ATLANTIS PRESS

Advances in Intelligent Systems Research, volume 134

2nd International Conference on Control, Automation, and Artificial Intelligence (CAAI 2017)

A Parallel Decision Tree Based Algorithm on MPI for Multi-label Classification Learning

Yihao Zhou, Zhenzhou Ji* and Kaiyu Wang

Department of Computer Science and Technology Harbin Institute of Technology Weihai, 264209, China

*Corresponding author

***Abstract*—Multi-label classification is an important area of data mining, in where decision tree is one of the effective means to solve the problem. It faced a huge challenge of performance caused by large size of data. First, we translate the multi-label classification to several binary classifications. Then we analyzed the potential parallelism of decision tree based multi-label classification algorithm from four parts and overall applied them in the training and predicting phases. The parallel algorithm was implemented with MPI and the performance of parallel decision tree based multi-label classification algorithm is analyzed and compared program designations and experiments, which demonstrate that our parallel algorithm could improve the computing efficiency and still has some extensibilities.**

Keywords-multi-label classification; decision tree; parallelization; MPI

I. INTRODUCTION

With the rapid development of Internet, the amount of the data processed by data mining algorithm is increasing, so that, any single machine is hard to deal with this amount of data. We are facing critical demands of performance, in this case, the High Performance Computing and parallel computing is becoming a hot research area. In parallel computing, there is a famous tool called MPI (Message Passing Interface). It is a standardized and portable message-passing system designed by a group of researchers from academia and industry to function on a wide variety of parallel computing architectures. Because of its high-performance, portability and extensibility, MPI is widely used in the implementation of several area of parallel algorithms improving.

Classification problem is one of the classical issues of machine learning, and is a supervised learning framework. The algorithm learns a classifier from the training set. The classifier learn from the training set can be used to predict label of an unseen instance. However, one real world object can never be labeled by single label, they have multiple semantics, which means that an instance can be associated with a set of labels simultaneously[1]. Multi-label classification algorithm can be widely applied to diverse problems from automatic annotation for multimedia contents to bioinformatics, web mining, rule mining, information retrieval, tag recommendation, etc.

II. THEORY OVERVIEW

A. Classification Learning Framework

Classification learning problem is a classic supervised learning problem. In this learning problem, every object in the real world is an instance which is represented by a multi-dimension feature vector extracted from the object, every dimension represents an attribute of feature. In the meantime, instance is associated with a label represent its semantics. Instance feature vector and its label generates an example, furthermore, thousands of examples contribute to a training set. Classification learning framework is mainly made up by two steps. First, classifier training: algorithm processing the training set to generate a classifier represented by a certain model. Then, unseen instance prediction: our algorithm predicts label of an unseen instance through the classifier trained in form step. Especially, the classification learning problem we researched is binary classification learning problem, which means every label only has two value (0, 1) for whether the instance has the semantics represented by the label.

Formally, let x denote the instance space and y denote the label space, the task of classification learning algorithm is to learn a function $f: x \to y$ from the training set $\{(x_i, y_i) | 1 \le i \le m\}$. Here, x_i is an instance characterizing the features of an object and y_i is the corresponding label characterizing its semantics. For the unseen instance x_{unseen}, we predict its label $y_{predicted}$ by the above classifier.

B. Decision Tree Based Classification Learning Algorithm

Decision Tree is a simple and widely used classifier, Decision tree learned from training set can predict labels of unseen instances effectively, and mainly has two advantages: (1) decision tree is easy to read, describe and understand, which means it can be helpful in manual analysis of masters; (2) High efficiency, the algorithm induces the decision tree once that can be used repeatedly and the computation times won't exceed the maximum of depth of the decision tree.

The process of decision tree training is the process of generating tree node recursively. ID3 and C4.5 are two of the traditional decision tree training methods, formed can only process discrete attribute and the latter can deal with both discrete attribute and continuous attribute. Decision tree training process is as following:

Step 1 Making the entire training set as the root node of our decision tree.

Step 2 Trying every attribute and its value as the split condition and finding the best split condition.

Step 3 Splitting data set to Left and Right tree child tree node when the split condition is continuous attribute and splitting data set to m child tree nodes when the split condition is discrete attribute and the attribute has m distinct values.

Step 4 Repeating operations in ***Step 2*** and ***Step 3*** to the child tree node until ending condition raised.

In the above process, finding the best split condition means finding a value of continuous attribute or a discrete attribute which makes minimum $gini\ index$[2] as Eq(1) and Eq(2). Let n denotes the size of the training set, n_i denotes the number of examples in tree node i, n_{ij} denotes the number of examples share the label j in tree node i, c denotes the number of distinct values of the label and d denotes the number of child tree nodes.

$$gini_i = 1 - \sum_{j=1}^{c} (n_{ij}/n_i)^2 \qquad \text{Eq(1)}$$

$$gini_{split} = \sum_{i=1}^{d} (n_i/n) \qquad \text{Eq(2)}$$

As for continuous splitting attribute, the splitting value split the training set to two child node, examples goes Left child tree node when the attribute value is lower than splitting value and other examples goes Right child tree node. As for discrete splitting attribute, we only define one way to split training set which is splitting training set to the child tree nodes having the exact same value.

As we can see in the FIGURE I., decision tree is a binary tree or non-binary tree. Tree nodes can be dived into inner nodes and leaf nodes. Inner node has a splitting condition and leaf node has a label value. When the unseen instance go through the inner node(s) and finally arrive the leaf node, the prediction of an unseen is made according the label value stored in the leaf node. In another perspective, every path from root node to leaf node is a classification.

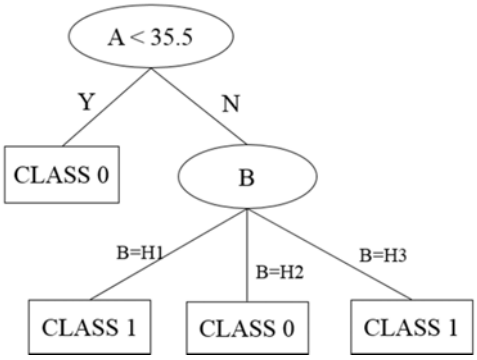

FIGURE I. DECISION TREE

C. Multi-label Classification Learning Framework

Multi-label classification learning framework is an extension of the classification problem mentioned in II.*A.* , and is more closed to the real world situation. Different from the single-label classification problem, every example is associated with several labels(binary label) simultaneously as well as every real world object has multiple semantics from several aspects. The multi-label classification learning algorithm can predict a set of labels for an unseen instance.

Formally, let x denote the instance space and y denote the label space, the task of classification learning algorithm is to learn a function $f: x \to y$ from the training set $\{(x_i, y_i) | 1 \le i \le m\}$. Here, x_i is an instance characterizing the features of an object and y_i is the corresponding label set characterizing its semantics. For the unseen instance x_{unseen}, we predict its label set $y_{predicted}$ by the above classifier.

Due to the similarity of multi-label classification learning framework and single-label classification learning framework, the methods used to solve single-label classification problem can also be used to deal with multi-label classification problem. Usually, there has two way, one is "problem transformation" method which means fit data to algorithm; another is "algorithm adaptation" method which means fit algorithm to data[1]. The former way can be applied in decision tree based algorithm and has two branches, one is splitting the multi-label classification problem to b independent single-label classification problem, b is the number of labels[3]; another is translating the multi-label classification problem to multi-class classification problem, i.e. in the multi-class classification problem, the only label has 2^b distinct discrete values. But, if the original problem has too many labels, the only label in the multi-class classification problem will has too many distinct values, which means the size of the problem are exponentially bigger than original problem. And this kind of problem are not easy to parallelize, so we choose the former one to change the multi-label classification learning problem to several single-label classification learning problem.

III. PARALLEL MULTI-LABEL CLASSIFICATION LEARNING ALGORITHM BASED ON DECISION TREE

A. Analysis of Parallelism

As we all known, decision tree algorithm has potential parallelism in many sides, and multi-label classification algorithm based on decision tree inherit these parallelism. We have two important prerequisites, which is that the relationships between labels and relationships between attributes are not under our consideration[1,6]. This two prerequisites make it easier to parallelize the decision tree based multi-label classification algorithm. Then main parallelism can be induced as follows:

- Because of the prerequisites of labels relationships, we can translate the multi-label classification problem to several single-label classification problem by "problem transformation" method. And then we can parallelize the training process of every decision tree corresponding to every single-label classification problem.

- Because of the prerequisites of attributes relationships, we can parallelize the process of find best splitting condition especially for continuous attributes. However the algorithm need to broadcast all the data to all processing nodes which will exceed the communication overhead.

- Since our decision trees were trained at different processing node, label prediction can also be done parallelized.

- When the large training set comes, we can distribute large data to every processing node averagely. There are two ways to do this which are horizontal distribution and vertical distribution. The former one is distribute examples to processing

nodes averagely, the latter one is distribute attributes to processing nodes averagely.

B. Algorithm Implementation

Through the above analyzing of algorithm parallelism, we find many ways to parallelize the multi-label classification algorithm based on decision tree. This paper intend to use Coarse-Grained parallelization strategies and propose a parallel decision tree based multi-label classification algorithm and implement the algorithm on MPI.

Before all, we introduce a data structure called attribute table which was used in SPRINT[4] and ScalParC[5]. The attribute table is an array of tuple, and the tuple is made up of example id(rid), attribute value(value) and label set(cid1, …,cidn). FIGURE II. and FIGURE III. shows the attribute table.

rid	value	cid1	…	cidn
0	56.8	0	…	1
1	10.9	1	…	0
2	3.2	0	…	1
3	35.5	1	…	1

FIGURE II. ATTRIBUTE A LIST

rid	value	cid1	…	cidn
0	H1	0	…	1
1	H3	1	…	0
2	H3	0	…	1
3	H2	1	…	1

FIGURE III. ATTRIBUTE B LIST

rid	value	cid1	…	cidn
2	3.2	0	…	1
1	10.9	1	…	0
3	35.5	1	…	1
0	56.8	0	…	1

FIGURE IV. ORDERED ATTRIBUTE A LIST

The flow chart of the algorithm is shown as the FIGURE V., and the steps are as follows:

Step 1 Initialization. Initializing MPI environment and reading label definition information and training set.

Step 2 Pre-sorting. First, the algorithm build attribute tables for every attribute of examples. Then we sort the continuous attribute tables and FIGURE IV.shows the ordered continuous attribute table.

Step 3 Parallel Training. We translate the original data to several single label data, and then put the task of learning decision tree from single label examples to every MPI processing nodes. i.e. task of label i is send to then processing node $i\%number\_node$, in which $number\_node$ is the total number of MPI processing nodes. Each node trains the decision tree with depth-first strategy and the process is similar to the process described in II. *B.* Still, we have an improvement. In ***Step 2***, examples in training set were already split to several attribute tables which are ordered by attribute value. Then, the process of finding best splitting condition can be done by scanning every attribute table twice. First time, the algorithm gets the label value distribution of every example; second time, the algorithm tries every attribute value as the splitting value and find the best one according to the minimum $gini\ index$.

Step 4 Prediction. Through the above steps, we built decision trees for every single-label classification problem split from the original problem on all the MPI processing nodes. We'd rather predict labels according to the decision trees on all MPI processing nodes than gathering all decision trees to the master node, and then gather the predicted labels to master node.

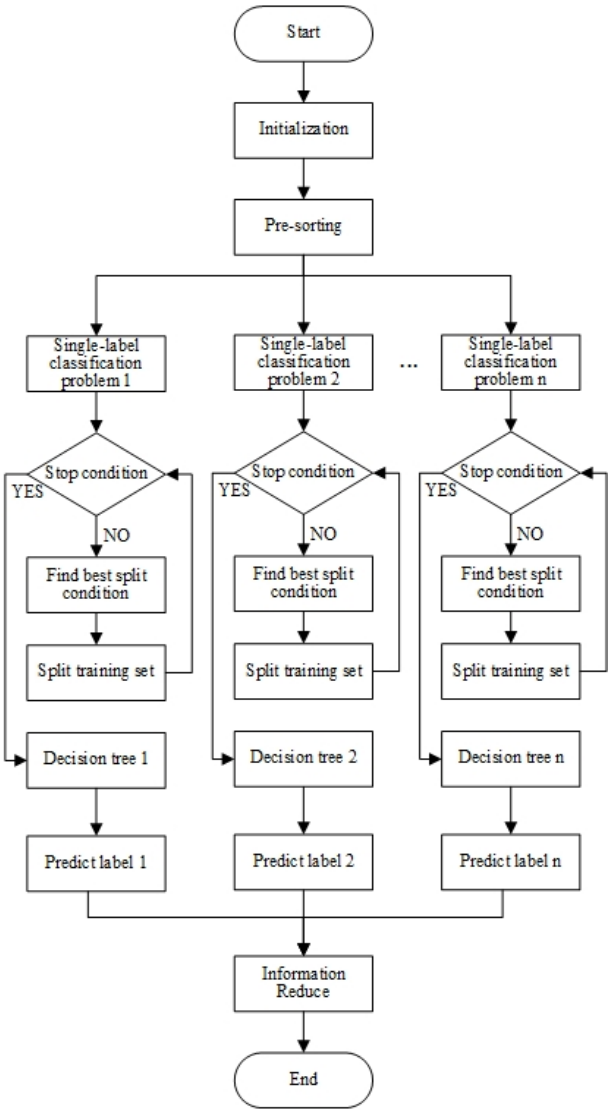

FIGURE V. FLOW CHART

IV. PERFORMANCE ANALYSIS

Intel® Core™ i5-4460, a quad-core CPU with 3.2GHz of basic frequency, is used on the test PC, which is equipped with 8G of memory. And the default compilation environment is Visual Studio 2015 and MPICH 3.2 based on Windows 7 system. Our training data set is $birds$ from Mulan, which has 19 labels and 260 attributes (2 of them are discrete attributes and the others are continuous attributes). For each value of experimental data, the average value of 5 tests was taken as the final results.

In this part, we only focus on the speeding performance rather than classification performance, because the decision trees built by parallel algorithm is exactly the same with serial algorithm. Now, we introduce speed-up ratio $S_n = T_s/T_p$ to evaluate the speed-up ratio between parallel and serial algorithm, T_s is run time of serial algorithm and T_p is rum time of parallel algorithm.

We run the serial and parallel algorithm on the same experiment environment, the processing nodes were simulated by the single quad-core CPU. The parallel algorithm was run under different numbers of processing nodes, and the speed-up ratios are shown in TABLE I. We can see that with the increasing of processing nodes, the speed-up ratio is increasing. The above increasing proves the parallel algorithm we proposed is efficiency. While the number of processing nodes exceeding 4, the speed-up ratio can't be proportional to the node number. This is due to the communication overhead of the parallel algorithm increasing quickly and the limitation of physical quad-core CPU.

TABLE I. PERFORMANCE COMPARISON OF DIFFERENT N T HREADS

nThreads	CPU Time(s)	s_n
1	16.13798	1
2	9.312078	1.733
4	4.978032	3.242
8	5.320034	3.033
16	5.238556	3.081

V. CONCLUSION

Multi-label classification algorithm is widely applied in many areas of data mining and faced extreme performance challenges. This paper focus on the parallel improvement of serial decision tree based multi-label classification algorithm. The "problem transformation" method is used to change the multi-label classification problem to several corresponding single-label classification problem. We proposed a parallel decision tree based multi-label classification algorithm and implemented it based on MPI. We analyzed the speed-up ration between serial algorithm and parallel algorithm and the results proved that our algorithm can improve the efficiency of the decision tree based multi-label classification algorithm.

ACKNOWLEDGEMENT

This paper is supported by The National Natural Science Foundation Project of China (61472100).

REFERENCES

[1] Zhang M L, Zhou Z H. A Review on Multi-Label Learning Algorithms[J]. Knowledge & Data Engineering IEEE Transactions on, 2014, 26(8):1819-1837.

[2] Park H, Kwon H C. Improved Gini-Index Algorithm to Correct Feature-Selection Bias in Text Classification[J]. Ieice Transactions on Information & Systems, 2011, 94-D(4):855-865.

[3] Boutell M R, Luo J, Shen X, et al. Learning multi-label scene classification ☆[J]. Pattern Recognition, 2004, 37(9):1757-1771.

[4] Shafer J C, Agrawal R, Mehta M. SPRINT: A Scalable Parallel Classifier for Data Mining[C]// Proceedings of the 22th International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc. 2000:544-555.

[5] Joshi M V, Karypis G, Kumar V. ScalParC: a new scalable and efficient parallel classification algorithm for mining large datasets[C]// Parallel Processing Symposium, 1998. Ipps/spdp 1998. Proceedings of the First Merged International. and Symposium on Parallel and Distributed Processing. IEEE, 1998:573-579.

[6] Clare A, King R D. Knowledge Discovery in Multi-label Phenotype Data[J]. Lecture Notes in Computer Science, 2002, 2168(2168):42-53.