International Journal of Data Science and Analytics (2020) 10:233–248
https://doi.org/10.1007/s41060-020-00216-2

REGULAR PAPER

Clustering of mixed-type data considering concept hierarchies: problem specification and algorithm

Sahar Behzadi[1] · Nikola S. Müller[2] · Claudia Plant[1,4] · Christian Böhm[3]

Received: 8 November 2019 / Accepted: 27 March 2020 / Published online: 25 April 2020

Abstract
Most clustering algorithms have been designed only for pure numerical or pure categorical data sets, while nowadays many applications generate mixed data. It raises the question how to integrate various types of attributes so that one could efficiently group objects without loss of information. It is already well understood that a simple conversion of categorical attributes into a numerical domain is not sufficient since relationships between values such as a certain order are artificially introduced. Leveraging the natural conceptual hierarchy among categorical information, concept trees summarize the categorical attributes. In this paper, we introduce the algorithm *ClicoT* (**CL**ustering mixed-type data **I**ncluding **CO**ncept **T**rees) as reported by Behzadi et al. (Advances in Knowledge Discovery and Data Mining, Springer, Cham, 2019) which is based on the minimum description length principle. Profiting of the conceptual hierarchies, ClicoT integrates categorical and numerical attributes by means of a MDL-based objective function. The result of ClicoT is well interpretable since concept trees provide insights into categorical data. Extensive experiments on synthetic and real data sets illustrate that ClicoT is noise-robust and yields well-interpretable results in a short runtime. Moreover, we investigate the impact of concept hierarchies as well as various data characteristics in this paper.

Keywords Mixed-type data · Information-theoretic clustering

1 Declarations

– *Availability of data and material* We used *MPG*, *Automobile* and *Adult* data sets from the UCI Public Data Repository [7] as well as *Airport* data set from the public project *Open Flights* (http://openflights.org/data.html).

– *Code availability* Our algorithm is implemented in Java and the source code as well as the data sets are publicly available here: https://tinyurl.com/ucp8289.

2 Introduction

Clustering mixed data is a non-trivial task and typically is not achieved by well-known clustering algorithms designed for a specific type. It is already well understood that converting one type to another one is not sufficient since it might lead to information loss. Moreover, relations among values (e.g., a certain order) are artificially introduced. In order to elaborate the issue, we generate a synthetic mixed-type data and investigate the impact of converting a categorical data type to a numerical one while applying well-known clustering algorithms.

Let Fig. 1 show a synthetically generated mixed-type data consisting of three different clusters illustrated by different shapes (rectangle, circle, cross), i.e., shapes are cluster IDs or ground truth. Thus, there are two Gaussian-shaped clusters where one of them (points with the shape rectangle) includes

✉ Sahar Behzadi
sahar.behzadi@univie.ac.at

Nikola S. Müller
nikola.mueller@helmholtz-muenchen.de

Claudia Plant
claudia.plant@univie.ac.at

Christian Böhm
boehm@ifi.lmu.de

1 Faculty of Computer Science, Data Mining, University of Vienna, Vienna, Austria

2 Helmholtz Research Center, Munich, Germany

3 University of Munich, Munich, Germany

4 ds:Univie, Vienna, Austria

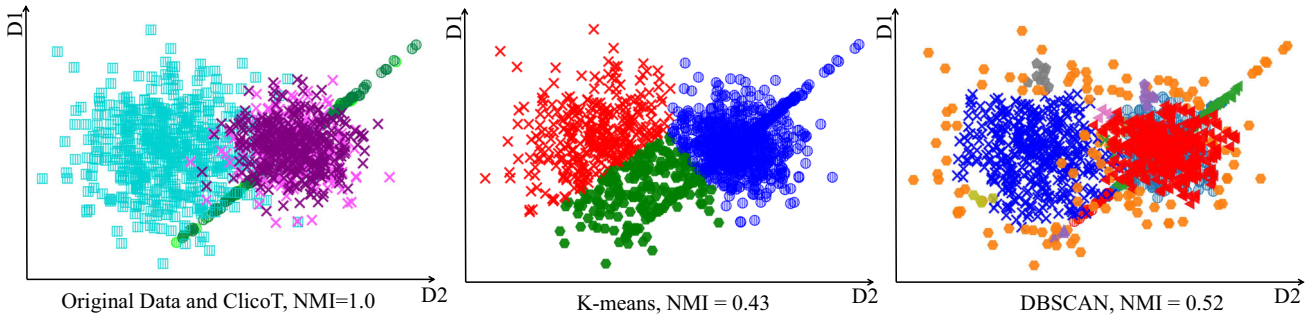


Fig. 1 Clustering results after converting categorical attribute *Color* to numerical (color figure online)

only data points having cyan as their color and the other cluster (points with the shape cross) includes data points having purple and rose as their color. The last cluster (points with the shape circle) is a line-shaped cluster consisting of dark green and light green data points.

The data set comprises two numerical attributes concerning the position of data objects and a categorical attribute representing the color of data points (rose, purple, light green, dark green and cyan). Therefore, a data object in our synthetic data looks like, for example (1, 2, purple). Numerical attributes are generated following various random Gaussian distributions. We simply converted the color to a numerical attribute by mapping numbers to various colors. Considering the *normalized mutual information* (NMI) [17] as an evaluation measure, Fig. 1 depicts the inefficiency of applying *K*-means and DBSCAN, two popular clustering algorithms, on the converted data. Therefore, integrating categorical and numerical attributes without any conversion is required since it preserves the original format of any attribute.

Utilizing the *minimum description length* (MDL) principle, we can regard the clustering task as a data compression problem such that the best clustering is linked to the strongest data set compression. MDL allows integrative clustering by relating the concepts of likelihood and data compression while for any attribute a representative model is required. Although for solely numerical data sets a *probability distribution function* (PDF) represents an approximation of data, finding an appropriate approximation for categorical attributes is not straightforward. Considering the natural hierarchy among categorical values, *concept hierarchies* are introduced to summarize the categorical information. Back to the running example, assuming pink as a higher-level hierarchy for the objects in the cluster consisting of rose and purple, points with the shape $\times$ more accurately represent the characteristics of the cluster.

Beyond the clustering approaches, detecting the most relevant attributes during this process improves the quality of clustering. However, considering a data set with an unknown distribution where only few subgroups in the data space are actually relevant to characterize a cluster, it is not trivial to recognize the cluster-specific attributes. Thus, we introduce an information-theoretic greedy approach to specify the most relevant attributes. As a result, the novel parameter-free **CL**ustering algorithm for mixed-type data **I**ncluding **CO**ncept **T**rees, shortly *ClicoT*, provides a natural interpretation. The approach consists of several contributions:

- *Integration* ClicoT integrates two types of information considering data compression as an optimization goal. ClicoT flexibly learns the relative importance of the two different sources of information for clustering without requiring the user to specify input parameters which are usually difficult to estimate.
- *Interpretation* In contrast to most clustering algorithms, ClicoT not only provides information about *which* objects are assigned to which clusters, but also gives an answer to the central question *why* objects are clustered together. As a result of ClicoT, each cluster is characterized by a signature of cluster-specific relevant attributes providing appropriate interpretations.
- *Robustness* The compression-based objective function ensures that only the truly relevant attributes are marked as cluster-specific attributes. Thereby, we avoid overfitting, enhance the interpretability and guarantee the validity of the result.
- *Usability* ClicoT is convenient to be used in practice since the algorithm scales well to large data sets. Additionally, the compression-based approach avoids difficult estimation of input parameters, e.g., the number or the size of clusters.

Moreover, in this paper we elaborate the concept hierarchies and investigate the impact of them on the performance of ClicoT. We also address whether or not various characteristics of data sets, e.g., proportion of categorical and numerical attributes, have any influence on the effectiveness of ClicoT via extensive experiments.

3 Clustering mixed data types

To design a mixed-type clustering algorithm, we need to address three fundamental questions: How to model numerical attributes to properly characterize a cluster? How to model categorical attributes? And finally how to efficiently integrate heterogeneous attributes when the most relevant attributes are specified? In principle, a PDF summarizes values by approximating meaningful parameters. However, the idea of using a background PDF for categorical attributes is not intuitive at first; therefore, we employ concept hierarchies.

3.1 Concept hierarchy

In this paper, a concept could be color of an object (e.g., light green or cyan), marital status (e.g., married) or the continent where a country is located (e.g., Asia). More precisely, a concept is a categorical value showing some characteristics of every data object. As mentioned, concept hierarchies allow us to express conceptual interchangeable values by selecting an inner node of a concept hierarchy to describe a cluster. Concept hierarchies not only capture more relevant categories for each cluster but also help to interpret the clustering result appropriately. Let $\mathcal{DB}$ denote a database consisting of n objects. An object o comprises m categorical attributes $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$ and d numerical attributes $\mathcal{X} = \{x_1, x_2, \ldots, x_d\}$. For a categorical attribute A_i, we denote different categorical values by ${A_i}^{(j)}$. An *Element* represents a categorical value or a numerical attribute and we denote the number of all *Element*s by E. Considering the natural hierarchy between different categories, for each categorical attribute A_i a concept hierarchy is already available as follows:

Definition 1 *Concept Hierarchy* Let $T_{A_i} = (N, \mathcal{E})$ be a tree with root A_i denoting the concept hierarchy corresponding to the categorical attribute A_i with the following properties:

1. T_{A_i} consists of a set of nodes $N = \{n_1, n_2, \ldots, n_s\}$ where any node is corresponding to a categorical concept. $\mathcal{E}$ is a set of directed edges $\mathcal{E} = \{e_1, e_2, \ldots, e_{(s-1)}\}$, where n_j is a parent of n_z if there is an edge $e_l \in \mathcal{E}$ so that $e_l = (n_j, n_z)$.
2. The level $l(n_j)$ of a node n_j is the height of the descendant sub-tree. If n_j is a leaf, then $l(n_j) = 0$. In a concept, tree leaf nodes are categorical values existing in the data set. The root node is the attribute A_i which has the highest level, also called the height of the concept hierarchy.
3. Each node $n_j \in N$ is associated with a probability $p(n_j)$ which is the frequency of the corresponding category in a data set.

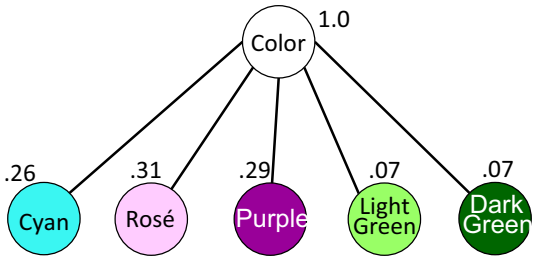


Fig. 2 A flat concept tree for the categorical attribute color (color figure online)

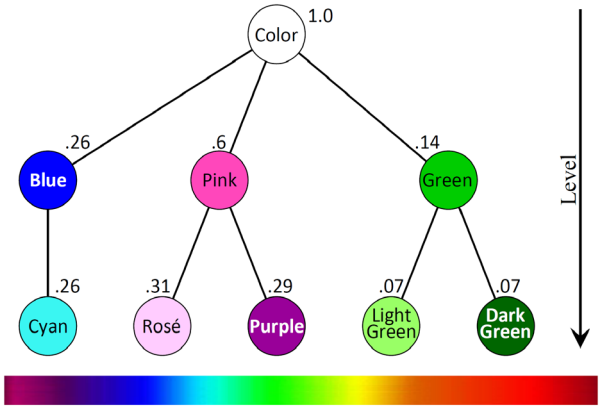


Fig. 3 Concept tree corresponding to the running example w.r.t. the natural hierarchy among various colors (color figure online)

4. Each node n_j represents a sub-category of its parent; therefore, all probabilities of the children sum up to the probability of the parent node.

To elaborate, let us consider the synthetic example illustrated in Sect. 2 (see Fig. 1). In this example, the only categorical attribute is color, while two other attributes are numeric. Therefore, $\mathcal{A} = \{A_1\}$ and $\mathcal{X} = \{x_1, x_2\}$ where A_1 is color and X_1, X_2 show X and Y-axis in a two-dimensional space. Every data point in this data set can have one of the following colors: cyan, rose, purple, light green and dark green. Thus, the set of categorical values w.r.t. A_1 is { cyan, rose, purple, light green and dark green}. As a default, we assume a flat concept tree to summarize the frequency of categorical values, especially, when there is no meaningful hierarchy among different categories. A flat hierarchy consists of a one level tree including all the categories in the leaf level without any hierarchy. Figure 2 depicts a default flat concept tree corresponding to the running example. However, usually, for each categorical attribute a concept hierarchy is available due to the natural hierarchy among different categories. For instance, considering the natural scalable range of colors, one can categorize different colors as illustrated in Fig. 3. Here, the height is 2 showing another concept level (level 1) which categorizes the color of data points, e.g., green categorizes

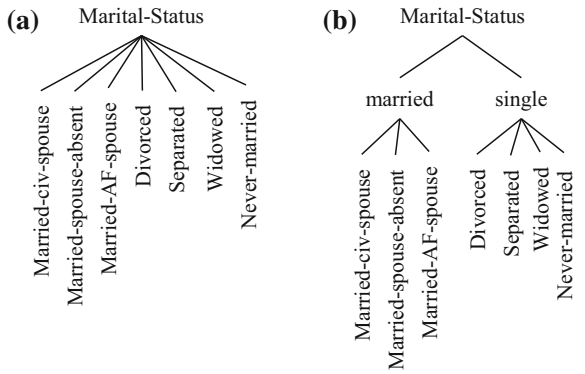


Fig. 4 Concept tree for categorical attribute *marital status* w.r.t. Adult data set

dark green and light green in a same category based on the natural scalable range of colors. The node labels show background probabilities $p(n_j)$ (i.e., frequency) for each node. This initialization of the background distribution is once performed before assigning objects to clusters.

Categorical attributes are more often observed in real applications, e.g., population surveys. As an example, we focus on *Adult* data, a real-world data set from the UCI repository [7]. Adult data set without missing values, extracted from the census bureau database, consists of 48,842 instances of 11 attributes. The class attribute Salary indicates whether the salary is over 50K or lower. Categorical attributes consist of different information about people in this survey, e.g., work-class, education, occupation and marital status. Focusing on the categorical feature marital status, every person belongs to a unique category including divorced, never married, married-spouse-absent, etc. The leaf level shows various marital status one could have in both concept trees illustrated in Fig. 4. The left concept tree without any hierarchy (Fig. 4a) shows the default flat hierarchy can be considered in the beginning. However, we can categorize various status based on whether or not a person is married. In this case, three different categories, i.e., married-civ-spouse, married-spouse-absent and married-AF-spouse, fall in the same category, married. All other categorical values, i.e., divorced, separated, widowed and never-married, cannot be located in the same category since people having these status are single. Therefore, we consider another category, single, which seems more plausible for those status. Thus, Fig. 4b shows one of the possible concept hierarchies one can assume w.r.t. marital status for Adult data. We investigate this data set in more detail in Sect. 6.

ClicoT profits the concept hierarchy to provide more interpretable results. But also non-hierarchical categorical attributes can be regarded as a simple flat concept hierarchies with height one. We claim our algorithm performs appropriately in comparison with other algorithms for this case as well.

3.2 Cluster-specific elements

Besides an efficient clustering approach, finding relevant attributes to capture the best fitting model is important. Usually, the clustering result is disturbed by irrelevant attributes. To make the model for each cluster more precise, we distinguish between relevant and irrelevant attributes. Each cluster c is associated with a subset of the numerical and categorical relevant elements denoted by *cluster-specific elements*. Categorical cluster-specific elements are represented by a specific concept hierarchy which diverges from the background hierarchy (i.e., the concept hierarchy of the entire database).

Definition 2 *Cluster* A cluster c is described by:

1. A set of objects $\mathcal{O}_c \subset \mathcal{DB}$.
2. A cluster-specific subspace $I = \mathcal{X}_c \cup \mathcal{A}_c$, where $\mathcal{X}_c \subseteq \mathcal{X}$ and $\mathcal{A}_c \subseteq \mathcal{A}$.
3. For any categorical attribute $A_i \in \mathcal{A}_c$, the corresponding cluster-specific concept hierarchy is a tree ${T^c}_{A_i} = (N_c, \mathcal{E}_c)$ with nodes and edges as specified in Definition 1. $N_c \subset N$ indicates the cluster-specific nodes. For computing the probabilities associated with the cluster-specific nodes instead of all n objects, only the objects $\mathcal{O}_c$ in cluster c are applied, i.e., $p(n_j) = \frac{|n_j|}{|\mathcal{O}_c|}$.

The idea of cluster-specific nodes allows to specify an inner node as a representative for each cluster. ClicoT aims at finding a partition of $\mathcal{DB}$ into clusters, and simultaneously at discovering the cluster-specific subspace for each cluster.

3.3 Integrative objective function

Given the appropriate model corresponding to any attribute, MDL allows a unified view on mixed data. The better the model matches major characteristics of the data, the better the result is. Following the MDL principle [16], we encode not only the data but also the model itself and minimize the overall description length. Simultaneously, we avoid overfitting since the MDL principle tends to a natural trade-off between model complexity and goodness-of-fit.

Definition 3 *Objective Function* Considering cluster c the description length (DL) corresponding to this cluster is defined as:

$$\mathrm{DL}(c) = \mathrm{DL}_n(\mathcal{X}) + \mathrm{DL}_c(\mathcal{A}) + \mathrm{DL}(model(c)) \quad (1)$$

The first two terms, i.e., DL_n and DL_c, represent coding costs concerning numerical and categorical attributes, respectively. The last term ($\mathrm{DL}(model)$) denotes the model encoding cost. Essentially, numerical and categorical attributes contribute simultaneously and in the same way. We incorporate the

required coding cost for both types, numerical and categorical, without any data type conversion. Thus, instead of data type conversion we integrate all attributes avoiding information loss. Our proposed objective function minimizes the overall description length of the database which is defined as:

$$\mathrm{DL}(\mathcal{DB}) = \sum_{c \in \mathcal{C}} \mathrm{DL}(c) \tag{2}$$

Coding Numerical Attributes Considering Huffman coding scheme, the description length of a numerical value o_i is defined by $-\log_2 \mathrm{PDF}(o_i)$. We assume the same PDF to encode the objects in various clusters and clusters compete for an object while the description length is computed by means of the same PDF for every cluster. Therefore, any PDF would be applicable and using a specific model is not a restriction [4]. For simplicity, we select Gaussian PDF, $\mathcal{N}(\mu, \sigma)$. Moreover, we distinguish between the cluster-specific attributes in any cluster c, denoted by $\mathcal{X}_c$, and the remaining attributes $\mathcal{X} \setminus \mathcal{X}_c$ (Definition 2). Let μ_i and σ_i denote the mean and variance corresponding to the numerical attribute x_i in cluster c. If x_i is a cluster-specific element ($x_i \in \mathcal{X}_c$), we consider only cluster points to compute the parameters otherwise ($x_j \in \mathcal{X} \setminus \mathcal{X}_c$) the overall data points will be considered. Thus, the coding cost for numerical attributes in cluster c is provided by:

$$DL_n(\mathcal{X}) = \sum_{x_i \in \mathcal{X}} \sum_{o_i \in \mathcal{O}_c} -\log_2 \Big(\mathcal{N}(\mu_i, \sigma_i)\Big) \tag{3}$$

Coding Categorical Attributes Analogously, we employ Huffman coding scheme for categorical attributes. The associated probability to a category is its frequency w.r.t. either the specific or the background hierarchy (Definition 1). Similar to numerical attributes, we assume $\mathcal{A}_c$ as the set of cluster-specific categorical attributes and $\mathcal{A} \setminus \mathcal{A}_c$ for the rest. Let o_j denote a categorical object value corresponding to the attribute A_j. We define $f(A_j, o_j)$ as a function which maps o_j to a node in either a specific or a background hierarchy depending on A_j. In summary, $f(.)$ is defined as:

$$f(A_j, o_j) = \begin{cases} n_j \in T^c{}_{A_j} & A_j \in \mathcal{A}_c \\ n_j \in T_{A_j} & A_j \in \mathcal{A} \setminus \mathcal{A}_c \end{cases}$$

Thus, the categorical coding cost for a cluster c is given by:

$$\mathrm{DL}_c(\mathcal{A}) = \sum_{A_j \in \mathcal{A}} \sum_{o_j \in \mathcal{O}_c} -\log_2 \big(p(f(A_j, o_j))\big) \tag{4}$$

Model Complexity Without taking the model complexity into account, the best result will be a clustering consisting of singleton clusters. This result is completely useless in terms of the interpretation. Focusing on cluster c, the model complexity is defined as:

$$\begin{aligned} \mathrm{DL}(model(c)) = {} & idCosts(c) + SpecificIdCosts(c) \\ & + paramCosts(c) \end{aligned} \tag{5}$$

The idCosts are required to specify which cluster is assigned to a object while balancing the size of clusters. Employing the Huffman coding scheme, idCosts are defined by $|\mathcal{O}_c| \cdot log_2 \frac{n}{|\mathcal{O}_c|}$ where $|\mathcal{O}_c|$ denotes the number of objects assigned to cluster c. Moreover, in order to avoid information loss we need to specify whether an attribute is a cluster-specific attribute or not. That is, given the number of specific elements s in cluster c, the coding costs corresponding to these elements, *SpecificIdCosts*, is defined as:

$$SpecificIdCosts(c) = s \cdot \log_2 \frac{E}{s} + (E - s) \cdot \log_2 \frac{E}{(E - s)} \tag{6}$$

Following fundamental results from information theory [16], the costs for encoding the model parameters are reliably estimated by:

$$paramCosts(c) = \frac{numParams(c)}{2} \cdot \log_2 |\mathcal{O}_c| \tag{7}$$

For any numerical cluster-specific attribute, we need to encode its mean and variance while for a categorical one the probability deviations to the default concept hierarchy need to be encoded, i.e., $numParams(c) = |\mathcal{X}| \cdot 2 + \sum_{A_i \in \mathcal{A}} |N_c|$. Moreover, we need to encode the probabilities associated with the default concept hierarchy, as well as the default (global) means and variances for all numerical attributes. However, these costs are summarized to a constant term which does not influence our subspace selection and clustering technique.

4 Algorithm

Together with the main building blocks of ClicoT, two other steps are required to achieve an appropriate parameter free clustering: (1) recognizing the cluster-specific elements and (2) probability adjustments.

4.1 How to specify cluster-specific elements?

The optimization process in the objective function tends to mark an element with the most cost saving as a cluster-specific. Let the *specific coding cost* denote the cost where an element is marked as specific and the *non-specific coding cost*

Algorithm 1 Cluster-specific elements

Deviation ($Element \quad e_i$)
scc := specific coding cost
$nscc$:= non-specific coding cost
dev := deviation in terms of coding cost
if e_i is numerical **then**
  // case 1: e_i is specific
  // find $\mathcal{N}(\mu_i, \sigma_i)$ w.r.t. $\mathcal{O}_c$ and compute $DL_n(.)$
  $s = s + 1$
  $scc = \sum_{c \in \mathcal{C}} DL(c)$

  // case 2: e_i is not specific
  // find $\mathcal{N}(\mu_i, \sigma_i)$ w.r.t. $\mathcal{DB}$ and compute $DL_n(.)$
  $nscc = \sum_{c \in \mathcal{C}} DL(c)$

else if e_i is categorical **then**
  // case 1: e_i is specific
  A_j := categorical attribute w.r.t. e_i
  T_{A_j} := concept tree w.r.t. A_j
  // adjust T_{A_j} and get $T^c_{A_j}$
  for all $Vertex \quad V$ in T_{A_j} **do**
    ProcessHierarchy(V)
  end for
  // find $DL_c(.)$ w.r.t. specific concept tree $T^c_{A_j}$
  $P(o) = P(n)$ where $n \in T^c_{A_j}$
  $s = s + 1$
  $scc = \sum_{c \in \mathcal{C}} DL(c)$

  // case 2: e_i is not specific
  // find $DL_c(.)$ w.r.t. background concept tree T_{A_j}
  $P(o) = P(n)$ where $n \in T_{A_j}$
  $nscc = \sum_{c \in \mathcal{C}} DL(c)$
end if

// find the deviation
$dev = |nscc - scc|$
return dev

Algorithm 2 Concept tree updates

ProcessHierarchy ($Vertex \quad V$)
ssp := sum of specific probabilities
sup := sum of unspecific probabilities
if V is a leaf **then**
  if V is specific **then**
    return $(V.probability, 0)$
  end if
  return $(0, V.backgroundProbability)$
end if
// now V is not a leaf
$(ssp, sup) := (0, 0)$
for all C in $children(V)$ **do**
  $(s, u) := processHierarchy(C)$
  $(ssp, sup) := (ssp + s, sup + u)$
end for
if V is specific or root **then**
  $factor := (V.probability - ssp)/sup$
  for all C in $children(V)$ **do**
    $propagateDownFactor(C, factor)$
  end for
  return $(V.probability, 0)$
end if
return (ssp, sup)

indicates the cost otherwise. Consulting the idea that cluster-specific elements have the most deviation of specific and non-specific cost and therefore saves more coding costs, we introduce a greedy method to recognize them. Algorithm 1 summarizes how to find the coding cost deviation w.r.t. every element e_i. We sort the elements according to their deviations and specify the first element as a cluster-specific element. We continue marking elements until marking more elements does not pay off in terms of the coding cost. Note that different nodes of a concept hierarchy have the same opportunity to be specific. Additionally marking a categorical element influences the specific concept hierarchy; therefore, we have to consider the adapted probabilities (next section).

4.2 Probability adjustment

To adjust the probabilities for a numerical cluster-specific attribute, we can safely use mean and variance corresponding to the cluster. In contrast, learning the cluster-specific concept hierarchy is more challenging since we need to maintain the integrity of a hierarchy. According to Definition 1, we assure that node probabilities of siblings in each level sum up to the probability of the parent node. Moreover, node probabilities should sum up to one for each level.

Algorithm 2 summarizes the adjustment process where *ProcessHierarchy*() is a recursive procedure to update the concept tree assuming marked cluster-specific elements. It, firstly, determines all probabilities in a concept hierarchy starting from the following configuration: Initially, all nodes are assigned to the background probability of the overall data set ($V.backgroundProbability$). An arbitrary number of (internal and/or leaf) nodes are marked as *cluster-specific* and assigned to different probabilities, taken from the currently considered cluster ($V.probability$). The recursive procedure is always started at the root node. When descending the concept hierarchy recursively, for each node we keep track of two sums, that of the specific probabilities inside the complete subtree (ssp) and that of the unspecific ones (sup). When returning from a recursion, we pass exactly these two variables to the caller, enabling him to determine how much the remaining probabilities must be adjusted. Whenever we return from the recursion and reach a cluster-specific node, we determine an adjustment factor according to the formula

$$factor = \frac{V.probability - ssp}{sup} \tag{8}$$

which is the factor correcting the deviation between all cluster-unspecific nodes in the sub-tree from the probability which we have in the current specific node. This factor is propagated down the concept hierarchy using the procedure PropagateDownFactor() in Algorithm 3 which is again

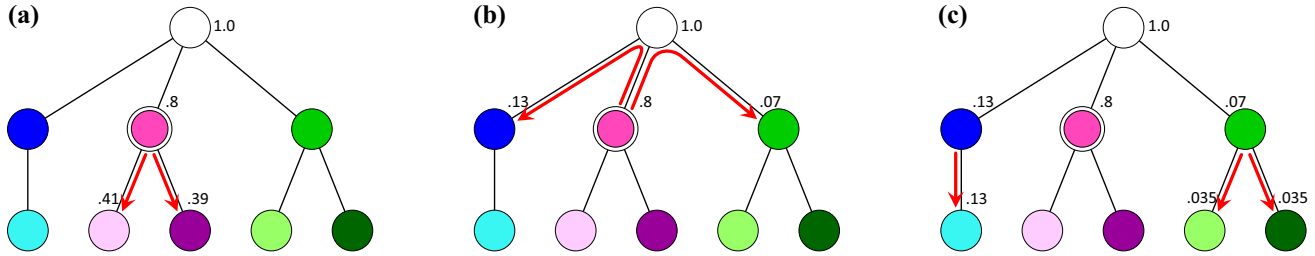


Fig. 5 Update concept hierarchies considering *pink* as a cluster-specific node (color figure online)

recursive and descends the sub-tree only in the non-cluster-specific nodes, because only for those we can adapt the probabilities. If ConceptTreeUpdate() returns to a cluster-unspecific node, it only sums up ssp and sup and delivers this information to its caller without directly down-propagating anything. Overall, the recursive method ProcessHierarchy() visits every node of the concept hierarchy once (and only once). During this whole recursive procedure, it is also guaranteed that PropagateDownFactor() also visits every node at most once. Thus, the method is linear in the number of nodes.

Algorithm 3 Down-propagation of the adjustment factor

```
1: PropagateDownFactor (Vertex V, double factor)
2: if V is unspecific then
3:     V.probability := V.probability · factor
4:     if V is not leaf then
5:         for all C ∈ V.children do
6:             PropagateDownFactor(C, factor)
7:         end for
8:     end if
9: end if
```

To clarify, let Fig. 5 show the procedure on the concept hierarchy corresponding to the running example (Fig. 1) where labels denote the frequencies. Moreover, let pink be a cluster-specific node for the cluster with the shape ×. The adjustment starts with the root node and processes its children. Then, it continues computing the relative probabilities for the specific concept hierarchy rather by background probability fraction (Fig. 5a). 80% relative probability should be distributed between two children, rose and purple, based on the computed propagation factor. During the next step, the remaining 20% probability is assigned level-wise to blue and green to assure that probabilities in each level sum up to 1 (Fig. 5b). Again each parent propagates down its probability (Fig. 5c). The result is a concept hierarchy best fitting to the objects when the background distributions are preserved.

4.3 ClicoT algorithm

ClicoT is a top-down parameter-free clustering algorithm. That is, we start from a cluster consisting of all objects and

Algorithm 4 ClicoT

```
1: input DB
2: learn background distributions of each attribute
3: C' = {C_0} with C'_0 = O_i ∈ DB
4: repeat
5:     // try to split until convergence
6:     C = C'
7:     cost = DL(DB|C) // current cost
8:     C' = {C'_1 ... C'_{k-1}} split worst C_i ∈ C to {C'_i, C'_k}
9:     while clustering C' changes do
10:        C'_i = {O_j : min_i DL(O_j|C'_i)} // assign objects
11:        Select cluster-specific elements by a greedy method for each
           cluster and compute costs
12:        Update each attribute of C'_i
13:    end while
14:    cost' = DL(DB|C') // split cost
15: until cost > cost'
16: k = |C|
17: return C, k
```

iteratively split down the most expensive cluster c in terms of the coding cost to two new clusters $\{c'_a, c'_b\}$. Then, we apply a k-means-like strategy and assign every point to closest cluster which is nothing else than the cluster with the lowest increase in the coding cost. Employing the greedy algorithm, we determine the cluster-specific elements and finally we compute the compression cost for clustering results in two cases, before and after splitting (Definition 3). If the compression cost after splitting, i.e., $\mathcal{C}'$ with $|\mathcal{C}'| = k + 1$, is cheaper than the cost of already accepted clustering $\mathcal{C}$ with $|\mathcal{C}| = k$, then we continue splitting the clusters. Otherwise the termination condition is reached and the algorithm will be stopped.

5 Related work

Driven by the need of real applications, the topic of clustering mixed-type data represented by numerical and categorical attributes has attracted attentions, e.g., CFIKP [19], CAVE [10], CEBMDC [8]. In between, most of the algorithms are designed based on the algorithmic paradigm of k-means, e.g., k-Prototypes [11]. Often in this category not only the number of clusters k but also the weighting between numerical

and categorical attributes in clustering has to be specified by the user. Among them, K-means-mixed (KMM) [1] avoids weighting parameters by an optimization scheme learning the relative importance of the single attributes during runtime, although it needs the number of clusters k as input parameter. KMM employs data conversion and discretize numerical attributes into categorical ones and then calculate the interactions in terms of categorical ways. Almost similarly, SpectralCAT [6] and CoupledMC [18] both conduct k-means clustering on continuous features and use a validity index to choose clustering label as new continuous features. These methods calculate the couplings based on discretized numerical data which may lead to information loss due to failure in capturing the distribution of the continuous data.

Following a mixture of Gaussian distributions, model-based clustering algorithms have been also proposed for mixed-type data. In between, ClustMD [13] is developed using a latent variable model and employing an expectation maximization (EM) algorithm to estimate the mixture model. Yet, this algorithm has a certain Gaussian assumption which does not have to be necessarily fulfilled. On the other hand, clustering algorithms for mixed-data often do not properly model dependencies and are limited to modeling meta–Gaussian distributions. Copulas, that provide a modular parameterization of joint distributions, can model a variety of dependencies, but their use with discrete data remains limited due to challenges in parameter inference. Authors in [15] use Gaussian mixture copulas, to model complex dependencies beyond those captured by meta–Gaussian distributions, for clustering. However, this approach may not only result in information loss but also fail to capture the discriminative information between objects.

Some of the approaches utilize the unique characteristics of any data type to avoid the drawbacks of converting a data type to another one. Profiting of the concept hierarchy, these algorithms introduce an integrative distance measure applicable for both numerical and categorical attributes. The algorithm DH [9] proposes a hierarchical clustering algorithm using a distance hierarchy which facilitates expressing the similarity between categorical and numerical values. As another method, MDBSCAN [2] employs a hierarchical distance measure to introduce a general integrative framework applicable for the algorithms which require a distance measure, e.g., DBSCAN. On the other hand, information-theoretic approaches have been proposed to avoid the difficulty of estimating input parameters. These algorithms regard the clustering as a data compression problem by hiring the minimum description length (MDL). The cluster model of these algorithms comprises joint coding schemes supporting numerical and categorical data. The MDL principle allows balancing model complexity and goodness-of-fit. INCONCO [14] and Integrate [5] are two representative for mixed-type clustering algorithms in this family. While Integrate has been designed for general integrative clustering, INCONCO also supports detecting mixed-type attribute dependency patterns.

Recently, deep neural networks are widely used for clustering. Among them, authors in [12] propose an auto-instructive representation learning scheme to enable margin-enhanced distance metric learning for a discrimination-enhanced representation. Finally, they feed the learned representation into both partition-based (k-means) and density-based (DBSCAN) clustering methods.

6 Evaluation

In this section, we assess the performance of ClicoT compared to other clustering algorithms in terms of NMI which is a common evaluation measure for clustering results. NMI numerically evaluates pairwise mutual information between ground truth and resulted clusters scaling between zero and one. We conducted several experiments evaluating ClicoT in comparison with KMM [1], INCONCO [14], DH [9], ClustMD [13], Integrate [5] and MDBSCAN [2]. In order to be fair in any experiment, we input the corresponding concept hierarchy to the algorithms which are not designed for dealing with it. That is, we encode the concept hierarchy as an extra attribute so that categorical values belonging to the same category have the same value in this extra attribute. Our algorithm is implemented in Java, and the source code as well as the data sets is publicly available[1].

6.1 Mixed-type clustering of synthetic data

In order to cover all aspects of ClicoT, we first consider a synthetic data set. Then, we continue experiments by comparing all algorithms in terms of the noise-robustness. Finally, we will discuss the runtime efficiency.

Clustering Results In this experiment, we evaluate the performance of all the algorithms on the running example (Fig. 1) while all parametric algorithms are set up with the right number of clusters. The data have two numerical attributes concerning the position of any data point and a categorical attribute showing the color of the points. Figure 6 shows the result of applying the algorithms where different clusters are illustrated by different colors. As it is explicitly shown in this figure, ClicoT, with NMI 1, appropriately finds the initially sampled three clusters where green, pink and blue are cluster-specific elements. Setting the correct number of cluster and trying various Gaussian mixture models, ClustMD results in the next accurate clustering. Although MDBSCAN utilizes the distance hierarchy, it is not able to capture the pink and green clusters. KMM cannot distinguish among various colors. Since two clusters pink and green

[1] https://tinyurl.com/ucp8289.

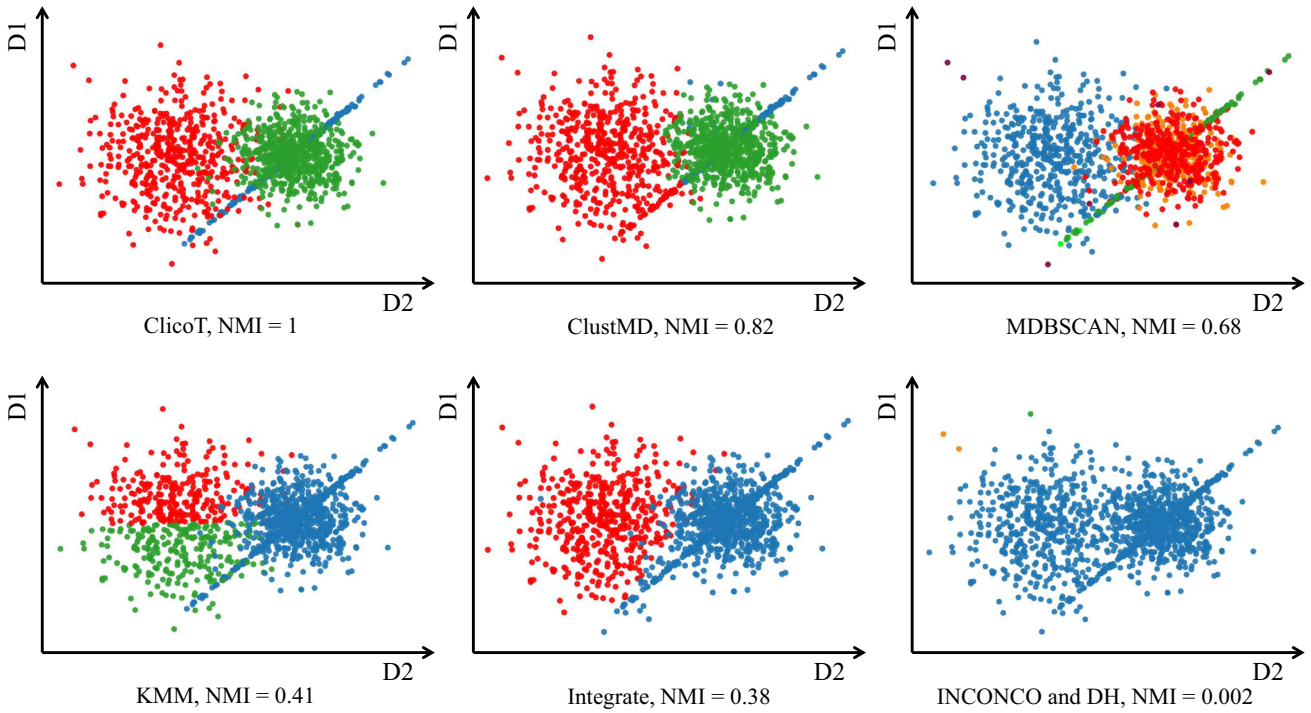
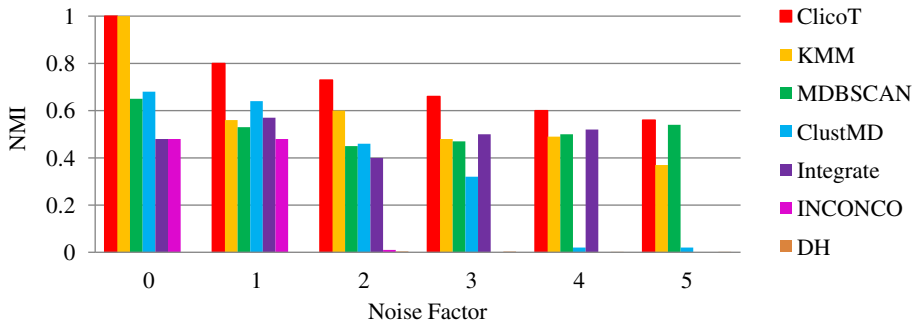


Fig. 6 Clustering results on the running example

Fig. 7 Comparing noise-robustness of ClicoT to other algorithms



are heavily overlapped, Integrate cannot distinguish among them. DH and INCONCO result on this data set inefficiently finding almost only one cluster.

Noise-robustness In this section, we benchmark noise-robustness of ClicoT w.r.t the other algorithms in terms of NMI by increasing the noise factor. To address this issue, we generate a data set with the same structure as the running example when adding another category, brown, to the categorical attribute color as noise. Regarding numerical attributes, we increase the variance of any cluster. We start from 5% noise (noise factor = 1) and iteratively increase the noise factor ranging to 5. Figure 7 clearly illustrates noise-robustness of ClicoT compared to others.

Flat Hierarchy In this section, we investigate the case when no appropriate hierarchy is considered. That is, we assume a flat concept tree with no hierarchy (e.g., Fig. 2) and run the following two experiments. Firstly, we focus on the running example introduced in Sect. 2 (see Fig. 1) and assume a flat hierarchy for the categorical attribute *Color* where no higher level concept categorizes the colors (Fig. 2).

As expected also observed from Fig. 8, ignoring a meaningful hierarchy for categorical attributes decreases the performance of ClicoT. However, ClicoT-flat (NMI = 0.60) is still comparable to MDBSCAN and more effective than KMM, Integrate, INCONCO and DH. In this data set, Cluster 3 (the line shape cluster illustrated by green circles in Fig. 1) highly overlaps two other clusters at some points. The data points in this cluster have the colors light green and dark green. As it is observed from the result of ClicoT-flat (Fig. 8), ignoring a meaningful hierarchy for the colors leads to an inefficiency in the sense that numerical attributes get cluster-specific and hence important while clustering. Therefore, parts of Cluster 3 which overlap with two other clusters (middle part and tail of Cluster 3) are wrongly grouped.

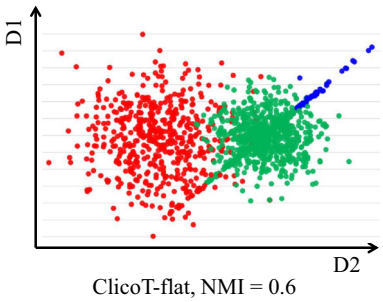


Fig. 8 Result of ClicoT applied on running example assuming a flat concept tree for colors (Fig. 2) (color figure online)

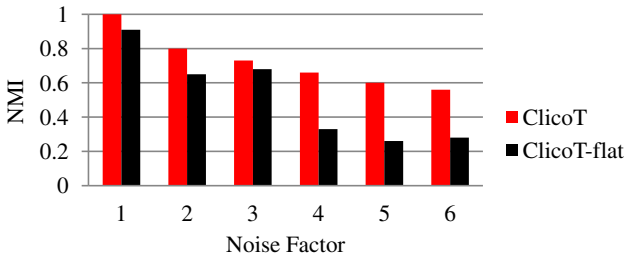


Fig. 9 Comparing ClicoT and ClicoT-flat assuming various noise factors

In the next investigation, we repeat the noise experiment applying ClicoT-flat. Here, the goal is to compare ClicoT and ClicoT-flat in various cases when the data set gets more noisy iteratively. As the plot in Fig. 9 depicts, ClicoT is always more effective in comparison with ClicoT-flat, although the performance of ClicoT-flat is still comparable in the beginning when the noise factor is smaller. It again approves the role of a meaningful hierarchy in order to increase the efficiency.

Scalability To evaluate the efficiency of ClicoT w.r.t the other algorithms, we generated a 10-dimensional data set (5 numerical and 5 categorical attributes) with three Gaussian clusters. Then, respectively, we increased the number of objects ranging from 2000 to 10,000. In the other case, we generated different data sets of various dimensionality ranging from 10 to 50 where the number of objects is fixed. Figure 10 depicts the performance of all algorithms in terms of the runtime complexity. Regarding the first experiment on the number of objects, ClicoT is slightly faster than others while increasing the dimensionality Integrate performs faster. However, the runtime of this algorithm highly depends on the number of clusters k initialized in the beginning (we set $k = 20$). That is, this algorithm tries a range of k and outputs the best results. Therefore, by increasing k the runtime is also increasing.

Proportion How would ClicoT behave when various proportions of categorical and numerical attributes are considered in the data sets? What happens when the majority of attributes are numerical and vice versa? In this experiment, we address the mentioned questions and generate various synthetic data sets each of which having a different proportion of categorical and numerical attributes. The x-axis in Fig. 11 shows the proportion factor, while for factor 1, for example, we generate 2 numerical and 2 categorical attributes. In Fig. 11, the yellow bins show the case when we increase the number of numerical attributes while the categorical attributes are set two, e.g., factor $3 = \frac{6 \quad numerical \quad attributes}{2 \quad categorical \quad attributes}$. For the categorical attributes, we assume a flat hierarchy with 3 various categories in every experiment. Analogously the green bins in Fig. 11 illustrate the results of applying ClicoT when the proportion factor is achieved by $proportion = \frac{\#categorical \quad attributes}{\#numerical \quad attributes}$.

As observed in Fig. 11 having various number of numerical or categorical attributes as well as different proportions does not influence the performance of our proposed algorithm. ClicoT is very well designed to deal with any kind of data structures since it always utilizes cluster-specific attributes and marks the most relevant attributes as specific.

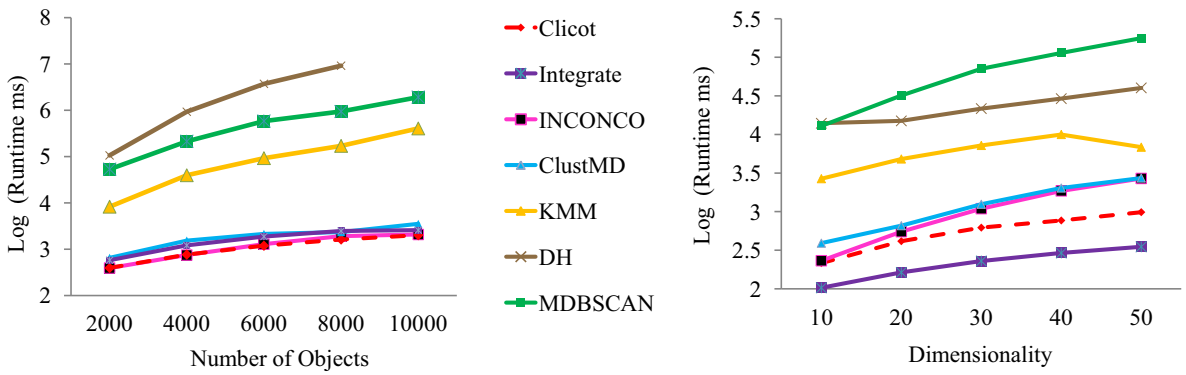


Fig. 10 Investigating the runtime efficiency of ClicoT in comparison with other algorithms. Two various cases are considered: **a** when the number of objects is increasing while the dimensionality is fixed, **b** when the number of objects is fixed and the dimensionality (number of categorical and numerical attributes) is increasing

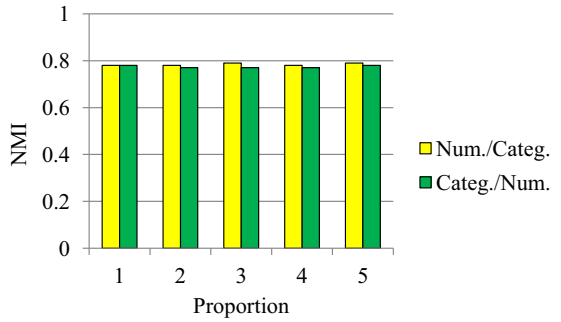


Fig. 11 Synthetic experiments to investigate the impact of various proportions of categorical and numerical attributes

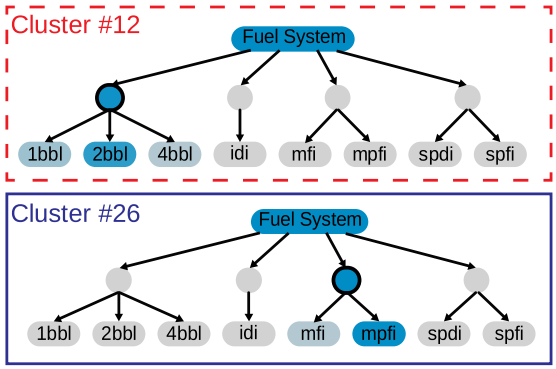


Fig. 12 Cluster-specific categories for Cluster 12 and Cluster 26 w.r.t. the categorical attribute *Fuel System*

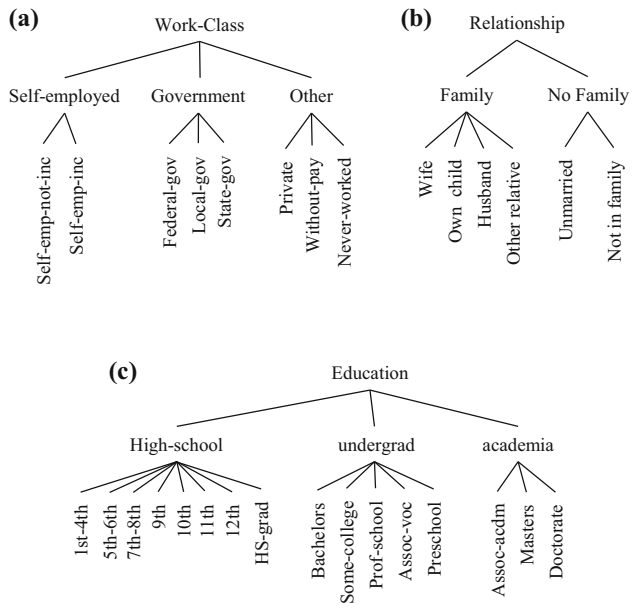


Fig. 13 Concept tree for 3 categorical attributes of Adult data set

Table 1 Cluster specific attributes for attribute *Relationship* based on the most deviation

	C_2	C_3
Family	− 0.24	0.359
Wife	0.025	− 0.047
Own child	0.111	− 0.154
Husband	**− 0.398**	**0.59**
Other relative	0.02	− 0.028
No family	0.24	− 0.359
Unmarried	0.074	− 0.105
Not in family	0.165	− 0.253

Bold numbers in the table show maximum deviations corresponding to each attribute

6.2 Experiments on real-world data

Finally, we evaluate clustering quality and interpretability of ClicoT on real-world data sets. We used *MPG*, *Automobile* and *Adult* data sets from the UCI Repository [7] as well as *Airport* data set from the public project *Open Flights*[2].

MPG MPG is a slightly modified version of the data set provided in the StatLib library. The data concern city-cycle fuel consumption in miles per gallon (MPG) in terms of 3 categorical and 5 numerical attributes consisting of different characteristics of 397 cars. We consider MPG ranging from 10 to 46.6 as the ground truth and divide the range to 7 intervals of the same length. Considering a concept hierarchy for the name of cars, we group all the cars so that we have three branches: European, American and Japanese cars. Moreover, we divide the range of model year attribute to three intervals: 70–74, 75–80 and after 80. We leave the third attribute as a flat concept hierarchy since there is no meaningful hierarchy between variation of cylinders. Comparing ClicoT (NMI = 0.4) to the other algorithms INCONCO (0.17), KMM (0.37), DH (0.14), MDBSCAN (0.02), ClustMD (0.33) and Integrate (0), ClicoT correctly finds 7 clusters each of which is compatible with one of the MPG groups. Cluster 2, for instance, is compatible with the first group of MPGs since the frequency of the first group in this cluster is 0.9. In this cluster, American cars with the frequency of 1.0 and cars with 8 cylinders with the frequency of 1 and model year in first group (70–74) with the frequency of 0.88 are selected as cluster-specific elements.

Automobile This data set provides 205 instances with 26 categorical and numerical attributes. The first attribute defining the risk factor of an automobile has been used as class label. Altogether there are 6 different classes. Due to many missing values, we used only 17 attributes. Comparing the best NMI captured by every algorithm, ClicoT

[2] http://openflights.org/data.html.

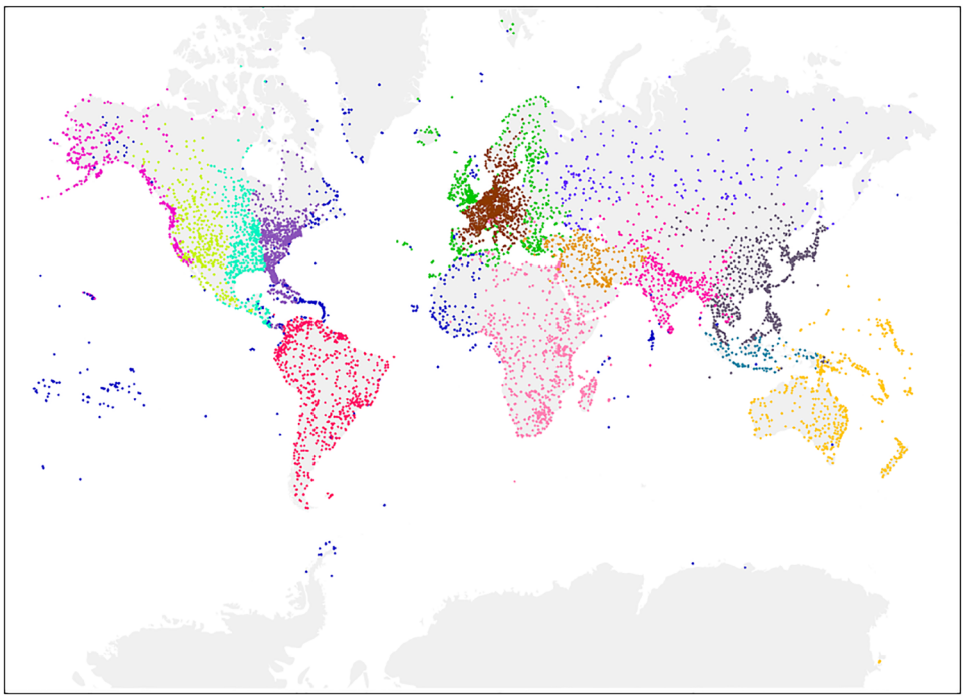


Fig. 14 Result of ClicoT on Open Flights data set

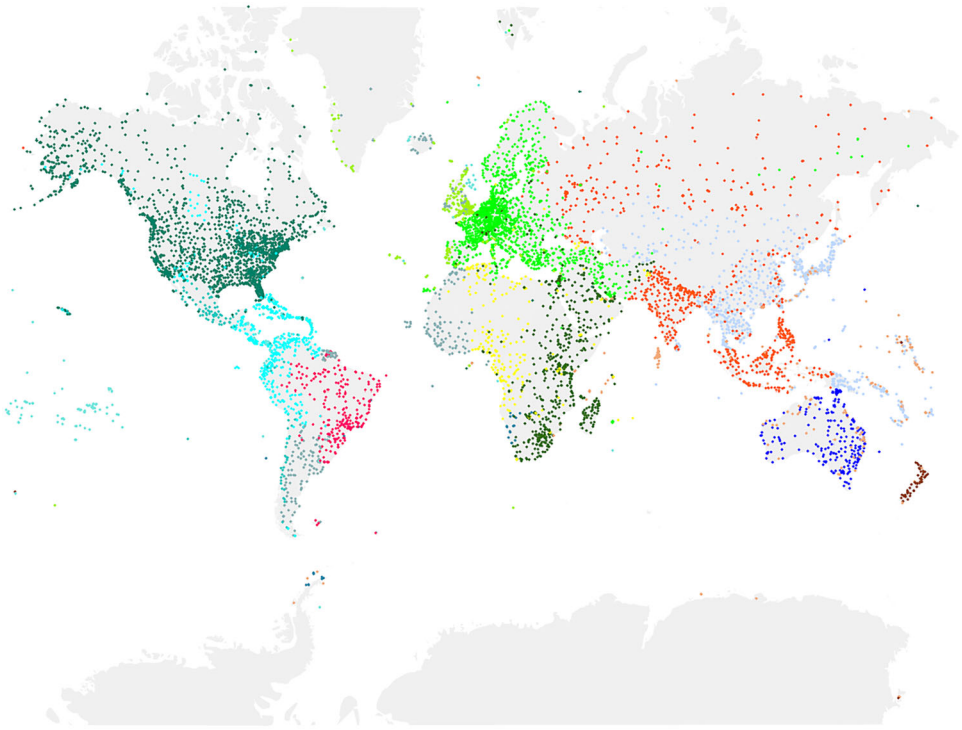


Fig. 15 Result of KMM on Open Flights data set

(NMI = 0.38) outperforms KMM (0.23), INCONCO (0.20), Integrate (0.17), DH (0.04), ClustMD (0.16) and MDBSCAN (0.02). Furthermore, ClicoT gives an insight into the interpretability of the clusters. As illustrated in Fig. 12, Cluster 12, for instance, is characterized mostly by the fuel system of *2bbl*, but also by *1bbl* and *4bbl*. Also we see that Cluster 26 is consisting of both *mpfi* and slightly of *mfi*, too. Concerning the risk analysis this clustering serves, ClicoT allows to recognize which fuel systems share the same insurance risk.

Adult Data Set Adult data set without missing values, extracted from the census bureau database, consists of 48,842 instances of 11 attributes. The class attribute Salary indicates whether the salary is over 50K or lower. Categorical attributes consist of different information, e.g., work-class,

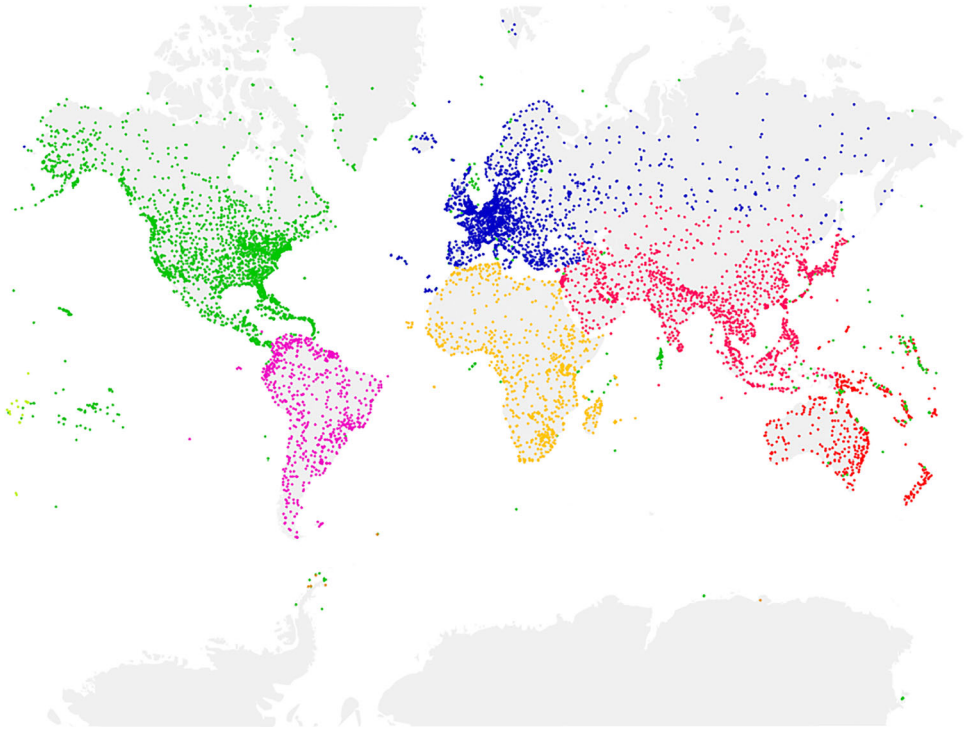


Fig. 16 Result of MDBSCAN on Open Flights data set

education, occupation. A detailed concept hierarchy is provided in Fig. 13. Although compared to INCONCO (0.05), ClustMD (0.0003), MDBSCAN (0.004), DH (0) and Integrate (0), our algorithm ClicoT (0.15) outperforms all other algorithms except KMM (0.16) which is slightly better. In order to give more insights into discovered clusters, we use two other evaluation measures, *Categorical Utility* (CU) and *Rand Index*, and compare the result of ClicoT to KMM which in this experiment is slightly more efficient in terms of NMI.

Before any comparison, we briefly explain about new evaluation strategies. Rand index is one of the most popular external clustering validation indices. Assuming P as the true clustering of data set with N data objects and C as clustering result, for each pair of data objects x_i and x_j, there are four different cases:

- *Case 1* x_i and x_j belong to the same clusters of C and the same category of P
- *Case 2* x_i and x_j belong to the same clusters of C but different categories of P
- *Case 3* x_i and x_j belong to different clusters of C but the same category of P
- *Case 4* x_i and x_j belong to different clusters of C and different categories of P

Let a, b, c, d correspond to number of pairs for the first to fourth cases and L is the total number of pairs ($L = a + b + c + d$). Thus, *Rand index* is defined as follows, with larger values indicating better results:

$$Rand\ index = \frac{a+d}{L}$$

On the other side, in order to evaluate the clustering result in terms of categorical attributes we apply the *categorical utility* criterion. CU attempts to maximize both the probability that two patterns in the same cluster have attribute values in common and the probability that patterns from different clusters have different values:

$$\text{CU}=\sum_k \left(\frac{C_k}{\mathcal{DB}} \sum_{A\in\mathcal{A}} \sum_j [P(A = A_j|C_k)^2 - P(A = A_j)^2] \right)$$

where $P(A = A_j|C_k)$ is the conditional probability that attribute A has the value A_j given cluster C_k, and $P(A = A_j)$ is the overall probability of attribute i having A_j in the entire data set. Obviously, the higher the CU value, the better the clustering performs.

Considering the *Rand index* as the metric, ClicoT (0.592) performs almost the same as KMM (0.604). However, ClicoT (0.41) slightly outperforms KMM (0.39) in terms of CU. Meaning that, clusters resulted by ClicoT are more efficiently distinguished in terms of categorical attributes compared to KMM.

On the other side, a deeper look to the clusters found by ClicoT shows interesting and interpretable results. ClicoT

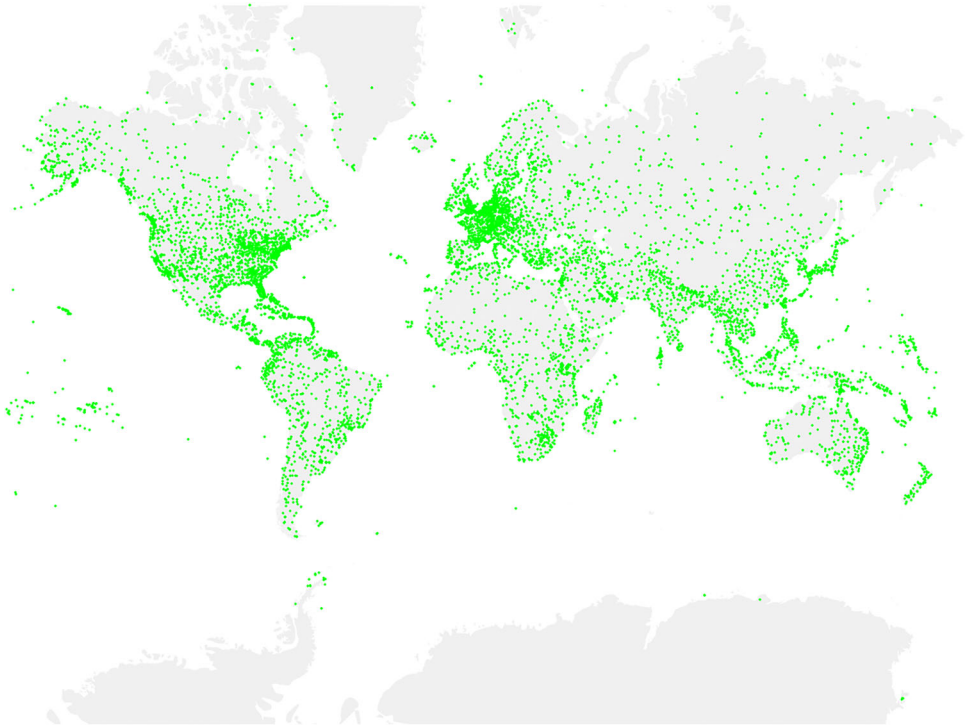


Fig. 17 Result of INCONCO and integrate on Open Flights data set

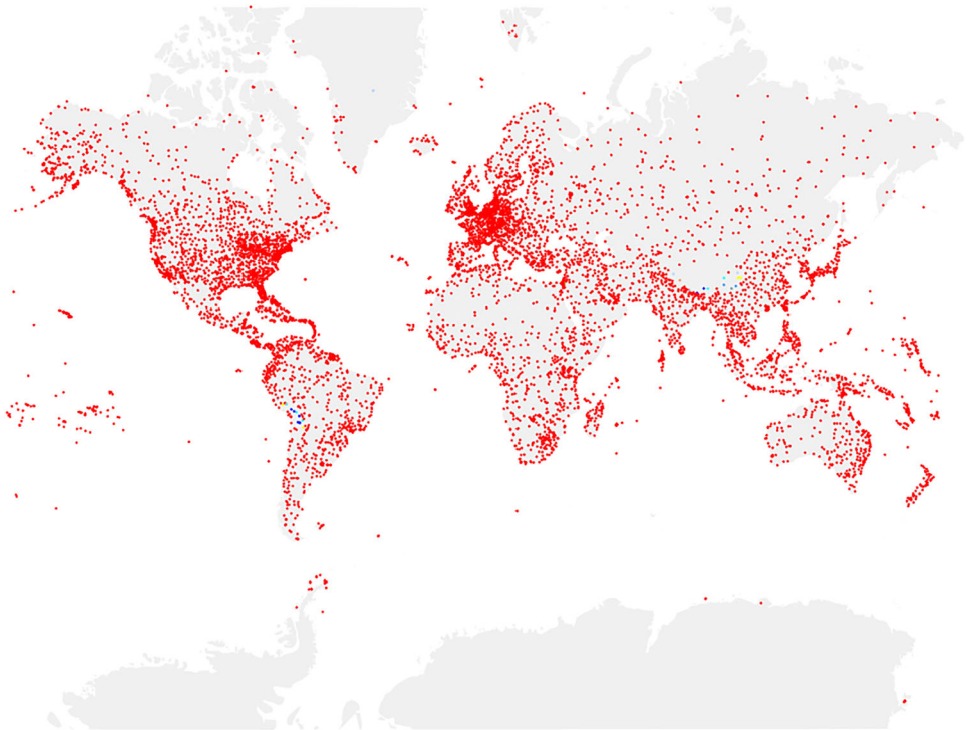


Fig. 18 Result of DH on Open Flights data set

finds 4 clusters in which Cluster 2, the biggest cluster, consists of almost 56% of objects. As Table 1 shows, in this cluster *Husband* is specified as the cluster-specific element, since it has the most deviation in terms of coding cost, but negative. The probability of instances having *Husband* as categorical value and the salary $<= 50K$ is zero in this cluster. Therefore, along with the negative deviation this means that in Cluster 2 persons with the role as husband in a family earn more than $50K$.

According to this table, for Cluster 3 *Husband* is cluster-specific as well. It has the most positive deviation and also the highest probability in this cluster, 0.99 which approves specifying this categorical value as cluster specific. In this cluster, almost 60% of persons having *Husband* as a role

earn more than 50k per year which is compatible with the overall distribution of the salary in Cluster 2 (Table 1).

Open Flights Data Set The public project Open Flights provides worldwide information about airports, flights and airlines. Here, we consider instances of airports in order to carry out a cluster analysis. The data set consists of 8107 instances each of which represents an airport. The numeric attributes show the longitude and latitude , the sea height in meters and the time zone. Categorical attributes consist of the country, where the airport is located and the day light saving time. We constructed the concept hierarchy of the country attribute so that each country belongs to a continent. Since there is no ground truth provided for this data set, we interpret the result of ClicoT (Fig. 14) and illustrate the result of applying other algorithms (Figs. 15, 16, 17, 18). INCONCO, Integrate and DH found almost only one cluster which makes any interpretation for this result nonsense (Figs. 17 and 18).

Clustering results illustrated in Fig. 14 consist of 15 clusters showing that ClicoT appropriately grouped almost geographically similar regions in the clusters. Therefore, we set the number of clusters for the other algorithms which required a user to specify it as 15. Starting from west to east, North American continent divided into five clusters. Obviously here the attribute of the time zone was chosen as specific because the clusters are uniquely made according to this attribute. In comparison with ClicoT, KMM found almost one cluster here and grouped all airports with different time zones together (Fig. 15). On the other hand, MDBSCAN groups all the airports continentally ignoring the time zone while the same concept hierarchy as ClicoT is given (Fig. 16).

Moving to the south, ClicoT pulled a plausible separation between South and North America. Considering South America as cluster-specific element and due to the rather low remaining airport density of South America ClicoT combined almost all of the airports to a cluster (red). In Western Europe, there are some clusters, which can be distinguished by their geographic location. Additionally, many airports around and in Germany are be grouped together.

7 Conclusion

To conclude, we have developed and demonstrated that ClicoT is not only able to cluster mixed-typed data in a noise-robust manner, but also yielded most interpretable cluster descriptions. By using data compression as the general principle ClicoT automatically detects the number of clusters within any data set without any prior knowledge. Moreover, the experiments impressively demonstrated that clustering can greatly benefit from a concept hierarchy. Therefore, ClicoT excellently complements the approaches for mining mixed-type data.

Acknowledgements Open access funding provided by University of Vienna.

References

1. Ahmad, A., Dey, L.: A k-mean clustering algorithm for mixed numeric and categorical data. Data Knowl. Eng. **63**, 503–527 (2007)
2. Behzadi, S., Ibrahim, M.A., Plant, C.: Parameter free mixed-type density-based clustering. In: Database and Expert Systems Applications (DEXA) (2018)
3. Behzadi, S., Müller, N.S., Plant, C., Böhm, C.: Clustering of mixed-type data considering concept hierarchies. In: Advances in Knowledge Discovery and Data Mining, pp. 555–573. Springer International Publishing, Cham (2019)
4. Böhm, C., Faloutsos, C., Pan, J., Plant, C.: Robust information-theoretic clustering. In: KDD (2006)
5. Böhm, C., Goebl, S., Oswald, A., Plant, C., Plavinski, M., Wackersreuther, B.: Integrative parameter-free clustering of data with mixed type attributes. In: PAKDD (1), pp. 38–47 (2010)
6. David, G., Averbuch, A.: Spectralcat: categorical spectral clustering of numerical and nominal data. Pattern Recognit. **45**(1), 416–433 (2012)
7. Frank, A., Asuncion, A.: UCI machine learning repository (2010). http://archive.ics.uci.edu/ml
8. He, Z., Xu, X., Deng, S.: Clustering mixed numeric and categorical data: a cluster ensemble approach. CoRR arXiv:cs/0509011 (2005)
9. Hsu, C.C., Chen, C.L., Su, Y.W.: Hierarchical clustering of mixed data based on distance hierarchy. Inf. Sci. **177**(20), 4474–4492 (2007)
10. Hsu, C.C., Chen, Y.C.: Mining of mixed data with application to catalog marketing. Expert Syst. Appl. **32**(1), 12–23 (2007)
11. Huang, Z.: Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Min. Knowl. Discov. **2**, 283–304 (1998)
12. Jian, S., Hu, L., Cao, L., Lu, K.: Metric-based auto-instructor for learning mixed data representation (2018)
13. Mcparland, D., Gormley, I.C.: Model based clustering for mixed data: ClustMD. Adv. Data Anal. Classif. **10**(2), 155–169 (2016)
14. Plant, C., Böhm, C.: Inconco: interpretable clustering of numerical and categorical objects. In: KDD, pp. 1127–1135 (2011)
15. Rajan, V., Bhattacharya, S.: Dependency clustering of mixed data with gaussian mixture copulas. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI'16, pp. 1967–1973. AAAI Press (2016)
16. Rissanen, J.: A universal prior for integers and estimation by minimum description length. Ann. Stat. **11**(2), 416–31 (1983)
17. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: ICML (2009)

18. Wang, C., Chi, C.H., Zhou, W., Wong, R.: Coupled interdependent attribute analysis on mixed data. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI'15, p. 1861–1867. AAAI Press (2015)
19. Yin, J., Tan, Z.: Clustering mixed type attributes in large dataset. In: ISPA, pp. 655–661 (2005)