Notes of《Divide-and-Shuffle Synchronization for Distributed Machine Learning》

1 introduction

1）简单介绍了分布式机器学习出现的背景

2）简单说明了其他基于数据并行的同步算法的思路，然后引出了本文的思路:降低同步的粒度

3）简单介绍了本文提出的算法(DS-Sync)的key factor，并和Ring all-reduce算法进行了简单对比

4）总结了本文的三点贡献

- 采用了通过降低同步粒度这一新的思路来解决分布式机器学习中的同步所有worker的瓶颈问题
- 保持了和BSP一样的收敛性能，并且泛化性更好
- 对本文提出的算法的收敛性以及同步粒度作了理论分析

2 related work

1）对已有工作的分类总结

- 通过设计更优的topology提升带宽利用率，典型的如Ring AllReduce, Tree Allreduce
- 减少通信内容或者降低通信频率，如对梯度进行压缩，只传递比较大的梯度；或者本地update多次后在进行一次全局update
- 半异步/完全异步并行（特别指出了此类算法的问题）

2）强调了本文的算法对比已有工作的优势，特别指出来本文方法与上述提到的方法和技巧是正交的，在同一个group内仍然可以使用这些方法和技巧以进一步提升performance

3 Methodology

3.1 DS-Sync Algorithm

给出了算法的详细步骤

0）随机将数据集平均分配给多个worker

1）每个worker根据本地数据计算梯度，并更新本地参数

2）交替分组

- 按顺序划分
- 交叉划分：要保证交叉划分的任何一个组要包含上一轮迭代中所有组的元素

这样的分组方式使得任一个worker每经过俩轮迭代就可以获得全局的更新

3）按组进行reduce：

同一组的worker会被更新为相同的参数

3.2 Synchronization Scale Analysis

1）给出了同步粒度的定义：

> Distributed Machine Learning Synchronization Scale: It is the number of inevitable serial executed and costly operations in synchronizing one element, typically one parameter of the model, among the needed workers.

2）分析对比了DS-Sync算法和PS,Ring-All-reduce,Tree-All-reduce的同步粒度和代价：

对比结论:DS-Sync的同步力度至少可以减半

3.3 Synchronization Scale Analysis

1）对于目标函数是凸函数的情况的收敛性给出了严格证明

2）对于非凸的情况，未给出严格证明，但是做了分析：结论是在非凸的情况，本文的算法也是work的

4 Experiments

详细介绍了实验部分

5 Conclusions and Future Works

1）对本文进行了总结

2）介绍了未来可能的工作方向

- micro scenarios：单节点/GPU中的多线程的并行
- macro scenarios：探索DS-Sync在联邦学习中的应用(上千节点)
- 研究不同的超参数，和梯度压缩，local SGD等方法结合

问题：没有考虑容错性