A Stein Variational Framework for Deep Probabilistic Modeling

Qiang Liu
Dartmouth College
data \{x_i\} \quad \text{models } p(x)
Data-Model Discrepancy

\[ \mathcal{D}(\text{data } \{x_i\}_{i=1}^n, \text{ model } p) \]

- **Learning (model estimation):** Given \( \{x_i\} \), find an optimal \( p \):
  \[ \min_p \mathcal{D}(\{x_i\}, p). \]

- **Inference (or sampling):** Given \( p \), find optimal \( \{x_i\} \):
  \[ \min_{\{x_i\}} \mathcal{D}(\{x_i\}, p). \]

- **Model checking (e.g., goodness of fit test):** Given both \( p \) and \( \{x_i\} \), tell if they are consistent:
  \[ \mathcal{D}(\{x_i\}, p) \neq 0. \]
In Reality ...

- Modern machine learning = Complex data + Complex models
Unnormalized Distributions

- In practice, many distributions have unnormalized densities:

\[ p(x) = \frac{1}{Z} \tilde{p}(x), \quad Z = \int \tilde{p}(x) dx. \]

\( Z \): 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:

  \[ \text{Given } p, \text{ find } \{x_i\} \text{ to approximation } p. \]

- **Two applications:**
  - Policy optimization in reinforcement learning.
  - Training neural networks to generate natural images.
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): $\min_{\theta \in \Theta} \text{KL}(q_\theta \mid\mid p)$.
- 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.
Directly minimize the Kullback-Leibler (KL) divergence between \( \{x_i\} \) and \( p \):

\[
\min_{\{x_i\}} \text{KL}(\{x_i\}, p)
\]

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

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 \( p \):

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

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 \( p \):

\[
\phi = \arg \max_{\phi \in \mathcal{F}} \left\{ \text{KL}(q \mid\mid p) - \text{KL}(q_{[\epsilon \phi]} \mid\mid p) \right\}
\]

\[
\approx \arg \max_{\phi \in \mathcal{F}} \left\{ -\frac{\partial}{\partial \epsilon} \text{KL}(q_{[\epsilon \phi]} \mid\mid p) \bigg|_{\epsilon=0} \right\},
\]

//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 $\phi$:

$$-\frac{\partial}{\partial \epsilon} \text{KL}(q_{\epsilon \phi} \mid \mid p)\bigg|_{\epsilon=0} = \mathbb{E}_{x \sim q}[T_p \phi(x)].$$

where $T_p$ is a linear operator called **Stein operator** related to $p$:

$$T_p \phi(x) \overset{\text{def}}{=} \langle \nabla_x \log p(x), \phi(x) \rangle + \nabla_x \cdot \phi(x).$$

---

\[1\] $\nabla_x \cdot \phi = \sum_i \partial_{x_i} \phi$
Stein Variational Gradient Descent (SVGD) [Liu Wang, 2016]

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

$$-\frac{\partial}{\partial \epsilon} \text{KL}(q[\epsilon\phi] \mid \mid p) \bigg|_{\epsilon=0} = \mathbb{E}_{x \sim q}[T_p \phi(x)].$$

where $T_p$ is a linear operator called Stein operator related to $p$:

$$T_p \phi(x) \overset{\text{def}}{=} \langle \nabla_x \log p(x), \phi(x) \rangle + \nabla_x \cdot \phi(x). \quad (1)$$

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 $\phi$:

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

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

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

- **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”.

$$\nabla_x \cdot \phi = \sum_i \partial_{x_i} \phi$$
Stein Discrepancy

The optimization is equivalent to

\[ \mathcal{D}(q \parallel p) \overset{\text{def}}{=} \max_{\phi \in \mathcal{F}} \left\{ \mathbb{E}_q[T_p \phi] \right\} \]

where \( \mathcal{D}(q \parallel p) \) is called Stein discrepancy: \( \mathcal{D}(q \parallel p) = 0 \) iff \( q = p \) if \( \mathcal{F} \) is “large” enough.
Stein Discrepancy

The optimization is equivalent to

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

where $D(q \mid\mid p)$ is called Stein discrepancy: $D(q \mid\mid p) = 0$ iff $q = p$ if $F$ is “large” enough.

- The choice of $F$ 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

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

where \( \mathbb{D}(q \parallel p) \) is called Stein discrepancy: \( \mathbb{D}(q \parallel p) = 0 \) 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 $k(x, x')$:
    \[
    \mathcal{D}(q \parallel p) \overset{\text{def}}{=} \max_{\phi \in \mathcal{H}} \left\{ \mathbb{E}_q[\mathcal{T}_p \phi] \quad \text{s.t.} \quad \|\phi\|_{\mathcal{H}} \leq 1 \right\}
    \]
  - **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:**
  \[
  \mathcal{D}(q, p)^2 = \mathbb{E}_{x, x' \sim q}[\mathcal{T}_p^x \mathcal{T}_p^{x'} k(x, x')] \\
  \mathcal{T}_p^x, \mathcal{T}_p^{x'}: \text{Stein operator w.r.t. variable } x, x'.
  \]
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 $k(x, x')$:

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

- **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 \( p \):

\[
\mathbb{D}(\{x_i\}, p)^2 \overset{\text{def}}{=} \frac{1}{n^2} \sum_{ij} T_p x_i T_p x_j^T k(x_i, x_j).
\]

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.
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:

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

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

Stein Variational Gradient Descent

SVGD: iteratively update \( \{x_i\} \) until convergence:

\[
x_i \leftarrow x_i + \epsilon \hat{E}_{x \sim \{x_i\}} \left[ \nabla_x \log p(x) k(x, x_i) + \nabla_x k(x, x_i) \right], \quad \forall i = 1, \ldots, n.
\]

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:

\[
    x_i \leftarrow x_i + \epsilon \hat{E}_{x \sim \{ x_i \}} \left[ \nabla_x \log p(x) k(x, x_i) + \nabla_x k(x, x_i) \right], \quad \forall i = 1, \ldots, n.
\]

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 \( p(x) \)).
SVGD vs. MAP and Monte Carlo

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

- 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 \(n = 1\).

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:

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

which decreases KL divergence monotonically

$$\frac{d}{dt} KL(q_t \| p) = -\mathbb{D}(q_t, p)^2.$$
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:

\[
\frac{\partial}{\partial t} q_t(x) = -\text{grad}_H KL(q_t \parallel p),
\]

\(\text{grad}_H KL(q \parallel p)\) 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

- Stein Variational Gradient Descent (Our Method)
- Stochastic Langevin (Parallel SGLD)
- Particle Mirror Descent (PMD)
- Doubly Stochastic (DSVI)
- Stochastic Langevin (Sequential SGLD)

(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]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Boston</td>
<td>2.977 ± 0.093</td>
<td>2.957 ± 0.099</td>
<td>−2.579 ± 0.052</td>
<td>−2.504 ± 0.029</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td>Concrete</td>
<td>5.506 ± 0.103</td>
<td>5.324 ± 0.104</td>
<td>−3.137 ± 0.021</td>
<td>−3.082 ± 0.018</td>
<td>33</td>
<td>24</td>
</tr>
<tr>
<td>Energy</td>
<td>1.734 ± 0.051</td>
<td>1.374 ± 0.045</td>
<td>−1.981 ± 0.028</td>
<td>−1.767 ± 0.024</td>
<td>25</td>
<td>21</td>
</tr>
<tr>
<td>Kin8nm</td>
<td>0.098 ± 0.001</td>
<td>0.090 ± 0.001</td>
<td>0.901 ± 0.010</td>
<td>0.984 ± 0.008</td>
<td>118</td>
<td>41</td>
</tr>
<tr>
<td>Naval</td>
<td>0.006 ± 0.000</td>
<td>0.004 ± 0.000</td>
<td>3.735 ± 0.004</td>
<td>4.089 ± 0.012</td>
<td>173</td>
<td>49</td>
</tr>
<tr>
<td>Combined</td>
<td>4.052 ± 0.031</td>
<td>4.033 ± 0.033</td>
<td>−2.819 ± 0.008</td>
<td>−2.815 ± 0.008</td>
<td>136</td>
<td>51</td>
</tr>
<tr>
<td>Protein</td>
<td>4.623 ± 0.009</td>
<td>4.606 ± 0.013</td>
<td>−2.950 ± 0.002</td>
<td>−2.947 ± 0.003</td>
<td>682</td>
<td>68</td>
</tr>
<tr>
<td>Wine</td>
<td>0.614 ± 0.008</td>
<td>0.609 ± 0.010</td>
<td>−0.931 ± 0.014</td>
<td>−0.925 ± 0.014</td>
<td>26</td>
<td>22</td>
</tr>
<tr>
<td>Yacht</td>
<td><strong>0.778 ± 0.042</strong></td>
<td>0.864 ± 0.052</td>
<td><strong>−1.211 ± 0.044</strong></td>
<td><strong>−1.225 ± 0.042</strong></td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Year</td>
<td>8.733 ± NA</td>
<td>8.684 ± NA</td>
<td>−3.586 ± NA</td>
<td><strong>−3.580 ± NA</strong></td>
<td>7777</td>
<td>684</td>
</tr>
</tbody>
</table>
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.
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

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

- 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

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

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

\[
p(\theta) \propto \exp\left( \frac{1}{\alpha} J(\theta) \right)
\]

\( \alpha \) : temperature parameter.
Stein Variational Policy Gradient  [Liu et al. 17, arXiv:1704.02399]

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

\[
\theta_i \leftarrow \theta_i + \frac{\epsilon}{n} \sum_{j=1}^{n} \left[ \nabla_{\theta_j} J(\theta_j) k(\theta_j, \theta_i) + \alpha \nabla_{\theta_j} k(\theta_j, \theta_i) \right]
\]

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

\[
p(\theta) \propto \exp \left( \frac{1}{\alpha} J(\theta) \right) = \arg \max_{q} \left\{ \mathbb{E}_q [J(\theta)] + \alpha H(q) \right\}.
\]

\( \alpha \) : 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.
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 $p$.

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

![Diagram showing Amortized SVGD process](Image)

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

\[
\Delta_{\text{stein}}x_i
\]

Target distribution $p(x)$

$\rightarrow$: Stein gradient

$f(\eta, \xi)$

Random seed $\xi$

Adjusting network parameter:

$\eta \leftarrow \eta + \epsilon \partial_\eta f(\eta, \xi) \Delta_{\text{stein}}x_i$
Learning energy-based models from data: Given observed data \( \{ x_{obs,i} \}_{i=1}^n \), want to learn model \( p_\theta(x) \):

\[
p_\theta(x) = \frac{1}{Z} \exp(\psi_\theta(x)), \quad Z = \int \exp(\psi_\theta(x)) dx.
\]

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

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

Gradient:

\[
\nabla_\theta L(\theta) = \mathbb{E}_{obs}[\partial_\theta \psi_\theta(x)] - \mathbb{E}_{p_\theta}[\partial_\theta \psi_\theta(x)]
\]

Average on observed data - 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:

$$\nabla_\theta L(\theta) = \hat{E}_{\text{obs}}[\partial_\theta \psi_\theta(x)] - E_{p_\theta}[\partial_\theta \psi_\theta(x)]$$

- **Average on observed data**
- **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
The table compares the performance of DCGAN and SteinGAN in terms of Inception Score and Testing Accuracy. The results are as follows:

<table>
<thead>
<tr>
<th></th>
<th>Real Training Set</th>
<th>500 Duplicate</th>
<th>DCGAN</th>
<th>SteinGAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception Score</td>
<td>11.237</td>
<td>11.100</td>
<td>6.581</td>
<td>6.711</td>
</tr>
<tr>
<td>Testing Accuracy</td>
<td>92.58 %</td>
<td>44.96 %</td>
<td>44.78 %</td>
<td>61.09 %</td>
</tr>
</tbody>
</table>
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