Learning Similarity Functions from Qualitative Feedback

Weiwei Cheng and Eyke Hüllermeier

FB Mathematik/Informatik, Philipps-Universität Marburg, Germany,
{cheng,eyke}@mathematik.uni-marburg.de

Abstract. The performance of a case-based reasoning system often depends on the suitability of an underlying similarity (distance) measure, and specifying such a measure by hand can be very difficult. In this paper, we therefore develop a machine learning approach to similarity assessment. More precisely, we propose a method that learns how to combine given local similarity measures into a global one. As training information, the method merely assumes qualitative feedback in the form of similarity comparisons, revealing which of two candidate cases is more similar to a reference case. Experimental results, focusing on the ranking performance of this approach, are very promising and show that good models can be obtained with a reasonable amount of training information.

1 Introduction

The concept of similarity lies at the heart of case based reasoning (CBR), and the success of a CBR system often strongly depends on the specification of a suitable similarity measure. Unfortunately, domain knowledge provided by human experts is often not sufficient to define an optimal measure by hand. This problem remains despite the existence of "divide-and-conquer" techniques such as the "local–global principle", stating that the (global) similarity between two cases can be obtained as an aggregation of various local measures pertaining to different dimensions or features of a case [1].

In fact, even though it is true that *local similarity measures* can sometimes be defined in a relatively straightforward way, the proper *combination* of these local measures often remains a challenging problem. The reason is that, usually, the definition of a local measure only requires the comparison of properties or attribute values that are measured on the same scale and, therefore, are indeed comparable. However, to aggregate different local measures into a global one, one has to combine properties that may not be easily comparable, and whose importance may be highly subjective or context-dependent.

In this paper, we address the above problem by using machine learning methods to elicit global similarity measures on the basis of feedback in the form of examples. In this regard, the type of feedback expected as input by a learning method is of special importance. Roughly, two types of feedback can be distinguished, namely *absolute* and *relative*. Typically, the former corresponds

to quantitative information about the degree of similarity between two cases, whereas the latter only provides qualitative information about the (order) relation between similarities. Even though absolute feedback is convenient from a learning point of view, it is of course demanding and hence hard to acquire from human experts. In this paper, we therefore proceed from qualitative feedback which is much less difficult to obtain: Given a reference case and two cases to compare with, we only expect information about which of these two cases is more similar. Essentially, this is what Stahl in [2] refers to as *relative case utility feedback*.[1]

The paper is organized as follows: In Section 2, we detail the formal setting underlying our learning method. The method itself is then introduced in its basic form in Section 3 and evaluated empirically in Section 4. We discuss possible extensions of the basic model in Section 5 and related work in Section 6. The paper ends with concluding remarks in Section 7. Before proceeding, we mention that, formally, our approach will not be developed in terms of similarity functions but instead resort to the dual concept of a distance function.

2 Problem Setting

A case base is a subset $\mathrm{CB} \subseteq \mathbb{C}$ with $|\,\mathrm{CB}\,| < \infty$, where $\mathbb{C} \neq \emptyset$ denotes the set of all conceivable cases. We assume the existence of d *local distance measures*

$$\delta_i : \mathbb{C} \times \mathbb{C} \to \mathbb{R}_+ \quad (i = 1 \ldots d). \tag{1}$$

For each pair of cases $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{C}$, $\delta_i(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{R}_+$ is a measure of the distance between these cases with respect to a certain aspect. For example, suppose cases to be represented as graphs, i.e., $\mathbb{C}$ is a set of graphs. A local distance $\delta_i(\boldsymbol{a}, \boldsymbol{b})$ may then be defined by $|n(\boldsymbol{a}) - n(\boldsymbol{b})|$, where $n(\boldsymbol{a})$ is the number of nodes in $\boldsymbol{a}$, or by $\max(n(\boldsymbol{a}), n(\boldsymbol{b})) - s$, where s is the size of the maximal common subgraph.

According to the *local–global principle*, the (global) distance between two cases can be obtained as an aggregation of the local distance measures (1):

$$\Delta(\boldsymbol{a}, \boldsymbol{b}) \;=\; \mathrm{AGG}\,(\,\delta_1(\boldsymbol{a}, \boldsymbol{b}), \delta_2(\boldsymbol{a}, \boldsymbol{b}) \ldots \delta_d(\boldsymbol{a}, \boldsymbol{b})\,)\,, \tag{2}$$

where AGG is a suitable aggregation operator. As a special case, consider a representation of cases in terms of d-dimensional feature vectors

$$\boldsymbol{a} = (a_1, a_2 \ldots a_d) \in \mathbb{A}_1 \times \mathbb{A}_2 \times \ldots \times \mathbb{A}_d, \tag{3}$$

where $\mathbb{A}_i$ is the domain of the i-th attribute A_i. $\mathbb{C}$ is then given by the Cartesian product of these domains, $\mathbb{A}_1 \times \mathbb{A}_2 \times \ldots \times \mathbb{A}_d$, and the local distances are of the form

$$\delta_i : \mathbb{A}_i \times \mathbb{A}_i \to \mathbb{R}_+, \tag{4}$$

[1] In a different context though quite similar way, relative feedback of that kind is also used in information retrieval [3].

i.e., $\delta_i(a_i, b_i)$ assigns a distance to each pair of attributes $(a_i, b_i) \in \mathbb{A}_i \times \mathbb{A}_i$; obviously, (4) is a special case of (1). Even though a feature-based representation is of course not always optimal (in terms of performance), it is often the most feasible approach and still predominant in practice [4]. In our experiments in Section 4, we shall use data sets with numerical attributes and local distances $\delta_i(a_i, b_i) = |a_i - b_i|$.

2.1 Linear Combination of Local Measures

For the time being, we shall focus on a special type of aggregation operator (2) which is simple and often used in practice, namely a linear combination:

$$\Delta(\boldsymbol{a}, \boldsymbol{b}) \;=\; \sum_{i=1}^{d} w_i \cdot \delta_i(\boldsymbol{a}, \boldsymbol{b}). \tag{5}$$

Note that it makes sense to require

$$\boldsymbol{w} = (w_1 \dots w_d) \;\geq\; 0 \tag{6}$$

in order to guarantee the monotonicity of the distance measure (2). That is, if a local distance increases, the global distance cannot decrease.

Despite its simplicity, the linear model (5) has a number of merits. For example, it is easily interpretable, as a weight w_i is in direct correspondence with the *importance* of a local measure. In principle, it thus also allows one to incorporate additional background knowledge in a convenient way, e.g., that attribute A_i is at least as important as attribute A_j ($w_i \geq w_j$). Finally, the linear model is attractive from a machine learning point of view, as it is amenable to efficient learning algorithms and, moreover, to non-linear extensions via "kernelization" [5]. We shall come back to this point in Section 5.

2.2 Learning Distance Measures and Learning to Rank

The problem we shall consider in the next section is to learn the weights w_i in (5) from user feedback. The kind of training information we assume to be given as input to a learner is *qualitative* feedback of the following form: case $\boldsymbol{a}$ is more similar to $\boldsymbol{b}$ than to $\boldsymbol{c}$. Information of this type will be denoted by a triplet $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \in \mathbb{C}^3$. Note that qualitative feedback of the above kind is typically much easier to acquire than absolute feedback, that is, the degree of distance $\Delta(\boldsymbol{a}, \boldsymbol{b})$ between two cases $\boldsymbol{a}$ and $\boldsymbol{b}$.

A global distance function induces for any query a total order on the case base: Given a query $\boldsymbol{q} = (q_1 \dots q_d) \in \mathbb{C}$ and two cases $\boldsymbol{a}, \boldsymbol{b} \in \mathrm{CB}$,

$$\boldsymbol{a} \succeq_{\boldsymbol{q},\Delta} \boldsymbol{b} \;\stackrel{\mathrm{df}}{\Longleftrightarrow}\; \Delta(\boldsymbol{q}, \boldsymbol{a}) \leq \Delta(\boldsymbol{q}, \boldsymbol{b}).$$

Indeed, it is often only the *ordering* of cases that really matters, not the distance degrees themselves. For example, to retrieve the k nearest neighbors in

NN retrieval, a correct ordering of the case base is sufficient. Seen from this point of view, it is actually not important to approximate the true distance (2) accurately in the sense of minimizing a norm $|\Delta - \Delta^{est}|$ (such as the $\mathcal{L}^2$ norm) on $\mathbb{C} \times \mathbb{C} \to \mathbb{R}_+$ mappings. Instead, it is more important to find an estimation Δ^{est} that induces (almost) the same rankings, i.e., an estimation for which

$$\succeq_{\boldsymbol{q},\Delta^{est}} \quad \approx \quad \succeq_{\boldsymbol{q},\Delta} \, . \tag{7}$$

In our experiments in Section 4, we shall therefore evaluate a distance function Δ^{est} by comparing the ranking induced by this function with the actually true ranking (in terms of standard distance measures for rankings).

3 The Learning Algorithm

Suppose to be given a set of training data $\mathbb{T}$, which consists of a finite number of exemplary similarity constraints $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$, where $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \text{CB}$. As mentioned previously, the basic learning problem is to find a distance function (5) which is as much as possible in agreement with these constraints and also satisfies the monotonicity property (6). Besides, this function should of course generalize as well as possible beyond these examples in the sense of (7).

3.1 Distance Learning as a Classification Problem

A key idea in our approach is to reduce the above learning problem to a *binary classification problem*. Due to the assumption of a linear distance model, this is indeed possible: The inequality $\Delta(\boldsymbol{a}, \boldsymbol{b}) < \Delta(\boldsymbol{a}, \boldsymbol{c})$ required by a constraint $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ is equivalent to

$$\langle \boldsymbol{w}, \boldsymbol{x} \rangle \; = \; \sum_{i=1}^{d} w_i \cdot x_i > 0,$$

where $x_i \stackrel{\text{df}}{=} \delta_i(\boldsymbol{a}, \boldsymbol{c}) - \delta_i(\boldsymbol{a}, \boldsymbol{b})$. From a classification point of view, $\boldsymbol{x} = T(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) = (x_1 \ldots x_d)$ is hence a positive example and $-\boldsymbol{x}$ a negative one. That is, a similarity constraint $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ can be transformed into two examples $(\boldsymbol{x}, +1)$ and $(-\boldsymbol{x}, -1)$ for binary classification learning; see Fig. 1 for a schematic illustration. Moreover, the vector $\boldsymbol{w} = (w_1 \ldots w_d)$ that defines the distance function (5) in a unique way also defines the model (hyperplane) of the associated classification problem.

3.2 Ensemble Learning

Binary classification is a well-studied problem in machine learning, and a large repertoire of corresponding learning algorithms is available. In principle, all these methods can be applied in our context. Here, we make use of an ensemble learning technique, mainly for two reasons. First, ensembles typically produce more

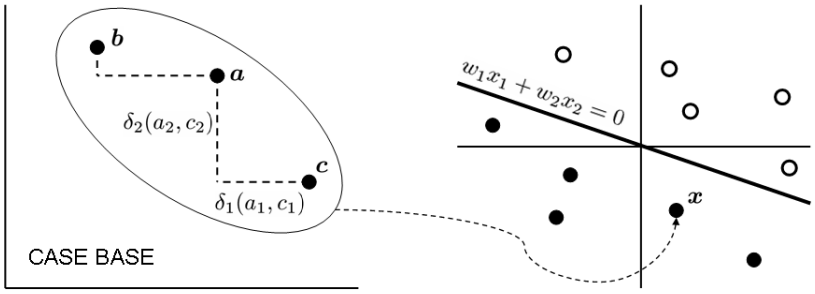

Fig. 1. Transformation of the distance learning problem to a classification problem: Each similarity constraint referring to a case triplet gives rise to a classification example.

accurate predictions than individual learners. Secondly, as will be detailed in Section 3.4, the ensemble technique is also useful in connection with the selection of informative queries to be given to the user.

More specifically, we train an ensemble of m linear perceptrons, using the noise-tolerant learning algorithm proposed in [6], on permutations of the original training data; the j-th perceptron is represented by a weight vector $\boldsymbol{w}^{(j)} = (w_1^{(j)} \dots w_d^{(j)})$. The output produced by this ensemble for an input $\boldsymbol{x} \in \mathbb{R}^d$ is given by the average of the individual outputs:

$$M(\boldsymbol{x}) = \frac{1}{m} \sum_{j=1}^{m} \sum_{i=1}^{d} w_i^{(j)} \cdot x_i \; = \; \sum_{i=1}^{d} w_i^* \cdot x_i. \tag{8}$$

The w_i^* can be taken as estimates of the w_i in the distance function (5).

In [7], it was shown that (8) approximates the center of mass of the version space and, hence, that this learning algorithm yields an approximation to a Bayes point machine. The latter seeks to find the midpoint of the region of intersection of all hyperplanes bisecting the version space into two halves of equal volume. This midpoint, the Bayes point, is known to be approximated by the center of mass of the version space.

3.3 Monotonicity

The monotonicity constraint (6) constitutes an interesting challenge from a machine learning point of view. In fact, this relatively simple property is not guaranteed by many standard machine learning algorithms. That is, a model that implements a distance function $\Delta(\cdot)$ may easily violate the monotonicity property, even if this condition is satisfied by all examples used as training data.

Fortunately, our learning algorithm allows us to incorporate the monotonicity constraint in a relatively simple way. The well-known perceptron algorithm is an error-driven on-line algorithm that adapts the weight vector $\boldsymbol{w}$ in an incremental way. To guarantee monotonicity, we simply modify this algorithm as follows:

Each time an adaptation of $\boldsymbol{w}$ produces a negative component $w_i < 0$, this component is set to 0. Roughly speaking, the original adaptation is replaced by a "thresholded" adaptation.

In its basic form, the perceptron algorithm provably converges after a finite number of iterations, provided the data is linearly separable. We note that this property is preserved by our modification (proof omitted due to space restrictions). Obviously, monotonicity of the single perceptrons implies monotonicity of their average (8).

3.4 Active Learning

So far, we did not address the question of where the training data $\mathbb{T}$ actually comes from. The simplest assumption is that $\mathbb{T}$ is just a random sample, even though this assumption is of course not always justified in practice. In this section, we consider the interesting scenario in which additional training examples can be gathered by asking for feedback from the user of a CBR system. Thus, feedback is derived by selecting two cases $\boldsymbol{b}, \boldsymbol{c}$ and a reference case $\boldsymbol{a}$, and asking the user whether $\boldsymbol{b}$ or $\boldsymbol{c}$ is more similar to $\boldsymbol{a}$.

Again, the simplest way to generate such a query is to choose it at random from CB. However, realizing that different queries can have different information content, the goal of this step should be the selection of a maximally informative query, i.e., an example that helps to improve the current distance function Δ^{est} as much as possible. This idea of generating maximally useful examples in a targeted way is the core of *active learning* strategies [8].

In the literature, numerous techniques for active learning have been proposed, most of them being heuristic approximations to theoretically justified (though computationally or practically infeasible) methods. Here, we resort to the *Query by Committee* approach [8]. Given an ensemble of models, the idea is to find a query for which the disagreement between the predictions of these models is maximal. Intuitively, a query of that kind corresponds to a "critical" and, therefore, potentially informative example. In our case, the models are given by the ensemble of perceptrons (cf. Section 3.2). Moreover, given a reference case $\boldsymbol{a}$ and two other cases $\boldsymbol{b}$ and $\boldsymbol{c}$, two models Δ_1, Δ_2 disagree with each other if $\Delta_1(\boldsymbol{a}, \boldsymbol{b}) < \Delta_1(\boldsymbol{a}, \boldsymbol{c})$ while $\Delta_2(\boldsymbol{a}, \boldsymbol{b}) > \Delta_2(\boldsymbol{a}, \boldsymbol{c})$.

Various strategies are conceivable for finding a maximally critical query, i.e., a query for which there is a high disagreement among the ensemble. Our current implementation uses the following approach: Let $W = \left\{\boldsymbol{w}^{(1)} \dots \boldsymbol{w}^{(m)}\right\}$ be the set of weight vectors of the perceptrons that constitute the current ensemble. In a first step, the two maximally conflicting models are identified, that is, two weight vectors $\left\{\boldsymbol{w}^{(i)}, \boldsymbol{w}^{(j)}\right\} \in W$ such that $\|\boldsymbol{w}^{(i)} - \boldsymbol{w}^{(j)}\|$ is maximal. Then, using these two weight vectors, two rankings π_i and π_j of the cases in CB are generated, respectively, taking a randomly selected reference case $\boldsymbol{a}$ as a query. Starting from the top of these rankings, the first conflict pair $(\boldsymbol{b}, \boldsymbol{c})$ is found, i.e., the first position k such that $\boldsymbol{b}$ and $\boldsymbol{c}$ are put on position k, respectively, by π_i

and π_j, and $\boldsymbol{b} \neq \boldsymbol{c}$.[2] This conflict pair then gives rise to a query for the user. Depending on the answer, either $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ or $(\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{b})$ is added as an example to the training data $\mathbb{T}$ (and the learner is retrained on the expanded data set).

4 Experimental Results

This section presents the results of experimental studies that we conducted to investigate the efficacy of our approach. The aim of the experiments was twofold. A first goal was to show that the performance is convincing in absolute terms, which means that good predictions can be achieved with a reasonable amount of training information. Second, we wanted to provide evidence for the effectiveness of the special features of our learning algorithm, namely the incorporation of the monotonicity constraint, the use of an ensemble of models, and the active learning strategy.

4.1 Quality Measures

Let π^{est} denote the ranking of the case base induced by a learned distance function Δ^{est}. That is, when ordering all cases according to their estimated distance to the query, $\pi^{est}(\boldsymbol{a})$ is the position of case $\boldsymbol{a}$. To evaluate Δ^{est}, we compare π^{est} with the ranking π induced by the true distance function Δ. To this end, we use three different quality measures: Kendall's tau, recall, and the position error.

Kendall's tau is a well-known and widely used rank correlation measure [9]. It calculates the number of pairwise rank inversions, i.e., the number of discordant pairs $(\boldsymbol{a}, \boldsymbol{b})$:

$$\#\left\{(\boldsymbol{a}, \boldsymbol{b}) \mid \pi(\boldsymbol{a}) < \pi(\boldsymbol{b}),\, \pi^{est}(\boldsymbol{a}) > \pi^{est}(\boldsymbol{b})\right\}.$$

More specifically, the Kendall tau coefficient normalizes this number to the interval $[-1, +1]$ such that $+1$ is obtained for identical rankings and -1 in the case of reversed rankings.

To complement the rank correlation, which takes the whole ranking into account, we employed a second measure that puts more emphasis on the top-ranks and is closely related to the recall measure commonly used in information retrieval. Let $\mathcal{K}$ be the set of top-k elements of the ranking π, that is, $\mathcal{K} = \{\boldsymbol{a} \in \text{CB} \mid \pi(\boldsymbol{a}) \leq k\}$, where k is an integer that is usually small in comparison with the size of the case base (as a default value, we use $k = 10$); likewise, let $\mathcal{K}^{est}$ denote the top-k elements of π^{est}. We then define

$$\text{recall}(\pi, \pi^{est}) \stackrel{\text{df}}{=} \frac{\#(\mathcal{K} \cap \mathcal{K}^{est})}{k}. \tag{9}$$

This measure corresponds to the percentage of top-k cases of the ranking π that are also among the predicted top-k cases. It is motivated by the assumption

[2] In principle, an additional strategy is needed for the case where the two orderings are identical. However, even though this problem is theoretically possible, it never occurred in our experiments. Therefore, we omit further details here.

that, typically, the top-k cases of a ranking are more important than the cases at lower ranks.

Focusing even more on the top and looking only at the case which is most similar to the query, we define the *position error* by the position of this case in the predicted ranking (minus 1): $\text{pos}(\pi^{est}) \stackrel{\text{df}}{=} \pi^{est}\left(\pi^{-1}(1)\right) - 1$, where π^{-1} is the inverse of π, i.e., $\pi^{-1}(1)$ is the topmost case in π.

4.2 Data

To analyze our algorithm under different conditions, we used data sets of varying size in terms of the number of features and the size of the case base: UNI (6/200), Iris (4/150), Wine (13/178), Yeast (24/2465), NBA (15,3924). The UNI data set is a ranking of the top-200 universities world-wide in 2006, provided by [10], where the universities are evaluated in terms of six numerical features (peer review score, recruiter review score, international faculty score, international students score, staff-to-student ratio, citation-to-staff ratio). Iris and Wine are widely used benchmark data sets that are publicly available from the UC Irvine machine learning repository [11]. Yeast is a genetic data set of phylogenetic profiles for the Yeast genome [12]. The genome consists of 2465 genes, and each gene is represented by an associated phylogenetic profile of length 24. The NBA data set records career statistics for regular seasons by NBA players. Each player is characterized by a set of 15 match statistics, e.g., scoring, rebound, turnover, steal, etc. This data set is part of the basketball player data set, which is published and maintained by *databasebasketball.com*.

4.3 Experiments

To answer the questions raised at the beginning of this section, we conducted three comparative studies:

- The first experiment investigates the advantage of using a modified perceptron learning algorithm that ensures monotonicity. We compare results for the standard perceptron algorithm (**standard**) with those for the modified one (**monotone**). For both variants, we use an ensemble of size $m = 10$ and non-active learning.
- The second experiment investigates the advantage of using an ensemble of models instead of a single model. Here, we compare the results obtained by training a single perceptron (**single**) with those of an ensemble of size $m = 10$ (**ensemble**). For both variants, we use monotone, non-active learning.
- Finally, we investigate the improvements due to our active learning strategy. To this end, we compare the active-learning strategy[3] (**active**) as described in Section 3 with the random strategy (**random**) that simply selects triplets $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \in \text{CB}$ at random.

[3] Initialized with 10 randomly chosen triplets

In all experiments, we derived quality measures for different numbers of training data, ranging from 10 to 100. In a single experiment, we randomly generated a weight vector $\boldsymbol{w}$ (uniformly in $[0,1]^d$) as the ground truth. A fixed number of training examples was then generated according to this vector, either by selecting triplets $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ at random or by using the active learning strategy. A model is then learned on this data. To evaluate its quality, a query is generated at random, and the ranking predicted for this query is compared to the true ranking; this is done repeatedly and results are averaged.

Figures 2, 3, and 4 show the results in terms of mean values and standard deviations obtained from 100 repetitions. As can clearly be seen from the learning curves in these figures, our approach to learning distance functions is indeed quite effective, and all its extensions do obviously pay off. This is especially true for the incorporation of the monotonicity constraint and the active learning strategy, where the learning curves show a visible improvement. The ensemble effect, on the other hand, yields only a slight improvement (the learning curves are often very close) which is, nevertheless, still statistically significant.

5 Extensions

The linearity assumption underlying model (5) is of course not always justified in practice. Instead, the aggregation (2) may be a nonlinear operator, and the classification examples $\boldsymbol{x} = T(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ created by triplets of cases (cf. Fig. 1) will no longer be linearly separable. As our idea of transforming the distance learning problem into a classification problem, as outlined in Section 3.1, strongly exploits the linearity of (5), one may wonder whether this approach can be extended to the nonlinear case. Indeed, there are different options for such an extension, two of which will be sketched in this section.

5.1 Kernel-based Learning

First, it is important to note that our transformation only exploits the linearity in the coefficients w_i, not the linearity in the local distances. Therefore, the approach can easily be extended to linear combinations of arbitrary functions of the local distances. An especially important example is a model which is, in a similar form, often used in fields like statistics and economics:

$$\Delta(\boldsymbol{a}, \boldsymbol{b}) = \sum_{i=1}^{d} w_i \cdot \delta_i(\boldsymbol{a}, \boldsymbol{b}) + \sum_{i=1}^{d} \sum_{j=i}^{d} w_{ij} \cdot \delta_i(\boldsymbol{a}, \boldsymbol{b}) \delta_j(\boldsymbol{a}, \boldsymbol{b}). \tag{10}$$

The terms $\delta_i(\boldsymbol{a}, \boldsymbol{b})\delta_j(\boldsymbol{a}, \boldsymbol{b})$, which are called *interaction terms*, enable the modeling of interdependencies between different local distances.

It is noticeable that (10) is closely related to the transformation induced by a quadratic kernel $(\boldsymbol{x}, \boldsymbol{x}') \mapsto \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^2$ in kernel-based learning. More generally, (10)

is actually equivalent to (5) when looking at the local distances δ_i as *features*. Indeed, both models are special cases of the representation

$$\Delta(\boldsymbol{a},\boldsymbol{b}) \;=\; \boldsymbol{v}\cdot\phi(\boldsymbol{d}(\boldsymbol{a},\boldsymbol{b})) = \sum_{\ell=1}^{k} v_\ell \cdot \phi_\ell(\boldsymbol{d}(\boldsymbol{a},\boldsymbol{b})), \tag{11}$$

where $\boldsymbol{d}(\boldsymbol{a},\boldsymbol{b}) = (\delta_1(\boldsymbol{a},\boldsymbol{b})\ldots\delta_d(\boldsymbol{a},\boldsymbol{b}))$, and the ϕ_ℓ are properties of this vector of local distances. This provides the basis for "kernelizing" our approach. Without going into technical detail, we just mention that finding a model with maximal (soft) margin then comes down to solving a quadratic program defined as follows:

$$\min_{\boldsymbol{v},\xi_i} \frac{1}{2}\|\boldsymbol{v}\| + C \sum_{(\boldsymbol{a}_i,\boldsymbol{b}_i,\boldsymbol{c}_i)} \xi_i \quad \text{s.t.} \begin{cases} \boldsymbol{v}\cdot(\phi(\boldsymbol{d}(\boldsymbol{a},\boldsymbol{c})) - \phi(\boldsymbol{d}(\boldsymbol{a},\boldsymbol{b}))) \geq 1-\xi_i \\ \xi_i \geq 0 \end{cases},$$

where the $(\boldsymbol{a}_i,\boldsymbol{b}_i,\boldsymbol{c}_i)$ are the training examples and C is a regularization parameter. Eventually, this leads to learning a model that can be represented as

$$\Delta(\boldsymbol{a},\boldsymbol{b}) = \sum_i \alpha_i \left(K(\boldsymbol{d}(\boldsymbol{a},\boldsymbol{b}),\boldsymbol{d}(\boldsymbol{a}_i,\boldsymbol{c}_i)) - K(\boldsymbol{d}(\boldsymbol{a},\boldsymbol{b}),\boldsymbol{d}(\boldsymbol{a}_i,\boldsymbol{b}_i))\right),$$

where $K(\cdot)$ is the kernel associated with the feature map $\phi(\cdot)$.

5.2 Nonlinear Classification and Sorting

Our original model as well as the extension (10) establish a one-to-one correspondence between the distance function $\Delta(\cdot)$ and the model for the induced classification problem. In fact, there is even a one-to-one correspondence between the parameters of $\Delta(\cdot)$ and the parameters of the corresponding (linear) classifier. A second extension is based on the observation that such a one-to-one correspondence, even if desirable, is in principle not needed.

Suppose we train a possibly nonlinear classifier $C(\cdot)$ that separates the classification examples induced by the similarity constraints given. From this model, it is perhaps not possible to recover the distance function $\Delta(\cdot)$ in explicit form. Still, given a query case $\boldsymbol{q}$ and any pair of cases $\boldsymbol{a},\boldsymbol{b} \in$ CB, the classifier $C(\cdot)$ can answer the question whether $\boldsymbol{a}$ or $\boldsymbol{b}$ is more similar to $\boldsymbol{q}$: In the first case $C(\boldsymbol{x}) = +1$, while in the second case $C(\boldsymbol{x}) = -1$, where $\boldsymbol{x} = T(\boldsymbol{q},\boldsymbol{a},\boldsymbol{b})$. As this information is a sufficient prerequisite for applying a *sorting algorithm*, it is, in principle, again possible to order the case base for the query $\boldsymbol{q}$. Such an algorithm cannot be applied directly, however, as a non-perfect classifier may produce non-transitive preferences. Yet, there are "noise-tolerant" ranking algorithms that can handle non-transitive preferences and yield good approximations to a true ranking [13].

6 Related Work

The learning and adaptation of similarity or distance measures has received considerable attention is CBR and related fields. In particular, the work of Stahl

[14, 15, 2, 16] shares a number of commonalities with ours. In fact, the problem considered in [14] is basically the same, namely to learn the weights in a linear combination of local similarity functions. However, the setting of the learning problem is quite different, just like the learning method itself. Stahl [14] applies a conventional gradient descent algorithm to minimize an "average similarity error". To obtain this error, he assumes the availability of a "similarity teacher" who, given a query case, is able to provide feedback in the form of a ranking of a subset of cases of the case base. In [17], Stahl and Gabel also address the problem of learning *local* similarity measures. They propose evolutationary optimization techniques as an approach to adaptation.

Methods for feature weighing and selection have also been studied by many other authors, especially in the context of k-NN classification [18–22]. In an early work, Wettschereck and Aha have proposed a general framework for comparing feature weighting methods [23]. They distinguish such methods along five dimensions, namely feedback, weight space, representation, generality, and knowledge. More recent methods for feature weighing can also be found in machine learning research [24, 25].

Finally, problems related to feature weighing, selection, and aggregation are of course also studied outside CBR and machine learning research, for example in fields like decision making and information fusion (e.g. [26]). A complete review of the literature, however, is beyond the scope of this paper.

7 Summary and Conclusions

To support the specification of similarity (distance) measures in CBR, we have proposed a machine learning algorithm that proceeds from predefined local distance functions and learns how to combine these functions into a global measure. The algorithm is quite user-friendly in the sense that it only assumes qualitative feedback in the form of similarity comparisons: Case $\boldsymbol{a}$ is more similar to $\boldsymbol{b}$ than to $\boldsymbol{c}$. First experiments have yielded promising results, showing that the algorithm is effective and, moreover, that its special features (monotonicity, ensemble learning, active selection of examples) lead to increased performance.

Apart from technical aspects, we consider the general idea of the approach as especially interesting, as it allows one to reduce the problem of distance learning to a conventional classification problem. Thus, distance learning becomes amenable to a large repertoire of existing and well-understood algorithms. In this regard, we are currently elaborating on several extensions of our basic model, such as those outlined in Section 5.

References

1. Richter, M.M.: Foundations of similarity and utility. The 20th International FLAIRS Conference, Key West, Florida (2007)
2. Stahl, A.: Learning similarity measures: A formal view based on a generalized CBR model. In: ICCBR–05, Chicago, USA, Springer (2005) 507–521

3. Joachims, T.: Optimizing search engines using clickthrough data. In: KDD–2002, Proc. of the ACM Conference on Knowledge Discovery and Data Mining. (2002)
4. Cunningham, P.: A taxonomy of similarity mechanisms for case-based reasoning. Technical Report UCD-CSI-2008-01, University College Dublin (2008)
5. Schölkopf, B., Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press (2001)
6. Khardon, R., Wachman, G.: Noise tolerant variants of the perceptron algorithm. The Journal of Machine Learning Research **8** (2007) 227–248
7. Herbrich, R., Graepel, T., Campbell, C.: Bayes point machines. Journal of Machine Learning Research **1** (2001) 245–279
8. Seung, H., Opper, M., Sompolinsky, H.: Query by committee. In: Computational Learning Theory. (1992) 287–294
9. Kendall, M.: Rank correlation methods. Charles Griffin, London (1955)
10. O'Leary, J.: World university rankings editorial - global vision ensures healthy competition. The Times Higher Education Supplement (2006)
11. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
12. Pavlidis, P., Weston, J., Cai, J., Grundy, W.: Gene functional classification from heterogeneous data. In: Journal of Comput. Biology. Volume 9. (2002) 401–411
13. Cohen, W., Schapire, R., Singer, Y.: Learning to order things. Journal of Artificial Intelligence Research **10** (1999)
14. Stahl, A.: Learning feature weights from case order feedback. In: ICCBR–01, Vancouver, Canada, Springer-Verlag (2001) 502–516
15. Stahl, A., Schmitt, S.: Optimizing retrieval in CBR by introducing solution similarity. In: Proc. Int. Conf. on Art. Intell., IC-AI, Las Vegas, USA (2002)
16. Stahl, A., Gabel, T.: Optimizing similarity assessment in case-based reasoning. In: Proc. 21th National Conf. on Artificial Intelligence, AAAI. (2006)
17. Stahl, A., Gabel, T.: Using evolution programs to learn local similarity measures. In: ICCBR–03, Trondheim, Norway, Springer (2003) 537–551
18. Bonzano, A., Cunningham, P., Smyth, B.: Using introspective learning to improve retrieval in CBR: A case study in air traffic control. In: ICCBR–97, Providence, Rhode Island, USA, Springer (1997) 291–302
19. Kononenko, I.: Estimating attributes: Analysis and extensions of RELIEF. In: European Conference on Machine Learning. (1994) 171–182
20. Ricci, F., Avesani, P.: Learning a local similarity metric for case-based reasoning. In: ICCBR–95. (1995) 301–312
21. Wilke, W., Bergmann, R.: Considering decision cost during learning of feature weights. In: European Workshop on CBR. (1996) 460–472
22. Branting, K.: Acquiring customer preferences from return-set selections. In: ICCBR–01, Vancouver, Canada, Springer (2001) 59–73
23. Wettschereck, D., Aha, D.: Weighting features. Proc. ICCBR–95, Sesimbra, Portugal (1995) pages 347–358
24. Wu, Y., Ianakiev, K., Govindaraju, V.: Improvements in k-nearest neighbor classification. In: ICAPR. (2001) 222–229
25. Toussaint, G.: Geometric proximity graphs for improving nearest neighbor methods in instance-based learning and data mining. Int. J. Comput. Geometry Appl. **15**(2) (2005) 101–150
26. Torra, V., Narukawa, Y.: Modeling Decisions: Information Fusion and Aggregation Operators. Springer (2007)

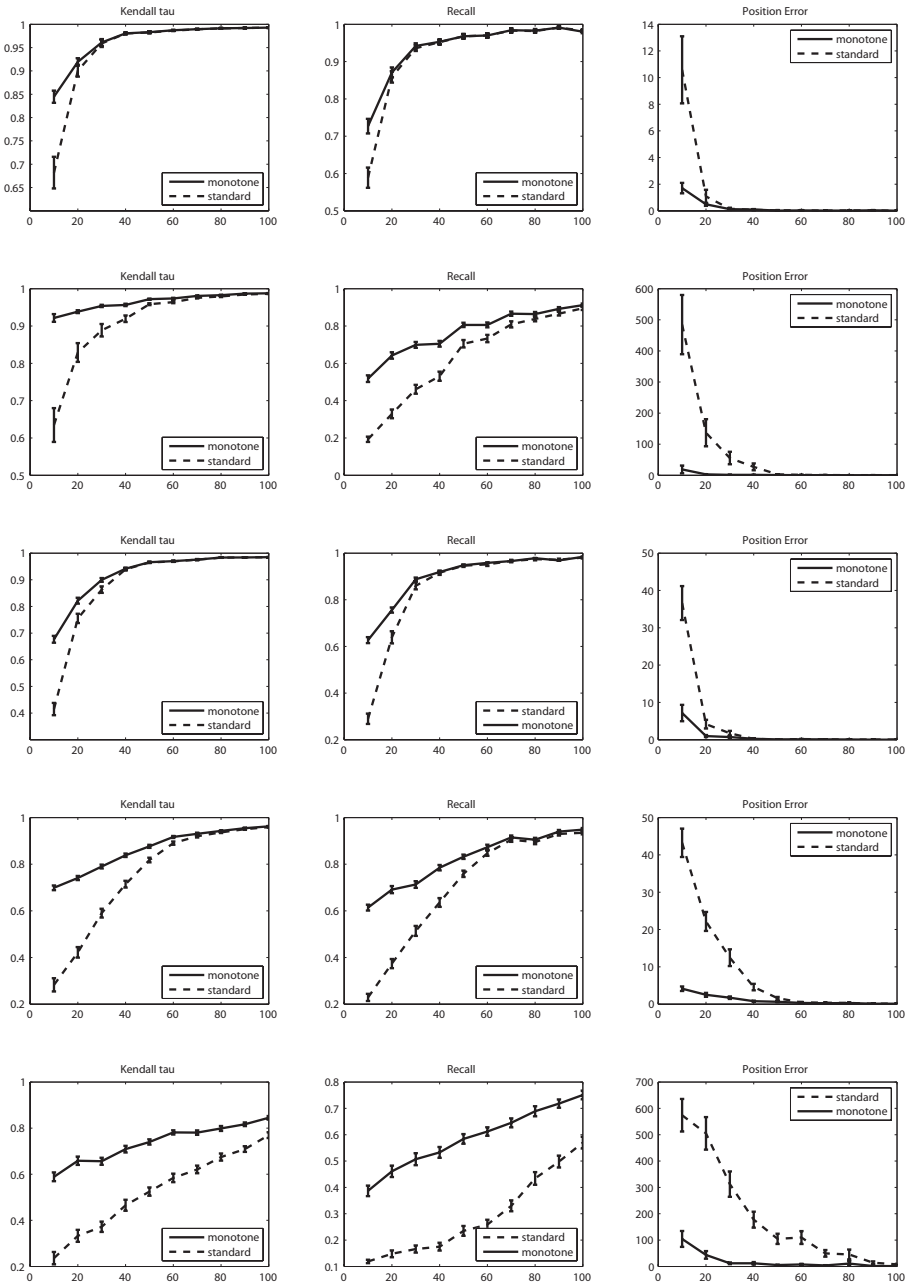

Fig. 2. Monotone vs. non-monotone learning: Experimental results in terms of rank correlation, recall, and position error as a function of the number of training examples (x-axis). Data sets from top to bottom: Iris, NBA, UNI, Wine, Yeast.

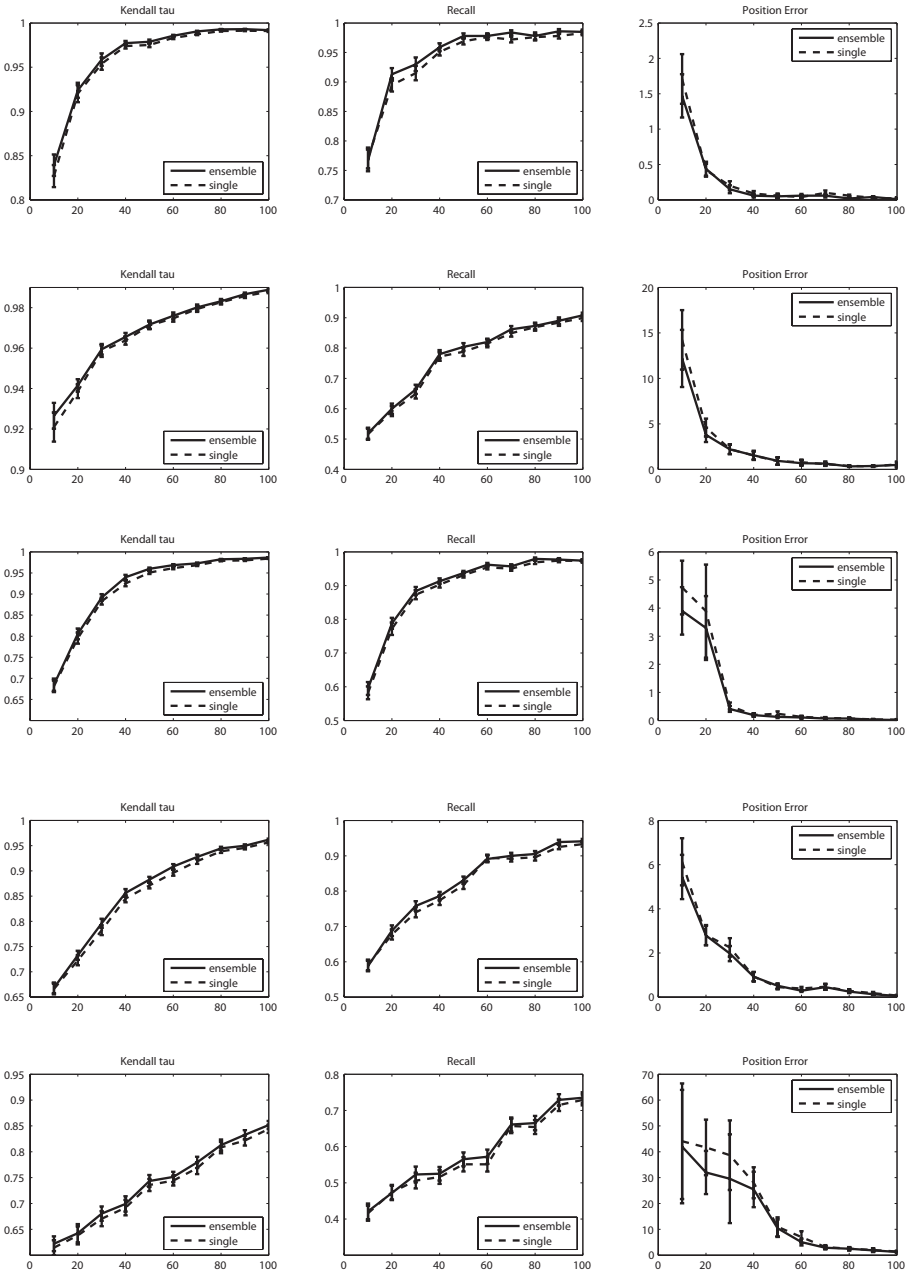

Fig. 3. Single vs. ensemble learning: Experimental results in terms of rank correlation, recall, and position error as a function of the number of training examples (x-axis). Data sets from top to bottom: Iris, NBA, UNI, Wine, Yeast.

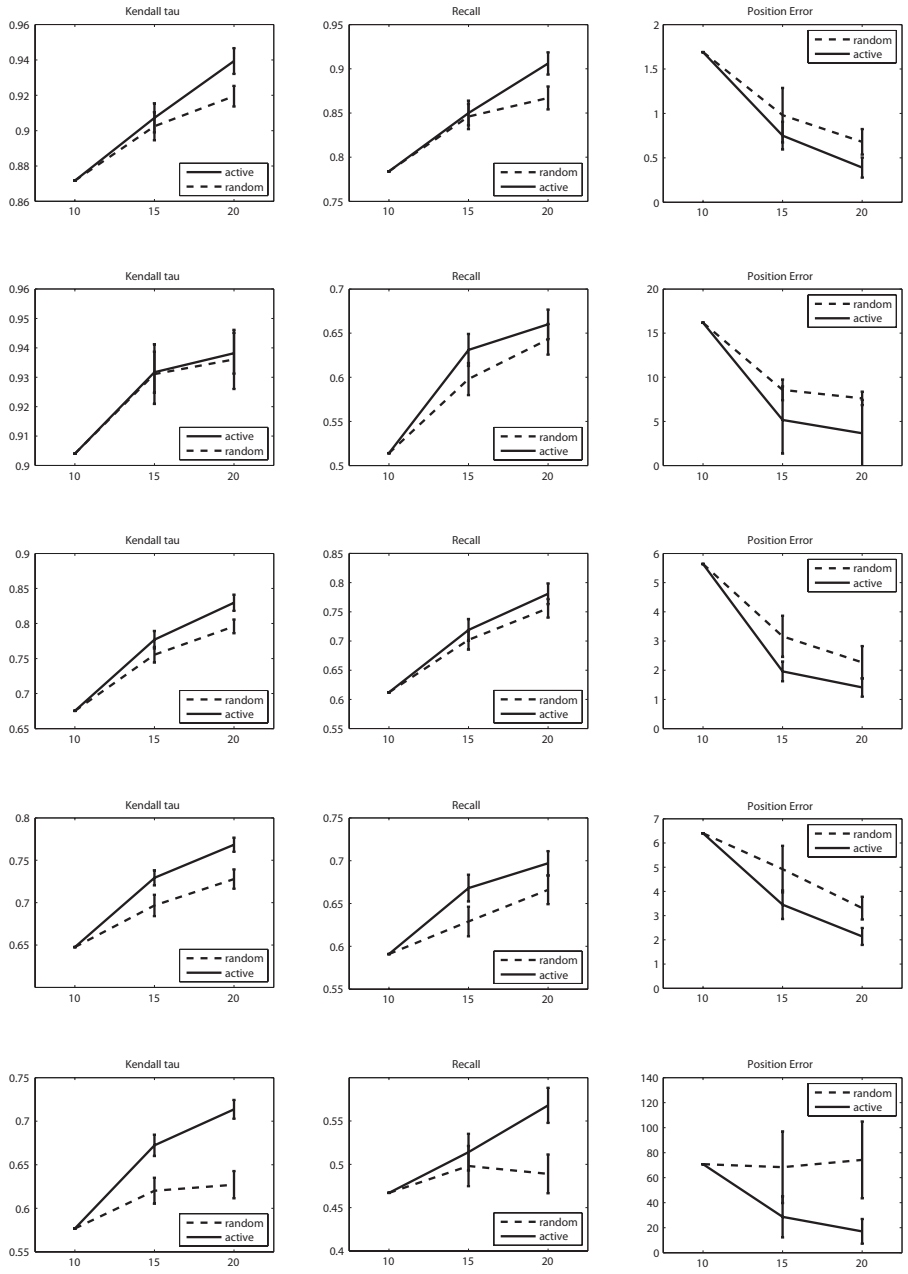

Fig. 4. Active vs. non-active learning: Experimental results in terms of rank correlation, recall, and position error as a function of the number of training examples (x-axis). Data sets from top to bottom: Iris, NBA, UNI, Wine, Yeast.