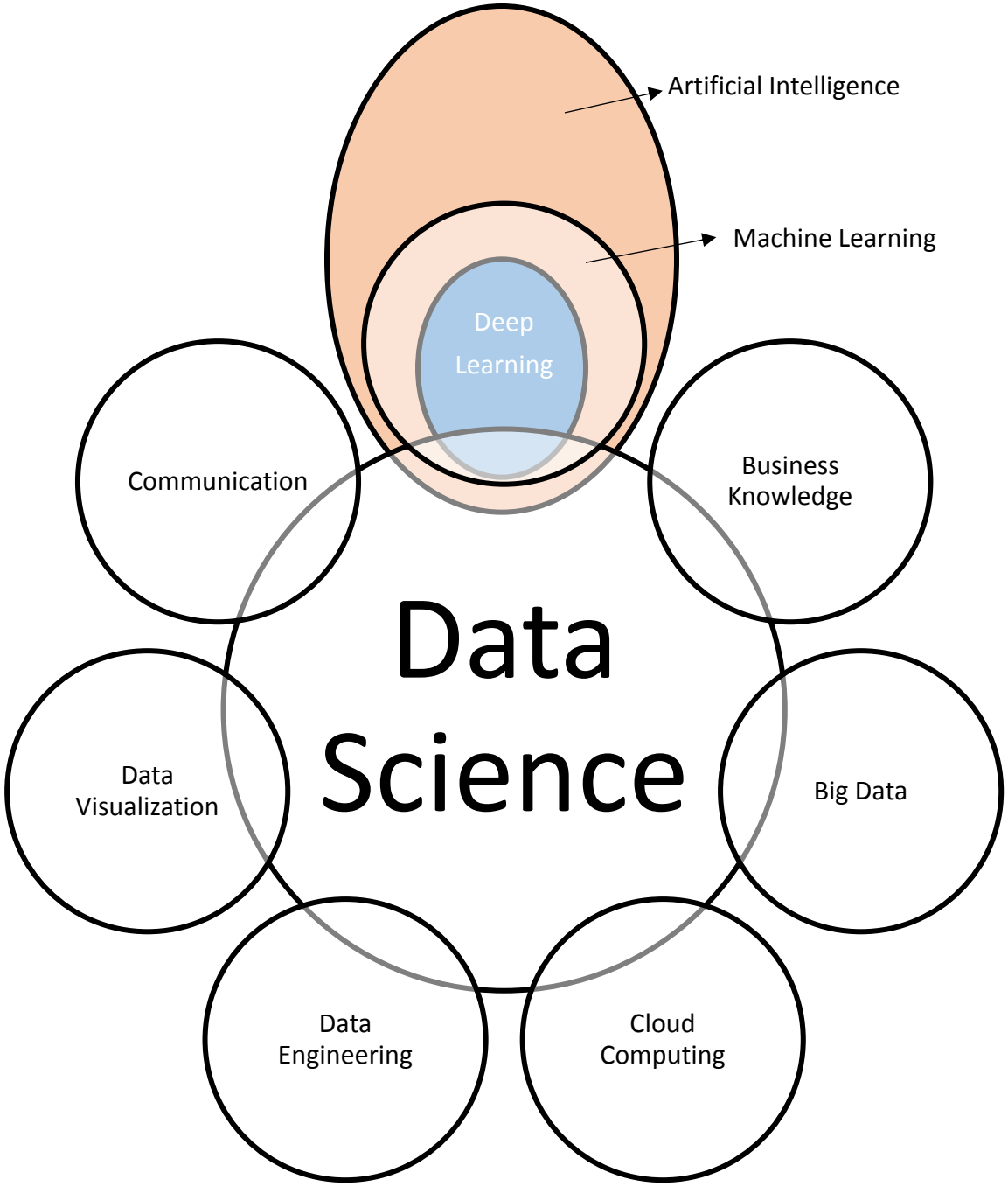
Dipping your toes into Deep Learning!

Devashish Khatwani

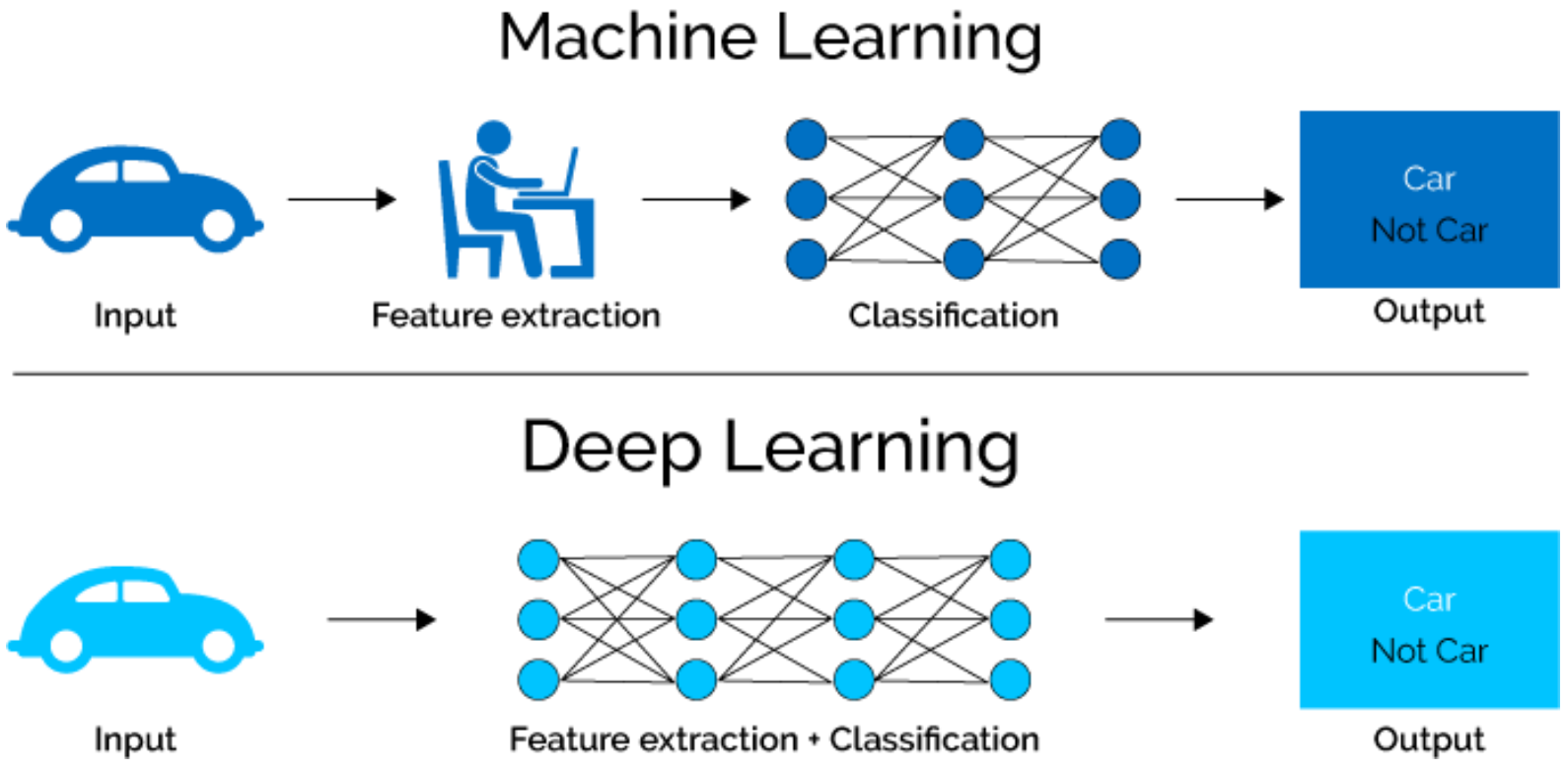
October 2017

Data Science, Machine Learning, Deep Learning, Artificial Intelligence??

I AM CONFUSED!

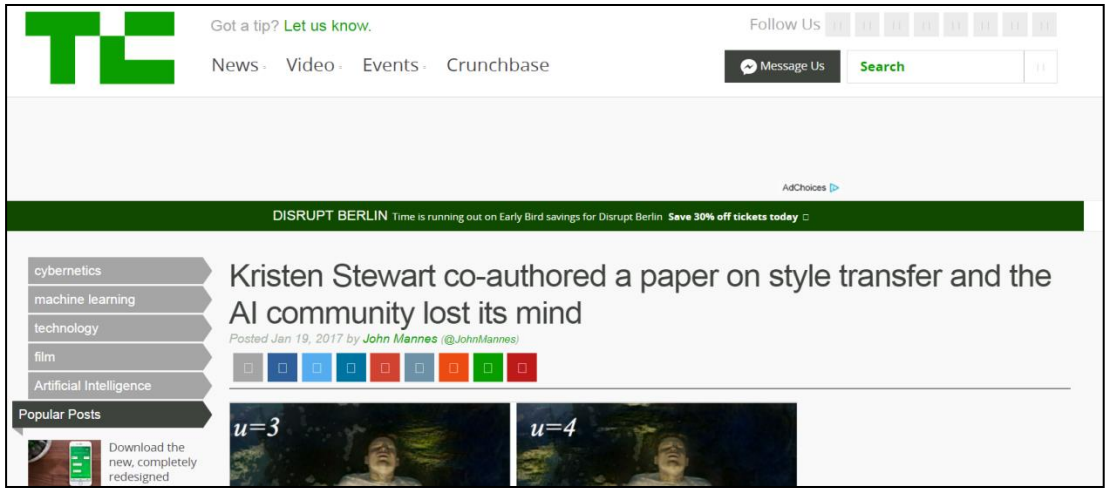


1. Deep Learning is a subfield of Machine Learning
2. Machine Learning itself is a subfield of Artificial Intelligence
3. Data Science is a field which touches many subfields and is a more application oriented domain

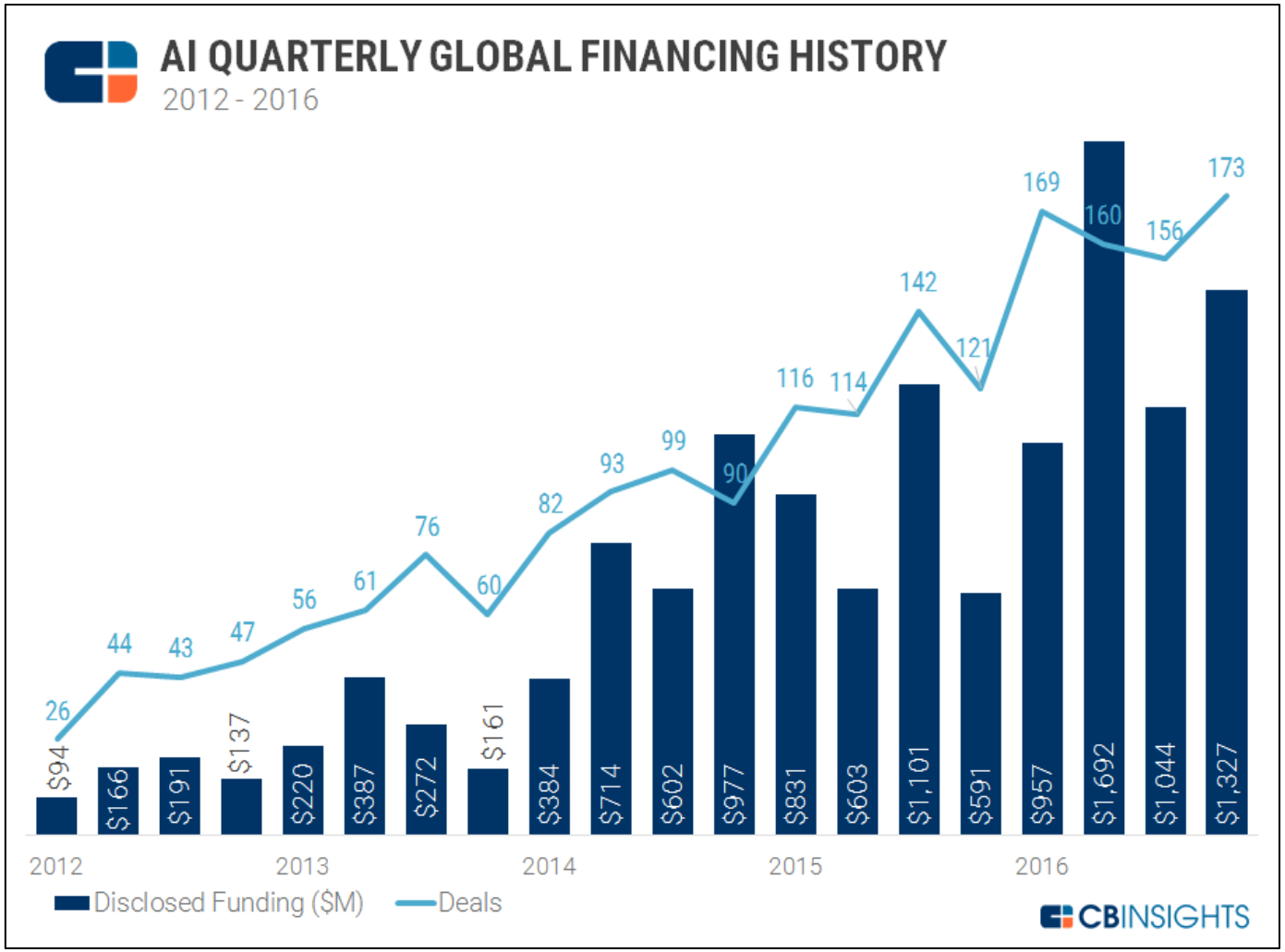


1. No need of feature engineering
2. Bias Variance Tradeoff no longer applicable
3. Can handle massive amounts of data
4. Change in Train, Validation, Test dataset split mindset

Hardly a days goes by without a media mention

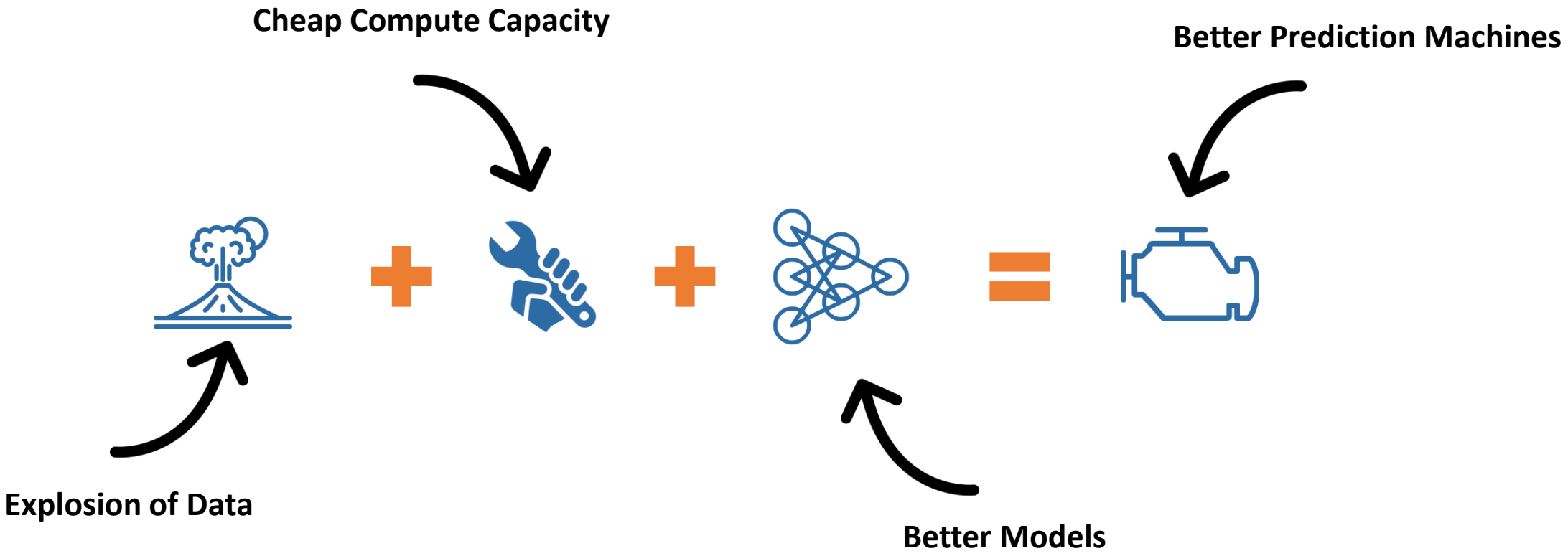


More Money flowing into AI based startups than ever before

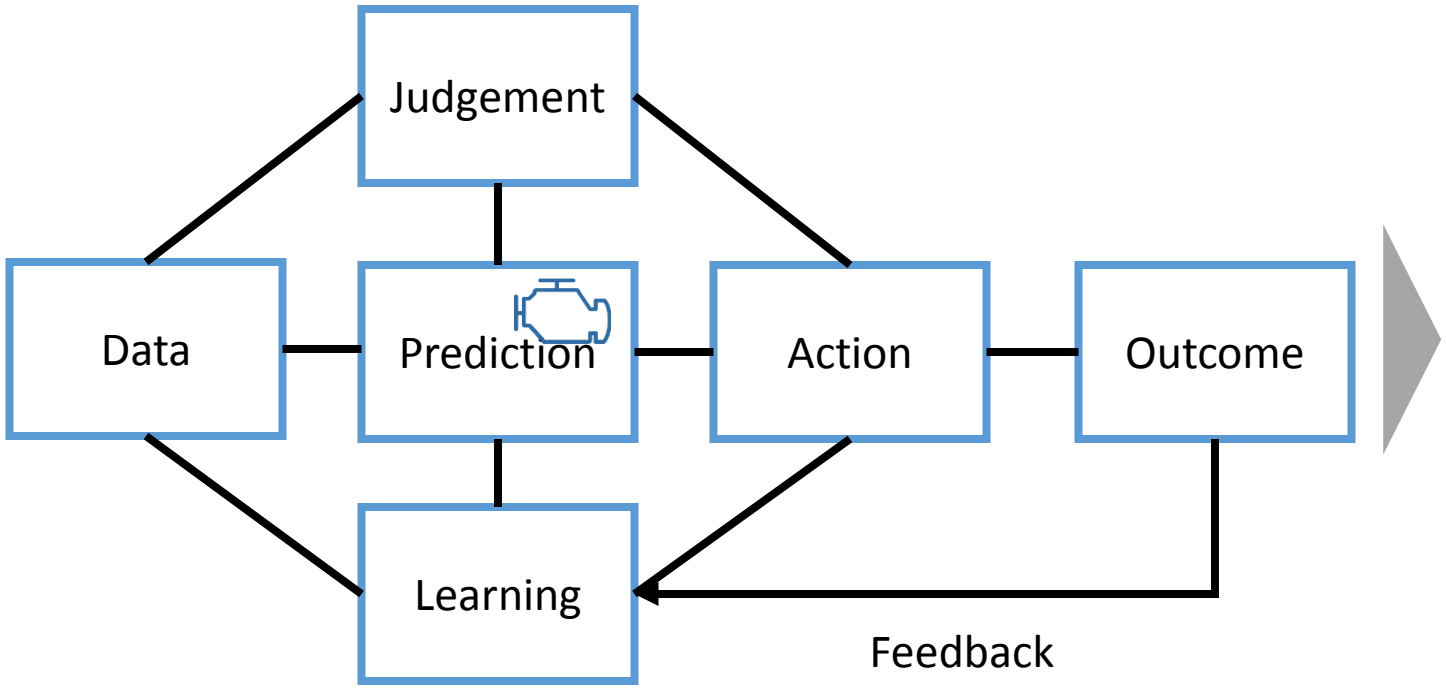


Why the hype, Bro?!

Why Now?

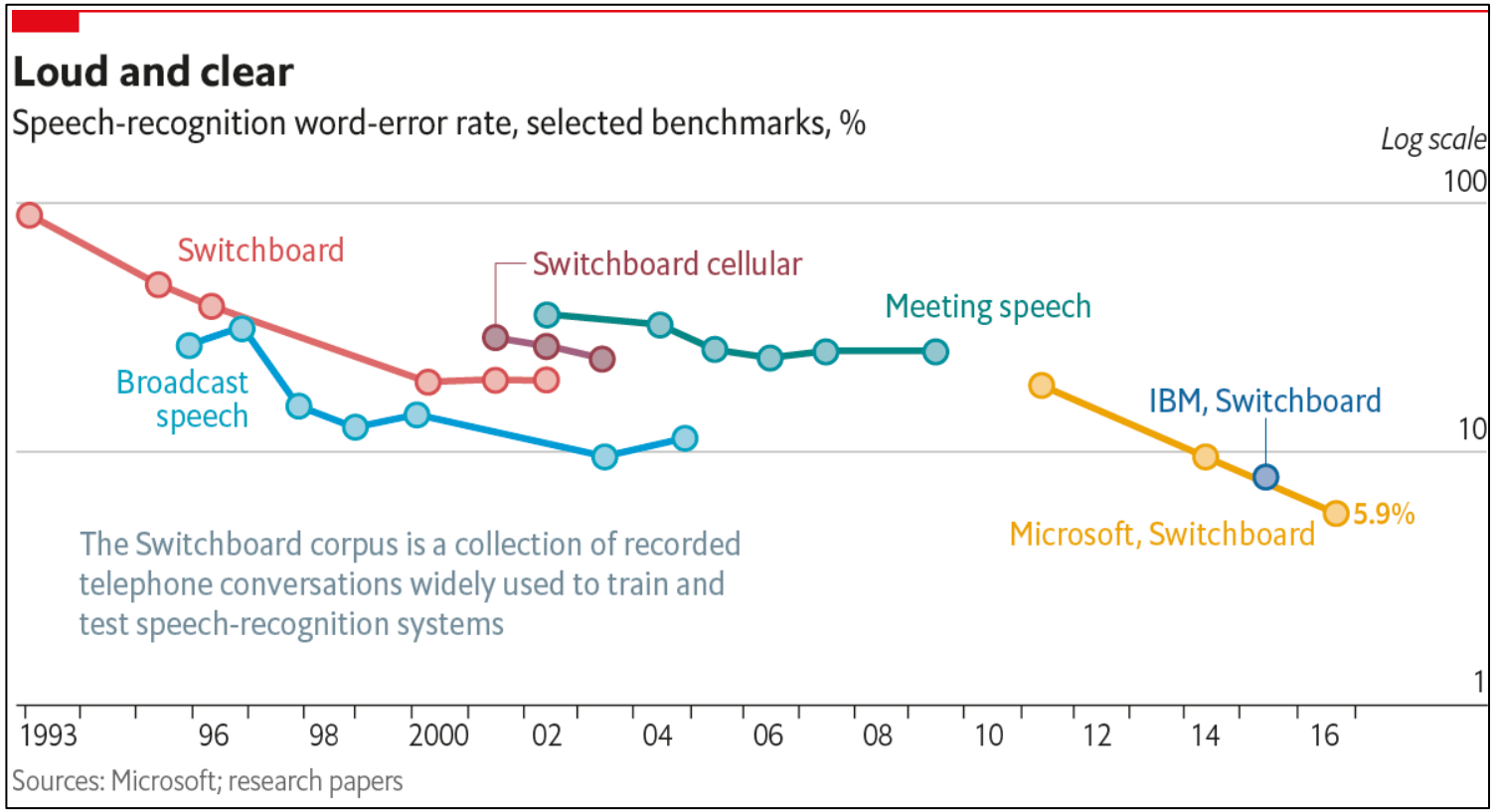

Cheap Compute Capacity
Better Prediction Machines
Explosion of Data
Better Models

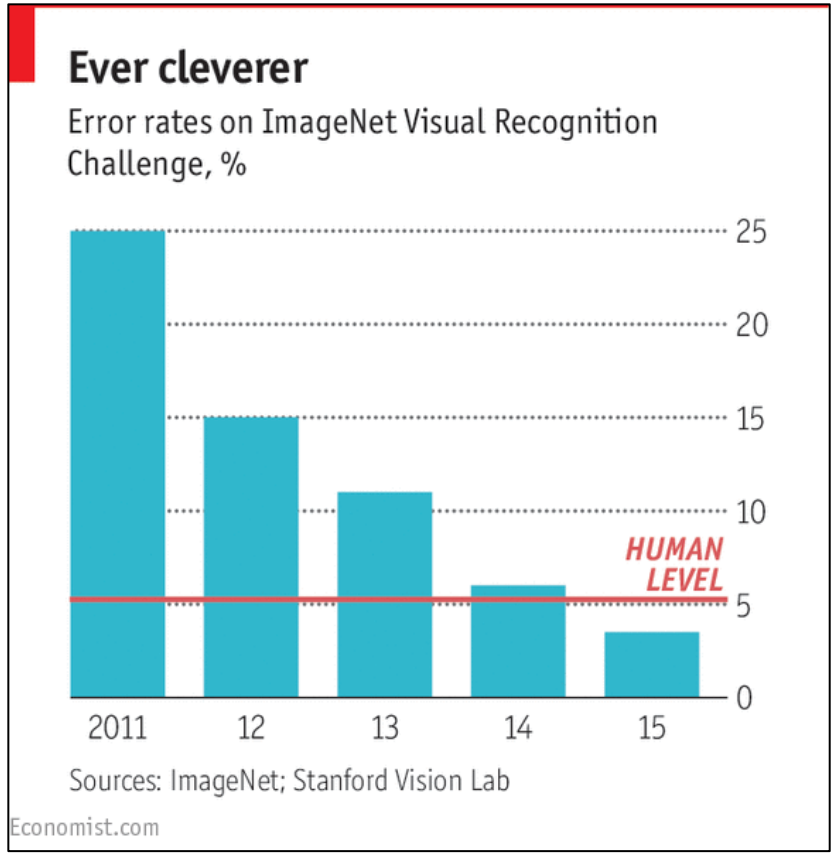
We have a better prediction machine, so what?



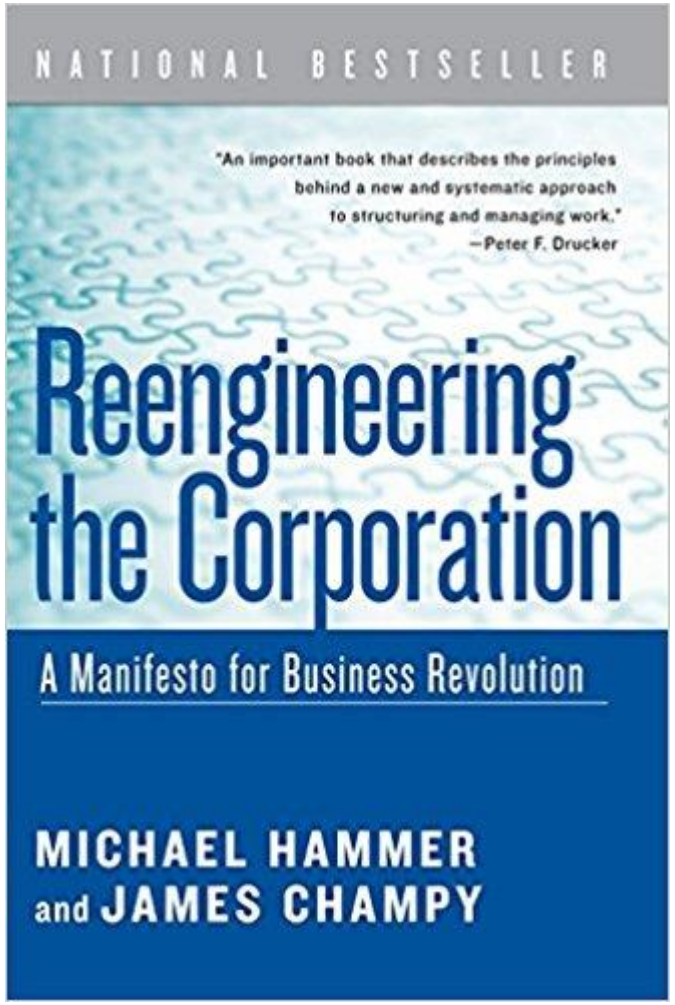
1. As the cost of a resource decreases we use more of it
2. The value of compliments goes up
3. The value of substitutes goes down
4. We start using the resource in places where we never used it before

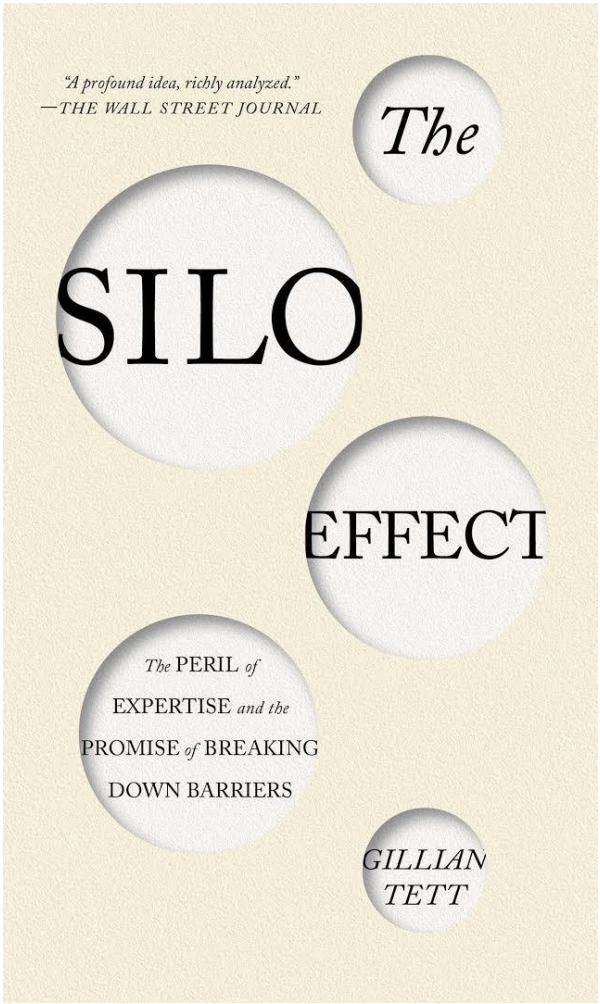
Deep Learning Model have beaten Humans at Audio and Image recognition tasks in last 3 years

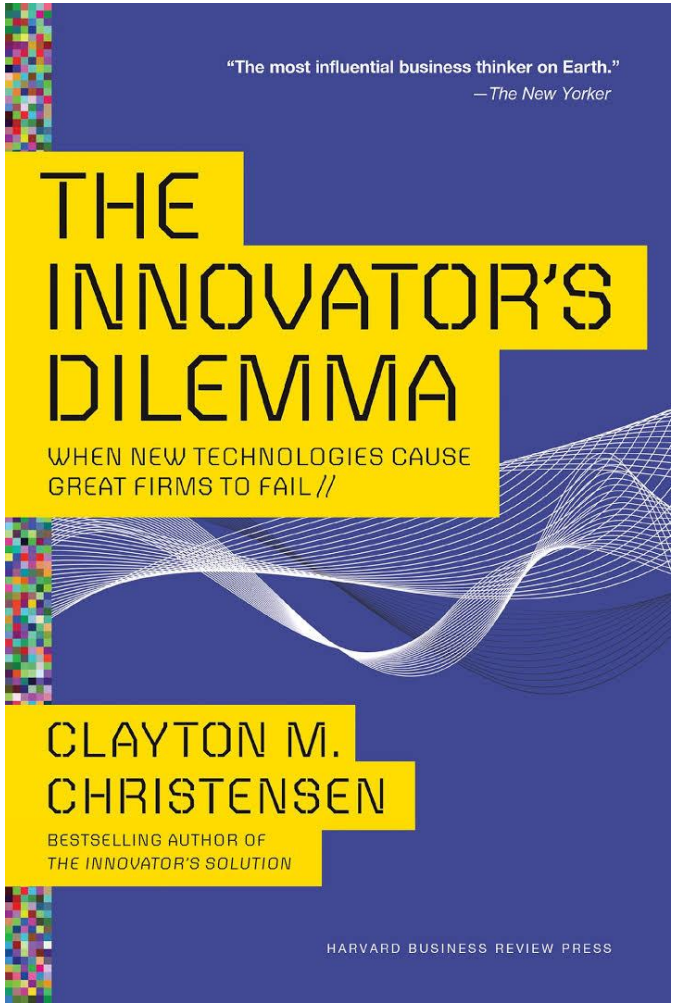




Tom's Christmas Reading List



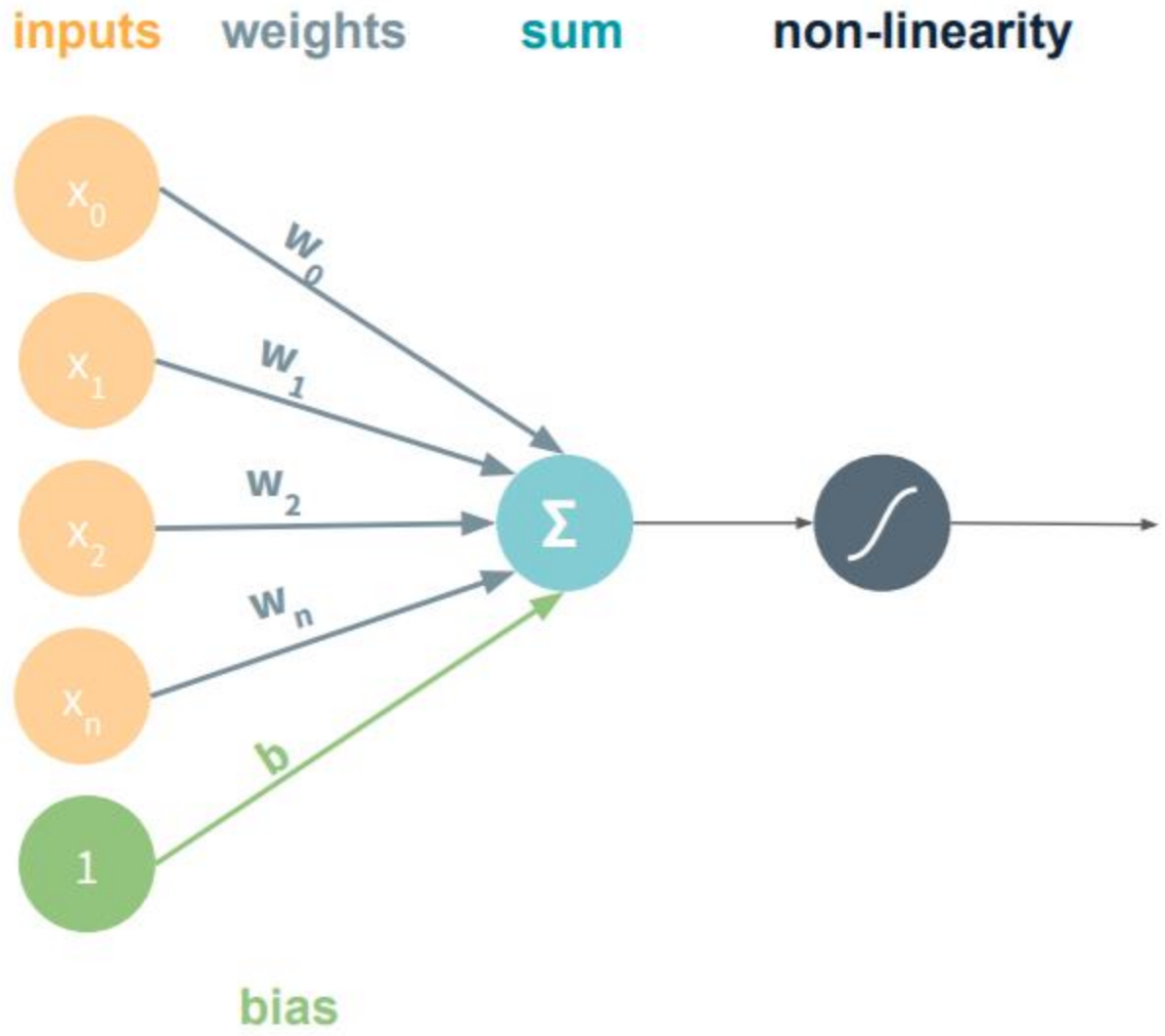




Lets get to the Meat and Potatoes now!

First the Potatoes(Pretty Pictures).

The Perceptron

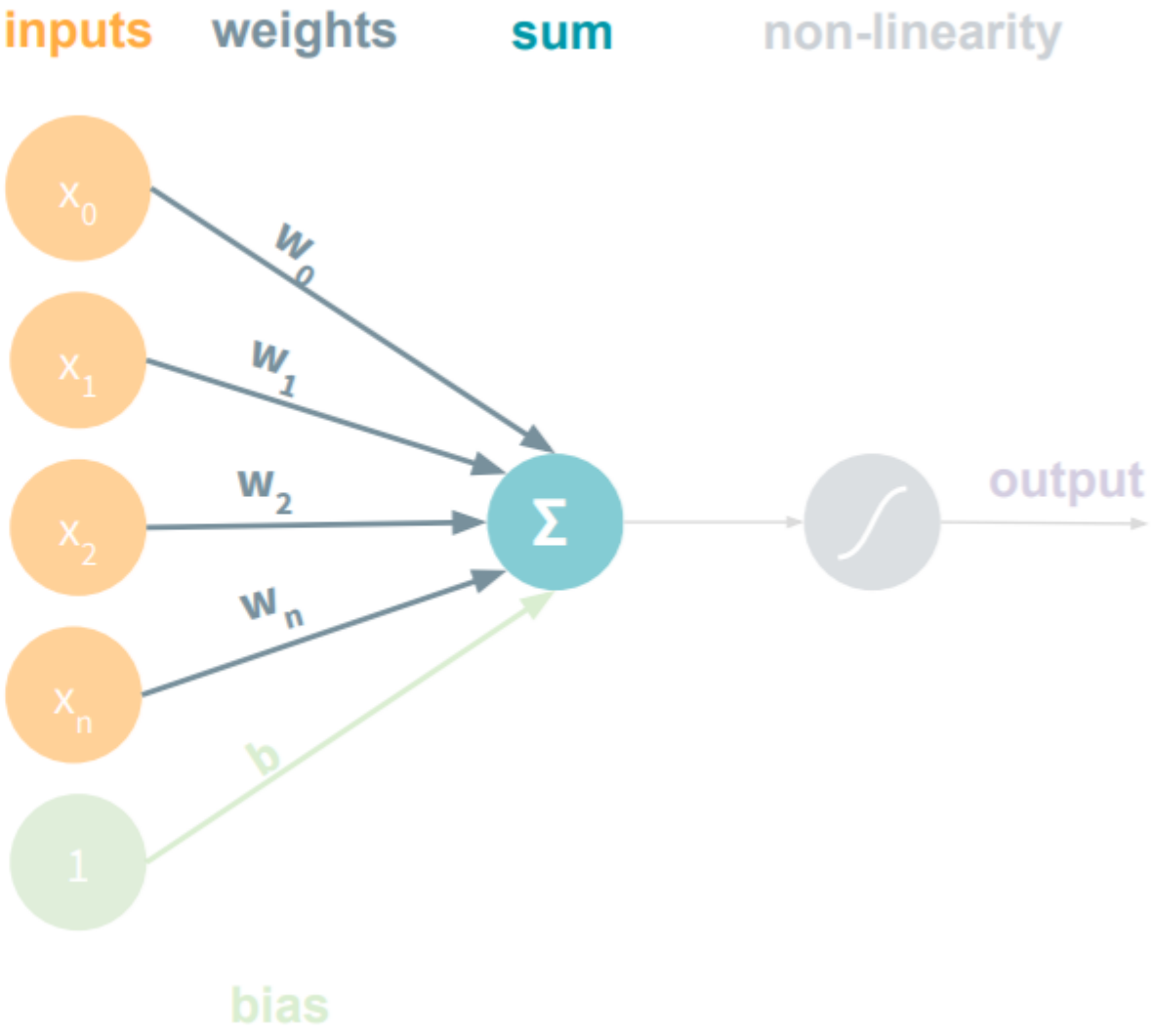


The Perceptron: The Forward Pass

$$output =$$

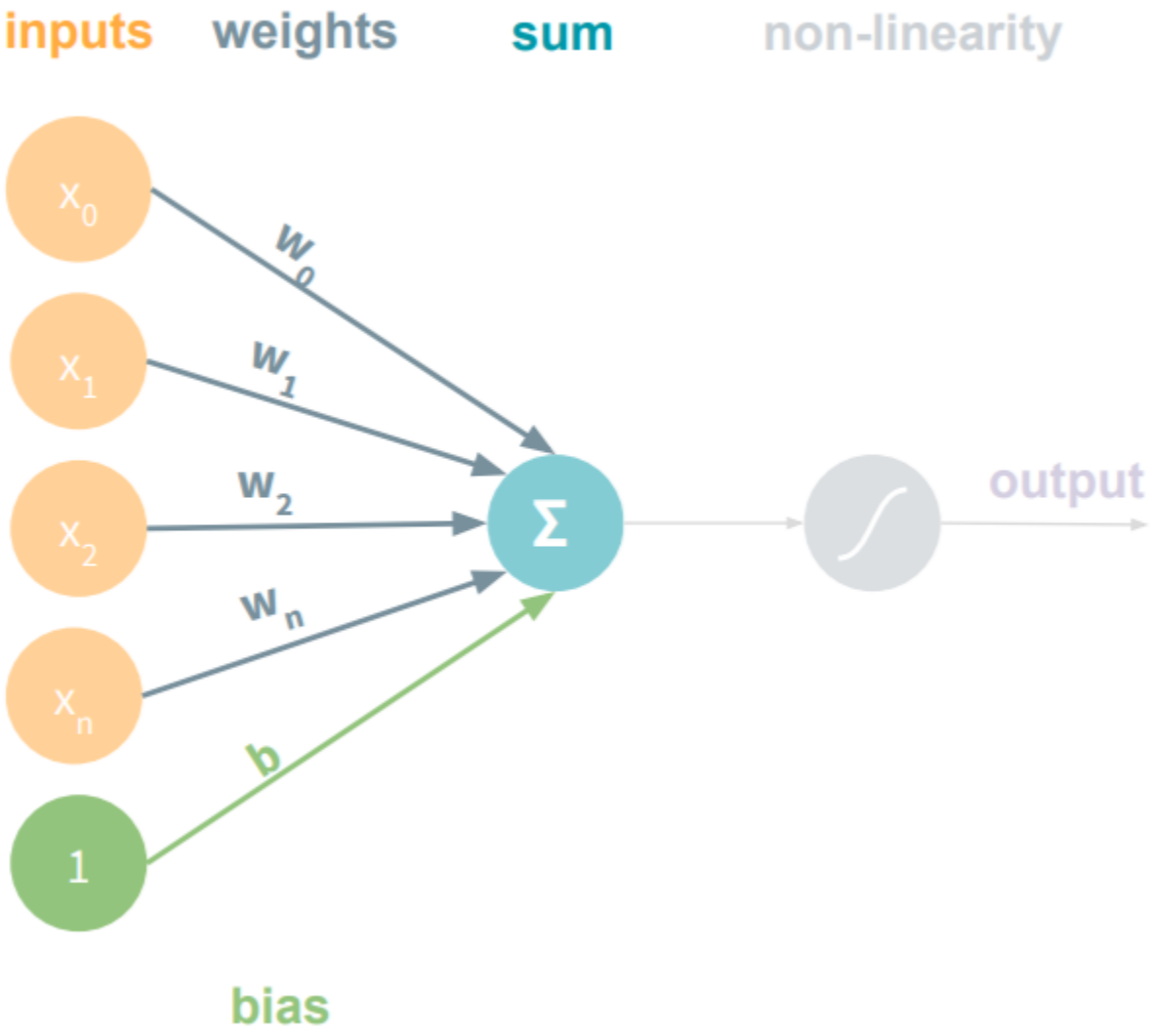
The Perceptron: The Forward Pass

$$output = \sum_{i=0}^{N} x_i * w_i$$



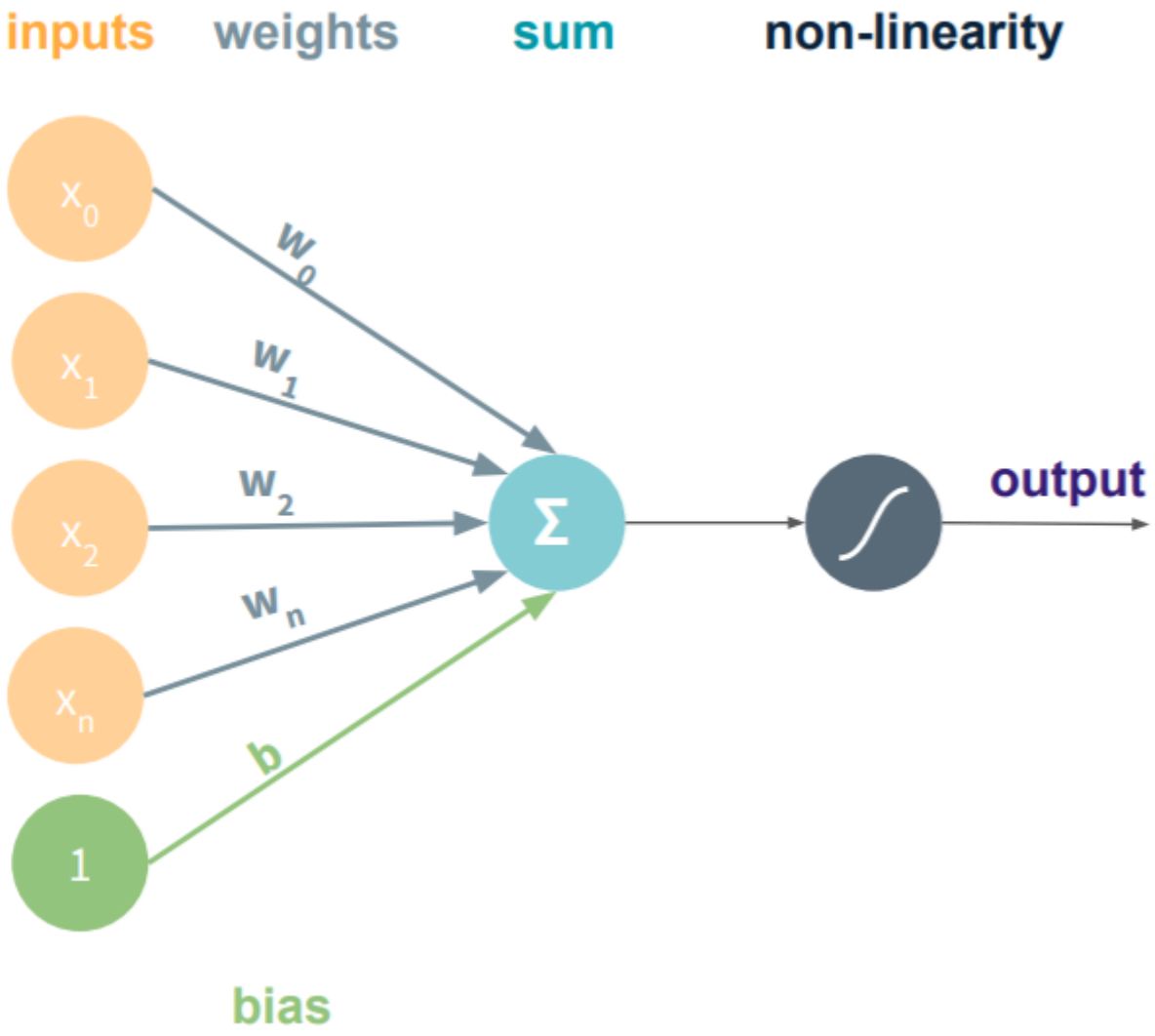
The Perceptron: The Forward Pass

$$output = \sum_{i=0}^{N} (x_i * w_i) + b$$



The Perceptron: The Forward Pass

$$output = g((\sum_{i=0}^{N} x_i * w_i) + b)$$

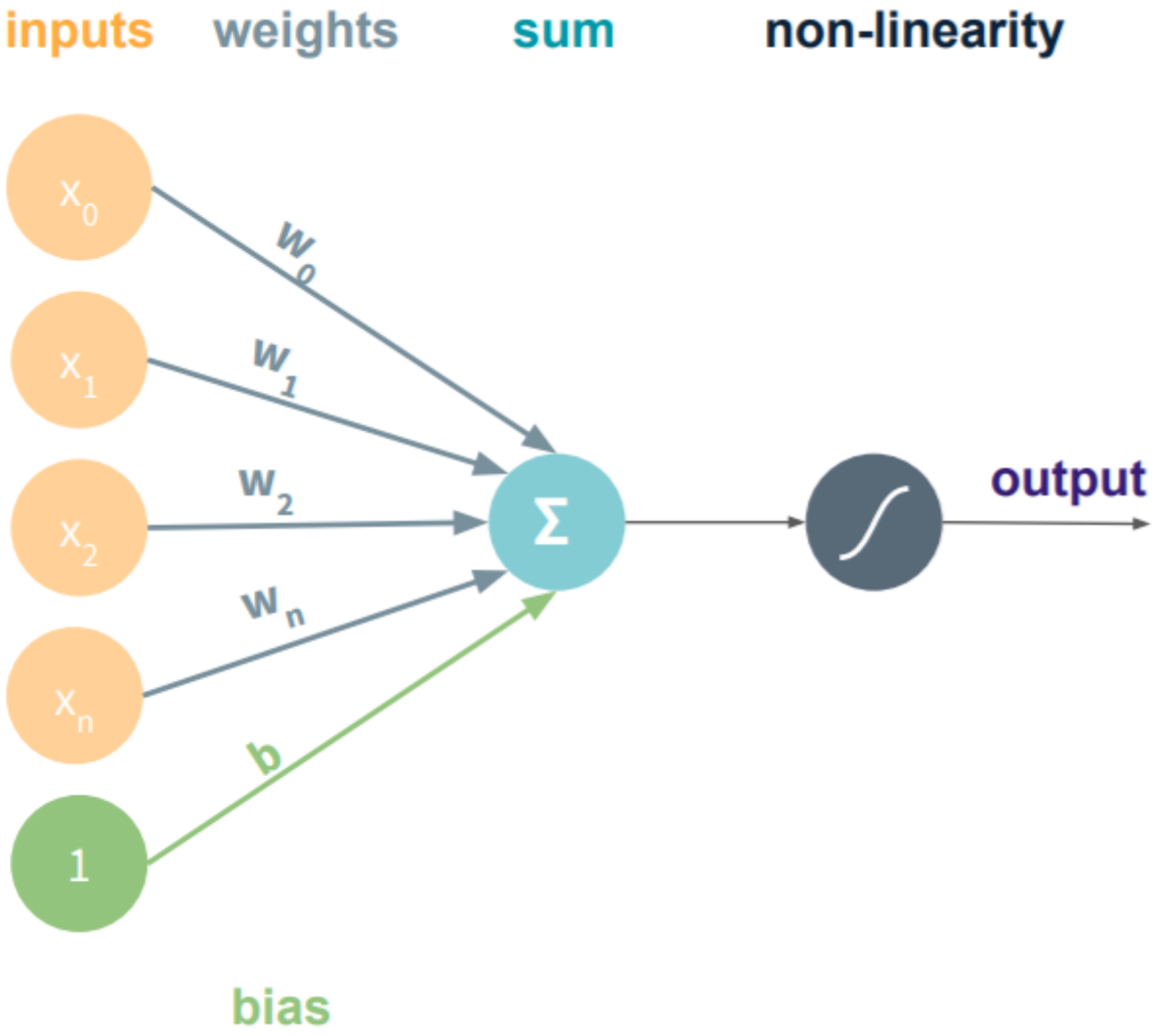


The Perceptron: The Forward Pass

$$output = g(XW + b)$$

$$X = x_0, x_1, \ldots x_n$$

$$W = w_0, w_1, \ldots w_n$$



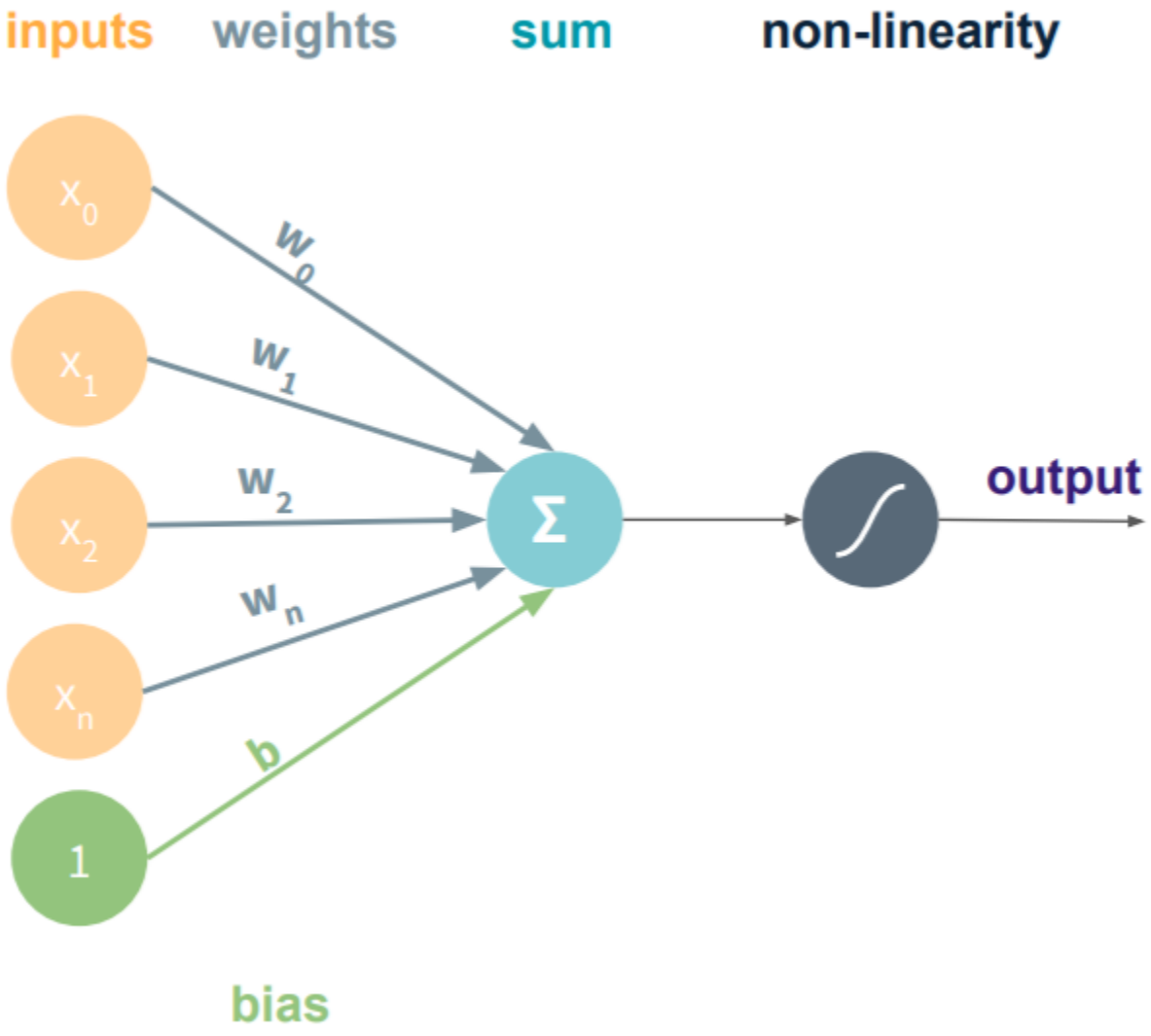
The Perceptron: The Forward Pass

Activation Function

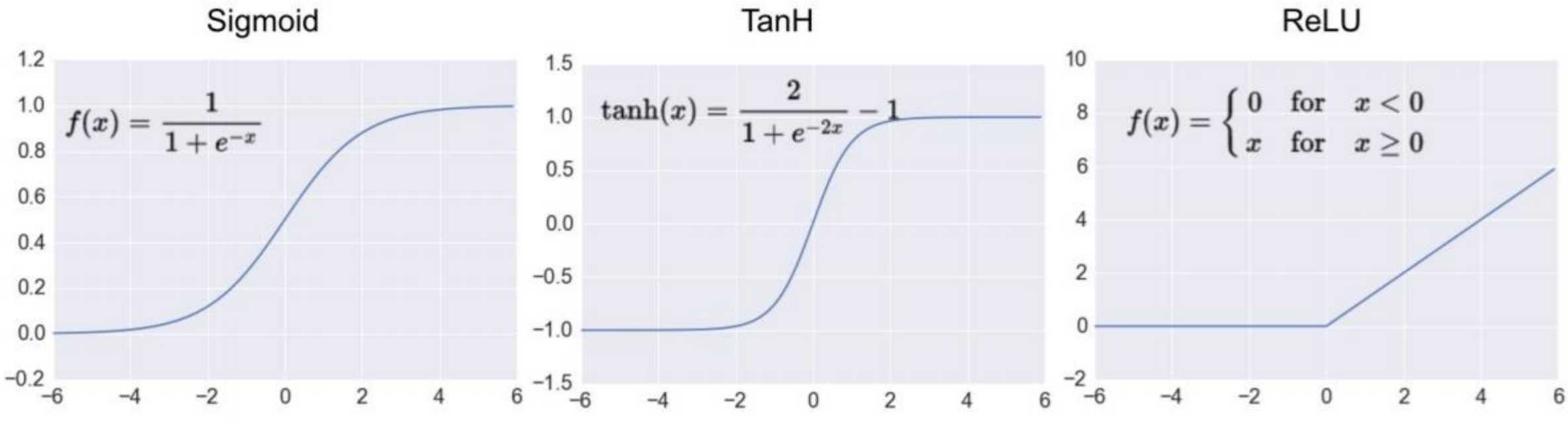
$$output = g(XW + b)$$

$$X = x_0, x_1, \ldots x_n$$

$$W = w_0, w_1, \ldots w_n$$

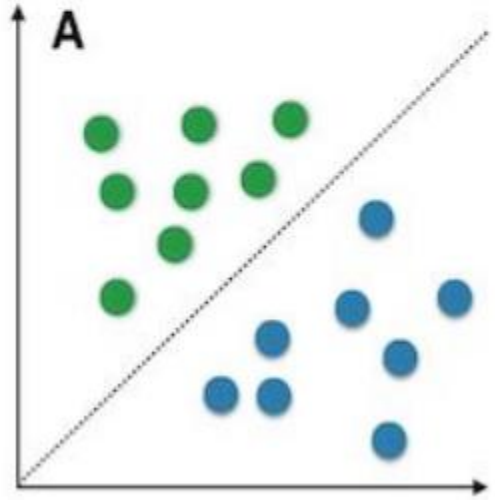


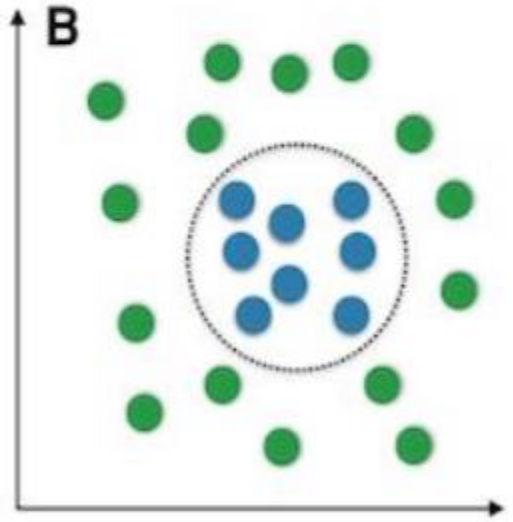
Common Activation Functions



Importance of Activation Functions

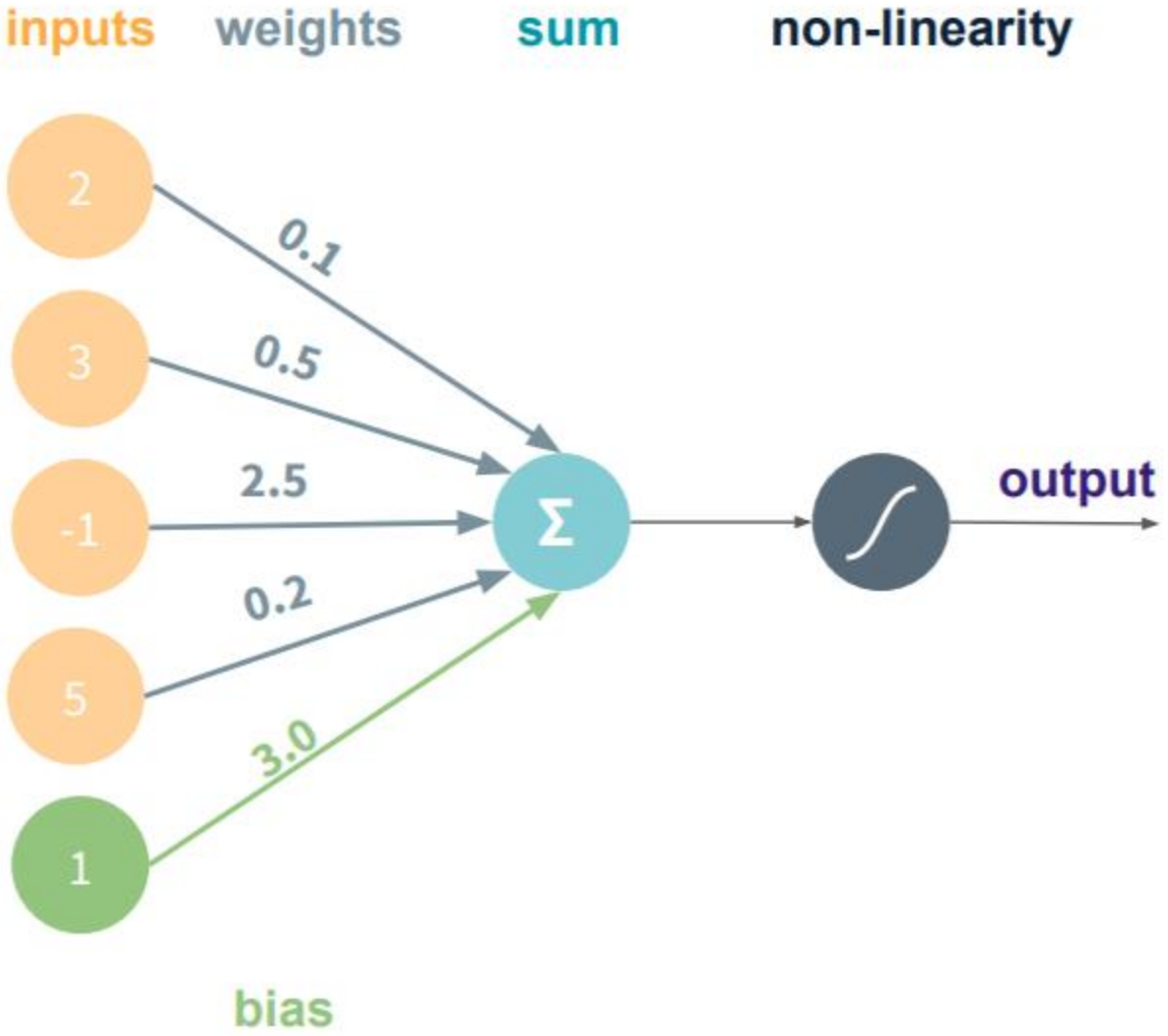
- Activation functions add non-linearity to our network's function
- Most real-world problems + data are **non-linear**





The Perceptron: The Forward Pass

$$output = g(XW + b)$$



The Perceptron: The Forward Pass

$$output = g($$

(2*0.1) +

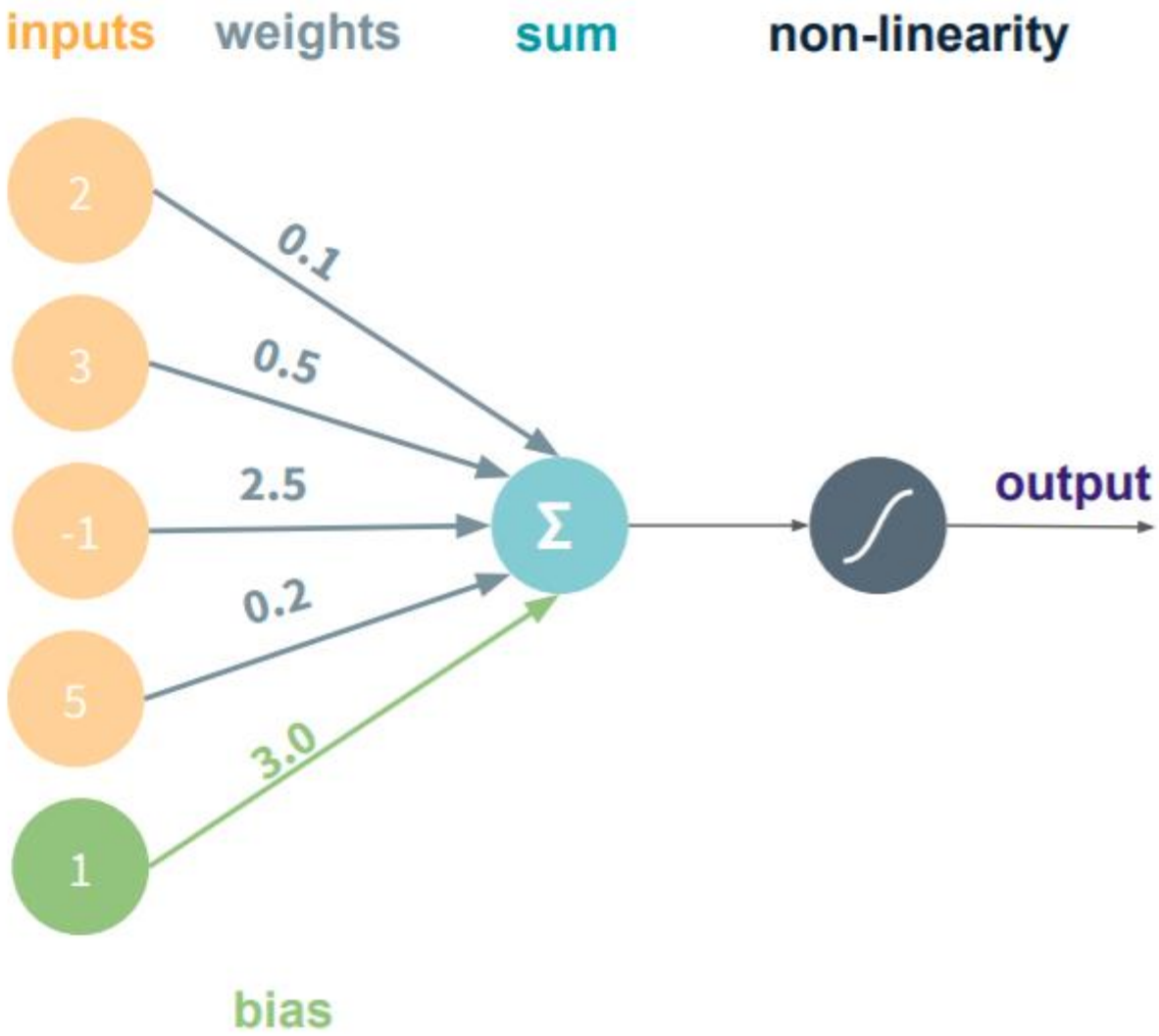
(3*0.5) +

(-1*2.5) +

(5*0.2) +

(1*3.0)

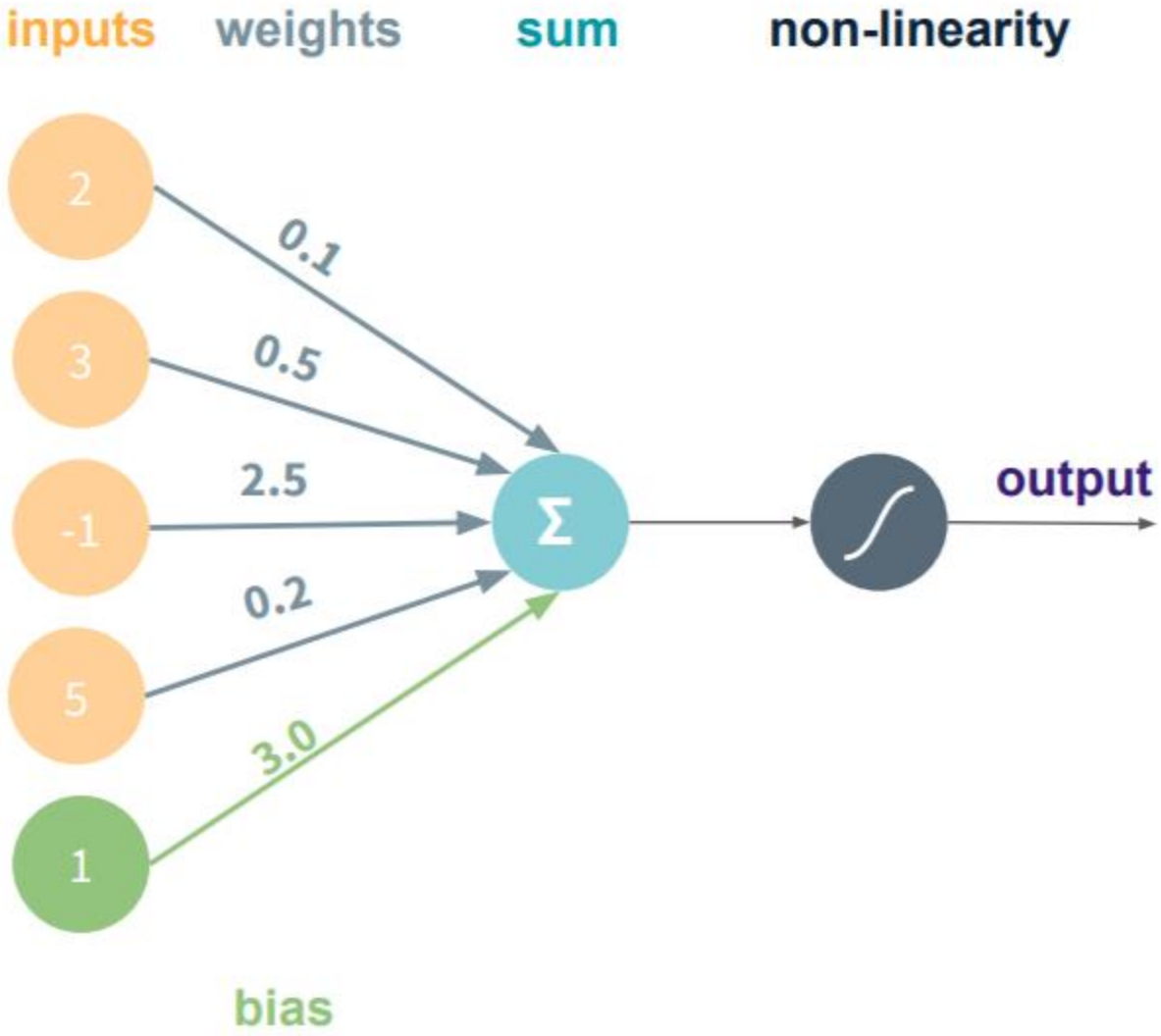
)



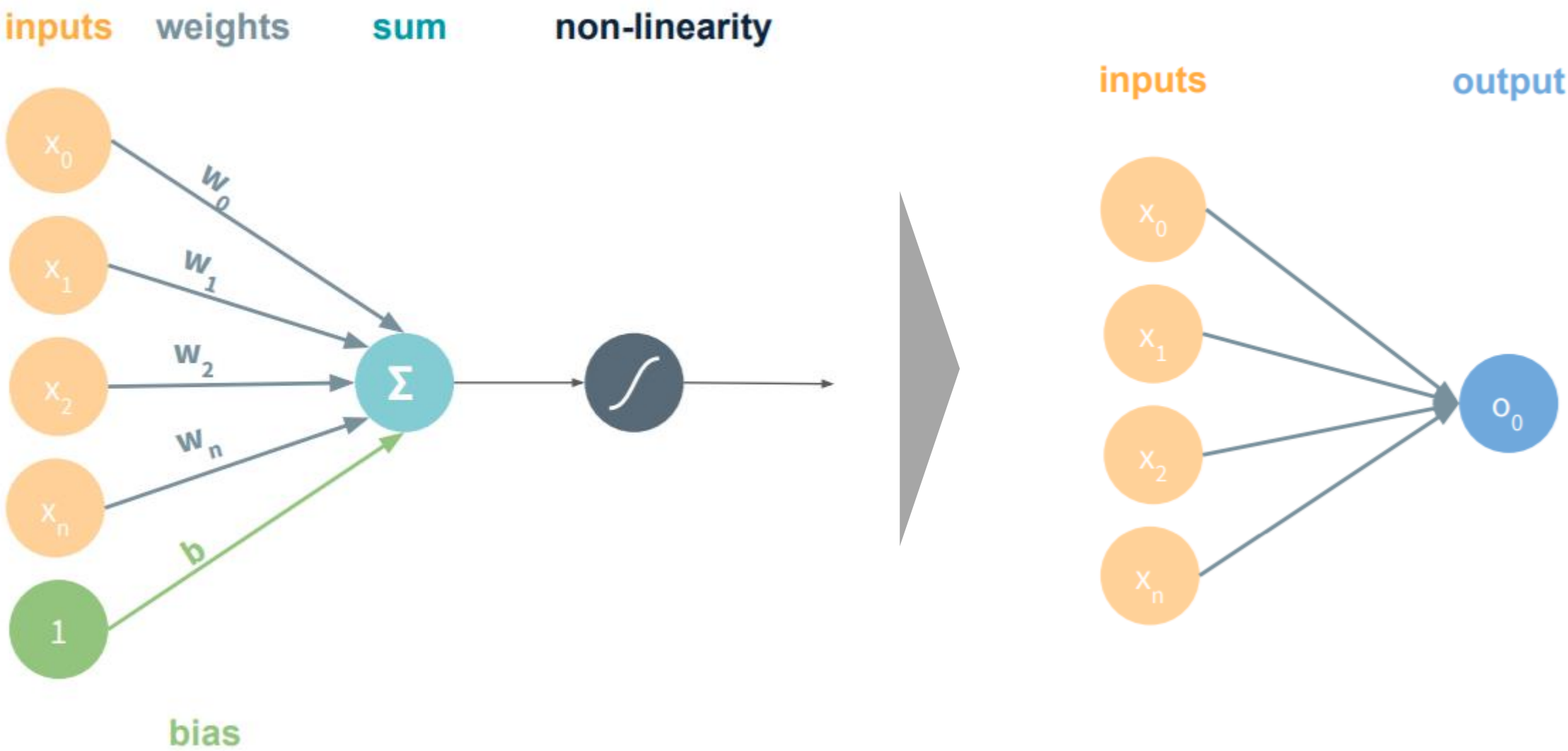
The Perceptron: The Forward Pass

$$output = g(3.2) = \sigma(3.2)$$

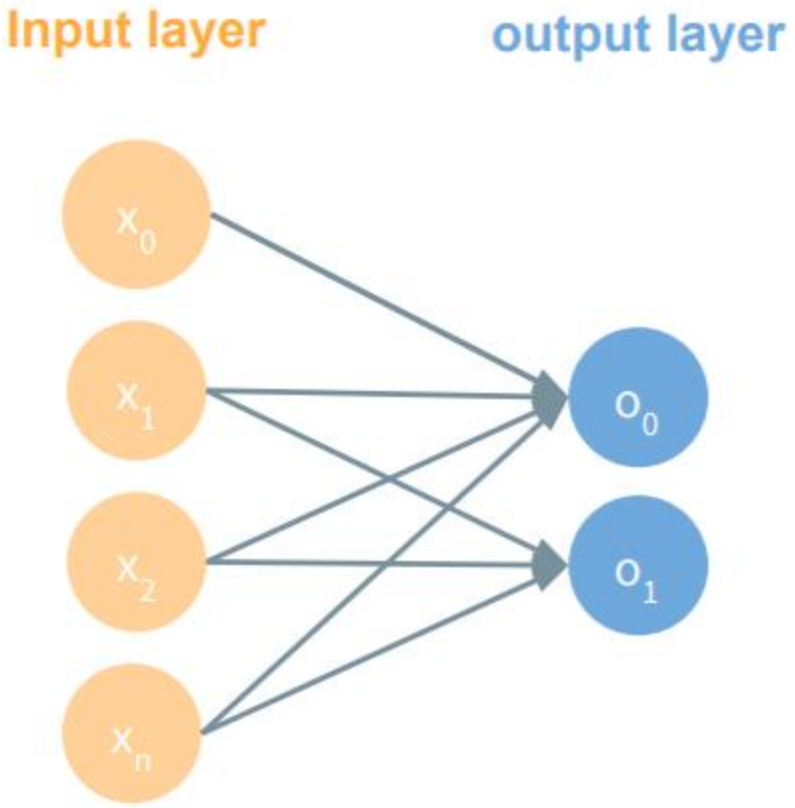
$$= \frac{1}{(1 + e^{-3.2})} = 0.96$$

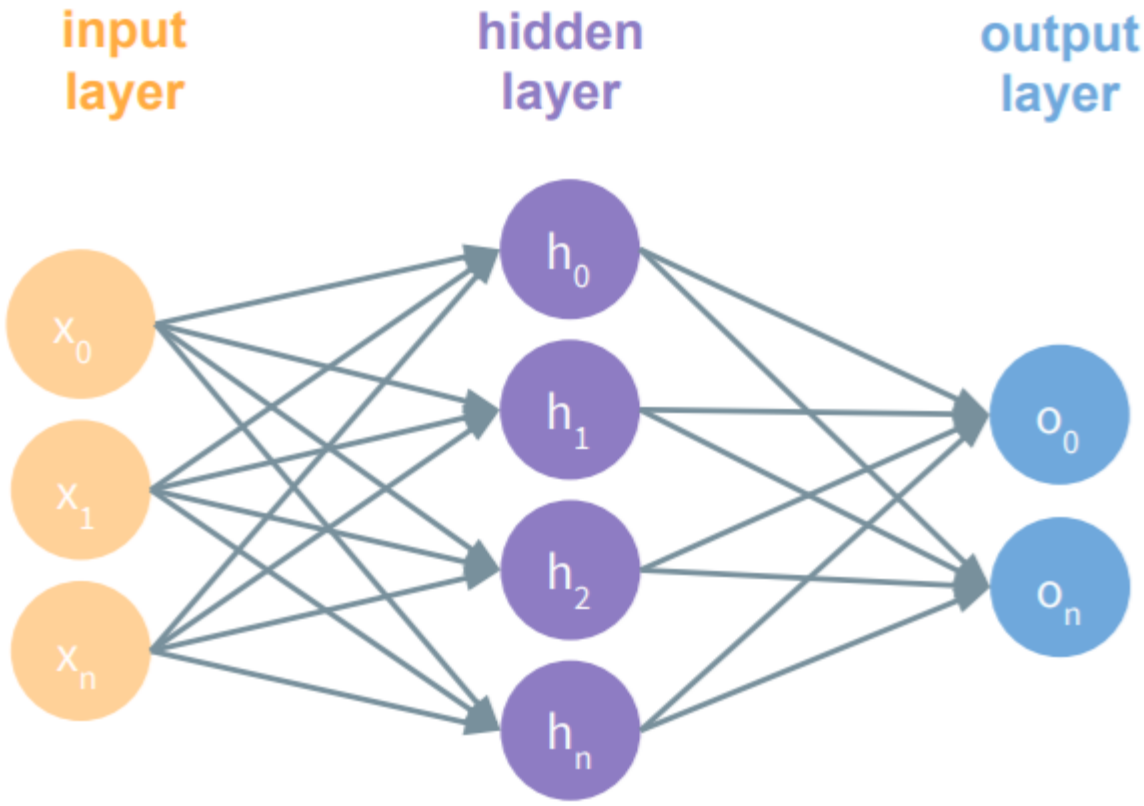


The Perceptron: Simplified

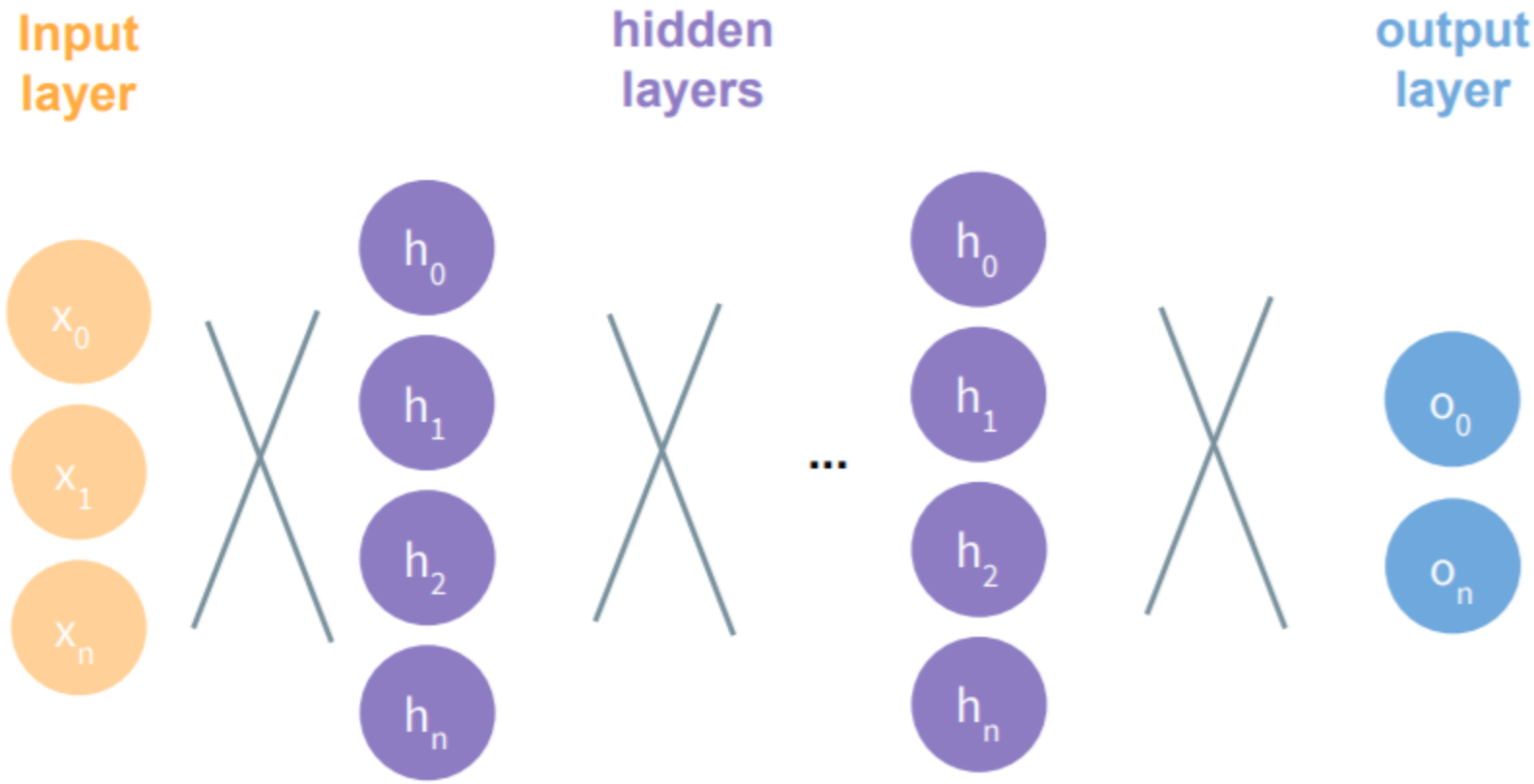


The Perceptron: Multiple Output and Multi Layered





Deep Neural Nets

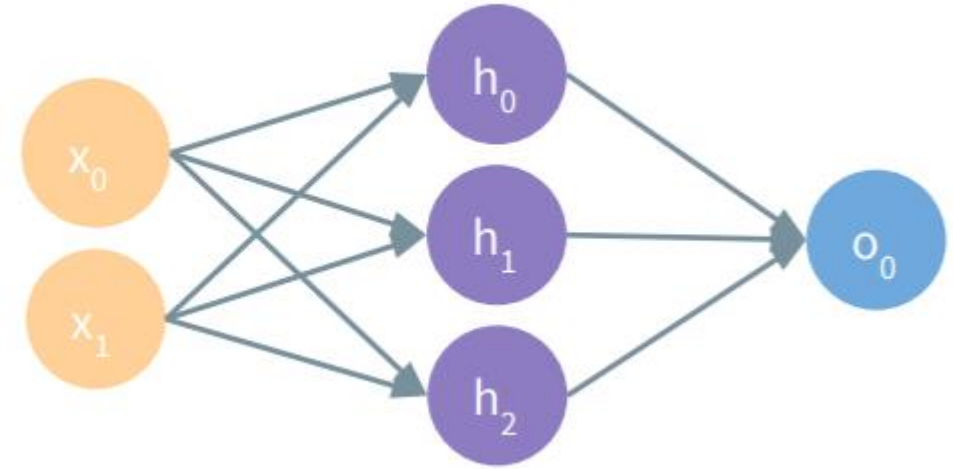


Now comes the Meat (Math)!

Will my flight be delayed?

Temperature -20° C
Wind speed 45 KM/hour

[-20, 45]

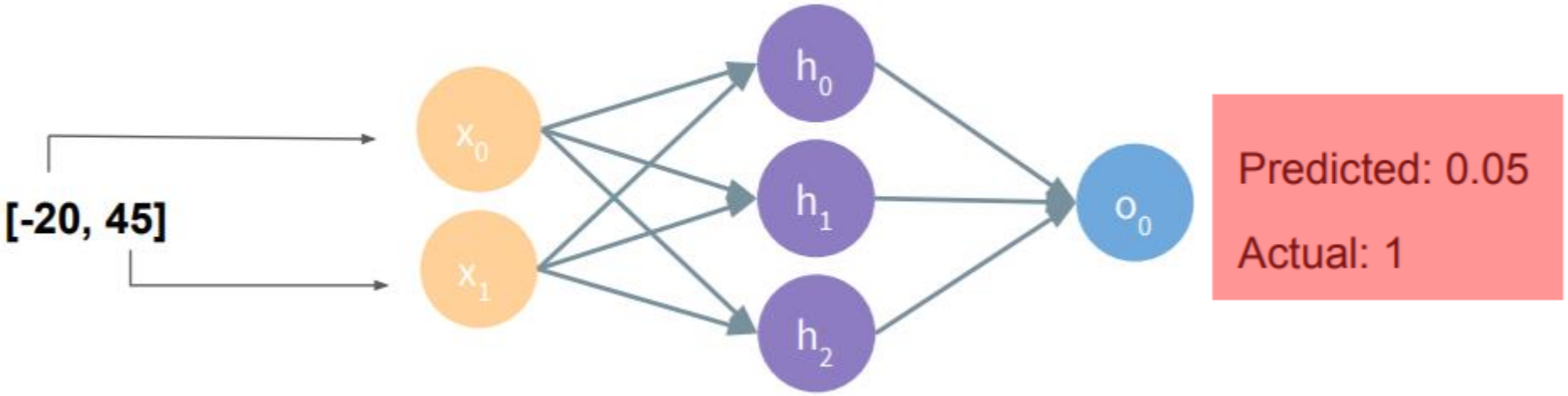


DEPARTURES

TIME	DESTINATION	FLIGHT	GATE	REMARKS
12:39	LONDON	BA 903	31	CANCELLED
12:57	SYDNEY	QF5723	27	CANCELLED
13:08	TORONTO	AC5984	22	CANCELLED
13:21	TOKYO	JL 608	41	DELAYED
13:37	HONG KONG	CX5471	29	CANCELLED
13:48	MADRID	IB3941	30	DELAYED
14:19	BERLIN	LH5021	28	CANCELLED
14:35	NEW YORK	AA 997	11	CANCELLED
14:54	PARIS	AF5870	23	DELAYED
15:10	ROME	AZ5324	43	CANCELLED

Quantifying Loss

Temperature -20° C
Wind speed 45 KM/hour



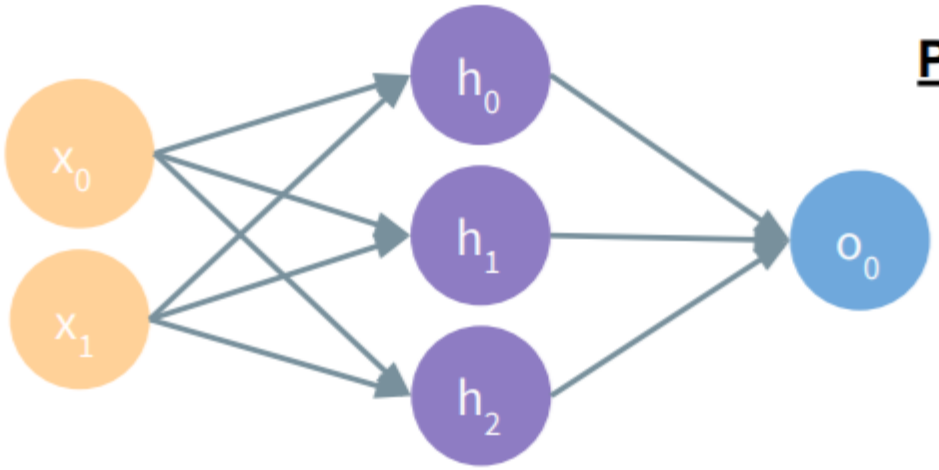
$$loss(f(x^{(i)};\theta), y^{(i)}))$$

Predicted Actual

Quantifying Loss

Input

[
[-20, 45],
[80, 0],
[4, 15],
[45, 60],
]



Predicted	**Actual**
[	[
0.05	1
0.02	0
0.96	1
0.35	1
]	]

$$\text{total loss} := J(\theta) = \frac{1}{N}\sum_{i} loss(f(x^{(i)};\theta), y^{(i)}))$$

Predicted Actual

Training the neural network – Minimize Loss Function

Minimize the number of incorrect predictions

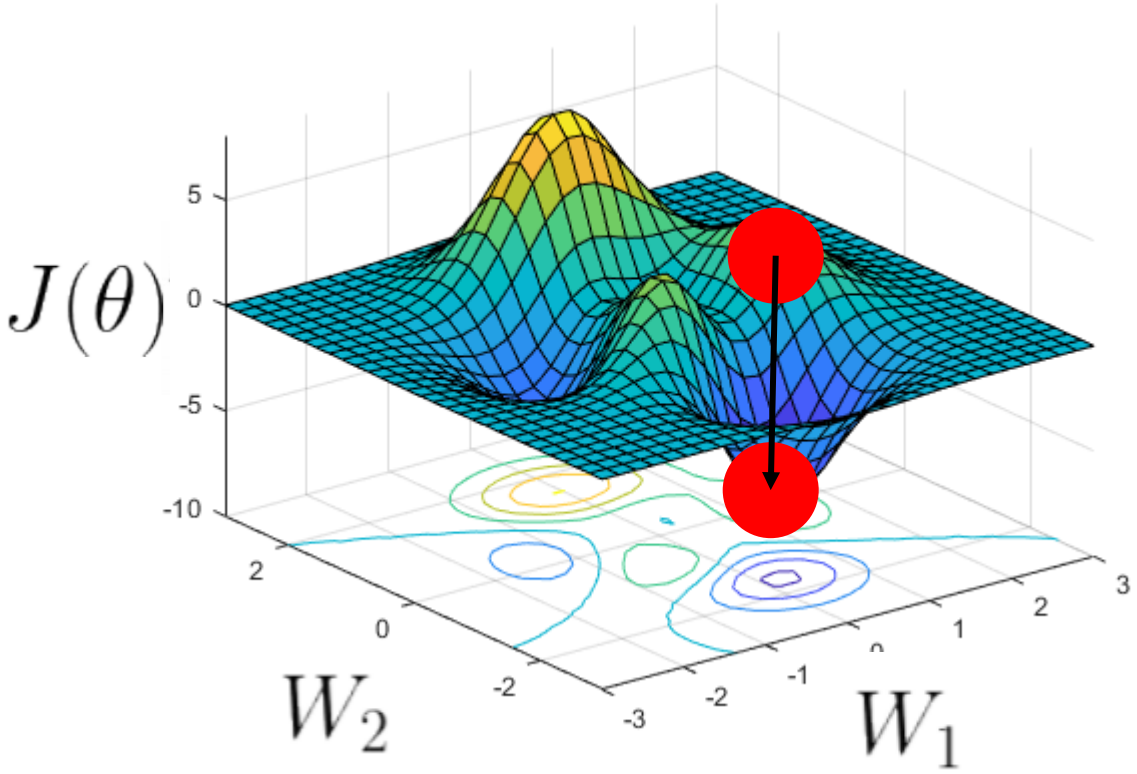
Minimize the Loss function

Mathematically

$$\theta = W_1, W_2 \ldots W_n$$

$$arg_{\theta} \min \frac{1}{N} \sum_{i}^{N} \text{loss}(f(x^{(i)}; \theta), y^{(i)})$$

Training the neural network – Gradient Descent



Gradient Descent Algorithm

- Initialize θ randomly
- For N Epochs
  - For each training example (x, y):
    - Compute Loss Gradient: $\frac{\partial J(\theta)}{\partial \theta}$
    - Update θ with update rule:

$$\theta := \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$

Training the neural network - Backpropagation

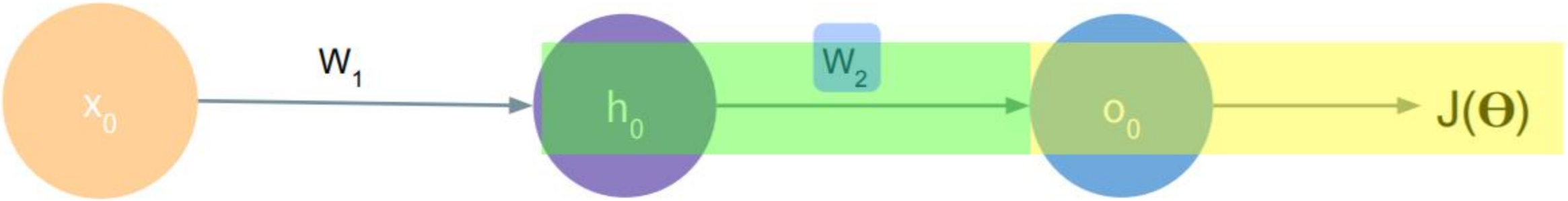
$$\frac{\partial J(\theta)}{\partial W_2} =$$

Training the neural network - Backpropagation

Apply the chain rule

$$\frac{\partial J(\theta)}{\partial W_2} = \frac{\partial J(\theta)}{\partial o_0}$$

Training the neural network - Backpropagation



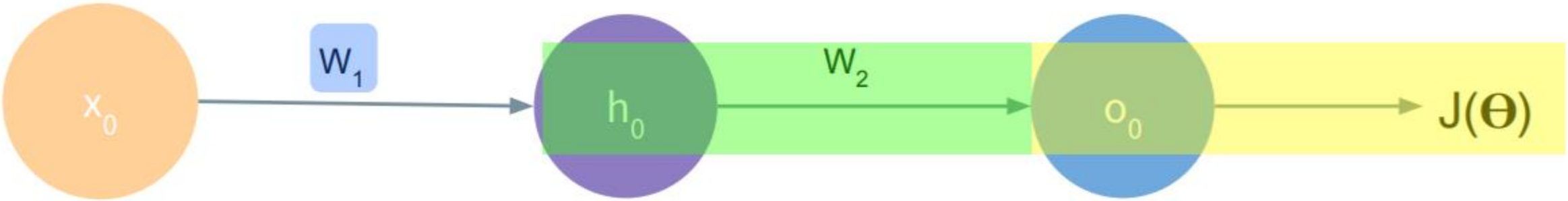
Apply the chain rule

$$\frac{\partial J(\theta)}{\partial W_2} = \frac{\partial J(\theta)}{\partial o_0} * \frac{\partial o_0}{\partial W_2}$$

Training the neural network - Backpropagation

$$\frac{\partial J(\theta)}{\partial W_1} =$$

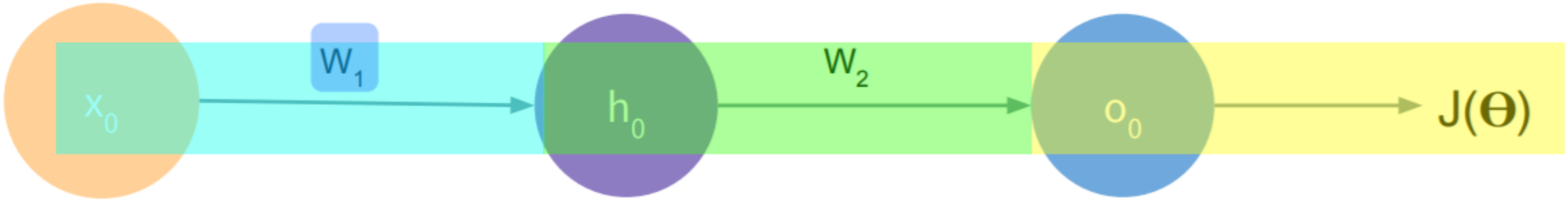
Training the neural network - Backpropagation



Apply the chain rule

$$\frac{\partial J(\theta)}{\partial W_1} = \frac{\partial J(\theta)}{\partial o_0} * \frac{\partial o_0}{\partial h_0}$$

Training the neural network - Backpropagation



Apply the chain rule

Apply the chain rule

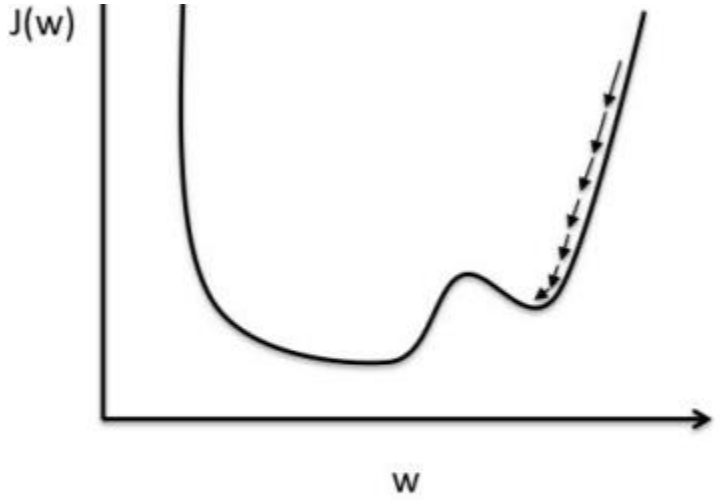
$$\frac{\partial J(\theta)}{\partial W_1} = \frac{\partial J(\theta)}{\partial o_0} * \frac{\partial o_0}{\partial h_0} * \frac{\partial h_0}{\partial W_1}$$

Training the neural network – Learning Rate

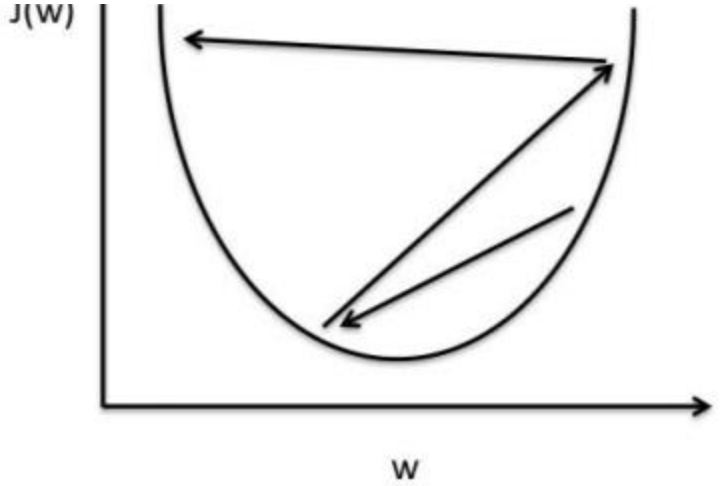
How to Choose Learning Rate?

Update Rule:

$$\theta := \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$



Small learning rate: Many iterations until convergence and trapping in local minima.



Large learning rate: Overshooting.

Training the neural network – Learning Rate

Try different Learning Rates and observe if solution converges

OR

Use an adaptive learning rate which is larger initially and decreases as we get close to the minima point

Now comes the Wine (Programming)!