Clustering with Attribute-Level Constraints

Jana Schmidt
Institut für Informatik/I12
TU München, Boltzmannstr. 3
Garching bei München, Germany
Email: jana.schmidt@in.tum.de

Elisabeth Maria Brändle
Institut für Informatik/I12
TU München, Boltzmannstr. 3
Garching bei München, Germany
Email: braendle@in.tum.de

Stefan Kramer
Institut für Informatik/I12
TU München, Boltzmannstr. 3
Garching bei München, Germany
Email: kramer@in.tum.de

Abstract—In many clustering applications the incorporation of background knowledge in the form of constraints is desirable. In this paper, we introduce a new constraint type and the corresponding clustering problem: attribute constrained clustering. The goal is to induce clusters of binary instances that satisfy constraints on the attribute level. These constraints specify whether instances may or may not be grouped to a cluster, depending on specific attribute values. We show how the well-established instance-level constraints, must-link and cannot-link, can be adapted to the attribute level. A variant of the k-Medoids algorithm taking into account attribute-level constraints is evaluated on synthetic and real-world data. Experimental results show that such constraints may provide better clustering results at lower specification costs if constraints can be expressed on the attribute level.

Keywords—constrained clustering, attribute level

I. INTRODUCTION

Clustering is a frequently used method to analyze and segment data sets and for many applications, users wish to include their knowledge about the expected result. Including such background knowledge in the form of constraints [1] may improve the clustering result or reduce the runtime. The first idea of such constraints was to restrict the instances that must (must-link) or may not (cannot-link) be grouped into a cluster. This type of constraint is called instance-level constraint. However, background knowledge is not only present on the instance level. There also is domain knowledge that can be expressed without even knowing specific instances of a given data set. Therefore, we introduce a new type of constraints: constraints on the attribute level, i.e., the properties of the instances in a cluster are described instead of the relationships between the instances themselves. We call this type of constrained clustering attribute-constrained clustering (ACC). This type of constraint has one additional benefit compared to the known instance-level constraints: the representation of background knowledge is much more compact when given on the attribute level. In summary, the contributions of this paper are as follows: First, we introduce must-Link and must-Link-Excl constraints on the attribute level in Section III. Second, the direct incorporation into a popular clustering algorithm is presented (cf. Sections III-B and III-C). Third, constraints on the attribute level are shown to be more compact (Section IV-B) than on the instance level and experimentally demonstrated to be useful (Section IV). Fourth, the relation to the well known instance-level constraints is established for each type of attribute-level constraint (cf. Sections III-B and III-C). The paper closes with a discussion.

II. RELATED WORK

The first clustering constraints in the literature were so-called must-link and cannot-Link constraints, which were incorporated into several algorithms [1], [2]. One main goal was to specify the types of constraints and to show how heuristics can be used to find a near-optimal clustering solution that satisfies all constraints. This was investigated, e.g., in the context of the k-Means algorithm [3]. Especially the use of must-Link, cannot-Link and cluster distance constraints were evaluated as to whether there exists a partition that satisfies the given constraints and if it is actually possible to find it. Although it is acknowledged that attribute-level constraints are important [4], they have not yet been explored extensively. One approach applied one type of attribute-level constraint to co-clustering [5]. The proposed constraint is not defined in general, but for specific attributes only, and may only be extended to pairs of attributes. In particular, the constraint can only be applied to a numerically ordered attribute and thus, the constraint may be defined on an interval or non-interval scale. A similar constraint setting is used to also define independent range-constraints on single attributes in a cluster [6]. Another type of constraints are cluster-level constraints [7]. The basic idea is to select clusters for a clustering out of a set of predefined possible clusters. The constraint then can define, e.g., which clusters must be incorporated in the clustering or how large, complete or disjoint clusters may become. A last approach to drive the clustering process is to define that patterns in general (e.g., frequent itemsets) should be found in the final clusters [8]. Although this approach also describes characteristics of clusters, no data set specific knowledge can be provided to the mining algorithm. In summary, attribute-level constraints are restricted in their expressiveness so far and were not yet incorporated into regular clustering.
III. CONSTRAINTS ON THE ATTRIBUTE LEVEL

This section first introduces the necessary notation. Then, the proposed constraints are formally defined including a running example. Subsequently, variations of the \textit{k}-Medoids algorithm that incorporate the attribute-level constraints directly are presented.

A. Problem Description

Let \(D = \{x_1^1, \ldots, x_m^1\}, \ldots, \{x_1^n, \ldots, x_m^n\}\) be a data set of binary instances \((x_j^i \in \{0, 1\})\), \(m\) being the number of attributes and \(n\) the number of instances. Additionally, a set of attribute-level constraints \(\Phi\) is provided. A constraint \(\varphi \in \Phi\) is in general any formula in propositional logic ranging over the propositional variables \(x_1\) to \(x_m\). In the remainder of the paper, the values 1 and 0 are interpreted as Boolean truth values, true and false. Although the propositional formulae could take any form in general (also normal forms, for instance), we restrict ourselves to conjunctions in the following. In a formula, an unnegated literal \(x_j\) means that the value of that variable has to be 1 in an instance, whereas a negated literal \(\neg x_j\) means that the variable has to take the value 0. For each attribute-level constraint \(\varphi \in \Phi\) and instance \(x^i\), expression \(\varphi(x^i)\) returns whether instance \(x^i\) fulfills constraint \(\varphi\), i.e., the instance is a model [9] for \(\varphi\).

\[
\varphi(x^i) = 1 \iff x^i \models \varphi.
\]

If this is the case, we say that instance \(x^i\) is in the \textit{scope} of constraint \(\varphi\). The task is to group the instances \(x^i \in D\) into a clustering that satisfies the given attribute-level constraints. A clustering \(C\) consists of \(k\) clusters \(C = \{C_1, \ldots, C_k\}\), where each cluster is a set of instances \(C_l \subseteq D\). Function \(f(\Phi, C)\) returns whether a clustering \(C\) satisfies the specified constraint set \(\Phi\). The conditions in which the constraints are satisfied, are given in the following sections for each specific constraint type. Although constraints can be used in any clustering scheme, this paper presents how they can be included in the \textit{k}-Medoids [10] algorithm. \textit{k}-Medoids was chosen because it is a standard clustering algorithm that is fast, easy to modify and because it was frequently considered in the field of constrained mining. Summarizing, the overall goal is to find \(k\) clusters for the data set \(D\) such that they satisfy the provided attribute-level constraints, \(f(\Phi, C) = \text{true}\), and that an objective function \(f\) is maximized.

B. Must-Link

The first constraint, \textit{must-Link} (\(ml\)), describes which instances must be clustered together due to their attribute characteristics. It is defined in Equation 2.

\[
\varphi = ml(x_1 \land \ldots \land x_m)
\]

A clustering \(C\) satisfies a set of \textit{must-Link} constraints \(\Phi_{ml}\), if and only if all instances that are in the scope of a specific constraint \(\varphi_i \in \Phi_{ml}\) are grouped into one cluster:

\[
f(\Phi_{ml}, C) = \text{true} \iff \forall \varphi_i \in \Phi_{ml} \exists C_l \subseteq C \forall x^k \in D : x^k \models \varphi_i \rightarrow x^k \in C_l
\]

The \textit{must-Link} constraint specifies that instances that are in the scope of a constraint \(\varphi_i\) must be grouped in the same cluster \(C_l\). Additionally, instances that are not in the scope of the constraint but are nearest to it, may also be grouped in that cluster. So, we obtain the instances satisfying the constraint plus the closest instances:

\[
C_l = \{x^k \mid x^k \models \varphi_i\} \cup \{x^j \mid \min_{C \in C} d(x^j, C) = C_l\}
\]

Then, the \textit{must-Link} \(\varphi_i\) is related to that cluster \(C_l\). This relation is the key point to include attribute-level constraints in the clustering process. Each instance can be checked whether it is in the scope of the \(ml\) constraint related to the clustering. An instance is in the scope of a \textit{must-Link} constraint, if it is a model for the constraint (cf. Formula 1), i.e., it has the necessary attribute setting.

Note that this type of constraint can also be related to the domain of instance-level constraints. More precisely, every attribute-level constraint \(\varphi_i\) induces, for pairs of instances \(x^k\) and \(x^j\) within the scope of the constraint, a set of instance-level \textit{must-link} constraints \(\{\text{mustLink}(x^k, x^j) \mid x^k \models \varphi_i \land x^j \models \varphi_i\}\). However, the \(ml\) constraint is not transformed into an instance level constraint but directly incorporated into the clustering process. The two main modifications necessary for the incorporation of \textit{must-Link} constraints in the \textit{k}-Medoids algorithm is first, a check to ensure that no initial medoid is in the scope of more than one \textit{must-Link} constraint. This guarantees that each constraint is only related to one cluster. The second modification takes place in the clustering step. If an instance is in the scope of exactly one \(ml\) constraint \(\varphi_j\) which is already assigned to a cluster, then the instance will immediately be placed in the corresponding cluster. If several constraints apply, the one with the nearest corresponding medoid is chosen for assignment. If the instances are not in the scope of a related constraint, the instance is assigned to its nearest cluster as usual. Last, each constraint that is not yet assigned to a cluster is inspected whether the instance is in its scope. If this is the case, the constraint is related to that cluster. In the following iterations, all instances that are in the scope of this constraint are then grouped into that cluster.

C. Must-Link-Excl

The second constraint, \textit{must-Link-Excl} (\(mlx\)), is a modification of the \textit{must-Link} constraint in a way that it not only defines which instances must be grouped together but moreover, which instances must not belong to this group. Equation 4 shows when a clustering \(C\) satisfies such a constraint set \(\Phi_{mlx}\). This is the case, if only instances...
that are in the scope of a \( \text{mlx} \)-constraint \( \varphi_i \in \Phi_{\text{mlx}} \) are combined into a cluster.

\[
F(\Phi_{\text{mlx}}, C) = \text{true} \iff \\
\forall \varphi_i \in \Phi_{\text{mlx}} \exists C_l \in C : \forall x^k \in C_l : x^k \models \varphi_i \wedge \\
\neg \exists C_j \in C : C_j \neq C_l : \exists x^m \in C_j : x^m \models \varphi_i
\]

(4)

Again, this constraint can also be expressed by a set of instance-level constraints, more precisely, a set of \textit{must-Link} and \textit{cannot-Link} constraints:

\[
\{ \text{mustLink}(x^k, x^l) \mid x^k \models \varphi_i \wedge x^l \models \varphi_i \} \cup \\
\{ \text{cannotLink}(x^k, x^l) \mid x^k \models \varphi_i \wedge x^l \not\models \varphi_i \}
\]

The first part is again a \textit{must-Link} constraint which specifies that instances which possess the given characteristics must be grouped together. However, for all instances that are not in the scope of the constraint, a \textit{cannot-Link} constraint is induced. In this case the instance cannot be grouped to the cluster, it \textit{violates} the constraint. Note, that although each attribute-level constraint can be transformed into an instance-level constraint, the opposite is not the case. Again, the attribute-level constraint \textit{mlx} is used in the clustering process: all instances which are in the scope of the constraint are clustered together. In other words, each cluster \( C_l \) contains exclusively the instances satisfied by the constraint \( \varphi_i; C_l = \{ x^k \mid x^k \models \varphi_i \} \). The scope of a \textit{mlx} constraint is defined as an \textit{mlx} constraint. Algorithm 1 shows how to include the \textit{mlx} attribute-level constraint in the \textit{k}-Medoids algorithm. The initialization is dependent on the

### Algorithm 1 mlx-k-Medoids (data set \( D \), int \( k \), \textit{mlx} constraints \( \Phi_{\text{mlx}} \))

1. \( \text{medoidsChange} = \text{true} \)
2. \( C = \text{initializeClusterMedoid}(k, D, \Phi_{\text{mlx}}) \)
3. while \( \text{medoidsChange} \) do
4.   for all \( x^i \in D \) do
5.     \( p = 1 \)
6.     if \( \exists \varphi_j \in \Phi_{\text{mlx}} \mid x^i \models \varphi_j \) then
7.       \( \text{assignIns2Clus}(c_j, C, \varphi_j, \text{belongsToCluster}()) \)
8.     else
9.       \( C = \text{getP-NearestCluster}(x^i, C, p) \)
10.      while \( \text{violatesConstr}(x^i, C, \Phi_{\text{mlx}}) \) do
11.         \( C = \text{getP-NearestCluster}(x^i, C, p++) \)
12.     end while
13.   end if
14. end for
15. \( \text{medoidsChange} = \text{calculateNewMedoids}(C) \)
16. end while

The scope of the constraint. Then, each remaining instance (from line 3) is evaluated whether it is in the scope of a \textit{mlx} constraint \( \varphi_j \). If this is the case, it is assigned to the corresponding cluster. The main adaptation to include \textit{must-Link-Excl} constraints is shown in line 9 to line 12, where the assignment of an instance to a cluster is not only dependent on the distance to its medoid but also its assignment would \textit{violate} the constraint. While this is the case, the second (or further) nearest cluster is chosen for assignment \( p \). If no such an assignment can be found, the algorithm stops.

### IV. Experiments

This section first introduces the synthetic datasets and an evaluation measure. As this paper is focused on binary instances, the Hamming distance (\( L_1 \) norm) was chosen for the clustering process.

#### A. Data Sets

To evaluate the proposed constraints, synthetic data sets were created. The basic idea for the \textit{must-Link} constraint is that it can connect instances which are very far apart but nevertheless belong to the same cluster. Then, a \textit{must-Link} constraint can help to find their connection by a small attribute description. To show the use of a \textit{must-Link-Excl} constraint, overlapping clusters are an interesting case. Many standard approaches cannot separate them, but including a \textit{must-Link-Excl} constraint allows specifying which instances must belong to another cluster. Figure 1 illustrates these ideas. Clusters with \textit{must-Link} constraints (Fig. 1a) are larger (big diameter) but are held together by the constraint. In contrast, unconstrained clusters are more compact. The data sets for \textit{must-Link-Excl} constraints consist of overlapping clusters (cf. Figure 1b) that cannot be separated with an unconstrained clustering. For each data set a predefined clustering is assumed that consists of \( k = 20 \) clusters, of which \( \text{numConstraints} \) clusters are constrained. Each cluster has a medoid of length \( \text{numAttr} \) with a probability of ones \( (x^i_j = 1) \) equal to 0.3. The parameter \( \text{numFixedAttr} \) defines how many literals are
included in the constraint. The instances for each cluster are derived from their medoids randomly inverting the medoid’s attributes (except the constrained attributes). The instances then must not be apart more than 0.1 apart from their medoid (unconstrained cluster) and 0.5, respectively. Altogether, numInstances instances are induced, where a cluster contains on average numInstances (resp. k) instances. Table I summarizes these parameters’ settings with their default values and the corresponding ranges. Altogether, for each constraint (ml, mlx) and parameter value, 20 datasets were created, to take into account the variance among data sets. To evaluate the quality of the induced clustering, the Adjusted Rand Index [11] (ARI) is measured. A second quality parameter is the change in runtime and iterations, respectively, when the constraints are incorporated into the clustering process. Due to the ordering dependency and the random initialization procedure of the k-Medoids algorithm, each experiment was repeated 10 times to eliminate incidental effects. Altogether, this gives 200 test results for each parameter value and constraint.

### B. Constraint Specification Costs

To illustrate the specification savings of attribute-level constraints compared to instance-level constraints, assume the data set setting with \( C = 4, I = 1000, k = 20 \). On average, each cluster contains 50 instances so that for a clustering with instance-level must-link constraints at least 200 constraints had to be provided to make sure that all instances of the constrained clusters are grouped appropriately. Comparing this to the number of attribute-level constraints (4), the constraint specification compression is evident. Moreover, during the clustering process with instance-level constraints, each instance must be compared against all instances of each cluster, whether there applies an instance-level constraint. In the worst case, this results in \( O(n^2) \) checks. Comparing this to attribute constrained clustering, only \( O(|C| \cdot n) \) scope checks have to be performed (\(|C| \) being the number of clusters), as only the related attribute-level constraint is tested against each instance. Especially for large datasets and few clusters, using attribute constraints can thus result in large savings of runtime.

### C. Results must-Link and must-Link-Excl

This section presents the resulting ARI and runtimes depending on the four varied parameters. For each evaluation, one figure (except for numAttr) is given that compares the ARI when must-Link (dashed curves) must-Link-Excl (bold curves) constraints are used (unfilled) for the clustering or not respectively (filled). To judge the effect of the constrained clustering on the runtime, a subsequent table gives the runtime (for one constraint type exemplarily) and the number of iterations (rounds) that were needed for the clustering with or without using the constraints.

1) Results numAttr: The clustering process is more difficult for small instances \((0 < ARI_{no} < 0.2, \text{no results shown})\), no matter whether constraints are used or not. For higher dimensions, the clustering results become better but do not exceed an \( ARI \) of 0.83 (when using ml-constraints) and 0.80 (without ml-constraints), respectively. Using mlx constraints always leads to better results (0.2 better ARI) than without constraints, for all instance dimensions. In contrast, ml constraints lead to better results only in small dimensions (0.15 better ARI for fewer than 100 dimension).

Table II shows the runtime and the number of rounds that were needed for the clustering. The larger the instances, the longer needs the clustering. This is mainly due to the longer instance comparison time, which of course increases for higher dimension. In fact, the clustering becomes easier for higher dimension for ml constraints, because instances can be distinguished better. Thus, the larger the instances’ dimensions, the more a clustering can benefit from ml constraints in terms of runtime. In contrast, mlx constraints lead to a slightly higher runtime (for 1000 attributes 7.71s with mlx vs. 6.8s without mlx). An explanation for this is that the initial assignment for an instance may be revised because it violates a constraint. Thus, additional computations have to be conducted. The number of rounds needed is decreased in higher dimension for both constraint types.

2) Results numConstraints: Figure 2 shows that if only few constraints are given, k-Medoids still gives a
good solution. This is a result of the data creation process. Unconstrained clusters are quite dense so that the grouping of the instances is straightforward. The more constraints are included, the more separated the instances become. Then, the cluster assignment without constraints is more and more difficult. In contrast, for a data set that contains very scattered instances (numFixedAttr → 20), the constraints are of much greater value to the clustering. The highest ARI-gain is observed for a large number of constraints. Table III shows that the runtime decreases when more constraints are included. The time that is saved by fewer rounds is consumed to process the constraints. Using the constraints is beneficial for both types of constraints, and the more constraints are used, the more savings can be achieved.

3) Results numFixedAttr: Overall, the ml and mlx constraints provide additional information for the clustering (cf. Figure 3), which is evident especially for mlx constraints. The best improvement can be observed when only few attributes are included in the constraints. The ARI-gain decreases for higher values of numFixedAttr, because then the instances become more and more similar so that k-Medoids is also able to induce the correct clustering. The ml-runtime increases with higher values of numFixedAttr, no matter whether the constraints are considered or not (not illustrated). Here, the unconstrained clustering is even comparable in the runtime. Although the runtime does not increase much for must-Link-Excl constraints, the runtime is also comparable to the case when no constraints are considered. The number of rounds is the same for all values of numFixedAttr, when using constraints, but slightly decreased for the standard approach. This can be explained by the fact that the separation of the instances becomes easier, because more fixed attributes mean that the resulting instances are more similar. Thus, the standard clustering converges earlier. However, in most cases the constrained clustering needs fewer rounds than the standard clustering approach.

4) Results numInstances: Figure 4 shows the ARI for differently sized data sets. The data sets include from 40 to 4000 instances each. The inclusion of both types of constraints is beneficial. Throughout the parameter values, an average improvement of 4.2% (ml) and 12% (mlx) can be observed. The biggest improvement was achieved for small data sets (ml: 6.8% and mlx: 15% respectively) but no general trend can be inferred from these numbers. Table IV shows that larger datasets increase the runtime and iterations. The inclusion of constraints is beneficial for the runtime and rounds. Both are lower when compared to the standard clustering approach.

5) Results on the Real-World Data Set: A last experiment shows the usability of the constraints on a real-world data set. The UCI zoo data set\(^1\) was chosen for this experiment. It contains 101 binary instances that describe animals. Additionally, the corresponding cluster membership (the biological class) is included. Six constraints were created using biological background knowledge about the similarities of animals and the given attributes: ml1(milk), ml2(feathers), ml3(fins & eggs), ml4(4Legs & toothed & eggs), ml5(6Legs & breathes), ml6(backbone & ~breathes). The same constraints were also created for the mlx type. Then, sets of constraints were created, including one to six constraints each, in order to show the individual contribution for each constraint as

\(^1\)http://archive.ics.uci.edu/ml/datasets/Zoo
ARI shows the mean was repeated 50 times for each constraint set. Figure 5 then used for the clustering process. Again, the clustering as well as the benefit of their combinations. For each possible combination, a separate constraint set was constructed and then used for the clustering process. Again, the clustering was repeated 50 times for each constraint set. Figure 5 shows the mean ARI for each constraint set size as well as the baseline result when no constraints are considered. The leftmost point(s) (above 1) give the average ARI when the constraint set consists of only one constraint, while the point(s) above 2 show the average ARI for the combination of two constraints each. The results indicate that the more constraints are applied, the better is the clustering quality. However, not every constraint or constraint combination performs equally well. Figure 5 gives the mean ARI for each mlx constraint and their combinations. Although the general trend shows that the inclusion of such constraints is beneficial, there exist constraints (and combinations) that do not improve the result notably. Especially the inclusion of constraints mlx3 and mlx6 leads to only small improvements. Most significantly, the constraint mlx6 alone even decreases the ARI compared to the baseline (leftmost point in Figure 5). This shows that such constraints (as any other type of constraints) may substantially improve the clustering performance, but need to be used with care.

V. Conclusion

In this paper we transferred the notions of the instance-level constraints must-Link and cannot-Link to the attribute level, where the cannot-Link effectively becomes a new type of constraint: must-Link-Excl. An adaptation of the well-known k-Medoids algorithm was presented that is able to incorporate the provided constraints. Each constraint was evaluated concerning several parameters. The results indicate that it is not only possible to define constraints on the attribute level, but also that they may be beneficial in the clustering process. Moreover, we discussed on which types of clustering problems the constraints should be applied and how much the specification costs can be reduced compared to standard instance-level constraints. For future work, we would like to consider the combination of the presented constraints, how this combination will affect the runtime and how a probably NP-hard problem (like for other constraint clusterers [12]) can be avoided. Moreover, we would like to examine how constraints that express, e.g., the amount of shared attributes between clusters, can be formalized and incorporated into the constrained mining process.

REFERENCES