Generative Adversarial Nets

NIPS 2014

Ian J. Goodfellow et. al

2017.7.15

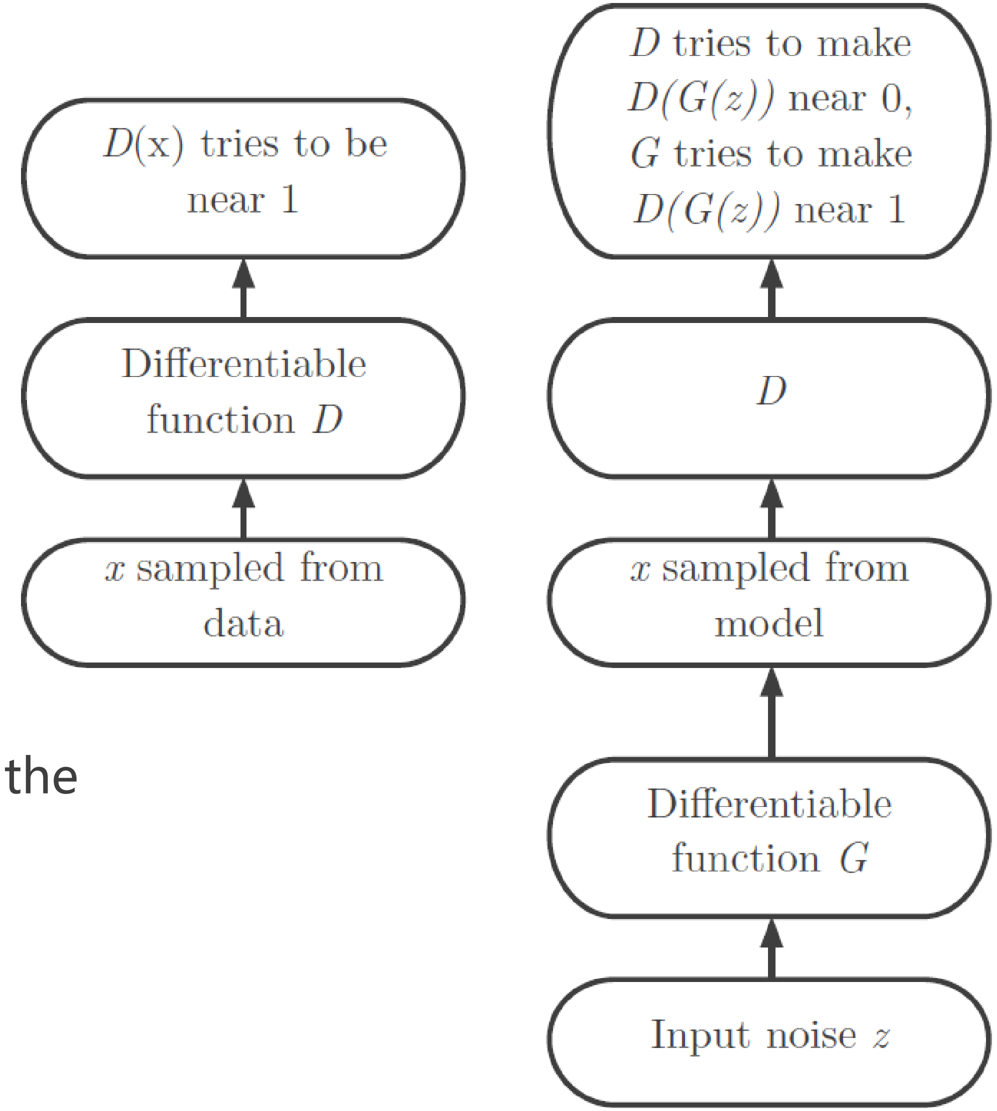
01 Adversarial nets

Input noise variables $p_z(z)$

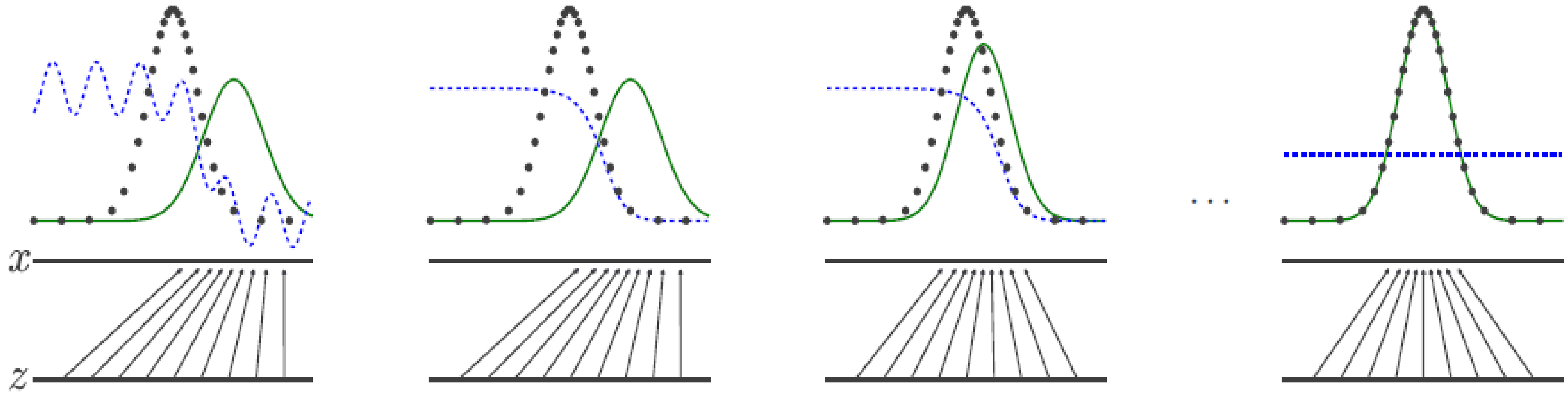
$G(z, \theta_g)$represent a mapping to data space a
G is a differentiable function represented by
 a multilayer perceptron with parameters θ_g.

Multilayer perceptron $D(\mathrm{x}, \theta_d)$ outputs a single scalar.

D(x) represents the probability that x came from the data rather than p_g.



01 Adversarial nets



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

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

02 Math

$$V(G,D) = E_{x\sim P_{data}}[\log D(x)] + E_{x\sim P_G}[\log(1-D(x))]$$

$$G^* = \arg\min_G(\max_D V(G,D))$$

$max_D V(G,D)$

$$V = E_{x\sim P_{data}}[\log D(x)] + E_{x\sim P_G}[\log(1-D(x))]$$

$$= \int P_{data}(x) \log D(x) dx + \int p_G \log\big(1-D(x)\big)\, dx$$

$$= \int [P_{data} \log D(x) + P_G(x) \log\big(1-D(x)\big)] dx$$

$$D^* \longrightarrow P_{data} \log D(x) + P_G(x) \log\big(1-D(x)\big)$$

02 Math

$$f(D) = a\log(D) + b\log(1-D)$$

$$\frac{df(D)}{dD} = \mathrm{a} \times \frac{1}{D} + b\log \times \frac{1}{1-D} \times (-1) = 0$$

$$\mathrm{a} \times \frac{1}{D^*} = b \times \frac{1}{1-D^*}$$

$$D^* = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}$$

$$\max V(G,D) = V(G,D^*)$$

$$= E_{x\sim P_{data}}\left[log\frac{P_{data}(x)}{P_{data}(x) + P_G(x)}\right] + E_{x\sim P_G}[log\frac{P_G(x)}{P_{data}(x) + P_G(x)}]$$

$$= \int P_{data}(x)log\frac{\frac{1}{2}P_{data}}{\frac{P_{data}(x) + P_G(x)}{2}}dx + \int P_G(x)log\frac{\frac{1}{2}P_G(x)}{\frac{P_{data} + P_G(x)}{2}}dx$$

02 Math

$$= -2log2 + KL(P_{data}\left|\right|\frac{P_{data}(x) + P_G(x)}{2}) + KL(P_G(x)||\frac{P_{data} + P_G(x)}{2})$$

$$= -2log2 + 2 \cdot JSD(p_{data}||p_g)$$

03 Framework

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 $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ 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:

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

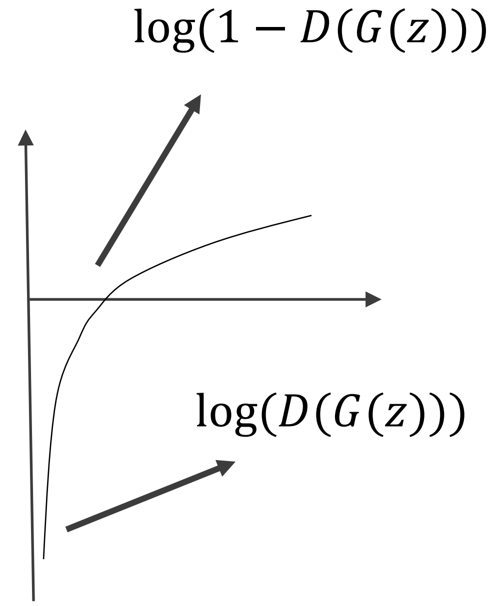
  end for

- 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:

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

end for

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



04 Cross-entropy cost function

$$C = \frac{1}{2n}\sum_x ||y(x) - a(x)||^2$$

$$C = \frac{1}{2}(y - a)^2$$

$$\frac{\partial C}{\partial w} = (a - y)\sigma(z)'x$$

$$\frac{\partial C}{\partial b} = (a - y)\sigma(z)'$$

$$C = -\frac{1}{n}\sum_x [ylna + (1 - y)\ln(1 - a)]$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\frac{\partial C}{\partial w_j} = -\frac{1}{n}\sum_x [\frac{y}{\sigma(z)} - \frac{1 - y}{1 - \sigma(z)}]\frac{\partial \sigma(z)}{\partial w_j}$$

$$= -\frac{1}{n}\sum_x [\frac{y}{\sigma(z)} - \frac{1 - y}{1 - \sigma(z)}]\sigma(z)'x_j$$

$$\sigma(z)' = \sigma(z)(1 - \sigma(z))$$

$$= -\frac{1}{n}\sum_x \left[\frac{\sigma(z)'x_j}{\sigma(z)\big(1 - \sigma(z)\big)}(\sigma(z) - y)\right] = \frac{1}{n}\sum_x x_j(\sigma(z) - y)$$

$$\frac{\partial C}{\partial b} = \frac{1}{n}\sum_x (\sigma(z) - y)$$

04 Cross-entropy cost function

$$\frac{\partial C}{\partial b} = \frac{\partial C}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial b} \qquad = \frac{\partial C}{\partial a}\sigma(z)'\frac{\partial(wx+b)}{\partial b} = \frac{\partial C}{\partial a}\sigma(z)' \qquad = \frac{\partial C}{\partial a}a(1-a)$$

$$\frac{\partial C}{\partial b} = (\mathrm{a}-\mathrm{y})\sigma(z)' \longrightarrow \frac{\partial C}{\partial b} = (\mathrm{a}-\mathrm{y})$$

$$\frac{\partial C}{\partial a}a(1-a) = a - y$$

$$C = -[ylna + (1-y)\ln(1-a)] + constant$$

05 Extra

Training processing

$$\theta^D \longrightarrow J^D \qquad\qquad \theta^G \longrightarrow J^G$$

The discriminator's cost

$$J^D(\theta^D, \theta^G) = -\frac{1}{2} E_{x\sim p_{data}} \log\big(D(x)\big) - \frac{1}{2} E_z log(1 - D(G(z)))$$

The simplest version of the game is a zero-sum game, in which the sum of all player's costs is always zero

$$J^G = -J^D$$

05 Extra

Because J^G is tied directly to J^D, we can summarize the entire game with a value function specifying the discriminator's payoff

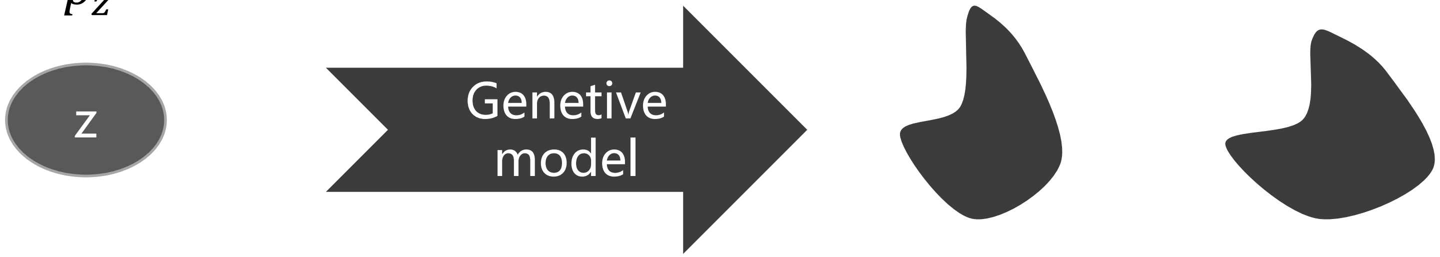
$$V(\theta^D, \theta^G) = -J^D(\theta^D, \theta^G)$$

Zero-sum games are also called minimax games because their solution involves minimization in an outer loop and maximization in an inner loop:

$$\theta^{G*} = \underset{\theta^G}{\operatorname{argmin}} \max_{\theta^D} V(\theta^D, \theta^G)$$

06 MLE

p_z $\quad G(z) = x$ $\quad P_G(x;\theta)$ $\quad P_{data}(x)$



$$D_{KL}(P||Q) = \int_{-\infty}^{+\infty} p(x)log\frac{p(x)}{q(x)}$$

$x^1, x^2, \dots, x^m$

$$L = \prod_{i=1}^{m} P_G(x^i;\theta)$$

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

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{m} p_G(x^i,\theta) \Leftrightarrow \underset{\theta}{\operatorname{argmax}}\, log \prod_{i=1}^{m} p_G(x^i,\theta)$$

06 MLE

$$= \underset{\theta}{\operatorname{argmax}} \sum_{i}^{m} logP_G(x^i, \theta) \approx \underset{\theta}{\operatorname{argmax}} E_{x \sim P_{data}}[logP_G(x;\theta)]$$

$$\Leftrightarrow \underset{\theta}{\operatorname{argmax}} \int P_{data}(x) logP_G(x;\theta)dx - \int P_{data}(x) logP_{data}(x)dx$$

$$= \underset{\theta}{\operatorname{argmax}} \int P_{data}(x) log \frac{P_G(x;\theta)}{P_{data}(x)} dx$$

$$\underset{\theta}{\operatorname{argmin}} \, KL(P_{data}||P_G(x;\theta))$$

07 Advantages and disadvantages

There is no explicit representation of p_g

D must be synchronized well with G during training

Markov chains are never needed, only backprop is used to obtain gradients

No inference is needed during learning

A wide variety of functions can be incorporated into the model