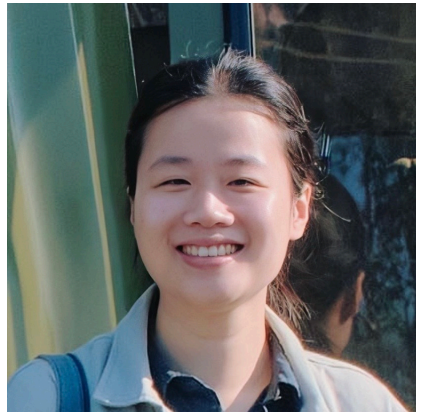
uai2025 Rio de Janeiro, Brazil I 21-25 July, 2025

FLATIRON INSTITUTE IMPERIAL

Modern Approximate Inference: Variational Methods and Beyond

Diana Cai

dcai@flatironinstitute.org

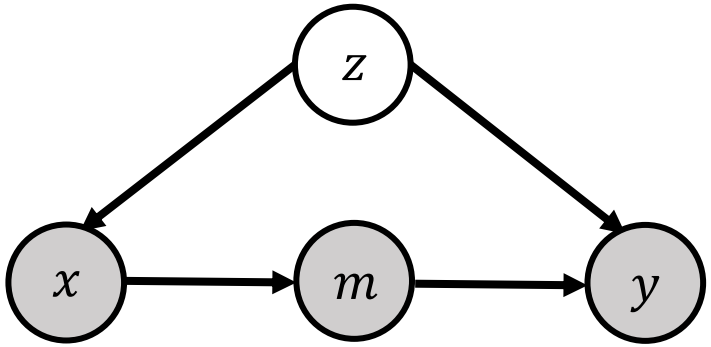


Yingzhen Li

yingzhen.li@imperial.ac.uk

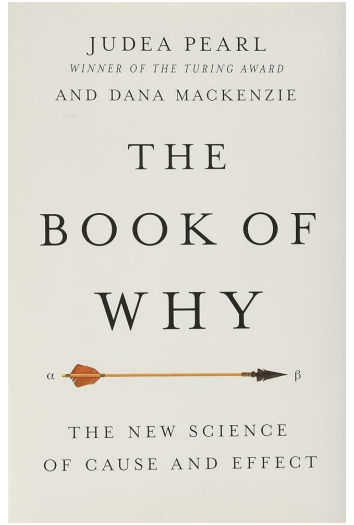
Why Probabilistic Inference?

Causal Inference and Causal Discovery



x: smoking
m: tar in lungs
y: lung cancer?

z: hidden confounder



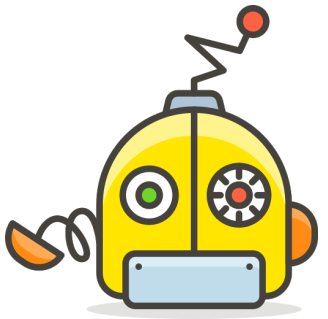
How to estimate causal effects and/or discover causal relationships,
when there exists hidden confounders?

Why Probabilistic Inference?

Uncertainty Quantification (Desiderata)

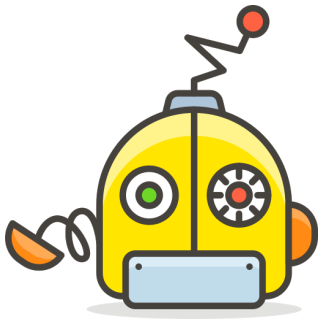
Here's this patient's health record:
…
Could you suggest some next actions to improve her conditions?

Here's a list of potential treatment options based on the health record:
[Treatment 1] with x% confidence (breakdown quantities)
…



Here's the conditions of this construction site:
…
Could you tell me what the potential safety issues are?

Here's a list of potential safety issues that need to be look after:
[Issue 1] with x% confidence (breakdown quantities)
…



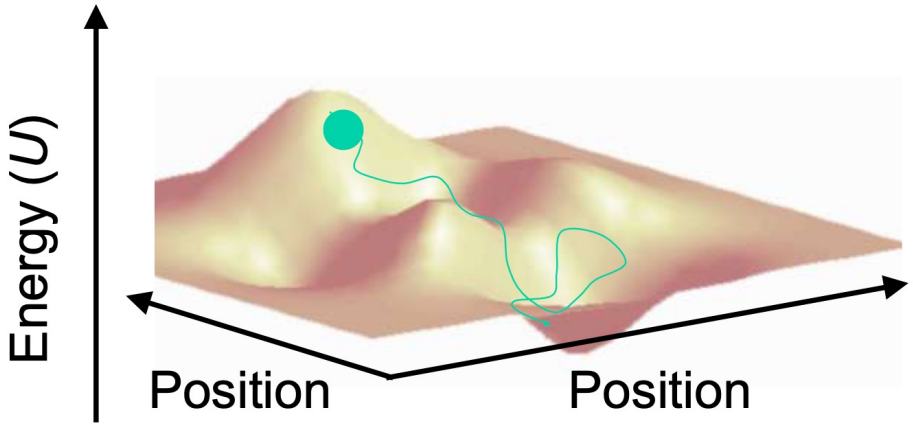
Why Probabilistic Inference?

Molecular Dynamics Simulations

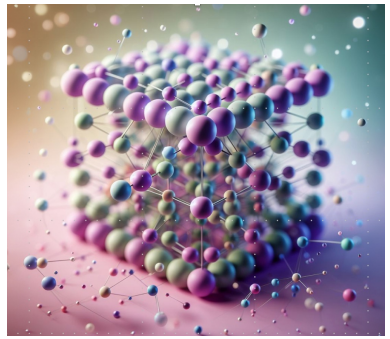
- Energy of the molecular state: $U(z)$
- "Atoms never stop jiggling"
  - Sampling from the Boltzmann distribution (target):

$$\pi_T \propto \exp[-\frac{U(z)}{k_B T}]$$

  - Computing "free energy"

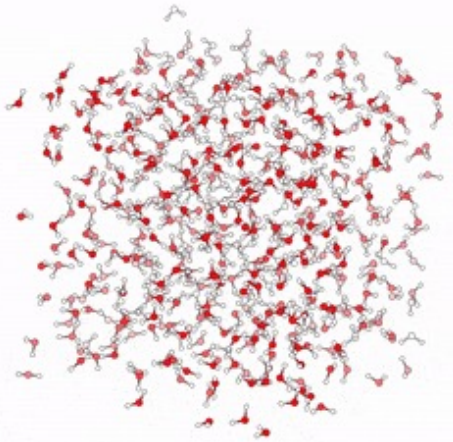


Drug discovery



Material design

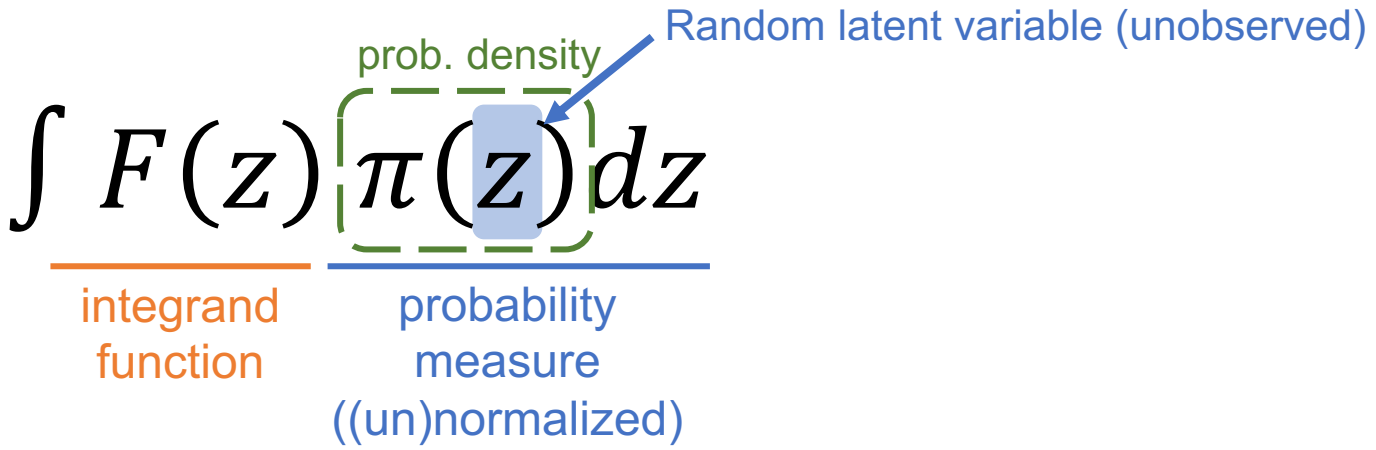
Protein folding



https://web.stanford.edu/class/cs279/lectures/lecture4.pdf
https://en.wikipedia.org/wiki/Thermodynamic_free_energy

The Central Computation Problem

- Inference: infer the unknowns
  - Unobserved/latent variables in the model
  - Quantities depending on the latent variables in the model



Computation for Inference:
"statistics about the unknown"

(We view Jensen Huang's "inference" terminology as "prediction".)
(For discrete probability measures, integration becomes discrete sum.)

Computation Challenge

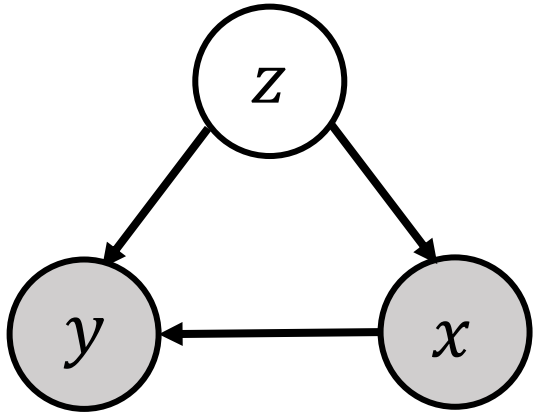
- The central computational task for probabilistic inference:

$$\int F(z)\,\pi(z)dz$$

Causal Inference and Causal Discovery: Counterfactuals

"What is the counterfactual outcome y if I change the cause value to x' instead?

$F(z) = p(y|x = x', z)$, $\pi(z) = p(z \mid x, y)$,
x, y = observed cause & outcome
x' = alternative cause value



Computation Challenge

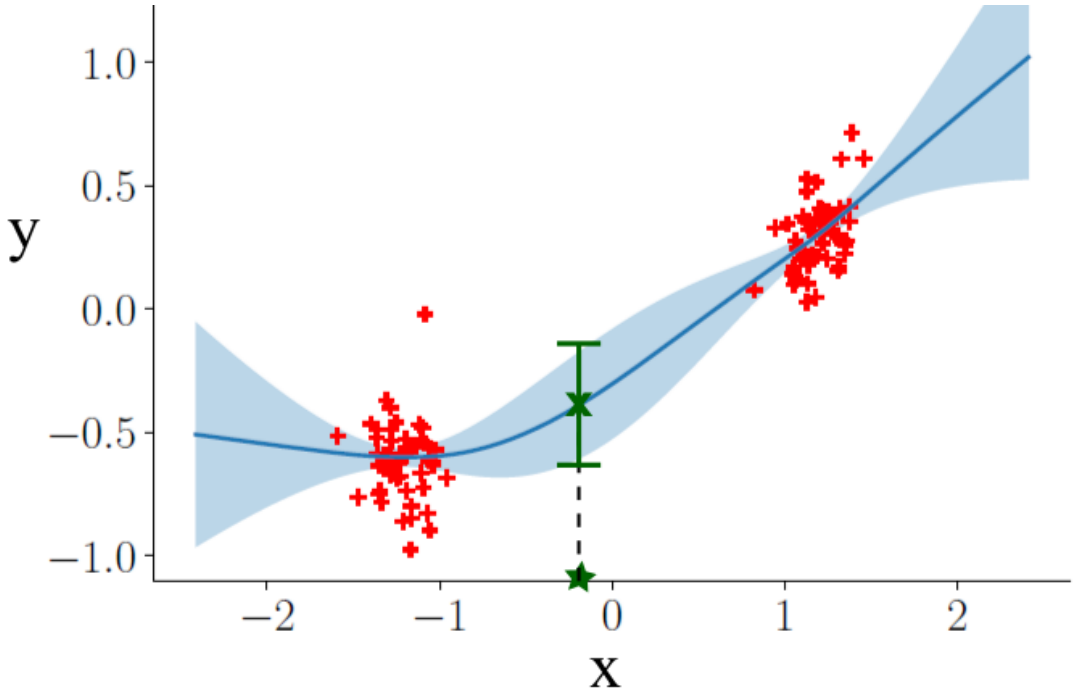
- The central computational task for probabilistic inference:

$$\int F(z)\,\pi(z)dz$$

Uncertainty Quantification

"What is the prediction distribution of the test output given a test input?"

$F(z) = p(y|x,z)$, $\pi(z) = p(z \mid D)$,
D = observed datapoints



Computation Challenge

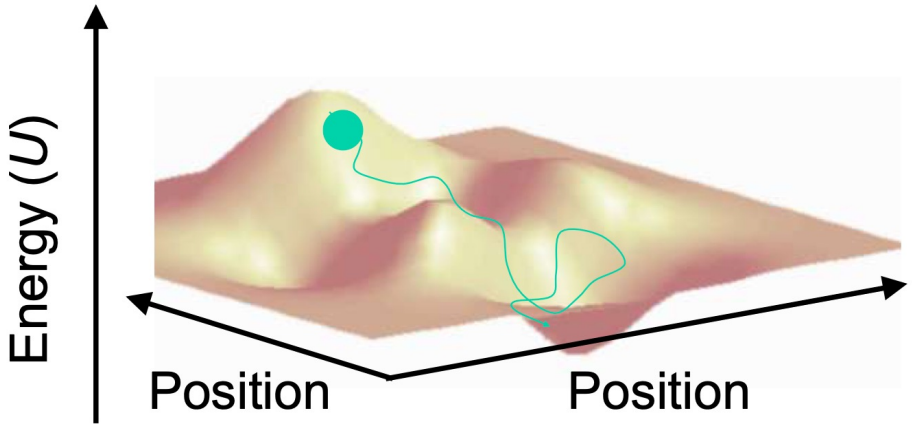
- The central computational task for probabilistic inference:

$$\int F(z)\,\pi(z)dz$$

Molecular Dynamics Simulations

"What is the free energy of a thermodynamic system at a particular end state?"

$$F(z) = 1,\ \pi(z) = \exp[-U(z)/k_B T],$$
$$\tilde{\pi} = \frac{1}{Z}\exp[-U(z)/k_B T]$$
$-k_B T \log Z$: "free energy"



(for biological conformational changes, ligand-macromolecule binding, chemical reaction mechanisms, etc.)

https://web.stanford.edu/class/cs279/lectures/lecture4.pdf
https://en.wikipedia.org/wiki/Thermodynamic_free_energy

Computation Challenge

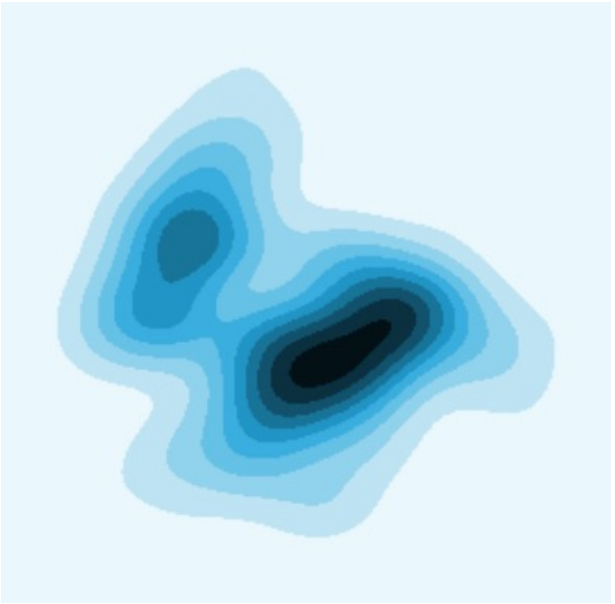
- The central computational task for probabilistic inference:

$$\int F(z)\,\pi(z)dz$$

Calculating Statistics

"What is the mean of this distribution?"

$F(z) = z$, $\pi(z)$ can be complicated and high dimensional



Computation Challenge

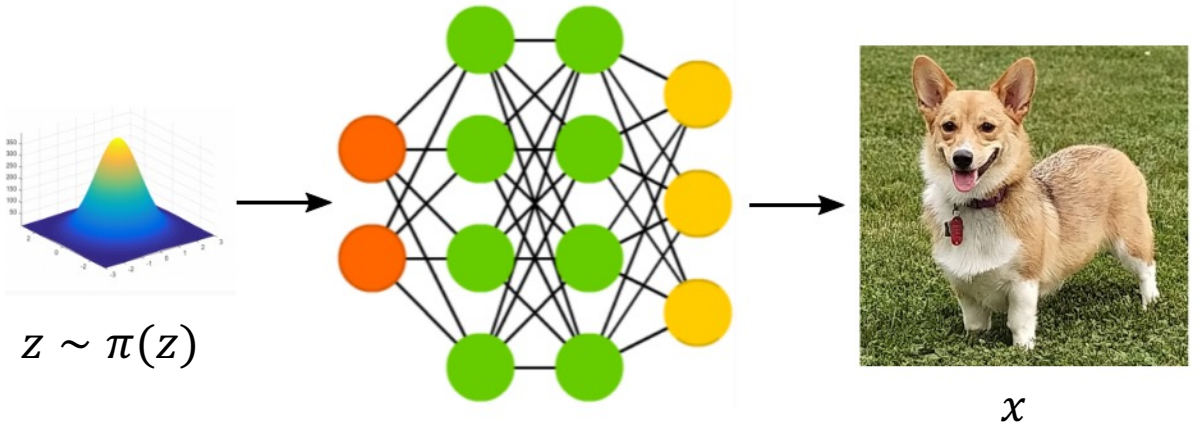
- The central computational task for probabilistic inference:

$$\int F(z)\,\pi(z)dz$$

Evaluating the Likelihood

"What is the probability of generating this image?"

$F(z) = N(net(z), \sigma^2 I)$, $\pi(z) = N(0, I)$



$z \sim \pi(z)$

x

Computation Challenge

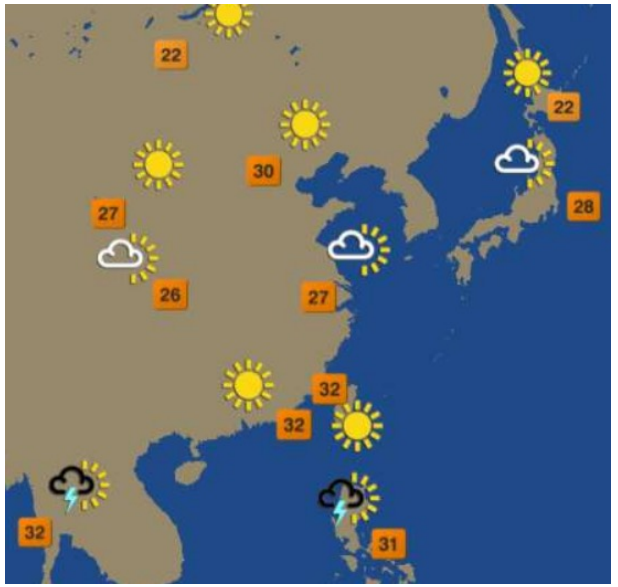
- The central computational task for probabilistic inference:

$$\int F(z)\,\pi(z)dz$$

Probabilistic Forecasting

"What is the weather forecast for tomorrow?"

Answering this in a Bayesian way:
z: forecasting simulator settings
D: historical weather record
$F(z) = Simulator(z)$, $\pi(z) = p(z \mid D)$



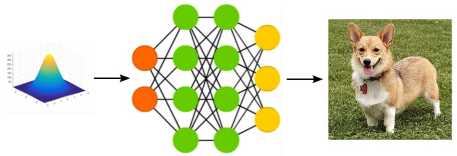
Nature laughs at the difficulties of integration.

-- Pierre-Simon Laplace

Gordon and Sorkin. *The Armchair Science Reader.* New York 1959

Integration in Bayesian Computation

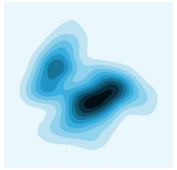
The integrand $F(z)$ is intractable

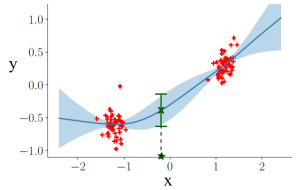


Implicit Models

Bayesian Optimization, Probabilistic Numerics

The probability distribution $\pi(z)$ is intractable





Approximate Inference

(in a strict sense)

This tutorial

$$\int F(z)\pi(z)dz$$

Both $F(z)$ and $\pi(z)$ are intractable

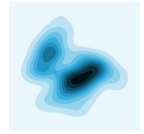
Approximate Bayesian Computation

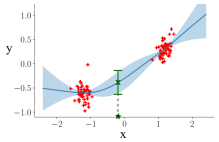
Approximate Inference

- Central task: approximate $\pi(z)$

$$q(z) \approx \pi(z)$$

The probability distribution $\pi(z)$ is intractable



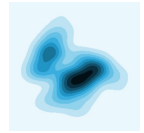


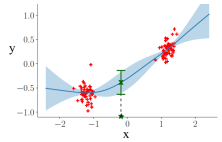
Approximate Inference
(in a strict sense)

This tutorial

Approximate Inference

The probability distribution $\pi(z)$ is intractable





Approximate Inference
(in a strict sense)

This tutorial

- Central task: approximate $\pi(z)$

😇 😈

$$q(z) \approx \pi(z)$$

Not simpler than computing $\int F(z)\,\pi(z)dz$ for a given integrand $F(z)$!

- Example: when $\pi(z) \propto \exp[-E(z)]$, estimating via self-normalized importance sampling

$$\int F(z)\,\pi(z)dz = E_{\pi(z)}[F(z)] = E_{q(z)}\left[\frac{\pi(z)}{q(z)}F(z)\right] \approx \sum_{k=1}^{K} w_k F(z^k)$$

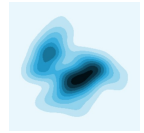
$$q(z) \gg \pi(z),\ z^k \sim q(z),\ \widetilde{w}_k := \frac{\exp[-E(z^k)]}{q(z^k)},\ w_k = \frac{\widetilde{w}_k}{\sum_{k'=1}^{K}\widetilde{w}_{k'}}$$

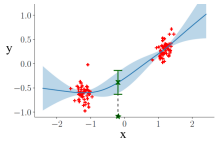
Solving $q(z) \approx \pi(z)$ can help computing $\int F(z)\,\pi(z)dz$ for any tractable integrand $F(z)$:

$$\int F(z)\,\pi(z)dz = E_{\pi(z)}[F(z)] \approx E_{q(z)}\left[\frac{\pi(z)}{q(z)}F(z)\right] \approx \sum_{k=1}^{K} F(z^k),\ z^k \sim q(z)$$

Approximate Inference

The probability distribution $\pi(z)$ is intractable





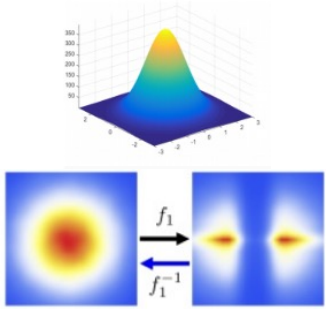
Approximate Inference
(in a strict sense)

This tutorial

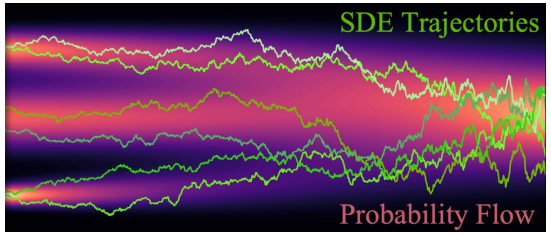
- Central task: approximate $\pi(z)$

$$q(z) \approx \pi(z)$$

Approximate distribution design



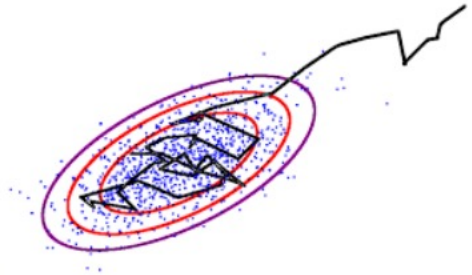
Explicit distributions



Distributions with samples but expensive/intractable density

Algorithm for fitting $q(z)$ to $\pi(z)$

$$\min \; Loss(q(z), \pi(z))$$



Optimisation-based approaches

Sampling/Transport-based approaches

Previous Tutorial at NeurIPS 2020

Basics

Probabilistic modelling

Approximate inference

Variational inference

Advances

Scalable variational inference

Monte Carlo techniques

Amortized inference

q distribution design

Optimization objective design

Applications

Bayesian neural networks

Partially observed VAEs

Future challenges

(NeurIPS 2020 tutorial “Advances in Approximate Inference” by Cheng Zhang and Yingzhen Li)

Prompt to Imagen 4
“A sci-fi image of two robots, one standing and doing calculations on whiteboard and another just sit in front of a computer, the computer screen shows codes, the whiteboard shows an integration equation ∫ F(z) π(z) dz”

This Tutorial at UAI 2025 (5 Years Later)

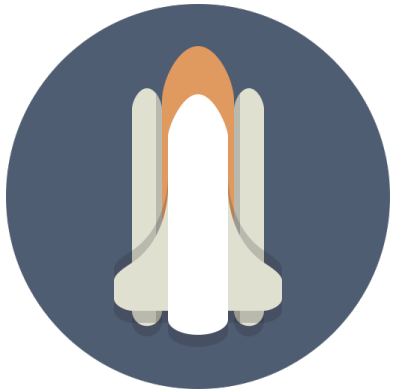
Modern Basics

Variational inference

Scaling & Monte Carlo

Amortized inference

Design principles I



New Advances

Score-based approximations

Diffusion-based approximations

Flow-based approximations

Design principles II

Applications

Molecular Dynamics Simulation

Simulation-based Inference

Deep Generative Models

Future challenges

Agenda for Today

- Basics
- (Coffee Break (30mins))
- Advances

- Small breaks (e.g., 5mins each) during sessions

Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

Example: Bayesian inference

$$\pi(z) = p(z \mid \text{data}) = \frac{p(z)\, p(\text{data} \mid z)}{\int p(z)\, p(\text{data} \mid z)\, dz}$$

Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

Example: Bayesian inference

$$\pi(z) = p(z \mid \text{data}) = \frac{p(z)\, p(\text{data} \mid z)}{\int p(z)\, p(\text{data} \mid z)\, dz}$$

Often can't be computed in closed form

Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

Example: Bayesian inference

$$\pi(z) = p(z \mid \text{data}) = \frac{p(z)\, p(\text{data} \mid z)}{\int p(z)\, p(\text{data} \mid z)\, dz}$$

Often can't be computed in closed form

Also recall: physics (Boltzmann distribution)

$$\pi(z) = \frac{\exp(-U(z)/(k_B T))}{\int \exp(-U(z)/(k_B T)) dz}$$

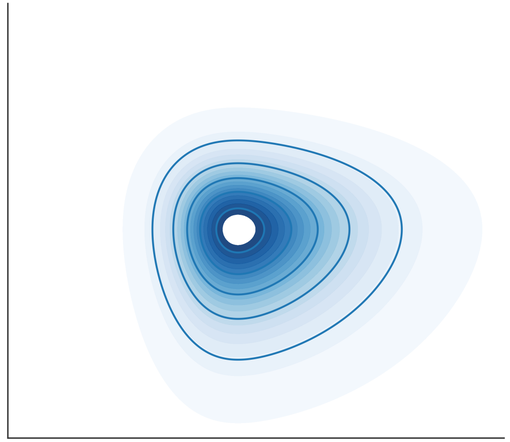
Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

Example: Bayesian inference

$$\pi(z) = p(z \mid \text{data}) = \frac{p(z)\, p(\text{data} \mid z)}{\int p(z)\, p(\text{data} \mid z)\, dz}$$

Often can't be computed in closed form



Also recall: physics (Boltzmann distribution)

$$\pi(z) = \frac{\exp(-U(z)/(k_B T))}{\int \exp(-U(z)/(k_B T)) dz}$$

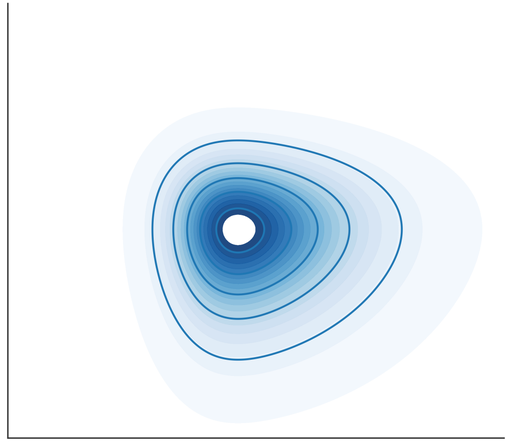
Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

Example: Bayesian inference

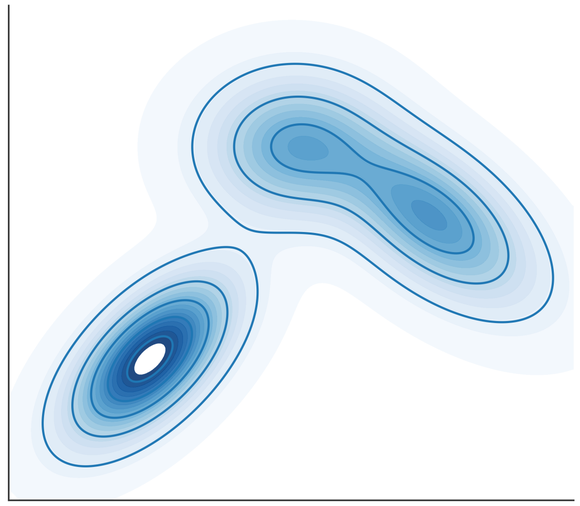
$$\pi(z) = p(z \mid \text{data}) = \frac{p(z)\, p(\text{data} \mid z)}{\int p(z)\, p(\text{data} \mid z)\, dz}$$

Often can't be computed in closed form



Also recall: physics (Boltzmann distribution)

$$\pi(z) = \frac{\exp(-U(z)/(k_B T))}{\int \exp(-U(z)/(k_B T)) dz}$$



Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

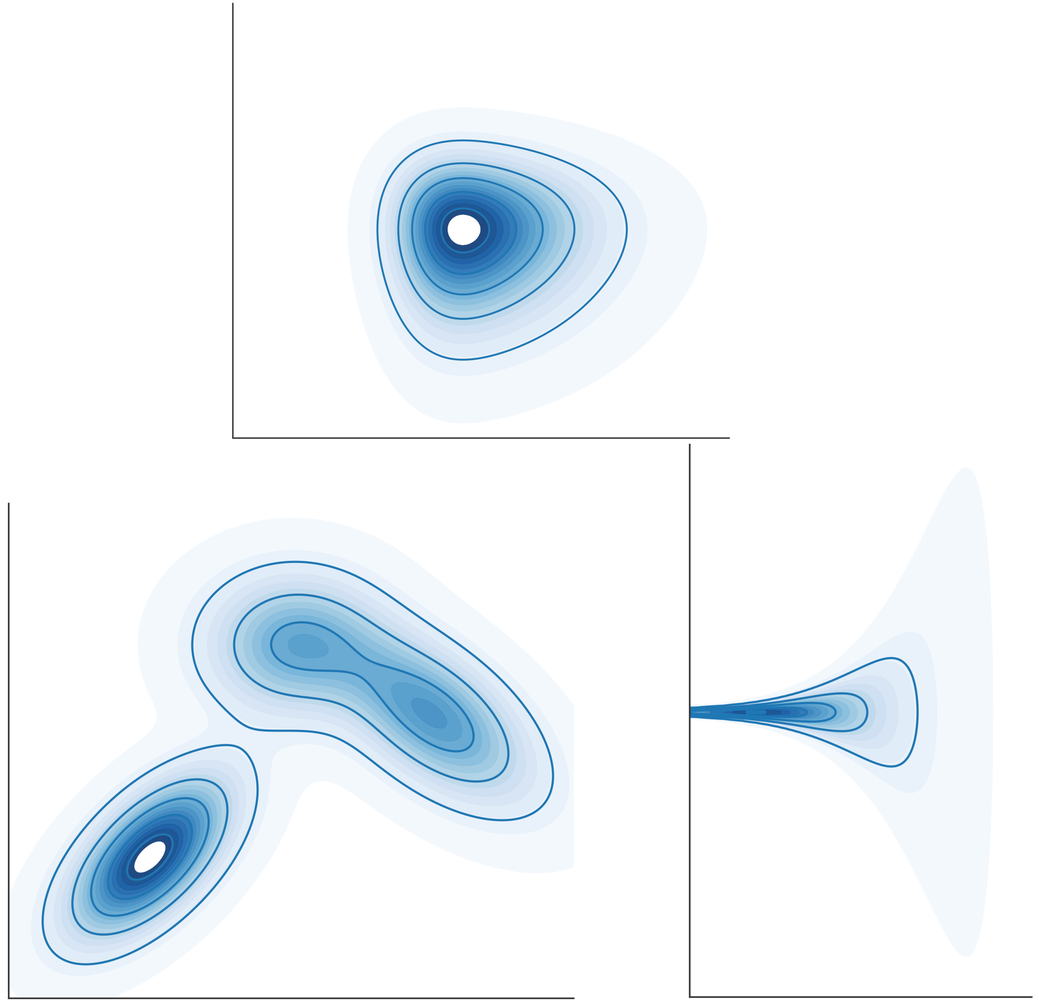
Example: Bayesian inference

$$\pi(z) = p(z \mid \text{data}) = \frac{p(z)\, p(\text{data} \mid z)}{\int p(z)\, p(\text{data} \mid z)\, dz}$$

Often can't be computed in closed form

Also recall: physics (Boltzmann distribution)

$$\pi(z) = \frac{\exp(-U(z)/(k_B T))}{\int \exp(-U(z)/(k_B T))dz}$$



Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

Variational inference: the basic idea

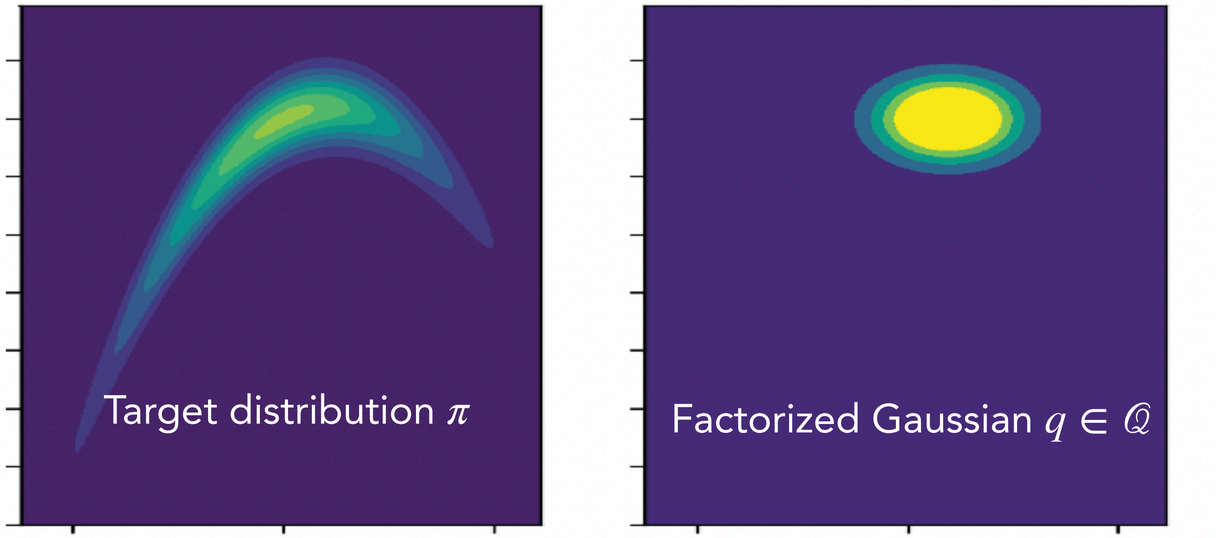
Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

Example: Gaussian family with diagonal covariance $\mathcal{N}(\mu, \Sigma)$
(Often called “factorized” or “mean field” Gaussian)

Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

Example: Gaussian family with diagonal covariance $\mathcal{N}(\mu, \Sigma)$
(Often called “factorized” or “mean field” Gaussian)



[Zoltowski et al., 2021]

Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

Step 1: Specify a *divergence* $\mathscr{D}(q;\pi)$ between variational density $q(z)$ and target $\pi(z)$

Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

Step 1: Specify a *divergence* $\mathscr{D}(q;\pi)$ between variational density $q(z)$ and target $\pi(z)$

Step 2: Find the member of the family $\mathcal{Q}$ that minimizes the divergence, i.e.,

Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

Step 1: Specify a *divergence* $\mathscr{D}(q;\pi)$ between variational density $q(z)$ and target $\pi(z)$

Step 2: Find the member of the family $\mathcal{Q}$ that minimizes the divergence, i.e.,

$$q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q;\pi)$$

Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

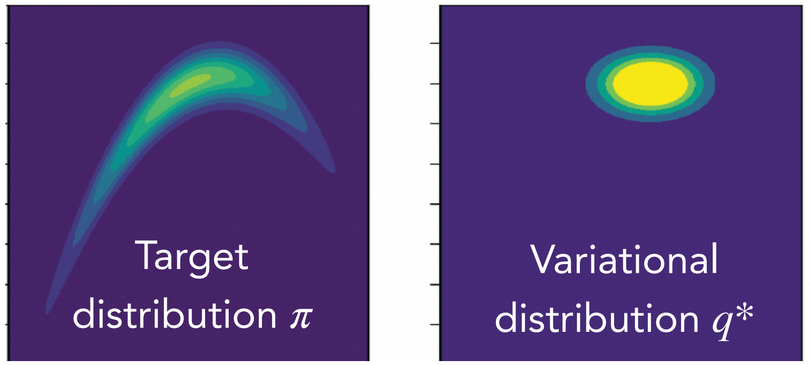
Step 1: Specify a *divergence* $\mathscr{D}(q;\pi)$ between variational density $q(z)$ and target $\pi(z)$

Step 2: Find the member of the family $\mathcal{Q}$ that minimizes the divergence, i.e.,

$$q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q;\pi)$$

Example: find the mean and (diagonal) covariance such that it is closest to π

$$\mu^*, \Sigma^* = \arg\min_{\mu,\Sigma} \mathscr{D}(N(\mu,\Sigma);\pi)$$



Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

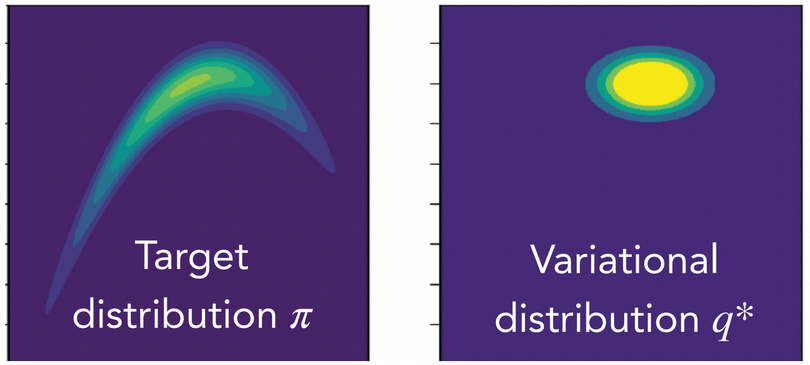
Step 1: Specify a *divergence* $\mathscr{D}(q;\pi)$ between variational density $q(z)$ and target $\pi(z)$

Step 2: Find the member of the family $\mathcal{Q}$ that minimizes the divergence, i.e.,

$$q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q;\pi)$$

Example: find the mean and (diagonal) covariance such that it is closest to π

$$\mu^*, \Sigma^* = \arg\min_{\mu,\Sigma} \mathscr{D}(N(\mu,\Sigma);\pi)$$



Variational inference: the basic idea

Goal: Approximate an *intractable* target density $\pi(z)$ by a simpler family of densities $\mathcal{Q}$

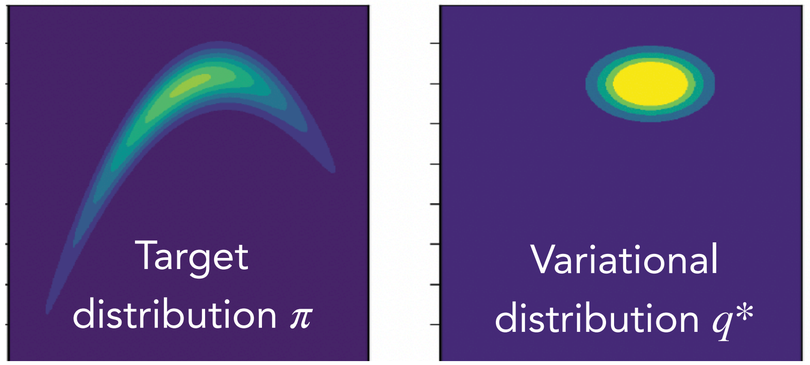
Step 1: Specify a *divergence* $\mathscr{D}(q;\pi)$ between variational density $q(z)$ and target $\pi(z)$

Step 2: Find the member of the family $\mathcal{Q}$ that minimizes the divergence, i.e.,

$$q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q;\pi)$$

Example: find the mean and (diagonal) covariance such that it is closest to π

$$\mu^*, \Sigma^* = \arg\min_{\mu,\Sigma} \mathscr{D}(N(\mu,\Sigma);\pi)$$



Step 3: Use variational density q^* instead of target $\pi(z)$ in downstream tasks: e.g.,

$$\mathbb{E}_{\pi(z)}[f(z)] \approx \mathbb{E}_{q^*(z)}[f(z)]$$

What divergence to use?

Kullback-Leibler (KL) divergence (“forward KL”)

$$\mathscr{D}_{KL}(\pi; q) = \int \log\left(\frac{\pi(z)}{q(z)}\right)\ \pi(z)\, dz$$

What divergence to use?

Kullback-Leibler (KL) divergence (“forward KL”)

$$\mathscr{D}_{KL}(\pi; q) = \int \log\left(\frac{\pi(z)}{q(z)}\right)\ \pi(z)\, dz$$

$$\mathscr{D}_{KL}(\pi; q) \neq \mathscr{D}_{KL}(q; \pi)$$

What divergence to use?

Kullback-Leibler (KL) divergence (“forward KL”)

$$\mathscr{D}_{KL}(\pi; q) = \int \log\left(\frac{\pi(z)}{q(z)}\right) \pi(z)\, dz$$

$$\mathscr{D}_{KL}(\pi; q) \neq \mathscr{D}_{KL}(q; \pi)$$

$$\mathscr{D}_{KL}(\pi; q) = 0 \text{ iff } \pi = q$$

What divergence to use?

Kullback-Leibler (KL) divergence (“forward KL”)

$$\mathscr{D}_{KL}(\pi; q) = \int \log \left(\frac{\pi(z)}{q(z)} \right) \pi(z)\, dz$$

$$\mathscr{D}_{KL}(\pi; q) \neq \mathscr{D}_{KL}(q; \pi)$$

$$\mathscr{D}_{KL}(\pi; q) = 0 \text{ iff } \pi = q$$

What divergence to use?

Kullback-Leibler (KL) divergence (“forward KL”)

$$\mathscr{D}_{KL}(\pi; q) = \int \log \left(\frac{\pi(z)}{q(z)} \right) \pi(z)\, dz$$

$$\mathscr{D}_{KL}(\pi; q) \neq \mathscr{D}_{KL}(q; \pi)$$

$$\mathscr{D}_{KL}(\pi; q) = 0 \text{ iff } \pi = q$$

Problem: we often don’t know $\pi(z)$ - can’t sample from it or evaluate it pointwise

What divergence to use?

Kullback-Leibler (KL) divergence ("forward KL")

$$\mathscr{D}_{KL}(\pi; q) = \int \log \left(\frac{\pi(z)}{q(z)} \right) \pi(z)\, dz$$

Most commonly used in VI: "reverse" KL divergence

$$\mathscr{D}_{KL}(\pi; q) \neq \mathscr{D}_{KL}(q; \pi)$$

$$\mathscr{D}_{KL}(\pi; q) = 0 \text{ iff } \pi = q$$

Problem: we often don't know $\pi(z)$ - can't sample from it or evaluate it pointwise

What divergence to use?

Kullback-Leibler (KL) divergence ("forward KL")

$$\mathscr{D}_{KL}(\pi; q) = \int \log\left(\frac{\pi(z)}{q(z)}\right) \pi(z)\, dz$$

$$\mathscr{D}_{KL}(\pi; q) \neq \mathscr{D}_{KL}(q; \pi)$$

$$\mathscr{D}_{KL}(\pi; q) = 0 \text{ iff } \pi = q$$

Problem: we often don't know $\pi(z)$ - can't sample from it or evaluate it pointwise

Most commonly used in VI: "reverse" KL divergence

$$\mathscr{D}_{KL}(q; \pi) = \int \log\left(\frac{q(z)}{\pi(z)}\right) q(z)\, dz$$

What divergence to use?

Kullback-Leibler (KL) divergence ("forward KL")

$$\mathscr{D}_{KL}(\pi; q) = \int \log \left(\frac{\pi(z)}{q(z)} \right) \pi(z)\, dz$$

$$\mathscr{D}_{KL}(\pi; q) \neq \mathscr{D}_{KL}(q; \pi)$$

$$\mathscr{D}_{KL}(\pi; q) = 0 \text{ iff } \pi = q$$

Problem: we often don't know $\pi(z)$ - can't sample from it or evaluate it pointwise

Most commonly used in VI: "reverse" KL divergence

$$\mathscr{D}_{KL}(q; \pi) = \int \log \left(\frac{q(z)}{\pi(z)} \right) q(z)\, dz$$

But I still can't evaluate $\pi(z)$.

What divergence to use?

Kullback-Leibler (KL) divergence ("forward KL")

$$\mathscr{D}_{KL}(\pi; q) = \int \log\left(\frac{\pi(z)}{q(z)}\right) \pi(z)\, dz$$

$$\mathscr{D}_{KL}(\pi; q) \neq \mathscr{D}_{KL}(q; \pi)$$

$$\mathscr{D}_{KL}(\pi; q) = 0 \text{ iff } \pi = q$$

Problem: we often don't know $\pi(z)$ - can't sample from it or evaluate it pointwise

Most commonly used in VI: "reverse" KL divergence

$$\mathscr{D}_{KL}(q; \pi) = \int \log\left(\frac{q(z)}{\pi(z)}\right) q(z)\, dz$$

But I still can't evaluate $\pi(z)$.

$$= \int \log\left(\frac{q(z)}{\rho(z)}\right) q(z)\, dz + \text{constant}$$

What divergence to use?

Kullback-Leibler (KL) divergence ("forward KL")

$$\mathscr{D}_{KL}(\pi; q) = \int \log\left(\frac{\pi(z)}{q(z)}\right) \pi(z)\, dz$$

$$\mathscr{D}_{KL}(\pi; q) \neq \mathscr{D}_{KL}(q; \pi)$$

$$\mathscr{D}_{KL}(\pi; q) = 0 \text{ iff } \pi = q$$

Problem: we often don't know $\pi(z)$ - can't sample from it or evaluate it pointwise

Most commonly used in VI: "reverse" KL divergence

$$\mathscr{D}_{KL}(q; \pi) = \int \log\left(\frac{q(z)}{\pi(z)}\right) q(z)\, dz$$

But I still can't evaluate $\pi(z)$.

$$= \int \log\left(\frac{q(z)}{\rho(z)}\right) q(z)\, dz + \text{constant}$$

What divergence to use?

Kullback-Leibler (KL) divergence ("forward KL")

$$\mathscr{D}_{KL}(\pi; q) = \int \log\left(\frac{\pi(z)}{q(z)}\right) \pi(z)\, dz$$

$$\mathscr{D}_{KL}(\pi; q) \neq \mathscr{D}_{KL}(q; \pi)$$

$$\mathscr{D}_{KL}(\pi; q) = 0 \text{ iff } \pi = q$$

Problem: we often don't know $\pi(z)$ - can't sample from it or evaluate it pointwise

Most commonly used in VI: "reverse" KL divergence

$$\mathscr{D}_{KL}(q; \pi) = \int \log\left(\frac{q(z)}{\pi(z)}\right) q(z)\, dz$$

But I still can't evaluate $\pi(z)$.

$$= \int \log\left(\frac{q(z)}{\rho(z)}\right) q(z)\, dz + \text{constant}$$

Quantities inside the second integral are **tractable**: I *can* evaluate $\rho(z)$,
e.g., in Bayesian inference it's the joint distribution $\rho(z) = p(z)\, p(\text{data} \mid z)$

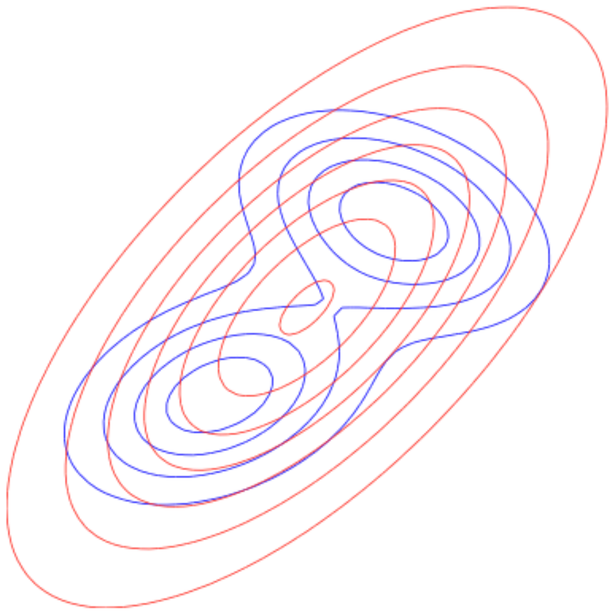
Forward KL vs Reverse KL

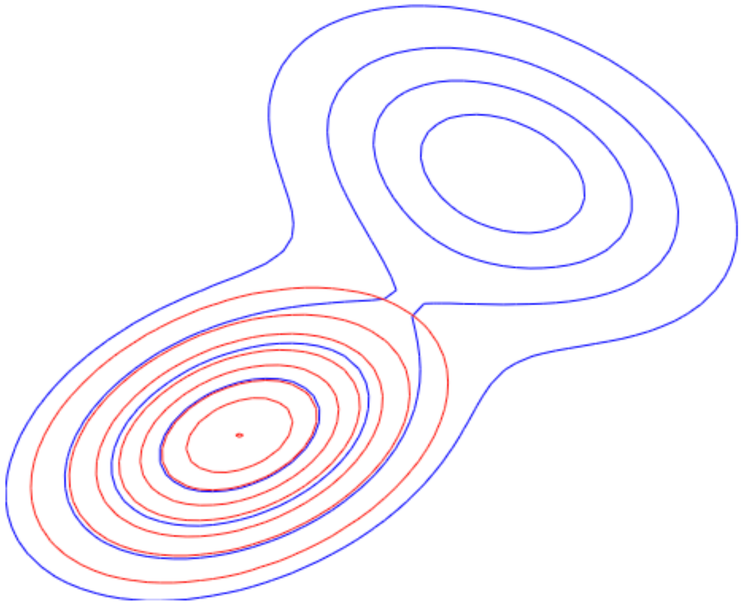
Forward KL

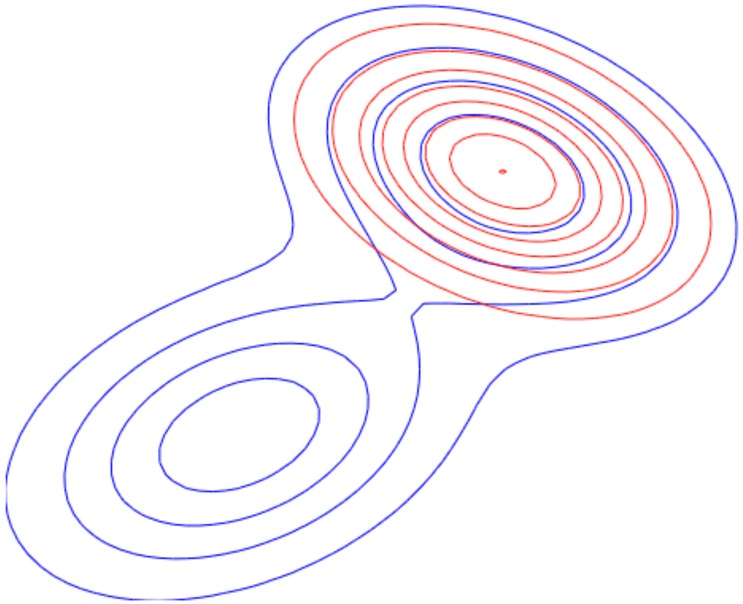
$\mathscr{D}_{KL}(\pi; q)$

Reverse KL

$\mathscr{D}_{KL}(q; \pi)$







Murphy. Probabilistic Machine Learning. Book 1.

Forward KL vs Reverse KL

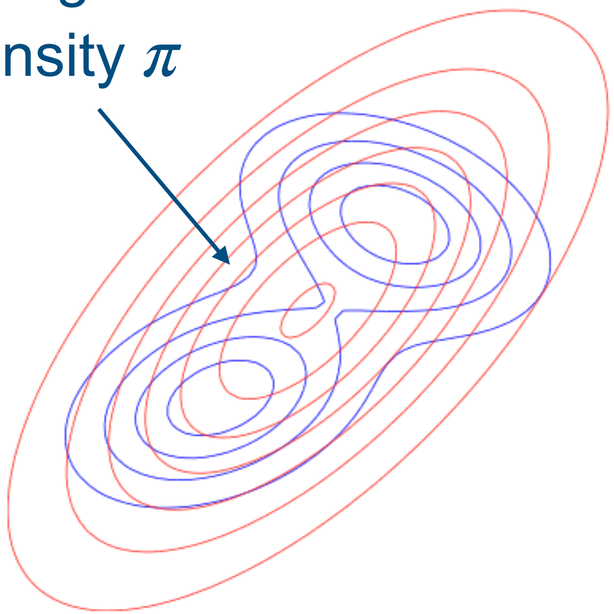
Forward KL

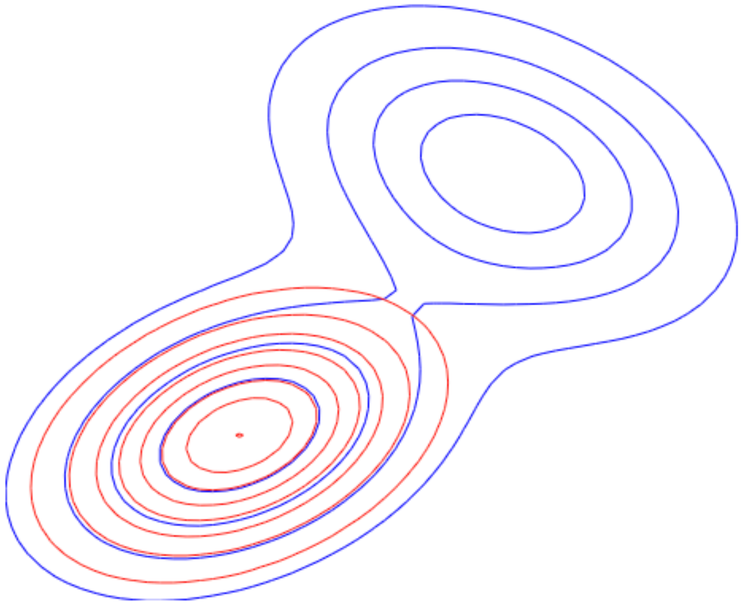
$\mathscr{D}_{KL}(\pi; q)$

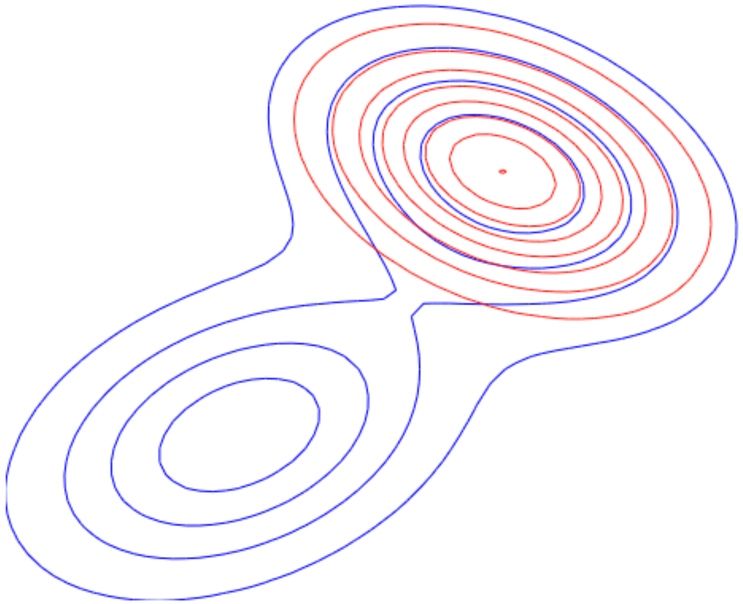
Reverse KL

$\mathscr{D}_{KL}(q; \pi)$

Target density π







Murphy. Probabilistic Machine Learning. Book 1.

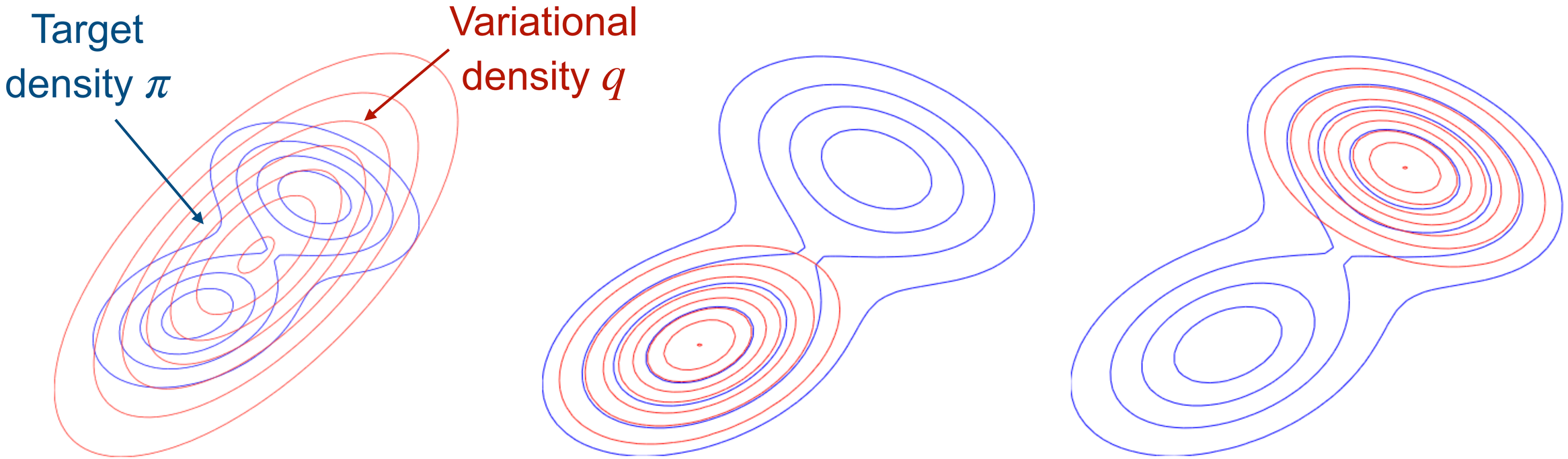
Forward KL vs Reverse KL

Forward KL

$\mathscr{D}_{KL}(\pi; q)$

Reverse KL

$\mathscr{D}_{KL}(q; \pi)$



Murphy. Probabilistic Machine Learning. Book 1.

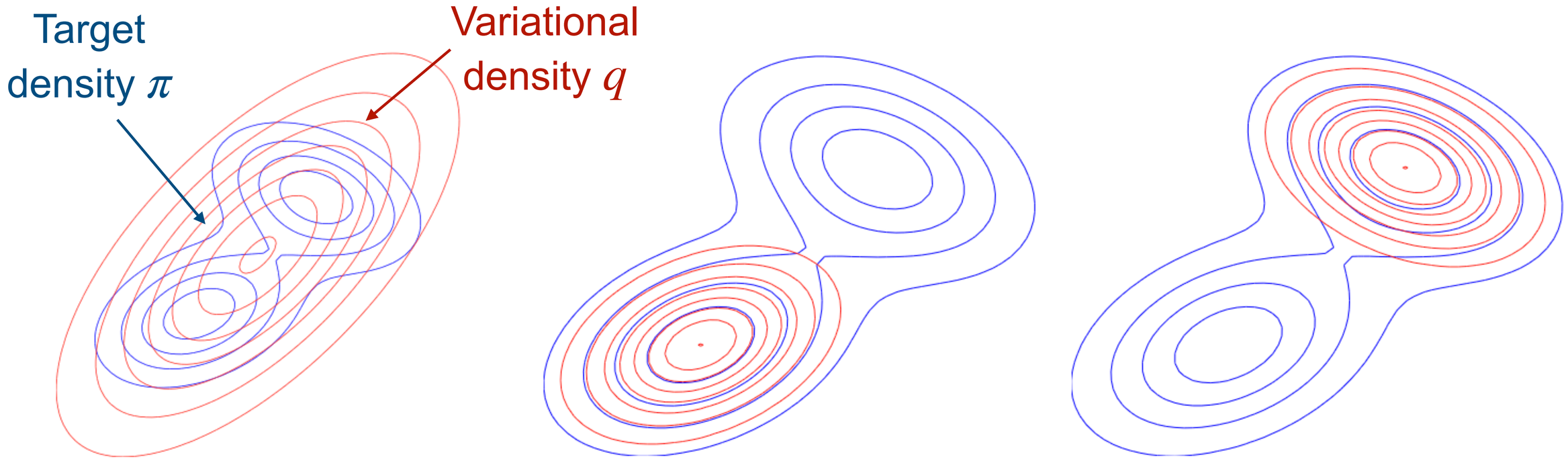
Forward KL vs Reverse KL

Forward KL

$\mathscr{D}_{KL}(\pi; q)$

Reverse KL

$\mathscr{D}_{KL}(q; \pi)$



Here q “covers” π

Murphy. Probabilistic Machine Learning. Book 1.

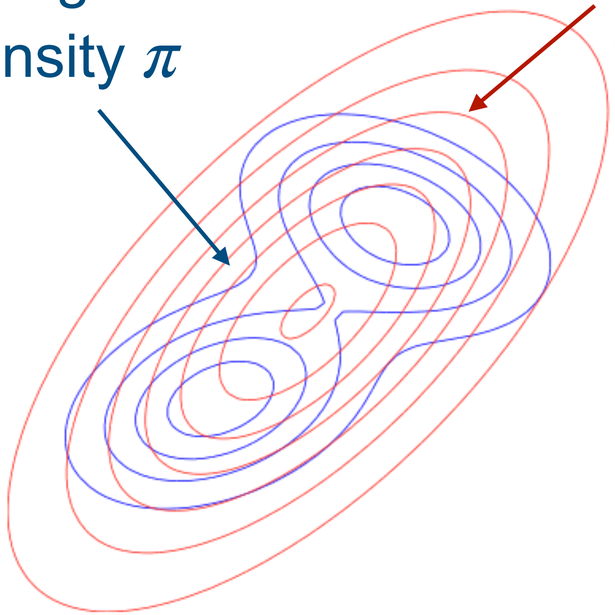
Forward KL vs Reverse KL

Forward KL

$\mathscr{D}_{KL}(\pi; q)$

Target density π

Variational density q

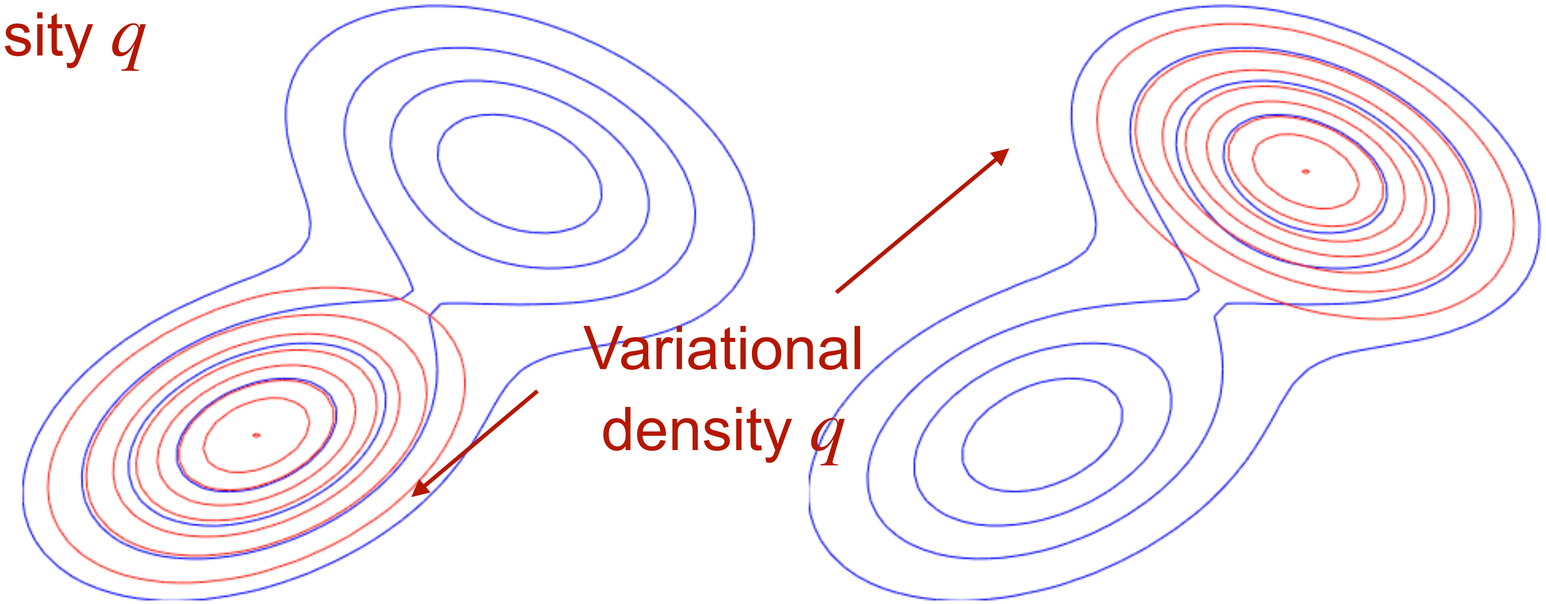


Here q "covers" π

Reverse KL

$\mathscr{D}_{KL}(q; \pi)$

Variational density q



Here q latches onto one of the modes of π

Murphy. Probabilistic Machine Learning. Book 1.

In practice: Evidence Lower Bound (ELBO)

The reverse KL divergence is intractable. Recall that we can write it in this tractable form:

$$\mathscr{D}_{KL}(q;p) = \int \log\left(\frac{q(z)}{\rho(z)}\right)\, q(z)\, dz + \text{constant}$$

In practice: Evidence Lower Bound (ELBO)

The reverse KL divergence is intractable. Recall that we can write it in this tractable form:

$$\mathscr{D}_{KL}(q;p) = \int \log\left(\frac{q(z)}{\rho(z)}\right) q(z)\,dz + \text{constant}$$

Bayes: $\rho(z) = p(z)\,p(\text{data}\,|\,z)$

In practice: Evidence Lower Bound (ELBO)

The reverse KL divergence is intractable. Recall that we can write it in this tractable form:

$$\mathscr{D}_{KL}(q;p) = \int \log\left(\frac{q(z)}{\rho(z)}\right) q(z)\,dz + \text{constant}$$

Bayes: $\rho(z) = p(z)\,p(\text{data} \mid z)$

The ***evidence lower bound*** (ELBO) is the negative KL (up to an additive constant):

In practice: Evidence Lower Bound (ELBO)

The reverse KL divergence is intractable. Recall that we can write it in this tractable form:

$$\mathscr{D}_{KL}(q;p) = \int \log\left(\frac{q(z)}{\rho(z)}\right) q(z)\,dz + \text{constant}$$

Bayes: $\rho(z) = p(z)\,p(\text{data} \mid z)$

The ***evidence lower bound*** (ELBO) is the negative KL (up to an additive constant):

$$\text{ELBO}(q) = \int \log\left(\frac{\rho(z)}{q(z)}\right) q(z)\,dz$$

In practice: Evidence Lower Bound (ELBO)

The reverse KL divergence is intractable. Recall that we can write it in this tractable form:

$$\mathscr{D}_{KL}(q;p) = \int \log\left(\frac{q(z)}{\rho(z)}\right) q(z)\,dz + \text{constant}$$

Bayes: $\rho(z) = p(z)\,p(\text{data}\,|\,z)$

The ***evidence lower bound*** (ELBO) is the negative KL (up to an additive constant):

$$\text{ELBO}(q) = \int \log\left(\frac{\rho(z)}{q(z)}\right) q(z)\,dz$$

In practice, we maximize the ELBO:

In practice: Evidence Lower Bound (ELBO)

The reverse KL divergence is intractable. Recall that we can write it in this tractable form:

$$\mathscr{D}_{KL}(q;p) = \int \log\left(\frac{q(z)}{\rho(z)}\right) q(z)\,dz + \text{constant}$$

Bayes: $\rho(z) = p(z)\,p(\text{data}\,|\,z)$

The ***evidence lower bound*** (ELBO) is the negative KL (up to an additive constant):

$$\text{ELBO}(q) = \int \log\left(\frac{\rho(z)}{q(z)}\right) q(z)\,dz$$

In practice, we maximize the ELBO:

- max ELBO(q) = min $\mathscr{D}_{KL}(q;p)$ w.r.t the parameters of q

In practice: Evidence Lower Bound (ELBO)

The reverse KL divergence is intractable. Recall that we can write it in this tractable form:

$$\mathscr{D}_{KL}(q;p) = \int \log\left(\frac{q(z)}{\rho(z)}\right) q(z)\,dz + \text{constant}$$

Bayes: $\rho(z) = p(z)\,p(\text{data}\,|\,z)$

The ***evidence lower bound*** (ELBO) is the negative KL (up to an additive constant):

$$\text{ELBO}(q) = \int \log\left(\frac{\rho(z)}{q(z)}\right) q(z)\,dz$$

In practice, we maximize the ELBO:

- max ELBO(q) = min $\mathscr{D}_{KL}(q;p)$ w.r.t the parameters of q
- Not a convex objective

In practice: Evidence Lower Bound (ELBO)

The reverse KL divergence is intractable. Recall that we can write it in this tractable form:

$$\mathscr{D}_{KL}(q;p) = \int \log\left(\frac{q(z)}{\rho(z)}\right) q(z)\,dz + \text{constant}$$

Bayes: $\rho(z) = p(z)\,p(\text{data}\,|\,z)$

The ***evidence lower bound*** (ELBO) is the negative KL (up to an additive constant):

$$\text{ELBO}(q) = \int \log\left(\frac{\rho(z)}{q(z)}\right) q(z)\,dz$$

In practice, we maximize the ELBO:

- max ELBO(q) = min $\mathscr{D}_{KL}(q;p)$ w.r.t the parameters of q
- Not a convex objective
- Bayes: it is a ***lower bound*** on the marginal likelihood, i.e., ELBO$(q) \leq \log p(\text{data})$

In practice: Evidence Lower Bound (ELBO)

The reverse KL divergence is intractable. Recall that we can write it in this tractable form:

$$\mathscr{D}_{KL}(q;p) = \int \log\left(\frac{q(z)}{\rho(z)}\right) q(z)\,dz + \text{constant}$$

Bayes: $\rho(z) = p(z)\,p(\text{data}\,|\,z)$

The ***evidence lower bound*** (ELBO) is the negative KL (up to an additive constant):

$$\text{ELBO}(q) = \int \log\left(\frac{\rho(z)}{q(z)}\right) q(z)\,dz \quad = \log p(\text{data}) - \mathscr{D}_{KL}(q(z);p(z\,|\,\text{data}))$$

In practice, we maximize the ELBO:

- max ELBO(q) = min $\mathscr{D}_{KL}(q;p)$ w.r.t the parameters of q
- Not a convex objective
- Bayes: it is a ***lower bound*** on the marginal likelihood, i.e., ELBO(q) $\leq \log p(\text{data})$

In practice: Evidence Lower Bound (ELBO)

The reverse KL divergence is intractable. Recall that we can write it in this tractable form:

$$\mathscr{D}_{KL}(q;p) = \int \log\left(\frac{q(z)}{\rho(z)}\right) q(z)\,dz + \text{constant}$$

Bayes: $\rho(z) = p(z)\,p(\text{data}\,|\,z)$

The ***evidence lower bound*** (ELBO) is the negative KL (up to an additive constant):

$$\text{ELBO}(q) = \int \log\left(\frac{\rho(z)}{q(z)}\right) q(z)\,dz \quad = \log p(\text{data}) - \mathscr{D}_{KL}(q(z);p(z\,|\,\text{data}))$$

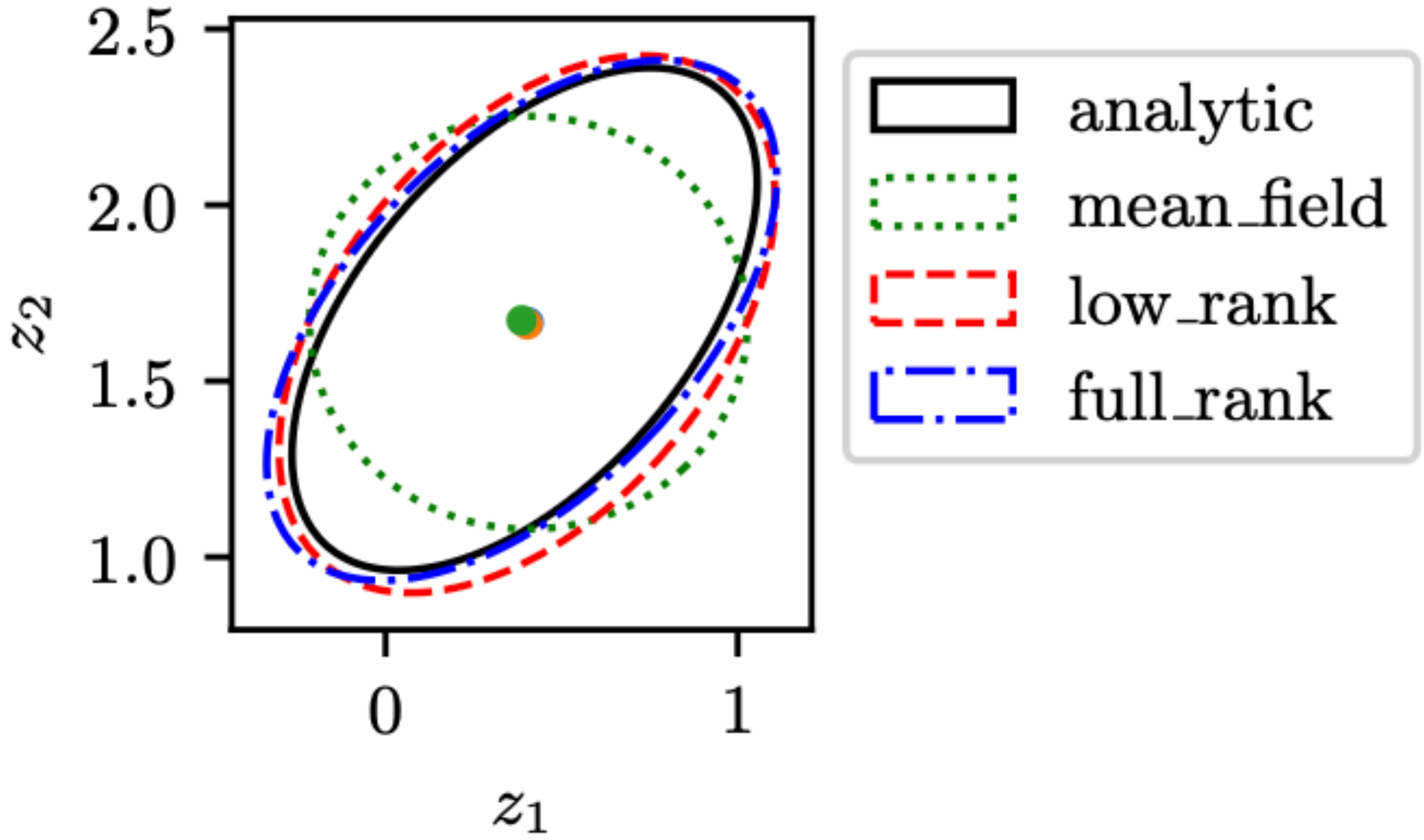
In practice, we maximize the ELBO:

- max ELBO(q) = min $\mathscr{D}_{KL}(q;p)$ w.r.t the parameters of q
- Not a convex objective
- Bayes: it is a ***lower bound*** on the marginal likelihood, i.e., ELBO(q) $\leq \log p(\text{data})$

Classical VI uses coordinate-ascent to maximize the ELBO

Blei et al. Variational Inference: A Review for Statisticians. Journal of the American Statistician, 2017.

Variational approximations with Gaussians



Murphy. Probabilistic Machine Learning. Book 2.

Many other divergences have been considered in VI

Some other divergences / distances considered in VI:

- α-divergence
- χ^2-divergence
- Hellinger distance
- Kernelized Stein discrepancy
- Fisher divergence

Long history behind variational inference

Early work on approximate inference, e.g., Metropolis-Hastings, importance sampling

pre-90s

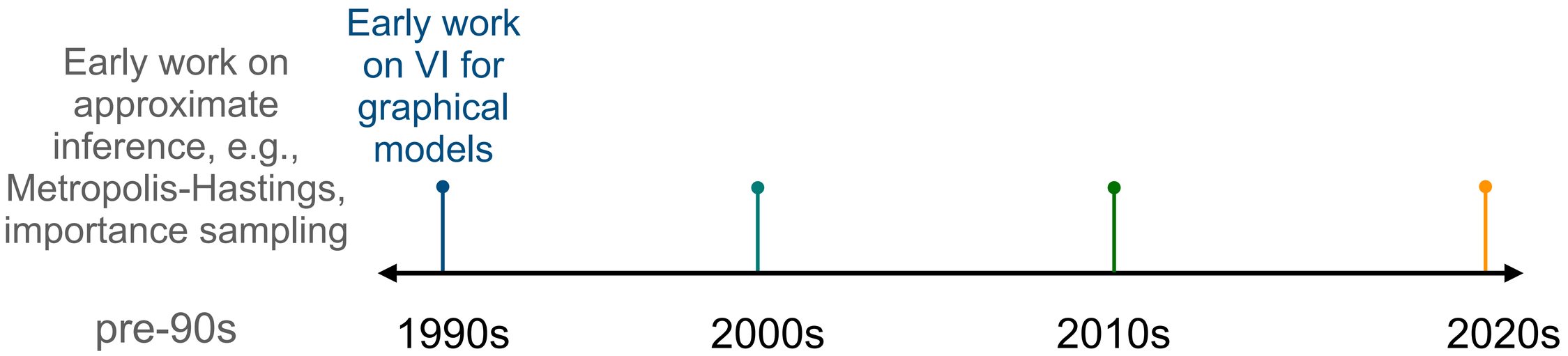
1990s

2000s

2010s

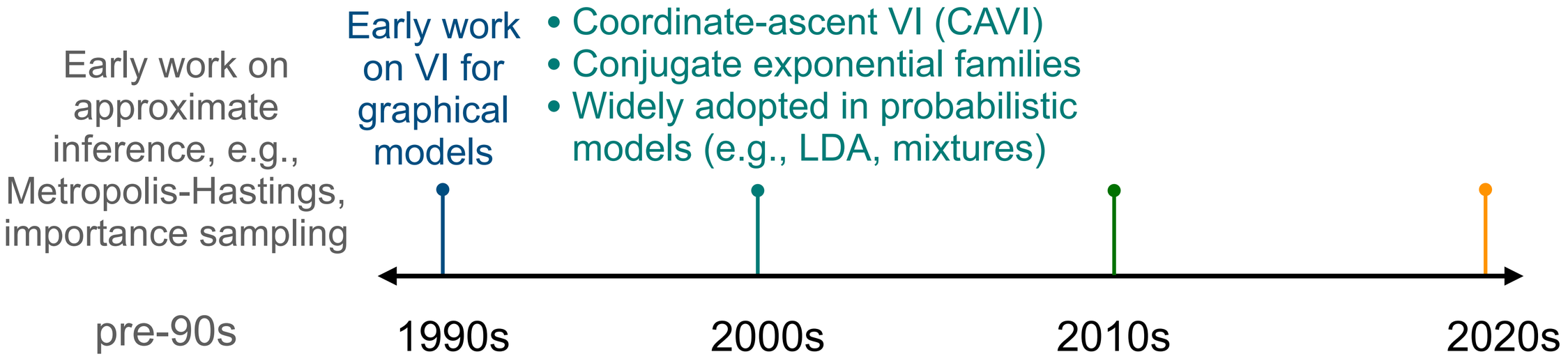
2020s

Long history behind variational inference



Jordan, Ghahramani, Jaakkola, Saul. Introduction to variational methods for graphical models. *Machine Learning*, 1999.

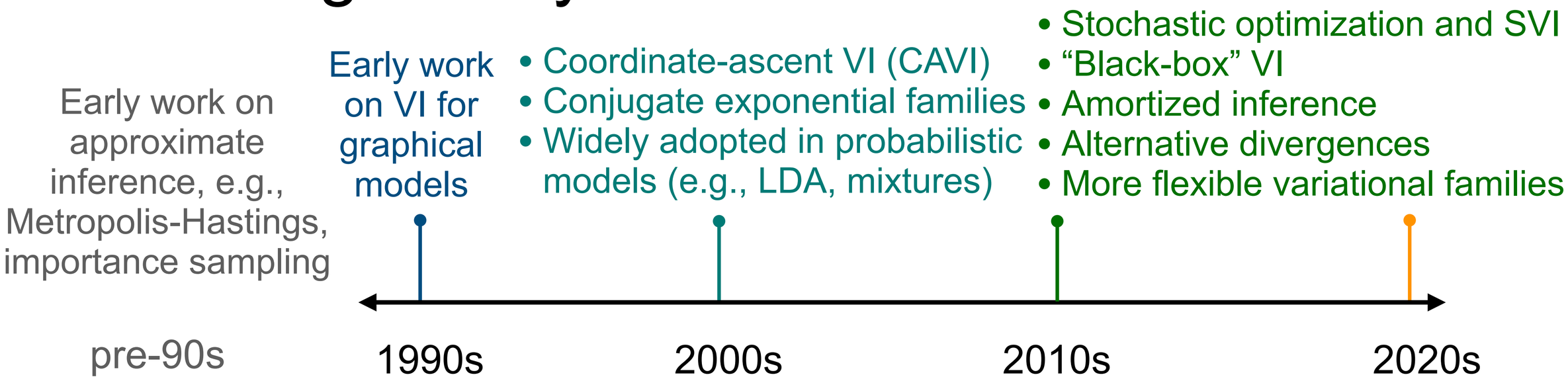
Long history behind variational inference



Jordan, Ghahramani, Jaakkola, Saul. Introduction to variational methods for graphical models. *Machine Learning*, 1999.

Blei, Ng, Jordan. Latent Dirichlet allocation. Journal of Machine Learning Research, 2003.

Long history behind variational inference

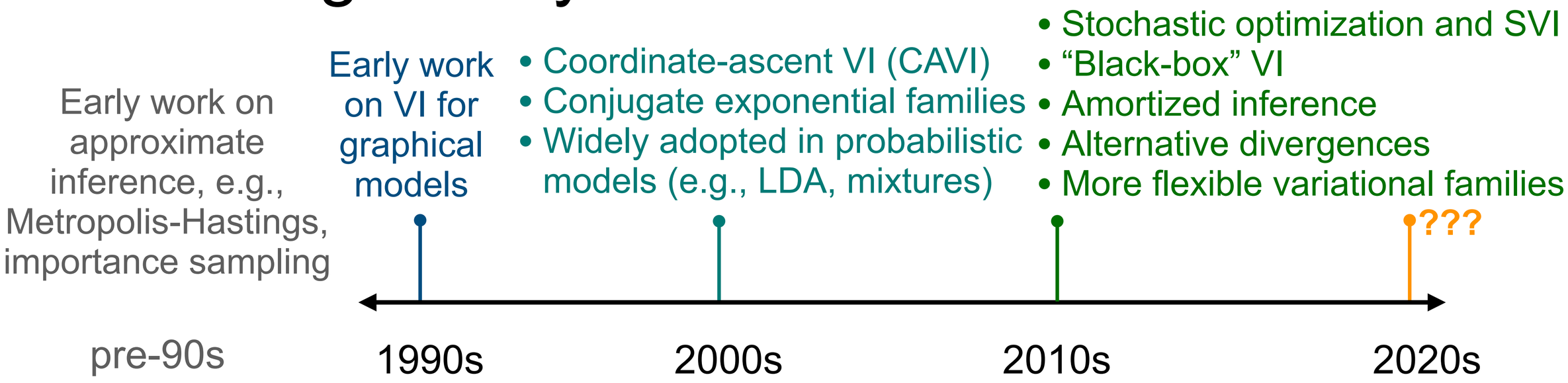


Jordan, Ghahramani, Jaakkola, Saul. Introduction to variational methods for graphical models. *Machine Learning*, 1999.

Blei, Ng, Jordan. Latent Dirichlet allocation. Journal of Machine Learning Research, 2003.

Hoffman, Blei, Wang, Paisley. Stochastic variational inference. Journal of Machine Learning Research, 2013.
Ranganath, Gerrish, Blei. Black box variational inference. AISTATS, 2014.
Rezende and Mohamed. Variational inference with normalizing flows. ICML, 2016.
Kucukelbir, Tran, Ranganath, Gelman, Blei. Automatic differentiation variational inference. JMLR, 2018.

Long history behind variational inference



Jordan, Ghahramani, Jaakkola, Saul. Introduction to variational methods for graphical models. *Machine Learning*, 1999.

Blei, Ng, Jordan. Latent Dirichlet allocation. Journal of Machine Learning Research, 2003.

Hoffman, Blei, Wang, Paisley. Stochastic variational inference. Journal of Machine Learning Research, 2013.
Ranganath, Gerrish, Blei. Black box variational inference. AISTATS, 2014.
Rezende and Mohamed. Variational inference with normalizing flows. ICML, 2016.
Kucukelbir, Tran, Ranganath, Gelman, Blei. Automatic differentiation variational inference. JMLR, 2018.

Long history behind variational inference

Elements of "classical" VI:
- Factorized (or "mean field") assumptions
- Coordinate-ascent and ELBO maximization
- Restrictive assumptions on target (long derivations)

- Stochastic optimization and SVI
- "Black-box" VI
- Amortized inference
- Alternative divergences
- More flexible variational families

importance sampling

???

pre-90s | 1990s | 2000s | 2010s | 2020s

Jordan, Ghahramani, Jaakkola, Saul. Introduction to variational methods for graphical models. *Machine Learning*, 1999.

Blei, Ng, Jordan. Latent Dirichlet allocation. Journal of Machine Learning Research, 2003.

Hoffman, Blei, Wang, Paisley. Stochastic variational inference. Journal of Machine Learning Research, 2013.
Ranganath, Gerrish, Blei. Black box variational inference. AISTATS, 2014.
Rezende and Mohamed. Variational inference with normalizing flows. ICML, 2016.
Kucukelbir, Tran, Ranganath, Gelman, Blei. Automatic differentiation variational inference. JMLR, 2018.

Long history behind variational inference

Elements of "classical" VI:
- Factorized (or "mean field") assumptions
- Coordinate-ascent and ELBO maximization
- Restrictive assumptions on target (long derivations)

- Stochastic optimization and SVI
- "Black-box" VI
- Amortized inference
- Alternative divergences
- More flexible variational families

importance sampling

???

pre-90s 1990s 2000s 2010s 2020s

This tutorial: "modern" variational inference
- "Black box" inference: few assumptions on target $p(z)$
- Automatic differentiation; supported in modern software
- Towards "expressive" variational families
- Alternative divergences

Jordan, Ghahramani, Jaakkola, Saul. Int

Blei, Ng, Jordan. Latent Dirichlet allocati

Hoffman, Blei, Wang, Paisley. Stochasti
Ranganath, Gerrish, Blei. Black box vari
Rezende and Mohamed. Variational inference with normalizing flows. ICML, 2016.
Kucukelbir, Tran, Ranganath, Gelman, Blei. Automatic differentiation variational inference. JMLR, 2018.

Part II: Key components of modern methods

Towards black-box approaches for VI

Classical approaches to VI used coordinate-ascent VI (CAVI) updates.

In order to make the problem tractable, typically additional assumptions imposed:

- The variational family is factorized (a “mean field” family)
- Structural assumptions about the target, e.g., “conditionally conjugate”

Challenging for practitioners to apply due to need for specialized derivations.

Towards black-box approaches for VI

Classical approaches to VI used coordinate-ascent VI (CAVI) updates.

In order to make the problem tractable, typically additional assumptions imposed:

- The variational family is factorized (a “mean field” family)
- Structural assumptions about the target, e.g., “conditionally conjugate”

Challenging for practitioners to apply due to need for specialized derivations.

“Black-box” VI approaches: very few assumptions needed about the target

- Bayesian modeling: usually just want the joint $p(z, x)$ to be differentiable
  - We will see algorithms that use $\nabla_z \log p(z \mid x)$ (“the score”)
- Autodiff tools: led to more user-friendly strategies for applying VI
- Implemented in many software packages

Overview of fundamental VI concepts

Variational Bayes: learn an approximation for $\pi(z) = p(z \mid x)$ for data x

latent variables z

Overview of fundamental VI concepts

Variational Bayes: learn an approximation for $\pi(z) = p(z \mid x)$ for data x

latent variables z

Gradient-based variational inference:

Overview of fundamental VI concepts

Variational Bayes: learn an approximation for $\pi(z) = p(z \mid x)$ for data x

latent variables z

Gradient-based variational inference:

Variational parameters ϕ

Overview of fundamental VI concepts

Variational Bayes: learn an approximation for $\pi(z) = p(z \mid x)$ for data x

latent variables z

Gradient-based variational inference:

Variational parameters ϕ

$$\text{ELBO}(\phi) = \int \log\left(\frac{p_\theta(z, x)}{q_\phi(z)}\right) q_\phi(z)\, dz$$

Overview of fundamental VI concepts

Variational Bayes: learn an approximation for $\pi(z) = p(z \mid x)$ for data x

latent variables z

Gradient-based variational inference:

Variational parameters ϕ

$$\text{ELBO}(\phi) = \int \log \left(\frac{p_\theta(z, x)}{q_\phi(z)} \right) q_\phi(z)\, dz$$

Joint distribution between z and x

Overview of fundamental VI concepts

Variational Bayes: learn an approximation for $\pi(z) = p(z \mid x)$ for data x

latent variables z

Gradient-based variational inference:

Variational parameters ϕ

$$\text{ELBO}(\phi) = \int \log \left(\frac{p_\theta(z, x)}{q_\phi(z)} \right) q_\phi(z)\, dz$$

Will need *gradients* of the ELBO: $\nabla_\phi \text{ELBO}(\phi)$

Joint distribution between z and x

Overview of fundamental VI concepts

Variational Bayes: learn an approximation for $\pi(z) = p(z \mid x)$ for data x

latent variables z

Gradient-based variational inference:

Variational parameters ϕ $\quad \text{ELBO}(\phi) = \int \log \left(\frac{p_\theta(z, x)}{q_\phi(z)} \right) q_\phi(z)\, dz$

Joint distribution between z and x

Will need *gradients* of the ELBO: $\nabla_\phi \text{ELBO}(\phi)$

What we will cover:

Overview of fundamental VI concepts

Variational Bayes: learn an approximation for $\pi(z) = p(z \mid x)$ for data x

latent variables z

Gradient-based variational inference:

Variational parameters ϕ $\quad \text{ELBO}(\phi) = \int \log \left(\frac{p_\theta(z, x)}{q_\phi(z)} \right) q_\phi(z)\, dz$

Will need *gradients* of the ELBO: $\nabla_\phi \text{ELBO}(\phi)$

Joint distribution between z and x

What we will cover:

Monte Carlo approximations of the gradient

Overview of fundamental VI concepts

Variational Bayes: learn an approximation for $\pi(z) = p(z \mid x)$ for data x

latent variables z

Gradient-based variational inference:

Variational parameters ϕ $\quad \text{ELBO}(\phi) = \int \log \left(\frac{p_\theta(z, x)}{q_\phi(z)} \right) q_\phi(z)\, dz$

Will need *gradients* of the ELBO: $\nabla_\phi \text{ELBO}(\phi)$

Joint distribution between z and x

What we will cover:

Monte Carlo approximations of the gradient

1) Monte Carlo approximation of $\nabla_\phi \text{ELBO}(\phi)$ via samples $z^1, \ldots, z^B \sim q_\phi$

Overview of fundamental VI concepts

Variational Bayes: learn an approximation for $\pi(z) = p(z \mid x)$ for data x

latent variables z

Gradient-based variational inference:

Variational parameters ϕ $\quad \text{ELBO}(\phi) = \int \log \left(\frac{p_\theta(z, x)}{q_\phi(z)} \right) q_\phi(z)\, dz$

Will need *gradients* of the ELBO: $\nabla_\phi \text{ELBO}(\phi)$

Joint distribution between z and x

What we will cover:

Monte Carlo approximations of the gradient

1) Monte Carlo approximation of $\nabla_\phi \text{ELBO}(\phi)$ via samples $z^1, \ldots, z^B \sim q_\phi$

Overview of fundamental VI concepts

Variational Bayes: learn an approximation for $\pi(z) = p(z \mid x)$ for data x

latent variables z

Gradient-based variational inference:

Variational parameters ϕ $\quad \text{ELBO}(\phi) = \int \log \left(\frac{p_\theta(z, x)}{q_\phi(z)} \right) q_\phi(z)\, dz$

Will need *gradients* of the ELBO: $\nabla_\phi \text{ELBO}(\phi)$

Joint distribution between z and x

What we will cover:

Monte Carlo approximations of the gradient

1) Monte Carlo approximation of $\nabla_\phi \text{ELBO}(\phi)$ via samples $z^1, \ldots, z^B \sim q_\phi$
2) Score function gradient estimator

Overview of fundamental VI concepts

Variational Bayes: learn an approximation for $\pi(z) = p(z \mid x)$ for data x

latent variables z

Gradient-based variational inference:

Variational parameters ϕ $\quad \text{ELBO}(\phi) = \int \log \left(\frac{p_\theta(z, x)}{q_\phi(z)} \right) q_\phi(z)\, dz$

Will need *gradients* of the ELBO: $\nabla_\phi \text{ELBO}(\phi)$

Joint distribution between z and x

What we will cover:

Monte Carlo approximations of the gradient

1) Monte Carlo approximation of $\nabla_\phi \text{ELBO}(\phi)$ via samples $z^1, \ldots, z^B \sim q_\phi$
2) Score function gradient estimator
3) Reparameterization gradient estimator

Overview of fundamental VI concepts

Variational Bayes: learn an approximation for $\pi(z) = p(z \mid x)$ for data x

latent variables z

Gradient-based variational inference:

Variational parameters ϕ $\quad \text{ELBO}(\phi) = \int \log \left(\frac{p_\theta(z, x)}{q_\phi(z)} \right) q_\phi(z)\, dz$

Will need *gradients* of the ELBO: $\nabla_\phi \text{ELBO}(\phi)$

Joint distribution between z and x

What we will cover:

Monte Carlo approximations of the gradient

1) Monte Carlo approximation of $\nabla_\phi \text{ELBO}(\phi)$ via samples $z^1, \ldots, z^B \sim q_\phi$
2) Score function gradient estimator
3) Reparameterization gradient estimator

Amortized inference: learn an approximation for $p(z \mid x)$ for general x
(uses ideas from VI + deep generative modeling)

General strategies for gradient estimation

Consider probabilistic objectives of the form:

$$\mathscr{L}(\phi) := \mathbb{E}_{q_\phi(z)}[f(z; \phi, \theta)]$$

Mohamed et al.. Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 2020.

General strategies for gradient estimation

Consider probabilistic objectives of the form:

$$\mathscr{L}(\phi) := \mathbb{E}_{q_\phi(z)}[f(z; \phi, \theta)]$$

Example: $\text{ELBO}(\phi) = \mathbb{E}_{q_\phi(z)}\left[\log\left(\frac{p_\theta(z, x)}{q_\phi(z)}\right)\right]$

Mohamed et al.. Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 2020.

General strategies for gradient estimation

Consider probabilistic objectives of the form:

$$\mathscr{L}(\phi) := \mathbb{E}_{q_\phi(z)}[f(z; \phi, \theta)]$$

Example: $\text{ELBO}(\phi) = \mathbb{E}_{q_\phi(z)}\left[\log\left(\frac{p_\theta(z, x)}{q_\phi(z)}\right)\right]$

Other applications: reinforcement learning, experimental design

Mohamed et al.. Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 2020.

General strategies for gradient estimation

Consider probabilistic objectives of the form:

$$\mathscr{L}(\phi) := \mathbb{E}_{q_\phi(z)}[f(z;\phi,\theta)]$$

Example: $\text{ELBO}(\phi) = \mathbb{E}_{q_\phi(z)}\left[\log\left(\frac{p_\theta(z,x)}{q_\phi(z)}\right)\right]$

Other applications: reinforcement learning, experimental design

Goal: learn the distributional parameters ϕ via gradient descent. Need gradients:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z;\phi,\theta)]$$

Mohamed et al.. Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 2020.

General strategies for gradient estimation

Consider probabilistic objectives of the form:

$$\mathscr{L}(\phi) := \mathbb{E}_{q_\phi(z)}[f(z;\phi,\theta)]$$

Example: $\text{ELBO}(\phi) = \mathbb{E}_{q_\phi(z)}\left[\log\left(\frac{p_\theta(z,x)}{q_\phi(z)}\right)\right]$

Other applications: reinforcement learning, experimental design

Goal: learn the distributional parameters ϕ via gradient descent. Need gradients:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z;\phi,\theta)]$$

But: typically this expectation is intractable!

Mohamed et al.. Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 2020.

General strategies for gradient estimation

Consider probabilistic objectives of the form:

$$\mathscr{L}(\phi) := \mathbb{E}_{q_\phi(z)}[f(z;\phi,\theta)]$$

Example: $\text{ELBO}(\phi) = \mathbb{E}_{q_\phi(z)}\left[\log\left(\frac{p_\theta(z,x)}{q_\phi(z)}\right)\right]$

Other applications: reinforcement learning, experimental design

Goal: learn the distributional parameters ϕ via gradient descent. Need gradients:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z;\phi,\theta)]$$

But: typically this expectation is intractable!

Two strategies that appear frequently in approximate inference:

1) Score function gradient estimator
2) Reparameterization gradient estimator

Mohamed et al.. Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 2020.

General strategies for gradient estimation

Consider probabilistic objectives of the form:

$$\mathscr{L}(\phi) := \mathbb{E}_{q_\phi(z)}[f(z;\phi,\theta)]$$

Example: $\text{ELBO}(\phi) = \mathbb{E}_{q_\phi(z)}\left[\log\left(\frac{p_\theta(z,x)}{q_\phi(z)}\right)\right]$

Other applications: reinforcement learning, experimental design

Goal: learn the distributional parameters ϕ via gradient descent. Need gradients:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z;\phi,\theta)]$$

But: typically this expectation is intractable!

Two strategies that appear frequently in approximate inference:

1) Score function gradient estimator
2) Reparameterization gradient estimator

We will also discuss two “black-box” strategies for VI based on these two estimators

Mohamed et al.. Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 2020.

Monte Carlo estimates

$$\mathscr{L}(\phi) := \mathbb{E}_{q_\phi(z)}[f(z;\phi,\theta)]$$

Example: $\text{ELBO}(\phi) = \mathbb{E}_{q_\phi(z)}\left[\log\left(\frac{p_\theta(z,x)}{q_\phi(z)}\right)\right]$

Monte Carlo estimates

$$\mathscr{L}(\phi) := \mathbb{E}_{q_\phi(z)}[f(z;\phi,\theta)]$$

Example: $\text{ELBO}(\phi) = \mathbb{E}_{q_\phi(z)}\left[\log\left(\frac{p_\theta(z,x)}{q_\phi(z)}\right)\right]$

Monte Carlo estimate: sample $z^b \sim q_\phi(z)$

Monte Carlo estimates

$$\mathscr{L}(\phi) := \mathbb{E}_{q_\phi(z)}[f(z;\phi,\theta)]$$

Example: $\text{ELBO}(\phi) = \mathbb{E}_{q_\phi(z)}\left[\log\left(\frac{p_\theta(z,x)}{q_\phi(z)}\right)\right]$

Monte Carlo estimate: sample $z^b \sim q_\phi(z)$

$$\mathscr{L}(\phi) \approx \frac{1}{B}\sum_{b=1}^{B} f(z^b;\phi,\theta)$$

Monte Carlo estimates

$$\mathscr{L}(\phi) := \mathbb{E}_{q_\phi(z)}[f(z;\phi,\theta)]$$

Example: $\text{ELBO}(\phi) = \mathbb{E}_{q_\phi(z)}\left[\log\left(\frac{p_\theta(z,x)}{q_\phi(z)}\right)\right]$

Monte Carlo estimate: sample $z^b \sim q_\phi(z)$

$$\mathscr{L}(\phi) \approx \frac{1}{B}\sum_{b=1}^{B} f(z^b;\phi,\theta)$$

How to estimate the gradient $\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z;\phi,\theta)]$ using Monte Carlo?

Monte Carlo estimates

$$\mathscr{L}(\phi) := \mathbb{E}_{q_\phi(z)}[f(z;\phi,\theta)]$$

Example: $\text{ELBO}(\phi) = \mathbb{E}_{q_\phi(z)}\left[\log\left(\frac{p_\theta(z,x)}{q_\phi(z)}\right)\right]$

Monte Carlo estimate: sample $z^b \sim q_\phi(z)$

$$\mathscr{L}(\phi) \approx \frac{1}{B}\sum_{b=1}^{B} f(z^b;\phi,\theta)$$

How to estimate the gradient $\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z;\phi,\theta)]$ using Monte Carlo?

Desirable properties of estimators:

- Unbiased: E[estimator] = true value
- Consistent: recover true value as $B \to \infty$
- Low variance: if we have two estimators that use the same amount of computation, we prefer the lower variance once
- Fast computation

Approach 1: Score function gradients

a.k.a. likelihood ratio method, REINFORCE estimator, “log derivative trick”

Approach 1: Score function gradients

a.k.a. likelihood ratio method, REINFORCE estimator, “log derivative trick”

Score function: (a.k.a. “Fisher” score)

$$\nabla_\phi \log q_\phi(z) = \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}$$

Approach 1: Score function gradients

a.k.a. likelihood ratio method, REINFORCE estimator, “log derivative trick”

Score function: (a.k.a. “Fisher” score)

$$\nabla_\phi \log q_\phi(z) = \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}$$

(Note: NOT the “Stein” score function $\nabla_z \log q_\phi(z)$)

Approach 1: Score function gradients

a.k.a. likelihood ratio method, REINFORCE estimator, “log derivative trick”

Score function: (a.k.a. “Fisher” score)

$$\nabla_\phi \log q_\phi(z) = \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}$$

(Note: NOT the “Stein” score function $\nabla_z \log q_\phi(z)$)

Gradient of the expectation:

Approach 1: Score function gradients

a.k.a. likelihood ratio method, REINFORCE estimator, “log derivative trick”

Score function: (a.k.a. “Fisher” score)

$$\nabla_\phi \log q_\phi(z) = \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}$$

(Note: NOT the “Stein” score function $\nabla_z \log q_\phi(z)$)

Gradient of the expectation:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \nabla_\phi \int q_\phi(z)\, f(z) dz$$

Approach 1: Score function gradients

a.k.a. likelihood ratio method, REINFORCE estimator, “log derivative trick”

Score function: (a.k.a. “Fisher” score)

$$\nabla_\phi \log q_\phi(z) = \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}$$

(Note: NOT the “Stein” score function $\nabla_z \log q_\phi(z)$)

Gradient of the expectation:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \nabla_\phi \int q_\phi(z)\, f(z) dz$$

Assume you can swap derivatives and integrals

Approach 1: Score function gradients

a.k.a. likelihood ratio method, REINFORCE estimator, “log derivative trick”

Score function: (a.k.a. “Fisher” score)

$$\nabla_\phi \log q_\phi(z) = \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}$$

(Note: NOT the “Stein” score function $\nabla_z \log q_\phi(z)$)

Gradient of the expectation:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \nabla_\phi \int q_\phi(z)\, f(z) dz$$

$$= \int \nabla_\phi q_\phi(z) f(z)$$

Assume you can swap derivatives and integrals

Approach 1: Score function gradients

a.k.a. likelihood ratio method, REINFORCE estimator, “log derivative trick”

Score function: (a.k.a. “Fisher” score)

$$\nabla_\phi \log q_\phi(z) = \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}$$

(Note: NOT the “Stein” score function $\nabla_z \log q_\phi(z)$)

Gradient of the expectation:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \nabla_\phi \int q_\phi(z)\, f(z) dz$$

$$= \int \nabla_\phi q_\phi(z) f(z)$$

Assume you can swap derivatives and integrals

Approach 1: Score function gradients

a.k.a. likelihood ratio method, REINFORCE estimator, “log derivative trick”

Score function: (a.k.a. “Fisher” score)

$$\nabla_\phi \log q_\phi(z) = \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}$$

(Note: NOT the “Stein” score function $\nabla_z \log q_\phi(z)$)

Gradient of the expectation:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \nabla_\phi \int q_\phi(z)\, f(z) dz$$

$$= \int \nabla_\phi q_\phi(z) f(z)$$

Assume you can swap derivatives and integrals

“Log derivative trick”

$$\nabla_\phi q_\phi(z) = q_\phi(z)\, \nabla_\phi \log q_\phi(z)$$

Approach 1: Score function gradients

a.k.a. likelihood ratio method, REINFORCE estimator, “log derivative trick”

Score function: (a.k.a. “Fisher” score)

$$\nabla_\phi \log q_\phi(z) = \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}$$

(Note: NOT the “Stein” score function $\nabla_z \log q_\phi(z)$)

Gradient of the expectation:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \nabla_\phi \int q_\phi(z)\, f(z) dz$$

$$= \int \nabla_\phi q_\phi(z) f(z)$$

Assume you can swap derivatives and integrals

“Log derivative trick”

$$\nabla_\phi q_\phi(z) = q_\phi(z)\, \nabla_\phi \log q_\phi(z)$$

$$= \int q_\phi(z) f(z)\, \nabla_\phi \log q_\phi(z)\, dz$$

Approach 1: Score function gradients

a.k.a. likelihood ratio method, REINFORCE estimator, “log derivative trick”

Score function: (a.k.a. “Fisher” score)

$$\nabla_\phi \log q_\phi(z) = \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}$$

(Note: NOT the “Stein” score function $\nabla_z \log q_\phi(z)$)

Gradient of the expectation:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \nabla_\phi \int q_\phi(z)\, f(z) dz$$

$$= \int \nabla_\phi q_\phi(z) f(z)$$

Assume you can swap derivatives and integrals

“Log derivative trick”

$$\nabla_\phi q_\phi(z) = q_\phi(z)\, \nabla_\phi \log q_\phi(z)$$

$$= \int q_\phi(z) f(z)\, \nabla_\phi \log q_\phi(z)\, dz$$

Approach 1: Score function gradients

a.k.a. likelihood ratio method, REINFORCE estimator, “log derivative trick”

Score function: (a.k.a. “Fisher” score)

$$\nabla_\phi \log q_\phi(z) = \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}$$

(Note: NOT the “Stein” score function $\nabla_z \log q_\phi(z)$)

Gradient of the expectation:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \nabla_\phi \int q_\phi(z)\, f(z) dz$$

$$= \int \nabla_\phi q_\phi(z) f(z)$$

Assume you can swap derivatives and integrals

“Log derivative trick”

$$\nabla_\phi q_\phi(z) = q_\phi(z)\, \nabla_\phi \log q_\phi(z)$$

$$= \int q_\phi(z) f(z)\, \nabla_\phi \log q_\phi(z)\, dz$$

Sample $z^b \sim q_\phi(z)$

Approach 1: Score function gradients

a.k.a. likelihood ratio method, REINFORCE estimator, “log derivative trick”

Score function: (a.k.a. “Fisher” score)

$$\nabla_\phi \log q_\phi(z) = \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}$$

(Note: NOT the “Stein” score function $\nabla_z \log q_\phi(z)$)

Gradient of the expectation:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \nabla_\phi \int q_\phi(z)\, f(z) dz$$

$$= \int \nabla_\phi q_\phi(z) f(z)$$

Assume you can swap derivatives and integrals

“Log derivative trick”

$\nabla_\phi q_\phi(z) = q_\phi(z)\, \nabla_\phi \log q_\phi(z)$

$$= \int q_\phi(z) f(z)\, \nabla_\phi \log q_\phi(z)\, dz$$

Sample $z^b \sim q_\phi(z)$

$$\approx \frac{1}{B} \sum_{b=1}^{B} \nabla_\phi \log q_\phi(z^b)\, f(z^b)$$

Approach 1: Score function gradients

a.k.a. likelihood ratio method, REINFORCE estimator, “log derivative trick”

Score function: (a.k.a. “Fisher” score)

$$\nabla_\phi \log q_\phi(z) = \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}$$

(Note: NOT the “Stein” score function $\nabla_z \log q_\phi(z)$)

Gradient of the expectation:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \nabla_\phi \int q_\phi(z)\, f(z) dz$$

$$= \int \nabla_\phi q_\phi(z) f(z)$$

Assume you can swap derivatives and integrals

“Log derivative trick”

$\nabla_\phi q_\phi(z) = q_\phi(z)\, \nabla_\phi \log q_\phi(z)$

$$= \int q_\phi(z) f(z)\, \nabla_\phi \log q_\phi(z)\, dz$$

Sample $z^b \sim q_\phi(z)$

$$\approx \frac{1}{B} \sum_{b=1}^{B} \nabla_\phi \log q_\phi(z^b)\, f(z^b)$$

Variance reduction (“control variate”)

$[f(z^b) - \beta]$

Approach 2: reparameterization gradient estimator

a.k.a. pathwise gradient estimator

Sampling paths: suppose we can alternatively generate samples as follows:

$$\hat{z} \sim q_\phi(z;\phi) \;\equiv\; \hat{z} = g(\hat{\epsilon};\phi), \quad \hat{\epsilon} \sim p(\epsilon)$$

Approach 2: reparameterization gradient estimator

a.k.a. pathwise gradient estimator

Sampling paths: suppose we can alternatively generate samples as follows:

$$\hat{z} \sim q_\phi(z;\phi) \;\equiv\; \hat{z} = g(\hat{\epsilon};\phi), \quad \hat{\epsilon} \sim p(\epsilon)$$

deterministic path

Approach 2: reparameterization gradient estimator

a.k.a. pathwise gradient estimator

Sampling paths: suppose we can alternatively generate samples as follows:

$$\hat{z} \sim q_\phi(z;\phi) \;\equiv\; \hat{z} = g(\hat{\epsilon};\phi), \quad \hat{\epsilon} \sim p(\epsilon)$$

deterministic path

base distribution that is independent of ϕ

Approach 2: reparameterization gradient estimator

a.k.a. pathwise gradient estimator

Sampling paths: suppose we can alternatively generate samples as follows:

$$\hat{z} \sim q_\phi(z;\phi) \;\equiv\; \hat{z} = g(\hat{\epsilon};\phi), \quad \hat{\epsilon} \sim p(\epsilon)$$

deterministic path — base distribution that is independent of ϕ

Example: Multivariate Gaussian $p(z;\phi) = \mathcal{N}(z \,|\, \mu, \Sigma)$,

base is $p(\epsilon) = \mathcal{N}(0, I)$; $\quad g(\epsilon;\theta) = \mu + L\epsilon$, where $\Sigma = LL^\top$

Approach 2: reparameterization gradient estimator

a.k.a. pathwise gradient estimator

Sampling paths: suppose we can alternatively generate samples as follows:

$$\hat{z} \sim q_\phi(z;\phi) \equiv \hat{z} = g(\hat{\epsilon};\phi), \quad \hat{\epsilon} \sim p(\epsilon)$$

deterministic path — base distribution that is independent of ϕ

Example: Multivariate Gaussian $p(z;\phi) = \mathcal{N}(z \mid \mu, \Sigma)$,

base is $p(\epsilon) = \mathcal{N}(0, I)$; $g(\epsilon;\theta) = \mu + L\epsilon$, where $\Sigma = LL^\top$

"Reparameterization trick": $\mathbb{E}_{q_\phi(z)}[f(z)] = \mathbb{E}_{p(\epsilon)}[f(g(\epsilon;\phi))]$

Approach 2: reparameterization gradient estimator

a.k.a. pathwise gradient estimator

Sampling paths: suppose we can alternatively generate samples as follows:

$$\hat{z} \sim q_\phi(z;\phi) \;\equiv\; \hat{z} = g(\hat{\epsilon};\phi), \quad \hat{\epsilon} \sim p(\epsilon)$$

deterministic path — base distribution that is independent of ϕ

Example: Multivariate Gaussian $p(z;\phi) = \mathcal{N}(z \,|\, \mu, \Sigma)$,

base is $p(\epsilon) = \mathcal{N}(0, I)$; $\quad g(\epsilon;\theta) = \mu + L\epsilon$, where $\Sigma = LL^\top$

"Reparameterization trick": $\mathbb{E}_{q_\phi(z)}[f(z)] = \mathbb{E}_{p(\epsilon)}[f(g(\epsilon;\phi))]$

Application to gradient estimator:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z;\phi)]$$

Approach 2: reparameterization gradient estimator

a.k.a. pathwise gradient estimator

Sampling paths: suppose we can alternatively generate samples as follows:

$$\hat{z} \sim q_\phi(z;\phi) \;\equiv\; \hat{z} = g(\hat{\epsilon};\phi), \quad \hat{\epsilon} \sim p(\epsilon)$$

deterministic path — base distribution that is independent of ϕ

Example: Multivariate Gaussian $p(z;\phi) = \mathcal{N}(z \mid \mu, \Sigma)$,

base is $p(\epsilon) = \mathcal{N}(0, I)$; $\quad g(\epsilon;\theta) = \mu + L\epsilon$, where $\Sigma = LL^\top$

"Reparameterization trick": $\mathbb{E}_{q_\phi(z)}[f(z)] = \mathbb{E}_{p(\epsilon)}[f(g(\epsilon;\phi))]$

Application to gradient estimator:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z;\phi)] = \mathbb{E}_{p(\epsilon)}[\nabla_\phi f(g(\epsilon;\phi))]$$

Approach 2: reparameterization gradient estimator

a.k.a. pathwise gradient estimator

Sampling paths: suppose we can alternatively generate samples as follows:

$$\hat{z} \sim q_\phi(z;\phi) \;\equiv\; \hat{z} = g(\hat{\epsilon};\phi), \quad \hat{\epsilon} \sim p(\epsilon)$$

deterministic path — base distribution that is independent of ϕ

Example: Multivariate Gaussian $p(z;\phi) = \mathcal{N}(z \,|\, \mu, \Sigma)$,

base is $p(\epsilon) = \mathcal{N}(0, I)$; $\quad g(\epsilon;\theta) = \mu + L\epsilon$, where $\Sigma = LL^\top$

"Reparameterization trick": $\mathbb{E}_{q_\phi(z)}[f(z)] = \mathbb{E}_{p(\epsilon)}[f(g(\epsilon;\phi))]$

Application to gradient estimator:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z;\phi)] = \mathbb{E}_{p(\epsilon)}[\nabla_\phi f(g(\epsilon;\phi))]$$

Draw samples $\hat{\epsilon}^b \sim p(\epsilon)$

Approach 2: reparameterization gradient estimator

a.k.a. pathwise gradient estimator

Sampling paths: suppose we can alternatively generate samples as follows:

$$\hat{z} \sim q_\phi(z;\phi) \equiv \hat{z} = g(\hat{\epsilon};\phi), \quad \hat{\epsilon} \sim p(\epsilon)$$

deterministic path — base distribution that is independent of ϕ

Example: Multivariate Gaussian $p(z;\phi) = \mathcal{N}(z \mid \mu, \Sigma)$,

base is $p(\epsilon) = \mathcal{N}(0, I)$; $g(\epsilon;\theta) = \mu + L\epsilon$, where $\Sigma = LL^\top$

"Reparameterization trick": $\mathbb{E}_{q_\phi(z)}[f(z)] = \mathbb{E}_{p(\epsilon)}[f(g(\epsilon;\phi))]$

Application to gradient estimator:

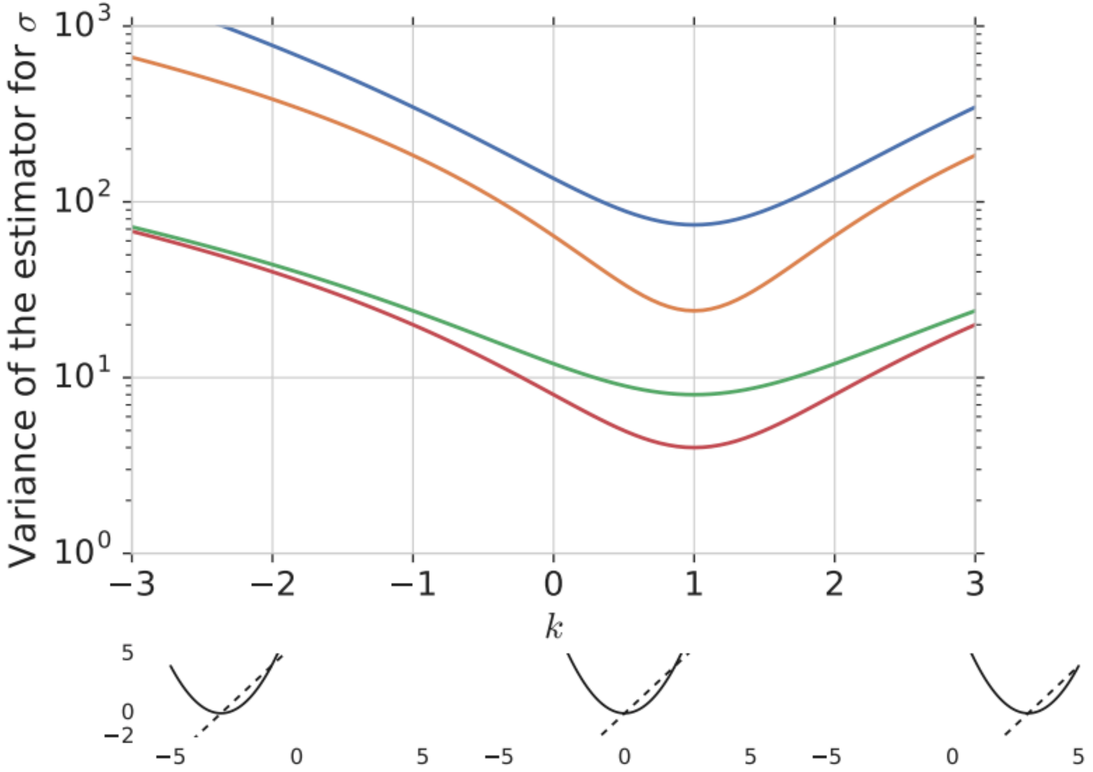
$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z;\phi)] = \mathbb{E}_{p(\epsilon)}[\nabla_\phi f(g(\epsilon;\phi))]$$

Draw samples $\hat{\epsilon}^b \sim p(\epsilon)$

$$\approx \frac{1}{B}\sum_{b=1}^{B} \nabla_\phi f(g(\hat{\epsilon}^b;\phi))$$

Comparison of estimators

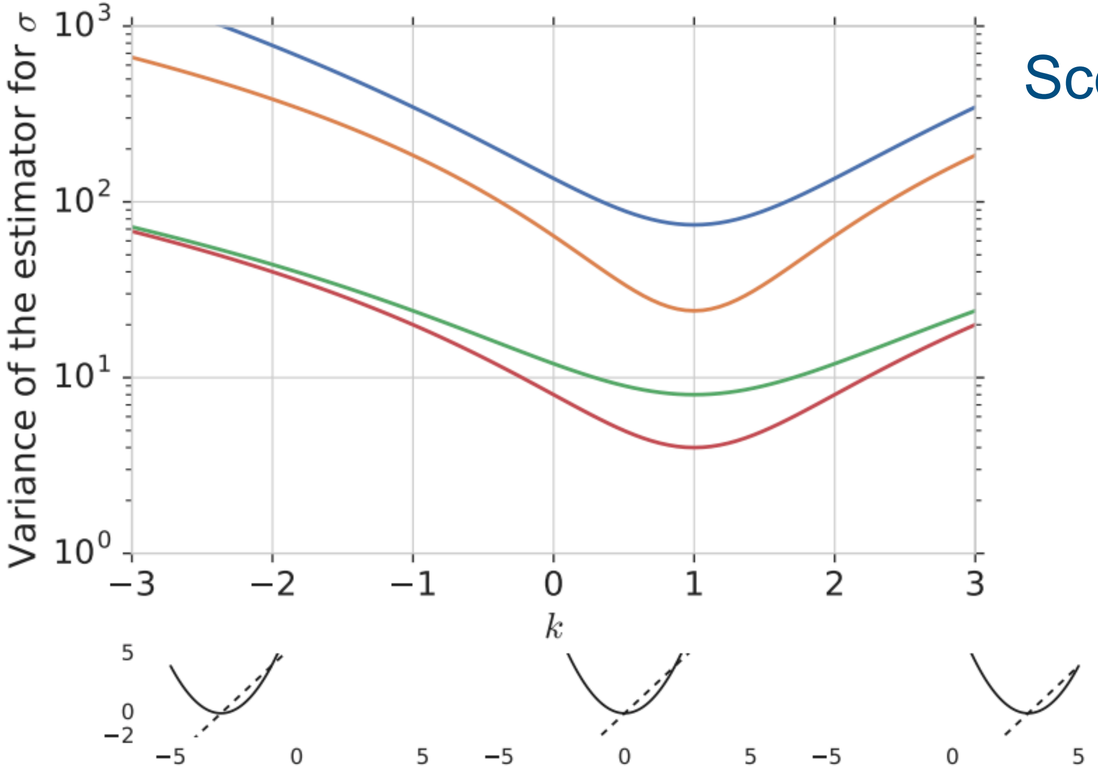
$$q(z;\phi) = \mathcal{N}(z\,|\,\mu,\sigma^2) \qquad \phi \in \{\mu,\sigma\} \qquad f(z) = (z-k)^2$$



Mohamed et al. Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 2020.

Comparison of estimators

$$q(z;\phi) = \mathcal{N}(z\,|\,\mu,\sigma^2) \qquad \phi \in \{\mu,\sigma\} \qquad f(z) = (z-k)^2$$

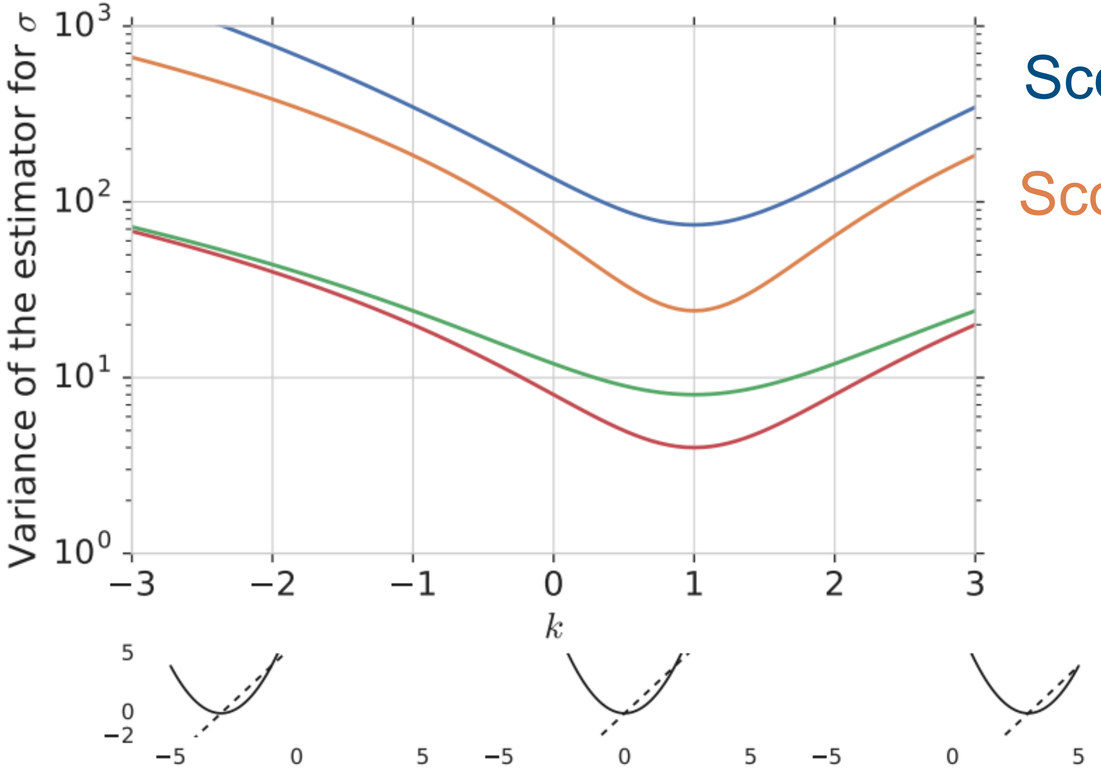


Score function

Mohamed et al. Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 2020.

Comparison of estimators

$$q(z;\phi) = \mathcal{N}(z\,|\,\mu,\sigma^2) \qquad \phi \in \{\mu,\sigma\} \qquad f(z) = (z-k)^2$$



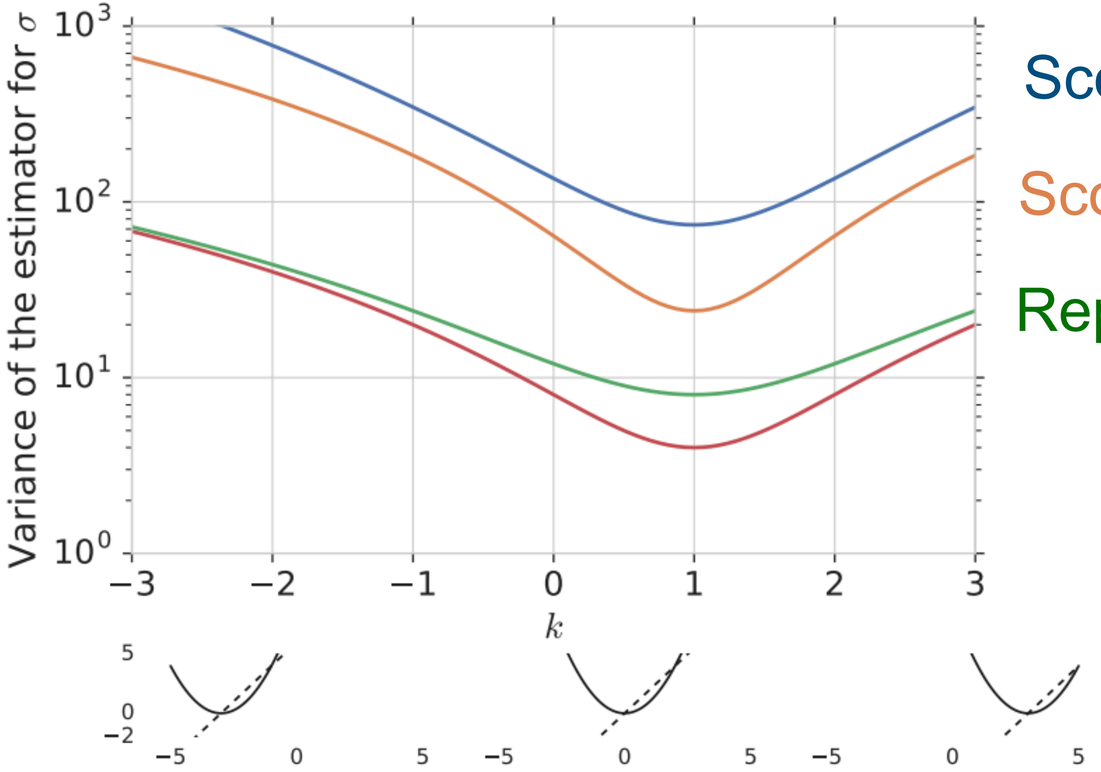
Score function

Score + variance reduction

Mohamed et al. Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 2020.

Comparison of estimators

$$q(z;\phi) = \mathcal{N}(z\,|\,\mu,\sigma^2) \qquad \phi \in \{\mu,\sigma\} \qquad f(z) = (z-k)^2$$



Score function

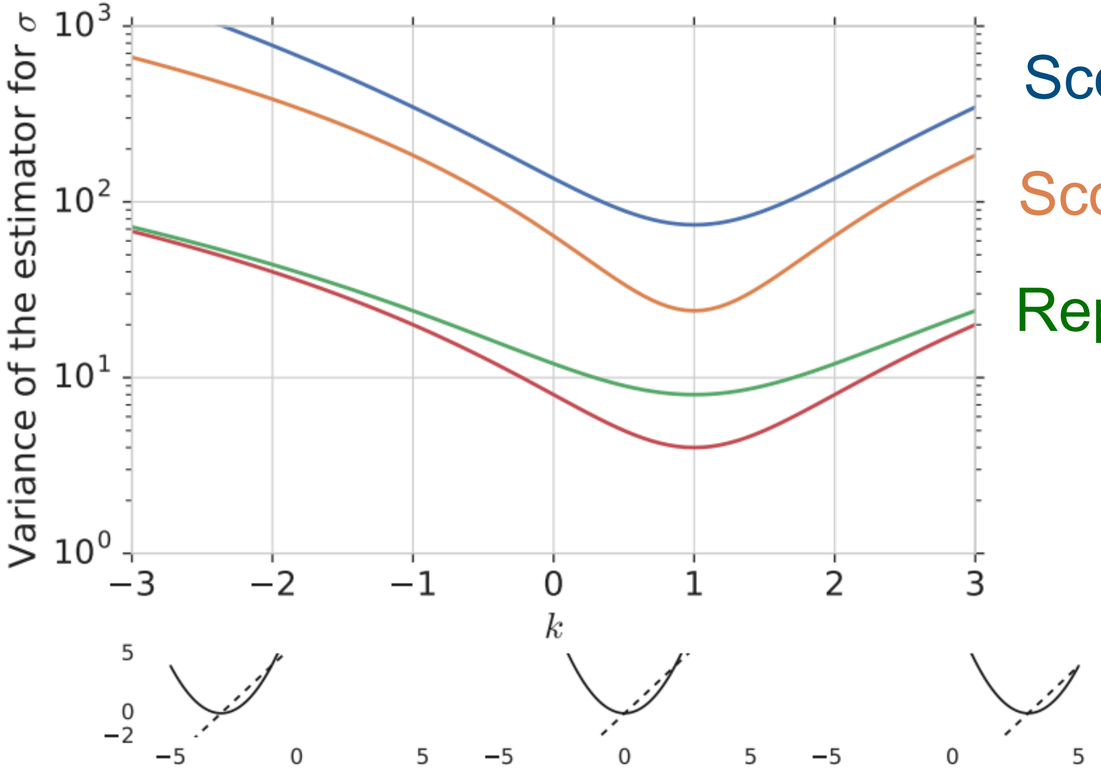
Score + variance reduction

Reparameterization

Mohamed et al. Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 2020.

Comparison of estimators

$$q(z;\phi) = \mathcal{N}(z\,|\,\mu,\sigma^2) \qquad \phi \in \{\mu,\sigma\} \qquad f(z) = (z-k)^2$$



Score function

Score + variance reduction

Reparameterization

Score function: more general; more sensitive to dimensionality; generally higher variance
Reparameterization: needs a differentiable cost function; variance more well behaved

Mohamed et al. Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 2020.

Black-box VI

Uses the score function gradient estimator of the ELBO

Sample $z^1, \ldots, z^B \sim q_\phi(z)$

Set the step size η

Gradient update

$$\phi \leftarrow \phi + \eta \frac{1}{B} \sum_{b=1}^{B} \nabla_\phi \log q_\phi(z^b) \log\left(\frac{p(z^b, x)}{q_\phi(z^b)}\right)$$

Ranganath, Gerrish, Blei. Black box variational inference. *AISTATS*, 2014.

Black-box VI

Uses the score function gradient estimator of the ELBO

Sample $z^1, \ldots, z^B \sim q_\phi(z)$

Set the step size η

Gradient update

$$\phi \leftarrow \phi + \eta \frac{1}{B} \sum_{b=1}^{B} \nabla_\phi \log q_\phi(z^b) \log\left(\frac{p(z^b, x)}{q_\phi(z^b)}\right)$$

score function gradient estimator of the ELBO

Ranganath, Gerrish, Blei. Black box variational inference. *AISTATS*, 2014.

Black-box VI

Uses the score function gradient estimator of the ELBO

Sample $z^1, \ldots, z^B \sim q_\phi(z)$

Set the step size η

Gradient update

$$\phi \leftarrow \phi + \eta \frac{1}{B} \sum_{b=1}^{B} \nabla_\phi \log q_\phi(z^b) \log\left(\frac{p(z^b, x)}{q_\phi(z^b)}\right)$$

score function gradient estimator of the ELBO

"Black box": few assumptions are needed on $p(z, x)$

- Sample from q
- Evaluate score
- Evaluate joint

Ranganath, Gerrish, Blei. Black box variational inference. *AISTATS*, 2014.

Black-box VI

Uses the score function gradient estimator of the ELBO

Sample $z^1, \ldots, z^B \sim q_\phi(z)$

Set the step size η

Gradient update

$$\phi \leftarrow \phi + \eta \frac{1}{B} \sum_{b=1}^{B} \nabla_\phi \log q_\phi(z^b) \log\left(\frac{p(z^b, x)}{q_\phi(z^b)}\right)$$

score function gradient estimator of the ELBO

"Black box": few assumptions are needed on $p(z, x)$

- Sample from q
- Evaluate score
- Evaluate joint

Black-box VI is used with techniques to reduce the variance of the estimator (Rao-Blackwellization and control variates)

Ranganath, Gerrish, Blei. Black box variational inference. *AISTATS*, 2014.

Black-box VI

Uses the score function gradient estimator of the ELBO

Sample $z^1, \ldots, z^B \sim q_\phi(z)$

Set the step size η

Gradient update

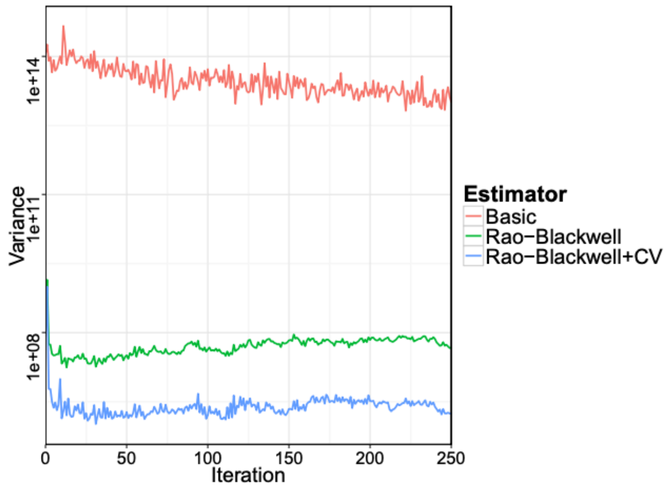
$$\phi \leftarrow \phi + \eta \frac{1}{B} \sum_{b=1}^{B} \nabla_\phi \log q_\phi(z^b) \log\left(\frac{p(z^b, x)}{q_\phi(z^b)}\right)$$

score function gradient estimator of the ELBO

"Black box": few assumptions are needed on $p(z, x)$

- Sample from q
- Evaluate score
- Evaluate joint

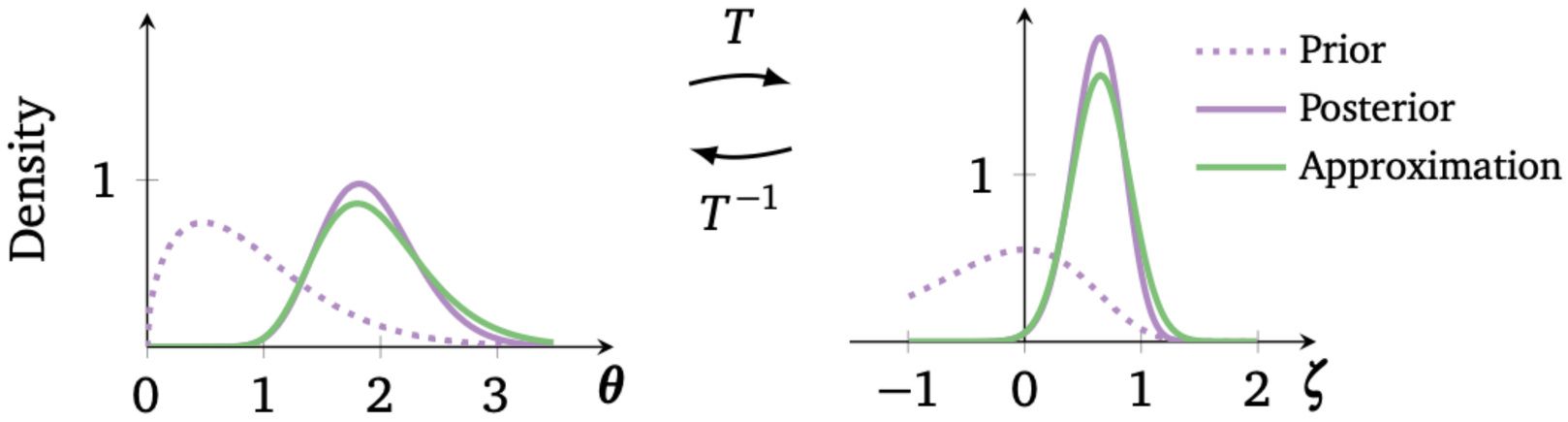
Black-box VI is used with techniques to reduce the variance of the estimator (Rao-Blackwellization and control variates)



Ranganath, Gerrish, Blei. Black box variational inference. *AISTATS*, 2014.

Automatic differentiation VI (ADVI)

Transform target π to one on real coordinate space $\tilde{\pi}$ (via change of variables formula)

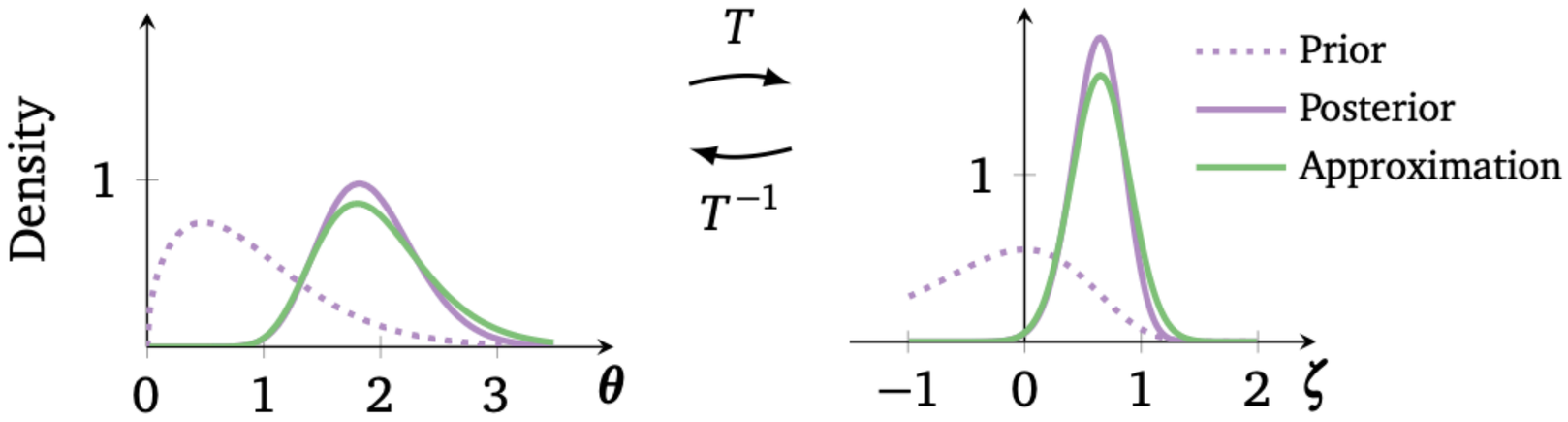


(a) Latent variable space **(b)** Real coordinate space

Kucukelbir, et al. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 2018.

Automatic differentiation VI (ADVI)

Transform target π to one on real coordinate space $\tilde{\pi}$ (via change of variables formula)



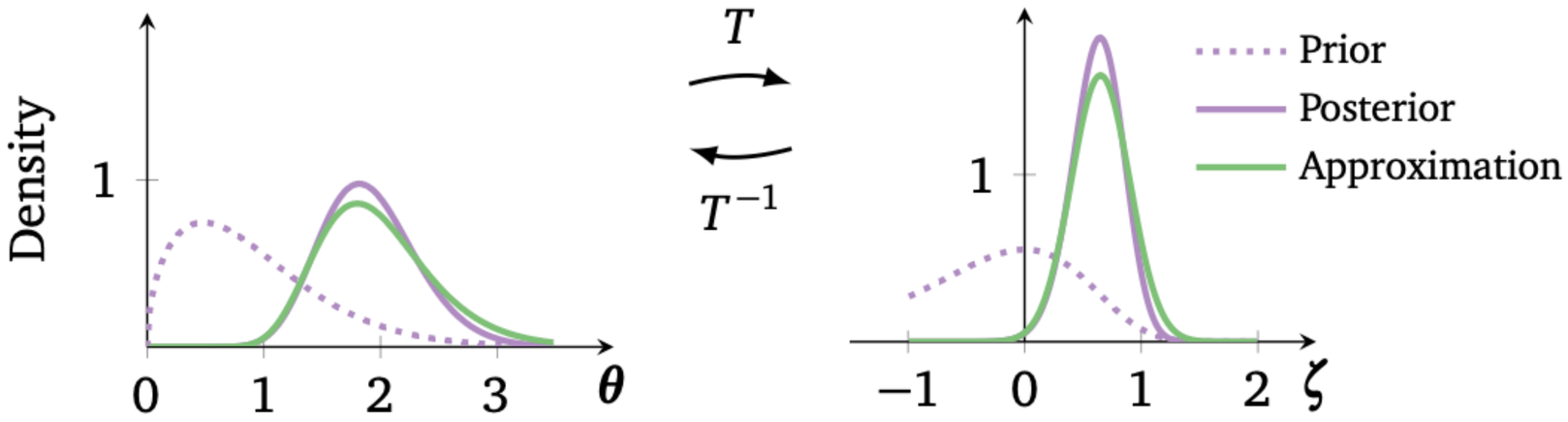
(a) Latent variable space **(b)** Real coordinate space

Uses the reparameterization gradient estimator of the ELBO:

Kucukelbir, et al. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 2018.

Automatic differentiation VI (ADVI)

Transform target π to one on real coordinate space $\tilde{\pi}$ (via change of variables formula)



(a) Latent variable space **(b)** Real coordinate space

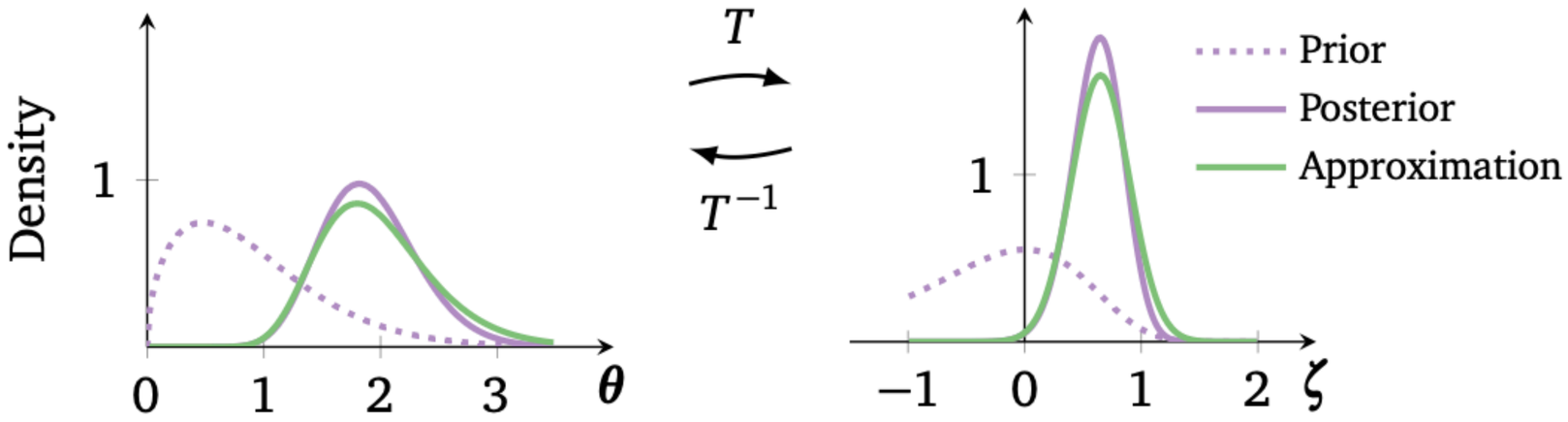
Uses the reparameterization gradient estimator of the ELBO:

Variational family is a Gaussian with μ and Σ (diagonal or full-covariance)

Kucukelbir, et al. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 2018.

Automatic differentiation VI (ADVI)

Transform target π to one on real coordinate space $\tilde{\pi}$ (via change of variables formula)



(a) Latent variable space **(b)** Real coordinate space

Uses the reparameterization gradient estimator of the ELBO:

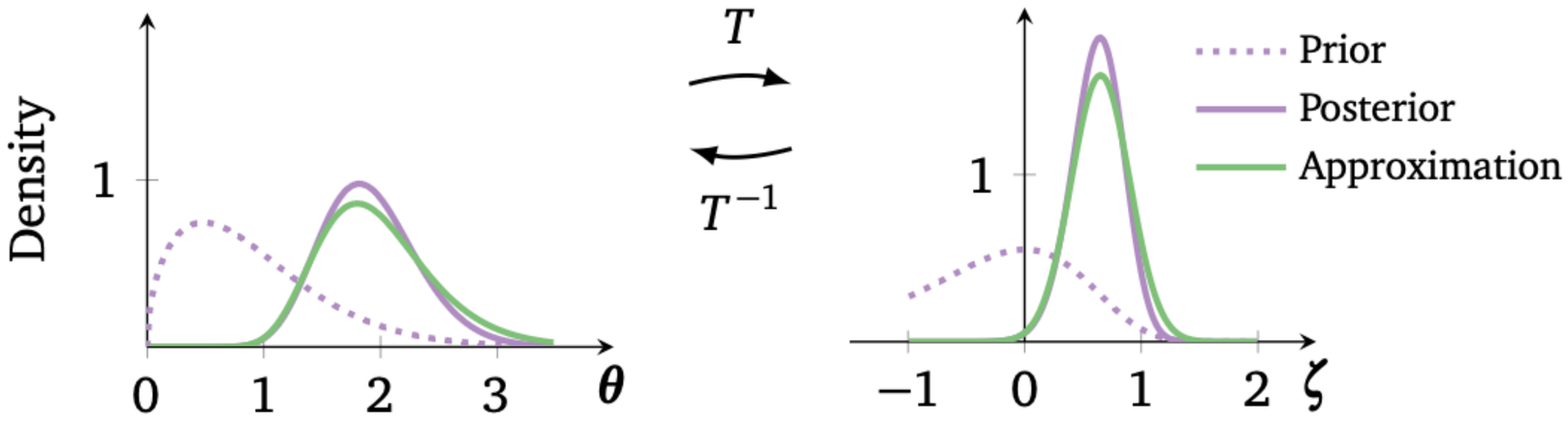
Variational family is a Gaussian with μ and Σ (diagonal or full-covariance)

Base distribution is $p(\epsilon) = \mathcal{N}(0, I)$ and e.g., $g(\epsilon; \theta) = \mu + L\epsilon$, where $LL^\top = \Sigma$

Kucukelbir, et al. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 2018.

Automatic differentiation VI (ADVI)

Transform target π to one on real coordinate space $\tilde{\pi}$ (via change of variables formula)



(a) Latent variable space **(b)** Real coordinate space

Uses the reparameterization gradient estimator of the ELBO:

Variational family is a Gaussian with μ and Σ (diagonal or full-covariance)

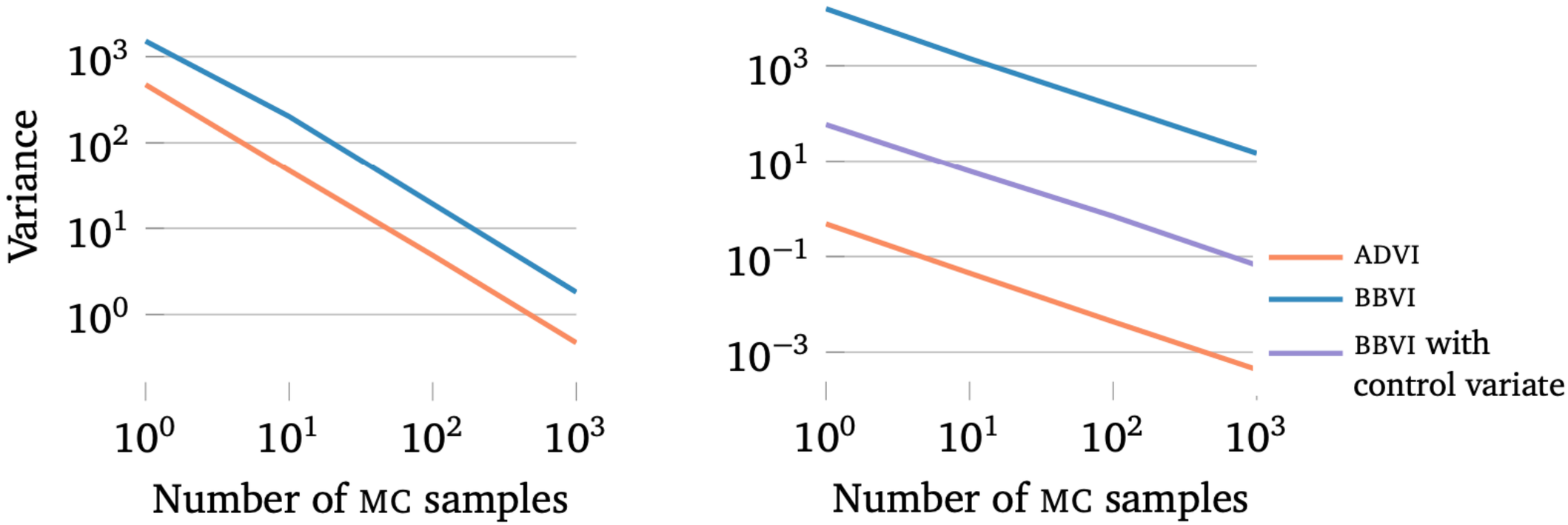
Base distribution is $p(\epsilon) = \mathcal{N}(0, I)$ and e.g., $g(\epsilon; \theta) = \mu + L\epsilon$, where $LL^\top = \Sigma$

Requires that the log joint distribution is differentiable (i.e., score $\nabla_\phi \log p(z, x)$) .

Kucukelbir, et al. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 2018.

Comparison of BBVI vs ADVI variance

score function gradient vs reparameterization gradient



Kucukelbir, Tran, Ranganath, Gelman, Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 2018.

Summary: score function vs reparameterization gradients

Score function gradient estimator

- More general: doesn't need differentiable model $\rho(z) = p(z, x)$
- Applies to continuous and discrete distributions
- High variance: needs to be used with variance reduction techniques

Reparameterization gradients

- Requires differentiable models
- Needs to be "reparameterizable": i.e., $z = g(\hat{\epsilon}, \phi)$
- Generally controlled variance (less sensitive to dimension)

Software implementations & other advances

Black-box VI and ADVI: Implemented in many software systems

Software implementations & other advances

Black-box VI and ADVI: Implemented in many software systems

Probabilistic programming software: e.g., Stan, Pyro, Turing, PyMC

Software implementations & other advances

Black-box VI and ADVI: Implemented in many software systems

Probabilistic programming software: e.g., Stan, Pyro, Turing, PyMC

Deterministic ADVI (DADVI): swaps differentiation and sampling; lower-variance, improved stability; alternative solvers

Giordano, et al. Black Box Variational Inference with a Deterministic Objective: Faster, More Accurate, and Even More Black Box. *JMLR* 2024.

Software implementations & other advances

Black-box VI and ADVI: Implemented in many software systems

Probabilistic programming software: e.g., Stan, Pyro, Turing, PyMC

Deterministic ADVI (DADVI): swaps differentiation and sampling; lower-variance, improved stability; alternative solvers

Implicit reparameterization: based on implicit differentiation;

- allows reparameterization to be applied to a broader class of distributions, e.g., beta, gamma, and Dirichlet
- implemented in, e.g., PyTorch, JAX (numpyro, distrax)

Giordano, et al. Black Box Variational Inference with a Deterministic Objective: Faster, More Accurate, and Even More Black Box. *JMLR* 2024.

Figurnov et al. Implicit reparameterization gradients. *NeurIPS* 2018.

Software implementations & other advances

Black-box VI and ADVI: Implemented in many software systems

Probabilistic programming software: e.g., Stan, Pyro, Turing, PyMC

Deterministic ADVI (DADVI): swaps differentiation and sampling; lower-variance, improved stability; alternative solvers

Implicit reparameterization: based on implicit differentiation;

- allows reparameterization to be applied to a broader class of distributions, e.g., beta, gamma, and Dirichlet
- implemented in, e.g., PyTorch, JAX (numpyro, distrax)

Scaling to large datasets: can subsample data to speed up VI, e.g.,

$$\sum_{n=1}^{N} \log p(x_n \mid z) \approx \frac{N}{M} \sum_{m \in \mathcal{M}} \log p(x_m \mid z)$$

Giordano, et al. Black Box Variational Inference with a Deterministic Objective: Faster, More Accurate, and Even More Black Box. *JMLR* 2024.

Figurnov et al. Implicit reparameterization gradients. *NeurIPS* 2018.

Software implementations & other advances

Black-box VI and ADVI: Implemented in many software systems

Probabilistic programming software: e.g., Stan, Pyro, Turing, PyMC

Deterministic ADVI (DADVI): swaps differentiation and sampling; lower-variance, improved stability; alternative solvers

Implicit reparameterization: based on implicit differentiation;

- allows reparameterization to be applied to a broader class of distributions, e.g., beta, gamma, and Dirichlet
- implemented in, e.g., PyTorch, JAX (numpyro, distrax)

Scaling to large datasets: can subsample data to speed up VI, e.g.,

$$\sum_{n=1}^{N} \log p(x_n \mid z) \approx \frac{N}{M} \sum_{m \in \mathcal{M}} \log p(x_m \mid z)$$

A different approach: **amortization**

Giordano, et al. Black Box Variational Inference with a Deterministic Objective: Faster, More Accurate, and Even More Black Box. *JMLR* 2024.

Figurnov et al. Implicit reparameterization gradients. *NeurIPS* 2018.

Amortized inference

Amortized inference

Bayesian inference Fitting a probabilistic model $p(z \mid x)$ for *specific* input x

- Need to re-do inference every time we see a new data set x (expensive)
- Typically relies on knowing the likelihood

Amortized inference

Bayesian inference Fitting a probabilistic model $p(z \mid x)$ for *specific* input x

- Need to re-do inference every time we see a new data set x (expensive)
- Typically relies on knowing the likelihood

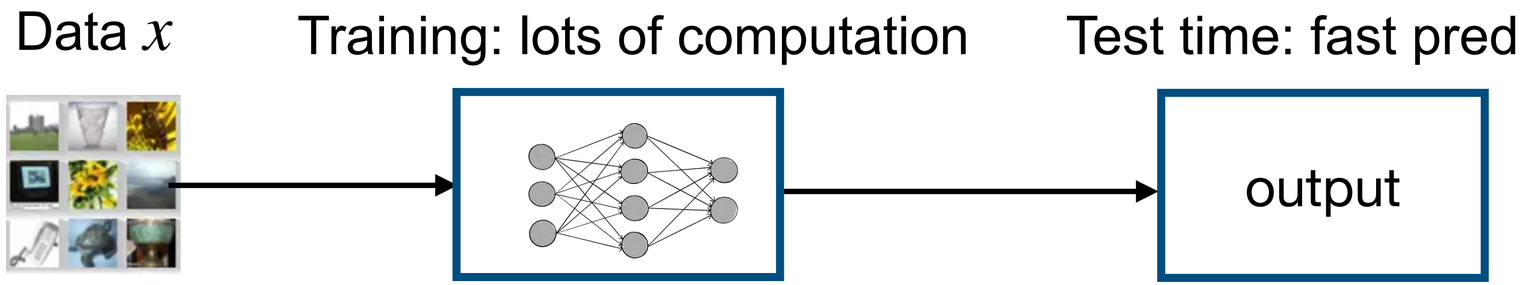
Amortized inference: draws from training/test ideas in ML but for the posterior

Amortized inference

Bayesian inference Fitting a probabilistic model $p(z \mid x)$ for *specific* input x

- Need to re-do inference every time we see a new data set x (expensive)
- Typically relies on knowing the likelihood

Amortized inference: draws from training/test ideas in ML but for the posterior

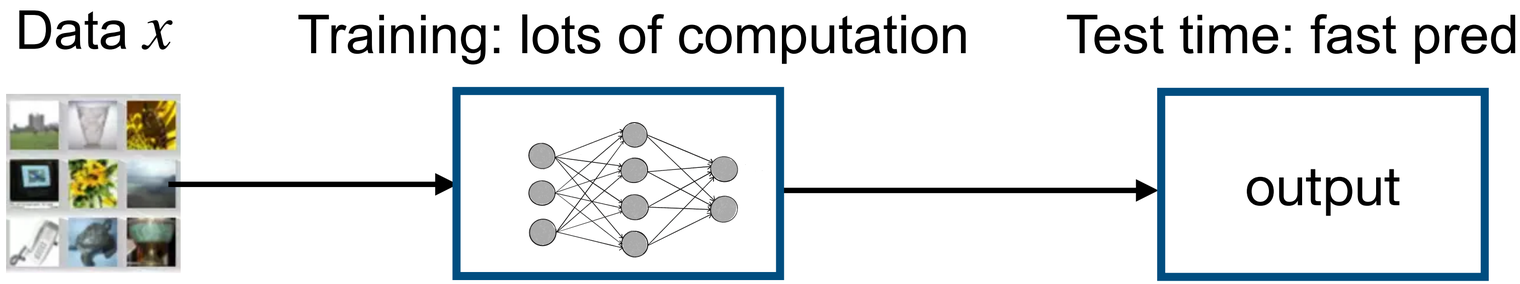


Amortized inference

Bayesian inference Fitting a probabilistic model $p(z \mid x)$ for *specific* input x

- Need to re-do inference every time we see a new data set x (expensive)
- Typically relies on knowing the likelihood

Amortized inference: draws from training/test ideas in ML but for the posterior



Learn probabilistic model $p(z \mid x)$

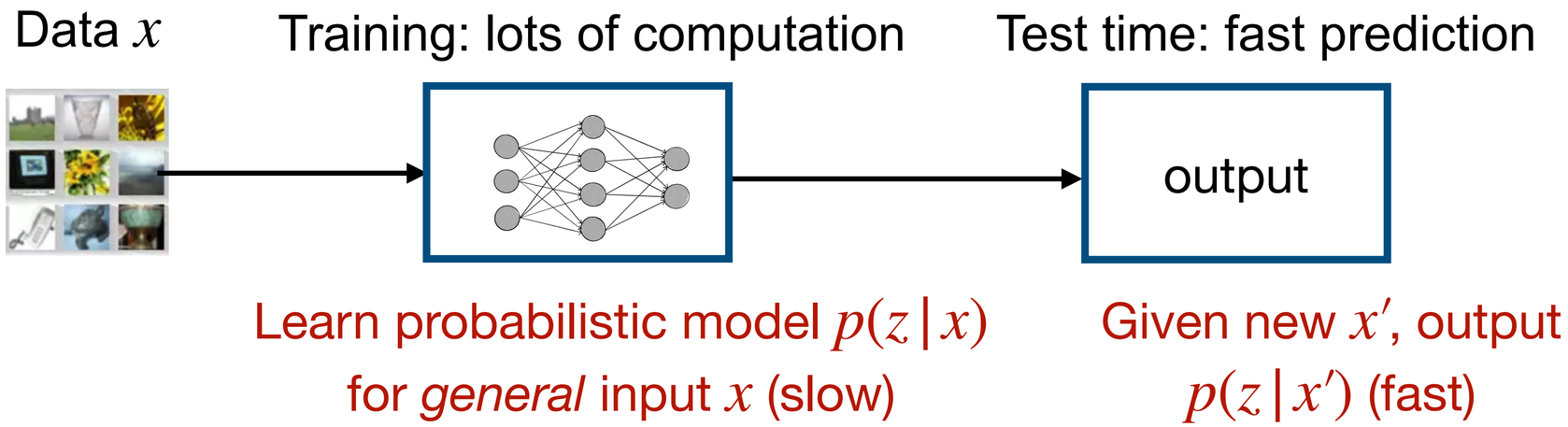
for *general* input x (slow)

Amortized inference

Bayesian inference Fitting a probabilistic model $p(z \mid x)$ for *specific* input x

- Need to re-do inference every time we see a new data set x (expensive)
- Typically relies on knowing the likelihood

Amortized inference: draws from training/test ideas in ML but for the posterior

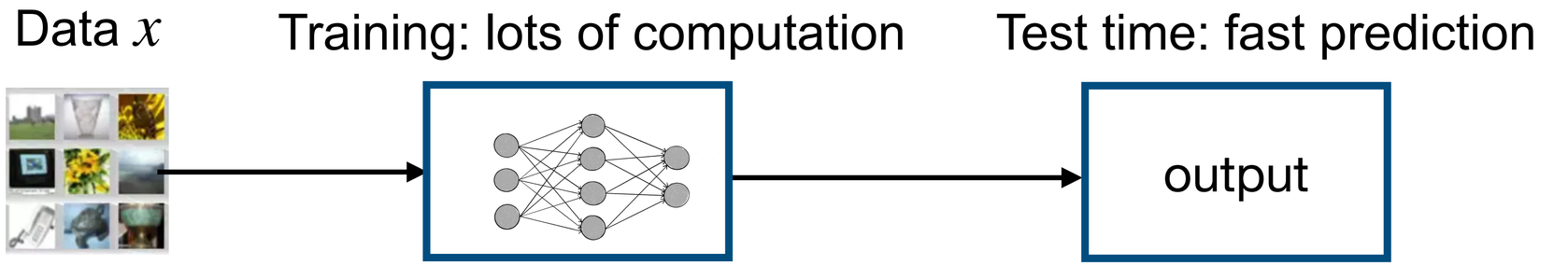


Amortized inference

Bayesian inference Fitting a probabilistic model $p(z \mid x)$ for *specific* input x

- Need to re-do inference every time we see a new data set x (expensive)
- Typically relies on knowing the likelihood

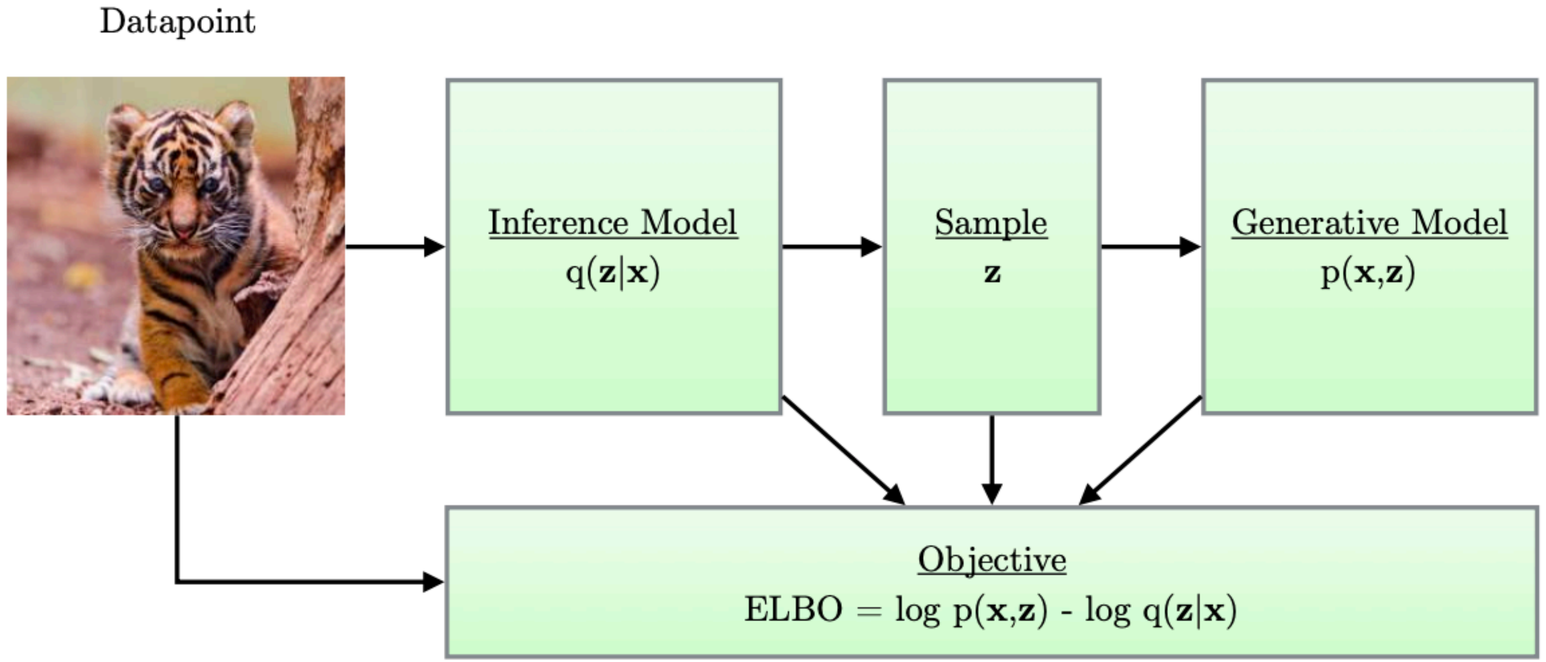
Amortized inference: draws from training/test ideas in ML but for the posterior



Applications:

- Generative modeling (e.g., variational autoencoders, normalizing flows)
- Simulation-based inference (e.g., neural posterior estimation)
- Bayesian optimization (e.g., hyperparameter inference)
- Meta-learning (e.g., neural processes)

Example 1: variational autoencoders (VAEs)



Key ideas in VAEs:

1. Use the ELBO to approximate (lower bound) the log likelihood function
2. Amortized inference is used to approximate the posterior
3. Reparameterization trick used for training

Kingma and Welling. Auto-encoding variational Bayes. *ICLR* 2014.
Rezende, Mohamed, Wierstra. Stochastic back propagation and approximate inference in deep generative models. *ICML* 2014.

Deep generative model and approximation

Consider a deep generative model, e.g.,

Deep generative model and approximation

Consider a deep generative model, e.g.,

Latent variable $z_n \sim \mathcal{N}(0, I)$

Observation  $x_n \mid z_n \sim \mathcal{N}(g_\mu(z_n; \theta), g_\Sigma(z_n; \theta))$

Deep generative model and approximation

Consider a deep generative model, e.g.,

Latent variable $z_n \sim \mathcal{N}(0, I)$

Observation  $x_n \mid z_n \sim \mathcal{N}(g_\mu(z_n;\theta), g_\Sigma(z_n;\theta))$ "Decoder"

Deep generative model and approximation

Consider a deep generative model, e.g.,

Latent variable $z_n \sim \mathcal{N}(0, I)$

Observation  $x_n \mid z_n \sim \mathcal{N}(g_\mu(z_n;\theta), g_\Sigma(z_n;\theta))$ "Decoder"

Output of a neural network with parameters θ

Deep generative model and approximation

Consider a deep generative model, e.g.,

Latent variable $z_n \sim \mathcal{N}(0, I)$

Observation  $x_n \mid z_n \sim \mathcal{N}(g_\mu(z_n;\theta), g_\Sigma(z_n;\theta))$ "Decoder"

Output of a neural network with parameters θ

Maximum likelihood of $p_\theta(x) = \int p_\theta(x,z)\,dz$ is intractable!

Deep generative model and approximation

Consider a deep generative model, e.g.,

Latent variable $z_n \sim \mathcal{N}(0, I)$

Observation  $x_n \mid z_n \sim \mathcal{N}(g_\mu(z_n; \theta), g_\Sigma(z_n; \theta))$ "Decoder"

Output of a neural network with parameters θ

Maximum likelihood of $p_\theta(x) = \int p_\theta(x, z)\, dz$ is intractable!

Instead, use approximate inference: $q_\phi(z \mid x) \approx p_\theta(z \mid x)$

Deep generative model and approximation

Consider a deep generative model, e.g.,

Latent variable $z_n \sim \mathcal{N}(0, I)$

Observation  $x_n \mid z_n \sim \mathcal{N}(g_\mu(z_n;\theta), g_\Sigma(z_n;\theta))$ "Decoder"

Output of a neural network with parameters θ

Maximum likelihood of $p_\theta(x) = \int p_\theta(x,z)\,dz$ is intractable!

Instead, use approximate inference: $q_\phi(z \mid x) \approx p_\theta(z \mid x)$

In VAEs, common to use a Gaussian, e.g.:

$$q_\phi(z \mid x) = \mathcal{N}(\Omega_\mu(x;\phi); \Omega_\Sigma(x;\phi))$$

Deep generative model and approximation

Consider a deep generative model, e.g.,

Latent variable $z_n \sim \mathcal{N}(0, I)$

Observation  $x_n \mid z_n \sim \mathcal{N}(g_\mu(z_n; \theta), g_\Sigma(z_n; \theta))$ "Decoder"

Output of a neural network with parameters θ

Maximum likelihood of $p_\theta(x) = \int p_\theta(x, z)\, dz$ is intractable!

Instead, use approximate inference: $q_\phi(z \mid x) \approx p_\theta(z \mid x)$

In VAEs, common to use a Gaussian, e.g.:

$q_\phi(z \mid x) = \mathcal{N}(\Omega_\mu(x; \phi); \Omega_\Sigma(x; \phi))$ "Encoder" or "Inference network"

Deep generative model and approximation

Consider a deep generative model, e.g.,

Latent variable $z_n \sim \mathcal{N}(0, I)$

Observation  $x_n \mid z_n \sim \mathcal{N}(g_\mu(z_n;\theta), g_\Sigma(z_n;\theta))$ "Decoder"

Output of a neural network with parameters θ

Maximum likelihood of $p_\theta(x) = \int p_\theta(x,z)\,dz$ is intractable!

Instead, use approximate inference: $q_\phi(z \mid x) \approx p_\theta(z \mid x)$

In VAEs, common to use a Gaussian, e.g.:

$q_\phi(z \mid x) = \mathcal{N}(\Omega_\mu(x;\phi); \Omega_\Sigma(x;\phi))$ "Encoder" or "Inference network"

Classical VI: *local latent variables for each data point* x_n

Deep generative model and approximation

Consider a deep generative model, e.g.,

Latent variable $z_n \sim \mathcal{N}(0, I)$

Observation  $x_n \mid z_n \sim \mathcal{N}(g_\mu(z_n;\theta), g_\Sigma(z_n;\theta))$ "Decoder"

Output of a neural network with parameters θ

Maximum likelihood of $p_\theta(x) = \int p_\theta(x,z)\,dz$ is intractable!

Instead, use approximate inference: $q_\phi(z \mid x) \approx p_\theta(z \mid x)$

In VAEs, common to use a Gaussian, e.g.:

$q_\phi(z \mid x) = \mathcal{N}(\Omega_\mu(x;\phi); \Omega_\Sigma(x;\phi))$ "Encoder" or "Inference network"

Classical VI: *local latent variables for each data point* x_n

Amortization: **shared parameter** ϕ **across all data points**

ELBO as a lower bound on the likelihood

Recall that we will use approximate inference: $q_\phi(z \mid x) \approx p_\theta(z \mid x)$

Write the ELBO as a function of the model parameters θ & variational parameters ϕ:

$$\text{ELBO}(\theta, \phi) = \mathbb{E}_{q_\phi(z|x)} \left[\log \left[\frac{p_\theta(x, z)}{q_\phi(z \mid x)} \right] \right]$$

ELBO as a lower bound on the likelihood

Recall that we will use approximate inference: $q_\phi(z \mid x) \approx p_\theta(z \mid x)$

Write the ELBO as a function of the model parameters θ & variational parameters ϕ:

$$\begin{aligned}\text{ELBO}(\theta, \phi) &= \mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{p_\theta(x,z)}{q_\phi(z \mid x)}\right]\right] \\ &= \log p_\theta(x) - \mathscr{D}_{KL}(q_\phi(z \mid x); p_\theta(z \mid x))\end{aligned}$$

ELBO as a lower bound on the likelihood

Recall that we will use approximate inference: $q_\phi(z \mid x) \approx p_\theta(z \mid x)$

Write the ELBO as a function of the model parameters θ & variational parameters ϕ:

$$\begin{aligned}
\text{ELBO}(\theta, \phi) &= \mathbb{E}_{q_\phi(z|x)} \left[\log \left[\frac{p_\theta(x, z)}{q_\phi(z \mid x)} \right] \right] \\
&= \log p_\theta(x) - \mathscr{D}_{KL}(q_\phi(z \mid x); p_\theta(z \mid x)) \\
&\leq \log p_\theta(x)
\end{aligned}$$

ELBO as a lower bound on the likelihood

Recall that we will use approximate inference: $q_\phi(z \mid x) \approx p_\theta(z \mid x)$

Write the ELBO as a function of the model parameters θ & variational parameters ϕ:

$$\begin{aligned}
\text{ELBO}(\theta, \phi) &= \mathbb{E}_{q_\phi(z|x)} \left[\log \left[\frac{p_\theta(x, z)}{q_\phi(z \mid x)} \right] \right] \\
&= \log p_\theta(x) - \mathscr{D}_{KL}(q_\phi(z \mid x); p_\theta(z \mid x)) \\
&\leq \log p_\theta(x)
\end{aligned}$$

Thus, maximizing ELBO w.r.t. θ, ϕ will simultaneously:

- maximize the marginal likelihood of $p_\theta(x)$
- minimize the KL divergence between q_ϕ and p_θ

Gradient of the ELBO: reparameterization trick

$$\text{ELBO}(\theta, \phi) = \mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{p_\theta(x, z)}{q_\phi(z \mid x)}\right]\right]$$

Gradient of the ELBO: reparameterization trick

$$\begin{aligned}\text{ELBO}(\theta, \phi) &= \mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{p_\theta(x,z)}{q_\phi(z\,|\,x)}\right]\right] \\ &= \mathbb{E}_{p(\epsilon)}\left[\log\left[\frac{p_\theta(x,z)}{q_\phi(z\,|\,x)}\right]\right], \quad z = g(\epsilon, \phi, x)\end{aligned}$$

Gradient of the ELBO: reparameterization trick

$$\text{ELBO}(\theta, \phi) = \mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{p_\theta(x,z)}{q_\phi(z\,|\,x)}\right]\right]$$

$$= \mathbb{E}_{p(\epsilon)}\left[\log\left[\frac{p_\theta(x,z)}{q_\phi(z\,|\,x)}\right]\right], \quad z = g(\epsilon, \phi, x)$$

Recall for a Gaussian $\mathcal{N}(\mu, \Sigma)$

$z = \mu + L\epsilon, \quad \Sigma = LL^\top$

$p(\epsilon) = \mathcal{N}(0, I)$

Gradient of the ELBO: reparameterization trick

$$\text{ELBO}(\theta, \phi) = \mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{p_\theta(x,z)}{q_\phi(z\,|\,x)}\right]\right]$$

$$= \mathbb{E}_{p(\epsilon)}\left[\log\left[\frac{p_\theta(x,z)}{q_\phi(z\,|\,x)}\right]\right], \quad z = g(\epsilon, \phi, x)$$

Recall for a Gaussian $\mathcal{N}(\mu, \Sigma)$

$z = \mu + L\epsilon, \quad \Sigma = LL^\top$

$p(\epsilon) = \mathcal{N}(0, I)$

Form Monte Carlo estimator of the ELBO $\widehat{\mathscr{L}}_{\theta,\phi}(x;\epsilon)$:

$$\hat{\epsilon} \sim p(\epsilon)$$

$$z = g(\hat{\epsilon}, \phi, x)$$

$$\widehat{\mathscr{L}}_{\theta,\phi}(x;\hat{\epsilon}) = \log p_\theta(x,z) - \log q_\phi(z\,|\,x)$$

Gradient of the ELBO: reparameterization trick

$$\text{ELBO}(\theta, \phi) = \mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{p_\theta(x,z)}{q_\phi(z\,|\,x)}\right]\right]$$

$$= \mathbb{E}_{p(\epsilon)}\left[\log\left[\frac{p_\theta(x,z)}{q_\phi(z\,|\,x)}\right]\right], \quad z = g(\epsilon, \phi, x)$$

Recall for a Gaussian $\mathcal{N}(\mu, \Sigma)$

$z = \mu + L\epsilon, \quad \Sigma = LL^\top$

$p(\epsilon) = \mathcal{N}(0, I)$

Form Monte Carlo estimator of the ELBO $\widehat{\mathscr{L}}_{\theta,\phi}(x;\epsilon)$:

$$\hat{\epsilon} \sim p(\epsilon)$$

$$z = g(\hat{\epsilon}, \phi, x)$$

$$\widehat{\mathscr{L}}_{\theta,\phi}(x;\hat{\epsilon}) = \log p_\theta(x,z) - \log q_\phi(z\,|\,x)$$

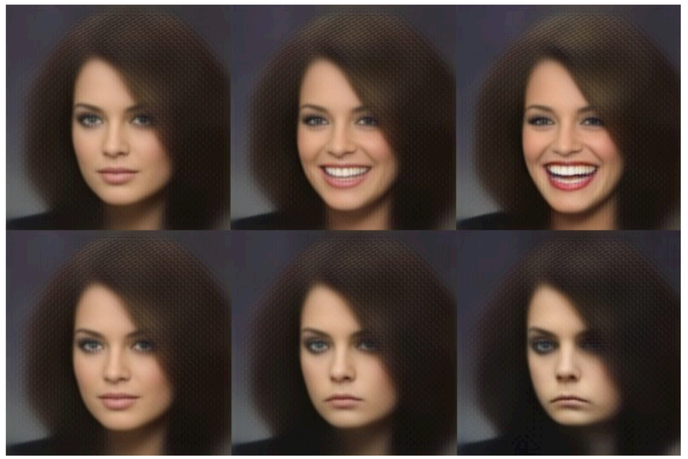
Gradient of the ELBO:

$$\nabla_{\theta,\phi}\text{ELBO}(\theta,\phi) = \mathbb{E}_{p(\epsilon)}[\nabla_{\theta,\phi}\widehat{\mathscr{L}}_{\theta,\phi}(x;\epsilon)]$$

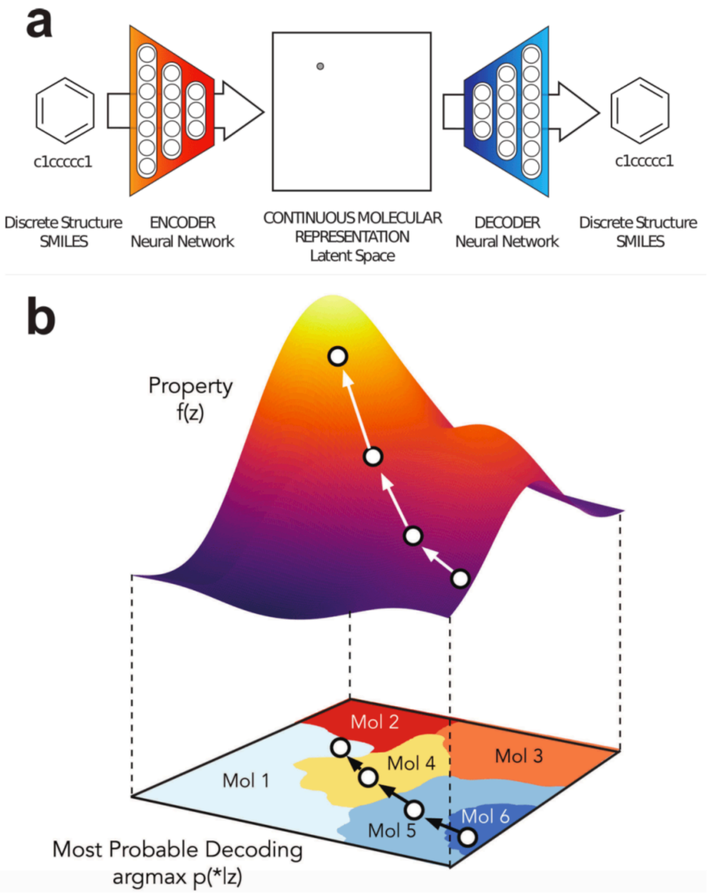
Applications of VAEs

Generating MNIST digits:

Image resynthesis:



Chemical design:



Rezende et al. Stochastic back propagation and approximate inference in deep generative models. *ICML* 2014.
Kingma and Welling. Introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 2019.

Example 2: Neural posterior estimation (NPE)

Goal: to compute the posterior with a test observation x_{obs}, i.e., $p(z \mid x_{obs})$

Papamakarios and Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation, 2018.
Greenberg, Nonnenmacher, and Macke. Automatic posterior transformation for likelihood-free inference, 2019

Example 2: Neural posterior estimation (NPE)

Goal: to compute the posterior with a test observation x_{obs}, i.e., $p(z \mid x_{obs})$

But… we do not have access to the likelihood. We can't evaluate $p(x \mid z)$

Papamakarios and Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation, 2018.
Greenberg, Nonnenmacher, and Macke. Automatic posterior transformation for likelihood-free inference, 2019

Example 2: Neural posterior estimation (NPE)

Goal: to compute the posterior with a test observation x_{obs}, i.e., $p(z \mid x_{obs})$

But… we do not have access to the likelihood. We can't evaluate $p(x \mid z)$

Examples: expensive (differential equation); intractable (e.g. integral)

Papamakarios and Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation, 2018.
Greenberg, Nonnenmacher, and Macke. Automatic posterior transformation for likelihood-free inference, 2019

Example 2: Neural posterior estimation (NPE)

Goal: to compute the posterior with a test observation x_{obs}, i.e., $p(z \mid x_{obs})$

But… we do not have access to the likelihood. We can't evaluate $p(x \mid z)$

Examples: expensive (differential equation); intractable (e.g. integral)

Suppose we *can* generate samples from $p(x \mid z)$

Papamakarios and Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation, 2018.
Greenberg, Nonnenmacher, and Macke. Automatic posterior transformation for likelihood-free inference, 2019

Example 2: Neural posterior estimation (NPE)

Goal: to compute the posterior with a test observation x_{obs}, i.e., $p(z \mid x_{obs})$

But… we do not have access to the likelihood. We can't evaluate $p(x \mid z)$

Examples: expensive (differential equation); intractable (e.g. integral)

Suppose we *can* generate samples from $p(x \mid z)$

In NPE, perform simulation-based inference: $q_\phi(z \mid x_{obs}) \approx p(z \mid x_{obs})$

Papamakarios and Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation, 2018.
Greenberg, Nonnenmacher, and Macke. Automatic posterior transformation for likelihood-free inference, 2019

Example 2: Neural posterior estimation (NPE)

Goal: to compute the posterior with a test observation x_{obs}, i.e., $p(z \mid x_{obs})$

But… we do not have access to the likelihood. We can't evaluate $p(x \mid z)$

Examples: expensive (differential equation); intractable (e.g. integral)

Suppose we *can* generate samples from $p(x \mid z)$

In NPE, perform simulation-based inference: $q_\phi(z \mid x_{obs}) \approx p(z \mid x_{obs})$

Use *amortization* so that we can learn a $q_\phi(z \mid x)$ for general x

Papamakarios and Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation, 2018.
Greenberg, Nonnenmacher, and Macke. Automatic posterior transformation for likelihood-free inference, 2019

Example 2: Neural posterior estimation (NPE)

Goal: to compute the posterior with a test observation x_{obs}, i.e., $p(z \mid x_{obs})$

But… we do not have access to the likelihood. We can't evaluate $p(x \mid z)$

Examples: expensive (differential equation); intractable (e.g. integral)

Suppose we *can* generate samples from $p(x \mid z)$

In NPE, perform simulation-based inference: $q_\phi(z \mid x_{obs}) \approx p(z \mid x_{obs})$

Use *amortization* so that we can learn a $q_\phi(z \mid x)$ for general x

Key ideas in NPE:

1. Amortized inference is used to approximate the posterior
2. Simulate pairs of the parameter and data
3. Minimize the expected forward KL via Monte Carlo

Papamakarios and Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation, 2018.
Greenberg, Nonnenmacher, and Macke. Automatic posterior transformation for likelihood-free inference, 2019

NPE setup

Posterior: $p(z \mid x) \propto p(z)\, p(x \mid z)$

NPE setup

Posterior: $p(z \mid x) \propto p(z)\, p(x \mid z)$

Prior

NPE setup

Posterior: $p(z \mid x) \propto p(z)\, p(x \mid z)$

Prior Simulator - can only generate samples (no likelihood)

NPE setup

Posterior: $p(z \mid x) \propto p(z)\, p(x \mid z)$

Prior Simulator - can only generate samples (no likelihood)

Chose a flexible model $q_\phi(z \mid x)$, where ϕ represents parameters of a neural network

NPE setup

Posterior: $p(z \mid x) \propto p(z)\, p(x \mid z)$

Prior Simulator - can only generate samples (no likelihood)

Chose a flexible model $q_\phi(z \mid x)$, where ϕ represents parameters of a neural network

Examples: Gaussian, **mixture of Gaussians**, normalizing flow

$$q_\phi(z \mid x) = \sum_{k=1}^{K} \Omega_{w_k}(x; \phi)\, \mathcal{N}(z \mid \Omega_{\mu_k}(x; \phi), \Omega_{\Sigma_k}(x; \phi))$$

NPE setup

Posterior: $p(z \mid x) \propto p(z)\, p(x \mid z)$

Prior Simulator - can only generate samples (no likelihood)

Chose a flexible model $q_\phi(z \mid x)$, where ϕ represents parameters of a neural network

Examples: Gaussian, **mixture of Gaussians**, normalizing flow

$$q_\phi(z \mid x) = \sum_{k=1}^{K} \Omega_{w_k}(x; \phi)\, \mathcal{N}(z \mid \Omega_{\mu_k}(x; \phi), \Omega_{\Sigma_k}(x; \phi))$$

Want to minimize the *expected **forward*** KL divergence over data sets:

$$\mathscr{L}(\phi) = \int \mathscr{D}_{KL}(p(z \mid x); q_\phi(z; x))\, p(x)\, dx$$

NPE setup

Posterior: $p(z \mid x) \propto p(z)\, p(x \mid z)$

Prior Simulator - can only generate samples (no likelihood)

Chose a flexible model $q_\phi(z \mid x)$, where ϕ represents parameters of a neural network

Examples: Gaussian, **mixture of Gaussians**, normalizing flow

$$q_\phi(z \mid x) = \sum_{k=1}^{K} \Omega_{w_k}(x; \phi)\, \mathcal{N}(z \mid \Omega_{\mu_k}(x; \phi), \Omega_{\Sigma_k}(x; \phi))$$

Want to minimize the *expected **forward*** KL divergence over data sets:

$$\mathscr{L}(\phi) = \int \mathscr{D}_{KL}(p(z \mid x); q_\phi(z; x))\, p(x)\, dx$$

Question: Why forward KL and not reverse KL? Problem: We can't evaluate $p(z, x)$

Deriving the neural posterior estimation objective

Goal: to compute the posterior with a test observation x_{obs}, i.e., $p(z \mid x_{obs})$

$$\mathscr{L}(\varphi) = \int \mathscr{D}_{KL}(p(z \mid x); q_{\varphi}(z; x))\, p(x) dx$$

Deriving the neural posterior estimation objective

Goal: to compute the posterior with a test observation x_{obs}, i.e., $p(z \mid x_{obs})$

$$\mathscr{L}(\varphi) = \int \mathscr{D}_{KL}(p(z \mid x); q_{\varphi}(z; x))\, p(x) dx$$

$$= \int \left[\int \log \left(\frac{p(z \mid x)}{q_{\varphi}(z; x))} \right) p(z \mid x) dz \right] p(x) dx$$

Deriving the neural posterior estimation objective

Goal: to compute the posterior with a test observation x_{obs}, i.e., $p(z \mid x_{obs})$

$$\mathscr{L}(\varphi) = \int \mathscr{D}_{KL}(p(z \mid x); q_\varphi(z; x))\, p(x) dx$$

$$= \int \left[\int \log \left(\frac{p(z \mid x)}{q_\varphi(z; x))} \right) p(z \mid x) dz \right] p(x) dx$$

$$= \int \int \log \left(\frac{p(z \mid x)}{q_\varphi(z; x))} \right) p(z, x)\, dz\, dx$$

Deriving the neural posterior estimation objective

Goal: to compute the posterior with a test observation x_{obs}, i.e., $p(z \mid x_{obs})$

$$\mathscr{L}(\varphi) = \int \mathscr{D}_{KL}(p(z \mid x); q_\varphi(z; x))\, p(x) dx$$

$$= \int \left[\int \log \left(\frac{p(z \mid x)}{q_\varphi(z; x))} \right) p(z \mid x) dz \right] p(x) dx$$

$$= \int \int \log \left(\frac{p(z \mid x)}{q_\varphi(z; x))} \right) p(z, x)\, dz\, dx$$

Deriving the neural posterior estimation objective

Goal: to compute the posterior with a test observation x_{obs}, i.e., $p(z \mid x_{obs})$

$$\mathscr{L}(\varphi) = \int \mathscr{D}_{KL}(p(z \mid x); q_\varphi(z; x))\, p(x) dx$$

$$= \int \left[\int \log \left(\frac{p(z \mid x)}{q_\varphi(z; x))} \right) p(z \mid x) dz \right] p(x) dx$$

Approximate with Monte Carlo samples!

$$= \int \int \log \left(\frac{p(z \mid x)}{q_\varphi(z; x))} \right) p(z, x)\, dz\, dx$$

Deriving the neural posterior estimation objective

Goal: to compute the posterior with a test observation x_{obs}, i.e., $p(z \mid x_{obs})$

$$\mathscr{L}(\varphi) = \int \mathscr{D}_{KL}(p(z \mid x); q_\varphi(z; x))\, p(x) dx$$

$$= \int \left[\int \log \left(\frac{p(z \mid x)}{q_\varphi(z; x))} \right) p(z \mid x) dz \right] p(x) dx$$

Approximate with Monte Carlo samples!

$$= \int \int \log \left(\frac{p(z \mid x)}{q_\varphi(z; x))} \right) p(z, x)\, dz\, dx$$

Draw $z^b \sim p(z)$

$$x^b \sim p(x \mid z)$$

Deriving the neural posterior estimation objective

Goal: to compute the posterior with a test observation x_{obs}, i.e., $p(z \mid x_{obs})$

$$\mathscr{L}(\varphi) = \int \mathscr{D}_{KL}(p(z \mid x); q_\varphi(z; x))\, p(x) dx$$

$$= \int \left[\int \log \left(\frac{p(z \mid x)}{q_\varphi(z; x))} \right) p(z \mid x) dz \right] p(x) dx$$

Approximate with Monte Carlo samples!

$$= \int \int \log \left(\frac{p(z \mid x)}{q_\varphi(z; x))} \right) p(z, x)\, dz\, dx$$

Draw $z^b \sim p(z)$

$x^b \sim p(x \mid z)$

$$\widehat{\mathscr{L}}(\varphi) = -\frac{1}{B} \sum_{b=1}^{B} \log q_\varphi(z^b; x^b)$$

Full NPE algorithm

Input: prior and simulator $p(z, x)$, conditional density estimator q_φ

For $b = 1, \ldots, B$

Draw $z^b, x^b \sim p(z, x)$

Train model to minimize the expected forward KL on the parameter-data pairs:

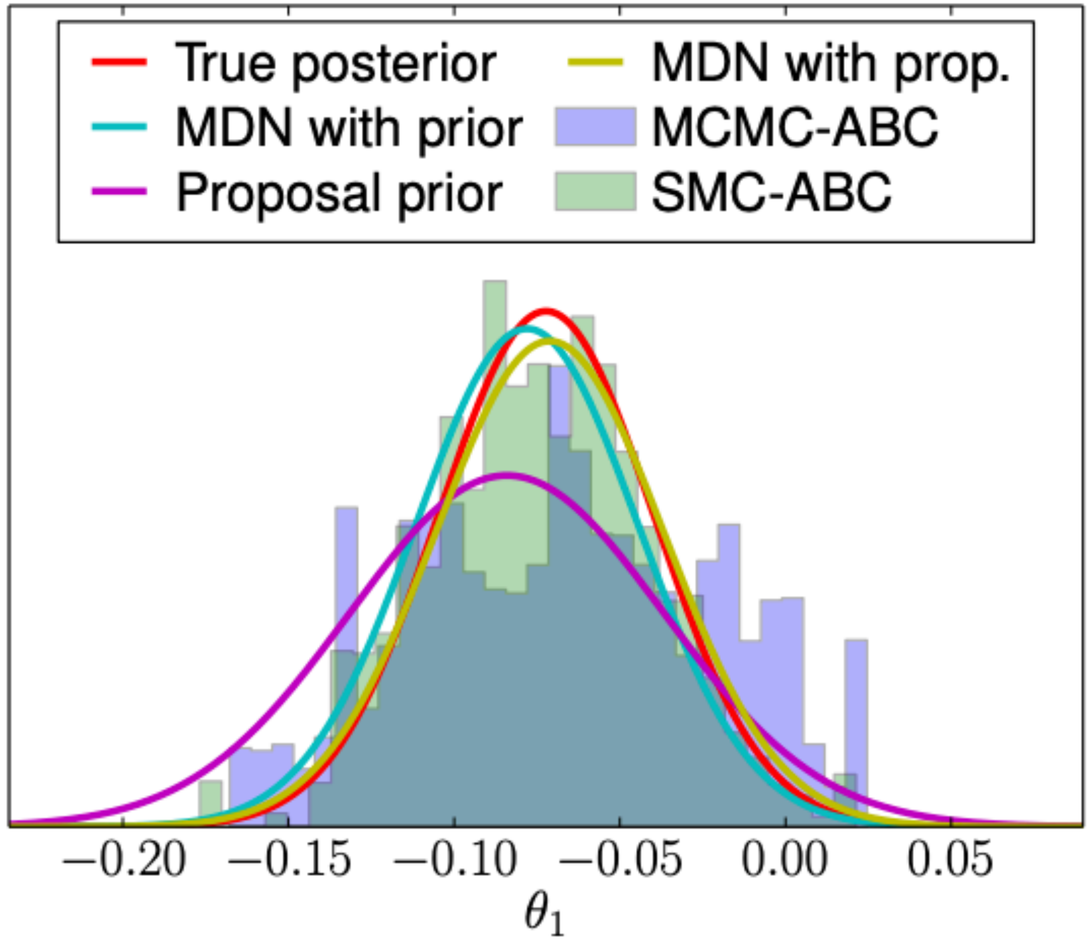
$$\varphi^* = \arg\min_\varphi -\frac{1}{B} \sum_{b=1}^{B} \log q_\varphi(z^b; x^b)$$

Output: $p(z \mid x_{obs}) \approx q_{\varphi^*}(z; x_{obs})$

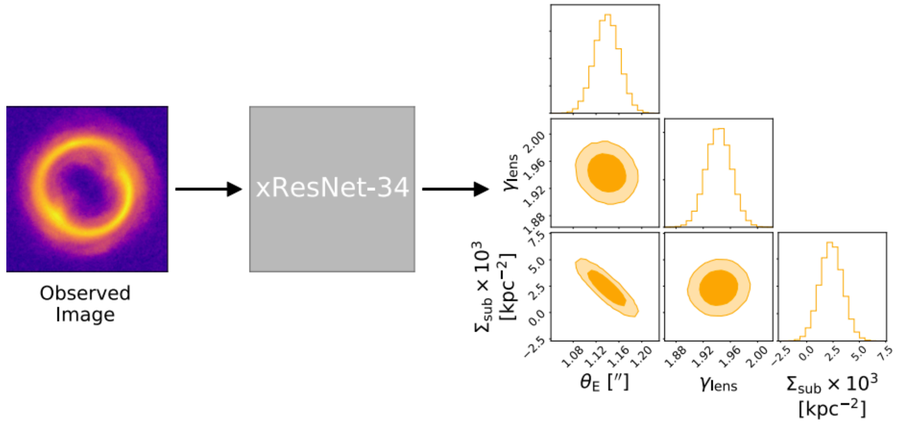
Often used in a *sequential* manner, known as sequential NPE (SNPE), where the prior is replaced by a changing proposal distribution.

Applications of NPE+

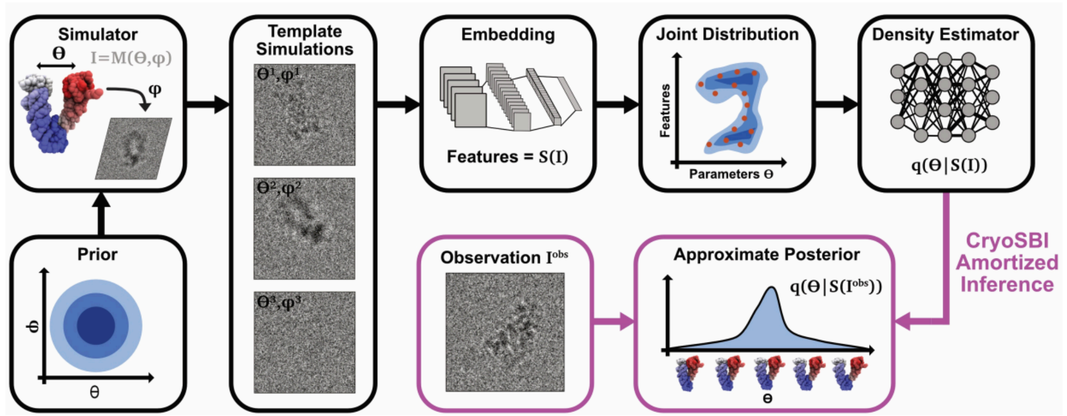
Bayesian linear regression



Astrophysics



Biophysics



Papamakarios and Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation, 2018.
Wagner-Carena et al. From Images to Dark Matter: End-To-End Inference of Substructure From Hundreds of Strong Gravitational Lenses, 2023.
Dingledein et al. Amortized template matching of molecular conformations from cryoelectron microscopy images using simulation-based inference, 2025.

Design principles I

Recall the basics of VI:

Step 1: Specify a *divergence* $\mathscr{D}(q;\pi)$ between variational density $q(z)$ and target $\pi(z)$

Step 2: Find the member of the family $\mathscr{Q}$ that minimizes the divergence, i.e.,

$$q^* = \arg\min_{q \in \mathscr{Q}} \mathscr{D}(q;\pi)$$

Step 3: Use variational density q^* instead of target $\pi(z)$ in downstream tasks: e.g.,

$$\mathbb{E}_{\pi(z)}[f(z)] \approx \mathbb{E}_{q^*(z)}[f(z)]$$

Design principles I

Recall the basics of VI:

Step 1: Specify a *divergence* $\mathscr{D}(q;\pi)$ between variational density $q(z)$ and target $\pi(z)$

Step 2: Find the member of the family $\mathcal{Q}$ that minimizes the divergence, i.e.,

$$q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q;\pi)$$

Step 3: Use variational density q^* instead of target $\pi(z)$ in downstream tasks: e.g.,

$$\mathbb{E}_{\pi(z)}[f(z)] \approx \mathbb{E}_{q^*(z)}[f(z)]$$

Design principles I

Recall the basics of VI:

Step 1: Specify a *divergence* $\mathscr{D}(q;\pi)$ between variational density $q(z)$ and target $\pi(z)$

Step 2: Find the member of the family $\mathcal{Q}$ that minimizes the divergence, i.e.,

$$q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q;\pi)$$

Step 3: Use variational density q^* instead of target $\pi(z)$ in downstream tasks: e.g.,

$$\mathbb{E}_{\pi(z)}[f(z)] \approx \mathbb{E}_{q^*(z)}[f(z)]$$

Expressivity

- correlations
- skew and kurtosis
- multi-modality

Design principles I

Recall the basics of VI:

Step 1: Specify a *divergence* $\mathscr{D}(q;\pi)$ between variational density $q(z)$ and target $\pi(z)$

Step 2: Find the member of the family $\mathcal{Q}$ that minimizes the divergence, i.e.,

$$q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q;\pi)$$

Step 3: Use variational density q^* instead of target $\pi(z)$ in downstream tasks: e.g.,

$$\mathbb{E}_{\pi(z)}[f(z)] \approx \mathbb{E}_{q^*(z)}[f(z)]$$

Expressivity

- correlations
- skew and kurtosis
- multi-modality

Tractability

- Easy and fast to evaluate $q(z)$
- Fast *sampling* from its distribution (e.g., to approximate expectations)
- Ease of optimization

Example of design principles

Expressivity

- correlations
- skew and kurtosis
- multi-modality

Tractability

- Easy and fast to evaluate $q(z)$
- Efficient sampling from its distribution (e.g., to approximate expectations)
- Ease of optimization

Example of design principles

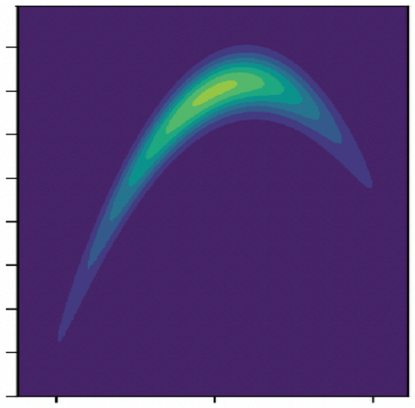
Expressivity

- correlations
- skew and kurtosis
- multi-modality

Tractability

- Easy and fast to evaluate $q(z)$
- Efficient sampling from its distribution (e.g., to approximate expectations)
- Ease of optimization

Target distribution



Example of design principles

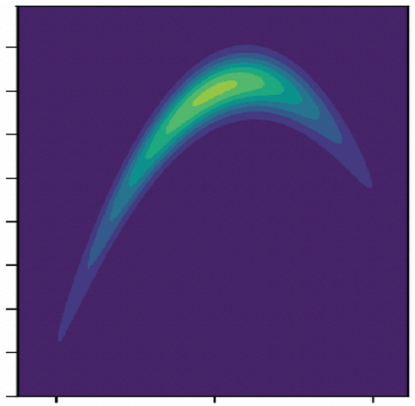
Expressivity

- correlations
- skew and kurtosis
- multi-modality

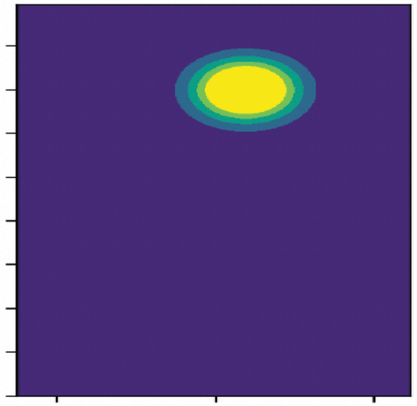
Tractability

- Easy and fast to evaluate $q(z)$
- Efficient sampling from its distribution (e.g., to approximate expectations)
- Ease of optimization

Target distribution



Factorized Gaussian



Example of design principles

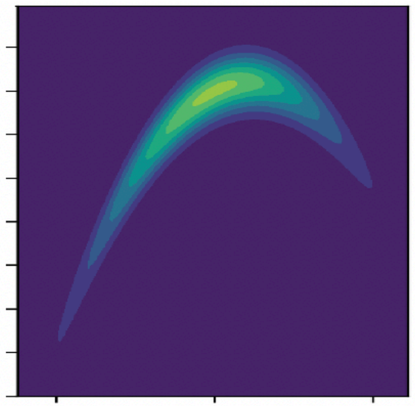
Expressivity

- correlations
- skew and kurtosis
- multi-modality

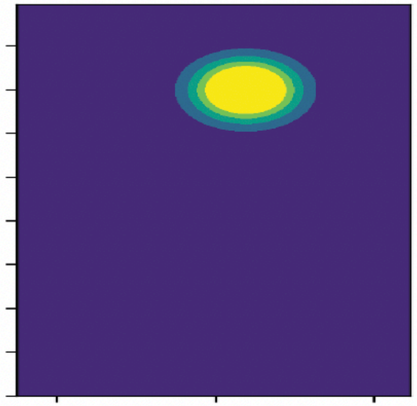
Tractability

- Easy and fast to evaluate $q(z)$
- Efficient sampling from its distribution (e.g., to approximate expectations)
- Ease of optimization

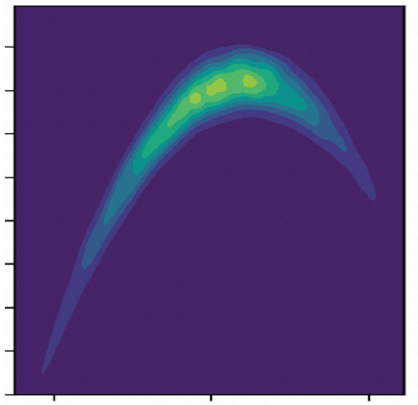
Target distribution



Factorized Gaussian



Energy-based model



$$q(z) = \frac{1}{Z_\phi} e^{-E_\phi(z)}$$

Part III: Advances

Score matching, flow matching, and diffusion

Black-box inference, variational inference, and misspecification

Black-box inference, variational inference, and misspecification

Black-box inference: no model-specific derivations, general targets, accessible to scientists

Black-box inference, variational inference, and misspecification

Black-box inference: no model-specific derivations, general targets, accessible to scientists

Variational inference (VI): casts the inference problem as an *optimization* problem: find the member of a simpler family that is closest to the target distribution

Black-box inference, variational inference, and misspecification

Black-box inference: no model-specific derivations, general targets, accessible to scientists

Variational inference (VI): casts the inference problem as an *optimization* problem: find the member of a simpler family that is closest to the target distribution

Black-box inference, variational inference, and misspecification

Black-box inference: no model-specific derivations, general targets, accessible to scientists

Variational inference (VI): casts the inference problem as an *optimization* problem: find the member of a simpler family that is closest to the target distribution

Expressivity

- correlations
- skew and kurtosis
- multi-modality

Black-box inference, variational inference, and misspecification

Black-box inference: no model-specific derivations, general targets, accessible to scientists

Variational inference (VI): casts the inference problem as an *optimization* problem: find the member of a simpler family that is closest to the target distribution

Expressivity

- correlations
- skew and kurtosis
- multi-modality

Tractability

- Easy to evaluate $q(z)$
- Can sample from its distribution (e.g., to approximate expectations)
- Ease of optimization

Black-box inference, variational inference, and misspecification

Black-box inference: no model-specific derivations, general targets, accessible to scientists

Variational inference (VI): casts the inference problem as an *optimization* problem: find the member of a simpler family that is closest to the target distribution

Expressivity

- correlations
- skew and kurtosis
- multi-modality

Tractability

- Easy to evaluate $q(z)$
- Can sample from its distribution (e.g., to approximate expectations)
- Ease of optimization

Most common in VI: factorized (mean-field) Gaussian — not so expressive but very tractable

Black-box inference, variational inference, and misspecification

Black-box inference: no model-specific derivations, general targets, accessible to scientists

Variational inference (VI): casts the inference problem as an *optimization* problem: find the member of a simpler family that is closest to the target distribution

Expressivity

- correlations
- skew and kurtosis
- multi-modality

Tractability

- Easy to evaluate $q(z)$
- Can sample from its distribution (e.g., to approximate expectations)
- Ease of optimization

Most common in VI: factorized (mean-field) Gaussian — not so expressive but very tractable

Many successes with this family for BBVI via stochastic gradient methods

Challenges of black-box variational inference

More expressive families face challenges in optimization: can have very slow convergence due to noisy gradients, sensitivity to learning rates, batch sizes, etc.

Cai et al. Batch and match: black-box variational inference with a score-based divergence. *ICML* 2024.

Challenges of black-box variational inference

More expressive families face challenges in optimization: can have very slow convergence due to noisy gradients, sensitivity to learning rates, batch sizes, etc.

Happens even in Gaussian (with full covariance)!

Cai et al. Batch and match: black-box variational inference with a score-based divergence. *ICML* 2024.

Challenges of black-box variational inference

More expressive families face challenges in optimization: can have very slow convergence due to noisy gradients, sensitivity to learning rates, batch sizes, etc.

Happens even in Gaussian (with full covariance)!

Question: What if we keep the “black box” nature of the target distribution, but capitalize on the *structure* of the variational family? Structure = Scores

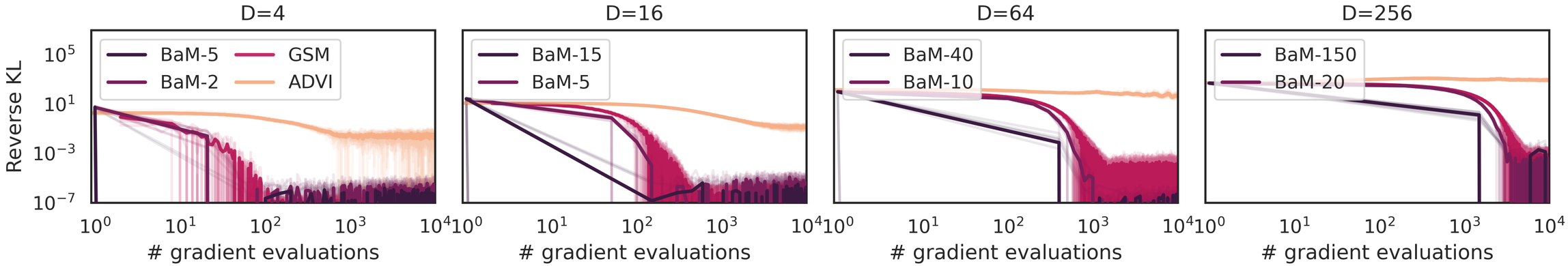
Cai et al. Batch and match: black-box variational inference with a score-based divergence. *ICML* 2024.

Challenges of black-box variational inference

More expressive families face challenges in optimization: can have very slow convergence due to noisy gradients, sensitivity to learning rates, batch sizes, etc.

Happens even in Gaussian (with full covariance)!

Question: What if we keep the "black box" nature of the target distribution, but capitalize on the *structure* of the variational family? Structure = Scores



Cai et al. Batch and match: black-box variational inference with a score-based divergence. *ICML* 2024.

VI with score matching?

Why scores? **“Black box”**: don’t need $\pi(z)$, just its (“Stein”) score $\nabla_z \log \pi(z)$

(Avoids normalizing constants)

VI with score matching?

Why scores? **“Black box”**: don’t need $\pi(z)$, just its (“Stein”) score $\nabla_z \log \pi(z)$

(Avoids normalizing constants)

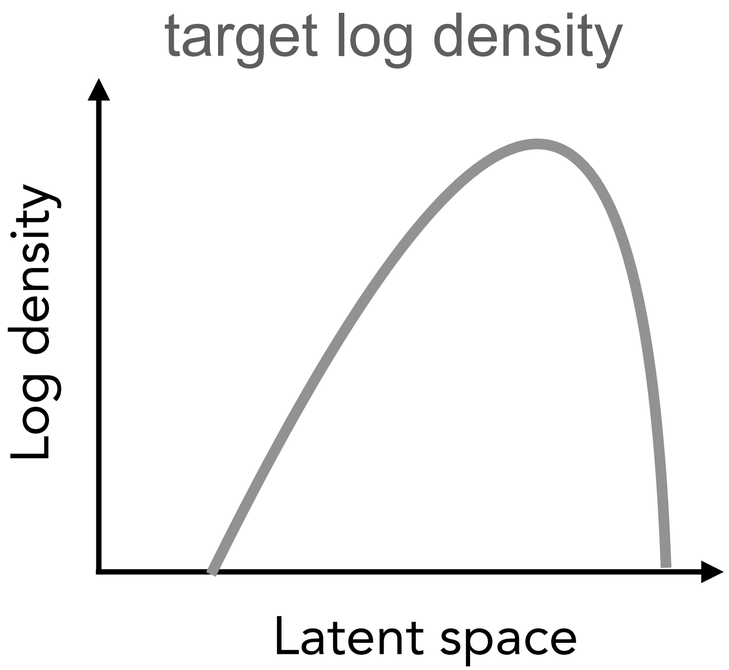
Given samples, how to make the scores close?

VI with score matching?

Why scores? **“Black box”**: don’t need $\pi(z)$, just its (“Stein”) score $\nabla_z \log \pi(z)$

(Avoids normalizing constants)

Given samples, how to make the scores close?

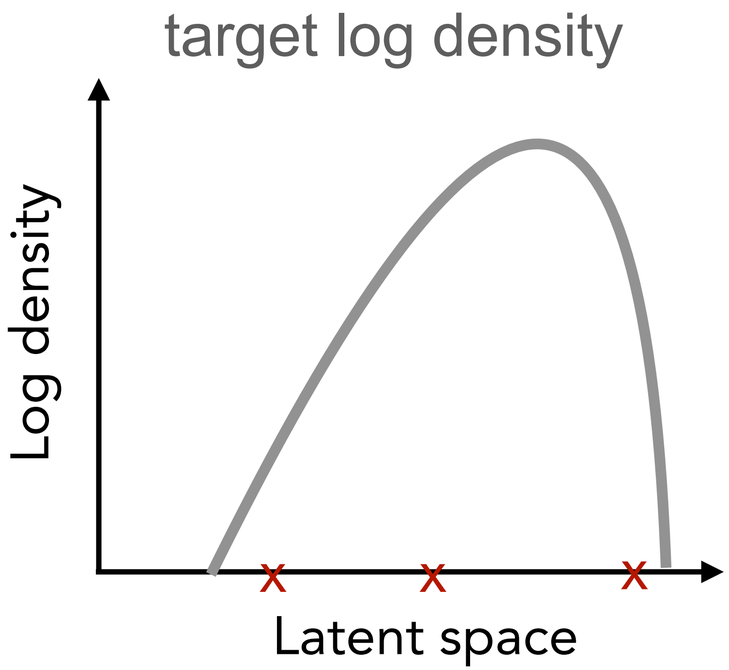


VI with score matching?

Why scores? **“Black box”**: don’t need $\pi(z)$, just its (“Stein”) score $\nabla_z \log \pi(z)$

(Avoids normalizing constants)

Given samples, how to make the scores close?

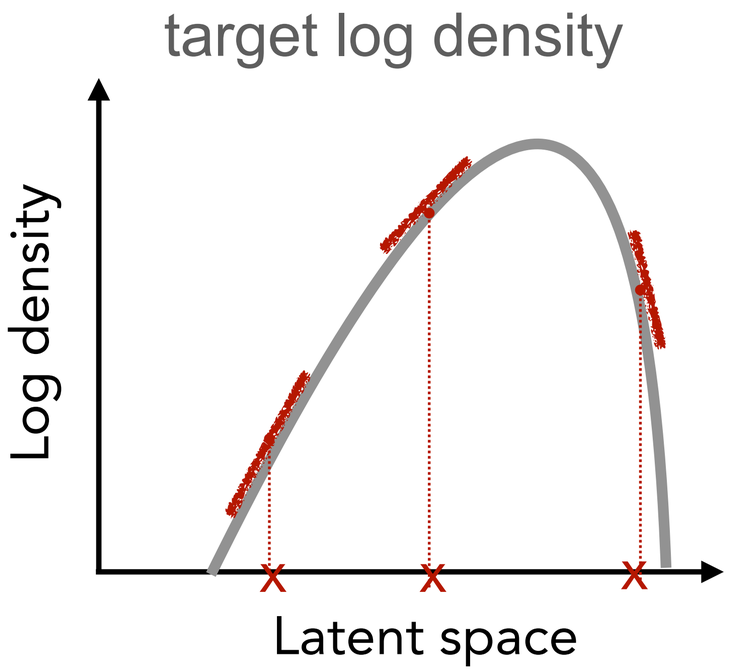


VI with score matching?

Why scores? **“Black box”**: don’t need $\pi(z)$, just its (“Stein”) score $\nabla_z \log \pi(z)$

(Avoids normalizing constants)

Given samples, how to make the scores close?

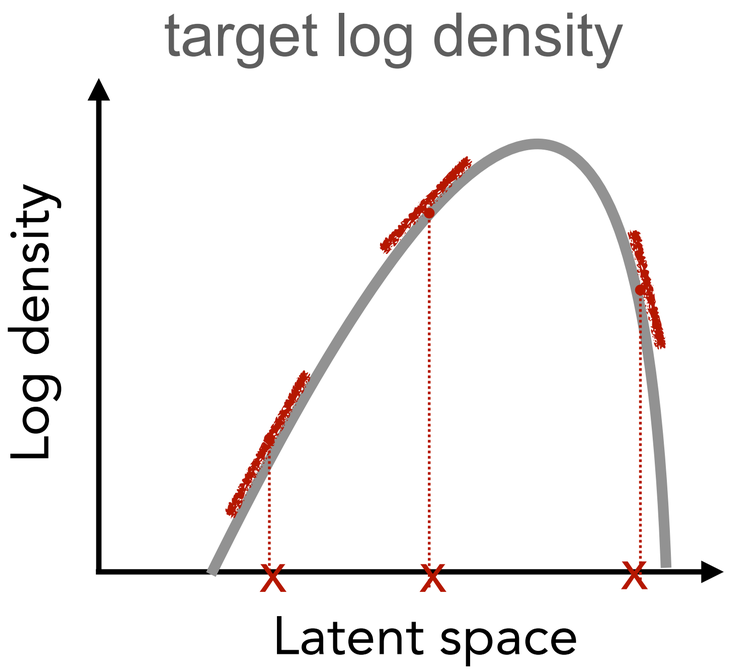


VI with score matching?

Why scores? **“Black box”**: don’t need $\pi(z)$, just its (“Stein”) score $\nabla_z \log \pi(z)$

(Avoids normalizing constants)

Given samples, how to make the scores close?



For Gaussian $\mathcal{Q}$:

- $\log q(z)$ *is a quadratic*
- $\nabla \log q(z) = \Sigma_q^{-1}(z - \mu_q)$

VI with score matching?

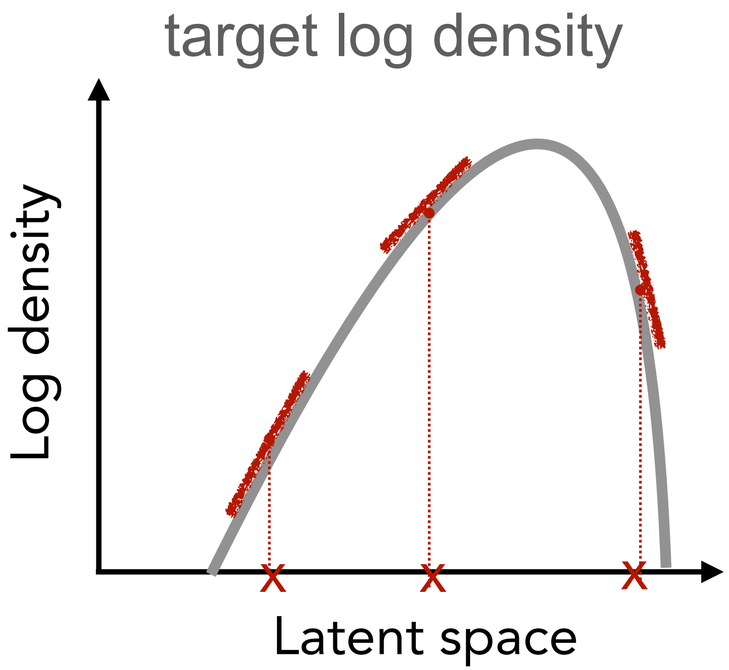
Why scores? **“Black box”**: don’t need $\pi(z)$, just its (“Stein”) score $\nabla_z \log \pi(z)$

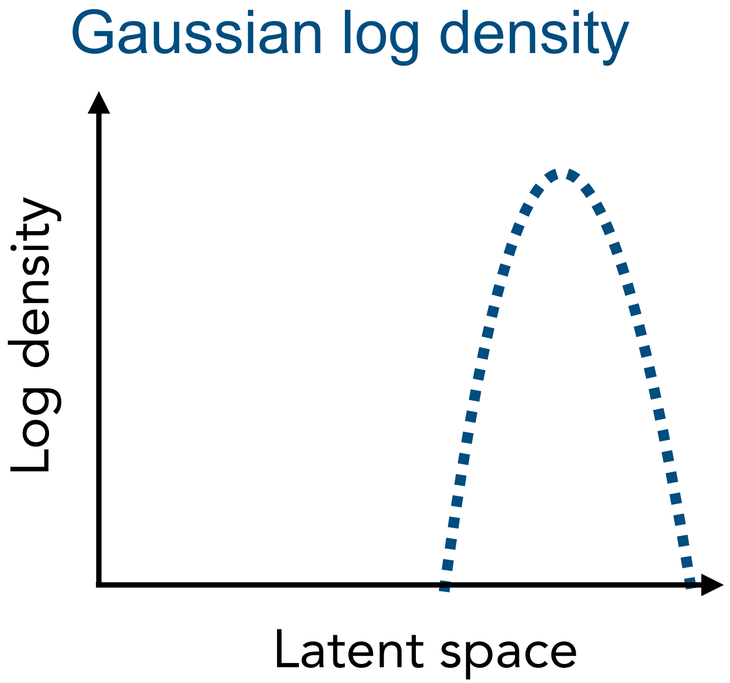
(Avoids normalizing constants)

Given samples, how to make the scores close?

For Gaussian $\mathcal{Q}$:

- $\log q(z)$ *is a quadratic*
- $\nabla \log q(z) = \Sigma_q^{-1}(z - \mu_q)$





VI with score matching?

Why scores? **“Black box”**: don’t need $\pi(z)$, just its (“Stein”) score $\nabla_z \log \pi(z)$

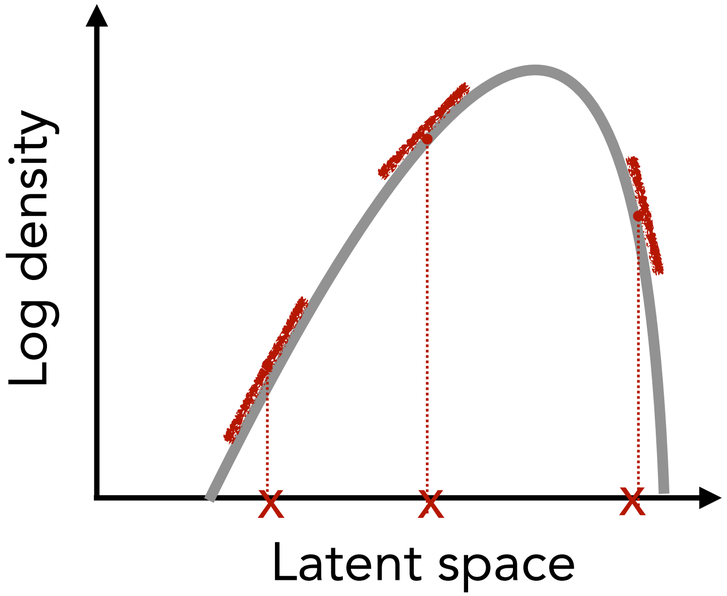
(Avoids normalizing constants)

Given samples, how to make the scores close?

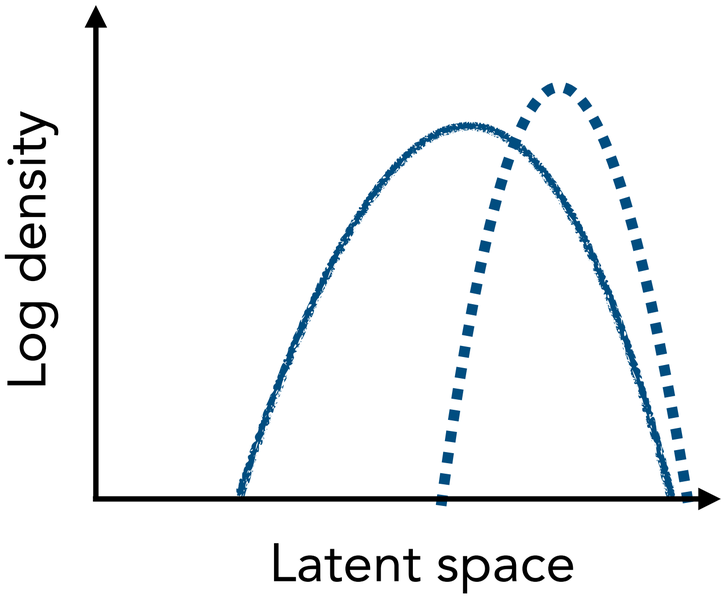
For Gaussian $\mathcal{Q}$:

- $\log q(z)$ *is a quadratic*
- $\nabla \log q(z) = \Sigma_q^{-1}(z - \mu_q)$

target log density



Gaussian log density



VI with score matching?

Why scores? **“Black box”**: don’t need $\pi(z)$, just its (“Stein”) score $\nabla_z \log \pi(z)$

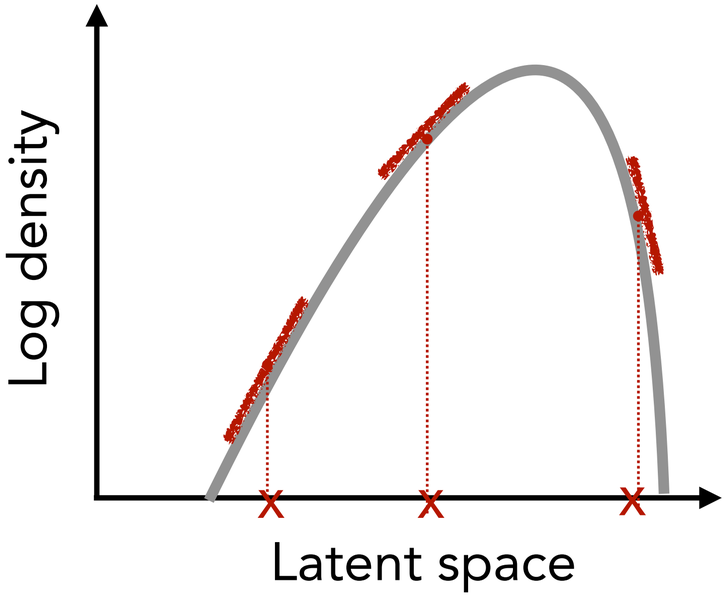
(Avoids normalizing constants)

Given samples, how to make the scores close?

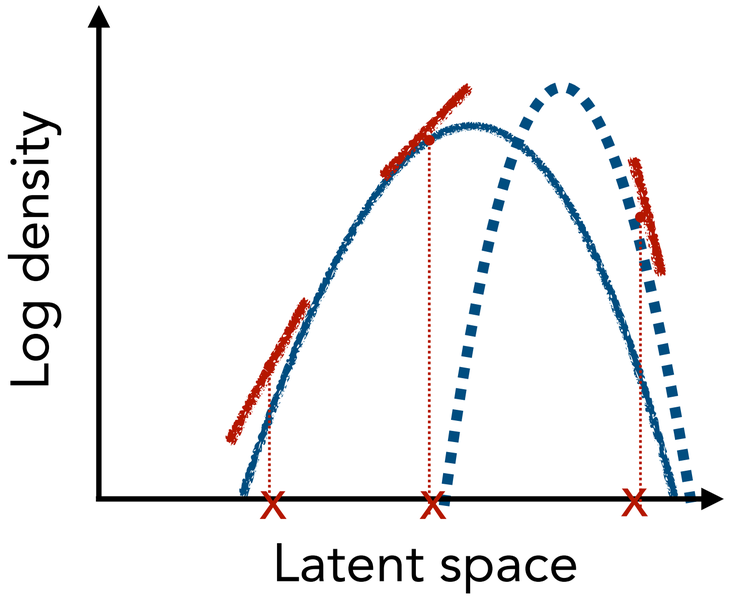
For Gaussian $\mathcal{Q}$:

- $\log q(z)$ *is a quadratic*
- $\nabla \log q(z) = \Sigma_q^{-1}(z - \mu_q)$

target log density



Gaussian log density

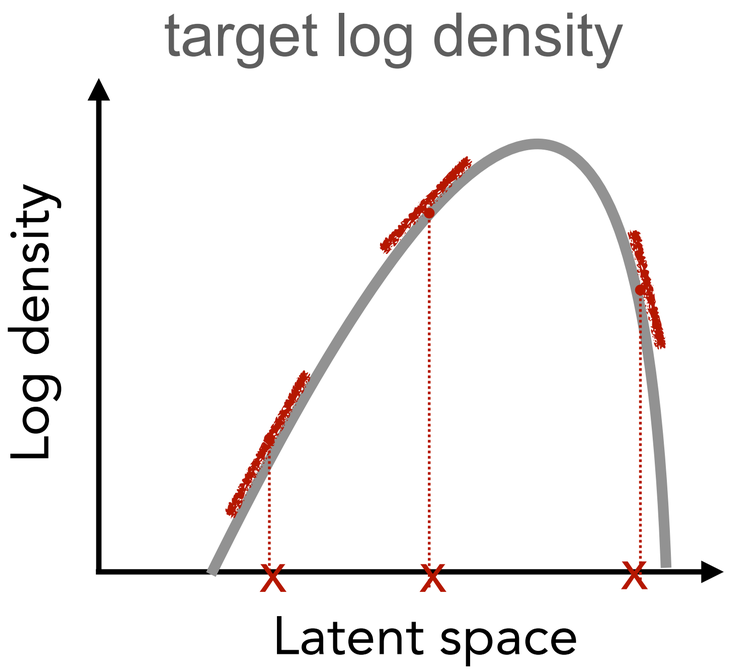


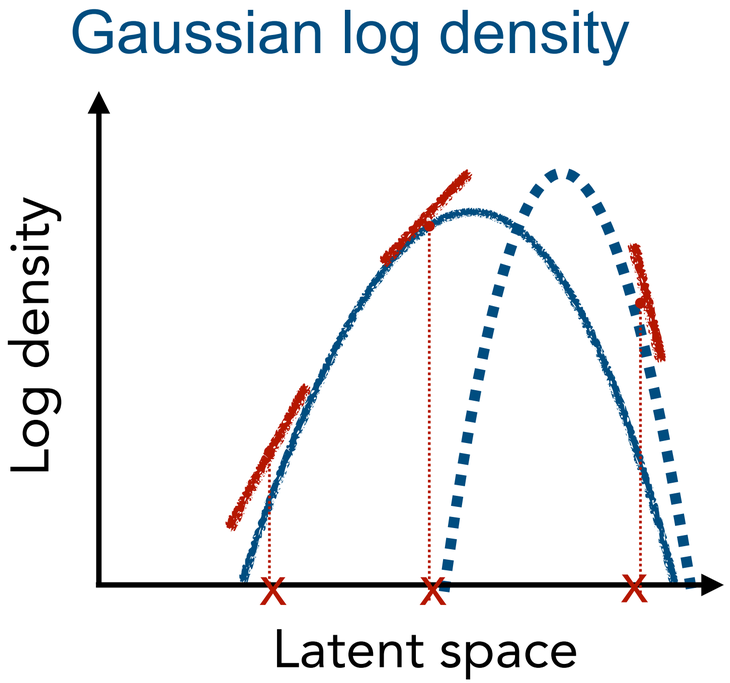
VI with score matching?

Why scores? **“Black box”**: don’t need $\pi(z)$, just its (“Stein”) score $\nabla_z \log \pi(z)$

(Avoids normalizing constants)

Given samples, how to make the scores close?





For Gaussian $\mathcal{Q}$:

- $\log q(z)$ *is a quadratic*
- $\nabla \log q(z) = \Sigma_q^{-1}(z - \mu_q)$

Under mild conditions,

$$\log p = \log q \iff \nabla \log p = \nabla \log q$$

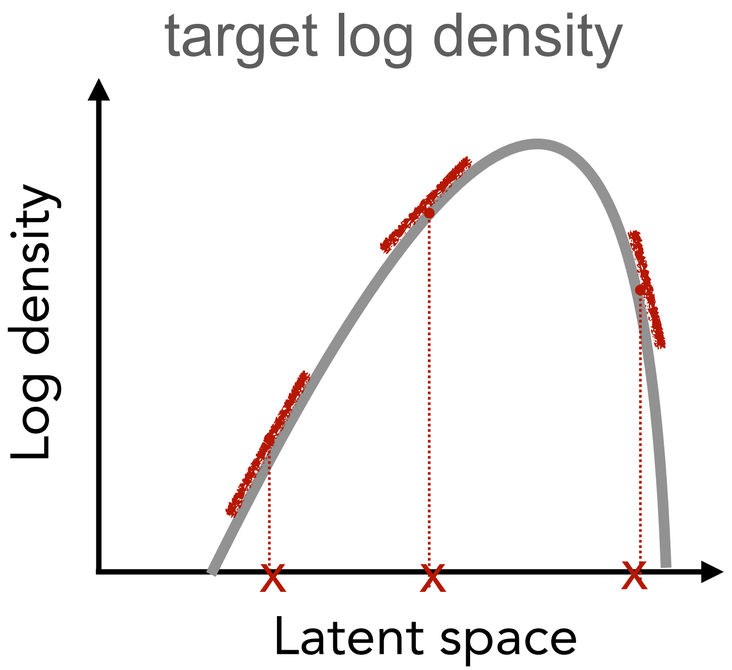
Modi et al.. Variational inference with Gaussian score matching. NeurIPS 2023.

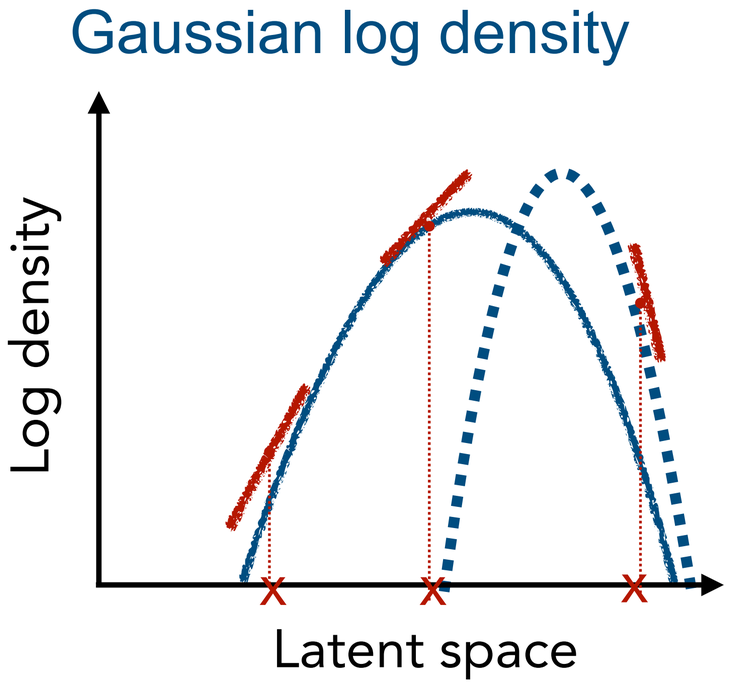
VI with score matching?

Why scores? **“Black box”**: don’t need $\pi(z)$, just its (“Stein”) score $\nabla_z \log \pi(z)$

(Avoids normalizing constants)

Given samples, how to make the scores close?





For Gaussian $\mathbb{Q}$:

- $\log q(z)$ *is a quadratic*
- $\nabla \log q(z) = \Sigma_q^{-1}(z - \mu_q)$

Under mild conditions,

$$\log p = \log q \iff \nabla \log p = \nabla \log q$$

But: can’t match scores exactly at multiple points for non-Gaussian targets

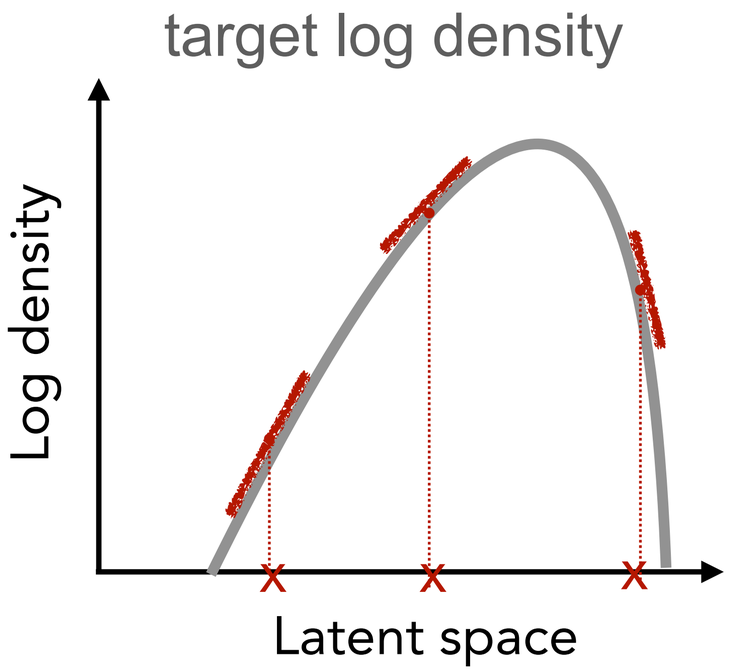
Modi et al.. Variational inference with Gaussian score matching. NeurIPS 2023.

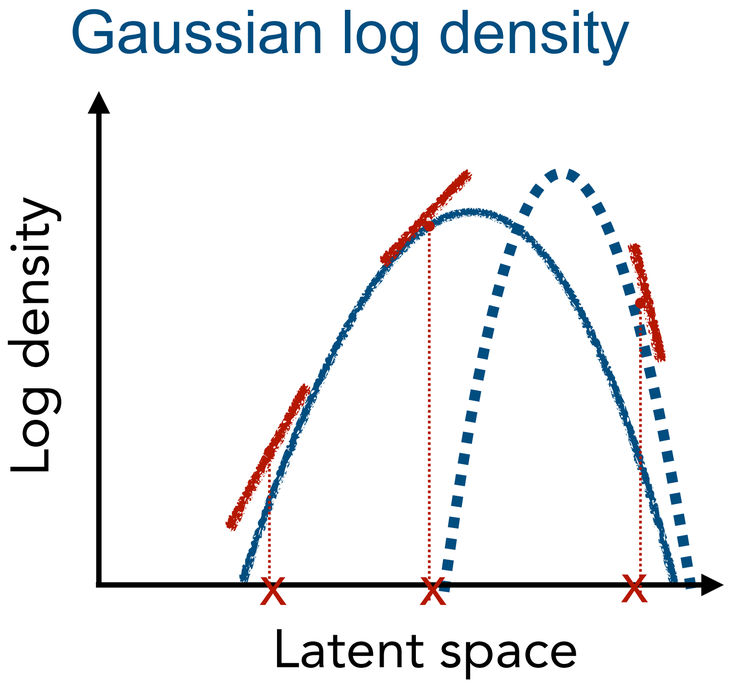
VI with score matching?

Why scores? **"Black box"**: don't need $\pi(z)$, just its ("Stein") score $\nabla_z \log \pi(z)$

(Avoids normalizing constants)

Given samples, how to make the scores close?





For Gaussian $\mathcal{Q}$:

- $\log q(z)$ *is a quadratic*
- $\nabla \log q(z) = \Sigma_q^{-1}(z - \mu_q)$

Under mild conditions,

$$\log p = \log q \iff \nabla \log p = \nabla \log q$$

But: can't match scores exactly at multiple points for non-Gaussian targets

Instead: Formulate variational inference with a *score-based divergence*

Modi et al.. Variational inference with Gaussian score matching. NeurIPS 2023.

Fisher divergence

Fisher Divergence: measures the agreement between the ***scores*** of q and π:

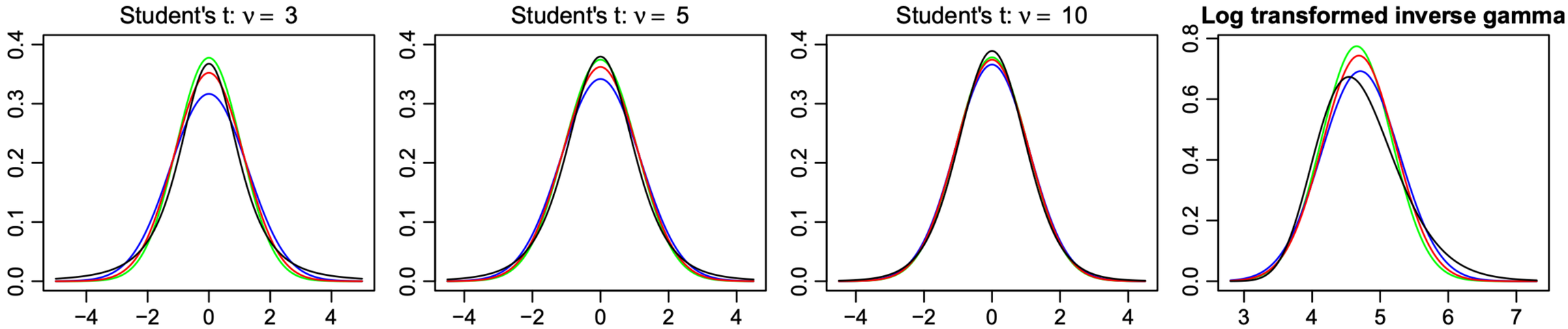
$$\mathscr{D}(q;\pi) = \int \left\| \nabla_z \log q(z) - \nabla_z \log \pi(z) \right\|^2 q(z)\, dz$$

Hyvarinen. Estimation of Non-Normalized Statistical Models by Score Matching. *JMLR*, 2005.

Fisher divergence

Fisher Divergence: measures the agreement between the ***scores*** of q and π:

$$\mathscr{D}(q;\pi) = \int \left\| \nabla_z \log q(z) - \nabla_z \log \pi(z) \right\|^2 q(z)\, dz$$



Hyvarinen. Estimation of Non-Normalized Statistical Models by Score Matching. *JMLR*, 2005.

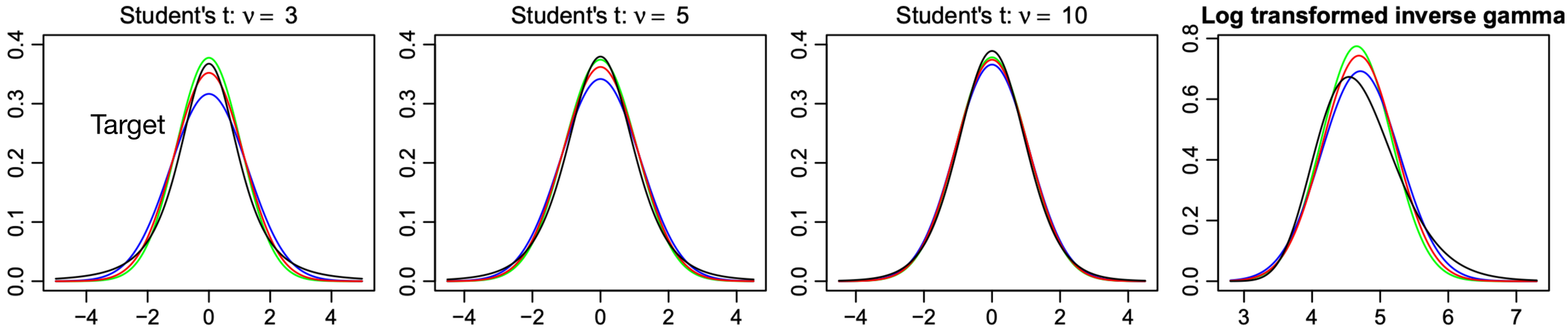
Yang et al. Variational approximations using Fisher divergence. arXiv, 2019.

Chen et al. Weighted Fisher divergence for high-dimensional Gaussian variational inference. arXiv, 2025

Fisher divergence

Fisher Divergence: measures the agreement between the ***scores*** of q and π:

$$\mathscr{D}(q;\pi) = \int \left\| \nabla_z \log q(z) - \nabla_z \log \pi(z) \right\|^2 q(z)\, dz$$



Hyvarinen. Estimation of Non-Normalized Statistical Models by Score Matching. *JMLR*, 2005.

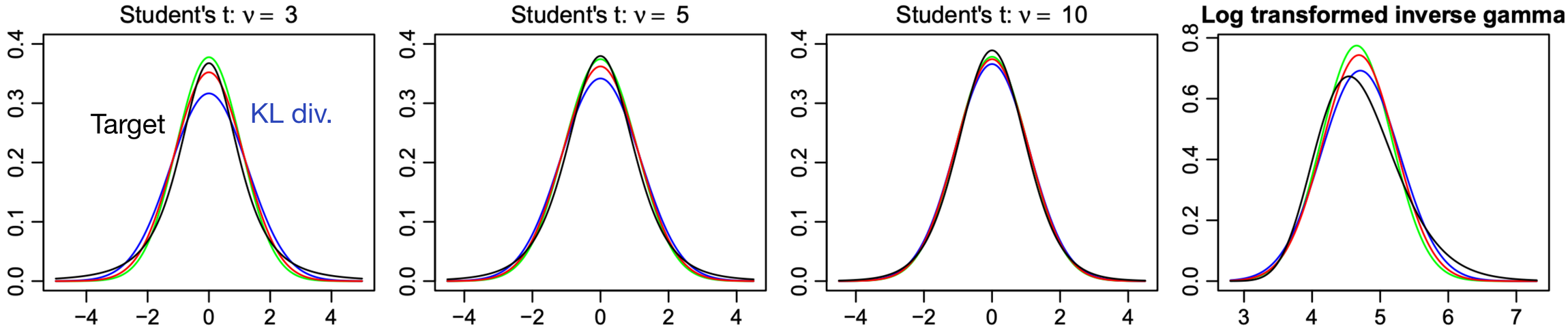
Yang et al. Variational approximations using Fisher divergence. arXiv, 2019.

Chen et al. Weighted Fisher divergence for high-dimensional Gaussian variational inference. arXiv, 2025

Fisher divergence

Fisher Divergence: measures the agreement between the ***scores*** of q and π:

$$\mathscr{D}(q;\pi) = \int \left\| \nabla_z \log q(z) - \nabla_z \log \pi(z) \right\|^2 q(z)\, dz$$



Hyvarinen. Estimation of Non-Normalized Statistical Models by Score Matching. *JMLR*, 2005.

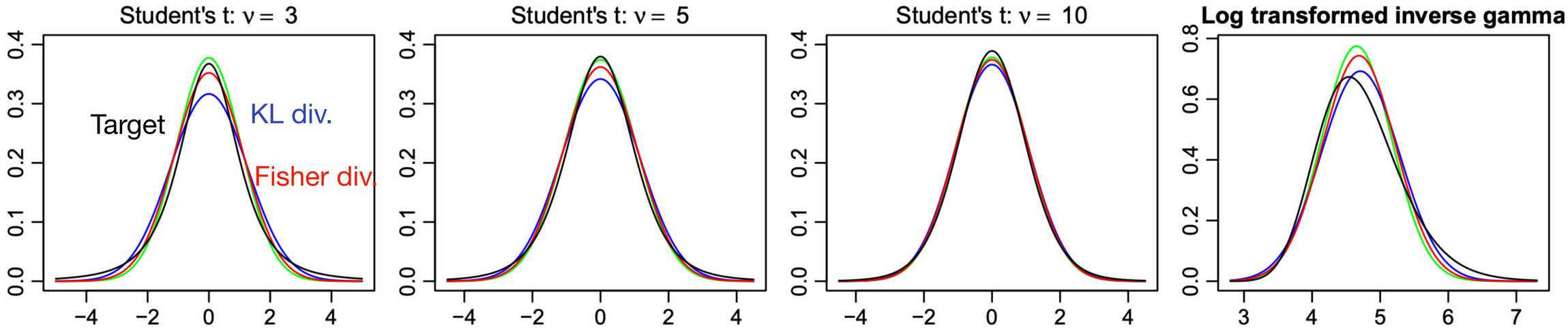
Yang et al. Variational approximations using Fisher divergence. arXiv, 2019.

Chen et al. Weighted Fisher divergence for high-dimensional Gaussian variational inference. arXiv, 2025

Fisher divergence

Fisher Divergence: measures the agreement between the ***scores*** of q and π:

$$\mathscr{D}(q;\pi) = \int \left\| \nabla_z \log q(z) - \nabla_z \log \pi(z) \right\|^2 q(z)\, dz$$



Hyvarinen. Estimation of Non-Normalized Statistical Models by Score Matching. *JMLR*, 2005.

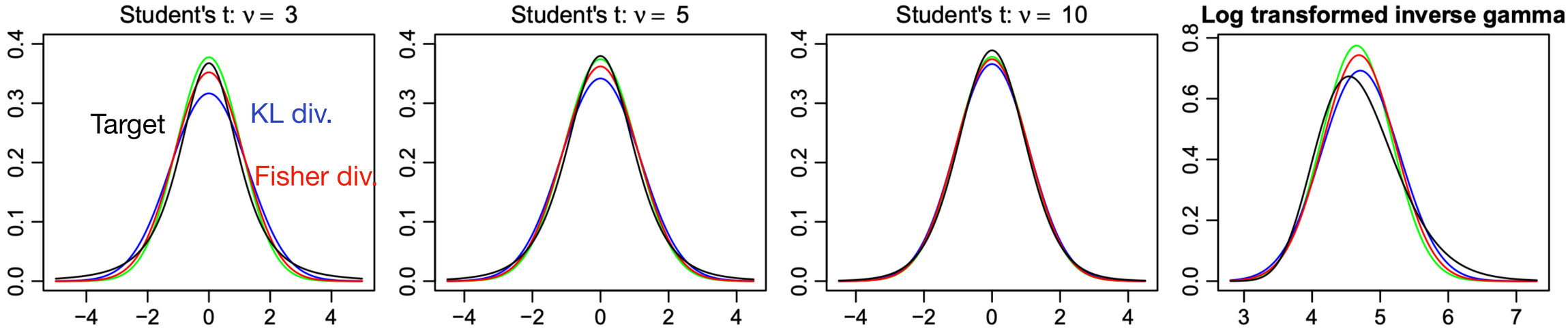
Yang et al. Variational approximations using Fisher divergence. arXiv, 2019.

Chen et al. Weighted Fisher divergence for high-dimensional Gaussian variational inference. arXiv, 2025

Fisher divergence

Fisher Divergence: measures the agreement between the ***scores*** of q and π:

$$\mathscr{D}(q;\pi) = \int \left\| \nabla_z \log q(z) - \nabla_z \log \pi(z) \right\|^2 q(z)\, dz$$



How do we optimize this divergence?

"Reparameterization trick"+SGD? Same issues as ADVI.

Hyvarinen. Estimation of Non-Normalized Statistical Models by Score Matching. *JMLR*, 2005.

Yang et al. Variational approximations using Fisher divergence. arXiv, 2019.

Chen et al. Weighted Fisher divergence for high-dimensional Gaussian variational inference. arXiv, 2025

Weighted Fisher divergence & full-covariance Gaussians

Divergence: measure the agreement between the ***scores*** of q and π:

$$\mathscr{D}(q;\pi) = \int \left\| \nabla_z \log q(z) - \nabla_z \log \pi(z) \right\|^2_{\mathrm{Cov}(q)} q(z)\, dz$$

$$\|x\|^2_\Sigma := x^\top \Sigma\, x$$

Cai et al. Batch and match: black-box variational inference with a score-based divergence. *ICML* 2024.

Weighted Fisher divergence & full-covariance Gaussians

Divergence: measure the agreement between the ***scores*** of q and π:

$$\mathscr{D}(q;\pi) = \int \left\| \nabla_z \log q(z) - \nabla_z \log \pi(z) \right\|^2_{\mathrm{Cov}(q)} q(z)\, dz$$

$$\|x\|^2_\Sigma := x^\top \Sigma\, x$$

Properties:

- Non-negativity ($\mathscr{D} \geq 0$ with equality iff $p = q$)
- Invariant to affine transformations $\Longrightarrow$ leads to closed-form solutions (not true for Fisher div.)

Cai et al. Batch and match: black-box variational inference with a score-based divergence. *ICML* 2024.

Weighted Fisher divergence & full-covariance Gaussians

Divergence: measure the agreement between the ***scores*** of q and π:

$$\mathscr{D}(q;\pi) = \int \left\| \nabla_z \log q(z) - \nabla_z \log \pi(z) \right\|^2_{\mathrm{Cov}(q)} q(z)\, dz$$

$$\|x\|^2_\Sigma := x^\top \Sigma\, x$$

Properties:

- Non-negativity ($\mathscr{D} \geq 0$ with equality iff $p = q$)
- Invariant to affine transformations $\Longrightarrow$ leads to closed-form solutions (not true for Fisher div.)

How do we optimize this divergence? Again could do “reparameterization trick” +SGD, but same issues as ADVI.

Cai et al. Batch and match: black-box variational inference with a score-based divergence. *ICML* 2024.

Observation for Gaussians: scores contain a lot of structure

Covariance-weighted Fisher divergence:

$$\mathscr{D}(q;\pi) = \int \left\| \nabla_z \log q(z) - \nabla_z \log \pi(z) \right\|^2_{\mathrm{Cov}(q)} q(z)\, dz$$

Observation for Gaussians: scores contain a lot of structure

Covariance-weighted Fisher divergence:

$$\mathscr{D}(q;\pi) = \int \left\| \nabla_z \log q(z) - \nabla_z \log \pi(z) \right\|^2_{\mathrm{Cov}(q)} q(z)\, dz$$

Form an empirical estimate:

Sample $z^1, \ldots, z^B$ from a distribution ν (that is not q)

Observation for Gaussians: scores contain a lot of structure

Covariance-weighted Fisher divergence:

$$\mathscr{D}(q;\pi) = \int \left\| \nabla_z \log q(z) - \nabla_z \log \pi(z) \right\|^2_{\mathrm{Cov}(q)} q(z)\, dz$$

Form an empirical estimate:

Sample $z^1, \ldots, z^B$ from a distribution ν (that is not q)

$$\widehat{\mathscr{D}}(q;\pi) = \frac{1}{B} \sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

Observation for Gaussians: scores contain a lot of structure

Covariance-weighted Fisher divergence:

$$\mathscr{D}(q;\pi) = \int \left\| \nabla_z \log q(z) - \nabla_z \log \pi(z) \right\|^2_{\mathrm{Cov}(q)} q(z)\, dz$$

Form an empirical estimate:

Sample $z^1, \ldots, z^B$ from a distribution ν (that is not q)

$$\widehat{\mathscr{D}}(q;\pi) = \frac{1}{B} \sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

If π is Gaussian and $B > D$,

Observation for Gaussians: scores contain a lot of structure

Covariance-weighted Fisher divergence:

$$\mathscr{D}(q;\pi) = \int \left\| \nabla_z \log q(z) - \nabla_z \log \pi(z) \right\|^2_{\mathrm{Cov}(q)} q(z)\, dz$$

Form an empirical estimate:

Sample $z^1, \ldots, z^B$ from a distribution ν (that is not q)

$$\widehat{\mathscr{D}}\,(q;\pi) = \frac{1}{B}\sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

If π is Gaussian and $B > D$,

$\hat{\mu}, \hat{\Sigma} = \arg\min_{\mu,\Sigma} \widehat{\mathscr{D}}\,(q_{\mu,\Sigma};\pi)$ recovers the exact mean and covariance!

Observation for Gaussians: scores contain a lot of structure

Covariance-weighted Fisher divergence:

$$\mathscr{D}(q;\pi) = \int \left\| \nabla_z \log q(z) - \nabla_z \log \pi(z) \right\|^2_{\mathrm{Cov}(q)} q(z)\, dz$$

Form an empirical estimate:

Sample $z^1, \ldots, z^B$ from a distribution ν (that is not q)

$$\widehat{\mathscr{D}}(q;\pi) = \frac{1}{B}\sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

If π is Gaussian and $B > D$,

$\hat{\mu}, \hat{\Sigma} = \arg\min_{\mu,\Sigma} \widehat{\mathscr{D}}(q_{\mu,\Sigma};\pi)$ recovers the exact mean and covariance!

E.g., for $D = 1$, only need 2 points! Compare to: 2 *samples* has error $\mathcal{O}(1/\sqrt{2})$

Observation for Gaussians: scores contain a lot of structure

Covariance-weighted Fisher divergence:

$$\mathscr{D}(q;\pi) = \int \left\| \nabla_z \log q(z) - \nabla_z \log \pi(z) \right\|^2_{\mathrm{Cov}(q)} q(z)\, dz$$

Form an empirical estimate:

Sample $z^1, \ldots, z^B$ from a distribution ν (that is not q)

$$\widehat{\mathscr{D}}(q;\pi) = \frac{1}{B}\sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

If π is Gaussian and $B > D$,

$\hat{\mu}, \hat{\Sigma} = \arg\min_{\mu,\Sigma} \widehat{\mathscr{D}}(q_{\mu,\Sigma};\pi)$ recovers the exact mean and covariance!

E.g., for $D = 1$, only need 2 points! Compare to: 2 *samples* has error $\mathcal{O}(1/\sqrt{2})$

How to turn this into an algorithm? Need a sampling distribution.

Score-based variational inference

Score-based VI: Want $q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q; \pi)$

Monte Carlo estimate (unbiased):

Sample $z^1, \ldots, z^B$ from q

$$\widehat{\mathscr{D}}(q; \pi) = \frac{1}{B} \sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

Score-based variational inference

Score-based VI: Want $q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q; \pi)$

Monte Carlo estimate (unbiased):

Sample $z^1, \ldots, z^B$ from q

$$\widehat{\mathscr{D}}(q; \pi) = \frac{1}{B} \sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

But: cannot simultaneously sample from and optimize over q!

Score-based variational inference

Score-based VI: Want $q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q; \pi)$

Monte Carlo estimate (unbiased):

Sample $z^1, \ldots, z^B$ from q

$$\widehat{\mathscr{D}}(q; \pi) = \frac{1}{B} \sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

But: cannot simultaneously sample from and optimize over q!

Iterative procedure:

Score-based variational inference

Score-based VI: Want $q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q; \pi)$

Monte Carlo estimate (unbiased):

Sample $z^1, \ldots, z^B$ from q

$$\widehat{\mathscr{D}}(q; \pi) = \frac{1}{B} \sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

But: cannot simultaneously sample from and optimize over q!

Iterative procedure:

Score-based variational inference

Score-based VI: Want $q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q; \pi)$

Monte Carlo estimate (unbiased):

Sample $z^1, \ldots, z^B$ from q

$$\widehat{\mathscr{D}}(q; \pi) = \frac{1}{B} \sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

But: cannot simultaneously sample from and optimize over q!

Iterative procedure:

Score-based variational inference

Score-based VI: Want $q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q; \pi)$

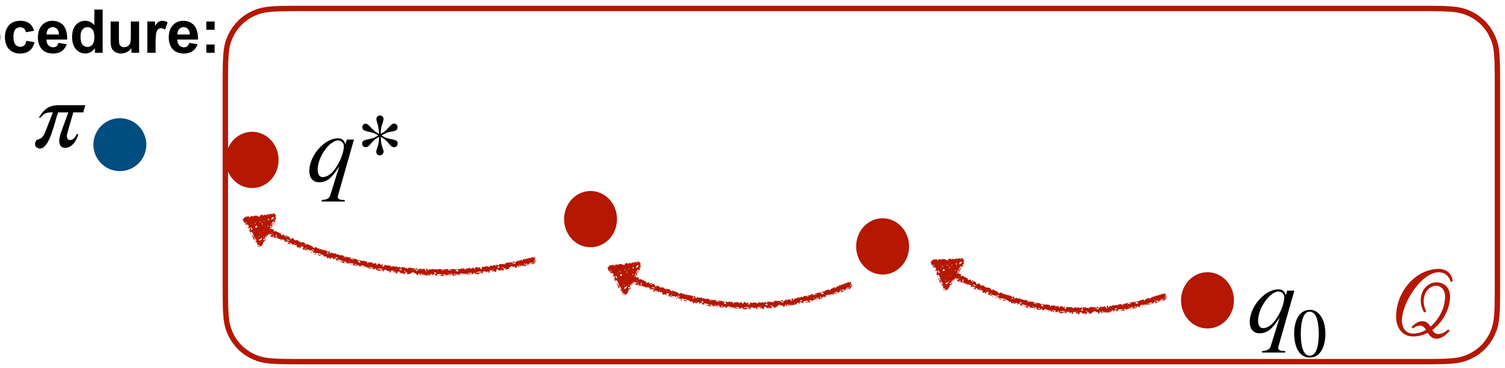
Monte Carlo estimate (unbiased):

Sample $z^1, \ldots, z^B$ from q

$$\widehat{\mathscr{D}}(q; \pi) = \frac{1}{B} \sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

But: cannot simultaneously sample from and optimize over q!

Iterative procedure:



Score-based variational inference

Score-based VI: Want $q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q; \pi)$

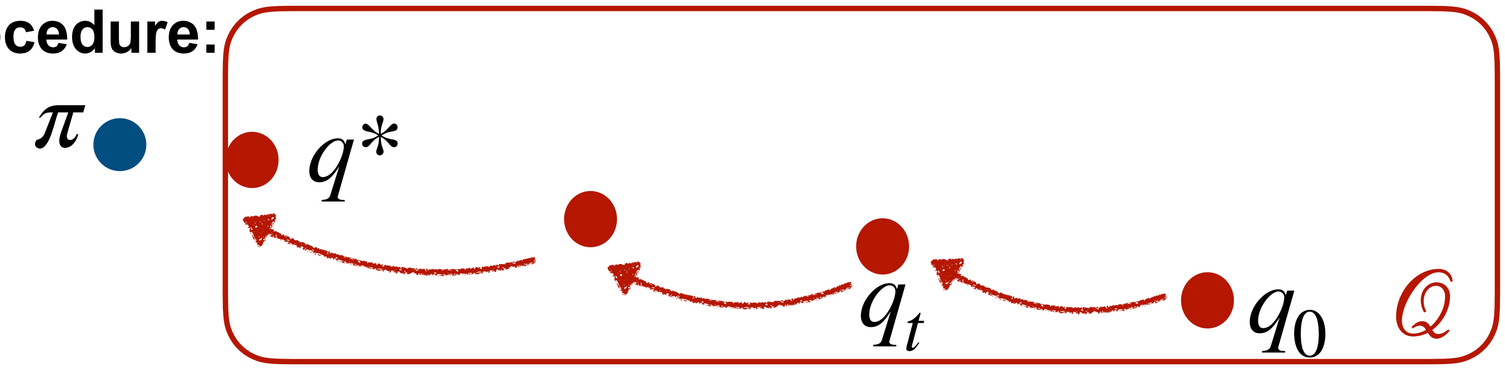
Monte Carlo estimate (unbiased):

Sample $z^1, \ldots, z^B$ from q

$$\widehat{\mathscr{D}}(q; \pi) = \frac{1}{B} \sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

But: cannot simultaneously sample from and optimize over q!

Iterative procedure:



Score-based variational inference

Score-based VI: Want $q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q; \pi)$

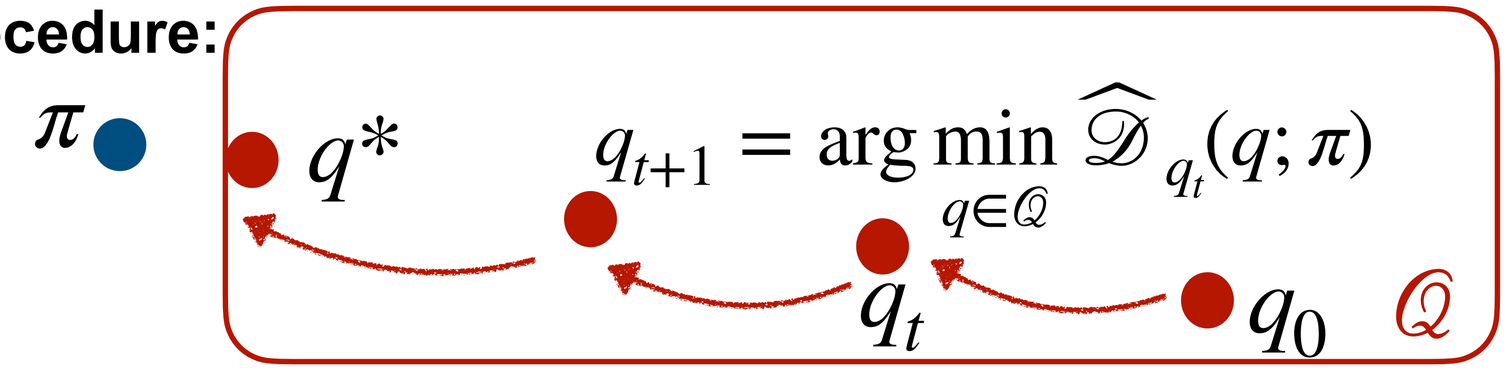
Monte Carlo estimate (unbiased):

Sample $z^1, \ldots, z^B$ from q

$$\widehat{\mathscr{D}}(q; \pi) = \frac{1}{B} \sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

But: cannot simultaneously sample from and optimize over q!

Iterative procedure:



Score-based variational inference

Score-based VI: Want $q^* = \arg\min_{q \in \mathcal{Q}} \mathcal{D}(q; \pi)$

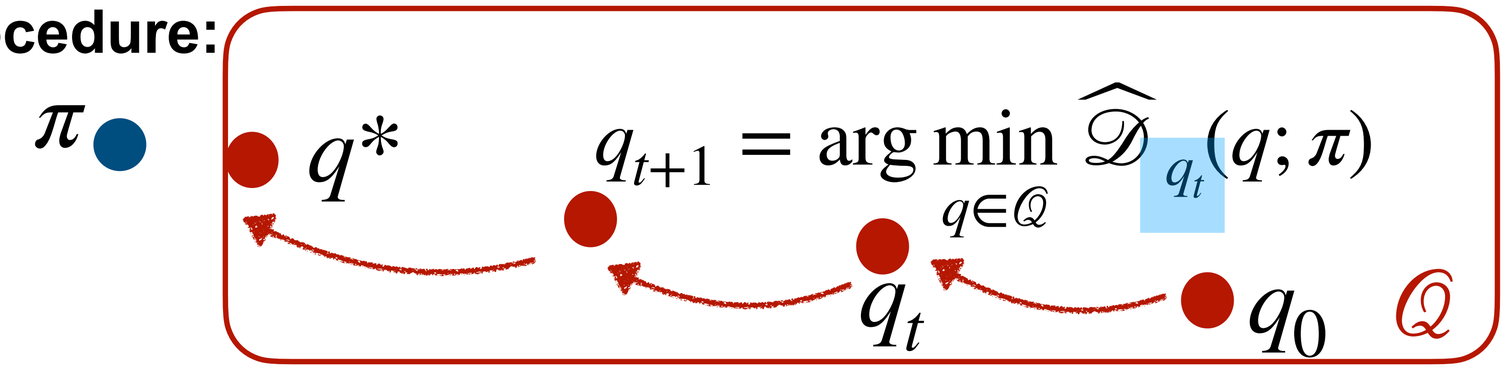
Monte Carlo estimate (unbiased):

Sample $z^1, \ldots, z^B$ from q

$$\widehat{\mathcal{D}}(q; \pi) = \frac{1}{B} \sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

But: cannot simultaneously sample from and optimize over q!

Iterative procedure:



Score-based variational inference

Score-based VI: Want $q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q; \pi)$

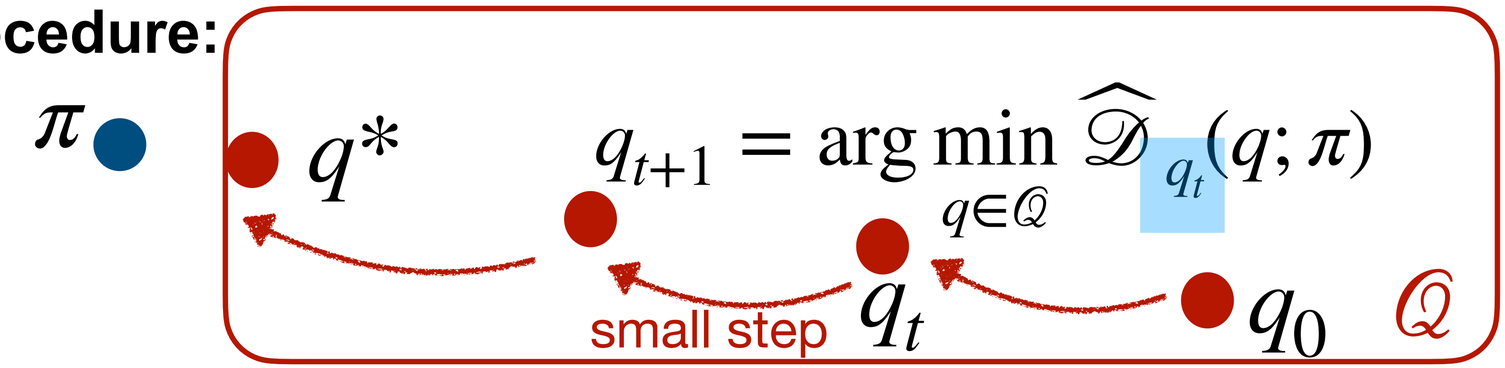
Monte Carlo estimate (unbiased):

Sample $z^1, \ldots, z^B$ from q

$$\widehat{\mathscr{D}}(q; \pi) = \frac{1}{B} \sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

But: cannot simultaneously sample from and optimize over q!

Iterative procedure:



Introduces bias: ***regularize*** the score divergence so we don't move too far from q_t

Score-based variational inference

Score-based VI: Want $q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q; \pi)$

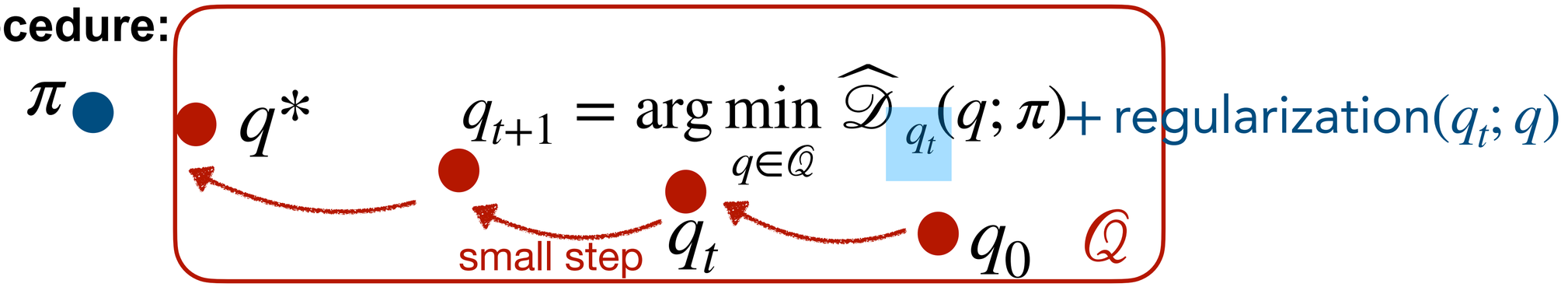
Monte Carlo estimate (unbiased):

Sample $z^1, \ldots, z^B$ from q

$$\widehat{\mathscr{D}}(q; \pi) = \frac{1}{B} \sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

But: cannot simultaneously sample from and optimize over q!

Iterative procedure:



Introduces bias: ***regularize*** the score divergence so we don't move too far from q_t

Score-based variational inference

Score-based VI: Want $q^* = \arg\min_{q \in \mathcal{Q}} \mathscr{D}(q; \pi)$

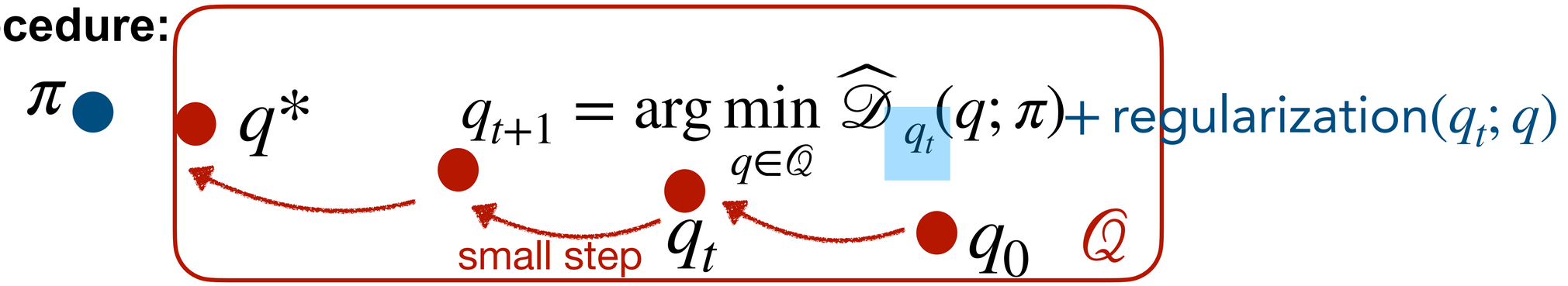
Monte Carlo estimate (unbiased):

Sample $z^1, \ldots, z^B$ from q

$$\widehat{\mathscr{D}}(q; \pi) = \frac{1}{B} \sum_{b=1}^{B} \left\| \nabla_z \log q(z^b) - \nabla_z \log \pi(z^b) \right\|^2_{\mathrm{Cov}(q)}$$

But: cannot simultaneously sample from and optimize over q!

Iterative procedure:



Introduces bias: ***regularize*** the score divergence so we don't move too far from q_t

Proximal point algorithm:

Here, the ***global minimum*** of the subproblem can be computed analytically!

Batch and match for full covariance Gaussians

Iterate:

Batch step: given current variational estimate q_t

Sample a *batch* of points $z^1, \ldots, z^B$ (call B the *batch size*)

Compute *empirical* score-based divergence $\widehat{\mathscr{D}}_{q_t}(q; \pi)$ on batch of points

Match step:

Fit new variational distribution $q_{t+1} = \arg\min_{q \in \mathcal{Q}} \widehat{\mathscr{D}}_{q_t}(q; \pi) + \frac{2}{\lambda_t} \mathrm{KL}(q_t; q)$

Batch and match for full covariance Gaussians

Iterate:

Batch step: given current variational estimate q_t

Sample a *batch* of points $z^1, \ldots, z^B$ (call B the *batch size*)

Compute *empirical* score-based divergence $\widehat{\mathscr{D}}_{q_t}(q;\pi)$ on batch of points

Match step:

Fit new variational distribution $q_{t+1} = \arg\min_{q \in \mathcal{Q}} \widehat{\mathscr{D}}_{q_t}(q;\pi) + \frac{2}{\lambda_t}\mathrm{KL}(q_t;q)$

empirical divergence

Batch and match for full covariance Gaussians

Iterate:

Batch step: given current variational estimate q_t

Sample a *batch* of points $z^1, \ldots, z^B$ (call B the *batch size*)

Compute *empirical* score-based divergence $\widehat{\mathscr{D}}_{q_t}(q;\pi)$ on batch of points

Match step:

Fit new variational distribution $q_{t+1} = \arg\min_{q \in \mathcal{Q}} \widehat{\mathscr{D}}_{q_t}(q;\pi) + \frac{2}{\lambda_t}\mathrm{KL}(q_t;q)$

empirical divergence

"learning rate"

Batch and match for full covariance Gaussians

Iterate:

Batch step: given current variational estimate q_t

Sample a *batch* of points $z^1, \ldots, z^B$ (call B the *batch size*)

Compute *empirical* score-based divergence $\widehat{\mathscr{D}}_{q_t}(q;\pi)$ on batch of points

Match step:

Fit new variational distribution $q_{t+1} = \arg\min_{q \in \mathcal{Q}} \widehat{\mathscr{D}}_{q_t}(q;\pi) + \frac{2}{\lambda_t}\mathrm{KL}(q_t;q)$

empirical divergence

"learning rate"

regularizer

Batch and match for full covariance Gaussians

Iterate:

Batch step: given current variational estimate q_t

Sample a *batch* of points $z^1, \ldots, z^B$ (call B the *batch size*)

Compute *empirical* score-based divergence $\widehat{\mathscr{D}}_{q_t}(q;\pi)$ on batch of points

Match step:

Fit new variational distribution $q_{t+1} = \arg\min_{q \in \mathcal{Q}} \widehat{\mathscr{D}}_{q_t}(q;\pi) + \frac{2}{\lambda_t}\mathrm{KL}(q_t;q)$

empirical divergence — "learning rate" — regularizer

In this problem, the ***global minimum*** of the subproblem can be computed analytically!

Batch and match for full covariance Gaussians

Iterate:

Batch step: given current variational estimate q_t

Sample a *batch* of points $z^1, \ldots, z^B$ (call B the *batch size*)

Compute *empirical* score-based divergence $\widehat{\mathscr{D}}_{q_t}(q; \pi)$ on batch of points

Match step:

Fit new variational distribution $q_{t+1} = \arg\min_{q \in \mathcal{Q}} \widehat{\mathscr{D}}_{q_t}(q; \pi) + \frac{2}{\lambda_t} \mathrm{KL}(q_t; q)$

empirical divergence

"learning rate"

regularizer

In this problem, the ***global minimum*** of the subproblem can be computed analytically!

Σ_{t+1} update: solve a *quadratic matrix equation*

$$\Sigma_{t+1} U_t \Sigma_{t+1} + \Sigma_{t+1} = V_t$$

Batch and match for full covariance Gaussians

Iterate:

Batch step: given current variational estimate q_t

Sample a *batch* of points $z^1, \ldots, z^B$ (call B the *batch size*)

Compute *empirical* score-based divergence $\widehat{\mathscr{D}}_{q_t}(q;\pi)$ on batch of points

Match step:

Fit new variational distribution $q_{t+1} = \arg\min_{q \in \mathcal{Q}} \widehat{\mathscr{D}}_{q_t}(q;\pi) + \frac{2}{\lambda_t}\mathrm{KL}(q_t;q)$

empirical divergence — "learning rate" — regularizer

In this problem, the ***global minimum*** of the subproblem can be computed analytically!

Σ_{t+1} update: solve a *quadratic matrix equation*

$$\Sigma_{t+1} U_t \Sigma_{t+1} + \Sigma_{t+1} = V_t$$

Matrices that depend on current covariance, statistics of the batch and scores

Batch and match for full covariance Gaussians

Iterate:

Batch step: given current variational estimate q_t

Sample a *batch* of points $z^1, \ldots, z^B$ (call B the *batch size*)

Compute *empirical* score-based divergence $\widehat{\mathscr{D}}_{q_t}(q;\pi)$ on batch of points

Match step:

Fit new variational distribution $q_{t+1} = \arg\min_{q \in \mathcal{Q}} \widehat{\mathscr{D}}_{q_t}(q;\pi) + \frac{2}{\lambda_t}\mathrm{KL}(q_t;q)$

empirical divergence — "learning rate" — regularizer

In this problem, the ***global minimum*** of the subproblem can be computed analytically!

Σ_{t+1} update: solve a *quadratic matrix equation*

$$\Sigma_{t+1} U_t \Sigma_{t+1} + \Sigma_{t+1} = V_t$$

Matrices that depend on current covariance, statistics of the batch and scores

mean update:

$$\mu_{t+1} = \mu_t + \frac{\lambda_t}{1+\lambda_t}\Sigma_{t+1}\,\bar{g}$$

Batch and match for full covariance Gaussians

Iterate:

Batch step: given current variational estimate q_t

Sample a *batch* of points $z^1, \ldots, z^B$ (call B the *batch size*)

Compute *empirical* score-based divergence $\widehat{\mathscr{D}}_{q_t}(q;\pi)$ on batch of points

Match step:

Fit new variational distribution $q_{t+1} = \arg\min_{q \in \mathcal{Q}} \widehat{\mathscr{D}}_{q_t}(q;\pi) + \frac{2}{\lambda_t}\mathrm{KL}(q_t;q)$

empirical divergence — "learning rate" — regularizer

In this problem, the ***global minimum*** of the subproblem can be computed analytically!

Σ_{t+1} update: solve a *quadratic matrix equation*

$$\Sigma_{t+1} U_t \Sigma_{t+1} + \Sigma_{t+1} = V_t$$

Matrices that depend on current covariance, statistics of the batch and scores

mean update:

$$\mu_{t+1} = \mu_t + \frac{\lambda_t}{1+\lambda_t}\Sigma_{t+1}\,\bar{g}$$

mean of scores computed on the batch

Application: deep generative modeling

Data: high-dimensional object (image) $x_n \in \mathbb{R}^{3072}$ 

Lower-dimensional latent representation $z_n \in \mathbb{R}^{256}$

Cai et al. Batch and match: black-box variational inference with a score-based divergence. *ICML* 2024.

Application: deep generative modeling

Deep generative model:

Data: high-dimensional object (image) $x_n \in \mathbb{R}^{3072}$ 

Lower-dimensional latent representation $z_n \in \mathbb{R}^{256}$

Cai et al. Batch and match: black-box variational inference with a score-based divergence. *ICML* 2024.

Application: deep generative modeling

Data: high-dimensional object (image) $x_n \in \mathbb{R}^{3072}$

Lower-dimensional latent representation $z_n \in \mathbb{R}^{256}$

Deep generative model:

$$z_n \sim \mathcal{N}(0, I)$$

$$x_n \mid z_n \sim \mathcal{N}(\Omega(z_n, \hat{\theta}), \sigma^2 I)$$

Cai et al. Batch and match: black-box variational inference with a score-based divergence. *ICML* 2024.

Application: deep generative modeling

Data: high-dimensional object (image) $x_n \in \mathbb{R}^{3072}$

Lower-dimensional latent representation $z_n \in \mathbb{R}^{256}$

Deep generative model:

$$z_n \sim \mathcal{N}(0, I)$$

$$x_n \mid z_n \sim \mathcal{N}(\Omega(z_n, \hat{\theta}), \sigma^2 I)$$

Parameters of the neural network
(pre-trained to maximize likelihood)

Cai et al. Batch and match: black-box variational inference with a score-based divergence. *ICML* 2024.

Application: deep generative modeling

Data: high-dimensional object (image) $x_n \in \mathbb{R}^{3072}$ 

Lower-dimensional latent representation $z_n \in \mathbb{R}^{256}$

Deep generative model:

$$z_n \sim \mathcal{N}(0, I)$$

$$x_n \mid z_n \sim \mathcal{N}(\Omega(z_n, \hat{\theta}), \sigma^2 I)$$

Parameters of the neural network (pre-trained to maximize likelihood)

Problem: given a test image x', approximate $p(z' \mid x')$ using VI

Cai et al. Batch and match: black-box variational inference with a score-based divergence. *ICML* 2024.

Application: deep generative modeling

Data: high-dimensional object (image) $x_n \in \mathbb{R}^{3072}$ 

Lower-dimensional latent representation $z_n \in \mathbb{R}^{256}$

Deep generative model:

$$z_n \sim \mathcal{N}(0, I)$$

$$x_n \mid z_n \sim \mathcal{N}(\Omega(z_n, \hat{\theta}), \sigma^2 I)$$

Parameters of the neural network (pre-trained to maximize likelihood)

Problem: given a test image x', approximate $p(z' \mid x')$ using VI

Reconstruct x' by feeding the posterior mean estimate into the neural network.

Cai et al. Batch and match: black-box variational inference with a score-based divergence. *ICML* 2024.

Application: deep generative modeling

Data: high-dimensional object (image) $x_n \in \mathbb{R}^{3072}$

Lower-dimensional latent representation $z_n \in \mathbb{R}^{256}$

Deep generative model:

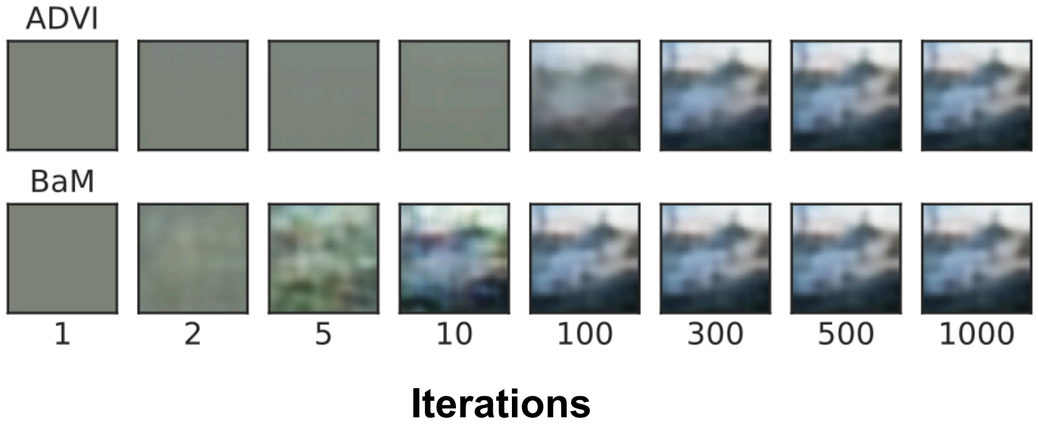
$$z_n \sim \mathcal{N}(0, I)$$

$$x_n \mid z_n \sim \mathcal{N}(\Omega(z_n, \hat{\theta}), \sigma^2 I)$$

Parameters of the neural network (pre-trained to maximize likelihood)

Problem: given a test image x', approximate $p(z' \mid x')$ using VI

Reconstruct x' by feeding the posterior mean estimate into the neural network.



Cai et al. Batch and match: black-box variational inference with a score-based divergence. *ICML* 2024.

Application: deep generative modeling

Data: high-dimensional object (image) $x_n \in \mathbb{R}^{3072}$

Lower-dimensional latent representation $z_n \in \mathbb{R}^{256}$

Deep generative model:

$$z_n \sim \mathcal{N}(0, I)$$

$$x_n \mid z_n \sim \mathcal{N}(\Omega(z_n, \hat{\theta}), \sigma^2 I)$$

Parameters of the neural network (pre-trained to maximize likelihood)

Problem: given a test image x', approximate $p(z' \mid x')$ using VI

Reconstruct x' by feeding the posterior mean estimate into the neural network.

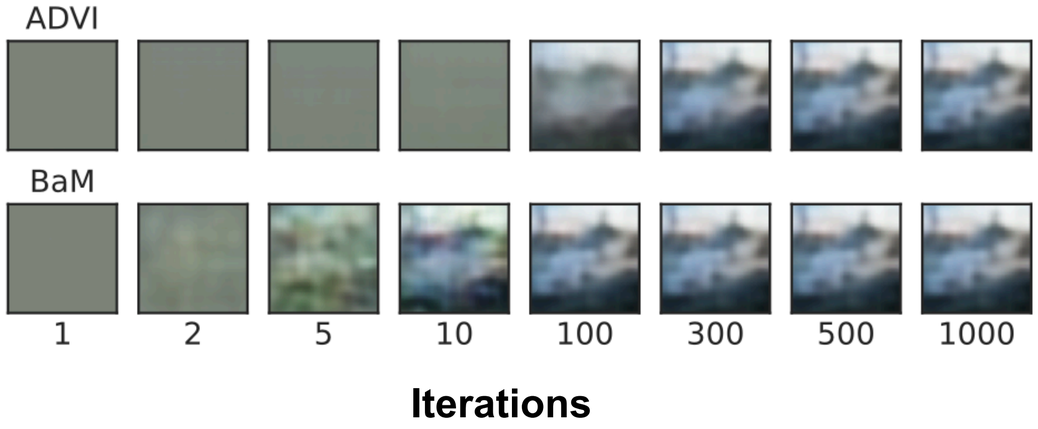
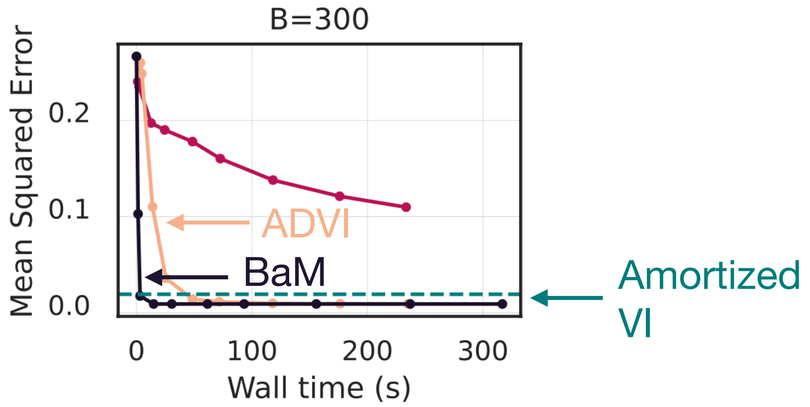


Image reconstruction error



Cai et al. Batch and match: black-box variational inference with a score-based divergence. *ICML* 2024.

Application: deep generative modeling

Data: high-dimensional object (image) $x_n \in \mathbb{R}^{3072}$

Lower-dimensional latent representation $z_n \in \mathbb{R}^{256}$

Deep generative model:

$$z_n \sim \mathcal{N}(0, I)$$

$$x_n \mid z_n \sim \mathcal{N}(\Omega(z_n, \hat{\theta}), \sigma^2 I)$$

Parameters of the neural network (pre-trained to maximize likelihood)

Problem: given a test image x', approximate $p(z' \mid x')$ using VI

Reconstruct x' by feeding the posterior mean estimate into the neural network.

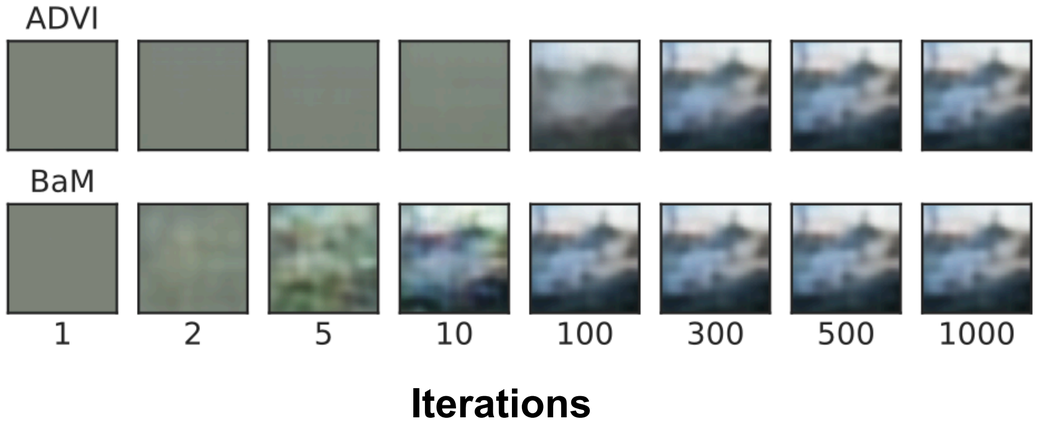
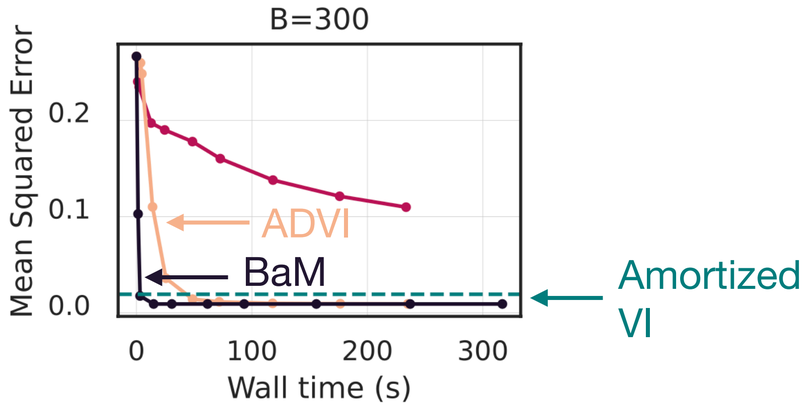


Image reconstruction error



BaM with $B = 300$ converges faster than ADVI with any batch size.

Cai et al. Batch and match: black-box variational inference with a score-based divergence. *ICML* 2024.

Extensions: high dimensional, large data sets

Cai, Modi, Margossian, Gower, Blei, Saul. EigenVI: score-based variational inference with orthogonal function expansions. *NeurIPS* 2024.
Modi & Cai, Saul. Batch, match, and patch: low-rank approximations for score-based variational inference. *AISTATS* 2025.

Extensions: high dimensional, large data sets

Assumes we can fit a covariance with $\mathcal{O}(D^2)$ parameters. What about in high dimensions?

- A "patch" step: we project the covariance update to low-rank + diagonal
- The resulting covariance update has $\mathcal{O}(D)$ cost

Cai, Modi, Margossian, Gower, Blei, Saul. EigenVI: score-based variational inference with orthogonal function expansions. *NeurIPS* 2024.
Modi & Cai, Saul. Batch, match, and patch: low-rank approximations for score-based variational inference. *AISTATS* 2025.

Extensions: high dimensional, large data sets

Assumes we can fit a covariance with $\mathcal{O}(D^2)$ parameters. What about in high dimensions?

- A "patch" step: we project the covariance update to low-rank + diagonal
- The resulting covariance update has $\mathcal{O}(D)$ cost

How can we use score-based VI for Bayesian inference with massive data sets?

- Can use BaM with a noisy score $\hat{\nabla} \log p(z)$, computed from a subsampled data set

Cai, Modi, Margossian, Gower, Blei, Saul. EigenVI: score-based variational inference with orthogonal function expansions. *NeurIPS* 2024.
Modi & Cai, Saul. Batch, match, and patch: low-rank approximations for score-based variational inference. *AISTATS* 2025.

Extensions: high dimensional, large data sets

Assumes we can fit a covariance with $\mathcal{O}(D^2)$ parameters. What about in high dimensions?

- A "patch" step: we project the covariance update to low-rank + diagonal
- The resulting covariance update has $\mathcal{O}(D)$ cost

How can we use score-based VI for Bayesian inference with massive data sets?

- Can use BaM with a noisy score $\hat{\nabla} \log p(z)$, computed from a subsampled data set

What about richer variational families?

Cai, Modi, Margossian, Gower, Blei, Saul. EigenVI: score-based variational inference with orthogonal function expansions. *NeurIPS* 2024.
Modi & Cai, Saul. Batch, match, and patch: low-rank approximations for score-based variational inference. *AISTATS* 2025.

Extensions: high dimensional, large data sets

Assumes we can fit a covariance with $\mathcal{O}(D^2)$ parameters. What about in high dimensions?

- A "patch" step: we project the covariance update to low-rank + diagonal
- The resulting covariance update has $\mathcal{O}(D)$ cost

How can we use score-based VI for Bayesian inference with massive data sets?

- Can use BaM with a noisy score $\hat{\nabla} \log p(z)$, computed from a subsampled data set

What about richer variational families?

- EigenVI: orthogonal function expansions

Cai, Modi, Margossian, Gower, Blei, Saul. EigenVI: score-based variational inference with orthogonal function expansions. *NeurIPS* 2024.
Modi & Cai, Saul. Batch, match, and patch: low-rank approximations for score-based variational inference. *AISTATS* 2025.

Extensions: high dimensional, large data sets

Assumes we can fit a covariance with $\mathcal{O}(D^2)$ parameters. What about in high dimensions?

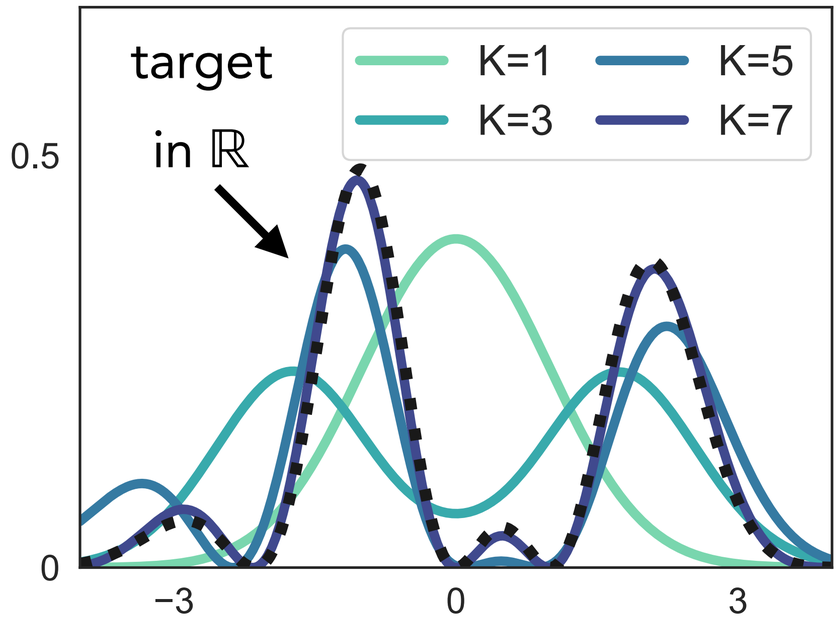
- A "patch" step: we project the covariance update to low-rank + diagonal
- The resulting covariance update has $\mathcal{O}(D)$ cost

How can we use score-based VI for Bayesian inference with massive data sets?

- Can use BaM with a noisy score $\hat{\nabla} \log p(z)$, computed from a subsampled data set

What about richer variational families?

- EigenVI: orthogonal function expansions



Cai, Modi, Margossian, Gower, Blei, Saul. EigenVI: score-based variational inference with orthogonal function expansions. *NeurIPS* 2024.
Modi & Cai, Saul. Batch, match, and patch: low-rank approximations for score-based variational inference. *AISTATS* 2025.

Extensions: high dimensional, large data sets

Assumes we can fit a covariance with $\mathcal{O}(D^2)$ parameters. What about in high dimensions?

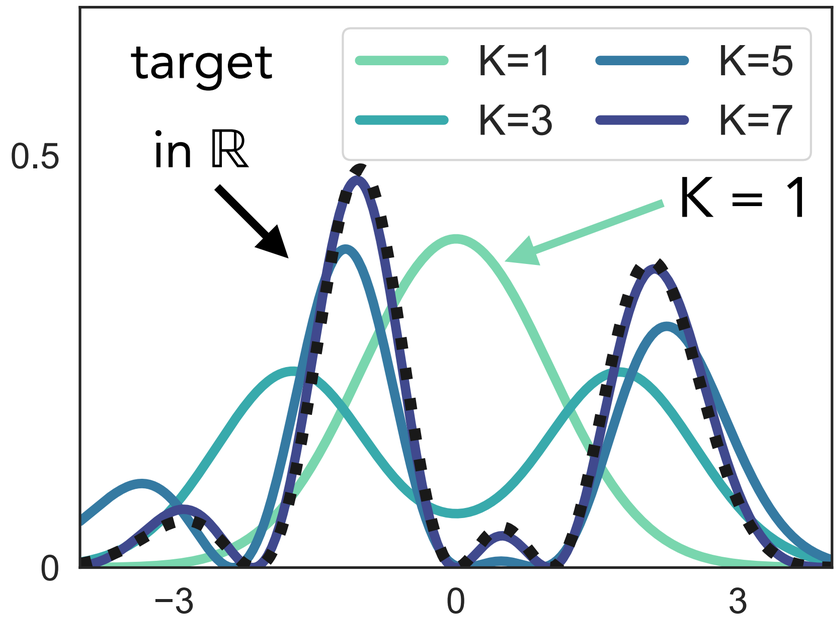
- A "patch" step: we project the covariance update to low-rank + diagonal
- The resulting covariance update has $\mathcal{O}(D)$ cost

How can we use score-based VI for Bayesian inference with massive data sets?

- Can use BaM with a noisy score $\hat{\nabla} \log p(z)$, computed from a subsampled data set

What about richer variational families?

- EigenVI: orthogonal function expansions



Cai, Modi, Margossian, Gower, Blei, Saul. EigenVI: score-based variational inference with orthogonal function expansions. *NeurIPS* 2024.
Modi & Cai, Saul. Batch, match, and patch: low-rank approximations for score-based variational inference. *AISTATS* 2025.

Extensions: high dimensional, large data sets

Assumes we can fit a covariance with $\mathcal{O}(D^2)$ parameters. What about in high dimensions?

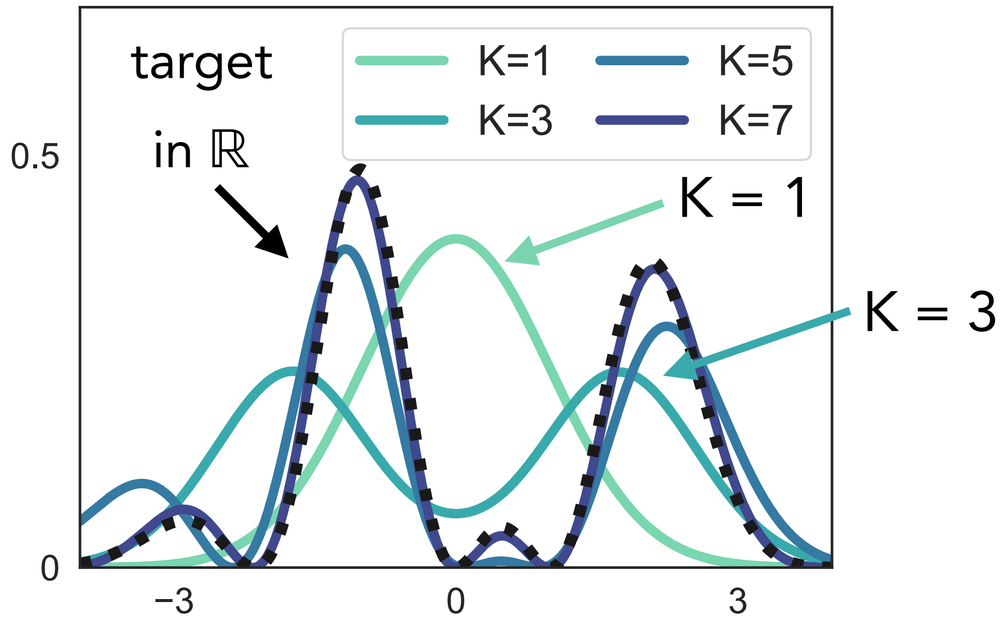
- A "patch" step: we project the covariance update to low-rank + diagonal
- The resulting covariance update has $\mathcal{O}(D)$ cost

How can we use score-based VI for Bayesian inference with massive data sets?

- Can use BaM with a noisy score $\hat{\nabla} \log p(z)$, computed from a subsampled data set

What about richer variational families?

- EigenVI: orthogonal function expansions



Cai, Modi, Margossian, Gower, Blei, Saul. EigenVI: score-based variational inference with orthogonal function expansions. *NeurIPS* 2024.
Modi & Cai, Saul. Batch, match, and patch: low-rank approximations for score-based variational inference. *AISTATS* 2025.

Extensions: high dimensional, large data sets

Assumes we can fit a covariance with $\mathcal{O}(D^2)$ parameters. What about in high dimensions?

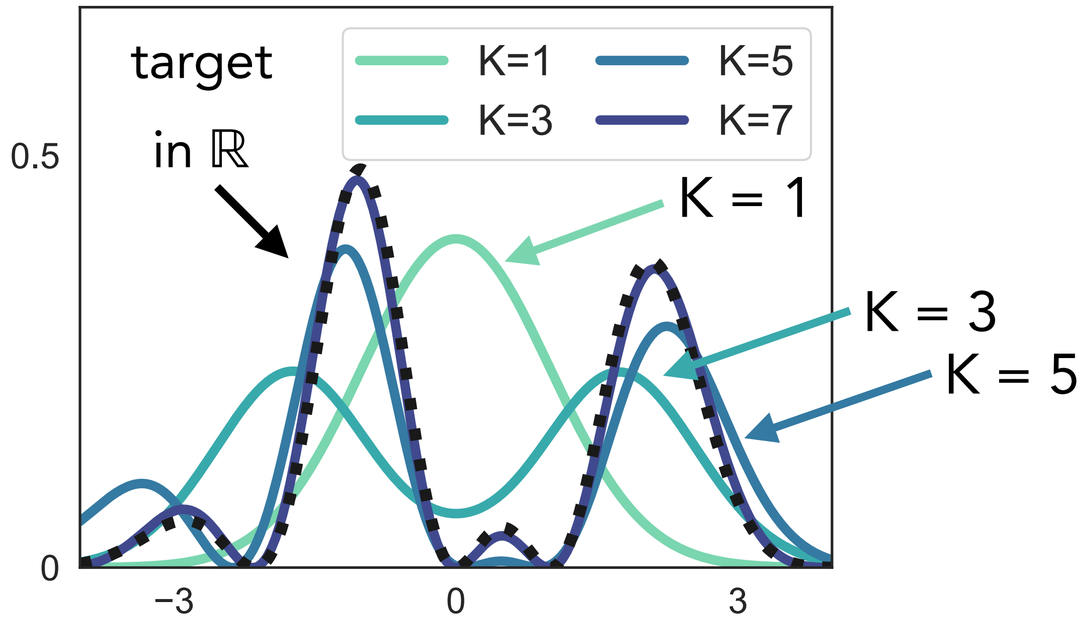
- A "patch" step: we project the covariance update to low-rank + diagonal
- The resulting covariance update has $\mathcal{O}(D)$ cost

How can we use score-based VI for Bayesian inference with massive data sets?

- Can use BaM with a noisy score $\hat{\nabla} \log p(z)$, computed from a subsampled data set

What about richer variational families?

- EigenVI: orthogonal function expansions



Cai, Modi, Margossian, Gower, Blei, Saul. EigenVI: score-based variational inference with orthogonal function expansions. *NeurIPS* 2024.
Modi & Cai, Saul. Batch, match, and patch: low-rank approximations for score-based variational inference. *AISTATS* 2025.

Extensions: high dimensional, large data sets

Assumes we can fit a covariance with $\mathcal{O}(D^2)$ parameters. What about in high dimensions?

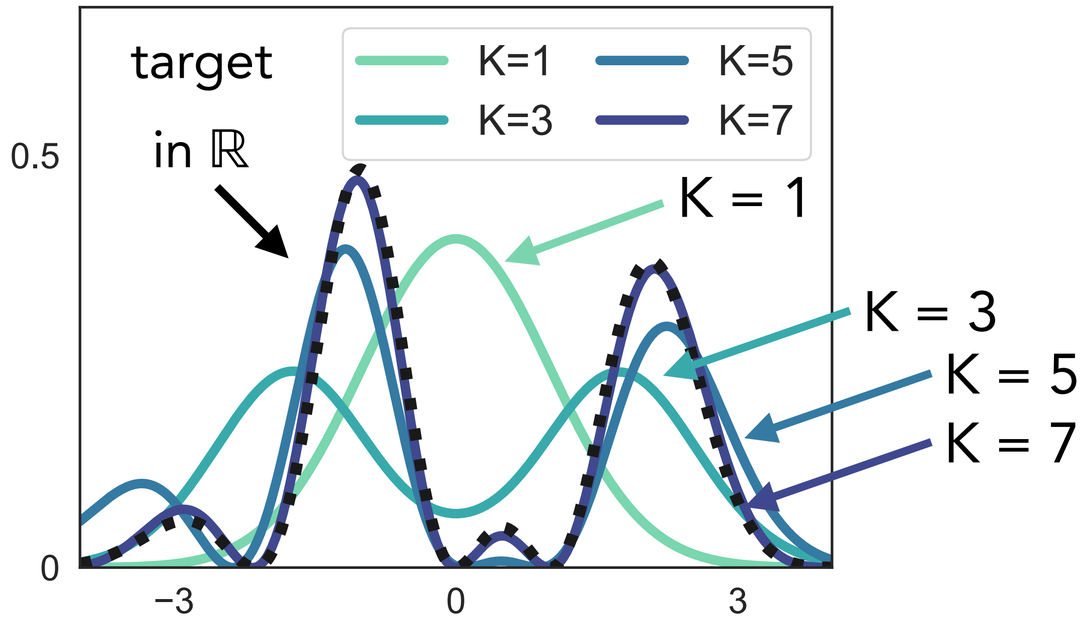
- A "patch" step: we project the covariance update to low-rank + diagonal
- The resulting covariance update has $\mathcal{O}(D)$ cost

How can we use score-based VI for Bayesian inference with massive data sets?

- Can use BaM with a noisy score $\hat{\nabla} \log p(z)$, computed from a subsampled data set

What about richer variational families?

- EigenVI: orthogonal function expansions



Cai, Modi, Margossian, Gower, Blei, Saul. EigenVI: score-based variational inference with orthogonal function expansions. *NeurIPS* 2024.

Modi & Cai, Saul. Batch, match, and patch: low-rank approximations for score-based variational inference. *AISTATS* 2025.

Extensions: high dimensional, large data sets

Assumes we can fit a covariance with $\mathcal{O}(D^2)$ parameters. What about in high dimensions?

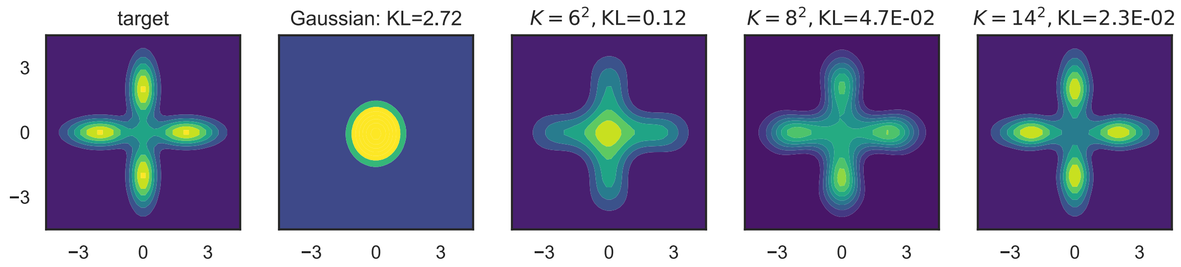
- A "patch" step: we project the covariance update to low-rank + diagonal
- The resulting covariance update has $\mathcal{O}(D)$ cost

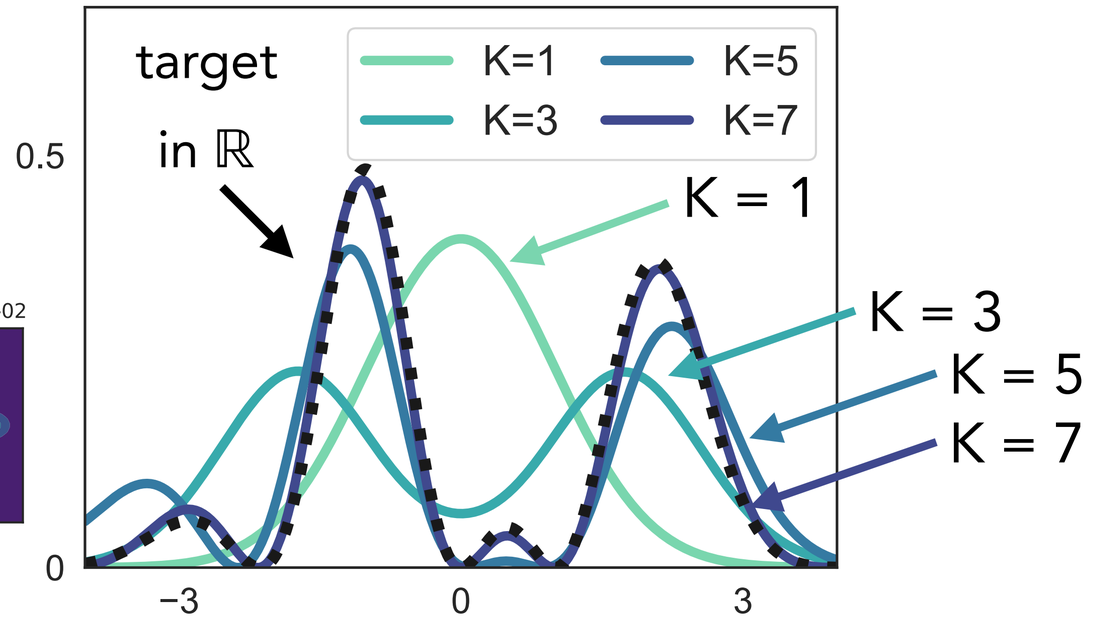
How can we use score-based VI for Bayesian inference with massive data sets?

- Can use BaM with a noisy score $\hat{\nabla} \log p(z)$, computed from a subsampled data set

What about richer variational families?

- EigenVI: orthogonal function expansions





Cai, Modi, Margossian, Gower, Blei, Saul. EigenVI: score-based variational inference with orthogonal function expansions. *NeurIPS* 2024.
Modi & Cai, Saul. Batch, match, and patch: low-rank approximations for score-based variational inference. *AISTATS* 2025.

Extensions: high dimensional, large data sets

Assumes we can fit a covariance with $\mathcal{O}(D^2)$ parameters. What about in high dimensions?

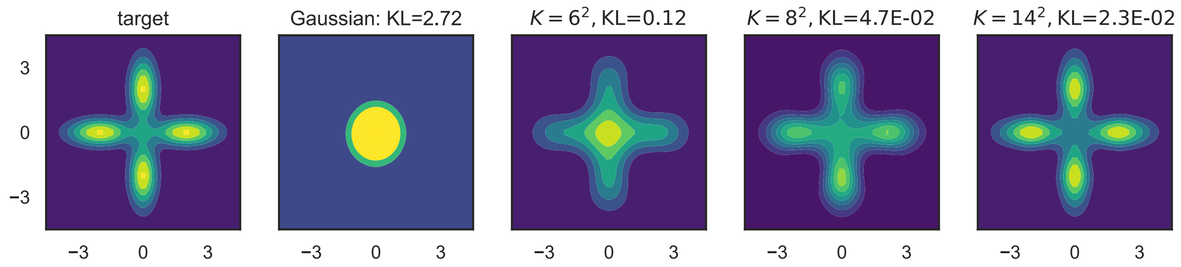
- A "patch" step: we project the covariance update to low-rank + diagonal
- The resulting covariance update has $\mathcal{O}(D)$ cost

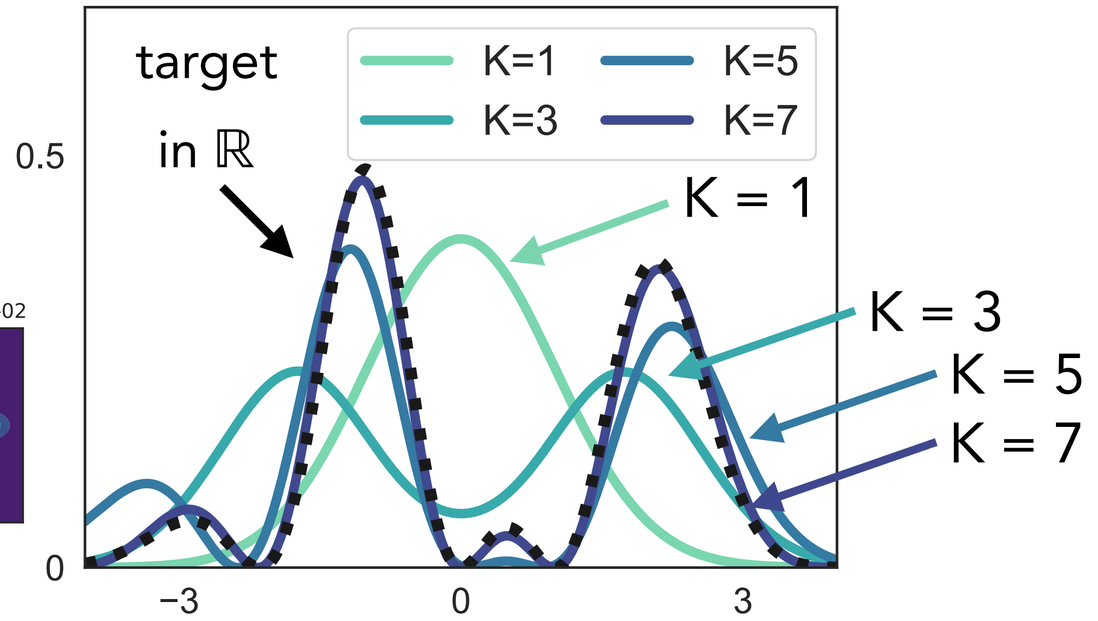
How can we use score-based VI for Bayesian inference with massive data sets?

- Can use BaM with a noisy score $\hat{\nabla} \log p(z)$, computed from a subsampled data set

What about richer variational families?

- EigenVI: orthogonal function expansions
- Products of experts





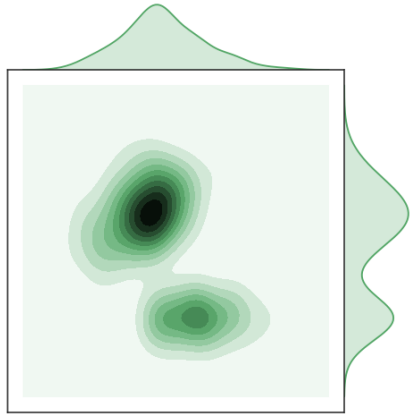
Cai, Modi, Margossian, Gower, Blei, Saul. EigenVI: score-based variational inference with orthogonal function expansions. *NeurIPS* 2024.
Modi & Cai, Saul. Batch, match, and patch: low-rank approximations for score-based variational inference. *AISTATS* 2025.

Summary of score-based VI

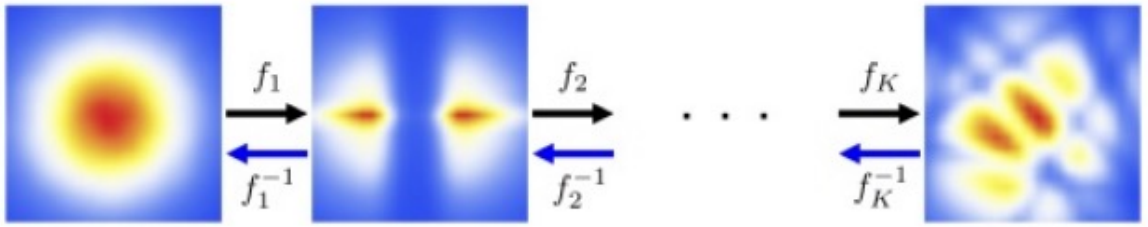
We discussed score-based variational inference

- black-box and only require computing the score of the target $\nabla_z \log \pi(z)$
- VI with score-based divergences, e.g., Fisher divergence
  - For Gaussian $\mathcal{Q}$, can apply reparameterization trick to optimize
- VI with a covariance-weighted Fisher divergence
  - Has some nice properties, including computational benefits
  - For Gaussian $\mathcal{Q}$, can also use reparameterization trick to optimize
  - Batch & match: Gaussian full covariance, closed-form proximal updates
    - Fast convergence for near-Gaussian targets
- Extensions of score-based VI to high dimensions, large data sets, and non-Gaussian families

Designing q Distributions



Auxiliary variables & mixture distributions



Normalizing flows

Use many layers (hierarchical) + continuous-time limit

Diffusion SDE & Continuous Normalizing Flow posteriors for approximate inference

Auxiliary Variables & Mixture Distributions

- Construct $q(z|x)$ as a (hierarchical) mixture distribution

$$q(z|x) = \int q(z\,|a,x)\; q(a|x)\; da$$

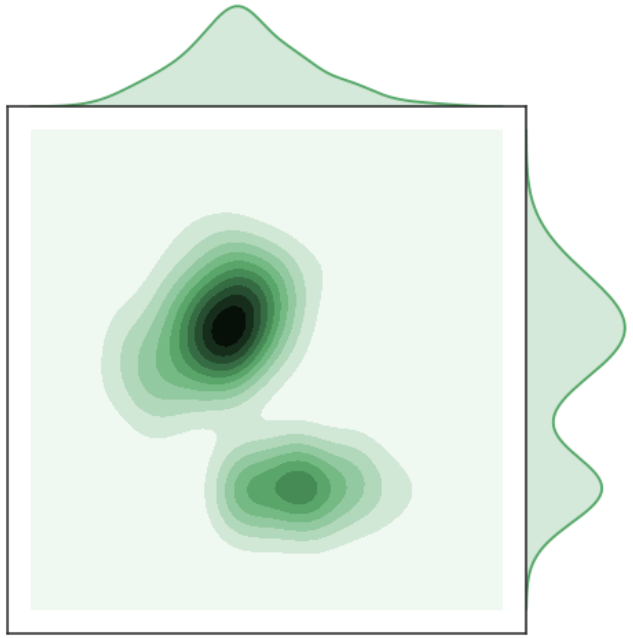
  - a is the auxiliary variable used to enrich the approximate posterior

  - Example: Mixture of Gaussians

$$a \sim q(a|x) = Categorical(\pi_1, \ldots, \pi_K)$$

$$z \sim q(z\,|a,x) = N(\theta; m_a, \Sigma_a)$$

Can be very flexible with many components!



Auxiliary Variables & Mixture Distributions

- Construct $q(z|x)$ as a (hierarchical) mixture distribution

$$q(z|x) = \int q(z\,|a,x)\, q(a|x)\, da$$

  - a is the auxiliary variable used to enrich the approximate posterior
  - Now the variational lower-bound becomes intractable:

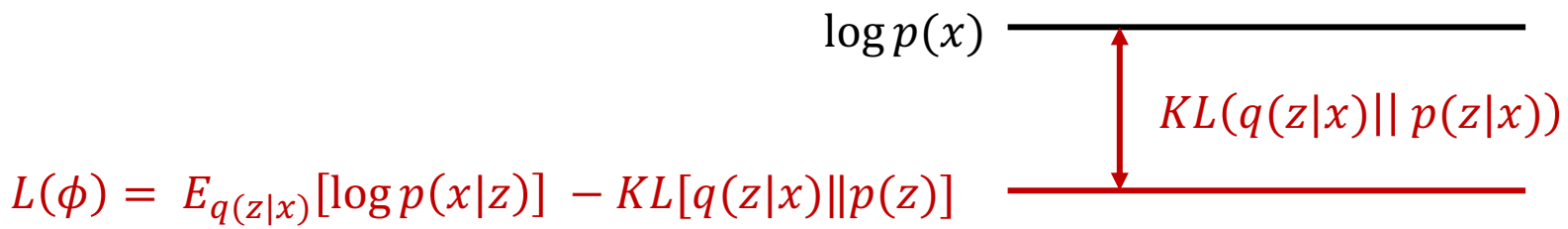
$$L(\phi) = E_{q(z|x)}[\log p(x,z)] - E_{q(z|x)}[\log q(z|x)]$$

Estimated by Monte Carlo:
$a^k \sim q(a|x), z^k \sim q(z \mid a^k, x)$

Intractable density
$q(z|x) = \int q(z|a,x)q(a|x)\, da$

Auxiliary Variables & Mixture Distributions

- Solution: introducing an auxiliary variational lower-bound $L(\phi, r)$ with an auxiliary distribution $r(a|z,x)$:



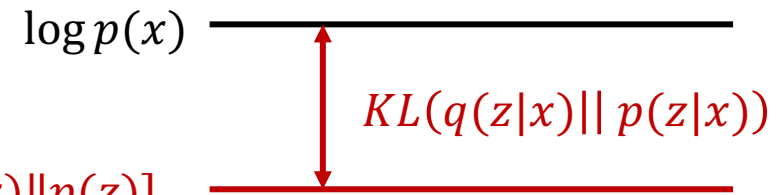
$$\log q(z|x) = \log \frac{q(z|a,x)q(a|x)}{q(a|z,x)} = \log \frac{q(z|a,x)q(a|x)}{r(a|z,x)} - \log \frac{q(a|z,x)}{r(a|z,x)}$$

(Bayes' rule) (insert in the auxiliary distribution)

Agakov and Barber. An Auxiliary Variational Method. ICONIP 2004
Salimans et al. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. ICML 2015
Ranganath et al. Hierarchical Variational Models. ICML 2016

Auxiliary Variables & Mixture Distributions

- Solution: introducing an auxiliary variational lower-bound $L(\phi, r)$ with an auxiliary distribution $r(a|z,x)$:



$$\log q(z|x) = \log \frac{q(z|a,x)q(a|x)}{q(a|z,x)} = \log \frac{q(z|a,x)q(a|x)}{r(a|z,x)} - \log \frac{q(a|z,x)}{r(a|z,x)}$$

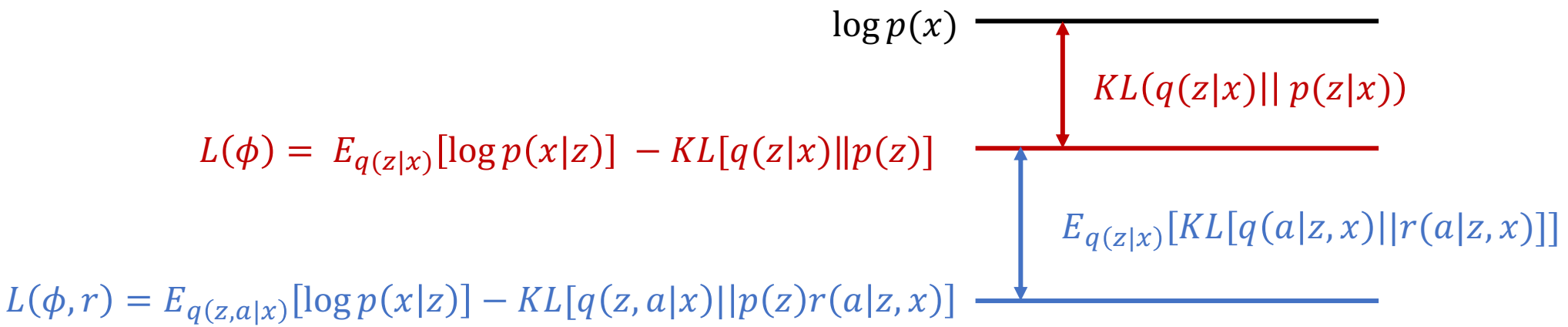
$$\Rightarrow KL[q(z|x)\|p(z)] = \underbrace{E_{q(z,a|x)}\left[\log \frac{q(z|a,x)q(a|x)}{p(z)r(a|z,x)}\right]}_{= KL[q(z,a|x)\|p(z)r(a|z,x)]} - \underbrace{E_{q(z,a|x)}[\log \frac{q(a|z,x)}{r(a|z,x)}]}_{= E_q[KL[q(a|z,x)\|r(a|z,x)]]}$$

(drop this last term)

Agakov and Barber. An Auxiliary Variational Method. ICONIP 2004
Salimans et al. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. ICML 2015
Ranganath et al. Hierarchical Variational Models. ICML 2016

Auxiliary Variables & Mixture Distributions

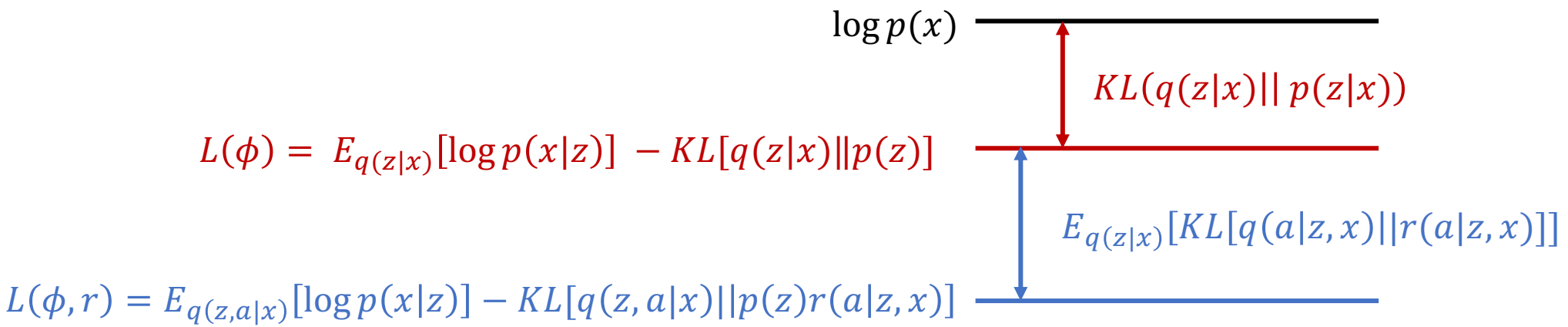
- Solution: introducing an auxiliary variational lower-bound $L(\phi, r)$ with an auxiliary distribution $r(a|z,x)$:



Agakov and Barber. An Auxiliary Variational Method. ICONIP 2004
Salimans et al. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. ICML 2015
Ranganath et al. Hierarchical Variational Models. ICML 2016

Auxiliary Variables & Mixture Distributions

- Solution: introducing an auxiliary variational lower-bound $L(\phi, r)$ with an auxiliary distribution $r(a|z,x)$:



- Optimize $r(a|z,x)$ to close the gap!
- $L(\phi, r)$ estimated by Monte Carlo:$a^k \sim q(a|x), z^k \sim q(z \mid a^k, x)$

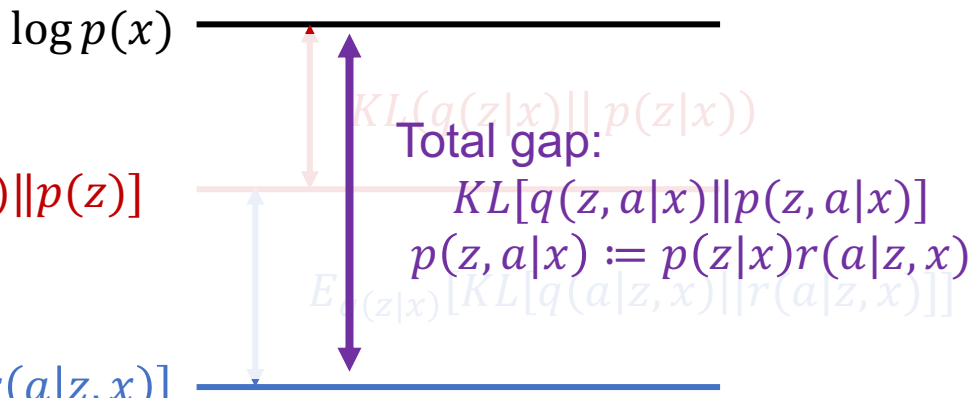
Agakov and Barber. An Auxiliary Variational Method. ICONIP 2004
Salimans et al. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. ICML 2015
Ranganath et al. Hierarchical Variational Models. ICML 2016

Auxiliary Variables & Mixture Distributions

- Solution: introducing an auxiliary variational lower-bound $L(\phi, r)$ with an auxiliary distribution $r(a|z,x)$:

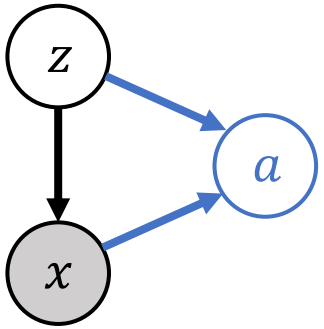
$$L(\phi) = E_{q(z|x)}[\log p(x|z)] - KL[q(z|x)\|p(z)]$$

$$L(\phi, r) = E_{q(z,a|x)}[\log p(x|z)] - KL[q(z,a|x)||p(z)r(a|z,x)]$$



Augmented generative model view:

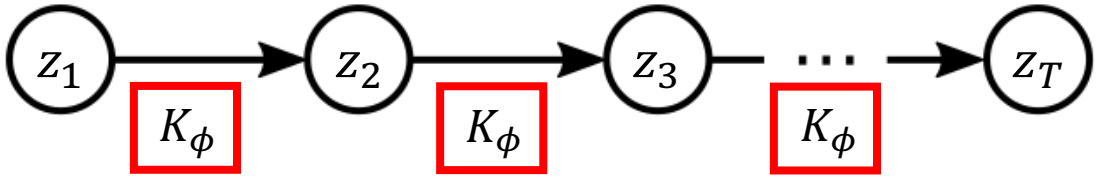
$$p(x,z,a) \coloneqq p(z)p(x|z)r(a|z,x) \quad \Rightarrow \quad p(z,a|x) = p(z|x)r(a|z,x)$$



Agakov and Barber. An Auxiliary Variational Method. ICONIP 2004
Salimans et al. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. ICML 2015
Ranganath et al. Hierarchical Variational Models. ICML 2016

Auxiliary Variables & Mixture Distributions

- Hierarchical mixture distributions for $q(z, a|x)$
  - VI-MCMC hybrid: build $q(z)$ with a Markov Chain:

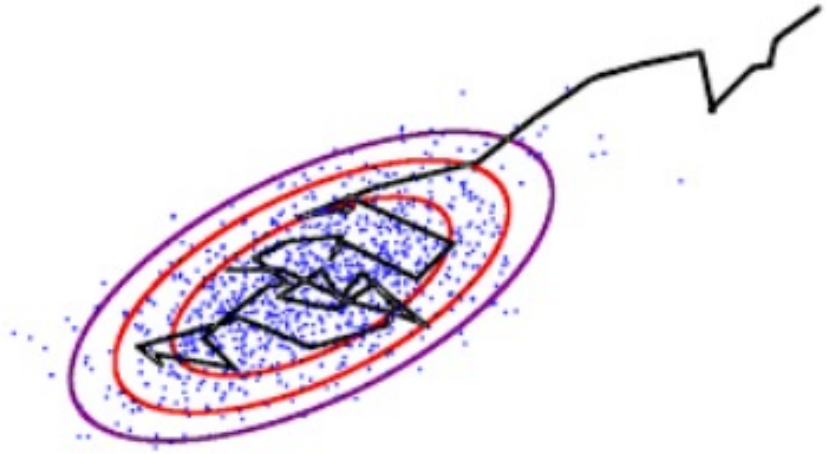


Learn the transition kernel with VI:

$$z \coloneqq z_T, a = \{z_{1:T-1}\}$$

$$q(z_{1:T}|x) = \prod_{t=1}^{T} K_\phi\,(z_t|z_{t-1}), \quad z_0 \coloneqq x$$

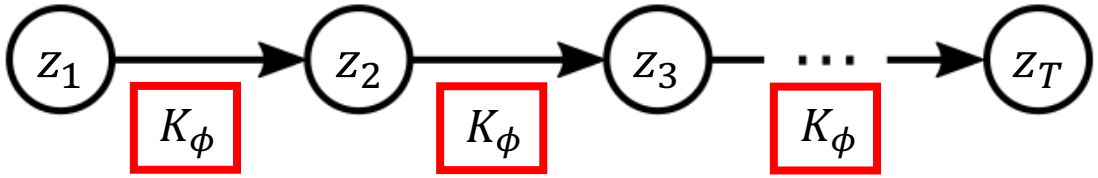
$$r(z_{1:T-1}|z_T) = \prod_{t=2}^{T} r\,(z_{t-1}\,|z_t)$$



Salimans et al. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. ICML 2015
Huang et al. Improving Explorability in Variational Inference with Annealed Variational Objectives. NeurIPS 2018

Auxiliary Variables & Mixture Distributions

- Hierarchical mixture distributions for $q(z, a|x)$
  - VI-MCMC hybrid: build $q(z)$ with a Markov Chain:

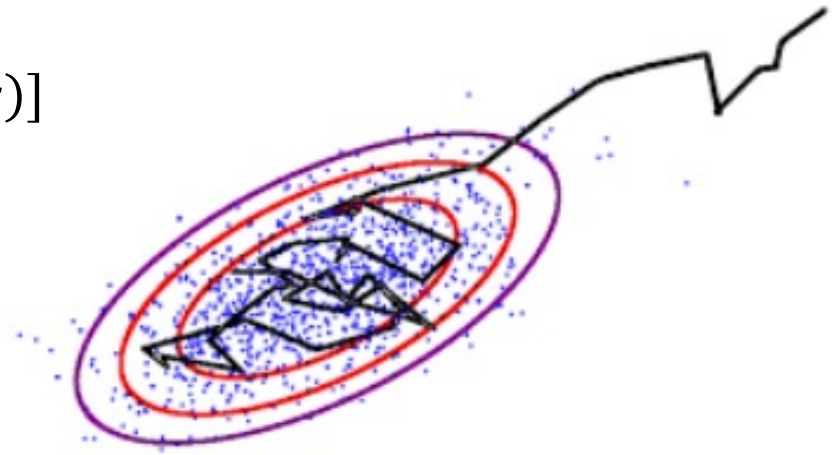


Learn the transition kernel with VI:

$$L(\phi, r) = E_{q(z,a|x)}[\log p(x|z)] - KL[q(z, a|x)||p(z)r(a|z, x)]$$

Equivalent to minimizing $KL[q(z, a|x)||p(z, a|x)]$,
$p(z, a|x) := p(z|x)r(a|z, x)$:

$$\log p(x) - L(\phi, r) = KL[q(z_{1:T}|z_0)||r(z_{1:T-1}|z_T)p(z_T|x)]$$



Salimans et al. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. ICML 2015
Huang et al. Improving Explorability in Variational Inference with Annealed Variational Objectives. NeurIPS 2018

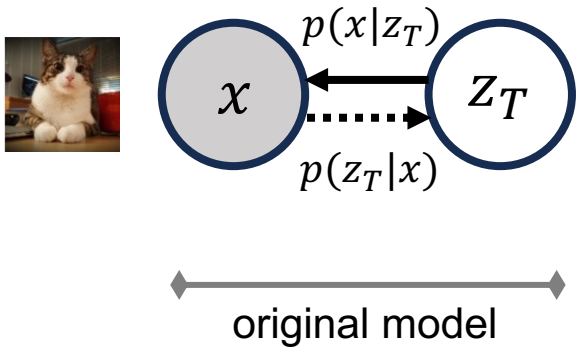
Auxiliary VI → Diffusion Posterior (Discrete Time)

- Consider the general case $(z_T \coloneqq z)$

$$\log p(x) - L(\phi, r) = KL[q(z_{1:T}|x) \| r(z_{1:T-1}|z_T) p(z_T|x)]$$

Original model: $p(x, z_T) = p(x|z_T)p(z_T)$

posterior: $p(z_T|x)$
(the target)



Auxiliary VI → Diffusion Posterior (Discrete Time)

- Consider the general case $(z_T \coloneqq z)$

$$\log p(x) - L(\phi, r) = KL[q(z_{1:T}|x) \| r(z_{1:T-1}|z_T) p(z_T|x)]$$

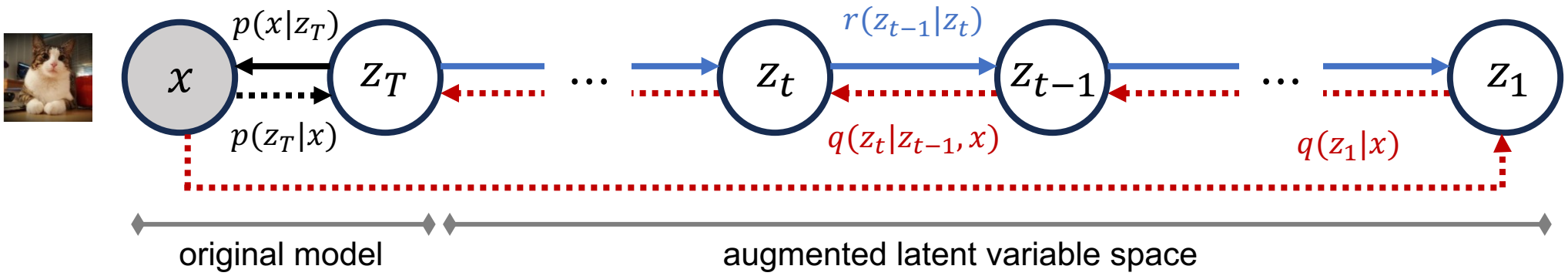
Original model: $p(x, z_T) = p(x|z_T)p(z_T)$

posterior: $p(z_T|x)$
(the target)

Augmented model: $p(x, z_T) r(z_{1:T-1}|z_T)$

$r(z_{1:T-1}|z_T) = \prod_{t=2}^{T} r(z_{t-1}|z_t)$

approx. posterior: $q(z_{1:T}|x) = q(z_1|x) \prod_{t=2}^{T} q(z_t|z_{t-1}, x)$



Auxiliary VI → Diffusion Posterior (Discrete Time)

- Consider the general case $(z_T \coloneqq z)$

$$\log p(x) - L(\phi, r) = KL[q(z_{1:T}|x) \| r(z_{1:T-1}|z_T) p(z_T|x)]$$

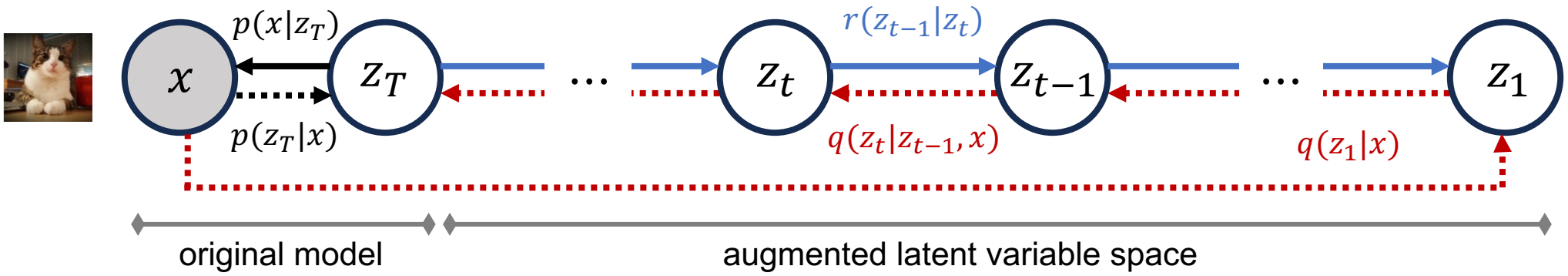
Original model: $p(x, z_T) = p(x|z_T)p(z_T)$

posterior: $p(z_T|x)$
(the target)

Gaussian (fixed)
Augmented model: $p(x, z_T) \prod_{t=2}^{T} r(z_{t-1}|z_t)$

$r(z_{t-1}|z_t) = N(z^{t-1}; \sqrt{1-\beta_t} z_t, \beta_t I)$

approx. posterior: $q(z_{1:T}|x) = q(z_1|x) \prod_{t=2}^{T} q(z_t|z_{t-1}, x)$



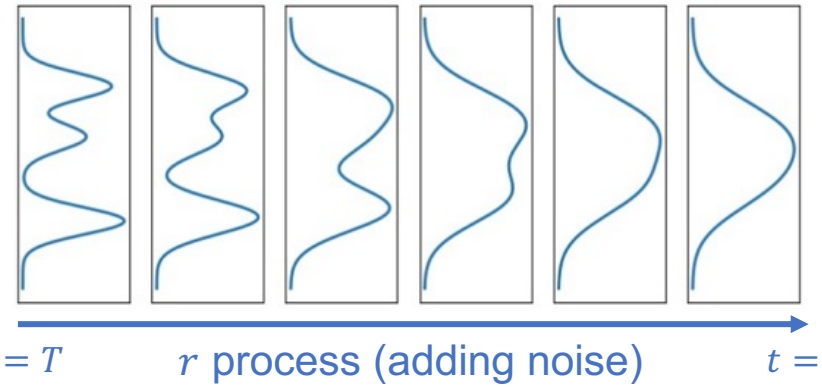
Auxiliary VI → Diffusion Posterior (Discrete Time)

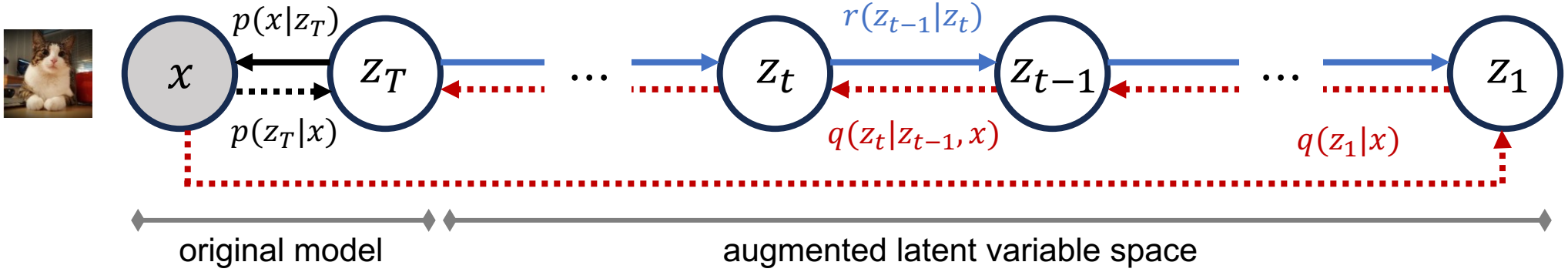
- Consider the general case $(z_T \coloneqq z)$

$$\log p(x) - L(\phi, r) = KL[q(z_{1:T}|x) \| r(z_{1:T-1}|z_T) p(z_T|x)]$$

Original model: $p(x, z_T) = p(x|z_T)p(z_T)$

posterior: $p(z_T|x)$
(the target)





Auxiliary VI → Diffusion Posterior (Discrete Time)

- Consider posterior approximation case:

Target distribution
(density eval up to constant)

Gaussian (fixed) Augmented model: $p(z_T|x)\prod_{t=2}^{T} r\,(z_{t-1}\,|z_t)$

$r(z_{t-1}|z_t) = N(z^{t-1}; \sqrt{1-\beta_t} z_t, \beta_t I)$

$(z_T \coloneqq z)$

approx. posterior: $q(z_{1:T}|x) = q(z_1|x)\prod_{t=2}^{T} q\,(z_t\,|z_{t-1}, x)$

Objective:

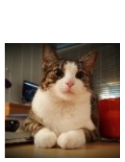
- VI-style:

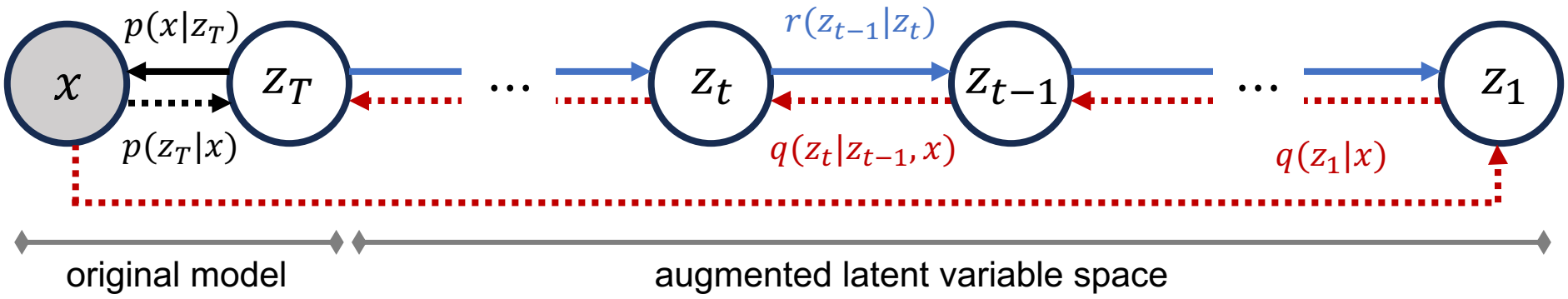
$$KL[q(z_{1:T}|x)\|r(z_{1:T-1}|z_T)p(z_T|x)]$$

learning to denoise the noisy posterior

adding noise to posterior

target (no samples)





Continuous-Time Limit: Diffusion SDE

- Re-define the time index: $t \to t/T$, $z_t \to z_{t/T}$, and take limit $T \to \infty$

Target distribution
(density eval up to constant)

Objective:

- VI-style:

$KL[q(z_{[0,1]}|x) \| r(z_{[0,1]})]$

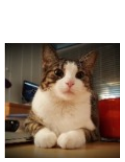
SDE
Augmented model: $r(z_{[0,1]})$ satisfying $r(z_1) = p(z_1|x)$,

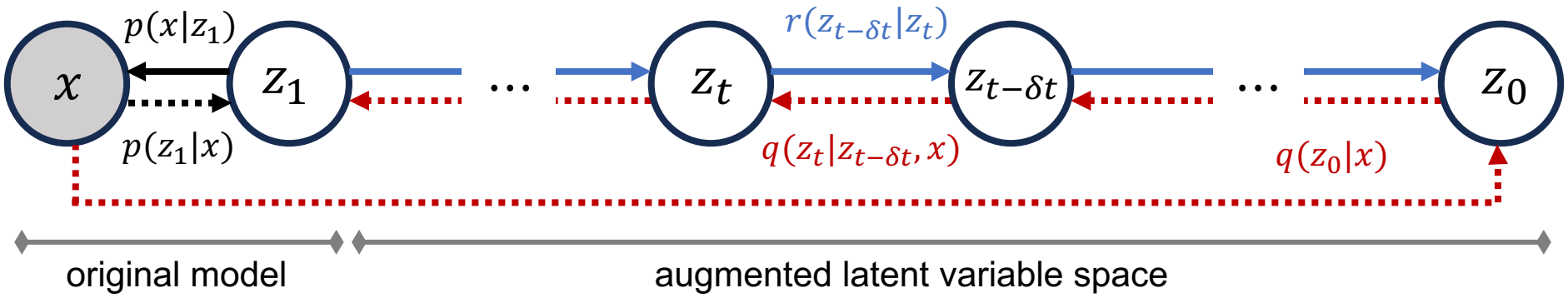
$dz_t = f_t(z_t)dt + \gamma_t dW_t$

$(z_1 := z)$

approx. posterior: $q(z_{[0,1]}|x)$: $q(z_0|x) = N(m, \Sigma)$,

$dz_t = \mu_t(z_t, x)dt + \sigma_t d\bar{W}_t$





He et al. No Trick, No Treat: Pursuits and Challenges Towards Simulation-free Training of Neural Samplers. arXiv:2502.06685

Vargas et al. Denoising Diffusion Samplers. ICLR 2023

Continuous-Time Limit: Diffusion SDE

- Re-define the time index: $t \to t/T$, $z_t \to z_{t/T}$, and take limit $T \to \infty$

Target distribution
(density eval up to constant)

Objective:

- VI-style:

$KL[q(z_{[0,1]}|x) \| r(z_{[0,1]})]$

SDE
Augmented model: $r(z_{[0,1]})$ satisfying $r(z_1) = p(z_1|x)$,

$dz_t = f_t(z_t)dt + \gamma_t dW_t$

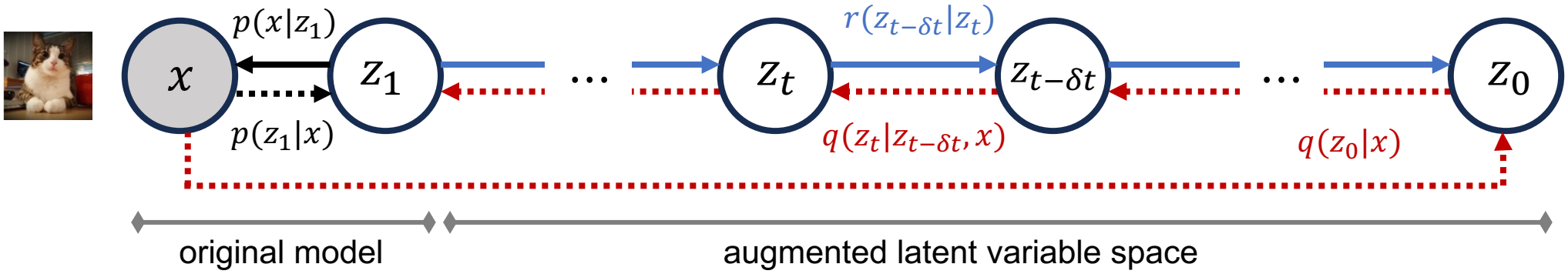
$(z_1 := z)$

Integrate from $t: 1 \to 0$
from target to noise ("forward")

approx. posterior: $q(z_{[0,1]}|x)$: $q(z_0|x) = N(m, \Sigma)$,

$dz_t = \mu_t(z_t, x)dt + \sigma_t d\bar{W}_t$

Integrate from $t: 0 \to 1$
from noise to target ("backward")



He et al. No Trick, No Treat: Pursuits and Challenges Towards Simulation-free Training of Neural Samplers. arXiv:2502.06685
Vargas et al. Denoising Diffusion Samplers. ICLR 2023

Continuous-Time Limit: Diffusion SDE

- Re-define the time index: $t \to t/T$, $z_t \to z_{t/T}$, and take limit $T \to \infty$

SDE
Augmented model: $r(z_{[0,1]})$ satisfying $r(z_1) = p(z_1|x)$,

$$dz_t = f_t(z_t)dt + \gamma_t dW_t,\ t: 1 \to 0$$

$$(z_1 \coloneqq z)$$

approx. posterior: $q(z_{[0,1]}|x)$: $q(z_0|x) = N(m, \Sigma)$,

$$dz_t = \mu_t(z_t, x)dt + \sigma_t d\bar{W}_t,\ t: 0 \to 1$$

Objective:

- VI-style:

$$KL[q(z_{[0,1]}|x) \| r(z_{[0,1]})]$$

- Conditions for a well-defined KL divergence between SDEs:
  - Going in the same direction for analytic form of the KL: reverse either the r SDE or the q SDE

He et al. No Trick, No Treat: Pursuits and Challenges Towards Simulation-free Training of Neural Samplers. arXiv:2502.06685
Vargas et al. Denoising Diffusion Samplers. ICLR 2023
Li et al. Scalable Gradients for Stochastic Differential Equations. AISTATS 2020
Song et al. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR 2021

Continuity Equation

- Forward SDE evolution of z_t, $t: 0 \to 1$

$$dz_t = \underbrace{f_t(z_t)}_{\text{(“forward drift”)}} dt + \sigma_t(z_t) dW_t, \quad z_0 \sim p_0(z_0)$$

- Fokker-Planck Equation: marginal density evolution of $p_t(z)$, $t: 0 \to 1$, for any z (not necessarily for $z = z_t$)

$$\partial_t\, p_t(z) = -\nabla_z \cdot \big(f_t(z) p_t(z)\big) + \frac{1}{2} \nabla_z^2 \cdot (\sigma_t^2(z) p_t(z))$$

- Reverse SDE evolution of z_t, $t: 1 \to 0$ that *preserves the density path*

$$dz_t = \underbrace{[f_t(z_t) - \sigma_t^2(z_t) \nabla_z \log p_t(z_t)]}_{\coloneqq \tilde{f}_t(z_t) \text{ (“backward drift”)}} dt + \sigma_t(z_t) d\bar{W}_t, \quad z_1 \sim p_1(z_1)$$

$$\Rightarrow \quad f_t(z_t) = \tilde{f}_t(z_t) + \sigma_t^2(z_t) \nabla_z \log p_t(z_t)$$

Anderson. Reverse-time diffusion equation models. *Stochastic Process. Appl.*, 12(3): 313–326, 1982

Continuous-Time Limit: Diffusion SDE

- Re-define the time index: $t \to t/T$, $z_t \to z_{t/T}$, and take limit $T \to \infty$

SDE Augmented model: $r(z_{[0,1]})$ satisfying $r(z_1) = p(z_1|x)$,

$$dz_t = f_t(z_t)dt + \gamma_t dW_t,\ t: 1 \to 0$$

$$(z_1 \coloneqq z)$$

approx. posterior: $q(z_{[0,1]}|x)$: $q(z_0|x) = N(m, \Sigma)$,

$$dz_t = \mu_t(z_t, x)dt + \sigma_t d\bar{W}_t,\ t: 0 \to 1$$

Objective:

- VI-style:

$$KL[q(z_{[0,1]}|x) \| r(z_{[0,1]})]$$

- Conditions for a well-defined KL divergence between SDEs:
  - Going in the same direction for analytic form of the KL: reverse either the r SDE or the q SDE

Reverse r SDE: $dz_t = [f_t(z_t) + \gamma_t^2 \nabla_z \log r_t(z_t)]dt + \gamma_t d\bar{W}_t,\ t: 0 \to 1$

He et al. No Trick, No Treat: Pursuits and Challenges Towards Simulation-free Training of Neural Samplers. arXiv:2502.06685
Vargas et al. Denoising Diffusion Samplers. ICLR 2023
Li et al. Scalable Gradients for Stochastic Differential Equations. AISTATS 2020
Song et al. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR 2021

Continuous-Time Limit: Diffusion SDE

- Re-define the time index: $t \to t/T$, $z_t \to z_{t/T}$, and take limit $T \to \infty$

SDE
Augmented model: $r(z_{[0,1]})$ satisfying $r(z_1) = p(z_1|x)$,

$$dz_t = f_t(z_t)dt + \gamma_t dW_t,\ t: 1 \to 0$$

$$(z_1 \coloneqq z)$$

approx. posterior: $q(z_{[0,1]}|x)$: $q(z_0|x) = N(m, \Sigma)$,

$$dz_t = \mu_t(z_t, x)dt + \sigma_t d\bar{W}_t,\ t: 0 \to 1$$

Objective:

- VI-style:

$$KL[q(z_{[0,1]}|x) \| r(z_{[0,1]})]$$

- Conditions for a well-defined KL divergence between SDEs:
  - Going in the same direction for analytic form of the KL: reverse either the r SDE or the q SDE

Reverse r SDE: $dz_t = [f_t(z_t) + \gamma_t^2 \nabla_z \log r_t(z_t)]dt + \gamma_t d\bar{W}_t,\ t: 0 \to 1$

  - Same amount of diffusion: $\sigma_t = \gamma_t$

$$KL[q(z_{[0,1]}|x) \| r(z_{[0,1]})] = KL[q(z_0|x) \| r(z_0)] + \frac{1}{2} E_q\left[\int_0^1 \frac{1}{\sigma_t^2} \| f_t(z_t) + \gamma_t^2 \nabla_z \log r_t(z_t) - \mu_t(z_t, x) \|_2^2 \, dt\right]$$

He et al. No Trick, No Treat: Pursuits and Challenges Towards Simulation-free Training of Neural Samplers. arXiv:2502.06685
Vargas et al. Denoising Diffusion Samplers. ICLR 2023
Li et al. Scalable Gradients for Stochastic Differential Equations. AISTATS 2020
Song et al. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR 2021

Continuous-Time Limit: Diffusion SDE

- Re-define the time index: $t \to t/T$, $z_t \to z_{t/T}$, and take limit $T \to \infty$

Diffusion SDE $r(z_{[0,1]})$ satisfying $r(z_1) = p(z_1|x)$,

Augmented model: $dz_t = \beta_t z_t dt + \sigma\sqrt{2\beta_t} dW_t,\ t: 1 \to 0$

$(z_1 \coloneqq z)$

Objective:

- VI-style:

$KL[q(z_{[0,1]}|x) \| r(z_{[0,1]})]$

approx. posterior: $q(z_{[0,1]}|x)$: $q(z_0|x) = N(m, \Sigma)$,

$dz_t = \mu_t(z_t, x)dt + \sigma\sqrt{2\beta_t} d\bar{W}_t,\ t: 0 \to 1$

- Conditions for a well-defined KL divergence between SDEs:
  - Going in the same direction for analytic form of the KL: reverse either the r SDE or the q SDE

Reverse r SDE: $dz_t = [\beta_t z_t + 2\sigma^2 \beta_t \nabla_z \log r_t(z_t)]dt + \sigma\sqrt{2\beta_t} d\bar{W}_t,\ t: 0 \to 1$

  - Same amount of diffusion: $\sigma_t = \gamma_t = \sigma\sqrt{2\beta_t}$

$$KL[q(z_{[0,1]}|x) \| r(z_{[0,1]})] = KL[q(z_0|x) \| r(z_0)] + \frac{1}{2\sigma^2} E_q\left[\int_0^1 \frac{1}{2\beta_t} \| \beta_t z_t + 2\sigma^2 \beta_t \nabla_z \log r_t(z_t) - \mu_t(z_t, x) \|_2^2 \, dt\right]$$

He et al. No Trick, No Treat: Pursuits and Challenges Towards Simulation-free Training of Neural Samplers. arXiv:2502.06685

Vargas et al. Denoising Diffusion Samplers. ICLR 2023

Li et al. Scalable Gradients for Stochastic Differential Equations. AISTATS 2020

Song et al. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR 2021

Continuous-Time Limit: Diffusion SDE

- Re-define the time index: $t \to t/T$, $z_t \to z_{t/T}$, and take limit $T \to \infty$

Diffusion SDE $r(z_{[0,1]})$ satisfying $r(z_1) = p(z_1|x)$,

Augmented model: $dz_t = \beta_t z_t dt + \sigma\sqrt{2\beta_t} dW_t,\ t: 1 \to 0$

$(z_1 \coloneqq z)$

Objective:

- VI-style:

$KL[q(z_{[0,1]}|x) \| r(z_{[0,1]})]$

Score param. approx. posterior: $q(z_{[0,1]}|x)$: $q(z_0|x) = N(m, \Sigma)$,

$dz_t = [\beta_t z_t + 2\sigma^2 \beta_t s_\phi(z_t, x, t)]dt + \sigma\sqrt{2\beta_t} d\bar{W}_t,\ t: 0 \to 1$

- Conditions for a well-defined KL divergence between SDEs:
  - Going in the same direction for analytic form of the KL: reverse either the r SDE or the q SDE

Reverse r SDE: $dz_t = [\beta_t z_t + 2\sigma^2 \beta_t \nabla_z \log r_t(z_t)]dt + \sigma\sqrt{2\beta_t} d\bar{W}_t,\ t: 0 \to 1$

  - Same amount of diffusion: $\sigma_t = \gamma_t = \sigma\sqrt{2\beta_t}$

$$KL[q(z_{[0,1]}|x) \| r(z_{[0,1]})] = KL[q(z_0|x) \| r(z_0)] + \sigma^2 E_q\left[\int_0^1 \beta_t \| \nabla_z \log r_t(z_t) - s_\phi(z_t, x, t) \|_2^2 \, dt\right]$$

He et al. No Trick, No Treat: Pursuits and Challenges Towards Simulation-free Training of Neural Samplers. arXiv:2502.06685

Vargas et al. Denoising Diffusion Samplers. ICLR 2023

Li et al. Scalable Gradients for Stochastic Differential Equations. AISTATS 2020

Song et al. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR 2021

Estimating the Score

(we can sample $z_{[0,1]} \sim q$)

$$KL[q(z_{[0,1]}|x)\|r(z_{[0,1]})] = KL[q(z_0|x)\|r(z_0)] + \sigma^2 E_q[\int_0^1 \beta_t \|\nabla_z \log r_t(z_t) - s_\phi(z_t, x, t)\|_2^2 \, dt]$$

Diffusion SDE
Augmented model: $r(z_{[0,1]})$ satisfying $r(z_1) = p(z_1|x)$,

$$dz_t = \beta_t z_t dt + \sigma\sqrt{2\beta_t} dW_t, \; t: 1 \to 0$$

$$:= r_t(z_t|z_1)$$

Time marginal:
(ignore x notation)

$$r_t(z_t) = \int N(z_t|\sqrt{1-\lambda_t} z_1, \sigma^2 \lambda_t I) r(z_1) dz_1, \quad \lambda_t = 1 - \exp[-2\int_t^1 \beta_\tau d\tau]$$

He et al. No Trick, No Treat: Pursuits and Challenges Towards Simulation-free Training of Neural Samplers. arXiv:2502.06685
Vargas et al. Denoising Diffusion Samplers. ICLR 2023

Estimating the Score

(we can sample $z_{[0,1]} \sim q$)

$$KL[q(z_{[0,1]}|x)\|r(z_{[0,1]})] = KL[q(z_0|x)\|r(z_0)] + \sigma^2 E_q[\int_0^1 \beta_t \|\nabla_z \log r_t(z_t) - s_\phi(z_t, x, t)\|_2^2 \, dt]$$

Diffusion SDE $r(z_{[0,1]})$ satisfying $r(z_1) = p(z_1|x)$,

Augmented model: $dz_t = \beta_t z_t dt + \sigma\sqrt{2\beta_t} dW_t, \; t: 1 \to 0$

$\coloneqq r_t(z_t|z_1)$

Time marginal: $r_t(z_t) = \int N(z_t|\sqrt{1-\lambda_t} z_1, \sigma^2 \lambda_t I) r(z_1) dz_1, \quad \lambda_t = 1 - \exp[-2\int_t^1 \beta_\tau d\tau]$

(ignore x notation)

Denoising Score Identity:

(log derivative trick + Bayes' Rule)

$$\nabla_{z_t} \log r_t(z_t) = \frac{\int \nabla_{z_t} r_t(z_t|z_1) r(z_1) dz_1}{r_t(z_t)} = \int \frac{r_t(z_t|z_1) r(z_1)}{r_t(z_t)} \nabla_{z_t} \log r_t(z_t|z_1) \, dz_1 = E_{r_t(z_1|z_t)}[\nabla_{z_t} \log r_t(z_t|z_1)]$$

He et al. No Trick, No Treat: Pursuits and Challenges Towards Simulation-free Training of Neural Samplers. arXiv:2502.06685
Vargas et al. Denoising Diffusion Samplers. ICLR 2023

Estimating the Score

(we can sample $z_{[0,1]} \sim q$)

$$KL[q(z_{[0,1]}|x)\|r(z_{[0,1]})] = KL[q(z_0|x)\|r(z_0)] + \sigma^2 E_q[\int_0^1 \beta_t \|\nabla_z \log r_t(z_t) - s_\phi(z_t, x, t)\|_2^2 \, dt]$$

Diffusion SDE Augmented model: $r(z_{[0,1]})$ satisfying $r(z_1) = p(z_1|x)$,

$$dz_t = \beta_t z_t dt + \sigma\sqrt{2\beta_t} dW_t, \; t: 1 \to 0$$

$$:= r_t(z_t|z_1)$$

Time marginal: (ignore x notation)

$$r_t(z_t) = \int N(z_t|\sqrt{1-\lambda_t} z_1, \sigma^2 \lambda_t I) r(z_1) dz_1, \quad \lambda_t = 1 - \exp[-2\int_t^1 \beta_\tau d\tau]$$

Denoising Score Identity:

(log derivative trick + Bayes' Rule)

$$\nabla_{z_t} \log r_t(z_t) = \frac{\int \nabla_{z_t} r_t(z_t|z_1) r(z_1) dz_1}{r_t(z_t)} = \int \frac{r_t(z_t|z_1) r(z_1)}{r_t(z_t)} \nabla_{z_t} \log r_t(z_t|z_1) \, dz_1 = E_{r_t(z_1|z_t)}[\nabla_{z_t} \log r_t(z_t|z_1)]$$

Target Score Identity:

$\nabla_{z_t} \log r_t(z_t|z_1) = -\nabla_{z_1} \log r_t(z_t|z_1)$ (Using properties of $r(z_t|z_1)$ as Gaussian)

$$= -\nabla_{z_1} \log \frac{r_t(z_1|z_t) r_t(z_t)}{r_1(z_1)} = \nabla_{z_1} \log r_1(z_1) - \nabla_{z_1} \log r_t(z_1|z_t)$$

He et al. No Trick, No Treat: Pursuits and Challenges Towards Simulation-free Training of Neural Samplers. arXiv:2502.06685
Vargas et al. Denoising Diffusion Samplers. ICLR 2023
De Bortoli et al. Target Score Matching. arXiv:2402.08667
Akhound-Sadegh et al. Iterated Denoising Energy Matching for Sampling from Boltzmann Densities. ICML 2024

Estimating the Score

(we can sample $z_{[0,1]} \sim q$)

$$KL[q(z_{[0,1]}|x)\|r(z_{[0,1]})] = KL[q(z_0|x)\|r(z_0)] + \sigma^2 E_q[\int_0^1 \beta_t \|\nabla_z \log r_t(z_t) - s_\phi(z_t, x, t)\|_2^2 \, dt]$$

Diffusion SDE Augmented model: $r(z_{[0,1]})$ satisfying $r(z_1) = p(z_1|x)$,

$$dz_t = \beta_t z_t dt + \sigma\sqrt{2\beta_t} dW_t, \; t: 1 \to 0$$

$$:= r_t(z_t|z_1)$$

Time marginal: (ignore x notation)

$$r_t(z_t) = \int N(z_t|\sqrt{1-\lambda_t} z_1, \sigma^2 \lambda_t I) r(z_1) dz_1, \quad \lambda_t = 1 - \exp[-2\int_t^1 \beta_\tau d\tau]$$

Denoising Score Identity:

(log derivative trick + Bayes' Rule)

$$\nabla_{z_t} \log r_t(z_t) = \frac{\int \nabla_{z_t} r_t(z_t|z_1) r(z_1) dz_1}{r_t(z_t)} = \int \frac{r_t(z_t|z_1) r(z_1)}{r_t(z_t)} \nabla_{z_t} \log r_t(z_t|z_1) \, dz_1 = E_{r_t(z_1|z_t)}[\nabla_{z_t} \log r_t(z_t|z_1)]$$

Target Score Identity:

$$\nabla_{z_t} \log r_t(z_t) = E_{r_t(z_1|z_t,x)}[\nabla_{z_1} \log r_1(z_1) - \nabla_{z_1} \log r_t(z_1|z_t)] = E_{r_t(z_1|z_t,x)}[\nabla_{z_1} \log r_1(z_1)]$$

He et al. No Trick, No Treat: Pursuits and Challenges Towards Simulation-free Training of Neural Samplers. arXiv:2502.06685
Vargas et al. Denoising Diffusion Samplers. ICLR 2023
De Bortoli et al. Target Score Matching. arXiv:2402.08667
Akhound-Sadegh et al. Iterated Denoising Energy Matching for Sampling from Boltzmann Densities. ICML 2024

Estimating the Score

(we can sample $z_{[0,1]} \sim q$)

$$KL[q(z_{[0,1]}|x)\|r(z_{[0,1]})] = KL[q(z_0|x)\|r(z_0)] + \sigma^2 E_q[\int_0^1 \beta_t \|\nabla_z \log r_t(z_t) - s_\phi(z_t, x, t)\|_2^2 \, dt]$$

Diffusion SDE $r(z_{[0,1]})$ satisfying $r(z_1) = p(z_1|x)$,

Augmented model: $dz_t = \beta_t z_t dt + \sigma\sqrt{2\beta_t} dW_t, \; t: 1 \to 0$

$\coloneqq r_t(z_t|z_1)$

Time marginal: $r_t(z_t) = \int N(z_t|\sqrt{1-\lambda_t} z_1, \sigma^2 \lambda_t I) r(z_1) dz_1, \quad \lambda_t = 1 - \exp[-2\int_t^1 \beta_\tau d\tau]$

(ignore x notation)

$$\nabla_{z_t} \log r_t(z_t) = E_{r_t(z_1|z_t)}[\nabla_{z_t} \log r_t(z_t|z_1)] = E_{r_t(z_1|z_t)}\left[\nabla_{z_1} \log r_1(z_1)\right]$$

Denoising Score Identity — Target Score Identity

He et al. No Trick, No Treat: Pursuits and Challenges Towards Simulation-free Training of Neural Samplers. arXiv:2502.06685
Vargas et al. Denoising Diffusion Samplers. ICLR 2023
De Bortoli et al. Target Score Matching. arXiv:2402.08667
Akhound-Sadegh et al. Iterated Denoising Energy Matching for Sampling from Boltzmann Densities. ICML 2024

Estimating the Score

(we can sample $z_{[0,1]} \sim q$)

$$KL[q(z_{[0,1]}|x)\|r(z_{[0,1]})] = KL[q(z_0|x)\|r(z_0)] + \sigma^2 E_q[\int_0^1 \beta_t \|\nabla_z \log r_t(z_t) - s_\phi(z_t, x, t)\|_2^2 \, dt]$$

Diffusion SDE
Augmented model: $r(z_{[0,1]})$ satisfying $r(z_1) = p(z_1|x)$,

$$dz_t = \beta_t z_t dt + \sigma\sqrt{2\beta_t} dW_t, \; t: 1 \to 0$$

$$:= r_t(z_t|z_1)$$

Time marginal:
(ignore x notation)

$$r_t(z_t) = \int N(z_t|\sqrt{1-\lambda_t} z_1, \sigma^2 \lambda_t I) r(z_1) dz_1, \quad \lambda_t = 1 - \exp[-2\int_t^1 \beta_\tau d\tau]$$

$$\nabla_{z_t} \log r_t(z_t) = E_{r_t(z_1|z_t)}[\nabla_{z_t} \log r_t(z_t|z_1)] = E_{r_t(z_1|z_t)}[\nabla_{z_1} \log r_1(z_1)]$$

Denoising Score Identity — Target Score Identity

Sampling $z_1 \sim r_t(z_1|z_t)$ via importance sampling:

$$:= r(z_t|z_1)$$

$$r_t(z_1|z_t) \propto r_1(z_1) \exp\left[-\frac{1}{2\sigma^2\lambda_t}\|z_t - \sqrt{1-\lambda_t} z_1\|_2^2\right] \propto r_1(z_1) \exp\left[-\frac{1-\lambda_t}{2\sigma^2\lambda_t}\left\|z_1 - \frac{z_t}{\sqrt{1-\lambda_t}}\right\|_2^2\right]$$

$$\propto N\left(z_1; \frac{z_t}{\sqrt{1-\lambda_t}}, \frac{\sigma^2\lambda_t}{1-\lambda_t} I\right) =: n_t(z_1|z_t)$$

He et al. No Trick, No Treat: Pursuits and Challenges Towards Simulation-free Training of Neural Samplers. arXiv:2502.06685
Vargas et al. Denoising Diffusion Samplers. ICLR 2023
De Bortoli et al. Target Score Matching. arXiv:2402.08667
Akhound-Sadegh et al. Iterated Denoising Energy Matching for Sampling from Boltzmann Densities. ICML 2024

Estimating the Score

(we can sample $z_{[0,1]} \sim q$)

$$KL[q(z_{[0,1]}|x)\|r(z_{[0,1]})] = KL[q(z_0|x)\|r(z_0)] + \sigma^2 E_q[\int_0^1 \beta_t \|\nabla_z \log r_t(z_t) - s_\phi(z_t, x, t)\|_2^2 \, dt]$$

Diffusion SDE Augmented model: $r(z_{[0,1]})$ satisfying $r(z_1) = p(z_1|x)$,

$$dz_t = \beta_t z_t dt + \sigma\sqrt{2\beta_t} dW_t, \; t: 1 \to 0$$

$$:= r_t(z_t|z_1)$$

Time marginal: (ignore x notation)

$$r_t(z_t) = \int N(z_t|\sqrt{1-\lambda_t} z_1, \sigma^2 \lambda_t I) r(z_1) dz_1, \quad \lambda_t = 1 - \exp[-2\int_t^1 \beta_\tau d\tau]$$

$$\nabla_{z_t} \log r_t(z_t) = E_{r_t(z_1|z_t)}[\nabla_{z_t} \log r_t(z_t|z_1)] = E_{r_t(z_1|z_t)}[\nabla_{z_1} \log r_1(z_1)]$$

Denoising Score Identity — Target Score Identity

Sampling $z_1 \sim r_t(z_1|z_t)$ via importance sampling:

$$N\left(z_1; \frac{z_t}{\sqrt{1-\lambda_t}}, \frac{\sigma^2 \lambda_t}{1-\lambda_t} I\right) =: n_t(z_1|z_t)$$

$$\nabla_{z_t} \log r_t(z_t) = E_{n_t(z_1|z_t)}[w(z_1) \nabla_{z_t} \log r_t(z_t|z_1)] = E_{n_t(z_1|z_t)}[w(z_1) \nabla_{z_1} \log r_1(z_1)]$$

$w(z_1) \propto \tilde{r}_1(z_1)$ (unnormalized target density)

He et al. No Trick, No Treat: Pursuits and Challenges Towards Simulation-free Training of Neural Samplers. arXiv:2502.06685
Vargas et al. Denoising Diffusion Samplers. ICLR 2023
De Bortoli et al. Target Score Matching. arXiv:2402.08667
Akhound-Sadegh et al. Iterated Denoising Energy Matching for Sampling from Boltzmann Densities. ICML 2024

Auxiliary VI → Diffusion Posterior (Continuous Time)

- A summary of the Diffusion algorithm applied to VI

Diffusion SDE Augmented model: $r(z_{[0,1]})$ satisfying $r(z_1) = p(z_1|x)$ (target density)

$$dz_t = \beta_t z_t dt + \sigma\sqrt{2\beta_t} dW_t,\ t: 1 \to 0$$

$$(z_1 \coloneqq z)$$

Score param. approx. posterior: $q(z_{[0,1]}|x)$: $q(z_0|x) = N(m, \Sigma)$,

$$dz_t = [\beta_t z_t + 2\sigma^2 \beta_t s_\phi(z_t, t)]dt + \sigma\sqrt{2\beta_t} d\bar{W}_t,\ t: 0 \to 1$$

- Objective: VI style, $KL[q(z_{[0,1]}|x) \| r(z_{[0,1]})]$:

  Fit $q(z_0|x)$, $s_\phi(z_t, x, t)$ by min. $KL[q(z_0|x) \| r(z_0)]$ plus

$$E_q[\int_0^1 \beta_t \| \nabla_z \log r_t(z_t) - s_\phi(z_t, x, t) \|_2^2 \, dt]$$

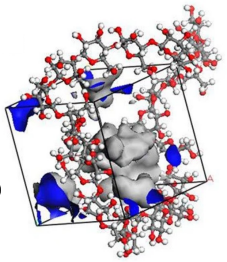
  (estimated via importance sampling + denoising/target score identity)

- After fitting the posterior (i.e., $s_\phi(z_t, t)$):

  Sample $z_0 \sim q(z_0|x)$,
  and simulate the approx. posterior SDE
  to get approximately $z_1 \sim p(z_1|x)$

He et al. No Trick, No Treat: Pursuits and Challenges Towards Simulation-free Training of Neural Samplers. arXiv:2502.06685
Vargas et al. Denoising Diffusion Samplers. ICLR 2023
Akhound-Sadegh et al. Iterated Denoising Energy Matching for Sampling from Boltzmann Densities. ICML 2024

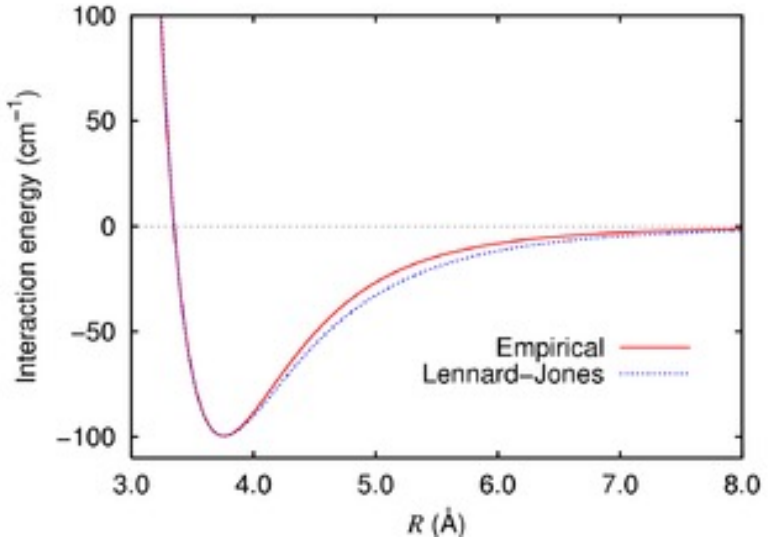
Example: Molecular Dynamics Simulations



Benchmark: Lennard-Jones Potential (LJ-N)

- Goal: study intermolecular interactions
- A molecule is a system of N atoms ("particles") $z = [z_1, \dots, z_N]$
- Each particle z_i has its own 3D coordinates (so $\dim(z) = 3N$)

$$z_i = [z_i^x, z_i^y, z_i^z]$$



- Define $r_{ij} = \|z_i - z_j\|_2^2$, the Lennard-Jones potential is:

$$V_{LJ}(r_{ij}, n, m) := \frac{A_n}{r_{ij}^n} - \frac{B_m}{r_{ij}^m}$$

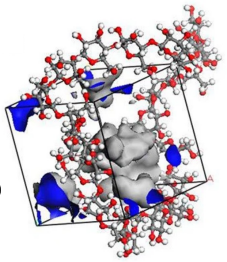
- Widely used example: Lennard-Jones 12-6 potential $(n = 12, A_n = 4\epsilon\sigma^{12}, m = 6, B_m = 4\epsilon\sigma^6)$

$$V_{LJ}(r_{ij}) := 4\epsilon\left[\frac{\sigma^{12}}{r_{ij}^{12}} - \frac{\sigma^6}{r_{ij}^6}\right]$$

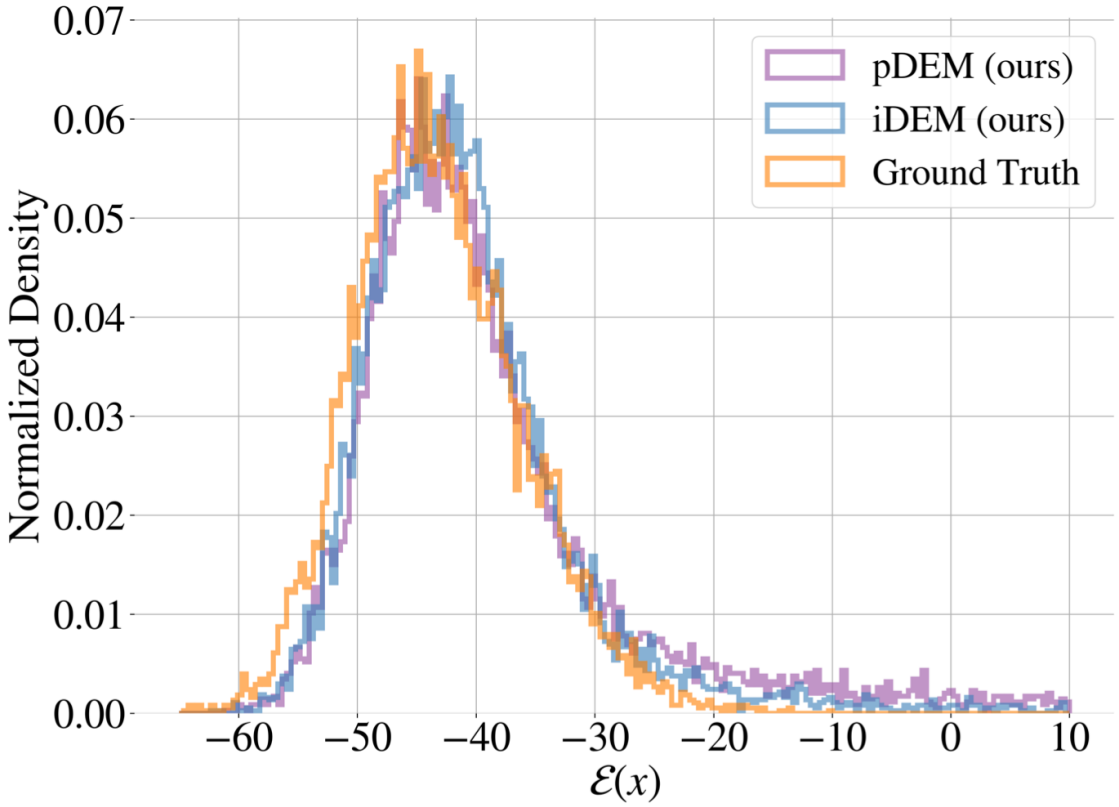
$(\min_r V_{LJ}(r) = -\epsilon)$

Target distribution: $\pi(z) \propto \exp[-\frac{E(z)}{k_B T}]$, $E(z) := \sum_{i,j\neq i} V_{LJ}(r_{ij})$

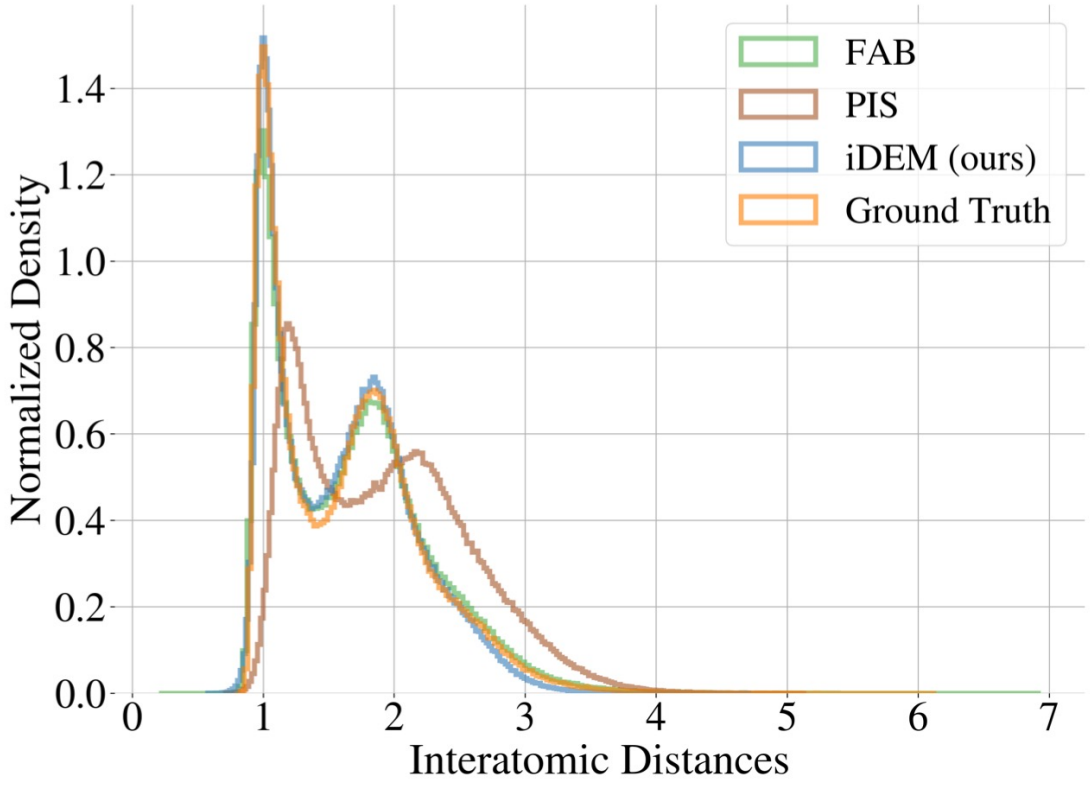
Example: Molecular Dynamics Simulations



LJ13: Lennard-Jones 12-6 potential, $N = 13$ particles ($\dim(z) = 39$)

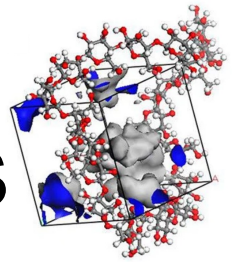


Histogram of energy values at samples



Histogram of r_{ij} at samples

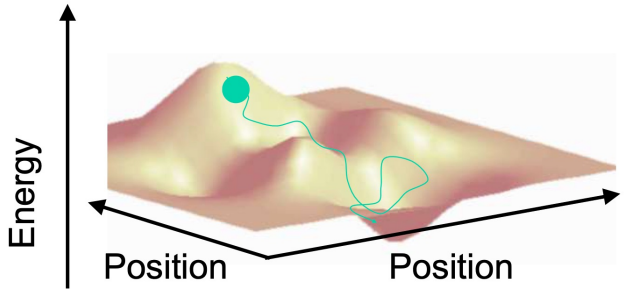
Example: Molecular Dynamics Simulations



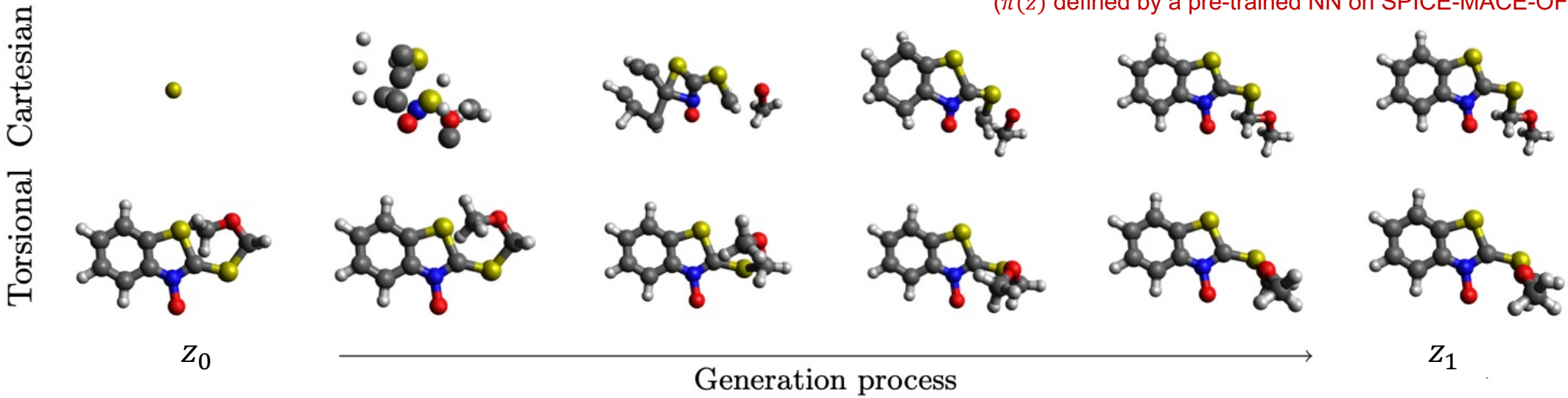
Sampling conformations based on energy functions in (quantum) density functional theory (DFT)

Conformers of a molecule:

- spatial arrangements determined by locally stable configurations of rotations around its bonds
  - i.e., local minima on the molecule's potential energy surface
- can be interconverted by rotations around single bonds



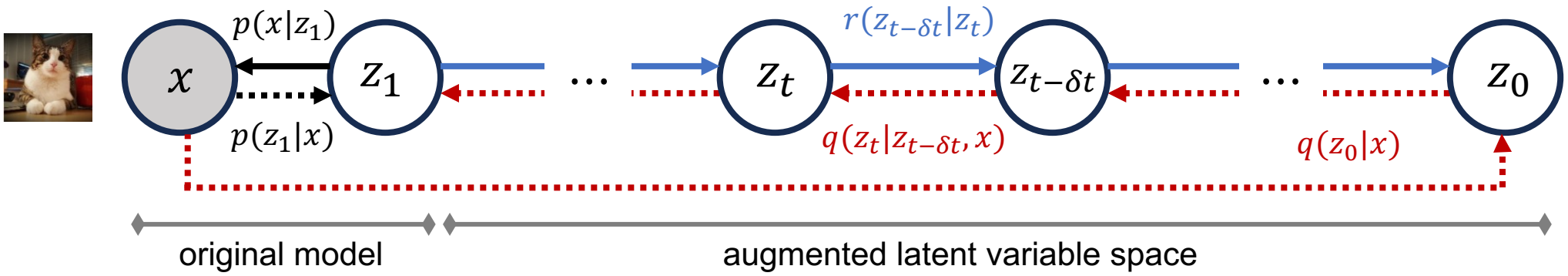
($\pi(z)$ defined by a pre-trained NN on SPICE-MACE-OFF dataset)



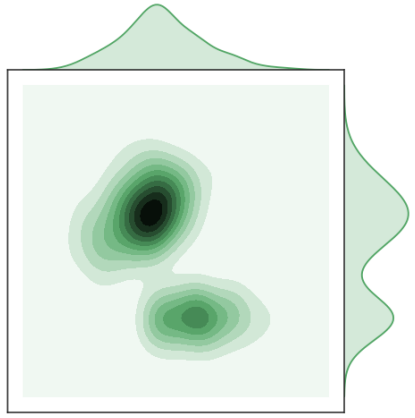
Summary (Diffusion-based Approximations)

Constructing flexible q distribution via mixtures:

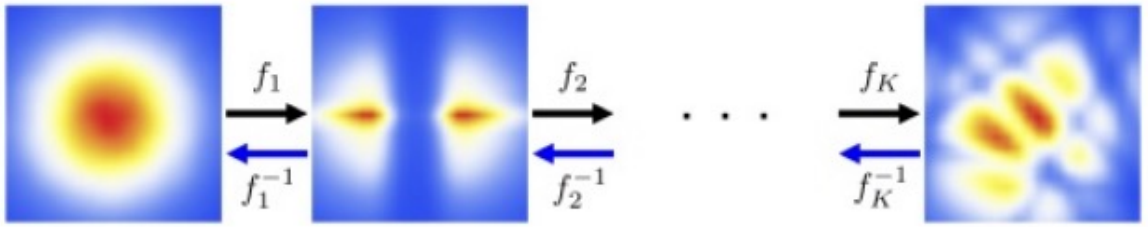
- Hierarchical mixtures (very flexible)
- The VI objective needs to be augmented (adding in the r distributions)
- Add many layers and take continuous-limit: Diffusion-based approximations
  - Be careful about the validity of the KL divergence definition
  - Similar to diffusion generative model training except for the score estimation method



Designing q Distributions



Auxiliary variables & mixture distributions



Normalizing flows

Use many layers (hierarchical) + continuous-time limit

Diffusion SDE & Continuous Normalizing Flow posteriors for approximate inference

Normalizing Flows (Discrete Time)

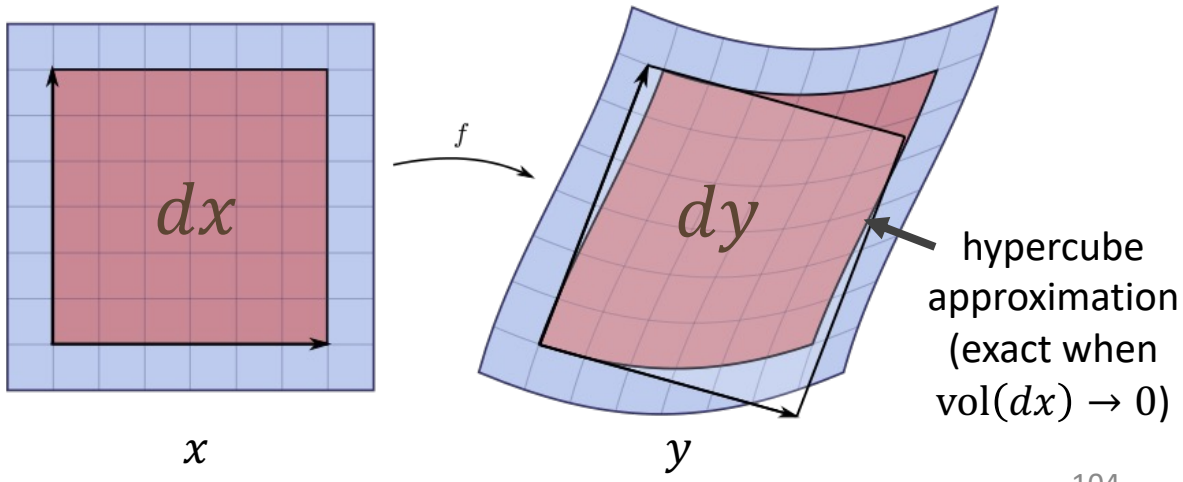
- Change-of-variable formula:
  - x is a random variable with probability density function (PDF) $p_X(x)$
  - $\mathrm{y} = f(x)$ is an invertible mapping
  - The probability mass is preserved, and the PDF for $y = f(x)$ satisfies

$$p_Y(y)dy = p_X(x)dx$$

prob. mass of region around y prob. mass of region around x

$$p_Y(y) = p_X(x)|\det(\frac{dx}{dy})|$$

$$p_X(x) = p_Y(y)|\det(\frac{dy}{dx})|$$



Normalizing Flows (Discrete Time)

- Variational inference with normalizing flow
  - Assume $q_0(z_0) = N(z_0; 0, I)$
  - Define $z = T_\phi(z_0)$ where $T_\phi(\cdot)$ is an invertible mapping parameterized by ϕ

$$q(z) = q_0(z_0)\left|\det\left(\frac{dz}{dz_0}\right)\right|^{-1} \qquad \text{with} \quad z_0 = T_\phi^{-1}(z)$$

  - Fit $q(z)$ to $p(x \mid z)$ with VI:

$$\begin{aligned} L\big(q(z)\big) &= E_{q(z)}[\log p(x \mid z) + \log p(z) - \log q(z)] \qquad \text{by def. of } q(z) \\ &= E_{q(z)}\left[\log p(x,\ z) - \log q_0(z_0 = T_\phi^{-1}(z))\left|\det\left(\frac{dz}{dz_0}\right)\right|^{-1}\right] \\ &= E_{q_0(z_0)}\left[\log p(x, T_\phi(z_0)) - \log q_0(z_0) + \log\left|\det\left(\frac{df_\phi}{dz_0}\right)\right|\right] \end{aligned}$$

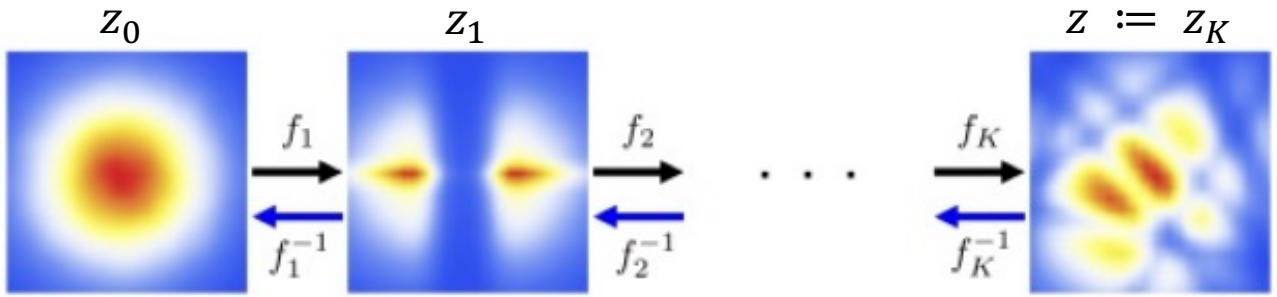
  - Computing ELBO requires $\log\left|\det\left(\frac{dT_\phi}{dz_0}\right)\right|$

reparam. trick:
$z \sim q(z) \Leftrightarrow z_0 \sim q_0(z_0), z = T_\phi(z_0)$

Rezende and Mohamed. Variational Inference with Normalizing Flows. ICML 2015

Normalizing Flows (Discrete Time)

- Variational inference with normalizing flow
  - Idea: define T_ϕ such that $\log|\det\left(\frac{dT_\phi}{dz_0}\right)|$ is easy to compute!
    - Chain simple invertible mappings together to make a flexible mapping



$$T_\phi = f_K \circ f_{K-1} \circ \cdots \circ f_1, f_k(\cdot) \coloneqq f_{\phi_k}(\cdot), \phi = \{\phi_k\}_{k=1}^K$$

- For each mapping, hopefully the Jacobian log-determinant is easy to compute

$$\Rightarrow \log|\det\left(\frac{dT_\phi}{dz_0}\right)| = \sum_{k=1}^{K} \log|\det(\frac{dz_k}{dz_{k-1}})|$$

How about the continuous-time limit?
$(k \to k/K, K \to \infty)$

Rezende and Mohamed. Variational Inference with Normalizing Flows. ICML 2015

Continuous-Time Limit: Continuous Normalizing Flows

- Continuous Normalizing Flows (CNFs):

$$z_1 = T(z_0), \qquad T(z_0) \coloneqq z_0 + \int_0^1 v_t(z_t)dt$$

- Motivation: Enhanced expressiveness
- Change-of-variable rule:

$$\partial_t \log p_t(z_t) = -\nabla_z \cdot v_t(z_t)$$

$$\log p_1(z_1) = \log p_0(z_0) - \int_0^1 \nabla_z \cdot v_t(z_t)\, dt$$

(expensive to compute!)

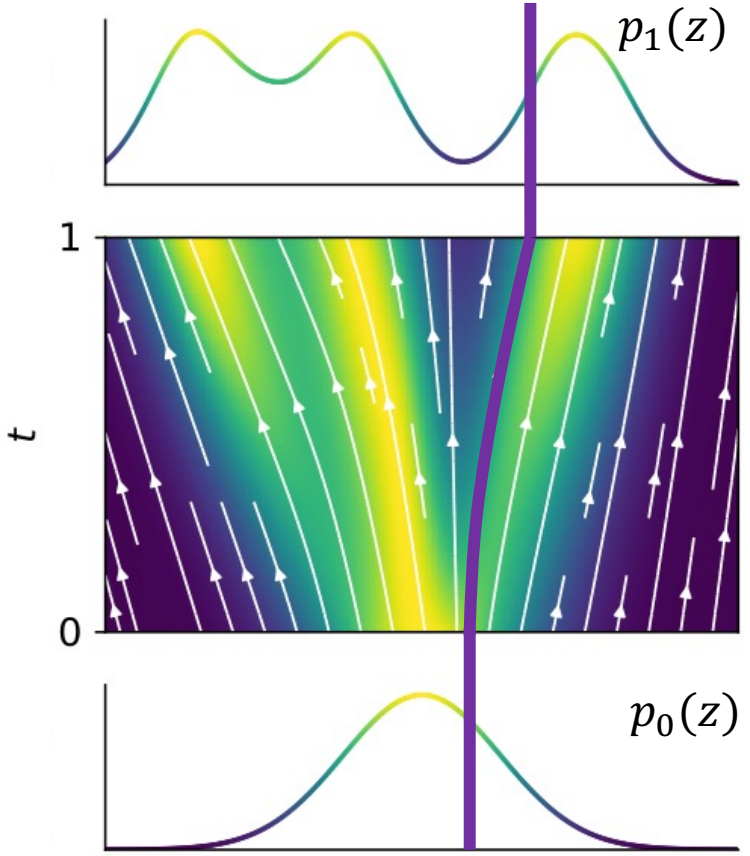


Figure adapted from Grathwohl et al. ICLR 2019

Continuity Equation

- Forward SDE evolution of z_t, $t: 0 \to 1$

$$dz_t = v_t(z_t)dt + \sigma_t(z_t)dW_t, \quad z_0 \sim p_0(z_0)$$

- Fokker-Planck Equation: marginal density evolution of $p_t(z)$, $t: 0 \to 1$, for any z (not necessarily for $z = z_t$)

$$\partial_t\, p_t(z) = -\nabla_z \cdot \big(v_t(z)p_t(z)\big) + \frac{1}{2}\nabla_z^2 \cdot (\sigma_t^2(z)p_t(z))$$

Continuity Equation (Cont.)

- Forward ODE evolution of z_t, $t: 0 \to 1$

$$dz_t = v_t(z_t)dt + \sigma_t(z_t)dW_t, \quad z_0 \sim p_0(z_0)$$

- Fokker-Planck Equation: marginal density evolution of $p_t(z)$, $t: 0 \to 1$, for any z (not necessarily for $z = z_t$)

$$\partial_t\, p_t(z) = -\nabla_z \cdot \big(v_t(z)p_t(z)\big) + \frac{1}{2}\nabla_z^2 \cdot \big(\sigma_t^2(z)p_t(z)\big)$$

Divide $p_t(z)$ on both sides: $\partial_t \log p_t(z) = -\nabla_z \cdot v_t(z) - \langle \nabla_z \log p_t(z), v_t(z) \rangle$

- Straight-forward reverse simulation of z_t, $t: 1 \to 0$

$$dz_t = v_t(z_t)dt, \quad z_1 \sim p_1(z_1) \qquad \Rightarrow \qquad z_{t-\delta_t} \approx z_t - v_t(z_t)\delta t$$

Continuous-Time Limit: Continuous Normalizing Flows

- Continuous Normalizing Flows (CNFs):

$$z_1 = T(z_0), \qquad T(z_0) \coloneqq z_0 + \int_0^1 v_t(z_t)dt$$

- Probability density evolves: $\{p_t(z)\}_{t\in[0,1]}$ satisfy

$$\partial_t \log p_t(z) = -\nabla_z \cdot v_t(z) - \langle \nabla_z \log p_t(z), v_t(z) \rangle$$

- Note the difference from change-of-variable rule:

$$\partial_t \log p_t(z_t) = -\nabla_z \cdot v_t(z_t)$$

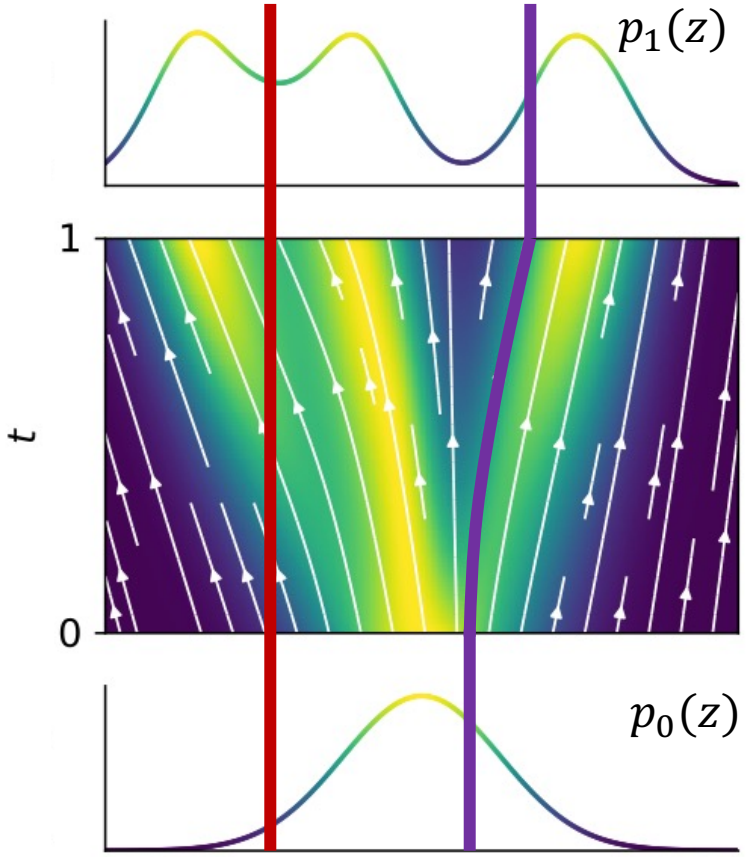


Figure adapted from Grathwohl et al. ICLR 2019

$$\pi(z) = \frac{1}{Z}\exp[-E(z)]$$

CNF-Based Posterior

E.g., tempering:
$E_t(z) = \beta_t E_0(z) + (1-\beta_t)E(z)$,
$\beta_0 = 1, \beta_1 = 0$

- Specify a density path:

$$\{p_t(z)\}_{t\in[0,1]}: \quad p_t(z) = \frac{1}{Z_t}\exp[-E_t(z)],$$

$p_0(z)$ easy to sample, $\quad p_1(z) = \pi(z)$, i.e., $E_1(z) \coloneqq E(z)$

Tian et al. Liouville Flow Importance Sampler. ICML 2024
Mate and Fleuret. Learning Interpolations between Boltzmann Densities. TMLR 2023
Chen et al. Neural Flow Samplers with Shortcut Models. arXiv:2502.07337

CNF-Based Posterior

E.g., tempering:
$E_t(z) = \beta_t E_0(z) + (1-\beta_t)E(z)$,
$\beta_0 = 1, \beta_1 = 0$

- Specify a density path:

$$\{p_t(z)\}_{t\in[0,1]}: \qquad p_t(z) = \frac{1}{Z_t}\exp[-E_t(z)],$$

$p_0(z)$ easy to sample, $\qquad p_1(z) = \pi(z)$, i.e., $E_1(z) \coloneqq E(z)$

- Learn a CNF network $v_\phi(z,t)$ by minimizing L2 error ("PINN loss"):

$$L(v_\phi) \coloneqq E_{q_t(z)}[\| \partial_t \log p_t(z) + \nabla_z \cdot v_\phi(z,t) + \langle \nabla_z \log p_t(z), v_\phi(z,t)\rangle \|_2^2]$$

Ensuring continuity equation to hold for every $z \sim q_t(z)$

Tian et al. Liouville Flow Importance Sampler. ICML 2024
Mate and Fleuret. Learning Interpolations between Boltzmann Densities. TMLR 2023
Chen et al. Neural Flow Samplers with Shortcut Models. arXiv:2502.07337

CNF-Based Posterior

E.g., tempering:
$E_t(z) = \beta_t E_0(z) + (1-\beta_t)E(z)$,
$\beta_0 = 1, \beta_1 = 0$

- Specify a density path:

$$\{p_t(z)\}_{t\in[0,1]}: \qquad p_t(z) = \frac{1}{Z_t}\exp[-E_t(z)],$$

$p_0(z)$ easy to sample, $\qquad p_1(z) = \pi(z)$, i.e., $E_1(z) \coloneqq E(z)$

- Learn a CNF network $v_\phi(z,t)$ by minimizing L2 error ("PINN loss"):

$$L(v_\phi) \coloneqq E_{q_t(z)}[\| \partial_t \log p_t(z) + \nabla_z \cdot v_\phi(z,t) + \langle \nabla_z \log p_t(z), v_\phi(z,t)\rangle \|_2^2]$$

Ensuring continuity equation to hold for every $z \sim q_t(z)$

- Simulate samples from $\pi(z)$ (approximately) by solving ODE:

$$x_0 \sim p_0(z), \qquad z_1 \coloneqq z_0 + \int_0^1 v_\phi(z_t,t)dt$$

Tian et al. Liouville Flow Importance Sampler. ICML 2024
Mate and Fleuret. Learning Interpolations between Boltzmann Densities. TMLR 2023
Chen et al. Neural Flow Samplers with Shortcut Models. arXiv:2502.07337

$$\pi(z) = \frac{1}{Z}\exp[-E(z)]$$

CNF-Based Posterior

$$= -\nabla_z E_t(z)$$

Training: $L(v_\phi) \coloneqq E_{q_t(z)}[\| \partial_t \log p_t(z) + \nabla_z \cdot v_\phi(z,t) + \langle \nabla_z \log p_t(z), v_\phi(z,t) \rangle \|_2^2]$

Sampling: $z_0 \sim p_0(z), \qquad z_1 \coloneqq z_0 + \int_0^1 v_\phi(z_t, t)dt$

- Challenges:
  - Selecting the "training distribution" $q_t(z)$ and estimating the expectation
    - Not necessary for $q_t(z) = p_t(z)$ but ideally $q_t(z) \approx p_t(z)$

Chen et al. Neural Flow Samplers with Shortcut Models. arXiv:2502.07337

$\pi(z) = \frac{1}{Z}\exp[-E(z)]$

CNF-Based Posterior

$= -\nabla_z E_t(z)$

Training: $L(v_\phi) \coloneqq E_{q_t(z)}[\| \partial_t \log p_t(z) + \nabla_z \cdot v_\phi(z,t) + \langle \nabla_z \log p_t(z), v_\phi(z,t) \rangle \|_2^2]$

Sampling: $z_0 \sim p_0(z), \qquad z_1 \coloneqq z_0 + \int_0^1 v_\phi(z_t, t)dt$

- Challenges:
  - Selecting the "training distribution" $q_t(z)$ and estimating the expectation
    - Not necessary for $q_t(z) = p_t(z)$ but ideally $q_t(z) \approx p_t(z)$
  - Estimating $\partial_t \log p_t(z)$:

$$p_t(z) \coloneqq \frac{1}{Z_t}\exp[-E_t(z)] \quad \Rightarrow \quad \partial_t \log p_t(z) = -\partial_t E_t(z) - \partial_t \log Z_t$$

(intractable)

Chen et al. Neural Flow Samplers with Shortcut Models. arXiv:2502.07337

$\pi(z) = \frac{1}{Z}\exp[-E(z)]$

CNF-Based Posterior

$= -\nabla_z E_t(z)$

Training: $L(v_\phi) \coloneqq E_{q_t(z)}[\| \partial_t \log p_t(z) + \nabla_z \cdot v_\phi(z,t) + \langle \nabla_z \log p_t(z), v_\phi(z,t) \rangle \|_2^2]$

Sampling: $z_0 \sim p_0(z), \qquad z_1 \coloneqq z_0 + \int_0^1 v_\phi(z_t, t)dt$

- Challenges:
  - Selecting the "training distribution" $q_t(z)$ and estimating the expectation
    - Not necessary for $q_t(z) = p_t(z)$ but ideally $q_t(z) \approx p_t(z)$
  - Estimating $\partial_t \log p_t(z)$:

    $p_t(z) \coloneqq \frac{1}{Z_t}\exp[-E_t(z)] \quad \Rightarrow \quad \partial_t \log p_t(z) = -\partial_t E_t(z) - \partial_t \log Z_t$

    (intractable)
  - Solving the ODE flow simulation in a fast way

Chen et al. Neural Flow Samplers with Shortcut Models. arXiv:2502.07337

$$\pi(z) = \frac{1}{Z}\exp[-E(z)]$$

Using "Training Data" $q_t(z)$

$$L(v_\phi) \coloneqq E_{q_t(z)}[\| \partial_t \log p_t(z) + \nabla_z \cdot v_\phi(z,t) + \langle \nabla_z \log p_t(z), v_\phi(z,t) \rangle \|_2^2]$$

- Sampling under $q_t(z) \approx p_t(z)$ via *velocity-driven* SMC:
  - A typical SMC method (e.g., Hamiltonian AIS):
    - Pick $0 = t_0 < t_1 < t_2 < \cdots < t_M = 1$ and run SMC with path $\{p_{t_m}(z)\}_{m=0}^{M}$ as proposals
    - Compute the importance weights by accumulating density ratios through time
    - Resampling is required by monitoring ESS
  - The steps for approximately drawing samples from $p_{t_m}(z)$:
    - Transport from previous step: $\tilde{z}_{t_m} = z_{t_{m-1}} + \int_{t_{m-1}}^{t_m} v_\phi(z_t, t)dt$ ("prediction")
    - Run (short-chain) HMC: $z_{t_m} = HMC(\tilde{z}_{t_m})$ ("correction")

Neal. Annealed Importance Sampling. Stats. Comp., 2001
Neal. MCMC using Hamiltonian Dynamics. Handbook of MCMC, 2010
Chen et al. Neural Flow Samplers with Shortcut Models. arXiv:2502.07337

$$\pi(z) = \frac{1}{Z}\exp[-E(z)]$$

Estimating $\partial_t \log Z_t$

Monte Carlo estimate can have high variance!

$$\partial_t \log Z_t = \frac{1}{Z_t}\partial_t \int \exp[-E_t(z)]\, dz = -\frac{1}{Z_t}\int \exp[-E_t(z)]\, \partial_t E_t(z) dz = -E_{p_t(z)}[\partial_t E_t(z)]$$

- Solution: Stein control variate with $z^k \sim p_t(z)$

(Langevin-Stein Operator)

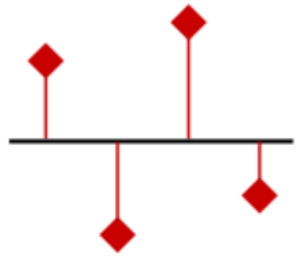
$$-E_{p_t(z)}[\partial_t E(z)] \approx \frac{1}{K}\sum_{k=1}^{K} -\partial_t E_t(z^k) + \beta\left[\nabla_z \cdot v_\phi(z^k, t) + \langle \nabla_x \log p_t(z^k), v_\phi(z^k, t)\rangle\right]$$

Liu et al. Action-dependent Control Variates for Policy Optimization via Stein Identity. ICLR 2018
Chen et al. Neural Flow Samplers with Shortcut Models. arXiv:2502.07337

Variance Reduction for Monte Carlo

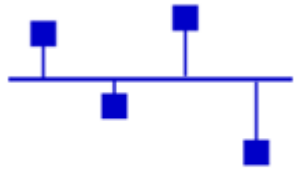
- Control variate method:
  - Assume we want to estimate with MC simulation

$$E_{q(z)}[F(z)] \approx \frac{1}{K}\sum_{k=1}^{K} F\left(z^k\right), \quad z^k \sim q(z)$$



$F(z)$

  - Control variate: define a control function $G(z)$ satisfying:
    - $V_{q(z)}[G(z)] < \infty$
    - Known or fast computable $E_{q(z)}[G(z)]$

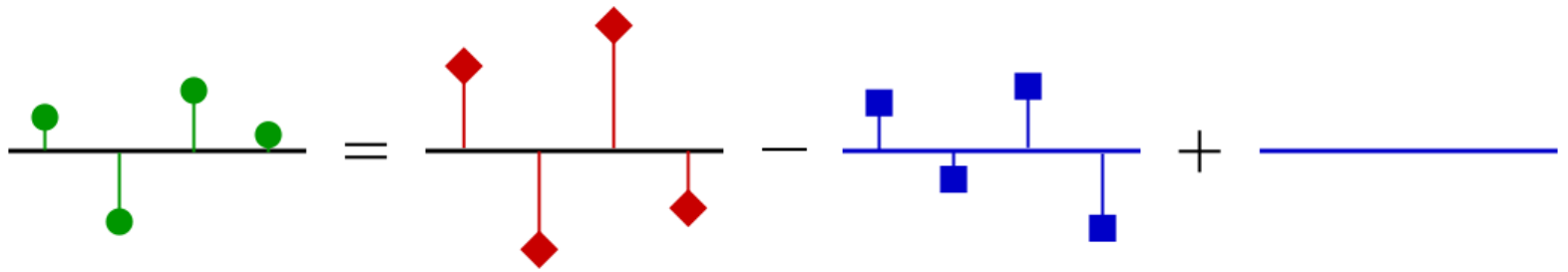


$G(z)$

Variance Reduction for Monte Carlo

- Control variate method:
  - Then define the new MC estimator

$$E_{q(z)}[F(z)] \approx \frac{1}{K}\sum_{k=1}^{K} \hat{F}\left(z^k\right), \; z^k \sim q(z),$$



$$\hat{F}(z) = F(z) - G(z) + E_{q(z)}[G(z)]$$

$$V_{q(z)}[\hat{F}(z)] = V_{q(z)}[F(z)] + V_{q(z)}[G(z)] - 2\,Cov_{q(z)}[F(z), G(z)]$$

< 0 if F and G are strongly and positively correlated

$$\pi(z) = \frac{1}{Z}\exp[-E(z)]$$

Estimating $\partial_t \log Z_t$

Monte Carlo estimate can have high variance!

$$\partial_t \log Z_t = \frac{1}{Z_t}\partial_t \int \exp[-E_t(z)]\, dz = -\frac{1}{Z_t}\int \exp[-E_t(z)]\, \partial_t E_t(z) dz = -E_{p_t(z)}[\partial_t E_t(z)]$$

- Solution: Stein control variate with $z^k \sim p_t(z)$

(Langevin-Stein Operator)

$$-E_{p_t(z)}[\partial_t E(z)] \approx \frac{1}{K}\sum_{k=1}^{K} -\partial_t E_t(z^k) + \beta\left[\nabla_z \cdot v_\phi(z^k, t) + \langle \nabla_x \log p_t(z^k), v_\phi(z^k, t)\rangle\right]$$

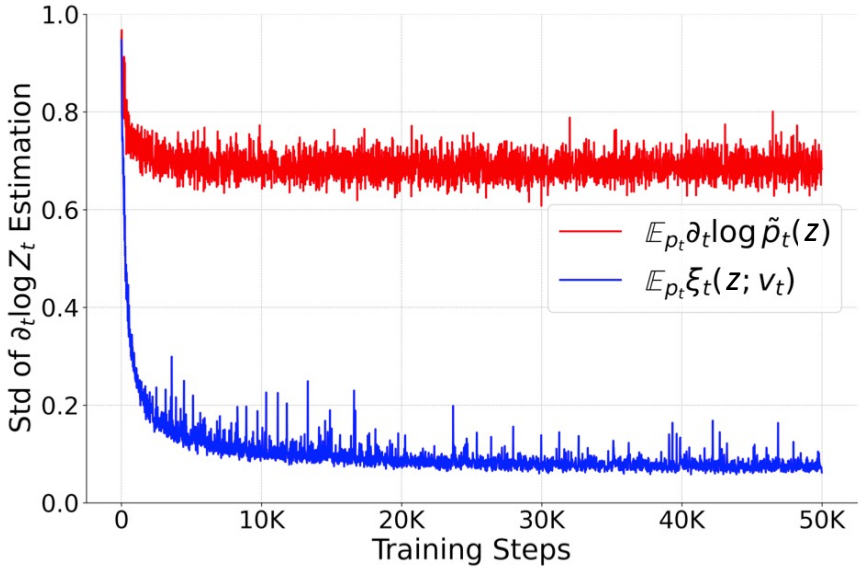
- Stein's Identity ensures unbiasedness: for any $v_\phi(z,t)$

$$E_{p_t(z)}\left[\nabla_z \cdot v_\phi(z,t) + \langle \nabla_z \log p_t(z), v_\phi(z,t)\rangle\right] = 0$$

- Continuity equation for *globally optimal* $v_\phi(z,t)$

$$\underbrace{-\partial_t E_t(z) + \left[\nabla_z \cdot v_\phi(z,t) + \left\langle \nabla_z \log p_t(z), v_\phi(z,t)\right\rangle\right]}_{:= \xi_t(z; v_\phi(z,t))} = \partial_t \log Z_t$$

⇒ Variance is 0 when $\beta = 1$ and $v_\phi(z,t)$ is optimal



Liu et al. Action-dependent Control Variates for Policy Optimization via Stein Identity. ICLR 2018
Chen et al. Neural Flow Samplers with Shortcut Models. arXiv:2502.07337

$$\pi(z) = \frac{1}{Z}\exp[-E(z)]$$

Speeding up with Shortcuts

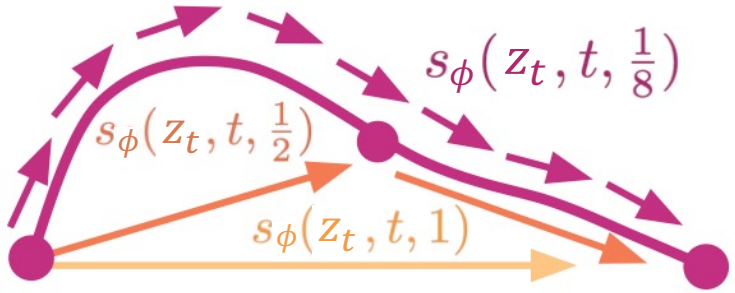
Sampling: $z_0 \sim p_0(z), \quad z_1 := z_0 + \int_0^1 v_\phi(z_t, t)dt$

Naïve Euler method requires a lot of steps!

Solution: Shortcut models

$$z_{t+d} \leftarrow z_t + s_\phi(z_t, t, d)d$$

- A shortcut model s_θ is a valid ODE solver if for any z_t:
  - $s_\phi(z_t, t, 0) := v_\phi(z_t, t)$
  - $s_\phi(z_t, t, 2d) = \frac{1}{2}s_\phi(z_t, t, d) + \frac{1}{2}s_\phi(z_{t+d}, t+d, d)$



- Training a neural flow shortcut sampler:

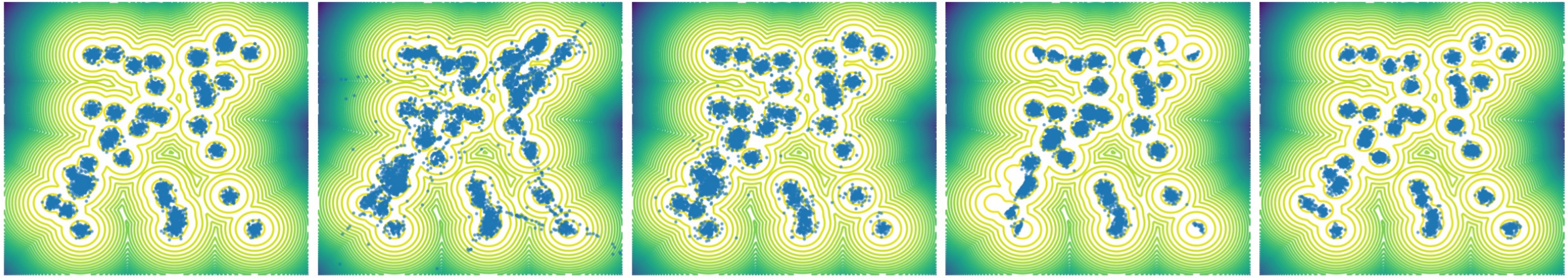
$$\tilde{L}(s_\phi) := L\left(s_\phi(\cdot,\cdot,0)\right) + E_{q(z_t,d)}\left[\|s_\phi(z_t, t, 2d) - \frac{1}{2}s_\phi(z_t, t, d) - \frac{1}{2}s_\phi(z_{t+d}, t+d, d)\|_2^2\right]$$

flow learning — enforcing consistency

Frans et al. One Step Diffusion via Shortcut Models. ICLR 2025

Examples & Applications

Target $\pi(z)$: mixture of 40 Gaussians



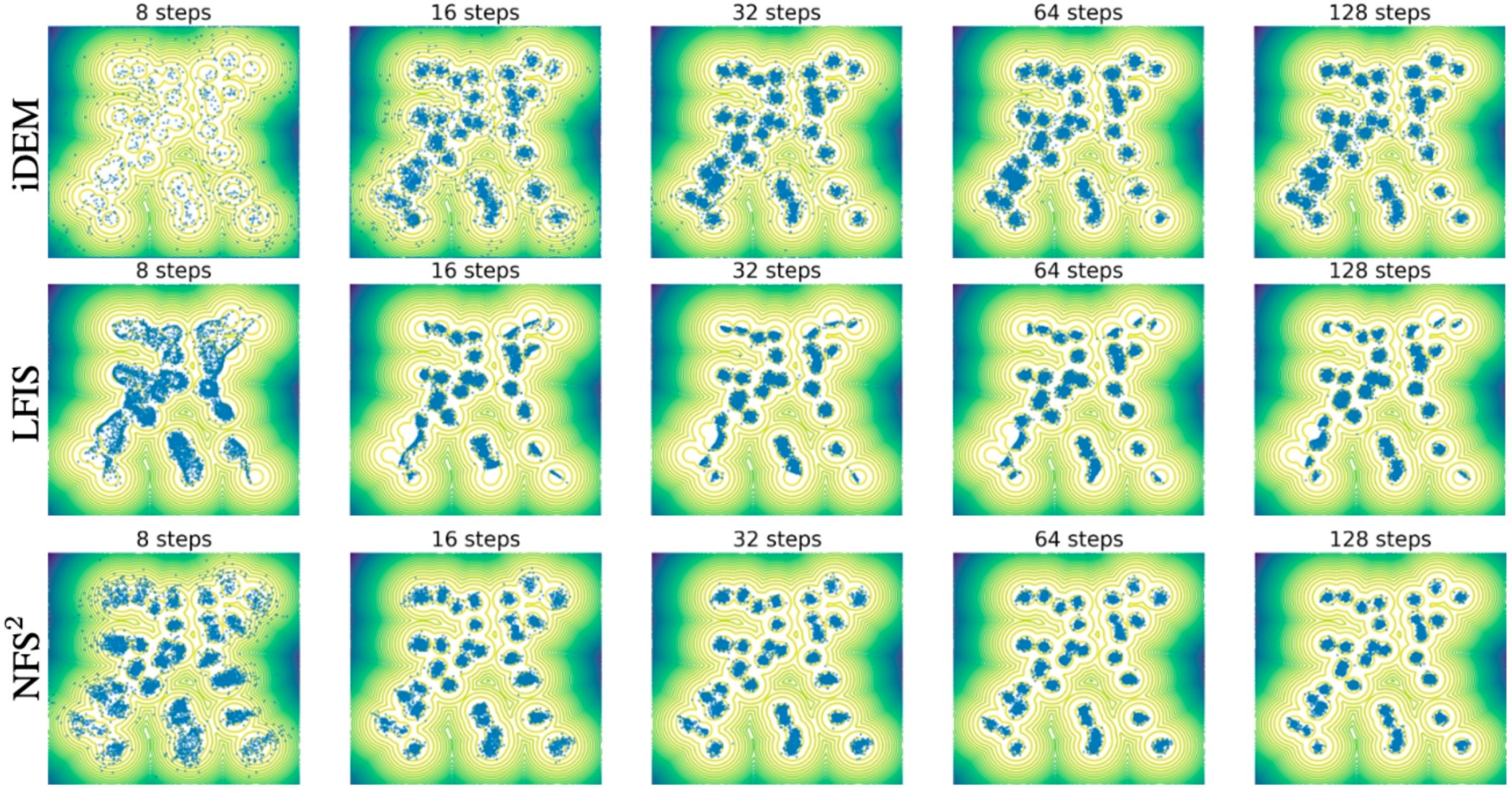
Ground Truth | FAB | iDEM | LFIS | NFS^2 (ours)

- "Ground Truth": samples from mixture of Gaussians
- FAB: normalising flow transport map, trained by alpha-divergence, "data" from AIS + replay buffer
- iDEM: diffusion-based, score estimation via importance sampling + replay buffer
- LFIS: continuity equation-based loss, no amortization across t, simple importance sampling for $\partial_t \log Z_t$

Midgley et al. Flow Annealed Importance Sampling Bootstrap. ICLR 2023
Akhound-Sadegh et al. Iterated Denoising Energy Matching for Sampling from Boltzmann Densities. ICML 2024
Tian et al. Liouville Flow Importance Sampler. ICML 2024
Chen et al. Neural Flow Samplers with Shortcut Models. arXiv:2502.07337

Examples & Applications

Target $\pi(z)$: mixture of 40 Gaussians



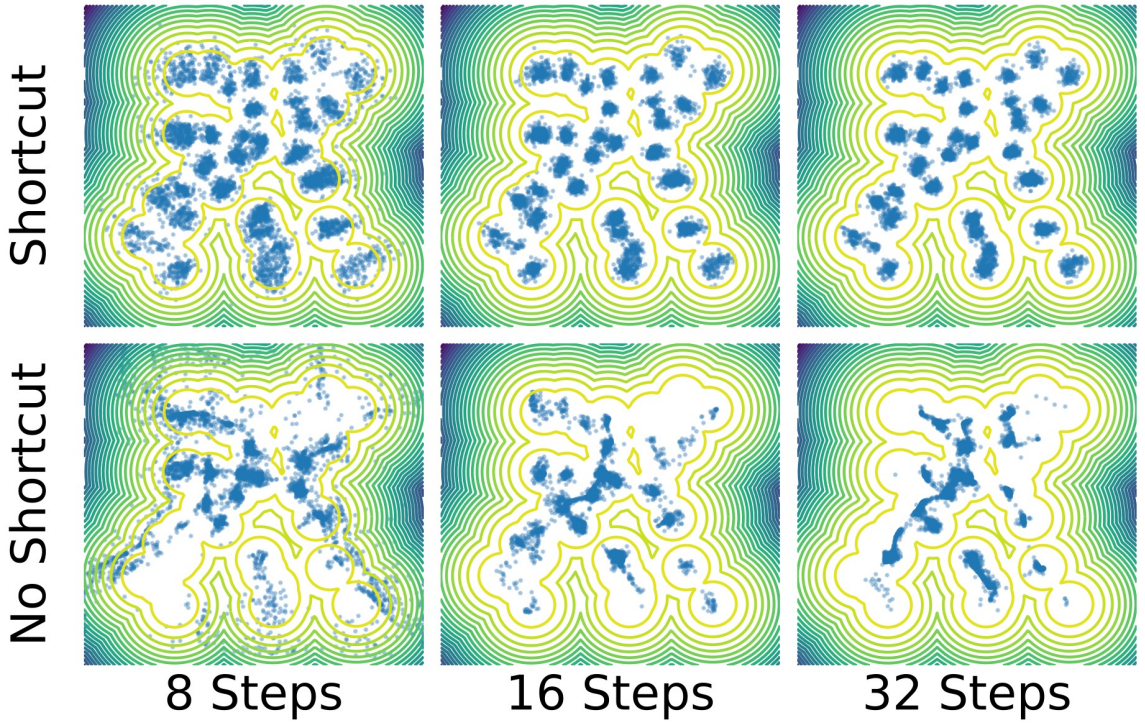
Chen et al. Neural Flow Samplers with Shortcut Models. arXiv:2502.07337

Examples & Applications

Target $\pi(z)$: mixture of 40 Gaussians

Ablation: Training with or without the shortcut consistency loss

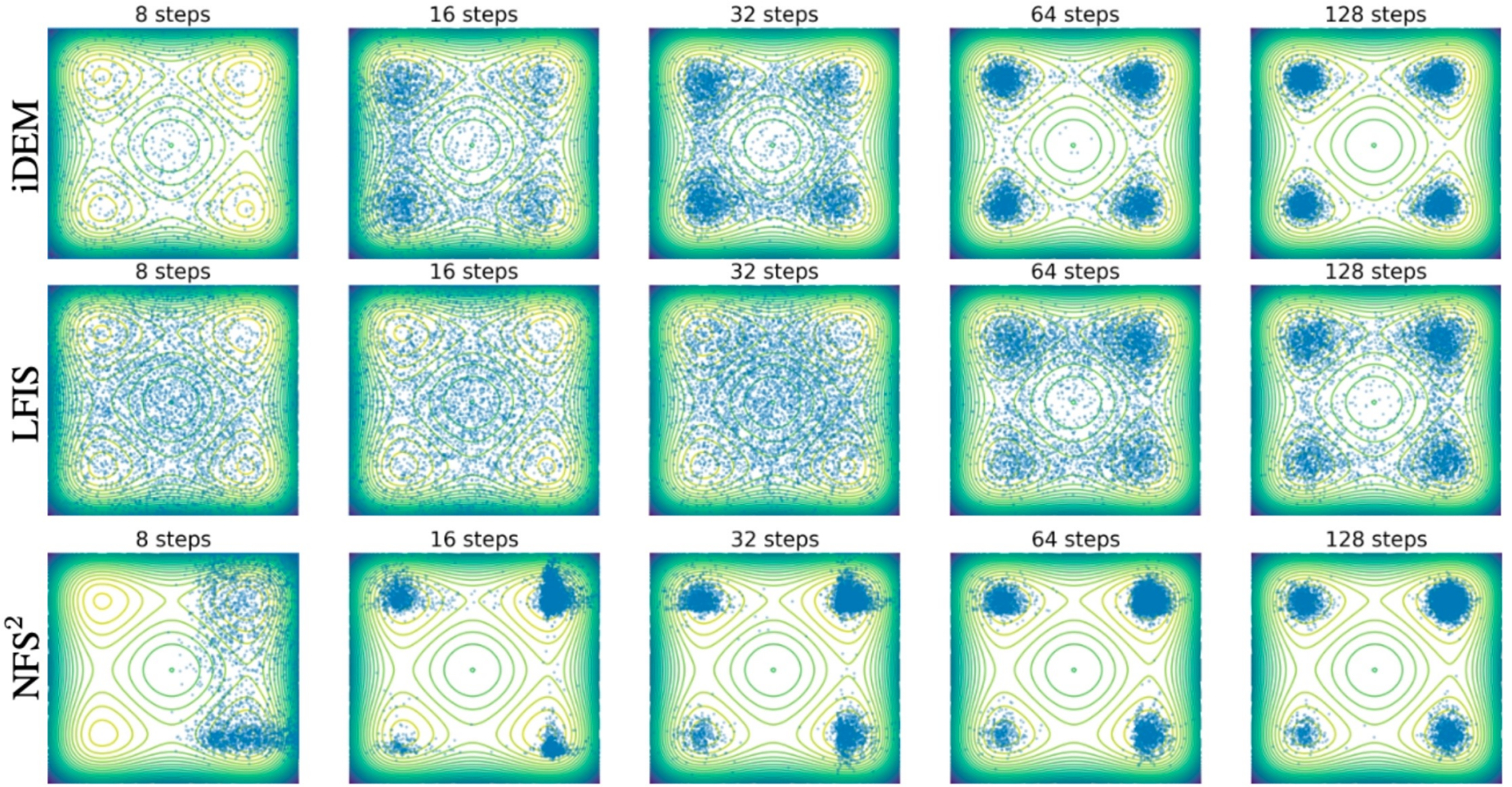
$$\tilde{L}(s_\phi) \coloneqq \underbrace{L\left(s_\phi(\cdot,\cdot,0)\right)}_{\text{flow learning}} + \underbrace{E_{q(z_t,d)}\left[\left\|s_\phi(z_t,t,2d) - \frac{1}{2}s_\phi(z_t,t,d) - \frac{1}{2}s_\phi(z_{t+d},t+d,d)\right\|_2^2\right]}_{\text{enforcing consistency}}$$



Chen et al. Neural Flow Samplers with Shortcut Models. arXiv:2502.07337

Examples & Applications

Target $\pi(z)$: MW32 (32D Many-Well potential with $2^{16} = 65{,}536$ modes) (only showing two dimensions here)



Chen et al. Neural Flow Samplers with Shortcut Models. arXiv:2502.07337

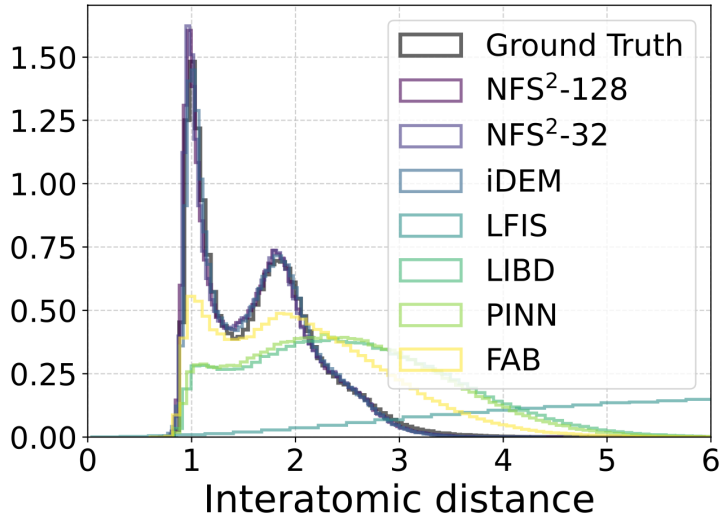
Examples & Applications

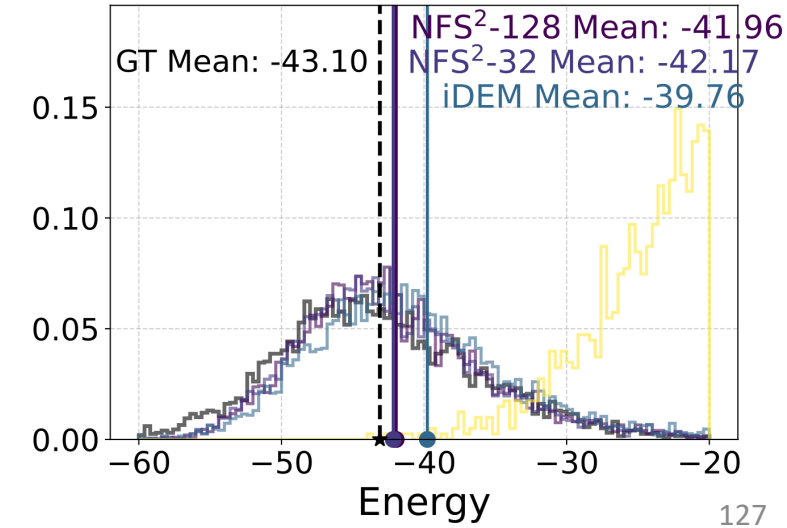
Target $\pi(z)$: LJ-13 (39D, energy dependent on atom distances)

$$\pi(z) \propto \exp[-\frac{E(z)}{k_B T}],\ E(z) \coloneqq \sum_{i,j\neq i} V_{LJ}(r_{ij}),\ r_{ij} = \left\|z_i - z_j\right\|_2^2$$

$$V_{LJ}(r_{ij}) \coloneqq 4\epsilon\left[\frac{\sigma^{12}}{r_{ij}^{12}} - \frac{\sigma^{6}}{r_{ij}^{6}}\right]$$

Method	Energy W_2	Energy TV	Distance TV
FAB	31.28 ± 0.31	0.94 ± 0.03	0.26 ± 0.01
iDEM	13.13 ± 5.30	0.31 ± 0.01	0.03 ± 0.01
LFIS	∞	$*$	0.88 ± 0.00
LIBD	49.89 ± 0.12	$*$	0.45 ± 0.01
PINN	48.3 ± 0.1	$*$	0.44 ± 0.00
NFS^2-128 (ours)	1.93 ± 0.01	0.19 ± 0.10	0.05 ± 0.00
NFS^2-64 (ours)	1.51 ± 0.10	0.19 ± 0.10	0.06 ± 0.01
NFS^2-32 (ours)	1.62 ± 0.20	0.20 ± 0.01	0.06 ± 0.00
NFS^2-8 (ours)	29.57 ± 16.06	0.30 ± 0.01	0.22 ± 0.01





Summary (CNF-based Approximations)

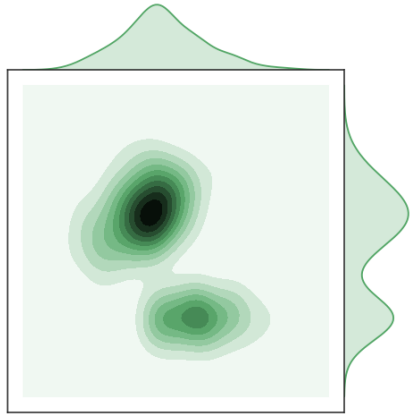
Constructing flexible q distribution via transport:

- Normalizing flows as stacking invertible transformations (very flexible)
- The VI objective needs to compute $\log|\det\left(\frac{dT}{dz}\right)|$
- Add many layers and take continuous-limit: CNF-based approximations
  - The log-determinant requires explicitly integrating an ODE (expensive)
  - "simulation-free" approach by leveraging the continuity equation ("PINN loss")
  - Many challenges need solutions

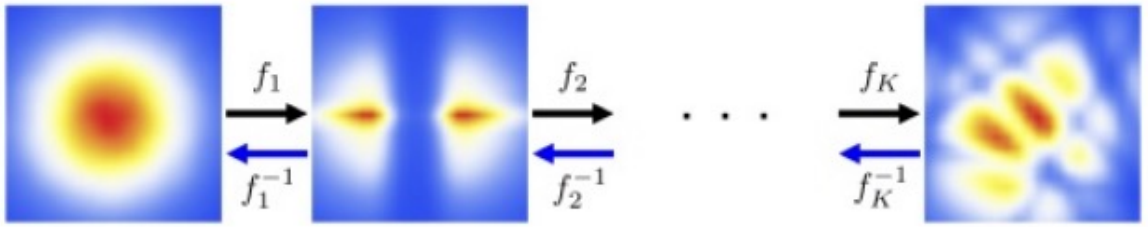
Training: $L(v_\phi) \coloneqq E_{q_t(z)}[\| \partial_t \log p_t(z) + \nabla_z \cdot v_\phi(z,t) + \langle \nabla_z \log p_t(z), v_\phi(z,t) \rangle \|_2^2]$

Sampling: $z_0 \sim p_0(z), \qquad z_1 \coloneqq z_0 + \int_0^1 v_\phi(z_t, t)dt$

Designing q Distributions



Auxiliary variables & mixture distributions



Normalizing flows

Use many layers (hierarchical) + continuous-time limit

Diffusion SDE & Continuous Normalizing Flow posteriors for approximate inference

Converting between each other?

CNF-Based Posterior

- Specify a density path:

$$\{p_t(z)\}_{t\in[0,1]}: \qquad p_t(z) = \frac{1}{Z_t}\exp[-E_t(z)],$$

$p_0(z)$ easy to sample, $\qquad p_1(z) = \pi(z)$, i.e., $E_1(z) \coloneqq E(z)$

- Learn a CNF network $v_\phi(z,t)$ by minimizing L2 error ("PINN loss"):

$$L(v_\phi) \coloneqq E_{q_t(z)}[\| \partial_t \log p_t(z) + \nabla_x \cdot v_\phi(z,t) + \langle \nabla_z \log p_t(z), v_\phi(z,t)\rangle \|_2^2]$$

Ensuring continuity equation to hold for every $z \sim q_t(z)$

- Simulate samples from $\pi(z)$ (approximately) by solving ODE:

$$x_0 \sim p_0(z), \qquad z_1 \coloneqq z_0 + \int_0^1 v_\phi(z_t,t)dt$$

CNF-Based Posterior

- Specify a density path:

This path can also be defined via diffusion!

$$dz_t = \beta_t z_t dt + \sigma\sqrt{2\beta_t} dW_t,\ t: 1 \to 0$$

$$\{p_t(z)\}_{t\in[0,1]}: \qquad p_t(z) = \frac{1}{Z_t}\exp[-E_t(z)],$$

$$p_0(z) \text{ easy to sample}, \qquad p_1(z) = \pi(z), \text{ i.e., } E_1(z) \coloneqq E(z)$$

- Learn a CNF network $v_\phi(z,t)$ by minimizing L2 error ("PINN loss"):

require estimations

$$L(v_\phi) \coloneqq E_{q_t(z)}[\| \boxed{\partial_t \log p_t(z)} + \nabla_x \cdot v_\phi(z,t) + \langle \boxed{\nabla_z \log p_t(z)}, v_\phi(z,t)\rangle \|_2^2]$$

Ensuring continuity equation to hold for every $z \sim q_t(z)$

- Simulate samples from $\pi(z)$ (approximately) by solving ODE:

$$x_0 \sim p_0(z), \qquad z_1 \coloneqq z_0 + \int_0^1 v_\phi(z_t, t)dt$$

CNF-Based Posterior

- Specify a density path:

This path can also be defined via diffusion!

$$dz_t = \beta_t z_t dt + \sigma\sqrt{2\beta_t}dW_t,\ t: 1 \to 0$$

$$\{p_t(z)\}_{t\in[0,1]}: \qquad p_t(z) = \frac{1}{Z_t}\exp[-E_t(z)],$$

$$p_0(z) \text{ easy to sample}, \qquad p_1(z) = \pi(z), \text{ i.e., } E_1(z) \coloneqq E(z)$$

Time marginal: $p_t(z_t) = \int N\left(z_t \middle| \sqrt{1-\lambda_t}z_1, \sigma^2\lambda_t I\right) p_1(z_1)dz_1, \quad \lambda_t = 1 - \exp[-2\int_t^1 \beta_\tau d\tau]$

$\coloneqq p_t(z_t|z_1)$

Estimate the score $\nabla_{z_t}\log p_t(z_t)$ via importance sampling: $N\left(z_1; \frac{z_t}{\sqrt{1-\lambda_t}}, \frac{\sigma^2\lambda_t}{1-\lambda_t}I\right) \coloneqq n_t(z_1|z_t)$

Denoising Score Identity

Target Score Identity

$$\nabla_{z_t}\log p_t(z_t) = E_{n_t(z_1|z_t)}[w(z_1)\nabla_{z_t}\log p_t(z_t|z_1)] = E_{n_t(z_1|z_t)}\left[w(z_1)\nabla_{z_1}\log r_1(z_1)\right]$$

$w(z_1) \propto \exp[-E_1(z_1)]$ (unnormalized target density)

$$\pi(z) = \frac{1}{Z}\exp[-E(z)]$$

CNF-Based Posterior

- Specify a density path:

This path can also be defined via diffusion!

$$dz_t = \beta_t z_t dt + \sigma\sqrt{2\beta_t}dW_t,\ t: 1 \to 0$$

$$\{p_t(z)\}_{t\in[0,1]}: \qquad p_t(z) = \frac{1}{Z_t}\exp[-E_t(z)],$$

$$p_0(z) \text{ easy to sample}, \qquad p_1(z) = \pi(z), \text{ i.e., } E_1(z) \coloneqq E(z)$$

Time marginal: $p_t(z_t) = \int N\left(z_t \middle| \sqrt{1-\lambda_t}z_1, \sigma^2\lambda_t I\right) p_1(z_1)dz_1, \quad \lambda_t = 1 - \exp[-2\int_t^1 \beta_\tau d\tau]$

$\coloneqq p_t(z_t|z_1)$

Estimate the score $\nabla_{z_t}\log p_t(z_t)$ via importance sampling: $N\left(z_1; \frac{z_t}{\sqrt{1-\lambda_t}}, \frac{\sigma^2\lambda_t}{1-\lambda_t}I\right) \coloneqq n_t(z_1|z_t)$

Denoising Score Identity

Target Score Identity

$$\nabla_{z_t}\log p_t(z_t) = E_{n_t(z_1|z_t)}[w(z_1)\nabla_{z_t}\log p_t(z_t|z_1)] = E_{n_t(z_1|z_t)}\left[w(z_1)\nabla_{z_1}\log r_1(z_1)\right]$$

$w(z_1) \propto \exp[-E_1(z_1)]$ (unnormalized target density)

Estimate the "time-score" $\partial_t \log p_t(z_t)$ also via importance sampling:

$$\partial_t \log p_t(z_t) = E_{n_t(z_1|z_t)}[w(z_1)\partial_t \log p_t(z_t|z_1)]$$

Converting Between ODEs and SDEs

- Forward ODE/SDE evolution of z_t, $t: 0 \to 1$, $z_0 \sim p_0(z_0)$

CNF-based sampler (ODE)

$$dz_t = v_t(z_t)dt$$

Diffusion-based sampler (SDE)

$$dz_t = f_t(z_t)dt + \sigma_t dW_t$$

- Fokker-Planck Equation: marginal density evolution of $p_t(z)$, $t: 0 \to 1$, for any z (not necessarily for $z = z_t$)

CNF-based sampler (ODE)

$$\partial_t\, p_t(z) = -\nabla_z \cdot \big(v_t(z)p_t(z)\big)$$

Diffusion-based sampler (SDE)

$$\partial_t\, p_t(z) = -\nabla_z \cdot \big(f_t(z)p_t(z)\big) + \frac{1}{2}\nabla_z^2 \cdot (\sigma_t^2 p_t(z))$$

Song et al. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR 2021

Converting Between ODEs and SDEs

- Forward ODE/SDE evolution of z_t, $t: 0 \to 1$, $z_0 \sim p_0(z_0)$

CNF-based sampler (ODE)

$$dz_t = v_t(z_t)dt$$

Diffusion-based sampler (SDE)

$$dz_t = f_t(z_t)dt + \sigma_t dW_t$$

- Fokker-Planck Equation: marginal density evolution of $p_t(z)$, $t: 0 \to 1$, for any z (not necessarily for $z = z_t$)

CNF-based sampler (ODE)

$$\partial_t\, p_t(z) = -\nabla_z \cdot \big(v_t(z)p_t(z)\big)$$

Diffusion-based sampler (SDE)

$$\partial_t\, p_t(z) = -\nabla_z \cdot \big(f_t(z)p_t(z)\big) + \frac{1}{2}\nabla_z^2 \cdot (\sigma_t^2 p_t(z))$$

$$\nabla_z^2 \cdot \Big(\sigma_t^2 p_t(z)\Big) = \nabla_z \cdot \Big[\nabla_z \Big(\sigma_t^2 p_t(z)\Big)\Big]$$

$$\nabla_z \Big(\sigma_t^2 p_t(z)\Big) = p_t(z)\sigma_t^2 \nabla_z \log p_t(z)$$

Song et al. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR 2021

Converting Between ODEs and SDEs

- Forward ODE/SDE evolution of z_t, $t: 0 \to 1$, $z_0 \sim p_0(z_0)$

CNF-based sampler (ODE)

$$dz_t = v_t(z_t)dt$$

Diffusion-based sampler (SDE)

$$dz_t = f_t(z_t)dt + \sigma_t dW_t$$

- Fokker-Planck Equation: marginal density evolution of $p_t(z)$, $t: 0 \to 1$, for any z (not necessarily for $z = z_t$)

CNF-based sampler (ODE)

$$\partial_t\, p_t(z) = -\nabla_z \cdot \big(v_t(z) p_t(z)\big)$$

Diffusion-based sampler (SDE)

$$\partial_t\, p_t(z) = -\nabla_z \cdot \left(\left[f_t(z) - \frac{\sigma_t^2}{2}\nabla_z \log p_t(z)\right] p_t(z)\right)$$

Making both samplers to induce the same density path: $v_t(z) = f_t(z) - \frac{\sigma_t^2}{2}\nabla_z \log p_t(z)$

("Probability flow ODE")

Song et al. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR 2021

Converting Between CNF and Diffusion Samplers

- Building the density path by diffusing $z_1 \sim p_1(z_1) \coloneqq \pi(z_1)$:

$$dz_t = \beta_t z_t dt + \sigma\sqrt{2\beta_t} dW_t,\ t: 1 \to 0$$

$$p_t(z_t) = \int N\left(z_t \middle| \sqrt{1-\lambda_t} z_1, \sigma^2 \lambda_t I\right) p_1(z_1) dz_1, \quad \lambda_t = 1 - \exp[-2\int_t^1 \beta_\tau d\tau]$$

- Forward ODE/SDE evolution of $z_t,\ t: 0 \to 1,\ z_0 \sim p_0(z_0)$

CNF-based sampler (ODE)

$$dz_t = v_\phi(z_t, t)dt$$

Diffusion-based sampler (SDE)

$$dz_t = [\beta_t z_t + 2\sigma^2 \beta_t s_\phi(z_t, t)]dt + \sigma\sqrt{2\beta_t} dW_t$$

Making both samplers to induce the same density path $\{p_t(z)\}_{t\in[0,1]}$:

$$v_\phi(z_t, t) = \beta_t z_t + 2\sigma^2 \beta_t s_\phi(z_t, t) - \sigma^2 \beta_t \nabla_z \log p_t(z) \approx \beta_t z_t + \sigma^2 \beta_t s_\phi(z_t, t)$$

Song et al. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR 2021

Converting Between CNF and Diffusion Samplers

sample evolution: SDE vs Probability flow ODE, $v_t(z) = f_t(z) - \frac{\sigma_t^2}{2}\nabla_z \log p_t(z)$

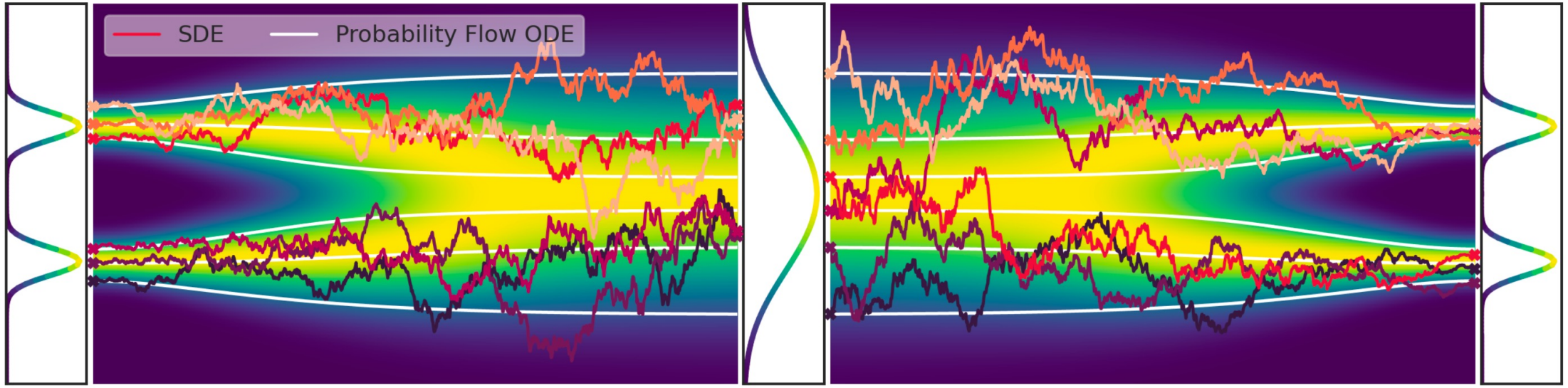
$dz_t = f_t(z_t)dt - \sigma_t^2 \nabla_z \log p_t(z_t) + \sigma_t dW_t,\ t: 1 \to 0$

$dz_t = f_t(z_t)dt + \sigma_t dW_t,\ t: 0 \to 1$

z_1 z_0 z_1

$dz_t = v_t(z_t)dt,\ t: 1 \to 0$

$dz_t = v_t(z_t)dt,\ t: 0 \to 1$



$$\partial_t\, p_t(z) = -\nabla_z \cdot \big(f_t(z)p_t(z)\big) + \frac{1}{2}\nabla_z^2 \cdot (\sigma_t^2 p_t(z))$$

$p_1(z) \coloneqq \pi(z)$ $p_0(z)$ $p_1(z) \coloneqq \pi(z)$

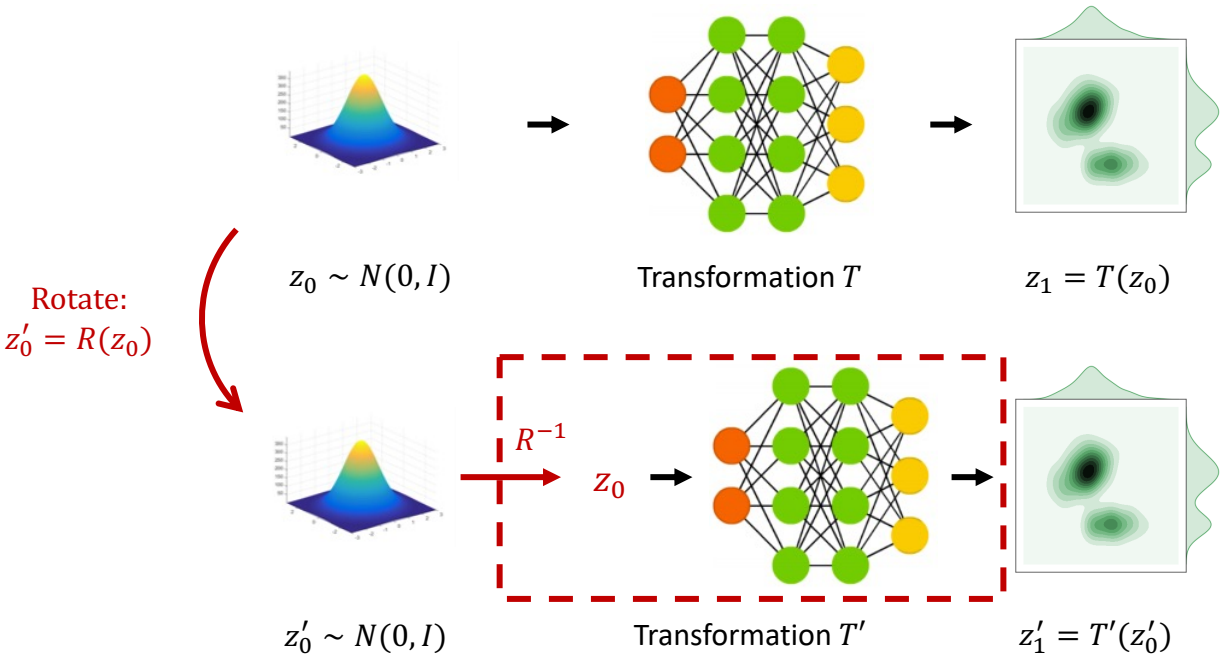
density evolution

$$\partial_t\, p_t(z) = -\nabla_z \cdot \big(v_t(z)p_t(z)\big)$$

Figure adapted from Song et al. ICLR 2021

Why Can We Construct So Many Different Diffusion/CNF-based Approximations?

- Ultimately, we only care about distribution match $q(z_1) \approx p_1(z_1) \coloneqq \pi(z_1)$!



Unidentifiability:
Given $q_{CNF}(z_1) = q_{SDE'}(z_1)$,
we cannot show $CNF = CNF'$,

where CNF' = probability flow of SDE',
(unless you make further assumptions)!

(The causal representation learning community should have known this result by heart :)

Design Principles I

Recall Design Principles I:

Step 1: Specify a divergence $D(q;\pi)$ between variational density $q(z)$ and target $\pi(z)$

Step 2: Find the member of the family $q^* \in Q$ that minimizes the divergence, i.e.,

$$q^* = argmin_{q \in Q}\, D(q;\pi)$$

Step 3: Use variational density q^* instead of target π in downstream tasks: e.g.,

$$E_{\pi(z)}[F(z)] \approx E_{q^*(z)}[F(z)] \approx \frac{1}{K}\sum_k^K F(z_k)\,, z_k \sim q^*(z)$$

Expressivity

- correlations
- Skew and kurtosis
- Multi-modality

Tractability

- Easy and fast to evaluate $q(z)$
- Fast sampling from its distribution (e.g., to approximate expectations)
- Ease of optimization

Design Principles II

a differentiable and easy-to-evaluate objective $L(q;\pi)$
whose global optimum is $q^* = \pi$ if $\pi \in Q$

Step 1: Specify ~~a divergence $D(q;\pi)$ between variational density $q(z)$ and target $\pi(z)$~~

Step 2: Find the member of the family $q^* \in Q$ that minimizes the divergence, i.e.,

~~$q^* = argmin_{q \in Q} D(q;\pi)$~~ $q^* = argmin_{q \in Q} L(q;\pi)$

Step 3: Use variational density q^* instead of target π in downstream tasks: e.g.,

$$E_{\pi(z)}[F(z)] \approx E_{q^*(z)}[F(z)] \approx \frac{1}{K}\sum_k^K F(z_k)\,, z_k \sim q^*(z)$$

Expressivity 👍👍👍

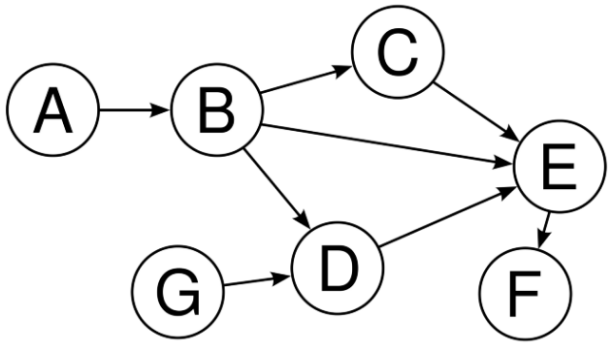
towards universal distribution approximators

Tractability

- ~~Easy and fast to evaluate $q(z)$~~ no need in many cases
- ~~Fast sampling~~ can accept slower sampling speed from its distribution (e.g., to approximate expectations)
- Ease of optimization and not too slow

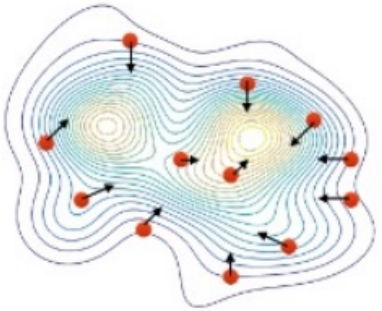
Other Methods (Not Covered Today)

Other designs of q distributions



Structured approximations

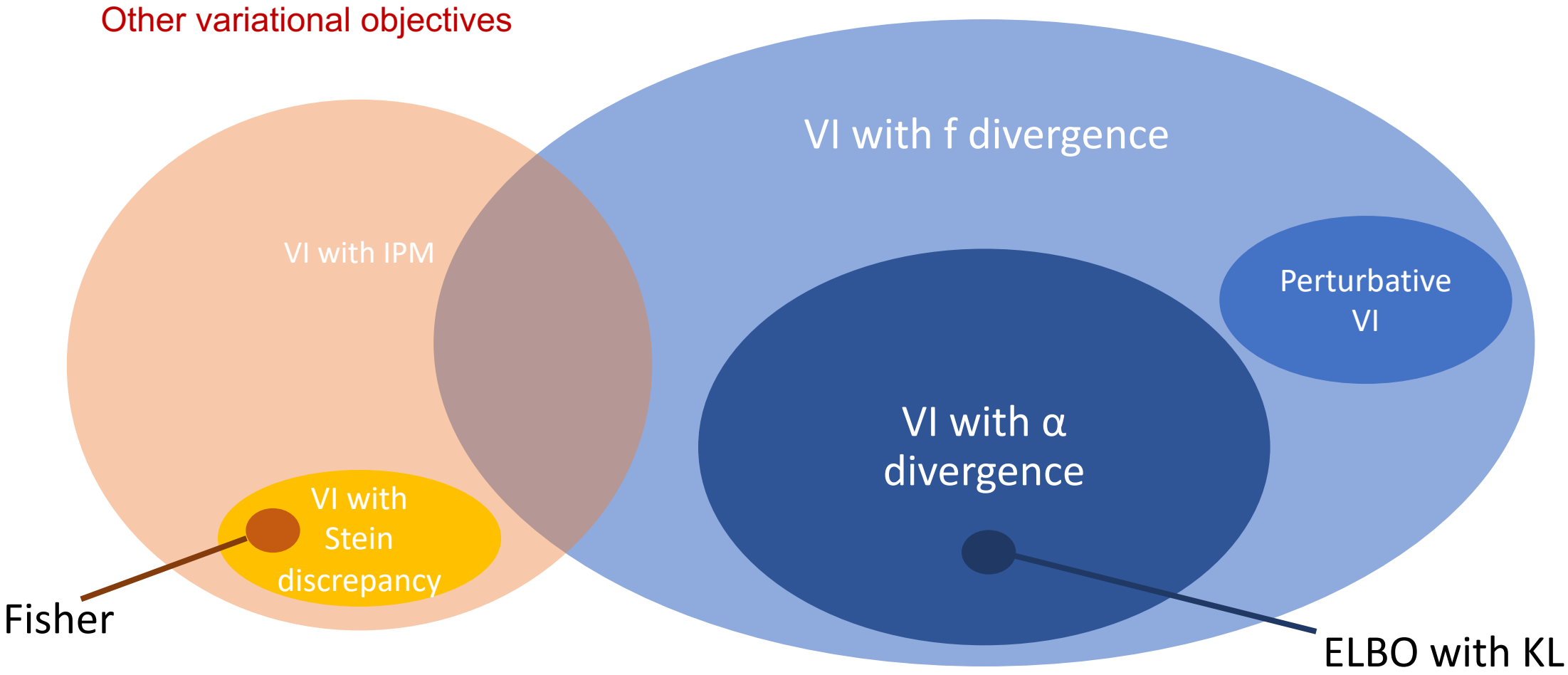
Implicit approximate posteriors



Particle-based posteriors

Mescheder et al. Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks. ICML 2017
Tran et al. Hierarchical Implicit Models and Likelihood-Free Variational Inference. NeurIPS 2017
Li and Turner. Gradient Estimators for Implicit Models. ICLR 2018
Yin and Zhou. Semi-Implicit Variational Inference. ICML 2018

Other Methods (Not Covered Today)



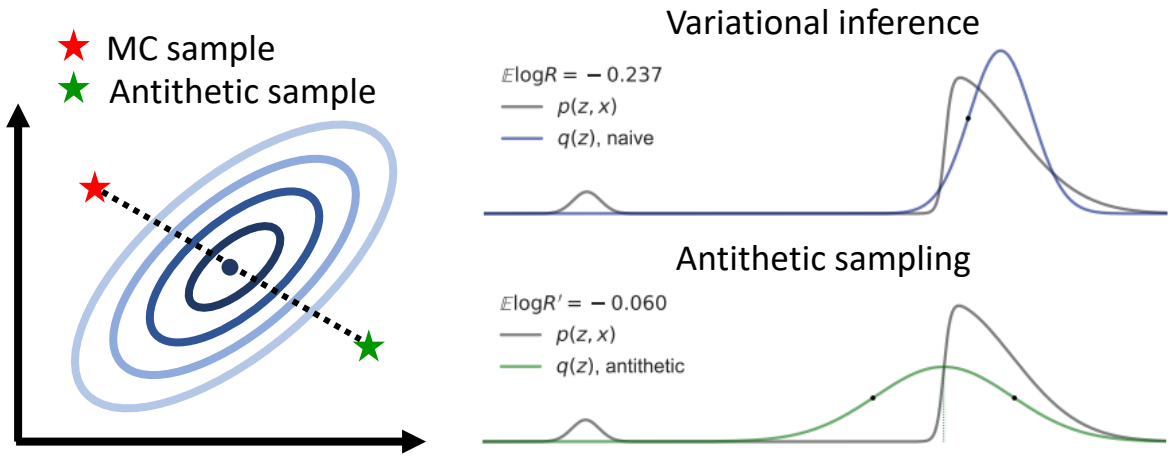
(VI with f-divergences, Wasserstein distance, Stein discrepancy, SVGD, etc… ask us for references)

Other Methods (Not Covered Today)

Deriving $\log Z$ estimates → an approximate inference method to obtain q

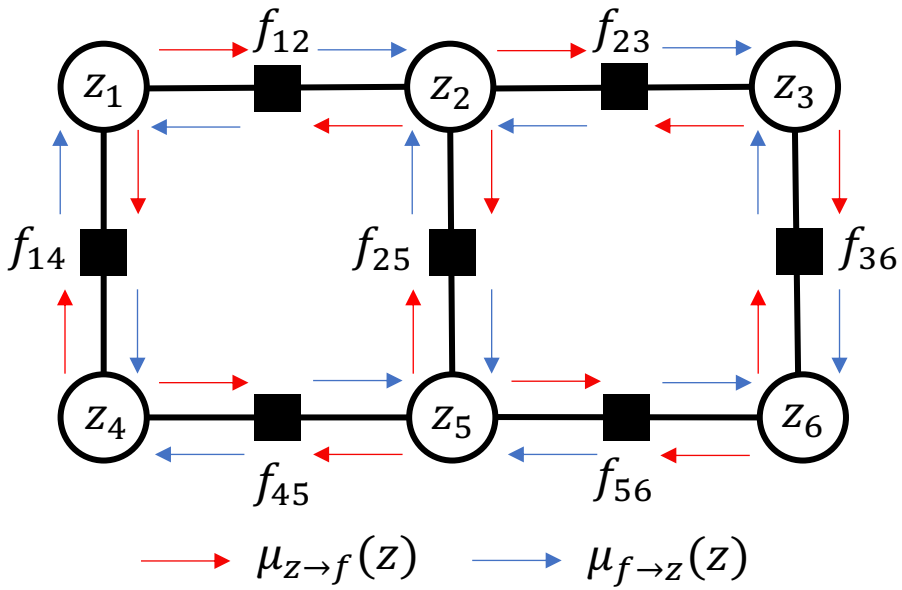
Monte Carlo estimators:

$$Z = E_{q(z)}[R(z)]$$
$$\Rightarrow \log Z = \log E_{q(z)}[R(z)] \geq E_{q(z)}[\log R(z)]$$



Free-energy approximation methods:

$$\log Z \approx L_{\text{Bethe}}[q(z)]$$

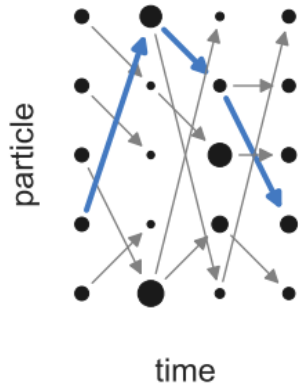


Domke and Sheldon. Divide and Couple: Using Monte Carlo Variational Objectives for Posterior Approximation. NeurIPS 2019

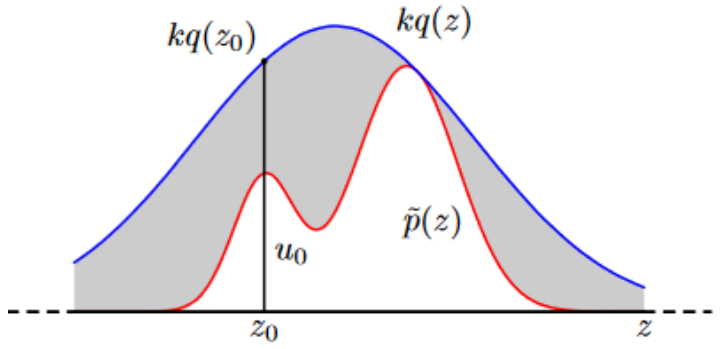
Wainwright and Jordan. Graphical Models, Exponential Families, and Variational Inference. FTML 2008.

Other Methods (Not Covered Today)

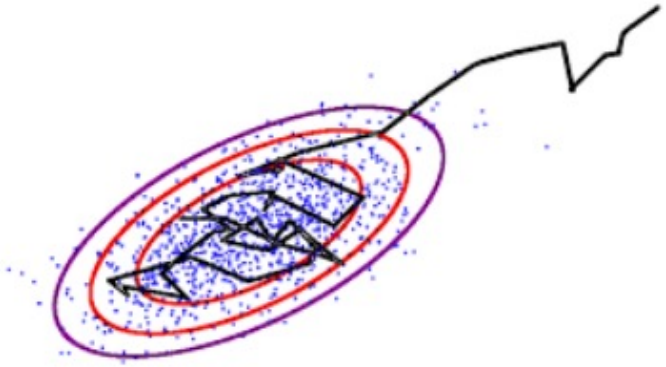
Combining approximate inference and sampling



Importance Sampling & SMC



Rejection Sampling & MH



MCMC (e.g., Langevin dynamics)

When needing efficient sampling for all $p(z|x), \forall x$:
find the optimal proposal distributions via amortized approximate inference

(Neural SMC, Amortised MCMC/IS estimation, Sliced Sampling, Meta learning for MCMC, etc… ask us for references)

Future Challenges & Opportunities

Score-based approximate inference approaches

- Methods for further advances:
  - more expressive families
  - scale to high-dimensions?
- Statistical properties:
  - estimation guarantees?
  - computational vs statistical tradeoffs? (consistency + convergence)
- Applications:
  - Amortized inference?
  - New insights for generative modeling?

Future Challenges & Opportunities

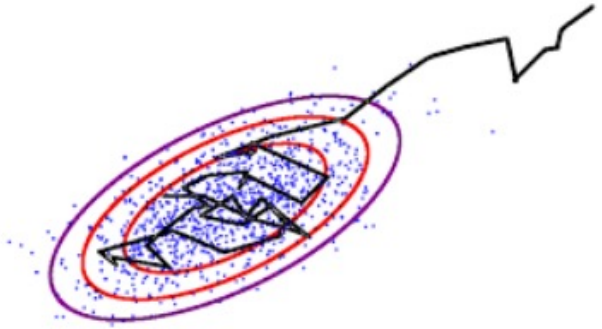
Diffusion & Flow based approaches

- Training difficulties: truly simulation-free approach?
  - Not truly simulation-free: requiring (importance) sampling from the density path $\{p_t(z)\}_{t\in[0,1]}$
    - Diffusion: estimating the score
    - CNF with PINN loss: constructing "training data" $q_t(z)$
    - Quality of IS/SMC is crucial to the empirical success!
  - Can we develop other training methods that are truly simulation-free?

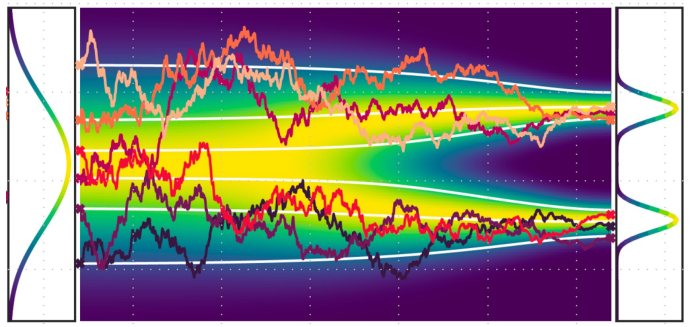
Future Challenges & Opportunities

Diffusion & Flow based approaches

- "Is it worth it?"
  - Sometimes your scientist friend only cares about **only one** (very difficult) potential function…
  - Maybe yes, in amortized setting: train $q(z|x) \approx p(z|x)$
  - vs (gradient-based) MCMC? Is it really better?
    - E.g. in training energy-based model with contrastive divergence?



vs



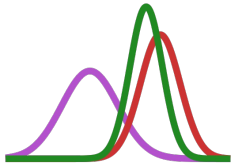
Future Challenges & Opportunities

Generally not solved yet:

- Better computation & implementation
  - Hardware-aware methods
    - E.g., GPU acceleration with CUDA and/or Triton, etc
  - Robust Implementation of advanced VI in statistics software
    - Flexible q distributions
    - Good optimization

Stan

NumPyro

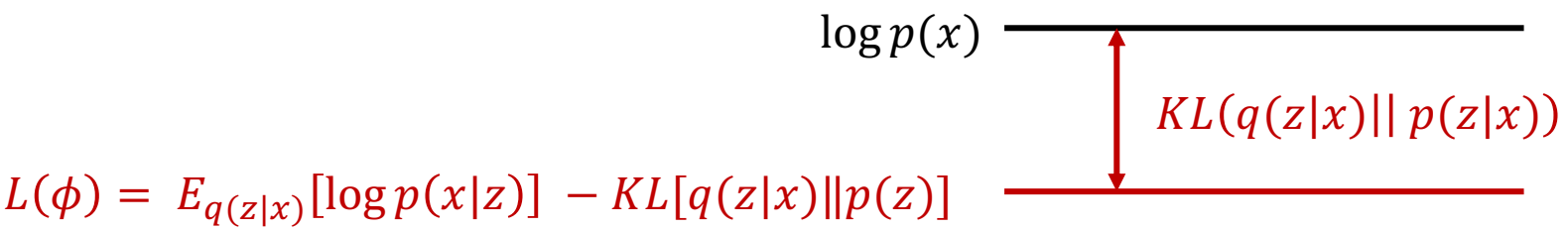


Turing.jl

Future Challenges & Opportunities

Generally not solved yet:

- Better statistical theory for approximate inference
  - Optimization & statistical properties
    - e.g. (non-)asymptotic behavior, estimation of moments
    - Convergence behavior in optimization?
  - VAE learning for model p: bias induced by sub-optimal Q family?



Future Challenges & Opportunities

Doing probabilistic inference for/with LLMs?

- Speeding-up LLM decoding?
  - E.g., speculative decoding as a “weird way” to do rejection sampling
- LLM “reasoning”?
  - RL-based finetuning: connection with VI/SMC in SSMs
  - “Steered sampling”: connections with IS/SMC/MCMC
- Using LLMs to build informative priors?
  - Posterior inference still requires approximations
- LLM “in-context learning” as Bayesian inference?
  - Prediction-centric formulation, still requires approximations

Gemini

deepseek

Claude

LLaMA

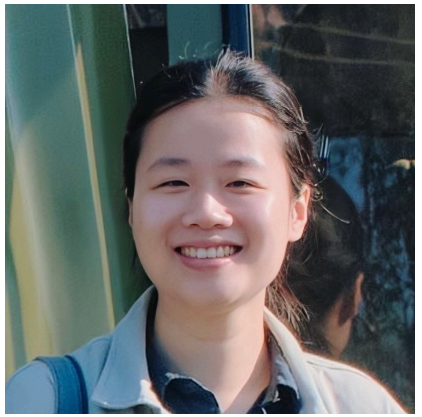
(too many references due to LLM community efforts, ask us if interested)

Thank You!

Questions? Ask now or contact us via email :)

Diana Cai

dcai@flatironinstitute.org



Yingzhen Li

yingzhen.li@imperial.ac.uk