



# Knowledge aggregation in decision-making process with C-fuzzy random forest using OWA operators

Łukasz Gadomer<sup>1</sup> · Zenon A. Sosnowski<sup>1</sup>

Published online: 31 January 2018  
© The Author(s) 2018. This article is an open access publication

## Abstract

The idea of knowledge aggregation contained in C-fuzzy decision tree nodes with OWA operators during the C-fuzzy random forest decision-making process is presented in this paper. C-fuzzy random forest is a new kind of ensemble classifier which consists of C-fuzzy decision trees. There are proposed three kinds of OWA operators for the given problem, called Local OWA, global OWA for each tree in the forest and global OWA for the whole forest. Weights of OWA operators are optimized using a genetic algorithm. In order to evaluate the created classifier, experiments were performed using ten datasets. The classifier was checked in comparison with C4.5 rev. 8 decision tree and single C-fuzzy decision tree. The influence of randomness and proposed OWA operators on the classification accuracy was tested.

**Keywords** C-fuzzy random forest · OWA operators · C-fuzzy decision tree

## 1 Introduction

In this paper we would like to present the method of using OWA operators in C-fuzzy random forest classification process. These operators aggregate the knowledge contained in C-fuzzy decision tree nodes which are part of C-fuzzy random forest. This kind of forest is a classification solution which joins fuzzy random forest and C-fuzzy decision trees. We have presented the idea of C-fuzzy random forest on CISIM 2016 conference and described it in conference materials (Gadomer and Sosnowski 2016). Preliminary results achieved on four datasets were also presented there. In this paper we proposed the way of using OWA operator in our classifier. We also improved the research methodology and performed complete experiments on ten datasets. The main purpose of this work was to expand the classifier created before on OWA operators in order to achieve the flexible ensemble classifier which can be applied to many different datasets. Thanks to this flexibility, the classifier should allow

to achieve results better than the other classifiers (or at least comparable to them), which would make it a competitive solution that could be chosen to deal with many classification problems.

In the first part of this paper, three main solutions connected with the created ensemble classifier are presented: C-fuzzy decision trees (Sect. 1.1), fuzzy random forest (Sect. 1.2) and OWA operators (Sect. 1.3). Then the details of C-fuzzy random forest classifier are described in Sect. 2. After that, the idea of knowledge aggregation contained in C-fuzzy decision trees which are part of C-fuzzy random forest with OWA operators is presented in Sect. 3. Three kinds of OWA operators are also shown which we proposed and used in our classifier. In Sect. 4 the experiments are described, and in Sect. 5 results are discussed. The classification accuracy using C-fuzzy random forest with OWA operators is compared with C4.5 decision tree and C-fuzzy decision trees working singly. The results of classification using different OWA Operators are compared. Also, the strength of randomness for the given problem is checked by comparing results achieved using random node selection with the results obtained without it.

### 1.1 C-fuzzy decision trees

The new class of decision trees, called C-fuzzy decision trees, is proposed by Pedrycz and Sosnowski (2005). The motiva-

Communicated by V. Loia.

✉ Zenon A. Sosnowski  
z.sosnowski@pb.edu.pl

Łukasz Gadomer  
l.gadomer@pb.edu.pl

<sup>1</sup> Faculty of Computer Science, Bialystok University of Technology, Bialystok, Poland

tion to create this tree was to propose the classifier which is able to deal with the main problems of traditional tree. The fundamentals of decision trees are:

- The decision tree operates on a small (usually) set of discrete attributes,
- To split the node, the single attribute which brings the most information gain is chosen,
- In their traditional form, the decision tree is designed to deal with discrete class problems—the continuous problems are handled by regression trees.

These fundamentals bring the following problems:

- To handle continuous values, it is necessary to perform the discretization which can impact the overall performance of the tree.
- Information brought by the nodes which were not selected to split the node is kind of lost.

C-fuzzy decision tree was developed to deal with these problems. The idea of this kind of tree was based on the assumption of treating data as the collection of information granules, which are almost the same as fuzzy clusters. The proposed tree is spanned over these granules. C-fuzzy decision tree assumes grouping data in such granules, which are characterized by low variability (which means the similar objects get to the same cluster). These granules are the main building blocks of the tree.

The first step in the tree’s construction process is grouping the data set into  $c$  clusters in the way the similar objects are placed in the same cluster. The prototype (centroid) of each cluster is randomly selected first and then improved iteratively. Then the diversity of the each cluster is computed using the given heterogeneity criterion. The node which has the greatest value of diversity (the most heterogenous) is chosen to split from all of the tree’s nodes. Using fuzzy clustering method, the selected node is divided into  $c$  clusters. For each of the nodes created that way, the diversity is computed and again, the most heterogenous node in the whole tree is selected to split. The same process repeats until the stop criterion is achieved. It can be easily noticed that each node has 0 or  $c$  children and the tree growth can be breadth intensive or deep intensive.

In order to make this publication self-contained, we would like to describe the tree’s construction process in a formal way. Let’s do the following assumptions:

- $c$  is a number of clusters,
- $N$  is a number of training instances,
- $i = 1, 2, \dots, c$ ,
- $k = 1, 2, \dots, N$ ,
- $U = [u_{ik}]$  is a partition matrix,

- $m$  is a fuzzification factor (usually  $m = 2$ ),
- $d_{ik}$  is a distance function between the  $i$ th prototype and the  $k$ th instance,
- $f_i$  is the prototype of the cluster,
- $Z = \{x(k), y(k)\}$  is an input–output pair of data instances,
- $z_k = [x_1(k)x_2k \dots x_n(k)y(k)]^T$ .

Building clusters and grouping objects into them are based on fuzzy clustering technique called fuzzy C-means technique (FCM) Bezdek (1981). The clusters are built through a minimization of objective function  $Q$ , which assumes the format:

$$Q = \sum_{i=1}^c \sum_{k=1}^N u_{ik}^m d_{ik}^2 \tag{1}$$

Partitions  $u_{ik}$  and prototypes  $f_i$  are updated during the iterations of fuzzy C-means process according to the following expressions:

$$u_{ik} = \frac{1}{\sum_{j=1}^c \left(\frac{d_{jk}}{d_{ik}}\right)^{2/(m-1)}} \tag{2}$$

$$f_i = \frac{\sum_{k=1}^N u_{ik}^m z_k}{\sum_{k=1}^N u_{ik}^m} \tag{3}$$

To describe the node splitting criterion, let’s do the following assumptions:

- $X_i = \{x(k) | u_i(x(k)) > u_j(x(k)) \text{ for all } j \neq i\}$ , where  $j$  pertains to the nodes originating from the same parent, denotes all elements of the data set which belong to the given node in virtue of the highest membership grade,
- $Y_i = \{y(k) | x(k) \in X_i\}$  collects the output coordinates of the elements that have already been assigned to  $X_i$ ,
- $U_i = [u_i(x(1))u_i(x(2)) \dots u_i(x(Y_i))]$  is a vector of the grades of membership of the elements in  $X_i$ ,
- $N_i = \langle X_i, Y_i, U_i \rangle$ ,
- $m_i$  is the representative of this node positioned in the output space,
- $V_i$  is the variability of the data in the output space existing at the given node.

The variability  $V_i$  is computed according to the following expression:

$$V_i = \sum_{(x(k), y(k)) \in X_i \times Y_i} u_i(x(k))(y(k) - m_i)^2 \tag{4}$$

In this notation,  $m_i$  is computed the following way:

$$m_i = \frac{\sum_{(x(k),y(k)) \in X_i \times Y_i} u_i(\mathbf{x}(k))y(k)}{\sum_{(x(k),y(k)) \in X_i \times Y_i} u_i(\mathbf{x}(k))} \tag{5}$$

After the construction process is complete, the tree can be used in classification mode. The classifying object starts from the root of the tree. First, the membership degrees of this object to the children nodes are computed. Each degree is the number of the [0, 1] range, and all of these degrees sum to 1. The object is getting to the node which corresponds with the highest of the computed membership values. This operation repeats as long as object achieves the node which doesn't have any children. The result of the classification process is the class assigned to the node where the object gets to. The same process is repeated for each of the objects which have to be classified.

### 1.2 Fuzzy random forests

The solution with joins fuzzy trees and random forests was first presented by Bonissone et al. (2008b) and then widely described by Bonissone et al. (2008a, 2010). The information about constructing this classifier, learning process and data classification is similar in all of these publications—they differ only in details. The most extensively fuzzy random forest issue was presented by Bonissone et al. (2010) which achieved the biggest popularity from these three papers.

The mentioned solution was based on fuzzy decision trees (Janikow 1998) and random forests (Breiman 2001). The assumptions of fuzzy random forest were to combine:

- The robustness of ensemble classifiers,
- The power of the randomness to decrease the correlation between the trees and increase the diversity of them,
- The flexibility of fuzzy logic for dealing with imperfect data.

The forest construction process is similar to Forest-RI, described by Breiman (2001). When the forest is constructed, for each tree algorithm begins its working from the root of the tree. A random set of attributes, which has the same size for each node, is chosen. Using all of the objects from the training set, the information gain is computed for each of these attributes. The attribute with the greatest information gain is chosen to node split. After this operation the attribute is removed from the set of available attributes to divide the next nodes. Then, for all following tree nodes, the operation is repeated using the same training set and the new random attribute set with the given size (attributes used before are excluded). For each object of the training set, the given object's membership degree to the given node is computed when the node is dividing. For each node, before the division,

the membership degree is equal to 1. After the division, each object can belong to one or to the greater number of created leaves. If this object belongs to only one leaf, its membership degree is equal to 1, for the other leaves it is equal to 0. If the object belongs to more than one leaf, the membership degree can be different for different leaves—it can take values between 0 and 1 and it sums to 1 in the set of all children of the given node. If the value of the attribute chosen to perform the division is missing, the object is assigned to each split node with the same membership degree, which depends on to the number of the nodes.

All leaves in the tree are constructed according to the described algorithm. Then the same process is repeated for each tree, but with different, randomly selected set of attributes. It makes every tree in the forest different than others.

Authors improved the fuzzy random forest classifier in their later works. For example, in one of their publications (Cadenas et al. 2012) they focused on using this classifier with imperfect data. They presented that fuzzy random forest can deal with missing, crisp, probabilistic uncertainty and imprecise values. In one of their earlier papers (Garrido et al. 2010), authors proposed a classification and regression technique which handle heterogenous and imperfect information. They created a model described by Gaussian mixture to deal with this issue. The fuzzy random forest was the other idea of handling imperfect data, which shows the advantages and strength of this classifier.

### 1.3 OWA operators

OWA operators were created by R. R. Yager and presented in one of his papers (Yager 1988). The motivation for creating that structure was the lack of fuzzy operator between t-norm and co-t-norm. The t-norm operator expresses pure “anding,” and the co-t-norm operator expresses pure “oring.” The OWA operators' objective was to create the operator which would be placed between these two extreme ones, adjusting their degree in the aggregation.

Let's do the following assumptions:

- $n$  is the size of the collection,
- $W = [W_1, W_2, \dots, W_n]$  is the vector of weights, where:
  - $W_i \in [0, 1]$ ,
  - $\sum_{i=1}^n W_i = 1$ ,
- $A = [a_1, a_2, \dots, a_n]$ —collection of objects,  $a_i \in [0, 1]$ ,
- $B = [b_1, b_2, \dots, b_n]$ —vector A sorted in descending order.

According to these assumptions, OWA operator can be defined as:

$$F(a_1, a_2, \dots, a_n) = \sum_{i=1}^n W_i b_i \quad (6)$$

For any vector  $B$  and for any weights  $W$  which fulfill the given assumptions it is true that:

$$0 \leq F(B) \leq 1 \quad (7)$$

This operator has many usages in different areas, and it is still developed. As an example, (Yager and Alajlan 2014) authors propose the OWA operator with the capability to perform aggregations in a discrete set of possible outcomes or an interval of possible outcome. They also provide the OWA operator with the ability to perform aggregations in situations where the arguments being aggregated have an associated uncertainty, probability of occurrence. In the other paper (Mesiar et al. 2015), authors discuss three generalizations of OWA operators like GOWA, IOWA and OMA. They also propose new types of generalizations. The next paper (Alajlan et al. 2013) treats about using OWA operator for the classification of hyperspectral images. The example of using OWA for a web selection process is presented by Yager et al. (2011). In the last example (Yager and Beliakov 2010), the ways OWA operators can be used for regression problems are described. There are much many works treating about different OWA usages and modifications. These few examples show how widely this operator is used.

#### 1.4 Other approaches

In previous paragraphs we presented theoretical aspects of issues directly connected with our work which is a subject of this paper. However, there are many other approaches which were designed in order to face the problem of data classification. In the original fuzzy random forest authors used Janikow's fuzzy trees (Janikow 1998) and in our forest, we used C-fuzzy decision trees (Pedrycz and Sosnowski 2005), but there also many other kind of fuzzy decision trees which could be used to construct the ensemble classifier. For example, in Lertworapachaya et al. (2014) authors proposed the concept of interval-valued fuzzy decision trees. They noticed that during fuzzy decision tree construction process the uncertainty associated with their membership values is not considered. The main idea of this kind of trees is to represent fuzzy membership values as intervals in order to model the uncertainty and construct the tree using look-ahead-based fuzzy decision tree induction method. The forest also can be created in a different ways. For example, instead of creating the fuzzy random forest the way presented in Bonissone et al. (2010), the naive random forest can be created in order to increase diversity of the trees.

Although we concentrate on the scope of fuzzy clustering, fuzzy trees and random forests, there are also many others

data classification tools which can deal the same problem in a different way. One of the most popular classifiers is a neural network (Jain and Allen 1995), which bases on iterative modifications of neurons' weights during the training process. Radial basis neural network (Daqi et al. 2005) is the classifier which bases on neural network. Logistic model tree (Landwehr et al. 2005) adapts an idea of tree induction methods and linear models for classification problem. Support vector machine—SVM (Vapnik 1995) classifier maps the space of the input data into high-dimensional space of features and constructs a hyperplane which divides all of the object classes. Those classifiers are just examples which present the diversity of the solutions which can be used in order to deal with classification problems. In this work we decided to face these problems using C-fuzzy random forest approach, but it is worth to notice just the one of possible ways of resolving them.

## 2 C-fuzzy random forest

### 2.1 Notation

In order to present our classifier in a formal way, we also used the following notations [based on works performed by Bonissone et al. (2010) and Pedrycz and Sosnowski (2005)]:

- $c$  is the number of clusters,
- $T$  is the number of trees in the C-FRF ensemble,
- $t$  is the particular tree,
- $N_t$  is the number of nodes in the tree  $t$ ,
- $n$  is a particular leaf reached in a tree,
- $E$  is a training set,
- $e$  is a data instance,
- $I$  is the number of classes,
- $i$  is a particular class,
- $C\_FRF$  is a matrix with size  $(T \times MAX_{N_t})$  with  $MAX_{N_t} = \max \{N_1, N_2, \dots, N_t\}$ , where each element of the matrix is a vector of size  $I$  containing the support for every class provided by every activated leaf  $n$  on each tree  $t$ ; this matrix represents C-fuzzy forest or C-fuzzy random forest,
- $\mathbf{w} = [w_1, w_2, \dots, w_c]$  are weight's of the OWA operator
- $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_c]$  is matrix of the OWA operators of the forest,
- $\mathbf{W}_t = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_c]$  is matrix of the OWA operators of the  $t$  tree,
- $\mathbf{W}_n = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_c]$  is matrix of the OWA operators of the  $n$  node,
- $M = \bigcup_{i=1}^c M_i$  is the of subsets of training objects belonging to the children of the given node (each element of the matrix corresponds with single node's child),

- $U = [U_1, U_2, \dots, U_{|E|}]$  is the tree's partition matrix of the training objects,
- $U_i = [u_1, u_2, \dots, u_c]$  are memberships of the  $i$ th object to the  $c$  cluster,
- $B = \{B_1, B_2, \dots, B_b\}$  are the unsplit nodes,
- $V = [V_1, V_2, \dots, V_b]$  is the variability vector.

## 2.2 The idea of C-fuzzy random forest classifier

C-fuzzy random forest is the new kind of classifier which we created and we would like to present in this section. Then, in Sect. 1.3, we described the idea of using OWA operators in C-fuzzy random forest classifier. The possibility of using OWA, thanks to the additional flexibility which allows for adjusting the classifier to the given problem in much better way, should bring the classifier to the new level. We expect that using OWA should improve the classification accuracy in many cases.

The idea of this classifier was inspired by two classifiers described before: fuzzy random forest and C-fuzzy decision tree. It assumes creating the ensemble of the C-fuzzy decision trees which works similarly to fuzzy random forest. The fuzzy random forest uses the strength of randomness to improve the classification accuracy. C-fuzzy decision tree is constructed randomly by definition as its clusters' centroids (the partition matrix) are selected randomly at the beginning of the construction algorithm. These two classifiers combined together are expected to give the promising results.

The randomness in C-fuzzy random forest is ensured by the following aspects:

- Random note to split selection—taken from the Random Forests:
  - The full randomness—selecting the random node to split instead of the most heterogenous,
  - The limited randomness—selecting the set of nodes with the highest diversity, then randomly selecting one of them to perform the division. The size of the set is given and the same for each split,
- Random creation of partition matrix—taken from the C-fuzzy decision trees:
  - At first, the centroid (prototype) of the each cluster selection is fully random. The objects which belong to the parent node are divided into clusters grouped around these centroids using the shortest distance criterion. Then the prototypes and the partition matrix are being corrected as long as they achieved the stop criterion,
  - Each tree in the forest, created the way described above, can be selected from the set of created trees. To create the single tree which will be chosen to the

forest, the set of trees can be build. Each tree from such set is tested, and the best of these trees (the one which achieved the best classification accuracy for the training set) is being chosen as the part of forest. The size of the set is given and the same for each tree in the forest.

It is easy to notice that the split selection idea is inspired by the method used in fuzzy random forest. We designed our way of splitting which corresponds with the kind of the tree (C-fuzzy decision tree) which we used in the ensemble. In the fuzzy random forest, to split the node there was chosen the random attribute. The choice of the node to split was determined by the tree growth strategy. In the C-fuzzy random forest there is no attribute chosen, because the C-fuzzy decision tree works another way. For each split all of the attributes are considered together, which is the assumption and the strength of this kind of tree. The choice concerns the node which has to be split. Some nodes will not be split at all, when the stop criterion is achieved. The same idea is adapted to two kind of trees with the different constructing algorithms. Each C-fuzzy decision tree in the forest can be drastically different or almost the same, depends on the number of clusters and the stop criterion. What is more, as the prototypes of each cluster are selected randomly and then corrected iteratively, some trees can work better than others. The diversity of the trees created that way also depends on the number of iterations of the correction process. To achieve the best results, it is possible to build many trees and choose only the best of them to the forest. The diversity of these trees in the forest can be set by changing the number of iterations and the size of the set from which the best tree is chosen. All of these parameters affect the degree of randomness in the ensemble, which can be adjusted by changing them. It allows to construct the classifier the best way according to the given problem with a great flexibility.

The simplified process of C-fuzzy random forest creation is presented in Fig. 1.

## 2.3 C-fuzzy random forest learning

The C-fuzzy random forest learning process works with the similar assumptions that the process of fuzzy random forest learning, proposed by Bonissone et al. (2010). The differences are about the following aspects:

- The kind of trees used in the forest: C-fuzzy decision trees instead of Janikow's fuzzy trees,
- The way of random selection of the node to split,
- The fact of using OWA operators (see Sect. 3).

The C-fuzzy random forest without using OWA operators is build according to Algorithm 1.

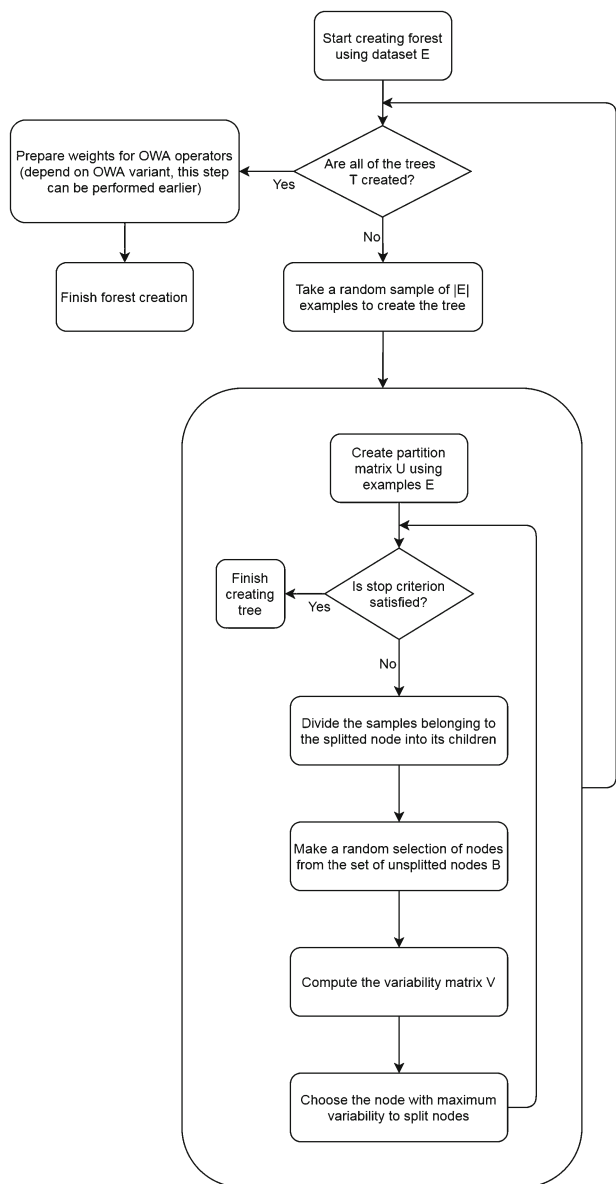


Fig. 1 The simplified process of C-fuzzy random forest creation

**Algorithm 1** C-fuzzy random forest learning

```

1: procedure C- FRFLEARNING
2:   for 1 to T do
3:     1. Take a random sample of |E| examples with replacement
       from the dataset E
4:     2. Apply Algorithm 2 to the subset of examples obtained in
       the previous step to construct C-fuzzy decision tree
5:   end for
6: end procedure
    
```

C-fuzzy decision trees in C-fuzzy random forest without using OWA operators is build according to Algorithm 2.

**Algorithm 2** C-fuzzy decision tree learning

```

1: procedure C- FDTLEARNING
2:   1. Start with the examples in E
3:   2. Create the partition matrix U randomly
4:   3. Perform FCM
5:   while Stop criterion is not satisfied do
6:     4. Divide the samples belonging to the split node into its chil-
       dren
7:     5. Make a random selection of nodes from the set of unsplit
       nodes B
8:     6. Compute the variability matrix V
9:     7. Choose the node with maximum variability to split nodes
10:    8. Perform FCM
11:  end while
12: end procedure
    
```

**2.4 C-fuzzy random forest classification**

The constructed C-fuzzy random forest can be used in classification process. The strategy used in our classifier assumes making decision by forest after the making decisions by each tree. There is also another decision-making strategy described by Bonissone et al. (2010) which assumes making the single decision by the forest without the individual decision of the trees, but we didn't use this strategy in our solution. In the variant without OWA operators it is performed as it is presented in Algorithm 3. It can be described by the following equation, based on the one presented by Bonissone et al. (2010):

$$D\_FRF(t, i, C\_FRF) = \begin{cases} 1 & \text{if } i = \arg \max_{j, j=1,2,\dots,I} \left\{ \sum_{n=1}^{N_t} C\_FRF_{t,n,j} \right\} \\ 0 & \text{otherwise} \end{cases}$$

**Algorithm 3** C-fuzzy random forest classification

```

1: procedure C- FRFCCLASSIFICATION
2:   DecisionOfTrees
3:   DecisionOfForest
4: end procedure
5: procedure DECISIONOFTREES
6:   for t = 1 to T do
7:     1. Run the example e to obtain the tree's partition matrix Ui
8:     2. Choose the class c where c = arg max_{i,i=1,2,\dots,I}
       D\_FRFt,i,C\_FRF
9:   end for
10: end procedure
11: procedure DECISIONOFFOREST
12:   Assign to class according to the simple majority vote of trees
       decisions
13: end procedure
    
```

### 3 Aggregating the knowledge contained in trees in decision making with OWA operators

The decision about belonging the object to the given node is based on membership degrees. It means that each object belongs to all of possible nodes with the different degrees. Sometimes it belongs to the given node with much higher degree than to the other nodes, but there are often situations the memberships are similar. In such situations low differences decide about the node where the object gets in. It can be supposed that the knowledge stored by the tree (or by the forest—all trees) could also have an influence on the decision, especially in the described situations. For this reason OWA operators were used for the classification and decision-making process.

The OWA operators were used in the three different ways:

- Local OWA—computed for each node of the each tree in the forest,
- Global OWA:
  - Global OWA for each tree in the forest—computed once for each tree in the forest,
  - Global OWA for the whole forest—computed once for the whole forest.

All of these ways are described in the following paragraphs. For all of them there was decided to use OWA operators to weigh the choice of the memberships during the node selection process. Instead of selecting the maximum membership, the result of the multiplication of the membership as the weight was chosen. As it can be easily noticed, for all equal weights the choice is not modified at all. The weights' influence grows with the increasing differences between different memberships' weights. The way of computing weights is described in the following paragraphs.

#### 3.1 Obtaining OWA operators weights using genetic algorithm

The basic problem connected with OWA operator is choosing the weights properly. The choice of weights should fit to the area where the operator is used. There are at least a few methods of obtaining OWA operators' weights. An example method of using the training data to pick the weights is described by Filev and Yager (1994). However, according to our research, the best suitable method of obtaining OWA operators' weights for our objectives appeared to be a genetic algorithm. This method is accurate and flexible—it allows for adjusting algorithm parameters according to

the given problem and to the circumstances. By modifying these parameters, the compromise between the computation time and the quality of the results can be achieved. In our research we decided to choose algorithm parameters the way to achieve best possible results for the price of computation time. To achieve such parameters, we used optimization process to recognize the border values—we were modifying the parameters as long as the further changes were not improving the results in a significant degree any longer. We used the same set of parameters for each dataset; however, they can be also selected for each set differently. It is possible that for specific datasets choosing different parameters could allow to achieve a bit shorter computation time with comparable results. In the datasets we used we haven't noticed significant benefits of optimizing parameters between them.

Optimal weights of OWA operator for the  $n$  node and  $c$  cluster are computed by maximizing the function:

$$F(W_{n_c}) = \sum_{j=1}^c \sum_{i=1}^{|M_j|} U_{ij} W_{n_{cj}} \tag{8}$$

The function is computed for each OWA operator's weights in the node. Its maximization is performed by a genetic algorithm. Let's do the following assumptions:

- $r$  is the crossover probability,
- $s$  is the mutation probability
- $p \in [0, 1]$  is randomly generated number which decides about crossover,
- $q \in [0, 1]$  is randomly generated number which decides about mutation,
- $k$  is randomly generated crossover point,
- $i$  is the number of epoches.

Crossover operates on the randomly chosen pair of vectors of weights. For such pair there is randomly generated the number  $p \in [0, 1]$ . If  $p \leq r$ , the split point  $k$  is chosen randomly. At this point  $k$  the vectors of weights are crossed and they are both normalized to 1. Mutation works on a single weights. For each weight in each vector there is randomly generated the number  $q \in [0, 1]$ . If  $q \leq s$ , the weight  $l \in [0, 1]$  is generated randomly. It replaces the mutated weight and the whole vector of weights is normalized to 1. If the fitness function after doing these operations is higher than before them, new objects replaces the old ones. This process is repeated for  $i$  epoches.

The partition matrix is passed to the algorithm, the seed of random weights is being sawn, and the algorithm starts. After it finishes, it returns the weights' set which was corresponding with the maximum function's value.

### 3.2 Local OWA

The first approach assumes using OWA in each node of the tree where any decision is made. As it was described in a previous paragraph, when the object is classified, a series of decision about assigning the object to one of the nodes is being made. The concept of local OWA assumes that every single decision is weighted by the OWA operator.

In order to meet this assumption to each node, there is assigned the set of OWA operators. The size of this set equals to the number of the node's children. For each of the children there is computed the different OWA operator. The size of the operator also equals to the number of the node's children. It is caused by the fact that it is impossible to assign the one proper set of weights for all memberships. In the most common situation one membership is much greater than the others—the only way to include this fact in the OWA operator is creating the different OWA operator for each membership.

#### 3.2.1 Learning

In local OWA approach weights of the OWA operators are computed during the tree expansion process. For the given node the computations take place after finding children of this node. The information about membership degrees of the each object from the training set to this node's children is given and stored. This information is used to compute the OWA operators weights. To compute the weights of the OWA operator corresponding with the given child, there are used training objects which have the highest membership degree corresponding with this node.

The C-fuzzy random forest using local OWA operators is created using Algorithm 4.

---

#### Algorithm 4 C-fuzzy random forest with local OWA learning

---

```

1: procedure C- FRFLOCALOWALEARNING
2:   for 1 to T do
3:     1. Take a random sample of  $|E|$  examples with replacement
       from the dataset  $E$ 
4:     2. Apply Algorithm 5 to the subset of examples obtained in
       the previous step to construct C-fuzzy decision tree
5:   end for
6: end procedure

```

---

Each tree in C-fuzzy random forest using local OWA operators is created using Algorithm 5.

#### 3.2.2 Classification

After all of the trees in the forest are constructed and all of the OWA operator's weights in each tree are computed, the forest can be used in the classification process. During this process, instead of choosing the maximum membership:

---

#### Algorithm 5 C-fuzzy decision tree with local OWA learning

---

```

1: procedure C- FDTLOCALOWALEARNING
2:   1. Start with the examples in  $E$ 
3:   2. Create the partition matrix  $U$  randomly
4:   3. Perform FCM
5:   while Stop criterion is not satisfied do
6:     4. Divide the samples belonging to the split node into its chil-
       dren
7:     5. Make a random selection of nodes from the set of unsplit
       nodes  $B$ 
8:     6. Compute the variability matrix  $V$ 
9:     7. Choose the node with maximum variability to split nodes
10:    8. Perform FCM
11:  end while
12:  for  $i = 1$  to  $N_t$  do
13:    9. Compute Local OWA operator for  $i$ th node
14:  end for
15: end procedure

```

---

$$U_{\max} = \max_{i \in \{1, 2, \dots, |M|\}, j \in \{1, 2, \dots, c\}} U_{ij} \quad (9)$$

there is chosen a maximum result of the multiplication of the membership degree and the corresponding weight get from the OWA operator for the  $n$  node:

$$s = \arg \max_{i \in \{1, 2, \dots, |M|\}, j \in \{1, 2, \dots, c\}} j U_{ij} \quad (10)$$

$$U_{\max} = \max_{i \in \{1, 2, \dots, |M|\}, j \in \{1, 2, \dots, c\}} U_{ij} \times W_{sj} \quad (11)$$

After the classification, the decision is made by the forest using majority vote. The classification is performed according to Algorithm 6.

---

#### Algorithm 6 C-fuzzy random forest with local OWA classification

---

```

1: procedure C- FRFLOCALOWACLASSIFICATION
2:   DecisionOfTrees
3:   DecisionOfForest
4: end procedure
5: procedure DECISIONOFTREES
6:   for 1 to T do
7:     1. Run the example  $e$  to obtain the tree's partition matrix  $U_i$ 
8:     2. Choose the leaf with the highest membership multiplied by
       Local OWA operator
9:   end for
10: end procedure
11: procedure DECISIONOFFOREST
12:   Assign to class according to the simple majority vote of trees
       decisions
13: end procedure

```

---

### 3.3 Global OWA

The local OWA concept was about using the oriented weight aggregation for each node of the tree. The global OWA



concentrates on using OWA operators in the process of the decision making by the tree or by the whole forest. As in the C-fuzzy decision trees each object can belong to more than one node with the different degrees, using different weights can result with the different final decisions.

### 3.3.1 Global OWA for each tree in the forest

The first variant of the global OWA approach is using the OWA operators in the decision making for each tree separately.

**Learning** The forest is constructed and learned in the way presented before. After the tree construction process is completed for each tree the set of weights is computed separately. To compute them the same rules are used as described in local OWA method. Training objects are grouped into  $c$  clusters (they are chosen by the maximum membership degree to the given cluster). These groups of objects are used to computed the  $c$  sets of OWA operators' weights.

Optimal weights of OWA operator for the  $t$  tree and the  $c$  cluster are computed by maximizing the function:

$$F(W_{t_c}) = \sum_{j=1}^c \sum_{i=1}^{|E|} U_{ij} W_{t_{cj}} \tag{12}$$

The C-fuzzy random forest using global OWA operators for each tree in the forest is created using Algorithm 7.

---

**Algorithm 7** C-fuzzy random forest with global OWA for each tree learning

---

```

1: procedure C- FRFGLOBALTREEOWALEARNING
2:   for 1 to T do
3:     1. Take a random sample of  $|E|$  examples with replacement from the dataset  $E$ 
4:     2. Apply Algorithm 8 to the subset of examples obtained in the previous step to construct C-fuzzy decision tree
5:   end for
6: end procedure

```

---

Each tree in C-fuzzy random forest using local OWA operators is created using Algorithm 8.

**Classification** After the forest is constructed and OWA operators are prepared, the decision-making process can be performed. In this process, for each testing object, instead of choosing the maximum membership degree to one of the nodes, as it would be done in majority vote:

$$U_{i_{\max}} = \max_{j \in \{1, 2, \dots, c\}} U_{ij} \tag{13}$$

---

**Algorithm 8** C-fuzzy decision tree with global OWA for each tree learning

---

```

1: procedure C- FDTGLOBALTREEOWALEARNING
2:   1. Start with the examples in  $E$ 
3:   2. Create the partition matrix  $U$  randomly
4:   3. Perform FCM
5:   while Stop criterion is not satisfied do
6:     4. Divide the samples belonging to the split node into its children
7:     5. Make a random selection of nodes from the set of unsplit nodes  $B$ 
8:     6. Compute the variability matrix  $V$ 
9:     7. Choose the node with maximum variability to split nodes
10:    8. Perform FCM
11:   end while
12:   9. Compute Global OWA operator for the given tree
13: end procedure

```

---

there is being chosen the maximum result of the multiplication of the membership degree and the corresponding weight get from the tree's  $t$  OWA operator:

$$s = \arg \max_{j \in \{1, 2, \dots, c\}} j U_{ij} \tag{14}$$

$$U_{i_{\max}} = \max_{j \in \{1, 2, \dots, c\}} U_{ij} \times W_{t_{sj}} \tag{15}$$

The classification is performed according to Algorithm 9.

---

**Algorithm 9** C-fuzzy random forest with global OWA for each tree classification

---

```

1: procedure C- FRFGLOBALTREEOWACLASSIFICATION
2:   DecisionOfTrees
3:   DecisionOfForest
4: end procedure
5: procedure DECISIONOFTREES
6:   for 1 to T do
7:     1. Run the example  $e$  to obtain the tree's partition matrix  $U_i$ 
8:     2. Choose the leaf with the highest membership multiplied by Global OWA for the given tree operator
9:   end for
10: end procedure
11: procedure DECISIONOFFOREST
12:   Assign to class according to the simple majority vote of trees decisions
13: end procedure

```

---

### 3.3.2 Global OWA for the whole forest

The second type of the global OWA is using the OWA operators in the decision making for all of the trees of the forest together.

**Learning** The forest is constructed and learned in the way presented before. When all of the trees are constructed, the one set of OWA operator's weights is computed. It is performed using all of the objects from the training set divided

into  $c$  subsets without dividing them into trees. The object's division criterion is analogous to the previously described global OWA for each tree method.

Optimal weights of OWA operator for the  $c$  cluster are computed by maximizing the function:

$$F(W_c) = \sum_{j=1}^c \sum_{l=1}^T \sum_{i=1}^{|E|} U_{lij} W_{cj} \tag{16}$$

where  $U_t$  is the  $t$  tree's partition matrix of the training objects.

The C-fuzzy random forest using global OWA operators for the whole forest is created using Algorithm 10.

---

**Algorithm 10** C-fuzzy random forest with global OWA for the whole forest learning

---

```

1: procedure C- FRFGLOBALOWALEARNING
2:   for 1 to T do
3:     1. Take a random sample of  $|E|$  examples with replacement
       from the dataset  $E$ 
4:     2. Apply Algorithm 11 to the subset of examples obtained in
       the previous step to construct C-fuzzy decision tree
5:   end for
6:   3. Compute Global OWA operator for the whole forest
7: end procedure

```

---

Each tree in C-fuzzy random forest using local OWA operators is created using Algorithm 11.

---

**Algorithm 11** C-fuzzy decision tree with global OWA for the whole forest learning

---

```

1: procedure C- FDTGLOBALOWALEARNING
2:   1. Start with the examples in  $E$ 
3:   2. Create the partition matrix  $U$  randomly
4:   3. Perform FCM
5:   while Stop criterion is not satisfied do
6:     4. Divide the samples belonging to the split node into its children
7:     5. Make a random selection of nodes from the set of unsplit nodes  $B$ 
8:     6. Compute the variability matrix  $V$ 
9:     7. Choose the node with maximum variability to split nodes
10:    8. Perform FCM
11:   end while
12: end procedure

```

---

**Classification** When the OWA operator is ready, the decision-making process can be performed according to the rules described for the global OWA for each tree method. For each testing object, instead of choosing the maximum membership degree to one of the nodes, as it would be done in majority vote:

$$U_{i_{\max}} = \max_{j \in \{1, 2, \dots, c\}} U_{ij} \tag{17}$$

there is being chosen the maximum result of the multiplication of the membership degree and the corresponding weight get from the forest's OWA operator:

$$s = \arg \max_{j \in \{1, 2, \dots, c\}} j U_{ij} \tag{18}$$

$$U_{i_{\max}} = \max_{j \in \{1, 2, \dots, c\}} U_{ij} \times W_{sj} \tag{19}$$

The classification is performed according to Algorithm 12.

---

**Algorithm 12** C-fuzzy random forest with global OWA for the whole forest classification

---

```

1: procedure C- FRFGLOBALOWACLASSIFICATION
2:   DecisionOfTrees
3:   DecisionOfForest
4: end procedure
5: procedure DECISIONOFTREES
6:   for 1 to T do
7:     1. Run the example  $e$  to obtain the tree's partition matrix  $U_i$ 
8:     2. Choose the leaf with the highest membership multiplied by
       Global OWA for the whole forest operator
9:   end for
10: end procedure
11: procedure DECISIONOFFOREST
12:   Assign to class according to the simple majority vote of trees
       decisions
13: end procedure

```

---

**3.4 Example of using OWA operators during classification process**

To make the usage of OWA operators more clear, let's take a look at the example. Consider the forest where for each tree  $c = 4$ . When the object  $i$  is being classified in the tree  $t$ , the vector of memberships of the  $i$  object to clusters  $U_i$  is being obtained. Assume that the vector  $U_i$  is

$$U_i = [0.2, 0.1, 0.4, 0.3]$$

Let's get the matrix of the OWA operators  $W$ , which can be taken from:

- a node where the object  $i$  belongs (for Local OWA),
- a tree  $t$  (for global OWA for each tree),
- a forest (for global OWA for the whole forest).

Assume that matrix of the OWA operators  $W$  is

$$W = \begin{bmatrix} 0.38, 0.22, 0.15, 0.25 \\ 0.19, 0.32, 0.26, 0.23 \\ 0.08, 0.17, 0.29, 0.46 \\ 0.13, 0.25, 0.21, 0.41 \end{bmatrix}$$

**Table 1** Datasets with their most important characteristics

Dataset	Number of instances	Number of attributes (without class)	Number of classes	Attribute types
Hepatitis	155	19	2	Categorical, integer, real
Dermatology	366	33	2	Categorical, integer
Semeion Handwritten Digit	1593	256	2	Integer
Diabetic retinopathy	1151	20	2	Integer, real
Pima Indians diabetes	768	8	2	Integer, real
Balance scale	625	4	3	Categorical
Indian liver patient	583	10	2	Integer, real
MUSK (Version 1)	476	168	2	Integer
QSAR biodegradation	1055	41	2	Integer, real
Climate model simulation crashes	540	18	2	Real

Obtain the index corresponding with the greatest value of vector  $U_i$ . It is the index of value 0,4: counting from one this index is 3.

Obtain the vector of weights  $w$  from the matrix  $W$  from the given index 3. This vector is

$$w = [0.08, 0.17, 0.29, 0.46]$$

Obtain the vector  $U_{wi}$  by multiplying corresponding values of vector  $U_i$  and  $w$

$$U_{wi} = [0, 2 \times 0.08, 0.1 \times 0.17, 0.4 \times 0.29, 0.3 \times 0.46] \\ = [0.016, 0.017, 0.116, 0.138]$$

The greatest value of vector  $U_{wi}$  is at index 4, so tree chooses value corresponding with 4th cluster as the tree's decision. Notice that that choice is different as it would be if weights wouldn't have been taken into account. After obtaining decision of all of the trees, forest makes his decision by majority vote.

### 4 Experiments description

In order to check a quality of created solution, several experiments were performed. In this section the research methodology is described.

The experiments were performed on ten datasets from UCI machine learning repository (Lichman 2013). These datasets with their most important characteristics are presented in Table 1. In our research we used relatively balanced datasets which allowed classifier to have enough training and testing data for each decision class to be learned and evaluated properly. In this work we were not trying to deal with the problem of unbalanced data in the datasets.

The objective of the research is to check how the each aspect of created classifier influences the classification accu-

racy. As it is shown, there are tested all proposed usages of OWA operators and the usage of randomness in the forest. Also how the classification accuracy changes with the different number of clusters is checked. Results are presented in Sect. 5.

Each experiment was performed using fivefold cross-validation. Four of five parts were used to train the classifier, one to test. This operation was repeated five times; each time the other part was out of bag. Then the classification accuracy of all five out of bag parts was averaged. To perform the fivefold cross-validation, each dataset was randomly divided into five parts, having equal size (or as close to the equal as it is possible). In the each of these parts there were the same proportions of objects representing each decision class as in the whole dataset (or as close to the same as possible). There were no situations when in some of parts there were no objects representing some of decision classes. This proportional and random division was saved and used for each experiment.

For each dataset, experiments using the following classifier configuration were performed:

- C-fuzzy forest without OWA,
- C-fuzzy forest using local OWA,
- C-fuzzy forest using global OWA,
- C-fuzzy forest using global tree OWA,
- C-fuzzy random forest without OWA,
- C-fuzzy random forest using local OWA,
- C-fuzzy random forest using global OWA,
- C-fuzzy random forest using global tree OWA.

Each forest was consisting of 50 trees. We obtained the optimal number of trees during the optimization process. According to our experiments, 50 trees were enough to achieve stable and reliable results. The further increase in the number of trees was not improving the classification accuracy

significantly and was causing the proportional increasing of computation time.

All of the researches were performed on the same single personal computer with 8 core CPU, 3, 5 GHz each.

The number of clusters to which every node in C-fuzzy decision trees divide was being optimized in order to achieve the best possible results. For each dataset and each of the configurations presented before, experiments were performed with the following numbers of clusters: 2, 3, 5, 8 and 13. The results for the number of clusters which allowed to achieve the highest classification accuracy for each dataset were chosen and presented in Sect. 5.

### 5 Results

The classification accuracy part contains the following information (mentioned from top to bottom):

- Classification accuracy achieved using C4.5 rev. 8 decision tree,
- Classification accuracy achieved using a single C-fuzzy decision tree,
- Classification accuracies achieved using C-fuzzy forest without OWA and with three proposed OWA variants,
- Classification accuracies achieved using C-fuzzy random forest without OWA and with three proposed OWA variants.

The results for tested datasets with the number of clusters which allowed to achieve the highest classification accuracy are presented in Table 2. In each column the best result for the given dataset is bold.

For the better readability the following shortcuts were used: C-FF as C-fuzzy forest and C-FRF as C-fuzzy random forest.

In most cases results were the best for around 2–5 clusters. This dependence differs a bit for each dataset, which means the number of clusters should be adjusted to the given problem. For each dataset there was at least one number of clusters which in almost all of the cases allowed to achieve better result than C4.5 rev. 8 Decision Tree.

In almost all of the cases (unusual exceptions) results achieved using C-fuzzy forest and C-fuzzy random forest were better than using single C-fuzzy decision tree. It clearly shows the strength of ensemble classifier build of this kind of trees: working together trees achieve much better results than working as a single classifier.

For some datasets C-fuzzy random forests achieved better results than C-fuzzy forests, for other ones C-fuzzy forests worked better. It means the decision about using randomness should be taken according to the given problem and to the classifier’s configuration.

**Table 2** Classification accuracies (%)

Dataset	Hepatitis	Dermatology	Semeion Handwritten Digit	Diabetic retinopathy	Pima Indians diabetes	Balance scale	Indian liver patient	MUSK	QSAR biodegradation	Climate model simulation crashes
Number of clusters	13	5	3	2	5	2	2	5	8	3
C4.5 rev. 8	58.06	94.32	94.79	64.00	73.73	78.40	66.84	80.21	82.94	92.41
Single C-fuzzy tree	55.49	94.00	94.35	64.64	73.7	82.24	66.04	77.72	75.55	89.07
C-FF; no OWA	65.81	<b>97.82</b>	96.92	65.07	75.13	83.04	68.78	<b>85.93</b>	<b>85.12</b>	<b>95.75</b>
C-FF; local OWA	59.36	97.54	96.86	65.07	75.13	83.04	68.12	84.88	83.69	<b>95.75</b>
C-FF; global OWA	<b>67.10</b>	<b>97.82</b>	96.92	65.07	75.13	83.04	68.78	<b>85.93</b>	<b>85.12</b>	<b>95.75</b>
C-FF; global tree OWA	63.23	97.54	96.92	65.07	75.13	83.04	68.78	85.30	83.79	<b>95.75</b>
C-FRF; no OWA	64.52	97.27	<b>97.30</b>	65.16	<b>75.78</b>	83.36	<b>69.81</b>	84.65	82.94	94.27
C-FRF; local OWA	64.52	97.54	<b>97.30</b>	<b>65.42</b>	<b>75.78</b>	<b>83.52</b>	69.29	83.61	82.18	94.45
C-FRF; global OWA	58.71	97.27	<b>97.30</b>	65.16	<b>75.78</b>	83.36	<b>69.81</b>	84.65	82.94	94.27
C-FRF; global tree OWA	63.87	96.72	<b>97.30</b>	65.16	<b>75.78</b>	83.36	<b>69.81</b>	84.44	83.41	94.27

**Table 3** Average tree structure—Semeion Handwritten Digit

Number of clusters	C-fuzzy forest					C-fuzzy random forest				
	2	3	5	8	13	2	3	5	8	13
Average number of clusters	362	341	312	343	457	369	346	308	331	508
Average tree width	68	88	110	150	179	72	91	112	145	197
Average tree height	13	9	7	6	6	13	10	6	6	7

**Table 4** Average tree structure—Pima Indians Diabetes

Number of clusters	C-fuzzy forest					C-fuzzy random forest				
	2	3	5	8	13	2	3	5	8	13
Average number of clusters	204	113	124	128	162	205	116	122	131	160
Average tree width	37	30	61	69	115	38	31	56	72	118
Average tree height	11	7	5	4	3	11	7	5	3	4

**Table 5** Computation times—Hepatitis

Number of clusters	C-fuzzy forest					C-fuzzy random forest				
	2	3	5	8	13	2	3	5	8	13
Learning times	00:58:42	01:20:28	01:21:22	01:31:29	05:19:59	00:59:06	01:10:13	01:17:25	01:28:56	05:16:34
Classification times										
Without OWA	00:00:20	00:00:31	00:01:08	00:01:44	00:02:54	00:00:20	00:00:27	00:00:56	00:01:43	00:02:58
With local OWA	00:00:20	00:00:31	00:01:06	00:01:35	00:02:42	00:00:20	00:00:27	00:00:55	00:01:35	00:02:49
With global OWA	00:01:45	00:03:06	00:06:57	00:10:17	00:15:42	00:01:47	00:02:44	00:05:38	00:09:54	00:15:53
With global tree OWA	00:01:48	00:03:13	00:07:03	00:10:35	00:16:19	00:01:51	00:02:49	00:05:48	00:10:13	00:16:32

For each dataset the results achieved with aggregating using one of the variants of OWA operators were better or the same that without using OWA operators. There was no variant of proposed OWA operators which always worked better than others. It allows to conclude that using OWA operators can increase the classification accuracy. However, the variant of used OWA operators should be chosen according to the given problem.

Most of the described percentage differences between two compared results translates into at least a few objects in the dataset. It means described improvements of the results are important or at least noticeable.

For the most of tested datasets tree structure changes for the different number of clusters were similar. The more clusters had the tree, the wider and shorter it was. The changes of the average number of nodes according to the number of clusters depend on the dataset. All of the tree structure changes were similar for the C-fuzzy forest and C-fuzzy random forest for each dataset. The example changes of average trees structures according to the number of clusters are presented in Table 3 for Semeion Handwritten Digit dataset and in Table 4 for Pima Indians Diabetes dataset.

The experiment duration was different for each dataset, and it mainly depended on the number of instances and the

number of attributes of the dataset. The example computation times are presented for Hepatitis dataset in Table 5 (the shortest time of all tested datasets) and for MUSK (version 1) dataset in Table 6 (the longest time of all tested datasets). All times are shown in hh:mm:ss format. As it is presented, forest learning and classification times significantly increases with the increasing number of clusters. Learning times usually take between a few minutes for a small number of clusters and a few hours for a large number of clusters. Such a long learning time is caused by the fact of using genetic algorithm for each tree node in order to compute OWA operators' weights. For the larger number of clusters there are more weights to optimize, which causes longer times in comparison with the lower number of clusters. Classification times usually differs from below second to over a dozen of minutes. (For some datasets it is longer, for example for Semeion Handwritten Digit dataset.) Usually classification times for global OWA and global tree OWA variants are much longer than without OWA or with local OWA. It is caused by the necessity of performing computations of weights in global OWA variants, which is much more time-consuming than that for local OWA.

**Table 6** Computation times—MUSK (version 1)

Number of clusters	2	3	5	8	13	2	3	5	8	13
	C-fuzzy forest					C-fuzzy random forest				
Learning times	00:10:31	00:10:08	00:12:59	00:18:35	00:22:44	00:10:36	00:10:09	00:12:59	00:17:49	00:22:24
Classification times										
Without OWA	00:00:01	00:00:01	00:00:02	00:00:04	00:00:07	00:00:01	00:00:01	00:00:02	00:00:04	00:00:07
With local OWA	00:00:01	00:00:01	00:00:02	00:00:04	00:00:07	00:00:01	00:00:01	00:00:02	00:00:04	00:00:07
With global OWA	00:00:21	00:00:31	00:00:53	00:01:29	00:02:22	00:00:21	00:00:31	00:00:53	00:01:31	00:02:18
With global tree OWA	00:00:24	00:00:36	00:01:03	00:01:49	00:03:03	00:00:24	00:00:37	00:01:04	00:01:51	00:02:58

## 6 Summary and conclusions

The idea of aggregating the knowledge contained in C-fuzzy decision tree nodes with OWA operators during the C-fuzzy random forest decision-making process was described in the previous paragraphs. Different ways of knowledge aggregation using OWA operators were presented in this ensemble classifier: local OWA, global OWA for each tree in the forest and global OWA for the whole forest. The created solution was tested on ten datasets. Results were compared with C4.5 rev. 8 decision tree and single C-fuzzy decision tree. The results achieved with each variant of aggregating using OWA operators were compared with another ones (and with the accuracies achieved without using any variant of weighted aggregation). The influence of the randomness on the classification accuracy was also tested.

Achieved results show that using OWA operators can increase the classification quality for some kind of problems and datasets. In most cases C-fuzzy forest and C-fuzzy random forest classifier give better results than C4.5 rev. 8 decision tree and single C-fuzzy decision tree classifiers. It also appears that using randomness in the created ensemble classifier can increase the classification accuracy.

Performed experiments showed that created ensemble classifier can be competitive in comparison with the other classifiers. One of the main advantages of this classifier is the flexibility. There are many parameters which can be adjusted to fit the given problem in the best possible way. In order to configure the classifier to deal with the given dataset as well as it is possible, the parameters of the C-fuzzy decision trees, the forest cardinality and the strength of the randomness and the OWA variant, together with the parameters of genetic algorithm which allows to obtain OWA operators weights, can be modified. Although it takes some time to optimize all of the parameters, this flexibility causes that the created classifier stands out from the concurrent ones.

**Acknowledgements** This work was supported by the grant S/WI/1/2013 from Białystok University of Technology founded by Ministry of Science and Higher Education.

## Compliance with ethical standards

**Conflict of interest** Łukasz Gadomer declares that he has no conflict of interest. Zenon A. Sosnowski declares that he has no conflict of interest.

**Ethical approval** This article does not contain any studies with human participants or animals performed by any of the authors.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

## References

- Alajlan N, Bazi Y, Alhichri H, Melgani F, Yager R (2013) Using OWA fusion operators for the classification of hyperspectral images. *IEEE J Sel Top Appl Earth Obs Remote Sens* 6(2):602–614
- Bezdek JC (1981) Pattern recognition with fuzzy objective function algorithms. Kluwer Academic Publishers, Norwell
- Bonissone P, Cadenas J, Garrido M, Diaz-Valladares R (2008a) Combination methods in a fuzzy random forest. In: *IEEE international conference on systems, man and cybernetics, 2008. SMC 2008*, pp 1794–1799
- Bonissone P, Cadenas JM, Garrido MC, Diaz-Valladares RA (2010) A fuzzy random forest. *Int J Approx Reason* 51(7):729–747
- Bonissone PP, Cadenas JM, Garrido MC, Diaz-valladares RA (2008b) A fuzzy random forest: fundamental for design and construction. In: *In Proceedings of the 12th international conference on information processing and management of uncertainty in knowledge-based systems (IPMU'08)*, pp 1231–1238
- Breiman L (2001) Random forests. *Mach Learn* 45(1):5–32
- Cadenas JM, Garrido MC, Martínez R, Bonissone PP (2012) Extending information processing in a fuzzy random forest ensemble. *Soft Comput* 16(5):845–861
- Filev D, Yager R (1994) Learning OWA operator weights from data. In: *IEEE World Congress on Computational Intelligence, Proceedings of the third IEEE conference on fuzzy systems, 1994*, vol 1, pp 468–473
- Daqi G, Chen M, Li Y (2005) A single-layer radial basis function network classifier and its applications. In: *2005 IEEE international joint conference on neural networks, 2005. IJCNN '05. Proceedings, vol 2*, pp 1045–1050
- Gadomer L, Sosnowski Z (2016) Fuzzy random forest with C-fuzzy decision trees. In: *Computer information systems and industrial management 2016, LNCS 9842*, pp 481–492

- Garrido MC, Cadenas JM, Bonissone PP (2010) A classification and regression technique to handle heterogeneous and imperfect information. *Soft Comput* 14(11):1165–1185
- Janikow C (1998) Fuzzy decision trees: issues and methods. *IEEE Trans Syst Man Cybern B Cybern* 28(1):1–14
- Jain LC, Allen GN (1995) Introduction to artificial neural networks. In: *Electronic technology directions to the year 2000, 1995. Proceedings*, pp 36–62
- Landwehr N, Hall M, Frank E (2005) Logistic model trees. *Mach Learn* 59(1–2):161–205
- Lertworapachaya Y, Yang Y, John R (2014) Interval-valued fuzzy decision trees with optimal neighbourhood perimeter. *Appl Soft Comput* 24:851–866
- Lichman M (2013) UCI machine learning repository. <http://archive.ics.uci.edu/ml>
- Mesiar R, Stupnanova A, Yager R (2015) Generalizations of OWA operators. *IEEE Trans Fuzzy Syst* PP(99):1–1
- Pedrycz W, Sosnowski Z (2005) C-fuzzy decision trees. *IEEE Trans Syst Man Cybern C Appl Rev* 35(4):498–511
- Vapnik VN (1995) *The nature of statistical learning theory*. Springer, New York
- Yager R (1988) On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Trans Syst Man Cybern* 18(1):183–190
- Yager R, Alajlan N (2014) Probabilistically weighted OWA aggregation. *IEEE Trans Fuzzy Syst* 22(1):46–56
- Yager R, Beliakov G (2010) OWA operators in regression problems. *IEEE Trans Fuzzy Syst* 18(1):106–113
- Yager R, Reformat M, Gumrah G (2011) Fuzziness, OWA and linguistic quantifiers for web selection processes. In: *2011 IEEE international conference on fuzzy systems (FUZZ)*, pp 1751–1758