Decision Tree using Spiking Neural Networks

Ayush Kumar Singh
ai20btech11028@iith.ac.in

Srikaran Perambuduri
ai20btech11018@iith.ac.in

Digjoy Nandi
ai20btech11007@iith.ac.in

Omkaradithya R Pujari
ai20btech11017@iith.ac.in

Abstract

Decision trees are widely used in various applications to classify input data and dimensionality reduction. However, traditional decision trees need higher memory requirements and computational complexity. Choosing the depth of the decision tree is difficult as it can lead to overfitting if left unchecked. Selecting the best feature and threshold is also tricky, as it involves trying out many possible pairs. Hence, we aim to improve its efficiency by combining it with spiking neural networks, which are event data-driven neural networks.

1. Introduction:

Decision trees are a powerful and versatile tool used in various fields, such as data mining, machine learning, and artificial intelligence, for making decisions based on a set of conditions. They provide a highly intuitive graphical representation of a series of decisions and their associated outcomes, which simplifies complex decision-making processes.

A decision tree consists of three primary components: nodes, branches, and leaves. Nodes represent decision points where a specific condition or attribute is evaluated. Branches corresponding to the possible outcomes or choices that emanate from each node, leading to subsequent nodes or the final decisions represented by leaves.

The decision-making process starts at the root node, where the initial condition is evaluated. Based on the outcome, the process moves along the appropriate branch to the next node, continuing through the tree until a leaf node is reached, representing the final decision.

One of the key advantages of decision trees is their ease of interpretation and understanding. They can be easily visualized and comprehended even by non-experts, facilitating communication and decision-making across various stakeholders. Moreover, decision trees can handle both categorical and numerical data, which broadens their applicability.

However, decision trees are also prone to certain limitations, such as overfitting, where the model becomes too tailored to the training data and fails to generalize well to new, unseen data. To address this issue, techniques such as pruning and the use of ensemble methods like random forests can be employed.

Decision trees are a valuable tool for navigating complex decision-making scenarios, offering a clear and interpretable representation of the processes involved. Despite their limitations, they can be highly effective across multiple domains with the right techniques and adjustments.

In this report, we tried to reduce the computational cost of implementing a decision tree using an event data-driven neural network, i.e. spiking neural network. We first tried to design the decision tree using artificial neuron networks, which perform the same or better than the decision tree in most benchmarking test datasets. We further attempted to modify this pipeline into a spiking neural pipeline for better computational cost.

2. Approach:

In our novel approach towards improving decision trees using SNN, we tried to replace mathematical probabilistic-based learning in decision trees with end-to-end learning using a suitable loss function which we call a "Neural Decision Tree". We replaced decision nodes with neural nodes, each associated with a weight and bias vector. We later attempted to replace the artificial nodes in the Neural Decision Tree with LIF(leaky integrate and fire neurons) to generate spikes. We called this model the "Spiking Decision Tree", which is trained from the spikes generated from the LIF neurons.

2.1. Neural Decision Tree:

We use an appropriate loss function to replace the mathematical probabilistic-based learning in the Neural Decision Tree with end-to-end learning. We replaced decision nodes with neural nodes, each associated with a weight and bias

vector. This parameter is a unit vector whose linear combination gives the splitting criteria. Each node takes a given input vector and outputs a decision which we mathematically represented using coefficients for child nodes.

We are required to get positive coefficients for the selected child node and zero for the rejected one, which makes the entire subtree inactive for that particular input. At the leaf nodes, we assign a label (0 or 1) to each node representing the class. Each leaf node outputs a probability of a particular class assigned to it, and a softmax function is applied to those probabilities. Finally, a binary cross-entropy function is used to calculate the loss.

The significant advantage of our method over traditional decision trees is that the splitting criteria can be a linear combination of features rather than a single one. We aim to improve the combination of features by introducing higher-order terms.

2.2. Spiking Decision Tree

We used the conversion technique to implement the Neural Decision Tree to Spiking Decision Tree. We use surrogate gradient descent to train our model. Surrogate Gradient Spiking is a technique used in deep learning with spiking neural networks that replaces the non-differentiable spike function with a differentiable function to allow for gradient-based optimization during training.

3. Experimental Results:

We use a variety of benchmarking datasets to benchmark our Neural Decision Tree. But to compare the accuracy and performance of the Neural Decision Tree and Spiking Decision Tree, we used a toy dataset with the label 0 if $x < 5$ and 1 otherwise.

3.1. Datasets used

The figure 1 is a toy dataset used to test depth 1 neural decision tree. It returns 0, if $x < 5$ and, 1 otherwise.

The figure 2 is a toy dataset used to test depth 2 neural decision tree. It returns 1, if $0 \leq x < 5$ and, 0 otherwise.

The model was also tested on Iris Dataset and the Bank Dataset.

3.2. Neural Decision Tree

3.2.1 Training procedure

We follow the decision tree structure for this model, where each parent node has a direct connection to two child nodes. The preliminary version of this model is only suitable for binary classification. In the learning phase, the parameter assigned to each node represents the splitting criteria. We chose ReLU as our activation function as it rejects the negative values and outputs only the positive ones, which is required for selecting coefficients of the child node. To avoid exploding gradient problem, we keep normalizing the output produced by each decision node.

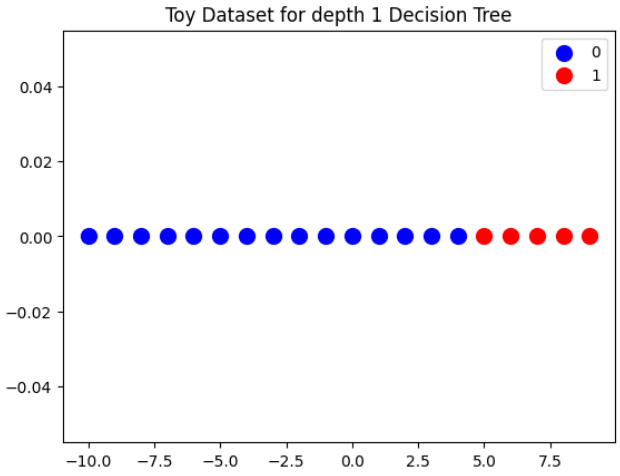

Figure 1. Toy Dataset for testing Depth 1 Decision Tree

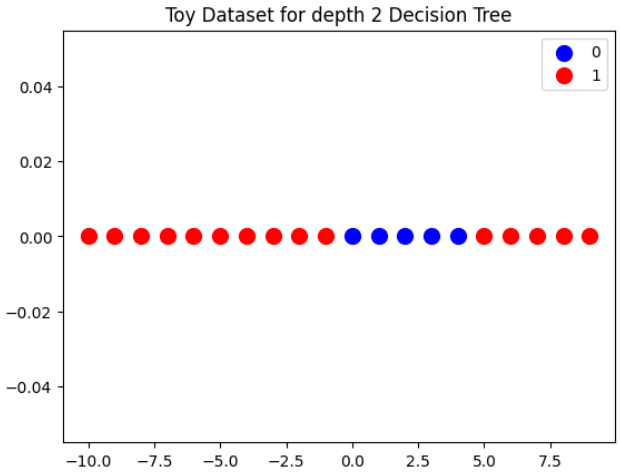

Figure 2. Toy Dataset for testing Depth 2 Decision Tree

3.2.2 Results

[1] The following table are the tabulated results of our approach with the original decision trees.

3.3. Spiking Neural Tree

We attempted to convert the neural decision tree to spiking neural network using heaviside step function to generate spikes and surrogate gradient descent.

We faced some issues to directly convert the model to SNN although, the approach has been working for the first

[1] Github link: https://github.com/srikaran-p/NeuromorphicProject

Dataset	Vanilla DT	Neural DT
Toy Dataset 1	1.0	1.0
Toy Dataset 2	1.0	1.0
Iris Dataset	1.0	1.0
Bank Dataset	1.0	0.966

Table 1. Results comparing the accuracy of vanilla decision tree and neural decision tree

toy dataset using depth 1 and depth 2 decision trees (giving perfect accuracy).

4. Future Work

There are certain modifications we would like to add in the future:

1. Convert the Neural Decision Tree to Spiking Decision Tree for n-depth decision trees.
2. Extend the model to do multi-class classification from binary classification. Our Neural Decision Tree overfits quickly when trained using a multi-class dataset.
3. Early pruning to avoid overfitting.
4. Add a regularization term to prevent overfitting

5. Conclusion

This approach of using neural networks to find the parameters and the thresholds improves the accuracy. Also, we can display the parameters used at each node of the decision tree. We attempted to convert it to SNN. With more modifications, it would improve the accuracy and less energy would be consumed to train the model.

6. References

1. NBDT: Neural-based decision tree by Alvin Wan et al.