Generative Adversarial Nets(GAN) 논문 리뷰

1. Abstract

적대적 프로세스를 통한 생성 모델을 추정하는 새로운 프레임워크를 제안
1) 생성모델 G(generative model) : 판별모델 D가 correct label을 제대로 판별할 수 없을 정도로 훈련 데이터과 유사한 데이터를 만드는 모델
2) 판별모델 D(discriminative model) : 생성모델 G가 아닌 훈련 데이터에서 sample이 나올 확률을 추정하는 모델

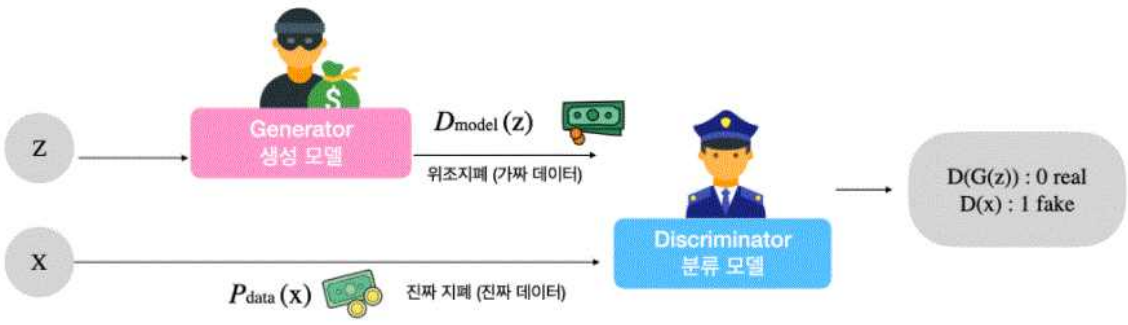
이때 모델 G의 훈련과정은 D가 실수할 확률을 최대화하는 것이다.
이는 마치 minmax 2 player game이라고 볼 수 있다. 임의의 G, D에 의한 함수 공간에서 G는 훈련 데이터의 분포를 학습하여, 임의의 노이즈를 입력 받아 훈련 데이터와 같은 분포로 생성하고, D는 해당 인풋이 생성된 이미지인지 훈련 데이터로부터 나온 이미지인지에 대한 확률이 1/2(구분할 수 없게 되는 확률)가 되게 한다.

기존의 연구들에서 제시된 주요 생성모델들이 기반을 두고 있는 Markov chains, unrolled approximate inference network가 전혀 필요 없다.

2. Introduction

과거 딥러닝은 계층적이고 풍부한 모델을 학습하여 이미지, 음성, 자연어 특징과 같은 데이터를 인공지능을 활용하여 확률로 나타내어 주었다.
-> 이러한 딥러닝의 성공에는 backpropagation, dropout 등의 알고리즘과 선형함수를 사용한 가중치 적용에 기초하여 고차원의 특성, 많은 입력을 가지는 구별하는 모델을 사용하였다.
-> 이러한 과정에서 Maximum Likelihood Estimation(MLE, 최대우도추정)과 같은 전략들에서 나오는 많은 확률론적 계산의 어려움과 생성적인 맥락에서 선형단위의 이점을 활용하는 것의 어려움 때문에 Deep generative model이 영향을 크게 받지 못했다.

: 따라서 본 논문에서는 이러한 어려움을 피하는 새로운 생성모델을 소개하였다.



위조지폐와 경찰을 예로 설명한 GAN 모델 (Goodfellow, Ian el al., 2014)

adversarial nets 프레임워크에서 생성모델 G는 판별모델 D를 속이도록 세팅되고 판별모델 D는 sample이 생성모델 G가 모델링한 분포에서 나온 것인지 실제 데이터 분포에서 나온 것인지 결정하는 법을 학습한다. 이러한 경쟁구도는 두 모델이 각각의 목적을 달성시키기 위해 스스로를 개선하도록 한다.

- 판별모델 D는 sample이 원본인지 생성모델이 만든 데이터인지 그 판별해준다.
- 위와 같이 생성모델 G는 위조지폐를 만드는 위조지폐범과 비슷하고 판별모델 D는 이 위조지폐를 검거하려는 경찰과 비슷하다.
- GAN은 결국 위조지폐범(G)이 경찰(D)를 속이기 위해 위조지폐를 만들어내고 경찰은 이를 진짜 지폐를 1로, 가짜 지폐를 0으로 판별해낸다. 이 과정에서 위조지폐범은 더욱 정교한 위조지폐를 만들어내고 경찰 역시 훈련할수록 판별능력이 높아져 50%로 수렴하게 된다. 즉, 경찰이 위조지폐와 실제 화폐를 구분할 수 없는 상태에 이르도록 한다.

3. Adversarial nets

adversarial modeling 프레임워크는 모델이 둘 다 다층 퍼셉트론일 때 가장 간단히 적용할 수 있다.

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

위는 adversarial nets의 목적함수이다. D는 실제 데이터와 생성된 데이터에 대해 적절한 label을 할당하도록 하는 확률을 최대화한다. 또한, log(1-D(G(z)))를 최소화 하도록 G를 동시에 훈련시킨다.

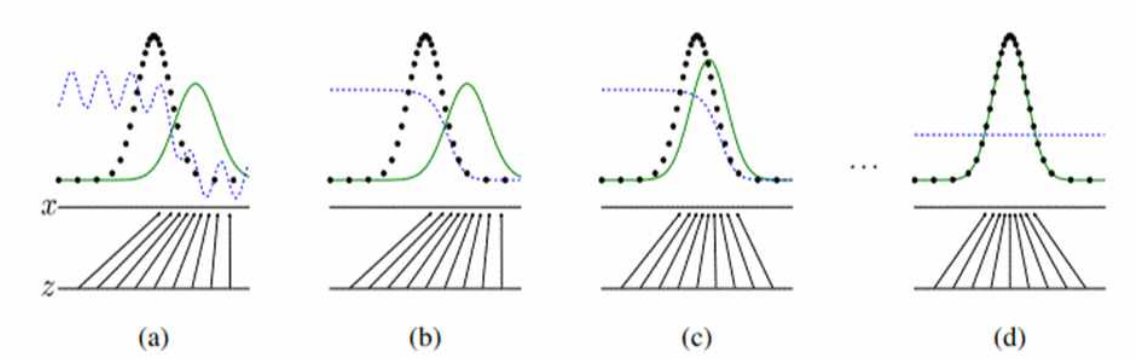
- 첫 번째 항 : 실제 데이터 x를 discriminator 모델에 넣었을 때 나오는 결과(x가 실제 데이터 분포에서 나왔을 확률)에 log를 취해 얻는 기댓값
- 두 번째 항 : fake 데이터 z를 generator 모델에 넣은 결과를 discriminator 모델에 넣었을 때 결과에 log(1-결과)를 취해 얻는 기댓값

이를 각 모델의 입장에서 한번 살펴보도록 한다.

- 판별모델(D) : G가 생성한 가짜 데이터가 들어오면 0을 출력하고 실제 데이터가 들어오면 1을 출력해야 한다. 따라서 목적함수의 각 항이 0의 값이 되도록 만들어야 하고, V(D,G)를 최대화하는 값은 0이 된다.
- 생성모델(G) : D가 샘플이 실제 데이터 분포에서 나온 것으로 판단하게 만들어야 하기 때문에 D(G(z))의 값이 1이 되도록 만들어야 한다. 즉, V(D,G)가 음의 무한대 값(log0)으로 가도록 만들어야 한다.

이렇게 판별모델 D 입장에서는 V(D,G)를 최대화 시키려고, 생성모델 G 입장에서는 V(D,G)를 최소화 시키려고 하는 것을 two-player minmax game으로 비유하여 설명하였다.

학습시키는 과정에서 inner loop에서 D를 최적화하는 것은 계산량이 많고 유한한 데이터 셋에서는 과적합을 초래한다. 따라서, D의 가중치 계산을 줄이기 위해 D를 최적화하는 k step과 G를 최적화하는 1 step을 번갈아 수행한다. 이를 통해 D는 최적의 솔루션에 가깝게 유지가 되었고, 따라서 G도 충분히 천천히 변화했다.
또한, G 모델이 초반에 형편없는 데이터를 만들기 때문에 초반에는 D의 판단력이 우세하다. 이 경우 log(1 − D(G(z)))가 포화상태가 되므로 log(1 − D(G(z)))를 최소화 시키는 것이 아니라 log D(G(z))를 최대화하는 방법을 사용할 수도 있다고 한다. G의 성능이 형편없을 때에는 log(1 − D(G(z)))의 기울기를 계산했을 때 너무 작은 값이 나오므로 학습이 느리기 때문이다.



(파란색 점선 : D가 모델링하는 분포, 녹색 실선 : G가 모델링하는 분포, 검은 점선 : 실제 데이터 생성 분포)
위의 그림 순서에 맞게 설명해보도록 한다.
(a) 학습 초기 상태 - 학습 초기에는 실제와 G의 분포 차이가 심하다. D 역시 성능이 높진 않아 보인다.
(b) D의 분포가 전에 비해 분명하게 데이터를 판별하고 있음을 확인할 수 있다.
(c) 어느 정도 D의 학습이 이루어지면, G는 실제 데이터의 분포를 모사하여 D가 판별하기 힘들게 학습한다.
(d) 이 과정을 반복하여 실제 데이터 분포와 G에 의해 생성된 분포가 거의 비슷해져 D는 1/2의 값에 가까운 확률을 보여준다.

4. Theoretical Results

G는 확률 분포를 $z \sim p_z$일 때 얻은 표본 G(z)의 분포로 암묵적으로 정의한다. 따라서, 우리는 충분한 양과 훈련 시간이 주어진다면 알고리즘 1이 p_{data}의 좋은 추정치로 수렴되기를 바란다. 이 섹션의 결과는 non-parametric하게 수행된다. 예를 들어 확률 밀도 함수의 공간에서의 수렴을 연구하여 무한 용량을 가진 모델을 나타낸다.

Algorithm 1:
적대신경망의 훈련을 위한 미니배치 확률적 경사 하강법. 하이퍼파라미터 k는 구별모델에 적용하기 위한 스텝을 나타낸다. 우리는 실험에서 가장 저렴한 k=1을 사용하였다.

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.

1. noise prior $p_g(z)$로부터 noise 시킨 m개의 미니배치 $\{z^{(1)},\cdots,z^{(m)}\}$ 샘플을 만든다.
2. 데이터 생성분포 $p_{data}(x)$로부터 미니배치 m개의 샘플 $\{x^{(1)},\cdots,x^{(m)}\}$을 평가한다.
3. 확률적 경사 상승법을 이용하여 D를 업데이트한다.

k번의 스텝이후

k-1. noise prior $p_g(z)$로부터 noise 시킨 m개의 미니배치 $\{z^{(1)},\cdots,z^{(m)}\}$샘플을 만든다.
k-2. 확률적 경사 하강법을 이용하여 G를 업데이트한다.

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

Proposition 1. G가 고정 되어 있을 때 최적의 D는 아래와 같다.

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

주어진 어떤 G에 대해 D의 훈련법은 V(G,D)를 최대화 시키는 것이므로 이에 대한 증명은 아래와 같다.

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

임의의 $(a,b) \in R^2 \setminus \{0,0\}$ 에 대하여 함수 y → a log(y) + b log(1 - y) 는 $\frac{a}{a+b}$

로 [0,1]에서 최대치를 달성한다. (위 식을 미분할 경우에 y는 $\frac{a}{a+b}$에서 최대를 가짐)

D에 대한 훈련 목표는 조건부 확률 P(Y = y|x)를 추정하기 위한 로그 우도를 최대화하는 것으로 해석될 수 있다. 여기서 Y는 x가 p_{data}(with y = 1) 또는 p_g(with y = 0)로부터 오는지를 나타낸다. 방정식 1에서의 minimax 게임을 다음과 같이 재구성할 수 있다.

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

Theorem 1. 가상 훈련 기준 C(G)의 전역 최솟값은 $p_g = p_{data}$인 경우에만 달성된다. 이 시점에서 C(G)는 다음과 같은 값을 달성합니다. - log 4.

증명 : (방정식 2에 따르면) $p_g = p_{data}$에 대해 $D_G^*(x) = \frac{1}{2}$이다. 따라서 $D_G^*(x) = \frac{1}{2}$에서의 방정식 4를 보면 $C(G) = \log\frac{1}{2} + \log\frac{1}{2} = -\log 4$인 것을 확인할 수 있다. 이 식이 $p_g = p_{data}$에 대해 최적의 C(G)값인지 다음 식을 확인하시오.

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

그리고 이 식을 에서 $C(G) = V(D_G^*, G)$에서 빼면 다음과 같은 값을 얻을 수 있다.

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

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

두 분포 사이의 Jensen-Shannon 발산은 항상 음이 아니고 그것들이 같을 때만 0이기 때문에, 우리는 $C^* = -\log(4)$가 C(G)의 전역 최솟값이며, 유일한 값은 $p_g = p_{data}$이다. 이는 생성모델이 완벽하게 데이터를 생성했다고 본다.

4.2 Convergence of Algorithm 1

Proposition 2. G와 D가 충분한 능력을 가지고 있고 알고리즘 1의 각 단계에서 D는 최적의 G에 도달할 수 있으며 p_g는 정책을 개선하기 위해 갱신된다.

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

5. Experiments

- MNIST, the Toronto Face Database (TFD), and CIFAR-10 데이터셋을 이용하여 적대 신경망을 훈련시켰다.

- G는 rectifier linear, sigmoid 활성화 함수를 섞어서, D는 maxout 을 사용하였다.
- D에는 Dropout이 적용되었다.
- 이론상으로는 G의 중간 계층에서 드롭아웃 및 기타 노이즈를 사용할 수 있지만, 우리는 노이즈를 G의 맨 아래 계층에서 입력으로만 사용했다.

Model	MNIST	TFD
DBN [3]	138 ± 2	1909 ± 66
Stacked CAE [3]	121 ± 1.6	$\mathbf{2110 \pm 50}$
Deep GSN [6]	214 ± 1.1	1890 ± 29
Adversarial nets	$\mathbf{225 \pm 2}$	$\mathbf{2057 \pm 26}$

Gaussian Parzen window를 G에 의해 생성된 샘플들에 fitting하고 이렇게 추정된 분포 하에 얻어진 log-likelihood를 확인함으로써 저자들은 p_g하에서 test set 데이터의 확률을 추정하였다. 해당 방법을 옳은 평가 척도라고 할 수 없지만 이전 모델과 비교했을 때, 경쟁력을 갖추고 있고, 잠재력을 보여준다.
이렇게 G가 생성해낸 샘플이 기존 방법으로 만든 샘플보다 좋다고 주장할 수 없지만, 더 나은 생성 모델과 경쟁할 수 있다고 생각하며, adversarial framework의 잠재력을 강조한다.

6. Advantages and disadvantages

Disadvantages
- 일차적으로 $p_g(x)$의 명시적인 표현이 없다.
- D와 G가 균형을 잘 맞춰 성능이 향상되어야 한다.(G가 너무 많은 z 값을 x의 동일한 값으로 분해하여 다양성 갖추지 못하는 "헬베티카 시나리오"를 피하기 위해 D를 업데이트하지 않고 G를 너무 많이 훈련해서는 안 된다)

Advantages
- Markov chains 이 필요하지 않다.
- 기울기 조정을 위해 역전파만 사용된다.
- 학습 중에 inference가 필요하지 않으며 다양한 기능이 모델에 통합될 수 있다.
- Markov chains를 기반으로 하는 방법보다 선명한 이미지를 얻을 수 있다.

7. Conclusion and future work

- 클래스 레이블을 추가하여 Conditional generative model(조건부 생성 모델) p(x|c)을 얻을 수 있다.
- 준지도학습 : discriminator에 의해 얻어지는 중간단계 feature들은 레이블이 일부만 있는 데이터를 사용할 수 있을 때, discriminator의 성능을 향상시킬 수 있다.
- 효율성 향상 : G와 D를 조정하는 더 나은 방법을 설명하거나 학습 중에 z 샘플에 더 나은 분포를 결정함으로써 학습의 속도를 높일 수 있다.