Rule Mining and Classification in Imperfect Databases

K. K. R. G. K. Hewawasam, K. Premaratne, S. P. Subasingha, and M.-L. Shyu

***Abstract*— A rule-based classifier learns rules from a set of training data instances with assigned class labels and then uses those rules to assign a class label for a new incoming data instance. To accommodate data imperfections, a probabilistic relational model would represent the attributes by probabilistic functions. One extension to this model uses belief functions instead. Such an approach can represent a wider range of data imperfections. However, the task of extracting frequent patterns and rules from such a "belief theoretic" relational database has to overcome a potentially enormous computational burden. In this work, we present a data structure that is an alternate representation of a belief theoretic relational database. We then develop efficient algorithms to query for belief of itemsets, extract frequent itemsets and generate corresponding association rules from this representation. This set of rules is then used as the basis on which an unknown data instance, whose attributes are represented via belief functions, is classified. These algorithms are tested on a data set collected from a testbed that mimics an airport threat detection and classification scenario where both data attributes and threat class labels may possess imperfections.**

***Index Terms*— Data imperfections, data ambiguities, data mining, association rules, classification, Dempster-Shafer belief theory**

I. Introduction

Unless they are appropriately modeled and accommodated, one is compelled to make various "assumptions" and "interpolations" to avoid the difficulties associated with data imperfections in a database. In most practical applications however such a strategy can severely impair the integrity of the decision-making process and yield inferences that are not trustworthy. For example, in a battlefield object identification scenario, one typically has to fuse evidence presented by various heterogeneous sources to make an informed decision. The reliability of sensors, their 'footprints,' differences in domain expert opinions, etc., are all sources of data imperfections that have to be properly accounted for. For instance, data imperfections generated from subjectivity of evidence can be critical; ignoring or making assumptions/interpolations regarding such evidence may lead to costly mistakes. Lack of better techniques for accommodating data imperfections is in fact a significant hindrance to the application of various computer engineering methodologies in other domains [1].

The probabilistic relational database model allows probabilistic information to be associated with attributes. In our work, we use a Dempster-Shafer (DS) belief theoretic model where each attribute is modeled via its own belief function. Such a *DS belief theoretic relational database (DS-DB)* provides a very convenient framework for modeling several common types of data imperfections: missing data, incomplete data and ambiguities generated from lack of evidence to discern among a set of hypotheses. More importantly, a DS-DB can better withstand modeling errors. Probabilistic approaches require one to make initial assumptions regarding the model (e.g., independence of events, equiprobabilities, etc.). When these are close to reality, they perform extremely well; otherwise, conclusions made can be quite misleading. On the other hand, DS models are significantly more robust to such modeling errors (see [2] and references therein). We believe that such a methodology is exactly what is needed in the present context where one may not be able to justify the typical assumptions required for a probabilistic approach.

However, discovering interesting patterns and rules from a DS-DB is an enormous challenge because of the potentially crippling computational burden. Inspired by the *itemset tree* concept in [3], the work in [4] proposes a data structure referred to as the *belief itemset tree (BIT)* to address this difficulty. It however accommodates only data ambiguities — a much narrower class of data imperfections (see [5] and Table III for the nomenclature). In this work, we extend the BIT so that it can

All authors are with the Department of Electrical and Computer Engineering, University of Miami, 1251 Memorial Drive, Coral Gables, FL 33146 USA. Email: k.hewawasam@umiami.edu, kamal@miami.edu, s.subasingha@umiami.edu and shyu@miami.edu. This material is based upon work supported by the National Science Foundation (NSF) under the ITR (Medium) Grant IIS-0325260.

accommodate general belief theoretic data imperfections; this generalization gives rise to an efficient association rule mining (ARM) algorithm enabling one to discover interesting rules hidden in the DS-DB.

This paper is organized as follows: Section II provides a brief review of essential DS notions; Section III describes how we model database imperfections within the DS framework and the types of data imperfections that we capture; Section IV describes the BIT; Section V describes how ARM is performed from the BIT; Section VI describes the task of classification; Section VII presents experimental results; finally, Section VIII provides the concluding remarks.

II. DS Theory: A Primer

A. Basic Notions

Let $\Theta = \{\theta^{(1)}, \ldots, \theta^{(n)}\}$ be a finite set of mutually exclusive and exhaustive hypotheses about some problem domain. It signifies the corresponding 'scope of expertise' and is referred to as its *frame of discernment (FoD)* [6]. A proposition $\theta^{(i)}$ — referred to as a *singleton* — represents the lowest level of discernible information in this FoD. Elements in 2^{Θ}, the power set of Θ, form all hypotheses of interest. A hypothesis that is not a singleton is referred to as a *composite hypothesis*, e.g., $(\theta^{(1)}, \theta^{(2)})$. From now on, the term "hypotheses" is used to denote both singletons and composite hypotheses.

We use $|\Theta|$ to denote the cardinality of Θ. The set $A \setminus B$ denotes all singletons in $A \subseteq \Theta$ that are not included in $B \subseteq \Theta$, viz., $A \setminus B = \{\theta^{(i)} \in \Theta : \theta^{(i)} \in A,\ \theta^{(i)} \notin B\}$; $\overline{A}$ denotes $\Theta \setminus A$.

In DS theory, the 'support' for any hypothesis A is provided via a *basic probability assignment (BPA):*

Definition 1 (BPA): The mapping $m : 2^{\Theta} \mapsto [0,1]$ is a *BPA* for the FoD Θ iff: (i) $m(\emptyset) = 0$; and (ii) $\sum_{A \subseteq \Theta} m = 1$. □

The BPA of a hypothesis is free to move into its individual singletons. This is how DS theory allows one to model the notion of *ignorance*. The set of hypotheses $\mathcal{F}$ that possesses nonzero BPAs forms the *focal elements* of Θ; the triple $\{\Theta, \mathcal{F}, m\}$ is referred to as the corresponding *body of evidence (BoE)*.

Definition 2 (Belief, Plausibility): Given a BoE $\{\Theta, \mathcal{F}, m\}$ and $A \subseteq \Theta$, (i) Bl : $2^{\Theta} \mapsto [0,1]$ where $\text{Bl}(A) = \sum_{B \subseteq A} m(B)$ is the *belief of A*; and (ii) Pl : $2^{\Theta} \mapsto [0,1]$ where $\text{Pl}(A) = 1 - \text{Bl}(\overline{A})$ is the *plausibility of A*. □

So, while $m(A)$ measures the support assigned to hypothesis A *only*, the belief assigned to A takes into account the supports for all proper subsets of A as well; $\text{Bl}(A)$ represents the total support that can move into A without any ambiguity. $\text{Pl}(A)$ represents the extent to which one finds A plausible. When each focal set contains only one element, i.e., $m(A) = 0,\ \forall |A| \neq 1$, the BPA, belief and plausibility all reduce to probability.

B. Evidence Combination

The evidence provided by two 'independent' BoEs could be 'pooled' to form a single BoE via *Dempster's rule of combination (DRC):*

Definition 3 (DRC): Suppose the two BoEs $\{\Theta, \mathcal{F}_i, m_i\},\ i = \overline{1,2}$, span the same FoD Θ. Then, if $K \equiv \sum_{C,D:C \cap D = \emptyset} m_1(C)\, m_2(D) \neq 1$, the *DRC* generates the BPA $m(.) : 2^{\Theta} \mapsto [0,1]$ where

$$m(A) = \frac{\sum_{C,D:C \cap D = A} m_1(C)\, m_2(D)}{1 - K},\ \forall A \subseteq \Theta.$$

This combination operation is denoted as $m = m_1 \oplus m_2$.

III. Representation of Imperfect Data

A. Database

We denote the database by $DB = \{R_i\},\ i = \overline{1, nDB}$, where R_i indicates a data record and nDB its cardinality (i.e., 'size'). Each R_i is taken to be of the following form:

$$R_i = [AV_i, CL_i],\ \text{with } AV_i = [A_{1,i}, \ldots, A_{nA,i}]. \quad (1)$$

Here, AV_i and CL_i denote the i-th attribute vector and its corresponding class label. Each attribute vector is taken to consist of nA attributes; $A_{j,i},\ j = \overline{1, nA}$, denotes the j-th such attribute. From now on, unless the discussion warrants otherwise, we will ignore the subscript that identifies one data record from another, i.e., R denotes R_i, AV denotes AV_i, and so on.

B. Relevant FoDs

1) Attributes: The FoD of $A_j,\ j = \overline{1, nA}$, is taken to be finite and equal to Θ_{A_j}, viz.,

$$\text{FoD}[A_j] = \Theta_{A_j} = \{\theta_{A_j}^{(1)}, \ldots, \theta_{A_j}^{(n\Theta_{A_j})}\}, \quad (2)$$

where $n\Theta_{A_j}$ denotes the number of possible values A_j may assume. The FoD Θ_{AV} of each attribute vector AV is then the cross-product of these attribute FoDs, viz.,

$$\text{FoD}[AV] = \Theta_{AV} = \prod_{j=1}^{nA} \Theta_{A_j}; \quad (3)$$

its cardinality is $|\Theta_{AV}| = \prod_{j=1}^{nA} 2^{\Theta_{A_j}}$.

We use two BPAs to capture attribute imperfections.

Definition 4 (Intra-BPA, Inter-BPA): For data record $AV = [A_1, \ldots, A_{nA}]$,

(i) an *intra-attribute BPA (intra-BPA)* is a BPA $m_{A_j} : 2^{\Theta_{A_j}} \mapsto [0,1]$ defined on the FoD Θ_{A_j} and $\{\Theta_{A_j}, \mathcal{F}_{A_j}, m_{A_j}\}$ is the *intra-attribute BoE (intra-BoE)* it generates; and

(ii) an *inter-attribute BPA (inter-BPA)* is a BPA $M_{AV} : 2^{\Theta_{AV}} \mapsto [0,1]$ defined on the FoD Θ_{AV} and $\{\Theta_{AV}, \mathcal{F}_{AV}, M_{AV}\}$ is the *inter-attribute BoE (inter-BoE)* it generates. □

From now on, we adopt the convention of an underline to denote the assumed *value* of a variable. For example, $< AV = \underline{AV} >$ where $\underline{AV} = [\underline{A}_1, \ldots, \underline{A}_{nA}]$ indicates that the attribute vector has assumed the value $\underline{AV}$. Here, $\underline{A}_j \subseteq \Theta_{A_j},\ j = \overline{1, nA}$. Note that, $\underline{A}_j = \emptyset$ denotes that the attribute A_j is "not applicable" for the attribute vector [7]. We assume that an attribute vector whose attributes are *all* "not applicable" — this corresponds to the "null set" of Θ_{AV} — is non-existent. To simplify our notation even further, we will denote $< AV = \underline{AV} >$ by simply $< \underline{AV} >$ whenever no confusion can arise.

Example. Table I shows 04 attribute records and the intra-BPA of each attribute. Each attribute record is of the type $AV = [A_1, A_2, A_3]$ where $\Theta_{A_i} = \{1, 2, 3\},\ i = \overline{1, 3}$. Inter-BPA for each attribute record is in Table II. □

TABLE I

INTRA-BPA OF EACH ATTRIBUTE IN 04 ATTRIBUTE RECORDS

Record	$< \underline{A}_1 >$		$< \underline{A}_2 >$		$< \underline{A}_3 >$	
	$\underline{A}_1$	$m_{A_1}(.)$	$\underline{A}_2$	$m_{A_2}(.)$	$\underline{A}_3$	$m_{A_3}(.)$
1	1	1.0	2	1.0	1	1.0
2	2	0.6	3	1.0	2	1.0
	(2.3)	0.4				
3	(1,2)	0.7	(1,2)	0.6	1	1.0
	(1,2,3)	0.3	3	0.4		
4	1	1.0	2	0.7	1	1.0
			3	0.3		

TABLE II

INTER-BPA FOR EACH ATTRIBUTE RECORD IN TABLE I

Record	$< \underline{A}_1 \times \underline{A}_2 \times \underline{A}_3 >$					$M_{AV}(.)$
1	1	×	2	×	1	1.00
2	2	×	3	×	2	0.60
	(2,3)	×	3	×	2	0.40
3	(1,2)	×	(1,2)	×	1	0.42
	(1,2)	×	3	×	1	0.28
	(1,2,3)	×	(1,2)	×	1	0.18
	(1,2,3)	×	3	×	1	0.12
4	1	×	2	×	1	0.70
	1	×	3	×	1	0.30

2) Class Labels: The FoD of CL is taken to be finite and equal to Θ_{CL}, viz.,

$$\text{FoD}[CL] = \Theta_{CL} = \{\theta CL^{(1)}, \ldots, \theta CL^{(n\Theta_{CL})}\}, \quad (4)$$

where $n\Theta_{CL}$ denotes the number of different class labels that are to be discerned. As for the attributes, we allow each CL to be described via its own intra-BPA $\{\Theta_{CL}, \mathcal{F}_{CL}, m_{CL}\}$.

C. Types of Imperfections

An intra-BPA allows several types of common data imperfections that may occur in a database to be conveniently modeled. For example, suppose $\Theta_{A_j} = \{\theta^{(1)}, \theta^{(2)}, \theta^{(3)}\}$. Then, the types of data imperfections that the intra-BPA $m_{A_j}(.)$ enables one to model are shown in Table III [5].

TABLE III

TYPES OF IMPERFECTIONS

Type of Data	Intra-BPA
Hard (perfect)	$m_{A_j}(\theta^{(2)}) = 1.0$
Probabilistic (focal elements are singletons)	$m_{A_j}(\theta^{(1)}) = 0.1$ $m_{A_j}(\theta^{(2)}) = 0.7$ $m_{A_j}(\theta^{(3)}) = 0.2$
Possibilistic (consonant belief function)	$m_{A_j}(\theta^{(1)}) = 0.7$ $m_{A_j}(\theta^{(1)}, \theta^{(3)}) = 0.2$ $m_{A_j}(\Theta_{A_j}) = 0.1$
Ambiguous (value is ambiguous)	$m_{A_j}(\theta^{(1)}, \theta^{(2)}) = 1.0$
Missing/unknown (complete ambiguity)	$m_{A_j}(\Theta_{A_j}) = 1.0$
Belief theoretic (most general)	$\sum_{B \subseteq \Theta_{A_j}} m_{A_j}(B) = 1.0$

The work in [4] can only accommodate ambiguous data. One major contribution of this current work is to extend this to general intra-BPAs.

D. From Intra-BPA to Inter-BPA

The intra-BPA captures the uncertainty among the values each attribute may take; an Inter-BPA is extremely useful when one is interested in capturing the interrelationship among different attributes. Given the intra-BPA for each attribute, we may generate an inter-BPA via

Definition 5 (Cylinderical Extension): [8] *Cylindrical extension* of $< \underline{A}_j >$ to Θ_{AV} is $\text{cyl}_{\Theta_{AV}}(\underline{A}_j) = \left[\Theta_{A_1}, \ldots, \Theta_{A_{j-1}}, \underline{A}_j, \Theta_{A_{j+1}}, \ldots, \Theta_{A_{nA}}\right] \subseteq \Theta_{AV}$. □

Consider the intra-BoE $\{\Theta_{A_j}, \mathcal{F}_{A_j}, m_{A_j}\}$ of the j-th attribute. Then, the following is obvious:

Lemma 1: $M_{AV}^{(j)} : 2^{\Theta_{AV}} \mapsto [0,1]$ where

$$M_{AV}^{(j)}(\text{cyl}_{\Theta_{AV}}(\underline{A}_j)) = m_{A_j}(\underline{A}_j),\ \forall \underline{A}_j \in \mathcal{F}_{A_j},$$

is a valid inter-BPA. We refer to it as the *inter-BPA generated from the intra-BPA* m_{A_j}. □

Definition 6 (Data Record BPA, Database BPA): Consider the database DB.

(i) For a given attribute record AV, suppose $M_{AV}^{(j)}(.)$ denotes the inter-BPA generated from the intra-BPA $m_{A_j}(.)$ via Lemma 1. Then, the BPA [8]

$$M_{AV}(B) = \bigoplus_{j=1}^{nA} M_{AV}^{(j)}(B),\ \forall B \subseteq \Theta_{AV},$$

is referred to as the *data record BPA* of the given data record; the corresponding BoE is the *data record BoE.*

(ii) Let $freq(AV = \underline{AV})$ be the sum of the BPAs allocated to exactly the attribute vector $\underline{AV}$ in all the data record BPAs. Then, the BPA

$$M_{DB}(AV = \underline{AV}) = \frac{freq\,(AV = \underline{AV})}{nDB},$$

for all $\underline{AV}$ that may appear as a focal element of at least one data record BPA is referred to as the *database BPA* of the database DB; the corresponding BoE is the *database BoE.* □

Example. Table II actually gives the data record BPA of each data record in Table I. □

IV. Belief Itemset Tree (BIT)

In this section, the *belief itemset tree (BIT)* first introduced in [4] is extended to accommodate general belief theoretic data imperfections.

Definition 7 (Projection, Extension): For $<\underline{AV}>$,

(i) its *k-projection* $AV^{\downarrow k}$ is the k-attribute vector $<\underline{AV}^{\downarrow k}>$, where $\underline{AV}^{\downarrow k} = [\underline{A}_1, \ldots, \underline{A}_k]$, for $k = \overline{1, nA}$ and $\underline{AV}^{\downarrow 0} = \emptyset$; and

(ii) its *k-extension* $AV^{\uparrow k}$ is the nA-attribute vector $<\underline{AV}^{\uparrow k}>$, where $\underline{AV}^{\uparrow k} = [\underline{AV}^{\downarrow k}, \Theta_{A_{k+1}}, \ldots, \Theta_{A_{nA}}]$. □

Note that $\underline{AV}^{\uparrow 0}$ generates the 'completely ambiguous' attribute vector, viz., $\underline{A}_j = \Theta_{A_j},\ \forall j = \overline{1, nA}$.

Definition 8 (Ancestor, Parent, Child): Given $<\underline{AV}>$, let $<\underline{AV}_1> = <\underline{AV}^{\uparrow k}>$ and $<\underline{AV}_2> = <\underline{AV}^{\uparrow \ell}>$. Then we say the following:

(i) $<\underline{AV}_1>$ is an *ancestor* of $<\underline{AV}_2>$ if $k < \ell$.

(ii) $<\underline{AV}_1>$ is a *parent* of $<\underline{AV}_2>$, or equivalently, $<\underline{AV}_2>$ is a *child* of $<\underline{AV}_1>$, if $<\underline{AV}_1>$ is an ancestor of $<\underline{AV}_2>$ and $k = \ell - 1$; we denote this as $<\underline{AV}_1> \lhd <\underline{AV}_2>$ or $<\underline{AV}_2> \rhd <\underline{AV}_1>$. □

So, an attribute vector can have multiple child attribute vectors; it can have only one parent though.

A. Belief Itemset Tree (BIT)

Definition 9 (BIT): The *belief itemset tree (BIT)* of the database DB consists of a set of nodes arranged in a tree structure of $nA + 1$ hierarchical levels. The set of nodes in level k, referred to as the *level-k nodes*, is denoted by $N^{(k)},\ k = \overline{0, nA}$; level-$nA$ nodes are also called the *leaf nodes* of the BIT. Level-k nodes $N^{(k)}$ correspond to the k-projections generated by only those attribute vectors in DB.

(i) For $k = \overline{1, nA}$, an individual level-k node $n_{\ell_k}^{(k)} \in N^{(k)}$ is identified via the particular value of its k-th attribute and its parent node as

$$n_{\ell_k}^{(k)} = n_{\ell_k}^{(k)}\left(\underline{A}_k, n_{\ell_{k-1}}^{(k-1)}\right),\ \ell_k \in \mathcal{I}(n_{\ell_{k-1}}^{(k-1)}),$$

where $n_{\ell_k}^{(k)}$ is the k-projection $n_{\ell_k}^{(k)} : <\underline{AV}> \mapsto <\underline{AV}^{\downarrow k}> = <[\underline{AV}^{\downarrow(k-1)}, \underline{A}_k]>$ and $n_{\ell_{k-1}}^{(k-1)} \lhd n_{\ell_k}^{(k)}$. Here, $\mathcal{I}(n_{\ell_{k-1}}^{(k-1)})$ is an index set; it spans over all the distinct k-projections that can be generated from only those attribute records in DB that map via the parent node $n_{\ell_{k-1}}^{(k-1)}$. In the BIT, 'branches' indicate only these parent-child relationships; 'node' corresponding to $n_{\ell_k}^{(k)}$ indicates its k-th attribute value $\underline{A}_k$ and a parameter $freq\,(n_{\ell_k}^{(k)}) > 0$ which denotes the sum of masses of attribute vectors that map to $n_{\ell_k}^{(k)}$.

(ii) Level-0 consists of one and only one node $N^{(0)} = n_{\ell_0}^{(0)},\ \ell_0 = \{1\}$, which corresponds to the completely ambiguous attribute vector, and we use the convention $n_{\ell_0}^{(0)} = n_{\ell_0}^{(0)}(\emptyset, \emptyset)$ and $freq\,(n_{\ell_0}^{(0)}) = nDB$. □

An algorithm that enables one to construct the BIT and, in general, update the current BIT with an incoming arbitrary attribute vector with its own data record BPA, appears in Table IV.

TABLE IV

Algorithm for updating the level-k of the BIT with the attribute vector $<\underline{AV}>$

```
1   Insert (AttributeVector AV, M_AV(AV),Level k) {
2   generate the k-projection AV^{↓k} = [A_1, ..., A_k];
3   if ∃ a 'branch' n_{ℓ_{k-1}}^{(k-1)} (A_{k-1}, .) i ◁ n_{ℓ_k}^{(k)} (A_k, n_{ℓ_{k-1}}^{(k-1)})
        for some ℓ_k ∈ I(n_{ℓ_{k-1}}^{(k-1)}) then
4       freq(n_{ℓ_k}^{(k)}) = freq(n_{ℓ_k}^{(k)}) + M_AV(AV);
5   else {
6       I(n_{ℓ_{k-1}}^{(k-1)}) = I(n_{ℓ_{k-1}}^{(k-1)}) ∪ {L} where L is a new index;
7       create the level-k node n_L^{(k)}(A_k, n_{ℓ_{k-1}}^{(k-1)}) ▷
            n_{ℓ_{k-1}}^{(k-1)} (A_{k-1}, .) with freq(n_L^{(k)}) = M_AV(AV);
8       if k < nA then InsertNewTree (n_L^{(k)}) }
9   Insert (AV, k + 1) };
10
11  InsertNewTree (Node N) {
12  create a new branch leading from N;
13  insert an empty node at its termination };
```

Example. The BIT of the data records and their data record BPAs in Table II is in Fig. 1. □

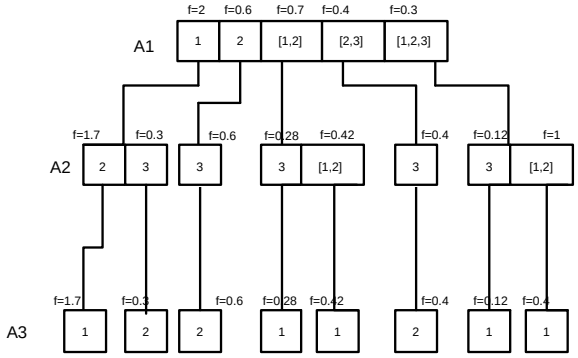

Fig. 1. BIT corresponding to the database in Table II.

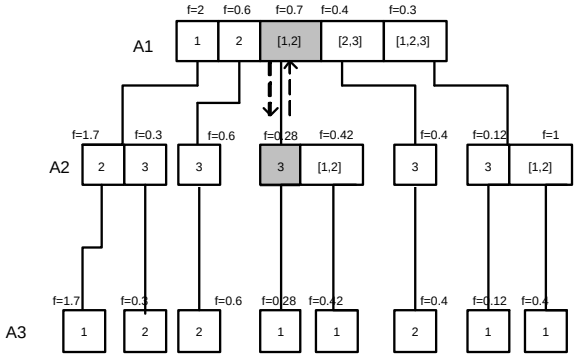

Fig. 2. Propagation of the query $Q1$ in the BIT in Fig. 1

B. BIT and Database BoE

The following result establishes the relationship between the BIT and the database BoE. Its proof is quite similar to the one in [4].

Theorem 1: Consider the database DB. Then,

(i) all attribute vectors with an identical k-projection $\underline{AV}^{\downarrow k}$ map to one and only one level-k node $n_{\ell_k}^{(k)}$;

(ii) $freq\left(n_{\ell_k}^{(k)}\right)$ denotes the sum of BPAs of all attribute vectors with an identical k-projection $\underline{AV}^{\downarrow k}$; and

(iii) the database BPA is $M_{DB}(AV = \underline{AV}) = freq\left(AV = \underline{AV}\right)/nDB = freq\left(n_{\ell_n A}^{(nA)}\right)/nDB$. □

This forms the basis which enables the BIT to be used as a very convenient tool for computing the beliefs of itemsets based on the database BPA. See Table V.

TABLE V
ALGORITHM FOR COMPUTING THE BELIEF OF THE ARBITRARY PROPOSITION $<\underline{G}>$

```
1   ComputeBelief (Belief B,Proposition G, Level k) {
2   generate the k-projection G^{↓k} = [G_1, ..., G_k];
3   for each ℓ_k ∈ I(n^{(k-1)}_{ℓ_{k-1}}) {
4     go to node n^{(k)}_{ℓ_k} = n^{(k)}_{ℓ_k}(A_k, n^{(k-1)}_{ℓ_{k-1}});
5     if G_k ⊇ A_k then {
6       if G_j = Θ_{A_j}, ∀j = k+1, nA then
7         B = B + freq(n^{(k)}_{ℓ_k});
8       else {
9         go to child node n^{(k+1)}_{ℓ_{k+1}} = n^{(k+1)}_{ℓ_{k+1}}(A_{k+1}, n^{(k)}_{ℓ_k});
10        B = B+ComputeBelief (B, G, k + 1); } } }
11  return B };
```

Example. Fig. 2 shows how the algorithm in Table V uses the BIT in Fig. 1 to respond to the query $Q1 = \text{Bl}\left([(1,2),3,\Theta_{A_3}\right)$. Note that this requires 02 nodes to be visited. Fig. 3 shows how the same BIT is used to respond to the query $Q2 = \text{Bl}\left([\Theta_{A_1},(1,2),1\right)$. Note that this requires 07 nodes to be visited. □

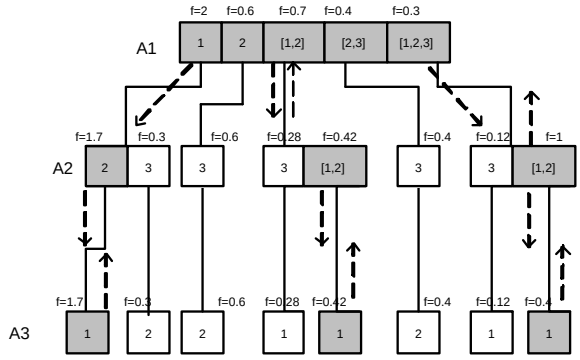

Fig. 3. Propagation of the query $Q1$ in the BIT in Fig. 1

V. ASSOCIATION RULE MINING (ARM)

An association rule is an expression of the form $R_{ant} \implies R_{con}$, where the *antecedent* R_{ant} and *consequence* R_{con} are sets of attributes. The task of association rule mining (ARM) is to generate all association rules with support and confidence measures above given thresholds [9]. If attribute and class label imperfections are to be accommodated, we must have $R_{ant} \subseteq \Theta_{AV}$ and $R_{con} \subseteq \Theta_{CL}$. Available methods [9] however require that each R_{ant} and R_{con} be a singleton, e.g., $< AV_1 = 3 > \wedge < AV_2 = 4 > \implies < CL = 1 >$. They are therefore incapable of discovering rules that accommodate imperfections, e.g., $< AV_1 = (2,3) > \wedge < AV_2 = 4 > \implies < CL = (1,2) >$.

ARM in such situations creates several significant challenges. For example, although neither of the attribute records $[2,4,\Theta_{A_3}]$ nor $[3,4,\Theta_{A_3}]$ may pass the minimum support threshold on its own, when considered together as $[(2,3),4,\Theta_{A_3}]$, they may prove successful thus potentially generating a useful rule. To perform ARM in the presence of such data imperfections, we need the following notions [4].

A. Frequent Itemsets

Definition 10 (Itemset, Support, Frequent Itemset): Consider $<\underline{AV}>$.

(i) It is referred to as a *k-itemset* if exactly $nA - k$ of its feature attributes are equal to their corresponding FoDs; I_k denotes the set of k-itemsets (by convention, I_0 is the completely ambiguous attribute vector).

(ii) $\mathrm{Bl}_{DB}(AV = \underline{AV})$ is referred to as the *support of* $<\underline{AV}>$ and is denoted by $Sp\,[AV = \underline{AV}]$.

(iii) $<\underline{AV}> \in I_k$ is said to be *frequent* if $Sp\,[AV = \underline{AV}] \geq minSp$, where $minSp$ denotes a user-defined minimum support threshold; the set of frequent k-itemsets is denoted by FI_k. □

Table VI provides an efficient **apriori**-like algorithm that is capable of generating frequent itemsets in the presence of general belief theoretic imperfections. The

TABLE VI

ALGORITHM FOR GENERATING FREQUENT ITEMSETS. FI_{all} DENOTES THE SET OF ALL FREQUENT ITEMSETS

```
1   GenerateFrequentItemSets() {
2   FI_all = ∅; FI_1 = ∅;
3    I_1 = GenerateOneItemsets(DB);
4   for each itemset I ∈ I_1
5      if Bl_DB(I) ≥ minSp then FI_1 = FI_1 ∪ I;
6    AmbiguousI_1 = GenerateAmbiguousItemsets(I_1 \ FI_1);
7   for each itemset I ∈ AmbiguousI_1
8      if Bl_DB(I) ≥ minSp then FI_1 = FI_1 ∪ I;
9   RemoveRedundantItemsets(FI_1);
10  n = 1;
11  while FI_n ≠ ∅ {
12      FI_all = FI_all ∪ FI_n;
13      I_{n+1} = GenerateCandidateItemsets(FI_n);
14      FI_{n+1} = ∅;
15     for each itemset I ∈ I_{n+1}
16       if Bl_DB(I) ≥ minSp then FI_{n+1} = FI_{n+1} ∪ I;
17      n = n + 1 }
18  return FI_all };
```

salient steps in the algorithm are as follows:

Line #3: The entire database DB is scanned to find all its 1-itemsets. The frequent itemsets that would be eventually generated from these would only consist of those attribute vectors that are present in DB. Such a strategy may prevent us from discovering other potentially important rules. For example, suppose the first attribute of each vector is modeled as a singleton. Then, with line #3 alone, it would be impossible to create potentially interesting relationships with an antecedent having an imperfect first attribute!

Line #6: To discover such relationships, we form imperfect attributes by combining the attributes of *non-frequent* 1-itemsets obtained in line #3.

Line #8: If these newly formed 1-itemsets turn out to be frequent, they are included in FI_1. For example, if $\underline{AV}_1 = [1, \Theta_{A_2}, \ldots, \Theta_{A_{nA}}] \notin AI_1$ and $\underline{AV}_2 = [2, \Theta_{A_2}, \ldots, \Theta_{A_{nA}}] \notin FI_k$, we would form the 1-itemset $\underline{AV}_1 \cup \underline{AV}_2 = [(1,2), \Theta_{A_2}, \ldots, \Theta_{A_{nA}}]$ (line #6). It is now quite possible that $\underline{AV}_1 \cup \underline{AV}_2 \in FI_1$.

Line #9: This strategy however may create 'redundant' frequent 1-itemsets. For example, suppose DB contains the records $\underline{AV}_1$ and $\underline{AV}_1 \cup \underline{AV}_2$. Then, $\underline{AV}_1 \in FI_1 \implies \underline{AV}_1 \cup \underline{AV}_2 \in FI_k$. Since $\underline{A}_{1,1} \subseteq (\underline{A}_{1,1}, \underline{A}_{1,2})$, for discovering rules that have $\underline{A}_{1,1}$ in its antecedent, it is sufficient to retain only $\underline{AV}_1 \cup \underline{AV}_2$, i.e., $\underline{AV}_1 \in FI_1$ can be considered redundant. In other words, any frequent 1-itemset that is a strict subset of another frequent 1-itemset has to be pruned.

Line #13: The next step is to form 2-itemsets from the frequent 1-itemsets thus obtained. Not all 1-itemset pairs produce 2-itemsets. For example, $\underline{AV}_1 \cap \underline{AV}_2 \notin I_2$ while $\underline{AV}_1 \cap \underline{AV}_3 \in I_2$ where $\underline{AV}_3 = [\Theta_{A_1}, 3, \Theta_{A_3}, \ldots, \Theta_{A_{nA}}] \in I_1$.

Line #16: The frequent 2-itemsets are those that pass *minSp* value.

Line #17: This same procedure is followed to generate frequent k-itemsets in general.

This algorithm calls for the computation of $\mathrm{Bl}_{DB}(AV = \underline{AV})$ often (line #16) and this is where the BIT comes in extremely handy.

B. Rule Generation

We now need to utilize an appropriate belief theoretic measure to indicate the support we place on each rule in which both attribute value and class label may possess imperfections. DS conditional notions are ideal candidates for this [10] and we define the threshold for the confidence in a rule via

$$minCf = \mathrm{Bl}_{DB}(R_{con}|R_{ant}), \quad R_{ant} \subseteq \Theta_{AV}, R_{con} \subseteq \Theta_{CL}. \quad (5)$$

Although various DS conditional notions are available in the literature, for our work, we prefer the Fagin-Halpern (FH) conditionals [11] because they can be considered the more natural extensions of the Bayesian conditional [12], [13]. We select only those rules that meet this $minCf$ threshold; denote this rule set via $\mathcal{R}_{ARM}$.

VI. CLASSIFICATION

The classifier must be able to classify incoming data instances that would possess attribute intra-BPAs.

A. Rule BPA and Rule Discount Factor

Partition $\mathcal{R}_{ARM}$ such that the antecedent of each partition is identical. Suppose the antecedent of partition $R^{(k)}$ is $\underline{R}_{ant}^{(k)}$ where $k = \overline{1, K}$.

Definition 11 (Rule BPA, Rule Discount Factor): For the partition $R^{(k)}$, define the following:

(i) $m_{CL}^{(k)} : 2^{\Theta_{CL}} \mapsto [0.1]$ s.t. $m_{\Theta_{CL}}(\underline{CL}_i) = m_{\Theta_{DB}}(\underline{CL}_i|\underline{AV}_k)$, is the *rule BPA*; and

(ii) $d^{(k)} = \left[1 + Ent^{(k)}\right]^{-1} \left[1 + \log\left[|\underline{AV}_k|\right]\right]^{-1}$, where $Ent^{(k)} = \sum_{C \subseteq \Theta_{CL}} m_{\Theta_{CL}}^{(k)}(C) \cdot \log\left[m_{\Theta_{CL}}^{(k)}(C)\right]$, is the *rule discount factor.* □

One may use the iterative technique in [13] to compute the conditional BPA required for the rule BPA. It computes the conditional BPA for each singleton first; then, it computes the conditional BPA for all doubletons, etc. This iteration can be terminated when the calculated BPA reaches a preset threshold close to unity; the remainder is assigned to the complete set Θ_{CL}. The quantity $\text{Ent}^{(k)}$ accounts for the 'uncertainty' in the rule about the class label while the term $1/(1 + \log[|\underline{AV}_k|])$ accounts for the ambiguity in the rule antecedent. Hence, $d^{(k)}$ can be considered a measure of the total uncertainty in the rule. In essence, each rule in $\mathcal{R}_{ARM}$ can be identified via the triple $\{\underline{AV}_k, m_{\Theta_{CL}}^{(k)}, d^{(k)}\}$.

When classifying an incoming attribute vector $<\underline{AV}>$ with its own intra-BPAs, the classifier first needs to generate the corresponding data record BPA as detailed in Definition 6. Then it needs to find a set of rules $R_{\underline{AV}} \subseteq \mathcal{R}_{ARM}$ that 'match' $<\underline{AV}>$. Different criteria may be used for this. We used

$$R_{\underline{AV}} = \left\{ \left\{ \underline{AV}_i, m_{\Theta_{CL}}^{(i)}, d^{(i)} \right\} : \underline{AV} \subseteq \underline{AV}_i \right\}; \quad (6)$$

if $R_{\underline{AV}} = \emptyset$, we used the classification algorithm in [14] with the distance measure $0.5\left[|\underline{AV} \setminus \underline{AV}_i| + |\underline{AV}_i \setminus \underline{AV}|\right]$. The rule BPAs of the rules in $R_{\underline{AV}}$ are then combined using the DRC with the rule discount factor taken into account [6]. The class beliefs thus generated are weighted averaged using the BPAs of corresponding focal elements of the generated data record BPA.

B. Decision Criterion

The class beliefs themselves can be considered the decision, or one can make a decision on the class label as follows: If there exists a singleton class label whose belief is greater than the plausibility of any other singleton class label, use the maximum belief with non-overlapping interval strategy [15] to make a hard decision on the class label; if such a class label does not exist, a soft decision is made in favor of the composite class label constituted of the singleton label that has the maximum belief and those singleton labels that have a higher plausibility value. If this strategy leads to decisions that are too ambiguous, one can restrict the maximum cardinality of the decision to a pre-determined number and use the maximum belief criterion.

VII. Experimental Results

The proposed strategy was applied in a security threat assessment scenario. For this purpose, an experimental platform located at the *Distributed Decision Environments (DDE) Laboratory* in University of Miami is used to mimic an airport terminal. The platform is partitioned into a grid of $3 \times 3 = 9$ areas. Passengers are assumed to pose different threat levels, $ThreatLevel$, depending on the properties of what they carry. We use 4 different types of emitters to mimic these properties: ultrasonic, sound, light and magnetic. The presence of various combinations of these properties, or $PaxType$, is sensed using sensors placed at specific locations within the terminal. The locations of passengers are tracked by a camera placed above the platform.

Sensor footprints introduce imperfections into the collected data. Data were gathered for two different scenarios: (S1) normal situation where there is no apparent threat; and (S2) abnormal situation where there can be a potential threat. A data record consists of 9 attributes, each representing the 9 grid areas. Each attribute is described via an intra-BPA defined on the FoD $\Theta_{PaxType} = \{P1, P2, P3, P4\}$. The threat level for each training set instance is taken to be *either T1 or T2,* i.e., they are perfectly classified.

Table VII shows the confusion matrix obtained for the experiment when C4.5 and the proposed algorithm are applied. The data set had to be slightly modified so that

TABLE VII
CONFUSION MATRICES FOR C4.5 AND PROPOSED ALGORITHM

$ThreatLevel$	C4.5		Proposed Algorithm		
	$T1$	$T2$	$T1$	$T2$	$(T1, T2)$
$T1$	141	16	136	6	15
$T2$	11	118	3	120	6

it could be applied to C4.5 [16]. The 'perfect' attribute values that C4.5 requires were obtained by applying the pignistic probability [2] on the intra-BPA. Note how our proposed strategy allows "soft" decisions to be made.

Table VIII compares the time taken to construct the BIT and to run 1000 belief queries on it with the time taken to compute the same belief queries by scanning

TABLE VIII

COMPUTATIONAL TIME (IN SEC)

Ambiguities	BIT		Database
	Construction	Querying	Scanning
0%	0.31	0.14	1.97
5%	0.33	0.17	1.95
10%	0.38	0.19	1.92
15%	0.39	0.19	1.95
20%	0.36	0.20	1.95
25%	0.34	0.20	1.88
30%	0.39	0.22	1.88
35%	0.36	0.22	1.89
40%	0.39	0.25	1.92

the database. Comparison was made on the synthetic threat assessment database used in [4]. Computational times were measured for datasets having different levels of ambiguities. Note that time consumed in constructing the BIT is a one-time cost for a given database. Typically a rule mining algorithm involves computing beliefs of several hundreds of thousands of itemsets and it is evident from the results that the BIT speeds up the process by a significant margin as opposed to computing these beliefs by scanning the database

VIII. CONCLUSION

As a way of modeling and accounting for data imperfections, DS belief theoretic notions have attracted considerable attention. In this paper, the BIT data structure is being proposed as a way to mitigate the otherwise heavy computational burden that is typically associated with such methods. The BIT is in essence an alternate representation of a database that may contain attribute and class label imperfections that are modeled as intra-BPAs. It allows one to efficiently respond to "belief" queries on the attribute vectors. The proposed ARM algorithm uses the BIT to extract both perfect and imperfect rules; they form the basis on which classification of an incoming data instance is performed.

Preliminary results show that this algorithm provides high classification accuracy; when sufficient evidence is lacking, it classifies the input into a "soft" class. In a real threat assessment application, various other issues also need to be taken into account, e.g., under-estimating the threat level can be more harmful than over-estimating, there may be situations where human intervention and/or immediate action have to be taken etc. To see how the proposed strategy can be used/modified in such situations, further studies have to be performed.

REFERENCES

[1] National Science Foundation. (2004) IDM 2004 Workshop. NSF Information and Data Management Program. Arlington, VA. [Online]. Available: http://www.cs.dartmounth.edu/~idm2004

[2] P. Smets, "Practical uses of belief functions," in *Proc. Conference on Uncertainty in Artificial Intelligence (UAI'99)*, K. B. Laskey and H. Prade, Eds. San Francisco, CA: Morgan Kaufmann, 1999, pp. 612–621.

[3] M. Kubat, A. Hafez, V. V. Raghavan, J. R. Lekkala, and W. K. Chen, "Itemset trees for targeted association querying," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1522–1534, Nov./Dec. 2003.

[4] K. K. R. G. K. Hewawasam, K. Premaratne, M.-L. Shyu, and S. P. Subasingha, "Rule mining and classification in the presence of feature level and class label ambiguities," in *Intelligent and Unmanned Systems, Intelligent Computing: Theory and Applications III*, ser. Proc. SPIE, Mar./Apr. 2005, Defense and Security Symposium 2005.

[5] P. Vannoorenberghe, "On aggregating belief decision trees," *Information Fusion*, vol. 5, no. 3, pp. 179–188, Sept. 2004.

[6] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton University Press, 1976.

[7] S. S. Anand, D. A. Bell, and J. G. Hughes, "EDM: A general framework for data mining based on evidence theory," *Data and Knowledge Engineering*, vol. 18, pp. 189–223, 1996.

[8] Z. Elouedi, K. Mellouli, and P. Smets, "Classification with belief decision trees," in *Proc. International Conference on Artificial Intelligence: Methodology, Systems, and Applications (AIMSA'00)*, ser. Lecture Notes in Artificial Intelligence, S. A. Cerri, , and D. Dochev, Eds. Berlin, Germany: Springer-Verlag, 2000, vol. 1904, pp. 80–90.

[9] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proc. International Conference on Very Large Data Bases (VLDB'94)*, Santiago de Chile, Chile, Sept. 1994, pp. 487–499.

[10] H. Xu and P. Smets, "Reasoning in evidential networks with conditional belief functions," *International Journal of Approximate Reasoning*, vol. 14, no. 2/3, pp. 155–185, Feb./Apr. 1996.

[11] R. Fagin and J. Y. Halpern, "A new approach to updating beliefs," in *Proc. Conference on Uncertainty in Artificial Intelligence (UAI'91)*, P. P. Bonissone, M. Henrion, L. N. Kanal, and J. F. Lemmer, Eds. New York, NY: Elsevier Science, 1991, pp. 347–374.

[12] P. Walley, *Statistical Reasoning with Imprecise Probabilities*. London, UK: Chapman and Hall, 1991.

[13] E. C. Kulasekere, K. Premaratne, D. A. Dewasurendra, M.-L. Shyu, and P. H. Bauer, "Conditioning and updating evidence," *International Journal of Approximate Reasoning*, vol. 36, no. 1, pp. 75–108, Apr. 2004.

[14] J. Zhang, S. P. Subasingha, K. Premaratne, M.-L. Shyu, M. Kubat, and K. K. R. G. K. Hewawasam, "A novel belief theoretic association rule mining based classifier for handling class label ambiguities," in *Foundations in Data Mining (FDM) Workshop, IEEE International Conference on Data Mining (ICDM'04)*, Brighton, UK, Nov. 2004.

[15] I. Bloch, "Some aspects of Dempster-Shafer evidence theory for classification of multi-modality medical images taking partial volume effect into account," *Pattern Recognition Letters*, vol. 17, no. 8, pp. 905–919, July 1996.

[16] J. R. Quinlan, *C4.5: Programs for Machine Learning*, ser. Representation and Reasoning Series. San Francisco, CA: Morgan Kaufmann, 1993.