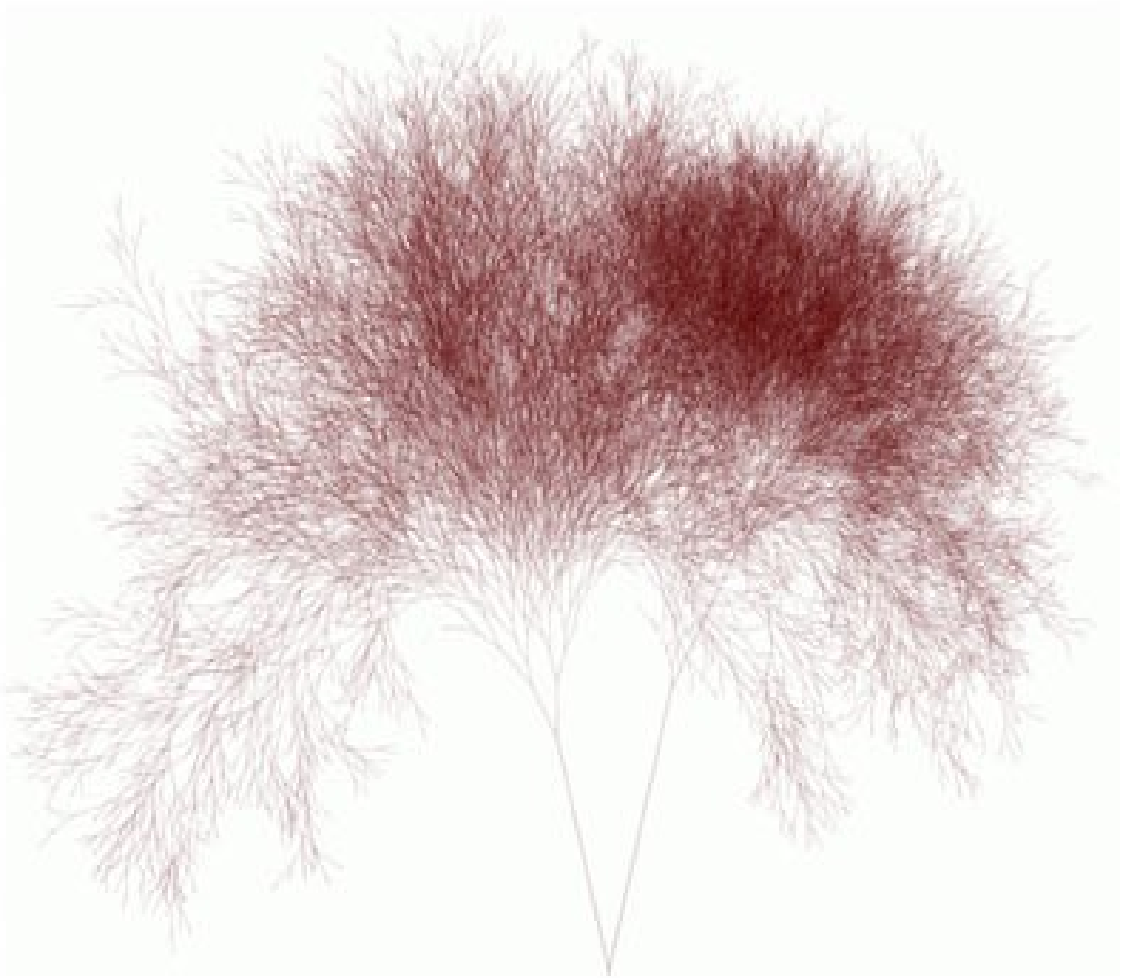
A Review of Distributed Decision Tree Induction



Gregory Gay
West Virginia University
greg@greggay.com

Data Mining 101

- Data mining = finding patterns in heaps of data
- Classification = Given data and a series of possible labels, gather evidence and assign a label to each piece of data.

Decision Tree Induction

- These algorithms greedily and recursively select the attribute that provides the most *information gain*.
- This attribute is used to partition the data set into subsets around a particular value.
- Stops when the leaf node in the tree only contains data with the proper class label.

Splitting the Data

- Split data based on the (attribute,value) pair that maximizes:
- $$gain(T,a) = I(T) - \sum_{v \in Values(a)} \frac{|T_v|}{|T|} * I(T_v)$$
- I(T) is an impurity measure. This is usually *entropy*(C4.5) or *gini*(SMP):

$$entropy(T) = -\sum_j \frac{|C_j|}{|T|} * log_2(\frac{|C_j|}{|T|})$$

$$gini(S) = 1 - \sum_j (\frac{|C_j|}{|T|})^2$$

Climbing Mount Everest

- The success of data mining is its downfall.
- Fast, accurate results expected on huge data sets.
- Parallelism of data and/or the algorithm.

Paradigm A: Data Parallelism

- Given the large size of these data sets, gathering all of the data in a centralized location or in a single file is neither desirable nor always feasible because of the bandwidth and storage requirements.
- Also, data may be fragmented in order to address privacy and security constraints.
- In these cases, there is a need for algorithms that can learn from fragmented data.
- Two methods – horizontal, vertical

Horizontal Fragmentation

- The entries are equally divided into a number of subsets equal to the number of different storage sites.
- The learner uses frequency counts to find the attribute that yields the most information gain to further partition the set of examples.
- Given: |E|= # examples in the data set T , |A|= #attributes, V= max(values per attribute), M=#sites, N=#classes, size(D)=#nodes in tree D
- Time complexity = |E||A|size(D)
- Communication complexity = N|A|VMsize(D)

Vertical Fragmentation

- Individual attributes or subsets of attributes, along with a list of (value, identifier) pairs for each are distributed to sites.
- Can suffer from load imbalance and poor scalability.
- Given: |E|= # examples in the data set T , |A|= #attributes, V= max(values per attribute), M=#sites, N=#classes, size(D)=#nodes in tree D
- Time complexity = |E||A|Msize(D)
- Comm. complexity = (|E|+N|A|V)M size(D)

When Distributed Data Wins

- It would be possible to collect those fragments and reassemble them in a centralized location. So, why we should leave the data at distributed sites?
- Privacy or security - learner must perform solely on statistical summaries.
- Distributed versions of algorithms compare favorably with the corresponding centralized technique whenever its communication cost is less than the cost of collecting all of the data in one place.

Paradigm B: Task Parallelism

- The construction of decision trees in parallel.
- Single process begins the construction process. When the #child nodes = #available processors, the nodes are split among them. Each processor then proceeds with the tree construction algorithm.
- Implementing parallel algorithms for decision tree classification is a difficult task for multiple reasons.

Paradigm C (B.5?): Hybrid Parallelism

- Most parallel algorithm don't practice strict task parallelism
- Instead – parallelized algorithm over split data.
- Some (parallelized C4.5), switch between data/task parallelism when communication cost is too high.
- Others always operate in both modes (SMP, INDUS).

Algorithm: C4.5 Parallel

- Parallel implementation of Ross Quinlan's C4.5 decision tree algorithm.
- Data parallelism at the beginning, task parallelism at the lower nodes of the tree.
- Horizontal fragmentation scheme. Uniform load balance is achieved by equally distributing data among the processors and using a breadth-first strategy to build the actual decision tree. When a process finished exploring it nodes, it sends a request for more nodes.

- Each processor is responsible for building its own attribute lists and class list.
- The continuous attribute lists are globally sorted. Each processor has a list of sorted values.
- Before evaluating the possible split points in their entries, the distributions must reflect the data entries assigned to the other processors.
  - Discrete – store the data everywhere
  - Continuous - the gain calculated based on the distributions before and after the split point.
- After evaluating all of the local data, the processors communicate among themselves in order to find the best split.

- Horizontal data fragmentation as long as the #examples covered by the nodes > a pre-defined threshold (when communication cost for building a node > cost of building it locally plus the cost of moving the set of examples between processors)
- When #data entries < threshold, they are evenly distributed among the processors.

Algorithm: SMP Tree Classifier

- SMP clusters = shared-memory nodes with a small number of processors (usually 2-8) connected together with a high-speed interconnect.
- Two-tiered architecture where a combination of shared-memory and distributed-memory algorithms can be deployed.
- The training dataset is horizontally fragmented across the SMP nodes so each node carries out tree construction using an equal subset of the overall data set.

- Attribute lists are dynamically scheduled to take advantage of the parallel light-weight threads
- One of these threads is designated as a master thread. Is responsible for processing the attribute lists as well as exchanging data and information between SMP nodes.
- The numerical attribute lists are globally sorted such that the first node has the first portion of the data set
- When a tree node is split into its children, the local attribute lists are partitioned among those children. Thus, no communication costs are incurred due to the attribute lists as the tree is grown in each SMP node.

- Finding global best split causes synchronization
- The algorithm will proceed in a breadth-first manner, processing all of the leaf nodes before processing any of the new child nodes.
- Before finding the new best split point, attribute lists inserted into a queue shared by all of the threads running on the SMP node. This queue is used to schedule attribute lists to threads.
- Each leaf node calculates and compares its split points with all of those along the leaf nodes
- SMP node 0 can compute the overall best split point for all attributes. It will broadcast this point to all of the SMP nodes.

INDUS

- INDUS is a multi-agent system used for knowledge acquisition from data sources
- Can deal with horizontal or vertical data, only needs statistical summaries of the data.
- Designed to provide a unified query interface over a set of distributed, heterogeneous, and autonomous data sources as if the dataset were a table.
- These queries are driven by an internal tree learner that uses them whenever a new node needs to be added to the tree.

Summary

- We want to classify data. Decision tree classifiers are a good way to do it.
- We need to classify BIG data sets.
- We can improve performance by parallelizing the data (horizontally or vertically) or the algorithm. Often both.
- We looked at a few algorithms to do this.
- They all perform well, show a promising improvement over standard methods.

Now you can climb the mountain!

Questions? Comments?

- Paper: http://greggay.com/pdf/dsys1.pdf
- greg@greggay.com
- http://twitter.com/Greg4cr
- http://facebook.com/greg.gay