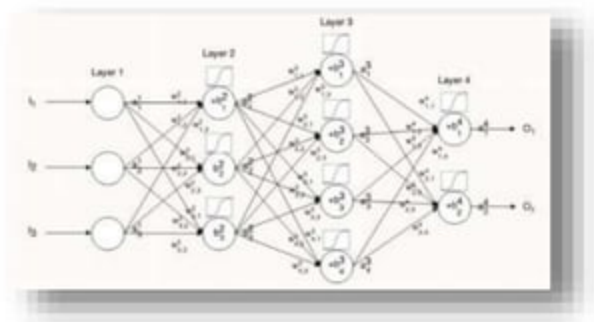
## Additional Reading:

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

CrossValidated (2015)

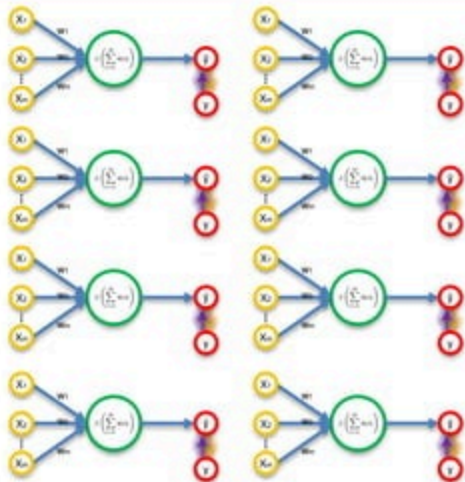
Link:

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



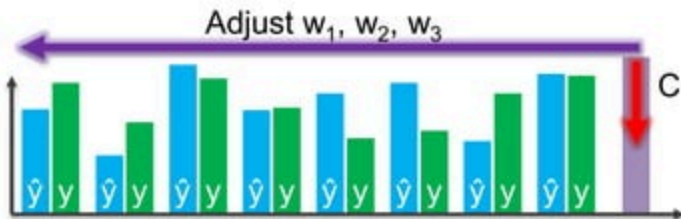
# Gradient Descent

# Gradient Descent

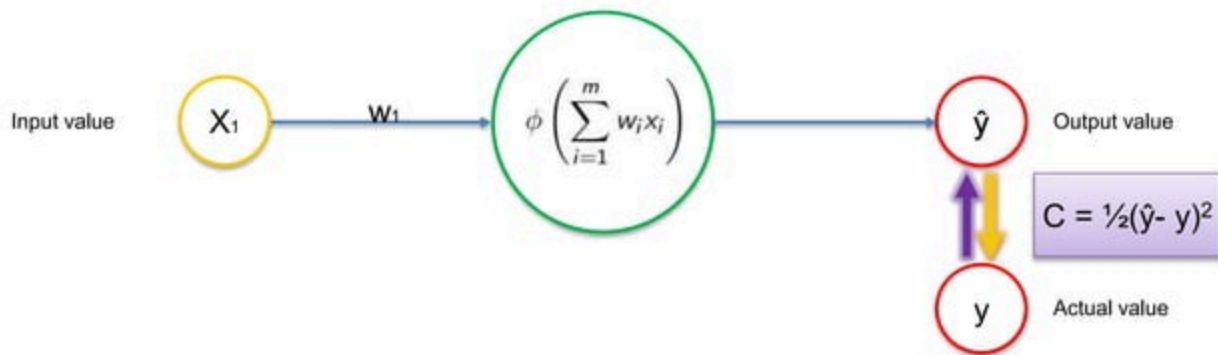


Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

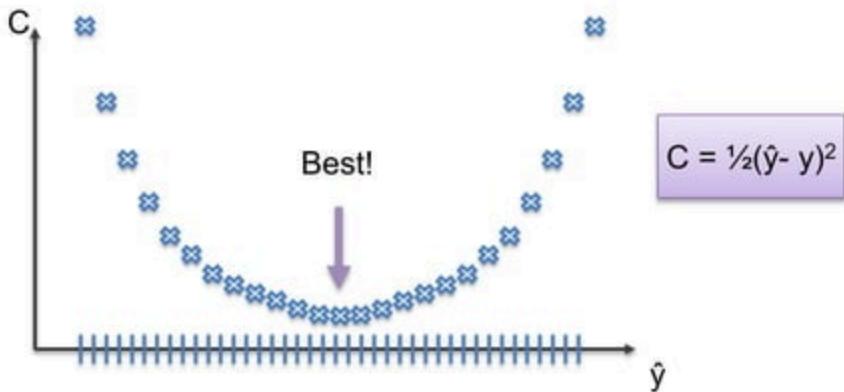
$$C = \sum \frac{1}{2}(\hat{y} - y)^2$$



# Gradient Descent

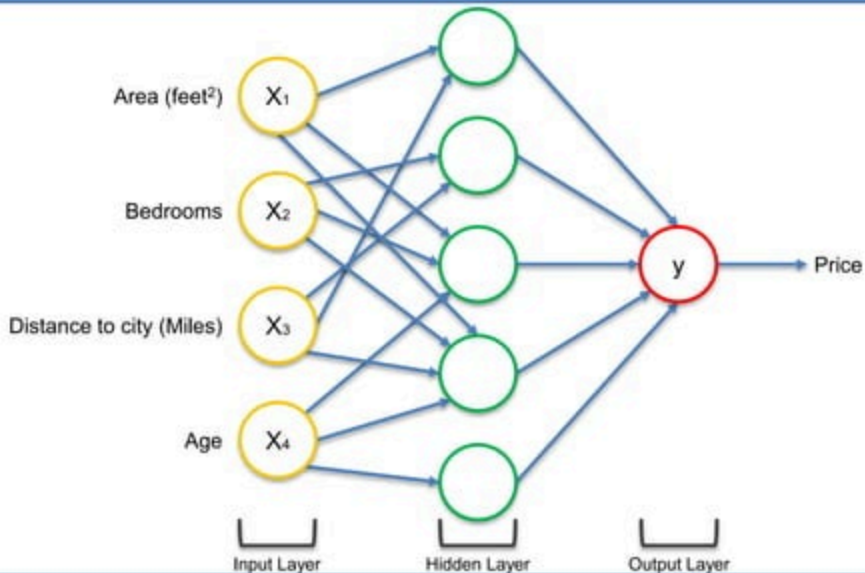


# Gradient Descent



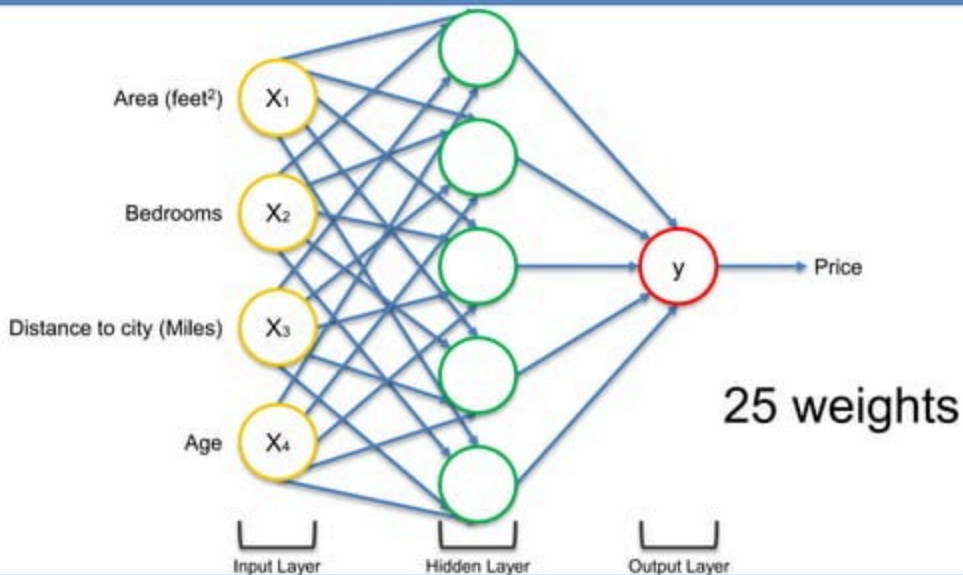
## Curse of Dimensionality

# Gradient Descent





# Gradient Descent



# Gradient Descent

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

Sunway TaihuLight: World's fastest Super Computer

93 PFLOPS

$93 \times 10^{15}$

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

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

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



# Gradient Descent

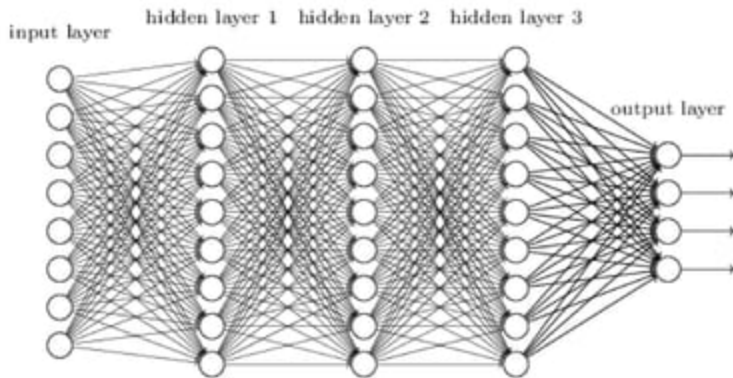


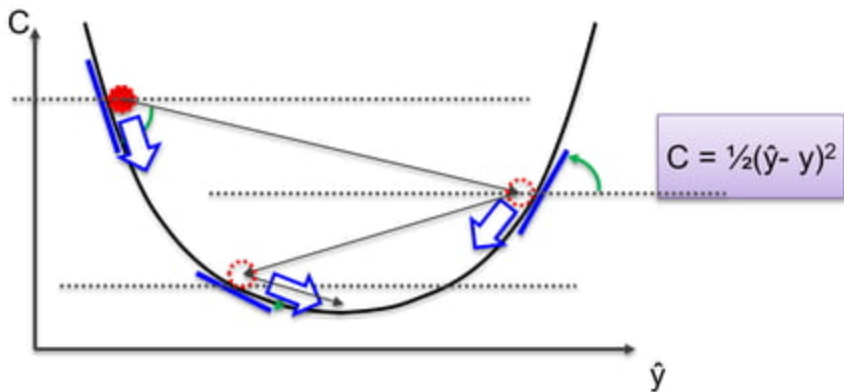
Image Source: [neuralnetworksanddeeplearning.com](http://neuralnetworksanddeeplearning.com)

# Gradient Descent

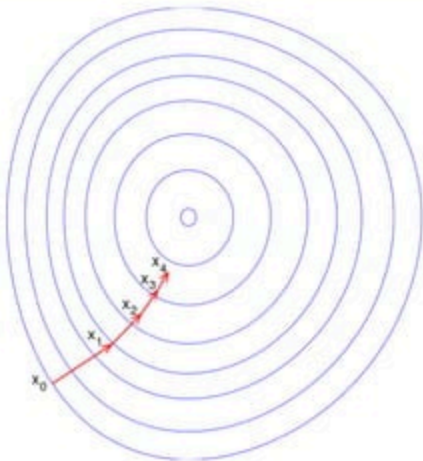
---

Gradient Descent

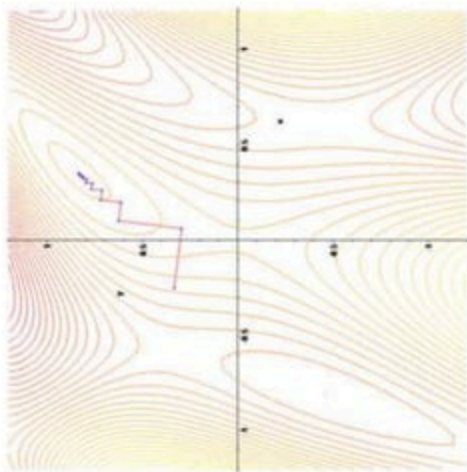
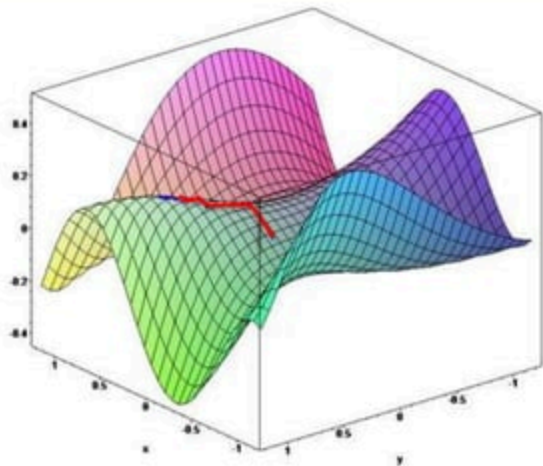
# Gradient Descent



# Gradient Descent



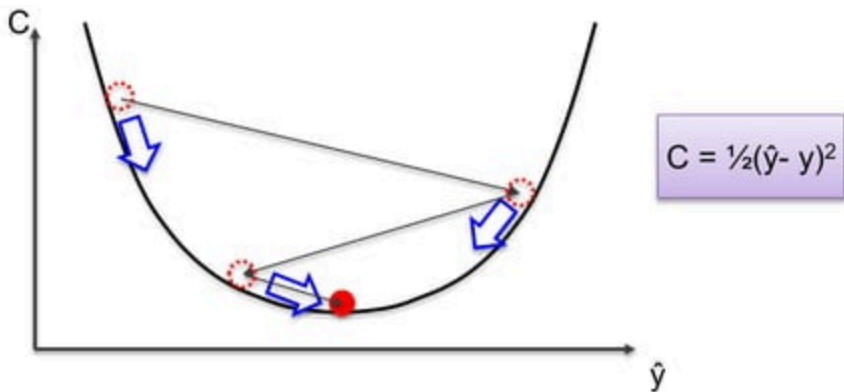
# Gradient Descent



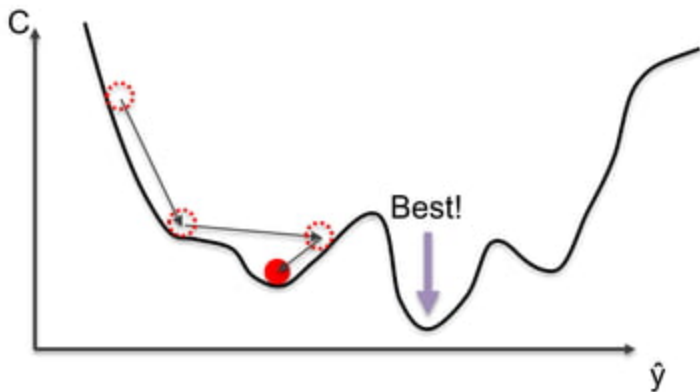
# Stochastic Gradient Descent



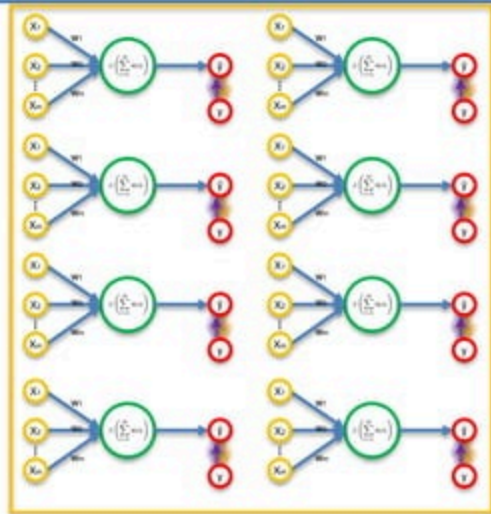
# Stochastic Gradient Descent



# Stochastic Gradient Descent



# Stochastic Gradient Descent

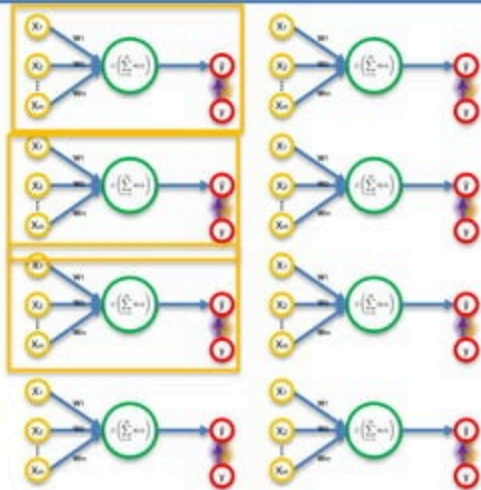


Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

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

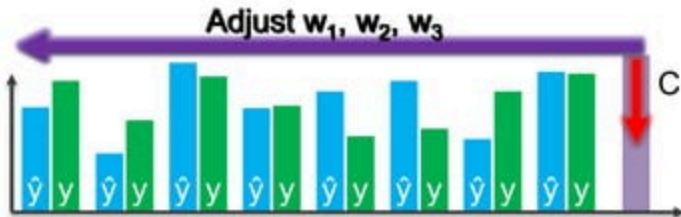


# Stochastic Gradient Descent

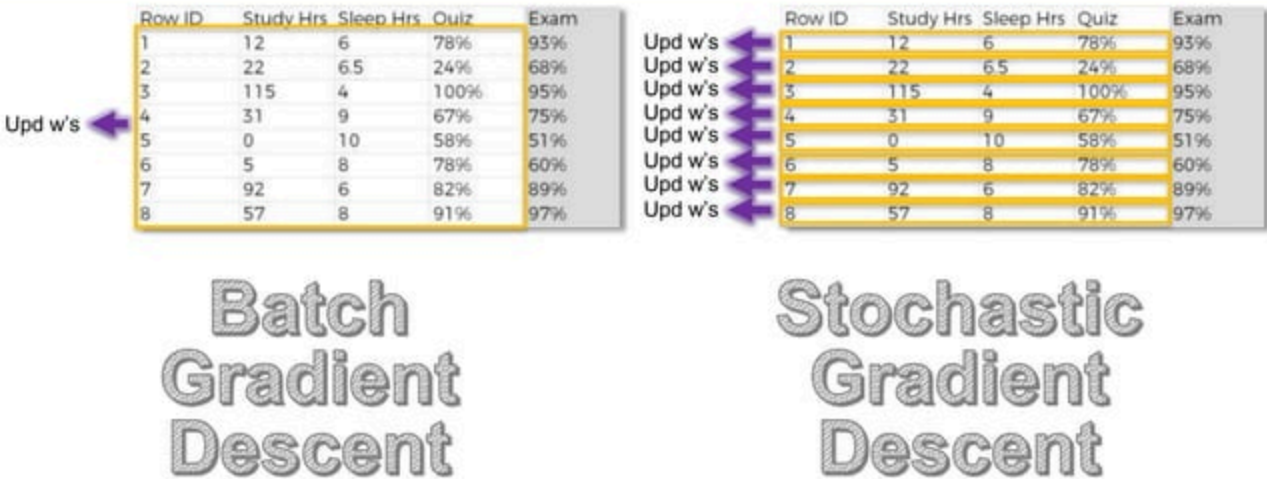


Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

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



# Stochastic Gradient Descent



# Stochastic Gradient Descent

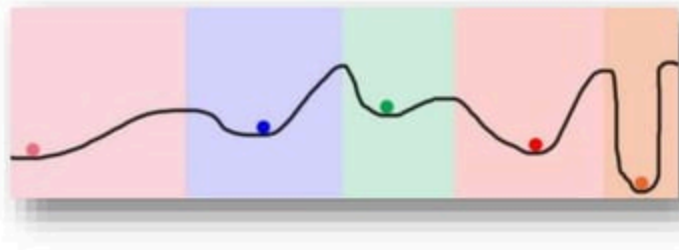
## Additional Reading:

*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

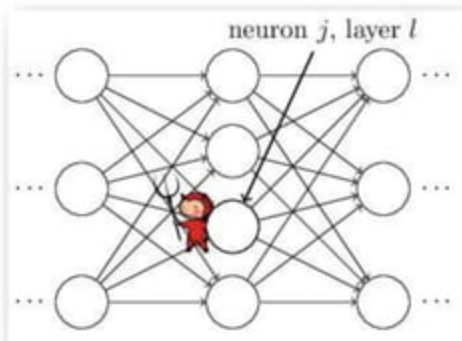
## Additional Reading:

*Neural Networks and Deep Learning*

Michael Nielsen (2015)

Link:

<http://neuralnetworksanddeeplearning.com/chap2.html>



# Backpropagation



# Gradient Descent

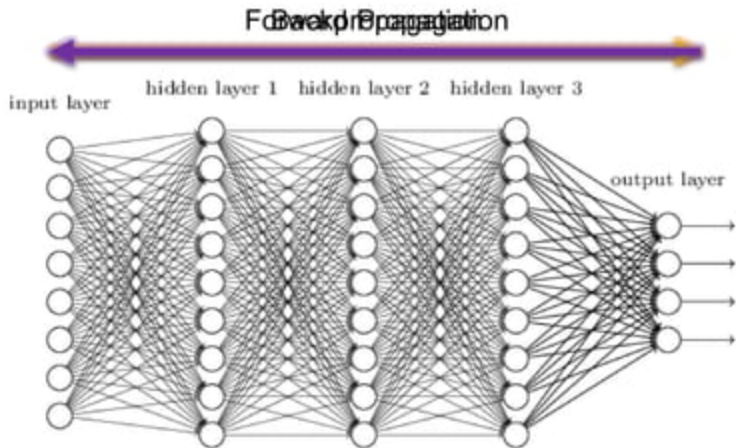


Image Source: [neuralnetworksanddeeplearning.com](http://neuralnetworksanddeeplearning.com)

# Stochastic Gradient Descent

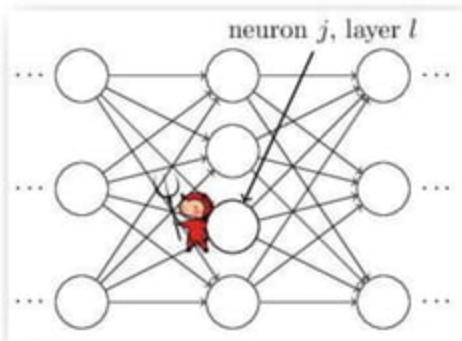
## Additional Reading:

*Neural Networks and Deep Learning*

Michael Nielsen (2015)

Link:

<http://neuralnetworksanddeeplearning.com/chap2.html>



# 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.