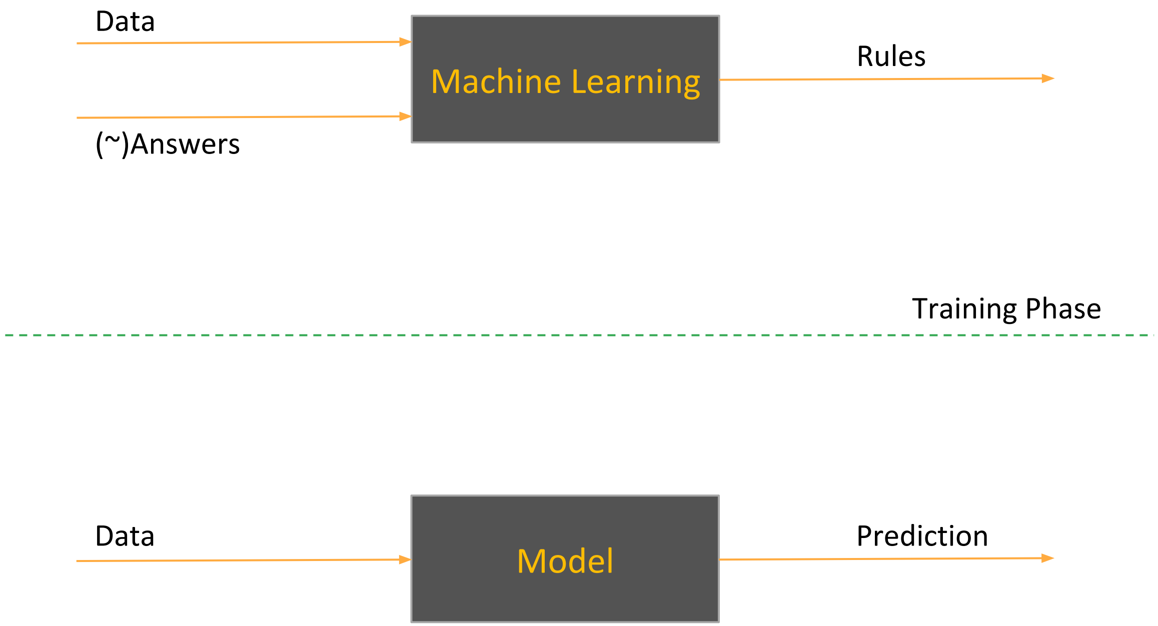
Neural Networks with Tensorflow 2.0

What is Machine Learning

Traditional Vs ML

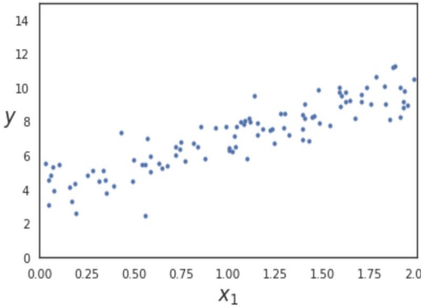


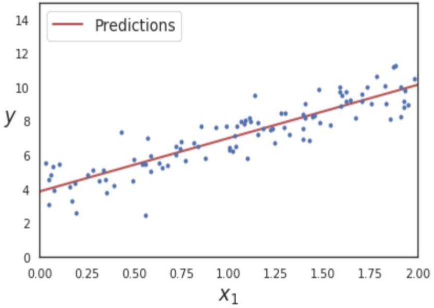
Types of Learning

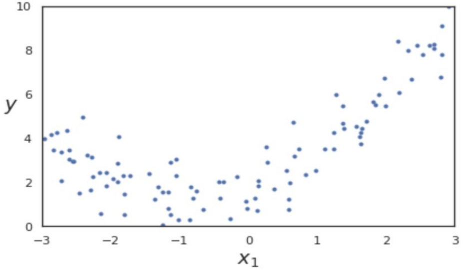
- Supervised
- Unsupervised

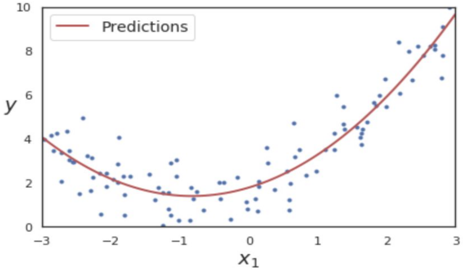
Supervised Learning Tasks

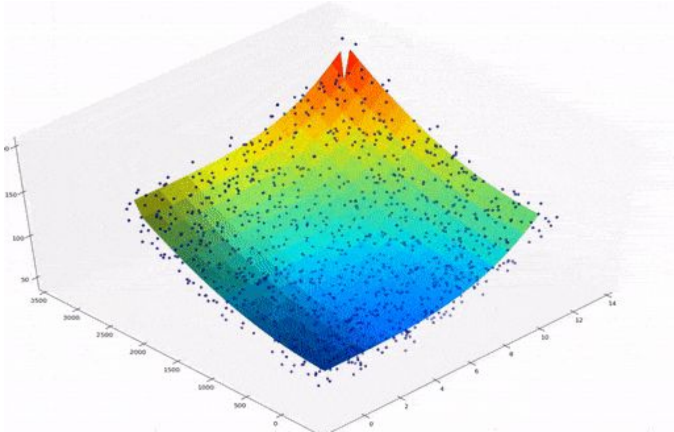
Linear Regression





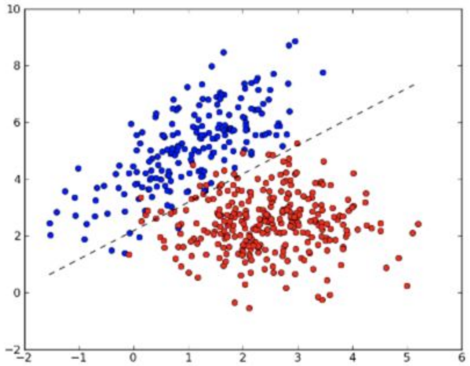


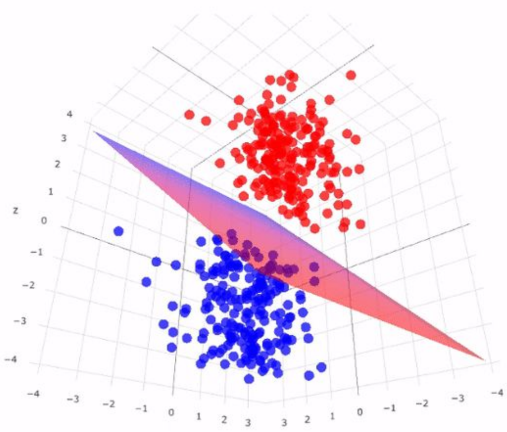




Supervised Learning Tasks

Classification





Training Data

Input Features and Output Labels For regression

Input Features

Area	Bedrooms	Bathroom	Study room
800	2	1	0
1300	3	3	1
400	1	1	0
2100	4	4	1
2700	4	4	1

no. of example = m

no. of features = n

Output Labels

Price
22.87 Lacs
51.56 Lacs
17.56 Lacs
80.8 Lacs
100.12 Lacs

Training Data

Input Features and Output Labels For Classification

Input Features

Area	Bedrooms	Bathroom	Study room
800	2	1	0
1300	3	3	1
400	1	1	0
2100	4	4	1
2700	4	4	1

no. of example = m

no. of features = n

Output Labels

Cheap	Affordable	expensive
1	0	0
0	1	0
1	0	0
0	0	1
0	0	1

Representing Data

Representing a House as Vector

$$\begin{array}{cl} \begin{bmatrix} 800 \\ 2 \\ 1 \\ 0 \end{bmatrix} & \begin{array}{l} \text{Area} \\ \text{Bedrooms} \\ \text{Bathrooms} \\ \text{Study} \end{array} \end{array}$$

4×1

Column Vector of dimension 4 by 1

$$\begin{array}{cccc} \text{Area} & \text{Bedrooms} & \text{Bathrooms} & \text{Study} \\ \end{array}$$

$$\begin{bmatrix} 800 & 2 & 1 & 0 \end{bmatrix}_{1 \times 4}$$

Row Vector of dimension 1 by 4

Representing Data

Representing a Houses as

Matrix

$$
\begin{array}{c c l}
 & \begin{array}{cccccc} \text{Feature -1} & & \text{Feature -3} & & & \text{Feature -n} \end{array} & \\
\text{no. of Houses/examples = m} & \begin{bmatrix} 800 & 2 & 1 & - & - & 0 \\ 1300 & 3 & 3 & - & - & 1 \\ 400 & 1 & 1 & - & - & 0 \\ 2100 & 4 & 4 & - & - & 1 \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ 2700 & 4 & 4 & - & - & 1 \end{bmatrix} & \begin{array}{l} \text{House 1} \\ \text{House 2} \\ \\ \\ \\ \text{House m} \\ \\ \end{array} \\
 & \text{no. of features = n} &
\end{array}
$$

$m \times n \rightarrow$ Shape

Matrix

Element-Wise operation

$$\begin{bmatrix} 2 & 3 & 7 \\ 4 & 8 & 2 \end{bmatrix} + \begin{bmatrix} 5 & 1 & 2 \\ 3 & 9 & 2 \end{bmatrix} = \begin{bmatrix} 7 & 4 & 9 \\ 7 & 17 & 4 \end{bmatrix}$$

Sum

$$\begin{bmatrix} 2 & 3 & 7 \\ 4 & 8 & 2 \end{bmatrix} \odot \begin{bmatrix} 5 & 1 & 2 \\ 3 & 9 & 2 \end{bmatrix} = \begin{bmatrix} 10 & 3 & 14 \\ 12 & 72 & 4 \end{bmatrix}$$

Multiplication

$$\begin{bmatrix} 2 & 3 & 7 \\ 4 & 8 & 2 \end{bmatrix} - \begin{bmatrix} 5 & 1 & 2 \\ 3 & 9 & 2 \end{bmatrix} = \begin{bmatrix} -3 & -2 & 5 \\ 1 & -1 & 0 \end{bmatrix}$$

Subtraction

Shape of matrices must be same

Matrix

Applying Functions

$$f(\begin{bmatrix} 2 & 3 & 7 \\ 4 & 8 & 2 \end{bmatrix}) = \begin{bmatrix} f(2) & f(3) & f(7) \\ f(4) & f(8) & (2) \end{bmatrix} = \begin{bmatrix} 0.88079708 & 0.95257413 & 0.99908895 \\ 0.98201379 & 0.99966465 & 0.88079708 \end{bmatrix}$$

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

Matrix

Matrix Multiplication

Dot Product of two vector :

$$\begin{bmatrix} 0 & 2 & 4 & 6 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 7 \\ 13 \\ 19 \end{bmatrix} = 0 * 1 + 2 * 7 + 4 * 13 + 6 * 19 = 180$$

Two matrix can only be multiplied when:
number of columns in first matrix = number of rows in the second matrix

$$\begin{bmatrix} \# & \# \\ \# & \# \end{bmatrix}_{2*\underline{2}} \cdot \begin{bmatrix} \# & \# \\ \# & \# \\ \# & \# \end{bmatrix}_{\underline{3}*2}$$

Cannot be multiplied

$$\begin{bmatrix} \# & \# \\ \# & \# \end{bmatrix}_{2*\underline{2}} \cdot \begin{bmatrix} \# & \# & \# \\ \# & \# & \# \end{bmatrix}_{\underline{2}*3} = \begin{bmatrix} \# & \# & \# \\ \# & \# & \# \end{bmatrix}_{2*3}$$

Shape of resultant matrix
number of row in first matrix -by- number of columns in the second matrix

Matrix

Matrix Multiplication

Thumb Rule: Dot product of rows of first matrix with columns of second matrix

$$\begin{bmatrix} 0 & 2 & 4 & 6 \\ 8 & 10 & 12 & 14 \end{bmatrix} \cdot \begin{bmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \\ 13 & 15 & 17 \\ 19 & 21 & 23 \end{bmatrix} = \begin{bmatrix} \boxed{180} & 204 & 228 \\ 500 & 588 & 676 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 & 4 & 6 \\ 8 & 10 & 12 & 14 \end{bmatrix} \cdot \begin{bmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \\ 13 & 15 & 17 \\ 19 & 21 & 23 \end{bmatrix} = \begin{bmatrix} 180 & \boxed{204} & 228 \\ 500 & 588 & 676 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 & 4 & 6 \\ 8 & 10 & 12 & 14 \end{bmatrix} \cdot \begin{bmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \\ 13 & 15 & 17 \\ 19 & 21 & 23 \end{bmatrix} = \begin{bmatrix} 180 & 204 & \boxed{228} \\ 500 & 588 & 676 \end{bmatrix}$$

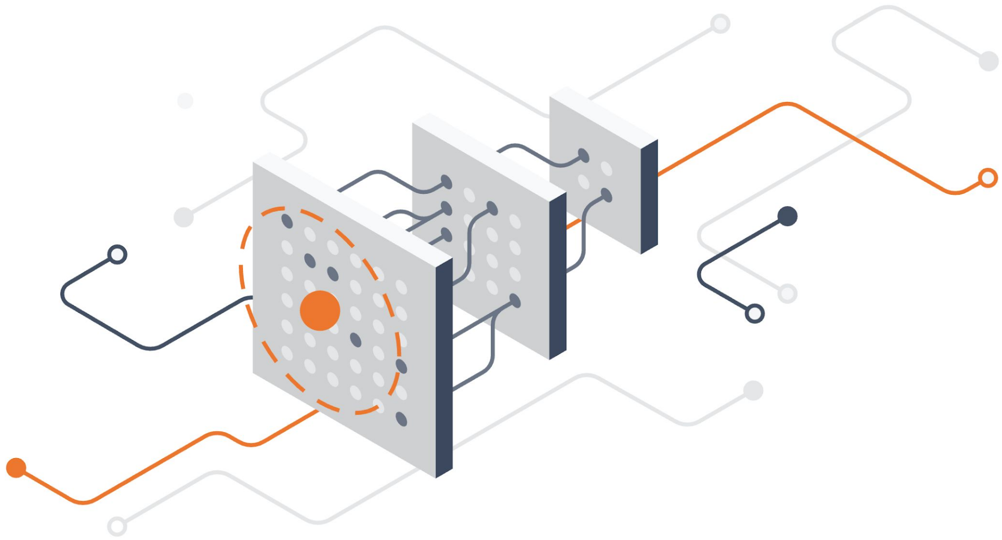
Matrix

Question

$$\begin{bmatrix} 1 * n \end{bmatrix} \cdot \begin{bmatrix} ? * ? \end{bmatrix} = \begin{bmatrix} 1 * i \end{bmatrix}$$

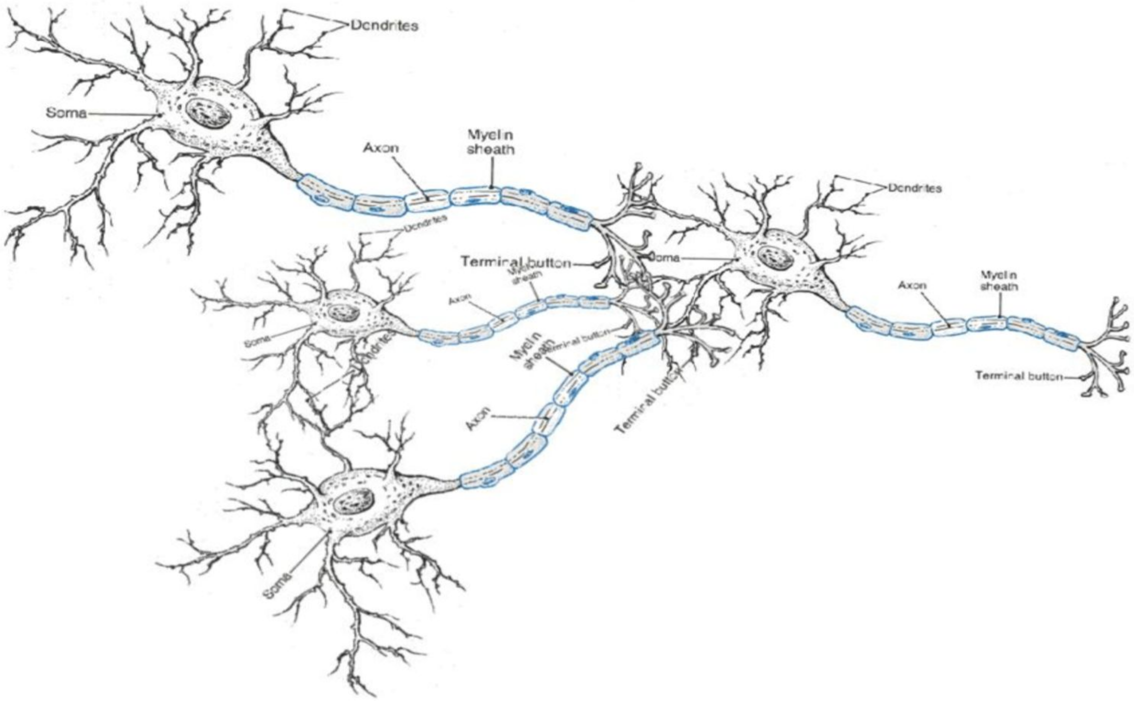
Part II

Neural Networks



The Brain

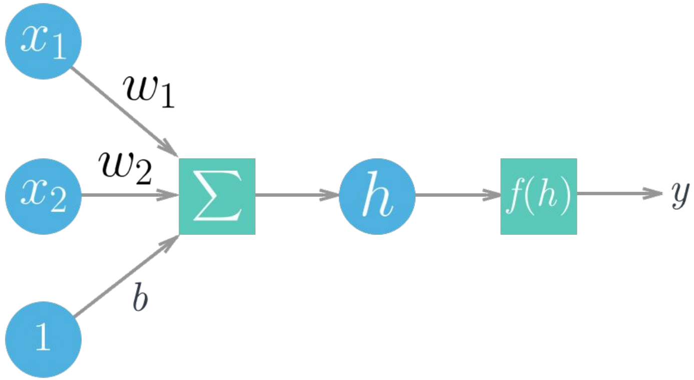
How does it work?



Perceptron Model

Weighted Sum of

Inputs



$$h = w_1 x_1 + w_2 x_2 + b = \sum_{i=1}^{2} w_i x_i + b$$

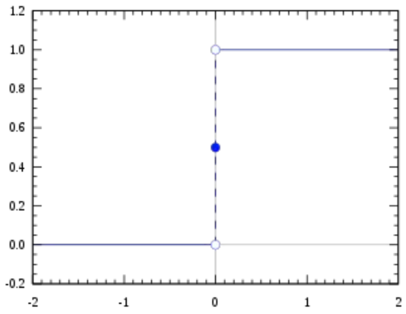
$$y = f\,(h)$$

x_i = i^{th} Input

w_i = weight for i^{th} Input

f = Activation Function

y = Output

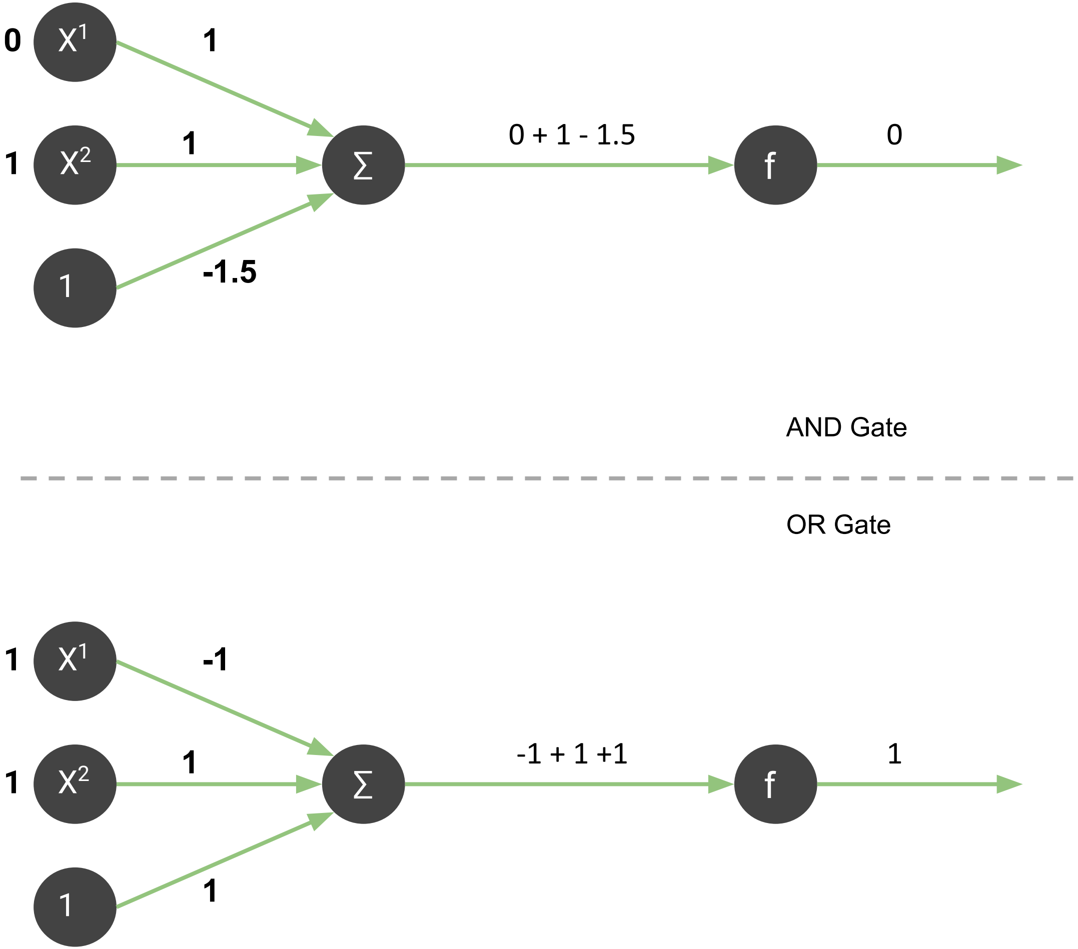


Step Function

Perceptron Model

AND & OR

Perceptrons



Perceptron Model

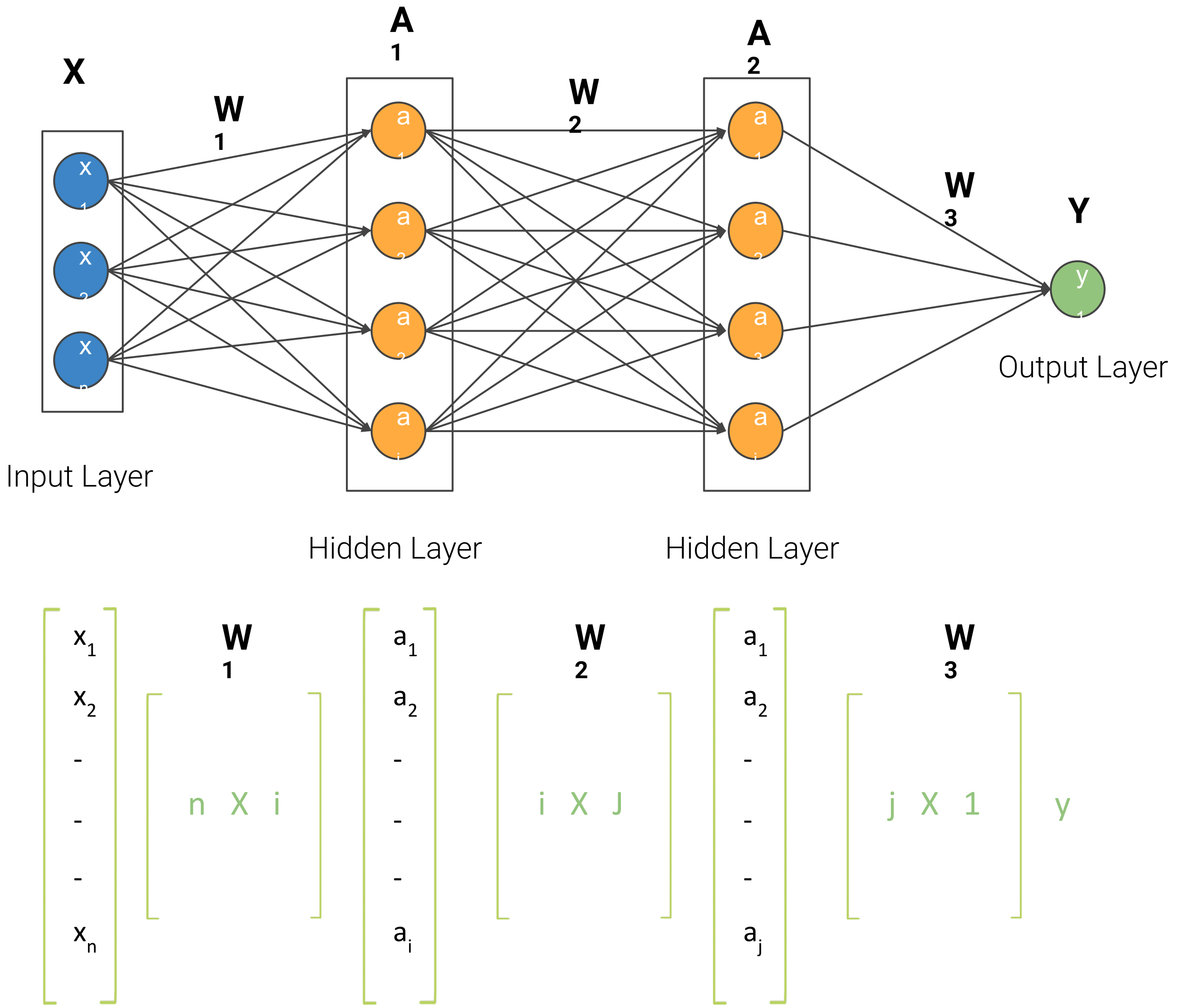
So, by changing weight same model behaves differently

We’ll see that weights are what neural networks learn, to make prediction

Neural Network

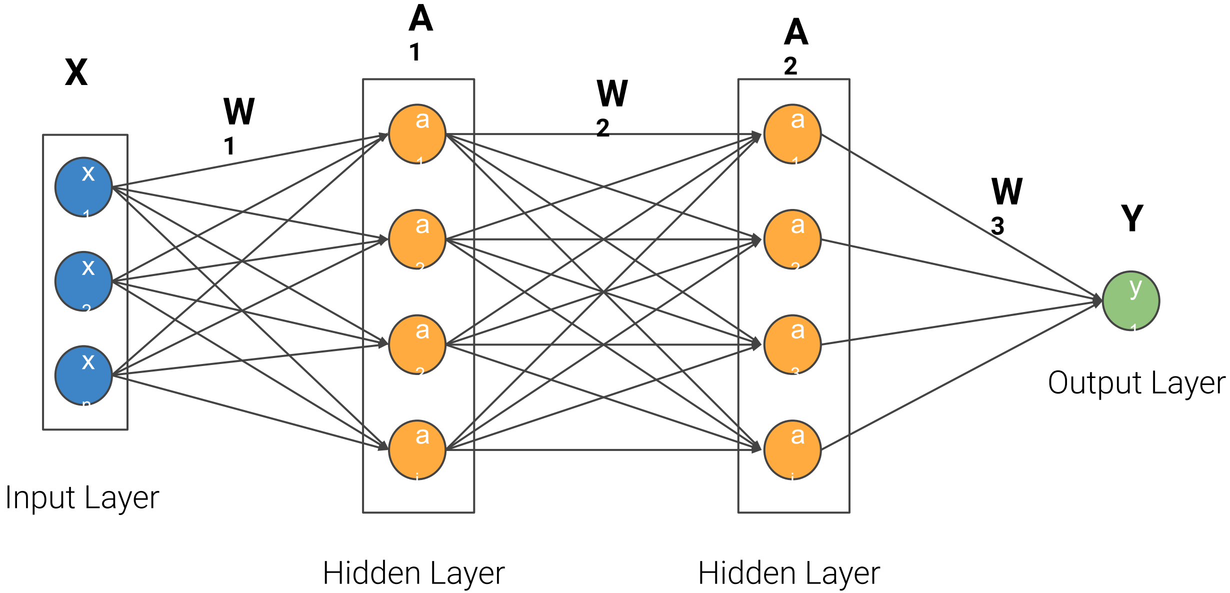
Perceptrons are now called neuron or unit

And we will now talk in terms of layers



Neural Network

FeedForward = Calculating Y



$H^1 = X \cdot W^1$ Dot Product

$A^1 = \sigma(H^1)$ Element-wise

$H^2 = X \cdot W^2$ Dot Product

$A^2 = \sigma(H^2)$ Element Wise

$$\hat{y} = A^2 \cdot W^3$$

Splitting Data

n = features

m
Training
samples

800	2	1	0
1300	3	3	1
400	1	1	0
2100	4	4	1
2700	4	4	1
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
2800	5	5	1

Train = 70%

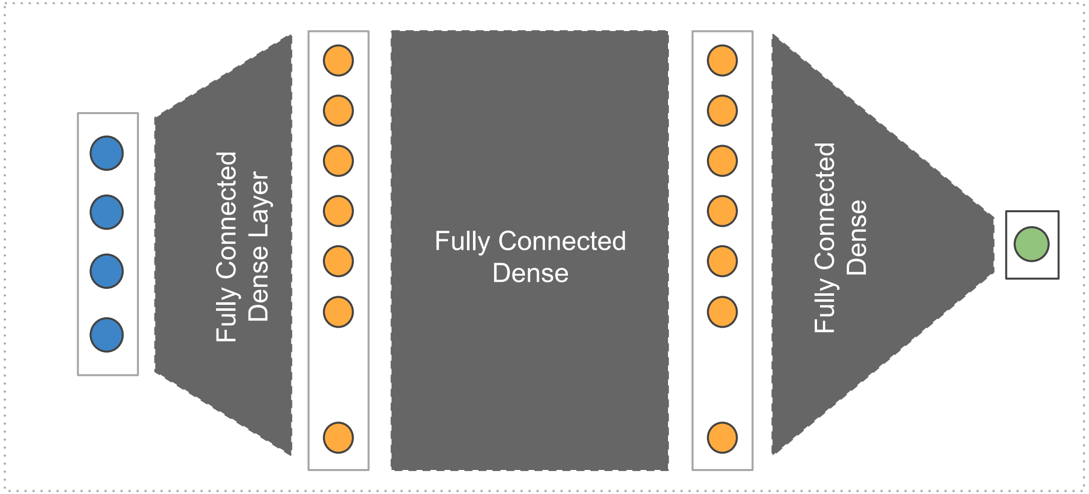
Validation = 10%

Test = 20%

matrix = m by n

Coding Neural Nets

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```




```
model = keras.Sequential()

model.add(layers.Dense(64, activation=tf.nn.relu, input_shape=num_features))

model.add(layers.Dense(64, activation=tf.nn.relu))

model.add(layers.Dense(1))
```

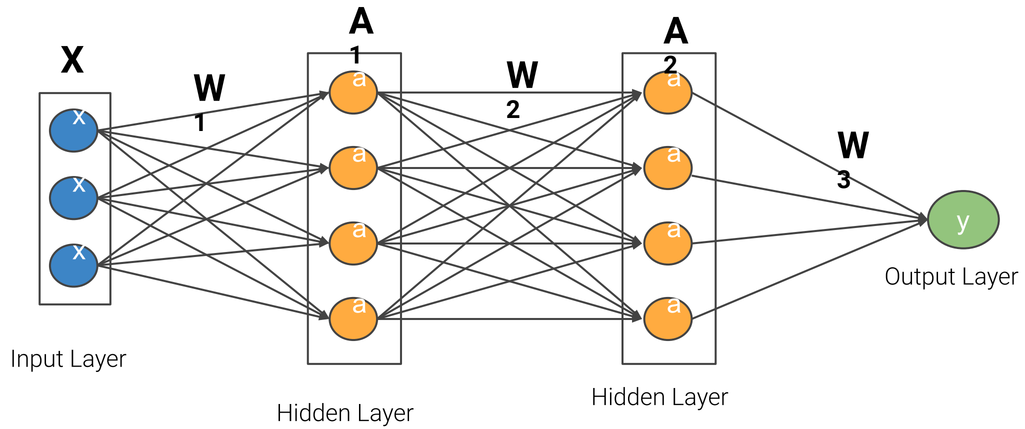
Coding Neural Nets

Hands-On

http://bit.ly/cop-reg

Calculating Error

Error == Loss == Cost



We Have:

- $[x_1, x_2, x_3, \ldots\ldots, x_n]$ = Input features
- y = Actual Label

We know how to calculate:

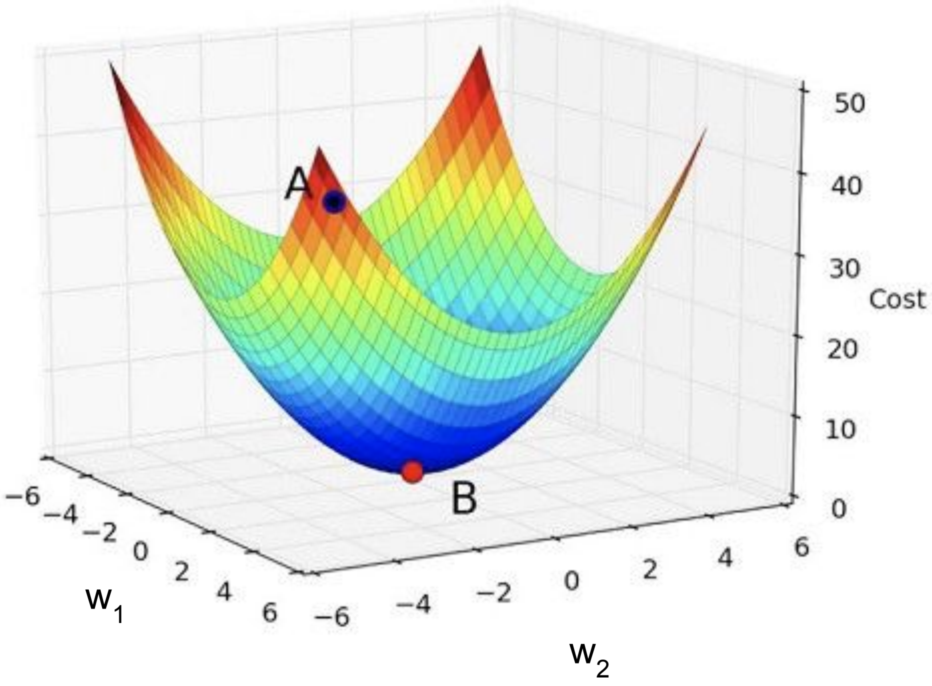
- y' = Predicted Value

We Define error as

$$J(W, X) = ((y - y')^2)/2$$

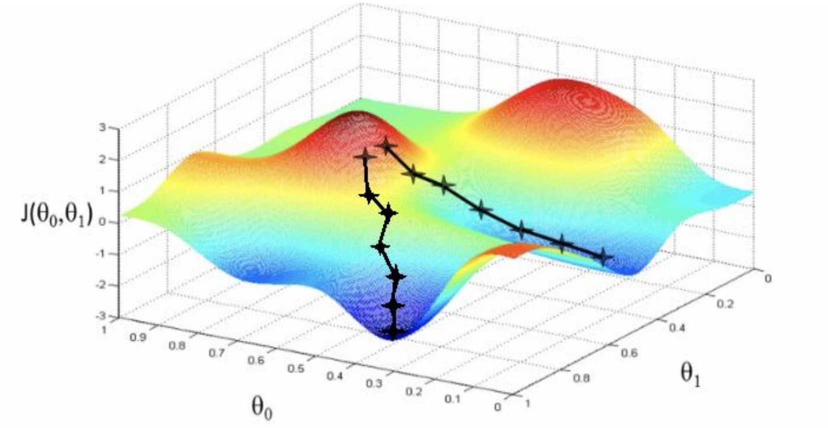
Our goal is to minimize $J(W)$

Minimizing or Optimizing



Gradient Descent

- It is an optimization algorithm
- that reaches minima, by updating parameters/weights,
- moving in a direction opposite to gradient
- iteratively



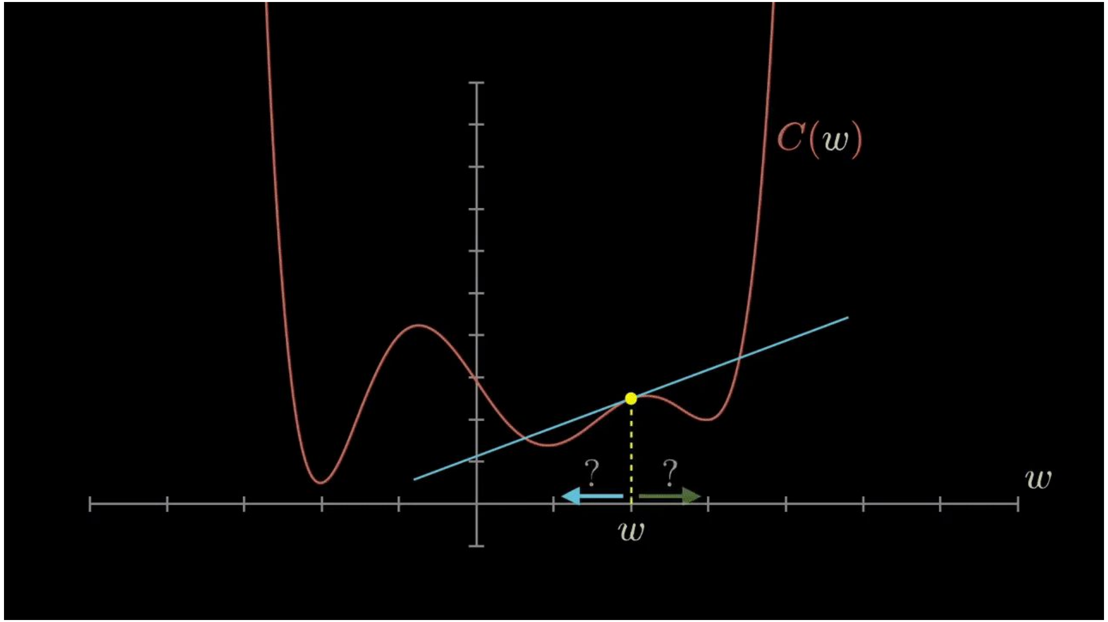
Coding Neural Nets

```
optimizer = tf.keras.optimizers.RMSprop(0.001)

model.compile(loss='mean_squared_error',
                optimizer=optimizer,
                metrics=['mean_absolute_error',
                 'mean_squared_error'])


EPOCHS = 1000

history = model.fit(
  normed_train_data, train_labels,
  epochs=EPOCHS, validation_split = 0.2, verbose=0,
  callbacks=[PrintDot()])
```

Gradients

- Take it to white Board

Gradients Descent



Gradient Descent

$$W_{ij} = W_{ij} + \Delta W_{ij}$$

where $\Delta W_{ij} = \alpha * \left(-\frac{d(J)}{d(W_{ij})}\right)$

Learning rate

negative of the gradient

$$\implies W_{ij} = W_{ij} - \alpha \frac{d(J)}{d(W_{ij})}$$

Coding Neural Nets

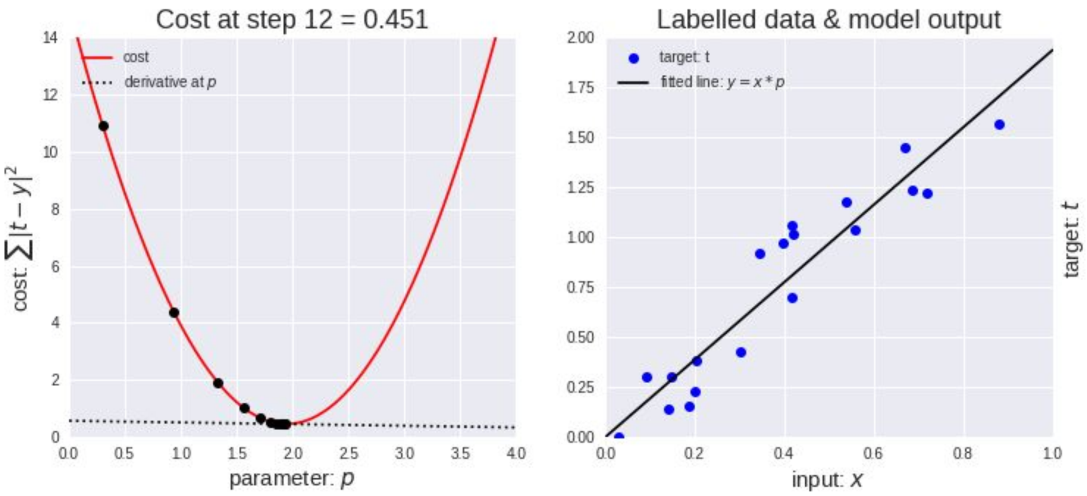
Notebook

http://bit.ly/cop-reg

Gradients Descent

What is actually

Happening?



Parameters vs Hyperparameters

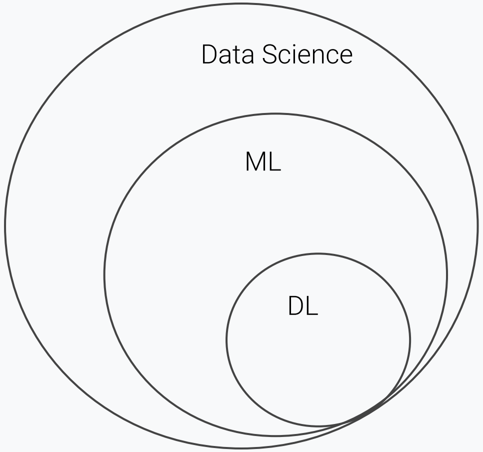
Parameters	Hyperparameters
Weight	Learning Rate
Biases	Optimizer
	Number of Layers
	Number of neuron in those layers
	Activation Function
	Kernel initializer
	Dropout*
	L2 Regularization*

Treat your model as a lab rat

What does COP offer?

- Study Group based learning & discussion
- More sessions on building interesting models
- You can request for more sessions on Neural Network or Tensorflow/Pytorch
- Or you can discuss your ideas about cop with Puneet and Mahesh

ML/DL Resources

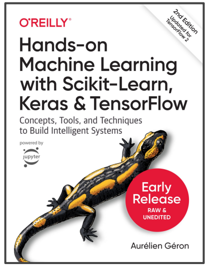


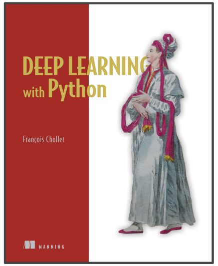
- Udacity Tensorflow Course
- Google’s Machine Learning Crash Course
- Introduction to Deep Learning With Pytorch ++
- Coursera Machine Learning Course by Andrew Ng ++
- Deep Learning by Andrew Ng (4 Courses)

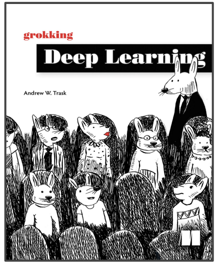
- Numpy
- Pandas
- Matplotlib

- Learn and develop in groups
- Start with just enough maths and then dive a little deeper as required
- Start with a project, gain more knowledge and apply

For Maths, start with 3Blue1Brown for Linear Algebra and Calculus







Questions!

QA

Thank you!

“Technology is a powerful force in our society. Data, software, and communication can be used for bad: to entrench unfair power structures, to undermine human rights, and to protect vested interests. But they can also be used for good: to make underrepresented people’s voices heard, to create opportunities for everyone, and to avert disasters. This book is dedicated to everyone working toward the good.”

-Martin Kleppmann

Designing Data Intensive Applications