Incremental Classification Rules Based on Association Rules Using Formal Concept Analysis

Anamika Gupta, Naveen Kumar ,Vasudha Bhatnagar

Department of Computer Science, University of Delhi, India

Abstract. Concept lattice, core structure in Formal Concept Analysis has been used in various fields like software engineering and knowledge discovery.In this paper, we present the integration of Association rules and Classification rules using Concept Lattice. This gives more accurate classifiers for Classification. The algorithm used is incremental in nature. Any increase in the number of classes, attributes or transactions does not require the access to the previous database. The incremental behavior is very useful in finding classification rules for real time data such as image processing. The algorithm requires just one database pass through the entire database. Individual classes can have different support threshold and pruning conditions such as criteria for noise and number of conditions in the classifier.

Keywords: Classification rules, Formal concept analysis, Data Mining, Concept lattice

1 Introduction

Data Mining can be described as a process of exploration and analysis of large data sets in order to discover meaningful patterns and rules. Data Mining involves scientists from a wide range of disciplines, including mathematicians, computer scientists and statisticians, as well as those working in fields such as machine learning, artificial intelligence, information retrieval and pattern recognition. Classification rule mining and association rule mining are two important data mining techniques. Classification rule mining discovers a small set of rules in the database where consequent of the rule is a class [Q1992][Q1986].Association rule mining discovers all possible rules in the database that satisfy a user specified minimum support and minimum confidence [AS1994] [SA1996] [PBL1999]. Classification Based on Association rule (CBA) mining aims to find the rules of the form COND $\rightarrow$ CL where COND is the set of conditions and CL is the class label [LHM1998]. Few CBA algorithms have been proposed to address this issue [LHM1998] [HLZC1999] . In this paper we are discussing the method of generating Classification rules Based on Association rules using Concept Lattice of Formal Concept Analysis (CBALattice).

CBALattice provides an efficient method for constructing the concept lattice corresponding to each class label and then it provides an efficient method for

building a classifier from the lattice. This method needs only one database pass through the whole procedure. CBALattice can classify data sets irrespective of the number of classes, number of objects (i.e. rows) and number of attributes (i.e. columns). As Association rules deals with whole of data, they give more accurate rules. Since CBALattice is based on association rules so it provides more accurate rules as compared to other traditional methods such as ID3 [Q1986], C4.5 [Q1992]etc.As concepts deal with maximal item sets, concept lattice-based method provides results faster as compared to traditional methods such as ID3, C4.5 etc [HLZC1999] .

CBALattice is incremental in nature. Any increase in number of objects, attributes and classes does not need reading the previous database. Once Classification rules have been generated, concept lattice can be stored. In case of increase in objects, attributes or classes, concepts generated from the incremented database can be combined with concepts stored earlier and new classification rules can be generated.

Traditional association rule mining uses only a single minimum support in rule generation, which is inadequate for unbalanced class distribution. CBALattice method allows us to specify different minimum support for different class label.

Since CBALattice constructs a separate concept lattice for each class label so different pruning conditions can be mentioned for each class label such as criterion of deciding for noise. Also this technique can generate rule, which has many conditions. These rules may be important for accurate classification but it is difficult to find such rules in the CBA methods proposed earlier.

1.1 Related Work

Classification rule mining has been in common use since the emergence of data mining. Several algorithms have been produced such as ID3, C4.5 etc. [LHM1998] proposes an approach of integrating association rules and classification rules. [LHM1998] cannot specify different pruning conditions for each class label except minimum support threshold. Also [LHM1998]is not incremental in nature. It does not consider increase in objects, attributes or classes.

[MAR1996] proposes an algorithm SLIQ, which is scalable. This algorithm can handle large amount of data. But although scalable it is not incremental. Once rules have been generated, any change in the data is not considered. Also SLIQ makes at most two passes over the data for each level of the decision tree. CBALattice makes just one pass of the whole database. SLIQ requires pre-sorting of the data while is not needed in case of our approach.

There are few algorithms present which talk about use of concept lattice in finding Classification rules [HLZC1999] , [LN90], [GMA1995], [XHLL2002], [FFNN2004] , [CR1993] , [MZ2000]. [HLZC1999] gives an algorithm to integrate association rules and classification rules using lattices (CLACF). But it is not incremental. Also it uses same minimum support for all class labels. It cannot define different pruning conditions for each class label. Concept Lattice produces only maximal item-sets so they are much faster than traditional methods.

[HLZC1999] compares apriori and C4.5 with CLACF and found CLACF to be faster than those algorithms.

LEGAL [LM1990]can classify datasets with two classes (positive and negative examples of one concept) only. CBALattice can handle any number of classes. Also CBALattice can handle increase in number of attributes, objects and classes. LEGAL is non-incremental in nature. CLNN and CLNB [XHLL2002] use non-incremental algorithm to build a lattice. GALOIS [GMA1995]generated concepts. It does not generate the rules after that. So objects of test data have to be tested against concepts and not against classification rules. RULEARNER [S1995]uses incremental approach, but it does not make use of Concept Lattice. Our algorithm deals with only concepts, which are less in number.

Table 1 gives a fair idea of comparison of CBALattice with other lattice-based methods. Features of GRAND GALOIS, RULEARNER, CLNN CLNB has been taken from [FFNN2004] .

Table 1.

	Type of Data	Increase in Classes	Increase in Attri--butes	Increase in Objects	Lattice Type	Integra--tion of AR and CR	Min sup for different classes	Pruning condi--tions for different classes
GRAND	Bin	No	No	No	Lattice	No	Same	No
LEGAL	Bin	No	No	Yes	Lattice	No	Same	No
GALOIS	Attr/val	No	No	Yes	Lattice	No	Same	No
RULEAR-NER	Attr/val	No	No	No	Lattice	No	Same	No
CLNN, CLNB	Symb. Num.	No	No	No	Concept Lattice	No	Same	No
CLACF	Attr/val	No	No	No	Concept Lattice	Yes	Same	No
CBALattice	Bin	Yes	Yes	Yes	Concept Lattice	Yes	Different	Yes

2 Formal Concept Analysis

Formal Concept Analysis is a field of applied mathematics based on the mathematization of concept and conceptual hierarchy [GW1999]. It thereby activates mathematical thinking for conceptual data analysis and knowledge processing.

A formal context K= (G,M,I) consists of two sets G and M and a relation I between G and M. The elements of G are called the objects and the elements of M are called the attributes of the context. For a set A⊆G of objects $A' = \{\ m \epsilon M \quad | \quad gIm \ for \ all \ g \epsilon A\ \}$ (the set of all attributes common to the objects in A). Correspondingly, for a set B of attributes we define

$B' = \{\ g \epsilon G \mid gIm\ for\ all\ m \epsilon B\ \}$ (the set of objects common to the objects in A). A formal concept of the context (G,M,I) is a pair (A,B) with A⊆G,B⊆M, A′=B and B′=A. A is called the extent and B is the intent of the concept (A,B). ζ(G,M,I) denotes the set of all concepts of the context (G,M,I).

If (A1,B1) and (A2, B2) are concepts of a context, (A1,B1) is called a sub concept of (A2,B2), provided that A1⊆A2 (which is equivalent to B2⊆B1). In this case, (A2,B2) is a super concept of (A1,B1) and we write (A1,B1) ≤(A2,B2). The relation ≤ is called the hierarchical order of the concepts. The set of all concepts of (G, M, I) ordered in this way is denoted by ζ(G,M,I) and is called the concept lattice of the context (G,M,I).

An ordered set V: = (V,≤) is a lattice, if for any two elements x and y in V the supremum x⋁y and the infimum x⋀y always exist. V is called a complete lattice, if the supremum ⋁X and the infimum ⋀X exist for any subset X of V. Every complete lattice V has a largest element, ⋁X, called the unit element of the lattice, denoted by 1_v. Dually, the smallest element 0_v is called the zero elements.

Proposition 1. *Each concept of a context (G,M,I) has the form (X″,X′) for some subset X⊆G and the form (Y′,Y″) for some subset Y⊆M. Conversely all such pairs are concepts. This implies every extent is the intersection of attribute extents and every intent is the intersection of object intents.*

Using the above proposition this paper gives a method for finding all concepts of the context. Using all concepts now we draw a concept lattice corresponding to each class label.

2.1 Building the Lattice

This algorithm builds Concept Lattice for all class labels. Each class label has a different lattice. The lattice is constructed taking into consideration the minimum support threshold and maximum number of levels. We can prune the rules according to noise. If size of extent of node is less than some threshold then we can consider it as noise. Also while drawing the lattice, we can decide upon the number of levels for the lattice. More levels means more conditions in the rule. This way we can find the rules with desired number of conditions.

```
findConcepts(minSupport,maxlevel,noiseThreshold)
 { for all class labels
     findExtent(classlabel)
  // for all class labels, build a lattice
  // first find all extents of all attributes corresponding to a
     class label
  for all attributes
    findExtent(attribute)
    findIntersection(attributeExtent,classlabelExtent,intersect
    Extent)
```

```
    if support(intersectExtent) < minSupport
          break;
    if (size(intersectExtent)) < noiseThreshold)
          break;
    if intersectExtent  not present in concept list
          addExtentInConceptList(intersectExtent)
          for all concepts in ConceptList
               findIntersection(conceptExtent,intersectExtent,
               extent)
    if extent not present in concept list
              addExtentInConceptList(extent)
    endif
  addZeroElement()
  storeConceptListInFile()
  drawLattice(maxLevel)

}
```

For example, consider the following database [HK2001]

Fig 1

	a	b	c	d	e	f	g	h	i	j	CL1	CL2
1.	X					X		X	X			X
2.	X					X		X		X		X
3.		X				X		X	X		X	
4.			X		X			X	X		X	
5.			X		X			X	X		X	
6.		X		X			X			X	X	

a: age $\leq$30, b: age=31..40, c: age>40, d: income=low, e: income= medium, f: income=high, g: student=yes, h: student=no, i: credit _rating = fair, j: credit_rating = excellent, Class Label (CL1): buys_computer = yes, CL2: buys_computer = no

Concepts generated for Class Label CL1 from this database are: {36,b}, {45,cehi}, {6,bdgj}, {3,bfhi}, {345,hi}

Here we have assumed that no support threshold and no limit on level have been specified. Concept lattice:

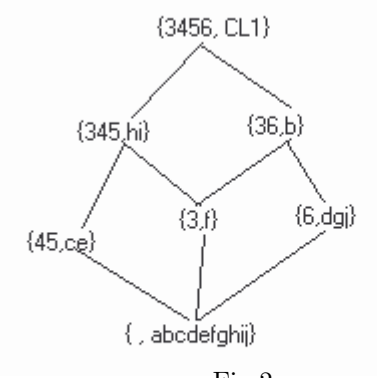

Fig 2

From the concept lattice we can read the Classification rules, which are based on following proposition:

Proposition 2. *An implication $A \rightarrow B$ holds in (G,M,I) if and only if $B \subseteq A$". This means an implication is an Association rule with 100% confidence. Method to read an implication from the concept lattice is as follows: It is sufficient to describe this procedure for implications of the form $A \rightarrow m$, since $A \rightarrow B$ holds if and only if $A \rightarrow m$ holds for each $m \epsilon B$. $A \rightarrow m$ holds if and only if (m',m") (A',A"), i.e. if $\mu\ m \geq \cap \{\ \mu\ n\ —\ n \epsilon\ A\}$. This means that we have to check in the concept lattice whether the concept denoted by m is located above the infimum of all concepts denoted by an n from A.*

2.2 Building the classifier

Using above Proposition we can now generate Classification rules.

```
findClassifier() { Start from zero element
    go up level by level
    for each branch at all levels
        keep storing the attributes
        last element i.e. unit element is the class label
        (rhs of the rule)
        all attributes connected by 'and' connective is the
        lhs of the rule
    endfor
}
```

From the Fig 2, we can find the Classifiers as

1. c $\bigwedge$ e $\bigwedge$ h $\bigwedge$ i $\Rightarrow$ CL1
2. b $\bigwedge$ f $\bigwedge$ h $\bigwedge$ i $\Rightarrow$ CL1
3. b $\bigwedge$ d $\bigwedge$ g $\bigwedge$ j $\Rightarrow$ CL1

2.3 Incremental Classification rules

Here we have described the algorithm for increase in objects (i.e. the rows). Similarly we can find concepts when increase in attributes or increase in class labels is performed.

```
incrementalLattice(minSuport, maxlevel,noiseThreshold)

{   if (objectIncremental)
    {
      for all class labels
          findExtent(newClassLabelExtent)
          read oldClassLabelExtent from file
          ClassLabelExtent = oldClassLabelExtent U
          newClassLabelExtent
          readConceptFromFile()
          for all attributes
             findAttributeExtent(newExtent)of incremented context
             readAttributeExtent(oldExtent) from file
             incrementExtent = newExtent U oldExtent
             if support(incrementExtent) < minSupport
                  break;
             if (size(intersectExtent)) < noiseThreshold)
                  break;
             if incrementExtent not present in concept list
                addExtentInConceptList(incrementExtent)
                for all concepts in ConceptList
                  findIntersection(concept,incrementExtent,extent)
                  if extent not present in concept list
                  addExtentInConceptList(extent)
             endif
      endfor
    endfor
    }
}
```

Let's assume that the incremented database is as given in Fig 3.

Fig 3

	a	b	c	d	e	f	g	h	i	j	CL1	CL2
1.	X					X		X	X			X
2.	X					X		X		X		X
3.		X				X		X	X		X	
4.			X		X			X	X		X	
5.			X		X			X	X		X	
6.		X		X			X			X	X	
7.	X			X			X		X		X	
8.			X		X		X		X		X	

Concepts generated for class label 1 from this incremented database are {7,adig}, {3, bfhi}, {45,cehi}, {3, bfhi}, {36,b}, {458,cei}, {67,dg}, {3, bfhi}, {678,g}, {345,hi}, {34578,I}, {6,bdjg}, {8,cegi}, {45,cehi}, {78,gi}

Concept Lattice:

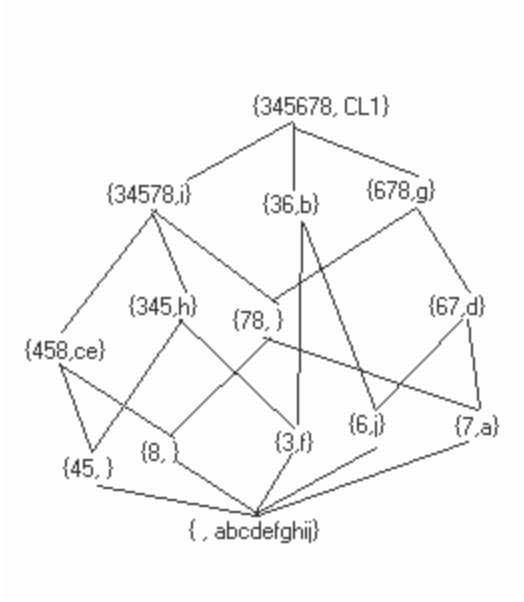

Fig 4

Few classification rules generated are:

1. a ⋀ d ⋀ i ⋀ g ⇒ CL1
2. c ⋀ e ⋀ i ⇒ CL1
3. b ⋀ f ⋀ h ⋀ i ⇒ CL1
4. c ⋀ e ⋀ h ⋀ i ⇒ CL1

3 Experiments

3.1 Computation Effort

The algorithm to find extents of all attributes requires one database pass through the entire database. After finding the extents, we put the extent of class in the list of concepts. Then we find intersection of every attribute extent with the extents in the list of concepts. This requires checking of attribute extent against previously found extents. If intersected extent does not exist in the list then it is put in the list. Effort involved here is checking of intersection of extents. If there are m attributes and n objects then effort required is checking against $m(m+1)/2$ extent list where size of each extent list is less than n. In fact size of each extent list is very less as compared to n. If we increase the number of attributes to m+k then effort involved in our algorithm is $m(k(k+1)/2)$ whereas if we start the whole process again then effort required is $(m+k)(m+k+1)/2$.

3.2 Accuracy

We have tried our algorithms on dataset from UCI [MM1996]and found that our algorithm gives quite accurate results.

Table 2.

Dataset	No of Attrs	No of Binary Attrs	No.- Classes	Size	Error Rate	No. CR with no support	No. CR with support Threshold = 60%
Tic-tac-toe	9	27	2	958	4%	346	20
Zoo	16	36	7	101	4.2%	256	15
Car	6	21	1	1728	6%	3506	436
Hepatitis	19	33	2	155	15%	1692	208

Column 1 denotes the name of the dataset. Column 2 denotes the number of attributes present in the dataset. These attributes may be binary, numeric or continuous. Since CBALattice deals with binary variables only so numeric attributes have been converted to binary attributes. Continuous variables wherever present, have been ignored. Column 3 denotes the number of binary attributes that have been obtained after conversion. In case of Hepatitis dataset, 15 attributes out of 19 attributes have been converted to binary attributes and attributes like Bilirubin, Alk Phosphate, Sgot, Albumin, and Protime have been ignored (being continuous attributes). Other datasets considered above does not have continuous variables. Column 4 denotes the number of objects present in the dataset. Column 5 denotes the error rate on the datasets. Column 6 denotes the classification rules generated from the concept lattice. Here we have assumed that no support threshold has been mentioned and no other pruning such as number of conditions in the rule, has been performed. If we perform pruning then number of rules generated will be very less. Column 7 denotes the no. of classification rules generated with support threshold = 60%.

4 Conclusion

This paper proposes a framework to integrate association rule and classification rule mining based on concept lattice of formal concept analysis. We propose an algorithm that builds a concept lattice for each class label and then finds the classification rules. The algorithm is incremental in nature. Any increase in number of objects, attributes and classes can be handled very efficiently. Also this algorithm provides a way to define different pruning conditions for different classes.

5 Future Work

CBALattice deals with only binary data. Future version will be able to handle other data. CBALattice can handle large amount of data and since CBALattice is incremental in nature so theoretically it should be scalable also. Scalability can be tested in future.

References

[AS1994] Agrawal, R. and Srikant, R. 1994, Fast algorithms for mining association rules, VLDB-94, 1994

[CR1993] Carpineto, C., Romano, G.: Galois: An order-theoretic approach to conceptual clustering. In Proceedings of ICML'93, Amherst (1993) 33-40

[FFNN2004] Huniyu Fu, Huaiguo Fu, patrik Njiwoua, Engelbert Mephu Nguifo, A Comparative study of FCA- based Supervised Classification of Algorithms, ICFCA 2004.

[GMA1995] Robert Godin, Missaoui, Hassan Alaoui, Incremental concept formation algorithm based on Galois lattice, Computational Intelligence 11: 246-267 (1995)

[GW1999] Bernhard Ganter, Rudolf Wille, Formal Concept Analysis, Mathematical Foundations, Springer (1999)

[HLZC1999] Keyun Hu, Yuchang Lu, Lizhu Zhou, Chunyi Shi, Integrating Classification and Association Rule Mining : A Concept Lattice framework, RSFDGrC 1999, 443-447

[HK2001] Jiawei Han, Micheline Kamber, Data Mining: Concepts and Techniques

[LM1990] Liquiere,M., Mephu Nguifo, E. : LEGAL: Learning with Galois lattice.

[LHM1998] Bing Liu, Wynne Hsu, Yiming Ma, Integrating Classification and Association Rule Mining. In proceedings of KDD-98, 1998.

[MAR1996] Manish Mehta, Rakesh Agrawal, Jorma Rissanen, SLIQ: A Fast Scalable Classifier for Data Mining, Proc. of the fifth Int'l Conference on Extending Database Technology

[MM1996] Merz, C.J, and Murthy, P. 1996, UCI repository of machine learning database [ftp://ftp.ics.uci.edu/pub/machine-learning-databases/]

[MZ2000] Mohammed J. Zaki, Generating Non-Reduntant Association Rules, In proceedings of the 6th International Conference on Knowledge Discovery and Data Mining (KDD'00).

[PBL1999] Pasquier N.,Bastide Y., Lakhal L. Discovering frequent closed itemsets for association rules. In proceedings of the 7th International Conference on Database Theory (ICDT'99)

[S1995] Sahami M. : Learning Classification Rules using Lattices, In Proceedings of ECML'95.

[SA1996] Srikant, R. and Agrawal, R. 1996. Mining quantitative association rules in large relational tables, SIGMOD-96

[Q1986] Quinlan, J.R. Induction of decision tree. Machine Learning, 1986

[Q1992] Quinlan, J.R. 1992, C4.5: Program for machine learning, Morgan Kaufmann

[XHLL2002] Zhipeng XIE, Wynne HSU, Zongtian LIU, Mong Li LEE, Concept Lattice based Composite Classifiers for High Predictability. Journal of Experimental and Theoretical Artificial Intelligence 14 (2002) 143-156