Preference-Based CBR: First Steps Toward a Methodological Framework

Eyke Hüllermeier and Patrice Schlegel

Department of Mathematics and Computer Science
Marburg University, Germany
{eyke,schlegel}@mathematik.uni-marburg.de

Abstract. Building on recent research on preference handling in artificial intelligence and related fields, our general goal is to develop a coherent and universally applicable methodological framework for CBR on the basis of formal concepts and methods for knowledge representation and reasoning with preferences. A preference-based approach to CBR appears to be appealing for several reasons, notably because case-based experiences naturally lend themselves to representations in terms of preference relations, even when not dealing with preference information in a literal sense. Moreover, the flexibility and expressiveness of a preference-based formalism well accommodate the uncertain and approximate nature of case-based problem solving. In this paper, we make a first step toward a preference-based formalization of CBR. Apart from providing a general outline of the framework as a whole, we specifically address the step of case-based inference. The latter consists of inferring preferences for candidate solutions in the context of a new problem, given such preferences in similar situations. Our case-based approach to predicting preference models is concretely realized for a scenario in which solutions are represented in the form of subsets of a reference set. First experimental results are presented to demonstrate the effectiveness of this approach.

1 Introduction

Despite its great practical success, work on the theoretical foundations of CBR is still under way, and a coherent and universally applicable methodological framework is yet missing. The *CBR cycle* proposed by Aamodt and Plaza [1] is a commonly accepted process model, which nicely illustrates the main aspects of the case-based problem solving paradigm. Likewise, the metaphor of *knowledge containers*, introduced by Richter [2], provides a general framework for the structuring of knowledge in CBR. However, both are high-levels models and still rather far from the conceptual realization and implementation of a case-based problem solver. On the other extreme, many CBR systems have been designed for solving concrete problems. These, however, are mostly tailored for a specific purpose and not easily applicable to a wider range of problems.

In-between these two extremes, there is arguably space for developing CBR *methodologies* [3]. On the one hand, a CBR methodology should be sufficiently

general and abstract, so as to allow for the development of generic algorithms, for analyzing formal properties, proving theorems, etc. On the other hand, it should also be sufficiently concrete, so as to support the development of specific applications. To make this idea more tangible, consider as an analogy the formalism of graphical models, by now an established methodology for the design of probabilistic expert systems [4]. This class of models disposes of a formal theory and generic algorithms, but also tools for supporting the design of models for concrete applications.

In [5], a general *constraint-based framework* of CBR has been developed. Roughly speaking, the core assumption of CBR, suggesting that similar problems have similar solutions, is formally interpreted as a constraint: given a new query problem to be solved in conjunction with previous experience in the form of a solution to a similar problem, it restricts the set of candidate solutions of the query. Starting with the classical view of a constraint as a subset of feasible elements of a relation, more flexible variants of this approach have later been developed on the basis of different frameworks of approximate reasoning and reasoning under uncertainty. Taking such models as a point of departure, case-based reasoning can be realized, respectively, as constraint-based, probabilistic, or fuzzy logic-based reasoning in a formally sound way.

In this paper, we make a first step toward an alternative methodological framework for case-based reasoning on the basis of formal concepts and methods for knowledge representation and problem solving with *preferences*. The topic of preferences has recently attracted considerable attention in artificial intelligence (AI) research and plays an increasingly important role in several AI-related fields, including, e.g., agents, constraint satisfaction, decision theory, planning, machine learning, and argumentation [6–8]. Preference-based methods are especially appealing from an AI perspective, notably as they allow one to specify desires in a declarative way, to combine qualitative and quantitative modes of reasoning and to deal with inconsistencies and exceptions in a quite flexible manner. Indeed, a preference can be considered as a relaxed constraint, which, if necessary, can be violated to some degree.

The important advantage of an increased flexibility of a preference-based problem solving paradigm is nicely explained in [9]: "Early work in AI focused on the notion of a goal—an explicit target that must be achieved—and this paradigm is still dominant in AI problem solving. But as application domains become more complex and realistic, it is apparent that the dichotomic notion of a goal, while adequate for certain puzzles, is too crude in general. The problem is that in many contemporary application domains ... the user has little knowledge about the set of possible solutions or feasible items, and what she typically seeks is the best that's out there. But since the user does not know what is the best achievable plan or the best available document or product, she typically cannot characterize it or its properties specifically. As a result, she will end up either asking for an unachievable goal, getting no solution in response, or asking for too little, obtaining a solution that can be substantially improved."

Our claim is that the above insights do not only apply to AI in general but to CBR in particular. In fact, as will be argued in more detail below, case-based experience can be modeled in terms of preference information in a quite convenient way and, moreover, case-based inference can be realized quite elegantly in the form of preference processing. As pointed out in the above quotation, a key advantage in comparison to a constraint-based approach, as developed in [5], is an increased flexibility and expressiveness, which appears to be especially advantageous for CBR. To some extent, these points were already put forward in [10], albeit within a much more narrow scope.

The remainder of the paper is organized as follows: In the next section, the main ideas of our approach to preference-based CBR are outlined in an informal way. A formal model of a core part of preference-based CBR, namely the inference step responsible for predicting a "contextualized" preference relation on the solution space, is then introduced in Section 3. Section 4 is devoted to an experimental study, in which we seek to demonstrate the general feasibility of case-based learning on the basis of preference information. The paper ends with some concluding remarks and an outlook on future work in Section 5.

2 Main Ideas and Basic Setting

Even though several generalizations have been proposed in recent years, experience in CBR is most commonly represented in the form of problem/solution tuples $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbf{X} \times \mathbf{Y}$, where $\boldsymbol{x}$ is an element from a problem space $\mathbf{X}$, and $\boldsymbol{y}$ an element from a solution space $\mathbf{Y}$. Despite its apparent simplicity, this representation is quite expressive, especially since $\mathbf{X}$ and $\mathbf{Y}$ can be arbitrarily complex spaces. Yet, upon closer examination, it also exhibits some disadvantages and principle limitations, both from a knowledge acquisition and reuse point of view.

- *Existence of correct solutions*: First, the representation assumes the existence of a "correct" solution for each problem, and implicitly even its uniqueness. In many application domains, this assumption is not tenable. Take the cooking domain as an example: If the problem is to prepare a vegetarian pasta meal for two persons, there is definitely not a single "correct" recipe. Instead, there will be many possible alternatives, maybe more or less preferred by the user.
- *Verification of optimality*: Even if the existence of a single correct solution for each problem could be assured, it will generally be impossible to verify the optimality of the solution that has been produced by a CBR system. Consequently, a solution $\boldsymbol{y}$ may only be a suboptimal solution to a problem $\boldsymbol{x}$, so that storing and later on reusing the case $(\boldsymbol{x}, \boldsymbol{y})$ can be misleading. This problem is less critical, though does not dissolve, if only "acceptable" instead of optimal solutions are required.
- *Loss of information*: Storing only a single solution $\boldsymbol{y}$ for a problem $\boldsymbol{x}$, even if it can be guaranteed to be optimal, may come along with a potential loss of information. In fact, during a problem solving episode, one typically tries or at least compares several candidate solutions. Retaining a single one then

captures the information that it was the best candidate. The potentially useful piece of experience that, for example, a second candidate $\boldsymbol{y}'$, despite being worse than $\boldsymbol{y}$, was still better than a third alternative $\boldsymbol{y}''$ is lost, however. On the other hand, if a problem solving process was canceled before a provably optimal solution could be found, there is in principle no case to be stored in the case base. Again, however, retaining the piece of knowledge that a candidate $\boldsymbol{y}$ was not acceptable, or that a candidate $\boldsymbol{y}'$, even if suboptimal, was at least better than $\boldsymbol{y}''$, could be useful.

- *Limited guidance*: From a reuse point of view, a retrieved case $(\boldsymbol{x}, \boldsymbol{y})$ only suggests a single solution, namely $\boldsymbol{y}$, for a query problem $\boldsymbol{x}_0$. Thus, it does not imply a possible course of action in the case where the suggestion fails: If $\boldsymbol{y}$ is not a good point of departure, for example since it cannot be adapted to solve $\boldsymbol{x}_0$, there is no concrete recommendation on how to continue.

2.1 Preference-based Knowledge Representation

To avoid these problems, we propose a *preference-based* approach to representing and processing experiences in CBR. The basic idea is to replace experiences of the form "solution $\boldsymbol{y}$ (optimally) solves problem $\boldsymbol{x}$" by information of the form "$\boldsymbol{y}$ is better (more preferred) than $\boldsymbol{y}'$ as a solution for $\boldsymbol{x}$". More specifically, the basic "chunk of information" we consider is symbolized in the form $\boldsymbol{y} \succeq_{\boldsymbol{x}} \boldsymbol{y}'$ and suggests that, for the problem $\boldsymbol{x}$, the solution $\boldsymbol{y}$ is at least as good as the solution $\boldsymbol{y}'$. This type of knowledge representation obviously overcomes the problems discussed above. As soon as two candidate solutions $\boldsymbol{y}$ and $\boldsymbol{y}'$ have been tried as solutions for a problem $\boldsymbol{x}$, these two alternatives can be compared and, correspondingly, a strict preference in favor of one of them or an indifference can be expressed (recall that, from a weak preference relation $\succeq$, a strict preference $\succ$ and an indifference $\sim$ are derived as follows: $\boldsymbol{y} \succ \boldsymbol{y}'$ iff $\boldsymbol{y} \succeq \boldsymbol{y}'$ and $\boldsymbol{y}' \not\succeq \boldsymbol{y}$, and $\boldsymbol{y} \sim \boldsymbol{y}'$ iff $\boldsymbol{y} \succeq \boldsymbol{y}'$ and $\boldsymbol{y}' \succeq \boldsymbol{y}$). To this end, it is by no means required that one of these solutions is optimal. It is worth mentioning, however, that knowledge about the optimality of a solution $\boldsymbol{y}^*$, if available, can be handled, too, as it simply means that $\boldsymbol{y}^* \succ \boldsymbol{y}$ for all $\boldsymbol{y} \neq \boldsymbol{y}^*$. In this sense, the conventional CBR setting can be considered as a special case of preference-based CBR.

The above idea of a preference-based approach to knowledge representation in CBR also suggests a natural extension of the case retrieval and inference steps, that is, the recommendation of solutions for a new query problem: Instead of just proposing a single solution, it would be desirable to predict a *ranking* of several (or even all) candidate solutions, ordered by their (estimated) degree of preference:

$$\boldsymbol{y}_1 \succeq_{\boldsymbol{x}} \boldsymbol{y}_2 \succeq_{\boldsymbol{x}} \boldsymbol{y}_3 \succeq_{\boldsymbol{x}} \cdots \succeq_{\boldsymbol{x}} \boldsymbol{y}_n \tag{1}$$

This is indeed comparable to an information retrieval scenario, such as web search, where normally not only a single solution is shown to the user, but instead a complete list of potential matches. Thus, the last problem mentioned above, namely the lack of guidance in the case of a failure, can be overcome.

As a side remark, we note that a kind of ranking of solutions can in principle also be obtained in the classical approach to CBR, namely by ordering the

Table 1. Exemplary preferences of different persons regarding four coffee drinks.

Anne:	Cappuccino	$\succ$ Espresso	$\succ$ Latte	$\succ$ Americano
Lisa:	Cappuccino	$\succ$ Latte	$\succ$ Espresso	$\succ$ Americano
Peter:	Americano	$\succ$ Latte	$\succ$ Espresso	$\succ$ Cappuccino
Paul:	Latte	$\succ$ Americano	$\succ$ Espresso	$\succ$ Cappuccino

solutions associated with the k cases in the case base which are most similar to the query problem. However, apart from lacking a formal foundation, there is a very important difference to our approach. In fact, our fundamental assumption is that, with each problem, one can (at least theoretically) associate a preference order over the set of potential solutions, instead of just a single (correct) solution. Needless to say, this order will normally not coincide with the ordering obtained by sorting the presumably best solutions (top-choices) for similar problems.

As an illustration, consider the simple example in Table 1, where four persons are listed in decreasing order of their similarity to the query person, say, Mary. For each person, the preferences regarding four types of coffee drinks are given in terms of a total order. To predict Mary's preferences on the four alternatives, one could, for example, simply adopt the preference relation of the most similar person, which is Anne, or aggregate the preference relations of all persons (e.g., by a simple Borda count, which yields Latte $\succ$ Cappuccino $\succ$ Espresso $\sim$ Americano). In any case, the result will be different from the order obtained by sorting the top-choices of the four people (Cappuccino, Cappuccino, Americano, Latte). In fact, the example also shows that this approach will normally not even lead to a proper ranking: While some alternatives may never occur, since they are never ranked first (like Espresso), others may occur multiple times (Cappuccino appears on the first two positions). In any case, the example makes clear that sorting solutions in the classical CBR setting does in general not yield a proper preference relation (ranking) on the set of all candidate solutions.

2.2 Important Issues in Preference-Based CBR

The approach outlined so far is overly simplistic and needs to be refined in several respects. Important problems to be addressed include the following:

- How to represent, organize and maintain case-based experiences, given in the form of preferences referring to a specific context, in an efficient and effective way?
- How to select and access the experiences which are most relevant in a new problem solving situation?
- How to combine these experiences and exploit them to infer a solution or, more generally, a preference order on a set of candidate solutions, for the problem at hand?

Regarding the notion of a "problem", we like to mention that the problem description in CBR can be quite general. In particular, in addition to properties of the actual problem itself, it may contain further information, for example about a user, so that the term "context" would perhaps be even more appropriate. From a preference point of view, this is very important, since different users, e.g., members of a web community, may have different preferences. For instance, it makes a great difference whether a culinary preference is expressed by a vegetarian or by a non-vegetarian. In general, we assume the problem to be specified by a finite number of attributes. The domain of an attribute can simply be an unordered or totally ordered set (e.g., categorical or numeric attribute), but can also have a hierarchical structure. In the cooking domain, for example, attribute values are often organized in the form of taxonomies (allowing for the specification of values at different levels of abstraction).

2.3 Structure of the Solution Space

A solution in CBR can be as simple as a single value (like in CBR for classification and regression tasks [11, 12]) but may also appear as a complex object assembled from a number of basic components. Needless to say, the concrete structure of the solution space will be important from a methodological point of view, because different types of problem solving will call for different methods. The following (non-exhaustive) list of problem types and related solution spaces may be envisioned in increasing order of complexity.

- A solution space $\mathbf{Y}$ is specified by the conventional attribute-value representation, i.e., the description of a solution in terms of a fixed number of attribute values (numerical, categorical, etc.). This type of representation is natural, commonly used, and even relevant for structured representations, which can often be mapped to flat feature vectors in a reasonable way. For example, if the nutritional value of a dish is of main interest, a recipe can be mapped to an amount of protein, vitamins, etc.
- Spaces of the form $\mathbf{Y} = 2^C$, where C is a finite set of labels. Thus, a solution is a subset of C. Despite its simplicity, this specific structure is quite general and indeed relevant for many applications. For example, in its most basic form, a cooking recipe is simply represented by a set of ingredients, that is, by a subset of the set C of all potential ingredients (perhaps on different levels of abstraction as specified by an underlying taxonomy).
- A combination of the two previous scenarios is of the form $\mathbf{Y} = 2^D$, where D is a set of objects characterized in terms of an attribute-value representation. For example, instead of just knowing the name of an ingredient, it is now possible to capture some of its properties. Thus, each solution is a subset of objects, where each object is in turn a feature vector. This type of representation of alternatives has recently been advocated in [13], especially from a preference handling point of view.
- Another generalization of the first scenario is a solution space $\mathbf{Y}$ based on a representation in terms of a feature vector with set-valued attributes, i.e.,

in which each attribute can assume several values of an underlying domain simultaneously instead of only a single one. For example, a recipe could be described, amongst others, by an attribute `SideDish` with underlying domain $\{\texttt{potatoes}, \texttt{rice}, ...\}$. Most recipes will include exactly one side dish, but dishes with no side dish or more than one do of course exist.
- Solution spaces $\mathbf{Y}$ in the form of a specific class of *graphs*, another generic data structure that can be used for modeling purposes in a rather flexible way. In particular, in addition to the components themselves, it allows one to capture relationships between them. This type of solution space has recently been considered in CBR in connection with workflows [14].

It is worth mentioning that the "construction" of a solution in the first four scenarios can in principle be reduced to solving a fixed number of "prediction" problems, namely assigning a value to each attribute (in the case of multi-valued attributes, prediction comes down to solving a multi-label classification problem [15]). However, it is also important to recognize that these problems are not independent of each other, due to interactions and interdependencies between the attributes, so this simplistic approach is likely to produce suboptimal results.

Finally, let use note that the actual output space on which a preference-based CBR system is operating, at least implicitly, is not given by a solution space $\mathbf{Y}$ itself, but instead by $\mathfrak{P}(\mathbf{Y})$, namely the class of all preference structures on $\mathbf{Y}$. These spaces may become extremely complex, and will therefore not be dealt with in an explicit way.

3 Case-Based Inference

Leaving questions of problem representation, case base organization, case retrieval, etc. (essentially raised by the first two items in Section 2.2) aside, our focus in this paper is on case-based inference (the third item). Given a new query problem $\boldsymbol{x}_0$, standard case-based inference starts by retrieving a subset of (presumably) most relevant cases from the case base, and then proceeds by combining, in one way or another, the solutions of these problems into a candidate solution for $\boldsymbol{x}_0$. The type of aggregation procedure which is applied to this end strongly depends on the structure and representation of solutions, and on the type of preference relation defined on the solution space. In this regard, recall the main scenarios (types of solution spaces) distinguished above.

It is important to recall that problems are not associated with single solutions but rather with preferences over solutions, that is, with elements from $\mathfrak{P}(\mathbf{Y})$. Consequently, we have to consider the problem of combining the preferences associated with the nearest neighbors of the query $\boldsymbol{x}_0$ into a preference relation on candidate solutions. Ideally, such a relation is given in the form of a total order (1), though depending on the completeness of the information at hand, this will of course not always be possible. Roughly, the problem can be stated as follows: Given a set of preferences on candidate solutions coming from a set of relevant problems (associated with corresponding similarity degrees), find a

global preference relation on these candidates which is as consistent with these preferences as possible.

3.1 Case-based Inference as Probability Estimation

To solve this problem in a theoretically sound way, we approach it as a statistical *estimation problem*. Recall that $\mathfrak{P}(\mathbf{Y})$ denotes the set of preference structures on the solution space $\mathbf{Y}$, for example the set of all total orders (rankings) of the solutions $\boldsymbol{y} \in \mathbf{Y}$. Since the true preference model $\mathbf{R}_{\boldsymbol{x}_0} \in \mathfrak{P}(\mathbf{Y})$ associated with the query $\boldsymbol{x}_0$ is not known, we consider it as a random variable Z with distribution $\mathbf{P}(\cdot \mid \boldsymbol{x}_0)$, where $\mathbf{P}(\cdot \mid \boldsymbol{x}_0)$ is a distribution $\mathbf{P}_\theta(\cdot)$ parametrized by $\theta = \theta(\boldsymbol{x}_0) \in \Theta$. Thus, $\mathbf{P}_\theta(\mathbf{R}_{\boldsymbol{x}_0})$ is the probability that $Z = \mathbf{R}_{\boldsymbol{x}_0}$. The problem, then, is to estimate this distribution or, equivalently, θ on the basis of the information available. This information consists of the preferences $\boldsymbol{y} \succ_{\boldsymbol{x}} \boldsymbol{y}'$ between solutions observed for the neighbors $\boldsymbol{x}$ of $\boldsymbol{x}_0$; let $\mathcal{D}$ denote the complete set of observed preferences collected from the nearest neighbors of $\boldsymbol{x}_0$.

The basic assumption underlying nearest neighbor estimation is that the conditional probability distribution of the output given the input is (approximately) locally constant, that is, $\mathbf{P}(\cdot \mid \boldsymbol{x}_0) \approx \mathbf{P}(\cdot \mid \boldsymbol{x})$ for $\boldsymbol{x}$ close to $\boldsymbol{x}_0$. This assumption justifies considering the preferences $\mathcal{D}$ observed for the neighbors of $\boldsymbol{x}_0$ as a representative sample of $\mathbf{P}_\theta(\cdot)$ and, hence, estimating θ via maximum likelihood (ML) by

$$\theta^{ML} \;=\; \arg\max_{\theta \in \Theta} \mathbf{P}_\theta(\mathcal{D}) \;\; . \tag{2}$$

An important prerequisite for putting this approach into practice is a suitable data generating process, i.e., a process generating preferences in a stochastic way. Moreover, efficient (and probably approximate) inference procedures are needed to estimate the parameters of this process.

3.2 A Discrete Choice Model

Our data generating process is based on the idea of a discrete choice model as used in choice and decision theory [16]. More specifically, we assume that the (absolute) preference for a solution $\boldsymbol{y} \in \mathbf{Y}$ depends on its distance $\Delta(\boldsymbol{y}, \boldsymbol{y}^*) \geq 0$ to an "ideal" solution $\boldsymbol{y}^*$. The distance measure Δ depends on the application and is supposed to model background knowledge regarding the suitability of solutions. Roughly speaking, $\Delta(\boldsymbol{y}, \boldsymbol{y}^*)$ can be seen as a "degree of suboptimality" of $\boldsymbol{y}$: The larger $\Delta(\boldsymbol{y}, \boldsymbol{y}^*)$, the less suitable is $\boldsymbol{y}$ as a substitute of the solution $\boldsymbol{y}^*$. For example, one may think of Δ as a kind of edit distance measuring the cost of an adaptation, i.e., the cost of transforming $\boldsymbol{y}$ into $\boldsymbol{y}^*$.

Suppose the latent *utility* of $\boldsymbol{y}$ is of the form

$$U(\boldsymbol{y}) \;=\; -\beta\, \Delta(\boldsymbol{y}, \boldsymbol{y}^*) + \epsilon \;\; ,$$

with $\beta \geq 0$ and an error term ϵ having an extreme value distribution. In conjunction with the assumption of independence between the error terms of different

solutions, this leads to the *logit* model of discrete choice:

$$\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{y}') = \frac{1}{1 + \exp\left(-\beta(\Delta(\boldsymbol{y}', \boldsymbol{y}^*) - \Delta(\boldsymbol{y}, \boldsymbol{y}^*)\right)} \tag{3}$$

Thus, the probability of observing the (revealed) preference $\boldsymbol{y} \succ \boldsymbol{y}'$ depends on the degree of suboptimality of $\boldsymbol{y}$ and $\boldsymbol{y}'$, namely their respective distances to the ideal solution, $\Delta(\boldsymbol{y}, \boldsymbol{y}^*)$ and $\Delta(\boldsymbol{y}', \boldsymbol{y}^*)$: The larger the difference $\Delta(\boldsymbol{y}', \boldsymbol{y}^*) - \Delta(\boldsymbol{y}, \boldsymbol{y}^*)$, i.e., the less optimal $\boldsymbol{y}'$ in comparison to $\boldsymbol{y}$, the larger the probability to observe $\boldsymbol{y} \succ \boldsymbol{y}'$; if $\Delta(\boldsymbol{y}', \boldsymbol{y}^*) = \Delta(\boldsymbol{y}, \boldsymbol{y}^*)$, then $\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{y}') = 1/2$.

The coefficient β can be seen as a measure of precision of the agent's decisions. For large β, the probability (3) converges toward 0 if $\Delta(\boldsymbol{y}', \boldsymbol{y}^*) < \Delta(\boldsymbol{y}, \boldsymbol{y}^*)$ and toward 1 if $\Delta(\boldsymbol{y}', \boldsymbol{y}^*) > \Delta(\boldsymbol{y}, \boldsymbol{y}^*)$; this corresponds to a deterministic (error-free) decision. The other extreme case, namely $\beta = 0$, models an agent making decisions completely at random.

3.3 Maximum Likelihood Estimation

The probabilistic model outlined above is specified by two parameters: the ideal solution $\boldsymbol{y}^*$ and the (true) precision parameter β^*. Consider the problem of estimating these parameters, i.e., the parameter vector $\theta^* = (\boldsymbol{y}^*, \beta^*)$, from a given set $\mathcal{D} = \{\boldsymbol{y}^{(i)} \succ \boldsymbol{z}^{(i)}\}_{i=1}^N$ of observed preferences. In our case, $\mathcal{D}$ is given by the set of preferences collected from the query's nearest neighbors.

To solve this problem, we refer to the maximum likelihood (ML) estimation principle. Assuming independence of the preferences, the likelihood of $\theta = (\boldsymbol{y}, \beta)$ is given by

$$L(\theta) = L(\boldsymbol{y}, \beta) = \mathbf{P}(\mathcal{D} \,|\, \theta) = \prod_{i=1}^{N} \frac{1}{1 + \exp\left(-\beta(\Delta(\boldsymbol{z}^{(i)}, \boldsymbol{y}) - \Delta(\boldsymbol{y}^{(i)}, \boldsymbol{y})\right)}. \tag{4}$$

Numerically, it is more convenient to deal with the log-likelihood

$$\ell(\theta) = \ell(\boldsymbol{y}, \beta) = -\sum_{i=1}^{N} \log\left(1 + \exp\left(-\beta(\Delta(\boldsymbol{z}^{(i)}, \boldsymbol{y}) - \Delta(\boldsymbol{y}^{(i)}, \boldsymbol{y})\right)\right). \tag{5}$$

The maximum likelihood estimation (MLE) $\theta_{ML} = (\boldsymbol{y}^{ML}, \beta^{ML})$ of θ^* is given by the maximizer of (5) (and hence the maximizer of (4)). Noting that, if $\mathbf{Y}$ is a discrete solution space, $\boldsymbol{y}$ is a discrete parameter while β is a continuous one, we tackle the corresponding optimization problem in two steps.

For a fixed $\boldsymbol{y}$, (5) becomes a one-dimensional function of β. The optimum of this function cannot be found analytically, however, since it is differentiable, an optimal β can easily be found by means of standard numerical optimization techniques like the Newton method. In other words, an optimal β, which we denote $\beta^{ML}(\boldsymbol{y})$, can easily be determined as a function of $\boldsymbol{y}$.

Assuming a discrete solution space $\mathbf{Y}$, the optimization of $\boldsymbol{y}$ can be implemented by means of any general purpose search method. The perhaps simplest

approach is hill climbing: Starting with an initial solution $\boldsymbol{y}$, one determines

$$\boldsymbol{y}' = \arg \max_{\boldsymbol{y} \in \mathcal{N}(\boldsymbol{y})} \ell(\boldsymbol{y}, \beta_{ML}(\boldsymbol{y})) \ , \tag{6}$$

where $\mathcal{N}(\boldsymbol{y}) \subset \mathbf{Y}$ is the neighborhood of $\boldsymbol{y}$. The current solution $\boldsymbol{y}$ is then replaced by $\boldsymbol{y}'$, and this process is continued until $\boldsymbol{y} = \boldsymbol{y}'$. Upon termination of this process, the MLE θ^{ML} is given by the currently optimal solution $(\boldsymbol{y}, \beta^{ML}(\boldsymbol{y}))$.

Since hill climbing is prone to local optima, it is important to find a good initial solution. To this end, we make use of the concept of a *generalized median*. Recall that the generalized median of a set of elements $\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\} \subset \mathbf{Y}$ is given by

$$\boldsymbol{y}_{med} \ = \ \arg \min_{\boldsymbol{y} \in \mathbf{Y}} \sum_{i=1}^{N} \Delta(\boldsymbol{y}, \boldsymbol{y}^{(i)}) \ . \tag{7}$$

To verbalize, the generalized median is the element that minimizes the sum of distances to the $\boldsymbol{y}^{(i)}$. Now, suppose a set of preferences $\{\boldsymbol{y}^{(i)} \succ \boldsymbol{z}^{(i)}\}_{i=1}^{N}$ (instead of a set of single elements) to be given. As an extension of (7), we propose to look for

$$\boldsymbol{y}_{med} \ = \ \arg \min_{\boldsymbol{y} \in \mathbf{Y}} \left(\sum_{i=1}^{N} \Delta(\boldsymbol{y}, \boldsymbol{y}^{(i)}) - \sum_{i=1}^{N} \Delta(\boldsymbol{y}, \boldsymbol{z}^{(i)}) \right) \ , \tag{8}$$

that is, for a solution which is as close as possible to the preferred solutions $\boldsymbol{y}^{(i)}$ and, at the same time, as distant as possible from the non-preferred solutions $\boldsymbol{z}^{(i)}$. This solution is then taken as a starting point of the hill climbing procedure.

Note that a MLE $(\boldsymbol{y}^{ML}, \beta^{ML})$ induces a ranking (with ties) of the complete solution space $\mathbf{Y}$: For all $\boldsymbol{y}, \boldsymbol{z} \in \mathbf{Y}$,

$$\boldsymbol{y} \succeq \boldsymbol{z} \quad \text{iff} \quad \Delta(\boldsymbol{y}, \boldsymbol{y}^{ML}) \leq \Delta(\boldsymbol{z}, \boldsymbol{y}^{ML}) \ . \tag{9}$$

Also note that the MLE $\boldsymbol{y}^{ML}$ is the unique top-element of this ranking (at least provided that $\Delta(\boldsymbol{y}, \boldsymbol{y}') = 0$ implies $\boldsymbol{y} = \boldsymbol{y}'$).

3.4 The Subset Solution Space

As a concrete example, that we shall return to in Section 4 below, consider the special case of the "subset solution space", that is, the case where $\mathbf{Y} = 2^C$, with C being a finite set of labels or items (note that the subset-relation defines a complete lattice structure on $\mathbf{Y}$). Alternatively, exploiting a one-to-one correspondence between subsets (of a finite reference set) and binary vectors (of fixed length), we let $\mathbf{Y} = \{0,1\}^M$, where $M = |C|$. Thus, solutions are vectors $\boldsymbol{y} = (y_1, \ldots, y_M) \in \{0,1\}^M$, with $y_i = 1$ if the i-th item is contained in the corresponding set and $y_i = 0$ otherwise.

There are several commonly used distance measures for subsets (binary vectors), including the Hamming distance

$$\Delta_H(\boldsymbol{y}, \boldsymbol{y}') = \frac{1}{M} \sum_{i=1}^{M} |y_i - y'_i|$$

and the Jaccard distance

$$\Delta_J(\boldsymbol{y}, \boldsymbol{y}') = \frac{\sum_{i=1}^{M} \min(y_i, y_i')}{\sum_{i=1}^{M} \max(y_i, y_i')} \quad .$$

Given a measure of this kind, a reasonable definition of the neighborhood in (6) is $\mathcal{N}(\boldsymbol{y}) = \{\boldsymbol{y}' \in \mathbf{Y} \mid \Delta(\boldsymbol{y}, \boldsymbol{y}') = 1\}$, i.e., $\boldsymbol{y}'$ is a neighbor of $\boldsymbol{y}$ if these two vectors differ by exactly one entry. For the Hamming loss, the determination of the initial solution (8) becomes especially simple. In fact, it is easily verified that a solution $\boldsymbol{y}^{med}$ is given by

$$y_j^{med} = \begin{cases} 1 & \text{if } \sum_{i=1}^{N} \mathbb{I}(y_j^{(i)} = 0) + \mathbb{I}(z_j^{(i)} = 1) < N \\ 0 & \text{otherwise} \end{cases} ,$$

where $\mathbb{I}$ is the indicator function, i.e., $\mathbb{I}(P) = 1$ if the predicate P is true and $= 0$ if P is false.

4 Experiments

We conducted a number of experiments which are meant to provide a first validation of our approach. The main goal of these experiments is to show that, in principle, preference-based CBR is feasible. More specifically, we seek to show that a CBR agent can indeed learn from *indirect* feedback in the form of preferences. Moreover, we compare this kind of learning with the standard approach in which *direct* supervision is available.

4.1 Problem Solving Scenario

In agreement with the motivation underlying preference-based CBR, we consider a scenario in which the supervision is limited in the sense that, despite the possibility to *compare* candidate solutions, it is difficult or even impossible to determine an optimal or correct solution. Again, one may think of applications like cooking as an example: Two recipes can be compared, e.g., by cooking the meals and then testing which of them has a better taste, but there is usually no way to figure out the optimal solution. As explained earlier, the standard problem/solution representation becomes arguable in this setting.

As an alternative, we implement the following CBR procedure simulating a problem solving agent, which, roughly speaking, has the ability to compare candidate solutions and to "guess" new solutions, but not to verify the optimality of a solution. The agent proceeds from an initial case base, in which a case is a problem $\boldsymbol{x}$ together with a set of p pairwise preferences of the form $\boldsymbol{y} \succ_{\boldsymbol{x}} \boldsymbol{z}$. The agent then solves a sequence of problems in turn, and each problem solving episode consists of the following steps:

(i) Retrieval: Given a new query problem $\boldsymbol{x}_0$, the agent retrieves the preferences associated with the k nearest neighbors of $\boldsymbol{x}_0$ in the current case base. Thus, the agent gathers a set $\mathcal{D}$ of preferences, consisting of $k \cdot p$ comparisons in total.

(ii) Prediction: Based on this set of preferences, the agent derives a new solution for $\boldsymbol{x}_0$. More specifically, using our discrete choice model, it derives the ranking (9) and takes the top-ranked element $\boldsymbol{y}^{ML}$ as a prediction.

(iii) Evaluation: To measure the performance of the agent's solution, this solution is compared with the truly optimal solution $\boldsymbol{y}_0$ (which is not known to the agent). More specifically, we define the performance in terms of the Hamming distance $\Delta_H(\boldsymbol{y}_0, \boldsymbol{y}^{ML})$.

(iv) Indirect supervision: The agent is given feedback in the form of comparisons of its solution with p alternative solutions $\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(p)}$. Thus, a set of p pairwise preferences of the form $\boldsymbol{y}^{ML} \succ \boldsymbol{y}^{(i)}$ or $\boldsymbol{y}^{(i)} \succ \boldsymbol{y}^{ML}$ is produced. These pairwise preferences are stored together with $\boldsymbol{x}_0$ as a new case in the case base.

The alternative solutions $\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(p)}$ in step (iv) may originate from different sources. For example, the agent itself may try different solutions. To this end, it may sample suboptimal alternatives from the ranking (9), e.g., using sampling methods like tournament selection [17]. One may also imagine that the agent participates in a competition, in which its solution $\boldsymbol{y}^{ML}$ is compared with the solutions of other participants. In our experiments, we simply generated the $\boldsymbol{y}^{(i)}$ at random (i.e., by sampling from a uniform distribution on $\mathbf{Y}$). Each comparison ($\boldsymbol{y}^{ML}$ vs. $\boldsymbol{y}^{(i)}$) is implemented by means of our discrete choice model (3); recall that this model allows for erroneous comparisons, and that the corresponding level of noise is determined by the precision parameter β.

4.2 Problem Solving Domain

As an application domain, we consider the problem of *multi-label classification* (MLC), an extension of the standard classification problem that has received increasing attention in machine learning in recent years [18]. There are mainly two reasons for this choice. First, the output to be predicted in MLC is a subset of labels relevant for the query instance and, therefore, exactly matches the assumptions of the subset solution space. Second, there are benchmark data set available for the MLC problem.[1]

As a concrete example, we consider the *emotions* data that was created from a selection of songs from 233 musical albums [19]. From each song, a sequence of 30 seconds after the initial 30 seconds was extracted. The resulting sound clips were stored and converted into wave files of 22050 Hz sampling rate, 16-bit per sample and mono. From each wave file, 72 numerical features have been extracted, falling into two categories: rhythmic and timbre. Then, in the emotion labeling process, 6 main emotional clusters are retained corresponding to the Tellegen-Watson-Clark model of mood: amazed-surprised, happy-pleased, relaxing-calm, quiet-still, sad-lonely and angry-aggressive. The task is to predict the subset of labels that apply to each individual song.

[1] http://mlkd.csd.auth.gr/multilabel.html

4.3 Results

We implemented the scenario described in Section 4.1 with different values for the parameters p (number of pairwise comparisons per case) and β (precision of the comparisons); the number of retrieved cases was fixed to $k = 3$. As a baseline, we compared with "standard" CBR, in which the supervision is *direct*: Instead of the indirect supervision in step (iv) of our problem solving scenario, the true solution $\boldsymbol{y}_0$ is shown to the agent (and stored in the case base). Given a new query problem $\boldsymbol{x}_0$, the agent retrieves the solutions of its k nearest neighbors and derives the generalized median (7) as a prediction. As a distance measure on the problem space $\mathbf{X}$, we always used the simple Euclidean distance.

In Fig. 1, the performance is shown for the emotions data in terms of a curve $t \mapsto P(t)$, where $P(t)$ is the average performance in the first t problem solving episodes (i.e., the average distance $\Delta_H(\boldsymbol{y}_0, \boldsymbol{y}^{ML})$). Since this curve depends on the order in which problems are encountered, it was "smoothed" by averaging over several random permutations of a data set. The initial case base always comprised the first k cases, for which pairwise comparisons between randomly generated solutions (or, in the case of the baseline, the true solutions) were added.

As can be seen, standard k-NN performs slightly stronger than preference-based CBR. This, of course, was to be expected: While k-NN is fully supervised, having access to the true solutions of the cases stored in the case base, preference-based CBR is only guided through indirect hints in the form of pairwise comparisons (which are perhaps even noisy). Seen from this point of view, it still performs rather well, and the difference between the methods becomes smaller with an increasing number of preferences per case. Moreover, the learning effects due to an extension of the case base are quite comparable. Similar results, which are omitted here due to space restrictions, were obtained for other MLC data sets

5 Concluding Remarks

Our project agenda envisions a methodological framework of preference-based CBR, which disposes of a sound theoretical basis and, at the same time, accommodates a wide spectrum of potential applications. Ideally, a user can easily "parametrize" this framework, e.g., by choosing the type of output space and the distance measure defined on this space, whereas the methods themselves are completely generic and essentially independent of the concrete application at hand. In this regard, as already remarked in the introduction, we are to some extent guided by AI methodologies like probabilistic graphical models and constraint satisfaction.

Needless to say, this paper is only a first step toward this goal. First, by focusing on the case-based inference part, we only considered one aspect of CBR, albeit an important one. Apart from this, there are of course a number of further issues that need to be addressed, such as case-base organization and maintenance. Second, as already noticed earlier, the methods used for implementing

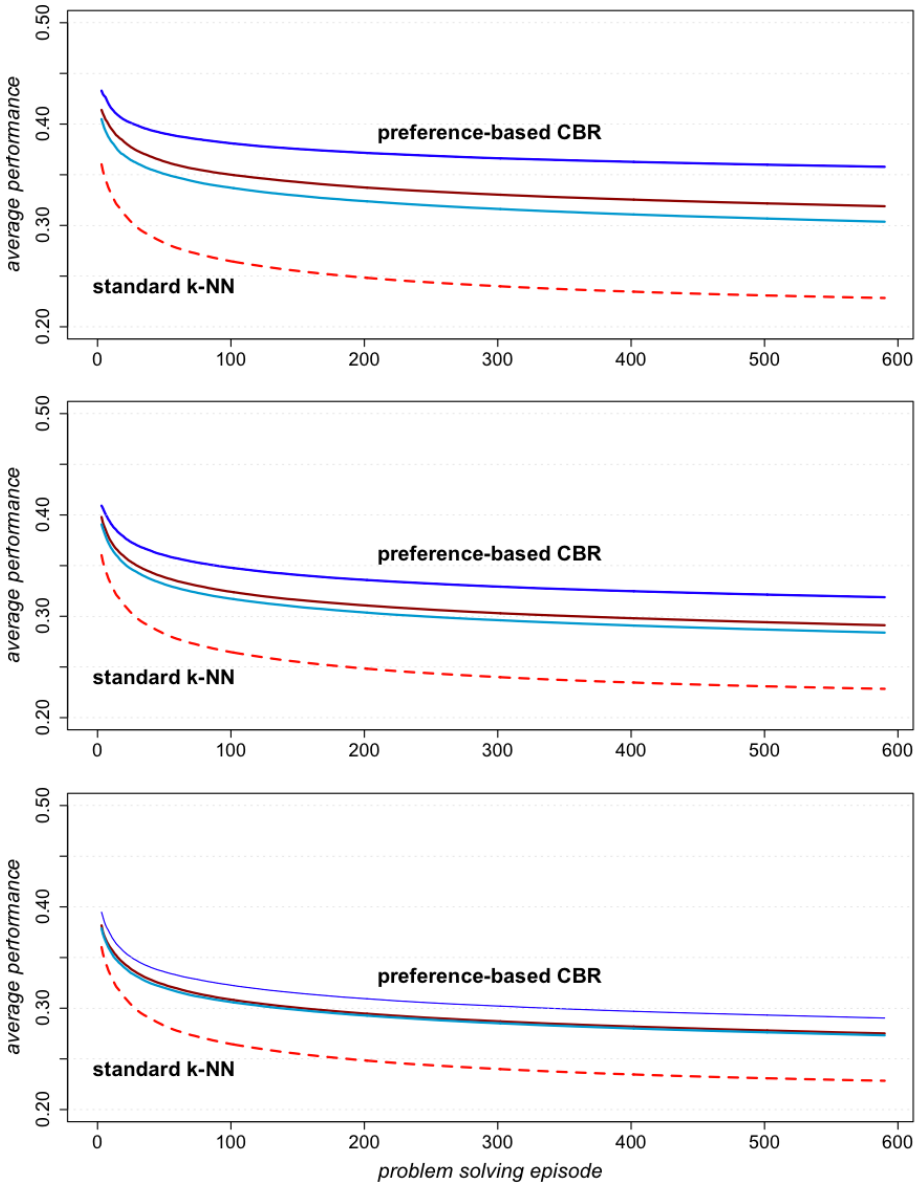

Fig. 1. Performance curves for the emotions data: Standard 3-NN as a baseline (dashed curve) and preference-based CBR (solid) with 7 (upper), 15 (middle) and 30 (lower) preferences per case and precision values of $\beta = 5$, 10 and 20, respectively.

preference-based CBR strongly depend on the type and structure of the solution space. Although the formal approach outlined in Section 3 is quite general, we later on focused on the subset solution space. Thus, similar methods have to be developed and validated for other types of solution spaces, too.

References

1. A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.
2. M.M. Richter. The knowledge contained in similarity measrues. Invited talk at ICCBR-95. `wwwagr.informatik.uni-kl.de/~lsa/cbr/richtericcbr95remarks.html`, 1995.
3. I. Watson. Case-based reasoning is a methodology not a technology. In R. Mile, M. Moulton, and M. Bramer, editors, *Research and Development in Expert Systems XV*, pages 213–223. London, 1998.
4. D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge, Massachusetts, 2009.
5. E. Hüllermeier. *Case-Based Approximate Reasoning*, volume 44 of *Theory and Decision Library, Series B: Mathematical and Statistical Methods*. Springer-Verlag, Heidelberg, Berlin, 2007. 370 pages.
6. J. Doyle. Prospects for preferences. *Comput. Intell.*, 20(2):111–136, 2004.
7. J. Goldsmith and U. Junker. Special issue on preference handling for Artificial Intelligence. 29(4), 2008.
8. C. Domshlak, E. Hüllermeier, S. Kaci, and H. Prade. Preferences in AI: An overview. *Artificial Intelligence*. To appear.
9. R.I. Brafman and C. Domshlak. Preference handling–an introductory tutorial. *AI Magazine*, 30(1), 2009.
10. K. Brinker and E. Hüllermeier. Label ranking in case-based reasoning. In M. Richter and R. Weber, editors, *Proceedings ICCBR–2007, 7th International Conference on Case-Based Reasoning*, number 4626 in LNAI, pages 77–91, Belfast, Northern Ireland, 2007. Springer-Verlag.
11. D. Kibler, D.W. Aha, and MK. Albert. Instance-based prediction of real-valued attributes. *Computational Intelligence*, 5:51–57, 1989.
12. D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.
13. M. Binshtok, R.I. Brafman, C. Domshlak, and S.E. Shiomony. Generic preferences over subsets of structured objects. *Journal of Artificial Intelligence Research*, 34:133–164, 2009.
14. M. Minor, R. Bergmann, S. Görg, and K. Walter. Towards case-based adaptation of workflows. In *Proc. ICCBR–2010*, pages 421–435, Alessandria, Italy, 2010.
15. W. Cheng and E. E. Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2–3):211–225, 2009.
16. M. Peterson. *An Introduction to Decision Theory*. Cambridge Univ. Press, 2009.
17. M. Butz, K. Sastry, and D.E. Goldberg. Tournament selection: Stable fitness pressure in XCS. In *Proc. GECCO-03, Genetic and Evolutionary Computation Conference, Part II*, pages 1857–1869, Chicago, IL, 2003. Springer.
18. G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *Int. J. of Data Warehousing and Mining*, 3(3):1–13, 2007.
19. K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas. Multilabel classification of music into emotions. In *Proc. Int. Conf. Music Information Retrieval*, 2008.