NCTU Spring 2020 Deep Learning and Practice

Lab #4

0616015 劉姿利

Table of Content

1 Introduction

這次的 lab 要用 LSTM 來實作一個矯正錯字的 RNN，並以 BLEU-4 來 evaluate。

2 Derivation of BPTT

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}$$
$$\boldsymbol{h}^{(t)} = \tanh\left(\boldsymbol{a}^{(t)}\right)$$
$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}$$
$$\hat{\boldsymbol{y}}^{(t)} = \operatorname{softmax}\left(\boldsymbol{o}^{(t)}\right)$$
$$p_{\text{model}}\left(\boldsymbol{y}^{(t)} | \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}\right) = \prod_i \left(\hat{y}_i^{(t)}\right)^{\mathbf{1}\left(y_i^{(t)}=1\right)}$$
$$L^{(t)} = -\log p_{\text{model}}\left(\boldsymbol{y}^{(t)} | \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}\right)$$
$$L\left(\left\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}\right\}, \left\{\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(t)}\right\}\right) = \sum_t L^{(t)}$$

$$\nabla_W L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right)\left(\nabla_W h_i^{(t)}\right)$$

$$\nabla_{\boldsymbol{h}^{(t)}} L = \left(\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}\right)^T (\nabla_{\boldsymbol{h}^{(t+1)}} L) + \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}}\right)^T (\nabla_{\boldsymbol{o}^{(t)}} L)$$

$$= \boldsymbol{W}^T \boldsymbol{H}^{(t+1)} (\nabla_{\boldsymbol{h}^{(t+1)}} L) + \boldsymbol{V}^T (\nabla_{\boldsymbol{o}^{(t)}} L)$$

$$\boldsymbol{H}^{(t+1)} = \left(\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{a}^{(t+1)}}\right)^T$$

$$\nabla_{o^{(t)}} L = \hat{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{(t)}$$

$$\nabla_{\boldsymbol{h}^{(\tau)}} L = \boldsymbol{V}^T (\nabla_{\boldsymbol{o}^{(t)}} L) = \boldsymbol{V}^T \left(\hat{\boldsymbol{y}}^{(\tau)} - \boldsymbol{y}^{(\tau)}\right) \text{ where } \tau \text{ is the last}$$

$$\nabla_{\boldsymbol{W}} L = \sum_t \boldsymbol{H}^{(t)} (\nabla_{\boldsymbol{h}^{(t)}} L) \boldsymbol{h}^{(t-1)T}$$

3 Implementation Details

3.1 Models

將 Sample Code 中 Encoder 與 Decoder 的 GRU 換成 LSTM，而 training 時 encoder_hidden 的 initialization 改成

```
encoder_hidden = (encoder.initHidden(), encoder.initHidden())
```

3.1.1 Encoder

```python
class EncoderRNN(nn.Module):
    def __init__(self, input_size, hidden_size):
        super(EncoderRNN, self).__init__()
        self.hidden_size = hidden_size

        self.embedding = nn.Embedding(input_size, hidden_size)
        self.lstm = nn.LSTM(hidden_size, hidden_size)

    def forward(self, input, hidden):
        # input = input.view(-1)
        # print(input.shape)
        # print(input)
        embedded = self.embedding(input).view(1, 1, -1)
        output = embedded
        output, hidden = self.lstm(output, hidden)
        return output, hidden

    def initHidden(self):
        return torch.zeros(1, 1, self.hidden_size, device=device)
```

3.1.2 Decoder

```
class DecoderRNN(nn.Module):
    def __init__(self, hidden_size, output_size):
        super(DecoderRNN, self).__init__()
        self.hidden_size = hidden_size

        self.embedding = nn.Embedding(output_size, hidden_size)
        self.lstm = nn.LSTM(hidden_size, hidden_size)
        self.out = nn.Linear(hidden_size, output_size)
        # self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden):
        output = self.embedding(input).view(1, 1, -1)
        output = F.relu(output)
        output, hidden = self.lstm(output, hidden)
        output = self.out(output[0])
        return output, hidden

    def initHidden(self):
        return torch.zeros(1, 1, self.hidden_size, device=device)
```

3.1.3 Dataloader

```
class DataSet(data.Dataset):
    def __init__(self, mode):
        path = 'data/'+mode+'.json'

        f = open(path)
        f_dict = json.loads(f.read())
        f.close()

        self.input = []
        self.target = []
        for voc in f_dict:
            for _in in voc['input']:
                self.input.append(_in)
                self.target.append(voc['target'])

        self.rand = [ i for i in range(len(self.input)) ]
        random.shuffle(self.rand)

    def __getitem__(self, idx):
        idx = idx % len(self)
        return self.input[self.rand[idx]], self.target[self.rand[idx]]

    def __len__(self):
        return len(self.input)
```

另外 data preprocessing 我定義了幾個 functions：

- `charToIndex(c)`：將字母 c 轉為特定的 token 值
- `indexToChar(idx)`：`chatToIndex(c)` 的 inverse
- `stringToTorch(str)`：將字串轉為對應的 tensor

3.2 Screenshot of the Testing Part

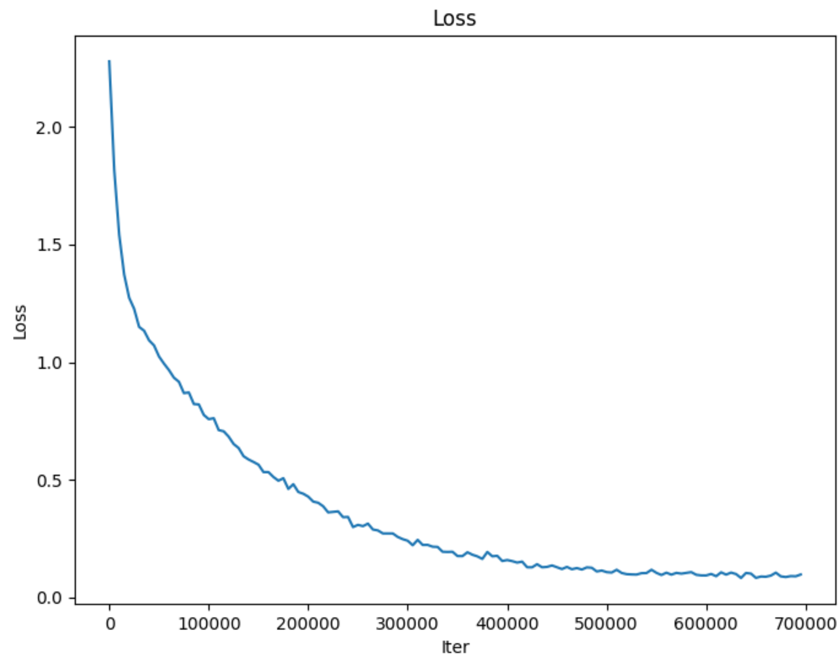
```python
def test(input_vocab, encoder, decoder):
    encoder.eval()
    decoder.eval()
    global print_output

    with torch.no_grad():
        input_tensor = stringToTorch(input_vocab, is_tar=True).to(device)

        encoder_hidden = (encoder.initHidden(), encoder.initHidden())
        input_length = input_tensor.size(0)

        for ei in range(input_length):
            encoder_output, encoder_hidden = encoder(input_tensor[ei], encoder_hidden)

        decoder_input = torch.tensor([[SOS_token]], device=device)
        decoder_hidden = encoder_hidden
        decoder_outputs = ''

        for di in range(25):
            decoder_output, decoder_hidden, = decoder(decoder_input, decoder_hidden)
            topv, topi = decoder_output.topk(1)
            decoder_input = topi.squeeze().detach()  # detach from history as input
            decoder_outputs += indexToChar(decoder_input)

            if decoder_input.item() == EOS_token:
                break

    return decoder_outputs
```

4 Results

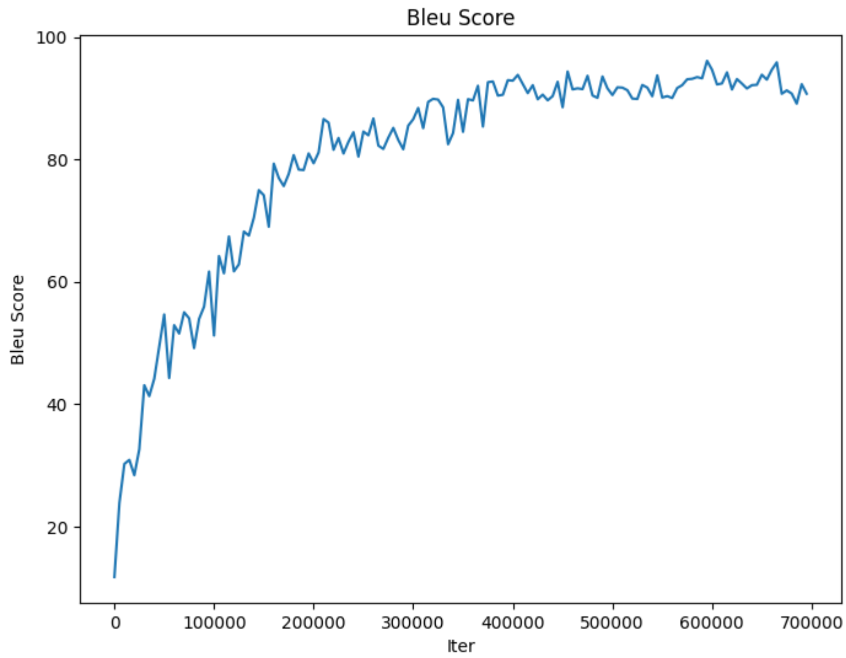
4.1 Spelling Correction

```
bleu_score:  0.9610280341854175
---
< substracts
= subtracts
> subtracts
---
< decieve
= deceive
> deceive
---
< daing
= doing
> doing
---
< repatition
= repartition
> repartition
---
```

4.2 Training Loss Curve



4.2 BLEU-4 Score Testing Curve



5 Discussion

這次的 lab 我花了非常多時間在搞懂 PyTorch RNN 的 input、target 維度要怎麼給，跟以往 CNN (batch, channel, h, w) 不同 RNN 是 (sequence, batch, input_size)。還有一個不同的就是，以往 trainning 都是 accuracy 穩定上升，除非 train 到 overfit 才開始下降，這次的 bleu score 卻是上上下下幅度很大的變動。

曾嘗試使用過 word dropout 卻沒有什麼效果，不過在 trainning 的過程中，我發現 LSTM 都是短的字先對，或是有特殊長字節 (ex. bility) 的 inputs 表現比較穩定，所以我中途有稍微濾出比較長的 training data 特別多 train 幾次，表現就比較穩定了。