# Lab3: Conditional Sequence-to-sequence VAE

0851915 徐玉山

Report (50%)
- ◆ Introduction (5%)
- ◆ Implementation details (15%)

A. Describe how you implement your model (encoder, decoder, reparameterization trick, dataloader, etc.). (paste/screenshot your code)

```
class EncoderRNN(nn.Module):
    def __init__(self, word_size=28, hidden_size = 256, latent_size=32):
        super(EncoderRNN, self).__init__()
        self.word_size = word_size
        self.hidden_size = hidden_size
        self.condition_size = condition_size
        self.latent_size = latent_size

        self.word_embedding = nn.Embedding(word_size, hidden_size)
        self.gru = nn.GRU(hidden_size, hidden_size)

    def forward(self, inputs, hidden):
        # get (seq, 1, hidden_size)
        x = self.word_embedding(inputs).view(-1, 1, self.hidden_size)
        # get (seq, 1, hidden_size), (1, 1, hidden_size)
        outputs, hidden = self.gru(x, hidden )
        return outputs, hidden
```

Encoder:
使用 gru 來做 RNN
hidden unit 維度 256
輸出的 latent vector 維度 32
輸入的部分給單一字元的 one-hot 像是 [1, 0, 0, … ] 這樣的並用 nn.embedding() 轉換成維度 256 的 input tensor

```
c = self.condition( condition_scalar )
encoder_hidden = torch.cat( (self.initHidden(), c), dim = 2)
encoder_output = 0
input_length = input_tensor.size(0)
for i in range(input_length):
    encoder_output, encoder_hidden = self.E( input_tensor[i] , encoder_hidden)
```

Condition 也要 embedding 過，維度為 8。因此 init_hidden 的維度為 256-8 = 248

```python
class Reparameter(nn.Module):
    def __init__(self, encoder_output_size =256, latent_size = 32):
        super(Reparameter, self).__init__()
        self.latent_size = latent_size
        self.mu = nn.Linear(encoder_output_size, latent_size)
        self.log_var = nn.Linear(encoder_output_size, latent_size)

    def forward(self, encoder_output):
        mu = self.mu(encoder_output)
        log_var = self.log_var(encoder_output)
        std = torch.exp(0.5 * log_var)
        #eps = torch.randn_like(std)
        eps = torch.randn(1,1,self.latent_size,device=device)
        return mu, log_var, mu + eps*std
```

Reparameter:

把 encoder 最後的 hidden 拿來當 input。

計算出 mu, log_var 並從高斯分布中抽一個 eps 計算出 latent vector

```python
class DecoderRNN(nn.Module):
    def __init__(self, word_size=28, hidden_size=256, latent_size=32, condition_size=8):
        super(DecoderRNN, self).__init__()
        self.hidden_size = hidden_size
        self.word_size = word_size

        self.latent_to_hidden = nn.Linear( latent_size+condition_size, hidden_size )

        self.word_embedding = nn.Embedding(word_size, hidden_size)# logic bug maybe...
        self.gru = nn.GRU(hidden_size, hidden_size)
        self.out = nn.Linear(hidden_size, word_size)


    def forward(self, x, hidden ):
        # get (1, 1, hidden_size)
        x = self.word_embedding(x).view(1, 1, self.hidden_size)
        # get (1, 1, hidden_size) (1, 1, hidden_size)
        output, hidden = self.gru(x, hidden )
        # get (1, word_size)
        output = self.out(output[0])
        return output, hidden
```

Decoder:

拿 latent vector 經過 nn.linear() 轉換成維度 256 當作 init_hidden

第一個字元輸入 [SOS]，跟 Encoder 一樣用 nn.embedding()轉換成維度 256 的 input tensor

若使用 teacher forcing，把 target 的字元輸入

若否則把 decoder 的輸出字元當作輸入。

B. Specify the hyperparameters (KL weight, learning rate, teacher forcing ratio, epochs, etc.)

```python
def KLDW(iter, method = ''):
    return 0
    if method == 'M': #Monotonic
        slope = 0.001
        w = iter * slope
        if w > 0.1:
            w = 0.1
        return w
    elif method == 'C': #Cyclical
        slope = 0.001
        scope = 10
        w = (iter % scope) * slope
        if w > 0.01:
            w = 0.01
        return w
```

teacher_forcing_ratio = 0.5

learning_rate = 0.01

epochs = 200000 (我寫的程式不知道為什麼會 memory leak, 所以我常常需要關掉然後再拿 checkpoint 重新開始訓練)

◆ Results and discussion (30%)

A. Show your results of tense conversion and generation (5%)

```
Sample out:
 mumbling -> mumbling
Save model...
Save model complete ...
('abandon', 'abandoned', 0, 3, 'abondonEOS') 0.3549481056010053
('abet', 'abetting', 0, 2, 'abetEOS') 0.345720784641941
('begin', 'begins', 0, 1, 'beginEOS') 0.7598356856515925
('expend', 'expends', 0, 1, 'expendEOS') 0.8091067115702212
('sent', 'sends', 3, 1, 'sentEOS') 0.26591479484724945
('split', 'splitting', 0, 2, 'splitEOS') 0.44632361378533286
('flared', 'flare', 3, 0, 'flaredEOS') 0.8187307530779819
('functioning', 'function', 2, 0, 'functioningEOS') 0.6872892787909722
('functioning', 'functioned', 2, 3, 'functioningEOS') 0.68752775993657
('healing', 'heals', 2, 1, 'healingEOS') 0.4482700320176827
Average bleu score =  0.562366751992055
```

上面是訓練的抽樣

下方是 test data 的結果

B. Plot the Crossentropy loss, KL loss and BLEU-4 score curves during training and discuss the results according to your setting of teacher forcing ratio, KL weight, and learning rate. Notice: This part mainly focuses on your discussion, if you simply just paste your results, you will get a low score.