LoRA（Low-Rank Adaptation）详解

目录

1. 前言
2. 1. 背景知识
    2. 1.1 什么是秩?
    3. 1.2 过参数化
3. 2. LoRA
    3. 2.1 计算原理
    4. 2.2 Transformer中的LoRA
    5. 2.3 秩的大小
4. 3. 总结

前言

随着最近大规模语言模型（Large Language Model，LLM）的出现，数十亿乃至千亿的参数量级成为了LLM的标配。如此参数量级的模型意味着传统的模型微调或者线性探测无法同时在训练效率和效果上同时满足开发者的要求。在之前我们介绍过PEFT（Parameter-Efficient Fine-Tuning）的重要的**提示学习**（Prompt Learning）和**适配器学习**（Adapter Learning）。提示学习的问题是模型的效果对提示的依赖非常严重，无论是离散提示还是连续提示，提示怎么有效的构造永远是提示学习的一个痛点。另外对于适配器学习来说，它们一般会向网络层中插入一些可学习的模块，同时这也带来了推理时间的增加。我们这里介绍一个近期训练LLM普遍使用的PEFT算法：**LoRA**（Low Rank Adaptation）[1]，顾名思义，LoRA的核心思想是基于低秩的适配器进行优化。

1. 背景知识

1.1 什么是秩?

那么什么是秩呢？矩阵的秩（rank）分为行秩和列秩，行秩指的是矩阵的线性无关的行的个数，列秩同理。因为一个矩阵的行秩和列秩总是相等的，因此它们统一被叫做矩阵的**秩**。在机器学习中，我们通常使用一个矩阵来表示一个全连接层，但是这个全连接层往往是过参数化的，这意味着我们可以通过计算这个矩阵的

秩来确定哪些特征是重要和相关的。例如在主成分分析（PCA）和奇异值分解（SVD）中，我们通过一个较低维度的表示来近似表示一个高维矩阵或数据集（图1）。换句话说，我们试图找到原始特征空间（或矩阵）中少数维度的（线性）组合，能够捕捉数据集中大部分的信息。

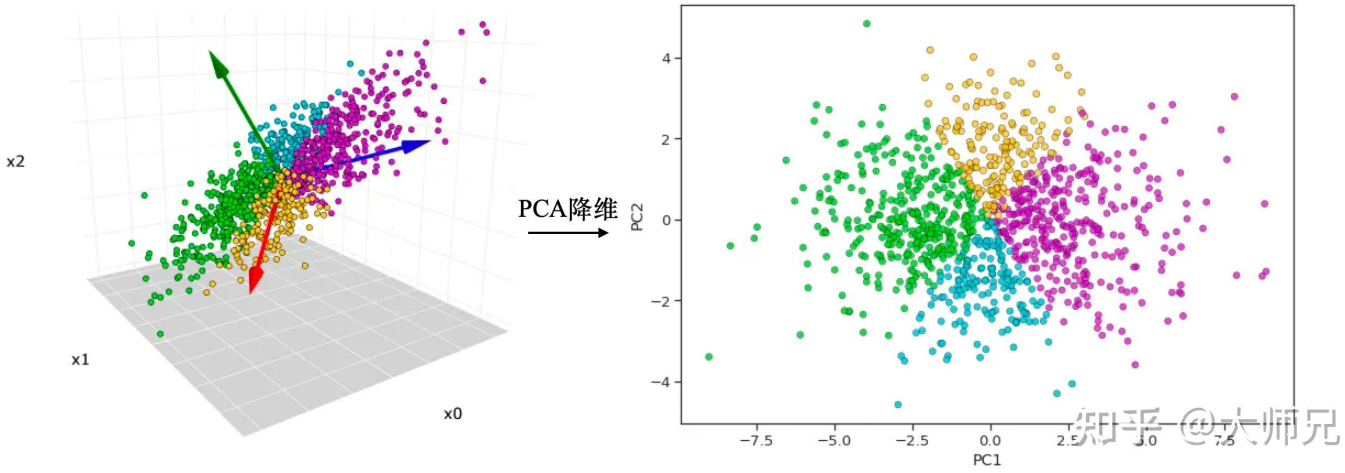

图1：PCA将三维特征压缩为二维图

1.2 过参数化

现在深度学习的参数动不动就有几百万，LLM的参数更是数十亿起步。许多工作[2]已经表明，深度学习的矩阵往往是过参数化的（over-parametrized）。特征的内在维度（intrinsic dimension）指的是在深度学习中的真实或潜在的低维结构或信息的维度。它表示特征中存在的有效信息的维度，与特征的实际维度可能不同。事实上许多问题的内在维度比人们认为的要小的多，而对于某个数据集，内在维度在不同参数量级的模型上差距

并不大。这个内在维度指的是我们解决这个问题实际上需要的参数空间的维度，我们对模型的微调通常调整的也是这些低秩的内在维度。这个结论说明了两个现象：

1. 一旦我们找到了足够解决问题的参数空间，再增加这个参数空间的大小并不会显著提升模型的性能。
2. 一个过参数的模型的参数空间是有压缩的空间的，这也就是LoRA的提出动机。

2. LoRA

2.1 计算原理

和其它串行的适配器算法不同，LoRA的做法是在LLM的某些矩阵（$\boldsymbol{W} \in \mathbb{R}^{d \times k}$）旁插入一个和它并行的新的权值矩阵$\boldsymbol{\Delta W} \in \mathbb{R}^{d \times k}$，但是因为模型的低秩性的存在，我们可以将$\boldsymbol{\Delta W}$拆分成降维矩阵$\boldsymbol{A} \in \mathbb{R}^{r \times k}$和升维矩阵$\boldsymbol{B} \in \mathbb{R}^{d \times r}$（图2），其中 $r \ll \min(d, k)$，从而实现了以极小的参数数量训练LLM。在训练时，我们将LLM的参数固定，只训练矩阵 $\boldsymbol{A}$和 $\boldsymbol{B}$。根据式(1)，在模型训练完成之后，我们可以直接将 $\boldsymbol{A}$和 $\boldsymbol{B}$加到原参数上，从而在推理时不会产生额外的推理时延。

$$\boldsymbol{h} = \boldsymbol{W}_0\boldsymbol{x} + \boldsymbol{\Delta W}\boldsymbol{x} = (\boldsymbol{W}_0 + \boldsymbol{\Delta W})\boldsymbol{x} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{A}\boldsymbol{x}$$

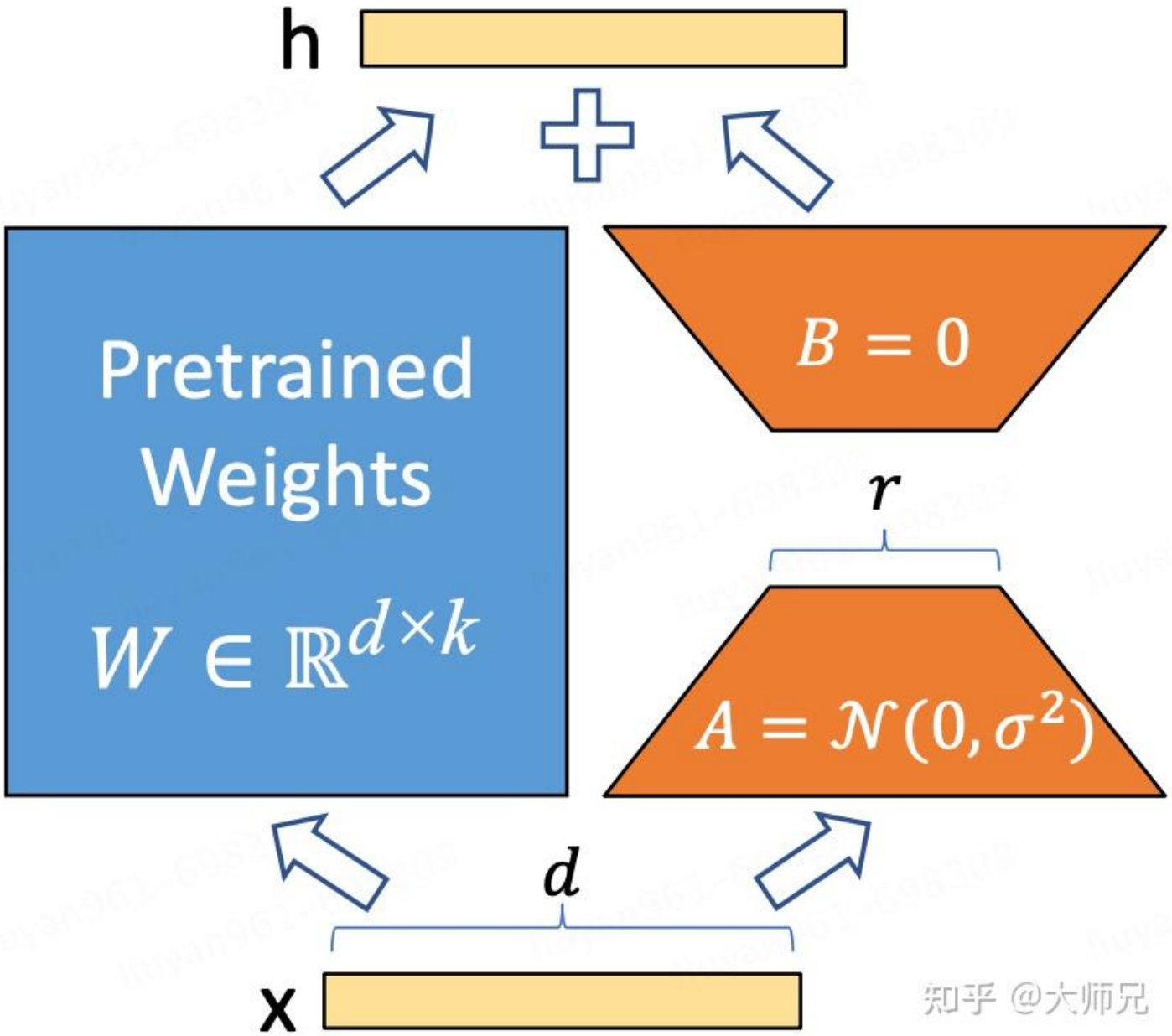

图2：在训练时，LoRA在预训练权值旁插入了一组和它并行的低秩矩阵

在初始化时， $\boldsymbol{A}$使用高斯初始化， $\boldsymbol{B}$使用的零矩阵$\mathbf{0}$进行的初始化。因为 r通常是一个非常小的值（实验证明1，2，4，8的效果就非常好），所以LoRA在训练时引入的参数量是非常小的，因此它的训练也是非常高效的，也不会带来显著的显存增加。

LoRA要求 $\boldsymbol{A}$或者 $\boldsymbol{B}$其中之一必须使用零矩阵进行初始化，这样当数据第一次通过网络时，它和预训练的结果是一致的，这样便保证了模型在初始阶段便有一个不错的效果。苏剑林老师指出，这种一个全零，一个非全零的方式带来了不对称的问题[3]，其实我们也可以使用两个非全零矩阵进行初始化，但是需要事先将预训练权重减去初始化的值，即：$\boldsymbol{W}=\boldsymbol{W}_0-\boldsymbol{B}_0\boldsymbol{A}_0+\boldsymbol{B}\boldsymbol{A}$。

LoRA实现起来非常简单，注意在下面代码的第17行有一个参数 α，它是一个缩放参数，通常是一个常数。通过设置 α有助于在变化 r时减少重新调整超参数的需求。

```
input_dim = 768 # 例如，预训练模型的隐藏大小
output_dim = 768 # 例如，层的输出大小
rank = 8 # 低秩适应的等级'r'
W = ... # 来自预训练网络的权重，形状为 input_dim x output_dim
W_A = nn.Parameter(torch.empty(input_dim, rank)) # LoRA权重A
W_B = nn.Parameter(torch.empty(rank, output_dim)) # LoRA权重B
# 初始化LoRA权重
nn.init.kaiming_uniform_(W_A, a=math.sqrt(5))
nn.init.zeros_(W_B)

def regular_forward_matmul(x, W):
```

```
    h = x @ W
    return h

def lora_forward_matmul(x, W, W_A, W_B):
    h = x @ W # 常规矩阵乘法
    h += x @ (W_A @ W_B) * alpha # 使用缩放的LoRA权重
    return h
```

2.2 Transformer中的LoRA

理论上LoRA的思想可以应用到任何权值矩阵上，例如在Transformer的自注意机制中我们就有四个权值矩阵 $\boldsymbol{W}_q$， $\boldsymbol{W}_k$，$\boldsymbol{W}_v$， $\boldsymbol{W}_o$，另外在Transformer的全连接中也有两个权值矩阵。关于LoRA在Transformer的作用位置，作者在自注意力层做了一组对照实验，实验结果如表1。从表1的实验结果中我们可以看出，如果只将LoRA作用到某个单一矩阵上，$\boldsymbol{W}_q$和 $\boldsymbol{W}_k$的效果并不理想。而如果考虑两个矩阵，$\boldsymbol{W}_q$和 $\boldsymbol{W}_v$的组合是一个不错的选择。而最好的方式是在所有的权值矩阵都加上LoRA，因为这样有利于模型捕捉到所有矩阵的关键信息。

	# 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**

表1：LoRA在不同的自注意力权值矩阵上的对照实验

2.3 秩的大小

关于秩的大小，作者也做了一组对照实验。从表2的实验结果可以看出，秩的值并不是越大越好，一般在 $r = 8$的时候便达到了最优解。而且1或者2这种很小的秩的表现也不差。证明了之前的权值矩阵可能拥有很小的内在秩的假设。

	Weight Type	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
WikiSQL(±0.5%)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI (±0.1%)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

表2：LoRA在不同秩大小上的对照实验

通过对比 $r = 8$和 $r = 64$计算得到的矩阵经过SVD得到的两个右奇异矩阵的格拉斯曼距离相似性，作者得出 $r = 8$的特征包含在了 $r = 64$的其中的8个特征中，设置更大的奇异值反而引入了额外的噪声。

3. 总结

LoRA是PEFT非常重要的一个算法，也是笔者在训练LLM模型时实践出的一个效果非常好的算法。LoRA并不复杂但是设计的非常巧妙：首先LoRA并不会带来任何的推理时间的增加。其次LoRA并不会更改原始模型，而是只训练一个新增的额外参数，而且这个参数仅用来适配当前任务。但是这也意味着LoRA在训练多任务时需要多个不同的 ΔW，多任务的学习对于LoRA来说比较困难，除非把它们当成同一个任务。

参考

1. ^Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." *arXiv preprint arXiv:2106.09685* (2021).
2. ^Li, Chunyuan, et al. "Measuring the intrinsic dimension of objective landscapes." *arXiv preprint arXiv:1804.08838* (2018).
3. ^苏剑林 "梯度视角下的LoRA：简介、分析、猜测及推广" https://kexue.fm/archives/9590 (2023 Apr).