LoRA微调

2.1 什么是LoRA微调

LoRA微调的全程是Low-Rank Adaptation，这种方法通过引入低秩矩阵分解，可以通过仅训练少量参数实现大模型的适配。

2.2 全参数微调的缺陷

全参数微调有几个重大的缺陷

2.2.1 训练成本高

全参数微调，需要更新千亿甚至万亿级别的参数，会消耗大量计算资源，为了调整海量参数时不跑偏，还需要准备相当规模的数据集，费时费力。

2.2.2 训练时间长

全参数微调训练时间会比较长，有些场合需要快速迭代，跟不上业务的节奏

2.2.3 灾难性遗忘

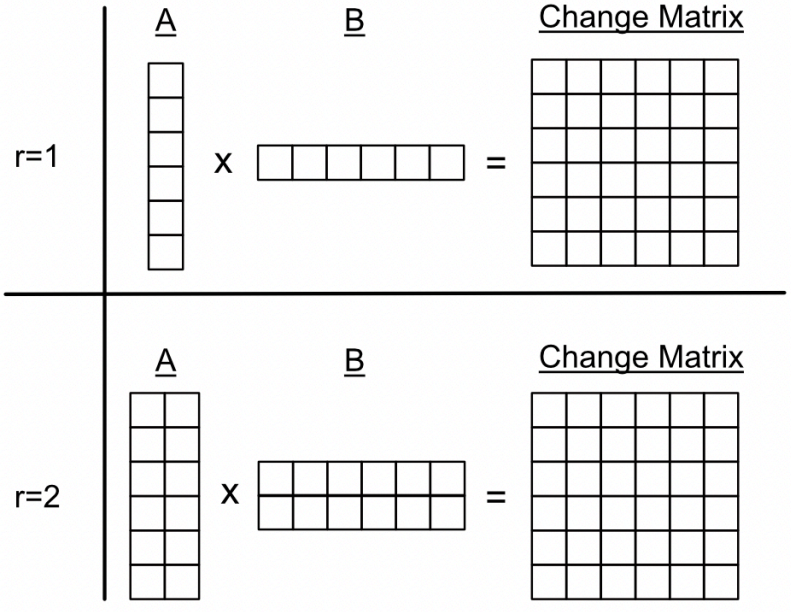
全参数微调需要根据新任务，调整和重构预训练模型的所有参数，有可能导致原有的知识表征被破坏，导致新任务上表现虽好，但是旧任务上的表现急剧下降，这就是大模型的灾难性遗忘。

2.3 LoRA微调怎样避免这些问题?

需要训练一个巨大的参数矩阵W_{M*N}，为了避免灾难性遗忘，训练的时候，我们不能调整原始矩阵，而是要调整一个同样大小的矩阵ΔW_{M*N}，使得推理时可以用$W_{M*N}+\alpha\Delta W_{M*N}$代替原来的$W_{M*N}$。

此时训练的参数量仍未M*N，如何改善?

机器学习里的PCA使用过一个思路：矩阵分解。很多算法都使用类似的方法降低计算量（潜在语义分析LSA，推荐系统里的协同过滤）



不是真正地训练一个巨大的矩阵ΔW_{M*N}，而是训练两个小型矩阵A_{M*r}和B_{r*N}，使得$A_{M*r}*B_{r*N}\approx\Delta W_{M*N}$，只要这两个结果足够近似，在工程上就是可行的。

如果r远小于M 、N，就可以极大地压缩参数量。

假设M=300000，N=500000，那么ΔW_{M*N}的总参数量为M*N=1500亿。

如果取r=16，那么A_{M*r}的参数量为M*r=480万，

同理B_{r*N}的参数量为r*N=800万，

总参数量为1280万，仅为ΔW_{M*N}总参数量的0.000085，即不到万分之一。

2.4 实际做法

- 每个transformer都添加低秩旁路
- 矩阵A高斯初始化
- 矩阵B初始化为全0矩阵

