

Online Structure Learning of Probabilistic Circuits

Stefan Wagner¹ Anish Kuttetira¹ YooJung Choi¹

¹School of Computing and Augmented Intelligence, Arizona State University

Motivation

Probabilistic circuits (PCs) recursively compose sum, product, and univariate leaf nodes. Their structural constraints enable exact and efficient probabilistic inference.

- **Decomposability and smoothness** support tractable marginal and conditional inference.
- **Determinism** enables tractable MAP, entropy, and closed-form maximum likelihood parameter estimation.
- **Structured decomposability** makes KL-based comparisons between circuits tractable.

Existing deterministic PC structural learners, like *STRUDEL* [1] or *LearnPSDD* [2] are **offline**, but real data is rarely stationary.

Goal: Learn a deterministic PC directly from a data stream, **without** complete training history.

Main Contributions

- **Online structure learning** in PCs that preserves determinism and structured decomposability throughout training.
- **Distribution-preserving conditional splits** initialized from maintained flow statistics.
- **Divergence-guided merging** to revise structure and control growth.

Flows as Statistics

We use flows to guide our structural manipulations and maintain historical information. We specifically keep track of **aggregate**, **marginal**, and **joint flows** for every edge $e_{i,j}$.

$$a(i, j) = \sum_{\mathbf{x}} \mathbb{I}[e_{i,j} \text{ active}],$$

$$m_u(i, j) = \sum_{\mathbf{x}} \mathbb{I}[e_{i,j} \text{ active}] x_u,$$

$$J_{u,v}(i, j) = \sum_{\mathbf{x}} \mathbb{I}[e_{i,j} \text{ active}] x_u x_v.$$

Aggregate flows allow us to compute our edge weights in closed form due to determinism:

$$\theta_{i,j} = \frac{a(i, j)}{\sum_k a(i, k)}.$$

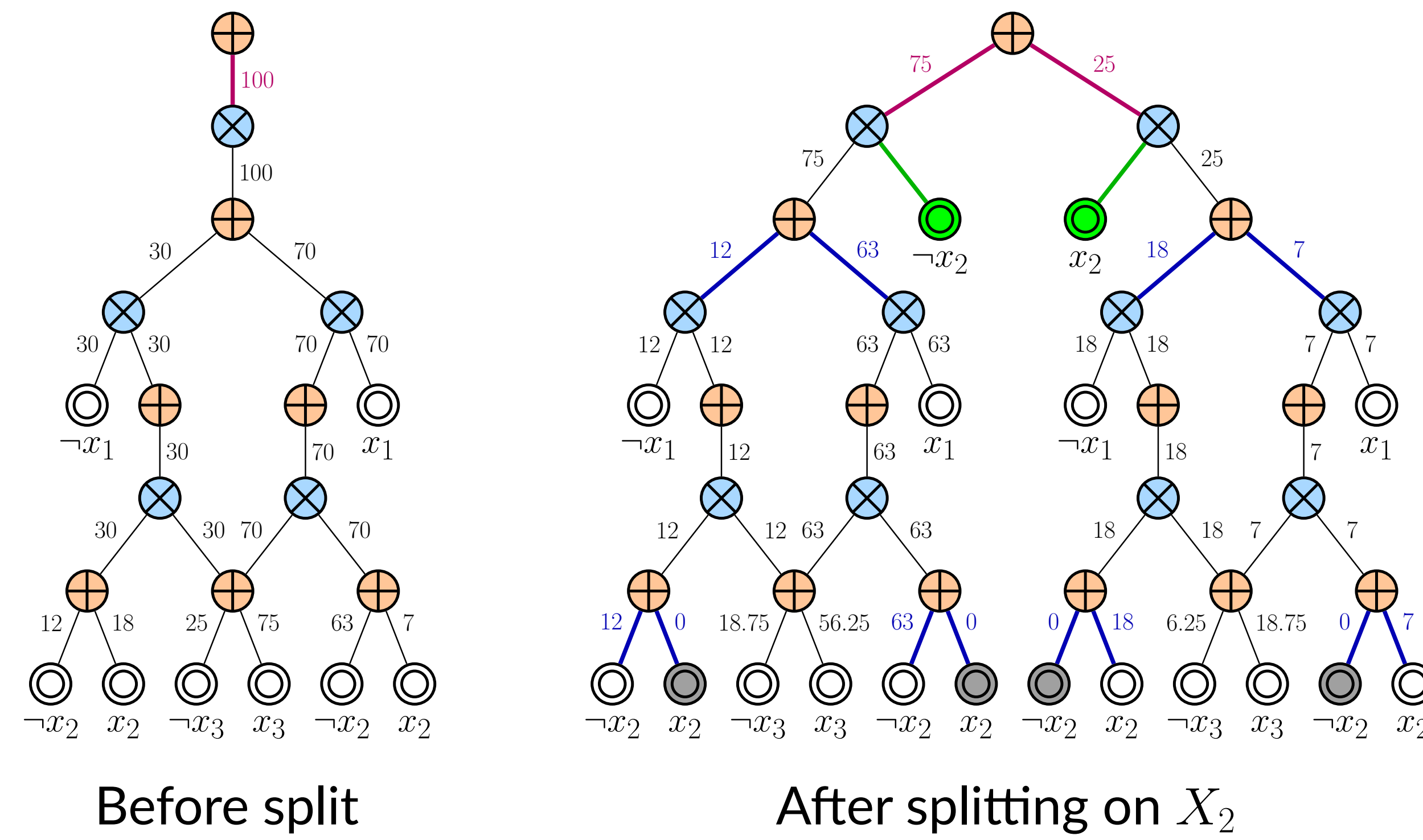
Marginal and **joint flows** are used for splitting heuristics and updating aggregate flows.

We can easily **increment** and **decay** these statistics as we see new data.

Distribution-Preserving Conditional Splits

A **split** duplicates a local subcircuit and **conditions** the two copies on a selected variable X_k .

We **update** the **flows** of all affected edges to keep the distribution identical as the circuit grows.



Aggregate flows

$$a^+ = m_k,$$

$$a^- = a - m_k.$$

Marginal flows

$$m_u^+ = J_{u,k},$$

$$m_u^- = m_u - J_{u,k}.$$

Joint flows

$$J_{u,v}^\pm \approx \frac{m_u^\pm m_v^\pm}{a^\pm}.$$

Updated Flows

- **Which Edge to Split?** We select the edge with the most data passing through it, i.e. the edge with the highest **aggregate flow**. This heuristic is very computationally efficient.
- **Which Variable to Split on?** Once we selected the edge, we chose the variable with the highest **pairwise mutual information** towards all other variables in its scope.

Structural Merging

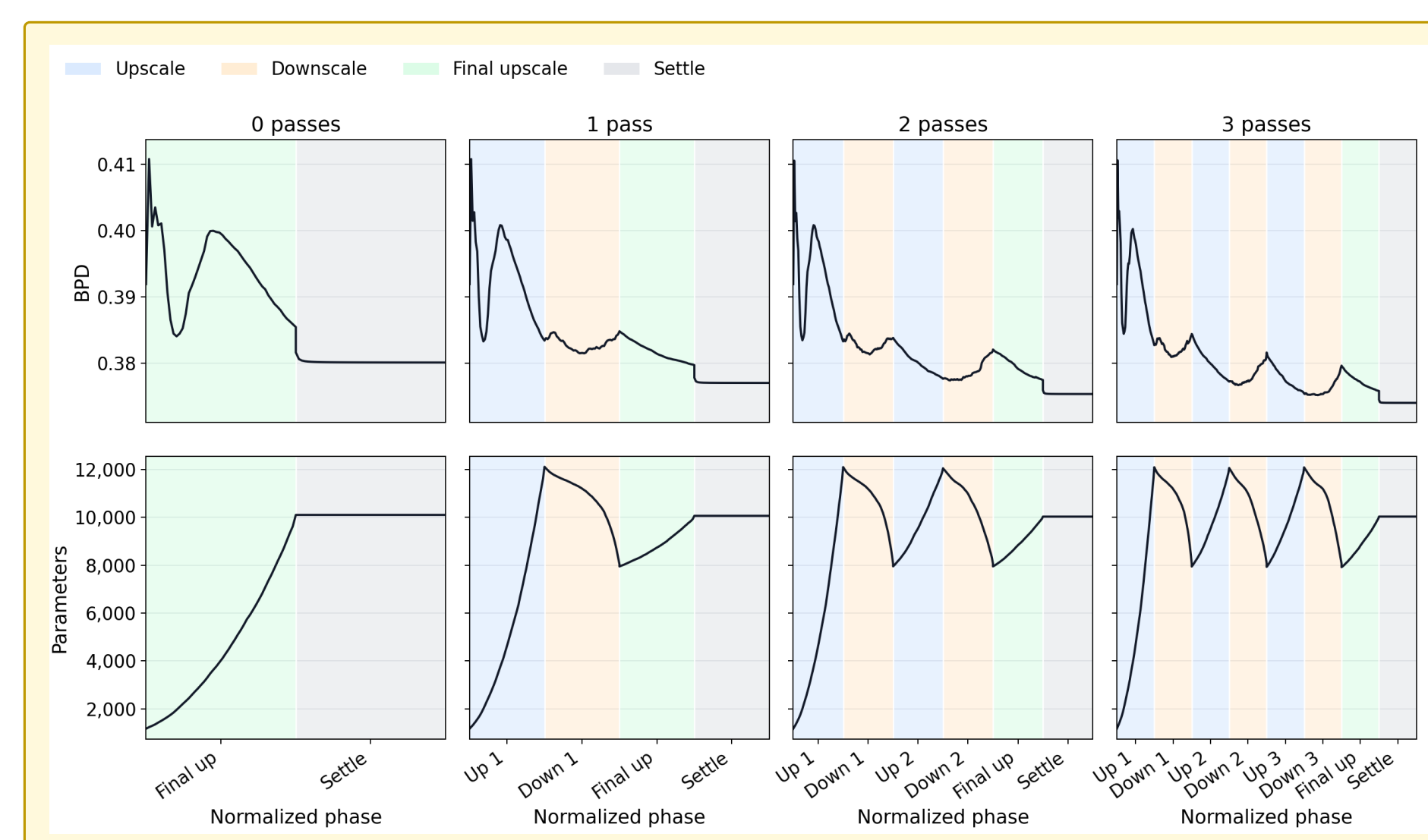
When the circuit reaches a predetermined size, we perform a **merge**.

We compare branches differing only by one conditional variable using **Jeffreys' divergence**:

$$JD(P, Q) = \frac{1}{2} D_{KL}(P \| Q) + \frac{1}{2} D_{KL}(Q \| P).$$

The closest pair is merged by **pruning** the larger branch, undoing its conditioning, and **projecting** its **flow statistics** into the retained branch.

Merging is not only compression: it lets the learner revise earlier structural choices.



Future Work

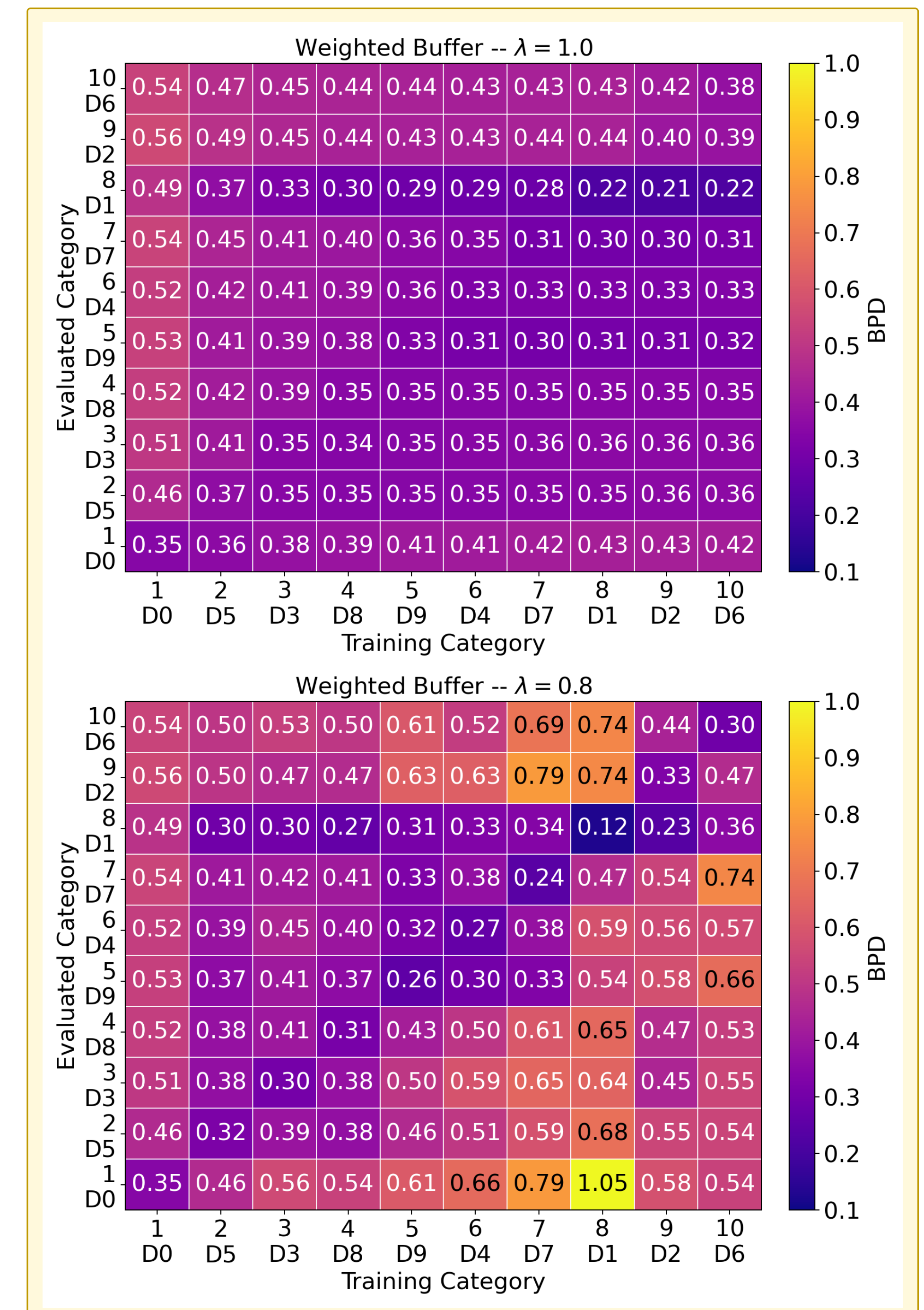
- Size-independent **merging/splitting** criteria.
- Higher level **edge flow approximations** for computational efficiency.

Parameter Decay

We use $\lambda \in (0, 1]$ as a **decay parameter** which allows us to shift the training behavior.

- Large $\lambda \rightarrow$ more **balanced training**.
- Smaller $\lambda \rightarrow$ more **aggressive adaptation**.

We train on binarized MNIST one digit at a time.



References:

- [1] Dang, M. et al. (2022). *Strudel: A fast and accurate learner of structured-decomposable probabilistic circuits*. Int. Jour. of Approx. Reasoning.
- [2] Liang, Y. et al. (2017). *Learning the structure of probabilistic sentential decision diagrams*. UAI 2017.