2016-12-12
TF Study

GAN

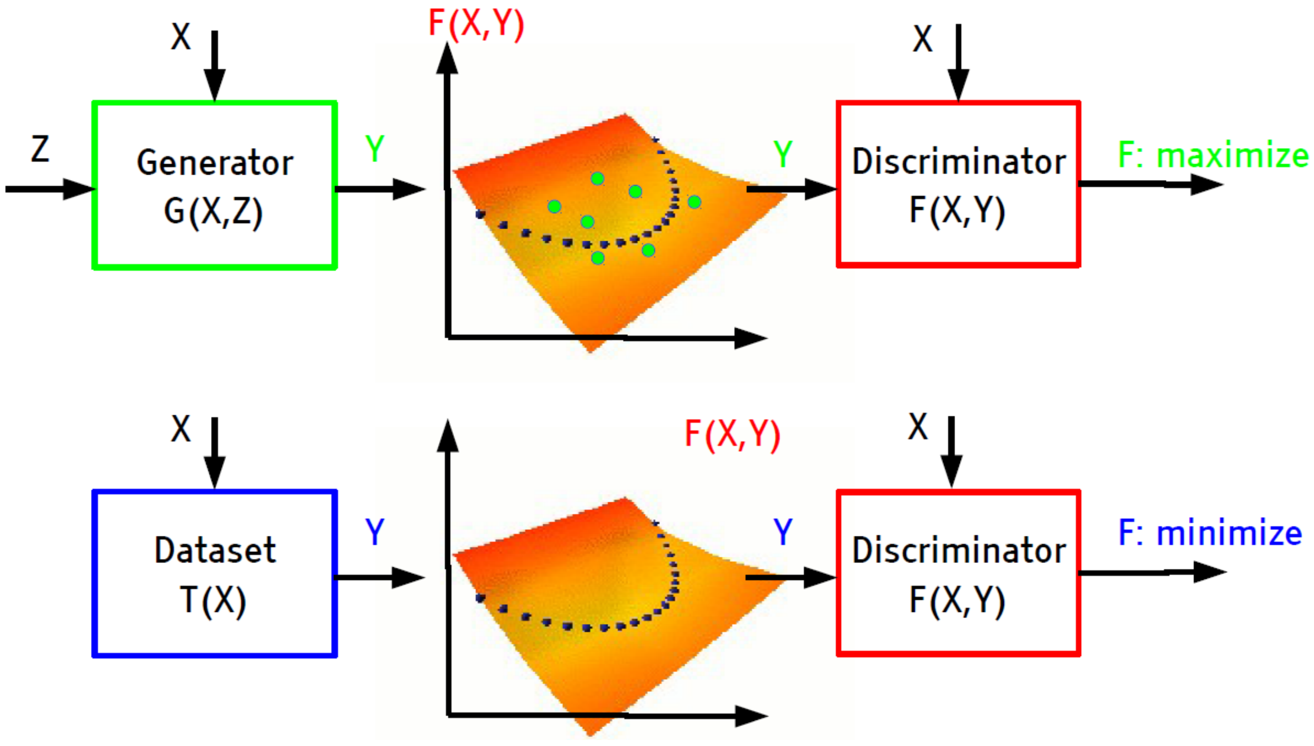
CONTENTS

- Purpose:
  1. Train the manifold of data
  2. Generate distribution of data

- Process:
  1. Generator에 Input X & Z
  2. Generator produces predictions Y
  3. Discriminator discriminate train data & Y
  4. Train

  G : D가 Y와 Train data를 구별할 수 없도록

  D : G가 만든 데이터와 실제 데이터를 잘 구별하도록



Yann Lecun : nips 2016 keynote https://drive.google.com/file/d/0BxKBnD5y2M8NREZod0tVdW5FLTQ/view

- Method:
  - $p_z(z)$: *prior on input noise, D(x) : probability that* ***x*** *came from data rather than* p_g
  - D and G play the following two-player minimax game with value function V(G,D)

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

Proved)

- $p_g = p_{data}$ 인 Glabal optimal 존재 ➔ 학습이 잘 되면 G가 data의 원래 분포를 그대로 따라가는 것이 가능
- G, D 에 충분한 capacity ➔ converge 보장

- Method:

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**
   • Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
   • Sample minibatch of m examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
   • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right].$$

k 번
D 학습

  end for
  • Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
  • Update the generator by descending its stochastic gradient:

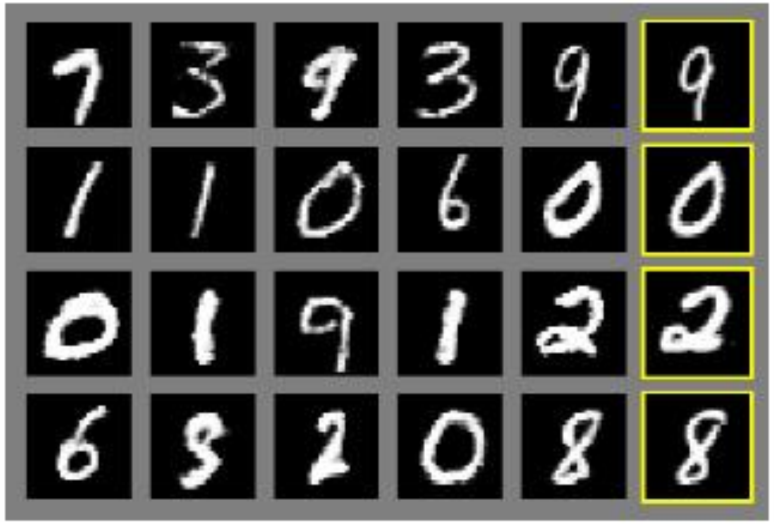
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right).$$

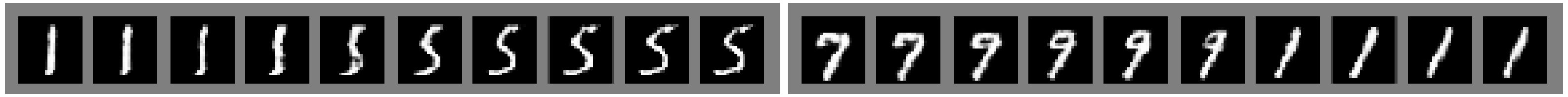
G 학습

end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

- Ian Goodfellow #AIWTB 2016 https://www.youtube.com/watch?v=HN9NRhm9waY

- Experiment:
  - MNIST , Toronto Face Database, CIFAR-10 적용
  - Generator – rectifier linear & sigmoid / Discriminator – maxout , drop out





- Ian Goodfellow #AIWTB 2016 https://www.youtube.com/watch?v=HN9NRhm9waY

- Failure:
  - Training GAN = Finding Nash Equilibrium of non-convex game
  - Gradient descent ➔ not suitable for game theory ➔ often fail to converge
  - 2-player game에서 G,D가 서로 min / max 최적화를 하려다 보니 converge 하지 않는 경우 발생

- Improved:
  - ***Feature matching*** : maximum mean discrepancy 사용
  - ***Mini batch features*** : batch normalization 적용

- Salimans, Tim, et al. "Improved techniques for training gans." *Advances in Neural Information Processing Systems*. 2016.

- ***Feature matching***
  - Prevent Generator from overtraining on the current Discriminator
  - D를 통해서 나온 output 을 사용한 학습 ➔ 중간 레이어의 high-level feature들로 학습
  - New objective for generator
    $||\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\mathbf{f}(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\mathbf{f}(G(\boldsymbol{z}))||_2^2.$: $f\ (x)$: $activationson\ an\ intermediate\ layer\ of\ D$
  - Discriminator는 그대로 학습

- Salimans, Tim, et al. "Improved techniques for training gans." *Advances in Neural Information Processing Systems*. 2016.

▪ ***Mini batch features***

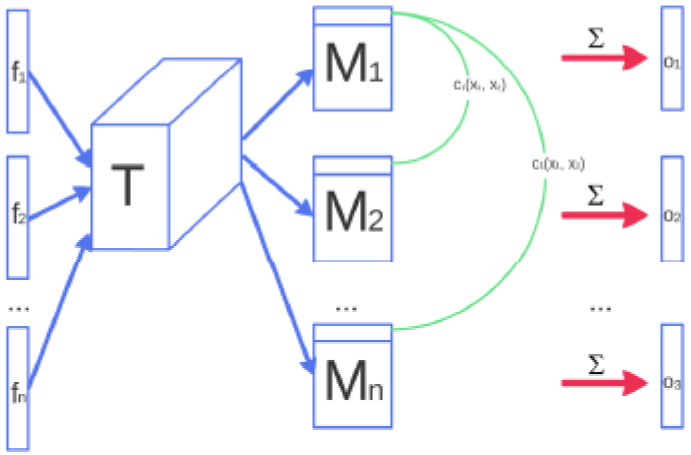
- Generator often collapse to a parameter setting / always emits the same point
- 서로 다른 데이터면 다른 곳으로 보내야하는데… 모두 같은 곳으로 보내도 objective function은 만족하는….
- 각각의 데이터를 independent 하게 확인하기 때문 ➔ 동시에 여러 개를 확인해서 조합하게끔 변형 ➔ D의 중간 부분을 변형
- Define ***closeness*** of examples in minibatch ➔ use it as regularization
- $f(x_i) \in \mathbb{R}^A,\ multiply\ tensor\ T \in \mathbb{R}^{A\times B\times C},\ result\ matrix\ M_i \in \mathbb{R}^{B\times C}$

  compute L1-distance between rows of M_i and apply negative exponential ➔ $c_b(x_i, x_j) = \exp(-\|M_{i,b} - M_{j,b}\|_{L1})$
- $o(x_i)$: output for this minibatch layer

$$o(\boldsymbol{x}_i)_b = \sum_{j=1}^{n} c_b(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathbb{R}$$

$$o(\boldsymbol{x}_i) = \left[o(\boldsymbol{x}_i)_1, o(\boldsymbol{x}_i)_2, \ldots, o(\boldsymbol{x}_i)_B\right] \in \mathbb{R}^B$$

$$o(\mathbf{X}) \in \mathbb{R}^{n\times B}$$



- Concatenate $o(x_i)$ to $f(x_i)$ and feed to next layer

- Salimans, Tim, et al. "Improved techniques for training gans." *Advances in Neural Information Processing Systems*. 2016.

- ***Other improves***
  - Historical averaging

    : D, G 의 cost에 과거 정보를 지우는 방식을 도입 ➔ online learning style
  - One sided label smoothing

    : 0,1 로 class 를 나누는 방식 말고 다른 숫자로 smoothing 해서 class 분리
  - Virtual batch normalization

    : DCGAN에서 효과적이었던 Batch normalization을 다소 수정하여 도입

- Salimans, Tim, et al. "Improved techniques for training gans." *Advances in Neural Information Processing Systems*. 2016.

- ***Semi-supervised learning***
  - sample from Generator

    : G를 통해 생성한 sample data를 추가하여 semi-supervised learning 이 가능
  - Label them with new generated class

    : 1개의 class를 추가하여 Classifier를 만듦.

    : $p_{model}(y = K + 1|x) = 1 - D(x)$ 라 하면
  - Loss function

    :
$$\begin{aligned} L &= -\mathbb{E}_{\boldsymbol{x},y\sim p_{\text{data}}(\boldsymbol{x},y)}[\log p_{\text{model}}(y|\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim G}[\log p_{\text{model}}(y = K+1|\boldsymbol{x})] \\ &= L_{\text{supervised}} + L_{\text{unsupervised}}, \text{ where} \\ L_{\text{supervised}} &= -\mathbb{E}_{\boldsymbol{x},y\sim p_{\text{data}}(\boldsymbol{x},y)} \log p_{\text{model}}(y|\boldsymbol{x}, y < K+1) \\ L_{\text{unsupervised}} &= -\{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})} \log[1 - p_{\text{model}}(y = K+1|\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim G} \log[p_{\text{model}}(y = K+1|\boldsymbol{x})]\}, \end{aligned}$$

  - ➔ effective to improve the quality of generated images as judged by human annotators

- Salimans, Tim, et al. "Improved techniques for training gans." *Advances in Neural Information Processing Systems*. 2016.

- Experiment:
  - Semi-supervised learning ➔ MNIST, CIFAR-10, SVHN dataset
  - Sample generation ➔ MNIST, CIFAR-10, SVHN, ImageNet

Model	Number of incorrectly predicted test examples for a given number of labeled samples			
	20	50	100	200
DGN [21]			333 ± 14	
Virtual Adversarial [22]			212	
CatGAN [14]			191 ± 10	
Skip Deep Generative Model [23]			132 ± 7	
Ladder network [24]			106 ± 37	
Auxiliary Deep Generative Model [23]			96 ± 2	
Our model	1677 ± 452	221 ± 136	93 ± 6.5	90 ± 4.2
Ensemble of 10 of our models	1134 ± 445	142 ± 96	86 ± 5.6	81 ± 4.3

Table 1: Number of incorrectly classified test examples for the semi-supervised setting on permutation invariant MNIST. Results are averaged over 10 seeds.

Model	Test error rate for a given number of labeled samples			
	1000	2000	4000	8000
Ladder network [24]			20.40 ± 0.47	
CatGAN [14]			19.58 ± 0.46	
Our model	21.83 ± 2.01	19.61 ± 2.09	18.63 ± 2.32	17.72 ± 1.82
Ensemble of 10 of our models	19.22 ± 0.54	17.25 ± 0.66	15.59 ± 0.47	14.87 ± 0.89

Table 2: Test error on semi-supervised CIFAR-10. Results are averaged over 10 splits of data.

- Salimans, Tim, et al. "Improved techniques for training gans." *Advances in Neural Information Processing Systems*. 2016.

- Experiment:
  - Sample generation ➔ MNIST, CIFAR-10, SVHN, ImageNet

Figure 3: *(Left)* samples generated by model during semi-supervised training. Samples can be clearly distinguished from images coming from MNIST dataset. *(Right)* Samples generated with minibatch discrimination. Samples are completely indistinguishable from dataset images.

Model	Percentage of incorrectly predicted test examples for a given number of labeled samples		
	500	1000	2000
DGN [21]		36.02±0.10	
Virtual Adversarial [22]		24.63	
Auxiliary Deep Generative Model [23]		22.86	
Skip Deep Generative Model [23]		16.61±0.24	
Our model	18.44 ± 4.8	8.11 ± 1.3	6.16 ± 0.58
Ensemble of 10 of our models		5.88 ± 1.0	

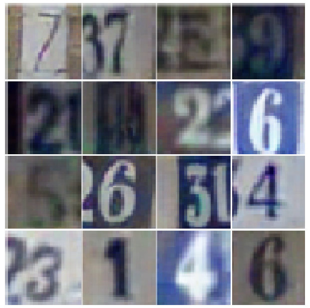


Figure 5: *(Left)* Error rate on SVHN. *(Right)* Samples from the generator for SVHN.

- Salimans, Tim, et al. "Improved techniques for training gans." *Advances in Neural Information Processing Systems*. 2016.

Presentation

Thanks for Watching