MonoFlow:

Rethinking Divergence GANs via the Perspective of Wasserstein Gradient Flows

Mingxuan Yi[1], Zhanxing Zhu[2 3], Song Liu[1]

[1] University of Bristol [2] Changping National Lab [3] Peking University

The adversarial game [Goodfellow et al., 2014]:

$$\min_g \max_d V(g, d) = \mathbb{E}_{\mathbf{x} \sim p_{\mathrm{data}}} \{ \log \sigma[d(\mathbf{x})] \} + \quad \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} \{ \log (1 - \sigma[d(g(\mathbf{z}))]) \} \tag{1}$$

Existing issues:

1. The discriminator $d(\mathbf{x})$ loses the dependence on the generator's parameter. Integrating out $\mathbf{x}$ in the expectation, V is not a function of g.
2. The generator only minimizes the second term of the Jensen-Shannon divergence $\mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} \{ \log (1 - \sigma[d(g(\mathbf{z}))]) \}$ which is, however, a KL divergence up to a constant.
3. Practical algorithms are inconsistent with the theory, a heuristic trick "non-saturated loss" is commonly used to mitigate the gradient vanishing problem. The NS loss takes the form $-\mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} \{ \log \sigma[d(g(\mathbf{z}))] \}$.

We can even modify the generator loss to the logit loss $-\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\{d(g(\mathbf{z}))\}$ or the arcsinh loss $-\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\left\{\operatorname{arcsinh}\left(d(g(\mathbf{z}))\right)\right\}$.

Figure 1: Generated Celeb-A faces with the logit loss and the arcsinh loss.

All of the above generator losses satisfy

$$-\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\{h[d(g(\mathbf{z}))]\},$$

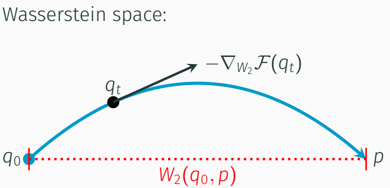
where $h\colon \mathbb{R} \to \mathbb{R}$ is a monotonically increasing function with $h'(\cdot) > 0$.

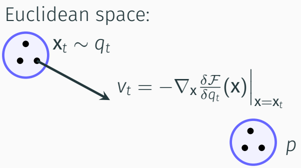
Introduction

The adversarial game framework lacks a rigorous explanation to these issues.

GAN theory needs to be reformulated!

Wasserstein Gradient Flows





The marginal q_t evolves along the gradient flow to decrease $\mathcal{F}(q_t)$ and the associated particles evolve with the vector field v_t [Ambrosio et al., 2008].

Probability Flow ODEs

Given f-divergences

$$\mathcal{F}(q_t) = \int f(r_t(\mathbf{x}))\, q_t(\mathbf{x})\mathrm{d}\mathbf{x}, \quad r_t(\mathbf{x}) = \frac{p(\mathbf{x})}{q_t(\mathbf{x})},$$

where $f''(\mathbf{x}) > 0$ implies f is strictly convex.

Wasserstein gradient flows define a probability flow ODE in Euclidean space,

$$\mathrm{d}\mathbf{x}_t = v_t(\mathbf{x}_t)\mathrm{d}t$$

The vector field of the probability flow ODE:

$$v_t(\mathbf{x}) = r_t(\mathbf{x})^2 f''(r_t(\mathbf{x}))\nabla_{\mathbf{x}} \log r_t(\mathbf{x}), \tag{2}$$

such that the non-negative term $r_t(\mathbf{x})^2 f''(r_t(\mathbf{x}))$ rescales $\nabla_{\mathbf{x}} \log r_t(\mathbf{x})$.

MonoFlow

MonoFlow is defined by the following ODE:

$$\mathrm{d}\mathbf{x}_t = \nabla_{\mathbf{x}} h\big(\log r_t(\mathbf{x}_t)\big)\mathrm{d}t = h'\big(\log r_t(\mathbf{x}_t)\big)\nabla_{\mathbf{x}} \log r_t(\mathbf{x}_t)\mathrm{d}t$$

where $h: \mathbb{R} \to \mathbb{R}$ is a monotonically increasing function with $h'(\cdot) > 0$,

Property

Implicitly defines Wasserstein gradient flows of f-divergences: Given a function h with $h'(\cdot) > 0$, there exists a strictly convex function f satisfying

$$h(\log r) = r f'(r) - f(r),$$

MonoFlow is the probability Flow ODE of the above f-divergence.

Simulating MonoFlow

1. Two sample density ratio estimation (training the discriminator):

$$\max_{d} \mathbb{E}_{\mathsf{x}\sim p}\left[\phi\big(d(\mathsf{x})\big)\right] + \mathbb{E}_{\mathsf{x}\sim q_t}\left[\psi\big(d(\mathsf{x})\big)\right], \qquad (3)$$

where ϕ and ψ are scalar functions. Under certain conditions, the optimal d^* satisfies

$$r_t(\mathsf{x}) := p(\mathsf{x})/q_t(\mathsf{x}) = -\psi'\big(d^*(\mathsf{x})\big)/\phi'\big(d^*(\mathsf{x})\big)$$

The vector field is obtained via:

$$v_t(\mathsf{x}) = \nabla_{\mathsf{x}} h\big(\log r_t(\mathsf{x})\big)$$

2. Learning to parameterize MonoFlow (distilling):

- Sample $\mathsf{x}_t = g_\theta(\mathsf{z}) \sim q_t, \mathsf{z} \sim p_\mathsf{z}$, where g_θ is a generator taking as input random noises z.
- Move particles along the vector field with step size α, i.e. forward Euler method, $\mathsf{x}_{t+\alpha} = \mathsf{x}_t + \alpha v_t(\mathsf{x}_t)$
- Minimize the loss

$$\min_\theta \mathbb{E}_{\mathsf{z}\sim p_\mathsf{z}} \|g_\theta(\mathsf{z}) - \mathsf{x}_{t+\alpha}\|_2^2 \Longleftrightarrow \min_\theta -\mathbb{E}_{\mathsf{z}\sim p_\mathsf{z}}[h(\log r_t(g_\theta(\mathsf{z})))],$$

to encourage the generator to draw particles more similar to $\mathsf{x}_{t+\alpha}$.

Unified Formulation of Divergence GANs

The objectives for the discriminator and the generator can be entirely different,

$$\max_{d} \mathbb{E}_{\mathsf{x}\sim p_{\mathrm{data}}}\left[\phi\big(d(\mathsf{x})\big)\right] + \mathbb{E}_{\mathsf{z}\sim p_{\mathsf{z}}}\left[\psi\big(d(g(\mathsf{z}))\big)\right]$$
$$\min_{g} -\mathbb{E}_{\mathsf{z}\sim p_{\mathsf{z}}}\left[h_{\mathcal{T}}\big(d(g(\mathsf{z}))\big)\right],$$

where $h_{\mathcal{T}}(d) = h\big(\log(\mathcal{T}(d))\big)$, $\mathcal{T}(d) = -\psi'\big(d\big)/\phi'\big(d\big)$ and h can be any increasing function with $h'(\cdot) > 0$.

Let's go back to the GAN [Goodfellow et al., 2014]. For a binary classification problem,

$$\max_{d} \mathbb{E}_{\mathsf{x}\sim p_{\mathrm{data}}}\{\log \sigma[d(\mathsf{x})]\} + \quad \mathbb{E}_{\mathsf{z}\sim p_{\mathsf{z}}}\{\log(1-\sigma[d(g(\mathsf{z}))])\},$$

where $\phi(d) = \log \sigma(d)$ and $\psi(d) = \log(1-\sigma(d))$.

The optimal d^* satisfies

$$\begin{aligned} r(\mathsf{x}) &:= p_{\mathrm{data}}(\mathsf{x})/p_g(\mathsf{x}) = -\psi'\left(d^*(\mathsf{x})\right)/\phi'\left(d^*(\mathsf{x})\right) \\ &\implies d^*(\mathsf{x}) = \log r(\mathsf{x}) \end{aligned}$$

Empirical Results

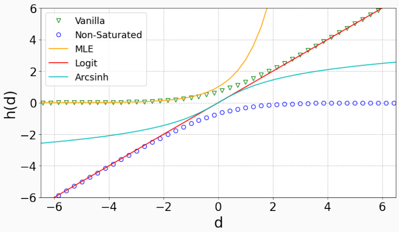


Figure 2: Generator losses

1. Vanilla loss: $h(d) = -\log(1 - \sigma(d))$
2. Non-saturated (NS) loss: $h(d) = \log(\sigma(d))$ ✓
3. Maximum likelihood estimation (MLE): $h(d) = \exp(d)$
4. Logit loss: $h(d) = d$ ✓
5. Arcsinh loss: $h(d) = \text{arcsinh}(d)$ ✓

An Embarrassingly Simple Trick to Fix the Vanilla GAN

Shifting the vanilla loss

$$h(d) = -\log(1 - \sigma(d + C))$$

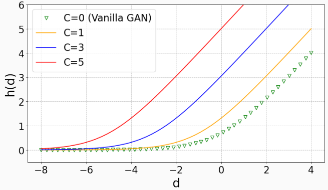


Figure 3: Generator losses

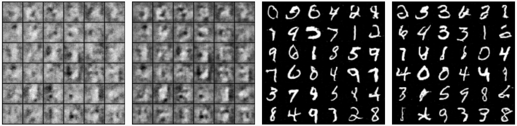


Figure 4: From left to right $C = 0, 1, 3, 5$

References

Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures.* Springer Science & Business Media, 2008.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *In NeurIPS*, 2014.