**ML Algorithms**

1. **Liner Regression:** *It is supervised machine learning algo.* 
   1. **Simple Linear regression**
      1. *It has only single input and single output or label column*
   2. **Multiple Linear regression**
      1. *If data has more than one input column*
   3. **Polynomial Linear regression**
      1. *If your data is not linear than you will use this algo*
   4. **Regularization**

**How to proceed for solving ML Problems?**

1. Plot your data.
2. Separate your Input and output/label columns
3. Do train train-test split
4. Train the model on training set.
5. Predict the with test set.

In Regression y = mx+b . where m is weightage of x (how much x depends on y) and b is minimum value if mx will zero.

**Mathematical intuition of Linear Regression:**

**Y = mx + b**

How to find the values of m and c:

1. **Closed form**(used when dimensions are low)
   1. OLS (Ordinary least square)
2. **Non-Closed form**(used when dimensions are high)
   1. Gradient descent (*used due to low time complexity*)

**OLS (Direct formula):**

For simple linear regression b =

m = error function =

where d = distance from best fit line. and d =

So we can rewrite the error function as =

Where

Then the error function will be

**We want to minimize the error for best fitting the line**

Now from this point we have to do some calculus partial derivation

Partial derivation of y, m = 0 and b = -1

By dividing both side with -2 we get

Now we write it as =

Now

So we can write the whole equation as = b =

Where n is the number of rows in the table

Now we have value of b we can now get the value (m) with the help derivation and the equation will be = m =

**For multiple linear regression**

Formula =

Where n = number of rows and m = number of columns

**Gradient Descent**

*Gradient descent is a first-order iterative optimization algorithm for finding a local differentiable function. The idea is to take repeated steps in the opposite direction of the gradient or approx. gradient of the function at the current point, because this is the direction of steepest descent.*

Note: It is just an optimization technique which gives you the minima of function you gave to it.

Used in:

1. Linear Regression
2. Logistic Regression
3. TSNE
4. It is back bone of Deep Learning

Mathematical formulation:

Steps. Start with a random value of b and m

Let epoch = 100 and ƞ(learning rate) = 0.01

we want to minimize the loss here we know the values of y and x, the loss function depends on the m and b, let assume for understanding we know the m and loss function depends only on b

We will do derivation w.r.t (b)

We will get the b\_slope =

We will do derivation w.r.t (m)

We will get the m\_slope =

For x in epoch:

bnew = bold  - ƞ\*slope(b)

mnew = mold - ƞ\*slope(m)

**Types of Gradient Descent**

1. Batch GD
   1. In batch GD it reads whole of the rows from your data then update its m,b
   2. It is slow
   3. Above GD was batch GD
2. Stochastic GD
   1. Error prone
   2. Fast
   3. Good for big data
   4. Update the m,b after reading single row
3. Mini Batch GD
   1. Update m,b after the size of batch let say after reading 10 rows a single update

**Mathematical formulation of Batch GD for n dimensions**

Formula of Y =

Steps: Pick random values for all betas, generally they are

Now set epoch let say = 100, and leaning rate ƞ = 0.1

If loop

= - ƞ\*slope()

= - ƞ\*slope()

= - ƞ\*slope()

Slope will be the partial derivation of loss function w.r.t that

Now will get for

Problems with Batch GD:

1. Slow for big data. Because it performs large number of calculations.
2. Hardware problem

**Stochastic Gradient Descent**

In SGD, instead of using the entire dataset for each iteration, only a single random training example (or a small batch) is selected to calculate the gradient and update the model parameters. This random selection introduces randomness into the optimization process, hence the term “stochastic” in stochastic Gradient Descent

for

**When to use:**

1. When you have big data.
2. When you have non-convex function. (When you both global and local minima)

**Mini Batch Gradient Descent**

Parameters are updated after computing the gradient of  the error with respect to a subset of the training set.

| **Batch Gradient Descent** | **Stochastic Gradient Descent** | **Mini-Batch Gradient Descent** |
| --- | --- | --- |
| Since the entire training data is considered before taking a step in the direction of gradient, therefore it takes a lot of time for making a single update. | Since only a single training example is considered before taking a step in the direction of gradient, we are forced to loop over the training set and thus cannot exploit the speed associated with vectorizing the code. | Since a subset of training examples is considered, it can make quick updates in the model parameters and can also exploit the speed associated with vectorizing the code. |
| It makes smooth updates in the model parameters | It makes very noisy updates in the parameters | Depending upon the batch size, the updates can be made less noisy – greater the batch size less noisy is the update |

**Polynomial Regression**

**Polynomial Regression**is a form of linear regression in which the relationship between the independent variable x and dependent variable y is modeled as an *nth-degree* polynomial. Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y, denoted E(y | x).

**How does a Polynomial Regression work?**

If we observe closely then we will realize that to evolve from linear regression to polynomial regression. We are just supposed to add the higher-order terms of the dependent features in the feature space. This is sometimes also known as feature engineering but not exactly.

Simple polynomial regression =

**Regression Metrics**

1. MAE (*mean absolute error*)
   1. y^ = prediction

(It’s not differentiable at zero due to modulus, Robust to outliers)

1. MSE (*mean squared error*)

(It’s differentiable at zero due to square, not Robust to outliers)

1. RMSE (*root over MSE*)

(not Robust to outliers)

1. R2 Score ( *not affected by data context works on every type of data*)
2. Adjusted R2 SME

**Bias Variance Trade-off**

In ability of the machine learning model to truly capture relationship in the training data.