Generate Chinese poems with WGAN

Xinyun Chen[†] **Zhifeng Zhao**[†] **Zhenyu Mao**[†]
Shanghai University of Finance and Economics

Abstract

We apply GANs on natural language processing area to generate Chinese poems that somehow make sense. We refer to the WGAN and use the gradient penalty method to make it much effective than traditional GANs on NLP. We have also design our unique models for poems generating task. The result remains to be improve but the framework shows the potential of GANs in not only images but also language models.

1 Introduction

Generative adversarial nets (Goodfellow et al., 2014) have been popular in Deep Learning for its advantages of generating better samples and no need of markov chains or other intractable inference. It has been applied in many fields and has made great achievements. However, GANs hardly works on natrual language processing. Goodfellow, the "father of GANs"said that the poor effect of GANs in NLP is because that GANs is not suitable for discrete data space. But with some methods put forward to improve GANs, it becomes possible and appealing to apply GANs on NLP. There has been some work using GANs in the field of NLP such as neural dialogue generation(Li et al., 2017) and got good result. Combining GANs and Reinforcement Learning seems make better effect than the standard SEQ2SEQ model and mutual information model(Li et al,. 2016).
So that we will refer to these improved methods on GANs to experiment on a specific text generating task: Chinese poems. We choose this objective is because that most poems share the similar restricts on rhythm, tone, and some other latent critics. So we treat all these as abstract features and we expect that GANs would exploit these features itself without prior information. Particularly, we want to make people not able to distinguish the poems generated by the machine from the poems chanted by poets like Li Bai and Zhang Ruoxu hundreds of years ago, which is similar to the idea of the Turing test(Turing,1950). This task is nodoubtly a tough, and of course an attractive one.

2 Background

Generative Adversarial Nets(GANs) are a powerful class of generative models which include two nets: generator and discriminator. The Generator tries to produce fake data from a Gaussian noise while the discriminator tries to discriminates while a data is produced by generator or is from real data. This framework corresponds to minimax two-player game, and the discriminator and generator are trained simultaneous. Finally, discriminator and generator reaches the point of nash equilibrium where generator recovers the data distribution and discriminator gives 1/2 everywhere.
While GANs have many difficulties, they are still appealing for that they can produce samples with high quality. But typical GANs are incapable of discrete data such as text, for that the training process is to change the output according to the gradient given by discriminator, which is meaningless to text. Some structure like SeqGAN(Yu et al., 2016) combines GANs and reinforcement learning, they treat the output of discriminator as reward and optimize the generator via policy gradient to handle the problems of discrete data space. And they also use the MCMC method to get result of a partial sequence. But these methods seem to be unwieldy because of the introducing of reinforcement learning.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

In particular, (Arjovsky and Bottou) analysis the difficulty in convergence of typical GANs and provide the method to compute Wasserstein distance instead of Jason Shannon divergence and give the condition to work on discrete data space. This method is a rather simple one with not simple theory behind. It just changes a little on original GANs but makes dramatic progress on the whole frameworks. We will discuss the theory of this method in the next section and explain the relation with our poem generation task.

3 Wasserstein distance and WGAN

3.1 Wasserstein distance

The Wasserstein distance of two distribution (p, q) is defined as the minimum cost of transporting mass in order to transform the distribution p into q. This distance enjoys a superiority to JS divergence. When the measure of the overlap of the two distributions can be ignored, JS divergence makes little sense while Wasserstein distance could still reflect the distance of them. However, it is hard to compute directly for its computing method is described as follows:

$$W(p, q) = \inf_{\gamma \in \Pi(p,q)} E_{(x,y)\sim\gamma}[\|x - y\|] \tag{1}$$

This has an equivalent transformation according to (Arjovsky et al., 2017):

$$W(p, q) = \frac{1}{K} \sup_{\|f\|_L \leq K} E_{x\sim p}[f(x)] - E_{x\sim q}[f(x)] \tag{2}$$

And it can be approximate by the formula bellows:

$$W(p, q) \simeq \frac{1}{K} \max_{\|f_w\|_L \leq K} E_{x\sim p}[f_w(x)] - E_{x\sim q}[f_w(x)] \tag{3}$$

Here f_w is a differential function with respect to K-Lipschitz constraint. So that we could use a neural network and to approximate this term through optimizing methods. If we treat f_w as the discriminator, it will approximate the Wasserstein distance between the real data distribution and the generated data distribution, and the generator could be optimized through the gradient to minimize the Wasserstein distance. Since the task of the discriminator has changed, it should be named as a critic. And this is all about WGAN. From the view of the result, it just eliminates the sigmoid activition and log function in original objective and cuts the weight to a fixed value to satisfy the Lipschitz constraint.
Now that the framework is capable to cope with discrete data like text for that the Wasserstein distance could give useful information even between a one-hot vector and a probability distribution. So we could use it to give much helpful instructions to the generator.

3.2 WGAN with gradient penalty

There is still some problems need to be considered for the WGAN. To satisfy the Lipschitz constraint, the weight of the critic is clipped to a fixed value to force the output changes in a compact space. The weight-clipping method may leads to the unstable of training process and sometimes even causes gradient vanishing. To solve this problem, (Gulrajani et al.) proposed gradient penalty method instead of weight clipping to enforce the Lipschitz constraint on the objective. Particularly, the new objective for the critic is:

$$L = E_{x\sim p_d}[C(x)] - E_{x\sim p_g}[C(x)] + \lambda E_{x\sim p_{\hat{x}}}[(\|\nabla_{\hat{x}}\|_2 - 1)^2] \tag{4}$$

They use a regularization term to penalty the weights with rather big values. Here $p_{\hat{x}}$ are the random samples between the real samples and generated ones. We can see the differences between these two methods. In the weight-clipping methods, almost all of the weights gather to the boundary while the weights under gradient penalty methods enjoys a better distribution. Also, it is less likely to suffer from gradients exploding or vanishing.

Now the WGAN with Gradient Penalty is able to be trained with improved stability. We will apply this idea as the overall architecture of our Chinese poem generation task. Our Algorithm will have several hyper parameters like λ, the critic number n_c that the critic updates for n_c times while the generator update once. And of course the embedding dimension d for all the models. In our experiments, we choose $\lambda = 10$, $n_c = 5$, $d = 50$.

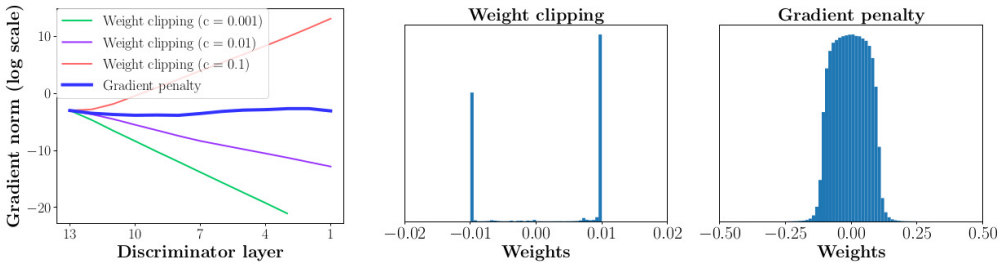


Figure 1: Comparison between weight-clipping and gradient penalty

4 Experiment

In this section we will introduce the specific models of our generator and critic. For the critic, we have designed two different frameworks and for the generator, we also applied two methods. And we will estimate the result under different choices of frameworks.

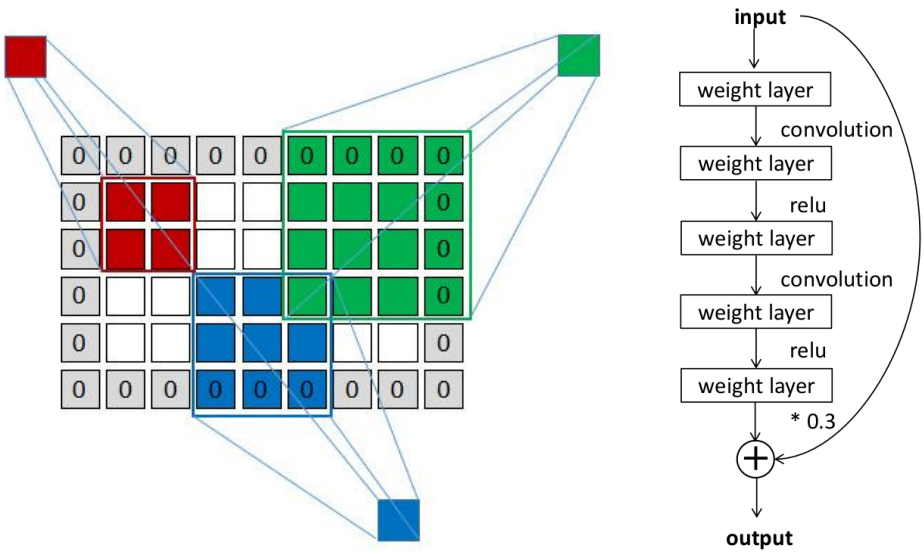
4.1 Data preprocess

We collected 30 thousand poems from the Internet, but the length of these sentences are different. For simplicity, we just choose poems with length 5 and length 7 to test our deep neural network model. We also eliminate the poems with more uncommon words. This is because that when we train the GANs, the generator is tend to generate a 'safe' sample, and uncommon words have rather less impact on the final feedback. So the uncommon words will become more common than they should have been. And finally we have 3242 poems with 1924 words. We use one-hot encoding to describe every characters, so each poem can be transformed into a tensor with shape (4,7,1924) .

4.2 Critic

Since our objective is to generate Chinese poems, we expect that the critic can give the information of how likely the sequence of characters to be a real poem. So we designed the critic to extract the inherent features of a poem. We treat the 7-quatrains as a 4*7 picture with 1928 channels, and this step is the same for both of the frameworks. For the first method, we used several convolutional layers with different sizes of kernels to extract features at different scales. We expect the critic get sufficient information from the poems to compute the Wasserstein distance between a real poem and a fake output. In particular, the input poems are first embedding in a decided dimension to lower the expenses. As Figure 2(a) shows, the filters of size 2*2,3*3 and 4*4 are all retained and applied on the embedding output through locally-connected layer. We use locally-connected instead of convolutional layer because we think that for different positions in the poem, the metric should be different. Finally, we merge these output from locally-connected transformation together and put them into a dense classifier with one final output.

For another method to measure the poem, we introduce the residual networks(He et al.) to build a rather deep net to give consideration to both abstract, global features and explicit, local features within the poems. In particular, a residual block includes two convolution layer and two activation layer of relu(Glorot et al.), and the output consists of the origin input and the transformed output with a preset ratio of summation. It should be mentioned that to sum up the output with the origin input, the shape of these two tensors must be the same. So the convolution process throughout the residual block uses same padding to ensure this. Figure 2(b) shows the whole frame of a residual block. We repeat the residual block for 5 times to build a rather deep networks. Then we put the final output which includes hybrid information of features into a dense classifier like we did in the first method.



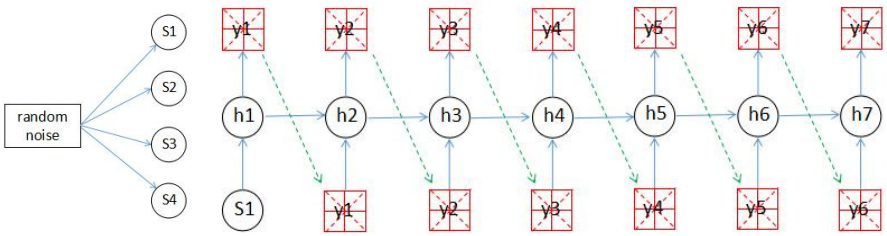
(a) Convolution filters (b) Residual block

Figure 2: Critic frameworks

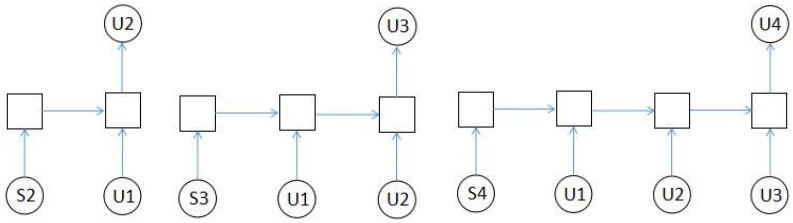
4.3 Generator

The generator in the original GANs receives a random noise as input so that the process of learning is without supervision. Since our poems is with no additional information about the features within such as theme, style, rhyme, we expect that GANs could find these inner correlations itself. Therefore, our generator is just a model to transform a random noise of fixed dimension of a sequence of probability distributions and each stands for a specific character. And for the transforming process, we tried two frameworks, one based on RNN method, and the other on residual network just like the critic.
RNN method is natural for its effectiveness on natural language processing, but here we need to make a little difference. Since that RNN predicts the output according to the previous state, but during the training process we need to input the whole sequence, which does not make sense in our generator because we don't want all the input are from random noise. (Wang et al.) provides an approach to generate Chinese poems via RNN. The generating of next character is based on previous characters and assigned sub-theme for the verse. We refer to their method and treat the original input as the theme of the whole poem and generate sub-theme for every verse. As Figure 3 shows, we generate the first character y_1 from the sub-theme s_1, and y_2 from s_1 and y_1, y_3 from s_1, y_1 and y_2, etc. So we used 7 LSTM cells in total for the generation of the first verse. And then we treat the entire verse as a unit u_1 and generate the rest directly from it to simplify the model as well as enhance relations among verses. The process is as follows:

The residual network is generally the same as the critic except the shape of input and output. We first put the original random noise into a dense layer to fit the shape of expected poem, except that the number of channels is much smaller. Then we apply the residual network on it to enhance the capability of the model. We add an additional convolutional layer with softmax activation to get the final output of probability distribution. This generating method is not a plain one for it generates all the charactors at the same time. It is different from our regular manner of speaking, but it does make sense to computer since it may get more information before speaking any word.



(a) Outlines (b) LSTM for the first verse



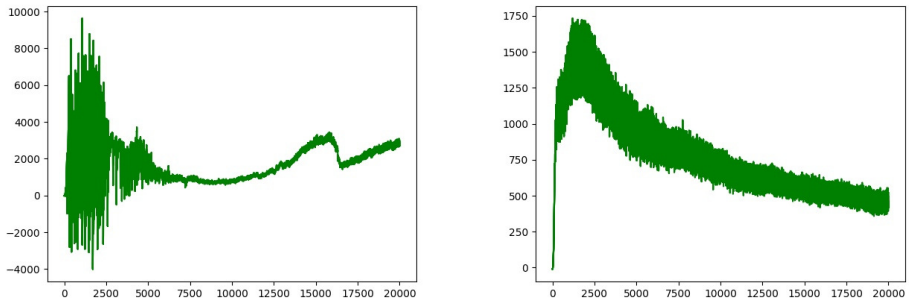
(c) Second (d) Third (e) Fourth

Figure 3: LSTM Generator

5 Result

5.1 Training stability under different frames

Although the theory of WGAN and WGAN-GP are persuasive and reasonable, our practical training process shows that the stability still exists in all the frameworks above. We speculate that WGAN with GP method is still sensitive about the value of initial parameters. In the condition of choosing big λ and n_c with small d, the gradient of critic even explodes. Also, the convergence of these models do not perform as well as we expected. We have run each framework for at least 5k iterations, but none of them produces an acceptable result. Here we give the negative losses of critic, which stands for the Wasserstein distance of the data and the model:



(a) Residual (b) LSTM with Convolutional

Figure 4: Wasserstein distance after 20k iterations

The generation under LSTM cells and critic under convolution methods are stable, but the deep structure of residual networks are vibrating acutely from the very beginning.

5.2 Generated poems

Overall, the residual network performs much better in both generator and critic. We consider that the information and features within the poems is largely depend on the choice of characters directly. LSTM cells are stable during training, but it is with high computational cost because of the rather complicated framework and also hard on convergence.
Throughout the process of the experiment, the best results we could get are as follows:

服州盏量韩化坛，诚印切鲁韶彩虚。
返敛施狼依困损，空值童拜冯吐潺。

宋禊折偏续速假，遇知代薄二渔滞。
文甚至妙斗登意，切异郎恶弯茶案。

This is the generation under residual networks on both generator and critic, after 10k iterations. We can see that the global features, or the theme of the poem are explicit and some combinations of adjacent words are somehow making sense, such like 'zhimiao' and 'kongzhi'. But as an entire verse, it is barely idiomatic, let alone as an entire poem. We take it for that the model does not completely converge. It just learns some of the features independently.

6 Future improvements

As the defective of the whole models to generate Chinese poems we have talked about, the framework and the parameters, even the method of measuring the Wasserstein distance remains to be perfected. We post two schemes which may be helpful to our models.
The first scheme is to put a prior distribution on the input instead of totally random noise. Generative models like VAEs(Kingma and Welling, 2013) have produce methods to generate sentences from a continuous space for it learns codes not as single points, but as soft ellipsoidal regions in latent space and forces the codes to fill the space rather than memorizing the training data as isolated codes.(Bowman et al.) So that the generation process is under supervision, which will impose restrictions on the capability of GANs but beneficial to our poems generating task. Another scheme is to redesign the architecture of generator and critic. Since we observe that the features are not accurately caught by the model, we should eliminate some global features and take more account of features with medium sizes to prompt the smooth and idiomatic of the verse, and the poem.

Acknowledgment

Thanks to Qi Deng, Hui Fang, and Zhipeng Fan for their suggestion of GAN for NLP,and their valuable comments and suggestions throughout the whole process.

References

[1]Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza,Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron-Courville, and Yoshua Bengio.(2014) Generative adversarial nets. In Advances in Neural Information Processing Systems. pages 2672–2680.

[2]Alan M Turing. (1950) Computing machinery and intelligence.Mind **59**(236):433–460.

[3]Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter and Dan Jurafsky.(2017)Adversarial Learning for Neural Dialogue Generation.

[4]Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao and Dan Jurafsky. EMNLP.(2016)Deep Reinforcement Learning for Dialogue Generation.

[5]Arjovsky M, Bottou L (Towards Principled Methods for Training Generative Adversarial Networks.). Arjovsky M, Chintala S, Bottou L (Wasserstein GAN.2017).

[6]Bowman SR, Vilnis L, Vinyals O, Dai AM, Jozefowicz R (Generating Sentences from a Continuous Space.).

[7]Glorot X, Bordes A, Bengio Y (Deep Sparse Rectifier Neural Networks.).

[8]Goodfellow IJ, Pouget-Abadie J, Mirza M (Generative Adversarial Nets.).

[9]Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville A (Improved Training of Wasserstein GANs.).

[10]He K, Zhang X, Ren S, Sun J (Deep Residual Learning for Image Recognition.).

[11]Kingma DP, Welling M (Auto-Encoding Variational Bayes.2013).

[12]Wang Z, He W, Wu H, Wu H, Li W (Chinese Poetry Generation with Planning based Neural Network.).

[13]Yu L, Zhang W, Wang J, Yu Y (SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient.2016).