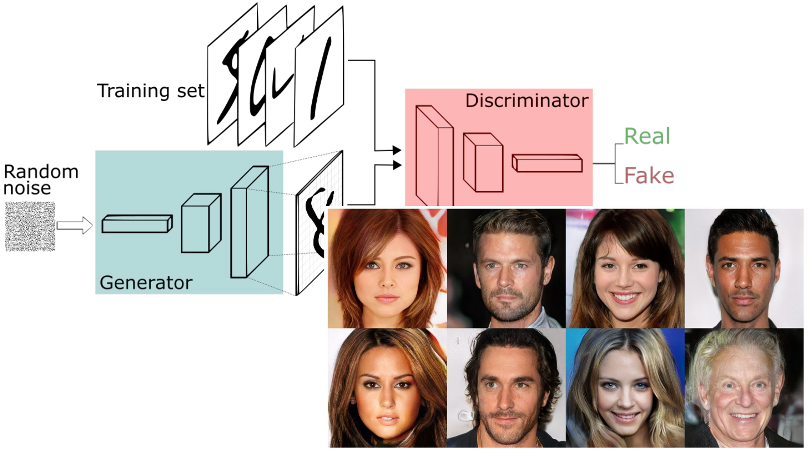
Probability Functional Descent:

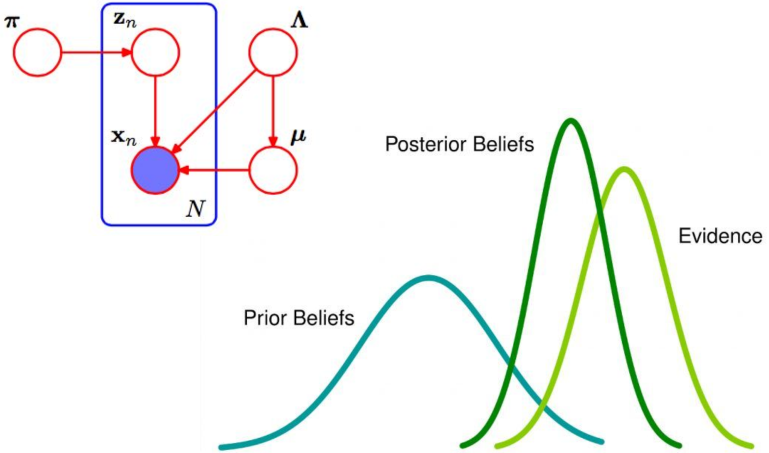
A Unifying Perspective on GANs, VI, and RL

Casey Chu *<caseychu@stanford.edu>* Jose Blanchet Peter Glynn

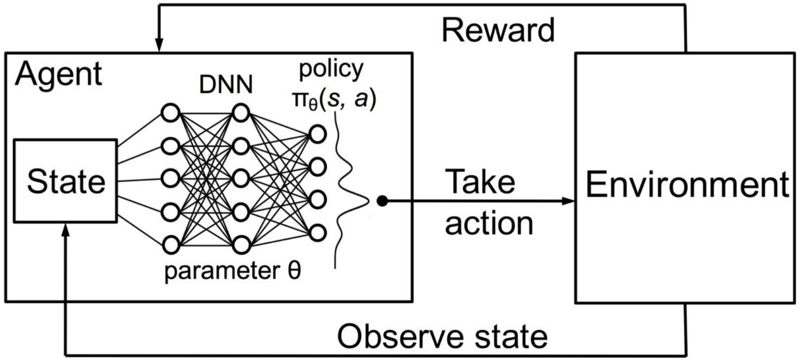
Generative adversarial networks



Variational inference



Reinforcement learning



Lots of algorithms...

Generative adversarial networks

- Minimax GAN
- Non-saturating GAN
- Wasserstein GAN
- f-GAN

Variational inference

- Black-box variational inference
- Adversarial variational Bayes

Reinforcement learning

- Policy iteration/Q-learning
- REINFORCE
- Deep deterministic policy gradient
- Dual actor-critic
- Soft actor-critic

Lots of algorithms… **but same “under the hood”!**

Generative adversarial networks

- Minimax GAN
- Non-saturating GAN
- Wasserstein GAN
- f-GAN

Variational inference

- Black-box variational inference
- Adversarial variational Bayes

Reinforcement learning

- Policy iteration/Q-learning
- REINFORCE
- Deep deterministic policy gradient
- Dual actor-critic
- Soft actor-critic

They all minimize *probability functionals*

Generative adversarial networks

$$J_{\mathrm{GAN}}(\mu) = D(\mu \,||\, \nu_0)$$

Variational inference

$$J_{\mathrm{VI}}(q) = D_{\mathrm{KL}}(q(\theta) \,||\, p(\theta|x))$$

Reinforcement learning

$$J_{\mathrm{RL}}(\pi) = -\mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R_t\Big]$$

They all minimize *probability functionals*

$$J : \mathcal{P}(X) \to \mathbb{R}$$

Probability distributions over elements in X
(in the GAN setting: generators of images)

Review: gradient descent in **R**^{n}

$$f : \mathbb{R}^n \to \mathbb{R}$$

$$x_{\text{new}} \leftarrow x_0 - \alpha \nabla f(x_0)$$

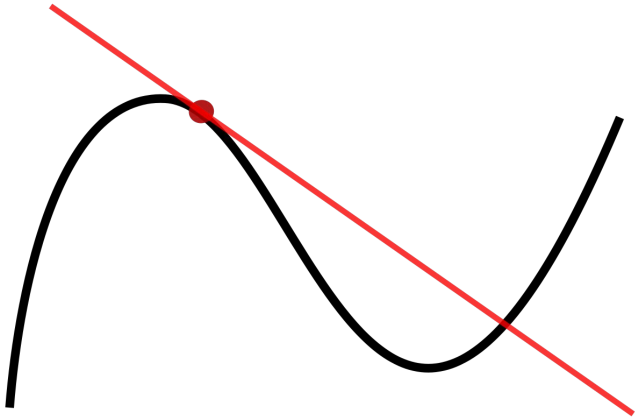
Review: gradient descent in $\mathbf{R}^n$

$$f_{\text{linear}}(x) = f(x_0) + (x - x_0) \cdot \nabla f(x_0)$$

Review: gradient descent in $\mathbf{R}^n$

$$f(x) \approx f_{\text{linear}}(x) = f(x_0) + (x - x_0) \cdot \nabla f(x_0)$$

Gradient descent says: **if we move *x* such that f_{linear} decreases, hopefully *f* will decrease as well**



Review: gradient descent in $\mathbf{R}^n$

$$f(x) \approx f_{\text{linear}}(x) = f(x_0) + (x - x_0) \cdot \nabla f(x_0)$$

Gradient descent says: **if we move *x* such that *f*$_{\textbf{linear}}$ decreases, hopefully *f* will decrease as well**

$$x_{\text{new}} \leftarrow x_0 - \alpha \nabla f(x_0)$$

$$\begin{aligned} f_{\text{linear}}(x_{\text{new}}) &= f(x_0) + (x_0 - \alpha \nabla f(x_0) - x_0) \cdot \nabla f(x_0) \\ &= f(x_0) - \alpha ||\nabla f(x_0)||^2 \\ &\leq f_{\text{linear}}(x_0) \end{aligned}$$

Review: gradient descent in $\mathbf{R}^n$

$$f(x) \approx f_{\text{linear}}(x) = \cancel{f(x_0)} + (x - x_0) \cdot \nabla f(x_0)$$

Gradient descent says: **if we move *x* such that f_{linear} decreases, hopefully *f* will decrease as well**

$$x_{\text{new}} \leftarrow x_0 - \alpha \nabla f(x_0)$$

$$\begin{aligned} f_{\text{linear}}(x_{\text{new}}) &= f(x_0) + (x_0 - \alpha \nabla f(x_0) - x_0) \cdot \nabla f(x_0) \\ &= f(x_0) - \alpha ||\nabla f(x_0)||^2 \\ &\leq f_{\text{linear}}(x_0) \end{aligned}$$

Review: gradient descent in $\mathbf{R}^n$

$$f(x) \approx f_{\text{linear}}(x) = \cancel{f(x_0)} + (x - \cancel{x_0}) \cdot \nabla f(x_0)$$

Gradient descent says: **if we move *x* such that f_{linear} decreases, hopefully *f* will decrease as well**

$$x_{\text{new}} \leftarrow x_0 - \alpha \nabla f(x_0)$$

$$\begin{aligned} f_{\text{linear}}(x_{\text{new}}) &= f(x_0) + (x_0 - \alpha \nabla f(x_0) - x_0) \cdot \nabla f(x_0) \\ &= f(x_0) - \alpha ||\nabla f(x_0)||^2 \\ &\leq f_{\text{linear}}(x_0) \end{aligned}$$

Review: gradient descent in $\mathbf{R}^n$

$$f(x) \approx f_{\text{linear}}(x) = \cancel{f(x_0)} + (x - \cancel{x}_0) \cdot \nabla f(x_0)$$

Gradient descent says: **find *x*$_{\textbf{new}}$ such that**

$$x_{\text{new}} \cdot \nabla f(x_0) \leq x_0 \cdot \nabla f(x_0)$$

Linear approximation of a probability functional

$$f(x) \approx f(x_0) + \nabla f(x_0) \cdot (x - x_0)$$

$$J(\mu) \approx J(\mu_0) + \int \nabla J(\mu_0)\, d(\mu - \mu_0)$$

$$J : \mathcal{P}(X) \to \mathbb{R} \qquad\qquad \nabla J(\mu_0) : X \to \mathbb{R}$$

Linear approximation of a probability functional

$$f(x) \approx f(x_0) + \nabla f(x_0) \cdot (x - x_0)$$

$$\begin{aligned} J(\mu) &\approx J(\mu_0) + \int \nabla J(\mu_0)\, d(\mu - \mu_0) \\ &= J(\mu_0) + \mathbb{E}_{x \sim \mu}[\nabla J(\mu_0)(x)] - \mathbb{E}_{x \sim \mu_0}[\nabla J(\mu_0)(x)] \end{aligned}$$

$$J : \mathcal{P}(X) \to \mathbb{R} \qquad\qquad \nabla J(\mu_0) : X \to \mathbb{R}$$

Linear approximation of a probability functional

$$f(x) \approx f(x_0) + \nabla f(x_0) \cdot (x - x_0)$$

$$J(\mu) \approx J(\mu_0) + \int \nabla J(\mu_0)\, d(\mu - \mu_0)$$

$$= \cancel{J(\mu_0)} + \mathbb{E}_{x\sim\mu}[\nabla J(\mu_0)(x)] - \cancel{\mathbb{E}_{x\sim\mu_0}[\nabla J(\mu_0)(x)]}$$

$$J : \mathcal{P}(X) \to \mathbb{R} \qquad \nabla J(\mu_0) : X \to \mathbb{R}$$

Linear approximation of a probability functional

$$f(x) \approx f(x_0) + \nabla f(x_0) \cdot (x - x_0)$$

$$\begin{aligned} J(\mu) &\approx J(\mu_0) + \int \nabla J(\mu_0)\, d(\mu - \mu_0) \\ &= \cancel{J(\mu_0)} + \mathbb{E}_{x\sim\mu}[\nabla J(\mu_0)(x)] - \cancel{\mathbb{E}_{x\sim\mu_0}[\nabla J(\mu_0)(x)]} \\ &= \mathbb{E}_{x\sim\mu}[\nabla J(\mu_0)(x)] + \text{const.} \end{aligned}$$

$$J : \mathcal{P}(X) \to \mathbb{R} \qquad \nabla J(\mu_0) : X \to \mathbb{R}$$

Linear approximation of a probability functional

$$f(x) \approx f(x_0) + \nabla f(x_0) \cdot (x - x_0)$$

$$\begin{aligned} J(\mu) &\approx J(\mu_0) + \int \nabla J(\mu_0)\, d(\mu - \mu_0) \\ &= \cancel{J(\mu_0)} + \mathbb{E}_{x\sim\mu}[\nabla J(\mu_0)(x)] - \cancel{\mathbb{E}_{x\sim\mu_0}[\nabla J(\mu_0)(x)]} \\ &= \mathbb{E}_{x\sim\mu}[\nabla J(\mu_0)(x)] + \text{const.} \end{aligned}$$

$$J : \mathcal{P}(X) \to \mathbb{R} \qquad \nabla J(\mu_0) : X \to \mathbb{R}$$

von Mises influence function

Probability functional descent

1. Compute or approximate $\nabla J(\mu_0)$
2. Find a distribution μ such that

$$\mathbb{E}_{x\sim\mu}[\nabla J(\mu_0)(x)] \leq \mathbb{E}_{x\sim\mu_0}[\nabla J(\mu_0)(x)]$$

$$J : \mathcal{P}(X) \to \mathbb{R}$$

$$\nabla J(\mu_0) : X \to \mathbb{R}$$

von Mises influence function

Probability functional descent

1. Compute or approximate $\nabla J(\mu_0)$
2. Find a distribution μ such that

$$\mathbb{E}_{x\sim\mu}[\nabla J(\mu_0)(x)] \leq \mathbb{E}_{x\sim\mu_0}[\nabla J(\mu_0)(x)]$$

2 (in practice). Parameterize μ with θ and take a gradient descent step on the function

$$\theta \mapsto \mathbb{E}_{x\sim\mu_\theta}[\nabla J(\mu_{\theta_0})(x)]$$

Probability functional descent

1. Compute or approximate $\nabla J(\mu_0)$
2. Find a distribution μ such that

$$\mathbb{E}_{x\sim\mu}[\nabla J(\mu_0)(x)] \leq \mathbb{E}_{x\sim\mu_0}[\nabla J(\mu_0)(x)]$$

1 (in practice). Fit a neural network to $\nabla J(\mu_0) : X \to \mathbb{R}$

2 (in practice). Parameterize μ with θ and take a gradient descent step on the function

$$\theta \mapsto \mathbb{E}_{x\sim\mu_\theta}[\overbrace{\boxed{\nabla J(\mu_{\theta_0})}}^{\text{Neural network}}(x)]$$

PFD for **minimax GANs**

$$J(\mu) = D_{\text{JS}}(\mu || \nu_0)$$

$$\nabla J(\mu)(x) = \frac{1}{2} \log \frac{\mu(x)}{\mu(x) + \nu_0(x)}$$

Probability functional descent

1. Compute or approximate $\nabla J(\mu_0)$
2. Take a gradient descent step on $\theta \mapsto \mathbb{E}_{x \sim \mu_\theta}[\nabla J(\mu_{\theta_0})(x)]$

PFD for **minimax GANs**

$$J(\mu) = D_{\mathrm{JS}}(\mu || \nu_0)$$

1) Fit a neural network to this
(a "discriminator") $D(x)$

$$\nabla J(\mu)(x) = \frac{1}{2} \log \boxed{\frac{\mu(x)}{\mu(x) + \nu_0(x)}}$$

2) Take a gradient step on the generator

$$\theta \mapsto \mathbb{E}_{x \sim \mu_\theta}[\tfrac{1}{2} \log D(x)]$$

Probability functional descent

1. Compute or approximate $\nabla J(\mu_0)$
2. Take a gradient descent step on $\theta \mapsto \mathbb{E}_{x \sim \mu_\theta}[\nabla J(\mu_{\theta_0})(x)]$

PFD for **variational inference**

$$J(q) = D_{\mathrm{KL}}(q(z) \,||\, p(z|x))$$

$$\nabla J(q)(z) = \log \frac{q(z)}{p(x|z)\,p(z)}$$

Probability functional descent

1. Compute or approximate $\nabla J(\mu_0)$
2. Take a gradient descent step on $\theta \mapsto \mathbb{E}_{x \sim \mu_\theta}[\nabla J(\mu_{\theta_0})(x)]$

PFD for **variational inference**

$$J(q) = D_{\mathrm{KL}}(q(z) \,||\, p(z|x))$$

$$\nabla J(q)(z) = \boxed{\log \frac{q(z)}{p(x|z)\,p(z)}}$$

1) We can compute this exactly, no need to fit a network to it

2) Take a gradient step on the ELBO

$$\theta \mapsto \mathbb{E}_{z \sim q_\theta}\left[\log \frac{q_{\theta_0}(z)}{p(x|z)\,p(z)}\right]$$

Probability functional descent

1. Compute or approximate $\nabla J(\mu_0)$
2. Take a gradient descent step on $\theta \mapsto \mathbb{E}_{x \sim \mu_\theta}[\nabla J(\mu_{\theta_0})(x)]$

PFD for **actor-critic RL algorithms**

$$J(\pi) = -\mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R_t\Big]$$

$$\nabla J(\pi)(s,a) = -(1-\gamma)\big(Q^\pi(s,a) - V^\pi(s)\big)$$

Probability functional descent

1. Compute or approximate $\nabla J(\mu_0)$
2. Take a gradient descent step on $\theta \mapsto \mathbb{E}_{x\sim\mu_\theta}[\nabla J(\mu_{\theta_0})(x)]$

PFD for **actor-critic RL algorithms**

$$J(\pi) = -\mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R_t\Big]$$

1) Fit a neural network to this advantage function (e.g. by minimizing the Bellman residuals)

$$\nabla J(\pi)(s,a) = -(1-\gamma)\boxed{\big(Q^\pi(s,a) - V^\pi(s)\big)}$$

2) Take a gradient step on

$$\theta \mapsto -\mathbb{E}_{(s,a)\sim\pi_\theta}\left[Q^{\pi_{\theta_0}}(s,a) - V^{\pi_{\theta_0}}(s)\right]$$

Probability functional descent

1. Compute or approximate $\nabla J(\mu_0)$
2. Take a gradient descent step on $\theta \mapsto \mathbb{E}_{x\sim\mu_\theta}[\nabla J(\mu_{\theta_0})(x)]$

PFD for **minimax GANs**

$$J(\mu) = D_{\mathrm{JS}}(\mu || \nu_0)$$

1) Fit a neural network to this
(a "discriminator") $D(x)$

$$\nabla J(\mu)(x) = \frac{1}{2} \log \boxed{\frac{\mu(x)}{\mu(x) + \nu_0(x)}}$$

2) Take a gradient step on the generator

$$\theta \mapsto \mathbb{E}_{x \sim \mu_\theta}[\tfrac{1}{2} \log D(x)]$$

Probability functional descent

1. Compute or approximate $\nabla J(\mu_0)$
2. Take a gradient descent step on $\theta \mapsto \mathbb{E}_{x \sim \mu_\theta}[\nabla J(\mu_{\theta_0})(x)]$

PFD for **any convex function**

convex conjugate

$$J^\star(\varphi) = \sup_{\mu \in \mathcal{M}(X)} \int \varphi \, d\mu - J(\mu)$$

$$J(\mu) \text{ convex}$$

$$\nabla J(\mu) = \arg\max_{\varphi \in \mathcal{C}(X)} \left[\mathbb{E}_{x\sim\mu}[\varphi(x)] - J^\star(\varphi) \right]$$

Probability functional descent

1. Compute or approximate $\nabla J(\mu_0)$
2. Take a gradient descent step on $\theta \mapsto \mathbb{E}_{x\sim\mu_\theta}[\nabla J(\mu_{\theta_0})(x)]$

PFD for **any convex function**

convex conjugate

$$J^\star(\varphi) = \sup_{\mu \in \mathcal{M}(X)} \int \varphi \, d\mu - J(\mu)$$

$$J(\mu) \text{ convex}$$

1) Fit a neural network by maximizing the inner objective (can use SGD)

$$\nabla J(\mu) = \arg\max_{\varphi \in \mathcal{C}(X)} \left[\mathbb{E}_{x\sim\mu}[\varphi(x)] - J^\star(\varphi) \right]$$

2) Take a gradient step on

$$\theta \mapsto \mathbb{E}_{x\sim\mu_\theta}[\varphi(x)]$$

Probability functional descent

1. Compute or approximate $\nabla J(\mu_0)$
2. Take a gradient descent step on $\theta \mapsto \mathbb{E}_{x\sim\mu_\theta}[\nabla J(\mu_{\theta_0})(x)]$

PFD for **any convex function**

convex conjugate

$$J^\star(\varphi) = \sup_{\mu \in \mathcal{M}(X)} \int \varphi \, d\mu - J(\mu)$$

$$J(\mu) \text{ convex}$$

1) Fit a neural network by maximizing the inner objective (can use SGD)

$$\nabla J(\mu) = \arg\max_{\varphi \in \mathcal{C}(X)} \left[\mathbb{E}_{x\sim\mu}[\varphi(x)] - J^\star(\varphi) \right]$$

Neatly summarized as a minimax game!

$$\inf_{\mu \in \mathcal{P}(X)} \sup_{\varphi \in \mathcal{C}(X)} \mathbb{E}_{x\sim\mu}[\varphi(x)] - J^\star(\varphi)$$

2) Take a gradient step on

$$\theta \mapsto \mathbb{E}_{x\sim\mu_\theta}[\varphi(x)]$$

Probability functional descent

1. Compute or approximate $\nabla J(\mu_0)$
2. Take a gradient descent step on $\theta \mapsto \mathbb{E}_{x\sim\mu_\theta}[\nabla J(\mu_{\theta_0})(x)]$

PFD for **Wasserstein GAN**

$$J(\mu) = W_1(\mu, \nu_0)$$

1) Fit a neural network by maximizing the inner objective (can use SGD)

$$\nabla J(\mu) = \arg\max_{\varphi \in \mathcal{C}(X)} \left[\mathbb{E}_{x\sim\mu}[\varphi(x)] - J^\star(\varphi) \right]$$

2) Take a gradient step on

$$\theta \mapsto \mathbb{E}_{x\sim\mu_\theta}[\varphi(x)]$$

Probability functional descent

1. Compute or approximate $\nabla J(\mu_0)$
2. Take a gradient descent step on $\theta \mapsto \mathbb{E}_{x\sim\mu_\theta}[\nabla J(\mu_{\theta_0})(x)]$

PFD for **Wasserstein GAN**

$$J^\star(\varphi) = \begin{cases} \mathbb{E}_{x\sim\nu_0}[\varphi(x)] & \text{if } \varphi \text{ is 1-Lipschitz} \\ \infty & \text{otherwise} \end{cases}$$

$$J(\mu) = W_1(\mu, \nu_0)$$

1) Fit a neural network by maximizing the inner objective (can use SGD)

$$\nabla J(\mu) = \arg\max_{\varphi\in\mathcal{C}(X)} \left[\mathbb{E}_{x\sim\mu}[\varphi(x)] - J^\star(\varphi)\right]$$

2) Take a gradient step on

$$\theta \mapsto \mathbb{E}_{x\sim\mu_\theta}[\varphi(x)]$$

Probability functional descent

1. Compute or approximate $\nabla J(\mu_0)$
2. Take a gradient descent step on $\theta \mapsto \mathbb{E}_{x\sim\mu_\theta}[\nabla J(\mu_{\theta_0})(x)]$

PFD for **Wasserstein GAN**

$$J(\mu) = W_1(\mu, \nu_0)$$

$$J^\star(\varphi) = \begin{cases} \mathbb{E}_{x\sim\nu_0}[\varphi(x)] & \text{if } \varphi \text{ is 1-Lipschitz} \\ \infty & \text{otherwise} \end{cases}$$

1) Fit a neural network by maximizing the inner objective (can use SGD)

$$\nabla J(\mu) = \underset{\varphi\in\mathrm{Lip}_1(X)}{\arg\max}\left[\mathbb{E}_{x\sim\mu}[\varphi(x)] - \mathbb{E}_{x\sim\nu_0}[\varphi(x)]\right]$$

2) Take a gradient step on

$$\theta \mapsto \mathbb{E}_{x\sim\mu_\theta}[\varphi(x)]$$

Probability functional descent

1. Compute or approximate $\nabla J(\mu_0)$
2. Take a gradient descent step on $\theta \mapsto \mathbb{E}_{x\sim\mu_\theta}[\nabla J(\mu_{\theta_0})(x)]$

Lots of algorithms... **but same "under the hood"!**

Generative adversarial networks

- Minimax GAN
- Non-saturating GAN
- Wasserstein GAN
- f-GAN

Variational inference

- Black-box variational inference
- Adversarial variational Bayes

Reinforcement learning

- Policy iteration/Q-learning
- REINFORCE
- Deep deterministic policy gra
- Dual actor-critic
- Soft actor-critic

Gradient descent in the space of probability distributions!