Knowledge-Based Systems 35 (2012) 49–63

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Computationally efficient induction of classification rules with the PMCRI and J-PMCRI frameworks

Frederic Stahl [a,*], Max Bramer [b]

[a] Bournemouth University, School of Design, Engineering & Computing, Poole House, Talbot Campus, BH12 5BB Poole, United Kingdom
[b] University of Portsmouth, School of Computing, Buckingham Building, Lion Terrace, PO1 3HE Portsmouth, United Kingdom

ARTICLE INFO

Article history:
Received 28 January 2011
Received in revised form 10 April 2012
Accepted 11 April 2012
Available online 21 April 2012

Keywords:
Parallel computing
Parallel rule induction
Modular classification rule induction
PMCRI
J-PMCRI
Prism

ABSTRACT

In order to gain knowledge from large databases, scalable data mining technologies are needed. Data are captured on a large scale and thus databases are increasing at a fast pace. This leads to the utilisation of parallel computing technologies in order to cope with large amounts of data. In the area of classification rule induction, parallelisation of classification rules has focused on the *divide and conquer* approach, also known as the Top Down Induction of Decision Trees (TDIDT). An alternative approach to classification rule induction is *separate and conquer* which has only recently been in the focus of parallelisation. This work introduces and evaluates empirically a framework for the parallel induction of classification rules, generated by members of the Prism family of algorithms. All members of the Prism family of algorithms follow the *separate and conquer* approach.

1. Introduction

Many application areas are confronted with the problem of applying classification rule induction algorithms or data mining technologies in general on very large datasets. Such application areas include bioinformatics and chemistry which are confronted with large data sets, for example data generated in molecular dynamics simulation experiments. Researchers in this area need ways to manage, store and find complex relationships in the simulation data [1]. The molecular dynamics datasets can reach 100s of gigabytes of data for a single simulation and the community is just starting to be able to store these massive amounts of simulation data [2]. A further area confronted with massive amounts of data is astronomy. Here some databases consist of terabytes of image data and are still growing as further data are collected in relation to the GSC-II [3] and the still ongoing Sloan survey [4]. Large international business corporations collect and share customer transactions in databases worldwide. Loosely speaking there is a significant need for well scaling knowledge discovery and data mining technologies for massive datasets for both the scientific and business world. Parallelisation seems to be one of the methods used in the data mining community to tackle the problem of scalability in computational terms [33,10,21].

One of the major challenges in data mining is the induction of classification rules on massive datasets. There are two general approaches to inducing classification rules, the *divide and conquer* and the *separate and conquer* approaches. The induction of classification rules can be traced back to the 1960s [5]. The *divide and conquer* approach induces classification rules in the form of a decision tree by recursively splitting the classification problem [6]. Its most popular representatives are the C4.5 [7] and C5.0 systems. Contrary to decision trees the *separate and conquer* approach induces classification rules directly that explain a part of the training data. *Separate and conquer* can be traced back to the 1960s [8]. Parallel classification rule induction has focused on the *divide and conquer* approach. A notable development here is SPRINT [9]. Joshi et al. [10] points out that in some cases SPRINT may suffer from workload balancing issues and the ScalParC algorithm is proposed. However there are virtually no approaches to scaling up the *separate and conquer* approach.

The Prism [11] family of algorithms follows the *separate and conquer* approach and addresses some of the shortcomings of decision trees, such as the replicated subtree problem outlined in Section 2.1. More recent variations of Prism have demonstrated a similar classification accuracy compared with decision trees and in some cases even outperform decision trees [17,15]. An implementation of Prism is also available in the WEKA data mining package [35]. This work proposes and evaluates the Parallel Modular

* Corresponding author.
E-mail addresses: fstahl@bournemouth.ac.uk (F. Stahl), Max.Bramer@port.ac.uk (M. Bramer).

0950-7051/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2012.04.014

Classification Rule Induction framework (PMCRI) which parallelises the Prism family of algorithms, in order to computationally scale up Prism algorithms to large datasets. PMCRI could potentially scale up further algorithms, that follow the *separate and conquer* approach, to large datasets, however, it may not be applicable to all of them.

In the PMCRI framework the parallelisation is aimed at a network of computer workstations with the reasoning that modest sized organisations may not have the financial strength to afford a supercomputer but will most likely have a network of workstations in place which could be used to run parallel algorithms. PMCRI partitions the training data according to the features space and assigns equally sized subsets of the feature space to each computing node. Each computing node processes its part of the feature space and then cooperates with the other computing nodes in order to combine the computed results to classification rules. The computational performance of PMCRI is evaluated in terms of its execution time dependent on the number or data instances and the number of attributes/features. Furthermore PMCRI is evaluated to show how much computational benefit is gained by using p processors instead of one, dependent on the size of the datasets.

This paper is organised as follows: Section 2 introduces the Prism family of algorithms and compares them with decision trees; Section 3 discusses the PMCRI framework and Section 4 evaluates it. Section 5 introduces a version of PMCRI that incorporates a prepruning facility (J-PMCRI) and evaluates it in computational terms. Finally Section 6 closes the paper with a brief summary, concluding remarks and an outlook to future work.

2. The Prism family of algorithms

The Prism family of algorithms is a representative of the 'separate and conquer' approach outlined in Section 1 as opposed to the 'divide and conquer' approach.

2.1. The replicated subtree problem

The 'divide and conquer' approach induces classification rules in the intermediate form of a tree whereas the 'separate and conquer' approach, and thus the Prism family of algorithms, induces modular rules that do not necessarily fit into a decision tree. Modular rules such as

IF A = 1 AND B = 1 THEN class = x
IF C = 1 AND D = 1 THEN class = x

will not fit into a decision tree as they have no attribute in common. In order to represent them in a decision tree, additional logically redundant rule terms would have to be added. This can result in complex and confusing trees as Cendrowska shows in [11] and is also known as the replicated subtree problem [12]. Cendrowska's example illustrates the replicated subtree problem for the two rules listed above. She assumes that all attributes can have three possible values and only the two rules above classify for class x. Fig. 1 shows the simplest possible decision tree expressing the two rules above, all remaining classes that are not class x are labelled y.

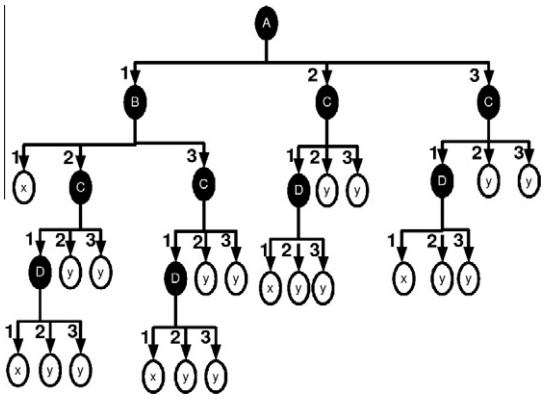

Fig. 1. Cendrowska's replicated subtree example.

Cendrowska's claim that decision tree induction algorithms grow needlessly complex is vindicated by extracting the rules for class x from the tree in Fig. 1, which are:

IF A = 1 AND B = 1 THEN class = x
IF A = 1 AND B = 2 AND C = 1 AND D = 1 THEN class = x
IF A = 1 AND B = 3 AND C = 1 AND D = 1 THEN class = x
IF A = 2 AND C = 1 AND D = 1 THEN class = x
IF A = 3 AND C = 1 AND D = 1 THEN class = x

2.2. The 'separate and conquer' approach

Algorithms induced by the 'separate and conquer' approach aim to avoid the replicated subtree problem by avoiding the induction of redundant rule terms just for the representation in a decision tree. The basic 'separate and conquer' approach can be described as follows:

```
While Stopping Criterion not satisfied{
  rule = Learn_Rule;
  Remove all data instances covered from Rule;
  add rule to the rule set;
}
```

The *Learn_Rule* procedure (or specialisation process) induces the 'best' rule for the current subset of the training set by searching for the best conjunction of attribute-value pairs (rule terms). The perception of 'best' depends on the heuristic used to measure the goodness of the rule, for example its coverage or predictive accuracy. The *Learn_Rule* procedure can be computationally very expensive, especially if all possible conjunctions of all possible rule terms have to be considered. After a rule is induced, all examples that are covered by that rule are deleted and the next rule is induced using the remaining examples until a *Stopping Citerion* is fulfilled. Different 'separate and conquer' algorithms implement different methods to reduce the search space of the *Learn_Rule* procedure and the *Stopping Criterion*. Some examples of algorithms that follow the 'separate and conquer' approach are [11,36,8,37].

2.3. The Prism approach

In the Prism approach, first a Target Class (TC), for which a rule is induced, is selected. Prism then uses an information theoretic approach [11] based on the probability with which a rule covers the TC in the current subset of the training data to specialise the rule. Once a rule term is added to the rule, a further rule term is induced only on the subset of the training data, that is covered by all the rule terms induced so far. This is done until the rule currently being induced only covers instances that match the TC. In Cendrowska's original Prism algorithm the TC is selected at the beginning by the user and only rules for the TC are induced. Alternatively the algorithm can be given a list of possible classes and is executed for each class in turn. Bramer's PrismTCS (Target Class Smallest first) algorithm [13] sets the TC for each new rule to be induced to the current minority class. Another member of the Prism family is PrismTC which sets the TC for each new rule to be induced to the current majority class. Unpublished experiments by Bramer revealed that PrismTC does not compete well