Mini Batch Gradient Descent Method

Newton Luttah

March 2023

Mini-batch gradient descent is a popular optimization algorithm used in machine learning for training deep neural networks. It is a variant of the gradient descent algorithm that updates the model weights in small batches rather than updating the weights after every training example.

In mini-batch gradient descent, the training dataset is divided into small batches of fixed size. Each batch contains a subset of the training data, and the algorithm updates the model weights using the gradient of the loss function computed on that batch. The size of the batch is typically chosen based on the available memory resources and the computational efficiency of the algorithm. Common batch sizes are in the range of 32 to 256.

The algorithm updates the model weights using the following steps:

1.Initialize the model weights with random values.
2.Divide the training dataset into small batches of fixed size.
3.For each batch, compute the gradient of the loss function with respect to the model weights.
4.Update the model weights by subtracting a fraction of the gradient from the current weights. This fraction is called the learning rate and controls the step size of the update.
5.Repeat steps 3 and 4 for a fixed number of iterations or until convergence.

I'm going to attach a python code in GitHub to show how the mini batch gradient descent method works but this is just an overview of how the Python code works.

There is a function 'mini_batch_gradient_descent' which takes in the training examples 'x', target values 'y', learning rate 'alpha', number of passes over the training set 'epochs' and size of mini-batch 'batch_size'. It returns the optimal parameters 'theta' and the training loss at each epoch 'loss'.

The training process involves shuffling the training set and processing it

in mini-batches. For each mini-batch, the gradient of the cost function is computed and used to update the parameters.
The training loss is also computed at each epoch and stored in 'loss'.
The script generates synthetic data with 'm=1000' examples and 'n=10' features. The model is trained using 'mini_batch_gradient_descent' and the training time is printed.
The training loss over time is then plotted using 'matplotlib'.