Transition Matrix Representation of Trees with Transposed Convolutions

Jaemin Yoo[1*] and **Lee Sael**[2]

[1] Seoul National University

[2] Ajou University

* Currently in Carnegie Mellon University

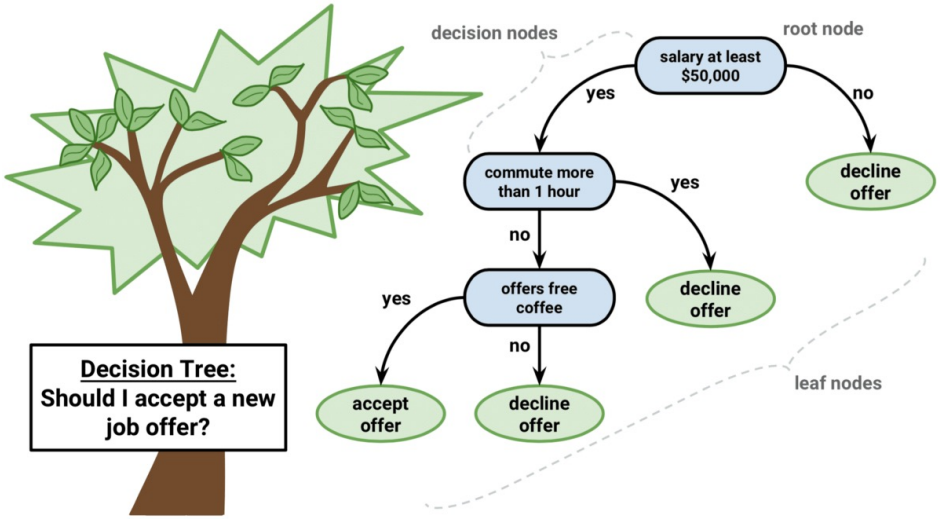
SDM 2022

Outline

- **Introduction**
- Proposed Method
- Further Analysis
- Experiments
- Conclusion

Tree Models

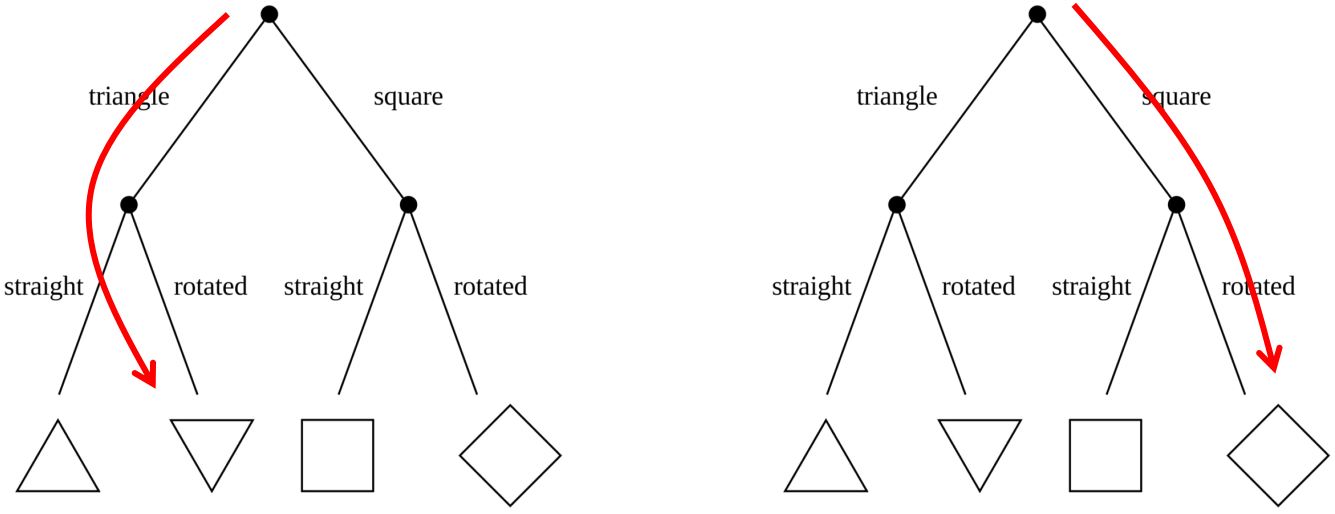
- Tree models have been used for various tasks
- They make **hierarchical** decisions, which split given examples into leaf nodes



https://regenerativetoday.com/simple-explanation-on-how-decision-tree-algorithm-makes-decisions/

Why Tree Models?

- Tree models make **interpretable** decisions
  - A decision path reflects the characteristic of $\mathbf{x}$
  - Similar examples go to the same leaf node



https://commons.wikimedia.org/wiki/File:Simple_decision_tree.svg

Components of Trees

- Tree models consist of three components:
  - **Decision function** f_i at each internal node i
    - How to pass $\mathbf{x}$ to the next layer
  - **Classifier function** g_l at each leaf node l
    - How to make the final prediction
  - **Tree shape**
    - How given examples are clustered
- There are various types of tree models
  - Decision trees, soft decision trees, and so on

Decision Trees

- **Decision trees (DT)**
  - The simplest and most popular tree models
- Each function f_i of DTs works as follows:
  1. Pick a feature index k
  2. Learn the best value b for split
  3. Pass $\mathbf{x}$ based on its k-th attribute x_k
     - If $x_k > b$, pass x to the right child
     - If $x_k \leq b$, pass x to the left child

Soft Decision Trees

- **Soft decision trees (SDT)**
  - Improve the accuracy of DTs with soft decisions
- Each function f_i is a logistic classifier:

$$f_i(x) = \sigma\left(\mathbf{w}_i^{\mathsf{T}}\mathbf{x} + b_i\right)$$

  - Uses all elements at every decision, unlike DTs
  - An example x is passed to the children with soft arrival probabilities, e.g., $(0.3, 0.7)$

Research Motivation

- There are various tree models to be used

Q. *Which one should we use for given data?*

  - It is hard to find an optimal tree model without a systematic way to categorize/understand them

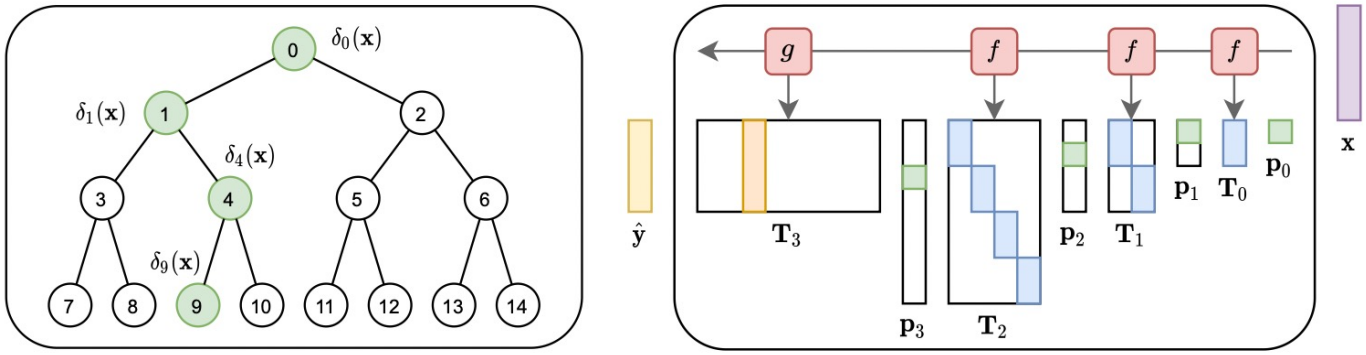
- We propose a **unifying framework** of trees
  1. Generalizes tree models with a few param.
  2. Helps finding optimal choices of components
  3. Improves the efficiency of implementation

Outline

- Introduction
- **Proposed Method**
- Further Analysis
- Experiments
- Conclusion

Overview

- We propose **TART** for unifying tree models
  - Generalizes deep, shallow, binary, or n-way trees
- TART consists of the two main ideas:
  1. Transition matrix representation of trees
  2. Optimization with transposed convolutions

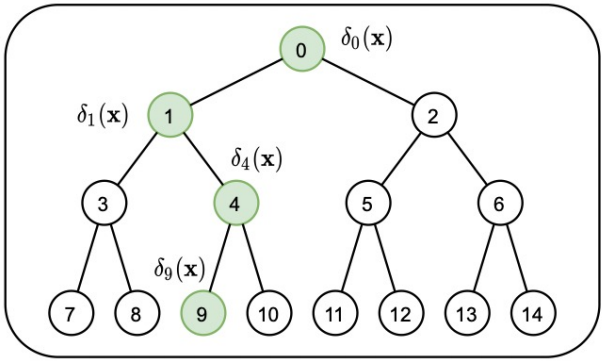


(a) A traditional view of a tree model.

(b) A TART view of a tree model.

Idea 1: Motivation

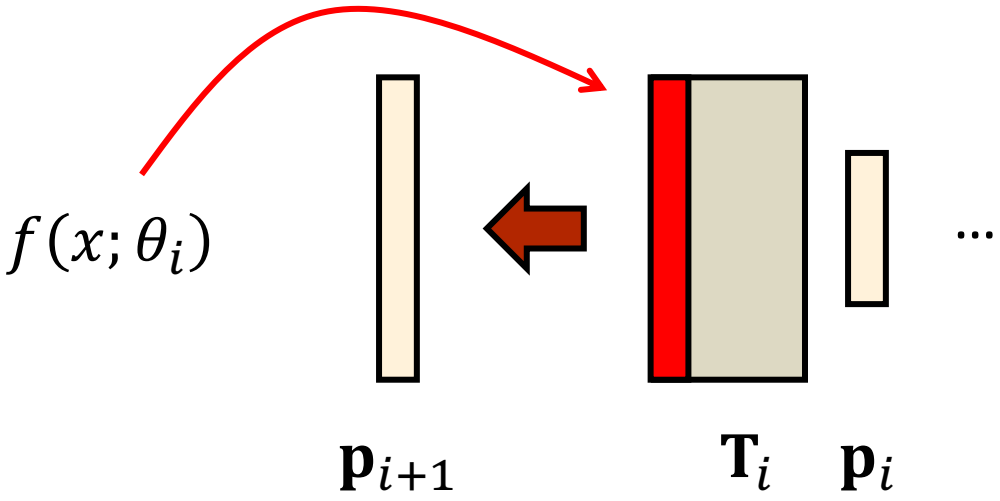
- Traditionally, to understand a tree model $\mathcal{T}$ is
  - Follow a decision path of each example $\mathbf{x}$
    - Compute $\delta_0(\mathbf{x})$
    - Compute $\delta_1(\mathbf{x})$
    - Compute $\delta_4(\mathbf{x})$
    - Compute $\delta_9(\mathbf{x})$
    - …
    - Return the leaf node



  - Such decisions are **sequential** and **dependent** of previous decisions → **hard generalization**

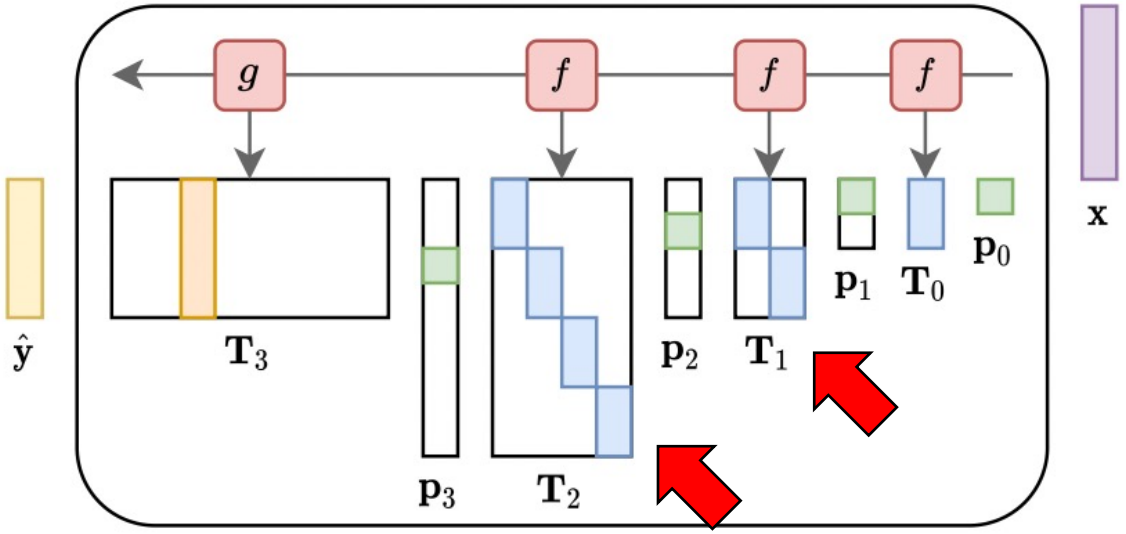
Idea 1: Transition Matrices

- Understand decisions as matrix multiplications
  - Given a decision function f
  - Compute the **transition matrix** $\mathbf{T}_i$ for every layer i
  - Multiply all transition matrices as $\mathbf{T}_3\mathbf{T}_2\cdots\mathbf{T}_0$



Idea 1: Diagonal Entries

- $\mathbf{T}_i$ is a diagonal matrix in most tree models
  - Non-zero entries determine the shape of a tree
    - E.g., non-zeros at $(2c, c)$ and $(2c+1, c)$ represent a binary tree structure, where c is the column index



Idea 1: Summary

- We call this **transition matrix representation**
- **Strengths in efficiency**
  - Parallel computation in GPUs are well supported
- **Strengths in generalization**
  - Applicable to all DAG-structured decision models
    - E.g., 3-way or n-way trees
  - Separate the decision function and the tree shape
    - We can tune each component independently

Idea 2: Motivation

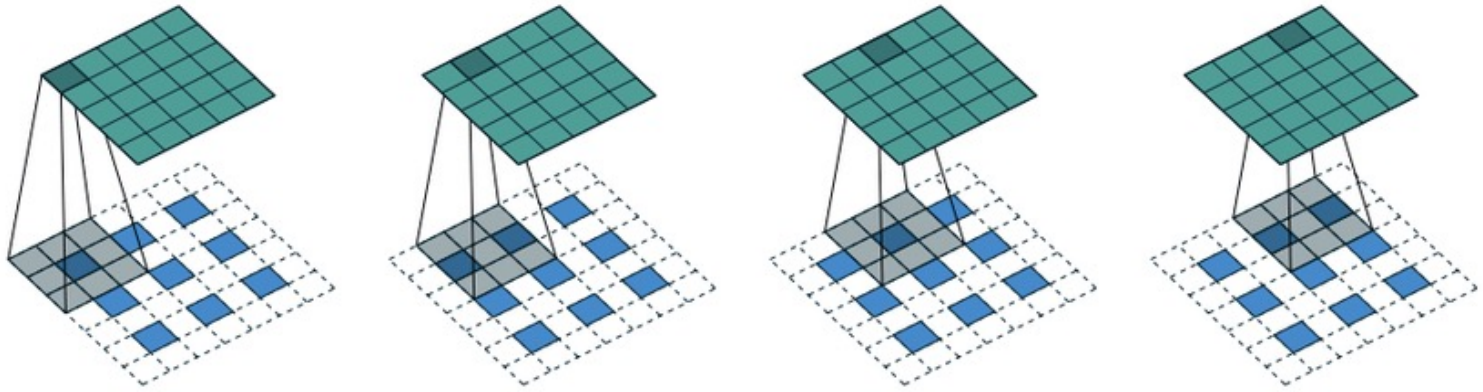
- It is expensive to make $\mathbf{T}_i$ for every layer i
  - The cost can grow exponentially with the depth
  - E.g., in a binary tree, the size of T_i is $2^i \times 2^{i+1}$
- Most entries in $\mathbf{T}_i$ are zero in many cases

Q. How to avoid the direct generation of $\mathbf{T}_i$?

  - Note that we want to maintain the generalizability

Idea 2: Transposed Conv.

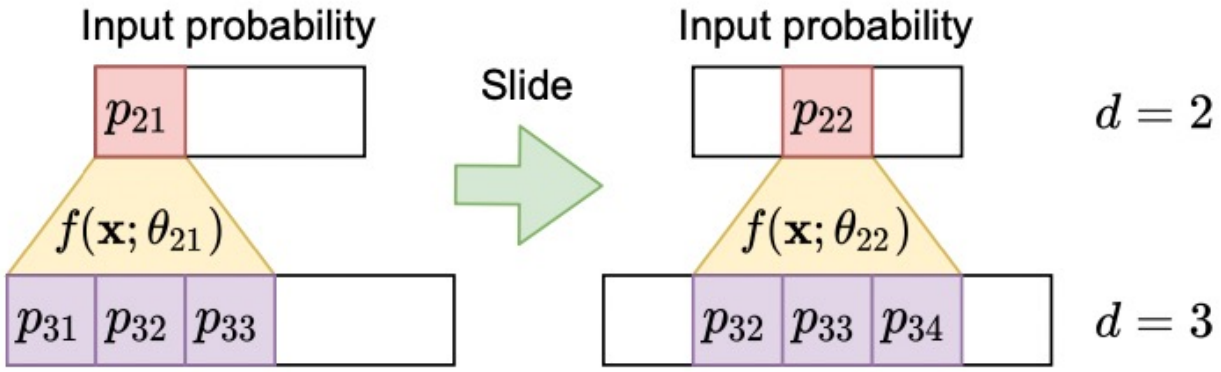
- We use **transposed convolutions (TC)**
  - It is used widely in generative models for images
  - Increases the size of input with convolution filters



https://www.researchgate.net/publication/324783775_Text_to_Image_Synthesis_Using_Generative_Adversarial_Networks

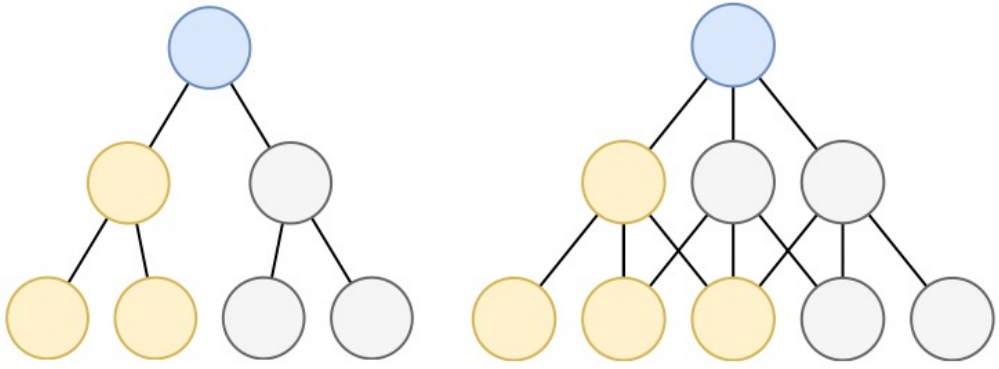
Idea 2: TC to Tree Models

- 1-dimensional TCs make a tree structure
- We insert f to a convolution kernel
  - It slides in a layer to pass x to the next layer



Idea 2: Various Structures

- Parameters in TCs determine the tree shape
  - Let W be the width of a kernel
  - Let S be the length of a stride when a kernel slides
- Tree models with different values of W and S:



(a) TConv ($W = S = 2$)

(b) TConv ($W = 3, S = 1$)

Outline

- Introduction
- Proposed Method
- **Further Analysis**
- Experiments
- Conclusion

Generalizability

- TART generalizes various binary tree models
  - Traditional models use linear decision functions
  - Recent models use deep neural networks as f

Model	$f(\mathbf{x};\theta_i)$	$g(\mathbf{x};\theta_j)$
DT [1]	$\mathbb{I}(s_i(\mathbf{1}_i^\top \mathbf{x} - b_i) > 0)$	$\text{Onehot}(\theta_j)$
SDT [12]	$\sigma(\mathbf{w}_i^\top \mathbf{x} + b_i)$	$\text{Categorical}(\theta_j)$
NDF [3]	$\text{MLP}_i(\text{rand}(\mathbf{x}))$	$\text{Categorical}(\theta_j)$
DNDF [15]	$\text{CNN}(\mathbf{x};i)$	$\text{Categorical}(\theta_j)$
NRF [23]	$\text{CNN}(\mathbf{x};i,\text{depth}(i))$	$\text{Gaussian}(\theta_j)$

Design Parameters

- We introduce five design parameters of TART
  - The depth D of a tree
  - The number H of layers in f
  - The number L of layers in g
  - Parameters W and S for TCs
- We assume f are g are modeled with MLPs
  - To easily control their nonlinearity

Structure-based View

- We categorize existing models with TART
  - $D > 0$ means a decision-based classifier
  - $H > 1$ represents nonlinear decision functions
  - $L > 1$ represents nonlinear leaf classifiers

Models	D	H	L
Logistic regression	$D = 0$	-	$L = 1$
Multilayer perceptrons [21]	$D = 0$	-	$L > 1$
Simple ensembles of experts	$D > 0$	$H = 0$	Any L
Trees of type 1 [1, 12]	$D > 0$	$H = 1$	$L = 0$
Trees of type 2 [15, 23]	$D > 0$	$H > 1$	$L = 1$
Trees of type 3 [19, 20]	$D > 0$	$H = 1$	$L > 1$

Promising Tree Structures

- We propose promising structures of trees
  - TART-A is a typical binary tree of depth $D = 6$
  - TART-B is a small tree having expressive g
    - Tree depth is only $D = 2$, but g has $L = 4$ layers
  - TART-C has a balanced structure and classifiers

Model	W	S	D	H	L	Property
TART-A	2	2	6	1	1	Strong in small data
TART-B	2	2	2	1	4	Strong in large data
TART-C	3	2	3	1	2	Best balance

Outline

- Introduction
- Proposed Method
- Further Analysis
- **Experiments**
- Conclusion

Datasets

- We use 121 tabular datasets for classification
- We categorize them based on their sizes:
  - They contain 9, 37, and 75 datasets, resp.

Group	Examples		Features	Labels
	Min	Max	Avg ± Std	Avg ± Std
Large	10,992	130,064	19.0 ± 15.8	8.2 ± 8.5
Mid	1,000	8,124	40.2 ± 48.4	12.2 ± 26.6
Small	10	990	24.4 ± 37.9	4.1 ± 3.7
All	10	130,064	28.8 ± 40.8	6.9 ± 15.5

Classification Accuracy

Q1. How accurate are our TART models?

A. They show the highest accuracy in general

- TART models are better than MLPs
  - l in MLP-l means # of layers
- TART-A, B, and C work the best on different sizes

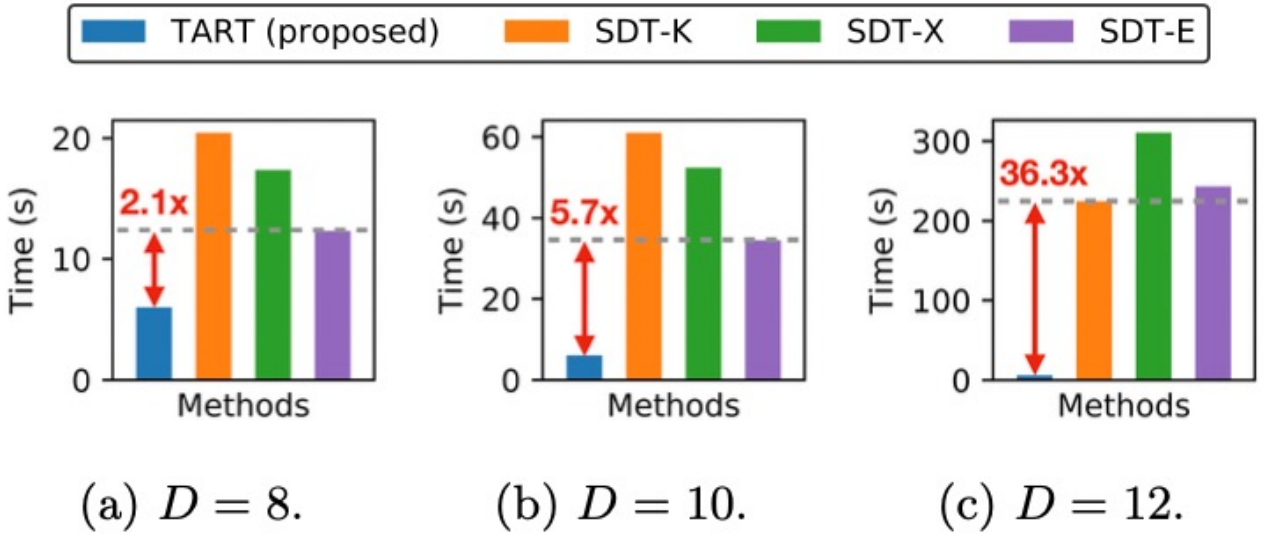
Model	Large	Medium	Small
DT	88.3±0.1	76.3±0.2	71.9±0.7
LR	79.1±0.1	80.8±0.2	75.8±0.3
SVM-lin	77.7±0.1	79.0±0.2	74.9±0.5
SVM-rbf	87.6±0.0	81.1±0.1	**77.0±0.2**
MLP-1	78.7±0.1	78.9±0.3	73.4±0.4
MLP-2	87.8±0.1	83.0±0.4	76.5±0.4
MLP-4	91.8±0.1	83.0±0.2	76.8±0.2
MLP-8	91.5±0.1	82.5±0.3	76.0±0.5
MLP-16	85.3±0.9	78.3±0.2	75.1±0.6
TART-A	88.2±0.2	82.6±0.2	**77.0±0.6**
TART-B	**92.1±0.1**	82.7±0.4	76.0±0.3
TART-C	89.6±0.4	**83.1±0.2**	76.3±0.1

Training Time

Q2. Does it improve efficiency of tree models?

A. TART speeds up training time up to $36.3\times$

- We compare various implementations of SDTs



(a) $D = 8.$ (b) $D = 10.$ (c) $D = 12.$

Outline

- Introduction
- Proposed Method
- Further Analysis
- Experiments
- **Conclusion**

Conclusion

- We propose **TART** for unifying tree models
  - Transition matrix representation
    - Allows us to represent a tree with matrix multiplications
  - Optimization with transposed convolutions
    - Improves efficiency without losing the generalizability
- The main strengths of TART are
  - We can easily search for an optimal tree model
  - We can utilize parallel computations in GPUs

Thank You!

Jaemin Yoo (jaeminyoo@cmu.edu)

https://github.com/leesael/TART