УДК 004.02

Parallel construction of decision trees

B.V. Parshentsev, E.G. Tolstoluzhskaya
V. N. Karazin Kharkiv National University, Ukraine

Decision trees are a well-established set of methods for classification, recognition and decision support in the machine learning, the identification, the data analysis and the situational management. The decision tree must be compact - it lessens expenses when answering questions. Moreover, compact trees have a better prognostic ability. In some applications, such as Data Mining, a dataset to be learned is very large. In those cases it is highly desirable to construct univariate decision trees within a reasonable period of time. This can be accomplished by parallelizing univariate decision tree algorithms.

Key words: *hypersingular integral equations, numerical solution, computer-based experiment, diffraction problem, parallelism, ID3, knot.*

Деревья принятия решений - хорошо зарекомендовавший себя набор методов классификации, распознавания и поддержки принятия решений в компьютерном обучении, идентификации, анализе данных, ситуационном управлении. Дерево решений должно быть компактным - это экономит затраты при ответе на вопросы; Кроме того, компактные деревья имеют лучшую прогностическую способность. В нескольких приложениях, в основном в том числе Data Mining, набор данных, который нужно изучить, очень велик. В этих случаях очень желательно построить одномерные деревья решений в разумные сроки. Это может быть достигнуто путем параллелизации одномерных алгоритмов дерева решений.

Ключевые слова: *гиперсингулярное интегральное уравнение, численное решение, компьютерный эксперимент, задача дифракции, параллельность, ID3, узел.*

Дерева рішень - це добре встановлений набір методів класифікації, визнання та підтримки прийняття рішень при машинному вивченні, ідентифікації, аналізу даних, ситуаційного управління. Дерево рішень має бути компактним - це економить витрати при відповіді на запитання; Крім того, компактні дерева мають кращу прогностичну здатність. У ряді додатків, в основному, включаючи Data Mining, набір даних, який потрібно вивчити, дуже великий. У цих випадках дуже бажано побудувати однорічні дерева рішень в розумний час. Це може бути досягнуто шляхом розпаралелювання одномірних алгоритмів дерева рішень.

Ключові слова: *гіперсингулярне інтегральне рівняння, чисельний розв'язок, комп'ютерний експеримент, задача дифракції, паралельність, ID3, вузол.*

1 Introduction

Regression trees and classifications, also known under a common name as decision trees, are data structures that allow interpreting data for study purposes. Because decision trees are so easy to interpret, they are among the most widely used data-mining methods in business analysis, medical decision-making, and policymaking. Often, a decision tree is created automatically, and an expert uses it to understand the key factors and then refines it to better match her beliefs. This process allows machines to assist experts and to clearly show the reasoning process so that individuals can judge the quality of the prediction. Decision trees have been used in this manner for such wide-ranging applications as customer profiling, financial risk analysis, assisted diagnosis, and traffic prediction. [1] Decision trees can be drawn by

© Parshentsev B. V., Tolstoluzhskaya E. G., 2017

hand or created with a graphics program or specialized software. Informally, decision trees are useful for focusing discussion when a group must make a decision. Programmatically, they can be used to assign monetary/time or other values to possible outcomes so that decisions can be automated. Classification is the task of learning a target function f that maps each attribute set x to one of the predefined class labels y. The target function is also known informally as a classification model. A classification model is useful for the following purposes. [2]

Each tree has three types of nodes. Root node that has no incoming edges and zero or more outgoing edges. Internal nodes, each of which has exactly one incoming edge and two or more outgoing edges. Leaf or terminal nodes, each of which has exactly one incoming edge and no outgoing edges. The aim is to obtain a classification tree of symbolic notation class and, by the use of regression trees, continuous values are returned. For a decision tree you need to provide sample data, so you need to pre-assemble the data. On the one hand, the data may be prepared by an expert, and on the other hand, the accumulation of the facts relating to this task may be envisaged. Various applications can be created on the base of the purely interpretable data. In practice, each element under consideration can be represented by a set of attributes against which the decision tree predicts an unknown attribute.

Tree solutions can be used as a component responsible for making decisions on the bank accounting. Each situation is represented as a set of attributes, so in each particular situation the decision tree can suggest the best way of doing things. Furthermore, regression trees could be used to evaluate the strengths of any object (or to predict the outcome, both positive and negative). Tree solutions are divided into two types. The application of classification trees leads to categorical responses, and the regression trees return continuous values. In each node decision-making provides verification of the condition determined on the variables of forecasting. The number of ribs corresponds to the number of possible verification results. The tree levels form a hierarchical structure. Sheet nodes return answers, either categorical or continuous. The simulation algorithm uses a sample of data for passing through the tree according to the results of each condition verification. The tree training algorithm acts on the basis of the recursive partition principle on non-overlapping sets based on the use of the most suitable variable for this, which leads to the receipt of subsets containing the minimum number of third-party inclusions.

The corresponding decision node is created in the tree, and the process continues recursively. The best way to improve training is to make the dataset more manageable by using additional verification kits. The classification model presented in the form decision tree is intuitive and simplifies the understanding of the problem being solved. The result of the work of algorithms for constructing decision trees, in contrast, for example, from neural networks, which are "black boxes", is easily interpreted by user. This property of decision trees is not only important when referring to defined class of the new object, but also useful in interpreting the model classification as a whole. The decision tree makes it possible to understand and explain why a particular object belongs to a particular class. The decision trees make it possible to extract rules from the database on a natural language. Example rule: If Age> 35 and Income> 200, then issue a loan. The decision trees make it possible to create classification models in those areas where it is difficult to formalize knowledge. The algorithm for constructing the decision tree does not require the user to choose input attributes (independent variables). The input of the algorithm can be fed all existing attributes, the algorithm itself will choose the most significant among them, and only they will be used to build a tree. In comparison with the neural networks, for example, this greatly facilitates the user's work, because in the neural networks the number of input attributes affects the learning time significantly. The accuracy of models created with decision trees usage is comparable to other methods of constructing classification models (statistical methods, neural network). [3]. First introduced in 1960, the decision trees is one of the most effective methods of data mining; they are widely used in several disciplines. [4] An example of the medical use of the decision trees is the health diagnosis based on the symptoms, in which classes defined by the decision tree, can be either different clinical subtypes, or condition, or patients with a condition that should receive different treatment. [5]

2 ID3

The iterative algorithm of Dichotomiser 3 (ID3) is one of the most popular tree induction design solutions. It is not tolerant of missing values or noisy, and the value of attributes must come from an infinite fixed set. ID3 uses entropy to calculate the sample homogeneity, as well as for separation. Information coefficient G for each attribute A is calculated using the following equation: to the last root of the tree the highest rate of information gain is assigned. Then the new subtree is built recursively upon each value of the attribute bound to the root. ID3 (C4.5 and CART) builds a tree induction solutions in descending recursively divide and overcome the way through the space of possible decision trees using greedy strategy. The decision is taken at each step, which is done for the optimizations purposes. For each node, locate the test the condition of the best segment assigns a workout data. Features tree induction solutions in the case include ID3 the following:

1. Each node excluding the leaf of the tree corresponds to an input attribute, each arc to a possible value of that attribute
2. Entropy is used to determine how informative a particular input attribute is about the output class on a given dataset
3. The recursive algorithm [6]

To evaluate the information gain associated with a split of T based on attribute ai, we must determine the total count of positive and negative examples as well as the number of positive and negative instances for each value of attribute ai. The computational complexity of ID3 (for categorical attributes) at each node of the decision tree is (jNj jAj), where N is the number of examples and A is the number of attributes examined at the node. ID3 time requirements indicate that the total cost of the algorithm is superquadratic as for the set of workouts size. Obviously, the use of continuous data largely coincides with the computational time required to build a decision tree. To accelerate the the examination of candidates (and corresponding frequency calculation amount), ID3 firstly sorts the training examples using a continuous attribute as the sort key. It is the sorting operation, which increases the computational requirements to (N log2N), that leads to potentially exorbitant CPU time for large training sets. There is the solution to the above problem is called peepholing; the basic idea is to discard a sufficient number of candidates for threshold values so that the computational expense of sorting is lessened. It is (approximately) an intelligent sampling of the candidates that aims to create a small “window" of threshold values; this window is then sorted as usual. Empirical results have shown that peepholing produces significant improvements over the traditional ID3 algorithm for several large training sets, however there is no guarantee that this approach will perform with consistent accuracy over all possible domains.

The main advantage of ID3 is its simplicity. Due to this reason, ID3 algorithm is frequently used for teaching purposes. However, ID3 has several disadvantages:

1. ID3 does not guarantee an optimal solution; it can get stuck in local optimums because it uses a greedy strategy. To avoid local optimum, backtracking can be used during the search.
2. ID3 can overfit to the training data. To avoid overfitting, smaller decision trees should be preferred over larger ones. This algorithm usually produces small trees, but it does not always produce the smallest possible tree.
3. ID3 is designed for nominal attributes. Therefore, continuous data can be used only after converting them to nominal bins. ID3 is designed for nominal attributes.

Therefore, continuous data can be used only after converting them to nominal bins.[6]

3 Parallel Tree

Panda et al. (2009) proposed a parallel implementation of tree induction with MapReduce framework. MapReduce is one of the most popular parallel programming frameworks for data mining. In this framework, programmers need to specify a map function which processes key/value pairs to emit a set of intermediate key/value pairs and a reduce function that aggregates all intermediate values associated with the same intermediate key. The MapReduce framework was pioneered by Google and then popularized by open-source Apache Hadoop project. While there are other parallel programming frameworks (such as CUDA and MPI), MapReduce has become the industry standard and implemented on cloud computing services such as Amazon EC2 and various companies such as Cloudera provide services to ease Hadoop deployment [6]. Among the classification methods, the ID3 is computationally inexpensive - at first glance, it may not seem appropriate to apply parallelism to get benefit at runtime. However, when applied to huge amounts of data, you can expect that the maximum size of the learning set will require non-trivial amounts of computation. In addition, you can use the total available memory of multiprocessors with distributed memory or workstation clusters to accommodate ever-increasing sets of training materials that may not be possible on individual machines. Model-driving parallelization strategy to tree construction seems to be the most natural strategy of assigning processing elements to nodes of the decision tree and reflects the “divide and conquer" nature of the algorithm. Although appropriate to many search strategies such as branch-and-bound, the limitations of this approach when applied to tree construction become apparent. It is difficult to partition the workload among available processors (as the actual workload is not known in advance) - if the least loaded processor will be chosen, the processor assigned the middle branch of the root of the tree will complete first and will be idle. Alternatively, a “master-worker" scheme for a task assignment, where the available processors are assigned to the task of determining the best attribute for splitting of a single node and then is returned to a “waiting pool" may exhibit excessively fine-grained parallelism. The overall computation time may be influenced by the overhead of the computation time or by the overhead of approaches.

The efficiency is limited by the fact that the bulk of the computational work is associated with the root of the decision tree, where the entire training set T as well as all m-attributes must be examined to determine the subsets represented in the immediate children. Attribute based decomposition is another strategy that associates m processing elements with each attribute in X so that the evaluation of gain for all m-attributes can proceed concurrently. Pearson evaluated the performance of a combination of the "master-worker" approach and attribute based decomposition to compensate for growing workloads at lower levels of the decision tree and the accompanying increase in the ratio of parallel overheads to the "useful" work. Pearson conclusion that "none of the results show a speed reduction (in proportion) with the possible parallel computation" highlights the shortcomings of this strategy. A further limitation of this approach is the shared-memory model that must be assumed to allow global access to training examples by the individual tasks; this assumption severely limits the scalability of the algorithm (Fig. 1).

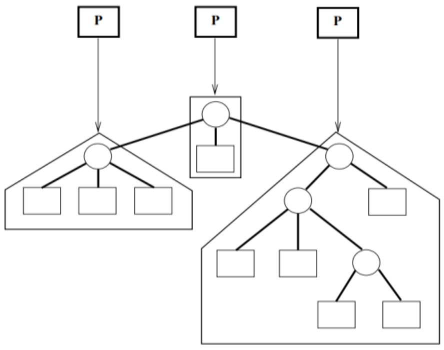


Figure 1. model-driven model.

An alternative method of parallelizing tree construction employs a data-parallel approach and is depicted in Fig. 2. [7]

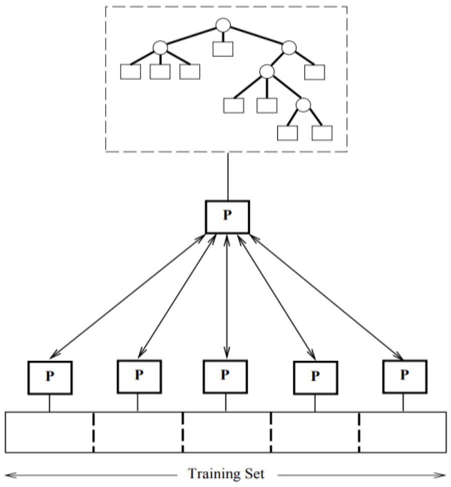
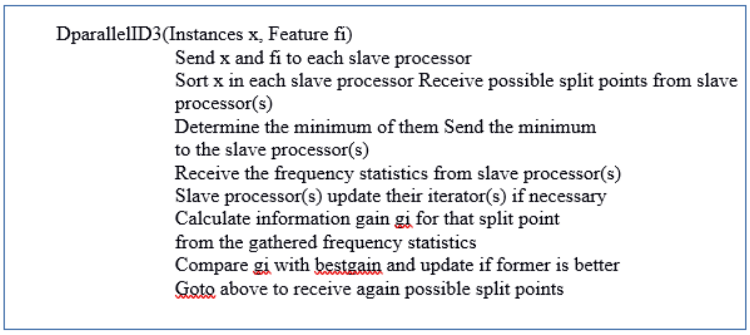


Figure 2 Data-parallel model.

The motivation behind this decomposition strategy arises from the observation that most trees construction algorithms rely on frequency statistics derived from the data itself. In particular, the fundamental operation in ID3-like algorithms is the counting of the attribute value/class membership frequencies of the training examples. Parallel Decision Trees is a strategy for dataparallel tree construction. The machine model employed assumes the availability of P processing elements (PE), each with associated local memory. The interprocessor communication primitives required are minimal: each PE must be able to send a contiguous block of data to its nearest neighbor; additionally, each PE must be able to send data to a distinguished \host"processor. This machine model is general enough so that the strategy may be employed on currently-available massively parallel systems as well as networks of workstations. Because the communication patterns involved are regular, with the bulk of transfers involving only nearest neighbor Pes, the additional overhead incurred due to inter-processor communication is kept to a minimum.



```
DparallelID3(Instances x, Feature fi)
            Send x and fi to each slave processor
            Sort x in each slave processor Receive possible split points from slave
            processor(s)
            Determine the minimum of them Send the minimum
            to the slave processor(s)
            Receive the frequency statistics from slave processor(s)
            Slave processor(s) update their iterator(s) if necessary
            Calculate information gain gi for that split point
            from the gathered frequency statistics
            Compare gi with bestgain and update if former is better
            Goto above to receive again possible split points
```

Figure 3. Parallelization (pseudocode)

As attributes are chosen for splitting criteria associated with internal nodes of the decision tree, the master broadcasts the selected criterion to worker processors, who use this information to “mask" training events that do not belong to the subset of cases that are examined at lower levels of the tree (Fig. 4).

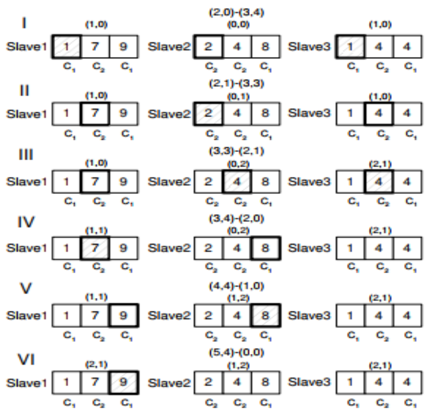


Figure 4. Parallelization example

Tree partitions the entire training set among the available PEs so that each processor contains within its local memory at most [N/P] examples from T. This partitioning is static throughout tree construction and subsequent pruning. No examples are allocated to the “host" processor, which is instead responsible for:

- Receiving frequency statistics or gain calculations from the “worker" PEs and determining the best split.
- Notifying the PEs of the selected split at each internal node.
- Maintaining the data structures for the tree itself.

Conclusion

The ID3-like algorithms, induce, axis-parallel, decision trees, in which each test one attribute value is equivalent to a hyperplane parallel to the size attribute in object space. Recently, methods for induction of oblique decision tree, such as Oblique Classier 1 (OC1) have been developed which have the potential of capturing a more general class of concepts more succinctly and accurately than their axis-parallel counterparts OC1 use ascent and randomization to identify inclined splits training data. Because OC1's search starts at an axis-parallel split that is subsequently perturbed into oblique orientations to find better partitions, the strategy outlined by tree can benefit algorithms such as OC1. ID3 is a non-incremental (or batch) inductive procedure; all training examples are required prior to construction of the classier. Several modifications to ID3 have been proposed to provide for an incremental variant of the algorithm. ID5P tree is an example of the incremental method, which increases the decision tree frequency counter associated with each non-leaf node. Information needed for the construction of the original tree, is available in our parallel formulation. Thus, the "bootstrap" structure can benefit from parallel structure using PDT method, and the resulting tree can be modified as new examples are presented.

ЛИТЕРАТУРА

1. Pang-Ning Tan, Vipin Kumar. Introduction to Data Mining. Introduction to Data Mining 1st Edition 2014 – 146p-154.
2. Hastie TJ, Tibshirani RJ, Friedman JH. The Elements of Statistical Learning: Data Mining Inference and Prediction. Second Edition. Springer; 2009.
3. Data Mining: учебное пособие /И.А. Чубукова.- 2-е изд., испр. – М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2008. — 100с
4. Hastie TJ, Tibshirani RJ, Friedman JH. The Elements of Statistical Learning: Data Mining Inference and Prediction. Second Edition. Springer; 2009. 74p.
5. Bater M. Learning Data Mining with R.2015. 79p
6. Lior R. Data Mining with Decision Trees: Theory and Applications. 2014 Dec. 79 p
7. Kesav P., Utpal B., David G., Alex N. Languages and Compilers for Parallel Computing. LCPC 1994 155-169 p.

Надійшла у першій редакції 03.12.2017, в останній - 20.12.2017.