Lazy Adaptation Knowledge Learning based on Frequent Closed Itemsets

Emmanuel Nauer, Jean Lieber, and Mathieu d'Aquin

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Abstract. This paper focuses on lazy adaptation knowledge learning (LAKL) using frequent closed itemset extraction. This approach differs from eager adaptation knowledge learning (EAKL) by the number of cases used in the learning process and by the moment at which the process is triggered. Where EAKL aims to compute adaptation knowledge once on the whole case base with the idea of solving every future target problem, LAKL computes adaptation knowledge on a subset of cases close to the target problem when a new problem has to be solved. The paper presents experiments on generated datasets from Boolean functions and on a real-world dataset, studying especially how the size of the case base used impacts performance. The results show that LAKL outperforms EAKL in precision and correct answer rate and that it is therefore better not to use the whole case base for adaptation knowledge learning.

Keywords: adaptation knowledge learning, lazy learning, closed itemset extraction, case-based reasoning

1 Introduction

Case-based reasoning (CBR [15]) aims at reusing past experience, represented by cases, and case adaptation plays an important part in many CBR applications. Adaptation often relies on the use of adaptation knowledge (AK), hence the usefulness of an AK learning (AKL) process. For example, AKL has been shown useful to improve the results of Taaable, a cooking system that adapts cooking recipes [3]. AK can take many forms. In this paper, it is assumed to be given by a set of adaptation rules.

If adaptation rules can be extracted in a general context (that is, by exploiting the cases of the whole case base), it appears that the rules extracted from a focused set of cases provide better results. In Taaable, preliminary work showed this interest in proposing an AKL suited to a given recipe (e.g., an apple tart recipe) by computing adaptation rules from recipes of the same type (e.g., fruit tart recipes) [7]. However, this approach, which focuses on a subset of recipes, computes adaptation rules a priori without considering the target problem. This paper addresses adaptation knowledge discovery and its application for adaptation with respect to the target problem, by considering the cases close to the target problem as input data for learning of adaptation rules. This work follows the idea of [2] in which knowledge acquisition is triggered opportunistically at the time of problem solving. However, if such an approach has already been used, a comparative evaluation between exploiting the whole case base or a focused subset of it has, to the best of our knowledge, never been studied. This paper addresses this issue, in the context of a Boolean representation of cases, with the AKL process being based on the extraction of frequent closed itemset (FCI) variations between pairs of cases to compute adaptation rules, as in [9, 14] using only positive cases.

These two ways of using learning, either as an offline process or as a process done at problem solving time, are respectively called *eager* and *lazy* learning [1]. It can be noted that the whole CBR process can be likened to lazy learning. Indeed, one can consider that retrieving k cases nearest to the target problem and learning a solution from them matches closely the case-based problem-solving process. Now, what is studied in this paper, is lazy learning for adaptation, not for the whole CBR process and the supervised learning performed here is not on cases but on variations between cases. The objective of this paper is to study lazy adaptation learning and compare it with the performance of eager adaptation learning. This study shows that, on the tested datasets, lazy learning outperforms eager learning in an AKL process.

Section 2 recalls some general notions about CBR, AKL and the technique for extracting FCIs. Section 3 presents the approach used for AKL in a classical (i.e. eager AKL, or EAKL) way: this reuses some previous work with a little–yet significant–improvement. Section 4 presents lazy AKL (LAKL): from a general viewpoint and the way it is applied using FCI extraction. EAKL and LAKL are evaluated and compared in Section 5. Section 6 concludes and points out some future work.

2 Preliminaries

This section first introduces the main notions and notations related to CBR. Then it describes the principles of adaptation knowledge learning (AKL). Finally, it presents the notions related to the extraction of frequent closed itemsets (FCIs).

2.1 Case-based reasoning: notions, notations, and assumptions

Let $\mathcal{P}$ and $\mathcal{S}$ be two sets. A *problem* (resp., a *solution*) is an element of $\mathcal{P}$ (resp., of $\mathcal{S}$). The existence of a binary relation with the semantics "has for solution" is assumed, although it is not completely known to the CBR system. Moreover, in this article it is assumed that this relation is functional (every problem has exactly one correct solution). Let $\mathtt{f}$ be the function from $\mathcal{P}$ to $\mathcal{S}$ such that $\mathtt{y} = \mathtt{f}(\mathtt{x})$ if $\mathtt{y}$ is the solution of $\mathtt{x}$. A *case* is a pair $(\mathtt{x}, \mathtt{y}) \in \mathcal{P} \times \mathcal{S}$ where $\mathtt{y} = \mathtt{f}(\mathtt{x})$.

A CBR system on $(\mathcal{P}, \mathcal{S}, \mathtt{f})$ is built with a knowledge base $\mathtt{KB} = (\mathtt{CB}, \mathtt{DK}, \mathtt{RK}, \mathtt{AK})$ where $\mathtt{CB}$, the case base, is a finite set of cases, $\mathtt{DK}$ is the domain knowledge, $\mathtt{RK}$ is the retrieval knowledge (in this work, $\mathtt{RK} = \mathtt{dist}$, a distance function on $\mathcal{P}$), and $\mathtt{AK}$ is the adaptation knowledge often represented by a set of adaptation rules.

A CBR system in $(\mathcal{P}, \mathcal{S}, \mathtt{f})$ aims to associate to a query problem $\mathtt{x}^{\mathtt{tgt}}$ a $\mathtt{y}^{\mathtt{tgt}} \in \mathcal{S}$, denoted by $\mathtt{y}^{\mathtt{tgt}} = \mathtt{f}_{\mathrm{CBR}}(\mathtt{x}^{\mathtt{tgt}})$. The function $\mathtt{f}_{\mathrm{CBR}}$ is intended to be an approximation of $\mathtt{f}$. It is built using the following functions:

- The retrieval function, with the profile $\mathtt{retrieval} : \mathtt{x}^{\mathtt{tgt}} \mapsto (\mathtt{x}^s, \mathtt{y}^s) \in \mathtt{CB}$;
- The adaptation function, with the profile $\mathtt{adaptation} : ((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}}) \mapsto \mathtt{y}^{\mathtt{tgt}} \in \mathcal{S}$; it is usually based on $\mathtt{DK}$ and $\mathtt{AK}$. $((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}})$ is an *adaptation problem*.

Thus, $\mathtt{f}_{\mathrm{CBR}}(\mathtt{x}^{\mathtt{tgt}}) = \mathtt{adaptation}(\mathtt{retrieval}(\mathtt{x}^{\mathtt{tgt}}), \mathtt{x}^{\mathtt{tgt}})$.

Adaptation principle using adaptation rules. Generally speaking, an adaptation rule $\mathtt{ar}$ is a function that maps an adaptation problem $((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}}) \in \mathtt{CB} \times \mathcal{P}$ into a $\mathtt{y}^{\mathtt{tgt}} \in \mathcal{S} \cup \{\mathtt{failure}\}$. If $\mathtt{y}^{\mathtt{tgt}} \neq \mathtt{failure}$ then it is proposed as a solution to $\mathtt{x}^{\mathtt{tgt}}$, by adaptation of $(\mathtt{x}^s, \mathtt{y}^s)$ according to $\mathtt{ar}$. The adaptation consists in selecting a set $\mathtt{ApplicableAR}$ of applicable adaptation rules, i.e. for each $\mathtt{ar} \in \mathtt{ApplicableAR}$, $\mathtt{ar}((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}}) \neq \mathtt{failure}$ (the number of selected adaptation rules and the way they are selected depends on the approach and is presented further in the article). Then, the multiset of solutions proposed by the $\mathtt{ar} \in \mathtt{ApplicableAR}$ is combined (e.g. by a majority vote) in order to propose a solution $\mathtt{y}^{\mathtt{tgt}}$ to $\mathtt{x}^{\mathtt{tgt}}$.

One way to represent an adaptation rule is by an ordered pair $(\Delta\mathtt{x}, \Delta\mathtt{y})$ where $\Delta\mathtt{x}$ is a *problem variation* and $\Delta\mathtt{y}$ is a *solution variation*. Intuitively, a problem variation is the representation of changes in a problem knowing that this notion depends on the way the problems are represented (and this is similar for solutions): this idea is clarified in Section 3.

Given an adaptation rule $\mathtt{ar} = (\Delta\mathtt{x}, \Delta\mathtt{y})$ and an adaptation problem $((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}})$, $\mathtt{ar}((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}})$ is computed according to the following principle:

- The variation $\Delta\mathtt{x}^{s,\mathtt{tgt}}$ from $\mathtt{x}^s$ to $\mathtt{x}^{\mathtt{tgt}}$ is computed.
- If $\Delta\mathtt{x}^{s,\mathtt{tgt}}$ does not match exactly $\Delta\mathtt{x}$ then the value $\mathtt{failure}$ is returned (the adaptation rule is not applicable to this adaptation problem).
- Else, from $\mathtt{y}^s$ and $\Delta\mathtt{y}$, a solution $\mathtt{y}^{\mathtt{tgt}}$ is computed, such that the variation $\Delta\mathtt{x}^{s,\mathtt{tgt}}$ from $\mathtt{y}^s$ to $\mathtt{y}^{\mathtt{tgt}}$ matches $\Delta\mathtt{y}$, and, if such a $\mathtt{y}^{\mathtt{tgt}}$ exists, it is returned (else the process returns $\mathtt{failure}$).

2.2 Adaptation knowledge learning

Adaptation based on adaptation rules requires the acquisition of rules that can be done thanks to a learning process. In [10], an approach to adaptation knowledge learning (AKL) is presented that is based on the case base. The intuition is that, given $(\mathtt{x}^i, \mathtt{y}^i), (\mathtt{x}^j, \mathtt{y}^j) \in \mathtt{CB}$, one can consider $((\mathtt{x}^i, \mathtt{y}^i), \mathtt{x}^j)$ as an adaptation problem—$(\mathtt{x}^i, \mathtt{y}^i)$ is the case to be adapted and $\mathtt{x}^j$ is the target problem—and $\mathtt{y}^j$ is a solution to this adaptation problem. Therefore, each pair of distinct source cases can be used as an example for a supervised learning process. This idea has been successfully applied to CBR in many studies (e.g. [4, 11, 5]).

More precisely, the idea is to consider, for a pair of distinct source cases $((\mathtt{x}^i, \mathtt{y}^i), (\mathtt{x}^j, \mathtt{y}^j))$, the problem variation $\Delta\mathtt{x}^{ij}$ from $\mathtt{x}^i$ to $\mathtt{x}^j$ and the solution variation $\Delta\mathtt{y}^{ij}$ from $\mathtt{y}^i$ to $\mathtt{y}^j$. Then, the supervised learning system provides a model that maps a problem variation $\Delta\mathtt{x}$ into a solution variation $\Delta\mathtt{y}$. This model may be of different natures, for example, it can be a neural network (see e.g. [6]). In this article, the learning technique used is frequent closed itemset extraction, in the continuation of previous studies (see e.g. [14]).

2.3 Frequent Closed Itemset extraction

Itemset extraction is a collection of data mining methods to extract regularities from data, by aggregating object items that appear together. Like formal concept analysis, itemset extraction algorithms start from a *formal context* $\mathcal{K}$, defined by $\mathcal{K} = (\mathcal{G}, \mathcal{M}, \mathcal{I})$, where $\mathcal{G}$ is a set of objects, $\mathcal{M}$ is a set of items, and $\mathcal{I}$ is the relation on $\mathcal{G} \times \mathcal{M}$ stating that an object is described by an item [8].

An *itemset* I is a set of items, and the *support* of I, $\mathtt{supp}(I)$, is the number of objects of the formal context that have all the items of I. I is frequent, with respect to a threshold $\tau_{\mathtt{supp}}$, whenever $\mathtt{supp}(I) \geq \tau_{\mathtt{supp}}$. I is closed if there does not exist a proper superset J of I ($I \subsetneq J$) with the same support.

Given an itemset $I \subseteq \mathcal{M}$ with at least two items, an association rule R generated from I is any pair (U, V) of nonempty itemsets such that $I = U \cup V$ and $U \cap V = \emptyset$. Such a rule is denoted by $U \to V$. The support of R is the support of I and the confidence of R is $\mathtt{conf}(R) = \frac{\mathtt{supp}(U \cup V)}{\mathtt{supp}(U)}$. The confidence of R can also be defined as the conditional probability of an object having the items of V knowing that it has the items of U (given a uniform probability distribution on G).

3 EAKL based on FCI Extraction

In previous studies, AKL based on FCI extraction has already been studied as an offline process, that is, as an eager AKL [9, 14]. This section recalls the principles of this approach with an improvement in the way the set of adaptation rules is selected at adaptation time.

This (eager) learning process, as mentioned in Section 2.2, uses the following training set:

$$\mathtt{TS} = \left\{ \left(\Delta\mathtt{x}^{ij}, \Delta\mathtt{y}^{ij}\right) \mid (\mathtt{x}^i, \mathtt{y}^i), (\mathtt{x}^j, \mathtt{y}^j) \in \mathtt{CB}, (\mathtt{x}^i, \mathtt{y}^i) \neq (\mathtt{x}^j, \mathtt{y}^j) \right\} \qquad (1)$$

This involves the issue of the representation of problem and solution variations. It is assumed that a problem can either be represented by a set of attribute-value pairs with Boolean values or that it can be translated into such a formalism,[1] which is consistent with the choice of FCI as a learning technique.

[1] This translation may involve some loss of information. For example, a numerical feature is typically translated into several Boolean attributes, each of them consisting in the membership to an interval.

The attribute-value pair $(a, 1)$ is simply written a and the attribute-value pair $(a, 0)$ is simply written $\overline{a}$. It is also assumed that problems are fully described: for a problem $\mathtt{x}$ and for each attribute a in $\mathcal{V}_\mathcal{P}$, either $a \in \mathtt{x}$ or $\overline{a} \in \mathtt{x}$.

Now, let us consider the following examples of problems:

$$\mathtt{x}^i = \{a, \overline{b}, \overline{c}, d\}$$
$$\mathtt{x}^j = \{a, \overline{b}, c, \overline{d}\}$$

The variation $\Delta\mathtt{x}^{ij}$ is represented by a set of expressions of the form a^v where $a \in \mathcal{V}_\mathcal{P}$ and v is a *variation symbol.* The variation symbols considered here are `=0`, `=1`, `+`, and `-`, meaning respectively "staying false", "staying true", "changing from false to true", and "changing from true to false". Therefore,

$$\Delta\mathtt{x}^{ij} = \{a^{\mathtt{=1}}, b^{\mathtt{=0}}, c^{\mathtt{+}}, d^{\mathtt{-}}\}$$

Solution variations are represented similarly, on the set of attributes $\mathcal{V}_\mathcal{S}$ (and it is assumed that $\mathcal{V}_\mathcal{P} \cap \mathcal{V}_\mathcal{S} = \emptyset$).

Then, the EAKL process builds a formal context $\mathcal{K} = (\mathcal{G}, \mathcal{M}, \mathcal{I})$ from TS as follows:

– $\mathcal{G}$ is the set of elements of TS, indexed by the pairs (i, j), thus $|\mathcal{G}| = |\mathtt{CB}| \times (|\mathtt{CB}| - 1)$;
– $\mathcal{M}$ is the set of a^v for $a \in \mathcal{V}_\mathcal{P} \cup \mathcal{V}_\mathcal{S}$ and $v \in \{\mathtt{=0}, \mathtt{=1}, \mathtt{+}, \mathtt{-}\}$ thus $|\mathcal{M}| = 4\,|\mathcal{V}_\mathcal{P} \cup \mathcal{V}_\mathcal{S}|$;
– $\mathcal{I}$ is defined by the fact that the object indexed by (i, j) is described by the item a^v meaning that $a^v \in \Delta\mathtt{x}^{ij} \cup \Delta\mathtt{y}^{ij}$.

After this, FCI extraction is run on $\mathcal{K}$ and the results is filtered in order to keep only the itemsets I having at least one item a^v where $a \in \mathcal{V}_\mathcal{P}$ and one item A^w where $A \in \mathcal{V}_\mathcal{S}$ (with v and w, two variation symbols). Each FCI I is then translated into an adaptation rule. For example, $I = \{a^{\mathtt{=1}}, c^{\mathtt{+}}, A^{\mathtt{-}}\}$ is translated into $\mathtt{ar} = (\Delta\mathtt{x}, \Delta\mathtt{y})$ with $\Delta\mathtt{x} = \{a^{\mathtt{=1}}, c^{\mathtt{+}}\}$ and $\Delta\mathtt{y} = \{A^{\mathtt{-}}\}$.

In this way, a set AK of NER adaptation rules is built and can be used at adaptation time, where NER is a parameter that sets the number of rules to be extracted, selecting the ones with greater support. The value for NER has to be chosen carefully, since an insufficient number of rules can cause the adaptation process to fail, while a large number of rules would make the process too time-consuming. Given an adaptation problem $((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}})$, $\Delta\mathtt{x}^{s,\mathtt{tgt}}$ is computed and, for each adaptation rule $(\Delta\mathtt{x}, \Delta\mathtt{y})$ the application process follows the principle given in Section 2.1 knowing that

– $\Delta\mathtt{x}^{s,\mathtt{tgt}}$ matches $\Delta\mathtt{x}$ if $\Delta\mathtt{x} \subseteq \Delta\mathtt{x}^{s,\mathtt{tgt}}$ (and the same definition of matching holds for solutions);
– $\Delta\mathtt{y}$ is said to be *applicable* to $\mathtt{y}^s$ if there exists a $\mathtt{y} \in \mathcal{S}$ such that the variation from $\mathtt{y}^s$ to $\mathtt{y}$ matches $\Delta\mathtt{y}$ (for example, $\{A^{\mathtt{+}}\}$ is applicable on $\{\overline{A}, B\}$ but not on $\{A, B\}$);
– If $\Delta\mathtt{y}$ is not applicable to $\mathtt{y}^s$, then the application of the rule fails, else, for each $A \in \mathcal{V}_\mathcal{S}$:

- If no A^w occurs in $\varDelta\mathtt{y}$ then the value of attribute A in $\mathtt{y}^s$ is reused for $\mathtt{y}^{\mathtt{tgt}}$ (if $A \in \mathtt{y}^s$ then $A \in \mathtt{y}^{\mathtt{tgt}}$ else $\overline{A} \in \mathtt{y}^{\mathtt{tgt}}$);
- Else, the value of attribute A in $\mathtt{y}^{\mathtt{tgt}}$ is computed according to its value in $\mathtt{y}^s$ and the variation w (e.g. if $\mathtt{y}^s = \{A\}$ and $w = \mathtt{-}$ then $\mathtt{y}^{\mathtt{tgt}} = \{\overline{A}\}$).

For the purpose of selecting adaptation rules among AK, it is useful to assign them a score used to rank adaptation rules in order of preference: the higher the score, the more preferred the adaptation rule. In the previous work, this score was simply the support of the itemset on which the adaptation rule is built. In this work, the notion of confidence of an adaptation rule has been used and has led to better results. We therefore switched to the use of confidence in our experiments. This confidence is defined as follows. First, four new variation symbols $\mathtt{1}\bullet$, $\bullet\mathtt{1}$, $\mathtt{0}\bullet$ and $\bullet\mathtt{0}$ are introduced: $\mathtt{1}\bullet$ (resp. $\bullet\mathtt{1}$) corresponds to a variation from a true value to any value (resp. from any value to a true value), and similarly for the two other variations. Therefore, $\{A^{\mathtt{=1}}\}$ can be rewritten as $\{A^{\mathtt{1}\bullet}, A^{\bullet\mathtt{1}}\}$, $\{A^{\mathtt{+}}\}$ can be rewritten as $\{A^{\mathtt{0}\bullet}, A^{\bullet\mathtt{1}}\}$, etc. This decomposition is useful to separate the information on $\mathtt{y}^s$ (in the premise of the rule) from the information on $\mathtt{y}^{\mathtt{tgt}}$ (in its conclusion). Let $(\varDelta\mathtt{x}, \varDelta\mathtt{y})$ be an adaptation rule built from an itemset I, where the A^w for $A \in \mathcal{V}_{\mathcal{S}}$ are replaced with A^{w_1} and A^{w_2}, $w_1 \in \{\mathtt{1}\bullet, \mathtt{0}\bullet\}$ and $w_2 \in \{\bullet\mathtt{1}, \bullet\mathtt{0}\}$. Then, the confidence of the adaptation rule $(\varDelta\mathtt{x}, \varDelta\mathtt{y})$ is the confidence of the association rule $U \to V$ where V is the subset of I with variation symbols $\bullet\mathtt{1}$ and $\bullet\mathtt{0}$, and $U = I \setminus V$. For example, if $\varDelta\mathtt{x} = \{a^{\mathtt{=1}}, b^{\mathtt{-}}, c^{\mathtt{=0}}\}$ and $\varDelta\mathtt{y} = \{A^{\mathtt{-}}\}$ then $U = \{a^{\mathtt{=1}}, b^{\mathtt{-}}, c^{\mathtt{=0}}, A^{\mathtt{1}\bullet}\}$ and $V = \{A^{\bullet\mathtt{0}}\}$.

4 LAKL based on FCI Extraction

LAKL consists in learning AK at adaptation time, that is, when an adaptation problem $((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}})$ is known. Therefore, unlike EAKL, LAKL can rely on knowledge of the retrieved case $(\mathtt{x}^s, \mathtt{y}^s)$ and the target problem $\mathtt{x}^{\mathtt{tgt}}$. The general principle of LAKL is presented in Section 4.1 and its application using FCI extraction, in Section 4.2. In Section 4.3, it is explained that an approach to CBR developed in previous studies can be considered as a particular approach to LAKL.

4.1 Main principles

The main idea of LAKL is to restrict the training set TS defined by (1) by taking into account the adaptation problem: given two distinct source cases $(\mathtt{x}^i, \mathtt{y}^i)$ and $(\mathtt{x}^j, \mathtt{y}^j)$, the pair $(\varDelta\mathtt{x}^{ij}, \varDelta\mathtt{y}^{ij})$ is filtered out from the training set if these source cases do not satisfy conditions related to $(\mathtt{x}^s, \mathtt{y}^s)$ and $\mathtt{x}^{\mathtt{tgt}}$. Generally speaking, three types of conditions are considered below in an informal way.

$(\mathtt{x}^i, \mathtt{x}^j \simeq \mathtt{x}^{\mathtt{tgt}})$ The problems $\mathtt{x}^i$ and $\mathtt{x}^j$ are "around" $\mathtt{x}^{\mathtt{tgt}}$: the way adaptation works for the target problem is assumed to be better represented in its neighborhood then elsewhere in the problem space;

$(\Delta\mathtt{x}^{ij} \simeq \Delta\mathtt{x}^{s,\mathtt{tgt}})$ The variation between the two source problems $\mathtt{x}^i$ and $\mathtt{x}^j$ are similar to the variation between the problem $\mathtt{x}^s$ of the retrieved case and the target problem.

$(\mathtt{y}^i \simeq \mathtt{y}^{\mathtt{tgt}})$ The variation $\Delta\mathtt{y}^{ij}$ from $\mathtt{y}^i$ to $\mathtt{y}^j$ has to be "compatible" with the value of the solution of the retrieved problem so that rules that are not applicable to $\mathtt{y}^s$ are avoided; this condition adds a constraint on the "left part" of $\Delta\mathtt{y}^{ij}$, that is, $\mathtt{y}^i$.

Our approach to applying LAKL, as described in the following, consists in specifying and implementing these conditions.

4.2 Lazy adaptation knowledge learning based on FCI extraction

The principles of LAKL have been implemented for an AKL based on FCI extraction.

Condition $(\mathtt{x}^i, \mathtt{x}^j \simeq \mathtt{x}^{\mathtt{tgt}})$ is implemented simply by choosing the k nearest cases to the target problem that are different from the case $(\mathtt{x}^s, \mathtt{y}^s)$. This can be done at retrieval time: $k + 1$ nearest cases are retrieved, the nearest one being the retrieved case, and the k other ones being selected for building the training set. Different values of k can be considered. In our experiments, those values have been chosen to correspond to different percentages of the size of the case base, starting with $10\% \,|\mathtt{CB}|$.

One effect of applying this condition $(\mathtt{x}^i, \mathtt{x}^j \simeq \mathtt{x}^{\mathtt{tgt}})$ on the training set is to reduce the number of objects in the formal context $\mathcal{K} = (\mathcal{G}, \mathcal{M}, \mathcal{I})$, which goes from $|\mathtt{CB}|\,(|\mathtt{CB}| - 1)$ to $k(k-1)$, hence a reduction of about $(|\mathtt{CB}|\,/k)^2$ of its size. For example, if $k = 10\% \,|\mathtt{CB}|$, this leads to a reduction by a factor 100.

In this version, the condition $(\Delta\mathtt{x}^{ij} \simeq \Delta\mathtt{x}^{s,\mathtt{tgt}})$ has not been implemented for practical reasons and this is left for future work.

Finally, the condition $(\mathtt{y}^i \simeq \mathtt{y}^{\mathtt{tgt}})$ is implemented in this work considering that $\mathtt{y}^i$ and $\mathtt{y}^{\mathtt{tgt}}$ must be equal. This can be justified as follows, in the situation where there is only one solution attribute A (as in the experiments). Considering an $(\mathtt{x}^i, \mathtt{y}^i)$ such that $\mathtt{y}^i \neq \mathtt{y}^s$ would lead to the extraction of FCIs that are interpreted as inapplicable rules. For example, $\mathtt{y}^i = \{\overline{A}\}$ would lead to extract adaptation rules containing either $A^{\mathtt{+}}$ or $A^{\mathtt{=0}}$. Therefore, if $\mathtt{y}^i \neq \mathtt{y}^s$ then $\mathtt{y}^s = \{A\}$ and no such adaptation rule can be applied on $\mathtt{y}^s$, hence not contributing to the learning effort. This condition further reduces the number of objects in the formal context. Moreover, the set of items $\mathcal{M}$ is also reduced since items not related to any object are not included in the formal context: in the example above, the items $A^{\mathtt{+}}$ and $A^{\mathtt{=0}}$ are excluded from $\mathcal{M}$ (and this reduces the computing time).

4.3 Analogical extrapolation seen as a LAKL process

In [13], the analogical extrapolation approach to CBR is described. It is based on analogical proportions, i.e. quaternary relations denoted by $a\!:\!b\!::\!c\!:\!d$ ("a is to b as c is to d") where a, b, c, and d are four objects of the same space. The analogical proportion on Booleans used in this article is defined by $a\!:\!b\!::\!c\!:\!d$ if

$a = b$ and $c = d$, or $a = c$ and $b = d$, and on tuples of Booleans by $a\!:\!b\!::\!c\!:\!d$ if $a_p\!:\!b_p\!::\!c_p\!:\!d_p$ for each component p. Analogical extrapolation, at adaptation time, consists in the following steps:

- Selection of all pairs $((\mathtt{x}^a, \mathtt{y}^a), (\mathtt{x}^b, \mathtt{y}^b)) \in \mathtt{CB}^2$ such that $\mathtt{x}^a\!:\!\mathtt{x}^b\!::\!\mathtt{x}^s\!:\!\mathtt{x}^{\mathtt{tgt}}$;
- For each of these pairs, solve the analogical equation $\mathtt{y}^a\!:\!\mathtt{y}^b\!::\!\mathtt{y}^c\!:\!y$ in y and, if it has at least one solution, add this solution to a multiset Y of candidate solutions (initially, Y is empty);
- The solution $\mathtt{y}^{\mathtt{tgt}}$ is based on a vote on Y (this vote is weighted in [12] using the notions of support and confidence of a case pair, that are similar to the notions of support and confidence of an FCI).

This analogical extrapolation adaptation process can be reformulated as an adaptation following a LAKL process with the training set corresponding to some implementation choices for the conditions introduced in 4.1. More precisely, conditions $(\mathtt{x}^i, \mathtt{x}^j \simeq \mathtt{x}^{\mathtt{tgt}})$ and $(\mathtt{y}^i \simeq \mathtt{y}^{\mathtt{tgt}})$ are not considered and condition $(\Delta\mathtt{x}^{ij} \simeq \Delta\mathtt{x}^{s,\mathtt{tgt}})$ is defined as follows:

- The set of variation symbols is reduced to $\{\mathtt{-}, \mathtt{+}\}$ (i.e. items of the forms $a^{\mathtt{=1}}$ and $a^{\mathtt{=0}}$ are discarded[2]);
- The matching between the two problem variations is implemented as an equality: $\Delta\mathtt{x}^{ij} = \Delta\mathtt{x}^{s,\mathtt{tgt}}$.

5 Evaluation

The objective of the evaluation is to study the behavior of LAKL compared to EAKL, in representative types of Boolean functions as well as on a "real life" dataset, in order to determine which approach achieves greater performance and under which circumstances. The experimental results are presented and discussed.

5.1 Experiment setting

In the experiment, problems have 16 Boolean features and $\mathcal{S}$, 1. The function $\mathtt{f} : \mathcal{P} \to \mathcal{S}$ is generated randomly using the following generators that are based on two normal forms, with the purpose of generating a wide variety of functions:

`DNF` $\mathtt{f}$ is generated in a disjunctive normal form, i.e., $\mathtt{f}(\mathtt{x})$ is a disjunction of n_{conj} conjunctions of literals, for example $\mathtt{f}(\mathtt{x}) = (\mathtt{x}_1 \wedge \neg\mathtt{x}_7) \vee (\neg\mathtt{x}_3 \wedge \mathtt{x}_7 \vee \mathtt{x}_8) \vee \mathtt{x}_4$. The value of n_{conj} is randomly chosen uniformly in $\{3, 4, 5\}$. Each conjunction is generated on the basis of two parameters, $p^+ > 0$ and $p^- > 0$, with $p^+ + p^- < 1$: each variable $\mathtt{x}_i$ occurs in the disjunct in a positive (resp. negative) literal with a probability p^+ (resp., p^-). In the experiment, the values $p^+ = p^- = 0.1$ were chosen.

[2] This is related to the following result. Let $\delta\mathtt{x}^{ij} = \{a^v \in \Delta\mathtt{x}^{ij} \mid v \in \{\mathtt{-}, \mathtt{+}\}\}$. Then, it can be shown that $\mathtt{x}^a\!:\!\mathtt{x}^b\!::\!\mathtt{x}^s\!:\!\mathtt{x}^{\mathtt{tgt}}$ iff $\delta\mathtt{x}^{ab} = \delta\mathtt{x}^{s\,\mathtt{tgt}}$.

`Pol` is the same as `DNF`, except that the disjunctions ($\vee$) are replaced with exclusive or's ($\oplus$), thus giving a polynomial normal form. The only different parameter is $p^- = 0$ (only positive literals occur in polynomial normal form).

A case base `CB` of a given size ($|\texttt{CB}| \in \{128, 256, 512\}$) is generated randomly in the following way: each source case $(\texttt{x}, \texttt{y})$ is generated by randomly choosing `x` in $\mathcal{P}$ with a uniform distribution and $\texttt{y} = \texttt{f}(\texttt{x})$.

Let `ntp` be the number of target problems posed to the system, `na` be the number of (correct or incorrect) answers ($\texttt{ntp} - \texttt{na}$ is the number of target problems for which the system fails to propose a solution), and `nca` be the number of correct answers (i.e., number of problems for which the predicted answer is the correct answer). For each method, the following scores are computed:

The precision `prec` is the average of the ratios $\dfrac{\texttt{nca}}{\texttt{na}}$.

The answer rate `arate` is the average of the ratios $\dfrac{\texttt{na}}{\texttt{ntp}}$.

The correct answer rate `car` is the average of the ratios $\dfrac{\texttt{nca}}{\texttt{ntp}}$.

The average is calculated by solving 100 problems for each of 20 generated functions, for each function generator. For the sake of reproducibility, the code for this experiment is available at `https://tinyurl.com/EAKLvsLAKLTests` with the detailed results of this paper (generated functions and details of the evaluation).

5.2 Results

Comparing EAKL and LAKL requires a lot of attention because some parameters involved in the evaluation may have an impact on the interpretation of the results. For example, one of the parameters is the number of rules to be learned using FCI extraction, and it is not fair to use the same number of rules for EAKL, in which rules are extracted once to solve all coming target problems, as for LAKL, in which a focused rule extraction is done for each target problem. The number of extracted rules has a direct and important impact on the answer rate for EAKL, since it is less likely to find applicable rules for a given target problem if that number is too small. The impact of the number of rules is lower for LAKL, and the same number of rules will more likely lead to a higher answer rate in LAKL than in EAKL. Table 1 illustrates this with precision and answer rate scores for EAKL with a number of rules extracted ranging from 100 to $2,400$. It can be seen that an answer rate of 100% is almost always achieved if a sufficient number of rules is used. This sufficient number of rules depends on the size of the case base, sometimes requiring more than $1,000$ rules. However, a (close to perfect) score of 99% is reached in all cases from 400 rules.

The precision also increases with the number of rules until a plateau is reached. Therefore, using more rules (e.g. $2,000$ for `DNF` functions with a size of case base of 256) improves precision and, thanks to an almost perfect answer rate, the correct answer rate. It appears that precision could have a higher score

NER	100	200	300	400	500	600	700	800	1000	1200	1400	1600	1800	2000	2200	2400
DNF																
128 prec	84	83	83	84	85	85	85	85	85	85	**86**	**86**	**86**	**86**	**86**	**86**
σ	8	8	8	**7**	**7**	**7**	**7**	**7**	**7**	**7**	8	8	8	8	8	8
arate	94	99	99	99	99	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
car	79	82	83	84	84	85	85	85	85	85	**86**	**86**	**86**	**86**	**86**	**86**
256 prec	87	87	87	88	88	89	89	89	89	89	89	89	89	**90**	**90**	**90**
σ	9	9	9	8	8	7	8	7	7	7	7	7	**6**	**6**	**6**	**6**
arate	95	99	99	99	99	99	99	99	99	**100**	**100**	**100**	**100**	**100**	**100**	**100**
car	83	86	87	88	88	89	89	89	89	89	89	89	89	**90**	**90**	**90**
512 prec	**95**	94	93	93	93	93	93	93	93	93	93	93	93	93	93	93
σ	4	4	4	4	4	4	4	**3**	4	4	4	4	4	4	4	4
arate	91	96	98	99	99	99	99	99	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
car	87	90	91	92	**93**	**93**	**93**	**93**	**93**	**93**	**93**	**93**	**93**	**93**	**93**	**93**
POL																
128 prec	74	**75**	**75**	**75**	74	74	74	74	74	74	74	74	74	74	74	74
σ	15	**14**	15	15	15	16	16	16	16	16	17	17	17	17	17	17
arate	96	**99**	**99**	**99**	**99**	**99**	**99**	**99**	**99**	**99**	**99**	**99**	**99**	**99**	**99**	**99**
car	71	**74**	**74**	**74**	**74**	**74**	**74**	**74**	**74**	**74**	**74**	**74**	**74**	**74**	**74**	**74**
256 prec	74	75	76	77	77	78	78	78	**79**	**79**	**79**	**79**	**79**	**79**	**79**	**79**
σ	18	16	17	16	15	15	15	15	**14**	15	**14**	**14**	**14**	**14**	**14**	**14**
arate	99	99	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
car	73	75	76	77	77	78	78	78	**79**	**79**	**79**	**79**	**79**	**79**	**79**	**79**
512 prec	79	79	80	81	81	81	81	82	82	82	82	82	**83**	**83**	**83**	**83**
σ	20	18	17	16	16	16	16	15	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**
arate	91	98	99	99	99	99	99	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
car	72	78	80	80	80	81	81	82	82	82	82	82	**83**	**83**	**83**	**83**

Table 1. Precision, standard deviation, answer rate, and correct answer rate (in %) for EAKL (100% of CB) with various numbers of extracted rules (NER) to solve the target problems.

with fewer rules (e.g., 100 rules for DNF functions using 512 cases), but in this case, at the cost of a lower answer rate.

The standard deviation is also stable with lower values when the number of rules is high. Based on these observations, the tests for comparing EAKL and LAKL (presented in the next section) have been performed with the extraction and use at most 2,400 rules for EAKL and 1,000 for LAKL, with the idea of having the best possible results for EAKL (best answer rate, best precision and therefore best correct answer rate) and of studying how LAKL performs in comparison.

EAKL vs. LAKL results. Table 2 shows the precision, standard deviation, answer rate, and correct answer rate obtained with the nearest neighbor adaptation (NN, i.e., simply reusing the solution of the most similar problem) used as baseline, EAKL, and LAKL. The left part of the table presents the results for NN and EAKL while the right part presents the LAKL results with various

				LAKL with x% of |`CB`|									
		NN	EAKL	10	20	30	40	50	60	70	80	90	100
				DNF									
128	`prec`	76	[83 ; 86]	86	88	88	89	89	89	89	89	**90**	85
	σ	5	[7 ; 8]	6	6	**5**	6	6	7	7	7	7	7
	`arate`	100	[94 ; 100]	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	99	**100**
	`car`	76	[79 ; 86]	86	88	88	89	89	89	89	89	**90**	85
256	`prec`	79	[87 ; 90]	91	**92**	**92**	**92**	**92**	**92**	**92**	**92**	**92**	89
	σ	6	[6 ; 9]	5	**4**	5	5	5	6	6	6	6	7
	`arate`	100	[95 ; 100]	**100**	**100**	**100**	99	99	99	99	99	99	99
	`car`	79	[83 ; 90]	91	**92**	**92**	**92**	**92**	**92**	**92**	**92**	**92**	89
512	`prec`	83	[93 ; 95]	96	96	**97**	**97**	**97**	96	96	96	96	93
	σ	5	[3 ; 4]	**2**	3	**2**	3	3	3	3	3	4	4
	`arate`	100	[91 ; 100]	**100**	99	99	99	99	99	99	99	99	**100**
	`car`	83	[87 ; 93]	96	96	**97**	**97**	**97**	96	96	95	95	93
				POL									
128	`prec`	65	[74 ; 75]	69	72	75	76	77	**78**	**78**	**78**	**78**	74
	σ	13	[14 ; 17]	20	19	17	17	**16**	**16**	**16**	**16**	18	**16**
	`arate`	100	[96 ; 99]	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	99	99
	`car`	65	[71 ; 74]	69	72	75	76	77	**78**	**78**	**78**	**78**	74
256	`prec`	64	[74 ; 79]	73	79	81	**82**	81	80	79	79	78	79
	σ	17	[14 ; 18]	19	15	**13**	14	14	15	16	16	17	14
	`arate`	100	[99 ; 100]	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	99	**100**
	`car`	64	[73 ; 79]	73	79	81	**82**	81	80	79	79	78	79
512	`prec`	69	[79 ; 83]	85	**89**	88	87	85	84	83	84	84	82
	σ	15	[14 ; 20]	14	**11**	12	14	16	17	17	17	16	14
	`arate`	100	[91 ; 100]	**100**	**100**	99	99	99	99	99	99	99	**100**
	`car`	69	[72 ; 83]	85	**89**	88	87	85	84	83	83	83	82

Table 2. Precision, standard deviation, answer rate, and correct answer rate (in %) for all approaches.

sizes of subparts of the case base used for learning. For EAKL, all measures vary according to the number of rules used in the adaptation process; the detailed EAKL results of Table 1 are presented through an interval in the column titled EAKL. The interval is defined by the minimum and maximum scores, using between 100 and 2,400 rules. For example, in the first line, for $|\texttt{CB}| = 128$, the precision of the eager approach varies between 83% and 86% depending on the number of rules used in the adaptation process.

The first remark is that EAKL always outperforms the NN approach. This is not a surprise since previous studies (e.g. [9]) already showed this. Also unsurprisingly, the larger the case base size, the better the results of EAKL in precision and correct answer rate (up to +4% and +6% for both measures, respectively, for `DNF` and `Pol` functions when $|\texttt{CB}| = 512$).

The comparison of EAKL and LAKL shows that LAKL performs better with some variations in the `DNF` and `Pol` functions. For `DNF`, it can be observed that

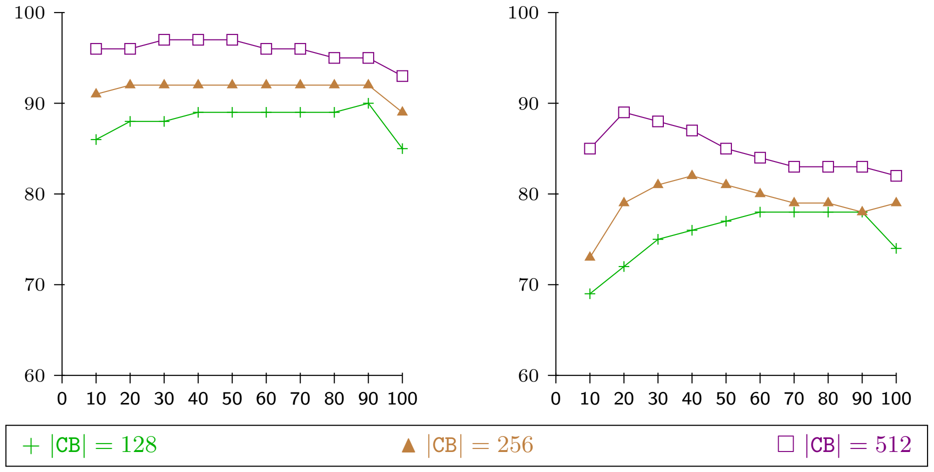

Fig. 1. Correct answer rate (y-axis) given a percentage (x-axis) of `CB` used for 1000 rules extraction (`DNF` at the left, `Pol` at the right).

LAKL always provides better scores for all measures regardless of the percentage of cases used for learning, except for the limits (10% and 100% of `CB`). At best, the correct answer rate has been improved respectively by +4%, +2% and +4% for |`CB`| of 128, 256 and 512. For the `Pol` functions, the results are slightly different. The same observations can be made (i.e. improvement of all measures) but in different conditions. The best correct answer rate scores are, respectively, +4%, +3% and +6% for |`CB`| of 128, 256 and 512, and depend more on the percentage of `CB` used for learning: concentrated between 50% and 90% with |`CB`| = 128, between 30% and 50% with |`CB`| = 256, and between 20% and 30% with |`CB`| = 256. Moreover, except for `Pol` with |`CB`| = 128, a lower standard deviation is also achieved with LAKL.

The best precision, as well as the best correct answer rate (because the answer rate is at its maximum), is obtained with a size of the subpart of the case base between 20% and 50% of `CB`. Fig. 1 shows the correct answer rate according to the percentage of `CB` used. This measure is relatively stable, except for the extreme percentage: 10% and 100% for which the adaptation rules are of lower quality. Thus, an unexpected result is that it is better not to use the whole case base to learn adaptation rules.

Computation aspects. The experiments use Coron, a software platform that implements efficient algorithms for symbolic data mining and, especially, FCI computation [16].

LAKL provides an important advantage for large case bases. Indeed, computing frequent closed itemset is a process with a high level of complexity, especially with the use of case variations. For 1024 cases, the formal context is

				LAKL with x% of $\|$`CB`$\|$									
		NN	EAKL	10	20	30	40	50	60	70	80	90	100
128	`prec`	58	56	59	**62**	60	59	57	57	56	56	55	56
	σ	5.95	6.55	3.10	**0.45**	3.50	1.85	1.15	0.85	1.00	1.40	4.20	6.55
	`arate`	**100**	**100**	95	98	99	99	99	99	99	99	99	**100**
	`car`	58	56	56	**60**	**60**	58	57	56	56	55	55	56
256	`prec`	58	53	**60**	59	57	56	56	54	53	53	52	53
	σ	4.00	2.25	**0.05**	5.25	6.70	6.00	7.10	5.50	3.40	2.90	1.80	2.25
	`arate`	**100**	98	98	98	98	98	99	99	99	99	99	98
	`car`	58	52	**59**	58	56	55	55	54	53	52	52	52

Table 3. Precision, standard deviation, answer rate, and correct answer rate for NN, EAKL and LAKL for various percentages of `CB` used for adaptation rules learning, for the student dataset.

of $1,024 \times 1,023$ pairs of cases, so over one million objects. Extracting up to $2,500$ rules requires to compute closed itemsets with a low support (around 1% of the number of objects). Our experiment has shown that this computation is not possible on a computer without at least 32Gb of memory available for this process with Coron. Therefore, computing the adaptation rules on a smaller formal context (e.g. with $|$`CB`$| = 1024$, taking 10% produces a formal context of around $10,000$ objects, instead of $1,000,000$) is an interesting option for a large case base. This remark holds also for the execution time, for which the smaller the set of cases used to compute adaptation rules, the shorter the computational time. For example, for the `DNF` functions, with $|$`CB`$| = 512$ and $2,500$ rules extracted for EAKL, the mean computation time is 41.78 seconds. This can be compared to the time increases from 8.66 seconds using 20% of `CB` to 11.6 for 50%, up to 31.2 seconds for 90%, when getting $1,000$ rules with LAKL.

However, even if LAKL is more scalable, FCI extraction being done at adaptation time, and therefore for every target problem, its computational time cannot be claimed to be advantageous.

5.3 Experiment on a real-world dataset

To further validate the benefits of LAKL, we conducted an experiment on a real-world dataset. This dataset, taken from the UCI Machine Learning Repository, describes students and their results in a final exam.[3] In fact, there are 2 datasets, but we only use the one with the final result of the math exam. The raw attributes that are binary, multivalued, or even numeric have been transformed to fit our Boolean representation formalism. The details of the transformation are described at `https://tinyurl.com/EAKLvsLAKLTests`.

In this experiment, problems have 29 Boolean features and $\mathcal{S}$, 1. The dataset contains 395 records, which produces 391 cases after removing duplicates. The distribution of the solutions of these 391 cases is 207 (resp. 184) results over (resp.

[3] `https://archive.ics.uci.edu/ml/datasets/student+performance`

lower or equal to) the class's mean result (10.42). One run consists of a random selection in CB and the evaluation of 100 random target problems that are not in CB, using $1,000$ rules for EAKL as well as for LAKL, as this leads to high answer rates. 20 runs were performed for $|\mathtt{CB}| = 128$ and $|\mathtt{CB}| = 256$. Table 3 presents the results for NN, EAKL and LAKL depending on various percentages of CB used for adaptation rule learning. EAKL and LAKL with 100% give the same results because they both use the same number of rules and the same number of cases. The analysis of the results is similar to the experiment on Boolean functions: LAKL outperforms EAKL. In this particular experiment, when at most 60% of CB is used. The results are especially good for low percentages. For $|\mathtt{CB}| = 128$ (resp. 256), the better result is when using 20% (resp. 10%) of CB, with an improvement of the correct answer rate of +4% (resp. +7%) compared to EAKL. Moreover, in this experiment, EAKL leads to results that are less satisfactory than those of NN, which is not the case with LAKL.

6 Conclusion

This paper addresses FCI-based adaptation rule learning using a lazy approach. The AKL is triggered at problem-solving time to compute adaptation rules using cases close to the target problem, rather than a priori and on the entire case base, with the idea of solving every target problem. The lazy approach has many advantages. Experiments on Boolean functions and on a real-world dataset show that the lazy approach (LAKL) outperforms the eager approach (EAKL) in precision and correct answer rate. Moreover, computing adaptation rules in a lazy process is demonstrably more scalable as it requires a fewer number of objects to be involved in the most complex step, namely FCI extraction.

A first direction of work is the study of various ways of implementing the conditions $(\mathtt{x}^i, \mathtt{x}^j \simeq \mathtt{x}^{\mathtt{tgt}})$, $(\Delta\mathtt{x}^{ij} \simeq \Delta\mathtt{x}^{s,\mathtt{tgt}})$, and $(\mathtt{y}^i \simeq \mathtt{y}^{\mathtt{tgt}})$ that restricts the training set for LAKL. Two implementations have already been considered: the one studied in this article and the one corresponding to analogical extrapolation as a LAKL (cf. Section 4.3).

Another direction of work consists in studying LAKL with other assumptions on case representation. First, in some domains, cases cannot adequately be translated into sets of Boolean properties. This motivates the study of LAKL with other learning techniques, suited to other case representations (e.g. texts, logical formulas or complex objects). Second, the assumption of an a priori separation between the problem and solution parts in a case is not always acceptable: for instance, in Taaable, an apple pie recipe can be seen as a solution of many problems (e.g. "Recipe of dessert with apples?" or "Recipe of pie with fruits?"). On the contrary, the knowledge at problem-solving time of the query/target problem makes it possible to make this distinction. Therefore, LAKL can benefit from this knowledge whereas EAKL cannot: studying this in detail is a future work. For example, the confidence of an adaptation rule relies on this problem-solution distinction, and using this confidence is beneficial for AKL.

References

1. Aha, D.W.: Lazy learning. Springer Science & Business Media (2013)
2. Badra, F., Cordier, A., Lieber, J.: Opportunistic Adaptation Knowledge Discovery. In McGinty, L., Wilson, D.C., eds.: 8th International Conference on Case-Based Reasoning - ICCBR 2009. Volume 5650., Seattle, United States, Springer (July 2009) 60–74 The original publication is available at www.springerlink.com.
3. Cordier, A., Dufour-Lussier, V., Lieber, J., Nauer, E., Badra, F., Cojan, J., Gaillard, E., Infante-Blanco, L., Molli, P., Napoli, A., Skaf-Molli, H.: Taaable: a Case-Based System for personalized Cooking. In Montani, Stefania, Jain, C., L., eds.: Successful Case-based Reasoning Applications-2. Volume 494 of Studies in Computational Intelligence. Springer (January 2014) 121–162
4. Craw, S., Wiratunga, N., Rowe, R.C.: Learning adaptation knowledge to improve case-based reasoning. Artificial Intelligence **170**(16-17) (2006) 1175–1192
5. d'Aquin, M., Badra, F., Lafrogne, S., Lieber, J., Napoli, A., Szathmary, L.: Case base mining for adaptation knowledge acquisition. In Veloso, M.M., ed.: IJCAI 2007, Proc. of the 20th Int. Joint Conf. on Artificial Intelligence, Hyderabad, Jan. 6-12. (2007) 750–755
6. d'Aquin, M., Nauer, E., Lieber, J.: A Factorial Study of Neural Network Learning from Differences for Regression. In: International Conference on Case-Based Reasoning, ICCBR 2022. Volume 13405 of Lecture Notes in Computer Science., Nancy, France, Springer International Publishing (September 2022) 289–303
7. Gaillard, E., Lieber, J., Nauer, E.: Adaptation knowledge discovery for cooking using closed itemset extraction. In: The Eighth International Conference on Concept Lattices and their Applications - CLA 2011, Nancy, France (October 2011)
8. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer (1999)
9. Gillard, T., Lieber, J., Nauer, E.: Improving Adaptation Knowledge Discovery by Exploiting Negative Cases: First Experiment in a Boolean Setting. In: Proc. of ICCBR 2018 - 26th International Conference on Case-Based Reasoning, Stockholm, Sweden (July 2018)
10. Hanney, K., Keane, M.T.: Learning adaptation rules from a case-base. In Smith, I., Faltings, B., eds.: Advances in Case-Based Reasoning – Proc. of the Third Eur. Workshop, EWCBR'96. LNAI 1168, Springer Verlag, Berlin (1996) 179–192
11. Jalali, V., Leake, D., Forouzandehmehr, N.: Learning and applying adaptation rules for categorical features: An ensemble approach. AI Communications **30**(3-4) (2017) 193–205
12. Lieber, J., Nauer, E., Prade, H.: Improving analogical extrapolation using case pair competence. In: Case-Based Reasoning Research and Development, 27th International Conference (ICCBR-2019), Otzenhausen, France (September 2019)
13. Lieber, J., Nauer, E., Prade, H., Richard, G.: Making the best of cases by approximation, interpolation and extrapolation. In: ICCBR 2018 - 26th Int. Conf. on Case-Based Reasoning. (2018)
14. Lieber, J., Nauer, E.: Adaptation knowledge discovery using positive and negative cases. In: ICCBR 2021 - 29th International Conference on Case-Based Reasoning, Salamanca (Virtual), Spain (September 2021)
15. Riesbeck, C.K., Schank, R.C.: Inside Case-Based Reasoning. Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey (1989) Available on line.
16. Szathmary, L., Napoli, A.: CORON: A Framework for Levelwise Itemset Mining Algorithms. Supplementary Proc. of The Third International Conference on Formal Concept Analysis (ICFCA '05), Lens, France (2005) 110–113