SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient

Lantao Yu, Weinan Zhang, Jun Wang, Yong Yu

Association for Advancement of Artificial Intelligence, 2017

Problems Addressed

1. Generative Adversarial Networks have limitations when the goal is for generating sequences of discrete tokens.
2. A major reason lies in that the discrete outputs from the generative model make it difficult to pass the gradient update from the discriminative model to the generative model.
3. The discriminative model can only assess a complete sequence, as balancing the current score of a partially generated sequence and the future score of a fully generated sequence is problematic.
4. Another problem addressed is that task specific scores like BLEU are often not available for specialized applications like poem generation. This too may be alleviated by the Discriminator in GAN architecture.

Proposed Solution

1. The main idea is to model the data generator as a stochastic policy in reinforcement learning (RL).
    - SeqGAN bypasses the generator differentiation problem by directly performing gradient policy update.
    - The RL reward signal comes from the GAN discriminator judged on a complete sequence, and is passed back to the intermediate state-action steps using Monte Carlo search.
    - The state is the generated tokens so far and the action is the next token to be generated.
2. The sequence generation problem is denoted as follows:
    - Given a dataset of real-world structured sequences, train a θ-parameterized generative model G_θ to produce a sequence $Y_{1:T} = (y_1, ..., y_t, ..., y_T), y_t \in \mathcal{Y}$, where $\mathcal{Y}$ is the vocabulary of candidate tokens.
    - In timestep t, the state s is the current produced tokens $(y_1, ..., y_{t-1})$ and the action a is the next token y_t to select.
    - Hence, the policy model $G_\theta(y_t|Y_{1:t-1})$ is stochastic.
    - However, the state transition model is deterministic after a particular action has been chosen i.e. $\delta^a_{s,s'} = 1$ for the next state $s' = Y_{1:t}$ if current state s $= Y_{1:t-1}$ and the action $a = y_t$.
3. A ϕ parametrized discriminative model D_ϕ is also trained. $D_\phi(T_{1:T})$ is a probability indicating how likely a sequence $y_{1:T}$ is from real sequence data.
   A benefit of using the discriminator D_ϕ as a reward function is that it can be dynamically updated to further improve the generative model iteratively.

4. Since there is no intermediate reward, the objective of $G_\theta(y_t|Y_{1:t-1})$ is to generate a sequence from start state s_0 to maximize expected end reward:

$$J(\theta) = \mathbb{E}[R_T|s_0, \theta] = \sum_{y_1 \in \mathcal{Y}} G_\theta(y_1|s_0) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1)$$

Where R_T is the reward for the complete sequence.
$Q_{D_\phi}^{G_\theta}$ is the action-value function of a sequence, i.e. the expected accumulative reward starting from state s, taking action a, and then following policy G_θ.

5. The action-value function is estimated using the REINFORCE algorithm

$$Q_{D_\phi}^{G_\theta}(a = y_T, s = Y_{1:T-1}) = D_\phi(Y_{1:T})$$

6. However, the discriminator only provides a reward value for a finished sequence.

   - Since the longterm reward is important, at every timestep, the resulting future outcome should also be considered.
   - Thus, to evaluate the action-value for an intermediate state, Monte Carlo search with a roll-out policy G_β to sample the unknown last $T - t$ tokens is applied.
     An N time Monte Carlo search is represented as

$$\{Y_{1:T}^1, ..., Y_{1:T}^N\} = MC^{G_\beta}(Y_{1:t}; N)$$

     where $Y_{1:t}^n = (y_1, ..., y_t)$ and $Y_{t+1:T}^n$ is sampled based on the roll-out policy G_β and the current state.
   - For this paper, G_β is the same as G_θ, although other models can be used for better speed.
   - To reduce the variance and get more accurate assessment of the action value, the roll-out policy is run N times, starting from current state till the end of the sequence to get a batch of output samples.

$$Q_{D_\phi}^{G_\theta}(a = y_T, s = Y_{1:T-1}) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} D_\phi(Y_{1:T}^n), (Y_{1:T}^n) \in MC^{G_\beta}(Y_{1:t}; N) & for\ t < T \\ D_\phi(Y_{1:T}^n) & for\ t = T \end{cases}$$

7. Once more realistic sequences start getting generated, the discriminator is re-trained as follows:

$$\min_\phi -\mathbb{E}_{Y \sim p_{data}}[logD_\phi(Y)] - \mathbb{E}_{Y \sim G_\theta}[log(1 - D_\phi(Y))]$$

8. A parametrized policy is to be optimized to maximize long-term reward. The gradient of the objective function $J(\theta)$ w.r.t the generator's parameters θ can be derived as

$$\nabla_\theta J(\theta) = \mathbb{E}_{Y_{1:t} \sim G_\theta}[\sum_{y_t \in \mathcal{Y}} \nabla_\theta G_\theta(y_t|Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t)]$$

9. Using likelihood ratios, an unbiased estimation for the previous equation can be built (for one episode):

$$\begin{aligned} \nabla_\theta J(\theta) &\simeq \frac{1}{T}\sum_{t=1}^{T}\sum_{y_t \in \mathcal{Y}} \nabla_\theta G_\theta(y_t|Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t) \\ &= \frac{1}{T}\sum_{t=1}^{T}\sum_{y_t \in \mathcal{Y}} G_\theta(y_t|Y_{1:t-1}) \nabla_\theta log G_\theta(y_t|Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t) \\ &= \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{y_t \sim G_\theta(y_t|Y_{1:t-1})}[\nabla_\theta log G_\theta(y_t|Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t)] \end{aligned}$$

   where $Y_{1:t-1}$ is the observed intermediate state sampled from G_θ.

10. Since expectation can be approximated by sampling methods, the generator's parameters are updated as

$$\theta \leftarrow \theta + \alpha_h \nabla_{theta} J(\theta)$$

where α_h is the corresponding learning rate. Advanced update rules like RMSProp or Adam can be used here.

11. At the beginning, the generator is pre-trained on training set $\mathcal{S}$ with MLE. After pre-training, generator and discriminator are trained alternately.

12. To reduce variability, different sets of negative samples combined with positive ones are used (similar to bootstrapping).

13. LSTM is used as the generator model in this architecture, while CNN is used as discriminator.

Experiments

1. Synthetic data experiments:

   - A randomly initialized LSTM is used as the true model, aka, the oracle, to generate the real data distribution $p(x_t|x_1, ..., x_{t-1})$ for the experiments.
   - Such an oracle provides the training dataset and also evaluates the exact performance of the generative models, which will not be possible with real data.
   - The parameters of the oracle are initialized following the normal distribution $\mathcal{N}(0, 1)$.
   - The oracle is then used to generate 10,000 sequences of length 20 as the training set $\mathcal{S}$.

2. Real world experiments:

   - For text generation scenarios, the proposed SeqGAN is applied to generate Chinese poems and Barack Obama political speeches.
     - In the poem composition task, a corpus of 16,394 Chinese quatrains, each containing four lines of twenty characters in total, is used.
     - To focus on a fully automatic solution and stay general, no prior knowledge of special structure rules in Chinese poems such as specific phonological rules are used.
     - In the Obama political speech generation task, a corpus consisting of 11,092 paragraphs from Obama's political speeches is used.
   - BLEU score is used as an evaluation metric to measure the similarity degree between the generated texts and the human-created texts.
     - Specifically, for poem evaluation, the n-gram is set to be 2 (BLEU-2) since most words (dependency) in classical Chinese poems consist of one or two characters.
     - For the similar reason, BLEU-3 and BLEU-4 are used to evaluate Obama speech generation performance.
   - In addition to BLEU, poem generation is also chosen as a case for human judgement since a poem is a creative text construction and human evaluation is ideal.
     - Specifically, the 20 real poems and 20 each generated from SeqGAN and MLE are mixed.
     - 70 experts on Chinese poems are invited to judge whether each of the 60 poem is created by human or machines.
     - Once regarded to be real, a sample gets +1 score, otherwise 0.
   - For music composition, Nottingham dataset is used as training data, which is a collection of 695 music of folk tunes in midi file format.
     - The solo track of each music piece is studied. 88 numbers are used to represent 88 pitches, which correspond to the 88 keys on the piano.

- With the pitch sampling for every 0.4s, the midi files are transformed into sequences of numbers from 1 to 88 with the length 32.
- To model the fitness of the discrete piano key patterns, BLEU is used as the evaluation metric.
- To model the fitness of the continuous pitch data patterns, the mean squared error is used for evaluation.

3. The generated samples are given label 0 while the training set samples are given label 1.
4. The kernel size of CNN is from 1 to T and their numbers vary from 100 to 200[4]. Dropout and L2 regularization are used to avoid over-fitting.

Results

Table 1: Sequence generation performance comparison. The p-value is between SeqGAN and the baseline from T-test.

Algorithm	Random	MLE	SS	PG-BLEU	SeqGAN
NLL	10.310	9.038	8.985	8.946	**8.736**
p-value	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	

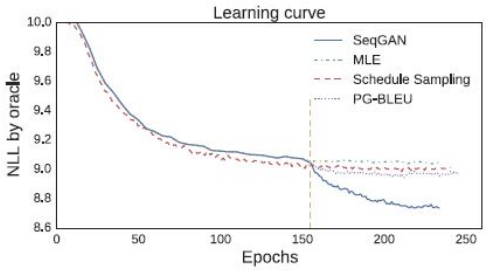
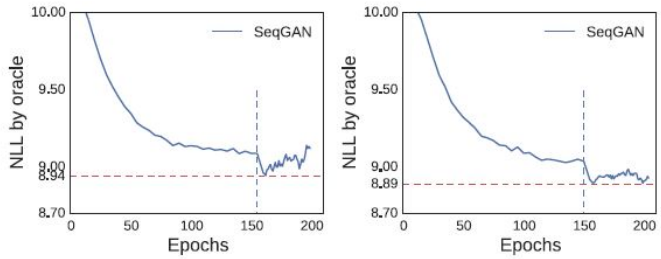
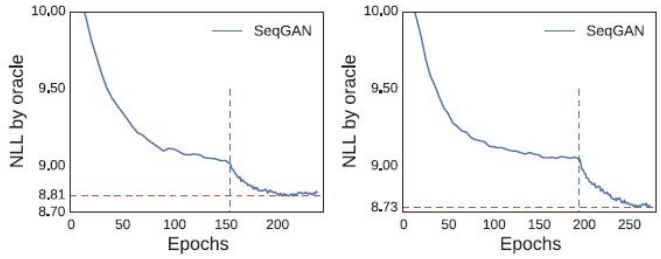


Figure 2: Negative log-likelihood convergence w.r.t. the training epochs. The vertical dashed line represents the end of pre-training for SeqGAN, SS and PG-BLEU.



(a) g-steps=100, d-steps=1, k=10

(b) g-steps=30, d-steps=1, k=30



(c) g-steps=1, d-steps=1, k=10

(d) g-steps=1, d-steps=5, k=3

Figure 3: Negative log-likelihood convergence performance of SeqGAN with different training strategies. The vertical dashed line represents the beginning of adversarial training.

Table 2: Chinese poem generation performance comparison.

Algorithm	Human score	p-value	BLEU-2	p-value
MLE	0.4165	0.0034	0.6670	$< 10^{-6}$
SeqGAN	**0.5356**		**0.7389**	
Real data	0.6011		0.746	

Table 3: Obama political speech generation performance.

Algorithm	BLEU-3	p-value	BLEU-4	p-value
MLE	0.519	$< 10^{-6}$	0.416	0.00014
SeqGAN	**0.556**		**0.427**	

Table 4: Music generation performance comparison.

Algorithm	BLEU-4	p-value	MSE	p-value
MLE	0.9210	$< 10^{-6}$	22.38	0.00034
SeqGAN	**0.9406**		**20.62**	

1. SeqGAN significantly outperforms the maximum likelihood methods, scheduled sampling and PG-BLEU.

2. In three real world tasks, i.e. poem generation, speech language generation and music generation, SeqGAN significantly outperforms the compared baselines in various metrics including human expert judgement.

Bibliography

1. Bachman, P., and Precup, D. 2015. Data generation as sequential decision making. In NIPS, 3249–3257.
2. Bahdanau, D.; Brakel, P.; Xu, K.; et al. 2016. An actor-critic algorithm for sequence prediction. arXiv:1607.07086.
3. Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473.
4. Bengio, Y.; Yao, L.; Alain, G.; and Vincent, P. 2013. Generalized denoising auto-encoders as generative models. In NIPS, 899–907.
5. Bengio, S.; Vinyals, O.; Jaitly, N.; and Shazeer, N. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In NIPS, 1171–1179.
6. Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; et al. 2012. A survey of monte carlo tree search methods. IEEE TCIAIG 4(1):1–43.
7. Cho, K.; Van Merri¨enboer, B.; Gulcehre, C.; et al. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. EMNLP.
8. Denton, E. L.; Chintala, S.; Fergus, R.; et al. 2015. Deep generative image models using a laplacian pyramid of adversarial networks. In NIPS, 1486–1494.
9. Glynn, P. W. 1990. Likelihood ratio gradient estimation for stochastic systems. Communications of the ACM 33(10):75–84.
10. Goodfellow, I., et al. 2014. Generative adversarial nets. In NIPS, 2672–2680.
11. Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. Deep learning. 2015.
12. Goodfellow, I. 2016. Generative adversarial networks for text. http://goo.gl/Wg9DR7.
13. Graves, A. 2013. Generating sequences with recurrent neural networks. arXiv:1308.0850.
14. He, J.; Zhou, M.; and Jiang, L. 2012. Generating chinese classical poems with statistical machine translation models. In AAAI.
15. Hingston, P. 2009. A turing test for computer game bots. IEEE TCIAIG 1(3):169–186.
16. Hinton, G. E.; Osindero, S.; and Teh, Y.-W. 2006. A fast learning algorithm for deep belief nets. Neural computation 18(7):1527–1554.
17. Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. Neural computation 9(8):1735–1780.
18. Husz´ar, F. 2015. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? arXiv:1511.05101.
19. Kim, Y. 2014. Convolutional neural networks for sentence classification. arXiv:1408.5882.
20. Kingma, D. P., and Welling, M. 2014. Auto-encoding variational bayes. ICLR.
21. Lai, S.; Xu, L.; Liu, K.; and Zhao, J. 2015. Recurrent convolutional neural networks for text classification. In AAAI, 2267–2273.
22. Manaris, B.; Roos, P.; Machado, P.; et al. 2007. A corpus-based hybrid approach to music analysis and composition. In NCAI, volume 22, 839.
23. Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In ACL, 311–318.
24. Quinlan, J. R. 1996. Bagging, boosting, and c4. 5. In AAAI/IAAI, Vol. 1, 725–730.

25. Salakhutdinov, R. 2009. Learning deep generative models. Ph.D. Dissertation, University of Toronto.

26. Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; et al. 2016. Mastering the game of go with deep neural networks and tree search. Nature 529(7587):484–489.

27. Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. JMLR 15(1):1929–1958.

28. Srivastava, R. K.; Greff, K.; and Schmidhuber, J. 2015. Highway networks. arXiv:1505.00387.

29. Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In NIPS, 3104–3112.

30. Sutton, R. S.; McAllester, D. A.; Singh, S. P.; Mansour, Y.; et al. 1999. Policy gradient methods for reinforcement learning with function approximation. In NIPS, 1057–1063.

31. Vesel'y, K.; Ghoshal, A.; Burget, L.; and Povey, D. 2013. Sequence discriminative training of deep neural networks. In INTERSPEECH, 2345–2349.

32. Wen, T.-H.; Gasic, M.; Mrksic, N.; Su, P.-H.; Vandyke, D.; and Young, S. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. arXiv:1508.01745.

33. Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning 8(3-4):229–256.

34. Yi, X.; Li, R.; and Sun, M. 2016. Generating chinese classical poems with RNN encoder-decoder. arXiv:1604.01537.

35. Zhang, X., and Lapata, M. 2014. Chinese poetry generation with recurrent neural networks. In EMNLP, 670–680.

36. Zhang, X., and LeCun, Y. 2015. Text understanding from scratch.arXiv:1502.01710.