1.1 微调算法原理

有监督微调 (Supervised Finetuning, SFT) 又称指令微调 (Instruction Tuning)，是指通过使用有标注的特定任务数据进行进一步的微调预训练模型。

1.1.1 提示学习和语境学习

提示学习 (Prompt-based Learning) 的过程为，首先选用一个由文本组成的模板，模板有两个槽位：[X] 槽位用于填充输入，[Z] 槽位用于生成答案。然后用输入 x 填充模板中的 [X] 槽位得到 x'，然后将 x' 输入语言模型让其预测 [Z] 槽位。举个例子，在情感分类任务中，选用模板"[X] Overall, it was a [Z] movie."。输入 x="I love this movie."。则 x'="I love this movie. Overall, it was a [Z] movie."。

语境学习 (Incontext Learning, ICL) 其概念最早随着 GPT-3 的诞生而提出。是指模型可以从上下文中的几个例子中学习：向模型输入特定任务的一些具体例子以及要测试的样例，模型可以根据给定的示例续写出测试样例的答案。例子见图1.1。ICL 最吸引人的优点在于可

图 1.1: 语境学习例子

以不做任何参数更新就用于许多任务 (只要提供任务的 few shot)，但是 ICL 有几个缺点：

- ICL 会产生大量的计算、内存和存储成本，因为它在每次进行预测时处理所有样例。
- ICL 的效果也不如 fine-tuning。
- 样例的措辞和顺序对效果有着巨大且不可测的影响。

1.1.2 参数高效微调

对大规模 PLM 进行完整微调的代价过高，参数高效微调 (Parameter Efficient Fine Tuning,PEFT) 仅微调少量额外的模型参数，优秀的 PEFT 技术实现了与完整微调相当的性能。

Adapter Tuning [1] 是针对 BERT 的 PEFT 微调方式，拉开了 PEFT 研究的序幕。如图1.2所示 Adapter 结构，将 Adapter 嵌入 Transformer 的结构里面，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 结构进行微调。

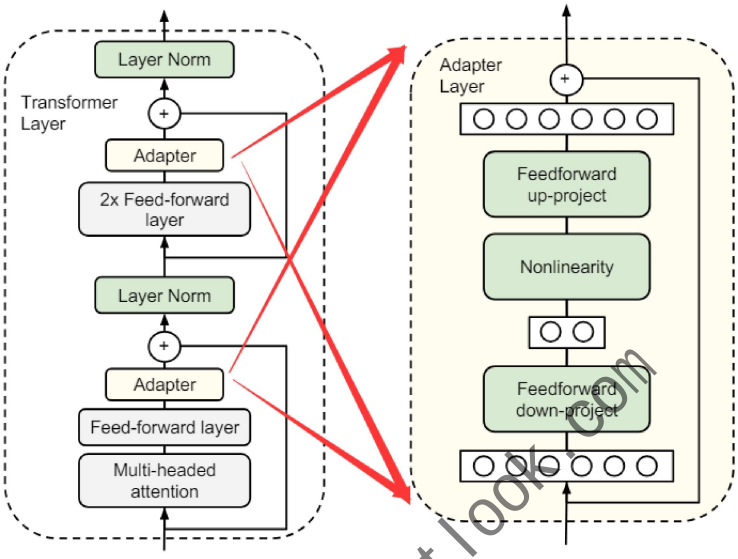

图 1.2: Adapter Tuning 架构示意图

Prefix Tuning [2] 是在输入 token 之前构造一段任务相关的 virtual tokens 作为 Prefix，然后训练的时候只更新 Prefix 部分的参数，而 Transformer 中的其他部分参数固定。Prefix 经过 embedding layer 后，把对应的 embedding 向量放在每个 multi-head attention 模块的 K, V 矩阵的前面几列。Prompt Tuning [3] 可以看作是 Prefix Tuning 的简化版本。固定预训练参数，为每一个任务额外添加一个或多个 embedding，之后拼接 query 正常输入 LLM，并只训练这些 embedding。P-Tuning v1 [4] 与 Prefix-Tuning 类似，区别点在于

- 用 MLP+LSTM 的方式来对 virtual token Embedding 进行一层处理。
- virtual token，但仅限于输入层，没有在每一层都加。
- virtual token 的位置也不一定是前缀，插入的位置是可选的。

P-Tuning v2 [5] 在每一层都加入了 virtual token Embedding 作为输入，而不是仅仅加在输入层。可以看出，Prefix Tuning，Prompt Tuning，P-Tuning v1 和 P-Tuning v2 的出发点都是在输入层加参数，加入任务相关的 token 并训练其对应的 embedding 参数。区别在于这些 embedding 加入的姿势不同，或则这些 embedding 的预处理不同。

LoRA(Low-Rank Adaptation of Large Language Models) [6] 是调整式??中的 W_h^q, W_h^v 两个矩阵，调整前后公式变化如下：

$$Q_h = W_h^q X^T, \quad V_h = W_h^v X^T$$
$$\Downarrow$$
$$Q_h = (W_h^q + \hat{W}_h^q) X^T, \quad V_h = (W_h^v + \hat{W}_h^v) X^T$$

其中 $\hat{W}_h^q, \hat{W}_h^v$ 是微调得到的参数 (W_h^q, W_h^v 在微调中保持不变)，他们都是一个低秩矩阵，所有可以用很少的训练参数得到，如下:

$$\hat{W}_h^q = \hat{A}_h^q \hat{B}_h^q$$

其中 $\hat{W}_h^q \in R^{d\times k}, \hat{A}_h^q \in R^{d\times r}, \hat{B}_h^q \in R^{r\times k}$，其中 r 非常小，所以需要微调中更新的参数很少。可以看出 LoRA 没有改变网络结构，所以不会减慢推理速度。注意，[6] 中实验发现只微调的 W_h^q, W_h^v 取得了最好的效果，不必微调 W_h^k。LoRA 算法在 RoBERTa,GPT-3 等大语言模型上取得了很好的效果。AdaLoRA [7] 在微调过程中根据各权重矩阵对下游任务的重要性动态调整秩的大小，用以进一步减少可训练参数量的同时保持或提高性能。

IA3 [8] 大大减少需要训练参数的数量使得微调更加高效。比如，对于 T0 模型 (T5 的多任务微调版)，IA3 模型仅具有约 0.01% 的可训练参数，而 LoRA 超过 0.1%。IA3 在每层的 encoder 和 decoder 中加入 3 个参数向量，l_k, l_v, l_{ffn}。对于 self-attention 模块，改变如下：

$$softmax(\frac{QK^T}{\sqrt{d}})V \Rightarrow softmax(\frac{Q(l_k \odot K^T)}{\sqrt{d}})(l_v \odot V)$$

对于 FFN 模块，改变则为：

$$\gamma(xW_1)W_2 \Rightarrow l_{ffn} \odot \gamma(xW_1)W_2$$

其中，$l \odot M = l_j M_{i,j}$,γ 为激活函数。可以看到 fine tuning 完成后，只需要把 fine tuning 前的 W^k 和 W^v 分别与 l_k, l_v 相乘得到新的 W^k, W^v 即可，所以不会导致推理时延的增加。IA3 在训练模型 (更新新增的参数) 时往目标函数中追加了两个 loss 来提升评价指标 (rank classification)：unlikelihood loss (加大对错误预测的惩罚) 和 Length Normalization (避免模型倾向输出短序列，因为短序列的概率更大)。作者的实验指出，IA3 参数量大大少于 LoRA 且效果比其更好，甚至比 full-model-fine-tuning 效果更好。

参考文献

[1] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, 2019.

[2] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021.

[3] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning, 2021.

[4] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too, 2023.

[5] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks, 2022.

[6] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[7] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning, 2023.

[8] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning, 2022.