ML Bootcamp

Anant Upadhyay
IIT (ISM) Dhanbad

Description:- We will be provided with training dataset and we would need to train different machine learning algorithms from it.We will be required to train n-layer neural networks,linear and polynomial regression,logistic regression,decision trees,KNN and K-means.After implementation we would need prepare report on jupyter notebook explaining the whole development process(all The things that we tried,tested and implemented with reasoning).After that we would need to create our own python library from which our algorithms can be easily accessed.

Features:-

- Linear regression:-Train a linear model based on the training dataset using gradient descent.Find the line of best fit from the training dataset.
- Polynomial regression:-Train a polynomial model containing square,cube or higher powers of the features.
- Logistic regression:-Train the model using the sigmoid function.This is used for classification problems as then we can classify using a threshold.Eg:- a>=0.5 then outcome is positive otherwise negative.
- Neural networks:- Construct a n-layer neural network to make predictions on the given dataset.Then adjust the model's parameters by the help of back propagation.We will train the model by nudging weights and biases of each neuron.
- Decision trees:-Create a model which starts with a root node and further splits into more nodes.The model is then trained by making it learn the right decision rules.
- KNN:-This algorithm finds the “k” nearest neighbors of the datapoint that is to be predicted, and then bases its prediction on its nearest neighbors.
- K-means:-This algorithm classifies the data into k different clusters.The clusters improves by every iteration by trying to minimize the distance of the points within a cluster.
- Other parameters/procedures:- We add an additional parameter λ to regularize our data(for dealing with bias and variance).Also for increasing speed and accuracy we can normalize the given data(If required)

Technology stack:-

- Basic knowledge of the python programming language
- Knowledge of the numpy library to easily work with arrays,vectors and matrices
- Knowledge of pandas library to load,edit and manipulate data
- Knowledge of the matplotlib library to show visualizations of our model using graphs,contour plots,scatter plots etc
- Knowing the usage of Jupyter and google Collab notebooks(to document our project and present it in the form of a report).Eg:-Usage of markup and code cells,running cells etc.
- Knowledge of classes and vectorization in python(To make implementation easier,faster and later convert the work in a python library)

- Knowledge of the working and implementation of different algorithms and also how to train them.(Knowing how to create models using:- linear regression,polynomial regression,logistic regression,KNN,K-means,neural networks,decision trees).And also knowing how to train them (using gradient descent,backpropagation etc).
- Some basic knowledge of statistics(for feature scaling) and differential calculus(for the gradient descent and backpropagation algorithms)

Brief Implementation Details:-

We would implement different algorithms on the training dataset such as linear regression,polynomial regression,logistic regression,KNN,K-means,neural networks decision trees and present it in the form of a report(Also convert it into a library if time permits).

Linear regression:-
Model:-

$$f_{w,b}(x^{(i)}) = w \cdot x^{(i)} + b$$

w(weight) and b(bias) are parameters
Cost function:-
We calculate the loss of each datapoint to measure how far our model is from the training example.Then we sum all the losses to create the cost function

$$J(w, b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$$

$x^{(i)}, y^{(i)}$ represent the ith training example from our training dataset
Training(By gradient descent):-

$$w = w - \alpha\frac{\partial J}{\partial w}$$

$$b = b - \alpha\frac{\partial J}{\partial b}$$

α is the learning rate.We simultaneously update w,b after each iteration

Polynomial regression:-
The implementation is almost same as shown for linear regression just the model is tweaked a little bit to account for higher powers of x.
Model:-

$$f_{w,b}(x^{(i)}) = w_1 \cdot x^{(i)} + w_2 \cdot (x^{(i)})^2 + w_3 \cdot (x^{(i)})^3 + b$$

More higher powers can also be used.Note the regularization parameter λ may be introduced if required.

Logistic regression:-
Here we use the sigmoid function to adress classification problems.In such problems the input is Some real number and the output can be represented as a 0 or 1.

Model:-

$$f_{w,b}(x^{(i)}) = \frac{1}{1+e^{-(w\cdot x^{(i)}+b)}}$$

Here we use a different kind of loss function to penalize the model.

Loss function:-

$$L(f_{w,b}(x^{(i)}), y^{(i)}) =- y^{(i)}ln(f_{w,b}(x^{(i)})) - (1 - y^{(i)})ln(1 - f_{w,b}(x^{(i)}))$$

Cost function:-

$$J(w, b) = \frac{1}{m}\sum_{i=1}^{m} L(f_{w,b}(x^{(i)}), y^{(i)})$$

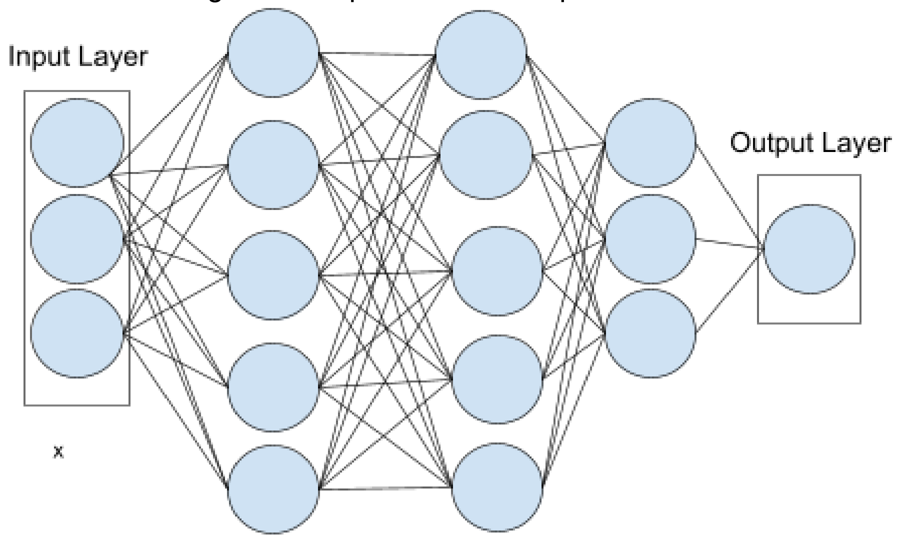
Training(By gradient descent):-

$$w = w - \alpha\frac{\partial J}{\partial w}$$

$$b = b - \alpha\frac{\partial J}{\partial b}$$

Neural Network:-

We have multiple layers of neuron in a neural network.The input is 1st propagated through the Input layer.Each neuron takes the activation of all neurons in it’s previous layer as input and Uses it to compute its own activation using its activation function.The activation function of all neurons(except the output neuron) are Relu function.The activation function of output layer is Chosen according to the requirement of the problem.



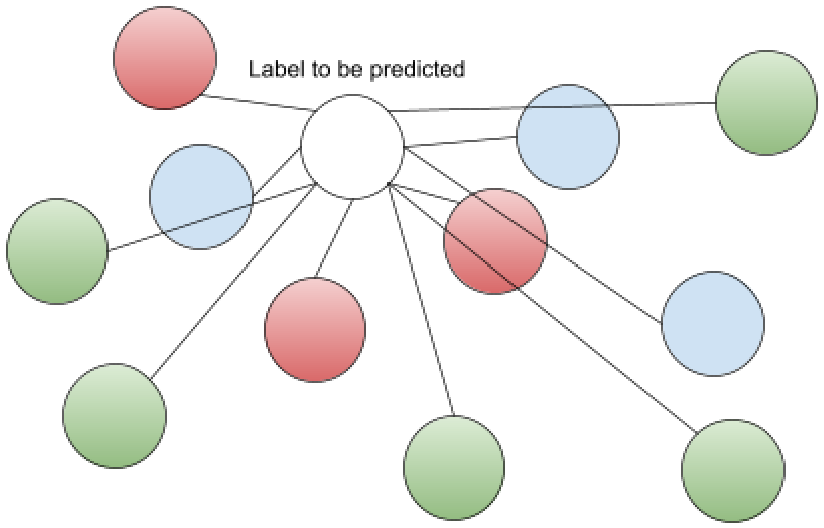
The neural network computes the output by forward propogation.

Forward propagation:-

$$a_j^{[L]} = g(\vec{w}_j^{[L]} \cdot \vec{a}^{[L-1]} + b_j^{[L]})$$

Above equation represents the activation of the jth neuron in the Lth layer of the neural network. Here g(x) represents the activation function of the neuron.

KNN:-



First we find the distance of the k nearest neighbors from the point,using the formula:-

$$d(x, y) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Here x,y are the data points and x_i and y_i represent the ith feature of x and y.

For classification:-

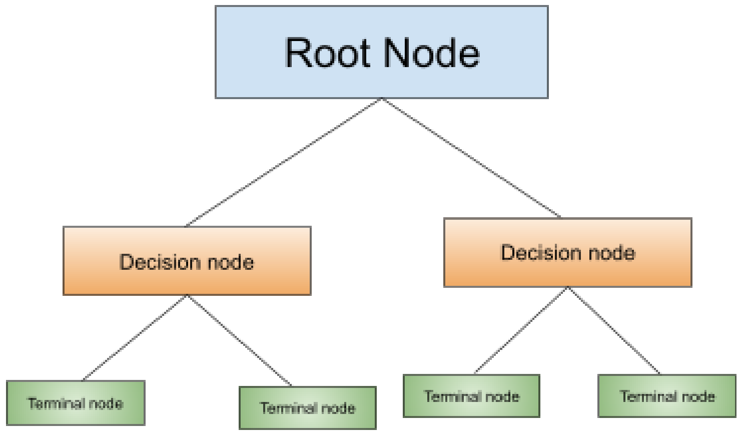
We classify the label by seeing the class which is most common to its k nearest neighbors.

For regression:-

We return the output as an average of its k nearest neighbors

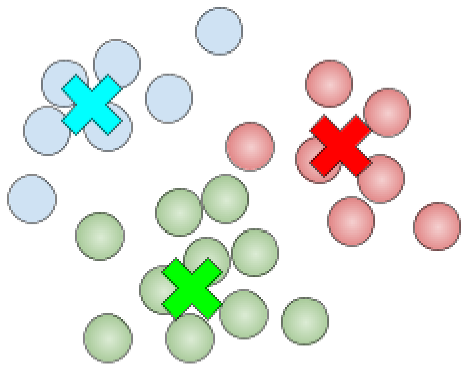
$$y_{predicted} = \frac{\sum_i y_i}{k}$$

Descision trees:-



The root node represents the entire data which is further divided into smaller and smaller nodes. When a subnode is split into 2 or more subnodes they are called decision nodes.When the data reaches the terminal node it represents the output and it can’t be divided into further sub-nodes. By training the decision tree we teach it to take the right decision steps while splitting into sub-nodes.

K-means:-



Here cross represents the centroid of the cluster

In the k-means algorithm we first specify the number of clusters(k) we need.Then we randomly assign the centroid of these clusters to any of the given data points.After that the data points nearest to the cluster's centroid are classified into that particular cluster.Then in the next Iteration we again compute the centroid of the cluster(by taking an average of the data points in that cluster) and then again calculate the points nearest to that cluster, Hence forming a new cluster. We keep repeating this process until the centroid of the clusters don't change significantly.

Centroid of the cluster can be computed as:-

$$\overline{x}_j = \frac{\sum_{i=1}^{n} x_j^{(i)}}{n}$$

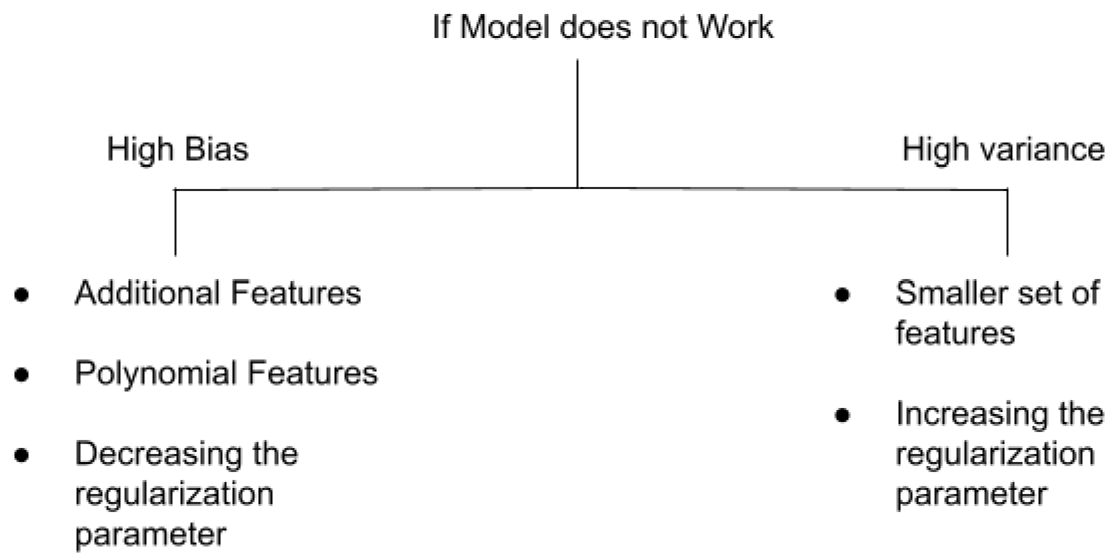
Here n is the number of data points in the cluster, and j represents the jth feature of the data point.

Development process:-
Just creating a model and training it on the training data(Using gradient descent) is not enough to ensure a successful model.For example we may need to increase/decrease the learning rate α to an optimum level so that training of model is not too slow and also ensure our model is not overshooting.So to effectively develop our model we split our training data into two parts J_{train} and J_{test}.After training our data on J_{train} we test it on J_{test}.

By using J_{train} and J_{test} we draw the following conclusions:-

- If J_{train} and J_{test} both are high the model has **high bias**(model is underfitting)
- If J_{train} is low but J_{test} is high model has **high variance**(model is overfitting)

Flowchart for dealing with bias and variance:-



Note here λ is the regularization parameter.This parameter is introduced in the cost function to avoid the problem of overfitting(**High variance**).Regularization keeps the values of all the Weights low in order to avoid overfitting.
Implementation:-

$$J(\vec{w}, b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

As you can see an additional regularization term is introduced in the cost function $J(\vec{w}, b)$
Feature engineering(If required):-
We could add/remove some features to adress underfitting and overfitting.If the range of data varies a lot among given features we could normalize our data for faster and more accurate results.
Z-score normalization:-

$$z = \frac{x-\mu}{\sigma}$$

Here z is the normalized data,x is the original data,μ is the mean of the data and σ is the Standard deviation of the data.

Week Wise Timeline:-

Week 1: 20/02/23-26/02/23
Complete course1 and 2 of the machine learning course available on coursera by Andrew Ng:-
https://www.coursera.org/specializations/machine-learning-introduction
Also refer to the additional resources on KNN,K-means and Decision trees provided by seniors. By completing the course it's meant that all the videos,labs,quizzes and assignments are completed.

Week 2: 27/02/23-05/02/23
Implement linear regression,logistic regression,polynomial regression and neural networks.Also

simultaneously make a report on jupyter notebook documenting the whole development process showing how I dealt with high bias and variance.Eg:-Showing the training logs,tweaking of hyperparameters and visualizations(show graphs,scatter plots,contour plots using matplotlib).

Week 3: 06/02/23-12/02/23
Implementing KNN,K-means,decision trees on my jupyter notebook along with documentation of the process.Also test the developed algorithms for bias and variance and document the procedure followed for correcting it.

Week 4: 13/02/23-19/02/23
After the implementation of all algorithms,try using classes to optimize the already made algorithms and try converting the work in a python library(If time permits).

About Me:

- **Personal Details**

Name:- Anant Upadhyay
Branch:- 1st year,B-tech in Computer Science and Engineering
Admission No:- 22JE0110
D.O.B:- 24/11/2004

- 1 star coder in codechef :- https://www.codechef.com/users/anant_up11
- Made a project on the game tic tac toe:- https://github.com/Anant-Upadhyay/Tictactoe
- Github profile:-https://github.com/Anant-Upadhyay
- Completion of 1st course on Coursera:- https://coursera.org/share/38b6e1ab1e5d6b6ef796f0df070c337b
- Completion of 2nd course on Coursera:- https://www.coursera.org/account/accomplishments/certificate/LVZBLU7NC9NP

- **Why should I be selected for this project?**

I am a keen learner with a childhood interest in machine learning.I look forward for pursuing a career in machine learning and artificial intelligence as these topics have always fascinated me.Being part of Cyberlabs will allow me to explore these fields deeper and explore topics like Computer vision,NLP,Supervised Learning,Unsupervised Learning etc.Being in cyberlabs will provide me with a team of like-minded people which will help in my personal growth,as well as the growth of the cyberlabs team.I would also like to participate in hackathons and discuss research papers on machine learning as I have always been fascinated by machine learning.

- **Do you have any other commitments during the program?**

On average I would be able to devote 2 hours per day to the project.Also I am staying on campus during the holi break,hence that free time will be utilized wisely for working on the project.