K-Random Forests: a K-means style algorithm for Random Forest clustering

Manuele Bicego
Computer Science Department, University of Verona, Verona, Italy
manuele.bicego@univr.it

***Abstract*—In this paper we present a novel clustering approach based on Random Forests, a popular classification and regression technique whose usability in the clustering scenario has been investigated to a lesser extent. In the clustering context, the most used class of approaches is based on the exploitation of a single Random Forest to derive a proximity measure between points, to be used with any distance-based clustering technique. On the contrary, our scheme exploits *a set* of Random Forests, each one devoted to model one cluster, in a spirit similar to that of the mixture models approach. These Random Forests, which provide flexible cluster descriptors, are iteratively updated using a K-means-like clustering algorithm. The proposed scheme, which we call *K-Random Forests (K-RF)*, has been evaluated on five datasets: the obtained results suggest that it represents a valid alternative to classic Random Forest clustering algorithms as well as to other established clustering approaches.**

I. Introduction

Random Forests [1]–[3], firstly introduced by Breiman in the early 2000s [1], represent a famous and widely applied approach for Pattern Recognition, whose usefulness has been demonstrated in a huge variety of application fields – bioinformatics [4], biometrics [5], remote sensing [6], computer vision [7] and medical image analysis [8], just to cite a few. This technique represents an ensemble of decision trees [9], [10], a widely known classification tool which realizes a hierarchical splitting of the feature space, each split being based on a threshold on a single feature. Random Forests are based on the "divide and conquer" principle: in their classic version, different sets of random samples are extracted from the training set, each one used to build a decision tree. The final model is then obtained by aggregating the different trees. This aggregation shows interesting theoretical properties, the most famous being the one shown by Breiman in [1], who defined an upper bound on the generalization error of the Random Forests in terms of correlation and strengths of individual trees. The key aspect in Random Forests is represented by the randomization injected in the learning of the model, which permits to have strongly diverse trees: this randomization can be driven to the extreme, for example by randomly choosing also the splitting threshold – these aspects have been largely investigated by Geurts and colleagues in their work on Extremely Randomized trees [11].

Generally speaking, Random Forest approaches have been almost always studied as regression or classification tools: in these contexts they represent state-of-the-art machines, able to compete with the most effective and established approaches (like SVM or Neural Networks). Their exploitation in other pattern recognition contexts, such as clustering, density estimation, one-class classification and others, has received less attention. Clearly, excellent works in these scenarios exist: for example, Criminisi and colleagues provided in [3] a unified view of Random Forests (called by them Decision Forests) usable for classification, regression, density estimation, manifold learning and semi-supervised learning; moreover, specific Random Forest models have been also proposed for ranking problems (Ranking Forests – [12]), survival analysis (Random survival forests – [13]), one-class classification (One-class Random Forests – [14], [15]) and multi-label classification [16], just to cite a few.

This paper is inserted in the above described research stream, and proposes a novel approach for the scarcely investigated scenario of clustering using Random Forests. In this context, the most established class of approaches [17]–[19] exploits a Random Forest to derive a similarity between points: given two points, their similarity is obtained by counting how many times they fall in the same leaf of a tree of the forest with respect to the total number of trees. Given the proximity, any distance-based clustering approach (such as Spectral Clustering [20]) can be used. This basic scheme has been extended in recent years in different ways (e.g. [21]), providing excellent performances in many different tasks [22]–[26]. However, these approaches almost all maintain the same basic idea: to exploit a trained Random Forest to get a proximity measure, used as input to a distance-based clustering algorithm. Even if few other interesting methods have been proposed (see Section II), clustering with Random Forests remains an open problem.

In this paper we make one step forward along this direction, proposing a novel approach for clustering based on these tools. In particular, the basic idea is that, instead of building a single Random Forest to describe *the whole problem*, we can use a single Random Forest to describe *every cluster*, in a spirit similar to that of finite mixture models [27]: thanks to the flexibility of Random Forests, we can characterize every cluster with a flexible and accurate boundary. Since we are interested in modelling a given cluster without considering the rest of the patterns, we use *One-class Random Forests* [14], [15], [28], [29], i.e. Random Forests able to be trained

M. Bicego was partially supported by the University of Verona through the program "Bando di Ateneo per la Ricerca di Base 2015" and "Programma di internazionalizzazione di Ateneo 2018 - Azione 4, categoria B"

978-1-7281-2009-6/$31.00 paper N-19210.pdf

with only positive examples[1]. We devise a simple iterative strategy, which takes inspiration from the classic K-means algorithm [30] or from some of its extensions [31], [32]. In particular, at every iteration we assign every point to the One-class Random Forest which most accurately models that point (similarly, in the K-means, we assign a point to the cluster with the nearest mean); subsequently, we re-train each One-class Random Forest using only points assigned to it (in K-means, we re-compute each cluster mean using only points assigned to it). Starting from a random assignment, these two steps are repeated until convergence. Here we employed as one-class forests the so called Isolation Forests [28], [29], one of the best methods for one-class classification (i.e. anomaly detection) according to [33]. In its basic original form[2] this method builds a set of Extremely Randomized Trees [11]; then, given an input point, the corresponding "anomaly score" can be computed by looking at the length of the path the point has to follow to reach its leave, averaged over all trees. The intuition is that outliers will fall in the upper part of every tree, since there is no need for many splits to isolate that point: as a result the averaged path should be shorter.

The proposed approach, which we call *K-Random Forests (K-RF)*, has been preliminary evaluated on five different standard datasets from the UCI-ML repository, investigating the effect of the different parameters (such as the number of trees). We compare our approach with the classic Random Forest approach for clustering [17], with encouraging results. A further comparison with alternative clustering methods suggests that the proposed scheme represents a viable alternative to classical as well to advanced clustering algorithms. The remainder of the paper is organized as follows: in Section II we summarize the related work, whereas in Section III we describe the proposed scheme. The experimental evaluation is then presented in Section IV; finally Section V concludes the paper.

II. Related work

As reported in the introduction, the exploitation of Random Forests in the clustering scenario has received little attention. The most famous and widely applied class of approaches extends early works on clustering trees [34], [35], and proposes to employ a Random Forest to derive a proximity measure between points [17]–[19]. More in detail, the main idea is to create a Random Forest predictor, and to run all the training objects down each tree: reasonably, points which are nearby in the feature space will fall on the same leave. We can therefore obtain a similarity between objects by counting how many times this happens, i.e. in how many trees x_i and x_j fall in the same leaf (normalized by the total number of trees). Given the proximity, any distance-based clustering approach (such as Spectral Clustering [20]) can be used. The Forest used to derive the proximity can be built in different ways: the most common solution is to use a classification forest which discriminates between the original data and a synthetically generated negative class [17], typically obtained by sampling points from the product of empirical marginal distributions of the observed data (this permits to remove the dependency structure of the original data). Another option is exploit Extremely Randomized Trees [11] enriched with some aside information which can be available (e.g. in [18], [19], the goal is to learn a visual dictionary – a clustering problem – for image classification or categorization, for which labels are available and can be exploited). This scheme has been recently extended to get even more robust similarity matrices for spectral clustering [21]. The obtained proximity seems to be very attractive: among other properties, it is invariant with respect to monotonic transformations of the variables and can deal with high dimensional problems. Actually the scheme, in its basic and extended forms, has shown excellent results in many different applications, like analysis of tumor marker data [24], [25], genomic data analysis [22], chemometrics [23] or computer vision [18], [19], [21].

Some other approaches appeared in recent years, all more or less characterized by the exploitation of a Random Forest-like ensemble mechanism (more than by the exploitation of an actual Random Forest). One example is [26], where the video segmentation problem is faced by getting multiple partitions using different clustering trees, partitions which are then fused together via intersection to get a more robust result. Another very interesting example is [36], which, similarly to what Random Forests do for classification, extracts random projections on which good local clusterings are found, to be aggregated to get the final result. In this sense, also this represents an ensemble clustering method.

Finally, it is worth to note that a clustering can be also derived from a density estimation procedure (e.g. via mode seeking); interestingly, in recent years, there has been few attempts to derive Random Forests for density estimation [3], [37]: to the best of our knowledge, however, their capabilities in the clustering context have never been investigated.

III. The K-RF approach

In this section the proposed scheme is presented. In particular, we will first summarize the Random Forest model used to characterize each cluster; subsequently we will introduce the iterative clustering procedure.

A. The cluster model

In our approach every cluster is described with a One-class Random Forest, and in particular with a slight modification of the so-called Isolation Forest [28], [29], a Random Forest approach largely and successfully employed for anomaly detection [33]. This scheme, after building a set of Extremely Randomized Trees [11], derives an *anomaly score* which is inversely proportional to the average length of the path a point has to travel to reach a leave. The intuition is that anomalies are typically few and with feature values which are very different from those of normal instances: therefore, as written in [28], "anomalies are more likely to be isolated

[1] An alternative not investigated here is to consider classic Random Forests; in this case, we can use as negative class the points of the other clusters.

[2] A recent extension has been proposed in [15].

closer to the root of the tree, whereas normal points are more likely to be isolated at the deeper end of the tree". Given the forest, we can exploit this property to derive a *membership score* which describes how well a given point is modelled by the forest.

More precisely, an Isolation Forest is built by learning M trees, each one trained using a random fraction ρ ($0 < \rho < 1$) of the available patterns. At each node, a split is generated by i) randomly selecting a feature and ii) randomly selecting a split value between the minimum and the maximum value of that feature. In [28], [29], since they are interested in modeling the anomalies (which should be in the upper part of the trees), the tree is grown only until it reaches the expected averaged length $\log_2(n)$ (with n the number of objects in the training set). Here we are interested in the "inliers", therefore we build the tree until the end, i.e. when i) each node contains one element or ii) it contains data with the same value.

Once learned the forest, authors in [28], [29] propose to characterize the anomaly of a given point x with the so-called *anomaly score* $s(x, n)$. This score is computed by running the point x down each tree and computing the number of the edges the point has to travel to reach a terminal node (the so called path length of the point $h_t(x)$ – for a given tree t)[3]. In particular, given the set of $\{h_t(x)\}$ computed for all trees of the forest RF (built using n points), and given its average $E(h_t(x))$, the anomaly score is computed as:

$$s_{RF}(x, n) = 2^{-\frac{E(h_t(x))}{c(n)}} \tag{1}$$

where $c(n)$ is a normalization function defined as:

$$c(n) = 2H(n-1) - (2(n-1)/n) \tag{2}$$

with $H(n)$ the Harmonic number, estimated with $ln(n) + 0.5772156649$ (Euler's constant).

In our approach, we reverse the reasoning, and evaluate how well a given forest RF describes an input point x by computing the following *membership score*:

$$w_{RF}(x) = 1 - 2^{-\frac{E(h_t(x))}{c(n)}} \tag{3}$$

(i.e. $w_{RF}(x) = 1 - s_{RF}(x, n)$), where we dropped the dependence on n for clarity of presentation.

B. The clustering procedure

The K-RF algorithm is described in Figure 1. In particular, the procedure takes in input the points $X = \{x_1, ..., x_n\}$ to be clustered, the number of clusters K, the initialization for the clustering assignment Y_{init}, and the maximum number of iterations $maxiter$. After initializing the clustering result $Y^{(0)}$, the iteration counter t and the memberships $w_{k,i}^{(0)}$ (lines 2–4), the algorithm iteratively repeats two steps: a model update step (lines 7–10) and an assignment step (lines 11–19). In the first step (model update), we first determine the set of points to be used to train each RF cluster model (line 8). For a given cluster k, this set X_k contains the points assigned to the k-th cluster in the previous iteration, plus the point x_o, which represents the "nearest" outlier – it is the point, among those not assigned to cluster k, which membership $w_{k,i}^{(t-1)}$ is maximum[4]. We added this point in order to force the presence of at least one outlier in the set X_k: actually Isolation Forests somehow assume that *there are* outliers; authors in [28], [29] train Forests using all available data, suggesting that training with only inliers may decrease the performances. Given the training set X_k, the procedure then uses it to train the k-th cluster model (line 9).

Input: $X, Y_{init}, K, maxiter$
Output: Y, t, E
1: **procedure** BASIC K-RF($X, Y_{init}, K, maxiter$)
2: $Y^{(0)} \leftarrow Y_{init}$
3: $t \leftarrow 0$
4: $w_{k,i}^{(0)} \leftarrow 1/K \quad \forall i = 1...n \ \forall k = 1...K$
5: **repeat**
6: $t \leftarrow t + 1$
7: **for** $k \leftarrow 1...K$ **do**
8: $X_k \leftarrow \{x_i \text{ s.t. } y_i^{(t-1)} == k\} \cup \{x_o\}$
9: $RF_k \leftarrow trainRF(X_k)$
10: **end for**
11: $E^{(t)} \leftarrow 0$
12: **for** $i \leftarrow 1...n$ **do**
13: **for** $k \leftarrow 1...K$ **do**
14: $w_{k,i}^{(t)} \leftarrow w_{RF_k}(x_i)$
15: **end for**
16: $y_i^{(t)} \leftarrow \arg\max_k w_{k,i}^{(t)}$
17: $E^{(t)} \leftarrow E^{(t)} + \frac{\max_k w_{k,i}^{(t)}}{\sum_{j=1}^{K} w_{j,i}^{(t)}}$
18: **end for**
19: $Y^{(t)} \leftarrow \{y_1^{(t)} ... y_n^{(t)}\}$
20: **until** ($Y^{(t)} == Y^{(t-1)}$) OR ($t == maxiter$)
21: **return** $Y^{(t)}, t, E^{(t)}$
22: **end procedure**

Fig. 1. The basic K-RF Algorithm.

In the second step (the assignment step, lines 11–19), the algorithm computes, for each point x_i, the membership score $w_{RF}(x)$ defined in eq. (3) with respect to all models $RF_1, \cdots RF_K$ (lines 13–15). Then, the new cluster assignment $y_i^{(t)}$ for point x_i corresponds to the cluster for which the membership is maximum (line 16). In this second step, we also compute the energy $E^{(t)}$ (lines 11 and 17), which is aimed at quantifying the goodness of the fitting at iteration t. This energy resembles the optimization function of the K-means, and is defined as:

$$E^{(t)} = \sum_{i=1}^{n} \frac{\max_k w_{k,i}^{(t)}}{\sum_{j=1}^{K} w_{j,i}^{(t)}} \tag{4}$$

These two steps are repeated until there are no changes in the clustering assignments or the maximum number of iterations has been reached (line 20). An example of few iterations of

[3]More precisely $h_t(x)$ is computed as the number of edges the point x has to travel in the tree t to reach a leaf l, plus a correction factor $c(size(l))$. This last quantity represents the averaged path length of a tree built using the points in l (see below for the definition of the function $c(n)$).

[4]x_o is the point so that $o = \arg\max_{j,r,y_j^{(t-1)} != k} w_{r,j}^{(t-1)}$.

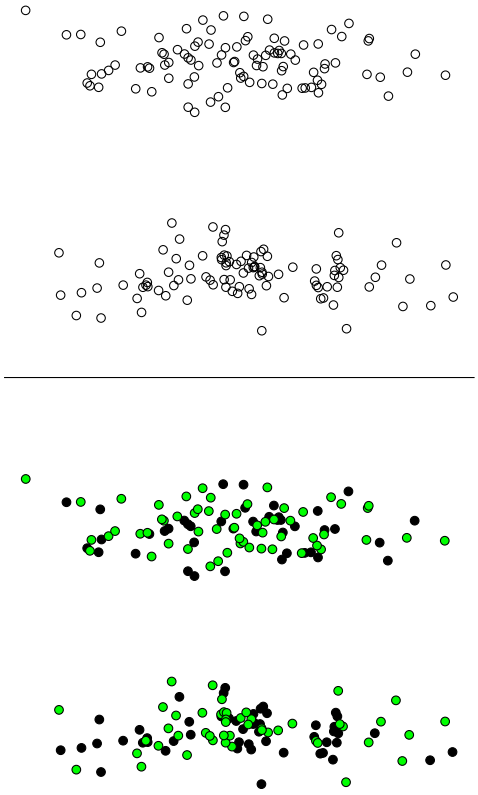

Fig. 2. Original Dataset (top) and starting random assignment of the clustering labels (bottom).*(Best viewed in color).*

the algorithm execution on a toy example (Figure 2) is shown in Figure 3.

C. Properties

Convergence. Similarly to what happened to other methods which implement k-means like algorithm [31], [32], the formal proof of the convergence of this procedure remains an open problem. Here difficulties are even larger, since the training step contains a random component (Isolation Forests are built using Extremely Randomized Trees).

From an empirical point of view, we noted that the convergence strongly depends on the initialization of the cluster assignment Y_{init}. To increase the robustness and to enforce the empirical convergence, we employed two solutions: first, we added a simple numerical stabilization of the memberships via a *damping factor* – similarly to what done in other approaches e.g. [38] – which gives an inertia to the membership computed in the previous iteration. More precisely, line 14 of the algorithm presented in Fig. 1 is substituted with:

$$w_{k,i}^{(t)} \leftarrow (1-\lambda) w_{RF_k}(x_i) + \lambda w_{k,i}^{(t-1)} \qquad (5)$$

where λ is the damping factor. Second, we have observed in our experiments (thousands of runs) that when we start from random assignments there are mainly two cases: i) the procedure converges very fast (few iterations), or ii) the procedure continues to oscillate between two regimes. Thus we adopted the following scheme: if the convergence of the procedure is not reached within the first few iterations, we re-initialize the labels, starting again the procedure. After $maxtrials$ re-initializations without convergence, the solution with the maximum energy in eq. (4) is returned. The resulting algorithm is summarized in Figure 4.

TABLE I
DETAILS OF THE DATASETS EMPLOYED FOR TESTING.

Name	#objects	#features	#clusters	#obj per cluster
Parkinsons	195	22	2	48,147
Iris	150	4	3	50,50,50
Wine	178	13	3	59,71,48
WBC	683	9	2	444,239
Auto-mpg	398	6	2	229,169

Computational Costs. The K-RF algorithm is more computationally demanding than the classic scheme, especially for what concerns the number of Forests to learn. In the classic scheme we have to train a single Forest on the whole dataset (enriched with the negative class)[5], whereas in the K-RF algorithm, we have to train K Random Forests for every iteration of the algorithm. However, the forests are trained using only a part of the dataset (all points are split among the K-clusters), and in many cases with few iterations the algorithm converged, thus maintaining the added computational burden to a reasonable level.

IV. EXPERIMENTAL EVALUATION

The K-RF algorithm has been preliminary evaluated using five classic datasets, all downloadable from the UCI Machine Learning Repository[6] or from the prtools library[7]. The details of the datasets are presented in Table I. As usual in clustering studies, we used labelled datasets (typically used for classification), removing labels before applying clustering algorithms, using them to assess the goodness of the results.

In the proposed approach, we let the number of trees M and the subsampling size ρ of Isolation Forests vary in a reasonable range, in order to have a deeper understanding of the method. In particular we tested our approach with 50, 100, 200 and 400 trees, using for each tree 50% and 80% of the training set. As explained in Section III-B, we built each tree until the end, i.e. until a leaf contains one point or equal points.

We started the clustering from random assignments of the clustering labels; after a preliminary evaluation not shown here, we set the number of iterations $maxiter$ to 15, the maximum number of re-initializations $maxtrials$ to 10, and the damping factor λ to 0.8. Actually, it turned out that these parameters were not so critical, and performances varied only slightly when varying them.

The proposed approach has been compared with the classic approach for Random Forest clustering (ProxRF) [17]–[19],

[5]In this scheme there is also the cost of the proximity-based algorithm used to get the cluster

[6]https://archive.ics.uci.edu/ml/datasets.html

[7]http://prtools.tudelft.nl/software/

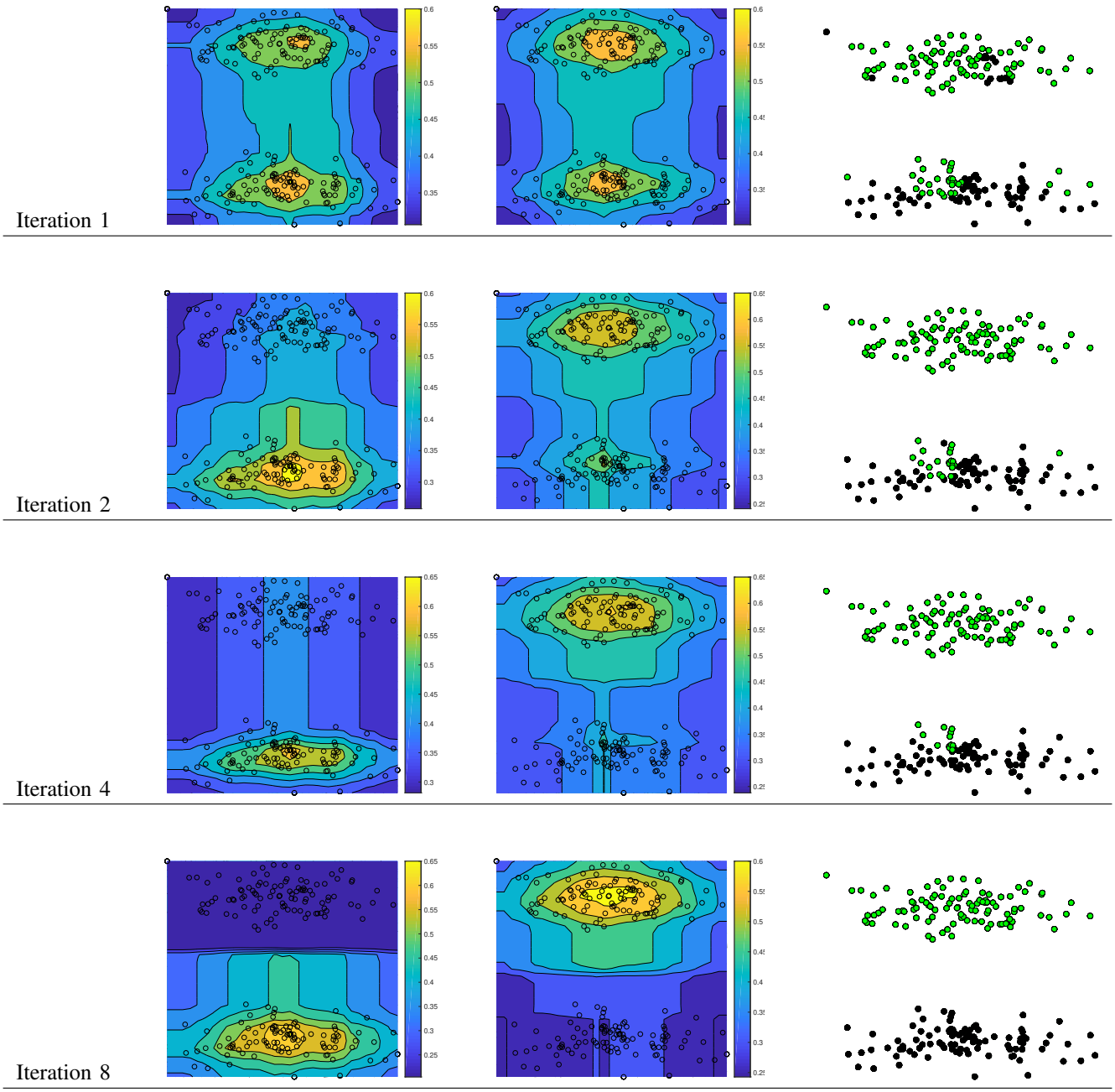

Fig. 3. Few iterations of K-RF on the toy example shown in Figure 2(top), starting with the initialization shown in Figure 2(bottom). The plots in the first two columns represent the contour lines of the memberships score defined in eq. 3 (in the first column we have the model of the first cluster, in the second column the model of the second). The last column reports the cluster assignments. It can be noticed that at the very beginning both models are focused on both clusters; as soon as the number of iterations grows, every model specializes itself on a specific cluster. Convergence in this case is obtained in 8 iterations. *(Best viewed in color).*

[21]–[26], which exploits the forest to derive a proximity between points to be used in a distance-based clustering scheme. In particular, here we implemented the version proposed in [17], which trains a classical classification forest using a synthetically generated negative class. In our experiments the negative class was obtained by randomly sampling a set of n points (with n the size of the original dataset to be clustered) from the product of empirical marginal distributions of the observed data. The decision forest was built with the classic Gini index, growing each tree until the end (leaves with one element or equal elements). Given the proximity, we derived the final clustering with Spectral Clustering [20], as done in more recent RF-clustering works [21]; in particular we used the Ng-Jordan-Weiss normalized version [20], repeating the inner k-means 20 times and taking the best result in terms of the objective function. Number of trees and subsampling size were set as for the K-RF algorithm, comparing the two frameworks when varying these parameters.

Input: $X, Y_{init}, K, maxiter, maxtrials$
Output: Y, t, E
procedure ENHANC. K-RF($X, Y_{init}, K, maxiter, maxtrials$)
  $\ell \leftarrow 0$
  repeat
    $\ell \leftarrow \ell + 1$
    $Y_\ell, t_\ell, E_\ell \leftarrow BasicK - RF(X, Y_{init}, K, maxiter)$
    if $t_\ell < maxiter$ **then**
      return Y_ℓ, t_ℓ, E_ℓ
    end if
  until $(\ell == maxtrials)$
  $opt \leftarrow \arg\max_\ell E_\ell$
  return $Y_{opt}, t_{opt}, E_{opt}$
end procedure

Fig. 4. The Enhanced K-RF Algorithm.

We computed clustering accuracies using the *purity index* [39] and the *adjusted Rand index* (ARI) [40], [41], two widely used indices to compare partitions. The former is obtained by assigning to each cluster the class label that is most frequent in that cluster; then the purity is determined as the proportion of examples assigned to the correct label (0 (worst case) and 1 (best case)). The ARI index is obtained by first building a contingency table between the clustering and the true labeling; then the index is computed by measuring the agreement between the two partitions (the Rand index) corrected for the chance of the formation of the clusters. Also in this case, the higher the index value, the better the clustering. For each method, each dataset and each configuration of parameters, we repeated the clustering 30 times, computing and reporting in Table II the median clustering accuracy (top: purity, bottom: ARI, in bold the best value between the two alternatives). For each set of experiments, we also performed a statistical test to measure how significant were the reported differences. In particular we performed a two-sided rank sum test [42], under the null hypothesis that performances of our method and the classic proximity based RF approach in the 30 repetitions are independent samples from identical continuous distributions with equal medians, against the alternative that they do not have equal medians. In Table II with put a "(*)" when the test is passed (i.e. the difference is statistically significant) with a p-value less than 0.1, and a "(**)" when the test is passed with a p-value less than 0.01.

Different information can be derived from the table. In general, it is interesting to note that K-RF outperforms the classic counterpart in the datasets Parkinsons, iris, WBC and Auto-mpg, for almost all configurations of number of trees and subsampling ratio. In Parkinsons, the statistical significance is always present at the highest level, whereas for WBC, iris and Auto-mpg it depends on the particular configuration. However, when comparing the best configuration for the two methods (last row of the table) the improvement of our approach over the classic proximity-based scheme is significant and supported by a strong statistical evidence. For what concerns the wine dataset, results are mixed: for some configurations our method is better, for some others not; on this problem the classic scheme already works very well (please note the very high purity/ARI), so that improvements are difficult to be obtained.

For what concerns the parameters, it is interesting to note that K-RF seems to perform better with less trees: for iris, wine and WBC the best result is obtained with 50 trees, and increasing this number is not so beneficial for the scheme – this confirms results in [28], [29], where authors claimed that a reduced number of trees (100 in [28], [29]) is good to get reasonably results. On the contrary, the classic ProxRF method follows a more established trend, increasing the performances when increasing the number of trees. Concerning the subsampling ratio, both schemes prefer small fraction (50% of the training set): actually this seems reasonable (and in line with other studies on Random Forests), since it permits to introduce in the models more variability. These two observations are not valid for the Parkinsons and for the Auto-mpg datasets, for which however accuracies were all somehow low. Interestingly, the classic Random Forest approach completely fails in recovering the clustering on these two datasets.

A. Comparison with alternative clustering approaches

This section compares the K-RFapproach with other clustering techniques: we tested both classical approaches like k-means, hierarchical schemes or mixtures of Gaussians, as well as more sophisticated approaches, such as *affinity propagation* [38], k-means++ [43], *spectral clustering* [20] and the very recent *Binary Embedding clustering* [44]. For k-means and agglomerative clustering (single link, complete link, and Ward link), we used the versions implemented in Matlab; for Gaussian mixtures we employed the implementation from the Netlab toolbox[8]; for k-means++ we used a Matlab implementation from the Mathworks web site[9]; for affinity propagation we used the code downloaded from the authors' web site[10]; finally, for Binary Embedding Clustering we used the author's implementation.

In the k-means approaches ("k-means" and "k-means++"), we repeated the procedure 20 times, starting from random initializations (k-means) or from carefully chosen points as described in [43] (k-means++), keeping the best result (in terms of objective function). In the agglomerative clustering approaches, complete link ("HierCl-CL"), single link ("HierCl-SL") and Ward link ("HierCl-WL"), we used Euclidean distances. We used three versions of Gaussian mixture models: "GMM (diag)", with diagonal covariance matrices, "GMM (full)", with full covariance matrices, and "GMM (spher)", with spherical covariance matrices. In all versions we initialized the EM with 5 iterations of k-means, stopping the procedure at likelihood convergence. We used three versions of spectral clustering, one with the unnormalized graph Laplacian ("SpectClus"), and two using normalized graph Laplacians, in the version of Shi-Malik ("SpectClus (SM)") and Jordan-Weiss ("SpectClus (JW)"). For affinity propagation ("AffProp"), we

[8]http://www.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/downloads/
[9]https://it.mathworks.com/matlabcentral/fileexchange/28804-k-means++
[10]http://www.psi.toronto.edu

TABLE II
CLUSTERING RESULTS FOR DIFFERENT DATASETS AND DIFFERENT PARAMETER CONFIGURATIONS: (TOP) PURITY, (BOTTOM) ADJUSTED RAND INDEX. "PROXRF" IS THE CLASSIC PROXIMITY BASED CLUSTERING BASED ON RF, WHEREAS "K-RF" REPRESENTS OUR PROPOSED APPROACH.

Purity

		Parkinsons		**Iris**		**Wine**		**WBC**		**Auto-mpg**	
Ntrees	ρ	ProxRF	K-RF	ProxRF	K-RF	ProxRF	K-RF	ProxRF	K-RF	ProxRF	K-RF
50	0.5	0.753	**0.773**(**)	0.872	**0.893**(**)	0.938	**0.955**	0.924	**0.952**(**)	0.678	**0.793**
50	0.8	0.753	**0.789**(**)	0.836	**0.869**(*)	0.895	**0.912**	0.916	**0.919**	0.678	**0.846**(**)
100	0.5	0.753	**0.784**(**)	0.866	**0.893**	**0.949**	0.935	0.919	**0.947**(*)	0.678	**0.848**(**)
100	0.8	0.753	**0.789**(**)	0.812	**0.846**	**0.912**	0.898	0.913	**0.930**(*)	0.678	**0.802**(**)
200	0.5	0.753	**0.784**(**)	0.876	**0.889**	0.944	**0.955**	0.923	**0.946**(**)	0.678	**0.805**(**)
200	0.8	0.753	**0.799**(**)	0.826	**0.862**(*)	**0.944**(*)	0.881	0.916	**0.947**(**)	0.678	**0.816**(**)
400	0.5	0.753	**0.784**(**)	0.866	**0.886**	0.949	**0.949**	0.922	**0.930**	0.678	**0.848**(**)
400	0.8	0.753	**0.784**(**)	0.836	**0.852**	**0.915**	0.898	0.906	**0.933**(*)	0.678	**0.846**(**)
Best Conf		0.753	**0.799**(**)	0.876	**0.893**(**)	0.949	**0.955**	0.924	**0.952**(**)	0.678	**0.848**(**)

Adjusted Rand Index

		Parkinsons		**Iris**		**Wine**		**WBC**		**Auto-mpg**	
Ntrees	ρ	ProxRF	K-RF	ProxRF	K-RF	ProxRF	K-RF	ProxRF	K-RF	ProxRF	K-RF
50	0.5	0.145	**0.244**(**)	0.686	**0.731**(**)	0.815	**0.862**	0.718	**0.815**(**)	0.121	**0.348**(**)
50	0.8	0.160	**0.250**(**)	0.621	**0.683**(**)	0.711	**0.746**	0.693	**0.703**	0.121	**0.480**(**)
100	0.5	0.152	**0.268**(**)	0.670	**0.731**(*)	**0.854**	0.806	0.700	**0.799**(*)	0.121	**0.483**(**)
100	0.8	0.151	**0.224**(**)	0.577	**0.649**(**)	**0.748**	0.707	0.681	**0.738**(*)	0.121	**0.365**(**)
200	0.5	0.144	**0.242**(**)	0.691	**0.725**	0.832	**0.862**	0.715	**0.796**(**)	0.121	**0.371**(**)
200	0.8	0.164	**0.249**(**)	0.594	**0.672**(**)	**0.831**(*)	0.669	0.691	**0.799**(**)	0.121	**0.399**(**)
400	0.5	0.153	**0.247**(**)	0.667	**0.716**	**0.849**	0.845	0.713	**0.740**	0.121	**0.483**(**)
400	0.8	0.151	**0.229**(**)	0.611	**0.652**	**0.760**	0.714	0.659	**0.748**(*)	0.121	**0.479**(**)
Best Conf		0.164	**0.268**(**)	0.691	**0.731**(**)	0.854	**0.862**	0.718	**0.815**(**)	0.121	**0.483**(**)

employed the version which allows setting the number of clusters. Finally, for Binary Embedding Clustering ("BinEmbClus") we set the parameters as reported in the original paper [44]. For some of these techniques a space normalization was necessary in order to get reasonable results (this was especially true for those technique based on euclidean distances, which may suffer for badly scaled spaces) – please note this was not needed with our RF-based technique. For what concerns our approach, for each dataset we selected the best number of trees and subsampling ratio, as determined by the analysis shown in the previous section. Among the 30 runs, we selected the clustering with the highest energy, as defined in eq. (4) ("K-RF" in the table). To have a direct comparison, we also reported the maximum accuracy obtained among the 30 runs ("K-RF (Max)"). All these results are displayed in Table III.

Even if preliminary, the analysis shows that the proposed approach compares very well with classic as well as with sophisticated alternative approaches, with performances which are among the top ones for all datasets. More in detail, for Parkinsons and WBC our scheme outperforms all the others, whereas for iris, wine and Auto-mpg it represents the second best technique. Please note that these performances can be increased even more by properly solving the initialization issue: when considering the maximum performances (last row), the presented method always outperforms all other approaches (the only exception is the very recent Binary Embedding Clustering [44] scheme on the Auto-mpg dataset).

V. CONCLUSIONS

In this paper a novel Random Forest approach for clustering has been introduced. The approach uses a different Random Forest to characterize every cluster; these Random Forests are iteratively updated with a K-means-like algorithm. The approach has been evaluated on five different datasets, with encouraging results.

ACKNOWLEDGEMENTS

The author would like to thank M. Denitto (Humatics s.r.l.) for helpful discussions.

REFERENCES

[1] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.
[2] G. Biau and E. Scornet, "A random forest guided tour," *TEST*, vol. 25, no. 2, pp. 197–227, 2016.
[3] 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*, vol. 7, no. 2-3, pp. 81–227, 2012.
[4] R. Díaz-Uriarte and S. A. de Andrés, "Gene selection and classification of microarray data using random forest," *BMC Bioinformatics*, vol. 7, p. 3, 2006. [Online]. Available: https://doi.org/10.1186/1471-2105-7-3
[5] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool, "Random forests for real time 3d face analysis," *International Journal of Computer Vision*, vol. 101, no. 3, pp. 437–458, 2013.
[6] S. Joelsson, J. Benediktsson, and J. Sveinsson, "Random forest classifiers for hyperspectral data," in *Proc. IEEE Int. Geoscience & Remote Sensing Symposium*, 2005, p. 4.
[7] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Commun. ACM*, vol. 56, no. 1, pp. 116–124, 2013.
[8] A. Criminisi and J. Shotton, *Decision Forests for Computer Vision and Medical Image Analysis*. Springer-Verlag, 2013.
[9] L. Breiman, J. Friedman, C. Stone, and R. Olshen, *Classification and Regression Trees*. Chapman and Hall, 1984.
[10] J. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.

TABLE III
COMPARATIVE RESULTS

	Purity					*Adjusted Rand Index*				
Method	**Parkinsons**	**Iris**	**Wine**	**WBC**	**Auto-mpg**	**Parkinsons**	**Iris**	**Wine**	**WBC**	**Auto-mpg**
HierCl-SL	0.753	0.664	0.396	0.651	0.574	-0.007	0.558	-0.007	0.003	-0.001
HierCl-CL	0.753	0.879	0.836	0.906	0.796	-0.000	0.706	0.577	0.653	0.349
HierCl-WL	0.753	0.886	0.927	0.968	0.673	0.006	0.720	0.790	0.874	0.114
K-means	0.753	0.886	0.966	0.960	0.844	-0.000	0.716	0.897	0.846	0.472
GMM (Diag)	0.753	0.906	0.972	0.650	0.753	0.108	0.759	0.915	0.000	0.255
GMM (Full)	0.753	0.966	0.977	0.849	0.753	-0.000	0.904	0.928	0.473	0.255
GMM (Spher)	0.753	0.886	0.960	0.946	0.673	0.006	0.716	0.879	0.794	0.114
SpectClus	0.753	0.671	0.396	0.651	0.690	-0.007	0.566	-0.007	0.003	0.140
SpectClus (SM)	0.753	0.725	0.396	0.940	0.685	0.000	0.424	-0.007	0.771	0.132
SpectClus (JW)	0.753	0.812	0.396	0.968	0.834	0.000	0.580	-0.007	0.874	0.445
K-means++	0.753	0.886	0.966	0.962	0.829	0.188	0.716	0.897	0.852	0.466
AffProp	0.753	0.899	0.910	0.959	0.811	0.089	0.745	0.741	0.841	0.386
BinEmb Clus	0.753	0.960	0.977	0.943	0.867	0.215	0.886	0.931	0.784	0.537
K-RF	0.830	0.960	0.972	0.974	0.851	0.337	0.886	0.915	0.897	0.494
K-RF (Max)	*0.851*	*0.966*	*0.983*	*0.975*	*0.854*	*0.408*	*0.904*	*0.947*	*0.902*	*0.501*

[11] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006.

[12] S. Clemencon, M. Depecker, and N. Vayatis, "Ranking forests," *Journal of Machine Learning Research*, vol. 14, pp. 39–73, 2013.

[13] H. Ishwaran, U. Kogalur, E. Blackstone, and M. Lauer, "Random survival forests," *The Annals of Applied Statistics*, vol. 2, pp. 841–860, 2008.

[14] C. Désir, S. Bernard, C. Petitjean, and L. Heutte, "One class random forests," *Pattern Recognition*, vol. 46, pp. 3490–3506, 2013.

[15] N. Goix, N. Drougard, R. Brault, and M. Chiapino, "One class splitting criteria for random forests," in *Proc. of Asian Conf. on Machine Learning, ACML*, 2017, pp. 343–358.

[16] A. Joly, P. Geurts, and L. Wehenkel, "Random forests with random projections of the output space for high dimensional multi-label classification," in *Machine Learning and Knowledge Discovery in Databases*, 2014, pp. 607–622.

[17] T. Shi and S. Horvath, "Unsupervised learning with random forest predictors," *Journal of Computational and Graphical Statistics*, vol. 15, no. 1, pp. 118–138, 2006.

[18] F. Moosmann, B. Triggs, and F. Jurie, "Fast discriminative visual codebooks using randomized clustering forests," in *Advances in Neural Information Processing Systems 19*, 2006, pp. 985–992.

[19] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, 2008.

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

[21] X. Zhu, C. Loy, and S. Gong, "Constructing robust affinity graphs for spectral clustering," in *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014*, 2014, pp. 1450–1457.

[22] E. Allen, S. Horvath, F. Tong, P. Kraft, E. Spiteri, A. Riggs, and Y. Marahrens, "High concentrations of long interspersed nuclear element sequence distinguish monoallelically expressed genes," *Proceedings of the National Academy of Sciences*, vol. 100, no. 17, pp. 9940–9945, 2003.

[23] N. Afanador, A. Smolinska, T. Tran, and L. Blanchet, "Unsupervised random forest: a tutorial with case studies," *Journal of Chemometrics*, vol. 30, no. 5, pp. 231–231, 2016.

[24] D. Seligson, S. Horvath, T. Shi, H. Yu, S. Tze, M. Grunstein, and S. Kurdistani, "Global histone modification patterns predict risk of prostate cancer recurrence," *Nature*, vol. 435, no. 7046, pp. 1262–6, 2005.

[25] T. Shi, D. Seligson, A. Belldegrun, A. Palotie, and S. Horvath, "Tumor classification by tissue microarray profiling: Random forest clustering applied to renal cell carcinoma," *Modern Pathology*, vol. 18, pp. 547–557, 2005.

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

[27] G. McLachlan and D. Peel, *Finite Mixture Models*. Wiley, 2000.

[28] F. Liu, K. Ting, and Z. Zhou, "Isolation-based anomaly detection," *ACM Trans. on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, pp. 3:1–3:39, 2012.

[29] ——, "Isolation forest," in *Proc. of Int. Conf. on Data Mining*, 2008, pp. 413–422.

[30] A. Jain and R. Dubes, *Algorithms for clustering data*. Prentice Hall, 1988.

[31] F. Camastra and A. Verri, "A novel kernel method for clustering," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 801–805, 2005.

[32] M. Bicego and M. Figueiredo, "Soft clustering using weighted one-class support vector machines," *Pattern Recognition*, vol. 42, pp. 27–32, 2009.

[33] A. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong, "Systematic construction of anomaly detection benchmarks from real data," in *Proc. ACM SIGKDD Workshop on Outlier Detection and Description*, 2013, pp. 16–21.

[34] H. Blockeel, L. D. Raedt, and J. Ramon, "Top-down induction of clustering trees," in *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, 1998, pp. 55–63.

[35] B. Liu, Y. Xia, and P. Yu, "Clustering through decision tree construction," in *Proceedings of the 2000 ACM CIKM International Conference on Information and Knowledge Management*, 2000, pp. 20–29.

[36] D. Yan, A. Chen, and M. Jordan, "Cluster forests," *Computational Statistics & Data Analysis*, vol. 66, pp. 178–192, 2013.

[37] H. Liu, M. Xu, H. Gu, A. Gupta, J. Lafferty, and L. Wasserman, "Forest density estimation," *Journal of Machine Learning Research*, vol. 12, pp. 907–951, 2011.

[38] B. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, p. 972976, 2007.

[39] C. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

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

[41] D. Steinley, "Properties of the Hubert-Arabie adjusted Rand index," *Psychological methods*, vol. 9, no. 3, p. 386, 2004.

[42] M. Hollander and D. Wolfe, *Nonparametric Statistical Methods*, 2nd ed. John Wiley & Sons, Inc., 1999.

[43] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proc. ACM-SIAM symposium on Discrete algorithms*, 2007, pp. 1027–1035.

[44] M. Bicego and M. Figueiredo, "Clustering via binary embedding," *Pattern Recognition*, vol. 83, pp. 52–63, 2018.