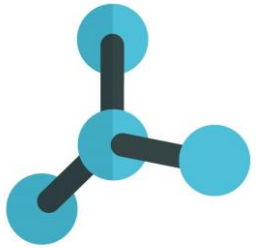
Discrete Tree Flows via Tree Structured Permutations

Mai Elkady*†, Jim Lim*, David I. Inouye

*Equal contribution, †Presenter



Modeling discrete categorical data has multiple applications

- Discrete datasets
  - DNA sequences
  - Medical records
  - Molecular structure (binary)
  - Text
- Applications for modeling discrete distribution
  - Synthetic data generation
  - Compression
  - Detecting outliers

Flow-based approaches use invertible models for **exact** likelihood and sampling

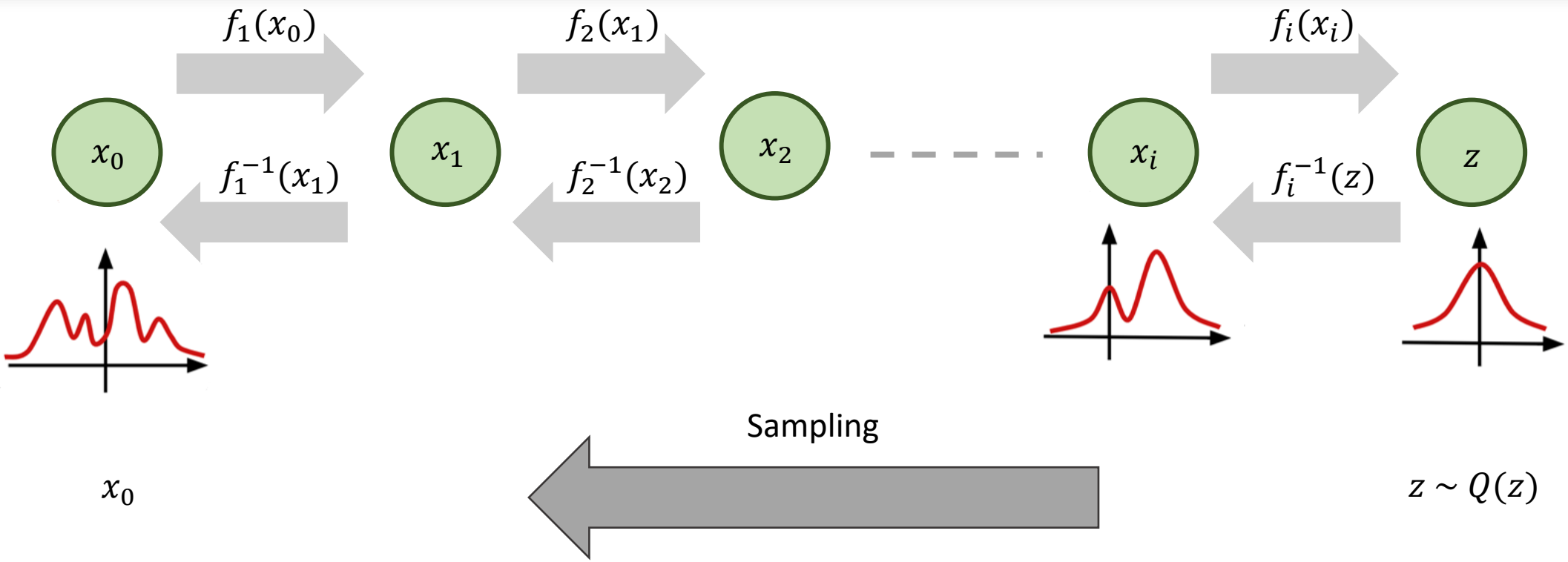


Image modified from Weng, Lilian (2021).

Flow models for *continuous* data compute exact likelihood via change of variables

- Depend on the **change of the variables formula**, for continuous random variables, this is:

$$P(x) = Q(z)\left|\frac{\partial z}{\partial x}\right|$$

Such that:

$z = f(x)$

f: $\mathbb{R}^d \rightarrow \mathbb{R}^d$ which is invertible

When sampling x $= f^{-1}(z)$

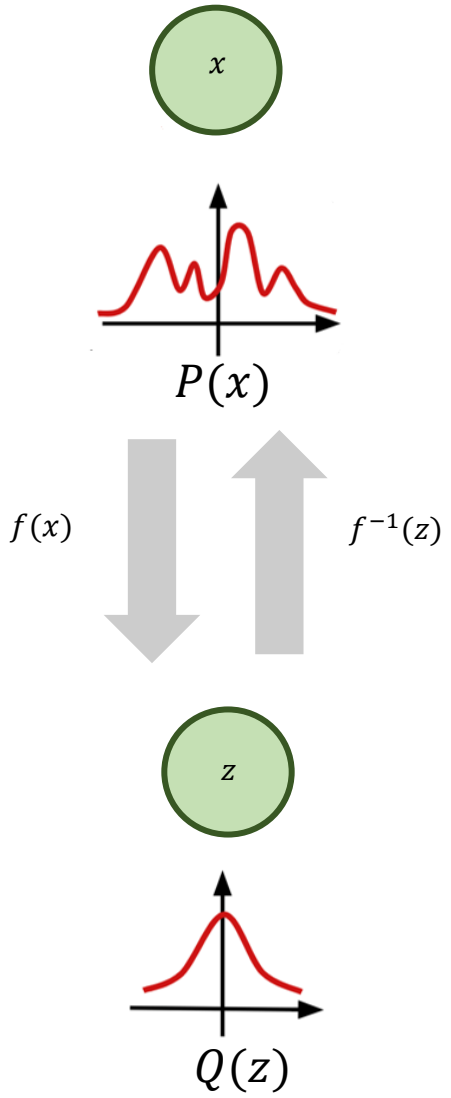


Image modified from Weng, Lilian (2021).

Discrete flow models simplify to a permutation and a base distribution

- For discrete random variables:

$$P(x) = Q(z)$$

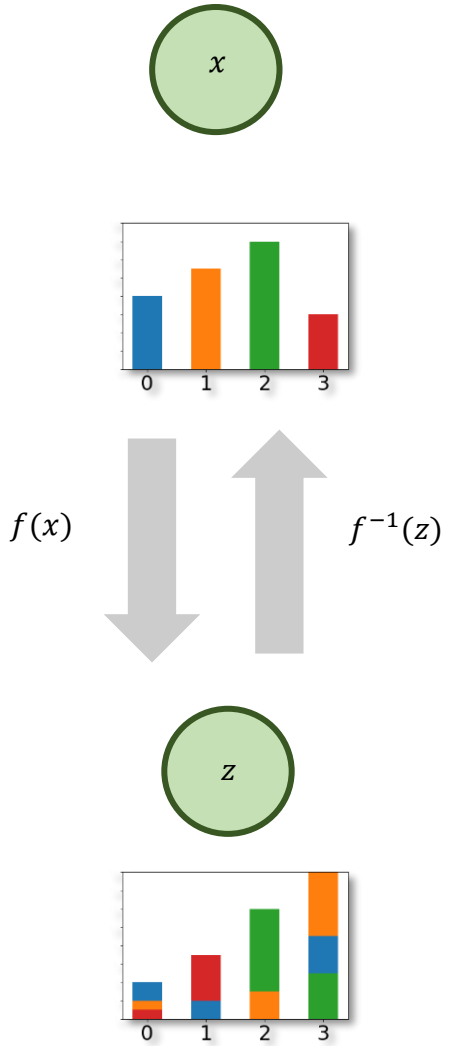
Such that

$$z = f(x)$$

$f\colon \mathbb{Z}^d \rightarrow \mathbb{Z}^d$ is invertible (i.e., a **permutation**)

When sampling x $= f^{-1}(z)$

- The $\left|\frac{\partial z}{\partial x}\right|$ term is 1 because f does not change volume
- Goal: Estimate f and Q
  (i.e., permutation and base distribution)



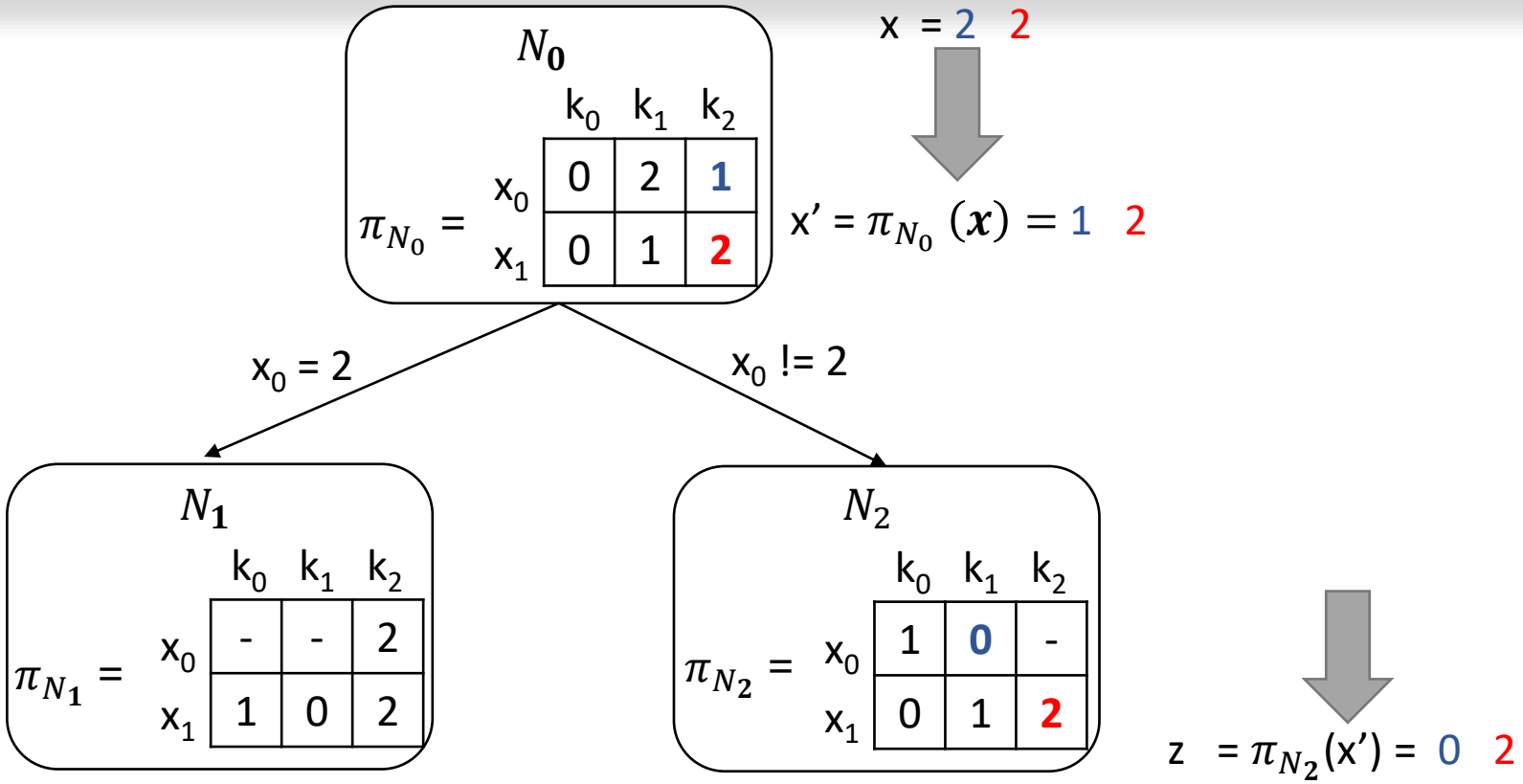
Prior works rely on relaxation or approximation but do not handle on a fundamental level

- Embed into the continuous space
  - Latent Flows by Ziegler & Rush [2019]
  - CNF by Lippe & Gavves [2021]
  - Argmax flows by Hoogeboom et al. [2021]
- Use STEs to approximate gradients
  - Autoregressive Flows and Bipartite Flows by Tran et al. [2019]
  - IDF by Hoogeboom et al. [2019]
  - IDF++ by Van den Berg et al. [2020]
- Use both continuous and discrete learning
  - Discrete Denoising Flows by Lindt & Hoogeboom [2021]

Our approach: Discrete Tree Flows using Tree Structured Permutations

- Utilizes tree-structured permutations (TSPs) for compactly parameterizing a set of permutations
- Discrete Tree Flow (DTF) model is merely a composition of multiple TSPs

Tree structured permutations (TSP) apply **independent** permutations at each node



Theory: Invertibility and universal approximation of DTFs

- TSPs are invertible when a simple condition is satisfied. (Theorem 1)
- A DTF (a composition of TSPs) is a universal approximator of any permutation. (Appendix D)

Learning TSPs in two stages

- Our goal is to minimize the Negative Loglikelihood (NLL) assuming an **independent** base distribution Q_z

$$\underset{\sigma_{\mathcal{T}}}{\operatorname{argmin}} \min_{Q_z} \frac{-1}{n} \sum_{i=1}^{n} \log(Q_z(\sigma_{\mathcal{T}}(x_i)))$$

Q_z: is an independent distribution.

$\sigma_{\mathcal{T}}(x_i)$: is the evaluation of all permutations encountered in $\mathcal{T}$ for the path that $x_i$takes

- Learning is done in 2 parts:
  1. Learn the structure of the tree
  2. Learn the permutations associated with each node

Stage 1: Learn structure of the tree

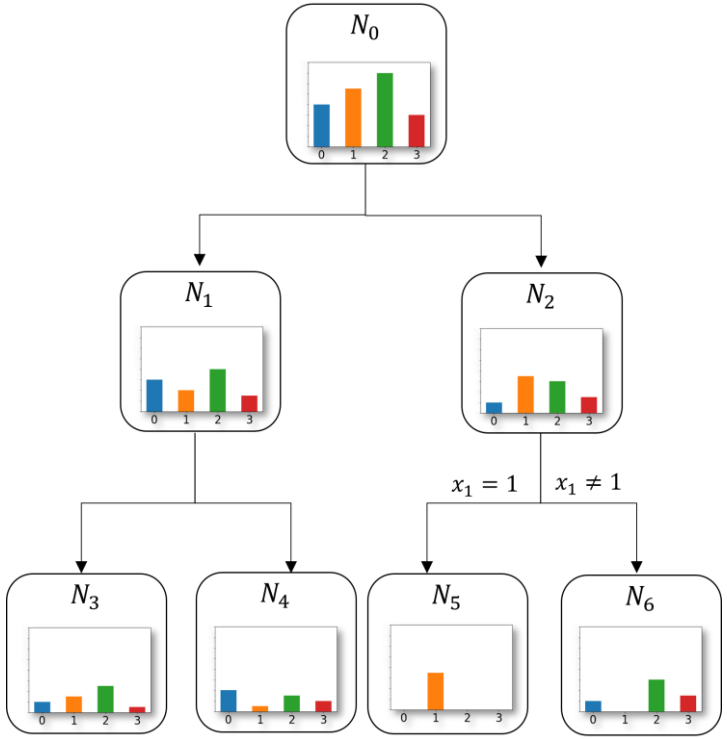
1. RND – Random splitting criteria

2. GLP – Greedy local permutation splitting criteria

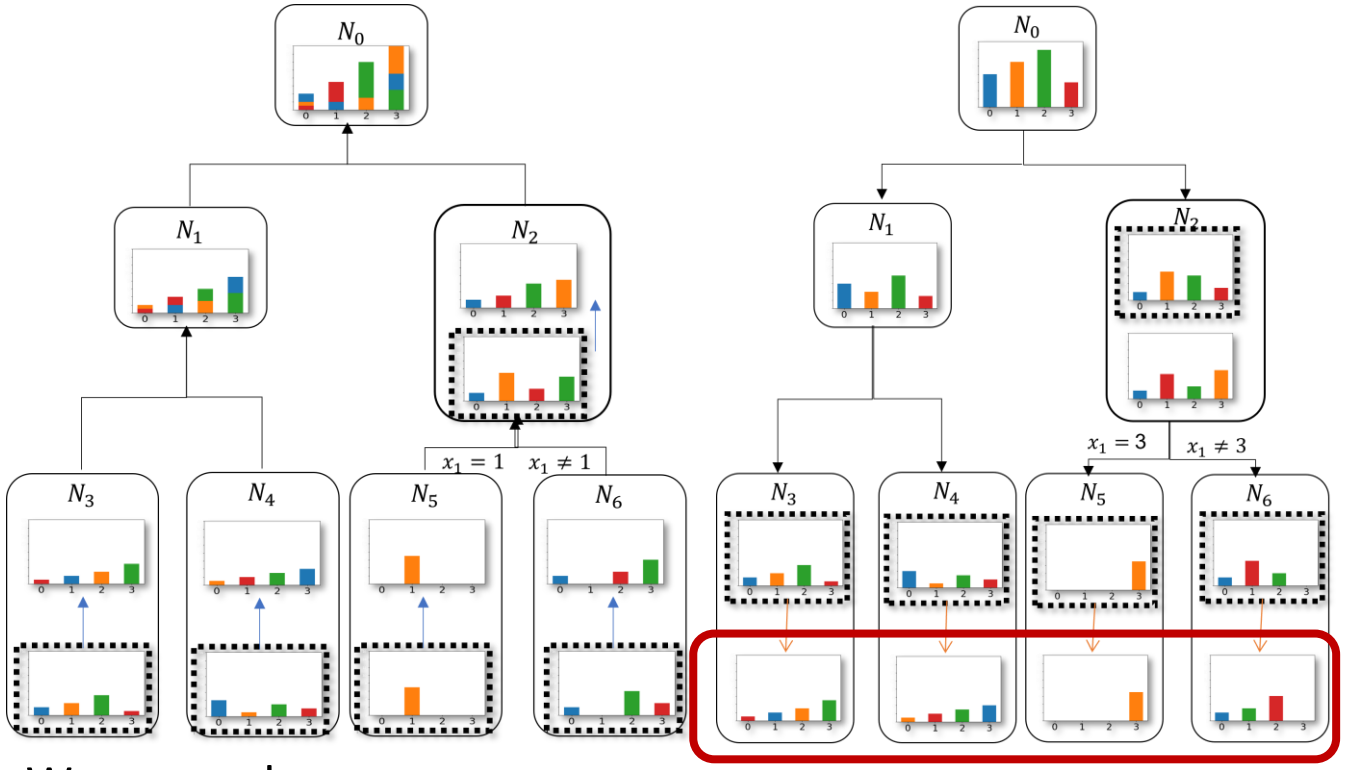
   A heuristic relying on the hypothetical decrease in NLL

Stage 2: Learn node permutations

Stage 1: Learn Tree Structure



Stage 2: Learn node permutations



We prove that:

1) ***rank consistent*** (i.e., sorted counts) TSPs are optimal *given* tree structure.

2) our stage 2 algorithm finds this rank consistent TSP.

Synthetic: DTF is faster than deep learning counterparts with competitive log-likelihood

	AF	BF	DDF	DTF_{GLP}
		8GAUSSIAN		
NLL	6.92 (± 0.06)	7.21 (± 0.09)	**6.42** (± 0.03)	6.5 (± 0.03)
TT	155.9 (± 2.2)	231.6 (± 5.2)	119.8 (± 0.8)	**7.3** (± 0.1)
		COP-H		
NLL	1.53 (± 0.02)	1.47 (± 0.06)	1.46 (± 0.1)	**1.33** (± 0.02)
TT	10.7 (± 0.2)	13.2 (± 0.2)	58.1 (± 1.0)	**≤0.1** (± 0.0)
		COP-M		
NLL	1.76 (± 0.1)	1.62 (± 0.05)	1.51 (± 0.16)	**1.4** (± 0.02)
TT	10.6 (± 0.02)	13.3 (± 0.06)	77.9 (± 1.8)	**≤0.1** (± 0.0)
		COP-W		
NLL	2.42 (± 0.02)	2.35 (± 0.03)	2.29 (± 0.07)	**2.22** (± 0.02)
TT	10.5 (± 0.01)	13.2 (± 0.1)	77.3 (± 1.7)	**≤0.1** (± 0.0)

: Best Performance : Second Best Performance

Experimental Results: Synthetic Datasets

AF	Autoregressive Flows
BF	Bipartite Flows
DDF	Discrete Denoising Flows

Real datasets: RND is fastest while GLP has comparable log-likelihood

	AF	BF	DDF	DTF_{GLP}	DTF_{RND}
Mushroom Dataset					
NLL	24.87 (± 2.28)	23.02 (± 2.3)	19.18 (± 3.48)	**14.15** (± 2.44)	16.66 (± 2.98)
Training Time in sec	29.3 (± 2.0)	20.9 (± 2.7)	175.8 (± 1.9)	9.9 (± 0.2)	**0.5** (± 0.0)
MNIST Dataset					
NLL	206.014 (± 0.32)	205.94 (± 0.26)	**144.78** (± 10.52)	177.75 (± 0.56)	187.44 (± 1.17)
Training Time in sec	12104.6 (± 359.2)	3290.5 (± 13.3)	2909.3 (± 45.4)	5213.7 (± 204.9)	**105.6** (± 0.1)
Genetic Dataset					
NLL	490.55 (± 0.69)	471.54 (± 1.87)	446.86 (± 8.64)	**437.19** (± 1.02)	470.9 (± 6.1)
Training Time in sec	834.0(± 2.1)	251.6 (± 0.5)	209.4 (± 0.6)	411.5 (± 2.3)	**5.9** (± 0.0)

: Best Performance

: Second Best Performance

Thanks!