ISSN: 0975-766X
CODEN: IJPTFI
Available Online through *Research Article*
www.ijptonline.com

A TECHNIQUE FOR CLASSIFYING MASSIVE DATASET USING PARALLEL ID3 APPROACH

Lokesh Sharma, Nancy Victor
School of Information Technology and Engineering, VIT University, Vellore.
Email: tiwarilucky92@gmail.com

Received on 25-10-2016 *Accepted on 02-11-2016*

Abstract

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to predict the target class for each case in the data. ID3 algorithm is used to generate a decision tree. Decision tree breaks down a data set into small subsets and tree consists of decision nodes and decision leaf nodes. Nodes can have two or more branches which represents the value for the attribute tested. Leaf nodes produces the final result. Decision tree generation for classification of massive data set is a very tedious task. Use of parallel processing and multi processor environment can be used to decrease the time in decision making. A detailed study and implementation of parallel ID3 algorithm for data classification is given in this paper. The parallelization of ID3 has been done using parallel programming approaches by multi threading in java. The information calculation as well as the decision tree generation has been parallelized, thereby reducing processing time for decision tree generation. This parallel algorithm is tested on standard data sets of varying number of records and attributes.

The results of classification with sequential and parallel decision tree generation is implemented. There is an overall decrease in the decision tree generation time and classification of attributes in a large data set. So the algorithm establishes a relationship between the constructing times.

I. Introduction

One of the most studied and generally used method for classification is ID3 algorithm. This algorithm is difficult to execute and utilizes a decision tree as data structure. Decision Tree induction is the learning of decision trees from class labelled training tuples. ID3 algorithm produce the results on the bases of entropies and information gain about the data set. It takes all unused attributes and calculate their entropies andInformation gain is based on the decrease in Entropy after a data set is split according to attributes. Theone with the highest information gain for classification is selected.

$$\text{Entropy-}(S) = -p \log_2 p - q \log_2 q$$

Information Gain (S, A) = Entropy-(S) – Entropy- (S, A)

A decision-tree is a flowchart like tree structure, where each internal node (a non-leaf node) represents the value for the attribute tested, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label. The topmost node in a tree is the root node. Leaf nodes produce a homogeneous result. This decision-tree once created for a training data-set or historical data can be used to classify the newly coming data into some unique class. Implementation of ID3-algorithm using sequential approach in java, takes more processing time while generating the decision tree. In this paper we introduce ID3-algorithm using a parallel-approach using java. We use multi-threading while generating the decision tree. By using multithreading, we reduce the decision-tree generation time. We have established relationship between the generating times of the decision-tree through Sequential and parallel-processing techniques.

II. Related works

Decision-tree classification algorithm can be implemented-in a serial or parallel fashion based on the volume of data, memory space available on the computer resource and scalability of the algorithm. Various approaches have been proposed-that parallelize the decision-tree generation for ID3 [1] [2]. All of these actually have given good solutions for parallelization by calculating the entropy-simultaneously for all the attributes, but the main advance we are looking in is that, with the advantage of parallelizing entropy-calculation-we can also parallelize the tree generation process i.e. the decision-tree is generated as soon as we get entropies of all the attributes, which refers to the strategy where to bring out parallelization-in the decision -tree algorithms. This paper proposes two algorithms, one is based on synchronous tree construction approach and the other is based on partitioned tree construction approach also a hybrid approach which utilizes advantages of both methods. Another-paper presents similar work in parallelization of C4.5 decision-tree algorithm for continuous-valued attributes [3]. The complexity of the sequential algorithm is substantially reduced. The new algorithm has a time complexity of O (MN), where m is the size of the training data and n is the number of attributes. One of the methods exploits two levels of divide-and-conquer parallelism in the tree builder, at the outer level across the tree nodes, and at the inner level within each tree node. For this, lightweight parallel threads are used to express highly irregular and dynamic parallelism in a natural-manner. The shared memory approach for parallelization is initiated in the paper. Each of the P processors begins with 1/P of the data instances, but the partitioning can soon become highly unbalanced. So the work done per thread is very-small in comparison to the approaches-implementing sequential algorithm. The majority of research papers published in this area employs the method of generation of decision-tree using threading. Here, we look-for increased advantage of shared memory

concept and multi-threading environment provided by java. We also explore new feasibilities in the ID3-algorithm to bring optimizations.

III. ID3 Decision Tree Generation

ID3-algorithm basically works on decision-tree approach. ID3 split the whole-the data set based on attributes. Each splitting is based on one nominal feature and considers its complete domain. Calculate the Entropy-values of each attribute. Entropy-value means frequency-of the attribute. Information gain is calculated using the entropy value. After that generate the decision-tree according to attributes of the data-set. In decision-tree root nodes contains-attributes and leaf nodes contains the results or final decisions.

ID3-algorithmPseudo code:

```
id3(examples, attributes)
  Node = Decision Tree Node (examples)
  Dictionary = Summarize Examples (examples, Target-attribute)
  For key in dictionary:
    If dictionary[key] == total number of examples
      Node. Label = key
      Return node
  If attributes are empty or number of examples < minimum allowed per branch:
    Node. Label = most common value
    Return node
  Best A = the attribute with the most information gain
  Node. Decision = best A
  For each possible value v of best A:
    Subset = the subset of examples that have value v for best A
    If subset is not empty:
      Node.add Branch (id3(subset, Target-attribute, attributes-best A))
  Return node
```

Parallel ID3-Algorithm:

ID3-algorithm is already implemented in sequential processing. It is an algorithm which is basically useful for doing some-prediction on the basis of training data-set. It will generate decision-tree according to target-attribute. Parallel

ID3-algorithm is implemented to establish relationship between the generating times of decision -tree through Sequential and parallel processing techniques.

Begin:

Create a Root node for the tree.

If all data sets have the same value of the target-attribute, return the single node tree Root with

Label = the value of target-attribute.

If Attributes is empty, Return the single node tree Root with label = most common value of

Target-attribute in Examples.

Else

Begin

Let A = the attribute from Attributes that best classifies data-set.

Let the decision attribute for Root = A.

For Each possible value of A

Assign to multi-threading

Add a new tree branch below Root corresponding to the test A = value.

If data set is empty, then.

Add a leaf node with label = most common value of Target-attribute in data set to the new

Branch.

Else

Add a subtree ID3 (data-set, Target-attribute, Attributes– {A}) below this new branch.

End For

End

Return Root

End

Steps of Parallel Approach

Load Data-set:

Training dataset contains whole data in the form of text. Training data set is an input for the algorithm. In this project we worked on tennis dataset. Tennis data-set is in form of text it contains information about the weather. Through

given information in the dataset we are predicting the results like whether the current weather is suitable for tennis match.

Split-Data:

Here, the data set is split into various small data sets according to the attributes. After splitting calculate the entropy of each attribute and on the basis of entropy, information gain is to be found.

Assign to Parallel java:

In this step, processes would be assigned to multiple threads in parallel java. So it takes less time to process. After calculating the values of entropy-and information gain, the data is assigned to the multiple threads for generating the decision trees [5].

Generate sub-trees:

This is the main step of our project. Sub trees are generated in this step. Here tasks should be divided into numbers of small tasks and should be allocated to each thread. Each threads starts working on task which is allocated. Here, all threads-works parallel and produce result separately. The use of lightweight threads also allows us to express a large amount of parallelism, so that the load can be dynamically balanced by the thread-implementation [4].

Join sub-trees:

Here, the various sub trees generated are combined together to generate the final decision tree using the parallel ID3 approach.

Generate the final decision tree:

On the bases of all sub-trees here we will generate final decision tree. As a parallel technique is adopted for generating the decision trees, this would be very fast when compared to the sequential approach.

Comparison:

The basic ID3-algorithm was first implemented using a sequential approach. The time complexity is also calculated for generating the tree. After that, the modified parallel ID3 algorithm is implemented which proved that it takes less time to generate the final decision tree when compared to the basic sequential approach.

Sequential Decision -Tree computation time:

```
public void print Statistics () {
System.  out. Println ("In Sequential approach, Time to construct Decision  Tree = “+ (eTime – sTime) + “ms");
    System.  out. Println ();
    System.  out. Println ("Target-attribute=“+ All Attributes [index Target Attribute]);
```

```
    System.  out. Print ("Other attributes = ");
    For (String attribute: all Attributes) {
    If (! attribute. Equals (all Attributes [index Target Attribute])) {
                System.  out. Print (attribute + " ");
    }
        }
System. out. Println ();
}
```

Parallel Decision -Tree computation time:

```
    Public void print Statistics () {
System.  out. Println ("In Parallel JAVA approach, Time to construct Decision Tree = "+ (eTime – sTime) + " ms");
    System. out. Println ();
    System. Out. Println ("Target-attribute = "+ all Attributes [index Target Attribute]);
    System. out. Print ("Other attributes = ");
                For (String attribute: all Attributes) {
If (! attribute. Equals (all Attributes [index Target Attribute])) {
    System. out. Print (attribute + " ");
        }
        }
    System. out. Println ();
        }
```

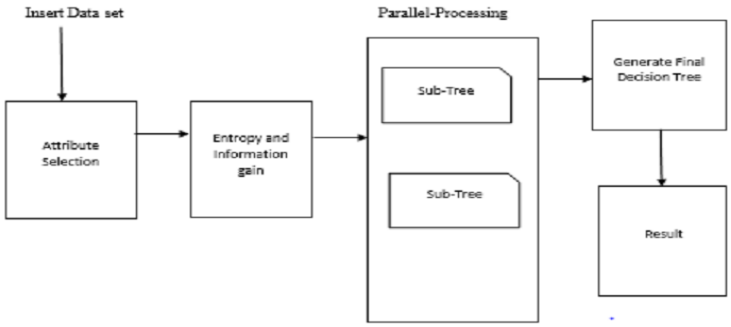
IV. Parallelism in Java:

Parallelism is a technique in java which is useful to perform more than one task at a time. Parallelism means that an application splits its tasks into smaller subtasks which can be processed in parallel, for instance on multiple CPUs at the exact same time. In java, parallelism can be done by multi-threading approach. Multi-threading is a collection of more than one threads which works together.

Multi-threading split task into subtasks which can be completed in parallel. In our project, we are generating decision-tree by using multi-threading. We allocated generating sub-trees with the help of multithreads. Each thread

is generating some piece of tree. This process can be done by multi-threading so the whole process is executing at the same time.

Block Diagram of Parallel ID3



V. Results and Discussion

The basic ID3-algorithm was first implemented using a sequential approach. The time complexity is also calculated for generating the tree. After that, the modified parallel ID3 algorithm is implemented which proved that it takes less time to generate the final decision tree when compared to the basic sequential approach. Joining the threads may be complex in the case of large amount of attributes and massive data-set, but multi-threading itself involves synchronization between the threads.

In this algorithm, we have worked on the tennis data set. It contains five attributes and sixty-six data records. The processing time taken by the program to display the results either in the form of decision tree or the decision rules in the form of if-then-else clause is identified [6]. The programs have been tested on a Pentium Intel Centrino Architecture with CPU speed 2.21 GHz and 2GB of shared main memory. In this program we used two threads for generating sub-trees. Each thread works individually without disturbing the other thread. While generating the decision tree through sequential approach we got 40 milliseconds of time to generate a final decision tree. The parallel ID3 approach took only 1 ms of time to generate the final decision tree.

```
run:
File Addtess =  tennis.txt
File Addtess =  tennis.txt
Sequential, Time to construct decision tree = 40 ms

Parallel JAVA, Time to construct decision tree = 1 ms

Target attribute = play
Other attributes = outlook temp humid wind
Sequential -> DECISION TREE
   outlook->
     sunny
       humid->
         normal
             =yes
         high
             =no
     overcast
         =yes
     rain
       wind->
         strong
             =no
         weak
             =yes
```

VI. Conclusions

This algorithm is an approach to improve the prevalent ID3 decision-tree generation algorithm using the multiprocessing technique. The results show the desired effect on the tree-generation time. The number of records and attributes has significant impact on the processing time. The scheme can be improved further if we use parallel and distributed-processing architecture i.e. the tree-generation is done on different systems placed at distant places and then the whole tree is generated at particular system serving as the master computer. This makes the algorithm architecture dependent but would surely reduce the processing time further due to the scheduling of segments of task into several processors and systems. There is an overall decrease in the decision tree generation time and classification of attributes in a large data set. Computing time would be reduced as compared to the sequential-processing techniques.

VII. Reference

1. Jin, Chen, Luo De-lin, and Mu Fen-xiang. "An improved ID3 decision tree algorithm." Computer Science & Education, 2009. ICCSE'09. 4th International Conference on. IEEE, 2009.
2. Halls, L.O., Bowayer and Chawla, N., K.W.1998 Decision Tree Learning on Very Large Data-sets International Conference on Cybirneticsend Systems MAN.
3. Ruggieri, Salvatore. "Efficient C4. 5 [classification algorithm]." IEEE transactions on knowledge and data engineering 14.2 (2002): 438-444.
4. H.Su,Zhang, and J. A Fast Decision -Tree Learning Algorithm, American Association for Artificial Intelligence
5. Singh Han, E., Kumar, V. and, Srivastava, A., Parallel Formulations of Decision-Tree Classification Algorithms International Conference on Parallel Processing.
6. Chen, X. S., Liu,Dan, X. R., C. F. and Liu, D. W. Improvement and Research on the Decision -Tree Classification Algorithm of Data mining Software Guide.