LLM 微调

1.高效参数微调

数高效微调是指微调少量或额外的模型参数，固定大部分预训练模型（LLM）参数，从而大大降低了计算和存储成本，同时，也能实现与全量参数微调相当的性能。参数高效微调方法甚至在某些情况下比全量微调效果更好，可以更好地泛化到域外场景。

高效微调技术可以粗略分为以下三大类：

1. 增加额外参数（A）：Prefix Tuning, Prompt Tuning, Adapter Tuning 及其变体
    a. 类适配器（Adapter-like）方法
    b. 软提示（Soft prompts）：
2. 选取一部分参数更新（S）：BitFit
3. 引入重参数化（R）：LoRA、AdaLoRA、QLoRA
4. 混合高效微调：MAM Adapter、UniPELT

2.增加额外参数

2.1 Prefix Tuning

Prefix Tuning 提出**固定预训练 LM，为 LM 添加可训练，任务特定的前缀**，这样就可以为不同任务保存不同的前缀，微调成本也小；**在每一个 Transformer 层都带上一些 virtual token 作为前缀，以适应不同的任务**。

- **针对自回归架构模型**：**在句子前面添加前缀**，得到 `z = [PREFIX; x; y]`，合适的上文能够在固定 LM 的情况下去引导生成下文（比如：GPT3 的上下文学习）。
- **针对编码器-解码器架构模型**：**Encoder 和 Decoder 都增加了前缀**，得到 `z = [PREFIX; x; PREFIX0; y]`。Encoder 端增加前缀是为了引导输入部分的编码，Decoder 端增加前缀是为了引导后续 token 的生成。

特点：

- 前缀 Token 会占用序列长度，有一定的额外计算开销。
- Prefix Tuning 的线性插值是比较复杂的。

2.2 Prompt Tuning

通过反向传播更新参数来学习 prompts，而不是人工设计 prompts；同时冻结模型原始权重，只训练 prompts 参数，训练完以后，用同一个模型可以做多任务推理。

Prompt Tuning 方法可以看作是 Prefix Tuning 的简化版本，它给**每个任务定义了自己的 Prompt，然后拼接到数据上作为输入，但只在输入层加入 prompt tokens**，并且不需要加入 MLP 进行调整来解决难训练的问题。

特点：

- 相对于 Prefix Tuning，参与训练的参数量和改变的参数量更小，更节省显存。
- 对一些简单的 NLU 任务还不错，但对硬序列标记任务（即序列标注）表现欠佳。

2.3 P-Tuning

将 Prompt 转换为可以学习的 Embedding 层，并用 MLP+LSTM 的方式来对 Prompt Embedding 进行一层处理。相比 Prefix Tuning，仅在**输入层加入的可微的 virtual token**；另外，virtual token 的位置也不一定是前缀，插入的位置是可选的。

特点：引入一个 prompt encoder（由一个双向的 LSTM+ 两层 MLP 组成）来建模 virtual token 的相互依赖会收敛更快，效果更好。

2.4 P-Tuning v2

该方法**在每一个 Transformer 层都加入了 prompt token 作为输入，引入多任务学习，针对不同任务采用不同的提示长度**。并且回归传统的分类标签范式，而不是映射器。

特点：

- 解决了 Prompt Tuning 无法在小模型上有效提升的问题。
- 移除了对模型效果改进较小的重参数化的编码器（如：Prefix Tuning 中的 MLP、P-Tuning 中的 LSTM）。
- 对于一些复杂的硬序列标记任务（即序列标注）取得了不错的效果。

2.5 Adapter Tuning

该方法**设计了 Adapter 结构，并将其嵌入 Transformer 的结构里面，针对每一个 Transformer 层，增加了两个 Adapter 结构，在训练时，固定住原来预训练模型的参数不变，只对新增的 Adapter 结构和 Layer Norm 层进行微调**。

每个 Adapter 模块主要由**两个前馈（Feedforward）子层组成**，第一个前馈子层（down-project）将 Transformer 块的输出作为输入，将原始输入维度 `d`（高维特征）投影到 `m`（低维特征），通过控制 m 的大小来限制 Adapter 模块的参数量，通常情况下，`m<<d`。

特点：通过在 Transformer 层中嵌入 Adapter 结构，在推理时会额外增加推理时长。

2.6 AdapterFusion

一种融合多任务信息的 Adapter 的变体，**在 Adapter 的基础上进行优化，通过将学习过程分为两阶段来提升下游任务表现**。

- **知识提取阶段**：在不同任务下引入各自的 Adapter 模块，用于学习特定任务的信息。
- **知识组合阶段**：将预训练模型参数与特定于任务的 Adapter 参数固定，**引入新参数（AdapterFusion）来学习组合多个 Adapter 中的知识，以提高模型在目标任务中的表现**。

2.7 AdapterDrop

该方法在不影响任务性能的情况下，**对 Adapter 动态高效的移除，尽可能的减少模型的参数量**，提高模型在反向传播（训练）和正向传播（推理）时的效率。

特点：通过从较低的 Transformer 层删除可变数量的 Adaper 来提升推理速度。 当对多个任务执行推理时，动态地减少了运行时的计算开销，并在很大程度上保持了任务性能。

3.选取一部分参数

3.1 BitFit

对微调机制的一种积极探索，也很简单，**通过仅调整 bias 效果就能有不错的效果**，但没有具体阐述原理，就是通过猜测加实验得到的结果。同时，作者提出一个观点：

微调的过程不是让模型适应另外的数据分布，而是让模型更好的应用出本身的表征能力。

特点：

- 训练参数量极小（约 0.1%）。
- 在大部分任务上效果会差于 LoRA、Adapter 等方法。

4.引入重参数化

4.1 LoRA

该方法**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**。

特点：

- 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。

4.2 AdaLoRA

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。

- **调整增量矩分配**。AdaLoRA 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。
- **以奇异值分解的形式对增量更新进行参数化，并根据重要性指标裁剪掉不重要的奇异值，同时保留奇异向量**。由于对一个大矩阵进行精确 SVD 分解的计算消耗非常大，这种方法通过减少它们的参数预算来加速计算，同时，保留未来恢复的可能性并稳定训练。
- **在训练损失中添加了额外的惩罚项**，以规范奇异矩阵 P 和 Q 的正交性，从而避免 SVD 的大量计算并稳定训练。

4.3 QLoRA

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。

特点：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。

5.混合高效微调

5.1 MAM Adapter

一种在 Adapter、Prefix Tuning 和 LoRA 之间建立联系的统一方法。最终的模型 MAM Adapter 是用于 FFN 的并行 Adapter 和 软提示的组合。

特点：整体上来说，最终的模型 MAM Adapter 效果会优于单个高效微调方法。

5.2 UniPELT

一种将不同的 PELT 方法 LoRA、Prefix Tuning 和 Adapter 作为子模块，并通过门控机制学习激活最适合当前数据或任务的方法。

特点：

- 相对于 LoRA，BitFit，Prefix-tuning，训练的参数量更大；同时，推理更耗时；并且，输入会占用额外的序列长度。
- 多种 PELT 方法的混合涉及 PLM 的不同部分对模型有效性和鲁棒性都有好处。

6.多种不同的高效微调方法对比

总的来说，像 P-Tuning v2、LoRA 等都是综合评估很不错的高效微调技术。如果显存资源有限可以考虑 QLoRA；如果只是解决一些简单任务场景，可以考虑 P-Tuning、Prompt Tuning 也行。