A Rule-Based Similarity Measure

Michèle Sebag[1] and Marc Schoenauer[2]

[1] LMS-CNRS URA 317, Ecole Polytechnique, 91128 Palaiseau France
and LRI, Université Paris-XI Orsay, 91405 Orsay France
[2] CMAP-CNRS URA 756, Ecole Polytechnique, 91128 Palaiseau France

Abstract.
An induction-based method for retrieving similar cases and/or easily adaptable cases is presented in a 3-steps process : first, a rule set is learned from a data set ; second, a reformulation of the problem domain is derived from this ruleset ; third, a surface similarity with respect to the reformulated problem appears to be a structural similarity with respect to the initial representation of the domain. This method achieves some integration between machine learning and case-based reasoning : it uses both compiled knowledge (through the similarity measure and the ruleset it is derived from) and instanciated knowledge (through the cases).

1 Introduction

In Case-Based Reasoning (CBR), the first step is retrieving cases similar to the current one among the case base. The success of the next steps, e.g. reusing the retrieved cases to achieve the current goal, and retaining from this experience, heavily depends on the quality of the retrieval phase [1]. On the other hand, the retrieving phase must be fast for it involves the overall experience of the system. Many approaches to this key problem are proposed.

The most widely-used approach is that of syntactical similarity : one computes the weighted distance between the features of the current case and that of every stored case ; the weights may be either equal or provided by the expert [16] or even optimized by genetic algorithms [15]. These approaches are restricted to propositional domains and many attempts are done to extend syntactical similarities to more complex representations.
In opposition to surface similarities are structural similarities [11]. These similarities take into account the ultimate purpose of retrieval, such as analogical transfer or adaptation [14]. But building more sophisticated similarity relationships nearly always request a strong background theory or the thorough support of the expert [3]. In the field of analogy, the structural mapping of Gentner [10] proposes an evaluation of the degree of analogy between two cases, based on a cognitive approach. In inductive learning, Bisson [6] developed a similarity measure in order to cluster and classify first order examples.

A commonly shared opinion is that much knowledge is hidden in a (good) similarity measure. Reversing this claim, we propose to compile a knowledge base into a similarity measure in a 3-step process : first, a ruleset is learned from a data set ; second, this ruleset is used to change the representation (reformulate) the problem ; third, a surface similarity with respect to the new representation appears to be a structural similarity with respect to the initial representation of the problem. The central claim of this paper is that this procedure permits to build automatically a structural similarity — since induction captures to some extent the semantics of the relation between the description of a case, and the concept it belongs to. In the meanwhile, induction may be purposely required to consider the concepts relevant for retrieving, e.g. further adaptation or classification.

This paper is organized as follows. Second section describes a change of representation based on a rule set, and defines several similarity measures, called rule-based similarities (RBS) on the reformulated problem. The coarseness of a RBS is studied with respect to the characteristics of the rule set. Third section compares RBS and weight-based similarities. A theoretical comparison focuses on the properties of idempotence [13] and invariance by translation. Experimental comparison is done on two well-studied classification problems [15]. Last section focuses on integrating machine learning and case-based reasoning through this 3-steps scheme. This scheme is briefly compared to some related works [1, 18, 2].

2 Principle

This section focuses on reformulating a problem domain given a set of rules. Whatever the initial representation of the domain, it is mapped onto a boolean space. The properties of this mapping are studied with respect to both the rule set, and the induction algorithm (learner) used to derive this ruleset from a data set.

2.1 A Rule-Based Reformulation

Let Ω be the problem domain, and let Th be a set of production rules defined on Ω.

$$Th = \{R_1, \ldots R_N\}$$

where R_i is composed of an hypothesis part and a conclusion part $R_i : H_i \rightarrow C_i$. For any tractable example E in Ω, checking whether E satisfies hypothesis H_i (in that case, it is said that E fires rule R_i, or equivalently that R_i covers E) is computable if rule R_i is to be of any use. Therefore a set of N rules defines a mapping from Ω into the boolean space of dimension N :

$$\begin{array}{ll} C_{Th} : & \Omega \rightarrow \{0,1\}^N \\ \forall E \in \Omega, & C_{Th}(E) = (R_1(E), ..R_N(E)) \\ \textit{where} & R_i(E) = 1 \text{ if E satisfies } H_i, \text{ 0 otherwise.} \end{array}$$

Let us see graphically the effects of such transformation, with the 2D space R^2 as problem domain Ω ; a conjunctive hypothesis can be thought of as a rectangle: *If x in I_1, And y in I_2, Then ...* Two rules R_1 and R_2 with hypotheses H_1 and H_2 so define a mapping from R^2 into $\{0,1\}^2$ (Fig 1.) Mapping C_{Th} is not injective, (nor is it surjective in the general case). It induces a very peculiar "topology" on the initial domain : the right and left part of rectangle H_1 have same image (H_1 $true, H_2$ $false$) which differs from the image of the central part of H_2 (H_1 $true, H_2$ $true$).

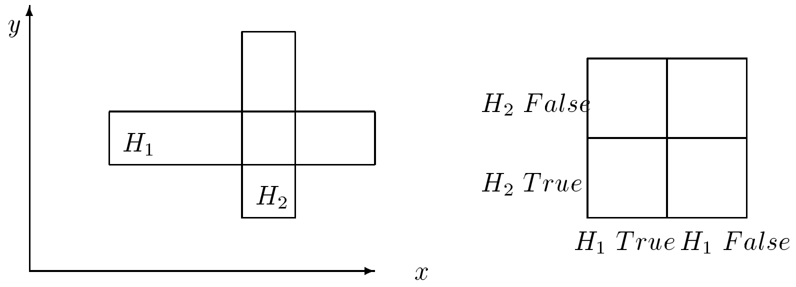

Figure 1 : A rule-based mapping from R^2 to $\{0,1\}^2$

Remark. This mapping is the inverse of the usual extensional view of examples and rules. A rule generally is considered with respect to its extension, i.e. the examples satisfying its hypothesis. Reversely, one may associate to an example the set of rules whose hypotheses are satisfied by this example. Mapping C_{Th} corresponds to the ensemblist representation of examples as subsets of rules.

2.2 Similarities on the reformulated problem

The simplest similarity between two elements in space $\{0,1\}^N$ is the number of their bits of identical value. Given two examples E_1 and E_2 in Ω, we consider the similarity of their images $C_{Th}(E_1)$ and $C_{Th}(E_2)$: the similarity $S_1(E_1, E_2)$ is set to the number of rules which are fired by both or by none of E_1 and E_2.

Definition *(similarity S_1)*
Similarity S_1 is a function defined from Ω^2 to R^+, by :

$$\forall E_1, E_2 \in \Omega, S_1(E_1, E_2) = \# \{R_i \ / \ R_i \in Th, R_i(E_1) = R_i(E_2)\}$$

where notation $\#A$ stands for the number of elements in set A.

However, similarity S_1 is not of pratical use on real-world problems because when dealing with a large set of rules there are a lot of rules which are satisfied by none of any two examples. A second similarity is then defined, where only rules effectively satisfied by both examples, are accounted for.

Definition *(similarity S_2)*
Similarity S_2 is a function defined from Ω^2 to R^+, by :

$$\forall E_1, E_2 \in \Omega, \ S_2(E_1, E_2) = \# \{R_i \ / \ R_i \in Th, R_i(E_1) = R_i(E_2) \ = 1 \ \}$$

Similarity S_2 does not make any difference between rules : any rule satisfied by two examples contributes to their similarity the same. However, some rules are more significant than others. We then weight the rules (the weight of a rule is set to the number of training examples it covers). Similarity S_3 is defined as the sum of the weights of rules fired by both examples :

Definition *(similarity S_3)*
Let w(R) denote the weight of rule R. Similarity S_3 is a function defined from Ω^2 to R^+, by :

$$\forall E_1, E_2 \in \Omega,\ S_3(E_1, E_2) = \sum_{R_i(E_1) = R_i(E_2) = 1} w(R_i)$$

2.3 Abstraction or Coarseness

It is long known that a ruleset defines a straightforward index on cases : a case may be indexed according to the rules it fires. So the question is twofold : why are these rule-based indices widely ignored in the CBR community ? When could they be useful and what are their limitations ?

The first question refers to the respective positions of CBR and machine learning. Machine learning (ML) provides an abstract view of problems domains : a rule is abstracted from (a number of) cases ; much information is purposely lost during the induction process. On the contrary, case-based reasoning encapsulates a rich and detailed knowledge in a case, as nobody knows what could be useful to handling further cases. It is argued [17] that the applicability of such instanciated knowledge is much more flexible than that of a rule (given an adaptation mechanism...).

The basis of this opposition may become clearer by answering the second question, when possibly could a rule-based similarity be of any use. Consider what happens with a rule-based index based on a concise ruleset. Assume that an example fires a unique rule. Then any two examples either fire the same rule — and they are similar ; or they do not — and they are dissimilar. To put it another way, *a concise ruleset defines a boolean similarity* on a problem domain. A boolean similarity is not of any help — except if it derives from a "perfect" ruleset ; but in that case there is no need for CBR in general... So, a flexible rule-based similarity should be based on a redundant ruleset. But redundancy is considered a defect in "classical" AI as a redundant knowledge base is hard to maintain and to evolve. In this line, ML often attempts to induce consistent and concise knowledge bases, such as encoded by decision trees [22, 5] : an example satisfies exactly one leaf in a decision tree. The discussion then can be summed up as follows : a rule gives an abstract and poor view of a case. So concise learning (especially decision-trees) does not enable to build usable rule-based indices for too much information has been lost.

The other extreme, i.e. considering equally all the available information may also be misleading. So we propose a mid-term. On one hand, redundancy is a feature quite easily tunable within a bottom-up induction algorithm (see [19, 24]) : when we consider the rules generalizing the current example, we may either retain the best rule only, i.e. the rule that covers the maximum number of training examples (and so obtain a concise ruleset), or retain all rules good enough, i.e. rules covering a significant number of training examples (and we thus obtain a redundant ruleset) [3]. On the other hand, if a rule gives some viewpoint on a case, then all the relevant flavour of a case could be captured by considering a number of viewpoints, given by a redundant set of rules. Building a usable rule-based similarity measure thus only requires to use a redundant learner.

Remark. Note that this approach applies whatever the initial formalization of the domain : it only needs this formalism to be tractable for a redundant learner. This requirement holds for propositional [9, 24] and first-order logic [6, 25].

2.4 Accounting for Adaptation

The rule-based similarity can use humanly provided rules as well as rules learned from a data set ; it can also combine both. However, the available knowledge if any is usually not sufficient to build a usable similarity.

One then has to extract a rule set from the case base. Supervised learning (classification of labelled observations, i.e. examples) is preferred to unsupervised learning (clustering of observations) because unsupervised learning generally builds a hierarchy of concepts [8], without enough redundancy to provide a rich rule-based similarity.

So the cases must be labelled to be tractable by supervised learning. Our approach, primarily motivated by classification, still applies on others problems provided that the reuse task involves adapting a finite number of plans. In such problems, a case is labelled according to the plan(s) which is (are) adapted to this case ; supervised learning then characterizes the conditions of applicability of each plan (given its "positive examples", i.e. the cases the plan applies on, and its "negative examples"). The hypothesis part of a rule then gives sufficient conditions on the features of a case for a given plan to apply. Note that there may be a number of reasons to apply a given plan : e.g. in war, retreat may be a sign of defeat or a trap. One rule (with conjunctive premises) stands for *one* context where the plan applies. Therefore, if two cases are detected similar by such a rule-based similarity, one knows that same plans apply on both examples and *for the same reasons*. Such a similarity so enables to retrieve easy-to-adapt cases.

However, it is worth noting that the rule-based similarity definitions do not depend on the rules conclusions ; the learner is free to consider any concept set by the expert, and to build inconsistent rules. Let us assume that two examples both fire two inconsistent rules ; they thus share an "ambiguity" of the ruleset. This ambiguity should not, according to us, result in a weaker link (than it would, if they have fired consistent rules). The only implicit assumption is that the concept to learn, the considered training set and the learner are such that :

[3] From a practical point of view [24], we call *density of an example* the maximum number of training examples covered by a rule generalizing this example. Let ρ denote the redundancy rate, then all rules covering a number of examples greater or equal than the density divided by ρ are retained. Taking $\rho = 1$ thus leads to a concise rule set ; more and more rules are retained as ρ increases.

• if we consider the decision support resulting from these concept, training set and learner, then
• if the same arguments, pro or cons, are encountered by this decision support when handling two cases,
• then, these cases are similar with respect to the CBR goal.

3 Comparing Rule-based and Weight-Based Similarities

This section compares some theoretical properties of rule-based and weight-based similarities. An empirical comparison is run on two problems well-studied in the machine learning literature.

3.1 Some Properties of Weight-Based Dissimilarities

Let the space domain E be described by K attributes $x_1, ..x_K$. A weight-based similarity S on space E is usually based on a weighted distance D on E, with

$$\forall e_1, e_2 \in E, S(e_1, e_2) = A - B \times D(e_1, e_2) \ ; \ D(e_1, e_2) = \sum_{i=1}^{K} w_i \times d_i(x_i(e_1), x_i(e_2))$$

where w_i is the weight defined on attribute x_i and d_i a distance defined on the domain of x_i[4]. On qualitative domains, $d_i(v_1, v_2)$ usually takes value 1 if v_1 and v_2 are distinct, 0 otherwise. On numerical domains, d_i is the usual distance. Weights are usually supplied by the expert.

The properties of a weight-based similarity (WBS) are linked to that of the underlying distance. A WBS takes its maximal value for a pair (e, f) iff $e = f$ (assuming that no attribute has a null weight). This means first that the point nearest to a given point is itself ; and second, that all points are similar to themselves with the same strength. The latter property, called *idempotence property* by [13], may be unwanted from a knowledge acquisition standpoint. For instance, if attribute *color* takes its value in *red, blue, other-color*, objets sharing the property of being both *blue* should be more similar, everything else being equal, than two objects sharing the property of being of an *other-color* [21].

WBS, like the usual numerical distances they are based on, are invariant by numerical translation : if X and Y stand for vectors in R^K[5],

$$if\ d(X, Y) = \sum_{i=1}^{K} |X_i - Y_i|,\ then\ d(X + H, Y + H) = d(X, Y)$$

More generally let τ_V be an application defined on domain E by translating every numerical attribute x_i of an amount of V_i, then one has $S(e_1, e_2) =$

[4] Euclidean-like weight-based dissimilarities are frequently used too ; one then has $D(e_1, e_2) = (\sum_{i=1}^{N} (v_i d_i(e_1, e_2))^2)^{1/2}$

[5] Similarly, if $d(X, Y) = (\sum_{i=1}^{K} |X_i - Y_i|^2)^{1/2}$, then $d(X + H, Y + H) = d(X, Y)$.

$S(\tau_V(e_1), \tau_V(e_2))$. Similarly, if τ is an application defined on E by permuting the values of any qualitative attribute x_j, the invariance property with respect to τ holds. This entails that local modifications of WBS are impossible : modifying the similarity between any pair of cases will have side-effects on the whole space. This effect of "uniformity" may also be unwanted from a knowledge acquisition standpoint : as everybody working with experts knows, modifying the same detail on two cases may lead the expert to evaluate completely differently their similarity.

3.2 Some Properties of Rule-Based Similarities

We distinguish among similarities S_1, S_2 and S_3.

The idempotence property holds for S_1 : the similarity between a case and itself always amounts to the number of rules in the ruleset. But idempotence does not hold neither for S_2 nor for S_3. For S_2, the similarity between a case and itself amounts to the number of rules fired by this example. The number of rules fired by an example depends on the density of counter-examples in the region the example lies in : if there is a lot of counter-examples, the induction process is so constrained that few solutions (i.e. terms covering this example without covering counter-examples) are found. Hence, the farther a case is from counter-examples (according to the induction goal set, see 2.4) the more similar it is to itself.

For S_3, the similarity between a case and itself amounts to the sum of the weights of the rules fired, where the weight of a rule is the number of examples covered by this rule. The similarity between a case and itself thus increases with the number of examples and as the number of counter-examples decreases in the same region.

RBSs do not satisfy the invariance property. Consider the two following pairs of cases :

	Color	*Shape*	*Size*
e_1	Green	Circle	Small
e_2	Blue	Circle	Small

	Color	*Shape*	*Size*
f_1	Green	Triangle	Medium
f_2	Blue	Triangle	Medium

Figure 2 : Two pairs of examples

Case e_1 differs from e_2 in that that the former is green while the latter is blue. Same difference is observed between f_1 and f_2. Consider now rule

$$R : If\ (Color\ =\ Green)\ and\ (Shape\ =\ Circle),\ Then\ Green\_Pea$$

Rule R makes a difference between e_1 and e_2 because it is fired by e_1 and not by e_2 ; but it does not make any difference between f_1 and f_2, as it is fired by none of them. On the other hand f_1 and f_2 are respectively the images of e_1 and e_2 by permuting the values of attribute *shape* and *size* (e_1 and e_2 are both *circle*s of *small* size, while f_1 and f_2 are *medium triangle*s). Hence RBS are not invariant by permuting the values of a qualitative attribute (or translating a numerical attribute).

In summary RBS are more flexible than WBS : it recognizes the significance of particular combinations of factors, as requested by Ashley [4], rather than considering each feature independantly.

3.3 Experimental Validation

Two classification problems fitting within attributes-values formalism, are considered. The first one (*Iris*) consists of 150 examples divided into 3 classes and described by 4 attributes. The second one (*Glass*) is composed of 214 examples divided into 6 classes and described by 9 attributes.

The reference results are those of J. Kelly and L. Davis [15]. The data set is divided into a training set (4/5 of the data) and a test set. This selection is done at random, except that the classes distributions in the training set are same as in the total data set. The result is the percentage of examples in the test set correctly identified. Legend *KNN* denotes a classical K-nearest neighbors method using a weight-based similarity with equal weights. Legend *GA-KNN* denotes a K-nearest neighbors method using a weight-based similarity whose weights are optimized by genetic algorithms.

	KNN	*GA-WKNN*
IRIS	90	94 - 93
GLASS	58	60 - 62

Table 1 : KNN and GA-KNN Results

Our approach is denoted *RKNN*, for Rule-based K Nearest Neighbors. Rules are learned from a training set including 2/5 of the data. The case base comprises 4/5 of the data, including the training set ; the test set is the remaining 1/5. Rules are learned by using a star-like generalization algorithm [24], that allows for tuning the rules redundancy.

The predictive results obtained on the test set are averaged over five independant selections of the training and test sets. Here are the results obtained by *RKNN* for similarities S_1, S_2 and S_3. The redundancy rate (cf 2.3) ranks from 1 (concise rules) to 5 (this corresponds to multiplying the rules number by about 2.5 in the considered problems).

	RKNN														
	red. 1			red. 2			red. 3			red. 4			red. 5		
	S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3
IRIS	92	91	91	93	93	93	93	93	93	92	91	91	91	91	91
GLASS	65	64	64	62	68	67	64	68	68	56	67	67	52	70	70

Table 2 : RKNN Results

These results show that rule-based similarities S_1, S_2 and S_3 behave in different ways depending on the data at hand.

On the problem *Iris*, our results are very similar to those obtained by weight-based similarity (with weights optimized by using genetic algorithms) ; and S_1, S_2 and S_3 nearly lead to the same results. This is connected to the fact that there is no matter of distribution in these data (all classes are equally represented) ; so there is no difference between S_2 (rules fired by both examples contribute to their similarity) and S_3 (same as in S_2, but the amount of contribution depends on the number of training examples covered by a rule). The only advantage of our approach regarding the well distributed *Iris* problem and compared to *GA-KNN* is to lessen the computational cost (about 10 minutes on a Symbolics Ivory-based Lisp machine, against 10 seconds on a HP 710 work station).

On the ill-distributed problem *Glass*, similarity S_1 slightly outperforms similarity S_2 and S_3 when redundancy is low ; the inverse is true when considering redundant rulesets. This can be explained as follows. First and overall, S_1 becomes less and less accurate when redundancy increases, because any two examples, no matter how different they are, will *not* fire a number of rules in a redundant ruleset. In opposition, S_2 and S_3 improve as expected. Last, on the *Glass* problem, our results are significantly better than the reference results (from 5 to 8 points).

4 Related Works

This section describes some integrations of CBR and machine learning (ML) and discusses the proposed scheme. Unfortunately, space limitations prohibit discussing the reformulation aspects of our work with respect to [12] and [23] among others.

4.1 Some Integrations of CBR and ML

It is generally acknowledged that coupling case-based and induction-based techniques could lead to more performant than standalone systems. As a matter of fact, some learning is already embedded into CBR : the last phase of a CBR process is retaining from the current experience. However, the induction used in "classical" CBR differs from that of machine learning, with respect to its output representation (for instance similarity versus rules) as well as the optimization criteria : CBR is much more reluctant to drop information than ML. Our purpose is not to draw any (controversial) borderline, but to limit our discussion to considering some integrations of CBR and "classical" machine learning, without pretending to exhaustivity. The couplings can be divided into two categories: those using CBR to supplement an insufficient KB, and those using ML to improve a CBR system.

In the first category falls the INRECA project [18]. This integration scheme is mainly motivated by handling incompletely described examples. A concise knowledge, such as built in a decision tree, has difficulties in dealing with unknown values and explaining the eventual decisions. The idea is then to use the decision tree as a pre-retrieval phase for a CBR process ; when a question (a node in the decision tree) is answered *Unknown*, then the CBR process is run with the subset of cases meeting the previous nodes. Here, induction is used to pre-select cases. A somewhat similar scheme is proposed by [2] ; the difference is that pre-selection is data-driven in [18], while it is conclusions-driven in [2]. A weakness of this scheme is that it works with *incomplete but correct* knowledge; otherwise, the preselection may biase the search toward irrelevant regions of space.

Another work falling in the first category is the tuning of rules by cases proposed by Nakatani and Israel [20]. This scheme deals with domains where domain theories both are available and suffer many exceptions. The idea is to attach to rules a CBR process ; a case stores the description of a rule firing that did not lead to a satisfactory solution, plus the hypothesized explanations, plus some alternative solutions proposed by the user ; when the current context fires this rule, the CBR process is run to check whether the current case meets the exception environments. The exceptions handling thus is stored as a case base ; this case base may be used off-line to refine the domain theory.

In the second category is the work of Aamodt [1]. The proposed integration of ML and CBR tackles open domains with incomplete and/or incertain theory. The same formalism — a frame network — is used to represent all kinds of knowledge, ranking from general to case-specific knowledge. This frame network is abductively used (because of its incompleteness and uncertainity) ; the difference between reasoning from general knowledge or from cases then vanishes. All phases of CBR are performed as 3-steps processes : *activation*, where the input information is propagated along the network ; *explanation*, which stands both for summarizing the activations acting on nodes, and for explaining this summary[6] ; *focusing*, that handles external constraints and performs some consensus or choice among the solutions recommended by the explanation functionnality. For instance, the learning phase notices the nodes or structures activated during retrieval and reuse (activation step) ; it detects (and justifies) whether new structures are to be created ; the case findings are updated (generalized) if required by justifications or by the user (explanation step) ; finally, the structures to create are put in a new frame if necessary (focusing step). The step most apparented to machine learning (inductive logic programming) lies in detecting whether and which new structures are to be created. Unfortunately, this step is too succintly described to see how it is related to ILP. This work is described by the author as a continuous knowledge maintenance process ; it beautifully emphasizes the continuity that exists between knowledge-based and case-based reasoning.

[6] Explanation is a key feature, as it is the only way for the expert to control (debug) the system.

4.2 The proposed integration

The scheme we propose also fits in the second category : the ML component is used to achieve retrieving of similar and/or adaptable cases. One part of the data is used as training set to learn rules ; from these rules then derives a similarity measure, that is used together with the case base for CBR (Fig. 3).

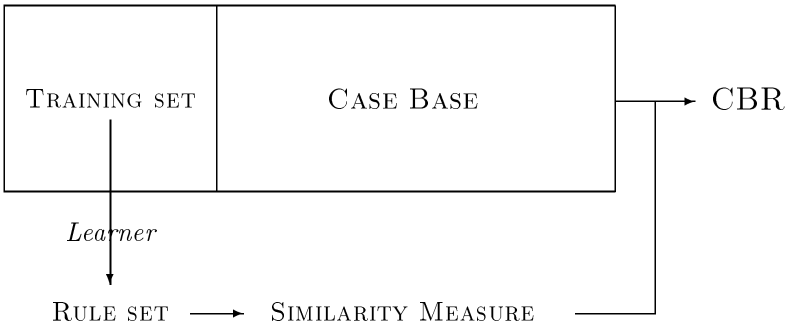

Figure 3 : The proposed integration of ML and CBR

This scheme achieves a 2-gears learning : a slow, expensive learning is done off-line ; cases are compiled into rules and from rules derives a similarity measure ; but in opposition to machine learning, cases remain available in the case base. Besides this "long-term learning", a fast, on-line learning consists in just enriching the case base.

Moreover, this scheme takes advantages from both a machine learning and a CBR standpoint. From the CBR standpoint, it provides a *structural similarity without requiring any background knowledge*[7] : a majority of retrieved cases is guaranteed to be relevant with respect to the current classification or adaptation purpose according to the induction goal set to the learner (the relevance of the ruleset of course reflects the quality of the training set). On the other hand, this approach does not incur the whole adaptation cost : the current case is matched against only the rules hypotheses. From the machine learning standpoint, it decreases a lot the induction cost : using the case base allows for correcting most of the errors in the ruleset, as in the "tuning rules by cases" effect [20]. When the comparison is possible, learning from a few examples and using a large case base is equivalent to learning from significantly more examples.

A critical point is deciding when a next long-term learning should be run. By analogy with human beings, learning is necessary when the error rate of the system increases, or does not decrease as expected. Note that induction can consider the initial description of the cases, as well as the index derived from previous rules. In the latter case, what is learned is overcoming the defects and

[7] Of course providing the learner with background knowledge can ease the induction task and improve its results ; but it is not a *sine qua non* condition. Therefore we claim that this similarity building does not demand a strong interaction with the expert, such as required in Protos [3] for instance.

biases of the previous index.

5 Conclusion and Perspectives

This paper presents a way to using inductive learning to provide CBR with a similarity measure. This similarity compiles the relationships linking the description of a case to the CBR goal, be it either classification or adaptation. This way, a structural similarity can be built in the absence of background knowledge. Rule-based similarities are not invariant by translating numerical attributes or permuting qualitative attributes, in opposition to weight-based similarities. They thus enable different behaviors in different regions of the problem domain.

The opposition between "classical" induction, delivering abstract information, and CBR storing rich and instanciated knowledge, is overcome by using redundant induction. A redundant set of rules provides a number of different viewpoints on a case, by which one can recollect the "flavour" of the case. This method applies whatever the initial representation of the domain is ; it only requires a redundant learner to work within this representation.

It is acknowledged that there are different phases in the life of a CBR system ; the desirable properties of a similarity measure depend on the current phase of the system. So our further research will focus on studying the similarity properties (symmetry, triangular inequality,..) adapted to the different phases, and accordingly modifying rules-based similarities.

References

1. A. Aamodt, *Explanation-Driven Retrieval, Reuse and Retain of Cases*, in [7].
2. S.K. Bamberger, K. Goos, *Integration of CBR and Inductive Learning Methods*, in [7].
3. Bareiss R., *Exemplar-based knowledge acquisition.* Boston, MA, Academic Press.
4. Bareiss R. et al., *Panel discussion on indexing vocabulary*, DARPA CBR Workshop 1989, Morgan Kaufman.
5. F. Bergadano, A Giordana, L. Saitta, *Automated Concept Acquisition in Noisy Environments*, IEEE Trans on Pattern Analysis and Machine Intelligence, PAMI-10, pp 555-578, 1988.
6. G. Bisson, *KBG A Knowledge Based Generalizer.* 7^{th} ICML, R. Porter B. Mooney Eds, Morgan Kaufmann, 1990.
7. Proceedings of 1^{st} EWCBR, M. Richter, S. Wess, K.-D. Althoff, F. Maurer Eds, University of Kaiserslautern, Germany, nov 1993.
8. D. Fisher, *Cobweb : Knowledge acquisition via Conceptual clustering*, Machine Learning Vol2, 1987.
9. J.G. Ganascia, *AGAPE et CHARADE, deux techniques d'apprentissage symbolique appliquées à la construction de bases de connaissances*, Thèse d'Etat, 1987, Orsay.
10. D. Gentner, *Structure Mapping : A Theoretical Framework for Analogy.* Cognitive Science, 1983, Vol 7 N 2, pp 155-170.
11. K.J. Holyoak, L. Koh, *Analogical Problem Solving : Effects of Surface and Structural Similarity in Analogical Transfer*, Midwestern Psychological Association Ed, 1986.

12. B. Indurkhya, *On the Role of Interpretive Analogy in Learning*. Algorithmic Learning Theory, S. Arikawa et al. Eds, Springer Verlag 1990, pp 174-189.
13. K.P. Jantke, S. Lange, *Case-Based Representation and Learning of Pattern Languages*, in [7].
14. M.T. Keane, *Analogical Problem Solving*, Chichester, Ellis Horwood 1988.
15. J. D. Kelly, L. Davis, *A Hybrid Genetic Algorithm for Classification*, Proc. IJCAI 1991, J. Mylopoulos, R. Reiter Eds, Morgan Kaufmann Publishers, pp 645-650.
16. D. Kibler, D. A. Aha *Learning representative exemplars of concepts: An initial case study* Proc. of the 4^{th} IWML, reprinted in Readings in Machine Learning, J.W. Shavlik T. G. Dietterich, Morgan Kaufman 1990, pp 108-115.
17. J.L. Kolodner, *Extending problem solver capabilities through case-based inference*, Proceedings 4^th Workshop on ML, UC Irvine 1987.
18. M. Manago, K-D Althoff, E. Auriol, R. Traphoner, S. Wess, N. Conruyt, F. Maurer, *Induction and Reasoning from Cases*, in [7].
19. R.S. Michalski, *A theory and methodology for inductive learning* , Machine Learning: An Artificial Intelligence Approach, I, R.S. Michalski, J.G. Carbonnell, T.M. Mitchell Eds, Springer Verlag, (1993), p 83-134.
20. Y. Nakatani, D. Israel, *Tuning Rules by Cases*, in [7].
21. Nicolas J, Lebbe J, Vignes R, *From Knowledge to Similarity*, Numeric-Symbolic Learning and Data Analysis, Diday Ed, Nova Sciences, 1991.
22. R. Quinlan, R.M. Cameron-Jones, *FOIL ; A Midterm Report*, ECML 93, P.B. Bradzil Ed, Springer-Verlag, 1993, pp 3-20.
23. S. J. Russell, *The Use of Knowledge in Analogy and Induction*, Pitman, London, 1989.
24. M. Sebag M. Schoenauer, *Incremental Learning of Rules and Meta-Rules.* 7^{th} ICML, R. Porter & B. Mooney Eds, Morgan Kaufmann, 1990.
25. M. Sebag, *A Constraint-based Induction Algorithm in FOL*, ICML-94, W. Cohen & H. Hirsh Eds, Morgan Kaufmann, 1994.

This article was processed using the LaTeX macro package with LLNCS style