Using Logical Decision Trees for Clustering*

Luc De Raedt and Hendrik Blockeel
Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
email: {Luc.DeRaedt,Hendrik.Blockeel}@cs.kuleuven.ac.be
Tel: ++ 32 16 32 76 43 Fax : ++ 32 16 32 79 96

Abstract

A novel first order clustering system, called C 0.5, is presented. It inherits its logical decision tree formalism from the TILDE system, but instead of using class information to guide the search, it employs the principles of instance based learning in order to perform clustering. Various experiments are discussed, which show the promise of the approach.

1 Introduction

A decision tree is usually seen as representing a theory for classification of examples. If the examples are positive and negative examples for one specific concept, then the tree defines these two concepts. One could also say, if there are k classes, that the tree defines k concepts.

Another viewpoint is taken in Langley's *Elements of Machine Learning* [Langley, 1996]. Langley sees decision tree induction as a special case of the induction of *concept hierarchies*. A concept is associated with each node of the tree, and as such the tree represents a kind of taxonomy, a hierarchy of many concepts. This is very similar to what many clustering algorithms do (e.g. COBWEB, [Fisher, 1987]). Indeed, Langley views both techniques as instantiations of the same general technique, namely induction of concept hierarchies.

Concept hierarchies can be induced in a supervised or unsupervised manner. Decision trees are an example of the former, while typical clustering algorithms are unsupervised. Regression trees, as discussed by e.g. Kramer [Kramer, 1996] should be regarded as supervised clustering.

The ILP system TILDE [Blockeel and De Raedt, 1997] induces first order logical decision trees from classified examples. In this paper, we show how to adapt the TILDE system to perform clustering, resulting in the C 0.5 system. To realize this, principles from instance based learning are employed. More specifically, we assume that a distance measure is given that computes the distance between two examples. Furthermore, in order to compute the distance between two clusters (i.e. sets of examples), we employ a function that computes a prototype of a set examples. A prototype can be regarded as an example. The distance between two clusters is then defined as the distance between the prototypes of the two clusters. Now, the distance measure employed determines whether the learning process is supervised or unsupervised. If the distance measure employs class information, learning is supervised, if no class information is employed, learning is unsupervised.

All logical aspects of TILDE are inherited by C 0.5. This includes the logical representation of binary decision trees and the learning from interpretations setting (in which examples are logical interpretations). The use of interpretations to represent examples simplifies the search for an elegant distance measure. In fact, distance metrics known from propositional learning can be used as easily as relational distance metrics. Even if the distance metric is purely propositional, the description of a cluster still makes use of first order logic.

In an early implementation of our framework, the C 0.5. system, only propositional distance metrics are used.

This paper is structured as follows. In Section 2 we discuss the representation of the data and the induced theories. The C 0.5 system is presented in Section 3, and Section 4 identifies possible applications of clustering in ILP. In Section 5 we present some experiments that illustrate some of these applications. Section 6 presents conclusions and related work.

2 The representation

2.1 Learning from interpretations

We use the *learning from interpretations* setting. In this setting, each example is a Prolog program encoding the specific properties of the example. One may also specify background knowledge in the form of a Prolog program.[1]

[1] The interpretation corresponding to each example e is then the minimal Herbrand model of $B \wedge e$.

See [De Raedt and Džeroski, 1994; De Raedt, 1996] for more details on learning from interpretations.

For instance, examples for the well-known mutagenesis problem [Srinivasan *et al.*, 1994] can be described by interpretations. Here, an interpretation is simply an enumeration of all the facts we know about one single molecule: its class, *lumo* and *logp* values, the atoms and bonds occurring in it, certain high-level structures, ... We can represent it e.g. as follows:

```
logmutag(-0.7).
neg.
lumo(-3.025).
logp(2.29).
nitro([d189_12,d189_15,d189_21,d189_22]).
atom(d189_1,c,22,-0.11).
atom(d189_2,c,22,-0.11).
bond(d189_1,d189_2,7).
bond(d189_2,d189_3,7).
...
```

2.2 Logical decision trees

A logical decision trees is a binary decision tree in which each node contains a conjunction of literals. This conjunction may share variables with nodes above it in the tree. The test that is to be performed at the node consists of its conjunction, together with the conjunctions on the path from the root of the tree to this node that share variables with this node or with other nodes sharing variables with it. This test has two possible outcomes (it may fail or succeed), upon which the splitting of a cluster of examples into two subclusters is based.

The format of logical decision trees makes them perfectly fit as a representation for a cluster hierarchy. Each node in the tree corresponds to a cluster of examples, and the hierarchical structure of the tree shows the way that clusters are split into subclusters. The test in a node can be seen as a discriminant description of the two clusters in which the current cluster of examples is divided. One cluster is described by saying that the test succeeds, the other by saying that it fails.

An example of a clustering decision tree, in the mutagenesis context, is show in Figure 1. Note that in a classical logical decision tree leaves would contain classes. Here, leaves simply contain sets of examples that belong together. Variables occurring in tests are existentially quantified. The root test, for instance, tests whether there occurs an atom of type 14 in the molecule.

This view is in correspondence with Langley's viewpoint that a test in a node is not just a decision criterion, but also a description of the subclusters formed in this node. In [Blockeel and De Raedt, 1997], we have shown how a logical decision tree can be transformed into an equivalent logic program. Each node in the tree then corresponds to a unique (propositional) predicate that is defined in the logic program. Furthermore, the predicate corresponding to a node will succeed only for

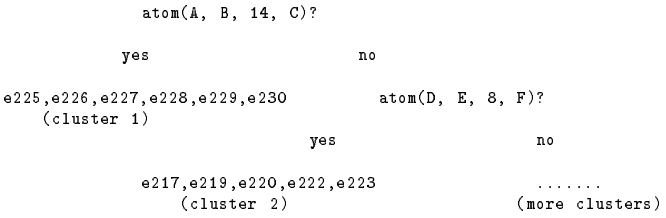

Figure 1: A clustering tree

those examples belonging to the node's cluster. Thus, a logical decision tree can be directly mapped into a discriminant description of the clusters in the tree in the form of a logic program. The logic program thus allows to unambiguously classify the examples into the clusters.

For instance, the tree shown in Figure 1 offers the following discriminant description of cluster 2 :

$$p_1 \leftarrow atom(A, B, 14, C)$$
$$p_2 \leftarrow \neg p1, atom(D, E, 8, F)$$

In words: there does not occur an atom of type 14 in the molecule, but there does occur an atom of type 8.

2.3 Instance Based Learning and Distances

The purpose of conceptual clustering is to obtain clusters such that intra-cluster distance (i.e. the distance between examples belonging to the same cluster) is as small as possible and the inter-cluster distance (i.e. the distance between examples belonging to different clusters) is as large as possible.

In this paper, we assume that a distance measure d that computes the distance $d(e_1, e_2)$ between examples e_1 and e_2 is given. Furthermore, there is also a need for measuring the distance between different clusters (i.e. between sets of examples). Therefore we will assume as well the existence of a prototype function p that computes the prototype $p(E)$ of a set of examples E. The distance between two clusters C_1 and C_2 is then defined as the distance $d(p(C_1), p(C_2))$ between the prototypes of the clusters. This shows that the prototype should be considered as (possibly) partial example descriptions. The prototypes should be sufficiently detailed as to allow the computation of the distances.

For instance, on the mutagenesis problem, the distance could be the Euclidean distance d_1 between the activities of the two compounds, or the Euclidean distance d_2 between the points in the three-dimensional space corresponding to the *lumo*, *logp* and *activity* values of the two compounds, or it could be the distance d_3 as measure by a first order distance measure such as used in RIBL [Emde and Wettschereck, 1996] or KBG [Bisson, 1992a] or [Hutchinson, 1997].

Given the distance at the level of the examples, the principles of instance based learning can be used as to compute the prototypes. E.g. on the mutagenesis problem, d_1 would result in a prototype function p_1 that would simply compute the average activity of the compounds in the cluster, d_2 would result in a prototype function p_2 that would compute the average instance along the *lumo*, *logp* and *activity* values of the compounds in the cluster, whereas d_3 could result in function p_3 that would compute the (possibly reduced) least general generalisation[2] of the compounds in the cluster.

In this preliminary paper on first order clustering we employ only propositional distance measures and the prototype functions that correspond to the instance averaging methods along the lines of [Langley, 1996]. However, we wish to stress that - in principle - we could use any distance measure.

In this respect, it is worth mentioning a potential advantage of learning from interpretations. Because there are no relations between different examples, the definition of first order distances can be simplified. Indeed, some of the complications in the definition of distance metrics that are caused by possible relationships between examples are avoided in this way. For instance, in Bisson's framework [Bisson, 1992b] the distance between two examples may depend on the distance between two other examples to which these are somehow related; but the latter may depend again on the distance of these two examples. This results in a system of equations which may need to be solved iteratively. Also, the distance itself has a quite complex definition.

On the other hand, even within learning from interpretations there may still be relational information within a single example, that needs to be taken into account by first order distance measures. The distance measure can be simplified, but not to the extent of propositional distances.

Notice that although we employ only propositional distance measures, we obtain first order descriptions of the clusters through the representation of logical decision trees. Hence, one could say that we realize 0.5th order Clustering, which explains the name of our system C 0.5.

2.4 Problem-specification

By now we are able to formally specify the clustering problem:

Given

- a set of examples E (each example is a definite clause theory),
- a background theory B in the form of a definite clause theory,
- a distance measure d that computes the distance between two examples or prototypes,
- a prototype function p that computes the prototype of a set of examples,

Find: clusters in the form of a logical decision tree.

Notice that in this problem-setting, the interpretation i corresponding to an example e is the least Herbrand model of $B \wedge e$. So, conceptually, the clustering process learns from interpretations. This is similar to the Claudien [De Raedt and Dehaspe, 1997], ICL [De Raedt and Van Laer, 1995] and TILDE [Blockeel and De Raedt, 1997] systems.

3 C 0.5 : Clustering of order 0.5

3.1 The language of logical decision trees

C 0.5 employs the basic TDIDT framework as it is also incorporated in TILDE. The only point where our algorithms C 0.5 and TILDE differ from the propositional TDIDT algorithm is in the computation of the tests to be placed in a node. To this aim, we employ a classical refinement operator under θ-subsumption [Plotkin, 1970; Muggleton and De Raedt, 1994]. A clause c_1 θ-subsumes another clause c_2 if and only if there is a variable substitution θ such that $c_1\theta \subseteq c_2$. A refinement operator under θ-subsumption is an operator ρ mapping clauses onto sets of clauses, such that for any clause c and $\forall c' \in \rho(c)$, c θ-subsumes c'. The operator could, for instance, add literals to the clause, or unify several variables in it.

For our decision tree induction algorithms, we will use a refinement operator that adds one or more literals to a clause. When a node N is to be refined, the set of all the conjunctions that can possibly be put in that node is computed as

$$\{C | (\leftarrow Q, C) \in \rho(\leftarrow Q)\}$$

where Q is the query associated with the node, as defined in [Blockeel and De Raedt, 1997]. Roughly speaking, the query associated with a node is the set of accumulated conditions that hold on the path from root to the present node. These queries are also the basis of the transformation from decision trees to logical theories, this is formally worked out in [Blockeel and De Raedt, 1997].

Declarations defining the refinement operator ρ are an input parameter of the algorithm. Since ρ determines the language bias, a separate language bias specification is not needed. When we refer to the language bias specification, we mean the definition of ρ.

The implementation of C 0.5 and TILDE assumes that the predicate *rmode* is available that indicates which

[2] Using Plotkin's [Plotkin, 1970] notion of θ-subsumption or the variants corresponding to structural matching [Bisson, 1992a; Raedt *et al.*, 1997].

conjunctions are eligible for addition to a query (this is similar to mode declarations as in Muggleton's Progol [Muggleton, 1995]).

For instance, `rmode(8:(p(+X,-Y, Z),q(Z)))` specifies that the conjunction $p(X, Y, Z), q(Z)$ can occur maximally 8 times in the associated query of a node, and allows possible unifications of the variables of p with other variables in the query (`+X`, `-Y` and `Z` indicating that X must be unified with an existing variable, Y may but need not be unified, and Z is a new variable). A node with associated query $\leftarrow a(A), b(B, C)$ can then be refined in the following ways: a literal p can be added with first argument A, B or C, second argument A,B,C or a new variable Y, the third argument Z, and the literal q is simultaneously added with argument Z. This yields 12 possible refinements (or less, if type restrictions apply).

In addition to this, so-called lookahead specifications can be provided. These allow TILDE and C 0.5 to perform several successive refinement steps at once. This alleviates the well-known problem in ILP that a refinement may not yield any gain, but may introduce new variables that are crucial for classification. By performing successive refinement steps at once, TILDE can look ahead in the refinement lattice and discover such situations.

For instance, `lookahead((p(X,Y,Z),q(Z)), r(X))` specifies that whenever the conjunction containing p and q is added, additional refinement by adding $r(X)$ (with X the first variable in p) is possible in the same refinement step.

Finally, C 0.5 also inherits a discretization procedure from TILDE and ICL (see [Van Laer *et al.*, 1996] for a description). This discretization procedure makes it possible to derive theories that can use numerical information.

3.2 The heuristics

The other point where C 0.5 deviates from the classical TDIDT algorithm is in the heuristics used. TDIDT algorithms typically use two heuristics: a splitting criterion to select the best test in a node, and a stopping criterion to decide whether a given node or cluster should be turned into a leaf or should be split further.

The splitting criterion used in C 0.5 works as follows. For a given a cluster C, and a test T that will result in two disjoint subclusters C_1 and C_2 of C, C 0.5 computes the distance $d(p(C_1), p(C_2))$. The best test T is then the one that maximizes this distance. This reflects the principle that the inter-cluster distance should be as large as possible.

Stopping criteria for conceptual clustering are usually less clear. Typically, the stopping criterion depends on the number and size of clusters one wants to obtain. In C 0.5, we consider two criteria. The first criterion is the simplest (and is used in the present experiments), and requires that each leaf of the tree should contain a minimal number of examples. The second criterion is derived from classical principles of clustering as formulated by [Michalski, 1987]. It states that subclusters C_1 and C_2 of cluster C are not allowed if

$$w \times \sum_{e \in C} d(e, p(C)) \leq \sum_{e_1 \in C_1} d(e_1, p(C_1)) + \sum_{e_2 \in C_2} d(e_2, p(C_2))$$

where w is weight between 0 and 1. This stopping criterion reflects the principle that each clustering step should result in a decrease of the intra-class distance. The weight w can then be used to influence the degree of the required decrease. The larger w is, the more clusters will be obtained.

4 Applications of Clustering in ILP

We see a number of interesting applications of clustering in ILP. We will divide these into characterisation, classification and regression tasks. But first, we make an observation that will be important to all these tasks.

Once a decision tree has been induced, there can be two separate arguments for saying that an example belongs to a cluster. One is based on the tests in the tree. If an example is sorted into a specific node based on those tests, one has reason to say that the example really belongs to the cluster corresponding to that node. A second argument is based on the distance metric. Without performing any tests in the tree, one can still say that an example belongs to a cluster because it is close to the examples in that cluster. Both arguments are quite independent from one another (cf. also [Kramer, 1996]).

This means that information can flow in two opposite directions. One can assign an example to a cluster based on the tree, and predict that its distance to the examples in that cluster will be small; or one can assign it to a cluster based on its distance to the examples in the cluster, and predict that it will satisfy the description of that cluster that the tree gives. While the first information flow has a flavour of classification, the second comes closer to characterisation (although the difference between the two is not always very clear-cut).

4.1 Characterisation of clusters

The fact that logical decision trees offer a first order description of clusters makes it possible to perform a kind of abduction. Based on the fact that an example belongs to a specific cluster (according to the distance metric that is used), one can predict that it fulfills the intensional description of that cluster.

For instance, consider the propositional example shown in Figure 2. If an unseen example e_{10} is close

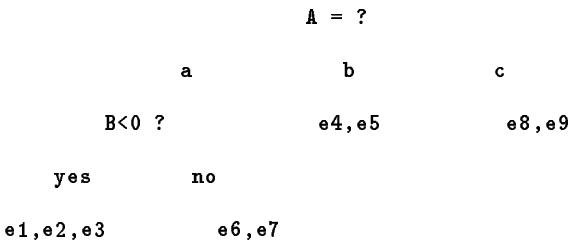

Figure 2: A propositional clustering tree

to e_6 and e_7, then one would expect it to have the following properties: $A = a$ and $B \geq 0$.

If the distance measure is based on certain easily observable properties of the example, then unobservable properties can be predicted. For instance, based on the specific value of the mutagenicity of a molecule, one could predict that a certain structure occurs in the molecule (e.g. a benzene ring).

Of course, the intensional description offered by the decision tree itself is very limited. The tree typically gives the smallest possible description of clusters that allows to discriminate between them. From the point of view of abduction, this is not very interesting.

However, a much richer form of prediction is possible if one applies a discovery system such as Claudien [De Raedt and Dehaspe, 1997] to each of the clusters. One then obtains a maximally specific description characterizing each cluster. As pointed out in [De Raedt and Dehaspe, 1997] such theories can be used for predicting missing information about an example. For instance, in the above example, a clausal discovery system, might induce for the cluster e_1, e_2, e_3 the following partial theory :

$$C = d \leftarrow B < -5$$

$$C = e \leftarrow B \geq -5$$

In this case, if we would know that an example e_{11} has value -3 for B, and the example would be classified in this cluster (either using the distance or using the logical decision tree), we would predict that the value of C would be e. Though the example is only propositional, the argument also holds for first order logic.

4.2 Classification

It is quite clear that first order clustering can be used for classification; in fact, the original TILDE system is such a classification system. Following Langley's viewpoint, it is basically a clustering system where the "distance" metric is the class entropy within the clusters : lower class entropy within a cluster means that the examples in that cluster are more similar with respect to their classes.

Since TILDE needs information about the classes of its training examples, it can be called a supervised learning system. In general, however, clustering can also be done in an unsupervised manner. When making use of a distance metric to form clusters, this distance metric may or may not use information about the classes of the examples. Even if it does not use class information, clusters may be coherent with respect to the class of the examples in them. This will be illustrated in our experiments.

This principle leads to a classification technique that is very robust with respect to missing class information. Indeed, even if only a small percentage of the examples is labelled with a class, one could perform unsupervised clustering, and assign to each leaf in the concept hierarchy the majority class in that leaf. If the leaves are coherent with respect to classes, this method would yield relatively high classification accuracy with a minimum of class information available. This is quite similar in spirit to Emde's method for learning from few classified examples, implemented in the COLA system [Emde, 1994].

4.3 Regression

The above shows that first order clustering can be used for characterisation of clusters in the data, as well as for classification. An application that has a flavour of both, is predicting numerical values. If clusters are coherent with respect to some numerical attribute of the examples, one can compute the average value for a cluster of examples and use this to predict the value of unseen examples. This is basically what Kramer's SRT system [Kramer, 1996] does. SRT builds so-called structural regression trees: these are decision trees in which each leaf predicts a numerical value instead of a symbolic class. The "structural" refers to the fact that the tests put in nodes are conjunctions of literals. With respect to the representation of the induced theory, SRT is very close to TILDE and C 0.5.

The heuristic that SRT uses to decide which test should be put in a node, is minimisation of the sum of squared errors of the values within subclusters with respect to the average value in the subcluster. This means that euclidean distance within clusters is minimised. This euclidean distance is based on the numerical values that are to be predicted; therefore this is a supervised clustering method.

The clustering method we propose is more general than Kramer's SRT system in two ways. First of all, one is not restricted to the prediction of one single attribute. Given a cluster, it is possible to predict any numerical or symbolic value for an unseen example by predicting the average or most frequent value. The induction of regression trees could easily be extended to predict vectors of values instead of single values though.

A second difference is that the distance metric is totally independent from the numerical attributes that will be predicted. One can use only these attributes in the distance metric (leading to classical regression), mix them with other attributes, or not use them at all (unsupervised clustering). We believe that this is a more important difference with classical regression techniques.

5 Experiments

We have evaluated some of the ideas proposed in this paper using a preliminary implementation of C 0.5.

Concerning the evaluation of a clustering system, there is the fundamental problem of how to evaluate the quality of a set of clusters as there is no clear and objective criterion to measure this quality. The situation is therefore different from that of concept-learning, where the quality of the output of a system is measured in terms of accuracy. Therefore, we cannot directly evaluate the result of the clustering task. Instead, as frequently done in conceptual clustering (cf. [Fisher, 1987]), we will measure the quality of the obtained clusters using the accuracy with which these clusters can be used for prediction. Because of this, when reading this section, the reader might get the impression that C 0.5 is a classification system. This impression is false: C 0.5 is a clustering system that can also be used for classification. So, classification is only used to demonstrate that the clusters output by C 0.5 are of high quality.

Despite the fact that we use a real-life data set (the mutagenesis data), we doubt that all our experiments are useful from the application or chemical point of view. The experiments are only meant to illustrate the type of tasks that C 0.5 could be useful for and this on a sufficiently complex data-set. They are not meant to contribute to the chemical knowledge on mutagenicity, though the results of some of the experiments may give more insight into the mutagenesis problem to inductive logic programming researchers.

5.1 Experimental setup

Originally, mutagenicity was represented by a number on a logarithmic scale. For prediction purposes, a molecule is considered to be of class *positive* if this number is positive, and is of class *negative* otherwise. In all experiments we use euclidean distance metrics based on the numerical information about molecules, i.e. their *logp*, *lumo* and *logm* (which stands for *log-mutagenicity*, the number describing the activity of the molecule) values. Several distance metrics have been tried out. First of all, supervised learning (using *logm* for distance computation) was compared to unsupervised learning (using *logp* and *lumo*). This was done for three different hypothesis languages corresponding to the background knowledges BG1-BG3 as defined in [Srinivasan *et al.*, 1995]. BG1 contains only structural information (atoms and bonds), BG2 adds to this the charges of each individual atom, BG3 adds to BG2 the *lumo* and *logp* values of each molecule.

We want to stress that these different background knowledges are background knowledge with respect to the theory that will be built; i.e. BG1 means that the theory will not make use of *lumo* or *logp*. However, during the induction process itself these values are available, in the case of unsupervised learning; they are used to compute distances. In fact, it seems that we need to distinguish between background knowledge that can be used during the induction process, and background knowledge that will be available at the time of classification of unseen examples. One could say that in classical ILP systems language bias defines the latter, while background knowledge refers to the former; but this is not really true, as class information is usually not considered to be background knowledge. The point is that, with respect to evaluation of the quality of a refinement, other information than only the class information may be useful.

The mutagenesis dataset has been split into regression-friendly and regression-unfriendly data. We have performed experiments on both sets separately. Since for the regression-unfriendly data *lumo* and *logp* seem to be bad indicators for mutagenicity, one would expect the unsupervised method (where clustering is based on *lumo* and *logp*) to work less well than the supervised method on this dataset.

In order to evaluate the results of the clustering algorithm, we have computed accuracies as follows. For a whole cluster the *logm* value of the prototypical element is computed. If this is a positive number, then the whole cluster is predicted to be positive, otherwise it's predicted to be negative. This results in a *pos* or *neg* classification for each element, on the basis of which classification accuracy can be computed. Classification accuracy on the regression-friendly set is based on tenfold crossvalidation, for the regression-unfriendly set a leave-one-out procedure was adopted.

5.2 Classification Accuracy

In a first experiment we compare supervised and unsupervised clustering, on both the regression-friendly and regression-unfriendly datasets. Table 1 shows the results of our experiments. The results for supervised learning can be compared with results obtained with other systems (Progol, TILDE), but for unsupervised learning this comparison is harder in the case of BG1 and BG2, since C 0.5 does use *logp* and *lumo* information during the induction process, even if the theories that are derived do not use them. For the regression-friendly data we have included results obtained with the TILDE system. The refinement operator and other parameters were ex-

actly the same in C 0.5 as in TILDE, only the minimal coverage of clusters was higher (10 examples for the regression-friendly data, 4 for the regression-unfriendly data) because C 0.5 does not have a good stopping criterion yet. We have not tried to find an optimal value for the minimal coverage, but consistently used 10 (or 4).

Table 1 shows that the results of both supervised and unsupervised learning are comparable with results for TILDE (with the exception of BG3 where a significantly lower score is obtained). Especially for unsupervised learning this is sort of a surprise. It shows that *lumo* and *logp* are very closely related to the class; examples of which the *lumo* and *logp* values are similar will almost always have the same class. This explains in part why FOIL, on the Mutagenesis dataset with BG3, finds a theory simply consisting of a large number of intervals for *logp* and *lumo* (see [Srinivasan *et al.*, 1995]).

In order to check this out in more detail, we have additionally experimented (on BG2) with distances based on *logp* only, on *lumo* only, and on the three numerical values together. Results are shown in Table 2 (first row). There are no significant differences for the different distance functions.

The results on the regression-unfriendly set (Table 1) are much harder to explain. The expected behaviour (less accuracy for unsupervised than for supervised learning) occurs clearly in BG3, but not in BG1 and BG2.[3] We do not have an explanation for this.

5.3 Learning From Incomplete Examples

Next to comparing different distance functions, Table 2 also shows what happens if numerical information is missing. The dataset was changed so that the *lumo* information was removed in approximately 50% of the models, chosen at random (every model has probability 0.5 of having this value removed), and the same was done independently for *logp* and *logm* values.

The same method has been repeated with only 25% of the models retaining information about a numerical attribute, and finally with 10%. It can be seen in Table 2 that in the absence of a lot of information, classification accuracy slowly decreases. However, when using a distance measure that combines several numerical attributes, prototypes can be computed more accurately than if only one numerical attribute is used. Absence of information about one attribute can be compensated for by knowledge of another value. This causes the distance function using all three attributes (*lumo, logp, logm*) to be very robust w.r.t. missing information; predictive accuracy decreases more slowly than for the other distance functions.

[3]The hypothesis that there is no difference in classification accuracy between the two methods (in BG3) is rejected at the 5% level.

	Regression friendly			Regression unfriendly		
	unsup.	sup.	TILDE	unsup.	sup.	TILDE
BG1	0.73	0.74	0.75	0.76	0.76	0.83
BG2	0.81	0.80	0.79	0.74	0.76	0.83
BG3	0.79	0.78	0.85	0.71	0.86	0.79

Table 1: Comparison of supervised and unsupervised clustering on both regression-friendly and unfriendly data; results with TILDE included for reference.

available numerical data	lumo	logp	lumo+logp	logm	all three
100%	0.79	0.79	0.81	0.80	0.81
50%	0.77	0.79	0.81	0.78	0.79
25%	0.73	0.72	0.73	0.72	0.77
10%	0.65	0.65	0.67	0.67	0.74

Table 2: Classification accuracies obtained on BG2 with several distance functions, and on several levels of missing information.

These results suggest that even if more complicated distance functions do not yield higher accuracy when all information is present, they make learning from incomplete examples more feasible.

6 Conclusions and Related Work

We have presented a novel first order clustering system C 0.5 within the TDIDT class of algorithms. C 0.5 integrates ideas from concept-learning (TDIDT), from instance based learning (the distances and the prototypes), and from inductive logic programming (the representations) to obtain a clustering system. Several experiments were performed on a real world data set that illustrate the type of tasks C 0.5 is useful for. Further experiments should be performed in order to explore the other types of tasks sketched (e.g. predicting with a Claudien type of characteristic description of clusters), other domains, and also to explore first order distance and prototype functions.

As far as related work is concerned, our work is related to KBG by [Bisson, 1992a], which also performs first order clustering. In contrast to the current version of C 0.5., KBG does use a first order similarity measure, which could also be used within C 0.5. Furthermore, KBG is an agglomerative (bottom-up) clustering algorithm and C 0.5 a divisive one (top-down). The divisive nature of C 0.5 makes C 0.5 as efficient as classical TDIDT algorithms. A final difference with KBG is that C 0.5 directly obtains logical descriptions of the difference clusters through the use of the logical decision tree format. For KBG, these descriptions have to be derived in a separate step because the clustering process only produces the clusters (i.e. sets of examples) and not their description.

With respect to first order distance functions, we want to mention the instance-based learner RIBL [Emde and Wettschereck, 1996]. This system uses an advanced first order distance metric that might be a good candidate for incorporation in C 0.5.

Our work also exploits many of the ideas contained in Langley's book, as he makes the link between TDIDT and clustering. From this point of view, our work is closely related to Kramer's SRT, who builds regression trees in a supervised manner. From the discussion in the experimental section, it should be clear that C 0.5. can be considered a generalization of Kramer's SRT in that C 0.5 can also build trees in an unsupervised manner, and that Kramer's distance measure is based on a single attribute. Finally, we should also refer to a number of other approaches to first order clustering, which include Kietz's and Morik's Kluster [Kietz and Morik, 1994], [Thompson and Langley, 1991] and [Ketterlin *et al.*, 1995].

7 Acknowledgements

Luc De Raedt is supported by the Fund for Scientific Research, Flanders. Hendrik Blockeel is supported by the Flemish Institute for the Promotion of Scientific and Technological Research in the Industry (IWT). This work is also part of the European Community Esprit project no. 20237, Inductive Logic Programming 2. The authors wish to thank Luc Dehaspe, Wim Van Laer and Nico Jacobs for proofreading this text.

References

[Bisson, 1992a] G. Bisson. Conceptual clustering in a first order logic representation. In *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 458–462. John Wiley & Sons, 1992.

[Bisson, 1992b] G. Bisson. Learning in fol with a similarity measure. In *Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-92)*. AAAI Press, 1992.

[Blockeel and De Raedt, 1997] H. Blockeel and L. De Raedt. Experiments with top-down induction of logical decision trees. Technical Report CW 247, Dept. of Computer Science, K.U.Leuven, January 1997. Also in Periodic Progress Report ESPRIT Project ILP2, January 1997.

[De Raedt and Dehaspe, 1997] L. De Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 26:99–146, 1997.

[De Raedt and Džeroski, 1994] L. De Raedt and S. Džeroski. First order *jk*-clausal theories are PAC-learnable. *Artificial Intelligence*, 70:375–392, 1994.

[De Raedt and Van Laer, 1995] L. De Raedt and W. Van Laer. Inductive constraint logic. In *Proceedings of the 5th Workshop on Algorithmic Learning Theory*, volume 997 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1995.

[De Raedt, 1996] L. De Raedt. Induction in logic. In R.S. Michalski and Wnek J., editors, *Proceedings of the 3rd International Workshop on Multistrategy Learning*, pages 29–38, 1996.

[Emde and Wettschereck, 1996] W. Emde and D. Wettschereck. Relational instance-based learning. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning*, pages 122–130. Morgan Kaufmann, 1996.

[Emde, 1994] W. Emde. Inductive learning of characteristic concept descriptions. In S. Wrobel, editor, *Proceedings of the 4th International Workshop on Inductive Logic Programming*, volume 237 of *GMD-Studien*, pages 51–70, Sankt Augustin, Germany, 1994. Gesellschaft für Mathematik und Datenverarbeitung MBH.

[Fisher, 1987] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[Hutchinson, 1997] A. Hutchinson. Metrics on terms and clauses. In *Proceedings of the 9th European Conference on Machine Learning*, 1997.

[Ketterlin *et al.*, 1995] A. Ketterlin, P. Gancarski, and J.J. Korczak. Conceptual clustering in structured databases : a practical approach. In *Proceedings of KDD-95*, 1995.

[Kietz and Morik, 1994] J.U. Kietz and K.. Morik. A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, 14:193–217, 1994.

[Kramer, 1996] S. Kramer. Structural regression trees. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*, 1996.

[Langley, 1996] P. Langley. *Elements of Machine Learning*. Morgan Kaufmann, 1996.

[Michalski, 1987] R.S. Michalski. Clustering. In S. Shapiro, editor, *Encyclopedia of artificial intelligence*. Wiley, 1987.

[Muggleton and De Raedt, 1994] S. Muggleton and L. De Raedt. Inductive logic programming : Theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.

[Muggleton, 1995] S. Muggleton. Inverse entailment and progol. *New Generation Computing*, 13, 1995.

[Plotkin, 1970] G. Plotkin. A note on inductive generalization. In *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press, 1970.

[Raedt *et al.*, 1997] L. De Raedt, P. Idestam-Almquist, and G. Sablon. Theta-subsumption for structural matching. In *Proceedings of the 9th European Conference on Machine Learning*, 1997.

[Srinivasan *et al.*, 1994] 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 4th International Workshop on Inductive Logic Programming*, volume 237 of *GMD-Studien*, pages 217–232. Gesellschaft für Mathematik und Datenverarbeitung MBH, 1994.

[Srinivasan *et al.*, 1995] A. Srinivasan, S.H. Muggleton, and R.D. King. Comparing the use of background knowledge by inductive logic programming systems. In L. De Raedt, editor, *Proceedings of the 5th International Workshop on Inductive Logic Programming*, 1995.

[Thompson and Langley, 1991] K. Thompson and P. Langley. Concept formation in structured domains. In D. Fisher, M. Pazzani, and P. Langley, editors, *Concept formation: knowledge and experience in unsupervised learning*. Morgan Kaufmann, 1991.

[Van Laer *et al.*, 1996] W. Van Laer, S. Džeroski, and L. De Raedt. Multi-class problems and discretization in ICL (extended abstract). In *Proceedings of the ML-net Familiarization Workshop on Data Mining with Inductive Logic Programming (ILP for KDD)*, 1996.