微调

2025年2月26日　3 分钟阅读

#training #finetuning #DeepSeek-R1 #Unsloth #LoRA

> *本文介绍了微调的常见挑战及其克服方法，并详细介绍了如何使用Unsloth在消费级GPU上对DeepSeek-R1进行微调。*

为什么需要微调?

微调是使DeepSeek-R1等通用语言模型适应特定任务、行业或数据集的关键一步。这就是为什么微调很重要：

领域特定知识：预训练模型在大量通用知识语料库上进行训练。微调允许针对医疗保健、金融或法律分析等特定领域进行专业化。

提高准确性：自定义数据集有助于模型理解利基术语、结构和措辞，从而获得更准确的响应。

任务适应：微调使模型能够以更高的效率执行聊天机器人交互、文档摘要或问答等任务。

偏差减少：基于特定数据集调整模型权重有助于减轻原始训练数据中可能存在的偏差。 通过微调DeepSeek-R1，开发人员可以根据其特定用例对其进行定制，从而提高其有效性和可靠性。

微调中的常见挑战及其克服方法

微调大规模人工智能模型存在若干挑战。以下是一些最常见的挑战及其解决方案：

计算资源限制

挑战：微调大型语言模型（LLMs）需要配备大量显存和内存资源的高端图形处理器（GPU）。

解决方案：使用低秩自适应（LoRA）和4位量化来降低计算负荷。将某些进程卸载到中央处理器（CPU）或者谷歌云端硬盘（Google Colab）、亚马逊云服务（AWS）等基于云的服务也有助于解决问题。

在小数据集上过拟合

挑战：在小数据集上进行训练可能会导致模型记忆响应而不是很好地泛化。

解决方案：使用数据增强技术和正则化方法，如丢弃法（dropout）或提前停止（early stopping），以防止过拟合。

文章信息

字数

阅读时间

发布时间

更新时间

标签

#training #finetuning #Unsloth #LoRA

训练时间长

挑战：微调所需的时间可能长达数天甚至数周，这取决于硬件和数据集的大小。

解决方案：利用梯度检查点（gradient checkpointing）和低秩自适应（LoRA）来加快训练速度并保持效率。

灾难性遗忘

挑战：微调后的模型可能会忘记其预训练阶段所学到的通用知识。

解决方案：使用包含特定领域数据和通用知识数据的混合数据集，以保持模型的整体准确性。

微调模型中的偏差

挑战：微调后的模型可能会继承数据集中存在的偏差。

解决方案：整理多样化且无偏差的数据集，应用去偏技术，并使用公平性指标评估模型。

有效应对这些挑战能够确保微调过程既稳健又高效。

使用Unsloth微调DeepSeek R1

理解DeepSeek - R1

DeepSeek - R1是由DeepSeek开发的一个开源推理模型。它在需要逻辑推理、数学问题解决和实时决策的任务中表现出色。与传统的LLM（大型语言模型）不同，DeepSeek - R1在其推理过程中提供了透明度，这使其适用于解释性至关重要的应用场景。

像DeepSeek - R1这样的大规模AI模型的微调可能非常耗费资源，但使用合适的工具，就有可能在消费级硬件上高效地进行训练。让我们探索如何使用LoRA（低秩适应）和Unsloth来优化DeepSeek - R1的微调，从而实现更快且更具成本效益的训练。

DeepSeek的最新R1模型正在推理性能方面树立新的标杆，可与专有模型相媲美，同时保持开源状态。通过其在Llama 3和Qwen 2.5上训练得到的精炼版本，DeepSeek - R1现在针对使用Unsloth进行微调进行了高度优化，Unsloth是一个用于高效模型适应的框架。

Unsloth是一个用于高效模型适应的框架。它通过将大型模型的权重分解为低秩矩阵，只对这些低秩矩阵进行微调，从而显著减少了内存使用。这使得Unsloth成为微调大型语言模型的理想选择，特别是在消费级硬件上。

在这篇文章中，我们将使用LoRA（低秩适应）和Unsloth在消费级GPU上对DeepSeek - R1进行微调。

环境搭建

硬件要求

微调大型语言模型（LLMs）需要大量的计算资源。以下是推荐的配置：

硬件	最小配置	推荐配置
GPU	None（CPU only）	RTX 3090/4090 24GB显存
内存	48GB	64GB+
存储	250GB	1TB SSD

软件安装

确保你已安装Python 3.8及以上版本，并安装必要的依赖项： pip install unsloth torch transformers datasets accelerate bitsandbytes

```
pip install unsloth torch transformers datasets accelerate bitsandbytes
```

加载预训练模型和分词器

使用Unsloth，我们可以高效地以4位量化加载模型，以减少内存使用。

```
model_name = "unsloth/DeepSeek-R1-Distill-Llama-8B-unsloth-bnb-4bit"
max_seq_length = 2048
model, tokenizer = FastLanguageModel.from_pretrained(
    model_name=model_name,
    max_seq_length=max_seq_length,
    load_in_4bit=True,
)
```

准备你的数据集

微调需要结构化的输入-输出对。让我们假设一个用于指令跟随任务的数据集：

```
{"instruction": "What is the capital of France?", "output": "The capital of France is Paris."}
{"instruction": "Solve: 2 + 2", "output": "The answer is 4."}
```

使用Hugging Face的datasets库加载数据集：

```
from datasets import load_dataset

dataset = load_dataset("json", data_files={"train":
"train_data.jsonl", "test": "test_data.jsonl"})
```

使用聊天风格的提示模板格式化数据集：

```
prompt_template = """Below is an instruction that
describes a task. Write a response that
appropriately completes the request.

### Instruction

{instruction}

### Response

"""
def preprocess_function(examples):
    inputs =
[prompt_template.format(instruction=inst) for inst
in examples["instruction"]]
    model_inputs = tokenizer(inputs,
max_length=max_seq_length, truncation=True)
    return model_inputs
tokenized_dataset =
dataset.map(preprocess_function, batched=True)
```

使用LoRA进行高效微调

LoRA允许通过仅训练模型的特定部分来微调，从而显著减少内存使用。

```
model = FastLanguageModel.get_peft_model(
    model,
    r=16,  # LoRA rank
    target_modules=["q_proj", "v_proj"],  # Fine-
tune key attention layers
    lora_alpha=32,
    lora_dropout=0.05,
    bias="none",
    use_gradient_checkpointing=True,
)
```

训练模型

配置训练参数： 初始化和开始训练：

```
from transformers import Trainer

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset["train"],
    eval_dataset=tokenized_dataset["test"],
    tokenizer=tokenizer,
)
trainer.train()
```

评估和保存模型

训练后，评估和保存微调的模型：

```
# Evaluate the model

eval_results = trainer.evaluate()
print(f"Perplexity: {eval_results['perplexity']}")

# Save the model and tokenizer

model.save_pretrained("./finetuned_deepseek_r1")
tokenizer.save_pretrained("./finetuned_deepseek_r1")
```

部署模型进行推理

微调后，使用模型进行推理：

使用llama.cpp进行本地部署：

```
./llama.cpp/llama-cli \
    --model unsloth/DeepSeek-R1-Distill-Llama-8B-
GGUF/DeepSeek-R1-Distill-Llama-8B-Q4_K_M.gguf \
    --cache-type-k q8_0 \
    --threads 16 \
    --prompt '<|User|>What is 1+1?<|Assistant|>' \
    --n-gpu-layers 20 \
    -no-cnv
```

有用的参考和进一步阅读

要探索更多关于DeepSeek-R1微调和相关技术的内容，请查看以下资源：

- DeepSeek-R1发布公告

DeepSeek-R1技术报告

使用LoRA进行LLMs的高效微调

微调DeepSeek R1（推理模型）

DeepSeek-R1模型在Hugging Face

结论

通过利用LoRA和Unsloth，我们成功地在消费级GPU上微调了DeepSeek-R1，显著减少了内存和计算需求。这使得在没有昂贵硬件的情况下，更快、更可访问的AI模型训练成为可能。

Unsloth 在Kaggle上对DeepSeek R1的微调

参考 https://app.datacamp.com/learn/tutorials/fine-tuning-deepseek-r1-reasoning-model?registration_source=google_onetap 和 https://www.kaggle.com/code/kingabzpro/fine-tuning-deepseek-r1-reasoning-model/notebook

数据集是： https://huggingface.co/datasets/FreedomIntelligence/medical-o1-reasoning-SFT?row=46

Unsloth 在Kaggle上的参考demo

参考demo

Fine-tuning DeepSeek R1 (Reasoning Model) Medical-o1-reasoning-SFT dataset

参考论文和博客

生成式 AI 生命周期

AWS 上的生成式 AI：构建情境感知、多模态推理应用 –O'Reilly 的这本书深入探讨了生成式 AI 生命周期的各个阶段，包括模型选择、微调、适应、评估、部署和运行时优化。

多任务、指令微调

Scaling Instruction-Finetuned Language Model - 以任务、模型大小和思维链数据为重点进行 Scaling 微调。

介绍 FLAN：通过指令微调实现更通用的语言模型 - 这篇博客（和文章）探讨了指令微调，其目的是使语言模型在执行 NLP 任务时更好地进行零点推理。

模型评估指标

HELM - 语言模型的整体评估 - HELM 是一个活的基准，用于更透明地评估语言模型。

通用语言理解评估（GLUE）基准 - 本文介绍了 GLUE，这是一个在多样化自然语言理解（NLU）任务中评估模型的基准，并强调了改进通用 NLU 系统的重要性。

SuperGLUE - 本文介绍了 SuperGLUE 基准，该基准旨在评估各种 NLP Model 在一系列具有挑战性的语言理解任务中的表现。

ROUGE：摘要自动评估软件包 - 本文介绍并评估了 ROUGE 摘要评估软件包中的四种不同测量方法（ROUGE-N、ROUGE-L、ROUGE-W 和 ROUGE-S），它们通过将摘要与理想的人工生成摘要进行比较来评估摘要的质量。

测量大规模多任务语言理解（MMLU） - 本文提出了一种新的测试方法来测量文本模型的多任务准确性，强调了在实现专家级准确性方面进行实质性改进的必要性，并解决了在社会重要主题上的片面表现和低准确性问题。

BigBench-Hard - 该论文介绍了 BIG-bench，这是一个在具有挑战性的任务中评估语言模型的基准，提供了关于 Scale、校准和社会偏见的见解。

参数高效微调（PEFT）

参数高效微调（PEFT） - 本文系统概述了讲座视频中讨论的所有三个类别中的参数效率微调（PEFT）方法。 Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning - 本文系统概述了讲座视频中讨论的所有三个类别中的参数效率微调（PEFT）方法。

论参数效率微调的有效性 - 本文分析了 NLP 中预训练模型的稀疏微调方法。

LoRA

LoRA Large Language Model 的低秩适应 - 本文提出了一种参数高效微调方法，利用低秩分解矩阵减少微调语言模型所需的可训练参数数量。

QLoRA：量化 LLMs 的高效微调 - 本文介绍了一种基于量化的、在单个 GPU 上对 Large Language Model 进行微调的高效方法，在基准测试中取得了令人印象深刻的结果。

使用软提示进行 Prompt 调整

The Power of Scale for Parameter-Efficient Prompt Tuning - 这篇论文探讨了“Prompt Tuning”，这是一种利用学习到的软提示对语言模型进行调节的方法，与完全微调相比，这种方法取得了具有竞争力的性能，并使模型能够在许多任务中重复使用。

分享这篇文章

相关文章推荐

DeepSeek R1 论文解读

本文介绍了深度求索（DeepSeek）公司推出...