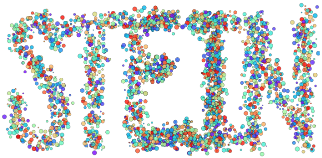
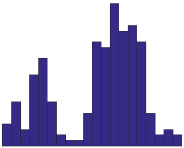
# A Stein Variational Framework for Deep Probabilistic Modeling

**Qiang Liu**
Dartmouth College

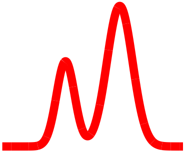


# Machine Learning and Statistics

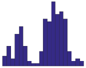
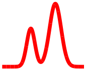
data $\{x_i\}$



models $p(x)$



## Data-Model Discrepancy

$\mathbb{D}($  ,  $)$

data $\{x_i\}_{i=1}^n$ , model $p$

- **Learning (model estimation):** Given $\{x_i\}$, find an optimal $p$:

$$\min_p \mathbb{D}(\{x_i\}, \quad p).$$

- **Inference (or sampling)**: Given $p$, find optimal $\{x_i\}$:

$$\min_{\{x_i\}} \mathbb{D}(\{x_i\}, \quad p).$$

- **Model checking (e.g., goodness of fit test)**: Given both $p$ and $\{x_i\}$, tell if they are consistent:

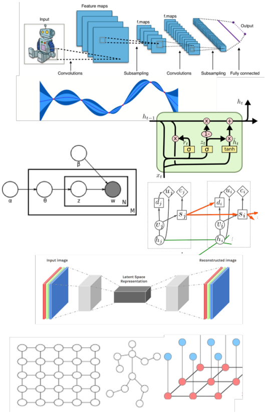
$$\mathbb{D}(\{x_i\}, \quad p) \stackrel{?}{=} 0.$$

## In Reality ...

- Modern machine learning $=$ Complex data $+$ Complex models

Complex data $\{x_i\}$


Complex models $p(x)$



# Unnormalized Distributions

- In practice, many distributions have unnormalized densities:

$$p(x) = \frac{1}{Z}\bar{p}(x), \qquad Z = \int \bar{p}(x)dx.$$

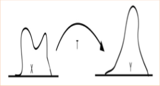
$Z$: normalization constant, critically difficult to calculate!

- Widely appear in
  - Bayesian inference,
  - Probabilistic graphical models,
  - Deep energy-based models,
  - Log-linear models,
  - and many more ...
- Highly difficult to learn, sample and evaluate.

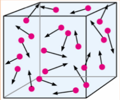
- Scalable computational algorithms are the key.
- Can benefit from integrating tools in different areas ...

**Stein**'s Method, probability theory

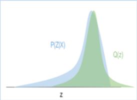
Optimal transport, gradient flow, etc



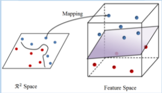

Numerical PDE, Interacting particle systems, etc



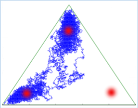
**Variational** Inference




Kernel Method (RKHS)




Monte Carlo



**Deep** Learning

Compute vision

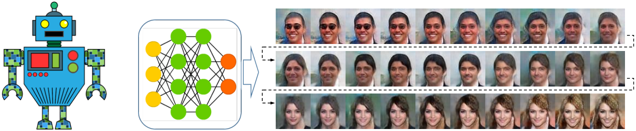
Reinforcement Learning

# This Talk

- This talk focuses on the inference (sampling) problem:

  **Given $p$, find $\{x_i\}$ to approximation $p$.**

- **Two applications**:
  - Policy optimization in reinforcement learning.
  - Training neural networks to generate natural images.



## Classical Methods for Inference (Sampling)

- **Sampling: Given $p$, find $\{x_i\}$ to approximation $p$.**

- Monte Carlo / Markov chain Monte Carlo (MCMC):
  - Simulate random points.
  - Asymptotically "correct", but slow.

- Variational inference:
  - Approximate $p$ with a simpler $q_\theta$ (e.g., Gaussian): $\min_{\theta \in \Theta} \mathrm{KL}(q_\theta \mid\mid p)$.
  - Need parametric assumption: fast, but "wrong".

- Optimization (maximum a posteriori (MAP)):
  - Find a single point approximation: $x^* = \arg\max p(x)$.
  - Faster, local optima, no uncertainty assessment.

# Stein Variational Gradient Descent (SVGD) [Liu Wang, 2016]

- Directly minimize the Kullback-Leibler (KL) divergence between $\{x_i\}$ and $p$:

$$\min_{\{x_i\}} \mathrm{KL}(\{x_i\}, \quad p)$$

- An ill-posed problem? $\mathrm{KL}(\{x_i\}, \quad p) = \infty$.

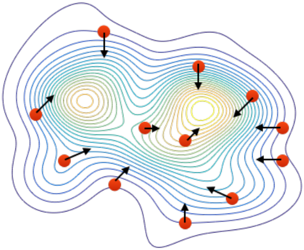
- Turns out to be doable, with some new insights.

## Stein Variational Gradient Descent (SVGD) [Liu Wang, 2016]

Idea: Iteratively move $\{x_i\}_{i=1}^n$ towards the target $p$ by updates of form

$$x_i' \leftarrow x_i + \epsilon \phi(x_i),$$

$\epsilon$: step-size. $\phi$: a perturbation direction chosen to maximally decrease the KL divergence with $p$:



$$\phi = \arg\max_{\phi \in \mathcal{F}} \Big\{ \underbrace{\mathrm{KL}(q \,||\, p)}_{\text{old particles}} - \underbrace{\mathrm{KL}(q_{[\epsilon\phi]} \,||\, p)}_{\text{updated particles}} \Big\}$$

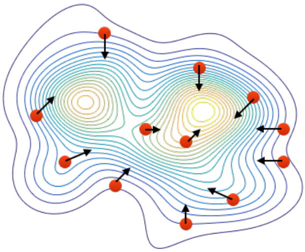
where $q_{[\epsilon\phi]}$ is the density of $x' = x + \epsilon\phi(x)$ when the density of $x$ is $q$.

# Stein Variational Gradient Descent (SVGD) [Liu Wang, 2016]

Idea: Iteratively move $\{x_i\}_{i=1}^n$ towards the target $p$ by updates of form

$$x_i' \leftarrow x_i + \epsilon \phi(x_i),$$

$\epsilon$: step-size. $\phi$: a perturbation direction chosen to maximally decrease the KL divergence with $p$:



$$\phi = \arg\max_{\phi \in \mathcal{F}} \left\{ \mathrm{KL}(q \,||\, p) - \mathrm{KL}(q_{[\epsilon\phi]} \,||\, p) \right\}$$

$$\approx \arg\max_{\phi \in \mathcal{F}} \left\{ -\frac{\partial}{\partial \epsilon} \mathrm{KL}(q_{[\epsilon\phi]} \,||\, p) \Big|_{\epsilon=0} \right\},$$ //when step size $\epsilon$ is small

where $q_{[\epsilon\phi]}$ is the density of $x' = x + \epsilon\phi(x)$ when the density of $x$ is $q$.

## Stein Variational Gradient Descent (SVGD) [Liu Wang, 2016]

Key: the objective is a *simple, linear functional* of $\phi$:

$$-\frac{\partial}{\partial \epsilon} \mathrm{KL}(q_{[\epsilon\phi]} \mid\mid p)\big|_{\epsilon=0} = \mathbb{E}_{x\sim q}[\mathcal{T}_p \phi(x)].$$

where $\mathcal{T}_p$ is a linear operator called **Stein operator** related to $p$:

$$\mathcal{T}_p \phi(x) \overset{def}{=} \langle \nabla_x \log p(x), \ \ \phi(x) \rangle + \nabla_x \cdot \phi(x).^1$$

[1] $\nabla_x \cdot \boldsymbol{\phi} = \sum_i \partial_{x_i} \boldsymbol{\phi}$

## Stein Variational Gradient Descent (SVGD) [Liu Wang, 2016]

Key: the objective is a *simple, linear functional* of $\phi$:

$$-\frac{\partial}{\partial\epsilon}\mathrm{KL}(q_{[\epsilon\phi]} \mid\mid p)\big|_{\epsilon=0} = \mathbb{E}_{x\sim q}[\mathcal{T}_p\phi(x)].$$

where $\mathcal{T}_p$ is a linear operator called **Stein operator** related to $p$:

$$\mathcal{T}_p\phi(x) \overset{def}{=} \langle \underbrace{\nabla_x \log p(x)}_{\text{score function}}, \quad \phi(x)\rangle + \nabla_x \cdot \phi(x).^1$$

Score function $\nabla_x \log p(x) = \frac{\nabla_x p(x)}{p(x)}$, independent of the normalization constant $Z$!

[1] $\nabla_x \cdot \phi = \sum_i \partial_{x_i}\phi$

## Stein Variational Gradient Descent (SVGD) [Liu Wang, 2016]

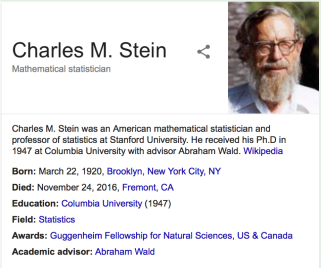
Key: the objective is a *simple, linear functional* of $\phi$:

$$-\frac{\partial}{\partial \epsilon}\mathrm{KL}(q_{[\epsilon\phi]} \mid\mid p)\big|_{\epsilon=0} = \mathbb{E}_{x\sim q}[\mathcal{T}_p \phi(x)].$$

where $\mathcal{T}_p$ is a linear operator called **Stein operator** related to $p$:

$$\mathcal{T}_p\phi(x) \overset{def}{=} \langle \nabla_x \log p(x), \ \ \phi(x)\rangle + \nabla_x \cdot \phi(x).^1$$

- **Stein's method**: a set of theoretical techniques for proving fundamental approximation bounds and limits (such as central limit theorem) in probability theory.
- A large body of theoretical work. Known to be "remarkably powerful".




[1] $\nabla_x \cdot \phi = \sum_i \partial_{x_i}\phi$

## Stein Discrepancy

The optimization is equivalent to

$$\mathbb{D}(q \mid\mid p) \overset{def}{=} \max_{\phi \in \mathcal{F}} \left\{ \mathbb{E}_q[\mathcal{T}_p \phi] \right\}$$

where $\mathbb{D}(q \mid\mid p)$ is called Stein discrepancy: $\mathbb{D}(q \mid\mid p) = 0$ iff $q = p$ if $\mathcal{F}$ is "large" enough.

## Stein Discrepancy

The optimization is equivalent to

$$\mathbb{D}(q \mid\mid p) \stackrel{def}{=} \max_{\phi \in \mathcal{F}} \left\{ \mathbb{E}_q[\mathcal{T}_p \phi] \right\}$$

where $\mathbb{D}(q \mid\mid p)$ is called Stein discrepancy: $\mathbb{D}(q \mid\mid p) = 0$ iff $q = p$ if $\mathcal{F}$ is "large" enough.

- The choice of $\mathcal{F}$ is critical.
- Traditional Stein discrepancy is not computable: casts challenging infinite dimensional functional optimizations.
  - Imposing constraints only on finite numbers of points [Gorham, Mackey 15; Gorham et al. 16]
  - Obtaining closed form solution using reproducing kernel Hilbert space [Liu et al. 16; Chwialkowski et al. 16; Oates et al. 14; Gorham, Mackey 17]

## Stein Discrepancy

The optimization is equivalent to

$$\mathbb{D}(q \mid\mid p) \stackrel{def}{=} \max_{\phi \in \mathcal{F}} \left\{ \mathbb{E}_q[\mathcal{T}_p \phi] \right\}$$

where $\mathbb{D}(q \mid\mid p)$ is called Stein discrepancy: $\mathbb{D}(q \mid\mid p) = 0$ iff $q = p$ if $\mathcal{F}$ is "large" enough.

- The choice of $\mathcal{F}$ is critical.
- Traditional Stein discrepancy is not computable: casts challenging infinite dimensional functional optimizations.
  - Imposing constraints only on finite numbers of points [Gorham, Mackey 15; Gorham et al. 16]
  - Obtaining closed form solution using reproducing kernel Hilbert space [Liu et al. 16; Chwialkowski et al. 16; Oates et al. 14; Gorham, Mackey 17]

## Kernel Stein Discrepancy [Liu et al. 16; Chwialkowski et al. 16]

- **Computable Stein discrepancy using kernel**:
  - Take $\mathcal{F}$ to be the unit ball of any reproducing kernel Hilbert space (RKHS) $\mathcal{H}$, with positive kernel $k(x, x')$:

$$\mathbb{D}(q \mid\mid p) \stackrel{def}{=} \max_{\phi \in \mathcal{H}} \left\{ \mathbb{E}_q[\mathcal{T}_p \phi] \quad s.t. \quad ||\phi||_{\mathcal{H}} \leq 1 \right\}$$

  - Closed-form solution:

$$\begin{aligned} \phi^*(x) &\propto \mathbb{E}_{x \sim q}[\mathcal{T}_p k(x, \cdot)] \\ &= \mathbb{E}_{x \sim q}[\nabla_x \log p(x) k(x, \cdot) + \nabla k(x, \cdot)] \end{aligned}$$

  - Kernel Stein Discrepancy:

$$\mathbb{D}(q, \ p)^2 = \mathbb{E}_{x, x' \sim q}[\mathcal{T}_p^x \mathcal{T}_p^{x'} k(x, x')]$$

    - $\mathcal{T}_p^x$, $\mathcal{T}_p^{x'}$: Stein operator w.r.t. variable $x$, $x'$.

## Kernel Stein Discrepancy [Liu et al. 16; Chwialkowski et al. 16]

- **Computable Stein discrepancy using kernel**:
  - Take $\mathcal{F}$ to be the unit ball of any reproducing kernel Hilbert space (RKHS) $\mathcal{H}$, with positive kernel $k(x, x')$:

$$\mathbb{D}(q \mid\mid p) \overset{def}{=} \max_{\phi \in \mathcal{H}} \left\{ \mathbb{E}_q[\mathcal{T}_p \phi] \quad s.t. \quad ||\phi||_{\mathcal{H}} \leq 1 \right\}$$

  - Closed-form solution:

$$\phi^*(x) \propto \mathbb{E}_{x \sim q}[\mathcal{T}_p k(x, \cdot)]$$
$$= \mathbb{E}_{x \sim q}[\nabla_x \log p(x) k(x, \cdot) + \nabla k(x, \cdot)]$$

  - Kernel Stein Discrepancy:

$$\mathbb{D}(q, \ p)^2 = \mathbb{E}_{x, x' \sim q}[\mathcal{T}_p^x \mathcal{T}_p^{x'} k(x, x')]$$

- $\mathcal{T}_p^x$, $\mathcal{T}_p^{x'}$: Stein operator w.r.t. variable $x$, $x'$.

## Kernel Stein Discrepancy

Kernel Stein discrepancy provides a computational tool for comparing samples $\{x_i\}$ (from unknown $q$) with unnormalized models $p$:

$$\mathbb{D}(\{x_i\},\ p)^2 \stackrel{def}{=} \frac{1}{n^2} \sum_{ij} \mathcal{T}_p^x \mathcal{T}_p^{x'} k(x_i, x_j).$$

**Applications:**

- *Goodness-of-fit test* for unnormalized distributions [Liu et al. 16; Chwialkowski et al. 16].
- *Black-box importance sampling* [Liu, Lee. 16]: importance weights for samples from unknown distributions by minimizing Stein discrepancy, with super-efficient convergence rates.

- **Evaluation**

$$\mathbb{D}(\{x_i\},\ p) \stackrel{?}{=} 0.$$

## Stein Variational Gradient Descent

SVGD: Approximating $\mathbb{E}_{x\sim q}[\cdot]$ with empirical averaging $\hat{\mathbb{E}}_{x\sim\{x_i\}_{i=1}^n}[\cdot]$ over the current points:

$$x_i \leftarrow x_i + \epsilon\hat{\mathbb{E}}_{x\sim\{x_i\}_{i=1}^n}[\nabla_x logp(x)k(x,x_i) + \nabla_x k(x,x_i)], \quad \forall i = 1,\ldots,n.$$

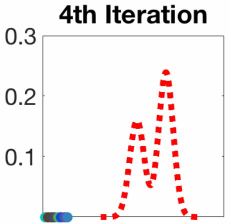
- Iteratively move particles $\{x_i\}$ to fit $p$.

SVGD: iteratively update $\{x_i\}$ until convergence:

$$x_i \leftarrow x_i + \epsilon \hat{\mathbb{E}}_{x \sim \{x_i\}_{i=1}^n} [ \underbrace{\nabla_x \log p(x) k(x, x_i)}_{\text{weighted sum of gradient}} + \underbrace{\nabla_x k(x, x_i)}_{\text{repulsive force}} ], \quad \forall i = 1, \ldots, n.$$

Two terms:

- $\nabla_x \log p(x)$: moves the particles $\{x_i\}$ towards high probability regions of $p(x)$.
- Nearby particles share gradient with weighted sum.
- $\nabla_x k(x, x')$: enforces diversity in $\{x_i\}$ (otherwise all $x_i$ collapse to modes of $p(x)$).



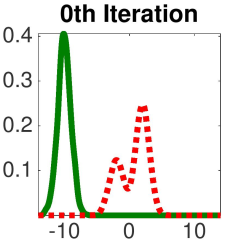
## Stein Variational Gradient Descent

SVGD: iteratively update $\{x_i\}$ until convergence:

$$x_i \leftarrow x_i + \epsilon \hat{\mathbb{E}}_{x \sim \{x_i\}_{i=1}^n} [ \underbrace{\nabla_x logp(x) k(x, x_i)}_{\text{weighted sum of gradient}} + \underbrace{\nabla_x k(x, x_i)}_{\text{repulsive force}} ], \quad \forall i = 1, \ldots, n.$$

Two terms:

- $\nabla_x logp(x)$: moves the particles $\{x_i\}$ towards high probability regions of $p(x)$.
- Nearby particles share gradient with weighted sum.
- $\nabla_x k(x, x')$: enforces diversity in $\{x_i\}$ (otherwise all $x_i$ collapse to modes of $p(x)$).



# SVGD vs. MAP and Monte Carlo

$$x_i \leftarrow x_i + \epsilon \hat{\mathbb{E}}_{x \sim \{x_i\}_{i=1}^n} [\underbrace{\nabla_x logp(x)}_{\text{gradient}} k(x, x_i) + \underbrace{\nabla_x k(x, x_i)}_{\text{repulsive force}}], \quad \forall i = 1, \ldots, n.$$

- When using a single particle ($n = 1$), SVGD reduces to standard gradient ascent for $\max_x \log p(x)$ (i.e., maximum a posteriori (MAP)):

$$x \leftarrow x + \epsilon \nabla_x \log p(x).$$

- MAP (SVGD with $n = 1$): already performs well in many practical cases.
- Typical Monte Carlo / MCMC: perform worse when $n = 1$.

# SVGD as Gradient Flow of KL Divergence [Liu 2016, arXiv:1704.07520]

The empirical measures of the particles weakly converge to the solution of a nonlinear Fokker-Planck equation, that is a gradient flow of KL divergence:

$$\frac{\partial}{\partial t} q_t = -\mathrm{grad}_{\mathcal{H}} \mathrm{KL}(q_t \mid\mid p),$$

which decreases KL divergence monotonically

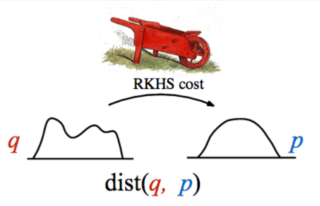
$$\frac{\mathrm{d}}{\mathrm{d}t} \mathrm{KL}(q_t \mid\mid p) = -\mathbb{D}(q_t,\ p)^2.$$

# SVGD as Gradient Flow of KL Divergence [Liu 2016, arXiv:1704.07520]

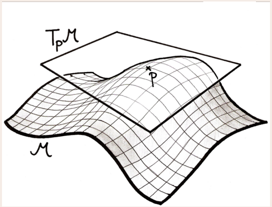
The empirical measures of the particles weakly converge to the solution of a nonlinear Fokker-Planck equation, that is a gradient flow of KL divergence:

$$\frac{\partial}{\partial t} q_t(x) = -\mathrm{grad}_{\mathcal{H}} \mathrm{KL}(q_t \mid\mid p),$$

$\mathrm{grad}_{\mathcal{H}} \mathrm{KL}(q \mid\mid p)$ is a functional gradient defined w.r.t. a new notion of distance between distributions.

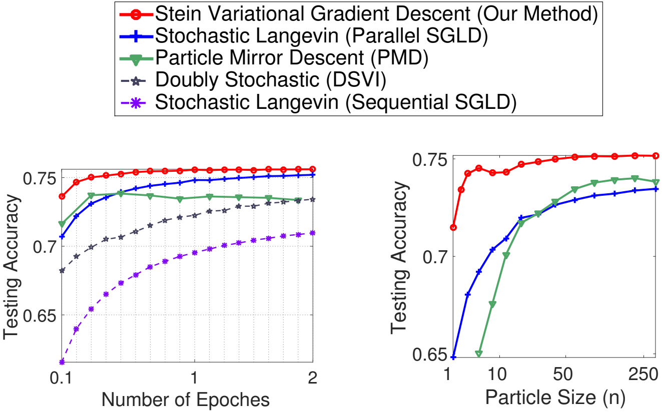



The minimum cost of transporting the mass of $q$ to $p$.




A new geometry structure on the space of distributions.

# Bayesian Logistic Regression




(a) Results with particle size $n = 100$ (b) Results at the 3000th iteration

## Bayesian Neural Network

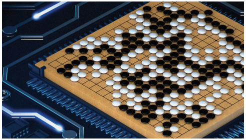
- Test Bayesian neural nets on benchmark datasets.
- Used 20 particles.
- Compared with probabilistic back propagation (PBP) [Hernandez-Lobato et al. 2015]

| | Avg. Test RMSE | | Avg. Test LL | | Avg. Time (Secs) | |
|---|---|---|---|---|---|---|
| **Dataset** | **PBP** | **Our Method** | **PBP** | **Our Method** | **PBP** | **Ours** |
| Boston | $2.977 \pm 0.093$ | $\mathbf{2.957 \pm 0.099}$ | $-2.579 \pm 0.052$ | $\mathbf{-2.504 \pm 0.029}$ | 18 | **16** |
| Concrete | $5.506 \pm 0.103$ | $\mathbf{5.324 \pm 0.104}$ | $-3.137 \pm 0.021$ | $\mathbf{-3.082 \pm 0.018}$ | 33 | **24** |
| Energy | $1.734 \pm 0.051$ | $\mathbf{1.374 \pm 0.045}$ | $-1.981 \pm 0.028$ | $\mathbf{-1.767 \pm 0.024}$ | 25 | **21** |
| Kin8nm | $0.098 \pm 0.001$ | $\mathbf{0.090 \pm 0.001}$ | $0.901 \pm 0.010$ | $\mathbf{0.984 \pm 0.008}$ | 118 | **41** |
| Naval | $0.006 \pm 0.000$ | $\mathbf{0.004 \pm 0.000}$ | $3.735 \pm 0.004$ | $\mathbf{4.089 \pm 0.012}$ | 173 | **49** |
| Combined | $4.052 \pm 0.031$ | $\mathbf{4.033 \pm 0.033}$ | $-2.819 \pm 0.008$ | $\mathbf{-2.815 \pm 0.008}$ | 136 | **51** |
| Protein | $4.623 \pm 0.009$ | $\mathbf{4.606 \pm 0.013}$ | $-2.950 \pm 0.002$ | $\mathbf{-2.947 \pm 0.003}$ | 682 | **68** |
| Wine | $0.614 \pm 0.008$ | $\mathbf{0.609 \pm 0.010}$ | $-0.931 \pm 0.014$ | $\mathbf{-0.925 \pm 0.014}$ | 26 | **22** |
| Yacht | $\mathbf{0.778 \pm 0.042}$ | $0.864 \pm 0.052$ | $\mathbf{-1.211 \pm 0.044}$ | $-1.225 \pm 0.042$ | 25 | 25 |
| Year | $8.733 \pm \mathrm{NA}$ | $\mathbf{8.684 \pm NA}$ | $-3.586 \pm \mathrm{NA}$ | $\mathbf{-3.580 \pm NA}$ | 7777 | **684** |

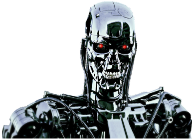
# SVGD as a Search Heuristic

- Particles collaborate to explore large space.
- Can be used to solve challenging non-convex optimization problems.

- **Application: Policy optimization in deep reinforcement learning.**

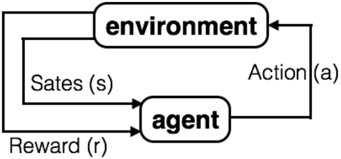


Breakout and Space Invaders, 2 of the 49 Atari games used in the paper



## A Very Quick Intro to Reinforcement Learning

- Agents take actions $a$ based on observed states $s$, and receive reward $r$.




- Policy $\pi_\theta(a|s)$, parameterized by $\theta$.
- Goal: find optimal policy $\pi_\theta(a|s)$ to maximize the expected reward:

$$\max_\theta J(\theta) = \mathbb{E}[r(s,a) \mid \pi_\theta].$$

- Viewed as a black-box optimization.

## Model-Free Policy Gradient

Model-free policy gradient methods:

- Estimate the gradient (without knowing the transition and reward model), and perform gradient descent:

$$\theta \leftarrow \theta + \epsilon \nabla_\theta J(\theta).$$

- Different methods for gradient estimation:
  - Finite difference methods.
  - Likelihood ratio methods: REINFORCE, etc.
  - Actor-critic methods: Advantage Actor-Critic (A2C), etc.

# Model-Free Policy Gradient

- Advantages:
  - Better convergence, work for high dimensional, continuous control tasks.
  - Impressive results on Atari games, vision-based navigation, etc.

- Challenges:
  - Converge to local optima.
  - High variance in gradient estimation.

# Stein Variational Policy Gradient [Liu et al. 17, arXiv:1704.02399]

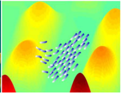
- Stein variational policy gradient: find a group of $\{\theta_i\}$ by

$$\theta_i \leftarrow \theta_i + \frac{\epsilon}{n}\sum_{j=1}^{n}[\underbrace{\nabla_{\theta_j} J(\theta_j) k(\theta_j, \theta_i)}_{\text{gradient sharing}} \; + \; \alpha \underbrace{\nabla_{\theta_j} k(\theta_j, \theta_i)}_{\text{repulsive force}}]$$

- Similar to collective behaviors in swarm intelligence.







## Stein Variational Policy Gradient [Liu et al. 17, arXiv:1704.02399]

- Stein variational policy gradient: find a group of $\{\theta_i\}$ by

$$\theta_i \leftarrow \theta_i + \frac{\epsilon}{n}\sum_{j=1}^{n}[\underbrace{\nabla_{\theta_j} J(\theta_j)k(\theta_j,\theta_i)}_{\text{gradient sharing}} + \alpha \underbrace{\nabla_{\theta_j} k(\theta_j,\theta_i)}_{\text{repulsive force}}]$$

- Can be viewed as sampling $\{\theta_i\}$ from a Boltzmann distribution:

$$p(\theta) \propto \exp(\frac{1}{\alpha} J(\theta))$$

$\alpha$ : temperature parameter.

## Stein Variational Policy Gradient [Liu et al. 17, arXiv:1704.02399]

- Stein variational policy gradient: find a group of $\{\theta_i\}$ by

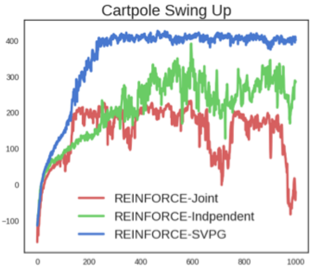
$$\theta_i \leftarrow \theta_i + \frac{\epsilon}{n}\sum_{j=1}^{n}[\underbrace{\nabla_{\theta_j} J(\theta_j) k(\theta_j, \theta_i)}_{\text{gradient sharing}} + \alpha \underbrace{\nabla_{\theta_j} k(\theta_j, \theta_i)}_{\text{repulsive force}}]$$

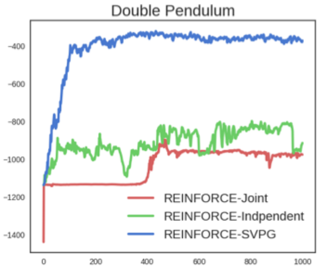
- Can be viewed as sampling $\{\theta_i\}$ from a Boltzmann distribution:

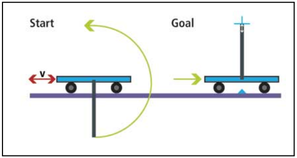
$$p(\theta) \propto \exp(\frac{1}{\alpha} J(\theta)) = \arg\max_q \{\mathbb{E}_q[J(\theta)] \underbrace{+ \alpha H(q)}_{\substack{\text{entropy regularization} \\ \text{encourage exploration}}}\}.$$

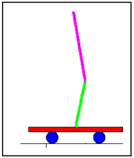
$\alpha$ : temperature parameter. $H(q)$: entropy.

- REINFORCE-SVPG: Stein variational gradient ($n = 16$ agents).
- REINFORCE-Independent: $n$ independent gradient descent agents.
- REINFORCE-Joint: a single agent, using $n$ times as many data per iteration.

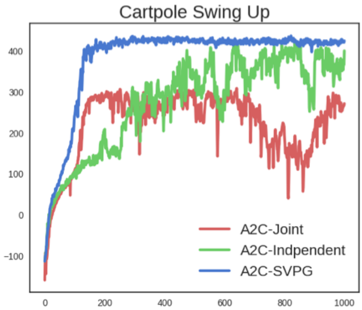


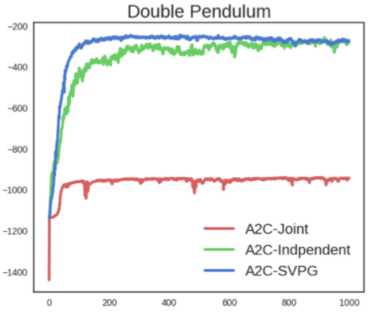



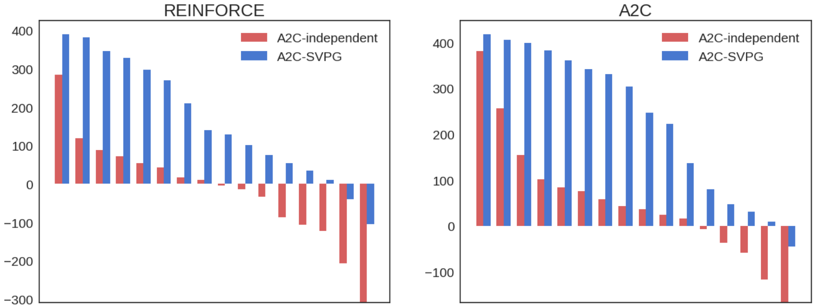



- A2C-SVPG: Stein variational gradient ($n = 16$ agents).
- A2C-Independent: $n$ independent gradient descent agents.
- A2C-Joint: a single agent, using $n$ times as many data per iteration.

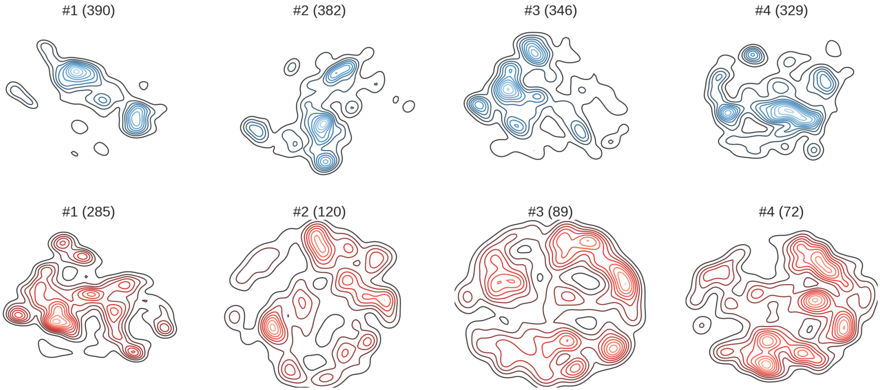


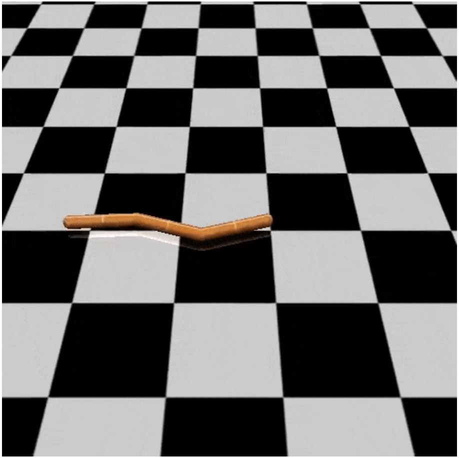
- Average returns of the policies given by SVGD (blue) and independent A2C (red), for Cartpole Swing Up.



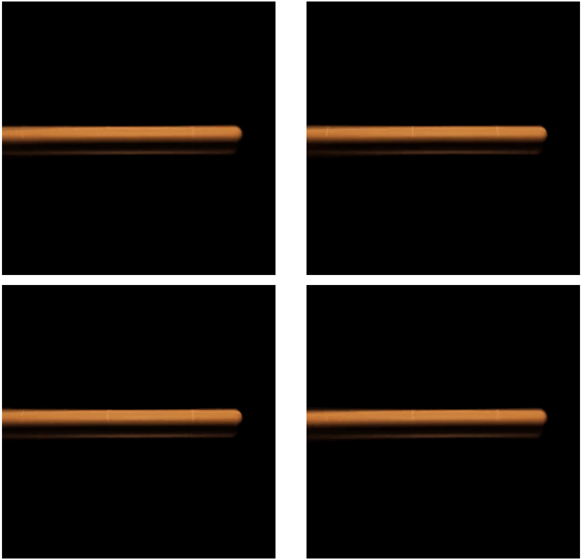
State visitation density of the top 4 policies given by SVGD (upper) and independent REINFORCE (lower), for Cartpole Swing Up.



# Swimmer

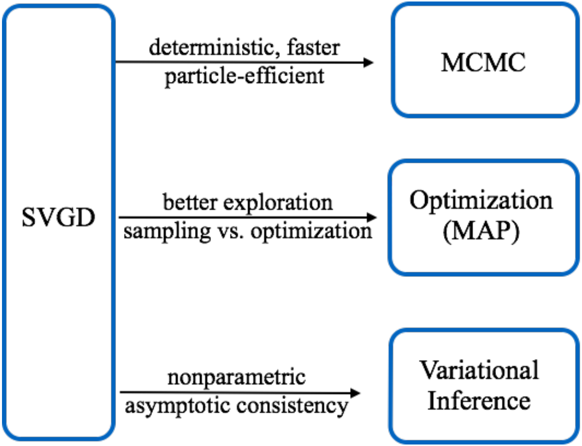


# Top Four Policies by SVPG



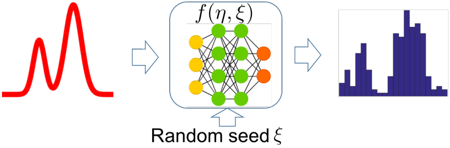
# Stein Variational Gradient Descent

- SVGD: a simple, efficient algorithm for sampling and non-convex optimization.



# Amortized SVGD: Learning to Sample

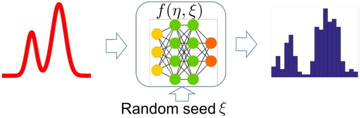
- SVGD is designed for sampling **individual distributions.**
- What if we need to solve **many similar inference problems** repeatedly?
  - Posterior inference for different users, images, documents, etc.
  - sampling as inner loops of all other algorithms.
- We should not solve each problem from scratch.
- **Amortized SVGD**: train feedforward neural networks to learn to draw samples by mimicking the SVGD dynamics.



## Learning to Sample

**Problem formulation:**

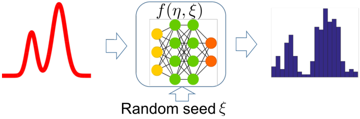
- Given $p$ and a neural net $f(\eta, \xi)$ with parameter $\eta$ and random input $\xi$.
- Find $\eta$ such that the random output $x = f(\eta, \xi)$ approximates distribution $p$.



## Learning to Sample

**Problem formulation:**

- Given $p$ and a neural net $f(\eta, \xi)$ with parameter $\eta$ and random input $\xi$.
- Find $\eta$ such that the random output $x = f(\eta, \xi)$ approximates distribution $p$.

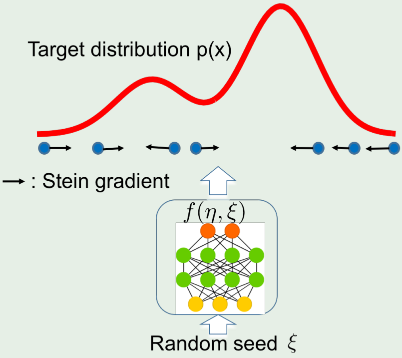



- Critically challenging to solve, when the structure of $f$ and input $\xi$ is complex, or even unknown (black-box).
- Progresses made only very recently:
  - Amortized SVGD: sidestep the difficulty using Stein variational gradient.
  - Other recent works: [Ranganath et al. 16, Mescheder et al. 17, Li et al. 17]

.

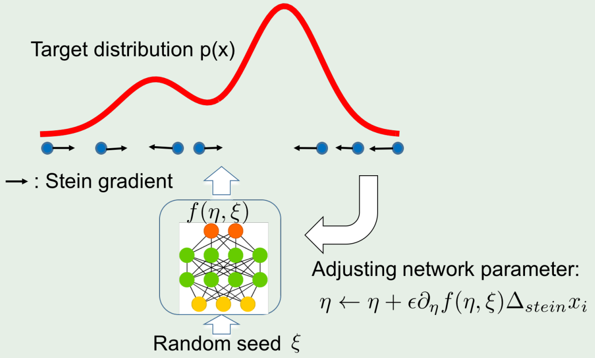
# Amortized SVGD [Wang, Liu 16, arXiv:1611.01722; Liu, Feng 16, arXiv:1612.00081]

- Amortized SVGD: Iteratively adjust $\eta$ to make the output move along the Stein variational gradient direction.



# Amortized SVGD [Wang, Liu 16, arXiv:1611.01722; Liu, Feng 16, arXiv:1612.00081]

- Amortized SVGD: Iteratively adjust $\eta$ to make the output move along the Stein variational gradient direction.



**Learning energy-based models from data**: Given observed data $\{x_{obs,i}\}_{i=1}^n$, want to learn model $p_\theta(x)$:

$$p_\theta(x) = \frac{1}{Z}\exp(\psi_\theta(x)), \qquad Z = \int \exp(\psi_\theta(x))dx.$$

- Deep energy model (when $\psi_\theta(x)$ is a neural net), graphical models, etc.
- Classical method: estimating $\theta$ by maximizing the likelihood:

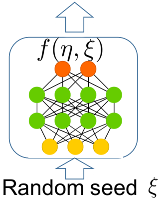
$$\max_\theta \left\{ L(\theta) \equiv \hat{\mathbb{E}}_{obs}[\log p_\theta(x)] \right\}.$$

Gradient: $$\nabla_\theta L(\theta) = \underbrace{\hat{\mathbb{E}}_{obs}[\partial_\theta \psi_\theta(x)]}_{\text{Average on observed data}} - \underbrace{\mathbb{E}_{p_\theta}[\partial_\theta \psi_\theta(x)]}_{\text{Expectation on model } p_\theta}$$

- Difficulty: requires to sample from $p(x|\theta)$ at every iteration.

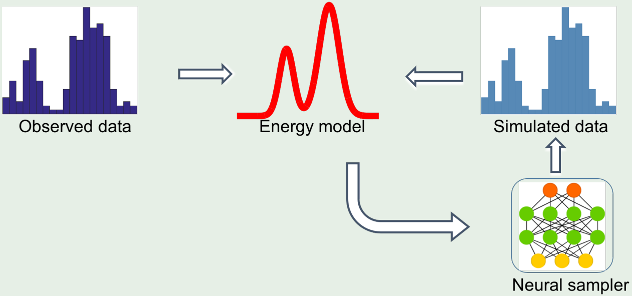
- Difficulty: requires to sample from $p(x|\theta)$ at every iteration.

Gradient: $\nabla_\theta L(\theta) = \underbrace{\hat{\mathbb{E}}_{obs}[\partial_\theta \psi_\theta(x)]}_{\text{Average on observed data}} - \underbrace{\mathbb{E}_{p_\theta}[\partial_\theta \psi_\theta(x)]}_{\text{Expectation on model } p_\theta}$



# Amortized MLE as an Adversarial Game

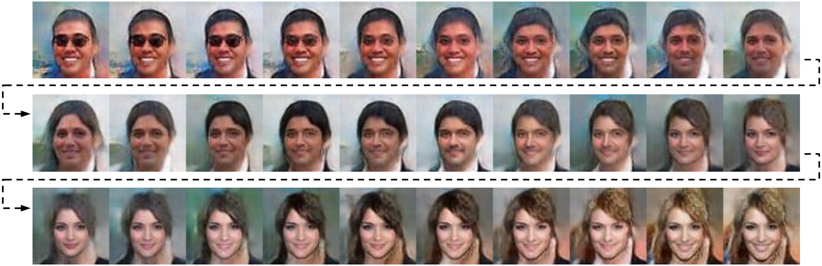
- *Can be treated as an adversarial process between the energy model and the neural sampler.*
- *Similar to generative adversarial networks (GAN) [Goodfellow et al., 2014].*



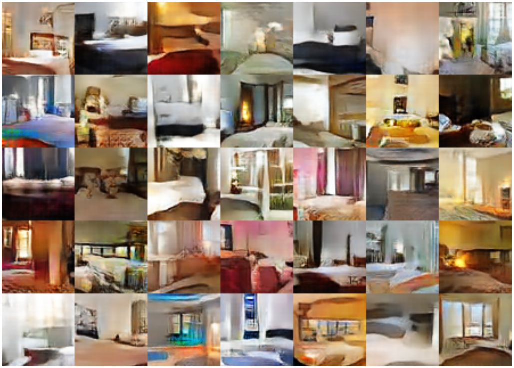
Real images

Generated by Stein neural sampler

- It captures the semantics of the data distribution.
- Changing the random input $\xi$ smoothly.



Real images



Generated by Stein neural sampler

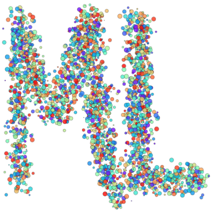

DCGAN SteinGAN

| | Real Training Set | 500 Duplicate | DCGAN | SteinGAN |
|---|---|---|---|---|
| Inception Score | 11.237 | 11.100 | 6.581 | 6.711 |
| Testing Accuracy | 92.58 % | 44.96 % | 44.78 % | 61.09 % |

## What do we learn?

- The traditional maximum likelihood (MLE) framework failed to generate realistic-looking images, over-dominated by the recent GAN approaches.
- It turns out amortized inference is the key.
- Connecting these two approaches allows us to combine their advantages.

# Thank You



*Powered by SVGD*

# References I

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems, pages 2672–2680, 2014.