LOOKING AT THE CLASS ASSSOCIATIVE CLASSIFICATION ALGORITHM

Q. Mahmood[1], F. Thabtah[1] and L.McCluskey[1]
[1] University of Huddersfield, Queensgate, Huddersfield HD1 3DH, UK

ABSTRACT

Associative classification (AC) is a branch in data mining that utilises association rule discovery methods in classification problems. In this paper, we propose a new training method called Looking at the Class (LC), which can be adapted by any rule-based AC algorithm. Unlike the traditional Classification based on Association rule (CBA) training method, which joins disjoint itemsets regardless of their class labels, our method joins only itemsets with similar class labels during the training phase. This prevents the accumulation of too many unnecessary mergings during the learning step, and consequently results in huge saving in computational time and memory.

Keywords: Associative Classification, Classification, Data Mining, Itemset, Training Phase

1. INTRODUCTION

Information science is developing very rapidly and resulting in a significant increase in data warehousing. The computer hardware storage capabilities have grown by leaps and bounds in recent years, which have contributed to the large amount of information storage in almost all fields of life. Due to the wide variety of data being captured, efficient management and quick retrieval of information is very important for decision making. Data mining is the science of extracting meaningful information from these large data warehouses (Witten and Frank, 2000). Data mining and knowledge discovery techniques have been applied to several areas including, market analysis, industrial retail, decision support and financial analysis. Knowledge Discovery from Databases (KDD) (Fayyad, et al., 1998) involves data mining as one of its main phases to discover useful patterns. Other phases in KDD are data selection, data cleansing, data reduction, pattern evaluation and visualisation of discovered information (Elmasri and Navathe, 1999).

Two common data mining tasks are classification rule mining and association rule mining (ARM). The task of ARM can be defined according to (Agrawal, 1993) as follows: Let D be a database of sales transactions, and $I = \{i_1, i_2, \ldots, i_m\}$ be a set of binary literals called items. A transaction T in D contains a set of non empty items called an itemset, such that $T \subseteq I$.

Definition: The support of an itemset is defined as the proportion of transactions in D that contain that itemset.

Definition: An association rule is an expression $X \rightarrow Y$, where X, $Y \subseteq I$ and $X \cap Y = \theta$.

Definition: The confidence of an association rule is defined as the probability that a transaction contains Y given that it contains X, and given as support ($X \cup Y$)/support(X).

Given a transactional database D, the association rule problem is to find all rules that have supports and confidences greater than certain user-specified thresholds, denoted by minimum support (MinSupp) and minimum confidence (MinConf), respectively. In ARM, the ultimate aim is the discovery of the most significant associations between the items in a transactional data set (Agrawal and Srikant, 1994). This process involves primarily the discovery of so called frequent itemsets, i.e. itemsets that occurred in the transactional data set above MinSupp and MinConf. The discovered association rules represent useful information presented in the transactional data set that relates to item relationships and trends. These rules are very useful and can help in making necessary planning decisions such as item shelving (Agarwal et al., 1993).

The main difference between classification and ARM is the outcome of the rules generated. In case of classification, the outcome is pre-determined, i.e. the class attribute. Classification also tends to discover only a small set of rules in order to build a model (classifier), which is then used to forecast the class labels of previously unseen data sets as accurately as possible. On the other hand, the main goal of ARM is to discover correlations between items in a transactional data set. In other words, the search for rules in classification is directed to the class attribute, whereas, the search for association rules are not directed to any specific attribute.

AC is a branch in data mining that combine's classification and association rule mining. In other words, it utilises association rule discovery methods in classification data sets. Many AC algorithms have been proposed in the last few years, i.e. (Liu et al., 1998, Li et al., 2001, Thabtah et al., 2004, Thabtah, et al., 2005), and produced highly competitive experimental results with respect to classification accuracy if compared with that of traditional classification approaches such as decision trees (Quinlan, 1993), probabilistic (Duda and Hart, 1973) and rule induction (Cohen, 1995).

The rule generation phase is the common first step in most AC algorithms, and the number of rules generated in this phase might be very large especially when the data set is massive or dense. The ultimate aim of introducing our new Looking at Class (LC) training approach is to focus on the processing time taken to generate the rules. In all AC algorithms, the process of rule generation combines disjoint itemsets irrespective of their class labels. For example, if two itemsets have uncommon class labels, the majority of AC methods join them in the rule discovery step. We argue in this paper that if we only merge itemsets with common class labels, this may significantly reduce the costs associated with processing time and memory usage.

This paper is organised as follows: Section 2 defines the AC problem and discusses some of its related works. In Section 3, we present our training algorithm and explain it using an example. Section 4 is devoted to the experimental results and finally we conclude this paper in Section 5.

2. ASSOCIATIVE CLASSIFICATION PROBLEM AND RELATED WORK

(Thabtah, et al., 2005) defined the AC problem as: Let a training data set *T* has *m* distinct attributes $A_1, A_2, \ldots, A_m$ and *C* is a list of class labels. The number of rows in *T* is denoted $|T|$. Attributes could be categorical (meaning they take a value from a finite set of possible values) or continuous (where they are real or integer). In the case of categorical attributes, all possible values are mapped to a set of positive integers. For continuous attributes, a discretisation method is first used to transform these attributes into categorical ones.

Definition 1: An *item* can be described as an attribute name A_i and its value a_i, denoted (A_i, a_i).
Definition 2: The j_{th} *row* or a *training object* in *T* can be described as a list of items $(A_{j1}, a_{j1}), \ldots, (A_{jk}, a_{jk})$, plus a class denoted by c_j.
Definition 3: An *itemset* can be described as a set of disjoint attribute values contained in a training object, denoted $< (A_{i1}, a_{i1}), \ldots, (A_{ik}, a_{ik})>$.
Definition 4: A *ruleitem r* is of the form <*cond*, *c*>, where condition *cond* is an itemset and $c \varepsilon C$ is a class.
Definition 5: The actual occurrence (*actoccr*) of a *ruleitem r in T* is the number of rows in *T* that match *r's* itemset.
Definition 6: The support count (*suppcount*) of *ruleitem r* = <*cond*, *c*> is the number of rows in *T* that matches *r's* itemset, and belongs to a class *c*.
Definition 7: The occurrence (*occitm*) of an itemset *I* in *T* is the number of rows in *T* that match *I*.
Definition 8: An itemset *i* passes the minimum support (*minsupp*) threshold if $(occitm(i)/|T|) \geq minsupp$. Such an itemset is called *frequent* itemset.
Definition 9: A *ruleitem r* passes the *minsupp* threshold if, $suppcount(r)/ |T| \geq minsupp$. Such a *ruleitem* is said to be a *frequent ruleitem*.
Definition 10: A *ruleitem r* passes the minimum confidence (*minconf*) threshold if $suppcount(r) / actoccr(r) \geq minconf$.
Definition 11: An associative rule is represented in the form: $cond \rightarrow c$, where the antecedent is an itemset and the consequent is a class.

The problem of AC is to discover a subset of rules with significant supports and high confidences. This subset is then used to build an automated classifier that could be used to predict the classes of previously unseen data. It should be noted that MinSupp and MinConf terms in ARM are different than those defined in AC since classes are not considered in ARM, only itemsets occurrences are used for the computation of support and confidence.

Classification Based on Associations (CBA) was presented by (Liu et al., 1998) and it uses Apriori candidate generation method (Agrawal and Srikant, 1994) for the rule discovery step. CBA operates in three steps, where in step 1, it discretises continuous attributes before mining starts. In step 2, all frequent ruleitems which pass the MinSupp threshold are found, finally a subset of these that have high confidence are chosen to form the classifier in step3. Due to a problem of generating many rules for the dominant classes or few and sometime no rules for the minority classes, CBA (2) has introduced by (Liu *et al*. 1999), which uses multiple support thresholds for each class based on class frequency in the training data set. Experiment results have shown that CBA (2) outperforms CBA and C4.5 in terms of accuracy.

Classification based on Multiple Association Rules (CMAR) adopts the FP-growth ARM algorithm (Han et al., 2000) for discovering the rules and constructs an FP-tree to mine large databases efficiently (Li et al., 2001). It consists of two phases, rule generation and classification. It adopts a FP- growth algorithm to scan the training data to find the complete set of rules that meet certain support and confidence thresholds. The frequent attributes found in the first scan are sorted in a descending order, i.e. F-list. Then it scans the training data set again to construct an FP-tree. For each tuple in the training data set, attribute values appearing in the F-list are extracted and sorted according to their ordering in the F-list. Experimental results have shown that CMAR is faster than CBA and more accurate than CBA and C4.5. The main drawback documented in CMAR is the need of large memory resources for its training phase.

Classification based on Predictive Association Rules (CPAR) is a greedy method proposed by (Yin and Han, 2003). The algorithm inherits the basic idea of FOIL in rule generation (Cohen, 1995) and integrates it with the features of AC. Multi-class Classification based on Association Rule (MCAR) is the first AC algorithm that used a vertical mining layout approach (Zaki et al., 1997) for finding rules. As it uses vertical layout, the rule discovery method is achieved through simple intersections of the itemsets Tid-lists, where a Tid-list contains the item's transaction identification numbers rather than their actual values. The MCAR algorithm consists of two main phases: rules generation and a classifier builder. In the first phase, the training data set is scanned once to discover the potential rules of size one, and then MCAR intersects the potential rules Tid-lists of size one to find potential rules of size two and so forth. In the second phase, the rules created are used to build a classifier by considering their effectiveness on the training data set. Potential rules that cover a certain number of training objects will be kept in the final classifier. Experimental results have shown that MCAR achieves 2-4% higher accuracy than C4.5, and CBA.

Multi-class, Multi-label Associative Classification (MMAC) algorithm consists of three steps: rules generation, recursive learning and classification. It passes over the training data set in the first step to discover and generate a complete set of rules. Training instances that are associated with the produced rules are discarded. In the second step, MMAC proceeds to discover more rules that pass MinSupp and MinConf from the remaining unclassified instances, until no further potential rules can be found. Finally, rule sets derived during each iteration are merged to form a multi-label classifier that is then evaluated against test data. The distinguishing feature of MMAC is its ability to generate rules with multiple classes from data sets where each data objects is associated with just a single class. This provides decision makers with useful knowledge discarded by other current AC algorithms.

3. THE PROPOSED RULE DISCOVERY ALGORITHM

Since CBA adopts Apriori candidate generation method in its rule discovery step, the discovery of frequent itemsets is accomplished by levels wise search, where in the first level, CBA counts the support of itemsets of length one (1-itemsets), and determines whether or not they are frequent. Then, in each subsequent level, the procedure starts with itemsets found to be frequent in the previous level and merges them, regardless of their class labels, in order to produce candidate itemsets in the current level. Our idea is to improve the merging of the disjoint frequent itemsets in the CBA training phase at each level by looking at itemsets class labels. If both itemsets are associated with the same class, join them, otherwise don't join them.

In the training phase of CBA (Liu et al., 1998) and CBA (2)(Liu et al., 1999), we noticed after the initial iteration that the merging of itemsets of size *K* in order to produce itemsets of size *K*+1, is done without considering the class labels of these itemsets and thus, wasting a considerable amount of CPU time. We aimed to decrease the computational time during the frequent itemsets discovery phase of CBA by considering the class labels of any frequent itemset pairs prior to merging. For instance, if A and B are two itemsets found at iteration 1, our approach considers merging itemsets A and B only if A and B share a common class. This may improve the search process by reducing the number of mergings during each iteration and consequently reduce the computational time significantly, especially for large and dense data sets.

We can summarise the LC algorithm as follows:

1. Scans the database to find candidate 1- itemsets, which are then pruned using the support threshold to generate frequent 1- itemsets
2. Candidate 1- ruleitems of the form < $A_i \rightarrow$ c>, where A_i represents an itemset and 'c' a class label, are formed by scanning the database again.
3. Frequent 1 – ruleitems are generated, those are ruleitems which pass the MinSupp threshold. It should be noted that there may be more than one class associated with an itemset, in this case we consider the highest frequency class associated with that itemset.
4. Frequent 1-ruleitems are used for the generation of candidate 2-ruleitems, with the consideration of common class labels. In other words, only 1-ruleitems with common class labels are joined to form candidate 2-ruleitems.
5. The process is repeated at each iteration until all the frequent ruleitem are formed.
6. After all ruleitems are found, we generate them as rules and rank them based on confidence, support and rule length.

To explain our proposed training algorithm, consider for example Table 1 which contains three attributes (Age, Income, Has_car) and a class label (Buy_car) and represents whether an individual will buy a new car. Assume that MinSupp is set to 2. After the first iteration, frequent 1- ruleitems can be seen in Table 1(a). In the second iteration, disjoint frequent ruleitems are merged based on their classes; so in this case, <Age=Senior, yes> and <Income=Middle, yes> are merged because they have the same class, i.e., "YES", "Senior" and "high" are also merged in the same way. Our method does not consider joining "Senior" with "Low" since they have uncommon classes, whereas, CBA would consider joining these itemsets without checking their classes. Table 2 illustrates the itemsets produced by CBA from Table 1 using a MinSupp of 2, and Table 2(a) displays the possible 2-candidate itemsets obtained after merging frequent 1-itemsets. It is obvious from Table 2(a) that the number of merges performed by CBA is larger that LC. The difference between two approaches LC and CBA is the consideration of class labels before merging disjoint frequent ruleitems in LC, which is not the case in CBA. Significant advantage of new approach LC over CBA can be clearly seen in reduction of execution times and memory usage. This is achieved by reducing considerably the number of merging in each iteration of LC algorithm.

Table 1: Training data used By Both Approaches (LC & CBA)

Age	Income	Has_Car	Buy_car
senior	middle	n	yes
youth	low	y	no
junior	high	y	yes
youth	middle	y	yes
senior	high	n	yes
junior	high	n	no
senior	low	n	no

Table 1(a): Frequent 1-ruleitems generated By New Approach (LC)

ITEMSET	CLASS	support
senior	yes	2/7
middle	yes	2/7
low	no	2/7
high	yes	2/7
y	yes	2/7
n	no	2/7
n	yes	2/7

Table 1 (b): Candidate-2 itemsets Generated By New Approach (LC)

Itemset	Itemset	**CLASS**
senior	middle	**yes**
senior	high	**yes**
senior	n	**yes**
senior	y	**yes**
middle	n	**yes**
middle	y	**yes**
high	n	**yes**
high	y	**yes**
low	n	**no**

Table 2: Frequent 1-itemsets produced by CBA

ITEMSET	support
senior	3/7
junior	2/7
youth	2/7
middle	2/7
low	2/7
high	3/7
y	2/7
n	4/7

Table 2 (a): Candidate-2 itemsets generated by CBA using frequent 1- itemsets in the previous iteration

ITEMSET	ITEMSET		ITEMSET	ITEMSET
senior	middle		junior	middle
senior	high		junior	high
senior	low		junior	low
senior	y		junior	y
senior	n		junior	n
youth	middle		middle	y
youth	high		middle	n
youth	low		low	y
youth	y		low	n
youth	n		high	y
			high	n

Table 3: The number of merging difference of all iterations of both approaches

COMPARISON OF THE NUMBER OF MERGING AT EACH STAGE(CYCLE) FOR CBA AND OR APPROACH

Number of times itemsets have been merges at each iteration

Approach	**Data set**	**Iteration 1**	**Iteration 2**	**Iteration 3**	**Iteration 4**	**Iteration 5**	**Iteration 6**	**Total Number of merging in all stages**
LC*	Balloon	14	12	4				30
CBA**		14	32	16				62
LC	Contact	23	28	3				54
CBA		23	44	0				67
LC	Iris-Id	27	12	3				42
CBA		27	52	19				98
LC	Vote	60	423	775	897	791	0	2946
CBA			2605	8832	17396	21159	8386	58438
LC	Zoo	196	208	296	318	235	0	1253
CBA		196	4374	10220	15187	14875	6414	51266
LC	Led7	140	14	0	0	0	0	154
CBA		140	280	560	669	432	62	2143
LC	Glassd	90	66	32	9	1	0	198
CBA		90	525	885	759	325	42	2626
LC	Lymph	93	631	909	642	212	0	2487
CBA		93	2585	8460	16103	18549	7381	53171
LC	Sick	45	537	1173	1721	1722	0	5198
CBA		45	1103	3136	5650	6669	2829	19432
LC	Cleaved	50	380	697	935	796	3	2861
CBA		50	2008	8356	21794	35928	16959	85095
LC	Weather	19	20	3				42
CBA		19	44	14				77

LC*= Looking at the class CBA **= without looking at the class

4. EXPERIMENTAL RESULTS

Experiments on different data sets from UCI data collection (Merz and Murphy, 1996) were conducted. The experiments have been performed using visual C++.net implementations for both the CBA training step and our proposed algorithm on a 1 GHz processor machine with 256MB memory. We compared between our rule learning algorithm and CBA rule generation method (Agrawal and Srikant, 1994) with reference to CPU time, memory usage and more importantly the number of times itemsets are merged in the training phase in each method. The MinSupp and MinConf used in the experiments were set to 5% and 40%, respectively as in (Thabtah, at al., 2004; Thabtah, et al., 2005).

The ultimate aim of the experiments is to compute the number of times itemsets have been joined (merged) during each iteration in both CBA and our proposed method. We would like also to investigate whether reducing the number of merging during the training phase has an impact on processing time and memory usage (paged memory, physical memory and virtual memory). It should be noted that we are only investigating the training phase (learning the rules) and not the classification step (building a classifier). In other words, only the rule generation phase is experimented in this paper.

Table 3 shows the number of times itemsets have been joined in each iteration for different classification benchmark problems (Merz and Murphy, 1996) using the two approaches we consider, LC and CBA. Particularly, we compute the number of times itemsets have been merged at each iteration and for each data set we use. With the new approach, the number of itemsets that have been joined during each iteration is reduced significantly for “Vote”, “Zoo”, “Led7”, “Glassd”, “Lymph” and “Cleaved” data sets. LC has also reduced the number of joinings in the training phase for the rest of the data sets.

Furthermore, it is notable from Table 3 that the differences in the number of joinings between LC and CBA in the later iterations of large data sets like "Lymph" and "Zoo" are significant. This is because the number of itemsets available before any merging in the latter iterations is often larger that of the early iterations. For instance in the "Lymph" data set, the number of times itemsets have been merged using CBA are 2585, 8460, 16103, 18549, and 7381 for iterations, 2,3,4,5,6, respectively. Whereas, LC significantly drops the number of times itemsets have been merged during the same iterations to 631, 909, 642, 212, and 0, respectively. In general and according to Table 3, our approach saves many unnecessary itemsets merging for most data sets, which therefore should reduce the processing time and memory usage.

The processing time for both approaches is recorded and presented in Table 4. As an example, for "Lymph" data set, the execution time has reduced from 320080 ms in the CBA approach to 26007 ms in the LC, a significant difference of 91%. It should be noted that the values in iterations 3, 4 5, and 6 of the "Led7" data set for LC algorithm are zero's in Table 3 due to the fact that "Led7" data set has several different classes. In other words, after iteration 2, the remaining itemsets have different class labels which explain the zero value, and consequently lead to a large saving of 88% with reference to processing time.

It is obvious from the numbers displayed in Table 4 that the proposed algorithm saves a large amount of processing time if compared to CBA. This is because LC avoids unnecessary merging of itemsets that have uncommon class labels in iterations that follow the initial iteration. This eventually reduces the search and consequently decreases CPU time and memory usage. The processing time results of our approach on the ten data sets are consistently better than the CBA approach with the exception of the "Sick" data set. After analyzing the "Sick" data set, it turns out that it contains only two classes and the frequency of one class is much higher than the other one. In fact, almost all the itemsets in this data set are associated with the dominant class "negative", which means that the majority of the itemsets that survived the MinSupp threshold at iteration 1 are associated with an identical class. Further, since our approach looks at the class labels while merging itemsets, it will consume longer time than the CBA rule generation phase, which merges itemsets without the need to look at class labels. This explains the high CPU time and higher memory results for LC on this particular data set over that of CBA.

Table 5 shows the memory usage in terms of physical, paged and virtual for both the approaches during the training phase. The memory usage of LC in terms of physical, paged and virtual is also less for all the data sets except "Sick" than CBA because of the facts described above.

5. CONCLUSIONS

In this paper, we propose a training algorithm called LC in AC mining that merges itemsets with common class labels. We compare the proposed algorithm with the CBA rule generation algorithm on ten data sets from the UCI data repository. The proposed algorithm has shown good results, especially in terms of number of mergings in each iteration and execution times for almost all the data sets we consider. The physical memory usage is also reduced for most the data sets used in the experimental section. For future development, the LC classifier approach will be tested and validated against further test data sets. This new approach of merging itemsets can be used in most rule-based associative algorithms, to improve the execution times and to decrease the memory usage.

Table 4: Execution Time (milliseconds)

Data set	LC	CBA	Difference (%)
Balloon	140	992	85.8871
Contact	160	241	33.60996
Iris-Id	181	190	4.736842
Vote	244612	770828	68.26633
Sick	1316453	550765	-139.023
Cleved	359146	1009219	64.41347
Led7	8052	68999	88.33027
Zoo	20229	164697	87.71744
lymph	26007	320080	91.87484
weather	140	151	7.284768
glassd	1271	3054	58.38245

Table 5: Physical Memory Usage (bytes) and paged memory virtual memory

	Physical		Paged		Virtual	
Data set	LC	Apriori	LC	Apriori	LC	Apriori
Balloon	3432448	3440640	6307840	6307840	82763776	82763776
Contact	2719744	3432448	3846144	7340032	69636096	82763776
Iris-Id	3428352	3432448	6307840	6307840	82763776	82763776
Vote	12066816	12103680	14458880	14462976	130703360	130703360
Sick	12070912	11382784	14450688	13963264	130678784	126967808
Cleved	12075008	12107776	14458880	14458880	131203072	130678784
Led7	11313152	11354112	13946880	13971456	126418944	126967808
Zoo	12013568	12029952	14446592	14446592	130678784	130703360
lymph	12029952	12038144	14454784	14450688	130703360	130703360
weather	3432448	3440640	6307840	7340032	82763776	82763776
glassd	10354688	11476992	13750272	14254080	126078976	129814528

REFERENCES

1. Agrawal, R., Amielinski, T., and Swami, A. (1993) Mining association rule between sets of items in large databases. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* (pp. 207-216). Washington, DC.
2. Agrawal, R., and Srikant, R. (1994) Fast algorithms for mining association rule. *Proceedings of the 20th International Conference on Very Large Data Bases* (pp. 487-499). Santiago, Chile.
3. Cohen, W. (1995) Fast effective rule induction. *Proceedings of the 12th Internaional Conference on Machine Learning*, (pp. 115-123). CA, USA.
4. Duda, R., and Hart, P. (1973) Pattern classification and scene analysis. John Wiley & son, 1973.

5. Elmasri, R., Navathe, S. (1999) Fundamentals of database systems, Fourth Edition, Addison-Wesley.
6. Fayyad, U., Piatetsky-Shapiro, G., Smith, G., and Uthurusamy, R. (1998) Advances in knowledge discovery and data mining. AAAI Press, 1998.
7. Han, J., Pei, J., and Yin, Y. (2000) Mining frequent patterns without candidate generation. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, (pp. 1-12). Dallas, Texas.
8. Li, W., Han, J., and Pei, J. (2001) CMAR: Accurate and efficient classification based on multiple-class association rule. *Proceedings of the ICDM'01* (pp. 369-376). San Jose, CA.
9. Liu, B., Hsu, W., and Ma, Y. (1999) Mining association rules with multiple minimum supports. *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp.337-341). San Diego, California.
10. Liu, B., Hsu, W., and Ma, Y. (1998) Integrating classification and association rule mining. *Proceedings of the KDD*, (pp. 80-86). New York, NY.
11. Merz, C., and Murphy, P. (1996) UCI repository of machine learning databases. Irvine, CA, University of California, Department of Information and Computer Science.
12. Quinlan, J. (1993) C4.5: Programs for machine learning. San Mateo, CA: Morgan Kaufmann.
13. Thabtah, F., Cowling, P., and Peng, Y. (2005x2) MCAR: Multi-class classification based on association rule approach. *Proceeding of the 3rd IEEE International Conference on Computer Systems and Applications* (pp. 1-7).Cairo, Egypt.
14. Thabtah, F., Cowling, P., and Peng, Y. (2004x1) MMAC: A new multi-class, multi-label associative classification approach. *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM '04), (*pp. 217-224). Brighton, UK. (*Nominated for the Best paper award*).
15. Witten, I., and Frank, E. (2000) Data mining: practical machine learning tools and techniques with Java implementations. San Francisco: Morgan Kaufmann.
16. Yin, X., and Han, J. (2003) CPAR: Classification based on predictive association rule. *Proceedings of the SDM (*pp. 369-376). San Francisco, CA.
17. Zaki, M., Parthasarathy, S., Ogihara, M., and Li, W. (1997) New algorithms for fast discovery of association rules. *Proceedings of the 3rd KDD Conference* (pp. 283-286). Menlo Park, CA.