Rule mining and classification in the presence of feature level and class label ambiguities

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

Department of Electrical and Computer Engineering
University of Miami, Coral Gables, Florida

ABSTRACT

Numerous applications of topical interest call for knowledge discovery and classification from information that may be inaccurate and/or incomplete. For example, in an airport threat classification scenario, data from heterogeneous sensors are used to extract features for classifying potential threats. This requires a training set that utilizes non-traditional information sources (e.g., domain experts) to assign a threat level to each training set instance. Sensor reliability, accuracy, noise, etc., all contribute to feature level ambiguities; conflicting opinions of experts generate class label ambiguities that may however indicate important clues. To accommodate these, a belief theoretic approach is proposed. It utilizes a data structure that facilitates belief/plausibility queries regarding "ambiguous" itemsets. An efficient **apriori**-like algorithm is then developed to extract frequent such itemsets and to generate corresponding association rules. These are then used to classify an incoming "ambiguous" data instance into a class label (which may be "hard" or "soft").

To test its performance, the proposed algorithm is compared with C4.5 for several databases from the UCI repository and a threat assessment application scenario.

Keywords: imperfect data, missing data, data ambiguities, data mining, association rules, classification, Dempster-Shafer belief theory

1. INTRODUCTION

In a classification scenario, lack of relevant statistical information usually compels one to utilize knowledge gleaned from a training set of correctly classified feature vectors to decide on a class label for an incoming feature vector. The presence of imperfections in both the training set and the feature vector that is yet to be classified however makes this task extremely challenging. These imperfections may have been generated by reliability of information sources, heterogeneity of their 'scopes of expertise,' lack of access to the global knowledge, differing opinions of domain experts whose expertise is being sought for classifying the training set, and a myriad of other causes.

Making "assumptions" and "interpolations" to avoid such imperfections can impair the decision-making process and render the inferences made less trustworthy. Critical evidence may be destroyed if one chooses to simply ignore certain types of imperfections. Indeed, development of better techniques of handling data imperfections for making improved inferences and decisions can be considered a chronic problem hampering data-driven studies that attempt to discern among competing hypotheses.[1] It has been identified as one of the most challenging problems confronting the application of methodologies developed within the realm of computer science to other domains.[2]

In this work, we use belief theoretic notions to represent data with ambiguous attributes and class labels. Via a novel data structure—we refer to this as a *belief itemset tree*—we then propose a methodology that enables the extraction of a set of "frequent" itemsets which is then used to extract association rules[3] consisting of possibly ambiguous feature attributes and class labels. These association rules can be interpreted as a "compact" representation of a given database of instances. The effectiveness of such a representation for classification purposes has previously been demonstrated[4]; class label ambiguities have also been incorporated into this same approach.[5] Our purpose is to accommodate *both* feature level and class label ambiguities.

Further author information: (Send correspondence to K.P.) K.K.R.G.K.H. E-mail `k.hewawasam@umiami.edu`, Tel 1 305 284 6503; K.P.: E-mail `kamal@miami.edu`, Tel 1 305 284 4051; M.-L.S.: E-mail `shyu@miami.edu`, Tel 1 305 284 5566; S.P.S.: E-mail `s.subasingha@umiami.edu`, Tel 1 305 284 6503

2. DS THEORY: A PRIMER

Let $\Theta = \{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(n)}\}$ 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 $\theta^{(i)}$ represents the lowest level of discernible information in this FoD; it is referred to as a *singleton.* Elements in 2^{Θ}, the power set of Θ, form all hypotheses of interest in DS theory. 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 $B \subseteq \Theta$; $\overline{A}$ denotes $\Theta \setminus A$.

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

Definition 2.1 (Basic Probability Assignment (BPA)). *The mapping* $m_{\Theta} : 2^{\Theta} \mapsto [0,1]$ *is a* basic probability assignment (BPA) *for the FoD* Θ *iff: (i)* $m_{\Theta}(\emptyset) = 0$*; and (ii)* $\sum_{A \subseteq \Theta} m_{\Theta}(A) = 1$.

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 $\mathcal{F}_{\Theta}$ that possesses nonzero BPAs forms the *focal elements* of Θ; the triple $\{\Theta, \mathcal{F}_{\Theta}, m_{\Theta}\}$ is referred to as the corresponding *body of evidence (BoE).*

Definition 2.2 (Belief and Plausibility). *Given a BoE* $\{\Theta, \mathcal{F}_{\Theta}, m_{\Theta}\}$*, define the following notions regarding* $A \subseteq \Theta$*: (i)* Belief *is* $Bl_{\Theta} : 2^{\Theta} \mapsto [0,1]$ *where* $Bl_{\Theta}(A) = \sum_{B:B \subseteq A} m_{\Theta}(B)$*; and (ii)* Plausibility *is* $Pl_{\Theta} : 2^{\Theta} \mapsto [0,1]$ *where* $Pl_{\Theta}(A) = \sum_{B:A \cap B \neq \emptyset} m_{\Theta}(B) = 1 - Bl_{\Theta}(\overline{A})$.

So, while $m_{\Theta}(A)$ 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. In other words, $\mathrm{Bl}_{\Theta}(A)$ represents the total support that can move into A without any ambiguity; and $\mathrm{Pl}_{\Theta}(A)$ represents the extent to which one finds A plausible. When each focal set contains only one element, the BPA, belief and plausibility all reduce to probability, i.e., $m_{\Theta}(A) = \mathrm{Bl}_{\Theta}(A) = \mathrm{Pl}_{\Theta}(A) = \mathrm{Pr}_{\Theta}(A)$.

The evidence provided by two 'independent' BoEs could be 'pooled' to form a single BoE via[6]

Definition 2.3 (Demspter's Rule of Combination (DRC)). *Suppose the two BoEs* $\{\Theta, \mathcal{F}_{\Theta,1}, m_{\Theta,1}\}$ *and* $\{\Theta, \mathcal{F}_{\Theta,2}, m_{\Theta,2}\}$ *span the FoD* Θ*. Then, if* $K_{\mathrm{conf}} \equiv \sum_{C,D:C \cap D = \emptyset} m_{\Theta,1}(C)\, m_{\Theta,2}(D) \neq 1$*, the* Dempster's rule of combination (DRC) *generates the BPA* $m_{\Theta}(.) : 2^{\Theta} \mapsto [0,1]$ *where* $m_{\Theta}(A) = \sum_{C,D:C \cap D = A} m_{\Theta,1}(C)\, m_{\Theta,2}(D) \div (1 - K_{\mathrm{conf}})$, $\forall A \subseteq \Theta$.

3. REPRESENTATION OF IMPERFECT DATA

In this section, we discuss how DS belief theoretic notions are being utilized to represent database imperfections—in particular, feature level and class label ambiguities—for rule mining and classification purposes.

Database. We denote the database by $DB = \{R_i\}$, $i = \overline{1, nDB}$, where R_i indicates a data record and nDB indicates the cardinality of DB, i.e., its 'size.' Each R_i is taken to be of the form $R_i = [FV_i, CL_i]$, where $FV_i = [F_{1,i}, F_{2,i}, \ldots, F_{nF,i}]$. Here, FV_i and CL_i denote the i-th feature vector and its corresponding class label. Each feature vector is taken to consist of nF features; $F_{j,i}$, $j = \overline{1, nF}$, denotes the j-th such feature embedded within R_i. From now on, unless the situation calls for distinguishing among different data records, we will simply ignore the subscript that identifies one data record from another, e.g., R_i will be denoted by R.

Relevant FoDs. We now identify the relevant FoDs.

Features. The FoD of F_j, $j = \overline{1, nF}$, is taken to be finite and equal to $\Theta[F_j] = \{\theta F_j^{(1)}, \ldots, \theta F_j^{(n\Theta[F_j])}\}$, where $n\Theta[F_j]$ is the number of possible values F_j may assume. Then, the FoD of each feature vector FV is the cross-product of all $\Theta[F_j]$, $j = \overline{1, nF}$, i.e., $\Theta[FV] = \times_{j=1}^{nF} \Theta[F_j]$. We allow each F_j to be described via its own BoE

$$\{\Theta[F_j], \mathcal{F}_{\Theta[F_j]}, m_{\Theta[F_j]}\}, \text{ where } m_{\Theta[F_j]}(.) : 2^{\Theta[F_j]} \mapsto [0,1] \text{ and } |\mathcal{F}_{\Theta[F_j]}| = 1. \quad (1)$$

Class Label. The FoD of each CL is taken to be finite and equal to $\Theta[CL] = \{\theta CL^{(1)}, \ldots, \theta CL^{(n\Theta[CL])}\}$, where $n\Theta[CL]$ is the number of different class labels. Each class label is described via its own BoE

$$\{\Theta[CL], \mathcal{F}_{\Theta[CL]}, m_{\Theta[CL]}\}, \text{ where } m_{\Theta[CL]}(.) : 2^{\Theta[CL]} \mapsto [0,1] \text{ and } |\mathcal{F}_{\Theta[CL]}| = 1. \quad (2)$$

The types of feature and class label imperfections we accommodate are those that can be modeled via such BoEs having only one focal element. Although this imposes substantial restrictions, the algorithms we propose capture several types of data imperfections that are of critical importance. For example, consider the FoD $\Theta[F_j] = \{\theta[F_j]^{(1)}, \theta[F_j]^{(2)}, \theta[F_j]^{(3)}\}$. Then, the BPA $m_{\Theta[F_j]}(\theta[F_j]^{(2)}) = 1.0$ models a 'precise' value; the BPA $m_{\Theta[F_j]}((\theta[F_j]^{(2)}, \theta[F_j]^{(3)})) = 1.0$ models an 'ambiguous' value; and the BPA $m_{\Theta[F_j]}(\Theta[F_j]) = 1.0$ models a 'missing' or 'unknown' value. From now on, for convenience, we will broadly refer to these types of imperfections as *data ambiguities.*

We will use an underline to denote the assumed *value* of a variable, e.g., $< FV = \underline{FV} >$ where $\underline{FV} = [\underline{F}_1, \underline{F}_2, \ldots, \underline{F}_{nF}]$, $\underline{F}_j \subseteq \Theta[F_j]$, $j = \overline{1, nF}$. Note that, $\underline{F}_j = \emptyset$ denotes that the attribute F_j is "not applicable" for the feature vector.[7] We assume that a feature vector whose attributes are *all* "not applicable" (corresponding to the "null set" of $\Theta[FV]$) is non-existent.

DEFINITION 3.1 (DATABASE BPA). *Let* $freq(FV = \underline{FV})$ *be the number of times that* $< FV = \underline{FV} >$ *appears in DB. Then the* database BPA *refers to* $m_{\Theta[DB]} : 2^{\Theta[FV]} \mapsto [0, 1]$, *where* $m_{\Theta[DB]}(FV = \underline{FV}) = freq(FV = \underline{FV}) \div nDB$. *The corresponding BoE generated* $\{\Theta[DB], \mathcal{F}_{\Theta[DB]}, m_{\Theta[DB]}\}$ *is called the* database BoE.

This is indeed a valid BPA in the sense of Definition 2.1. One may compute the corresponding *database belief* and *database plausibility* functions via $\mathrm{Bl}_{\Theta[DB]}(FV = \underline{FV}) = \sum_{G:G \subseteq \underline{FV}} m_{\Theta[DB]}(G)$ and $\mathrm{Pl}_{\Theta[DB]}(FV = \underline{FV}) = \sum_{G:G \cap \underline{FV} \neq \emptyset} m_{\Theta[DB]}(G)$.

4. BELIEF ITEMSET TREE (BIT)

This section introduces our most important result—a data structure that we refer to as the *belief itemset tree (BIT).* It can be thought of as a generalization for handling ambiguities of the *itemset tree* proposed by Kubat, et al.[8] While the itemset tree provides a convenient tool for targeted querying of associations, the BIT enables one to respond to targeted querying of the belief of an arbitrary feature vector. We first need

DEFINITION 4.1 (PROJECTION AND EXTENSION). *Consider the feature vector* $< FV = \underline{FV} >$. *Then, for* $k = \overline{0, nF}$, *define the following: (i) The* k-projection *of* FV, *denoted by* $FV^{\downarrow k}$, *is the* k*-attribute feature vector* $< FV^{\downarrow k} = \underline{FV}^{\downarrow k} >$ *where* $\underline{FV}^{\downarrow k} = [\underline{F}_1, \ldots, \underline{F}_k]$; *by convention,* $\underline{FV}^{\downarrow 0} = \emptyset$. *(ii) The* k-extension *of* FV, *denoted by* $FV^{\uparrow k}$, *is the* nF*-attribute feature vector* $< FV^{\uparrow k} = \underline{FV}^{\uparrow k} >$ *where* $\underline{FV}^{\uparrow k} = [\underline{FV}^{\downarrow k}, \Theta[F_{k+1}], \ldots, \Theta[F_{nF}]]$; $\underline{FV}^{\uparrow 0}$ *is called the* completely ambiguous feature vector *for which* $\underline{F}_j = \Theta[F_j]$, $\forall j = \overline{1, nF}$.

DEFINITION 4.2 (ANCESTORS, PARENT AND CHILD). *Given the feature vector* $< FV = \underline{FV} >$, *generate the* nF*-attribute feature vectors* $< FV_1 = \underline{FV}_1 >$ *and* $< FV_2 = \underline{FV}_2 >$ *where*

$$\underline{FV}_1 = \underline{FV}^{\uparrow k} = [\underline{F}_1, \ldots, \underline{F}_k, \Theta[F_{k+1}], \ldots, \Theta[F_{nF}]]; \quad \underline{FV}_2 = \underline{FV}^{\uparrow \ell} = [\underline{F}_1, \ldots, \underline{F}_\ell, \Theta[F_{\ell+1}], \ldots, \Theta[F_{nF}]].$$

Then we say the following: (i) $< FV_1 = \underline{FV}_1 >$ *is an* ancestor *of* $< FV_2 = \underline{FV}_2 >$ *if* $k < \ell$. *(ii)* $< FV_1 = \underline{FV}_1 >$ *is a* parent *of* $< FV_2 = \underline{FV}_2 >$, *or equivalently,* $< FV_2 = \underline{FV}_2 >$ *is a* child *of* $< FV_1 = \underline{FV}_1 >$, *if* $< FV_1 = \underline{FV}_1 >$ *is an ancestor of* $< FV_2 = \underline{FV}_2 >$ *and* $k = \ell - 1$; *we denote this as* $< FV_1 = \underline{FV}_1 > \lhd < FV_2 = \underline{FV}_2 >$ *or* $< FV_2 = \underline{FV}_2 > \rhd < FV_1 = \underline{FV}_1 >$.

Clearly, a given feature vector can have multiple children; it can have only one parent though.

Belief Itemset Tree (BIT). We are now in a position to introduce

DEFINITION 4.3 (BELIEF ITEMSET TREE (BIT)). *The* belief itemset tree (BIT) *of the database DB consists of a set of nodes arranged in a tree structure of* $nF + 1$ *hierarchical levels. The set of nodes in level* k, *referred to as the* level-k nodes, *is denoted by* $N^{(k)}$, $k = \overline{0, nF}$; *level-*nF *nodes are also called the* leaf nodes *of the BIT. Level-*k *nodes* $N^{(k)}$ *correspond to the* k*-projections generated by only those feature vectors in DB.*

(i) For $k = \overline{1, nF}$, *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* $n_{\ell_k}^{(k)} = n_{\ell_k}^{(k)}\left(\underline{F}_k, n_{\ell_{k-1}}^{(k-1)}\right)$, $\ell_k \in \mathcal{I}(n_{\ell_{k-1}}^{(k-1)})$, *where* $n_{\ell_k}^{(k)}$ *is the* k*-projection* $n_{\ell_k}^{(k)} : < FV = \underline{FV} > \mapsto < FV^{\downarrow k} = \underline{FV}^{\downarrow k} = [\underline{FV}^{\downarrow(k-1)}, \underline{F}_k] >$ *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 feature 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* $\underline{F}_k$ *and a parameter* $freq(n_{\ell_k}^{(k)}) > 0$ *which denotes the frequency of occurrence of feature 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 feature 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 of a given database DB and, in general, update the current BIT with an incoming arbitrary feature vector, appears in Table 1.

Table 1. Algorithm for updating the level-k of the BIT with the incoming arbitrary feature vector $< FV = \underline{FV} >$

```
1   Insert (FeatureVector FV,Level k) {
2   generate the k-projection FV^{↓k} = [F_1, ..., F_{k-1}, F_k];
3   if ∃ a 'branch' n_{ℓ_{k-1}}^{(k-1)} (F_{k-1}, .) ◁ n_{ℓ_k}^{(k)} (F_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)}) + 1;
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)}(F_k, n_{ℓ_{k-1}}^{(k-1)}) ▷ n_{ℓ_{k-1}}^{(k-1)} (F_{k-1}, .) with freq(n_L^{(k)}) = 1;
8     if k < nF then InsertNewTree (n_L^{(k)}) }
9   Insert (FV,k + 1) };
10
11  InsertNewTree (Node N) {
12  create a new branch leading from N;
13  insert an empty node at its termination };
```

Example. Consider the set of feature vectors given in Table 2 where $\Theta[F] \equiv \Theta[F_i] = \{1, 2, 3\}$, $\forall i = \overline{1,5}$. The corresponding BIT implied by Definition 4.3 is shown in Fig. 1.

Table 2. A set of feature vectors containing attribute ambiguities; the FoD of each attribute is identical with $\Theta[F] \equiv \Theta[F_i] = \{1, 2, 3\}$, $\forall i = \overline{1,5}$.

Record R_i	$< F_1 = \underline{F}_1 >$	$< F_2 = \underline{F}_2 >$	$< F_3 = \underline{F}_3 >$	$< F_4 = \underline{F}_4 >$	$< F_5 = \underline{F}_5 >$
1	1	1	2	3	1
2	1	1	3	2	1
3	1	2	2	1	2
4	1	1	(2,3)	1	3
5	2	2	1	2	3
6	2	1	2	3	(1,2,3)
7	(1,2)	1	(1,2)	1	2
8	2	(1,2)	1	3	2
9	(1,2,3)	(1,2)	1	3	2
10	(1,2,3)	(1,2,3)	1	2	1

BIT and Database BoE. The relationship between the BIT and the database BoE is given by

THEOREM 4.4. *Consider the feature record* $< FV = \underline{FV} >$ *in a database DB of cardinality nDB. Then, (i) all feature records that have an identical k-projection* $\underline{FV}^{\downarrow k}$ *map to one and only one level-k node* $n_{\ell_k}^{(k)}$; *(ii)* $freq(n_{\ell_k}^{(k)})$ *denotes the frequency of occurrence of the feature vectors in DB that have an identical k-projection* $\underline{FV}^{\downarrow k}$; *and (iii) the database BPA in Definition 3.1 of DB is given by* $m_{\Theta[FV]}(FV = \underline{FV}) = \frac{freq(FV=\underline{FV})}{nDB} = \frac{freq(n_{\ell_{nF}}^{(nF)})}{nDB}$.

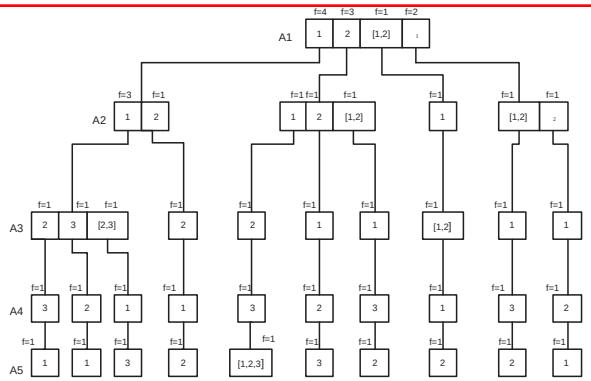

Figure 1. BIT corresponding to the database in Table 2.

Proof. Let $\underline{FV} = [\underline{F}_1, \ldots, \underline{F}_{k-1}, \underline{F}_k, \ldots, \underline{F}_{nF}] \Longrightarrow \underline{FV}^{\downarrow k} = [\underline{F}_1, \ldots, \underline{F}_{k-1}, \underline{F}_k] = [\underline{FV}^{\downarrow(k-1)}, \underline{F}_k]$. Now apply Definition 4.3: $\underline{FV}^{\downarrow k}$ maps to $n_{\ell_k}^{(k)} = n_{\ell_k}^{(k)}\left(\underline{F}_k, n_{\ell_{k-1}}^{(k-1)}\right)$, via $n_{\ell_{k-1}}^{(k-1)} = n_{\ell_{k-1}}^{(k-1)}\left(\underline{F}_{k-1}, n_{\ell_{k-2}}^{(k-2)}\right)$, and so on. Clearly, all feature vectors with an identical k-projection must trace this path. Moreover, a feature vector with a different k-projection must trace a different path. This proves (i). Claim (ii) is obvious from how $\text{freq}\,(n_{\ell_k}^{(k)})$ is defined. Claim (iii) follows from (i) and (ii) when one notes that the nF-projection of a feature vector is itself. □

Thus the BIT can be used as a convenient tool for computing the database belief function. See Table 3.

Table 3. Algorithm for computing the database belief of the arbitrary proposition $< G = \underline{G} >$

```
1   ComputeBelief (Proposition G, Level k) {
2   generate the k-projection G^{↓k} = [G_1, ..., G_k];
3   B = 0;
4   for each ℓ_k ∈ I(n_{ℓ_{k-1}}^{(k-1)}) {
5     go to node n_{ℓ_k}^{(k)} = n_{ℓ_k}^{(k)}(F_k, n_{ℓ_{k-1}}^{(k-1)});
6     if G_k ⊇ F_k then {
7       if G_j = Θ[F_j], ∀j = k+1, nF then
8         B = B + freq(n_{ℓ_k}^{(k)})/nDB;
9       else {
10        go to child node n_{ℓ_{k+1}}^{(k+1)} = n_{ℓ_{k+1}}^{(k+1)}(F_{k+1}, n_{ℓ_k}^{(k)});
11        B = B + ComputeBelief (G, k+1) } } }
12  return B };
```

Example. Suppose we are interested in the beliefs of the following two queries from the database in Table 2: $Q1 = \text{Bl}_{\Theta[DB]}([(1,2), 1, \Theta[F_3], \Theta[F_4], \Theta[F_5]])$ and $Q2 = \text{Bl}_{\Theta[DB]}([\Theta[F_1], 1, 2, 2, \Theta[F_4], 2])$. Fig. 2 shows how these are handled by the algorithm in Table 3.

5. ASSOCIATION RULE MINING (ARM)

An association rule is an expression of the form $R_{ant} \Longrightarrow R_{con}$, where the *antecedent* R_{ant} and *consequence* R_{con} are sets of attributes. The task of discovering association rules is to generate all association rules with certain support and confidence measures above given thresholds.[3] Over the years, several methods that use association rule mining (ARM) for classification have been developed.[3,9] An association rule used for this purpose consists of an antecedent that is a subset of the feature attribute vector and a consequence that consists of a class label. These methods however require that each antecedent attribute and consequence class label be a singleton, e.g., $< FV_1 = 3 > \wedge < FV_2 = 4 > \Longrightarrow < CL = 1 >$. They are therefore incapable of discovering rules such as $< FV_1 = (2,3) > \wedge < FV_2 = 4 > \Longrightarrow < CL = (1,2) >$ where an attribute and the class label are ambiguous.

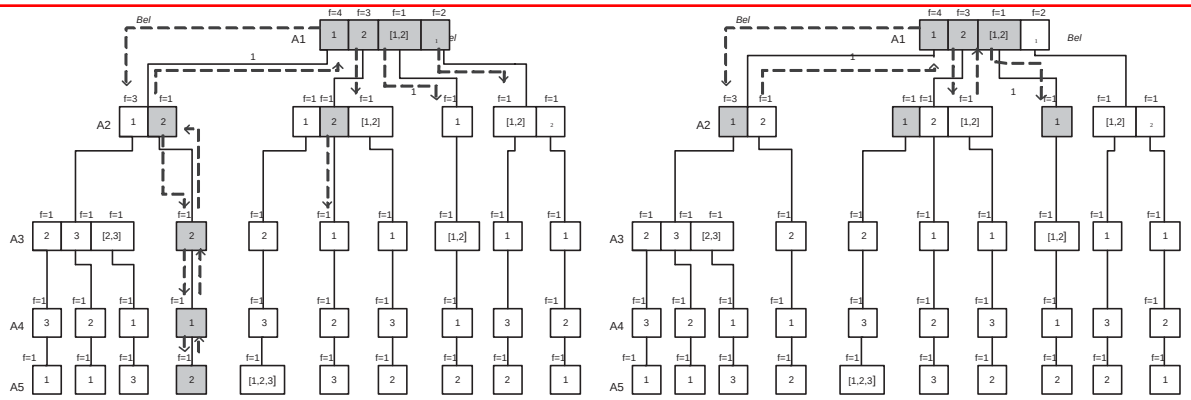

Figure 2. Propagation of the two queries $Q1$ and $Q2$ in the BIT corresponding to the database in Table 2. Note: 4 nodes are visited to respond to $Q1$; 6 nodes are needed to respond to $Q2$.

How does one perform ARM in such situations? What are suitable support and confidence measures? For example, neither of the feature records $[2, 4, \Theta[F_3], \ldots, \Theta[F_{nF}]]$ and $[3, 4, \Theta[F_3], \ldots, \Theta[F_{nF}]]$ may pass the minimum support threshold on its own. But, when considered together as $[(2, 3), 4, \Theta[F_3], \ldots, \Theta[F_{nF}]]$, they may prove successful thus potentially generating a useful rule. If one considers $\{(2, 3)\}$ as a new feature attribute that is distinct from either $\{2\}$ or $\{3\}$, it would not account for the facts $\{2\} \in \{(2, 3)\}$ and $\{3\} \in \{(2, 3)\}$. To perform ARM in the presence of such data imperfections and discover rules that allow ambiguities, we propose an approach based on DS belief theory.

Frequent Itemsets. We first need

DEFINITION 5.1 (k-ITEMSET AND SUPPORT). *Consider* $< FV = \underline{FV} = [\underline{F}_1, \ldots, \underline{F}_{nF}] >$ *in a database* DB. *(i)* $< FV = \underline{FV} >$ *is referred to as a* k-itemset *if exactly* $nF - k$ *of its feature attributes are equal to their corresponding FoDs; the set of* k*-itemsets is denoted by* I_k *(by convention,* I_0 *contains the completely ambiguous feature vector only). (ii)* $Bl_{\Theta[DB]}(FV = \underline{FV})$ *is referred to as the* support of $< FV = \underline{FV} >$ *and is denoted by* $Sp[FV = \underline{FV}]$. *(iii)* $< FV = \underline{FV} > \in I_k$ *is said to be* frequent *if* $Sp[FV = \underline{FV}] \geq minSp[DB]$, *where* $minSp[DB]$ *denotes a user-defined minimum support threshold; the set of frequent* k*-itemsets is denoted by* FI_k.

LEMMA 5.2. *Consider* $< FV = \underline{FV} > \in I_k$, $< FV_1 = \underline{FV}_1 > \in I_m$ *and* $< FV_2 = \underline{FV}_2 > \in I_n$ *in a database* DB. *Then, (i)* $< FV = \underline{FV} > = < FV_1 = \underline{FV}_1 > \cap < FV_2 = \underline{FV}_2 >$ *for some* $< G_1 = \underline{G}_1 > \in I_{k_1}$ *and* $< G_2 = \underline{G}_2 > \in I_{k_2}$ *(not necessarily in* DB*) where* $k_1 + k_2 = k$; *and conversely, (ii)* $< FV_1 \cap FV_2 = \underline{FV}_1 \cap \underline{FV}_2 > \in I_\ell$ *where* $\ell \leq k$.

Proof. These claims are obvious when one notices Definition 5.1. □

The support measure in Definition 5.1 should be contrasted with the probabilistic measure that is customary in ARM.[3] The justification for its use is that the belief of an itemset captures the 'support' available for all its subset itemsets (including itself). Table 4 shows an efficient **apriori**-like algorithm[3] that is capable of generating frequent itemsets from an ambiguous database. The steps in the algorithm are as follows:

First, the entire database DB is scanned to find all its 1-itemsets (line #3). The frequent itemsets that would be eventually generated from these would only consist of those feature vectors that are present in DB. Such a strategy may prevent us from discovering other potentially important rules. For example, suppose all the feature vectors in DB possess singletons modeling their first attributes. Then, with line #3 alone, it would be impossible to create potentially interesting relationships with an antecedent having an ambiguous first attribute!

To discover such relationships, we form ambiguous attributes by combining the attributes of *non-frequent* 1-itemsets obtained in line #3 (line #6). If these newly formed 1-itemsets turn out to be frequent, they are included in FI_1 (line #8). For example, if $\underline{FV}_1 = [1, \Theta[F_2], \ldots, \Theta[F_{nF}]] \notin FI_1$ and $\underline{FV}_2 = [2, \Theta[F_2], \ldots, \Theta[F_{nF}]] \notin FI_k$, we would form the 1-itemset $\underline{FV}_1 \cup \underline{FV}_2 = [(1, 2), \Theta[F_2], \ldots, \Theta[F_{nF}]]$ (line 6). Now, since Bl(.) is a monotonic function,[6] it is quite possible that $\underline{FV}_1 \cup \underline{FV}_2 \in FI_1$. This strategy however may create 'redundant' frequent 1-itemsets. For example, suppose DB contains the records $\underline{FV}_1$ and $\underline{FV}_1 \cup \underline{FV}_2$. Then, due to the monotonicity

Table 4. Algorithm for generating large itemsets. Note: FI_{all} denotes the set of all frequent itemsets

```
1   GenerateLargeItemSets() {
2   FI_all = ∅; FI_1 = ∅;
3   I_1 = GenerateOneItemsets(DB);
4   for each itemset I ∈ I_1
5     if Bl_Θ[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_Θ[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_Θ[DB](I) ≥ minSp then FI_{n+1} = FI_{n+1} ∪ I;
17    n = n + 1 }
18  return FI_all };
```

of $\mathrm{Bl}(\centerdot)$, $\underline{FV}_1 \in FI_1 \implies \underline{FV}_1 \cup \underline{FV}_2 \in FI_k$. Since $\underline{F}_{1,1} \subseteq (\underline{F}_{1,1}, \underline{F}_{1,2})$, for discovering rules that have $\underline{F}_{1,1}$ in its antecedent, it is sufficient to retain only $\underline{FV}_1 \cup \underline{FV}_2$, i.e., $\underline{FV}_1 \in FI_1$ can be considered redundant. In other words, any frequent 1-itemset that is a strict subset of another frequent 1-itemset can be pruned (line #9).

The next step is to form 2-itemsets from the frequent 1-itemsets thus obtained (line #13). Not all 1-itemset pairs produce 2-itemsets (see Lemma 5.2. For example, $\underline{FV}_1 \cap \underline{FV}_2 \notin I_2$ while $\underline{FV}_1 \cap \underline{FV}_3 \in I_2$ where $\underline{FV}_3 = [\Theta[F_1], 3, \Theta[F_3], \ldots, \Theta[F_{nF}]] \in I_1$. The frequent 2-itemsets are those that pass *minSp* value (line #16). This same procedure is followed to generate frequent k-itemsets in general. This algorithm calls for the computation of $\mathrm{Bl}_{\Theta[DB]}(FV = \underline{FV})$ often (line #16) and this is where the BIT comes in extremely handy.

Rule Generation. For classification purposes, each association rule antecedent R_{ant} would have a set of attributes while the consequence R_{con} would have the corresponding class label. The frequent itemsets we have generated however are allowed to possess attribute value ambiguities; the class label is also allowed to be ambiguous. We now need to utilize an appropriate belief theoretic measure to indicate the support we place on each rule. DS conditional notions are ideal candidates for such a measure.[10] We prefer the Fagin-Halpern (FH) conditional notions[11] for this purpose because they can be considered the more natural extensions of the Bayesian conditional.[12,13] Hence, we define the confidence in a rule via the lower and upper bounds

$$\begin{aligned}
\underline{Cf} &\equiv \mathrm{Bl}_{\Theta[DB]}(R_{con}|R_{ant}) = \mathrm{Bl}_{\Theta[DB]}(R_{ant} \cap R_{con}) \div \left[\mathrm{Bl}_{\Theta[DB]}(R_{ant} \cap R_{con}) + \mathrm{Pl}_{\Theta[DB]}(R_{ant} \setminus R_{con})\right]; \\
\overline{Cf} &\equiv \mathrm{Pl}_{\Theta[DB]}(R_{con}|R_{ant}) = \mathrm{Pl}_{\Theta[DB]}(R_{ant} \cap R_{con}) \div \left[\mathrm{Pl}_{\Theta[DB]}(R_{ant} \cap R_{con}) + \mathrm{Bl}_{\Theta[DB]}(R_{ant} \setminus R_{con})\right]. \qquad (3)
\end{aligned}$$

Now we select only those rules that meets this $\underline{Cf}$ threshold; denote this rule set via $\mathcal{R}_{ARM}$.

6. CLASSIFICATION

We now use the rule set developed in Section 5 to arrive at a classifier. Classification is the task of assigning a class label for a new incoming data instance. Since the training data set itself contains both attribute and class label ambiguities, an incoming data instance can be classified into either a single class ("hard" decision) or multiple classes ("soft" decision).

Rule BPA and Rule Discount Factor. Consider the following partition of the rule set $\mathcal{R}_{ARM}$: $\mathcal{R}_{ARM} = \bigcup_{k=1}^{K} R^{(k)}$ where the antecedent of *all* rules in $R^{(k)}$ is $\underline{FV}_k$ and $\underline{FV}_i \neq \underline{FV}_j$, $i \neq j$. We now propose

DEFINITION 6.1 (RULE BPA AND RULE DISCOUNT FACTOR). *For the partition $R^{(k)}$, define the following: (i)* Rule BPA: $m^{(k)}_{\Theta[CL]} : 2^{\Theta[CL]} \mapsto [0.1]$ *s.t.* $m_{\Theta[CL]}(\underline{CL}_i) = m_{\Theta[DB]}(\underline{CL}_i|\underline{FV}_k)$; *(ii)* Rule Discount Factor: $d^{(k)} = \left[1 + Ent^{(k)}\right]^{-1} \left[1 + \log\left[|\underline{FV}_k|\right]\right]^{-1}$, *where* $Ent^{(k)} = \sum_{C \subseteq \Theta[CL]} m^{(k)}_{\Theta[CL]}(C) \cdot \log\left[m^{(k)}_{\Theta[CL]}(C)\right]$.

Kulasekere, et al.[13] provide an iterative technique for computing 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 BPA calculated reaches a preset threshold close to unity; the remaining mass is assigned to the complete set $\Theta[CL]$. The quantity $\text{Ent}^{(k)}$ required for the rule discount factor accounts for the 'uncertainty' in the rule about the class label while the term $1/(1 + \log[|\underline{FV}_k|])$ accounts for the ambiguity in the rule antecedent. Hence, $d^{(k)}$ can be considered a measure of the total uncertainty in the rule. Now, each rule in $\mathcal{R}_{ARM}$ can be identified via the triple $[\underline{FV}_k, m^{(k)}_{\Theta[CL]}, d^{(k)}]$.

When classifying an incoming feature vector $\underline{FV}$, the classifier first needs to find a set of rules $R_{\underline{FV}} \subseteq R_{ARM}$ that 'match' $\underline{FV}$. Different criteria may be used for this. We used $R_{\underline{FV}} = \left\{[\underline{FV}_i, m^{(i)}_{\Theta[CL]}, d^{(i)}] : \underline{FV} \subseteq \underline{FV}_i\right\}$; if there is no $\underline{FV}_i$ s.t. $\underline{FV} \subseteq \underline{FV}_i$, we use the classification algorithm in Zhang, et al.[5] with the distance measure $0.5\left[|\underline{FV} \setminus \underline{FV}_i| + |\underline{FV}_i \setminus \underline{FV}|\right]$. The rule BPAs of the rules in $R_{\underline{FV}}$ are then combined using the DRC with the rule discount factor taken into account.[6]

Decision Criterion. Having computed this BPA generated from $R_{\underline{FV}}$, we make a decision 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[14] 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. This strategy sometimes leads to decisions that are too ambiguous; in such cases, one can restrict the maximum cardinality of the decision to be a pre-determined number and use the maximum belief criterion.

7. EXPERIMENTAL RESULTS

This section summarizes our experimental results.

UCI Repository of Machine Learning Databases. The proposed algorithm has been tested against several databases from the UCI repository.[15] To demonstrate its performance in the presence of imperfect data, 'noise' was introduced into the attribute values by changing the value of a randomly picked attribute of a data instance to its neighboring values. Table 5 compares the proposed algorithm with C4.5.[16] Note that we have used the scoring mechanism suggested by Zhang, et al.,[5] for comparing the two algorithms in the presence of soft decisions.

Table 5. Classification accuracy of C4.5 and the proposed algorithm for different attribute noise levels in 05 UCI databases.

Database	Algorithm	No noise	5% noise	10% noise	15% noise	20% noise
Monks	C4.5	79.37	76.21	75.60	71.90	63.90
	Proposed	80.52	76.52	75.44	73.83	71.29
Nursery	C4.5	97.26	91.72	86.06	81.50	76.35
	ARM	92.30	91.23	87.88	83.75	82.40
Diabetes	C4.5	68.78	63.92	63.77	58.33	54.98
	ARM	68.64	67.01	66.58	65.61	61.62
Car	C4.5	92.23	82.80	77.65	68.92	63.60
	ARM	91.05	83.32	81.10	73.98	72.10
Iris	C4.5	95.65	95.10	80.00	74.57	72.00
	ARM	98.28	96.08	90.68	82.11	77.00

Application in a Threat Assessment Scenario. Consider an airport area that has been divided into two security zones {Z0,Z1} where Z1 has a higher priority. Suppose Z0 and Z1 have been divided into 6 and 4 grid locations respectively. Assume that there are 4 different types of passengers {P0,P1,P2,P3} where P0 poses no danger while P1, P2 and P3 pose increasing levels of danger.

Suppose there are two experts {E1,E2} whose opinions are being sought to allocate threat levels to a set of training instances. E1 allocates a threat level as $T_1 = X_{01} + X_{02} + X_{03} + X_{11} + X_{12} + X_{13}$ by taking into account the number of different passenger types in *each zone;* E2, on the other hand, allocates a threat level as $T_2 = X_1 + X_2 + X_3$ by taking into account the different passenger types in the *entire airport.* We use the following heuristic rules to mimic the opinions of E1 and E2:

$$X_{0j} = \begin{cases} 2^{j-1}N_{0j}/6, & \text{if } N_{0j} < 3; \\ 2^{j-1}, & \text{otherwise;} \end{cases} \quad X_{1j} = \begin{cases} 2^{2+j}N_{1j}/4, & \text{if } N_{1j} < 2; \\ 2^{2+j}, & \text{otherwise;} \end{cases} \quad X_j = \begin{cases} 2^{j+2}N_j/10, & \text{if } N_j < 3; \\ 2^{j+2}, & \text{otherwise.} \end{cases} \tag{4}$$

Here, X_{ij} denotes the 'contribution' of passenger type Pj, $j = \overline{1,3}$, located in zone Zi, $i = \overline{0,1}$, to E1's opinion; N_{ij} denotes the number of passenger type Pj, $j = \overline{1,3}$, located in zone Zi, $i = \overline{0,1}$; X_j denotes the 'contribution' of passenger type Pj, $j = \overline{1,3}$, to E2's opinion; N_j denotes the number of passenger type Pj, $j = \overline{1,3}$, in the entire airport. Finally, T_1 and T_2 are each linearly mapped to 4 threat classes {T1,T2,T3,T4}.

We generated a corresponding data set where each data instance has 10 attributes and an allocated threat class. The 10 attributes describe the passenger type at each of the 10 locations (6 in Z0 and 4 in Z1). Ambiguities in attribute values can arise due to imperfections of information channels/sensors used for determining passenger type. The ambiguities in the threat class are due to the difference in opinions of E1 and E2. Fig. 3 compares the proposed algorithm with C4.5. For C4.5, the ambiguous values are treated as distinct values. The increase

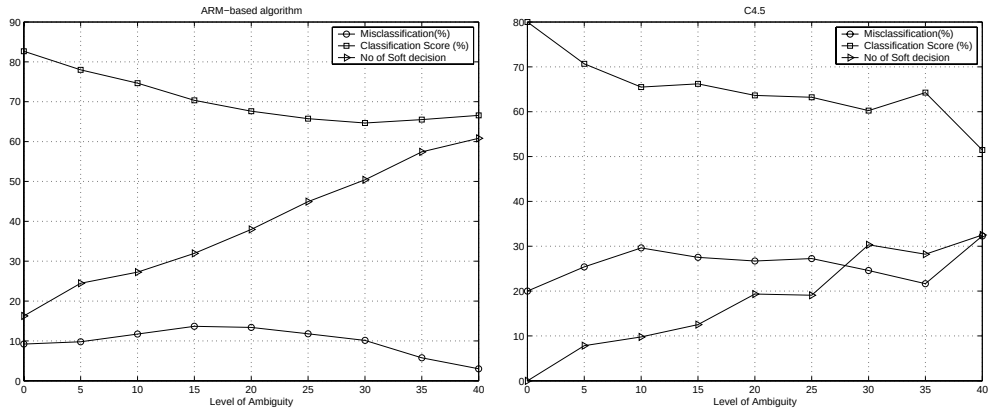

Figure 3. Comparison of the proposed algorithm with C4.5.

in the percentage of the soft decisions with increasing levels of ambiguities, while keeping the mis-classification rate low, is an indication of the robustness of the proposed algorithm. The decrease in the classification score[5] with increasing ambiguity is due to the increasing number of soft decisions.

8. CONCLUSION

Data imperfections can be considered one of the most challenging hurdles confronting the use of various algorithms developed within the realm of computer science and engineering to other application domains.[2] As a way of modeling and accounting for such imperfections, DS belief theretic notions have attracted considerable attention.[1] Nevertheless, how one may reduce the associated computational burden remains an extremely active area of research. The work presented in this paper proposes a new data structure—the *belief itemset tree (BIT)*—that can effectively represent a database that may contain feature and class label ambiguities and respond to *belief* queries. The proposed ARM algorithm uses the BIT to efficiently extract both ambiguous and non-ambiguous rules; they form the basis on which classification of an incoming feature vector is performed.

The attribute ambiguities considered in this algorithm could vary depending on the application. For example, in the ***GenerateAmbiguousItemsets(.)*** routine in Table 4, depending on the type of attribute that we combine to form ambiguous 1-itemsets, different approaches may be employed. When an attribute takes continuous values, it has to be discretized so that it fits into the rule mining framework. Suppose such an attribute has been

discretized into the eight levels $\{1, 2, \ldots, 8\}$. It is highly unlikely that we will have rules containing ambiguous attribute values such as $(1, 6)$, $(2, 8)$, etc. On the other hand, ambiguities such as $(1, 2)$, $(3, 4)$, etc., where adjoining levels appear together, are more likely. The same would be true with an attribute taking nominal values with an inherent temporal component, e.g., $\Theta_{Season} = \{Spring, Summer, Autumn, Winter\}$. Another question that arises when generating ambiguous itemsets is the maximum number of distinct values of an attribute that need to be combined. This again depends on the application and nature of the dataset available.

This work is restricted to the case of data imperfections that can be modeled via BoEs having only one focal element, viz., data ambiguities. An interesting and critically important research problem would be to find ways to remove this restriction so that other types of data imperfections (e.g., probabilistic and possibilistic types) can also be accommodated within the basic framework developed here.

ACKNOWLEDGMENTS

The support of NSF Grants IIS-0325260 (ITR Medium) and EAR-0323213 is gratefully acknowledged.

REFERENCES

1. A. Motro and P. Smets, eds., *Uncertainty Management in Information Systems: From Needs to Solutions*, Kluwer Academic Publishers, Boston, MA, 1997.
2. National Science Foundation, "IDM 2004 Workshop," 2004.
3. 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)*, pp. 487–499, (Santiago de Chile, Chile), Sept. 1994.
4. B. Liu, W. Hsu, and Y. M. Ma, "Integrating classification and association rule mining," in *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pp. 80–86, (New York, NY), Aug. 1998.
5. 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.
6. G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, NJ, 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* **18**, pp. 189–223, 1996.
8. 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* **15**, pp. 1522–1534, Nov./Dec. 2003.
9. M. Deshpande and G. Karypis, "Using conjunction of attribute values for classification," in *Proc. International Conference on Information and Knowledge Management (CIKM'02)*, pp. 356–364, (McLean, VA), Nov. 2002.
10. H. Xu and P. Smets, "Reasoning in evidential networks with conditional belief functions," *International Journal of Approximate Reasoning* **14**, 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., pp. 347–374, Elsevier Science, New York, NY, 1991.
12. P. Walley, *Statistical Reasoning with Imprecise Probabilities*, Chapman and Hall, London, UK, 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* **36**, pp. 75–108, Apr. 2004.
14. 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* **17**, pp. 905–919, July 1996.
15. C. L. Blake and C. J. Merz, "UCI repository of machine learning databases," 1998.
16. J. R. Quinlan, *C4.5: Programs for Machine Learning*, Representation and Reasoning Series, Morgan Kaufmann, San Francisco, CA, 1993.