Gradient Descent

Definition

- Gradient Descent is a parameter optimization technique built upon the concept pf Calculus.
- It is used in wide variety of applications in Machine learning:

i)Linear Regression

ii)Logistic Regression

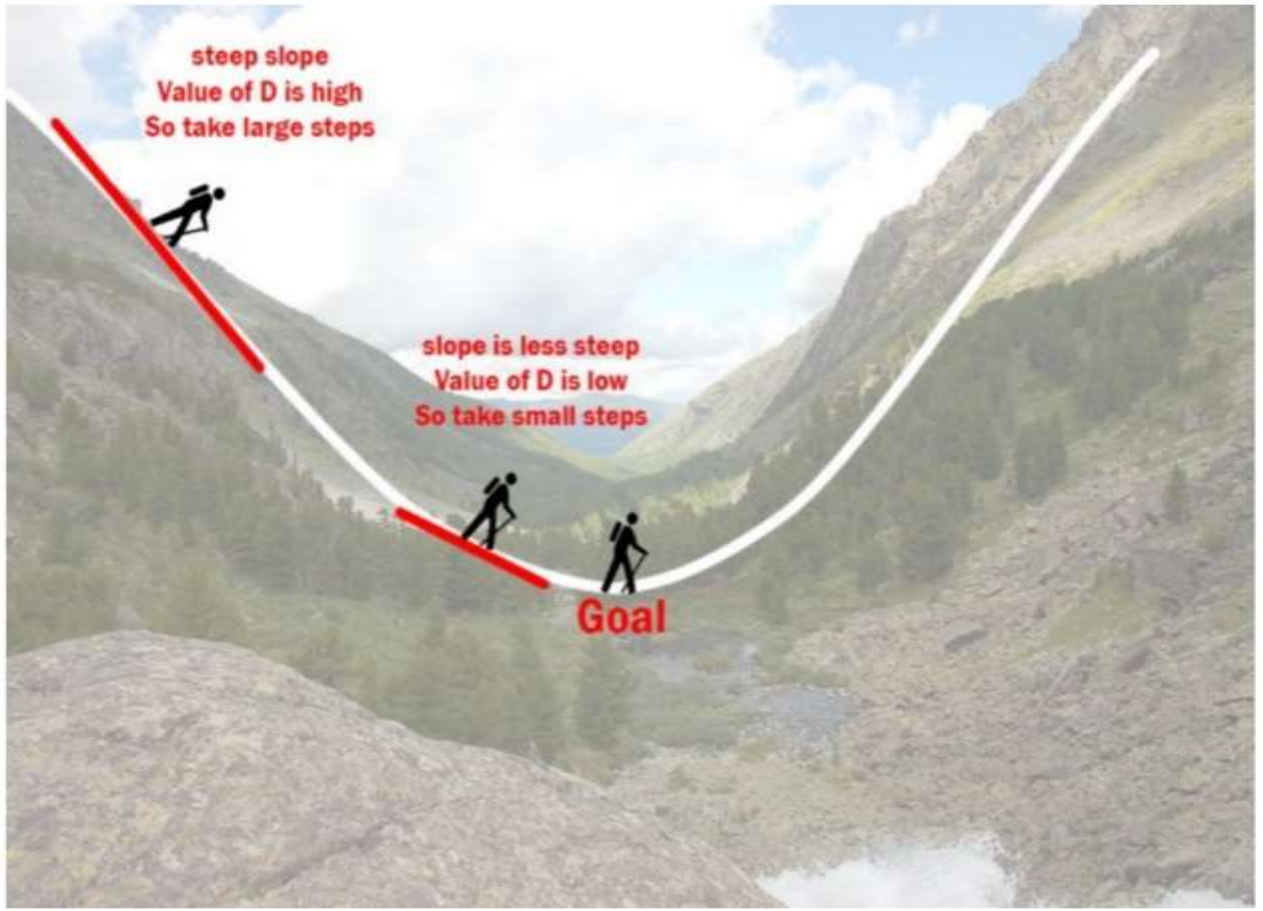
iii)PCA

iv) Neural Network

Contd.

- Gradient descent is an iterative optimization algorithm to find the minimum of a function. Here that function is our Loss Function.

Understanding Gradient Descent



Example

- Imagine a valley and a person with no sense of direction who wants to get to the bottom of the valley.
- He goes down the slope and takes large steps when the slope is steep and small steps when the slope is less steep.
- He decides his next position based on his current position and stops when he gets to the bottom of the valley which was his goal.

Example

- Let's try applying gradient descent to m and c and approach it step by step:
  - Initially let m = 0 and c = 0. Let L be our learning rate. This controls how much the value of m changes with each step. L could be a small value like 0.0001 for good accuracy.
  - Calculate the partial derivative of the loss function with respect to m, and plug in the current values of x, y, m and c in it to obtain the derivative value D.

Example

$$D_m = \frac{1}{n}\sum_{i=0}^{n} 2(y_i - (mx_i + c))(-x_i)$$

$$D_m = \frac{-2}{n}\sum_{i=0}^{n} x_i(y_i - \bar{y}_i)$$

- D_m is the value of the partial derivative with respect to m. Similarly lets find the partial derivative with respect to c, Dc :

$$D_c = \frac{-2}{n}\sum_{i=0}^{n} (y_i - \bar{y}_i)$$

Example

Now we update the current value of m and c using the following equation:

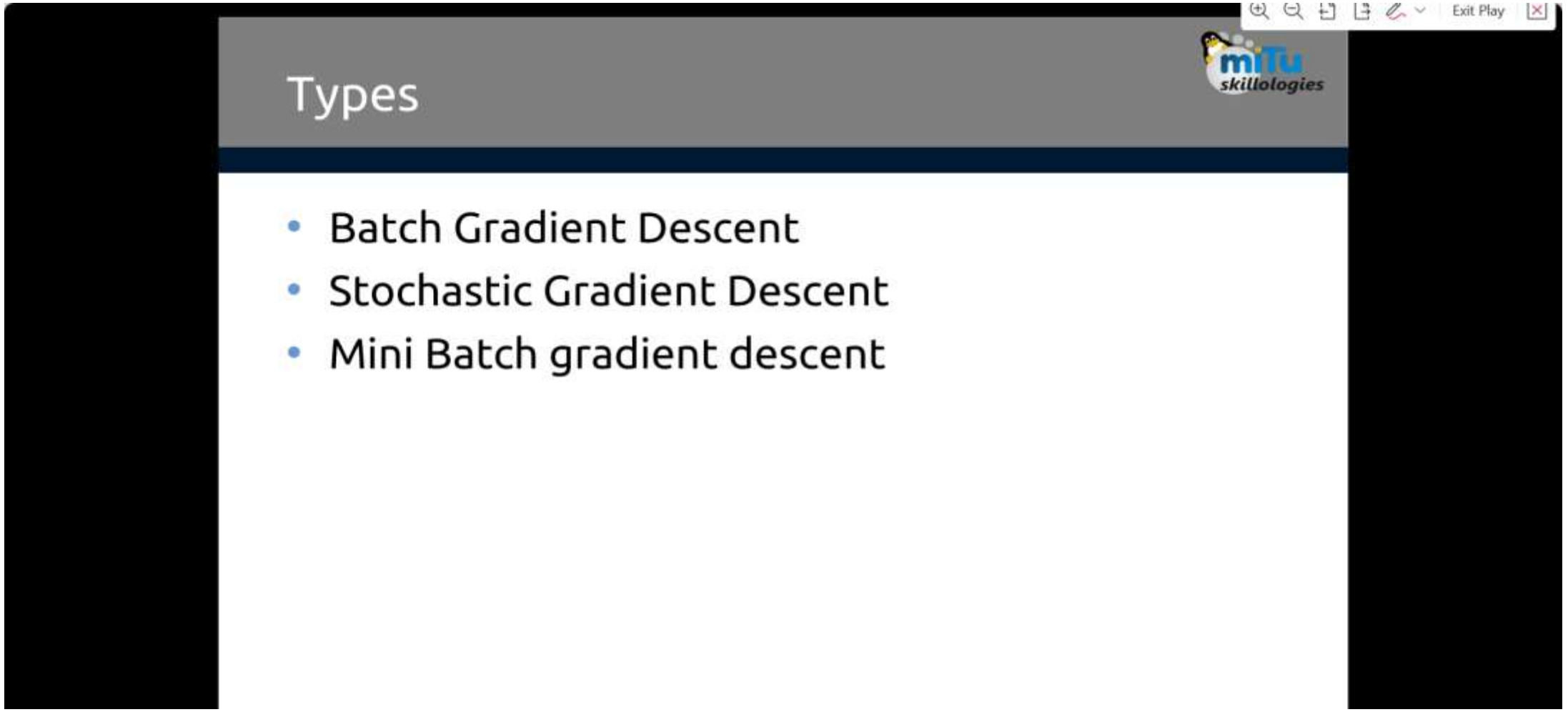
$$m = m - L \times D_m$$

$$c = c - L \times D_c$$

- 4. We repeat this process until our loss function is a very small value or ideally 0 (which means 0 error or 100% accuracy). The value of m and c that we are left with now will be the optimum values.

Example

- Now going back to our analogy, m can be considered the current position of the person. D is equivalent to the steepness of the slope and L can be the speed with which he moves.
- Now the new value of m that we calculate using the above equation will be his next position, and L×D will be the size of the steps he will take.
- When the slope is more steep (D is more) he takes longer steps and when it is less steep (D is less), he takes smaller steps. Finally he arrives at the bottom of the valley which corresponds to our loss = 0.
- Now with the optimum value of m and c our model is ready to make predictions !


Exit Play
Types
miTu skillologies
Batch Gradient Descent
Stochastic Gradient Descent
Mini Batch gradient descent

Batch Gradient Descent

- This is a type of gradient descent which processes all the training examples for each iteration of gradient descent.
- But if the number of training examples is large, then batch gradient descent is computationally very expensive.
- Hence if the number of training examples is large, then batch gradient descent is not preferred. Instead, we prefer to use stochastic gradient descent or mini-batch gradient descent.

Stochastic Gradient Descent

- This is a type of gradient descent which processes 1 training example per iteration.
- Hence, the parameters are being updated even after one iteration in which only a single example has been processed.
- Hence this is quite faster than batch gradient descent.
- But again, when the number of training examples is large, even then it processes only one example which can be additional overhead for the system as the number of iterations will be quite large.

Mini Batch Gradient Descent

- This is a type of gradient descent which works faster than both batch gradient descent and stochastic gradient descent.
- Here b examples where b<m are processed per iteration. So even if the number of training examples is large, it is processed in batches of b training examples in one go.
- Thus, it works for larger training examples and that too with lesser number of iterations.