BOAZ 2022-2 방학 스터디 [GAN때문이야]

chpter6.

단순한 1010 패턴

2022.07.26 1주차 발제자 최가희

목차

Generative Model

- 무엇인가 생성한다? : Unsupervised Learning
- 답이 정해져 있는 Supervised Learning과 비교 했을 때, 매우 어려운 문제.
- 대표적 생성 모델 문제 솔루션

  1) Boltzmann machine

  2) Auto-encoder (AE)

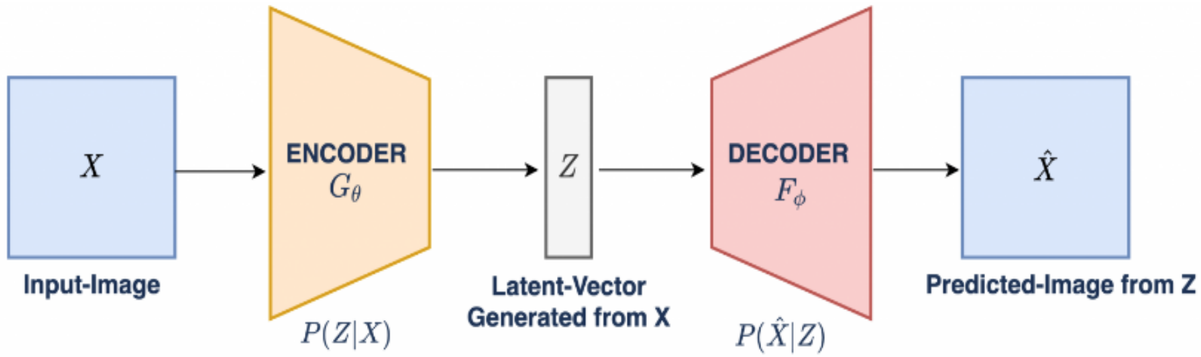
  3) Generative Adversarial Network (GAN)
- Auto-encoder (AE) : UL문제를 SL문제처럼 해결하는 방법

  ✔ 입력데이터를 label로 활용.

  ✔ Encorder : 이미지를 잘 압축하도록.

  ✔ Decorder : 압축데이터를 원본 이미지를 보고 재생성하도록.

[AE 구조]



1.2 GAN? - GAN 정의

GAN?

- Generative Adversarial Network : 생성적 적대 신경망

- 구조 : Generator + Descriminator 두 개의 신경망으로 구성

1) Generator

▲ 목적: 진짜처럼 보이게 하는 가짜 생성

▲ 학습: 무작위 sampling을 통해 생성된 값을 훈련시켜 실제와 유사한 '분포' 가 되도록 학습.

2) Descriminator

▲ 목적: 진짜는 진짜로, 가짜는 가짜로 판별하는 분류기.

▲ 학습: 두개의 서로 다른 분포가 얼마나 다른지 학습하고, 그 두개의 분포를 명확히 구분할 수 있도록 학습.

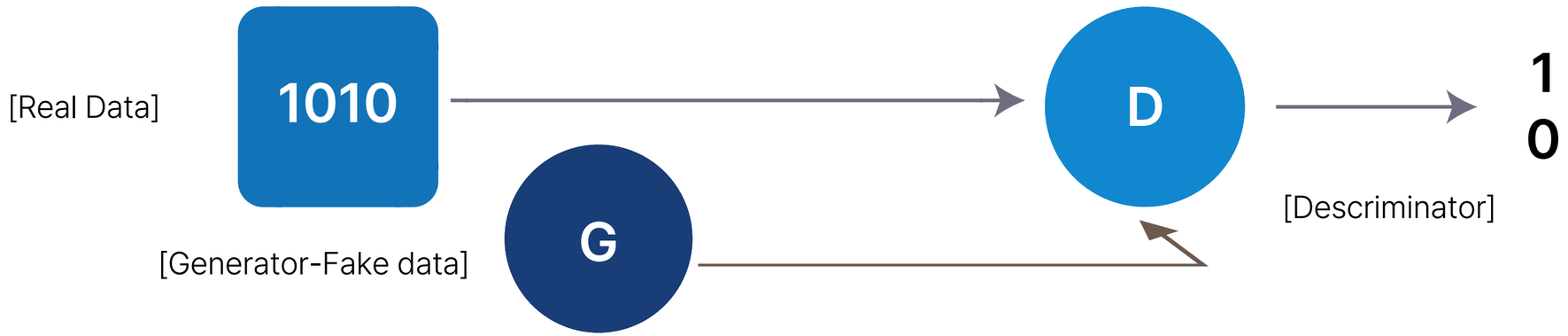
[지폐위조범과 경찰]

[GAN을 활용한 오바마 전 대통령 연설 입모양 생성]

2. 1010 패턴 개요

1010 패턴이란?

- 말 그대로 [1,0,1,0] 구조를 가지는 tensor 데이터
- [1,0,1,0] 패턴을 생성하고 판단하기 위한 데이터 생성
  1) train data 에 해당하는 값 : generate_real()
  2) test data에 해당하는 값 : generate_random()
- 본 교재에서는 정확한 정수 값이 아닌 실제로 만들어질법한 '근사한 실수' 로 생성 함.
  1) generate_real()에 사용된 random.uniform(): 두 수 사이의 랜덤 실수 값 반환. → 이를 torch.FloatTensor 통해 tensor로
  2) generate_random()에 사용된 torch.rand(): 0-1사이의 실수를 주어진 텐서 크기에 맞게 생성 → 바로 텐서 구조

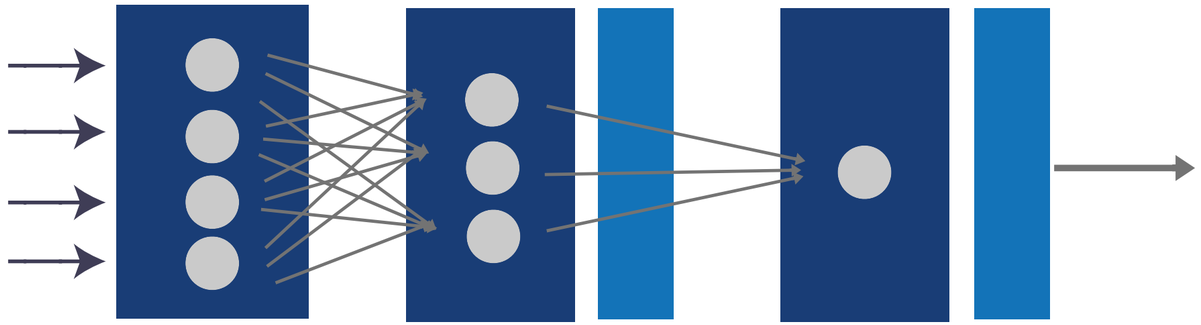


Descriminator

- foward() 를 구현하는 일반적인 분류기 신경망 (layer + loss + optimizer).
- 추후 D를 통해 G를 학습시키기 때문에 우선 정의.
- 입력 레이어: [1,0,1,0] 4개 -> 출력 레이어: 참, 거짓 中 1개

```
self.model=nn.Sequential(
                    nn.Linear(4,3),          # 중간 노드 3개로 출력
                    nn.Sigmoid(),            # 로지스틱 회귀 활성화 함수
                    nn.Linear(3,1),          # 한 개의 값으로 출력
                    nn.Sigmoid()     )
```

[layer 정의]

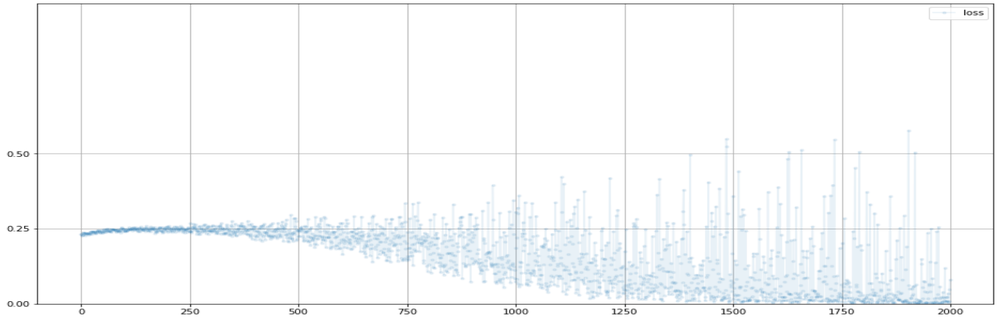


[Descriminator]

- Train()

```
D=Discriminator()                            # 판별기 객체생성
for i in range(10000):                       # Epoch
    D.train(generate_real(),torch.FloatTensor([1.0]))
    # 참을 참으로 판별하기 위한
    D.train(generate_random(4), torch.FloatTensor([0.0]))
    # 거짓을 거짓으로 판별하기 위한
```

[Train 코드]



[손실 차트]

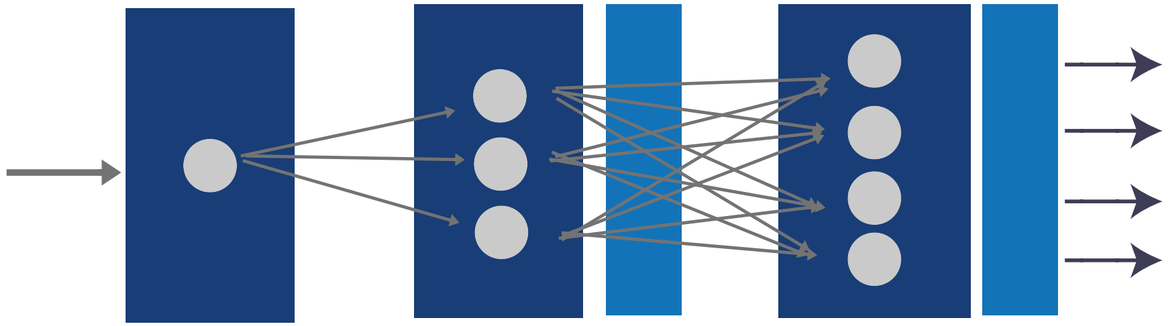
3.2 판별기 및 생성기 실습 - Generator

Generator

- 신경망 구조 : Descriminator 와 반대.
- 입력 레이어: 1개 -> 출력 레이어: [1,0,1,0] 4개
- 손실함수 정의 X -> 판별기로부터 계산된 손실값을 활용

```
self.model = nn.Sequential(
                    nn.Linear(1,3),          # 하나의 노이즈 값 입력
                    nn.Sigmoid(),
                    nn.Linear(3, 4),         # 4개의 [1,0,1,0] 값 출력
                    nn.Sigmoid()       )
```

[layer 정의]



[Generator]

- Train()

```
def train(self,D,inputs,targets):
    g_output=self.forward(inputs)              # 생성기 신경망 모델
    d_output=D.forward(g_output)               # 생성기 모델을 통해 만든 output 판별기에 넣기
    loss=D.loss_function(d_output,targets)     # 생성기를 통해 분류된 결과와의 loss 계산
```

[Train 코드]

" G 생성기를 통해 만들어진 결과를 D에 넣었을 때 계산된 LOSS로 최적화 "

(D손실함수에 G옵티마이저 활용)

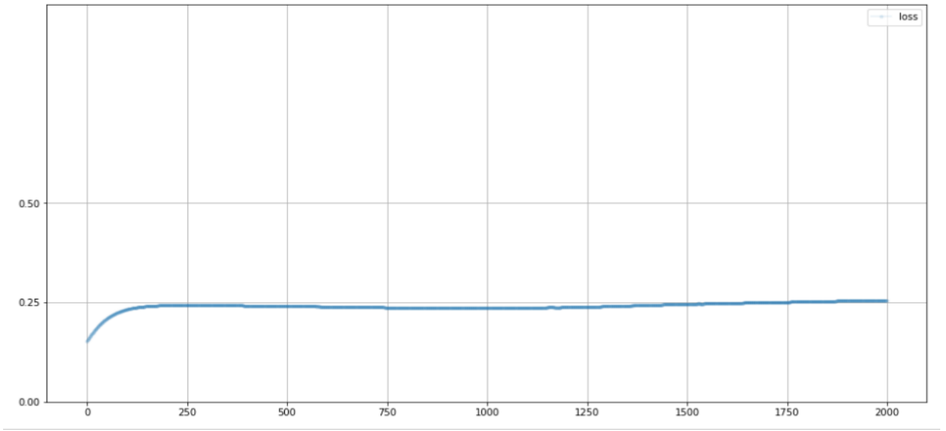
판별기 및 생성기 훈련

- 하나 하나 분리가 아닌, 동시에 훈련
- 최종 목표: D가 판별할 확률 " 0.5 "가 되도록.
- 3단계 훈련

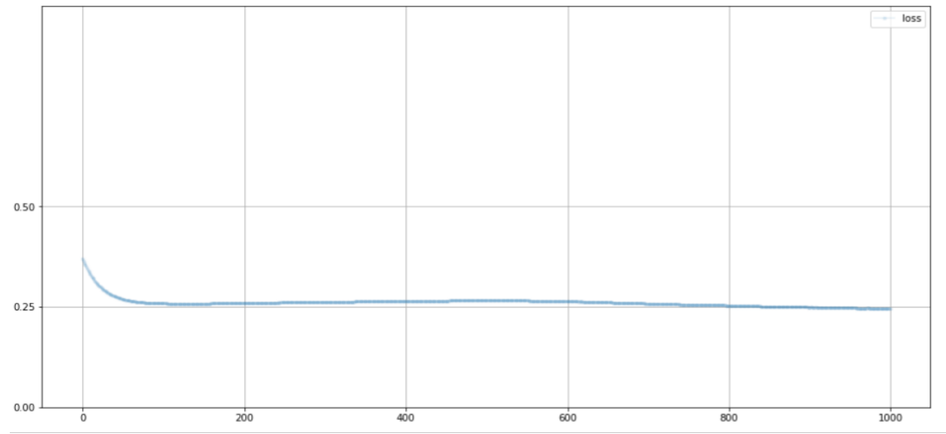
  1) 참에 대한 판별기 2) 거짓에 대한 판별기 3) 생성기 훈련

```
for i in range(10000):                          # EPOCH

    #1. 참을 참이라고 판단하도록
    D.train(generate_real(),torch.FloatTensor([1.0]))

    #2. 가짜를 가짜라고 판단하도록
    D.train(G.forward(torch.FloatTensor([0.5])).detach(),
                                        torch.FloatTensor([0.0]))

    #3. 가짜가 점점 진짜처럼 만들어지도록
    G.train(D,torch.FloatTensor([0.5]),torch.FloatTensor([1.0]))
```

[Train 코드]



[D.plot_progress()]



[G.plot_progress()]

3.4 판별기 및 생성기 실습 - Detach

✔ Detach?

"Returns a new Tensor, detached from the current graph.
The result will never require gradient." - PyTorch docs

즉, 파이토치 프레임워크는 tensor에서 이루어진 모든 연산을 추적해서 기록해 놓는다 (graph). 이 연산 기록으로부터 도함수가 계산되고 역전파가 이루어지게 된다. detach()는 이 연산 기록으로부터 분리한 (연산이 되지 않도록 하는) tensor을 반환하는 method.

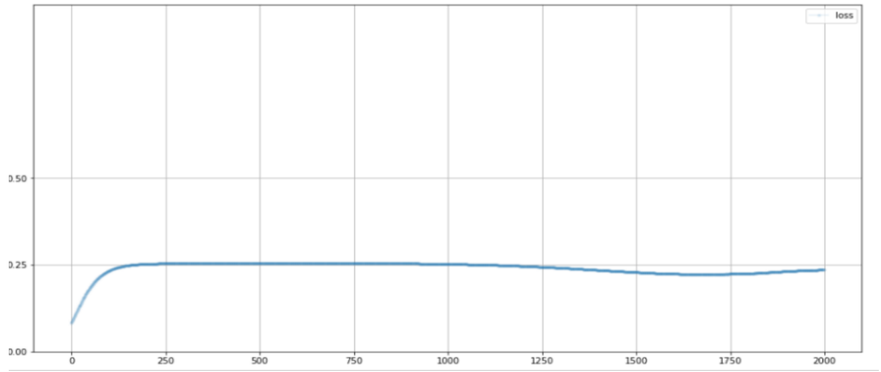
" G.forward(torch.FloatTensor([0.5])).detach() "

: D의 loss 연산 기록으로부터 분리하여 gradient 전파가 안되는 텐서

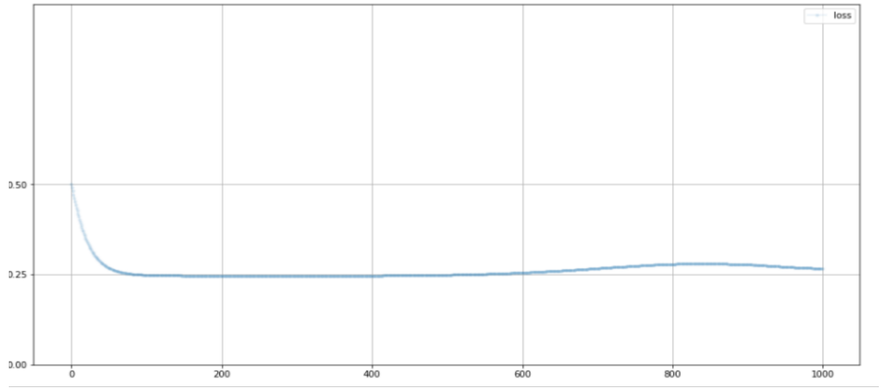
→ detach안해도 큰 영향은 끼치지 않지만 필요 없는 연산은 지양해야 효율적이고 빠른 연산 가능

detach 안해도 상관없나?

: 본 실습에서 2초 정도 느리게 훈련되었고, 손실 결과가 큰 차이 X



[D.plot_progress()]



[G.plot_progress()]