MLDS 2017 Assignment 4

Problem Description

Open-domain single-turn chatbot

給予模型一句起始的對話句子，模型會產生一個合理且與輸入句子相對應的句子。

Example:

Environment

- OS: Linux
- CPU: Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz, Memory: 64GB
- GPU: GeForce GTX 1080, Memory: 8GBa
- Libraries:
  - Python - 3.5
  - TensorFlow - 1.0
  - NumPy - 1.12.0
  - Progressbar2 - 3.18.1
  - SciPy - 0.19.0
  - Scikit-image - 0.13.0
  - nltk - 3.2.2

Model Description

Data Preprocessing

使用 Marsan-Ma/chat_corpus 以及 cornell movie-dialogs corpus 作為訓練資料，只取長度小於12的句子（包含<BOS>以及<EOS>），並捨棄出現次數小於5次的字。對於縮寫，則將「’」取代成空格，例如說「can’t」會變拆成「can」和「t」兩個字。

Model Structure

- Sequence-to-sequence model (encoder-decoder model)
  - LSTM cell, initial forget bias: 1.0, hidden size: 256, activation: SELU
  - word embedding matrix dimension: 100
  - 1 layer without attention and bi-directional technique
  - 使用的 loss function 是每一個時間點 t 預測的action a_t 和正確答案 $\hat{a}_t$ 所計算的 cross-entropy loss。
- Reinforce policy gradient model
  - 在模型架構上 reinforce policy gradient model 和 sequence-to-sequence model 中的參數以及變數的使用是採用相同的架構。
  - 兩者之間的差異在於 reinforce policy gradient model 的 loss function 是先預測出該輪的 action，接著利用這一輪所預測出來的結果和正確答案去計算每一個位置 t 所得到的 reward r_t，再將這計算出來的每一個時間點 t 的 reward r_t 當作 weight，action 當作 label 做 weighted cross entropy loss，透過這次計算出的結果更新模型中的參數。

- Actor-critic model[1]
  - 使用 Dzmitry Bahanau 提到的 actor-critic model，先 pretrain seq2seq model 以及 critic model，並使用 delay actor 來產生 training actions，使用 delay critic 作為 critic update 的依據。
  - 在每次 update 完 actor 以及 critic 後，會用線性內差的方式 update 兩個 delay model 的 weight。
  - 為了避免因為 action dimension 過大而造成 sample 數量不足的情況，critic 有 penalty term 使得彼此之間的output不會差太多。
  - 我們採用 actor model 和 critic model 共用部份 weights 的方式，而在 actor update 時，屬於 critic 的部份會被 fixed，critic update 時也是同理。

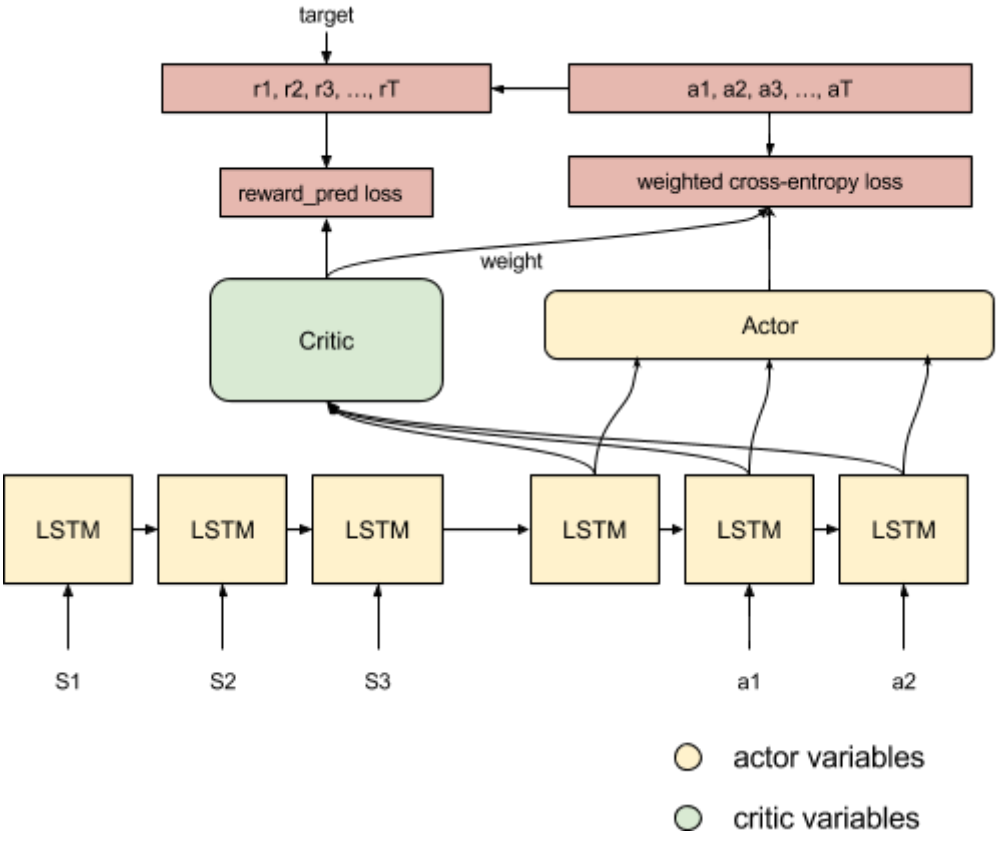

Figure 1: illustrasion of actor-critic model

- Reward function
  - BLEU score、word classification、edit distance
    - BLEU score: 對於一筆資料的正確答案 target 以及隨機產生的 action 算 4-gram bleu BLEU 當作該次 action sequence 的 reward
    - word classification: 對於一筆資料的正確答案 target 以及產生的 action 中，每個 timestep $target_t$ 和 a_t 是否相同，若相同則 r_t 為1，否則為-1
    - edit distance: 對於 target 以及 action 算 edit distance 作為 reward，若做 reward shaping 的話，$r_t = edit\ distance(target_{1...t}, action_{1...t})$
    - 以上三種算法無法判斷某個字 w_i 與另一個字 w_j 是否相似，只能判斷是 w_i 或不是 w_i，以下稱作 non-smoothed reward。
  - word similarity
    - 對於一筆資料的 target 以及 action，我們可以使用embedding maxtrix V 得到兩個 sequence 中，每個 timestep 的 word representation，並計算每個 timestep 的 reward $r_t = cosine\ simularity(V(target_t), V(action_t))$

[1] "An Actor-Critic Algorithm for Sequence Prediction." 。 < https://arxiv.org/abs/1607.07086 >

- 相對於 non-smoothed reward，word similarity 的方法使得即使該 timestep 的字不同，仍有一定的reward，稱作 smoothed reward。我們最後的model採用的reward function為word similarity。

Improvements

SELU activation

一般我們希望作為cell input 的 feature 有經過正規化 (normalized) ，batch normalization 的方法也被很多實驗證實是比較好的，但是 batch normalization 必須增加一些 需要學習的參數。而根據 Günter Klambauer[2] 的作法，透過事前定好的參數，便可有效地將 feature normalize 成指定的 mean 及 variance (SELU)。因此，在這次作業便嘗試了將 LSTM cell 的 tanh activation 換成 SELU activation。

$$SELU(x) = \lambda * \{x,\ if\ x > 0;\ \alpha e^x - \alpha,\ otherwise\}$$

$$\lambda, \alpha = 1.0507009873554802, 1.6732632423543774\ for\ mean,\ var = 0,\ 1$$

Smoothed reward

在 non-smoothed reward 中，不同的字被視為錯誤，但是很多情況即使將字代換也還是通順的。例如：

input: What did she buy?

$output_1$ (target): she bought a cat

$output_2$ (action): she bought a dog

$output_3$: she bought a bought

在這樣的情況下，我們很難從文字上去判斷哪個是正確的，但是若將文字用低維度的 embedding 表示，也許 dog 跟 cat 相似度很高，這樣 cosine similarity 就會接近 1，產生 $output_2$ 的機率便會提高。但在 non-smoothed reward 的情況，reward 無法表現出 $output_2$ 跟 $output_3$ 差異，但很顯然 $output_2$ 比 $output_3$ 好得多。

Reward shaping

根據 Dzmitry Bahanau 的說法，若只採用整個 action 的 reward，因為 sparse signal 的關係，會影響到 training 效率，因此我們將 reward 細分至 timestep-level，使 model 更明確的知道要提升或降低哪些 timestep 的機率。

$$r_t = R(action_{1...t},\ target) - R(action_{1...t-1},\ target)$$

Experiments

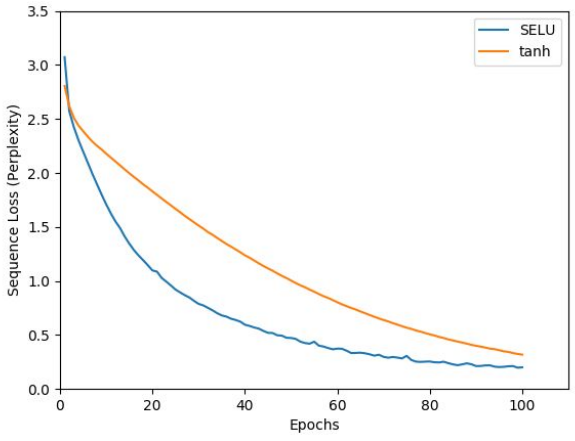

Figure 2.1: different activations

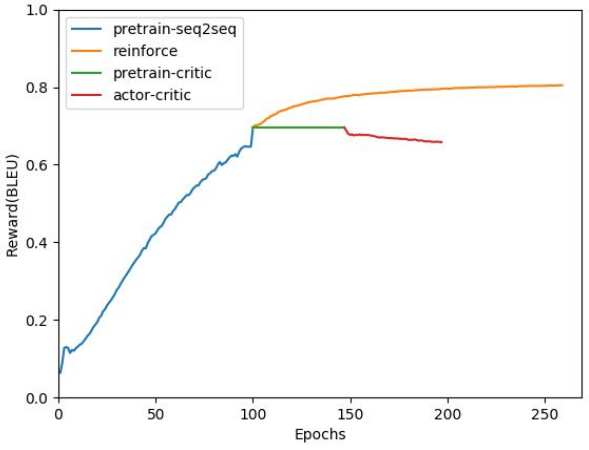

Figure 2.2: different models

[2] "Self-Normalizing Neural Networks." 。9 6月. 2017, <https://arxiv.org/abs/1706.02515>

Observations and Discussions

- **compare different activations**
  - 從 learning curve 來看我們可以明顯的看出，使用 SELU activation function的 training loss 的下降速度比起原先使用的 tanh 還要快上許多，原因可能是 SELU 會將每一筆通過 activation function 的資料做了正規化，這使得在訓練時所計算出的 gradient 較不容易產生忽大忽小的情形，讓模型收斂的速度加快了許多，而不同於 batch normalization，SELU 可以在訓練資料中有 noise 出現時，即便沒有辦法將 noise data 控制在 std = 1 時，SELU 仍然能利用上下界將 layer 的輸出結果控制在一個範圍之內。
- **compare different models**
  - **pretrain-seq2seq:** 隨著 epoch 數越多，performance 穩定成長。
  - **reinforce:** 第100個 epoch 後加入 reinforce policy gradient model，performance緩慢成長。
  - **actor-critic:** 加入 actor-critic 後，performance 反而下降，推測可能的原因為critic 沒有調至適當的參數，導致預測得不夠準確，使得 actor 更新的方向錯誤。
- **disadvantages of word similarity reward**
  - 計算每個 timestep 的 word cosine similarity 可以提高與 target 相似字的機率，但是這種算法並沒有考慮到該相似字與整個句子之間的關係，而且 embedding model 的對象不是我們要最佳化的 score (例如: BLEU)。

Training Details

- LSTM cell hidden size: 256
- word embedding size: 100
- optimizer: AdamOptimizer
- initial learning rate: 1e-3 for actor and critic pretraining, 1e-4 for reinforcement learning
- batch size: 100
- pretrain actor epochs: 100
- pretrain critic epochs: 100

Team Division

組員	分工
江東峻 r05922027	data preprocessing, RL, Seq2Seq, report
陳翰浩 r05922021	experiments, report
鄭嘉文 r05922036	experiments, report

Reference

Bahdanau, Dzmitry, et al. "An actor-critic algorithm for sequence prediction." ***arXiv preprint arXiv:1607.07086*** **(2016).**

Klambauer, Günter, et al. "Self-Normalizing Neural Networks." ***arXiv preprint arXiv:1706.02515*** **(2017).**