Normalizing Flows for Molecular Modeling

José Miguel Hernández–Lobato
Department of Engineering
University of Cambridge

http://jmhl.org, jmh233@cam.ac.uk

Introduction and motivation

- The probability of 3D atom locations in a molecule is given by the **Boltzmann distribution**:

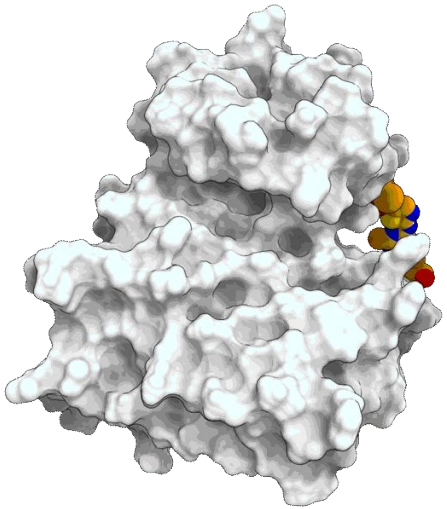
  $$p(x) \propto \{ -E(x) / (kT) \},$$

  where E(x) is the energy, k and T are constants.

- It allows us to answer important questions:

  Does a drug molecule bind to a target protein?

  Many other applications as well!



Animation source: D. E. Shaw Research.

Markov Chain Monte Carlo (MCMC)

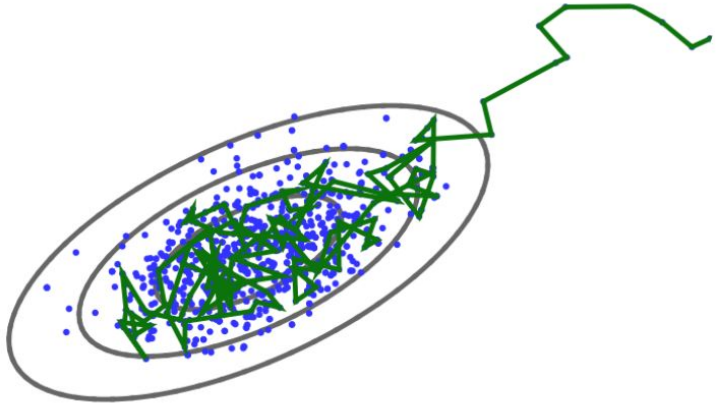
Based on slides by I. Murray

Main idea: construct a biased random walk that explores a target distribution $p_\star(\mathbf{x})$ whose normalization constant may not be known.

The random walk **transition operator** follows the Markov assumption:

$$\mathbf{x}_t \sim T(\mathbf{x}_t|\mathbf{x}_{t-1})\,.$$

The stationary distribution of $\{\mathbf{x}_t\}$ will be $p_\star(\mathbf{x})$:



$\{\mathbf{x}_t\}$ are approximate, correlated samples from $p_\star(\mathbf{x})$.

Detailed balance

Means that transitions $a \to b$ and $b \to a$ are equally probable in the chain:

$$T(\mathbf{x}'|\mathbf{x})p_\star(\mathbf{x}) = T(\mathbf{x}|\mathbf{x}')p_\star(\mathbf{x}')\,. \tag{1}$$

Detailed balance implies that the invariant distribution is $p_\star(\mathbf{x}')$:

$$\sum_{\mathbf{x}} T(\mathbf{x}'|\mathbf{x})p_\star(\mathbf{x}) = p_\star(\mathbf{x}') \sum_{\mathbf{x}} T(\mathbf{x}|\mathbf{x}') = p_\star(\mathbf{x}')\,.$$

$\{\mathbf{x}\}$ satisfies detailed balanced $\Leftrightarrow$ $\{\mathbf{x}\}$ is reversible, that is, $x_1, \ldots, x_N$ and $x_N, \ldots, x_1$ have the same probability distribution:

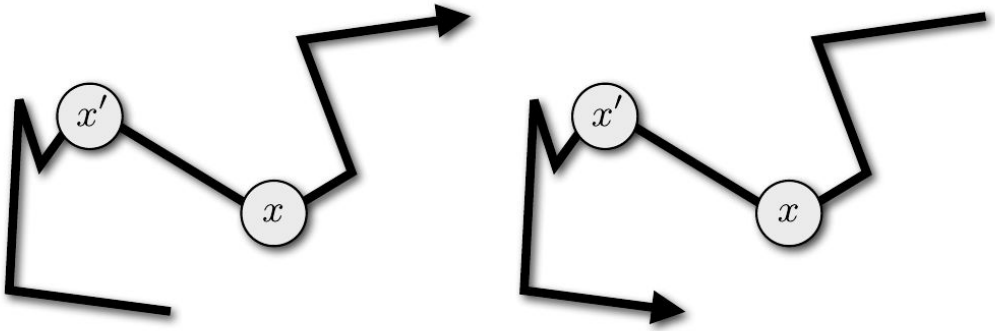


Figure source: Ryan P. Adams

To construct a chain that samples from $p_\star(\mathbf{x}')$, just find $T(\mathbf{x}'|\mathbf{x})$ satisfying (1).

Metropolis-Hastings

Based on slides by I. Murray

One of the algorithms with highest influence in science and engineering!

Works by sampling from the **transition operator** given by

1 - Draw a *proposal* from an *easy* distribution $q(\mathbf{x}'|\mathbf{x})$, e.g., $\mathcal{N}(\mathbf{x}'|\mathbf{x}, \sigma\mathbf{I})$.

2 - Accept with probability $\min\left(1, \frac{p_\star(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')}{p_\star(\mathbf{x})q(\mathbf{x}'|\mathbf{x})}\right)$.

3 - Otherwise the next state $\mathbf{x}'$ in chain is a copy of current state $\mathbf{x}$.

Acceptance ratio does not change if $p_\star(\mathbf{x})$ **is not normalized**.

The MH transition operator can be shown to satisfy **detailed balance**:

$$p_\star(\mathbf{x})T(\mathbf{x}'|\mathbf{x}) = p_\star(\mathbf{x})q(\mathbf{x}'|\mathbf{x})\min\left(1, \frac{p_\star(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')}{p_\star(\mathbf{x})q(\mathbf{x}'|\mathbf{x})}\right) = \min\left(p_\star(\mathbf{x})q(\mathbf{x}'|\mathbf{x}), p_\star(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')\right)$$

$$= p_\star(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')\min\left(\frac{p_\star(\mathbf{x})q(\mathbf{x}'|\mathbf{x})}{p_\star(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')}, 1\right) = p_\star(\mathbf{x}')T(\mathbf{x}|\mathbf{x}')$$

Example

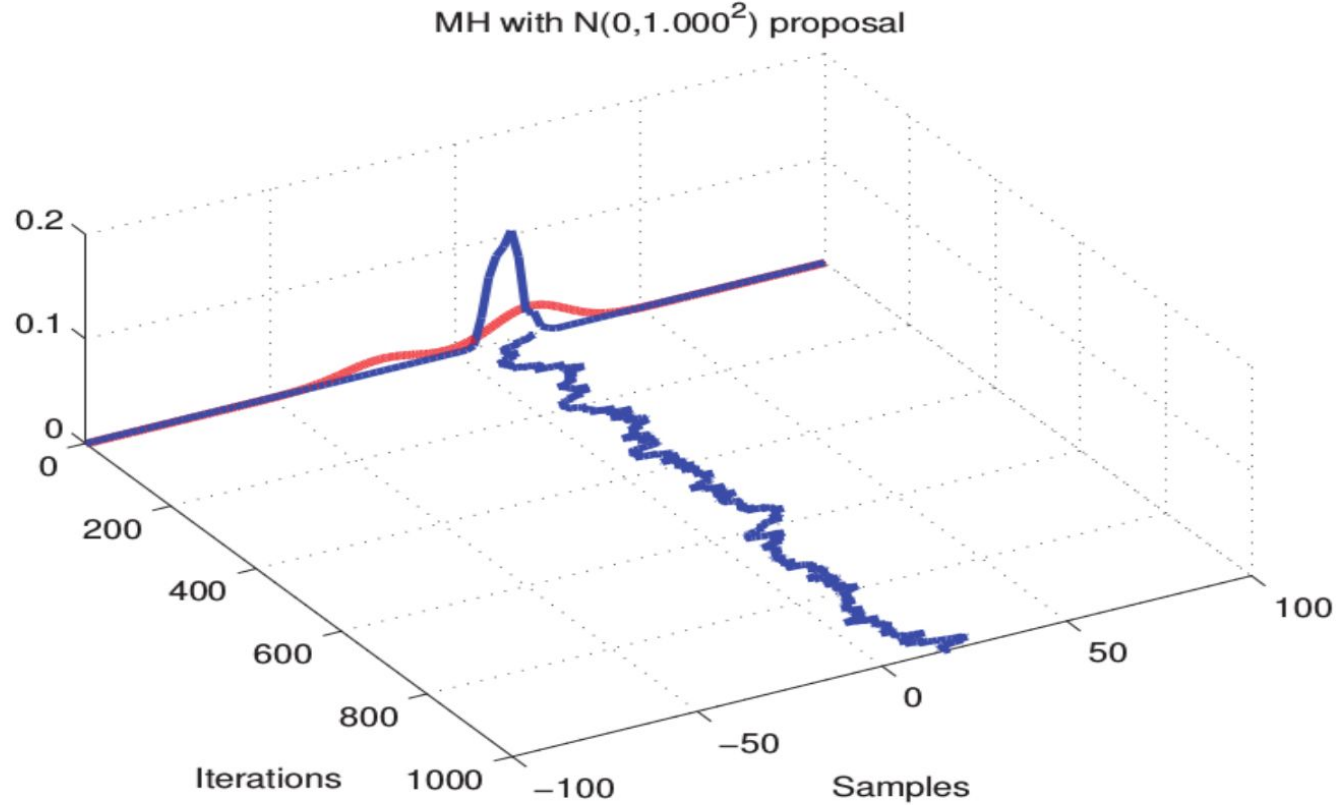


Figure source: K. Murphy

Example

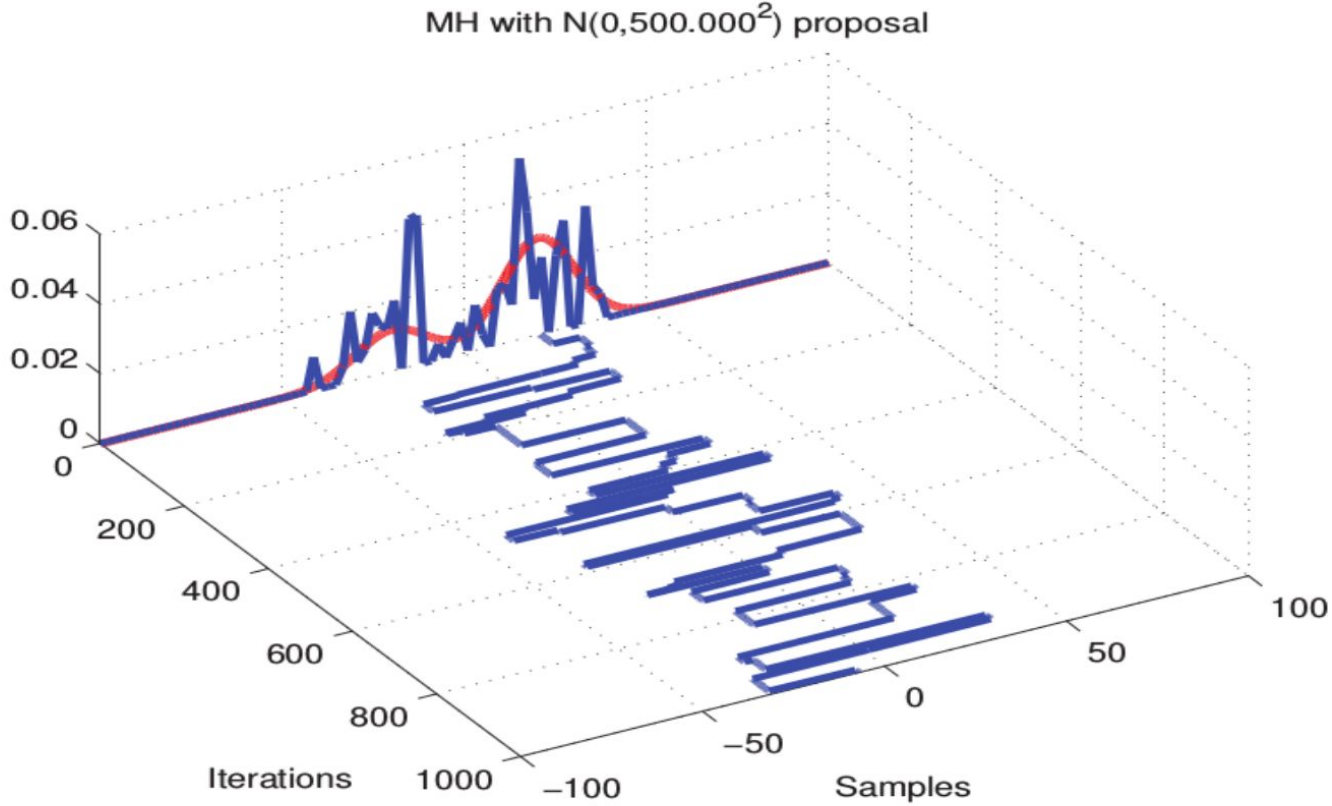


Figure source: K. Murphy

Example

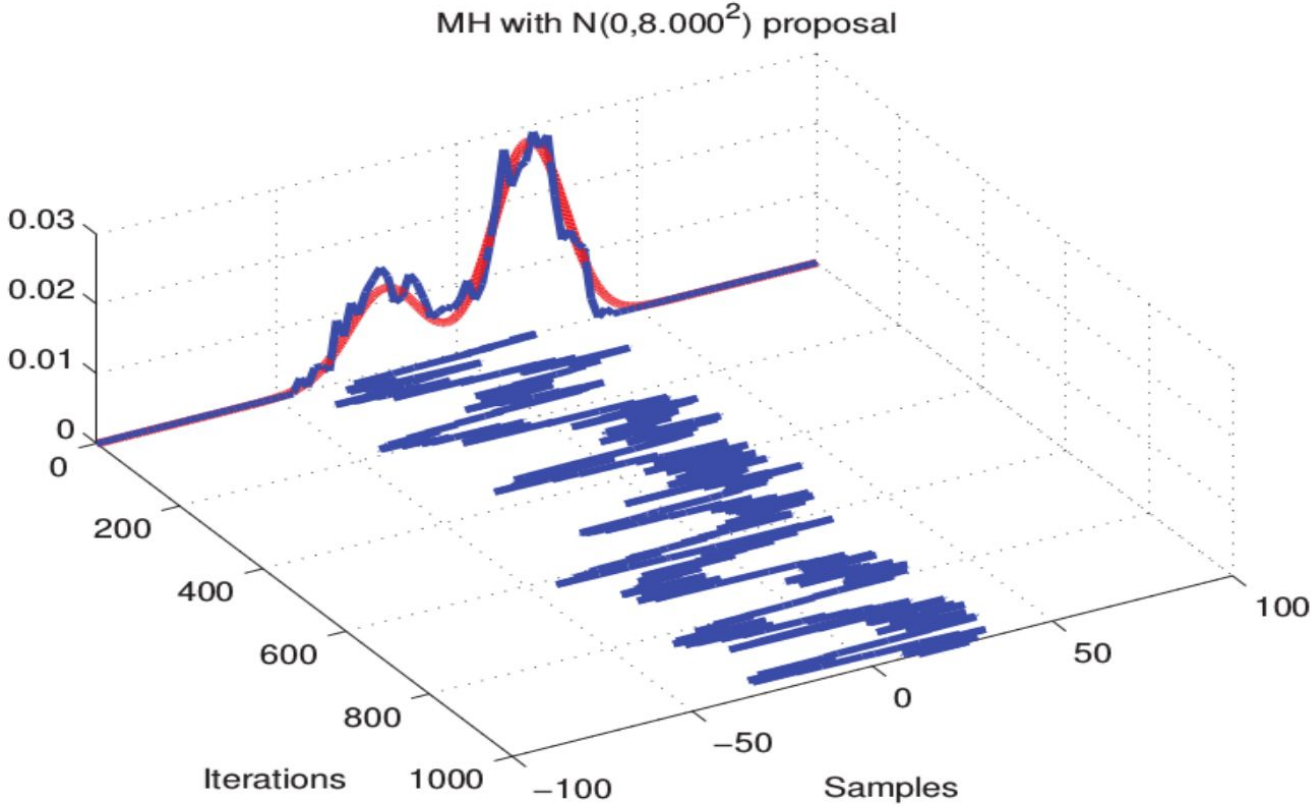
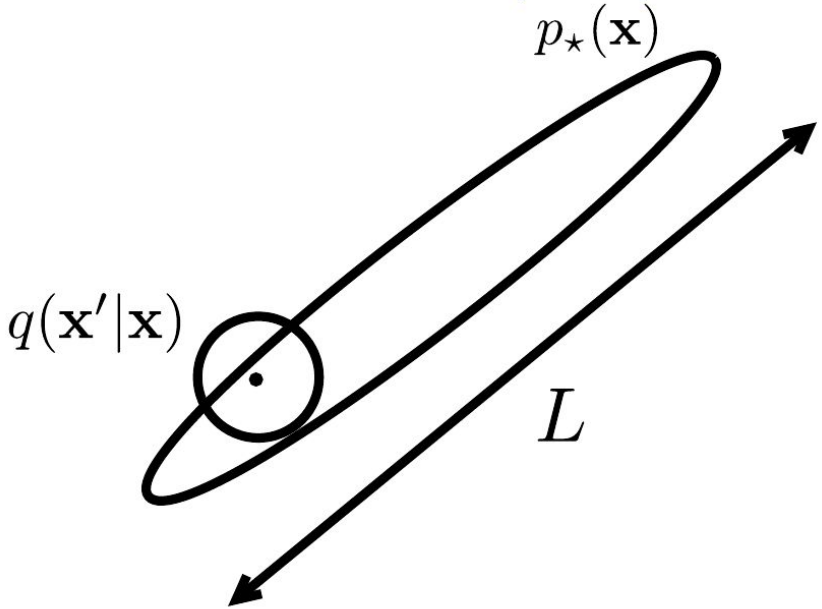


Figure source: K. Murphy

Limitations of Metropolis-Hastings (MH)

Based on slides by I. Murray



- Typically, $q(\mathbf{x}'|\mathbf{x}) = \mathcal{N}(\mathbf{x}'|\mathbf{x}, \sigma\mathbf{I})$ and proposals follow a **random walk**.
- If σ is large, we reject a lot!
- If σ is small, the chain diffuses very slowly: $\approx L^2/\sigma^2$ steps required to obtain independent samples.

Figure source: Ian Murray.

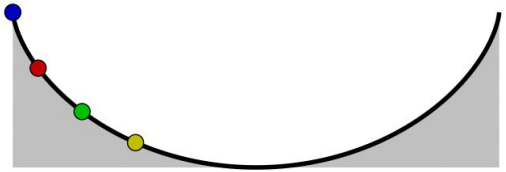
Hamiltonian dynamics

Introduce velocity v carrying **kinetic energy** $K(v) = \frac{1}{2} v^{\mathrm{T}} v$

Some physics:

- Total energy or **Hamiltonian**: $H(x,v) = E(x) + K(v)$
- Frictionless ball rolling $(x,\ v) \rightarrow (x',\ v')$ satisfies $H(x,v) = H(x',\ v')$

$$\frac{dx}{dt} = \frac{\partial H(x,v)}{\partial v} \quad \text{and} \quad \frac{dv}{dt} = -\frac{\partial H(x,v)}{\partial x}$$



- Ideal Hamiltonian dynamics are time reversible:
  — reverse v and the ball will return to its start point

Hamiltonian Monte Carlo

Define a joint distribution as

$$p(x, v) \propto e^{-E(x)} e^{-K(v)} = e^{-E(x)-K(v)} = e^{-H(x,v)}$$

where velocity v variables are independent and Gaussian distributed.

Markov Chain Transition Operator

1. Sample velocity v from its marginal
2. **Simulate Hamiltonian dynamics** then flip sign of velocity

   — MH proposal q is deterministic and reversible $q(x', v' \mid x, v) = q(x, v \mid x', v')$

   — Conservation of energy means $p(x, v) = p(x', v')$

   — MH acceptance probability is 1

Leap-frog dynamics

A discrete approximation to Hamiltonian dynamics:

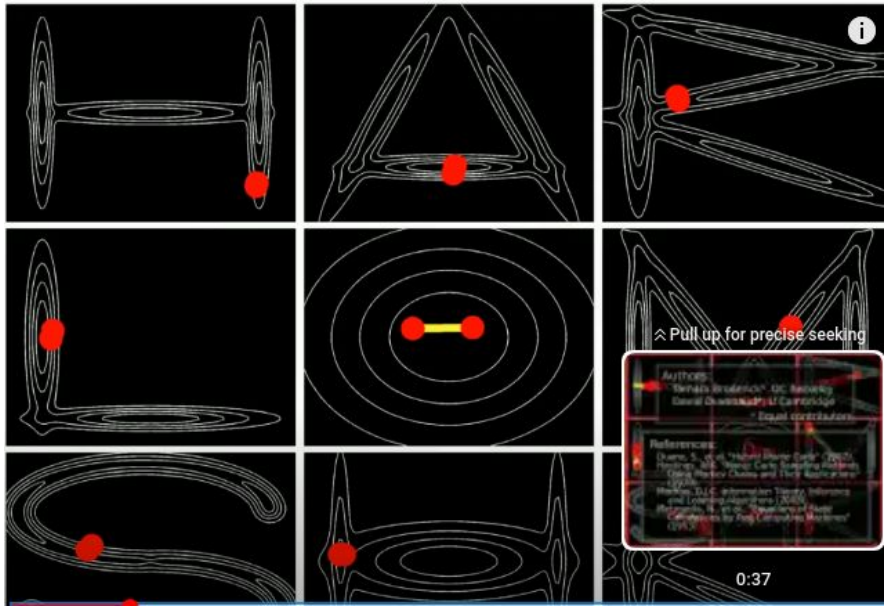
$$v_i(t + \epsilon/2) = v_i(t) - \frac{\epsilon}{2} \frac{\partial E}{\partial x_i}(x(t))$$

$$x_i(t + \epsilon) = x_i(t) + \epsilon v_i(t + \epsilon/2)$$

$$v_i(t) = v_i(t + \epsilon/2) - \frac{\epsilon}{2} \frac{\partial E}{\partial x_i}(x(t) + \epsilon)$$

- H is not conserved
- dynamics are still deterministic and reversible
- Acceptance probability becomes $\min[1, exp\{H(v', x') - H(v, x)\}]$

Hamiltonian Monte Carlo vs Metropolis Hastings



https://www.youtube.com/watch?v=Vv3f0QNWvWQ

Limitations of MCMC methods

- Often stuck into a few modes in multimodal distributions
- Generated samples are **correlated** and **not independent**!
- Samples from the past are not used to improve the generation of new samples
- Small step sizes needed to guarantee acceptance
- Very slow in practice!

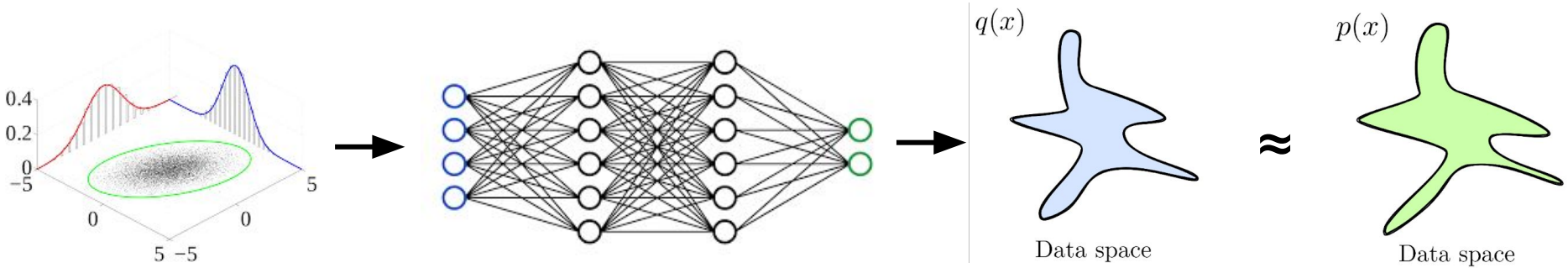
  — E.g. Sampling a molecular events may take years on a supercomputer

Alternative: use deep generative models!

They are **neural networks** that transform **tractable random variables** into **complicated ones**.

Their goal is to **approximate a complicated target distribution** (often the data distribution).

Tractable noise → **Neural network** → **Approximate distribution** ≈ **Target distribution**



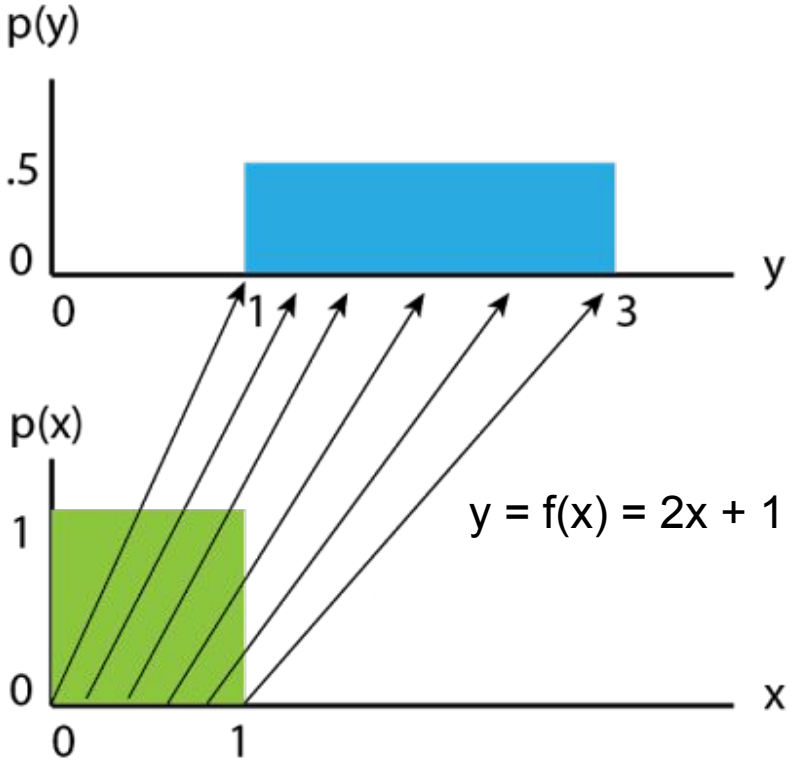
Once trained, they can often do tractable **independent data sampling** and **density evaluation**.

Noé, Frank, et al. "Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning." *Science* 365.6457 (2019).

Invertible transformations of random variables

Probability mass is preserved

$$p(x)\,dx = p(y)\,dy$$



Slide source Eric Jang

Density of the transformed variables

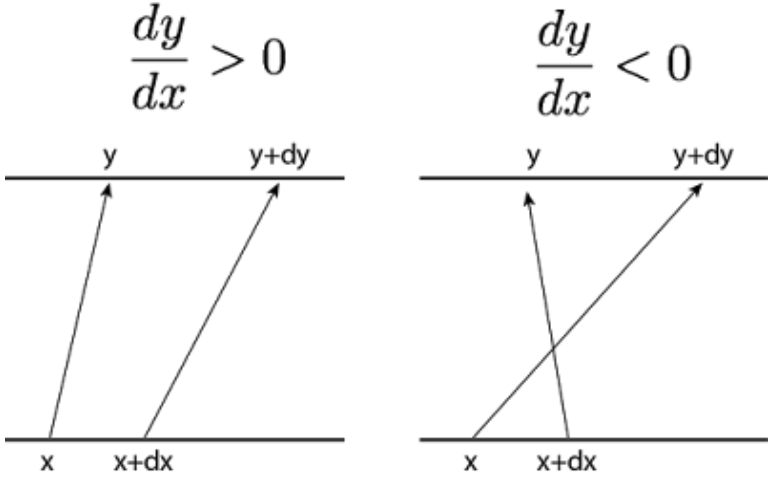
p(y) dy = p(x) dx

p(y) = p(x) |dx / dy|

p(y) = p(x) / |dy / dx|

log p(y) = log p(x) - log |dy / dx|

Why use absolute values ?



Slide source Eric Jang

Higher dimensions

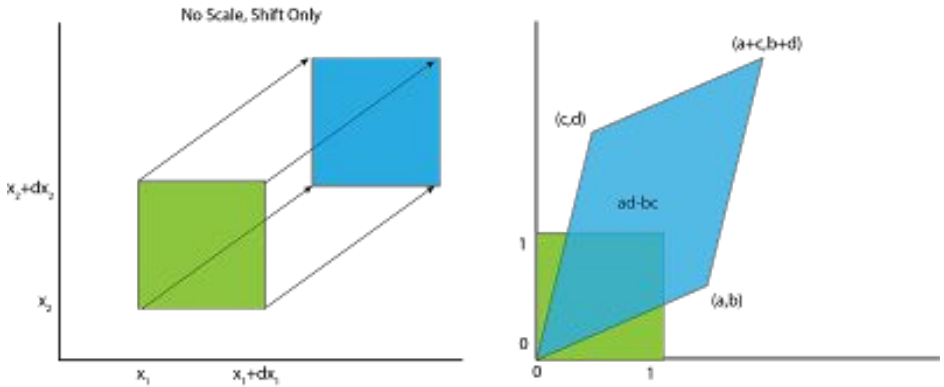
In higher dimensions, change in volume is the determinant of the Jacobian of **f**(**x**).

J(f)(x) $= \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$

log p(y) = log p(x) - log |det J(f)(x)|

Example:

Sending a unit square (area = 1) to some parallelogram (area = ad - bc)



Slide source Eric Jang

Main idea behind normalizing flows

- Use a base distribution p(x) for which log p(x) is tractable and sampling is easy.
- Learn invertible function y = f(x) with associated inverse x = f^{-1}(y).
- Model learned by maximum likelihood using log p(y) = log p(x) - log |det J(f)(x)|.
- Key challenge: **keep computation of f, f^{-1} and log |det J(f)(x)| tractable**.
- Often **f** obtained by combining **multiple transformations or layers**:

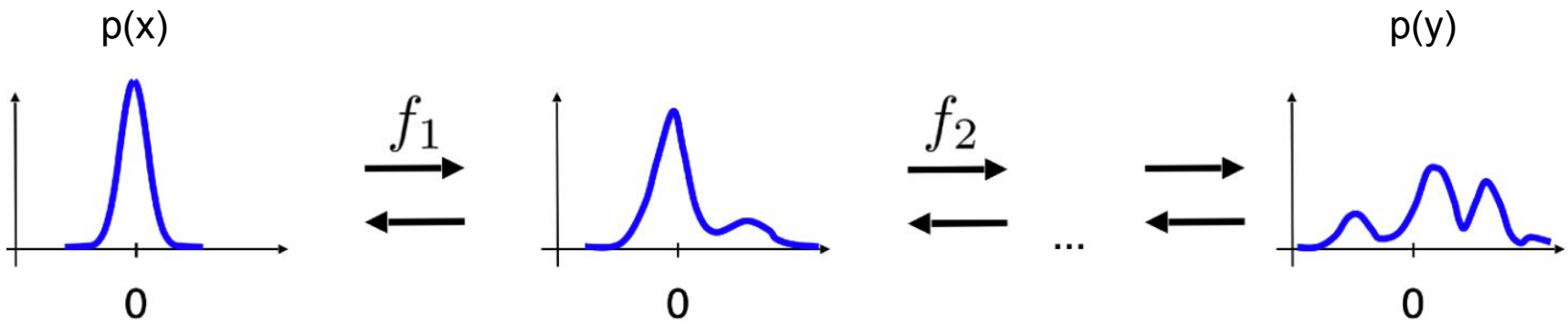


Figure source: Jakub M. Tomczak.

How to formulate invertible layers?

1) Coupling layers

$$\begin{aligned} \mathbf{y}_a &= \mathbf{x}_a \\ \mathbf{y}_b &= \exp\left(s\left(\mathbf{x}_a\right)\right) \odot \mathbf{x}_b + t\left(\mathbf{x}_a\right) \end{aligned} \quad \leftrightarrow \quad \begin{aligned} \mathbf{x}_b &= \left(\mathbf{y}_b - t(\mathbf{y}_a)\right) \odot \exp\left(-s(\mathbf{y}_a)\right) \\ \mathbf{x}_a &= \mathbf{y}_a \end{aligned}$$

Jacobian is tractable!

$$\det(\mathbf{J}) = \prod_{j=1}^{D-d} \exp\left(s\left(\mathbf{x}_a\right)\right)_j = \exp\left(\sum_{j=1}^{D-d} s\left(\mathbf{x}_a\right)_j\right)$$

2) Permutation layers

$$\det(\mathbf{J}) = 1$$

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. ICLR, 2017

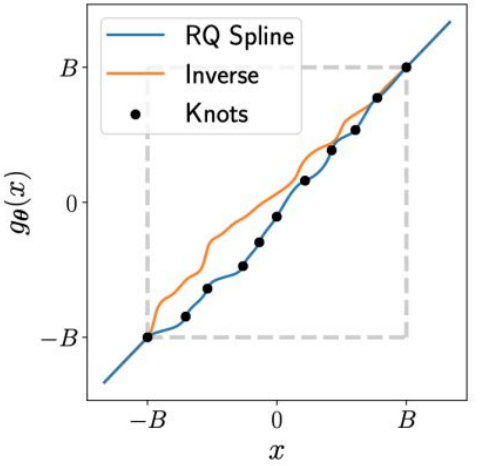
Slide source: Jakub M. Tomczak.

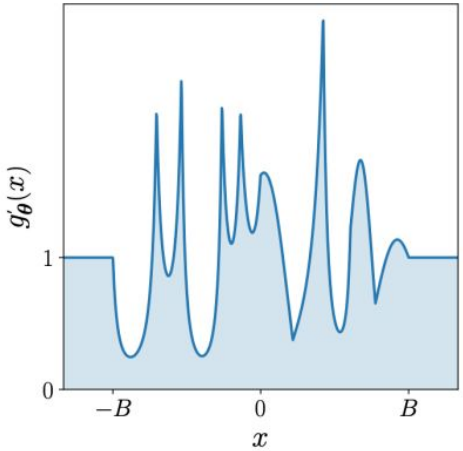
Example

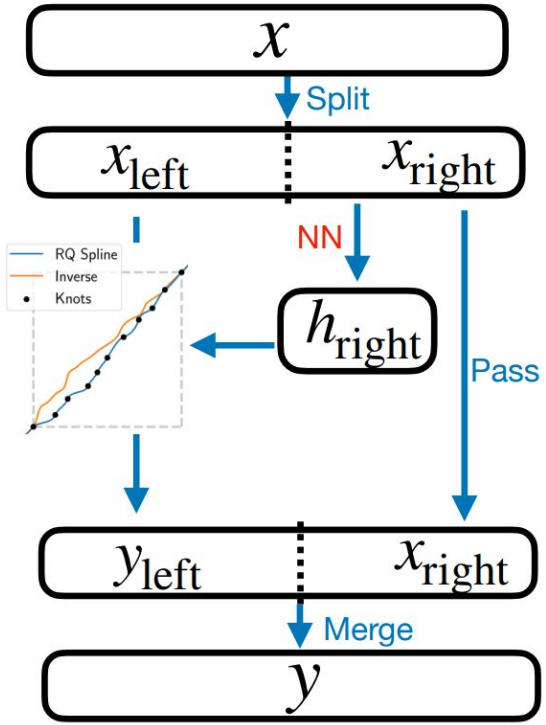
Kingma, D.P., and Prafulla D. "Glow: Generative flow with invertible 1x1 convolutions." NeurIPS, 2018

Neural spline flows

- Uses concept of coupling layer
- But transform given by monotonic rational quadratic splines
- Multimodal transforms
- Analytic inverse
- Applicable to compact intervals and circular domains







Durkan et al. Neural spline flows. NeurIPS, 2019.

Figure source: Jonas Köhler, Andreas Krämer and Frank Noé

Example

Neural spline flows with two coupling layers, each with K = 128 bins.

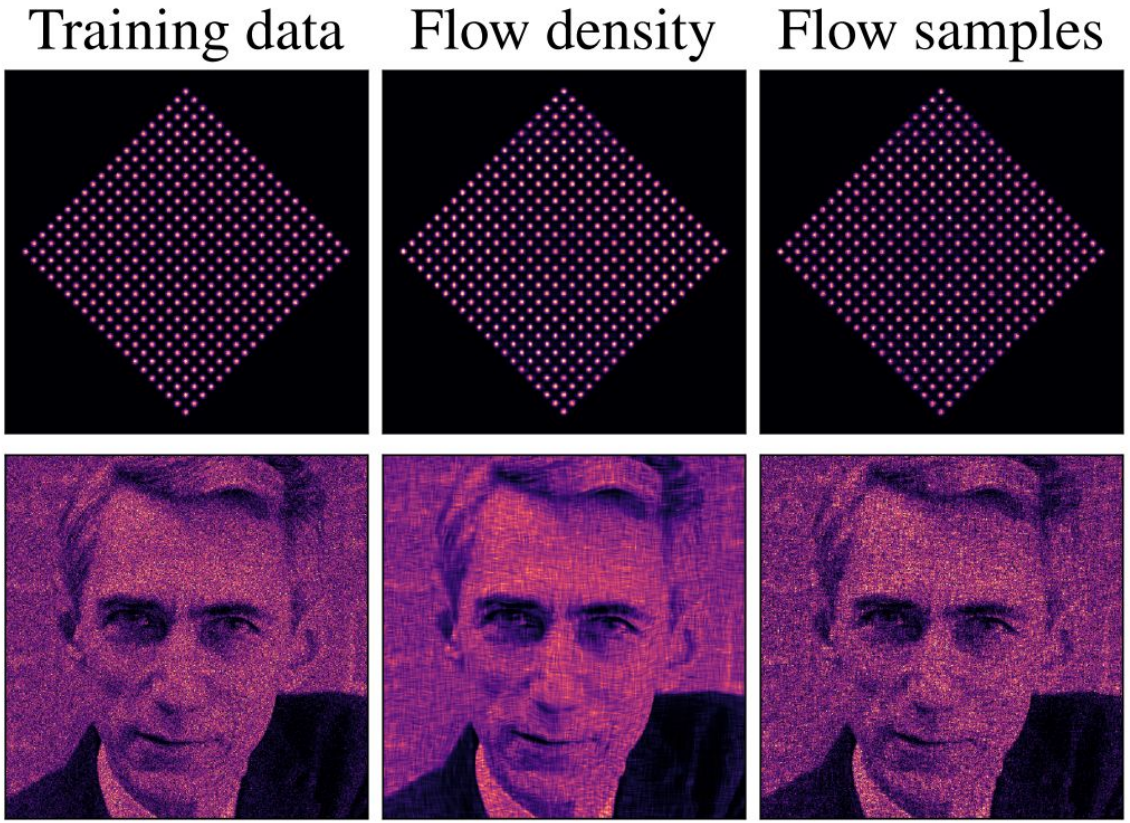


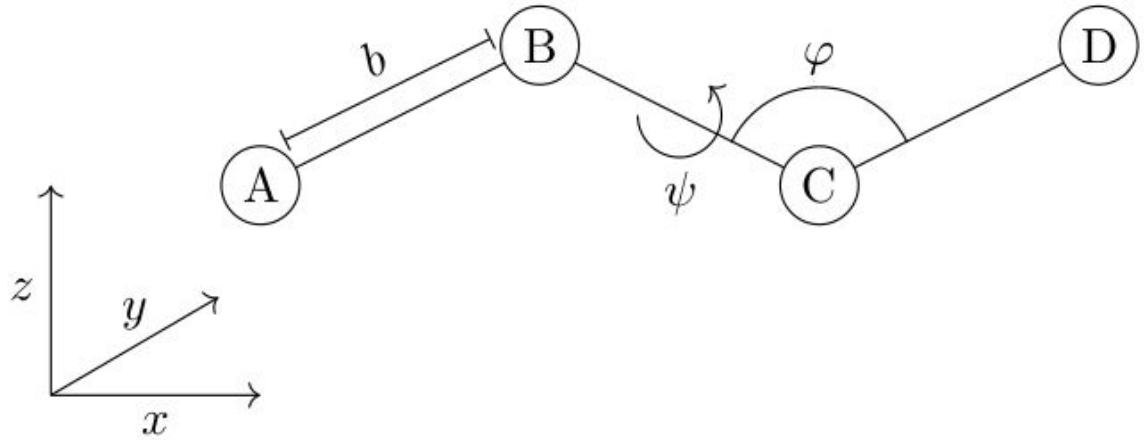
Figure source: Durkan et al. Neural spline flows. NeurIPS, 2019.

Internal coordinates for molecule representation

Molecules can be described via the spatial coordinates x, y, and z of each atom.

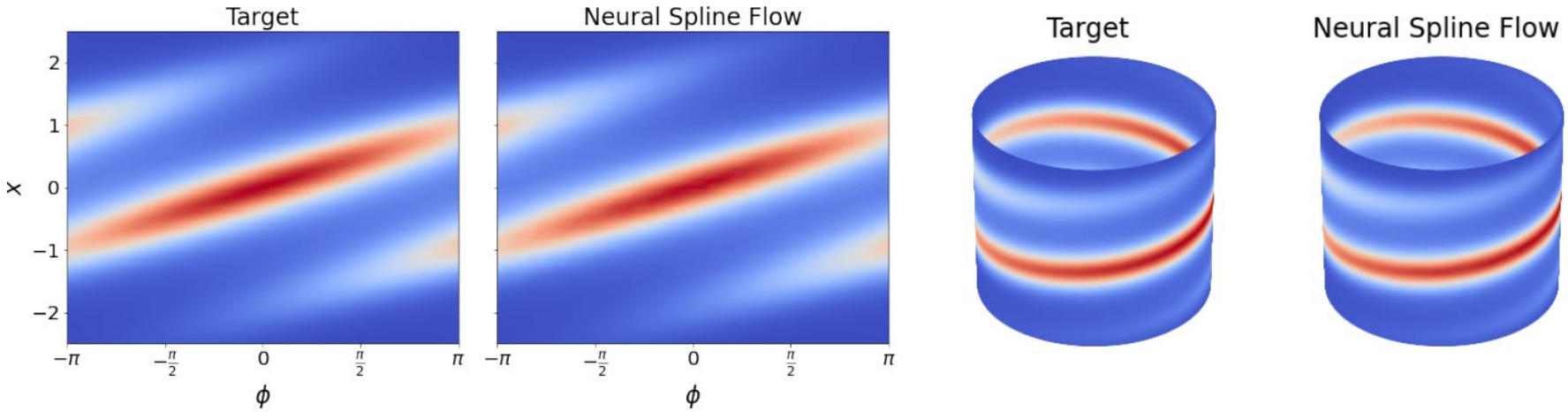
However, there are advantages in using **internal coordinates** such as

- **bond lengths**, e.g. b,
- **bond angles**, e.g. φ,
- **dihedral angles**, e.g. ψ,



which naturally capture **invariances to rotations and translations**.

Normalizing flows on circular coordinates



Neural spline flow with the following constraints:

- g(0) = 0, g(2π) = 2π, g'(x) > 0, g'(0) = g'(2π)
- Avoid 0 and 2π being fixed points by doing phase translation with parameter ϕ: x → x + ϕ mod 2π

Rezende et al. Normalizing flows on circles, tori and spheres. In ICML, 2020.

Figure source: Stimper et al., normflows: A PyTorch Package for Normalizing Flows, arXiv:2302.12014, 2023

Eliminate flow sampling bias with importance sampling

Write instead the integral as an **expectation under** $q(x)$:

$$\int f(x)p(x)\,dx = \int f(x)\frac{p(x)}{q(x)}q(x)\,dx, \qquad q(x) > 0 \;\; if \;\; p(x) > 0$$

$$\approx \frac{1}{N}\sum_{n=1}^{N} f(x_n)\underbrace{\frac{p(x_n)}{q(x_n)}}_{w_n} = \frac{1}{N}\sum_{n=1}^{N} f(x_n)w_n\,, \qquad x_i \sim q(x)\,.$$

The w_n are known as **importance weights**.

Can be applied **even if the integral is not an expectation**.

Given $p(x)$, what is the best sampling proposal q?

Effective sample size of importance samples

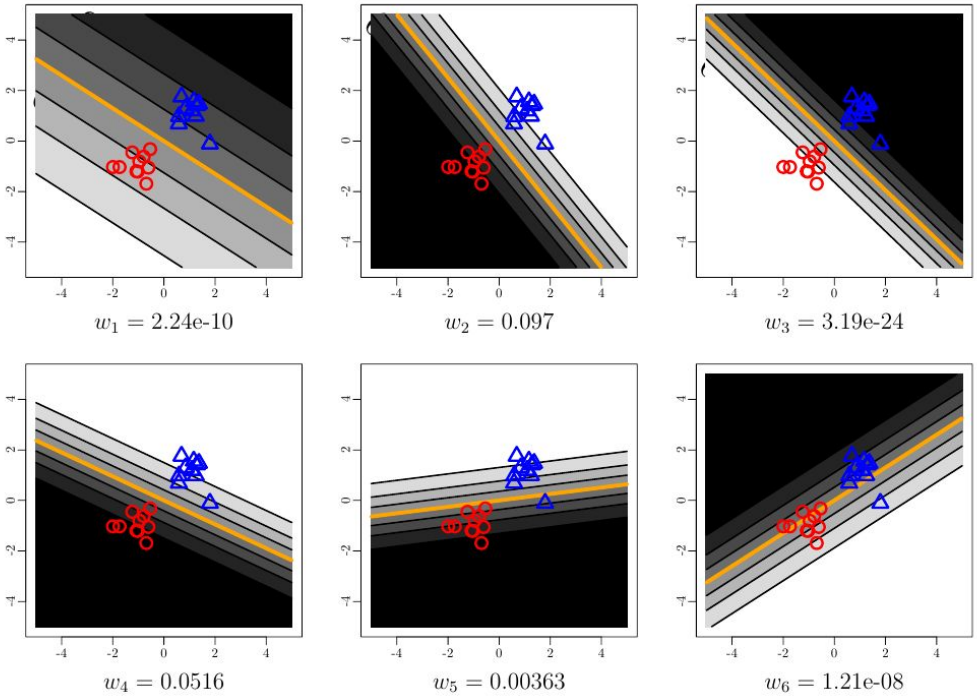
Logistic regression example:

q equal to Gaussian prior.

Weights can exhibit high variance, reducing the **effective sample size (ESS)** of the generated samples. Let us define $\tilde{w}_i = w_i/(\sum_{i=1}^n w_i)$ then

$$\mathrm{ESS} = \frac{1}{\sum_{i=1}^n \tilde{w}_i^2} = \frac{(\sum_{i=1}^n w_i)^2}{\sum_{i=1}^n w_i^2}.$$

ESS is n when $\tilde{w}_i$ = 1 / n and 1 if only one w_i takes a much larger value.



Many samples do not contribute to the expectation!

Annealed importance sampling

Given a target density p, a proposal density q and a sequence $0 = \beta_0 \leq \cdots \leq \beta_K = 1$, define

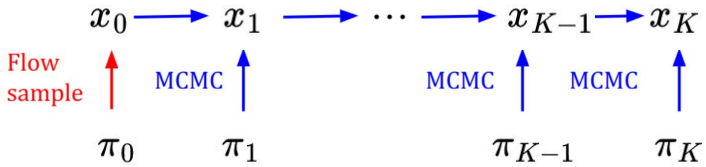
$$\pi_k(x) = p(x)^{\beta_n} q(x)^{(1-\beta_k)},$$

and let $T_1(x, x'), \ldots, T_K(x, x')$ be a sequence of transition kernels such that T_k leaves π_k invariant. Annealed Impoprtance Sampling amounts to drawing $x_0 \sim \pi_0(x)$ followed by

$$x_k \sim T_k(x_{k-1}, x_k) \text{ for } k = 1, \ldots, K,$$

and return the sample x_K together with the importance weight

$$w = \frac{\pi_1(x_0)}{\pi_0(x_0)} \frac{\pi_2(x_1)}{\pi_1(x_1)} \cdots \frac{\pi_K(x_K)}{\pi_{K-1}(x_K)}$$



When

$$\beta_k - \beta_{k-1} = \frac{1}{K}$$

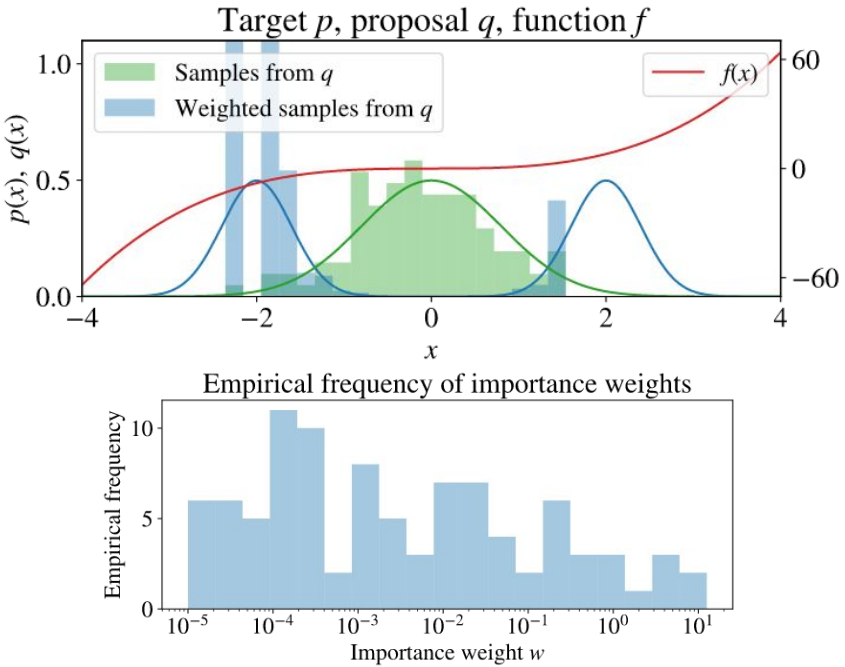
It is easy to show that

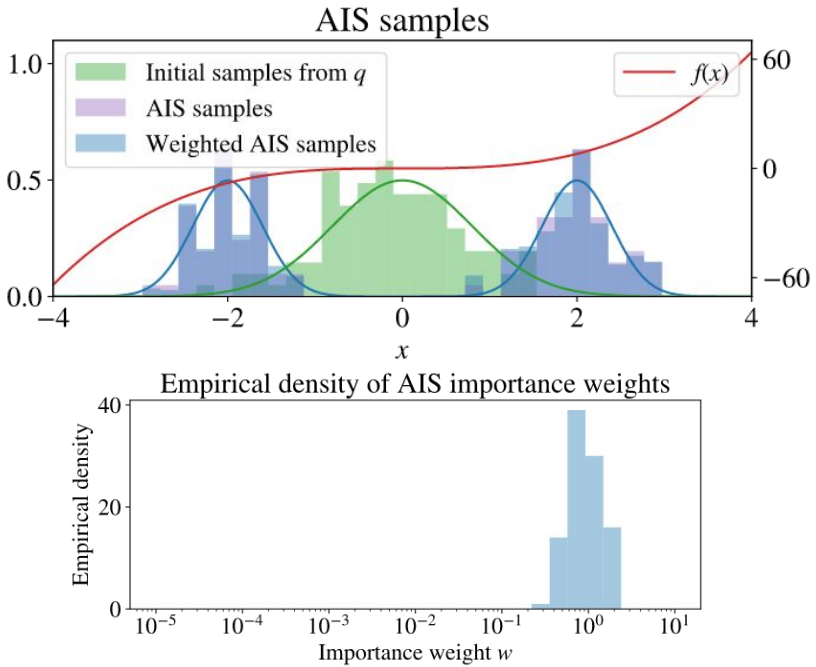
$$\log w = \frac{1}{K} \sum_{k=1}^{K} \{\log p(x_k) - \log q(x_k)\}$$

Under some assumptions, variance will be reduced asymptotically as 1 / K.

Neal, Radford M. "Annealed importance sampling." Statistics and computing 11 (2001): 125-139.

Example





IS estimate,100 samples: 0.10 ± 13.309 → Low ESS!

AIS estimate,100 samples: 0.30 ± 2.218 → High ESS!

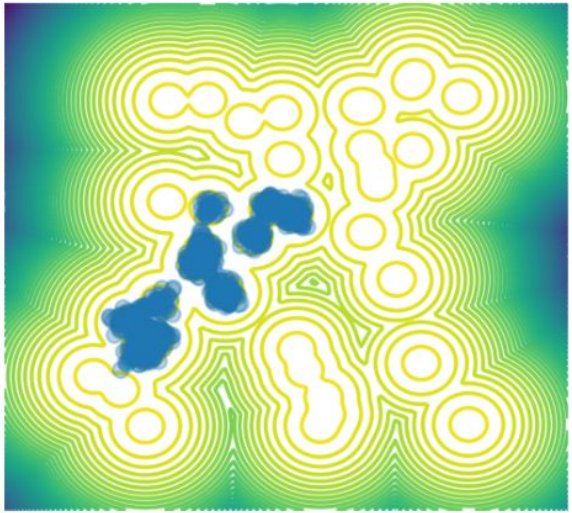
Slide source: https://random-walks.org/content/misc/ais/ais.html#id2

Challenges fitting flows to multimodal distributions from energy

Training flows from energy can be done by minimizing:

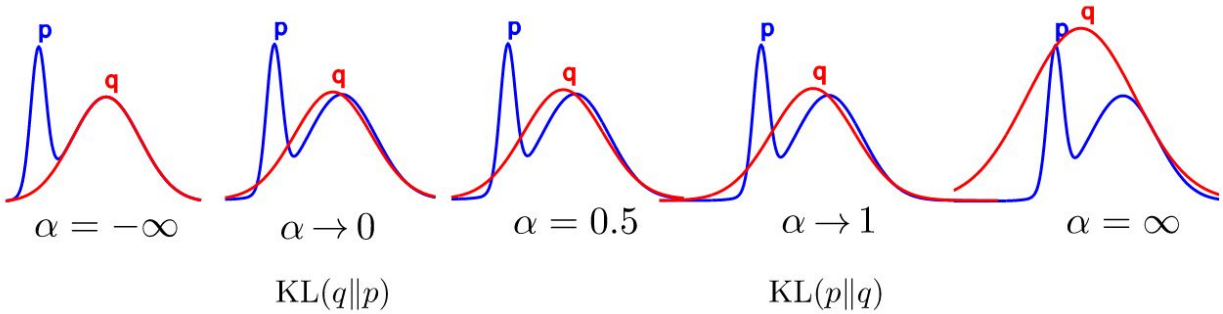
$$\mathrm{KL}(q||p) = \int q(x) \log \frac{q(x)}{p(x)} \mathrm{d}x$$

However, this objective results in **modes being missed**.



Alpha divergence

$$D_\alpha(p\|q) \propto \int \frac{p(x)^\alpha}{q(x)^\alpha} q(x)\, dx$$



- Mode-seeking when α <= 0
- Mass-covering when α >= 1
- When **α = 2**, it quantifies the **variance of importance sampling weights:**

$$D_{\alpha=2}(p\|q) \propto \int \frac{p(\mathbf{x})^2}{q(\mathbf{x})} d\mathbf{x} = \mathrm{E}_{q(\mathbf{x})}\left[w_{\mathrm{IS}}(\mathbf{x})^2\right]$$

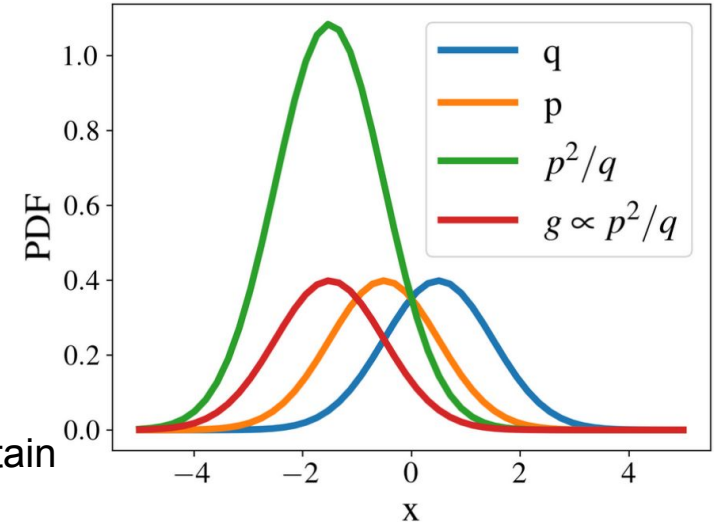
How to efficiently optimize the $\alpha = 2$ divergence?

Recall that $D_{\alpha=2}(p\|q) \propto \int \frac{p(x)^2}{q(x)} dx$.

The **optimal** IS distribution for estimating $D_{\alpha=2}(p\|q)$ is

$$g \propto \frac{p^2}{q}$$

We draw **samples from *g* using AIS.**



Assuming that $f_\theta(x) = p(x)^2/q_\theta(x)$ is normalized, we obtain

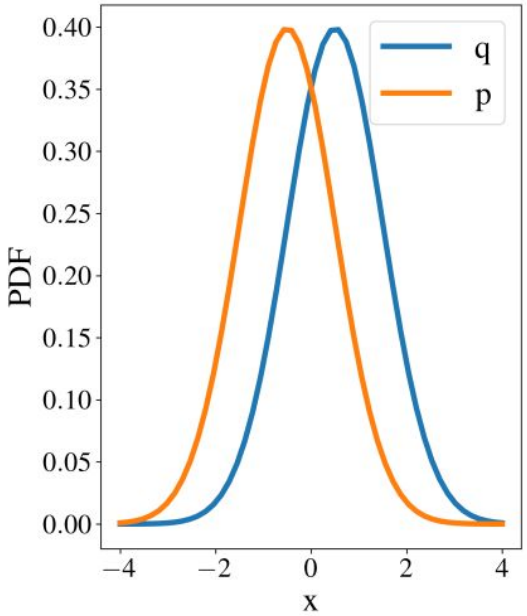
$$\nabla_\theta D_{\alpha=2}(p\|q) = \mathrm{E}_{g(\mathbf{x})}\left[\frac{\nabla_\theta f_\theta(\mathbf{x})}{f_\theta(\mathbf{x})}\right] = \mathrm{E}_{g(\mathbf{x})}\left[\nabla_\theta \log f_\theta(\mathbf{x})\right] =$$

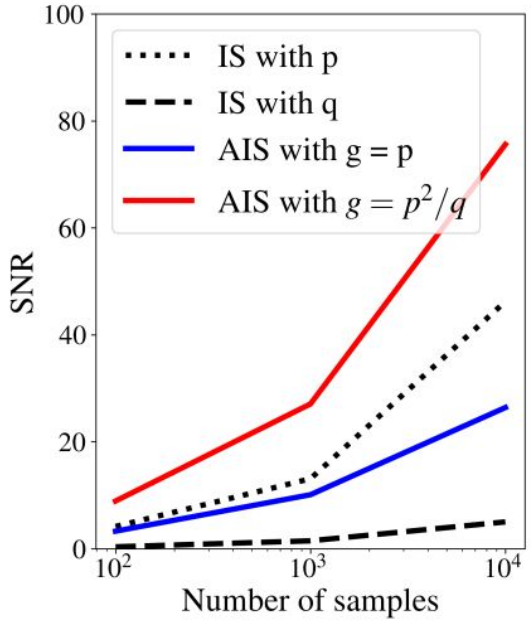
$$- \mathrm{E}_{g(\mathbf{x})}\left[\nabla_\theta \log q_\theta(\mathbf{x})\right] = - \mathrm{E}_{\mathrm{AIS}}\left[w_{\mathrm{AIS}} \nabla_\theta \log q_\theta(\bar{\mathbf{x}}_{\mathrm{AIS}})\right]$$

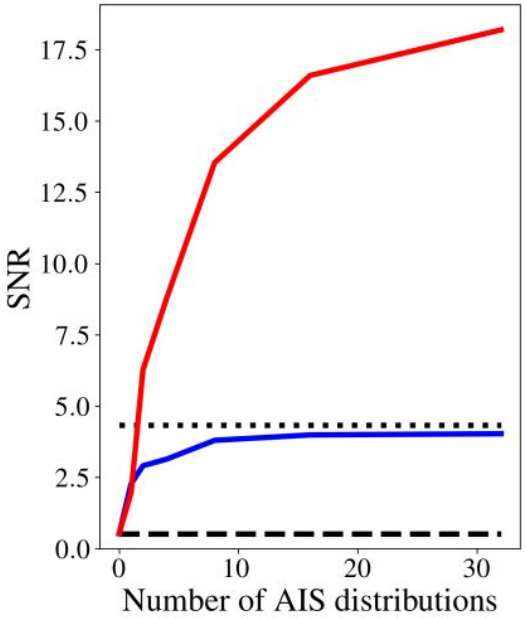
We call our method **Flow Annealed** Importance Sampling **Bootstrap**: **FAB**

Signal to noise ratio of gradient estimators

How well do we estimate $\nabla_\theta D_{\alpha=2}(p\|q) = \text{constant} \times \nabla_\theta \int \frac{p(x)^2}{q_\theta(x)}\, dx$ by Monte Carlo?







A replay buffer to reuse AIS samples in FAB

- Generate *M* samples with AIS and add them to buffer:
  a. Sample $\mathbf{x}_q^{(1:M)}$ from q_θ obtaining also $\log q_\theta(\mathbf{x}_q^{(1:M)})$.

A replay buffer to reuse AIS samples in FAB

- Generate *M* samples with AIS and add them to buffer:
  a. Sample $\mathbf{x}_q^{(1:M)}$ from q_θ obtaining also $\log q_\theta(\mathbf{x}_q^{(1:M)})$.
  b. Obtain $\mathbf{x}_{\text{AIS}}^{(1:M)}$ and $\log w_{\text{AIS}}^{(1:M)}$ by running AIS with input $\mathbf{x}_q^{(1:M)}$ and $\log q_\theta(\mathbf{x}_q^{(1:M)})$.

A replay buffer to reuse AIS samples in FAB

- Generate *M* samples with AIS and add them to buffer:
  a. Sample $\mathbf{x}_q^{(1:M)}$ from q_θ obtaining also $\log q_\theta(\mathbf{x}_q^{(1:M)})$.
  b. Obtain $\mathbf{x}_{\text{AIS}}^{(1:M)}$ and $\log w_{\text{AIS}}^{(1:M)}$ by running AIS with input $\mathbf{x}_q^{(1:M)}$ and $\log q_\theta(\mathbf{x}_q^{(1:M)})$.
  c. Add $\mathbf{x}_{\text{AIS}}^{(1:M)}$ to buffer, together with their $\log w_{\text{AIS}}^{(1:M)}$ and $\log q_\theta(\mathbf{x}_q^{(1:M)})$.

A replay buffer to reuse AIS samples in FAB

- Generate *M* samples with AIS and add them to buffer:
  a. Sample $\mathbf{x}_q^{(1:M)}$ from q_θ obtaining also $\log q_\theta(\mathbf{x}_q^{(1:M)})$.
  b. Obtain $\mathbf{x}_{\text{AIS}}^{(1:M)}$ and $\log w_{\text{AIS}}^{(1:M)}$ by running AIS with input $\mathbf{x}_q^{(1:M)}$ and $\log q_\theta(\mathbf{x}_q^{(1:M)})$.
  c. Add $\mathbf{x}_{\text{AIS}}^{(1:M)}$ to buffer, together with their $\log w_{\text{AIS}}^{(1:M)}$ and $\log q_\theta(\mathbf{x}_q^{(1:M)})$.
- Process *L* minibatches with *N* samples drawn from the buffer:
  a. Sample $\mathbf{x}_{\text{AIS}}^{(1:N)}$ from buffer with probability $w_{\text{AIS}}^{(1:N)}$ and retrieve $\log q_{\theta_{\text{old}}}(\mathbf{x}_{\text{AIS}}^{(1:N)})$.

A replay buffer to reuse AIS samples in FAB

- Generate *M* samples with AIS and add them to buffer:
  a. Sample $\mathbf{x}_q^{(1:M)}$ from q_θ obtaining also $\log q_\theta(\mathbf{x}_q^{(1:M)})$.
  b. Obtain $\mathbf{x}_{\text{AIS}}^{(1:M)}$ and $\log w_{\text{AIS}}^{(1:M)}$ by running AIS with input $\mathbf{x}_q^{(1:M)}$ and $\log q_\theta(\mathbf{x}_q^{(1:M)})$.
  c. Add $\mathbf{x}_{\text{AIS}}^{(1:M)}$ to buffer, together with their $\log w_{\text{AIS}}^{(1:M)}$ and $\log q_\theta(\mathbf{x}_q^{(1:M)})$.
- Process *L* minibatches with *N* samples drawn from the buffer:
  a. Sample $\mathbf{x}_{\text{AIS}}^{(1:N)}$ from buffer with probability $w_{\text{AIS}}^{(1:N)}$ and retrieve $\log q_{\theta_{\text{old}}}(\mathbf{x}_{\text{AIS}}^{(1:N)})$.
  b. Calculate AIS weight correction $\log w_{\text{correction}}^{(1:N)} = \log q_{\theta_{\text{old}}}(\mathbf{x}_{\text{AIS}}^{(1:N)}) - \text{stop-grad}(\log q_\theta(\mathbf{x}_{\text{AIS}}^{(1:N)}))$.

A replay buffer to reuse AIS samples in FAB

- Generate *M* samples with AIS and add them to buffer:
  a. Sample $\mathbf{x}_q^{(1:M)}$ from q_θ obtaining also $\log q_\theta(\mathbf{x}_q^{(1:M)})$.
  b. Obtain $\mathbf{x}_{\text{AIS}}^{(1:M)}$ and $\log w_{\text{AIS}}^{(1:M)}$ by running AIS with input $\mathbf{x}_q^{(1:M)}$ and $\log q_\theta(\mathbf{x}_q^{(1:M)})$.
  c. Add $\mathbf{x}_{\text{AIS}}^{(1:M)}$ to buffer, together with their $\log w_{\text{AIS}}^{(1:M)}$ and $\log q_\theta(\mathbf{x}_q^{(1:M)})$.
- Process *L* minibatches with *N* samples drawn from the buffer:
  a. Sample $\mathbf{x}_{\text{AIS}}^{(1:N)}$ from buffer with probability $w_{\text{AIS}}^{(1:N)}$ and retrieve $\log q_{\theta_{\text{old}}}(\mathbf{x}_{\text{AIS}}^{(1:N)})$.
  b. Calculate AIS weight correction $\log w_{\text{correction}}^{(1:N)} = \log q_{\theta_{\text{old}}}(\mathbf{x}_{\text{AIS}}^{(1:N)}) - \text{stop-grad}(\log q_\theta(\mathbf{x}_{\text{AIS}}^{(1:N)}))$.
  c. Update $\log w_{\text{AIS}}^{(1:N)}$ and $\log q_{\theta_{\text{old}}}(\mathbf{x}_{\text{AIS}}^{1:N})$ in buffer to $\log w_{\text{AIS}}^{(1:N)} + \log w_{\text{correction}}^{(1:N)}$ and $\log q_\theta(\mathbf{x}_{\text{AIS}}^{1:N})$.

A replay buffer to reuse AIS samples in FAB

- Generate *M* samples with AIS and add them to buffer:
  a. Sample $\mathbf{x}_q^{(1:M)}$ from q_θ obtaining also $\log q_\theta(\mathbf{x}_q^{(1:M)})$.
  b. Obtain $\mathbf{x}_{\text{AIS}}^{(1:M)}$ and $\log w_{\text{AIS}}^{(1:M)}$ by running AIS with input $\mathbf{x}_q^{(1:M)}$ and $\log q_\theta(\mathbf{x}_q^{(1:M)})$.
  c. Add $\mathbf{x}_{\text{AIS}}^{(1:M)}$ to buffer, together with their $\log w_{\text{AIS}}^{(1:M)}$ and $\log q_\theta(\mathbf{x}_q^{(1:M)})$.
- Process *L* minibatches with *N* samples drawn from the buffer:
  a. Sample $\mathbf{x}_{\text{AIS}}^{(1:N)}$ from buffer with probability $w_{\text{AIS}}^{(1:N)}$ and retrieve $\log q_{\theta_{\text{old}}}(\mathbf{x}_{\text{AIS}}^{(1:N)})$.
  b. Calculate AIS weight correction $\log w_{\text{correction}}^{(1:N)} = \log q_{\theta_{\text{old}}}(\mathbf{x}_{\text{AIS}}^{(1:N)}) - \text{stop-grad}(\log q_\theta(\mathbf{x}_{\text{AIS}}^{(1:N)}))$.
  c. Update $\log w_{\text{AIS}}^{(1:N)}$ and $\log q_{\theta_{\text{old}}}(\mathbf{x}_{\text{AIS}}^{1:N})$ in buffer to $\log w_{\text{AIS}}^{(1:N)} + \log w_{\text{correction}}^{(1:N)}$ and $\log q_\theta(\mathbf{x}_{\text{AIS}}^{1:N})$.
  d. Evaluate and optimize loss $-1/N \sum_i^N w_{\text{correction}}^{(i)} \log q_\theta(\mathbf{x}_{\text{AIS}}^{(i)})$ with ADAM.

2D mixture of Gaussians problem

Multimodal p and flow q with **pathological initialization:** samples concentrate in small region.

Coupling flow with **15 layers**. **K=1** intermediate AIS distributions with MCMC transitions given by **1 Metropolis-Hastings step**.

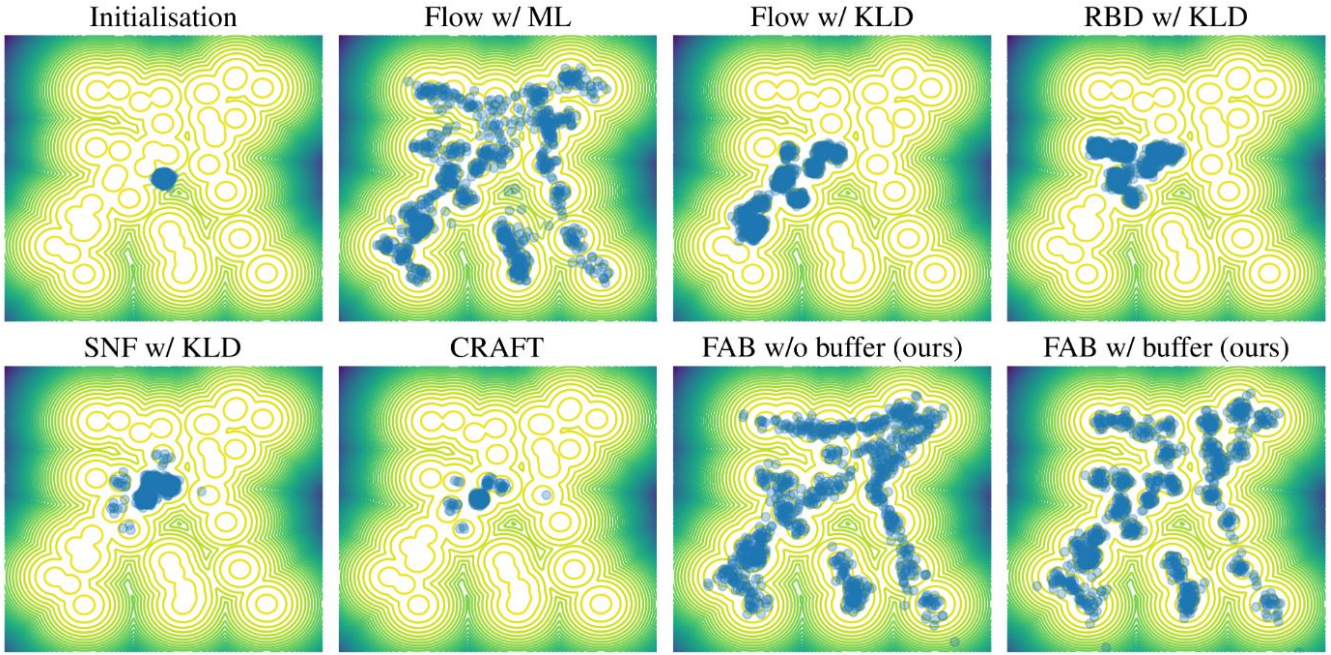


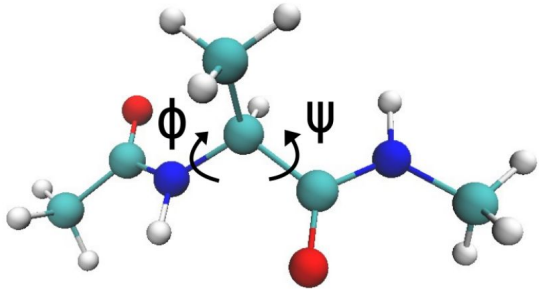
Figure: Contour lines for the target distribution p and samples (blue discs) drawn from the approximation q_θ obtained by different methods.

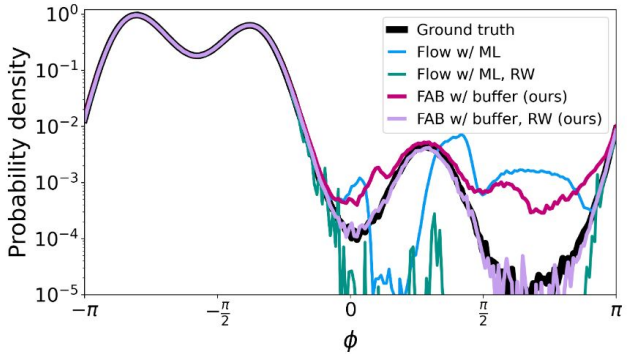
Results for alanine dipeptide

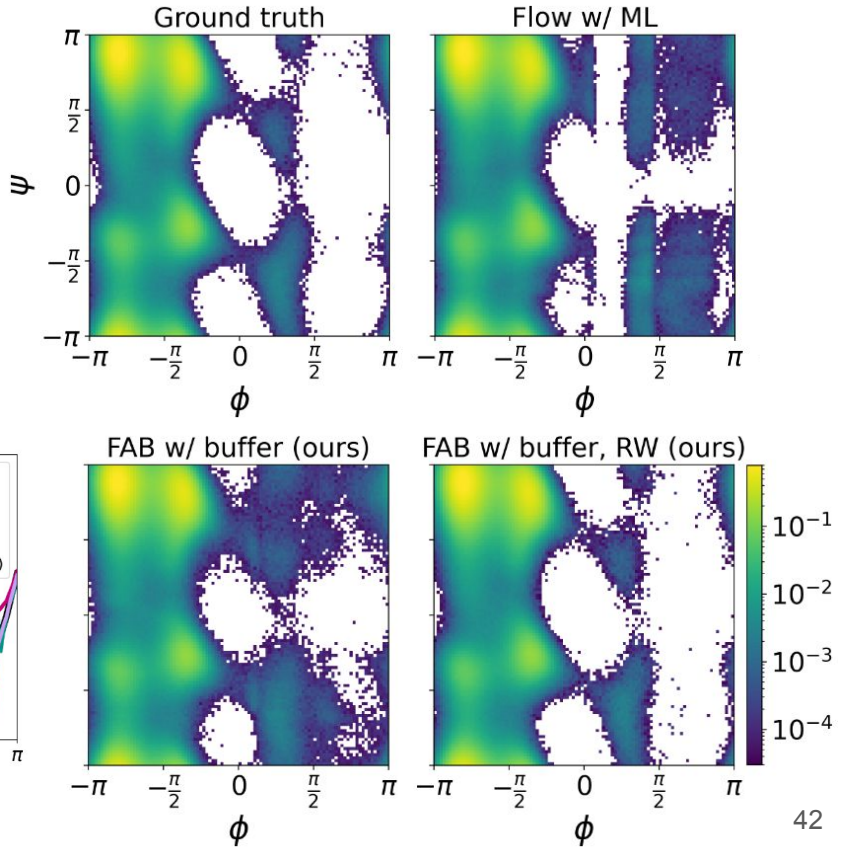
Neural Spline Flows with **12 layers**. Some bond angles treated as **circular** coordinates.

K = 8 intermediate AIS distributions.

Hamiltonian Monte Carlo with 4 Leapfrog steps as AIS transition operator.







Results for alanine dipeptide

Table: ESS, log-likelihood on the test set, and KL divergence (KLD) of Ramachandran plots with and without reweighting (RW) for each method. Our methods are marked in *italic* and best results are emphasized in **bold**.

	ESS (%)	$\mathrm{E}_{p(\mathbf{x})}[\log q(\mathbf{x})]$	KLD	KLD w/ RW
Flow w/ ML	2.8 ± 0.6	209.22 ± 0.28	$\mathbf{(7.57 \pm 3.80) \times 10^{-3}}$	$(2.58 \pm 0.80) \times 10^{-2}$
Flow w/ $D_{\alpha=2}$	0.011 ± 0.000	73.5 ± 1.3	2.96 ± 0.13	17.5 ± 0.2
Flow w/ KLD	54 ± 12	100 ± 32	3.17 ± 0.20	3.15 ± 0.19
RBD w/ KLD	44 ± 18	143 ± 22	3.00 ± 0.05	3.00 ± 0.04
SNF w/ KLD	0.16 ± 0.11	N/A $\pm$ N/A	8.71 ± 3.36	9.58 ± 2.68
FAB w/o buffer	52.2 ± 1.3	211.13 ± 0.03	$(6.28 \pm 0.33) \times 10^{-2}$	$(2.66 \pm 0.90) \times 10^{-2}$
FAB w/ buffer	$\mathbf{92.8 \pm 0.1}$	$\mathbf{211.54 \pm 0.00}$	$\mathbf{(3.42 \pm 0.45) \times 10^{-3}}$	$\mathbf{(2.51 \pm 0.39) \times 10^{-3}}$

Take home messages

We have proposed FAB, a method that

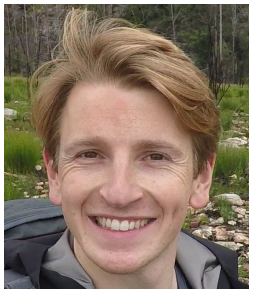
- Allows flows to fit **multimodal distributions** when **training from energy only**.
- Key ingredients: α-divergence, AIS, bootstrap training, replay buffer and minimum variance importance sampling distributions.
- Only requires the target density, but **no samples from the target**.
- First method to approximate Boltzmann distribution of **alanine dipeptide** without using MD samples while using 100 × fewer target evaluations.

Collaborators

Bernhard Schölkopf

Laurence Midgley

Vincent Stimper



Gregor Simm

Thanks!