Pattern Recognition 133 (2023) 109036

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

DisRFC: a dissimilarity-based Random Forest Clustering approach

Manuele Bicego

Department of Computer Science, University of Verona, Strada le grazie, 15, Verona 37134, Italy

A R T I C L E   I N F O

Article history:
Received 21 December 2021
Revised 18 August 2022
Accepted 6 September 2022
Available online 11 September 2022

Keywords:
Random forests clustering
Dissimilarities
Unsupervised learning
Clustering
Non-vectorial representation

A B S T R A C T

In this paper we present a novel Random Forest Clustering approach, called *Dissimilarity Random Forest Clustering (DisRFC)*, which requires in input only pairwise dissimilarities. Thanks to this characteristic, the proposed approach is appliable to all those problems which involve non-vectorial representations, such as strings, sequences, graphs or 3D structures. In the proposed approach, we first train an *Unsupervised Dissimilarity Random Forest* (UD-RF), a novel variant of Random Forest which is completely unsupervised and based on dissimilarities. Then, we exploit the trained UD-RF to project the patterns to be clustered in a binary vectorial space, where the clustering is finally derived using fast and effective K-means procedures. In the paper we introduce different variants of DisRFC, which are thoroughly and positively evaluated on 12 different problems, also in comparison with alternative state-of-the-art approaches.

1. Introduction

Random Forests (RFs) [1] are ensemble classifiers, based on decision trees [2], models which define in their basic version a hierarchical splitting of the feature space – each split being defined with a threshold on a single feature. From a general perspective, Random Forests were mostly studied for regression and classification, representing in these fields state-of-the-art methods, able to compete with the most effective and established techniques (like SVM or Neural Networks). In different scenarios, such as clustering, their exploitation has not been completely investigated. Actually, even if some excellent methods are present (see Section 2 for a detailed description), the exploitation of RFs in the clustering scenario is far from being as mature as in classification/regression, and there is still large room for improvements. For example, one limit of current RF-clustering schemes is that they can be applied only if patterns to be clustered are represented with *vectors of features*. However, in many contexts it can be very challenging to define a vector of relevant features, and patterns can be more naturally represented with *non-vectorial representations*, such as sequences, sets, graphs, and so on. It seems therefore very interesting to define RF-clustering methods able to deal with these *non-vectorial* representations, and this represents the main goal of this paper.

Please note that, in the classification context, there has been an increasing interest in defining variants of Random Forests able to work with non-vectorial patterns; among the different approaches (see Section 2), the most promising and recent methods follow the idea of designing RFs which are completely based on *dissimilarities*[1] computed among non-vectorial objects. Using dissimilarity is definitely a preferable option, because, when dealing with non-vectorial representations, it happens very often that it is easier to derive a good dissimilarity rather than a set of discriminant features [3] – consider for example the many powerful distances for strings, sequences, or graphs developed in the past.

In this paper we introduce a novel RF-clustering scheme, called *DisRFC (Dissimilarity Random Forest Clustering)*, which is completely based on dissimilarities. Thanks to this characteristic, the method can be applied also to all those contexts for which a good set of features can not be derived but there exists a reasonable dissimilarity measure which can be computed. The proposed scheme consists of two main elements. The first is a novel variant of RF, which we call *Unsupervised Dissimilarity Random Forest (UD-RF)*, in which operations are completely *unsupervised* and *based on dissimilarities*. To build the model we define four training strategies, based on different ideas: (i) random training (similar in spirit to the Extremely Randomized Trees [4]), (ii) training by maximizing separation (using the Hausdorff distance [5]), (iii) training by minimizing the scatter and (iv) training by maximizing information divergence (using a recent dissimilarity-based estimator of the Rényi divergence measure [6]). The second element of DisRFC exploits the fact that each tree defines a hierarchical partition of the patterns of the problem, and derives the final clustering by using concepts and tools which are typical of the *clustering ensemble* field [7]. In detail, we exploit and extend some recent theoretical findings in the field of K-means-based consensus clustering [8,9], and define four different clustering schemes, all based on a K-means-type al-

[1] We use interchangeably the terms “dissimilarity” and “distance”.

https://doi.org/10.1016/j.patcog.2022.109036
0031-3203/© 2022 Elsevier Ltd. All rights reserved.

gorithm operating in a binary embedding space derived from the partitions induced by the learned UD-RF.

The proposed framework, in all its variants, has been thoroughly evaluated on 12 different problems related to non-vectorial objects. We analyzed different aspects, such as different training strategies, different embeddings and different ensemble schemes. We also performed a comparative analysis with other literature techniques, which confirms that DisRFC can represent a viable alternative to classic as well as to advanced dissimilarity based clustering approaches.

2. Related work

As stated in the introduction, the exploitation of Random Forests in the clustering scenario is not as mature as in classification/regression. Generally speaking, the RF-clustering approaches can be mainly divided into two groups: the first contains methods which exploit RFs (or RF-like methods) to directly extract the clustering; the second contains techniques which use RFs to derive a measure of dissimilarity between objects; the clustering is then obtained by applying to such dissimilarity a classic distance-based clustering method (like hierarchical clustering or spectral clustering [10]). More in detail, in the first group, Moosmann et al. [11] proposes a clustering approach to define dictionaries for Bag of Words classification of images; authors employed Extremely Randomized Trees [4] to derive clustering forests which are seen as partitioners of the space – each leaf representing a distinct visual word. Shotton et al. [12] extends the approach of Moosmann et al. [11] along different directions, e.g., by considering each tree as a hierarchy of clusters (we will use this idea in our approach). In [13], the authors propose a two-step clustering algorithm, in which they merged the multiple partitions defined by the different trees using a graph-based algorithm. Bicego [14] introduced a K-means-style clustering algorithm, in which every cluster is represented with a particular one-class variant of RF, called Isolation Forest [15]. Finally, Yan et al. [16] presents the “Cluster Forests” algorithm, based on an RF-like scheme, which defines an ensemble of clusterings by finding projections on which good local clusterings exist; the final result is then obtained by aggregation.

The second group of Random Forest clustering methods exploits the forest to derive similarities between objects, to be subsequently fed as input to a standard distance-based clustering algorithm. The basic idea, which goes back to Breiman [1], Shi and Horvath [17], is straightforward: given a tree, two objects can be considered similar if they end up in the same leaf, since they have answered in the same way to all the tests in their path (their paths are equal). This idea can be naturally extended to a forest level, and the most natural similarity measure is represented by the number of times – over the whole set of trees – two objects end up in the same leaf. Given this measure, Shi and Horvath [17] derives the final grouping using the PAM (Partitioning Around Medoids) algorithm. More recently, Zhu et al. [18] extended the approach in Breiman [1], Shi and Horvath [17] by considering as similarity between two objects the length of the common path they are following in each tree. This refines the previous binary definition (same/different leaf), considering as somehow similar also points which paths separate very deep in the tree – a very recent approach [19] further refined the concept.

All these RF-clustering approaches are inherently based on vectorial representations, i.e., objects to be clustered should be represented in a feature space. However, as already said in the introduction, there are many contexts in which it is difficult to derive a set of discriminative features, and objects are more naturally described with complex representations such as strings, sequences, graphs and so on. In the context of classification/regression, there has been an increased interest in devising Random Forest variants for modeling such non-vectorial objects. A first possibility is to derive summary features from them, like in Generalized Random Shapelet Forests [20]; an alternative and more promising trend is to develop RF variants based on similarity/dissimilarity, which permit to leverage the huge amount of meaningful distances between complex representations defined throughout many years of research. In this last research line all RF mechanisms should be defined in terms of similarity/dissimilarity; for example, in the method introduced in Douzal-Chouakria and Amblard [21] there are two prototypes, one for each branch, and the object follows the branch corresponding to the nearest prototype. More recent approaches introduce more sophisticated schemes, such as the use of a threshold on projections defined by distances between pairs of points [22], the use of triplet comparisons [23], or even the inclusion in the training of multiple dissimilarity measures [24]. In the clustering scenario, there are no RF-based methods dealing with non-vectorial objects; exporting to these cases the effectiveness and interpretability of RF-clustering approaches would be very valuable, and represents the main goal of this paper.

Summarizing, the main contributions of this paper are:

- the introduction of the first RF-clustering approach that is entirely based on dissimilarities, thus being able to work with non-vectorial representations. This method permits to export the efficacy and interpretability of RF-clustering approaches to many scenarios dealing with complex non-vectorial objects.
- the introduction of a novel variant of RF, the Unsupervised Dissimilarity Random Forest, which can be trained without labels and using only dissimilarities; this novel RF, equipped with non-conventional learning strategies based on the concepts of scatter, Hausdorff distance and Rényi divergence, can be exploited also in some other non-vectorial unsupervised scenarios (e.g., outlier detection [15]).

A preliminary version of this paper appeared in the proceedings of the Int. Conf. on Data Mining (ICDM) [25]. With respect to that version here we introduced novel learning schemes for the Unsupervised Dissimilarity Random Forest, we enlarged the experimental part with novel datasets, tables and comparisons, and in general we added thorough clarifications and explanations of all the parts of the methodology, providing details and algorithms.

3. The proposed approach: DisRFC

This section defines the proposed DisRFC (Dissimilarity Random Forest Clustering). In particular, it consists of two steps: i) given a dissimilarity function which permits to compute the dissimilarity between all pairs of objects to be clustered (or given the whole dissimilarity matrix), we train an Unsupervised Dissimilarity RF (UD-RF); ii) given the trained UD-RF, we combine the partitions defined by the trees of the UD-RF to get the final clustering. In the following, we will present the two steps into details.

3.1. Unsupervised dissimilarity random forests

The Unsupervised Dissimilarity Random Forest (UD-RF – see Algorithm 1) is an ensemble of Unsupervised Dissimilarity Trees (UD-T – see Algorithm 2). In the UD-RF, each UD-T is built using a subset of objects randomly sampled without replacement from the training set. We will start our definition by introducing the UD-T. Given a set of objects $\mathcal{O}$, the UD-T is a *complete* binary tree, which, to be trained, requires in input only the set of dissimilarities between the objects of $\mathcal{O}$. Without losing generality, we assume that the whole matrix of pairwise dissimilarities $D = [dis(o_i, o_h)]$ ($\forall o_i, o_h \in \mathcal{O}$, $dis(o_i, o_h)$ is the dissimilarity between objects o_i and o_h) is available in input. One important feature of the proposed approach is that we do not make any assumption on the nature of

Algorithm 1 *UD-RF*$(D, T, \psi, mls, np, tr)$.

```
Input: D: matrix of dissimilarities between objects; T: number of trees; ψ: subsampling size; mls: minimum leaf size; np: number of pairs; tr: training strategy
Output: R: an ensemble of T UD-Trees
1: $\mathcal{R} \leftarrow \emptyset$
2: $\mathcal{O} \leftarrow$ objects with dissimilarity matrix $D$
3: for $t \leftarrow 1 \ldots T$ do
4:     $S \leftarrow sample(\mathcal{O}, \psi)$        // sample $\psi$ objects from $\mathcal{O}$
5:     $D' \leftarrow D(S, S)$        // distances between all objects in $S$
6:     $\mathcal{R} \leftarrow \mathcal{R} \bigcup$ UD-Tree$(D', mls, np, tr)$
7: end for
8: $\mathcal{R}.numtrees \leftarrow T$
9: return $\mathcal{R}$
```

Algorithm 2 UD-Tree(D, mls, np, tr).

```
Input: D: matrix of dissimilarities between objects; mls: minimum leaf size; np: number of pairs; tr: training strategy
Output: an UD-Tree
1: $\mathcal{O} \leftarrow$ objects with dissimilarity matrix $D$
2: if $|\mathcal{O}| < mls$ then
3:     return leaf
4: end if
5: $V \leftarrow$ all $(o_i, o_j)$ $(i \neq j)$ in $ValidPairs(\mathcal{O})$
6: switch tr do
7:     case Rand
8:         $[\hat{o}^L, \hat{o}^R] \leftarrow samplepair(V, 1)$        // sample 1 pair from V
9:     case HausD
10:        $V' \leftarrow samplepair(V, np)$        // sample $np$ pairs from V
11:        $[\hat{o}^L, \hat{o}^R] \leftarrow \arg\max_{(a,b)\in V'}$ SHD(Nearer($\mathcal{O}$,a,b),Nearer($\mathcal{O}$,b,a))        // eq.(3)
12:    case Scatter
13:        $V' \leftarrow samplepair(V, np)$        // sample $np$ pairs from V
14:        $[\hat{o}^L, \hat{o}^R] \leftarrow \arg\min_{(a,b)\in V'}$ Scat(Nearer($\mathcal{O}$,a,b)) + Scat(Nearer($\mathcal{O}$,b,a))        // eq.(5)
15:    case RenyiD
16:        $V' \leftarrow samplepair(V, np)$        // sample $np$ pairs from V
17:        $[\hat{o}^L, \hat{o}^R] \leftarrow \arg\max_{(a,b)\in V'}$ SRD(Nearer($\mathcal{O}$,a,b),Nearer($\mathcal{O}$,b,a))        // eq.(8)
18: $O_l \leftarrow$Nearer$(\mathcal{O}, \hat{o}^L, \hat{o}^R)^*$, $O_r \leftarrow$Nearer$(\mathcal{O}, \hat{o}^R, \hat{o}^L)$  (*)
19: $D_l \leftarrow D(O_l, O_l)$, $D_r \leftarrow D(O_r, O_r)$
20: $node.Left \leftarrow$ UD-Tree$(D_l, mls, tr)$
21: $node.Right \leftarrow$ UD-Tree$(D_r, mls, tr)$
22: $node.LeftProt \leftarrow \hat{o}^L$
23: $node.RightProt \leftarrow \hat{o}^R$
24: return node
```

NOTES: *Nearer*($\mathcal{O}$,a,b) (lines 11, 14, 17) returns all elements in $\mathcal{O}$ which are nearer to a than to b; *ValidPairs*($\mathcal{O}$) (line 5) contains all pairs of objects in ($\mathcal{O}$) which permit to split the objects in two non empty sets.

the dissimilarity (which can be also non-Euclidean)– this is a typical scenario when dealing with complex objects described with non-vectorial representations [26].

To define the UD-T, we have to define the traversal scheme and the training strategy. For what concerns the former, we borrow here the mechanism defined in Proximity Forests [24], a recent variant of RF completely based on dissimilarities defined for supervised classification. More specifically, each node n_j of the UD-T has associated two objects o^L_j and o^R_j, called prototypes. These objects define the traversal scheme of the node n_j: if an object o is more similar to o^L_j than to o^R_j, i.e $dis(o, o^L_j) < dis(o, o^R_j)$, then it will follow the left path, otherwise it will follow the right path. The traversal scheme is used during both the training and the testing steps. For what concerns the training strategy, the input is represented by a set of objects, for which we have at disposal (or we can compute) all pairwise dissimilarities. The training procedure is a classical tree-training strategy. Starting from the root n_1, which accommodates all the training objects, we create two children nodes according to the distance to the two prototypes o^L_1 and o^R_1; in detail, objects nearest to o^L_1 will follow the left branch, i.e., they will be assigned to the left child, the others will follow the right branch. We then continue this splitting procedure until we met some ending conditions (e.g., we cannot split a node anymore, or we have reached a predefined depth of the tree). We will denote as S_j the set of objects of the training set of the tree which are passing through the node n_j, and as S^L_j (S^R_j) the objects choosing the left (right) child of n_j. Clearly $S^L_j \cup S^R_j = S_j$, and $S^L_j \cap S^R_j = \emptyset$. The learning procedure indicates how to choose, for every node n_j, the two prototypes o^L_j and o^R_j. In particular, to choose the prototypes of the node n_j we are using objects in S_j, i.e., the training objects arriving at that node. We propose four different approaches to select the prototypes, described in the following, which exploit different concepts and ideas, but are all based on dissimilarities. Please note that we are in an unsupervised scenario (clustering), and therefore no labels are available.[2]

"Rand": this represents the simplest strategy in which, for a node n_j, we randomly choose o^L_j, o^R_j, $o^L_j \neq o^R_j$, in the set S_j. Random training of decision trees has been shown to be surprisingly good; after the pioneering work of Geurts and colleagues [4] on Extremely Randomized Trees for classification, this solution was successfully adopted in many contexts (e.g., Moosmann et al. [11], Shotton et al. [12], Liu et al. [15], just to cite a few). **"HausD"**: this scheme starts from the idea that a proper split in a node n_j should divide the objects S_j in two *well separated* sets. To implement this idea, we propose to measure the separation between S^L_j and S^R_j using the Hausdorff distance [5], a classic measure to compare sets of objects, working very well in different contexts, which, crucially, can be computed starting only from pairwise distances. Formally, given a distance between objects $dis(x, y)$, and given two sets $X = \{x_1, \ldots x_N\}$ and $Y = \{y_1 \ldots y_M\}$, the Hausdorff distance between X and Y is defined as:

$$HD(X, Y) = \max_{i\in[1...N]} \min_{j\in[1...M]} dis(x_i, y_j). \tag{1}$$

The proposed "HausD" training scheme chooses the prototypes to maximize the Hausdorff distance between the left child and the right child sets. More formally, the pair $(\hat{o}^L_j, \hat{o}^R_j)$ for the node n_j is selected as:

$$(\hat{o}^L_j, \hat{o}^R_j) = \arg\max_{o^L_j, o^R_j \in S_j} SHD(S^L_j, S^R_j), \tag{2}$$

with $SHD(S^L_j, S^R_j)$ being the Symmetrized Hausdorff distance between S^L_j and S^R_j, defined as

$$SHD(S^L_j, S^R_j) = \frac{1}{2}\Big(HD(S^L_j, S^R_j) + HD(S^R_j, S^L_j)\Big). \tag{3}$$

From a computational perspective, the analysis of all possible pairs (o^L_j, o^R_j) is clearly not efficient; we overcome this problem with a classic trick employed in standard classification Random Forests [1], namely picking the best pair among a *small* number of random pairs (in our experiments we set this number to 20).

[2] In some works on RF-clustering [17,18], the problem is solved by sampling a negative class from the feature space, and use a classification RF. Clearly this solution can not be adopted here, since in input we do not have a feature space.

"Scatter": the intuition behind this scheme is that a proper split in a node n_j should divide the objects S_j in two *highly compact* sets. To measure the compactness of a set of objects starting from dissimilarities, we used the classic concept of *Scatter*, namely the averaged pairwise distance between all objects in the set. More formally, given a set of objects $X = \{x_1, \dots x_N\}$, and a distance $dis(x, y)$ between them, the scatter $Scat(X)$ is defined as:

$$Scat(X) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=i}^{N} dis(x_i, x_j). \tag{4}$$

In the "Scatter" scheme, the best prototypes for a node n_j are those for which the scatter of S_j^L and S_j^R are minimized; more formally, the pair $(\hat{o}_j^L, \hat{o}_j^R)$ for the node n_j is selected as:

$$(\hat{o}_j^L, \hat{o}_j^R) = \arg \min_{o_j^L, o_j^R \in S_j} \left(Scat(S_j^L) + Scat(S_j^R) \right). \tag{5}$$

Similarly to the HausD scheme, this approach chooses the best pair among a small set of pairs randomly sampled from the objects S_j. Please note that this criterion is similar in spirit to classic impurity-based criteria for classification, like the Gini one. However, in such classic criteria, the contribution to the total impurity of the two parts S_j^L and S_j^R is weighted by the size of S_j^L and S_j^R. Clearly also in our case we can use a weighted variant, even if we think that in this clustering context the unweighted version would favour purer nodes, not considering their size. **"RényiD"**: the intuition behind this last scheme is that a proper split of a node n_j should divide the objects S_j into two sets which *convey different information*. To measure this, we can employ concepts and tools from Information Theory, this being a classic solution employed in many Random Forests training schemes [1]. More in detail, here we decide to select the pair of objects which permits to have the highest Rényi divergence [27] between S_j^L and S_j^R. We chose the Rényi divergence, a generalization of the Kullback–Leibler divergence between probability distributions, since the estimation of the classic divergence based on the Shannon entropy is often more problematic (see Pál et al. [28]). In particular, we employed here the recent estimation procedure proposed in Noshad et al. [6], which seems to be particularly suited for our scenario: i) this technique permits a non-parametric estimation, i.e., no assumptions on the underlying distribution should be done; this is reasonable for our case, since a priori we do not know the shape of the data we are clustering; ii) this technique implements a *by-pass estimator* [28], i.e., it does not require the explicit estimation of the density (avoiding all issues related to density estimation, such as the determination of the partition function); iii) the computation employs nearest neighbor ratios, thus being based only on dissimilarities; iv) finally, this technique is able to derive proper estimates also when the input dissimilarity is non-Euclidean, thus being suitable for our scenario.

Formally, given two sets $X = \{x_1, \dots x_N\}$ and $Y = \{y_1 \dots y_M\}$, the Rényi divergence between X and Y can be estimated via the following equation:

$$RD(X, Y) = \frac{1}{(\alpha - 1)} \log \left[\frac{\eta^\alpha}{M} \sum_{i=1}^{M} \left(\frac{N_i}{M_i + 1} \right)^\alpha \right]. \tag{6}$$

In the above formula, $\eta = M/N$, N_i and M_i represent the number of K-nearest neighbors of y_i in $\{X \cup Y\}$ belonging to X and Y, respectively, and α defines the Rényi divergence ($\alpha > 0, \alpha \neq 1$)– all the mathematical details can be found in the original paper [6]. In our "RényiD" learning scheme, the best prototypes for a node n_j are those for which the Rényi divergence between S_j^L and S_j^R is maximized. More formally, $(\hat{o}_j^L, \hat{o}_j^R)$ are defined as:

$$(\hat{o}_j^L, \hat{o}_j^R) = \arg \max_{o_j^L, o_j^R \in S_j} SRD(S_j^L, S_j^R), \tag{7}$$

where $SRD(S_j^L, S_j^R)$ represents the symmetrized Rényi divergence between S_j^L and S_j^R, defined as:

$$SRD(S_j^L, S_j^R) = \frac{1}{2} \left(RD(S_j^L, S_j^R) + RD(S_j^R, S_j^L) \right). \tag{8}$$

Also in this case the best pair is chosen among a small set of pairs randomly sampled from the objects S_j.

Let us conclude with a final consideration. Obviously, given a dissimilarity, it is possible to embed the problem in a vectorial space (e.g., via multidimensional scaling), and use standard vector-based RF-clustering schemes. However, when dealing with non euclidean (or even non-metric) dissimilarities (typical scenario with non-vectorial objects [26]), the embedding in vectorial spaces may be rather complicated, and methods working directly with dissimilarities are preferred. We will provide some evidence of this in the experimental part.

3.2. Ensemble clustering with UD-RF

The second step of the proposed framework exploits the trained UD-RF to determine the final clustering. In particular, we rely on the idea, already presented in early approaches for RF-clustering [11–13,29], that given a tree we can aggregate the different objects into groups by looking at the path they are following from the root to the leaves. The trained UD-RF is a set of UD-T, and therefore defines a set of different clusterings, one for each UD-T. To derive the final clustering, we can merge all these clusterings, exploiting concepts and tools from Consensus Clustering (also known as Ensemble clustering) [7]. Generally speaking, this scheme can be seen a hierarchical divisive approach (a less common strategy for deriving a hierarchical clustering), in which the typical computational issues associated with divisive methods are faced by aggregating simple and somehow approximated clusterings (each one derived from a single tree). More in detail, the scheme we propose has its roots in the works of Topchy et al. [8], Mirkin [30], which showed that it is possible to transform the maximization of the Category Utility Function [30] –a widely known (and computationally demanding) criterion for consensus clustering–, into a fast K-means scheme with squared Euclidean distances in a properly defined space derived from the input partitions. The authors of Wu et al. [9] extended this elegant result by showing that, under some conditions, it is possible to prove the same equivalence for other criteria used in consensus clustering, typically by using other distances within the K-means. Interestingly, the authors of Wu et al. [9] also provided necessary and sufficient conditions for transforming a consensus clustering problem into a K-means clustering problem.

Being inspired by these findings, in our framework we exploit the trained UD-RF to define an embedding space for the objects of the problem; the final clustering is then obtained by a clustering procedure in such space. The two steps are presented in the following.

Embedding with UD-RF In our approach, we present two different embedding schemes, the "OP" (One Partition) and the "MP" (More Partitions), both defined starting from the hierarchical clusterings determined by the different trees of the UD-RF. As the name suggests, the difference between the two schemes is in the *number of partitions* of each tree used in the embedding; in particular, "OP" exploits only one of the partitions defined by the tree, whereas "MP" uses more than one. Even if being more computationally demanding, the second method may permit to better exploit the information contained in the clustering hierarchy defined by the tree. Before entering into the details of the two embeddings, let us present the general framework. Given a UD-RF with T trees, the embedding $\mathcal{E}(o)$ of an object o is obtained by concatenating the T embeddings of o in the T trees:

$$\mathcal{E}(o) = [E^1(o)E^2(o)\dots E^T(o)], \tag{9}$$

with $E^t(o)$ being the embedding of o obtained from a tree t. For simplicity, let us remove the superscript t, and present OP/MP for a single tree. Let us denote as $\{n_1,\dots,n_N\}$ the set of the nodes of the tree, and $\mathcal{P}(o) \subset \{n_1,\dots,n_N\}$ the path of the object o to reach its leaf $\ell(o)$.

For a pre-specified depth d ($d \leq d_{\max}$, the maximum depth of the tree), let us define $\mathcal{P}_d(o)$ as the path of o in the tree until the depth d; this can also be seen as the set of nodes of the path which have a depth less or equal to d:

$$\mathcal{P}_d(o) = \{n_j \in \mathcal{P}(o) | dep(n_j) \leq d\}. \tag{10}$$

In the above formula $dep(n_j)$ represents the depth of the node n_j. Clearly, $\mathcal{P}_0(o)$ contains only the root, whereas $\mathcal{P}_{d_{\max}}(o) = \mathcal{P}(o)$. Finally, we define as $c_d(o)$ the node with maximum depth in $\mathcal{P}_d(o)$:

$$c_d(o) = \arg\max_{n \in \mathcal{P}_d(o)} dep(n). \tag{11}$$

Now, given a depth d, we can create a partition of the objects on the basis of their $c_d(o)$, assigning objects with the same $c_d(o)$ to the same cluster. In other words, a cluster at depth d contains all those objects which reach at depth d the same node. It is easy to see that our definition generalizes the definition of Moosmann et al. [11], Shotton et al. [12], Perbet et al. [13], which specified that a cluster must contain all those objects which fall in the same leaf. In our generalization, we permit to define the level (i.e., the depth) at which we should examine the hierarchical clustering defined by the tree. This can be useful if we want to set a precise number of clusters, or if we want clusters with a minimum size.[3] More formally, the objects to be clustered $\mathcal{O}$ can be partitioned (at a given depth d) into the clustering $\mathcal{C}_d$, defined by using the set $C_d = \{\tilde{c}_d^1, \dots \tilde{c}_d^{K_d}\}$ of K_d unique $c_d(o)$ for all $o \in \mathcal{O}$:

$$\mathcal{C}_d = \{\mathcal{C}_d^1, \dots \mathcal{C}_d^{K_d}\}, \tag{12}$$

where

$$\mathcal{C}_d^k = \{o_j \in \mathcal{O} | c_d(o_j) = \tilde{c}_d^k\}. \tag{13}$$

Given this definition, we are now ready to define the "OP" embedding (One Partition) for an object o and given a depth d:

$$E^{OP_d}(o) = \left[e_1^{OP_d}(o), e_2^{OP_d}(o), \dots e_{K_d}^{OP_d}\right], \tag{14}$$

where

$$e_j^{OP_d}(o) = \begin{cases} 1 & \text{if } c_d(o) = \tilde{c}_d^j \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

$E^{OP_d}(o)$ represents a binary vector of K_d entries, where we have all zeros except in one position, namely the position of the node reached by the object o at depth d (i.e., $c_d(o)$). It is easy to show that, when we plug Eq. (14) into Eq. (9), we get an embedding which is equivalent to the encoding defined in Topchy et al. [8], Wu et al. [9], Mirkin [30] if we assume that the clusterings to be combined are those induced by the set of the different $\mathcal{C}_d$ (one for each tree of the forest).

For what concerns the second embedding, MP, the starting idea is that a tree does not define only a clustering, but a *hierarchy* of clusterings (one for each depth d), which can be exploited to better characterize the objects. Actually, in MP the idea is to define the embedding starting from more than one partition: in particular we can consider a set of H depths $\mathcal{H} = \{d_1, \dots d_H\}$, and concatenate the OP embeddings derived from each depth in $\mathcal{H}$ to get a richer embedding. Formally,

$$E^{MP}(o) = \left[E^{OP_{d_1}}(o) \dots E^{OP_{d_H}}(o)\right], \quad d_h \in \mathcal{H}. \tag{16}$$

With this strategy, we can benefit from information at different levels of granularity, starting from coarser partitions (at small depths) up to finer ones (larger depths). In our experiments we employed a few partitions at equi-spaced depths, in order to cover the whole scale of resolutions. Also, in this case it is easy to show that when plugging Eq. (16) into Eq. (9) we can still use the perspective of Topchy et al. [8], Wu et al. [9], Mirkin [30]. Actually the MP embedding simply induces more partitions to be combined, with different number of clusters; since the theory presented in Topchy et al. [8], Wu et al. [9], Mirkin [30] does not assume to have a fixed number of clusters, we are still guaranteed to have a proper embedding.

A summary of the embedding step can be found in Algorithm 3

Algorithm 3 *Embed($D, \mathcal{R}, \mathcal{H}$).*

```
Input: $D$: matrix of dissimilarities between objects; $R$: the trained UD-RF; $\mathcal{H}$: the set of depths
Output: $\mathcal{E}$: the embedding of objects
1: $\mathcal{O} \leftarrow$ objects with dissimilarity matrix $D$
2: for $o \in \mathcal{O}$ do
3:     $\mathcal{E}(o) \leftarrow \emptyset$                          // Initialization
4: end for
5: for $t \leftarrow 1\dots\mathcal{R}.numtrees$ do
6:     $tree \leftarrow \mathcal{R}[i]$                          // i-th tree
7:     for $o \in \mathcal{O}$ do
8:         $\mathcal{P}(o) \leftarrow getPath(o, tree)$          // nodes from root to leaf
9:     end for
10:    for $d \in \mathcal{H}$ do
11:        for $o \in \mathcal{O}$ do
12:            $c_d(o) \leftarrow \mathcal{P}(o)[d]$          // d-th node in the path $\mathcal{P}(o)$
13:        end for
14:        $C_d \leftarrow unique_{o\in\mathcal{O}}(\{c_d(o)\})$          // unique $c_d(o)$
15:        for $o \in \mathcal{O}$ do
16:            $i \leftarrow 0$
17:            for $\tilde{c}_d \in C_d$ do
18:                $i \leftarrow i + 1$
19:                if $c_d(o) = \tilde{c}_d$ then
20:                    $E(o)[i] \leftarrow 1$          // i-th element of array $E(o)$
21:                else
22:                    $E(o)[i] \leftarrow 0$
23:                end if
24:            end for
25:            $\mathcal{E}(o) \leftarrow [\mathcal{E}(o)E(o)]$          // Concatenation
26:        end for
27:    end for
28: end for
29: return $\mathcal{E}$
```

– please note that to have the *OP* embedding at depth d we just have to set $\mathcal{H} = \{d\}$.

Clustering As stated above, the works in Topchy et al. [8], Mirkin [30] theoretically proved that the optimum of the Category Utility Function can be found by running a classic K-means with squared Euclidean distances in a specific space which encodes the different input partitions. Further, the work of Wu et al. [9] theoretically shows that to optimize alternative consensus clustering criteria we can still use K-means in the same binary space, but changing distance. Having shown that our OP and MP embeddings represent proper encodings in the view of Topchy et al. [8], Wu et al. [9], Mirkin [30], we can leverage all the results proven there and derive a set of effective and fast consensus clustering methods for aggregating the information contained in all the trees. In particular, we

[3] Please note that this cannot be achieved by setting a maximum depth in the tree, since each tree is built on a subsample of the objects to be clustered, and thus we do not know a priori how many objects will fall in a given leaf.

define four different clustering schemes, based on this paradigm, where the first two directly derive from the framework of Topchy et al. [8], Wu et al. [9], Mirkin [30], and the last two are generalizations. More in detail we defined:

- **KM**: a K-means clustering method in the embedding space; here we used squared Euclidean distances, thus corresponding to the approach presented in Topchy et al. [8], Mirkin [30];
- **KMKL**: again a K-means in the embedding space, but using the Kullback–Leibler divergence to measure distance between objects. In particular, in this variant, in the assignment step of the K-means an object o_i is assigned to the cluster $\hat{c}$ for which the Kullback–Leibler divergence is minimum

  $$\hat{c} = \arg\min_{c=1\ldots K} d_{KL}(\mathcal{E}(o), \mathcal{E}_c),$$

  where $\mathcal{E}_c$ is the average of the embeddings of the objects assigned to the cluster c at previous iteration and d_{KL} is the Kullback–Leibler divergence. According to Wu et al. [9], with this scheme we are optimizing a Shannon-entropy derived consensus clustering criterion;
- **Kmed, Kmod**: two generalizations of **KM** and **KMKL**, in which we used in the embedding space two classical solutions for categorical clustering, namely the K-medoids algorithm with the Hamming distance [31] and the K-modes approach [32]. Even if we will show in the experimental evaluation that these variants represent fast and accurate options, we lose the theoretical properties of **KM** and **KMKL**: actually the results of Topchy et al. [8], Wu et al. [9], Mirkin [30] only apply if the clustering algorithm is the K-means; extending this theory to **Kmed** and **Kmod** will be the subject of our future research.

As a summary, Algorithm 4 describes the whole DisRFC clustering pipeline[4] As a final note, it should be observed that in our framework the number of clusters has to be fixed and known, which is unrealistic in many real situations; a possible solution is to use in the last step of the pipeline a clustering algorithm that is able to automatically determine the best number of clusters – an excellent candidate in this sense would be the x-means method [33].

Algorithm 4 *DisRFC($D, T, \psi, mls, np, tr, \mathcal{H}, K, Y_{init}, cl$).*

```
Input: D: matrix of dissimilarities between objects; T: number of
    trees; ψ: subsampling size; mls: minimum leaf size; np: number of pairs;
    tr: training strategy; H: set of depths; K: number of clusters;
    Y_init: initialization of clustering; cl: type of clustering
Output: Y: the clustering; e the value of the optimized function
1: $\mathcal{R} \leftarrow$ UD-RF($D, T, \psi, mls, np, tr$)
2: $\mathcal{E} \leftarrow$ Embed($D, \mathcal{R}, \mathcal{H}$)
3: $[Y, e] \leftarrow$ Cluster($\mathcal{E}, K, Y_{init}, cl$)
4: return $Y, e$
```

4. Experimental evaluation

This section presents the experimental analysis. After introducing the experimental details, we will present the results for the different variants of the DisRFC approach, analyzing the impact of the different options; finally, we present a comparative analysis with some literature alternatives.

4.1. Experimental details

As usually done in the clustering domain, in our empirical evaluation we employed supervised datasets, in which the labels are removed for computing the clustering and used to check the goodness of the result; in particular we used here the adjusted Rand index (ARI – [34]), a classical measure which exploits a contingency table between the clustering and the true labeling; from this table the agreement between the two groupings is quantified and corrected for the chance of the formation of the clusters – the higher this index, the better the clustering.

Our experiments are based on 12 problems, briefly presented in Table 1 and described in Section 1 of the supplementary material. These datasets cover different aspects, such as different representations and dissimilarities (most of them are non-Euclidean), different numbers of objects and clusters, different dimensionality of the clusters. For each problem, we have as input a matrix directly containing all pairwise dissimilarities between the objects of the problem. It is worth to observe that in all cases except FlowCyto1 and FlowCyto2 we are dealing with non-vectorial representations (such as sequences, strings, 3D structures or graphs), this permitting to evaluate the proposed approach as a general RF clustering method usable to cluster non-vectorial objects. In the experiments we evaluate the proposed approach in all its possible variants, i.e. by combining all the training schemes (Rand, HausD, Scatter, and RényiD) with all the embeddings (OP and MP) and the clustering methods (KM, KMKL, Kmed and Kmod). We employed 100 or 200 trees to build each UD-RF; we trained each tree with 128 objects randomly chosen from the training set (in the CatCortex case we used all the 65 objects). Training with such a few objects, on top of permitting a faster training and a reasonably sized embedding space, permits to increase the diversity of the obtained partitions, which represents a crucial aspect for any successful ensemble clustering – preliminary experiments, not shown here, confirmed that increasing this number does not lead to any substantial improvement. Training with few samples has shown to be effective also in other scenarios, especially those alternative to classification or regression, such as outlier detection [15]. In all training schemes, we stop the recursive splitting of a node when it contains fewer than 10 objects. In the Rand strategy the two splitting prototypes inside a node are randomly chosen among all *valid* pairs of objects reaching that node (we denote as *valid* a pair which permits to split objects in two non-empty sets); for the other training schemes, i.e., HausD, Scatter and RényiD, the two prototypes are selected among 20 *valid* random pairs by optimizing the criteria in Eqs. (3), (5), and (8), respectively. In the computation of the Rényi divergence, we used $\alpha = 0.999$, and $K = \sqrt{\Psi}$ (as recommended by Noshad et al. [6]), where Ψ is the number of the objects in the training set of the tree. For the OP embedding we select $d = d_{\max}$, i.e., we used the clustering induced by the leaves; for the MP embedding we used $H = 3$ partitions, choosing them as approximately equispaced in the hierarchy. After a few trials, we found the following rule of thumb: we divided the interval $[1 : d_{\max}]$ into $H + 1$ intervals, keeping as d_j the central level of all intervals (except the first). This permits to have partitions from coarse to fine granularity – we remove the partition in the first interval, since too coarse. Finally, in order to cope with the randomness present in the whole pipeline (random selection of training objects, random selection of candidate pairs and so on), for each variant and each dataset we repeated the whole procedure 30 times.

4.2. Results

Given the large number of experiments, we present the whole set of results in Section 2 of the supplementary material, providing here only a summary. In particular, we present in Tables 2(a),

[4] We did not provide the algorithm of *Cluster*($\mathcal{E}, K, Y_{init}, cl$), since it represents one of the four well known clustering approaches described above applied in the embedding space $\mathcal{E}$.

Table 1
Datasets used in the experiments (N: number of objects, K: number of clusters).

Problem	Acronym	Description	N	K
CatC	CatC	Connections between cortical areas of a cat	65	4
Protein	Prot	Evol. diss. of sequences of proteins	213	4
CoilDelft	CoD	Spectral dist. of 4 COIL obj. graphs	288	4
ChickenPieces	ChiP	Weigh. Edit dist. of 2D shape contours	446	5
Newsgroups	News	Non-metric correl. of newsgroups messages	600	4
FlowCyto1	Fl1	L1 dist. of flow-cytometer histogr. (ch. 3)	612	3
Flowcyto2	Fl2	L1 dist. of flow-cytometer histogr. (ch. 4)	612	3
WoodyPlants	WoP	Shape distances among leaves of plants	791	14
Delftgestures	DeGe	DTW diss. between gesture signs (video)	1500	20
Zongker	Zon	Deform. templ. match. of handwritten dig.	2000	10
TwoPendigits	2Pe	Edit dist. between two digits contour	2287	2
Prodom	Prod	Struct. alignm. between protein domains	2604	4

Table 2
Analysis of different aspects of the proposed approach: (a) Training, (b) Embedding.

	Training *(Total: 480)*			
Problem	Rand	HausD	Scatter	RényiD
(a)				
CatC	0.6189	0.6362	0.3522	**0.6670***
Prot	0.6268	0.5595	0.5323	**0.8592***
CoD	0.0930	0.0865	0.0833	**0.1601***
ChiP	0.3114	0.2805	0.3113	**0.3351***
News	0.0882	0.1191	0.1203	**0.2175***
Fl1	0.0792	0.0634	**0.0940***	0.0557
Fl2	0.0870	0.0726	**0.0979***	0.0689
WoP	**0.5363***	0.5277	0.5048	0.4514
DeGe	0.7099	**0.7120**	0.6853	0.6225
Zon	**0.7312**	0.7197	0.6897	0.6765
2Pe	0.9110	0.8631	0.8349	**0.9696***
Prodom	0.1507	0.1064	0.0962	**0.3061***
Average	0.4120	0.3956	0.3668	**0.4491***

	Embedding *(Total: 960)*	
Problem	OP	MP
(b)		
CatC	**0.5933***	0.5439
Prot	0.5869	**0.7020***
CoD	**0.1068**	0.1046
ChiP	0.3076	**0.3115***
News	**0.1524***	0.1201
Fl1	0.0684	**0.0777***
Fl2	0.0740	**0.0892***
WoP	**0.5066**	0.5035
DeGe	**0.6939***	0.6709
Zon	**0.7154***	0.6932
2Pe	0.8381	**0.9512***
Prodom	**0.1777***	0.1520
Average	0.4018	**0.4100***

Table 3
Analysis of different aspects of the proposed approach: clustering.

	Clustering *(Total: 480)*			
Problem	KM	KMKL	Kmed	Kmod
CatC	0.6494	0.2050	0.6962	**0.7237***
Prot	0.7058	0.3313	0.7296	**0.8111***
CoD	**0.1449***	0.0466	0.1063	0.1251
ChiP	**0.3220***	0.2911	0.3163	0.3090
News	**0.2217***	0.1357	0.0682	0.1194
Fl1	**0.0810***	0.0767	0.0612	0.0734
Fl2	**0.0938***	0.0798	0.0694	0.0834
WoP	0.5400	0.3896	**0.5467**	0.5440
DeGe	0.7498	0.5025	**0.7516**	0.7258
Zon	0.7444	0.5539	0.7450	**0.7739***
2Pe	0.9771	0.9402	0.6838	**0.9775**
Prodom	0.1528	0.1669	0.1523	**0.1875***
Average	0.4486	0.3100	0.4105	**0.4545**

(b) and 3 a set of results aimed at comparing the different possible options for each of the three steps (training, embedding, clustering). More in detail, for each step, we compute and compare the average of the ARI values of the different options computed across all the other aspects. For example, when analyzing the training, we compute the average of all the ARI values obtained with Rand (or HausD, Scatter, RényiD) for all different forest parameters (number of trees), all embeddings and all clustering schemes, thus resulting, for each dataset and each option, in the average of 480 values (2 parametrizations $\times$ 2 embeddings $\times$ 4 clusterings $\times$ 30 repetitions). We show these averages in Tables 2(a), (b) and 3, for the training, the embedding and the clustering, respectively; a bold value indicates the best result among the different options for a particular step (training, embedding, clustering). In order to assess the statistical significance, we also performed a paired t-test among the ARI values of the best option and those of the second best, where the null hypothesis is that the mean of the differences is zero. In the tables, an asterisk indicates that the difference between the best option and the second best is statistically significant according to our statistical test (with a significance level of 0.05).

The results allow for different observations. Regarding the training procedure, it is interesting to note that the RényiD strategy is in general the best version. In many problems its averaged ARI is larger than that of the alternative options; please note that in some cases (like Protein, CoilDelft, Newsgroup and Prodom) the difference is quite high. There are some other cases, i.e. WoodyPlants, Delftgestures and Zongker, in which the RényiD variant represents the worst choice; note that all these problems contain many clusters (14, 20 and 10), which probably cannot be organized in a hierarchy (as that returned by a UD-tree); this misalignment becomes more relevant with more accurate hierarchies (as those derived by using the RényiD scheme). We can also observe that the Rand scheme represents a fast but very reasonable option, working very well almost everywhere, confirming findings found in other contexts [4,11,15]. For what concerns the embedding, we can observe that the differences between OP and MP are not that large, the only exception being the TwoPendigits case, where MP drastically outperforms OP. Finally, regarding the clustering, we can observe that the best results are in general obtained with K-Means with Squared Euclidean distances and K-modes; only in two cases K-medoids returns the best accuracy, even if not significantly better than the second one. In general, KMKL has the most unstable behavior across datasets; for some datasets it is drastically worse than the alternatives (e.g., the CatCortex problem), for some others it represents the best option (WoodyPlants and Delftgestures). We are still investigating this strange behavior, trying to link it to the characteristics of the problem or to the difficulties in estimating the KL divergence.

4.2.1. Guidelines

Starting from all the presented results, we derived a few guidelines, which can help the user in choosing the proper variant in a given scenario. For large enough datasets (e.g., containing more than 128 objects) with a reduced number of clusters e.g., less than 10), our suggestion is to employ the RényiD training procedure; in all other cases, we suggest using the Rand one. For the remaining aspects, our suggestion is to use the OP embedding at the leaves level (this being less computationally demanding than the MP) and the K-Means with Squared Euclidean distances as clustering method (more theoretically sound than K-modes). In the next section, we will show the effectiveness of these guidelines via a comparison with other approaches present in the literature.

4.3. Comparison with the literature

In this section, we compare the proposed approach with some other distance-based clustering approaches. We tried to include in our analysis different methods, ranging from classic approaches (like Hierarchical Methods or K-medoids) up to more complex and recent schemes (like kernel methods or density-based approaches). We also provided a comparison with standard Random Forest-based clustering schemes [17–19,35]. Please note that, since these methods require in input a vectorial representation, we had to derive a feature space from the dissimilarity matrix, which is obtained in this case with the classical Multidimensional Scaling. All these competitors, together with the implementation details, are described in Section 3 of the supplementary material, together with a summarizing critical diagram.

For the proposed approach, we propose a rule to automatically select the best result among the 30 results obtained in the 30 repetitions. In particular, similarly to what is done with classic K-means clustering [36], we keep the clustering which leads to the lowest value of the optimization function of the last step (ensemble clustering). Actually, even if using different distances (the squared Euclidean distance, the KL divergence or the Hamming distance) and different representatives (the mean or the mode), all the four clustering schemes minimize the sum of the distances of the objects assigned to each cluster to the relative representative, and this value can be used, similarly to the general K-means case [36], to select in a completely automatic way the best result among a set of repetitions.

Tables 4 and 5 present the comparative analysis; the row “DisRFC(G)” contains the results of the proposed approach, using the variant obtained with the “guidelines” version presented in the previous section. In bold, we put the best result among all approaches for each dataset. For completeness, in the last row (“DisRFC”) we also report the result obtained by the best variant of the proposed approach in each dataset. Results are very promising. In 8 cases over 12 our approach represents the best choice, with some remarkable improvements. In 3 of the remaining 4 cases (ChickenPieces, Newsgroups and TwoPendigits) it still ranks very well among the 20 methods. Only in the FlowCyto1 problem it is not very adequate, being outperformed by several methods. However, this represents a very difficult problem, with very low ARIs, and we probably need a more careful tuning of the parameters to get better results. This can be confirmed by considering the best variant of our approach (reported in the last row – “DisRFC”), which outperforms all the competitors in the FlowCyto1 dataset as well as in the ChickenPieces and TwoPendigits problems.

Table 4
Comparison with alternatives: part 1.

Method	CatC	Prot	CoD	ChiP	News	Fl1
HC-CL	0.1038	0.1654	0.1126	0.3224	0.0041	−0.0380
HC-AvL	0.8734	0.1430	0.0000	0.2318	0.1201	−0.0016
K-centres	0.1021	0.0503	0.0133	0.1956	0.0000	0.0217
K-medoids	0.3587	0.7605	0.0997	0.3086	0.2010	0.0589
kNN-MS	0.1798	0.5061	0.0922	0.3346	0.1647	0.0840
SpectClus	0.7313	0.5429	0.1003	0.0224	**0.3602**	0.0462
AffProp	0.4247	0.7612	0.0843	0.3085	0.2081	0.0656
DomSet	0.2610	0.5940	0.0160	**0.3725**	0.0021	0.0182
CDP	0.4920	0.6891	0.0004	0.1013	0.0000	−0.0063
SP-CDP	0.3972	0.7443	0.0004	0.1013	0.0000	−0.0063
Ker-KM	0.5215	0.8973	0.1429	0.2928	0.3434	0.0519
LocSMK-KM	0.7189	0.8434	0.1672	0.3607	0.0352	0.0577
mdsGMM-D	0.6764	0.8410	0.1504	0.2783	0.3090	0.0374
mdsGMM-F	0.5065	0.4710	0.1071	0.3228	0.2108	0.0612
RFC-Shi	0.3230	0.1211	0.0022	0.0540	0.0082	0.0242
RFC-Zhu2	0.7132	0.7193	0.0555	0.2663	0.3157	0.0573
RFC-Zhu3	0.6138	0.6683	0.1286	0.2743	0.1867	0.0483
RFC-Ayr	0.7775	0.6191	0.0597	0.2714	0.2221	0.0744
RFC-RaRF	0.7332	0.6226	0.0648	0.2917	0.3171	**0.1108**
DisRFC(G)	**0.8758**	**0.9345**	**0.1678**	0.3394	0.3304	0.0457
DisRFC	*0.9107*	*0.9781*	*0.1752*	*0.4204*	*0.3304*	*0.1318*

Table 5
Comparison with alternatives: part 2.

Method	Fl2	WoP	DeGe	Zon	2Pe	Prodom
HC-CL	−0.0592	0.4972	0.4233	0.4142	0.0000	0.0373
HC-AvL	−0.0598	0.2560	0.1867	0.0003	0.0000	0.0362
K-centres	0.0000	0.3062	0.2630	0.0701	0.0864	0.0000
K-medoids	0.0000	0.5631	0.7624	0.2009	0.9861	0.1281
kNN-MS	0.0000	0.2707	0.5443	0.1170	**0.9983**	0.1295
SpectClus	0.0400	0.5705	0.2094	0.7285	−0.0003	0.0080
AffProp	0.0371	0.5801	0.7466	0.1993	0.9861	0.1281
DomSet	0.0000	0.1915	0.2801	0.0423	0.0000	0.0270
CDP	−0.0062	0.0419	0.1581	0.0000	0.5556	0.0000
SP-CDP	−0.0062	0.0419	0.1593	0.0000	0.0000	0.0000
Ker-KM	0.0515	0.5108	0.4577	0.7977	−0.0003	0.0005
LocSMK-KM	0.0125	0.5607	0.4486	0.5380	0.0111	0.0085
mdsGMM-D	0.0660	0.2940	0.3329	0.3301	0.9878	0.1252
mdsGMM-F	0.0498	0.2953	0.3802	0.2276	0.9913	0.1329
RFC-Shi	0.0275	0.0070	0.0076	0.0022	0.0006	0.0039
RFC-Zhu2	0.0681	0.2161	0.2042	0.2055	0.0120	0.1354
RFC-Zhu3	0.0555	0.0957	0.1959	0.1596	0.0101	0.1432
RFC-Ayr	0.0613	0.2453	0.2289	0.2301	−0.0001	0.1243
RFC-RaRF	0.0506	0.2201	0.2530	0.2885	0.0130	0.1987
DisRFC(G)	**0.0777**	**0.5912**	**0.8280**	**0.8003**	0.9895	**0.3688**
DisRFC	*0.1521*	*0.5932*	*0.8317*	*0.8305*	*1.0000*	*0.4416*

5. Conclusions

In this paper we presented DisRFC, the first *dissimilarity-based* Random Forest clustering approach that works only with dissimilarities. The approach is based on the definition of an Unsupervised Dissimilarity Random Forest, a novel variant of RF introduced in this paper, used to embed the objects in a binary space where we obtain the clustering with a K-means style algorithm. We empirically evaluate the proposed scheme using 12 dissimilarities datasets, with promising results, also in comparison with alternative approaches. This approach can be further extended and exploited in different ways: for example, we aim at defining alternative clustering strategies in the binary space, possibly deriving theoretical guarantees about the ensemble learning criterion optimized. Moreover, it would be interesting to study how the approach can be extended to multiview clustering, i.e. to clustering with more than one dissimilarity matrix. Finally, we aim at investigating how the proposed UD-RF can be exploited also in some other non-vectorial unsupervised scenarios, like for example unsupervised outlier detection.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[2] J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers Inc., 1993.
[3] E. Pekalska, R. Duin, The Dissimilarity Representation for Pattern Recognition: Foundations And Applications, World Scientific, 2005.
[4] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Mach. Learn. 63 (1) (2006) 3–42.
[5] D. Huttenlocher, G. Klanderman, W. Rucklidge, Comparing images using the Hausdorff distance, IEEE Trans. Pattern Anal. Mach. Intell. 15 (1993) 850–886.
[6] M. Noshad, K. Moon, S. Sekeh, A.O. Hero, Direct estimation of information divergence using nearest neighbor ratios, in: Proc. Int. Symposium on Information Theory (ISIT), 2017, pp. 903–907.
[7] S. Vega-Pons, J. Ruiz-Shulcloper, A survey of clustering ensemble algorithms, Int. J. Pattern Recognit. Artif. Intell. 25 (3) (2011) 337–372.
[8] A. Topchy, A. Jain, W. Punch, Combining multiple weak clusterings, in: Proc. Int. Conf. on Data Mining (ICDM), 2003, pp. 331–338.
[9] J. Wu, H. Liu, H. Xiong, J. Cao, J. Chen, K-means-based consensus clustering: a unified view, IEEE Trans. Knowl. Data Eng. 27 (1) (2015) 155–169.
[10] U. von Luxburg, A tutorial on spectral clustering, Stat. Comput. 17 (4) (2007) 395–416.
[11] F. Moosmann, B. Triggs, F. Jurie, Fast discriminative visual codebooks using randomized clustering forests, Adv. Neural Inf. Process. Syst. 19 (2006) 985–992.
[12] J. Shotton, M. Johnson, R. Cipolla, Semantic texton forests for image categorization and segmentation, in: Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR), 2008, pp. 1–8.
[13] F. Perbet, B. Stenger, A. Maki, Random forest clustering and application to video segmentation, in: British Machine Vision Conference, BMVC 2009, 2009, pp. 1–10.
[14] M. Bicego, K-random forests: a K-means style algorithm for random forest clustering, in: Proc. Int. Joint Conf. on Neural Networks (IJCNN2019), 2019, pp. 1–8.
[15] F. Liu, K. Ting, Z. Zhou, Isolation forest, in: Proc. of Int. Conf. on Data Mining (ICDM), 2008, pp. 413–422.
[16] D. Yan, A. Chen, M. Jordan, Cluster forests, Comput. Stat. Data Anal. 66 (2013) 178–192.
[17] T. Shi, S. Horvath, Unsupervised learning with random forest predictors, J. Comput. Graph. Stat. 15 (1) (2006) 118.
[18] X. Zhu, C. Loy, S. Gong, Constructing robust affinity graphs for spectral clustering, in: Proc. Int. Conf. on Computer Vision and Pattern Recognition, (CVPR), 2014, pp. 1450–1457.
[19] M. Bicego, F. Cicalese, A. Mensi, RatioRF: anovel measure for random forest clustering based on the Tversky's ratio model, IEEE Trans. Knowl. Data Eng. In press, 2022.
[20] I. Karlsson, P. Papapetrou, H. Bostrom, Generalized random shapelet forests, Data Min. Knowl. Discov. 30 (5) (2016) 1053–1085.
[21] A. Douzal-Chouakria, C. Amblard, Classification trees for time series, Pattern Recognit. 45 (3) (2012) 1076–1091.
[22] S. Sathe, C. Aggarwal, Similarity forests, in: Proc. Int. Conf, on Knowledge Discovery and Data Mining (KDD), 2017, pp. 395–440.
[23] S. Haghiri, D. Garreau, U. von Luxburg, Comparison-based random forests, in: Proc. Int. Conf. on Machine Learning (ICML), 2018, pp. 1871–1880.
[24] B. Lucas, A. Shifaz, C. Pelletier, L. O'Neill, N. Zaidi, B. Goethals, F. Petitjean, G. Webb, Proximity forest: an effective and scalable distance-based classifier for time series, Data Min. Knowl. Discov. 33 (2019) 607–665.
[25] M. Bicego, Dissimilarity random forest clustering, in: Proc. Int. Conf. on Data Mining (ICDM), 2020, pp. 936–941.
[26] R.P.W. Duin, E. Pekalska, Non-euclidean dissimilarities: causes and informativeness, in: Proc. Int. Workshop on Structural, Syntactic, and Statistical Pattern Recognition, 2010, pp. 324–333.
[27] A. Rényi, On measures of entropy and information, in: Proc. of Berkeley Symp. on Mathematical Statistics and Probability, 1961, pp. 467–561.
[28] D. Pál, B. Póczos, C. Szepesvári, Estimation of Rényi entropy and mutual information based on generalized nearest-neighbor graphs, Adv. Neural Inf. Process. Syst. 23 (2010) 1849–1857.
[29] H. Blockeel, L.D. Raedt, J. Ramon, Top-down induction of clustering trees, in: Proc. Int. Conf. on Machine Learning (ICML 1998), 1998, pp. 55–56.
[30] B. Mirkin, Reinterpreting the category utility function, Mach. Learn. 45 (2) (2001) 219–228.
[31] L. Kaufman, P. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, New York, 1990.
[32] Z. Huang, Extensions to the k-means algorithm for clustering large data sets with categorical values, Data Min. Knowl. Discov. 2 (1998) 283–304.
[33] D. Pelleg, A. Moore, X-means: extending k-means with efficient estimation of the number of clusters, in: Proc. Int. Conf. on Machine Learning, 2000, pp. 727–734.
[34] L. Hubert, P. Arabie, Comparing partitions, J. Classif. 2 (1985) 193–218.
[35] S. Aryal, K. Ting, T. Washio, G. Haffari, A comparative study of data-dependent approaches without learning in measuring similarities of data objects, Data Min. Knowl. Discov. 34 (1) (2020) 124–162.
[36] P. Franti, S. Sieranoja, How much can k-means be improved by using better initialization and repeats? Pattern Recognit. 93 (2019) 95–112.

Manuele Bicego is an associate professor at the Computer Science Dept. of the Univ. of Verona (Italy) since 2017. His research interests are in statistical pattern recognition and bioinformatics, e.g. on the probabilistic modelling, representation, clustering and biclustering of biological and medical data. Currently he is very interested in problems related to the exploitation of Random Forests for unsupervised learning. He is author of more than 150 papers, published in international journals, edited books and conferences. He is AE of Pattern Recognition, and PC member of many different international conferences and workshops related to his research interests.