An Enhanced Intelligent Database Engine by Neural Network and Data Mining

Chua Boon Lay, Marzuki Khalid* and Rubiyah Yusof
Center for Artificial Intelligence and Robotics (CAIRO)
Faculty of Electrical Engineering,
Universiti Teknologi Malaysia,
Jalan Semarak,
54100 Kuala Lumpur
e-mail: marzuki@utmnet.utm.my
(* all correspondence should be sent to)

Abstract: An Intelligent Database Engine (IDE) is developed to solve any classification problem by providing two integrated features: decision-making by a Back-propagation (BP) neural network (NN) and decision support by Apriori, a data mining (DM) algorithm. Previous experimental result shows the accuracy of NN (90%) and DM (60%) distinct drastically. Thus, effort to improve DM accuracy is crucial to ensure a well-balanced hybrid architecture. The DM poor performance is caused by either too few rules or too many poor rules are generated in the classifier. Thus, the first problem is curbed by generating multiple level rules, through incorporating multiple attribute support and level confidence to the initial Apriori. While the second problem is tackled by implementing two strengthen procedures, confidence and Bayes verification to filter out the unpredictive rule. Experiment with more dataset is carried out to compare the performance of initial and improved Apriori. Great improvement is obtained for the latter.

Keywords
Database, Data Mining, Neural Network, Decision Support, Knowledge Discovery.

I. HYBRID ARCHITECTURE

The Intelligent Database Engine (IDE) hybrid architecture is formed by combining two data analysis techniques: the intelligent decision-making (IDM) by neural network (NN) and intelligent decision support (IDS) by data mining (DM) [1]. For IDM, a set of training data is used to teach the network to acquire domain expertise. When the network is trained, a set of numerical real value inhabits in the network architecture, which represents the learned knowledge is obtained. Meanwhile, the IDS is being inputted with the same set of training data, and from this data, a different set of knowledge representation is generated. In this IDS, the data mining algorithm will model the decision-making rationale of the domain expert in the form of simple if-then rules. Finally, the trained network and the set of rules formed the architecture of this hybrid system which is able to perform decision-making and decision support.

However, before the combination is done, we tested the performance of the trained net and the rule classifier separately with a set of test data (unseen data). After testing with four datasets, it is observed that NN surprisingly performs much better than the rule classifier. The average performance of NN is 90% and rule classifier is around 60% [1]. The drastic difference alerts that if the two techniques were to combine at this stage, the overall performance might not be satisfying. Thus, we intent to improve the performance of the rule classifier before it is integrated into the hybrid system.

II. PROBLEM IDENTIFIED

To initiate the improvement, we have identified the cause of DM poor performance. It is mainly contributed by the cases of "no-rule-match". This is a case where none of the rule generated by the DM algorithm matches with the instance being evaluated. As a result, we conclude that with the absence of a result, DM makes a wrong decision. Upon investigation, the problem of "no-rule-match" is created by either too few rules or too many poor rules are generated in the classifier. With this observation, we have initiated the efforts of improvement.

III. IMPROVEMENTS

The improvements are twofold. First, we try to curb the problem of insufficient rule through multiple level rule generation. This is to modify the initial Apriori algorithm by incorporating the concepts of multiple attribute support and level confidence in order to populate the rule. Second, the poor quality rule problem is tackled by implementing two strengthen procedures, namely confidence verification and Bayes verification to filter out the unpredictive rule. Experiments with ten datasets are then carried out to test the accuracy of the rules generated after the improvements are done. Results are compared to the performance of the initial Apriori generated rule.

A. Rule Insufficiency Problem

The original Apriori algorithm faces the problem of generating insufficient rules to describe the class. Obviously, this can be seen from the original Apriori algorithm which uses only the last k-criteria-set to generate rules [2]. We argue that this algorithm has two limitations if used in digging classification rules. First, it assumes that only the last k-criteria-set is significant, other levels such as k-1, k-2... criteria-sets are not significant in concluding the class output. This is not true because the k-criteria-set is actually generated from k-1 criteria-set. Second, if all k-criteria-set do not have confidence value higher than the user specified minimum confidence (minconf), there is no rule generated at all for the particular class. As a result, all test records belonging to that class will be assigned to no-rule-match category by the classifier and this tremendously decrease the accuracy by contributing to the no-rule-match percentage. Therefore, measure should be imposed to populate the rule candidate in the classifier by keeping the valid lower level (k-1, k-2, ...) rules.

A1. Multiple Levels of Rules

To rectify the rule scarcity problem, we have implemented a multiple level rules concept to the original algorithm. As oppose to single level rule, multiple level rules take all or last *n* level of criteria-set to generate rules. *n* is the number of level of rules users want to keep. *n* should be set carefully for a large *n* will result in keeping more rules and thus consume more storage and processing time. On the other hand, a small *n* will cause the rule scarcity problem unsolved. Usually it is the last three levels of the rules which are worth keeping. Hence, by using the multiple level rules' concept, a set of rule for *n* last level criteria-sets is generated. Fig. 1 shows a graphical representation of the levelled rules.

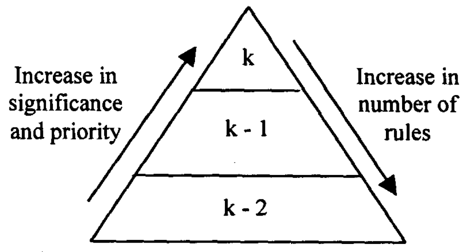
k
k - 1
k - 2
Increase in significance and priority
Increase in number of rules

Fig. 1 : Hierarchical view of levelled rules

A2. Minimum Attribute Support

Minimum attribute support is derived from the concept of multiple minimum support, which is first discussed by Liu [3]. Liu has noticed that in real work dataset, most of the items are not equally frequent. For example, in sales transaction database of a supermarket, frequency of people buying food processor and cooking pan is much less than frequency or people buying bread and milk. Therefore, there is a dilemma in setting the initial minimum support (minsup) for mining real world datasets. A low setting will cause *combinatorial explosion* where lots of frequent items of low predictive power will be produced. Conversely, a high setting will result in *rare item problem* in which item with rare frequency will be left out. Thus, in his paper, Liu argues that using a single minsup for whole dataset is inadequate because it cannot capture the inherent natures and/or frequency difference of the items in the database.

As a result, Liu has suggested a term called *minimum item support* (MIS) affiliated to individual item for his market basket analysis. User can specify different MIS for each item and thus different rules may need to satisfy different minsup depending on what items are in the rules. This implementation helps to achieve the objective of producing rare item rules without causing frequent items to generate too many meaningless rules.

Referring to Liu's work, we have initiated a multiple minimum support concept, also based on the logic of different frequency. We called it *minimum attribute support* (MAS). The idea is to assign different minsup to different attribute depends on the number of distinct value it contains. Hence, we have derived a formula for automatically calculating the MAS for each attribute. The principle is that the smaller the number of distinct value an attribute contains, the larger the MAS it has to satisfy and vice versa. The formula for counting MAS is given below.

$$\text{MAS} = 0.5 * \frac{\text{max count of attr - val}}{\text{current count of attr - val}} * \text{user defined minsup} \qquad (1)$$

```
If MAS < user defined minsup then
   MAS = user defined minsup
End if
```

(where the max count of attr-val is the maximum distinct value among the attributes and current count of attr-val is the count of distinct value for the current attribute).

For an illustration, attribute such as "occupation" and "gender" in credit approval analysis exhibits a good explanation. "Occupation" might contains 5 values (professional, government servant, skilled worker, unskilled worker and foreign worker) while gender has only 2 possible values (male, female). For a dataset of 1000 records, and user defined minsup as 20%, table below shows the calculation using formula 1 to determine the MAS value.

Table 1: Multiple minimum attribute support determination

	Occupation	Gender
Number of distinct value	5	2
Average frequency in 1000 records	200	500
MAS	20%	25%
Number of records needed for a criteria-set to be large	200	250

From Table 1, "Gender" which has smaller distinct value has to satisfy 5% more MAS than "Occupation" which is fairer with regards to their average frequency.

A3. Minimum Level Confidence

Besides MAS, we have initiated a term call minimum level confidence, which is derived from multiple minimum confidence. Multiple minimum confidence has been discussed by Fu in [4], Fu suggested that for finding multiple level association rules, different minsup and/or minconf could be specified at different "conceptual hierarchy" of the database based on a hierarchy-information structure. Conceptual hierarchy is a taxonomic organisation for concepts or objects in a database [5]. For example in a shopping transaction, conceptual hierarchy can be viewed as below:

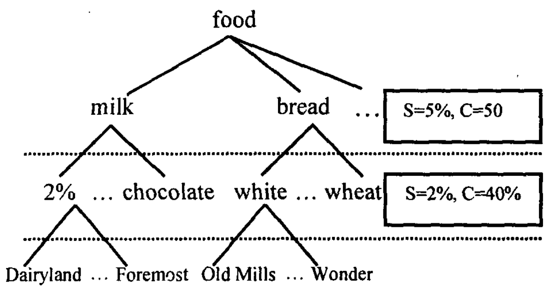

Fig. 2 : Conceptual hierarchy for shopping transaction database

Refer Fig. 2, the information becomes more detail from top to bottom. Thus, Fu set minsup as 5%, minconf as 50% for the first level and minsup as 2%, minconf as 40% for second level. The rationale for this decrement is that as the data is getting more detail; the frequency of the data is decreasing. This hierarchical level gives an idea to us for determining the minimum level confidence.

Despite the absence of a conceptual hierarchy in our database model, most of the classification problem can be viewed as a hierarchical decision tree where the nodes of the tree contains attribute and value, which direct to a leaf that contains the class value. Fig. 3 shows a possible hierarchical model of a loan application analysis based on several criteria-sets.

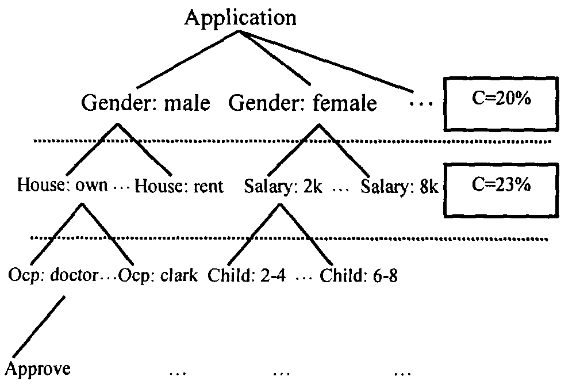

Fig. 3 : Criteria-set hierarchy for loan application analysis

In Fu's model, as the levels are going downward, minconf is decreasing (Fig. 2). However, instead of decreasing, our model increases the minconf as the level is downwarding. Refer Fig. 3, minconf for the first level is 20%, second level is 23%. This is implemented to ensure that the more details the rule provides; the more strength it is suppose to contribute to the class result (strength induces by confidence). The calculation of multiple level confidence is also automatically done based on this logic assumption. The formula to calculate the level minconf is given below:

Minimum level confidence for k-level =
User defined minconf + (k-1)* level minconf change (2)
Where level minconf change =
0.5 * (user defined minconf / max level count) (3)
where max level count =
max count of attribute selected for evaluation, excluding the class attribute
and k is the length of the criteria-set.

An example of calculating minimum level confidence is given below.

Assume:
User defined minconf = 20%
Max count of evaluation criteria = 5

Calculation:
Max level count = 5
Level minconf change = 0.5 (20/5) = 2 (using formula 3)
k-level minconf is given in Table 2 (using formula 2)

Table 2 : k-level minconf

k	2	3	4	5
Minconf (%)	22	24	26	28

Observe that the rules for different levels (k-value) will have to satisfy different minconf. At higher level, the rule has to satisfy higher minconf to ensure a larger k-criteria-set rule is always stronger than k-1 or k-2 or smaller k level rule.

Besides minimum attribute support and level confidence, there are two more modifications done to the Apriori algorithm. First, the downward closure sorting which is meant to ensure the correctness of the initial Apriori and second, the pruning 1-criteria-set by Bayes' principle, which is meant to improve the efficiency of the initial algorithm.

A4. Downward Closure Sorting

Apriori algorithm is a progressive deepening algorithm, more criteria-sets are accumulated in the antecedence of a rule at each k level. As the criteria-sets increase, the problem of determining the minsup for a rule appears. Which MAS should a rule satisfy as the criteria-sets involved have different MAS values?

To ensure a rule which involves frequent criteria-set should satisfy higher minsup while rule involves less frequent criteria-set should satisfy lower minsup, Liu [3] has defined the minsup of a rule as the smallest MAS among the criteria involved in the antecedence. However, this definition will break the downward closure property of the initial Apriori, in which it states that if a k-criteria-set satisfy minsup, then all its (k-1)-subsets should also satisfy the minsup. Thus, criteria-set which has one or more (k-1)-subsets not large in L_{k-1} can be pruned at an early stage.

The conflict between the rule minsup and the downward closure property can be illustrated by using four attributes with different number of distinct values, W, X, Y, Z. Let MAS(*i*) denotes the MAS of attribute *i*. Assume that by using Formula 1, the calculation of minimum attribute support of each attribute is given below.

MAS(W) = 10%
MAS(X) = 20%
MAS(Y) = 5%
MAS(Z) = 6%

The conflict happens when we found that 2-criteria-set (W and X) has only 9% support, for example. By implementing the downward closure in Apriori algorithm, neither one of the k-1 subset is large. Thus, this criteria-set will be discarded. However, the potential 3-criteria-set {(W and X and Y), (W and X and Z)} may be large because the minsup for rule involves (W and X and Y) is 5% and involves (W and X and Z) is 6%. It is thus wrong to discard (W and X). But if we do not discard (W and X), the downward closure property is lost.

Therefore, to curb the conflict between the rule minsup and downward closure property, Liu suggested that the attributes will be sorted according the their MAS values in ascending order before the rule generation process begin. This is called the *downward closure sorting*. The suggested minsup of a rule is the lowest MAS value or the first criteria-set MAS value among the criteria-sets in the rule. Meanwhile, to ensure the downward closure property, a constraint is enforced in the pruning step of the Apriori-gen function.

In the modified Apriori algorithm, Apriori-gen function also takes L_{k-1} as argument and returns a superset of all large k-criteria-set. It also consists of two steps, the join step and the prune step. The join step is the same as that in the initial Apriori-gen function. Basically, it joins any two criteria-sets in L_{k-1} whose first k – 2 criteria-set are the same, but the last criteria-set are different. Next, the prune step will delete those candidate criteria-sets that are impossible to be large (downward closure property). It checks and deletes any (k-1)-subset s_{k-1} of c that does not exist in L_{k-1}.

However, there is an exception in this modified Apriori algorithm, which happens when s_{k-1} does not include c[1]. This means that the first criteria-set of c, which has the lowest MAS value, is not in s_{k-1}. Then even if s_{k-1} is not in L_{k-1}, c cannot be deleted because we cannot be sure that s_{k-1} does not satisfy MAS(c[1]), although we know that it does not satisfy MAS(c[2]), unless we known that MAS(c[2]) = MAS(c[1]).

Therefore, for the previous problem, W, X, Y, Z will be sorted as Y (5%), Z (6%), W (10%) and X (20%). For example, a criteria-set of (Y,Z,W) will be checked in the pruning step of Apriori-gen. If ZW is not found in the previous large criteria-set L_2, it cannot be pruned as c[1] = Y is not in ZW. Unless we confirm that MAS(Z) = MAS(Y).

With this sorting order and the constraint enforced, the conflict is solved. Now, the constraint will help to prune those appropriate candidates that are not large at the early stage; while the minsup of the rule ensures that an appropriate minsup for the rule is assigned.

A5. Prune 1-Criteria-set by Bayes' Principle

Another enhancement done to the initial Apriori algorithm is the pruning of the first criteria-set using the principle of Naïve Bayesian. The Naïve Bayesian principle assumes all attributes are independent given the class value. Therefore, it is actually logic to prune those 1-criteria-set that contribute the least to the class value on which the rules are generated based on the probability distribute of the criteria-set.

The procedure works as follows. After generating all potential 1-criteria-set for a particular class, for each candidate c_1 in 1-criteria-set, we calculate the confidence it contributes to each class y_i by using the formula below:

$$P(y_i \mid c_1) = P(y_i)\frac{P(c_1 \mid y_i)}{P(c_1)} \qquad (4)$$

The y value of minimum $P(y_i|c_1)$ is selected and checked against the current class value. If is it equal, this means that the 1-criteria-set is actually the least frequent for that class. Instead, it is actually frequent for others and thus can be pruned to safe processing time in the later steps. By implementing this pruning step, we can discard those criteria-sets that are not significant for a particular output value at an early stage.

B. Poor Quality Rule Problem

By modifying the original Apriori to generate multiple level rules, we might generate more rules. However, among these rules, there exist many poor quality rules. These rules if can be filtered, will increase the accuracy of the classifier besides reducing the processing time and data storage. Therefore, measures should be taken to ensure the rules are correct and applicable. To achieve this objective, we have used two strengthen procedures, confidence verification and Bayes verification. These two verification techniques promise for a set of rules of higher quality.

B1. Confidence Verification

Confidence verification is a procedure for calculating the confidence validity of a rule. Confidence validity (CV) is a parameter used to measure the degree of confidence for the antecedence to the consequence of the rule. It has maximum value of 1 and minimum value of 0.

This CV is being exploited by Ali [6] to ensure every rule generated is individually accurate though the rules may not cover all classes or all examples of a given class. It is calculated by using the follow below.

$$CV = \frac{P(C \mid A)}{P(C \mid \neg A)} = \frac{\sup(C \wedge A)/\sup(A)}{(\sup(C) - \sup(C \wedge A))/(1 - \sup(A))} \quad (5)$$

Thus, we will filter the multiple level rules by pruning the rule which has CV < 0.5 (Formula 5). This enforcement ensures that all rules left are highly predictive because the probability of the criteria-set being present in the class is higher than the probability of its absence.

B2. Bayes Verification

Besides the confidence verification, we also adopt the naïve Bayesian Classifier concept [7] to the rules generated by the modified Apriori, aiming to improve the quality of the rules. Similar to confidence validity, we calculate a parameter based on naïve Bayes principle of classification. This Bayes verification is meant to make sure that statistically, each rule is contributing maximally to the class of the rule based on individual criteria-set of the rule.

To explain further, suppose we are given the values of N input attributes, $A = \{x_1, x_2, \dots x_N\}$, which can be considered independent both unconditionally and conditionally given y, the class-set. We are going to calculate the probability of the joint outcome of x which can be written as a product,

$$P(A) = P(x_1) . P(x_2) . \dots P(x_N), A = \{x_1, x_2, \dots x_N\} \quad (6)$$

And so can the probability of A within each class y where $y \in C$ be written as follows,

$$P(A|y) = P(x_1|y) . P(x_2|y) \dots P(x_N|y) \quad (7)$$

With the help of this, it is possible to write

$$P(y \mid A) = P(y)\frac{P(A \mid y)}{P(A)} = P(y)\prod_{i=1}^{N}\frac{P(x_i \mid y)}{P(x_i)} \quad (8)$$

The Bayes verification is summarised as follows. For each rule generated by Apriori, the P(y|A) for each y in C, the class attribute will be calculated by using Formula 8. We then identify the min P(y|A) of y. If y = rule.y, delete the rule. With this, we ensure that every rule is significant in terms of probability distribution. Notice that this verification is similarly used in pruning 1-criteria-set with Bayes' principle.

C. Modified Apriori

```
1) M = sort(A, MAS) (using Formula 1 to calculate MAS)
2) For each distinct value in the output field (y)
   (Part I:
   Input: a set of evaluation criteria and the output field,
   Output: a set of large k-criteria-set)
3) Filter for records that have output field value = y
4) Calculate the total number of record consisting y ($N_y$)
5) For each evaluation criteria
6)     Calculate the MAS in terms of number of records
7)     $minsup = \frac{MAS(x)}{100} * N_y$          (9)
   End for
8) $L_1 = \{<c_1>, c_1.count \geq MAS(c_1)\}$
9) Call Prune_by_Bayesian($L_1$, output_value)
10)k = 2
11)Do while $L_{k-1} \neq \phi$
12)    Call Calculate_minimum_level_conf(k,
           max_level_cnt)
13)    $C_k$ = Apriori-gen($L_{k-1}$)
14)    For each $c \in C_k$
15)        c.count = count-c(c,y)
16)    End for
17)    $L_k = \{c \in C_k \mid c.count \geq MAS(c[1])\}$
       (Part II:
       Input: a set of large k-criteria-set
       Output: a set of "if... then..." rule)
18)    $R_k = \{r \in R_k \mid r.confidence \geq minconf_k\}$
19)    k = k + 1
20)end while
21)end for
22)Levelled Rules = $\cup_k R_k$
```

```
Prune_by_Bayesian($L_1$, output_value)
    For each $c_1$ in $L_1$
        Calculate its confidence for all the class value by
        using Formula 4 below:
            $P(y_i \mid c_1) = P(y_i)\frac{P(c_1 \mid y_i)}{P(c_1)}$
        if min($P(y_i|c_1)$) of y = output_value then
            delete $c_1$
    Next $c_1$
End Prune_by_Bayesian
```

```
Calculate_minimum_level_conf(k, max_level_cnt)
    Level_minconf_change =
    0.5 * (user_defined_minconf / max_level_cnt)
    $minconf_k$ =
    user_defined_minconf + (k-1) * evel_minconf_change
End Calculate_minimum_level_conf
```

```
Apriori-gen($L_{k-1}$)
    (join step)
    insert into $C_k$
    select p.criteria-set$_1$, p.criteria-set$_2$, ..., p.criteria-set$_{k-1}$,
    q.criteria-set$_{k-1}$ from $L_{k-1}$ p, $L_{k-1}$ q
    where p.criteria-set$_1$ = q.criteria-set$_1$, ..., p.criteria-set$_{k-2}$
    = q.criteria-set$_{k-2}$, p.criteria-set$_{k-1}$ < q.criteria-set$_{k-1}$
    (prune step)
    for each $c \in C_k$ do
        for each (k-1)-subset $s_{k-1} \subset c$ do
            if ($c[1] \in s_{k-1}$) or (MAS(c[2]) = MAS(c[1])) then
                if ($s_{k-1} \notin L_{k-1}$) then
                Remove c from $C_k$
        next $s_{k-1}$
    next c
End Apriori-gen

count-c(c,y)
count-c = count the number of records with
            (antecedence = c and consequence = y)
End count-c
```

IV. EXPERIMENTS AND COMPARISON

Two experiments have been carried out. The first experiment is meant to compare the performance of the rule classifier generated by initial and improved Apriori (refer Table 1). The second experiment is aim to compare the accuracy rate of the IDE, NN and DM (refer Table 2).

Table 3 : Accuracy comparison between initial and enhanced Apriori

No	Dataset	Initial		Improved	
		N (%)	A (%)	N (%)	A (%)
1.	German Credit	43.6	24.7	0.0	73.0
2.	Breast-cancer	65.7	34.3	10.3	83.6
3.	Nursery	2.4	78.0	0.0	90.5
4.	Dermatology	55.2	22.9	2.5	70.9
5.	Zoo	40.0	60.0	12.5	80.0
6.	Mushroom	42.2	57.8	3.1	87.7
7.	Iris	17.8	80.0	0.0	93.3
8.	Car	12.0	70.3	4.4	78.6
9.	Japanese credit	50.5	45.7	5.7	73.6
10.	Australian credit	36.2	57.3	0.0	77.2
Average :		36.6	53.1	3.9	80.8

N : no-rule-match A : accuracy

From Table 3, it is observed that the no-rule-match rate has been reducing by 32.7%, which is from 36.6% to 3.9%. Meanwhile, accuracy rate has been increased by 27.7%, which is from 53.1% to 80.8%. It is proved that multiple-level rules are valid and successful in eradicating rule scarcity problem.

Table 4 : Comparison of accuracy

No	Dataset	IDE (%)	NN (%)	DM (%)
1.	German Credit	78.8	77.7	73.0
2.	Breast-cancer	98.3	96.1	83.6
3.	Nursery	99.2	96.9	90.5
4.	Dermatology	92.3	79.7	70.9
5.	Zoo	97.0	95.0	80.0
6.	Mushroom	100.0	100.0	87.7
7.	Iris	97.6	95.6	93.3
8.	Car	98.8	95.2	78.6
9.	Japanese credit	93.3	87.5	73.6
10.	Australian credit	90.1	84.8	77.2
Average:		94.5	90.9	80.8

From Table 4, it is observe that the accuracy of the IDE is the highest (94.5), 3.6% higher than NN and 13.7% higher than DM.

V. CONCLUSION

Significant improvements are obtained by the enhanced Apriori. No-rule-match rate is reduced to 3.9% and accuracy is increased to 80.8%. When the two classifiers (NN and DM) are combined in the IDE, it is proved that the accuracy of the IDE result is more reliable. Therefore, it is concluded that the IDE is a unique classification tool which has the benefit of decision-making and decision support for automating many decision-making processes.

VI. REFERENCES

[1] B.L. Chua, M. Khalid and R. Yusof, "Intelligent Database by Neural Network and Data Mining", Proc. of the First National Conference on Artificial Intelligence Applications in Industry, Kuala Lumpur, Malaysia, 20-21 Oct, 1999.

[2] R. Agrawal and R. Srikant, "Fast Algorithm for Mining Association Rules", Proc. of the 20th Internation Conference on Very Large Dtabase, Santiago, Chile, 1994.

[3] B. Liu, W. Hsu and Y. Ma, "Mining Association Rules with Multiple Minimum Support", ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-99), SanDiego, CA, USA, 15-18 Aug 1999.

[4] J.W. Han and Y.J. Fu, "Discovery of Multiple-level Association Rules", Proc. of the 21st International Conference on Very Large Database (VLDB-95), Zurich, Switzerland, 1995.

[5] Y.J. Fu, "Discovery of Multipel-level of Rules from Large Databases", Simon Fraser University: Ph. D. thesis.

[6] K. Ali, S. Manganaris and R. Srikant, "Partial Classification Using Association Rules", Proc. of the 3rd International Conference on Knowledge Discovery in Databases and Data Mining, Newport Beach, 1997.

[7] A. Holst, "The Use of a Bayesian Neural Network Model for Classification Tasks", Royal Institute of Technology: Ph. D. thesis.