💚

Generative Adversarial Nets

기간	@03/19/2025 → 03/25/2025
주차	3주차
논문	Generative Adversarial Nets.pdf
상태	완료
예습/복습	예습과제
참고 자료	참고 블로그1 참고 블로그2

7. Conclusions and future work(결론 및 발전 가능성)

1. 조건부 생성 모델(Conditional Generative Model)
2. 근사 추론(Auxiliary Network for Approximate Inference)
3. 조건부 확률 모델링(Generalized Conditional Probability Modeling)
4. 반지도 학습(Semi-Supervised Learning)
5. 학습 효율 개선(Training Efficiency Improvements)

▼ [관련 개념 공부]

13장 생성 모델(p.682~723)

13.1.1 생성 모델 개념

- **생성 모델(generative model)** : 주어진 데이터를 학습하여 데이터 분포를 따르는 유사한 데이터를 생성하는 모델.
  - 기존 합성곱 신경망) 입력 이미지(x) 있으면 그에 따른 정답(y) 찾는 것.

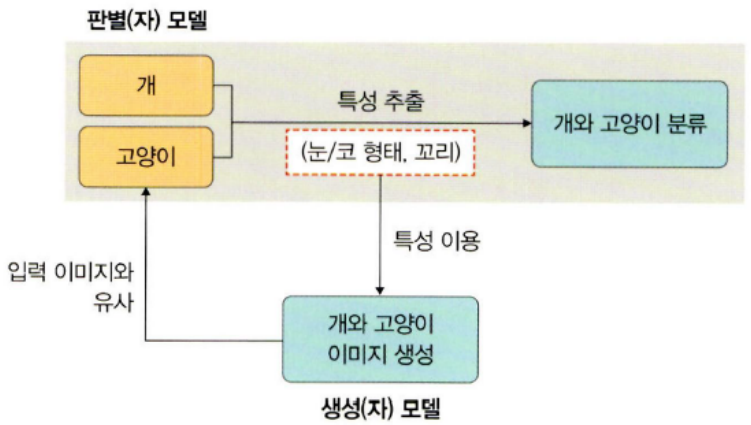
    ⇒ **'판별(자) 모델(discriminative model)'**

    ⇒ 판별자 모델에서는 이미지를 정확히 분류하고자 해당 이미지를 **대표하는 특성들을 잘 찾는 것**을 목표로 함.
  - 판별자 모델에서 추출한 특성들의 조합 이용하여 새로운 이미지를 생성할 수 있음.

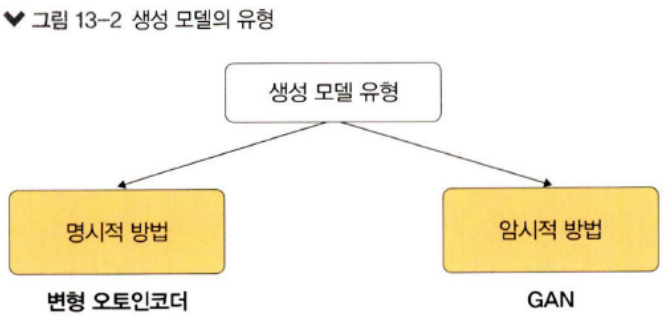
    ⇒ **'생성(자) 모델(generative model)'**

    ⇒ 입력 이미지에 대한 데이터 분포 $p(x)$ 학습하여 새로운 이미지(최대한 입력 이미지와 유사한 이미지) 생성하는 것을 목표로 함.

그림 13-1 생성 모델



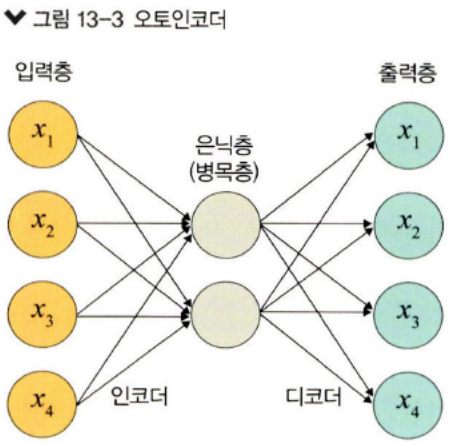
13.1.2 생성 모델의 유형



- 명시적 방법(explicit density)
  - 확률 변수 p(x) 정의하여 사용함.
  - 대표적 모델 : **변형 오토인코더 모델**
    - 이미지의 잠재 공간(latent space)에서 샘플링하여 완전히 새로운 이미지나 기존 이미지 변형하는 방식으로 학습.
- 암시적 방법(implicit density)
  - 확률 변수 p(x)에 대한 정의 없이 p(x)를 샘플링 하여 사용함.
  - 대표적 모델 : **GAN(Generative Adversarial Network)**
    - 생성자와 판별자가 서로 경쟁하면서 가짜 이미지를 진짜 이미지와 최대한 비슷하게 만들도록 학습.

13.2.1 오토인코더란?

- **오토인코더**
  - 단순히 입력을 출력으로 복사하는 신경망.
  - 은닉층(병목층)의 노드 수가 입력 값보다 적음. ⇒ 입력과 출력이 동일한 이미지
  - 왜 입력을 출력으로 복사할까?
    - 은닉층 때문. 오토 인코더의 병목층은 입력과 출력의 뉴런보다 훨씬 적은데, **적은 수의 병목층 뉴런으로 데이터 가장 잘 표현할 수 있는 방법**이 오코인코더임.



[4가지 주요 부분]

1. 인코더: 인지 네트워크(recognitioni network), 특성에 대한 학습 수행하는 부분.
2. 병목층(은닉층): 모델 뉴런 개수가 최소인 계층. 차원이 가장 낮은 입력 데이터의 압축 표현 포함됨.
3. 디코더: 생성 네트워크(generative network), 병목층에서 압축된 데이터를 원래대로 재구성(reconstruction)하는 역할. 최대한 입력에 가까운 출력을 생성.
4. 손실 재구성: 입력층, 출력층의 뉴런 개수 동일하다는 것만 제외하면 일반적인 다층 퍼셉트론(MLP)와 구조가 동일함. 압축된 입력을 출력층에서 재구성, 손실 함수는 입력과 출력(인코더와 디코더) 차이 가지고 계산함.

[수학적 접근]

- 입력 x와 출력 y는 같은 차원(R^d)에 존재한다는 가정하에 입력 데이터를 인코더 네트워크에 통과시켜 잠재 벡터 z 얻음.

$$z = h(x)$$

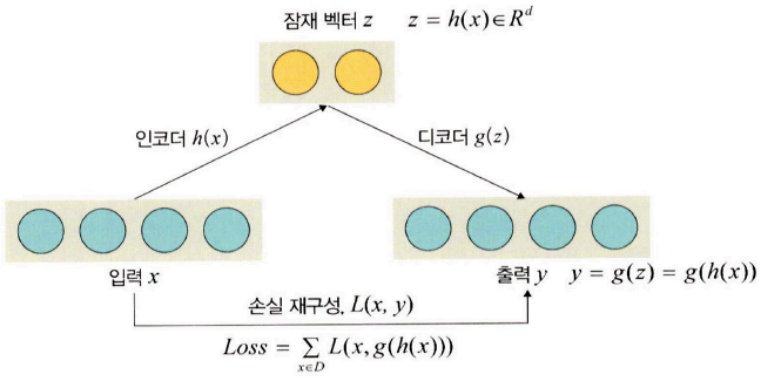
- 압축된 z 벡터에서 입력 데이터와 크기 같은 출력 값 계산함.

$$y = g(z) = g(h(x))$$

- 이때 손실(loss) 값은 입력 값 x와 디코더 통과한 y 값 차이로 계산함.

$$Loss = \sum_{x \in D} L(x, g(h(x))) = \| x - y \|^2$$

그림 13-4 오토인코더의 인코더와 디코더

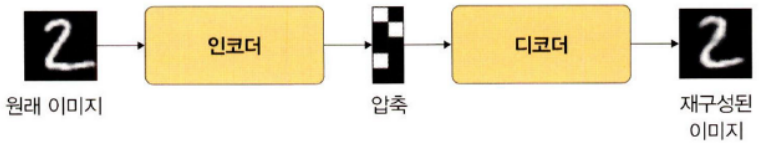


- 오토인코더가 중요한 이유
  1. 데이터 압축 : 메모리 측면에서 상당한 장점.
  2. 차원의 저주(curse of dimensionality) 예방 : 특성 개수 줄여 주기 때문에 데이터 차원 감소하여 차원의 저주 피할 수 있음.
  3. 특성 추출 : 비지도 학습, 자동으로 중요한 특성 찾아 줌.

13.2.2 변형 오토인코더

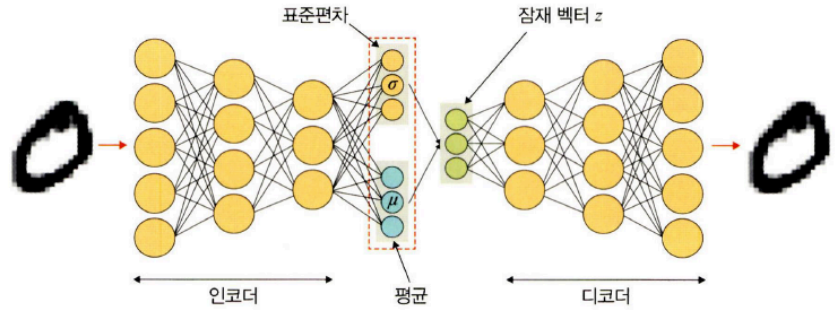
- 변형 오토인코더(variational autoencoder)
  - 오토인코더) 입력(숫자 2) → 인코더 → 압축(차원 축소) → 디코더 → 출력(숫자 2)

그림 13-9 오토인코더 실행 과정



    - 차원 줄이는 것이 목표, 새롭게 생성된 데이터의 확률 분포에는 관심이 없음.
  - 변형 오토인코더) 표준편차, 평균 이용하여 확률 분포 만들고, 거기에서 샘플링하여 디코더 통과시킨 후 새로운 데이터 만들어 냄.

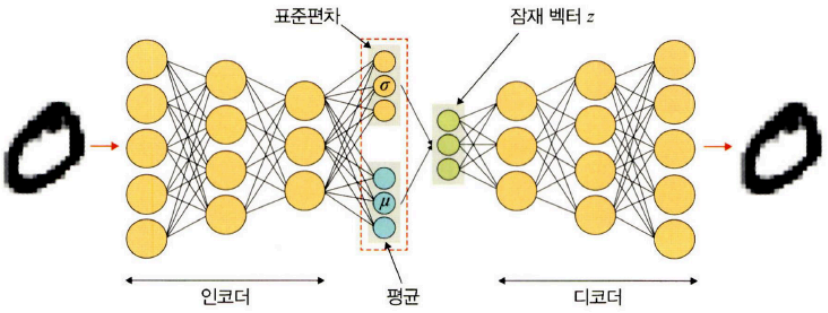
❯ 그림 13-10 변형 오토인코더 실행 과정



- 입력 데이터와 조금 다른 출력 데이터 만들어 내고, 이때 z라는 가우시안 분포 이용함.
- z 분포에서 벡터를 랜덤하게 샘플링하고 이 분포의 오차를 이용하여 입력 데이터와 유사한 다양한 데이터 만들어 냄.

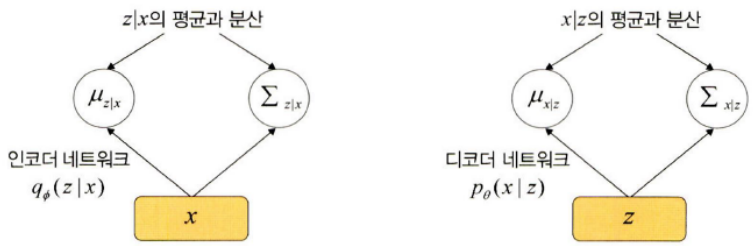
- 인코더&디코더 네트워크

❯ 그림 13-10 변형 오토인코더 실행 과정



  - $q_\phi(z|x)$: x 입력받아 잠재 벡터 z와 대응되는 평균과 분산 구하는 네트워크, 즉 인코더 네트워크.

❯ 그림 13-11 변형 오토인코더의 인코더와 디코더



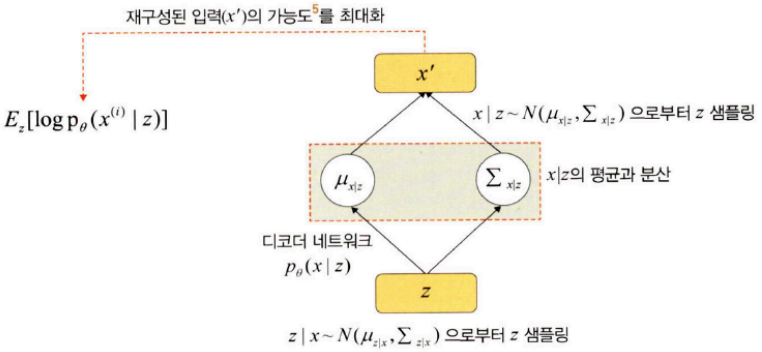
1. 입력 x를 인코더 네크워크 $q_\phi(z|x)$에 보내 $(\mu_{z|x}, \sum_{z|x})$ 출력하고, 이를 이용하여 다음 수식 (2)항에 대한 값 구함.

$$L(x^{(i)},\theta,\phi) = \underbrace{E_z[\log p_\theta(x^{(i)} \mid z)]}_{①} - \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \| p_\theta(z))}_{②}$$

2. $(\mu_{z|x}, \sum_{z|x})$의 가우시안 분포에서 z 샘플링함.

$$z \leftarrow z|x \sim N(\mu_{z|x},\ \Sigma_{z|x})$$

- $p_\theta(x|z)$: z 입력받아 x와 대응되는 평균과 분산 구하는 네트워크, 즉 디코더 네트워크.



1. 샘플링한 z를 디코더 네트워크 $p_\theta(x|z)$에 보내 $(\mu_{z|x}, \sum_{z|x})$ 출력한 후, 이를 이용하여 (1)항의 값 구함.
2. $(\mu_{x|z}, \sum_{x|z})$의 가우시안 분포에서 z 샘플링하고, x' (x와 유사한 이미지) 구함.

$$x' \leftarrow x|z \sim N(\mu_{x|z},\ \Sigma_{x|z})$$

3. 역전파 이용하여 $L(x^{(i)},\theta,\phi)$의 값이 높아지는 방향으로 기울기 업데이트. 가능도(likelihood) 증가하는 방향으로 파라미터 θ와 ϕ 업데이트.

- 인코더와 디코더에서 사용된 수식

$$\underbrace{E_z[\log p_\theta(x^{(i)} \mid z)]}_{①} - \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \| p_\theta(z))}_{②}$$

1. z 주어졌을 때 x' 표현하기 위한 확률밀도 함수, 디코더 네트워크.

a. 디코더의 가능도 클수록 θ가 그 데이터 잘 표현했다고 해석.

b. (1)항의 값이 **클수록** 모델 가능도 커짐.

2. x에서 z를 표현하는 확률밀도 함수, 인코더 네트워크와 가우시안 분포가 얼마나 유사한지 나타냄.

a. 유사한 정도 높을수록 쿨백-라이블러 발산(D_{KL})(두 확률 분포 차이 계산하는 데 사용) 낮은 값 → 가능도가 최대화됨.

b. (2)항 **작을수록** 모델 가능도 커짐.

13.3.1 GAN 동작 원리

- **GAN(Generative Adversarial Network) - 적대적 생성 신경망**
  - 경찰과 위조지폐범 사이의 게임에 비유

    - 경찰 : 진짜 지폐와 위조지폐 구분하는 **판별자**.

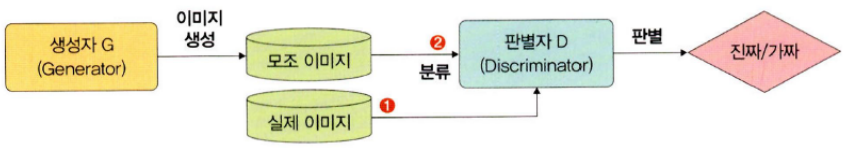
      ⇒ 진짜와 가짜 구별함.

    - 위조지폐범 : 위조지폐 생성하는 **생성자**.

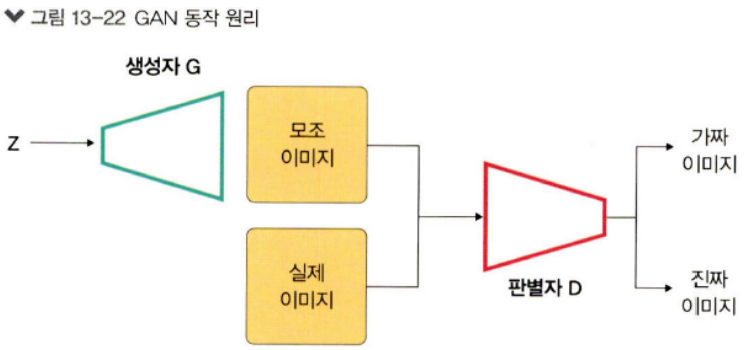
      ⇒ 최대한 진짜와 비슷한 데이터 생성하려 함.

  - 학습 과정
    - 판별자 먼저 학습시킨 후 생성자를 학습시키는 과정 반복함.

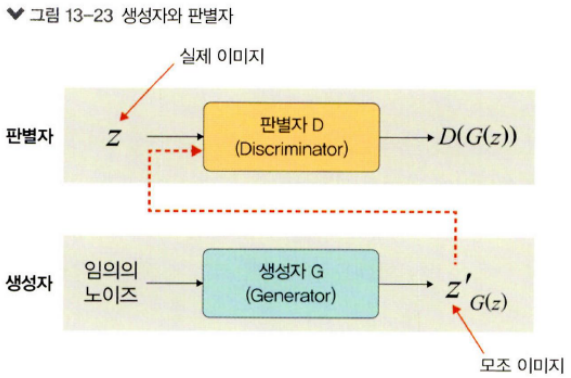
❯ 그림 13-21 적대적 생성 신경망 학습 과정



        - 2단계로 진행되는 판별자 학습
            - **실제 이미지** 입력해서 네트워크(신경망)가 해당 이미지를 **진짜**로 분류하도록 학습.
            - 생성자가 생성한 **모조 이미지** 입력해서 해당 이미지를 **가짜**로 분류하도록 학습.
        - 이러한 학습 과정 반복하면 판별자와 생성자가 서로 적대적인 경쟁자로 인식하여 모두 발전하게 됨. ⇒ 생성자는 진짜 이미지에 완벽히 가까울 정도의 유사한 모조 이미지 생성, 이에 따라 판별자는 실제 이미지와 모조 이미지 구분할 수 없게 됨.
            - 생성자는 분류에 성공할 확률 ↓ 판별자는 분류에 성공할 확률 ↑
- **GAN 동작 원리**
    - 생성자(**G**enerator) & 판별자(**D**iscriminator) 네트워크 2개로 구성.



    - **판별자** D
        - 주어진 입력 이미지가 진짜 이미지인지 가짜 이미지인지 구별함.
        - 이미지 x가 입력으로 주어졌을 때 판별자 D의 출력에 해당하는 $D(x)$가 진짜 이미지일 확률 반환.
    - **생성자** G
        - 판별자 D가 진짜인지 가짜인지 구별할 수 없을 만큼 진짜와 같은 모조 이미지를 노이즈 데이터 사용하여 만들어 냄.



- 실제 이미지 z가 입력일 때 판별자는 이를 학습하고, 생성자는 임의의 노이즈 데이터 사용하여 모조 이미지 $z'(G(z))$ 생성함.
- 이를 다시 판별자 D의 입력으로 주면 판별자는 $G(z)$가 실제 이미지일 확률 반환.

- **GAN 손실 함수**

$$\min_G \max_D V(D,G) = E_{x\sim Pdata}(x)[\log D(x)] + E_{z\sim P_z}(z)[\log(1-D(G(z)))]$$

  - $x\sim P_{data}(x)$: 실제 데이터에 대한 확률 분포에서 샘플링한 데이터
  - $z\sim P_z(z)$: 가우시안 분포 사용하는 임의의 노이즈에서 샘플링한 데이터
  - $D(x)$: 판별자 $D(x)$가 1에 가까우면 진짜 데이터로 0에 가까우면 가짜 데이터로 판단, 0이면 가짜를 의미.
  - $D(G(z))$: 생성자 G가 생성한 이미지 $G(z)$가 1에 가까우면 진짜 데이터로, 0에 가까우면 가짜 데이터로 판단.
  - 판별자 D는 실제 이미지 x를 입력받을 경우 $D(x)$를 1로 예측, 생성자가 잠재 벡터에서 생성한 모조 이미지 $G(z)$ 입력받을 경우 $D(G(z))$를 0으로 예측함.

    ⇒ 판별자가 모조 이미지 $G(z)$ 입력 받으면 1로 예측하도록 하는 것이 목표.

$$\max_D \log(D(x)) + \log(1-D(G(z)))$$

    - 판별자 D는 다음 식의 최댓값으로 파라미터 업데이트하는 것을 목표.
    - **판별자 입장)** $D(x)$=1, $D(G(z))$=0이 최상의 결과(진짜 1, 가짜 0)
      - $log(D(x))$ & $log(1-D(G(z)))$ 모두 최대가 되어야 함. → 모두 1이 되어야 모조 이미지 가짜라고 분류.

$$\min_G \log(1 - D(G(z)))$$

- 생성자 G는 다음 식의 최솟값으로 파라미터 업데이트하는 것을 목표.
- **생성자 입장)** D(G(z))=1이 최상의 결과(판별자가 가짜 이미지 1로 출력한 경우)

+) GAN 학습을 위해서는 판별자&생성자 번갈아 가며 업데이트. 서로의 파라미터 업데이트할 때는 나머지 파라미터 고정해야 함.

0. Abstract

- 적대적(Adversarial) 과정을 통해 생성 모델을 산출하는 새로운 프레임 워크 제안.
  - 이 과정에서 2개의 모델을 동시에 학습시킴.
    1. 데이터 분포를 학습하는 **생성 모델(G, Gernerator)**
    2. 샘플이 실제 훈련 데이터에서 온 것인지, 생성 모델에서 온 것인지 판별하는 **판별 모델(D, Discriminator)**
  - 이 프레임 워크를 minimax two-player game으로 표현.
  - 모델의 함수 공간이 충분히 크다면, 최적의 해는 G가 실제 훈련 데이터 분포를 완전히 복원(복사)하고, D가 모든 데이터에 대해 $\frac{1}{2}$를 출력하는 상태에 도달하는 것. (= 실제 데이터와 G가 생성해내는 데이터 간의 판별이 어려워진다는 뜻)
- 생성 모델과 판별 모델이 모두 다층 퍼셉트론(MLP)으로 정의될 경우, 전체 시스템은 **역전파만으로** 학습 가능.
  - 학습 또는 샘플 생성 과정에서 마르코프 체인(Markov Chains)이나 근사 추론 네트워크(Unrolled Approximate Inference Networks) 필요 X. → 실험을 통해 이 프레임 워크의 가능성 입증함.

1. Introduction

논문이 다루는 분야

- 딥러닝 활용한 생성 모델(Generative Model) 연구.

- 특히, 자연 이미지, 음성, 언어 데이터 등 **다양한 데이터에 대한 (모집단에 근사하는) 확률 분포를 학습할 수 있는 복잡하고 계층적인 모델 발견**하는 것이 목표.

해당 task에서 기존 연구 한계점

- 주목할 만한 성공 사례들 대부분 판별 모델(Discriminative Models)에 집중되어 있었음.
  - 고차원의 방대한 정제된 데이터를 특정 클래스 레이블에 1:1로 mapping하여 구분하는 모델 사용.

    ⇒ well-behaved gradient 갖는 선형 활성화 함수들을 사용한 역전파(Backpropagation) 및 드롭아웃(Dropout) 알고리즘 활용한 학습에 의해 가능했음.
- 반면, 생성 모델(Generative Models)의 연구는 상대적으로 좋은 성과 내지 못함.
  - **최대우도 추정(Maximum Likelihood Estimation, MLE)** 및 관련 기법을 사용할 때 발생하는 **계산적으로 다루기 어려운 확률적 연산** 때문.
  - 또한, 생성 모델에서 선형 활성화 함수의 장점을 충분히 활용하기 어려움.

논문의 contributions

- 앞선 문제들을 해결할 수 있는 **새로운 생성 모델 학습 절차** 제안.
- **Adversarial Nets 프레임 워크**
  - 생성 모델과 판별 모델을 서로 대립하는 관계로 설정.
  - 판별 모델 → 입력 데이터가 실제 데이터 분포에서 온 것인지, 생성 모델에서 생성된 것인지 판별하는 역할.
  - 이 개념은 위조 지폐 제조자와 위조 지폐 잡아내는 경찰의 관계에 비유 가능.
    - 생성 모델(위조 지폐 제조자) → 진짜 화폐와 구별되지 않는 위조 화폐 만들어 유통.
    - 판별 모델(경찰) → 위조 지폐 찾아내는 방법 학습.
    - 시간 지남에 따라, 양측이 서로의 전략 개선하며, 결국 위조 화폐가 원본과 완전히 구별되지 않는 수준에 도달하게 됨.
  - 다양한 모델 및 최적화 알고리즘 위한 학습 알고리즘 설계
    - 특히, 생성 모델이 **무작위 노이즈를 입력으로** 받아 다층 퍼셉트론 통해 **샘플을 생성**하는 경우 다룸.

        - 오직 **역전파와 드롭아웃** 알고리즘만을 사용하며, 생성 모델에서 샘플 추출 시 추론 과정 및 마르코프 체인 같은 복잡한 과정 필요 없음.

2. Related Work

- 기존 생성 모델은 주로 잠재 변수(latent variables) 포함하는 그래픽 모델 기반으로 함.
    1. 비지도 학습 위한 Undirected 그래픽 모델
        - 대표적인 예:
            - RBM(Restricted Boltzmann Machines)
            - DBM(Deep Boltzmann Machines)

            ⇒ 이러한 모델들은 정규화되지 않은 확률 함수의 곱 활용함.
        - 문제점:
            - 전체 확률 분포 구하기 위해 필요한 partition function 계산이 대부분의 경우 불가능함.
            - 이를 근사적으로 구하기 위해 Markov Chain Monte Carlo 방법이 필요, 이 방법은 **학습 속도가 느려지는** 문제 야기함.
    2. Hybrid 그래픽 모델
        - 대표적인 예:
            - Deep Belief Networks
                - 단일 undirected layer와 여러 개의 directed layer 결합한 형태
                - 일부 빠른 근사 학습 방법이 존재하지만, 여전히 계산 비용이 높음.
- 기타 대안 방법
    - log-likelihood 근사를 사용하지 않는 학습 방법
        - Score Matching : 학습된 확률 밀도가 정규화 상수까지 정확하게 주어진 경우에만 가능.
    - Noise-Contrastive Estimation (NCE)
        - 생성 모델 자체를 판별 모델로 활용하는 접근 방식.
        - 하지만, **고정된 노이즈 분포**를 사용하기 때문에 **학습 속도 매우 느려질 수 있음.**
    - 생성 샘플을 직접 출력하는 모델(Generative Machines)

        - 확률 분포를 명시적으로 정의하지 않고 **샘플 직접 생성**하는 방식.
        - 대표적인 연구:
            - GSN(Generative Stochastic Networks)
                - 일반화된 노이즈 제거 오토인코더(Generalized Denoising Autoencoders)의 확장 버전.
        - 문제점:
        - GSN은 여전히 마르코프 체인 기반 샘플링 필요함.

💡 **[기존 연구와의 차별점]**

- **Adversarial Nets**는 **마르코프 체인 전혀 사용하지 않으며**, 학습 과정에서 **오직 역전파만을 사용**함.
- 기존 생성 모델의 어려운 점이었던 piecewise linear units(선형 활성화 함수)의 이점을 충분히 활용할 수 있음.
    - **Auto-Encoding Variational Bayes**
    - **Stochastic Backpropagation**

    ➔ 생성 모델 학습하는 새로운 방법 제안했지만, **Adversarial Nets** 같이 판별 모델을 활용하는 방식은 아님.

3. 제안 방법론(Adversarial nets)

- **Adversarial Nets** 프레임 워크 → 모델이 모두 다층 퍼셉트론(MLP)인 경우 가장 적용하기 쉬운 방식.

기본 개념

1. **생성 모델(G, Generator)**
    - 입력으로 임의의 노이즈 샘플(random noise sample) z 받음.
    - 이를 Data Space로 매핑하는 함수 $G(z;\theta_g)$로 학습함.
    - G는 MLP로 표현된 미분 가능한 함수, 학습 가능한 파라미터 θ_g 가짐.
2. **판별 모델(D, Discriminator)**
    - 입력 데이터가 실제 데이터인지, 생성된 데이터인지 판별하는 함수 $D(x;\theta_d)$ 학습함.

- $D(x)$: 샘플이 실제 데이터일 확률을 출력하는 단일 스칼라 값 반환함.

학습 목표

- **판별 모델(D) 학습**:
  - 실제 데이터에 대해 높은 확률(1) 출력
  - 생성된 데이터에 대해 낮은 확률(0) 출력하도록 학습.

    ⇒ D는 올바른 정답 맞히는 확률을 최대화함.
- **생성 모델(G) 학습**:
  - 판별 모델 속여서 $D(G(z))$의 출력을 1로 만들도록 학습.

    ⇒ G는 D가 가짜 데이터 구별하지 못하게 만드는 방향으로 업데이트됨.

수식 정리

- 미니맥스(minimax) 게임 형태로 정리할 수 있음.

$$\underset{G}{min}\underset{D}{max}V(G,D) = \mathbb{E}_{x\sim pdata}[logD(x)] + \mathbb{E}_{z\sim p_z}[log(1-D(G(z)))]$$

- D는 이 값을 최대로 만들려고 하고, G는 이 값을 최소로 만들려고 함.
- 이론적으로, D가 최적의 상태에 도달하면,

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

  - D는 실제 데이터와 생성된 데이터의 상대적인 분포 반영하는 함수가 됨.
- 궁극적으로 G가 최적의 상태에 도달하면,

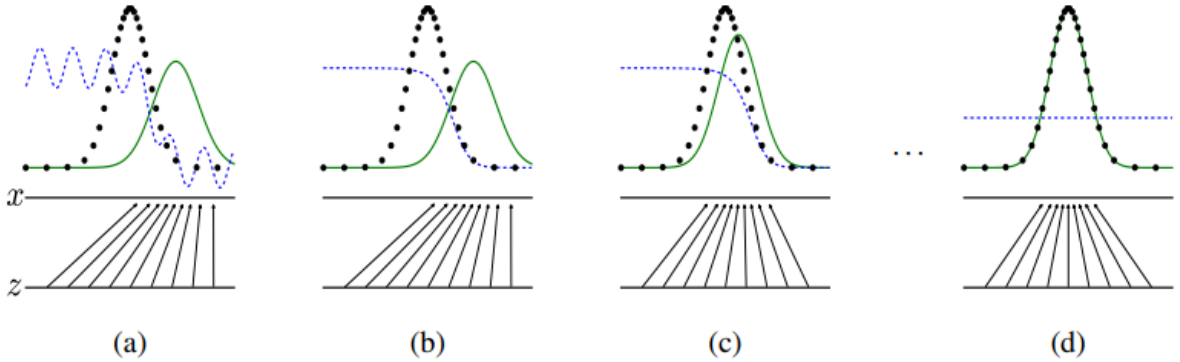
$$p_g(x) = p_{data}(x)$$

  - 생성된 데이터가 실제 데이터와 완전히 동일한 분포 가지게 됨.

실제 학습 방법

- 미니배치 확률적 경사 하강법(Stochastic Gradient Descent, SGD) 적용.
- D와 G의 학습을 교대로 진행.
- 초기에는 G가 너무 약하여 D가 쉽게 구별하기 때문에, **목표 함수를 변형**하여 강한 기울기(gradient) 받을 수 있도록 조정

$$\max_G \mathbb{E}_{z \sim p_z}[log D(G(z))]$$



- 검은색 점선: 실제 데이터 분포 / - 녹색 실선: 생성 모델 G가 만든 데이터 분포 / - 파란색 점선: 판별 모델 D(x)가 현재 판단하는 확률.

(a) 초기 상태: G와 D가 학습 시작할 때

- 초반에는 G가 제대로 학습하지 않았기 때문에, 생성된 데이터가 엉뚱한 곳에 위치.
- 판별자는 쉽게 실제/가짜 데이터 구별 가능.

(b) 판별자 D 학습

- D는 판별 능력을 최대한 끌어올림.
- $p_g(x)$가 작은 곳에서는 $D(x)$가 1에 가까운 값 가지게 됨.

(c) 생성자 G 학습

- G는 $D(x)$ 출력 보고, 자신이 만든 샘플이 더 실제처럼 보이도록 조정하려고 함.
- $p_g(x)$를 실제 데이터가 있는 위치로 이동시키면서 분포 조정함.
- 과정 반복되면서 $p_g(x)$가 점점 $p_{data}(x)$와 비슷해짐.

(d) 수렴 후 상태

- 최종적으로 G가 충분히 발전하면 $p_g(x)$는 $p_{data}(x)$와 거의 동일해짐.
- $D(x) = 0.5$ 되어 더 이상 구별할 수 없게 됨. ⇒ **완벽한 가짜 데이터 생성하는 수준**에 도달함!

4. Theoretical Results

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.

<1. D를 k번 학습 2. G를 1번 학습>하는 과정을 반복함. 이렇게 학습시키면 최적의 solution에 가까운 D, G가 만들어짐. 여기서 k=1로 설정함.

- G는 p_g를 $G(z)$의 분포로 정의 → 그래서 알고리즘 1이 $G(z)$를 p_{data}에 수렴하게 만들어주길 원함.
- 확률밀도 함수(probability density functions, pdf)에 수렴하는 것을 학습함으로써 무한한 능력을 가진 모델을 표현할 것.
  - 'Global Optimality of $p_g = p_{data}$'에서 minimax game이 $p_g = p_{data}$의 global optimum을 가지고 있음을 보여줄 것.
  - 'Convergence of Algorithm 1'에서 알고리즘 1이 value function $V(G, D)$ 최적화시켜 원하는 결과를 얻는 과정을 보여줄 것.

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

[Proposition 1]

- 생성 모델 G가 고정되었을 때, 최적의 판별 모델 D

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

  - **Proof**
    - D의 목표는 손실 함수 $V(G, D)$ 최대화하는 것.

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

- 이를 최적화하면, $D(x)$가 맨 위의 식과 같아짐.
- 앞서 세운 최적의 D는 올바른 식임을 알 수 있음.

💡**주목할 점**

D를 학습시키는 목적 → 조건부 확률 $P(Y = y|x)$의 log-확률(log-likelihood)을 최대화하기 위함으로 해석.

- 여기서 Y는 입력 데이터 x가 진짜 데이터(p_{data})에서 왔는지($y = 1$) 아니면 모방해서 만든 데이터(p_g)에서 왔는지($y = 0$) 나타내는 변수.

⇒ 그렇기 때문에 minimax game 식은 global minimum of the virtual training criterion $C(G)$로 다시 작성 가능.

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

[Theorem 1]

- 생성 모델 G가 최적으로 학습될 경우, $p_g(x) = p_{data}(x)$ 만족해야 하고, 이때 손실 함수는 $-log4$ 값을 가짐.
  - **Proof**
    - $p_{data}(x) = p_g(x)$면 $p_{data}(x)/(p_{data}(x) + p_g(x)) = D_G^* = 0.5$, $C(G)$에 대입하면,

$$C(G) = log\frac{1}{2} + log\frac{1}{2} = -log4$$

    - $C(G) = V(D*_G, G)$에서 -log(4)가 나오는 부분을 빼면,

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

- JSD 이용하여 표현,

$$C(G) = -log4 + 2 \cdot JSD(p_{data}||p_g)$$

- Jensen-Shannon Divergence(JSD) ⇒ 두 확률 분포 사이의 차이를 측정하는 값.
- $C(G) = -log4$ 가 $C(G)$의 minimum
- $C(G)$ 최소화하는 G는 결국 $p_g(x) = p_{data}(x)$가 되도록 학습됨.

4.2. Convergence of Algorithm 1

[Proposition 2]

- G와 D가 충분한 능력 가지고, 알고리즘 1의 각 단계에 있을 때, D는 G가 최적의 상태에 도달하게 만들어주고 p_g는 다음의 criterion 향상시키는 방향으로 업데이트 되어 p_{data}로 수렴함.

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

  - **Proof**
    - $V(G, D) = V(p_g, D)$를 p_g의 함수로 생각하면, $V(p_g, D)$ 는 p_g 에서 정점(최댓값의 도함수) 찍음.
    - 여기서 만약 $f(p_g) = \sup_{p\in D} f_D(p_g)$이고 $f_D(p_g)$가 모든 D 마다 p_g 에서 정점을 찍는다면, 대응하는 G가 주어진 최적의 D 에서 p_g에 대한 gradient - descent update 와 동일함.

      ⇒ 따라서 p_g에 대한 적은 수의 update 만으로도 $p_g \rightarrow p_{data}$ 수렴.
  - 알고리즘 1이 global optimal 찾아주는 것을 입증함.

5. 실험(Experiments)

Dataset

- MNIST (손글씨 숫자)
- Toronto Face Database (얼굴 이미지)
- CIFAR-10 (일반 이미지)

Baseline

- **생성 모델(G)**
  - 다층 퍼셉트론(MLP) 사용.
  - ReLU+시그모이드 활성화 함수 적용.
- **판별 모델(D)**
  - Maxout 활성화 함수 적용.
  - 드롭아웃(Dropout) 사용.
- **실험 방법**
  - 평가 방법:
    - Parzen Window 기반 로그 가능도(log-likelihood) 추정
    - G가 생성한 샘플을 가우시안 커널 기반 밀도 함수로 근사하여 테스트 데이터의 로그 가능도를 평가.
    - 기존 생성 모델(RBM, DBN, Stacked CAE, Deep GSN)과 비교.

결과

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	$\mathbf{225 \pm 2}$	$\mathbf{2057 \pm 26}$

- MNIST와 TFD 데이터에서 GAN이 가장 높은 성능을 보임.
- CIFAR-10에서도 기존 생성 모델과 유사한 성능을 기록.

a) MNIST b) TFD c) CIFAR-10 (완전 연결 모델) d) CIFAR-10 (컨볼루션 판별기, "디컨볼루션" 생성기)

- 다음은 학습 완료한 generative model에서 생성한 데이터의 일부
  - 가장 오른쪽 열은 바로 인접한 훈련 예제를 보여줌.
  - 기존 모델과 비교하여 꽤 경쟁력 있는 품질을 가짐.

6. Advantages and disadvantages

6.1. 장점 (Advantages)

1. **마르코프 체인 불필요**
   - 기존 RBM, DBM, GSN 등은 마르코프 체인 기반 샘플링이 필요하지만, GAN은 **순전파(Forward Propagation)만으로 샘플 생성 가능**.
2. **오직 역전파(Backpropagation)만으로 학습 가능**
   - 복잡한 추론 과정 없이 표준 신경망 학습 기법만으로 학습 가능.
3. **더 넓은 범위의 모델 구조 적용 가능**
   - 비선형 변환이 포함된 복잡한 모델도 학습 가능.
4. **Sharp한(명확한) 분포 표현 가능**

- 기존 마르코프 체인 기반 모델들은 확률 분포가 부드러워야 샘플링 가능하지만, GAN은 **매우 Sharp한 분포도 표현 가능**.

6.2. 단점 (Disadvantages)

1. **D와 G의 학습 균형 조절 필요**
   - G가 너무 빠르게 학습되면 D가 제대로 학습되지 않을 수 있음.
   - 즉, 학습이 **불안정(unstable)할 가능성**이 있음.
2. **$p(x)$를 직접 평가하기 어려움**
   - GAN은 확률 분포 $p_g(x)$를 명시적으로 정의하지 않음.
   - 따라서 샘플 품질을 정량적으로 평가하는 것이 어려움.
3. **Mode Collapse 문제**
   - G가 한정된 유형의 샘플만 생성하는 경우 발생.
   - 다양한 데이터를 생성하기 위해 학습 안정화 기법이 필요함.

	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

다양한 접근 방식으로 직면한 Challenges

7. Conclusions and future work(결론 및 발전 가능성)

- 본 논문에서는 **Adversarial Networks**를 활용한 새로운 생성 모델 학습 방법을 제안함.
- 실험 결과, 제안된 프레임워크는 기존의 생성 모델과 비교하여 **경쟁력 있는 샘플 품질**을 보여주었음.

⇒ 다양한 확장 가능성을 제공하며, 다음과 같은 연구 방향이 고려될 수 있음.

1️⃣ 조건부 생성 모델(Conditional Generative Model)

- 특정 조건 c를 주어, 이를 기반으로 데이터를 생성하는 모델 $p(x \mid c)$을 학습할 수 있음.
- 즉, 생성 모델 G와 판별 모델 D에 추가 입력 c를 제공하면 **조건부 생성(Conditional Generation)이 가능.**

  예) 클래스 레이블을 입력으로 받아 특정 숫자만 생성하는 GAN.

2️⃣ 근사 추론(Auxiliary Network for Approximate Inference)

- 생성된 샘플 x에서 노이즈 z를 역추정하는 **추론 네트워크(Inference Network) 학습 가능.**
- 이는 Wake-Sleep 알고리즘과 유사하지만, **생성 모델이 학습된 후 별도로 학습 가능**하다는 장점 존재.

3️⃣ 조건부 확률 모델링(Generalized Conditional Probability Modeling)

- 특정 인덱스 S를 기준으로, 부분적인 데이터 x_S를 주었을 때 나머지 $x_{\neg S}$를 생성하는 모델 학습 가능.

  예) 이미지의 일부가 가려진 경우, 보이지 않는 부분을 생성하는 GAN.
- 이는 MP-DBM(Multi-Prediction Deep Boltzmann Machines)을 확장한 확률적 버전으로 활용 가능.

4️⃣ 반지도 학습(Semi-Supervised Learning)

- **GAN의 판별 모델(D)의 피처(feature)를 활용하여, 지도 학습 성능을 향상**시킬 수 있음.

  예) 라벨이 적은 데이터셋에서 GAN을 활용한 데이터 표현 습(Feature Learning) 후, 이를 활용한 분류 모델 개선 가능.

5️⃣ 학습 효율 개선(Training Efficiency Improvements)

- **생성 모델(G)과 판별 모델(D)의 학습 균형을 맞추는 새로운 기법 연구**.
- 보다 **효율적인 샘플링 기법을 개발**하여 학습 속도를 가속화할 수 있음.

  예) GAN이 학습 초기에 더 나은 기울기 정보를 받을 수 있도록 샘플링 기법 개선.