Very Fast Decision Tree

As described in "Mining High-Speed Data Streams" (Domingos & Hulten, 2000).

Limited by

- Time
- Memory
- Data evolves over time - nonstationary distribution

About VFDT

- It's fast incremental learning algorithm
- Needs a single pass over data
- Does not store examples in main memory
- It's Anytime algorithm which means you can stop the algorithm anytime with the same performance guarantee.
- The great thing about VFDT is that the tree produced by the algorithm is asymptotically similar to the tree produced by the batch learner.
- Hoeffding tree algorithm implementation is VFDT

From Lecture Notes

The working principle of VFDT is as follows:

- It builds Hoeffding tree iteratively as it sees more examples.
- It starts with the root node and picks an attribute to split based on usual node split criterion such as Entropy, Information Gain, Gini Index.
- It does not split the node until the node has seen at least n
- It does not split the node until the node has seen at least n examples so that with probability at least 1 − δ, difference in node's heuristic measure G is where is given by

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$$

- Where R is the range of the attribute. Above equation Hoeffding bound and hence Hoeffding tree comes from.

Hoeffding Decision Tree

- Classical Decision Tree learners are limited by main memory size
- How to decide how many are necessary? Hoeffding Bound!

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$$

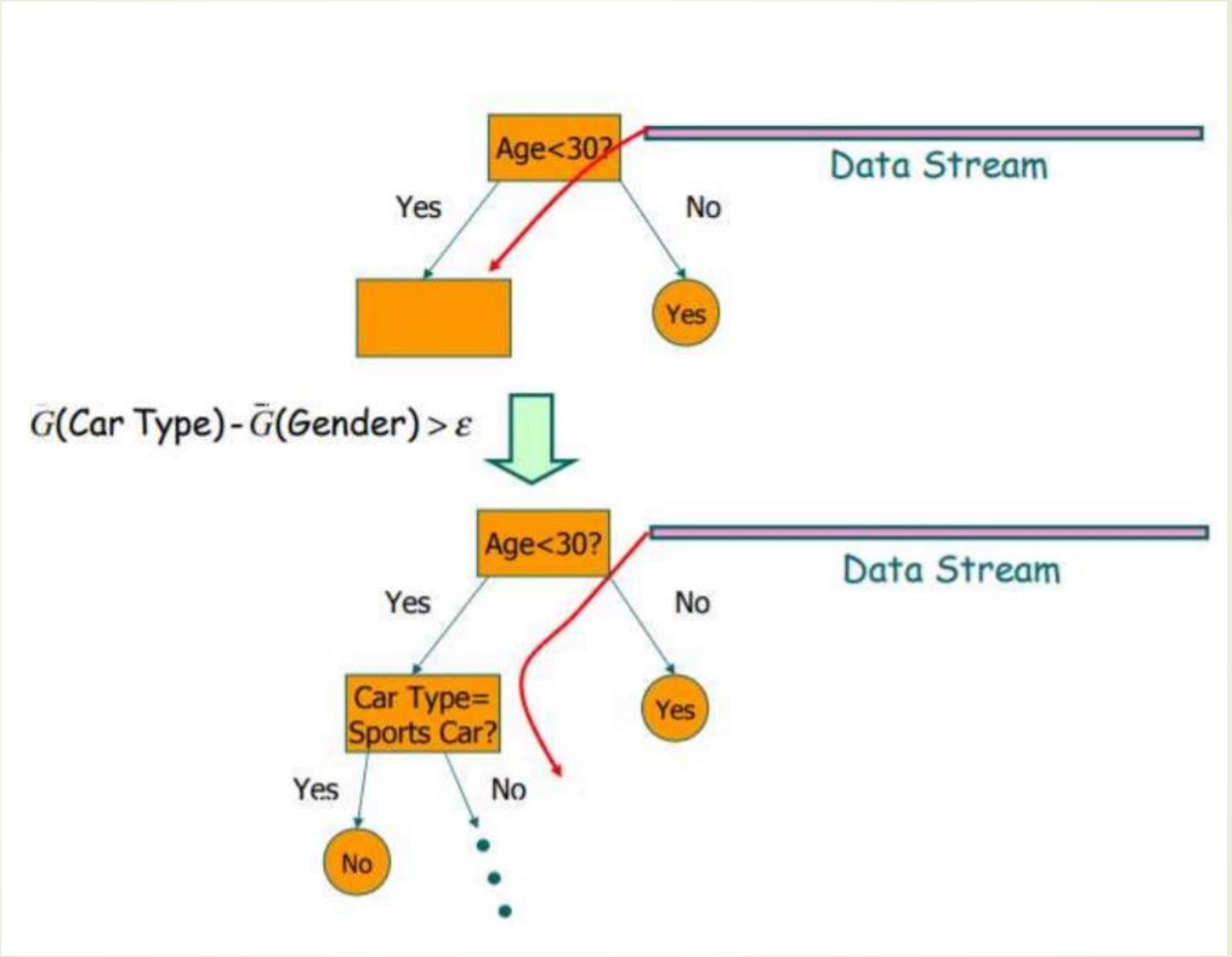
"Suppose we have made n independent observations of a variable r with domain R, and computed their mean r . The Hoeffding bound states that, with probability $1 - \delta$, the true mean of the variable is at least $r - \epsilon$"

Hoeffding Tree Algorithm

- Compute the heuristic measure for the attributes and determine the best two attributes
- At each node check for the condition

$$\Delta\bar{G} = \bar{G}(X_a) - \bar{G}(X_b) > \epsilon$$

- If true, create child nodes based on the test at the node; else, get more examples from stream.



https://www.slideshare.net/Davix95/mining-high-speed-data-streams-hoeffding-and-vfdt

In Few Words

- Learning in Hoeffding tree is constant time per example (instance) and this means Hoeffding tree is suitable for data stream mining.
- Requires each example to be read at most once (incrementally built).
- With high probability, a Hoeffding tree is asymptotically identical to the decision tree built by a batch learner.

$$E[\Delta_i(HT_\delta, DT_*)] \leq \frac{\delta}{p}$$

- Tie Breaking
- Bound Calculation
- Poor Attributes – Pre pruning null value
- Ram based
- Rescan

About Dataset Used

- I used bank dataset which contains details of Bank customers. The bank needs to check who is eligible to get loan from them.
- This dataset is widely used for Decision tress implementation so we chose the same. In my task I devided the dataset into 4 set which contain 4 different size of dataset. This is because to test the sizeof data increases the hoeffding MLE will result less error value.
- The features contains : ['age', 'job', 'marital', 'education', 'default', 'balance', 'housing', 'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'poutcome']

	age	job	marital	education	...	pdays	previous	poutcome	y
0	30	unemployed	married	primary	...	-1	0	unknown	no
1	33	services	married	secondary	...	339	4	failure	no
2	35	management	single	tertiary	...	330	1	failure	no
3	30	management	married	tertiary	...	-1	0	unknown	no
4	59	blue-collar	married	secondary	...	-1	0	unknown	no
5	35	management	single	tertiary	...	176	3	failure	no
6	36	self-employed	married	tertiary	...	330	2	other	no
7	39	technician	married	secondary	...	-1	0	unknown	no
8	41	entrepreneur	married	tertiary	...	-1	0	unknown	no
9	43	services	married	primary	...	147	2	failure	no

About codes and results:

- Our Code is consist of two main classes. In one class we define nodes and implement splitter, gini and hoeffding tree and in another class we defined some parameters for hoeffding bound calculation, providing training datas and performing prediction and calculating accuracy.

```
Total data size:  4521

Test set (tail):  500

Training set: 1000, ACCURACY: 0.8840

Training set: 2000, ACCURACY: 0.8840

Training set: 3000, ACCURACY: 0.8840

Training set: 4000, ACCURACY: 0.8860

--- Running time: 0.287230 seconds ---
```

Thank you

By,

Nayeem Ahmed

Neptun code: MCVGRB