Support Vector Regression of membership functions and belief functions - Application for pattern recognition

Hicham Laanaya[(*)1,2], Arnaud Martin[2], Driss Aboutajdine[1], and Ali Khenchaf[2],

[1]GSCM-LRIT, Mohammed V-Agdal University, Faculty of sciences of Rabat, Morocco,

[(*)1]hicham.laanaya@gmail.com, Arnaud.Martin@ensieta.fr

[2]ENSIETA-E^{3}I^2-EA3876, 2, rue François Verny 29806 Brest Cedex 9, France

Abstract

Caused by many applications during the last few years, many models have been proposed to represent imprecise and uncertain data. These models are essentially based on the theory of the theory of fuzzy sets, the theory of possibilities and the theory of belief functions. These two first theories are based on the membership functions and the last one on the belief functions. Hence, it could be interesting to learn these membership and belief functions from data and then we can, for example, deduce the class for a classification task. Therefore, we propose in this paper a regression approach based on the statistical learning theory of Vapnik. The membership and belief functions have the same properties; that we take as constraints in the resolution of our convex problem in the support vector regression.

The proposed approach is applied in a pattern recognition context to evaluate its efficiency. Hence, the regression of the membership functions and the regression of the belief functions give two kinds of classifiers: a fuzzy SVM and a belief SVM. From the learning data, the membership and belief functions are generated from two classical approaches given respectively by fuzzy and belief k-nearest neighbors. Therefore, we compare the proposed approach, in terms of classification results, with these two k-nearest neighbors and with support vector machines classifier.

Keywords: SVR, SVM, regression, belief functions, membership functions.

1 Introduction

The study of uncertain environments using more and more complex systems is necessary in many applications. We must then evaluate these systems from uncertain and imprecise data. Several choices are suitable to tackle these imperfections (uncertainty and imprecision): either we try to remove them, which requires a understanding of the physics of the sensors used for data acquisition or we seek to develop a robust system to deal with these imperfections or again we try to model them.

A precise model of uncertain and imprecise data can be carried out using the theories of uncertainty such as the fuzzy sets theory [48], the theory of possibilities [49, 12] or the theory of belief functions [7, 38]. These theories of uncertainty are based either on membership functions or on belief functions in order to represent imprecise and uncertain data. Hence, these functions are used in many applications dealing with uncertain environments such as pattern recognition, clustering, data mining, assessment, tracking, control, etc (for some reviews of such applications see for example [19, 50] for the fuzzy sets theory and [40, 11] for the theory of belief functions).

Consequently, it could be interesting for a lot of applications to learn these membership and belief functions from data. Therefore, a regression approach based on the statistical learning theory of Vapnik [45, 46] is proposed to learn membership and belief functions. The goal of the paper is not to propose a new fuzzy regression [42, 30, 47] or belief regression [33].

Support Vector Machines (SVM), introduced by Vapnik [45, 46], are first a binary classification method. The simplicity of this approach and its capacity of extension to non linear classification involved significant development and wide use of SVM, particularly in linear regression [41, 13]. This is the reason why, we used a support vector regression (SVR) for the regression of the membership functions and the belief functions that have similar properties which are introduced as constraints in the optimization problem. The proposed method is different from the classical multiple regression by SVM due to the constraint over the outputs.

The evaluation of the proposed approach is made in a pattern recognition context. For pattern recognition, many methods were developed within the framework of the theories of uncertainty using existing methods such as neural networks, k-nearest neighbors or decision trees, giving new approaches for classification such as fuzzy classifiers [5, 17, 18] or belief classifiers [9, 8, 10, 44, 26].

Various attempts were proposed to integrate fuzziness in the SVM. Indeed, learning can be

carried out using weighted membership functions [14]. In [16, 1, 43], a membership function is introduced to deal with the ambiguity in the shaded regions where multi-classification is ambiguous. Finally, [15] proposes a regression on fuzzy triangular numbers which presents some differences with our approach.

Belief functions were also used in the SVM classifier context, in order to combine the binary classifiers according to different strategies such as one-against-one or one-against-all [35, 4, 27].

The regression of the membership functions and the regression of the belief functions applied for pattern recognition, give two kinds of classifiers: a fuzzy SVM and a belief SVM. From the learning data, the membership functions are generated from the classical approach given in [18], and the belief functions from the approach given in [8]. Therefore, we naturally compare the proposed approach, in terms of classification results, with the fuzzy k-nearest neighbors [18], with the belief k-nearest neighbors [8] and with the support vector machines classifier. The tests are conducted on both generated data and real-world data (sonar images that are particularly hard to classify). The obtained results show the interest of the regression of membership and belief functions in the pattern recognition domain.

In section 2, we recall the principle of the SVR and SVM, as a particular case of SVR. Then, in section 3, we define the notations of membership functions and belief functions for the understanding of the rest of the document. In section 4, we describe the suggested approach of regression by support vector machines. Finally, this method is compared and discussed in section 5, using generated data and sonar images.

2 Support Vector Regression

We assume to get training data $\{(x_t, y_t), t = 1, \ldots, l\}$, $x_t \in \mathbb{R}^d$, $y_t \in \mathbb{R}$. The goal of regression is to find a function f that fits the relationship between the input x and the output $y = f(x)$. For ε-SV regression proposed by Vapnik [45, 46], using ε-insensitive loss function, we have to find a function f that has at most ε deviation between the learning target y_t and $f(x_t)$ for all the training data.

2.1 Linear regression

We begin the presentation of the support vector regression with the case of linear functions f. Consider the problem of approximation of y_t by $f(x_t)$ for the set of data $\{(x_t, y_t)\}_{t=1,\ldots,l}$,

$x_t \in \mathbb{R}^d$ and $y_t \in \mathbb{R}$, with a linear function:

$$f(x) = w.x + b. \tag{1}$$

The optimal function is given by the minimum of the functional

$$\frac{1}{2}\|w\|^2 + C\sum_{t=1}^{l}(\xi_t + \xi_t^*), \tag{2}$$

under the constraints

$$\begin{cases} y_t - w.x_t - b & \leq \quad \varepsilon + \xi_t \\ w.x_t - y_t + b & \leq \quad \varepsilon + \xi_t^* \\ \xi_t, \xi_t^* & \geq \quad 0 \end{cases} \tag{3}$$

where C is a user defined constant, and ξ_t, ξ_t^* are slack variables representing upper and lower constraints on the outputs of the system.

For an ε-insensitive loss function [45, 46]

$$|\xi|_\varepsilon := \begin{cases} 0 & \text{if } |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise.} \end{cases} \tag{4}$$

the solution is given by

$$\max_{\alpha,\alpha^*} W(\alpha, \alpha^*) = \max_{\alpha,\alpha^*} -\frac{1}{2}\sum_{t=1}^{l}\sum_{t'=1}^{l}(\alpha_t - \alpha_t^*)(\alpha_{t'} - \alpha_{t'}^*)x_t.x_{t'} + \sum_{t=1}^{l}\alpha_t(y_t - \varepsilon) - \alpha_t^*(y_t + \varepsilon) \tag{5}$$

thus, the solution $\bar{\alpha}, \bar{\alpha}^*$ are given by the optimization of

$$\operatorname*{argmin}_{\alpha,\alpha^*} \frac{1}{2}\sum_{t=1}^{l}\sum_{t'=1}^{l}(\alpha_t - \alpha_t^*)(\alpha_{t'} - \alpha_{t'}^*)x_t.x_{t'} - \sum_{t=1}^{l}(\alpha_t - \alpha_t^*)y_t + \sum_{t=1}^{l}(\alpha_t + \alpha_t^*)\varepsilon \tag{6}$$

under the constraints,

$$0 \leq \alpha_t, \alpha_t^* \leq C, \quad t = 1, \ldots, l \tag{7}$$

$$\sum_{t=1}^{l}(\alpha_t - \alpha_t^*) = 0. \tag{8}$$

Solving the Equation (5) under the constraints (7), (8), determines the Lagrange multipliers, α, α^*, and the regression function is given by Equation (1), where

$$\bar{w} = \sum_{t=1}^{l}(\alpha_t - \alpha_t^*)x_t \tag{9}$$

$$\bar{b} = -\frac{1}{2}\bar{w}.(x_r + x_s). \tag{10}$$

where x_r and x_s are any vector that satisfy $\alpha_r > 0$ and $\alpha_s > 0$.

The Karush-Kuhn-Tucker (KKT) conditions are satisfied by the solution, and given by

$$\bar{\alpha_t}\bar{\alpha_t^*} = 0, \quad t = 1, \ldots, l. \tag{11}$$

Therefore, the support vectors are points where exactly one of the Lagrange multipliers is greater than zero.

2.2 Non linear regression

If the relation between the inputs and outputs is non-linear, non-linear mapping using a kernel can be used to map the data into a high dimensional feature space where linear regression is performed. The non-linear SVR solution is given by

$$\max_{\alpha,\alpha^*} W(\alpha, \alpha^*) = \max_{\alpha,\alpha^*} \sum_{t=1}^{l} \alpha_t(y_t - \varepsilon) - \alpha_t^*(y_t + \varepsilon) - \frac{1}{2}\sum_{t=1}^{l}\sum_{t'=1}^{l}(\alpha_t - \alpha_t^*)(\alpha_{t'} - \alpha_{t'}^*)K(x_t.x_{t'}) \tag{12}$$

with constraints:

$$0 \leq \alpha_t, \alpha_t^* \leq C, \ t = 1, \ldots, l \tag{13}$$

$$\sum_{t=1}^{l}(\alpha_t - \alpha_t^*) = 0. \tag{14}$$

Solving Equation (12) with constraints Equations (13), (14), determines the Lagrange multipliers, α_t, α_t^*, and the regression function is given by

$$f(x) = \sum_{t \in \text{SVs}} (\alpha_t - \alpha_t^*)K(x_t, x) + \bar{b} \tag{15}$$

where $\text{SVs} = \{t; 0 < \alpha_t, \alpha_t^* < C\}$ and

$$\bar{b} = -\frac{1}{2}\sum_{t=1}^{l}(\alpha_t - \alpha_t^*)(K(x_t, x_r) + K(x_t, x_s)). \tag{16}$$

The most used kernels are the polynomial kernel $K(x, x_t) = (x.x_t + 1)^p$, $p \in \mathbb{N}$, and the Gaussian kernel $K(x, x_t) = e^{-\gamma\|x - x_t\|^2}$, $\gamma \in \mathbb{R}^+$.

2.3 Support Vector Regression: Application to classification

The classical approach of support vector machines proposed by Vapnik [45, 46] can be seen as a support vector regression application. In this case, we have $\varepsilon = 1$ and we consider outputs lying in $\{-1, +1\}$. The problem defined by the Equation (2) with the constraints given by the Equation (3), becomes

$$\min_{w,b,\xi} J(w, \xi), \tag{17}$$

under the constraints given for all t by

$$\begin{cases} y_t(w.x_t + b) \geq 1 - \xi_t \\ \xi_t \geq 0 \end{cases} \tag{18}$$

where

$$J(w, \xi) = \frac{1}{2} \parallel w \parallel^2 + C \sum_{t=1}^{l} \xi_t, \tag{19}$$

and C is a constant chosen by the user. High value of C places extreme penalty on errors. This problem is solved in the same way as the previous section, but the Lagrange multipliers satisfies $0 \leq \alpha_t \leq C$.

The solution to the optimization problem of Equation (17) subject to the constraints (18) is given by the saddle point of the Lagrangian

$$L_P(w, b, \xi, \alpha, \mu) = J(w, \xi) + \sum_{t=1}^{l} \alpha_t [1 - \xi_t - y_t(w.x_t + b)] - \sum_{t=1}^{l} \mu_t \xi_t. \tag{20}$$

where α, μ are the Lagrange multipliers. The Lagrangian has to be minimized with respect to w, b, ξ and maximized with respect to α, μ.

The minimum with respect to w, b, ξ of the Lagrangian L_P, is given by

$$\begin{cases} \dfrac{\partial L_P}{\partial b} = 0 \quad \Longrightarrow \quad w = \displaystyle\sum_{t=1}^{l} \alpha_t y_t x_t \\ \dfrac{\partial L_P}{\partial w} = 0 \quad \Longrightarrow \quad \displaystyle\sum_{t=1}^{l} \alpha_t y_t = 0 \\ \dfrac{\partial L_P}{\partial \xi} = 0 \quad \Longrightarrow \quad \alpha_t + \mu_t = C. \end{cases} \tag{21}$$

Hence, the solution to the problem is given by

$$\bar{\alpha} = \underset{\alpha}{\operatorname{argmin}} \frac{1}{2} \sum_{t=1}^{l} \sum_{t'=1}^{l} \alpha_t \alpha_{t'} y_t y_{t'} x_t.x_{t'} - \sum_{v=1}^{l} \alpha_v \tag{22}$$

with constraints,

$$0 \leq \alpha_t \leq C \quad t = 1, \ldots, l$$
$$\sum_{t=1}^{l} \alpha_t y_t = 0. \tag{23}$$

In order to classify a new pattern x, we use the decision function given by

$$f(x) = \text{sign}\left(\sum_{t \in \text{SVs}} y_t \alpha_t^0 x_t.x + b_0 \right), \tag{24}$$

where $\text{SVs} = \{t; \alpha_t^0 > 0\}$ for the separable case and $\text{SVs} = \{t; 0 < \alpha_t^0 < C\}$ for the non-separable case, are the sets of all support vectors.

In the non linear cases, the principle of the SVM is to map the data in a high dimensional space (using a kernel function) in which the data can be linearly separated.

In this case, the optimization problem of Equation (22) becomes

$$\bar{\alpha} = \underset{\alpha}{\text{argmin}} \frac{1}{2} \sum_{t=1}^{l} \sum_{t'=1}^{l} \alpha_t \alpha_{t'} y_t y_{t'} K(x_t, x_{t'}) - \sum_{v=1}^{l} \alpha_v, \tag{25}$$

where $K(x, x')$ is the kernel function used for the non-linear mapping into feature space, and the constraints are unchanged

$$0 \leq \alpha_t \leq C \quad t = 1, \ldots, l$$
$$\sum_{t=1}^{l} \alpha_t y_t = 0. \tag{26}$$

Solving Equation (25) under the constraints (26), determines the Lagrange multipliers, and thus, the classification of a new pattern x is given by the decision function

$$f(x) = \text{sign}\left(\sum_{t \in \text{SVs}} y_t \alpha_t^0 K(x, x_t) + b_0 \right). \tag{27}$$

3 Theories of Uncertainty

In many situations, it is important to model uncertain environments from complex information and also from complex systems. The acquisition systems and the difficulty related to the scene make that we handle uncertain and imprecise data. For an identification task, we must use

classifiers that can manage such data. The theories of uncertainty such as the fuzzy subsets theory [48], the theory of possibilities [12] or the theory of belief functions [7, 38] allow the modeling of uncertain and imprecise data.

Many classifiers were developed in the framework of these theories such as fuzzy neural networks [5], belief neural networks [9, 26], belief k-nearest neighbors [8, 51], fuzzy k-nearest neighbors [17, 18] or belief decision trees [10, 44].

These theories of uncertainty are based either on membership functions or on belief functions in order to represent imprecise and uncertain data. We recall bellow these functions and give an example of fuzzy k-nearest neighbors and an example of belief k-nearest neighbors.

3.1 The membership functions

The membership functions have been introduced by [48] to describe a fuzzy membership to a class. The membership of an observation x to a class C_i among N_c classes, is given by a function $\mu_i(x)$ with values in $[0, 1]$. Moreover, we can consider without loss of generality, that:

$$\sum_{i=1}^{N_c} \mu_i(x) = 1. \tag{28}$$

This assumption corresponds to a closed world (*i.e.* there is no other possible classes than the N_c classes C_i). If it is not verified then we can add an N_c+1th class to obtain the Equation (28).

In this case, we consider the fuzzy classes: a pattern may belong to various classes, with different degrees of membership. In the case of crisp classes, the knowledge of the membership of the pattern is described by a possibility function. Note that we can consider membership functions as numerically equivalent to possibility distribution functions [49].

3.1.1 Example of membership function for fuzzy k-nearest neighbors

We present here the approach proposed in [18]. Initially, we calculate the membership function of a training vector x_t given by:

$$\mu_i(x_t) = \frac{k_i(x_t)}{k_f}, \tag{29}$$

where k_f is the number of the nearest neighbors chosen for the fuzzy neighborhood V_{k_f} and $k_i(x_t)$ is the number of neighbors that belong to class C_i, where $V_{k_f}(x_t)$ is the set of the k_f nearest neighbors to the training vector x_t. Therefore, we calculate the membership function of

a vector x to classify:

$$\mu_i(x) = \frac{\sum_{j=1}^{k} \frac{\mu_i(x_{t_j})}{\|x - x_{t_j}\|^2}}{\sum_{j=1}^{k} \frac{1}{\|x - x_{t_j}\|^2}}. \tag{30}$$

where $x_{t_j}, j = 1, \ldots, k$ are the k-nearest neighbors of x and the distance used here is the euclidean distance.

The membership class of x is then decided in a classical way as the class giving the maximum of the membership functions.

This approach provides a way to learn the membership functions from data, we can refer to [29] for other methods to generate membership functions in the context of pattern recognition.

3.2 Belief functions

The theory of belief functions manipulates the mass functions. A mass function m is the mapping from elements of the power set 2^Θ onto $[0, 1]$. 2^Θ is the set of all the disjunctions of the frame of discernment $\Theta = \{C_1, \ldots, C_{N_c}\}$, where C_i represents the hypothesis "the observation belongs to the class i". It fulfills the condition:

$$\sum_{X \in 2^\Theta} m(X) = 1, \tag{31}$$

where $m(.)$ represents the mass function. The first difficulty is to define these mass functions according to the problem. From these mass functions, other belief functions can be defined, such as the credibility functions (bel), representing a minimal belief, and such as the plausibility functions (pl) representing a maximal belief.

In order to preserve maximum of information, it is preferable to stay on a credal level (*i.e.* to handle belief functions) during the information combination stage to make the decision on the belief functions resulting from this combination. As $\text{bel}(A) \leq \text{betP}(A) \leq \text{pl}(A)$, the pignisitic probability, introduced by [39], is often considered as a compromise. The pignistic probability is given for all $X \in \Theta$ by:

$$\text{betP}(X) = \sum_{Y \in 2^\Theta, Y \neq \emptyset} \frac{|X \cap Y|}{|Y|} \frac{m(Y)}{1 - m(\emptyset)}. \tag{32}$$

3.2.1 Example of belief function for belief k-nearest neighbors

Denœux [8] proposes an estimation of the mass functions using a distance model, defined for each nearest neighbors by:

$$\begin{cases} m_j(C_i|x_j)(x) = \alpha_i e^{\gamma_i d^2(x,x_j)} \\ m_j(\Theta|x_j)(x) = 1 - \alpha_i e^{\gamma_i d^2(x,x_j)} \end{cases} \tag{33}$$

where C_i is the associated class to x_j, and $\{x_j\}_{j=1,...,k}$ are the k nearest training patterns to x. The distance d used is the euclidean distance. α_i is a discounting coefficient, and and γ_i is a coefficient of normalization which can be optimized [51]. The k mass functions calculated for each x are combined by the normalized orthogonal rule of Dempster given for all $X \in 2^\Theta$, $X \neq \emptyset$ by:

$$m_{\text{DS}}(X) = \frac{m_{\text{Conj}}(X)}{1 - m_{\text{Conj}}(\emptyset)}, \tag{34}$$

with

$$m_{\text{Conj}}(X) = \sum_{Y_1 \cap ... \cap Y_k = X} m_1(Y_1)...m_k(Y_k), \tag{35}$$

and $m_{\text{DS}}(\emptyset) = 0$. The decision is then taken by the maximum of the mass functions which is, in this case, equivalent to the maximum of pignistic probability because the only focal elements are singletons and ignorance $m(\Theta)$.

This example provides a mean to generate belief functions from learning data. Another approach also widely used is proposed in [3].

4 Support vector regression of membership and belief functions

Support vector machines also provide a way to perform a linear regression to predict continuous functions [45, 46]. In the case of functions with values in $\mathbb{R}^N$ various solutions were proposed, for example in [41, 13] a simple regression is carried out on each dimension assuming that the outputs are independent. For the membership functions or the belief functions the condition (31) requires that each dimension of the function should be predicted jointly and not independently like in [32, 36]. Identical constraints of the membership functions and the belief functions (with values in $[0, 1]$ and with sum equal to 1), allow us to rewrite the multiple linear and non linear regression while extending the work of [32, 36].

Hong and Hwang [15] propose a regression on fuzzy triangular numbers. They examine a particular case of a multiple regression in dimension 3, but the fundamental difference with our approach lies on the constraints on the membership functions and the belief functions.

4.1 Linear regression

We assume to get training patterns $x_t \in \mathbb{R}^d$ and the associated outputs $y_t \in \mathbb{R}^N$, where $N = N_c$ is the number of classes in the case of membership functions and $N = 2^{N_c}$ in the case of mass functions. By multiple linear regression, we seek to find a functional $f = (f_1, \ldots, f_N)$ where the f_n are linear, of the form $f_n(x) = w_n.x + b_n$. We attempt to find this functional such as for the (x_t, y_t) of the training database, $|y_{t,n} - w_n.x_t + b^n|$ does not exceed a fixed ε for all n. It is possible to consider an $\varepsilon_{t'}$ for each direction. With this formulation we assume that all the points are inside the ε-tube. In the general case, we associate a factor for each point outside the ε-tube. If the components of each y_t are independent, the approach suggested in [41] can be considered. In our case, we do not have independence since $y_{t,n} \in [0, 1]$ and verify $\sum_{n=1}^{N} y_{t,n} = 1$ for all t. Thus, the convex optimization problem is similar to the one exposed in Equation (2), and we have to minimize:

$$\frac{1}{2}\sum_{n=1}^{N}\|w_n\|^2 + C\sum_{n=1}^{N}\sum_{t=1}^{l}(\xi_{t,n} + \xi^*_{t,n}), \tag{36}$$

under the constraints given for all t and all n:

$$\begin{cases} y_{t,n} - w_n.x_t - b_n & \leq & \varepsilon + \xi_{t,n}, \\ w_n.x_t + b_n - y_{t,n} & \leq & \varepsilon + \xi^*_{t,n}, \\ \sum_{n=1}^{N}(w_n.x_t + b_n) & = & 1, \\ w_n.x_t + b_n & \geq & 0, \\ w_n.x_t + b_n & \leq & 1, \\ \xi_{t,n}, \xi^*_{t,n} & \geq & 0. \end{cases} \tag{37}$$

The Equation (36) differs from the proposed approach in [32, 36], by the slack variables $\xi_{t,n}, \xi^*_{t,n}$, used to limit the error; here they can be different for each direction.

Thus, the lagrangian is given by:

$$\begin{aligned} L &= \frac{1}{2}\sum_{n=1}^{N}\|w_n\|^2 + C\sum_{n=1}^{N}\sum_{t=1}^{l}(\xi_{t,n}+\xi^*_{t,n}) \\ &- \sum_{n=1}^{N}\sum_{t=1}^{l}(\eta_{t,n}\xi_{t,n}+\eta^*_{t,n}\xi^*_{t,n}) \\ &- \sum_{n=1}^{N}\sum_{t=1}^{l}\alpha_{t,n}(\varepsilon+\xi_{t,n}-y_{t,n}+w_n.x_t+b_n) \\ &- \sum_{n=1}^{N}\sum_{t=1}^{l}\alpha^*_{t,n}(\varepsilon+\xi^*_{t,n}+y_{t,n}-w_n.x_t-b_n) \\ &- \sum_{n=1}^{N}\sum_{t=1}^{l}\beta_{t,n}(w_n.x_t+b_n) \\ &- \sum_{n=1}^{N}\sum_{t=1}^{l}\beta^*_{t,n}(1-w_n.x_t-b_n) \\ &- \sum_{t=1}^{l}\gamma_t\left(1-\sum_{n=1}^{N}(w_n.x_t+b_n)\right) \end{aligned} \tag{38}$$

where η, α, β and γ the Lagrange multipliers and are positive.

At the saddle point of the Lagrangian L, we have for all t and all n, $\partial L/\partial b_n = 0$, $\partial L/\partial w_n = 0$, $\partial L/\partial \xi_{t,n} = 0$ et $\partial L/\partial \xi^*_{t,n} = 0$. Thus:

$$\begin{cases} \sum_{t=1}^{l}\sigma_{t,n} = 0, \\ w_n = \sum_{t=1}^{l}\sigma_{t,n}x_t, \\ \eta_{t,n} = C-\alpha_{t,n}, \\ \eta^*_{t,n} = C-\alpha^*_{t,n}, \end{cases} \tag{39}$$

with $\sigma_{t,n} = \alpha_{t,n} - \alpha^*_{t,n} + \beta_{t,n} - \beta^*_{t,n} - \gamma_t$.

By integrating these Equations (39) in the Lagrangian (*cf.* Equation (38)), the problem is then to maximize:

$$L = -\frac{1}{2}\sum_{n=1}^{N}\sum_{t,t'=1}^{l}\sigma_{t,n}\sigma_{t',n}x_t.x_{t'} - \sum_{n=1}^{N}\sum_{t=1}^{l}\beta^*_{t,n} + \frac{\gamma_t}{N} - \sum_{n=1}^{N}\sum_{t=1}^{l}\alpha^*_{t,n}(\varepsilon+y_{t,n})$$

$$-\sum_{n=1}^{N}\sum_{t=1}^{l}\alpha_{t,n}(\varepsilon - y_{t,n}) \tag{40}$$

under the constraints:

$$\begin{cases} \sum_{t=1}^{l}\sigma_{t,n} = 0, \\ \alpha_{t,n} \in [0, C], \\ \alpha^*_{t,n} \in [0, C], \\ \beta_{t,n} \geq 0, \\ \beta^*_{t,n} \geq 0, \\ \gamma_t \geq 0. \end{cases} \tag{41}$$

Finally, to predict the output of the n^{th} component $\tilde{y}_n$, of a new element x, we use the equation:

$$\tilde{y}_n = \sum_{t=1}^{l}\sigma_{t,n}x_t.x + b_n, \tag{42}$$

where b_n is deduced from the Kuhn, Karush and Tucker conditions:

$$\begin{cases} \alpha_{t,n}(\varepsilon + \xi_{t,n} - y_{t,n} + w_n.x_t + b_n) &= 0, \\ \alpha^*{}_{t,n}(\varepsilon + \xi^*_{t,n} + y_{t,n} - w_n.x_t - b_n) &= 0, \\ (C - \alpha_{t,n})\xi_{t,n} &= 0, \\ (C - \alpha^*_{t,n})\xi^*_{t,n} &= 0, \\ \beta_{t,n}(w_n.x_t + b_n) &= 0, \\ \beta^*_{t,n}(1 - w_n.x_t - b_n) &= 0. \end{cases} \tag{43}$$

If for a given t_0, $\alpha_{t_0,n} \in]0, C[$ then, $\xi_{t_0,n} = 0$, which imply $b_n = y_{t_0,n} - w_N.x_{t_0} - \varepsilon$, the same reasoning for α^* gives $b_n = y_{t_0,n} - w_N.x_{t_0} + \varepsilon$.

4.2 Non linear regression

Until now, we assumed that the relation is linear between the output y_t and the input x_t. In a similar way to the SVR and SVM, we can represent the data using a kernel in a higher dimensional space. The resulting algorithm is similar to the linear one, except that every the scalar product between the training database samples is replaced by a kernel: the scalar product $x.x'$ becomes $K(x, x')$. Therefore we apply a linear regression in the feature space.

The optimization problem of Equation (40) becomes

$$\begin{aligned} L &= -\frac{1}{2}\sum_{n=1}^{N}\sum_{t,t'=1}^{l}\sigma_{t,n}\sigma_{t',n}K(x_t,x_{t'}) - \sum_{n=1}^{N}\sum_{t=1}^{l}\beta^*_{t,n} + \frac{\gamma_t}{N} - \sum_{n=1}^{N}\sum_{t=1}^{l}\alpha^*_{t,n}(\varepsilon + y_{t,n}) \\ &\quad -\sum_{n=1}^{N}\sum_{t=1}^{l}\alpha_{t,n}(\varepsilon - y_{t,n}) \end{aligned} \tag{44}$$

under the same constraints:

$$\begin{cases} \sum_{t=1}^{l}\sigma_{t,n} = 0, \\ \alpha_{t,n} \in [0, C], \\ \alpha^*_{t,n} \in [0, C], \\ \beta_{t,n} \geq 0, \\ \beta^*_{t,n} \geq 0, \\ \gamma_t \geq 0. \end{cases} \tag{45}$$

Solving the Equation (44) under the constraints (45) determines the Lagrange multipliers, and to predict the output $\tilde{y}$, of an element x, we consider:

$$\tilde{y}_n = \sum_{t=1}^{l}\sigma_{t,n}K(x,x_t) + b_n. \tag{46}$$

4.3 Remarks

From this regression approach on the membership functions or the belief functions, we obtain a fuzzy SVM classifier by making the decision through the maximum of the membership functions or a belief SVM classifier by making the decision through the maximum of pignistic probability (32). We note that the value of ε controls the sensitivity of the results to the outputs (membership functions or belief functions).

For this new approach, the choice of the parameters (C, ε and eventually the kernel parameters) is crucial. The choice of ε can be done by a similar method as in [37] using a new parameter ν that asymptotically controls the number of training samples being inside the ε-tube. Another way is the use of genetic algorithms to optimize all these parameters [20].

The training stage for regression is expensive considering the dimension of the optimization problem to solve. One solution to this problem is the use of decomposition methods, where at

each iteration, we solve a subproblem of small size [31]. In the extreme case we use a subproblem of dimension 2, we use the decomposition by *Sequential Minimal Optimization* (SMO) [34].

4.4 SMO for belief and fuzzy regression by SVM

The maximization problem defined by Equations (40), (41) is a quadratic optimization problem (See Appendix A for more details). It can be written as:

$$\begin{cases} \text{Min} \quad \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ \mathbf{A}\mathbf{x} = 0 \\ 0 \leq \mathbf{x} \leq \mathbf{C} \end{cases} \tag{47}$$

The matrix $\mathbf{Q}$ is of great dimension, traditional methods for resolution of such problem is difficult, however, it is difficult to store such matrix (*e.g.* we need 1.7GB of memory storage for a problem with $l = 1000$ and $N = 3$). Chunking methods are used to deal with this kind of problem. Unlike the most optimization methods where, at each iteration, the $\mathbf{x}$ is updated, decomposition methods use only one sub-vector of $\mathbf{x}$ at each iteration. This subset is noted as B, it gives a subproblem to be solved for each iteration. The extreme case is SMO where B contains only two elements.

The advantage of using two pairs for B is that for a subset of 2 elements, the optimization subproblem can be analytically solved without using quadratic optimization software. The basic structure of the method is presented by algorithm 1.

The quadratic optimization problem (47) is similar to that of SVM and SVR. In [22, 23] we found a proof of convergence of this decomposition method for Support Vector Machines. The implementation of our approach is based on *libSVM* developed by Chih-Chung Chang and Chih-Jen Lin [6].

5 Experimental results

In this section, we present the experiments carried out on artificial and real-world data to show the behavior of our approach. Two artificial databases with classes not highly overlapped and a third real-world database with highly overlapped classes. We give in the following section the measures used to evaluate the classification results for generated database (results given in Section 5.2) and for the real database (results given in Section 5.3).

Algorithm 1 Decomposition algorithm by SMO

1. Find $\mathbf{x}^1$ satisfying the constraints of (47) (or feasible solution), $n \leftarrow 1$.
2. If $\mathbf{x}^1$ is an optimal solution of (47), we stop. Else, we search two elements $B = \{t, t'\} \subset \{1, \ldots, L\}$ (with $N' = 5Nl$). Let $\bar{B} \equiv \{1, \ldots, N'\} \setminus B$ and $\mathbf{x}_B^n$ and $\mathbf{x}_{\bar{B}}^n$ be two sub-vectors of $\mathbf{x}^n$.
3. Solve the subproblem of minimization according to $\mathbf{x}_B$ of:

$$\begin{aligned} J(\mathbf{x}_B) &:= \frac{1}{2}[\mathbf{x}_B^T \quad (\mathbf{x}_{\bar{B}}^n)^T] \begin{bmatrix} \mathbf{Q}_{BB} & \mathbf{Q}_{B\bar{B}} \\ \mathbf{Q}_{\bar{B}B} & \mathbf{Q}_{\bar{B}\bar{B}} \end{bmatrix} \begin{bmatrix} \mathbf{x}_B \\ \mathbf{x}_{\bar{B}} \end{bmatrix} + [\mathbf{c}_B^T \quad (\mathbf{c}_{\bar{B}}^n)^T] \begin{bmatrix} \mathbf{x}_B \\ \mathbf{x}_{\bar{B}} \end{bmatrix} \\ &= \frac{1}{2}\mathbf{x}_B^T \mathbf{Q}_{BB} \mathbf{x}_B + (\mathbf{c}_B + \mathbf{Q}_{B\bar{B}} \mathbf{x}_{\bar{B}}^n)^T \mathbf{x}_B + \text{Constant} \\ &= \frac{1}{2}[\mathbf{x}_t \quad \mathbf{x}_{t'}] \begin{bmatrix} \mathbf{Q}_{tt} & \mathbf{Q}_{tt'} \\ \mathbf{Q}_{t't} & \mathbf{Q}_{t't'} \end{bmatrix} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{x}_{t'} \end{bmatrix} + (\mathbf{c}_B + \mathbf{Q}_{B\bar{B}} \mathbf{x}_{\bar{B}}^n)^T \begin{bmatrix} \mathbf{x}_t \\ \mathbf{x}_{t'} \end{bmatrix} + \text{Constant} \end{aligned} \tag{48}$$

under the constraints

$$\begin{cases} 0 \leq \mathbf{x}_t, \mathbf{x}_{t'} \leq \mathbf{C}, \\ \mathbf{y_t} \mathbf{x}_t + \mathbf{y}_{t'} \mathbf{x}_{t'} = \mathbf{y}_{\bar{B}}^T \mathbf{x}_{\bar{B}}^n, \end{cases} \tag{49}$$

with $\begin{bmatrix} \mathbf{Q}_{BB} & \mathbf{Q}_{B\bar{B}} \\ \mathbf{Q}_{\bar{B}B} & \mathbf{Q}_{\bar{B}\bar{B}} \end{bmatrix}$ is a permutation of the matrix $\mathbf{Q}$ and $\mathbf{C} \in \{C, \infty\}$

4. $\mathbf{x}_B^{n+1}$ is the solution of (48) and $\mathbf{x}_{\bar{B}}^{n+1} \equiv \mathbf{x}_{\bar{B}}^n$.

   Do $n \leftarrow n + 1$ and go to step 2.

5.1 Evaluation of the classification

Typically, classification algorithms are evaluated using a confusion matrix. This matrix (CM) is composed by the numbers $\mathrm{CM}_{i_1 i_2}$ of elements of the class i_1 which are classified in class i_2. We can normalize this matrix to get rates that are easy to interpret:

$$\mathrm{NCM}_{i_1 i_2} = \frac{\mathrm{CM}_{i_1 i_2}}{\sum_{i_k=1}^{N_c} \mathrm{CM}_{i_1 i_k}} = \frac{\mathrm{CM}_{i_1 i_2}}{N_{i_1}}, \tag{50}$$

N_c is the number of classes and N_{i_1} is the number of elements of class i_1. We calculate from this confusion matrix a normalized vector of good classification rates (CR):

$$\mathrm{CR}_i = \mathrm{NCM}_{ii}, \tag{51}$$

The average of the classification rate CR_m is given by:

$$\mathrm{CR}_m = \frac{\sum_{i=1}^{N_c} CM_{ii}}{N}, \tag{52}$$

where N represents the total number of data used by the classification algorithm. The vector of probability error (PE) is defined by:

$$\mathrm{PE}_{i_1} = \frac{1}{2}\left(\sum_{i_2=1, i_2 \neq i_1}^{N_c} \mathrm{NCM}_{i_1 i_2} + \sum_{i_2=1, i_2 \neq i_1}^{N_c} \frac{\mathrm{NCM}_{i_2 i_1}}{N_c - 1} \right), \tag{53}$$

Hence, we define the weighted sum of the probabilities PE_{i_1} by:

$$\mathrm{PE}_m = \frac{\sum_{i=1}^{N_c} N_i \mathrm{PE}_i}{N}. \tag{54}$$

The evaluation of the classification of images can be difficult, especially on sonar images that we use here [24, 25]. The result of image classification can be visually evaluated by comparing it with the ground truth (reference image). However, to evaluate the classification algorithm, we must consider several possible configurations.

We proposed in [24, 25] an approach that takes into account the certainty and the imprecision of the expert: on real-world data such as sonar images, we have to consider tiles to extract texture features that can contain more than one class. Moreover, sonar experts are not sure of the real class of the considered tile on the image due to the uncertain environment and to the sensor. In order to take into account the tiles that contain more than one class, the confusion matrix is updated as follow: if a tile, of size $t_L \times t_L$, contains more than one class is classified as a class i_1, then, for $i_2 = 1, \ldots, N_c$ $\mathrm{NCM}_{i_1 i_2}$ will be $\mathrm{NCM}_{i_1 i_2} + N_{i_2}/t_L^2$, where N_{i_2} is the number of pixels i_2 on the tile. Secondly, in order to take into account the certainty level of the experts on the data, if a tile of class i_1 with certainty given by a weight $w \in [0, 1]$, is classified into class i_2 then $\mathrm{NCM}_{i_1 i_2}$ will be $\mathrm{NCM}_{i_1 i_2} + w$.

5.2 Generated data

As a simple experiment, we used generated Gaussian data. We defined on these data a membership function using Equation (30) and a mass function calculated with the normalized orthogonal

combination (Equation (34)) of the mass functions given by Equation (33). Both SVM classifiers based on the regression of the membership and belief functions are compared to the fuzzy k-nearest neighbors (with $k = 5$ and $k_f = 7$) and to belief k-nearest neighbors (with $k = 5$). The values of k and k_f are arbitrarily selected with the same value of k for both k-nearest neighbors approaches and a k_f greater than k. We developed a modified version of *libSVM* to integrate our approach using SMO for problem optimization. The fuzzy SVM and the belief SVM classifiers based on the regression are noted respectively f-SVR and b-SVR. They are parameterized with $C = 1$, $\varepsilon = 0.1$ and a linear kernel (the default values of the modified *libSVM*).

The first database is generated from two Gaussian distributions in $\mathbb{R}^2$, respectively with a mean of $\mu_1 = (1 \quad 0)^T$ and a covariance matrix of $\Sigma_1 = 0.25\mathbf{Id}$ for the first class and a mean of $\mu_2 = (-1 \quad 0)^T$ and a covariance matrix of $\Sigma_2 = \mathbf{Id}$ for the second class, where $\mathbf{Id}$ is the identity matrix. Each class contains 2000 samples (1000 for training and 1000 for validation). We repeated this operation 20 times to have a good estimation of classification rates for the different approaches.

Firstly we give some graphical illustrations to show the properties of the proposed approaches. Figure 1 gives the used data and the corresponding results (the assigned classes) with b-SVR and f-SVR using a linear kernel. We can see that b-SVR and f-SVR allow also to linearly separate classes like in SVM.

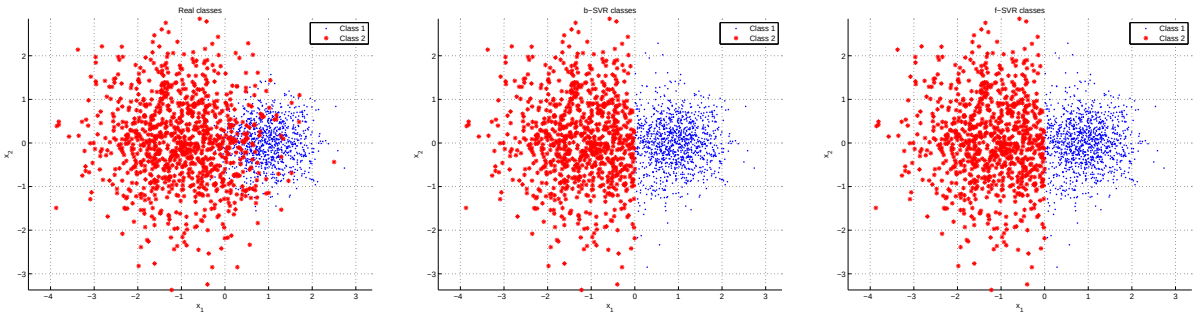

Figure 1: Real classes and found classes with b-SVR and f-SVR

We draw in Figure 2 the maximum of belief, the ignorance ($m(\Theta)$) and the mass of each class for the b-SVR approach. We note the variation of the maximum of belief and the ignorance in the border between class 1 and class 2. We see also that the mass function $m(C)$ is maximum on points that belong to class C and is small for data in the border between the two classes.

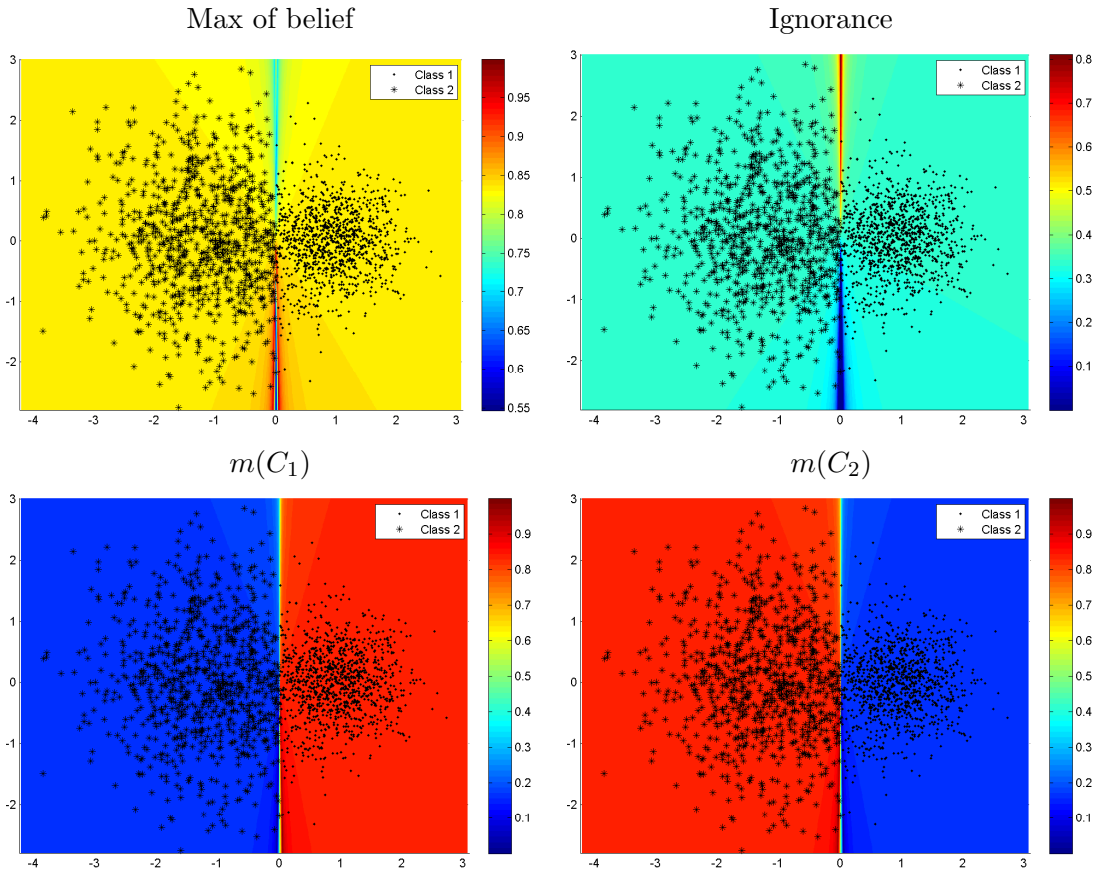

Figure 2: Max of belief, the ignorance and the mass to each class for each test data point

To illustrate the properties of the f-SVR approach, we drew the membership function to each class for each test data point (*cf.* Figure 3). We see that the membership function is equal to 0 or 1 and thus the maximum of membership is equal to 1 for all test data point (*cf.* Figure 3 and Figure 7).

Here, the used data are not linearly separable. We used a Gaussian kernel with $\sigma = 0.1$ to search for a non-linear relationship between data and belief functions or membership functions. Figure 4 gives the classes found by b-SVR and f-SVR.

Figure 5 shows that the mass function is maximum for data near the center of each class. Figure 6 gives the obtained results using f-SVR with a Gaussian kernel. These results show that the use of a kernel for both approaches allows for searching a nonlinear boundary between

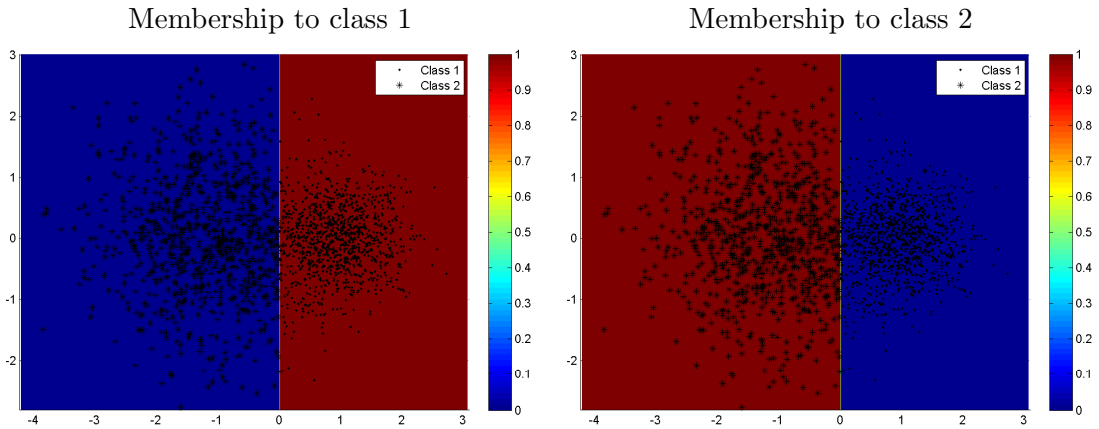

Figure 3: Membership function to each class for each test data point

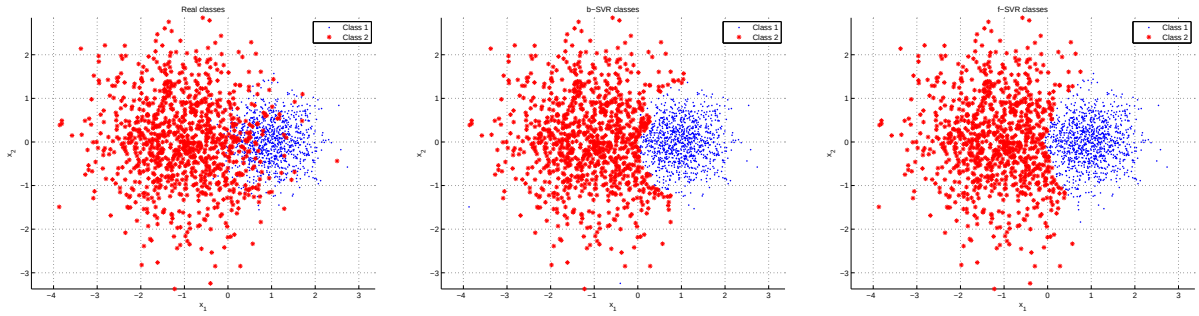

Figure 4: Real classes and found classes with b-SVR and f-SVR

classes.

We draw for both case, linear and Gaussian kernel, the distributions of the maximum of belief, the ignorance and the maximum of membership. Figure 7 shows that, for the linear kernel, the maximum of belief lies between 0.8 and 0.85 for most test data points and the ignorance lies between 0.3 and 0.4. We note also that the maximum of membership is equal to 1 for all test data points (*cf.* Figure 7) and the maximum of belief lies between 0.5 and 1 for most test data point and the ignorance is equal to 0 in the case of the Gaussian kernel.

The obtained normalized confusion matrices (NCM), the classification rates (CR_m) and the probability of errors (PE_m) for fuzzy k-nearest neighbors, belief k-nearest neighbors, the b-SVR and f-SVR classifiers with a linear regression with $C = 1$, $\varepsilon = 0.1$ and SVM with $C = 1$ and

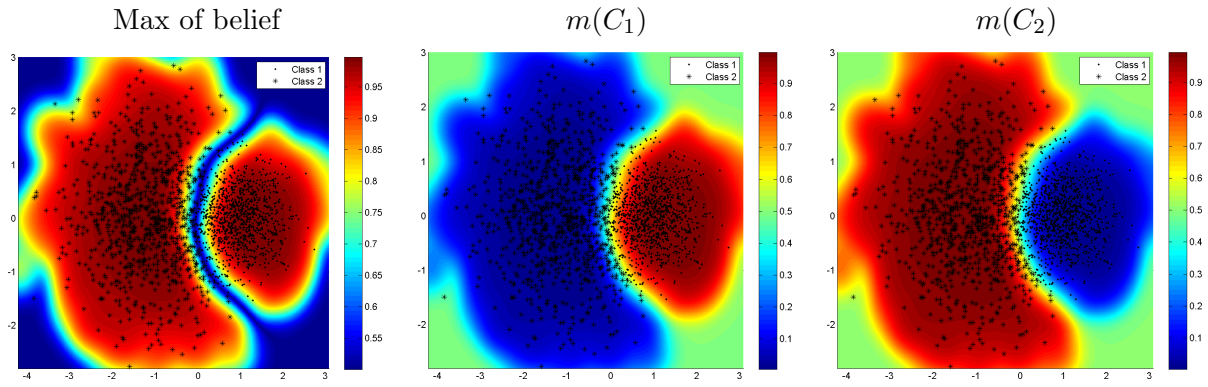

Figure 5: Max of belief, the ignorance and the mass to each class for each test data point

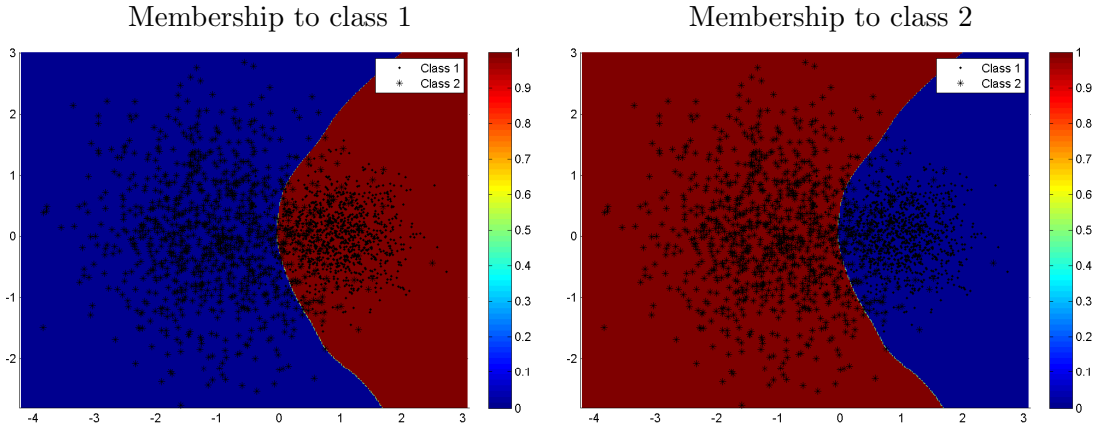

Figure 6: Membership function to each class for each test data point

Gaussian kernel with $\gamma = 1/2$ are presented in Table 1. On these generated data, a linear regression is enough for the b-SVR and f-SVR classifiers, whereas the SVM classifier needs the use of a kernel to obtain better results.

We obtained a classification rate of 93.79±1.67% for the b-SVR classifier and 93.90±1.66% for the f-SVR classifier. SVM gives an accuracy rate similar to that obtained by b-SVR and f-SVR classifier with 93.89±1.66% of examples are well classified. These rates are better than those found by the belief k-nearest neighbors (93.11±1.74%) and the fuzzy k-nearest neighbors (92.10±1.84%). These results show that the b-SVR, the f-SVR classifiers perform slightly better

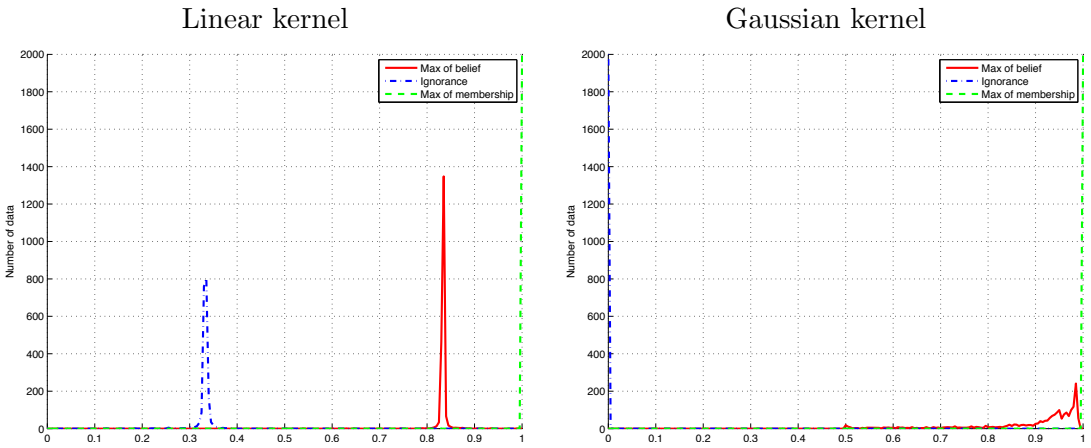

Figure 7: Distribution of the maximum of belief, the ignorance and the maximum of membership

	NCM (%)	CR_m (%)	PE_m (%)
belief k-nn	$\begin{pmatrix} 92.92 & 7.08 \\ 6.70 & 93.30 \end{pmatrix}$	93.11 ± 1.74	6.89 ± 1.41
fuzzy k-nn	$\begin{pmatrix} 91.94 & 8.06 \\ 7.73 & 92.27 \end{pmatrix}$	92.10 ± 1.84	7.90 ± 1.51
b-SVR	$\begin{pmatrix} 94.34 & 5.66 \\ 6.75 & 93.25 \end{pmatrix}$	93.79 ± 1.67	6.21 ± 1.33
f-SVR	$\begin{pmatrix} 94.32 & 5.68 \\ 6.52 & 93.48 \end{pmatrix}$	$\mathbf{93.90 \pm 1.66}$	6.10 ± 1.32
SVM	$\begin{pmatrix} 94.19 & 5.81 \\ 6.41 & 93.59 \end{pmatrix}$	93.89 ± 1.66	6.11 ± 1.32

Table 1: Confusion matrices for belief k-nearest neighbors, fuzzy k-nearest neighbors, b-SVR and f-SVR classifiers and SVM for 2D artificial data

than the two k-nearest neighbors approaches. However the results are not significantly different for all approaches.

The second database is generated from three Gaussian distributions of dimension 3 using, respectively, a mean of $(1\ 1\ 1)^T$, $(-1\ 1\ 0)^T$ and $(0\ -1\ 1)^T$ and covariance matrices of $0.25\mathbf{Id}$, $0.75\mathbf{Id}$ and $0.5\mathbf{Id}$. Each database, for 20 epochs used, contains 3000 vectors (1500 for training and 1500 for test where each class contains 500 examples).

The obtained results for fuzzy k-nearest neighbors, belief k-nearest neighbors, the b-SVR and f-SVR with linear regression with $C = 1$ and $\varepsilon = 0.1$ and SVM (using the one-versus-one approach) with $C = 1$ and Gaussian kernel with $\gamma = 1/3$ are presented in Table 2. Here also, a linear regression is enough for the b-SVR and f-SVR classifiers, whereas the SVM classifier need a kernel to obtain better results.

	NCM (%)	CR_m (%)	PE_m (%)
belief k-nn	$\begin{pmatrix} 94.16 & 3.17 & 2.67 \\ 5.11 & 88.43 & 6.46 \\ 3.27 & 5.09 & 91.64 \end{pmatrix}$	91.41 ± 1.06	8.59 ± 0.82
fuzzy k-nn	$\begin{pmatrix} 93.48 & 3.60 & 2.92 \\ 5.46 & 87.66 & 6.89 \\ 3.63 & 5.69 & 90.68 \end{pmatrix}$	90.61 ± 1.10	9.39 ± 0.86
b-SVR	$\begin{pmatrix} 97.66 & 0.98 & 1.37 \\ 7.38 & 86.10 & 6.51 \\ 4.60 & 3.51 & 91.90 \end{pmatrix}$	$\mathbf{91.89 \pm 1.03}$	8.11 ± 0.80
f-SVR	$\begin{pmatrix} 97.58 & 1.05 & 1.37 \\ 7.33 & 86.08 & 6.58 \\ 4.46 & 3.56 & 91.97 \end{pmatrix}$	91.88 ± 1.03	8.12 ± 0.80
SVM	$\begin{pmatrix} 94.18 & 3.45 & 2.38 \\ 5.17 & 88.91 & 5.92 \\ 3.01 & 4.82 & 92.18 \end{pmatrix}$	91.75 ± 1.04	8.25 ± 0.81

Table 2: Confusion matrices for belief k-nearest neighbors, fuzzy k-nearest neighbors, f-SVR and b-SVR classifiers and SVM for 3D artificial data

We obtained classification rates of 91.89±1.03% for the b-SVR classifier, 91.88±1.03% for the f-SVR classifier, 91.41±1.06% for the belief k-nearest neighbors and 90.61±1.10% for the fuzzy k-nearest neighbors. SVM gives an accuracy rate of 91.75±1.04%. We thus obtained better classification rates with the new approaches of f-SVR and b-SVR classifiers compared to the k-nearest neighbors based approaches and SVM. However, these results are not significantly different.

5.3 Real data

The real-world database contains 42 sonar images provided by the GESMA (Groupe d'Etudes Sous-Marines de l'Atlantique) (*cf.* Figure 8). These images were obtained with a Klein 5400 lateral sonar with a resolution of 20 to 30cm in azimuth and 3cm in range. The sea-bottom deep was between 15m and 40m.

Experts have manually segmented these images giving the kind of sediments (rocks, cobbles, sand, silt, ripple (vertical or at 45 degrees)), shadow or other (typically ships) parts on images (see Figure 8 for some examples of tiles). All sediments are given with three certainty levels (sure, moderately sure or not sure), and the boundary between two sediments is also given with a certainty (sure, moderately sure or not sure). Hence, every pixel of every image is labeled as being either a sediment or a shadow, or a boundary with one of the three certainty levels.

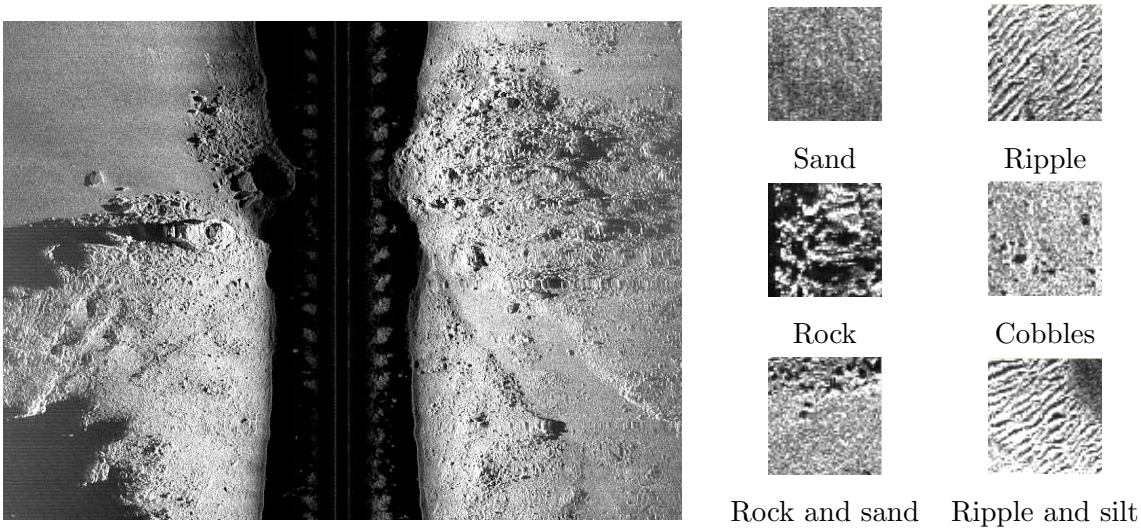

Figure 8: Example of sonar image and tiles with different kinds of sediments.

5.3.1 First database: homogeneous tiles

For a first database we consider only the homogeneous tiles (tiles with one kind of sediment). Indeed, the tiles which contain more than one sediment, must be consider differently, especially for the evaluation [25]. In each tile we have extracted six features (homogeneity, contrast, entropy, correlation, directivity and uniformity) from co-occurrence matrices [28].

We carried out a random drawing of 9,000 homogeneous tiles on the database (31,957 tiles). The training database contains different effectives for the considered classes: 15.53% of tiles contain rock and cobbles, 11.85% of tiles are ripple and the remaining 72.62% are sand or silt. The test database contains 3,000 tiles selected in a random way. We repeated this operation 20 times in order to obtain more reliable classification rates.

We have compared the SVM classifier (using the one-versus-one approach and with different values of C ($C_i, i = 1, 2, 3$, inversely proportional to the effective of each class), for each class to overcome the problem of imbalanced classes in SVM [2]) with a Gaussian kernel $\gamma = 2^{-3.6563}$ and $C = 2^{13.37}$ with the two proposed approaches with $C_f = 27864.849141$, $\gamma_f = 0.123493$, $\varepsilon_f = 0.000008$ for the f-SVR and $C_b = 20188.337768$, $\gamma_b = 0.088573$ and $\varepsilon_b = 0.000083$ for the b-SVR and with the two k-nearest neighbors approaches with the same value of k and k_f than previously used for generated data. Parameters $C, \gamma, C_f, \gamma_f, C_b, \gamma_b, \varepsilon_f$ and ε_b are optimized using genetic algorithms [20]. Here a Gaussian kernel for the regression gives better results for b-SVR and f-SVR classifiers than a linear regression.

The normalized confusion matrices are presented in Table 3. In this case, SVM gives the best results with small confusion between classes. The difference between the four other classifiers is weak. We obtained classification rates of 88.17±1.20% for the fuzzy k-nearest neighbors, 88.26±1.20% for the belief k-nearest neighbors, 87.66±1.23% for the f-SVR and 88.02±1.21% for the b-SVR. These classification rates are not significantly different. The probability of error vectors shows that the approaches based on the k-nearest neighbors give weak confusion for the second class (class of ripple) contrary to the two classifiers based on the SVR which detect less than the half of the effective of this class.

5.3.2 Second database: sonar images

In order to classify a full sonar image, we cut it into several tiles where we will extract texture features. These tiles may contain more than one sediment (inhomogeneous tiles). Unlike the first database, we have to classify these inhomogeneous tiles in one class.

We consider here the full database of 42 sonar images. This database is divided into two parts: a database of 24 images for the learning stage and a second database of 18 sonar images for the test stage. We conduct 10 random trials of these two databases to have consistent and significant results. The classification result is the average over the results of classification of the 10 random trials used. We use only homogeneous tiles (tiles containing one kind of sediment)

	NCM (%)	CR_m (%)	PE_m (%)
fuzzy k-nn	$\begin{pmatrix} 70.16 & 9.45 & 20.39 \\ 11.38 & 57.78 & 30.84 \\ 1.93 & 1.84 & 96.23 \end{pmatrix}$	88.17 ± 1.20	18,96±1,36
belief k-nn	$\begin{pmatrix} 69.35 & 8.67 & 21.98 \\ 10.85 & 55.35 & 33.80 \\ 1.44 & 1.48 & 97.08 \end{pmatrix}$	88.26 ± 1.20	19,56±1,38
f-SVR	$\begin{pmatrix} 65.59 & 6.15 & 28.26 \\ 12.53 & 45.65 & 41.83 \\ 0.91 & 0.45 & 98.64 \end{pmatrix}$	87.66 ± 1.23	22,53±1,46
b-SVR	$\begin{pmatrix} 71.19 & 4.69 & 24.13 \\ 11.90 & 49.84 & 38.26 \\ 1.15 & 0.85 & 98.01 \end{pmatrix}$	88.02 ± 1.21	20,24±1,40
SVM	$\begin{pmatrix} 75.25 & 4.94 & 19.81 \\ 11.46 & 62.75 & 25.79 \\ 0.87 & 0.82 & 98.30 \end{pmatrix}$	$\mathbf{90.57 \pm 1.10}$	15,93±1,27

Table 3: fuzzy and belief classification results for the first database.

of images for the learning stage.

We present in Table 4 the results obtained with the SVM classifier with a Gaussian kernel $\gamma = 2^{-3.6563}$ and $C = 2^{13.37}$ and with the two new approaches with $C_f = 27864.849141$, $\gamma_f = 0.123493$, $\varepsilon_f = 0.000008$ for the f-SVR and $C_b = 20188.337768$, $\gamma_b = 0.088573$ and $\varepsilon_b = 0.000083$ for the b-SVR. On this database, we cannot obtain results with the fuzzy k-nearest neighbors and the belief k-nearest neighbors because of the size of the database. We obtain an out of memory with a recent computer.

	SVM	b-SVR	f-SVR
NCM (%)	$\begin{pmatrix} 74.21 & 5.86 & 19.93 \\ 23.73 & 46.36 & 29.91 \\ 4.24 & 2.22 & 93.54 \end{pmatrix}$	$\begin{pmatrix} 73.72 & 3.24 & 23.03 \\ 31.22 & 33.67 & 35.11 \\ 2.57 & 0.31 & 97.11 \end{pmatrix}$	$\begin{pmatrix} 68.20 & 5.45 & 26.35 \\ 22.66 & 39.15 & 38.19 \\ 2.58 & 0.86 & 96.56 \end{pmatrix}$
CR_m (%)	83.00±0.03	82.74±0.08	$\mathbf{83.27 \pm 0.06}$
PE_m (%)	24.64±0.02	26.78±0.07	26.91±0.05

Table 4: Classification results for sonar images.

The results in Table 4 are presented with a modified confusion matrix to take into account the degree of certainty by the expert as presented in Section 5.1.

We obtained a classification rate of 83.00±0.03% with SVM, 82.74±0.08% for the b-SVR classifier and 83.27±0.06% for the f-SVR classifier. These approaches provide a low detection of second class (class of ripple). This class is highly confused with the two other classes. We see a better classification of the third class (sand and slit class). Both classifiers, b-SVR and f-SVR, give the best rate for this class. The SVM gives a better detection of the first class (rock and cobble class).

These results show that f-SVR gives the best classification accuracy and a worse detection of the second class by the three classifiers that can be explained. Indeed, this class has a low effective compared to the other classes. Thus, both b-SVR and f-SVR classifiers are sensitive to imbalanced data sets. Comparing with the obtained results on homogeneous tiles, we can conclude that both b-SVR and f-SVR classifiers perform better for data sets that are highly uncertain (classification of inhomogeneous tiles).

6 Conclusions

Membership and belief functions are widely used in the state of art in order to model uncertain and imprecise data. We propose in this paper, one approach to predict membership or belief functions from data. The method is based on the support vector regression of membership or belief functions. Therefore, we have modified the existing support vector regression approach in order to integrate the same constraints of the membership and belief functions. We can do either linear or nonlinear regression using or not a kernel. We also propose a solution to our optimization problem using the SMO approach. This approach has a solid theoretical background: the uniqueness of the solution of the optimization problem and the speed of convergence using the decomposition method.

This new regression approach can be used in many applications where membership or belief functions are able to model correctly uncertain and imprecise data. It can also be used to learn probabilities as they have the same properties (a probability is in $[0, 1]$ and the sum is equal to one) as belief or membership functions. Here we applied this new approach to pattern recognition. From the regression on membership functions and on belief functions, we have introduced two classifiers called respectively f-SVR and b-SVR. The classification results of

these classifiers are compared with the classical SVM classifier, a fuzzy k-nearest neighbors and a belief k-nearest neighbors, giving respectively the membership functions and the belief functions used for the regression. In pattern recognition theory, no classifier performs better than another. It depends on the application and thus on the used data. The results of the f-SVR and b-SVR classifiers on generated data and for sonar images classification which is a hard task, show the interest of these regressions. Especially, the classification of whole sonar images demonstrated that f-SVR gives the highest classification accuracy, but both f-SVR and b-SVR classifiers are sensitive to imbalanced classes. One solution let in perspective to correct this, is to use different constant C $(C_i, i = 1, \ldots, N)$ for each class as suggested in [21, 2] for the SVM approach where the optimal separating hyperplane is pushed away from the minority class. Therefore, for f-SVR or b-SVR, each constant $C_i > 0, i = 1, \ldots, N$ determines the trade-off between the flatness of f_i and the amount up to which deviations larger than ε are tolerated. As a result, the use of C_i for each f_i gives a flexibility to the estimation of f. The problem is then the choice of these parameters.

Another perspective to this work is the use of the regression of the membership or belief functions for other applications than pattern recognition.

A Resolution of belief and fuzzy optimization problem

Consider the learning data $x_t \in \mathbb{R}^d$ and the associated outputs $y_t \in \mathbb{R}^N$, $t = 1, \ldots, l$ where l is the learning database size. η, α, β et γ are Lagrange multipliers defined in Equation (38). C is a user defined constant associated to points outside the ε-tube.

The optimization problem defined by (40) and (41) is a quadratic optimization problem that can be written as

$$\begin{cases} \min\limits_{\mathbf{x}} \quad \sum\limits_{n=1}^{N} \left(\frac{1}{2}\mathbf{x}^{nT} Q \mathbf{x}^n + \mathbf{c}^{nT}\mathbf{x}^n\right) \\ \mathbf{x}^{nT}.[u^T \quad -u^T \quad u^T \quad -u^T \quad -u^T]^T = 0, \quad n = 1, \ldots, N \\ 0 \leq \alpha^{(*)^n} \leq C, \quad \beta^{(*)^n} \geq 0 \quad n = 1, \ldots, N \\ \gamma \geq 0 \end{cases} \tag{55}$$

where

$$Q = \begin{bmatrix} \Lambda\Lambda^T & -\Lambda\Lambda^T & \Lambda\Lambda^T & -\Lambda\Lambda^T & -\Lambda\Lambda^T \\ -\Lambda\Lambda^T & \Lambda\Lambda^T & -\Lambda\Lambda^T & \Lambda\Lambda^T & \Lambda\Lambda^T \\ \Lambda\Lambda^T & -\Lambda\Lambda^T & \Lambda\Lambda^T & -\Lambda\Lambda^T & -\Lambda\Lambda^T \\ -\Lambda\Lambda^T & \Lambda\Lambda^T & -\Lambda\Lambda^T & \Lambda\Lambda^T & \Lambda\Lambda^T \\ -\Lambda\Lambda^T & \Lambda\Lambda^T & -\Lambda\Lambda^T & \Lambda\Lambda^T & \Lambda\Lambda^T \end{bmatrix} \tag{56}$$

and

$$\mathbf{x}^n = \begin{bmatrix} \alpha^n \\ \alpha^{*n} \\ \beta^n \\ \beta^{*n} \\ \gamma^n \end{bmatrix}, \quad \mathbf{c}^n = \begin{bmatrix} \varepsilon - Y^n \\ \varepsilon + Y^n \\ \mathbf{0} \\ u \\ u/N \end{bmatrix}, \gamma^n = \gamma, \quad n = 1, \ldots, N \tag{57}$$

$$\Lambda = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \quad t' = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \quad \mathbf{0} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \gamma = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_N \end{bmatrix} \tag{58}$$

$$Y^n = \begin{bmatrix} y_{1,n} \\ y_{2,n} \\ \vdots \\ y_{l,n} \end{bmatrix} \quad \alpha^{(*)n} = \begin{bmatrix} {\alpha^{(*)}}_{1,n} \\ {\alpha^{(*)}}_{2,n} \\ \vdots \\ {\alpha^{(*)}}_{l,n} \end{bmatrix}, \quad \beta^{(*)n} = \begin{bmatrix} {\beta^{(*)}}_{1,n} \\ {\beta^{(*)}}_{2,n} \\ \vdots \\ {\beta^{(*)}}_{l,n} \end{bmatrix}, \quad n = 1, \ldots, N \tag{59}$$

or alternatively

$$\begin{cases} \text{Min} \quad \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ \mathbf{A}\mathbf{x} = 0 \\ 0 \leq \mathbf{x} \leq \mathbf{C} \end{cases} \tag{60}$$

with

$$\mathbf{Q} = \begin{bmatrix} Q & 0 & \ldots & \ldots & 0 \\ 0 & Q & 0 & \ldots & 0 \\ 0 & 0 & Q & 0 & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \ldots & \ldots & 0 & Q \end{bmatrix} \tag{61}$$

and

$$\mathbf{x} = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^N \end{bmatrix}, \quad \mathbf{c} = [c^1, c^2, \ldots, c^N], \quad \mathbf{A} = \begin{bmatrix} E_1 & 0 & \ldots & \ldots & 0 \\ 0 & E_1 & 0 & \ldots & 0 \\ 0 & 0 & E_1 & 0 & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \ldots & \ldots & 0 & E_1 \end{bmatrix} \tag{62}$$

where

$$E_1 = \begin{bmatrix} u \\ -u \\ u \\ -u \\ -u \end{bmatrix},$$

and $\mathbf{C} \in \{C, \infty\}$.

References

[1] S. Abe and T. Inoue. Fuzzy support vector machines for multiclass problems. *European Symposium on Artificial Neural Networks*, pages 113–118, 2002.

[2] R. Akbani, S. Kwek, and N. Japkowicz. Applying support vector machines to imbalanced datasets. In *Proceedings of the 15th European Conference on Machine Learning (ECML)*, pages 214–220, 2004.

[3] A. Appriou. Situation assessment based on spatially ambiguous multisensor measurements. *International Journal of Intelligent Systems*, 16(10):1135–1156, 2001.

[4] A. Aregui and T. Denœux. Fusion of one-class classifier in the belief function framework. In *International Conference on Information Fusion, Québec, Canada*, July 2007.

[5] J.J. Buckley and Y. Hayashi. Fuzzy neural networks: a survey. *Fuzzy Sets and Systems*, 66(1):1–13, 1994.

[6] C.C. Chang and C.J. Lin. LIBSVM: a library for support vector machines. *Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm/*, 2001.

[7] A.P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics*, 83:325–339, 1967.

[8] T. Denœux. A k-nearest neighbor classification rule based on Dempster-Shafer theory. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 25(5):804–813, 1995.

[9] T. Denœux. A neural network classifier based on Dempster-Shafer theory. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 30(2):131–150, 2000.

[10] T. Denœux and M. Skarstein Bjanger. Induction of decision trees from partially classified data using belief function. *Proceedings of SMC'2000, Nashville, USA*, pages 2923–2928, 2000.

[11] D. Dubois, M. Grabisch, H. Prade, and Ph. Smets. Using the transferable belief model and a qualitative possibility theory approach on an illustrative example: the assessment of the value of a candidate. *International Journal of Intelligence Systems*, 16:1245–1272, 2001.

[12] D. Dubois and H. Prade. *Possibility Theory: An Approach to Computerized Processing of Uncertainty*. Plenum Press, New York, 1988.

[13] S. Gunn. Support vector machines for classification and regression. *ISIS Tech. Report, University of Southampton*, 1998.

[14] H. Han-Pang and L. Yi-Hung. Fuzzy support vector machines for pattern recognition and data mining. *International Journal of Fuzzy Systems*, 14(3):25–28, 2002.

[15] D.H. Hong and C. Hwang. Support vector fuzzy regression machines. *Fuzzy Sets and Systems*, 138:271–281, 2003.

[16] T. Inoue and S. Abe. Fuzzy support vector machines for pattern classification. In *IJCNN '01. International Joint Conference on Neural Networks*, volume 2, pages 1449–1454, 2001.

[17] A. Jozwik. A learning scheme for a fuzzy k-nn rule. *Pattern Recognition Letters*, 1:287–289, 1983.

[18] J.M. Keller, M.R. Gray, and J.A. Givens. A fuzzy k-nn neighbor algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, 15:580–585, 1985.

[19] G.J. Klir, U.S. Clair, and B. Yuan. *Fuzzy set theory: foundations and applications*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1997.

[20] H. Laanaya, A. Martin, D. Aboutajdine, and A. Khenchaf. Knowledge discovery on database for seabed characterization. *MCSEAI 2006, Agadir, Morocco*, 7-9 Decembre 2006.

[21] S. Lessmann. Solving imbalanced classification problems with support vector machines. In *Arabnia, H. (Ed.): Proc. of the Int. Conf. on Artificial Intelligence (IC-AI 04), Las Vegas, Nevada, USA*, pages 39–50, 2004.

[22] C.-J. Lin. On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12:1288–1298, 2001.

[23] C.-J. Lin. Asymptotic convergence of an SMO algorithm without any assumptions. *IEEE Transactions on Neural Networks*, 13:248–250, 2002.

[24] A. Martin. Fusion for evaluation of image classification in uncertain environments. *The 9th International Conference on Information Fusion*, Florence, Italy, 10-13 July 2006.

[25] A. Martin, H. Laanaya, and A. Arnold-Bos. Evaluation for uncertain image classification and segmentation. *Pattern Recognition*, 39:1987–1995, 2006.

[26] A. Martin and C. Osswald. Experts fusion and multilayer perceptron based on belief learning for sonar images classification. *ICTTA, Damascus, Syria*, pages 7–11, April 2008.

[27] A. Martin and I. Quidu. Decision support with belief functions theory for seabed characterization. In *International Conference on Information Fusion, Cologne, Germany*, July 2008.

[28] A. Martin, G. Sévellec, and I. Leblond. Characteristics vs decision fusion for sea-bottom characterization. *Colloque Caractérisation in-situ des fonds marins, Brest, France*, 21-22 October 2004.

[29] S. Medasani, J. Kim, and R. Krishnapuram. An overview of membership function generation techniques for pattern recognition. *International Journal of Approximate Reasoning*, 19:391–417, 1998.

[30] A. Miyazaki, K. Kwon, H. Ishibuchi, and H. Tanaka. Fuzzy regression analysis by fuzzy neural networks and its application. In *Conference on Fuzzy Systems, Orlando, USA*, volume 1, pages 52 – 57, June 1994.

[31] E. Osuna, R. Freund, and F. Girosi. Improved training algorithm for support vector machines. *NNSP'97*, pages 24–26, 1997.

[32] F. Pérez-Cruz, G. Camps, E. Soria, J. Pérez, A.R. Figueiras-Vidal, and A. Artés-Rodriguez. Multi-dimensional function approximation and regression estimation. *International Conference on Artificial Neural Networks, Madrid, Espagne*, august 2002.

[33] S. Petit-Renaud and T. Denœux. Nonparametric regression analysis of uncertain and imprecise data using belief functions. *International Journal of Approximate Reasoning*, 35:1–28, 2004.

[34] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. *Microsoft Research Technical Report MSR-TR-98-14*, 1998.

[35] B. Quost, T. Denœux, and M. Masson. Pairwise classifier combination using belief functions. *Pattern Recognition Letters*, 28:644–653, 2007.

[36] M.P. Sanchez-Fernandez, M. de Prado-Cumplido, J. Arenas-Garcia, and F. Pérez-Cruz. SVM multiregression for non-linear channel estimation in multiple-input multiple-output systems. *IEEE Transactions on Signal Processing*, 58(8):2298 – 2307, 2004.

[37] B. Schölkopf, A. J. Smola, and R. Williamson. Shrinking the tube: A new support vector regression algorithm. In *Proceedings of Conference on Advances in Neural Information Processing Systems II*, pages 330–336, July 1999.

[38] G. Shafer. *A mathematical theory of evidence*. Princeton University Press, 1976.

[39] Ph. Smets. Constructing the pignistic probability function in a context of uncertainty. *Uncertainty in Artificial Intelligence*, 5:29–39, 1990.

[40] Ph. Smets. Practical uses of belief functions. In *Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden*, pages 612–621, July 1999.

[41] A.J. Smola and B. Schölkopf. A tutorial on support vector regression. *NeuroCOLT2 Technical Report NC2-TR-1998-030*, 1998.

[42] H. Tanaka, A. Uejima, and K. Asai. Linear regression analysis with fuzzy model. *IEEE Transactions on Systems, Man, and Cybernetics*, 12(6):903–907, 1982.

[43] D. Tsujinishi and S. Abe. Fuzzy least squares support vector machines for multiclass problems. *Neural Networks*, 16:785–792, 2003.

[44] P. Vannoorenberghe and T. Denœux. Handling uncertain labels in multiclass problems using belief decision trees. *IPMU'2002, Annecy, France*, 3:1919–1926, July 2002.

[45] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, N.Y., 1995.

[46] V.N. Vapnik. *Statistical Learning Theory*. John Wesley and Sons, 1998.

[47] B. Wu and N.-F. Tseng. A new approach to fuzzy regression models with application to business cycle analysis. *Fuzzy Sets and Systems*, 130:33–42, 2002.

[48] L.A. Zadeh. Fuzzy sets. *Information Control*, 8:338–353, 1965.

[49] L.A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

[50] H.J. Zimmermann. *Practical applications of fuzzy technologies*. The Handbook of fuzzy sets series, 1999.

[51] L.M. Zouhal and T. Denoeux. An evidence-theoric k-nn rule with parameter optimization. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 28(2):263–271, 1998.