Using Belief Networks to Neutralize Known Dependencies in Conceptual Clustering

Jan Ramon and Luc Dehaspe
{Jan.Ramon,Luc.Dehaspe}@cs.kuleuven.ac.be

Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Leuven, Belgium

Abstract. Conceptual clustering is an important descriptive data-mining task. Several good conceptual clustering algorithms exist in the attribute-value representation. These algorithms typically do not take into account dependencies known to the user. This is a problem if the user does not want these dependencies to influence the clustering process. A typical example is where a taxonomy over attributes is known to exist and should not be rediscovered. In this paper we propose to solve this problem by using belief networks to describe the clusters. This solution has an important application in relational clustering. It allows the use of massively dependent first order queries as attributes. We present an implementation of our approach as an extension of the COBWEB algorithm and evaluate our method on propositional and relational data.

1 Introduction

Clustering is an important descriptive data-mining task. Its goal is to divide a given set of examples into several subsets (clusters) such that examples of the same cluster are similar, while examples of different clusters are not. For conceptual clustering, it is also required that a (symbolic) description is given for each cluster. Several good conceptual clustering algorithms exist in the attribute-value representation (see e.g. COBWEB [11], AUTOCLASS [4]).

However, these algorithms assume the attributes of the data to be independent within each cluster. This assumption does not hold in all cases. There can be redundant information, known by users and therefore uninteresting to them. Also, the redundancy can be caused by the language in which the information is represented.

In this paper we propose a method to deal with these problems by allowing attributes to be dependent on each other and by allowing the user to specify his prior knowledge.

Recent machine learning algorithms (such as TILDE [2], ICL [8], WARMR [9]) use a more expressive first order (or multiple relation) representation. Some systems that do conceptual clustering on examples represented in first order logic exist, but often still need some information in propositional form. E.g. the system TIC [3] builds clustering trees but uses a propositional distance measure between examples and/or clusters.

The search for a good clustering is typically guided by one of two heuristics: an evaluation measure (also called objective function) on clusters or a distance function. Both heuristics are in the propositional case a function of values of attributes. However, in the first order case, there is no fixed number of attributes, and so one of these two heuristics would have to be upgraded. While distance measures between first order objects exist [19], [10], they are often quite ad hoc, representation-dependent and computationally expensive.

In this paper, we propose an alternative approach. We use first order queries as attributes. Because of the generality relations between these queries, the obtained attributes are highly dependent on each other. Therefore, we use our method to deal with dependent attributes. This has the advantage that no distance is needed, only the objective function has to be upgraded in a representation-independent way.

The paper is structured as follows: Section 2 describes a method to deal with non-independent attributes. Section 3 describes the extension to multiple relations. Next, section 4 describes an implementation where we incorporate these methods in the COBWEB algorithm. Section 5 shows some experimental validation of our method. Section 6 discusses some related work. Finally, section 7 gives some conclusions and ideas for further work.

2 Using known dependencies

Many clustering algorithms assume the attributes of the data to be independent. This assumption does not hold in all cases. In this section, we first describe some examples where this assumption does not hold. Next, we briefly review bayesian clustering and in the last part we describe the extension of the cluster description to belief networks.

2.1 Uninteresting dependencies known by the user

There can be dependencies in the data, known by the users and therefore uninteresting to them. Suppose that a supermarket has the database shown in table 1.

CLIENT

customer	1	2	3	4	5	6	7	8	9
income	H	H	L	H	L	L	H	L	H
social class	H	H	L	H	L	L	L	L	H
has_a_car	y	y	n	y	n	n	n	n	y
education_level	H	H	L	H	L	L	H	L	H
buys_pizza	n	y	n	y	n	y	n	y	n
buys_beer	n	y	n	y	n	y	n	y	n

Table 1. Supermarket example

Most clustering algorithms will produce two clusters: the cluster {1, 2, 4, 7, 9}, which could be labeled as the set of the 'rich' people and the cluster {3, 5, 6, 8} of 'poor' people. Perhaps an institution doing demographic research would be happy with this. However, the supermarket already knows this and is more interested in less trivial data, e.g. the fact that pizza and beer is often bought together.

2.2 Dependencies caused by the representation

Dependencies can also be caused by the language in which the examples are represented. One of the simplest situations where this can happen is in the case of attributes that are not applicable for some examples. For instance, an insurance company can have a database of clients, with information whether they have a life insurance, a car insurance, etc. If they have a life insurance, other information is known, e.g. the amount and the chosen type of the agreement. For those who don't have such an insurance, these attributes are not applicable. The fact that two examples have both 'not applicable' for some attribute does not affect their similarity.

Other examples of cases of dependencies caused by the representation are when a taxonomy over the attributes is available (see section 5.2) and when truth values of non-independent queries are used as attributes (see section 3.2).

2.3 Propositional clustering objective function

In this section we briefly review some elements from clustering in the propositional case. In the Bayesian clustering approach, a hypothesis is a description of a clustering of the set of all examples D. In the Bayesian clustering setting (see [4]), this typically means that the hypothesis contains for each cluster a probability distribution for the examples occurring in that cluster, and also a probability distribution giving the probabilities of an example belonging to each cluster. The clustering task is then to find the posterior distribution of the hypotheses given the examples. Typically, only the most probable hypothesis is generated and presented to the user.

Let $D = \{E_1, \ldots, E_N\}$ contain N examples. Let $\mathcal{D}$ be the (unknown) probability distribution from which the examples are drawn. Let $\mathcal{H}$ be the set of all possible hypothesises. The probability that some $H \in \mathcal{H}$ explains the data D is $P(H|D)$ and satisfies the rule of Bayes:

$$P(H|D) = \frac{P(D|H).P_{\mathcal{H}}(H)}{P_{\mathcal{D}}(D)} \tag{1}$$

with $P_{\mathcal{H}}(H)$ the prior probability of H. $P_{\mathcal{H}}(H)$ can depend on the hypothesis complexity (a more complex hypothesis is less likely) or on some other prior knowledge. $P_{\mathcal{D}}(D)$, the prior probability of the data can be treated as a normalisation constant and doesn't need to be calculated explicitly. All examples are assumed to be independent, so

$$P(D|H) = \prod_{i=1}^{n} P(E_i|H) \tag{2}$$

Let H partition D into k clusters $C_1, \ldots, C_k$. Let the $P(E_i \in C_j)$ be the probability that the example E_i belongs to the cluster C_j according to the hypothesis H. Then

$$P(E_i|H) = \sum_{j=1}^{k} P(E_i \in C_j) P(E_i|C_j)$$

where $P(E_i|C_j)$ is the probability of drawing E_i from the distribution of the cluster C_j. Since we want to use our method to make an extension to COBWEB and since COBWEB assigns each example to exactly one class (unlike e.g. AUTOCLASS), we can simplify this formula: $P(E_i \in C_j)$ equals 1 iff C_j is the cluster E_i belongs to and 0 otherwise.

$$P(E_i|H) = P(E_i|C_{c_i}) \tag{3}$$

where C_{c_i} is the cluster E_i is assigned to. Let $E_i = (e_{i1}, \ldots, e_{ia})$ with a the number of attributes. If the attributes are assumed to be independent within each cluster, this probability can be written as

$$P(E_i|C_j) = \prod_{l=1}^{a} P(A_l = e_{il}|C_j) \tag{4}$$

where $P(A_l = e_{il}|C_j)$ is the probability that the l-th attribute has the value e_{il} in the cluster C_j. Combining (1), (2), (3) and (4) and taking the logarithm gives

$$\log\big(P(H|D)\big) = \log(P_{\mathcal{H}}(H)/P_{\mathcal{D}}(D)) + \sum_{i=1}^{n} \sum_{l=1}^{a} \log\big(P(A_l = e_{il}|C_{c_i})\big)$$

This can be rewritten as

$$\log\big(P(H|D)\big) = \begin{array}{l} P_{prior} + \sum_{j=1}^{k} \sum_{l=1}^{a} \sum_{s=1}^{v_l} count(A_l = V_{ls}|C_j) \\ \log\big(P(A_l = V_{ls}|C_j)\big) \end{array}$$

where V_{ls}, $s = 1..v_l$ are the possible values of the l-th attribute and $count(A_l = V_{ls}|C_j)$ the number of examples with l-th attribute V_{ls} in C_j and $P_{prior} = \log(P_{\mathcal{H}}(H)/P_{\mathcal{D}}(D))$. This can be rewritten as

$$\begin{aligned} &\log\big(P(H|D)\big) \\ &= P_{prior} + n. \sum_{j=1}^{k} P(C_j) \sum_{l=1}^{a} \sum_{s=1}^{v_l} P(A_l = V_{ls}|C_j) \\ &\quad \log\big(P(A_l = V_{ls}|C_j)\big). \end{aligned}$$

It can be concluded that $P(H|D)$ is a linear function of the objective function presented in [5]:

$$\begin{aligned} &\tfrac{1}{k}\sum_{j=1}^{k} P(C_j) \sum_{l=1}^{a}\sum_{s=1}^{v_l}\Big[\\ &\quad P(A_l = V_{ls}|C_j)\log\big(P(A_l = V_{ls}|C_j)\big) - \\ &\quad P(A_l = V_{ls})\log\big(P(A_l = V_{ls})\big)\Big] \end{aligned} \tag{5}$$

except for the factor $1/k$, which is introduced because we prefer simple hypotheses with few clusters. This can be seen as an information-theoretic analog to the well-known Partition Utility [12] which is

$$\frac{1}{k}\sum_{j=1}^{k} P(C_j)\sum_{l=1}^{a}\sum_{s=1}^{v_l}\Big[P(A_l = V_{ls}|C_j)^2 - P(A_l = V_{ls})^2\Big] \tag{6}$$

In this paper however, we will use equation (5) as it is more straightforward to extend to represent belief network probabilities.

2.4 Using belief networks as models

In the normal setting for clustering, the description of each cluster consists of an independent probability distribution for each attribute. To avoid problems that can occur when attributes are not independent, we propose to use the more expressive description language of belief networks. We first introduce some definitions.

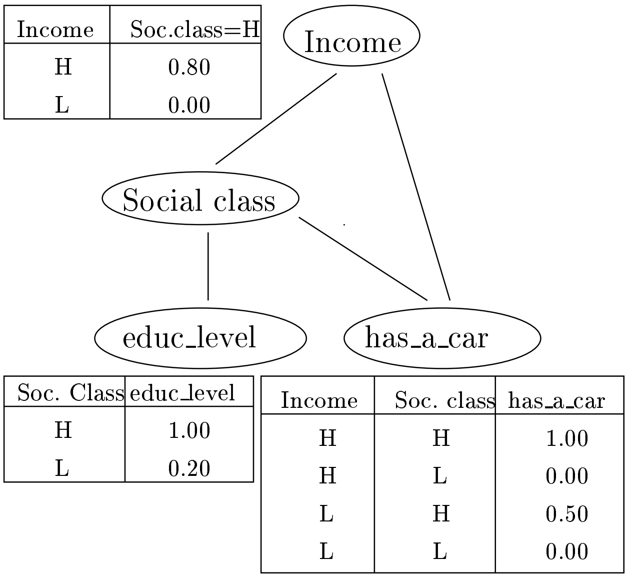
Definition 1 (ancestor,successor). *If $u \in V$ is a node in a graph $G(V,R)$, we define the set of all ancestors $anc^*(u) = \{v \in V \setminus \{u\} | (v,u) \in R\}$, the set of direct ancestors $anc(u) = anc^*(u) \setminus \bigcup_{v \in anc^*(u)} anc^*(v)$, the set of all successors $succ^*(u) = \{v \in V | (u,v) \in R\}$ and the set of all direct successors $succ(u) = succ^*(u) \setminus \bigcup_{v \in succ^*(u)} succ^*(v)$.*

Definition 2 (belief network). *A* belief network *(also called a* Bayesian network*) is a directed acyclic graph $G(V,R)$ where the nodes are random variables. Associated with each node, there is a probability table indicating the probability distribution of the corresponding variable given the values of its direct ancestors.*

An edge (u,v) in a belief network has the meaning "if the state of v is not known for sure, knowledge about the state of u may influence the knowledge about the state of v, no matter what else is known". Figure 1 shows an example of a belief network, which could be a representation of the prior knowledge of the supermarket in our example from section 2.1.

According to a belief network, the probability of an example E_i is

$$P(E_i) = \prod_{l=1}^{a} P(A_l = e_{il}|anc(A_l))$$



Income	Soc.class=H
H	0.80
L	0.00

Soc. Class	educ_level
H	1.00
L	0.20

Income	Soc. class	has_a_car
H	H	1.00
H	L	0.00
L	H	0.50
L	L	0.00

Fig. 1. Known dependencies

where $P(A_l = e_{il}|anc(A_l))$ is the probability of the l-th attribute having the value e_{il} given the values of the attributes in $anc(A_l)$. Given a set of examples D a belief network $G(V, R)$ on D, and k belief networks $G_j(V, R)$ on k clusters C_j of D, equation (4) becomes

$$P(E_i|C_j) = \prod_{l=1}^{a} P(A_l = e_{il}|C_j, anc_j(A_l)) \tag{7}$$

where $anc_j(A_l)$ is the set of direct ancestors of the l-th attribute in the belief network $G_j(V, R)$. The expression (5) becomes

$$\begin{aligned} &\tfrac{1}{k} \sum_{j=1}^{k} P(C_j) \sum_{l=1}^{a} \sum_{s=1}^{v_l} \Big[\\ &P(A_l = V_{ls}|C_j, anc_j(A_l)) \log\big(P(A_l = V_{ls}|C_j, anc_j(A_l))\big) \\ &-P(A_l = V_{ls}|anc(A_l)) \log\big(P(A_l = V_{ls}|anc(A_l))\big)\Big] \end{aligned} \tag{8}$$

It is hard to automatically infer the optimal belief network structure for a set of examples. Therefore, we use the same structure for the networks $G_j(V, R)$ as the structure of $G(V, R)$ and adapt the probabilities. This is sufficient for a good objective function in our applications. An algorithm that could improve the structure of the belief network of a cluster such that the estimated probability of the examples increases, would provide additional indications for similarity between the examples in that cluster, and would probably give even better results.

Table 2 illustrates the effect on our small supermarket example. Two possible partitions of the database are considered. The left column gives the values for equation (5), assuming independent attributes. The right column gives the values for equation (8), taking into account the prior knowledge of the supermarket, represented in figure 1. When the prior knowledge is taken into account, other clusters will be formed. These new clusters can give more insight in the things the supermarket really wants to know.

	indep. attr.	belief net
$\{\{1,2,4,7,9\},\{3,5,6,8\}\}$	-1.38	-1.18
$\{\{1,3,5,7,9\},\{2,4,6,8\}\}$	-1.97	-0.64

Table 2. Value of different ways of clustering with and without belief network

3 Upgrading to first order logic

In this section we further upgrade our approach to first order logic in a way similar to [13]. An other approach would be to extend the models describing the clusters further from bayesian nets to probabilistic logic models. However, though several proposals for such models were made (see e.g. [6], many problems are still unanswered in this area (e.g. learning the models), which make that alternative problematic.

Due to space limitations, we briefly illustrate rather than extensively introduce the first-order logic concepts used in the rest of the paper. These concepts are taken mostly from the "learning from interpretations" setting of Inductive Logic Programming. Formal aspects of this induction paradigm are discussed in [7].

3.1 Attributes and conjunctive queries

In this section, we derive attributes from a relational database. In contrast to [15], we do this exhaustively, which allows us in the next section to determine the dependencies between them in an easy way.

Consider a relational database D that consists of three relations shown in table 3 and conjunctive query[1]

$$Q_1\text{: } \mathit{customer}(X) \wedge \mathit{parent\_of}(X,Y) \wedge \mathit{buys}(Y,\mathit{coke})$$

[1] In practice we use Prolog to represent data and patterns. This means for instance that we can add recursively defined predicates to the database and use function symbols in the queries. In this paper we restrict ourselves to relational algebra merely to simplify the discussion.

Customer
Id
a
b
c
d

Buys	
Id	Item
a	wine
b	coke
b	pizza
d	wine

Parent_Of	
Id	Id
a	b
a	c
c	d

Table 3. A relational database D

i.e., "there exists a customer X who is parent of a coke buyer Y". If indeed such X, Y, Z are found in database D, query Q_1 *succeeds* w.r.t. D, otherwise, the query *fails* w.r.t. D. Notice query Q_1 succeeds with answerset $\{X = a,\ Y = b\}$.

Query Q_1 can be interpreted as a boolean attribute Q_1^A of customers: each customer either is or is not "parent of a coke buyer". To find Q_1 for a particular customer, we substitute the variable X in Q_1 with the customer's identifier, and evaluate against the database. For instance, query

$$customer(a) \wedge parent\_of(a, Y) \wedge buys(Y, coke)$$

i.e., "customer a is parent of a coke buyer Y", succeeds with $Y = b$. Hence attribute Q_1^A has value 1 for customer a. Observe substitutions of X with the three remaining customer identifiers b, c, d all result in failing queries, such that for these customers attribute Q_1^A has value 0.

In general, we can view conjunctive queries as attributes of examples (here customers) if they contain a set of variables (here singleton $\{X\}$) such that each substitution of those variables corresponds to an identifier of the example. The most straigthforward way to achieve this is to (1) have a separate relation in the database with example identifiers (here Customer), and (2) make the corresponding atom (here $customer(X)$) obligatory in each conjunctive query. In that case, we can roughly define the *frequency* of a conjunctive query as the number of example identifiers for which the query succeeds.

Given the above interpretation of conjunctive queries as boolean attributes, and given three more conjunctive queries with the obligatory $customer(X)$ atom:

$$Q_2\text{: } customer(X) \wedge parent\_of(X, Y) \wedge buys(Y, wine)$$
$$Q_3\text{: } customer(X) \wedge parent\_of(X, Y)$$
$$Q_4\text{: } customer(X) \wedge buys(X, Y)$$

we can generate the attribute-value description of our relational database D shown in table 4. Obviously, table Customer_avl is not equivalent to database D, but for some purposes, for instance clustering customers, we may judge it comes sufficiently close. We come back to this issue in paragraph 3.3.

3.2 Belief networks and query lattices

The customer data are now in a format similar to that in table 1 and we can apply the ideas introduced above. If for instance a dependency is assumed between the

CUSTOMER_AVL

ID	Q_1^A	Q_2^A	Q_3^A	Q_4^A
a	1	0	1	1
b	0	0	0	1
c	0	1	1	0
d	0	0	0	1

Table 4. Attribute-value representation of D

attributes "is parent of coke buyer" (Q_1^A) and "is parent of wine buyer" (Q_2^A) than this assumption can be expressed as before.

On top of these *subjective* dependencies however, a number of *objective* dependencies between attributes Q_i^A exist based on the logical "more-general-than" relation between corresponding conjunctive queries Q_i. For efficiency reasons, we here concentrate on a stronger variant of logical implication called θ-subsumption [18]. The advantage of θ-subsumption is that it can be computed directly from the syntax of queries: we say query Q_g is *more-general-under-θ-subsumption* than query Q_s if and only if there exists a substitution θ of the variables in Q_g such that all atoms in $Q_g\theta$ (i.e., the result of applying substitution θ to query Q_g) occur in query Q_s. For instance, by this definition, query Q_3 is more general than both queries Q_1 and Q_2, in both cases with the empty substitution. No other generality relations exist between queries Q_1-Q_4. Observe in particular that no substitution can be found that makes Q_4 a subset of Q_1 or Q_2. This agrees with the intuition that, e.g., "parent of a coke buyer" is not a special case of "buyer of something".

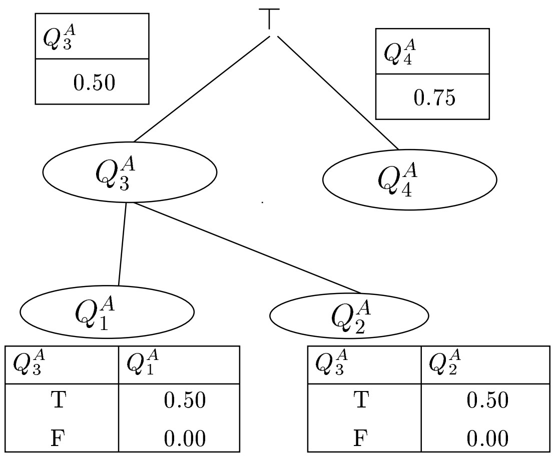


Fig. 2. Objectively known dependencies

The generality-under-θ-subsumption relation induces a lattice structure on a set of conjunctive queries Q_i, as explained, for instance, in [17]. This structure trivially maps to a belief network over the corresponding boolean attributes Q_i^A, such that each direct ancestor of node Q_i^A corresponds to a maximally specific generalization under θ-subsumption of query Q_i. The probability tables associated with each node can be computed directly from the data. The belief network for attributes Q_1^A-Q_4^A is shown in Figure 2.

3.3 Attribute selection

To make the link between conjunctive queries (lattices) and attributes (belief networks) operational we still have to solve the problem of how to select queries such as Q_1-Q_4 from an infinite space of possible queries on database D. Part of the solution is offered by a so-called *declarative language bias* formalism, well known in relational learning. Such a formalism allows the user to constrain the space of queries to sensible ones, e.g., by imposing type and mode constraints on variables. We further assume that the language bias definitions provided by the user indirectly – via the query lattice – determine the structure of the belief network used throughout the clustering process. As explained immediately below, extra cluster-specific constraints will be used to "suppress" some of the nodes in this possibly gigantic (infinite) network.

Within particular clusters, we use an additional constraint based on the observation that the influence of rarely succeeding, hence called *infrequent*, queries on clustering is negligible. Observe in that respect that the influence of a conjunctive query is at best proportional to $r.log(r)$, with r the relative frequency of the query. Therefore, when setting the parameters of the belief network for a particular cluster, we can safely ignore all nodes whose relative frequency is below a pre-defined threshold t, i.e., assume they have frequency 0. Since we only set and use the parameters associated with queries whose frequency exceeds t, we need an algorithm that selects these frequent queries from the user defined language of queries. WARMR [9] is such an algorithm.

WARMR is an instance of the family of levelwise frequent pattern discovery algorithms [16] that look at a level of the lattice at a time, starting from the most general pattern. WARMR iterates between candidate generation and candidate evaluation phases: in *candidate generation*, the lattice structure is used for pruning non-frequent queries from the next level; in the *candidate evaluation* phase, frequencies of candidates are computed w.r.t. the database. Pruning is based on monotonicity of the generality-under-θ-subsumption relation w.r.t. frequency: if a query is not frequent then none of its specializations is frequent. So, while generating candidates for the next level – this is essentially done by adding atoms to frequent queries of the last level – all the queries that are specializations of infrequent patterns can be pruned.

4 The REMIND system

We implemented the ideas in the previous sections in a system called RElational clustering MINus Dependencies. On the highest level, Fisher's COBWEB algorithm was used. Essentially, it incrementally creates a hierarchical clustering, starting with an empty tree and updating it each time an example is processed. The algorithm sorts example down the tree, at each step choosing the best of four possible operations using an objective function.

4.1 The COBWEB algorithm

The four possible operations in COBWEB at each node are: adding the example to a subnode, creating a new subnode with as only element the new example, joining two subnodes adding the example to one of them, and splitting a subnode into its subnodes adding the example to one of them. More details can be found in [11].

4.2 Incremental WARMR

Formula (8) was used as objective function. To calculate the value of this objective function, the frequency of the different values for all attributes is needed. These statistics are stored for each node in the tree, and incrementally updated as new examples are added.

When attributes are independent, the best one can do is to count the frequency of each value of each attribute. However, when there is a relation between the attributes (e.g. a taxonomy (see below for an example), or a query lattice in a relational setting), the data is often sparse and a more efficient approach is possible.

We extended the WARMR algorithm to work incrementally. Given is a set of already processed examples, and a set of frequent patterns on these examples with their frequencies. When a new example has to be added, the query frequencies are updated in a level-wise order. If a query does not succeed for the example, none of the frequencies of specialisations of that query have to be updated.

5 Experiments

In this section we report on some experimental results with our REMIND system. Evaluation of clustering systems is difficult. One of the most frequently used methods is to use clustering to predict unknown features of examples, and use the accuracy as performance measure. As we explain in section 6.2 we do not expect a large performance gain for normal propositional problems. Therefore, in this section we focus on showing that our technique can be used in situations where the normal clustering algorithms fail.

We used some synthetic datasets for our experiments. These can be obtained from http://www.cs.kuleuven.ac.be/˜ml/.

5.1 Benchmarks

We do not use the standard Cobweb objective function partition utility, but the information-theoretic equivalent. [12] proposes this as a good alternative, but does not report experimental results. Because of this, we first ran our algorithm on some standard UCI benchmarks (soybeans, voting, breast), for both objective functions. We observed that Cobweb induces similar trees for both objective functions. Also, the trees gave similar class prediction accuracies in the leave-one-out crossvalidations we did.

5.2 A transaction database

In a transaction database, each example is the set of all items some customer bought in one transaction. Large transaction databases typically contain many attributes and are very sparse. In these applications, to have oversight over the data, it is important to define a taxonomy (see also e.g. [1]). A taxonomy is an *is_a* hierarchy, an example of which is given in figure 3

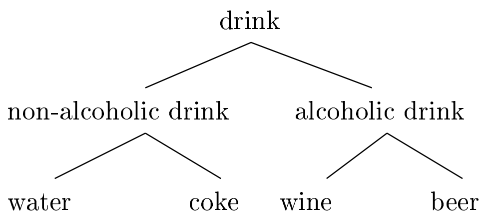


Fig. 3. A taxonomy

	items	taxonomy nodes
1	729	270
2	144	36
3	729	30
4	2916	121

Table 5. Some parameters of the transaction datasets

We created 4 synthetic transaction databases. Each database contains transactions from three groups of customers with different characteristics. Table 5 gives for each database the number of items and the number of taxonomy nodes.

Then three different versions of Cobweb were run.

1. Standard COBWEB on the examples without the taxonomy (*naive*), In this setting, each example has one attribute for each item that can be bought, giving the number of items the customer bought.
2. Standard COBWEB on the database with all ancestors in the taxonomy added as attributes (*taxonomy*). e.g. the attribute *drink* from figure 3 would be 1 iff some customer bought either one *coke*, one *beer*, one *wine* or one *water*.
3. REMIND which makes use of the knowledge in the taxonomy (*Remind*).

Each time, the test examples were sorted down the tree and the class was predicted in a leave-one-out cross-validation. The results obtained are summarized in table 6.

We observed that *taxonomy* performs best when a large part of the taxonomy is over the attributes important for the clustering (the attributes we defined different distributions for in the different clusters) such as in set 1 (in this dataset, 189 of the 270 nodes in the taxonomy are generalisations of attributes important for clustering). *Taxonomy* performs worse when there is a taxonomy over irrelevant (in this case pure noise) attributes such as in set 3, even if the taxonomy is rather small. *Naive* performs badly when there are too many items to see similarities between transactions, such as in set 4 where we generated more items (while customers remained buying on average the same amount of items). Therefore, we expect *naive* to do even worse when the databases are scaled up further to real-world size. For all sets, *Remind* is about as good as the best of the first two. We can conclude that REMIND is a more robust system.

	naive	taxonomy	Remind
set 1	38%	43%	42%
set 2	76%	65%	74%
set 3	80%	51%	81%
set 4	44%	58%	60%

Table 6. Results on the transaction databases

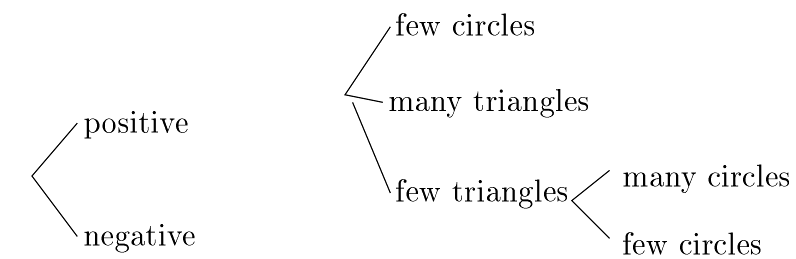
5.3 Bongard

We also did some experiments on bongard datasets. A bongard dataset contains scenes of figures (triangle, circles, ...), in certain relations to each another. e.g. the query

$$Q : triangle(X), circle(Y), circle(Z), not(Y = Z), in(X, Y)$$

succeeds for a scene if there are two circles and a triangle that is in one of the circles. We generated our database by randomly generating positive examples (where there is a circle in a triangle) and negative examples (where a triangle is in a circle).

We ran both standard COBWEB and REMIND. It could be observed that



a. Remind b. standard Cobweb

Fig. 4. Bongard clusterings

- Cobweb generated clusters with little figures and clusters with many figures (see e.g. figure 4b) with positive and negative examples mixed.
- Remind generated clusters with positive examples and clusters with negative examples (see figure 4a).

When we used the clusterings generated by both systems to predict the class value, the Cobweb clustering scored 59% while the Remind clustering scored 100%.

This result can be intuitively explained as follows. The query $Q_{t1} : triangle(X)$ is more general than $Q_{t2} : triangle(X), triangle(Y), not(X = Y)$. Also, Q_{t2} is more general than the query $Q_{t3} : triangle(X), triangle(Y), not(X = Y), triangle(Z), not(X = Z), not(Y = Z)$. For standard Cobweb, the queries Q_{ti} saying that there should be i different triangles, are all different attributes. Since there are a wide range of possible numbers of triangles, the number of triangles becomes very important for Cobweb. On the other hand, since these queries are dependent on each another, their total weight will be much smaller for Remind. In fact, Remind approximately acts as if there was one (multi-valued) attribute $number\_of\_triangles(N)$.

5.4 Mutagenesis

We also did experiments on the mutagenesis database [20]. In this dataset, an example is a molecule that is either mutagenic or non-mutagenic. There is a relation describing each atom, a relation describing each bond, and a relation describing some global chemical properties of the molecule. The conjunctive queries that can be generated from these relations are massively dependent. We generated all queries with minimal relative frequency of 5% and of maximal length 4.

We ran both standard Cobweb and Remind on a dataset with the truth value of these queries as attributes. We could observe that because of the massive dependencies of attributes, Cobweb produced a very unbalanced tree of 83 levels deep. Remind created more balanced trees.

However, it turned out to be difficult to use these clusterings to predict the mutagenicity of the molecules. This can be caused by the fact that similarity of structure does not necessary mean that molecules are similar in mutagenicity. Another reason is that the simple bayesian network from the query lattice does not cover all dependencies between queries. We did preliminary experiments with more refined versions of the belief network and these give more promising results (67% accuracy to predict mutagenicity). For instance, given queries $Q_1 : atom(X), bond(X,Y)$, $Q_2 : atom(X), bond(X,Y), element(Y,c)$ and $Q_3 : atom(X), bond(X,Y), element(Y,o)$ and the fact that Q_1 succeeds. The fact that Q_2 does not succeed now makes the success of Q_3 more probable (atom Y is of exactly one element type (carbon, hydrogen, oxygen,...)). A more thorough study of this is part of further work.

6 Related work

6.1 Promotion of interesting features

Other work already exists that uses dependencies between attributes to improve clustering. It is interesting to look at the differences. One possibility is described in [21], where the attributes are selected that are most predictive for the other attributes. This means that dependent features are given a larger weight, which is the converse as what we do. This method is used in the hope that irrelevant features will disturb the clustering process less when given a small weight. To understand the difference between this approach and ours, it is necessary to make a distinction between two kinds of dependencies:

- Dependencies caused by e.g. the language bias, which should not be discovered by the clustering system. Our approach is to try to neutralize them.
- Dependencies in the data, unknown to the user, which are hoped to be discovered. It can be interesting to try to promote them.

In fact, it could be useful to try to combine both methods.

6.2 The naive bayes assumption and performance

In the propositional setting, in many cases the naive bayes assumption holds. This means that for classification, it does not hurt much to assume all attributes to be independent. To some extend, a similar assumption can be made for (naive) bayesian clustering. Indeed it does not hurt too much if some attributes accidently happen to be a little dependent. Dependencies unknown by the user could be supposed to occur randomly and to cancel out each another at least partially. As long as intra-cluster correlations are strong enough, the clustering will not be influenced by much. Therefore, we do not expect large performance gains for attribute-value problems without systematic dependencies as described in section 2. Our approach intends to be a functional extension, making clustering possible in domains such as ILP where the language bias causes dependencies. Also, when the user has some prior knowledge, our approach could shift the focus away from this knowledge, in the hope that other unknown relations will be discovered. Since these relations are expected to be weaker, one can in that case even expect a loss in performance measured by the classical metrics. It would be interesting to evaluate the usefulness of this last aspect in a larger iterative knowledge discovery process involving cycles and user interaction.

6.3 Bayesian Classifier

The method discussed in this paper could also be used to develop a "less naive" bayesian classifier that compensates for the violation of the independence assumption of a naive bayesian classifier (see e.g. [13]) by the ILP context.

6.4 Multinets

The probability tables of the basic bayesian networks we use for ilp problems are sparse, In fact, other representations such as bayesian multinets [14] could be used to represent the dependencies in our networks. These representations allow one to represent probability tables with many zeros in a more elegant way. For simplicity, in this paper only the better known standard bayesian net representation was used.

6.5 Distances

The method described in this paper could also be used to make a distance measure that adapts itself to the application domain (by adapting the network structure and weights). Indeed, the utility of taking two examples together in a cluster is a measure of their similarity. This measure is less representation-dependent than other proposed measures (e.g. [19]) and is in many cases faster to compute. We hope that this measure can also perform well in distance-based applications, e.g. for instance based learning. Experiments on this is part of further work.

7 Conclusions and further work

We have proposed the use of the more expressive models of belief networks to describe clusters and presented the changes needed to the objective functions to take this into account. We argued that this has several advantages. First, it allows one more expressivity in the description of clusters. Second, even if no algorithm for the automatic inference of the structure of belief networks can be used, it allows to compensate for dependencies in the data caused by e.g. language bias. Next, we have shown that this can be applied to do clustering on examples in multiple relations using the truth values of non-independent queries. Finally, we did some experiments to evaluate this approach.

The limitations of our system are inherited from the WARMR and COBWEB systems. Level-wise pattern discovery systems such as WARMR can only be used if one is able to define a space of patterns and a lower frequency bound such that the search of frequent patterns is finite. COBWEB is an incremental system, which has both advantages and disadvantages. However, the same approach of using belief networks to describe clusters could be applied to other existing clustering algorithms with different properties.

There are several possible directions for further work. First, one could use an algorithm to learn the optimal structure of the belief networks describing the clusters. This would further increase the expressivity. Second, one could further investigate possible applications of this method (e.g. in databases having taxonomies) and its relation to other clustering aspects such as pruning methods. Third, further research could be done to structure the first order language in such a way that the dependencies between the queries can be found more easily.

Acknowledgements Jan Ramon is supported by the Flemish Institute for the Promotion of Science and Technological Research in Industry (IWT). Luc Dehaspe is supported by K.U.Leuven research grant PDM/98/89.

References

1. R. Agrawal and R. Srikant. Mining generalised association rules. In *Proceedings of the 21th VLDB Conference*, 1995.
2. H. Blockeel and L. De Raedt. Top-down induction of first order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, June 1998.
3. H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In *Proceedings of the 15th International Conference on Machine Learning*, pages 55–63, 1998.
4. P. Cheeseman and J. Stutz. Bayesian classification (autoclass): Theory and results. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. AAAI Press/MIT press, 1996.
5. J. Corter and M. Gluck. Explaining basic categories: feature predictability and information. *psychological bulletin*, (111):291–303, 1992.
6. J. Cussens. Loglinear models for first order probabilistic reasoning. In *Proc. of UAI*, 1999.
7. L. De Raedt. Logical settings for concept learning. *Artificial Intelligence*, 95:187–201, 1997.
8. L. De Raedt and W. Van Laer. Inductive constraint logic. In Klaus P. Jantke, Takeshi Shinohara, and Thomas Zeugmann, editors, *Proceedings of the Sixth International Workshop on Algorithmic Learning Theory*, volume 997 of *Lecture Notes in Artificial Intelligence*, pages 80–94. Springer-Verlag, 1995.
9. L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3(1):7–36, 1999.
10. W. Emde and D. Wettschereck. Relational instance-based learning. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 122–130. Morgan Kaufmann, 1996.
11. D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.
12. D. H. Fisher. Iterative optimization and simplification of hierarchical clusterings. *Journal of Artificial Intelligence Research*, 4:147–179, 1996.

13. P.A. Flach and N. Lachiche. 1bc: a first order bayesian classifier. In D. Page, editor, *Proceedings of the Ninth International Workshop on Inductive Logic Programming*, volume 1634, pages 93–103. Springer-Verlag, 1999.
14. D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and bayesian multinets. *Artificial Intelligence*, 82:45–74, 1996.
15. S. Kramer, B. Pfahringer, and C. Helma. Stochastic propositionalisation of non-determinate background knowledge. In *Proceedings of the Eighth International Conference on Inductive Logic Programming*, volume 1446 of *Lecture Notes in Artificial Intelligence*, pages 80–94. Springer-Verlag, 1998.
16. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241 – 258, 1997.
17. S.-H. Nienhuys-Cheng and R. Wolf. *Foundations of inductive logic programming*, volume 1228 of *Lecture Notes in Computer Science and Lecture Notes in Artificial Intelligence*. Springer-Verlag, New York, NY, USA, 1997.
18. G. Plotkin. A note on inductive generalization. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press, 1970.
19. J. Ramon and M. Bruynooghe. A framework for defining distances between first-order logic objects. In *Proceedings of the Eighth International Conference on Inductive Logic Programming*, Lecture Notes in Artificial Intelligence, pages 271–280. Springer-Verlag, 1998.
20. A. Srinivasan, S.H. Muggleton, R.D. King, and M.J.E. Sternberg. Mutagenesis: ILP experiments in a non-determinate biological domain. In S. Wrobel, editor, *Proceedings of the Fourth International Workshop on Inductive Logic Programming*, volume 237 of *GMD-Studien*, pages 217–232. Gesellschaft für Mathematik und Datenverarbeitung MBH, 1994.
21. L. Talavera. Feature selection as a preprocessing step for hierarchical clustering. In *Proceedings of the 16th International Conference on Machine Learning*, pages 389–397. Morgan Kaufmann, 1999.