deepseek实验一：Unsloth训练自己的推理模型

准备环境

下载unsloth

```
pip install unsloth
```

下载vllm

```
pip install vllm
```

下载pillow

```
pip install --upgrade pillow
```

下载trl

```
pip install trl
```

在开始训练之前，我们需要导入Unsloth的核心组件并进行初始化。首先使用PatchFastRL来追加GRPO和其他RL算法。

```
from unsloth import FastLanguageModel, PatchFastRL
PatchFastRL("GRPO", FastLanguageModel)
```

接下来加载DeepSeek-R1-Distill-Qwen-1.5B并设置训练参数。

导入和初始配置

- max_seq_length决定了模型能处理的最大文本长度
- lora_rank是LoRA（低秩适应）的参数，值越大模型越智能但训练速度越慢

```
from unsloth import is_bfloat16_supported
import torch
max_seq_length = 512 # Can increase for longer reasoning traces
lora_rank = 32 # Larger rank = smarter, but slower
```

加载DeepSeek-R1-Distill-Qwen-1.5B模型并用4位量化来减少内存占用，弃用vLLM来快速推理，控制GPU内存使用率为60%。

```
model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "./DeepSeek-R1-Distill-Qwen-1.5B",
    max_seq_length = max_seq_length,
    load_in_4bit = True, # False for LoRA 16bit
    fast_inference = True, # Enable vLLM fast inference
    max_lora_rank = lora_rank,
    gpu_memory_utilization = 0.6, # Reduce if out of memory
)
```

配置PEFT（参数高效微调）模型

使用LoRA方法对模型进行参数高效微调，指定需要微调的目标模块。启用梯度检查点功能以支持长文本微调，并设置随机种子确保结果可复现。

```
model = FastLanguageModel.get_peft_model(
    model,
    r = lora_rank, # Choose any number > 0 ! Suggested 8, 16, 32, 64, 128
    target_modules = [
        "q_proj", "k_proj", "v_proj", "o_proj",
        "gate_proj", "up_proj", "down_proj",
    ], # Remove QKVO if out of memory
    lora_alpha = lora_rank,
    use_gradient_checkpointing = "unsloth", # Enable long context finetuning
    random_state = 3407,
)
```

```
==((====))==  Unsloth 2025.2.5: Fast Qwen2 patching. Transformers: 4.48.3.
   \\   /|    GPU: NVIDIA GeForce RTX 4090. Max memory: 23.546 GB. Platform: Lin
O^O/ \_/ \    Torch: 2.5.1+cu124. CUDA: 8.9. CUDA Toolkit: 12.4. Triton: 3.1.0
\        /    Bfloat16 = TRUE. FA [Xformers = 0.0.28.post3. FA2 = False]
 "-____-"     Free Apache license: http://github.com/unslothai/unsloth
Unsloth: Fast downloading is enabled - ignore downloading bars which are red col
Unsloth: vLLM loading ./DeepSeek-R1-Distill-Qwen-1.5B with actual GPU utilizatio
```

```
Unsloth: Your GPU has CUDA compute capability 8.9 with VRAM = 23.55 GB.
Unsloth: Using conservativeness = 1.0. Chunked prefill tokens = 512. Num Sequenc
Unsloth: vLLM's KV Cache can use up to 6.47 GB. Also swap space = 6 GB.
INFO 02-12 21:58:44 config.py:542] This model supports multiple tasks: {'classif
INFO 02-12 21:58:44 llm_engine.py:234] Initializing a V0 LLM engine (v0.7.2) wit
INFO 02-12 21:58:45 model_runner.py:1110] Starting to load model ./DeepSeek-R1-D



Loading safetensors checkpoint shards:   0% Completed | 0/1 [00:00<?, ?it/s]



INFO 02-12 21:58:47 model_runner.py:1115] Loading model weights took 3.3158 GB
INFO 02-12 21:58:47 punica_selector.py:18] Using PunicaWrapperGPU.
INFO 02-12 21:58:52 worker.py:267] Memory profiling takes 5.53 seconds
INFO 02-12 21:58:52 worker.py:267] the current vLLM instance can use total_gpu_m
INFO 02-12 21:58:52 worker.py:267] model weights take 3.32GiB; non_torch_memory
INFO 02-12 21:58:53 executor_base.py:110] # CUDA blocks: 12817, # CPU blocks: 14
INFO 02-12 21:58:53 executor_base.py:115] Maximum concurrency for 512 tokens per
INFO 02-12 21:58:56 model_runner.py:1434] Capturing cudagraphs for decoding. Thi


Capturing CUDA graph shapes: 100%|██████████| 27/27 [00:22<00:00,  1.18it/s]

INFO 02-12 21:59:19 model_runner.py:1562] Graph capturing finished in 23 secs, t
INFO 02-12 21:59:19 llm_engine.py:431] init engine (profile, create kv cache, wa




./DeepSeek-R1-Distill-Qwen-1.5B does not have a padding token! Will use pad_toke


Unsloth 2025.2.5 patched 28 layers with 28 QKV layers, 28 O layers and 28 MLP la
```

准备数据集

使用数据准备脚本以及奖励函数。本示例用到了openAI的GSM8K数据集。

```
import re
from datasets import load_dataset, Dataset

# Load and prep dataset
SYSTEM_PROMPT = """
Respond in the following format:
<reasoning>
...
</reasoning>
<answer>
...
</answer>
"""

XML_COT_FORMAT = """\
<reasoning>
{reasoning}
</reasoning>
<answer>
{answer}
</answer>
"""

def extract_xml_answer(text: str) -> str:
    answer = text.split("<answer>")[-1]
    answer = answer.split("</answer>")[0]
    return answer.strip()

def extract_hash_answer(text: str) -> str | None:
    if "####" not in text:
        return None
    return text.split("####")[1].strip()

# uncomment middle messages for 1-shot prompting
def get_gsm8k_questions(split = "train") -> Dataset:
    data = load_dataset('./gsm8k', 'main')[split] # type: ignore
    data = data.map(lambda x: { # type: ignore
        'prompt': [
            {'role': 'system', 'content': SYSTEM_PROMPT},
            {'role': 'user', 'content': x['question']}
        ],
        'answer': extract_hash_answer(x['answer'])
    }) # type: ignore
    return data # type: ignore
```

```
dataset = get_gsm8k_questions()

# Reward functions
def correctness_reward_func(prompts, completions, answer, **kwargs) -> list[floa
    responses = [completion[0]['content'] for completion in completions]
    q = prompts[0][-1]['content']
    extracted_responses = [extract_xml_answer(r) for r in responses]
    print('-'*20, f"Question:\n{q}", f"\nAnswer:\n{answer[0]}", f"\nResponse:\n{
    return [2.0 if r == a else 0.0 for r, a in zip(extracted_responses, answer)]

def int_reward_func(completions, **kwargs) -> list[float]:
    responses = [completion[0]['content'] for completion in completions]
    extracted_responses = [extract_xml_answer(r) for r in responses]
    return [0.5 if r.isdigit() else 0.0 for r in extracted_responses]

def strict_format_reward_func(completions, **kwargs) -> list[float]:
    """Reward function that checks if the completion has a specific format."""
    pattern = r"^<reasoning>\n.*?\n</reasoning>\n<answer>\n.*?\n</answer>\n$"
    responses = [completion[0]["content"] for completion in completions]
    matches = [re.match(pattern, r) for r in responses]
    return [0.5 if match else 0.0 for match in matches]

def soft_format_reward_func(completions, **kwargs) -> list[float]:
    """Reward function that checks if the completion has a specific format."""
    pattern = r"<reasoning>.*?</reasoning>\s*<answer>.*?</answer>"
    responses = [completion[0]["content"] for completion in completions]
    matches = [re.match(pattern, r) for r in responses]
    return [0.5 if match else 0.0 for match in matches]

def count_xml(text) -> float:
    count = 0.0
    if text.count("<reasoning>\n") == 1:
        count += 0.125
    if text.count("\n</reasoning>\n") == 1:
        count += 0.125
    if text.count("\n<answer>\n") == 1:
        count += 0.125
        count -= len(text.split("\n</answer>\n")[-1])*0.001
    if text.count("\n</answer>") == 1:
        count += 0.125
        count -= (len(text.split("\n</answer>")[-1]) - 1)*0.001
    return count

def xmlcount_reward_func(completions, **kwargs) -> list[float]:
```

```
    contents = [completion[0]["content"] for completion in completions]
    return [count_xml(c) for c in contents]
```

```
Generating train split:   0%|          | 0/7473 [00:00<?, ? examples/s]
```

```
Generating test split:   0%|          | 0/1319 [00:00<?, ? examples/s]
```

```
Map:   0%|          | 0/7473 [00:00<?, ? examples/s]
```

训练模型

使用GRPO训练方法
改配置的主要特点是:

- 采用了内存效率较高的设置（8位优化器，混合进度训练）
- 使用较小的学习率和梯度剪裁以确保训练稳定性
- 提供了灵活的设置选项以适应不同的硬件条件
- 包含了完整的训练监控和模型保存机制

```
from trl import GRPOConfig, GRPOTrainer
training_args = GRPOConfig(
    use_vllm = True, # use vLLM for fast inference!
    learning_rate = 5e-6,
    adam_beta1 = 0.9,
    adam_beta2 = 0.99,
    weight_decay = 0.1,
    warmup_ratio = 0.1,
    lr_scheduler_type = "cosine",
    optim = "paged_adamw_8bit",
    logging_steps = 1,
    bf16 = is_bfloat16_supported(),
    fp16 = not is_bfloat16_supported(),
    per_device_train_batch_size = 1,
    gradient_accumulation_steps = 1, # Increase to 4 for smoother training
    num_generations = 6, # Decrease if out of memory
    max_prompt_length = 256,
    max_completion_length = 200,
```

```
    # num_train_epochs = 1, # Set to 1 for a full training run
    max_steps = 250,
    save_steps = 250,
    max_grad_norm = 0.1,
    report_to = "none", # Can use Weights & Biases
    output_dir = "outputs",
)
```

```
torch.distributed process group is initialized, but parallel_mode != ParallelMod
```

GRPO训练器

奖励函数

- xmlcount_reward_func：可能用于评估生成文本中XML标签的正确使用
- soft_format_reward_func：评估输出格式的基本符合程度
- strict_format_reward_func：严格检查输出格式是否完全符合要求
- int_reward_func：可能用于检查数值输出的准确性
- correctness_reward_func：评估整体输出的正确性

通过这些奖励函数的组合，模型能够学习到如何生成既格式正确又内容准确的输出。这种多维度的评估系统能更好的指导模型向着期望的方向学习和改进。

```
trainer = GRPOTrainer(
    model = model,
    processing_class = tokenizer,
    reward_funcs = [
        xmlcount_reward_func,
        soft_format_reward_func,
        strict_format_reward_func,
        int_reward_func,
        correctness_reward_func,
    ],
    args = training_args,
    train_dataset = dataset,
)
trainer.train()
```

```
==((====))==  Unsloth - 2x faster free finetuning | Num GPUs = 1
   \\   /|    Num examples = 7,473 | Num Epochs = 1
O^O/ \_/ \    Batch size per device = 1 | Gradient Accumulation steps = 1
```

```
\        /    Total batch size = 1 | Total steps = 250
 "-____-"     Number of trainable parameters = 36,929,536


-------------------- Question:
Ahmed and Emily are having a contest to see who can get the best grade in the cl
Answer:
100
Response:
First, I need to determine the total points for all assignments. Since there hav

Next, I'll calculate the total points each person has so far.

Emily has 92 points out of 100, and she scored a 90 on the final assignment, giv

Similarly, Ahmed has 91 points on his 9 assignments and is expected to score a m

To beat Emily, Ahmed's total points must be greater than 182. Setting up the ine
Extracted:
First, I need to determine the total points for all assignments. Since there hav

Next, I'll calculate the total points each person has so far.

Emily has 92 points out of 100, and she scored a 90 on the final assignment, giv

Similarly, Ahmed has 91 points on his 9 assignments and is expected to score a m

To beat Emily, Ahmed's total points must be greater than 182. Setting up the ine
```

```
<div>

  <progress value='250' max='250' style='width:300px; height:20px; vertical-alig
  [250/250 08:17, Epoch 0/1]
</div>
<table border="1" class="dataframe">
```

推理

采样参数:

- temperature：值越高生成越随机，越低越确定性
- top_p：控制采样时考虑的概率质量范围
- max_tokens：限制生成文本的最大长度

```
text = tokenizer.apply_chat_template([
    {"role" : "user", "content" : "Calculate pi."},
], tokenize = False, add_generation_prompt = True)

from vllm import SamplingParams
sampling_params = SamplingParams(
    temperature = 0.8,
    top_p = 0.95,
    max_tokens = 1024,
)
output = model.fast_generate(
    [text],
    sampling_params = sampling_params,
    lora_request = None,
)[0].outputs[0].text

output
```

```
Processed prompts: 100%|██████████| 1/1 [00:02<00:00,  2.87s/it, est. speed inpu
```

```
"To calculate pi, I will use the Monte Carlo method by generating random points
```

保存经过训练的LoRA权重数据

```
model.save_lora("grpo_saved_lora")
```

加载LoRA权重并推理

- 加载之前保存的LoRA权重
- 使用这些权重和配置的参数生成回答
- 返回生成的文本结果

```
text = tokenizer.apply_chat_template([
    {"role" : "system", "content" : SYSTEM_PROMPT},
    {"role" : "user", "content" : "Calculate pi."},
], tokenize = False, add_generation_prompt = True)

from vllm import SamplingParams
sampling_params = SamplingParams(
    temperature = 0.8,
    top_p = 0.95,
    max_tokens = 1024,
)
output = model.fast_generate(
    text,
    sampling_params = sampling_params,
    lora_request = model.load_lora("grpo_saved_lora"),
)[0].outputs[0].text

output
```

```
Processed prompts: 100%|██████████| 1/1 [00:03<00:00,  3.22s/it, est. speed inpu
```

```
"Okay, so I need to figure out how to calculate pi. I've heard that pi is a real
```