A neural network classifier based on Dempster-Shafer theory

ARTICLE *in* IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS - PART A SYSTEMS AND HUMANS · APRIL 2000

Impact Factor: 2.18 · DOI: 10.1109/3468.833094 · Source: CiteSeer

Thierry Denœux
Université de Technologie de Compiègne
212 PUBLICATIONS **3,430** CITATIONS

A Neural Network Classifier Based on Dempster-Shafer Theory

Thierry Denœux

***Abstract*—A new adaptive pattern classifier based on the Dempster–Shafer theory of evidence is presented. This method uses reference patterns as items of evidence regarding the class membership of each input pattern under consideration. This evidence is represented by basic belief assignments (BBA's) and pooled using the Dempster's rule of combination. This procedure can be implemented in a multilayer neural network with specific architecture consisting of one input layer, two hidden layers and one output layer. The weight vector, the receptive field and the class membership of each prototype are determined by minimizing the mean squared differences between the classifier outputs and target values. After training, the classifier computes for each input vector a BBA that provides a description of the uncertainty pertaining to the class of the current pattern, given the available evidence. This information may be used to implement various decision rules allowing for ambiguous pattern rejection and novelty detection. The outputs of several classifiers may also be combined in a sensor fusion context, yielding decision procedures which are very robust to sensor failures or changes in the system environment. Experiments with simulated and real data demonstrate the excellent performance of this classification scheme as compared to existing statistical and neural network techniques.**

***Index Terms*—Data fusion, decision making, neural networks, pattern classification, pattern recognition, uncertainty.**

I. INTRODUCTION

SINCE the early 1980's, the Dempster–Shafer (D–S) theory of evidence has generated considerable interest in the Artificial Intelligence community. In contrast, applications in statistical pattern recognition have until recently been very limited. In [22] and [28], D–S theory was shown to provide a suitable framework for combining the results of several independent classifiers, and thereby improve classification accuracy. In [5], we introduced a k-nearest neighbor (k-NN) classifier based on evidence-theoretic principles. This method allows to compute a basic belief assignment (BBA) over the set Ω of classes [23], in such a way that the mass of belief assigned to the reference set varies between 0 (perfect knowledge) and 1 (complete ignorance), depending on the informativeness of the training data with respect to the class membership of the pattern under consideration. *Ambiguity* and *distance* reject options were introduced using the concepts of *lower* and *upper* expected losses [6]. Simulations with artificial and real-world data sets revealed a better performance of the proposed scheme as compared to the voting and distance-weighted k-NN procedures.

In this paper, an adaptive version of this evidence-theoretic classification rule is proposed. In this approach, the assignment of a pattern to a class is made by computing distances to a limited number of prototypes, resulting in faster classification and lower storage requirements. Based on these distances and on the degree of membership of prototypes to each class, BBA's are computed and combined using Dempster's rule. This rule can be implemented in a multilayer neural network with specific architecture consisting of one input layer, two hidden layers and one output layer. The weight vector, the receptive field and the class membership of each prototype are determined by minimizing the mean squared differences between the classifier outputs and target values.

The organization of this paper is as follows. Section II starts with a brief presentation of the D–S theory and its application to pattern classification. The new method is then introduced in Section III, and decision-theoretic issues (including outlier rejection and novelty detection) are examined in Section IV. Finally, Section V describes numerical experiments which demonstrate the advantages of the classification scheme proposed in this paper.

II. BACKGROUND

A. Evidence Theory

In this section, the main concepts underlying the D–S theory of evidence are briefly recalled, and some basic notation is introduced. For a more complete introduction, the reader is invited to refer to Shafer's original work [23], and to more up-to-date sources for reports of recent developments [14], [25], [29]. Note that the general term "D–S theory" in fact encompasses several distinct models of reasoning under uncertainty, including the theory of hints [14] and the transferable belief model (TBM) [25]. The nonprobabilistic view of the latter model will be adopted in this paper.

Let Ω be a finite set of mutually exclusive and exhaustive hypotheses, called the *frame of discernment*. A *basic belief assignment* (BBA) is a function m from 2^Ω to [0, 1] verifying

$$m(\emptyset) = 0 \tag{1}$$

$$\sum_{A \subseteq \Omega} m(A) = 1. \tag{2}$$

For any $A \subseteq \Omega$, $m(A)$ represents the belief that one is willing to commit exactly to A, given a certain piece of evidence. The subsets A of Ω such that $m(A) > 0$ are called the *focal elements*

Manuscript received April 1, 1995; revised April 19, 1997 and July 30, 1999. This paper was recommended by Associate Editor M. Kam.

The author is with the U.M.R. CNRS 6599 Heudiasyc, Université de Technologie de Compiègne, BP 20529-F-60205 Compiègne Cedex, France (e-mail: Thierry.Denoeux@hds.utc.fr).

Publisher Item Identifier S 1083-4427(00)01175-9.

of m. Associated with m are a *belief* or *credibility* function bel and a *plausibility* function pl, defined, respectively, for all $A \subseteq \Omega$ as

$$\text{bel}(A) = \sum_{B \subseteq A} m(B) \tag{3}$$

$$\text{pl}(A) = \sum_{A \cap B \neq \emptyset} m(B). \tag{4}$$

The quantity bel(A) can be interpreted as a global measure of one's belief that hypothesis A is true, while pl(A) may be viewed as the amount of belief that could *potentially* be placed in A, if further information became available [26].

Two BBA's m_1 and m_2 on Ω, induced by two independent items of evidence, can be combined by the so-called *Dempster's rule of combination* to yield a new BBA $m = m_1 \oplus m_2$, called the orthogonal sum of m_1 and m_2, and defined as:

$$m(\emptyset) = 0 \tag{5}$$

$$m(A) = \frac{\sum_{B \cap C = A} m_1(B) m_2(C)}{\sum_{B \cap C \neq \emptyset} m_1(B) m_2(C)} \qquad A \neq \emptyset. \tag{6}$$

The computation of m is possible if and only if there exist at least two subsets B and C of Ω with $B \cap C \neq \emptyset$ such that $m_1(B) \neq 0$ and $m_2(C) \neq 0$. m_1 and m_2 are then said to be *combinable*. The orthogonal sum is commutative and associative.

Alternatively, Smets [24] proposed to use the *unnormalized* or *conjunctive* rule of combination $\cap$, defined for all $A \subseteq \Omega$ by

$$m = m_1 \cap m_2 \Leftrightarrow m(A) = \sum_{B \cap C = A} m_1(B) m_2(C), \quad \forall A \subseteq \Omega. \tag{7}$$

Note that, using this rule, positive masses can be assigned to the empty set, which violates the condition expressed by (1). Therefore, as a consequence of the adoption of the unnormalized rule, this condition has to be removed from the definition of a BBA. The definition of a belief function also has to be rewritten as

$$\text{bel}(A) = \sum_{\emptyset \neq B \subseteq A} m(B) \quad \forall A \in \Omega. \tag{8}$$

Smets [24] has shown that these modifications to the standard theory amount to accepting that none of the hypotheses in Ω might be true (open-world assumption). This helps to avoid some counterintuitive effects that may be encountered when combining contradictory pieces of evidence [24].

Having summarized all the available evidence in the form of a BBA, the question may arise of how to make a decision regarding the selection of one single hypothesis in Ω. This problem is not trivial, since the belief and plausibility functions may induce different rankings of single hypotheses, i.e., it is possible that, for some subsets A and B of Ω, bel$(A) <$ bel(B) whereas pl$(A) >$ pl(B). As remarked by Smets [26], decisions have to be based on probabilities if Dutch Books are to be avoided. A belief functions thus has to be transformed into a probability function for decision making. The only transformation satisfying elementary rationality requirements was shown by Smets to be the pignistic transformation [26], in which each mass of belief $m(A)$ is distributed equally among the elements of A for all $A \subseteq \Omega$. This leads to the pignistic probability distribution defined as

$$\text{BetP}(\omega) = \sum_{\omega \in A} \frac{m(A)}{|A|}, \quad \forall \omega \in \Omega \tag{9}$$

where $|A|$ denotes the cardinality of $A \subseteq \Omega$. As noted in [26], this approach can be better understood if one makes a distinction between two levels: a *credal* level at which pieces of evidence are taken into consideration and combined, and a *pignistic* level at which a probability distribution is constructed in order to allow decision-making.

B. Application to Pattern Classification

Let us consider the case where some pattern $\boldsymbol{x} \in \mathbb{R}^P$ has to be classified in one of M classes $\omega_1, \cdots, \omega_M$ using a training set $\mathcal{X}$ of N P-dimensional patterns with known classification.[1] Each training vector $\boldsymbol{x}^i$ sufficiently close to $\boldsymbol{x}$ according to some distance measure d can be regarded as a piece of evidence that influences our belief concerning the class of $\boldsymbol{x}$. This item of evidence can be represented by a BBA m^i over the frame of discernment $\Omega = \{\omega_1, \cdots, \omega_M\}$. If $\boldsymbol{x}^i$ belongs to class ω_q, then the unit mass should be distributed among two subsets of Ω: the singleton $\{\omega_q\}$ and Ω itself. If we consider as a reasonable assumption that the portion of belief committed to ω_q should be a decreasing function of the distance d^i between $\boldsymbol{x}$ and $\boldsymbol{x}^i$, then m^i can be written in the following form:

$$m^i(\{\omega_q\}) = \alpha \phi_q(d^i) \tag{10}$$

$$m^i(\Omega) = 1 - \alpha \phi_q(d^i) \tag{11}$$

$$m^i(A) = 0 \quad \forall A \in 2^\Omega \setminus \{\{\omega_q\}, \Omega\} \tag{12}$$

where $0 < \alpha < 1$ is a constant and ϕ_q is a monotonically decreasing function verifying $\phi_q(0) = 1$ and $\lim_{d \to \infty} \phi_q(d) = 0$. In [5], an exponential form[2] was postulated for ϕ_q:

$$\phi_q(d^i) = \exp(-\gamma_q (d^i)^2) \tag{13}$$

γ_q being a positive constant associated to class ω_q. A method for optimizing parameters α and γ_q has been described in [30].

The above discussion concerned an arbitrary training pattern $\boldsymbol{x}^i \in \mathcal{X}$. However, it is unlikely that all training patterns will be helpful in classifying $\boldsymbol{x}$, so that we can focus our attention on the set $\Phi_k(\boldsymbol{x})$ of its k nearest neighbors. For each $\boldsymbol{x}^i \in \Phi_k(\boldsymbol{x})$, we can construct a BBA m^i summarizing the information provided by $\boldsymbol{x}^i$ concerning the class of $\boldsymbol{x}$. In order to classify $\boldsymbol{x}$, we need to combine these BBA's, using one of the two combination rules mentioned in the previous section. In [5], the normalized rule was considered, so that the BBA representing the evidence of

[1]In this paper, the class of each training pattern is assumed to be perfectly known. However, the method can be easily extended to the more general case where class labels are imprecise and/or uncertain [5], [7].

[2]A justification of this choice is provided in [7].

the k nearest neighbors of $\boldsymbol{x}$ was defined as the orthogonal sum (5), (6) of the individual m^i, for all $\boldsymbol{x}^i \in \Phi_k(\boldsymbol{x})$:

$$m = \bigoplus_{\boldsymbol{x}^i \in \Phi_k(\boldsymbol{x})} m^i. \tag{14}$$

Since the focal elements of m are the classes represented among $\Phi_k(\boldsymbol{x})$, and Ω, we have bel$(\{\omega_q\}) = m(\{\omega_q\})$ and pl$(\{\omega_q\}) = m(\{\omega_q\}) + m(\Omega)$ for $q = 1, \cdots, M$. The pignistic probability distribution is defined as

$$\text{BetP}(\{\omega_q\}) = m(\{\omega_q\}) + \frac{m(\Omega)}{M} \tag{15}$$

for all $\omega_q \in \Omega$.

In the case of $\{0, 1\}$ losses (where the loss incurred by assigning a pattern to a class is 1 for misclassification and 0 otherwise) and if no possibility of rejection exists, it is well known that the Bayes rule leads to selecting the class with maximum posterior probability [10]. In a similar way, the risk relative to the pignistic probability distribution is minimized by choosing the class with maximum pignistic probability [6]. More sophisticated decision strategies will be examined in Section IV.

III. Adaptive Classifier

A. From Neighbors to Prototypes

The computational complexity of the search for the nearest neighbors in a training set is known to be an important drawback of k-NN techniques. This problem may be alleviated by synthesizing the learning set in the form of a limited number of representative patterns, or *prototypes*. We therefore considered a modification of the procedure described in Section II-B, in which the assignment of a pattern to a class is made by computing its distances to n prototypes: $\boldsymbol{p}^1, \cdots, \boldsymbol{p}^n$. Each prototype i is assumed to possess a degree of membership u_q^i to each class ω_q, with the constraint $\Sigma_{q=1}^M \; u_q^i = 1$. Full membership of a prototype to one class can be considered as a special case where $u_q^i = 1$ for some q and $u_l^i = 0$ for $l \neq q$.

A BBA describing the uncertainty pertaining to the class membership of $\boldsymbol{x}$ can be computed as a result of a three-step procedure:

Step 1: The distances d^i between $\boldsymbol{x}$ and each prototype vector $\boldsymbol{p}^i$ are computed according to some metric, for example the Euclidean one:

$$d^i = ||\boldsymbol{x} - \boldsymbol{p}^i|| \qquad i = 1, \cdots, n. \tag{16}$$

Step 2: The information provided by each prototype is represented by a BBA m^i depending on the class membership of $\boldsymbol{p}^i$, and on d^i. Assuming proportionality between $m^i(\{\omega_q\})$ and u_q^i, we have

$$\forall q \in \{1, \cdots, M\} \quad m^i(\{\omega_q\}) = \alpha^i u_q^i \phi^i(d^i) \tag{17}$$

$$m^i(\Omega) = 1 - \alpha^i \phi^i(d^i) \tag{18}$$

where ϕ^i is a decreasing function varying between 1 and 0 and $0 < \alpha^i < 1$ is a parameter associated to prototype i.This definition is quite similar to that given by (10)–(12), except that positive masses are now assigned to each class ω_q for which $u_q^i > 0$. Note that we have

$$\sum_{A \subseteq \Omega} m^i(A) = \sum_{q=1}^{M} m^i(\{\omega_q\}) + m^i(\Omega) = 1 \tag{19}$$

and $m^i(\emptyset) = 0$, so that m^i verifies the conditions expressed be (1) and (2). As before (13), ϕ^i can be defined as

$$\phi^i(d^i) = \exp(-\gamma^i(d^i)^2) \tag{20}$$

where γ^i is now a positive parameter associated to prototype i.

Step 3: The n BBA's $m^i, i = 1, \cdots, n$ are combined using either the conjunctive rule defined by (7), or the Dempster's rule. The former yields an unnormalized BBA

$$m = \bigcap_{i=1}^{n} m^i, \tag{21}$$

while the latter yields a normalized BBA:

$$m' = \bigoplus_{i=1}^{n} m^i. \tag{22}$$

It is straightforward to show that $m' = m/K$ with $K = \Sigma_{q=1}^M \; m(\{\omega_q\}) + m(\Omega)$.

In the case of $\{0, 1\}$ losses and no reject option, a decision can then be made to assign input vector $\boldsymbol{x}$ to the class $D(\boldsymbol{x}) = \omega_r$ verifying

$$m(\{\omega_r\}) = \max_q \; m(\{\omega_q\}). \tag{23}$$

Note that we also have $m'(\{\omega_r\}) = \max_q \; m'(\{\omega_q\})$, so that the consideration of m or m' leads to the same decision.

B. Connectionist Implementation

The classification method just introduced has some similarity with radial basis function (RBF) networks [18]. A RBF network is a neural network composed of an input layer, a hidden layer and an output layer (Fig. 1). The response of hidden unit i to an input vector $\boldsymbol{x}$ is defined as a decreasing function of the distance between $\boldsymbol{x}$ and a weight vector $\boldsymbol{p}^i$. The output signal o^j from the jth output unit with weight vector $\boldsymbol{w}^j$ is obtained as a weighted sum of the activations in the hidden layer:

$$o^j = \sum_{i=1}^{n} w_i^j s^i \tag{24}$$

where

w_i^j	ith component of vector $\boldsymbol{w}^j$;
s^i	output from hidden unit i;
n	number of hidden units.

The different parameters can be determined by gradient descent of some error function.

The evidence-theoretic classifier introduced in this paper can also be represented in the connectionist formalism as a neural network with an input layer, two hidden layers L_1 and L_2, and an output layer L_3 (Fig. 2). Each layer L_1 to L_3 corresponds to one step of the procedure described in the former section:

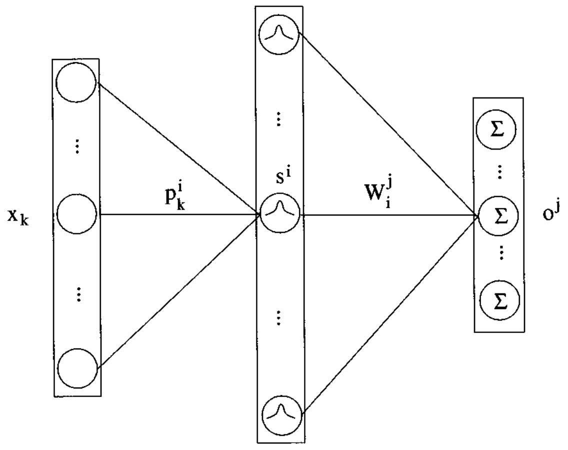

Fig. 1. Architecture of a radial basis function network.

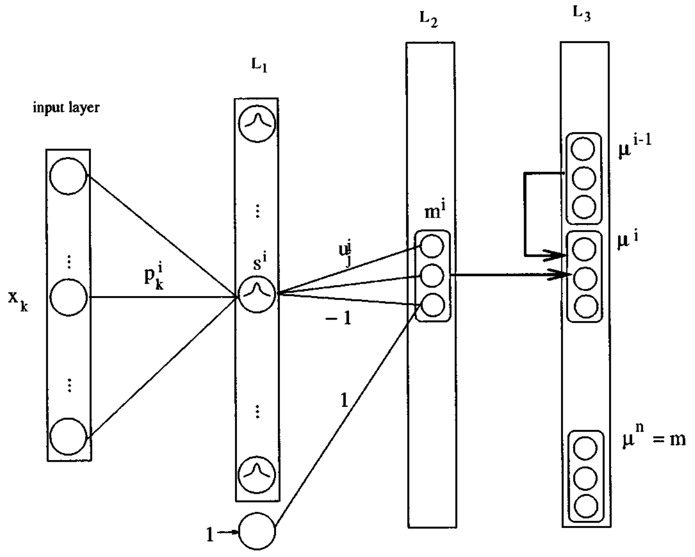

Fig. 2. Connectionist implementation of the evidence-theoretic classifier. The fixed-weight connections between layers L_2 and L_3, as well as inside layer L_3, are described in Fig. 3.

1) Layer L_1 contains n units with activation $s^i = \alpha^i \exp(-\gamma^i (d^i)^2)$ for $i = 1, \cdots, n$. It is identical to the hidden layer of an RBF network with exponential activation function.
2) Layer L_2 computes the BBA m^i associated to each prototype. It is composed of n modules of $M+1$ units each. The units of module i are connected to neuron i of the previous layer. The vector of activations $\boldsymbol{m}^i = (m^i_1, \cdots, m^i_{M+1})^t$ of module i corresponds to the belief masses assigned by m^i:

$$\boldsymbol{m}^i = (m^i(\{\omega_1\}), \cdots, m^i(\{\omega_M\}), m^i(\Omega))^t \tag{25}$$
$$= (u^i_1 s^i, \cdots, u^i_M s^i, 1 - s^i)^t. \tag{26}$$

3) The n BBA's $m^i, i = 1, \cdots, n$ are then combined in layer L_3, composed of n interconnected modules of $M+1$ sigma-pi units. The vector of activations $\vec{\mu}^i = (\mu^i_1, \cdots, \mu^i_{M+1})^t$ of module i in that layer is defined as

$$\vec{\mu}^i = (\mu^i(\{\omega_1\}), \cdots, \mu^i(\{\omega_M\}), \mu^i(\Omega))^t \tag{27}$$

where μ^i is the conjunctive combination of the BBA's $m^1, \cdots, m^i$:

$$\mu^1 = m^1 \tag{28}$$
$$\mu^i = \bigcap_{k=1}^{i} m^k = \mu^{i-1} \cap m^i \qquad i = 2, \cdots, n. \tag{29}$$

The activation vectors $\vec{\mu}^i$ for $i = 2, \cdots, n$ can be recursively computed using the following formula:

$$\mu^i_j = \mu^{i-1}_j m^i_j + \mu^{i-1}_j m^i_{M+1} + \mu^{i-1}_{M+1} m^i_j \qquad j = 1, \cdots, M \tag{30}$$
$$\mu^i_{M+1} = \mu^{i-1}_{M+1} m^i_{M+1}. \tag{31}$$

This calculation can be performed by each computing element in module $i > 1$ of layer L_3, provided it receives input from module $i-1$ in the same layer, and from module i of layer L_2 (Fig. 3). This can be achieved by fixed-weight connections between layers L_2 and L_3, and inside layer L_3.
4) The network output vector $\boldsymbol{m} = (m_1, \cdots, m_{M+1})^t$ is then defined as

$$\boldsymbol{m} = \mu^n \tag{32}$$

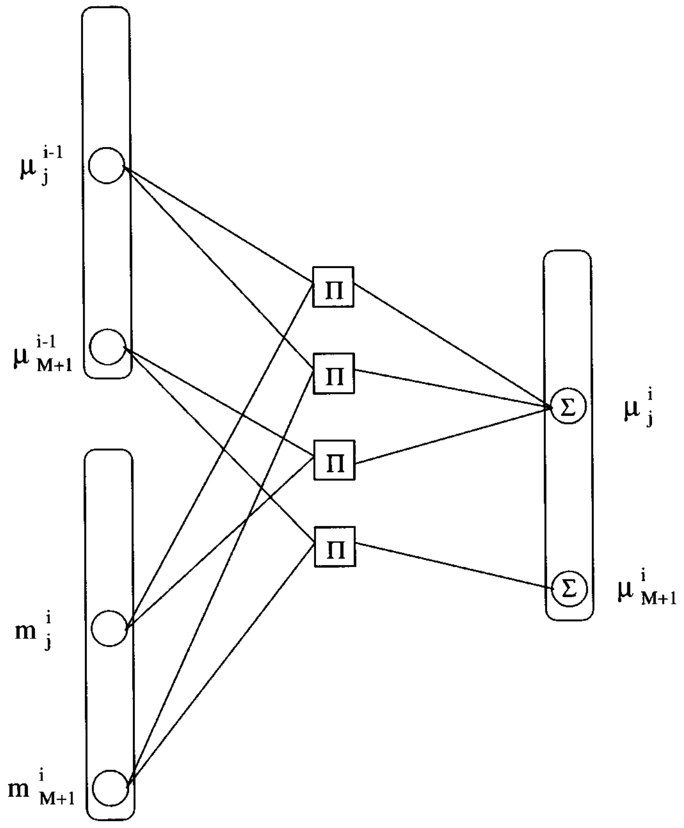

Fig. 3. Details of incoming connections to module i of layer L_3. All connection weights are fixed and equal to unity.

or, alternatively

$$\boldsymbol{m}' = \boldsymbol{m}/K \tag{33}$$

$$K = \sum_{k=1}^{M+1} m_k. \tag{34}$$

Note that, because of the commutativity and associativity of the conjunctive combination, the network output is independent from the order in which the prototypes are taken in consideration. It therefore remains unchanged by any permutation of the upper indices in layers L_1, L_2, and L_3.

Although the architecture of that network looks somewhat more complex than that of a RBF network, both models involve approximately the same amount of computation. In both networks, the computation of the activations in the first hidden layer requires $O(nP)$ arithmetic operations (where P denotes the number of inputs), and the combination of the outputs from the first hidden layer can be performed in $O(nM)$ operations. Hence, the overall complexity is $O(n(P+M))$ operations for the propagation of one input pattern.

C. Parameter Learning

The above network can be trained, in the same way as a RBF network, by optimizing a performance criterion.

Let us denote by $\boldsymbol{t} = (t_1, \cdots, t_M)^t$ the target output vector for pattern $\boldsymbol{x}$, with $t_j = \delta_{jq}$ if $\boldsymbol{x} \in \omega_q$. An output error for that pattern can be defined by comparing $\boldsymbol{t}$ to the classifier output vector $\boldsymbol{m}$ (or alternatively its normalized version $\boldsymbol{m}'$). Since $\boldsymbol{t}$ has M components while $\boldsymbol{m}$ is of length $M+1$, we need to define from $\boldsymbol{m}$ a vector of length M. This can be achieved by distributing a fraction ν of m_{M+1} (the mass assigned to the set Ω of classes) to each class. A transformed output vector $\boldsymbol{P}_\nu = (P_{\nu,1}, \cdots, P_{\nu,M})^t$ is then defined as

$$P_{\nu,q} = m_q + \nu m_{M+1} \qquad q = 1, \cdots, M \tag{35}$$

with $0 \le \nu \le 1$. Obviously, $P_{0,q}$, $P_{1,q}$ and $P_{1/M,q}$ represent, respectively, the credibility, the plausibility and the pignistic probability of class ω_q, with respect to the BBA m. The output error $E_\nu(\boldsymbol{x})$ for a given ν and input pattern $\boldsymbol{x}$ can then be defined as[3]

$$E_\nu(\boldsymbol{x}) = \tfrac{1}{2}||\boldsymbol{P}_\nu - \boldsymbol{t}||^2 = \tfrac{1}{2} \sum_{q=1}^{M} (P_{\nu,q} - t_q)^2. \tag{36}$$

The mean output error for the whole training set $\mathcal{X}$ of size N is then given by

$$E_\nu = \frac{1}{N} \sum_{\boldsymbol{x} \in \mathcal{X}} E_\nu(\boldsymbol{x}). \tag{37}$$

If the normalized output vector $\boldsymbol{m}'$ is used, then a corresponding transformed output vector $\boldsymbol{P}'_\nu$ and an associated error E'_ν can be defined in a similar fashion.

[3]Other error measures could be used, such as, for example, the cross entropy measure [1]:

$$CE(\boldsymbol{x}) = -\sum_{q=1}^{M} t_q \, \log_2(P_{\nu,q}) + (1-t_q) \, \log_2(1-P_{\nu,q}).$$

The parameters p_k^i, γ^i, α^i, and u_j^i can be adjusted so as to minimize E_ν (respectively, E'_ν), under the constraints

$$\gamma^i > 0 \tag{38}$$

$$0 < \alpha^i < 1 \tag{39}$$

$$\sum_{j=1}^{M} u_j^i = 1 \tag{40}$$

for all $1 \le i \le n$. These constraints can be taken into account by introducing new parameters η^i, ξ^i, and β_j^i such that

$$\gamma^i = (\eta^i)^2 \tag{41}$$

$$\alpha^i = \frac{1}{1+\exp(-\xi^i)} \tag{42}$$

$$u_j^i = \frac{(\beta_j^i)^2}{\sum_{k=1}^{M} (\beta_k^i)^2} \tag{43}$$

and minimizing E_ν (respectively, E'_ν) with respect to p_k^i, η^i, ξ^i, and β_j^i. The derivatives of $E_\nu(\boldsymbol{x})$ [respectively, $E'_\nu(\boldsymbol{x})$] with respect to all parameters are given in Appendix A. The derivatives of E_ν (respectively, E'_ν) are obtained by summing the derivatives of $E_\nu(\boldsymbol{x})$ [respectively, $E'_\nu(\boldsymbol{x})$] for all $\boldsymbol{x} \in \mathcal{X}$. As shown in Appendix A, calculation of the whole gradient can be performed in linear time with respect to the input dimension P, the number M of classes and the number n of prototypes. Convergence to a local minimum of the error function can be ensured using iterative gradient-based optimization procedures such as described in [2]. Note that the iterative minimization of E_ν corresponds to what is refered to as "batch learning" in the neural network literature.

A regularized version of the error criterion allowing to limit the impact of overparametrization on generalization performance is described in Appendix B.

IV. Decision Analysis

Once a classifier has been trained using the method described in the previous section, the question arises of how to use it for classifying previously unseen patterns. As mentioned in Section II-B, a straightforward decision strategy is to to select the class with maximum pignistic probability. More complex rules allowing for rejection of ambiguous patterns and novelty detection are discussed in this section. A much more detailed discussion on this issue may be found in [6].

To begin with, let us assume that a normalized BBA m has been computed for pattern $\boldsymbol{x}$, based on training set information. As in standard Bayesian decision theory, we now consider the problem of deciding between a finite set $\mathcal{A}$ of actions, based on m and on the losses $\lambda(\alpha|\omega)$ incurred for choosing action α whereas the pattern under consideration actually belongs to class ω, for each $(\alpha,\omega) \in \mathcal{A} \times \Omega$. The risk relative to the pignistic probability distribution BetP induced by m, defined for each $\alpha \in \mathcal{A}$ as

$$R_{\text{bet}}(\alpha) = \sum_{\omega\in\Omega} \lambda(\alpha|\omega)\text{BetP}(\{\omega\}) \tag{44}$$

$$= \sum_{\omega\in\Omega} \lambda(\alpha|\omega) \sum_{A\ni\omega} \frac{m(A)}{|A|} \tag{45}$$

$$= \sum_{A\subseteq\Omega} m(A)\frac{1}{|A|} \sum_{\omega\in A} \lambda(\alpha|\omega). \tag{46}$$

With different choices for $\mathcal{A}$ and $\lambda(\alpha|\omega)$ $(\alpha \in \mathcal{A}, \omega \in \Omega)$, the strategy of pignistic risk minimization will lead to different decision rules. To demonstrate the generality of this approach, let us examine how pattern rejection may be handled in this framework.

As remarked by Dubuisson and Masson [9], pattern rejection may be useful in at least two different situations: 1) when several classes seem almost equally likely, so that the risk of misclassification is very high [4] and 2) when the pattern under consideration is surprisingly different for previous data, which may be caused by gross measurement errors (due to sensor failure for example), or by the occurrence of an entity from a population that was not represented in the learning set.

To see how these two situations can be modeled in our approach, let us consider two distinct cases, corresponding to the hypotheses of existence or nonexistence of unknown classes, respectively.

Case 1: The training set contains samples from all classes. The possible actions are then assigned to class ω_q for $q \in \{1,\cdots,M\}$, denoted by α_q, and rejection α_0. As before, the losses are defined to be 0 for correct classification and 1 for misclassification. Furthermore, the loss of rejecting a pattern is assumed to be the same whatever the actual class of that pattern, and is noted λ_0. The risks relative to the pignistic probability distribution are then

$$\begin{aligned} R_{\text{bet}}(\alpha_q) &= 1 - \text{BetP}(\{\omega_q\}) \\ &= 1 - m(\{\omega_q\}) - \frac{m(\Omega)}{M} \qquad q \in \{1,\cdots,M\} \quad (47) \\ R_{\text{bet}}(\alpha_0) &= \lambda_0. \quad (48) \end{aligned}$$

Assignment to the class with the largest pignistic probability is then decided if that probability is greater than $1 - \lambda_0$. Rejection is preferred when the maximum probability is too small, which may occur when several classes are almost equally likely, that is to say, $m(\{\omega_i\}) \approx m(\{\omega_j\})$ for some i and j, or when $m(\Omega) \approx 1$. The first situation corresponds to an ambiguous pattern (situated close to the boundary between two classes), while the second one corresponds to an outlier (situated at a large distance from each of the training patterns). The same rule thus allows to reject both ambiguous patterns and outliers.

Case 2: Some classes may not be represented in the training set. This may happen for various reasons [6], for example:

1) some classes have very small prior probabilities;
2) Some classes correspond to states of nature that are systematically avoided for being too dangerous or too costly (such as certain faults in technological systems for example);
3) the number of classes is very high;
4) an exhaustive list of all possible states of nature is not available (some classes have never been observed and are not even conceived by the user).

In that case, the set of classes may be partitioned into a set K of known classes and a set U containing those states of nature which are not represented in the learning set. Since nothing is known about set U, it can be treated as a singleton: $U = \{\omega_u\}$. The set of classes Ω then contains $M + 1$ elements: $\Omega = \{\omega_1,\cdots,\omega_M,\omega_u\}$. As before, the possible actions are assignment to one (known or unknown) class, and rejection: $\mathcal{A} = \{\alpha_0,\alpha_1,\cdots,\alpha_M,\alpha_u\}$. Let us assume the losses to be 1 for wrong assignment to a known class, 0 for correct classification, λ_0 for rejection, and λ_1 for wrong assignment to the unknown class. It seems reasonable to assume $\lambda_1 < 1$, since declaring a pattern as unknown can be seen as a kind of rejection, which may have less severe consequences than a misclassification error. The pignistic risks are then

$$\begin{aligned} R_{\text{bet}}(\alpha_q) &= \sum_{j\neq q} m(\{\omega_j\}) + m(\Omega)\,\frac{M}{M+1} \\ &= 1 - \text{BetP}(\{\omega_q\}) \quad \text{for } q = 1,\cdots,M \quad (49) \\ R_{\text{bet}}(\alpha_0) &= \lambda_0 \quad (50) \\ R_{\text{bet}}(\alpha_u) &= \lambda_1 \sum_{j=1}^{M} m(\{\omega_j\}) + m(\Omega)\,\frac{M\lambda_1}{M+1} \quad (51) \\ &= \lambda_1\left(1 - \frac{m(\Omega)}{M+1}\right). \quad (52) \end{aligned}$$

The decision rule is then similar to the previous one, except that assignment to the unknown class is now decided whenever $m(\Omega)$ exceeds some threshold, i.e., when the pattern is very dissimilar from each of the training patterns.

These decision rules will be demonstrated in the next section.

V. Experimental Results

This section reports some experimental results that demonstrate various aspects of the classification method presented in this paper. It is subdivided into three parts.

1) First, a real data set is used to illustrate the form of the outputs generated by our method, as well as examples of the decision regions produced by the decision rules presented in the former section.
2) The performance of our method is then assessed quantitatively on two benchmark classification tasks for which results from a significant sample of other methods are available; the results obtained basically show that 1) our method outperforms most of the other classification methods tested on these two tasks and 2) it allows for efficient rejection of outliers.
3) The third part describes an experiment simulating a data fusion application; two classifiers are trained separately to classify patterns based on independent feature vectors

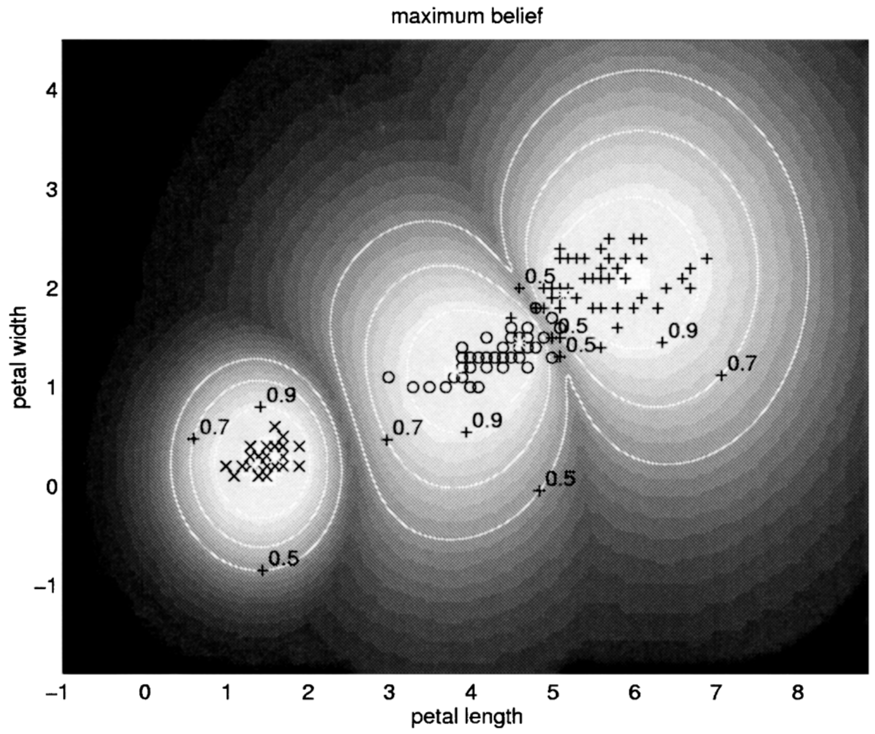

Fig. 4. Maximum credibility for the iris data, represented as greyscales with light grey as 1 and black as 0, and contours at 0.5, 0.7, and 0.9. (+: *iris virginica*, ∘: *iris versicolor*, ×: *iris setosa*).

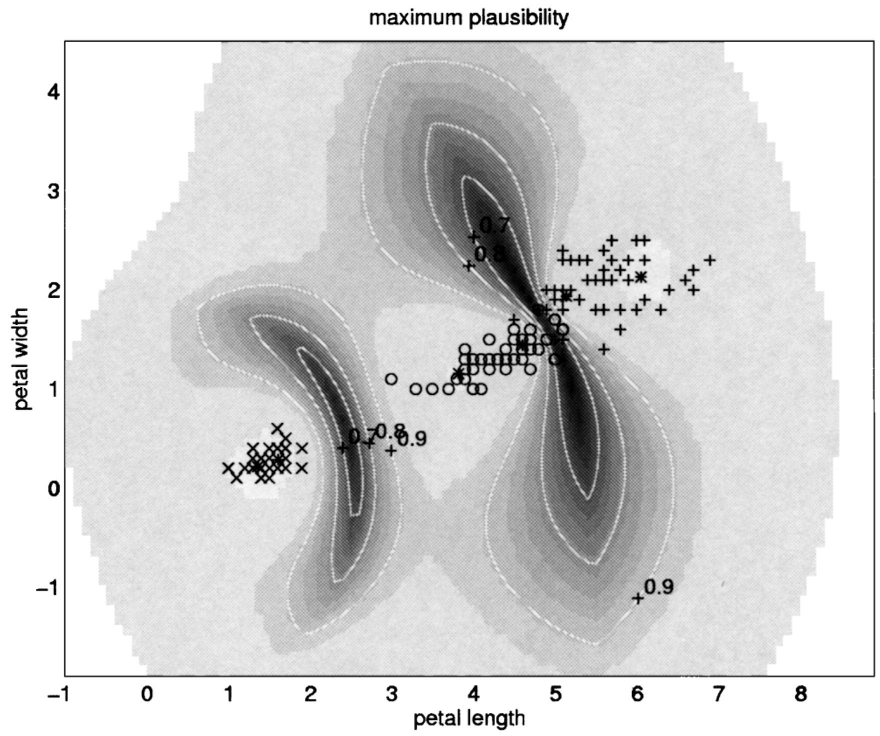

Fig. 5. Maximum plausibility for the iris data, represented as greyscales with light grey as 1 and black as 0, and contours at 0.7, 0.8, and 0.9 (+: *iris virginica*, ∘: *iris versicolor*, ×: *iris setosa*).

(from two different sensors), and their outputs are combined: 1) in the framework of Bayes theory with statistical and neural network classifiers and 2) in the framework of D–S theory with our method; our approach proves much more robust than the other methods to the failure of one sensor.

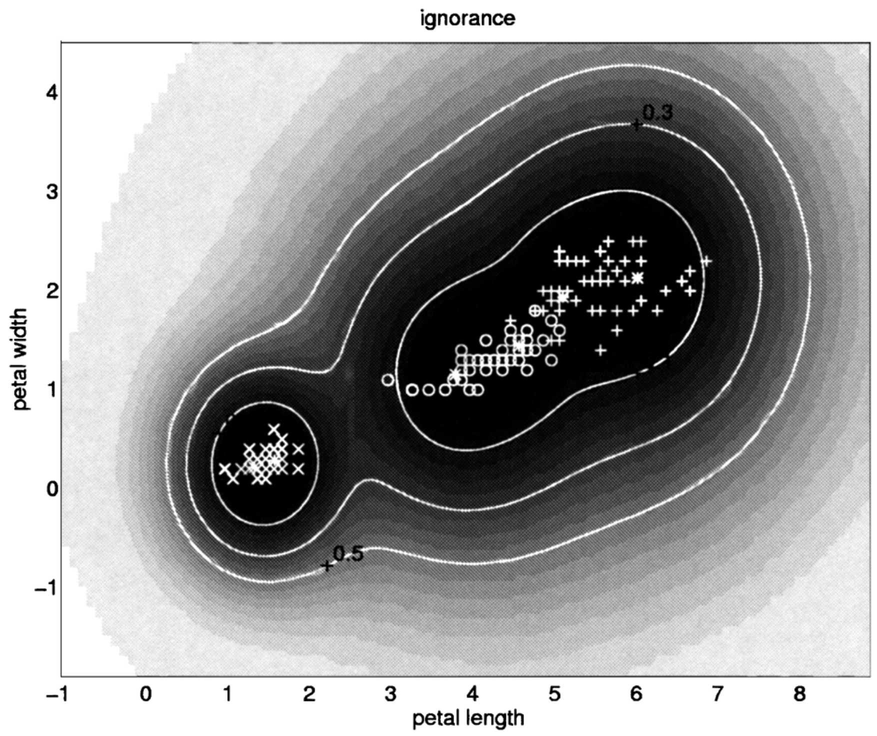

Fig. 6. Ignorance $(m(\Omega))$ for the iris data, represented as greyscales with light grey as 1 and black as 0, and contours at 0.1, 0.3, and 0.5 $(+$: *iris virginica*, $\circ$: *iris versicolor*, $\times$: *iris setosa*).

A. Iris Data

The well-known Anderson iris data consist of 150 samples belonging to three classes (*iris virginica*, *iris versicolor*, and *iris setosa*), in a four-dimensional (4-D) feature space. Only two features (petal width and petal length) were considered in this analysis. In the training phase, the 150 samples were presented to a network with six prototypes. The network was trained by minimizing E'_ν with $\nu = 1/3$. The initial prototype locations were determined by a simple clustering method. Since the classification task is very easy in this example, the results will only be interpreted qualitatively.

Different representations of the outputs produced by the classifier are shown in Figs. 4–7. Fig. 4 shows the maximum belief or credibility, i.e., the quantity

$$\text{bel}(\{\omega_r\}) = \max_q \text{bel}(\{\omega_q\})$$

which may be interpreted as the weight of the evidence directly supporting the assignment of the input vector to its predicted class. As expected, the credibility decreases with the distance to training vectors, as the available information become less reliable. The maximum plausibility

$$\text{pl}(\{\omega_r\}) = \max_q \ \text{pl}(\{\omega_q\}),$$

represented in Fig. 5 reflects the absence of evidence that contradicts the assignment of the input vector to its predicted class. It is high when there is little ambiguity, and it is minimum in the vicinity of the boundary between classes, where the available evidence points simultaneously to several hypotheses with almost equal strength. At each point of the feature space, the width of the interval $[\text{bel}(\{\omega_r\}), \text{pl}(\{\omega_r\})]$ is equal to the mass $m(\Omega)$, which characterizes the amount of belief that could not be committed to any particular class because of the weakness of the available information (Fig. 6); it is larger in those regions of the feature space which are far away from each of the training vectors, and can be interpreted as a measure of ignorance. Finally, the maximum pignistic probability

$$\text{BetP}(\{\omega_r\}) = \max_q \ \text{BetP}(\{\omega_q\}),$$

is shown in Fig. 7. It corresponds to an equal allocation of the uncommitted mass $m(\Omega)$ to each of the classes, and is particularly useful for decision making, as shown in Section IV.

Three different partitions of the feature space into decision regions are shown in Figs. 8–10. In all cases, it is assumed that $\lambda(\alpha_i|\omega_j) = 1 - \delta_{i,j}$, for $i, j \in \{1, \cdots, M\}$, and the strategy of pignistic risk minimization is used. In Fig. 8, there is no reject option, and the training set is supposed to be complete (no unknown class). In that case, one has no other choice than assigning each feature vector to one of the known classes, and the decisions are very close to what would be achieved with a conventional classifier. In Fig. 9, the training set is still assumed to be complete, but rejection is possible: this option tends to be chosen for ambiguous patterns and for outliers. In Fig. 10, an unknown class was explicitly included into the model specification. Outliers tend to be categorized as belonging to the unknown class, while dubious cases are rejected.

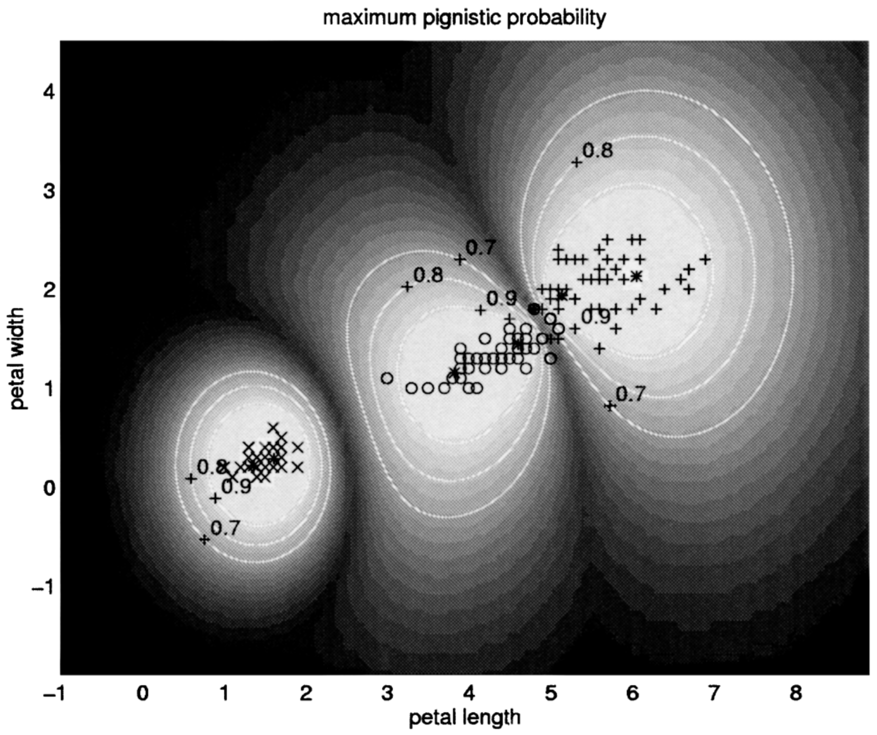

Fig. 7. Maximum pignistic probability for the iris data, represented as greyscales with light grey as 1 and black as 0, and contours at 0.7, 0.8, and 0.9 ($+$: *iris virginica*, $\circ$: *iris versicolor*, $\times$: *iris setosa*).

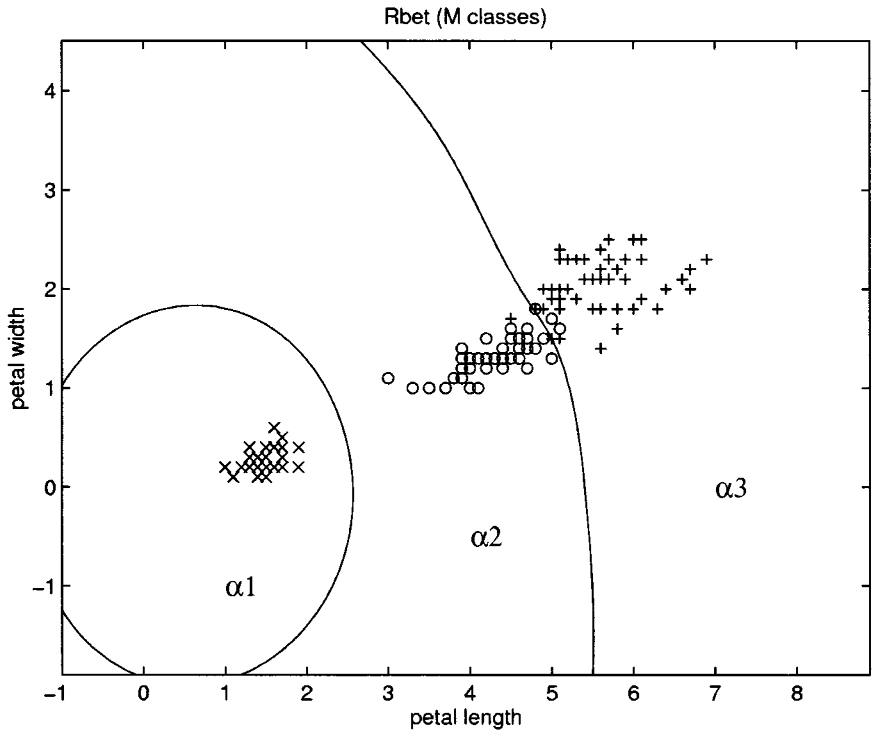

Fig. 8. Decision regions for the iris data, obtained by minimizing the pignistic risk with the $\{0, 1\}$ loss function. The only possible actions are assignment to one of the three classes ($+$: *iris virginica*, $\circ$: *iris versicolor*, $\times$: *iris setosa*).

B. Performance Comparison

1) Phoneme Recognition Data: As a first example of a complex real-world classification task, we considered the speech recognition data collected by Deterding [8] and used by Robinson [21] in a benchmarking study of neural network and statistical classifiers. The data set is composed of feature vectors obtained by recording examples of the eleven steady state vowels of English spoken by fifteen speakers [8], [21] Words containing each of these vowels were uttered once by

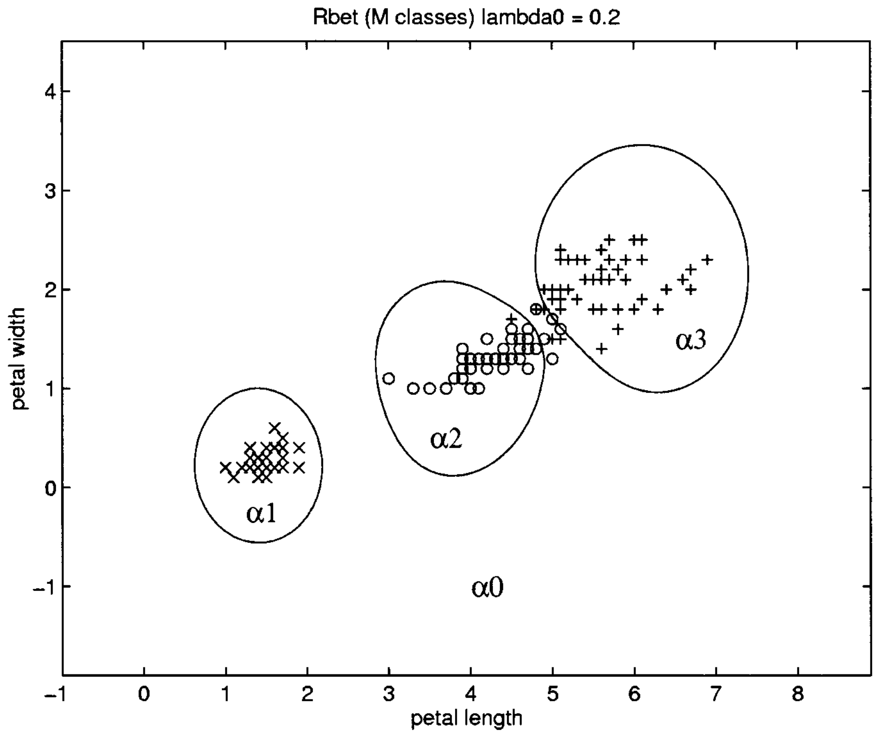

Fig. 9. Decision regions for the iris data, obtained by minimizing the pignistic risk with the $\{0, 1\}$ loss function. The possible actions are assignment to one of the three classes, and rejection with loss $\lambda_0 = 0.2$ ($+$: *iris virginica*, $\circ$: *iris versicolor*, $\times$: *iris setosa*).

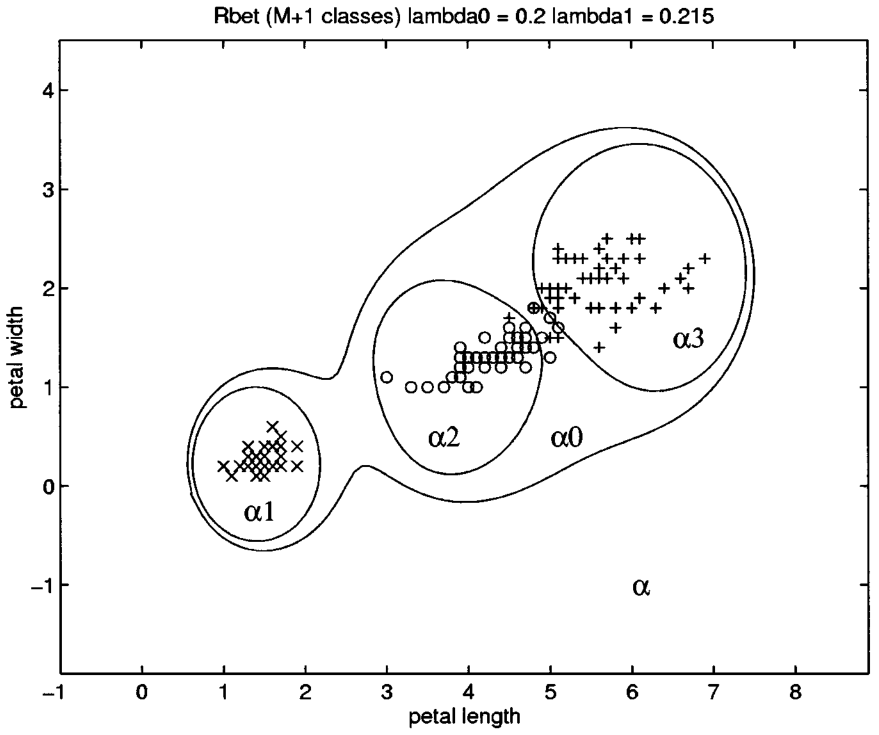

Fig. 10. Decision regions for the iris data, obtained by minimizing the pignistic risk with the $\{0, 1\}$ loss function. The possible actions are assignment to one of the three classes, rejection with loss $\lambda_0 = 0.2$, and assignment to the unknown class with $\lambda_1 = 0.215$ ($+$: *iris virginica*, $\circ$: *iris versicolor*, $\times$: *iris setosa*).

the fifteen speakers. Four male and four female speakers were used to build a training set, and the other four male and three female speakers were used for building a test set. After suitable preprocessing, 568 training patterns and 462 test patterns in a ten-dimensional (10-D) input space were collected. The results obtained with our method with one to five prototypes per class are shown in Table I, together with results reported by Robinson [8] and Hastie and Tibshirani [12] for various statistical and neural network classifiers. Test error rates are defined as the proportions of misclassifying samples in the test set. Note that, although none of the tested method appears at first sight to perform very well on this data, the test error rates obtained should be compared with that of the naive decision rule choosing the most frequent class in the training set, which is only equal to 0.91 in that case.

In this and the next experiment (Section V-B2), the initial prototype locations in our method were determined using a standard clustering procedure (a variant of the c-means algorithm), starting from random initial conditions. The initial values of the other parameters were set heuristically to $\gamma^i = 0.1$ and $\alpha^i = 0.5$ for $i = 1, \cdots, n$ (these values were fixed *a priori* and were not optimized). The initial membership values u_q^i of each prototype i to each class ω_q were defined as the proportion of training samples belonging to class ω_q in the neighborhood of prototype i. With this initialization procedure, there was almost no variation of the results in several runs.

Detailed descriptions of the experimental conditions for the other methods are given in [12] and [21]. In Robinson's simulations, each result was based on a single trial with random starting weights. In the radial basis function method as defined by Robinson, prototype locations were placed at the points defined by the input examples, whereas they were placed randomly in the Gaussian node network.

CART is a decision tree generation procedure developed by Breiman *et al.* [3]. MARS (multivariate regression splines) is an adaptive nonparametric regression technique, able to capture interactions in a hierarchical manner [11]. The *degree* is a parameter of the procedure that limits the order of the interactions allowed. BRUTO is an adaptive method for estimating an additive model using smoothing splines [13]. Both are powerful flexible discriminant analysis techniques [13].

As shown in Table I, our approach with at least three prototypes per class dominates the other techniques for this classification task.

2) Forensic Glass Data: This data set contains the description of 214 fragments of glass [17] originally collected for a study in the context of criminal investigation. Each fragment has a measured reflectivity index and chemical composition (weight percent of Na, Mg, Al, Si, K, Ca, Ba, and Fe). As suggested by Ripley [20], 29 instances were discarded, and the remaining 185 were re-grouped in four classes: window float glass (70), window nonfloat glass (76), vehicle window glass (17) and other (22). The data set was split randomly in a training set of size 89 and a test set of size 96.

Our method (with normalized outputs) was compared to three neural network classifiers: learning vector quantization (LVQ) [15], RBF networks and multilayer perceptrons. Each of the three methods based on prototypes (ETC, RBF, and LVQ) was provided with the same initial codebook vectors generated by a clustering procedure with random initial conditions (except for the case $n = 4$ were the initial prototypes were chosen as the sample mean in each class). In RBF networks, the second layer of weights was initialized using a pseudo-inverse approach. The initial parameters for the evidence-theoretic method were determined exactly in the same way as in the previous experiment. The RBF, MLP and evidence-theoretic networks were trained with the same optimization algorithm (gradient descent with adaptive learning rates). The number n of prototypes was varied

TABLE I
TEST ERROR RATES FOR THE PHONEME RECOGNITION DATA. LINES (1)–(11) ARE TAKEN FROM ROBINSON (1989); LINES (12)–(18) ARE TAKEN FROM HASTIE AND TIBSHIRANI (1994). ETC: EVIDENCE-THEORETIC CLASSIFIER

	Classifier	test error rate
(1)	Single-layer perceptron	0.67
(2)	Multi-layer perceptron (88 hidden units)	0.49
(3)	Multi-layer perceptron (22 hidden units)	0.55
(4)	Multi-layer perceptron (11 hidden units)	0.56
(5)	Radial Basis Function (528 hidden units)	0.47
(6)	Radial Basis Function (88 hidden units)	0.52
(7)	Gaussian node network (528 hidden units)	0.45
(8)	Gaussian node network (88 hidden units)	0.47
(9)	Gaussian node network (22 hidden units)	0.46
(10)	Gaussian node network (11 hidden units)	0.53
(11)	Nearest neighbor	0.44
(12)	Linear Discriminant Analysis	0.56
(13)	Quadratic Discriminant Analysis	0.53
(14)	CART	0.56
(15)	CART (linear combination splits)	0.54
(16)	BRUTO	0.44
(17)	MARS (degree=1)	0.45
(18)	MARS (degree=2)	0.42
(19)	ETC (11 hidden units)	0.47
(20)	ETC (22 hidden units)	0.42
(21)	ETC (33 hidden units)	0.38
(22)	ETC (44 hidden units)	0.37
(21)	ETC (55 hidden units)	0.37

from 4 to 8, and the number of hidden units for the MLP's was varied from 2 to 8. Each learning algorithm was run ten times in each configuration.

The average test error rates and standard deviations are reported in Table II, and represented graphically in Fig. 11. As can be seen, our method yields significantly better results than the three other neural network techniques. The best value of the mean error rate obtained is 0.29 for 7 prototypes. This value is to be compared with the experimental results reported by Ripley [20] with the same data, in an extensive study of several statistical and neural network pattern classifiers (Table III). As indicated by Ripley, the main neural network results used softmax and maximum-likelihood estimation, and were averaged over five runs using different random initial weights. The predictive method (Bayesian approach) with six hidden units was the only neural network method to yield an error rate below 30%. However, the method used 10 000 random samples from Gaussians about the peaks found in 20 optimization runs, and was thus much more computationally intensive than ours. Only two conventional statistical techniques performed well on these data: the nearest-neighbor method (with compar-

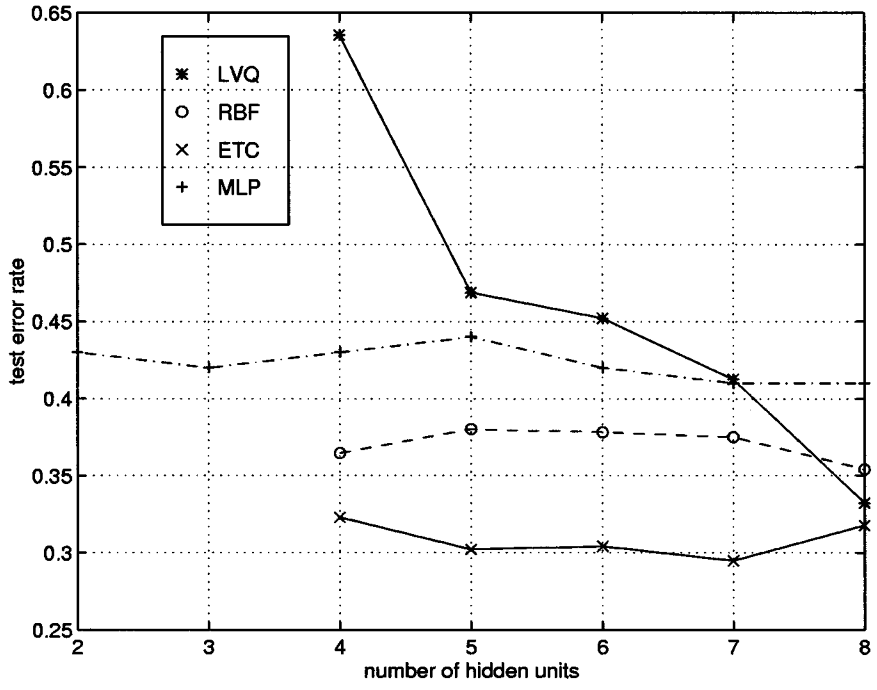

Fig. 11. Test error rates for the glass data, as a function of the number of hidden units (LVQ: learning vector quantization; RBF: radial basis function network; ETC: evidence-theoretic classifier; MLP: multilayer perceptron).

TABLE II
TEST ERROR RATES FOR THE FORENSIC GLASS EXAMPLE WITH DIFFERENT VALUES OF n (MEANS OF TEN RUNS WITH STANDARD DEVIATIONS). THE METHODS ARE LEARNING VECTOR QUANTIZATION (LVQ), RADIAL BASIS FUNCTION (RBF) NETWORK, MULTI-LAYER PERCEPTRON (MLP) AND EVIDENCE-THEORETIC CLASSIFIER (ETC)

n	2	3	4	5	6	7	8
LVQ			0.64	0.47	0.45	0.41	0.33
			± 0.00	± 0.00	± 0.02	± 0.03	± 0.06
RBF			0.36	0.38	0.38	0.38	0.35
			± 0.00	± 0.01	± 0.01	± 0.01	± 0.03
MLP	0.43	0.42	0.43	0.44	0.42	0.41	0.41
	± 0.02	± 0.03	± 0.02	± 0.04	± 0.02	± 0.05	± 0.04
ETC			0.32	0.30	0.30	**0.29**	0.32
			± 0.00	± 0.00	± 0.02	± 0.02	± 0.02

TABLE III
TEST ERROR RATES FOR THE GLASS DATA, FROM [20]

	Classifier	test error rate
(1)	Linear Discriminant Analysis	0.41
(2)	Nearest neighbor	0.26
(3)	Multi-layer perceptron (2 hidden)	0.38
(4)	Multi-layer perceptron (2 hidden), predictive	0.38
(5)	Multi-layer perceptron (6 hidden)	0.33
(6)	Multi-layer perceptron (6 hidden), predictive	0.28
(7)	Multi-layer perceptron (6 hidden), logistic outputs	0.39
(8)	CART	0.28
(9)	BRUTO	0.42
(10)	MARS (degree=1)	0.37
(11)	MARS (degree=2)	0.31
(12)	Projection pursuit regression	0.40

atively high computational and storage requirements), and the tree structured classifier (CART). Note however that no method was significantly better than ours (only differences of more than 4% are significant at the 5% level [20]). According to Ripley, the estimated lower bound for the Bayes risk was 11%, but no method comes close to this bound, which is probably too conservative.

The robustness of our method to outliers and its ability to reject them using the decision rules described in Section IV were also investigated using these data. For that purpose, outliers were artificially introduced in the test data set by replacing each input vector $\boldsymbol{x}$ by five copies corrupted by additive noise, of the form $\boldsymbol{x}+\epsilon$, where ϵ is a realization of a random vector of nine independent Gaussian variables with zero mean and standard deviation σ. Parameter σ allows to control the degree of corruption and was varied from 0 to 1 (the 9 input features have standard deviations between 0.003 and 1.4). A classifier was trained by our method using the (uncorrupted) learning set, and was used to classify corrupted samples using the decision rule presented in Section IV (case 1), for various values of λ_0. Fig. 12 shows the error and rejection rates as a function of λ_0, for $\sigma = 0$ (solid lines) and $\sigma = 0.5$ (dashed lines). Despite the importance of the corruption, the performance of the classifier is not dramatically affected. The error rate of 0.29 without rejection obtained with the original data can still be attained at the cost of rejecting approximately one third of the data. As a comparison, a multilayer perceptron with six hidden units was trained using the same learning data, and used to classify the corrupted test data using the Bayes decision rule

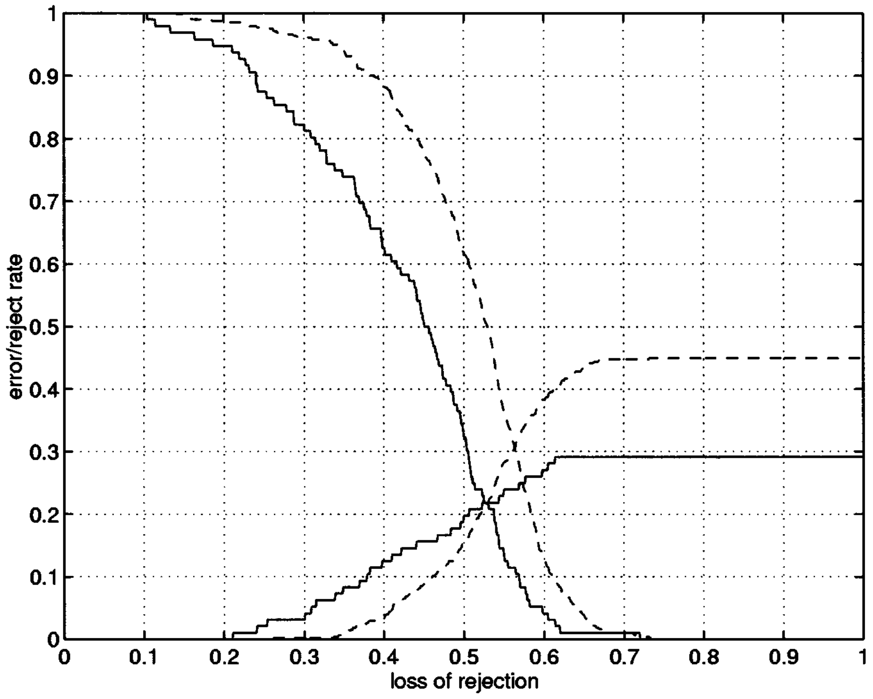

Fig. 12. Robustness experiment with the glass data: error rates (increasing curves) and rejection rates (decreasing curves) as a function of λ_0, for the ETC method applied to the uncorrupted data (solid lines) and corrupted data with $\sigma = 0.5$ (dashed lines).

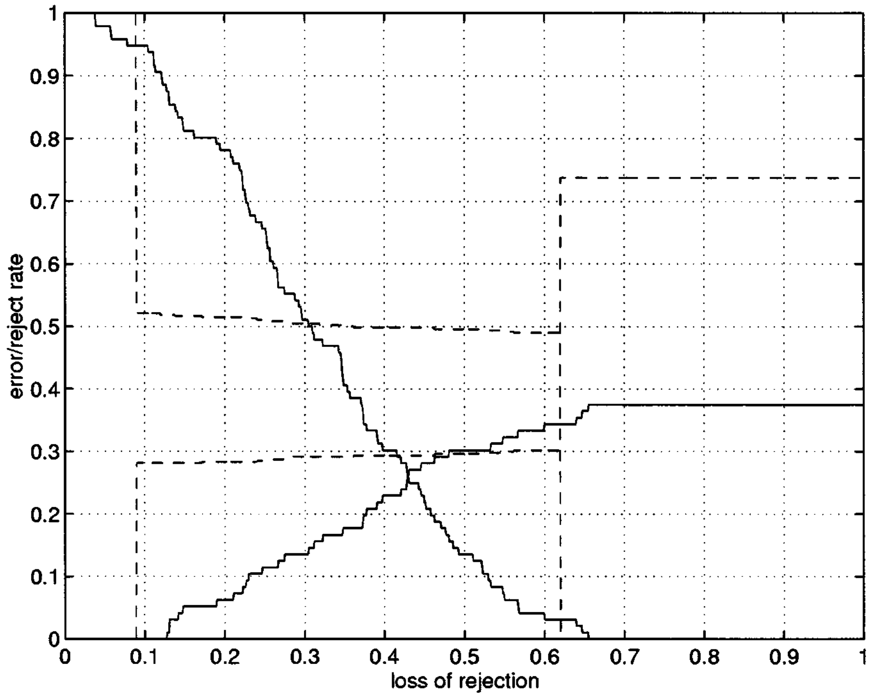

Fig. 13. Robustness experiment with the glass data: error rates (increasing curves) and rejection rates (decreasing curves) as a function of λ_0, for the MLP classifier (6 hidden units) applied to the uncorrupted data (solid lines) and corrupted data with $\sigma = 0.5$ (dashed lines).

with the same loss function as above. The error and rejection rates for $\sigma = 0$ (solid line) and $\sigma = 0.5$ (dashed line) are shown as a function of λ_0 in Fig. 13. With uncorrupted data, the MLP classifier has an error rate without rejection of 0.38, and rejection effectively allows to decrease the error rate to any desired level. However, the performance of the classifier collapses with the introduction of noise in the test data set. The achievable error rates are around 0.73 and 0.30 for 0 and 50% rejection rate, respectively. This classifier is thus less robust to outliers than the previous one, and its rejection rule is much less efficient in controlling the error rate. Fig. 14 shows the error rates obtained by the MLP and ETC methods, with 0, 0.1, and 0.3 rejection rates (the 0.1 and 0.3 rates are only obtainable by the ETC method), as a function of σ. These results confirm the much better performance of the ETC method in the presence of corrupted inputs.

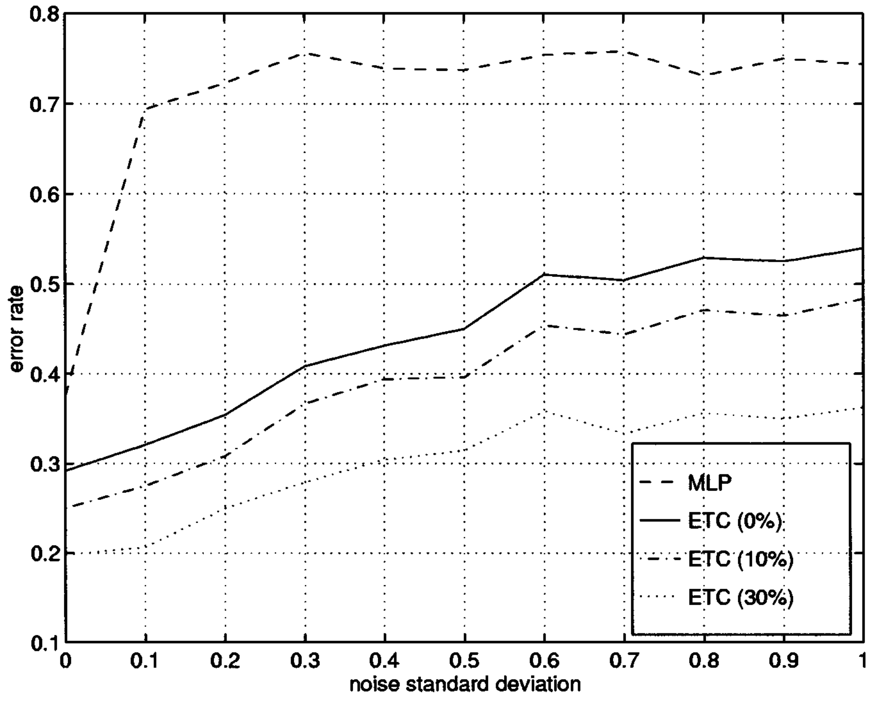

Fig. 14. Test error rates on the glass data as a function of noise standard deviation σ, for the MLP classifier and the ETC method with 0, 10, and 30% rejection rates.

C. Sensor Fusion Experiment

The main distinctive feature of our method as compared to conventional statistical or neural network classifiers concerns the nature of the outputs, which consist in a set of $M+1$ belief masses assigned to each individual class *and* to the set Ω of all classes. The fraction of the mass assigned to Ω reflects a lack of information available to make a decision, and can be used as an indication of the reliability of the classification procedure regarding the pattern under consideration. Sensor fusion provides a realistic context in which the value of this indication can be assessed. Assume that two classifiers C_1 and C_2 are trained separately to perform a classification task based on independent feature vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ provided by two sensors S_1 and S_2, respectively. A sensor fusion mechanism is used to combine the outputs from these two classifiers (Fig. 15). Then, in case of failure of one sensor, say S_1, the system should be able to detect the fact that classifier C_1 becomes less reliable, and should consequently decrease its influence on the final decision. If the reliability of classifier C_1 is correctly assessed, then the performance of the whole system should never get worse than the performance of classifier C_2 alone.

Before we describe our simulation experiment, let us examine how this fusion problem can be treated using two different frameworks: Bayes decision theory and D–S theory.

First, assume that C_1 and C_2 are probabilistic classifiers providing estimates for $P(\omega_i|\boldsymbol{x})$ and $P(\omega_i|\boldsymbol{y})$, respectively. If $\boldsymbol{x}$ and $\boldsymbol{y}$ are observed simultaneously, then the classification should be based on $P(\omega_i|\boldsymbol{x},\boldsymbol{y})$ for $i=1,2$, which can be computed as

$$P(\omega_i|\boldsymbol{x},\boldsymbol{y}) = \frac{f(\boldsymbol{x},\boldsymbol{y}|\omega_i)P(\omega_i)}{f(\boldsymbol{x},\boldsymbol{y})}. \tag{53}$$

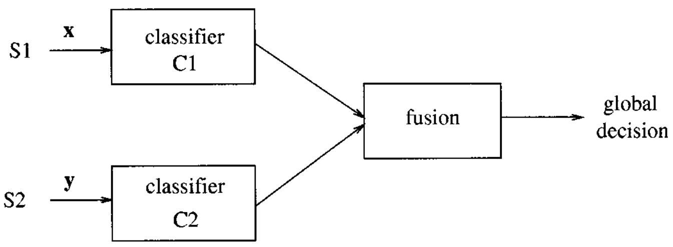

Fig. 15. The sensor fusion problem. Sensor S_1 provides feature vector $\boldsymbol{x}$ to classifier C_1, and sensor S_2 provides feature vector $\boldsymbol{y}$ to classifier C_2. The outputs of both classifiers are combined using either the Bayes rule for probabilistic classifiers, or the Dempster's rule for evidence-theoretic classifiers.

If $\boldsymbol{x}$ and $\boldsymbol{y}$ are assumed to be conditionally independent in each class, then $f(\boldsymbol{x},\boldsymbol{y}|\omega_i) = f(\boldsymbol{x}|\omega_i)f(\boldsymbol{y}|\omega_i)$, and we have

$$\begin{aligned} P(\omega_i|\boldsymbol{x},\boldsymbol{y}) &= \frac{f(\boldsymbol{x}|\omega_i)f(\boldsymbol{y}|\omega_i)P(\omega_i)}{f(\boldsymbol{x},\boldsymbol{y})} && (54)\\ &= \frac{P(\omega_i|\boldsymbol{x})P(\omega_i|\boldsymbol{y})}{P(\omega_i)}\ \frac{f(\boldsymbol{x})f(\boldsymbol{y})}{f(\boldsymbol{x},\boldsymbol{y})} && (55)\\ &\propto \frac{P(\omega_i|\boldsymbol{x})P(\omega_i|\boldsymbol{y})}{P(\omega_i)}. && (56)\end{aligned}$$

Hence, the probability distribution given $\boldsymbol{x}$ and $\boldsymbol{y}$ can be obtained by multiplying the posterior probabilities provided by each classifier, dividing by the priors, and renormalizing.

When C_1 and C_2 are evidence-theoretic classifiers, their outputs are BBA's that can be denoted $m(\cdot|\boldsymbol{x})$ and $m(\cdot|\boldsymbol{y})$, respectively. Since $\boldsymbol{x}$ and $\boldsymbol{y}$ are independent pieces of evidence, the Dempster's rule of combination can be applied:

$$m(\cdot|\boldsymbol{x},\boldsymbol{y}) = m(\cdot|\boldsymbol{x}) \oplus m(\cdot|\boldsymbol{y}). \tag{57}$$

Note that this combination rule does not make use of the prior probabilities, as was the case with the Bayes rule.

To compare the behavior of both data fusion procedures, we devised the following experiment. We considered a two-class problem (with equal priors) and two Gaussian random feature vectors $\boldsymbol{x} \in \mathbb{R}^5$ and $\boldsymbol{y} \in \mathbb{R}^3$ with following class-conditional probability distributions:

$$f(\boldsymbol{x}|\omega_i) \sim \mathcal{N}(\mu_i, \Sigma_i)$$
$$f(\boldsymbol{y}|\omega_i) \sim \mathcal{N}(\mu'_i, \Sigma'_i)$$

for $i = 1, 2$, with

$$\mu_1 = (1,1,1,1,1)^t \quad \mu_2 = (-1,-1,-1,-1,-1)^t$$
$$\Sigma_1 = \begin{pmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \Sigma_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$
$$\mu'_1 = (1,-1,1)^t \quad \mu'_2 = (-1,1,-1)^t$$
$$\Sigma'_1 = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{pmatrix} \quad \Sigma'_2 = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ were assumed to be conditionally independent, that is to say $f(\boldsymbol{x},\boldsymbol{y}|\omega_i) = f(\boldsymbol{x}|\omega_i)f(\boldsymbol{y}|\omega_i)$ for $i = 1, 2$.

Training and cross-validation sets of 60 and 100 labeled samples, respectively, were generated independently for each of the two classification tasks (class prediction from $\boldsymbol{x}$ and from $\boldsymbol{y}$, respectively). A test set of 5000 samples of $(\boldsymbol{x}, \boldsymbol{y})$ was also generated to evaluate the combination procedure.

Four different kinds of classifiers were trained on each learning set: multilayer perceptrons, RBF networks, quadratic discriminant analysis (i.e., the plug-in sample rule with heteroscedastic normal model [16]), and our method. For the three neural-network techniques, the number of hidden units was varied from 2 to 6, and the architecture yielding the lowest cross-validation error was selected. For each method, the two classifiers based on each "sensor" were then applied to the test data, and their outputs were merged using the appropriate combination rule (the Bayesian rule for MLP, RBF and quadratic classifiers, and Dempster's rule for our method).

In a first step, the whole procedure (data generation, training and testing) was repeated ten times. The results obtained by the four methods and by the Bayes classifier (the optimal decision rule based on the true probabilities and class-conditional densities) are shown in Table IV. The differences in error rates obtained by the four learning methods are small, as could be expected given the relatively low complexity of the classification problems involved. The RBF method has the largest error, while the best result is obtained by the quadratic rule, which has the advantage in this case of being based on the true probabilistic model (with estimated parameters).

TABLE IV
TEST ERROR RATES FOR THE SENSOR FUSION PROBLEM (UNCORRUPTED DATA)

Method	x alone	y alone	x and y
ETC	0.106	0.148	0.061
MLP	0.113	0.142	0.063
RBF	0.133	0.159	0.083
QUAD	0.101	0.141	0.049
BAYES	0.071	0.121	0.028

In a second step, the impact of the failure of one sensor on the classification accuracy of the system was simulated by corrupting the test set with additive noise. Each input vector $\boldsymbol{x}$ was replaced by a corrupted version $\boldsymbol{x} + \epsilon$, where ϵ is a realization of a normal random vector with zero mean and standard deviation σ. Parameter σ was varied from 0 to 10 to simulate various levels of noise. The test performances of the data fusion procedures based on each of the five classifiers are shown in Fig. 16. As can be seen, the procedures based on conventional classifiers are dramatically affected by the corruption of one of the two feature vectors. In contrast, the error rate of the procedure based on our method increases only moderately with noise level, and it ever gets higher than the error rate of the classifier based on feature vector $\boldsymbol{y}$ alone: the procedure is much more robust to strong changes in the distribution of input data.

To increase the robustness of conventional classifiers to outliers, Dubuisson and Masson [9] have suggested to modify the Bayes decision rule by adding a "distance rejection" mechanism: a pattern that was improbable under the current probabilistic model (or, equivalently, that was far from all training samples) is rejected. Following this idea, the four classifiers used in this study were modified as follows. For RBF and MLP classifiers, the distance rejection rule was

$$\text{if } ||\boldsymbol{x} - \boldsymbol{p}^i|| > d^i_{\max}\text{: reject } \boldsymbol{x},$$

where $\boldsymbol{p}^i$ is the nearest prototype to $\boldsymbol{x}$ and $d^i_{\max}$ is the maximum distance between prototype i and each of the training vectors. (Note that the prototypes generated by the RBF network were used to implement novelty detection in the MLP network.) For the quadratic classifier, the rule was

$$\text{if } \hat{f}(\boldsymbol{x}) < \hat{f}_{\min}\text{: reject } \boldsymbol{x},$$

where $\hat{f}(\boldsymbol{x})$ is the estimated probability density at $\boldsymbol{x}$ and $\hat{f}_{\min}$ is the lowest estimated density observed in the training set. A similar procedure was used for the Bayes classifier, but the true densities were used instead of the estimated ones.

The error rates obtained by each of the four modified procedures and the evidence-theoretic method for different noise levels are shown in Fig. 17. We can see that distance rejection effectively increases the robustness of the conventional methods,

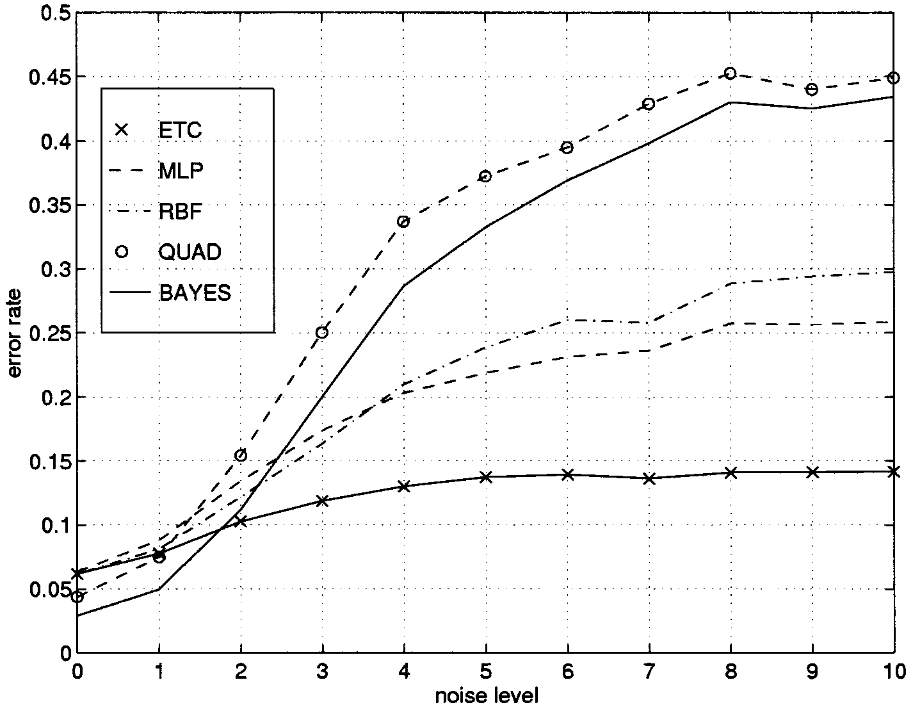

Fig. 16. Results of the sensor fusion experiment: test error rates as a function of noise standard deviation σ. The methods are evidence-theoretic classifier (ETC), multilayer-perceptron (MLP), radial basis function network (RBF), quadratic classifier (QUAD) and Bayes classifier (BAYES).

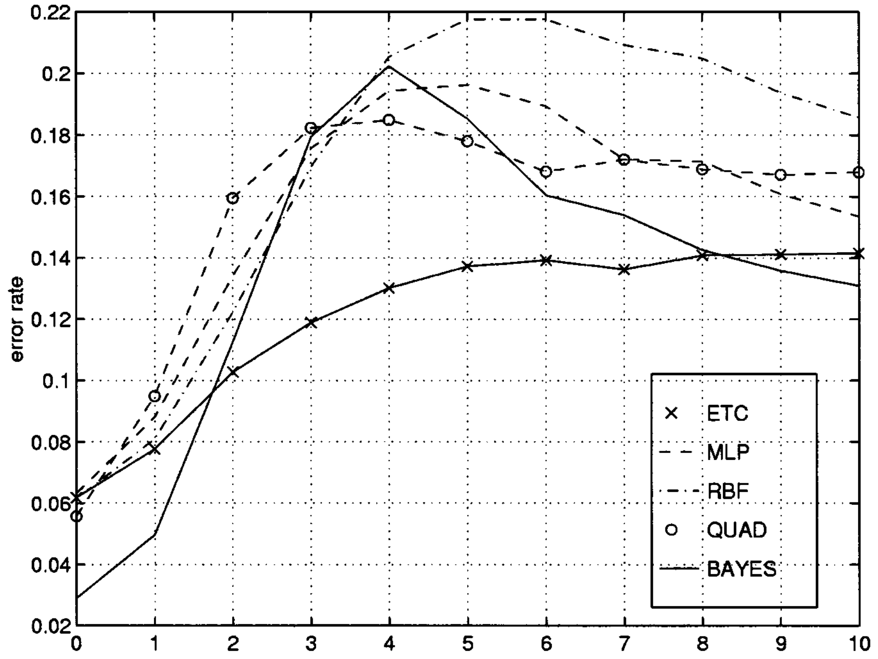

Fig. 17. Results of the sensor fusion experiment: test error rates as a function of noise standard deviation σ. The methods are the evidence-theoretic classifier (ETC), and conventional classifiers with distance rejection: multilayer-perceptron (MLP), radial basis function network (RBF), quadratic classifier (QUAD) and Bayes classifier (BAYES).

particularly for high levels of noise. However, the evidence-theoretic method still has the best performance, even when compared to the modified Bayes procedure, which uses additional information concerning the data generation process. This good behavior of our method may be explained by the effective use that it makes of the information about the reliability of each classifier, contained in the quantities $m(\Omega|\boldsymbol{x})$ and $m(\Omega|\boldsymbol{y})$.

VI. Conclusions

A new pattern classification method based on the Dempster–Shafer theory of evidence has been presented. This approach can be seen as combining ideas from supervised neural network models with local representation, and the evidence-theoretic k-NN rule described in [5]. As in RBF networks, classification is based on assessing the similarity of the input pattern with a set of reference patterns. This information is converted in the form of BBA's, which are then combined using Dempster's rule of combination. The system is trained by optimizing a performance criterion. This method can be implemented in a neural network architecture with two hidden layers and one output layer. After training, each of the neural network weights receives a natural interpretation. Each prototype is characterized by a weight vector $\boldsymbol{p}^i$, a receptive field parameter γ^i and a membership degree u_q^i to each class ω_q. The parameter α^i can be interpreted as an indication of the relative importance of the considered prototype in classifying new patterns. This method has exhibited excellent performance in several classification tasks as compared to some of the most widely used statistical and neural network classifiers, and has proved extremely robust to strong changes in the distribution of input data.

The introduction of Dempster–Shafer theory in statistical pattern recognition has several advantages, some of which were already mentioned in [5]. The classifier's output for each input vector has the form of a basic belief assignment, which essentially differs from a probability distribution in that it potentially assigns belief masses to all sets of classes. In particular, the mass assigned the whole frame of discernment Ω reflects the partial lack of information available for decision making. This indication can be used for rejecting the pattern under consideration if the associated uncertainty is too high, thus allowing to implement efficient novelty detection procedures. The same information could also be used qualitatively in situations where the classifier is essentially used as an aid to support human decision making. A further feature of Dempster–Shafer theory consists in the possibility of explicitly introducing unknown states of nature in the model, without making any assumption about these states. This makes the proposed method particularly suitable to complex classification tasks in which complete specification of the problem is difficult or impossible, such as encountered in medical diagnosis or system monitoring.

APPENDIX A

Gradient Calculation with Unnormalized Outputs: We first assume that the unnormalized output $\boldsymbol{m}$ is used. Let us first recall the propagation equations for input pattern $\boldsymbol{x}$:

For $i = 1, \cdots, n$:

$$d^i = ||\boldsymbol{x} - \boldsymbol{p}^i|| \tag{58}$$

$$s^i = \alpha^i \exp(-(\eta^i d^i)^2) \tag{59}$$

$$\alpha^i = (1 + \exp(-\xi^i))^{-1} \tag{60}$$

$$\boldsymbol{m}^i = (u_1^i s^i, \cdots, u_M^i s^i, 1 - s^i)^t \tag{61}$$

$$u_j^i = \frac{(\beta_j^i)^2}{\sum_{k=1}^{M} (\beta_k^i)^2}. \tag{62}$$

The output vector $\boldsymbol{m}$ is defined as

$$\boldsymbol{m} = (m(\{\omega_1\}), \cdots, m(\{\omega_M\}), m(\Omega)), \tag{63}$$

with $m = \cap_{i=1}^{n} m^i$. The error for pattern $\boldsymbol{x}$ is

$$E_\nu(\boldsymbol{x}) = \frac{1}{2} \sum_{q=1}^{M} (P_{\nu,q} - t_q)^2 \tag{64}$$

with

$$P_{\nu,q} = m_q + \nu m_{M+1}. \tag{65}$$

Derivatives w.r.t. β_j^i: The derivative of $E_\nu(\boldsymbol{x})$ w.r.t. β_j^i is given by

$$\frac{\partial E_\nu(\boldsymbol{x})}{\partial \beta_j^i} = \sum_{k=1}^{M} \frac{\partial E_\nu(\boldsymbol{x})}{\partial u_k^i} \frac{\partial u_k^i}{\partial \beta_j^i} \tag{66}$$

$$= \sum_{k \neq j} \frac{\partial E_\nu(\boldsymbol{x})}{\partial u_k^i} \frac{-2(\beta_k^i)^2 \beta_j^i}{\left(\sum_{l=1}^{M} (\beta_l^i)^2\right)^2} + \frac{\partial E_\nu(\boldsymbol{x})}{\partial u_j^i} \frac{2\beta_j^i \sum_{l=1}^{M} (\beta_l^i)^2 - 2(\beta_j^i)^3}{\left(\sum_{l=1}^{M} (\beta_l^i)^2\right)^2} \tag{67}$$

$$= \frac{2\beta_j^i}{\left(\sum_{l=1}^{M} (\beta_l^i)^2\right)^2} \left[\frac{\partial E_\nu(\boldsymbol{x})}{\partial u_j^i} \sum_{l=1}^{M} (\beta_l^i)^2 - \sum_{l=1}^{M} (\beta_l^i)^2 \frac{\partial E_\nu(\boldsymbol{x})}{\partial u_l^i}\right]. \tag{68}$$

Let us now compute $\partial E_\nu(\boldsymbol{x}) / \partial u_j^i$:

$$\frac{\partial E_\nu(\boldsymbol{x})}{\partial u_j^i} = \sum_{k=1}^{M} \frac{\partial E_\nu(\boldsymbol{x})}{\partial m_k} \frac{\partial m_k}{\partial u_j^i}. \tag{69}$$

Since the kth output m_k does not depend on u_j^i for $j \neq k$, this sum simplifies in

$$\frac{\partial E_\nu(\boldsymbol{x})}{\partial u_j^i} = \frac{\partial E_\nu(\boldsymbol{x})}{\partial m_j} \frac{\partial m_j}{\partial u_j^i} \tag{70}$$

$$= (P_{\nu,j} - t_j) \frac{\partial m_j}{\partial u_j^i}. \tag{71}$$

In order to express $\partial m_j / \partial u_j^i$, we use the commutativity and associativity of the $\cap$ operator to rewrite the output BBA m as the conjunctive combination of two terms, one of which does not depend on the parameters associated to prototype i:

$$m = m^i \cap \overline{m}^i, \tag{72}$$

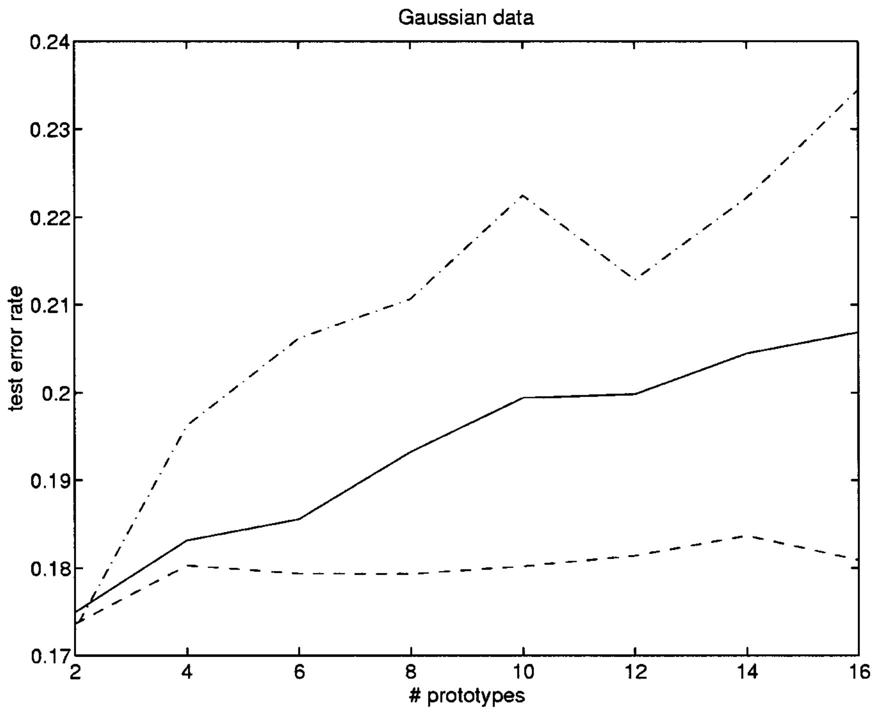

Fig. 18. Improvement of generalization by weight decay: average test error rates on Gaussian data as a function of number n of prototypes for RBF ($- \cdot -$), ETC with $\rho = 0$ (——) and ETC with $\rho = 0.01$ ($-\ -$).

with $\overline{m}^i = \cap_{k \neq i} m^k$. The vector

$$\boldsymbol{\overline{m}}^i = (\overline{m}^i(\{\omega_1\}), \cdots, \overline{m}^i(\{\omega_M\}), \overline{m}^i(\Omega))^t \tag{73}$$

can be computed by solving the following system of $M+1$ equations:

$$m_j = m^i_j(\overline{m}^i{}_j + \overline{m}^i{}_{M+1}) + m^i_{M+1}\overline{m}^i{}_j \qquad j = 1, \cdots, M \tag{74}$$

$$m_{M+1} = m^i_{M+1}\overline{m}^i{}_{M+1}, \tag{75}$$

yielding

$$\overline{m}^i{}_j = \frac{m_j - m_{M+1}m^i_j/m^i_{M+1}}{m^i_j + m^i_{M+1}} \qquad j = 1, \cdots, M \tag{76}$$

$$\overline{m}^i{}_{M+1} = \frac{m_{M+1}}{m^i_{M+1}}. \tag{77}$$

Note that the denominators in the above equations are always positive, since

$$m^i_{M+1} = 1 - s^i > 0.$$

With this notation:

$$\frac{\partial m_j}{\partial u^i_j} = s^i(\overline{m}^i{}_j + \overline{m}^i{}_{M+1}), \tag{78}$$

and

$$\frac{\partial E_\nu(\boldsymbol{x})}{\partial u^i_j} = (P_{\nu,j} - t_j)s^i(\overline{m}^i{}_j + \overline{m}^i{}_{M+1}). \tag{79}$$

Derivatives w.r.t. η^i, ξ, and p^i_j: The derivatives of $E_\nu(\boldsymbol{x})$ w.r.t. η^i, $\boldsymbol{xi}^i$, and p^i_j can be expressed as a function of $\partial E_\nu(\boldsymbol{x})/\partial s^i$:

$$\frac{\partial E_\nu(\boldsymbol{x})}{\partial \eta^i} = \frac{\partial E_\nu(\boldsymbol{x})}{\partial s^i}\,\frac{\partial s^i}{\partial \eta^i} = \frac{\partial E_\nu(\boldsymbol{x})}{\partial s^i}\,(-2\eta^i(d^i)^2 s^i) \tag{80}$$

$$\frac{\partial E_\nu(\boldsymbol{x})}{\partial \xi^i} = \frac{\partial E_\nu(\boldsymbol{x})}{\partial s^i}\,\frac{\partial s^i}{\partial \alpha^i}\,\frac{d}{\alpha^i}\,\frac{d}{\xi^i} = \frac{\partial E_\nu(\boldsymbol{x})}{\partial s^i}\,\exp(-(\eta^i d^i)^2)(1-\alpha^i)\alpha^i \tag{81}$$

$$\frac{\partial E_\nu(\boldsymbol{x})}{\partial p^i_j} = \frac{\partial E_\nu(\boldsymbol{x})}{\partial s^i}\,\frac{\partial s^i}{\partial p^i_j} = \frac{\partial E_\nu(\boldsymbol{x})}{\partial s^i}\,2(\eta^i)^2 s^i(x_j - p^i_j). \tag{82}$$

We therefore need to compute $\partial E_\nu(\boldsymbol{x})/\partial s^i$:

$$\frac{\partial E_\nu(\boldsymbol{x})}{\partial s^i} = \sum_{j=1}^{M} \frac{\partial E_\nu(\boldsymbol{x})}{\partial P_{\nu,j}}\,\frac{\partial P_{\nu,j}}{\partial s^i} = \sum_{j=1}^{M}(P_{\nu,j} - t_j)\left(\frac{\partial m_j}{\partial s^i} + \nu\frac{\partial m_{M+1}}{\partial s^i}\right). \tag{83}$$

Since

$$\frac{\partial m_j}{\partial s^i} = u_j^i(\overline{m}^i{}_j + \overline{m}^i{}_{M+1}) - \overline{m}^i{}_j \tag{84}$$

$$\frac{\partial m_{M+1}}{\partial s^i} = -\overline{m}^i{}_{M+1} \tag{85}$$

we have

$$\frac{\partial E_\nu(\boldsymbol{x})}{\partial s^i} = \sum_{j=1}^{M} (P_{\nu,j} - t_j)(u_j^i(\overline{m}^i{}_j + \overline{m}^i{}_{M+1}) - \overline{m}^i{}_j - \nu\overline{m}^i{}_{M+1}), \tag{86}$$

which can be introduced in (80)–(82).

Gradient Calculation with Normalized Outputs: If the normalized output vector $\boldsymbol{m}'$ is used, we consider the error function for input $\boldsymbol{x}$:

$$E'_\nu(\boldsymbol{x}) = \tfrac{1}{2} \sum_{j=1}^{M} (P'_{\nu,j} - t_j)^2 \tag{87}$$

with

$$P'_{\nu,j} = m'_j + \nu m'_{M+1} \tag{88}$$

$$\boldsymbol{m}' = \boldsymbol{m}/K \tag{89}$$

$$K = \sum_{k=1}^{M+1} m_k. \tag{90}$$

The expressions for the derivatives w.r.t. β_j^i, η^i, ξ, and p_j^i derived above are still valid if one replaces E_ν by E'_ν. However, we now have different expressions for $\partial E'_\nu(\boldsymbol{x})/\partial u_j^i$ and $\partial E'_\nu(\boldsymbol{x})/\partial s^i$.

As before:

$$\frac{\partial E'_\nu(\boldsymbol{x})}{\partial u_j^i} = \sum_{k=1}^{M} \frac{\partial E'_\nu(\boldsymbol{x})}{\partial m_k}\frac{\partial m_k}{\partial u_j^i} \tag{91}$$

$$= \frac{\partial E'_\nu(\boldsymbol{x})}{\partial m_j}\,\frac{\partial m_j}{\partial u_j^i}, \tag{92}$$

but we now have

$$\frac{\partial E'_\nu(\boldsymbol{x})}{\partial m_j} = \sum_{k=1}^{M} \frac{\partial E'_\nu(\boldsymbol{x})}{\partial P'_{\nu,k}}\,\frac{\partial P'_{\nu,k}}{\partial m_j} \tag{93}$$

$$= \sum_{k=1}^{M} (P'_{\nu,k} - t_k)\left(\frac{\partial m'_k}{\partial m_j} + \nu\,\frac{\partial m'_{M+1}}{\partial m_j}\right) \tag{94}$$

$$= \frac{1}{K^2}\sum_{k=1}^{M} (P'_{\nu,k} - t_k)(\delta_{kj}K - P_{\nu,k}) \tag{95}$$

$$= \frac{-1}{K^2}\sum_{k=1}^{M} (P'_{\nu,k} - t_k)P_{\nu,k} + \frac{1}{K}\,(P'_{\nu,j} - t_j). \tag{96}$$

Hence, using (78)

$$\frac{\partial E'_\nu(\boldsymbol{x})}{\partial u_j^i} = \frac{s^i(\overline{m}^i{}_j + \overline{m}^i{}_{M+1})}{K} \cdot \left((P'_{\nu,j} - t_j) - \frac{1}{K}\sum_{k=1}^{M}(P'_{\nu,k} - t_k)P_{\nu,k}\right). \tag{97}$$

Lastly

$$\frac{\partial E'_\nu(\boldsymbol{x})}{\partial s^i} = \sum_{k=1}^{M+1} \frac{\partial E'_\nu(\boldsymbol{x})}{\partial m_k}\,\frac{\partial m_k}{\partial s^i}. \tag{98}$$

In this sum, all the terms have already been calculated [(84), (85), and (96)], except $\partial E'_\nu(\boldsymbol{x})/\partial m_{M+1}$:

$$\frac{\partial E'_\nu(\boldsymbol{x})}{\partial m_{M+1}} = \sum_{k=1}^{M} \frac{\partial E'_\nu(\boldsymbol{x})}{\partial P'_{\nu,k}}\,\frac{\partial P'_{\nu,k}}{\partial m_{M+1}} \tag{99}$$

$$= \frac{1}{K^2}\sum_{k=1}^{M} (P'_{\nu,k} - t_k)(\nu K - P_{\nu,k}), \tag{100}$$

with completes the calculation of the gradient of $E'_\nu(\boldsymbol{x})$ and $E_\nu(\boldsymbol{x})$ w.r.t. all the parameters.

Complexity Considerations: Let us first consider the case of unnormalized outputs. For each $i = 1, \cdots, n$, the calculations can be done in the following order:

1) calculation of $\overline{\boldsymbol{m}}^i$ using (76) and (77),
2) calculation of $\partial E_\nu(\boldsymbol{x})/\partial u_j^i$ for $j = 1, \cdots, M$ using (79),
3) calculation of the terms $\Sigma_{l=1}^{M}\ (\beta_l^i)^2$ and $\Sigma_{l=1}^{M}\ (\beta_l^i)^2\ \partial E_\nu(\boldsymbol{x})/\partial u_l^i$ in (68),
4) calculation of $\partial E_\nu(\boldsymbol{x})/\partial \beta_j^i$ for $j = 1, \cdots, M$ using (68),
5) calculation of $\partial E_\nu(\boldsymbol{x})/\partial s^i$, $\partial E_\nu(\boldsymbol{x})/\partial \xi^i$, and $\partial E_\nu(\boldsymbol{x})/\partial \eta^i$ using (80), (81), and (86),
6) calculation of $\partial E_\nu(\boldsymbol{x})/\partial p_j^i$ for $j = 1, \cdots, P$ using (82).

Steps 1–5 require $O(M)$ arithmetic operations, and step 6 can be performed using $O(P)$ operations. Hence, the complexity of the whole gradient calculation in that case is $O(n(M+P))$.

In the case of normalized outputs, the only difference resides in the calculation of the derivatives w.r.t. u_j^i and s^i. The former can be computed using a constant number of operations using (97) after calculation of the term $(1/K)\ \Sigma_{k=1}^{M}\ (P'_{\nu,k} - t_k)P_{\nu,k}$, while the latter requires $O(M)$ operations. Hence, the complexity of the gradient calculation is also $O(n(M+P))$ in that case.

APPENDIX B

A common approach for improving generalization in neural network classifiers consists in penalizing complexity by means of a weight-decay term added to the error function [19], [27]. The definition of such a term is particularly easy in our model since the parameter α^i controls the influence of each hidden unit i on the output: when $\alpha^i = 0$, m^i is the vacuous BBA and consequently has no effect on the classification. Therefore, a natural choice for a penalty term is $\Sigma_{i=1}^{n} \alpha^i$. The criterion to be minimized then becomes

$$C_\nu = E_\nu + \rho \sum_{i=1}^{n} \alpha^i \tag{101}$$

or

$$C'_\nu = E'_\nu + \rho \sum_{i=1}^{n} \alpha^i, \tag{102}$$

where ρ is a regularization parameter. Parameter ρ can be either determined by cross-validation or fixed to a given value chosen from experience.

The gradient of C_ν (respectively, C'_ν) is identical to that of E_ν (respectively, E'_ν) except for

$$\frac{\partial C_\nu}{\partial \alpha^i} = \frac{\partial E_\nu}{\partial \alpha^i} + \rho. \tag{103}$$

The efficiency of this method in eliminating irrelevant prototypes was tested on two-class Gaussian data. Class 1 was composed of two Gaussian clusters with means $(1\ 0\ 0)^t$ and $(0\ 1\ 0)^t$ and the same covariance matrix I (the identity matrix). Class 2 was composed of a single cluster with mean $(0\ 0\ 1)^t$ and covariance matrix $4I$. The prior probability of each cluster was equal to 1/3.

The generalization performances of RBF networks and evidence-theoretic classifiers (with $\rho = 0$ and $\rho = 0.01$) were tested for numbers of prototypes ranging from 2 to 16, for ten independent training sets of 60 samples. The probabilities of misclassification were estimated using a test set of size 5000. The initialization procedure was the same as in the previous example.

The average test error rates as a function of the number of prototypes are plotted in Fig. 18. The best performance is attained with two prototypes by all three methods. The error rate increases as a function of n for RBF networks and evidence-theoretic method without regularization, but remains stable for evidence-theoretic method with regularization.

REFERENCES

[1] E. B. Baum and F. Wilczek, "Supervised learning of probability distributions by neural networks," in *Neural Information Processing Systems*, D. Z. Anderson, Ed. New York: AIP, 1988, pp. 52–61.

[2] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth, 1984.

[4] C. K. Chow, "On optimum recognition error and reject tradeoff," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 41–46, 1970.

[5] T. Denœux, "A k-nearest neighbor classification rule based on Dempster-Shafer theory," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 804–813, May 1995.

[6] ——, "Analysis of evidence-theoretic decision rules for pattern classification," *Pattern Recognit.*, vol. 30, no. 7, pp. 1095–1107, 1997.

[7] ——, "Application du modèle des croyances transférables en reconnaissance de formes," *Trait. Signal*, vol. 14, no. 5, pp. 443–451, 1998.

[8] D. H. Deterding, "Speaker normalization for automatic speech recognition," Ph.D. dissertation, Univ. Cambridge, U.K., 1989.

[9] B. Dubuisson and M. Masson, "A statistical decision rule with incomplete knowledge about classes," *Pattern Recognit.*, vol. 26, no. 1, pp. 155–165, 1993.

[10] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[11] J. Friedman, "Multivariate adaptive regression splines (with discussion)," *J. Amer. Stat. Assoc.*, vol. 19, no. 1, pp. 1–141, 1991.

[12] T. J. Hastie and R. J. Tibshirani, "Nonparametric regression and classification—Part II: Nonparametric classification," in *From Statistics to Neural Networks*, V. Cherkassly, J. H. Friedman, and H. Wechsler, Eds. Berlin, Germany: Springer-Verlag, 1994, pp. 70–82.

[13] T. J. Hastie, R. J. Tibshirani, and A. Buja, "Flexible discriminant analysis by optimal scoring," AT&T Bell Labs., Tech. Rep., 1993.

[14] J. Kohlas and P.-A. Monney, *A Mathematical Theory of Hints. An Approach to the Dempster–Shafer Theory of Evidence*. Berlin, Germany: Springer-Verlag, 1995.

[15] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, pp. 1464–1480, Sept. 1990.

[16] G. J. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*. New York: Wiley, 1992.

[17] P. M. Murphy and D. W. Aha, "UCI repository of machine learning databases [machine-readable data repository]," Dept. Inform. Comput. Sci., Univ. California,, Irvine, 1994.

[18] T. Poggio and F. Girosi, "A theory of networks for approximation and learning," M.I.T, Tech. Rep. A.I. Memo no. 1140, 1988.

[19] R. Reed, "Pruning algorithms: A survey," *IEEE Trans. Neural Networks*, vol. 4, pp. 740–747, Sept. 1993.

[20] B. D. Ripley, "Flexible nonlinear approaches to classification," in *From Statistics to Neural Networks*, V. Cherkassly, J. H. Friedman, and H. Wechsler, Eds. Berlin, Germany: Springer-Verlag, 1994, pp. 105–126.

[21] A. J. Robinson, "Dynamic error propagation networks," Ph.D. dissertation, Cambridge Univ., Cambridge, U.K., 1989.

[22] G. Rogova, "Combining the results of several neural network classifiers," *Neural Networks*, vol. 7, no. 5, pp. 777–781, 1994.

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

[24] P. Smets, "The combination of evidence in the transferable belief model," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 447–458, May 1990.

[25] ——, "The transferable belief model for quantified belief representation," in *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, D. M. Gabbay and P. Smets, Eds. Dordrecht, The Netherlands: Kluwer, 1998, vol. 1, pp. 267–301.

[26] P. Smets and R. Kennes, "The transferable belief model," *Artif. Intell.*, vol. 66, pp. 191–243, 1994.

[27] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, "Generalization by weight-elimination with application to forecasting," *in Neural Information Processing 3*, pp. 875–882, 1991.

[28] L. Xu, A. Krzyzak, and C. Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 3, pp. 418–435, 1992.

[29] R. R. Yager, M. Fedrizzi, and J. Kacprzyk, *Advances in the Dempster–Shafer Theory of Evidence*. New York: Wiley, 1994.

[30] L. M. Zouhal and T. Denœux, "An evidence-theoretic k-NN rule with parameter optimization," *IEEE Trans. Syst., Man, Cybern. C*, vol. 28, pp. 263–271, May 1998.

Thierry Denœux graduated as an engineer in 1985 from the Ecole Nationale des Ponts et Chaussées, Paris, France, and received the Ph.D. degree from the same institution in 1989. He received the "Habilitation à diriger des Recherches" degree from the Institut National Polytechnique de Lorraine, France, in 1996.

From 1989 to 1992, he was a Research Scientist at the Lyonnaise des Eaux Research Center, Compiègne, France, where he was in charge of research projects concerning the application of neural networks to forecasting and process monitoring. He joined the Compiègne University of Technology in 1992, where he is now a Professor in the Department of Information Processing Engineering. His research interests include statistical pattern recognition, uncertainty modeling and data fusion.