Machine Learning

12장 GAN(Generative Adversarial Networks)

고려대학교 통계학과

박유성

Contents

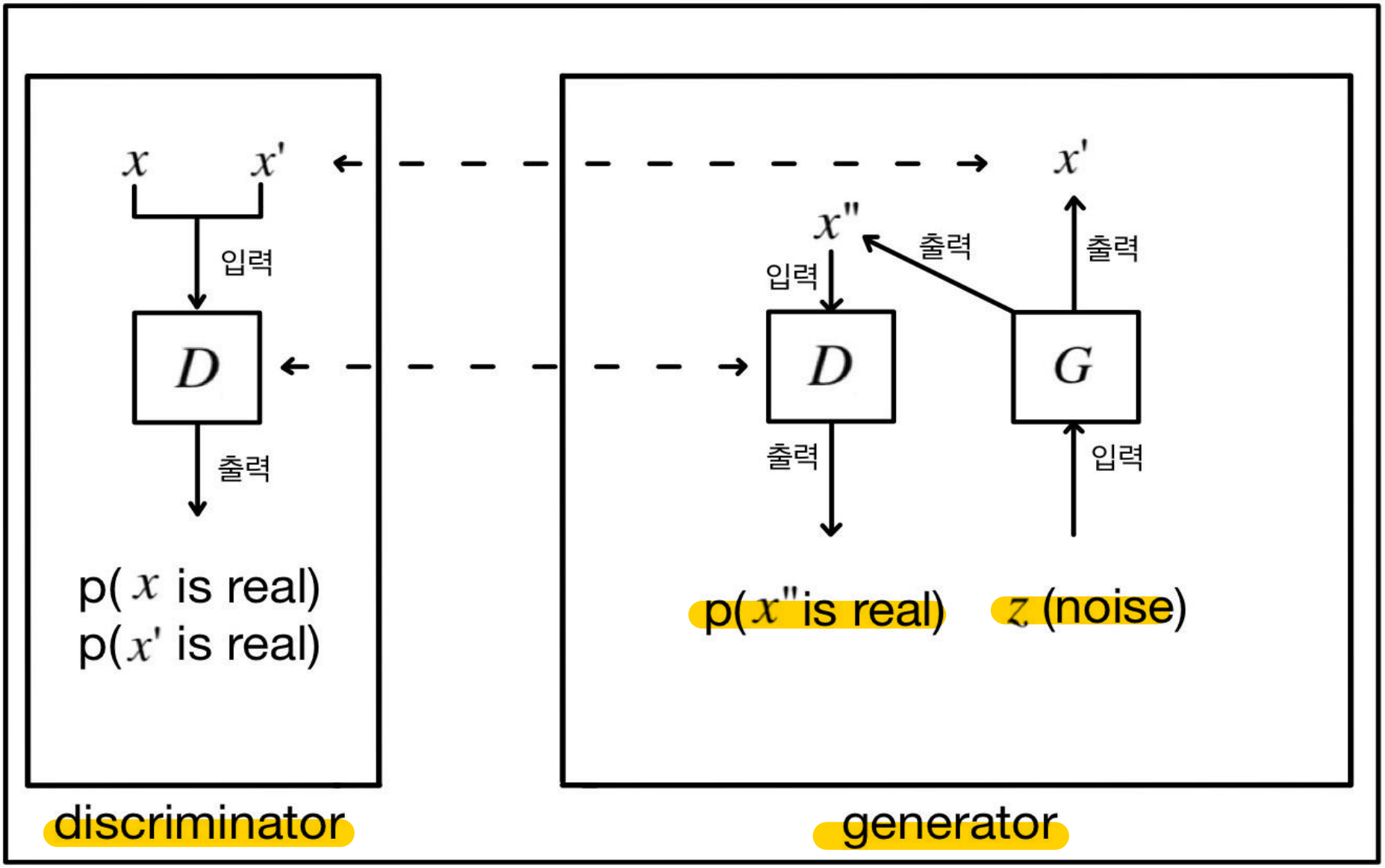
GAN (Generative Adversarial Networks)

- GAN모형의 목적도 VAE와 동일하게 입력변수의 분포를 추정하는데 있지만 입력 자료의 특성을 유지하면서 새롭고 창조적인 자료의 생성에 중점을 두고 있다.
- GAN은 generator와 discriminator로 구성되어 있다.
- generator는 임의의 잡음으로부터 합성한 가짜 자료를 생성하여 discriminator에 전달하고,
- discriminator는 가짜 자료와 진짜 입력자료를 구별하고 이 결과를 다시 generator에 전달한다.
- 그러면 generator는 discriminator가 전달한 결과를 기초로 좀 더 정교한 가짜 자료를 생성하여 궁극적으로 discriminator가 실제 자료와 가짜 자료를 구별하지 못할 때까지 위 과정을 반복한다.

GAN (Generative Adversarial Networks)

- 학습이 완료된 generator를 이용하여 임의의 잡음을 입력하여 실제 입력자료와 구별하지 못하는 새롭고 창조적인 자료를 생성한다.
- 이는 화폐위조범과 경찰과의 관계로 설명할 수 있다.
- 화폐위조범이 generator이고 경찰이 discriminator가 된다.
- 화폐위조범은 처음에는 조악한 화폐를 만들기 때문에 경찰은 손쉽게 위조화폐를 구별할 것이지만
- 위조범은 경찰이 위조화폐임을 알아차린 이유를 알게 될 것이고
- 이를 반복하면 궁극적으로 경찰이 실제화폐와 위조화폐를 구별하지 못할 정도의 경지에 이를 것이라는 개념이 GAN모형이다.

GAN (Generative Adversarial Networks)



GAN (Generative Adversarial Networks)

- 그림에서 D는 discriminator를, G는 generator를 표시하고 있다.
- 1 batch의 잡음을 generator G에 입력하여 가짜 자료 x'을 출력하여 Discriminator 박스에 전달한다.
- 실제 자료와 concatenate하여 discriminator D에 입력하고 D는 0~1사이 값을 출력한다. 실제 자료 x에 label 1을 부여하고 x'에 label 0을 부여한다.
- 그러므로 D(x) 는 가능한 한 1(real)에 가깝도록 하고 D(x') (=D(G(z)) 는 가능한 한 0(fake)이 되도록 D를 학습시킨다.
- 그러므로 discriminator의 손실함수는 일반적인 binary crossentropy가 된다.

GAN (Generative Adversarial Networks)

$$\ell(D) = -E_x log\big(D(x)\big) - E_z log\left(1 - D\big(G(z)\big)\right)$$: discriminator 손실함수

- 여기에서 $E_x log(D(x))$는 $\frac{1}{n}\sum_{i=1}^{n} logD(x_i)$로 추정되고 $E_z log\left(1 - D\big(G(z)\big)\right)$는 $\frac{1}{n}\sum_{i=1}^{n} log(1 - D(z_i))$로 추정된다. 여기에서 n은batch 크기이다.
- 손실함수를 계산하고 backpropagation에 의해 discriminator D를 개선하고 이 D를 generator 박스에 전달한다.
- 그러면 또 다른 잡음을 generator G에 입력하여 가짜 자료 x''를 출력하는데 $D(x'')$가 가능한 한 1로 출력할 수 있도록 generator G를 학습시킨다.
- 그러므로 가짜 자료 x''에 label 1(real)을 부여하여 binary crossentropy

$$\ell(D(G(z)) = -E_z\left(logD\big(G(z)\big)\right)$$: generator 손실함수

GAN (Generative Adversarial Networks)

가 최소가 되도록 G를 학습시킨다.

- 여기에서 label이 1만 있으므로 label 0에 해당하는 항 ($-E_z \log(1 - D(G(z))$)은 존재하지 않는다.
- 손실함수 $\ell(D(G(z))$를 최소화할 때, G를 학습시키는 동안 D의 모수는 고정(freeze)해야 한다.
- Discriminator 손실함수의 $D(G(z))$와 generator 손실함수의 $D(G(z))$ 정확하게 반대 방향으로 작동하고 있어,
- discrimator와 generator의 사소한 변경이 모수추정이 안되는(동일하게 모수가 수렴하지 않은) 원인이 되므로 매우 조심스럽게 아키텍쳐를 구성하여야 한다.
- 이러한 이유로 좀 더 안정적이고 향상된 GAN 모형의 필요성이 대두되었다.

01 DCGAN (Deep CNN GAN)

- keras가 제공하는 CIFAR10 이미지 데이터 중 개구리 이미지를 이용하기 때문에, 입력변수 x로 개구리이미지의 크기인 (32,32,3) 3D 텐서를 가정하고 있다.
- discriminator 학습을 위해 1 배치(CIFAR10으로부터 20개의 개구리 이미지자료)의 실제 개구리 이미지자료에 label 1을 부여한다.
- 1 배치의 잡음을 정규분포로부터 추출하여 generator모형에 적용, 1배치의 가짜 이미지 자료를 생성하고 label 0을 부여한다.
- 앞에서 만들어진 자료를 concatenate하여(그러므로 이미지자료가 40개가 됨) discriminator를 학습시킨다.
- 또 다른 1 배치의 잡음을 생성하여 label 1을 부여하여 gan에 입력하여 gan을 학습시킨다. 이때 discriminator의 모수는 고정되어 있으므로 실제적으로 generator를 학습시키게 된다.

01 DCGAN (Deep CNN GAN)

- 이 과정을 1 에폭이 끝날 때까지 반복한다.
- 위 과정을 10,000번 반복한다. 그러므로 배치의 크기가 20이므로 32에폭이 된다.

02 GAN 학습을 위한 손실함수

- 실제 데이터의 분포 : p_{data}, generator에 의한 분포 추정: p_g
- Kullback-Leibler divergence

$$D_{KL}(p_{data}||p_g) = E_{x\sim p_{data}} log \frac{p_{data}(x)}{p_g(x)} = \int \left(log \frac{p_{data}(x)}{p_g(x)} \right) p_{data}(x) dx \quad (1)$$

- (1)은 $p_{data} = p_g$ 이면 최소값
- Disscriminator(DCGAN)의 손실

$$\ell(D) = -E_x log\big(D(x)\big) - E_z log\left(1 - D\big(G(z)\big)\right)$$

$$= -E_{x\sim p_{data}} log D(x) - E_{x\sim p_g} \log\big(1 - D(x)\big) \quad (2)$$

02 GAN 학습을 위한 손실함수

$$= -\int \big(logD(x)\big)p_{data}(x)dx - \int \log(1 - D(x))p_g(x)dx$$

$$= -\int [\big(logD(x)\big)p_{data}(x) + \big(\log\big(1 - D(x)\big)\, p_g(x)\big)]dx$$

가 된다.

- $\big(logD(x)\big)p_{data}(x) + \big(\log\big(1 - D(x)\big)\, p_g(x)\big)]$ 를 최대로 하는 $D(x)$는

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

- 이를 (2)에 대입하면

$$l(D^*) = -E_{x\sim p_{data}} log \frac{p_{data}}{p_{data}+p_g} - E_{x\sim p_g} \log\left(1 - \frac{p_{data}}{p_{data}+p_g}\right)$$

$$= 2log2 - E_{x\sim p_{data}} \log\left(\frac{p_{data}}{p_{data}+p_g}\right) - E_{x\sim p_g}\left(\frac{p_g}{p_{data}+p_g}\right)$$

02 GAN 학습을 위한 손실함수

$$= 2log2 - D_{KL}(p_{data}||\frac{p_{data}+p_g}{2}) - D_{KL}(p_g||\frac{p_{data}+p_g}{2})$$

$$= 2log2 - 2D_{JS}(p_{data}||p_g)$$

- D_{JS}: $Jensen - Shannon\ divergence$

  → $l(D^*)$의 최소화는 D_{JS}의 최대화

- D_{KL}과 D_{JS}의 특성

  eg.

$$p_{data}(x_1, x_2) = p(x_2|x_1)p(x_1),\ \mathrm{p}(x_1 = 1) = 1, p(x_2|x_1 = 1) \sim U(0,1)$$

$$p_g(x_1, x_2) = p(x_2|x_1)p(x_1),\ \mathrm{p}(x_1 = 2) = 1, p(x_2|x_1 = 2) \sim U(0,1)$$

02 GAN 학습을 위한 손실함수

- $D_{KL}(p_{data}||p_g) = \int_0^1 log\frac{1}{0} dx_1$

  $D_{KL}(p_g||p_{data}) = \int_0^1 log\frac{1}{0} dx_2$

- $D_{JS}(p_{data}||p_g) = \frac{1}{2}\int_0^1 log\frac{1}{\frac{1+0}{2}} dx_2 + \frac{1}{2}\int_0^1 log\frac{1}{\frac{0+1}{2}} dx_2 = log2$
- 그러므로 D_{KL}은 존재하지 않고 D_{JS}는 상수이므로 미분값이 zero

  ⇒ 모수의 최신화가 불가능
- 즉, p_g가 p_{data}와 겹치는 부분이 없으면 손실함수가 상수가 됨
- 초기에 두 분포가 겹치는 부분이 있더라도 update 과정에서 겹치지 않거나 아주 작은 부분만 겹친다면 모수사라짐(vanishing)현상이 일어남.

  ⇒ DCGAN의 수렴문제 발생

02 GAN 학습을 위한 손실함수

- p_{data}와 p_g의 거리를 측정하는 새로운 손실함수가 필요
- 손실함수가 존재하고 손실함수의 미분값이 0이 아닌 손실함수

  ⇒ Wasserstein 1 함수

  Discriminator 손실:

$$l(D) = -E_{x\sim p_{data}} D_w(x) + E_{x\sim p_g} D_w(x') = -E_{x\sim p_{data}} D_w(x) + E_z D_w\big(G(z)\big)$$

$l\big(D(G)\big) = -E_z D_w(D_\theta(z))$: generator 손실

기본 idea: $l(D)$의 모수를 update한 후 아주 작은 값으로 w를 clipping

(eg. $-0.01 \le w \le 0.01$)

03 WGAN의 구현

- Keras를 이용한 WGAN아키텍쳐를 MNIST 데이터에 적용하여 구현하여 보기로 한다.
- WGAN의 label은 실제자료에는 1, 가짜자료에는 −1로 주어지므로 MNIST 데이터로 구성된 x_train을 −1~1 사이 값을 갖도록 x_train=(x_train.astype('float32')−127.5)/127.5으로 표준화하였다.
- n_critic=5는 discriminator 모수 w를 5번 최신화한 후 generator의 모수 θ를 한번 최신화하기 위한 옵션이고, discriminator 모수 w를 −0.01~0.01로 clipping하기 위해 clip_value=0.01로 주었으며,
- 최적화 알고리즘인 RMSprop의 학습률은 0.00005로 하였다. 이러한 옵션은 WGAN 모형의 수렴에 많은 영향을 미치므로 가능한 한 그대로 사용하는 것이 좋다.

04 LSGAN의 구현

- WGAN의 손실함수는 generator로부터 생성된 자료가 실제 자료와 너무 다르면 손실함수값을 줄이는 데 기여하지 못함

  ⇒ generator모수를 최신화하는데 기여하지 못함.

- 이를 보완한 손실함수가 squared error로 이 제곱오차를 최소화하는 GAN모형을 LSGAN이라고 한다.

$$l(D) = E_{x\sim p_{data}}(D(x)-1)^2 + E_z\left(D\left(G(z)\right)^2\right)$$

$$l\left(D(G)\right) = E_z\left(D\left(G_\theta(z)\right)-1\right)^2$$

- 그러므로 실제 데이터는 label을 1, 가짜 데이터에는 label을 0을 부여함.

04 LSGAN의 구현

- LSGAN 아키텍쳐를 위한 프로그램은 discriminator와 generator의 손실함수를 제외하고 WGAN과 동일하다.
- 이러한 이유로 generator와 discriminator 최종출력층의 활성함수를 변형해야 하고 실제자료와 가짜자료의 label을 각각 1과 0으로 변경하여야 한다.
- WGAN에서는 discriminator를 5번 학습하는 동안 generator를 1번 학습시킨 바 있다.
- 대부분의 교재나 github에에서는 LSGAN의 경우, discriminator와 generator를 교대로 학습시키는 프로그램을 제공하고 있다.
- 그러나 이러한 프로그램을 적용한 결과 수렴하지 않거나 성능이 WGAN보다 나쁜 것으로 나타나, WGAN의 아키텍쳐를 그대로 LSGAN에도 적용하기로 하였다.

Q & A