Dissimilarity Random Forest Clustering

Manuele Bicego
Computer Science Dept., University of Verona, Verona, Italy
Email: manuele.bicego@univr.it

***Abstract*—In this paper we present *DisRFC (Dissimilarity Random Forest Clustering)*, a novel Random Forest Clustering approach which, contrarily to current methods which require in input a *vectorial representation*, works only with dissimilarities, thus being applicable also to all those problems where a vectorial representation is not available but a descriptive dissimilarity measure can be computed. In the DisRFC approach objects to be clustered are first modelled with a novel RF variant called *Unsupervised Dissimilarity Random Forest* (UD-RF), which functioning mechanisms are both unsupervised and based on dissimilarities. The trained UD-RF is then used to project objects in a binary vectorial space, where effective K-means procedures can be used to obtain the final clustering. In the paper we present different variants of DisRFC, thoroughly and positively evaluated using 10 different problems.**

***Keywords*-Random Forest clustering, Rényi divergence, Unsupervised Random Forest, Ensemble clustering**

I. Introduction

Random Forests (RFs) [1], [2] represent a renowned pattern recognition and data mining tool, largely and successfully applied in many different contexts. Generally speaking, Random Forests have have been mostly studied for regression and classification, representing in these contexts state-of-the-art approaches. In alternative scenarios, such as clustering, the usefulness of RFs has not been completely investigated, even if some excellent approaches appeared. In particular, more than early works on clustering with trees [3], RF-clustering approaches can be mainly divided into two groups: in the first, RFs (or RF-inspired schemes) are employed to directly derive the clustering [4], [5], [6], while in the second RFs are used to derive a similarity measure to be used as input to a distance-based clustering algorithm [1], [7], [8]. One common limit of all these approaches is that they are designed to work with *vectorial* representations, i.e. objects to be clustered should be represented with a vector of features. However, there are many applications in which it is difficult to derive a set of discriminative features, and objects are more naturally described with *non vectorial* representations, such as strings, sequences, graphs, sets and so on: in these cases classic RF-clustering schemes can not be applied, and novel methods should be devised. It is important to note that, in the classification context, the interest in deriving RF variants able to deal with non vectorial objects has drastically increased in recent years: the most promising and recent methods are based on RFs in which all mechanisms are based on *dissimilarities*[1] computed among non vectorial objects [9], [10], [11], [12]: this represents a very reasonable and general option, since, for these problems, it is very often easier to derive a dissimilarity measure than to extract a set of features [13] – consider for example the powerful distances for strings, sequences or graphs that have been developed in the past.

In this paper we present *DisRFC (Dissimilarity Random Forest Clustering)*, a novel RF-clustering scheme which is entirely based on dissimilarities, thus being applicable also to all those problems where a vectorial representation is not available but a descriptive dissimilarity measure can be computed. The proposed approach is based on two ingredients: the first is a novel variant of RF, called *Unsupervised Dissimilarity Random Forest (UD-RF)*, in which all mechanisms are entirely *unsupervised* and *based on dissimilarities*. We introduced three learning strategies to train a UD-RF, one based on random mechanisms (like in Extremely Randomized Trees [14]), one on Hausdorff distance [15], and one on the Rényi divergence measure [16]. The second ingredient exploits the fact that each tree induces a hierarchical partition of the objects, and tackles the problem of finding the final clustering as a clustering ensemble task. In particular, exploiting and extending recent theoretical advances in K-means based consensus clustering [17], [18], we develop four fast and effective K-means style clustering schemes which operate in a binary space defined, for all objects of the problem, from the trained UD-RF.

All the different variants of the proposed approach have been thoroughly evaluated using 10 different problems involving non vectorial objects; a comparative analysis with classic and advanced dissimilarity based clustering approaches confirms that DisRFC can represent a promising approach to clustering.

II. The proposed approach: DisRFC

The proposed DisRFC (Dissimilarity Random Forest Clustering) approach is based on two steps: i) given dissimilarities between input objects, an Unsupervised Dissimilarity RF (UD-RF) is trained; ii) subsequently, the clustering is obtained by combining the information contained in the different trees of the trained UD-RF via a Consensus Clustering approach.

[1]We will use, along the manuscript, the terms "dissimilarity" and "distance" in an interchangeable way.

2374-8486/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDM50108.2020.00105

A. Unsupervised Dissimilarity Random Forests

The Unsupervised Dissimilarity Random Forest (UD-RF) represents an ensemble of Unsupervised Dissimilarity Trees (UD-T), each one trained using a different random subset of the objects of the problem. Let us introduce the UD-T: it is a *complete* binary tree, which can be built starting from dissimilarities between the objects of a set $\mathcal{O}$. For simplicity, let us assume that we have in input the whole matrix of pairwise dissimilarities $D = [dis(o_i, o_h)]$ $(\forall o_i, o_h \in \mathcal{O})$, where $dis(o_i, o_h)$ represents the dissimilarity between object o_i and object o_h. It is important to note that we do not need to make any assumption on the nature of this dissimilarity, which can be even non Euclidean. In the proposed UD-T, the traversal mechanism is borrowed from Proximity Forests [12], one of the dissimilarity-based forests introduced in the classification field. In particular, a node n_j of a UD-T is defined by two prototypes o_j^L and o_j^R, associated to the left and the right branches, respectively: an object o follows the left path if it is more similar to the left prototype (i.e $dis(o, o_j^L) < dis(o, o_j^R)$), the right path otherwise. Let us call S_j the set of objects reaching node n_j, and S_j^L (S_j^R) the set of nodes going through the left (right) branch of node n_j.

The UD-T is trained starting from a set of objects: we start from the root (which contains all objects), define a split and create two children nodes, containing the points nearest to the left and to the right prototypes, respectively; then we continue with the splitting procedure until a node can not be split further, or until the tree has reached a predefined maximum depth. To choose the pair of prototypes which determine the split inside a node, we propose three possible schemes, of increasing complexity. Please remember that, in the clustering case, no labels are available[2].

"Rand": in this first scheme the two prototypes in a node n_j are randomly chosen among the objects reaching node n_j (i.e. $o_j^L, o_j^R \in S_j$), similarly to what is done in Extremely Randomized Trees for classification [14].

"HausD": this second scheme is based on the idea that a good split should produce two sets of objects which are *well separated*, in order to get a better partition of the space. Here, to measure the separation between S_j^L and S_j^R we use the Hausdorff distance [15], a classic measure to compare sets of objects, which – crucially – can be computed starting only from pairwise distances. Given two sets $X = \{x_1, \cdots x_N\}$ and $Y = \{y_1 \cdots y_M\}$, and a distance measure $dis(x, y)$ between objects, the Hausdorff distance between X and Y is defined as:

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

[2]Some approaches for RF-clustering like [7], [8] circumvented the problem by sampling a negative class from the vector space – here this solution can not be adopted since we have only dissimilarities.

In our "HausD" training strategy, at each node n_j the best pair is the one which maximizes the Symmetrized Hausdorff distance between S_j^L and S_j^R, defined as:

$$SHD(S_j^L, S_j^R) = \frac{1}{2}\left(HD(S_j^L, S_j^R) + HD(S_j^R, S_j^L)\right) \quad (2)$$

Clearly, analysing all possible pairs would be computationally too demanding, therefore we use a classic RF trick: we randomly selected a small number of random pairs, choosing the one which maximizes eq. (2).

"RényiD": this scheme is based on the idea that a good split should produce two sets of points which *convey different information*. To quantify this, we can exploit Information Theory measures, as extensively done in many other RF training schemes [1], [2]. In particular, in the "RényiD" training scheme we search for the pair of objects leading the largest Rényi divergence [16] between S_j^L and S_j^R. The Rényi divergence represents an information theoretic measure which generalises the Kullback-Leibler divergence between probability distributions; to estimate it we use the recent method proposed in [16], which is particularly suited for our case for different reasons: i) it is non parametric, i.e. it does not make any assumption on the underlying distribution; ii) it belongs to the class of *by-pass estimators* [19], i.e. it does not need to directly estimate the density (thus avoiding all the problems typically involved in density estimation, like the computation of the partition function); iii) it is based on nearest neighbor ratios, thus relying only on dissimilarities; iv) it produces reliable estimates also with non Euclidean distances, thus being suitable for our scenario.

More in details, given two sets $X = \{x_1, \cdots x_N\}$ and $Y = \{y_1 \cdots y_M\}$, the Rényi divergence is estimated as:

$$RD(X, Y) = \frac{1}{(\alpha - 1)} \log \left[\frac{\eta^\alpha}{M} \sum_{i=1}^{M} \left(\frac{N_i}{M_i + 1} \right)^\alpha \right] \qquad (3)$$

where α is the parameter of the Rényi divergence, $\eta = M/N$, N_i (M_i) represents the number K-nearest neighbors of y_i in $\{X \cup Y\}$ belonging to X (Y) – for all the technical details see [16]. Given this definition, in the "RényiD" training strategy the best pair is the one which maximizes the Symmetrized Rényi divergence between S_j^L and S_j^R, defined as:

$$SRD(S_j^L, S_j^R) = \frac{1}{2}\left(RD(S_j^L, S_j^R) + RD(S_j^R, S_j^L)\right) \quad (4)$$

Also in this case, to maintain the computational load to a reasonable level, we choose the best pair among a small number of randomly selected pairs.

B. Ensemble clustering with UD-RF

Given the trained UD-RF, which provides a description of the objects of the problem, the second step is aimed at obtaining the final clustering. We start from an idea already introduced in early RF-clustering approaches [3], [4], [5]:

each tree can be considered as a partitioner, since it permits to divide the input objects into different (hierarchical) clusters, according to the path each object follows in the tree. Assuming this perspective, we can get the final clustering from a UD-RF by aggregating the partitions defined by the different trees contained in it, i.e. by using approaches of Consensus Clustering. More specifically, we propose a scheme for combining the information contained in the different trees which has its roots in the works of [20], [17]. In such papers, authors show that it is possible to solve the demanding combinatorial optimization problem of maximizing the Category Utility Function – a widely known criterion for consensus clustering [20] – via a classic K-means algorithm in a specific feature space defined by the partitions. This elegant idea has been further extended in [18]: authors show that, under some conditions, many other different criteria for consensus clustering can be seen from this perspective, typically by changing the distance used inside the K-means. Taking inspiration from these findings, in our approach we embed the objects of the problem in a space derived from the trained UD-RF, getting the final result via clustering in such space.

Embedding with UD-RF. We define two possible embeddings, both based on the partitions induced by the trees: in particular, the "OP" (One Partition) embedding uses a single partition for each tree, whereas the "MP" (More Partitions) embedding uses more than one partition, thus better exploiting the whole hierarchy defined by the tree at the price of increasing the dimensionality of the embedding. In both cases the final embedding of an object o is obtained by concatenating the T embeddings of o in the T trees. Let us present the two embeddings for a single tree. Let $\{n_1, \cdots, n_N\}$ be the nodes of the tree, $\mathcal{P}(o)$ the set of nodes passed by the object o when falling down the tree (i.e. the path), and $\ell(o)$ the leaf reached by o. Given a depth value d, we define $\mathcal{P}_d(o)$ as the set of nodes in the path of o which depth is less than or equal to d, i.e. the path of o up to the depth d:

$$\mathcal{P}_d(o) = \{n_j \in \mathcal{P}(o)) | dep(n_j) \leq d\},\ (1 \leq d \leq d_{max}) \quad (5)$$

where $dep(n_j)$ is the depth of node n_j, and d_{max} is the maximum depth of the tree. We also define $c_d(o)$ as the deepest node in $\mathcal{P}_d(o)$:

$$c_d(o) = \arg \max_{n \in \mathcal{P}_d(o)} dep(n) \quad (6)$$

Now, we can define a cluster at depth d as a set containing all the points reaching the same node at depth d, i.e. objects with the same $c_d(o)$. Please note that the definition given in [4], [5], [21], in which clusters were defined as containing objects falling in the same leaf, is a special case of our definition, which also permits to decide at which depth to consider the partition – useful for example if we want to fix the number of clusters.

We are now ready to define the two embeddings. Given the set of objects $\mathcal{O}$, and given a depth d, let $\{\tilde{c}_d^1, \cdots \tilde{c}_d^{K_d}\}$ be the set of K_d unique $c_d(o)$ for all $o \in \mathcal{O}$, and let $\mathcal{C}_d$ be the associated partition:

$$\mathcal{C}_d = \{\mathcal{C}_d^1, \cdots \mathcal{C}_d^{K_d}\},\ \mathcal{C}_d^k = \{o_j \in \mathcal{O} | c_d(o_j) = \tilde{c}_d^k\} \quad (7)$$

The OP embedding (One Partition) of an object o, at depth d, is defined as:

$$E^{OP_d}(o) = \left[e_1^{OP_d}(o), e_2^{OP_d}(o), \cdots e_{K_d}^{OP_d}\right] \quad (8)$$

where $e_j^{OP_d}(o)$ is 1 if $c_d(o) = \tilde{c}_d^j$, i.e. $E^{OP_d}(o)$ is a vector with all zeros except in the position of the node reached by o at depth d. It is easy to show that the OP embedding corresponds exactly to the encoding defined in [20], [17], [18] when assuming that the input partitions are those defined by $\mathcal{C}_d$ (one for each tree of the forest).

The MP embedding (More Partitions) is a generalization of the OP embedding, and exploits the fact that a tree does not realize a single partition, but a whole hierarchy of partitions. In particular we propose to consider more than one partition: by considering a set of H depths $\{d_1, \cdots d_H\}$, the MP embedding is obtained by concatenating the OP embeddings for each depth in $\{d_1, \cdots d_H\}$. This permits to exploit the information at different levels of granularity, i.e. from partitions with few clusters up to more detailed groupings. Please note that also the MP embedding can be seen from the perspective of [20], [17], [18]: actually, we simply have more input partitions, with different number of clusters; since the findings of [20], [17], [18] do not require a fixed number of clusters, we are still guaranteed to have a proper embedding.

Clustering. Following [20], [17], [18] the final clustering can be obtained by running a K-means in the properly defined embedding space. Here we investigated two different clustering schemes based on this paradigm: the first ("KM-SED"), uses K-means with squared Euclidean distances, and corresponds to the method proposed in [17]; the second ("KM-KL") uses K-means with the KL divergence, and corresponds to the optimization of a Shannon-entropy derived consensus clustering criterion [18]. We also generalizes the approach, by using, in the embedding space, two variants of K-means which are classically used for categorical clustering: the K-medoids algorithm [22] with the Hamming distance ("K-medoids") and the K-modes algorithm [23] ("K-modes"); even if lacking the theoretical guarantee of optimizing a consensus clustering criterion, they share the same idea of clustering in the embedding space – in practical scenarios they may represent good alternatives (see the experimental session).

III. Experimental evaluation

This section presents the experimental evaluation. As commonly done in clustering, the evaluation is performed

Table I
DATASETS EMPLOYED IN THE EMPIRICAL EVALUATION (N: OBJECTS, K: CLUSTERS, MIN/MAX: SMALLEST AND LARGEST CLUSTER).

Problem	Description	N	K	Min/Max
CatCortex	Connections between cat's cortical areas	65	4	10/19
Protein	Evolutionary distances of protein sequences	213	4	30/72
CoilDelft	Spectral distances between graphs of 4 COIL objects	288	4	72/72
ChickenPieces	Weighted Edit distances between 2D objects contours	446	5	61/117
FlowCyto	L1 norms between flow-cytometer histograms	612	3	131/335
WoodyPlants	Shape dissimilarities between plant leaves	791	14	51/66
Delftgestures	DTW distances between video gesture signs	1500	20	75/75
Zongker	Deform. template matching distances between digits	2000	10	200/200
TwoPendigits	Edit distance between contour sequences of two digits	2287	2	1143/1144
Prodom	Structural alignments between protein domains	2604	4	271/1051

using supervised datasets: after removing labels, we determine the clusters, comparing them with the original classes. We quantified the quality of the results with a classical measure, the adjusted Rand index (ARI – [24]). In our experiments we used 10 problems[3], briefly summarized in Table I. Each problem comes as a dissimilarity matrix, which contains the pairwise dissimilarity between all objects of the problem: please note that in all cases (except FlowCyto) the input objects are described with non vectorial representations (such as strings, graphs, sequences or 3D structures).

For what concerns the proposed approach, we evaluated all possible variants, i.e. all possible combinations of the training procedures ("Rand", "HausD", and "RényiD"), embeddings ("OP" and "MP"), and clustering ("KM-SED", "KM-KL", "K-medoids" and "K-modes"). We used forests with 100 and 200 trees, training each tree with a rather small number of objects (128 in all our experiments except CatCortex – where we used 65). Each tree was completely grown until a node contains less than 10 points. For the "Rand" training strategy, the splitting pair inside a node is randomly chosen among all possible pairs composed by the training objects reaching that node – we discarded invalid pairs, i.e. pairs which do not permit to split data of the node in two non empty sets; for the "HausD" and the "RényiD" strategies, the selected pair is the one, among 20 valid random pairs, which maximizes eq. (2) and (4), respectively.

[3]All available from `http://37steps.com/distools/`. For ChickenPieces we used the version with NORM = 40 and COST = 120; for FlowCyto we used the first version; for TwoPendigits we selected the first and the third digits from the Pendigits dataset; finally, Zongker and Prodom were converted from similarities to dissimilarities using the `distools` routine `dissimt(S,'sim2dis')`.

To compute the Rényi divergence we set $\alpha = 0.999$ and $K = \sqrt{n}$ (as suggested in [16]), where n is the number of the objects in the training set. For what concerns the embeddings, for "OP" we used the partition defined by the leaves, i.e. $d = d_{max}$ in eq. (8); for "MP" we used the partition defined by the leaves plus two other partitions equally spaced in the hierarchy (i.e. $H = 3$): this permits to better exploit the information contained in the different trees still maintaining a reasonable dimension of the embedding space. Finally, for all K-means-like clustering schemes, we performed 20 random initializations of the labelling, keeping the clustering which minimizes the objective function. For each dataset and each configuration (training, embedding, clustering), we repeated the experiments 30 times.

A. Results

In order to have a direct comparison between the different aspects of the DisRFC approach, we present in Table II only a set of summarizing results, useful to compare the different options relative to each of the three steps (training, embedding, clustering). In particular, for each step, we compute the average of the ARI values of the different alternatives by varying all other aspects: for example, when analysing the embedding, we compute the average of all results obtained with OP (and MP) for all different parameters, all trainings, all clusterings, and all 30 runs, thus resulting, for each dataset, in 720 values (2 parametrizations $\times$ 3 training $\times$ 4 clustering $\times$ 30 repetitions). In the table, for each aspect and each dataset, we highlight in bold the best result. We also perform an unpaired t-test (significance level 0.05) between the ARI values of the best option and those of the second best: in the table, an asterisk indicates when their difference is not statistically significant.

Looking at the results, for the training we can infer that the RényiD variant represents the best option, with many datasets in which the average ARI is higher than that of the other two options – in some (Protein, CoilDelft and Prodom), this difference is remarkably high. In few cases (WoodyPlants, Delftgestures and Zongker) the RényiD variant is worst than the others: interestingly, these problems are all characterized by a large number of clusters (14, 20 and 10); probably, these clusters can not be organized in a hierarchy, and with more accurate hierarchies (as those derived by using the RényiD scheme) this misalignment becomes more relevant. It is also interesting to observe that the Rand version works almost everywhere reasonably well, confirming the findings obtained in other scenarios [14]. Regarding the embedding, in most of the cases OP and MP are more or less equivalent, except in the TwoPendigits case, where MP drastically outperforms OP. Finally, concerning the clustering schemes, KM-SED and K-modes represent almost always the best option.

From all these results we distilled some guidelines, to be used to practically choose a variant for a given problem: i)

Table II
ANALYSIS OF DIFFERENT ASPECTS OF THE PROPOSED APPROACH. "TOTAL" IS THE NUMBER OF EXPERIMENTS OVER WHICH THE AVERAGE IS TAKEN.

	Training *(Total: 480)*			**Embedding** *(Total: 720)*		**Clustering** *(Total: 360)*			
Problem	**Rand**	**HausD**	**RényiD**	**OP**	**MP**	**KM-SED**	**KM-KL**	**K-medoids**	**K-modes**
CatCortex	0.6189	0.6362	**0.6670**	**0.6408***	0.6406	0.7338	0.2615	0.7719	**0.7956**
Protein	0.6268	0.5595	**0.8592**	0.6258	**0.7379**	0.7380	0.4313	0.7462	**0.8119**
CoilDelft	0.0930	0.0865	**0.1601**	**0.1150***	0.1115	**0.1490**	0.0619	0.1105	0.1315
ChickenPieces	0.3114	0.2805	**0.3351**	**0.3118**	0.3062	**0.3250**	0.2910	0.3127	0.3074
FlowCyto	**0.0792**	0.0634	0.0557	0.0658	**0.0664***	0.0717	**0.0768**	0.0520	0.0637
WoodyPlants	**0.5363**	0.5277	0.4514	**0.5108**	0.4995	0.5301	0.4247	**0.5352***	0.5306
Delftgestures	0.7099	**0.7120***	0.6225	**0.6951**	0.6678	0.7500	0.5089	**0.7515***	0.7155
Zongker	**0.7312***	0.7197	0.6765	**0.7219**	0.6964	0.7473	0.5780	0.7406	**0.7707**
TwoPendigits	0.9110	0.8631	**0.9696**	0.8658	**0.9633**	0.9791	0.9707	0.7263	**0.9821***
Prodom	0.1507	0.1064	**0.3061**	**0.1998**	0.1757	0.1808	0.1843	0.1735	**0.2123**
Average	0.4768	0.4555	**0.5103**	0.4753	**0.4865**	0.5205	0.3789	0.4921	**0.5321***

when the dataset is large enough (e.g. large than 128), and the number of clusters is low (e.g. less than 10), we suggest to use the RényiD training procedure; in the other cases we suggest to use the Rand one; ii) as embedding, we suggest the OP one for easy problems (i.e. problems with an average ARI larger than 0.5), and to leave the more computationally demanding MP option for complex problems; iii) finally, as clustering scheme we suggest to use KM-SED, since more theoretically sound than K-modes. We will use these guidelines in the comparative analysis of the next section.

B. Comparison with alternatives

In this section we compare the performances of the proposed approach with those obtained using some other distance-based clustering schemes. In particular we considered both simple and advanced schemes, as briefly described in the following (we also report a link to the implementation we used – the details can be found there). We used: i) *HC-SL* the Single Link, ii) *HC-CL* the Complete Link and iii) *HC-AvL* the Average Link variants of the agglomerative hierarchical scheme (as implemented in the Matlab `linkage` function); iv) *DK-Cen* and v) *DK-Med*, two dissimilarity versions of K-means (as implemented in the Matlab ClusterTools package[4]); vi) *kNN-MS*, a KNN-mode seeking scheme based on distances [25] (again in the implementation on the Matlab ClusterTools package); vii) *SpectClus*, the Spectral Clustering method, using the Ng-Jordan-Weiss normalized version [26] (we used a web implementation[5]); viii) *AffProp*, the Affinity-Propagation algorithm (we used the authors version which allows setting the number of clusters[6]); ix) *DomSet*, the Dominant Set Clustering [27] (as implemented in the DominantSetLibrary[7], leaving all parameters as default).

All the results are shown in Table III, together with the results of the proposed approach. In particular, for DisRFC, for each variant and each dataset, among the 30 repetitions we kept the result which leaded to the lowest value of the optimization function of the last step (ensemble clustering). The column titled "DisRFC" reports the result of the best variant obtained by the proposed approach in each dataset. We can observe that the DisRFC approach outperforms all competitors, with improvements which are remarkable in some cases (0.2169 of improvement in Protein, 0.1020 in Zongker, 0.3121 in Prodom). In the last column ("DisRFC (Guideln)") we report results obtained using the variant of DisRFC derived from the guidelines presented in the previous section. As can be seen, also in this case the results are satisfactory, with "DisRFC (Guideln)" outperforming all the competitors in 7 cases out of 10. In two cases, it represents the second top performing method (outperformed by Dominant Set Clustering in the ChickenPieces and by kNN mode seeking in the TwoPendigits problem). In the last case, FlowCyto, it is outperformed by 3 techniques (K-medoids, KNN mode seeking and Affinity Propagation): however, this represents the most difficult problem, with very low ARI values, and probably a more careful tuning of parameters is needed to get better results (e.g. the best variant uses the HausD training and the MP embedding).

IV. CONCLUSIONS

In this paper we introduced DisRFC, a novel RF-clustering method which functioning mechanisms are all based on dissimilarity operations, thus being suitable when objects to be clustered do not have a proper vectorial representation, but a descriptive dissimilarity measure can be defined. We presented and positively evaluated different variants of the proposed approach, also providing some guidelines useful to choose the best variant according to the characteristics of the problem.

REFERENCES

[1] L. Breiman, "Random forests," *Mach. Learn.*, 45: 5–32, 2001.

[4] `http://37steps.com/software/`

[5] https://github.com/areslp/matlab/blob/master/spectral_clustering/ SpectralClustering.m

[6] See `http://www.psi.toronto.edu`

[7] `https://github.com/xwasco/DominantSetLibrary`

Table III
COMPARISON WITH ALTERNATIVE APPROACHES. SEE THE TEXT FOR THE ACRONYMS.

Problem	**HC-SL**	**HC-CL**	**HC-AvL**	**DK-Cent**	**DK-Med**	**kNN-MS**	**SpectClus**	**AffProp**	**DomSet**	**DisRFC**	***DisRFC (Guideln)***
CatCortex	0.0160	0.1038	0.8734	0.1021	0.3587	0.1798	0.7313	0.3643	0.2610	**0.9107**	*0.8758*
Protein	0.0710	0.1654	0.1430	0.0000	0.7605	0.5061	0.5429	0.7612	0.5940	**0.9781**	*0.9345*
CoilDelft	0.0000	0.1126	0.0000	0.0133	0.0997	0.0922	0.1003	0.0843	0.0160	**0.1748**	*0.1746*
ChickenPieces	0.0187	0.3224	0.2318	0.1956	0.3086	0.3346	0.0224	0.3085	0.3725	**0.4204**	*0.3409*
FlowCyto	-0.0063	-0.0380	-0.0016	0.0217	0.0589	0.0840	0.0462	0.0656	0.0182	**0.1280**	*0.0462*
WoodyPlants	-0.0001	0.4972	0.2560	0.3062	0.5631	0.2707	0.5859	0.5801	0.1915	**0.5912**	*0.5912*
Delftgestures	0.1129	0.4233	0.1867	0.2630	0.7624	0.5443	0.2094	0.7466	0.2801	**0.8317**	*0.8280*
Zongker	0.0000	0.4142	0.0003	0.0000	0.0000	0.1170	0.7285	0.1993	0.0423	**0.8305**	*0.8003*
TwoPendigits	0.0000	0.0000	0.0000	0.0864	0.9861	0.9983	-0.0003	0.9861	0.0000	**1.0000**	*0.9895*
Prodom	0.0026	0.0373	0.0362	0.0000	0.1281	0.1295	0.0080	0.1281	0.0270	**0.4416**	*0.3697*

[2] A. Criminisi, J. Shotton, and E. Konukoglu, "Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning," *Foundations and Trends in Computer Graphics and Vision*, 7(2-3):81–227, 2012.

[3] H. Blockeel, L. D. Raedt, and J. Ramon, "Top-down induction of clustering trees," in *ICML*, 1998, pp. 55–63.

[4] F. Moosmann, B. Triggs, and F. Jurie, "Fast discriminative visual codebooks using randomized clustering forests," in *NIPS*, 2006, pp. 985–992.

[5] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *CVPR*, 2008.

[6] M. Bicego, "K-random forests: a K-means style algorithm for random forest clustering," in *IJCNN*, 2019, pp. 1–8

[7] T. Shi and S. Horvath, "Unsupervised learning with random forest predictors," *J. of Comp. and Graph. Stat.*, 15(1): 118–138, 2006.

[8] X. Zhu, C. Loy, and S. Gong, "Constructing robust affinity graphs for spectral clustering," in *CVPR*, 2014, pp. 1450–1457.

[9] S. Balakrishnan and D. Madigan, "Decision trees for functional variables," in *ICDM*, 2006, pp. 798–802.

[10] A. Douzal-Chouakria and C. Amblard, "Classification trees for time series," *Patt. Rec.*, 45(3): 1076–1091, 2012.

[11] S. Sathe and C. Aggarwal, "Similarity forests," in *KDD*, 2017, pp. 395–403.

[12] B. Lucas, A. Shifaz, C. Pelletier, L. O'Neill, N. Zaidi, B. Goethals, F. Petitjean, and G. Webb, "Proximity forest: an effective and scalable distance-based classifier for time series," *Data Mining and Knowl. Disc.*, 33: 607–635, 2019.

[13] E. Pekalska and R. Duin, *The Dissimilarity Representation for Pattern Recognition: Foundations And Applications*. World Scientific, 2005.

[14] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, 63(1): 3–42, 2006.

[15] D. Huttenlocher, G. Klanderman, and W. Rucklidge, "Comparing images using the hausdorff distance," *TPAMI*, 15: 850–863, 1993.

[16] M. Noshad, K. Moon, S. Sekeh, and A.-O. Hero, "Direct estimation of information divergence using nearest neighbor ratios," in *ISIT*, 2017, pp. 903–907.

[17] A. Topchy, A. Jain, and W. Punch, "Combining multiple weak clusterings," in *ICDM*, 2003, pp. 331–338.

[18] J. Wu, H. Liu, H. Xiong, J. Cao, and J. Chen, "K-means-based consensus clustering: A unified view," *TKDE*, 27(1): 155–169, 2015.

[19] D. Pál, B. Póczos, and C. Szepesvári, "Estimation of rényi entropy and mutual information based on generalized nearest-neighbor graphs," in *NIPS*, 2010, pp. 1849–1857.

[20] B. Mirkin, "Reinterpreting the category utility function," *Mach. Learn.*, 45(2): 219–228, 2001.

[21] F. Perbet, B. Stenger, and A. Maki, "Random forest clustering and application to video segmentation," in *BMVC*, 2009, pp. 1–10.

[22] X. Jin and J. Han, "K-medoids clustering," in *Enc. of Machine Learning*, C. Sammut and G. Webb, Eds. Springer, 2011.

[23] Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," *Data Mining and Know. Disc,*, 2: 283–304, 1998.

[24] L. Hubert and P. Arabie, "Comparing partitions," *J. of Classification*, pp. 193–218, 1985.

[25] R. Duin, A. Fred, M. Loog, and E. Pekalska, "Mode seeking clustering by KNN and mean shift evaluated," in S+SSPR, 2012, pp. 51–59.

[26] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[27] M. Pavan and M. Pelillo, "Dominant sets and pairwise clustering," *TPAMI*, 29(1): 167–172, 2006.