Generative Adversarial Network

Hamid Karimi

May 10, 2017

Michigan State University
Department of Computer Science and Engineering
428 South Shaw Lane
East Lansing, MI 48824

Loss function.
Equations 2 and 3 show the loss functions of generator and discriminator, respectively.

$$L_D = \frac{1}{M} \times \sum_k \Big[\log P(x_k) + \log\big(1 - P(G(z_k))\big) \Big] \tag{2}$$

$$L_G = \frac{1}{M} \times \sum_k \log P(G(z_k)) \tag{3}$$

M is batch size, $1 \leqslant k \leqslant M$ is index of batch samples, x_k is real input sample, z_k is latent variable(s), $G(z_k)$ is the fake sample generated by generator from latent variable(s) z_k, and $P(X)$ returns likelihood of sample X (either fake or real) belonging to real distribution.

Model design.
You can find network parameters and other settings in Table 1. I set number of hidden units GAN twice big as VAE's. This is because GAN was converging slower.

Training and balance strategy.
At each iteration, *percentage* of improvement of generator and discriminator loss function are calculated. Then, if their absolute difference is less a threshold $(0, 01)$, both of them are selected for update. Otherwise, the one which have got less improvement will be selected and the other one won't. Also, when it is discriminators turn, it is optimized k times (here 5). Because discriminator is more likely to be left behind, I have given it more chance to keep up. Note that I am calculating percentage of improvement since last optimization. So, threshold 0.01 corresponds to 1%.

Remark 1: at each iteration, discriminator network is called twice: once for real sample and once for generated (i.e., fake) sample. In the second call, *reuse* option is true to tell tensorflow to use the same variables and not raise any error.

Remark 2: discriminator and generator are optimized separately via passing their own variables to the optimizer.

Results.
Figure 1 shows visualizations of original distribution and generated one over different epochs. In epoch 401, they overlap each other quite well. Figure 2 plots evolution of generated Swiss-roll at different epochs. As you can see, the shape is getting more similar to real Swiss-roll distribution. Figure 3 shows discriminator, generator and total loss of GAN vs training iteration and how they stabilize.

VAE vs. GAN
For this problem (i.e., generating Swiss-roll), VAE achieved better performance. VAE was faster and quality of its generated distributions was higher.

Observations

- An interesting phenomenon observed was GAN produces more samples from those areas of a distribution which enjoy more density. This agrees with findings of [**?**], GAN's original paper.

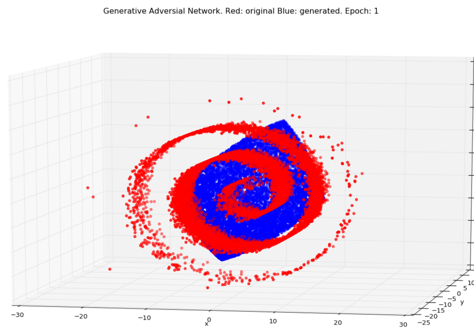
- As the optimization proceeds, $P(G(z))$ becomes closer to $\frac{1}{2}$. $G(z)$ is a fake sample generated by generator. In another words, discriminator is fooled and is left with no choice other than the pure random guess of $\frac{1}{2}$. Figure 4 shows $P(G(z))$ vs iterations. As you can see, it is quite close to $\frac{1}{2}$. This is also in agreement with analytical proof shown in [**?**].

Discriminator	Layer1	200 units. Nonlinearity : tanh
	Layer2	200 units. Nonlinearity : tanh
	Layer3	1 unit. Linear (yielding $P(x \in real \vert x)$ or $P(z \in real \vert z)$)
Generator	Layer1	200 units. Nonlinearity : tanh
	Layer2	200 units. Nonlinearity : tanh
	Layer3	3 units. Linear (generating Swiss-roll fake points)
z dimension	2	
Batch size	100	
Learning rate	0.001	
Optimizer	AdamOptimizer	
Number of Swiss-roll points	10000	
Number of trained epochs	1000	
Convergence epoch	400	
Balance threshold	0.01	
k	5	

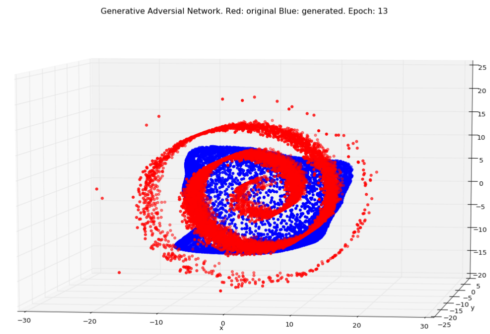
Table 1: GAN network architecture and settings

Question 3.

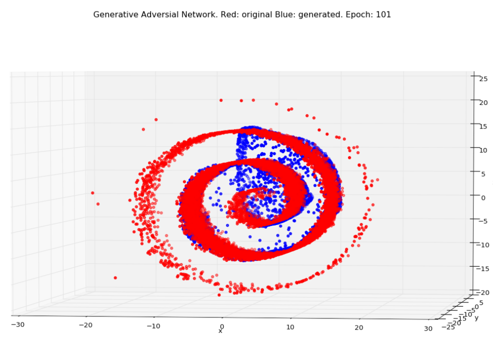
I was involved with this homework for 3 days. I hope homework#4 and project compensate my shortcomings in the previous assignments and help to boost my grade.



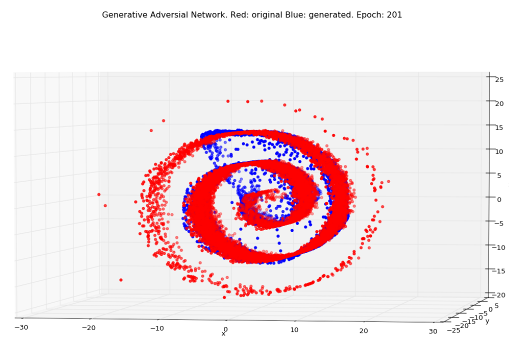
(a) epoch 1



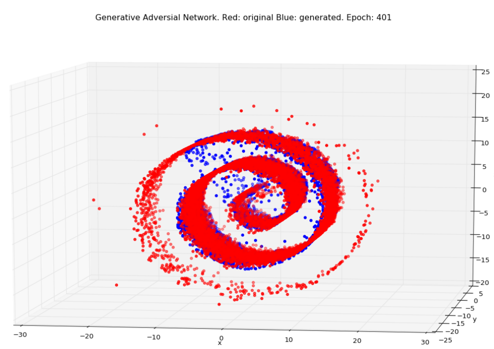
(b) epoch 13



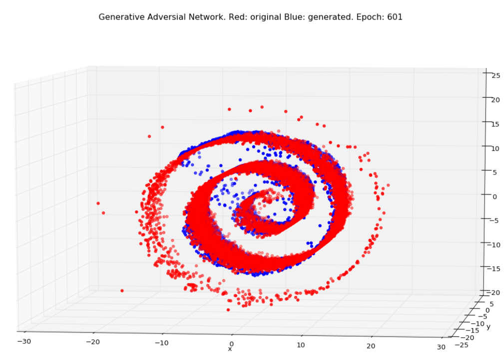
(c) epoch 101



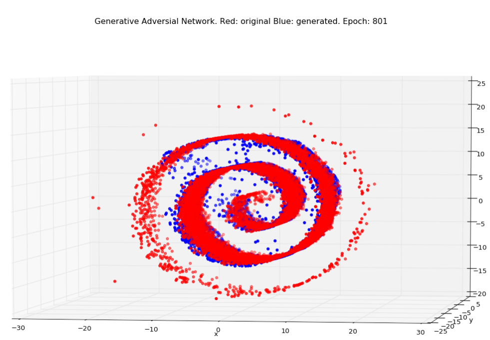
(d) epoch 201



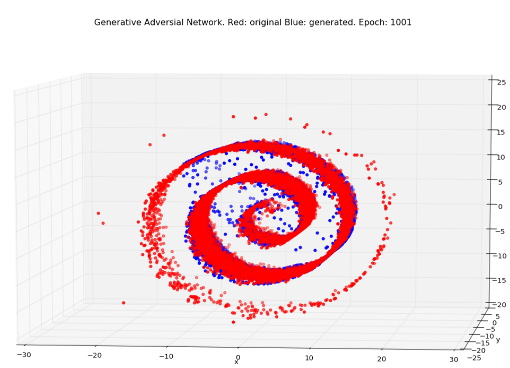
(e) epoch 401



(f) epoch 601

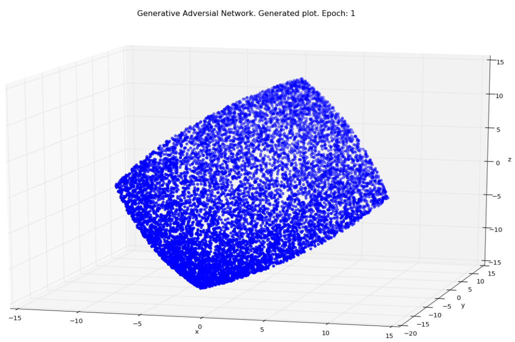


(g) epoch 801

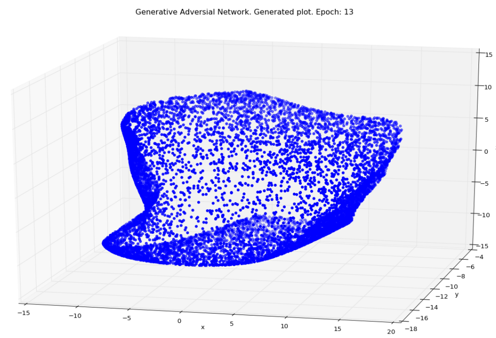


(h) epoch 1001

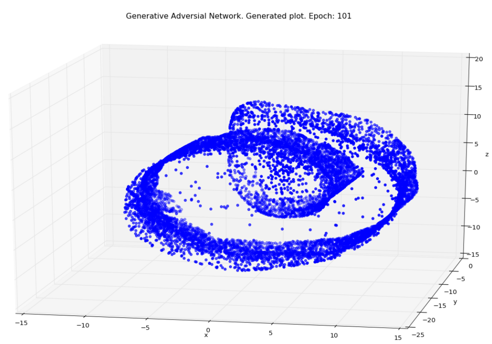
Figure 1: Generative Adversarial Network . red: original blue: generated



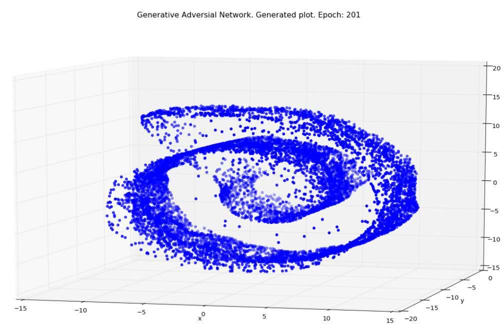
(a) epoch 1



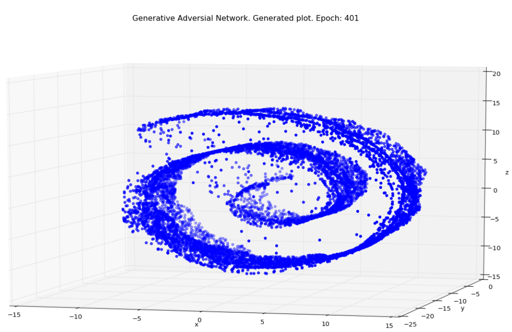
(b) epoch 13



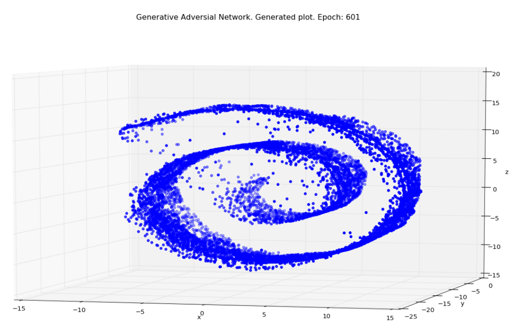
(c) epoch 101



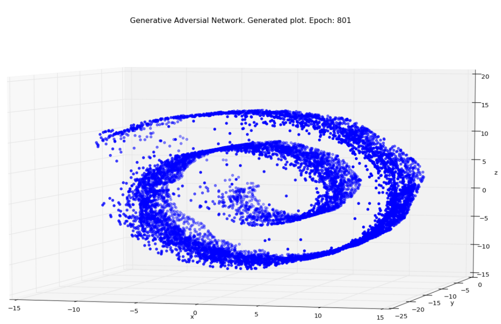
(d) epoch 201



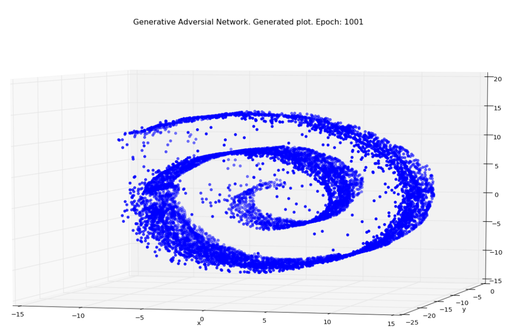
(e) epoch 401



(f) epoch 601



(g) epoch 801



(h) epoch 1001

Figure 2: Generative Adversarial Network. Generated plots

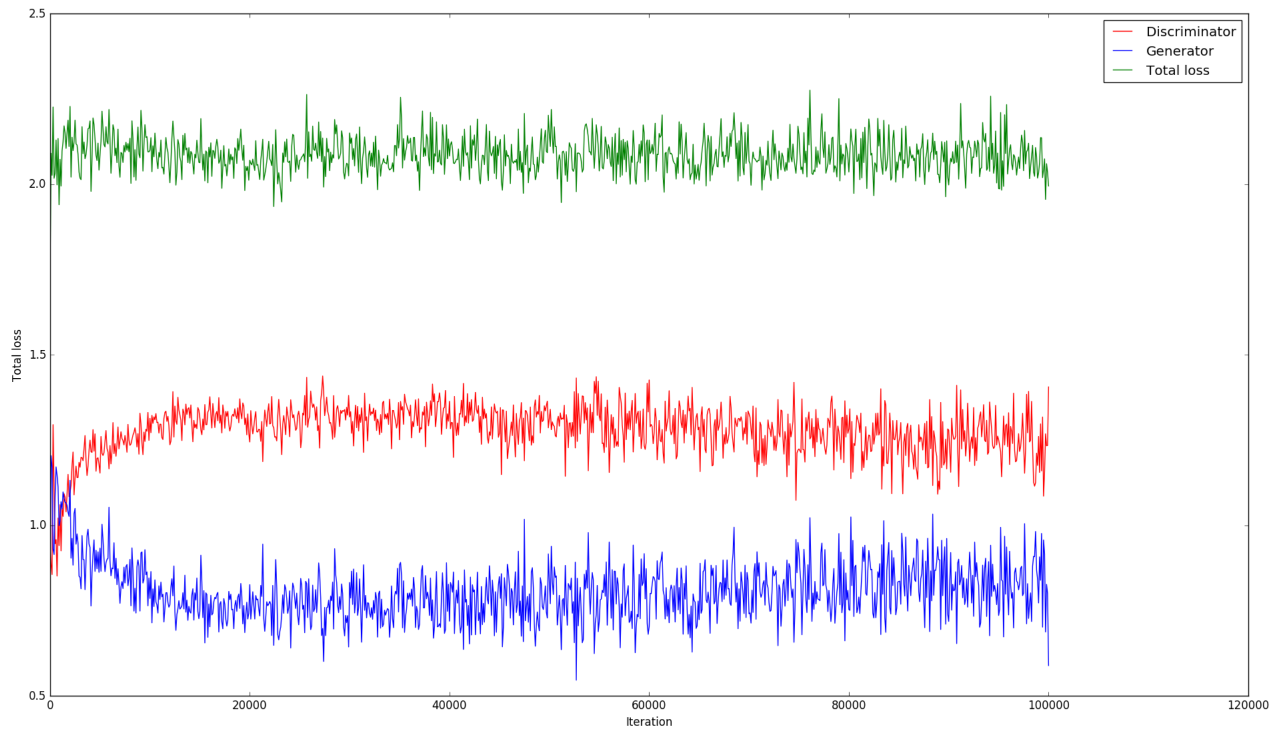


Figure 3: Discriminator, generator, and total loss vs. iterations

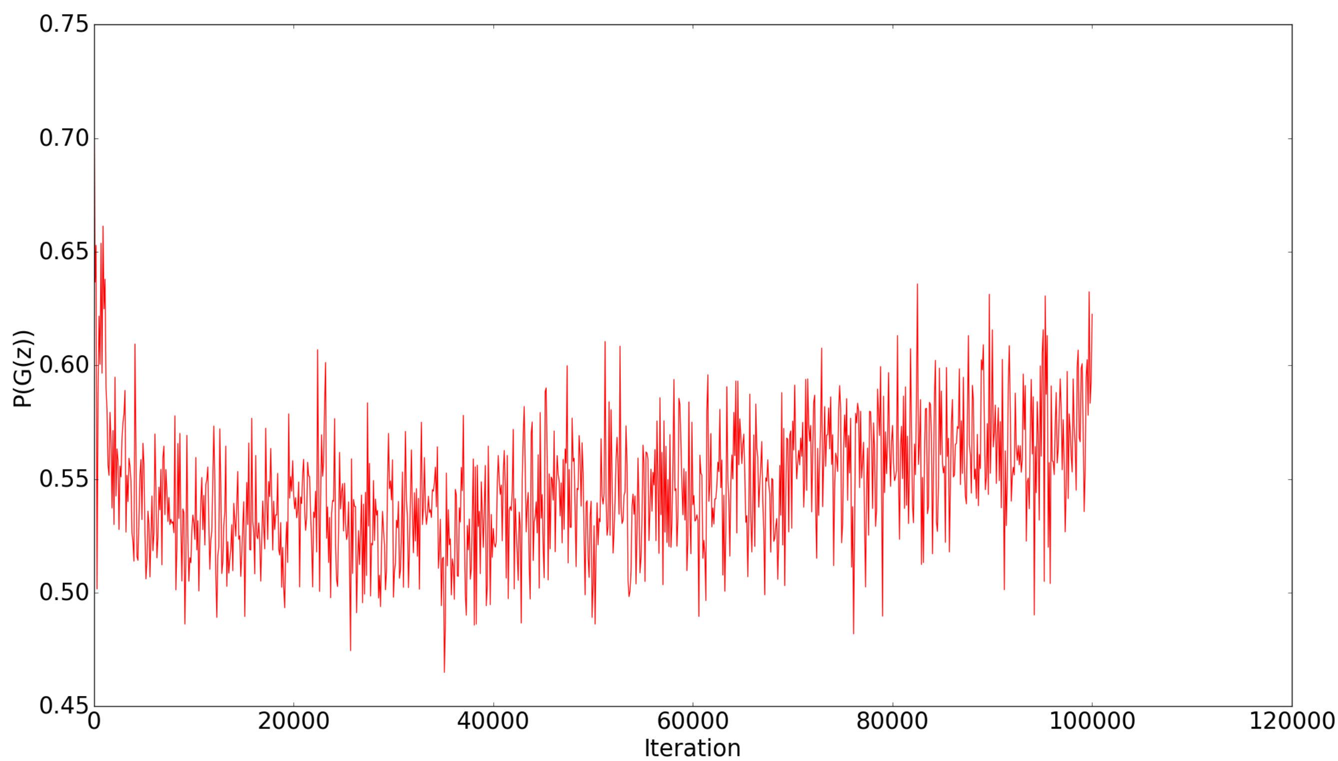


Figure 4: P(G(z)) is close to $\frac{1}{2}$