Journal of Theoretical and Applied Information Technology
31st March 2013. Vol. 49 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ASSOCIATIVE CLASSIFICATION WITH KNN

ZAIXIANG HUANG, ZHONGMEI ZHOU, TIANZHONG HE

Department of Computer Science and Engineering, Zhangzhou Normal University, Zhangzhou 363000, China

E-mail: huangzaixiang@126.com

ABSTRACT

Associative classification usually generates a large set of rules. Therefore, it is inevitable that an instance matches several rules which classes are conflicted. In this paper, a new framework called Associative Classification with KNN (AC-KNN) is proposed, which uses an improved KNN algorithm to address rule conflicts. Traditional K-Nearest Neighbor (KNN) is low efficient due to its calculation of the similarity between the test instance and each training instance. Furthermore, the accuracy of KNN is largely depended on the selection of a "good value" for *K*. AC-KNN generates for each test instance a specific training set composed of instances covered by best *n* rules which match the test instance. Thus, the nearest neighbors from the specific training set are not only similar to but also associative with the test instance. As a result, such nearest neighbors will make better decision on classifying a conflict instance. Our experiments on 12 UCI datasets show that AC-KNN outperforms both AC and KNN on accuracy. Compare with KNN, AC-KNN is more efficient and more stable to the number of nearest neighbors.

Keywords: *Data Mining; Classification; Association Rule; KNN; Rule conflicts*

1. INTRODUCTION

As one of the most important task in data mining and machine learning, classification aims to predict the class of an unseen instance as accuracy as possible. In recent years, associative classification (AC), which integrates association rule with classification, has been proposed [1, 2]. Because of easy interpretation and high accuracy, associative classification has become one of hot topics in data mining [3, 4, 5, 6, 7].

Most associative classification consist of two major stages, a class association rules (CARs) mining stage and a classification stage. During the first stage, a set of CARs is generated from the training set by using association rule mining algorithms, such as Apriori [8] or FPgrowth [9]. A class association rule is a specific type of association rule where the consequent is a class value. AC algorithms normally derive a large set of rules. As a result, pruning techniques are necessary to reduce redundant or misleading rules. The removal of such rules can make the classification process more effective and accurate.

During the classification stage, one of key issues is rule conflicts, i.e., several rules that match a test instance often predict different class values. To deal with this rule conflicts, there exist three different approaches: (1) using the best rule; (2) using best *n* rules; (3) using all rules. For example, CBA [1] uses the single highest ranking rule that matches an instance to classify it. This method has two shortcomings: (1) different rule ranking approaches have impact on accuracy; (2) ignoring a large number of high ranking rules that might agree with each other and disagree with the highest ranking rule. CPAR [10] uses the best *n* rules, and CMAR [2] uses all matched rules to classify an unseen instance. This method divides the rules into groups according to class value. The difficulty of this method is how to calculate the predict power of each group of rules. CMAR uses a chi-square weighting to combine the class predictions of a group of rules. ARC [11] predicts class value based on the average confidence of a group of rules.

Lindgren et al. [12] proposed a new approach to resolve rule conflicts by double induction. The idea of double induction is to induce new rules based on the instances that are covered by the rules in conflict. Antonie et al. [3] introduce a new technique to solve rule conflicts. The rules are used to transform the feature space. Then a neural network in this new feature space is used as a classifier. Depaire et al. [4] also use the learned CARs to transform the feature space. But their classification is done through case-based reasoning.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

K-Nearest Neighbor (KNN) [13] is a type of lazy learning where all computation is deferred until classification. It classifies objects based on closest training instances in the feature space. An object is classified by a majority vote of its neighbors. However, it suffers from two major deficiencies: (1) it is high computation cost because it needs to calculate the distance between the test data and each training instances; (2) it is sensitive to the number of nearest neighbors. If the K is set to small, it is easily interfered by noises. If the K is set to large, the nearest neighbors will contain many instances with other class.

In this paper, a new framework called Associative Classification with KNN (AC-KNN) is proposed, which combines the advantages of both associative classification and KNN. AC-KNN adopts an AC algorithm to generate a set of classification rules. Then it also selects the best *n* rules to predict the class value of new instances. Instead of computing the scores of each group of rules, AC-KNN applies an improved KNN algorithm to address the rule conflicts. It selects the K nearest neighbors from these instances covered by the best *n* rules rather than from all training instances. The K nearest neighbors vote to assign a class value to the test instance.

AC-KNN has three major advantages:

(1) More accurate. The K nearest neighbors selected by AC-KNN is both similar to and associative with the test instance. These nearest neighbors will make better decision on classifying the test instance. As a result, AC-KNN is better at solving rules conflicts and improves classification accuracy.

(2) More efficient than KNN. These instances covered by the best n rules are usually much smaller than training set. As a result, AC-KNN is faster than KNN to find K nearest neighbors.

(3) More stable than KNN. The K nearest neighbors is from these training instances which are associated with the test instance. Therefore, the accuracy is less sensitive to the number of the nearest neighbors.

2. PREREQUISITES AND CONCEPTION

2.1 Associative Classification

In associative classification, the training data set T has m distinct attributes $A_1, A_2, \ldots, A_m$ and a list of classes $c_1, c_2, \ldots, c_n$. An attribute can be categorical or continuous. For a categorical attribute, all the possible values are mapped to a set of consecutive positive integers. For a continuous attribute, its value range is discretized into intervals, and the intervals are also mapped to consecutive positive integers.

In general, we call each attribute-value pair an item. An itemset $X = (a_{i_1}, ..., a_{i_k})$ is a set of values of different attributes. K-itemset is an itemset that contains k values. An instance is said to match an itemset $X = (a_{i_1}, ..., a_{i_k})$, if and only if for $(1 \le j \le k)$, the instance has value a_{i_j} in attribute A_{i_j}.

The number of instances in T matching itemset X is called the support of X, denoted as $s(X)$. An itemset is called frequent itemset when the support of the itemset passes the minimum support threshold (*minsup*).

Given a training data set T, let C_i is a class label. A class association rule (CAR) is of the form: $X \to C_i$, where X is an itemset. The support count of the rule (denoted as $s(XC_i)$) is the number of objects in T that match X and belong to C_i. The support of a rule is $s(XC_i)/|T|*100\%$, where |T| is the size of the data set, and the confidence of a rule is $s(XC_i)/s(X)*100\%$.

2.2 KNN

The K Nearest Neighbors (KNN) is a simple but effective method for classification. It is a case-based learning method, which keeps all the training data for classification. For an instance to be classified, its K nearest neighbors are retrieved based on some similarity measures. For discrete attributes, the similarity between a training instance i and a new instance j is measured as follows:

$$D(i, j) = (p - m)/p \quad (1)$$

where p is the number of attributes, m is the number of share attribute values of instance i and j.

Once the K nearest neighbors is retrieved, the test instance is classified based on the majority class of its nearest neighbors.

In the majority voting approach, every neighbor has the same impact on the classification. This makes the algorithm sensitive to the choice of K. One way to reduce the impact of K is to weight the influence of each nearest neighbor according to its distance [14].

Journal of Theoretical and Applied Information Technology
31st March 2013. Vol. 49 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3. ASSOCIATIVE CLASSIFICATION WITH KNN

AC-KNN proposed in this paper consists of two phases: (1) Rule generation based on AC algorithm; (2) classification with KNN.

In the first phase, the difference of AC-KNN from other ACs is that AC-KNN mines such frequent itemsets in which items are associated with each other. The frequent and associated itemsets contribute to classification.

In the second phase, AC-KNN adopts an improved KNN algorithm to classify new instances. The difference from other KNN is that AC-KNN finds nearest neighbors from these instances covered by best *n* rules. Therefore, these nearest neighbors are not only similar to but associative with the test instance. As a result, these nearest neighbors will classify it better.

3.1 Rule Generation based on AC Algorithm

Rule generation based on AC algorithm consists of tree phases: (1) discover frequent and associated itemsets; (2) Rule ranking; (3) Rule pruning.

3.1.1 Discover frequent and associated itemsets

Our algorithm finds frequent and associated itemsets to generate class association rules (CARs). Frequent and associative itemsets are those itemsets whose support and all-confidence are greater than threshold, respectively.

All-confidence [15] of itemset $X = (a_{i_1},...,a_{i_k})$, denoted as *allconf(X)*, is defined as follows:

$$allconf(X) = s(X) / \max(s(a_{i_1}),...,s(a_{i_k})) \quad (2)$$

It represents the minimum confidence of all association rules extracted from an itemset. We use all-confidence to measure the degree of mutual association in an itemset.

Once a frequent and associative itemset has been identified, the confidence of all rules with that frequent and associative itemset as condition is calculated. Only the rule with the largest confidence is considered as a CAR.

3.1.2 Rules ranking

To select the appropriate rule for classifying new instances, most associative classifications usually rank rules firstly. Rule ranking plays an important role in associative classification [16]. CBA ranks the rules mainly according to confidence and support. When several rules have the same confidences and supports, CBA randomly chooses one of the rules, which may degrade accuracy.

To addressing this issue, we rank rules according to not only confidence but also mutual association between itemsets and predictive class which is measured by all-confidence. All-confidence of a rule $r_i: X \rightarrow c_i$ is defined as follows:

$$allconf(r) = s(r) / \max(s(X), s(c_i)) \quad (3)$$

A total order on the generated rules is defined as follows:

Given two rules, $r_i: X_i \rightarrow c_i$ and $r_j: X_j \rightarrow c_j$, r_i precedes r_j if:

1. the confidence of r_i is greater than that of r_j, or

2. their confidences are the same, but the all-confidence of r_i is greater than that of r_j, or

3. the confidence and all-confidence of r_i and r_j are the same, but all-confidence of X_i is greater than that of X_j, or

4. the confidence, all-confidence of r_i and r_j and all-confidence of X_i and X_j are the same, but the support of r_i is greater than that of r_j, or

5. all above criteria are identical for r_i and r_j, but r_i has fewer conditions in its left hand side than that of r_j.

3.1.3 Rules Pruning

The number of CARs can be huge. To make the classification effective and also efficient, we need to prune rules to delete redundant and noisy information.

Our algorithm employs the following methods for rule pruning.

First, we delete these single items with high information entropy. Information entropy of item X, denoted as E(X) is defined as follows [17]:

$$E(X) = -\frac{1}{\log_2 k} \sum_{i=1}^{k} p(C_i \mid X) \log_2 p(C_i \mid X) \quad (4)$$

where k is the number of classes, and $p(C_i \mid X)$ is the probability that an object matching X belongs to C_i.

The rationale is that these itemsets carry little information for classification when its information entropy is approximately 1.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Second, we use general rule to pruning more specific rule. A rule $r_1 : X \rightarrow C_i$ is said a ***general rule*** with respect to rule $r_2: X' \rightarrow C_j$, if and only if $X \subset X'$.

Given two rules r_i and r_j, where r_i is a **general rule** with respect to r_j. We prune r_j if the confidence of r_i is greater than that of r_j. Thus more specific rules with low confidence should be pruned. The pruning strategies are pursued when the class association rules are inserted into the set of rules.

Third, we prune rules based on database coverage just like used in CBA. This pruning is pursued when the rule mining process is finished.

3.1.4 Rule generation based on AC algorithm

The detail of Rule generation based on AC (RGAC) algorithm is shown in Algorithm 1.

Algorithm 1 RGAC(T)

Input: Training data set T; Minimum Support threshold (minsup); Minimum all-confidence threshold (minallconf); Maximum entropy threshold (Maxentropy).

Output: A set of rules R.

1: C_1←init_pass(T);
2: ruleGen(C_1; L_1;R);
3: for (k = 2;$L_{k-1} \neq \Phi$; k ++) do
4: $C_k \leftarrow$ candidateGen(L_{k-1});
5: supportCalculation(C_k);
6: ruleGen(C_k; L_k;R);
7: end for
8: Sort(R);
9: DatabaseCoverage(R, 1);
10: return R;

Function ruleGen(C_k; L_k; R)
1: for all X ∈ C_k do
2: compute allconf(X);
3: if (s (X)≥ minsup and allconf(X) ≥ minallconf)
4: calculates the confidence of all rules with X as condition, and selects the rule with the largest confidence $r_i : X \rightarrow C_i$;
5: find all general rules of r_i in R
6: if (the confidence of each general rules of r_i is less than the confidence of r_i)
7: push r_i into R;
8: end if
9: push X into L_k;
10: end if
11: end for

In this algorithm, C_k is the set of candidate k-itemset. L_k is the set of frequent and associative k-itemset. R is the set of rules. Line 1 represents the first pass of the algorithm. In this pass, RGAC records the occurrences (rowIds) of each single item inside fast access data structures. It is convenience to get the class distribution of an itemset. Then the function ruleGen is executed, also done in each subsequent pass (line 6).

For each subsequent pass, the algorithm performs 3 major operations: candidate itemset generation (line 4), support calculation (line 5) and rule generation (line 6). When rules generation is finished, RGAC sorts rules (line 8), then uses database coverage method to prune rules (line 9).

Function candidateGen uses the frequent itemset in L_{k-1} found in the (k-1) *th* pass to generate the candidate itemset pushed into C_k, which is similar to the function apriori-gen in algorithm Apriori.

Function supportCalculation is executed to calculate the support of each itemset in C_k by intersecting the rowIds of two (k-1)-itemset which are joined into k-itemset.

In Function ruleGen, RGAC calculates the all-confidence of each candidate itemset (line 2), and only selects these itemsets which pass minimum support and all-confidence threshold (line 3). Once a frequent and associative itemset has been identified, RGAC algorithm calculates the confidence of all rules with that item set as condition. Only the rule with the largest confidence is considered as a class association rule (line 4).

If the confidence of any general rule of r_i is less than that of r_i, we regard r_i as a CAR. Otherwise, we discard it (line 5-8).

3.2 Classification based on KNN

The distinction between AC-KNN and other associative classification is that AC-KNN uses KNN to address the rule conflicts. When the matched rules is conflicted, AC-KNN makes a decision by k nearest neighbors which are selected from these training instances covered by the best *n* rules. The k nearest neighbors is not only similar to but also associative with the test instance. Therefore, the k nearest neighbors will make better decision to classify the test instance.

The Classification based on KNN algorithm is shown in algorithm 2.

Algorithm 2 Classification based on KNN

Input: Training data set T; a set of rules R, Test instance O.

Output: class value assigned to O.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

1: select best *n* rules that matched test instance O from R;
2: if the best *n* rules predict the same class value C
3: assigned C to O;
4: else
5: collect all training instances $T_1 \subseteq T$ covered by the best *n* rules;
6: for all $t \in T_1$ do
7: calculate the distance between t and O;
8: end for
9: sort T_1 by distance in ascending order;
10: select k nearest neighbors with lowest distance;
11: divides k nearest neighbors into groups according to class value;
12: calculate average distance for each group;
13: assign the class value C of the group with the lowest average distance to O;
14: end if

To classify a new instance, AC-KNN selects best *n* rules the matched that new instance (line 1). If the best *n* rules consistently predict the same class value, this class value is assigned to the new instance (line 2-3). If the best *n* rules are conflict, a KNN algorithm is applied (line 5-13). We select k training instances with lowest distance and divide these instances into groups according to class value. The class value of a group with the minimum average distance is assign to the new instance.

4. EXPERIMENTAL RESULTS

To evaluate the accuracy and efficiency of AC-KNN, we have performed an extensive performance study. In this section, we report our experimental results on comparing AC-KNN against AC and KNN. It shows that AC-KNN outperforms AC and KNN in terms of average accuracy. And we also evaluate the impact of the number of nearest neighbors on accuracy. The results show AC-KNN has more stable performance than traditional KNN.

All the experiments are performed on a 3GHz Core i5 PC with 4G main memory, running Microsoft Windows XP. We test 12 data sets from UCI Machine Learning Repository. We adopt CBA's discretize utility to discretize continuous attributes. The characteristics of 12 datasets are summarized in Table 1. In this table, inst, attr, cls are representative of instances, attributes and classes, respectively.

Table 1: UCI Dataset Characteristics

datasets	inst	attr	cls	datasets	inst	attr	cls
Anneal	898	38	6	Diabetes	768	8	2
Austral	690	14	2	Heart	270	13	2
Auto	205	25	7	Horse	368	22	2
Breast	699	10	2	Sonar	208	60	2
Cleve	303	13	2	Votes	435	16	2
Crx	690	15	2	Wine	178	13	3

A 10-fold cross validation is performed. The results are given as average of the accuracy obtained for each fold.

For associative classification, we select best *n* (*n*=3) rules to classification new instances. We design three different methods to address rule conflicts respectively: (1) using the best rule (donated AC-BR); (2) using average confidence of group of rules (donated AC-AC); (3) using KNN (donated AC-KNN). We set support threshold to 1%, all-confidence threshold to 10% and Entropy information threshold to 0.95.

For KNN, we select 5 nearest neighbors to predict the class value of the test instance. The 5 nearest neighbors are divided into groups according to class value. We assign the class value of the group with the lowest average distance to the test instance.

The accuracy results achieved by the three algorithms are shown in table 2. As we can see from the table, AC-KNN outperforms both AC-BR and AC-AC on accuracy. Furthermore, out of the 12 datasets, AC-KNN achieves the best accuracy in 8 ones. In some datasets, e.g. Auto, Horse, Sonar, AC-KNN wins the second place over 3% in accuracy. In conclusion, it is more effective to address the rule conflicts by using KNN.

Table 2: The Comparison of AC-BR, AC-AC, AC-KNN and KNN on Accuracy

datasets	AC-BR	AC-AC	AC-KNN	KNN
Anneal	96.1	96.0	**97.5**	96.9
Austral	86.4	**86.5**	**86.5**	79.6
Auto	77.0	77	81.5	**83.5**
Breast	94.5	94.5	**95.2**	95.1
Cleve	**83.3**	83	82.0	75.7
Crx	85.1	85.1	**86.1**	78.4
Diabetes	71.1	**73.4**	**73.4**	65.9
Heart	81.9	81.9	**83.0**	74.1
Horse	83.6	83.3	**86.4**	82.8
Sonar	80.5	80.5	**84.0**	76.0

Journal of Theoretical and Applied Information Technology
31st March 2013. Vol. 49 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Votes	**94.4**	94.2	92.6	90.9
Wine	95.9	92.4	96.5	**97.6**
Average	85.8	85.6	**87.1**	83.0

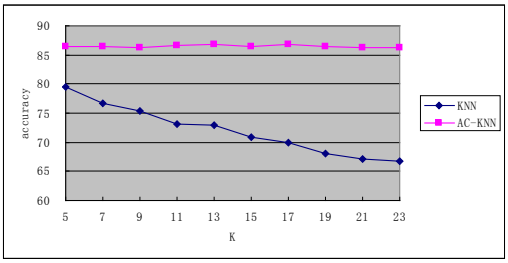
Compare column 4 with column 5, we can draw a conclusion that AC-KNN also outperforms KNN on accuracy. Out of the 12 datasets, AC-KNN achieves the best accuracy in 10 ones. In some datasets, e.g. austral, crx, diabetes, Heart, Sonar, AC-KNN wins KNN over 7% in accuracy.

Table 3: The Comparison of KNN and AC-KNN on Test Time

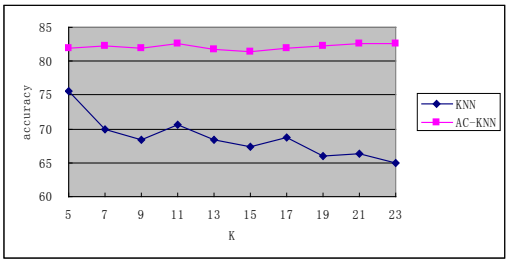
datasets	KNN	AC-KNN		datasets	KNN	AC-KNN
Anneal	2.88	0.59		Diabetes	1.02	0.86
Austral	1.10	0.23		Heart	0.14	0.07
Auto	0.13	0.03		Horse	0.38	0.12
Breast	0.93	0.37		Sonar	0.23	0.05
Cleve	0.20	0.10		Votes	0.46	0.19
Crx	1.14	0.26		Wine	0.07	0.01
				Average	0.72	0.24

Table 3 compares the test time of KNN with AC-KNN. For all datasets, AC-KNN is more efficient than KNN. On average, the test time of AC-KNN is nearly **1/3** of that of KNN. The reason is that AC-KNN finds the K nearest neighbors from these instances coved by best k rules which is much smaller than training set.

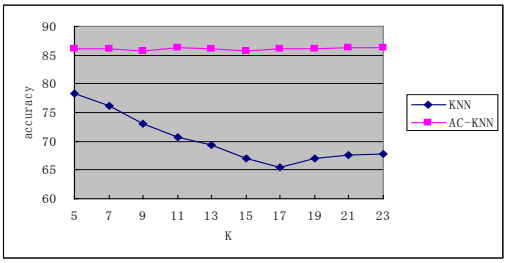
The number of nearest neighbors plays an important role on accuracy For KNN. We also compare the performance of KNN and AC-KNN on the number of nearest neighbors. The results are shown in Figure 1. From these figures, one can see that the accuracy achieved by KNN declined over 10% with the number of nearest neighbors increasing from 5 to 23. However, the accuracy results rise and fall within 2% for AC-KNN. Thus AC-KNN is less sensitive to the number of nearest neighbors than KNN.



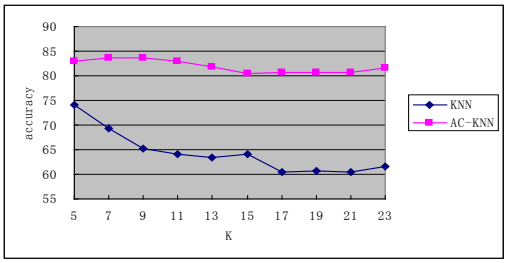
(a) Austra



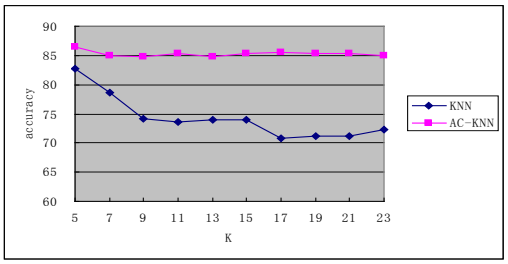
(b) Cleve



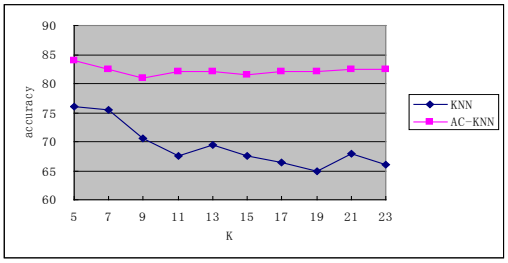
(c) Crx



(d) Heart



(e) Horse



(f) Sonar

Figure 1: The Effect of the Number of Nearest Neighbors on Accuracy

5. CONCLUSIONS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The rule conflicts are inevitable in classification stage for associative classification. The rule conflicts are addressed better, the prediction for new instances is more accuracy. In this paper, a new approach called AC-KNN is proposed, which adopts a improved KNN algorithm to address rule conflicts. AC-KNN retrieves the K nearest neighbors from the subset of training instances covered by best *n* rules. Therefore, the K nearest neighbors is not only similar to but associative with the test instance. Our experiments on UCI datasets show that AC-KNN outperforms both AC and KNN, and AC-KNN is less sensitive to the number of nearest neighbors than KNN.

ACKNOWLEDGEMENT

This work is funded by China NSF program (No.61170129).

REFRENCES:

[1] B.Liu, W.Hsu, and Y.Ma, "Integrating classification and association rule mining", *KDD*, 1998, pp.80-86.

[2] W.Li, J.Han, and J.Pei, "CMAR: Accurate and efficient classification based on multiple class-association rules", *ICDM*, 2001, pp.369-376.

[3] M.L.Antonie, O.R.Zaïane, and R.C.Holte, "Learning to use a learned model: A two-stage approach to classification", *ICDM*, 2006, pp.33-42.

[4] B.Depaire, K.Vanhoof, and G.Wets, "ARUBAS: An Association Rule Based Similarity Framework for Associative Classifiers", *ICDM*, 2008, pp. 692-699.

[5] Y.Jiang, Y.Liu, and X.Liu, "Integrating classification capability and reliability in associative classification: A β-stronger model", *Expert Systems with Applications*, vol. 37, 2010, pp.3953-3961.

[6] G.Simon, V.Kumar, and P.Li, "A Simple Statistical Model and Association Rule Filtering for Classification", *KDD*, 2011, pp. 823-831.

[7] K.Yu, X.Wu, and W.Ding, "Causal Associative Classification", *ICDM*, 2011, pp.914-923.

[8] R.Agrawal, and R.Srikant, "Fast algorithms for mining association rules", *VLDB*, 1994, pp. 487-499.

[9] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation", *SIGMOD*, 2000, pp.1-12.

[10] X. Yin and J. Han, "CPAR: classification based on predictive association rules", *SDM*, 2003, pp.331-335.

[11] M.L.Antonie, and O.R.Zaïane, "Text document categorization by term association", *ICDM*, 2002, pp.19-26.

[12] T.Lindgren, and H.Boström, "Resolving rule conflicts with double induction", *Proceedings of the 5th International Symposium on Intelligent Data Analysis*, 2003, pp.60-67.

[13] T.M. Cover, and P.E. Hart, "Nearest neighbor pattern classification", *IEEE Transactions on Information Theory*, Vol. 13, No.1, 1967, pp. 21-27.

[14] P.N.Tan, M.Steinbach, and V.Kumar, "Introduction to data mining", *Addison-Wesley*, 2006, pp.225-226.

[15] E.Omiecinski, "Alternative interest measures for mining associations", *IEEE Transactions on Knowledge and Data Engineering*, Vol.15, No 1, 2003, pp. 57-69

[16] F.A.Thabtah, "A review of associative classification mining", *The Knowledge Engineering Review*, Vol.22, No.1, 2007, pp. 37-65.

[17] Y. Zhao, and G. Karypis, "Criterion functions for document clustering: Experiments and analysis", *Machine Learning*, 2002.