An Optimistic Data Mining Approach for Handling Large Data Set using Data Partitioning Techniques

Dipak V. Patil
Department of Computer Engineering
Sandip Institute of Technology and Research Centre, Nasik, M.S., India.

R. S. Bichkar
Department of Computer Engineering
G.H. Raisoni College of Engineering and Management, Pune, M.S., India.

ABSTRACT

The use of the Internet for various purposes leads to collection of large volume of data. The knowledge contents of large data can be utilized to improve decision-making process of an organization. The knowledge discovery on this high volume data becomes very slow, as it has to be done serially on currently available terabyte plus data sets. In some cases, mining of large data set may become impossible due to limitations of processor and memory. The proposed algorithm is based on Tim Oates and Davis Jensen's [1] findings which state that increasing size of training data does not considerably increase classification accuracy of a classifier. The proposed algorithm also follows survival of the fittest principal used in genetic algorithm. The solution provides partitioning algorithm wherein decision trees can be learned on partitioned data that are disjoint subsets of a complete data set. These learned decision trees have comparable accuracies with each other and that is equivalent to the tree learned on complete data set. The algorithm finds a single tree with highest accuracy amongst the learned decision trees. The selected decision tree is used for classification of unseen data. The results on 12 benchmark data sets from UCI data repository indicate that the final learned decision tree have equal accuracy and in many cases, significant improvement in classification accuracy is observed, improvement in classification performance as compared to decision trees learned on the entire data set. An experiment on big data set Census-income (KDD) also supports the claim. The most important aspect of this approach is that it is very simple as compared to other methods with enhanced classification performance.

Key Words: - Data partitioning, decision tree, survival of fittest.

1. INTRODUCTION

The volume of data in databases is growing to quite large sizes, both in the number of attributes and instances. Data mining provides tools for discovery of relationships, patterns, and knowledge in databases. Organizations use these databases to inference rules to boost their businesses. Data mining on a very large set of records from a database is quite complex task. The number of data records may overload a computer systems memory and processor making the learning process very slow. Data sets used for inference may be very large, may be up to terabytes. The solution to handle the large training data set is the divide and conquers technique. The proposed technique partitions given data set horizontally into number of non-overlapping subsets, trains classifiers on data and uses, the fittest solution amongst them for data mining task.

There are several proposed classification models over the years like neural networks, decision trees and genetic algorithms [2]-[4]. Among these models, decision trees are predominantly suited for data mining [5] as they are based on simple, powerful, analytical and expressive learning paradigm. The visual presentation makes the decision tree easy to understand and a decision tree represents learned functions or a set of *if-then* rules to improve readability. Decision tree learning methods are fast and accurate and are robust to errors. As a result, the decision tree has become a very popular data mining technique and hence here we have used decision trees for experimentation.

1.1 Decision tree learning

Decision tree algorithms build trees by recursively partitioning training set. A training set consists of set of attributes and a class label. An attribute can have real, Boolean or ordinal values. A decision node states a test to be carried on a particular attribute value of an instance. A branch is present for each probable output of the test. Thus, a tree is traversed from the root to a leaf of the decision tree to identify the class of the instance. The specified class at the leaf is the classification by the decision tree [6], [7]. The generalized decision tree algorithm is explained here.

The tree construction algorithms use a divide and conquer approach to construct a decision tree. It evolves a decision tree for a given training set *T* consisting of set of training instances. An instance denotes values for a set of attributes and a class. Let the classes be denoted by the set of classes $\{C_1, C_2, ..., C_n\}$.

The algorithm works as follows, initially the class frequency is computed for instances in training set *T*. If all instances belong to same class, node *K* with that class is constructed. However, if set *T* contains instances belonging to more than one class, the test for selecting attribute for splitting is executed and the attribute satisfying splitting criteria is chosen for the test at the node. The training set *T* is then partitioned into *k* exclusive subsets are $T_1, T_2, ..., T_k$ on the basis of this test and the algorithm is recursively applied on each nonempty partition. The algorithm for construction of a decision tree is given below.

Construct (*T*)

1. Calculate freq (C_i, *T*).
2. If (all instances belong to same class).
   Return leaf.
3. For every attribute *A* test for splitting criteria.
   Attribute satisfying test is test node *K*.
4. Recur Construct (T_i) on each partition T_i.
   Add those nodes as children of node *K*.
5. Stop.

The C4.5 algorithm uses information gain whereas CART [8] uses gini index as splitting criteria.

A decision trees is called optimal if it correctly classifies the data set and has minimal number of nodes. The decision tree algorithms use local greedy search method by means of information gain as target function to split the data set. Construction of optimal decision tree is identified as NP-Complete problem [9] and hence suggests use of powerful search and optimization technique like genetic algorithms. The genetic algorithm is used to handle combinatorial optimization problems. It has several advantages. It works sound for global optimization problems with the discontinuous objective function or with several local minima. It can work without using auxiliary information such as gradients. Different authors have proposed use of methodologies that integrate genetic algorithms and decision tree learning in order to evolve optimal decision trees. Although the methods are different the goal is to obtain optimal decision trees.

A. Papagelis and D. Kalles [10] proposed GATree, an algorithm for genetically evolving decision trees. The genetic algorithms use binary string as initial populations but GATree uses binary decision trees as initial populations. A binary decision tree that includes one decision node with two different leaves. Initially to construct such initial trees, a random attribute is selected. If that attribute is nominal valued one of its possible values is randomly selected and in case of continuous attributes, an integer value from its minimum to maximum range is randomly selected. Thus, the size of the search space is reduced. Two arbitrary nodes from population of sub-trees are selected and nodes of those sub-trees are swapped to perform crossover operation. In view of the fact that a predicted class value depends just on leaves, the crossover operator does not affect the decision trees consistency. An arbitrary node of a preferred tree is selected and it substitutes the node's test-value with a new arbitrary chosen value to perform mutation. In case if the arbitrary node is a leaf, it substitutes the installed class with a new arbitrary chosen class. Validation is performed after crossover and mutation to get final decision tree. The fitness function for evaluation is percentage of correctly classified instances on the test data set by the decision tree.

The paper is organized as follows. Next section describes related work on handling large data sets in brief. Section 3 presents proposed algorithm. In Section 4 and 5 experimental method and results are presented and finally in Section 6 we summarize our findings.

2. RELATED WORK

The work on handling large data set is done by several researchers. Tim Oates and David Jensen [1], [11] proved that increasing size of training data does not considerably increase classification accuracy of a classifier. It has been found that as numbers of training instances are increased the complexity of the classifier also increases without significant increase in classification performance The hypothesis constructed with large number of training instances of data are often unnecessarily complex and bulky as contrary to the assurance of better parameter estimation provided by large data sets. The authors proposed to build classifiers on data samples.

Hall et al. [12] presented combining decision trees learned in parallel. The proposed algorithm builds decision trees with n disjoint data subsets of a complete data set in parallel, constructs rule set and after that combines them into a single rule set. The experiments on two data sets illustrate that there is enhancement in quantity of rules generated by decision tree. Data partitioning is used to partition data files, the reasons are; the files are too big for single disk or because file access rate cannot be supported by a single disk. Round robin partitioning, Range partitioning and Hash partitioning are the some of available horizontal data partitioning techniques [13]. Round robin is simplest partitioning strategy that divides instances in data partitions in round robin manner.

Hash partitioning technique selects one or more attributes from data set as partitioning attribute and hashing function is applied on them. The function specifies the placement of the data instance in particular partition. Hash function has a range 0 to *n-1*. If hash function returns *i*, the data instance is placed in i^{th} partition. The applications that need barely sequential and associative access to the data are appropriate applications for hash partitioning.

Range partitioning clubs together data instances with similar data values. The example is, country = India and salary > 50K. Range partitioning suffers from problem of data skew. Hashing and Round robin are less vulnerable to the skew problems. The Round robin partitioning method is most suitable method proposed algorithm, as it does not suffer from data skew.

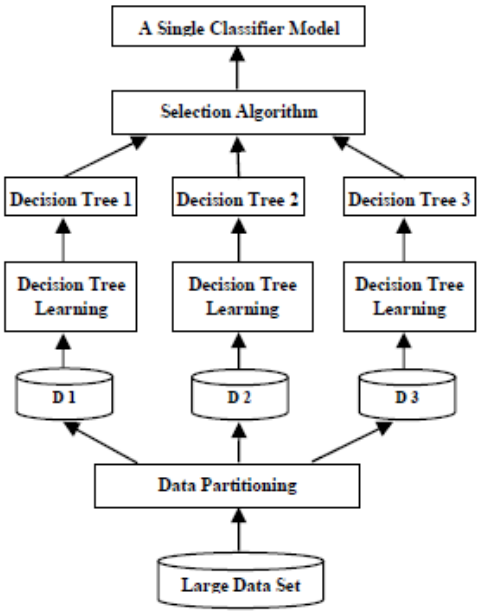

Fig. 1 Proposed data mining algorithm for three data partitions

3. CONSTRUCTION OF DECISION TREES ON PARTITIONED DATA

Let T_f be a full training set containing n training instances. When data set T_f is partitioned, let $\{T_1, \ldots, T_n\}$ be set of instances. Let $|T|$ be a cardinality of the training data set. Let the data set be partitioned in *n* subsets of data where

$T = T_1 \cup T_2 \ldots \cup T_n$ and
$|T_f| = |T_1| + |T_2| \ldots + |T_n|$.

Let the decision tree hypothesis build on T_f be H_f and the hypothesis built on subset *Ti* be *Hi* and the classification accuracy on H_f be $X(T_f)$ and on *Hi* be *X(Ti)*.

In the proposed approach, the algorithm horizontally partitions a large data set into n disjoint sub-sets using round robin method. Round robin partition minimizes the data skew. The training data can be partitioned in several partitions and each partitioned fragment is used to train the decision tree. A decision tree is learned on each of n partitions. The decision

trees are trained on the each partition of data by the various decision tree learning algorithms like J48, CART, GATree etc. Applying principal of survival of fittest, the decision tree with highest classification accuracy is selected for classification. The objective is to have a single decision tree after learning is done on *n* disjoint sub-sets of data. The selection algorithm selects suitable decision tree having highest classification accuracy of decision tree as a function to select decision trees for classification. The resulting hypothesis can be used to classify unseen examples.

The proposed partitioning based optimistic data mining algorithm illustrated in Fig. 1.

3.1 Optimistic data mining Algorithm

Optimistic data mining algorithm considers that nothing will go wrong and unseen instances will be classified with selected fittest classifier.

1. Partition the data horizontally using round robin partitioning.
2. Each partition is trained with data mining algorithm.
3. Use classification accuracy as a fitness function to select fittest decision trees for classification.
4. The fittest system can be used to classify unseen examples.
5. Calculate classification accuracy for unseen data on the same.

4. EXPERIMENTAL METHOD

Experiments were performed on 12 data sets from UCI, repository [14]. In test method, the first step is data partitioning. The data-partitioning tool partitions data into equal non-overlapping *n* sub-sets using round robin partitioning method. The experiments were performed to obtain tree classification accuracy of the trees on three disjoints subsets of data. We have used three decision tree classifiers namely, GATree [9], J48 and CART [15]. The classification accuracy was obtained using five fold cross validation method. Experiments were also performed with complete non-partitioned data set T_f to calculate 5 fold cross-validated classification accuracy. The default parameter settings were used for J48. In case of GATree, the parameters were set as follows: crossover rate = 0.99, mutation rate = 0.01, stopping criterion = 100 generations.

Table1. Accuracy on GATree

Data set	$X(T_f)$	$X(T_1)$	$X(T_2)$	$X(T_3)$
Australian	85.36	86.52	85.22	82.61
Breast-w	95.80	92.17	96.53	97.40
Credit	85.21	82.61	86.52	85.78
Diabetes	73.73	73.73	73.51	73.73
Heart	75.56	76.66	75.56	81.11
Kr-Vs- Kp	92.51	90.58	91.46	87.04
Lymph	77.24	68.00	66.66	77.55
Monks	42.50	65	57.5	75.00
Mushroom	95.82	97.19	82.51	89.13
Vote	95.63	93.80	96.56	93.10
Waveform	65.32	63.60	66.55	65.28
WDBC	90.97	92.11	87.37	90.27

Table 2. Accuracy on Census-income (KDD) data

Accuracy	Classifier	
	GATree	J48
$X(T_f)$	93.70	95.39
$X(T_1)$	94.08	95.35
$X(T_2)$	94.16	95.35
$X(T_3)$	94.20	95.31

Table 3. Comparison accuracy

Sr. No.	Data set	GATree			J48			CART		
		X_f	X_O	ΔX	X_f	X_O	ΔX	X_f	X_O	ΔX
1	Australian	85.36	86.52	1.16	85.51	85.22	-0.29	84.35	86.09	1.74
2	Breast-w	95.80	97.40	1.60	95.28	96.57	1.29	94.42	97.42	3.00
3	Credit	85.21	86.52	1.31	85.94	87.34	1.40	85.07	85.59	0.52
4	Diabetes	73.73	73.73	0.00	74.09	77.73	3.64	73.56	73.44	-0.12
5	Heart	75.56	81.11	5.55	77.78	76.49	-1.29	77.41	83.33	5.92
6	Kr-Vs-Kp	92.51	91.46	-1.05	99.53	98.50	-1.03	99.34	98.22	-1.12
7	*Lymph*	77.24	77.55	0.31	77.03	80.00	2.97	79.73	78.90	-0.83
8	Monks	42.50	75.00	32.50	44.35	75.61	31.26	53.22	75.61	22.39
9	Mushroom	95.82	97.19	1.37	100.00	100.00	0.00	99.94	99.78	-0.16
10	Vote	95.63	96.56	0.93	96.78	95.86	-0.92	95.63	97.24	1.61
11	Waveform	65.32	66.55	1.23	75.52	75.99	0.47	76.82	76.81	-0.01
12	WDBC	90.97	92.11	1.14	93.85	99.47	5.62	92.79	93.12	0.33
	Average	**81.30**	**85.14**	**3.84**	**83.81**	**87.40**	**3.59**	**84.36**	**87.13**	**2.77**

Table 4. Comparison accuracy on census-income (KDD) data

Sr. No.	GATree			J48		
	X_f	X_O	ΔX	X_f	X_O	ΔX
1	93.70	93.91	0.21	95.39	95.35	-0.04

Fig. 2 Comparison accuracy GATree

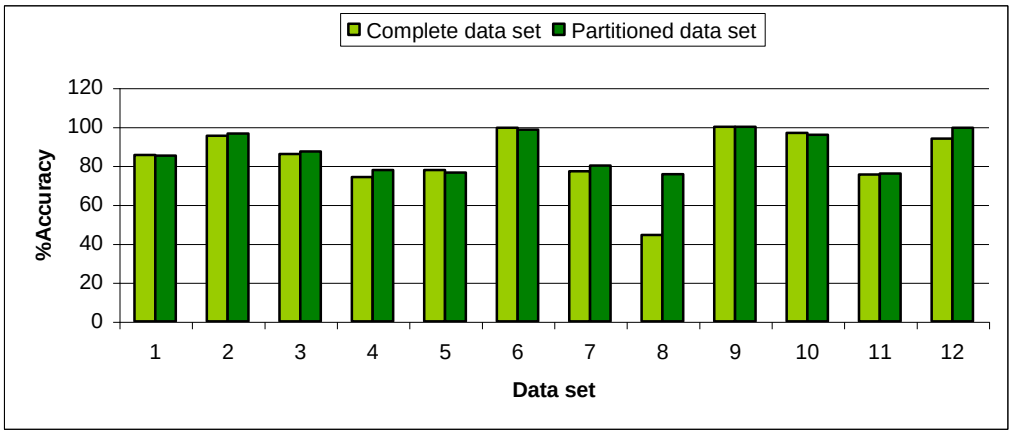

Fig. 3 Comparison accuracy J48 WEKA

Fig. 4 Comparison accuracy CART WEKA

5. RESULTS

Table 1 presents classification accuracy of decision trees obtained on partitioned data and on complete data set using classifier GATree. The decision trees learned on partitioned data are as accurate as on the complete data set. Using these results, we can select the most accurate classifier for each data set. For example, for Australian data set, decision tree obtained on partition 1 is selected as it provides higher accuracy (i.e. 86.52%). Similar results were obtained using two other decision tree algorithms namely J48 and CART. Tables 3 and 4 presents summary of results where X_f is accuracy with complete data set T_f, X_o is accuracy with optimistic data mining algorithm and ΔX is improvement in accuracy X_o with respect to X_f.

In Table 3 the average accuracy obtained by proposed algorithm, using GATree it is 85.14% as compared to average accuracy of 81.30% on complete data set by conventional method. Similarly, tests on J48 provide average accuracy of

87.40% as compared to 83.81% using complete data set and tests on CART provide average accuracy 87.13 % as compared 84.36%. Thus the average improvements in accuracy are 3.84%, 3.59% and 2.77% on GATree, J48 and CART respectively.

The maximum improvement in accuracy is obtained in case of Monks- problem data set, the improvement in accuracies are 32.5%, 31.26% and 22.39% on GATree, J48 and CART respectively. However, in some data sets (Heart-Statlog data set and Kr-Vs-Kp data set) there is slight loss of accuracy of around 1%.

The validation of proposed method for large set is done with Census–income data set having 299285 instances.

The results are presented in Table 2 and 4. The classification accuracy on both the classifiers is equivalent and slight enhancement is observed in case of GATree classifier, whereas we could not get results on CART due to memory limitation. Despite of 3.5 GB JM memory provided to CART, it displayed out of memory error.

6. CONCLUSIONS

The proposed algorithm enables handling of large data set by overcoming limitations of memory and processor capacity. The classification performance of proposed method is equivalent to the performance on complete data set. Horizontal decomposing the data sets into, disjoint subsets avoids the problem of running out of memory particularly with large data set. The data sets can be partitioned into a size that can be efficiently managed on available memory and processor. It is expected that classification accuracy on complete data set should be higher than that obtained on partioned data set; contrary to it the proposed method presents enhanced classification performance in several cases. The enhancement in classification of performance is probably due to reduction in outliers in data sets because of data partitioning. The algorithm is very simple to implement and is better than the previous approaches proposed to handle large data sets [1], [12].

7. REFERENCES

[1] Tim Oates, David Jensen (1997). "The Effect of Training Set Size On Decision Tree Complexity", Proc. 14th International Conference on Machine Learning.1997.pp. 254-262.

[2] Tim Mitchell, (1997). *Machine Learning*, The McGraw-Hill Companies, Inc.

[3] S. Rajasekaran, G.A. Vijayalakshmi Pai (2004). Neural Networks, Fuzzy Logic and Genetic Algorithms Synthesis and Applications. Prentice-Hall of India.

[4] D. E. Goldberg, (1999). Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley.

[5] M. Mehta, R. Agrawal and J. Rissanen (1996). SLIQ: A fast scalable classifier for data mining. *Proc. of the Fifth International Conference on Extending Database Technology*, Avignon, France. pp. 18-32.

[6] J. R. Quinlan, (1993). *C4.5: Programming for Machine Learning*. San Francisco, CA: Morgan Kaufman.

[7] S. Ruggieri (2002). Efficient C4.5. *IEEE Transaction On Knowledge and Data Engineering*, Vol. 14, No. 2, pp. 438-444.

[8] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone (1984). *Classification and Regression Trees.* Wadsworth International Group, Belmont, CA.

[9] Murthy S. K (1998). Automatic construction of decision trees from data: A multidisciplinary survey. *Data Mining and Knowledge Discovery*, Vol. 2, No. 4, pp. 345-389.

[10] A. Papagelis and D. Kalles (2000). GATree: Genetically evolved decision trees. *Proc. 12th International Conference On Tools With Artificial Intelligence*, pp. 203-206.

[11] Tim Oates and David Jensen (1998). Large Datasets Lead to Overly Complex Models: An Explanation and a Solution., Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining. August 1998.

[12] L. O. Hall, N. Chawla and K. Bowyer (1998). Combining decision trees learned in parallel. *Distributed Data Mining Workshop at International Conference of Knowledge Discovery and Data Mining*. pp. 77-83.

[13] Chhanda Ray (2009). *Distributed database systems*, Pearson Education India pp. 26-29.

[14] Frank, A. and Asuncion, A. (2010). UCI Machine Learning Repository Irvine, CA. [http://archive.ics.uci.edu/ml]. University of California, School of Information and Computer Science.

[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.