深度学习

兰韵诗

本次Lab没有作业

Lab8参考代码

```python
def decode(self, dec_y, enc_hidden, state, is_first_step = True):
    dec_embs = self.emb_layer(dec_y)
    trg_len = dec_y.shape[1]

    outputs = []
    if is_first_step:
        state = self.mlp1(enc_hidden).mean(1).unsqueeze(1)
        # print(state.shape)
    else:
        state = state.transpose(1, 0)


    for t in range(trg_len):
        scores = torch.bmm(state, enc_hidden.transpose(2, 1))

        # 计算attention weights
        alpha = self.softmax(scores)  # (batch_size, 1, seq_len)

        # 计算context vector
        cont_vec = torch.bmm(alpha, enc_hidden).squeeze(1)
        input_vec = torch.cat([cont_vec, dec_embs[:, t, :]], 1).unsqueeze(1)
        sent_hidden, state = self.decoder(input_vec, state.transpose(0, 1))
        state = state.transpose(0, 1)

        # 输出预测字符
        pred = self.mlp2(sent_hidden)
        outputs += [pred]

    sent_outputs = torch.cat(outputs, dim = 1)

    return sent_outputs, state
```

常见错误

1. decoder 初始状态计算错误

- decoder 初始状态错误地只用encoder最后一个时间步的输出作为初始状态，忽略了题目中要求的对所有 encoder 输出的平均

代码块

```
query = state.transpose(0, 1)
```

代码块

```
if t == 0 and is_first_step:
    query = enc_hidden[:, -1, :]  #第一步用编码器的最后隐藏状态
else:
    query = state.squeeze(0)    #后续步骤使用当前的隐藏状态
```

- 初始状态初始化顺序错误，`mean` 应该作用在 `mlp1(enc_hidden)` 上而非先 `mean` 后 `mlp1`

代码块

```
if is_first_step:
    state = self.mlp1(enc_hidden.mean(dim=1, keepdim=True)).permute(1, 0, 2)  # (1, B, H)
```

正确答案是先将 encoder 的所有时间步输出 `enc_hidden` 通过 `mlp1` 映射，再在时间维度上取均值：

代码块

```
state = self.mlp1(enc_hidden).mean(1).unsqueeze(1)
```

常见错误

2. Attention机制中的key错误

错误地把 `mlp1(enc_hidden)` 作为key，按照题目要求的模型结构不需要额外变换，key就是 `enc_hidden`

代码块

```
keys = self.mlp1(enc_hidden)
...
scores = torch.bmm(query, keys.transpose(1, 2))
```

常见错误

3. Attention机制中的query错误

错误地使用 decoder 输入作为 query，应该使用当前 decoder 的隐藏状态作为 query

代码块

```
# 当前时间步的解码器输入
dec_input = dec_emb[:, t, :].unsqueeze(1)  # (B, 1, E)

# 计算注意力分数（点积模型）
attention_scores = torch.bmm(dec_input, enc_hidden.transpose(1, 2))  # (B, 1, S)
```

错误使用 `mlp1` 投影 query，按照题目要求的模型结构，`mlp1` 仅用于初始化 decoder 的初始状态，不应再用于 Attention 阶段。

代码块

```
query = self.mlp1(query)
```

常见错误

4.缺少基于时间步的解码循环

缺乏 for t in range(seq_len) 的解码循环，而是直接将所有时间步的 decoder embedding 和 context 送入 RNN，与题目要求的解码方式不一致。

代码块

```
if is_first_step:
    # 推理阶段，每次输入1个字符，S=1
    query = dec_emb  # (B, 1, E)
    key = enc_hidden  # (B, S_enc, H)

    # 点积 attention: (B, 1, E) × (B, S_enc, E)^T -> (B, 1, S_enc)
    attention_scores = torch.bmm(query, key.transpose(1, 2))  # (B, 1, S_enc)
    attention_weights = self.softmax(attention_scores)  # (B, 1, S_enc)

    context_vector = torch.bmm(attention_weights, enc_hidden)  # (B, 1, H)
    dec_input = torch.cat((query, context_vector), dim=-1)  # (B, 1, E+H)

else:
    # 训练阶段：批量处理整个序列
    B, S, E = dec_emb.size()
    query = dec_emb  # (B, S, E)
    key = enc_hidden  # (B, S, H)

    attention_scores = torch.bmm(query, key.transpose(1, 2))  # (B, S, S)
    attention_weights = self.softmax(attention_scores)  # (B, S, S)
    context_vector = torch.bmm(attention_weights, key)  # (B, S, H)

    dec_input = torch.cat((query, context_vector), dim=-1)  # (B, S, E+H)

dec_output, state = self.decoder(dec_input, state)  # (B, S, H)
sent_outputs = self.mlp2(dec_output)  # (B, S, vocab_size)

return sent_outputs, state
```