GENERATIVE FLOW NETWORKS
SIRIUS PROJECT REPORT

Shamil Gabitov, Gleb Korelin, Maxim Soldatov, Karam Suleimanov, Egor Dobshikov
Supervisor: Nikita Morozov

ABSTRACT

Generative Flow Network (GFlowNets) are method for train the policy to sample compositional discrete objects with probabilities proportional to given reward function via sequence of actions. In this paper we compare different methods to train GFlowNet.

1 INTRODUCTION

Generative Flow Networks Bengio et al. (2021) are models created for learn sampler from a complex discrete space according to a given function with unnormalized output.The main thing is what GFlowNet is based on is addition sequental structure of the space: any object can be generated by following a sequence of actions from the initial state. In essense, GFlowNet aims to train the policy to select actions and navigate in this graph. In our educational research project, we compared various methods of training GFlowNet such as Detailed Balance and Trajectory Balance. We conducted numerous experiments with different environments for prediction. In Methods section we describe the structure of GFlowNet, structure of Markovian flows and different loss function for neural network. In Experiments section we compare different ways to learn neural network with various types of environment.

2 METHODS

In this section, we review fundamental definitions and characteristics of graphs, which form the foundation for flow networks and GFlowNets.

A **directed graph** is a couple $G = (S, A)$, where S is a finite set of vertices, and $A \subset V \times V$ representing directed edges. Elements of A are denoted $s \to s'$. A **trajectory** in such a graph is a sequence $\tau = (s_0 \to s_1 \to ... \to s_n)$ of elements of S such that every transition $s_t \to s_{t+1} \in A$ and $n > 0$.

A **directed acyclic graph (DAG)** is a directed graph in which there is no trajectory $\tau = (s_0 \to s_1 \to ... \to s_n)$ satisfying $s_n = s_1$. Let $s_0 \in S$ be the special initial (source) state, the only state with no incoming edges, and designate vertices with no outgoing edges as terminal. A **complete trajectory** is a sequence of transitions $\tau = (s_0 \to s_1 \to ... \to s_n)$ going from the initial state s_0 to a terminal state s_n with $s_t \to s_{t+1} \in A$. Let T be the set of complete trajectories. **Trajectory flow** is any non-negative function $F : T \to \mathbb{R}^+$ defined on the set of complete trajectories T.

The measure of the set of complete trajectories passing through a specific state is represented by the **state flow** $F : S \to \mathbb{R}^+$. Let's define the state flow for vertices and edges:

$$F(s) = \sum_{\tau \in T : s \in \tau} F(\tau) \qquad F(s \to s') = \sum_{\tau \in T : s \to s' \in \tau} F(\tau) \tag{1}$$

As a consequence of this definition, the state flows and edge flows satisfy:

$$F(s) = \sum_{s' \in Child(s)} F(s \to s') \qquad F(s') = \sum_{s \in Par(s')} F(s \to s') \tag{2}$$

The **total flow** Z is the measure of the whole set T , corresponding to the sum of the flows of all the complete trajectories:

$$Z = F(T) = \sum_{\tau \in T} F(\tau) \tag{3}$$

Flow probability is the probability measure P over the measurable space:

$$\forall \tau \in T \qquad P(\tau) = \frac{F(\tau)}{Z} \tag{4}$$

The trajectory flow F is **Markovian flow** if there exist action distributions $P_F(-|s)$ over the children of each nonterminal state s such that the distribution P has a factorization

$$P(\tau = (s_0 \to s_1 \to ... \to s_n)) = \prod_{t=1}^{n} P_F(S_t|s_{t-1}) \tag{5}$$

Equivalently there are distributions $P_B(-|s)$ over the parents of each noninitial state s such that for any terminal x

$$P(\tau = (s_0 \to s_1 \to ... \to s_n)|s_n = x) = \prod_{t=1}^{n} P_B(S_{t-1}|s_t) \tag{6}$$

By definition of the forward transition probability:

$$P_F(s'|s) = \frac{P(s' \to s)}{P(s)} = \frac{F(s' \to s)}{F(s)} \tag{7}$$

$$P_B(s|s') = \frac{P(s' \to s)}{P(s')} = \frac{F(s' \to s)}{F(s')} \tag{8}$$

F, P_B, and P_F jointly correspond to a flow if and only if the **detailed balance conditions** holds

$$\forall s \to s' \in A \qquad F(s)P_F(s'|s) = F(s')P_B(s|s') \tag{9}$$

We now consider the general class of problems where some constraints or preferences over flows are given. Our goal is to find functions such as the state flow function $F(s)$ or the transition probability function $P(\to s'|s)$ that best match these desiderata. Such learning machines are called Generative Flow Networks. We focus on scenarios where we are given a target reward function $R : X \to \mathbb{R}^+$, and aim at estimating flows F that satisfy:

$$\forall x \in X \qquad F(x) = R(x) \tag{10}$$

Detailed balance objective

A neural network model, characterized by parameters θ, takes an input s and produces three types of outputs: an estimated state flow $F_\theta(s)$, an estimated distribution over children $P_F(-|s;\theta)$, and an estimated distribution over parents $P_B(-|s;\theta)$. The policy $P_F(-|-;\theta)$ and the initial state flow $F_\theta(s_0)$ together determine a Markovian flow F_θ, which may not be compatible with the estimated backward policy $P_B(-|-;\theta)$. The error in satisfying the detailed balance constraint is minimized by optimizing actions $(s \to s')$ taken between nonterminal nodes observed during trajectories sampled from the training policy.

$$\lambda_{DB}(s,s') = \left(\log \frac{F_\theta(s)P_F(s'|s;\theta)}{F_\theta(s')P_B(s|s';\theta)}\right)^2 \qquad \lambda'_{DB}(s,s') = \left(\log \frac{F_\theta(s)P_F(s'|s;\theta)}{R(s')P_B(s|s';\theta)}\right)^2 \tag{11}$$

The parameters are updated with stochastic gradient

$$E_{(s_0\to s_1\to\ldots\to s_n)\sim\pi_\theta}\nabla_\theta\left[\sum_{t=1}^{n-1}\lambda_{DB}(s_{t-1},s_t)+\lambda'_{DB}(s_{n-1},s_n)\right] \tag{12}$$

A GFlowNet is specified by finding the global minimum of the expected detailed balance loss under a training policy π_θ, which samples trajectories.This means that the flow F_θ generated by the GFlowNet will sample from the target distribution.

Trajectory balance

The **trajectory balance constraint** for any complete trajectory can be obtained by performing a direct algebraic manipulation of equations (3), (4), (5), and (6), where F is a Markovian flow, P is the corresponding distribution over complete trajectories, and P_F and P_B are the forward and backward policies determined by F.

$$Z\prod_{t=1}^{n}P_F(s_t|s_{t-1}) = F(x)\prod_{t=1}^{n}P_B(s_{t-1}|s_t) \tag{13}$$

Trajectory balance as an objective

Our suggestion is to transform equation (13) into an objective function that can be optimized using trajectories generated from a training policy. Assuming we have a model with parameters θ that produces estimated forward policy $P_F(-|s;\theta)$ and backward policy $P_B(-|s;\theta)$ for states s, as well as a global scalar Z_θ that estimates $F_{(s0)}$, the scalar Z_θ and the forward policy $P_F(-|-;\theta)$ together determine an implicit Markovian flow F_θ.

For a trajectory $\tau = (s_0 \to s_1 \to ... \to s_n = x)$, define the **trajectory loss**

$$\lambda_{TB}(\tau) = \left(\log \frac{Z_0\prod_{t=1}^{n}P_F(s_t|st-1;\theta)}{R(x)\prod_{t=1}^{n}P_B(s_{t-1}|st;\theta)}\right)^2 \tag{14}$$

If π_θ is a training policy – usually that given by $P_F(-|-;\theta)$ or a tempered version of it – then the trajectory loss is updated along trajectories sampled from π_θ, i.e., with stochastic gradient

$$E_{\tau\sim\pi_\theta}\nabla_\theta\lambda_{TB}(\tau) \tag{15}$$

3 EXPERIMENTS

3.1 HYPERGRID ENVIRONMENT

In this subsection, we study a synthetic hypergrid environment. In this environment, the nonterminal states S° form a D-dimensional hypergrid with side length H

$$S^\circ = \{(s^1,...,s^D)\,|s^D \in \{0,1,...,H-1\}, d = 1,...,D\}$$

and actions are operations of incrementing one coordinate in a state by 1 without exiting the grid. The initial state is (0, . . . , 0). For every nonterminal state s, there is also a termination action that

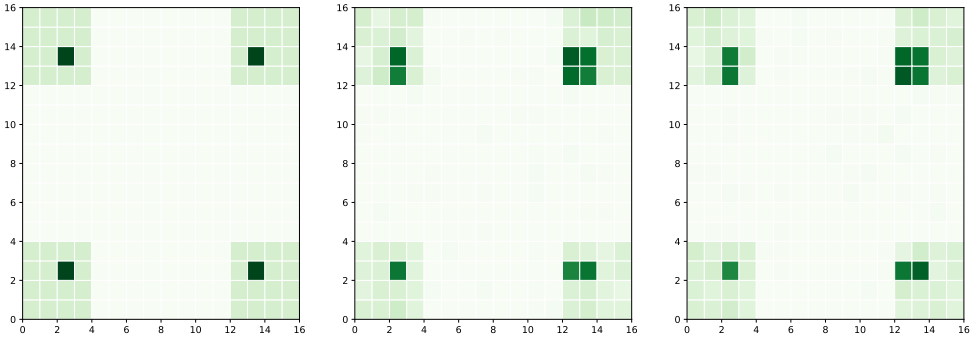

Figure 1: Hypergrid 1-rewards, 2-DB distribution, 3-TB distribution

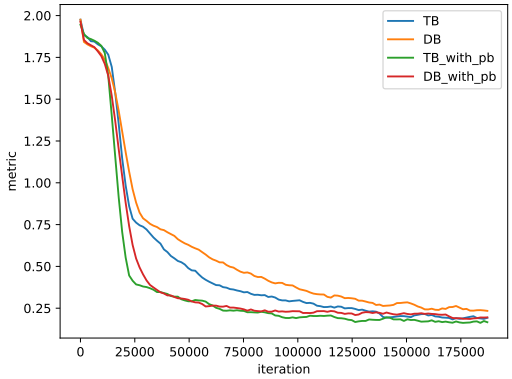

Figure 2: $R0 = 0.001$, $H = 8$, $D = 4$

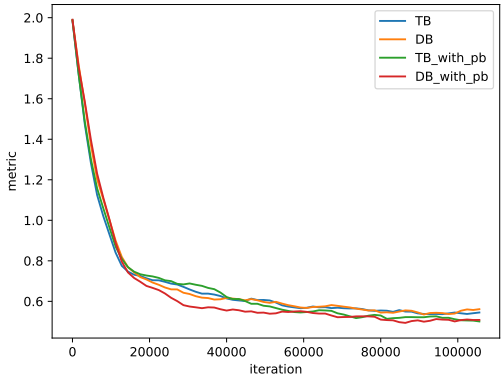

Figure 3: $R_0 = 0.1$, $H = 8$, $D = 4$

transitions to a corresponding terminal state s^T. The reward at a terminal state $s^T = (s^1, ..., s^D)^T$ is given by

$$R(s^T) = R_0 + 0.5 \prod_{d=1}^{D} \mathbb{I}\left[\left|\frac{s^d}{H-1} - 0.5\right| \in (0.25, 0.5]\right] + 2 \prod_{d=1}^{D} \mathbb{I}\left[\left|\frac{s^d}{H-1} - 0.5\right| \in (0.3, 0.4)\right]$$

Where $\mathbb{I}$ is an indicator function and R_0 is a constant controlling the difficulty of exploration. This reward has peaks of height $2.5 + R_0$ near the corners of the hypergrid, surrounded by plateaux of height $0.5 + R_0$. These plateaux are separated by wide troughs with reward R_0. An illustration with $H = 16$ and $D = 2$. This environment evaluates the ability of a GFlowNet to generalize from visited states to infer the existence of yet-unvisited modes. In Figure 1, we visualize the learned distribution and the actual reward.

Our neural network has 4 types: with Trajectory balance equality with learning backward policy (TB with pb in legend) and with fixed (TB in legend) and with Detailed balance equality also with fixed backward policy (DB in legend) and with learning (DB with pb in legend). The experimental results are presented in Figure 2 and Figure 3. We noticed that in different situations different methods can be optimal. For example with R0 = 0.1 and grid D = 4 and H = 8 DB with learning pb shows the best results, TB with learning pb a bit worse, symmetric situation we can see with R0 = 0.001: TB with pb shows the best results, BD with pb a bit worse. But TB with fixed policy backward has more determined comparing with DB results: on all our graphics TB shows better results than DB on long distance. We can suppose, that difference between DB with learning pb and TB with learning pb are bonded with their unstable behavior, that appear by learning pb. So in dependence from situation should choose DB with learning pb or TB with learning pb, but simple TB always works better.

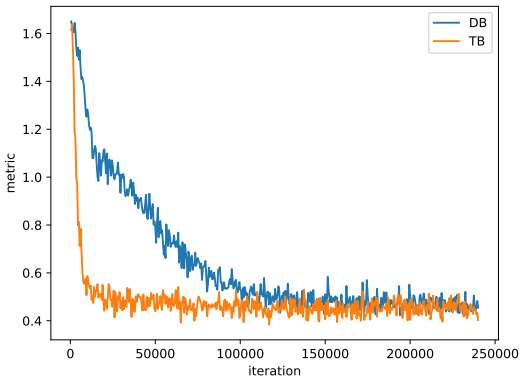

Figure 4: Inversions with $N = 6$.

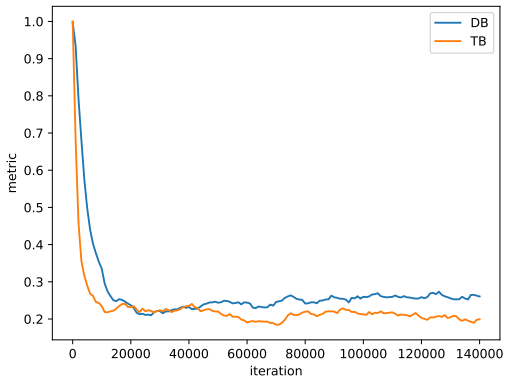

Figure 5: Inversions with $N = 9$.

3.2 Inversions in permutations

Task. The goal is to generate permutations of a fixed length N, where reward is defined as $R(a) = exp(-F(a))$, where $F(a)$ is the count of inversions in a, where a is some permutation of length N.

We built a graph in which the states are some incomplete permutations (permutations for which we have already set values in some positions, but not yet in others), and transitions are set value in some free position.

For the GFlowNet policy model, we use an MLP with 1 hidden layer of 100 hidden units. We train all models with a learning rate of 0.01 (P_F) and 0.1 (Z_θ).

First experiment. In first case we sampled $240k$ permutations with Detailed Balance (DB) and Trajectory Balance (TB) and compared results with empirical error. We noticed that TB learns faster than DB, and that they both converge to values around 0.5. We present the results in Figure 4.

Second experiment. In second case we sampled $140k$ permutations with Detailed Balance (DB) and Trajectory Balance (TB) and compared results with empirical error. We noticed that TB learns faster than DB, and that TB converge to values around 0.2 and DB converge to values around 0.26. We present the results in Figure 5.

Result. As a result of both experiments, we noticed that in this task TB is better.

4 Conclusion

In conclusion we can say that DB is better than TB in general, because in our experiments it was better in 3 of 4 cases. We also noticed that methods with learning P_B were better in all our experiments.

References

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.