
Master research Internship


Bibliographic report

# Knowledge Discovery in Multimedia Content by Diversion of Supervised Machine Learning Techniques

**Domain : Machine Learning - Multimedia**

*Author:*
Amélie Royer

*Supervisors:*
Vincent Claveau
Guillaume Gravier
**LinkMedia** - Inria/Irisa

**Abstract**

Knowledge discovery deals with extracting new information from large databases, such as recurrent patterns or structural cues. In this work, we focus on the sub-problem of cluster analysis; In its usual form, it refers to the task of partitioning the data space such that two samples in the same cluster are similar, while those in different ones are not. Clustering algorithms exploit a similarity measure between the data samples, which should be fine-tuned with the data format and the application at hand. This, however, is a complicated task when the user lacks prior knowledge or in case of complex data structures for example.

The purpose of this internship is to investigate an approach for defining such a similarity measure by taking advantage of the discriminative power of state-of-the-art classification techniques. While classification systems usually require a set of samples annotated with their ground-truth classes, several works have shown they can also be exploited for clustering problems, where no such annotation is provided. It is in fact possible to generate artificial, carefully chosen, samples and annotations, in order to deduce a similarity measure from a classifier's outputs trained on this synthetic data set. In this bibliographic review, we introduce the context of the internship, as well as prospective issues and considered applications to multimedia content.

# Contents

## Introduction

In the domain of data mining, the task of knowledge discovery deals with inferring new and previously unknown information from a data set. This obviously covers a wide range of sub-problems. For instance, one may be interested in extracting frequent patterns in a database, which often convey meaningful structural information (e.g., finding frequent items and items associations in customers' purchases), or in trying to obtain a model of the data in the form of a function or probability distribution (e.g., regression, text modeling). In this work, we focus on *cluster* analysis (or *clustering*), the problem of grouping data samples, such that two points lying in the same cluster are similar according to a given distance measure, and conversely, two samples in different clusters are as dissimilar as possible. Many knowledge discovery problems can be formulated as such. For instance, finding frequent patterns in a data set amounts to grouping the different samples according to their similarity so that each cluster represents the "equivalence class" of a potential pattern.

Building a good clustering raises two main issues. The first one is the problem of selecting an adequate underlying similarity measure on the data; This is critical for obtaining clusters that are meaningful once applied to a real-life application. However, this choice is often not obvious, as it is highly dependent on the features used to describe the input data, and on the characteristics we are interested for the cluster analysis to bring to light. The second issue is the clustering process itself, which, taking these distance constraints as input, constructs a partitioning of the data set. The performance of this task is strongly influenced by the data distribution and the application at hand. Finally, due to the rapid advances in data acquisition in recent years, databases are often very large, thus requiring a further need for computational efficiency.

The internship will focus on the first problem, i.e., the choice of a similarity measure between samples. As we will discuss in more details later, there are in fact many cases where usual distances are not appropriate, due to a complex data structure or simply a lack of prior knowledge; hence the motivation to investigate a method for building an adequate measure with few prior knowledge. The starting point of this internship relies on the following thought: intuitively, clustering is very similar to classification, which is the task of associating a data point $x$ with a label $y$ belonging to a set of classes, $\mathcal{Y}$, defined beforehand. However they differ in essence; First, for classification, the class space is known, as well as the number of clusters and what they represent. Secondly, it is a *supervised* task, which means we have access to an annotated database (i.e., a subset of samples and their ground truth classes). Thus the usual workflow is to identify characteristics of the classes using this training database in order to later correctly classify new unseen inputs. Clustering works the other way round as it uses (non-annotated) input samples to form clusters, and only then the result is used to determine specific characteristics of the data samples that may have led to this particular clustering. Clustering is a fundamentally *unsupervised* task, in the sense that it does not require any direct prior information about the clusters. The core idea of this internship is to take advantage of the discriminative power of supervised classification techniques, and divert them for the purpose of cluster analysis. In recent years, this concept has been applied in various domains [**Liu et al., 2000**] such as medical analysis [**Shi and Horvath, 2006**] and knowledge discovery in multimedia content [**Claveau and Ncibi, 2014**] [**Claveau and Gros, 2014**]. More specifically, these articles propose to use supervised learning algorithms to derive a similarity measure on the data set, under the assumption that two objects often classified together share some resemblance; This similarity measure is then used as input of a clustering algorithm. This method has several advantages for the purpose of clustering: in fact, this similarity measure is defined directly for the data and application at hand, hence chances are it is better fitted than a measure defined

without any prior insights. Furthermore, the proposed approach requires few prior knowledge, and in particular, the measure does not have to be explicitly specified by the user. On the other hand, the main difficulty of this approach is that data samples for clustering are originally unlabeled; Hence, an adequate supervised framework must be built before being able to apply supervised classification techniques. This amounts to generating a synthetic, carefully chosen, annotation of the data set to use as input of a supervised algorithm.

In this bibliographic review, we begin by giving a more formal definition of the clustering problem and the usual associated evaluation techniques. Secondly, we describe in further details the notion of diverting supervised learning techniques for the purpose of cluster analysis, on which the internship is based; We focus our explanation around several articles that implemented this approach in recent years. In the last section, we introduce two knowledge discovery problems in multimedia content that will be considered during the internship, namely, *unsupervised labeling of named entities* in text documents and *motif discovery* in audio content. In fact, both frameworks are typical situations where it is difficult to define an adequate similarity measure on the data samples because of the lack of prior knowledge; These applications will be used to validate our approach experimentally. Finally, we discuss prospective issues and early reflections on the upcoming work in the conclusion.

# 1 State-of-the-art of cluster analysis

In this section, we formally define the problem of cluster analysis. Then, we introduce some of the most classical measures used to evaluate clustering techniques. Finally, we present several state-of-the art clustering algorithms, as well as some considerations about the influence of the input distance choice on the resulting clusters.

## 1.1 Problem formulation

### 1.1.1 Definition

Let the data set, $\mathcal{X}$, be a set of $N$ points, $\mathcal{X} = \{x_1 \dots x_N\}$, lying in a space $\mathcal{E}$, the data description domain. A clustering of $\mathcal{X}$ is a set of groups of samples $\mathcal{C} = \{C_1 \dots C_k\}$ that cover the whole data set; i.e., $\forall i \in [\![1; k]\!],\ C_i \neq \varnothing,\ C_i \subset \mathcal{X}$ and $\mathcal{X} = \bigcup_i C_i$. The sets $C_i$ are called *clusters*, and their number, $k$, is not necessarily known beforehand. Note that the term *clustering* refers to the problem of finding the clusters as well as the resulting partition of the data space.

In the most usual definition, the clusters are required to be disjoint ($\forall i \neq j,\ C_i \cap C_j = \varnothing$), thus each data point belongs to only one cluster. This definition of the problem is sometimes called *hard* or *crisp* clustering. However, this requirement is softened in some frameworks. For instance, *hierarchical* clustering forms a hierarchy of clusters, which therefore has a tree-like nested structure. Intuitively, each level of the hierarchy yields a different regular hard clustering, and any cluster of the $n$-th level is included in some cluster of the $(n+1)$-th. Similarly, in *fuzzy* clustering methods, a degree of membership to each cluster is computed for every data sample, instead of directly returning the cluster the point belongs to. This information is especially useful when a point's membership to a cluster is not obvious, for example, for points lying at the border of a cluster. We illustrate how these variants differ in terms of output in Figure 1. In this section, we stick to the usual framework as we introduce the state-of-the-art for hard clustering. However, the main idea we aim to present in this bibliographic review, i.e., defining a similarity measure by diverting supervised techniques, is not subject to such a restriction and can be applied with any type of clustering.

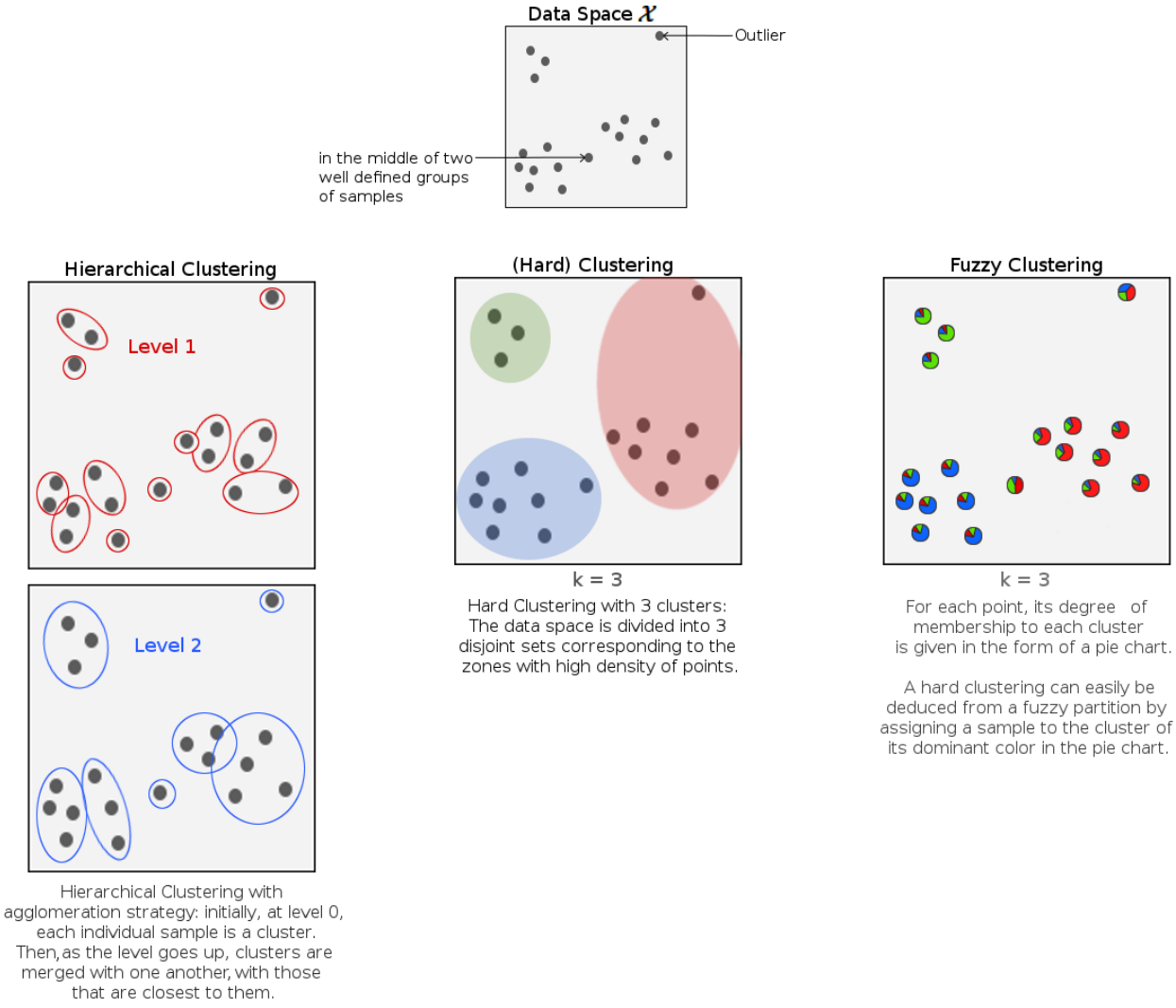


Figure 1: Hard Clustering and its variants illustrated. On the left is a hierarchical clustering built with agglomeration strategy; The middle column is an example of hard clustering with 3 clusters; Finally, the right column presents the corresponding fuzzy partition, with 3 clusters as well.

### 1.1.2 Goals and issues

Intuitively, a good clustering should minimize the *intra-class* distance (i.e., points in the same cluster should be similar) and maximize the *inter-class* distance (i.e., any two points in different clusters share little resemblance). This notion of similarity is defined by an input distance $D : \mathcal{E} \times \mathcal{E} \to \mathbb{R}$.

The first issue is the process of building the clusters, which mostly depends on the data distribution and the expected number of clusters. For example, the data set may contain outliers, corresponding to regions with few observations. In these cases, the algorithm has to choose between adding the point to another cluster (which may lead to unbalanced or distorted clusters since outliers are far away from others), or creating a new separate cluster (which may lead to unnecessary partitions and a surplus of clusters). Secondly, clustering techniques usually do not rely on any

(or few) prior information on the clusters. This is an unsupervised framework and this lack of information is one of the challenges to overcome. However, indirect knowledge on the clusters can be introduced trough the parameters of the algorithm. This raises questions such as how to choose (or dynamically determine) the number of clusters, or whether to search for specific-shaped clusters (e.g., ellipses, bounding boxes) or not.

The second main issue is the choice of the input distance, $D$, which is strongly linked with how the data is described in practice, and should reflect the characteristics we want the final clustering to exhibit. This criterion strongly impacts the resulting clusters and how meaningful they are, i.e., if they make sense in practice for the considered application, or not. For instance, let us consider a clustering problem on audio signals described as real-valued vectors. If we are interested in grouping together exact repetitions of the same pattern, then a simple Euclidean distance is befitting. However, in most applications, audio signals are pre-treated and possibly re-sampled, hence measures taking temporal distortions into account, such as dynamic time warping (DTW), are better suited. Both issues are addressed in further details in the remaining of the section.

## 1.2 Building a partition

### 1.2.1 Taxonomy of cluster analysis algorithms

Cluster analysis is a popular problem that has been studied in many domains, which led to a wide range of different clustering algorithms. However, for the same reason, many algorithms tend to be best suited for a specific application, and do not perform well on others. Based on a 2005 survey [**Rui and Wunsch II, 2005**], we describe several usual clustering techniques in this subsection.

**Error-based optimization.** Cluster analysis can be formulated as an optimization problem, where the clusters are refined until some error function reaches a minimum. A well-known example is the *k-means* algorithm, which aims at minimizing the distances from any sample to the centroid (mean) of its cluster. Note that this algorithm does not ensure to find a global minimum. There exists other clustering algorithms yielding an exact solution, but they are often more expensive in terms of computational cost. The core idea of $k$-means is to iteratively build clusters around $k$ centroids until the whole partitioning is stable (i.e., no or few changes between consecutive iterations); We give an explanation of the algorithm in Figure 2.

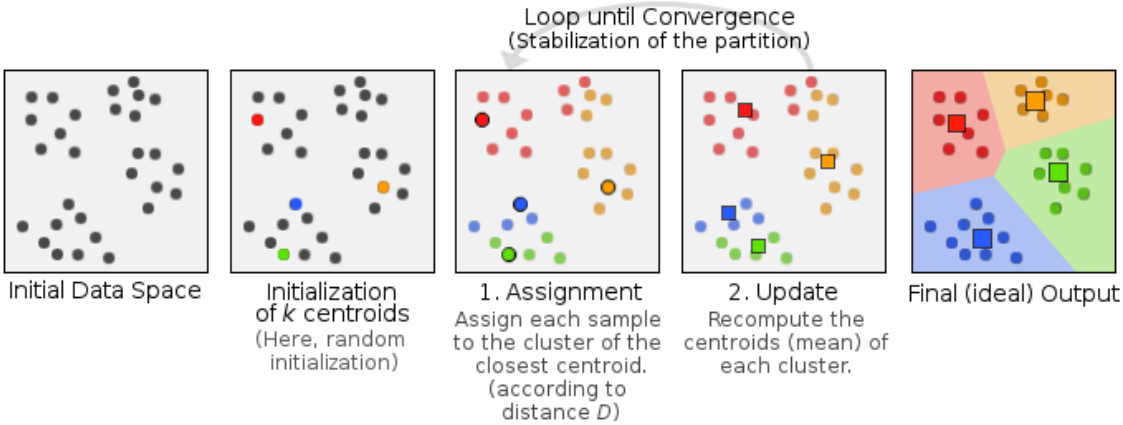


Figure 2: Workflow of the $k$-means algorithm for $k = 4$; The input parameters are the data samples, the distance $D$, and the number of clusters $k$. The algorithm outputs a hard clustering of $k$ clusters.

The main advantage of the $k$-means algorithm is its simplicity. However, it requires the number of clusters, $k$, which may be unknown, as input. Furthermore, the initialization of the centroids influences the result and should be carefully tuned. Another short-coming is that the resulting clusters have a constrained shape and tend to be of similar size; In fact, they can be seen as Voronoï cells centered on their centroid. Hence, $k$-means are best suited for convex clusters. Finally, the input distance, $D$, has to be defined on the whole domain, $\mathcal{E}$, in order to compute distances between the data samples and newly-computed centroids.

On the contrary, some techniques only need $D$ to be defined on the data set $\mathcal{X}$. This is the case for the *$k$-medoids* algorithm (or *PAM* clustering: *P*artition *A*round *M*edoids). It is a variant of $k$-means where instead of choosing a centroid as the cluster representative, a medoid (the most centrally located point in the cluster) is used: these are conceptually equivalent, with the difference that a medoid always belongs to the data set $\mathcal{X}$.

**Density-based clustering.** Density-based clustering methods aim to discover clusters as regions of the data space containing a high density of points, separated by empty or low-density regions. A classical example is the *DBSCAN* algorithm (*D*ensity-*B*ased *S*patial *C*lustering of *A*pplications with *N*oise). It takes as input a distance $\varepsilon$ (size of considered neighborhoods) and a minimum support $m$ (which controls the clusters' density). The idea of the algorithm is to iteratively expand clusters by browsing each sample's $\varepsilon$-neighborhood (i.e., all points lying at a distance less than $\varepsilon$ from the considered sample). Let $x$ be a sample visited during the expansion of a cluster $C$: if $x$'s neighborhood contains more than $m$ points, it is considered as a dense region and $x$ is added to the cluster $C$ (if not already assigned to another cluster). On the contrary, if the neighborhood contains less than $m$ points, then $x$ belongs to a sparse region and is marked as outlier.

This algorithm has several advantages over the $k$-means. In fact, DBSCAN does not require the number of clusters as input, and is able to manage and detect outliers. Furthermore, no computation of new distances on the data domain is required, and finally, it is suited for arbitrary-shaped clusters. However, DBSCAN necessitates information about the size of neighborhoods to explore, $\varepsilon$, as well as the clusters' average density (support parameter $m$). Furthermore, both parameters are fixed inputs, which leads the algorithm to search for clusters with similar density.

**Graph clustering** Graph clustering algorithms identify clusters as connected components in a graph, such that the total edges' weight is low between two different components and high inside a component. To fit the framework of graph clustering, the data set can be seen as a graph whose vertices are the data samples, and whose weighted edges represent the distance function $D$ (possibly with a threshold to dismiss edges between very dissimilar points).

Graph clustering algorithms are usually based on random walks on the graph, as for example the *M*arkov *C*luster *A*lgorithm (*MCL*)[1]. This algorithm is based on the assumption that if a dense cluster is visited during a random walk on the graph, then the walk will likely stay in the same region until many of its vertices have been visited. The clusters are built by iteratively updating the transitions weights of the walk, forcing the flow to concentrate in high density regions. Intuitively, this family of algorithms is similar to density-based clustering (focus on high density regions, random walk on the samples...) and has similar advantages over the $k$-means. The main difference is that MCL does not require a minimum density threshold as input parameter, but therefore can not use such a threshold to mark outliers, as it was the case with DBSCAN.

[1] MCL - a cluster algorithm for graphs, http://micans.org/mcl/

**Spectral clustering.** From a general point of view, spectral clustering refers to the family of clustering algorithms that modify the original data set, in order to obtain a new space where the clustering problem is easier, and then, apply simpler clustering techniques (such as $k$-means) in this new space. The most classical example is to use the reduced eigenspace of the similarity matrix instead of the original data set. The intuition is the same as dimension reduction methods which work in the eigenspace rather than the original space in order to exhibit the most important directions of the original data set (e.g., principal component analysis). Finally, spectral clustering algorithms can be reformulated as graph clustering algorithms, and share the same advantages.

### 1.2.2 About the distance choice

All algorithms presented before require explicit information about the similarity between samples. As already mentioned, this similarity measure is an essential factor, as its discriminative power will strongly influence the resulting clusters. Note that instead of true metric distances defined on the whole data domain, $\mathcal{E} \times \mathcal{E}$, some algorithms use partial distances defined on the data set $\mathcal{X} \times \mathcal{X}$ only (e.g., $k$-medoids). As a matter of fact, it is always possible to extend such restricted distance to the whole domain, for instance by building a Mahalonobis distance respecting the restricted distance constraints using metric learning [**Kulis, 2013**]. Another solution is to project the data samples on the metric space $\mathbb{R}^d$ such that the projections respect the constraints; This can be done using the *MDS* (*M*ulti *D*imensional *S*caling) algorithm for example [**Cox and Cox, 2000**].

When the data domain is a metric space, a natural choice is to rely on its underlying distance. For example, if data samples are real-valued vectors of numerical attributes, the Euclidean distance, or more generally Minkowski distances, are commonly used. There are many other distances to choose from, and each of them leads to particular geometrical properties on the clusters. For instance, the $k$-means algorithm is often used with the Euclidean distance, which tends to form hyper-spherical clusters. The same holds for categorical data, i.e., attributes whose value ranges into a discrete interval, with no total order on it. The task is more complicated as there is no underlying metric, and often depends on the meaning of each attribute. For instance, a simple similarity measure for categorical data is the Hamming similarity, which counts the number of categorical attributes for which two samples have the same value. This obviously eludes a lot of information, but little can be done when there is no sufficient prior knowledge. See [**Boriah et al., 2008**] for a more detailed survey of similarity measures for categorical data.

Lastly, in more complex cases (e.g., mixed attributes type, complex data structures, particular application), there is often no conventional similarity measure to use. For example, if data samples are described by histograms, they can be seen as real-valued vectors whose order and relative positions of the coordinates matter. Depending on the application, a bin-to-bin comparison may not be as befitting as a similarity taking into account the inter-bin correlation (e.g., in images, the bins corresponding to "red" and "dark red" are visually similar, contrary to "red" and "cyan"). This example illustrates why we aim to build a similarity with no or few prior knowledge required, which is hopefully better suited to the task of clustering than a similarity chosen "by default" by the user.

## 1.3 Assessing the quality of a partition

As mentioned before, a good clustering maximizes the similarity between points in the same cluster, and minimizes the similarity between different ones. This property is evaluated through *internal* evaluation, by measures such as the *Dunn Index*.

**Dunn index.** The *Dunn index* for a clustering $\mathcal{C}$, $DI(\mathcal{C})$, relies on two measurements: $\Delta_i$ is the diameter of the cluster $C_i$ and represents the notion of intra-class distance. For instance, one can choose $\Delta_i$ to be the mean of all pairwise distances in $C_i$, or the maximum of these distances. Similarly, we define $\delta_{i,j}$, the inter-class measure, as a distance between clusters: e.g., the distance between the clusters' centroids. Given these, the Dunn index is then defined as:

$$DI(\mathcal{C}) = \frac{\min_{i \neq j} \delta_{i,j}}{\max_p \Delta_p}$$

We observe that a high value of $DI(\mathcal{C})$ means large inter-class distances and low intra-class distances, which exactly corresponds to the idea of a good clustering.

However, internal evaluation only ensures that the clustering process divides the space in a satisfying way, not that the resulting clusters are meaningful for the real-life application at hand. Therefore, a second type of evaluation, namely *external* evaluation, is needed. For a given application, a data set $\mathcal{X}$ and input distance $D$, we denote the optimal clustering as $\mathcal{C}^*(\mathcal{X}, \mathcal{D})$; $\mathcal{C}^*$ is often given as ground-truth, i.e., annotated data used during the evaluation process only. External evaluation compares the clustering obtained by the algorithm against the optimal one. We present an example of such measures below; See for example [**Halkidi et al., 2001**] for a more exhaustive study.

**Rand index.** The *Rand index* is an intuitive mean of comparing two clusterings. Mathematically, it corresponds to the ratio of pairs of data points on which both clusterings agree; The higher the Rand index, the closer the clusterings are. Given two clusterings $\mathcal{C}^1$ and $C^2$ of $\mathcal{X}$ we define the Rand index $RI(\mathcal{C}^1, \mathcal{C}^2)$ by:

$$RI(\mathcal{C}^1, \mathcal{C}^2) = \frac{\#\{\{x, y\} \mid x \neq y \text{ and } (s(x, y, \mathcal{C}^1) \text{ iff } s(x, y, \mathcal{C}^2))\}}{\binom{N}{2}}$$

where $s(x, y, \mathcal{C}) =$ `True` *iff* $\mathcal{C}$ groups $x$ and $y$ in the same cluster.

A major inconvenient of the Rand index is that its expected value is different for any two different clusterings, thus introducing a bias in the comparison. The corrected-for-chance version is the adjusted Rand index, whose expected value is constant of value 0 and maximum value is 1. It is currently one of the most usual evaluation measures for clustering tasks.

Following this introduction to clustering, we introduce in the next section the concept of diverting supervised techniques to infer a similarity measure on the data set for the purpose of clustering.

## 2 Clustering by diverting supervised learning approaches

In the recent years, several articles have used the idea of diverting supervised machine learning techniques for applications to cluster analysis. In this section, we begin by proposing an unified framework for these approaches, in order to explain the intuition behind them. Then, we present several specific applications in more details.

### 2.1 Learning a similarity

As mentioned previously, the task of classification is closely related to clustering. Formally, the goal is to build a classifier $f$ that maps any input sample $x \in \mathcal{X}$ to a class $y \in \mathcal{Y}$; where $\mathcal{Y}$ is a set of classes, known beforehand. Furthermore, in the usual framework of (fully) supervised classification, the user is provided with an annotated data base, i.e., samples which are labeled with

their ground-truth class; These are used as a training set to learn a classifier. On the opposite, cluster analysis is an unsupervised task, which means the input data is unlabeled, and we usually have no, or very few, information about the optimal clusters. Therefore, we need to provide clustering algorithms with an explicit similarity measure, while classification algorithms are able to learn the needed information through their training step.

Classification algorithms can be used to infer a notion of similarity on the data set from their outputs. In fact, when two objects are classified together, they are similar in the context of the classifier's internal representation. We propose to formalize this idea by defining, for any classifier $f$, a binary similarity function, $s_f$, which is induced by the classifier's outputs:

$$\forall x,\ y \in \mathcal{X},\ s_f(x,y) = \begin{cases} 1, & \text{if } f(x) = f(y) \\ 0, & \text{otherwise} \end{cases}$$

This measure is easily refined into a continuous one when the classifier yields probabilistic outputs, i.e., for each sample, the classifier outputs a degree of membership to each class. Another way of refining this similarity is by combining several classifiers, for example by computing the mean of each one of their induced $s_f$ functions. This definition is more robust to the bias introduced by the learning process (e.g., choice of the training set) than the one with only one classifier.

Such similarity measure is an adequate candidate as input to a clustering algorithm. In fact, we can take advantage of the accuracy of state-of-the-art classifiers and of the background in the supervised learning field in order to refine the measure and investigate its different properties. Secondly, this similarity measure can be defined on any type of data, as long as a befitting classifier exists; This is especially useful in frameworks where usual metrics are not applicable. For example, in [**Claveau and Gros, 2014**], the authors divert supervised ILP (inductive logic programming) techniques in order to define a similarity on relational data for clustering purposes. In fact, most usual distance measures do not apply in this case, as the database does not follow the classical model where a sample is a real-valued vector (attribute-value model). Finally, this definition does not require any prior knowledge on how to specifically construct the similarity measure. This is in fact mostly managed through the supervised classification. Furthermore, since clustering problems often have a supervised counterpart, it naturally leads to choosing the corresponding classifier.

However, in order to to build such a measure for clustering, we first need to extend it to the unsupervised framework. This is done by generating adequate synthetic data and annotation which are then given as input of supervised classifiers in order to define the aforementioned similarity. In the two next subsections, we detail some applications which applied this approach to cluster analysis; The first analysis focuses on an application to the domain of medical analysis, while the second subsection deals with an application to text mining.

## 2.2 Random forest clustering

One of the first examples of applications of such technique, to the best of our knowledge, appears in a 2006 article [**Shi and Horvath, 2006**] and the corresponding technical report [**Shi and Horvath, 2005**]. This article introduces the "Random Forest dissimilarity", built by diverting a set of supervised *decision tree* classifiers, for the purpose of cluster analysis. It is then used as input of a $k$-medoids algorithm, and applied to a medical analysis problem.

This work is partially based on an older article [**Liu et al., 2000**]; Here, the authors do not explicitly introduce the notion of similarity but rely on the same idea to use a decision tree in an unsupervised framework for clustering.

### 2.2.1 Random forests

Random forest classifiers [**Breiman, 2001**] are *bagging* procedures based on decision trees. Decision trees are very intuitive supervised classifiers which take as input samples described by vectors of attributes. The core idea is to split the data space on the attribute that separates it the best (the split criterion is evaluated by measures such as the entropy). This process is iterated on the resulting sub-trees, until no attribute is left or all samples in the sub-tree belong to the same class. For categorical (discrete) attributes, the split yields as many sub-trees as possible values; For numerical (continuous) attributes, we use thresholding rules, yielding 2 subtrees each. In Figure 3, we present an example of decision tree.

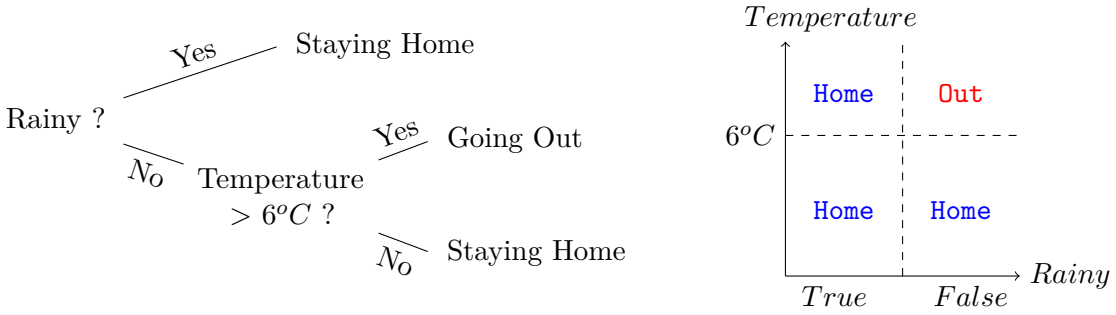


Figure 3: On the left is an example of decision tree with mixed binary and numerical attributes; On the right is the corresponding partitioning of the attributes space.

A *bagging* process is a set of classifiers, trained independently, possibly with different parameters. Each sample is sent through each classifier of the bagging, and their outputs is then combined to yield the final decision. In the case of random forests for example, the class of a new input sample is given by a majority vote on the outputs of the decision trees in the forest.

### 2.2.2 Diverting the random forest and decision tree classifiers

Both articles rely on the idea of using decision trees to distinguish the original unlabeled data from a set of synthetic data samples, $\mathcal{S} = \{s_1 \dots s_P\}$. In this particular framework, samples are described by a set of attributes, i.e., real-valued vectors. We denote by $\omega_o$ the class assigned to the original samples, and by $\omega_s$ the one of the synthetic data. Then, $T$ decision trees $\{f_1 \dots f_T\}$ are trained on different subsets of $\mathcal{S} \cup \mathcal{X}$ and used for clustering purposes.

In [**Liu et al., 2000**], a single decision tree is used ($T = 1$), with the goal of distinguishing dense regions from sparse ones in the original unlabeled data set. The synthetic data set is generated as $P$ randomly uniformly sampled points on the whole data domain. The authors observe that dense regions in the original data set should contain more original than synthetic samples, and thus be separated from the sparse regions, which logically contain more synthetic points. Based on this observation, the decision tree is trained on $\mathcal{S} \cup \mathcal{X}$, and the partition induced by the tree is returned as the final clustering (see Figure 3 for an illustration of how decision trees induce a partition of the attributes space). The main disadvantage of this method is that it only generates hyper-rectangular clusters, which may not be suited for real-life applications; In fact the real-life experiments in the original paper lack details.

In the second article [Shi and Horvath, 2006], synthetic samples are generated by independently sampling each attribute from its empirical marginal distribution in the original data. Then, each decision tree is trained on a different subset $T \subset \mathcal{S} \cup \mathcal{X}$; The rest of the samples is run through the tree, yielding a binary similarity measure for these samples, as defined previously. The final similarity on the original data set $\mathcal{X}$ is obtained by averaging these measures for the whole forest:

$$s = \frac{\sum_i s_{f_i}}{T}_{|(\mathcal{X}\times\mathcal{X})} \tag{1}$$

As for the number of trees $T$ and their individual training parameters, the authors use classical parameters of bagging processes and random forests.

In this framework, the only major difference between the synthetic and original samples is the independence of the attributes; This drives the decision trees to focus their splits on variables that are dependent in the original data. In the specific application to medical analysis proposed in the original article, those appear to be important variables, as the resulting clusters are more meaningful than with the Euclidean distance (in terms of post-survival chances of the patients). Furthermore, the use of random forests enable the authors to infer which variables are most important to explain the resulting clusters. This interpretability is especially useful for medical analysis.

## 2.3 Unsupervised labeling of named entities

Our second example [Claveau and Ncibi, 2014] deals with the task of labeling named entities in text documents. Named entities are textual information about the actors, placs or time of an event, e.g., names of persons, organizations, places, dates. Given a set of unlabeled named entities and their textual context, the goal is to correctly categorize them. This has applications for automatic description of textual contents for example.

### 2.3.1 Data domain and conditional random fields

As opposed to the previous framework, textual data has a sequential structure, which requires a different type of classifier, as random forests would perform poorly on this kind of data.

The first step is to identify all named entities in the text since those are the samples we want to cluster, not the other, common, words. This step is also a clustering/classification process (distinguishing named entities from normal words), however it is totally separated from the task at hand (categorizing the named entities); Therefore, it is easier to assume we already known which words are the named entities. This is represented by a *BIO* labeling of the input data samples: "O" labels are associated with common words. For named entities, "B" labels are associated with the beginning word and "I" labels with the remaining ones, if any.

The second step is the labeling of the named entities, i.e., we want to complete the BIO information with the type of the named entity: O labels stay the same, but BI labels should be completed by a type; e.g., B-date, I-date... This is the actual clustering step we want to investigate. In fact, defining a similarity between named entities is a difficult task because we need to take the word itself into account, but also the context in which it appears. This motivates the idea to infer a similarity measure by diverting Conditional Random Fields (CRF), which are state-of-the-art classifiers for labeling and segmenting text, making them a natural choice. They were first introduced in 2001 [Lafferty et al., 2001], and a more detailed study can be found in the following technical report [Klinger and Tomanek, 2007].

CRF are widely used for applications on textual data because they efficiently capture sequential information such as relationships between data samples and their annotation, as well as other samples and annotations around them. For computation sake, linear-chain CRF are used in practice. This means that for a sentence, at each position, we only express a relation between the current label, the preceding one, and all words around. These relations are defined by a set of binary feature functions $f_j(\mathbf{x}, y_t, y_{t-1}, t) : \mathcal{X}^d \times \mathcal{Y} \times \mathcal{Y} \times \mathbb{N} \to \{0; 1\}$ which correspond to rules automatically inferred from the training set, rather than defined manually. For instance, the rule $x_t = "February" \wedge y_t = "Date" \wedge y_{t-1} = "Preposition"$ means that the word "February", when following a preposition, refers to a date. Each feature function $f_j$ is then associated with a weight $\lambda_j$, determined by an optimization strategy. Finally, the probability of a sentence $\mathbf{x} = \{x_1 \dots x_d\}$ being annotated by a labeling $\mathbf{y} = \{y_1 \dots y_d\}$ is given by the following equation:

$$\mathbb{P}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} \prod_t \exp\left(\sum_j \lambda_j f_j(\mathbf{x}, y_t, y_{t-1}, t)\right) \qquad \text{where } Z \text{ is a normalization constant.}$$

### 2.3.2 Synthetic data generation

Recall that in the random forest example, the goal was to generate synthetic data different from the original one, yet similar enough, so that the classifiers focus on the small, non-trivial, and hopefully meaningful, differences between the two classes. Applying the same principle in the current case would amount to generate true sentences from natural language, with named entities distributed in a slightly different manner than the original data; The generation step would be too complicated, since it requires generating positive examples of a language with correct use of named entities. Instead, the authors choose to use the original data set $\mathcal{X}$ annotated with $n$ synthetic labels.

In practice, we introduce $T$ CRF: $\{f_1, \dots f_T\}$. Each classifier $f_i$ is trained on a training subset $\mathcal{T}_i \subset \mathcal{X}$ where every sample is uniformly randomly annotated with one of the $n$ synthetic labels. The remaining samples, $\mathcal{X} \setminus \mathcal{T}_i$, are run through the CRF, which yields a binary similarity $s_{f_i}$ for these samples, as defined previously. In addition to this, the authors penalize samples grouped together in a class that is over-represented in the training set; In fact the larger a class, the less informative it is from a discriminative point of view; This takes the form of a weighting function $w$. For each classifier, we therefore define its weighted induced similarity measure, $s'_{f_i}$: $\forall x_p,\ x_q$ grouped in the same class $y_j$, $s'_{f_i}(x_p, x_q) = w(y_j) \times s_{f_i}(x_p, x_q)$. The final similarity measure is defined as the average of the induced $s'$ functions for all CRF, similarly to Equation 1.

The main issues of this method are the number of synthetic labels and their distribution. In fact, if too few labels are used then the groupings are too general and not very informative. On the other hand, if the number of labels, $n$, is high, there will be fewer test samples classified together, i.e., each similarity measure brings little information, hence we need to increase the number of classifications, $T$. Moreover, the synthetic annotation influences the distribution of the training set; The approach presented here already takes into account the over-representation of a class. Similarly, since the annotation is a random process, it is possible that some samples are often grouped together in the training database, while some others are not. This could introduce a bias in the training process and in the resulting similarity. Finally, the article does not address the problem of the number of classifiers to use. This however directly impacts the computational cost of the process, which, even if easily parallelizable, requires many training and testing steps.

In the next section, we present the two applications that we will investigate during this internship, in the context of diverting supervised learning techniques.

# 3 Applications to multimedia content

The core idea of this internship is to further investigate the similarity measure introduced in the previous section. Several approaches have successfully applied this technique in recent years, however we believe it can be refined and eventually improved. For our experimental validation, we will focus on two applications on multimedia content that we describe in this section.

## 3.1 Unsupervised labeling of named entities in text documents

This application was presented in the previous section, and will be the first one to be studied in the internship, with the purpose of extending the existing work. This study is motivated by the fact that the method has already been proved to perform well on this application and that we are in a typical framework where choosing an adequate similarity for clustering is difficult: in fact, each named entity occurs several times in the data set, in different contexts; All this information characterizes the entity, hence has to be taken into account and conveyed by the similarity measure.

In the original contribution [**Claveau and Ncibi, 2014**], the authors compare the CRF diversion to a manually-defined similarity measure, as input of the same graph clustering procedure. Note that this task in an unsupervised framework is rather new, as most approaches to named entities labeling use supervision; Therefore there does not exist many state-of-the-art methods to compare against. The CRF diversion is evaluated against the similarity measure introduced in [**Ebadat et al., 2012**]: in this article, the authors propose to describe a named entity with several bag of words features (i.e., vector counting the words frequencies) built from the context of each occurrence of the entity (the context is represented by a limited window centered around the word). This description format is then coupled with usual similarities such as the *Cosine* similarity for example. This approach in fact obtains good results, since the similarity definition is fine-tuned to the task and data, but it illustrates the fact that defining an adequate similarity is not easy; Furthermore, it is outperformed by the aforementioned similarity measure by CRF diversion, as the latter captures sequential information better. This shows that the proposed approach is promising.

## 3.2 Audio motif discovery

### 3.2.1 Presentation and state-of-the-art

The second application deals with audio motif discovery and will be more exploratory, as the similarity we defined previously has never been used in this context. The problem of audio motif discovery is to extract recurrent patterns in audio signals. These patterns are called *motifs*, analogously to the fields of bioinformatics and genomics. From a discovery point of view, those patterns carry meaningful information about the data. For instance, in real-life audio signals (e.g., conversations, radio shows), recurring motifs are often directly linked to the topic of the content, and thus may be used for automatic audio summary, or indexation by keywords.

This is typically an unsupervised framework as we do not know beforehand what kind of motifs to search for, or their number of occurrences. Furthermore, we are interested on working directly on the acoustic signals, not on a textual representation of the audio signal. The main difference is that we are in a continuous, instead of discrete, setting, and that techniques from natural language processing do not apply. However, by using directly the audio signal, we preserve acoustic information, and save the effort of translating the audio to a text representation. This particular setting was, among others, addressed in [**Muscariello et al., 2009**] and [**Park and Glass, 2008**].

### 3.2.2 Contextualization

Our approach is motivated by the fact that audio motif discovery can be formulated as a clustering problem, for which there exists no obvious adequate similarity measure. Hence, to address this lack of prior knowledge on which similarity measure to use, we propose to define a measure by taking advantage of supervised techniques that fit this context. Similarly to the labeling of named entities, the clustering process can be broken down in two steps: the first one is to find the audio pieces that are actually repeated (analogously to named entities). Those are the motif candidates; The second step is to group these pieces, such that two of them falling in the same cluster means they are occurrences of the same motif.

The approach in [**Park and Glass, 2008**] first applies silence detection to the audio signal in order to split the data into smaller excerpts. Then, they use a distance called *Segmental Dynamic Time Warping* (SDTW); The idea is to compute several local Dynamic Time Warping measures between the two words to compare. Obviously this step is computationally expensive as they have to do so for each possible pairs of the previously extracted segments. Then, a pruning step is applied to keep only pairs of signals with a good enough similarity (motif candidates), and a final step of graph clustering is applied on these candidates using the SDTW measure as similarity. On the contrary, in [**Muscariello et al., 2009**], the focus is set on the detection of the motif candidates. The authors use an efficient dictionary building in order to extract all possible candidates in one-pass; Finally, they also use a similarity based off DTW, called Segmental Locally Normalized Dynamic Time Warping (SLNDTW), for gathering the audio segments corresponding to the same motif. As previously, we will focus on the second step, i.e., how to assess if two audio signals are occurrences of the same motif.

Defining a suitable similarity is difficult as two audio excerpts may be very different in tone or speed, but still correspond to the same word. Furthermore, as with named entities, the context in which the word occurs is an important information too. This motivates us to build a similarity measure by diverting classification techniques to this particular context. However, the data domain is different from the problem of labeling named entities, as we have continuous features here: most implementations of CRF use categorical data, hence may not fit this framework. A first solution would be to discretize the features and use CRF again. Another solution is to use hidden Markov models (HMM) in place of CRF, since they are classifiers more fitted to audio signals. As a matter of fact, HMM and CRF are very similar conceptually, thus going from CRF to HMM makes sense.

## Conclusion

In this bibliographic review, we have introduced the task of clustering as a tool for knowledge discovery problems. One of its most important factors is the choice of a similarity measure between the data samples, defined before applying the clustering algorithm. However, the usual distances used in such tasks are not fit for all applications. In fact, it is sometimes needed to define more complex and fine-tuned similarity measures, in order to build meaningful and accurate clusters. For this internship, we aim to investigate an idea proposed in the recent years: using classification techniques, which are the supervised counterpart of clustering algorithms, to build a befitting similarity. This enables us to evade the problem of having to define such similarity "manually" or to choose a similarity by default, but rather, we use the discriminative power of existing supervised techniques. The underlying assumption is that two samples often classified together are similar

in the context of the classifier, and this is also motivated by the fact that most state-of-the-art classifiers nowadays yield very good accuracy.

The internship will deal with two applications on multimedia content. The first one is the unsupervised labeling of named entities. The previously mentioned method has already been studied in this framework, thus it will mostly be a work of expansion hopefully leading to improvements. The second application is audio motif discovery. It is a new framework to investigate, but as we explained previously it appears to be promising and adequate for the use of such similarity. Based on the reading of articles that applied this idea of diverting supervised technique for the purpose of building a similarity measure, we identify several issues that can be seen as prospective areas of work for the internship.

The first and probably most important point in the definition of the measure is the synthetic data generation. In the applications with random forests, the user incorporates indirect prior knowledge through the choice of this synthetic annotation: in fact, the decision trees are used to distinguish observed from synthetic data, thus the artificial annotation directly influence the focus of the classifier. On the contrary, in the second case with CRF, generating negative synthetic examples was more complicated, hence the authors resort to annotate the observed samples with $n$ random fake labels, where the parameter $n$ needs to be investigated.

Another problem is the combination of the similarities induced by each individual classifier. In the previous examples, a (possibly weighted) average is used. Several other methods can be investigated (weighted mean, squared mean...). Moreover, we could consider the classifiers to be sequential and not independent, in order to use the results of previous classifications to iteratively refine the similarity. With this formulation, it is similar to the problem of *prediction with expert advice*: each classifier $f$ is an expert represented by its associated binary similarity function $s_f$, and we aim at defining the best possible final similarity from their decision. Similarly, as we have observed in the case of the CRF example, it may be useful to use different weighting techniques in order to balance the bias introduced by the training set. Controlling this bias is a common problem for bagging procedures. We can therefore study existing results from this field.

Finally, the question of the stopping criterion is important in order to control the computational cost. From this point of view, we observe that the process of defining the similarity is the same as an indexing procedure (each classifier yields a different partitioning of the data space; This yields an indexing of a sample as the vector containing the classes output by each classifier). It is in fact similar to iterative indexing algorithms, such as *Locality Sensitive Hashing* (LSH). In LSH, we repeatedly hash the space (by random projections on lines) until reaching a satisfying partition. This application, which is closer to indexing than knowledge discovery tasks, is for example used in [**Perbet et al., 2009**]: the authors apply random forests to a set of unlabeled samples, annotated with synthetic, uniformly sampled, annotations. This yields a vector-index for each sample, which is then used to define a graph on which graph clustering is applied. By seeing the problem as an indexing strategy, we could make use of the existing studies on the problem of the stopping criterion and optimal number of hashing steps in similar procedures, such as LSH for example.

## References

[Boriah et al., 2008] Boriah, S., Chandola, V., and Kumar, V. (2008). Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the eighth SIAM International Conference on Data Mining*, pages 243–254.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Journal of Machine Learning*, 45(1):5–32.

[Claveau and Gros, 2014] Claveau, V. and Gros, P. (2014). Clustering de données relationnelles pour la structuration de flux télévisuels. In *14 ème conférence Extraction et Gestion des Connaissances, EGC 2014*.

[Claveau and Ncibi, 2014] Claveau, V. and Ncibi, A. (2014). Knowledge discovery with CRF-based clustering of named entities without a priori classes. In *Conference on Intelligent Text Processing and Computational Linguistics CICLing*, volume 8403 of *LNCS*, pages 415–428.

[Cox and Cox, 2000] Cox, T. F. and Cox, M. (2000). *Multidimensional Scaling, Second Edition*. Chapman and Hall/CRC, 2 edition.

[Ebadat et al., 2012] Ebadat, A.-R., Claveau, V., and Sébillot, P. (2012). Semantic Clustering using Bag-of-Bag-of-Features. In *CORIA - COnférence en Recherche d'Information et Applications*, pages 229–244.

[Halkidi et al., 2001] Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145.

[Klinger and Tomanek, 2007] Klinger, R. and Tomanek, K. (2007). Classical probabilistic models and conditional random fields. Algorithm Engineering Report, Technische Universität Dortmund.

[Kulis, 2013] Kulis, B. (2013). Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364.

[Lafferty et al., 2001] Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289.

[Liu et al., 2000] Liu, B., Xia, Y., and Yu, P. (2000). Clustering through decision tree construction. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, CIKM '00, pages 20–29.

[Muscariello et al., 2009] Muscariello, A., Gravier, G., and Bimbot, F. (2009). Audio keyword extraction by unsupervised word discovery. In *INTERSPEECH 2009: 10th Annual Conference of the International Speech Communication Association*.

[Park and Glass, 2008] Park, A. and Glass, J. (2008). Unsupervised pattern discovery in speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(1):186–197.

[Perbet et al., 2009] Perbet, F., Stenger, B., and Maki, A. (2009). Random forest clustering and application to video segmentation. In *Proceedings of the British Machine Vision Conference 2009*.

[Rui and Wunsch II, 2005] Rui, X. and Wunsch II, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678.

[Shi and Horvath, 2005] Shi, T. and Horvath, S. (2005). Unsupervised learning with random forest predictors. Technical Report, University of California, Los Angeles.

[Shi and Horvath, 2006] Shi, T. and Horvath, S. (2006). Unsupervised learning with random forest predictors. *Journal of Computational and Graphical Statistics*, 15(1):118–138.