Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Information Fusion xxx (2004) xxx–xxx

INFORMATION FUSION

www.elsevier.com/locate/inffus

On aggregating belief decision trees

Patrick Vannoorenberghe *

Perception Systèmes Information, Laboratoire PSI, FRE 2645 CNRS, Université de Rouen, UFR des Sciences, Place Emile Blondel, 76821 Mont Saint Aignan Cedex, France

Received 16 January 2003; received in revised form 10 January 2004; accepted 11 January 2004

Abstract

Recently, Belief Decision Trees, induction methods based on the Dempster–Shafer Theory of evidence (or belief functions theory), have been introduced. Such trees give the possibility to interpret each decision rule and quantify the uncertainty on the prediction. In addition, due to its main ability to represent different kinds of knowledge (from total ignorance to full knowledge), Dempster–Shafer theory allows us to process training sets whose labeling has been specified with belief functions. This paper investigates the aggregation of such belief decision trees with several machine learning techniques including bagging and randomization. Bagging builds a decision rule by aggregating outputs of several classifiers optimized from different learning sets. Randomization consists in building several classifiers where each classifier is trained using a given number of variables drawn randomly from the original set of features. Several simulations show how such techniques can improve the performance of belief decision trees.

Keywords: Decision trees; Belief functions; Uncertain labels; Bagging; Randomization

1. Introduction

A pattern recognition problem consists in assigning an input pattern to a class, given a learning set $\mathscr{L}$ composed of n individual patterns with known classification. Each pattern in $\mathscr{L}$ is represented by a p-dimensional feature vector $\mathbf{x}_i$ and its corresponding class label ω_i. In recent years, the decision tree approach has become increasingly popular in the Machine Learning community [5,25,28] to solve this problem. Such techniques give the possibility to interpret each decision rule in terms of individual features. Unfortunately, decision trees are very sensitive to small perturbations in the learning set $\mathscr{L}$ as remarked by several authors [4,8]. Multiple Classifiers Systems (MCS) have been proposed to overcome this problem [29]. Systems that combine the individual outputs of a set (pool/commitee/ensemble/team) of classifiers are known to be more accurate than the best classifier in the set. In this context, two ways have been investigated: classifier fusion [14,20] and resampling methods [2,4,8]. These last ones, which include Bagging, Boosting, Randomization and variants, generate multiple classifiers by manipulating the training data given to the learning algorithm. They are known to decrease the variance of the classification rule and are well adapted to learning algorithms such as decision trees [9,11].

Recently, several induction methods based on the Dempster–Shafer Theory (DST) of evidence [6,12] have been introduced. Due to their links with belief function theory and decision trees, these approaches have been called Belief Decision Trees (BDT's). Due to its main ability to represent different kinds of knowledge (from total ignorance to full knowledge), DST allows us to process learning sets whose labeling has been specified with belief functions (which can included probabilistic, possibilistic or imprecise labels). In [33], we introduce another multi-class generalization of the method in [6], which allows us to handle the most general case in which each example is labeled by a general belief function for K-class problems. In this approach, an impurity measure, based on a total uncertainty criterion, is used to

* Tel.: +023-514-6552; fax: +023-514-6618.
E-mail address: patrick.vannoorenberghe@univ-rouen.fr (P. Vannoorenberghe).

1566-2535/$ - see front matter © 2004 Published by Elsevier B.V.
doi:10.1016/j.inffus.2004.01.001

grow the tree and has the advantage to define simultaneously the pruning strategy. In addition, BDT's give the possibility to quantify the uncertainty on the prediction and process uncertain labels.

In this paper, we illustrate how the use of Bagging and Randomization seems to allow further reductions of classification error rates in the context of Belief Decision Trees. Basic concepts of belief function theory are first briefly introduced (see Section 2). The approach based on the induction of BDT's and the way to handle uncertain labels in this framework are described in Section 3. The problem of aggregating BDT's using Bagging and Randomization is introduced in the Section 4 and finally illustrated with experimental results in the Section 5.

2. Belief functions and uncertainty

2.1. Belief functions

Let $\Omega = \{\omega_1, \ldots, \omega_k, \ldots, \omega_K\}$ be a finite space, called the *frame of discernment*. A belief function *bel* is a function from 2^Ω to $[0,1]$ defined as:

$$bel(A) = \sum_{\emptyset \neq B \subseteq A} m(B) \quad \forall A \subseteq \Omega, \tag{1}$$

where m, called basic belief assignment (bba), is a function from 2^Ω to $[0,1]$ verifying

$$\sum_{A \subseteq \Omega} m(A) = 1.$$

Each subset $A \subseteq \Omega$ such as $m(A) > 0$ is called a focal element of m. Functions m and *bel* are in one-to-one correspondence [30] and can be seen as two facets of the same piece of information. [1] From this definition, several concepts have been derived in order to manage uncertainty encoded by belief functions. Discounting, aggregating, coarsening, refinement are very powerful tools which allow for us to manipulate uncertain information. In this paper, we adopt the Transferable Belief Model (TBM) point of view defined by Smets [32]. Based on rationality arguments developed in this model, Smets proposes to transform m into a probability function p_m on Ω (called the *pignistic* probability function) defined for all $\omega_k \in \Omega$ as:

$$p_m(\omega_k) = \sum_{A \ni \omega_k} \frac{m(A)}{|A|}, \tag{2}$$

where $|A|$ denotes the cardinality of $A \subseteq \Omega$. In this transformation, the mass of belief $m(A)$ is distributed equally among the elements of A. This probability function is used for decision making at the pignistic level (as opposed to the credal level where beliefs are held and combined) of the TBM.

[1] In the sequel and when necessary, the notation $m^\Omega[data]$ will be used to denote a bba on domain Ω based on observed $[data]$.

2.2. Uncertainty measures

Because a belief function can represent several kinds of knowledge, it constitutes a rich and flexible way to represent uncertainty. As mentioned by Klir [18], a belief function can model two different kinds of uncertainty: non-specificity and conflict. A measure of non-specificity, which generalizes the Hartley measure [15] to belief functions, was introduced by Dubois and Prade [10]. It is defined as:

$$N(m) = \sum_{A \subseteq \Omega} m(A) \log_2 |A|. \tag{3}$$

Since focal elements of probability measures are singletons, non-specificity is null for probability functions, and it is maximal ($\log_2 |\Omega|$) for the vacuous belief function ($m(\Omega) = 1$). Several measures of conflict, viewed as generalized Shannon entropy measure, have also been introduced [19,1]. A well-known measure defined within the framework of TBM is *discord*, defined as:

$$D(m) = -\sum_{A \subseteq \Omega} m(A) \log_2 p_m(A). \tag{4}$$

Another measure of conflict has been introduced in [26,27] which leads to the definition:

$$H(m) = -\sum_{A \subseteq \Omega} m(A) \log_2 m(A). \tag{5}$$

These measures have several desirable properties including continuity, subadditivity and required range ($[0, f(|\Omega|)]$). In this paper, we propose to use H instead D for the conflict measure because it has a unique maximum contrary to D. Finally, a measure U_λ of total uncertainty can be defined using a linear combination of N and H:

$$U_\lambda(m) = (1-\lambda)N(m) + \lambda H(m), \tag{6}$$

where $\lambda \in [0,1]$ is a coefficient. This criterion has the same properties as $N(m)$ and $H(m)$ and can give more importance to one of these two measures according to the value of λ. The choice of this parameter is not theoretically justified (Klir recommends taking $\lambda = 0.5$). In the sequel, we shall see that it can be used as a regularization parameter in the context of Belief Decision Trees.

2.3. Combination's operator

Let now assume that we have two pieces of evidence expressed by m_1 and m_2 representing two distinct items concerning the truth value of A. These two bba's can be aggregated with the conjunctive operator $\odot$, yielding to

a unique belief function corresponding to bba $m_{\textcircled{\cap}}$ defined as:

$$m_{\textcircled{\cap}}(A) = (m_1 \textcircled{\cap} m_2)(A) \triangleq \sum_{B \cap C = A} m_1(B) m_2(C) \quad \forall A \subseteq \Omega. \tag{7}$$

This rule is sometimes referred to as the (unnormalised) Dempster's rule of combination. If necessary, the normality assumption $m(\emptyset) = 0$ may be recovered by dividing each mass by a normalisation coefficient. The resulting operator which is knows as Dempster's rule denoted by $m_{\oplus}$ is defined as:

$$(m_1 \oplus m_2)(A) \triangleq \frac{(m_1 \textcircled{\cap} m_2)(A)}{1 - m(\emptyset)} \quad \forall \emptyset \neq A \subseteq \Omega, \tag{8}$$

where the quantity $m(\emptyset)$ is called the degree of conflict between m_1 and m_2 and can be computed using:

$$m(\emptyset) = (m_1 \textcircled{\cap} m_2)(\emptyset) = \sum_{B \cap C = \emptyset} m_1(B) m_2(C). \tag{9}$$

The use of Dempster's rule is possible only if m_1 and m_2 are not totally conflicting, i.e., if there exist two focal elements B and C of m_1 and m_2 satisfying $B \cap C \neq \emptyset$. This rule verifies some interesting properties (associativity, commutativity, non-idempotence). In Dempster's rule, the normalisation step, which redistributes conflicting belief masses to non-conflicting ones, can introduce unexpected results when evidence conflicts due to a sensibility problem [17]. In [22], authors proposed the weighted operator which has been created to overcome this problem. The idea of the weighted operator is to distribute the conflicting belief mass $m(\emptyset)$ on some subsets of Ω according to additional knowledge. More precisely, a part of the mass $m(\emptyset)$ is assigned to a subset $A \subseteq \Omega$ according to a weighting factor denoted w. This weighting factor can be a function of the considered subset A and belief functions $\mathbf{m} = \{m_j, j = 1, \ldots, J\}$ which are involved in the combination and have caused the conflict. This idea is formalized in the following definition of the Weighted Operator.

Definition 1 (*Weighted Operator*). Let $\mathbf{m} = \{m_j, j = 1, \ldots, J\}$ be the set of belief functions defined on Ω to be combined. The combination of the belief functions $\mathbf{m}$ with the weighted operator, denoted $\textcircled{\cap}_w$, is defined as:

$$m_{\textcircled{\cap}_w}(\emptyset) \triangleq w(\emptyset, \mathbf{m}) \cdot m(\emptyset), \tag{10}$$

$$m_{\textcircled{\cap}_w}(A) \triangleq m_{\textcircled{\cap}}(A) + w(A, \mathbf{m}) \cdot m(\emptyset) \quad \forall A \neq \emptyset. \tag{11}$$

In the definition of the weighted operator $\textcircled{\cap}_w$, the first term, $m_{\textcircled{\cap}}$, corresponds to the conjunctive rule of combination. The second one is the part of the conflicting mass assigned to each subset A and added to the conjunctive term. The symbol $\textcircled{\cap}_w$ has been chosen to highlight these two aspects. Weighting factors $w(A, \mathbf{m}) \in [0, 1]$ are coefficients which depend on each subset $A \subseteq \Omega$ and on the belief functions $\mathbf{m}$ to be combined. They must be constrained by:

$$\sum_{A \subseteq \Omega} w(A, \mathbf{m}) = 1 \tag{12}$$

so as to respect the property that the sum of mass functions must be equal to 1. In order to completely define this operator, we need additional information to choose the values of w which allow to have a particular behaviour of the operator. In [22], authors show that it is easy to define a family of weighted operators according to the choice of weights w. An example has been proposed by the authors in the case of a pattern recognition problem. It consists in optimising the weighting factors values according to some criteria minimisation. This application gives interesting performance when compared to Dempster's rule. While this operator is not associative as remarked by Haenni [23], an n-ary version exists. In this context, combining several input belief functions simultaneously can be a practical substitute for associativity in many real world applications.

2.4. Partially labeled data

In the context of partially labeled data, the available learning set can be written in the form:

$$\mathscr{L} = \{(\mathbf{x}_i, m_i^{\Omega}) \mid i = 1, \ldots, n\}, \tag{13}$$

where m_i^{Ω} is a bba defined on Ω and represents the knowledge on the label of the ith example. This belief function can represent different forms of label including:

- hard labels (HL); represented by a belief function with one focal element singleton in Ω, $m(\{\omega_2\}) = 1$ for example,
- imprecise labels (IL); when the mass is assigned to a subset of the frame. The example $m(\{\omega_1, \omega_2\}) = 1$ where it is impossible to choose between ω_1 and ω_2 according to the lack of information,
- probabilistic labels (PrL); when the belief function corresponds to a probability function, $m(\{\omega_1\}) = m(\{\omega_2\}) = 0.2$, $m(\{\omega_3\}) = 0.6$,
- possibilistic (PoL) labels; A possibility measure is known to be formally equivalent to a consonant belief function, i.e., a belief function with focal elements ordered by set inclusion [7], $m(\{\omega_3\}) = 0.7$, $m(\{\omega_1, \omega_3\}) = 0.2$ and $m(\{\Omega\}) = 0.1$ with $\{\omega_3\} \subset \{\omega_1, \omega_3\} \subset \Omega$.

Table 1 illustrates an example of these evidential labels on a three-class frame. Unlabeled samples can be encoded using the vacuous belief function m_v^{Ω} defined as $m_v^{\Omega}(\Omega) = 1$.

Table 1
Example of uncertain labeling with belief functions (HL: Hard Label, IL: Imprecise Label, PrL: Probabilistic Label, PoL: Possibilistic Label, UL: Unknown Label)

$A \subseteq \Omega$	HL	IL	PrL	PoL	UL
$\{\omega_1\}$	0	0	0.2	0	0
$\{\omega_2\}$	1	0	0.6	0	0
$\{\omega_1,\omega_2\}$	0	1	0	0	0
$\{\omega_3\}$	0	0	0.2	0.7	0
$\{\omega_1,\omega_3\}$	0	0	0	0.2	0
$\{\omega_2,\omega_3\}$	0	0	0	0	0
Ω	0	0	0	0.1	1

3. Belief decision trees

In this paper, we only consider the Belief Decision Tree's approach introduced by Denœux and Skarstein-Bjanger [6] and extended to multi-class problems by Vannoorenberghe et al. [33]. Due to its main ability to represent different kinds of knowledge (from total ignorance to full knowledge), DST allows us to process training sets whose labeling has been specified with belief functions (see Section 3.1). An impurity measure, based on a total uncertainty criterion, is used to grow the tree and has the advantage to define simultaneously the pruning strategy Section 3.2. Finally, we present in paragraph Section 3.3, a multi-class generalization of the method introduced in [6] which allows us to handle the most general case in which each example is labeled by a general belief function [33].

3.1. Principle

A decision tree is a specific graph in which each node is either a decision node or a leaf node. To each decision node is associated a test based on attribute values, and a node has two or more successors (depending on the number of possible outcomes of the test). The most commonly used decision tree classifiers are binary trees which use a single feature at each node with two outcomes. In [6], the problem of handling uncertain labels is solved for two-class problems. In this context, the available learning set is given by: $\mathscr{L} = \{(\mathbf{x}_i, m_i^{\Omega}) \mid i = 1,\ldots,n\}$ where m_i^{Ω} is defined on $\Omega = \{\omega_1,\omega_2\}$ and represents the knowledge on the label of the ith example. The belief function $m^{\Omega}[t]$ at node t is then derived from the $n(t)$ belief functions m_i^{Ω} (by induction using the Dempster's rule of combination) and becomes:

$$m^{\Omega}[t](\{\omega_1\}) = \sum_{(j,k)|j+k\leqslant n(t)} \alpha_{jk}\frac{j}{j+k+1}, \tag{14}$$

where α_{jk} are coefficients which depend only on the functions m_i^{Ω}. Similar expressions for $m^{\Omega}[t](\{\omega_2\})$ and $m^{\Omega}[t](\Omega)$ can be obtained. In Eq. (14), $n(t)$ is the total number of examples reaching the node t. These equations are derived from a theoretical result on credal inference presented by Smets in [31]. Demonstrations concerning the extension to the more general case of belief functions have been proposed by Denœux and can be found in [6,33]. They are recalled in Appendix A of this paper.

3.2. Induction

For each node t, an impurity measure is computed from the belief function $m^{\Omega}[t]$ using the total uncertainty measure:

$$U_{\lambda}(t) = (1-\lambda)N(m^{\Omega}[t]) + \lambda H(m^{\Omega}[t]). \tag{15}$$

This impurity measure is used at node t to choose a candidate split s which divides t into two nodes t_L and t_R. The goodness of a split s is defined as a decrease in impurity by:

$$\Delta U_{\lambda}(s,t) = U_{\lambda}(t) - (p_L U_{\lambda}(t_L) + p_R U_{\lambda}(t_R)), \tag{16}$$

where p_L and p_R are, respectively, the proportions of examples reaching t_L and t_R. The best split $\hat{s}$ is chosen by testing all possible splits for each attribute.

One of the advantages of this technique is that the tree growing can be controlled using parameter λ. In fact, according to the value of λ, it is possible to give more importance to the non-specificity term which penalizes small nodes. Optimizing this parameter by cross-validation allows us to build smaller trees, thus avoiding overtraining. Unfortunately, this induction method is only available for two-class problems but can be generalized as explained in the next section.

3.3. Dichotomous approach for K-class problems

A standard way of handling a K-class problems is to decompose it into several two-class subproblems. One way to do this is to train K binary classifiers, each classifier attempting to discriminate between one class ω_k and all other classes. When the learning set is of the form $\mathscr{L} = \{(\mathbf{x}_i, m_i^{\Omega}) \mid i = 1,\ldots,n\}$, where m_i^{Ω} is a bba defined on Ω, this approach implies transforming each bba m_i^{Ω} originally defined on Ω into a bba defined on the two-class coarsened frame. For each coarsening, a tree is grown, and the resulting K trees are combined using the averaging operator. More precisely, let us denote by Ω_k the following coarsening of Ω:

$$\Omega_k = \{\{\omega_k\}, \overline{\{\omega_k\}}\}, \tag{17}$$

where $\overline{\{\omega_k\}}$ denotes the complement of $\{\omega_k\}$. Each bba m_i^{Ω} defined on Ω may be transformed into a bba $m_i^{\Omega_k}$ on Ω_k using the following transformation:

$$m_i^{\Omega_k}(\{\omega_k\}) = m_i^{\Omega}(\{\omega_k\}), \tag{18}$$

$$m_i^{\Omega_k}(\overline{\{\omega_k\}}) = \sum_{A \subseteq \overline{\{\omega_k\}}} m_i^{\Omega}(A), \tag{19}$$

$$m_i^{\Omega_k}(\Omega_k) = 1 - m_i^{\Omega_k}(\{\omega_k\}) - m_i^{\Omega_k}(\overline{\{\omega_k\}}). \tag{20}$$

Each of the K coarsenings thus leads to a training set $\mathscr{L}_k = \{(\mathbf{x}_i, m_i^{\Omega_k}) \mid i = 1, \ldots, n\}$, which is used to build a decision tree.

At the testing step, we obtain, for each input vector $\mathbf{x}$, K bba's $m_{\mathbf{x}}^{\Omega_k}$, each defined on a distinct coarsening Ω_k. Each of these bba's can be trivially carried back to Ω using the transformation:

$$m_{\mathbf{x},k}^{\Omega}(\{\omega_k\}) = m_{\mathbf{x}}^{\Omega_k}(\{\omega_k\}), \tag{21}$$

$$m_{\mathbf{x},k}^{\Omega}(\Omega \setminus \{\omega_k\}) = m_{\mathbf{x}}^{\Omega_k}(\overline{\{\omega_k\}}), \tag{22}$$

$$m_{\mathbf{x},k}^{\Omega}(\Omega) = m_{\mathbf{x}}^{\Omega_k}(\Omega_k). \tag{23}$$

Because information sources are not independent, Dempster's rule of combination cannot be used to combine the bba's $m_{\mathbf{x},k}^{\Omega}$, $k = 1, \ldots, K$. An alternative is to use the weighted operator as explained in Section 2.3, which leads to:

$$m_{\mathbf{x}}^{\Omega} = \circledcirc_{\hat{w}}\, m_{\mathbf{x},k}^{\Omega}, \tag{24}$$

where $\hat{w}$ are coefficients to be optimized. This dichotomous approach of Belief Decision Trees allows us to quantify the uncertainty of the prediction of vector $\mathbf{x}$ (the belief function $m_{\mathbf{x}}^{\Omega}$ itself), process learning sets whose labeling has been specified with belief functions (m_i^{Ω} for each learning example) and is available for K-class pattern recognition problems. In the Section 4, we investigate the aggregation of such BDT's in a Multiple Classifier System.

4. Aggregating BDT's

Several learning techniques, often called resampling methods, including Bagging, Boosting, Randomization and variants have been proposed by several authors [2,4,8]. They generate multiple classifiers by manipulating the training data given to the learning algorithm. They are known to decrease the variance of the classification rule and are well adapted to decision trees which are very sensitive to learning data.

4.1. Resampling methods for the multiple BDT's system

Bagging (Bootstrap aggregating) is a popular approach that builds a decision rule by aggregating outputs of several classifiers optimized from different learning sets [4]. More precisely, the technique consists in building B bootstrap replicates of the learning set by drawing n examples uniformly (with replacement) from $\mathscr{L}$. After the optimization on each bootstrap sample $\mathscr{L}_b$, decisions are generally combined by a majority vote.

Another technique, sometimes called Randomization, consists in building R classifiers where each classifier is trained using a given number p' of variables drawn randomly from the original set of features [2,16]. This technique is often referred as Multiple Feature Subset and has been used for several classification rules [3,21]. Bagging and randomization are known to decrease the variance of the classification rule [4] and are well adapted to decision trees due to the instability of such classifiers, which are very sensitive to small perturbations in the learning set. In this paper, we investigate these two ways to build more stable classifiers.

In order to take advantages of these techniques, we also propose to mix in a random bagging architecture. This last one consists in constructing ensemble of M classifiers where each one is trained using a bootstrap replicate with p' variables drawn randomly.

4.2. Fusion of BDT's

In the context of belief decision trees, the classifiers are aggregated at the belief function level using the approach proposed in [13]. For bagging, outputs of each classifier m_b^{Ω} are aggregated into a resulting belief function m_B^{Ω} defined as:

$$m_B^{\Omega}(A) = \circledcirc_{\hat{w}}\, m_b^{\Omega}(A) \quad \forall A \subseteq \Omega. \tag{25}$$

Similar expressions m_R^{Ω} and m_M^{Ω} are respectively obtained for randomization and the mixed system. In this paper, we choose to aggregate the belief functions with the weighted operator where the coefficients $\hat{w}$ are optimized by minimization of a error criterion. As mentioned by François et al. [13], the sum operator can be used in this context by its main properties: idempotency (averaging B times a same belief function leads to the function itself), commutativity and linearity (between credal and pignistic levels). Using the weighted operator for aggregating such rules at the belief function level allows us to optimize the classifiers fusion in order to take into account uncertainty of each classifier. The multiple classifier system has a more certain prediction and seems to be richer than aggregating classification rules at decision level.

4.3. Optimization technique

Performance assessment is an important issue in the design of a multiple classifier system. In a decision-theoretic setting, this problem is formalized by considering a set of actions $\mathscr{A}$, and a loss function $L : \mathscr{A} \times \Omega \mapsto \mathbb{R}$, where $L(\alpha, \omega)$ is the loss incurred if one selects action α and the true state of nature is ω. Typically, each action in $\mathscr{A}$ corresponds to the choice of a class, and the loss is one for misclassification, and 0 for correct classification.

The performance of a classifier $c : \mathbb{R}^d \mapsto \mathscr{A}$ can then be measured by taking the expectation of $L(c(\mathbf{x}), \omega)$ over both $\mathbf{x}$ and ω. This expectation is usually estimated by a sample average over a test set composed of n' examples $(\mathbf{x}_i, \omega_i)$, $i = 1, \ldots, n'$:

$$E = 1 - \frac{1}{n'} \sum_{i=1}^{n'} \sum_{\omega_k \in \Omega} p_{\hat{m}_i}(\omega_k) u_i^k, \tag{26}$$

where $u_i^k = 1$ if pattern i belongs to class k and $u_i^k = 0$ otherwise. In this equation, $p_{\hat{m}_i}$ corresponds to the pignistic probability estimated by the classifier for pattern i. In our case, this framework needs to be extended because the output of a BDT classifier is a belief function. We then need to define the loss associated to an output bba $\hat{m}$ when the true state of nature is ω. A solution was proposed in [6] which postulates the following loss function:

$$L(\hat{m}, m) = 1 - \sum_{A \subseteq \Omega} m(A) p_{\hat{m}}(A), \tag{27}$$

where $\hat{m}$ is the output bba produced by the classifier, and m is a bba that quantifies the uncertainty concerning the true state of nature ω. A nice property of this loss function is that, when $m(\Omega) = 1$, $L(\hat{m}, m) = 0$ whatever $\hat{m}$, which seems reasonable. Optimizing the parameters of the algorithm in minimizing this loss function allows for an increase in the performance of the multiple BDT's system.

5. Simulations

For the following simulations, a first learning set $\mathscr{L}_1$ is generated using three classes ($\Omega = \{\omega_1, \omega_2, \omega_3\}$) containing 50 bidimensional vectors each. Each vector $\mathbf{x}$ from class k was generated by first drawing a vector $\mathbf{z}$ from a Gaussian $f(\mathbf{z}|\omega_k) \sim \mathscr{N}(\mu_k, \Sigma_k)$, and applying a non-linear transformation $\mathbf{z} \mapsto \mathbf{x} = \exp(0.3\mathbf{z})$ to obtain non-Gaussian data. The means of the three Gaussian distributions were taken as: $\mu_1 = (-1, -1)'$, $\mu_2 = (1, 2)'$, $\mu_3 = (-1.5, 2)'$ and the variance matrices were of the form $\Sigma_k = D_k A D_k'$ with

$$A = \begin{pmatrix} \sqrt{3} & 0 \\ 0 & \sqrt{3}/3 \end{pmatrix} \quad D_q = \begin{pmatrix} \cos\theta_q & -\sin\theta_q \\ \sin\theta_q & \cos\theta_q \end{pmatrix}$$

with $\theta_1 = \pi/3$, $\theta_2 = \pi/2$, $\theta_3 = -\pi/3$. For each vector $\mathbf{x}_i$ with corresponding class label ω_i in the learning set, imprecise labeling was obtained as follows. We selected randomly a subset A such that $\omega_i \in A \subseteq \Omega$ and assigned the whole mass to this subset:

$$m_i^{\Omega}(A) = 1. \tag{28}$$

Another learning set $\mathscr{L}_2$ was generated using two classes with Gaussian distributions of dimension $p = 10$ with mean vectors $\mu_1 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0)'$ and $\mu_2 = (3, 3, 3, 3, 3, 3, 3, 3, 3, 3)'$. Covariance matrix Σ_k were diagonal matrix of the form

$$\Sigma_1 = diag(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),$$
$$\Sigma_2 = diag(1, 2, 3, 4, 5, 5, 4, 3, 2, 1),$$

where $diag(V)$ is a diagonal matrix with V as main diagonal. For vector $\mathbf{x}_i$ with known class ω_i in the learning set $\mathscr{L}_2$, imprecise labeling was generated in transferring a random part κ of belief on subset Ω using:

$$m_i^{\Omega}(\{\omega_i\}) = \kappa, \tag{29}$$

$$m_i^{\Omega}(\Omega) = 1 - \kappa. \tag{30}$$

Simulating this kind of labeling is useful if we consider that the expert knowledge from which m_i^{Ω} has been derived is not fully reliable. In this context, the coefficient κ represents some form of metaknowledge about the expert reliability, which could not be encoded initially in m_i^{Ω}. In the sequel, this set has been called Gaussian dataset $\mathscr{L}_2$.

5.1. Qualitative analysis

We first demonstrate the qualitative effects of bagging and randomization on the multiple classifier system using the first learning set $\mathscr{L}_1$. Fig. 1 shows the maximum pignistic probabilities as grey values for the BDT and the bagged version of the BDT. For each vector $\mathbf{x}$, this value is obtained using

$$\max_{\omega_k \in \Omega} p_{\hat{m}_{\mathbf{x}}}(w_k), \tag{31}$$

where $\hat{m}_{\mathbf{x}}$ corresponds to the output belief function of the multiple classifier system. As can be seen from this figure, the bagged version of BDT's seems to be more precise in the learning phase. In fact, higher values of the pignistic probabilities [2] in regions without ambiguity can be observed which leads probably to a more robust decision in the testing step. The bagged version is built from $B = 200$ classifiers and parameters are optimized using a cross-validation set.

The number B of classifiers for the bagged version has been fixed heuristically as mentioned in [8]. An experiment shows that only a few number of classifiers can be sufficient to reduce error rates significantly as we can see in the left part of Fig. 2. This plot shows the test error rate vs. the number B of classifiers implied in the bagged version for the dataset $\mathscr{L}_1$. In the sequel, we choose to adjust heuristically the value $B = 20$ classifiers, as the small improvement achieved by higher values is not worth the computation cost.

For the randomized version of the BDT, we evaluate the evolution of the test error rate vs. the effect of the number p' of variables selected for the Gaussian learning

[2] Left and right figures have different scales!

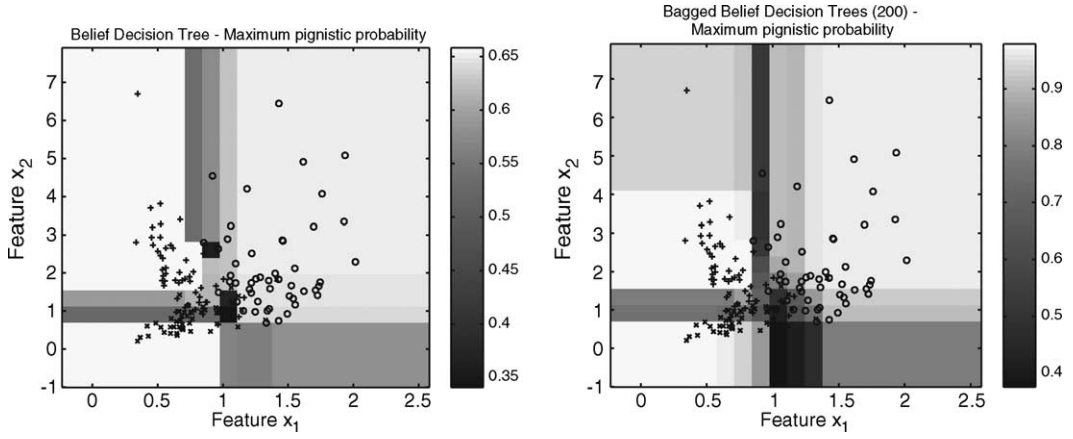

Fig. 1. Maximum pignistic probabilities—BDT (left)—Bagged version of the BDT (right), (learning samples with $\times = \omega_1$, $\circ = \omega_2$, $+ = \omega_3$).

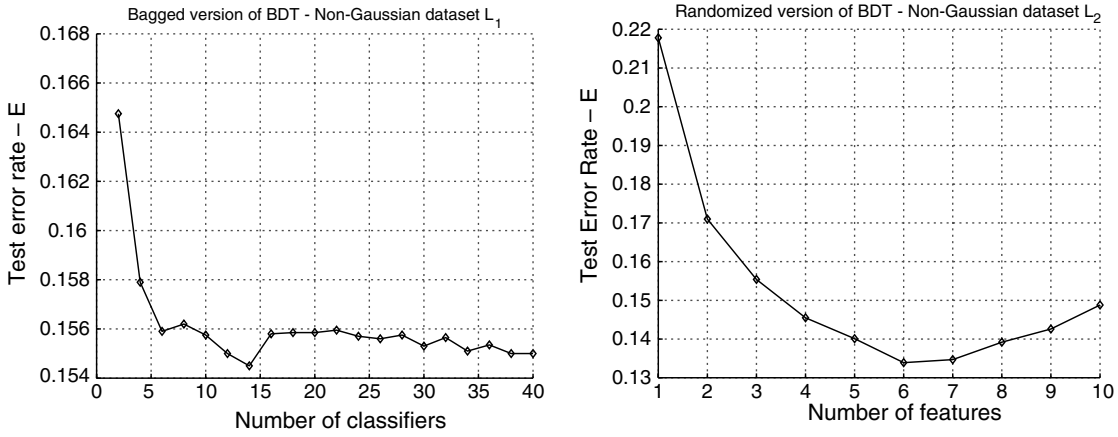

Fig. 2. Test error rate vs. number of classifiers for the bagged version (left) and number of features for the randomized version (right) of Belief Decision Trees.

data $\mathscr{L}_2$. The result is represented in the right part of Fig. 2. This simulation shows that an appropriate choice of the value of p' is decisive for a minimal error rate. To have a good idea for selecting the value of p', the proposed algorithms have been evaluated using several databases drawn from the UCI Repository [3] [24]. Although these data only contain hard class labels, the experiments are reported in Table 2 and illustrate that an optimal number of features can be chosen to improve the performance of the classification rule. The first column of this table shows the error rate of a single BDT averaged using ten cross-validations. The second column presents the minimal error rate obtained for the number of features indicated in brackets. Finally, the last column is the analogous error rate obtained for the value $p' = \mathrm{Int}\left(\frac{p}{2}\right) + 1$ where Int() is the integer part. As design guidelines, we preconize to select this number of features $p' = \mathrm{Int}\left(\frac{p}{2}\right) + 1$. This choice is used for the rest of the simulations of this paper.

Table 2
UCI databases: errors rates for the BDT, the minimal error rate and for the value $p' = \mathrm{Int}\left(\frac{p}{2}\right) + 1$

Database	BDT	Optimal number	$p' = \mathrm{Int}\left(\frac{p}{2}\right) + 1$
DIABETES(8)	0.269	0.242(4)	0.250
GLASS(9)	0.325	0.232(5)	0.232
IRIS(4)	0.052	0.045(2)	0.045
SONAR(60)	0.256	0.201(17)	0.226
VEHICLE(18)	0.296	0.252(10)	0.252
WAVEFORM(21)	0.260	0.211(11)	0.211

Number of features used to perform the error rates are in brackets.

5.2. Empirical comparison

In order to illustrate the gain obtained by aggregating BDT's, a learning set $\mathscr{L}_2$ was generated using two classes with Gaussian distributions. For vector $\mathbf{x}_i$ with known class ω_i in the learning set $\mathscr{L}_2$, imprecise labeling

[3] The UCI Repository databases are well known in the machine learning community.

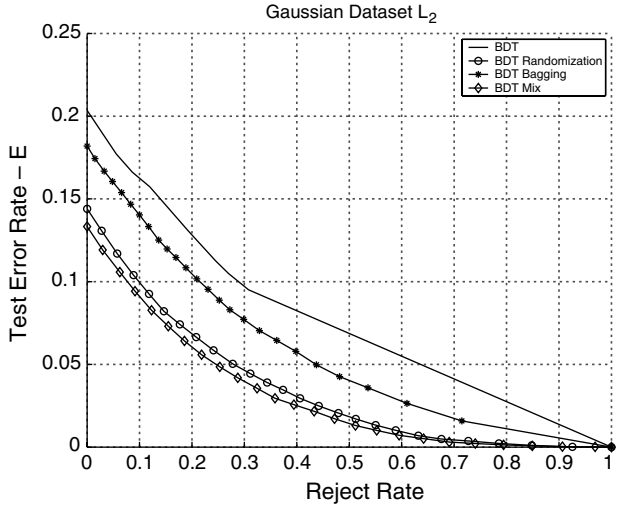

Fig. 3. Test error rate vs. reject rate for the initial, bagged, randomized and mixed methods.

was generated as previously mentioned (Eqs. (29) and (30)). To compare the performances of the four methods, experiments were repeated ten times with independent training sets and test sets $\mathscr{T}$ with 15,000 samples. We chose heuristically, as previously mentioned 20 classifiers ($B = R = 20$) and $p' = 6$ features for this dataset. Parameters of the algorithm were optimized by cross-validation. Fig. 3 shows the compared performances of the four methods: the initial classifier, the bagged, randomized and the mixed classification rules. This last version of the MCS is obtained using bootstrap replicates of the learning set $\mathscr{L}_2$ and a given number p' of variables drawn randomly from the original set of features.

As expected, bagging and randomization have the effect of improving the performances of the decision tree classifier. With this simulation, the gain obtained by bagged and randomized versions of classical decision trees (already observed by several authors) is generalized to BDT's and uncertain labels. Finally, mixing bagging and randomization seems to be the ideal methodology for such decision trees.

5.3. Real world application

The method described in this paper was applied to real data concerning acoustic emission testing of pressure vessels. [4] The data consists in 37 examples described by 27 features. Each training pattern corresponds to a cluster of acoustic emission signals, and belongs to one of three classes: minor, major, or critical source. Two different experts were asked to assess, for each training example, the degree of possibility that this example belongs to each class, resulting in two different possibility distributions for each example. Fig. 4 displays the data in a two-dimensional subspace of the feature space, together with possibilistic labels respectively provided by each expert. As mentioned previously, a possibility measure is known to be formally equivalent to a consonant belief function. Hence, possibilistic labels are a special case of evidential labels considered in this paper. Three training sets were considered: the data labeled by each of the two experts ($\mathscr{E}_1$ and $\mathscr{E}_2$), and the data $\mathscr{E}_{1+2}$ labeled by a conjunctive combination of the labels provided by the two experts (by taking the minimum of the possibility distributions, and normalizing). The possibilistic labels were transformed in belief functions and used as explained in Section 3.1.

For these experiments, we used 10-fold cross-validation to optimize the value of the parameters and to evaluate the performance of the proposed method. For each training set, we considered two learning strategies. In the first one, the possibilistic labels were transformed into hard labels by selecting only, for each example, the class with the highest possibility. In the second strategy, possibilistic labels are transformed into evidential labels and used for learning. The results are summarized in Tables 3 and 4 (strategy 1) and (strategy 2) for the four methods (initial BDT, bagged version, randomized version and mixed version). For strategy 1 (training with hard labels), the misclassification error rate E was used as a performance criterion. For strategy 2 (training with possibilistic labels), the criterion L defined in Eq. (27) was investigated.

According to these tables, we can see that:

- training with possibilistic labels tends to decrease the error rate, which is an indication that our method succeeds in using more refined information than just hard labels (similar results were reported in [6,7,33] with different data sets);
- combining the expert information tends to improve the results, whatever the method used. This shows that collecting information from several experts may be useful when the class of training patterns can only be assessed subjectively;
- bagged, randomized and mixed versions of BDT improve significantly the performances of the Belief Decision Tree.

6. Conclusion

In this paper, the aggregation of Belief Decision Trees with several machine learning techniques has been investigated. Such trees allow us to quantify the uncertainty of the prediction and process learning sets whose labeling has been specified with belief functions for K-

[4] These data were collected by the Centre Technique des Industries Mécaniques (CETIM) in Senlis, France.

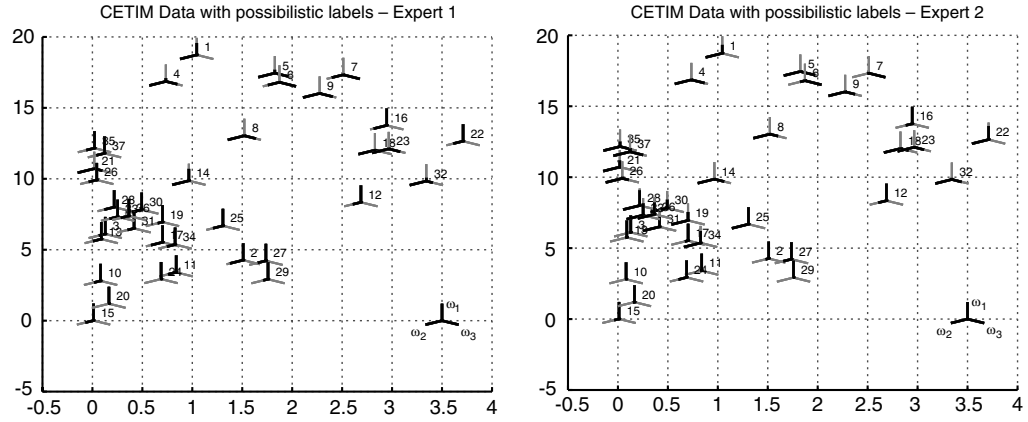

Fig. 4. The data in a two-dimensional subspace of the feature space. For each example, the three bars correspond to the three classes, and their lengths are proportional to the degrees of possibility given by expert 1 (left) and 2 (right).

Table 3
Strategy 1: Test error rates E; Learning with hard labels for the four methods

Data set	E	BAG	RAND	MIX
$\mathscr{E}_1$	0.32	0.29	0.28	0.25
$\mathscr{E}_2$	0.35	0.32	0.32	0.29
$\mathscr{E}_{1+2}$	0.33	0.29	0.29	0.28

Table 4
Strategy 2: Test errors rates based on loss function L (except first column); Learning with possibilistic labels four the four methods

Data set	E	L	BAG	RAND	MIX
$\mathscr{E}_1$	0.28	0.39	0.37	0.35	0.34
$\mathscr{E}_2$	0.29	0.41	0.40	0.40	0.33
$\mathscr{E}_{1+2}$	0.27	0.37	0.35	0.37	0.34

class pattern recognition problems. Bagged, randomized and mixed versions of these belief decision trees have been introduced in a general Multiple Classifiers System. Aggregation of the BDT'S is realized at the belief function level using a specific weighted operator. This allows us to take into account uncertainty of each classifier involved in the system. The proposed method seems to be richer than aggregating classification rules at decision level and allows us to have a more certain prediction. After a study on parameters values, we have demonstrated, with several simulations, that the use of Bagging and Randomization allows further reductions of classification error rates for different reject rates.

Acknowledgements

The author thanks Catherine Hervé and Cristel Rigault from CETIM for providing the acoustic emission data. Thierry Denœux is also gratefully acknowledged.

Appendix A

The aim of this Appendix A is not to give a detailed account of credal inference (i.e., statistical inference based on belief functions), but only to summarize a theoretical result obtained by Smets [31] and its extension given in [6,33]. The problem considered here is to derive the belief function concerning the outcome of a Bernoulli trial, having observed a sequence of past outcomes. For example, let us consider a coin toss game with the two events ω_1 (head) and ω_2 (tail) leading to the set $\Omega = \{\omega_1, \omega_2\}$. The available information consists in observed outcomes from n independent trials. Given that you have observed n_1 heads and n_2 tails (so, $n_1 + n_2 = n$), the belief function derived by Smets in [31] is defined as follows:

$$m^{\Omega}[n_1, n_2](\{\omega_1\}) = \frac{n_1}{n+1}, \tag{A.1}$$

$$m^{\Omega}[n_1, n_2](\{\omega_2\}) = \frac{n_2}{n+1}, \tag{A.2}$$

$$m^{\Omega}[n_1, n_2](\Omega) = \frac{1}{n+1}. \tag{A.3}$$

This belief function converges to the true probability when n tends to infinity. The method described in [31] can, in principle, be generalized to more than two outcomes. However, the calculations become quite cumbersome, and the counterparts of (A.1)–(A.3) in the general case are not available to date.

Let us suppose now that we have performed n independent Bernoulli experiments but that the outcomes could only be partially observed (for example, the urn experiment was observed at a distance, so that the results of some trials could only be partially observed). Let m_i^{Ω} be the basic belief assignment describing one's belief concerning the result of experiment i, and m^{Ω} the basic belief assignment quantifying one's belief regarding the

outcome of the next experiment. Based on the results in [31], it was shown in [6,33] that m^{Ω} is given by:

$$m^{\Omega}(\{\omega_1\}) = \sum_{(j,k)|j+k\leqslant n(t)} \alpha_{jk} \frac{j}{j+k+1},$$

$$m^{\Omega}(\{\omega_2\}) = \sum_{(j,k)|j+k\leqslant n(t)} \alpha_{jk} \frac{k}{j+k+1},$$

$$m^{\Omega}(\Omega) = \sum_{(j,k)|j+k\leqslant n(t)} \alpha_{jk} \frac{1}{j+k+1},$$

where α_{jk} is defined as:

$$\alpha_{jk} = \sum_{\{I_1,I_2,I_3\}} \left(\prod_{i_1\in I_1} m^{\Omega}_{i_1}(\{\omega_1\}) \times \prod_{i_2\in I_2} m^{\Omega}_{i_2}(\{\omega_2\}) \times \prod_{i_3\in I_3} m^{\Omega}_{i_3}(\Omega) \right),$$

where $\{I_1, I_2, I_3\}$ ranges over all partitions of $\{1,\ldots,n\}$ such that $|I_1| = j$ and $|I_2| = k$. Note that these expressions are similar to Eqs. (A.1)–(A.3) when the basic belief assignments m^{Ω}_i are derived from precise observations.

References

[1] J. Abellan, S. Moral, Completing a total uncertainty measure in Dempster–Shafer theory, International Journal of General Systems 28 (1999) 299–314.
[2] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: bagging, boosting and variants, Machine Learning 36 (1) (1999) 105–139.
[3] S.D. Bay, Nearest neighbor classification from multiple feature subsets, Intelligent Data Analysis 3 (3) (1999) 191–209.
[4] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.
[5] L. Breiman, J.H. Friedman, R. Olshen, C.J. Stone, Classification and regression trees, Wadsworth and Brooks/Cole, Monterey, CA, 1984.
[6] T. Denœux, M. Skarstein Bjanger, Induction of decision trees from partially classified data using belief functions, in: Proceedings of SMC'2000, Nashville, USA, IEEE, 2000, pp. 2923–2928.
[7] T. Denœux, L.M. Zouhal, Handling possibilistic labels in pattern classification using evidential reasoning, Fuzzy Sets ans Systems 122 (3) (2001) 47–62.
[8] T.G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomization, Machine Learning 40 (2) (2000) 139–157.
[9] P. Domingos, Why does bagging work? A Bayesian account and its implications, in: Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, AAAI Press, 1997, pp. 155–158.
[10] D. Dubois, H. Prade, A note on measures of specificity for fuzzy sets, International Journal of General Systems 10 (1985) 279–283.
[11] B. Efron, R. Tibshirani, An introduction to the bootstrap, in: Monographs on Statistics and Applied Probability, vol. 57, Chapman and Hall, New York, 1993.
[12] Z. Elouedi, K. Mellouli, Ph. Smets, Belief decision trees: theoretical foundations, International Journal of Approximate Reasoning 28 (2001) 91–124.
[13] J. Francois, Y. Grandvalet, T. Denoeux, J.-M. Roger, Resample and combine: an approach to improving uncertainty representation in evidential pattern classification, Information Fusion 4 (2) (2003) 75–85.
[14] J. Grim, J. Kittler, P. Pudil, P. Somol, Information analysis of multiple classifier fusion, in: J. Kittler, F. Roll (Eds.), Multiple Classifier Systems 2001, 2001, pp. 168–177.
[15] R.V.L. Hartley, Transmission of information, The Bell System Technical Journal 7 (3) (1928) 535–563.
[16] T.K. Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (8) (1998) 832–844.
[17] A. Jøsang, M. Daniel, P. Vannoorenberghe, Strategies for combining conflicting dogmatic beliefs, in: Sixth International Conference on Information Fusion, FUSION'2003, Cairns, Australia, July 2003, pp. 1133–1140.
[18] G.J. Klir, R.M. Smith, Recent developments in generalized information theory, International Journal of Fuzzy Systems 1 (1) (1999).
[19] G.J. Klir, M.J. Wierman, Uncertainty-Based Information, Physica-Verlag, Heidelberg, Germany, 1998.
[20] L.I. Kuncheva, A theoretical study on six classifier fusion strategies, IEEE Pattern Analysis and Machine Intelligence 24 (2) (2002) 281–286.
[21] P. Latinne, O. Debeir, C. Decaestecker, Different ways of weakening decision trees and their impact on classification accuracy of DT combination, in: Proceedings of the First International Workshop of Multiple Classifier Systems, MCS'2000, Cagliari, Italy, in: J. Kittler, F. Roli (Eds.), LCNS 1857, Springer, 2000, pp. 200–209.
[22] E. Lefevre, O. Colot, P. Vannoorenberghe, Belief functions combination and conflict management, Information Fusion 3 (2) (2002) 149–162.
[23] E. Lefevre, O. Colot, P. Vannoorenberghe, Reply to the comments of R. Haenni on the paper: Belief functions combination and conflict management, Information Fusion 4 (1) (2003) 63–65.
[24] P.M. Murphy, D.W. Aha, UCI Repository of Machine Learning Databases, University of California, Department of Information and Computer Science, Irvine, CA, 1994.
[25] S.K. Murthy, Automatic construction of decision trees from data: a multi-disciplinary survey, Data Mining and Knowledge Discovery 2 (4) (1998) 345–389.
[26] N.R. Pal, J.C. Bezdek, R. Hemasinha, Uncertainty measures for evidential reasoning I: a review, International Journal of Approximate Reasoning 7 (3/4) (1992) 165–183.
[27] N.R. Pal, J.C. Bezdek, R. Hemasinha, Uncertainty measures for evidential reasoning II: A new measure of total uncertainty, International Journal of Approximate Reasoning 8 (1) (1993) 1–16.
[28] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA, 1993.
[29] F. Roli, G. Giacinto, G. Vernazza, Methods for designing multiple classifier systems, in: J. Kittler, F. Roli (Eds.), Multiple Classifier Systems 2001, 2001, pp. 78–87.
[30] G. Shafer, A Mathematical Theory of Evidence, Princeton University Press, 1976.
[31] P. Smets, What is Dempster–Shafer's model, in: R.R. Yager, M. Fedrizzi, J. Kacprzyk (Eds.), Advances in the Dempster–Shafer Theory of Evidence, Wiley, 1994, pp. 5–34.
[32] Ph. Smets, R. Kennes, The transferable belief model, Artificial Intelligence 66 (2) (1994) 191–234.
[33] P. Vannoorenberghe, T. Denœux, Handling uncertain labels in multiclass problems using belief decision trees, in: Proceedings of IPMU'2002, Annecy, France, 2002, pp. 1919–1926.