GAN

A/Prof Richard Yi Da Xu
Yida.Xu@uts.edu.au
Wechat: aubedata
https://github.com/roboticcam/machine-learning-notes

University of Technology Sydney (UTS)

April 22, 2019

- $$\begin{aligned}\min_G \max_D L(D, G) &= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \\ &= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{x \sim p_g(x)}[\log(1 - D(x)]\end{aligned}$$

- KL divergence measures how one probability distribution p diverges from a second expected probability distribution q

$$D_{KL}(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)} \mathrm{d}x$$

DKL achieves the minimum zero when p(x) == q(x) everywhere.
It is noticeable according to the formula that KL divergence is asymmetric. In cases where p(x) is close to zero, but q(x) is significantly non-zero, the qs effect is disregarded. It could cause buggy results when we just want to measure the similarity between two equally important distributions.

$$D_{JS}(p\|q) = \frac{1}{2} D_{KL}(p\|\frac{p+q}{2}) + \frac{1}{2} D_{KL}(q\|\frac{p+q}{2})$$

Optimal value D^*:

- $$\min_G \max_D L(D,G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$
  $$= \mathbb{E}_{x\sim p_r(x)}\underbrace{[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1-D(x)]}_{L(G,D)}$$
  $$\implies L(G,D) = \int_x \Big(\underbrace{p_r(x)\log(D(x)) + p_g(x)\log(1-D(x))}_{f(D(x))}\Big)dx$$
  $$f(D(x)) = p_r(x)\log D(x) + p_g(x)\log(1-D(x))$$
  $$\frac{df(D(x))}{dD(x)} = p_r(x)\frac{1}{D(x)} - p_g(x)\frac{1}{1-D(x)} = \left(\frac{p_r(x)}{D(x)} - \frac{p_g(x)}{1-D(x)}\right)$$
  $$= \frac{p_r(x) - (p_r(x)+p_g(x))D(x)}{D(x)(1-D(x))}$$
- Let $\frac{df(D(x))}{dD(x)} = 0$:
  $$\frac{p_r(x) - (p_r(x)+p_g(x))D(x)}{D(x)(1-D(x))} = 0$$
  $$\implies p_r(x) - (p_r(x)+p_g(x))D(x) = 0$$
  $$D^*(x) \quad p_r(x)$$

global optimal

- knowing $D^*(x) = \frac{p_r(x)}{p_r(x)+p_g(x)}$, then optimal $p_g(x)$ is when it becomes identifical to $p_r(x)$

$$p_r(x) = p_g(x) \implies D^*(x) = \frac{1}{2}$$

$$\begin{aligned}
L(G, D^*) &= \int_x \Big(p_r(x) \log(D^*(x)) + p_g(x) \log(1 - D^*(x)) \Big) dx \\
&= \int_x \left(p_r(x) \log\left(\frac{1}{2}\right) + p_g(x) \log\left(1 - \frac{1}{2}\right) \right) dx \\
&= \log \frac{1}{2} \int_x p_r(x) dx + \log \frac{1}{2} \int_x p_g(x) dx \\
&= -2 \log 2
\end{aligned}$$

Linear Programming

- looking at:

$$\sum_y \gamma(x,y) = P_\theta(x)$$

$$\underbrace{\left[\begin{array}{c|c|c|c|c}
\mathbf{1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{1} & \cdots & \mathbf{0} & \mathbf{0} \\
\vdots & \vdots & \cdots & \vdots & \vdots \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{1} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{1} \\
\hline
1\ 0\ 0\ \ldots & 0\ 1\ 0 \ldots & \ldots & 0\ \ldots 1\ 0 & 0\ \ldots 0\ 1 \\
1\ 0\ 0\ \ldots & 0\ 1\ 0 \ldots & \ldots & 0\ \ldots 1\ 0 & 0\ \ldots 0\ 1 \\
\vdots & \vdots & \cdots & \vdots & \vdots \\
1\ 0\ 0\ \ldots & 0\ 1\ 0 \ldots & \ldots & 0\ \ldots 1\ 0 & 0\ \ldots 0\ 1 \\
1\ 0\ 0\ \ldots & 0\ 1\ 0 \ldots & \ldots & 0\ \ldots 1\ 0 & 0\ \ldots 0\ 1
\end{array}\right]}_{\mathbf{A}}
\underbrace{\begin{bmatrix}
\gamma(x_1, \mathbf{y}) \\
\gamma(x_2, \mathbf{y}) \\
\vdots \\
\gamma(x_{n-1}, \mathbf{y}) \\
\gamma(x_n, \mathbf{y})
\end{bmatrix}}_{\mathbf{x}}
=
\underbrace{\begin{bmatrix}
P_r(x_1) \\
P_r(x_2) \\
\vdots \\
P_r(x_{n-1}) \\
P_\theta(y_1) \\
P_\theta(y_2) \\
\ldots \\
P_\theta(y_n)
\end{bmatrix}}_{\mathbf{b}}$$

Consider some function . Let (the least of all maxima) and (the greatest of all minima). The argument that is simple: Any is automatically allowed as a candidate for the infimum of , but not the other way around.

For , at least one of these statements must be true:

for some . This is only possible if is not convex in , because is already an infimum for .

for some . This is only possible if is not concave in , because is already a supremum for . This means of course that, if is convex and is concave, then the minimax principle applies and . In our case, we can above already see from the underbrace that the convexity condition is met. Lets try changing to :

We see that the infimum is concave, as required. Because all functions that are Lipschitz continuous produce the same optimal solution for , and only they are feasible solutions of , we can turn this condition into a constraint. With that, we have the dual form of the Wasserstein distance:

This is our case of the Kantorovich-Rubinstein duality. It actually holds for other metrics than just the Euclidian metric we used. But the function is suitable to be approximated by a neural network, and this version has the advantage that the Lipschitz continuity can simply be achieved by clamping the weights.