Parallel Implementation of Classification Algorithms Based on Cloud Computing Environment

Lijuan Zhou, Hui Wang, Wenbo Wang
Capital Normal University, Information Engineering College, Beijing, China, 100048
e-mail: wanghui861218@163.com

Abstract
As an important task of data mining, Classification has been received considerable attention in many applications, such as information retrieval, web searching, etc. The enlarging volumes of information emerging by the progress of technology and the growing individual needs of data mining, makes classifying of very large scale of data a challenging task. In order to deal with the problem, many researchers try to design efficient parallel classification algorithms. This paper introduces the classification algorithms and cloud computing briefly, based on it analyses the bad points of the present parallel classification algorithms, then addresses a new model of parallel classifying algorithms. And it mainly introduces a parallel Naïve Bayes classification algorithm based on MapReduce, which is a simple yet powerful parallel programming technique. The experimental results demonstrate that the proposed algorithm improves the original algorithm performance, and it can process large datasets efficiently on commodity hardware.

Keywords: Naïve Bayes, Classification, MapReduce, Hadoop

Copyright © 2012 Universitas Ahmad Dahlan. All rights reserved.

1. Introduction
Now, the rapid growth of the Internet and World Wide Web has led to vast amounts of information available online considered as Big Data. The storing, managing, accessing, and processing of this vast amount of data represents a fundamental need and an immense challenge in order to satisfy needs to search, analyse, mine, and visualize this data as information. Efficient parallel classification algorithms and implementation techniques are the key to meeting the scalability and performance requirements entailed in such scientific data analyses. So far, several researchers have proposed some parallel classification algorithms. All these parallel classification algorithms have the following flaws [1]: a) they all assume that all objects can bide in memory simultaneously; b) The parallel systems have offered restricted programming models and used the restrictions to parallelize the computation automatically. Both assumptions are prohibitive for the datasets composed with millions of objects. Therefore, dataset oriented parallel classifying algorithms should be developed. And the parallel algorithms should run on tens, hundreds, or even thousands of servers.

For the emergence of cloud computing, parallel techniques are able to solve more challenging problems, such as heterogeneity and frequent failures. Cloud computing architectures which can support data parallel applications are a potential solution to the terabyte and petabyte scale data processing requirements of Big Data computing [2]. And several solutions have emerged including the MapReduce architecture pioneered by Google and now available in an open-source implementation called Hadoop used by Yahoo, Facebook, and others. In this paper, we adapt classification algorithms in MapReduce framework which is implemented by Hadoop to make the classifying method applicable to large scale data. We conduct comprehensive experiments to evaluate the proposed algorithm by actual datasets. The results demonstrate that the efficiency of the proposed algorithm is higher than the initial algorithm.

The rest of the paper is organized as follows. Section 2 introduces MapReduce. Section 3 presents the parallel Naïve Bayes algorithm based on MapReduce framework. Section 4 shows experimental results and evaluations. Finally, the conclusions and future work are presented in Section 5.
2. MapReduce Overview

MapReduce is a software framework introduced by Google in 2004 to support distributed computing on large data sets on clusters of computers. The MapReduce programming mode is designed to compute large volumes of data in a parallel fashion [3]. The model divides the workload across the cluster. It divides the input into input splits. When clients submit a job to the framework, a single map processes an input split. And each split is divided into records; the map processes each record in turn. The client does not need to deal with InputSplits directly, because they are created by an InputFormat. An InputFormat is responsible for creating the input splits and dividing them into records. The framework assigns one split to each map function. The JobTracker pushes work out to available TaskTracker nodes in the cluster, striving to keep the work as close to the data as possible by the rack-aware file system. The TaskTracker will process records in turn. The MapReduce framework makes the guarantee that the input to every reducer is sorted by key. The process performs the sort and transfers the map outputs to the reducers as inputs known as the shuffle. The map function not simply writes its output to disk. The process takes advantage of buffering written in memory and doing some pre-sorting for efficiency reasons. Figure 1 shows what happens.

3. Parallel Naïve Bayes Algorithm Based on MapReduce

In this section we present the main design for Parallel Naïve Bayes based on MapReduce. Firstly, we give a brief overview of Naïve Bayes algorithm and analyse the parallel parts and serial parts in the algorithms. Then we explain how the necessary computations can be formalized as map and reduce operations in detail.

3.1. Naïve Bayes Algorithm

Naïve Bayes is a statistical classification method. It is a well-studied probabilistic algorithm which often used in classifications. It uses the knowledge of probability and statistics for classification. Studies comparing classification algorithms have found Naïve Bayes is comparable in performance with decision tree and selected neural network classifiers. Naïve Bayes have also exhibited high accuracy and speed when applied to large databases. The Naïve Bayes classifier assumes that the presence of a particular feature of a class is unrelated...
to the presence of any other features on a given the class variable. This assumption is called class conditional independence.

To demonstrate the concept of Naïve Bayes Classification, consider the knowledge of statistics. Let \( Y \) be the classification attribute and \( X = (x_1, x_2, \ldots, x_k) \) be the vector valued array of input attributes, the classification problem simplifies to estimating the conditional probability \( P(Y | X) \) from a set of training patterns. \( P(Y | X) \) is the posterior probability, and \( P(Y) \) is the prior probability.

Suppose that there are \( m \) classes, \( Y_1, Y_2, \ldots, Y_m \). Given a tuple \( X \), the classifier will predict that \( X \) belongs to the class having the highest posterior probability. The Naïve Bayes classifier predicts that tuple \( X \) belongs to the class \( Y_i \) if and only if

\[
P(Y_i | X) \geq P(Y_j | X)
\]

(1)

The Bayes rule states that this probability can be expressed as the formulation

\[
P(Y_i | X) = \frac{P(X | Y_i)P(Y_i)}{P(X)}
\]

(2)

As \( P(X) \) is constant for all classes, only \( P(X | Y_i)P(Y_i) \) needs be maximized. The prior probabilities are estimated by the probability of \( Y_i \) in the training set. In order to reduce computation in evaluating \( P(X | Y_i) \), the Naïve Bayes assumption of class conditional independence is made. So the equation can be written into the form of

\[
P(X | Y_i) = \prod_{k=1}^{n} P(x_k | Y_i)
\]

(3)

and we easily estimate the probabilities \( P(X_1 | Y_i), P(X_2 | Y_i), \ldots, P(X_k | Y_i) \) from the training tuples. The predicted class label is the class \( Y_i \) for which \( P(X | Y_i)P(Y_i) \) is the maximum.

### 3.2. Naïve Bayes Based on MapReduce

Cloud Computing can be defined as a provision through the Internet of all computing services. It is the most advanced version of the client-server architecture and takes the system to a very high level of resource which is sharing and scaling. The resource pools composed of a large number of computing resources which are used to create highly virtualized resources dynamically for users. But for the analysis task of massive data, the cloud platform lack parallel implementation of massive data mining and analysis algorithms [4]. Therefore, a new cloud computing model of massive data mining includes the pre-processing for huge amounts of data, cloud computing for massive parallel data mining algorithms, the new massive data mining methods and so on [5].

The critical problem of the massive data mining is the algorithm parallelization of data mining. Cloud computing uses the new computing model known as MapReduce, which means that the existing data mining algorithms and parallel strategies cannot be applied directly to cloud computing platform for massive data mining, so some transformation must be done. Based on this, for the characteristics of massive data mining algorithms, the cloud computing model has been optimized and expanded to make it more suitable for massive data mining [6]. Therefore, this paper adopts the Hadoop distributed system infrastructure, which provides the storage capacity of HDFS and the computing capability of MapReduce to implement parallel classification algorithms.

The implementation of the parallel Naïve Bayes’s MapReduce model is divided into training and prediction stages.

#### 3.2.1. Training Stage

The distributed computing of Hadoop is divided into two phases which are called Map and Reduce. First, the InputFormat which is belonged to the Hadoop framework loads the input data into small data blocks known as data fragmentation, and the size of each data...
fragmentation is 5M, and the length of all of them is equal, and each split is divided into records. Each map processes a single split, and the map task passes the split to the getRecordReader() method on InputFormat to gain a RecordReader for that split. The RecordReader is iterators of the records. Then the map task uses a RecordReader to generate record key-value pairs, which passes to the map function.

Secondly, the map function statistics the categories and properties of the input data, including the values of categories and properties. The attributes and categories of the input records are separated by a comma, and the final attribute is the property of classification.

Finally, the reduce function aggregates the number of each attribute and category value, which results in the form of (category, Index1:count1, Index2:count2, Index3:count3, \ldots, Indexn:countn), and then output the training model. Its implementation is described as follows.

Algorithm Produce Training: map(key, value)
Input: the training dataset
Output: <key’, value’> pair, where key’ is the category, and value’ the frequency of attribute value

1 for each sample do begin
2 \hspace{1em} \text{Parse the category and the value of each attribute}
3 \hspace{1em} count—the frequency of the attributes
4 \hspace{1em} for each attribute value do begin
5 \hspace{2em} \text{Take the label as key’, and attribute index: the frequency of the attribute value as value’}
6 \hspace{1em} output<key’, value’>
7 end
8 end

Algorithm Produce Training: reduce(key, value)
Input: the key and value output by map function
Output: <key’,value’> pair, where key’ is the lable, and value’ the result of frequency of attribute values

1 \hspace{1em} \text{sum-0}
2 \hspace{1em} for each attribute value do begin
3 \hspace{2em} \text{sum+=value.next.get()}
4 \hspace{1em} end
5 \hspace{1em} \text{Take key as key’, and sum as value’}
6 \hspace{1em} output<key’, value’>

3.2.2. Prediction Stage
Predicate the data record with the output of the training model. The implementation of the algorithm is stated as follows: first, use the statistical values of attribute values and category values to train the unlabeled record. In addition, use the distributed cache to improve the efficiency of the algorithm in the procession of the algorithm implementation. Its implementation is described as follows.

Algorithm Produce Testing: map (key,value)
Input: the test dataset and the Naive Bayes Model
Output: the labels of the samples

1 \hspace{1em} modeltype—newModelType()
2 \hspace{1em} categories—modeltype.getCategorys()
3 \hspace{1em} for each attribute value not NULL do begin
4 \hspace{2em} \text{Obtain one category from categories}
5 \hspace{2em} end
6 \hspace{1em} for each attribute value do begin
7 \hspace{2em} \text{pct—counter(attribute, category)/Counter(category)}
8 \hspace{2em} \text{result—result*pct}
9 \hspace{2em} end
10 \hspace{1em} end
11 \hspace{1em} output<key’,value’>

12 \hspace{1em} take the category of the max result as key’, and the max result as value’
4. Experimental Results

In this section, we perform some preparatory experiments to test the efficiency and scalability of parallel Naïve Bayes algorithm proposed in this paper. We build a small cluster with 3 business machines (1 master and 2 slaves) on Linux, and each machine has two cores with 3.10GHz, 4GB memory, and 500GB disk. We use the Hadoop version 0.20.2 and java version 1.6.0_26. We use the UCI data sets to verify the results. Experimental data sets are shown in Table one.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Number of samples</th>
<th>Dimension</th>
<th>Numbers of categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Wine</td>
<td>178</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>2 Vertebral</td>
<td>310</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>3 Bank-data</td>
<td>600</td>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>4 Car</td>
<td>1728</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>5 Abalone</td>
<td>4177</td>
<td>8</td>
<td>28</td>
</tr>
<tr>
<td>6 Adult</td>
<td>32561</td>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>7 PokerHand</td>
<td>1000000</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

First, the pre-treatment over the above data sets must be done, all property types normalized to nominal attributes. Then, the Naïve Bayes classifier implemented by the MapReduce trains the training data sets to generate the classify model, and then use the model to classify the removed category samples. The experiment is run on the cluster composed with three machines, and the results is shown in Figure 2, compared with the general method of test results.

![Figure 2. Executing time with different sizes](image)

The comparing experiment shows that the performance of the improved algorithms is higher than the general methods with large data set. And this verifies the Bayesian algorithm runs on the cloud environment is more efficient than the traditional Bayesian algorithm. However, due to the size of data sizes, attributes, and the number of different categories, the time that the algorithm spent is not appear a linear relationship. Since running Hadoop jobs, start the cluster first which takes a little of time, so when the size of data set is smaller, the data processing time is relatively longer. And this also verified the Hadoop is perfect to process huge amounts of data.
5. Conclusions

As data classifying has attracted a significant amount of research attention, many classification algorithms have been proposed in the past decades. However, the enlarging data in applications makes classifying of very large scale of data a challenging task. In this paper, we propose a fast parallel Naïve Bayes algorithm based on MapReduce, which has been widely embraced by both academia and industry. Preparatory experiments show that the parallel algorithms can not only process large datasets, but also enhance the efficiency of the algorithm. In the future work, we will further implement other classification algorithms and conduct the experiments and consummate the parallel algorithms to improve usage efficiency of computing resources.

Acknowledgements

This research was supported by China National Key Technology R&D Program (2012BAH20B03), National Nature Science Foundation (31101078), Beijing Nature Science Foundation (4122016), Beijing Nature Science Foundation (4112013), and Beijing Educational Committee science and technology development plan project (KM201110028018), "The computer application technology“ Beijing municipal key construction of the discipline.

References