Do you know GAN? (1/2)

최근 딥러닝 분야에서 가장 뜨겁게 연구되고 있는 주제인 GAN(Generative Adversarial Network)을 소개하고 학습할 수 있는 글을 연재하려고 한다. GAN은 이름 그대로 뉴럴 네트워크(neural network)를 이용한 생성 모델이다. 흔하게는 이미지를 생성하는 것에서부터 음성, 문자에 이르기까지 다양한 분야에 적용되고 있다. 2014년에 이안 굿펠로우(Ian Goodfellow)가 GAN을 처음 선보였을 때부터 이미 대박의 '낌새'가 보였지만, NIPS 2016 이후부터는 GAN에 대한 관심이 더욱 폭발적으로 늘어나고 있다. 원래도 핫한 기계학습 분야에서도 GAN에 대한 연구는 유독 빠르게 발전하고 있다. 일주일이 멀다 하고 새로운 아이디어를 사용한 논문이 나오는 한편, 재미있는 방향의 관련 어플리케이션들이 속속들이 나오면서 GAN의 기세는 더해 가고 있다. 페이스북의 얀 레쿤(Yann Lecun) 교수도 GAN을 최근 10~20년 간에 기계학습 분야에서 나온 아이디어 중 최고라는 찬사를 보내는 만큼, 그 열기가 쉽게 식지는 않을 것 같다.

이번 호에서는 이렇게 사람들이 열광하는 GAN이 무엇인지, 어떤 원리로 그림을 생성하는지에 대해 먼저 소개하고자 한다. 그리고 독자들이 GAN의 개념과 그 밑에 깔려 있는 기본적인 수학적 배경을 이해하게 하고, 추후 새로운 GAN 연구를 봤을 때 관련 흐름을 따라갈 수 있는 기초적인 지식을 제공하는 것을 목표로 합니다.

글 | 유재준 jaejun2004@gmail.com

카이스트 바이오및뇌공학과 학부를 졸업하고 동 대학원 박사 과정을 밟고 있는 공돌이. 바이오영상신호처리 연구실에서 전통적인 신호처리 이론을 이용한 영상 복원이나 대수적 위상수학을 이용한 뇌 네트워크 분석을 연구하였으나, 최근에는 기계 학습에 빠져서 이를 이용한 의료 영상 복원에 적용해보고 있다. 처음 가본 NIPS 2016 학회에서 GAN을 접하고 매우 흥분하여 열심히 가지고 놀아보고 있지만 사실 당장 졸업에 필요한 연구 주제와는 전혀 관계가 없다는 것이 함정. 딥러닝은 이제 갓 1년차 공부 중인 초짜 대학원생이며, 혼자 공부하던 것을 블로그에 정리하던 것이 취미였는데 많은 분들이 좋아해 주셔서 매우 신이 나있다. 최근 졸업이 다가오면서 외부활동이 점차 줄어들고 있지만, 앞으로 더 많은 분들과 교류하고 배워 이 분야에서 내가 아는 것을 꾸준히 나눌 수 있는 사람이 되는 것이 꿈이다.

Generative Adversarial Networks

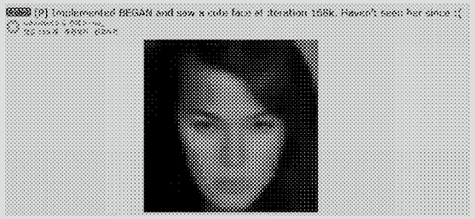
GAN 이전에도 생성 모델(generative model)은 매우 다양하게 있었는데 왜 하필 GAN만 유독 이렇게 난리일까? "잘되니까."

[그림 1]은 구글에서 올해 3월에 발표한 BEGAN 모델로 생성한 이미지들이다.

[그림 1] BEGAN 결과

즉, 이 사진들은 모두 실존하는 인물의 것이 아니란 것이죠. 상당히 그럴듯하지 않은가? 자세히 뜯어봐도 실제 사람의 사진인 것으로 착각할 정도이다. 워낙 (이미지 생성이) 잘되다 보니 우리나라로 치면 디시인사이드(dcinside.com)와 같은 레딧(Reddit)의 'Machine Learning' 게시판에서는 한때 [그림2]와 아래의 글이 게시되어 세간의 눈길을 끌기도 했다.

[그림 2] Reddit에 게시된 이미지



"BEGAN으로 학습시키던 중 16만8천 번째쯤 귀여운 그녀를 보았습니다. 그리고... 다시는 보지 못했어요. :("

사실 GAN이란 모델을 이렇게 잘 학습시키는 것은 쉬운 일이 아니다. 그 이유에 대해서는 나중에 차차 설명하도록 하고, GAN의 원리에 대해 먼저 소개하고자 한다.

GAN의 개념

GAN은 이름만 뜯어봐도 큰 줄기를 알 수 있다:

· Generative : CNN을 이용한 이미지 분류기(classifier)와 같이 이미지의 종류를 구별하는 것이 아닌 이미지를 만들어 내는 방법을 배우는 '모델'이라는 것을 알 수 있다.

· Adversarial: 이 단어의 사전적 의미는 "대립하는, 적대하는"이다. 대립하려면 상대가 있어야하니 GAN은 크게 두 부분으로 나뉘어 있다는 것을 직관적으로 알 수 있다.

· Network : 뉴럴 네트워크을 사용해서 모델이 구성되어 있다.

GAN은 이미지를 만들어 내는 네트워크(generator)와 이렇게 만들어진 이미지를 평가하는 네트워크(discriminator)가 있어서 서로 대립(adversarial)하며 서로의 성능을 점차 개선할 수 있는 구조로 만들어져 있다. 좀 더 직관적으로 이해하고 싶다면, generator를 지폐위조범, discriminator를 경찰이라 생각해 보면 된다.

> **지폐위조범(generator)은 위조 지폐를 최대한 진짜와 같이 만들어 경찰을 속이기 위해 노력하고, 경찰(discriminator)은 이렇게 위조된 지폐를 진짜와 구별(classify) 하려고 노력한다. 이런 과정을 반복하면서 두 그룹이 각각 서로를 속이고 구별하는 능력이 발전하게 된다. 궁극적으로는 지폐위조범의 솜씨가 매우 좋아져서 경찰이 더 이상 진짜 지폐와 위조 지폐를 구별할 수 없을 정도(구별할 확률 p=0.5)가 된다는 것.**

앞선 예시를 수학적 용어를 섞어 표현하면 다음과 같다. Generative 모델 G는 우리가 갖고 있는 실제 이미지 x의 분포(data distribution)를 알아내려고 노력한다. 만약 G가 정확히 데이터 분포를 모사할 수 있다면, 이 분포로부터 뽑은 (혹은 생성한) 샘플 이미지는 진짜 이미지와 전혀 구별할 수 없다.. 즉, G는 $z \sim p_z$일 때 가짜 이미지 샘플G(z)의 분포 p_g를 정의하는 모델로 생각될 수 있다. 여기서 z는 G라는 샘플링 모델에 들어갈 입력(input) 값이다. 보통 z의 분포 p_z는 가우시안(Gaussian) 분포를 사용하는데 이 부분은 차차 설명하겠다.

한편, discriminator모델 D는 현재 자기가 보고 있는 샘플이 진짜 이미지 x인지 혹은 G로부터 만들어진 가짜 이미지 G(z)인지 구별하여 샘플이 진짜일 확률을 계산한다. 앞서 예를 든 것처럼 $p_g = p_{data}$가 된다면 각각의 분포로부터 뽑힌 샘플만을 가지고 어느 쪽에서 왔는지 구별할 방법이 없기 때문에 D가 할 수 있는 최선은 '동전 던지기'이다. 즉, 임의의 샘플 x에 대해서는 [수식 1]처럼 나타낼 수 있다.

[수식 1]

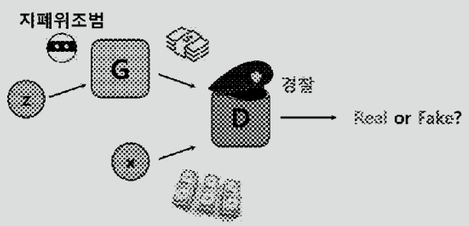
$$D(x) = \frac{1}{2}$$

최소최대의 문제 (minimax problem)

우리의 목적은 generator는 점점 더 실제 이미지와 닮은 샘플을 생성하게 하고 discriminator는 샘플을 점점 더 잘 구별하는 모델을 만드는 것이다. 따라서 [그림 3]을 보면 알 수 있듯이, D의 입장에서는 data로부터 뽑은 샘플 x는 D(x) = 1이 되고, G에 임의의 input z 넣고 만들어진 샘플에 대해서는 D(G(z)) = 0이 되도록

노력한다. 즉, D는 실수할 확률을 낮추기(minimize) 위해 노력하고 반대로 G는 D가 실수할 확률을 높이기(maximize) 위해 노력하는데, 따라서 둘을 같이 놓고 보면 "minimax two-player game or minimax problem"이라 할 수 있다.

[그림 3] 지폐위조범 VS 경찰



이를 수식으로 정리하면 우리가 하고자 하는 것은 다음과 같은 가치 함수(value function) V(G, D)에 대한 최소최대 문제(minimax problem)를 푸는 것과 같아진다.

[수식 2]

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

무엇이든지 간에 이런 수식이 있으면, 극단적인 예시를 넣어 이해하는 것이 빠르다. 먼저 G 입장에서 가장 이상적인 상황을 생각해 보자. G가 진짜 이미지와 완벽히 닮은 샘플을 만들고, G(z)가 만들어 낸 이미지가 진짜일 확률이 1이라고 D가 잘못된 결론을 내린다면, D(G(z))=1이므로 두 번째 항의 값이 $-\infty$가 된다. 이 때 G의 입장에서 V가 "최솟값"이라는 것은 자명하다.

반면에 D가 진짜 이미지와 가짜 이미지를 완벽하게 잘 구별하는 경우, 현재 보고 있는 샘플 x가 실제로 data distribution으로부터 온 녀석일 때 D(x) =1, 샘플이 G(z)가 만들어 낸 녀석이라면 D(G(z))= 0이 되므로 위 수식 첫 번째 항과 두 번째 항이 모두 0으로 사라지죠. 이 때 D의 입장에서 V의 "최댓값"을 얻을 수 있다는 것 역시 자명하다.

이제 드디어 우리가 풀 문제를 수식으로 명확히 정의하였다. 그런데 보시다시피 이 문제는 변수 G와 D, 두 개가 서로 엮여 있다. 이런 관계 덕분에 서로에게 피드백을 줘서 성능이 각각 좋아지기도 하지만, 한편으로 한쪽 모델에 대해 문제를 풀면 다른 한쪽은 손해를 보기 때문에 둘 모두를 만족시키는 평형점을 찾기가 쉽지 않다.

따라서 실제로 문제를 풀기 위해서는 한쪽을 상수로 고정하고 다른 변수에 대해 문제를 푸는 방식의 접근을 할 수밖에 없다. 예를 들자면, 먼저 현재 생성 모델을 G'으로 고정하고, D와 관련해서는 다음 문제를 먼저 풀어서 D'을 구한다. [수식 3] 이렇게 얻은 D'를 넣고 G에 대해서는 다음 문제를 번갈아 푸는 전략을 쉽게 떠올려 볼 수 있다. [수식 4]

[수식 3]

$$\max_D V(D, G')$$

[수식 4]

$$\min_G V(D', G)$$

이론적 근거(theoretical results)

이제 방법도 알았고 문제를 열심히 잘 풀기만 하면 될 것 같지만, 그러기 전에 아직 해결해야 할 것들이 몇 가지 남아 있다. 먼저 우리가 정의한 이 문제가 실제로 정답이 있는지(existence), 만약 해가 존재한다면 유일한지(uniqueness) 확인할 필요가 있다. 그리고 제안한 방법이 실제로 원하는 해를 찾을 수 있는지(convergence) 확인해야 한다. 애초에 답이 없는 문제를 풀거나, 답이 있더라도 여러 가지이거나, 풀 방법을 제안했는데 그 방법이 해로 수렴한다는 보장이 없으면 여러 모로 곤란해진다.

따라서 이제부터는 이 부분들을 하나씩 체크해 보겠다. 이 장에서 중요한 내용은 크게 두 가지로 나눌 수 있다.

1) 최소최대 문제(minimax problem)가 $p_g = p_{data}$에서 전역해 (global optimum)로 유일해(unique solution)를 갖는다는 것과
2) 알고리듬이 global optimum로 수렴한다는 것이다.

일단, 첫 번째 주제인 global optimality에 관한 얘기를 하기 위해서는 먼저 가져가야 할 도구가 하나 있다. 임의의 G에 대하여, 최적의 discriminator D는 다음과 같다.

[수식 5]

$$D^*_G(x) = \frac{P_{data}(x)}{p_{data}(x) + p_a(x)}$$

위 수식을 증명하는 것은 크게 어렵지 않다. 먼저 V(G, D)를 다음과 같이 풀어 쓸 수 있다.

[수식 6]

$$\begin{aligned} V(G,D) &= E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1-D(G(z))) \\ &= \int_x P_{data}(x)\log(D(x))\,dx + \int_z P_z(z)\log\left(1-D(G(z))\right)dz \\ &= \int_x P_{data}(x)\log(D(x)) + p_g(x)\log(1-D(x))\,dx \end{aligned}$$

마지막 식의 형태를 참고하면, D*을 구하는 문제는 임의의 [수식 7]에 대하여.

[수식 7]

$$(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$$

[수식 8] 함수에서 최댓값을 구하는 문제로 단순화할 수 있다.

[수식 8]

$$y \to a\log(y) + b\log(1-y)$$

위 함수는 [수식 9]에서 최댓값을 갖기 때문에 이것으로 증명이 완료된다.

[수식 9]

$$y^* = \frac{a}{a+b}$$

결국, 위 결과를 바탕으로 최소최대의 문제(minimax problem)을 다시 써 보면 이제 변수가 G 하나인 C(G)라는 문제로 나타낼 수 있다.

[수식 10]

$$\begin{aligned} C(G) &= max_D V(G,D) \\ &= E_{x\sim p_{data}}[\log D^*_G(x)] + E_{x\sim p_z}[\log(1-D^*_G(G(z))] \\ &= E_{x\sim p_{data}}[\log D^*_G(x)] + E_{x\sim p_g}[\log(1-D^*_G(x)] \\ &= E_{x\sim p_{data}}\left[\log\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right] + E_{x\sim p_g}\left[\log\frac{p_g(x)}{p_{data}(x)+p_g(x)}\right] \end{aligned}$$

전역적 최적해(global optimum) 증명

이제 이 도구를 사용해서 주요 정리(main theorem)을 증명해 보고자 한다. C(G)의 global minimum은 오직 $p_g = p_{data}$으로 유일하게 존재하며, 이때 C(G)의 값은 -log(4)이다.

증명은 양방향으로 진행된다. 먼저 가장 이상적으로 $p_g = p_{data}$ 일 때 C(G)가 어떤 값을 갖는지 확인하려 한다. 앞서 구한 [수식 5]에 $p_g=p_{data}$를 입력하면 [수식 11]이 나오고

[수식 11]

$$D^*_G(x) = \frac{1}{2}$$

이를 [수식 2]에 대입하면 다음과 같이 나타낼 수 있다.

[수식 12]

$$C(G) = \mathbb{E}_{x\sim p_{data}}[-log(2)] + \mathbb{E}_{x\sim p_g}[-log(2)] = -log(4).$$

한편, C(G) 수식을 바꿔 표현해서 -log(4)가C(G)가 가질 수 있는 유일한 최적값임을 알기 위해서는 다음과 같이 C(G)를 표현하는 것이 가장 중요하다.

[수식 13]

$$\begin{aligned} C(G) &= C(G) + \log(4) - \log(4) \\ &= -\log(4) + KL(p_{data} \,\|\, \tfrac{P_{data}+p_g}{2}) + KL(p_g \,\|\, \tfrac{P_{data}+p_g}{2}) \\ &= -\log(4) + 2\cdot JSD(P_{data} \,\|\, p_g) \end{aligned}$$

첫 번째 등호에서 두 번째 등호로 넘어가기 위해서는 쿨백-라이블러 발산(Kullback-Leible divergence)에 대한 정의를 알아야 한다. 쿨백-라이블러 발산은 P라는 확률 분포와 Q라는 확률 분포가 있을 때 두 분포가 얼마나 다른지를 측정하는 값이다.

[수식 14]

$$D_{KL}(P\|Q) = \sum_i P(i) log \frac{P(i)}{Q(i)}.$$

P와 Q의 분포가 일치하면 log 안의 값이 1이 되어 divergence가 0이 되는 것을 알 수 있다. 따라서 첫 번째 수식의 C(G)+log(4)를 풀어서, [수식 15]와 같이 나타낼 수 있다.

[수식 15]

$$E_{x\sim p_{data}}\left[\log\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right] + E_{z\sim p_x(z)}\left[\log\frac{p_g(x)}{p_{data}(x)+p_g(x)}\right] + \log(2) + \log(2)$$

이와 같이 나타낸 후 각 기댓값(expectation) 안으로 log(2)를 넣어 주면 두 번째 등호의 수식 형태로 나오게 된다. 두 번째에서 세 번째 등호도 JSD(Jensen-Shannon divergence)의 정의를 알면 자연스럽게 따라온다.

[수식 16]

$$JSD(P \,\|\, Q) = \tfrac{1}{2}D_{KL}(P \,\|\, M) + \tfrac{1}{2}D_{KL}(Q \,\|\, M).$$

[수식 17]

$$M = \frac{1}{2}(P+Q)$$

여기서 [수식 17]이므로 그대로 원 수식에 비교 대조해 보면, 쉽게 유도할 수 있다. 결국 JSD는 분포 간의 차이를 나타내는 값으로 항상 양수이며, 비교하는 두 분포가 일치할 때만 값이 0이기 때문에 C*=-log(4)가 C(G)의 global minimum이며 그 유일한 해가 $p_g=p_{data}$임을 알 수 있다.

앞서 정의한 minimax problem을 잘 풀기만 하면(즉, 전역적 최적해를 찾으면), generator가 만드는 확률분포(probability distribution) p_g가 data distribution p_{data}와 정확히 일치하도록 할 수 있다. generator가 뽑은 샘플을 discriminator가 실제와 구별할 수 없게 된다는 것이다.

수렴성(convergence) 증명

남은 것은 앞서 제시한 것과 같이 G와 D에 대해 번갈아 가며 문제를 풀었을 때, $p_g = p_{data}$를 얻을 수 있는지 확인하는 것이다. 여기서 증명을 용이하게 하기 위해 몇 가지 조건에 대한 상정(想定)이 필요하다. 먼저 G와 D라는 모델이 각각 우리가 원하는 데이터의 확률 분포를 표현하거나 구별할 수 있는 모델을 학습할 수 있을 만큼 용량(capacity)이 충분하고, discriminator D가 주어진 G에 대해 최적값인 D*으로 수렴한다고 가정합니다.

위 가정이 충족되었을 때, [수식 18]에 대하여 G의 최적값을 구하면 p_g가 p_{data}로 수렴한다.

[수식 18]

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

먼저 D가 고정일 때, V(G,D)=U(p_g,D)라 표현하려 한다. 증명의 방향은 먼저 U(p_g, D)가 p_g에 대해서 볼록(convex) 함수임을 확인하는 것이다. U(p_g, D)의 유일한 해가 $p_g = p_{data}$라는 것은 앞선 정리에서 확인하였기 때문에 이로써 증명이 끝나게 된다. 정의에 의해 [수식 19]가 도출된다.

[수식 19]

$$U(p_g, D) = \int_x p_{data}(x)\, log\big(D(x)\big) + p_g(x)\, log\big(1 - D(x)\big)\, dx$$

p_g에 대해 미분을 하면 log(1-D(x))만 남게 되고 이는 p_g에 대해서 상수이기 때문에 U(p_g, D)가 선형 함수임을 알 수 있다. 그러므로 [수식 20]이 p_g에 대해 볼록 함수*로 유일한 전역해(global optimum)를 갖는다.

[수식 20]

$$\sup_D U(p_g, D)$$

따라서 p_g에 대한 적은 수의 gradient decent 계산만으로도*1 [수식 21]이 도출될 수 있다.

[수식 21]

$$p_g \rightarrow p_{data}$$

앞서 제안한 전략이 전역적 최적해로 해가 수렴한다는 것까지도 증명했다. 단! 지금까지 전개한 논리들에는 몇 가지 꼭 알고 넘어가야 할 내용들이 있다.

첫째, 지금까지 논리를 전개하면서 G와 D를 만들 모델에 대해 어떤 제약을 두지 않았다. 즉, 여기서 G와 D는 무한대의 표현 용량을 가진(임의의 어떤 함수라도 표현할 수 있는) 모델로서 앞서 진행한 증명들은 모두 확률 밀도 함수(probability density function) 공간에서 전개된 것이다. 그러나 실제로 모델을 만들 때 위와 같은 non-parametric 모델을 만드는 것은 현실적으로 어려움이 많다.

그런데 뉴럴 네트워크은 충분한 용량만 주어진다면 보편 근사기(universal function approximator)로서 역할을 잘해 주며, 미분이 가능한 함수들로 표현되므로 역전파(backpropagation)라는 매우 직관적이고 효율적인 학습이 가능하다. 또한 뉴럴 네트워크은 확률분포 (probability distribution)를 학습하기 위해 마르코프 사슬 몬테 카를로(Markov Chain Monte Carlo, MCMC)와 같은 다소 소모적인 방법을 사용하지 않아도 되기 때문에 상대적으로 빠르다. 이러한 여러 장점들 때문에 실용적인 측면에서 GAN에서는 뉴럴 네트워크을 활용한다.

각각의 G와 D를 단순한 뉴럴 네트워크인 다중층 인식망(multilayer perceptron, MLP)으로 모델을 만들면, G는 입력값(input) z에 대해 출력값(output)으로 이미지 샘플을 뽑아 주는 G(z; θ_g)로 표현할 수 있다. 한편, D 역시 D(x; θ_d)로 나타내며 output으로 샘플 x가 p_{data}로부터 뽑혔을 확률값을 내보낸다. 여기서 θ_g와 θ_d는 각 MLP의 매개변수(parameter)이다.

그러나 이렇게 함수 공간을 일반적인 공간에서 뉴럴 네트워크이 표현할 수 있는 공간으로 바꾸게 되면, GAN이 표현할 수 있는 p_g가 G(z;θ_g) 함수로 나타낼 수 있는 종류로 제한이 된다. 뿐만 아니라 더 이상 p_g에 대해 직접 최적화 문제를 푸는 것이 아닌 θ_g에 대해 최적화 문제를 푸는 것으로 모든 구조가 바뀌기 때문에 앞서 증명들에서 사용한 가정들이 모두 깨지게 된다. 즉, MLP를 모델로 사용하면 G가 모수 공간(parameter space)에서 다중임계점(multiple critical point)을 가지기 때문에 완전히 증명한 바와 합치하지는 않는다. 다만 실제로 학습을 해 보면 성능이 잘 나오는 것으로 미루어 봤을 때, MLP가 이론적 보장이 좀 부족할지라도 실용적으로 쓰기에는 합리적인 모델이라고 할 수 있다.

둘째, 수렴성 증명에 들어간 가정들은 '생각보다' 매우 강력한 제약이다. 실제로는 계산량이나 학습 시간에 한계가 있기 때문에 D*를 얻기 위해 D가 수렴할 때까지 학습시키는 것은 현실적으로 어렵다. 게다가 모델의 용량이 p_{data}를 학습하기에 충분한지에 대한 문제 역시 다루기 어렵다. 최대한 이론적 상황과 비슷하게 맞춰 주기 위하여 D에 대한 학습을 G에 비해 보다 많이 수행해 주는 등 기술적인 노하우를 사용하기는 하지만 근본적인 해결책이라 할 수는 없다. 따라서 어떤 면에서는 유명무실한 증명이라 할 수도 있다.

셋째, 심지어 구현 시 사용하는 가치 함수(value function)의 형태가 이론과 다르다. 실제로는 $\min_G$ log(1-D(G(z))) 대신 $\max_G$ log(D(G(z)))로 모델을 바꾸어 학습시킨다. 학습 초기에는 G가 매우 이상한 이미지를 생성하기 때문에 D가 너무도 쉽게 이를 진짜와 구별하고 log(1-D(G(z)))의 기울기가 아주 작아서 학습이 엄청 느리다. 문제를 G = $\max_G$ log(D(G(z)))로 바꾸면, 위와 같은 문제가 생기지 않기 때문에 쉬운 해결 방법이다. 하지만 문제를 다른 형태로 바꿨기 때문에 이 역시도 아쉬운 점이라 할 수 있다.

이로써 GAN에 대해 원 논문에서 나온 이론적 증명까지 모두 살펴보았습니다. 다음 호에서는 이런 배경지식을 바탕으로 기존 생성 모델과 GAN의 차이점에 대해 살펴보고 GAN의 특징과 장단점에 대해 본격적으로 소개하겠다.

*1 참고 | P_g에 대해 볼록함수이기 때문에 이 함수의 최소값을 구하는 것을 gradient descent로 iterative하게 찾으면 적은수의 업데이트 만에 최소값 P_g를 찾게 된다.