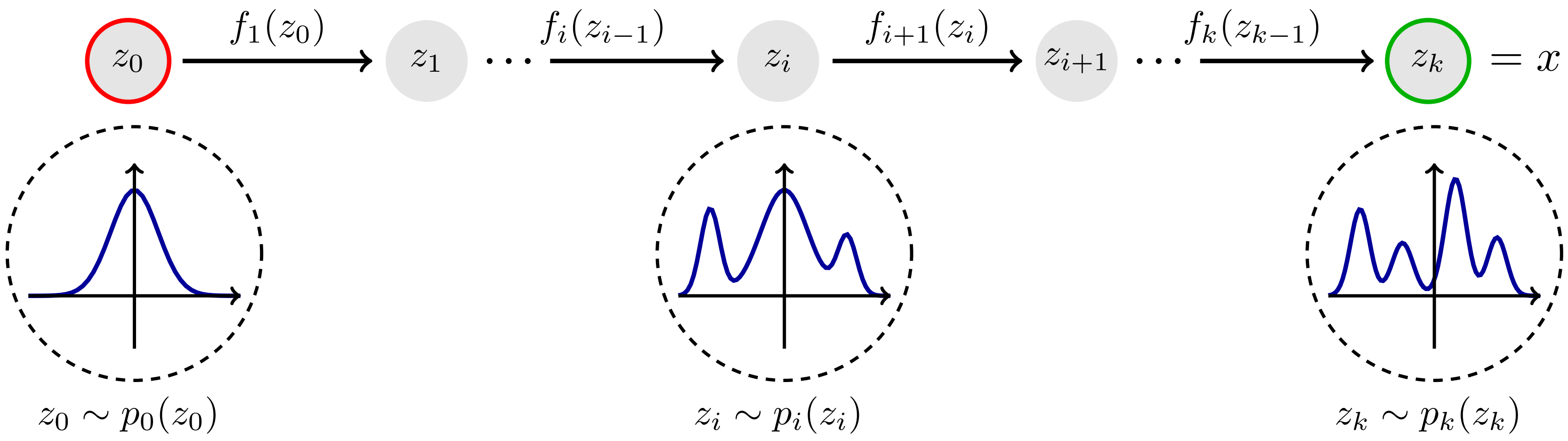
Normalizing Flow: Basics & Applications

Anning Gao 高安宁 2024/05/24



Credit: [6]

OUTLINE

- Definition
- Constructing Flows
- Application
    - Code Example
    - Application in Astronomy

Definition

Objective: Model a unknown distribution $p_{\boldsymbol{x}}$ given a sample $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_n$.

$\boldsymbol{x} \sim p_{\boldsymbol{x}}$ ⟵ Unknown

$\boldsymbol{u} \sim p_{\boldsymbol{u}}(\cdot\,|\boldsymbol{\theta})$ ⟵ Known "base distribution"

$\boldsymbol{x} = T(\boldsymbol{u}|\boldsymbol{\phi})$ $\boldsymbol{\theta}, \boldsymbol{\phi}$: Learnable parameters

$$p_{\boldsymbol{x}}(\boldsymbol{x}) = p_{\boldsymbol{u}}(\boldsymbol{u})\ \left|\det J_T(\boldsymbol{u})\right|^{-1} \qquad \boldsymbol{u} = T^{-1}(\boldsymbol{x})$$

Definition

Requirements:

1. $p_{\boldsymbol{u}}$ is simple

Continuous: (Multivariate) Normal / Uniform

Discrete: Bernoulli / Categorical

2. T is differentiable and **invertible**

Definition

Loss Function:

1. KL Divergence

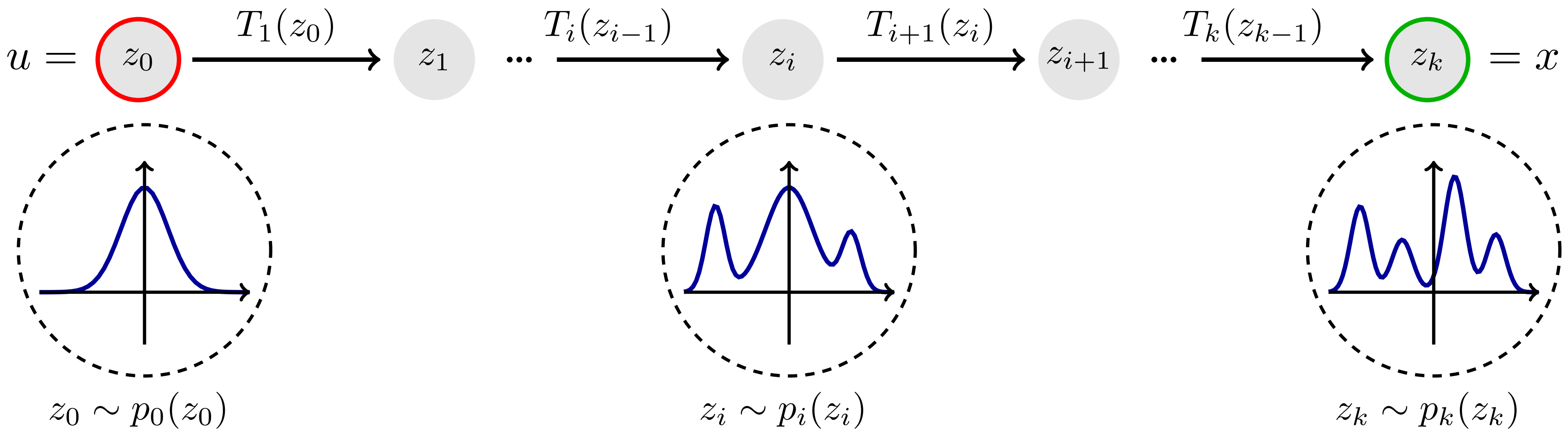
$$KL(p_{\boldsymbol{x}}|p) = \mathbf{E}_{p_x}\left(\log\frac{p_{\boldsymbol{x}}}{p}\right) \approx -\frac{1}{n}\sum_{i=1}^{n}\log p(\boldsymbol{x}_i) + \text{const}$$

2. f-divergence, Integral Probability Metrics, Wasserstein Distance, …

Definition

But where is **Flow**?

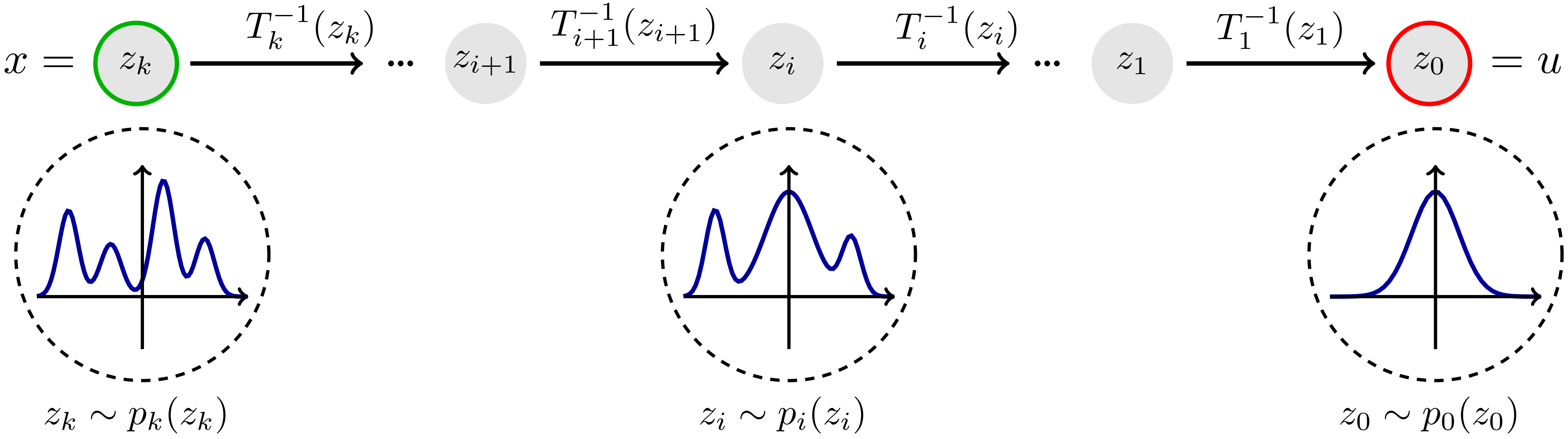
$$T = T_k \circ T_{k-1} \circ \cdots \circ T_1$$



Definition

But where is **Normalizing**?

$$T^{-1} = T_1^{-1} \circ T_2^{-1} \circ \cdots \circ T_k^{-1}$$



Constructing Flows

Autoregressive flows	Transformer type: – Affine – Combination-based – Integration-based – Spline-based	Conditioner type: – Recurrent – Masked – Coupling layer
Linear flows	Permutations Decomposition-based: – PLU – QR Orthogonal: – Exponential map – Cayley map – Householder	
Residual flows	Contractive residual Based on matrix determinant lemma: – Planar – Sylvester – Radial	

Credit: [1]

1. Invertible
2. Expressivity
3. Efficient computation of $\det J_T$

Constructing Flows

1. Linear Flows

$$T(\boldsymbol{u}) = \boldsymbol{A}\boldsymbol{u} + \boldsymbol{b}, \qquad \text{where } \boldsymbol{A} \text{ is invertible}$$

Disadvantages:

- Limited expressiveness
- **Inefficient!** $\boldsymbol{A} \in \mathbf{R}^{D\times D}, \qquad \det \boldsymbol{A} : O(D^3)$

Constructing Flows

1. Linear Flows

$$T(\boldsymbol{u}) = \boldsymbol{A}\boldsymbol{u} + \boldsymbol{b}, \qquad \text{where } \boldsymbol{A} \text{ is invertible}$$

- Diagonal $\boldsymbol{A}$?
- Triangular $\boldsymbol{A}$? With a permutation / orthogonal?
- (P)LU Factorization?

Constructing Flows

2. Autoregressive Flows

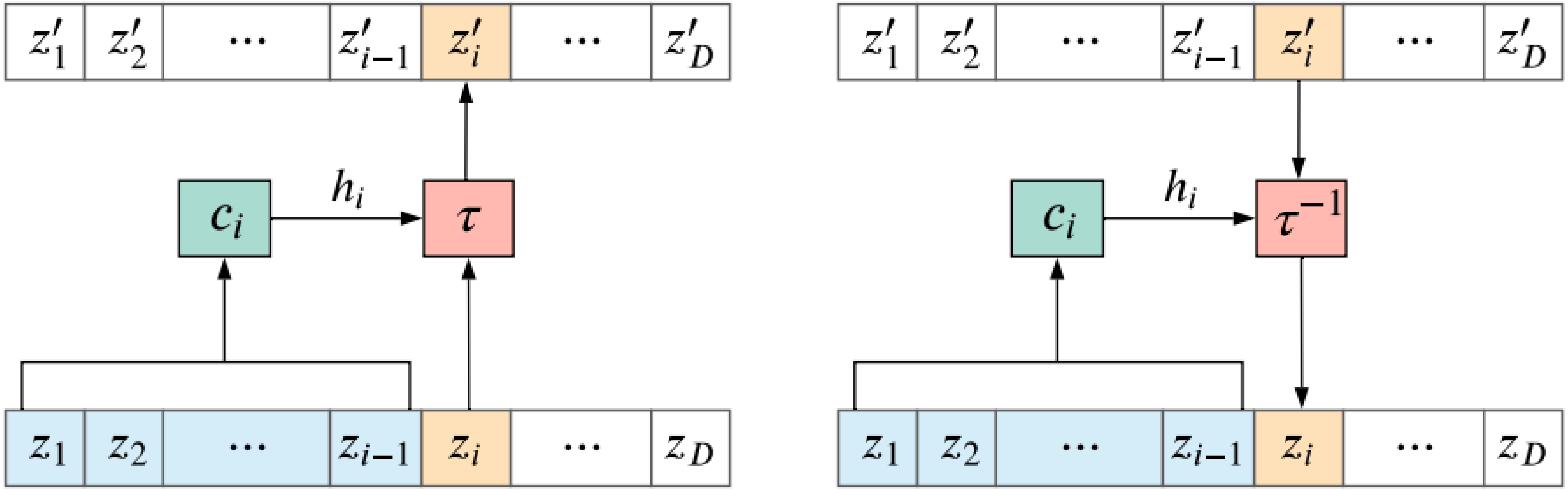
$$\boldsymbol{z}_i' = T(\boldsymbol{z}_i; \boldsymbol{h}_i), \quad \text{where } \boldsymbol{h}_i = C_i(\boldsymbol{z}_{<i})$$

- T: Transformer

- C_i: Conditioner $\longrightarrow$ **Does NOT need to be invertible!**

Constructing Flows

2. Autoregressive Flows



(a) Forward (b) Inverse

Credit: [1]

Constructing Flows

2. Autoregressive Flows

Advantages:

- Universal approximator

- Efficient

$$J_T(\boldsymbol{z}) = \begin{pmatrix} \frac{\partial T}{\partial z_1}(z_1; \boldsymbol{h}_1) & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{L}(\boldsymbol{z}) & \cdots & \frac{\partial T}{\partial z_D}(z_D; \boldsymbol{h}_D) \end{pmatrix}$$

Constructing Flows

2.1 Implementing the Transformer

Affine transformer:

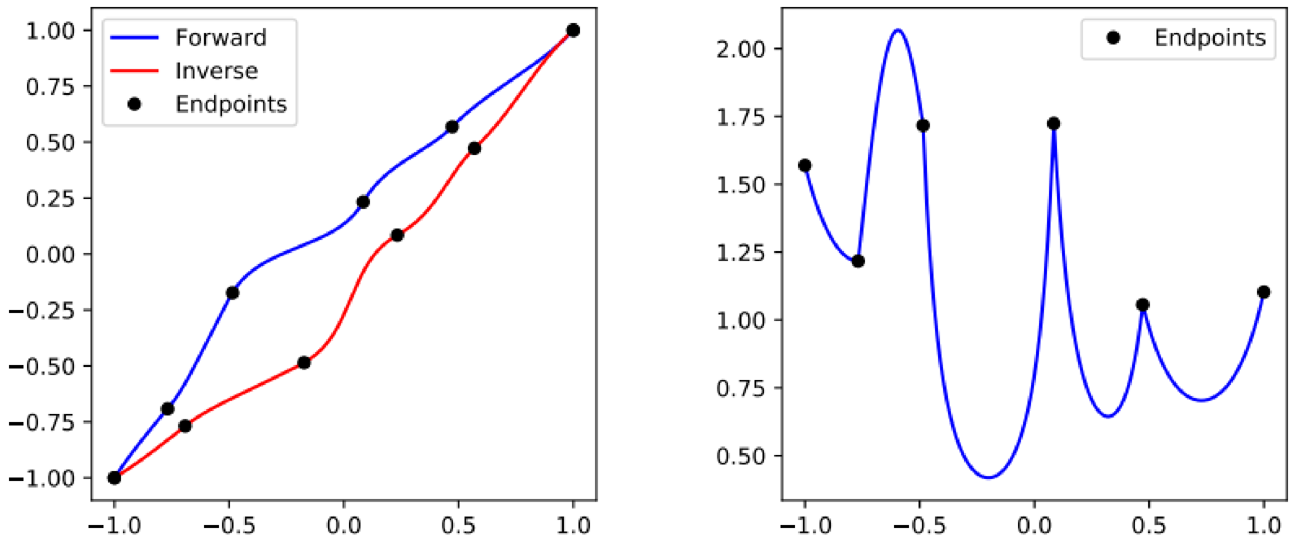
$$T(z_i; \boldsymbol{h}_i) = \alpha_i z_i + \beta_i, \quad \text{where } \boldsymbol{h}_i = \{\alpha_i, \beta_i\}$$

- Limited expressiveness

Constructing Flows

2.1 Implementing the Transformer

Spline-based transformer:



(a) Forward and inverse transformer

(b) Transformer derivative

Credit: [1]

$\boldsymbol{h}_i$: widths & heights of each bin, and the derivatives at internal knots.

Cubic / Linear-Rational / **Rational-Quadratic** / …

Constructing Flows

2.2 Implementing the Conditioner

$$\boldsymbol{z}_i' = T(\boldsymbol{z}_i; \boldsymbol{h}_i), \quad \text{where } \boldsymbol{h}_i = C_i(\boldsymbol{z}_{<i})$$

Model each C_i separately? "Prohibitively expensive" → Share the parameters!

Recurrent conditioner: use a RNN as the conditioner

- Sequential computation, hard to parallelize

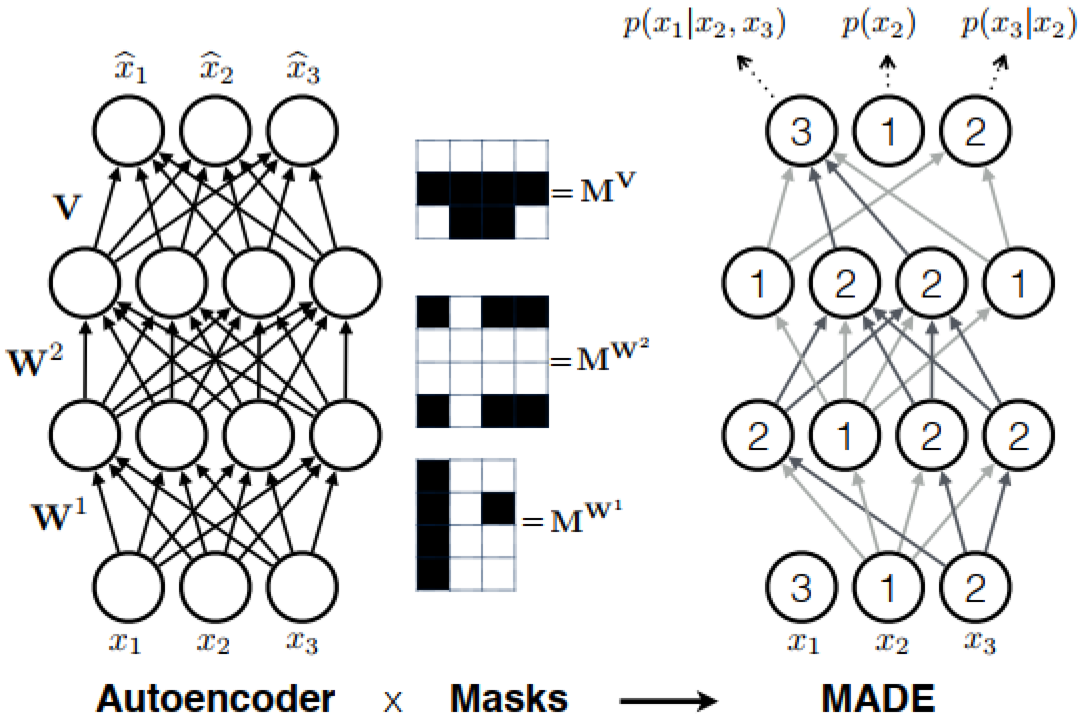
Constructing Flows

2.2 Implementing the Conditioner

Masked conditioner:

Given $\boldsymbol{z}$, output all $\boldsymbol{h}_i$ at once.

- Inefficient to invert.



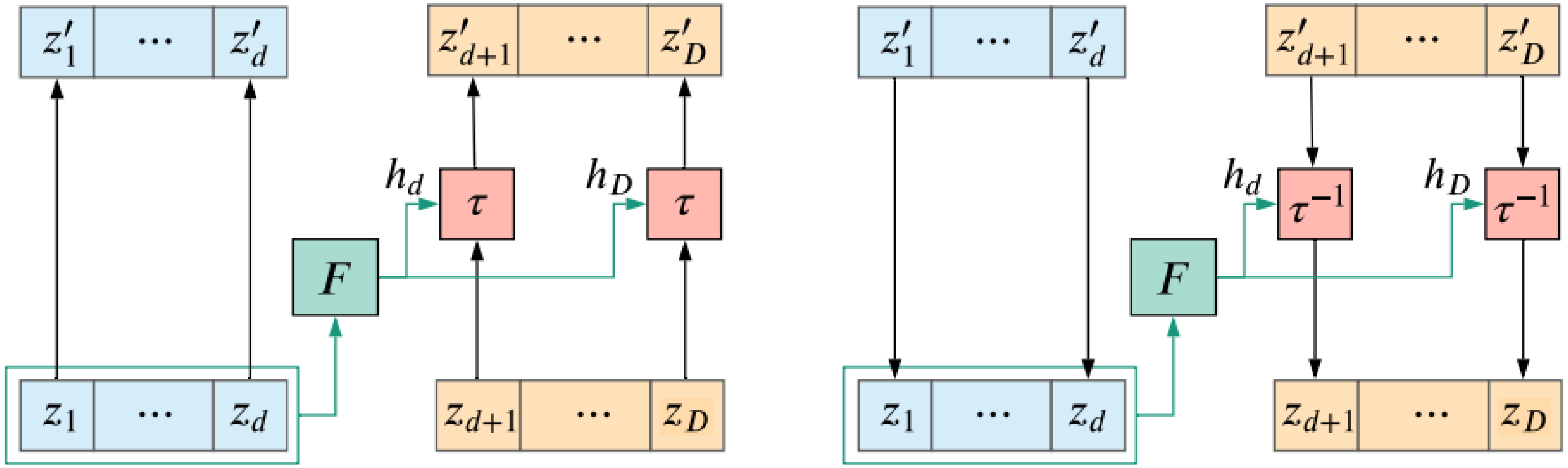
Credit: [5]

Constructing Flows

2.2 Implementing the Conditioner

Coupling layer:

- Reduced expressivity



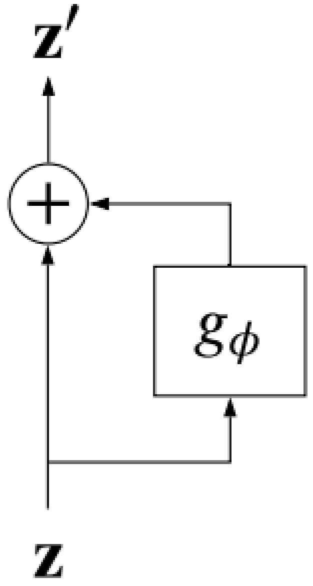
(a) Forward (b) Inverse

Credit: [1]

Constructing Flows

3. Residual Flows

$$\boldsymbol{z}' = \boldsymbol{z} + g_{\boldsymbol{\phi}}(\boldsymbol{z})$$



Credit: [1]

4. Continuous Flows

5. …

Application

Useful NF packages (based on Pytorch):

- `nflows`: https://github.com/bayesiains/nflows/
- `zuko`: https://github.com/probabilists/zuko/

Application

An official example from **`nflows`**:

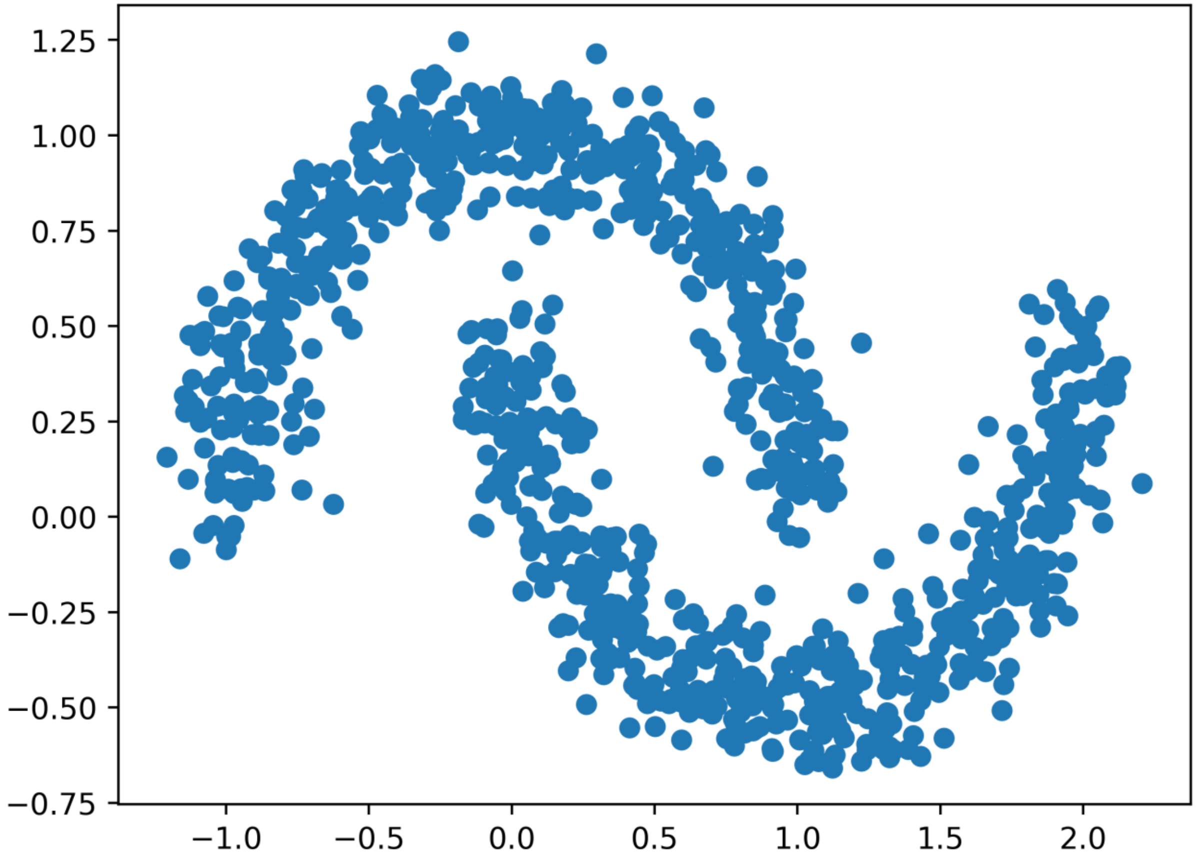
```python
import matplotlib.pyplot as plt
import sklearn.datasets as datasets

import torch
from torch import nn
from torch import optim

from nflows.flows.base import Flow
from nflows.distributions.normal import StandardNormal
from nflows.transforms.base import CompositeTransform
from nflows.transforms.autoregressive import MaskedAffineAutoregressiveTransform
from nflows.transforms.permutations import ReversePermutation

x, y = datasets.make_moons(1024, noise=.1)
plt.scatter(x[:, 0], x[:, 1])

```



Application

Construct the flow:

```
num_layers = 5
base_dist = StandardNormal(shape=[2])

transforms = []
for _ in range(num_layers):
    transforms.append(ReversePermutation(features=2))
    transforms.append(MaskedAffineAutoregressiveTransform(features=2,
                                                          hidden_features=4))
transform = CompositeTransform(transforms)

flow = Flow(transform, base_dist)
optimizer = optim.Adam(flow.parameters())
```

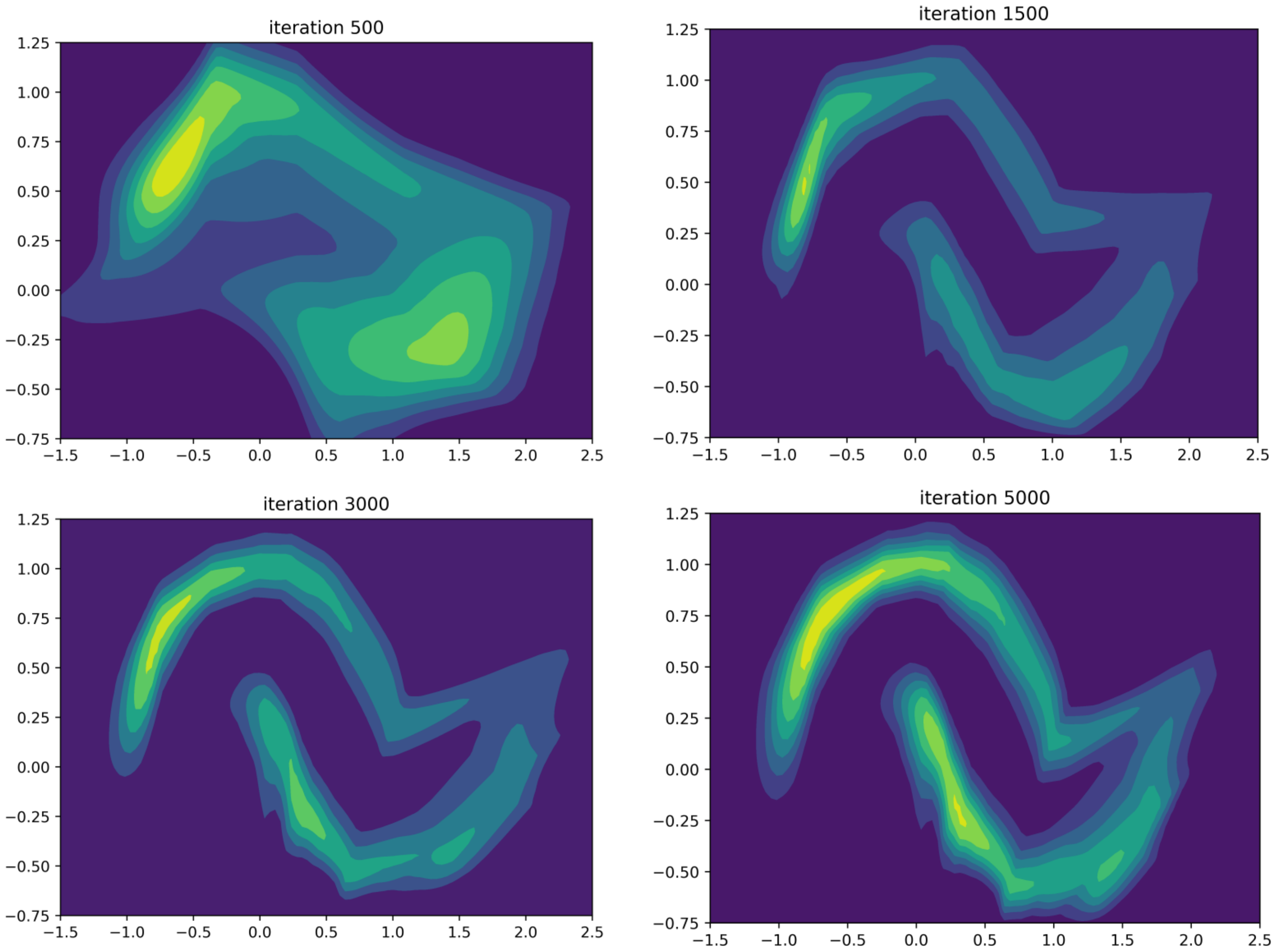
Application

Train the flow:

```python
num_iter = 5000
for i in range(num_iter):
    optimizer.zero_grad()
    loss = -flow.log_prob(inputs=x).mean()
    loss.backward()
    optimizer.step()

```

CPU: ~40s



Application

A **conditional** flow:

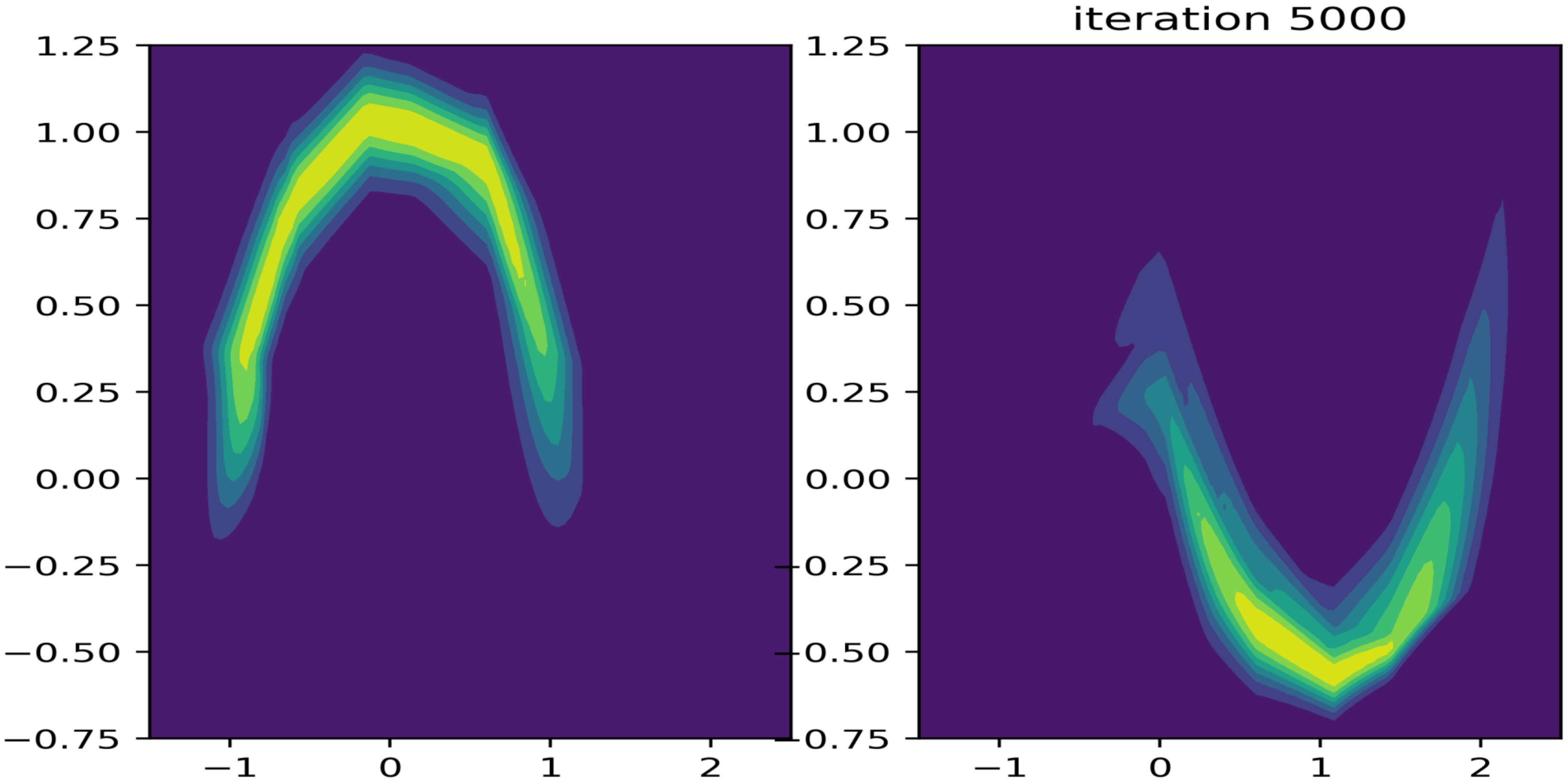
```
num_layers = 5
base_dist = ConditionalDiagonalNormal(shape=[2],
                                      context_encoder=nn.Linear(1, 4))

transforms = []
for _ in range(num_layers):
    transforms.append(ReversePermutation(features=2))
    transforms.append(MaskedAffineAutoregressiveTransform(features=2,
                                                          hidden_features=4,
                                                          context_features=1))
transform = CompositeTransform(transforms)

flow = Flow(transform, base_dist)
optimizer = optim.Adam(flow.parameters())
```

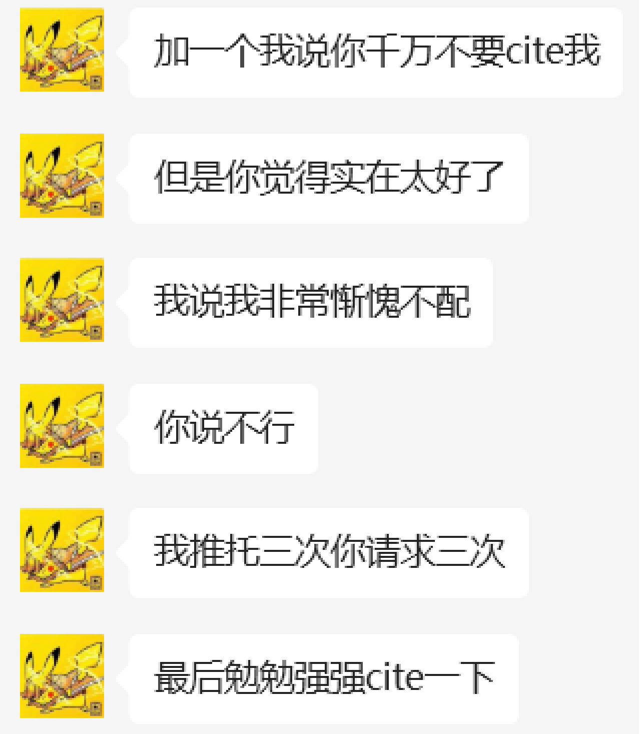
Application

Train the **conditional** flow:



Application

- Photometric Redshift



https://arxiv.org/abs/2310.20125

zephyr : Stitching Heterogeneous Training Data with Normalizing Flows for Photometric Redshift Inference

Zechang Sun (孙泽昌)*
Department of Astronomy
Tsinghua University
Beijing, China 100084
szc22@mails.tsinghua.edu.cn

Joshua S. Speagle (沈佳士)†
Department of Statistical Sciences
University of Toronto
Toronto, ON, Canada
j.speagle@utoronto.ca

Song Huang (黄崧)
Department of Astronomy
Tsinghua University
Beijing, China 100084
shuang@tsinghua.edu.cn

Yuan-Sen Ting (丁源森)‡
Research School of Astronomy & Astrophysics
Australian National University
Cotter Rd., Weston, ACT 2611, Australia
yuan-sen.ting@anu.edu.au

Zheng Cai (蔡峥)
Department of Astronomy
Tsinghua University
Beijing, China 100084
zcai@mail.tsinghua.edu.cn

Abstract

We present zephyr , a novel method that integrates cutting-edge normalizing flow techniques into a mixture density estimation framework, enabling the effective use of heterogeneous training data for photometric redshift inference. Compared to previous methods, zephyr demonstrates enhanced robustness for both point estimation and distribution reconstruction by leveraging normalizing flows for density estimation and incorporating careful uncertainty quantification. Moreover, zephyr offers unique interpretability by explicitly disentangling contributions from multi-source training data, which can facilitate future weak lensing analysis by providing an additional quality assessment. As probabilistic generative deep learning techniques gain increasing prominence in astronomy, zephyr should become an inspiration for handling heterogeneous training data while remaining interpretable and robustly accounting for observational uncertainties.

Application

- Galaxy properties https://ui.adsabs.harvard.edu/abs/2024AJ....167...16L/abstract

PopSED: Population-level Inference for Galaxy Properties from Broadband Photometry with Neural Density Estimation

Jiaxuan Li (李嘉轩)[1], Peter Melchior[1,2], ChangHoon Hahn[1], and Song Huang (黄崧)[3]
[1] Department of Astrophysical Sciences, 4 Ivy Lane, Princeton University, Princeton, NJ 08544, USA; jiaxuanl@princeton.edu
[2] Center for Statistics & machine Learning, Princeton University, Princeton, NJ 08544, USA
[3] Department of Astronomy and Tsinghua Center for Astrophysics, Tsinghua University, Beijing 100084, People's Republic of China
Received 2023 September 29; revised 2023 November 9; accepted 2023 November 10; published 2023 December 11

Abstract

We present POPSED, a framework for the population-level inference of galaxy properties from photometric data. Unlike the traditional approach of first analyzing individual galaxies and then combining the results to determine the physical properties of the entire galaxy population, we directly make the population distribution the inference objective. We train normalizing flows to approximate the population distribution by minimizing the Wasserstein distance between the synthetic photometry of the galaxy population and the observed data. We validate our method using mock observations and apply it to galaxies from the GAMA survey. POPSED reliably recovers the redshift and stellar mass distribution of 10^5 galaxies using broadband photometry within <1 GPU hr, being 10^{5-6} times faster than the traditional spectral energy distribution modeling method. From the population posterior, we also recover the star-forming main sequence for GAMA galaxies at $z < 0.1$. With the unprecedented number of galaxies in upcoming surveys, our method offers an efficient tool for studying galaxy evolution and deriving redshift distributions for cosmological analyses.

Unified Astronomy Thesaurus concepts: Stellar populations (1622); Galaxy photometry (611); Galaxy evolution (594); Neural networks (1933); Astrostatistics (1882); Sky surveys (1464)

Application

- Galaxy properties

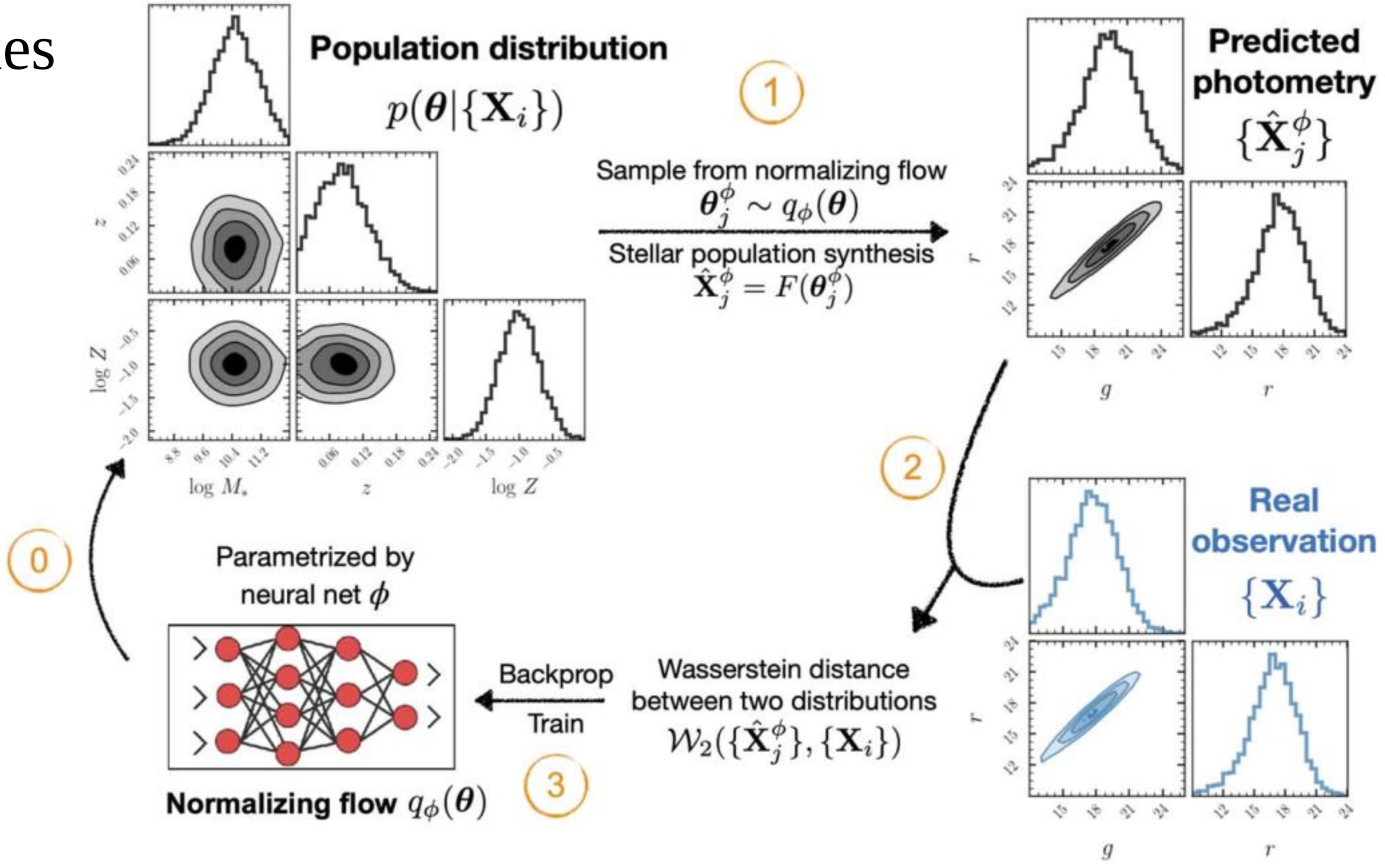


Figure 1. A schematic diagram of POPSED (details in Section 2). The galaxy population distribution $p(\theta|\{X_i\})$ is approximated by a normalizing flow $q_{\phi}(\theta)$. We sample from the normalizing flow and forward model the synthetic photometry $\{\hat{X}_j^{\phi}\}$ using the galaxy SED emulator $F(\theta_j^{\phi})$. Then we compare the distributions of the observed photometry and the synthetic photometry by calculating the Wasserstein distance $\mathcal{W}_2(\{\hat{X}_j^{\phi}\}, \{X_i\})$, which is used as a loss to train the normalizing flow until the synthetic photometry from the normalizing flow agrees with the observed photometry.

Application

- Galaxy properties

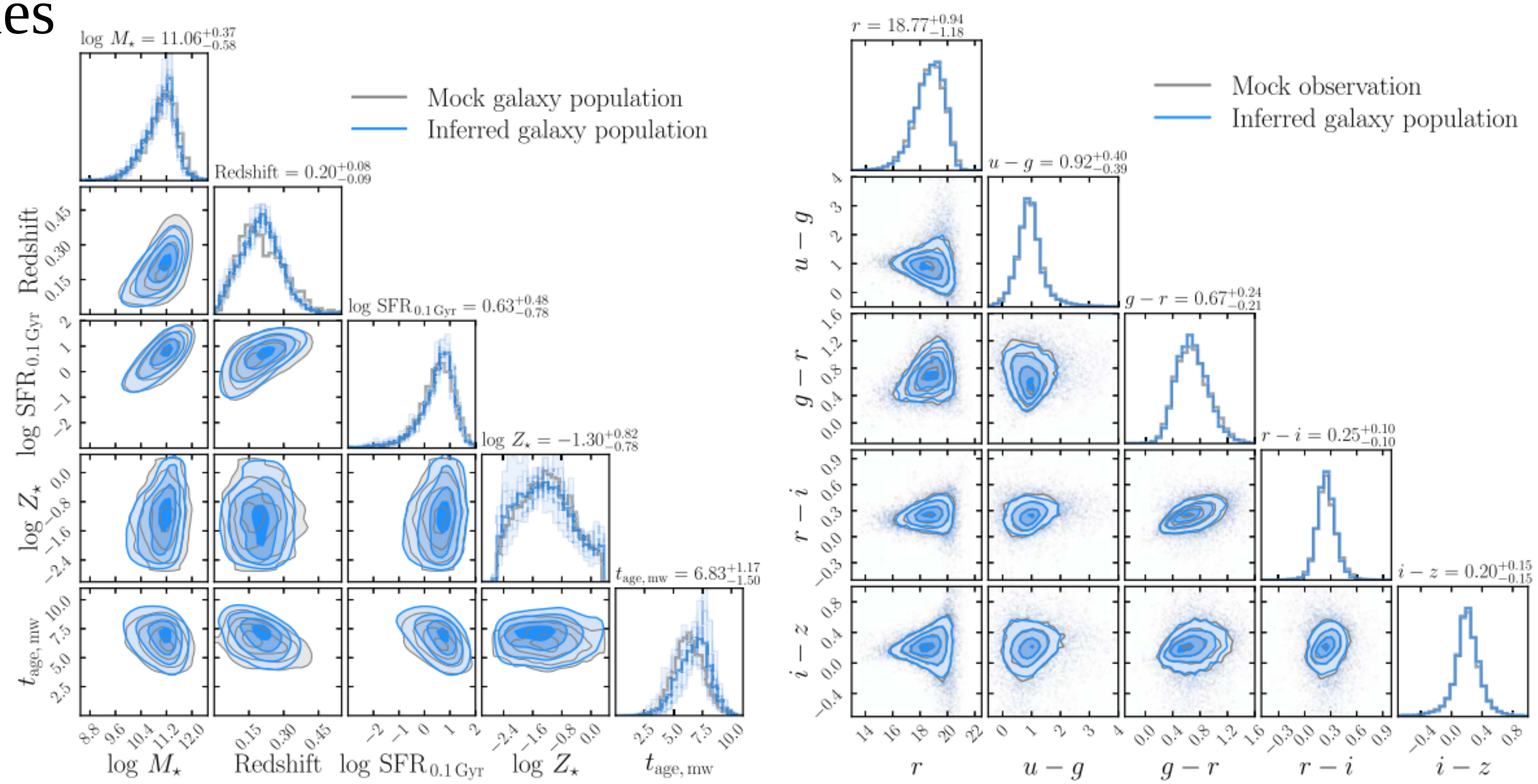


Figure 2. Left: the mock galaxy population (gray contours) and the inferred galaxy population (blue contours) using our method. We calculate the average SFR within the past 0.1 Gyr ($\log\mathrm{SFR}_{0.1\mathrm{Gyr}}$) and the mass-weighted age ($t_{\mathrm{age,MW}}$) using the inferred SPS parameters. The lighter blue histograms show the individual normalizing flows, and the dark blue histogram is the result after averaging 10 flows. The inferred galaxy population agrees with the truth very accurately. Right: the mock photometric data in the SDSS *ugriz* bands (gray contours) are practically indistinguishable from the photometry of the inferred galaxy population using POPSED (blue contours).

Application

- Stellar properties https://arxiv.org/abs/2307.08753

A Novel Application of Conditional Normalizing Flows: Stellar Age Inference with Gyrochronology

Phil Van-Lane [1] **Joshua S. Speagle (沈佳士)** [2 1 3 4] **Stephanie Douglas** [5]

Abstract

Stellar ages are critical building blocks of evolutionary models, but challenging to measure for low mass main sequence stars. An unexplored solution in this regime is the application of probabilistic machine learning methods to gyrochronology, a stellar dating technique that is uniquely well suited for these stars. While accurate *analytical* gyrochronological models have proven challenging to develop, here we apply conditional normalizing flows to photometric data from open star clusters, and demonstrate that a data-driven approach can constrain gyrochronological ages with a precision comparable to other standard techniques. We evaluate the flow results in the context of a Bayesian framework, and show that our inferred ages recover literature values well. This work demonstrates the potential of a probabilistic data-driven solution to widen the applicability of gyrochronological stellar dating.

luminosity parameter space. This is particularly true in the case of non-coeval field stars with weakly constrained cluster membership, where the presence of multiple populations may hinder the identification of isochrones. Other stellar dating methods such as asteroseismology (Cunha et al., 2007; Kurtz, 2022), Lithium abundances (Jeffries, 2014; Beck et al., 2017; Deliyannis et al., 2019) and x-ray luminosity (Beck et al., 2017; Deliyannis et al., 2019) struggle in this regime as well. One technique that avoids such issues is gyrochronology, which relies on the concept of stellar spin-down to infer age using measurements of rotation period and MS location (ie. mass, proxied with an observable such as colour). This makes gyrochronology uniquely well-suited for dating low mass MS stars, especially in the current era of abundant stellar rotation data from missions such as *Kepler* and *Transiting Exoplanet Survey Satellite (TESS)*.

Beginning with several foundational papers by Barnes (2003; 2007; 2010), most gyrochronological studies have leveraged empirical models calibrated on stellar popula-

Application

- Gravitational Wave https://arxiv.org/abs/2405.09475

Robust inference of gravitational wave source parameters in the presence of noise transients using normalizing flows

Chun-Yu Xiong,[1] Tian-Yang Sun,[1] Jing-Fei Zhang,[1] and Xin Zhang[1,2,3,*]

[1]*Key Laboratory of Cosmology and Astrophysics (Liaoning) & College of Sciences, Northeastern University, Shenyang 110819, China*

[2]*Key Laboratory of Data Analytics and Optimization for Smart Industry (Ministry of Education), Northeastern University, Shenyang 110819, China*

[3]*National Frontiers Science Center for Industrial Intelligence and Systems Optimization, Northeastern University, Shenyang 110819, China*

Gravitational wave (GW) detection is of paramount importance in fundamental physics and GW astronomy, yet it presents formidable challenges. One significant challenge is the removal of noise transient artifacts known as "glitches," which greatly impact the search and identification of GWs. Recent research has achieved remarkable results in data denoising, often using effective modeling methods to remove glitches. However, for glitches from uncertain or unknown sources, current methods cannot completely eliminate them from the GW signal. In this work, we leverage the inherent robustness of machine learning to obtain reliable posterior parameter distributions directly from GW data contaminated by glitches. Our network model provides reasonable and rapid parameter inference even in the presence of glitches, without needing to remove them. We also investigate various factors affecting the rationality of parameter inference in our normalizing flow network, including glitch and GW parameters. The results demonstrate that the normalizing flow can reasonably infer the source parameters of GWs even with unknown contamination. We find that the nature of the glitch itself is the only factor that can affect the rationality of the inferred results. With improvements to our model, we anticipate accelerating the localization of electromagnetic counterparts and providing priors for more accurate deglitching, thereby speeding up subsequent data processing procedures.

Take-Home Message

- Definitions
  - Transform from a base distribution
  - Use KL divergence as loss function
- Constructing Flows
  - Linear flows
  - Autoregressive flows
- Application in Astronomy
  - Photometric Redshift Inference
  - Galaxy & Stellar Property Inference
  - Inference with Noise
  - ...

Reference

[1] Papamakarios, George, et al. "Normalizing flows for probabilistic modeling and inference." *Journal of Machine Learning Research* 22.57 (2021): 1-64.

[2] Kobyzev, Ivan, Simon JD Prince, and Marcus A. Brubaker. "Normalizing flows: An introduction and review of current methods." IEEE transactions on pattern analysis and machine intelligence 43.11 (2020): 3964-3979.

[3] Durkan, Conor, et al. "Neural spline flows." Advances in neural information processing systems 32 (2019).

[4] Papamakarios, George, Theo Pavlakou, and Iain Murray. "Masked autoregressive flow for density estimation." Advances in neural information processing systems 30 (2017).

[5] Germain, Mathieu, et al. "Made: Masked autoencoder for distribution estimation." International conference on machine learning. PMLR, 2015.

[6] https://github.com/janosh/awesome-normalizing-flows