Machine Learning Applications in Mechanical Engineering ; A Hands-On Journey

Mentors

- Harshit Kumar
- Karthik S Pillai
- Mayank Jhunjhunwala

ARCHIL MOGRA
Y23 B.TECH. (ME)
9521745518

Project Overview

This project is mainly aimed towards developing proficiency in python and its libraries and applying fundamental machine learning algorithms.

Apart from Basics including KNNs & decision boundaries, ...Regression techniques, regularisation processes ,perceptrons and SVM , ANNs were also covered in this project.

Basics of Python

I learnt about basic python syntax, variables , comments, various data types , numbers, python -casting , strings , booleans, different types of operators etc. ,

Also I explored sets, lists, tuples, dictionaries, loops , conditionals which is required to handle any type of mechanical dataset

https://www.w3schools.com/python/

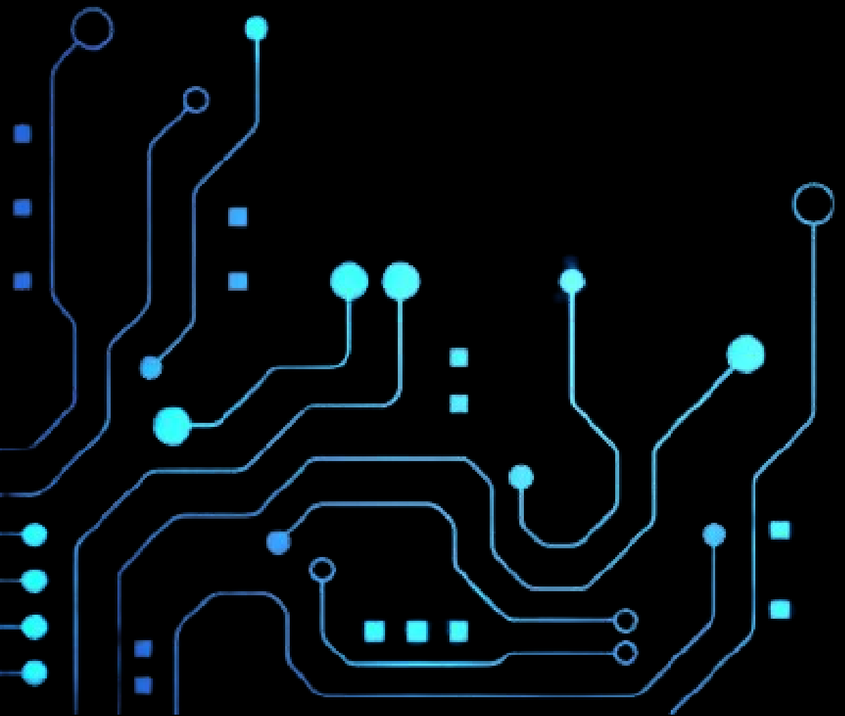
NumPy, Pandas Matplotlib

Creating 1D, 2D, 3D numPy arrays , accessing the arrays , indexing , slicing ,reshaping ,matrices ,and various numPy functions for reversing the array , finding determinant of 3D matrix , fibonacci series etc. were covered

Creating and plotting graphs of different types like histogram and scatter plot , subplotting , labelling, 3D contours & plots were also covered properly in matplotlib

ASSIGNMENT-1

In assignment-1 , we have given some basic codes on numPy to get on handy for the Machine Learning Alogorithms



1. Write a NumPy program to reverse an array

```
[ ]   1 Start coding or generate with AI.
```

2. Write a NumPy Program for finding the determinant of the 3D input matrix

```
[ ]   1 Start coding or generate with AI.
```

3.Write a numpy program for finding the traces of determinant

```
[ ]   1 Start coding or generate with AI.
```

4. Create a Fibonacci series and find the sum of first 100 terms

```
[ ]   1 Start coding or generate with AI.
```

K-Nearest Neighbor(KNN) Algorithm

The K-NN algorithm works by finding the K nearest neighbors to a given data point based on a distance metric, such as Euclidean distance. The class or value of the data point is then determined by the majority vote or average of the K neighbors. This approach allows the algorithm to adapt to different patterns and make predictions based on the local structure of the data.

STEP-1

Step 1: Selecting the optimal value of K

STEP-2

Calculating distance

STEP-3

Finding Nearest Neighbors

Decision Boundaries & its relation with KNN

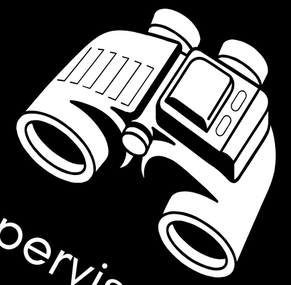
Mathematically, the boundary is defined as the set of points for which the decision criterion (majority vote) is exactly at the border between different classes.

Impact of 'K' on Decision Boundaries

The number of neighbors (k) affects the shape and smoothness of the decision boundary.

- **Small k**: When k is small, the decision boundary can become very complex, closely following the training data. This can lead to ***overfitting.***
- **Large k**: When k is large, the decision boundary smooths out and becomes less sensitive to individual data points, potentially leading to ***underfitting.***

Linear Regression



IN simple language , regression estimates dependent variable (such as I) for different values of V (V = IR)

SIMPLE REGRESSION MODEL $y\_pred. = w0 + w1*x$

We find w0 and w1 by minimising the cost function (Error function)
and by using the concept of maxima and minima

Put dE/dw0 and dE/dw1 = 0 , we get

w0 = y_mean - x_mean*w1

w1 = Sxy /Sxx

L.R. CONTINUES

SOME IMPORTANT RELATIONS

$$R = \frac{S_{xy}}{\sqrt{S_{xx} S_{yy}}}$$

Correlation Coefficient

$$R^2 = \frac{SS_R}{SS_T} = 1 - \frac{SS_{res}}{SS_T} \qquad 0 \le R^2 \le 1$$

Coefficient of determination

(Goodness of fit)

$$S_{xy} = \sum_{i=1}^{n} \left[(x_i - \overline{x}) (y_i - \overline{y}) \right]$$

$$S_{xx} = \sum_{i=1}^{n} (x_i - \overline{x})^2$$

$$S_{yy} = \sum_{i=1}^{n} (y_i - \overline{y})^2$$

$$w_1 = \frac{S_{xy}}{S_{xx}}$$

$$w_0 = \bar{y} - \bar{x} w_1$$

$$\widehat{y} = w_0 + w_1 x$$

$$\widehat{y} - \overline{y} = (x - \overline{x}) w_1$$

$$y - \widehat{y} = (y - \bar{y}) - (x - \bar{x}) w_1$$

Evaluation Metrics

Goodness of Fit

Rsq. -> 1	regression line runs close to data points , y is captured well
Rsq. -> 0	Fails to capture the variation in y
Rsq. > 0	y increases with x
Rsq. < 0	y decreases with x
R = 0	x and y are linearly independent

ASSIGNMENT - 2

Assignment 2 - Linear regression

The quench bath temperature, during heat treatment, affects the R

Test	Temperature (°C)	Hardness
1	30	55.8
2	30	59.1
3	30	54.8
4	30	54.6
5	40	43.1
6	40	42.2
7	40	45.2
8	50	31.6
9	50	30.9
10	50	30.8
11	60	17.5
12	60	20.5
13	60	17.2
14	60	16.9

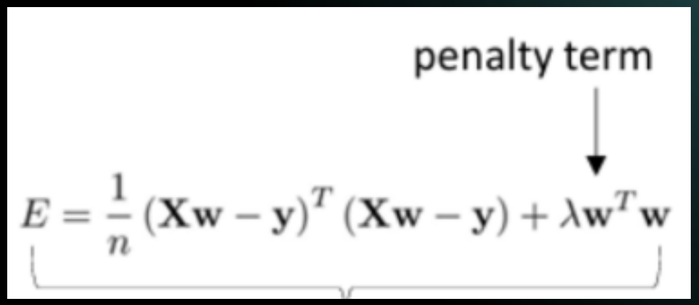
```
9
10 x_avg = np.mean(x_temp)
11 y_avg = np.mean(y_hardness)
12
13 Sxy = np.sum((x_temp- x_avg)*(y_hardness - y_avg))
14 Sxx = np.sum((x_temp - x_avg)**2)
15
16 w1 = Sxy/Sxx
17 w0 = y_avg - x_avg*w1
18
19 print("w0 = ",w0)
20 print("w1 = ",w1)
21
22 # predicted output
23 y_predic = w0 + w1*x_temp
24
25 y_avg_overall = np.mean(y_hardness)
26
27 # now coefficient of determination
28 SSr = np.sum((y_predic - y_avg_overall)**2)
29 SSy = np.sum((y_hardness - y_avg_overall)**2)
30 r2 = SSr/SSy
31
32 print("r2 = ",r2)
33
34
35 # now visualisation using matplotlib
36 plt.scatter(x_temp, y_hardness, label='Data Points')
37 plt.plot(x_temp, y_predic, color='red', label=f'Linear Regression (y = {w1:.2f}x + {w0:.2f})')
38 plt.xlabel('Quench Bath Temperature')
39 plt.ylabel('Rockwell Hardness')
```

In assignment - 2 we have covered Simple Linear Regression Model on Temperature vs. Hardness data during the heat treatment

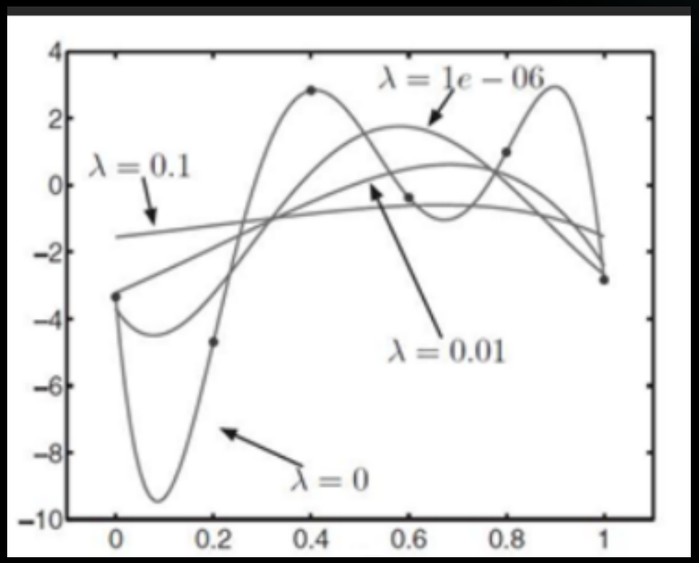
Regularisation (to control overfitting)

also known as ridge regression
(Thikonov regularisation)

In regularised regression , we define the cost function with lambda as penalty parameter



If lambda approaches 0 then it is our classic least sqaure regression



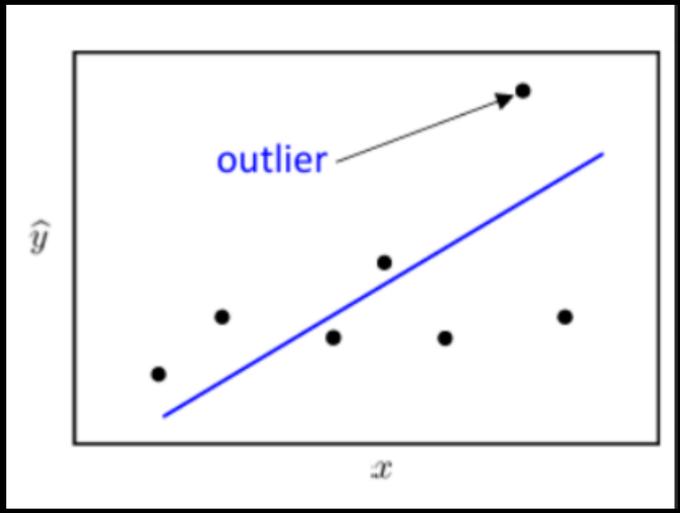
Lambda is decided by the cross validation
if we increase lambda , fluctuations decreases

Other types of regression such as Lasso , Elastic Net Regression are also covered in this project

Least Absolute Deviation

LAD regression is more robust than the least-squares method because it minimizes mean absolute errors instead of mean squared errors. This makes it less sensitive to outliers.

This method reduces the cost function to a much lower value as compared to the least squared Regression in general cases



Outliers - Some data points which do not follow the trend heavily

NON-LINEAR REGRESSION

N-linear Regression have non-linear normal equations which can be solved using appropriate numerical methods

One best way to avoid non-linearity is **LINEARISATION**

STEP-1

Direct minimisation of Error function

STEP-2

Solving non-linear normal equations

STEP-3

LINEARISATION

Non-Linear Regn. continues

Example of Non-Linear curve for regression

$$\hat{y} = w_0 x^{w_1} \quad \Rightarrow \ln \hat{y} = \ln (w_0) + w_1 \ln x$$

These techniques can be used in linearisation , but all non-linear models can not be handled alone by LINEARISATION

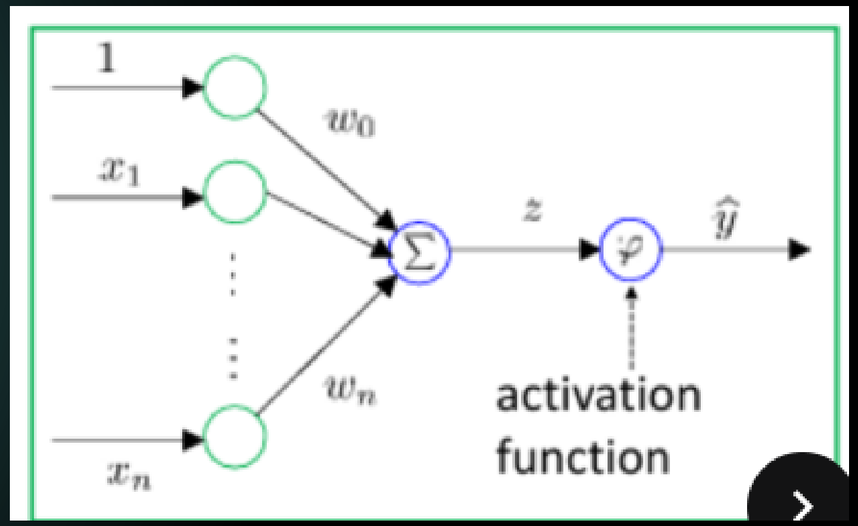
$$y \approx \theta_0 + \theta_1 x^{\theta_2} + \theta_3 x^{\theta_4}$$

Certain models like this cannot be solved like the above procedure because the terms are additive in nature

PERCEPTRON

$$\hat{y} = \varphi(z) \qquad z = w_0 + w_1 x$$

Perceptron solves ***<u>two class classification</u>*** problemby its specific learning algorithm in finite number of iterative steps if the data is *linearly separable*



<u>Online Learning</u> : Learning algorithm in which model uses one training data after another (completed in stages)
<u>Offline Learning</u> : uses all the training data together

- **<u>Certain Limitations</u>**
- Not reliable for test points near the boundary
- sensitive to outliers
- works only with linearly separable data

REMEDY
- SVM

PERCEPTRON continues

Activation Function

Activation function

$$\varphi(z) = \begin{cases} 1 & z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k)} + rL_i\mathbf{x}_i$$

$$\mathbf{w}^{(k+1)} \leftarrow \mathbf{w}^{(k)}$$

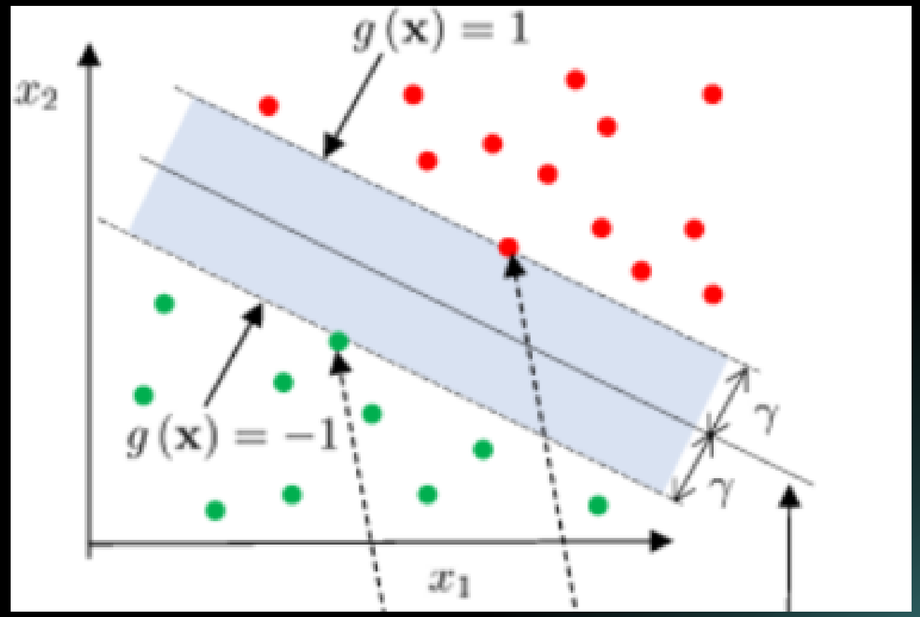
Iterative approach for weights

$$L_i = y_i - \widehat{y}_i \qquad L_i \in \{0, 1, -1\}$$

Loss Function

SUPPORT VECTOR MACHINE

- Support Vectors are the closest points , in different classes to the DB
- SVM classifies by creating a margin that depends only on those closest vectors making it insensitive to outliers
- Also support vector machine ***maximises the margin***

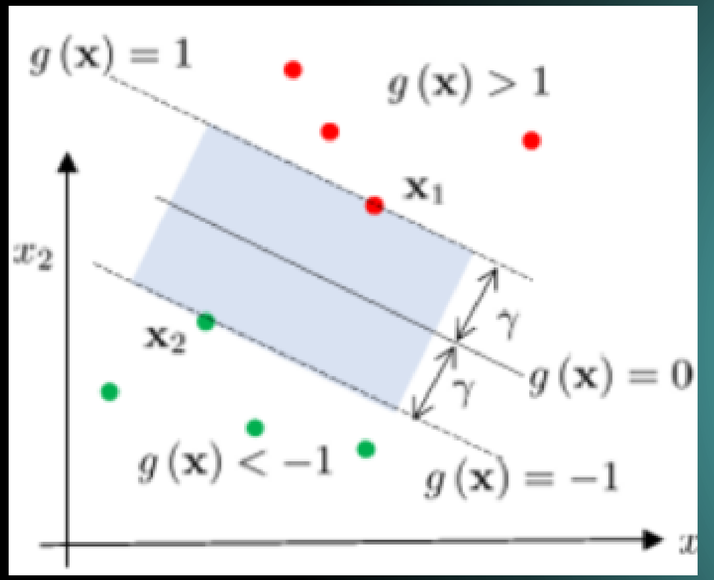


SVM maximizes margin $\gamma = \frac{1}{\|\mathbf{w}\|}$

where $yg(\mathbf{x}) \geq 1$ $g(\mathbf{x}) = w_0 + \mathbf{w}^T\mathbf{x}$ $y = \{-1, 1\}$

maximizing $\frac{1}{\|\mathbf{w}\|}$ is equivalent to minimizing $\|\mathbf{w}\|$

same as minimizing $\frac{1}{2}\mathbf{w}^T\mathbf{w}$



ARTIFICIAL NEURAL NETWORKS

Weight Initialisation inANNs

Xavier Initialisation

Uniform random number in the interval $\left[-\frac{1}{\sqrt{m+n}}, -\frac{1}{\sqrt{m+n}}\right]$

m,n are no. of incoming and outgoing connectors to a node

Various Error Functions

$$E = \frac{1}{n}\sum_{i=1}^{n} L_i$$

Mean Squared Error (MSE)

$$E = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(y_{ij}\ln\hat{y}_{ij}\right)$$

Multiclass cross entropy

$$E = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \ln\hat{y}_i + (1-y_i)\ln(1-\hat{y}_i)\right]$$

predicted probability of being in class 1

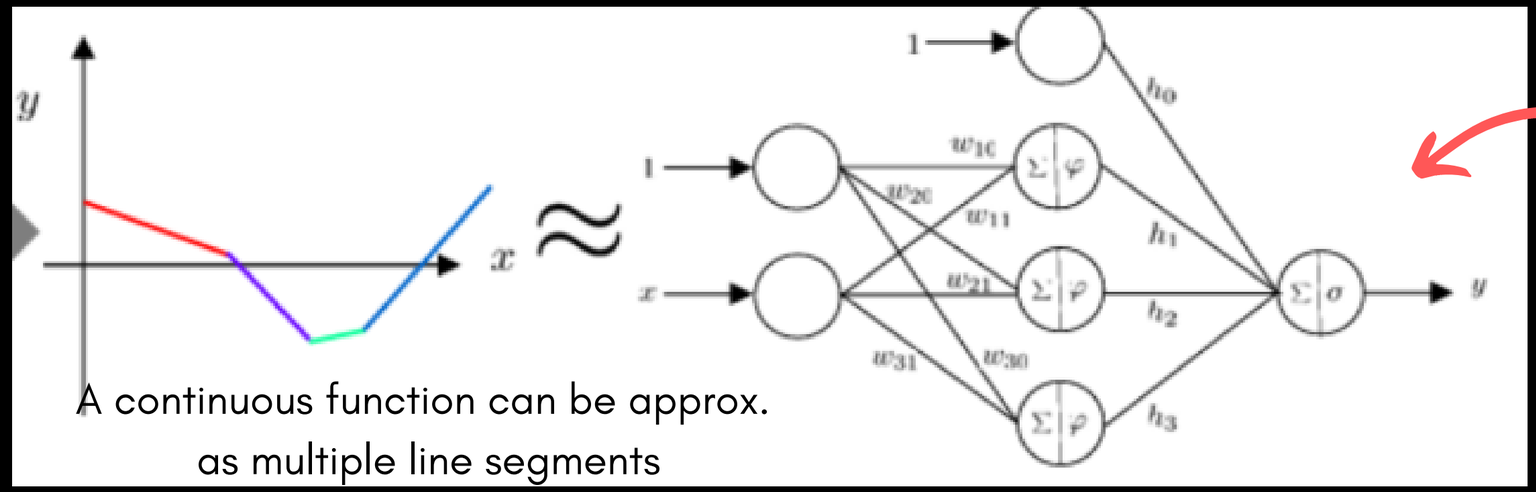
predicted probability of being in class 0

Binary cross entropy

$$E = \frac{1}{n}\sum_{i=1}^{n}|L_i|$$

Mean Absolute Error (MAE)

UNIVERSAL APPROXIMATION THEOREM



A continuous function can be approx. as multiple line segments

A properly designed *shallow (1 hidden layer)* ANN can approximate any continuous function

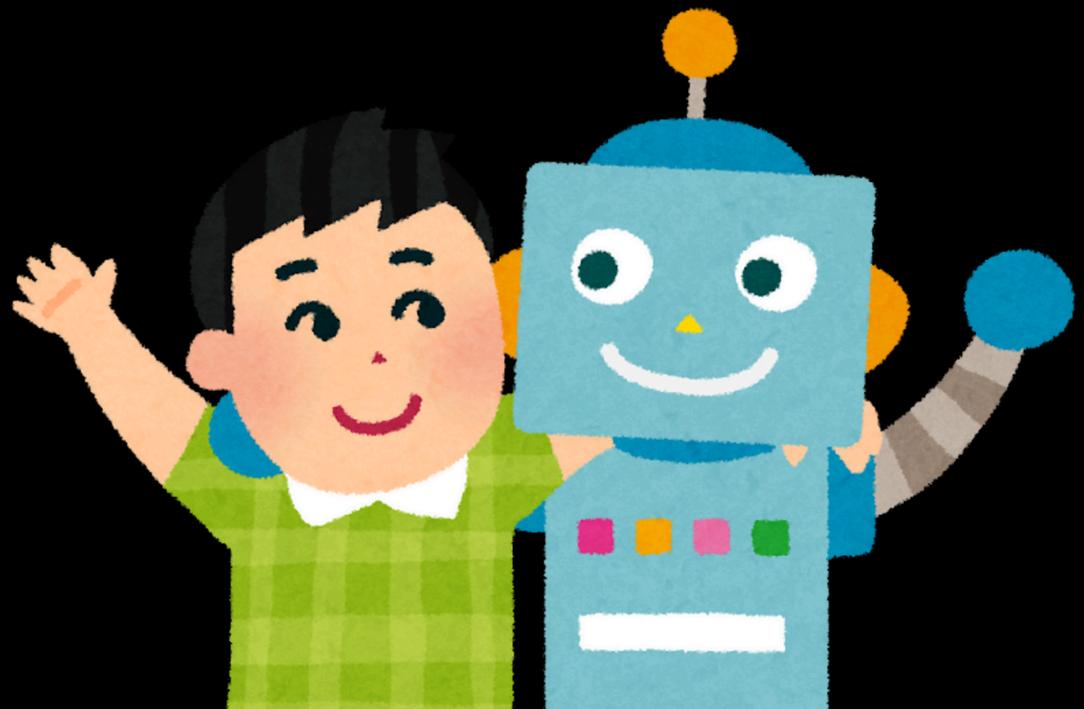
$$y = \sigma\left[h_0 + \sum_{i=1}^{n} h_i \varphi_i\left(w_{i0} + w_{i1}x\right)\right]$$

The output y of this network will be in this form

$$\varphi\left(w_{10} + w_{11}x\right)$$
$$\varphi\left(w_{20} + w_{21}x\right)$$
$$\varphi\left(w_{30} + w_{31}x\right)$$

The output y is the linear combination of ALL 3 equations

THANK YOU



ARCHIL MOGRA