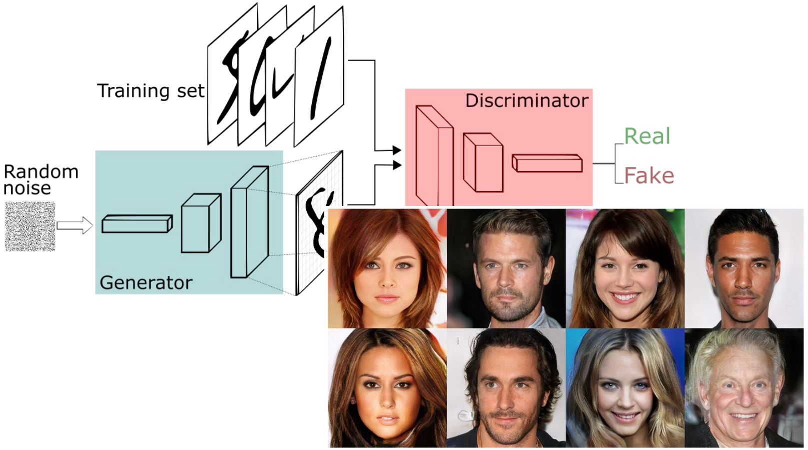
Probability Functional Descent:

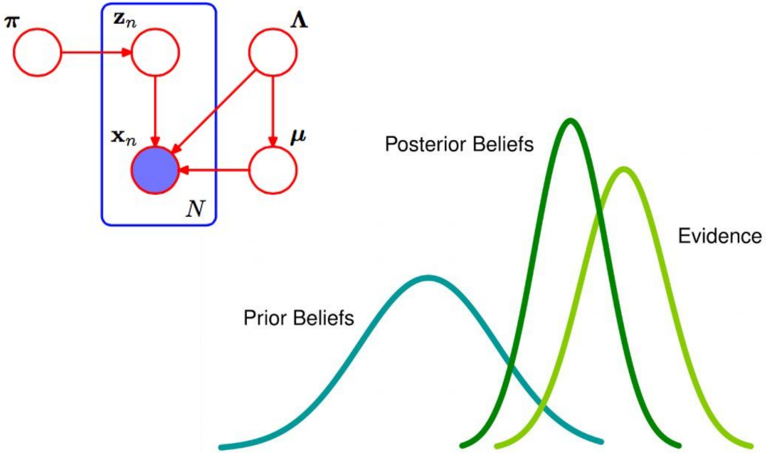
A Unifying Perspective on GANs, VI, and RL

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

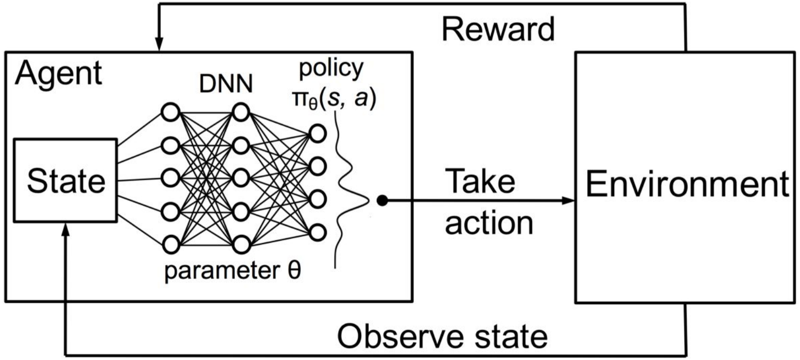
Generative adversarial networks



Variational inference



Reinforcement learning

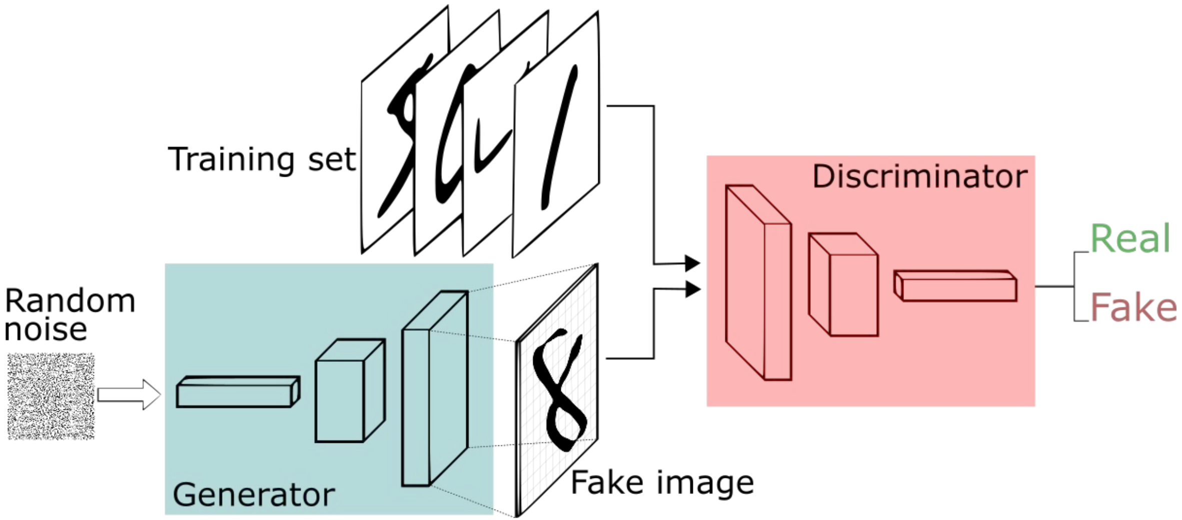


Generative adversarial networks

StyleGAN (Karras et al. 2018)

GANs can generate incredibly realistic images!

Generative adversarial networks



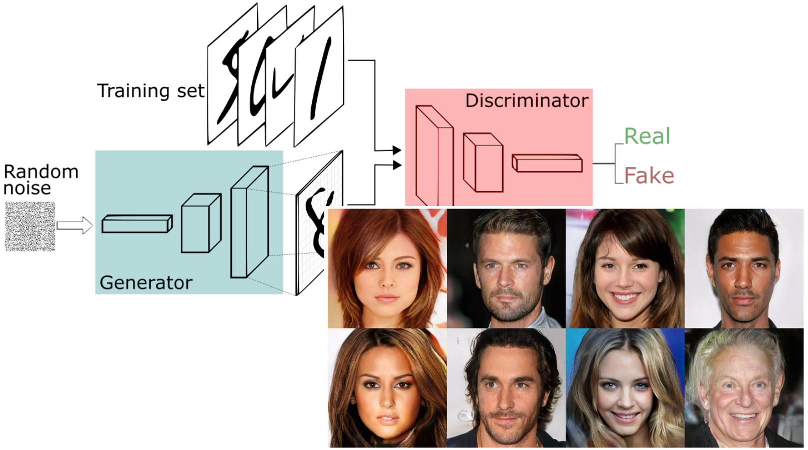
$$\min_{\theta} \max_{\phi} \mathbb{E}_{x \sim p_{\text{data}}}[\log D_{\phi}(x)] + \mathbb{E}_{z \sim \mathcal{N}}[\log(1 - D_{\phi}(G_{\theta}(z)))]$$

Variational inference

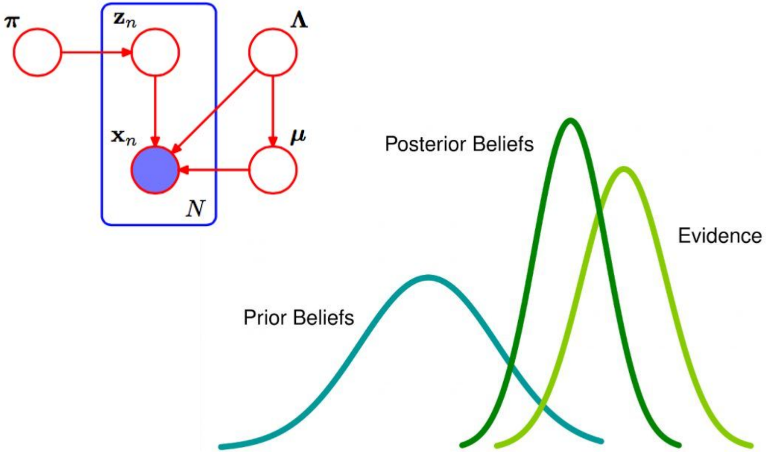
$$\underbrace{p(z|x)}_{\text{posterior}} = \frac{\overbrace{p(x|z)}^{\text{likelihood}}\,\overbrace{p(z)}^{\text{prior}}}{\int p(x|z)\,p(z)\,dz}$$

$$\mathrm{KL}(\overbrace{\boxed{q_\theta(z)}}^{\text{approximate posterior}} \,||\, p(z|x)) = \log p(x) + \overbrace{\boxed{\mathbb{E}_{z\sim q_\theta}\left[\log \frac{q_\theta(z)}{p(x|z)\,p(z)}\right]}}^{\text{negative ELBO}}$$

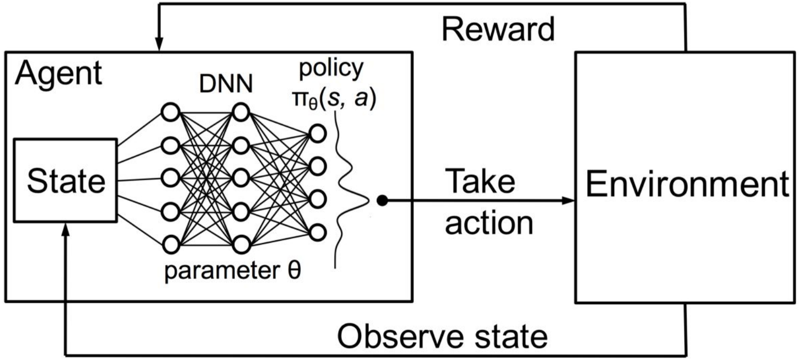
Generative adversarial networks



Variational inference



Reinforcement learning



Hidden connections…

Generative adversarial networks

Variational inference

Reinforcement learning

Hidden connections...

Generative adversarial networks | **Variational inference**

Reinforcement learning

GANs:

The **generator** produces an image.
The **discriminator** judges how good the image is.

RL:

The **actor (policy)** produces an action.
The **critic (value function)** judges how good the action is.

Hidden connections...

Generative adversarial networks

Variational inference

Reinforcement learning

RL as inference: Maximum-entropy reinforcement learning can be implemented as a variational inference procedure.

Hidden connections...

Generative adversarial networks

Variational inference

Reinforcement learning

A lot of the same techniques:

- Reparameterization trick
- Log-derivative trick/REINFORCE
- Stochastic gradient descent
- Neural networks

Hidden connections...

Generative adversarial networks

Variational inference

Reinforcement learning

There are a surprising number of connections among these fields: is there an underlying explanation?

Hidden connections...

Generative adversarial networks

We are interested in an optimal generator. The **discriminator** tells us how to improve the **generator**.

Variational inference

We are interested in an optimal approximate posterior. The **ELBO** tells us how to improve the **approximate posterior**.

Reinforcement learning

We are interested in an optimal policy. The **critic [value function]** tells us how to improve the **actor [policy]**.

Hidden connections…

Generative adversarial networks

We are interested in an optimal generator. The **discriminator** tells us how to improve the **generator**.

Variational inference

We are interested in an optimal approximate posterior. The **ELBO** tells us how to improve the **approximate posterior**.

Reinforcement learning

We are interested in an optimal policy. The **critic [value function]** tells us how to improve the **actor [policy]**.

First-order optimization

We are interested in an optimal value of a variable. The **gradient of the loss function** tells us how to improve the **value of the variable**.

Hidden connections...

GAN/RL/VI algorithms minimize some loss function, for which the discriminator/value function/ELBO *is* the “gradient”!

GANs minimize the Jensen-Shannon divergence

$$\min_\theta \max_\phi \mathbb{E}_{x\sim p_{\text{data}}}[\log D_\phi(x)] + \mathbb{E}_{\boxed{z\sim\mathcal{N}}}[\log(1 - D_\phi(\boxed{G_\theta(z)}))]$$

$$\min_\theta \max_\phi \mathbb{E}_{x\sim p_{\text{data}}}[\log D_\phi(x)] + \mathbb{E}_{x\sim p_\theta}[\log(1 - D_\phi(x))]$$

GANs minimize the Jensen-Shannon divergence

$$\min_{\theta} \max_{\phi} \mathbb{E}_{x \sim p_{\text{data}}}[\log D_\phi(x)] + \mathbb{E}_{\boxed{z \sim \mathcal{N}}}[\log(1 - D_\phi(\boxed{G_\theta(z)}))]$$

$$\min_{\theta} \boxed{\max_{\phi} \mathbb{E}_{x \sim p_{\text{data}}}[\log D_\phi(x)] + \mathbb{E}_{x \sim p_\theta}[\log(1 - D_\phi(x))]}$$

$$D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_\theta(x)}$$

GANs minimize the Jensen-Shannon divergence

$$\min_\theta \max_\phi \mathbb{E}_{x\sim p_{\text{data}}}[\log D_\phi(x)] + \mathbb{E}_{\boxed{z\sim\mathcal{N}}}[\log(1 - D_\phi(\boxed{G_\theta(z)}))]$$

$$\min_\theta \boxed{\max_\phi \mathbb{E}_{x\sim p_{\text{data}}}[\log D_\phi(x)] + \mathbb{E}_{x\sim p_\theta}[\log(1 - D_\phi(x))]}$$

$$\min_\theta \text{JSD}(p_{\text{data}}, p_\theta)$$

Jensen-Shannon divergence

$$\text{JSD}(p,q) = \tfrac{1}{2}\text{KL}(p \,\|\, \tfrac{1}{2}p + \tfrac{1}{2}q) + \tfrac{1}{2}\text{KL}(q \,\|\, \tfrac{1}{2}p + \tfrac{1}{2}q)$$

GANs, VI, RL all minimize loss functions

$$J_{\text{GAN}}(p) = \text{JSD}(p_{\text{data}}, p)$$

$$J_{\text{VI}}(q) = \text{KL}(q(z) \,||\, p(z|x))$$

$$J_{\text{RL}}(\pi) = -\mathbb{E}_\tau\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]$$

GANs, VI, RL all minimize **probability functionals**

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

Contains *all* probability distributions over elements in X

GANs, VI, RL all minimize **probability functionals**

$$J_{\text{GAN}}(p) = \text{JSD}(p_{\text{data}}, p)$$

In the GAN case: X is the space of images $\mathbb{R}^{64\times 64\times 3}$

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

Contains *all* probability distributions over elements in X

In the GAN case: P(X) is the space of all distributions over images.

GANs, VI, RL all minimize **probability functionals**

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

In the VI case: X is the space of parameters (z)

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

Contains *all* probability distributions over elements in X

In the VI case: P(X) is the space of all distributions over parameters (all possible posteriors).

GANs, VI, RL all minimize **probability functionals**

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

$$\pi(s,a) := \pi(a|s)\,\pi(s)$$

In the RL case: X is the space of states and actions $S \times A$

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

Contains *all* probability distributions over elements in X

In the RL case: P(X) is the space of all distributions over state-action pairs (≈ all possible policies)

GANs, VI, RL all minimize **probability functionals**

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

von Mises influence function

probability functional

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

$$\Big\downarrow \text{gradient } \nabla J(\mu)$$

von Mises influence function

$$\Psi : X \to \mathbb{R}$$

von Mises influence function

The **von Mises influence function**

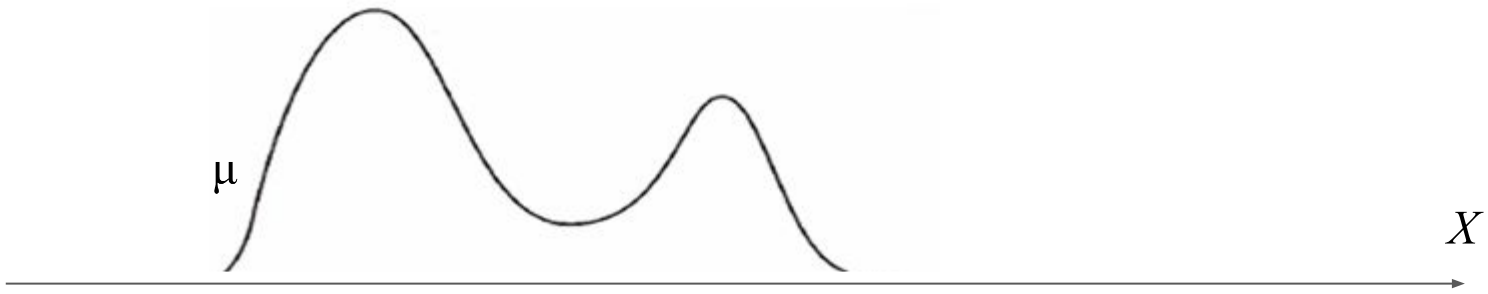
$$\nabla J(\mu) : X \to \mathbb{R}$$

is the function Ψ, unique up to an additive constant, such that for all distributions ν,

$$\mathbb{E}_{x\sim\nu}[\Psi(x)] - \mathbb{E}_{x\sim\mu}[\Psi(x)] = \lim_{\epsilon\to 0} \frac{J((1-\epsilon)\mu + \epsilon\nu) - J(\mu)}{\epsilon}$$

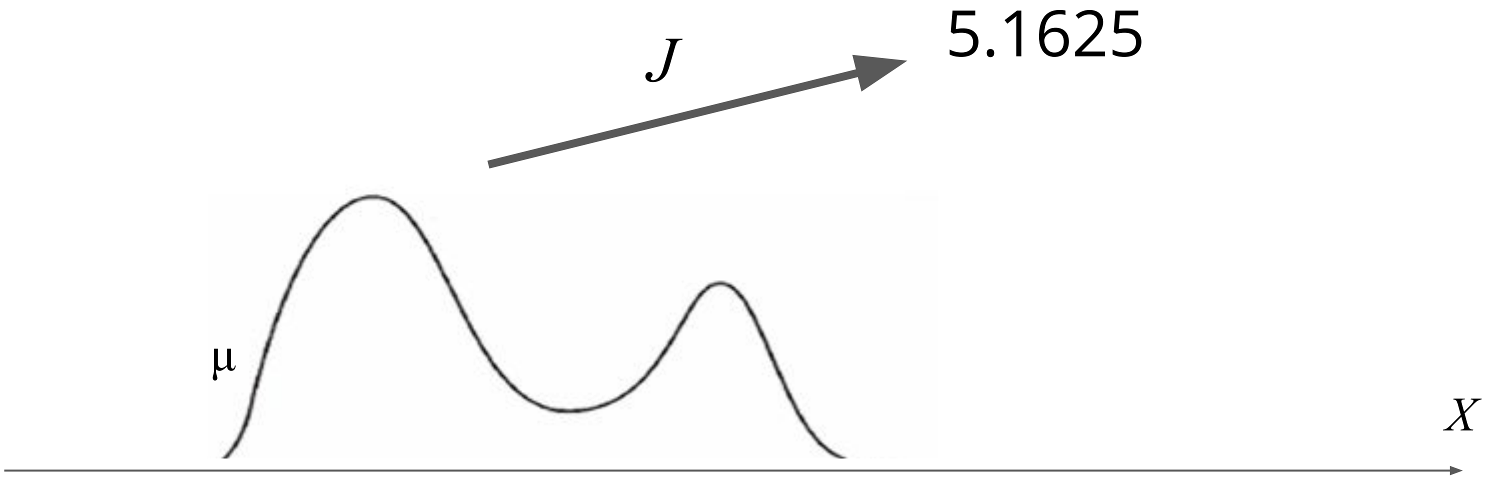
von Mises influence function

Let $J : \mathbf{P}(X) \to \mathbb{R}$ be a probability functional, and let $\mu \in \mathbf{P}(X)$.



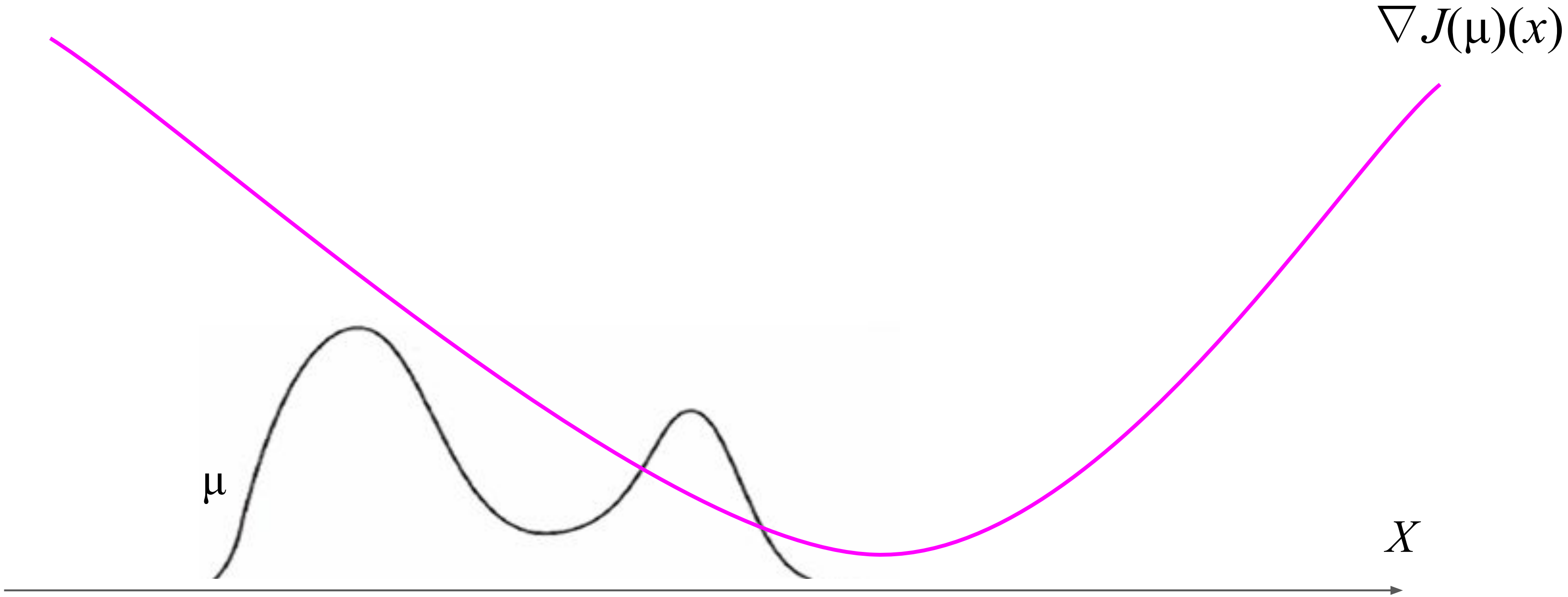
von Mises influence function

Let $J : \mathbf{P}(X) \to \mathbb{R}$ be a probability functional, and let $\mu \in \mathbf{P}(X)$.



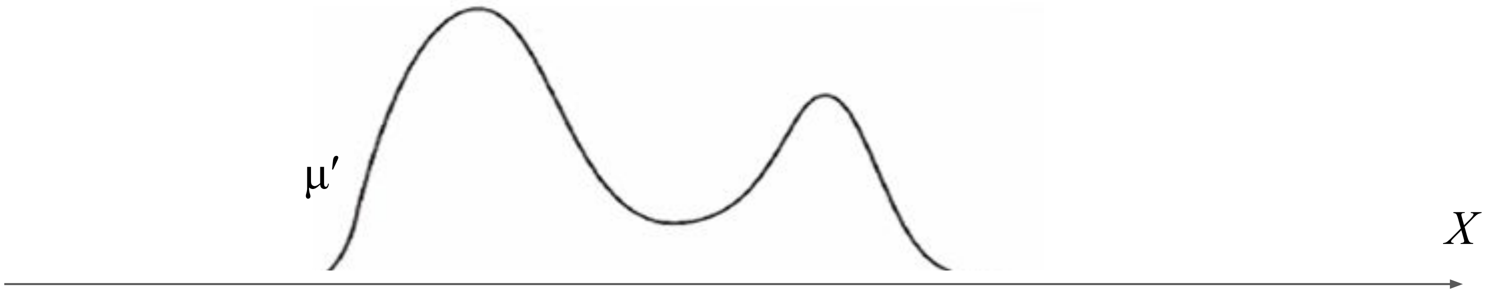
von Mises influence function

Let $J : \mathbf{P}(X) \to \mathbb{R}$ be a probability functional, and let $\mu \in \mathbf{P}(X)$.



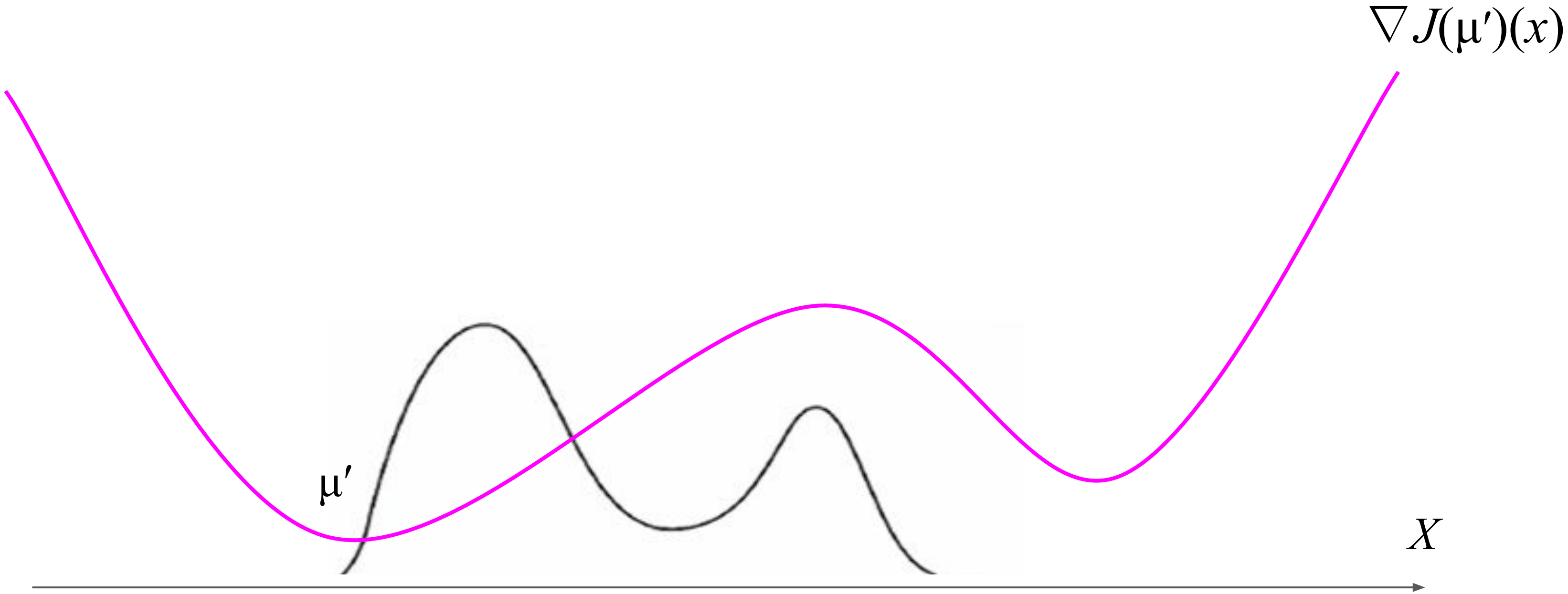
von Mises influence function

Let $J : \mathbf{P}(X) \to \mathbb{R}$ be a probability functional, and let $\mu \in \mathbf{P}(X)$.



von Mises influence function

Let $J : \mathbf{P}(X) \to \mathbb{R}$ be a probability functional, and let $\mu \in \mathbf{P}(X)$.



von Mises influence function

$$\nabla J_{\mathrm{GAN}}(p)(x) = \frac{1}{2}\log\frac{p(x)}{p_{\mathrm{data}}(x) + p(x)}$$

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

$$\nabla J_{\mathrm{RL}}(\pi)(s,a) = -\frac{1}{1-\gamma}\big(Q^{\pi}(s,a) - V^{\pi}(s)\big)$$

von Mises influence function

$$\nabla J_{\mathrm{GAN}}(p)(x) = \frac{1}{2}\log \boxed{\frac{p(x)}{p_{\mathrm{data}}(x) + p(x)}}$$

optimal discriminator

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

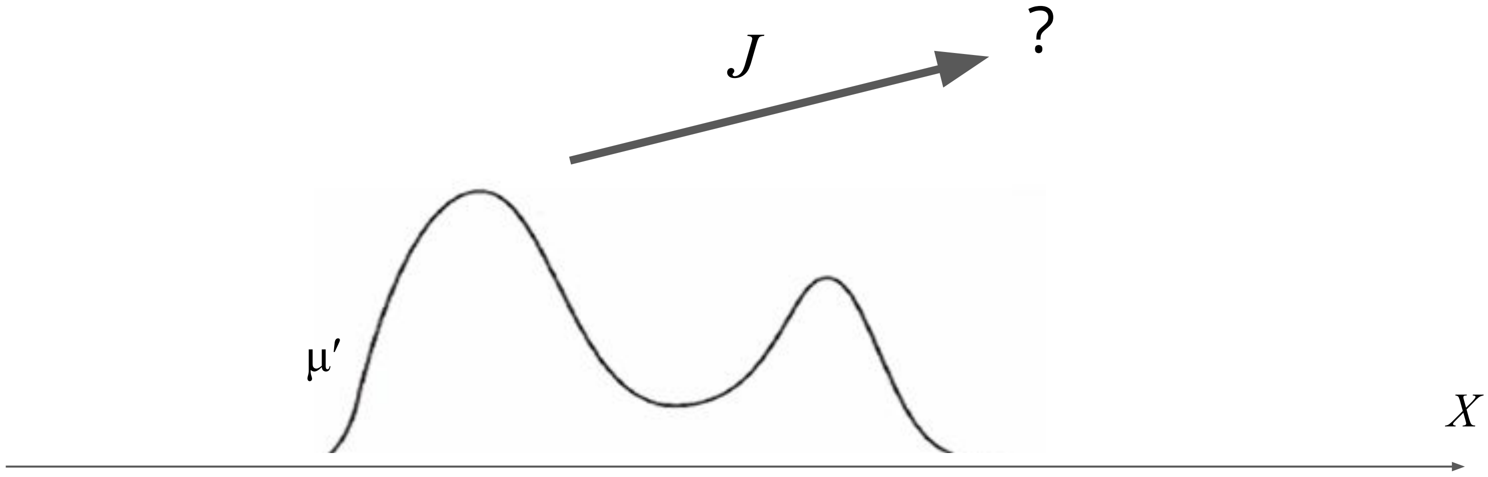
negative ELBO

value functions (advantage)

$$\nabla J_{\mathrm{RL}}(\pi)(s,a) = -\frac{1}{1-\gamma}\left(\boxed{Q^{\pi}(s,a) - V^{\pi}(s)}\right)$$

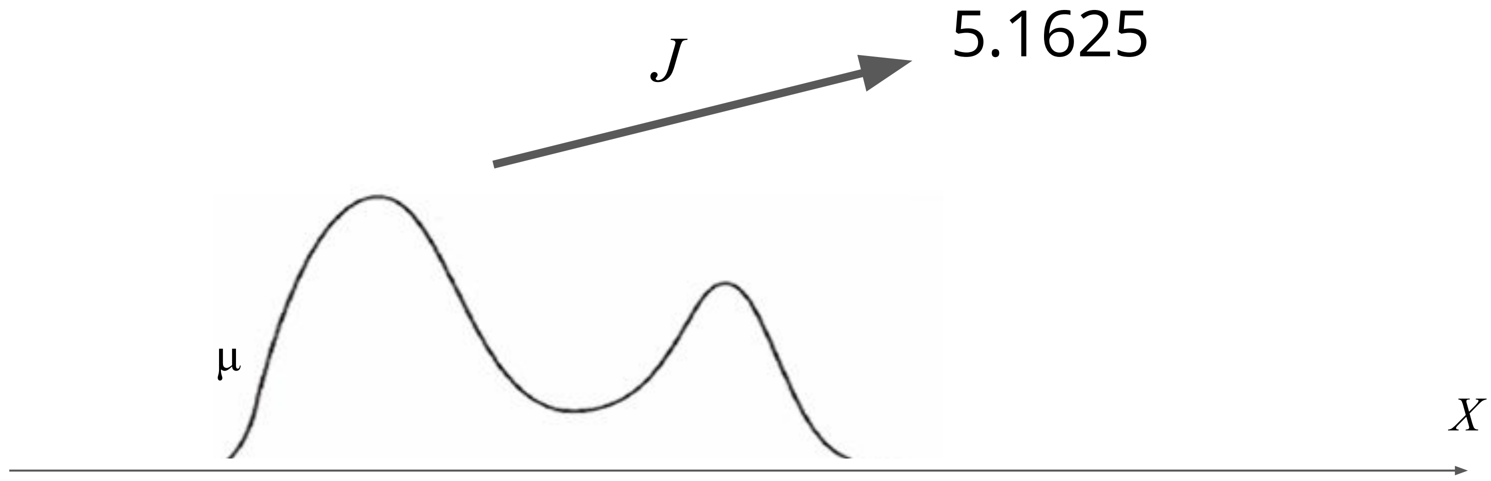
von Mises influence function

Let $J : \mathbf{P}(X) \to \mathbb{R}$ be a probability functional, and let $\mu \in \mathbf{P}(X)$.



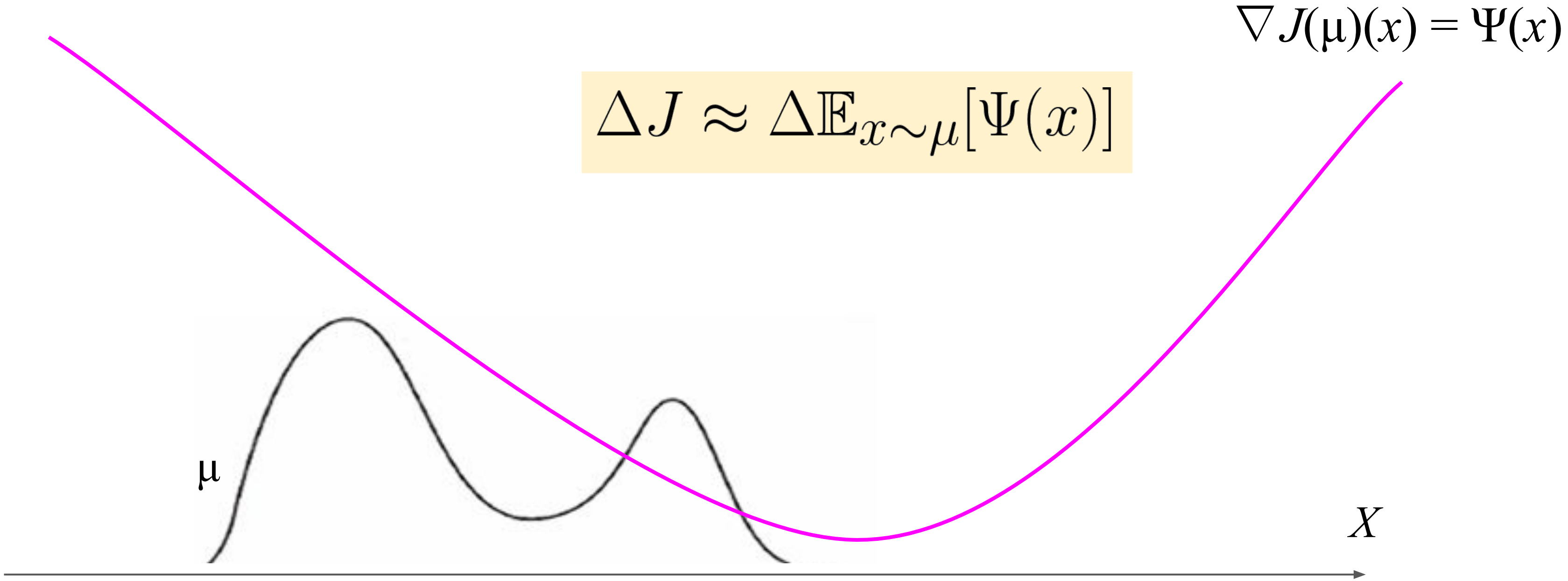
von Mises influence function

Let $J : \mathbf{P}(X) \to \mathbb{R}$ be a probability functional, and let $\mu \in \mathbf{P}(X)$.



von Mises influence function

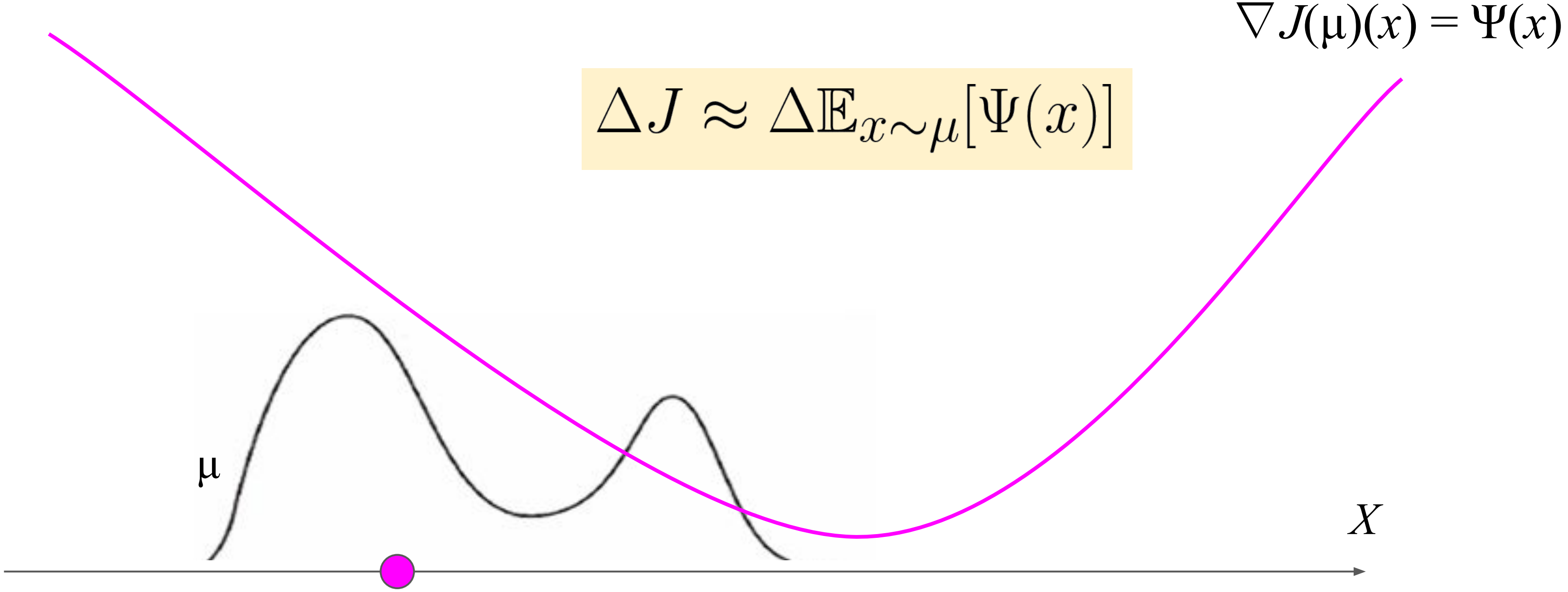
Let $J : \mathbf{P}(X) \to \mathbb{R}$ be a probability functional, and let $\mu \in \mathbf{P}(X)$.



von Mises influence function

Let $J : \mathbf{P}(X) \to \mathbb{R}$ be a probability functional, and let $\mu \in \mathbf{P}(X)$.

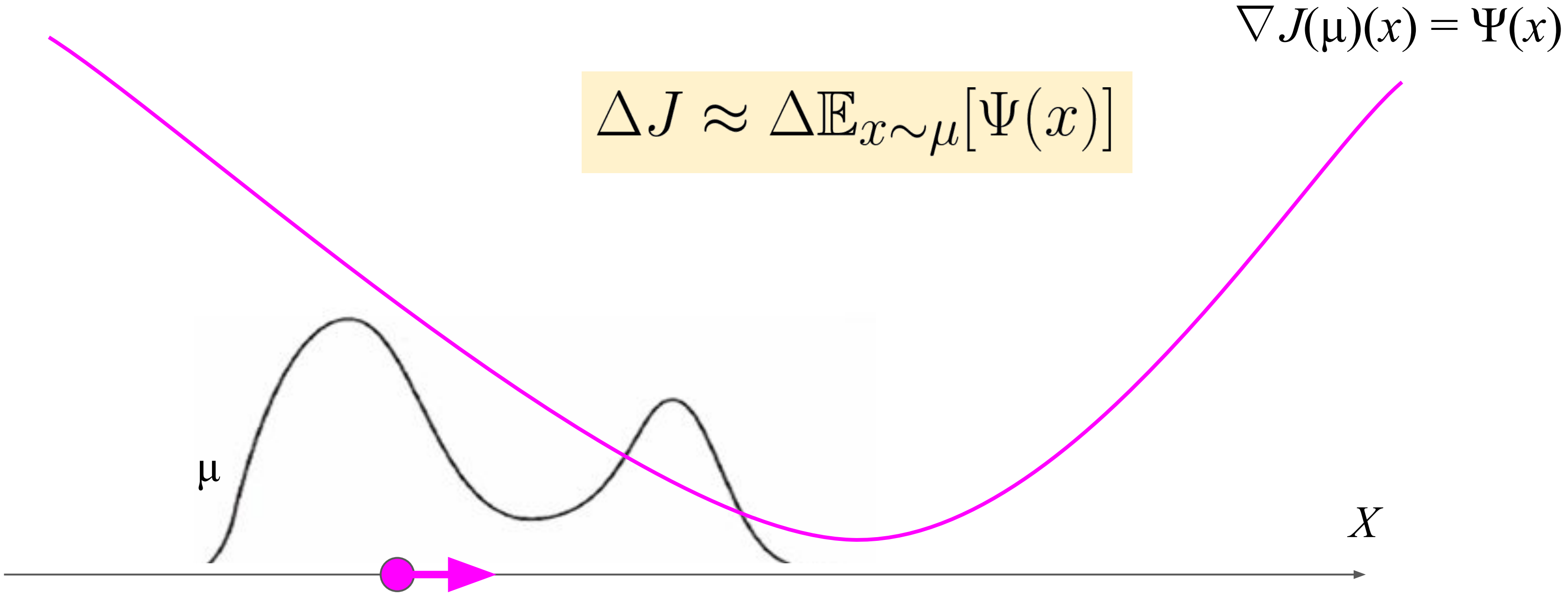
$$\Delta J \approx \Delta \mathbb{E}_{x\sim\mu}[\Psi(x)]$$



von Mises influence function

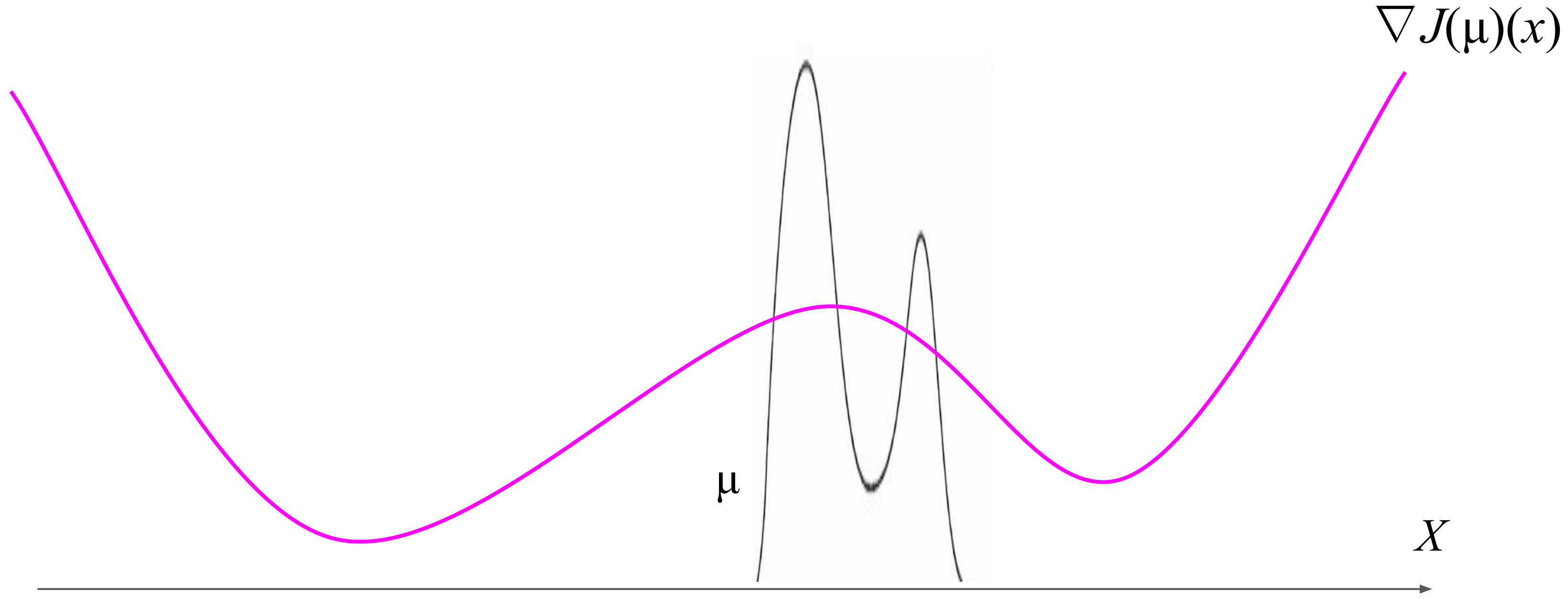
Let $J : \mathbf{P}(X) \to \mathbb{R}$ be a probability functional, and let $\mu \in \mathbf{P}(X)$.

$$\Delta J \approx \Delta \mathbb{E}_{x\sim\mu}[\Psi(x)]$$



von Mises influence function

Let $J : \mathbf{P}(X) \to \mathbb{R}$ be a probability functional, and let $\mu \in \mathbf{P}(X)$.



Theorem 1 (chain rule):

$$\nabla_\theta J(\mu_\theta) = \nabla_\theta \, \mathbb{E}_{x\sim\mu_\theta}[\Psi(x)]$$

where

$$\Psi(x) = \nabla J(\mu_\theta)(x)$$

is the von Mises influence function, treated as **constant** w.r.t. θ.

Theorem 1 (chain rule):

$$\nabla_\theta J(\mu_\theta) = \nabla_\theta \, \mathbb{E}_{x\sim\mu_\theta}[\Psi(x)]$$

where

$$\Psi(x) = \nabla J(\mu_\theta)(x)$$

is the von Mises influence function, treated as **constant** w.r.t. θ.

$$\nabla_\theta J(\mu_\theta) \text{ ``$=$'' } \nabla_\mu J(\mu_\theta) \times \nabla_\theta \, \mu_\theta$$

Theorem 1 (chain rule):

$$\nabla_\theta J(\mu_\theta) = \nabla_\theta \, \mathbb{E}_{x \sim \mu_\theta}[\Psi(x)]$$

where

$$\Psi(x) = \nabla J(\mu_\theta)(x)$$

is the von Mises influence function, treated as **constant** w.r.t. θ.

Probability functional descent

1. Initialize parameters θ arbitrarily
2. Fit a neural network to the von Mises influence function:

$$\hat{\Psi}(x) \approx \nabla J(\mu_\theta)(x)$$

3. Perform the gradient update

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathbb{E}_{x \sim \mu_\theta}[\hat{\Psi}(x)]$$

4. Repeat 2 and 3.

Theorem (chain rule):

$$\nabla_\theta J(\mu_\theta) = \nabla_\theta \mathbb{E}_{x \sim \mu_\theta}[\Psi(x)]$$

Probability functional descent

1. Initialize parameters θ arbitrarily
2. Fit a neural network to the von Mises influence function:

$$\hat{\Psi}(x) \approx \nabla J(\mu_\theta)(x)$$

$$\mathbb{E}_{z\sim\mathcal{N}}[\nabla_\theta \hat{\Psi}(f_\theta(z))]$$

$$\mathbb{E}_{x\sim\mu_\theta}[\hat{\Psi}(x)\nabla_\theta \log \mu_\theta(x)]$$

3. Perform the gradient update

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathbb{E}_{x\sim\mu_\theta}[\hat{\Psi}(x)]$$

4. Repeat 2 and 3.

Theorem (chain rule):

$$\nabla_\theta J(\mu_\theta) = \nabla_\theta \mathbb{E}_{x\sim\mu_\theta}[\Psi(x)]$$

Probability functional descent (GANs)

1. Initialize generator parameters θ arbitrarily
2. Fit a neural network to the von Mises influence function:

$$\hat{\Psi}(x) \approx \frac{1}{2} \log \frac{p_\theta(x)}{p_{\text{data}}(x) + p_\theta(x)}$$

Solve this optimization problem

$$\max_\phi \mathbb{E}_{x \sim p_{\text{data}}}[\log D_\phi(x)] + \mathbb{E}_{x \sim p_\theta}[\log(1 - D_\phi(x))]$$

3. Perform the gradient update

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathbb{E}_{x \sim p_\theta}[\hat{\Psi}(x)]$$

4. Repeat 2 and 3.

Probability functional descent (VI)

1. Initialize approx. posterior parameters θ arbitrarily
2. We can evaluate the von Mises influence function directly:

$$\hat{\Psi}(z) = \log \frac{q_\theta(z)}{p(x|z)\,p(z)}$$

3. Perform the gradient update

$$\theta \leftarrow \theta - \alpha \nabla_\theta \boxed{\mathbb{E}_{z\sim q_\theta(z)}[\hat{\Psi}(z)]}$$

Negative ELBO

4. Repeat 2 and 3.

Probability functional descent (RL)

1. Initialize policy parameters θ arbitrarily
2. Fit a neural network to the von Mises influence function:

$$\hat{\Psi}(s,a) \approx -\frac{1}{1-\gamma}\left(Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)\right)$$

Minimize the Bellman residual

3. Perform the gradient update

$$\theta \leftarrow \theta - \alpha\, \mathbb{E}_{s\sim d^\pi}\left[\nabla_\theta\, \mathbb{E}_{a\sim\pi_\theta(a|s)}[\hat{\Psi}(s,a)]\right]$$

4. Repeat 2 and 3.

$$d^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t p_t^\pi(s)$$

Probability functional descent (RL)

1. Initialize policy parameters θ arbitrarily
2. Fit a neural network to the von Mises influence function:

$$\hat{\Psi}(s,a) \approx -\frac{1}{1-\gamma}\left(Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)\right)$$

Minimize the Bellman residual

3. Perform the gradient update

$$\theta \leftarrow \theta - \alpha\, \mathbb{E}_{s\sim d^\pi}\left[\nabla_\theta\, \mathbb{E}_{a\sim\pi_\theta(a|s)}[\hat{\Psi}(s,a)]\right]$$

4. Repeat 2 and 3.

$$d^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t p_t^\pi(s)$$

$$\nabla J_{\mathrm{RL}}(\pi)(s,a) = -\frac{1}{1-\gamma}\frac{d^\pi(s)}{\pi(s)}\left(Q^\pi(s,a) - V^\pi(s)\right)$$

$$\theta \leftarrow \theta - \alpha\nabla_\theta\, \mathbb{E}_{s,a\sim\pi_\theta(s,a)}[\hat{\Psi}(s,a)]$$

Probability functional descent

1. Initialize parameters θ arbitrarily
2. Fit a neural network to the von Mises influence function:

$$\hat{\Psi}(x) \approx \nabla J(\mu_\theta)(x)$$

3. Perform the gradient update

$$\theta \leftarrow \theta - \alpha \nabla_\theta \, \mathbb{E}_{x \sim \mu_\theta}[\hat{\Psi}(x)]$$

4. Repeat 2 and 3.

Lots of algorithms… **but PFD “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

- REINFORCE
- Deep deterministic policy gradient
- Dual actor-critic
- Soft actor-critic

Probability functional descent

1. Initialize parameters θ arbitrarily

Requires some creativity!

2. Fit a neural network to the von Mises influence function:

$$\hat{\Psi}(x) \approx \nabla J(\mu_\theta)(x)$$

3. Perform the gradient update

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathbb{E}_{x \sim \mu_\theta}[\hat{\Psi}(x)]$$

4. Repeat 2 and 3.

Influence function via convex duality

Suppose J is convex. Then:

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

where

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

is the **convex conjugate** of J.

Probability functional descent (convex function)

1. Initialize parameters θ arbitrarily
2. Fit a neural network to the von Mises influence function:

$$\hat{\Psi}(x) \approx \underset{\varphi \in \mathcal{C}(X)}{\arg\max} \left[\mathbb{E}_{x \sim \mu_\theta}[\varphi(x)] - J^\star(\varphi) \right]$$

3. Perform the gradient update

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathbb{E}_{x \sim \mu_\theta}[\hat{\Psi}(x)]$$

4. Repeat 2 and 3.

We've recovered a minimax, adversarial game!

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

Lots of algorithms... **but PFD "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

- REINFORCE
- Deep deterministic policy gradient
- Dual actor-critic*
- Soft actor-critic

Gradient descent in the space of probability distributions!

Lots of algorithms... **but PFD "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
- REINFORCE
- Deep deterministic policy gradient
- Dual actor-critic*
- Soft actor-critic

Your field!
- Your algorithm!

Gradient descent in the space of probability distributions!

What is gradient descent really?

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

Taylor expansion

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

Generic descent algorithm: choose x such that $\Delta f < 0$.

Gradient descent

$$x = x_0 - \alpha \nabla f(x_0)$$

$$\begin{aligned}\Delta f &\approx \nabla f(x_0) \cdot (-\alpha \nabla f(x_0)) \\ &= -\alpha ||\nabla f(x_0)||^2 \\ &\leq 0\end{aligned}$$

Gradient descent in the space of probabilities

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

Suppose X is a finite set with n elements, so that $P(X)$ is simply a subset of $\mathbf{R}^n$.

Generic descent algorithm: choose $\mathbf{p}$ such that $\Delta J < 0$.

$$\begin{aligned} \Delta J &\approx \nabla J(\mathbf{p}_0) \cdot (\mathbf{p} - \mathbf{p}_0) \\ &= \nabla J(\mathbf{p}_0) \cdot \mathbf{p} - J(\mathbf{p}_0) \cdot \mathbf{p}_0 \\ &= \mathbb{E}_{i \sim \mathbf{p}}[(\nabla J(\mathbf{p}_0))_i] - \mathbb{E}_{i \sim \mathbf{p}_0}[(\nabla J(\mathbf{p}_0))_i] \end{aligned}$$

Gradient descent in the space of probabilities

$$\Delta J \approx \mathbb{E}_{i\sim\mathbf{p}}[(\nabla J(\mathbf{p}_0))_i] - \mathbb{E}_{i\sim\mathbf{p}_0}[(\nabla J(\mathbf{p}_0))_i]$$

Generalize to general sets *X*,
not necessarily discrete!

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

Gradient descent in the space of probabilities

$$\Delta J \approx \mathbb{E}_{i\sim\mathbf{p}}[(\nabla J(\mathbf{p}_0))_i] - \mathbb{E}_{i\sim\mathbf{p}_0}[(\nabla J(\mathbf{p}_0))_i]$$

Generalize to general sets *X*, not necessarily discrete!

The gradient is now a *function*

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

$$\Delta J \approx \mathbb{E}_{x\sim\mu}[\nabla J(\mu_0)(x)] - \mathbb{E}_{x\sim\mu_0}[\boxed{\nabla J(\mu_0)}(x)]$$

Von Mises influence function

The **von Mises influence function**

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

is the function Ψ, unique up to an additive constant, such that for all ν,

$$\mathbb{E}_{x\sim\nu}[\Psi(x)]-\mathbb{E}_{x\sim\mu}[\Psi(x)] = \lim_{\epsilon\to 0} \frac{J((1-\epsilon)\mu+\epsilon\nu)-J(\mu)}{\epsilon}$$

Gradient descent in the space of probabilities

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

Generic descent algorithm: choose μ such that $\Delta J < 0$.

At every update, we'd like to find a distribution μ such that

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

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**

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

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

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

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

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