*Report on CuDecisionTree*

*BY:*

**Anshika Mittal**

(Enrolment Number: **19114009**)

1. **Problem Statement**

Implement a new parallelized decision tree algorithm on a CUDA (compute unified device architecture), which is a GPGPU solution provided by NVIDIA.

1. **Novelty**

In order to improve data processing latency in huge data mining, in this project I design and implement a new parallelized decision tree algorithm on a CUDA. By leveraging the existing CUDA components, such as prefix-sum and parallel sorting, my proposed CuDecisionTree system performs well and gets major performance improvement than sequential decision tree algorithms.

1. **Evaluation Parameters/ Dataset used**

Some of the evaluation parameters in my model are total cost time, accuracy , and the size of the system.

Data Points are generated randomly in the code.

**Input Format is:** {num\_of\_samples,attr\_count}

**Sample Input:** 200000 784

**Sample Output :** Total nodes in the tree: 63583

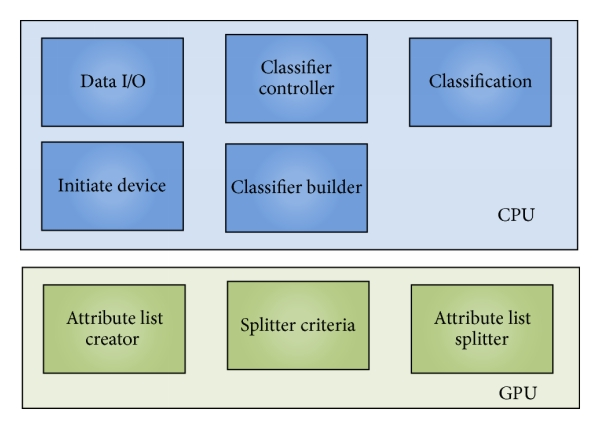
1. **Methodology**

Figure: The CuDecisionTree system components.



The principle of CuDecisionTree is dispatching flow control, handling, and communication tasks to CPU and on the other hand assigning computing intensive jobs to GPU. The blue parts in the figure are running on the CPU while the green parts are running on a GPU.

There are 7 major steps in the CuDecisionTree system.

1. Training and testing data are loaded to host memory from disks.
2. Initialization of the device includes query device information, allocation memory space, and copy of training data into the device.
3. The system sets up some parameters for the user. For instance, the minimum numbers of data of a leaf, the maximum depth of the classifier etc.
4. Creating attribute lists in device by moving each attribute to corresponding position and then sorting all attribute lists in devices.
5. This is the most important step of the system where an iterative breadth first scheme is used instead of the recursive model of decision tree building algorithm. Host is in charge of the working flow of the whole system.
6. Now the classification is performed on the host sequentially.
7. The results are presented on hosts.

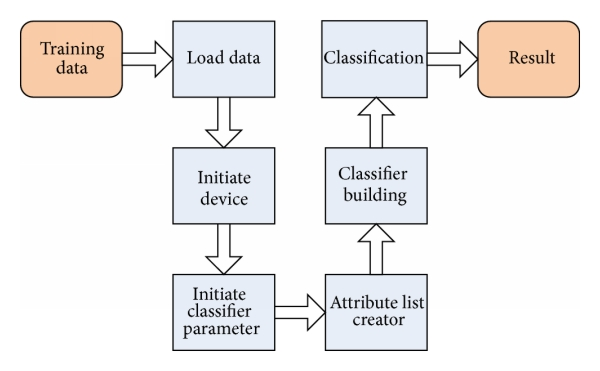


Figure: Flowchart of the proposed CuDecisionTree algorithm.



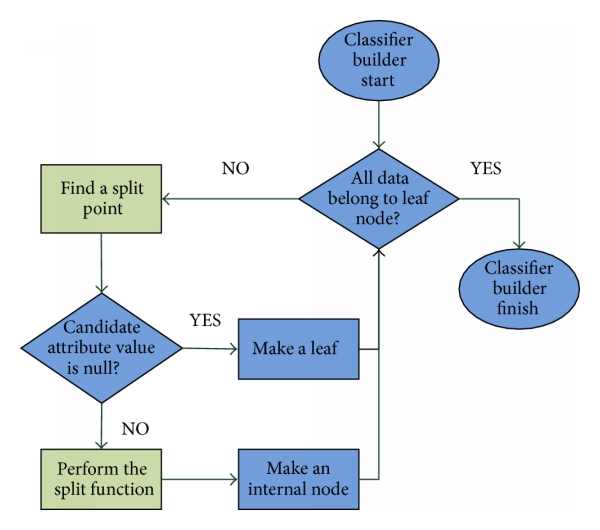
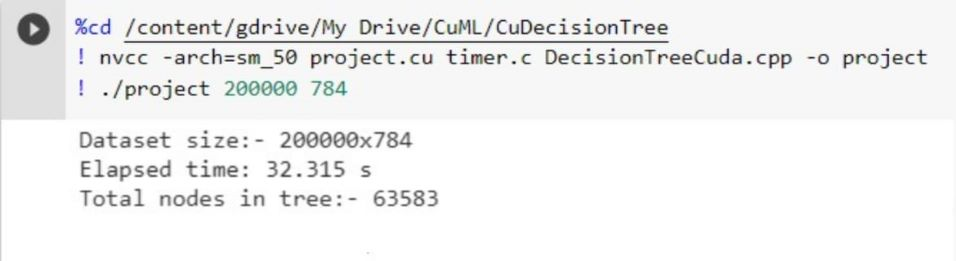


Figure: Flowchart of the classifier building phase



1. **Results**



1. **Conclusion**

For solving problems that are able to be solved in parallel and with high density computation,using GPUs normally brings remarkable improvement of performance. Many machine learning algorithms have been developed on CUDA GPUs as they show performance improvement compared to CPU. This project shows that by leveraging the existing CUDA components, such as prefix-sum and parallel sorting, our proposed CuDecisionTree system performs well and gets major performance improvement than sequential decision tree algorithms.

1. **References**
2. [Getting Started with CUDA](https://www.nvidia.com/content/cudazone/download/Getting_Started_w_CUDA_Training_NVISION08.pdf)
3. [About CUDA](https://developer.nvidia.com/about-cuda)
4. [Parallel Implementation of Decision Tree Research paper](https://www.researchgate.net/publication/220773622_Parallel_Implementation_of_Decision_Tree_Learning_Algorithms#:~:text=The%20approach%2C%20called%20Parallel%20Decision,machine%20learning%20and%20data%20mining.)
5. [Google Colab Introduction](https://colab.research.google.com/notebooks/intro.ipynb)
6. [Neural networks and deep learning](http://neuralnetworksanddeeplearning.com/)

<https://www.hindawi.com/journals/tswj/2014/745640/>