[논문리뷰] GAN [2014]

Paper Link

Generative Adversarial Networks

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution,

https://arxiv.org/abs/1406.2661

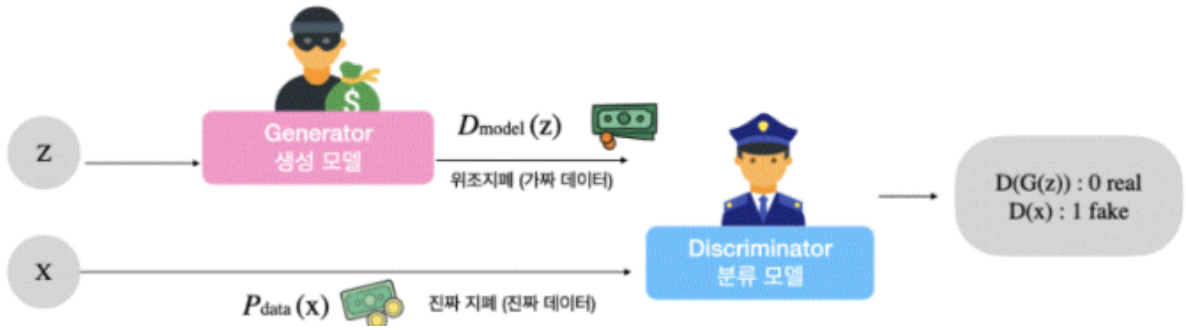
0. Abstract

1. Introduction

- 해당 논문은 **적대적인(Adversarial)** 과정을 통해 Generative model을 추정하는 새로운 프레임워크를 제안한다.
- Generator와 Discriminator 두 모델을 경쟁적으로 학습시켜 둘 모두를 동시에 **(Simulaneous)** 최적화한다.
- Generative model G는 데이터의 분포를 모방해 데이터를 생성해내는 모델 → Dicriminative model D가 구별하지 못하도록 D가 실수할 확률을 최대화하는 것이 학습의 목표이다.
- Dicriminative model D는 sample 데이터가 G로부터 나온 데이터가 아닌 실제 traing data로부터 나온 데이터일 확률을 추정하는 모델이다.

⇒ 이 논문에서는 이러한 프레임워크를 minimax two-player game이라고 표현한다.

✔️ 임의의 G, D에 대한 함수 공간에서 G는 훈련 데이터의 분포를 학습하게 되면서, D는 데이터가 실제인지 G가 생성해낸 데이터인지 판별하는 확률은 1/2가 된다. (이 확률이 1/2이라는 것은 구분해내지 못한다는 것이다.)



This framework can yield specific training algorithms for many kinds of model and optimization algorithm. In this article, we explore the special case when the generative model generates samples by passing random noise through a multilayer perceptron, and the discriminative model is also a multilayer perceptron. We refer to this special case as *adversarial nets*. In this case, we can train both models using only the highly successful backpropagation and dropout algorithms [16] and sample from the generative model using only forward propagation. No approximate inference or Markov chains are necessary.

- G와 D 모두 MLP로 구성되어있다면 다른 네트워크 필요없이 Backpropagation과 Dropout 알고리즘으로 학습을 진행하며, 샘플 데이터는 G의 forward propagation를 사용해 만든다. 2014년 기준으로 기존에 쓰이던 Approximate inference나 Markov chains 방법은 사용하지 않는다.

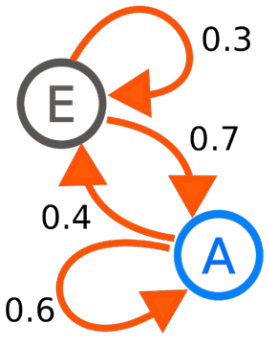
▼ ***Markov chain***

마르코프 성질을 가진 이산 **확률과정**이다.

- **확률과정(Stochastic process)**란 어떤 확률분포에 의해 발생되는 시행(trial)의 수열(sequence)을 만들어낼 수 있는 구조를 뜻한다.
  - ***e.g.*** 동전던지기는 결과가 앞면 또는 뒷면이므로 베르누이시행이다. 이를 베르누이시행을 k번 수행한다면 이는 Sequence $(a_1, a_2, a_3, ..., a_k)$ = $(\text{앞면}, \text{뒷면}, \text{뒷면}, ..., \text{앞면})$ 처럼 표현할 수 있다. 이 수열은 베르누이과정에 의해 생성된 것이다.
- **마르코프 성질**

과거와 현재 상태가 주어졌을 때의 **미래 상태의 조건부 확률 분포가 과거 상태와는 독립적으로 현재 상태에 의해서만 결정되는 것**

- ***e.g.***



https://ko.wikipedia.org/wiki/마르코프_연쇄

동전던지기는 독립시행이기에 마르코프 성질을 만족하지 않는다 하지만 위의 그림처럼 현재 상태가 A일 때 미래 상태가 A일 확률은 0.6이고 현재 상태가 E일 때 미래 상태가 A일 확률은 0.7인 것처럼 현재 상태에 의해서만 미래 상태가 결정될 때 마르코프 성질을 만족한다.

2. Related Work

3. Adversarial nets

- **Adversarial Modeling framework**는 모델 G, D가 MLP일 때 적용하기 더 쉽다.

The adversarial modeling framework is most straightforward to apply when the models are both multilayer perceptrons. To learn the generator's distribution p_g over data $\boldsymbol{x}$, we define a prior on input noise variables $p_{\boldsymbol{z}}(\boldsymbol{z})$, then represent a mapping to data space as $G(\boldsymbol{z};\theta_g)$, where G is a differentiable function represented by a multilayer perceptron with parameters θ_g. We also define a second multilayer perceptron $D(\boldsymbol{x};\theta_d)$ that outputs a single scalar. $D(\boldsymbol{x})$ represents the probability that $\boldsymbol{x}$ came from the data rather than p_g. We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize $\log(1 - D(G(\boldsymbol{z})))$. In other words, D and G play the following two-player minimax game with value function $V(G, D)$:

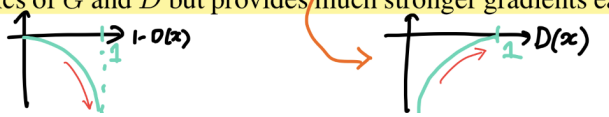
- 원본 데이터 x에 대해 생성된 분포 p_g를 학습하기 위해 입력에서 더해질 noise 변수를 $p_z(z)$라고 정의한다. 이 때 데이터 공간에 대한 mapping을 $G(z;\theta_g)$라고 하는데, 여기서 G는 파라미터 θ_g를 가진 MLP로 표현되는 미분가능한 함수이다. 또한 단일 스칼라를 출력하는 두 번째 MLP를 $D(x;\theta_d)$라고 정의한다. D는 x가 p_g보다는 원본데이터가 만

든 분포일 확률을 나타낸다. D와 G가 하는 minimax two-player game을 다음과 같은 수식으로 표현할 수 있다:

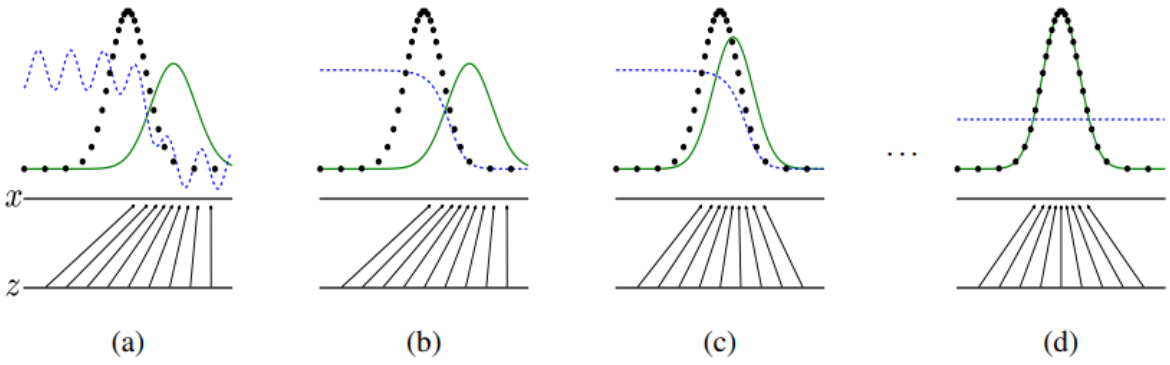
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \quad (1)$$

In practice, equation 1 may not provide sufficient gradient for G to learn well. Early in learning, when G is poor, D can reject samples with high confidence because they are clearly different from the training data. In this case, $\log(1 - D(G(\boldsymbol{z})))$ saturates. Rather than training G to minimize $\log(1 - D(G(\boldsymbol{z})))$ we can train G to maximize $\log D(G(\boldsymbol{z}))$. This objective function results in the same fixed point of the dynamics of G and D but provides much stronger gradients early in learning.

4 Theoretical Results



- **G의 loss를 최소화하는 방향**이면서 **D의 loss를 최대화 하는 방향**이 Adversarial net의 학습목표가 된다.
- **G를 최소화하는 것**은 **G가 만들어낸 데이터 $x = G(z)$를 가지고 D를 잘 속여 $D(G(z))$를 1에 근사시켜 $logD(G(z))$를 0에 근사시켜 최대화하는 것**이고, 이는 다시 말해서 $1 - D(G(z)$**는 0에 근사시켜 $log(1 - D(G(z)))$를 $-\infty$에 근사시는 것**과 같은 말이다.
-



- z : latent space에서 나온 latent vector
- 초록색 선 : generative distribution p_g
- 파란색 선 : discriminative distribution
- 검은색 선 : (sample) data generating distribution p_x

- **처음에는 노이즈(z)로부터 생성된 결과물이 학습이 덜 된 G의 분포를 가지기에 실제 데이터셋과 거리가 있다.** 그러나, **학습이 진행될수록 G의 분포가 데이터셋의 분포를 따라가게 되며 이상적인 결과 (d)에서는 D가 실제 데이터와 생성 데이터의 차이를 찾지 못하여 어떠한 경우에도 1/2 확률을 나타낸다.**
- latent space에서 나온 벡터 z를 x에 근사시킨다 = generative distribution을 data generating distribution에 맞춘다 → discriminator는 generative distribution과 data generating distribution 2개를 구분하지 못하고 헷갈리게 되므로 discriminative distribution은 모든 영역에 있어서 $D(x) = 1/2$이 될 것이다.

4. Theoretical Results

latent space

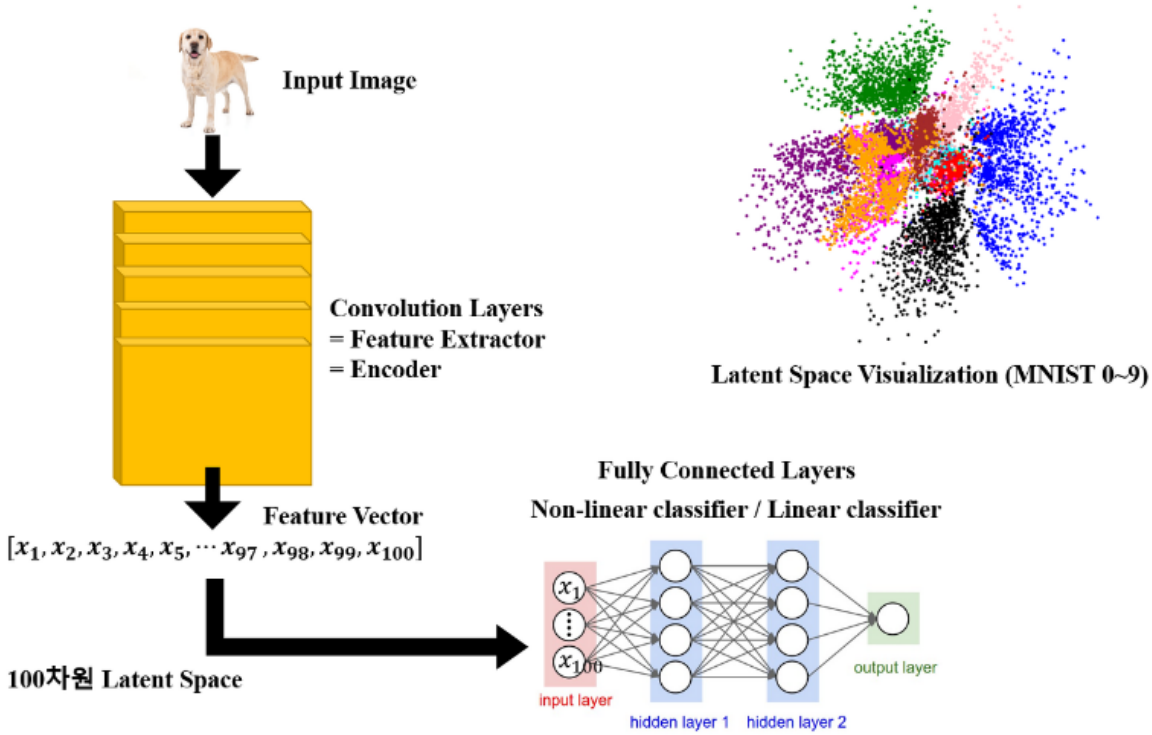


사진 출처 : https://unist.tistory.com/4

✔️ Convolution연산을 통해 이루어지는 **feature extraction**은 데이터로부터 **representation**을 얻어내는 것이다. 이는 정보를 축약하는 encoding 과정이라고도 할 수 있다.
왜 representation을 뽑아내느냐? 그 이유를 이미지 데이터로 예를 들면 이미지는 차원이 너무 커서 연산이 오래걸린다. 비효율적이라는 것이다. 때문에 효율성을 위해 차원을 낮추면서 필요한 feature만 뽑아내는 것이다. 뽑아낸 이 **latent vector**들로 **latent space**를 만들 수 있고, 만약 MNIST 데이터셋으로부터 classifier를 만들어냈다고 가정한다면 숫자 10개마다 시각화 figure처럼 latent space가 만들어지는 것이다.
시각화 figure에서 하나의 점은 하나의 샘플이고 색깔마다 class를 구분한 것이다. **하나의 색깔을 가진 점들의 분포를 가장 잘 표현하는 벡터를 latent vector** 또는 feature vector라고 하는 것이고 **딥러닝 모델에서 학습의 목적은 이 latent vector를 클래스를 가장 잘 represent하는 이상적인 latent vector에 근사시키는 것**이다.
앞서 예로 들었던 MNIST 데이터셋의 학습시킨 classifier가 가지고 있는 latent space를 가지고 inference를 한다면 input으로 들어온 이미지로부터 만들어낸 latent vector가 각 숫자와 가장 유사한 representation을 갖는 latent vector, 다시말해 확률이 가장 높은 class로 분류를 하는 것이다.

☑ ~~질문한 내용~~

Q1 : **latent vector와 feature vector는 같다고 할 수 있는가?**

→ 동일하게 쓰는 것 같다. 데이터를 표현하는 어떤 게 잠재적으로 있을 것이다. 사람은 모르지만 이 잠재적인 것을 기반으로 데이터가 만들어졌다고 하는 것이다. 이 잠재적인 것을 latent vector라고 한다.

Q2 : **feature map은 latent vector 또는 feature vector의 set이라고 할 수 있는가?**

→ 집합으로 볼 수 있을 것 같지만 set의 정의가 올바른지는 확실하지 않음. feature map은 레이턴트를 만들어놓은 제너레이터로 볼 수도 있다.

KL divergence

☐ 찾아봐야함

Algorithm 1

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** SGD

- 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.

- 하이퍼파라미터 k는 discriminator 모델에 적용하기위한 스텝
- noise prior $p_g(z)$로부터 m개의 noise sample을 미니배치로 샘플링한다.
- data generating distribution $p_{data}(x)$로부터 m개의 sample을 미니배치로 샘플링한다.
- SGD를 사용하여 D를 업데이트한다.

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

We first consider the optimal discriminator D for any given generator G.

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

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

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

$$\begin{aligned} V(G,D) &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_z p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx \end{aligned} \tag{3}$$

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$, the function $y \to a \log(y) + b \log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. The discriminator does not need to be defined outside of $Supp(p_{\text{data}}) \cup Supp(p_g)$, concluding the proof. □

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

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

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

Proof. For $p_g = p_{\text{data}}$, $D^*_G(\boldsymbol{x}) = \frac{1}{2}$, (consider Eq. 2). Hence, by inspecting Eq. 4 at $D^*_G(\boldsymbol{x}) = \frac{1}{2}$, we find $C(G) = \log \frac{1}{2} + \log \frac{1}{2} = -\log 4$. To see that this is the best possible value of $C(G)$, reached only for $p_g = p_{\text{data}}$, observe that

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

and that by subtracting this expression from $C(G) = V(D^*_G, G)$, we obtain:

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

where KL is the Kullback–Leibler divergence. We recognize in the previous expression the Jensen–Shannon divergence between the model's distribution and the data generating process:

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

Since the Jensen–Shannon divergence between two distributions is always non-negative and zero only when they are equal, we have shown that $C^* = -\log(4)$ is the global minimum of $C(G)$ and that the only solution is $p_g = p_{\text{data}}$, i.e., the generative model perfectly replicating the data generating process. □

4.2 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. Consider $V(G, D) = U(p_g, D)$ as a function of p_g as done in the above criterion. Note that $U(p_g, D)$ is convex in p_g. The subderivatives of a supremum of convex functions include the derivative of the function at the point where the maximum is attained. In other words, if $f(x) = \sup_{\alpha \in \mathcal{A}} f_\alpha(x)$ and $f_\alpha(x)$ is convex in x for every α, then $\partial f_\beta(x) \in \partial f$ if $\beta = \arg\sup_{\alpha \in \mathcal{A}} f_\alpha(x)$. This is equivalent to computing a gradient descent update for p_g at the optimal D given the corresponding G. $\sup_D U(p_g, D)$ is convex in p_g with a unique global optima as proven in Thm 1, therefore with sufficiently small updates of p_g, p_g converges to p_x, concluding the proof. □

5. Experiments

6. Advantages and disadvantages

This new framework comes with advantages and disadvantages relative to previous modeling frameworks. The disadvantages are primarily that there is no explicit representation of $p_g(\boldsymbol{x})$, and that D must be synchronized well with G during training (in particular, G must not be trained too much without updating D, in order to avoid "the Helvetica scenario" in which G collapses too many values of $\mathbf{z}$ to the same value of $\mathbf{x}$ to have enough diversity to model p_{data}), much as the negative chains of a Boltzmann machine must be kept up to date between learning steps. The advantages are that Markov chains are never needed, only backprop is used to obtain gradients, no inference is needed during learning, and a wide variety of functions can be incorporated into the model. Table 2 summarizes the comparison of generative adversarial nets with other generative modeling approaches.

The aforementioned advantages are primarily computational. Adversarial models may also gain some statistical advantage from the generator network not being updated directly with data examples, but only with gradients flowing through the discriminator. This means that components of the input are not copied directly into the generator's parameters. Another advantage of adversarial networks is that they can represent very sharp, even degenerate distributions, while methods based on Markov chains require that the distribution be somewhat blurry in order for the chains to be able to mix between modes.

앞서 기술한

- 단점
  - $p_g(x)$가 명시적으로 존재하지 않는다
  - D, G가 균형을 맞춰 성능이 향상되어야한다(G가 D보다 빠르게 발전되면 G가 노이즈데이터 z를 많이 붕괴시키기 때문이다)
    - 한쪽을 고정시켜 놓고 학습을 하기도 한다.
- 장점 :
  - Markov chains이 사용되지 않고 gradients를 얻기위해 backpropagation만 사용된다.
  - Markov chains을 사용할 때보다 선명한 이미지를 얻을 수 있다.
  - traning 중에 inference가 필요하지 않다.

참고문헌

딥러닝 Representation, Feature, Latent Space, Encoder 뜻 설명

딥러닝, 특히 이미지 프로젝트를 수행하면 위 단어들을 정말 많이 보게 됩니다. 이것들이 어떤 뜻인지 직관적으로 알아보겠습니다. Representation은 말 그대로 '표현' 입니다. 고양이 이미지를 어떻게 표현할 수 있을까요? 픽셀 별

https://unist.tistory.com/4