Relevant Code

Code Repository

Repository: https://github.com/deepseek-ai/DeepSeek-Math

Status: Official Implementation

Language: Python

Last Updated: February 2024

Stars/Popularity: High (official DeepSeek AI repository)

Architecture Overview

The DeepSeekMath repository provides a comprehensive evaluation framework for mathematical language models rather than the full training pipeline. The codebase is structured around model evaluation with support for multiple mathematical reasoning formats including chain-of-thought (CoT), program-of-thought (PoT), and tool-integrated reasoning. The repository focuses on benchmark assessment across various mathematical datasets like GSM8K, MATH, and Chinese mathematical benchmarks.

The architecture is modular, with separate components for inference engines, answer extraction, mathematical evaluation, and few-shot prompting. The evaluation system supports both standard transformers and optimized inference through vLLM, enabling scalable assessment of model performance. The codebase demonstrates practical implementation of the mathematical reasoning evaluation methodologies described in the paper.

Directory Structure

```
DeepSeek-Math/
├── evaluation/
│   ├── infer/              - Inference engines for different reasoning formats
│   ├── data_processing/    - Answer extraction and data preprocessing utilities
│   ├── eval/              - Mathematical evaluation scripts and utilities
│   ├── few_shot_prompts/  - Prompt templates for different benchmarks
│   └── utils.py           - General utilities for model loading and generation
```

```
├── images/                - Paper figures and evaluation results
├── replicate/             - Replication configuration
└── README.md              - Documentation and usage instructions
```

Key Implementation

Mathematical Answer Extraction

This component implements sophisticated mathematical answer parsing from model outputs, handling LaTeX expressions, mathematical notation, and answer formatting:

```
def _fix_fracs(string):
    substrs = string.split("\\frac")
    new_str = substrs[0]
    if len(substrs) > 1:
        substrs = substrs[1:]
        for substr in substrs:
            new_str += "\\frac"
            if len(substr) > 0 and substr[0] == "{":
                new_str += substr
            else:
                try:
                    assert len(substr) >= 2
                except:
                    return string
                a = substr[0]
                b = substr[1]
                if b != "{":
                    if len(substr) > 2:
                        post_substr = substr[2:]
                        new_str += "{" + a + "}{" + b + "}" + post_substr
                    else:
                        new_str += "{" + a + "}{" + b + "}"
                else:
                    if len(substr) > 2:
                        post_substr = substr[2:]
                        new_str += "{" + a + "}" + b + post_substr
                    else:
                        new_str += "{" + a + "}" + b
    string = new_str
    return string
```

Key aspects:
- Handles LaTeX fraction notation conversion
- Ensures proper mathematical formatting
- Robust error handling for malformed expressions
- Supports multiple mathematical notation formats

Chain-of-Thought Inference Engine

This implements the core inference pipeline for chain-of-thought mathematical reasoning:

```
def infer(args, test_data):
    global tokenizer
    if tokenizer is None:
        tokenizer = AutoTokenizer.from_pretrained(args.tokenizer_name_or_path, trust_remote_code=Tr

    if args.prompt_format == 'few_shot':
        assert args.few_shot_prompt is not None
        prompting = eval(args.few_shot_prompt)()

    prompts = []
    for example in test_data:
        prompt = ""
        if args.prompt_format == 'few_shot':
            prompt = prompting.format_prompt(example['messages'][-2]['content'], example['messages'
        else:
            for mess in example['messages']:
                if args.prompt_format == 'sft':
                    if mess['role'] == 'user':
                        prompt += f"{tokenizer.eos_token}User: {mess['content'].strip()}\n\nAssista
                    elif mess['role'] == 'assistant':
                        prompt += mess['content'].rstrip()
                else:
                    raise NotImplementedError()
            prompt = prompt.lstrip()
            if args.prompt_format == 'sft' and prompt.startswith(tokenizer.eos_token):
                prompt = prompt[len(tokenizer.eos_token):].lstrip()
        example['prompt'] = prompt
        prompts.append(prompt.lstrip())
```

Key aspects:
- Supports both few-shot and supervised fine-tuning formats
- Handles message-based conversation formatting
- Integrates with HuggingFace Transformers tokenizer
- Flexible prompt engineering capabilities

Program Extraction for Tool-Integrated Reasoning

This component extracts Python code from model outputs for program-of-thought reasoning:

```
def extract_program(result: str, last_only=True):
    """
    extract the program after "```python", and before "```"
    """
    program = ""
    start = False
    for line in result.split("\n"):
        if line.startswith("```python"):
            if last_only:
                program = "" # only extract the last program
            else:
                program += "\n# ========\n"
            start = True
        elif line.startswith("```"):
            start = False
        elif start:
            program += line + "\n"
    return program
```

Key aspects:
- Extracts executable Python code from markdown blocks
- Supports multiple program extraction modes
- Handles tool-integrated mathematical reasoning
- Enables programmatic problem solving capabilities

Few-Shot Prompt Templates

Example implementation of mathematical few-shot prompting for Chinese benchmarks:

```
few_shot_prompt = """
问题 1.    已知 $\\alpha, \\beta, \\gamma$ 是互不相同的锐角, 则在 $\\sin \\alpha \\cos \\beta, \\sin \\b
从以下选项中选择:    (A)0 (B)1 (C)2 (D)3
问题 1的解析:   1. 如果 $\\alpha, \\beta, \\gamma$ 均小于 $60^\\circ$，那么他们的正弦值都小于 $\\frac{1}{2
2. 如果有一个角大于 $60^\\circ$，假设为 $\\alpha$，那么对应的正弦值大于 $\\frac{1}{2}$。此时，由于三角形内角和
3. 如果有两个角大于 $60^\\circ$，例如 $\\alpha$ 和 $\\beta$，那么由于三角形内角和为 $180^\\circ$，我们可以得到
的余弦值都小于 $\\frac{1}{2}$，因此三个值中不可能有大于 $\\frac{1}{2}$ 的值。
4. 如果三个角都大于 $60^\\circ$，显然不符合题意。
综上所述，当有一个角大于 $60^\\circ$ 时，大于 $\\frac{1}{2}$ 的个数的最大值是 2。
答案是 C
"""


class CoTGaoKaoMathQAPrompt(FewShotPrompting):
    def __init__(self):
        super().__init__()

    def format_prompt(self, task_input, task_output):
        prompt = f"{few_shot_prompt}\n\n\n问题 6.    {task_input}\n问题 6的解析:   {task_output}"
        return prompt.rstrip()

    def stop_words(self):
        return ["\n问题 "]
```

Key aspects:
- Provides structured mathematical reasoning examples
- Supports multilingual mathematical content
- Demonstrates step-by-step problem solving approach
- Includes answer extraction from formatted responses

Relation to Paper

Paper Section	Code Component	Notes
Section 3.1: SFT Data Curation	`evaluation/few_shot_prompts/`	Implements chain-of-thought and program-of-thought prompting templates

Paper Section	Code Component	Notes
Section 4.1: GRPO Algorithm	Not included in repository	Training algorithm not publicly available, only evaluation code
Section 2.1: Data Collection	Not included in repository	Data processing pipeline not released
Equation 1: PPO Objective	Not included in repository	RL training code proprietary
Mathematical Evaluation	`evaluation/eval/eval_utils.py`	Implements answer extraction and mathematical evaluation
Tool-Integrated Reasoning	`evaluation/infer/run_tool_integrated_eval.py`	Supports Python-based mathematical problem solving
Figure 1: Performance Comparison	`images/`	Contains benchmark results and model comparisons

Key Differences from Paper

- **Limited to Evaluation**: The repository only provides evaluation and inference code, not the full training pipeline
- **No GRPO Implementation**: The core Group Relative Policy Optimization algorithm is not included
- **No Data Pipeline**: The mathematical corpus construction methodology is described but not implemented
- **Model Access Only**: Pre-trained models are available via HuggingFace, but training code is proprietary
- **Focus on Benchmarking**: Emphasis is on reproducing paper results through standardized evaluation

The repository serves as a comprehensive evaluation framework that allows researchers to reproduce the paper's benchmark results and build upon the mathematical reasoning evaluation methodologies, while the core training innovations remain proprietary to DeepSeek AI.