Pruning trees in C-fuzzy random forest

Łukasz Gadomer¹ · Zenon A. Sosnowski¹

Published online: 28 October 2020
© The Author(s) 2020

Abstract

Pruning decision trees is the way to decrease their size in order to reduce classification time and improve (or at least maintain) classification accuracy. In this paper, the idea of applying different pruning methods to C-fuzzy decision trees and Cluster–context fuzzy decision trees in C-fuzzy random forest is presented. C-fuzzy random forest is a classifier which we created and we are improving. This solution is based on fuzzy random forest and uses C-fuzzy decision trees or Cluster–context fuzzy decision trees—depending on the variant. Five pruning methods were adjusted to mentioned kind of trees and examined: Reduced Error Pruning (REP), Pessimistic Error Pruning (PEP), Minimum Error Pruning (MEP), Critical Value Pruning (CVP) and Cost-Complexity Pruning. C-fuzzy random forests with unpruned trees and trees constructed using each of these pruning methods were created. The evaluation of created forests was performed on eleven discrete decision class datasets (forest with C-fuzzy decision trees) and two continuous decision class datasets (forest with Cluster–context fuzzy decision trees). The experiments on eleven different discrete decision class datasets and two continuous decision class datasets were performed to evaluate five implemented pruning methods. Our experiments show that pruning trees in C-fuzzy random forest in general reduce computation time and improve classification accuracy. Generalizing, the best classification accuracy improvement was achieved using CVP for discrete decision class problems and REP for continuous decision class datasets, but for each dataset different pruning methods work well. The method which pruned trees the most was PEP and the fastest one was MEP. However, there is no pruning method which fits the best for all datasets—the pruning method should be chosen individually according to the given problem. There are also situations where it is better to remain trees unpruned.

Keywords C-fuzzy random forest · Pruning · C-fuzzy decision tree · Cluster–context fuzzy decision tree

1 Introduction

Decision trees created during learning process are often overgrown. They meet the problem of overfitting to the training dataset—some nodes are created to fit the single objects of this dataset and they seem to be redundant. The existence of such nodes distort the power of generalization which should characterize the classifier. It also extends a classification time, which is especially important in large trees.

While working with our classifier—C-fuzzy random forest—we met the same problem. We noticed that C-fuzzy

decision trees and Cluster–context fuzzy decision trees which are the part of created forest are too large and overfitted to training dataset. The objective of this work is to reduce trees' size using several popular and widely known pruning methods, adapted to our classifier, and check the way pruning influences the classification accuracy and computation time. We expect to improve classification quality (or at least maintain the original one) and reduce classification time. We would also like to check which of the popular pruning methods works with our classifier the best way. Pruning decision trees in the created solution should increase its quality and make it more competitive in comparison with other classifiers.

In the literature, there are not papers which mentioned about pruning C-fuzzy decision trees nor Cluster–context fuzzy decision trees, no matter whether they are a part of the forest or not. This issue is novel—pruning such trees in C-fuzzy random forest is described in this paper for the first time. What is more, there are not many papers which

Communicated by V. Loia.

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

Łukasz Gadomer
l.gadomer@pb.edu.pl

¹ Faculty of Computer Science, Białystok University of Technology, Białystok, Poland

treat about pruning fuzzy trees, especially with different post-pruning methods (which is the main subject of this paper); however, this aspect is mentioned by some authors. The closest to our research in this aspect is Naseer et al. (2015). The authors applied the same post-pruning methods which we used in this paper to C4.5 and fuzzy C4.5 tree. They tested these methods on one dataset only—Statlog (Heart Dataset). Similarly to our research, the smallest tree was produced with Pessimistic Error Pruning method. In the mentioned experiment, Reduced Error Pruning achieved the best results and Pessimistic Error Pruning the worst ones—other pruning methods achieved satisfactory performance. These conclusions are quite similar to our ones; however, it is hard to compare results presented there to our ones because of the fact that only one dataset was evaluated. Our research proves that for each dataset the other pruning method can be optimal. The authors of Ribeiro et al. (2013) also evaluated the influence of pruning on C4.5 decision trees and fuzzy decision trees, but they rather concentrated on comparison between no pruning, post-pruning and pre-pruning. They evaluate only one post-pruning method for fuzzy decision tree—the same which is used in C4.5 decision tree. Experiments performed in Ribeiro et al. (2013) show that post-pruning methods for different datasets work better than pre-pruning ones or variant without pruning. These results are hard to compare with our ones as we adopted a different approach. There are also some papers where a comparative analysis of methods for pruning decision trees is performed, like Esposito et al. (1997); however, they concentrate on traditional, not fuzzy, decision trees, which makes research performed there hard to compare with our ones.

In Sect. 2, the idea of C-fuzzy decision trees is presented. Cluster–context fuzzy decision trees—the kind of trees based on C-fuzzy decision trees which use Cluster–context fuzzy clustering—are presented in Sect. 3. The idea of an ensemble classifier which uses C-fuzzy decision trees or Cluster–context fuzzy decision trees, called C-fuzzy random forest, is described in Sect. 4. The main topic of this paper—pruning trees in C-fuzzy random forest—is presented in Sect. 5. In this section, all of the pruning methods used in C-fuzzy random forest are described and the way of using them in the created classifier is explained. In Sect. 6, performed experiments which check the influence of pruning trees in C-fuzzy random forest on the prediction quality are described and achieved results are presented.

1.1 Notation

In this paper, to present the classifier in a formal way, we also used the following notations:

- N is the number of objects in the training set,
- n is the number of attributes in the dataset,

- $X = \{x_1, x_2, \dots, x_N\}$ is a training set,
- x is a data instance,
- y is the decision attribute,
- p is a particular node in a tree,
- c is the number of clusters,
- d is the given cluster,
- K is the number of contexts,
- A is the given context,
- $f_k = A(x_k)$ is the level of involvement of x_k in the considered context,
- S is the number of classes,
- s is a particular class,
- T is the number of trees in the C-fuzzy random forest ensemble,
- T_{size} is the number of trees from which the best is being chosen during C-fuzzy random forest creation process,
- t is the particular tree,
- P_t is the number of nodes in the tree t ,
- C_FRF is a matrix with size $(T \times \text{MAX}_{P_t})$ with $\text{MAX}_{P_t} = \max \{P_1, P_2, \dots, P_t\}$; this matrix represents C-fuzzy random forest classifier,
- $M = \bigcup_{i=1}^c M_i$ is the sum of subsets of training objects belonging to the children of the given node,
- m_p is a decision class (classification) or an average value of decision attribute (regression) corresponding with node p ,
- $U = [U_1, U_2, \dots, U_N]$ is the tree's partition matrix of the training objects,
- $U_k = [u_{1k}, u_{2k}, \dots, u_{ck}]$ are memberships of the i th object to the given cluster,
- $B = \{B_1, B_2, \dots, B_b\}$ are the unsplit nodes,
- B_{size} is the number of unsplit nodes,
- $V = [V_1, V_2, \dots, V_b]$ is the variability vector (Pedrycz and Sosnowski 2005),
- T_p is the subtree with the root p before pruning,
- m is the number of subnodes of the subtree with the root p ,
- l is the given leaf,
- G_p is the number of training set examples at node p ,
- gb_p is the number of misclassified training set examples at node p before pruning,
- g_p is the number of misclassified training set examples at node p after pruning,
- g'_p is the number of misclassified examples after pruning the subnodes of p corrected by constant value 0,5 (Quinlan 1987),
- g_l is the number of misclassified training set examples by leaf l ,
- $\sum g_l$ is the number of misclassified training set examples by subtree with the root p before pruning,

- g'_{T_p} is the number of misclassified examples achieved by subtree with the root p before pruning corrected by value specified by the number of subnodes of p ,
- $SE(g'_{T_p})$ is the standard error for the number of misclassification achieved by subtree with the root p before pruning (Quinlan 1987),
- $s_{\max}$ is the class represented by the greatest number of objects among all classes in node p ,
- $G_{ps_{\max}}$ is the number of training examples assigned to class $s_{\max}$ in node p ,
- $Err(p)$ is expected error rate of pruning in node p (Niblett and Bratko 1987),
- $WErr(p)$ is the expected weighted error rate of pruning in node p (Niblett and Bratko 1987),
- $V_{\min}$ is the minimum variability value—a threshold which decides about pruning the tree or not,
- $V_{\max}$ is the maximum variability value—a threshold which decides about pruning the tree or not,
- T_{prun} is the set of pruned trees—among this set the best tree is chosen as a final result of pruning,
- $|T_{\text{prun}}|$ is the number of pruned trees, among which the best one is chosen as a final result of the pruning,
- $R(p)$ is the error cost of node p after pruning the tree,
- $R(T_p)$ is the error cost of unpruned subtree T_p ,
- P_{T_p} is the number of leaves of unpruned subtree T_p ,
- α is the complexity cost—the cost of the extra leaf in tree (Breiman et al. 1984),
- T_{alpha} is the set of pruned trees among which the one with the lowest complexity cost is chosen as a result of the pruning.

2 C-fuzzy decision trees

Pedrycz and Sosnowski proposed their new classifier, C-fuzzy decision trees, in Pedrycz and Sosnowski (2005). They designed this kind of tree as a response to traditional decision trees limits. They usually operate on a relatively small set of discrete attributes. To split the node when the tree is constructed, only a single attribute which brings the most information gain is chosen. Traditional decision trees are also designed to deal with discrete class problems (regression trees can operate on continuous class problems). C-fuzzy decision trees are created the way which allows to avoid these problems. This kind of trees treat data as collection of information granules, which are almost the same as fuzzy clusters. These granules are generic building blocks of the tree—the data are grouped in such multivariable granules characterized by low variability.

The C-fuzzy decision tree construction starts from grouping the dataset into c clusters the way that the similar objects are placed into the same cluster. Each cluster is characterized by its centroid, called prototype, which at first is selected

randomly and then improved iteratively. When the grouping process is finished, the given heterogeneity criterion is used to compute the diversity of the each cluster. The computed diversity value decides whether the node is selected to split or not. To perform the split, the node corresponding with the lowest diversity value is chosen. The selected node is divided into c clusters using fuzzy clustering method described in Bezdek (1981). The same steps are repeated for each node until the algorithm achieves the given stop criterion. Each node of the tree has 0 or c children.

The C-fuzzy decision tree learning process is performed according to Algorithm 1. This version of tree learning algorithm is a modification of traditional version, presented in Pedrycz and Sosnowski (2005)—it includes randomness aspect, used in C-fuzzy random forest (described in Sect. 4).

Algorithm 1 C-fuzzy decision tree learning

Input: X —training dataset

Output: t —learned C-fuzzy decision tree

procedure C- FDTINC- FRFLEARNING

1. Start with the examples in X ,

2. Create the partition matrix U randomly

3. Perform FCM (Bezdek 1981)

while Stop criterion is not satisfied **do**

4. Divide the samples belonging to the split node into its children

5. Compute the variability matrix V_k

6. Choose B_{size} nodes with the highest variability and add them to B

7. Choose the random node from B

8. Perform FCM (Bezdek 1981)

end while

end procedure

The constructed tree can be used to classify new instances. Similarly to other kind of trees, each instance to classify starts from the root node. The membership degrees of the instance to the children of the given node are computed. These degrees can be numbers between 0 and 1 and sum up to 1 within the single node's children. The classifying instance becomes assigned to the node with the highest corresponding membership degree. The analogous operation is repeated until the object reaches the leaf (a node without children).

3 Cluster–context fuzzy decision trees

Cluster–context fuzzy decision tree is the classifier which joins context-based clustering (Pedrycz 1996) and C-fuzzy decision trees (Pedrycz and Sosnowski 2005), described in 2. This kind of trees were presented in Sosnowski (2012). The author predicted that joining these two algorithms should allow to achieve better results than C-fuzzy decision trees, especially for regression problems. Cluster–context fuzzy decision tree consists of K C-fuzzy decision trees, where K

is the number of contexts. It means that the structure called “tree” when writing about Cluster–context fuzzy decision trees refers to the group of C-fuzzy decision trees, not a single tree. It can be confusing, so it is worth remembering about it.

Before the construction of Cluster–context fuzzy decision tree, the decision attribute should be divided into contexts. The optimal number of contexts can be different for different datasets, so it should be adjusted according to the given problem. In theory, the number of contexts should respond the number of object groups in the dataset. However, experiments performed in Gadomer and Sosnowski (2017) shown that sometimes it is possible to achieve better results with breaking that assumption. The division into contexts can be performed using any membership function. Experiments described in this paper were performed using three membership functions: Gaussian, trapezoidal and triangular. The example division result using these functions into five contexts is presented in Fig. 1.

It is also possible to configure the shape of membership function to cope with the given problem in the best possible way. It can be performed using “context configuration” parameter. By default, this parameter’s value is 1; higher numbers make the context shorter, and lower numbers cause the context wider. In Fig. 1, the default context configuration value was presented. The example divisions using context configuration values 0.6 and 1.6 are presented in Fig. 2.

After preparing contexts, a Cluster–context fuzzy decision tree can be created. For each of these contexts, C-fuzzy decision tree is created using objects from the training set which meet the given context. It is performed according to Algorithm 2.

Algorithm 2 Cluster–context fuzzy decision tree learning

Input: X —training dataset

Output: t —learned Cluster–context fuzzy decision tree

procedure CC- FDTLEARNING

1. Create Cluster–context fuzzy decision tree t

for $i = 1$ to K **do**

2. Start with the examples in X_i

3. Create C-fuzzy decision tree t_i for context A_i using Algorithm

1

4. Add C-fuzzy decision tree t_i to Cluster–context fuzzy decision tree t

end for

end procedure

When the tree is created, it can be used in the classification and regression process. Each of the C-fuzzy decision trees classifies the new instance and it is assigned to class corresponding with cluster characterized by the greatest partition matrix value from all contexts and clusters (discrete problems) or the distance between the computed value and the

corresponding decision value in the node chosen by maximum partition matrix value (continuous problems).

4 C-fuzzy random forest

C-fuzzy random forest is our novel ensemble classifier. The idea of this classifier is to create the forest based on assumptions of fuzzy random forest (Bonissone et al. 2010), which is relatively new classifier and is still developed and used in different researches and approaches. In Chen et al. (2020), the authors used two-layer fuzzy multiple random forest for speech emotion recognition in human–robot interaction. In their research, the mentioned classifier has been developed to recognize emotional states using speech signal. Also version without randomness is widely used. The authors of Ozigis et al. (2020) performed detection of oil pollution impacts on vegetation with fuzzy forest and random forest methods. Another work which uses such classifier is Ciobanu et al. (2020), where the authors used it to predict recurrent major depressive disorder in the elderly cohort. These are also a few examples which show that this classifier is popular and widely used solution for different kinds of problems. A lot of popular solutions based on fuzzy trees and forests are presented in Sosnowski and Gadomer (2019).

C-fuzzy random forest uses C-fuzzy decision trees or Cluster–context fuzzy decision trees instead of fuzzy decision trees. The detailed description of C-fuzzy random forest with C-fuzzy decision trees was presented in Gadomer and Sosnowski (2016) and Gadomer and Sosnowski (2019). The other variant of C-fuzzy random forest, which uses Cluster–context decision trees, was presented in Gadomer and Sosnowski (2017).

In both mentioned variants, the randomness in the created classifier is ensured by two main aspects. The first of them refers to the assumptions of random forest. During the tree construction process, the node to split is selected randomly. This randomness can be full (selecting the random node to split instead of the most heterogeneous) or limited (selecting the set of nodes with the highest diversity and then randomly selecting one of them to split). The second one refers to the C-fuzzy decision tree partition matrix creation process. As C-fuzzy decision trees are a part of Cluster–context fuzzy decision trees, the same issue also refers to this kind of trees. Before the creation algorithm’s start, each clusters’ centroids’ (prototypes) coordinates are selected randomly. Instances which belong to the parent node are divided into clusters grouped around these prototypes using the shortest distance criterion. Then the iterative algorithm of correcting the prototypes and the partition matrix is being performed until the stop criterion is achieved. The second aspect of randomness is about selecting each tree in the forest from the set of created trees. Each tree from this set is evaluated using

Fig. 1 Example division of decision attribute into five contexts using (from left) triangular, Gaussian and trapezoidal membership functions

Fig. 2 Example division of decision attribute into four contexts using Gaussian function with context configuration value (from left) 0.6 and 1.6

training set, and the best of them is being chosen as the part of forest. The size of the set, marked as T_{size} , can be adjusted, defining the level of randomness, and during the whole forest creation algorithm, it is the same for the each tree.

The splitting method in C-fuzzy random forest is inspired by the one used in fuzzy random forest, but it refers to tree nodes instead of attributes. In fuzzy random forest, when the tree was being constructed, the random attribute was being chosen to split. In C-fuzzy random forest, the choice concerns the node to split selection. When the stop criterion is achieved, tree construction process stops and some nodes will not be split. That means that the randomness will decide which node will be split or not instead of the variability. This randomness also can be full or limited. It means each C-fuzzy decision tree or Cluster-context fuzzy decision tree in the forest can be similar or different, depending on the chosen algorithm's parameters. It means the classifier can be adjusted to the given problem in a flexible way.

Trees which take part in C-fuzzy random forest are constructed the way described in previous sections. When all of the trees are constructed, they are grouped into the forest. The C-fuzzy random forest creation is performed according to Algorithm 3.

Algorithm 3 C-fuzzy random forest learning

Input: X —training dataset

T —number of trees in the forest

T_{size} —number of trees among which the best is chosen

C_FRF —C-fuzzy random forest

algorithm stop criteria

Output: C_FRF —learned C-fuzzy random forest

procedure C- FRFWITHC- FDTLEARNING

for $t = 1$ to T **do**

for $j = 1$ to T_{size} **do**

 1. Create C-fuzzy decision tree t_j

if $j = 1$ or t_j achieved better results on dataset X than tree t

then

 2. Assign $t = t_j$

end if

end for

 3. Add tree t to C_FRF

end for

end procedure

During the new instance's classification process, each tree makes the decision about its membership. Then, according to the trees' decisions, the forest makes the final decision. The decision making process can be also performed in a weighted way. We designed our novel methods of weighting decisions using OWA operators. These methods are presented in Gadomer and Sosnowski (2019). The other way of weighting

which we proposed uses the estimated quality of trees in the forest. We presented this idea in Gadomer and Sosnowski (2017).

5 Pruning trees in C-fuzzy random forest

In this section, the methods of pruning C-fuzzy decision trees and Cluster-context fuzzy decision trees in C-fuzzy random forest are presented. There are two main groups of pruning methods: pre-pruning and post-pruning. Both of them are described in this section.

5.1 Pre-pruning

The pre-pruning idea is to stop the tree growth in a certain situations to avoid its overgrowth. This idea is also used in C-fuzzy decision trees and Cluster-context fuzzy decision trees, which take part in C-fuzzy random forest, by definition. These trees are designed the way they have several growth stop criteria:

- All nodes achieve higher heterogeneity than assumed boundary value,
- There are not enough elements in any node to perform the split. The minimal number of elements in the node which allows for the split is equal to the number of clusters,
- The structurability index achieves the lower value than assumed boundary value,
- The number of iterations (splits) achieved the boundary value.

Using this criteria means that the idea of pre-pruning is build into C-fuzzy decision trees and Cluster-context fuzzy decision trees creation algorithm.

5.2 Post-pruning

The following post-pruning methods were implemented and used in C-fuzzy random forest:

- Reduced Error Pruning (REP),
- Cost-Complexity Pruning (CCP),
- Pessimistic Error Pruning (PEP),
- Minimum Error Pruning (MEP),
- Critical Value Pruning (CVP).

All of these methods were applied to C-fuzzy random forest in both variants: with C-fuzzy decision trees and with Cluster-context fuzzy decision trees. All of these methods in the applied way can deal with discrete decision class (classification) problems. Pessimistic Error Pruning and Minimum

Error Pruning methods were not adapted for continuous decision class (regression) problems, and the rest of mentioned methods were implemented the way they can deal with such problems.

In the following sections, all of the mentioned post-pruning methods are described. For each method, the algorithm of pruning C-fuzzy decision tree is presented. As the Cluster-context fuzzy decision tree consists of multiple C-fuzzy decision trees, the algorithm of pruning this kind of tree is common for all of the pruning methods—it calls the different C-fuzzy decision tree pruning algorithm depending on the method. It is performed according to Algorithm 4.

Algorithm 4 Pruning Cluster-context fuzzy decision tree

Input: t —Cluster-context fuzzy decision tree

Output: t —pruned Cluster-context fuzzy decision tree

procedure CC-FDTPruning

for $i = 1$ to K **do**

 1. Prune tree t for context A_i with the given pruning method, passing the node p as the root node

end for

end procedure

5.2.1 Reduced Error Pruning (REP)

One of the simplest, most understandable and most popular post-pruning methods is Reduced Error Pruning (REP), proposed by Quinlan in Quinlan (1987). The idea of this method is to replace every non-leaf node (searching is performed from bottom to top) with a leaf and compare the classification accuracy. If the result achieved with replaced node is better or the same as before the replacing, the tree remains pruned and the next nodes are checked. Otherwise, if the number of misclassified training set examples at node p before pruning gb_p is smaller than the number of misclassified training set examples at node p after pruning, $g_p: gb_p < g_p$, the original subtree with the root p is restored. As a result, the smallest tree that works with the same classification accuracy or better than the original tree is achieved.

The Reduced Error Pruning of C-fuzzy decision tree in C-fuzzy random forest is performed according to Algorithm 5. The tree is searched from bottom to top.

Beyond the intelligibility and simplicity, the main advantage of this method is the linear computational complexity. As every node is visited only once, this method works relatively fast, especially in comparison with the other ones. This method requires separate pruning dataset, which can be a problem for small datasets. Despite its naivety, Reduced Error Pruning, according to different researches, can be useful pruning method and often allows to achieve better results than most complicated ones (Esposito et al. 1997).

Algorithm 5 Reduced Error Pruning of C-fuzzy decision tree**Input:** p —root node of C-fuzzy decision tree**Output:** p —root node of pruned C-fuzzy decision tree**procedure** C- FDT- REP **if** node p has children nodes **then** **for** $j = 1$ to c **do** 1. Run this algorithm (5), passing j th children of current p node as a new p node **end for** **end if** **if** node p has no parent node (is the root of the tree) or node p has been already checked **then** **return** node p **end if** 2. Assign $p =$ parent of the current node 3. Perform prediction of objects from pruning set, belonging to node p to achieve gb_p 4. Delete children nodes of node p 5. Perform prediction of objects from pruning set, belonging to node p (now this node does not have children) to achieve g_p **if** $gb_p < g_p$ **then** 6. Restore children nodes of node p **end if** 7. Run this algorithm (5), passing current p node as a new p node **return** node p **end procedure****Algorithm 6** Pessimistic Error Pruning of C-fuzzy decision tree**Input:** p —root node of C-fuzzy decision tree**Output:** p —root node of pruned C-fuzzy decision tree**procedure** C- FDT- PEP **if** node p has children nodes **then** 1. Perform prediction of objects from training set, belonging to node p for each leaf, to achieve g_l 2. Count nodes of the subtree with root p to achieve P_{T_p} 3. Count g'_{T_p} 4. Count $SE(g'_{T_p})$ 5. Delete children nodes of node p 6. Perform prediction of objects from training set, belonging to node p (now this node does not have children) to achieve g_p 7. Count g'_p **if** $g'_p \leq g'_{T_p} + SE(g'_{T_p})$ **then** 8. Restore children nodes of node p **for** $i = 1$ to c **do** 9. Run this algorithm (6), passing i th children of current p node as a new p node **end for** **end if** **end if** **return** node p **end procedure****5.2.2 Pessimistic Error Pruning (PEP)**

Another method proposed by Quinlan in Quinlan (1987) is Pessimistic Error Pruning (PEP). It bases on observation that misclassification data achieved by tree using training dataset are excessively optimistic, which leads to overgrown trees when used for pruning. To achieve the more realistic estimation of misclassification rate, Quinlan proposes to use the continuity correction for the binominal distribution. The tree's size is reduced when corrected number of misclassifications achieved by pruned tree is higher than corrected error before pruning increased by standard error.

Pruning C-fuzzy decision tree in C-fuzzy random forest with PEP method is performed according to Algorithm 6.

The tree is searched from top to bottom, so in the most pessimistic variant all of the internal nodes are checked once, which makes algorithm's computational complexity linear in the worst case. In addition, performed computations are relatively simple, which makes the algorithm fast even in comparison with other pruning methods considered as relatively quick. What is more, this method does not require pruning dataset, which is another advantage.

On the other hand, the statistical justification of this method is questionable (Mingers 1989). The constant value used to correct the misclassification is chosen heuristically and it is not certain whether it fits to the all kind of problems. However, experiments show that this post-pruning method often gives good results, which, connected with all of the undeniable advantages, makes this method an interesting pruning option.

5.2.3 Minimum Error Pruning (MEP)

Niblett and Bratko in Niblett and Bratko (1987) proposed their pruning method called Minimum Error Pruning (MEP) (later improved in Cestnik and Bratko (1991)). This pruning method works the following way. For each internal node p (searching from bottom to top), the expected error rate of pruning is computed. Then it is compared with the sum of weighted error rates of all subnodes of p in the original tree, computed the same way. (Weighting is performed using the number of objects in each of the subnodes; it can be performed searching the subtree from top to bottom.) If the weighted error rate of not pruned tree consisting of subnodes of the node p is greater or equal than error rate for node p after pruning its subtree: $WErr(p) \geq Err(p)$, the tree is pruned. (The subnodes of p are removed.)

The described way of pruning C-fuzzy decision trees in C-fuzzy random forest is performed according to Algorithm 7.

The described method does not require pruning dataset. Each node is visited once, but for each node it is required to iterate through its subtree, which makes minimum error pruning method slower than linear computational complexity methods. The degree of pruning is strongly affected by the number of classes, which can lead to unstable results. Also, the best results would be achieved, when in each class would be the similar number of instances, which happens hardly ever.

Algorithm 7 Minimum Error Pruning of C-fuzzy decision tree**Input:** p —root node of C-fuzzy decision tree**Output:** p —root node of pruned C-fuzzy decision tree**procedure** C- FDT- MEP **if** node p has children nodes **then** **for** $j = 1$ to c **do** 1. Run this algorithm (7), passing j th children of current p node as a new p node **end for** **end if** **if** node p has no parent node (is the root of the tree) or node p has been already checked **then** **return** node p **end if** 2. Assign $p =$ parent of the current node 3. Compute $Err(p)$ 4. Compute $WErr(p)$ **if** $WErr(p) \geq Err(p)$ **then** 5. Delete children nodes of node p **end if** 6. Run this algorithm (7), passing current p node as a new p node **return** node p **end procedure****5.2.4 Critical Value Pruning (CVP)**

Mingers proposed Critical Value Pruning (CVP) method in Mingers (1987). It is post-pruning method, but it is similar to pre-pruning technique. The idea of this method is to use threshold to estimate the importance of a node. If the node's critical value is below the threshold, the node is pruned. However, if the given node does not reach the critical value, but any of its subnodes reaches this value, the node is not pruned. Tree is searched from bottom to top. The value of the threshold decides about the scale of pruning. The larger the critical value is, the more nodes are pruned.

In C-fuzzy decision trees and Cluster-context fuzzy decision trees, the parameter which acts as a threshold value is the variability of the given node. In the classic version of C-fuzzy decision tree, this parameter decides about the splitting the given node or not. It seems reasonable to choose this parameter as a threshold value which decides about pruning tree in the given node or not.

Critical Value Pruning of C-fuzzy decision tree can be performed according to Algorithm 8.

In the original form, Critical Value Pruning does not require pruning set, but it can be useful to avoid potential problems. For C-fuzzy decision trees and Cluster-context decision trees in C-fuzzy random forest, pruning set is mandatory.

5.2.5 Cost-Complexity Pruning (CCP)

Cost-Complexity Pruning (CCP) is used in CART algorithm. It was proposed in Breiman et al. (1984). This method com-

Algorithm 8 Critical Value Pruning of C-fuzzy decision tree**Input:** p —root node of C-fuzzy decision tree V_{min} —minimal variability value V_{max} —maximal variability value $|T_{prun}|$ —umber of iterations (trees among which one will be chosen as pruning result)**Output:** p —root node of pruned C-fuzzy decision tree**procedure** C- FDT- CVP **for** $t = 1$ to $|T_{prun}|$ **do** 1. Compute V_i :

$$V_i = V_{min} + i \times \frac{V_{max} - V_{min}}{|T_{prun}|} \quad (1)$$

 2. Prune tree using V_i according to Algorithm 9 3. Add pruned tree to T_{prun} **end for** **for** $t = 1$ to $|T_{prun}|$ **do** 4. Perform prediction of objects from pruning set with tree t **end for** 5. Choose the best tree from T_{prun} as the final result of the pruning**end procedure****Algorithm 9** Critical Value Pruning of C-fuzzy decision tree—single iteration**Input:** p —root node of C-fuzzy decision tree V_i —critical value of node variability**Output:** p —root node of pruned C-fuzzy decision tree**procedure** C- FDT- CVP- IT **if** node p has children nodes **then** **for** $j = 1$ to c **do** 1. Run this algorithm (9), passing j th children of current p node as a new p node **end for** **end if** **if** node p has no parent node (is the root of the tree) or node p has been already checked **then** **return** node p **end if** 2. Assign $p =$ parent of the current node **if** $V_p < V_i$ **then** 3. Delete children nodes of node p **end if** 4. Run this algorithm (9), passing current p node as a new p node **return** node p **end procedure**

putes α for each internal node of the tree and prunes the node which has the lowest α . The pruned tree is saved, and the same step is repeated for the pruned tree. The algorithm works until only the root node is left. As a result, the set of pruned trees is achieved. From this set of trees, as a final tree, the tree which achieved greatest classification accuracy is chosen.

Let T_{α} be the set of pruned trees among which the one with the lowest complexity cost is chosen as a result of the pruning. Cost-Complexity Pruning of C-fuzzy decision tree can be performed according to Algorithm 10.

The main disadvantage of this method is computation time. Computations of the error cost and complexity cost

Algorithm 10 Cost-Complexity Pruning of C-fuzzy decision tree

Input: p —root node of C-fuzzy decision tree
 X —pruning dataset
Output: p —root node of pruned C-fuzzy decision tree

```

procedure C- FDT- CCP
  while  $P_t > 1$  do
    1. Run algorithm 11, passing the root of the tree as node  $p$ 
    2. Choose the best tree from  $T_{alpha}$  (with the lowest complexity cost  $\alpha$ ) and add this tree to  $T_{prun}$ 
  end while
  for  $t = 1$  to  $|T_{prun}|$  do
    3. Perform prediction of objects from pruning set with tree  $t$ 
  end for
  4. Choose the best tree from  $T_{prun}$  as the final result of the pruning
end procedure

```

Algorithm 11 Cost-Complexity Pruning of C-fuzzy decision tree—single iteration

Input: p —root node of C-fuzzy decision tree
Output: p —root node of pruned C-fuzzy decision tree

```

procedure C- FDT- CCP- IT
  if node  $p$  has children nodes then
    for  $j = 1$  to  $c$  do
      1. Run this algorithm (11), passing  $j$ th children of current  $p$  node as a new  $p$  node
    end for
  end if
  if node  $p$  has no parent node (is the root of the tree) or node  $p$  has been already checked then
    return node  $p$ 
  end if
  2. Assign  $p$  = parent of the current node
  3. Perform prediction of objects from training set, belonging to node  $p$  to achieve  $gb_p$ 
  4. Compute  $R(p)$ 
  5. Delete children nodes of node  $p$ 
  6. Perform prediction of objects from training set, belonging to node  $p$  (now this node does not have children) to achieve  $g_p$ 
  7. Compute  $R(T_p)$ 
  8. Compute  $P_{T_p}$ 
  9. Compute  $\alpha$ 
  10. Add pruned tree to  $T_{alpha}$ 
  11. Restore children nodes of node  $p$ 
  12. Run this algorithm (11), passing current  $p$  node as a new  $p$  node
  return node  $p$ 
end procedure

```

are expensive. In each algorithm step, the whole tree must be analyzed and costs for each node must be computed. What is more, in each step only nodes with the lowest α are pruned. If the original tree has great number of leaves, also the great number of iterations is performed and the great number of pruned trees is saved to choose the best one. This method also requires pruning dataset.

6 Experimental results

In order to check how the implemented post-pruning methods deal with classification and regression problems, the several experiments were performed. These experiments are described in this section, and the results are presented.

6.1 Experiment description

All of experiments presented in this section were performed on UCI machine learning datasets (Dua and Graff 2019). The objective of this research was to check which pruning method works in the best way with our ensemble classifier and to test whether randomness can improve classification quality. In order to meet these objectives, each dataset was evaluated with C-fuzzy forest and C-fuzzy random forest. For both these variants, forests without pruning and with all of the mentioned pruning types were tested. They were also compared with results achieved by single C-fuzzy decision tree and C4.5 rev. 8 decision tree as a reference classifier.

For the classification problems, the experiments were performed on datasets presented and are summarized in Table 1. Chosen datasets differ from each other with their characteristics. The smallest one has 155 instances and the largest one has 1593. The number of conditional attributes differs from 4 to 256. There are different types of attributes in different datasets, including categorical, integer and real ones. Most of datasets have 2 decision classes, and one of them has 3.

For the regression problems, the experiments were performed on datasets presented and are summarized in Table 2. Two datasets were chosen—Auto Data and Housing. Auto Data dataset has 205 instances and 25 categorical, integer and real conditional attributes. Housing data have 506 instances and 13 real and integer conditional attributes.

Selected datasets are relatively balanced. In most cases, the classifier has enough training and testing objects of each class to learn properly and be evaluated honestly. In the current research, we do not try to deal with the problem of the datasets with unbalanced data.

All of these datasets were used in our previous studies to test the other aspects of the created classifier, so the current results can be compared with them. The results of our previous experiments are available in Gadomer and Sosnowski (2017) and Gadomer and Sosnowski (2019). When performing the comparison, it is important to notice that all results (especially for the forest without modification) were computed once more and with different training sets (with the same divisions, but using three instead of four parts, as the fourth part of each fold were used as pruning dataset, which is explained further). For that reason, the results for the same structures performed before and in the current experiment may differ.

Table 1 Discrete decision attribute datasets with 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
Ionosphere	351	34	2	Real

Table 2 Continuous decision attribute datasets with most important characteristics

Dataset	Number of instances	Number of attributes (without decision attribute)	Attribute types
Auto Data	205	25	Categorical, integer, real
Housing	506	13	Integer, real

Each dataset was divided into five parts. The division was performed in the way which ensures that each part has equal size (or similar, when equality is not possible). What is more, in each part there is similar number of objects representing each class. (It concerns discrete decision class datasets.) There were no situations that in some parts objects representing some decision class were missing. This division was saved and used for each experiment. Additionally, the same divisions were used in previous research.

Prepared divisions were used to perform fivefold cross-validation. To perform experiments, we needed a pruning dataset. As we decided to use the same division as in previous experiments, the following proportions were used in fivefold crossvalidation:

- Three parts were used as training set,
- One part was used as testing set,
- One part was used as pruning set.

This proportion is different than in our previous experiments, where four parts were used as training set and one was as testing set. It also causes the noticeable differences between the current and previous results for standard forest variants.

For the i th iteration of the crossvalidation, where $0 \leq i \leq 4$, parts $i \bmod 5$, $(i + 1) \bmod 5$ and $(i + 2) \bmod 5$ were used as training set, $(i + 3) \bmod 5$ as testing set and $(i + 4) \bmod 5$ as pruning set. The results of each fold was saved, and the final results were computed by averaging the results of all five folds.

Each forest variant was consisting of 50 trees. The number of 50 trees was achieved in optimization process and we decided it is the optimal number. The learning and classification process takes longer for a forest with more trees, and such forest does not achieve significantly better results than forest with 50 trees. The forest with less trees can achieve a bit more random and unstable results.

All of the experiments were performed on the same machine which was used in experiments performed in Gadomer and Sosnowski (2019). It was a single personal computer with 8 core CPU, 3.5 GHz each.

For forest variant with C-fuzzy decision trees, the following number of clusters were evaluated: 2, 3, 5, 8 and 13. These values are taken from Fibonacci series, and they express the assumption that differences between results achieved with lower number of clusters are more significant than differences between results achieved with greater number of clusters. For this reason, there is no point in checking results achieved with all numbers of clusters incrementing by one, but it is possible to use several numbers of clusters like Fibonacci series. For all of the mentioned numbers of clusters, the experiments on all of the forest and pruning variants were performed.

The forest with Cluster-context fuzzy decision trees has more parameters which allow to adjust the classifier to the given problem in the best possible way. Checking all of these parameters' combinations would take a lot of time. For this reason, we decided to perform parameters optimization. We used only the C-fuzzy forest with Cluster-context fuzzy decision trees (without any pruning variant), consisting of

50 trees, to check the following parameters' combinations (Cartesian product of all of the mentioned):

- Numbers of clusters: 2, 3, 5, 8 and 13,
- Numbers of contexts: from 2 to 9, incremented by 1,
- Membership functions: Triangular, Trapezoidal and Gaussian,
- Context configuration parameter: from 0.6 to 1.6, incremented by 0.2.

The parameters which were used to created the forest which achieved the best results were chosen to perform the experiment. Using these parameters, forests with all pruning variants were created and evaluated.

6.2 Results achieved with C-fuzzy random forest with C-fuzzy decision trees

Table 3 presents the best results achieved for all evaluated datasets. For each dataset, only the cluster which allowed to achieve the best result in an experiment is contained. There are also presented all results other than the best, for all pruning types and without pruning, for the given cluster. As it can be observed in this table, C-fuzzy forest was allowed to achieve the best results for a given dataset four times; with C-fuzzy random forest, it was possible nine times. The mentioned results include two drafts—for two datasets, C-fuzzy forest and C-fuzzy random forest achieved the same result. It clearly shows that using randomness can often improve the classification accuracy. All the differences between classification accuracies correspond with at least a few objects, which makes them significant.

According to the results presented in Table 3, the ranking of pruning methods, ordered by the number of datasets for which the given method allowed to achieve the best result, was prepared. It looks as follows:

1. Critical Value Pruning (CVP)—4,
2. Pessimistic Error Pruning (PEP)—3,
3. Cost-Complexity Pruning (CCP)—2,
4. Minimum Error Pruning (MEP)—1,
5. Reduced Error Pruning (REP)—1.

There are also important notices about the results which have to be mentioned:

- For all of the datasets, at least one classifier's variant allowed to achieve better results than C4.5 rev. 8 decision tree and single C-fuzzy decision tree,
- For a two times, the classifier achieved better results without pruning than using any of pruning methods; it means that for a nine times pruning improved the classification accuracy,

Table 3 Classification accuracies for discrete decision class datasets (%)

Dataset	Hepatitis	Dermatology	Semeion handwritten digit	Diabetic retinopathy	Pima betes	Indians dia-	Balance scale	Indian liver patient	MUSK (version 1)	QSAR biodegradation	Climate model simulation crashes	Ionosphere
Number of clusters	2	5	2	3	5	2	2	13	5	5	5	8
C4.5 rev. 8	58.06	94.32	94.79	64	73.73	78.4	78.4	66.84	80.21	82.94	92.41	82.89
Single C-fuzzy tree	65.16	94	94.35	62.47	73.7	82.24	82.24	50.76	77.72	80.38	90.93	84.33
C-FF, no pruning	63.23	97.82	97.05	62.03	76.03	82.08	82.08	55.21	84.46	83.03	94.26	87.75
C-FF, REP pruning	67.74	97.27	95.16	62.55	75.65	87.04	87.04	69.11	80.88	82.56	91.49	86.61
C-FF, CCP pruning	63.22	97.82	96.3	62.38	76.16	84.97	84.97	69.81	80.9	83.13	91.67	87.46
C-FF, PEP pruning	70.33	97.54	90.08	57.17	74.08	87.04	87.04	69.46	64.47	72.8	91.49	86.32
C-FF, MEP pruning	65.16	96.45	96.04	63.25	75.78	84.48	84.48	54.72	82.99	83.32	94.08	87.18
C-FF, CVP pruning	65.16	97.82	97.18	62.03	75.78	86.08	86.08	68.1	83.62	82.94	93.71	87.19
C-FRF, no pruning	63.23	97.82	97.05	61.69	75.77	81.92	81.92	52.65	84.87	82.75	95.19	87.18
C-FRF, REP pruning	68.39	97.54	95.61	62.12	75.78	87.04	87.04	68.96	79.83	82.27	91.49	86.61
C-FRF, CCP pruning	61.29	97.82	96.05	62.38	76.17	84.97	84.97	69.47	82.56	83.7	91.85	86.61
C-FRF, PEP pruning	70.33	97.54	90.08	58.47	73.95	87.04	87.04	71.18	64.27	72.61	91.49	86.32
C-FRF, MEP pruning	67.1	97.27	96.3	64.38	75.91	84.8	84.8	61.91	83.2	83.22	93.89	87.17
C-FRF, CVP pruning	65.81	98.09	96.86	62.21	75.39	87.04	87.04	70.15	83.82	84.27	93.89	87.18

The highest result in a given column (highest classification / regression accuracy for a given dataset) is marked in bold

Table 4 Standard deviations for discrete decision class datasets

Dataset	Hepatitis	Dermatology	Semeion written digit	Diabeticretinopathy	Pima Indians diabetes	Balance scale	Indian liver patient	MUSK (ver- sion 1)	QSAR biodegra- dation	Climate simulation crashes	Ionosphere model
Number of clusters	2	5	2	3	5	2	13	5	5	5	8
C-FF, no pruning	4.83	2.21	0.51	3.53	3.01	2.69	8.09	0.99	3.16	0.7	4.75
C-FF, REP pruning	14.71	1.21	0.61	1.64	2.42	1.67	2.61	2.23	3.16	0.33	5.32
C-FF, CCP pruning	9.26	1.63	0.58	3.08	2.96	2.81	1.68	2.13	2	0.55	4.81
C-FF, PEP pruning	15.59	1.82	0.14	3.66	3.74	1.67	3.27	5.32	1.48	0.33	5.4
C-FF, MEP pruning	6.25	2.38	0.42	1.55	2.11	1.53	9.2	2.22	4.03	1.58	5.35
C-FF, CVP pruning	13.28	1.38	0.44	2.66	3.11	1.26	3.84	0.96	2.99	1.57	4.51
C-FRF, no pruning	3.29	1.84	0.58	1.5	2.87	1.63	8.66	1.95	2.37	0.68	4.22
C-FRF, REP pruning	13.28	1.82	0.6	1.94	2.61	1.67	2.53	1.85	2.51	0.33	3.89
C-FRF, CCP pruning	8.16	1.39	0.72	3.29	2.32	2.36	1.31	3.04	3.08	0.37	5.31
C-FRF, PEP pruning	15.59	1.34	0.14	1.87	3.73	1.67	1.08	5.88	1.73	0.33	5.4
C-FRF, MEP pruning	5.55	1.94	0.36	1.24	2.02	1.49	6.37	1.11	3.4	0.7	5.72
C-FRF, CVP pruning	12.84	1.39	0.52	1.87	2.42	1.67	0.55	1.96	2.68	1.47	4.33

- For a Balance Scale dataset, the best result was achieved in five variants: for C-fuzzy forest with REP pruning and PEP pruning and for C-fuzzy random forest with REP pruning, PEP pruning and CVP pruning,
- For a Hepatitis dataset, the best result was achieved in two variants: for both C-fuzzy forest and C-fuzzy random forest, with PEP pruning.

Standard deviations corresponding with results from Table 3 are presented in Table 4. Presented standard deviations show take different values depending on the dataset and pruning method. For some datasets, like Hepatitis and Indian Liver Patient, standard deviations are relatively high and diversified between different pruning methods. For other ones, like Semeion Handwritten Digit and Climate Model Simulation Crashes, standard deviations are low and similar, no matter which pruning method were chosen. These values should be taken into consideration when the choice of optimal pruning method is being performed—even if achieved result is relatively good, it is worth considering another pruning method if standard deviation is high.

To show the general tendencies and make conclusions from the experiments, the sample complete results for one dataset (Dermatology) are presented in Table 5. This table shows all classification accuracies achieved for this dataset for all evaluated numbers of clusters. For some numbers of clusters, the results are better than for other ones. It is caused by the fact the number of clusters should be adjusted to a given problem—for each dataset, the other number of clusters is an optimal one. In Table 3, only results achieved for an optimal number of clusters are presented; however, for each dataset, the experiments for 2, 3, 5, 8 and 13 clusters were performed to choose the optimal one. These results look analogous to the ones presented in Table 5; however, because of their capacity and low conclusion making value, they are not presented in this paper.

Table 6 presents the average number of tree nodes, average tree width and average tree height for all pruning types and experiment variants for Dermatology dataset (as a sample one—similar tables were prepared for all evaluated datasets). It can be observed that the average number of nodes in the tree also depends on the number of clusters. For smaller datasets with the relatively small number of nodes, it rises with the increasing of the number of clusters. For the bigger datasets which uses the greater number of nodes, for the small (2, 3) and large (13) number of clusters there are more nodes in the tree than for medium (5, 8) number of clusters. The tree width rises with the increasing number of clusters for most of the datasets (which is also determined by C-fuzzy decision tree assumptions). The tree height usually decreases with the increasing number of clusters. This tendency is similar for all datasets, but there are some exceptions.

Table 5 Classification accuracies for dermatology dataset (%)

Number of clusters	2	3	5	8	13
C4.5 rev. 8 (1)	94.32	94.32	94.32	94.32	94.32
Single C-fuzzy tree (2)	94.81	94.81	94	75.68	51.34
C-FF. no pruning	95.63	95.63	97.82	92.63	51.9
C-FF. REP pruning	95.9	95.09	97.27	92.63	66.09
C-FF. CCP pruning	95.63	95.36	97.82	95.08	64.48
C-FF. PEP pruning	94.26	94.53	97.54	90.45	59.84
C-FF. MEP pruning	95.9	95.63	96.45	85.24	58.17
C-FF. CVP pruning	95.36	95.63	97.82	92.63	60.91
C-FRF. no pruning	95.63	95.63	97.82	92.63	49.71
C-FRF. REP pruning	95.9	95.09	97.54	95.08	62.27
C-FRF. CCP pruning	95.63	95.36	97.82	93.71	62.85
C-FRF. PEP pruning	94.26	94.53	97.54	90.98	61.46
C-FRF. MEP pruning	95.9	95.63	97.27	87.16	58.46
C-FRF. CVP pruning	95.08	95.36	98.09	94.81	62.86

The different pruning methods reduced trees' size in a different way. Below the order of pruning methods, sorted from the one which prunes tree the most (allows to achieve

the smallest tree) to the one which prunes the least number of nodes (produces the largest tree) is presented.

1. Pessimistic Error Pruning (PEP),
2. Reduced Error Pruning (REP),
3. Minimum Error Pruning (MEP),
4. Critical Value Pruning (CVP),
5. Cost-Complexity Pruning (CCP).

This order is not always kept, but in most of the cases it looks a presented way. Pessimistic Error Pruning always produced the smallest tree and it is important to notice that this method pruned C-fuzzy trees in a drastic way (by keeping only a several nodes). Critical Value Pruning and Cost-Complexity Pruning often produced larger trees than the other three methods, which is determined by algorithm's assumptions—these two methods choose the best tree from the set of pruned trees and the other three methods try to prune the given tree as much as it is possible. Of course, if one of THE two mentioned methods decide that smaller tree is better than larger one, it can produce the smaller tree than the other pruning algorithms.

Table 6 Average number of nodes, tree widths and tree heights for dermatology dataset

Number of clusters	2	3	5	8	13	2	3	5	8	13
Classifier variant	C-fuzzy forest					C-fuzzy random forest				
<i>Without pruning</i>										
Average number of nodes	38	26	54	108	169	38	26	55	108	171
Average tree width	6	8	22	39	52	6	7	22	41	54
Average tree height	11	6	5	5	6	11	6	5	5	6
<i>With REP pruning</i>										
Average number of nodes	15	11	23	31	32	15	11	23	31	33
Average tree width	3	4	11	16	20	3	4	11	16	20
Average tree height	7	4	4	3	3	7	4	3	3	3
<i>With CCP pruning</i>										
Average number of nodes	35	23	49	85	132	36	23	48	87	125
Average tree width	5	7	20	33	42	5	7	19	34	43
Average tree height	10	6	5	5	6	10	6	4	5	5
<i>With PEP pruning</i>										
Average number of nodes	10	10	26	35	21	10	10	24	35	24
Average tree width	4	5	12	17	16	4	5	11	18	17
Average tree height	5	3	4	3	2	5	4	3	3	2
<i>With MEP pruning</i>										
Average number of nodes	10	9	17	39	86	10	9	17	39	85
Average tree width	4	4	9	17	32	4	4	9	17	31
Average tree height	5	4	3	4	5	5	4	3	4	5
<i>With CVP pruning</i>										
Average number of nodes	29	21	50	81	115	29	22	45	77	124
Average tree width	4	7	20	30	38	4	7	19	29	40
Average tree height	10	5	4	5	5	10	6	4	5	5

Table 7 contains forest learning times classification times for Dermatology dataset (similarly to a situation described for Table 6, this table is only an example—analogue tables were prepared for each dataset). It can be noticed that learning and classification times highly depend on number of clusters. This observation can be obvious—more clusters causes the greater tree which require more time required to learn or classify—but it should be taken into consideration when the choice of optimal number of clusters is being performed.

For learning and classification times presented in this table, it is a bit harder to prepare a ranking similar to the previous one. In most cases, it looks the following way (sorting from the fastest one to the slowest one):

1. Minimum Error Pruning (MEP),
2. Reduced Error Pruning (REP),
3. Pessimistic Error Pruning (PEP),
4. Critical Value Pruning (CVP),
5. Cost-Complexity Pruning (CCP).

The differences in computation times between Minimum Error Pruning, Reduced Error Pruning and Pessimistic Error pruning are often unnoticeable. These differences depend on the dataset and the number of clusters. The important observation is that these three methods work much faster than Critical Value Pruning and Cost-Complexity Pruning. The reason of these differences is the same as for the differences in pruned trees' sizes: while CVP and CCP produce the set of trees and choose the best one, MEP, REP and PEP prune the single tree. Definitely the slowest method is Cost-Complexity Pruning. For some (smaller) datasets, it is only a two–three times slower than the fastest ones, but in a most drastic examples the forest learning time using Cost-Complexity Pruning was about 250 times slower than the fastest methods. The difference between Critical Value Pruning computation time and the other methods was not so significant—it hardly ever was more than 1.5 times slower than the other methods. However, this difference is still noticeable.

The summary ranking of post-pruning methods for C-fuzzy random forest with C-fuzzy decision trees is presented in Table 8. The content of the table are the places of each method in comparison with the other ones. (The lower number means the method was better according to the criterion given in column.) Generally, the greatest classification accuracy achieved Critical Value Pruning, the worst—Reduced Error Pruning. The fastest method was Minimum Error Pruning, the slowest—Cost-Complexity Pruning. The method which reduced tree size was Pessimistic Error Pruning and the one which reduced tree size in the least significant way was Cost-Complexity Pruning. It is important to notice that these are general results—they can differ between datasets and there is no method which is always the best or the worst; it should be always chosen according to a given problem.

Table 7 Classification times for dermatology dataset

Number of clusters	C-fuzzy forest					C-fuzzy random forest				
	2	3	5	8	13	2	3	5	8	13
<i>Learning times</i>										
Without pruning	00:01:07.21	00:01:21.91	00:03:16.27	00:07:48.45	00:30:48.31	00:01:03.68	00:01:21.62	00:03:22.73	00:07:19.85	00:28:42.38
With REP pruning	00:01:08.72	00:01:26.59	00:03:17.90	00:07:59.01	00:31:10.59	00:01:04.80	00:01:23.90	00:03:09.85	00:07:29.14	00:29:59.01
With CCP pruning	00:05:32.17	00:02:50.70	00:09:11.56	00:23:36.97	01:00:44.17	00:05:20.39	00:02:37.69	00:08:51.89	00:22:47.79	00:56:59.24
With PEP pruning	00:01:09.93	00:01:27.98	00:03:25.47	00:08:13.32	00:31:40.83	00:01:07.31	00:01:23.11	00:03:09.38	00:07:35.58	00:30:01.22
With MEP pruning	00:01:05.79	00:01:24.28	00:03:19.81	00:07:55.49	00:30:11.34	00:01:03.12	00:01:22.59	00:03:03.61	00:07:32.36	00:29:16.93
With CVP pruning	00:01:30.92	00:02:00.83	00:04:37.87	00:11:07.28	00:37:41.62	00:01:27.26	00:01:56.11	00:04:29.16	00:11:19.93	00:37:24.66
<i>Classification times</i>										
Without pruning	00:00:03.08	00:00:03.75	00:00:10.26	00:00:22.91	00:00:49.67	00:00:02.88	00:00:03.63	00:00:10.02	00:00:22.03	00:00:47.59
With REP pruning	00:00:02.33	00:00:03.20	00:00:08.41	00:00:18.27	00:00:39.52	00:00:02.25	00:00:03.09	00:00:08.21	00:00:17.28	00:00:38.55
With CCP pruning	00:00:03.05	00:00:03.69	00:00:10.26	00:00:20.34	00:00:42.95	00:00:02.83	00:00:03.56	00:00:09.64	00:00:19.17	00:00:40.83
With PEP pruning	00:00:02.15	00:00:03.16	00:00:08.20	00:00:16.56	00:00:34.30	00:00:02.09	00:00:03.07	00:00:07.74	00:00:16.20	00:00:33.34
With MEP pruning	00:00:02.22	00:00:03.13	00:00:07.73	00:00:17.07	00:00:40.93	00:00:02.15	00:00:03.04	00:00:07.38	00:00:16.74	00:00:39.97
With CVP pruning	00:00:02.77	00:00:03.60	00:00:10.09	00:00:19.86	00:00:40.33	00:00:02.67	00:00:03.50	00:00:09.34	00:00:19.05	00:00:42.43

Table 8 Ranking of post-pruning methods for C-fuzzy random forest with C-fuzzy decision trees

Post-pruning method	Classification accuracy	Computation time	Tree size
REP pruning	5	2	2
CCP pruning	3	5	5
PEP pruning	2	3	1
MEP pruning	4	1	3
CVP pruning	1	4	4

6.3 Results achieved with C-fuzzy random forest with Cluster-context fuzzy decision trees

The best results achieved for tested continuous decision attribute datasets are shown in Table 9. For each dataset, there are only presented the results achieved for the best parameter set obtained during parameter optimization process. As it can be observed in this table, for 1985 Auto Imports Database, the best result was achieved using C-fuzzy random forest with REP pruning. The same pruning method allowed to achieve also the second result, but for C-fuzzy forest (without randomness). What is more, for this dataset the results computed using all of the pruning methods were better than without using any pruning method. (The exception was Critical Value Pruning for C-fuzzy random forest.)

The results achieved for Boston Housing Data dataset were a bit different. The best result was achieved for C-fuzzy random forest without any pruning method. For C-fuzzy forest, Reduced Error Pruning and Cost-Complexity Plug-in produced better results than version without pruning.

Those results show that for continuous decision attribute datasets, the significance of the improvement with achieved results depends on the given problem. For one of the tested datasets, each pruning method improved the achieved results significantly; for the other one, pruning methods were unable to improve the final results.

For both datasets, results achieved with C-fuzzy random forest were significantly better than for C-fuzzy forest. It shows the strength of the randomness for the continuous decision attribute problems.

The parameters which allowed to achieve the lowest classification errors are presented in Table 10. For both datasets, the best results were achieved using two contexts. The rest of parameters were different. For 1985 Auto Imports Database dataset, the optimal number of clusters was 8, which is much more than for Boston Housing Data (2). Also the optimal membership function and context configuration differ for each dataset—for 1985 Auto Imports Database, the optimal membership function is triangular one with context configuration 1.6, while for Boston Housing Data Gaussian function with context configuration 0.6 worked the best way.

Table 9 Classification errors for continuous decision attribute datasets

Dataset	1985 Auto Imports Database	Boston Housing Data
C-FF, no pruning	3683.41	3.79
C-FF, REP pruning	3597.67	3.71
C-FF, CCP pruning	3665.93	3.71
C-FF, CVP pruning	3662.4	4.17
C-FRF, no pruning	3611.84	3.65
C-FRF, REP pruning	3578.42	3.74
C-FRF, CCP pruning	3605.67	3.71
C-FRF, CVP pruning	3662.11	3.79

The highest result in a given column (highest classification / regression accuracy for a given dataset) is marked in bold

Table 11 presents the average number of tree nodes, average tree width and average tree height for all pruning types and experiment variants for both datasets. As only two datasets for continuous decision attribute problems were evaluated, full results tables could be presented. The table shows that for both datasets the order of pruning methods, sorted from the one which prunes tree the most to the one which prunes the least, is the following:

1. Reduced Error Pruning (REP),
2. Cost-Complexity Pruning (CCP),
3. Critical Value Pruning (CVP).

This order is similar to discrete decision class datasets, but for the tested continuous decision attribute datasets Cost-Complexity Pruning method produced smaller trees than Critical Value Pruning. (The order of these two methods was different for discrete decision attribute datasets.) As it was before, Reduced Error Pruning produced smaller trees than another methods.

Table 12 contains forest learning and classification times for both datasets. As it could be supposed according to trees sizes, general learning and classification times for Boston Housing Data are much larger than for 1985 Auto Imports Database. For both datasets, it can be observed that Cost-Complexity Pruning and Critical Value Pruning build forest slower than Reduced Error Pruning. The differences in computation times are not significant for small dataset (1985 Auto Imports Database), but it is more noticeable for larger one (Boston Housing Data). The Cost-Complexity Pruning is definitely the slowest pruning methods beyond the tested ones. The order of pruning methods, sorted by ascending computation times, is the following:

1. Reduced Error Pruning (REP),
2. Critical Value Pruning (CVP),

Table 10 Classifier's parameters for evaluated datasets

Dataset	1985 Auto Imports Database	Boston Housing Data
Number of clusters	8	3
Number of contexts	2	2
Membership function	Triangular	Gaussian
Context configuration	1.6	0.6

Table 11 Average number of nodes, tree widths and tree heights for continuous decision attribute datasets

Classifier variant	C-fuzzy forest	C-fuzzy random forest	C-fuzzy forest	C-fuzzy random forest
Dataset	1985 Auto Imports Database		Boston Housing Data	
<i>Without pruning</i>				
Average number of nodes	74	73	726	727
Average tree width	54	53	217	218
Average tree height	6	6	78	78
<i>With REP pruning</i>				
Average number of nodes	43	43	305	305
Average tree width	30	29	88	87
Average tree height	5	5	16	16
<i>With CCP pruning</i>				
Average number of nodes	52	49	418	418
Average tree width	32	35	125	125
Average tree height	6	5	17	17
<i>With CVP pruning</i>				
Average number of nodes	55	55	592	414
Average tree width	40	40	104	126
Average tree height	6	6	29	15

Table 12 Forest learning and classification times for continuous decision attribute datasets

Classifier variant	C-fuzzy forest	C-fuzzy random forest	C-fuzzy forest	C-fuzzy random forest
Dataset	1985 Auto Imports Database		Boston Housing Data	
<i>Learning times</i>				
Without pruning	00:24:38.03	00:25:55.90	01:05:01.88	01:04:56.17
With REP pruning	00:25:24.82	00:26:19.78	01:05:07.80	01:05:25.64
With CCP pruning	00:29:52.41	00:29:47.06	06:02:51.92	05:59:45.81
With CVP pruning	00:28:35.24	00:27:18.27	03:01:07.83	03:00:43.77
<i>Classification times</i>				
Without pruning	00:00:10.61	00:00:09.56	00:00:05.88	00:00:05.28
With REP pruning	00:00:08.93	00:00:09.32	00:00:04.62	00:00:05.15
With CCP pruning	00:00:09.21	00:00:09.17	00:00:05.61	00:00:06.70
With CVP pruning	00:00:09.97	00:00:09.53	00:00:05.94	00:00:05.85

3. Cost-Complexity Pruning (CCP).

The same tendencies were observed for discrete decision class problems.

The summary ranking of post-pruning methods for C-fuzzy random forest with Cluster-context fuzzy decision trees is presented in Table 13. The content of the table are the

places of each method in comparison with the other ones. (The lower number means the method was better according to the criterion given in column.) The ranking clearly shows that for continuous decision class problems Reduced Error Pruning were the best solution. Critical Value Pruning achieved the worst results for both datasets. The greatest

Table 13 Ranking of post-pruning methods for C-fuzzy random forest with Cluster–context fuzzy decision trees

Post-pruning method	Classification accuracy	Computation time	Tree size
REP pruning	1	1	1
CCP pruning	2	2	3
CVP pruning	3	3	2

pruned tree size was produced by Cost-Complexity Pruning method.

The best 20 results of parameters optimization for Auto Data dataset, as an example of produced results which allowed to choose the best classifier's parameters, are presented in Table 14. These results are ordered the way the best results are at the top of the table. The optimization was performed for three membership functions, the following numbers of contexts and clusters: 2, 3, 5, 8, 13 and context configurations from 0.6 to 1.6 with step 0.2. As we can see, triangular and Gaussian membership functions allowed to achieve better results than trapezoidal one. Almost all of the best results (one exception) were achieved with only two contexts. None of the best twenty results were computed with 13 clusters. Except for that, the numbers of clusters and the context configuration were diversified among the whole searching space. Analogous optimization was performed for Boston Housing Data dataset. It was not needed to present these results here—the same tendencies were observed for this dataset as for a presented one.

7 Conclusions

In this paper, the idea of applying different methods to prune C-fuzzy decision trees and Cluster–context fuzzy decision trees in C-fuzzy random forest was presented. Five post-pruning methods were described and implemented into our ensemble classifier: Reduced Error Pruning (REP), Cost-Complexity Pruning (CCP), Pessimistic Error Pruning (PEP), Minimum Error Pruning (MEP) and Critical Value Pruning (CVP). These methods were evaluated using eleven discrete decision class datasets (using C-fuzzy random forest with C-fuzzy decision trees) and two continuous decision attribute datasets (using C-fuzzy random forest with Cluster–context fuzzy decision trees). Results were compared with

the ones achieved without pruning, with single C-fuzzy decision tree and C4.5 rev. 8 decision tree as a reference classifier. There was also tested the influence of using randomness on the achieved results.

According to achieved results, pruning C-fuzzy decision trees and Cluster–context fuzzy decision trees in C-fuzzy random forest can improve the classification accuracy and the precision of predicting decision attribute with reducing the size of trees, which also reduces classification time. Each of the evaluated pruning methods worked well for some datasets. Some of methods were significantly worse for some problems. Sometimes all of the methods are allowed to achieve comparable results. For discrete decision class problems, Critical Value Pruning allowed to achieve the best results in the greatest number of datasets; however, this method was relatively slow. For continuous decision attribute problems, Reduced Error Pruning appeared to be the best choice.

Performed researches allow us to conclude that the choice of pruning method should be made according to the given problem. Experiments showed that there is no universal pruning method which always works the best for all datasets. For different datasets, the different pruning methods allowed to achieve the best classification accuracy or predict the decision attribute in the best possible way. There are also datasets which should be classified without using any pruning method. What is more, the choice of pruning method should be performed according to the goal which we want to achieve. If classification time is more important than classification accuracy, another pruning method should be chosen than in situation where classification quality is the most significant and time is not. The choice of the pruning method should be always performed after analyzing the problem nature, needs and circumstances.

Current and previous studies showed that C-fuzzy random forest is a solution which can be competitive to other classifiers. It can be adjusted to many problems thanks to the different configuration parameters, which can be optimized according to the given dataset in order to achieve the best results. By implementing different pruning methods into the solution, we improved its quality and added another parameter which allows to fit the given problems and datasets in the best possible way. The flexibility of the classifier, combined with the strength of randomness and used pruning methods, makes C-fuzzy random forest the classifier which can be used for many classification and regression problems.

Table 14 Parameters optimization results for Auto Data dataset (20 best results)

Order	Membership function	Number of contexts	Number of clusters	Context configuration	Classification error (testing set)
1	Triangular	2	8	1.6	3278.612756
2	Triangular	2	8	1.4	3333.116369
3	Triangular	2	3	1.6	3501.166546
4	Gaussian	2	2	1.2	3598.238358
5	Gaussian	2	5	0.6	3610.350815
6	Triangular	2	2	1.4	3613.193789
7	Gaussian	2	2	0.6	3628.739464
8	Gaussian	2	3	0.6	3630.667416
9	Gaussian	2	2	1	3688.719952
10	Gaussian	2	3	1	3705.461266
11	Triangular	2	5	1.4	3728.756749
12	Gaussian	2	3	1.2	3733.855098
13	Gaussian	3	3	0.6	3740.699254
14	Triangular	2	5	1	3758.115725
15	Gaussian	2	5	0.8	3773.958063
16	Gaussian	2	2	1.4	3774.576382
17	Triangular	2	2	1.6	3797.393122
18	Trapezoidal	2	3	1.6	3804.182724
19	Triangular	2	5	1.6	3815.970543
20	Gaussian	2	3	1.4	3824.094031

Acknowledgements This work was supported by the Grant S/WI/3/2018 from the Białystok University of Technology founded by the Ministry of Science and Higher Education.

Compliance with ethical standards

Conflict of interest The authors declare that they have 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 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

- Bezdek JC (1981) Pattern recognition with fuzzy objective function algorithms. Kluwer Academic Publishers, Norwell
- Bonissone P, Cadenas JM, Garrido MC, Diaz-Valladares RA (2010) A fuzzy random forest. *Int J Approx Reason* 51(7):729–747
- Breiman L, Friedman J, Olshen R, Stone C (1984) Classification and regression trees. Wadsworth and Brooks, Monterey
- Cestnik B, Bratko I (1991) On estimating probabilities in tree pruning. In: *Machine learning—EWSL-91*. Springer, Berlin, pp 138–150
- Chen L, Su W, Feng Y, Wu M, She J, Hirota K (2020) Two-layer fuzzy multiple random forest for speech emotion recognition in human–robot interaction. *Inf Sci* 509:150–163
- Ciobanu LG, Sachdev PS, Trollor JN, Reppermund S, Thalamuthu A, Mather KA, Cohen-Woods S, Stacey D, Toben C, Schubert KO, Baune BT (2020) Downregulated transferrin receptor in the blood predicts recurrent mdd in the elderly cohort: a fuzzy forests approach. *J Affect Disord* 267:42–48
- Dua D, Graff C (2019) UCI machine learning repository. <http://archive.ics.uci.edu/ml>. University of California, School of Information and Computer Science, Irvine, CA
- Esposito F, Malerba D, Semeraro G, Kay J (1997) A comparative analysis of methods for pruning decision trees. *IEEE Trans Pattern Anal Mach Intell* 19(5):476–491
- 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
- Gadomer L, Sosnowski Z (2017) Using cluster–context fuzzy decision trees in fuzzy random forest. In: *Computer information systems and industrial management 2017*, LNCS 10244, pp 180–192

- Gadomer L, Sosnowski ZA (2019) Knowledge aggregation in decision-making process with C-fuzzy random forest using OWA operators. *Soft Comput* 23(11):3741–3755
- Mingers J (1987) Expert systems—rule induction with statistical data. *J Oper Res Soc* 38(1):39–47
- Mingers J (1989) An empirical comparison of pruning methods for decision tree induction. *Mach Learn* 4(2):227–243
- Naseer T, Asghar S, Zhuang Y, Fong S (2015) A comparative analysis of pruning methods for c4.5 and fuzzy c4.5. *Front Artif Intell Appl* 275:304–312
- Niblett T, Bratko I (1987) Learning decision rules in noisy domains. In: *Proceedings of expert systems '86, the 6th annual technical conference on research and development in expert systems III*, Cambridge University Press, New York, NY, USA, pp 25–34
- Ozigis MS, Kaduk JD, Jarvis CH, da Conceicao Bispo P, Balzter H (2020) Detection of oil pollution impacts on vegetation using multifrequency SAR, multispectral images with fuzzy forest and random forest methods. *Environ Pollut* 256:113360
- Pedrycz W (1996) Conditional fuzzy C-means. *Pattern Recogn Lett* 17(6):625–631. [https://doi.org/10.1016/0167-8655\(96\)00027-X](https://doi.org/10.1016/0167-8655(96)00027-X)
- Pedrycz W, Sosnowski Z (2005) C-fuzzy decision trees. *IEEE Trans Syst Man Cybern Part C Appl Rev* 35(4):498–511
- Quinlan JR (1987) Simplifying decision trees. *Int J Man Mach Stud* 27(3):221–234
- Ribeiro MV, Camargo HA, Cintra ME (2013) A comparative analysis of pruning strategies for fuzzy decision trees. In: *2013 Joint IFSA world congress and NAFIPS annual meeting (IFSA/NAFIPS)*, pp 709–714
- Sosnowski ZA (2012) Decision rules with fuzzy granulation of knowledge (in Polish). *Symulacja w Badaniach i Rozwoju* 3(4):225–232
- Sosnowski ZA, Gadomer L (2019) Fuzzy trees and forests—review. *WIREs Data Min Knowl Discov* 9(6):1–17

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