大数据分析系统 note

中国科学院大学大数据分析课程笔记系列

本节主讲老师：庞亮

笔记整理：@天空鱼(ict)

10.1 数据与计算的演变历程

1. 第一阶段（数据中心）----第二阶段（数据并行，百万级，传统 ML）----第三阶段（云计算，十亿级别）[特征 1：数据量大，数据多样性（结构化数据与非结构化数据如文本/图片...） 特征 2：DL 模型更复杂]
2. 挑战：

数据很难存储在同一个物理节点上，难以均匀采样得到数据整体分布
隐私性，安全性
大规模模型部署，实时处理非结构化数据等问题

一种解决方案：peer2peer paradigm

paper1: https://arxiv.org/ftp/arxiv/papers/1311/1311.3070.pdf

paper2: https://link.springer.com/chapter/10.1007/978-3-642-19542-6_63

3. 计算范式演变（回到了体系结构课程嘿嘿嘿）

单核 cpu 提升频率-->功耗开销制约性能上限
单机多核 cpu 并行-->Amdahl 定律制约加速上限
分布式系统

10.2 大数据分布式计算模型

1. 机器学习/深度学习算法与传统数据处理方式最大区别在于：ML/DL 解决的是一个优化问题，不受个别错误中间步影响 or 较小。只要方法选的对，总会收敛到极小值的。

2. 传统机器学习 fit in 大数据分析时的问题：

基于内存的算法需求大量的内存开销，如决策树/随机森林...（但是 xgboost 对数据加载做了优化）；而有些算法计算密集，开销大，如 SVM（recall: svm 在求超平面系数时会对样本点两个两个相乘，即 kernel-method，这个空间开销和时间开销都贼高）。
传统 ML 往往无法处理多样性、高维度、稀疏的大数据（比如，在高维空间里用 KNN/K-means 等往往效果不好，因为在高维空间中距离度量已几乎无作用: “dimension curse”）
有效的降维（PCA/K-rank 近似...）和有效的特征选择

3. 数据与模型分发策略

   既然采用分布式系统，首先数据要切割成块分发到不同的机器上训练。这样会有一个问题：每个机器在一次 epoch 中只能看到全部数据的一部分，没法学到完整的分布。一个解决办法是每个 epoch 之后把全体数据 shuffle 后重新分块分发。Hopefully, 多看几次总能看全吧!这个方法需要大量的数据传送，数据搬运开销大。

   在每台机器上训练后，每个机器对参数 W 都会生成一个"delta W"，即更新量。在优化问题中，正是因为全部的更新量可以分成小部分并行生成才有了分布式的可能。那么对于上一轮的参数 W，如何把每台机器的更新量收上来统一更新？这就涉及到两个问题：分布式架构设计和同步策略。我们放在下一个内容。

   以上讨论的是数据并行的分发策略。

   至于模型并行~~(就是个玄学)~~，需要把一个大模型，如神经网络的全部节点（或者计算流图里的广义结点），切割成几部分发到不同的机器上做训练。这里必须要维护节点之间的连接，在前传和后传时计算流图不能错。当然维护起来非常困难。如果再加点其他机制进去，比如残差/attention，维护起来可能更加费劲。

4. 分布式架构设计

   首先来看 MapReduce 架构。这里不谈详细的实现步骤(mapper-conbiner/shuffler-reducer)，它的灵魂在于生成<key, value>对，之后对同一个 key 计数。回到机器学习的优化问题，如 SGD，它只是一个求累和 accumulate 的过程，没有 key-value 的天然结构（当然，你可以给每一个更新量起一个 Id 作为 key。但是这样会导致所有的 key 都不同，就没什么用。）

   第二代架构就是参数服务器 parameter-server。它维护一个参数 W 的副本。它从所有的机器上把更新量收上来做更新【1】，之后把更新后的参数分发到机器上做下一轮训练。Google 的 DistBelief 架构就基于此。它是目前最好的大数据分析系统模型。

   Spark 跳过，和 mapreduce 差不多。

5. 同步策略

   上文【1】中提到了"把所有机器的更新量都收上来做更新"。在分布式集群中必然存在着速度快和速度慢的机器，那么提交给参数中心服务器的更新量也就有前有后。这就涉及到同步策略，本质上是在高性能和快速收敛之间的 trade-off。

BSP	设置一个 barrier，即每个 epoch 等到最慢的机器提交完更新量后统一更新。当然，这样性能很好，就是速度慢，worker 浪费时间等待
ASP	谁先训练完，谁就马上提交给中心服务器，服务器马上更新并把更新结果发给它。但是由于单计算节点只看到部分数据，用部分数据做更新不准确（想象一下梯度下降时往错误的方向迈了一步）。这样做会偏向于速度快的机器，因为当慢机器提交时快机器可能已经更新了好多次。用快机器学习的参数再应用到慢机器上，效果往往不好。
SSP	Balance BSP&ASP

6. TensorFlow

   2.0 版本已经采用动态图。并用 keras 作为后端。