Classification of Type-2 Fuzzy Sets Represented as Sequences of Vertical Slices

Lorenzo Livi, Member, IEEE, Hooman Tahayori, Antonello Rizzi, Member, IEEE, Alireza Sadeghian, Senior Member, IEEE, and Witold Pedrycz, Fellow, IEEE

Abstract—Granulation of information by using type-2 fuzzy sets is receiving more attention nowadays. This is due to the superior capability of type-2 fuzzy sets in handling the data uncertainty. From a theoretical perspective, a set containing type-2 fuzzy sets has no trivial geometric structure; therefore a proper metric cannot be easily defined. As a consequence, common pattern recognition systems, which in one way or another rely on some (geo)metric structure of the input space, are not easily applicable to a space of type-2 fuzzy sets. In this paper, we study the problem of designing a classifier in the input space of type-2 fuzzy sets. Type-2 fuzzy sets are hence interpreted as (granular) patterns forming a given input dataset. By decomposing a type-2 fuzzy set into a sequence of simpler (lower type) fuzzy sets, we explore the possibility of defining and building dissimilarity and kernel based classification systems on input spaces of type-2 fuzzy sets. Such an interpretation provided in terms of sequences allows us to conceive an effective sequence matching strategy, which can be suitably embedded into well-established pattern recognition systems. We support the methodological developments by performing experiments on synthetically generated classification problems for datasets composed of type-2 fuzzy sets, with adjustable and controlled level of difficulty. Results are promising and suggest to further investigate on the possibility of interpreting type-2 fuzzy sets as input patterns of a given data-driven inference system.

Index Terms—Classification in non-(geo)metric spaces; General type-2 fuzzy sets; Type-2 fuzzy set representation; Similarity and dissimilarity measures.

I. INTRODUCTION

Pattern classification is a well-established subdiscipline of pattern recognition [1]–[3]. Designing a classification system concerns with two main aspects: defining a classification model, \(M\), and developing a suitable learning algorithm, \(A\). In the classification setting, the model \(M\) is usually a parametric (analytical) function – also called discriminating function. It could be expressed in terms of logical inference rules. Learning (synthesizing) a classification model from a finite input dataset \(S \subset X\), where \(X\) is the input domain/space, is typically implemented by means of a suitable optimization problem, which is guided by some performance criterion related to \(M\). Notably, a classification model can be trained in the so-called supervised or unsupervised settings (although more recently also hybrid approaches have been proposed). In the supervised setting, the model is synthesized by using a suitable training set, \(S_r \subset S\), containing input-output pairs of the form \((p_i, l_i)\), where \(p_i \in S\) is a pattern instance and \(l_i \in L\) is the corresponding class label. Conventional, \(X = \mathbb{R}^d\) and \(L\) is the finite set containing the classes considered in the specific classification problem.

A crucial aspect of classifiers is the definition of an effective procedure to quantify the closeness between two input patterns. This concept is usually expressed in terms of similarity or dissimilarity measures [1]. A similarity measure, denoted as \(s : X \times X \rightarrow \mathbb{R}\), is a real-valued function of two arguments that quantifies their closeness in the input domain. A dissimilarity measure \(d(\cdot, \cdot)\) has an analogous definition, although it is conceived to quantify the degree to which two patterns differ from each other. When patterns are represented in \(\mathbb{R}^d\), there are many intuitive ways of defining either \(s(\cdot, \cdot)\) and \(d(\cdot, \cdot)\) [1], [2].

The literature of mathematical models, learning algorithms, and (dis)similarity measures for classifiers operating in \(\mathbb{R}^d\) is nowadays well-established; many solutions have been proposed. However, different interesting patterns, such as biochemical compounds and complex events in smart electric grids, find a more appropriate and complete representation in terms of other formalisms. In fact, more recently modeling patterns in terms of labeled graphs and sequences of generic objects has attracted considerable attention [4]–[6]. Notwithstanding, when departing from the \(\mathbb{R}^d\) input data representation, many technical problems arise. Three main approaches have been developed in the literature [3] for tackling such pattern recognition problems: (i) dealing with the problem directly in \(X\) by means of an appropriate (dis)similarity measure, (ii) by casting the problem in the kernel functions context, and (iii) by embedding the input data in a vector space, bringing back the problem to a more conventional setting.

Information granulation is a key aspect in any data analysis context, whose main goal, in short, is the synthesis of appropriate information granules (IGs) from given input data [7]–[9]. IGs convey sound semantics with respect to the raw data from which they are generated [10]–[16]. IGs can be modeled in many different mathematical settings, such as fuzzy sets [17], shadowed sets [18], and rough sets [19], whose
choice depends on the nature of the problem at hand. Aside from the important problem of information granulation, a compelling question related to the usefulness of the IGs arises: “is it possible to perform inductive inference over a space of IGs?” A positive answer to such a question indicates new perspectives with theoretical and practical implications. The first theoretical challenge relates to the geometrical aspects of the domain of IGs that, if it exists, it is certainly non-trivial. From a practical viewpoint, implementing inductive inference in the domain of IGs allows dealing with problems, such as prediction and recognition of new IGs, on the basis of the information provided in the form of a training set of IGs. For instance, in the aforementioned setting of supervised classification, input patterns would be defined as pairs $(p_i, l_i)$, where $p_i$ is now an IG – e.g., a fuzzy set. However, facing conventional pattern recognition and data mining problems over datasets of IGs requires elaboration. The semantic gap between a raw input domain and a domain of IGs, suggests that a problem defined in the original domain would be significantly different from a problem defined in the space of IGs.

In this paper, we explore the possibility of defining dissimilarity and kernel based classification systems for input data modeled as general Type-2 Fuzzy Sets (T2FSs). T2FSs, which have been introduced as an extension to Type-1 Fuzzy Sets (T1FSs), provide a powerful framework for information granulation [20–27]. Here we develop a general methodology for computing the (dis)similarity of T2FSs based on slicing T2FSs. Such a method, given a T2FS, generates a sequence of T1FSs. In this perspective, such T1FSs are considered as extracted features of the original general T2FSs. This interpretation allows us to conceive an effective sequence matching strategy, which is suitably embedded into a discriminative classification system. In particular, we define and discuss two types of discriminative classification systems: (i) a dissimilarity-based classifier operating directly in the input space, and (ii) a kernelized support vector machine (SVM). Experimental results, obtained on two-class classification problems of varying difficulty, demonstrate the effectiveness and the potential of the proposed solutions.

To our knowledge, in the literature there is only one work [28] that is strictly related to our contribution. However, in Ref. [28], the authors deal with kernel methods for patterns defined as T1FSs, which makes our contribution novel since we deal with (granular) patterns defined as T2FSs. In addition, it is worth citing the recent contribution of Gacek and Pedrycz [29], in which the authors explore a perspective of clustering in the domain of granular data – defined as intervals. It goes without saying that, once it is possible to use effective and well-established classifiers in the space of T2FSs, many interesting applications could be conceived in a data-driven setting. A preliminary version of this study appeared in Ref. [30]. Here we extend the contribution by considering also the context of kernel methods for pattern recognition applications.

The paper is organized as follows. Section II provides the background information on T2FSs that is required in this study. Section III introduces the sequence matching strategy that we designed specifically for general T2FSs. Notably, we will focus on the definition of kernel functions for T2FSs. Section IV presents the integration of the matching strategy into the considered classifiers. Experimental results are presented and discussed in Section V while Section VI provides the conclusions pointing at the future directions. Appendix A introduces the theoretical background related to similarity and dissimilarity measures in pattern recognition.

II. REVIEW OF TYPE-2 FUZZY SETS

This section provides some background material on T2FSs. First, we start with some basic definitions [31]. Then, in the following two subsections, we introduce the problem of T2FS membership functions elicitiation (Sec. II-A), and finally we discuss the context of similarity measures for T2FSs (Sec. II-B).

A T1FS, $f_j$, defined on the domain $X$ is characterized by its membership function $f : X \rightarrow [0, 1]$ that assigns a number in $[0, 1]$, known as membership grade, to each element of $X$ [32]. A T2FS is in turn a fuzzy set with membership grades treated as fuzzy sets defined in the $[0, 1]$ domain. A general T2FS, $\tilde{A}$, on the universe of discourse $X$ is defined as:

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) : x \in X\}, \quad (1)$$

$$\mu_{\tilde{A}}(x) = \{(u, f_x(u)) : u \in f_x \subseteq [0, 1], f_x(u) \in [0, 1]\}.$$

We refer to $\mu_{\tilde{A}}(x)$ as the fuzzy membership grade of $x$ in $\tilde{A}$. From (1) it is evident that $\mu_{\tilde{A}}(x)$ is a T1FS defined on $[0, 1]$; it is also referred to as the vertical slice (VS) of $\tilde{A}$ at $x \in X$. Moreover, in (1), $x$ is called primary variable, $f_x$ represents the set of primary membership values of $x$, and $f_x(u)$ is named secondary grade. $X$, as well as $f_x$, $\forall x \in X$, can be continuous or discrete sets, defining respectively continuous or discrete T2FSs.

If $f_x(u) = 1$ holds for every $u \in f_x$, then a T2FS reduces to the so-called Interval T2FS (IT2FS) [33]. In case of IT2FSs, $\mu_{\tilde{A}}(x)$ is referred to as interval (or interval-valued) membership grade. An IT2FS is fully characterized by the Footprint Of Uncertainty (FOU), which reads:

$$\text{FOU}(\tilde{A}) = \bigcup_{x \in X} \{x, [\mu_{\tilde{A}}(x), \overline{\mu_{\tilde{A}}}(x)]\}. \quad (2)$$

The FOU is a bounded region describing the uncertainties associated with the membership grades of an IT2FS. The FOU can be completely characterized by two T1FSs, named Upper Membership Function (UMF) and Lower membership Function (LMF), respectively defined as follows:

$$\text{UMF}(\tilde{A}) = \text{FOU}(\tilde{A}) = \{x, \mu_{\tilde{A}}(x) : x \in X\}, \quad (3)$$

$$\text{LMF}(\tilde{A}) = \overline{\text{FOU}(\tilde{A})} = \{x, \overline{\mu_{\tilde{A}}}(x) : x \in X\}. \quad (4)$$

Note that in Eqs. (2), (3), and (4), $\mu_{\tilde{A}}(x)$ and $\overline{\mu_{\tilde{A}}}(x)$ denote, respectively, the upper and lower endpoints of $f_x$, which in this paper is assumed to be a subinterval of $[0, 1]$. A more detailed discussion on general T2FSs, their operations, representations, and related applications can be found in the following references [17], [34–35].
A. T2FS Membership Function Elicitation

Data-driven generation of T2FSs membership functions typically results in a complex task [14], [21], [44]–[46]. To offer some motivation behind classification problems involving IGs – especially concerning T1FSs and T2FSs – we take into account the following practically relevant situation of information granulation. Let us consider a time series (see Fig. 1, for which we use a temporal window T to perform data granulation [16]. By using the principle of justifiable granularity [46], [47] on each time window, we form T1FSs for the amplitude, $A_i$, and for the change of amplitude, $\partial A_i$, yielding the pairs: $(A_1, \partial A_1), (A_2, \partial A_2), \ldots, (A_N, \partial A_N)$. Let us assume that, with each time window, we also have some associated class to the time series, say $\omega_1, \omega_2, \ldots, \omega_c$. This could be interpreted as having a suitable training set in the form $((A_1, \partial A_1), \omega_1), ((A_2, \partial A_2), \omega_2), \ldots, ((A_N, \partial A_N), \omega_c)$. This is an example of classification problem involving T1FSs as objects (i.e., patterns) to be classified.

![Figure 1. Sample time series. Data granulation is applied in each time window of the time series. The generated T1FSs are associated to class labels $\omega_i, i = 1, 2, \ldots, c$, so that it is possible to conceive a supervised classification problem.](image)

Let us now assume that we have a family of $p$ different time series: for instance, time series recorded in the same season but in $p$ successive years. See Fig. 2 for an illustrative example; we consider the same time window for implementing the granulation. For each time series and each time window, we end up with a group of fuzzy sets. For the first time window we have:

$$\text{T1FSs of the first time window} = \{(A_1[1], \partial A_1[1]), (A_2[1], \partial A_2[1]), \ldots, (A_N[p], \partial A_N[p])\}.$$

For each of the above groups of T1FSs, we apply again the principle of justifiable granularity, obtaining this time T2FSs of the form $(A_1, \partial A_1), (A_2, \partial A_2), \ldots, (A_N, \partial A_N)$.

B. Similarity Measures for General T2FSs

It should be recalled that in the literature a similarity measure for T2FSs [48]–[52] is a (nonnegative) bounded function $s : \mathcal{F}(X) \times \mathcal{F}(X) \to \mathbb{R}^+$, where $\mathcal{F}(X)$ denotes the domain of all general T2FSs defined on $X$. $s(\cdot, \cdot)$ is usually required to satisfy the following four axioms:

1) (symmetry) $s(\widetilde{A}, \widetilde{B}) = s(\widetilde{B}, \widetilde{A})$, $\forall \widetilde{A}, \widetilde{B} \in \mathcal{F}(X)$;
2) (overlapping) $s(\widetilde{A}, \widetilde{B}) > 0 \iff \widetilde{A} \cap \widetilde{B} \neq \emptyset$, $s(\widetilde{A}, \widetilde{B}) = 0$ otherwise;
3) (maximization) $\forall \tilde{E} \in \mathcal{F}(X)$, $s(\tilde{E}, \tilde{E}) = \max_{\widetilde{A}, \widetilde{B} \in \mathcal{F}(X)} s(\widetilde{A}, \widetilde{B})$;
4) (monotonicity) $\forall \widetilde{A}, \widetilde{B}, \widetilde{C} \in \mathcal{F}(X)$, if $\widetilde{A} \subseteq \widetilde{B} \subseteq \widetilde{C}$, then $s(\widetilde{A}, \widetilde{B}) \geq s(\widetilde{A}, \widetilde{C})$ and $s(\widetilde{B}, \widetilde{C}) \geq s(\widetilde{A}, \widetilde{C})$.

Although such four axioms are expressed in terms of similarity, they can be easily converted to fit for the dissimilarity measure case. The first axiom remains the same, the third one is expressed in terms of minimization (if two input elements are equal, their distance is minimum – usually zero), and the fourth axiom changes by replacing $\geq$ with $\leq$. The second axiom would be modified by requiring the dissimilarity measure to assume the maximum distance value when the two input T2FSs are “non-overlapping” (i.e., when their intersection is empty).

The literature on (dis)similarity measures for general T2FSs is relatively limited [48], [50], [52], [53], although more studies have been performed on IT2FSs and T1FSs; see [54], [55] and references therein. In Ref. [49] the authors proposed the following similarity measure for two T2FSs, $\widetilde{A}, \widetilde{B}$, defined on the continuous universe of discourse $X$,

$$s(\widetilde{A}, \widetilde{B}) = \int_{x \in X} \left( \frac{\mu_{\widetilde{A}}(x) \min \{ u \cdot f_s(u), u \cdot g_s(u) \} dx}{\mu_{\widetilde{A}}(x) \max \{ u \cdot f_s(u), u \cdot g_s(u) \} dx} \right) dx,$$

where $f(\cdot)$ and $g(\cdot)$ denote the secondary membership functions of $\widetilde{A}$ and $\widetilde{B}$, respectively. Eq. 5 could be understood as a generalization of the well-known Jaccard’s index for T1FSs [55]. In fact, $J_u$ computes the Jaccard’s index on each VS of the input T2FSs. In Ref. [49] a very similar method with respect to (5) is proposed. Recently, McCulloch et al. [52]
developed a method to derive a similarity measure for general T2FSs by combining suitable similarity measures for IT2FSs. Such IT2FSs are formed considering the $z$-Slices [40] extracted from the original T2FSs.

### III. The Proposed Approach for Matching General T2FSs

In this section, we describe the matching strategy that we use as key component in order to develop a classifier for T2FSs. In Fig. 3 we report a high-level abstraction of a classifier for T2FSs.

![Schematic view of a classifier for T2FSs, which operates by means of the proposed matching algorithm. Training and test datasets are composed of different general T2FSs, i.e., the input patterns. The T2FSs are processed by the decomposition–aggregation scheme (denoted as “T2FS Matching Algorithm” in the figure), which is embedded into a well-established classifier. The classifier receives in input sequences of VSs (i.e., the sequenced T2FSs). This illustration highlights the main difference involved in processing T2FSs with respect to T1FSs. In fact, membership degrees of T1FSs are real numbers, which can be handled more easily.](image)

The proposed matching strategy operates in two stages and it is presented in different configurations, since it is characterized by a general algorithmic scheme. The first stage performs the decomposition of input T2FSs as sequences of lower-order fuzzy sets. The lower-order fuzzy sets are interpreted as features extracted from the input T2FSs. In this paper, we implement only one T2FS decomposition, which in practice generates a sequence of VSs, i.e., T1FSs. The decomposition is followed by the aggregation stage (second stage), which is in charge of combining the (dis)similarity values computed among the T1FSs characterizing the sequences. The aggregation stage yields the overall (dis)similarity value pertaining two general T2FSs.

Representing two given input T2FSs $\tilde{A}, \tilde{B}$ as sequences, $s_{\tilde{A}}$ and $s_{\tilde{B}}$, enables us conceiving a matching algorithm for general T2FSs in terms of matching algorithm for sequences of objects. Moreover, it allows us to employ well-established “core” (dis)similarity measures for T1FSs, which we denote as $d_{\text{core}}(\cdot, \cdot)$. In the following, we express the matching procedure among two sequenced T2FSs in terms of a dissimilarity measure, denoted as $d_{T2}(\cdot, \cdot)$. Moreover, we focus on the analysis of T2FSs defined on the discrete domain $X$; however, T2FSs defined on a continuous domain can be handled as well by means of a suitable quantization of $X$.

Fig. 4 provides a high-level conceptualization of the decomposition–aggregation scheme.

![Schematic descriptions of the decomposition (4(a)) and aggregation (4(b)) stages of the proposed matching algorithm for T2FSs.](image)

#### A. Decomposition of General T2FSs into Sequences of Vertical Slices

Let $\tilde{A}$ be a general T2FS on $X$, $n = |X|$. By definition, for each $x_i \in X$, we have $\mu_{\tilde{A}}(x_i) = f_i$, where $f_i$ is a T1FS (a VS) characterizing the second degree of uncertainty for $x_i$. By establishing a total ordering on $X$, we can interpret a general T2FS $\tilde{A}$ as a sequence $s_{\tilde{A}} = (f_1, f_2, \ldots, f_n)$ of $n$ T1FSs, each describing a specific VS, $f_i$, of $\tilde{A}$ at a specific $x_i \in X$. Therefore, here we interpret each VS as a feature of the original T2FS. In the remainder of this paper, the $i$-th VS of $\tilde{A}$ is denoted also as $s_{\tilde{A}}[i]$.  

---

1063-6706 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
B. Aggregation Schemes of Dissimilarity Values

The second stage of the proposed matching strategy consists in aggregating the dissimilarity values computed between the features characterizing two sequences $s_{\mathcal{A}}, s_{\mathcal{B}}$. Since such sequences are composed of VFs, in this paper we employ dissimilarity measures for T1FSs. In the following, we denote the core dissimilarity measure as $d_{\text{core}}(\cdot, \cdot)$, avoiding any specification if not strictly required; in the experiment section we will be more specific. We propose two methods of aggregation: the first one is linear in the sequence length, while the second one is characterized by a quadratic number of operations. However, the second approach can handle variable-length sequences, which can become interesting when different classes of patterns are characterized by different features–VSs.

1) Linear Weighted Aggregation: Given two sequenced T2FSs, $s_{\mathcal{A}}$ and $s_{\mathcal{B}}$, the weighted average of the dissimilarity values between the elements of the sequences is given by,

$$d_{T2}(s_{\mathcal{A}}, s_{\mathcal{B}}) = \frac{1}{n} \sum_{i=1}^{n} w_i \times d_{\text{core}}(s_{\mathcal{A}}[i], s_{\mathcal{B}}[i]),$$

where $w_i \in [0, 1]$, with $\sum_{i=1}^{n} w_i = 1$. The weights $w_i$ introduced here could be useful when the input domain $X$ is characterized by elements with different relevance to the problem at hand; in general, we can set $w_i = 1/n$.

2) Quadratic Aggregation: Alignment based techniques provide a well-known approach for defining a dissimilarity measure for variable-length sequences. Such an approach consists in computing the minimum amount (and related magnitude) of operations needed to transform a sequence into the other. This process can be either global or local. In the first case, the alignment is performed considering the sequences as a whole, while in the latter case the algorithm focuses on subsequences of maximum similarity, resulting in a faster but less accurate solution. The Dynamic Time Warping (DTW) [56]–[58] global alignment scheme is able to optimally align general sequences of complex objects, defined over the space $\mathcal{A}$. The key component of DTW is the definition of a core dissimilarity measure, $d_{\mathcal{A}}: \mathcal{A} \times \mathcal{A} \rightarrow \mathbb{R}^+$, which quantifies the cost related to object substitutions. The underlying algorithmic scheme of DTW is founded on the dynamic programming framework; the computation of the minimum cost of alignment is therefore achieved by evaluating $d_{\mathcal{A}}(\cdot, \cdot)$ a quadratic number of times.

This second aggregation method is directly based on the DTW algorithm. Notably, the dissimilarity measure is calculated by feeding DTW with the two sequenced T2FSs, i.e., $s_{\mathcal{A}}$ and $s_{\mathcal{B}}$:

$$d_{T2}(s_{\mathcal{A}}, s_{\mathcal{B}}) = \text{DTW}(s_{\mathcal{A}}, s_{\mathcal{B}}).$$

As mentioned before, the DTW algorithm relies on the internal dissimilarity measure, $d_{\mathcal{A}}(\cdot, \cdot)$, which in the case of our problem we set as $d_{\text{core}}(\cdot, \cdot)$. Algorithm 1 delivers the pseudo-code of the DTW algorithm.

The quadratic aggregation, theoretically, enables us to compare sequences of different lengths, i.e., using a different number of features for the input T2FS sequence representations. Although this perspective can be interesting for different reasons – for instance, the important information for discriminating two T2FSs may reside only in specific regions of $X$ – in this paper we will restrict the application of the DTW algorithm to sequences of the same length.

3) Properties Derived from the Aggregation Schemes: Both linear weighted and quadratic aggregations make use of sum and multiplication operations by a positive constant. In other words, the outcomes produced by the core dissimilarity $d_{\text{core}}(\cdot, \cdot)$ are aggregated by sum and multiplication operations only. Therefore, the overall behavior of $d_{T2}(\cdot, \cdot)$ is in turn completely determined by the properties of the specific $d_{\text{core}}(\cdot, \cdot)$.

Let us consider the properties with respect to the four axioms of fuzzy similarity measures listed in Sec. I-B.

If $d_{\text{core}}(\cdot, \cdot)$ is symmetric, then $d_{T2}(\cdot, \cdot)$ will be symmetric as well. This fact holds also in the DTW case, since the DTW algorithm is a symmetric algorithm of its arguments. If $d_{\text{core}}(\cdot, \cdot)$ is normal, i.e., it yields values within the $[0, 1]$ interval, the outcomes of Eq. [6] will also fall within the $[0, 1]$ interval. Conversely, to obtain the corresponding normal dissimilarity measure for T2FSs when using the DTW, we need to use the $1/\max(n, m)$ normalization factor in [7]. Since $d_{\text{core}}(\cdot, \cdot)$ is normal, each cost computed in the DTW cost matrix is bounded in $[0, 1]$. As a consequence, the maximum alignment cost for two sequences of length $n$ and $m$ is given by $\max(n, m)$. If $d_{\text{core}}(\cdot, \cdot)$ satisfies the minimization requirement (i.e., it yields zero only when the input arguments are equal), then also $d_{T2}(\cdot, \cdot)$ – considering the formulations given in Eqs. [6] and [7] – will satisfy the requirement as well; due to the accumulation of a null cost in the related expressions. If $d_{\text{core}}(\cdot, \cdot)$ is compliant with the non-overlapping requirement, then, given two input, non-overlapping T2FSs, our algorithm will always accumulate the maximum possible dissimilarity value. Finally, the monotonicity is obviously preserved as a consequence of the monotonicity involved in the aggregation (i.e., the nonnegative dissimilarity values are always added, and never subtracted).

It is worth mentioning that the proposed decomposition–aggregation scheme can be conceived even when $d_{\text{core}}(\cdot, \cdot)$ is not a fuzzy (dis)similarity measure, i.e., when some of the aforementioned axioms are not properly satisfied. In fact, for the wider purpose of designing a classification system based on a matching algorithm among T2FSs, the choice of $d_{\text{core}}(\cdot, \cdot)$ is driven by specific context-dependent factors and by the mathematical requirements imposed, as in our case, by the

---

### Algorithm 1 DTW algorithm for sequenced T2FSs

**Input:** $s_{\mathcal{A}}, s_{\mathcal{B}}$ both with $n$ VFSs

**Output:** The minimum cost of alignment for $s_{\mathcal{A}}$ and $s_{\mathcal{B}}$

1: Allocate a $n \times n$ distance matrix, $D$
2: for $i = 1, 2, \ldots, n$ do
3: \hspace{1em} $D(i,0) = D(0,i) = \infty$
4: end for
5: Set $D(0,0) = 0$
6: for $i = 1, 2, \ldots, n$ do
7: \hspace{1em} for $j = 1, 2, \ldots, n$ do
8: \hspace{2em} $\delta = d_{\text{core}}(s_{\mathcal{A}}[i], s_{\mathcal{B}}[j])$
9: \hspace{2em} $D(i,j) = \delta + \min(D(i-1,j), D(i,j-1), D(i-1, j-1))$
10: \hspace{1em} end for
11: end for
12: return The minimum cost of alignment, $D(n,n)$
model of the adopted classification system.

C. On Defining Positive Definite Kernel Functions for T2FSs

A considerable number of positive definite (PD) kernel functions and related learning systems have been developed over Euclidean input spaces [2]. Such functions can also be defined in spaces without a straightforward Euclidean geometry, such as sets of graphs and sequences [3], [4], [59]. It is important to notice that PD kernel functions can be obtained by means of the so-called convolution property [4]. Formally speaking, kernel functions are closed under pointwise addition, product, and multiplication by positive constants. Let \( k_1(\cdot, \cdot) \) and \( k_2(\cdot, \cdot) \) be two PD kernels, then \( k_1(\cdot, \cdot) \times k_2(\cdot, \cdot) \) and \( a \times k_1(\cdot, \cdot), a \in \mathbb{R}^+ \), are PD kernel functions as well.

We make use of the aforementioned properties of kernel functions to conceive a (convoluted) PD kernel function for sequenced T2FSs. The PD kernel function for T2FSs is meant to be used within the algorithmic scheme introduced in the previous section (i.e., decomposition and aggregation). This is achieved by defining \( d_{\text{core}}(\cdot, \cdot) \) as a suitable PD kernel function. It should be noted that PD kernel functions are by definition similarity measures. However, as specified in Appendix A, similarity and dissimilarity functions are related concepts; one can be obtained from the other with appropriate elaborations.

In the following, we report a number of well-known kernel functions that are suitable for our purposes. We distinguish between kernel functions that rely on a distance function with those that are based on the inner product operator.

1) Distance-based PD Kernels for T1FSs: The most famous PD kernel is probably the Radial Basis Function (RBF), whose general definition reads:

\[
k(x, y) = \exp(-\gamma \times d(x, y)^2).
\]

If \( \gamma = 1/(2\sigma^2) \), \( \sigma > 0 \), then (8) is called Gaussian RBF kernel. The parameter \( \sigma \), called kernel size or width, is of utmost importance since it permits to adapt the kernel to the data distribution at hand. By generalizing the distance function \( d_{\text{core}}(\cdot, \cdot) \) in [6], it is possible to adapt the RBF kernel to various input domains.

Livi and Rizzi [60] (see also [44], [45]) have proposed the Fuzzy Information-Theoretic Kernel (FITK) function, which elaborates over the Gaussian RBF scheme (GA). The FITK was originally proposed by using a fuzzy divergence measure for T1FSs. The FITK underlying scheme can be generalized to other distance-based PD kernel functions, such as the rational quadratic (RQ) kernel [44]. The RQ is known to be less sensitive to the kernel size \( \sigma \) and to the variations of dissimilarities among input patterns.

By interpreting a (finite) T1FS \( f \) as a vector of \( n \) membership values, denoted as \( f \), and accordingly assuming an underlying Euclidean metric structure on the set of such vectors, we can implement the distance among two T1FSs also by means of the Euclidean distance:

\[
d_2(f, h) = \sqrt{\sum_{i=1}^{n} (f_i - h_i)^2}.
\]

The Euclidean distance can be used in the following two PD kernel functions,

\[
k(f_i, f_j) = \exp\left(-\frac{1}{2\sigma^2} \times d_2(f_i, f_j)^2\right), \quad \sigma > 0,
\]

\[
k(f_i, f_j) = 1 - \frac{d_2(f_i, f_j)^2}{d_2(f_i, f_j)^2 + \sigma}, \quad \sigma > 0,
\]

which we denote in this paper as GA-EUC and RQ-EUC, respectively.

2) Inner Product based PD Kernels for T1FSs: PD kernel functions can be constructed also by exploiting the inner product operator [17]. A finite T1FS \( f \) can be interpreted as a vector of membership values. A measure of similarity among two T1FSs can therefore be computed by exploiting the inner product operator. In particular, the well-known cosine kernel function reads

\[
k(x, z) = \frac{x \cdot z}{\|x\| \times \|z\|},
\]

where \( \|x\| \) is the Euclidean norm of \( x \). When the input vectors contain nonnegative real values, the cosine kernel (COS) yields similarity values in \([0, 1]\) only; this is the case of vectors of membership values. In the literature there are other inner product based PD kernel functions [2] (for instance, the polynomial kernel); however in this paper we use the cosine kernel only.

IV. LEARNING AND TESTING A CLASSIFICATION MODEL IN THE DOMAIN OF T2FSs

In this section, we introduce the two classifiers that we use for the experiments. First we introduce a classifier based on \( k\)-NN rule, which operates directly in the input space. Then, we discuss how to use the well-known SVM classifier in a domain of T2FSs.

A. Dissimilarity-based Classification System Operating Directly in the Input Space

A universal way of forming a classification rule over various input spaces is by means of the \( k \) nearest neighborhood (NN) rule. Such a classification rule relies completely on the possibility of defining a meaningful dissimilarity measure in the input space. When facing classification problems defined in the domain of T2FSs, a meaningful dissimilarity measure for T2FSs must be conceived and plugged into the \( k\)-NN rule based classifier – perspective explored in Refs. [30], [61]. Note that, technically, the \( k\)-NN does not form an explicit classification model and no learning phase is performed to compress the useful information in the training set. However, it is usually characterized by a good generalization capability; for this reason it is often taken into consideration as a reference system and/or to benchmark novel dissimilarity measures.

B. Kernelized Support Vector Machines for T2FSs

SVM is a commonly used classification system adopted in pattern recognition [2]. It is founded on the definition of a hyperplane in the input data space, by following the maximum-margin criterion. The underlying optimization problem is
defined as a convex optimization problem, which is addressed by the well-known Lagrangian technique. While hyperplane-based classifiers are only capable of linear classification, the SVM framework allows to include a suitable (nonlinear) PD kernel function, enabling thus to define a nonlinear model. Detailed descriptions on the SVM model and related synthesis can be found in Ref. [2]. In this paper, we consider the kernelized SVM version, known as (kernelized) soft margin SVM (C-SVM). The dual optimization problem faced by C-SVM reads:

\[
\begin{aligned}
\max_{\alpha} & \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j K_{ij} \\
\text{s.t.} & \quad 0 \leq \alpha_i \leq C, \forall i \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0.
\end{aligned}
\]

where \( K \) is the Gram matrix containing the inner products, evaluated through the kernel function, \( K_{ij} = k(x_i, x_j); C > 0 \) is a hyper-parameter that controls the trade-off between the misclassified training data and the size of the margin, \( y_i \in \{-1, 1\} \) is a class label and \( \alpha \) are Lagrangian coefficients, which form the solution of the convex optimization problem. A new test pattern \( x' \) is classified according to the following expression,

\[
y' = \text{sgn}(w \cdot \phi(x') + b) = \text{sgn} \left( \sum_{i=1}^{n} \alpha_i y_i k(x_i, x') + b \right),
\]

where \( \text{sgn}(\cdot) \) is the sign function, while the hyperplane is identified by \( w = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) \) and \( b \). It is clear from the discussed equations that both training and test stages of C-SVM can be completely adapted to input spaces where it is possible to define a PD kernel function (see for example [59], [60]).

In this paper, we test the feasibility and the effectiveness of using C-SVM in the space of T2FSs. We exploited the proposed matching algorithm configured with PD kernel functions introduced in Sec. [III-C].

V. EXPERIMENTS

The conducted experiments aim to help us understand and evaluate the performance of the two classification systems presented in the previous section, mostly in terms of test set classification accuracy. For each classification system we consider several configurations, which are differentiated by the settings of the proposed T2FS matching algorithm. Therefore, Sec. [V-A] introduces the settings adopted for the experiments. In Sec. [V-B] we describe the data generation method used to synthesize the batch of classification problem instances. Finally, Sec. [V-C] presents and discusses the results.

A. Configurations of the Classification Systems

The k-NN (Sec. [V-A]) is the most general classifier, since it completely relies on a dissimilarity measure, which is not constrained to adhere to specific requirements – it is used as a proximity measure to generate the neighborhoods. C-SVM, on the other hand, strictly requires a PD kernel function for T2FSs. In addition to the PD kernels introduced in Sec. [III-C] we also consider the Jaccard’s index [54], [55],

\[
d(f_i, f_j) = 1 - \frac{\Sigma(f_i^T f_j)}{\Sigma(f_i^\top f_j)},
\]

where \( \top \) is a t-norm, \( \perp \) is a t-conorm, and \( \Sigma(f_i) \) calculates the cardinality of a finite T1FS, \( f_i \). Note that (15) is taken into account for the k-NN only. In order to make a comparison between k-NN and C-SVM, we consider in the former the same PD kernel functions although expressed as dissimilarity measures. Table [I] summarizes all system configurations adopted for the experiments.

B. Generation of Synthetic Datasets with Varying Difficulty

A pattern is a pair \((A_i, l_i)\), where \( A \) is a T2FS defined on \( X \) and \( l_i \in \mathcal{L} \) denotes its class label. Here we focus on two-class classification problems (i.e., \( \mathcal{L} = \{l_1, l_2\} \)). There are two, disjoint training \( S_t \) and testing \( S_e \) datasets. Let \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \) be the generating T2FSs that describe the two prototypical elements of the classes. We generate datasets for each problem instance by replicating those two T2FSs. Notably, we apply a white Gaussian noise based scheme in order to alter their models, obtaining thus different distorted versions of the original patterns. The level of noise directly controls the difficulty of the classification problem: a higher level of noise implies higher overlap of the distorted patterns and thus a more complex classification problem. In the following, we present the generation method for the distorted patterns focusing on one class only. Fig. 5 shows the two generating T2FSs used in the experiments.

Figure 5. The two generating T2FSs used to create the two classes of patterns characterizing the classification problems. T2FSs are defined on the numeric input domain \( X = [0, 200] \). Related VSs are modeled as symmetric, triangular T1FSs.

Let \( \mathcal{A} \) be the generating T2FS for a given class of patterns. In this paper, we have interpreted a finite T2FS \( \mathcal{A} \) as a sequence \( s_{\mathcal{A}} \) of VSs. Accordingly, to create the \( j \)-th distorted pattern instance, denoted as \( \mathcal{A}(j) \), we apply white noise to the membership values describing each VS, i.e., \( s_{\mathcal{A}}(i) \).

Let \( s_{\mathcal{A}}(i)[u], u = 1, \ldots, m \), be the \( u \)-th membership value of the \( i \)-th VS, \( f_i \). Let \( g_\eta \in [0, 1] \) be a value extracted from a Gaussian distribution with zero-mean and standard deviation equal to \( \eta > 0 \); note that \( g_\eta \) is applied to the entire VS domain, i.e., \([0, 1] \). The distorted sequence \( s_{\mathcal{A}(j)} \) is therefore obtained as:

\[
s_{\mathcal{A}(j)}[u] = s_{\mathcal{A}}[u] + g_\eta, \quad \forall u = 1, \ldots, m, \quad \forall i = 1, \ldots, n.
\]


Figure 6. Different distortion levels applied to a VS of a generating T2FS shown in Fig. 5. All figures show how the original VS (red lines) is modified by the noise (green lines). Fig. 6(a), Fig. 6(b), and Fig. 6(c) show high, medium, and low distortion levels, respectively.

The value of $\eta$ directly controls the distortion level and hence the “recognizability” of the distorted pattern instances. Using high values for $\eta$ produces difficult classification problem instances; conversely, by using small values we obtain easy problems. Here we generate 15 distinct classification problem instances of decreasing difficulty. In each problem, training $S_t$ and test $S_s$ sets are composed of 400 patterns, equally distributed among the two classes. The input domain, $X$, of T2FSs is sampled with 200 VSs ($n = 200$), while the primary and secondary memberships in [0, 1] are sampled with a resolution of 100 points (i.e., $m = 100$). Fig. 6 shows how a sample VS is modified by the discussed noise scheme.

C. Classification Results and Discussion

Fig. 7 presents the results for the $k$-NN rule, with $k = 1, 3, 5$, and the C-SVM based system configurations. In all experiments, the $C$ parameter of C-SVM in Eq. (13) is set to 2; this choice has been made on the basis of preliminary tests. Test set accuracy results are in general compliant with the expectations, i.e., we obtain reasonable results considering the problem difficulty. However, we note in general very good test set classification accuracy results when considering the C-SVM based configurations. It is possible to observe that the system configurations that operate by means of GA-EUC and RQ-EUC kernel functions perform considerably well, especially when used with the DTW-based aggregation scheme in C-SVM (Fig. 7(d)). In Fig. 7(c) we show the best results achieved with either the $k$-NN rule and C-SVM. Note that those two system configurations are consistent for either $k$-NN and C-SVM, i.e., they consider both a Gaussian kernel and the PD kernel is used as a dissimilarity measure. A considerable performance gap can be observed, which stresses the superior performance of C-SVM in terms of generalization capability on the experiments under consideration.

The average number of support vectors used in the first seven problems amounts to roughly 280. The number of support vectors drops below 100 only starting from the tenth problem instance. However, we note that a more parsimonious model could be obtained in the C-SVM case by suitably adapting the $C$ parameter.

We implemented the software in standard C++ on a Linux 64-bit operating system. All software components are executed serially on a machine equipped with an Intel(R) Core(TM) i7-3930K CPU. As suggested by the different asymptotic computational complexities, we note a substantial difference, in terms of computing time, between configurations operating with linear-time aggregation scheme (Sec. III-B1) and those using the DTW-based aggregation (Sec. III-B2). Table II summarizes the computing time performance results for the linear-time aggregation.

<table>
<thead>
<tr>
<th>System</th>
<th>Average time</th>
<th>Test time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$-NN ($k = 1$)</td>
<td>-</td>
<td>4.29</td>
</tr>
<tr>
<td>C-SVM</td>
<td>4.40</td>
<td>4.34</td>
</tr>
</tbody>
</table>

Configurations operating with the DTW-based aggregation are more than 100 times slower with respect to those operating...
with the linear-time aggregation. Such a considerable difference might be relevant when tackling particular applications that require effectiveness in computing time. However, this gap could be justified by stressing that, by means of DTW-based aggregation scheme, it would be possible to process sequences of variable length. We did not explore such a perspective in this paper, although we suggest that it could be useful for two main reasons. First, reducing the number of features (i.e., the number of extracted VSs) would imply a reduction of the main memory load, which might become
useful in resource-constrained applications. Second, the use of different samplings for the input T2FSs might allow to tailor the approach for the particular application at hand. In this perspective, it could be interesting to use feature selection techniques on the sequenced T2FSs.

VI. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we have studied the problem of defining a discriminative classification system in the space of general T2FSs. The approach that we have proposed is based on the decomposition of a T2FS as a sequence of lower order T2FSs. The sequenced T2FSs are compared by means of discriminative classification system in the space of general T2FS representations, different IG models (e.g., intervals–time. It is worth pointing out that, in principle, the methodology proposed in this paper could be adapted to include other T2FS representations, different IG models (e.g., intervals–sets, rough sets, and shadowed sets), and several classification systems. The only requirement is the possibility of defining meaningful and appropriate dissimilarity and kernel functions in the input space. Finally, although here we have focused on classification problems, it could be interesting to deal also with function approximation problems defined in the T2FSs domain.

APPENDIX A
DISSIMILARITY AND SIMILARITY MEASURES IN PATTERN RECOGNITION

The pattern recognition discipline has been traditionally conceived for data represented in a suitable vector space, which is normally (a subset of) \( \mathbb{R}^d \); this representation is commonly termed as feature-based representation. The feature-based representation automatically provides a way to define the concept of position of a pattern in the feature space. The similarity (degree of nearness) of two patterns \( \mathbf{x}, \mathbf{z} \in \mathbb{R}^d \) can be favorably elaborated by starting from the inner product mathematical construct,

\[
\langle \mathbf{x}, \mathbf{z} \rangle = \mathbf{x} \cdot \mathbf{z} = \sum_{i=1}^{d} x_i \times z_i.
\] (17)

The concept of distance (degree of separation) is the dual of the similarity concept. In fact, in an inner product space the (Euclidean) distance between patterns can be obtained directly from their similarity:

\[
d_2(\mathbf{x}, \mathbf{z}) = \sqrt{\langle \mathbf{x} - \mathbf{z}, \mathbf{x} - \mathbf{z} \rangle} = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle + \langle \mathbf{z}, \mathbf{z} \rangle - 2 \langle \mathbf{x}, \mathbf{z} \rangle}.
\] (18)

The converse is also true, since in fact position, similarity, and distance are intimately related concepts. Let \( \mathbf{X} \in \mathbb{R}^{n \times d} \) be the (centered) data matrix (i.e., each row of this matrix contains a feature vector), and let \( \mathbf{K} = \mathbf{X} \mathbf{X}^T \in \mathbb{R}^{n \times n} \) be the similarity matrix – also called Gram matrix – containing the pairwise inner products,

\[
K_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \quad 1 \leq i, j \leq n.
\] (19)

The configuration in \( \mathbb{R}^d \), i.e., the vector-based representation of the data, can be retrieved from the similarity matrix \( \mathbf{K} \) as well, through a linear embedding. The methods exploits the eigendecomposition of \( \mathbf{K} \):

\[
\mathbf{K} = \mathbf{U} \Lambda \mathbf{U}^T; \quad \mathbf{X} = \mathbf{U} \Lambda^{1/2}.
\] (20) (21)

In (20), \( \mathbf{U} \) is a square orthogonal matrix containing the eigenvectors (arranged as column vectors) of \( \mathbf{K} \), and \( \Lambda^{1/2} \) is a diagonal matrix containing the square roots of corresponding non-zero eigenvalues. Nonetheless, according to Eqs. (18) and (19) the distance matrix \( \mathbf{D} \) can be obtained from the similarity matrix \( \mathbf{K} \) as well:

\[
D_{ij} = \sqrt{K_{ii} + K_{jj} - 2K_{ij}}, \quad 1 \leq i, j \leq n.
\] (22)

The converse is also possible via the following expression:

\[
\mathbf{K} = \frac{1}{2} \mathbf{D}^2 \mathbf{C}.
\] (23)

Note that \( \mathbf{D}^2 \) is a matrix containing the squared distances, and \( \mathbf{C} \), called centering matrix, is defined as \( \mathbf{C} = \mathbf{I} - \mathbf{J}/n \), where \( \mathbf{I} \) is the identity matrix, \( \mathbf{J} \) is the all-one matrix, and finally \( n \) is the size of input data.

The algebraic relations herein introduced can be generalized to virtually any input space, \( \mathcal{X} \), by means of the so-called kernel embedding provided by positive definite (PD) kernel functions. A PD kernel function \( k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \) is a symmetric function that satisfies

\[
\sum_{i,j \in \{1, \ldots, n\}} c_i c_j k(x_i, x_j) \geq 0,
\] (24)

\[\forall n \in \mathbb{N}, x_1, \ldots, x_n \in \mathcal{X} \text{ and } c_1, \ldots, c_n \in \mathbb{R} \]. PD kernel functions are Mercer’s kernels and therefore their evaluation in the input space \( \mathcal{X} \) corresponds to the evaluation of an inner product in an implicit and usually unknown high-dimensional Hilbert space, \( \mathcal{H} \). This property is referred to as the kernel trick [2], and can be formally summarized with the following well-known expression:

\[
k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}},
\] (25)

where \( \phi : \mathcal{X} \rightarrow \mathcal{H} \) is an implicit map. Examples of popular PD kernels are the RBF, exponential, polynomial, and linear functions. Kernel functions are similarity functions of particular importance in pattern recognition and data mining, since

1063-6706 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
they allow to include non-linear and adaptable components in learning machines, such as the well-known (kernelized) SVM [2].

In many practical applications, however, the information about the input data is available in terms of a matrix, $D$, containing the pairwise dissimilarity values between the data; in the literature this is called dissimilarity matrix. Accordingly, the function $d(\cdot, \cdot)$ that is used to construct such a matrix is called dissimilarity measure [1], [3], [5]. Although in the general case $d(\cdot, \cdot)$ might violate some of the metric requirements, the information contained in $D$ can be of great help in the design of pattern recognition systems for “unconventional” input spaces. In fact, $D$ can be used to construct a dissimilarity space embedding or it can be rectified in order to induce a PD (kernel) matrix and therefore also an Euclidean embedding of the input data.

Lorenzo Livi (M’14) received the B.Sc. and M.Sc. degrees from the Department of Computer Science, SAPIENZA University of Rome, Italy, in 2007 and 2010, respectively. In 2014, he received the Ph.D. degree from the Department of Information Engineering, Electronics, and Telecommunications at the same University. During his studies, he has also worked in the ICT industry. He is currently a Post Doctoral Fellow with the Department of Computer Science, Ryerson University, Toronto, Canada. His main research interests include computational intelligence methods, time series analysis, and complex dynamical systems, with applications in biochemistry and biophysics. Dr. Livi is currently serving as chair of the IEEE signals and computational intelligence chapter of Toronto.

Hooman Tahayori received the Ph.D. degree in informatics from the University Degli Studi di Milano, Milano, Italy, in 2009. He received the Master’s (Hons.) degree in robotics and artificial intelligence and the Ph.D. degree in computer science-hardware design from Shiraz University, Shiraz, Iran, in 2000 and 1997, respectively. His main areas of research interests include fuzzy sets of higher order, granular computing, and their applications in various disciplines. He has published more than 45 technical papers in the areas of type-2 fuzzy sets, shadowed sets, and granular computing.

Antonello Rizzi received the Ph.D. in Information and Communication Engineering in 2000, from the University of Rome “La Sapienza”. In September 2000 he joined the “Information and Communication” Department (INFO-COM Dpt., “La Sapienza”) as an Assistant Professor. Since July 2010 he joined the “Information Engineering, Electronics and Telecommunications” Dpt., in the same University. His major research interests are in the area of Soft Computing, Pattern Recognition and Computational Intelligence, including supervised and unsupervised data driven modeling techniques, neural networks, fuzzy systems and evolutionary algorithms. His research activity concerns the design of automatic modeling systems, focusing on classification, clustering, function approximation and prediction problems. Currently, he is working on different research topics and projects, such as smart grids and micro-grids modelling and control, intelligent systems for sustainable mobility, energy storage systems modeling and control, predictive diagnostic systems for condition based maintenance in smart grids, Granular Computing, Data Mining systems in industrial applications, classification and clustering systems for structured patterns, graph and sequence matching, agent-based clustering. Since 2008, he serves as the scientific coordinator and technical director of the R&D activities in the Intelligent Systems Laboratory within the Research and Technology Transfer Center for Sustainable Mobility of Lazio Region. Dr. Rizzi (co-)authored more than 90 journal/conference articles and book chapters. He is a member of IEEE.

Alireza Sadeghian (S’94–M’00–SM’06) received the B.A.Sc. (Hons.) degree in electrical and computer engineering from Tehran Polytechnic University, Tehran, Iran, and the M.A.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada. Dr. Sadeghian holds the position of Professor with the Department of Computer Science, Ryerson University, Toronto, Canada. He has published over 120 technical papers and his research interests include computational intelligence, neural networks, fuzzy sets of higher order, and data-driven modeling. Dr. Sadeghian is an Associate Editor of IEEE Access, Expert Systems Journal, and serves on the Editorial Board of Applied Soft Computing. He is also actively involved with a number of professional and academic boards, including NAIPS board (North American Fuzzy Information Processing Society), and IEEE executive committee, Toronto Section.
Witold Pedrycz is Professor and Canada Research Chair (CRC) in Computational Intelligence in the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada. He is also with the Systems Research Institute of the Polish Academy of Sciences, Warsaw, Poland. He holds an appointment of special professorship in the School of Computer Science, University of Nottingham, UK. In 2009 Dr. Pedrycz was elected a foreign member of the Polish Academy of Sciences. In 2012 he was elected a Fellow of the Royal Society of Canada. Witold Pedrycz has been a member of numerous program committees of IEEE conferences in the area of fuzzy sets and neurocomputing. In 2007 he received a prestigious Norbert Wiener award from the IEEE Systems, Man, and Cybernetics Council. He is a recipient of the IEEE Canada Computer Engineering Medal 2008. In 2009 he has received a C大街 Prize for Soft Computing from the European Centre for Soft Computing for “pioneering and multifaceted contributions to Granular Computing”. In 2013 has was awarded a Killam Prize and received a Fuzzy Pioneer Award 2013 from the IEEE Computational Intelligence Society.

His main research directions involve Computational Intelligence, fuzzy modeling and Granular Computing, knowledge discovery and data mining, fuzzy control, pattern recognition, knowledge-based neural networks, relational computing, and Software Engineering. He has published numerous papers in this area. He is also an author of 15 research monographs covering various aspects of Computational Intelligence, data mining, and Software Engineering.

Dr. Pedrycz is intensively involved in editorial activities. He is an Editor-in-Chief of Information Sciences and Editor-in-Chief of WIREs Data Mining and Knowledge Discovery (Wiley). He currently serves as an Associate Editor of IEEE Transactions on Fuzzy Systems and is a member of a number of editorial boards of other international journals.

REFERENCES


