# Model Optimization

Taylor Guo, 2020年12月08日 @ Biren上海

目录

[Model Optimization 1](#_Toc58372869)

[Auto-Precision Scaling for Distributed Deep Learning 1](#_Toc58372870)

[提出问题： 1](#_Toc58372871)

[解决方法： 2](#_Toc58372872)

[APS算法思路与伪代码： 3](#_Toc58372873)

[技术细节： 4](#_Toc58372874)

[代码分析： 5](#_Toc58372875)

## Auto-Precision Scaling for Distributed Deep Learning

<https://arxiv.org/abs/1911.08907> <https://arxiv.org/pdf/1911.08907.pdf>

### 提出问题：

1. 分布式训练的一个主要问题是梯度同步成为瓶颈，限制了训练的规模。低精度梯度可以减少带宽。

自动精度调整(Auto Precision Scaling, APS)算法可以在以低精度浮点值传递时改善准确度。它可以改善精度的准确度而使用较低的传递成本。图像分类和分割测试中用8位浮点梯度准确率降低<0.05%。8个Nvidia GTX1080Ti训练BERT/Wikipedia可能会占用40%的时钟时间。

2. 用低精度梯度会损害推断准确率，训练无法收敛。低精度容易下溢或上溢。有很多0值和INF值，训练无法收敛。

### 解决方法：

1. APS方案：APS是一个高效的梯度传输基于网络层级的自适应算法。它可以是大批次训练在8位甚至4位符号类型收敛。

APS，网络层级自适应算法，改善低精度下的准确率；

可以在8节点的分布式系统中训练8bit的分类和分割模型；

可以在256节点的分布式系统中，8bit梯度训练ResNet-50；

可以应用于任意低精度浮点运算；

2. 其他方案：

梯度同步的优化有好几种方法:

a. 特殊的网络拓扑结构：

多组三阶段实现all-reduce操作；

通常的做法是计算前一层网络的梯度，然后一起传输所有计算好的梯度。

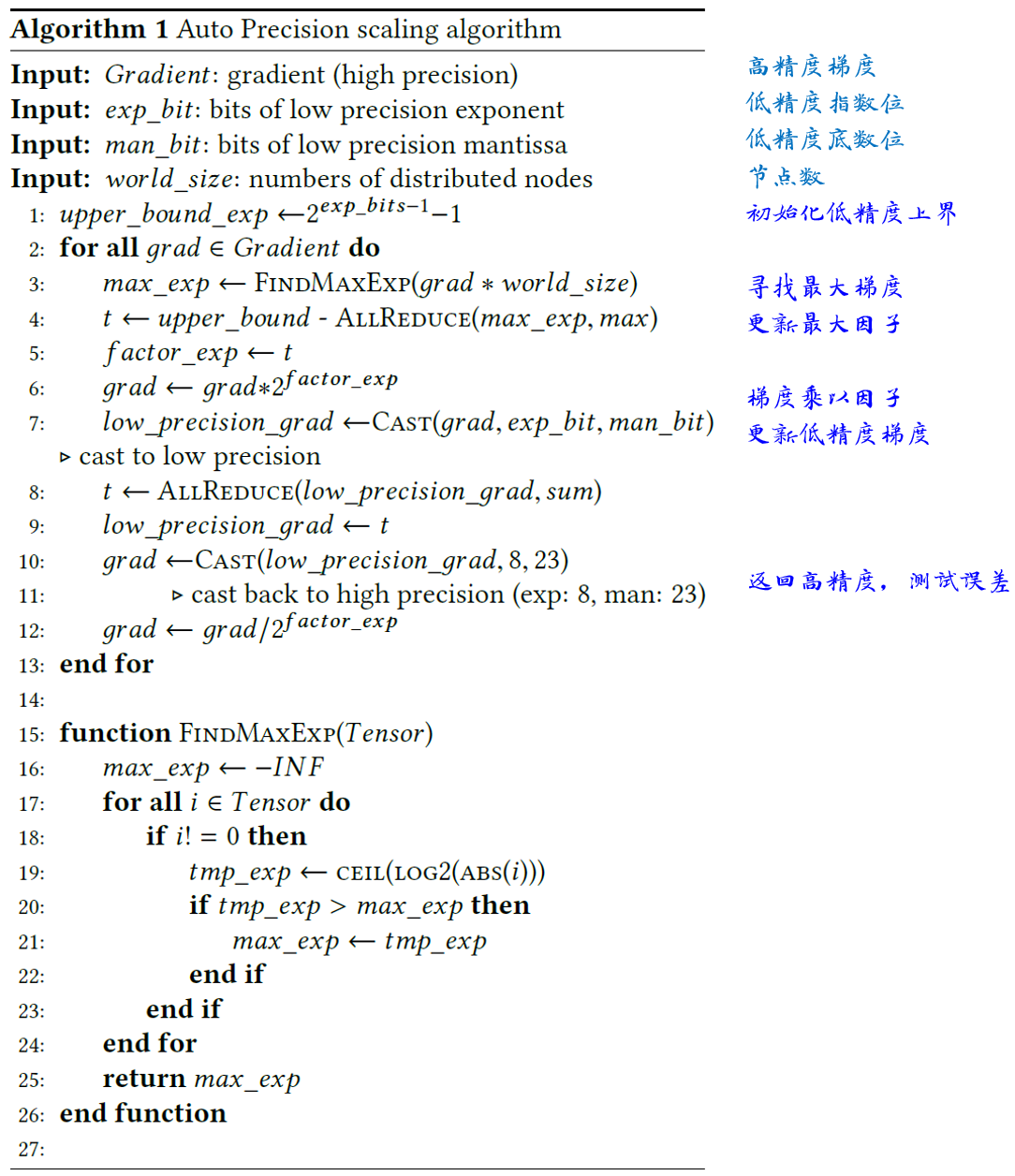
b. 梯度稀疏化

对梯度设置阈值，每次迭代只传输大于阈值的梯度；设置阈值的方法有测量每个梯度的重要性的L1范数；DGC方法累加局部梯度。

c. 梯度量化：

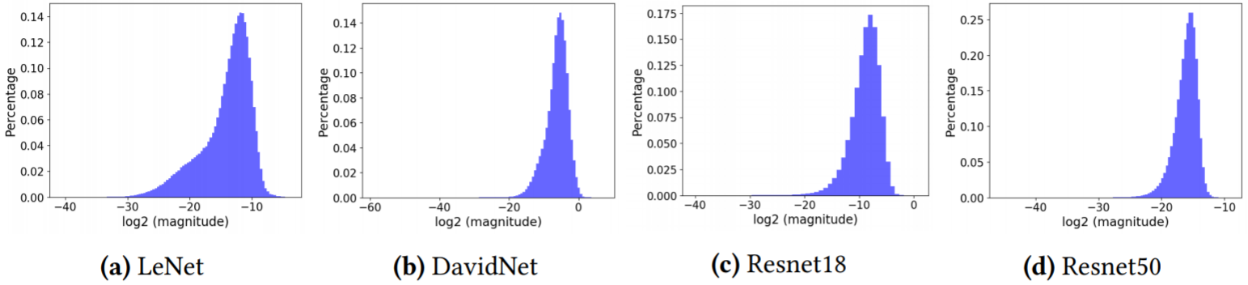
缩减损失值来缩减梯度值

### APS算法思路与伪代码：

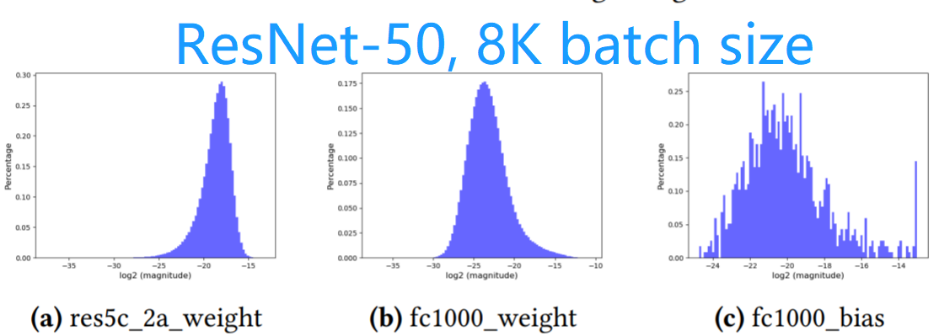


#### 1. 损失缩放的局限

需要寻找合适的损失缩放因子，不同模型的梯度分布非常不同。



相同的模型，不同层的梯度分布非常不同。

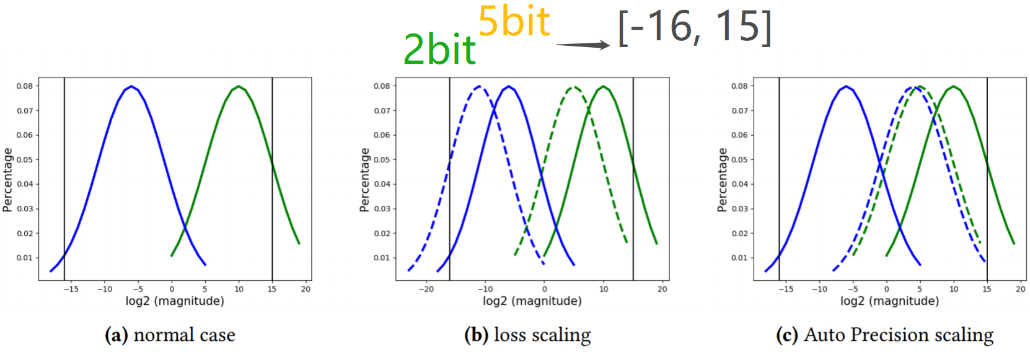


训练过程中单层梯度分布的变化，会使得损失缩放算法在实践当中极不稳定。

#### 2. 层级精度调整梯度

同步层的梯度，先计算最大绝对值梯度和阶码；然后用all-reduce在整个分布式系统上获得全局最大值；再根据这个全局最大梯度值将局部最大梯度移位变成低精度计数。n层的阶码最大值获得之后，将每层都移位为低精度。所有低精度处理完后，再移位回高精度的阶码。在实践中，同步梯度，可以将后续几层的梯度做为一个张量加速通信减少延迟。

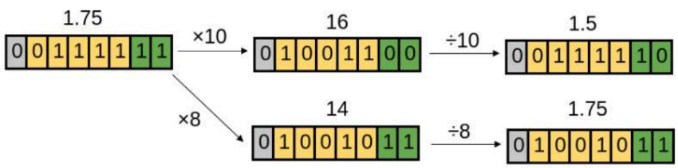
损失缩放对所有层的梯度使用固定值，手工选值防止溢出。可以避免上溢，但可能导致小值产生下溢，被置为0。APS每层用不同的值，也就是会自动选择缩放因子不会导致上下溢。



### 技术细节：

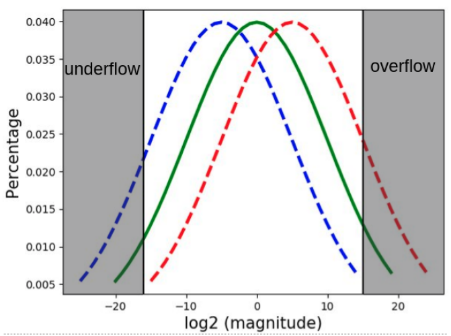
#### 1. APS使用2的幂次方作为缩放因子

可以减少舍入误差。比如下面的5位阶码，2位底数。



使用10和8作为缩放因子的区别，灰色部分是符号位，黄色部分是幂，绿色部分是底。因子是2的n次幂时，底是不变的，只是阶码发生变化。如果是10的话，阶码和底都会发生变化，可能会截断数值。乘以10或除以10都会导致舍入误差。

#### 2. 下溢或上溢的选择



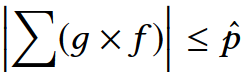
用低位来表示高位会产生下溢或者上溢。

缩放因子大于1绿色曲线会移向红色曲线，产生上溢；小于1，则移向蓝色曲线，产生下溢。但对深度神经网络来说，上溢经常比下溢危害更大。反向传播的时候，后面的网络层的梯度是基于前面的网络层的梯度来计算的。如果后面的网络层上溢，被置为INF，所有之前的网络梯度都会被置为INF。如果有INF参与运算，那么算子的输出都为INF。这样，整个训练就无法收敛，就会丢失重要信息。

如果采用对小梯度设置阈值进行丢弃算法的话，局部梯度累加超过阈值再进行通信传递。但对APS算法，我们会丢弃小的梯度值，这些值可能会下溢。比如，如果我们用8位来处理，5位阶码，2位底数，当乘以因子的时候，最大的梯度大概是215，这已经是该精度的上界了。那么，小于下界2-16就会溢出。如果采用阈值的方法，就会被大量累加，效果类似达到最大梯度。同时，阶码变大，底数不变。由于只有2位底数，那么这个值将是4倍，会导致舍入错误，小梯度应该被丢弃。在这些值里，选择最大的，就会使最小的掉到下溢范围内。

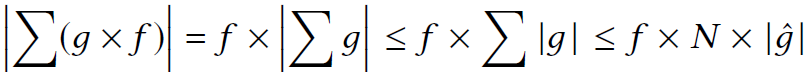
#### 3. 寻找最大缩放因子

选择最大缩放因子来避免上溢。

 ❶

g是梯度，f是放大因子， 特定精度的上界；

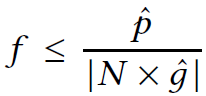
每个节点只知道局部梯度，很难得到最大因子；所以APS，放松了等式的边界；

 ❷

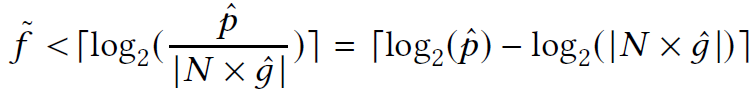
N是节点数；

比较直觉的方法就是只传递每个节点的局部最大梯度，从而获得全局最大梯度来计算因子。

为了进一步优化，❷带入❶等式又可以写为❸：

 ❸

如果只用2幂次方作为因子，❸可以变为：

那么，给定精度要求，log对数值就等于阶码，所以只需要通信这个值就可以用8位来传递阶码。

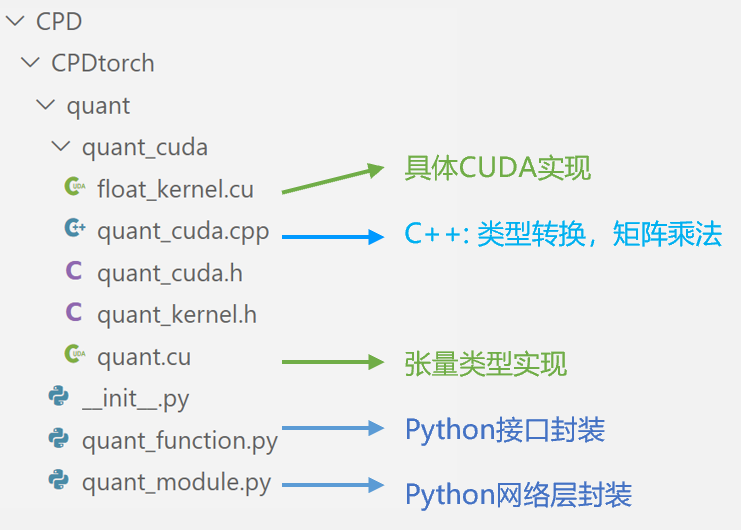
### 实验结论：

分布式训练系统中的LARS(Layer-Wise Adaptive Rate Scaling 分层自适应学习率: 权重2范数除以梯度2范数，再乘原学习率)和低精度算法是否会影响准确率：会；但APS不会。

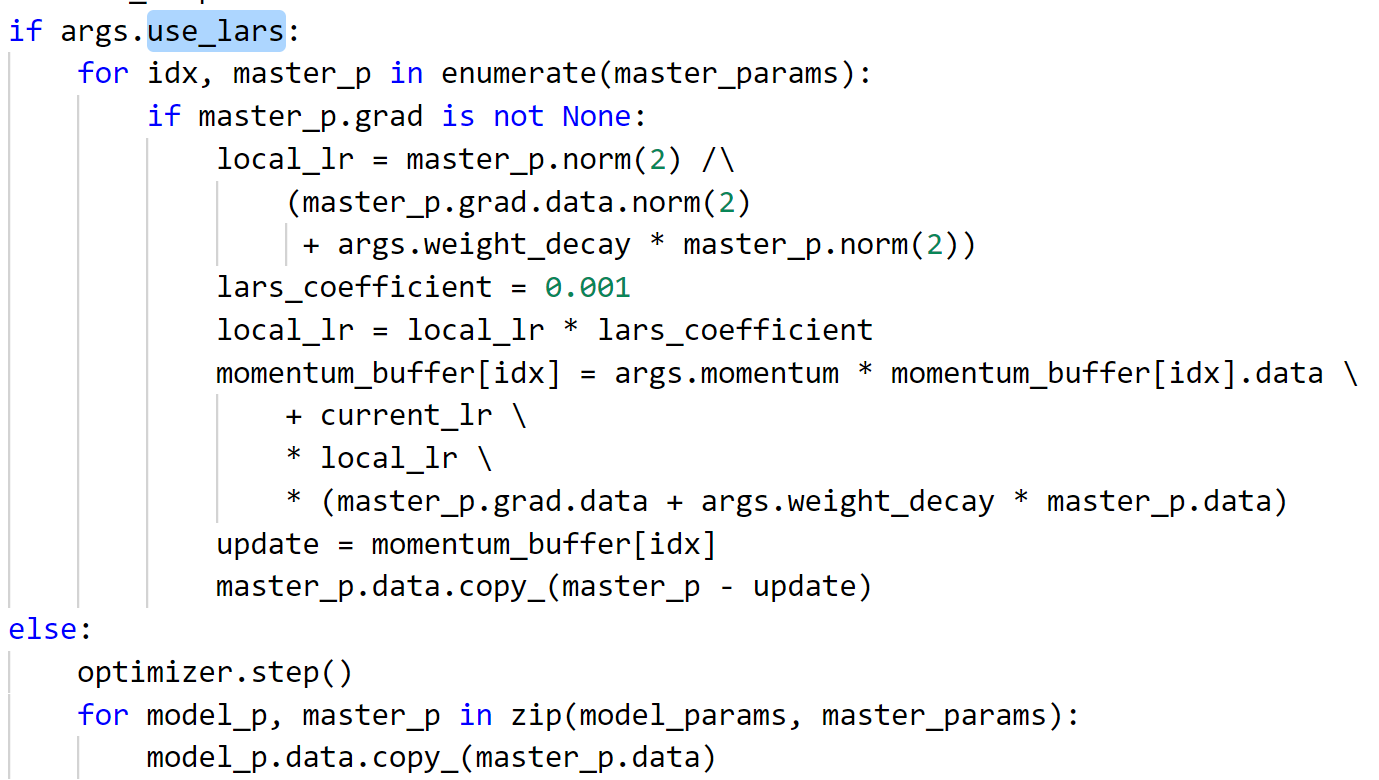
定制化低精度深度学习：

* 任意低精度 -> 阶码<=8bit; 底数<=23bit；
* Kahan summation 算法：补偿截断误差，提高低精度梯度累加更新与矩阵乘法的准确率；
* 低精度加法器计算GEMM；
* 低精度reduce/all-reduce

### 代码分析： [CPD Code](https://github.com/drcut/CPD)



**训练过程中的LARS：**



**Kahan加法去精度误差：**

