[3주차] 논문리뷰

0. 교재 예습

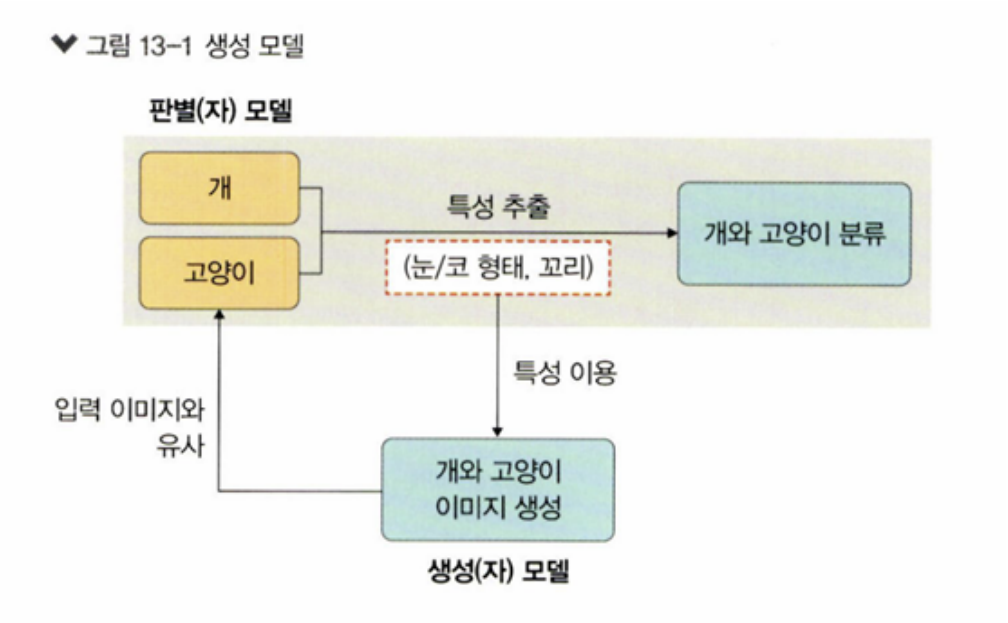
▼ 13.1 생성 모델이란?

주어진 데이터를 학습하여 데이터 분포를 따르는 유사한 데이터를 생성하는 모델.

13.1.1 생성 모델 개념

- 기존 합성곱 신경망 → 판별자 모델(discriminative model)

    - 정확한 분류가 목표 / 대표 특성 찾는 것이 중요
- 생성자 모델(generative model)
    - 판별자 모델 추출 특성들 조합으로 새로운 이미지 생성
    - 입력 이미지에 대한 데이터 분포 학습하여 유사 이미지 생성



13.1.2 생성 모델의 유형

1. 명시적 방법
    - 확률 변수 p(x)를 정의하고 구하는 모델
    - ex) 변형 오토인코더
        - 이미지의 잠재공간에서 샘플링하여 완전히 새로운 이미지나 시존 이미지를 변형
2. 암시적 방법
    - 확률 변수를 이용하지 않고 p(x)를 샘플링하여 사용
    - ex) GAN
        - 생성자와 판별자가 서로 경쟁하면서 가짜 이미지를 진짜 이미지와 비슷하게 만들도록 학습 진행

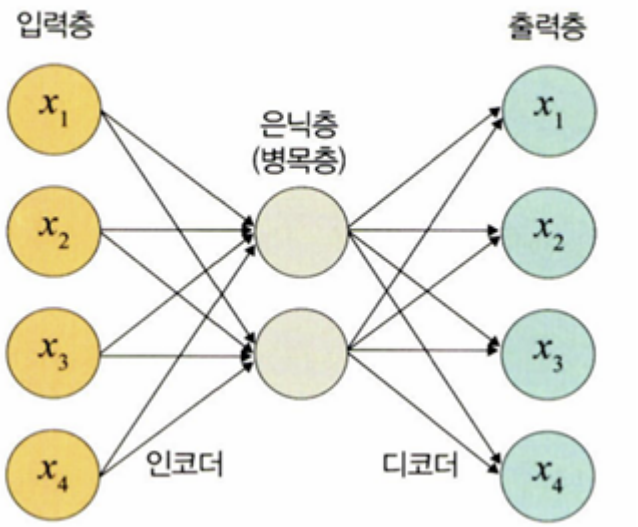
▼ 13.2 변형 오토인코더

- 변형 오토인코더 = 오토 인코더의 확장

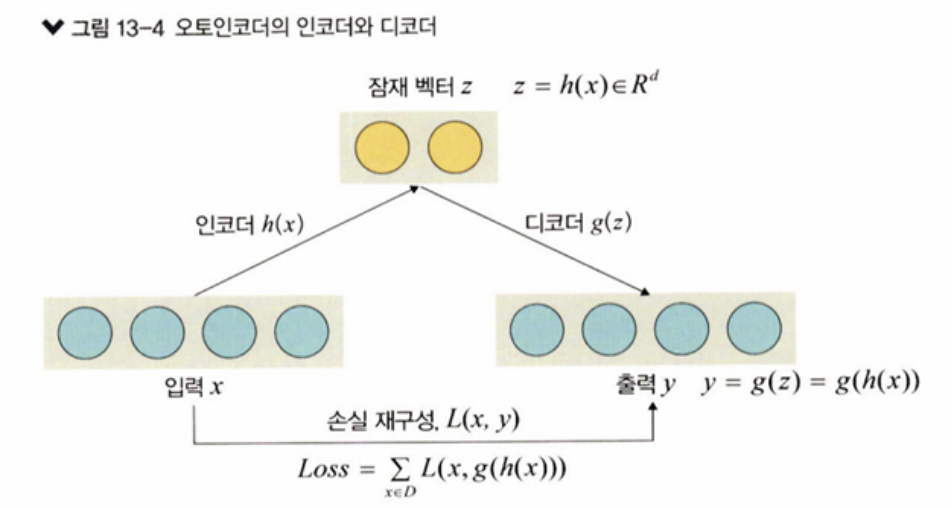
13.2.1 오토인코더란

- 입력을 출력으로 복사하는 신경망
- 은닉층(= 병목층) 노드 수가 입력, 출력 노드 수보다 적음
- 적은 수의 병목층 뉴런으로 데이터를 가장 잘 표현할 수 있는 방법
- 구성
  1. 인코더 = 인지 네트워크 : 특성에 대한 학습 수행
  2. 병목층 = 은닉층 : 모델 뉴런 수 최소. 입력데이터가 차원이 가장 낮게 압축 표현 됨
  3. 디코더 = 생성 네트워크 : 병목층에서 압축된 데이터를 원래대로 재구성. 최대한 입력에 가까운 출력 생성
  4. 손실 재구성(by 손실함수) : 입력과 출력의(인-디코더) 차이를 가지고 계산

그림 13-3 오토인코더



- 오토인코더 수학적 접근



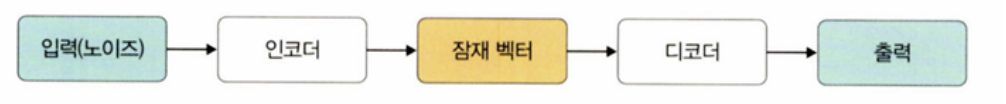
- x와 y는 같은 차원에 존재.
- 입력 : x
- 잠재벡터 : z=h(x) (인코더 네트워크 통과 결과)
- 출력 : y = g(z) = g(h(x))
- 손실(Loss) : x와 y의 2-norm

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

- 오토인코더가 중요한 이유
  1. 데이터 압축 : 메모리 측면 장점 - 용량 작고 품질 좋아짐

     (?용량이 작아지는 건 ok 품질은 왜 좋아지지?)
  2. 차원의 저주 예방 : 특성 개수를 줄여주므로

     (data sparse, 거리개념 무의미, 과적합, 계산량 증가 등의 문제)
  3. 특성 추출 : 비지도 학습으로 자동으로 중요한 특성 찾아줌
- 오토인코더 구현 과정 (코드 제외)

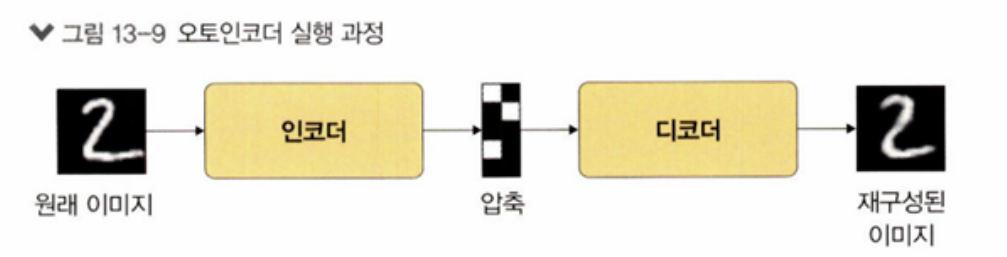


1. 라이브러리 호출

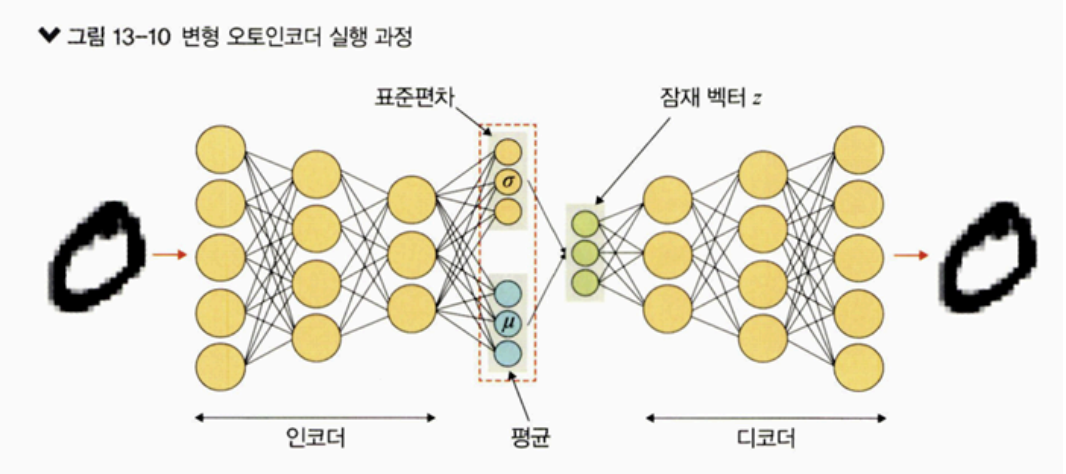
2. MNIST 데이터셋 호출 - 텐서 변경 - 데이터로더 전달
3. 인코더 및 디코더 네트워크 생성
4. 손실 함수(평균제곱오차, 이진 크로스 엔트로피)와
   옵티마이저(아담, RMSProp, adadelta) 지정
5. 모델 학습 함수, 테스트 함수 생성
6. 노이즈 데이터 생성
7. 이미지 시각화
8. 모델 학습

13.2.2 변형 오토인코더

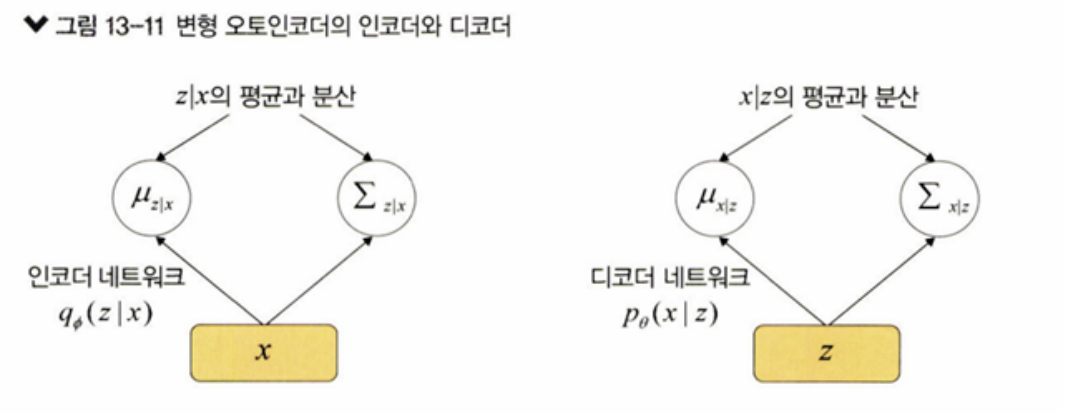
- 오토인코더
  → 동일한 이미지 출력이 목표
  → 차원 줄이는 게 목표 / 새 데이터 확률 분포 관심 X



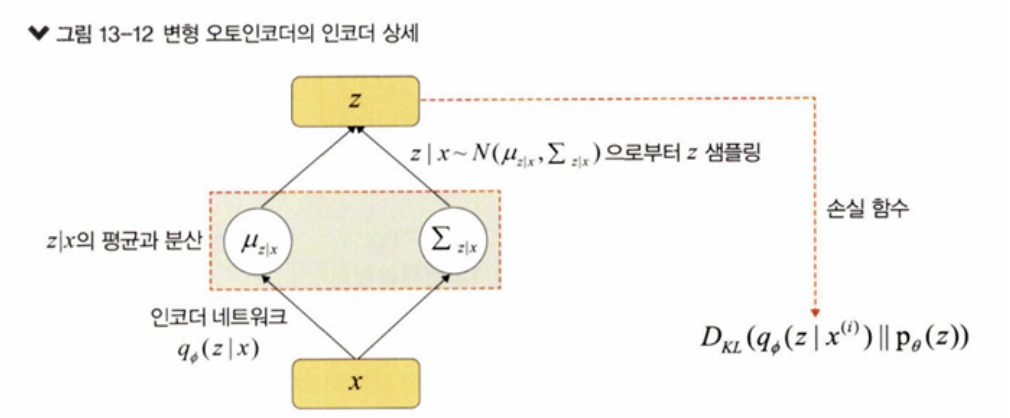
- 변형 오토인코더
  → 유사한 이미지 출력이 목표
  → 표준편차와 평균 이용해서 확률분포 만들고, 샘플링 및 디코더 통과로 입력 데이터와 조금 다른 출력 데이터 만들어냄
  → 가우시간 분포(잠재벡터 z)에서 벡터 랜덤하게 생성



- 변형 오토인코더 네트워크



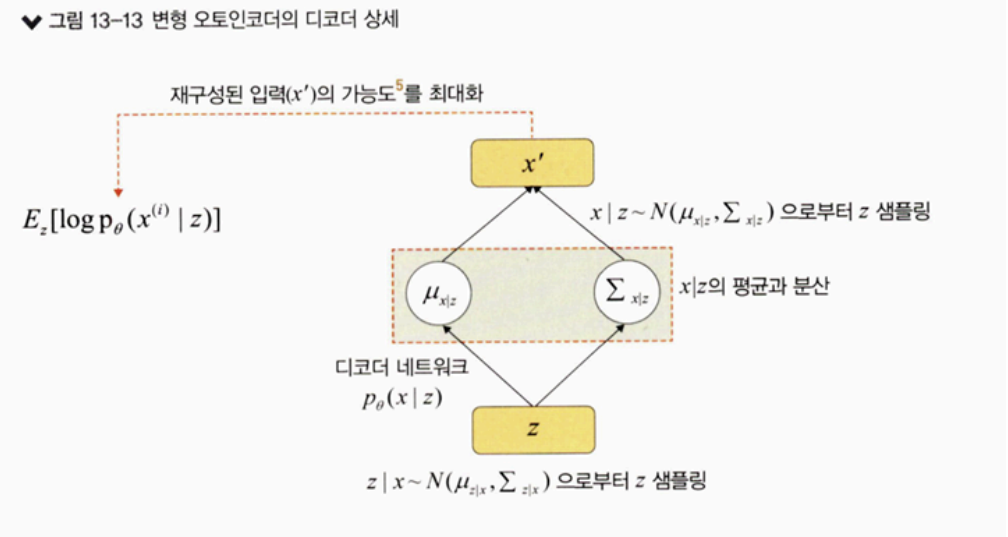
- 인코더 네트워크 상세 (257p다시보기)
  : q(z|x) : x입력받아 z와 대응되는 평균과 분산 구함
  → x를 인코더 네트워크에 보내서 평균과 표준편차 계산하여 출력하고 이를 이용하여 아래 2번째 항 값 구함



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

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

- 디코더 네트워크 상세
  : p(z|x) : z입력받아 x와 대응되는 평균과 분산 구함



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

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

- 수식 정리

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

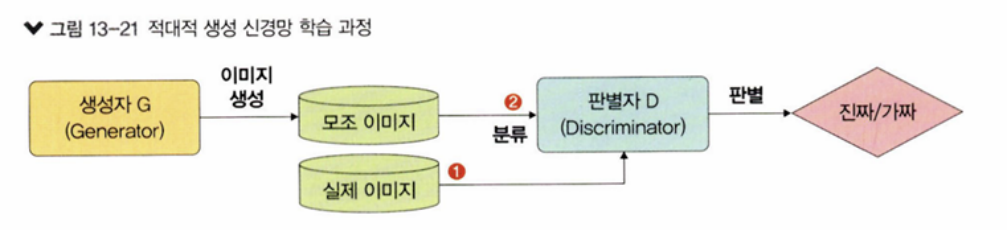
  - 1항은 z가 주어졌을 때 x프라임 표현 - 디코더 네트워크. 가능도 클수록 세타가 그 데이터 잘 표현한 것. 1항이 클수록 모델 가능도 커짐
  - 2항은 x에서 z를 표현하는 확률밀도함수 - 인코더 네트워크와 가우시안 분포가 얼마나 유사한지. 잘 나타낼수록 가능도 최대화됨. 따라서 2항 작을수록 모델 가능도 커짐

- 텐서보드에서 에포크 진행에 따른 오차 확인
    - 라이브러리 호출 / MNIST 데이터셋 이용
    - 인코더 디코더 네트워크 생성 - 평균과 분산 반환
    - 잠재 벡터 만드는 함수 생성
    - 손실함수 정의 (262p 다시 보기),
    - 모델 학습함수, 평가함수 정의
    - 모델 학습 및 텐서보드(학습과정 시각적 확인)에서 오차 확인

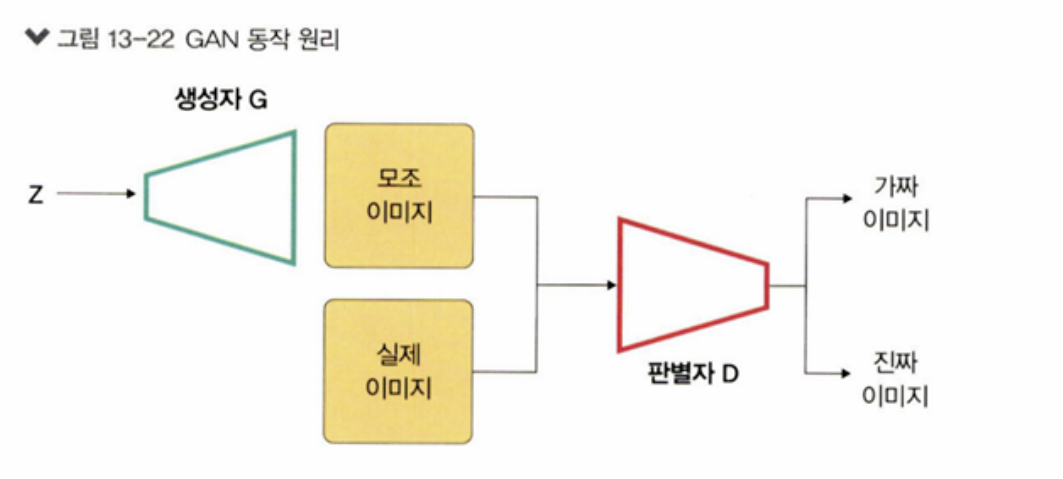
  → 결과 : KLD와 재구성 오차가 반비례, 오차 작아짐 : 이미지 생성이 잘 되고 있음
  → KLD가 일정 범위에서 수렴 : 입력값과 출력값이 가까워지고 있음

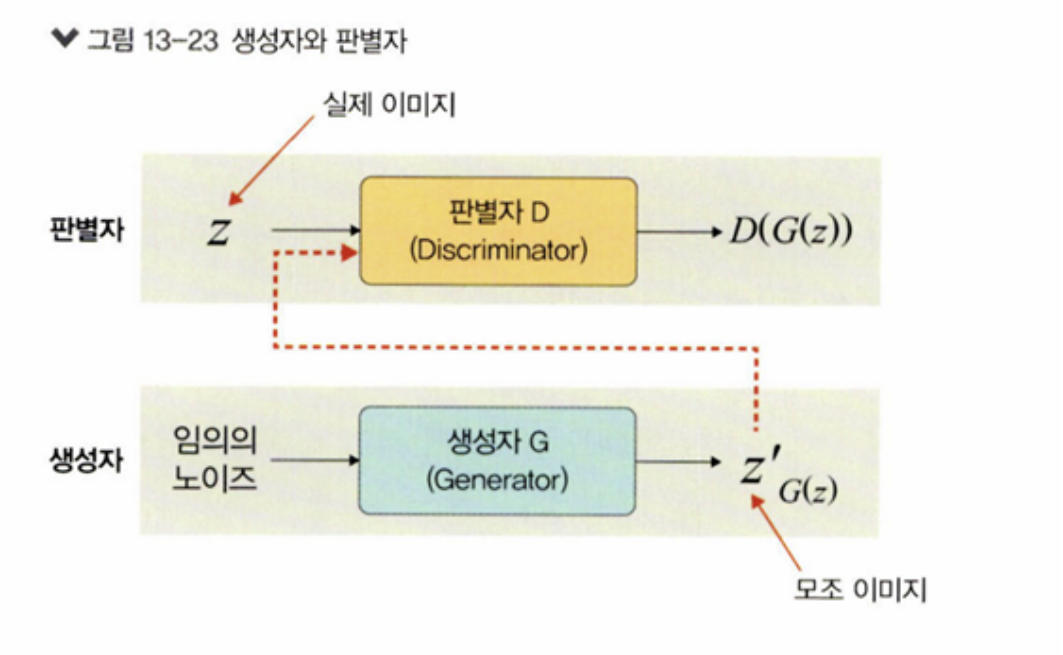
▼ 13.3 적대적 생성 신경망(GAN)

- 경찰(=판별자)과 위조지폐범(=생성자) 사이 검거와 범죄가 반복되면 위조지폐범 실력은 더 정교해질 것.
  → 최대한 진짜같은 데이터 생성하는 생성자와 최대한 가짜를 구분해내려는 판별자가 서로 적대적으로 학습
- 판별자 먼저 학습시킨 후 생성자 학습시키는 과정 반복
- 판별자 학습
    1. 진짜 이미지를 진짜로 분류하도록 학습
    2. 생성자 이미지를 가짜 이미지로 분류하도록 학습



13.3.1 GAN 동작 원리





- 구성
  - 생성자 : Generator - D를 속이려고 최대한 비슷한 이미지 만들도록 학습
    - 실제 이미지 z가 입력으로 주어졌을 때 z를 학습
    - 노이즈 데이터를 사용하여 모조이미지 z′(=G(z))를 생성
  - 판별자 : Discriminator - G가 만든 이미지를 잘 찾아내도록 학습
    - 이미지 x가 입력으로 주어졌을 때 판별자 D의 출력에 해당하는 D(x)가 진짜 이미지일 확률을 반환
    - z′를 입력으로 주면 실제 이미지일 확률 반환해줌
    - D 학습시킬 때 생성자 G고정한 채 실제 이미지는 높은 확률을 반환하는 방향으로 모조 이미지는 낮은 확률을 반환하는 방향으로 가중치 업데이트

- 손실함수 (아래의 식은 D입장, G입장이 함께 들어가 있음)

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

- $x\sim P_{data}(x)$: 실제 데이터에 대한 확률 분포에서 샘플링한 데이터
- $z\sim P_z(z)$: 가우시안 분포를 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 이미지인 $G(z)$가 1에 가까우면 진짜 데이터로, 0에 가까우면 가짜 데이터로 판단

  - 판별자 D 입장

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

    - 이 식이 값이 최대가 되도록 함.
    - 진짜 이미지 = 1, 생성이미지 = 0 일 때 값이 최대가 됨
  - 생성자 G 입장

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

    - 이 식의 값이 최소가 되도록 함
    - D가 판별결과 진짜라고 판별할 때 값이 최소가 됨
  - 판별자와 생성자 파라미터는 번갈아 업데이트
  - 각 파라미터 업데이트 할때 반대 파라미터는 고정\

13.3.2 GAN 구현

1. 라이브러리 호출 / MNIST 데이터셋
2. 필요한 변수 값 설정
3. 생성자 및 판별자 네트워크 생성
4. 옵티마이저와 손실 함수 정의
5. 생성된 이미지와 저장 함수 정의

6. 판별자 및 생성자 학습을 위한 함수 정의
7. 모델 학습 및 생성자와 판별자의 오차 확인
8. 생성된 이미지 출력

1. Introduction

💡 논문에서 다루고 있는 주제가 무엇인지와 해당 주제의 필요성이 무엇인가
논문에서 제안하는 방법이 기존 방법의 문제점에 대응되도록 제안 되었는가

논문이 다루는 분야

- (Deep) Generative Model

해당 task에서 기존 연구 한계점

1. MLE(최대 우도 추정)나 관련 전략에 관한 연산들이 어렵거나 불가능함
2. piecewise linear units의 이점을 받기 어려움
    - 부분 선형 유닛 : 딥러닝에서 사용하는 비선형 활성 함수 중 하나
    - ex) ReLU 등 입력 값의 범위에 따라 선형으로 작동하지만 전체적으로는 비선형 특성을 유지함
    - 기울기 소실 문제 줄여주고, 계산과 학습을 쉽게 해줌
    - 판별자는 명확한 정답(레이블)이 주어지므로 비교적 안정적인 지도학습 환경이어서 ReLU를 사용해도 학습 신호가 강해서 기울기 소실 문제가 잘 발생하지 않는데, 생성자는 판별자가 주는 간접적 피드백만으로 학습하므로 학습신호가 약해하고 불안정해서 ReLU의 gradient가 0이 될 수도 있다는 특성이 기울기 소실 등의 문제로 발전할 수 있음
      → 챗지피티를 통해 해결한 것임 - 자료를 통한 추가적인 공부 필요

→ 이러한 이유들로 discriminative models에 비해 발전이 더딤

논문의 contributions

- **adversarial nets 제안**
    - generative model과 discriminative model이 적대적으로 학습
    - G : MLP에 넣어 랜덤 노이즈 더해서 D를 속일 수 있는 이미지 생성

    - D : G가 생성한 가짜이미지 MLP에 넣어서 구분
    - 위조지폐범 생성과 적발 예시를 들어 성능이 좋아지는 이유 설명
- G와 D 모두 backpropagation과 dropout만으로 완전한 학습이 가능
- G 모델은 forward propagation만 사용함
- 근사 추론이나 마르코프 체인이 전혀 필요 없음

2. Related Work

Introduction에서 언급한 기존 연구들에 대해 어떻게 서술하는가

제안 방법의 차별성을 어떻게 표현하고 있는가

- **RBMs, DBMs**
  →directed graphical models + latent variables 의 대안
  → undirected graphical models + latent variables
  → But, 위 모델들에서의 변수 간 상호작용은 예외적인 상황을 제외하고 계산이 불가능하며 MCMC (Markov Chain Monte Carlo)로 근사적으로 추정

  (정규와 상수, 분배 함수에 관한 부분-위 모델들에 대한 추가적인 이해 필요)
  → But, MCMC에 의존하는 알고리즘은
  mixing 문제가 존재
- **Deep Belief Networks (DBNs)**
  → 위 두 구조의 하이브리드 모델
  → 빠른
  layer-wise 학습 기준이 있지만, 두 모델 모두의 계산적 단점을 모두 포함하고 있음
- **Score matching 방식**
  → pdf가 정규화 상수를 제외하고 명시되어야 함
  → But, DBNs이나 DBMs에서는 정규화되지 않은 pdf조차도 계산 가능한 형태로 표현하기가 어렵다
- **NCE (noise-contrastive estimation) 방식**
  → pdf가 정규화 상수를 제외하고 명시되어야 함
  → GAN과 유사하게 discriminative criterion 사용함
  → But, 판별자로 직접 사용 : 노이즈 분포에서 샘플링 된 데이터와 생성한 데이터 구분

하도록
→ 어느 정도 노이즈 분포 학습하면 구분이 쉬워져서 학습이 느려짐

- **GSN (Generative Stochastic Network)**
  → 일반화된 노이즈 제거 오토인코더를 확장한 형태
  → 원하는 분포로부터 샘플을 생성하는 생성기계를 역전파를 통해 훈련
  → 두 모델 모두 마르코프 체인의 한 단계를 수행하는 과정에서 파라미터 학습하는 방식
  → But, GAN은 마르코프 체인 필요 없어서 훨씬 단순
  →
  GAN은 피드백 루프가 없어서 부분 선형 유닛을 효과적으로 활용 : 앞서 다른 생성형 모델들이 ReLU 등 부분 선형 유닛 을 활용하지 못했다고 했는데, GAN은 그 문제를 해결한 것 (?맞게 이해한건가?)
  → 요즘은 Auto-Encoding Variational Bayes, Stochastic Backpropagation 등도 등장

3. 제안 방법론

💡 Introduction에서 언급된 내용과 동일하게 작성되어 있는가

Introduction에서 언급한 제안 방법이 가지는 장점에 대한 근거가 있는가

제안 방법에 대한 설명이 구현 가능하도록 작성되어 있는가

Main Idea

[Adversarial nets]

- 교재에서의 변수 정의 부분 차용

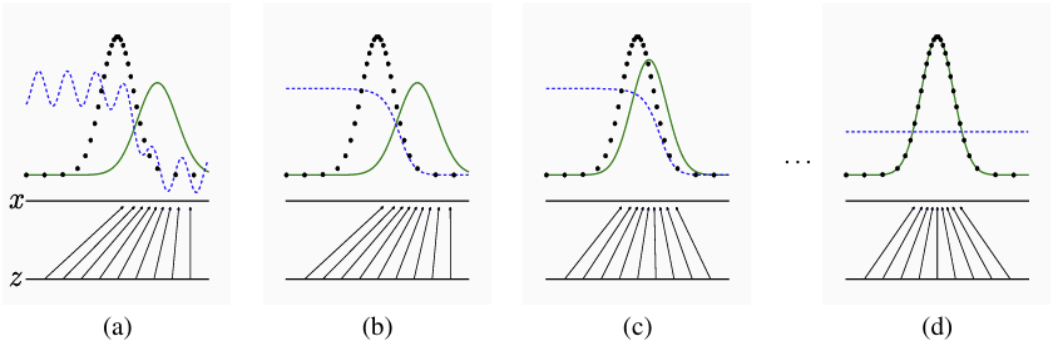
$$\min_G \max_D V(D,G) = E_{x\sim Pdata}(x)[\log D(x)] + E_{z\sim Pz}(z)[\log(1 - D(G(z)))]$$

- $x\sim P_{data}(x)$: 실제 데이터에 대한 확률 분포에서 샘플링한 데이터
- $z\sim P_z(z)$: 가우시안 분포를 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 이미지인 $G(z)$가 1에 가까우면 진짜 데이터로, 0에 가까우면 가짜 데이터로 판단

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

- V함수에 대해, D는 V를 높이고자 하고 G는 V를 낮추고자 하는 방향으로 학습하는 수식이다.
- 왼) 원본 데이터에서 이미지 뽑아 넣었을 때 D에 넣은 것
- 오) 노이즈 샘플링해서 생성자에 넣어 이미지 만들고 D에 넣은 값을 1에서 뺀 것
- D는 원본이면 1 가짜면 0 쪽으로 값을 뱉어낸다
- G는 오른쪽 항이 작아지도록 즉, D가 1을 뱉어내는 쪽으로 학습한다.
- 다시말해, G 가짜임을 안 들키는 방향으로, D는 G를 잘 걸러내고 진짜를 진짜로 걸러내는 방향으로 minmax game을 진행한다.

- 시간이 지나면서 생성 모델 G가 원본 데이터의 분포를 학습한다
  → 초록선 : 생성모델 분포
  → 파란선 : 원본 데이터의 분포



- 학습이 잘 되었다면 통계적으로 평균적인 특징을 가지는 데이터를 쉽게 생성할 수 있다.

[Theoretical Results]

(이해가 안되는 부분들은 chatGPT의 도움을 받았습니다)

(참고 영상 : https://www.youtube.com/watch?v=AVvIDmhHgC4)

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.

- 위의 식은 D(D가 포함된 두 항)에 대해 D가 진짜를 1로 뱉고 가짜를 0으로 뱉을 수 있도록 업데이트하는 수식
- 아래의 식은 G(G가 포함된 오른쪽 항)에 대해 G를 업데이트 하는 수식

4.1 Global Optimality of pg=pdata

- GAN의 훈련 목적 함수가 어떤 상황에서 최적에 도달하는지에 대한 이론적 설명이 이루어지는 부분
- 생성자 G가 만들어내는 데이터 분포 pg가 실제 데이터 분포 pdata와 정확하게 일치할 때 이 훈련 구조가 global optimum에 도달함을 증명함
- 최적의 D

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

- 최적의 D를 목적함수에 대입해서 수식을 전개하면,

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

- Jensen-Shannon Divergence을 기반으로 수식 정리 가능

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

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

- 결론적으로 pg=pdata일 때 JSD=0으로 C*의 최솟값 도달

$$
C^* = -\log(4)
$$

4.2 Convergence of Algorithm 1

- GAN의 학습 알고리즘이 수렴한다는 것을 이론적으로 뒷받침하는 내용
- G와 D가 번갈아가며 학습을 진행할 때, 두 모델이 균형 상태에 도달하는 것이 수학적으로 타당한지 다룸
- 이상적 조건 가정
  - 각 스텝마다 D는 현재의 G에 대해 최적의 D*을 정확히 학습
    - 아래 수식은 G에 대해 판별자 D가 최적으로 훈련되었을 때의 V

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

  - G는 목적함수 V에 따라 경사하강법을 통해 업데이트 됨

    → “D는 매번 완벽하게 훈련되고, G는 그에 따라 천천히 업데이트"되는 상황을 상정

- Theorem1
  → G와 D가 충분히 큰 신경망이면(충분한 표현력을 가지고 있으면) 위 가정에서 매 스

텝마다 D는 최적화되고, G는 위 수식을 개선하는 방향으로 조금씩 업데이트된다면 pg는 pdata에 수렴하게 된다.

Contribution

4. 실험 및 결과

💡 Introduction에서 언급한 제안 방법의 장점을 검증하기 위한 실험이 있는가

Dataset

- MNIST
- TFD (Toronto Face Database)
- CIFAR-10

Baseline

- generator nets
  - 활성화 함수 : ReLU, 시그모이드 함수를 혼합
  - noise를 입력 데이터로 받아 데이터를 생성
  - 이론적인 프레임워크→ 중간층에 dropout, noise 허용
  - 실험 → 맨 하위계층에 input에만 noise 적용
- discriminator net
  - 활성화 함수 : maxout
  - 학습시킬 때 dropout 적용
- G를 통해 생성된 sample들에 Gaussian Parzen window를 피팅함
- 이 분포에서 log-likelihood를 reporting함으로써 pg에서의 test set data의 확률을 측정
- Gaussia에애서 시그마 구할 때 cross validation 사용
  (likelihood 추정하는 방식)

결과

1. 실험 결과

- Parzen window 기반으로 추정한 로그 가능도

  → 학습된 생성 모델(GAN 등)이 얼마나 그럴듯한 데이터를 잘 만들어내는지를 수치로 평가한 값 (챗지피티의 도움 받아 해석)

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

  - **MNIST** : 테스트셋에 대한 샘플의 평균 로그 가능도 / 표준 오차는 샘플 간 평균을 기준으로 계산
  - **TFD** : 폴드마다 다른 시그마를 검증셋으로 선택 / 각 폴드의 평균 로그 가능도를 계산
  - Adversarial nets가 MNIST와 TFD 양쪽 모두에서 다른 모델들보다 더 높은 성능을 보였음을 확인할 수 있다.

- 생성형 모델이 생성한 샘플 시각화
  → 맨 오른쪽 열은 생성 이미지와 가장 유사한 실제 훈련 샘플
  → 외워서 만든 게 아님을 확인할 수 있다.
  →
  마르코프 체인에 의존하지 않았기에 샘플들이 uncorrelated하다?
  → 기존 모델들보다 훨씬 잘 생성해냈음을 확인할 수 있음.

- 잠재공간 z에서 좌표를 선형 보간하여 생성한 MNIST 숫자 이미지 시각화
  → 연속적인 변화를 학습했음을 나타내며, 부드러운 전이가 가능함을 시각적으로 입증함

- 생성모델 간 비교
  → 학습, 설계, 샘플링, 추론, 평가 측면에서 어려움과 특성을 요약

	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

- GAN을 살펴보면 다른 모델에 비해 상대적으로 단순하고 효과적임
  → 학습 : G와 D 동기화만 잘 하기
  → 추론 :
  학습된 근사 추론 → 입력값 x에서 z 추론하는 게 아니라 z 샘플링해서 G(z)를 통해 샘플을 생성한다는 뜻
  → 샘플링 : 필요 없음
  → 평가 : 명시적 확률 분포 없이 Parzen estimation 사용
  → 설계 : 거의 모든 미분가능 함수 이론적으로는 사용 가능함

2. 장점과 단점

- 장점
  - Markov chains 필요 없음 backprop만 사용됨
  - 학습할 때 추론 X
  - 다양한 종류의 functions이 모델에 통합될 수 있음
  - 계산이 잘 됨!
  - x를 직접 학습하는 게 아니라서 입력 데이터 정보가 생성자 가중치로 그대로 복사되는 현상 방지 가능
    → 훈련데이터 외워버리는 과적합 방지 가능
- 단점
  - pg(x)에 대한 명시적 표현 불가

- 학습할 때 D랑 G 싱크가 잘 안 맞고 G가 너무 학습되면 “the Helvetica scenario” 문제 발생

5. 결론 (배운점)

연구의 의의 및 한계점, 본인이 생각하는 좋았던/아쉬웠던 점 (배운점)

- 의의 및 한계점
  1. 조건부 GAN 구조로 쉽게 확장 가능하다
  2. x를 통해 잠재변수 z를 예측하는 보조네트워크를 학습시키는 방향으로 근사추론 학습이 가능하다.
  3. x의 일부가 주어진 상황에서 나머지를 예측하는 조건부 모델을 근사적으로 모델링 할 수 있다.
     → GAN으로 MP-DBM의 확률적 버전을 구현 가능하다
  4. semi-supervised learning에 활용 가능하다
     → 라벨이 적은 경우 GAN에서 학습된 표현 활용
  5. 학습 효율 향상 가능성
     → G와 D를 잘 조율하거나, z 샘플링 시 분포를 잘 설계하면 학습 가속화 가능하다.
- 좋았던/아쉬웠던 점/배운 점
  - 확률변수를 구하는 게 아니라 함수 자체를 샘플링하는 방식이 인상깊었다.
  - 모델을 생성자와 판별자로 나누어서 서로 경쟁적으로 학습하게 만드는 구조가 인상깊었다.
  - 로그 가능도 말고 뭔가 다른 성능 지표가 있으면 좋을 것 같다는 생각이 들었다.

    → Inception Score
    → Frechet Inception Distance
    → Precision and Recall for GANs
    → MM-SSIM
    → User Study
    : 챗지피티 간단한 질의응답 결과임. 더 찾아보기

6. 궁금한 점

[교재]

- 오토인코더가 중요한 이유
  1. 데이터 압축 : 메모리 측면 장점 - 용량 작고 품질 좋아짐

     (?용량이 작아지는 건 ok 품질은 왜 좋아지지?)

     → 기존 이미지나 음질보다 더 나아졌다는 말보다는 노이즈가 줄어들었다는 측면에서 품질이 좋아진 것이다.
- 변형 오토인코더 네트워크 상세 부분 - 잘 이해 안감