Ian J. Goodfellow

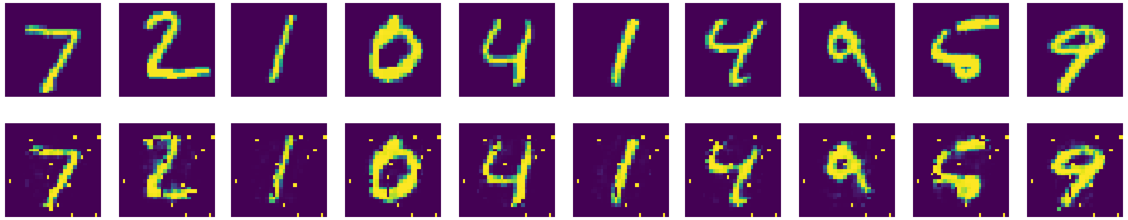
->google brain

->made GAN

->author of Deep Learning

. Deep generative models

- Autoencoder
- Generative Adversarial Networks



Not same!

https://github.com/gudrb/GAN/blob/master/autoencoder.ipynb

Deep generative models have had less of an impact

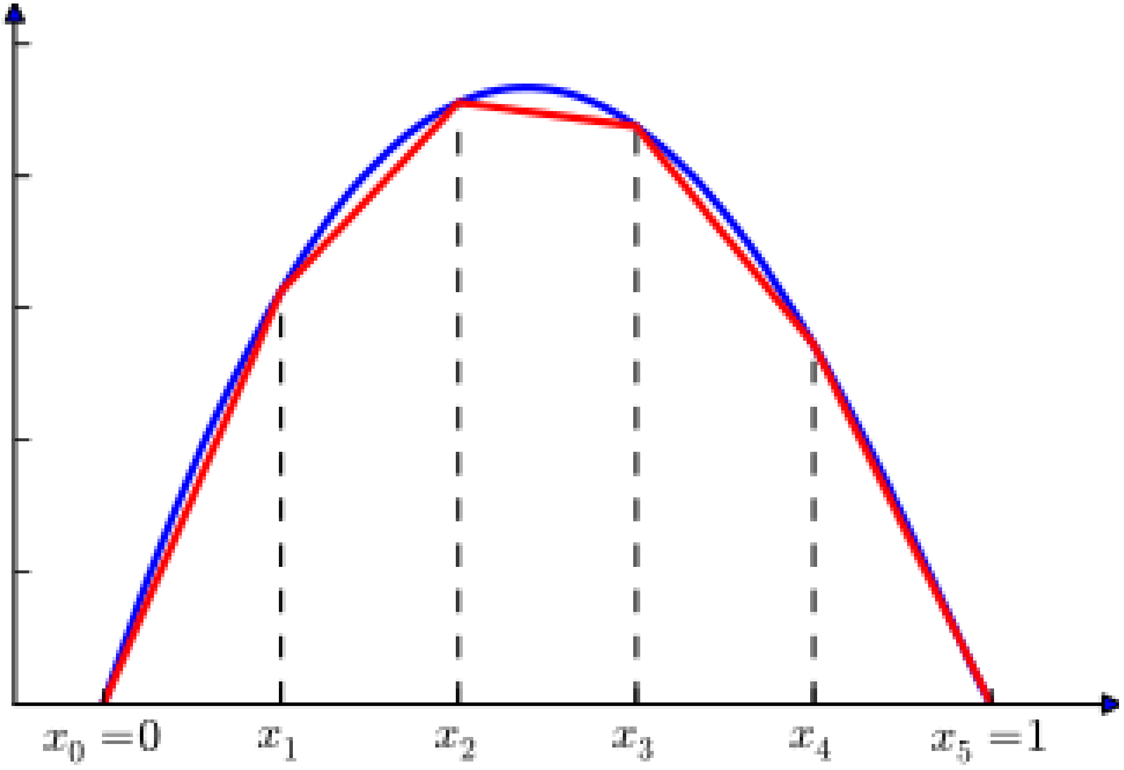
- Difficulty of Maximum likelihood estimation
- Difficulty of leveraging the benefits of piecewise linear units in the generative context

Succession of Deep generative models

- Backpropagation
- Drop out algorithm

Generative Adversarial Networks

Piecewised-linear



adversarial process

- 대립하는, 적대하는

-> generator, discriminator (two models(G,D))

G:capture data distribution

D:estimates the probability of real sample

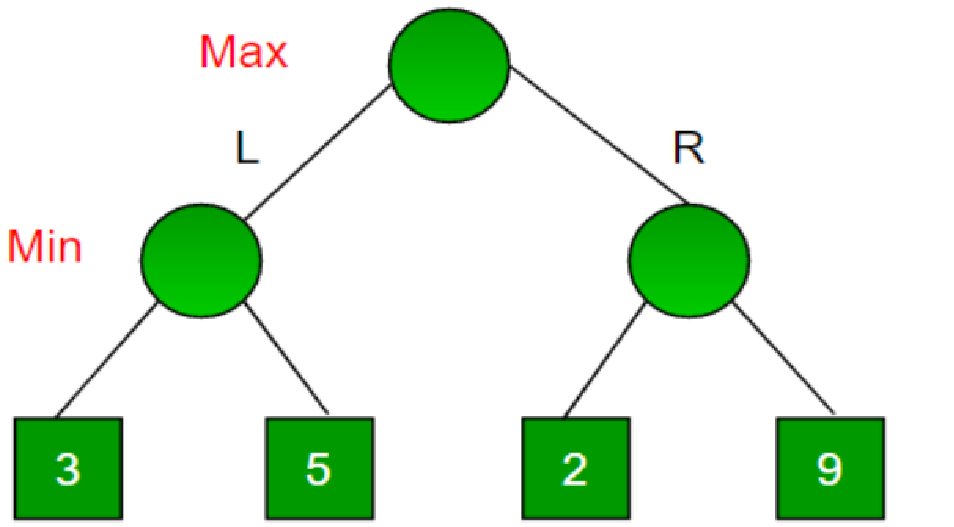
=>minimax two-player game

minimax two-player game

- Maximizer, Minimizer

Maximizer: effort to get high score

Minimizer: effort to get low score



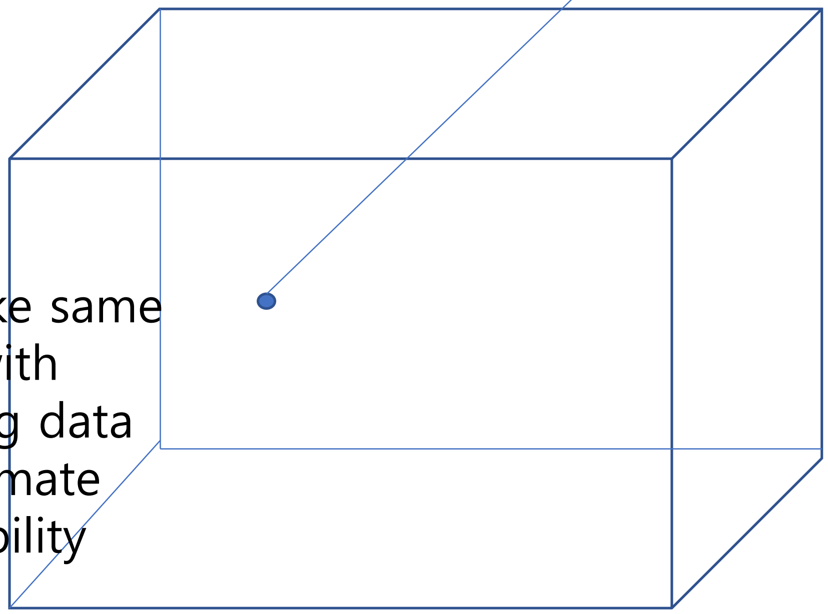
Max – right
Min – left
⇒ 3

Max – left
Min – right
⇒ 2
Choice: 3

Backtracking

Unique answer

(G,D)
G: make same data with training data
D: estimate probability =1/2



D and G play the following two-player minimax game with value function V (G, D):

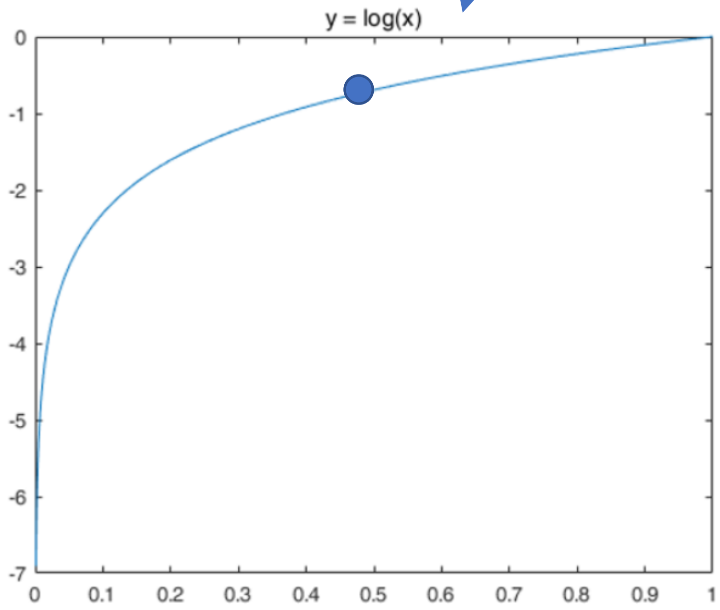
x: data variable , z: noise variable, G(z): fake data

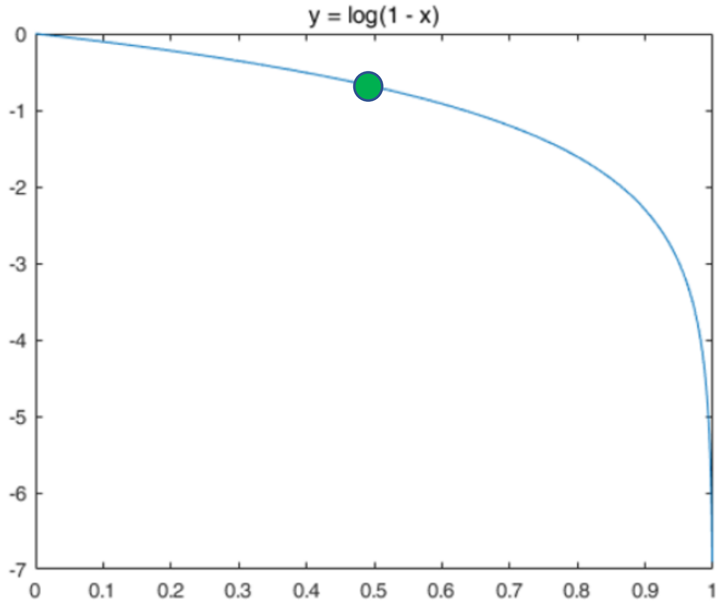
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

D:
D(x)=1
D(G(x))=0

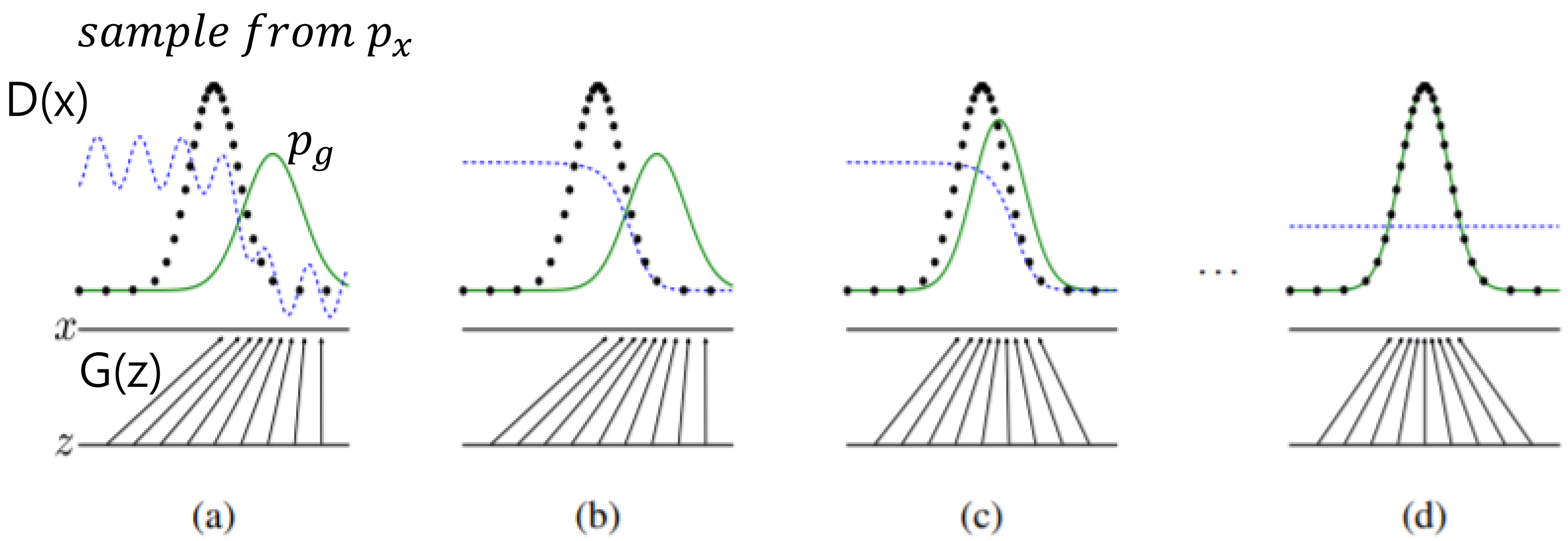
G:
D(x)=0
D(G(x))=1

D(x)=1/2
D(G(x))=1/2





Optimizing D inner loop is prohibitive

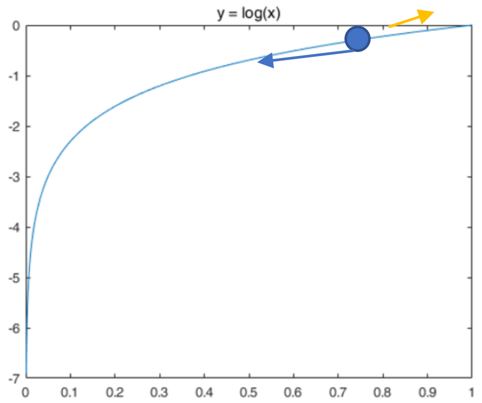


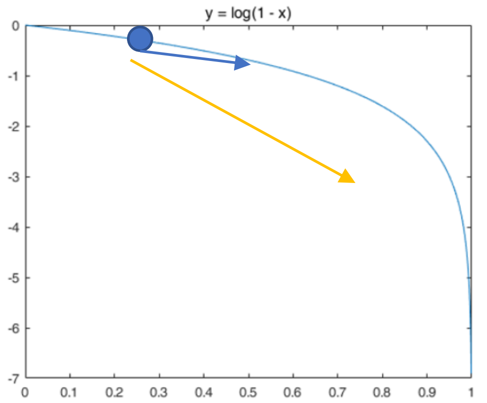
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1-D(G(\boldsymbol{z})))].$$

k steps of optimizing D and one step of optimizing G
D: near optimal solution
G: change slowly enough

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

If not,
D: overfitting
G: change fastly
(yellow arrow)





Proof of optimal D

f(x)를 가지는 연속형 확률변수의 기대값(Expectation Value) E(X) = $\int_{-\infty}^{\infty} xf(x)dx$

$$V(G,D) = \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_{z} p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz$$

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

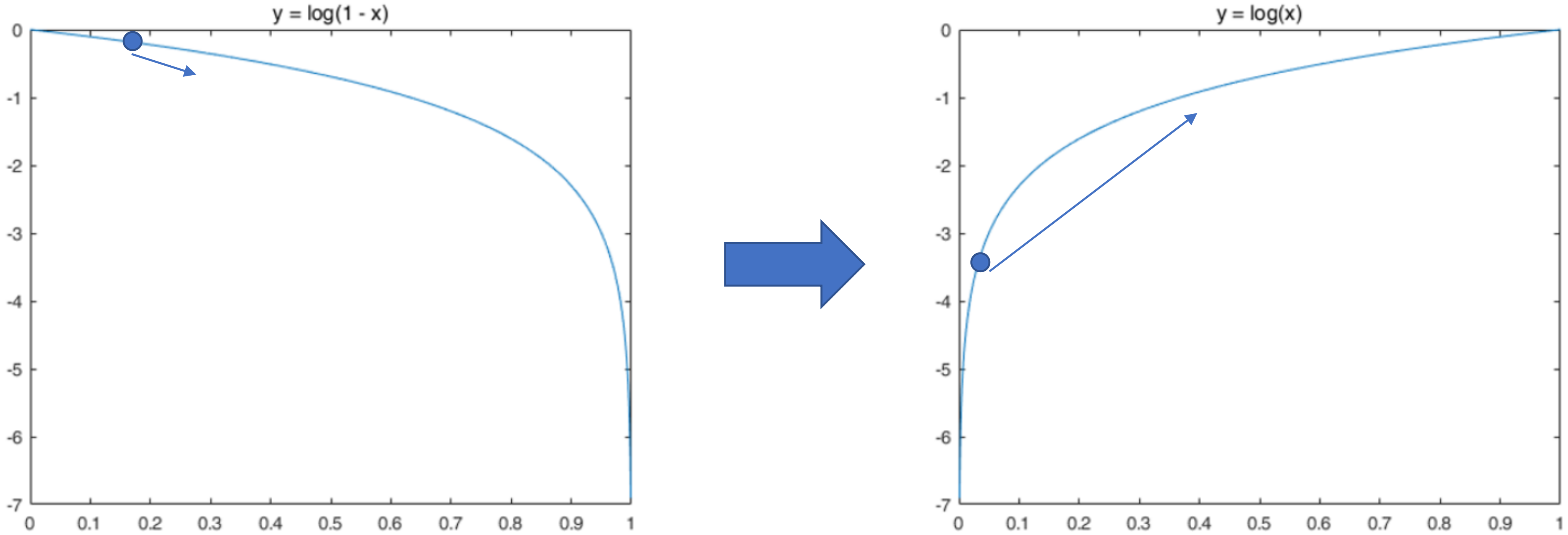
-> y=$\frac{a}{a+b}$

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

Global min: $p_{data} = p_x$

-> -log4

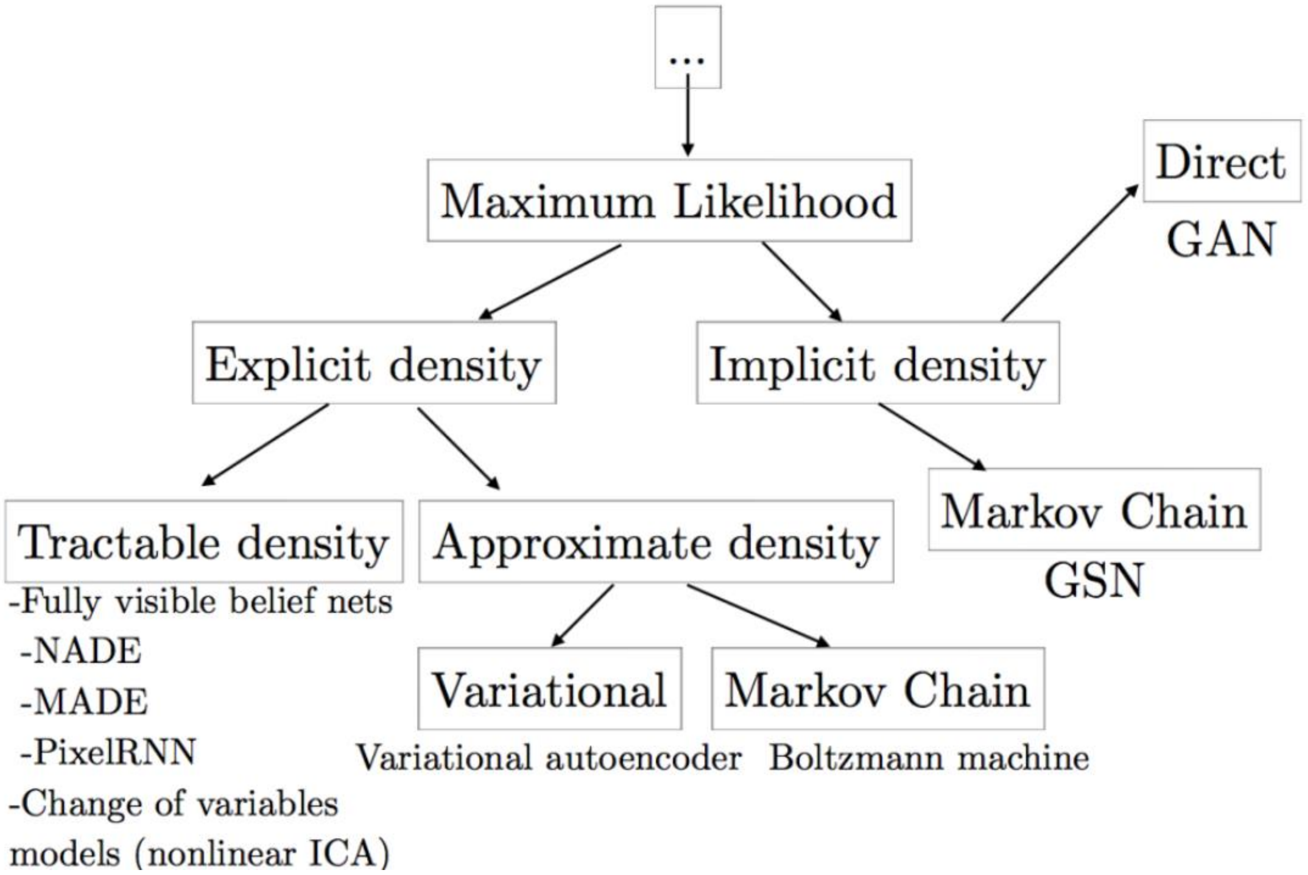
For G to learn more well



G : log(1-D(G(z)) Minimize

G : log(D(G(z)): Maximize

Explicit, Implicit



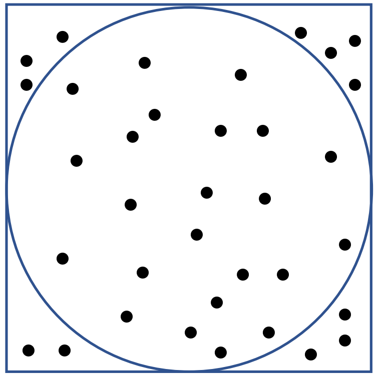
Markov chains

- Markov chains or unrolled approximate inference networks during either training or generation of samples ??
- s. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples. ??
- and sample from the generative model using only forward propagation

MCMC(Markov Chain Monte Carlo)

- 어떤 목표 확률분포(Target Probability Distribution)로부터 샘플링을 통해 랜덤 샘플을 얻는 방법이다

Monte Carlo



원의넓이: $\frac{\text{원 안의 점 갯수}}{\text{전체 점 개수}} \times 4$

Markov Chain <-> stochastic sampling(mini batch)

$$P(X_k|X_{k-1}, X_{k-2}, \ldots, X_1, X_0) = P(X_k|X_{k-1})$$

몇 가지 추가 조건이 만족한다면, k가 충분히 커지면 Xk의 분포는 특정한 값으로 수렴한다
즉, stationary distribution 된다.

목표분포를 stationary distribution으로 만들고 샘플링 한다.
샘플은 stationary distribution을 따른다

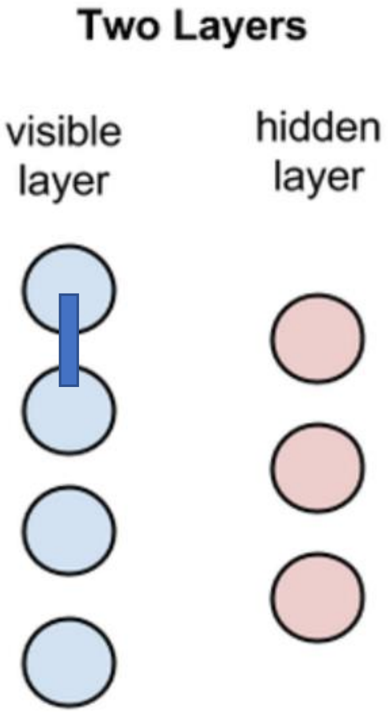
-> gradient 측정가능

Restricted Boltzmann Machines
-> Markov chain

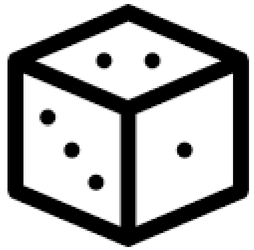
- *While RBMs are occasionally used, most practitioners in the machine-learning community have deprecated them in favor of generative adversarial networks or variational autoencoders.*

Restricted: no intra-layer communication

- Undirected layer



Probability Density Function, PDF

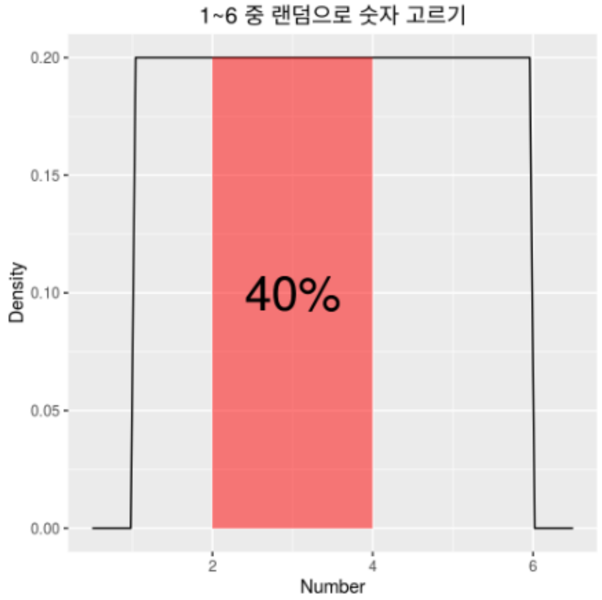


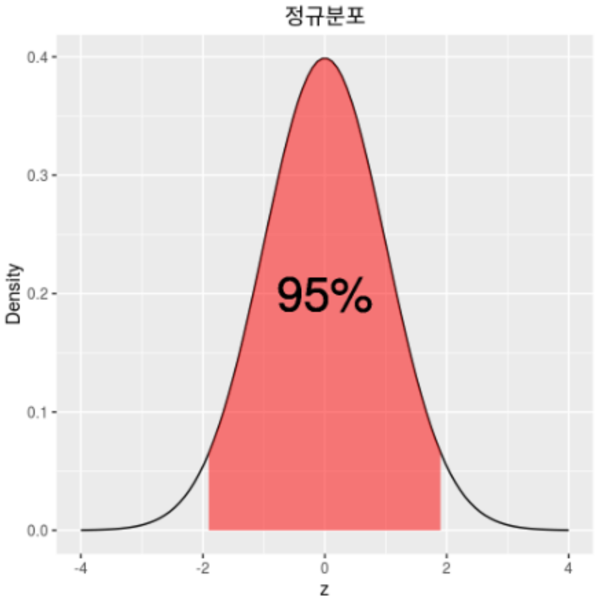
	probability
1	1/6
2	1/6
3	1/6
4	1/6
5	1/6
6	1/6

	probability
1	1/∞
2	1/∞
3	1/∞
4	1/∞
5	1/∞
6	1/∞

0~6

	probability
0~1	1/6
1~2	1/6
2~3	1/6
3~4	1/6
4~5	1/6
5~6	1/6





정규분포

	probability	likelyhood
0(MLE)	0	0.4
999	0	0

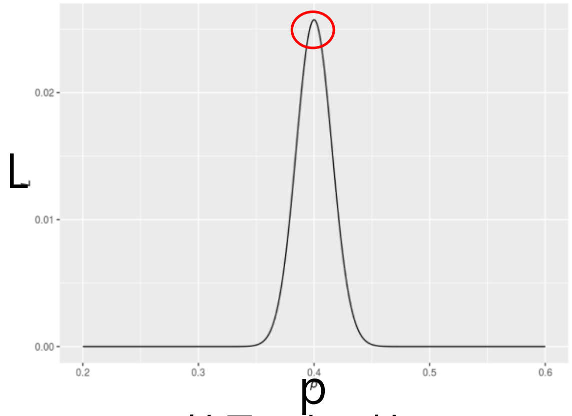
- 가능도의 직관적인 정의 : 확률분포함수의 y값
  - 셀 수 있는 사건: **가능도 = 확률**
  - 연속 사건: **가능도 $\neq$ 확률, 가능도 = PDF값**

Maximum likelihood estimation

- **최대우도법**(最大尤度法)은 어떤 확률변수에서 표집한 값들을 토대로 그 확률변수의 모수를 구하는 방법이다. 어떤 모수가 주어졌을 때, 원하는 값들이 나올 가능도를 최대로 만드는 모수를 선택하는 방법이다.

모수: 동전 앞면 나올 확률(p)
표집 값: 1000번 던짐
가능도: 모수에 따른 앞면이 나올 확률 $L = {}_{1000}C_{400}p^{400}(1-p)^{600}$

앞: 400
뒷: 600
$L = {}_{1000}C_{400}p^{400}(1-p)^{600}$



확률 밀도함수

미분후
L′ =0
인 p 값 구하면됨