Associative Classification Based on Incremental Mining (ACIM)

Mohammed H. Alnababteh, M. Alfyoumi, A. Aljumah, and J. Ababneh

***Abstract*—Associative classification (AC) is an approach in data mining that uses association rule to build classification systems that are easy to interpret by end-user. When different data operations (adding, deleting, updating) are applied against certain training data set, the majority of current AC algorithms must scan the complete training dataset again to update the results (classifier) in order to reflect change caused by such operations. This paper deals with data insertion issue within the incremental learning in AC mining. Particularly, we modified a known AC algorithm called CBA to treat one aspect of the incremental data problem which is data insertion. The new algorithm called Associative Classification based on Incremental Mining (ACIM). Experimental results against six data sets from UCI data repository showed that the proposed incremental algorithm reduces the computational time if compared to CBA, and almost derives the same accuracy of it.**

***Index Terms*—Associative classification, CBA, data mining, Incremental mining.**

I. Introduction

Recent developments of information technology and computer networks caused the production of large quantities of databases. These databases normally contain hidden useful information that can be utilized in decision making and corporate planning. Therefore, efficiently finding and managing the useful information from these large databases become a necessity. One of the common tools which discovers and extracts non obvious knowledge from different types of data is data mining. The importance of data mining is growing rapidly in recent years since it can be used for several different tasks including classification, clustering, regression, association rule discovery, and outlier analysis [1].

Incremental learning is one of the challenges related to data mining tasks especially association rule and classification. In classification context, this problem involves updating the classification model (classifier) whenever the training data collection gets updated. In most real world applications like stock market exchange, online transaction, retail marketing, and banking, data usually are updated on a daily basis, and therefore handling the incremental learning problem becomes crucial in these applications.

Manuscript received September 25, 2013; revised November 10, 2013. This work was supported by the deanship of scientific research at Salman bin Abdulaziz University under the research project # 1/ت/1433.

M. H. Alnababteh, M. Alfyoumi, and A. Aljumah are with the Salman Bin Abdulaziz University, Kharj, KSA (e-mail: nababteh@gmail.com, fayomi66@yahoo.com, aljumah88@hotmail.com).

J. Ababneh is with the World Islamic Sciences and Education University, Amman, Jordan (e-mail: ababnehjafar@yahoo.com, jafar.ababneh@wise.edu.jo).

In association rule discovery several incremental algorithms have been developed such as Fast Update (FUP) [2], FUP2 [3], Insertion, Deletion and Updating [4], Galois Lattice theory [5], and New Fast Update (NFUP) [6]. However, in classification data mining especially associative classification [7] and rule induction [8], scholars have paid little attention to the incremental learning issue. Further, since classification is a common task in data mining and has a great number of important applications in which data are often collected by these applications on daily, weekly or monthly there is a great interest to develop or at least enhance the current classification methods to handle the incremental learning problem. This is the primary motivation of our algorithm.

In the last few years an approach that integrates association rule and classification called Associative Classification (AC) was proposed [7]. Several research studies, i.e. [7], [9], [10] provide evidence that AC often successfully builds more accurate classifiers than traditional classification approaches such as rule induction and decision trees. Furthermore, many application domains including image analysis [11] and document classification [12] have adopted AC since it generates simple "IF-THEN" rules that are easy to interpret by end-user.

One of the well-known AC algorithms is CBA [7], which operates in two phases where in phase one it discovers candidate class association rules (CARs). In the second phase CBA selects the highest accurate rules from the complete set of CARs to represent the classifier. However, what if the training data which the CARs have been derived from, is updated.

In AC algorithms, when new dataset added to original dataset, we can deal with it in two ways, first use the current classifier without any consideration to incremental dataset (d^+) to classify unseen data, thus the classifier will not reflect the changes on the dataset, consequently the accuracy of classifier will be decreased, because a lot of previous CARs (class association rule) of classifier become invalid after dataset update, second rebuild the classifier from scratch depending on the original dataset (D) and the incremental dataset (d^+) (i.e. D U d^+), but this approach require a complete scan to the whole training data set in order to reflect the new changes on the classifier. This means regenerating most of the CARs that were produced in the previous scan. Now, since new rules are generated and existed rules may be discarded after a data operation is executed against the training data set, which definitely causes time overhead [13].

This paper, presents a new AC based on incremental mining algorithm called **ACIM** it based on CBA algorithm to

deal only with row insertion aspect of the incremental learning.

It is the firm belief of the authors that all current AC algorithms do not consider the problem of incremental learning into consideration. Thus, when the original data set gets updated these algorithms do not reflect the changes done on the classifier, consequently, the classification accuracy may negatively be impacted since the complete set of knowledge in the classifier are not represented within the input data set.

The rest of the paper is organized as follows: Section II describes the framework of the proposed Associative Classification based on Incremental Mining (**ACIM**), and the algorithm details. The experimental results are demonstrated in Section III.

II. The Developed Algorithm ACIM Description

Fig. 1 depicts the developed AC based on incremental mining (**ACIM**) algorithm which work as the following:

The first classifier is built from the original database *D*; when increment data d_i added to the original dataset, the new classifier builds using the incremental data and original data *D* by utilizing the classifier which built in previous phase.

The proposed algorithm extract all strong rules (CARs) which satisfy minimum support and minimum confidence threshold of the increment data (d_i) where the minimum support and confidence threshold of d_i is equal to original minimum support and confidence , the new CARs which generated from the d_i are match with the current classifier to update the support and confidence of the CARs that exist in the current classifier and frequent in the d_i , then remove any CAR if same or general rule exist in the classifier.

The CARs that generated from d_i but not exist in the classifier are check against original database then remove all CARS which don't satisfy minimum support and minimum confidence.

The rules of the classifier that not frequent in the incremental data will check against incremental data d_i to update their supports and confidences; the CARs which already exist in the classifier but don't satisfy minimum support and confidence after database update will not remove from the classifier because they will use for prediction, that will explain later in this section. The CARs in the classifier and the generated frequent CARS from d_i that satisfies minsup and minconf will merge together as in Fig. 2. The whole previous rules will take a weight as follow:

1) The CARs that were exist in the classifier and generated as frequent from d_i will take very high weight.
2) The CARs which is frequent in d^+ and satisfy minsup and minconf against whole database but not exist in current classifier will take high weight.
3) The CARs that were exist in the classifier and infrequent in incremental data d^+ but still frequent in the entire database will take mid weight.
4) The CARs that were exist in the classifier but become infrequent in the whole database will take low weight.

The whole remaining CARs will rearrange according to following criteria:

If there are two rules r_1 and r_2, then $r_1 > r_2$ if and only if:

1) If weight of the r_1 is greater than the weight of r_2
2) If the weight of r_1 and r_2 is equal but the confidence of r_1 is greater than confidence of r_2.
3) If the weight and confidence of r_1 and r_2 are equal but the support of r_1 greater than the support of r_2.
4) If the weight, confidence and support are equal of r_1 and r_2, but r_1 has minimum attributes in the left-hand side than r_2.

After rule generation from d_i and amalgamation with classifier rules, we must have a relatively large number of rules and accordingly there are a lot of redundant rules that impact negatively on the efficiency and accuracy of the final classifier. We cannot preserve all the rules that generated in previous step.

In **ACIM** algorithm the database coverage pruning method used as in Fig. 2 to test all rules against whole database. The database coverage pruning technique finds all the training data case that fully match by any rule, which is ranked in previous step. First all rules cover weight is set 0 (where the cover weight is weight indicate to the number of times the rule entry to level I or level II) If at least one training case match the rule, It rule will insert to the end of level I classifier and its cover weight (*coverw)* increased by 1 and all the cases (tuples) in the training data discard, else the rule insert into level II classifier and its cover weight decreased by 1 .

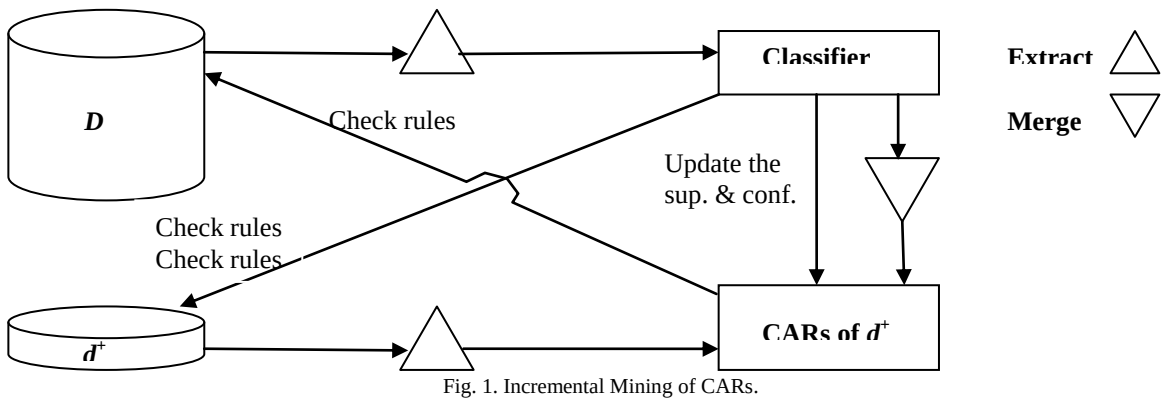

Fig. 1. Incremental Mining of CARs.

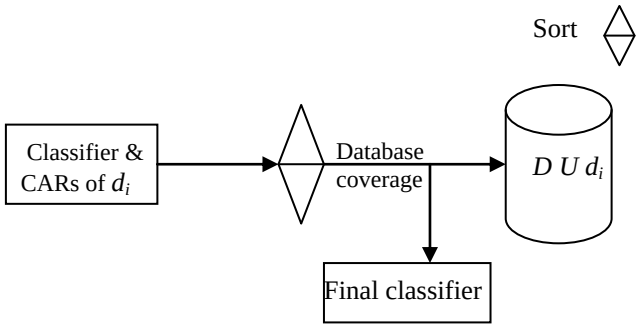

Fig. 2. Generate the classifier in ACIM.

Our algorithm divides the classifier into two levels, where the first level contains the rules which cover at least one case in the training dataset, the level II contain the rules that didn't cover any training data set.

The rules in the level II almost have weight less than level I rules. Our algorithm try to control the number of rules that exist in level I and level II (i.e. the final classifier) by remove all the rules the don't satisfy the minimum cover weight threshold δ.

The pseudo code in Fig. 3 demonstrates how the **ACIM** build a new classifier in which the notations used is explained in Table I.

TABLE I: METHODS AND NOTATION OF THE PROPOSED ALGORITHM

Notation	**Detail**
I	Indicator of the incremental data set number
D	Original data set
d_i	Incremental data set
Methods	**Detail**
Buildclassifier_CBA()	Build classifier using CBA algorithm and testing.
classifier_ testing()	Test the rule

Inputs: Original Database (D), minsup, minconf, weight threshold δ, weight constant.

1. *Output: new classifier.*
2. *// build the classifier using original data*
3. *buildclassifier_CBA(D)*
4. *do until (no incremental data){*
5. *CARs= CBA_RG(d_i)*
6. *For each rule r Є CAR_{di} do*
7. *{*
8. *For each rule R Є classifier do*
9. *// if any rule R in classifier is general or same to r*
10. *// then the support and confidence of R is update based on r*
11. *//and the weight is assign to R, also r will remove from CAR_{di} .*
12. *// all of that encapsulate in function match*
13. *Match(r, R);*
14. *}*
15. *// check remaining rule in CAR_{di} against original dataset D*
16. *For each r Є CAR_{di} do*
17. *For each row Є D do*
18. *{*
19. *// update the support ,confidence and weight*
20. *// for the rule that not exist in the classifier*
21. *if row satisfies the conditions of r then*
22. *{*
23. *Update_supp(r);*
24. *Update_weight(r);*
25. *}*
26. *}*
27. *// check the rules which exist in the classifier but infrequent in d_i*
28. *For each R Є classifier do*
29. *For each row Є d_i do*
30. *{*
31. *// update the support, confidence and weight for all R*
32. *// for rules which exist in the classifier but infrequent*
33. *// in CAR_{di}*
34. *Update_supp(R);*
35. *Update_weight(R);*
36. *}*
37. *// rearrange all rules*
38. *NCAR=Sort(R U r);*
39. *For each rule r Є NCAR do*
40. *{*
41. *for each row Є D do*
42. *{*
43. *if row satisfies the conditions of r then*
44. *{*
45. *store d.id in array1 and store r in array2 if it correctly classifies d;*
46. *}*
47. *if r is In array2 then*
48. *{*
49. *insert r at the level I classifier C and increased its cover weight;*
50. *delete all the rows which has id in array1 from D;*
51. *}*
52. *Else*
53. *{*
54. *insert r at the level II classifier C and decreased its cover weight;*
55. *}*
56. *}}*
57. *// the default class is the major class in level I and level II*
58. *selecting a default class for the current classifier C;*
59. *// remove all rules that don't satisfy minimumcover weight*
60. *// threshold from level II*
61. *remove all rules don't satisfy minicoverwieght δ;*
62. *}*

Fig. 3. Building classifier in ACIM.

According to Fig. 3. In line 3 the classifier of the original training dataset is built. In line 5, the CARs of the first part of the incremental data generated. in line 6-14, the rules in the classifier and the generated rule in line 5 are matched, if any rule *R* in classifier is general or same to *r* then the support and confidence of *R* is update based on *r* and the weight is assign to *R*, also *r* will remove from CARdi. All of that encapsulate in function **match**. In line 16-26, the remaining rules of the generated rules in line 5 (CARdi) is checked against original dataset D to update the support, confidence and weight for the rule that not exist in the classifier. In line

28-36 the algorithm check the rules *R* which exist in the classifier but infrequent in d_i and update the support, confidence and weight for all rules which exist in the classifier but infrequent in CARdi.

In line 38, the algorithm rearranges all rules according the weight, support and confidence and stores the final rules in NCAR. In line 39-62, the database coverage is applied on the NCAR against entire dataset (D U di). The final rules stored in the classifier, where the rule which match at least one object in dataset enter to level I and the rule does not satisfy any object insert to level II else the rule remove from the final classifier. At the end of process the default class is selected and all the rules that did not satisfy minicoverwieght δ will be removed.

When new unseen object give the algorithm search in level I of the first rule which has antecedent (left-hand side) match the give object and assign its class to this object, if no rules match in level I the proposed algorithm seek in level II, if no rule also match the given object the default class will assign to given object.

III. Experimental Results

performance study have been conducted to evaluate the classification accuracy and the efficiency for the proposed associative classification based on incremental mining algorithm **ACIM** in on hand, on the other hand the classic CBA algorithm. The proposed method adopts CBA algorithm for rule generation, pruning and prediction procedures to build the first classifier i.e. from original dataset D. Different data sets from UCI machine learning repository [14] are used during the experimentation. These are "Car eval. ", "led7", "pageblocks ", "pen digits ", "waveform ", "wine qu.". The selection of these data sets is based on the size and the data quality since we have looked for medium to large size training data set because we divide the dataset into original and incremental data, also the above data sets don't contain missing values. Explanation of the characteristics of each dataset can be found in [14]. Table II shows the characteristics of the using datasets.

TABLE II: UCI Datasets Characteristics

Data set	Number of attributes	Number of records	Class no.
Car evaluation	6	1728	4
Page block	10	5473	5
Pen digit	16	10992	10
Wav form	21	5000	3
Wine quality	12	4898	7
Led7	7	3200	10

Each UCI data set has been divided into two partitions one for training and one for the incremental learning process (data to be inserted). Then, we further divided the incremental data set into five data blocks for testing purposes and particularly to perform multiple data insertions on the training data set during training and evaluation steps.

All experiments were performed on Celeron 2.16 a computing machine with the following specifications: 1GB main memory, and running Microsoft windows XP. The LUCS-KDD implementation of the CBA algorithm is used [15]. The main thresholds that control the number of rules generated and prediction accuracy in AC mining are minimum support and minimum confidence, and for both the proposed algorithm (**ACIM**) and CBA, these thresholds have been set to 1% and 50%, respectively. This setting is similar to other previous research conducted in AC, i.e. [16], [7].
Our experimental study on the developed algorithm is study the effect of the following issue on the performance of the developed algorithm (**ACIM**) against **CBA** algorithm:

1) Mining the frequent rules from incremental data.
2) Update the current classifier through incremental data.
3) Merge classifier rules with frequent rules from incremental data.
4) Rule ranking.
5) Rule pruning.
6) Prediction techniques.

In the experiments **ACIM** algorithm which shown in Fig. 3, the training dataset divided into two parts one for original training data and one for incremental data, the incremental data divided into five parts.

The implementation performed on five incremental data, thus the second part of the training data that dedicated for incremental data will be divided into five parts.

In the experiments, The first classifier built from original training data (i.e. first part of training data), then the accuracy of this classifier after applying TCV (Ten-Cross Validation) stored. After that new incremental data (first part of incremental data-1/5) add to original data. The developed algorithm implemented and the accuracy stored. At the end of this step the algorithm add the incremental data to the original data and consider both as original data in the second step. The above process repeat on the five incremental training data and the average accuracy of the classifier on the five steps will be computed.

Table III and Table IV show the average accuracy and average runtime for CBA and our algorithm respectively. Where, the minimum cover threshold is set to 0 to disable it in our implementation. The weight for very high, high, mid and low rules is set 4,3,2,1 respectively.

Table III shows the average accuracy of CBA and the **ACIM**. We can see from Table III our method achieve better accuracy than CBA in three datasets that's refer to reduce the dependency on default class because our method go to level II classifier if no rules match the unseen object .

TABLE III: Average Accuracy of CBA and ACIM Algorithm

Data set	CBA Average Accuracy	ACIM Average accuracy
Car evaluation	90.90	**92.1**
Led7	71.90	71.62
Page blocks	93.68	92.18
Pen digits	91.30	**92.42**
Waveform	75.60	**76.23**
Wine quality	44.09	44.06

Table IV shows the runtime of CBA and **ACIM**. The

ACIM algorithm outperform on CBA in all dataset and that's make sense because our method doesn't build the new classifier from scratch, the developed method utilize the previous classifier to build new one. The developed method generates all candidate itemsets for incremental data only.

Fig. 4 and Fig. 5 show the average accuracy and average runtime of CBA and **ACIM** graphically.

TABLE IV: AVERAGE RUNNING TIME FOR CBA AND ACIM ALGORITHM

Data set	CBA Avg. runtime(s) in seconds	Our method Avg. runtime(s) in seconds
Car evaluation	.26	**.21**
Led7	.36	**.23**
Page blocks	1.15	**.85**
Pen digits	145.0	**35.16**
Waveform	140.4	**45.21**
Wine quality	3.55	**2.01**

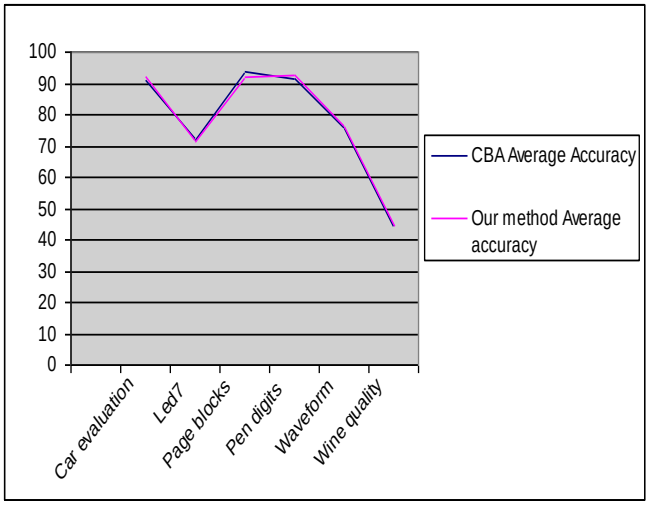

Fig. 4. Average accuracy of CBA and ACIM.

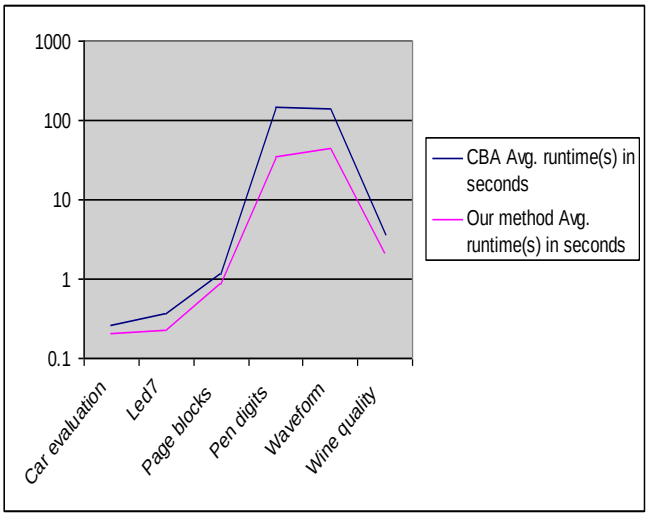

Fig. 5. Average runtime of CBA and ACIM.

IV. SUMMARY

In this paper we introduced associative classification based on incremental mining algorithm called **ACIM**. This algorithm deals with data insertion problem in associative classification context. The method uses new technique to build the classifier when nontrivial data insertion operation occur on the original dataset which used to build the old classifier before update. The experimental results against six UCI data collection showed that our method reduces the computational time for all data sets comparing with that of CBA, and almost gives the same accuracy of it.

ACKNOWLEDGMENT

This research was supported by the deanship of scientific research at Salman bin Abdulaziz University under the research project # 1/ت/1433.

REFERENCES

[1] U. Fayyad, G. Piatetsky-Shapiro, G. Smith and R. Uthurusamy, *Advances in knowledge discovery and data mining*. Cambridge, Mass.: AAAI Press/The MIT Press, 1996, ch. 4.

[2] D. Cheung, J. Han, V. Ng, and C. Wong, "Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique," in *Proc. 12th the International Conference on Data Engineering*, New Orleans, 1996, pp.106-114.

[3] D. Cheung, S. Lee, and B. Kao, "A general Incremental Technique for Mining Discovered Association Rules," *in Proc. 5th International Conference on Database System for Advanced Applications*, Melbourne, 1997, pp. 185-194

[4] P. Tsai, C. Lee and A. Chen, "An efficient approach for incremental association rule mining," in *Proc. 3rd Pacific-Asia Conference on Methodologies for Knowledge Discovery and Data Mining*, London, 1999, pp. 74-83.

[5] V. Petko, M. Rokia, R. Mohamed, and G. Hacene Robert (2003). "Incremental Maintenance of Association Rule Bases," in *Proc. 2nd Intl. Workshop on Data Mining and Discrete Mathematics*, San Francisco, 2003, 12 p.

[6] C. Chang, Y. Li, and J. Lee, "An Efficient Algorithm for Incremental Mining of Association Rules," in *Proc. 15th International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications*, Washington, 2005, pp.3-10.

[7] B. Liu, W. Hsu, and Y. Ma, "Integrating classification and association rule mining," in *Proc. 4th International Conference on Knowledge Discovery and Data Mining*, New York, 1998, pp. 80-86.

[8] J. Quinlan and R. Cameron-Jones, "FOIL: A midterm report," in *Proc. 1993 European Conference on Machine Learning*, Vienna, 1993, pp. 3-20.

[9] F. Thabtah, P. Cowling, and Y. Peng, "MCAR: Multi-class classification based on association rule approach," in *Proc. 3rd IEEE International Conference on Computer Systems and Applications*, Cairo, 2005, pp. 1-7.

[10] X. Yin and J. Han, "CPAR: Classification based on predictive association rule," in *Proc. 2003 Siam International Conference on Data Mining*, San Francisco, 2003, pp. 369-376.

[11] M. Antonie, O. Za ïane, and A. Coman, "Associative classifiers for medical images," *Lecture Notes in Artificial Intelligence 2797, Mining Multimedia and Complex Data*, Springer-Verlag, 2003, pp. 68-83,

[12] Y. Yoon and G. Lee, "Practical application of associative classifier for document classification," in *Proc. 2nd Asia Information Retrieval Symposium*, Jeju-island, Korea, 2005, pp. 467-478.

[13] M. Nababteh, R. Al-Shalabi, F. Thabtah and M. Najeeb, "An Incremental Data Insertion Algorithm for Associative Classification Mining," in *Proc. 15th International Business Information Management Conference on Knowledge Management and Innovation: A Business Competitive Edge Perspective*, Cairo, 2010, pp. 1806-1812.

[14] C. Merz and P. Murphy, "UCI repository of machine learning databases," Irvine, CA, University of California, Department of Information and Computer Science, 1996.

[15] F. Coenen. LUCS_KDD implementation of CBA. (2004). Department of Computer Science, the University of Liverpool. [Online]. Available: http://www.csc.liv.ac.uk/~frans/KDD/Software/CBA/cba.html

[16] W. Li, J. Han and J. Pei, "CMAR: Accurate and efficient classification based on multiple-class association rule," in *Proc. 2011 International Conference on Data Mining*, San Jose, 2001, pp. 369-376.

M. H. Alnababteh received the B.S. Degree in Computer Sciences from Philadelphia University, Jordan in 2001 and M.S. Degree in Computer Information System from AABFS University, Jordan in 2005 and Doctor of Philosophy in computer information system from AABFS University, Jordan in 2011. From 2002 to 2005, he worked as a programmer in Web Development and Engineering company, Jordan. He was working as a lecturer in Amman Training College during 2006-2011. He was an assistant professor in software Engineering Department at Jadara University, Jordan in 2011. Currently he is an assistant professor in Computer Information System at Salman bin Abdulaziz University, KSA. His current research focuses on data mining, information retrieval and cloud computing.

M. Al-Fayoumi received the B.S. Degree in Computer Science from Yarmouk University, Irbid, Jordan in 1988. He received the M.S. degree in computer science from the University of Jordan, Amman, Jordan in 2003. In 2009, he received a Ph.D. degree in computer science from the Faculty of Science and Technology at Anglia University, UK. In 2009, he joined the Al-Zaytoonah University, in Jordan, as an assistant professor. Currently, he is an assistant professor and chairman of computer science department at Salman bin Abdul Aziz University, Saudi Arabia. His research interests include areas like computer security, cryptography, identification and authentication, wireless and mobile networks security, e-application security, simulation and modeling, algorithm analyzes and design, information retrieval, data mining and any other topics related to them. He has published more than 10 research papers in international journals and conferences.

Abdullah Aljumah received Ph.D. in Electronic Engineering from University of Wales, Cardiff, UK in 1999. His main area of research is in artificial intelligence, digital design, and data mining. He is currently working as an associate professor as well as dean of the College of Computer Engineering and Sciences, Salman Bin Abdulaziz University, Alkharj, Saudi Arabia. He is also a consultant for several Government Organizations and a member of councils of various boards and commissions. He has published a number of research papers in reputed conferences and journals.

J. Ababneh is an assistant professor. He received his Ph.D. Degree from Arab Academy for Banking & Financial Sciences, Jordan in 2009. He received his M.Sc degree in computer engineering from University of the Yarmouk, Jordan in 2005. He earned his B.Sc. in Telecommunication engineering from University of Mu'ta (Jordan) in 1991. He joined in 2009 the World Islamic Sciences and Education (WISE) University as a head of the departments of computer information systems and network systems in the school of information technology beside assistance dean of information technology faculty. He has published many research papers, book chapters, and books in different fields of science in refereed journal and international conference proceedings.

His field research lies in development and performance evaluation of multi-queue nodes queuing systems for congestion avoidance at network routers using discrete and continuous time, also his research interests includes computer networks design and architecture, wire and wireless communication, artificial intelligence and expert system, knowledge base systems, security systems, data mining and information retrieval.