2021/1/1 이유진 (youjin lee)

Paper Review

Generative Adversarial Nets

Ian J. Goodfellow , Jean Pouget-Abadie , Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair , Aaron Courville, Yoshua Bengio

NIPS, 2014

## Abstract

우리는 2개의 모델(generative 모델 G – 데이터 분포를 포착, discriminative 모델 D – 샘플이 모델 G보다는 학습 데이터로부터 나왔을 확률을 추정)을 동시에 학습하는 adversarial 과정을 통해 generative 모델을 평가하는 새로운 프레임워크를 제안한다. G의 학습 절차는 D가 실수할 확률을 최대화하는 하는 것이다. 이 프레임워크는 2명의 플레이어 간 minimax 게임과 일치한다. 임의의 함수 G와 D의 공간에는 학습 데이터 분포를 복구한 G와 D가 어디에서나 1/2로 같은 독특한 솔루션이 존재한다. G와 D가 다층 퍼셉트론에 의해 정의된 경우 전체 시스템은 역전파에 의해 학습될 수 있다. 각각 샘플을 학습하고 생성하는 동안 어떤 Markov chains나 unrolled approximate inference networks도 필요 없다. 실험은 생성된 샘플의 정성적 · 정량적인 평가를 거쳐 이 프레임워크의 잠재성을 증명했다.

## Introduction

딥러닝의 목적은 인공지능과 맞닥뜨리는 데이터의 확률 분포를 표현하는 rich, hierarchical 모델을 발견하는 것이다. 지금까지 가장 두드러진 성공은 discriminative 모델이 포함되며, generative 모델은 다루기 힘든 확률 계산 추정의 어려움과 linear unit의 레버리지 효과 때문에 영향이 미비했다. 그래서 우리는 이 어려움을 회피하는 추정 절차를 지닌 새로운 generative model을 제안한다.

여기서 제안하는 adversarial nets는 generative 모델을 상대 모델과 대항하게 만드는데: 그 상대 모델인 discriminative 모델은 샘플이 G 모델 분포로부터 나온 것인지 데이터 분포로부터 나온 것인지 결정하는 것을 학습한다. G는 가짜 통화를 생산하고 사용하는 위조 지폐범들과 유사하고, 반면 D는 이 위조 지폐를 찾는 경찰이라고 보면 된다. 이 게임 내의 경쟁은 진짜로부터 위조를 구별할 수 없게 될 때까지 양쪽 모두 각자의 방법을 향상시키도록 이끈다. 이 구조는 많은 종류의 모델들과 최적화 알고리즘에 대한 구체적인 학습 알고리즘을 산출할 수 있다. 본 논문에서는 우리는 다층 퍼셉트론을 통해 랜덤 노이즈를 전달하여 generative model이 샘플을 생성하고 discriminative 모델 또한 다층 퍼셉트론으로 구성된 특별한 케이스를 탐색하고 이를 adversarial nets이라고 한다.

## Related work

최근까지 대부분의 generative 모델 관련 연구들은 확률 분포 함수의 파라미터 구체화를 제공하는 데 집중되어 있었고, 이 부류의 모델 중에서는 Boltzmann machine이 가장 성공적이었다. 이러한 모델들은 일반적으로 아주 다루기 힘든 likelihood 함수를 가졌고 이 likelihood의 gradient에 대한 막대한 계산이 필요했는데, 이러한 점 때문에 likelihood를 나타내지 않는 “generative machines” 개발을 목표로 하게 되었다. Generative

stochastic network는 복잡한 계산 대신 역전파로 학습했고, 위 연구는 generative stochastic network에서 사용된 Markov chain을 제거해 모델을 더욱 확장했다.

본 연구는 다음과 같은 관찰을 통해 generative 과정을 거쳐 derivatives를 역전파한다.

$$\lim_{\alpha \to 0} \nabla_x \mathbb{E}_{\epsilon \sim \mathcal{N}(0,\sigma^2 I)} f(x+\epsilon) = \nabla_x f(x).$$

해당 연구를 진행했던 당시 Kingma, Welling, Rezende et al.이 더 일반적인 stochastic backpropagation rules을 개발했음을 인지하지 못했는데 이 rules은 우리 연구에서 하이퍼파라미터로 다룬 generator의 조건부 분산을 학습할 수 있게 해줄 수 있다. 그들은 rules을 활용하여 variational autoencoders(VAEs)의 학습에 사용했다. VAEs는 GANs과 마찬가지로 구별이 가능한 generator network와 2번째 인공 신경망과 쌍을 이루는 반면, VAE의 2번째 신경망이 대략적인 추론을 수행하는 인식 모델이라는 점이 다르다. GANs은 visible units을 통한 differentiation이 필요하기에 이산형 데이터를 모델링할 수 없고, VAEs는 hidden units을 통한 differentiation이 필요해 이산형 잠재 변수를 가질 수 없다.

또한 이전 연구에서 generative 모델 학습을 위해 discriminative 기준을 사용하여 접근했는데, 이 방법에 사용된 기준은 generative 모델에 적용되기에 매우 까다로웠다. 심지어 모델을 추정하는 것조차 어려웠다. Noise-contrastive estimation(NCE)는 고정된 noise 분포에서 데이터를 식별하는데 유용한 가중치를 학습함으로써 generative 모델을 학습시키는 방법이다. 이전에 학습된 모델을 사용하는 것은 성능을 향상시키는 일련의 모델을 학습이 가능하다. 하지만 NCE는 "discriminator"를 noise 분포 확률 밀도와 모델 분포 확률 밀도의 비율로 정의하기 때문에 두 밀도를 역전파하고 평가할 수 있는 능력이 필요하다는 주요한 한계점이 존재한다.

몇몇의 이전 연구들은 2개의 인공 신경망을 경쟁시킨다는 일반적인 개념을 사용했고, 그와 가장 관련 있는 연구는 predictability minimization이다. 이 방법은 각각의 hidden unit은 다른 모든 hidden units의 값이 주어진 그 hidden unit의 값을 예측하는 2번째 네트워크의 output과 다르게 학습되었다.

| Predictability minimization | Generative adversarial network |
|---|---|
| 네트워크 간의 경쟁만이 유일한 학습 기준 | |
| 경쟁 본질 차이<br>Output - single scalar | G - rich & high dimensional vector 생성 |
| 학습 과정 구체화 차이<br>목적 함수를 최소화하는 optimization 문제 | Minimax 게임 |

GANs은 때때로 데이터와 유사하지만 잘못 분류된 예제를 찾기 위해 분류 네트워크 입력에 직접 gradient based optimization을 사용하여 찾은 예제인 "adversarial examples"라는 관련 개념과 혼동되어 왔다. Adversarial examples은 generative 모델을 학습하기 위한 메커니즘이 아니라는 점에서 위 논문의 연구 내용과 다르지만, 이는 신경망이 사람의 예상과 다른 방식으로 행동함을 보여주는 분석 도구이다.

### Adversarial nets

Adversarial 모델링 프레임워크는 두 모델 모두 MLP일 때 적용하는 것이 가장 간단하다. 데이터 $x$에 대한 generator 분포 $p_g$를 학습하기 위해 input noise 변수를 $p_z(z)$로 정의하고 $G(z;\theta_g)$라고 표현한다. 또한 $p_g$가 아닌 데이터로부터 나온 확률을 $D(x)$로 표현하고 outputs이 single scalar인 $D(x;\theta_d)$로 정의한다. 우리는 G로부터 나온 샘플과 학습 예제 둘 모두에 맞는 레이블을 할당할 확률을 최대화하는 D를 학습시키고 동시에 $\log\big(1-D\big(G(x)\big)\big)$를 최소화하는 G를 학습시킨다. 다시 말해 D와 G는 아래와 같은 minimax 게임을 한다.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}\left[\log\left(1-D\left(G(z)\right)\right)\right] \quad (1)$$

식 (1)은 G가 잘 학습되도록 충분한 gradient를 제공하지는 못할 수도 있다. 학습 초기에 G가 형편없을 때, D는 학습 데이터와 명백히 다르다는 이유로 높은 confidence를 가진 채 G의 샘플을 거절할 수 있는데 이 경우 $\log\left(1-D\left(G(x)\right)\right)$는 saturates된다. 그 결과 $\log\left(1-D\left(G(x)\right)\right)$ 최소화가 아닌 $\log D\left(G(x)\right)$를 최대화하는 G로 학습될 수도 있다.

## Theoretical Results

우리는 Adversarial nets의 이론적 분석을 제시하여 G와 D가 충분한 capacity를 제공함에 따라 학습 기준이 데이터 생성 분포를 복구할 수 있음을 보여준다. 이 방법에 대해 덜 형식적이지만 더 교육적인 설명은 Figure1을 참고하면 된다. 실제로 우리는 반복적이고 수치적인 접근법을 사용해 해당 게임을 수행해야만 했다. D가 학습의 내부 루프내에서만 완전히 최적화하는 것은 계산적으로 금지되었고 한정적인 데이터는 overfitting을 초래한다. 대신 G의 최적화 단계와 G의 최적화 단계를 한 단계씩 번갈아 진행했고 이는 G가 충분히 천천히 변화하는 한 D를 optimal solution 근처에 유지되는 결과를 가져왔다. 이 과정은 Algorithm 1에 설명되어 있다.

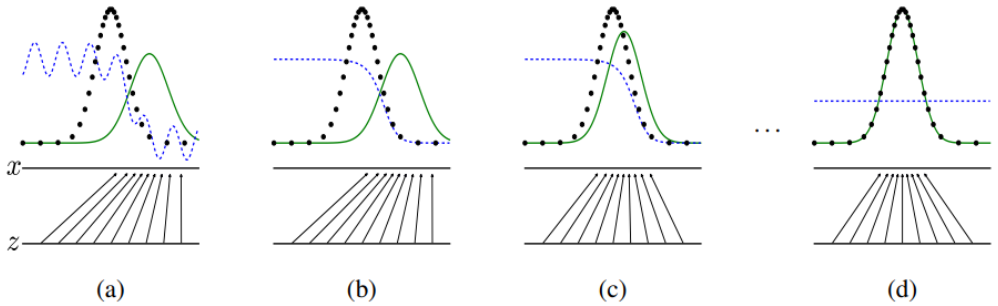


Figure 1: Generative adversarial nets are trained by simultaneously updating the **d**iscriminative distribution ($D$, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\boldsymbol{x}}$ from those of the **g**enerative distribution $p_g$ (G) (green, solid line). The lower horizontal line is the domain from which $\boldsymbol{z}$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $\boldsymbol{x}$. The upward arrows show how the mapping $\boldsymbol{x} = G(\boldsymbol{z})$ imposes the non-uniform distribution $p_g$ on transformed samples. $G$ contracts in regions of high density and expands in regions of low density of $p_g$. (a) Consider an adversarial pair near convergence: $p_g$ is similar to $p_{\text{data}}$ and $D$ is a partially accurate classifier. (b) In the inner loop of the algorithm $D$ is trained to discriminate samples from data, converging to $D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x})+p_g(\boldsymbol{x})}$. (c) After an update to $G$, gradient of $D$ has guided $G(\boldsymbol{z})$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if $G$ and $D$ have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(\boldsymbol{x}) = \frac{1}{2}$.

### < Global Optimality of $p_g = p_{data}$ >

우선 주어진 generator G에 대해 optimal discriminator D가 있다고 생각해보자.

**전제 조건 1.** 고정된 G에 대해 optimal discriminator D는

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

**증명.** 어떤 Generator G가 주어지든 관계없이 Discriminator D의 학습 기준은 $V(G,D)$를 최대화하는 것이다.

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

$$= \int_x p_{data}(x)\log\big(D(x)\big)\,dx + p_g(x)\log\big(1-D(x)\big)\,dx \tag{3}$$

$(a,b) \in \mathbb{R}^2$일 때, 함수 $y \to a\log(y)+b\log(1-y)$는 [0,1]에서 최솟값 $\frac{a}{a+b}$을 가진다. D는 $Supp(p_{data}) \cup Supp(p_g)$의 외부에서 정의될 필요가 없다. D의 학습 목적은 조건부 확률 $P(Y=y|x)$을 추정하는 log-likelihood를 최대화하는 것으로 해석될 수 있음에 주목한다. ($Y$는 $x$가 $p_{data}$로부터 나온 것인지($y=1$) $p_g$로부터 나온 것인지($y=0$)를 나타냄) 이에 따라 식 (1)을 다시 정리하면:

$$\begin{aligned} C(G) &= \max_D V(G,D) \\ &= \mathbb{E}_{x\sim p_{data}}[\log D_G^*(x)] + \mathbb{E}_{z\sim p_z}\left[\log\left(1-D_G^*(G(z))\right)\right] \\ &= \mathbb{E}_{x\sim p_{data}}[\log D_G^*(x)] + \mathbb{E}_{x\sim p_g}\left[\log\left(1-D_G^*(x)\right)\right] \\ &= \mathbb{E}_{x\sim p_{data}}\left[\log\frac{p_{data}(x)}{P_{data}(x)+p_g(x)}\right] + \mathbb{E}_{x\sim p_g}\left[\log\frac{p_g(x)}{P_{data}(x)+p_g(x)}\right] \end{aligned} \tag{4}$$

**정리 1.** 가상 학습 기준 $C(G)$의 global 최소값은 $p_g = p_{data}$ 인 경우에만 달성한다. 이 때 $C(G)$의 값은 $-\log 4$이다.

**증명.** $p_g = p_{data}$일 때, 식(2)에 의거하여 $D_G^*(x) = \frac{1}{2}$ 이다. 이에 따라 식 (4)에서 우리는 $C(G) = \log\frac{1}{2} + \log\frac{1}{2} = -\log 4$ 임을 알 수 있다. $C(G)$의 최적값을 찾기 위해서는 아래의 식을 관찰해보면, $\mathbb{E}_{x\sim p_{data}}[-\log 2] + \mathbb{E}_{x\sim p_g}[-\log 2] = -\log 4$ 이고 $C(G) = V(D_G^*, G)$에서 해당 표현을 제외하면 아래의 식을 얻을 수 있다. (KL = Kullback-Leibler divergence)

$$C(G) = -\log(4) + KL\left(p_{data} \,\|\, \frac{p_{data}+p_g}{2}\right) + KL\left(p_g \,\|\, \frac{p_{data}+p_g}{2}\right) \tag{5}$$

우리는 이전 표현에서 모델의 분포와 데이터 생성 과정 간의 Jensen-Shannon divergence를 인식한다. 두 분포 간의 JS divergence는 항상 양의 값을 가지므로 (같은 경우에는 0) 우리는 $C(G) = -\log 4$가 global minimum임과 그 solution은 $p_g = p_{data}$일 때만 나오는 것임을 보여주었다. 즉, generative 모델은 완벽히 데이터 분포를 복제할 수 있다.

$$C(G) = -\log(4) + 2\cdot JSD\big(p_{data} \,\|\, p_g\big) \tag{6}$$

**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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    - Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ 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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ 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.

< Convergence of Algorithm 1 >

**전제 조건 2.** 만약 G와 D가 충분한 capacity를 가지고 있다면, 그리고 Algorithm 1의 각 단계에서 D가 주어진 G에 대해 optimum에 도달할 수 있게 하고 $p_g$는 기준을 개선시키기 위해 아래와 같이 업데이트된다.

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

그리고 $p_g$ 는 $p_{data}$로 수렴된다.

**증명.** 위 기준에서 $p_g$의 함수 $V(G,D) = U(p_g, D)$로 생각하고 이 때 $p_g$는 $U(p_g, D)$에 convex하다는 점에 유의한다. Convex 함수의 상한 subderivative는 최대값이 달성된 지점에서 함수의 derivative를 포함한다. 즉, 만약 $f(x) \in sup_{\alpha\in A} f_\alpha(x)$이고 $f_\alpha(x)$가 모든 $\alpha$마다 $x$에서 convex라면 $\beta = \arg sup_{\alpha\in A} f_\alpha(x)$인 경우 $\partial f_\beta(x) \in \partial f$이다. 이는 주어진 G에 상응하는 최적의 D에서 $p_g$의 업데이트를 위해 gradient descent를 계산하는 것과 같다. $sup_D U(p_g, D)$가 Thm1에서 증명된 대로 unique global optima를 가진 $p_g$에서 convex하므로 $p_g$의 업데이트가 충분히 적으면 $p_g$는 $p_x$로 수렴된다.

## Experiments

우리는 MNIST, the Toronto Face Database(TFD), CIFAR-10을 포함한 다양한 데이터셋으로 adversarial nets을 학습시켰다. G는 ReLU, Sigmoid 활성화 함수를 사용한 반면 D는 maxout 함수를 사용했다. 또한 D를 학습시킬 때 Dropout을 사용했고, G의 최하층에 있는 noise만 inputs으로 사용했다. G에서 생성된 샘플에 Gaussian Parzen window를 적용하고 이 분포의 log-likelihood를 기록하며 테스트셋의 확률을 추정했다. 그 결과는 Table 1에 나와있는데 이 방법이 다소 높은 variance를 가지고 high dimensional 공간에서 높은 성능을 보이지는 않지만 우리가 아는 선에서는 가장 좋은 방법이다. Figure2, Figure3에서는 학습 후 generator가 생성한 샘플들이다. 우리는 기존 방법에서 생성된 샘플들보다 이 샘플들이 더 좋다고 주장하지는 않지만, 이 샘플들이 적어도 더 나은 generative 모델들과 경쟁적이고 adversarial network의 가능성을 강조할 수 있다고 믿는다.

| Model | MNIST | TFD |
|---|---|---|
| DBN [3] | $138 \pm 2$ | $1909 \pm 66$ |
| Stacked CAE [3] | $121 \pm 1.6$ | $\mathbf{2110 \pm 50}$ |
| Deep GSN [5] | $214 \pm 1.1$ | $1890 \pm 29$ |
| Adversarial nets | $\mathbf{225 \pm 2}$ | $\mathbf{2057 \pm 26}$ |

Table 1: Parzen window-based log-likelihood estimates. The reported numbers on MNIST are the mean log-likelihood of samples on test set, with the standard error of the mean computed across examples. On TFD, we computed the standard error across folds of the dataset, with a different $\sigma$ chosen using the validation set of each fold. On TFD, $\sigma$ was cross validated on each fold and mean log-likelihood on each fold were computed. For MNIST we compare against other models of the real-valued (rather than binary) version of dataset.

Figure 3: Digits obtained by linearly interpolating between coordinates in $z$ space of the full model.

## Advantages and disadvantages

우리가 제시한 이 새로운 프레임워크는 이전 것과 비례하여 장점과 단점을 지닌다. 단점으로는 주로 $p_g(x)$의 명시적 표현이 없고, D의 업데이트 없이 학습하는 동안 D가 G와 동기화가 잘 되어야 한다는 것이다. ("the Helvetica scenario") 장점으로는 Markov chains이 필요하지 않고, gradients를 얻기 위해 역전파만 있으면 된다는 것과 학습하는 동안 추론을 할 필요가 없다는 것이다. Table 2에 다른 generative 모델과 비교하여 위 프레임워크가 지닌 장점과 단점을 요약했다.

| | 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 | Models need to be designed to work with the desired inference scheme — some inference schemes support similar model families as GANs | 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.

## Conclusion and future work

1. 조건부 generative 모델 $p(x|c)$은 G와 D 모두에 inputs으로 $c$를 추가하여 얻을 수 있다.

2. $x$가 주어질 때 $z$를 예측하는 보조 network를 학습하여 '학습된 근사값 추론'을 수행할 수 있다. 이는 wake-sleep algorithm과 유사하지만 inference net이 G가 학습을 마친 후에 확정된 G 내에서 훈련될 수 있다는 장점이 있다.

3. 파라미터를 공유하는 조건부 모델군을 학습하여 $x$의 하위집합인 $S$에 대해 모든 조건부 $p(x_s|x_{not\ s})$를 대략적으로 모델링 할 수 있다.

4. 준-지도 학습: D나 inference net으로부터 나온 features들은 라벨링이 된 데이터의 양이 제한적일 때 분류 모델의 성능을 향상시키기 위해 사용할 수 있다.

5. 효율성 향상: 학습은 G와 D를 조정하거나 학습 동안 더 나은 분포 샘플 $z$를 결정하는 방법을 고안하여 가속화될 수 있다.