Theory of Generative Adversarial Nets

Wang, Yuanyuan

School of Statistics and Mathematics
Central University of Finance and Economics
Beijing, China

24 March 2019

Before Start

Yann LeCun's comment on GAN:

What are some recent and potentially upcoming breakthroughs in deep learning?

7 Answers

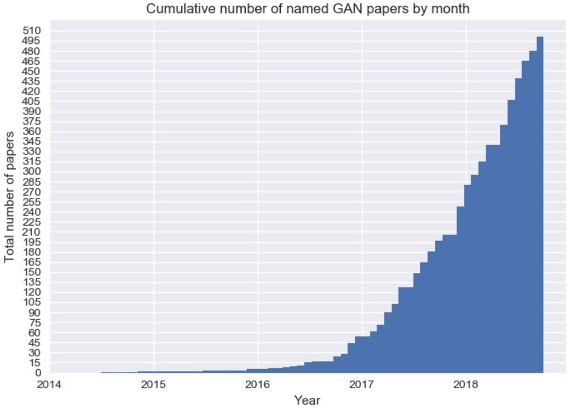
Yann LeCun, Director of AI Research at Facebook and Professor at NYU
Answered Jul 29, 2016 · Upvoted by Joaquin Quiñonero Candela, studied Machine Learning and Gokul Krishnan, M.Sc Computer Science & Machine Learning, ETH Zurich (2018)

The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This is an idea that was originally proposed by Ian Goodfellow when he was a student with Yoshua Bengio at the University of Montreal (he since moved to Google Brain and recently to OpenAI).

Before Start

Variations of GAN:

- ABC-GAN
- AC-GAN
- acGAN
- ...
- ZipNet-GAN
- αGAN
- βGAN
- ...



Before Start

Some applications

- Image generation and manipulation
- Text to image
- ...

Before Start

Other's experiments: some celebrity look images created by GAN:

Outline

1. The Basic Idea of GAN
2. How GAN Works
3. Proof of GAN's Strategy
4. Algorithm of GAN
5. Author's Experiments

Outline

Adversarial Nets Framework

GANs are deep neural net architectures comprised of two nets:

- A generative model: generate data that is as similar to real data as possible.
- A discriminative model: learns to determine whether a sample is from the model distribution or the data distribution

They are pitted against each other, this is where "adversarial" come from.

An analogous to GAN

- G: The generative model is analogous to counterfeiters
- D: the discriminative model is analogous to the police

Competition in this game drives both teams to improve their methods until the counterfeits are indistiguishable from the genuine articles.

Outline

Outline

Symbols and Notations

These are some notaions used:

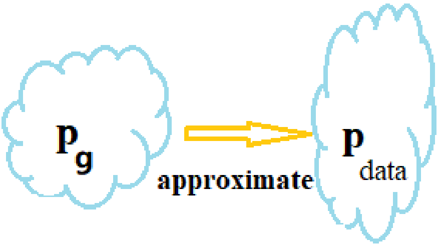
- $p_{data}(x)$: the distribution of real data
- $P_z(z)$: the prior distribution of Generator
- $G(z, \theta_g)$: a network (generator) map z to an desired output, θ_g is G's parameters
- p_g: the distribution of generator's output
- $D(x, \theta_d)$: a network (discriminator) map x to a scalar, θ_d is D's parameters

MLE's Strategy

Before GAN, MLE is commonly used:
Pre-define a distribution P_g, then sample data from P_{data}, using maximal likelihood to estimate parameters in P_g

$$L = \prod_{i=1}^{n} P_g(x_i, \theta)$$

$$\theta^* = \arg\max_{\theta} \prod_{i=1}^{m} P_g(x_i, \theta) = \arg\min_{\theta} KL(p_{data} || p_g)$$

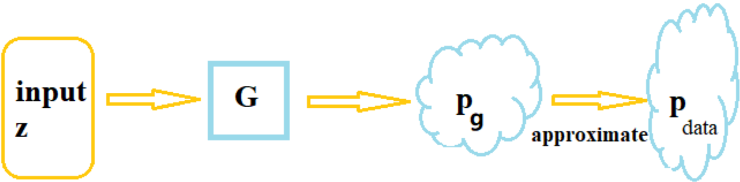


Proof of MLE's Strategy

$$
\begin{aligned}
\theta^* &= \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{m} P_g(x_i, \theta) \\
&= \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} log P_g(x_i, \theta) \\
&\approx \operatorname{argmax} E_{x \sim P_{data}}[log P_g(x, \theta)] \\
&= \underset{\theta}{\operatorname{argmax}} \int_x P_{data}(x) log(P_g(x, \theta)) dx \\
&= \underset{\theta}{\operatorname{argmax}} \int_x P_{data}(x) log(P_g(x, \theta)) dx - \int_x P_{data}(x) log(P_{data}(x)) dx \\
&= \underset{\theta}{\operatorname{argmin}} KL(p_{data} || p_g)
\end{aligned}
$$

GAN's Strategy

GAN does not pre-define a distribution of generator's output, but using discriminator to evaluate the divergence of P_g and P_{data}. Rather than optimize P_g itself, GAN optimize θ_g (params producing P_g):



$$G^* = \arg\min_G div(P_g, P_{data})$$

with P_g untractable and P_{data} unknown, how to calculate $div(P_g, P_{data})$?

GAN's Strategy

- Discriminator:
  - for real data: maximize $E_{x\sim p_{data}(x)}[logD(x)]$
  - for generated data: maximize $E_{z\sim p_z(z)}[log(1 - D(G(z)))]$
- Generator: minimize $E_{z\sim p_z(z)}[log(1 - D(G(z)))$

Combine the above criterion, we have,

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

Now objective turns to train D and G to get a minimax of V(D, G), namely,

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

Outline

Outline

The Inner Loop: For discriminator

Obejective:

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

Consider the inner loop, for G fixed, The training criterion for D is to maximize V:

$$\begin{aligned} V(G, D) &= \int_x P_{data}(x) log(D(x)) dx + \int_z P_z(z) log(1 - D(G(z))) dz \\ &= \int_x [P_{data}(x) log(D(x)) + P_g(x) log(1 - D(x))] dx \end{aligned}$$

To maximize $P_{data}(x)log(D(x)) + P_g(x)log(1 - D(x))$, we have:

$$D_G^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_g(x)}$$

The Outer Loop: For generator

Obejective:

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

With an optimal D, The obejective can be reformulated as:

$$\begin{aligned} C(G) =& E_{x \sim p_{data}(x)}[logD^*(x)] + E_{z \sim p_z(z)}[log(1 - D^*(G(z)))] \\ =& E_{x \sim p_{data}(x)}[logD^*(x)] + E_{x \sim p_g}[log(1 - D^*(x))] \\ =& E_{x \sim p_{data}(x)}[log\frac{P_{data}(x)}{P_{data}(x) + P_g(x)}] + E_{x \sim p_g}[log(\frac{P_g(x)}{P_{data}(x) + P_g(x)})] \\ =& - log4 + 2 * JSD(p_{data}||p_g) \end{aligned}$$

Now we need to minimize C(G), the minimal value is obtained when $P_g = P_{data}$

Solutions

The original objective is

$$G^* = \underset{G}{\operatorname{argmin}}\, div(P_g, P_{data})$$

Now with

$$D^* = \underset{D}{\operatorname{argmax}}\, V(D, G)$$

$V(D^*, G)$ is eqivalent to $div(P_g, P_{data})$, Then we get our new objective

$$\begin{aligned} G^* &= \underset{G}{\operatorname{argmin}}\, V(D^*, G) \\ &= \underset{G}{\operatorname{argmin}} \max_D V(D, G) \end{aligned}$$

Outline

Outline

Algorithm

for number of training iterations **do**

for k steps **do**

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

train D

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

end for

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

train G

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

end for

Minor changes

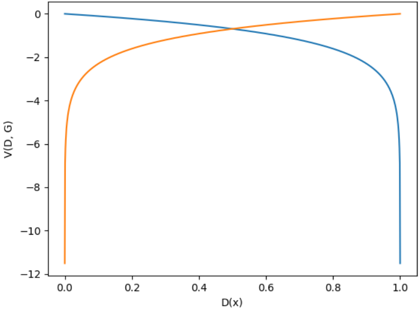
- Theoretically:

  minimize $E_{z\sim p_z(z)}[log(1 - D(G(z)))$

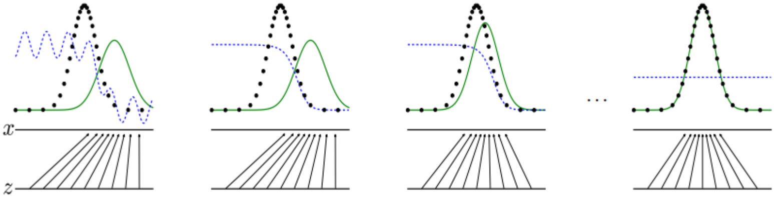
- Real implementation:

  maximize $E_{z\sim p_z(z)}[log(D(G(z)))$

- Reason:
  - Stronger gradient early in learning

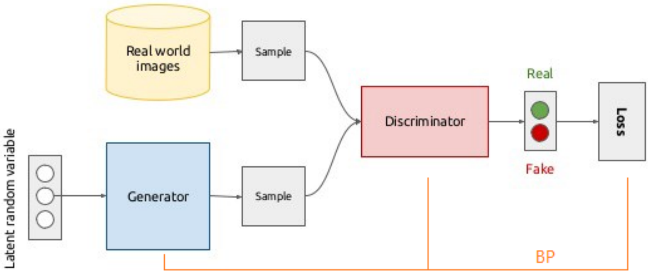


A less formal explanation



(a) (b) (c) ... (d)

Network Framework



- train D to maximize the probability of assigning the correct label to both training examples and samples from G.
- simultaneously train G to minimize this probability.

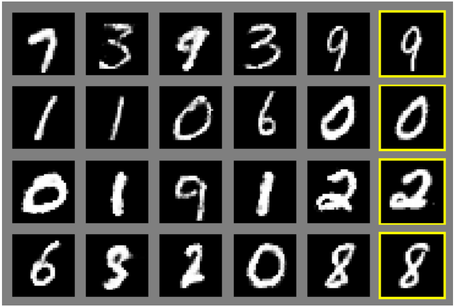
Outline

Outline

Experiments

Model	MNIST	TFD
DBN [3]	138 ± 2	1909 ± 66
Stacked CAE [3]	121 ± 1.6	$\mathbf{2110 \pm 50}$
Deep GSN [6]	214 ± 1.1	1890 ± 29
Adversarial nets	$\mathbf{225 \pm 2}$	$\mathbf{2057 \pm 26}$

Experiments



Thanks!

Thank you for your time and attention.

Questions?

References

Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]//Advances in neural information processing systems. 2014: 2672-2680.