A classification algorithm based on multi-dimensional fuzzy transforms

Ferdinando Di Martino¹ · Salvatore Sessa¹

Received: 9 December 2019 / Accepted: 7 June 2021 / Published online: 21 June 2021
© The Author(s) 2021

Abstract

We present a new classification algorithm for machine learning numerical data based on direct and inverse fuzzy transforms. In our previous work fuzzy transforms were used for numerical attribute dependency in data analysis: the multi-dimensional inverse fuzzy transform was used to approximate the regression function. Also here the classification method presented is based on this operator. Strictly speaking, we apply the K-fold cross-validation algorithm for controlling the presence of over-fitting and for estimating the accuracy of the classification model: for each training (resp., testing) subset an iteration process evaluates the best fuzzy partitions of the inputs. Finally, a weighted mean of the multi-dimensional inverse fuzzy transforms calculated for each training subset (resp., testing) is used for data classification. We compare this algorithm on well-known datasets with other five classification methods.

Keywords Data classification · K-fold cross-validation algorithm · Fuzzy transform

1 Introduction

In this paper we propose a new classification algorithm, called multi-dimensional F-transform classification (for short, MFC), in which the direct and inverse multi-dimensional fuzzy transforms are used to classify instances.

Our goal is to build a robust classification model based on the multi-dimensional fuzzy transform, which integrates a K-fold cross validation resampling method to overcome the problem of data over-fitting, which represents one of the major problems of classification models.

Classification tasks (Aggarwal 2014; Duda et al. 2001; Han et al. 2012; Hastie et al. 2013; Johnson and Wichern 1992; Mitchell 1997; Mitra et al. 2002; Nirmalraj and Nagarajan 2020; Mitra et al. 2002) consist of assigning patterns to only one of predefined categories, and it is a general problem that encompasses many diverse applications. Many techniques are used for data classification such as probabilistic

and decision tree algorithms, rule-based and instance-based methods, support vector machine and neural networks algorithms (Aggarwal 2014; Witten et al. 2016).

Classification algorithms require a training phase to acquire the knowledge to be fitted in the related model and to classify patterns. Then a testing phase follows in which the performance of the algorithm is measured as well.

A good machine learning classification algorithm must be robust with respect to data under-fitting and over-fitting. Under-fitting occurs when the algorithm fails to fit sufficiently the data, producing high training and test errors, thus an under-fitting machine learning algorithm produces errors on the training data. The presence of under-fitting is evaluated by measuring a performance index after the learning process. Over-fitting represents a problem versus under-fitting and it occurs when a machine learning algorithm captures noise in the data and the data in the training dataset are optimally fitted, but the fitting itself is less accurate. The evaluation of the presence of over-fitting is more complex. There are two main techniques that we can use to limit over-fitting in machine learning algorithms: to measure the accuracy by using a validation dataset or to adopt a resampling technique by using random subsets of data.

After selecting and tuning the machine learning algorithm on the training dataset, we can evaluate the learned models

✉ Salvatore Sessa
sessa@unina.it

Ferdinando Di Martino
fdimarti@unina.it

¹ Dipartimento di Architettura, and Centro Interdipartimentale di Ricerca A. Calza Bini, Università degli Studi di Napoli Federico II, Via Toledo 402, 80134 Naples, Italy

on the validation dataset and hence to measure how the models could perform on unseen data.

In Sect. 2 we give the preliminary concepts. In Sect. 3 we recall the multi-dimensional F-transform, in Sect. 4 the MFC algorithm is presented. In Sect. 5 we present the results of our experiments: we apply the MFC algorithm in many datasets known in literature, comparing it with other classification methods. Conclusions are given in Sect. 6.

2 The MFC algorithm

Now we model the input data as a collection of instances. Each instance is characterized by a pair $(\mathbf{X}, \mathbf{Y})$, where $\mathbf{X}$ is a set of numerical attributes $(X_1, \dots, X_s)$ and $\mathbf{Y}$ is a special attribute designated as class which has C categories. We apply the fuzzy transform (for short, F-transform) algorithm (Perfileva 2006) for finding a relation between attributes in the form:

$$Y = f(x_1, \dots, x_s) \quad (1)$$

where f is a discrete function $f: [a_1, b_1] \times [a_2, b_2] \times \dots \times [a_s, b_s] \rightarrow Y \in \{1, 2, \dots, C\}$ and $x_1 \in [a_1, b_1], \dots, x_s \in [a_s, b_s]$. The F-transform algorithm is a technique used to approximate an original function “ f ”, of which a priori only a finite number values is known, but the expression of the function itself is unknown. The F-transform concept has been used in image processing (Di Martino and Sessa 2007, 2019a, b, 2020; Di Martino et al. 2008), data analysis (Di Martino et al. 2010, 2011; Di Martino and Sessa 2017a, b, 2019a, b, 2020; Novak et al. 2014; Formato et al. 2014; Perfileva et al. 2008), time

series analysis (Di Martino et al. 2011; Di Martino and Sessa 2017a, b; Johnson and Wichern 1992), forecasting problems (Di Martino et al. 2011; Di Martino and Sessa 2017a, b) and fuzzy approximation method (Tanaka et al. 1982; Khastan et al. 2015, 2017).

In Lee and Yen (2004), a classification based on the dependency between attributes in datasets is proposed. Indeed, in the paper (Di Martino and Sessa 2017a, b) an attribute dependency based on the inverse multi-dimensional F-transform is presented as regression function and in the work (Di Martino et al. 2011) as prediction function in time series. In the algorithms based on F-transforms, a set of fuzzy partitions of the input data domains is created. An assigned family of n_i fuzzy sets $A_{i1}, \dots, A_{in_i}: [a_i, b_i] \rightarrow [0, 1]$ is considered for any input variable. We say that the training dataset is *sufficiently dense* w.r.t. the fuzzy partition if, for each combination of fuzzy sets of the input variables, exists at least an instance with positive membership degree. We illustrate an example in the case of two attributes defined in the universe of discourse $[a_1, b_1] \times [a_2, b_2]$ via two fuzzy partitions of fuzzy sets $\{A_{i1}, A_{i2}, \dots, A_{in_i}\}, i = 1, 2$. If $x_{i1} = a_i < x_{i2} < \dots < x_{in_i} = b_i$ is a subdivision of $[a_i, b_i]$, then we have that the fuzzy sets $A_{i1}, \dots, A_{in_i}: [a_i, b_i] \rightarrow [0, 1]$, $i = 1, 2$, must satisfy suitable properties in each interval $[x_{i1}, x_{i2}], \dots, [x_{i(n_i-1)}, x_{in_i}]$, respectively, (cfr. Sect. 3) and they stand for the instances of the training dataset represented from red points in the example of Fig. 1. We deduce that in this example the data are not sufficiently dense w.r.t. the chosen fuzzy partitions: indeed in the grey zone there are no instance p_{1i} and p_{2j} such that $x_{1,i-1} < p_{1i} < x_{1,i+1}$ and $x_{2,j-1} < p_{2j} < x_{2,i+1}$.

Fig. 1 Example of training dataset not sufficiently dense w.r.t. the fuzzy partitions

The case of Fig. 1 is due to the too fine-grained fuzzy partition of input variable domain in the case of a family of two fuzzy sets A_1 . The model fails to fit the output for input data in the grey zone of Fig. 1.

The F-transform attribute dependency algorithm (for short, FAD), presented in Di Martino and Sessa (2007), controls the presence of under-fitting and it verifies that the dataset is not sufficiently dense with respect to the fuzzy partitions of the input variable domains. The FAD algorithm is an iterative process in which a uniform fuzzy partition with n fuzzy sets is created for each input variable. If the data are not sufficiently dense w.r.t. the fuzzy partition, the process stops and the functional dependency cannot be found, otherwise the direct and inverse multi-dimensional F-transforms are calculated. Afterwards an index of determinacy, called MADMEAN (Di Martino and Sessa 2017a, b) is calculated; this index measures if the functional model created fits the data and it ranges in $[0,1]$. The value of MADMEAN is closer to 1, the more the model fits the data. If MADMEAN is greater than a threshold value α , the process stops and the inverse multi-dimensional F-transform is used to

approximate the functional dependency, else a finer fuzzy partition is set with $n := n + 1$. In Fig. 2 we illustrate the FAD algorithm.

However, even if the training data is sufficiently dense w.r.t. the fuzzy partition, the presence of over-fitting cannot be prevented. Hence we propose a new iterative classification algorithm based on multi-dimensional F-transforms in which we apply the K-fold (K is a positive integer) cross-validation resampling algorithm to control this presence.

The K-fold cross-validation is the most popular resampling technique: it allows to train and to test the model K times on different subsets of the training dataset, moreover an estimation of a machine learning model on unseen data is obtained as well.

Performance analysis of this classification algorithm is pointed out in the works (Wong 2015, 2017). In the paper (Wen et al. 2017), a new SVM K-fold cross-validation method is proposed to increase the efficiency of such approach.

A K-fold cross-validation algorithm works under K iterations. A partition of the sample dataset into K folds (i.e.,

Fig. 2 FAD algorithm

subsets) is done. At any iteration a fold is considered as validation subset which is joined to the other folds for constituting the training set and then the classification process is performed. The main advantage of this method w.r.t. other resampling techniques is that each fold is considered once as validation set. In Fig. 3 we schematize this technique for $K=4$.

As in FAD algorithm (Di Martino and Sessa 2017a, b), by using a K -fold cross-validation technique, we can calculate two indices (say CV_1 (resp., CV_2) for evaluating the performance of our algorithm. These indices are given by the average of the percentages of misclassified instances in the training data CV_1^k (resp., testing data CV_2^k) for the k th fold, $k=1, \dots, K$:

$$CV_1 = \frac{1}{K} \sum_{k=1}^K CV_1^k \quad (2)$$

$$CV_2 = \frac{1}{K} \sum_{k=1}^K CV_2^k \quad (3)$$

We set a threshold α (resp., β) for CV_1 (resp., CV_2), then the training (resp., testing) set is partitioned in K folds and a set of fuzzy partitions is created for the input attributes, verifying that each subset is sufficiently dense w. r. t. the set of fuzzy partitions. Then the direct F-transforms are calculated in each fold. The average of the inverse F-transforms calculated for each fold is useful to classify an instance. Strictly speaking, if CV_1^k (resp., CV_2^k) = $f_{n_1 n_2 \dots n_s}^{F_k}(p_1, p_2, \dots, p_s)$ is the inverse F-transform for $x_1 = p_1, \dots, x_s = p_s$, being $n_1, \dots, n_s$ are the number of fuzzy sets of the partition, respectively, we classify the corresponding variable Y in the k th fold of the training (resp., testing) as

$$Y = f_{n_1 n_2 \dots n_s}(p_1, p_2, \dots, p_s) = \frac{1}{K} \sum_{k=1}^K f_{n_1 n_2 \dots n_s}^{F_k}(p_1, p_2, \dots, p_s) \quad (4)$$

The class label assigned to the output variable is given by

Fig. 3 Schema of a fourfold cross-validation technique

$$y = \left\lceil \min_{c=1,\dots,C} \left(\left| f_{n_1 n_2 \dots n_s}(p_1, p_2, \dots, p_s) - c \right| \right) \right\rceil \quad (5)$$

where $\lceil a \rceil$ stands for the greatest integer containing the positive real number a . Then, we calculate the two indices CV_1 and CV_2 . If $CV_1 \geq \alpha$ or $CV_2 \geq \beta$, the process is iterated by considering a finer set of fuzzy partitions, otherwise the process stops. Summarizing, the MFC algorithm consists into the following steps:

1. The number of folds K and the thresholds α and β for the two indices CV_1 and CV_2 are defined.
2. The patterns formed by the input attributes $X_1, \dots, X_s$, and the output Y are extracted from the dataset. The input attributes are numerical attributes and the output attribute contain the class label assigned to each instance.
3. The set of instances is randomly partitioned in K folds of equal dimension, by using the schema of Fig. 3.
4. For each input attribute X_i , $i = 1, \dots, s$, a uniform fuzzy partition is created with the basic functions $\{A_{i1}, A_{i2}, \dots, A_{in_i}\}$.
5. For each fold we verify that the corresponding training subset is sufficiently dense w.r.t. the set of fuzzy partitions. If for at least onefold the sufficient density is not respected, the process stops and the classification is not found.
6. For each fold the direct fuzzy transform (4) is calculated.
7. For each fold the index CV_1^k , $k = 1, \dots, K$, is calculated as the number of all the misclassified patterns in the training subset, where the classification is performed by using the average (5).
8. For each fold the index CV_2^k , $k = 1, \dots, K$, is calculated as the number of all the misclassified patterns in the testing subset, where the classification is performed by using the average (5).
9. The indices CV_1 and CV_2 are calculated via Eqs. (2) and (3), respectively. If $CV_1 \geq \alpha$ or $CV_2 \geq \beta$, a finer fuzzy partition set is considered and the process returns to the step 3.
10. The classification is validated.

In Fig. 4 we schematize the MFC algorithm.

3 Multi-dimensional F-transforms

Following the definitions and notations of Perfilieva (2006), let $n \geq 2$ and $x_1, x_2, \dots, x_n$ be points of $[a, b]$, called nodes, such that $x_1 = a < x_2 < \dots < x_n = b$. The assigned family of fuzzy sets $A_1, \dots, A_n: [a, b] \rightarrow [0, 1]$, called basic

functions, is a fuzzy partition of $[a, b]$ if the following properties hold:

- (1) $A_i(x_i) = 1$ for every $i = 1, 2, \dots, n$;
 - (2) $A_i(x) = 0$ if $x \notin (x_{i-1}, x_{i+1})$ for $i = 2, \dots, n$;
 - (3) $A_i(x)$ is a continuous function on $[a, b]$;
 - (4) $A_i(x)$ strictly increases on $[x_{i-1}, x_i]$ for $i = 2, \dots, n$ and strictly decreases on $[x_i, x_{i+1}]$ for $i = 1, \dots, n-1$;
 - (5) $\sum_{i=1}^n A_i(x) = 1$ for every $x \in [a, b]$.
- We say that the fuzzy sets $\{A_1(x), \dots, A_n(x)\}$ form a uniform fuzzy partition if
- (6) $n \geq 3$ and $x_i = a + h \cdot (i-1)$, where $h = (b-a)/(n-1)$ and $i = 1, 2, \dots, n$ (that is, the nodes are equidistant);
 - (7) $A_i(x_i - x) = A_i(x_i + x)$ for every $x \in [0, h]$ and $i = 2, \dots, n-1$;
 - (8) $A_{i+1}(x) = A_i(x-h)$ for every $x \in [x_i, x_{i+1}]$ and $i = 1, 2, \dots, n-1$.

Avoiding the case of a continuous function studied in the paper (Perfilieva 2006) and considering the discrete case which is applicable in many real situations, we know that the function $f: [a, b] \rightarrow [0, 1]$ assumes assigned values in the points $P = \{p_1, \dots, p_m\} \subset [a, b]$. If the set $P = \{p_1, \dots, p_m\}$ is sufficiently dense w.r.t. the fixed partition $\{A_1, A_2, \dots, A_n\}$, that is for each $i = 1, \dots, n$, there exists an index $v \in \{1, \dots, m\}$ such that $A_i(p_v) > 0$, we define the n -tuple $(F_1, F_2, \dots, F_n)$ as the direct F-transform of f w.r.t. $\{A_1, A_2, \dots, A_n\}$, where each F_i is given by

$$F_i = \frac{\sum_{j=1}^m f(p_j) A_i(p_j)}{\sum_{j=1}^m A_i(p_j)} \quad (6)$$

for $i = 1, \dots, n$. Similarly, we define the inverse F-transform of f w.r.t. $\{A_1, A_2, \dots, A_n\}$ by setting

$$f_{F,n}(p_j) = \sum_{i=1}^n F_i A_i(p_j) \quad (7)$$

for every $j \in \{1, \dots, m\}$. We have the following theorem (Perfilieva 2006):

Theorem 1 *Let $f(x)$ be a function assigned on a set $P = \{p_1, \dots, p_m\} \subset [a, b]$ and assuming values in $[0, 1]$. Then for every $\varepsilon > 0$, there exists an integer $n(\varepsilon)$ and a related fuzzy partition $\{A_1, A_2, \dots, A_{n(\varepsilon)}\}$ of $[a, b]$ for which there P is sufficiently dense. Moreover, for every $p_j \in [a, b]$, $j = 1, \dots, m$, the following inequality holds:*

$$|f(p_j) - f_{F,n(\varepsilon)}(p_j)| < \varepsilon \quad (8)$$

Fig. 4 Schema of the MFC algorithm

The above concepts can be extended to functions in k (≥ 2) variables. Let the Cartesian product $[a_1, b_1] \times [a_2, b_2] \times \dots \times [a_s, b_s]$ of s intervals $[a_i, b_i]$, $i = 1, \dots, s$, be the universe of discourse. In the discrete case, the function $f(x_1, \dots, x_s)$ assumes determined values in m points $(p_{j1}, p_{j2}, \dots, p_{js}) \in [a_1, b_1] \times [a_2, b_2] \times \dots \times [a_s, b_s]$ for $j = 1, \dots, m$. We say that the set $P = \{(p_{11}, p_{12}, \dots, p_{1s}), (p_{21}, p_{22}, \dots, p_{2s}), \dots, (p_{m1}, p_{m2}, \dots, p_{ms})\}$ is sufficiently dense w.r.t. the chosen partitions $\{A_{11}, A_{12}, \dots, A_{1n_1}\}, \dots, \{A_{s1}, A_{s2}, \dots, A_{sn_s}\}$ if for any $\{h_1, \dots, h_s\} \in \{1, \dots, n_1\} \times \dots \times \{1, \dots, n_s\}$, there is some $(p_{v1}, p_{v2}, \dots, p_{vs}) \in P$, $v \in \{1, \dots, m\}$, such that $A_{1h_1}(p_{v1}) \cdot A_{2h_2}(p_{v2}) \cdot \dots \cdot A_{sh_s}(p_{vs}) > 0$. So we can define the $(h_1, h_2, \dots, h_s)$ th component $F_{h_1 h_2 \dots h_s}$ of

the direct F-transform of f w.r.t. the basic functions $\{A_{11}, A_{12}, \dots, A_{1n_1}\}, \dots, \{A_{s1}, A_{s2}, \dots, A_{sn_s}\}$ as

$$F_{h_1 h_2 \dots h_s} = \frac{\sum_{j=1}^m f(p_{j1}, p_{j2}, \dots, p_{js}) \cdot A_{1h_1}(p_{j1}) \cdot A_{2h_2}(p_{j2}) \cdot \dots \cdot A_{sh_s}(p_{js})}{\sum_{j=1}^m A_{1h_1}(p_{j1}) \cdot A_{2h_2}(p_{j2}) \cdot \dots \cdot A_{sh_s}(p_{js})} \quad (9)$$

Now we define the inverse F-transform of f w.r.t. the basic functions $\{A_{11}, A_{12}, \dots, A_{1n_1}\}, \dots, \{A_{s1}, A_{s2}, \dots, A_{sn_s}\}$ to be the following function by setting for each point $(p_{j1}, p_{j2}, \dots, p_{js}) \in [a_1, b_1] \times \dots \times [a_s, b_s]$:

$$f_{n_1 n_2 \dots n_s}^F(p_{j1}, p_{j2}, \dots, p_{js}) = \sum_{h_1=1}^{n_1} \sum_{h_2=1}^{n_2} \dots \sum_{h_s=1}^{n_s} F_{h_1 h_2 \dots h_s} \cdot A_{1h_1}(p_{j1}) \cdot \dots \cdot A_{sh_s}(p_{js}) \quad (10)$$

for $j=1, \dots, m$. The following theorem, which is an extension of Theorem 1, holds:

Theorem 2 Let $f(x_1, \dots, x_s)$ be a function assigned on the set of points $P = \{(p_{11}, p_{12}, \dots, p_{1s}), (p_{21}, p_{22}, \dots, p_{2s}), \dots, (p_{m1}, p_{m2}, \dots, p_{ms})\} \subset [a_1, b_1] \times [a_2, b_2] \times \dots \times [a_s, b_s]$ assuming values in $[0, 1]$. Then for every $\varepsilon > 0$, there exist integers $n_1(\varepsilon), \dots, n_s(\varepsilon)$ and related fuzzy partitions

$$\{A_{11}, A_{12}, \dots, A_{1n_1(\varepsilon)}\}, \dots, \{A_{s1}, A_{s2}, \dots, A_{sn_s(\varepsilon)}\} \quad (11)$$

such that the set P is sufficiently dense w.r.t. (11). Moreover, for every $(p_{j1}, p_{j2}, \dots, p_{js}) \in P, j=1, \dots, m$, the following inequality holds:

$$|f(p_{j1}, p_{j2}, \dots, p_{js}) - f_{n_1(\varepsilon)n_2(\varepsilon)}^F \dots f_{n_s(\varepsilon)}^F(p_{j1}, p_{j2}, \dots, p_{js})| < \varepsilon \quad (12)$$

4 Proposed algorithm

Continuing to maintain notations given in Sect. 1, we consider a dataset formed by s attributes $X_1 = [a_1, b_1], \dots, X_i = [a_i, b_i], \dots, X_s = [a_s, b_s]$, and an attribute $Y \in \{1, \dots, C\}$. We try to find a dependency between attributes in the form (1), where f is a discrete function $f: [a_1, b_1] \times [a_2, b_2] \times \dots \times [a_s, b_s] \rightarrow \{1, \dots, C\}$.

In the MFC algorithm we apply the K-fold validation algorithm by partitioning randomly the dataset in K folds having the same number of instances. Each fold is formed by a training subset with $m = \frac{K-1}{K} \cdot M$ instances and by a testing subset with $\frac{M}{K}$ instances, where M is the number of instances in the original dataset. Now we schematize the training subset in a fold with m instances $O_1, \dots, O_j, \dots, O_m$ in the following tabular form (Di Martino and Sessa 2017a, b):

	X_1	...	X_s	Y
O_1	p_{11}		p_{1s}	y_1
.	.	.	.	.
.	.	.	.	.
O_j	p_{j1}	.	p_{js}	y_j
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
O_m	p_{m1}	.	p_{ms}	y_m

where p_{ji} is the value of the attribute X_i for the instance O_j . Each attribute X_i can be considered as a numerical variable assuming values in the domain $[a_i, b_i]$, where $a_i = \min\{p_{1i}, \dots, p_{mi}\}$ and $b_i = \max\{p_{1i}, \dots, p_{mi}\}$. The parameters (to be fixed) are the number K of folds and the threshold α (resp., β) for CV_1 (resp., CV_2). We first construct

a uniform fuzzy partition for the domain of each input attribute $[a_i, b_i]$ constituted by following n_i basic functions defined as

$$A_{i1}(x) = \begin{cases} 0.5 \cdot \left(1 + \cos \frac{\pi}{h_i}(x - x_{i1})\right) & \text{if } x \in [x_{i1}, x_{i2}] \\ 0 & \text{otherwise} \end{cases}$$

$$A_{ij}(x) = \begin{cases} 0.5 \cdot \left(1 + \cos \frac{\pi}{h_i}(x - x_{ij})\right) & \text{if } x \in [x_{i(j-1)}, x_{i(j+1)}] \\ 0 & \text{otherwise} \end{cases}$$

$$A_{ini}(x) = \begin{cases} 0.5 \cdot \left(1 + \cos \frac{\pi}{h_i}(x - x_{ini})\right) & \text{if } x \in [x_{i(n-1)}, x_{ini}] \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

where $h_i = (b_i - a_i)/(n_i - 1)$, $x_{ij} = a_i + h_i \cdot (j - 1)$, $i = 1, 2, \dots, s$ and $j = 1, 2, \dots, n_i$. Initially we consider the coarsest grained uniform fuzzy partition by setting $n_1 = n_2 = \dots = n_s = 3$. Afterwards in each fold, we control that the data are sufficiently dense w.r.t. the fuzzy partition, that is we verify that for any combination of basic functions $A_{1h_1}, \dots, A_{sh_s}$, there exists at least one instance O_j such that $A_{1h_1}(p_{j1}) \cdot \dots \cdot A_{sh_s}(p_{js}) > 0$. If this condition is not satisfied in any fold, the process stops and the classification is not found, otherwise the direct F-transform is calculated in each fold.

By setting $f(p_{j1}, p_{j2}, \dots, p_{js}) = y_j$ for $j = 1, 2, \dots, m$, the direct F-transform of f is constructed by using Eq. (9) in the following form:

$$F_{h_1 h_2 \dots h_s}^k = \frac{\sum_{j=1}^m y_j \cdot A_{1h_1}(p_{j1}) \cdot \dots \cdot A_{sh_s}(p_{js})}{\sum_{j=1}^m A_{1h_1}(p_{j1}) \cdot \dots \cdot A_{sh_s}(p_{js})} \quad (14)$$

where $k = 1, \dots, K$. Given a set of values $p_{11}, \dots, p_{1s}, \dots, p_{ms}$ for the input variables $X_1, \dots, X_i, \dots, X_s$, we calculate the inverse F-transform $f_{n_1 n_2 \dots n_s}^{F^k}$ for each fold, defined as

$$f_{n_1 n_2 \dots n_s}^{F^k}(p_1, p_2, \dots, p_s) = \sum_{h_1=1}^{n_1} \sum_{h_2=1}^{n_2} \dots \sum_{h_s=1}^{n_s} F_{h_1 h_2 \dots h_s}^k \cdot A_{1h_1}(p_1) \cdot \dots \cdot A_{sh_s}(p_s) \quad (15)$$

The algorithm is schematized below in the form of pseudocode. The function Classify() return the class y_f assigned to a pattern with input values $p = (p_1, \dots, p_s)$ by using the inverse F-transform. The MFC algorithm compares the resulting class with the value y of the attribute Y assigned in the dataset to calculate the numbers of misclassified patterns CV_1^k and CV_2^k , for the k th training and testing subset, respectively, and to calculate the two indexes CV_1 and CV_2 .

Algorithm MFC

Arguments: α, β, K

Return value: Boolean values (TRUE = “classification found”, FALSE = “classification not found”)

Input: Dataset composed by m instances with input attributes $X_1, X_2, \dots, X_s$ and class attribute Y

Output: Direct F-transform components

```

1  status := FALSE
2   $n_1 := 3, \dots, n_s := 3$ 
3  WHILE (status == FALSE)
4    Partition randomly the dataset in  $K$  subsets creating the  $K$  folds
5    FOR  $i = 1$  to  $s$ 
6      Create a uniform fuzzy partition  $\{A_{i1}, A_{i2}, \dots, A_{in_i}\}$  of the interval  $[a_i, b_i]$  by using the basic functions (13)
7    NEXT  $i$ 
8    FOR  $k = 1$  to  $K$ 
9      IF the  $k$ th training subset is not dense w.r.t. the fuzzy partition product THEN
10     RETURN status // “Classification not found”
11   ELSE
12     Calculate the direct F-Transform components by using Eq. (9)
13   END IF
14   NEXT  $k$ 
15    $CV_1 := 0$ 
16    $CV_2 := 0$ 
17   FOR  $k = 1$  to  $K$ 
18      $CV_1^k := 0$ 
19     FOR each  $(p, y)$  in the  $k$ -th learning subset
20        $y_f = \text{Classify}(C, p)$  // class assigned to the pattern  $p$ 
21       IF  $(y_f \neq y)$  THEN
22          $C_{1k} := C_{1k} + 1$ 
23       END IF
24        $CV_1 := CV_1 + CV_1^k$ 
25      $CV_2^k := 0$ 
26     FOR each  $(p, y)$  in the  $k$ -th testing subset
27        $y_f = \text{Classify}(C, p)$  // class assigned to the pattern  $p$ 
28       IF  $(y_f \neq y)$  THEN
29          $CV_2^k := CV_2^k + 1$ 
30       END IF
31        $CV_2 := CV_2 + CV_2^k$ 
32     NEXT  $k$ 
33      $CV_1 := CV_1 / K$ 
34      $CV_2 := CV_2 / K$ 
35     IF  $(CV_1 \geq \alpha)$  OR  $(CV_2 \geq \beta)$  THEN
36       Set greater values for  $n_1, \dots, n_s$  // a finer fuzzy partitions set is selected
37     ELSE
38       Status := TRUE
39     END IF
40   END WHILE
41   RETURN status // “Classification found”
42 END

```

Function classify

Arguments: $C, p_1, \dots, p_s$

Return value: Class value calculated for the attribute Y

Input: Direct F-transform components

Output:

```

1   $y := 1$ 
2  FOR  $k = 1$  to  $K$ 
3    Calculate the inverse F-Transform
       $f^{F^k}_{n_1 n_2 \dots n_s}(p_1, p_2, \dots, p_s)$  by using Eq. (15)
4     $f := f + f^{F^k}_{n_1 n_2 \dots n_s}(p_1, p_2, \dots, p_s)$ 
5  NEXT  $k$ 
6   $f := f / K$ 
7   $w := |f - 1|$ 
8  FOR  $c = 2$  to  $C$ 
9    IF  $|f - c| < w$  THEN
10      $y := c$ 
11   END IF
12 NEXT  $c$ 
13 RETURN  $y$ 
14 END

```

5 Tests and results

Now we present our experiments on known sample of over 100 datasets extracted from UCI machine Learning repository (<http://archive.ics.uci.edu/ml/datasets.html>) and from Knowledge Extraction Evolution Learning (KEEL) dataset (<http://sci2s.ugr.es/keel/category.php?cat=clas>).

In each experiment we randomly select a set of instances to be stored in a testing dataset. The other instances are randomly partitioned in K folds. We set $K = 10$, $\alpha = 2\%$, $\beta = 4\%$. After application of the MFC algorithm, we use the classifier on the testing dataset obtaining the error as percentage of misclassified instances.

For brevity, we present the detailed results obtained only for four datasets: IRIS, balance-scale, banana and wine.

Now present the results for IRIS dataset which is formed by 150 instances and 5 attributes. The dataset consists of 50 samples, each of three species of plant: *Iris setosa*, *Iris versicolor* and *Iris virginica*. For each instance the length and the width in cm of the sepals and petals are measured. Based on the combination of these four features, the biologist R. Fisher (https://en.wikipedia.org/wiki/Iris_flower_data_set) developed a linear discriminant model to distinguish the species from each other. It is well known that only the *Iris setosa* is well linearly separable from the other two. In Table 1 we show the attributes in the IRIS dataset.

Table 1 Attributes in the IRIS dataset

Attribute	Description
Sepal length (cm)	Length of the sepal
Sepal width (cm)	Width of the sepal
Petal length (cm)	Length of the petal
Petal width (cm)	Width of the petal
Class	Type of iris plant <i>Iris setosa</i> <i>Iris versicolour</i> <i>Iris virginica</i>

Table 2 CV_1 and CV_2 indices obtained for each fold and their average values

Fold	CV_1 (%)	CV_2 (%)
01	0.74	0.00
02	1.48	6.67
03	2.22	6.67
04	2.22	6.67
05	0.74	0.00
06	0.74	0.00
07	0.74	0.00
08	1.48	6.67
09	2.22	6.67
10	0.74	0.00
Mean	1.33	3.33

In training (resp., testing) phase, each fold has $m = 135$ (resp., 15) instances. In Table 2, we show CV_1 and CV_2 obtained for each fold.

We use a testing dataset of 50 instances downloaded from the IRIS WEB page project (<http://netcologne.dl.sourceforge.net/project/irisdss/IRIS%20Test%20data.txt>). Table 3 shows the MFC classifies improperly the data only in one case.

Now we present the results obtained by applying the MFC algorithm on the balance-scale dataset: the attributes are the left weight, the left distance, the right weight and the right distance (Table 4).

The dataset is composed of 625 instances. We select randomly 600 instances, storing the other 25 instances for testing the classification results, we set $K = 10$ and each fold contains 540 (resp., 60) training (resp., testing) instances. In Table 5, we show the results obtained for each fold.

We apply the classification on the testing dataset of 25 instances. In Table 6, we show the results obtained. The instance is improperly classified only in one case.

We present a third experiment in which we apply the MFC algorithm to the Keel dataset Banana dataset by the Fraunhofer Intelligent Data Analysis Group (<https://www.iais.fraunhofer.de/>). It contains instances formed by three attributes: the first two ones represent the X and Y axis of the

Table 3 Test results obtained for a sample of 50 instances

Class	Number of instances	Wrongly classified instances	Percentage error (%)
<i>Iris setosa</i>	17	0	0.00
<i>Iris versicolour</i>	15	0	0.00
<i>Iris virginica</i>	18	1	5.56
Total	50	1	

Table 4 Attributes in the balance scale dataset

Attribute	Description
Left weight (cm)	Left weight (discrete value between 1 and 5)
Left distance (cm)	Left distance (discrete value between 1 and 5)
Right weight (cm)	Right weight (discrete value between 1 and 5)
Right distance (cm)	Right distance (discrete value between 1 and 5)
Class	Class name L (1) B (2) R (3)

Table 5 CV_1 and CV_2 indices obtained for each fold and their average values

Fold	CV_1 (%)	CV_2 (%)
01	3.38	6.35
02	3.02	4.76
03	3.20	6.35
04	2.67	4.76
05	3.02	6.35
06	2.85	4.76
07	3.38	6.35
08	2.85	4.76
09	3.20	6.35
10	3.02	6.35
Mean	3.06	5.71

cluster of banana's form, the last attribute assumes values -1 and 1 representing two forms of banana (Table 7).

The dataset contains 5300 instances. We select randomly 300 instances for the final testing phase and we partition randomly the other 5000 instances creating tenfold. Each fold consists of 4500 (resp., 500) training (resp., testing) instances. In Table 8, we show CV_1 and CV_2 obtained for each fold and their averages. In Table 9, we show the results obtained on the testing dataset.

Finally, we show the results of the tests performed on the UCI machine learning Wine dataset. This dataset is given by 178 instances having 13 numerical features characterizing the chemical composition of an Italian wine. Each wine belongs to one of three different crops; the

Table 6 Test results obtained for a sample of 25 instances

Class	Number of instances	Wrongly classified instances	Percentage error (%)
L (1)	7	0	0.00
B (2)	10	1	10.00
R (3)	8	0	0.00
Total	25	1	

Table 7 Attributes in the banana dataset

Attribute	Description
At1 (cm)	X axis
At2 (cm)	Y axis
Class	Class name - 1 1

Table 8 CV_1 and CV_2 indices obtained for each fold and their average values

Fold	CV_1 (%)	CV_2 (%)
01	0.62	1.60
02	0.58	1.40
03	0.56	1.40
04	0.60	1.60
05	0.67	1.80
06	0.56	1.60
07	0.69	1.80
08	0.58	1.40
09	0.62	1.60
10	0.56	1.40
Mean	0.60	1.56

Table 9 Test results obtained for a sample of 300 instances

Class	Number of instances	Wrongly classified instances	Percentage error (%)
- 1	156	2	1.28
1	144	1	0.64
Total	300	3	

dataset is partitioned in three classes identified as 1, 2 and 3 whose belong 59, 71, 48 instances, respectively.

We select randomly 150 instances, storing the other 28 instances for testing the classification results. We construct $K = \text{tenfold}$ containing 150 (resp., 28) training (resp., testing) instances. In Table 10 (resp., 11), we show the values of CV_1 and CV_2 obtained for each fold and their averages (resp., testing dataset) (Table 11).

Table 10 CV_1 and CV_2 indices obtained for each fold and their average values

Fold	CV_1 (%)	CV_2 (%)
01	1.17	2.08
02	1.22	2.24
03	1.13	2.01
04	1.35	2.41
05	1.28	2.19
06	1.10	1.94
07	1.24	2.03
08	1.31	2.15
09	1.26	2.12
10	1.16	2.04
Mean	1.23	2.13

Table 11 Test results obtained for a sample of 28 instances

Class	Number of instances	Wrongly classified instances	Percentage error (%)
1	10	0	0.00
2	10	1	10.00
3	8	0	0.00
Total	28	1	

For completeness, we present a comparison of the MFC algorithms with classification algorithms implemented in the mining tool WEKA (Waikato Environment for Knowledge Analysis, <https://www.cs.waikato.ac.nz/ml/weka/>; Witten et al. 2016). For each algorithm, we set $K = 10$. In Table 12, we show the mean running time (in ms) and the testing errors obtained by applying the MFC algorithm and the classification algorithms Decision tree-based J48 (Bhargawa et al. 2013; Mitchell 1997), Multi-Layer Perceptron (Pal and Mitra 1992, 2004; Chaudhuri and Bhattacharya 2007), naive Bayes (Dimitoglou et al. 2012; Panda and Matra 2007) and Lazy K-Nearest Neighbor IBK (Aha 1997; Bhargawa et al. 2013; Jiang et al. 2007). In Table 13, we show the mean running time, the mean and the standard deviation of the testing percentage errors obtained by applying the five classification algorithms to all the datasets used in our tests.

These results show that the performance of the MFC algorithm, obtained by setting $\alpha = 2\%$ and $\beta = 4\%$, are comparable with the ones obtained applying the Decision tree J48 and the Multilayer Perceptron algorithms. Moreover, MFC produces better performance results than the Lazy IBK and naive Bayes algorithms, although it gives slightly higher running times.

In Table 14, we show the mean accuracy, precision and recall classification measures obtained by running the five algorithms on all the test datasets.

Table 12 Running time, misclassified instances and testing percentage errors obtained for the datasets IRIS, balance scale, banana and wine by using different classification algorithms

Algorithm	Dataset	Running time (ms)	Misclassified instances	Testing percentage error (%)
MFC classifier	IRIS	54.22	1	2.00
	Balance scale	78.12	1	4.00
	Banana	122.38	3	1.00
	Wine	62.35	1	3.57
Decision tree J48	IRIS	48.25	1	2.00
	Balance scale	65.70	0	0.00
	Banana	99.63	2	0.67
	Wine	55.58	1	3.57
Multilayer perceptron	IRIS	51.16	1	2.00
	Balance scale	71.84	0	0.00
	Banana	114.29	2	0.67
	Wine	58.21	1	3.57
Naive Bayes	IRIS	43.94	2	4.00
	Balance scale	56.88	1	4.00
	Banana	85.11	5	1.67
	Wine	46.89	3	10.7
Lazy IBK	IRIS	47.23	2	4.00
	Balance scale	60.12	1	4.00
	Banana	92.24	4	1.33
	Wine	51.41	2	7.14

Table 13 Mean running time and mean and standard deviation testing percentage error obtained by using the 5 classification algorithms

Algorithm	Mean running time (ms)	Mean testing percentage error (%)	STD testing percentage error (%)
MFC classifier	81.22	1.16	1.17
Decision tree J48	73.66	0.95	1.15
Multilayer perceptron	76.18	0.98	1.16
Naive Bayes	69.34	3.21	1.41
Lazy IBK	74.17	2.38	1.25

These results show that MFC, in addition to producing accurate results, also has high precision and high sensitivity, comparable to those obtained using the Decision tree J48 and Multilayer Perceptron algorithms.

6 Conclusions

In this paper we propose a new classification algorithm based on direct and inverse F-transforms for machine learning data. We apply the K-fold cross-validation resampling algorithm to control the presence of over-fitting in the data. The MFC algorithm is an iterative algorithm in which the dataset is partitioned randomly in K subsets, then the

Table 14 Mean accuracy, precision and recall with 5 classification algorithms

Algorithm	Accuracy (%)	Precision (%)	Recall (%)
MFC classifier	98.15	98.09	97.36
Decision tree J48	98.38	98.17	97.51
Multilayer perceptron	98.22	98.23	97.48
Naive Bayes	96.55	91.89	90.65
Lazy IBK	97.17	93.30	91.44

algorithm calculates the performance indices CV_1^k (resp., CV_2^k) for each fold: they correspond to the percentage of misclassified instances in the training (resp., testing) subsets. Initially a coarse-grained fuzzy partition of the data domain is set: if the two indices are greater or equals to two specified thresholds for each fold, then the algorithm stops otherwise a finer fuzzy partition of the data domain is set.

We compare the MFC algorithm with other known classification algorithms implemented in the mining tool WEKA. The results obtained for over 100 classification datasets show that the performances of the MFC algorithms are better than the ones obtained by using the naive Bayes and Lazy IBK algorithms and they are comparable with the ones obtained by the Decision tree J48 and the Multilayer Perceptron algorithms.

The MFC algorithm has an high computational complexity when used on large and high-dimensional datasets and this is a shortcoming. In the future we intend to improve the algorithm to manage massive and multi-dimensional datasets. The inclusion in the preprocessing phase of feature selection techniques and the use of data compression methods allows to reduce the computational complexity of the search for the optimal fuzzy partition and the calculation of the components of the multi-dimensional direct F- transforms.

Funding Open access funding provided by Università degli Studi di Napoli Federico II within the CRUI-CARE Agreement. No funds were received for this research.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki Declaration and its later amendments or comparable ethical standards.

Informed consent Informed consent was obtained from all individual participants included in the study.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Aggarwal C (2014) Data classification: algorithms and application. CRC Press, Taylor & Francis Group, Abingdon. <https://doi.org/10.1201/b17320>
- Aha D (ed) (1997) Lazy learning. Kluwer Academic Publishers, Norwell. <https://doi.org/10.1007/9789401720533>
- Bhargava N, Sharma G, Bhargava R, Mathuria M (2013) Decision tree analysis on J48 algorithm for data mining. *Int J Adv Res Comput Sci Softw Eng* 3(6):1114–1119 (ISSN (Print): 22776451, ISSN (Online): 2277128X)
- Chaudhuri BB, Bhattacharya U (2007) Efficient training and improved performance of multilayer perceptron in pattern classification. *Neurocomputing* 34:11–27. [https://doi.org/10.1016/S0925-2312\(00\)003052](https://doi.org/10.1016/S0925-2312(00)003052)
- Di Martino F, Sessa S (2007) Compression and decompression of image with discrete fuzzy transforms. *Inf Sci* 177:2349–2362. <https://doi.org/10.1016/j.ins.2006.12.027>
- Di Martino F, Sessa S (2017a) Time series seasonal analysis based on fuzzy transforms. *Symmetry*. <https://doi.org/10.3390/sym9110281>
- Di Martino F, Sessa S (2017b) A fuzzy transform prediction in spatial analysis and its application to demographic balance data. *Soft Comput* 21(13):3537–3550. <https://doi.org/10.1007/s0050001726218>
- Di Martino F, Sessa S (2019a) Multilevel fuzzy transforms image compression. *J Ambient Intell Humaniz Comput* 10:2745–2756. <https://doi.org/10.1007/s1265201809714>
- Di Martino F, Sessa S (2019b) Fragile watermarking tamper detection via bilinear fuzzy relation equations. *J Ambient Intell Humaniz Comput* 10:2041–2061. <https://doi.org/10.1007/s1265201808063>
- Di Martino F, Sessa S (2020) Fuzzy transforms for image processing and data analysis core concepts, processes and applications. Springer, p 217 (ISBN 9783030446123)
- Di Martino F, Loia V, Perfilieva I, Sessa S (2008) An image coding/decoding method based on direct and inverse fuzzy transforms. *Int J Approx Reason* 48:110–131. <https://doi.org/10.1016/j.ijar.2007.06.008>
- Di Martino F, Loia V, Sessa S (2010) Fuzzy transforms method and attribute dependency in data analysis. *Inf Sci* 180(4):493–505. <https://doi.org/10.1016/j.ins.2009.10.012>
- Di Martino F, Loia V, Sessa S (2011) Fuzzy transforms method in prediction data analysis. *Fuzzy Sets Syst* 180(1):146–163. <https://doi.org/10.1016/j.fss.2010.11.009>
- Dimitoglou G, Adams JA, Jim CM (2012) Comparison of the C4.5 and a naive Bayes classifier for the prediction of lung cancer survivability. *J Comput* 4(2):1–9 (arXiv:1206.1121v2)
- Duda R, Hart P, Stork D (2001) Pattern classification, 2nd edn. John Wiley & Sons, Hoboken (ISBN: 9780471056690)
- Formato G, Troiano L, Vaccaro A (2014) Achieving consensus in self-organizing multi agent systems for smart microgrids computing in the presence of interval uncertainty. *J Ambient Intell Humaniz Comput* 5:821–828. <https://doi.org/10.1007/s1265201402311>
- Han M, Kamber M, Pei J (2012) Data mining: concepts and techniques, 3rd edn. Morgan Kaufmann (Elsevier) (ISBN 9780123814791, eBook ISBN: 9780123814807)
- Hastie T, Tibshirani R, Friedman J (2013) The elements of statistical learning: data mining, inference and prediction. Springer Verlag, New York. <https://doi.org/10.1007/9780387848587>
- Jiang L, Cai Z, Wang D, Jiang S (2007) Survey of improving K-nearest neighbor for classification. In: Fourth international conference on fuzzy systems and knowledge discovery, IEEE Computer Society Press, China. <https://doi.org/10.1109/FSKD.2007.552>
- Johnson RA, Wichern DW (1992) Applied multivariate statistical analysis. Prentice Hall International, London (ISBN 0131877151)
- Khastan A, Perfilieva I, Alijani Z (2015) A new fuzzy approximation method to Cauchy problems by fuzzy transform. *Fuzzy Sets Syst* 288:75–95. <https://doi.org/10.1016/j.fss.2015.01.001>[17]
- Khastan A, Alijani AZ, Perfilieva I (2017) Fuzzy transform to approximate solution of two-point boundary value problems. *Math Methods Appl Sci* 40(17):6147–6154. <https://doi.org/10.1002/mma.3832>
- Lee YS, Yen SJ (2004) Classification based on attribute dependency. In: Proceedings of 6th international conference DaWaK' 04, Lecture Notes in Computer Sciences, Springer, vol 5192, pp 259–268
- Mitchell TM (1997) Does machine learning really work. *AI Mag.* <https://doi.org/10.1609/aimag.v18i3.1303>
- Mitra S, Pal SK, Mitra P (2002) Data mining in soft computing framework: a survey. *IEEE Trans Neural Netw* 13(1):3–14. <https://doi.org/10.1109/72.977258>
- Nirmalraj S, Nagarajan G (2020) Biomedical image compression using fuzz transform and deterministic binary compressive sensing matrix. *J Ambient Intell Humaniz Comput.* <https://doi.org/10.1007/s1265202002103x>

- Novák V, Perfilieva I, Kreinovich V (2014) Filtering out high frequencies in time series using F-transform. *Inf Sci* 274(1):192–209. <https://doi.org/10.1016/j.ins.2014.02.133>
- Pal SK, Mitra S (1992) Multilayer perceptron, fuzzy sets, and classification. *IEEE Trans Neural Netw* 3(5):683–697. <https://doi.org/10.1109/72.159058>
- Pal SK, Mitra S (2004) Pattern recognition algorithms for data mining. CRC Press, Boca Raton (**ISBN 9780367394240**)
- Panda M, Matra M (2007) Network intrusion detection using naïve Bayes. *Int J Comput Sci Netw Secur* 12:258–263 (**Corpus ID: 6064057**)
- Perfilieva I (2006) Fuzzy transforms: theory and applications. *Fuzzy Sets Syst* 157(8):993–1023. <https://doi.org/10.1016/j.fss.2005.11.012>
- Perfilieva I, Novák V, Dvorák A (2008) Fuzzy transforms in the analysis of data. *Int J Approx Reason* 48:36–46. <https://doi.org/10.1016/j.ijar.2007.06.003>
- Tanaka H, Uejima S, Asa K (1982) Linear regression analysis with fuzzy model. *IEEE Trans Syst Man Cybern* 12(6):903–907. <https://doi.org/10.1109/TSMC.1982.4308925>
- Wen Z, Li B, Ramamohanarao K, Chen G, Chen Y, Zhang R (2017) Performance improving efficiency of SVM K-fold cross validation by alpha seed. In: *Proceedings of the AAAI conference on artificial intelligence*, San Francisco (USA), AAAI Press, pp 2768–2774
- Witten IH, Frank E, Hall MA, Pal CJ (2016) Data mining: practical machine learning tools and techniques. Morgan Kaufmann (Elsevier) (**ISBN: 9780123748560; eBook ISBN: 9780080890364**)
- Wong TT (2015) Performance evaluation of classification algorithms by K-fold and leave one out cross validation. *Pattern Recognit* 48(9):2839–2846. <https://doi.org/10.1016/j.patcog.2015.03.009>
- Wong TT (2017) Parametric methods for comparing the performance of two classification algorithms evaluated by K-fold cross validation on multiple data sets. *Pattern Recognit* 65:97–107. <https://doi.org/10.1016/j.patcog.2016.12.018>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.