LoRA 系列篇

来自： AiGC面试宝典

宁静致远

2023年09月28日 23:17

一、LoRA篇

1.1 什么是 LoRA?

- 介绍：通过低秩分解来模拟参数的改变量，从而以极小的参数量来实现大模型的间接训练。

1.2 LoRA 的思路是什么?

1. 在原模型旁边增加一个旁路，通过低秩分解（先降维再升维）来模拟参数的更新量；
2. 训练时，原模型固定，只训练降维矩阵A和升维矩阵B；
3. 推理时，可将BA加到原参数上，不引入额外的推理延迟；
4. 初始化，A采用高斯分布初始化，B初始化为全0，保证训练开始时旁路为0矩阵；
5. 可插拔式的切换任务，当前任务W0+B1A1，将lora部分减掉，换成B2A2，即可实现任务切换；

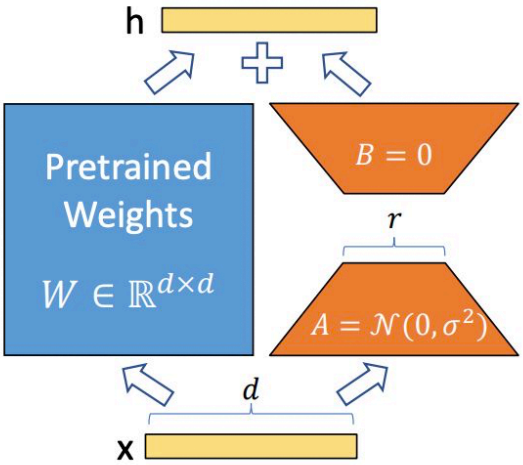

Figure 1: Our reparametrization. We only train A and B.

1.3 LoRA 的特点是什么?

- 将BA加到W上可以消除推理延迟；
- 可以通过可插拔的形式切换到不同的任务；
- 设计的比较好，简单且效果好；

1.4 简单描述一下 LoRA?

LoRA的实现思想很简单，就是冻结一个预训练模型的矩阵参数，并选择用A和B矩阵来替代，在下游任务时只更新A和B。

二、QLoRA篇

2.1 QLoRA 的思路是怎么样的?

- 使用一种新颖的高精度技术将预训练模型量化为 4 bit；
- 然后添加一小组可学习的低秩适配器权重，这些权重通过量化权重的反向传播梯度进行微调。

2.2 QLoRA 的特点是什么?

使用 QLoRA 微调模型，可以显著降低对于显存的要求。同时，模型训练的速度会慢于LoRA。

三、AdaLoRA篇

3.1 AdaLoRA 的思路是怎么样的?

对LoRA的一种改进，它**根据重要性评分动态分配参数预算给权重矩阵，将关键的增量矩阵分配高秩以捕捉更精细和任务特定的信息，而将较不重要的矩阵的秩降低，以防止过拟合并节省计算预算。**

四、LoRA权重是否可以合入原模型?

可以，将训练好的低秩矩阵（B*A）+原模型权重合并（相加），计算出新的权重。

五、ChatGLM-6B LoRA后的权重多大?

rank 8 target_module query_key_value条件下，大约15M。

六、LoRA 微调优点是什么?

1. 一个中心模型服务多个下游任务，节省参数存储量
2. 推理阶段不引入额外计算量
3. 与其它参数高效微调方法正交，可有效组合
4. 训练任务比较稳定，效果比较好
5. LoRA 几乎不添加任何推理延迟，因为适配器权重可以与基本模型合并

七、LoRA微调方法为啥能加速训练?

- **只更新了部分参数**：比如LoRA原论文就选择只更新Self Attention的参数，实际使用时我们还可以选择只更新部分层的参数；
- **减少了通信时间**：由于更新的参数量变少了，所以（尤其是多卡训练时）要传输的数据量也变少了，从而减少了传输时间；
- **采用了各种低精度加速技术**，如FP16、FP8或者INT8量化等。

这三部分原因确实能加快训练速度，然而它们并不是LoRA所独有的，事实上几乎都有参数高效方法都具有这些特点。**LoRA的优点是它的低秩分解很直观，在不少场景下跟全量微调的效果一致，以及在预测阶段不增加推理成本。**

八、如何在已有LoRA模型上继续训练?

理解此问题的情形是：已有的lora模型只训练了一部分数据，要训练另一部分数据的话，是在这个lora上继续训练呢，还是跟base 模型合并后再套一层lora，或者从头开始训练一个lora?
我认为把之前的LoRA跟base model 合并后，继续训练就可以，为了保留之前的知识和能力，训练新的LoRA时，加入一些之前的训练数据是需要的。另外，每次都重头来成本高。

九、LoRA 缺点是什么?

缺点很明显，**参与训练的模型参数量不多，也就百万到千万级别的参数量，所以效果比全量微调差很多。可能在扩散模型上感知没那么强，但在LLM上，个人感觉表现还是差距挺大的。**

十、LoRA这种微调方法和全参数比起来有什么劣势吗?

如果有足够计算资源以及有10k以上数据，我还是建议全参数微调，lora的一个初衷就是为了解决不够计算资源的情况下微调，只引入了少量参数，就可以在消费级gpu上训练，但lora的问题在于它不能节省训练时间，相比于

全量微调，他要训练更久，同时因为可训练参数量很小，在同样大量数据训练下，比不过全量微调。

Model	Training data	others	rewrite	classif-ication	generation	summari-zation	extract	open qa	brain-storming	closed qa	macro ave
LLaMA-7B+ LoRA	0.6M	0.358	0.719	0.695	0.816	0.65	0.448	0.315	0.793	0.51	0.589
LLaMA-7B+ LoRA	2M	0.364	0.795	0.676	0.854	0.617	0.472	0.369	0.808	0.531	0.61
LLaMA-7B+ LoRA	4M	0.341	0.821	0.677	0.847	0.645	0.467	0.374	0.806	0.639	0.624
LLaMA-13B+ LoRA	2M	0.422	0.810	0.696	0.837	0.700	0.537	0.435	0.823	0.577	0.648
LLaMA-7B+ FT	0.6M	0.438	0.869	0.698	0.917	0.701	0.592	0.477	0.870	0.606	0.686
LLaMA-7B+ FT	2M	0.399	0.871	0.775	0.920	0.734	0.603	0.555	0.900	0.633	0.710
LLaMA-7B + FT(2M) + LoRA	math0.25M	0.560	0.863	0.758	0.915	0.754	0.651	0.518	0.886	0.656	0.729
LLaMA-7B + FT(2M) + FT	math0.25M	0.586	0.887	0.763	0.955	0.749	0.658	0.523	0.872	0.652	0.738

十一、LORA应该作用于Transformer的哪个参数矩阵?

	# of Trainable Parameters = 18M						
Weight Type	W_q	W_k	W_v	W_o	W_q, W_k	W_q, W_v	W_q, W_k, W_v, W_o
Rank r	8	8	8	8	4	4	2
WikiSQL (±0.5%)	70.4	70.0	73.0	73.2	71.4	**73.7**	**73.7**
MultiNLI (±0.1%)	91.0	90.8	91.0	91.3	91.3	91.3	**91.7**

从上图我们可以看到:

- 将所有微调参数都放到attention的某一个参数矩阵的效果并不好，将可微调参数平均分配到 Wq 和 Wk 的效果最好
- 即使是秩仅取4也能在 ΔW 中获得足够的信息

因此在实际操作中，应当将可微调参数分配到多种类型权重矩阵中，而不应该用更大的秩单独微调某种类型的权重矩阵。

十二、LoRA 微调参数量怎么确定?

LoRA 模型中可训练参数的结果数量取决于低秩更新矩阵的大小，其主要由秩 r 和原始权重矩阵的形状确定。实际使用过程中，通过选择不同的 lora_target 决定训练的参数量。

以 LLama 为例:

```
--lora_target q_proj,k_proj,v_proj,o_proj,gate_proj,up_proj,down_proj
```

十三、Rank 如何选取?

Rank的取值作者对比了1-64，**效果上Rank在4-8之间最好，再高并没有效果提升**。不过论文的实验是面向下游单一监督任务的，因此在指令微调上根据指令分布的广度，Rank选择还是需要在8以上的取值进行测试。

十四、alpha参数 如何选取?

alpha其实是个缩放参数，本质和learning rate相同，所以为了简化我默认让alpha=rank，只调整lr，这样可以简化超参。

十五、LoRA 高效微调 如何避免过拟合?

减小r或增加数据集大小可以帮助减少过拟合。还可以尝试增加优化器的权重衰减率或LoRA层的dropout值。

十六、微调大模型时, 优化器如何?

除了Adam和AdamW，其他优化器如Sophia也值得研究，它使用梯度曲率而非方差进行归一化，可能提高训练效率和模型性能。

十七、哪些因素会影响内存使用?

内存使用受到模型大小、批量大小、LoRA参数数量以及数据集特性的影响。例如，使用较短的训练序列可以节省内存。

十八、LoRA权重是否可以合并?

可以将多套LoRA权重合并。训练中保持LoRA权重独立，并在前向传播时添加，训练后可以合并权重以简化操作。

十九、是否可以逐层调整LoRA的最优rank?

理论上，可以为不同层选择不同的LoRA rank，类似于为不同层设定不同学习率，但由于增加了调优复杂性，实际中很少执行。

二十、Lora的矩阵怎么初始化？为什么要初始化为全0?

矩阵B被初始化为0，而矩阵A正常高斯初始化
如果B，A全都初始化为0，那么缺点与深度网络全0初始化一样，很容易导致梯度消失(因为此时初始所有神经元的功能都是等价的)。
如果B，A全部高斯初始化，那么在网络训练刚开始就会有概率为得到一个过大的偏移值Δ W 从而引入太多噪声，导致难以收敛。
因此，一部分初始为0，一部分正常初始化是为了在训练开始时维持网络的原有输出(初始偏移为0)，但同时也保证在真正开始学习后能够更好的收敛。