Contents

DEVELOPMENT OF A CLUSTERING ALGORITHM BASED ON THE MONDRIAN PROCESS
Models and Numerical Methods course project

Silvia Maria Macrì

The scope of this report is to exploit the Mondrian process, a temporal stochastic process that hierarchically partitions the space, in order to develop an unsupervised method that classify unlabeled data. After a brief remind of what cluster analysis is, the Mondrian process and some of its properties are illustrated; then, a clustering algorithm based on the process is described. Finally, some examples are shown, in order to highlight some methodological characteristics and issues related to the formulation of the process.

1 Clustering

Cluster analysis is a machine learning approach whose scope is to divide a population of data into a certain number of groups. In contrast with supervised learning methods, like classification, in which data are grouped on the basis of some existing labeling, clustering deals with unlabeled data, and the division into classes is performed in order to maximize intra-class similarity and minimize inter-class similarity. Since there is not an unique and precise definition of similarity, each clustering algorithm is characterized by a specific metric.

The wide variety of clustering algorithms can be categorized in different ways; a fundamental difference is, for example, between hierarchical and partition clustering approach. The hierarchical clustering groups data by using top-down and bottom-up approach: the divisive or top-down algorithms start to consider the whole set of data as belonging to an unique initial cluster, that is progressively divided into an increasing number of clusters, in order to maintain more homogeneous groups of data; on the contrary, the aggregative or bottom-up algorithms start with each element of the set of data belonging to different clusters and progressively merges them until a specific condition is satisfied. The partition clustering doesn't have a hierarchical structure but assigns data into a certain number of clusters by progressively optimizing some criterion function.

Decision tree Given a set of data $X \subseteq \mathbb{R}^D$, a decision tree on X can be defined as a hierarchical, axis-aligned, binary partitioning of X [1]. It has a network structure, where each node represents a specific restriction of the initial dataset. The binarity of the tree means that each non-leaf node is linked to exactly two children and, except for the root node that represents the whole data space X, each node has only one father; in particular, the set of data corresponding to the father node splits into two subsets assigned respectively to the two children. The term axis-aligned means that each split of a father into two children is performed by imposing a threshold value on only one of the features; if we visualize the tree structure in a product space, this means that each cut is performed orthogonally to one of the axis. The union of the subspaces corresponding to the leaf nodes coincides with the initial space X and their intersection is an empty set. Because of its hierarchical structure, the decision tree can be used as hierarchical clustering method (with both top-down and bottom-up approach), by selecting a data-dependent splitting (or merging) criterion.

2 The Mondrian Process

2.1 Definition

The Mondrian process is a recursive generative process, defined over an axis-aligned box $\Theta \subseteq \mathbb{R}^D$, that hierarchically partitions the underlying space through axis-aligned cuts [10]. Given an axis-aligned box $\Theta = [a_1, b_1] \times \ldots \times [a_D, b_D] \subseteq \mathbb{R}^D$, defined as the cartesian product of bounded intervals, the process generates two subspaces $\Theta_<$ and $\Theta_>$ by splitting Θ with some hyperplane orthogonal to one of the coordinate axis; then, it similarly generates independent splits on $\Theta_<$ and $\Theta_>$, creating new subspaces that are recursively cut until a certain condition is satisfied. The following scheme shows the generative process in detail:

MONDRIAN(Θ, t_0):
1. $T \sim Exp(LD(\Theta))$
2. $d \sim$ Discrete$(p_1, ..., p_D)$ where $p_d \propto (b_d - a_d)$
3. $x \sim \mathcal{U}([a_d, b_d])$
4. $M_< \leftarrow$ MONDRIAN$(\Theta_<, t_0 + T)$ where $\Theta_< = \{z \in \Theta | z_d \leq x\}$
5. $M_> \leftarrow$ MONDRIAN$(\Theta_>, t_0 + T)$ where $\Theta_> = \{z \in \Theta | z_d \geq x\}$

return $(t_0 + T, d, x, M_<, M_>)$

The process is initialized by fixing the starting time t_0 and the initial axis-aligned box Θ that will be partitioned. Firstly, in line 1, the time T of the first cut is randomly generated from an exponential distribution, with rate the linear dimension of the box $LD(\Theta) = \sum_{d=1}^{D}(b_d - a_d)$; this means that cuts in smaller boxes are expected to occur later than in bigger ones. The absolute time $t_0 + T$ at which the cut is generated is called birth time. Then, the dimension d of the cut is generated from a discrete distribution, with a probability proportional to the length $b_d - a_d$ of the corresponding interval (line 2), and the location of the cut is subsequently uniformly chosen in the interval $[a_d, b_d]$ (line 3).
At this point, the initial space Θ has been cut by an hyperplane orthogonal to the d-dimension and intersecting the $[a_d, b_d]$ interval in $d = x$, and it has been divided into two axis-aligned boxes $\Theta^<$ and $\Theta^>$. Lines 4 and 5 generate independent Mondrians on the two new subspaces, meaning that steps 1,2 and 3 are independently repeated on $\Theta_<$ and $\Theta_>$, starting from an initial time equal to the birth time of the last cut.
The process recursively splits the space in more and more refined partitions and stops when, for every independent Mondrian that has been generated, the birth time $t_0 + T$ of the last cut is higher than a fixed lifetime parameter λ. The lifetime λ represents a budget assigned to the process and, as cuts appear, it is progressively spent; it follows that the time interval T, that is generated at each iteration, represents the cost related to a specific cut.

The Mondrian process as temporal stochastic process

The formulation of Mondrian process as recursive generative process leads to an other definition as temporal stochastic process. More precisely, given an axis-aligned box $\Theta \subseteq \mathbb{R}^D$, the Mondrian process on Θ, denoted as $MP(\Theta)$, can be defined as a temporal stochastic process $(M_t)_{t>0}$ taking values in the ensemble of guillotine partitions of Θ and with distribution specified by the generative process MONDRIAN(Θ, t_0) [1]. If we define a guillotine partition as a hierarchical partition of Θ obtained by recursively splitting the space through hyperplanes orthogonal to the coordinate axes, the random variable $(M_t)_{t>0}$ is a guillotine partition of Θ formed by the cuts with birth time $t_b \leq t$.

Tree structure of the Mondrian process

The hierarchical structure of Mondrian process reflects the structure of a rooted binary k-d tree; each node is represented by the tuple $(t_0 + T, d, x, M_<, M_>)$, which entries are the birth time, the dimension and the location of the cut and the two children of the node itself. Each non-leaf node has exactly two children and the ensemble of the leaf nodes represents the final and more refined partition of the initial space. The lifetime parameter λ of the stochastic process, that regulates the total number of cuts, can be interpreted as the depth parameter of the tree.

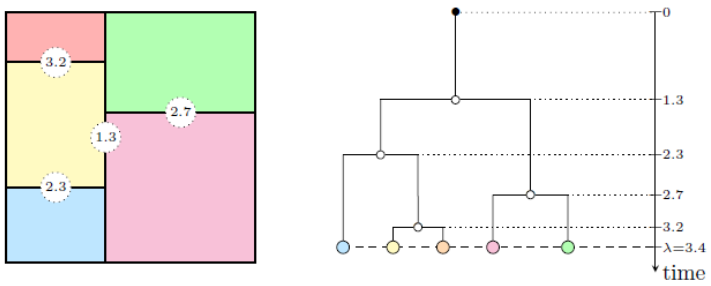

Figure 1: A Mondrian partition (left) with corresponding tree structure (right), which shows the evolution of the tree over time [7]

2.2 Properties of the Mondrian process

We firstly define a Poisson point process [4], through two equivalent definitions:

Definition 1 A Poisson point process is an arrival process[1] for which the interarrival times are independent and identically distributed random variables and in which the interarrival intervals have an exponential distribution function. This means that, for some $\lambda > 0$, called rate of the process, the interarrival times are represented by IID random variables $\{X_j : j = 1, 2, ...\}$ with the common density $f_X(t) = \lambda e^{-\lambda t}$, with $t \geq 0$.

Definition 2 An arrival process is a Poisson point process if its counting process $\{N(t); t > 0\}$ has the following properties:

- it is Poisson distributed $N(t) \sim Poisson(\lambda t)$
- $N(t_2) - N(t_1)$ has the same cumulative distribution function as $N(t_2 - t_1)$ for all $0 < t_1 < t_2$, that means the distribution of the number of events in some interval only depends on the size of the interval, and not on its starting point (**stationary increment property**)
- for any positive integer k and every k-tuple of times $0 < t_1 < t_2 < \cdots < t_k$, the random variables $N(t_1), N(t_2) - N(t_1), ..., N(t_k) - N(t_{k-1})$ are independent (**independent increment property**)

One-dimensional Mondrian process as Poisson point process

Consider a one-dimensional Mondrian process $MP(\Theta)$ with $\Theta = [a, b] \subseteq \mathbb{R}$ and with lifetime λ. Since, given n independent Poisson processes $N_1(t), ..., N_n(t)$ with rates $\lambda_1, ..., \lambda_n$, the combination of them $N(t) = N(t_1) + ... + N(t_n)$ is a Poisson point process with rate $\lambda = \lambda_1 + ... + \lambda_n$, and since the time intervals of $MP([a, b])$ are independent (for the memoryless property of the exponential) and generated from an exponential distribution with rate the length of the interval, the times of the cuts of the Mondrian process form a temporal Poisson process with rate $b - a$ (see **Definition 1**). It follows that the number of the times of the cuts in a time interval of length λ is $Poisson(\lambda(b - a))$ distributed.

Now consider the cut locations in the interval $\Theta = [a, b]$: as the process progresses, the cut locations are generated in each of the subintervals of Θ from a uniform distribution and, consequently, if we consider the ensemble of the cut locations generated in different subintervals, they are all uniformly generated in the whole interval Θ; moreover, as in the case of exponential distribution, also an extraction from a uniform distribution is independent from its past evolution. From these two properties, and since the number of the times of the cuts coincides with the number of the locations of the cuts and it has thus a $Poisson(\lambda(b - a))$ distribution with rate λ, the distribution of the cut locations of a one-dimensional Mondrian process run on $[a; b]$ with lifetime λ is a Poisson point process with constant intensity λ (see **Definition 2**).

Self-consistency

Consider a Mondrian process $MP(\Theta)$ over a box $\Theta = \Theta_1 \times ... \times \Theta_D$ and consider a smaller box contained within it, $\Phi_1 \times ... \times \Phi_D \subseteq \Theta_1 \times ... \times \Theta_D$. The stochastic process induced by $MP(\Theta)$ through its cuts intersecting the smaller box Φ is again a Mondrian process $MP(\Phi)$. This means that, considering all the cuts generated by the Mondrian process on Θ, the ones that don't intersect the subspace Φ don't affect the distribution of the cuts within Φ and this distribution is the same of that of a Mondrian process applied on Φ.

Mondrian slices Consider a subspace $\Phi \subseteq \Theta$, that is a cartesian product space of the following intervals: $\Phi_1 = \{x\}$ and $\Phi_d = \Theta_d$ for $d > 1$. Since the probability of having a cut in the $d = 1$ dimension is zero, no cut occurs in the first dimension. Consequently, for the self-consistency property, the restriction on the subspace Φ of a D-dimensional Mondrian process on Θ is a $(D - 1)$-dimensional Mondrian process. In particular, if Φ is a one-dimensional subspace of Θ, the restriction of the Mondrian process on it is a one-dimensional process; since a one-dimensional Mondrian process is a Poisson point process, the location of the cuts of a D-dimensional Mondrian process, marginally considered in each dimension, follows a Poisson point process.

Extension on $\mathbb{R}^D$ A consequence of the self-consistency property is that the definition of the Mondrian process on an axis-aligned box can be relaxed by defining it on the entire $\mathbb{R}^D$. The structure of the Mondrian process on $\mathbb{R}^D$ is an infinitely deep tree with no root and infinite number of leaves.

[1] An arrival process is a sequence of increasing random variables, $0 < S_1 < S_2 < \cdots$, where $S_i < S_{i+1}$ means that $S_{i+1} - S_i$ is a positive random variable.

Conditional Mondrians

Consider again a Mondrian process $MP(\Theta)$ over a box Θ and the Mondrian process $MP(\Phi)$ over the restriction $\Phi \subseteq \Theta$. Given the Mondrian process over Φ, the property of self-consistency allows to have information on the unknown Mondrian process running on the whole Θ; in other words, it is possible to calculate the conditional distribution $MP(\Theta)$, given $MP(\Phi)$. In particular, conditionally given the restriction $MP(\Phi)$ of a Mondrian process $MP(\Theta)$ to a smaller box Φ, consider the first cut C^Φ in $MP(\Phi)$ and let t^Φ be its time. Then:

- with probability $exp(t^\Phi(LD(\Theta) - LD(\Phi)))$, the cut C^Φ is the first cut in Θ (it extends throughout Θ)
- with complementary probability $1 - exp(t^\Phi(LD(\Theta) - LD(\Phi)))$ the first cut in Θ misses Φ, its time has the truncated exponential distribution with rate $LD(\Theta) - LD(\Phi)$ and truncation at t^Φ, and the cut location is uniformly distributed along the segment where making a cut doesn't hit Φ.

3 Datasets

In the next sections, the clustering algorithm based on the Mondrian process is described; in order to clarify the exposition, some examples are shown. The datasets that we used are the following (see Figure 2):

- ***blobs***: two groups of data elongated in the diagonal directions
- ***circles***: two concentric circles with some added noise
- ***moons***: two interleaving half circles with some added noise

Each dataset consists of two separate groups of points and we will check if the algorithm is able to identify the corresponding clusters.

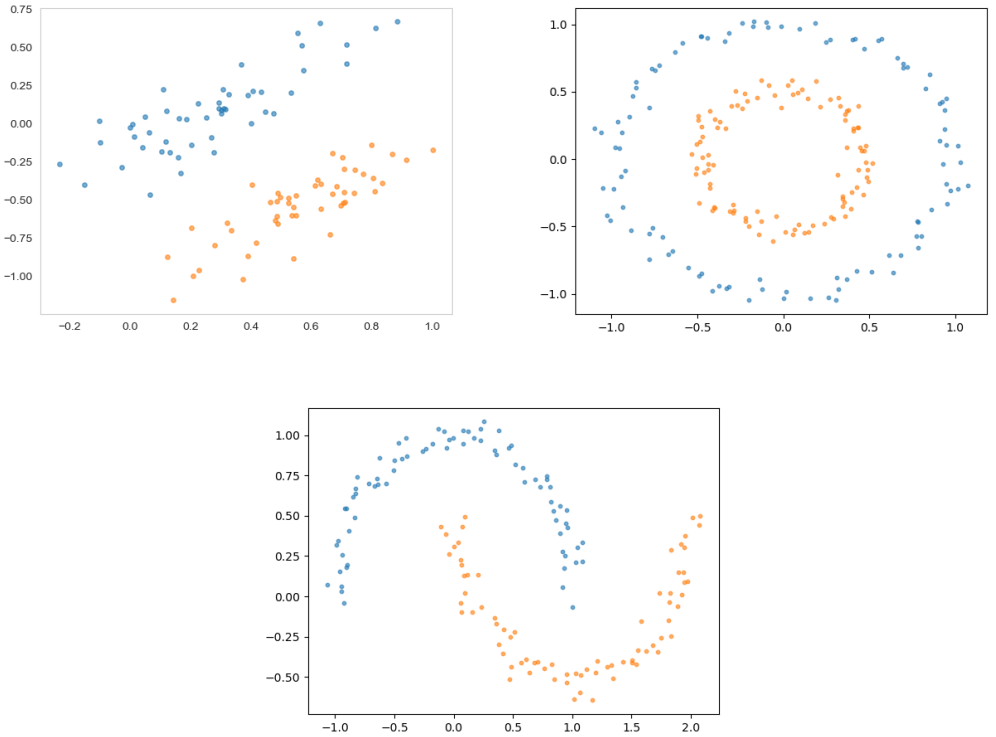

Figure 2: ***blobs***, ***circles*** and ***moons*** datasets.

4 Clustering algorithm based on the Mondrian Process

The tree structure of the Mondrian process, that reflects the way in which the Mondrian process partitions the space, naturally leads to its adaptation to a clustering hierarchical algorithm. The adapted process differs from the pure Mondrian process because the split criterion must rely on some characteristic of the data that we want to cluster.

The Mondrian clustering algorithm consists of a two-step process.The first part has a top-down approach and is the one that directly exploits the Mondrian process; it partitions the space on which the set of data is defined and, for each leaf node, assigns all the data belonging to the corresponding subspace to the same cluster. The second part of the two-step process has a bottom-up approach; it starts from considering each final subspace (corresponding to a leaf node of the tree) as an independent cluster and hierarchically merges them on the basis of the same metric used in the previous splitting process.

4.1 Partitioning

The partitioning of the space is the phase of the method that directly involves the adaptation of the Mondrian process. The main differences from the pure Mondrian process are that both the definition of the initial space and the cut choice depend on the input unlabeled data $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n) \in \mathbb{R}^D$; as regards the initial space, it is defined as the smaller axis-aligned box Θ that contains $\boldsymbol{X}$.
Following the scheme presented in Section 2.1, the new data-dependent Mondrian recursive generative process is shown below:

DATA-DEPENDENT-MONDRIAN($\boldsymbol{X}$,Θ,t_0):
1. $T \sim Exp(\text{vol}(\Theta(\boldsymbol{X})))$
2. h $\sim$ {hyperplane($\boldsymbol{x}_i, \boldsymbol{x}_j$) with $p \propto p_{i,j} | (\boldsymbol{x}_i, \boldsymbol{x}_j) \in \boldsymbol{X}$}
3. $\boldsymbol{X}_<, \boldsymbol{X}_>$ with $\boldsymbol{X}_< \in \Theta_<$, $\boldsymbol{X}_> \in \Theta_>$ and $\boldsymbol{X}_< \cup \boldsymbol{X}_> = \boldsymbol{X}$, $\Theta_< \cup \Theta_> = \Theta$
4. $M_< \leftarrow$DATA-DEPENDENT-MONDRIAN($\boldsymbol{X}_<, \Theta_<, t_0 + T$)
5. $M_> \leftarrow$ DATA-DEPENDENT-MONDRIAN($\boldsymbol{X}_>, \Theta_>, t_0 + T$)

return $(t_0 + T, h, M_<, M_>)$

Firstly, the time of the cut is generated from an exponential distribution with rate the volume of the initial space (line 1); this means that, as in the pure Mondrian process, the cut is expected to be sooner in larger boxes than in smaller ones. Then, for each pair of points $\in \boldsymbol{X}$, an hyperplane intersecting the space Θ is associated and a metric is calculated; the metric $p_{i,j}$ defines the degree of similarity between the two subset of points separated by the corresponding hyperplane. In line 2, the hyperplane of the cut is randomly generated from the ensemble of the hyperplanes that have been associated to each pair of data points, with probability of extraction proportional to the metric value. In line 3, the two just separated subsets of points $\boldsymbol{X}_<$ and $\boldsymbol{X}_>$ are associated to the corresponding subspaces $\Theta_<$ and $\Theta_>$. As in the pure Mondrian process, in lines 4 and 5, the previous steps are independently repeated over the two new subspaces.

The whole process stops when, as in the pure Mondrian process, the absolute time $t_0 + T$ is higher than λ, or when the set of data $\boldsymbol{X}$ contains only two points. The ensemble of the leafs of the Mondrian tree shows the final and more refined partition of the initial space and the union of all the sets of points contained in the leaf subspaces is equal to the initial set of data $\boldsymbol{X}$. All the data that are contained in the same subspace are considered to belong to the same cluster .

Figure 3 shows an example of one possible output of the partitioning phase run on ***circles*** datasets. It is obtained by hierarchically splitting the initial space through 15 cuts and the final partition consists of 16 adjacent polygones. Each subspace is associated to its characteristic number.

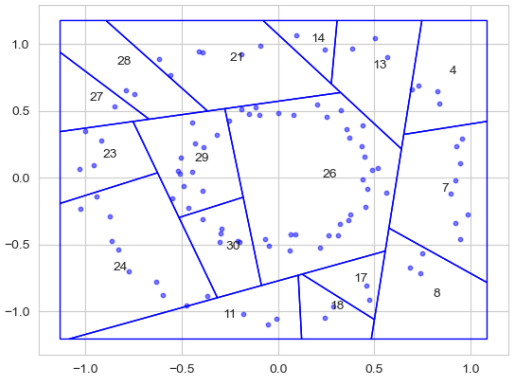

Figure 3: Output of the partitioning phase run on ***circles*** datasets

Hyperplane and metric choice

We considered three different choices for the hyperplane and the metric, that respectively are associated to each pair of points and estimates the similarity of two groups of data. While how they are calculated is explained in the following, the differences between the three methods are highlighted in Section 6.

Since a way to estimate the similarity of two groups of data is to evaluate how they are spatially separated, the chosen metrics are functions of the euclidean distance; in particular, the minimum distance between and within the two groups of data are compared.

Lower is the value of the metric, more similar are the two groups of points; for this reason this value can be used as probability of extraction of the corresponding cut (higher is the value of the metric, less similar are the two groups of data and higher is the probability to extract the hyperplane).

Algorithm 1 In this first case, we don't associate an hyperplane to each pair of points $\in \boldsymbol{X}$ but we consider their projections on the coordinate axis. For each dimension, an hyperplane is thus associated to each pair of consecutive projections; it is orthogonal to the corresponding d coordinate axis (as in the pure Mondrian process) and intersects the segment connecting the two projection points in its middle point.

The metric is calculated according to the following scheme:

METRIC($\boldsymbol{X}_<$,$\boldsymbol{X}_>$):

1. $d_{min} = \min(dist(\boldsymbol{X}_<, \boldsymbol{X}_>))$
   calculate the minimum distance between points belonging to the two different subspaces (*dist* is the euclidean distance)
2. $d_i = \min(dist(\boldsymbol{x}_i, \boldsymbol{X}_j)\ \forall \boldsymbol{x}_i \in \boldsymbol{X}_j$ where $j = <, >$
   for each point in $\boldsymbol{X}_<$ and $\boldsymbol{X}_>$, calculate the minimum distance between the considered point and an other point of the same group
3. $\bar{d} = (\sum_{i=1}^{D} d_i)/D$
   calculate the mean of the minimum distances within the groups (calculated in line 2)
4. $metric = |d_{min} - \bar{d}|$
   the metric is the absolute value of the difference between the minimum distance between the groups and the mean of the minimum distances within the groups

return(*metric*)

Algorithm 2 Here the metric is the same of the previous case, while the way to generate the hyperplanes is changed: the cuts are no more linked to be orthogonal to the coordinate axis, but can in principle take any direction of the space. For each pair of points, an hyperplane is associated: it is orthogonal to the segment connecting the two points and intersects it in its middle point.

Algorithm 3 In this third considered procedure, the hyperplanes can take any direction and are generated as in the previous case. The metric is instead lightly changed (see Figure 4 for a visualization of the considered segments):

METRIC-WITH-CORRECTION($\boldsymbol{X}_<$,$\boldsymbol{X}_>$):

1. $d_{min} = \min(dist(\boldsymbol{X}_<,\boldsymbol{X}_>))$
   calculate the minimum distance between points belonging to the two different subspaces (*dist* is the euclidean distance); we call the nearest pair $(\boldsymbol{x}_{<,1},\boldsymbol{x}_{>,1}) \in \boldsymbol{X}_< \times \boldsymbol{X}_>$
2. $d_i = \min(dist(\boldsymbol{x}_i,\boldsymbol{X}_j)\ \forall \boldsymbol{x}_i \in \boldsymbol{X}_j$ where $j = <,>$
   for each point in $\boldsymbol{X}_<$ and $\boldsymbol{X}_>$, calculate the minimum distance between the considered point and an other point of the same group
3. $\bar{d} = (\sum_{i=1}^{D} d_i)/D$
   calculate the mean of the minimum distances within the groups (calculated in line 2)
4. $d_{min,<} = \min(dist(\boldsymbol{x}_{<,2},\boldsymbol{X}_>))$
   consider $\boldsymbol{x}_{<,2}$, the nearest point to $\boldsymbol{x}_{<,1}$ belonging to $\boldsymbol{X}_<$; calculate the minimum distance between $\boldsymbol{x}_{<,2}$ and a point belonging to $\boldsymbol{X}_>$
5. $d_{min,>} = \min(dist(\boldsymbol{x}_{>,2},\boldsymbol{X}_<))$
   consider $\boldsymbol{x}_{>,2}$, the nearest point to $\boldsymbol{x}_{>,1}$ belonging to $\boldsymbol{X}_>$; calculate the minimum distance between $\boldsymbol{x}_{>,2}$ and a point belonging to $\boldsymbol{X}_<$
6. $metric = |d_{min} - \bar{d}| + d_{min,<} + d_{min,>}$

return(*metric*)

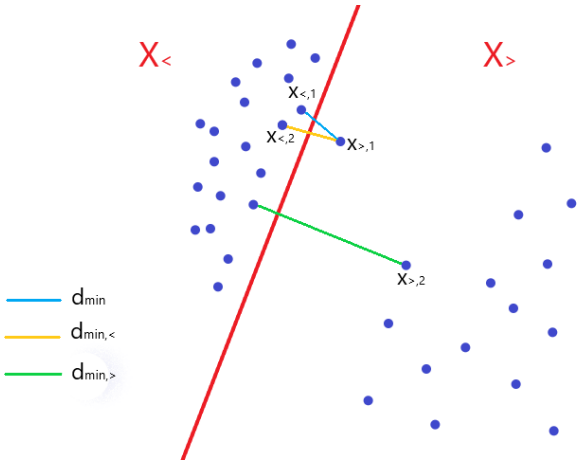

Figure 4: Points and segments considered in the calculation of the metric; $\boldsymbol{X}_<$ and $\boldsymbol{X}_>$ are the two sets of data separated by the red line.

Mathematical description

Hyperplane If we consider a D-dimensional space Θ, an hyperplane is a $(D-1)$-dimensional subspace of Θ. It can be described by a linear equation $a_1 x_1 + ... + a_{D-1} x_{D-1} = b$ or by a radius vector orthogonal to it; from a computational point of view, the description through the normal vector is more convenient, since it is more easily generalized to generic dimensions. In particular, each hyperplane is characterized by a set of $D-1$ point coordinates, that univocally determine the direction of the normal vector, and a scalar, representing its magnitude (that is the distance between the hyperplane and the origin of the axes).

Polytope While, if we consider only axis-aligned cuts, the generated subspaces are all axis-aligned boxes, in case of cuts with generic directions, the subspaces are no more cartesian products of boundend intervals but are generic convex polytopes (the convexity property holds if the initial space is convex). A polytope is mathematically described through its *half-space representation*, that means it is defined by intersection of a finite number of half spaces. In particular, starting from the equation of the hyperplane that splits the whole space into two half spaces, we can write each linear half space as linear inequality: $a_1 x_1 + ... + a_{D-1} x_{D-1} \leq b$ and $a_1 x_1 + ... + a_{D-1} x_{D-1} \geq b$. A polytope is thus

described as a system of linear inequalities, and the number of the inequalities is equal to the number of sides of the polytope:
$a_{1,1}x_1 + ... + a_{1,D-1}x_{D-1} \leq b_1$
$a_{2,1}x_1 + ... + a_{2,D-1}x_{D-1} \leq b_2$
$\cdots$
$a_{m,1}x_1 + ... + a_{m,D-1}x_{D-1} \leq b_m$
The system of inequalities can be written in matrix form $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$; a polytope is thus characterized by a $m \times (D-1)$ matrix $\mathbf{A}$ and a m-dimensional vector $\mathbf{b}$.

4.2 Merging

While the first part of the algorithm partitions the space and separates the data into several groups, the second part collects the subspaces into clusters and assigns them to different classes.

A bottom up approach is used; this means that, firstly, no more than one subspace is assigned to the same class and thus the number of classes is equal to the number of subspaces (remind that all the points belonging to the same subspace belong to the same class). The initial state is described by an undirected network G with number of nodes equal to the number of subspaces and without edges. Then, edges are progressively added to the network. The presence of an edge connecting two nodes means that two subspaces are merged, consequently assigning the corresponding groups of data to the same class; this means that, at each step, the total number of classes is equal to the number of connected components of the network. The order in which the edges are added to the network is determined by assigning a score to each edge; it represents a similarity measure between groups of data and is the same metric used to partition the space in the first part of the algorithm. The lower is the value of the metric, more similar are the two groups of data contained in neighbouring subspaces and more likely are the subspaces to belong to the same class. After the scores of all the possible edges are calculated, the edge with lower score is added to the network and the two subspaces corresponding to the nodes linked by the edge are merged; the metric is thus again calculated for the pairs of neighbouring subspaces and the score list is updated. Only edges that connect adjacent polytopes are considered.

Because both the metrics METRIC and METRIC-WITH-CORRECTION can't be calculated if one of the two subspaces contains single data, before creating the network G, each subspace with single data is merged to the corresponding nearest neighbouring subspace containing more than one data. More precisely, if $\boldsymbol{x}_1$ is the data contained in the single-data partition and $\boldsymbol{X}_2$ is the set of data contained in the second subspace, the closeness between two adjacent subspaces is determined by the following metric:

METRIC-SINGLE-DATA($\boldsymbol{x}_1$,$\boldsymbol{X}_2$):

1. $d_{min,1} = \min(dist(\boldsymbol{x}_1,\boldsymbol{X}_2))$
   calculate the distance between $\boldsymbol{x}_1$ and its nearest point belonging to $\boldsymbol{X}_2$; we call it $\boldsymbol{x}_{2,1}$
2. $\boldsymbol{x}_{2,2} \in \boldsymbol{X}_2$
   find the nearest point to $\boldsymbol{x}_{2,1}$ belonging to $\boldsymbol{X}_2$
3. $d_{min,2} = dist(\boldsymbol{x}_1, \boldsymbol{x}_{2,2}))$
   calculate the distance between $\boldsymbol{x}_{2,2}$ and $\boldsymbol{x}_1$;
4. $metric = d_{min,1} + d_{min,2}$

return(*metric*)

Figure 5 shows three steps of the merging process applied to a partition of ***blobs*** dataset. On the left, the partitioned space is shown, with the subspaces belonging to the same class characterized by the same color; the total number of colors is equal to the number of connected components of the corresponding network, shown on the right.

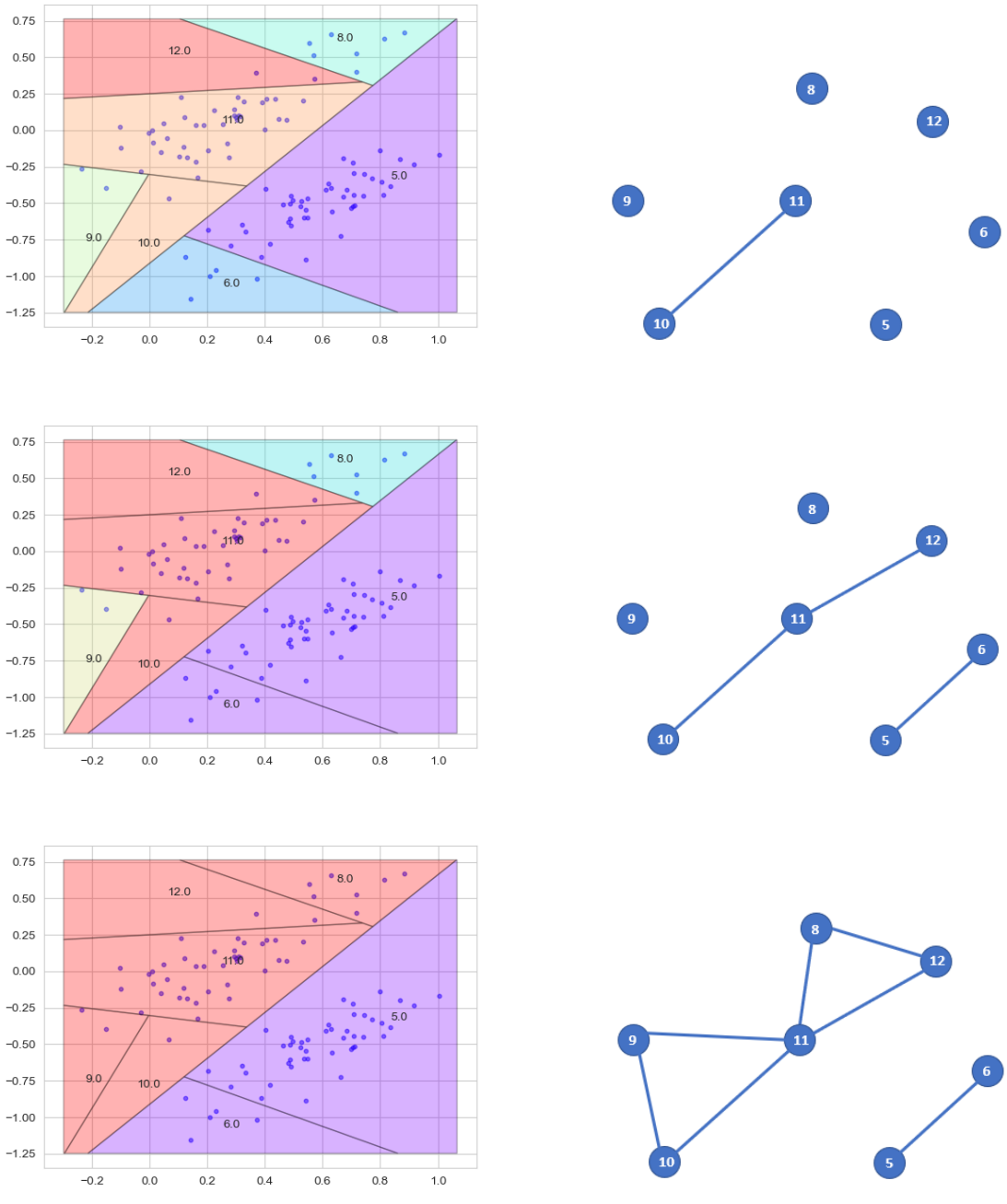

Figure 5: Three steps of merging process with corresponding graphs

5 Mondrian clustering forest

The output of the merging process doesn't give us information about how many clusters characterize the dataset. In fact, it considers different cases, each one characterized by a fixed number of clusters; for each case, it estimates how the subspaces are clustered, but is not able to determine what is the most probable among them. In order to automatically determine the number of clusters, a Mondrian clustering forest is considered; it is the equivalent of an unsupervised random forest based on Mondrian clustering algorithm.

The main advantage of the forest is that its output associates, to each point of the space, a measure of the probability to belong to a certain class. Figure 6 shows, on the left, a visualization of the averaged output of 10 Mondrian trees in case of ***moons*** dataset. The region characterized by a lighter color has a probability $p_1 = 1$ to belong to the first class and a probability $p_2 = 0$ to belong to the second class; on the contrary, the darker region has $p_1 = 0$ and $p_2 = 1$. The regions characterized by a range of intensities between the lighter and the darker ones are associated to different values of probabilities in the interval $[0,1]$ without extremes.

Moreover, the forest allows us to automatically determine the number of clusters in which the data and the corresponding space are divided. For this purpose, we consider, for each tree, the labeling associated to each point of the dataset, obtained by the merging procedure of the algorithm; for each possible number of cluster, we have a different set of labeled data. The labels obtained by two different trees are compared, through the *adjusted mutual information* score, for each possible fixed number of clusters, and this measure is repeated for each possible pair of considered trees. The *adjusted mutual information (AMI)* [8] is a function of the entropy of the two different partitions of the same set of data; it quantify the information shared by the two partitions and is thus a measure of the similarity of two labeling of the same set of data, ignoring permutations. It ranges between 0 and 1 and higher is its value, more similar are the two set of labels. Figure 6, on the right, shows a plot of the *adjusted mutual information* score in function of the number of clusters; each line corresponds to a specific pair of tree outputs; all the values of the score (corresponding to different pairs of trees) are averaged, for each fixed number of clusters, and the mean value is represented by the more intense blue dotted line. At this point, the number of clusters corresponding to the higher value of the *AMI* score is considered the most probable. The plot of *AMI* score vs the number of clusters in Figure 6 is referred to the forest of 10 Mondrian trees applied to ***moons*** dataset; the higher value of compatibility between tree is obtained for the number of clusters equal to 2.

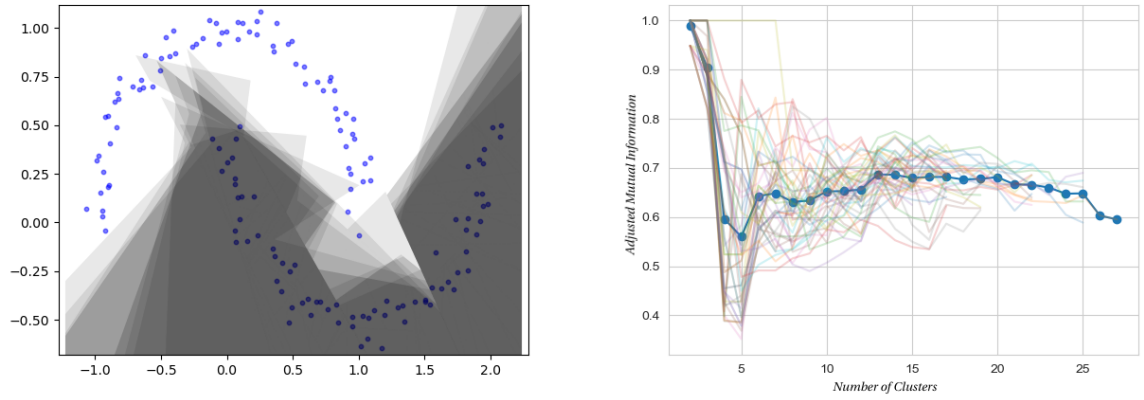

Figure 6: On the left, the output of a Mondrian clustering forest applied on ***moons*** dataset; on the right, the corresponding plot of *AMI* score vs number of clusters.

6 Examples in 2D

In the next sections, some example are shown, in order to clarify the performance of the algorithm, the differences of the three versions and to highlight some numeric issues related to methodological choices

6.1 Algorithm 1 vs Algorithm 2: evaluation of *partitioning* phase

In this section we consider the two versions of the algorithm characterized by different choices of the hyperplanes and the same similarity criterion METRIC.

Figures 7 and 8 show the partitioning phase of **Algorithm 1** and **Algorithm 2**, applied to datasets ***blobs*** and ***circles***. In order to evaluate the result of partitioning phase, we can firstly check if each final subspace contains only data belonging to the same class; moreover, since the number of splits is related to the computational cost of the algorithm, we can measure how many splits are necessary to completely separate the two clusters.

In each of the four considered cases, the space is splitted in order to have each polygon containing data belonging only to one class; from this point of view, it means that the two methods are equivalent and the partitioning output of both the algorithms can be considered a good input for the next *merging* phase.

As regards the evaluation of computational costs, consider Figure 7. While, in the partitioning procedure performed by **Algorithm 2**, the first split is enough to totally separate the two groups of data, in the other case the complete separation is obtained after nine splits; this means that the method with cuts that are not linked to be orthogonal to the coordinate axis is much more convenient from a computational point of view, especially when the datasets require a splitting procedure on the diagonal direction.

An other reason to prefer the non-axis-aligned cuts is the absence of preferred directions. Figure 8 shows ***circles*** dataset, that is rotation-invariant with respect to the axis perpendicular to the plane and intersecting the center of the circles. If we run several times **Algorithm 1** on the dataset, the cuts separating the two circles will be almost the same, because they are forced to be orthogonal to the axis; on the contrary, the output of **Algorithm 2** shown in the figure is only a possible realization of the algorithm: since the cuts can in principle take any direction and the process is randomized, different runs of the process give subspaces that contain the smaller circle with different shape. The absence of preferred directions is important especially when considering the outcome of the Mondrian forest: if we average several outputs of **Algorithm 2** trees, the probability distribution to belong to a certain class will reflect the rotation-invariance of the data, while in the other case the regions corresponding to different classes will be sharp-bordered and will show a presence of preferred direction that is not a real characteristic of the data.

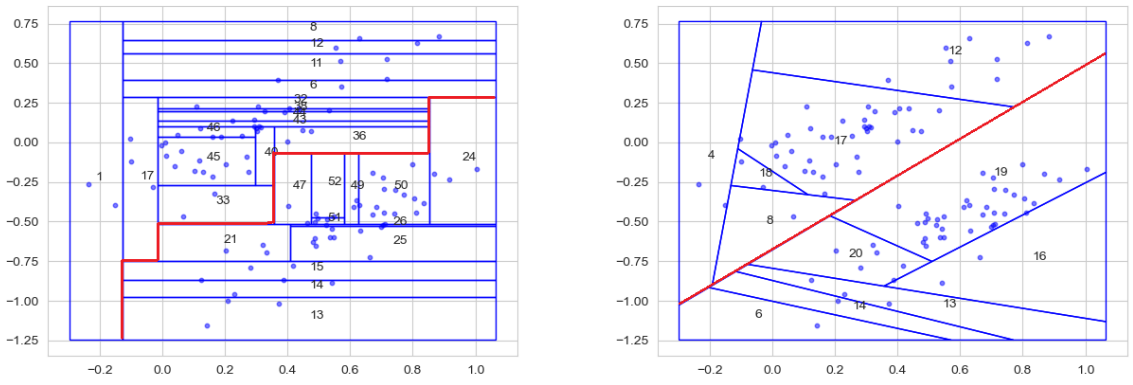

Figure 7: Plot of the final result of *partitioning* phase (the subspaces are all leaf of the corresponding tree), applied to the datasets ***blobs***. The splitting criterion is determined by **Algorithm 1** at the left and **Algorithm 2** at the right.

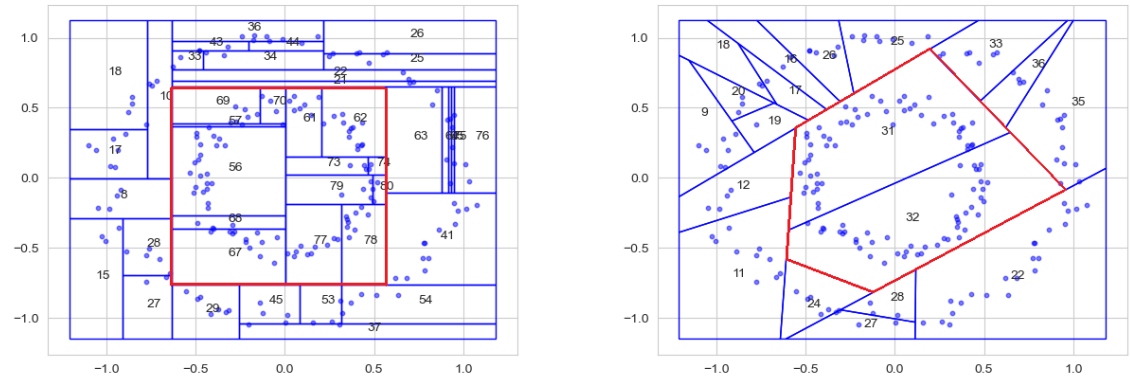

Figure 8: Plot of the final result of *partitioning* phase (the subspaces are all leaf of the corresponding tree), applied to the datasets ***circles***. The splitting criterion is determined by **Algorithm 1** at the left and **Algorithm 2** at the right.

6.2 Algorithm 2 vs Algorithm 3: evaluation of *merging* phase

Here we compare the merging procedures obtained by employing the two different metrics starting from the same output of partitioning phase. Figures 9 shows a space partition of ***moons*** dataset that don't completely separate the two groups of data; in particular, the subspace 14 contains one point belonging to the lower semicircle and all the other points belonging to the upper one. If the groups of data are not well divided by the partitioning procedure, meaning that some final subspace contains points belonging to more than one class, we need the correction of the second metric (**Algorithm 3**) to better assign the clusters.

The left plot of Figure 9 shows the output of the merging procedure with **Algorithm 2** for 2 number of clusters: by only considering the difference between the minimun distance between subspaces and the mean of the minimum distances within the subspaces (what METRIC does), the subspaces 14 and 10 are considered more similar than the subspaces 11 and 17 and, consequently, they are separated later; we see that, in this case, the division into two cluster doesn't correspond to the division into two semicircles.

If we add the correction to the metric (**Algorithm 3**), the score depends on more than one minimum distance between subspaces and it is thus a way to take into account the presence of outliers. The right plot of Figure 9 shows the correct classification of the dataset.

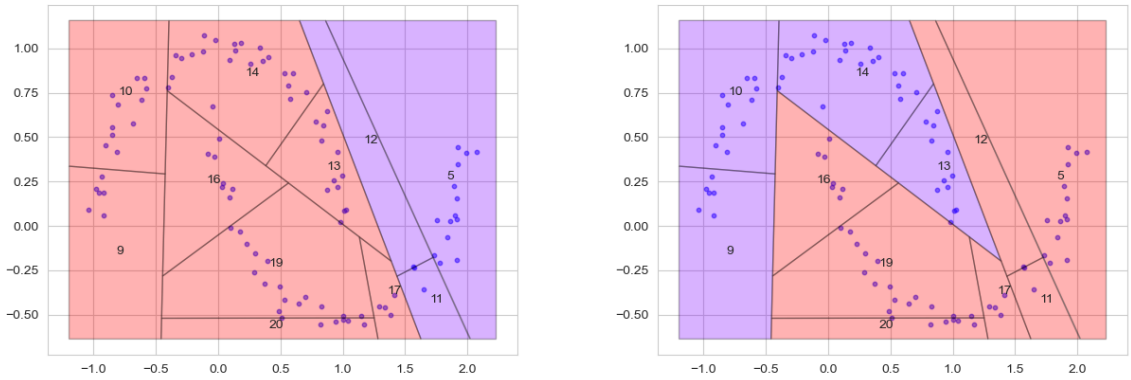

Figure 9: ***moons*** dataset: merging process output for 2 clusters with METRIC (left) and METRIC-WITH-CORRECTION (right)

7 Conclusions

The examples shown in this report suggest that a clustering algorithm based on the Mondrian process has some interesting aspects that it is worth to deeply investigate; some of these are the fact that the output of the algorithm is a probability distribution that estimates how likely each data is to belong to a certain cluster, the automatic determination of the number of clusters and the absence of preferred directions in phase of space partitioning. Some of the aspects that need to be explored are the following: firstly, other toy datasets should be used, in order to verify if the algorithm works with any distribution of data in terms of shape and point density; in particular, despite the code is written in order to work with generic dimension datasets, it is not obvious to obtain a good performance with more than 2-dimensional data. Moreover, it may be possible that, even when the unsupervised classification provides good results, the automatic determination of the number of clusters doesn't work. An other aspect that needs to be further studied is the role of the lifetime lambda and, in particular, how the complexity of the tree is related to the complexity of the dataset; this means that it may be possible to know, before performing the classification, how many splits are necessary to well classify the dataset. Finally, other alternatives to the formulation of the metric and the choice of the hyperplane should be find.

References

[1] Matej Balog and Yee Whye Teh. *The Mondrian Process for Machine Learning*. 2015. arXiv: 1507.05181 [stat.ML].

[2] L Castin. "Clustering with Decision Trees: Divisive and Agglomerative Approach". Master thesis in Computer science. University of Namur, 2017.

[3] Lauriane Castin and Benoît Frénay. "Clustering with Decision Trees: Divisive and Agglomerative Approach". In: *ESANN*. 2018.

[4] Robert Gallager. *6.262, Discrete stochastic processes*. Tech. rep. License: Creative Commons BY-NC-SA. Massachusetts Institute of Technology: MIT OpenCouseWare, Spring 2011. URL: https://ocw.mit.edu/.

[5] B Lakshminarayanan. "Decision trees and forests: a probabilistic perspective". In: June 2016.

[6] Balaji Lakshminarayanan, Daniel M. Roy, and Yee Whye Teh. *Mondrian Forests: Efficient Online Random Forests*. 2015. arXiv: 1406.2673 [stat.ML].

[7] Jaouad Mourtada, Stéphane Gaïffas, and Erwan Scornet. *Minimax optimal rates for Mondrian trees and forests*. 2019. arXiv: 1803.05784 [stat.ML].

[8] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[9] H. Pishro-Nik. *Introduction to probability, statistics, and random processes*. Kappa Research LLC, 2014. URL: https://www.probabilitycourse.com.

[10] Daniel M Roy and Yee Teh. "The Mondrian Process". In: *Advances in Neural Information Processing Systems*. Ed. by D. Koller et al. Vol. 21. Curran Associates, Inc., 2009. URL: https://proceedings.neurips.cc/paper/2008/file/fe8c15fed5f808006ce95eddb7366e35-Paper.pdf.

[11] Amit Saxena et al. "A Review of Clustering Techniques and Developments". In: *Neurocomputing* 267 (July 2017). DOI: 10.1016/j.neucom.2017.06.053.

[12] Nguyen Xuan Vinh, Julien Epps, and James Bailey. "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance". In: *J. Mach. Learn. Res.* 11 (Dec. 2010), pp. 2837–2854. ISSN: 1532-4435.