Variants of Gradient Descent

There are three different variants of Gradient Descent

- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini-batch Gradient Descent

Batch Gradient Descent/Vanilla gradient descent

- Batch Gradient Descent (BGD) is a variant of the Gradient Descent optimization algorithm used to minimize a cost function in machine learning and deep learning models.
- It's called "batch" because it processes the entire training dataset in each iteration to update the model parameters.

Batch Gradient Descent Algorithm:

- For a total of 'm' observations in a dataset, we use all these observations to calculate the cost function J
- Pass the entire training set (one epoch) to model, ○ Perform forward propagation and calculate the cost function
- And then update the parameters using the rate of change of this loss function (LF) wrt the parameters
- After one epoch, f/w propagation & b/w propagation are performed
- And the parameters are updated once per epoch
- Vectorized implementation can be used for weight and input multiplication.
- If the number of training examples is large, then vanilla gradient descent is going to take a long time to converge due to the fact that a weight update is only happening once per data cycle.

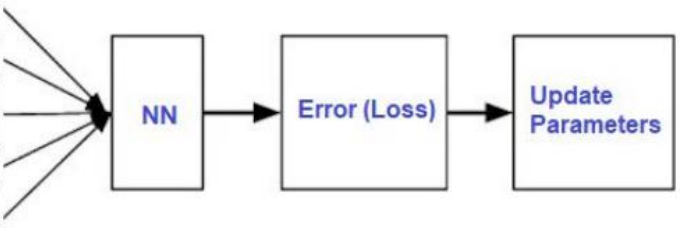
Stochastic Gradient Descent (SGD)

- SGD is a variant of the gradient descent optimization algorithm widely used in machine learning and deep learning.
- Unlike batch gradient descent, which computes the gradient using the entire dataset, SGD updates the model parameters using a single training example at a time.

Batch Gradient Descent Algorithm:

- Initialization: Start with initial guess for model parameters.
- For each epoch:
  - Shuffle the training dataset.
  - Iterate through each training example:
    - Compute the gradient of the cost function with respect to the current training example.
    - Update the model parameters using the computed gradient and a predefined learning rate.
- Repeat the process for a fixed number of epochs or until convergence criteria are met.
- if we have N=10,000 images, then we would have 10,000 weight updates per epoch.
- SGD tends to converge much faster because it's able to start improving itself after each and every weight update.

SGD Example	Feature 1	Feature 2	Feature 3
Observation 1	1	2	3
Observation 2	2	3	4
Observation 3	0	1	2
Observation 4	3	5	6
Observation 5	8	9	0



Mini-batch SGD

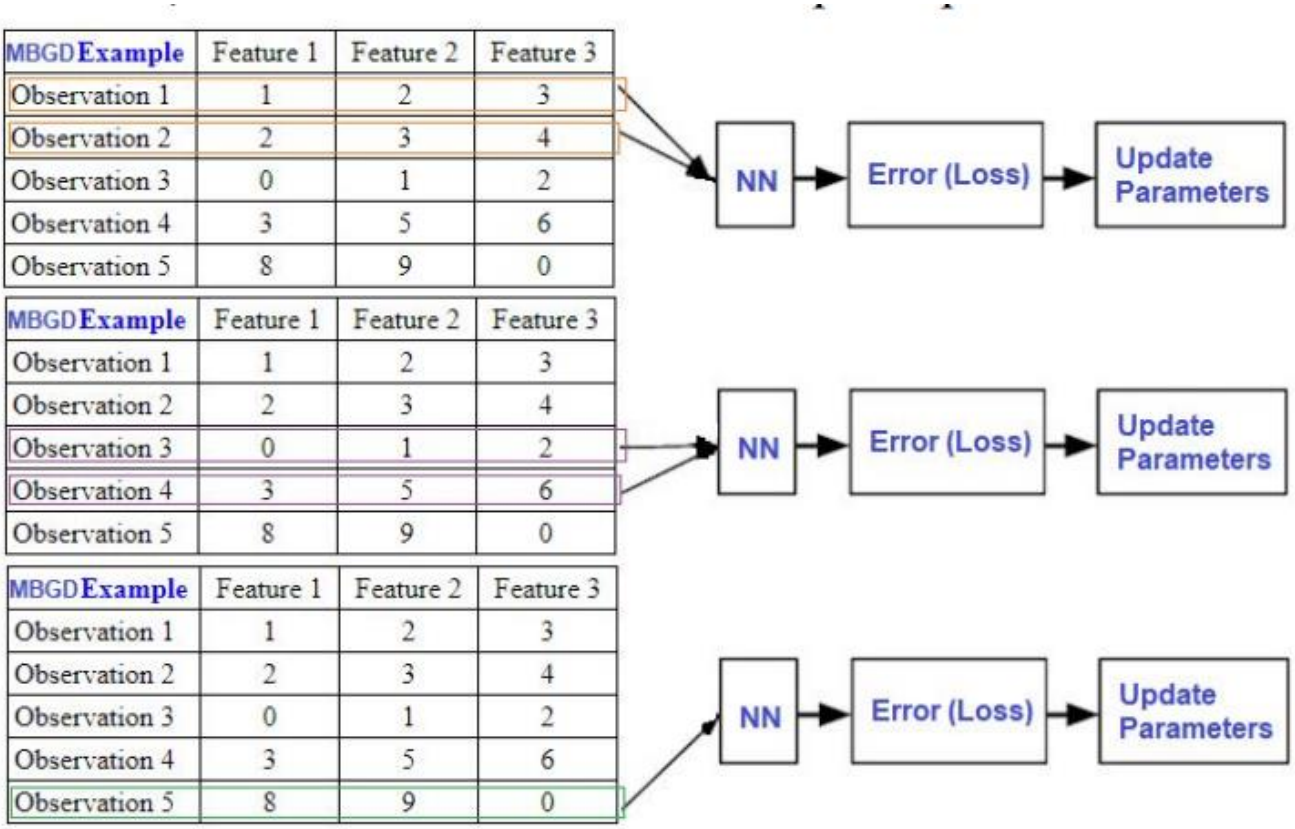
Mini-batch SGD introduces the concept of a batch size, *S*. Now, given a dataset of size *N*, there will be a total of *N / S* updates to the network. It takes a subset of the entire dataset to calculate the cost function

Mini-batch SGD Algorithm:

- Initialization: Start with an initial guess for the model parameters.
- Divide the training dataset into mini-batches of equal size (e.g., 32, 64, or 128 examples per batch).
- For each epoch:
- Shuffle the training dataset to introduce randomness and prevent the model from getting stuck in local minima.
- Iterate through each mini-batch:
  - Compute the gradient of the cost function with respect to the mini-batch.
  - Update the model parameters using the computed gradient and a predefined learning rate.
- Repeat the process for a fixed number of epochs or until convergence criteria are met.

Advantages:

- Faster than Batch
- Random selection will help avoid redundant examples


MBGD Example | Feature 1 | Feature 2 | Feature 3
Observation 1 | 1 | 2 | 3
Observation 2 | 2 | 3 | 4
Observation 3 | 0 | 1 | 2
Observation 4 | 3 | 5 | 6
Observation 5 | 8 | 9 | 0
NN
Error (Loss)
Update Parameters
MBGD Example | Feature 1 | Feature 2 | Feature 3
Observation 1 | 1 | 2 | 3
Observation 2 | 2 | 3 | 4
Observation 3 | 0 | 1 | 2
Observation 4 | 3 | 5 | 6
Observation 5 | 8 | 9 | 0
NN
Error (Loss)
Update Parameters
MBGD Example | Feature 1 | Feature 2 | Feature 3
Observation 1 | 1 | 2 | 3
Observation 2 | 2 | 3 | 4
Observation 3 | 0 | 1 | 2
Observation 4 | 3 | 5 | 6
Observation 5 | 8 | 9 | 0
NN
Error (Loss)
Update Parameters

Comparison of GD, SGD, and Mini-batch

Batch GD	SGD	Mini-batch GD
Take entire dataset for updation	Take single observation for updation	Take a subset of dataset for updation
Cost function reduces smoothly	Updation is not that smooth	Smoother cost function as compared to SGD
Computation cost is very high	Computation time is more	Computation time < SGD & Computation cost < BGD