Adversarial Generation of Natural Language

1 Main Idea

The paper tries to leverage the success of GANs in the computer vision domain to generate language. The authors attempt to address the discrete space problem without the usage of gradient estimators. They also intend to generate language based on context-free grammars and also from conditional generation using sentence attributes. [1]

2 Method

- Generative Adversarial Networks [2] are comprised of a generator that tries to mimic the original distribution $P(x)$ by sampling from Gaussian noise $P(z)$ using a generator function $G(z)$, and a discriminator that determines whether samples from 2 distributions $P(x)$ and $G(z)$ are genuine or fake.
- Most language models need to be trained using teacher-forcing, which ensures that the output at the last time-step of the prediction sequences is one of the inputs to the new timestep, which corresponds to MLE training of the model i.e. during training utilize the actual ground-truth of the previous timestep as an additional information signal for the present timestep. The paper also mentions the need to eliminate exposure bias, which is caused when a bad prediction early in the sequence has a cascading effect on the overall quality of the sequence.
- The paper comments on the the lipschitz constraint from the Wasserstein GAN paper [3] as an important finding. This constraint restricts the weights of the discriminator network such that they lie in a fixed interval, and that the discriminator is trained multiple times per generator training epoch. This is supposed to protect against allowing the discriminator to easily distinguish between the one-hot vector encoded true-data distribution and the dense real-valued predictions. WGANs are supposed to provide better gradients to the generator because the lipschitz constraint prevents saturation.

- Experiments are performed on both recurrent as well as convolutional models, as well as using curriculum learning
- Recurrent Model:
  - The model architecture itself doesn't seem novel from the point of view of a typical GAN. Teacher-forcing is used at each time step to condition the output of the current timestep on the ground-truth of the last timestep. The paper notes that in a vanilla RNN, the encoding phase requires the current time step to be conditioned on information from the previous timesteps contained in the hidden state, but this doesn't hold true for the generating (decoding) phase, which is a problem teacher-forcing tries to address.
  - There is a slight analogy between teacher-forcing and attention, in that teacher forcing peeks at the ground truth from the previous time-step and attention mechanisms peek at different hidden states from the encoding phase.
  - Greedy decoding is performed to predict the next character/word.
- Convolutional Model:
  - The convolutional model comprises of 5 residual blocks with 1D convolutional layers. A residual block involves 2 convolutional operations, followed by adding the original input to the output of the 2^{nd} convolutional layer.
  - No pooling or dropout is used. Regularization is done using batch-normalization layers.
- Curriculum Learning: As opposed to the timestep by timestep prediction techniques of the previous 2 methods, curriculum learning is used to predict entire sequences. [4]

3 Observations

-

References

[1] Sandeep Subramanian, Sai Rajeswar, Francis Dutil, Christopher Pal, and Aaron Courville. Adversarial generation of natural language. *ACL 2017*, page 241, 2017.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.