Project Report

on

Parallelising Stochastic Gradient Descent

Submitted by

Divyansh Choudhary - 170001023
Yasasvi V Peruvemba - 170002061

Computer Science and Engineering

3rd year

Under the Guidance of

Dr. Surya Prakash

Department of Computer Science and Engineering

Indian Institute of Technology Indore

Autumn 2019

Problem Statement

To understand, implement and parallelize the Stochastic Gradient Descent used a popular Machine Learning Algorithm. Stochastic Gradient Descent (SGD) is an improvisation on the basic Gradient Descent algorithm which makes use of batch training.

Stochastic Gradient Descent

Stochastic gradient descent (often abbreviated SGD) is an iterative method for optimizing an objective function with suitable smoothness properties (e.g. differentiable or subdifferentiable). It can be regarded as a stochastic approximation of gradient descent optimization since it replaces the actual gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data).

Efficient Sum Algorithm

We first divided the input set into **n/log(n)** parts with each part having log(n) values. We then assigned these **log(n)** values to each processor (Hence total processors **O(n/log(n))**). We now perform sequential addition on all these log(n) values in O(log(n)) time. We then use the synchronous parallel addition algorithm to find the sum in **O(log(n/log(n))** time. This whole process gives us a time complexity of **O(log(n))**. Our algorithm for the sum is also work and cost optimised as the work is **O(n)** and the cost is **O(n)**.

The Algorithm

N is the total number of examples in the dataset.
B is the Batch Size.
K is the number of Batches.
M is the number of features.

The algorithm splits the dataset into batches and works on them **parallelly**.

For each split batch, the algorithm works on every example in the batch **parallelly** as well.

We calculate the loss using the *efficientSum* function, hence giving us a time complexity of O(logM) per example, run parallelly.

This loss is accumulated again using the *efficientSum* and this can be regarded as the **batch loss,** this takes a total time of O(logB).

Now, every **batch loss** is aggregated using *efficientSum* and we determine a final **approximate loss** which we use to update the **weights** used per feature. The time taken for this is O(logK).

This concludes the training for the entire batch in 1 iteration of the algorithm.

Complexity Analysis

Serial Execution :

The Time complexity for the algorithm is :

O(K*B*M)

And the space complexity is :

O(N*M)

N is the total number of examples in the dataset.

B is the Batch Size.

K is the number of Batches.

M is the number of features.

Parallel Execution :

The Time complexity for the algorithm is :

O(log(K*B*M)) = O(logB + logK + logM)

And the space complexity is

O(N*M)

Work Complexity

O(K*B*M)

The parallel algorithm is both Cost and Work Efficient.

N is the total number of examples in the dataset.

B is the Batch Size.

K is the number of Batches.

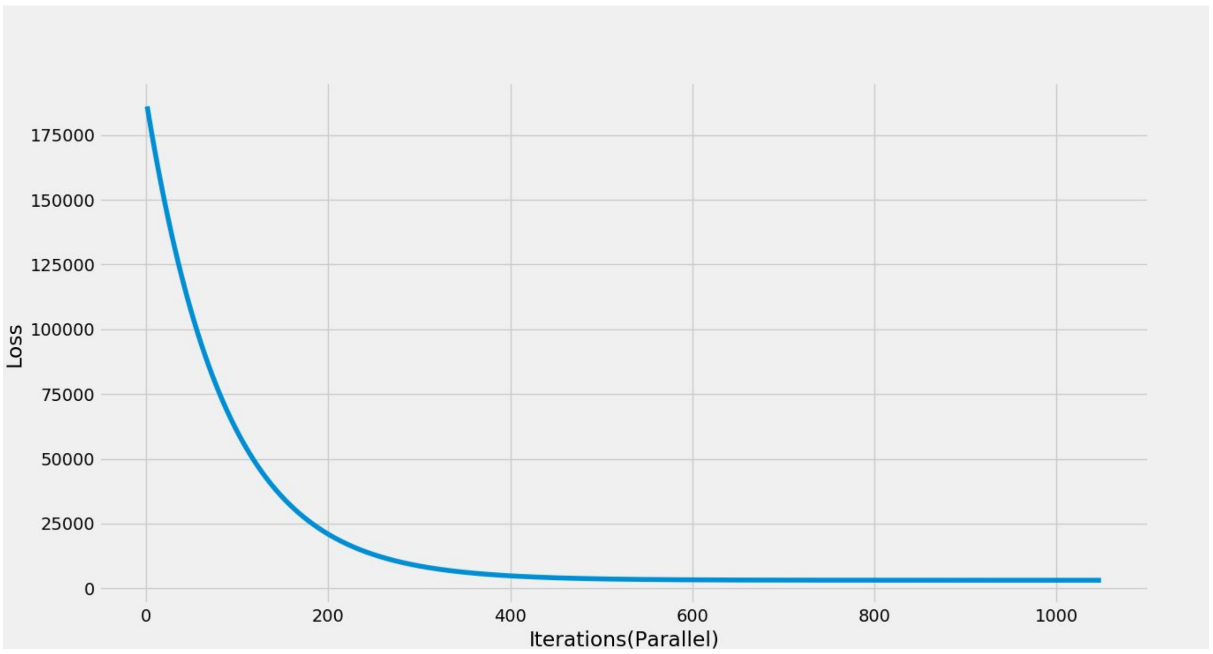
M is the number of features.

Serial vs Parallel Execution Time

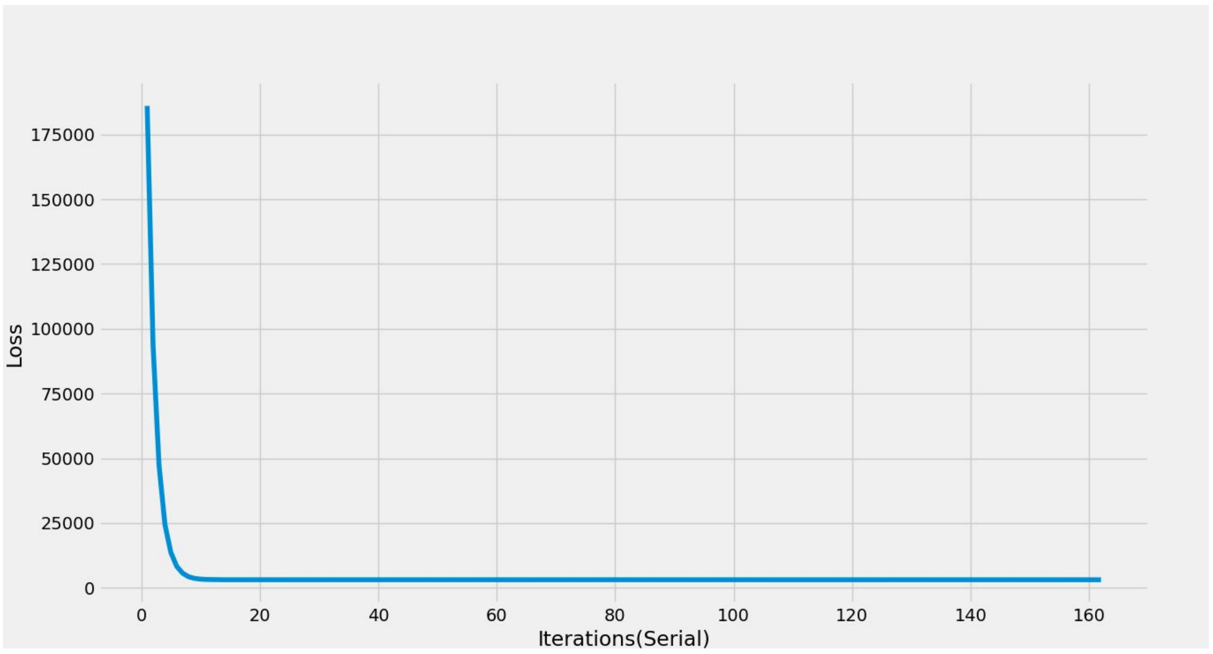
Time(µs)	**Parallel**	**Serial**
Server	413753	391310
PC	1173420	724180

Serial vs Parallel Loss Curves

Parallel Algorithm



Serial Algorithm



Since the serial algorithm updates weights after every batch, the rate of convergence to the minimal loss is much higher (due to more updates on weights). Hence, we can note from the graph that the serial algorithm approaches minimal loss in much fewer total iterations ~ 170. However, as the parallel algorithm updates weights after the entire iteration, i.e after accumulating gradients from the entire dataset, it takes more number of iterations and the loss curve isn't as steep as that of the serial algorithm.

Conclusion

We have a parallelized Stochastic Gradient Descent in a work and cost efficient manner with effective time complexity of O(logK + logB + logM). Our algorithm also converges to the same value as the best known sequential algorithm.

References

https://papers.nips.cc/paper/4006-parallelized-stochastic-gradient-descent.pdf

https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0179-2