Generative Adversarial Networks

: From Theory to Practice

Sunmook Choi

Dept. of Mathematics
Korea University

Generative Adversarial Networks

Given a dataset, we want to generate new data which seem to be obtained from the dataset, i.e., given p_{data}, we want to learn the generator's distribution p_g over data $\boldsymbol{x}$ s.t. $p_{\text{data}} = p_g$.

How?

- Define a prior on input noise/latent variable $p_z(\boldsymbol{z})$, and
- represent a mapping to a data space as $G(\boldsymbol{z}; \boldsymbol{\theta}_g)$, which is the generator.
- Define the discriminator $D(\boldsymbol{x}; \theta_d)$ that outputs the probability that $\boldsymbol{x}$ came from the data distribution p_{data} rather than p_g.
- Train D to maximize the probability of assigning the correct label to both training examples (real, label=1) and samples from G (fake, label=0).
- Simultaneously train G to minimize $\log(1 - D(G(z)))$.

Reference: Goodfellow, I. et al., 2014. Generative adversarial nets. In Advances in neural information processing systems. pp. 2672–2680. `https://arxiv.org/abs/1406.2661`

Objective Function

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

Recall: Binary Cross Entropy Loss for Logistic Regression

For binary dataset $\left\{\boldsymbol{x}^n, y^n\right\}_{n=1}^N$ where $y^n \in \{0,1\}$ with hypothesis model $h_{\boldsymbol{\theta}}$,

$$BCE(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}\Big[-y^n \log\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}^n)\right) - (1-y^n)\log\left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^n)\right)\Big]$$

- For D, $\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})$ should have label 1, and $G(\boldsymbol{z}) \sim p_g(\boldsymbol{x})$ should have label 0. Hence, maximizing $V(D,G)$ w.r.t. D is equivalent to minimizing BCE.

Objective Function

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

Recall: Binary Cross Entropy Loss for Logistic Regression

For binary dataset $\left\{\boldsymbol{x}^n, y^n\right\}_{n=1}^N$ where $y^n \in \{0,1\}$ with hypothesis model $h_{\boldsymbol{\theta}}$,

$$BCE(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}\left[-y^n \log\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}^n)\right) - (1-y^n)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}^n)\right)\right]$$

- Practically, minimizing $V(D,G)$ w.r.t. G may not provide sufficient gradient for G early in learning. Instead, we train G to maximize $\log D(G(\boldsymbol{z}))$, i.e., minimize $-\log D(G(\boldsymbol{z}))$.
- Hence, minimizing $-\log D(G(\boldsymbol{z}))$ w.r.t. G is equivalent to minimizing BCE.

Training Algorithm

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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ 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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ 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.

Theoretical Results

Proposition 1. For G fixed, the optimal discriminator D is $D^*_G(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x})+p_g(\boldsymbol{x})}$.

$\because V(D,G) = \int_x p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))\, dx + \int_z p_z(z) \log(1 - D(G(z)))\, dz = \int_x p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))\, dx$

For any $(a, b) \in \mathbb{R}^2 \setminus (0, 0)$, the function $y \mapsto a \log(y) + b \log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$.

Using the result above, the objective function w.r.t. G can be reformulated as follows:

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

Theoretical Results

Theorem 1. The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{\text{data}}$.

$\because$ Recall that $D_{KL}(p \,\|\, q) = \mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right]$ and $D_{JS}(p, q) = \frac{1}{2} D_{KL}\left(p \,\|\, \frac{p+q}{2}\right) + \frac{1}{2} D_{KL}\left(q \,\|\, \frac{p+q}{2}\right)$.

$$
\begin{aligned}
C(G) &= \mathbb{E}_{x\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] \\
&= D_{KL}\left(p_{\text{data}} \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right) + D_{KL}\left(p_g \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right) - 2\log 2 \\
&= D_{JS}\left(p_{\text{data}}, p_g\right) - \log 4
\end{aligned}
$$

Since the Jensen-Shannon divergence is a metric, we have shown that the global minimum of $C(G)$ is $-\log 4$ and that the only solution is $p_g = p_{\text{data}}$, i.e., the generative model perfectly replicates the data generating process.