CSE-325/425 SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient

Eashan Adhikarla
eaa418@lehigh.edu

Instructor- Prof. Sihong Xie
Date- Oct 16' 2019

Abstract

Due to generator differentiation problem, Generative Adversarial Networks have faced serious issues with updating the policy gradient. Seq-GAN is a unique approach which models the data generator as a stochastic policy in reinforcement learning to solve the problem.

1 Problem Statement

Past RNN (Recurrent Neural Network) training methods have shown some good results such as, provided the previously observed token, maximizing the log predictive probability for each true token in the training sequence. However, this $solution^1$ suffers from an exposure bias issue during inference. This discrepancy between training and inference yielded errors that can accumulate quickly along the generated sequence. In this, they suggested scheduled sampling (SS) to address this issue by including some synthetic data during the phrase of learning. Yet later SS has proved to be an incoherent learning technique to be implemented.

Another approach which was used very popularly is to built the loss function of the entire generated sequence instead of each transition. However, this approach did not show up to the mark results for some of the real life complex examples like music, dialog, etc.

The discriminator is well trained in distinguishing between the actual image and the artificial image even in the Generative Adversarial Network (GAN), but GAN performs poorly with the discrete token values since the guidance is too small to cause a change in the restricted dictionary space. The paper suggests using Reinforcement Learning to train the generator portion of GAN.

2 Network Architecture & Implementation

As we are using the Reinforcement Learning approach, the generator as a whole acts as a policy, previous tokens are the states and action is the next token generated. When there is no intermediate reward, the objective of the generator model (policy) $G_\theta(y_t|Y_{1:t1})$ is to generate a sequence from the start state s_0 to maximize its expected end reward. The architecture shown below shows the generator trained over real data and generated data by G.
The Discriminator chosen is a Convolutional Neural Network (CNN) due to its better performance

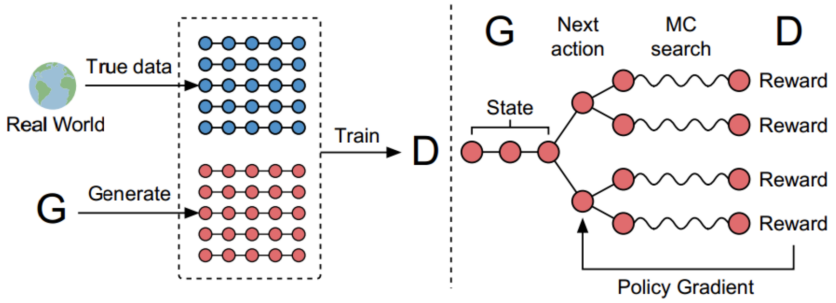

Figure 1: SeqGAN Architecture

in the past sequence classification task.

Generator -

$$J(\theta) = E[R_T|s_0, \theta] = \sum_{y_1 \in Y} G_\theta(y_t|Y_{1:t1}).Q_{D_\phi}^{G_\theta}(s_0, y_1)$$

$$\nabla_\theta J(\theta) = \sum_{t=1}^{T} E_{Y_{1:t-1} \sim G_\theta}[\sum_{y_1 \in Y} \nabla_\theta G(\theta)(y_t|Y_{1:t-1}).Q_{D_\phi}^{G_\theta}(s_0, y_1)] \Rightarrow$$

(gradient of) $J(\theta))$

Discriminator -

$$\varepsilon_{1:T} = x_1 \oplus x_2 \oplus x_3 ... x_T \Rightarrow c_i = \rho(w \otimes \varepsilon_{i:i+l-1} + b)$$

As for the network structure, standard LSTM is employed as the building block of the generator policy. In CNN whose filter size equals to the number of tokens is used as the discriminator and max-over-time pooling is applied after convolution.

3 Baseline & Final Experiments

Following are the baseline models I chose for the experiment;

- **Sequence Sampling:** For sequence-sampling is chosen because they proposed a curriculum learning (ω) approach to slowly change the training objective from an easy task, where the previous token is known, to a realistic one. This has claimed to be better in the sense, previous models have accomplished predicting one token at a time and the prediction depends on the previous correctness.
- **Maximum Likelihood Estimation:** MLE[2] is of fundamental importance in the theory of inference and is a basis of many inferential techniques

4 Dataset

- A randomly initialized LSTM is used to simulate a specific distribution.
- A music data-set contains multiple Nottingham Songs. For music composition, I will use $Nottingham$[2] data-set as my training data, which is a collection of 695 music of folk tunes in midi file format. I plan to study the solo tracks of each music.

5 Final Goals & Testing

- To effectively train generative adversarial nets for structured sequences generation via policy gradient.
- For all the 3 generative models I will keep the kernel size fixed range [1:T] between 100 to 200[3]. Using dropout and L2 regularization for avoiding over-fitting.
- The final goal is decomposed into sub-goals? I foresee to achieve the results in smaller scale, "For epochs around 80-100 **A: Negative log-Likelihood at-most 9.0** should be achievable based on precedent and result **B: NLL below 9.0** is more ambitious. I wish to work toward **B** and report achievement on **A** first."

References

Jaitly Bengio, Vinyals and Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In NIPS, 11711179, 2015.

F. Huszar. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *Neural Computation*, 2015.