DLP Lab4 Conditional Sequence-to-Sequence VAE

0711529 陳冠儒

1. Introduction (5%)

   這次的 lab4 是要使用 lstm conditional vae 來預測 seq2seq 的資料，並且要畫出 Crossentropy loss、KL loss 和 BLEU-4 score 的結果曲線，最後再來比較使用不同的 teacher forcing ratio、KL weight 和 learning rate 會有什麼不同。

   A. Dataset

   - Training

     總共有 1227 筆資料，每一筆資料都是四個不同詞性的相同單字，詞性的順序都是 simple present(sp)、third person(tp)、present progressive(pg)、simple past(p)。

     e.g., [abandon abandons abandoning abandoned]

   - Testing

     總共有 10 筆資料，都是由一種詞性轉變為另一種詞性。

     e.g., [abandon abandoned] (sp → p)

2. Derivation of CVAE (5%)

   Derive from the EM altorithm

   From the EM algorithm with conditional c, latent variables Z and visible variables X.

$$\log p(X \mid c;\theta) = \log p(X, Z \mid c;\theta) - \log p(Z \mid X, c;\theta)$$

   and we introduce an arbitrary distribution q(Z | c)

$$\begin{aligned}
&\int q(Z \mid c)\log p(X \mid c;\theta)dZ \\
&= \int q(Z \mid c)\log p(X, Z \mid c;\theta)dZ - \int q(Z \mid c)\log p(Z \mid X, c;\theta)dZ \\
&= \int q(Z \mid c)\log p(X, Z \mid c;\theta)dZ - \int q(Z \mid c)\log q(Z \mid c)dZ \\
&\qquad + \int q(Z \mid c)\log q(Z \mid c)dZ - \int q(Z \mid c)\log p(Z \mid X, c;\theta)dZ \\
&= \mathcal{L}(X, q, \theta) + KL(q(Z \mid c) \,||\, p(Z \mid X, c;\theta))
\end{aligned}$$

   where,

$$\mathcal{L}(X, q, \theta) = \int q(Z \mid c)\log p(X, Z \mid c;\theta)dZ - \int q(Z \mid c)\log q(Z \mid c)dZ$$

$$KL(q(Z \mid c) \,||\, p(Z \mid X, c;\theta)) = \int q(Z \mid c)\log q(Z \mid c)dZ - \int q(Z \mid c)\log p(Z \mid X, c;\theta)dZ$$

   Introduce distribution q(Z | X; θ′)

   As the equality holds for any choice of q(Z), we introduce a distribution

$q(Z \mid X, c; \theta')$. So that

$$\log p(X \mid c; \theta) - KL(q(Z \mid X, c; \theta') \mid\mid p(Z \mid X, c) = \mathcal{L}(X, q, \theta)$$

and the right hand side can be spell out as

$$\begin{aligned}
&\mathcal{L}(X, q, \theta) \\
&= E_{Z\sim q(Z|X,c;\theta')} \log p(X, Z \mid c; \theta) - E_{Z\sim q(Z|X,c;\theta')} \log q(Z \mid X, c; \theta') \\
&= E_{Z\sim q(Z|X,c;\theta')} \log p(X \mid Z, c; \theta) + E_{Z\sim q(Z|X,c;\theta')} \log p(Z \mid c) \\
&\qquad - E_{Z\sim q(Z|X,c;\theta')} \log q(Z \mid X, c; \theta') \\
&= E_{Z\sim q(Z|X,c;\theta')} \log p(X \mid Z, c; \theta) - KL(q(Z \mid X, c; \theta') \mid\mid p(Z \mid c))
\end{aligned}$$

Instead of directly maximizing the intractable $p(X \mid c; \theta)$, we attempt to maximize

$$\log p(X \mid c; \theta) - KL(q(Z \mid X, c; \theta') \mid\mid p(Z \mid X, c)$$

which amounts to maximizing

$$E_{Z\sim q(Z|X,c;\theta')} \log p(X \mid Z, c; \theta) - KL(q(Z \mid X, c; \theta') \mid\mid p(Z \mid c))$$

So the objective function of conditional VAE is the above function.

3. Derivation of KL Divergence (5%)

We know that the probability density function of Gaussian distribution can be written as:

$$p(z; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} \exp(\frac{-(z-\mu)^2}{2\sigma})$$

After taking the logarithm of above we get:

$$\begin{aligned}
\log p(z;\ \mu, \sigma) &= \log\left(\frac{1}{\sqrt{2\pi\sigma}}\right) + \left(-\frac{(z-\mu)^2}{2\sigma}\right) \\
&= -\frac{1}{2}\log(2\pi\sigma) - \frac{(z-\mu)^2}{2\sigma}
\end{aligned}$$

Given the prior $p(z) \sim N(0, I)$ and the posterior approximation $q(z \mid x; \theta) \sim N(\mu_\theta(x), \Sigma_\theta(x))$, and we calculate the KL divergence:

$$\begin{aligned}
&KL(q(z \mid x; \theta) \mid\mid p(z)) \\
&= \int q(z \mid x; \theta) \log q(z \mid x; \theta)\, dz - \int q(z \mid x; \theta) \log p(z)\, dz \\
&= \int q(z \mid x; \theta)[\log \frac{1}{\sqrt{2\pi \Sigma_\theta(x)}} - \frac{(z-\mu)^2}{2\Sigma_\theta(x)}]dz \\
&\qquad - \int q(z \mid x; \theta)[\log\left(\frac{1}{\sqrt{2\pi I}}\right) - \frac{z^2}{2I}]dz \\
&= \log \frac{1}{\sqrt{2\pi \Sigma_\theta(x)}} \int q(z \mid x; \theta) dz - \frac{1}{2\Sigma_\theta(x)} \int (z-\mu)^2 q(z \mid x; \theta) dz \\
&\qquad - \log\left(\frac{1}{\sqrt{2\pi I}}\right) \int q(z \mid x; \theta) dz + \frac{1}{2I} \int z^2 q(z \mid x; \theta) dz
\end{aligned}$$

$$= \frac{-1}{2}\log 2\pi \sum_{\theta}(x) - \frac{1}{2} + \frac{1}{2}\log 2\pi I + \frac{1}{2}(\mu_\theta(x)^2 + \sum_{\theta}(x))$$

$$= \frac{1}{2}(-1 - \log 2\pi(\Sigma_\theta(x) + I) + \mu_\theta(x)^2 + \Sigma_\theta(x))$$

The above equation could be generalized to multivariate cases (n dimensions) by summing over all the dimensions:

$$\mathrm{KL}(q(z \mid x; \theta) \,||\, p(z)) = \frac{1}{2}\Sigma(-1 - \log 2\pi(\Sigma_\theta(x) + I) + \mu_\theta(x)^2 + \Sigma_\theta(x))$$

So it can be the functions of $\mu_\theta(x)$ and $\Sigma_\theta(x)$, expressed as a closed-form expression.

4. Implementation details (15%)

- Model
  - Architecture (encoder & decoder)

```
Encoder(
  (embedding): Embedding(30, 256)
  (lstm): LSTM(256, 256)
  (hidden2mean): Linear(in_features=256, out_features=32, bias=True)
  (hidden2logv): Linear(in_features=256, out_features=32, bias=True)
  (cell2mean): Linear(in_features=256, out_features=32, bias=True)
  (cell2logv): Linear(in_features=256, out_features=32, bias=True)
)
Decoder(
  (latent2hidden): Linear(in_features=32, out_features=252, bias=True)
  (latent2cell): Linear(in_features=32, out_features=252, bias=True)
  (embedding): Embedding(30, 256)
  (lstm): LSTM(256, 256)
  (out): Linear(in_features=256, out_features=30, bias=True)
  (log_softmax): LogSoftmax(dim=-1)
)
```

  - Reparameterization trick

```python
def reparaterization_trick(self, mean, logv):
    std = torch.exp(0.5*logv)
    eps = torch.randn_like(std)
    return  mean + eps * std
```

```python
encoder_hidden = self.reparaterization_trick(hidden_means, hidden_logv)
encoder_hidden = self.decoder.latent2hidden(encoder_hidden)
encoder_cell = self.reparaterization_trick(cell_means, cell_logv)
encoder_cell = self.decoder.latent2cell(encoder_cell)
```

  - Dataloader

    我參考了 github [1]來製作我的 dataloader，其中幾個要點如下：

    a. split()：將單字拆成一個一個的字母，e.g., alphabet → [a, l, p, h, a, b, e, t]。

    b. sequence_to_indices()：將單字字母轉換為 index。

    c. 使用 EOS 來代表一個單字的結束，並用 PAD 來將每一個字的

長度都用成一樣。

d. mini_batches()：最後在取出的時候，將每個字母按照時間序列排序。

e.g.,

```
tensor([[10, 17, 11],
        [ 6, 10, 12],
        [ 9,  6,  9],
        [10, 14, 13],
        [ 9, 21, 15],
        [13, 12, 16],
        [10,  7, 12],
        [ 6,  1,  7],
        [11,  2,  1],
        [ 1,  2,  2],
        [ 2,  2,  2],
        [ 2,  2,  2],
        [ 2,  2,  2],
        [ 2,  2,  2],
        [ 2,  2,  2],
        [ 2,  2,  2]],
```

從左到右分別為單字：insisting、pinched、gestured。

※ Reference

[1] https://github.com/zake7749/Sequence-to-Sequence-101/blob/master/Epoch1-BasicSeq2Seq/dataset/DataHelper.py

- Text generation from Gaussian noise with 4 tense

```
decoder.eval()
words = []
for i in range(100):

    hidden_mean = torch.randn([1, 1, 32]).to(device)
    hidden_logv = torch.randn([1, 1, 32]).to(device)
    cell_mean = torch.randn([1, 1, 32]).to(device)
    cell_logv = torch.randn([1, 1, 32]).to(device)

    encoder_hidden = reparaterization_trick(hidden_mean, hidden_logv)
    encoder_hidden = decoder.latent2hidden(encoder_hidden)
    encoder_cell = reparaterization_trick(cell_mean, cell_logv)
    encoder_cell = decoder.latent2hidden(encoder_cell)

    tmp = []
    for i in range(4):
        hidden = torch.cat([encoder_hidden, label[i].view(1, 1, 4)], dim=2)
        cell = torch.cat([encoder_cell, label[i].view(1, 1, 4)], dim=2)
        decoded_indices = decoder.evaluate(context_vector=hidden, decoder_cell=cell)
        results = []
        for indices in decoded_indices:
            results.append(train_loader.vocab.indices_to_sequence(indices))
        tmp.append(results[0])
    words.append(tmp)
print(words)
print(Gaussian_score(words))
```

使用 torch.randn()來產生 hidden mean、logv 和 cell mean、logv 的 gaussian noise。

- Hyper-parameters
  - LSTM hidden size: 256
  - Latent size: 32

- KL weight: 0~1
- Teacher forcing ratio: 0~1
- Learning rate: 0.001
- Batch size: 128
- Optimizer: Adam

5. Results and discussion (20%)

A. Results

為了用出最好的結果所以我是用動態調整的方式(調整 KL weight 和 teacher forcing ratio)，所以 Cross Entropy Loss 和 KL loss 才會蠻有起伏的。

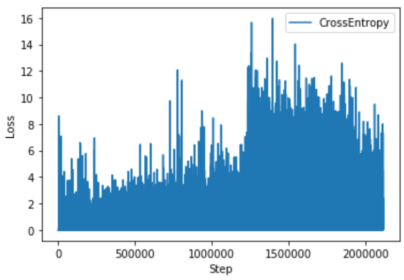
- Results of tense conversion

```
Src_true:      abandon  Trg_true:   abandoned  Predict:      abandon  Score:  0.75148
Src_true:         abet  Trg_true:    abetting  Predict:         abet  Score:  0.36788
Src_true:        begin  Trg_true:      begins  Predict:       begins  Score:  1.00000
Src_true:       expend  Trg_true:     expends  Predict:      expends  Score:  1.00000
Src_true:         sent  Trg_true:       sends  Predict:         ents  Score:  0.14644
Src_true:        split  Trg_true:   splitting  Predict:     spilting  Score:  0.38988
Src_true:       flared  Trg_true:       flare  Predict:       flared  Score:  0.75984
Src_true:  functioning  Trg_true:    function  Predict:     function  Score:  1.00000
Src_true:  functioning  Trg_true:  functioned  Predict:     function  Score:  0.77880
Src_true:      healing  Trg_true:       heals  Predict:        heals  Score:  1.00000
Total score: 0.7194311149760636
```

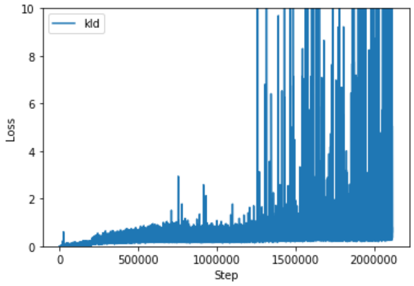
- Results of generation

```
[['repail', 'repails', 'repailing', 'remained'], ['compate', 'compates', 'compening', 'compened'], ['compress', 'compresses',
'competing', 'compressed'], ['compate', 'compates', 'competing', 'competed'], ['compete', 'competes', 'competing', 'compered'],
['disfen', 'disfers', 'disfesting', 'disfested'], ['meature', 'measures', 'meating', 'measures'], ['compate', 'compates', 'comp
ating', 'compated'], ['comber', 'combers', 'compating', 'combered'], ['sunder', 'sunders', 'sundering', 'sunteode'], ['compress
ed', 'compresses', 'compressing', 'compered'], ['sunder', 'sunders', 'sundering', 'sundered'], ['compen', 'competes', 'compenin
g', 'compened'], ['suffer', 'suffers', 'suffering', 'subled'], ['compete', 'competes', 'competing', 'competed'], ['survey', 'su
rveys', 'surveying', 'surveyed'], ['suffer', 'suffers', 'suffering', 'suffered'], ['suffer', 'suffes', 'suffering', 'suffes'],
['poathe', 'poathes', 'poathing', 'poatered'], ['suggest', 'suggests', 'suggesting', 'suggested'], ['suffer', 'suffers', 'suffe
ring', 'suffered'], ['bestore', 'bestows', 'bestoring', 'bestood'], ['obser', 'obsers', 'obsering', 'obserted'], ['serving', 's
erveys', 'serving', 'serveyed'], ['remain', 'remains', 'remaining', 'remained'], ['demand', 'demands', 'demanding', 'suffere
d'], ['siffen', 'surveys', 'suverling', 'suvered'], ['embrace', 'embrace', 'embracing', 'embraced'], ['compete', 'compates', 'c
ompeting', 'compered'], ['compete', 'competes', 'competing', 'competed'], ['comber', 'combers', 'compating', 'combered'], ['rep
air', 'repairs', 'repaying', 'repaired'], ['result', 'results', 'resulting', 'resulted'], ['moverage', 'movers', 'moveraging',
'movered'], ['compate', 'compates', 'compating', 'compated'], ['compete', 'competes', 'competing', 'parodied'], ['prack', 'prec
ames', 'pracking', 'pracked'], ['measure', 'measures', 'measuring', 'measured'], ['combine', 'combers', 'compating', 'combere
d'], ['replie', 'replies', 'repling', 'repained'], ['compen', 'competes', 'competing', 'compened'], ['demand', 'compates', 'com
peting', 'compated'], ['suffer', 'suffers', 'suffering', 'suffered'], ['distan', 'disunites', 'disunying', 'disunited'], ['meas
ure', 'measures', 'measuring', 'measured'], ['snoffer', 'snoffers', 'suffering', 'suffes'], ['repay', 'compates', 'repailing',
'repaired'], ['softens', 'spores', 'sporing', 'spoated'], ['suffer', 'suffers', 'suffering', 'suffered'], ['soften', 'softens',
'softening', 'softened'], ['suffer', 'suffes', 'suffering', 'suffes'], ['suffer', 'suffers', 'suffering', 'suffered'], ['permai
n', 'pobdes', 'permaining', 'pobtened'], ['compressed', 'competes', 'compening', 'compened'], ['mobered', 'moberates', 'moberat
ing', 'mobered'], ['suver', 'suvers', 'suverling', 'suvered'], ['compate', 'compates', 'compating', 'compated'], ['past', 'peas
es', 'peasing', 'pasted'], ['snore', 'sefles', 'suffering', 'suffered'], ['serming', 'sermings', 'serming', 'serming'], ['compa
re', 'compends', 'compending', 'compented'], ['suffer', 'suffers', 'suffering', 'suffered'], ['remake', 'remakes', 'remaking',
'remaked'], ['remake', 'remains', 'remaking', 'remained'], ['remain', 'remains', 'remaining', 'remained'], ['suffer', 'suffer
s', 'suffering', 'suffered'], ['compent', 'compenes', 'compening', 'compened'], ['demand', 'demands', 'demanding', 'softened'],
['comber', 'compates', 'compating', 'combered'], ['repay', 'repails', 'repaying', 'repailed'], ['measter', 'measters', 'meastin
g', 'measted'], ['disate', 'disunges', 'disuniting', 'disunated'], ['seffen', 'seffies', 'seffing', 'seffended'], ['bemoan', 'b
emoans', 'bemoaning', 'bemoane'], ['suffer', 'suffers', 'suffering', 'suffered'], ['soften', 'suffers', 'suffening', 'suffed'],
['demand', 'demands', 'demanding', 'demanded'], ['compete', 'competes', 'competing', 'competed'], ['serfieate', 'serfies', 'ser
fiting', 'serfied'], ['result', 'results', 'resulting', 'resulted'], ['remain', 'ounts', 'remaining', 'ountend'], ['spock', 'sp
orts', 'spocking', 'spokered'], ['result', 'results', 'resulting', 'resulted'], ['serve', 'serves', 'sefuting', 'seboled'], ['s
iffen', 'siffens', 'siffening', 'siffened'], ['onder', 'ondere', 'ondering', 'ondered'], ['ounter', 'ounters', 'ountering', 'ou
ntersank'], ['vank', 'vanks', 'vanking', 'vankered'], ['compate', 'compates', 'compating', 'compated'], ['sefute', 'sefutes',
'sefuting', 'sefuted'], ['suver', 'suvers', 'suvering', 'suvered'], ['pablish', 'parodies', 'pablishing', 'pablished'], ['suve
r', 'suvers', 'suverling', 'suvered'], ['disfeat', 'disfesses', 'disfeating', 'disfested'], ['bolle', 'boller', 'bolling', 'bol
lered'], ['combine', 'counters', 'compating', 'combered'], ['otser', 'otses', 'ottering', 'otsered'], ['repay', 'repairs', 'rep
aying', 'repaided'], ['compare', 'compares', 'compating', 'altered'], ['repair', 'repairs', 'repairing', 'repaired']]
Total Gaussion score: 0.23
```

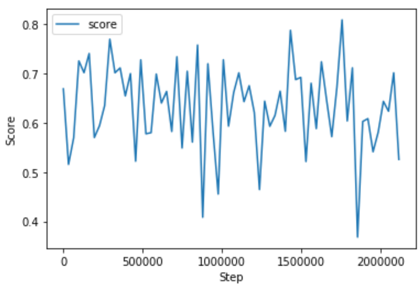
- Crossentropy loss



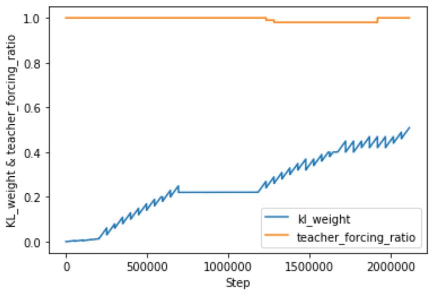
- KL loss



- BLEU-4 score



- KL weight & teacher forcing ratio

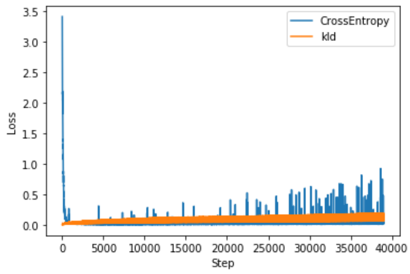
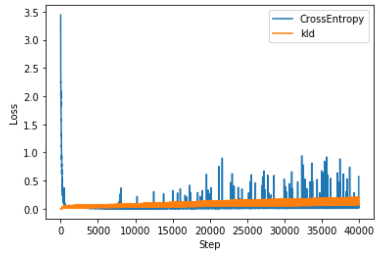
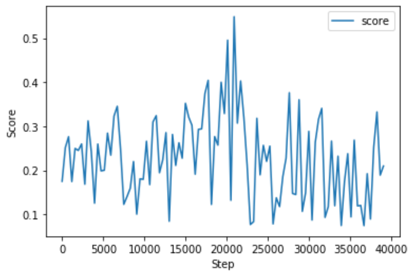
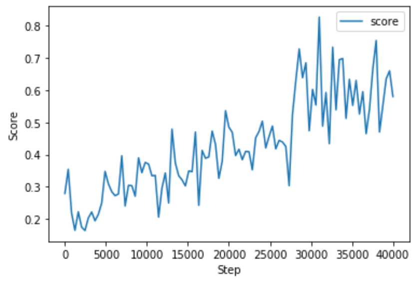


B. Discuss the results

- LSTM compare with GRU

  一開始我是先使用 GRU 來當作練習，把 GRU train 好後再改換到 LSTM 的架構，但是 LSTM 卻一直 train 不好，我覺得是因為 LSTM 多了很多的參數(有 hidden 跟 cell 兩種參數)，不像 GRU 只有少量的參數。

在相同 hyperparameters setting 下的 training 結果比較

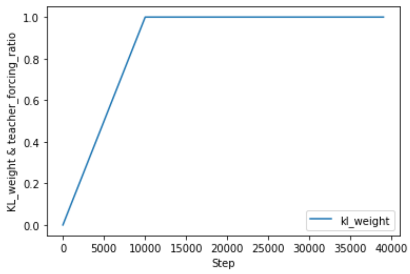
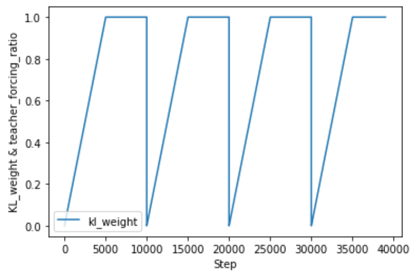
LSTM	GRU
	
Loss	Loss
	
BLEU-4 score (test)	BLEU-4 score (test)

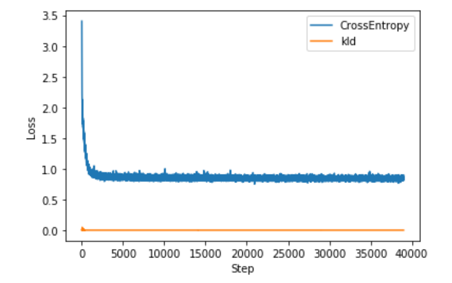
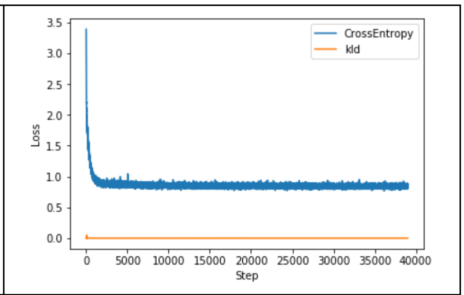
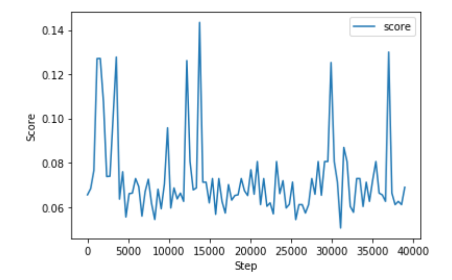
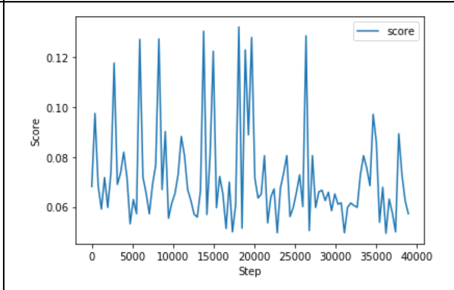
如上面比較的結果來看，LSTM 很難提高它的 testing score，而 GRU 的則可以穩定的上升，上網很多的文獻資料也都說在大部分的情況下，用 GRU 可以得到比 LSTM 更優的結果。

- KL weight

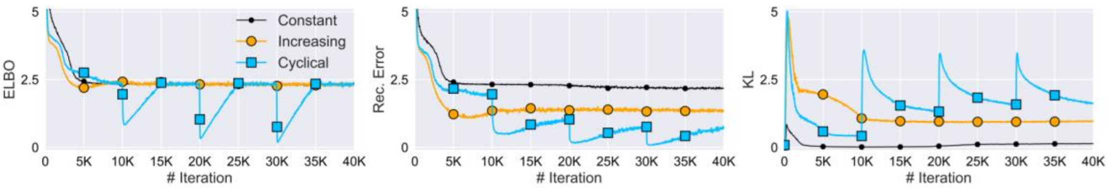
KL weight 是我覺得最難調的其中一個參數，當 KL weight 趨近於 1 時，loss 受到 KL loss term 的影響很大，因此最終會導致 posterior collapse，decoder 不再使用 latent variable z 的機率分布去 generate，因為 z 太多 noise 或是太弱沒甚麼訊息(變成像是 AE 一樣的點)，所以會直接從$q_\phi(z \mid x)$中進行學習。

KL annealing 分成兩種方式，Monotonic 及 Cyclical，接下來將會將這兩種套用在 LSTM 網路架構上進行結果比較於討論。

Monotonic	Cyclical
	
KL weight	KL weight

	
Loss	Loss
	
BLEU-4 score (test)	BLEU-4 score (test)
BLEU-4 score: 0.0688 Gaussian score: 0.1	BLEU-4 score: 0.0572 Gaussian score: 0.88
Testing outcome	Testing outcome

就結果來看 Cyclical 的效能比 Monotonic 還要好，更能夠去克服 Posterior Collapse 得到較高的 Gaussian score，而這個結果跟論文[1]中的也是一樣，如下圖所示：



(a) ELBO (b) Reconstruction Error (c) KL term

它比較了 constant、increasing(也就是 Monotonic)和 Cyclical 這三種 KL annealing 的方式，就結果而言其成效是 Cyclical > Increasing > Constant。

※ Reference

[1] Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing