GPTQ: ACCURATE POST-TRAINING QUANTIZATION FOR GENERATIVE PRE-TRAINED TRANSFORMERS

ICLR 2023

Shared by: **Jiaqi Zhao**

2023.11.21

Introduction:

大模型庞大的参数量在实际应用场景下对硬件要求极高，如何在应用场景下降低模型大小、加快推理速度并保持足够的精度是重要的研究方向。

模型量化:

一种有效的模型压缩方法，通过将网络的权值（weights）、激活值（activations）等由浮点数（如 Float32）转换为低比特数据（如 Int8、Int4）进行计算实现网络瘦身和加速。

量化方法:

QAT（Quantization-Aware Training）：利用训练数据重新训练网络，不实用；

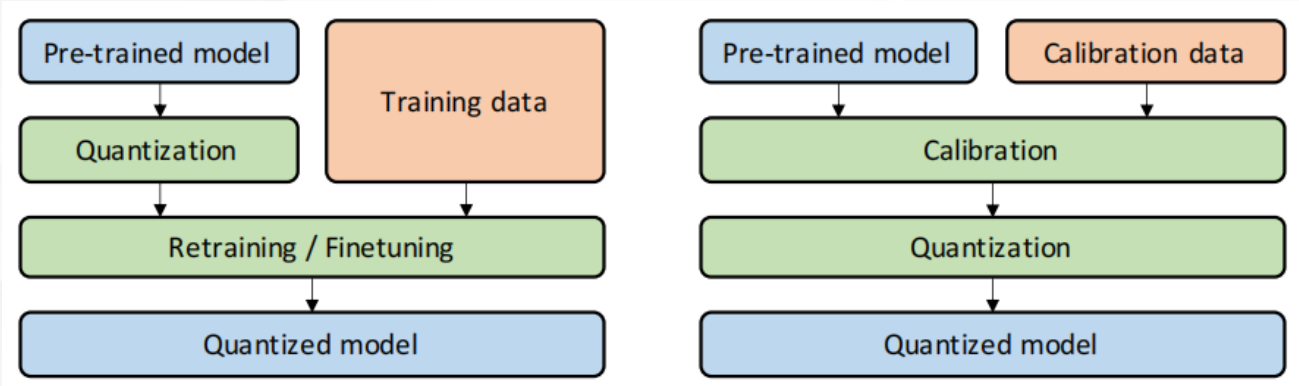
PTQ（Post-Training Quantization）：利用校准数据获得量化校准参数（或不需要），更常用。

排名全球第二！我国10亿参数规模以上大模型已发布近80个

奔流新闻 2023-11-09 19:34

人工智能技术的突飞猛进发展，为我们的生活带来巨大改变和机遇。据中国信息通信研究院测算，2022年中国人工智能核心产业规模达5080亿元人民币。目前，10亿参数规模以上的大模型已发布近80个。

据科技部"新一代人工智能发展研究中心"近期发布的《中国人工智能大模型地图研究报告》显示，中国研发的大模型数量排名全球第二，仅次于美国，目前中国10亿参数规模以上的大模型已发布79个。



创新点：

1. 提出一种新的PTQ量化算法GPTQ。该算法在最多几小时内完成千亿参数量大模型的量化，并不损失太多精度。

2. 在一些极端量化任务中（如2-bit 量化）表现出色；此外，首先实现了在一块A100或两块A6000上运行量化后的OPT-175B大模型。

3. 第一个证明了千亿参数量的大模型可以被量化至3/4-bit。

OBS （Optimal Brain Surgeon）：

一种模型剪枝方法，目的是找到并剪枝对模型损失函数影响最小的参数。

某参数位置发生参数变化带来的损失函数变化可表示为：

$$\Delta E = L(W + \Delta W) - L(W) = g^T \Delta W + \frac{1}{2} \Delta W^T H \Delta W + O(||\Delta W||^3)$$

其中*gT*表示损失函数对当前位置参数的梯度，*H*为海森矩阵，即损失函数对参数的二阶导。假设在模型已经充分收敛的情况下进行剪枝，则参数的一阶导为0。再忽略高阶项，简化为：

$$\Delta E = \frac{1}{2} \Delta \mathbf{w}^{\mathrm{T}} \mathbf{H} \Delta \mathbf{w}$$

假设对位置q的参数w_q进行剪枝，则可以表示为在当前位置参数变化ΔW_q为$-w_q$，进一步表示为等式约束条件：

$$\Delta W_q + w_q = 0$$

结合最初目的，找到合适参数使得剪枝该参数对损失函数的影响最小，可表示为最优化问题：

$$\min_{\mathbf{\Delta w},q} \frac{1}{2}\mathbf{\Delta w}^{\mathrm{T}}\mathbf{H}\mathbf{\Delta w} \quad s.t. \quad \mathbf{e_q^T}\cdot\mathbf{\Delta w}+w_q=0$$

其中$\mathbf{e_q}$为一个one-hot形式的向量，在第q维是1，其余位置为0。

求解该最优化问题，利用拉格朗日乘子法：

$$L=\frac{1}{2}\mathbf{\Delta w}^{\mathrm{T}}\mathbf{H}\mathbf{\Delta w}+\lambda(\mathbf{e_q^T}\cdot\mathbf{\Delta w}+w_q)$$

求得剩余权重的更新公式（左）及对剪枝该参数对损失函数的影响公式（右）：

$$\boldsymbol{\delta_p}=-\frac{w_q}{[\mathbf{H}^{-1}]_{qq}}\cdot\mathbf{H}^{-1}_{:,q}, \qquad L=\frac{1}{2}\frac{w_q^2}{[\mathbf{H}^{-1}]_{qq}}$$

其中，$[\mathbf{H}^{-1}]_{qq}$为海森矩阵逆矩阵在qq对角线位置的元素值，$H^{-1}_{,:q}$代表矩阵的第q列。

最后，求取海森矩阵逆矩阵，算得每个参数的影响大小并获得剪枝顺序，剪枝一个参数后再更新剩余所有参数，弥补剪枝造成的损失。

OBQ （Optimal Brain Quantization）：

OBS算法需要计算全参数的海森矩阵及其逆矩阵，计算量过大。

针对剪枝的OBC算法首先改写了损失函数：

$$\|\mathbf{W}_\ell \mathbf{X}_\ell - \widehat{\mathbf{W}}_\ell \mathbf{X}_\ell\|_2^2 \longrightarrow \sum_{i=1}^{d_{\text{row}}} \|\mathbf{W}_{\mathbf{i},:}\mathbf{X} - \widehat{\mathbf{W}}_{\mathbf{i},:}\mathbf{X}\|_2^2.$$

这种改写方式可看作将权重矩阵每行的方差求和，因此剪枝掉某个参数仅对当前行的目标函数产生影响，而各行之间相互独立。对改写后的损失函数对参数求海森矩阵，可得 $\mathbf{H} = 2\mathbf{X}\mathbf{X}^\top$。

另外，由于剪枝掉了一个参数，相应地会删除海森矩阵的行和列，因此根据高斯消元法，可得到近似的剪枝后的海森矩阵逆矩阵的简单求法：

$$\mathbf{H}_{-p}^{-1} = \left(\mathbf{H}^{-1} - \frac{1}{[\mathbf{H}^{-1}]_{pp}} \mathbf{H}_{:,p}^{-1} \mathbf{H}_{p,:}^{-1}\right)_{-p}$$

OBQ算法认为，剪枝是量化的特殊情况。q位置的参数量化后的参数变化为：

$$\mathbf{e_q^T} \cdot \mathbf{\Delta w} + w_q = quant(w_q)$$

相应地，参数更新公式转换为：

$$\boldsymbol{\delta_p} = -\frac{w_p - \text{quant}(w_p)}{[\mathbf{H}^{-1}]_{pp}} \cdot \mathbf{H}^{-1}_{:,p}.$$

设M为pruning mask，OBQ算法流程的伪代码为：

Algorithm 3 Quantize $k \leq d_{\text{col}}$ weights from row $\mathbf{w}$ with inverse Hessian $\mathbf{H}^{-1} = (2\mathbf{X}\mathbf{X}^\top)^{-1}$ according to OBS in $O(k \cdot d_{\text{col}}^2)$ time.

$M = \{1, \ldots, d_{\text{col}}\}$
for $i = 1, \ldots, k$ **do**
  $p \leftarrow \text{argmin}_{p \in M} \frac{1}{[\mathbf{H}^{-1}]_{pp}} \cdot (q(w_p) - w_p)^2$
  $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{H}^{-1}_{:,p} \frac{1}{[\mathbf{H}^{-1}]_{pp}} \cdot (w_p - q(w_p))$
  $\mathbf{H}^{-1} \leftarrow \mathbf{H}^{-1} - \frac{1}{[\mathbf{H}^{-1}]_{pp}} \mathbf{H}^{-1}_{:,p} \mathbf{H}^{-1}_{p,:}$
  $M \leftarrow M - \{p\}$
end for

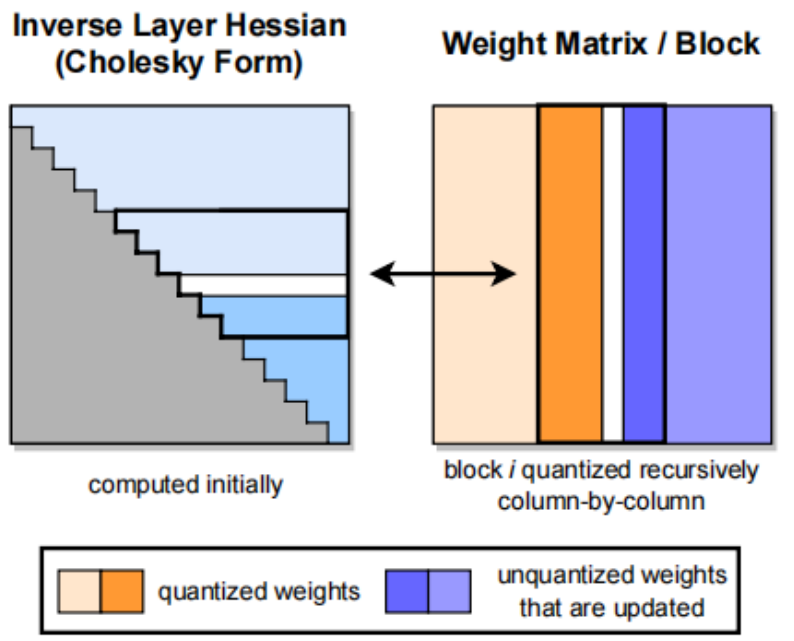
GPTQ：

GPTQ算法是OBQ算法针对大模型量化的改进。

GPTQ算法认为OBQ选择最优参数的贪心算法过程对于大模型来说提升有限，但大大增加了计算量，因此采用对固定位置参数进行量化，提高效率。

GPTQ算法采用了Lazy Batch Updates策略。一次处理权重矩阵的128列（一个block），减小内存访问频率提升效率。

GPTQ算法采用了Cholesky分解处理海森矩阵逆矩阵，增强数值稳定性减小误差积累，同时无需再更新海森矩阵逆矩阵，提升效率。



$$\delta_F = -\frac{w_q - \text{quant}(w_q)}{[\mathbf{H}_F^{-1}]_{qq}} \cdot (\mathbf{H}_F^{-1})_{:,q}.$$

$$\mathbf{H}_{-q}^{-1} = \left(\mathbf{H}^{-1} - \frac{1}{[\mathbf{H}^{-1}]_{qq}} \mathbf{H}_{:,q}^{-1} \mathbf{H}_{q,:}^{-1}\right)_{-p}.$$

Algorithm 1 Quantize $\mathbf{W}$ given inverse Hessian $\mathbf{H}^{-1} = (2\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}$ and blocksize B.

$\mathbf{Q} \leftarrow \mathbf{0}_{d_{\text{row}} \times d_{\text{col}}}$ *// quantized output*
$\mathbf{E} \leftarrow \mathbf{0}_{d_{\text{row}} \times B}$ *// block quantization errors*
$\mathbf{H}^{-1} \leftarrow \text{Cholesky}(\mathbf{H}^{-1})^\top$ *// Hessian inverse information*
for $i = 0, B, 2B, \ldots$ **do** 将H逆分成多个block
  for $j = i, \ldots, i + B - 1$ **do** 在block内进行下列计算
    $\mathbf{Q}_{:,j} \leftarrow \text{quant}(\mathbf{W}_{:,j})$ *// quantize column* 量化j列的权重
    $\mathbf{E}_{:,j-i} \leftarrow (\mathbf{W}_{:,j} - \mathbf{Q}_{:,j}) / [\mathbf{H}^{-1}]_{jj}$ *// quantization error* 计算量化误差，E矩阵的列数与block数相同
    $\mathbf{W}_{:,j:(i+B)} \leftarrow \mathbf{W}_{:,j:(i+B)} - \mathbf{E}_{:,j-i} \cdot \mathbf{H}^{-1}_{j,j:(i+B)}$ *// update weights in block* 计算完一列的量化误差后，更新block内的全部参数
  end for
  $\mathbf{W}_{:,(i+B):} \leftarrow \mathbf{W}_{:,(i+B):} - \mathbf{E} \cdot \mathbf{H}^{-1}_{i:(i+B),(i+B):}$ *// update all remaining weights* 当block内所有列全部更新完后，更新H逆剩余所有参数
end for

小模型的量化实验：

Method	RN18 – 69.76 % 4bit	RN18 – 69.76 % 3bit	RN50 – 76.13% 4bit	RN50 – 76.13% 3bit
AdaRound	69.34	68.37	75.84	75.14
AdaQuant	68.12	59.21	74.68	64.98
BRECQ	69.37	68.47	75.88	75.32
OBQ	69.56	68.69	75.72	75.24
GPTQ	69.37	67.88	75.71	74.87

Table 1: Comparison with state-of-the-art post-training methods for vision models.

大模型的量化时间实验：

OPT	13B	30B	66B	175B
Runtime	20.9m	44.9m	1.6h	4.2h
BLOOM	1.7B	3B	7.1B	176B
Runtime	2.9m	5.2m	10.0m	3.8h

Table 2: GPTQ runtime for full quantization of the 4 largest OPT and BLOOM models.

量化速度实验。相比之下ZeroQuant-LKD量化1.3B参数模型花费3h。

在两类大模型上的困惑度对比实验：

OPT	Bits	125M	350M	1.3B	2.7B	6.7B	13B	30B	66B	175B
full	16	27.65	22.00	14.63	12.47	10.86	10.13	9.56	9.34	8.34
RTN	4	37.28	25.94	48.17	16.92	12.10	11.32	10.98	110	10.54
GPTQ	4	**31.12**	**24.24**	**15.47**	**12.87**	**11.39**	**10.31**	**9.63**	**9.55**	**8.37**
RTN	3	1.3e3	64.57	1.3e4	1.6e4	5.8e3	3.4e3	1.6e3	6.1e3	7.3e3
GPTQ	3	**53.85**	**33.79**	**20.97**	**16.88**	**14.86**	**11.61**	**10.27**	**14.16**	**8.68**

Table 3: OPT perplexity results on WikiText2.

BLOOM	Bits	560M	1.1B	1.7B	3B	7.1B	176B
full	16	22.42	17.69	15.39	13.48	11.37	8.11
RTN	4	25.90	22.00	16.97	14.76	12.10	8.37
GPTQ	4	**24.03**	**19.05**	**16.48**	**14.20**	**11.73**	**8.21**
RTN	3	57.08	50.19	63.59	39.36	17.38	571
GPTQ	3	**32.31**	**25.08**	**21.11**	**17.40**	**13.47**	**8.64**

Table 4: BLOOM perplexity results for WikiText2.

超大模型实验：

Method	Bits	OPT-175B Wiki2	OPT-175B PTB	OPT-175B C4	OPT-175B LAMB. ↑	BLOOM-176B Wiki2	BLOOM-176B PTB	BLOOM-176B C4	BLOOM-176B LAMB. ↑
Baseline	16	8.34	12.01	10.13	75.59	8.11	14.59	11.71	67.40
RTN	4	10.54	14.22	11.61	71.34	8.37	15.00	12.04	66.70
GPTQ	4	**8.37**	**12.26**	**10.28**	**76.80**	**8.21**	**14.75**	**11.81**	**67.71**
RTN	3	7.3e3	8.0e3	4.6e3	0	571.	107.	598.	0.17
GPTQ	3	**8.68**	**12.68**	**10.67**	**76.19**	**8.64**	**15.57**	**12.27**	**65.10**
GPTQ	3/g1024	8.45	12.48	10.47	77.39	8.35	15.01	11.98	67.47
GPTQ	3/g128	8.45	12.37	10.36	76.42	8.26	14.89	11.85	67.86

Table 5: Results summary for OPT-175B and BLOOM-176B. “g1024” and “g128” denote results with groupings of size 1024 and 128, respectively.

实际加速实验（OPT-175B）：

GPU	FP16	3bit	Speedup	GPU reduction
A6000 – 48GB	589ms	130ms	4.53×	8 → 2
A100 – 80GB	230ms	71ms	3.24×	5 → 1

Zero-Shot实验：

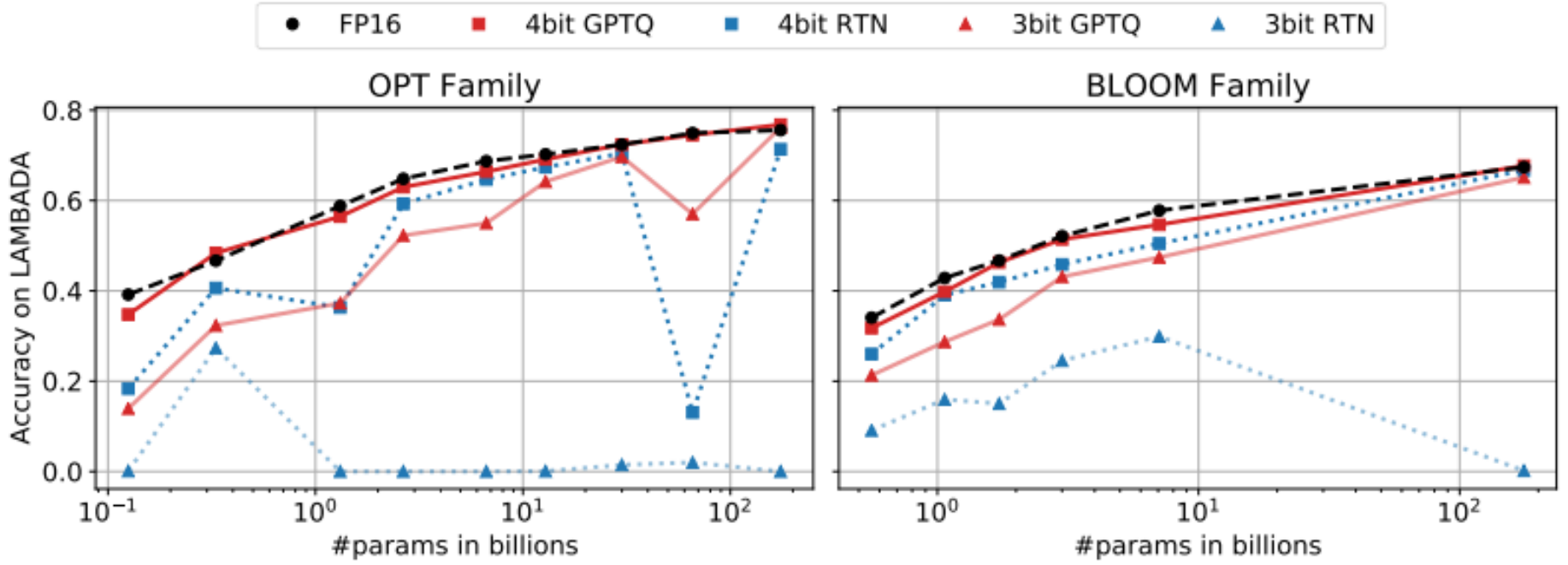


Figure 3: The accuracy of OPT and BLOOM models post-GPTQ, measured on LAMBADA.

THANKS FOR LISTENING