
Bibliography

- [Adleman, 1994] Leonard M. Adleman. Molecular Computation of Solutions to Combinatorial Problems. Science (journal) 266 (11): 1021-1024. 1994.
- [Adleman, 1998] Leonard M. Adleman. Computing with DNA. Scientific American 279: 54-61. 1998
- [Lipton, 1995] Richard J.Lipton. Using DNA to solve NP-Complete Problems. Science, 268:542-545. April 1995
- [Holland, 1975] J.H.Holland. Adaptation in Natural and Artificial Systems. MIT Press. 1975.
- [J.Castellanos, 1998] J.Castellanos, S.Leiva, J.Rodrigo, A.Rodríguez Patón. Molecular computation for genetic algorithms. First International Conference, RSCTC'98.
- [M.Calviño, 2006] María Calviño, Nuria Gómez, Luis F.Mingo. DNA simulation of genetic algorithms: fitness computation.
- [Macek, 1997] Milan Macek M.D. Denaturing gradient gel electrophoresis (DGDE) protocol. Hum Mutation 9: 136 1997.
- [Dove, 1998] Alan Dove. From bits to bases; Computing with DNA. Nature Biotechnology. 16(9):830-832; September 1998.
- [Mitchell, 1990] Melanie Mitchell. An Introduction to Genetic Algorithms. MIT Press, Boston. 1998.
- [Lee, 2005] S.Lee, E. Kim. DNA Computing for efficient encoding of weights in the travelling salesman problem. ICNN&B'05. 2005.
- [SY Shin, 2005] SY Shin, IH Lee, D Kim, BT Zhang. Multiobjective evolutionary optimization of DNA sequences for reliable DNA computing. IEEE Transactions, 2005.
- [Gerald Karp, 2005] Gerald Karp.Cell and molecular biology: Concepts and experiments, 2005, Von Hoffman press
- [Ayala, 1984] F.J. Ayala, J.A. Kiger. Modern genetics (2nd edition).
- [Crow, 1986] J.F. Crow. Basic concepts in population, quantitative, and evolutionary Genetics. W.H. Freeman and Co. New York.
- [Hartl, 1989] D.L. Hartl, A.G. Clark. Genetics of populations. Science Books International. Boston.
- [Ford, 1991] T.C. Ford, J.M. Graham. An introduction to centrifugation. Bios Scientific Publishers. Oxford.
- [Wilson, 1986] K. Wilson, K.H. Goulding. Principles and Techniques of Practical Biochemistry. Arnold LTD. Suffolk.

Authors' Information

Ángel Goñi Moreno – Natural Computing Group. Universidad Politécnica de Madrid, Boadilla del Monte, 28660 Madrid, Spain: e-mail: ago@alumnos.upm.es

A FRAMEWORK FOR FAST CLASSIFICATION ALGORITHMS

Thakur Ghanshyam, Ramesh Chandra Jain

Abstract: Today, due to globalization of the world the size of data set is increasing, it is necessary to discover the knowledge. The discovery of knowledge can be typically in the form of association rules, classification rules, clustering, discovery of frequent episodes and deviation detection. Fast and accurate classifiers for large databases are an important task in data mining. There is growing evidence that integrating classification and association rules mining, classification approaches based on heuristic, greedy search like decision tree induction. Emerging associative classification algorithms have shown good promises on producing accurate classifiers. In this paper we focus on performance of associative classification and present a parallel model for classifier building. For classifier building some parallel-distributed algorithms have been proposed for decision tree induction but so far no such work has been reported for associative classification.

Keywords: classification, association, and data mining.

1. Introduction

Data mining algorithms task is discovering knowledge from massive data sets. Building classifiers is one of the core tasks of data mining. Classification generally involves two phases, training and test. In the training phase the rule set is generated from the training data where each rule associates a pattern to a class. In the test phase the

generated rule set is used to decide the class that a test data record belongs to. Traditionally, greedy search techniques such as decision trees [8] and others are used to develop classifiers. Decision Tree Induction approaches have been preferred to other traditional techniques due to the generation of small rule set and transparent classifiers. Transparent classifier means that rules are straightforward and simple to understand, unlike some opaque classifiers, such as one generated by neural networks, where interpretation of rules is difficult. Greedy techniques in decision tree construction approaches tend to minimize overlapping between training data records to generate small rule sets. However small rule sets have some disadvantages. Greedy techniques may achieve global optimality if the problem has a optimal substructure. A novel technique of associative classification based on association rule mining searches globally for all rules that satisfy minimum support and count thresholds [5].

2. Work already done in the field.

Several methods for improving the efficiency of all approach have been proposed [4,5,7,9,10] based on a recursive method for constructing a decision tree. In associative classification the classifier model is composed of a particular set of association rules, in which consequent of each rule is restricted to classification class attribute. The experiments in [4,5,7] show that this approach achieves higher accuracy than traditional approaches. Many sequential algorithms have been proposed for associative classification [4,5,7,9,10]. However associative classification suffers from efficiency due to the fact that it often generates a very large number of rules in association rule mining and it also takes efforts to select high quality rules from among them [7].

Since data mining is mostly applied on databases, which are very large, to improve the performance parallel algorithms are needed. Many parallel approaches have been given for association rule mining [11] and decision tree classifiers [3], no parallel algorithm has been proposed for associative classification. In this paper a model is proposed for parallel approach of associative classification for significant performance improvement. We propose a parallel model for CBA [5] algorithm.

3. Proposed methodology

Parallel Approaches for Data Mining:

Since data mining is frequently applied over large datasets, performance of algorithms is of concern. Exploiting the inherent parallelism of data mining algorithms provide a direct solution to their performance lift. A classification of different approaches to parallel processing for data mining is presented in [3].

Parallel Approaches:

1.Task Parallelism

1. Divide & Conquer
2. Task Queue

Task-parallel algorithms assign portions of the search space to separate processors. The task parallel approaches can again be divided in two groups. The first group is based on a Divide and Conquer strategy that divides the search space and assigns each partition to a specific processor.

2.Data Parallelism

1. Record Based
2. Attribute Based

The second group is based on a task queue that dynamically assigns small portions of the search space to a processor whenever it becomes available. Data-parallel, approaches distribute the data set over the available processors. Data-parallel approaches are in two directions. A partitioning based on records will assign non-overlapping sets of records to each of the processors. Alternatively a partitioning of attributes will assign sets of attributes to each of the processors. Attribute-based approaches are based on the observation that many algorithms can be expressed in terms of primitives that consider every attribute in turn. If attributes are distributed over multiple processors, these primitives may be executed in parallel. Many other issues on parallel processing of data mining with respect to Association Rule mining have been presented in [11].

The main challenges include synchronization and communication minimization, workload balancing, finding good data layout, data decomposition, and disk I/O minimization.

The parallel design space spans three main components:

1. The hardware platform,
2. The type of parallelism,
3. The load-balancing strategy.

Two dominant approaches for using multiple processors have emerged:

Distributed memory (where each processor has a private memory)

In distributed-memory (DMM) architecture, each processor has its own local memory and independent hard disk, which only that processor can access directly. For a processor to access data in the local memory of another processor, message passing must send a copy of the desired data elements from one processor to the other. A distributed memory, message-passing architecture cures the scalability problem by eliminating the bus, but at the expense of programming simplicity

Shared memory (where all processors access common memory).

Shared-memory (SMP) architecture has many desirable properties. Each processor has direct and equal access to all the system's memory. Parallel programs are easy to implement on such a system. A different approach to multiprocessing is to build a system from many units, each containing a processor and memory. Although shared memory architecture offers programming simplicity, a common bus's finite bandwidth can limit scalability. Load balancing strategies entail static or dynamic approaches. Static load balancing initially partitions work among the processors using a heuristic cost function, no subsequent data or computation movement is available. Dynamic load balancing seeks to address this by taking work from heavily loaded processors and reassigning it to lightly loaded ones. Computation movement also entails data movement, because the processor responsible for a computational task needs the data associated with that task. Dynamic load balancing thus incurs additional costs for work and data movement, and also for the mechanism used to detect whether there is an imbalance. However, dynamic load balancing is essential if there is a large load imbalance or if the load changes with time. Dynamic load balancing is especially important in multi-user environments with transient loads and in heterogeneous platforms, which have different processor and network speeds. These kinds of environments include parallel servers and heterogeneous clusters, meta-clusters, and super-clusters (the so called grid platforms that are becoming common today). There are various approaches that can be applied to parallel processing of data mining algorithms. Cost measures for various parallel data mining strategies to predict their computation, data access and communication performance are presented in [6].

1. Associative Classification

Associative Classification is an integrated framework of Association Rule Mining (ARM) and Classification. Focusing on a special subset of association rules whose right-hand-side is restricted to the classification class attribute does the integration. This subset of rules is referred as the Class Association Rules (CARs). CBA (Classification Based on Associations) [5] is a sequential approach of building associative classifier. CBA consists of two parts, a rule generator (called CBA-RG), which is based on algorithm Apriori for finding association rules in [2], and a classifier builder (called CBA-CB). CBA approach is described below. Assuming given dataset is a normal relational table, which consists of N cases described by l distinct attributes. These N cases have been classified into q known classes. For Associative Classification it is assumed that in training data set all continuous attributes (if any) have been discretized as a preprocessing step. For all attributes, all the possible values are mapped to a set of consecutive positive integers. With these mappings, a data case can be treated as a set of (*attribute, integer-value*) pairs and a class label. Each (*attribute, integer-value*) pair is called an *item*. Let D be the dataset. Let I be the set of all items in D , and Y be the set of class labels. We say that a data case $d \in D$ contains $X \subseteq I$, a subset of items, if $X \subseteq d$. A classification rule (CAR) is an implication of the form $X \rightarrow y$, where $X \subseteq I$, and $y \in Y$. A rule $X \rightarrow y$ holds in D with confidence c if $c\%$ of cases in D that contain X are labeled with class y . The rule $X \rightarrow y$ has support s in D if $s\%$ of the cases in D contain X and are labeled with class y .

Rule Generator CBA-RG

The key operation of CBA-RG is to find all ruleitems that have support above minsup. A ruleitem is of the form: $\langle \text{condset}, y \rangle$, where condset is a set of items, $y \in Y$ is a class label. The support count of the condset (called

condsupCount) is the number of cases in D that contain the condset. The support count of the ruleitem (called rulesupCount) is the number of cases in D that contain the condset and are labeled with class y . Each ruleitem basically represents a rule: $\text{condset} \rightarrow y$, whose support is $(\text{rulesupCount} / |D|) * 100\%$, where $|D|$

is the size of the dataset, and whose confidence is $(\text{rulesupCount} / \text{condsupCount}) * 100\%$. Ruleitems that satisfy minsup are called *frequent ruleitems*, while the rest are called *infrequent ruleitems*. For example, the following is a ruleitem: $\langle \{(A, 1), (B, 1)\}, (\text{class}, 1) \rangle$, where A and B are attributes. If the support count of the condset $\{(A, 1), (B, 1)\}$ is 3, the support count of the ruleitem is 2, and the total number of cases in D is 10, then the support of the ruleitem is 20%, and the confidence is 66.7%. If minsup is 10%, then the ruleitem satisfies the minsup criterion. We say it is frequent. For all the ruleitems that have the same condset, the ruleitem with the highest confidence is chosen as the possible rule (PR) representing this set of ruleitems. If there are more than one ruleitem with the same highest confidence, we randomly select one ruleitem. For example, we have two ruleitems that have the same condset:

1. $\langle \{(A, 1), (B, 1)\}, (\text{class}: 1) \rangle$.
2. $\langle \{(A, 1), (B, 1)\}, (\text{class}: 2) \rangle$.

Assume the support count of the condset is 3. The support count of the first ruleitem is 2, and the second ruleitem is 1. Then, the confidence of ruleitem 1 is 66.7%, while the confidence of ruleitem 2 is 33.3%. With these two ruleitems, we only produce one PR (assume $|D| = 10$): $(A, 1), (B, 1)_{(\text{class}, 1)}$ [support = 20%, confidence = 66.7%]. If the confidence is greater than *minconf*, we say the rule is *accurate*. The set of class association rules (CARs) thus consists of all the PRs that are both frequent and accurate.

The CBA-RG algorithm generates all the frequent ruleitems by making multiple passes over the data. In the first pass, it counts the support of individual ruleitem and determines whether it is frequent. In each subsequent pass, it starts with the seed set of ruleitems found to be frequent in the previous pass. It uses this seed set to generate new possibly frequent ruleitems, called *candidate ruleitems*. The actual supports for these candidate ruleitems are calculated during the pass over the data. At the end of the pass, it determines which of the candidate ruleitems are actually frequent. From this set of frequent ruleitems, it produces the rules (CARs). Let k -ruleitem denote a ruleitem whose condset has k items. Each element F_k of this set is of the following form: $\langle (\text{condset}, \text{condsupCount}), (y, \text{rulesupCount}) \rangle$.

The CBA-RG algorithm is given in Figure 1.

II. Building a Classifier CBA-CB

To produce the best classifier out of the whole set of rules, a heuristic approach is used. A total order on the generated rules is defined. For more details on CBA approach readers are referred to [5]. This is used in selecting the rules for classifier.

Definition: Given two rules, r_i and r_j , $r_i \succ r_j$ (also called r_i precedes r_j or r_i has a higher precedence

```

F1 = {large 1-ruleitems};
CAR1 = genRules(F1);
for (k = 2; Fk-1 ≠ ∅; k++) do
  Ck = candidateGen(Fk-1);
  for each data case d ∈ D do
    Cd = ruleSubset(Ck, d);
    for each candidate c ∈ Cd do
      c.condsupCount++;
      if d.class = c.class then c.rulesupCount++;
    end
  end
  Fk = {c ∈ Ck | c.rulesupCount ≥ minsup};
  CARk = genRules(Fk);
end
CARs = ∪k CARk;

```

Figure 1

```

R = sort(R); // sort on precedence >
for each rule r ∈ R in sequence do
  temp = ∅;
  for each case d ∈ D do
    if d satisfies the conditions of r then
      store d.id in temp and mark r if it correctly
      classifies d;
  end
  if r is marked then
    insert r at the end of C;
    delete all the cases with the ids in temp from D;
    selecting a default class for the current C;
    compute the total number of errors of C;
  end
end
Find the first rule p in C with the lowest total number
of errors and drop all the rules after p in C;
Add the default class associated with p to end of C;
return C (our classifier).

```

Figure 2. The CBA-RG : M1 algorithm

than r_j) if 1). the confidence of r_i is greater than that of r_j , or 2). their confidences are the same, but the support of r_i is greater than that of r_j , or 3. both the confidences and supports of r_i and r_j are the same, but r_i is generated earlier than r_j ;

Let R be the set of generated CARs and D the training data. The basic idea of the algorithm is to choose a set of high precedence rules in R to cover D .

Our classifier is of the following format: $\langle r_1, r_2, r_3, \dots, r_n, \text{default_class} \rangle$, where $r_i \in R$, $r_a \succ r_b$ if $b > a$. default_class is the default class. In classifying an unseen case, the first rule that satisfies the case will classify it. If there is no rule that applies to the case, it takes on the default class as in C4.5. A pseudo code of algorithm M1 for building such a classifier is shown in Figure 2.

III. Parallel and Distributed Associative Classification

CBA is an associative classification algorithm that uses an Apriori based approach to mine CARs and produces a subset of these CARs after pruning to form a classifier. We adapt here popular CBA algorithm discussed in section 3 to present our approach of parallel associative classification. For both phases of associative classification, rule generation phase and classifier builder phase our approach is based on Distributed Memory Systems, Record Based data parallelism and uses Static Load Balancing. The above configuration of parallel approach suits to the inherent parallel nature of existing serial approach. Three parallel versions of Apriori are given in [1] on shared nothing architecture. We adapt count distribution algorithm of ARM mining for mining of CARs in associative classification and present parallel version of CBA-M1 for classifier building. Count distribution approach has minimized communication among the processors. For CARs mining the training data set is partitioned among P processors. Each processor works on its local partition of the database and performs same

set of instructions to mine CARs that have global min support and confidence. Later when all CARs are found, same partitions of training set are used in respective nodes and pruning process based on coverage analysis is applied in parallel to generate reduced set of CARs, to form classifier. Our approach simply achieves load balancing if training data sets are sufficiently randomly distributed over different processors to avoid any data skew.

It can simply be inferred

$$|D| = N \text{ and number of processors} = p$$

$$|D_i| = |D|/p \text{ (approximately), } i=1,2,\dots,p$$

C_k	Set of candidate k -ruleitems
F_k	Set of frequent k -ruleitems
D	Training Dataset
D_i	Local Training Dataset on i^{th} Processor
P_i	i^{th} Processor
R	Set of generated CARs

Figure 3. Notations

```

do in parallel
  k = 2;
  while ( $F_{k-1} \neq \emptyset$ ) do
    for each processor  $P_i$  ( $i=1..p$ ) do
       $C_k = \text{candidateGen}(F_{k-1})$ ;
      for each data case  $d \in D$  do
         $C_d = \text{ruleSubset}(C_k, d)$ ;
        // compute local support for ruleitems
        for each candidate  $c \in C_d$  do
           $c.\text{condsupCount}++$ ;
          if  $d.\text{class} = c.\text{class}$  then  $c.\text{rulesupCount}++$ 
        end
      end
      exchange local  $C_k$  counts with all other processors
      and synchronize;
      for each  $c \in C_k$   $c.\text{condsupCount} = \sum_{P_i(i=1..p)} c.\text{condsupCount}$ ;
       $c.\text{rulesupCount} = \sum_{P_i(i=1..p)} c.\text{rulesupCount}$ ;
      end
       $F_k = \{c \in C_k \mid c.\text{rulesupCount} \geq \text{minsup}\}$ ;
       $\text{CAR}_k = \text{genRules}(F_k)$ ;
       $k++$ ;
    end while
   $\text{CARs} = \cup \text{CAR}_k$ 
End parallel
    
```

Figure 4

The necessary communication among processors is through message broadcasting. There has been no need of dynamic load balancing as the Associative Classifier builder task does not involve multi-user environment with changing training data sets during learning. As in ARM in associative classification too the static load balancing is inherent in the partitioning of the database among processors because training data sets have been made available in a homogeneous environment. Parallel versions adapted from CBA for CAR generation and classifier builder are presented below.

Pseudo codes given in Figure 4 use notations given in Figure 3.

Parallel CAR generation phase

At the time of generating partitions D_i for processors P_i , F_1 and hence CAR_1 can be generated and distributed to distributed processors along with data partitions. Hence algorithm begins with a seed of F_1 . During partitioning *class_distribution* i.e. number of data cases for each of the classes will also be computed and distributed to processors to be used during class builder phase. Pseudo code of parallel rule generation algorithm is presented in figure 4.

In Parallel CAR generation algorithm each processor independently and in parallel generates identical C_k for $k > 1$ and calculates local counts and broadcast these to all other processors. At this step processors synchronize and wait for all other processors to compute and broadcast their local counts. Summing local counts of all processors global counts for C_k are computed. Each processor now computes F_k and generates CARs from F_k . Process is iterated for next k till until F_k is empty. At the end of rule generation algorithm each processor has complete set of CARs.

4. Cost Measures For Proposed Model

All this information exchanged is integer valued and its volume is very small.

Cost estimate of sequential CBA approach based on cost models in [6] is as follows. Global structure of both rule generation and class builder algorithms is a loop building more accurate concepts from those of the previous iterations. Suppose loop in rule generation algorithm and classifier builder algorithms executes k_{s1}, k_{s2} times and builds Ω_1, Ω_2 concepts respectively. Total size D is N and number of attributes in D is l . The cost estimate of the sequential CBA algorithm can be given by formula

$$Cost_{seq} = k_{s1} [STEP(N^*, \Omega_1) + ACCESS(N^*)] + k_{s2} [STEP(N^*, \Omega_2) + ACCESS(N^*)]$$

Where *STEP* gives the cost of single iteration of the loop, and *ACCESS* is the cost of accessing the data set once.

If the CBA is performed in parallel version of rule generation algorithm and classifier builder algorithms and requires k_{a1}, k_{a2} iterations respectively with number of p processors, formula for the cost can be given by

$$Cost_{par} = k_{a1} [STEP(N^*/p, \Omega_1) + ACCESS(N^*/p) + C_{e1}] + k_{a2} [STEP(N^*/p, \Omega_2) + ACCESS(N^*/p) + C_{e2}]$$

C_{e1}, C_{e2} are total cost of communication and information exchange between the processors.

It can be reasonably assumed that

$$STEP(N^*/p, \Omega_1) = STEP(N^*, \Omega_1)/p$$

and

$$ACCESS(N^*/p) = ACCESS(N^*)/p$$

So, we get significant p -fold speedup in executing parallel version except cost of overheads. In our model overhead cost is small as information exchanged is integer valued and its volume is very small.

5. Conclusion

In this paper the focus was on the performance of classifier builder approach known as associative classification. We proposed a model to show that associative classification task can be performed in parallel on distributed memory systems to achieve a significant performance lift. We have presented parallel versions for both ruled generation and class builder phase of sequential CBA algorithm for load balancing we have distributed almost equal number of data sets randomly on each of the local processors to avoid data skewness.

Bibliography

1. R. Agrawal and J. C. Shafer, "Parallel Mining of Association Rules", *IEEE Transactions On Knowledge And Data Engineering*, pages 962–969, 1996.
2. R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", In *Proc. of the Int. Conf. on Very Large Databases*, SanDiago, Chile, pages 487–499, 1994.
3. J. Chattratichat, "Large Scale Data Mining: Challenges and Responses," *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, 1997, pp 143-146.
4. W. Li, J. Han and J. Pei, "CMAR: Efficient classification based on multiple class-association rules. In *Proc. of the Int. Conf. on Data Mining*, pages 369–376, 2001.
5. B. Liu, W. Hsu and Y. Ma, "Integrating classification and association rule mining", In *Knowledge Discovery and Data Mining*, pages 80–86, 1998.
6. D. B. Skillicorn, "Strategies for parallel data mining", *IEEE Concurrency*, vol. 7, No. 4, 1999.
7. X. Yin and J. Han, "CPAR: Classification based on Predictive Association Rules", In *Proc. of the Int. Conf. on Data Mining*, SDM. SIAM, 2003.
8. F. Thabtah, P. Cowling and Y. Peng, "MCAR: Multi-class Classification based on Association Rule Approach. In *Proceeding of third IEEE International Conference on Computer Systems and Applications*, Cairo, Egypt, pages 1-7, 2005.
9. F. Thabtah, P. Cowling and Y. Peng, "MMAC: A New Multi-class, Multi-label Associative Classification Approach", In *Proceeding of fourth IEEE International Conference on Data Mining (ICDM '04)*, Brighton, UK, pages 217-224, Nov. 2004.

Authors' Information

Thakur S. Ghanshyam, Dr. R.C.Jain – Department of Computer Application; Samrat Ashok Technological Institute; Vidisha(M.P.), INDIA; e-mail: ghanshyamthakur@gmail.com

ENHANCING INFORMATION RETRIEVAL BY USING EVOLUTION STRATEGIES

Abdelmgeid Amin Aly

Abstract: Similar to Genetic algorithm, Evolution strategy is a process of continuous reproduction, trial and selection. Each new generation is an improvement on the one that went before. This paper presents two different proposals based on the vector space model (VSM) as a traditional model in information Retrieval (TIR). The first uses evolution strategy (ES). The second uses the document centroid (DC) in query expansion technique. Then the results are compared; it was noticed that ES technique is more efficient than the other methods.

1. Introduction

Since the 1940s the problem of Information Retrieval (IR) has attracted increasing attention, especially because of the dramatically growing availability of documents. IR is the process of determining relevant documents from a collection of documents, based on a query presented by the user.

There are many IR systems based on Boolean, vector, and probabilistic models. All of them use their model to describe documents, queries, and algorithms to compute relevance between user's query and documents.

Information Retrieval (IR) proposes solutions for searching, in a given set of objects, for those replying to a given description. IR tries to make a suitable use of these databases, allowing the users to access to the information which is really relevant in an appropriate time interval [1]. Unfortunately, commercial IR Systems (IRs), usually based on the Boolean IR model [2], have provided unsatisfactory results. Vector space, probabilistic and fuzzy models, which have been developed to extend the Boolean model [3], as well as the application of knowledge-