[Lab5] Conditional Sequence-to-Sequence VAE

0616014 楊政道

May 12, 2020

1 Introduction

1.1 實驗目的

本次實驗目的為用 LAB4 的 seq2seq rnn model 來建構一個 conditional seq2seq VAE 來完成英文單字的時態轉換以及用 Gaussian Noise 來生成出單字的 4 個時態。

1.2 Dataset

在訓練集中, 有 1227 種單字, 每種單字都有對應的 4 個時態變化, 例如 (dart darts darting darted) 分別對應到簡單式, 三單, 進行式還有過去式。

而在測試資料集中, 有 10 個單字組, 每組單字組包含兩個同種單字但是不同時態, 用來測試 CVAE 的單字時態轉換能力。

1.3 Project Structure

```
DLP_LAB4_0616014_ 楊政道.zip
├── dataset/
│   └── dataloader.py
├── model/
│   ├── Encoder.py
│   ├── Decoder.py
│   └── Seq2Seq.py
├── weight/
│   └── record.json
├── train.py
└── evaluate.py
```

1.3.1 data/dataloader.py

對 train.txt 和 test.txt 做資料處理, 並建構字母集來 encode 和 decode 字串和張量。

1.3.2 model/Encoder.py

定義 Seq2Seq 架構中 Encoder 的內容, 使用 LSTM 來實作。

1.3.3 model/Decoder.py

定義 Seq2Seq 架構中 Decoder 的內容, 使用 LSTM 來實作。

1.3.4 model/Seq2Seq.py

融合 Encoder 和 Decoder 兩個 model, 並且加入了 latent2hidden, mean-fc 和 logvar-fc 這些結構, 來組合出完整的 CVAE 架構。

1.3.5 weight/record.json

若訓練出來的 model 在 BLEU-4 score 的表現上比之前的 model weight 還突出, 就紀錄起來。

1.3.6 train.py

定義 train 函數來訓練 model。

1.3.7 evaluate.py

定義 evalute 函數來做時態轉換和 Gaussian score 測驗。

2 Derivation of CVAE

The marginal likelihoods of $x^{(i)}$

$$\begin{aligned} logp_\theta(x^{(i)}) &= logp_\theta(x^{(i)}, z) - logp_\theta(z|x^{(i)}) \\ &= D_{KL}(q_\phi(z|x^{(i)})\|p_\theta(z|x^{(i)})) + L(\theta, \phi; x^{(i)}) \\ &\geq L(\theta, \phi; x^{(i)}) = E_{q_\phi(z|x^{(i)})}[-logq_\phi(z|x^{(i)}) + logp_\theta(x^{(i)}, z)] \end{aligned} \tag{1}$$

$$\begin{aligned} L(\theta, \phi; x^{(i)}) &= E_{q_\phi(z|x^{(i)})}[-logq_\phi(z|x^{(i)}) + logp_\theta(x^{(i)}|z) + logp_\theta(z)] \\ &= -D_{KL}(q_\phi(z|x^{(i)})\|p_\theta(z)) + E_{q_\phi(z|x^{(i)})}[logp_\theta(x^{(i)}|z)] \end{aligned} \tag{2}$$

3 Implementation details

3.1 DataLoader

3.1.1 Dictionary

建立字母集, 其中包含 encode 和 decode 的操作方便轉換資料格式。

```
class Dictionary:
    def __init__(self, sigma):
        self.char2idx = {'SOS': 0, 'EOS': 1, 'UNK': 2}
        self.idx2char = {0: '', 1: '', 2: '?'}

        for c in sigma:
            if c in self.char2idx:
                continue
            idx = len(self.char2idx)
            self.char2idx[c] = idx
            self.idx2char[idx] = c
    def encode(self, w):
        return torch.tensor(
              [ self.char2idx[c] if c in self.char2idx else 2 for c in w ]
            + [ 1 ],
        device=device).view(-1, 1)
    def decode(self, t):
        return ''.join([self.idx2char[v.item()] for v in t.view(-1)])
```

3.1.2 TrainDataset

訓練用資料讀出所有資料並且壓成一個維度, 可以用 index 來判斷該單字的時態。___getitem___ 則回傳該 index 的英文單字和時態。

```
class TrainDataset(Dataset):
    def __init__(self, path):
        self.data = np.loadtxt(path, dtype=np.str).reshape(-1)
        self.dict = Dictionary("abcdefghijklmnopqrstuvwxyz")
        self.tense = {
            'sp': 0,
            'tp': 1,
            'pg': 2,
            'p': 3
        }
    def __len__(self):
        return len(self.data)
```

```
    def __getitem__(self, index):
        return self.data[index], index % len(self.tense)
```

3.1.3 TestDataset

測試用資料集為 10 組單字, 每組都是相同單字但是不同時態, 建立起每個 pair 對應的時態關係, 在 ___getitem___ 時把他們對應的單字和時態回傳。

```
class TestDataset(Dataset):
    def __init__(self, path):
        self.data = np.loadtxt(path, dtype=np.str)
        self.dict = Dictionary("abcdefghijklmnopqrstuvwxyz")
        self.tense = {
            'sp': 0,
            'tp': 1,
            'pg': 2,
            'p': 3
        }
        self.target = [
            ['sp',  'p'],
            ['sp',  'pg'],
            ['sp',  'tp'],
            ['sp',  'tp'],
            ['p',   'tp'],
            ['sp',  'pg'],
            ['p',   'sp'],
            ['pg', 'sp'],
            ['pg', 'p'],
            ['pg', 'tp']
        ]
    def __len__(self):
        return len(self.data)
    def __getitem__(self, index):
        return self.data[index][0], self.tense[self.target[index][0]], self.data[index][1],
            self.tense[self.target[index][1]]
```

3.2 Encoder

Encoder 要做的事情很單純, 就是將輸入的字母進去 embedding function 轉成一個向量, 然後在跟自己上一個 hidden 做 LSTM 運算即可。

```
class EncoderRNN(nn.Module):
    def __init__(self, num_of_word, hidden_size):
        super(EncoderRNN, self).__init__()
        self.hidden_size = hidden_size
        self.embedding = nn.Embedding(num_of_word, hidden_size)
        self.lstm = nn.LSTM(hidden_size, hidden_size)
    def forward(self, input, hidden):
        embedded = self.embedding(input).view(input.size(0), 1, -1)
        output, hidden = self.lstm(embedded, hidden)
        return output, hidden
    def name(self):
        return f'EncoderRNN-{self.hidden_size}'
```

3.3 Decoder

Decoder 要做的事情也很單純, 也是將輸入的字母進去 embedding function 轉成向量, 再和自己上一個 hidden 做 LSTM 運算, output 再接一個 fc 來預測這次要輸出的字母是什麼。

```
class DecoderRNN(nn.Module):
    def __init__(self, num_of_word, hidden_size):
        super(DecoderRNN, self).__init__()
        self.hidden_size = hidden_size
        self.embedding = nn.Embedding(num_of_word, hidden_size)
        self.lstm = nn.LSTM(hidden_size, hidden_size)
```

```
        self.out = nn.Linear(hidden_size, num_of_word)
    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 name(self):
        return f'DecoderRNN-{self.hidden_size}'
```

3.4 CVAE

CVAE 在 training 的階段時, 會收到一個單字和他的時態, 目標是要讓 encoder 學到資料的分佈, 並讓該分佈趨近於 N(0, 1), 再讓 decoder 成功解出該單字。其中 encoder 的初始 hidden 要塞入 conditional vector, 可以從 condition_embedding_function 得出。

離開 encoder 時, 要將 encoder 的 hidden 經過兩個 fc, 一個代表該分佈的平均值, 另一個代表該分佈的 log 變異數。

要進到 decoder 前, 要將估測出的平均值和 log 變異數平移縮放 N(0, 1) 並接上 conditional vector, 形成 decoder 的新 hidden，再讓 decoder 生出對應的單字。

```
class Seq2Seq(nn.Module):
    def __init__(self, hidden_size, num_of_word, cond_size, num_of_cond, latent_size):
        super(Seq2Seq, self).__init__()
        self.cond_embedding = nn.Embedding(num_of_cond, cond_size)
        self.encoder = EncoderRNN(num_of_word, hidden_size)
        self.mean = nn.Linear(hidden_size, latent_size)
        self.logvar = nn.Linear(hidden_size, latent_size)
        self.latent2hidden = nn.Linear(latent_size + cond_size, hidden_size)
        self.latent2cell = nn.Linear(latent_size + cond_size, hidden_size)
        self.decoder = DecoderRNN(num_of_word, hidden_size)
        self.hidden_size = hidden_size
        self.cond_size = cond_size
        self.latent_size = latent_size
    def condition(self, c):
        return self.cond_embedding(torch.LongTensor([c]).to(device)).view(1, 1, -1)
    def sample_z(self):
        return torch.normal(
            torch.FloatTensor([0] * (self.latent_size)),
            torch.FloatTensor([1] * (self.latent_size))
        ).to(device).view(1, 1, -1)
    def initHidden(self, hidden, cell, cond):
        return (
            torch.cat((hidden, cond), dim=2),
            cell
        )
    def forward(self, input, input_c, target, target_c, use_teacher_forcing):
        output, hidden = self.encoder(
            input,
            self.initHidden(
                torch.zeros(self.hidden_size - self.cond_size, device=device).view(1, 1, -1),
                torch.zeros(self.hidden_size, device=device).view(1, 1, -1),
                self.condition(input_c)
            )
        )
        m = self.mean(hidden[0])
        lgv = self.logvar(hidden[0])
        z = self.sample_z() * torch.exp(lgv / 2) + m
        hidden = (
            self.latent2hidden(torch.cat((z, self.condition(target_c)), dim=2).reshape(-1)).
                view(1, 1, -1),
            self.latent2cell(torch.cat((z, self.condition(target_c)), dim=2).reshape(-1)).view
                (1, 1, -1)
        )
```

```
        input = torch.tensor([[0]], device=device)
        ret = []
        for i in range(MAX_LENGTH):
            output, hidden = self.decoder(input, hidden)
            if use_teacher_forcing:
                if i == target.size(0):
                    break
                ret.append(output)
                input = target[i]
            else:
                ret.append(output)
                topv, topi = output.topk(1)
                input = topi.squeeze().detach()
                if input.item() == 1:
                    break
        return torch.stack(ret), m, lgv
```

生成文字的機率分佈使用 sample_z(), 他是一個 Gaussian noise

```
    def sample_z(self):
        return torch.normal(
            torch.FloatTensor([0] * (self.latent_size)),
            torch.FloatTensor([1] * (self.latent_size))
        ).to(device).view(1, 1, -1)
```

3.5 Hyperparameters

- KL weight: monotonic and cyclical
- learning rate: 0.5
- teacher forcing ratio: 0.6

4 Results and discussion

訓練時可以觀察 CrossEntropyLoss 和 KLLoss 來評斷該 CVAE 的學習狀況, 如果 CrossEntropyLoss 愈小, 時態轉換上和生成單字的正確性就會來得比較好, 但是在使用 Gaussian Noise 來生成單字的表現就會比較差, 原因是 Decoder 所希望接收的資料分佈當 KLLoss 很大時並不是 Gaussian Noise, 所以表現就會變差。相反的, 如果 KLLoss 降低, 可以讓 Decoder 接收到的資料分佈愈靠近 Gaussian Noise 分佈, 因次在 generation 的表現上就會有所提升, 但是 CrossEntropy 上升會導致時態轉換和單字正確性下降。所以訓練的目標就是如何讓兩個 Loss 下降, 仰賴於調整 KLD_weight。

若 KLD_weight 設定過大, 會讓 Loss function 較偏向於 KLLoss 而倒置 VAE 無法輸出正確且精準的英文單字還有時態轉換。相反的, 若 KLD_weight 設定過小, 雖然在英文單字時態轉換的 BLEU 分數很高, 但是由於資料分佈並不是趨近於 N(0, 1), 在 generation 時表現就會很差。

4.1 Plot during Training

4.1.1 Monotonic

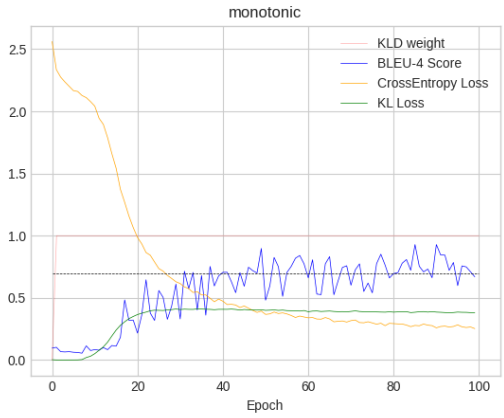

Figure 1: Monotonic KLD weight

4.1.2 Cyclical

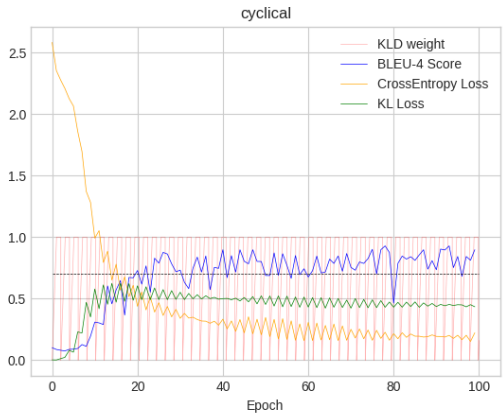

Figure 2: Cyclical KLD weight

可以看出兩個不同的 annealing 策略，結果有些許不同。

第 1 種是 Monotonic 的方法來調整 KLD_weight, 前期先讓 CrossEntropyLoss 訓練下來, 讓 VAE 的輸出先稍微穩定再來調整資料的分佈, 但這就很吃調整前期 KLD_weight 的爬升速度, 如果爬得太快會讓

CrossEntropyLoss 來不及訓練好就受到 KLLoss 的影響, 而使得訓練的結果不理想或是收斂速度太慢。

第 2 種是 Cyclical 的策略, 此種方法讓 CrossEntropyLoss 和 KLLoss 輪流下降, 可以從圖中看到當 KLD_weight 上升的時候, 因為在 Loss function KLLoss 的比重增加, 而 CrossEntropyLoss 就會上升, KLLoss 就會下降。相反的, 當 KLD_weight 掉回 0 的時候, 就是 CrossEntropyLoss 下降的時機, 但是會使得 KLLoss 上升。經過多次的循環, CrossEntropyLoss 和 KLLoss 都會慢慢下降。

從這兩張圖可以看出, Cyclical 策略的前期收斂速度會比 Monotonic 來得快, 後期的在 CrossEntropyLoss 的表現也比較優異。

4.2 Tense conversion

```
< abandon
= abandoned
> abandoned
----------------------
< abet
= abetting
> abetting
----------------------
< begin
= begins
> begins
----------------------
< expend
= expends
> expends
----------------------
< sent
= sends
> sends
----------------------
< split
= splitting
> splitting
----------------------
< flared
= flare
> flare
----------------------
< functioning
= function
> function
----------------------
< functioning
= functioned
> functioned
----------------------
< healing
= heals
> heals
----------------------
BLEU-4 score: 1.0
```

可以看到 CVAE 可以完成英文單字的時態轉換。

4.3 Generation

```
     blimm          blimme         blimming       blimmed
     bestae         bestains       bestaining     bestaeee
---> ensue          ensues         ensuing        ensued
     specion        specceee       specceeing     spinsered
     undrruc        undrruses      undrruiing     undrruted
---> achieve        achieves       achieving      achieved
```

```
     tett            tetts           tetting         tetted
---> recover         recovers        recovering      recovered
---> stalk           stalks          stalking        stalked
     bronn           bronns          bronning        bronned
---> stalk           stalks          stalking        stalked
---> abet            abets           abetting        abetted
     seet            seets           seeting         set
---> chance          chances         chancing        chanced
---> confess         confesses       confessing      confessed
---> crush           crushes         crushing        crushed
---> apprise         apprises        apprising       apprised
     overflae        overflaes       overflaiin      overflaed
     charge          charges         charging        chargeeed
     baught          bausts          bausting        baught
     essape          escapes         escapinng       escaped
     blust           blushes         blushing        blusted
---> breach          breaches        breaching       breached
     wid             wids            widing          wided
     consent         consents        consening       consented
     boll            bolls           bolling         bolled
     cartwhre        cartwhrs        cartwhriing     cartwhre
---> kill            kills           killing         killed
     twish           twishes         twishing        twished
     breater         berrares        berearing       berrared
     prohibit        inslifies       prohibiting     prohibieed
---> dip             dips            dipping         dipped
---> devise          devises         devising        devised
---> supplement      supplements     supplementing   supplemented
---> drag            drags           dragging        dragged
---> display         displays        displaying      displayed
     corre           coreeds         coreeing        coarned
     dismiss         disagies        disagreing      disagiee
     decline         declines        decliniing      declined
     coillot         coillots        coilicing       coiliced
---> grope           gropes          groping         groped
     smoth           smotts          shutting        shutted
     flap            flaps           flaping         flaped
---> switch          switches        switching       switched
---> resound         resounds        resounding      resounded
     conmemn         commands        conmending      commanded
     dive            dives           diving          dived
     negglf          blistsss        blistting       blistted
     saitocize       switicaee       switictiig      switicted
---> preserve        preserves       preserving      preserved
---> arrange         arranges        arranging       arranged
---> resign          resigns         resigning       resigned
---> stimulate       stimulates      stimulating     stimulated
---> flail           flails          flailing        flailed
     befit           beits           befiting        befit
     fill            fixes           fixing          fixed
     exerr           exerrs          exerring        exerred
     attract         attaches        attaching       attached
     foreach         foreaches       foreaching      foreached
     nuffer          nuffers         nuffering       nuffered
     feel            feels           feeling         feel
     pay             pays            paying          had
---> exhibit         exhibits        exhibiting      exhibited
     disagree        disagrees       disagreing      disagreed
---> strut           struts          strutting       strutted
     surrod          murmurs         murmuring       murmured
     decogniee       decognies       decoensing      decoonaedd
     permi           permies         permiing        permied
     divd            divds           dividing        divded
     cxptcipate      disqualifies    cxptcipating    disqualified
---> feign           feigns          feigning        feigned
---> whack           whacks          whacking        whacked
```

```
     oversle          respects         oepleping       oepleped
---> abuse            abuses           abusing         abused
     contine          contines         contining       contined
---> disobey          disobeys         disobeying      disobeyed
---> delay            delays           delaying        delayed
     trotibute        trotibutes       trotibuting     trotibuted
---> approve          approves         approving       approved
     covvice          composes         covvicing       composed
---> persist          persists         persisting      persisted
---> wax              waxes            waxing          waxed
     concern          concerns         concerning      accourted
---> delude           deludes          deluding        deluded
     leet             meets            leeting         leet
---> poison           poisons          poisoning       poisoned
---> continue         continues        continuing      continued
     sesturt          sesturts         sesturting      smothered
     deam             deams            deaming         deamed
     congaaae         cirgulates       congaaiing      congaaled
     chance           chances          chancing        crenched
     precede          preeeee          preceding       preeeedd
     eoss             eosss            eossing         eossed
     accipt           accipts          accipting       accipted
     absign           absirbs          absigning       absirbed
     bleck            blecks           blecking        blecked
     jab              jabs             jabing          jabed
     express          express          expresing       expresed
     receive          records          reterring       reterred
     oummon           oumpans          obcaiding       obtained
Gaussian score: 0.42
```

在 generation 的表現上, CVAE 也有超過 4 成的資料是出現在訓練資料內的。(—> 標示的為在訓練資料內的單字)