Coding-based tutorial about generative adversarial network (GAN)

Guo-Wei Wei[1,2,3] and Rui Wang[1]
[1] Department of Mathematics, Michigan State University, MI 48824, USA
[2] Department of Biochemistry and Molecular Biology
Michigan State University, MI 48824, USA
[3] Department of Electrical and Computer Engineering
Michigan State University, MI 48824, USA

Contents

1 Structure of GAN

The generative adversarial network (GAN) is an unsupervised learning method designed by Ian Goodfellow and his colleagues in 2014 [1], which has also proven useful for supervised learning, semi-supervised learning, and reinforcement learning.

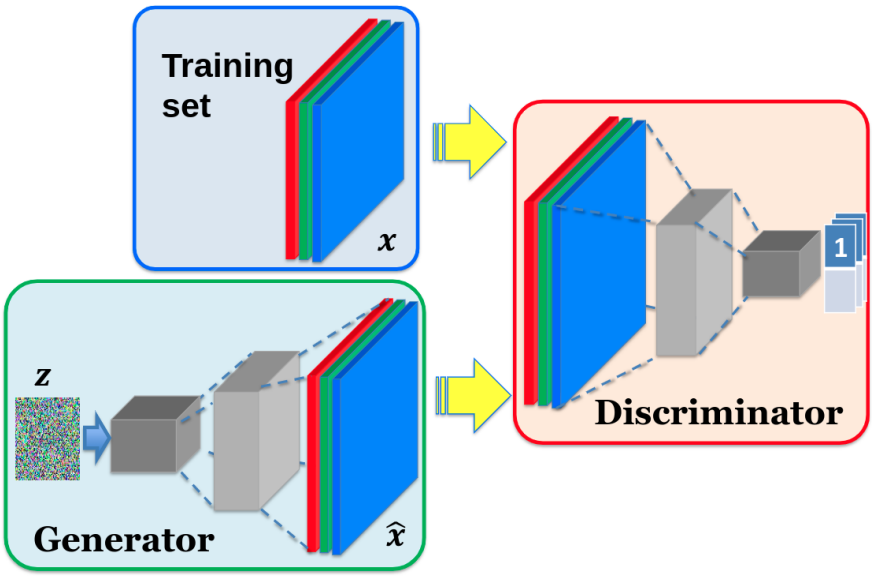

Figure 1: Structure of GAN.

1.1 Loss function

The binary cross entropy loss (BCE):

$$-y\log(p(y)) - (1-y)\log(1-p(y)) \tag{1}$$

In GAN, the generator and discriminator are the two players and take turns involving updates to their model weights. The min and max refer to the minimization of the generator loss and maximization of the discriminator's loss ($\min\max(D, G)$).

The discriminator seeks to maximize the average of the log probability of real images and the log of the inverse probability of fake images.

$$\max \log(D(x)) + \log(1 - D(G(z))). \tag{2}$$

The generator seeks to minimize the log of the inverse probability predicted by the discriminator for fake images.

$$\min \log(1 - D(G(z))). \tag{3}$$

However, this expression has issues with low gradient magnitude. Therefore, it can be reformulated as

$$-\log(D(G(z))) \tag{4}$$

1.2 Generator

1.2.1 Feed-forward

Assume the random input z (noise) has shape $(N, H1)$. Here, N is the batch size, and $H1$ is the length of the noise. Then the feed-forward process for the Generator will be:

1. $G_{z1} = zG_{W0} + G_{b0}$ $(N, H1)(H1, H2) + (1, H2) \rightarrow (N, H2)$
2. $G_{f1} = \tanh(G_{z1})$ $(N, H2)$
3. $G_{z2} = G_{f1}G_{W1} + G_{b1}$ $(N, H2)(H2, D) + (1, D) \rightarrow (N, D)$. Here G_{z2} is the fake_logit in the code.
4. $G_{f2} = \text{sigmoid}(G_{z2})$ (N, D). Here, G_{f2} is the fack_output in the code.
5. $G_{\hat{y}} = G_{f2}.\text{reshape}(-1, 28, 28)$ $(N, 28, 28)$. For visualization.

1.2.2 Back-propagation

The loss function for Generator is:

$$L(G) = -\log(D(G(z))), \tag{5}$$

where D(G(z))

1. $\frac{\partial L(G)}{\partial G_{W1}} = \frac{\partial L(G)}{\partial D(G)} \frac{\partial D(G)}{\partial \text{sigmoid}(G_{z2})} \frac{\partial \text{sigmoid}(G_{z2})}{\partial G_{W1}}$

1.3 Discriminator

1.3.1 Feed-forward

1. $D_{z1} = \text{img}D_{W0} + D_{b0}$ $(N, D)(D, H2) + (1, H2) \rightarrow (N, H2)$. Here, img can be the fake img from the generator and the true img from the training set.
2. $D_{f1} = \tanh(D_{z1})$ $(N, H2)$
3. $D_{z2} = D_{f1}D_{W1} + D_{b1}$ $(N, H2)(H2, 1) + (1, 1) \rightarrow (N, 1)$
4. $D_{f2} = \text{sigmoid}(D_{z2})$ $(N, 1)$
5. $D_{\hat{y}} = D_{f2}$ $(N, 1)$

1.3.2 Back-propagation

The loss function for Discriminator is:

$$L(D) = -\log(D(x)) - \log(1 - D(G(z))), \tag{6}$$

Here, x represents the real image, and $G(z)$ represent the fake image. Therefore, for the real image, we have

$$\frac{\partial L(D)}{\partial D_{W1}} = \frac{\partial L(D)}{\partial D(\text{img})} \frac{\partial D(\text{img})}{\partial (D_{z2})} \frac{\partial (D_{z2})}{\partial D_{W1}}. \tag{7}$$

Here, $\frac{\partial L(D)}{\partial D(\text{img})} = -\frac{1}{D(\text{img})}$.

For the fake imgage, we have

$$\frac{\partial L(D)}{\partial D_{W1}} = \frac{\partial L(D)}{\partial D(G(z))} \frac{\partial D(G(z))}{\partial (D_{z2})} \frac{\partial (D_{z2})}{\partial D_{W1}}. \tag{8}$$

Here, $\frac{\partial L(D)}{\partial D(G(z))} = \frac{1}{1 - D(G(z))}$.

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.