Original Article
International Journal of Fuzzy Logic and Intelligent Systems
Vol. 14, No. 4, December 2014, pp. 313-321
http://dx.doi.org/10.5391/IJFIS.2014.14.4.313

ISSN(Print) 1598-2645
ISSN(Online) 2093-744X

Big Numeric Data Classification Using Grid-based Bayesian Inference in the MapReduce Framework

Young Joon Kim[1] and Keon Myung Lee[2]
[1]Cyberdigm, Seoul, Republic of Korea
[2]Dept. of Computer Science, Chungbuk National University, Cheongju, Republic of Korea
kmlee@cbnu.ac.kr

ljfis

Abstract

In the current era of data-intensive services, the handling of big data is a crucial issue that affects almost every discipline and industry. In this study, we propose a classification method for large volumes of numeric data, which is implemented in a distributed programming framework, i.e., MapReduce. The proposed method partitions the data space into a grid structure and it then models the probability distributions of classes for grid cells by collecting sufficient statistics using distributed MapReduce tasks. The class labeling of new data is achieved by k-nearest neighbor classification based on Bayesian inference.

Keywords: big data, classification, data mining, Hadoop, MapReduce

1. Introduction

The recent technological advances in sensors and storage devices, as well as the advent of various multimedia-based applications, mean that huge volumes of data are accumulated, which need to be processed. Big data present new practical and theoretical challenges in terms of the volume of data, the variety of data types, and the speed of processing. Thus, various platforms and algorithms have been developed and put into practice to address these issues [1, 2, 5, 6].

In the present study, we address the problem of constructing classifiers from a big data training set that comprises continuous attributes. As the volume of data increases, classification techniques that require multiple-pass processing experience severe resource demands. Thus, existing algorithms can be parallelized to handle this problem, or new single-pass algorithms need to be developed. In general, many machine learning algorithms assume that the training dataset is not large and they have been developed for training datasets that are moderate in size. Indeed, some of these techniques are not considered to be applicable to high dimensional data due to the curse of dimensionality. However, big data application domains can be free from the curse of dimensionality because the volume of data is sufficiently large.

The k-nearest neighbor classification methods determine the class of data objects based on the class distribution of proximity-based nearest neighbors [7]. They do not require any preliminary training of a classifier because they determine the classes of data objects based on the class distributions of their neighbors. These methods might not be good choices in the big

Received: Nov. 2 2014
Revised : Nov. 25, 2014
Accepted: Dec. 6, 2014

Correspondence to: Keon Myung Lee
(kmlee@cbnu.ac.kr)

data application domain because they take a long time to locate the nearest neighbors and to compute the distances between the query data objects and the data objects in the training dataset. Grid-based approaches partition the data space into regularly organized grid cells [8] and they can handle a dataset in cell units instead of individual data objects. In the present study, we propose a classification method that exploits the concepts of k-nearest neighbor classification, grid-based partitioning, and Bayesian inference, which can be applied to big data classification. To handle big data in an efficient manner, some parts of the proposed method are implemented using the MapReduce paradigm [1] in the Hadoop framework [2].

The remainder of this paper is organized as follows. Section 2 provides a brief introduction to the Hadoop framework and the MapReduce programming paradigm, as well as k-nearest neighbor classifiers. Section 3 explains the proposed method for building a classifier from big data with continuous attributes. Section 4 presents some experiment results obtained using the proposed method and Section 5 gives our conclusions.

2. Related Work

2.1 Hadoop and MapReduce

Considerable amounts of time and space are required to handle large volumes of data. Occasionally, it might not be possible to accommodate a large dataset on a single computer. This is one of the main reasons for using a distributed file system, where a file system can be networked among multiple servers to manage big files. The servers in the distributed file system need to be robust against failure because a failure at any point could cause the entire system to malfunction. Thus, the files are partitioned into blocks and multiple replicas of the blocks are stored in different servers. Hadoop is an Apache project for a distributed computing platform and it provides a distributed file system called the Hadoop Distributed File System [2].

It is not sufficient to process or analyze big data to obtain a distributed file system. Instead, it is necessary to produce a framework for developing distributed parallel processing applications that can handle big data files. A simple approach to performing a task in a distributed, parallel manner is to decompose the task into independent subtasks, where they can be processed without communication between them. The MapReduce model [1] is a programming paradigm that allows a program to be organized with independently executable processing modules called *map* and *reduce*. *map* is an operation that takes an input and produces one or more pairs of $(key, value)$, where key usually acts as the identifier of a pair for which $value$'s are later collected. *reduce* is an operation that processes the *value*s with the same *key*. The *map* functions cannot share any information, which allows multiple instances to be executed simultaneously for different parts of the input data. To employ the MapReduce paradigm in data processing, we need to devise a method for describing data processing tasks as a sequence of *map* and *reduce* operations, which requires a suitable algorithm. The *map* and *reduce* operations are paired, but sometimes it is not sufficient to have a pair of *map* and *reduce*. In this case, a pair of *map* and *reduce* can be chained, where the *reduce* output from the preceding phase is used as an input for *map* in the following phase.

To execute MapReduce-based programs, a framework is required to deploy them in a distributed system and to manage resources, such as instantiating multiple *map* and *reduce* tasks, delivering data from and to the distributed file system, recovering from failure, and monitoring the status of the working tasks and the networks. Hadoop provides an efficient framework for running MapReduce-based programs [2].

2.2 k-Nearest Neighbor Classifiers

The k-nearest neighbor classifiers comprise nonparametric classification methods for classifying data objects based on the k closest training data objects in the data space. They classify a data object based on a majority vote among its neighbors, i.e., the k nearest neighbors of the data object. These methods do not construct explicit classifier models and they determine the class labels of incoming data by referring to their neighbors on the fly. These approaches are among the simplest machine learning classifiers.

When a query data object arrives, a search is performed to find the k nearest neighbors to the object. The distances between the query object and training data objects in the training set must be evaluated to determine the neighbors. Various distance measures for numeric data have been developed such as the Euclidean distance, Manhattan distance, Mahalanobis distance, and cosine distance [8]. If the data contain categorical attributes, other distance measures are needed such as the Jaccard distance and Tanimoto distance [14].

If the data occupy a high dimensional space and the number of data objects is large, it is not easy to locate the k nearest neighbors. Thus, several data structures have been developed for indexing neighbors, e.g., k-d tree, hierarchical k-means, ball tree, spill tree, and spatial tree [11]. However, the perfor-

mance of the indexing structure degrades as the volume of data increases. To overcome this difficulty, locality-sensitive hashing techniques [11] have been developed, which use hashing functions to make similar data objects collide in the same or neighboring buckets. These techniques require the maintenance of special hashing structures to record the data objects that belong to each bucket.

3. MapReduce-based k-nearest Neighbor Bayesian Classifier

In this study, we consider a classification problem for a dataset that comprises numeric data with continuous attributes. Each training data object is labeled with its class. The volume of data can be huge and thus we cannot construct classifiers by scanning the training data multiple times in practical applications. The volume of data can be a bottleneck for conventional k-nearest neighbor classifiers. Thus, we propose a new k-nearest neighbor classification method where we employ grid partitioning of the data space and Bayesian inference to make class decisions. To process big data in an effective manner, the time-consuming tasks in the proposed method are implemented using MapReduce operations.

It is assumed that the dataset $D = \{d_i | i = 1, \ldots, N,\ d_i = (a_{i1}, \ldots, a_{iM}; c_i)\}$ comprises a collection of data d_i with M numeric attribute values $(a_{i1}, \ldots, a_{iM})$ with class label c_i. Thus, there are M attributes $(A_1, \ldots, A_M)$ such that $a_{ij} \in Range(A_j) = [L_j, U_j],\ i = 1, \ldots, N,\ j = 1, \ldots, M$. Each data object has a class label, s.t., $c_i \in \{K_1, \ldots, K_p\}$.

To handle large volumes of data, the proposed method organizes the data space into a grid structure where the dimension of each attribute is partitioned at equidistant intervals. This partitioning process divides the data space into cells, where each cell maintains the class-wise statistical information for the data subset that belongs to the cell. Instead of using individual training data objects, the method models them based on the probability distributions of each class. The classification of query data is performed by Bayesian inference using the prior probabilities and likelihoods of the neighboring cells of the query data.

The proposed strategy has two phases: the training phase and inference phase. The inference phase comprises the following steps.

Step 1. Partition the data space into grids.

Step 2. Apply the k-means clustering algorithm to class-wise data in the grids with different numbers of clusters. Determine the appropriate number of clusters for each class in the grids.

Step 3. Compute the statistics for each cluster.

Step 4. Build Gaussian models of the grids.

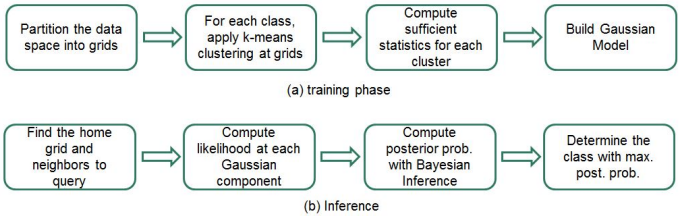

Figure 1. Proposed strategy for the classification of big numerical data. (a) Learning phase for collecting the information used in Bayesian inference. (b) Query inference phase.

After constructing Gaussian models for the grids, class inference is performed for a query according to the following sequence.

Step 1. Find the home grid and the neighbors of a query.

Step 2. Compute the likelihoods of the query for each Gaussian component.

Step 3. Compute the posterior probabilities of the classes by Bayesian inference.

Step 4. Determine the class of the query with the highest posterior probability.

3.1 Data Space Partitioning

The continuous attribute domains are partitioned into several intervals, e.g., equidistant intervals, user-specified intervals, equi-frequency intervals, or data distribution-based partitions. Data processing is straightforward for equidistant interval partitioning and user-specified partitioning, but equi-frequency interval partitioning requires that the attribute values are sorted, which is a computationally intensive task. Distribution-based partitioning requires some baseline information for each attribute, such as the means and standard deviations.

The proposed method partitions each attribute dimension at an equidistant interval. If we suppose that attribute $A_i (i = 1, \cdots, M)$ with the range $[L_i, U_i]$ is partitioned into R_i intervals, then the j-th interval I_{ij} of A_i is the following interval.

$$I_{ij} = [L_i + (j-1)\Delta_i, L_i + j * \Delta_i),\ \ j = 1, \cdots, R_i - 1 \tag{1}$$

$$I_{iR_i} = [L_i + (R_i - 1)\Delta_i, L_i + R_i * \Delta_i] \tag{2}$$

$$\Delta_i = (U_i - L_i)/R_i \tag{3}$$

Each attribute may have a different numbers of intervals. A cell in the data space is indexed by $(k_1, k_2, \cdots, k_M)$, where the

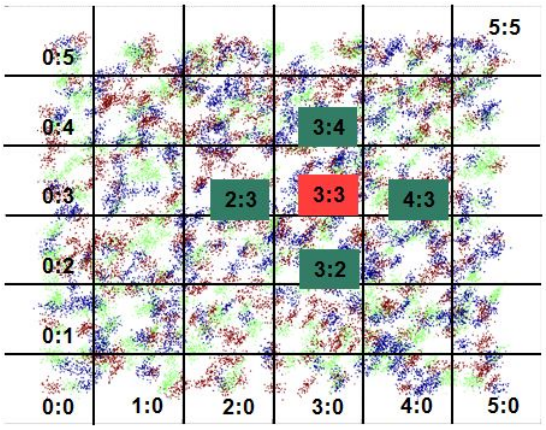

Figure 2. Space partitioning and labeling. The label of a grid is obtained by combining the labels of the corresponding axis labels. On each axis, the intervals are labeled with consecutive integers and thus the neighboring intervals can be identified by either addition or subtraction. The neighboring grids of the grid with the label 3:3 are labeled as 2:3, 3,4, 4:3, and 3:2.

corresponding cell is specified by the intervals $(I_{1k_1}, I_{2k_2}, \cdots, I_{Mk_M})$.

When a data object d_p is described by $(x_1, x_2, \cdots, x_M)$, the index $(k_1, k_2, \cdots, k_M)$ of its corresponding cell is determined as follows:

$$k_i = \lfloor (x_i - L_i)/\Delta_i \rfloor \quad i = 1, \ldots, M, \tag{4}$$

which means that we can easily locate the home cell of a data object via a simple arithmetic operation even in a high dimensional space. The neighboring cells are easily determined by the indices. The indices $(I_{1k_1}, I_{2k_2}, \ldots, I_{Mk_M})$ of 1-distance cells for a cell $(I_{1k_1}, I_{2k_2}, \ldots, I_{Mk_M})$ are determined as follows.

$$I_{ik_i} = I_{ik_i} + \delta_1 \quad \delta_1 \in \{-1, 0, 1\} \tag{5}$$

In a similar manner, the indices of s-distance cells are determined as follows.

$$I_{ik_i} = I_{ik_i} + \delta_s \quad \delta_s \in \{-s, \cdots, -1, 0, 1, \cdots, s\} \tag{6}$$

An s-distance cell is a cell located at a grid distance s from the home cell.

3.2 Class-wise k-Means Clustering for Cluster Determination

The proposed method models each grid as a mixture of Gaussians for each class. The constituent clusters are determined for each class using the k-means algorithm, which requires a pre-specified number of clusters. No information is available about how many clusters exist in a class of grids. Thus, a cluster validity index is used to determine the number of clusters. In a grid, a class usually has a small number of clusters, which might possibly be a single cluster.

Several cluster validity indices can be used to measure how well the clusters are formed [8, 20]. The R-squared (RS) index is a cluster validity index that measures the dissimilarity of clusters [20] as the degree of homogeneity between groups using values from 0 to 1, where 0 indicates that there is no difference among the clusters and 1 indicates that there are significant differences among the clusters. RS is defined as follows:

$$RS = \frac{SS_t - SS_w}{SS_t}, \tag{7}$$

where

$$SS_t = \sum_{i=1}^{|D|} ||x_i - \bar{x}||^2, SS_w = \sum_{i=1}^{N_c} \sum_{x_j \in C_i} ||x_j - \bar{x}_i||^2$$

, where x_i is the i-th data point, $\bar{x}$ is the mean of all the data, $|D|$ is the overall data size, C_i is the i-cluster, $\bar{x}_i$ is the centroid of the i-cluster, and N_c is the number of clusters. The RS index is always 0 for a dataset with a single cluster, and thus it is preferable to have more than one cluster.

In a mixture of Gaussian models, it is better to merge two neighboring clusters because small condensed Gaussian components can have biased effects on Bayesian inference. To measure the proximity of two clusters, we propose a *so-called* separability index $Sp(i, j)$ for a pair of clusters C_i and C_j:

$$Sp(i, j) = \frac{rad_p(i) + rad_p(j)}{d_{ij}}, \tag{8}$$

where $rad_P(i)$ indicates the distance to the centroid of the cluster C_i for data where the distance from the centroid is ranked among the p percentages of the top distances, and d_{ij} is the distance between the cluster centroids of clusters i and j.

The following strategy is used to determine the number of clusters for each class in a grid:

procedure Determine_No._of_Clusters

For $i = 2$ to K **do**

Apply the k-means algorithm to the dataset with i clusters

Compute the RS index score $RS(i)$ for the clustering results

end

Choose the number of clusters that yield the highest RS index

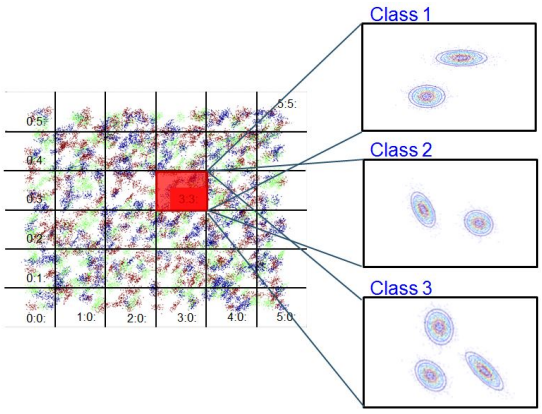

Figure 3. Identification of a mixture of Gaussians. The k-means clustering algorithm is applied to grids with various number of clusters. The number of clusters is determined for each class in the grids and their corresponding clusters.

score

For each pair of clusters among the selected number of clusters

Compute the separability index $Sp(i,j)$ for the pair (C_i, C_j)

If the index score $Sp(i,j) >$ the specified threshold θ_s

Merge the corresponding pair of clusters

3.3 Bayesian Inference and Sufficient Statistics for the Probability Distribution

Bayesian inference uses the Bayes' rule to derive the posterior probability $P(C|d)$ of a class C for the given data d as the product of the prior probability $P(C)$ of class C and the likelihood $P(d|C)$, which is the probability that class C generates the data d. The Bayes' rule is represented as follows.

$$P(C|d) = \frac{P(C)P(d|C)}{P(d)} \tag{9}$$

In the Bayesian-based classifier, a probability model is constructed for the likelihood function from the training dataset and the prior probability is determined for each class. When a data object d is given, the posterior probability for each class C is computed, and its class is assigned the class label with the highest posterior probability. In the Bayesian rule, the denominator $P(d)$ is not evaluated because it applies to each posterior computation, and thus it does not affect the determination of the class label with the highest posterior probability.

The proposed method models the probability distributions of the classes for each cell. The probability distributions are assumed to be Gaussian. Gaussian distributions can be specified by the mean and covariance matrix. The method keeps track of the sufficient statistics required for the Gaussian distributions, i.e., the numbers of data objects, sums of the attribute values, and squared sums of the attribute values for each class. To consider the correlations between attribute variables, we must maintain the sums of the products for each pair of attribute variables. Let N_{ij} be the size of the data subset D_{ij} that belongs to class C_j in cell G_i, where M_{ij} is the sum of the attribute values of data subset D_{ij} and $S_{ij,pq}$ is the sum of the products of the p-th and q-th attribute values of the data subset D_{ij}. After obtaining the sufficient statistics, the mean m_{ij} and covariance matrix Σ_{ij} for D_{ij} are computed as follows.

$$m_{ij} = \frac{M_{ij}}{N_{ij}} \tag{10}$$

$$(\Sigma_{ij})_{pq} = \frac{(S_{ij,pq})}{(N_{ij}-1)} - (m_i)_p * (m_i)_q \tag{11}$$

The sufficient statistics-based method can acquire statistical information during a single scan of the dataset, which can be updated incrementally when additional data objects are available due to the additive property of the sufficient statistics.

3.4 MapReduce Algorithm for Collecting the Sufficient Statistics

The following MapReduce operations are used to obtain the sufficient statistics for the mean and covariance matrix of the attribute values. The map function assumes that the first element of the input records is the class label and that the remaining elements are attribute values listed in a specific order. First, the map function finds the index $k_1k_2\cdots k_M$ of the grid cell occupied by the data R using Eq. (4). Next, it generates a list where the element '1' indicates the count of records, the elements from the second place to the $(M+1)$-th place contain the attribute values as they appear in the data R, and the pairwise products of the attribute values are appended to the end of the list.

procedure: map($key = null, value = record\ R$)

$L[\,]$:= splits(R)

Compute the index $k_1k_2\cdots k_M$ of the data that correspond to $L[2..M+1]$

Using Eq. (4)

Val := '1'

For i := 2 **to** $M+1$

Val := Val + " " + $L[i]$

For i : = 2 **to** $M+1$

For j:= i **to** $M+1$

$Val := Val$ + " " + $L[i] * L[j]$
write($(k_1k_2 \cdots k_M$,$L[1])$, $Val)$

The $reduce$ function aggregates all of the records that belong to the same cell by adding them in a position-wise manner. The output generated by the function comprises the key-$value$ pair(s), where key is the index for the corresponding grid cell, and $value$ comprises the $class$ label, the list of means, and the upper triangular portion of the covariance matrix.

procedure: reduce($key = (k_1k_2 \cdots k_M, class)$, $values$)
$count := 0$
$Sum[1..M] := 0$
$Var[1..(M(M+1)/2)] := 0$
Foreach val **in** $values$
$L[\ ]$ = splits(val)
$count := count + L[1]$
For i : = 1 **to** M
$Sum[i] = L[i+1]$
For i : = 1 **to** M
For $j := i + 1$ **to** M
$Var[i * (i-1)/2 + j] := Var[i * (i-1)/2 + j] + L[i * (i-1)/2 + j + M + 1]$
$Mean[1..M] := 0$
for i : = 1 **to** M
$Mean[i] := Sum[i]/count$
$Cov[1..M][1..M] = 0$
For i := 1 **to** M
For $j := i + 1$ **to** M
$Cov[i][j] := Var[i * (i-1)/2 + j]/(count - 1)$
For i := 1 **to** M
For $j := i$ **to** M
$Cov[i][j] := Cov[i][j] - Mean[i] * Mean[j]$
$Val := class$ + " " + $count$
For i : = 1 **to** M
$Val := Val$ + " " + $Mean[i]$
For i : = 1 **to** M
For $j := i$ **to** M
$Val := Val$ + " " + $Cov[i][j]$
Write($k_1k_2 \cdots k_M, Val$))

The map function produces an extended record, which is $M^2/2 + 3M/2$ times the original size M. This may impose a large burden when handling the intermediate files between map processes and $reduce$ processes. To handle this problem, we apply the $combine$ function after the map processes and before the $reduce$ processes. The $combine$ processes run on every node that runs the map processes. The input for the $combine$ process comprises all of the data generated by the map processes on a given node. The output from the $combine$ processes is then sent to the designated $reduce$ processes. The proposed method uses the $reduce$ function as the $combine$ function because their roles are similar, except for the computation of the model parameters such as the means and covariances. Using the $combine$ function dramatically reduces the bandwidth usage and the size of the intermediate files.

procedure: combine($key = (k_1k_2 \cdots k_M, class)$, $values$)
$count := 0$
$Sum[1..M] := 0$
$Var[1..(M(M+1)/2)] := 0$
Foreach val **in** $values$
$L[\ ]$ = splits(val)
$count := count + L[1]$
For i : = 1 **to** M
$Sum[i] = L[i+1]$
For i : = 1 **to** M
For $j := i + 1$ **to** M
$Var[i * (i-1)/2 + j] := Var[i * (i-1)/2 + j] + L[i * (i-1)/2 + j + M + 1]$
$Val := count$
For i : = 1 **to** M
$Val := Val$ + " " + $Sum[i]$
For i : = 1 **to** M
For $j := i + 1$ **to** M
$Val := Val$ + " " + $Var[i]$
Write(key, Val))

3.5 Classification of New Data Using Bayesian Inference

To classify a new query data, the proposed method uses the Bayesian inference rule. The MapReduce functions provide the model parameters for the Gaussian probability distributions of the classes in each grid cell. Each cell maintains the model parameters for the probability distributions of the classes. We refer to the cell that contains the query data as the home cell G_h. Let NB_h be the index set of neighboring cells and $d(G_i, G_h)$ is the distance of a cell G_i from the home cell G_h. $d(G_i, G_h) = s$ if G_i is an s-distance cell from G_h.

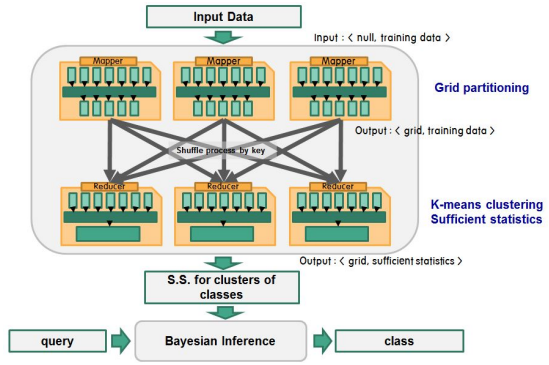

Figure 4. Proposed MapReduce-based training and inference scheme. The *Mapper* performs grid partitioning and the *Reducer* runs the k-means clustering algorithm to determine the clusters for the Gaussian components and to obtain the sufficient statistics. After a query is issued, Bayesian inference is conducted using the sufficient statistics to determine the most probable class label.

The prior probability of class C_k in cell G_i is defined as follows.

$$P_i(C_k) = \frac{N_{ik}}{\Sigma_j N_{ij}} \tag{12}$$

The likelihood $P_i(d_p|C_k)$ of data d_p in class C_k at grid G_i is defined as follows.

$$P_i(d_p|C_k) = \frac{1}{(2\pi)^{n/2}|\Sigma_{ik}|^{1/2}} exp(0.5(d_p - m_{ik})^\top \Sigma_{ik}(d_p - m_{ik})) \tag{13}$$

The posterior probability $P_i(C_k|d_p)$ of class C_k for data d_p is given as follows.

$$P_i(C_k|d_p) = \frac{P_i(C_k)P_i(d_p|C_k)}{P_i(d_p)} \tag{14}$$

The denominator $P_i(d_p)$ is the same for all classes C_k at G_i. Thus, we can compute the posterior probability without using $P_i(d_p)$, as follows.

$$P_i(C_k|d_p) = \frac{P_i(C_k)P_i(C_k|d_p)}{\sum_j P_i(C_j)P_i(C_j|d_p)} \tag{15}$$

The probability support $\hat{P}(C_k|d_p)$ of class C_k to d_p is accumulated as follows:

$$\hat{P}(C_k|d_p) = P_h(C_k|d_p) + \sum_{n \in NB_h} \frac{}{(d(G_n, G_h)+1)^w} P_n(C_k|d_p), \tag{16}$$

where $(d(G_n, G_h)+1)^w$ is a weighting factor that makes closer cells contribute more to class determination and $w(w > 0)$ is a control factor that determines the level of contribution. The scope of the neighbors can be controlled by fixing the maximum grid distance s from the home cell to the neighboring cells.

The final probability $P(C_k|d_p)$ is determined by computing the normalization of the probability supports, as follows.

$$P(C_k|d_p) = \frac{\hat{P}(C_k|d_p)}{\sum_j \hat{P}(C_j|d_p)} \tag{17}$$

The data object d_p is classified into the class $C(d_p)$ with the maximum final probability:

$$C(d_p) = \arg\max_k p(C_k|d_p). \tag{18}$$

4. Experiments

The proposed algorithm was implemented using the MapReduce libraries in Hadoop. The programs developed were executed on a Hadoop cluster with eight commodity computers. To assess the applicability of the proposed method, we applied the method to a numeric benchmark dataset: Iris[10], which comprised 150 objects with four numeric attributes and three classes. A tenfold cross-validation was performed using the dataset. To determine the appropriate probability distributions, we required a sufficiently large dataset; thus, we enlarged the dataset by sampling new data and adding Gaussian noise to the attribute values in the existing dataset, thereby obtaining a training dataset of 10,000 objects. Each attribute domain was partitioned into five equidistant intervals. The number of cells was 625. The accuracy obtained in the experiment was comparable to that obtained with the C4.5 algorithm using Iris data with 150 data objects. We found that the proposed method performed adequately in the classification of big data. The use of the *combine* function dramatically reduced the volume of intermediate files, thereby making the *reduce* phase execute over two times faster.

Another artificial dataset with three attributes and three classes was generated randomly using 900 Gaussian distributions, where the centroids of the covariance matrices were selected randomly. For each Gaussian distribution, 500 data points were generated that formed a cluster in the space. In the experiments, the range of each attribute was partitioned into five intervals. A tenfold cross-validation was performed using the dataset. According to the experiments, the average accuracy was 94.3%.

In other experimental settings, we changed the number of partitions of the attributes from two to six. There was no significant change in the accuracy, but there was a trade-off between the execution time and the storage required to maintain sufficient statistics for the grids as the number of partitions changed. The increase in the number of partitions decreased the data

Table 1. Effects of cluster merging

	without cluster merging	with cluster merging
resampled Iris data	88.6%	94%
artificial data with 900 clusters	75.8%	94.3%

volume handled by a single reducer and more reducers were instantiated, so the degree of parallelism was also increased. Because the reducers ran the k-means algorithm several times, the reduction in the data volume for each reducer improved the execution time significantly.

The merging of neighboring clusters using the separability index improved the accuracy considerably, as shown in Table 1. Thus, small condensed Gaussian distributions with small determinants in their covariance matrices had higher likelihoods for queries.

5. Conclusions

The MapReduce model is an efficient computational model for distributed parallel processing with big data. In this study, we proposed a grid-based Bayesian inference classification method that can handle big data.

The proposed method has the following advantages. It is a single-pass algorithm that does not require multiple scans of the training data to construct a classifier. It is very efficient at locating the neighbors because the home cell of a data query is determined by simple arithmetic computation and the neighboring cells are determined by index manipulation. It is scalable for big volumes of data because only the sufficient statistics for the probability distributions are maintained, which can be updated by simple arithmetic operations as data objects arrive. The neighboring class distribution can be visualized readily because the neighboring cells are easily located and their class probability distributions can be computed simply.

The proposed method has the following disadvantages. Fixed length interval-based partitioning of the data space might not capture the appropriate probability distributions of the classes. To mitigate this problem, we could decrease the interval lengths but there might be too many cells to be maintained in a high-dimensional data space. A sufficiently large training dataset is required to ensure that each grid cell has sufficient data subsets to characterize the parameters of the probability distributions. The proposed approach is not appropriate for classification problems with small training sets but it works well for big data. The costly k-means clustering algorithm is executed multiple times on each grid. As the number of grids increase, the volume of data handled by a reducer decreases and the processing time on a reducer also declines. This problem can be resolved by increasing the size of the Hadoop clusters. Further improvements can be achieved by employing a sampling technique that uses a subset of the dataset to determine the number of clusters and their centroids and covariance matrices, before collecting the sufficient statistics for the corresponding clusters with respect to the overall dataset.

Our experiments showed that the proposed system performed well as a classifier for big data. Thus, the method could be used for solving big data classification problems. In future research, we aim to extend the proposed method to the simultaneous handling of datasets with both categorical and numerical attributes.

Conflict of Interest

There are no potential conflicts of interest related to this study.

Acknowledgments

This study was supported by a research grant from Chungbuk National University in 2014.

References

[1] J. Dean, S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, *Proc. of the 6th Symp. on Operating Systems Design and Implementation*, 2004.

[2] ____,Hadoop, http://wiki.apache.org/hadoop/

[3] K.-M. Lee, K.M. Lee, C. H. Lee, "Linguistic Classification Pattern Extraction for Numeric Data," *Proc. of WSEAS*, 2012.

[4] L. Zhou, Z. Zhong, J. Chang, J. Li, J. Z. Huang, S. Feng, "Balanced Parallel FP-Growth with MapReduce," *Proc. of IEEE YC-ICT,* 2010.

[5] ____, Apache Mahout, http://mahout.apache.org/

[6] G. Caruana, M. Li, and M. Qi, "A MapReduce based Parallel SVM for Large Scale Spam Filtering," *Proc. of 8th*

Int. Conf. on Fuzzy Systems and Knowledge Discovery, pp.2659-2662, 2011.

[7] M. Geetika, "A Survey of Classification Methods and its Applications," *Int. J. of Computer Applications*, vol.53, No.17, 2012.

[8] S. Theodoridis, and K. Koutroumbnas, *Pattern Recognition*, Elsevier, 2009.

[9] D. Lu, Q. Weng, "A survey of image classification methods and techniques for improving classification performance," *Int. J. of Remote Sensing*, vol. 28, no.5, pp.823-870, 2007.

[10] M. M. Gaber, "Advances in data stream mining," *WIREs Data Mining Knowl. Discov.*, vol.2, pp.79-85, 2012.

[11] K. M. Lee, "Locality-Sensitive Hashing Techniques for Nearest Neighbor Search," *Int. J. of Fuzzy Logic and Intell. Syst.*, vol.12, no.4, pp.300-307, 2012.

[12] A. Bifet and R. Kirkby, *Data Stream Mining : A Practical Approach*, The University of Waikato, 2009.

[13] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[14] K. P. Murphy, *Machine Learning: A Probablisitc Perspective*, 2012.

[15] D. Koller, N. Friedman, *Probabilistic Graphical Model: Principles and Techniques*, The MIT Press, 2009.

[16] _____, Iris Data Set, http://archive.ics.uci.edu/ml/datasets/Iris

[17] P. Cunningham, and S. J. Delany, "k-Nearest Neighbor Classifiers," *Technical Report UCD-CSI-2007-4*, 2007.

[18] F. Kovacs, L. Csaba, and A. Babos. "Cluster validity measurement techniques," *Proc. of 6th International Symposium of Hungarian Researchers on Computational Intelligence*, 2005.

[19] K.M. Lee, C.H. Lee, K.M. Lee, "Statistical cluster validity indexes to consider cohesion and separation," *Proc. of 2012 Int. Conf. on Fuzzy Theory and Its Applications, iFUZZY 2012*, pp. 228-232, 2012.

[20] S.-B. Roh, J.-W. Jeong, T.-C. Ahn, "Fuzzy Learning Vector Quantization based on Fuzzy k-Nearest Neighbor Prototypes," *Int. J. of Fuzzy Logic and Intell. Syst.*, vol.11, no.2, pp.84-88, 2011.

[21] S. Ko, D. Kim, B.-Y. Kang, "A Matrix-Based Genetic Algorithm for Structure Learning of Bayesian Networks," *Int. J. of Fuzzy Logic and Intell. Syst.*, vol.11, no.3, pp.135-142, 2011.

Young Joon Kim is a big data engineer at Cyberdigm, Korea. He received his BS in computer science from Chungbuk National University, and MS in big data science from the same university. His research interest includs big data procesing and data mining.

Keon Myung Lee is a professor at Department of Computer Science, Chungbuk National University, Korea. He received his BS, MS, and Ph.D. degrees in computer science from KAIST, Korea and was a Post-doc fellow in INSA de Lyon, France. He was a visiting professor in University of Colorado at Denver and a visiting scholar in Indiana University, USA. His principal research interests are in data mining, machine learning, soft computing, big data processing, and intelligent service systems.