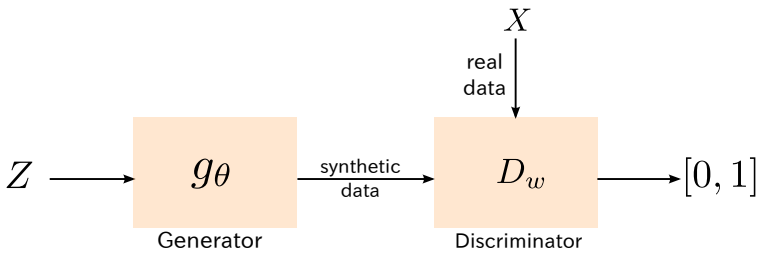
ECE598: Representation Learning: Algorithms and Models	Fall 2017
Lecture 12: Generative Adversarial Networks (GANs)	
Lecturer: Prof. Pramod Viswanath	*Scribe: Amir Taghvaei, Oct 26, 2016*

12.1 Generative Adversarial Network

Generative Adversarial Network (GAN) is an example of generative models. A generative model, is a model that is able to generate new samples from a hidden probability distribution. A classical and relatively simple example of a generative model is *mixture of Gaussians*. This model contains a parametric form of the hidden probability distribution, where the parameters are learned from a given set of samples. The learned distribution is then used to generate new samples. GAN is different from these probabilistic approaches as it does not contain the explicit form of the hidden distribution. Instead it directly generates new samples. GAN behaves as a black-box that takes a Gaussian random variable as input, and outputs new sample from the hidden distribution. The black box is often represented by a Neural Network. In this sense, GANs and Variational Auto Encoders are similar. Their main difference is in how they are trained. In fact, training is the most interesting feature of GAN.

Mathematical formulation: Let $X \in \mathcal{X}$ be the random variable of interest, and let P_X be its probability distribution. The objective of a generative model is to generates new samples from P_X, given training data $\{X_1, \dots, X_N\} \overset{\text{i.i.d}}{\sim} P_X$.

GAN has the following architecture:



The two main components of GAN are:

- **Generator:** The generator is a neural network represented by a map $g_\theta : \mathcal{Z} \to \mathcal{X}$ where the parameter θ represents the weights of the neural network. The map takes as input, a Gaussian random variable $Z \sim N(0, I)$ (usually of dimensions lower than X) and outputs $g_\theta(Z)$. The distribution of $g_\theta(Z)$ is denoted by P_θ. The goal of the generator is to choose θ such that the output $g_\theta(Z)$ has a distribution close to X.
- **Discriminator:** The discriminator is also a neural network, represented by a map $D_w : \mathcal{X} \to [0, 1]$ parametrized by weights w. The goal of the discriminator is to assign 1 to the samples from the real distribution P_X and 0 to the generated samples from P_θ.

The parameters of the GAN are (θ, w). They are obtained by solving the following min-max problem:

$$\min_{\theta} \max_{w} \ \mathbb{E}[\log(D_w(X)) + \log(1 - D_w(g_\theta(Z)))] \tag{12.1}$$

There are various (game theoretic) ways to motivate such a min-max formulation. However, I find the following motivation the most satisfactory (and the only theoretical justification): If the maximization over w is replaced with maximization over all functions D, then the min-max problem reduces to

$$\min_{\theta} \ JS(P_X \| P_\theta) \tag{12.2}$$

where $JS(P_X \| P_\theta)$ is the Jensen-Shannon divergence between P_X and P_θ. The JS divergence for any two probability distributions p and q is defined according to

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

where KL is the Kullback-Leiber divergence. The JS divergence provides a measure of distance between two probability distributions. Therefore the minimization over θ means, choosing the P_θ that is closest to the target distribution P_X in the JS divergence distance.

Training: Training is simply simultinous SGD (and its variants) on the minimization and max imization problem, where w moves in the positive gradient direction and θ moves in the negative gradient direction. In each iteration, one takes batch of samples from the target distribution $\{X_1, \ldots, X_N\} \sim P_X$ and synthetic samples from the generator $\{g_\theta(Z_1), \ldots, g_\theta(Z_N)\}$ and updates the parameters according to

$$w \to w + \eta \sum_{i=1}^{N} \nabla_w \left[\log(D_w(X_i)) + \log(1 - D_w(g_\theta(Z_i)))\right]$$
$$\theta \to \theta - \eta \sum_{i=1}^{N} \nabla_\theta \left[\log(1 - D_w(g_\theta(Z_i)))\right]$$

where η is the learning rate. It is not necessary to use the same learning rate, or the same number of samples for θ and w. These are tuning parameters that are found empirically.

Mode collapse: Mode collapse refers to the situation where the generator maps several inputs to the same output. For example the generator makes multiple images that contain the same color or texture. A cartoon illustration of mode collapse is presented in Figure 12.1. It is observed that the generator can not capture all modes of the target distribution, and it maps all data inputs to a single mode. One reason for mode collapse is that the simultaneous SGD does not differentiate between solving the min-max or max-min problem. If the minimization is performed first, i.e $\min_\theta \mathbb{E}[\log(1 - D_w(g_\theta(Z)))]$, then the optimal generator would map all data points to where the discriminator assigns high values (usually a single mode of the target distribution).

In order to circumvent this issue, one may want to perform the maximization to optimality before updating the generator. However, it is observed that the optimal discriminator has vanishing gradients on the support of the target and generator distributions (Thm. 2.2 in [AB17]). Vanishing gradients does not allow for the generator to be trained efficiently.

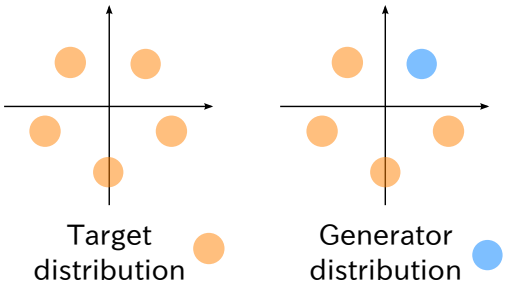


Figure 12.1: Cartoon illustration of the mode collapse phenomenon in GAN

12.2 Wasserstein GAN

Wasserstein GAN are introduced to address the mode collapse and other issues in the original formulation of GAN. The main point of WGAN is to replace the JS distance to the L^1-Wasserstein distance metric. The intuition behind this choice is that (i) Wasserstein distance respects the geometry of the underlying space, and (ii) it captures the distance of two probability measures when their support do not intersect whereas KL divergence type distances can not capture that. Not intersecting support is common in high dimensional applications where the target distribution lies in a low dimensional manifold.

Topological properties of the Wasserstein distance: Consider the following distances and divergences between two probability distributions P_X and P_Y:

- The *Total Variation* (TV) distance

$$\delta(P_X, P_Y) := \sup_A |P_X(A) - P_Y(A)|$$

- The *Kullback-Leiber* (KL) divergence

$$KL(P_X \| P_Y) := \int \log(\frac{p_X(x)}{p_Y(x)}) p_X(x) \mathrm{d}\mu(x)$$

  where p_X and p_Y are densities of P_X and P_Y with respect to the measure μ.

- The *Jensen-Shannon* (JS) divergence

$$JS(P_X \| P_Y) := KL(P_X \| P_m) + KL(P_Y \| P_m)$$

  where $P_M = \frac{P_X + P_Y}{2}$.

- The L^1-Wasserstein distance

$$W_1(P_X, P_Y) = \inf_\pi \mathbb{E}[|X - Y|]$$

  where π is any coupling between pair of random variables (X, Y) such that $X \sim P_X$ and $Y \sim P_Y$.

Note that the TV distance is the same as the L_0-Wasserstein distance due to Strassen's Theorem

$$\sup_A |P_X(A) - P_Y(A)| = \inf_\pi \mathbb{E}[1_{X \neq Y}] =: W_0(P_X, P_Y)$$

The topology induced by the distances can be compared by asking the following question: if a sequence converges in a distance, does it converge in the other distance or not? For example it can be shown that if $\{P_n\}$ converges in KL distance, then $\{P_n\}$ converges in JS distance. Therefore KL induces a stronger topology compared to JS (it is easier for sequences to converge in KL). Here is a complete comparison of the topologies induced by the distances (Thm. 2 in [ACB17]):

$$KL \quad \Rightarrow \quad (TV \Leftrightarrow JS) \quad \Rightarrow \quad (W_1 \Leftrightarrow \text{convergenc in dist.})$$

The following example illustrates the difference between these four distances in terms of continuity:

Example 12.1. Let $Z \sim \text{Unif}[0,1]$ be the uniform distribution on the unit interval. Let P_0 be the probability distribution of $(0, Z) \in \mathbb{R}^2$. And let P_θ be the family of probability distributions parametrized with θ corresponding to $(\theta, Z) \in \mathbb{R}^2$. Then

$$\delta(P_0, P_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$$

$$KL(P_0, P_\theta) = \begin{cases} \infty & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$$

$$JS(P_0, P_\theta) = \begin{cases} 2\log(2) & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$$

$$W_1(P_0, P_\theta) = |\theta|$$

The example implies that the Wasserstein distance is the only distance among the four distances which is continuous with respect to θ. In fact one can show that the continuity of the Wasserstein distance with respect to the parameter θ is true in general (Thm. 1 in [ACB17]).

$$\text{If } g_\theta \text{ is continuous w.r.t } \theta, \text{ then } W_1(P_X, P_{g_\theta(Z)}) \text{ is continuous w.r.t } \theta$$

Min-max formulation with Wasserstein distance: Motivated by nice topological properties of Wasserstein distance, we replace the JS distance with the Wasserstein distance in (12.2).

$$\min_\theta W_1(P_X, P_\theta)$$

Then we use the Kantorovich-Rubinstein duality [Vil03, Thm. 1.14] to compute the Wasserstein distance

$$W_1(P_X, P_\theta) := \sup_{\|D\|_L \leq 1} \mathbb{E}[D(X) - D(g_\theta(Z))]$$

where the sup is over all functions with Lipschitz constant less than 1. Finally the sup over all functions is replaced with parametrized family of functions D_w:

$$\min_\theta \max_{w \in \mathcal{W}} \mathbb{E}[D(X) - D(g_\theta(Z))], \quad \text{s.t} \quad \|D_w\|_L \leq K$$

where $\mathcal{W}$ is chosen such that the Lipschitz constant of D_w be smaller than a constant K. In practice this is enforced by constraining the infinity norm of the weights (known as clipping).

No mode collapse: The Wasserstein formulation is empirically shown to avoid mode collapse. The reason is that, in this formulation one can maximize the discriminator to optimality, before updating the generator. In contrast to the JS formulation, the optimal discriminator does not introduce vanishing gradients. This is illustrated in Figure 12.2 (The toy example illustrating this concept is Fig. 2 in [ACB17]).

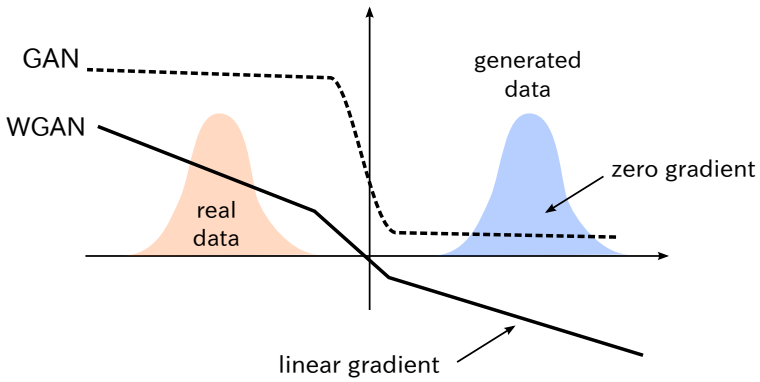


Figure 12.2: Cartoon illustration of how vanishing gradients is avoided in WGAN. The reason is that the Lischitz constant of the discriminator is constrained to be less than one in WGAN.

Open question: The main open question regarding both formulations is the convergence. The existing result regarding the convergence is that if the first maximization is performed optimally in the function space corresponding to D, then the gradient descent on the space of the probability distributions for P_θ would converge (since the distance is convex function of the probability distribution) (Prop. 2 in [GPAM+14]).

Another open question, is theoretical understanding of mode collapse even in the simple and ideal cases.

12.3 Other stuff

There are many existing empirical results regarding GANs. A good source is [Goo16] which also includes all practical tricks one may use (e.g using the labels). Also [ACB17] contains the empirical results for WGAN with the conclusion that WGAN are more robust with neural network architecture, the Wasserstein metric is a better measure for quality of the generated images, and mode collapse does not happen.

12.4 Appendix

Proof of equation (12.2): Express the expectation in integral form

$$\mathbb{E}[\log(D(X)) + \log(1 - D(\hat{X}_\theta))] = \int \left[\log(D(x))p_X(x) + \log(1 - D(x))p_{\hat{X}_\theta}(x)\right] \mathrm{d}x$$

Therefore the maximization over D reduces to the maximization over $D(x)$ for each fixed x. For each fixed x, the integrand is (up to a constant) the KL divergence of two Bernouli random variables with parameter $\frac{p_X(x)}{P_X(x)+P_{\hat{X}_\theta}(x)}$ and $D(x)$. Therefore the optimal $D(x)$ is equal to $\frac{p_X(x)}{P_X(x)+P_{\hat{X}_\theta}(x)}$. And the resulting expectation is

$$\begin{aligned}
&\int \left[\log(\frac{p_X(x)}{P_X(x)+P_{\hat{X}_\theta}(x)}) p_X(x) + \log(\frac{P_{\hat{X}_\theta}(x)}{P_X(x)+P_{\hat{X}_\theta}(x)}) p_{\hat{X}_\theta}(x) \right] \mathrm{d}x \\
&\quad = KL\left(P_X \| \frac{P_X + P_{\hat{X}_\theta}}{2}\right) - \log(2) + KL\left(P_{\hat{X}_\theta} \| \frac{P_X + P_{\hat{X}_\theta}}{2}\right) - \log(2) \\
&\quad = JS(P_X \| P_{\hat{X}_\theta}) - 2\log(2)
\end{aligned}$$

Bibliography

[AB17] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[Goo16] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[Vil03] Cédric Villani. *Topics in optimal transportation*. Number 58. American Mathematical Soc., 2003.