大数据算法-2025 春

Lecture 18: Mean Estimation

2024.6.3

Lecturer: 丁虎 *Scribe:* 张嘉贤

在训练大模型的时候，SGD（随机梯度下降）是一种简单有效的优化算法。SGD 的关键是计算得到模型参数关于训练数据的梯度信息，然后根据梯度信息更新模型参数。在分布式的场景下，每一个分布式节点都有一个模型的副本，各个节点根据自己的数据计算模型的梯度信息，然后将各自的梯度信息传输给中心服务器。中心服务器根据这些梯度信息求一个均值（或者加权均值），然后根据均值（或者加权均值）更新模型参数，并将模型参数分发给子节点。在这个场景里，节点和中心服务器的通信开销是巨大的，这是制约模型训练的瓶颈。因此，各个子节点如何尽可能的压缩梯度信息，并且使得中心服务器根据这些压缩后的梯度尽可能准确的计算梯度均值是一个重要的问题，这个问题就是本节要讨论的问题，Mean Estimation，即重心估计。

1 问题的定义

模型描述 假设在 d 维欧式空间 R^d 中存在 n 个向量 $\{x_1, x_2, ..., x_n\}$，每一个向量是一个分布式节点上的梯度信息向量。每一个节点 i 通过函数 f 来压缩自己的梯度向量为 $f(x_i)$，中心服务器根据函数 g 来解压缩每一个节点传来的梯度向量信息 $g(f(x_i))$。

目标 使得通信开销 $\sum_{i=1}^{n} f(x_i)$ 尽可能小，同时尽可能减小重心的期望估计误差

$$minE(||\hat{\bar{X}} - \bar{X}||_2^2)$$

其中 $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} x_i$ 表示精确的重心，$\hat{\bar{X}} = \frac{1}{n}\sum_{i=1}^{n} g(f(x_i))$ 表示估计的重心

如果一个实数使用 L bits 来表示，那么对向量信息不进行任何处理，直接传输的通信开销为 $\mathcal{O}(ndL)$

Remark 1.1. 对于这个问题，一个直接的想法是 JL 变换，它可以压缩节点的向量信息，降低通信开销。但是，由于 JL 变换只具有压缩功能，我们无法根据 JL 变换的压缩结果解压出原始的向量信息，因此就无法估计向量的重心，故在这个问题里 JL 变换是不可行的。

2 Stochastic Binary Quantization 算法

基本思路 Stochastic Binary Quantization 算法，即随机二值量化算法，其基本思路是将每一个实数值随机量化为两个数值 t_1, t_2 中的一个，因此对于一个 d 维向量，我们只需要存储 d 个 01 变量，表示他们是 t_1, t_2 中的哪一个即可。这样，我们便可以降低存储向量的代价，进而降低通信开销。

算法过程 对于任何一个 x_i, 令

$$x_i^{max} = max_{1 \le j \le d} x_i(j)$$

$$x_i^{min} = min_{1 \le j \le d} x_i(j)$$

其中 $x_i(j)$ 表示向量 x_i 的第 j 位。根据这个计算 y_i 如下:

$$y_i(j) = \begin{cases} x_i^{max} & \text{以概率} \dfrac{x_i(j) - x_i^{min}}{x_i^{max} - x_i^{min}} \\ x_i^{min} & \text{以概率} \dfrac{x_i^{max} - x_i(j)}{x_i^{max} - x_i^{min}} \end{cases}$$

于是我们将 x_i 通过上面的方法压缩成了 y_i, 即 $y_i = f(x_i)$

算法分析 接下来我们分析算法得到的估计的重心的准确性

$$E(y_i(j)) = x_i^{max} \frac{x_i(j) - x_i^{min}}{x_i^{max} - x_i^{min}} + x_i^{min} \frac{x_i^{max} - x_i(j)}{x_i^{max} - x_i^{min}} = x_i(j)$$

于是得到:

$$E(\hat{\bar{X}}) = \frac{1}{n} \sum_{i=1}^{n} y_i = \bar{X}$$

$$
\begin{aligned}
E(||\hat{\bar{X}} - \bar{X}||_2^2) &= \frac{1}{n^2} E(||\sum_{i=1}^{n}(y_i - x_i)||_2^2) \\
&= \frac{1}{n^2}\sum_{i=1}^{n} E(||y_i - x_i||_2^2) \\
&= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{d} E((y_i(j) - x_i(j))^2) \\
&= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{d}(x_i^{max} - x_i(j))(x_i(j) - x_i^{min}) \\
&\leq \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{d}\frac{(x_i^{max} - x_i^{min})^2}{4} \\
&\leq \frac{d}{2n^2}\sum_{i=1}^{n}||x_i||_2^2
\end{aligned}
$$

梯度向量的维度 d 表示模型的参数数量，一般来说，模型的参数数量是巨大的，因此，d 一般比较大，远远大于 n，所以这个算法得到的 bound 是一个非常粗的的 bound

该算法对于每一个向量 x_i, 需要存储两个实数 x_i^{max} 和 x_i^{min}, 这需要 $2L$bits, 还需要存储一个 d 维 01 向量表示每一维，因此需要 $d + 2L$bits 的存储空间，一共有 n 个向量，因此总共需要 $\mathcal{O}(n(d + 2L))$ 的存储空间。因此通信开销也就是 $\mathcal{O}(n(d + 2L))$

3 Stochastic K-level Quantization 算法

Stochastic Binary Quantization 算法将每个向量的每一维都映射成两个值，x_i^{max} 和 x_i^{min}，这导致算法估计的重心与精确重心误差较大，一个直接的改进方法是，将 Binary 改进为 K-level，即将每个向量的每一维都映射成更细化的多个等级的值，这就是 Stochastic K-level Quantization 算法的主要思想

算法过程 对于任何一个 x_i, 令

$$x_i^{max} = max_{1 \leq j \leq d} x_i(j)$$

$$x_i^{min} = min_{1 \leq j \leq d} x_i(j)$$

假设我们将该向量的每一维都映射成 k 个等级的值中的一个,那么相当于将区间 $[x_i^{min}, x_i^{max}]$ 均匀的划分成 $k-1$ 个子区间，假设 $x_i(j)$ 落在第 t 个子区间内，其中 $t \in \{0, 1, 2, ...t-1\}$，那么根据这个计算 y_i 如下：

$$y_i(j) = \begin{cases} x_i^{min} + t\dfrac{S_i}{k} & \text{以概率} \dfrac{x_i^{min} + (t+1)\frac{S_i}{k} - x_i(j)}{\frac{S_i}{k}} \\ x_i^{min} + (t+1)\dfrac{S_i}{k} & \text{以概率} \dfrac{x_i(j) - (x_i^{min} + t\frac{S_i}{k})}{\frac{S_i}{k}} \end{cases}$$

其中 $S_i = x_i^{max} - x_i^{min}$ 是区间长度,于是我们将 x_i 通过上面的方法压缩成了 y_i,即 $y_i = f(x_i)$

经过 K-level 改进后，该算法估计的重心错误率 bound 为 $\frac{d}{2n^2k^2}\sum_{i=1}^{n}||x_i||_2^2$，即：

$$E(||\hat{\bar{X}} - \bar{X}||_2^2) \leq \frac{d}{2n^2k^2}\sum_{i=1}^{n}||x_i||_2^2$$

改进后，每一个向量的每一维需要记录它属于 k 个等级的值的哪一个，因此需要 $\log k$ 的存储空间，故算法总共的通信复杂度为 $\mathcal{O}(n(d\log k + 2L))$

4 Stochastic Rotated Quantization 算法

该算法基于一个几何上的结论：假设 $||x_i||_2^2 = 1$，x_i 在 d 维单位球上随机地旋转，那么以很高的概率有 $x_i^{max} - x_i^{min} = \mathcal{O}(\sqrt{\frac{\log d}{d}})$

回忆之前讲过的 FJLT 的内容，我们当时为了加速 JL 变换，希望变换矩阵 P 是稀疏的，然而，当 P 是稀疏的，待变换的输入向量也是稀疏的时候，变换的结果会产生较大的误差，因此，我们希望输入的向量是稠密的。但是，输入向量的稀疏与否不可控制，因此我们可以在输入向量进行 JL 变换之前乘以一个随机旋转矩阵，依赖于这个几何结论，可以使得输入向量变得稠密，从何实现加速 JL 变换过程，又不显著增加误差的目的。在 FJLT 中，我们使用的随机旋转矩阵是 H · D，其中 H 是 Hadamard 矩阵，D 是一个随机对角方阵。（详细构造方法可以参考 FJLT 章节的内容）

类似的，我们也可以将向量 x_i 进行随机旋转，这样可以使得向量变得稠密，从而减小 $x_i^{max} - x_i^{min}$，根据之前的计算重心估计的误差的过程，我们于是可以减少重心估计的误差

算法过程 我们首先获得一个随机旋转矩阵 R，对于任何一个 x_i,将随机旋转矩阵 R 作用到 x_i 上，得到 Rx_i,然后，对 Rx_i 向量采用 Stochastic K-level Quantization 算法得到 y_i，并

将量化的结果 y_i 传输给中心服务器。中心服务器首先将 y_i 逆旋转得到 $R^{-1}y_i$，然后计算重心为

$$\hat{\bar{X}} = \frac{1}{n}\sum_{i=1}^{n} R^{-1}y_i$$

由于该算法在传输数据时，传输的是 K-level 算法的结果，因此通信复杂度和 Stochastic K-level Quantization 算法一样，都是 $\mathcal{O}(n(d\log k + 2L))$，但是由于随机旋转有

$$x_i^{max} - x_i^{min} = \mathcal{O}(\sqrt{\frac{\log d}{d}})$$

而我们之前在计算 Stochastic Binary Quantization 算法估计的重心误差 bound 时，有

$$E(||\hat{\bar{X}} - \bar{X}||_2^2) =\leq \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{d}\frac{(x_i^{max} - x_i^{min})^2}{4}$$

因此，该算法估计的重心误差 bound 为 $\mathcal{O}(\frac{\log d}{n^2k^2}\sum_{i=1}^{n}||x_i||_2^2)$，即：

$$E(||\hat{\bar{X}} - \bar{X}||_2^2) \leq \mathcal{O}(\frac{\log d}{n^2k^2}\sum_{i=1}^{n}||x_i||_2^2)$$

参考文章 [1]。

References

[1] A. T. Suresh, X. Y. Felix, S. Kumar, and H. B. McMahan. Distributed mean estimation with limited communication. In *International Conference on Machine Learning*, pages 3329–3337. PMLR, 2017.