DS 801: Advanced Optimization for Data Science **KAIST, Fall 2024**
Lecture #18: Generative Adversarial Networks May 8, 2024
Lecturer: Dabeen Lee

1 Outline

In this lecture, we cover

- introduction to Generative Adversarial Networks
- training GANS,
- f-GANs.

2 Introduction to Generative Adversarial Networks

Generative modelling is an area of machine learning where the goal is to model the underlying distribution of a given data set, such as images and speech data. One may use a generative model for the following scenarios. Given a data set of objects, we want to artificially generate similar objects. For example, by learning from images of cats, we want to generate realistic image samples of cats. For language modelling, we want to model the likelihood of some candidate sentences in some context.

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a given data set. For a grayscale image, we have $d = 10^6$. For the MNIST data set of handwritten digits, we have $d = 28 \times 28 = 784$. Our assumption is that the training data set $\mathcal{X}$ is generated by a probability distribution μ with density function $p(x)$. In other words, $\mathcal{X}$ connsists of samples drawn from μ. Then a generative model's goal is to learn and obtain a good approximation of μ based on $\mathcal{X}$.

In this lecture, we study **Generative Adversarial Networks (GANs)** [GPAM+14] that are a generative modelling framework. In fact, it is an **implicit generative model** in the sense that a GAN is trained to generate samples, not explicitly representing a probability distribution. To be more specific, the **generator network** G receives a **code vector** $z \in \mathbb{R}^k$ and produces a sample

$$x = G(z).$$

Here, $G : \mathbb{R}^k \to \mathbb{R}^d$ is a determistic mapping while z is drawn from a fixed distribution γ such as the standard Guassian distribution $N(0, I_k)$. Note that the generator network corresponds to the distribution ν with the density function

$$q(x) = \gamma\left(G^{-1}(x)\right).$$

Hence, the generator network produces a sample from the distribution ν with density function q without explicitly modeling q.

As GANs do not directly model a probability distribution, we cannot use maximum likelihood to train them. Instead, GANs use a novel training framework based on the **discriminator network** D. Here, the discriminator D takes a sample x and returns

$$D(x) \in [0, 1]$$

that measures the probability of x coming from the true data. Basically, when the generator G produces a sample $x = G(z)$, the discriminator classifies whether x comes from the training data set or were given by G. Then the objective of the generator is to output samples that are indistinguishable from the true data.

Training a GAN is done with a loss function given by

$$V(G, D) = \mathbb{E}_{x\sim\mu}\left[\log D(x)\right] + \mathbb{E}_{z\sim\gamma}\left[\log(1 - D(G(z)))\right].$$

Note that given a fixed generator G, maximizing $V(G, D)$ with respect to D requires making $D(x)$ high for x coming from the true data while making $D(G(z))$ small for $z \sim \gamma$. Hence, by considering

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

one can train the discriminator so that it assigns a high value to x from the true data and a small value to $G(z)$ returned by the generator. Meanwhile, the generator's goal is to fool the discriminator by making it end up assigning a high value to $G(z)$. One can achieve this goal by considering

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

This is a minimax optimization problem.

Recall that the generator G represents a distribution with density function $q(x) = \gamma(G^{-1}(x))$ and ν denotes the distribution. Then

$$\begin{aligned} V(G, D) &= \mathbb{E}_{x\sim\mu}\left[\log D(x)\right] + \mathbb{E}_{z\sim\gamma}\left[\log(1 - D(G(z)))\right] \\ &= \mathbb{E}_{x\sim\mu}\left[\log D(x)\right] + \mathbb{E}_{x\sim\nu}\left[\log(1 - D(x))\right] \\ &= \int_{\mathbb{R}^d} \log D(x) d\mu(x) + \int_{\mathbb{R}^d} \log(1 - D(x)) d\nu(x) \\ &= \int_{\mathbb{R}^d} \left(p(x)\log D(x) + q(x)\log(1 - D(x))\right) dx. \end{aligned}$$

Proposition 18.1. *For a fixed generator G that corresponds to density q, the optimal discriminator is attained by*

$$D^*(x) = \frac{p(x)}{p(x) + q(x)}$$

for $x \in \operatorname{supp}(\mu) \cup \operatorname{supp}(\nu)$.

Proof. For $x \in \operatorname{supp}(\mu) \cup \operatorname{supp}(\nu)$, we have $p(x), q(x) > 0$. Note that

$$h(y) := p(x)\log y + q(x)\log\log(1 - y)$$

is maximized at $y = p(x)/(p(x) + q(x))$. Hence, setting $D(x) = p(x)/(p(x) + q(x))$ for $x \in \operatorname{supp}(\mu) \cup \operatorname{supp}(\nu)$ would maximize $V(G, D)$. □

By Proposition 18.1, we have

$$\max_D V(G, D) = \int_{R^d} \left(p(x)\log\frac{p(x)}{p(x) + q(x)} + q(x)\log\frac{q(x)}{p(x) + q(x)}\right) dx.$$

The Kullback-Leibler(KL) divergence of two distributions p and q is defined as

$$D_{KL}(p||q) = \int_{\mathbb{R}^d} p(x)\log\left(\frac{p(x)}{q(x)}\right) dx.$$

Moreover, the Jensen-Shannon divergence of p and q is defined as

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

Theorem 18.2. *The optimal generator G that solves*

$$\min_G \left(\max_D \quad V(G, D) \right)$$

is given by setting

$$q(x) = p(x) \quad \text{and} \quad D(x) = \frac{1}{2}$$

for all $x \in \operatorname{supp}(\mu)$.

Proof. Note that when $q(x) = p(x)$, the optimal discriminator is given by $D(x) = 1/2$. In this case, we have

$$V(G, D) = \log \frac{1}{2} \int_{\mathbb{R}^d} (p(x) + q(x))\, dx = -\log 4.$$

In general,

$$\begin{aligned}
&\max_D V(G, D) \\
&= \int_{\mathbb{R}^d} \left(p(x) \log \frac{p(x)}{p(x)+q(x)} + q(x) \log \frac{q(x)}{p(x)+q(x)} \right) dx \\
&= \int_{\mathbb{R}^d} \left(p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \right) dx - \log \frac{1}{2} \int_{\mathbb{R}^d} (p(x) + q(x))\, dx \\
&= D_{KL}\left(p \middle|\middle| \frac{p+q}{2}\right) + D_{KL}\left(q \middle|\middle| \frac{p+q}{2}\right) - \log 4 \\
&= 2D_{JS}(p||q) - \log 4.
\end{aligned}$$

It is known that the Jensen-Shannon divergence is always nonnegative and has value zero only if $p = q$. ☐

Hence, this theorem implies that the optimal generator corresponds to the true distribution.

3 Training GANs

In the previous section, we saw that the optimal solution to the minimax formulation of a GAN corresponds to the true distribution. Then the next question is how we train a GAN to find an optimal generator. The next theorem suggests an iterative procedure to train a GAN.

Theorem 18.3. *Suppose that at each iteration, the discriminator D is allowed to reach its optimum D^* given q. Then an algorithm that updates q to minimize*

$$\int_{\mathbb{R}^d} (p(x) \log D^*(x) + q(x) \log(1 - D^*(x)))\, dx$$

guarantees that q converges to p.

In practice, we use neural networks for the generator and discriminator networks. Suppose that

$$G = G_\theta \quad \text{and} \quad D = D_\omega$$

where G_θ and D_ω are neural networks parametrized by θ and ω, respectively. Then the loss function is given by

$$V(\theta, \omega) = \mathbb{E}_{x\sim\mu}\left[\log D_\omega(x)\right] + \mathbb{E}_{z\sim\gamma}\left[\log(1 - D_\omega(G_\theta(z)))\right].$$

To solve the minimax optimization of $V(\theta, \omega)$, we obtain unbiased estimators of $\nabla_\theta V(\theta, \omega)$ and $\nabla_\omega(\theta, \omega)$ based on minibatch data samples. Algorithm 1 provides a general template for stochastic

Algorithm 1 A Template for Stochastic Gradient Methods for Training GANs

```
for number of training iterations do
    for $k$ steps do
        Sample minibatch of $m$ samples $z^1, \ldots, z^m$ from $\gamma$.
        Sample minibatch of $m$ data $x^1, \ldots, x^m$ from $\mu$
        Update the discriminator network $D_\omega$ by ascending its stochastic gradient:
            $\frac{1}{m}\sum_{i=1}^{m} \nabla_\omega \left(\log D_\omega(x^i) + \log(1 - D_\omega(G_\theta(z^i)))\right)$
    end for
    Sample minibatch of $m$ samples $z^1, \ldots, z^m$ from $\gamma$.
    Update the generator network $G_\theta$ by descending its stochastic gradient:
        $\frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \left(\log(1 - D_\omega(G_\theta(z^i)))\right)$
end for
```

gradient methods for training GANs. One may use various optimizers such as momentum methods.

4 f-GAN

Recall that when the discriminator D reaches optimum given by

$$D^*(x) = \frac{p(x)}{p(x) + q(x)},$$

we have

$$V(G, D^*) = D_{JS}(p||q) - \log 4.$$

Then an optimal generator can be found by computing q that minimizes $D_{JS}(p||q)$. Here, the Jensen-Shannon divergence $D_{JS}(p||q)$ measures the discrepancy between p and q. In fact, one may generalize GANs by considering the notion of f-divergence. The f-divergence of two distributions p and q is defined as

$$D_f(p||q) = \int_{\mathbb{R}^d} q(x) f\left(\frac{p(x)}{q(x)}\right) dx.$$

Based on the Fenchel conjugate of f, it follows that

$$\begin{aligned}
D_f(p||q) &= \int_{\mathbb{R}^d} q(x) \sup_t \left\{ t\frac{p(x)}{q(x)} - f^*(t) \right\} dx \\
&= \int_{\mathbb{R}^d} \sup_t \left\{ t \cdot p(x) - f^*(t) \cdot q(x) \right\} dx \\
&\geq \sup_{T \in \mathcal{T}} \int_{\mathbb{R}^d} \left(p(x)T(x) - q(x) f^*(T(x)) \right) dx \\
&= \sup_{T \in \mathcal{T}} \left(\mathbb{E}_{x\sim\mu} \left[T(x) \right] - \mathbb{E}_{x\sim\nu} \left[f^*(T(x)) \right] \right)
\end{aligned}$$

where $\mathcal{T}$ is a family of functions. Based on this formulation, we can consider

$$\min_\nu \max_T \quad \mathbb{E}_{x\sim\mu} \left[T(x) \right] - \mathbb{E}_{x\sim\nu} \left[f^*(T(x)) \right].$$

This framework is referred to as the f-GAN and the Variational Divergence Minimization (VDM) framework [NCT16].

Example 18.4. Let f be given by

$$f(y) = -\log(y+1) + \log y + (y+1)\log 2.$$

Then $f^*(t) = -\log(2 - e^t)$. In this case, we have

$$\min_\nu \max_T \quad \mathbb{E}_{x\sim\mu} \left[T(x) \right] + \mathbb{E}_{x\sim\nu} \left[\log(2 - e^{T(x)}) \right].$$

Setting

$$D(x) = 1 - \frac{1}{2} e^{T(x)},$$

the formulation is equivalent to

$$\min_\nu \max_T \quad \mathbb{E}_{x\sim\mu} \left[\log D(x) \right] + \mathbb{E}_{x\sim\nu} \left[\log(1 - D(x)) \right] + \log 4,$$

which is the first version of GAN.

We may model the function $T(x)$ by a neural network given by

$$T(x) = T_\omega(x) = g_f(V_\omega(x))$$

where g_f is an output activation function specific to the f-divergence and V_ω is a neural network parametrized by ω. Then the minimax optimization framework boils down to

$$\min_\nu \max_\omega \quad \mathbb{E}_{x\sim\mu} \left[g_f(V_\omega(x)) \right] + \mathbb{E}_{x\sim\nu} \left[-f^*(g_f(V_\omega(x))) \right].$$

To train an f-GAN, one may generalize Algorithm 1 (see [Wan20]). However, the original paper [NCT16] uses what is called the single-step gradient method that does not use inner loops to solve the inner maximization problem. The single-step gradient method is a stochastic version of gradient descent ascent.

References

[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 Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. 2

[NCT16] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. 4, 4

[Wan20] Yang Wang. A mathematical introduction to generative adversarial nets (gan), 2020. 4