1-Department of Computer Science, Purdue University, West Lafayette, IN, USA

2-Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA

Discrete Tree Flows via Tree-Structured Permutations

Mai Elkady[1], Jim Lim[2] & David I.Inouye[2]

PURDUE UNIVERSITY

Setting

Problem: Prior flow models for discrete data suffer from limitations, which are distinct from those of continuous flows. These limitations include:

- Straightforward optimization cannot be applied as gradients of discrete functions are undefined or zero.
- Previous models can only approximate discrete functions with pseudo gradients.
- Backpropagation is burdensome compared to alternative discrete algorithms.

Our approach seeks to remove computational burden and the need for pseudo-gradients by developing a discrete flow model based on decision trees.

Approach: We propose a decision tree algorithm based on our own novel tree structure defined as Tree-Structured Permutation (TSP). A TSP encodes a permutation on discrete data allowing for computing density value and sampling new data while also preserving an easy calculable inverse.

Constraints of TSP

Invertibility: To ensure our TSPs are invertible (and thus applicable to discrete flows), we prove that invertiblity is guaranteed if all node permutations $\pi_{\mathcal{N}}$ do not permute configurations that are outside of the node's domain, i.e., $\pi_{\mathcal{N}}(x) = x, \forall x \notin \mathcal{D}(\mathcal{N})$.

Tractability (Naïve TSP):
For the purpose of computational tractability and generalizability we choose to make some restrictions to the classes of permutations that we apply, as listed below:

- Restrict to independent feature-wise permutation which allows each feature to be permuted independently of other features.
- Restrict the class of node permutations be the class of permutations that swaps a single pair of possible categories while holding all other category values constant.

Learning TSPs

We present a greedy approach for building up a TSP where the goal is to minimize the negative log likelihood (NLL) over naïve TSP permutations and independent base distributions Q_z given our dataset $\mathcal{X}$:

$$\min_{Q_z \in \mathcal{Q}_{\text{Ind}}, \pi \in \Pi_{\text{TSP}}} \mathcal{L}(\pi, Q_z) = \min_{Q_z \in \mathcal{Q}_{\text{Ind}}, \pi \in \Pi_{\text{TSP}}} \sum_{i=1}^{n} -\log Q_z(\pi(x_i))$$

where $\mathcal{L}(\pi, Q_z)$ denotes the negative log-likelihood, $\mathcal{Q}_{\text{Ind}}$ is the set of independent distributions over categorical data, and Π_{TSP} is the set of naïve TSP permutations. Our proposed learning algorithm can be decomposed into node-wise subproblems where two main steps are required.

- **Permutation criteria**: To determine the best permutation for each node, we select the best single-feature node permutation π that minimizes the change in the NLL:

$$\min_{s, \pi \in \Pi_{\text{TSP}}(\mathcal{N}, s)} \mathcal{L}(\pi, Q'_z) - \mathcal{L}(\text{id}, Q_z)$$

- **splitting criteria**: To determine the best node to split among all current leaf nodes, we consider using a splitting criteria that maximizes the difference between the factorized distribution on the left and the factorized distribution on the right of the proposed split. We want to maximize the divergence between these two distributions, and for that we used the generalized Jensen-Shannon Divergence (JSD).

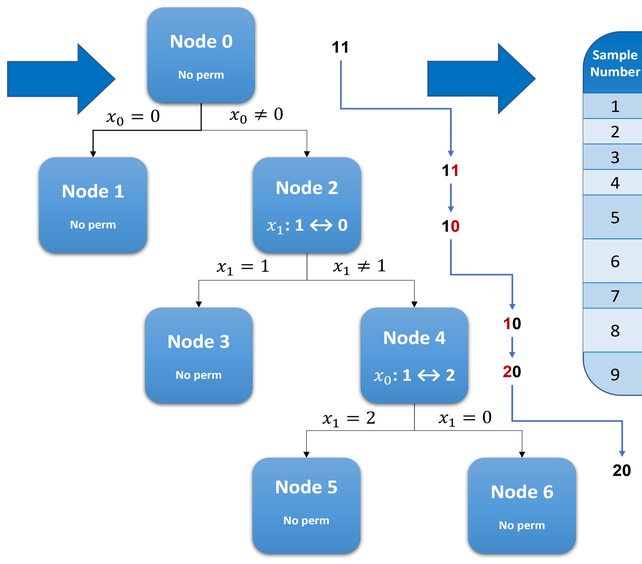
$$\max_{s,v} \text{JSD}(Q^{(s,v)}_{\text{left}}, Q^{(s,v)}_{\text{right}}; \boldsymbol{w}^{(s,v)})$$

$$= H(Q_{parent}) +$$

$$\max_{s,v} [-w_1 H(Q^{(s,v)}_{\text{left}}) - w_2 H(Q^{(s,v)}_{\text{right}})]$$

Tree-Structured Permutations (TSPs)

Sample Number	x_0	x_1
1	0	0
2	0	1
3	0	2
4	1	0
5	1	1
6	1	2
7	2	0
8	2	1
9	2	2



Sample Number	z_0	z_1	Path taken by sample
1	0	0	Node 1
2	0	1	Node 1
3	0	2	Node 1
4	1	1	Node 2, Node 3
5	2	0	Node 2, Node 4, Node 6
6	2	2	Node 2, Node 4, Node 5
7	2	1	Node 2, Node 3
8	1	0	Node 2, Node 4, Node 6
9	1	2	Node 2, Node 4, Node 5

We describe a TSP as a binary decision tree where each node contains both a permutation and a split information. Each TSP node can be defined recursively as follows:

$$f_{\mathcal{N}}(x) = \begin{cases} x & \text{if } \mathcal{N} \text{ is a leaf node} \\ f_{\text{left}(\mathcal{N})}(\pi_{\mathcal{N}}(x)) & \text{if} [\pi_{\mathcal{N}}(x)]_j = v \\ f_{\text{right}(\mathcal{N})}(\pi_{\mathcal{N}}(x)) & \text{otherwise} \end{cases}$$

where $f_{\mathcal{N}}$ is the evaluation of a node, $f_{\text{left}(\mathcal{N})}$ denotes the evaluation of the left child node (and similarly for the right node), $\pi_{\mathcal{N}}$ is the permutation associated with the node, and $[\pi_{\mathcal{N}}(x)]_j = v$ denotes the condition that the j-th feature after the permutation has value v.
An example of a TSP is given on the left.

Experimental Results

We present the negative loglikelihood results averaged across 5 folds for our method (DFT), when compared to the autoregressive flow model (AF), and the Bipratite flow model (BF), for different experimental setups.

Exp details	DFT	AF	BF
Exp = 1 d = 2, k = 2	**1.3333**	1.3338	1.3664
Exp = 2 d = 2, k = 2	**1.2521**	1.2527	1.3291
Exp = 3 d = 5, k = 5	8.0358	8.0301	**8.003**
Exp = 4 d = 10, k = 5	**16.0832**	16.2613	16.1122
Exp = 5 d = 5, k = 10	**11.5008**	11.5739	11.5412
Exp = 6 d = 22, k = 12	**16.6367**	24.9785	27.1999