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。

1. 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 )

1. Derivation of CVAE (5%)

**Derive from the EM altorithm**

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

and we introduce an arbitrary distribution

where,

**Introduce distribution**

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

and the right hand side can be spell out as

Instead of directly maximizing the intractable , we attempt to maximize

which amounts to maximizing

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

1. Derivation of KL Divergence (5%)

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

After taking the logarithm of above we get:

Given the prior p(z) ~ N(0, I) and the posterior approximation , and we calculate the KL divergence:

)

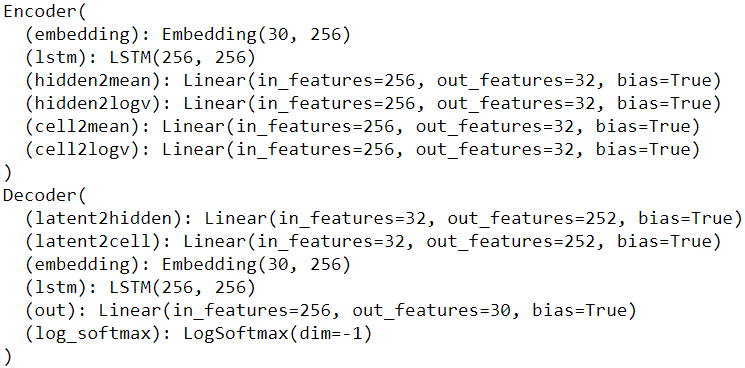
The above equation could be generalized to multivariate cases (n dimensions) by summing over all the dimensions:

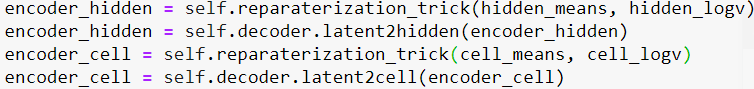
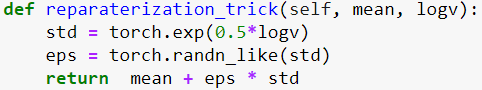
)

So it can be the functions of μθ(x) andΣθ(x), expressed as a closed-form expression.

1. Implementation details (15%)

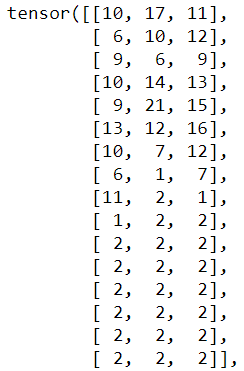
* Model
* Architecture ( encoder & decoder )



* Reparameterization trick  
  
* Dataloader

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

1. split()：將單字拆成一個一個的字母，e.g., alphabet → [a, l, p, h, a, b, e, t]。
2. sequence\_to\_indices()：將單字字母轉換為index。
3. 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。
4. mini\_batches()：最後在取出的時候，將每個字母按照時間序列排序。

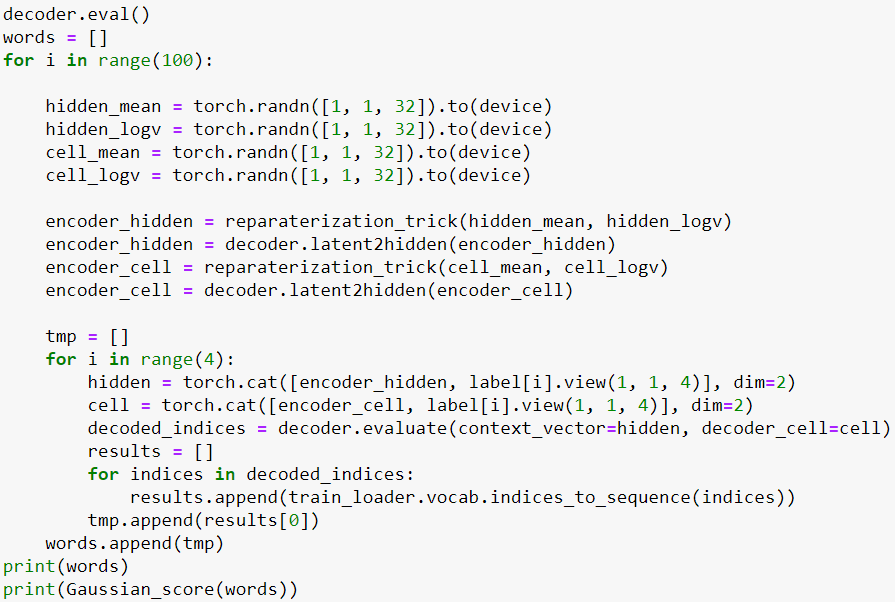
e.g., 

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



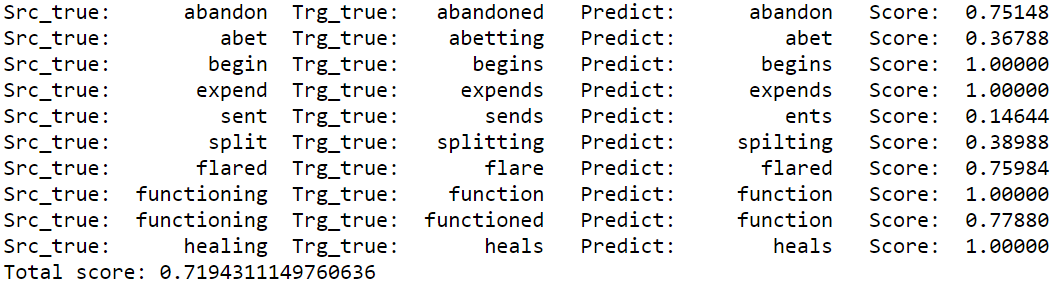
使用torch.randn()來產生hidden mean、logv和cell mean、logv的gaussian noise。

* 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

1. Results and discussion (20%)
   1. 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。

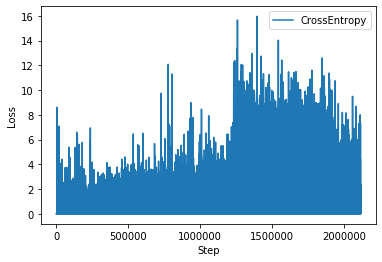
* Results of tense conversion



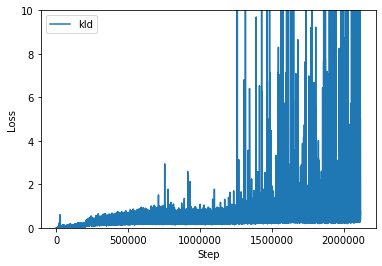
* Results of generation



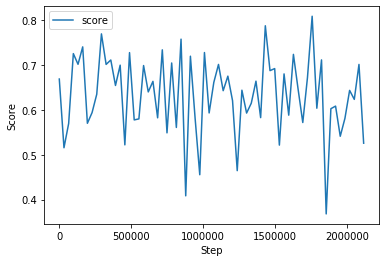
* Crossentropy loss



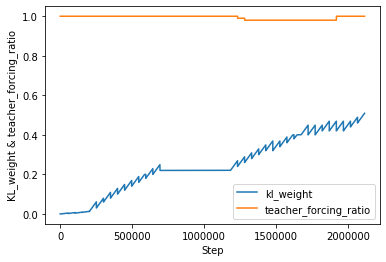
* KL loss



* BLEU-4 score



* KL weight & teacher forcing ratio



* 1. 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 |
| C:\Users\空白\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\75A10D94.tmp | C:\Users\空白\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\7EBE67C8.tmp |
| Loss | Loss |
| C:\Users\空白\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\CCAC8CC2.tmp | C:\Users\空白\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\C4D7C916.tmp |
| 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。

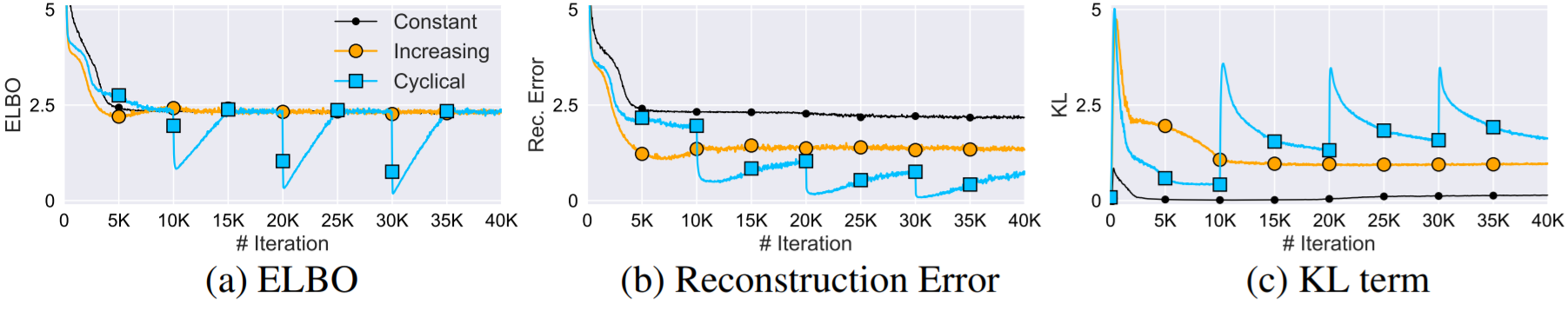
* KL weight

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 |
| C:\Users\空白\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\59080A20.tmp | C:\Users\空白\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\CD9DF778.tmp |
| KL weight | KL weight |
| C:\Users\空白\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\C10B952E.tmp | C:\Users\空白\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B284CE5A.tmp |
| Loss | Loss |
| C:\Users\空白\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\5E6B696C.tmp | C:\Users\空白\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\6C90E446.tmp |
| 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：



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