

Lab 5：循环神经网络

理论题

题目1：推导公式(6.40)和公式(6.41)中的梯度。

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \sum_{t=1}^T \sum_{k=1}^t \delta_{t,k} \mathbf{x}_k^\top, \quad (6.40)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \sum_{t=1}^T \sum_{k=1}^t \delta_{t,k}. \quad (6.41)$$

解答：

因为

$$\mathbf{z}_k = \mathbf{U}\mathbf{h}_{k-1} + \mathbf{W}\mathbf{x}_k + \mathbf{b}$$

所以

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{W}} &= \sum_{t=1}^T \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k} \frac{\partial \mathbf{z}_k}{\partial \mathbf{W}} \\ &= \sum_{k=1}^t \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k} \mathbf{x}_k^\top \end{aligned}$$

又因为定义了误差项 $\delta_{t,k}$ 为第 t 时刻的损失对第 k 步隐藏神经元的净输入 \mathbf{z}_k 的导数，所以上式等于

$$\sum_{t=1}^T \sum_{k=1}^t \delta_{t,k} \mathbf{x}_k^\top, \quad (6.40)$$

即公式(6.40)

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \sum_{t=1}^T \sum_{k=1}^t \delta_{t,k} \mathbf{x}_k^\top$$

同理可得公式(6.41)

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \sum_{t=1}^T \sum_{k=1}^t \delta_{t,k}$$

题目2：试验证公式(6.31)中的 \mathbf{z}_k ($\mathbf{z}_k = \mathbf{U}\mathbf{h}_{k-1} + \mathbf{W}\mathbf{x}_k + \mathbf{b}$) 对 u_{ij} 直接求偏导数 $\frac{\partial^+ \mathbf{z}_k}{\partial u_{ij}}$ 等价于递归下去对 \mathbf{h}_{k-1} 接着求导。

$$\frac{\partial \mathcal{L}_t}{\partial u_{ij}} = \sum_{k=1}^t \frac{\partial^+ \mathbf{z}_k}{\partial u_{ij}} \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k} \quad (6.31)$$

解答：

由于隐状态 \mathbf{h}_{k-1} 依赖于前一步的隐状态 \mathbf{h}_{k-2} ，故可以通过链式法则递归求导。由于

$$\begin{aligned} \mathbf{z}_k &= \mathbf{U}\mathbf{h}_{k-1} + \mathbf{W}\mathbf{x}_k + \mathbf{b}, \\ \mathbf{h}_{k-1} &= \text{activation}(\mathbf{z}_{k-1}) \end{aligned}$$

当递归传播到第 $k - 1$ 步的时候：

- 对 z_k 求导需要用到隐状态梯度： $\frac{\partial z_k}{\partial h_{k-1}} = U$ ，表明当前时间步的梯度将被传递到 U 上
- 递归对 h_{k-2} 求导时，将通过上一步 z_{k-2} 的依赖项继续传播
- 最终，完整的梯度展开形式为

$$\frac{\partial \mathcal{L}_t}{\partial u_{ij}} = \sum_{k=1}^t \frac{\partial \mathcal{L}_t}{\partial z_k} \frac{\partial z_k}{\partial h_{k-1}} \cdots \frac{\partial z_1}{\partial u_{ij}}$$

其中每一步递归传递都是由隐状态的依赖性实现的。故总的来说，递归求导过程中， h_{k-1} 的梯度被依赖结构传播到 z_k ，最终结果在数值上等价于直接求导的结果。

代码题

问题描述

利用循环神经网络(LSTM)，实现简单的古诗生成任务

代码补全

根据注释提示，在 `rnn.py` 中补全 `RNN` 的定义如下：

```
# RNN模型
# 模型可以根据当前输入的一系列词预测下一个出现的词是什么
class RNN_model(nn.Module):
    def __init__(self, vocab_len, word_embedding, embedding_dim,
lstm_hidden_dim):
        super(RNN_model, self).__init__()
        self.word_embedding_lookup = word_embedding
        self.vocab_length = (
            vocab_len # 可选的单词数目 或者说 word embedding层的word数目
        )
        self.word_embedding_dim = embedding_dim
        self.lstm_dim = lstm_hidden_dim
        #####
        # 这里你需要定义 "self.rnn_lstm"
        # 其中输入特征大小是 "word_embedding_dim"
        # 输出特征大小是 "lstm_hidden_dim"
        # 这里的LSTM应该有两层，并且输入和输出的tensor都是(batch, seq, feature)大小
        # (提示：LSTM层或许torch.nn中有对应的网络层,pytorch官方文档也许有说明)
        # 填空：
        self.device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
        self.rnn_lstm = nn.LSTM(
            input_size=self.word_embedding_dim,
            hidden_size=self.lstm_dim,
            num_layers=2,
            batch_first=True,
        )
        #####
        self.fc = nn.Linear(self.lstm_dim, self.vocab_length)
        nn.init.xavier_uniform_(self.fc.weight)

    def forward(self, sentence, batch_size, is_test=False):
```

```

        batch_input = self.word_embedding_lookup(sentence).view(
            batch_size, -1, self.word_embedding_dim
        )
        #####
        # 这里你需要将上面的"batch_input"输入到你在rnn模型中定义的lstm层中
        # lstm的隐藏层输出应该被定义叫做变量"output",
        # 初始的隐藏层(initial hidden state)和记忆层(initial cell state)应该是0向量.
        # 填空
        h0 = torch.zeros(2, batch_size, self.lstm_dim).to(self.device)
        c0 = torch.zeros(2, batch_size, self.lstm_dim).to(self.device)
        output, _ = self.rnn_lstm(batch_input, (h0, c0))
        #####
        out = output.contiguous().view(-1, self.lstm_dim)
        out = self.fc(out) # out.size: (batch_size * sequence_length
, vocab_length)
        if is_test:
            # 测试阶段(或者说生成诗句阶段)使用
            prediction = out[-1, :].view(1, -1)
            output = prediction
        else:
            # 训练阶段使用
            output = out
        return output

```

在 `main.py` 中, 补全预处理诗句的函数 `process_poems` 如下:

```

def process_poems(file_name):
    # 读取文件
    poems = []
    with open(
        file_name,
        "r",
        encoding="utf-8",
    ) as f:
        for line in f.readlines():
            try:
                content = line.rstrip("\n")
                content = start_token + content + end_token
                poems.append(content)
            except ValueError as e:
                pass
    #####
    # 填空-----
    words = [word for poem in poems for word in poem]
    words_counter = collections.Counter(words)
    words = sorted(words_counter.items(), key=lambda x: -x[1])
    words, _ = zip(*words)

    # 建立字符和索引间的映射
    word_int_map = {word: idx for idx, word in enumerate(words)}
    int_word_map = {idx: word for idx, word in enumerate(words)}

    # 将诗句编码
    poems_vector = [[word_int_map[word] for word in poem] for poem in poems]
    #####

```

```
print(np.array(poems_vector).shape) # 应该为 (12842, 26)
print(len(word_int_map.keys())) # 应该为 2001
print(len(int_word_map.keys())) # 应该为 2001
return poems_vector, word_int_map, int_word_map
```

为加速训练，将模型和数据放在GPU上：

```
my_device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
(...).to(my_device)
```

运行 `main.py`，训练并测试生成古诗：

```
epoch 30 batch number 395 loss is: 3.3753316402435303
epoch 30 batch number 396 loss is: 3.504737615585327
epoch 30 batch number 397 loss is: 3.5133848190307617
epoch 30 batch number 398 loss is: 3.409646987915039
epoch 30 batch number 399 loss is: 3.3393900394439697
epoch 30 batch number 400 loss is: 3.5309078693389893
finish save model of epoch : 30!
epoch using time 1.664
(12842, 26)
2001
2001
G日照香炉生，风开古木清。何时此相见，犹作白云心。E
(12842, 26)
2001
2001
G清明时节雨，独自林塘。草带青山远，烟微落日寒。E
(12842, 26)
2001
2001
G风光随处见，风雨夜来深。今日阳城里，相逢不可闻。E
(12842, 26)
2001
2001
G花开白露滴，花满绿萝开。此景皆如此，无心更有情。E
(12842, 26)
2001
2001
```

```
G雪后秋光媚，风开古木清。何时此相见，犹作白云心。E
(12842, 26)
2001
2001
G月明寒草绿，风起竹林风。此地难相见，何人更见招。E
(12842, 26)
2001
2001
G雨洗青山绿，花随绿殿新。谁知旧溪月，不见落花前。E
(12842, 26)
2001
2001
G日日千年月，山中见竹林。此时从此别，相见不知心。E
(12842, 26)
2001
2001
G朝来一里别，此别更何之。一路无人识，东南归不归。E
(12842, 26)
2001
2001
G三月不可见，望乡今已微。今宵见秋月，独自林塘。E
(12842, 26)
2001
2001
G九陌通天苑，东风起鼓鼙。江湖春不尽，风雨夜猿飞。E
PS D:\RUC\CS\DeepLearning\labs\lab5\src> █
```