Denny Cheong
Transformer Report
14/9/2024

I wrote most of the comments regarding my assignment in the code. I learned a lot through this. I think that when thinking about the transformer, it’s really important to consider the number of batches and just the number of dimensions you are working with. The code came up with a bunch of errors in the beginning because I found out in the middle of the code that multiheadattention forward function originally took in one x vector when it was actually supposed to take multiple. That took me a long time to fix. Other than that, it was really nice to learn the difference between nn.Embedding and F.embedding. It was a great experience learning the different mechanisms in pytorch that weren't really used in numpy. I get the following result after creating the transformer.

```
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/tqdm/auto.py:21: TqdmWarning: IProgress not found. Please update jupyter and ipy
  from .autonotebook import tqdm as notebook_tqdm
Epoch 1/50, Loss: 5.1616
Epoch 2/50, Loss: 4.7366
Epoch 3/50, Loss: 4.3905
Epoch 4/50, Loss: 4.0632
Epoch 5/50, Loss: 3.7684
Epoch 6/50, Loss: 3.5218
Epoch 7/50, Loss: 3.2887
Epoch 8/50, Loss: 3.0923
Epoch 9/50, Loss: 2.9154
Epoch 10/50, Loss: 2.7203
Epoch 11/50, Loss: 2.5692
Epoch 12/50, Loss: 2.3899
Epoch 13/50, Loss: 2.2471
Epoch 14/50, Loss: 2.1219
Epoch 15/50, Loss: 1.9659
Epoch 16/50, Loss: 1.8377
Epoch 17/50, Loss: 1.7242
Epoch 18/50, Loss: 1.6129
Epoch 19/50, Loss: 1.5000
Epoch 20/50, Loss: 1.3911
Epoch 21/50, Loss: 1.3013
Epoch 22/50, Loss: 1.2273
Epoch 23/50, Loss: 1.1481
Epoch 24/50, Loss: 1.0601
Epoch 25/50, Loss: 0.9960
...
Epoch 47/50, Loss: 0.2763
Epoch 48/50, Loss: 0.2634
Epoch 49/50, Loss: 0.2515
Epoch 50/50, Loss: 0.2422
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```python
# 테스트 데이터셋 생성 (학습 데이터와 동일한 방식으로 생성)
test_dataset = PatternDataset(num_samples=1000)  # 테스트용 샘플 수
test_dataloader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)

def test_model(model, dataloader):
    model.eval()  # 평가 모드로 설정
    total_correct = 0
    total_samples = 0

    with torch.no_grad():  # 그래디언트 계산 중지 (평가 시에는 필요하지 않음)
        for src, tgt in dataloader:
            # 입력 시퀀스 준비
            tgt_input = torch.zeros_like(tgt).unsqueeze(1)  # 예측 시작을 위한 빈 타겟 시퀀스
            tgt = tgt.unsqueeze(1)  # 타겟을 2D 텐서로 변환

            # 모델에 입력 시퀀스를 전달하고 예측 값 생성
            output = model(src, tgt_input)
            predicted = output.argmax(dim=-1)  # 예측 결과는 argmax를 통해 얻음

            # 실제 타겟과 예측값 비교
            correct = (predicted.view(-1) == tgt.view(-1)).sum().item()
            total_correct += correct
            total_samples += tgt.size(0)

    # 정확도 계산
    accuracy = total_correct / total_samples * 100
    print(f"Test Accuracy: {accuracy:.2f}%")

# 학습된 모델 테스트
test_model(model, test_dataloader)
```

(parameter) model: Unknown

```
Test Accuracy: 100.00%
```