IAI Hackathon 2025

Đề thi

Hà Nội, ngày 26 tháng 07 năm 2025

MoE

Mô tả

Mixture of Experts (MoE) là một kiến trúc mô hình trong học máy, kết hợp nhiều "chuyên gia" (experts) lại với nhau, trong đó một cơ chế định tuyến (router) quyết định chuyên gia nào sẽ xử lý từng phần của dữ liệu đầu vào. MoE thường được sử dụng để mở rộng quy mô mô hình một cách hiệu quả mà không làm tăng đáng kể chi phí tính toán.

Các thành phần chính của MoE:
Các chuyên gia (Experts):

- Mỗi chuyên gia là một mạng con (thường là một mạng feed-forward nhỏ) chịu trách nhiệm xử lý một phần cụ thể của dữ liệu.
- Ví dụ: Trong mô hình ngôn ngữ lớn, mỗi chuyên gia có thể chuyên về một lĩnh vực khác nhau (toán học, ngôn ngữ, lập trình, ...).

Bộ định tuyến (Router):

- Một mạng neural (thường là một lớp tuyến tính + softmax) quyết định trọng số phân phối đầu vào cho các chuyên gia.
- Đầu vào x được ánh xạ thành một vector trọng số g(x), xác định mức độ đóng góp của từng chuyên gia.

Kết hợp đầu ra (Weighted Sum):

- Đầu ra của mô hình là tổng có trọng số của các chuyên gia dựa trên kết quả từ bộ định tuyến

```
Structure: Input → Router → Top-k Experts Selected → Expert Processing →
Weighted Output
Input Tokens: [Token1, Token2, Token3]
               ↓       ↓       ↓
Router:       [Expert1: 0.7, Expert3: 0.3]  (Token1)
              [Expert2: 0.9, Expert1: 0.1]  (Token2)
              [Expert3: 0.6, Expert2: 0.4]  (Token3)
               ↓       ↓       ↓
Experts:      Expert1 processes Token1 (weight=0.7) + Token2 (weight=0.1)
              Expert2 processes Token2 (weight=0.9) + Token3 (weight=0.4)
              Expert3 processes Token1 (weight=0.3) + Token3 (weight=0.6)
               ↓       ↓       ↓
Output:           Weighted sum of expert outputs per token.
```

Vấn đề

Trong bài thi này, nhiệm vụ của bạn là thực hiện các phép nhân ma trận với expert và tổng hợp kết quả đầu ra. Dữ kiện về bộ định tuyến đã được cung cấp sẵn. Chương trình của bạn có chức năng tương tự đoạn mã Python sau:

```python
import torch
def torch_moe(hidden_states: torch.Tensor, # [num_token, model_dim]
              w1: torch.Tensor, # [num_expert, inter_dim*2, model_dim]
              w2: torch.Tensor, # [num_expert, model_dim, inter_dim]
              topk_weight: torch.Tensor, # [num_token, topk]
              topk_ids: torch.Tensor, # [num_token, topk]
              ):
    token_num = hidden_states.shape[0]
    topk = topk_weight.shape[1]
    expert, model_dim, inter_dim = w2.shape
    hidden_states = hidden_states.view(
        token_num, 1, model_dim).repeat(1, topk, 1)
    out = torch.zeros(
        (token_num, topk, model_dim),
        dtype=hidden_states.dtype,
        device=hidden_states.device,
    )
    for E_id in range(expert):
        mask = topk_ids == E_id
        if mask.sum():
            sub_tokens = hidden_states[mask]
            act_input = sub_tokens @ (w1[E_id].transpose(0, 1))
            gate, up = act_input.split([inter_dim, inter_dim], dim=-1)
            act_out = F.silu(gate) * up
            out[mask] = act_out @ (w2[E_id].transpose(0, 1))
    return (out * topk_weight.view(token_num, -1, 1)).sum(dim=1)
```

Input:

- 5 số nguyên dương num_token, model_dim, inter_dim, num_expert, topk
- Mảng số thực kích thước num_token*model_dim đại diện cho hidden_states
- Mảng số thực kích thước num_expert*inter_dim*2*model_dim đại diện cho w1
- Mảng số thực kích thước num_expert*model_dim*inter_dim đại diện cho w2
- Mảng số thực kích thước num_token*topk đại diện cho topk_weight
- Mảng số nguyên dương kích thước num_token*topk đại diện cho topk_ids

Output:

- Một dãy số thực gồm num_token*model_dim phần tử

Giới hạn:

- 1 <= num_token <= 4096
- 1 <= mode_dim <= 7168
- 1 <= inter_dim <= 2048
- 1 <= topk <= num_expert <= 512
- w1, w2, topk_weight mang kiểu float32
- 0 <= topk_ids[] < num_expert

Knapsack Scheduler

Mô tả

Một công ty vận tải chỉ sử dụng xe tải có tải trọng K, cần phải vận chuyển N món hàng. Hãy tìm số lượng xe ít nhất mà công ty cần phải sử dụng.

Input:

- 2 số nguyên dương N, K
- Mảng số nguyên dương kích thước N phần tử biểu diễn khối lượng của từng món hàng

Output:

- Số nguyên dương biểu diễn số lượng xe tải cần dùng

Giới hạn:

- Khối lượng các món hàng luôn nhỏ hơn K
- 1 <= K, N <= 10000

Parallel Bin-search

Mô tả

Bạn có một ma trận 2 chiều A kich thước N*N với mỗi dòng và cột đã được xắp xếp theo thứ tự tăng dần và một danh sách các truy vấn được biểu diễn dưới dạng ma trận 2 chiều Q kích thước M*2. Mỗi dòng của Q gồm 2 giá trị: p và x. Trong đó, p là chỉ số hàng (khi dương) và là chỉ số cột (khi âm) (chú ý là các hàng và cột được đánh số từ 1); x là giá trị cận trên. Bạn cần phải tìm ở hàng (hoặc cột) p có bao nhiêu số nhỏ hơn x.

Input:

- 2 số nguyên dương N, M
- Mảng số nguyên A gồm N*N phần tử
- Mảng số nguyên Q gồm M*2 phần tử

Output:

- Mảng số nguyên gồm M phần từ chứa đáp án của từng truy vấn

Giới hạn:

- 1 <= N <= 100000
- 1 <= M <= min(N*N, 1000000)

Find all Possibilities

Mô tả

Có N tuyến đường tạo thành chu trình Halminton độ dài N. Các tọa độ được đánh đánh số từ 0 tới N-1 và luôn có tuyến đường hai chiều nối giữa hai điểm liên tiếp với thời gian di chuyển cho mỗi tuyến đường luôn là 1 đơn vị thời gian. Nhiệm vụ của bạn là tính số lượng đường đi có thể để di chuyển từ tọa độ A đến tọa độ B trong đúng K đơn vị thời gian. Vì kết quả có thể rất lớn, bạn chỉ cần trả về phần dư của kết quả khi chia cho 10^9.

Input:

- 4 số nguyên N, K, A, B

Output:

- Số nguyên dương đại diện cho số cách di chuyển sau khi chia lấy dư cho 10^9

Giới hạn:

- 1 <= N <= 10000
- 0 <= A, B < N
- 1 <= K <= 10^18

RMSNorm Quantization

Mô tả

Bạn có một ma trận 2 chiều A kích thước M*N. Tính RMSNorm của từng dòng. Sau đó, với mỗi nhóm K phần tử liên tiếp trên mỗi dòng, chia cho giá trị tuyệt đối lớn nhất trong nhóm.

Input:

- 3 số nguyên M, N, K
- Mảng số thực A gồm M*N phần tử

Output:

- Mảng số thực gồm M*Nphần tử

Giới hạn:

- 1 <= M*N <= 10^10
- 1 <= K <= N