Parallel Approaches to Neighborhood Rough Sets: Classification and Feature Selection

Junbo Zhang, Chizheng Wang, Yi Pan and Tianrui Li

Abstract In these days, the ever-increasing volume of data requires that data mining algorithms should not only have high accuracy but also have high performance, which is really a challenge for the existing data analysis methods. Traditional algorithms, such as classification and feature selection under neighborhood rough sets, have been proved to be very effective in real applications. Parallel approach to these traditional algorithms could be a way to take the challenge. This is what we present in this paper, the design of parallel approaches to neighborhood rough sets and the implementation of classification and feature selection. Two optimizing strategies are proposed to improve the performance of the approaches: (1) The distributed cache is used to reduce I/O time. (2) Most of computations are put into the Map phase which helps reduce the overhead of communication. The experimental results show that the proposed algorithms scale pretty well and the speedup is getting higher with the increasing size of data.

J. Zhang · T. Li (✉)
School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China
e-mail: trli@swjtu.edu.cn

J. Zhang
e-mail: JunboZhang86@163.com; jbzhang@cs.gsu.edu

J. Zhang · Y. Pan
Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA
e-mail: pan@cs.gsu.edu

C. Wang
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
e-mail: cwang86@illinois.edu

Z. Wen and T. Li (eds.), *Knowledge Engineering and Management*, Advances in Intelligent Systems and Computing 278,
DOI: 10.1007/978-3-642-54930-4_1,

1 Introduction

People always say we are living in the information age [3]. What they say is actually we are living in *big data* age. Growing amount and kinds of data play important roles in uncountable fields, including business, astronomy, and bioinformatics. Data mining focuses on discovering knowledge from different kinds of data. Classification is one of the hot topics in data mining. There are basically two kinds of classification algorithms, namely eager learner and lazy learner [3]. Eager learner algorithm usually contains two steps, learning step and classification step. In the learning step, the classifier will create a model based on a set of data, which is usually called training data set. This model will be used to classify the test data. ID3, C4.5, and SVM are all eager learner algorithms.

The lazy algorithms also need a set of training data. However, unlike eager learner algorithm, lazy algorithms do not use the training set to create a model, instead they use the test data to compare with training data set to make decisions. The neighborhood classifiers (NEC), which is introduced in [5], and *k*-nearest-neighbor classifiers are both lazy learner algorithms. As the development of the research in data mining, many mathematical and statistical tools are involved to help to increase the performance of data mining. Rough set theory, introduced by Pawlak in 1982 [9], is one of the powerful mathematical techniques in data mining. It works well in decision situations which involve inconsistent information [6, 7, 13].

Nowadays, people not only concern about the accuracy but also the performance of a data mining program. There are usually just two ways to improve the performance, either increasing the speed of computers or decreasing the complexity of algorithms. However, both ways are tough. In the information age, the volume of data grows exponentially. The growing speed of CPU performance is much slower than the growing speed of data. It is even harder to decrease the complexity of algorithms that already exist. To solve this problem, many parallel techniques are developed. MapReduce is a new program model proposed by Google [1]. Its basic idea is to split a large task into small pieces and assigns them to different machines. It is easy to use and allows a big task to be split into small pieces and be done in different machines. The model abstracts the processes into two parallel processes, Map and Reduce. The two algorithms proposed in this paper are implemented with Hadoop [11], which is a framework driving from Google MapReduce. To process massive data with rough sets, Zhang et al. proposed a parallel method for computing rough set approximations [12, 14]. However, that method can only process categorical data rather than numerical data. In real-life applications, most of data are heterogeneous, including categorical and numerical. Therefore, in this paper, we discuss how to process numerical data in parallel for feature selection and classification.

The rest of the paper is organized as follows. The basic concepts and definitions are provided in Sect. 2. Section 3 presents two new parallel rough set-based algorithms. The experimental results are shown in Sect. 4. Section 5 describes the conclusion and future research directions.

2 Preliminaries

In this section, we review some basic concepts of neighborhood rough sets [4, 5] and the MapReduce model [1].

2.1 Neighborhood Rough Sets

An information system is defined as a 4-tuple (U, A, V, f), where U is a non-empty finite set that contains all objects. A is a non-empty finite set of features. $V = \bigcup_{a \in A} V_a, V_a$ is a domain of the feature a. $f : U \times A \rightarrow V$ is an information function. The information system is a decision table if $A = C \cup D$, where $C \cap D = \emptyset$. D is usually called the decision, while C is usually called features.

Definition 1 Let $B \subseteq C$ be a subset of attributes $x \in U$. The neighborhood $\delta_B(x)$ of x in the feature space B is defined as

$$\delta_B(x) = \{y \in U | \Delta_B(x, y) \leq \delta\} \tag{1}$$

where $\Delta_B(x, y)$ represents the distance between x and y.

The distance between x and y in B is defined as

$$\Delta_B(x, y) = \left(\sum_{i=1}^{m} |f(x, a_i) - f(y, a_i)|^P \right)^{1/P} \tag{2}$$

where $a_i \in B$.

Definition 2 Given a neighborhood decision table $NDT = (U, C \cup D, V, f)$, suppose $U/D = \{D_1, D_2, \cdots, D_N\}, \delta_B(x)$ is the neighborhood information granules including x and generated by attributes of $B \subseteq C$. Then the lower and upper approximations of the decision D with respect to B are defined as

$$\underline{N}_B(D) = \bigcup_{i=1}^{N} \underline{N}_B(D_i) \tag{3}$$

$$\overline{N}_B(D) = \bigcup_{i=1}^{N} \overline{N}_B(D_i) \tag{4}$$

where

$$\underline{N}_B(D_i) = \{x \in U | \delta_B(x) \subseteq D_i\}$$

$$\overline{N}_B(D_i) = \{x \in U | \delta_B(x) \cap D_i \neq \emptyset\}$$

The boundary region of D with respect to B is defined as

$$BN(D) = \overline{N}_B(D) - \underline{N}_B(D) \tag{5}$$

Definition 3 The lower approximations of the decision, also called the positive region of the decision, denoted by $POS_B(D)$, is the subset of the object set whose neighborhood granules consistently belong to one of the decision classes. It is easy to show that all elements in $\delta_B(x_i)$ have the same decision, $\forall x_i \in POS_B(D)$.

Definition 4 The dependency degree of B with respect to D is defined as

$$\gamma_B(D) = \frac{|POS_B(D)|}{|U|} \tag{6}$$

Definition 5 Given the decision table $(U, C \cup D), B \subseteq C$, the significance of the attribute $a \in B$ is defined as

$$\text{Sig}(a, B, D) = \gamma_B(D) - \gamma_{B-\{a\}}(D) \tag{7}$$

Another definition of the significance of an attribute is denoted as follows.

Definition 6 Given the decision table $(U, C \cup D), B \subseteq C$, the significance of the attribute $a \in C, a \notin B$ is defined as

$$\text{Sig}(a, B, D) = \gamma_{B\cup\{a\}}(D) - \gamma_B(D) \tag{8}$$

2.2 MapReduce Model

MapReduce is designed to handle and generate large scale, no dependency, data sets in distributed environment [1]. It provides a convenient way to parallelize data analysis process. Its advantages include conveniences, robustness, and scalability. The basic idea of MapReduce is to split the large input data set into many small pieces and assign small tasks to different devices. To work in the cluster, MapReduce model usually works with the distributed file system which provides storage [2].

3 Parallel Approaches Based on MapReduce

3.1 Parallel Neighborhood Classification

The basic idea of the neighborhood classification algorithm is to make decisions for the data based on its neighbor. For each test data, the program will calculate the distance between itself and each training data to determine the neighborhood and use the majority voting to make decisions. Algorithm 1 shows the sequential algorithm [5]. The value of δ is important to the accuracy. The program cannot make decisions if δ is too small because all the training data will not be included. The program could make wrong decisions if δ is too big because all the training data will be included. Hu et al. [5] calculated δ as

$$\delta = \min(\Delta(x_i, s)) + w \cdot \text{range}(\Delta(x_i, s)), w \leq 1 \tag{9}$$

Algorithm 1: NEighborhood Classifiers (NEC)

input : Training set: $(U, C \cup D)$, Test sample: s; Threshold δ; Specify the norm used.
output: Class of s

1 Compute the distance between s and $x_i \in U$ with the used norm;
2 Find the samples in the neighborhood $\delta(s)$ of s;
3 Find the class d_j with the majority training samples in $\delta(s)$;
4 Assign d_j to the test s.

Algorithm 2: PNEC-Map

input : $< key, value >$
// key is the byte offset of the line and value is the information of test data
output: $< key', value' >$
// key' is the ID of test data and value' is the decision of test data

```
1 begin
2   U = GetFileFromCache(); // get the training data set from
    distributed cache
3   for x_i ∈ U do
4     distance[i] = CalcDistance(value, x_i); // calculate all the distances
5   end
6   value'=Vote(distance); // majority vote
7   key'=GetID(value);
8   output.collect(key',value');
9 end
```

The parallel NEC proposed by this paper is designed to handle large test data sets which means the program will make decisions not just for one test data but for huge number of test data. The whole test data set will be put into the input file, one item per line. The training data set will be put in another file and added into the distributed cache. The test data will be separated into many Map tasks and assigned to different machines. The Map function will get the training data from the distributed cache and find out the neighborhood of the test data. After that majority voting will be employed to make decisions for the test data. The Parallel Neighborhood Classification (PNEC) is originally a Map-only algorithm.

3.2 Parallel Neighborhood Feature Selection

Even though many features have been recorded, some of them may not relate to the decision. Some of them may even guide the program to make wrong decisions. Too many features will make the program too time consuming. The purpose of feature selection algorithms is to improve the accuracy and performance of classifiers. The total number of possible combination is $2^{|C|}$, where $|C|$ is the number of features. It will be too time consuming to check all the possible combination, so a greedy feature selection algorithm was proposed [5]. The basic idea of the algorithm is to find the optimized combination in each iteration. There is a reduction set which is initially empty in the program. In each iteration, each candidate is firstly combined with the features in the reduction set, and then the significance of this combination is calculated. The feature which leads to maximal significance will be added into the reduction set. The program will terminate when the maximal significance is less than ε (a small positive real number used to control the convergence) or all candidates are added into the reduction set. Different from the neighborhood classifiers, the threshold of the neighborhood is a constant.

Algorithms 3, 4, 5 show the pseudo codes and Fig. 1 shows the process. In Algorithm 5, ε is a small positive real number used to control the convergence.

Algorithm 3: PNFS-Map

input : $< key, value >$, reduction set: red, candidate set: C, threshold δ
// key is the byte offset of the line and value is the information of training data
output: $< key', value' >$
// key' is the name or ID of the candidates. Its value will be 1 if this training data is in the positive region when the combination contains this feature and 0 means it is not in the positive region

```
1  begin
2      U = GetFileFromCache(); // get the training data set from the
       distributed cache
3      for a_i ∈ C do
4          boolean result = isPositive(value, U, δ, red ∪ a_i);
5          if result == True then
6              value' = 1;
7          else
8              value' = 0;
9          end
10         key' = a_i;
11         output.collect(key', value');
12     end
13 end
```

Algorithm 4: PNFS-Reduce

input : $< key, V >$ *// key is the name or ID of candidates and V is a list of 1 or 0*
output: $< key', value' >$ *// key' is the name or ID of candidates and value' is number of data in the positive region*

```
1  begin
2      key' = key, value' = 0;
3      for val ∈ V do
4          value' += val;
5      end
6      output.collect(key', value');
7  end
```

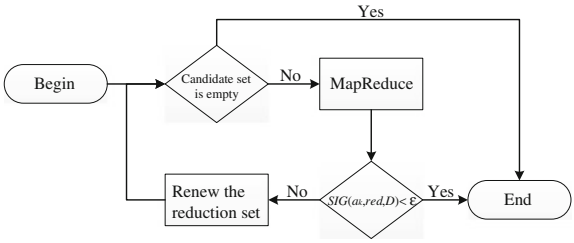

Fig. 1 Process of parallel neighborhood feature selection

Algorithm 5: Parallel Neighborhood based Feature Selection-Main

input : $(U, C \cup D)$ and δ
output: Reduction set: *red*

```
1  begin
2      ∅ → red;
3      while C ≠ ∅ do
4          Call MapReduce(U, red, C, δ); // Calculate SIG(a_i, red, D), ∀a_i ∈ C
5          Select the attribute a_k which satisfies SIG(a_k, red, D) = max(SIG(a_i, red, D));
6          if SIG(a_k, red, D) < ε then // ε is a little positive real number
7              break;
8          else
9              C − a_k → C;
10             red ∪ a_k → red;
11         end
12     end
13     Return red;
14 end
```

4 Experimental Analysis

4.1 Experimental Environment

Our experiments were conducted on large clusters of computing machines. In detail, the task nodes consist of two kinds of machines. One kind of machines has 16 GB main memory and uses AMD Opteron Processor 2376 with two Quad-Core CPUs. The other kind of machines has 8 GB main memory and uses Intel Xeon CPU E5410, comprising two Quad-Core CPUs. Hence, there are 64 working cores in all. The operating system in these machines is Linux CentOS 5.2 Kernel 2.6.18. All experiments run on the Apache Hadoop platform [11].

4.2 A Comparison of Sequential and Parallel Algorithms

Here, we give a comparison of sequential and parallel algorithms. To simplify the process, Euclidean distance is used in all the experiments. To compare with the sequential algorithms in [5], seven data sets from machine learning data repository,

Table 1 Data sets

	Data sets	Records	Features	Decision
1	Cardiotocography	2126	23	3
2	Credit	690	14	2
3	Ionosphere	351	34	2
4	Iris	150	4	3
5	MAGIC-gamma-telescope	19020	11	2
6	Sonar-mines-versus-rocks	208	60	2
7	WDBC	569	31	2

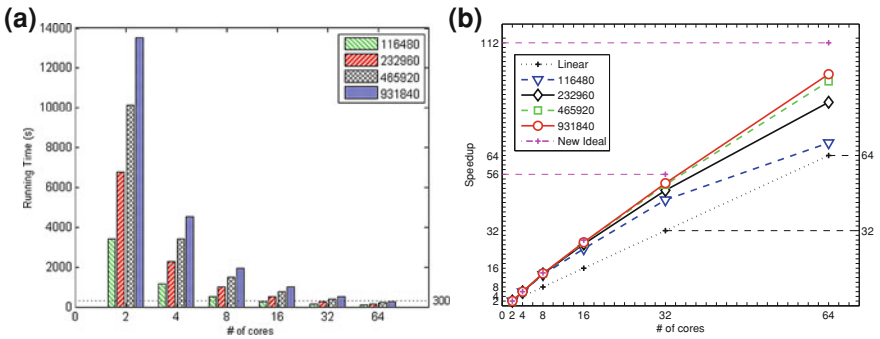

Fig. 2 **a** Running time and **b** Speedup plots of PNEC versus the number of cores used for 116480, 232960, 465920, 931840 records/60 features

University of California at Irvine [8], are selected in the experiments. Table 1 shows the basic information of data sets, all of which are numerical data.

First, we use sequential algorithm NFS and parallel algorithm parallel neighborhood feature selection (PNFS) to select relevant feature subsets with two different sizes of the neighborhood. The experiments demonstrate that the sequential and parallel algorithms have the same results.

4.3 Performance Analysis

In the experiment, to test the performance of PNEC and PNFS, the data set Sonar-Mines-versus-Rocks in Table 1 is used as input many times to stimulate the large input condition.

We mainly measured the speedup of both parallel algorithms [10]. Figure 2 shows the running time and speedup of different size data with different numbers of cores. Using the same number of cores, every time the input size is double, and the running time is double, and the running time is double. It is easy to find that the speedup is superlinear.

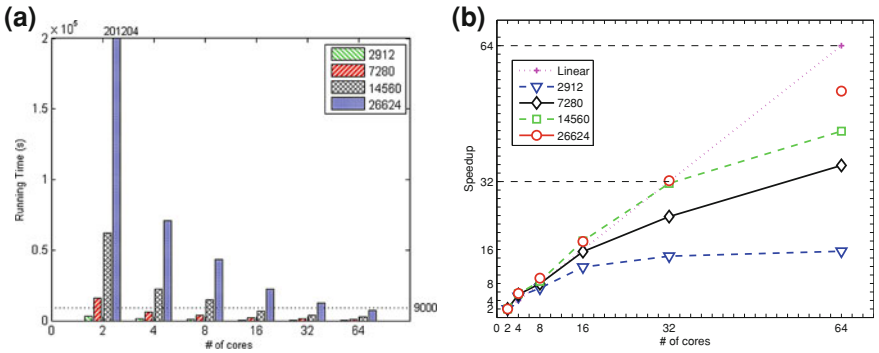

Fig. 3 Running time and speedup plots of PNFS versus the number of cores used for 2912, 7280, 14560, 26624 records/60 features

Figure 3 shows the running time and speedup of the PNFS algorithm with different input data sizes and different number of cores. Here, we find that PNFS is neither superlinear speedup nor as excellent as PNEC. In spite of this, the results show that PNFS still has a good speedup with different input data sizes. And the speedup is getting higher with the increasing size of data. When the input size is double, the running time is quadruple if the same number of cores is used.

5 Conclusions

There is no doubt that data plays an extremely important role nowadays. Discovering precise knowledge from data is one of the hot topics. However, the amount of data grows exponentially that the traditional techniques could not keep up with the rhythm. Parallelizing the sequential algorithm is one of the best ways to solve this problem. We mainly parallelized two algorithms in neighborhood rough sets: classification and feature selection. Similarly, the proposed parallel approaches are also applicable to other classification and feature selection algorithms based on neighborhood rough sets. The experimental results showed both of them not only had the same result with the sequential algorithm, but also had good scalability. Through implementing the algorithms in Hadoop and lots of experiments, we acquire the following optimizing strategies which can help process data efficiently.

- We find that the partial data will be used every time. Hence, we store this part of the data to the distributed cache, which greatly reduces I/O time.
- According to the characteristic of Hadoop and MapReduce, we put most of computation into the Map phase and only integrate results in Reduce phase, which greatly reduces the overhead of communication.

More issues about load balancing will be discussed in future. Besides, different kinds of data could be processed through the developing algorithms since the proposed algorithms in this paper can only process the numerical data.

Acknowledgments This work is supported by the National Science Foundation of China (Nos. 61175047, 61100117, 61202043), the US National Science Foundation (No. OCI-1156733), the Science and Technology Planning Project of Sichuan Province (No. 2012RZ0009), China, and the Fostering Foundation for the Excellent Ph.D. Dissertation of Southwest Jiaotong University (No. 2012ZJB), China.

References

1. Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. Commun ACM 51(1):107–113
2. Ghemawat S, Gobioff H, Leung ST (2003) The google file system. In: ACM symposium on operating systems principles, pp 29–43
3. Han J, Kamber M, Pei J (2011) Data mining: concepts and techniques, 3rd edn. Morgan Kaufmann, San Francisco
4. Hu Q, Yu D, Liu J, Wu C (2008) Neighborhood rough set based heterogeneous feature subset selection. Inf Sci 178(18):3577–3594
5. Hu Q, Yu D, Xie Z (2008) Neighborhood classifiers. Expert Syst Appl 34(2):866–876
6. Hu Q, Pedrycz W, Yu D, Lang J (2010) Selecting discrete and continuous features based on neighborhood decision error minimization. IEEE Trans Syst Man Cybern B Cybern 40(1):137–150
7. Li T, Ruan D, Geert W, Song J, Xu Y (2007) A rough sets based characteristic relation approach for dynamic attribute generalization in data mining. Knowl Based Syst 20:485–494
8. Newman D, Hettich S, Blake C, Merz C (1998) UCI repository of machine learning databases. University of California, Department of information and computer science, Irvine, CA. (http://archive.ics.uci.edu/ml/)
9. Pawlak Z (1991) Rough sets: theoretical aspects of reasoning about data, system theory, knowledge engineering and problem solving, vol 9. Kluwer Academic Publishers, Dordrecht
10. Ruan Y, Guo Z, Zhou Y, Qiu J, Fox G (2012) Hymr: a hybrid mapreduce workflow system. In: Proceedings of ECMLS 12
11. White T (2010) Hadoop: the definitive guide, 2nd edn. O'Reilly Media/Yahoo Press, Sebastopol, USA
12. Zhang J, Li T, Ruan D, Gao Z, Zhao C (2012) A parallel method for computing rough set approximations. Inf Sci 194:209–223
13. Zhang J, Li T, Ruan D, Liu D (2012) Rough sets based matrix approaches with dynamic attribute variation in set-valued information systems. Int J Approximate Reasoning 53(4):620–635
14. Zhang J, Wong JS, Li T, Pan Y (2014) A comparison of parallel large-scale knowledge acquisition using rough set theory on different mapreduce runtime systems. Int J Approximate Reasoning 55(3):896–907