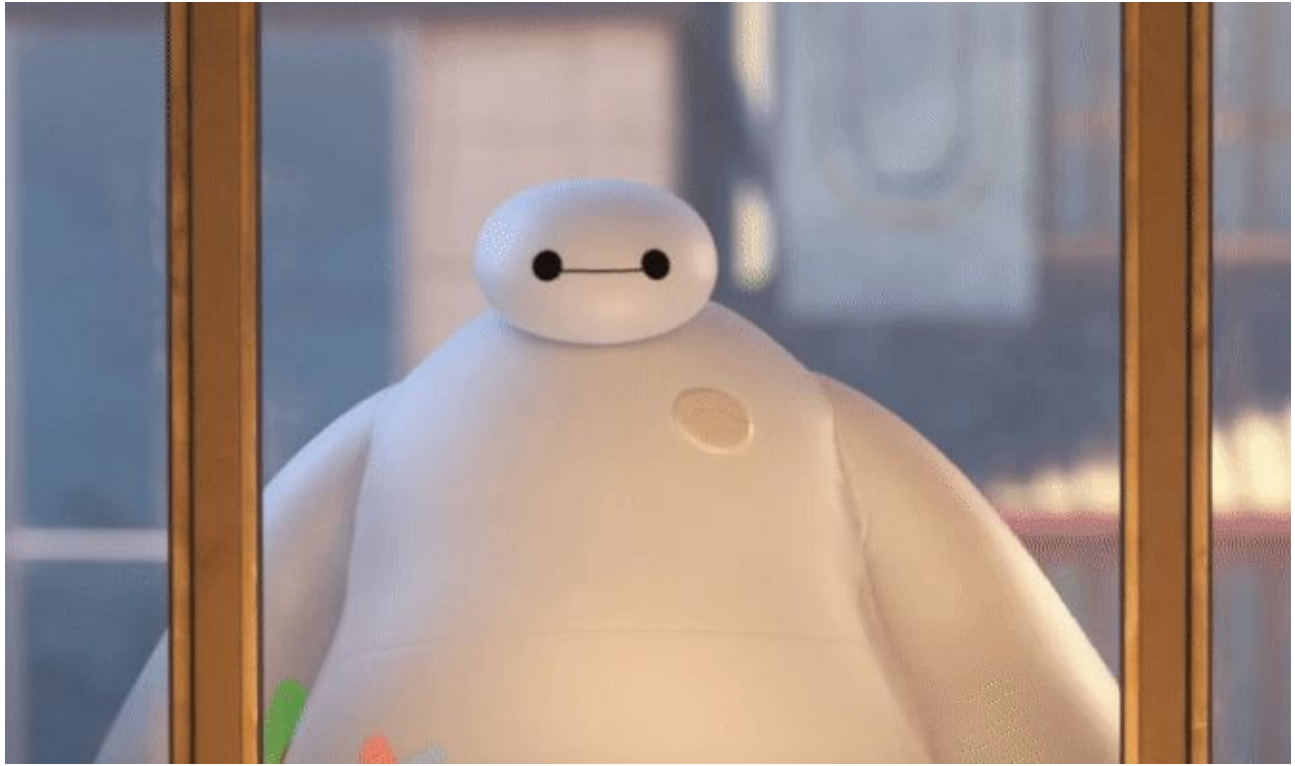
Introduction!

CSE 4883: Digital Image Processing – Finals Week 1

Lec Raiyan Rahman
Dept of CSE, UIU
raiyan@cse.uiu.ac.bd

Hello and Good Morning!



Hello and Good Morning!

A Little Introduction ☺️

Lec Raiyan Rahman
Dept of CSE, UIU & Guest Faculty, BUP

- Lecturer (Feb 2019 – Jul 2023)
  Department of Computer Science and Engineering
  Military Institute of Science and Technology (MIST), Dhaka

- M.Sc. (ongoing) and B.Sc. in Computer Science and Engineering
  Military Institute of Science and Technology (MIST)

Profile: https://cse.uiu.ac.bd/profiles/rahman-raiyan/

For any **queries**, send an email to: raiyan@cse.uiu.ac.bd or visit my office (Room 719D) during counseling hours.

Weekly Schedule (counseling hour): Click Here

A Little Introduction, your turn ☺️

Introduce yourself with:

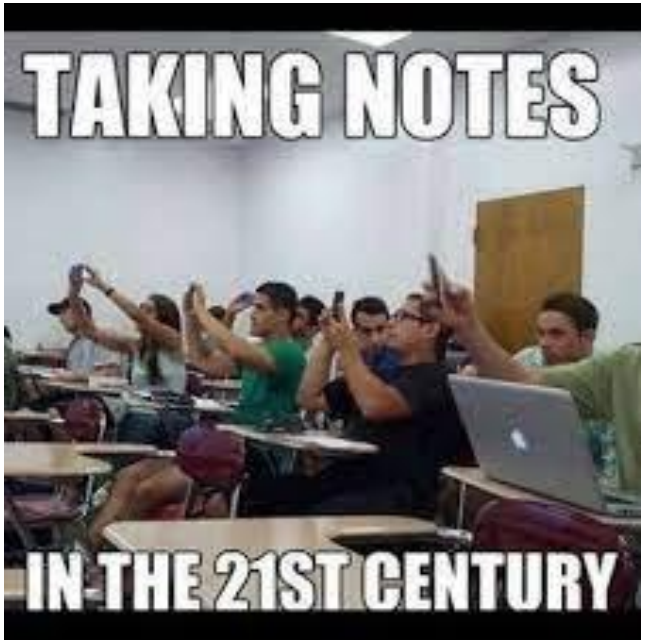
1. Your name
2. CGPA (no, kidding)
3. What you are liking the most in UIU.
4. And, what you dislike the most.

Welcome to DIP – Finals!

Before we start, here’s a (hopeful) promise:

If you follow my class closely, you will NOT need anything else for this course*.

*But



1. Pay attention and actually take notes, not pics.

2. Don’t talk during the lectures.

*But

3. Bring a bed if you want to sleep. Promise, I'll let you.

*But

4. Be present in class. For absence due to valid reasons, send an email within the class day to raiyan@cse.uiu.ac.bd.

Now, on with the classes!

An Overview of the ML Pipeline: Towards Using ML in DIP

CSE 4883: Digital Image Processing – Finals Week 1

Lec Raiyan Rahman
Dept of CSE, UIU
raiyan@cse.uiu.ac.bd

What is ‘learning’?

Think of a child and what messes they get themselves into!

What is ‘learning’?

What is ‘learning’?

But in time we learn, and become smart and responsible adults!

What is ‘learning’?

In time we learn:

- Where the edge of the bed is.
- Putting your fingers in electric sockets might not be a good idea!
- And much more!

What is ‘learning’?

Now the question is,

Can we make machines learn how human babies learn?

Well, no.

But they can, in fact, **learn**.

What is ‘learning’?

Machines learn using data. But -

- We have to **present the data** in machine readable format.
- We have to **point them** to what they’re looking for.
- We even have to tell them how they should look at and **how to learn** from that data.
- Then, how they should **communicate** and **interpret** the results of their learning.

The Potential of ML

But once we get machines to ‘think’ and take decisions, the potential can be amazing!
From predictive business analysis to genome analysis…

The Potential of ML

To power grid automation/efficiency to smart, connected city services.

We can solve a lot of the world's problems!

The Potential of ML

or, a popular plot of many movies,
maybe create evil sentient robot takeover on the process!

But jokes aside, let’s see how we can make computers learn with ML using the ML pipeline.

Problem 1 - Music Choice Prediction

Let’s take something most of us do on a daily basis - listening to music!

Although we all do it, we usually have very distinct choices on what music we like.

More importantly, it’s actually quite surprising how easy it is to predict choices given we know some factors about a person.

Problem 1 - Music Choice Prediction

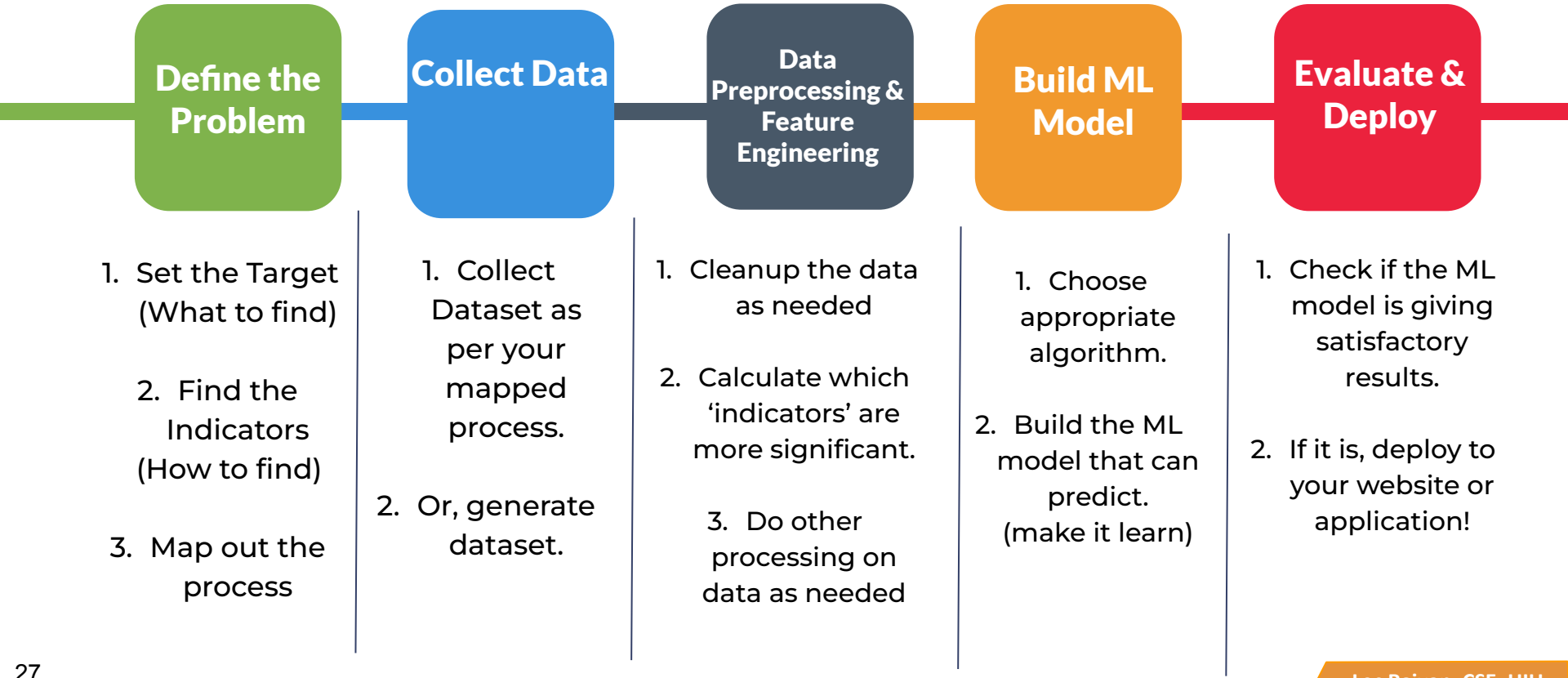
Recommending products to customers is one of the prime uses of ML.

For instance, think of Youtube video recommendation system!

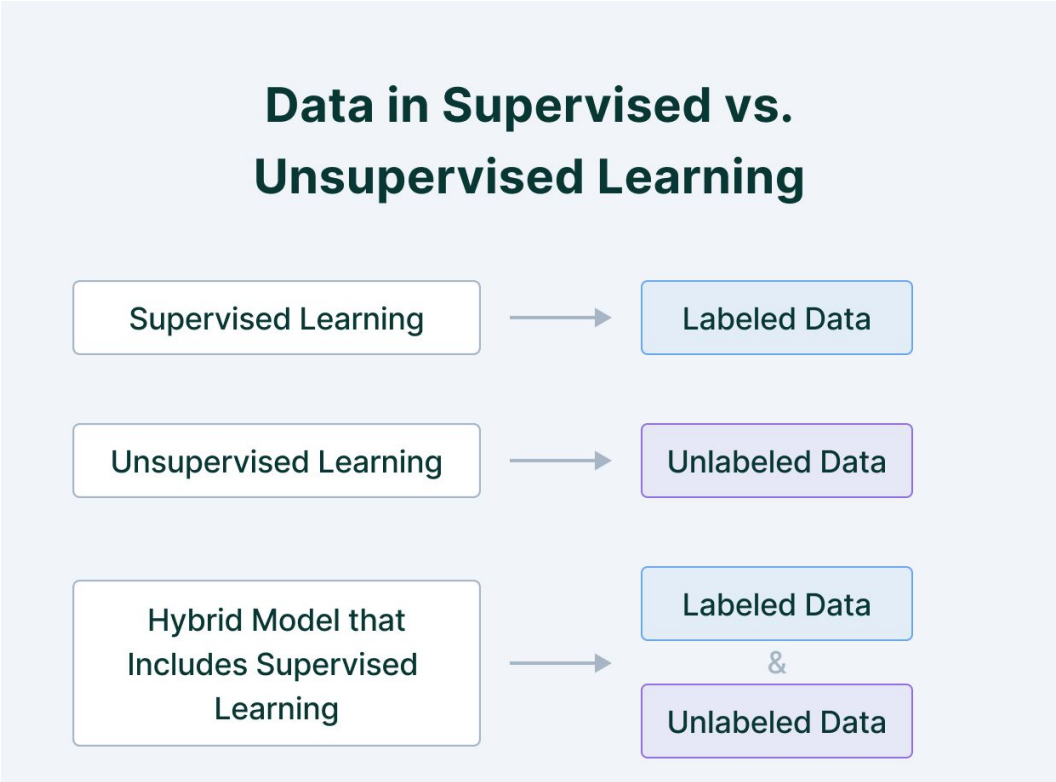
Let's try **building our own recommendation** system.

But, let's do it with the **simplest possible dataset** for starters.

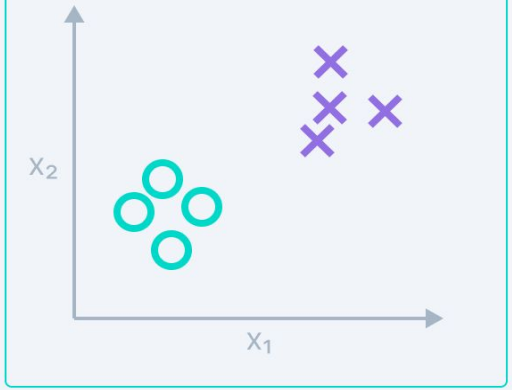
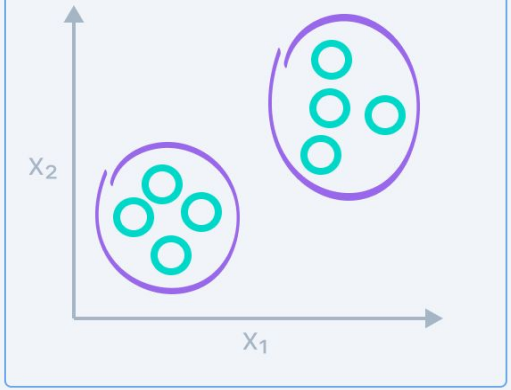
Overview of the ML Pipeline



Supervised Learning



Supervised vs Unsupervised Learning

Supervised learning	Unsupervised learning
Input data is labeled	Input data is unlabeled
Has a feedback mechanism	Has no feedback mechanism
Data is classified based on the training dataset	Assigns properties of given data to classify it
	

Classification vs Regression

enjoy algorithms

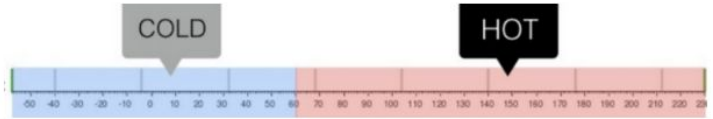
Regression

What will be the temperature tomorrow?

Fahrenheit

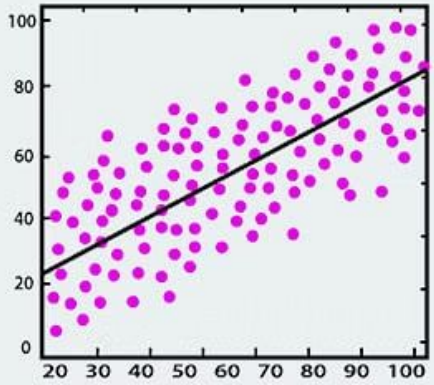
Classification

Will it be hot or cold tomorrow?



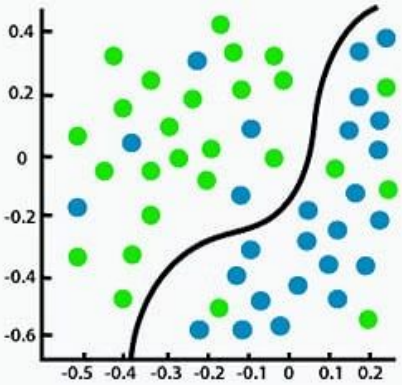
Fahrenheit

Classification vs Regression



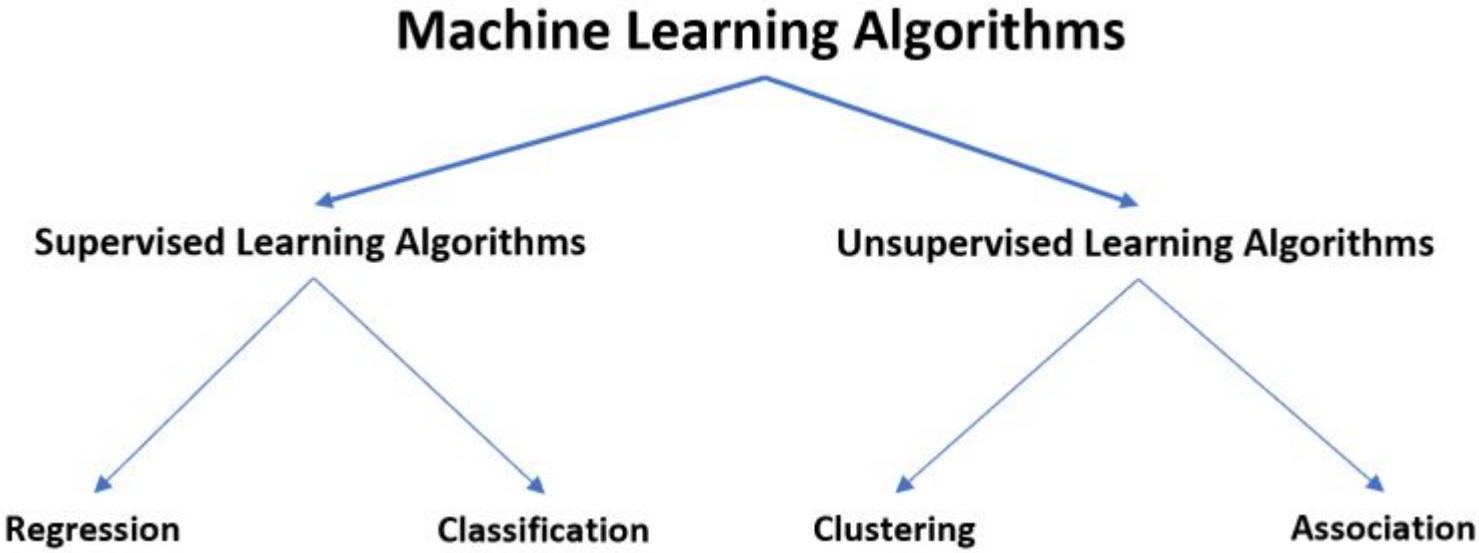
Regression

versus

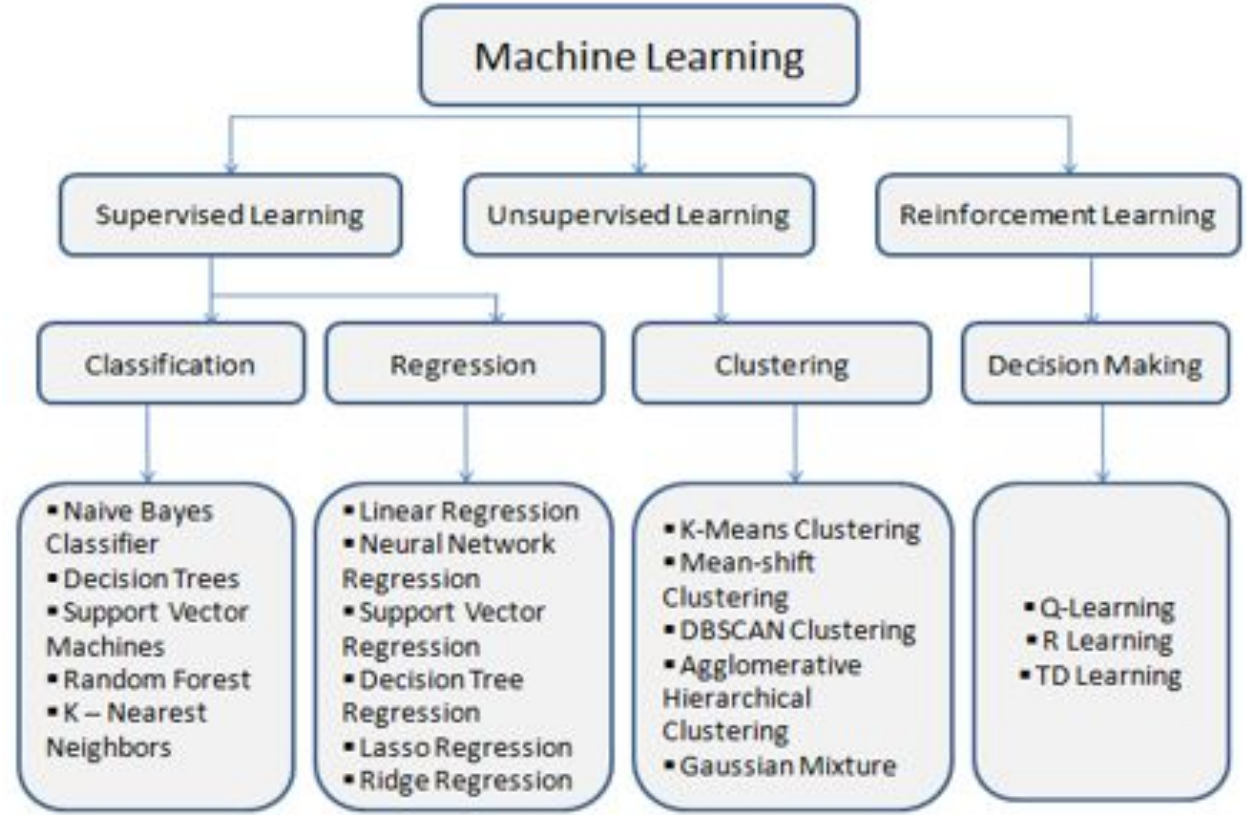


Classification

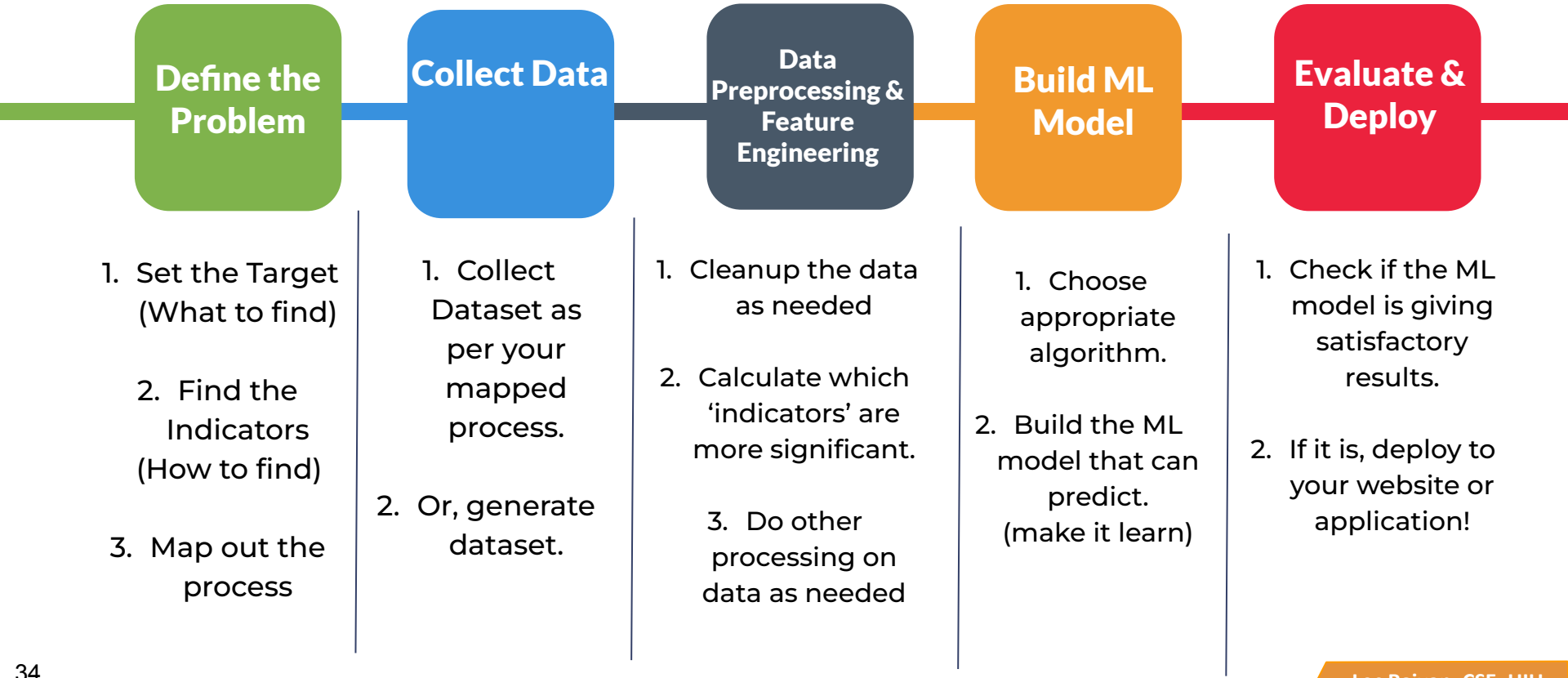
Classification vs Regression - ML



ML Algorithms - Classification



Overview of the ML Pipeline



Problem 1 - Music Choice Prediction

Implementation

Introduction to Artificial Neural Network and Using in DIP

CSE 4883: Digital Image Processing – Finals Week 1

Lec Raiyan Rahman
Dept of CSE, UIU
raiyan@cse.uiu.ac.bd

A Little Inspiration from Biology!

Outline

- Learning
- Forms of Learning
- Supervised Learning
- Artificial Neural Network
- Activation Functions and Performance Metrics
- Code!

What is “Learning” in AI?

An agent is **learning** if it improves its performance on future tasks after making observations about the world.

In which we describe agents that can improve their behavior through diligent study of their own experiences.

Forms of Learning

Any component of an agent can be improved by learning from **data**.

The improvements depend on four major factors:

- ✔ Which *component* is to be improved.
- ✔ What prior knowledge the agent already has.
- ✔ What representation is used for the data and the component.
- ✔ What feedback is available to learn from.

Forms of Learning Based on Feedback

There are three types of feedback that determine the three main types of learning:

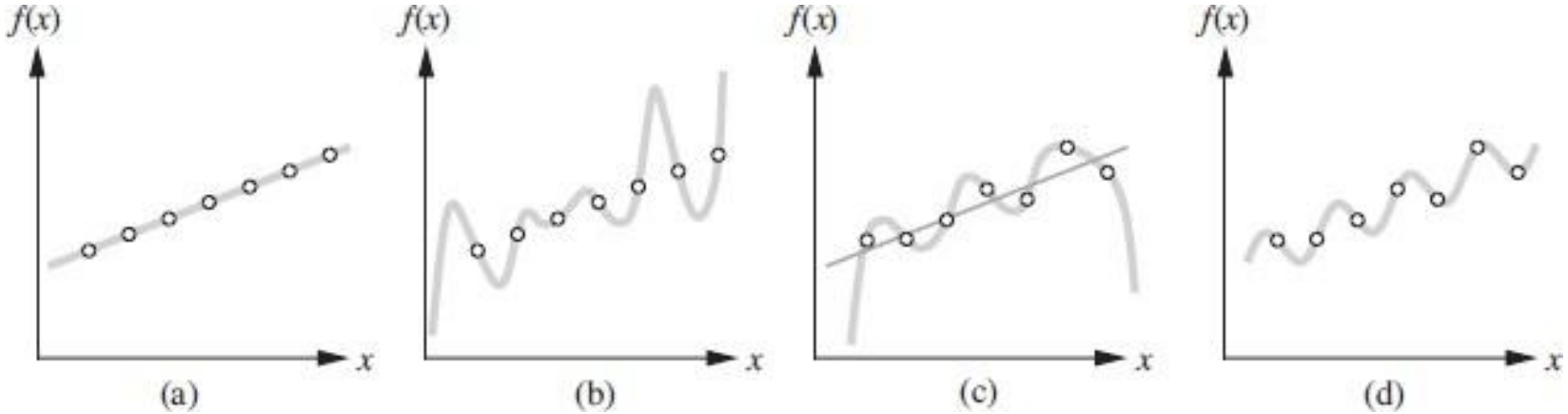
- Unsupervised learning
  - ✔ Agent learns patterns in the input even though no explicit feedback is supplied
  - ✔ Clustering – concept of "good traffic days" and "bad traffic days"
- Reinforcement learning
  - ✔ Agent learns from a series of reinforcements—rewards or punishments
  - ✔ Getting tips for taxi agent, winning point for chess game
- Supervised learning
  - ✔ Agent observes some example input–output pairsand learns a function that maps from input to output
  - ✔ Semi-supervised learning - a few labeled examples and a large collection of unlabeled examples

Supervised Learning

- **Training Set –** a set of N example input–output pairs
- **Hypothesis –** a possible function for generating output
- **Test Set –** a set of examples that are distinct from the training set
- **Learning –** a search through the space of possible hypotheses for one that will perform well

Supervised Learning

- **Hypothesis –** a possible function for generating output

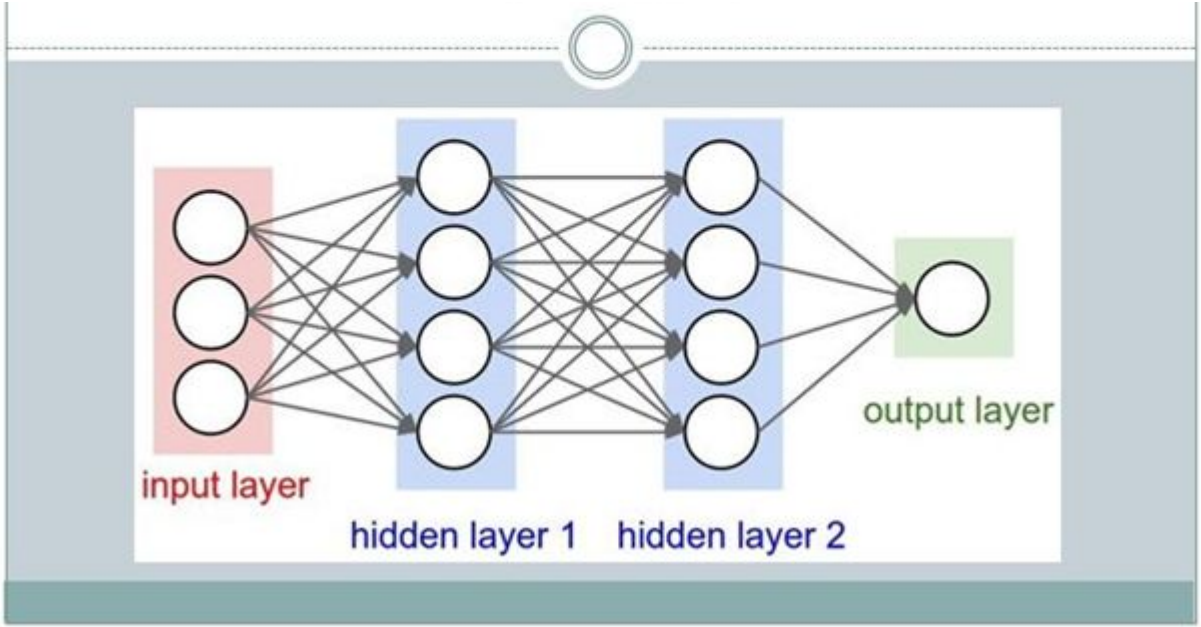


Supervised Learning..

- **Classification** – When the output y is one of a finite set of values (such as sunny, cloudy or rainy), the learning problem is called classification, and is called Boolean or binary classification if there are only two values.

- **Regression** – When y is a number (such as tomorrow's temperature), the learning problem is called regression.

What is ANN?



What is ‘Artificial neural network’?

- Artificial neural networks are one of the main tools used in machine learning.
- As the “neural” part of their name suggests, they are brain-inspired systems that are intended to replicate the way that we humans learn.
- Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns that are far too complex or numerous for a human programmer to extract and teach the machine to recognize.

Artificial Neural Network

Input layer: Number of neurons in this layer corresponds to the number of inputs to the neuronal network.

Hidden layer: This layer has arbitrary number of layers with arbitrary number of neurons.

Output layer: The number of neurons in the output layer corresponds to the number of the output values of the neural network.

Artificial Neural Network

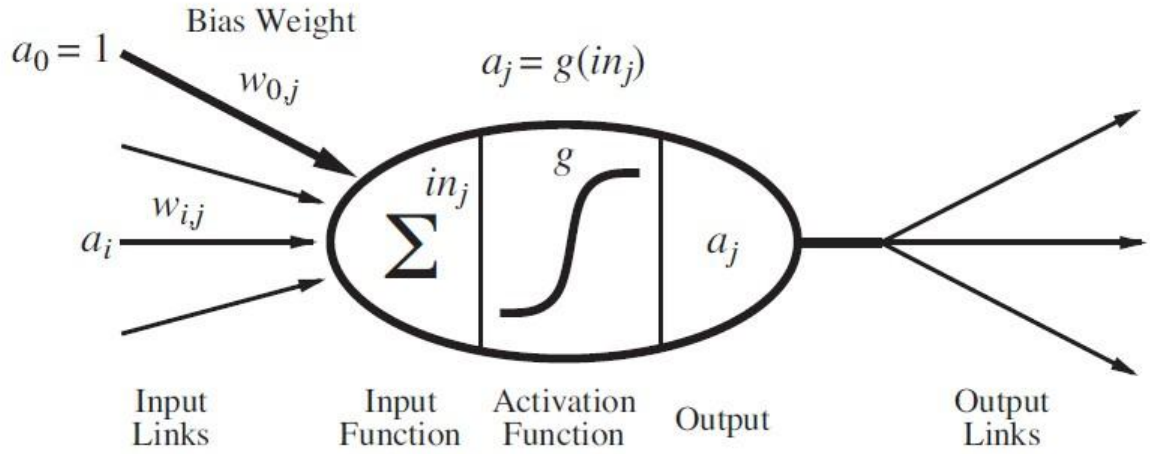
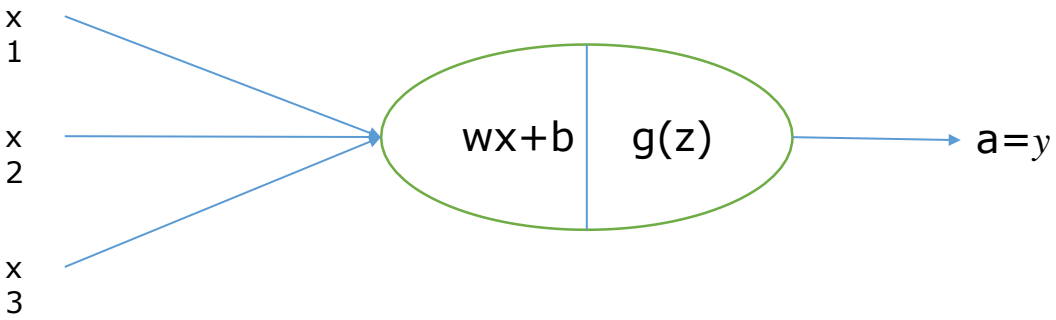


Figure 18.19 A simple mathematical model for a neuron. The unit's output activation is $a_j = g(\sum_{i=0}^{n} w_{i,j} a_i)$, where a_i is the output activation of unit i and $w_{i,j}$ is the weight on the link from unit i to this unit.

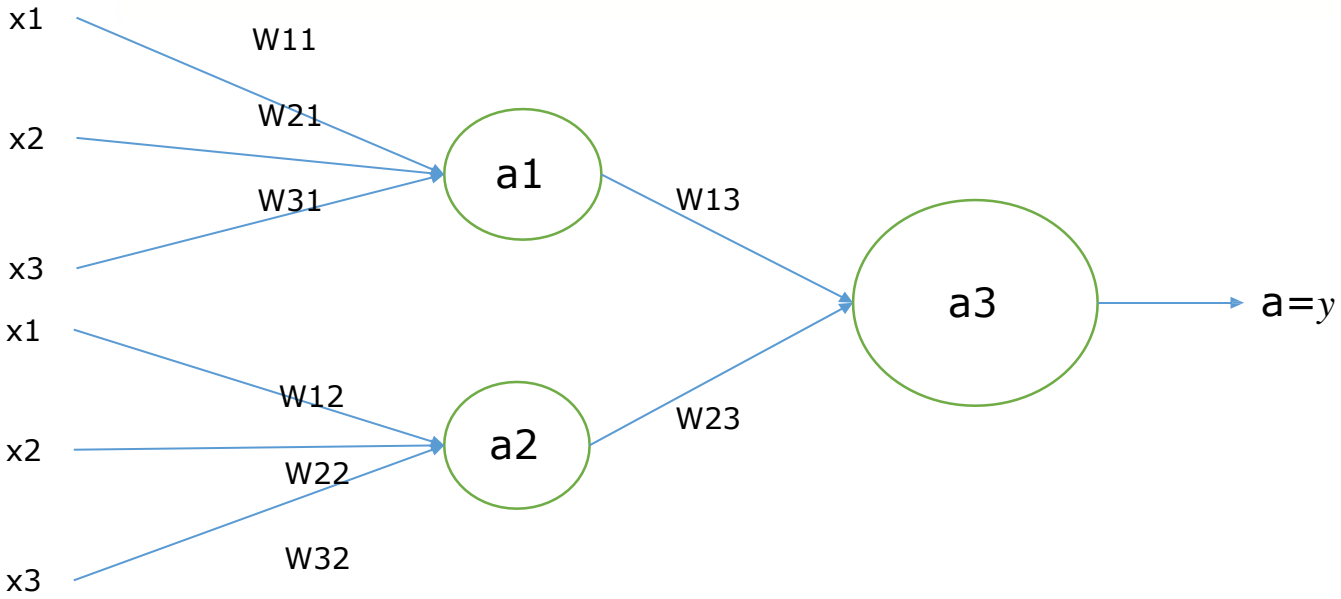
Artificial Neural Network

- Units
- Links
- Weight
- Activation
- Activation function

Artificial Neural Network



Artificial Neural Network

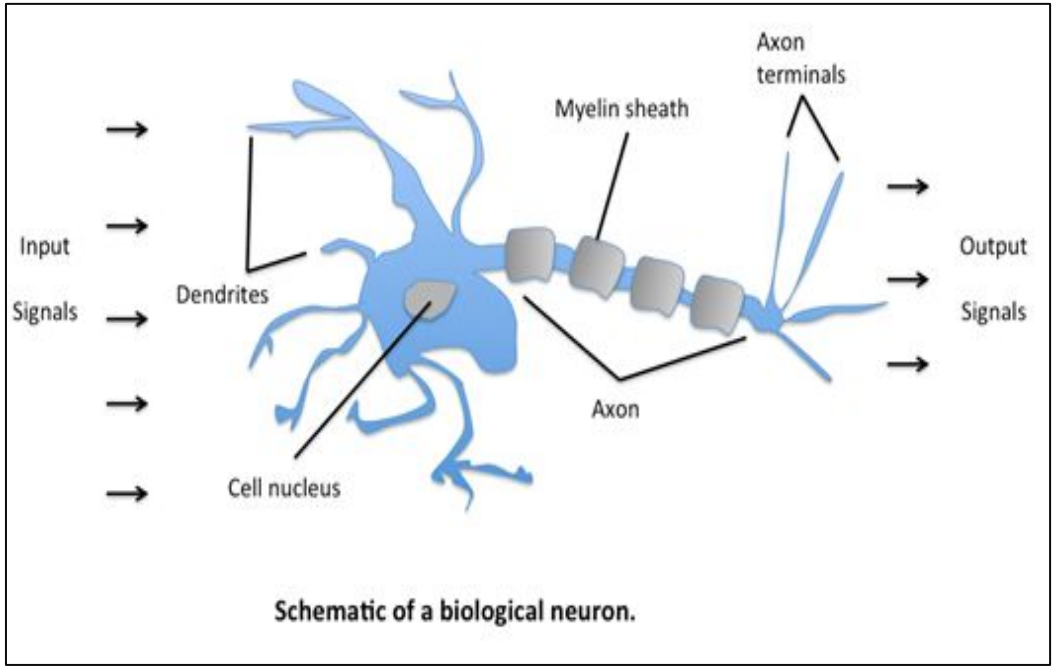


ANN - Activation Functions

1. tanh
2. Sigmoid
3. Relu
4. Leaky Relu
5. Softmax
6. Identity

Basic Structure of ANN

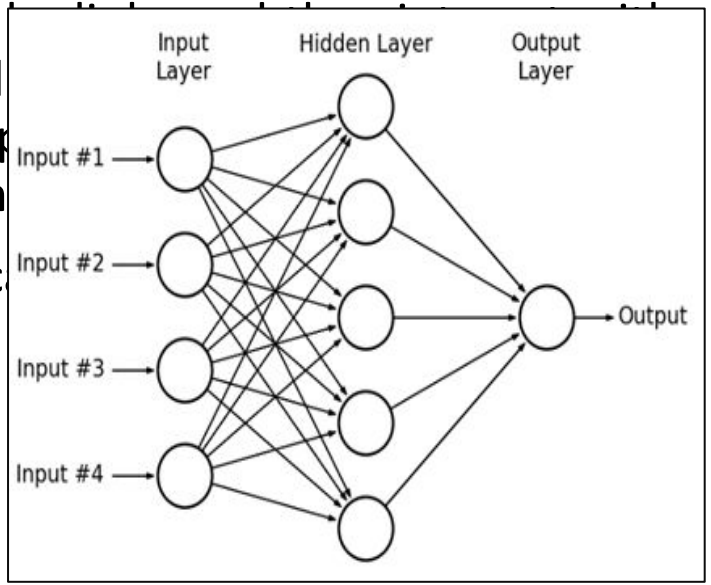
- The idea of ANNs is based on the belief that working of the human brain by making the right connections can be imitated using silicon and wires as living neurons and dendrites.



Schematic of a biological neuron.

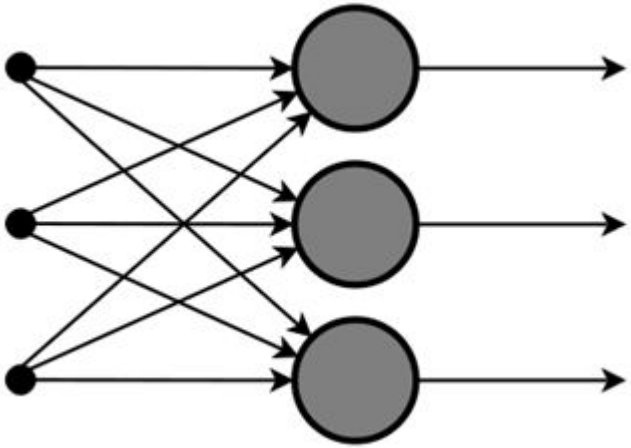
Basic Structure of ANN

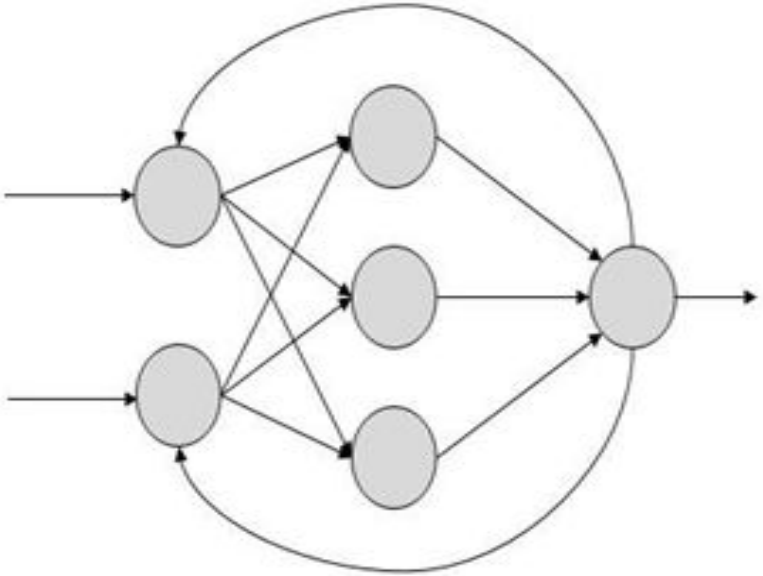
- ANNs are composed of multiple **nodes**, which imitate biological **neurons** of the human brain. The neurons are connected
each other. The nodes can take input data and
the data. The result of these operations is
output at each node is called its **activation** or **n**
- Each link is associated with **weight.** ANNs are c
place by altering weight values



Types of Artificial Neural Networks

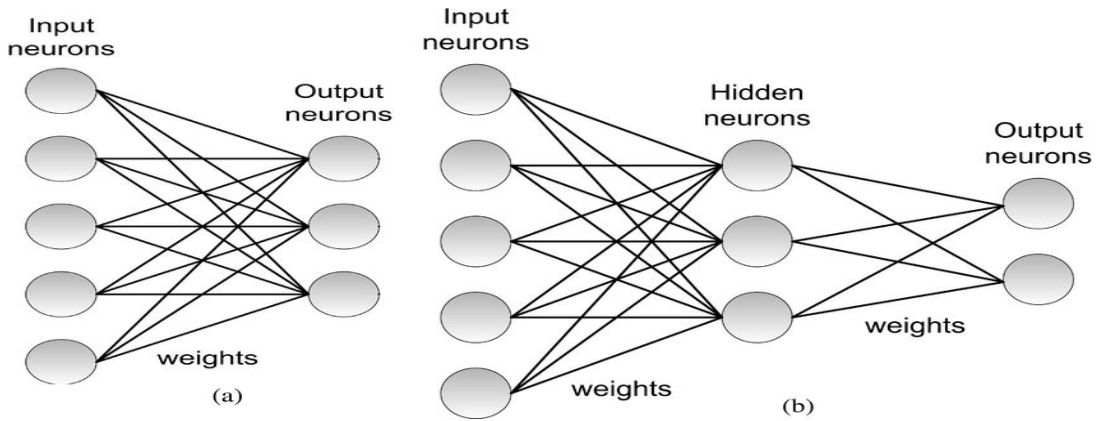
- There are two Artificial Neural Network topologies − **FeedForward** and **Feedback.**





Types of Artificial Neural Networks

- Considering the layer, there are two Artificial Neural Network topologies: A **single-layer perceptron** and a **multi-layer perceptron**.
- A single-layer perceptron is more suitable for simple and linear problems, while a multi-layer perceptron is more suitable for complex and non-linear problems.



Activation Functions and their Types

What is Activation Function?

- It's just a function that you use to get the output of the node. It is also known as **Transfer Function**.
- *Their main purpose is to convert an input signal of a node in an ANN to an output signal.* That output signal now is used as an input in the next layer in the stack.
- Specifically in A-NN we do the sum of products of inputs(**X**) and their corresponding Weights (**W**) and apply an Activation function **f(x)** to it to get the output of that layer and feed it as an input to the next layer.

It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function).

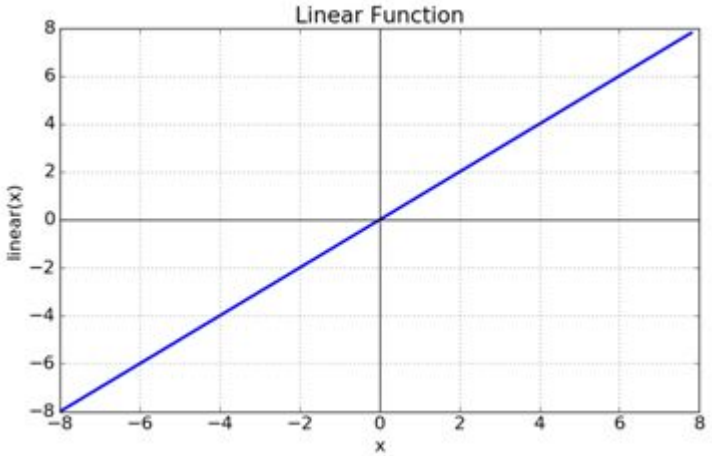
The Activation Functions can be based on 2 types-

- Linear Activation Function
- Non-linear Activation Functions

Linear Activation Functions

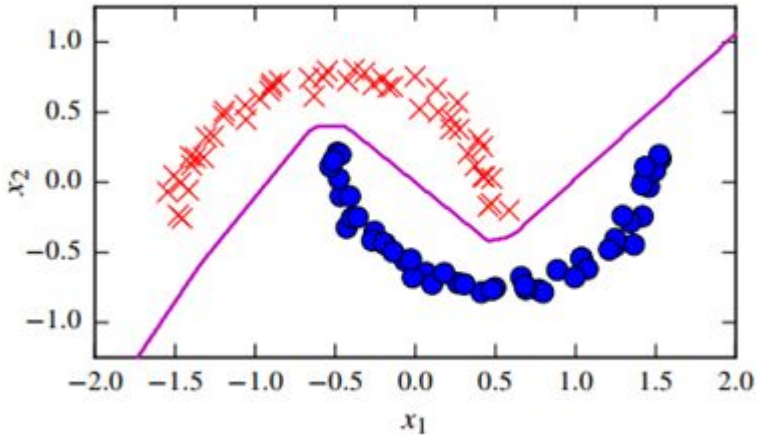
As you can see the function is a line or linear. Therefore, the output of the functions will not be confined between any range.

- **Equation:** f(x) = x
- **Range:** (-infinity to infinity)



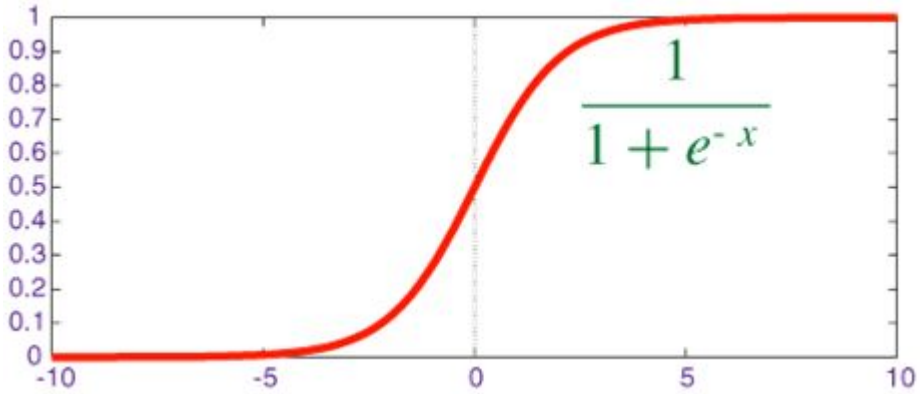
Non-linear Activation Functions

- The Nonlinear Activation Functions are the most used activation functions. Nonlinearity helps to makes the graph look something like this
- It makes it easy for the model to generalize or adapt to a variety of data and to differentiate between the outputs.



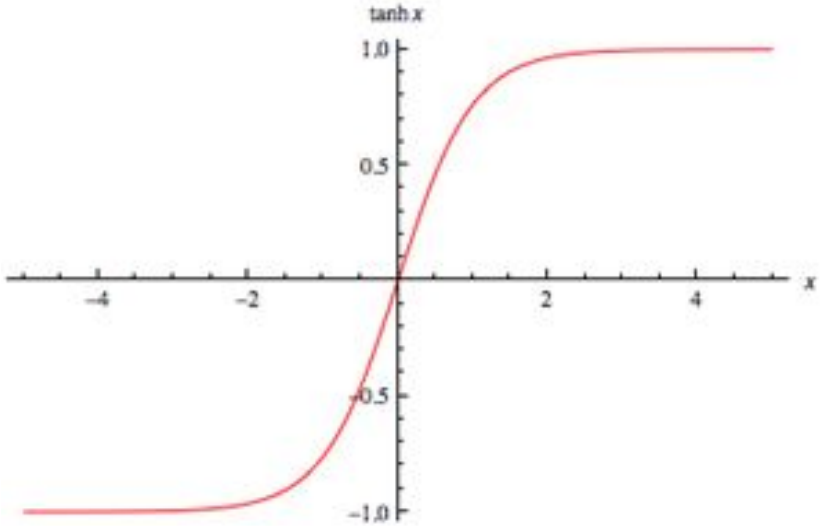
Sigmoid Activation Function

- The main reason why we use the sigmoid function is that it exists between **(0 to 1).** Therefore, it is especially used for models where we have to **predict the probability** as an output. Since the probability of anything exists only between the range of **0 and 1,** sigmoid is the right choice.



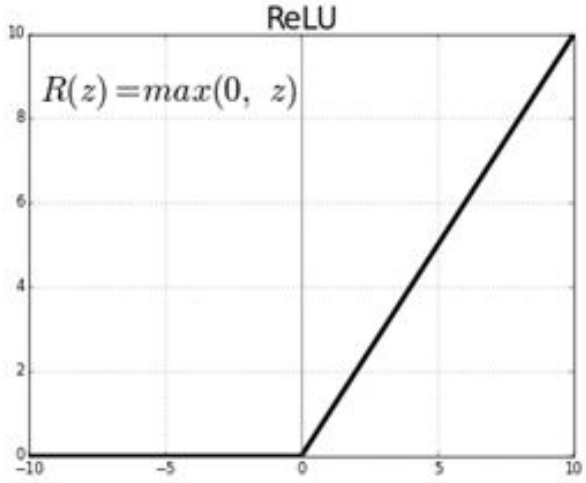
Tanh Activation Function

- tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s-shaped).
- Both tanh and logistic sigmoid activation functions are used in feed-forward nets.

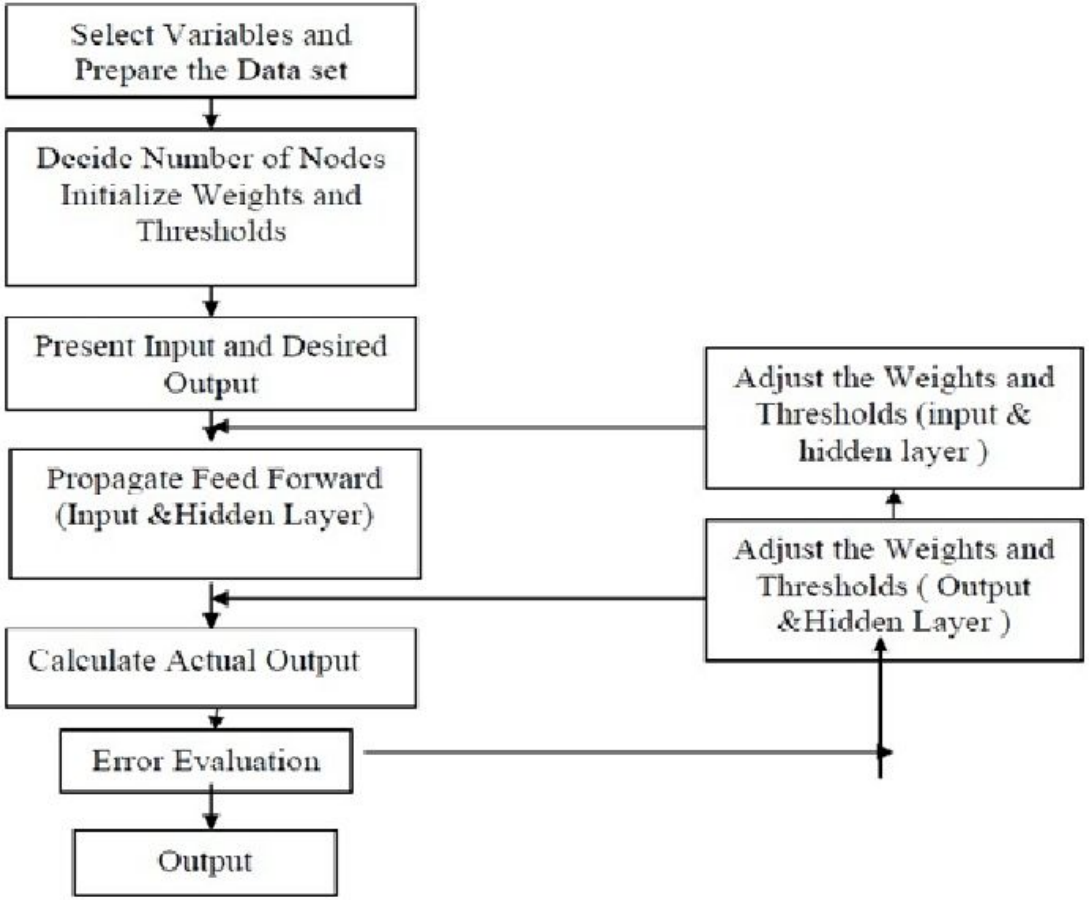


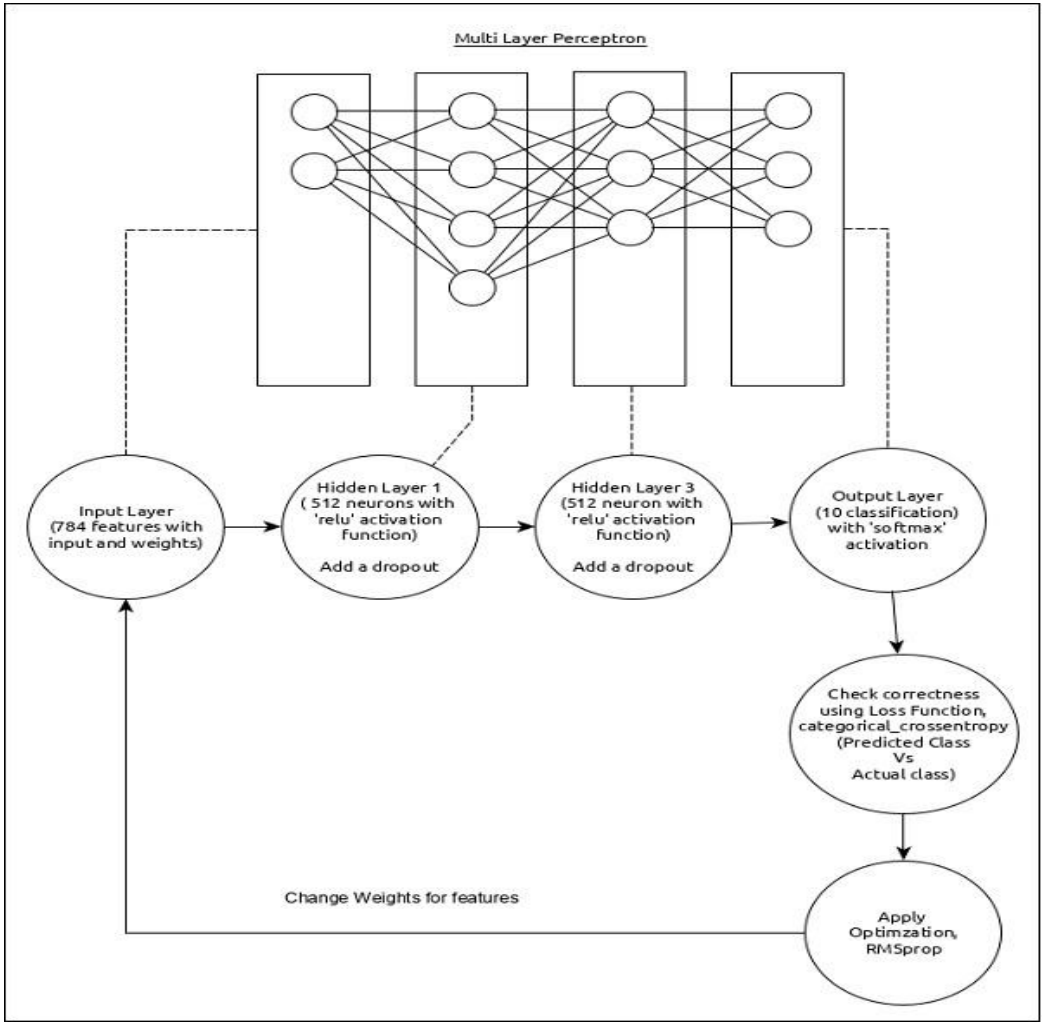
ReLU (Rectified Linear Unit) Activation Function

- The ReLU is the most used activation function in the world right now. Since it is used in almost all the convolutional neural networks or deep learning.
- As you can see, the ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.
- **Range:** [0 to infinity)



ANN Methodology

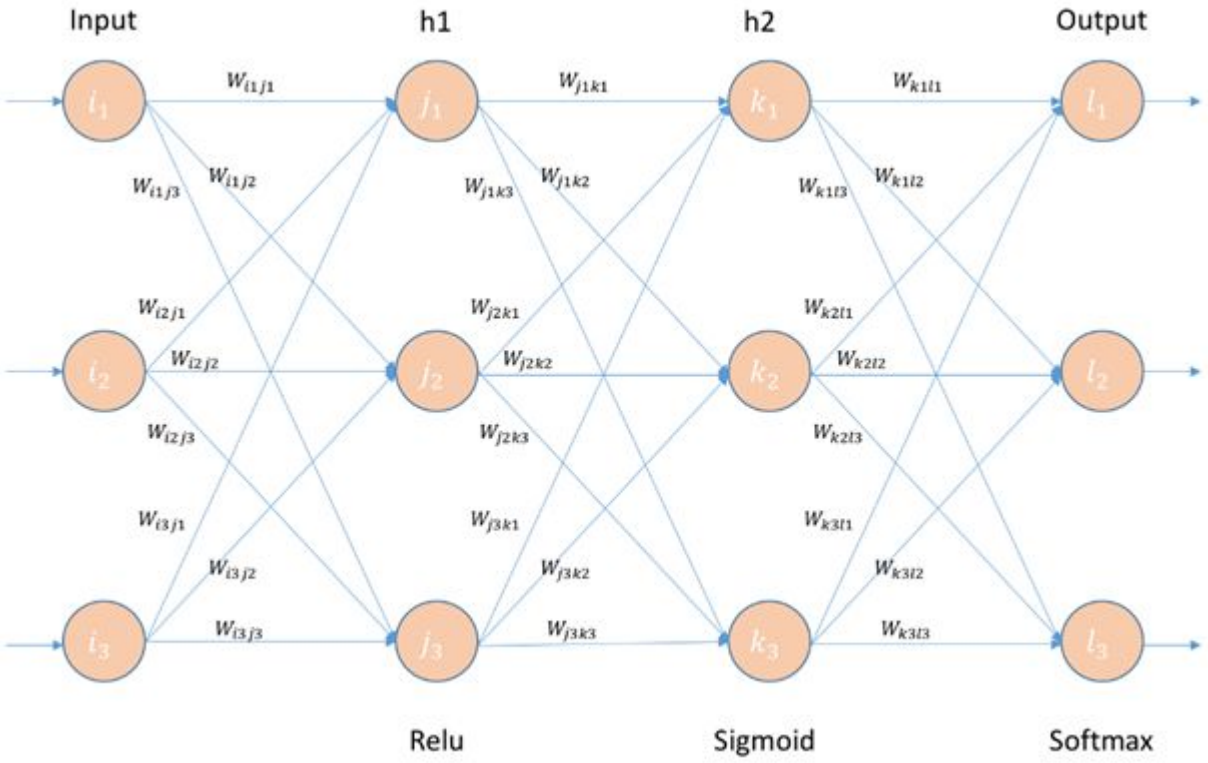



Multi Layer Perceptron
Input Layer
(784 features with
input and weights)
Hidden Layer 1
(512 neurons with
'relu' activation
function)
Add a dropout
Hidden Layer 3
(512 neuron with
'relu' activation
function)
Add a dropout
Output Layer
(10 classification)
with 'softmax'
activation
Check correctness
using Loss Function,
categorical_crossentropy
(Predicted Class
Vs
Actual class)
Apply
Optimzation,
RMSprop
Change Weights for features

ANN Methodology

- **Step 1:** Randomly initialize the Weights to a small number 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. Propagates 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 left to right, the error is back-Propagated. Update the weights according to how much they are responsible for the error. The Learning rate decides how much we update the weights.
- **Step 6:** Repeat step 1 to step 5 and updates the weights after each observation.
- **Step 7:** When the whole training set passed through the ANN that makes an epoch. Redo more epoch.

ANN Architecture



ANN Architecture

Inputs

$$\text{Inputs} = [i_1 \quad i_2 \quad i_3]$$

$$= [0.1 \quad 0.3 \quad 0.5]$$

Actual Outputs

$$\text{Outputs} = [y_1 \quad y_2 \quad y_3]$$

$$= [0.0 \quad 1.0 \quad 0.0]$$

Randomly initialize the Weights to small number close to 0 (but not (

Weights for Layer 1

$$W_{ij} = \begin{bmatrix} W_{i1j1} & W_{i1j2} & W_{i1j3} \\ W_{i2j1} & W_{i2j2} & W_{i2j3} \\ W_{i3j1} & W_{i3j2} & W_{i3j3} \end{bmatrix}$$

$$W_{ij} = \begin{bmatrix} 0.2 & 0.4 & 0.5 \\ 0.3 & 0.7 & 0.1 \\ 0.2 & 0.6 & 0.9 \end{bmatrix}$$

Weights for Layer 2

$$W_{jk} = \begin{bmatrix} W_{j1k1} & W_{j1k2} & W_{j1k3} \\ W_{j2k1} & W_{j2k2} & W_{j2k3} \\ W_{j3k1} & W_{j3k2} & W_{j3k3} \end{bmatrix}$$

$$W_{jk} = \begin{bmatrix} 0.4 & 0.6 & 0.1 \\ 0.2 & 0.7 & 0.3 \\ 0.8 & 0.1 & 0.5 \end{bmatrix}$$

Weights for Layer 3

$$W_{kl} = \begin{bmatrix} W_{k1l1} & W_{k1l2} & W_{k1l3} \\ W_{k2l1} & W_{k2l2} & W_{k2l3} \\ W_{k3l1} & W_{k3l2} & W_{k3l3} \end{bmatrix}$$

$$W_{kl} = \begin{bmatrix} 0.2 & 0.5 & 0.3 \\ 0.4 & 0.6 & 0.8 \\ 0.1 & 0.9 & 0.7 \end{bmatrix}$$

ANN Architecture

Feedfarwad Neural Network: 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. Propagates the activations until getting the predicted result y.

Layer 1 Matrix Operation

$$= [i_1 \quad i_2 \quad i_3] \times \begin{bmatrix} W_{i1j1} & W_{i1j2} & W_{i1j3} \\ W_{i2j1} & W_{i2j2} & W_{i2j3} \\ W_{i3j1} & W_{i3j2} & W_{i3j3} \end{bmatrix} + [b_{j1} \quad b_{j2} \quad b_{j3}]$$

$$= [0.1 \quad 0.3 \quad 0.5] \times \begin{bmatrix} 0.2 & 0.4 & 0.5 \\ 0.3 & 0.7 & 0.1 \\ 0.2 & 0.6 & 0.9 \end{bmatrix} + [1.0 \quad 1.0 \quad 1.0]$$

$$[h1_{in1} \quad h1_{in2} \quad h1_{in3}] = [1.2 \quad 1.6 \quad 1.5]$$

ANN Architecture

RELU operation on layer 1

$$\text{RELU} = \max(0, x)$$

$$[h1_{out1} \quad h1_{out2} \quad h1_{out3}] = [\max(0, h1_{in1}) \quad \max(0, h1_{in2}) \quad \max(0, h1_{in3})]$$

$$[h1_{out1} \quad h1_{out2} \quad h1_{out3}] = [1.2 \quad 1.6 \quad 1.5]$$

Layer 2 Matrix Operation

$$= [h1_{out1} \quad h1_{out2} \quad h1_{out3}] \times \begin{bmatrix} W_{j1k1} & W_{j1k2} & W_{j1k3} \\ W_{j2k1} & W_{j2k2} & W_{j2k3} \\ W_{j3k1} & W_{j3k2} & W_{j3k3} \end{bmatrix} + [b_{k1} \quad b_{k2} \quad b_{k3}]$$

$$= [1.2 \quad 1.6 \quad 1.5] \times \begin{bmatrix} 0.4 & 0.6 & 0.1 \\ 0.2 & 0.7 & 0.3 \\ 0.8 & 0.1 & 0.5 \end{bmatrix} + [1.0 \quad 1.0 \quad 1.0]$$

$$[h2_{in1} \quad h2_{in2} \quad h2_{in3}] = [3.0 \quad 3.0 \quad 2.4]$$

Sigmoid Operation on layer 2

$$\text{Sigmoid} = 1 / (1 + e^{-x})$$

$$[h2_{out1} \quad h2_{out2} \quad h2_{out3}] = [1/(1 + e^{-h2_{in1}}) \quad 1/(1 + e^{-h2_{in2}}) \quad 1/(1 + e^{-h2_{in3}})]$$

$$[h2_{out1} \quad h2_{out2} \quad h2_{out3}] = [1/(1 + e^{-3.0}) \quad 1/(1 + e^{-3.0}) \quad 1/(1 + e^{-2.4})]$$

$$[h2_{out1} \quad h2_{out2} \quad h2_{out3}] = [0.952 \quad 0.952 \quad 0.916]$$

Layer 3 Matrix Operation

$$= [h2_{out1} \quad h2_{out2} \quad h2_{out3}] \; \mathbf{X} \begin{bmatrix} W_{k1l1} & W_{k1l2} & W_{k1l3} \\ W_{k2l1} & W_{k2l2} & W_{k2l3} \\ W_{k3l1} & W_{k3l2} & W_{k3l3} \end{bmatrix} + [b_{l1} \quad b_{l2} \quad b_{l3}]$$

$$= [0.952 \quad 0.952 \quad 0.916] \; \mathbf{X} \begin{bmatrix} 0.2 & 0.5 & 0.3 \\ 0.4 & 0.6 & 0.8 \\ 0.1 & 0.9 & 0.7 \end{bmatrix} + [1.0 \quad 1.0 \quad 1.0]$$

$$[o_{in1} \quad o_{in2} \quad o_{in3}] = [1.7 \quad 2.9 \quad 2.7]$$

Softmax operation on layer 3

$$\text{Softmax} = e^{o_{ina}} / (\sum_{a=1}^{3} e^{o_{ina}})$$

$$[o_{out1} \quad o_{out2} \quad o_{out3}] = \left[e^{o_{ina}} / (\sum_{a=1}^{3} e^{o_{ina}}) \quad e^{o_{ina}} / (\sum_{a=1}^{3} e^{o_{ina}}) \quad e^{o_{ina}} / (\sum_{a=1}^{3} e^{o_{ina}}) \right]$$

$$[o_{out1} \quad o_{out2} \quad o_{out3}] = [0.1420 \quad 0.4717 \quad 0.3862]$$

The actual output is [0.0 1.0 0.0] but we got [0.1420 0.4717 0.3862]

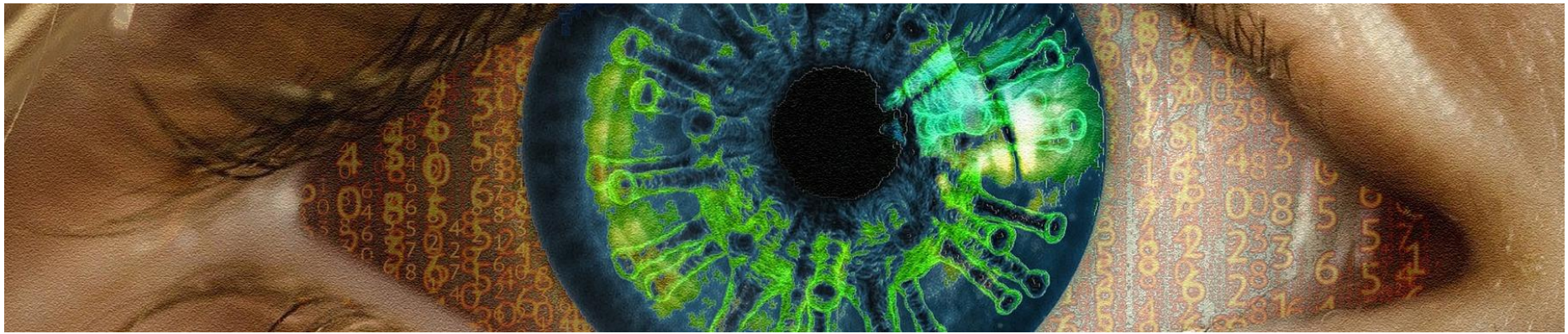
We have done the forward pass

Compare the predicted result to the actual result. Measure the generated error.

Logistic Loss

$$\text{Log Loss} = -y_i * \log(o_{outi}) - (1 - y_i) * \log(1 - o_{outi})$$

Cataract Detection



Kaggle Notebook

Practical Implementation of Artificial Neural Network

- In this part, you will be solving a data analytics challenge for a bank. You will be given a dataset with a large sample of the bank's customers. To make this dataset, the bank gathered information such as customer id, credit score, gender, age, tenure, balance, if the customer is active, has a credit card, etc. During 6 months, the bank observed if these customers left or stayed in the bank.

- Your goal is to make an Artificial Neural Network that can predict, based on geo-demographical and transactional information given above, if any individual customer will leave the bank or stay (customer churn). Besides, you are asked to rank all the customers of the bank, based on their probability of leaving. To do that, you will need to use the right Deep Learning model, one that is based on a probabilistic approach.

- If you succeed in this project, you will create significant added value to the bank. By applying your Deep Learning model the bank may significantly reduce customer churn.

Dataset Sample

CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.88	1
15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.57	1
15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.1	0
15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	0	149756.71	1
15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
15656148	Obinna	376	Germany	Female	29	4	115046.74	4	1	0	119346.88	1
15792365	He	501	France	Male	44	4	142051.07	2	0	1	74940.5	0
15592389	H?	684	France	Male	27	2	134603.88	1	1	1	71725.73	0
15767821	Bearce	528	France	Male	31	6	102016.72	2	0	0	80181.12	0

Import the Libraries

- In this step, we import three Libraries in Data Preprocessing part. A library is a tool that you can use to make a specific job. First of all, we import the numpy library used for multidimensional array then import the pandas library used to import the dataset and in last we import matplotlib library used for plotting the graph.

```
# import the Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Import the dataset

- In this step, we import the dataset to do that we use the pandas library. After import our dataset we define our dependent and independent variable. Our independent variables are 1 to 12 attributes as you can see in the sample dataset which we call ‘X’ and dependent is our last attribute which we call ‘y’ here.

```
# import the datset
dataset = pd.read_csv('Churn_Modelling.csv')
X = dataset.iloc[: , 3:13].values
y = dataset.iloc[: , 13].values
```

Encoding the Categorical data

- In this step, we Encode our categorical data. If we see our dataset then Geography & Gender attribute is in Text and we Encode these two attributes in this part use the LabelEncoder and OneHOTEncoder from the Sklearn.processing library.

```
# Encoding the categorical data
from sklearn.preprocessing import LabelEncoder , OneHotEncoder
labelencoder_X1 = LabelEncoder()
X[:, 1] = labelencoder_X1.fit_transform(X[ : , 1])
labelencoder_X2 = LabelEncoder()
X[:, 2] = labelencoder_X2.fit_transform(X[: , 2])
onehotencoder = OneHotEncoder(categorical_features = [1])
X = onehotencoder.fit_transform(X).toarray()
X = X[:, 1:]
```

Split the dataset for test and train

In this step, we split our dataset into a test set and train set and an 80% dataset split for training and the remaining 20% for tests. Our dataset contains 10000 instances so 8000 data we train and 2000 for the test.

```
# split the dataset into test set and train 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)
```

Feature Scaling

- Feature Scaling is the most important part of data preprocessing. If we see our dataset then some attribute contains information in Numeric value some value very high and some are very low if we see the age and estimated salary. This will cause some issues in our machinery model to solve that problem we set all values on the same scale there are two methods to solve that problem first one is Normalize and Second is Standard Scaler.
- Here we use **standard Scaler** import from Sklearn Library.

Standardisation	Normalisation
$x_{stand} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$	$x_{norm} = \frac{x - \min(x)}{\max(x) - \min(x)}$

```
# feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Import the Libraries

In this step, we import the Library which will build our ANN model. We import Keras Library which will build a deep neural network based on Tensorflow because we use Tensorflow backhand. Here we import the two modules from Keras. **The first one is Sequential used for initializing our ANN model and the second is Dense used for adding different layers of ANN.**

```
# importing the Keras Libraries and packages
import keras
from keras.models import Sequential    # used for init our ANN model
from keras.layers import Dense         # used for different layer structure
```

Initialize our ANN model

In this step, we initialize our Artificial Neural Network model to do that we use sequential modules.

Most of the ANN also has layers in sequential order and the data flows from one layer to another layer in the given order until the data finally reaches the output layer.

```
# initializing the ANN model
classifier = Sequential()
```

Adding the input layer and first hidden layer

- In this step, we use the Dense model to add a different layer.
- The parameter which we pass here first is units=6 which defines hidden layer=6
- The third parameter is activation= 'relu' here in the first hidden layer we use relu activation.
- And the last parameter which we pass in dense function is input_dim= 11 which means the input node of our Neural Network is 11 because our dataset has 11 attributes that's why we choose 11 input nodes.

```
# Adding the input layer and the first hidden layer
classifier.add(Dense(units=6, activation='relu', input_dim=11))
# Adding dropout to prevent overfitting
classifier.add(Dropout(rate=0.1))
```

Building our Model

Adding the second hidden layer

- In this step, we add another hidden layer

```
# Adding the second hidden layer
classifier.add(Dense(6, activation='relu'))
# Adding dropout to prevent overfitting
classifier.add(Dropout(rate=0.1))
```

Adding the output layer

- In this step, we add an output layer in our ANN structure units = 1 which means one output node here we use the sigmoid function because our target attribute has a binary class which is one or zero that's why we use sigmoid activation function.

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

Optimizer

- Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate by comparing the prediction and the loss function, in order to reduce the losses. How you should change your weights or learning rates of your neural network to reduce the losses is defined by the optimizers you use.
- Keras provides quite a few optimizer as a module, optimizers and they are as follows:

- SGD,
- Adagrad,
- Adadelta,
- Adam,
- Adamax,
- Nadam

Metrics

- In machine learning, **Metrics** is used to evaluate the performance of your model. It is similar to loss function, Keras provides quite a few metrics as a module, metrics and they are as follows

- **accuracy, F1 Score**
- binary_accuracy
- categorical_accuracy
- sparse_categorical_accuracy
- top_k_categorical_accuracy
- sparse_top_k_categorical_accuracy
- cosine_proximity
- clone_metric

- Similar to loss function, metrics also accepts below two arguments −

- y_true − true labels as tensors
- y_pred − prediction with same shape as y_true

Compile the model

- Keras model provides a method, **compile()** to compile the model. The argument and default value of the **compile()** method is as follows
- The important arguments are as follows –

- loss function
- Optimizer
- metrics

Compile the model

- In this step, we compile the ANN.
- Here, the first parameter is **optimizer = 'Adam'.** Here, **Adam** is very efficient one, so that's why we use Adam optimizer here.
- The second parameter is loss this corresponds to loss function. Here we use **binary_crossentropy** because if we see target attribute our dataset which contains the binary value that's why we choose the binary cross-entropy.
- The final parameter is metrics basically It's a list of metrics to be evaluated by the model and here we choose the **accuracy** metrics.

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

Model Training

Models are trained by NumPy arrays using fit(). The main purpose of this fit function is used to evaluate your model on training. This can be also used for graphing model performance. It has the following syntax –

model.fit(*X, y, epochs = , **batch_size =)***

- At the end of the batch, the predictions are compared to the expected output variables and an error is calculated. From this error, the update algorithm is used to improve the model, e.g. move down along the error gradient.

- **Batch Gradient Descent**. Batch Size = Size of Training Set
- **Stochastic Gradient Descent**. Batch Size = 1
- **Mini-Batch Gradient Descent**. 1 < Batch Size < Size of Training Set

Model Training

model.fit(*X, y,* ***epochs = ,*** *batch_size =* ***)***

- A One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters. An epoch is comprised of one or more batches. For example, as above, an epoch that has one batch is called the batch gradient descent learning algorithm.
- You can think of a for-loop over the number of epochs where each loop proceeds over the training dataset. Within this for-loop is another nested for-loop that iterates over each batch of samples, where one batch has the specified “batch size” number of samples.
- The number of epochs is traditionally large, often hundreds or thousands, allowing the learning algorithm to run until the error from the model has been sufficiently minimized. You may see examples of the number of epochs in the literature and in tutorials set to 10, 100, 500, 1000, and larger.

Model Training

```
# Fitting the ANN to the Training set
classifier.fit(X_train, y_train, batch_size=10, epochs=100)
```

- In this step we fit the training data in our model. X_train, y_train is our training data.
- Here a batch size is basically a number of observations after which you want to update the weights here we take batch size 10.
- And the final parameter is epoch is basically when whole the training set passed through the ANN here we choose the 100 number of the epoch.

Predict the test set Result

In this step, we predict our test set result here our prediction results in probability so we choose 1(customer leave the bank) if the probability is greater than 0.5 otherwise 0(customer don't leave the bank).

```
# predicting the test set result
y_pred = classifier.predict(X_test)
y_pred = (y_pred>0.5)
y_pred
```

Making the Prediction and Accuracy Result

Confusion matrix

In this step we make a confusion matrix of our test set result to do that we import confusion matrix from sklearn.metrics then in confusion matrix, we pass two parameters first is y_test which is the actual test set result and second is y_pred which predicted the result.

```
# Confusion Metric
from sklearn.metrics import confusion_matrix
confusion_metric = confusion_matrix(y_test, y_pred)
confusion_metric
```

```
array([[1541,   54],
       [ 265,  140]], dtype=int64)
```

Accuracy Score

In this step, we calculate the accuracy score based on the actual test result and predict test results.

So here we go we get 84.05% of our ANN model.

```
# accuracy score
from sklearn.metrics import accuracy_score
accuracy_score = accuracy_score(y_test, y_pred)
accuracy_score
```

```
0.8405
```

Thank You