# Remembering similitude terms in CBR

Eva Armengol and Enric Plaza

IIIA - Artificial Intelligence Research Institute,
CSIC - Spanish Council for Scientific Research,
Campus UAB, 08193 Bellaterra, Catalonia (Spain).
{eva, enric}@iiia.csic.es,

**Abstract.** In concept learning, inductive techniques perform a global approximation to the target concept. Instead, lazy learning techniques use local approximations to form an implicit global approximation of the target concept. In this paper we present C-LID, a lazy learning technique that uses LID for generating local approximations to the target concept. LID generates local approximations in the form of similitude terms (symbolic descriptions of what is shared by 2 or more cases). C-LID caches and reuses the similitude terms generated in past cases to improve the problem solving of future problems. The outcome of C-LID (and LID) is assessed with experiments on the Toxicology dataset.

## 1 Introduction

Concept learning can be achieved both using lazy learning and eager learning. Inductive concept learning is the typical eager approach, where learning is finding a description characterizing the instances of a concept (positive examples) and not the rest (negative examples). Lazy learning techniques like $k$-nearest neighbor and case-based reasoning define concepts extensionally, i.e. by using the extension of a concept enumerating the instances that belong to the concept (positive examples) and those that do not (negative examples). Moreover, lazy learning techniques are *problem-centered*, i.e. they take into account the new problem (sometimes called *query*) while using the information of the previous instances (*cases*). Inductive techniques are unable to do so, since by the time they observe the query they have already chosen their (global) approximation to the target function. In other words, eager techniques take a global approach to concept learning while lazy techniques (implicitly) represent the target function by combining many *local approximations* [6].

In this paper we present C-LID a lazy learning approach that reuses those *local approximations* used for solving past instances in order to improve the classification of new problems in case based-reasoning (CBR). C-LID (Caching LID) is a variant of the CBR technique LID (Lazy Induction of Descriptions) [3]. LID is a lazy concept learning technique for classification tasks in CBR based on the notion of *similitude term*. CBR approaches are based on finding the most similar (or relevant) cases for a particular problem $p$, and usually these degree of similarity is assessed using a distance metric. However, as explained in section

2, LID does not use a distance. Instead, LID builds a symbolic description of the similarity $D_p$ between $p$ and a set of cases $S_{D_p}$. We call $D_p$ a *similitude term* and contains that which is common to $p$ and the cases in $S_{D_p}$. In other words, $D_p$ is a generalization such that it covers the problem ($D_p \sqsubseteq p$) and the *retrieved cases* ($D_p \sqsubseteq c : \forall c \in S_{D_p}$). Thus, for LID, the similitude terms are the *local approximations* used for solving problems, and since they are symbolic descriptions we are able to cache them and reuse them in solving future problems. This is the approach of *Caching* LID presented in this paper. In order to define C-LID on top of LID we have only to specify two policies: a) the *caching policy* (i.e. which similitude terms are cached and which not), and b) the *reuse policy* (i.e. when and how the cached terms are used to solve a new problem) — see §3 below. Notice, however, that C-LID is still a lazy learning technique according to the definition given above. The generalizations created by LID and cached by C-LID are not global: they are local approximations.

The structure of the paper is as follows: section 2 explains the Lazy Induction of Descriptions technique, then section 3 introduces C-LID while section 4 presents an application of C-LID to a toxicology data set and reports on the accuracy results of both LID and C-LID. The paper closes with a discussion section.

## 2 Lazy Induction of Descriptions

In this section we explain the LID (*Lazy Induction of Descriptions*) method. The goal of LID is to classify a problem as belonging to one of the solution classes. The main idea of LID is to determine which are the more relevant features of the problem and to search in the case base for cases sharing these relevant features. The problem is classified when LID finds a set of relevant features shared by a subset of cases belonging all of them to the same solution class. Then, the problem is classified into that solution class.

Given a case base $B$ containing cases classified into one of the solution classes $C = \{C_1 \dots C_m\}$ and a problem $p$, the goal of LID is to classify $p$ as belonging to one of the solution classes. The problem and the cases in the case base are represented as feature terms.

*Feature terms* (also called *feature structures* or *$\psi$-terms*) are a generalization of first order terms. The intuition behind a feature term is that it can be described as a labelled graph. The edges of the graph are labelled with feature symbols and the nodes are the sorts of the feature values. Sorts have an informational order relation ($\preceq$) among them, where $\psi \preceq \psi'$ means that $\psi$ has less information than $\psi'$ or equivalently that $\psi$ is more general than $\psi'$. The minimal element ($\bot$) is called *any* and it represents the minimum information. When a feature has unknown value it is represented as having the value *any*. All other sorts are more specific that *any*. Figure 2 shows an example of sort/subsort hierarchy.

The semantic interpretation of feature terms brings an ordering relation among feature terms that we call *subsumption*. Intuitively, a feature term $\psi$ subsumes another feature term $\psi'$ ($\psi \sqsubseteq \psi'$) when all the information in $\psi$ is also

```
Function LID (S_D, p, D, C)
    if stopping-condition(S_D)
        then return class(S_D)
        else f_d := Select-leaf (p, S_D, C)
             D' := Add-path(π(root(p), f_d), D)
             S_D' := Discriminatory-set (D', S_D)
             LID (S_D', p, D', C)
    end-if
end-function
```

**Fig. 1.** The LID algorithm. $D$ is the similitude term, $S_D$ is the discriminatory set of $D$, $C$ is the set of solution classes, $class(S_D)$ is the class $C_i \in C$ to which all elements in $S_D$ belong.

contained in $\psi'$. In section 4.1 feature terms are explained with an example. For a more formal explanation of feature terms see [1].

We define the *similitude term*, $s$, of two cases $c_1$ and $c_2$ as a term such as $s \sqsubseteq c_1$ and $s \sqsubseteq c_2$ i.e. the similitude term of two cases subsumes both cases. In this framework, the task of similarity assessment is a search process over the space of similarity descriptions determined by the subsumption relation.

We call *discriminatory set* the set $S_D = \{b \in B | D \sqsubseteq b\}$ that contains the cases of B subsumed by the similitude term $D$.

The LID algorithm (figure 1) begins with the similitude term $D$ initialized to the most general feature term (i.e. the feature term *any* that has no features) and the discriminatory set $S_D$ initialized to the whole case base $B$ (since *any* subsumes all the cases). When there is some domain knowledge, the similitude term $D$ can be initialized to a value $D^0$ (where $D^0 \neq any$) as is described in [2].

Given the current similitude term $D$, the stopping condition of LID is that all the cases in $S_D$ belong to one solution class $C_k \in C$. In the first call, this condition is not satisfied because the initial similitude term $D$ subsumes the whole case base. The next step is to select a leaf for specializing $D$.

The specialization of a similitude term $D$ is achieved by adding features to it. In principle, any of the features used to describe the cases could be a good candidate. Nevertheless, LID uses two biases to obtain the set $F_l$ of features candidate to specialize $D$. First, of all possible features describing a case, LID will consider only those features present in the problem $p$ to be classified. As a consequence, any feature that is not present in $p$ will not be considered as candidate to specialize $D$. The second bias is to consider as candidates for specializing $D$ only those features that are leaf features of $p$ (i.e. to features having as values feature terms without features).

The next step of LID is the selection of a leaf feature $f_d \in F_l$ to specialize the similitude term $D$. Selecting the most discriminatory leaf feature in the set $F_l$ is heuristically done using the RLM distance [5] over the features in $F_l$. The RLM distance assesses how similar are two partitions in the sense that the lesser

the RLM distance is the more similar are the partitions. Let us suppose that the feature $f_i$ takes as value $v_i$ in the problem $p$. This feature induces a partition $P_i$ of the case-base formed by two sets: one containing the cases that have value $v_i$ and the other contain those cases with value different than $v_i$ in the feature $f_i$. For each feature in $F_l$, LID induces its associated partition.

The *correct partition* is a partition $P_c = \{C_1 \dots C_m\}$ where all the cases contained into a set $C_i$ belong to the same solution class. For each partition $P_i$ induced by a feature $f_i$, LID computes the RLM distance to the correct partition $P_c$. The proximity to $P_c$ of a partition $P_i$ estimates the relevance of feature $f_i$.

Let $P_i$ and $P_j$ the partitions induced by features $f_i$ and $f_j$ respectively. We say that the feature $f_i$ is *more discriminatory than* the feature $f_j$ iff $RLM(P_i, P_c) < RLM(P_j, P_c)$, i.e. when the partition induced by $f_i$ is closer to the correct partition $P_c$ than the partition induced by $f_j$. Intuitively, the most discriminatory feature classifies the cases in a more similar way to the correct classification. LID uses the *more discriminatory than* relationship to estimate the features that are more relevant for the purpose of classifying a current problem.

Let us call $f_d$ the most discriminatory feature in $F_l$. The feature $f_d$ is the leaf feature of path $\pi(root(p), f_d)$ in problem $p$. The specialization step of LID defines a new similitude term $D'$ by adding to the current similitude term $D$ the sequence of features specified by $\pi(root(p), f_d)$. After this addition $D'$ has a new path $\pi(root(D'), f_d)$ with all the features in the path taking the same value that they take in $p$. After adding the path $\pi$ to $D$, the new similitude term $D' = D + \pi$ subsumes a subset of cases in $S_D$, namely the discriminatory set $S_{D'}$ (the subset of cases subsumed by $D'$).

Next, LID is recursively called with the discriminatory set $S_{D'}$ and the similitude term $D'$. The recursive call of LID has $S_{D'}$ as first parameter (instead of $S_D$) because the cases that are not subsumed by $D'$ will not be subsumed by any further specialization. The process of specialization reduces the discriminatory set $S_D^n \subseteq S_D^{n-1} \subseteq \dots \subseteq S_D^0$ at each step.

Another stopping situation is when the current discriminatory set $S_D^n$ contains cases belonging to several solution classes $(C_i \dots C_j)$ but the similitude term $D^n$ cannot be further specialized. In this situation LID uses the majority rule for propose a solution class for $p$, i.e. $p$ is classified as belonging to the class $C_k$ such that $Card(S_D^n \cap C_k) = max\{Card(S_D^n \cap C_i) \dots Card(S_D^n \cap C_j)\}$.

Given a new problem $p$ and a case base $CB$, the result of LID is the solution class and a similitude term $D$. The similitude term can be seen as an explanation of why $p$ is classified as belonging to a solution class $C_i$. Moreover, the cases contained in the discriminatory set $S_D$ support this classification. Notice that the stopping condition means that the similitude term is able to discriminate cases belonging to the solution class $C_i$ with respect to the cases that does not belong to $C_i$. In this sense, the similitude term $D$ can be viewed as a *partial* description of $C_i$. $D$ is a partial description because, in general, it does not subsume all the cases belonging to $C_i$ but only a subset of them (those sharing the features of $D$ with the new problem). The similitude term $D$ depends on the new problem, for this reason there are several partial descriptions (i.e. similitude terms) for the

same solution class. In the next section we explain how the similitude term can be used as support to the LID solution.

## 3 Caching LID

Caching LID is implemented on top of LID by defining two policies: the caching policy and the reuse policy. C-LID considers the similitude terms built by LID as descriptions of the local approximations performed by this CBR technique. The *caching policy* determines which similitude terms are to be retained; from now one we will call *patterns* the similitude terms that are cached by C-LID. The *reuse policy* determines when and how the cached patterns are used to solve new problems.

The caching policy of C-LID states that similitude term $D$ will be cached if it is *univocal*, i.e. when all cases covered $D_p$ (all cases in $S_{D_p}$) belong to one class only. Thus, C-LID's caching policy retains only similitude terms that characterize the class of a problem $p$ and a set or retrieved cases $S_{D_p}$ without any uncertainty. The *reuse policy* of C-LID states that patterns will be used for solving a problem $p$ only when LID is unable to find a similitude term $D_p$ for $p$ that is *univocal*. In that case all patterns that subsume $p$ will be used together with $D_p$ to determine the most likely class of $p$. Below we explain in detail how this class is determined.

C-LID can be decomposed in two steps: 1) a preprocessing of the case base in order to obtain some similitude terms to be cached; and 2) the problem solving phase that uses LID together with the cached similitude terms for clasifying new problems.

*Preprocessing.* During this phase the caching policy is applied. The experiments described in section 4.2 are performed using the caching policy already explained. There are possible other less strict caching policies, and we briefly discuss them in section 5. The preprocessing phase is done using the leave-one-out technique over the whole training set $B$. For each case $c \in B$, C-LID uses LID to classify it. When similitude term $D_p$ is univocal C-LID caches it. Thus, at the end of the preprocessing phase C-LID has a set $M = \{D_1 \dots D_n\}$ of cached similitude terms (patterns).

*Problem solving.* In the problem solving phase the reuse policy is applied. First, a new problem $p$ is solved using LID. If LID finds a similitude term $D_p$ that is univocal then $p$ is classified in the same class as the cases in $S_{D_p}$. Otherwise, $S_{D_p}$ contains cases belonging to several solution classes, and the reuse policy states that relevant patterns in $M$ are to be used to solve the problem. The relevant patterns are defined as the subset of patterns that subsume $p$, i.e. $M_p = \{m \in M | m \sqsubseteq p\}$. In other words, the relevant patters are those that share with $p$ the same features that were used to classify previous problems. Let $S_{m_j}$ be the set of cases subsumed by $m_j \in M_p$, we define $S_{M_p}$ as the union of all cases subsumed by any relevant pattern. C-LID applies the *majority rule* to the set $S_{D_p} \cup S_{M_p}$; that is to say, C-LID classifies $p$ as belonging to the solution class to which belong a majority of the cases in $S_{D_p} \cup S_{M_p}$.

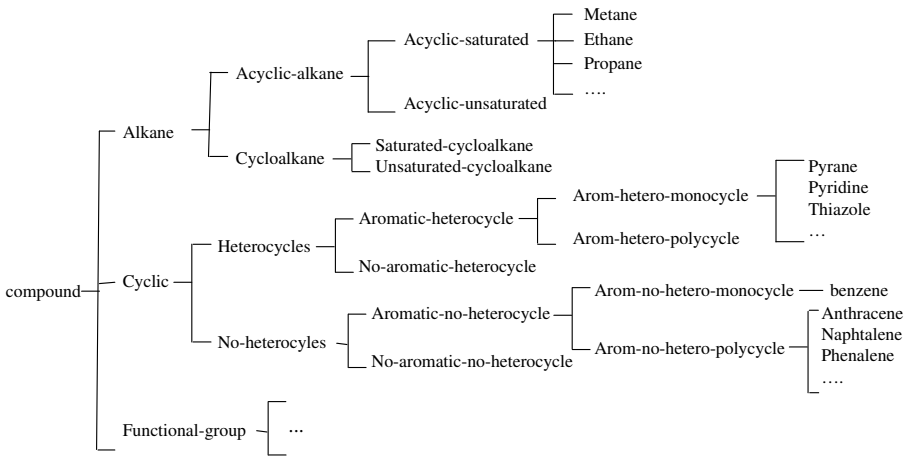



**Fig. 2.** Partial view of the Toxicology ontology

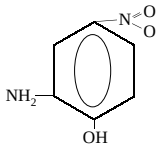
In the next section we explain some experiments on the Toxicology dataset and we evaluate the accuracy both of LID and C-LID.

## 4 The Toxicology Dataset

The Toxicology dataset has been provided by the US National Toxicology Program (NTP) (http://ntp-server.niehs.nih.gov). In this dataset there are descriptions of around 500 chemical compounds that may be carcinogenic for two animal species: rats and mice. The carcinogenic activity of the compounds has proved to be different in both species and also among the sex of the same species. Therefore there are, in fact, four datasets. The chemical compounds of the dataset can be classified into eight solution classes according to the laboratory experiments: *positive*, *clear evidence*, *some evidence*, *equivocal*, *equivocal evidence*, *inadequate study*, *negative* and *negative evidence*. Nevertheless, most of the authors working on this dataset consider the classes *positive*, *clear evidence* and *some evidence* as the class "positive"; the classes *negative* and *negative evidence* as the class "negative"; and the compounds belonging to the other classes are removed.

The classification task in this domain is specially difficult since the predictive accuracy exhibited by the domain human experts ranges from 28% to 78% [7]. In the Predictive Toxicology Challenge 2000-2001 (PTC) [4] several authors presented different approaches for solving the classification task. Most of them try to induce rules in order to characterize both classes. The accuracy obtained by the different methods is around 63%. In fact the maximum accuracy is 65% and it is considered as the default prediction.

In the next section the representation of the chemical compounds using feature terms is explained. Then we discuss the results obtained from the experiments done using both LID and C-LID.




$$TR\text{-}339 = \left[\begin{array}{l} \underline{\mathsf{compound}} \\ \mathsf{main\text{-}group} \doteq benzene \\ \mathsf{p\text{-}radicals} \doteq \left\{ \begin{array}{l} \left[\begin{array}{l} \underline{position\text{-}radical} \\ \mathsf{position} \doteq one \\ \mathsf{radicals} \doteq \left[\begin{array}{l} \underline{compound} \\ \mathsf{main\text{-}group} \doteq alcohol \end{array}\right] \end{array}\right] \\ \left[\begin{array}{l} \underline{position\text{-}radical} \\ \mathsf{position} \doteq two \\ \mathsf{radicals} \doteq \left[\begin{array}{l} \underline{compound} \\ \mathsf{main\text{-}group} \doteq amine \end{array}\right] \end{array}\right] \\ \left[\begin{array}{l} \underline{position\text{-}radical} \\ \mathsf{position} \doteq four \\ \mathsf{radicals} \doteq \left[\begin{array}{l} \underline{compound} \\ \mathsf{main\text{-}group} \doteq nitro\text{-}derivate \end{array}\right] \end{array}\right] \end{array} \right. \end{array}\right]$$

**Fig. 3.** Representation of the compound with identifier TR-339, the *2-amino-4-nitrophenol*, using feature terms.

### 4.1 Representation of the chemical compounds

The basis of the representation we propose is the *chemical ontology* used by chemist experts and that is implicit in the chemical nomenclature of the compounds. For instance, the *benzene* is an aromatic ring composed by six carbon atoms with some well-known properties, therefore it is not necessary to describe the individual atoms in the benzene when we have the *benzene* concept in our domain ontology.

Figure 2 shows part of the chemical ontology we used for representing the compounds in the Toxicology dataset. This ontology is based on the chemical nomenclature which, in turn, is a systematic way of describing a molecule. In fact, the name of a molecule provides to a chemist all the information needed to graphically represent the structure of the molecule.

In our representation (see Figure 3) a *compound* is a sort described by two features: main-group and p-radicals. The values of the feature main-group belong to someone of the sorts shown in Figure 2. The value of the feature p-radicals is a set whose elements are of sort *position-radical*. The sort *position-radical* is described using two features: radicals and position. The value of the feature radicals is also of sort *compound*. This is because both, main group and radicals, are the same kind of molecules, i.e. the benzene may be the main group in one compound and a radical in some other compounds. The feature position of the sort *position-radical* indicates where the radical is bound to the main group.

For example, the chemical compound with identifier TR-339 in the NTP Toxicology dataset, is the *2-amino-4-nitrophenol*. This compound has a phenol as main group. The phenol, is a molecule composed of one benzene with a radical alcohol in position one. Thus, the compound TR-339 has a benzene as main group and a set of three radicals: a radical with an *alcohol* as main group in position one; a radical with an *amine* as main group in position two; and a radical with

| Dataset | # cases | LID | C-LID | Answers | $M$ |
|---|---|---|---|---|---|
| MR | 297 | 58.27 | 60.54 | 39.96 | 57.67 |
| FR | 296 | 63.09 | 66.97 | 56.15 | 60.67 |
| MM | 296 | 52.39 | 53.95 | 32.99 | 53.35 |
| FM | 319 | 52.36 | 56.60 | 32.23 | 53.34 |

**Table 1.** Table comparing the performances of LID (column LID) and C-LID (column C-LID) on male rats (MR), female rats (FR), male mice (MM) and female mice (FM). *Answers* is the percentage of problems that can be classified using the patterns in the set $M$. The column $M$ is the predictive accuracy using only the patterns generated by LID.

a *nitro-derivate* in position four. Notice that this information has been directly extracted from the chemical name of the compound following the nomenclature rules. We have translated, with the support of a chemist, the compounds of the Toxicology dataset to this representation based on feature terms.

In the next section we describe the experiments done using both LID and C-LID for solving the classification task in the Toxicology dataset.

### 4.2 Experiments

The Toxicology dataset can be seen as formed by four separate datasets: male rats (MR), female rats (FR), male mice (MM) and female mice (FM). The classification task is solved separately for each one of these datasets, i.e. solving a case means solving four classification tasks: 1) the classification of the case according to its activity in MR; 2) the classification according to its activity in FR; 3) the classification according its activity in MM; and 4) the classification according its activity in FM. From now on, we will refer to the classification of a case as the classification of the case in one dataset.

For each one of the datasets we take into account only those cases having as solution *positive* or *negative* activity, i.e. we do not considered the cases with value of activity *unknown*, *equivocal* or *inadequate*.

The evaluation of the predictive accuracy of the methods has been made using 10-fold cross-validation. First of all we evaluated the accuracy of LID. The column labelled as LID in Table 1 shows the accuracy of LID for each one of the datasets. Notice that LID has an accuracy of 63.09% for FR that is near to that considered as the default accuracy (65%).

Table 1 also shows the predictive accuracy of C-LID. The predictive accuracy of C-LID improves the accuracies obtained using only LID in the four datasets. These results show that the caching policy is adequate since the cached patterns allow to increase the accuracy of the LID method. Notice that the caching policy stores only the similitude terms that are univocal, i.e. those subsuming cases belonging to only one solution class. With this policy C-LID takes into account only those patterns with clear evidence of a good discrimination among classes.

In order to assess the contribution of the patterns to the accuracy improvement in C-LID we evaluated the accuracy using the set of patterns $M$ alone.

When a problem $p$ is not subsumed by any similitude term in $M$, it cannot be classified. Column labeled as *Answers* in Table 1 shows the average of cases that have been solved using the patterns alone. Notice that only around a third of cases can be solved. The predictive accuracy using only patterns (column $M$ in Table 1) is greater than the accuracy of LID for mice but the accuracy for rats is lower than the LID accuracy.

The cached patterns contribute to C-LID's accuracy increase for two reasons. First, they are used only when LID is unable to yield a univocal solution. In this case C-LID wants to use patterns but not always is possible (since their applicability is about 1/3 of the cases as shown in Table 1). However, the experiments show that when they are used the accuracy of C-LID improves. The way in which patterns improve the system final decision is increasing the set of cases that C-LID take into account when the majority rule is applied. In fact, the second reason is that the cases added are really relevant to the problem C-LID is solving, as proved by the increase in accuracy that they provide.

## 5 Conclusions

We have presented C-LID, a lazy concept learning technique for case-based reasoning in classification tasks. C-LID is built on top of LID by caching and reusing the symbolic similitude terms generated by LID. The rationale of the C-LID is that similitude terms are the local approximations of the target function used to classify problems in CBR.

The cache policy of C-LID keeps only those similitude terms that perfectly classify the subsumed cases (i.e. those similitude terms whose cases in the discriminatory set belong all to a unique class). This policy has the rationale that it is worth caching those similitude terms that are good approximations. Clearly, there are less strict policies that are possible, e.g. caching all similitude terms that have a clear majority class among the cases in their discriminatory set. This policy retains more patterns (and thus increase their scope) but they increase the uncertainty when they are reused. We performed two experiments with less strict caching policies: similarity terms were cached when the majority class in the discriminatory set was greater that 2/3 and then greater than 3/4. The outcome was very similar to that patrons alone in Table 1. Therefore, the increase in scope is undermined by the uncertainty increase in less strict policies.

This result supports the caching policy of patterns, but why do cached patterns improve accuracy? For this we have to consider the reuse policy. First, when the "classic" approach to CBR embodied in LID works perfectly (all retrieved cases are in the same class) the patterns are not reused. That is to say, when the current local approximation is assessed to be without uncertainty no other local approximations (patterns) are reused. However, when LID's outcome involves some uncertainty we assess that this local approximation is not good enough and we search for past local approximations (patterns) that were good enough. Now, for a problem $p$, of all cached patterns $M$ the reuse policy only considers as subset such that $M_p = \{m_i \in M | m \sqsubseteq p\}$—that is to say, those

similitude terms whose informational content is also shared by $p$. Notice that each cached similitude term $m_i \in M_p$ was a good characterization of what was shared by some problem $p$ and a subset of the case base. Since $p$ also shares this symbolic description $m_i$ it is likely that it would be in $m_i$'s class. Thus, the patterns in $M_p$ (if they exist for $p$) can help in reducing the uncertainty intrinsic to the "pure" CBR approach of LID.

Finally, notice that C-LID is clearly a lazy approach to concept learning. C-LID uses local approximations of the target function and not a global approximation (even if it caches generalizations of examples). The contribution of C-LID is that it uses local approximations in a new way: C-LID builds in a problem-centered way a local approximation, and then assess its goodness against the existing case base; if this local approximation is found wanting then C-LID reuses *similar* local approximations that have been cached. The similar local approximations are those that share with the problem $p$ the content of a symbolic description of similarity among cases.

**Acknowledgements** This work has been supported by the MCYT-FEDER Project SAMAP (TIC2002-04146-C05-01). The authors thank Dr. Lluis Bonamusa for his assistance in developing the representation of chemical molecules.

## References

[1] E. Armengol and E. Plaza. Bottom-up induction of feature terms. *Machine Learning*, 41(1):259–294, 2000.

[2] E. Armengol and E. Plaza. Individual prognosis of diabetes long-term risks: A CBR approach. *Methods of Information in Medicine*, pages 46–51, 2001.

[3] E. Armengol and E. Plaza. Lazy induction of descriptions for relational case-based learning. In Luc De Raedt and Peter Flach, editors, *Machine Learning: ECML-2002*, number 2167 in Lecture Notes in Artificial Intelligence, pages 13–24. Springer-Verlag, 2001.

[4] C. Helma, R. King, S. Kramer, and A. Srinivasan. The predictive toxicology challenge 2000-2001. In *ECML/PKDD 2001. Freiburg*, 2001.

[5] Ramon López de Mántaras. A distance-based attribute selection measure for decision tree induction. *Machine Learning*, 6:81–92, 1991.

[6] Tom M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. Computer Science Series, 1997.

[7] Bernhard Pfahringer. (the futility of) trying to predict carcinogenicity of chemical compounds. In *Proceedings of the Predictive Toxicology Challenge Workshop, Freiburg, Germany, 2001.*, 2001.