Sliced Normalizing Flow optimization and sampling

Uroš Seljak
UC Berkeley

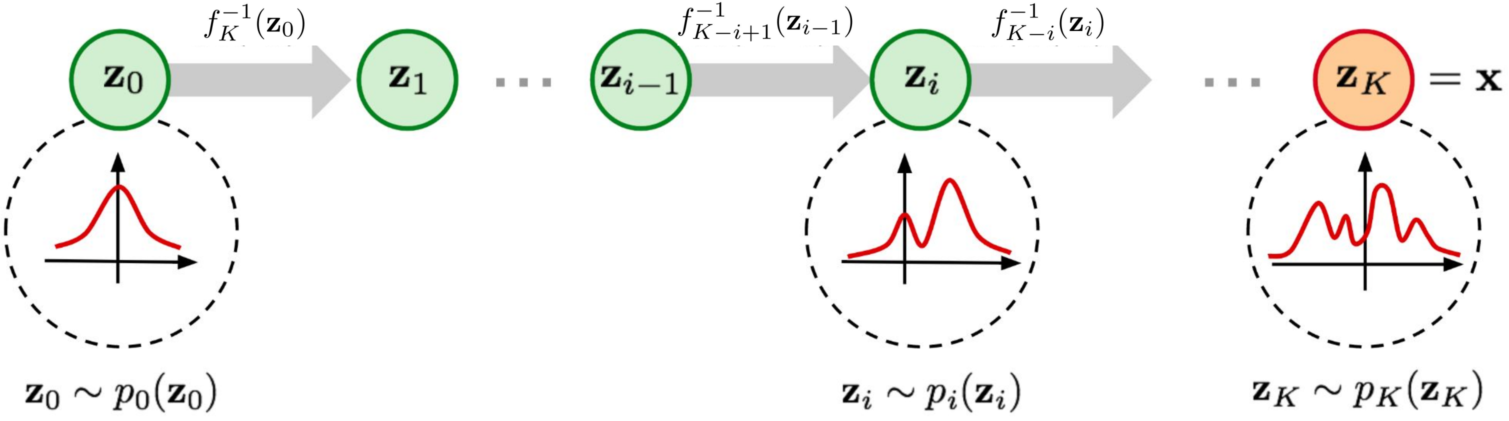
October 1 2021 @ Berkeley

Collaborators: **Biwei Dai**, George Stein, Richard Grumitt, Adrian Bayer, James Sullivan

Outline

- Introduction to Normalizing Flows
    - **S**liced (**I**terative) **N**ormalizing **F**low (SINF)
    - Anomaly detection application in HEP
- normalizing flow optimization and sampling

Normalizing Flows for density estimation, sampling



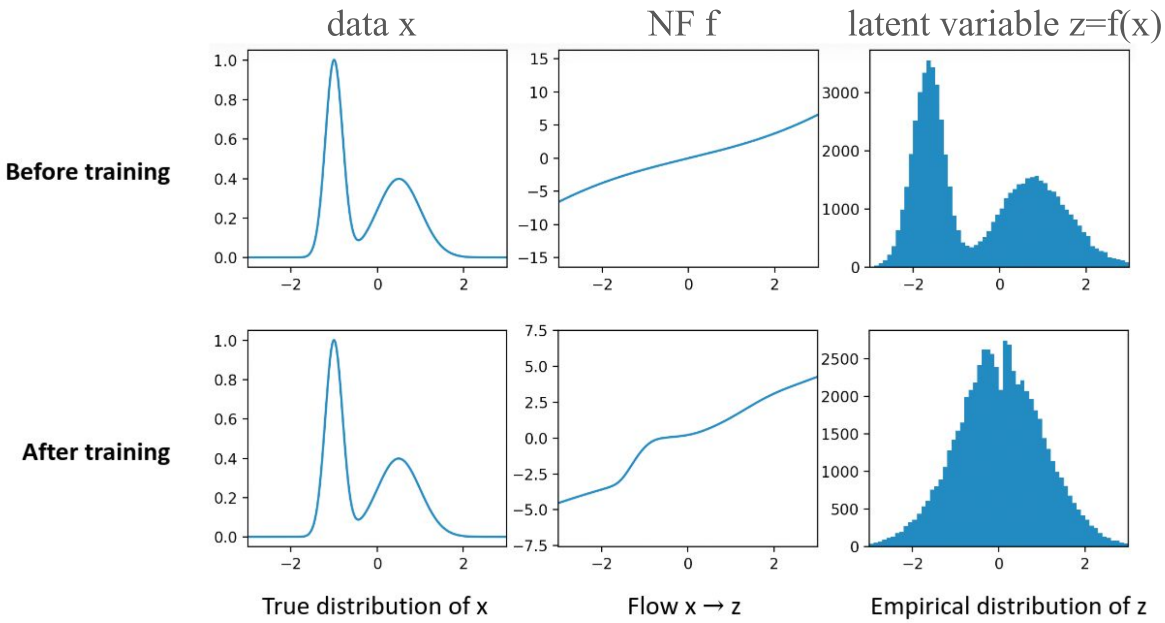
$f_K^{-1}(\mathbf{z}_0)$
$f_{K-i+1}^{-1}(\mathbf{z}_{i-1})$
$f_{K-i}^{-1}(\mathbf{z}_i)$
$\mathbf{z}_0$
$\mathbf{z}_1$
$\mathbf{z}_{i-1}$
$\mathbf{z}_i$
$\mathbf{z}_K = \mathbf{x}$
$\mathbf{z}_0 \sim p_0(\mathbf{z}_0)$
$\mathbf{z}_i \sim p_i(\mathbf{z}_i)$
$\mathbf{z}_K \sim p_K(\mathbf{z}_K)$

Credit: https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html

- Bijective mapping f between data x and latent variable z (z = f(x), z ~ π(z))
  - **Evaluate density**: p(x) = π(f(x)) |det(df/dx)|
  - **Sample**: x = f^{-1}(z) (z ~ π(z))

Normalizing Flows

- 1D example



data x

NF f

latent variable z=f(x)

Before training

After training

True distribution of x

Flow x → z

Empirical distribution of z

Credit: https://sites.google.com/view/berkeley-cs294-158-sp20/home

- One possible training objective: maximize **<log p(x)>**

  p(x) = π(f(x)) |det(df/dx)|

This objective equals to Kullback-Leibler divergence minimization between approximate and true distribution

- **Evaluate density**: p(x) = π(f(x)) |det(df/dx)|
- **Sample**: x = $f^{-1}(z)$ (z ~ π(z))

Normalizing Flows

In d dimensions, we need to parametrize d dimensional normalizing flow transformation $f = f_1 \circ f_2 \circ \ldots \circ f_K$ and train it on $\langle \log p(x) \rangle$ or some other loss function

Recall that **Evaluate density**: $p(x) = \pi(f(x))\, |\det(df/dx)|$

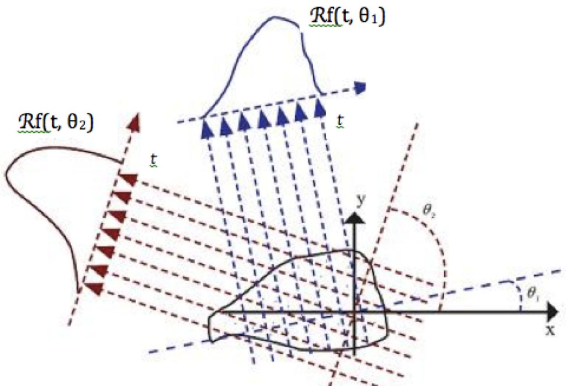
Sample: $x = f^{-1}(z)$ $(z \sim \pi(z)$

How to parametrize f, such that **its inverse and Jacobian determinant are easy to evaluate**?

Many NFs in the literature. Some are continuous (Ordinary Differential Equations, Stochastic DE).

Sliced Iterative Normalizing Flow (Dai & Seljak, ICML 2021) builds the flow as a sequence of sliced projections

Radon transform



Reconstruction of a 2d image from 1d slices

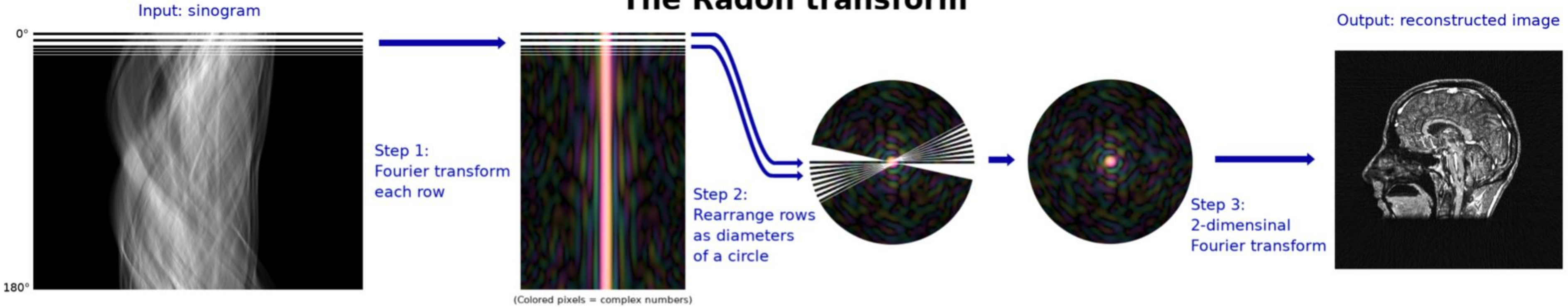
Invertible, almost the same as Fourier transform

$$\mathcal{R}^{-1}((\mathcal{R}p)(t,\theta))(x) = \int_{\mathbb{S}^{n-1}} ((\mathcal{R}p)(\cdot,\theta) * h)(\langle x,\theta\rangle)d\theta$$

h has the Fourier transform $\hat{h}(k) = c|k|^{d-1}$.

Computer Axial Tomography (CAT scan)

The Radon transform



Input: sinogram
0°
180°
(Colored pixels = complex numbers)
Output: reconstructed image

Step 1: Fourier transform each row

Step 2: Rearrange rows as diameters of a circle

Step 3: 2-dimensinal Fourier transform

Radon transform can represent a general density distribution with its slices. But too expensive in high dimensions

Sliced Wasserstein distances

$$(\mathcal{R}p)(t,\theta) = \int_{\mathbb{R}^d} p(x)\delta(t - \langle x, \theta \rangle)dx, \qquad (2)$$

where $\mathbb{S}^{n-1}$ denotes the unit sphere $\theta_1^2 + \cdots \theta_n^2 = 1$ in $\mathbb{R}^n$, $\delta(\cdot)$ is the Dirac delta function, and $\langle \cdot, \cdot \rangle$ is the standard inner product in $\mathbb{R}^n$. For a given θ, the function $(\mathcal{R}p)(\cdot, \theta) : \mathbb{R} \to \mathbb{R}$ is essentially the slice (or projection) of $p(x)$ on axis θ.

Radon transform **can represent a general density distribution with its slices**. This motivates

Sliced p-Wasserstein Distance (SWD), is defined as:

$$SW_p(p_1, p_2) = \left(\int_{\mathbb{S}^{d-1}} W_p^p(\mathcal{R}p_1(\cdot,\theta), \mathcal{R}p_2(\cdot,\theta))d\theta \right)^{\frac{1}{p}}. \qquad (5)$$

This is still very expensive in high dimensions (curse of dimensionality)

The SWD can be calculated by approximating the high dimensional integral with Monte Carlo samples. However, in high dimensions a large number of projections is required to accurately estimate SWD. This motivates to use the maximum Sliced p-Wasserstein Distance (max SWD):

$$\max\text{-}SW_p(p_1, p_2) = \max_{\theta \in \mathbb{S}^{d-1}} W_p(\mathcal{R}p_1(\cdot,\theta), \mathcal{R}p_2(\cdot,\theta)), \qquad (6)$$

which is the maximum of the Wasserstein distance of the 1D marginalized distributions of all possible directions. SWD

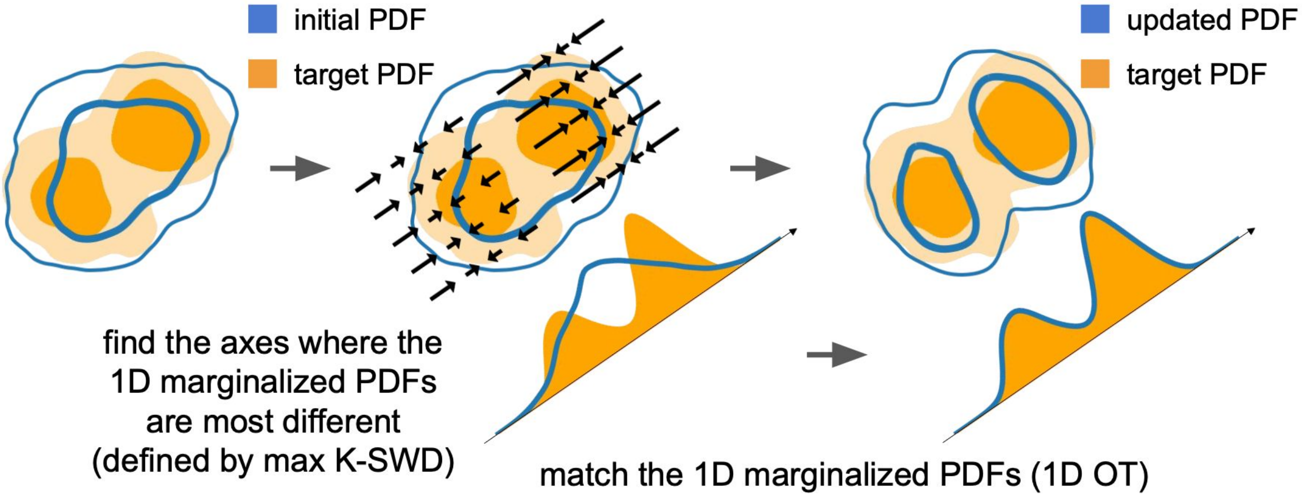
We generalize the idea of maximum SWD and propose maximum K-Sliced p-Wasserstein Distance (max K-SWD):

$$\max\text{-}K\text{-}SW_p(p_1, p_2) = \max_{\{\theta_1,\cdots,\theta_K\}\text{ orthonormal}} \left(\frac{1}{K} \sum_{k=1}^{K} W_p^p((\mathcal{R}p_1)(\cdot,\theta_k), (\mathcal{R}p_2)(\cdot,\theta_k)) \right)^{\frac{1}{p}}. \qquad (7)$$

Sliced Iterative Normalizing Flow: find directions of largest deviation from target distribution

When the target is a Gaussian we are using optimization to find K orthogonal most non-Gaussian slices

Then we Gaussianize and repeat



🟦 initial PDF

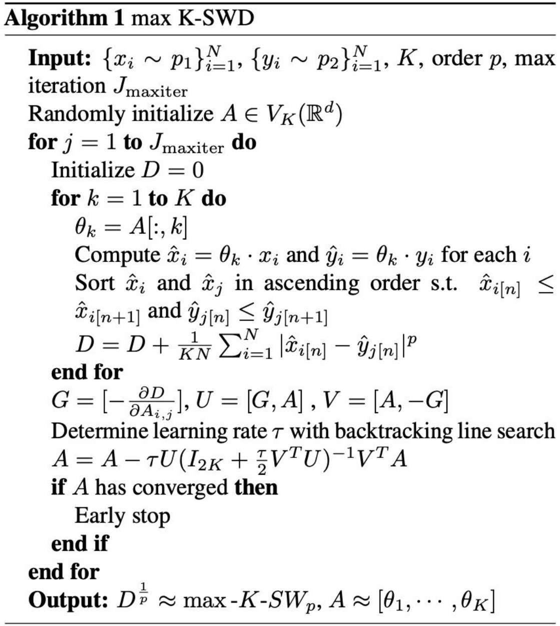
🟧 target PDF

🟦 updated PDF

🟧 target PDF

find the axes where the 1D marginalized PDFs are most different (defined by max K-SWD)

match the 1D marginalized PDFs (1D OT)



```
Algorithm 1 max K-SWD
Input: $\{x_i \sim p_1\}_{i=1}^N$, $\{y_i \sim p_2\}_{i=1}^N$, $K$, order $p$, max iteration $J_{\text{maxiter}}$
Randomly initialize $A \in V_K(\mathbb{R}^d)$
for $j = 1$ to $J_{\text{maxiter}}$ do
    Initialize $D = 0$
    for $k = 1$ to $K$ do
        $\theta_k = A[:, k]$
        Compute $\hat{x}_i = \theta_k \cdot x_i$ and $\hat{y}_i = \theta_k \cdot y_i$ for each $i$
        Sort $\hat{x}_i$ and $\hat{x}_j$ in ascending order s.t. $\hat{x}_{i[n]} \leq \hat{x}_{i[n+1]}$ and $\hat{y}_{j[n]} \leq \hat{y}_{j[n+1]}$
        $D = D + \frac{1}{KN}\sum_{i=1}^{N} |\hat{x}_{i[n]} - \hat{y}_{j[n]}|^p$
    end for
    $G = [-\frac{\partial D}{\partial A_{i,j}}]$, $U = [G, A]$, $V = [A, -G]$
    Determine learning rate $\tau$ with backtracking line search
    $A = A - \tau U(I_{2K} + \frac{\tau}{2}V^T U)^{-1} V^T A$
    if $A$ has converged then
        Early stop
    end if
end for
Output: $D^{\frac{1}{p}} \approx \text{max-}K\text{-}SW_p$, $A \approx [\theta_1, \cdots, \theta_K]$
```

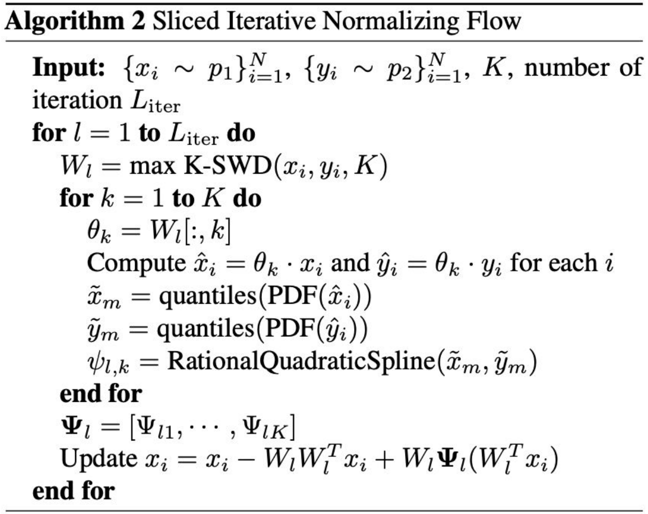
Sliced Iterative Normalizing Flow

Find directions of largest Max-K W distance
Gaussianize the distribution in that direction, repeat
Multiply the Jacobians

$$X_{l+1} = W_l \mathbf{\Psi}_l (W_l^T X_l) + X_l^{\perp}$$

$$X_l = W_l \mathbf{\Psi}_l^{-1} (W_l^T X_{l+1}) + X_l^{\perp}$$

$$\det\left(\frac{\partial X_{l+1}}{\partial X_l}\right) = \prod_{k=1}^{K} \frac{d\Psi_{lk}(x)}{dx}$$



Algorithm 2 Sliced Iterative Normalizing Flow

```
Input: $\{x_i \sim p_1\}_{i=1}^N$, $\{y_i \sim p_2\}_{i=1}^N$, $K$, number of iteration $L_{\mathrm{iter}}$
for $l = 1$ to $L_{\mathrm{iter}}$ do
    $W_l = \max$ K-SWD$(x_i, y_i, K)$
    for $k = 1$ to $K$ do
        $\theta_k = W_l[:, k]$
        Compute $\hat{x}_i = \theta_k \cdot x_i$ and $\hat{y}_i = \theta_k \cdot y_i$ for each $i$
        $\tilde{x}_m = \mathrm{quantiles}(\mathrm{PDF}(\hat{x}_i))$
        $\tilde{y}_m = \mathrm{quantiles}(\mathrm{PDF}(\hat{y}_i))$
        $\psi_{l,k} = \mathrm{RationalQuadraticSpline}(\tilde{x}_m, \tilde{y}_m)$
    end for
    $\mathbf{\Psi}_l = [\Psi_{l1}, \cdots, \Psi_{lK}]$
    Update $x_i = x_i - W_l W_l^T x_i + W_l \mathbf{\Psi}_l (W_l^T x_i)$
end for
```

This can be trained as a flow from data to Normal (GIS) or viceversa (SIG)
It has a NN structure:

1) linear combination of previous layer (weights), enforced to be orthogonal transforms.
2) Pointwise nonlinearity: spline, more general than ReLU

SINF as a generative model: SIG

- (SIG) trains in data space: allows directly optimizing the distribution of samples

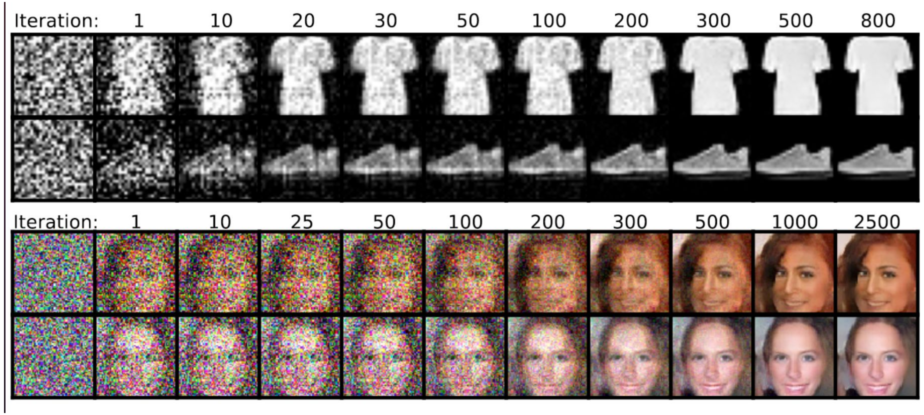
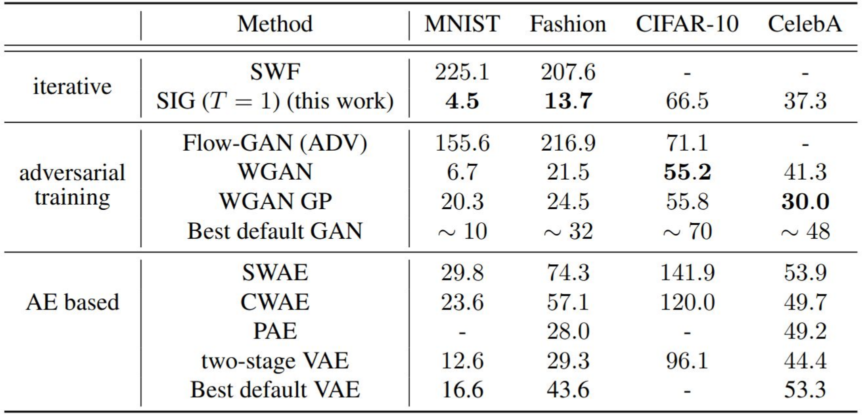
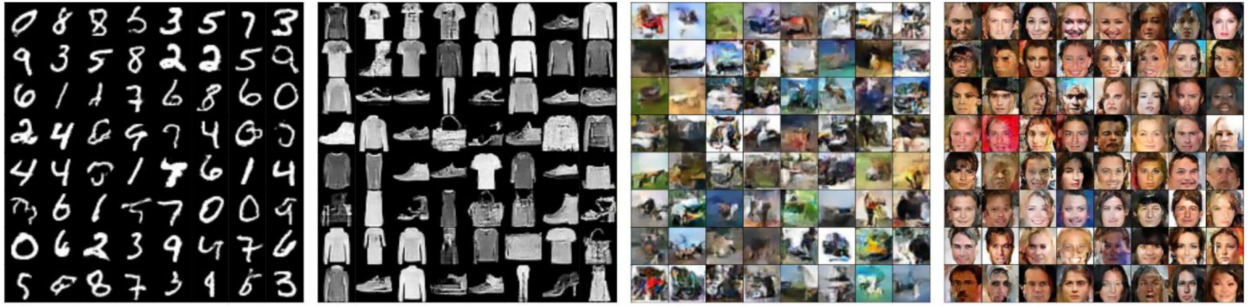

Iteration: 1 10 20 30 50 100 200 300 500 800
Iteration: 1 10 25 50 100 200 300 500 1000 2500

Table 2. FID scores on different datasets (lower is better). The errors are generally smaller than the differences.



	Method	MNIST	Fashion	CIFAR-10	CelebA
iterative	SWF	225.1	207.6	-	-
	SIG ($T = 1$) (this work)	**4.5**	**13.7**	66.5	37.3
adversarial training	Flow-GAN (ADV)	155.6	216.9	71.1	-
	WGAN	6.7	21.5	**55.2**	41.3
	WGAN GP	20.3	24.5	55.8	**30.0**
	Best default GAN	~ 10	~ 32	~ 70	~ 48
AE based	SWAE	29.8	74.3	141.9	53.9
	CWAE	23.6	57.1	120.0	49.7
	PAE	-	28.0	-	49.2
	two-stage VAE	12.6	29.3	96.1	44.4
	Best default VAE	16.6	43.6	-	53.3



(a) MNIST (b) Fashion-MNIST (c) CIFAR-10 (d) CelebA

These are 784-3072 dimensions examples

2007.00674
https://github.com/biweidai/SIG_GIS

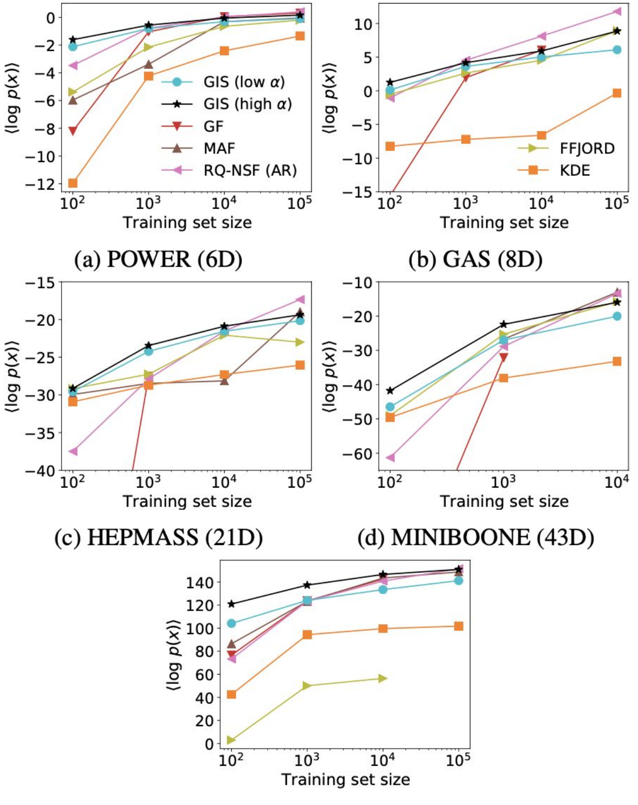
SINF as a density estimator (GIS)

- GIS Iteratively building the NF based on optimal transport on 1D slices from data to Normal
- GIS achieves the best density estimation results on small training sets. Better inductive bias?
- Very few hyperparameters
- KDE can be very poor density estimator
- GIS is much faster than existing NFs: O(1s) on CPU

Table 3. Averaged training time of different NF models on small datasets ($N_{\text{train}} = 100$) measured in seconds. All the models are tested on both a cpu and a K80 gpu, and the faster results are reported here (the results with * are run on gpus.). P: POWER, G: GAS, H: HEPMASS, M: MINIBOONE, B: BSDS300.

Method	P	G	H	M	B
GIS (low α)	0.53	1.0	0.63	3.5	7.4
GIS (high α)	6.8	9.4	7.3	44.1	69.1
GF	113*	539*	360*	375*	122*
MAF	18.4	-[1]	10.2	-[1]	32.1
FFJORD	1051	1622	1596	499*	4548*
RQ-NSF (AR)	118	127	55.5	38.9	391

[1] Training failures.



(a) POWER (6D)

(b) GAS (8D)

(c) HEPMASS (21D)

(d) MINIBOONE (43D)

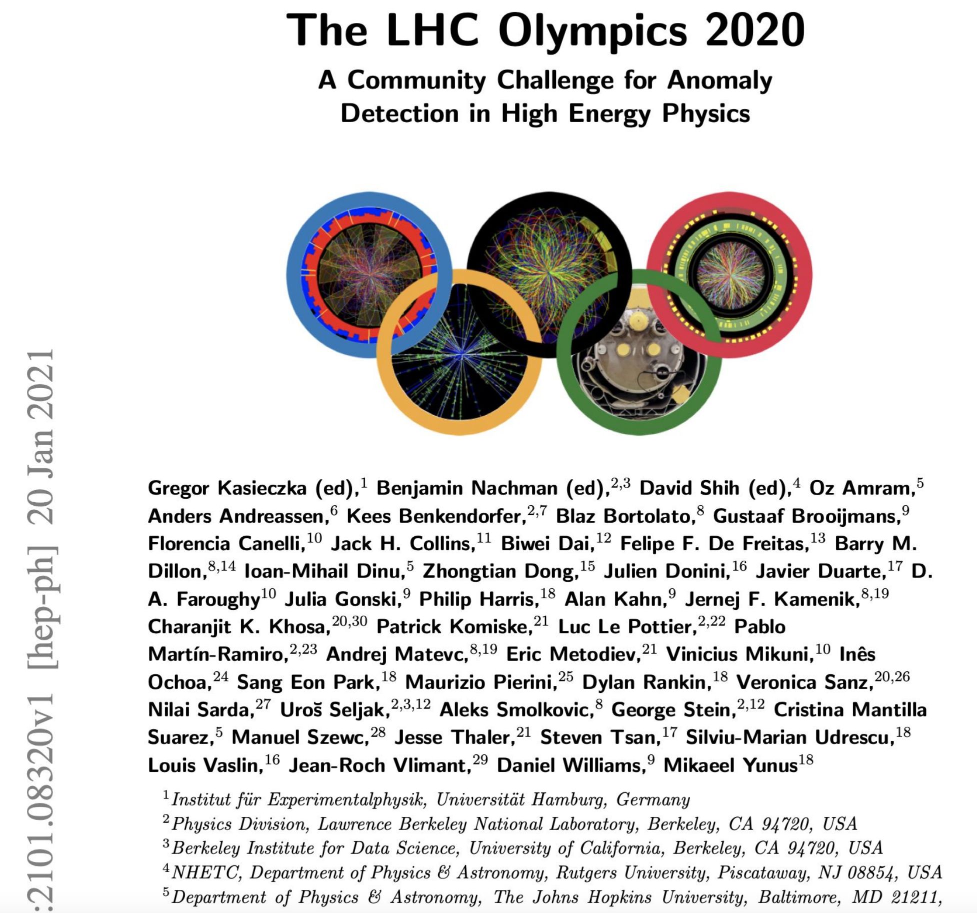
(e) BSDS300 (63D)

Figure 3. Density estimation on small training sets. The legends in panel (a) and (b) apply to other panels as well. At 100 training data GIS has the best performance in all cases.

SINF is an Outlier or Anomaly Detector

1 million Pythia simulated background events of an LHC hadronic event
Unknown number of resonant signal events
Task: find the mass, cross-sectional area etc.

Our approach: find jets, their mass, subjettiness, determine jet invariant mass M_{JJ}, compute SINF conditional density $p(x|M_{JJ})$ where x is 4-dimensional.



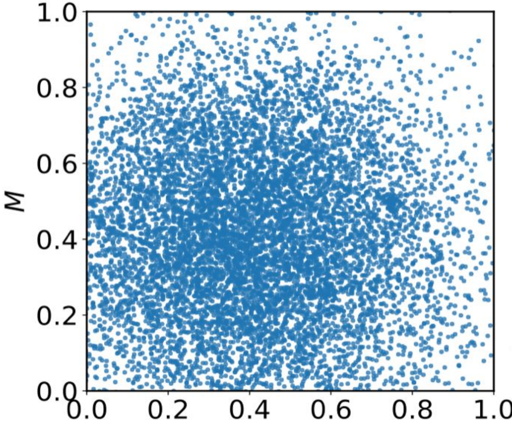
:2101.08320v1 [hep-ph] 20 Jan 2021

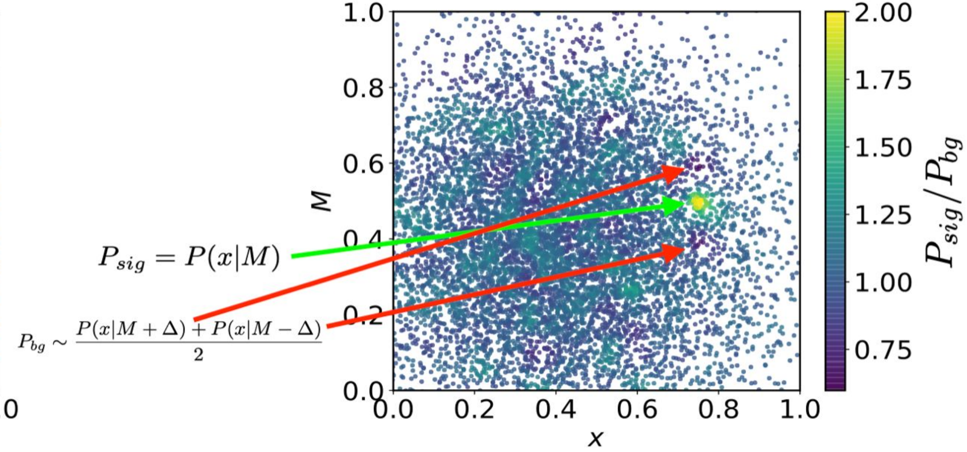
The LHC Olympics 2020

A Community Challenge for Anomaly Detection in High Energy Physics

Gregor Kasieczka (ed),[1] Benjamin Nachman (ed),[2,3] David Shih (ed),[4] Oz Amram,[5] Anders Andreassen,[6] Kees Benkendorfer,[2,7] Blaz Bortolato,[8] Gustaaf Brooijmans,[9] Florencia Canelli,[10] Jack H. Collins,[11] Biwei Dai,[12] Felipe F. De Freitas,[13] Barry M. Dillon,[8,14] Ioan-Mihail Dinu,[5] Zhongtian Dong,[15] Julien Donini,[16] Javier Duarte,[17] D. A. Faroughy[10] Julia Gonski,[9] Philip Harris,[18] Alan Kahn,[9] Jernej F. Kamenik,[8,19] Charanjit K. Khosa,[20,30] Patrick Komiske,[21] Luc Le Pottier,[2,22] Pablo Martín-Ramiro,[2,23] Andrej Matevc,[8,19] Eric Metodiev,[21] Vinicius Mikuni,[10] Inês Ochoa,[24] Sang Eon Park,[18] Maurizio Pierini,[25] Dylan Rankin,[18] Veronica Sanz,[20,26] Nilai Sarda,[27] Uroš Seljak,[2,3,12] Aleks Smolkovic,[8] George Stein,[2,12] Cristina Mantilla Suarez,[5] Manuel Szewc,[28] Jesse Thaler,[21] Steven Tsan,[17] Silviu-Marian Udrescu,[18] Louis Vaslin,[16] Jean-Roch Vlimant,[29] Daniel Williams,[9] Mikaeel Yunus[18]

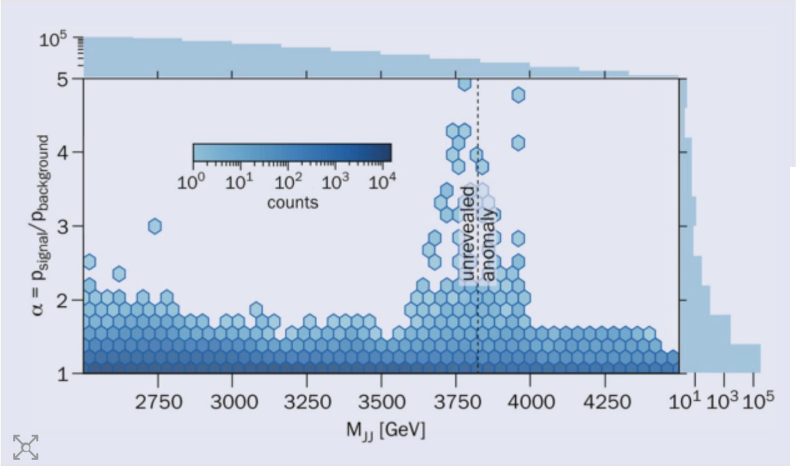
[1] *Institut für Experimentalphysik, Universität Hamburg, Germany*
[2] *Physics Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA*
[3] *Berkeley Institute for Data Science, University of California, Berkeley, CA 94720, USA*
[4] *NHETC, Department of Physics & Astronomy, Rutgers University, Piscataway, NJ 08854, USA*
[5] *Department of Physics & Astronomy, The Johns Hopkins University, Baltimore, MD 21211,*





$$P_{sig} = P(x|M)$$

$$P_{bg} \sim \frac{P(x|M+\Delta) + P(x|M-\Delta)}{2}$$



Olympian algorithm The "anomaly score", α, as a function of the invariant mass of the leading two jets of events in "black box 1" of the LHCO data challenge, in the analysis of Stein, Seljak and Dai, who used an early form of a technique that is now called "Gaussianising iterative slicing". A number of anomalous events are seen near 3750 GeV. Credit: arXiv:2101.08320

From CERN courier 8/31/2021
(Nachman & van Beekveld)

The best performance on the first black box in the LHCO challenge, as measured by finding and correctly characterising the anomalous signals, was by a team of cosmologists at Berkeley (George Stein, Uros Seljak and Biwei Dai) who compared the phase-space density between a sliding signal region and sidebands (see "Olympian algorithm" figure). Overall, the algorithms did well on the R&D dataset, and some also did well on the first black box, with methods that made use of likelihood ratios proving

Outline

- Introduction to Normalizing Flows
  - **S**liced (**I**terative) **N**ormalizing **F**low (SINF)
  - Anomaly detection application in HEP
- Data analysis: normalizing flows optimization and sampling

Normalizing Flows for Bayesian posteriors

- We want posteriors $p(y|x)$ of correlated parameters y, but we only know $p(x,y)=p(x|y)p(y)$
- Monte Carlo Markov Chain as the method of choice. Many choices: Metropolis-Hastings, Sequential MC, nested sampling, Hamiltonian MC, Langevin MC (MALA), Gibbs MC, sliced sampling, affine sampling…
- Many issues: samples are correlated, chains may not have converged (burn-in), some methods do not mix between the separated peaks...
- We may also want Bayesian evidence (normalization constant)
- For scientific applications likelihood can be expensive (e.g. an ODE or PDE to make predictions for the data)
- We often do not have access to the gradient (score)

Optimization view: surrogating the posteriors

Let’s surrogate the posterior p(y|x) with SINF $q_{\boldsymbol{\phi}}(y)$, where y are parameters and x data and $\boldsymbol{\phi}$ are the parameters of SINF

What we have access to is p(x|y)p(y)=p(x,y), but we do not know where it peaks and where it is low. We also do not know normalization (partition function) Z=p(x), so we do not know p(y|x)=p(x,y)/p(x)

Previously we had samples of x and we fitted density p(x). Now this is a harder problem since we do not have samples, so we have to create them

Basic idea: start from the prior p(y), use annealing flow from $\boldsymbol{\beta}$=0 to $\boldsymbol{\beta}$=1

We want to fit with small number of samples: SINF is a good choice

Annealing Flow ODE/PDE

Let’s define temperature dependent target $p(\beta,y)=p(x|y)^{\beta}p(y)$: $p(\beta=0,y)=p(y)$

Let’s define a loss function $L(\beta)$ that quantifies the dissimilarity between target $p(\beta,y)$ and current density $q_{\phi}(y)$ in terms of the samples from the current $q_{\phi}(y)$

E.g. $L(\beta)=E_{y\sim q(y)}(\ln p(\beta,y)-\ln q_{\phi}(y)-\ln Z)^2$ EL2O divergence (no score available)

$L(\beta)=E_{y\sim q(y)}(\nabla_y \ln p(\beta)-\nabla_y \ln q_{\phi})^2$ Fisher divergence (score available)

Annealing Flow: $\mathbf{dq_{\phi}/d\beta = -\lambda[\nabla_{\phi}L(\beta)]^T\nabla_{\phi}q_{\phi}}$: q_{ϕ} is normalized so no need to have explicit continuity

Let’s unpack this: $dq_{\phi}/d\beta=[\nabla_{\phi}q_{\phi}]^T d\phi/d\beta$. In terms of the flow of ϕ: $\mathbf{d\phi/d\beta = -\lambda\nabla_{\phi}L(\beta)}$

So this is a gradient descent of the NF parameters with the loss function gradient so that $q_{\phi}(y)$ relaxes to $p(\beta,y)$. However, the loss function is continuously modified in terms of the target density $p(\beta,y)$. Here $\lambda\beta$ is the learning rate of gradient descent.

Algorithm

Start at $\boldsymbol{\beta}$=0 by drawing samples from prior p(y)

Adaptive choice of discretization next $\boldsymbol{\beta}_{new}$: choose so that effective sample size (ESS) evaluated using p($\boldsymbol{\beta}_{new}$,y)/p($\boldsymbol{\beta}_{old}$,y) is 0.5 of total

Apply importance weight IW=p($\boldsymbol{\beta}_{new}$,y)/p($\boldsymbol{\beta}_{old}$,y) to the samples,
resample the samples with Bernoulli using IW, eliminating low IW samples

Fit $q_{\boldsymbol{\phi}}$(y) to p($\boldsymbol{\beta}_{new}$,y) on current samples using a few (stochastic) gradient descents on the dissimilarity loss function starting from previous fit (ODE/PDE): **optimization**

Metropolis-Hastings adjustment: Draw new samples y' from $q_{\boldsymbol{\phi}}$(y): **sampling.** Evaluate p($\boldsymbol{\beta}_{new}$,y') on new samples. Compare each new sample y' to one old sample y. Accept new sample with probability r=min(1,p($\boldsymbol{\beta}_{new}$,y')q(y)/q(y')[p($\boldsymbol{\beta}_{new}$,y)]). If not enough samples accepted repeat until 50% acceptance. Repeat until $\boldsymbol{\beta}$=1.

Markov Process view

Markov chain: new sample y' depends only on the property of previous sample y. But how do we choose transition proposal J(y'|y)?

To equilibrate to the stationary target distribution p(y|x) one must satisfy detailed balance p($\boldsymbol{\beta}$,y)T(y'|y)=p($\boldsymbol{\beta}$,y')T(y|y'), where T(y'|y) is transition probability of y' given y. T(y'|y)=rJ(y'|y)

To achieve detailed balance we accept the proposal y' with probability r=min(1,p($\boldsymbol{\beta}$,y')J(y|y')/[p($\boldsymbol{\beta}$,y)J(y'|y)])

Two main issues of MCMC: correlated samples (requiring thinning of the chains) and low acceptance rate

What if the proposal J(y'|y) is independent of y and only given by q(y')? Then the samples are uncorrelated: **perfect mixing**

r=min(1,p($\boldsymbol{\beta}$,y')q(y)/q(y')[p($\boldsymbol{\beta}$,y)]). If q(y)=p($\boldsymbol{\beta}$,y)/p(x) we have r=1. We achieved **perfect acceptance.**

We strive for acceptance of the order 0.5 at every $\boldsymbol{\beta}$. Quality of SINF q(y) fits is crucial.

Preconditioner view

We can view Normalizing Flows as powerful **preconditioners.** They can handle high condition numbers varying across parameter space.

If we have a NF map y=f(z) then in latent space the target distribution is $p(\boldsymbol{\beta},z)=p(\boldsymbol{\beta},f^{-1}(z))|df^{-1}(z)/dz|$ (Parno & Marzouk 2014).

With NF we may simplify the geometry in distribution space: a generalization of second order (Newton's) methods for sampling and optimization. Position dependent curvature in Riemannian geometry. It solves the high condition problem of samplers.

We can draw sample z' from a Gaussian centered at a previous sample z in NF latent space z=f(y): Metropolis-Hastings sampling in latent space. Acceptance can be high if z' close to z, but correlated, so we do it many times to decorrelate. Acceptance rate is $r=\min[1,p(\boldsymbol{\beta},z')/p(\boldsymbol{\beta},z)]$.

If we have gradient of $p(\boldsymbol{\beta},z)$ we can use Hamiltonian or Langevin dynamics in latent space for better sampling acceptance.

Riemannian geometry view

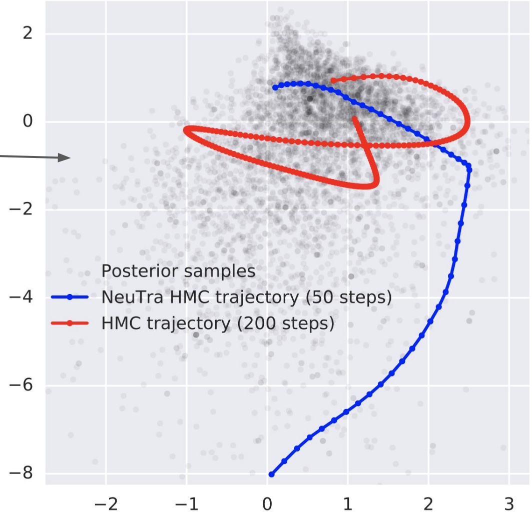
In Riemannian geometry metric and curvature are varying with position y

Define NF Jacobian $J=df^{-1}(z)/dz$ and potential $U=-\log p(\beta,f^{-1}(z))$

Hamiltonian in latent space is $H=U-\log|J|+m^{T}m/2$: we can run Hamiltonian MC dynamics

Define position dependent metric $G=(JJ^{T})^{-1}$ and $m'=J^{T-1}m$. G is also called mass matrix.

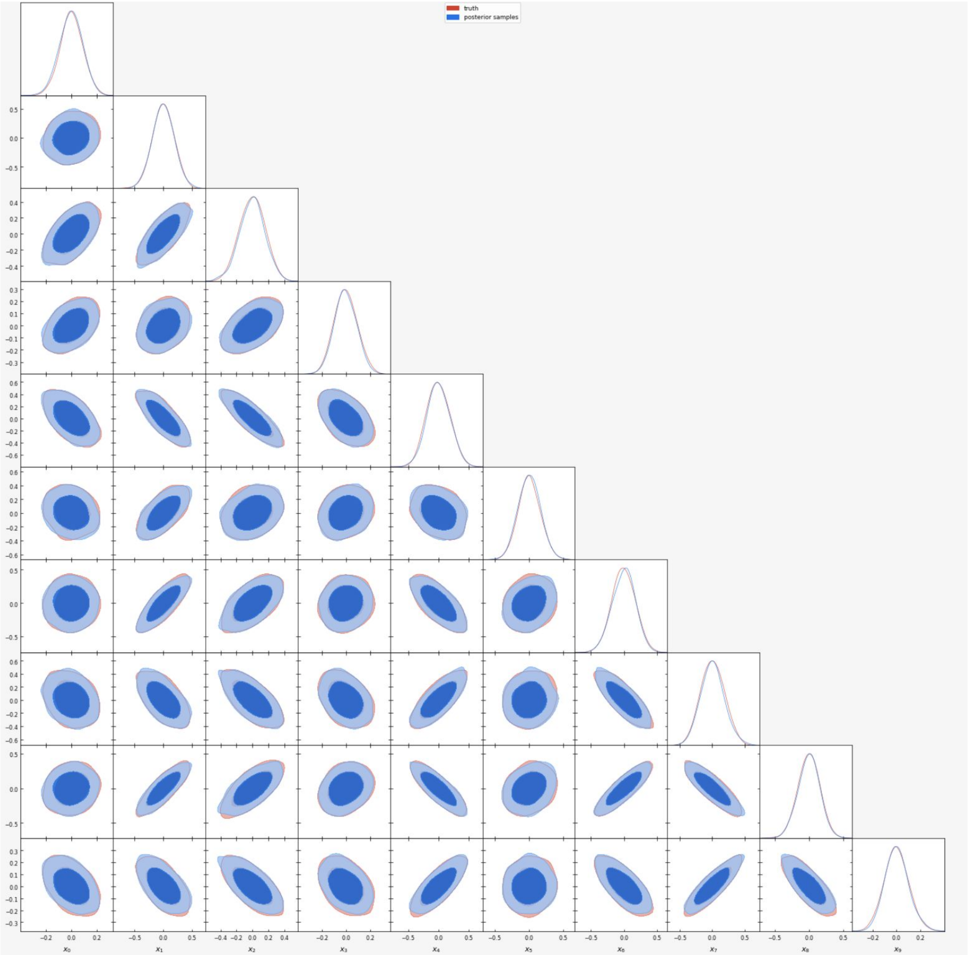
Then $H=U+m'^{T}G^{-1}m'/2+\log|G|/2$: Hamiltonian in a curved space (Girolami & Calderhead 2011, Hoffman etal 2019)



In this view NFs describe the variable metric space and map it to a space where geometry is simple (latent space: Gaussian with zero mean and unit variance)

While Hamiltonians are equal, it is easier to solve HMC dynamics in latent space and map samples to parameter space than to solve it in curved space because NFs have treatable Jacobians and their gradients

Does it work for Bayesian posteriors in higher dimensions?



Work in progress, but here is a 10 dimensional correlated Gaussian example with 3,000 samples drawn in total (**parallelized**, 200-400 per beta step) that converged after 10 beta steps to a near perfect solution (red versus blue). **No gradient (score) used!**

This is competitive or better than state of the art: it takes 2-3 times fewer evaluations than SMC

5 seconds on a laptop

Encouraging, but needs more stress-testing on harder distributions

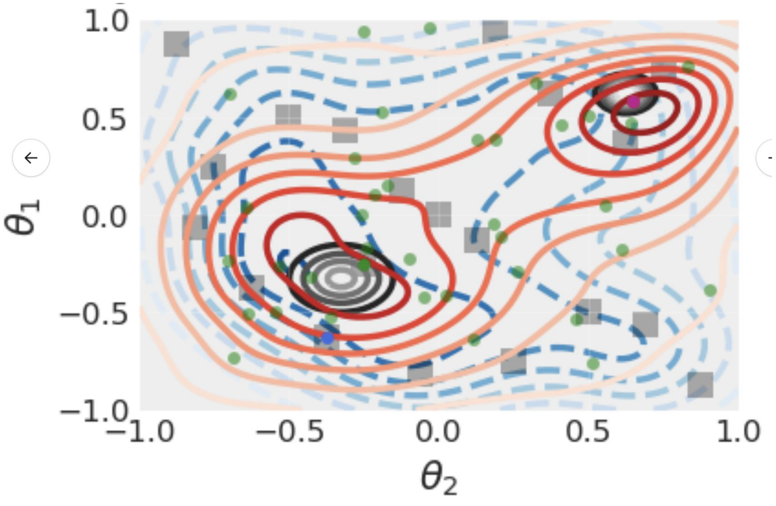
Multi-modal problems

Annealing is one of the best (the best?) methods to handle multi-modality in sampling applications

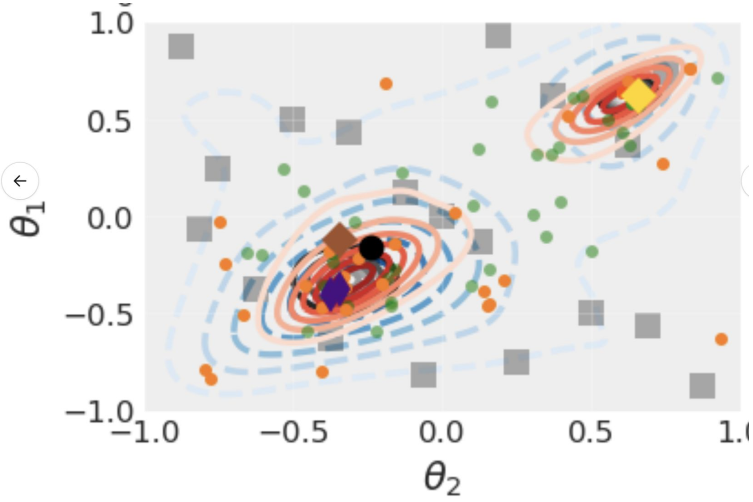
Since SINF is universal approximator it handles multimodal q(y) there is no change to the code

Example: 2d Gaussian mixture

β=0.04



β=1



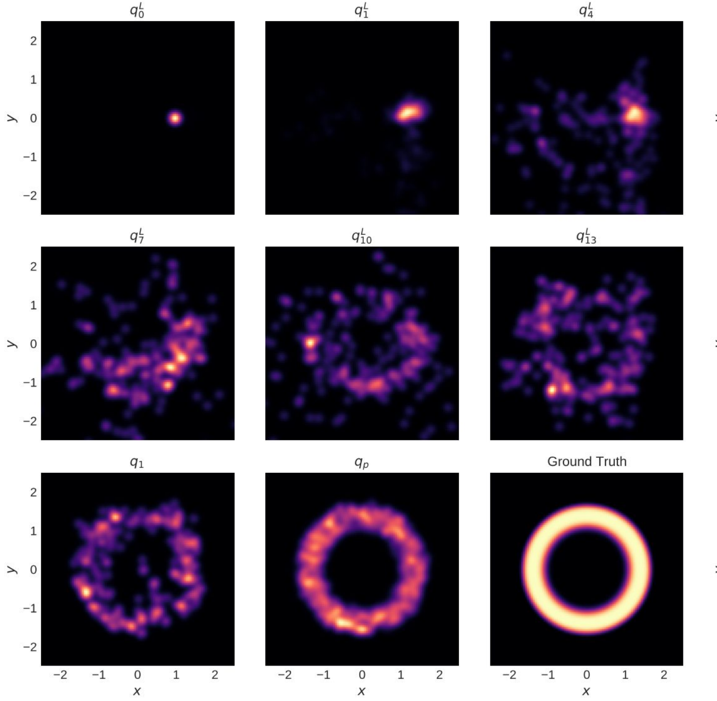
Score based NF Flows

If we have gradient (score) then other NF flows are possible

We are alternating updating particles (samples) anc density q(y). Langevin or Hamiltonian dynamics propagates particles into new regions of previously unseen posterior mass. Then q(y) gets updated, enabling better mixing across current q(y).

For Langevin dynamics we are alternately solving Langevin and Fokker-Planck

Example: donut target starting from a delta function



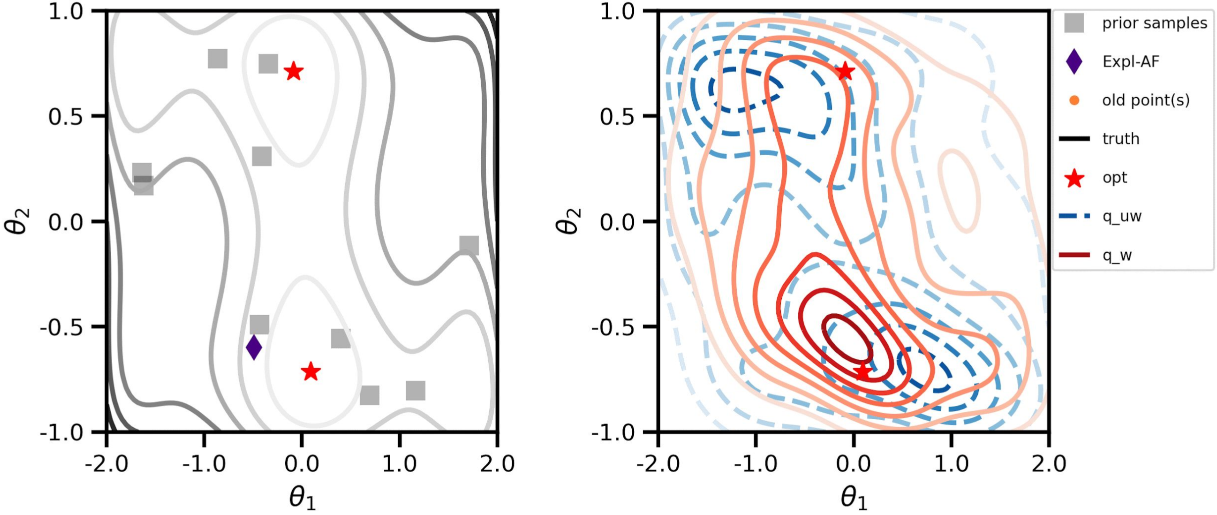
A related problem: global optimization

Global optimization can be a very hard problem to solve when multiple peaks are present in high dim.

It requires a combination of exploitation (going for the peak) and exploration (exploring regions that have not been sampled).

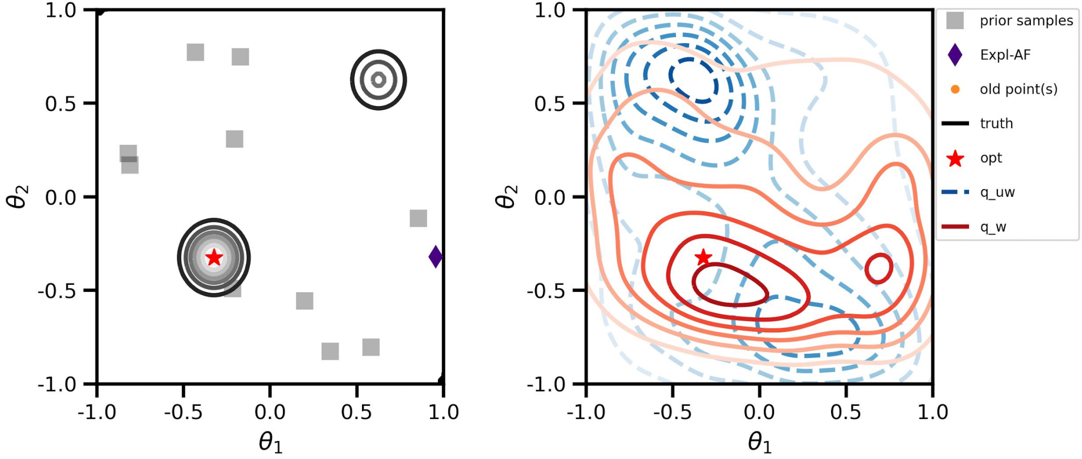
We use Acquisition Function that is ratio of q(y) (red contours) divided by density of samples (which can also be obtained from SINF, blue contours): high proposal value (exploitation) or low local density (exploration)

We go through several temperature levels to beta>>1. Example: six hump camelback (35 calls)



Double Gaussian example in 2d

40 calls



Red contours function we are approximating (surrogate model with SINF)

Blue contours: density of sampling points with SINF

This is competitive with he best GO algorithms (Bayesian Optimization, genetic algorithms etc.)

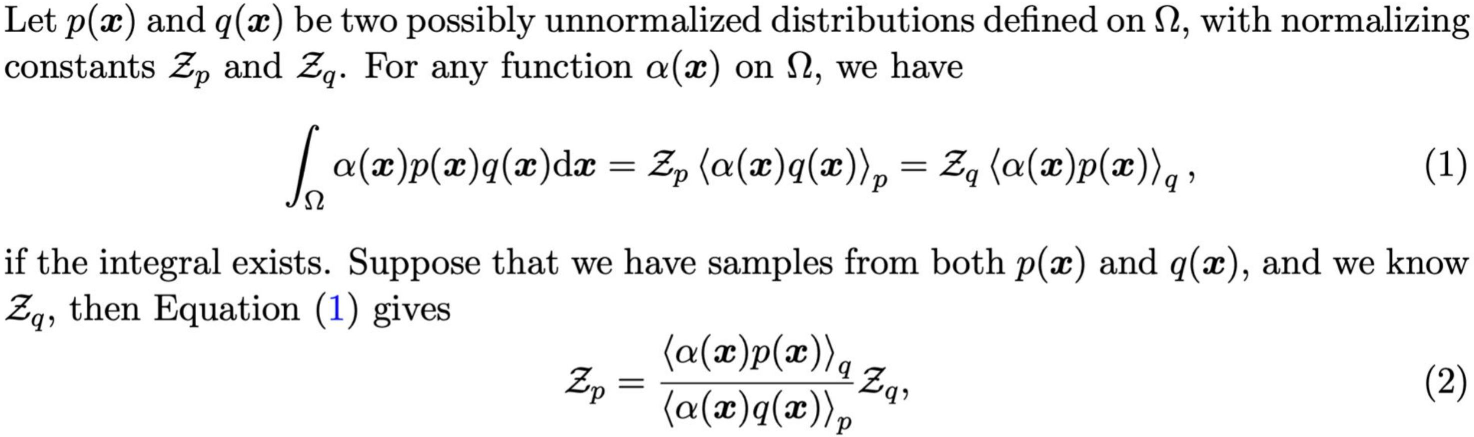
SINF applications to Bayesian inference: evidence

Bayesian Evidence=normalizing constant=marginal likelihood=partition function =integral over the prior of likelihood=average likelihood over the prior

- It is the standard Bayesian method for model selection
- When posterior volume << prior volume evidence is very expensive
- Needs specialized methods: Annealed Importance Sampling or Nested Sampling
- Suppose we have samples already: can we get evidence quickly?

He & Seljak (2019)

SINF fits to samples to obtain normalized q(x). Then GIS (Gaussianized Importance Sampling) uses importance weights (IW) to obtain normalization of p (alpha=1/q), while GBS (Gaussianized Bridge Sampling) adds bridge sampling to optimize on alpha using q and p



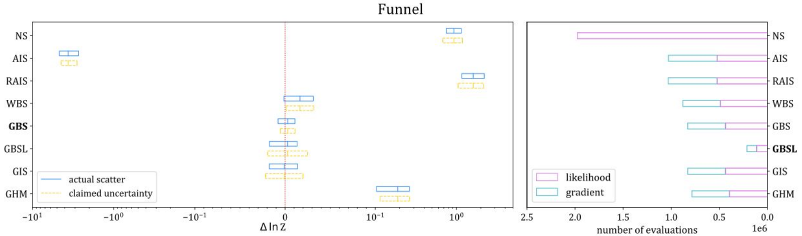
Let $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ be two possibly unnormalized distributions defined on Ω, with normalizing constants $\mathcal{Z}_p$ and $\mathcal{Z}_q$. For any function $\alpha(\boldsymbol{x})$ on Ω, we have

$$\int_\Omega \alpha(\boldsymbol{x})p(\boldsymbol{x})q(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \mathcal{Z}_p \left\langle \alpha(\boldsymbol{x})q(\boldsymbol{x})\right\rangle_p = \mathcal{Z}_q \left\langle \alpha(\boldsymbol{x})p(\boldsymbol{x})\right\rangle_q , \tag{1}$$

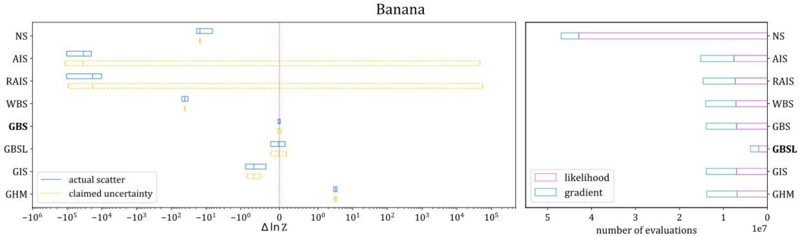
if the integral exists. Suppose that we have samples from both $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$, and we know $\mathcal{Z}_q$, then Equation (1) gives

$$\mathcal{Z}_p = \frac{\left\langle \alpha(\boldsymbol{x})p(\boldsymbol{x})\right\rangle_q}{\left\langle \alpha(\boldsymbol{x})q(\boldsymbol{x})\right\rangle_p}\mathcal{Z}_q, \tag{2}$$

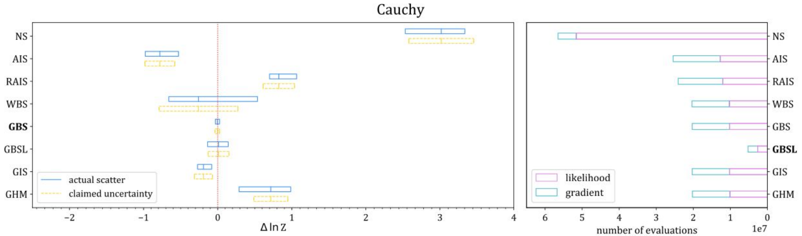
Note that GIS with alpha=1/q does not require p samples



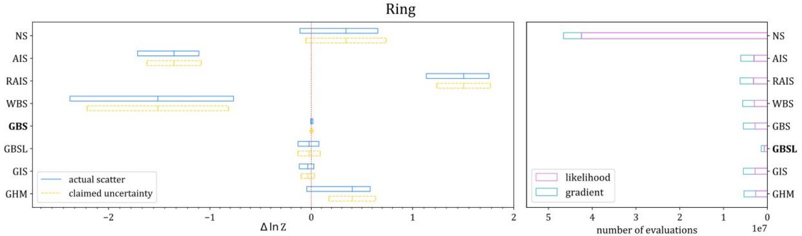
,

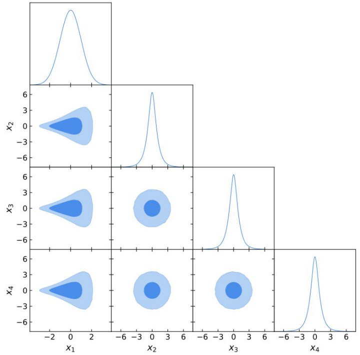


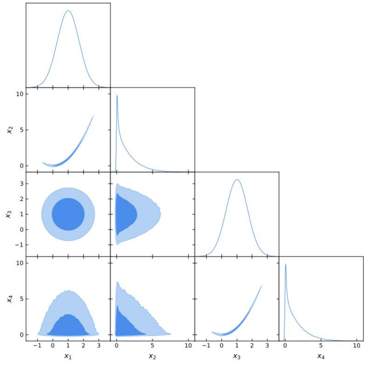
,

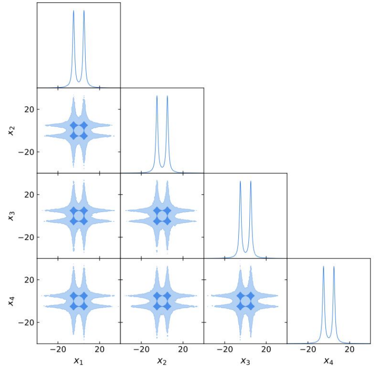


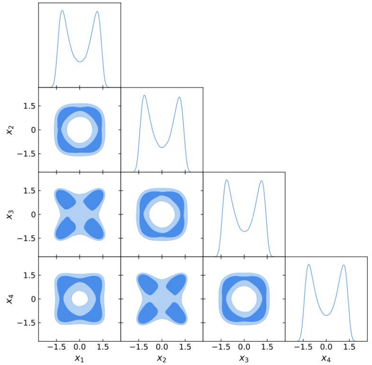
,











These examples assume samples are given (e.g. HMC)
GBS/GIS gives evidence basically at no additional cost

Discussion

- Normalizing Flows are universal approximators of probability distributions, SINF is based on sliced Optimal Transport with good generalization properties.

- Annealing Flow of Normalizing Flow combines optimization and sampling (and MH adjustment), but has very different structure from the Wasserstein flows discussed at this workshop.

- Normalizing Flows can simplify the geometry (preconditioning) and make standard Metropolis-Hastings, Hamiltonian and Langevin sampling more efficient.

- Normalizing Flows also offer another way to get samples that are not from a Markov chain process: samples are independent (**parallelization**). The acceptance rate depends on similarity between the target p(y|x) and NF q(y) on validation data.

- Which is better? Unclear, but NF annealing has a chance to beat standard sampling.It can afford to be quite inefficient and still win, since there is no need to thin the chains, or do leapfrog steps.

- What does it depend on? Number of samples, inductive bias of NF (overfitting?), choice of dissimilarity measure (loss L), use of gradient information in the dissimilarity (e.g. Fisher