Lab13 Pre-Report: Generative Adversarial Nets

Jiho Park, 2019142056
School of Electrical and Electronic Engineering, Yonsei University

Abstract

Generative Adversarial Nets(GAN) was a novel generative architecture which consists of generator(G) and discriminator(D) neural networks. The generator generates the data. The discriminator guesses whether the given data is fake(generated) data or real among them. The network is trained in adversarial manner, since G minimizes the loss function and D maximizes the loss function. Unlike what is famous for, the paper of GAN mainly addresses the mathematical analysis of it's architecture. Ian et al proved that there exists a unique solution of G and D, and G can capture the distribution of the train data. Lastly, they emphasized that this method doesn't require Markov Chain or Unrolled approximation which makes the computation intractable.

1 Adversarial Nets

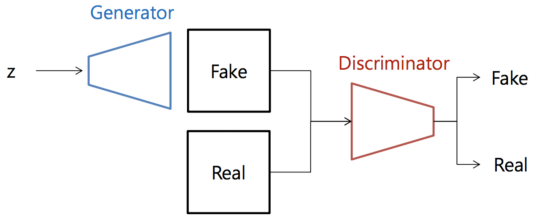

Figure 1: The Basic Architecture of GAN

G is defined as a multi-layer perceptron which generates the data from the input noise z. D is also defined as a multi-layer perceptron which outputs single scalar probability value of that the data is real. D and G are both optimized with the single loss function Eq 1. Since G and D are optimized in direction of minimizing and maximizing the function each, this network is called adversarial nets. And the optimizing process is often called minmax game.

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1-D(G(z)))] \quad (1)$$

Fig 2 is a visualization of the optimization process of GAN(in case G and D both have enough capacity). The black dots are the distribution of original data x. The green graph represents the generative distribution p_g. The blue graph is the scalar probability output of discriminator. After enough training, G and D will reach the point where $p_g = p_{data}$ and where $D_G(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$, like on (d).

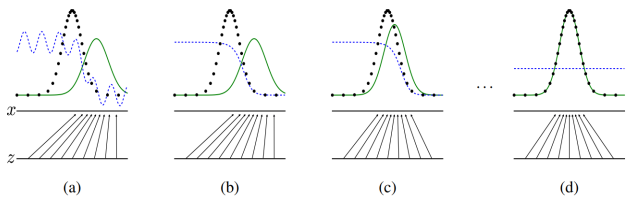

Figure 2: Visualization of Optimizing Process of GAN

In practice, there are more details in training the adversarial net. First, $\log(1-D(G(z)))$ can often saturates at early training. This is because of poor G which makes D to reject the generated data in high proportion. Therefore, they rather train G to maximize $(G(z))$, instead of minimizing $\log(1-D(G(z)))$. Second they optimize D for k steps while optimizing G for just one step. This is to maintain the D near to optimal solution.
The final optimization is as in Fig 3.

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**
  for k steps **do**
    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)},\dots,\boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)},\dots,\boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(\boldsymbol{x}^{(i)}\right)+\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

  end for
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)},\dots,\boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Figure 3: Optimization Algorithm

2 Theoretical Results

2.1 Global Optimality of $p_g = p_{data}$

Proposition 1:
For fixed generator G, the optimal discriminator D is as Eq 2.

$$D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)} \quad (2)$$

Proof:
For given G, the objective function $V(G,D)$ can be modified as Eq 4. And the function $y = a\log(x)+b\log(1-x)$ achieves its maximum in [0,1] where $x = \frac{a}{a+b}$. This can be easily proved by it's differential. Therefore, by looking p_{data} and p_g as a and b each, we can obtain the optimal point(Eq 2) that makes the maximum V.

$$V(G,D) = \int_x p_{data}(x)\log(D(x))dx + \int_z p_z(z)\log(1-D(g(z)))dz \quad (3)$$

$$= \int_x p_{data}(x)\log(D(x)) + p_g(x)\log(1-D(x))dx \quad (4)$$

Theorem 1. Global Optimality
The global minimum of $C(G) = \max_D V(G,D)$ is obtained if and only if $p_g = p_{data}$. At that point, achieves the value $-\log 4$.

Proof:
By using the result of **proposition 1**, the objective function with the optimal D^*, can be modified as Eq 6. Then since the $\log 2$ can come out from the expectation, $-\log 4$ can be extracted as in Eq 7. Each expectation can

be represented in KL-Divergence term(Eq 8). Since the two KL divergence term is just reversed version of each, the summation of them is the Jensen-Shanon Divergence(JSD). JSD is a distance metric, which is always same or larger than 0. As a result, the objective function with given optimal D^*, have the global minimum on $p_g = p_{data}$ with the value $-\log 4$.

$$C(G) = \max_D V(G,D) = E_{x\sim p_{data}(x)}[logD^*(x)] + E_{z\sim p_z(z)}[log(1-D^*(G(z)))] \quad (5)$$

$$= E_{x\sim p_{data}(x)}[\log \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}] + E_{x\sim p_g(x)}[\log \frac{p_g(x)}{p_{data}(x)+p_g(x)}] \quad (6)$$

$$= E_{x\sim p_{data}(x)}[\log \frac{2*p_{data}(x)}{p_{data}(x)+p_g(x)}] + E_{x\sim p_g(x)}[\log \frac{2*p_g(x)}{p_{data}(x)+p_g(x)}] - \log 4 \quad (7)$$

$$= \mathrm{KL}(p_{data}||\frac{p_{data}(x)+p_g(x)}{2}) + \mathrm{KL}(p_g||\frac{p_{data}(x)+p_g(x)}{2}) - \log 4 \quad (8)$$

$$= 2*\mathrm{JSD}(p_{data}||p_g) - \log 4 \quad (9)$$

2.2 Convergence of Algorithm 1

Proposition 2:
With enough capacity, at each step of Algorithm 1, the D can reach its optimum with given G, and p_g is updated in the direction of improving the criterion.

Proof:
Consider $V(G,D) = U(p_g,D)$ as a function of p_g. $U(p_g,D)$ is convex in p_g. The subderivatives of a supremum of convex functions include the derivative of the function at the point where the maximum is attained. $\sup_D U(p_g,D)$ is convex in p_g with a unique global optima, therefore with sufficiently small updates of p_g, p_g converges to p_x, concluding the proof. (This proof is directly quoted from the paper [1])

3 Experiments

Ian et al [1] used MNIST, TFD and CIFAR-10 dataset. For activation function of the network, mixture of ReLU and sigmoid was used for G and maxout was used for D. They evaluated probability of the test dataset under p_g by fitting a Gaussian Parzen window. For high dimensional data, this is method has large variance and not accurate. The sampling results after training are as in Fig 4. Ian et al said that they do not argue that their samples are better than of previous methods. However, they claimed that their method itself is competitive with better generative models and has large potential.

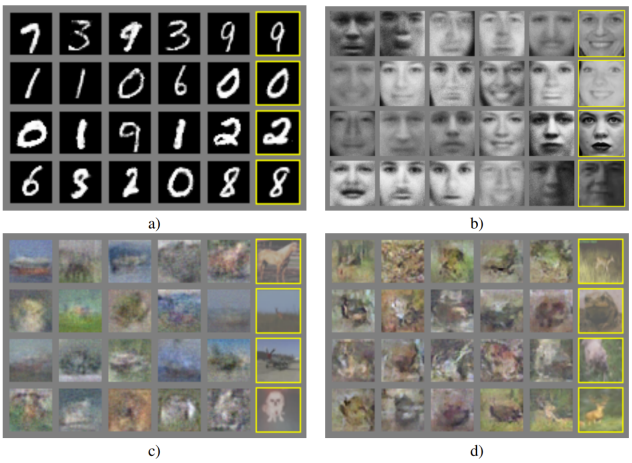

Figure 4: Experiment Results

4 Advantages and Disadvantages

Disadvantages: First, there is no explicit representation of $p_g(x)$. Second, while training, D must be synchronized well with G. This leads to the mode collapse problem.
Advantages: The main advantage is computation. It doesn't require any Markov chain computation, and just requires back-propagation. Also it can represent very sharp distributions.
The advantages and disadvantages of generative models are well organized in Fig 5.

	Deep directed graphical models	Deep undirected graphical models	Generative autoencoders	Adversarial models
Training	Inference needed during training.	Inference needed during training. MCMC needed to approximate partition function gradient.	Enforced tradeoff between mixing and power of reconstruction generation	Synchronizing the discriminator with the generator. Helvetica.
Inference	Learned approximate inference	Variational inference	MCMC-based inference	Learned approximate inference
Sampling	No difficulties	Requires Markov chain	Requires Markov chain	No difficulties
Evaluating $p(x)$	Intractable, may be approximated with AIS	Intractable, may be approximated with AIS	Not explicitly represented, may be approximated with Parzen density estimation	Not explicitly represented, may be approximated with Parzen density estimation
Model design	Nearly all models incur extreme difficulty	Careful design needed to ensure multiple properties	Any differentiable function is theoretically permitted	Any differentiable function is theoretically permitted

Table 2: Challenges in generative modeling: a summary of the difficulties encountered by different approaches to deep generative modeling for each of the major operations involving a model.

Figure 5: Comparison of Generative Models

Reference

[1] Mehdi Mirza Bing Xu David Warde-Farley Sherjil Ozair Aaron Courville Yoshua Bengio Ian J. Goodfellow, Jean Pouget-Abadie. Generative adversarial nets. *NIPS*.