Week13 Pre-Report: Overview of Generative Adversarial Nets (GAN)

Euijin Hong[1]
[1]Department of Electrical and Electronic Engineering, Yonsei University.

1 Introduction

In the paper [2] propose a novel framework for estimating generative models using an adversarial process. This process involves training two models simultaneously: a generative model (G) that captures the data distribution, and a discriminative model (D) that estimates the probability of a sample coming from the training data instead of G. The training objective for G is to maximize the probability of D making a mistake, creating a minimax two-player game. In the space of all possible functions G and D, there exists a unique solution where G learns to generate samples that closely resemble the training data distribution, and D becomes a constant function with a value of 1/2 everywhere. When G and D are represented by multilayer perceptrons (MLP), the entire system can be trained using backpropagation, without the need for Markov chains or unrolled approximate inference networks during training or sample generation. Experimental results showcase the potential of this framework, demonstrating both qualitative and quantitative evaluation of the generated samples.

2 Theory of Adversarial Nets

When the generative and discriminative models are all multilayer perceptrons, the adversarial framework can be straightforward to apply. To learn the distribution of the generator p_g over data $\mathbf{x}$, we introduce beforehand an input noise variables $p_z(z)$. We then use a multilayer perceptron, where its mapping to data space is denoted as $G(z;\theta_g)$, to map the noise variables z to the data space. The function G is differentiable and its parameters are denoted as θ_g. Additionally, we have another multilayer perceptron, $D(\mathbf{x};\theta_d)$, which outputs a scalar value. $D(\mathbf{x})$ represents the probability that x is a real data sample rather than generated from p_g. We train D to maximize the probability of correctly labeling both training examples and samples generated by G. Simultaneously, we train G to minimize the logarithm of 1 minus $D(G(z))$. In summary, D and G engage in a two-player minimax game with a value function $V(G,D)$ as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1-D(G(\boldsymbol{z})))].$$

Next, theoretical analysis of adversarial nets is provided, demonstrating that the training criterion enables recovery of the data generating distribution when G and D have sufficient capacity (in the non-parametric limit). Figure 1 offers a more intuitive explanation of the approach.

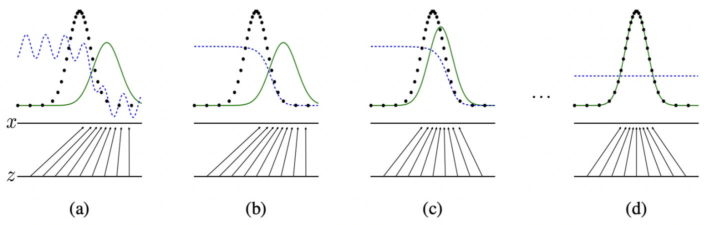

Figure 1: GAN is trained through the simultaneous updating of the discriminative distribution (D) and the generative distribution (G). The goal is for D to accurately distinguish samples from the data generating distribution (p_{data}) and the generative distribution (p_g). (a) At a near-converged state, p_g becomes similar to p_{data} and D becomes a partially accurate classifier. (b) In the inner loop of the algorithm, D is trained to discriminate samples from data, converging to the optimal discriminator $D^*(x) = p_{data}(x)/(p_{data}(x)+p_g(x))$. (c) After an update to G, the gradient of D guides the generated samples $G(z)$ towards regions that are more likely to be classified as data. (d) After several training steps, if G and D have sufficient capacity, they will reach a point where further improvement is not possible because p_g equals p_{data}. The discriminator is then unable to differentiate between the two distributions, resulting in $D(x) = 1/2$.

However, in practice, implementing the game requires an iterative numerical approach. Optimizing D to convergence in the inner loop of training is computationally impractical and prone to overfitting on finite datasets. To address this, alternating between k steps of optimizing D and one step of optimizing G is done. This ensures that D remains close to its optimal solution as long as G changes slowly enough. This approach is similar to how SML/PCD training [7, 8] maintains samples from a Markov chain between learning steps to avoid burn-in within the inner loop of learning. The procedure is outlined in Algorithm 1.

In practice, the above equation of $V(D,G)$ may not provide sufficient gradients for G to learn effectively. During the early stages of learning when G is poor, D can confidently reject samples that are clearly different from the training data, leading to saturation of the $log(1-D(G(z)))$ term. Instead of training G to minimize $log(1-D(G(z)))$, we can train G to *maximize* $logD(G(z))$. This alternative objective function yields the same fixed point for the dynamics of G and D but provides stronger gradients, particularly in the early stages of learning.

3 Theoretical Results of GAN

The generator G is responsible for generating samples $G(z)$ from a probability distribution p_g, where z is drawn from the distribution p_z. The objective of Algorithm 1 is to converge to a good estimator of the true data distribution p_{data}, given enough capacity and training time.

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.
```

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

First, let us consider the optimal discriminator D for any given generator G.

Proposition 1. *For G fixed, the optimal discriminator D is:*

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

Proof: The training criterion for the discriminator D, when given any generator G, is to maximize the quantity $V(G,D)$.

$$\begin{aligned} V(G,D) &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x})\log(D(\boldsymbol{x}))dx + \int_z p_{\boldsymbol{z}}(\boldsymbol{z})\log(1-D(g(\boldsymbol{z})))dz \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x})\log(D(\boldsymbol{x})) + p_g(\boldsymbol{x})\log(1-D(\boldsymbol{x}))dx \end{aligned}$$

The function $y \to alog(y)+blog(1-y)$ achieves its maximum in the interval $[0,1]$ at $a/(a+b)$, for any $(a,b)\in R^2$ $\{0,0\}$. Therefore, the discriminator does not need to be defined outside of the union of the support of p_{data} and p_g, which concludes the proof.

The training objective for the discriminator D can be interpreted as maximizing the log-likelihood for estimating the conditional probability $P(Y=y|x)$, where Y indicates whether x comes from p_{data} (with y = 1) or from p_g (with y = 0). The minimax game in Equation 1 can now be reformulated as:

This is an extended abstract. The full paper is available at the Generative Adversarial Nets.

$$
\begin{aligned}
C(G) =& \max_D V(G,D)\\
=&\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1-D^*_G(G(\boldsymbol{z})))]\\
=&\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1-D^*_G(\boldsymbol{x}))]\\
=&\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x})+p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x})+p_g(\boldsymbol{x})}\right]
\end{aligned}
$$

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At the point, $C(G)$ achieves the value $-log4$.*

Proof: For $p_g = p_{data}$, $D^*G(x) = 1/2$. Thus, we find $C(G) = log1/2 + log1/2 = -log4$. To prove this is the best obtainable value of $C(G)$, only being reached for $p_g = p_{data}$, inspect:

$$\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[-\log 2] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[-\log 2] = -\log 4$$

and by subtracting the equation from $C(G) = V(D^*_G, G)$, we can obtain:

$$C(G) = -\log(4) + KL\left(p_{\text{data}}\middle\|\frac{p_{\text{data}}+p_g}{2}\right) + KL\left(p_g\middle\|\frac{p_{\text{data}}+p_g}{2}\right)$$

where KL is the Kullback–Leibler divergence. The Jensen–Shannon divergence (JSD) between the model's distribution and the data generating process can be recognized from the previous equation as:

$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right)$$

Since the JSD between two distributions always shows non-negative and zero values only when they are equal, $C^* = -log(4)$ is the global minimum of $C(G)$ and the only solution is $p_g = p_{data}$, for example, the generative model (G) perfectly mimicking the data generating process.

3.2 Convergence of Algorithm 1

Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1-D^*_G(\boldsymbol{x}))]$$

then p_g converges to p_{data}.

Proof: When considering $V(G,D) = U(p_g, D)$ as a function of p_g, we find that $U(p_g, D)$ is convex in p_g. In the context of supremum of convex functions, the subderivatives of this function include the derivative of the function at the point where its maximum is achieved. In other words, if $f(x) = sup_{\alpha\in A} f_\alpha(x)$ and $f_\alpha(x)$ is convex in x for every α, then $\partial f_{\beta(x)\in\partial f}$ if $\beta = argsup_{\alpha\in A} f_\alpha(x)$.

Applying this concept, we can compute a gradient descent update for p_g at the optimal D given the corresponding G. The function $sup_D U(p_g, D)$ is convex in p_g and has a unique global optimum, as proven in Theorem 1. Consequently, with sufficiently small updates of p_g, it converges to p_x, thus concludes the proof.

Practically, adversarial nets use the function $G(z;\theta_g)$ to represent a restricted set of p_g distributions, and the optimization is performed on the parameters θ_g instead of p_g directly. The use of a multilayer perceptron to define G introduces multiple critical points in the parameter space. Despite the lack of theoretical guarantees, the remarkable performance of multilayer perceptrons in practical applications suggests that they are a viable model choice.

4 Experiments

The researchers trained adversarial nets on datasets including MNIST[5], the Toronto Face Database(TFD) [6], and CIFAR-10 [4]. They estimated probability of the test set data under p_g by fitting a Gaussian Parzen window to the samples generated with G and reporting the log-likelihood under this distribution. The σ parameter of the Gaussians was obtained by cross validation on the validation set. This method of estimating the likelihood has somewhat high variance and does not perform well in high dimensional spaces but is the best method so far.

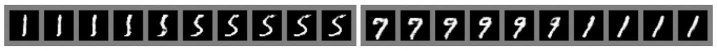

Figure 2: MNIST numbers obtained by linear interpolation between coordinates in z space of the entire model.

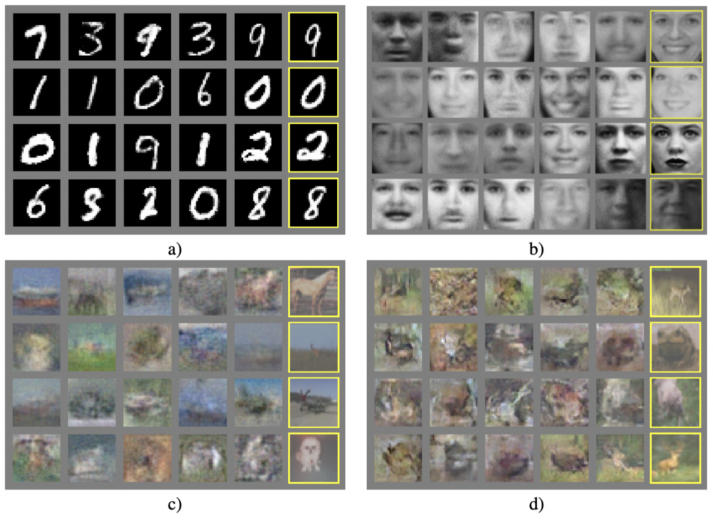

Figure 3: The visualized samples from the model. a) MNIST b) TFD c) CIFAR-10 (fully connected model) d) CIFAR-10 (convolutional discriminator and "deconvolutional" generator). The right end samples with yellow edges are the nearest training example of the neighboring sample, showing that the model did not memorize the training set. Samples are randomly drawn (not cherry-picked). These images show exact samples from the model distributions and uncorrelated since the sampling process does not depend on Markov chain mixing.

5 Advantages and Disadvantages of GAN

The disadvantages include the absence of an explicit representation of $p_g(x)$ and the need for considerable synchronization between the discriminator (D) and the generator (G) during training, in order to maintain diversity in the generated samples. However, the advantages of this framework are significant. It eliminates the need for Markov chains and inference during learning, relies solely on backpropagation for obtaining gradients, and allows for the incorporation of a wide range of functions into the model. Computational efficiency is also a key advantage. Furthermore, adversarial models benefit from not directly updating the generator's parameters with data examples but rather with gradients flowing through the discriminator, which can offer a statistical advantage. Additionally, adversarial networks have the ability to represent sharp and even degenerate distributions, whereas methods based on Markov chains require a certain level of blurriness in the distribution for effective mixing between modes.

6 Conclusion and Proposed Future Works of GAN

The GAN framework allows for several straightforward extensions:

- Conditional generative models: By incorporating an additional input variable c into both the generator G and discriminator D, it is possible to obtain a conditional generative model $p(x|c)$.
- Learned approximate inference: An auxiliary network can be trained to predict the latent variable z given an observed variable x. This approach is similar to the wake-sleep algorithm's inference net [3], but with the advantage that the inference net can be trained with a fixed generator network after the generator has completed training.
- Modeling conditional distributions: The framework can be used to approximate all conditionals $p(x_S|x'_S)$, where S is a subset of indices of the variable x. This is achieved by training a family of conditional models that share parameters, effectively extending the deterministic MP-DBM (Multi-Prediction Deep Boltzman Machine) [1].
- Semi-supervised learning: The features learned by the discriminator or inference net can be leveraged to improve the performance of classifiers when only limited labeled data is available.
- Efficiency improvements: Methods for better coordination between the generator and discriminator or for selecting more suitable distributions to sample z from during training can greatly accelerate the training process.

The success of the adversarial modeling framework demonstrated in this paper suggests that these research directions have the potential to be useful and further advance the field.

[1] Ian Goodfellow, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Multi-prediction deep boltzmann machines. *Advances in Neural Information Processing Systems*, 26, 2013.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C Courville, and Yoshua Bengio. Generative adversarial nets ian. *Mining of Massive Datasets; Cambridge University Press: Cambridge, UK*, 2014.

[3] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

[4] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[5] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] Joshua Susskind, Adam Anderson, and Geoffrey E Hinton. The toronto face dataset. Technical report, Technical Report UTML TR 2010-001, U. Toronto, 2010.

[7] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008.

[8] Laurent Younes. On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics: An International Journal of Probability and Stochastic Processes*, 65(3-4): 177–228, 1999.