# Learning Feature Weights from Case Order Feedback

Armin Stahl

University of Kaiserslautern, Computer Science Department
Artificial Intelligence - Knowledge-Based Systems Group
67653 Kaiserslautern, Germany
stahl@informatik.uni-kl.de

**Abstract.** Defining adequate similarity measures is one of the most difficult tasks when developing CBR applications. Unfortunately, only a limited number of techniques for supporting this task by using machine learning techniques have been developed up to now. In this paper, a new framework for learning similarity measures is presented. The main advantage of this approach is its generality, because its application is not restricted to classification tasks in contrast to other already known algorithms. A first refinement of the introduced framework for learning feature weights is described and finally some preliminary experimental results are presented.

## 1 Introduction

Anyone who has implemented a case-based application knows that the definition of an adequate similarity measure is one of the most important tasks, but unfortunately, one of the most difficult tasks, too. Although similarity measures are often defined on a syntactical level only, the experience has shown that these similarity measures do not usually provide the best results. This means, one has to encode particular domain knowledge into the similarity measure to guarantee a suitable case retrieval. However, the definition of such semantic-oriented similarity measures is much more difficult than defining pure syntactical measures. One difficulty is the fact that the necessary domain knowledge is usually only available in form of a human domain expert. Nevertheless, such a domain expert has often problems to describe his knowledge in an explicit way as required for the definition of a similarity measure. One way to overcome this problem is a close collaboration of the domain expert with a CBR expert with the necessary experience to elicit the required knowledge form the domain expert. However, such close collaboration is time consuming work and leads to high development costs for the CBR application.

An additional problem arises if the intended CBR application is required to perform case adaptation. Then it is not enough to concentrate on the semantic similarity between the query and the case. As described in [9], one has to consider the adaptation procedure during the definition of the similarity measure to guarantee the retrieval of ’adaptable’ cases. However, the particular adaptation method to be used does usually not reflect the domain experts way of thinking. In this case, the domain expert can probably not give useful advise how to modify the semantic similarity measure regarding the adaptation procedure. Nevertheless, he should be able to evaluate the result of the adaptation procedure and to give useful feedback about the quality of the final solution.

In principle, such feedback can be used to simplify the acquisition of the similarity measure for a retrieval only system, too. Here, the domain expert can give some feedback on the quality of the retrieval result determined by an initial similarity measure. In this paper, we develop a new framework how to use such a kind of feedback to learn similarity measures more or less automatically.

In the next section we will present the basic framework for learning similarity measures by introducing an abstract concept, called ’similarity teacher’. We will discuss possible ways in which the role of the teacher might be filled. In the remainder of the paper we will apply this framework on learning feature weights which are one important part of almost every similarity measure. Therefore, we describe a new feature weight learning algorithm whose application is not limited to classification tasks as similar algorithms developed in the past. Due to the fact that many actual CBR applications go beyond simple classification systems, this is a crucial advantage of our learning framework. To demonstrate the basic capability of our approach, we present the results of a first basic evaluation. Finally, we will close with a short discussion of related work and further research objectives respecting the learning of similarity measures.

## 2 A Framework for Learning Similarity Measures

Several approaches for applying machine learning techniques in Case-Based Reasoning have been investigated in the past. Nevertheless, the primary focus was on the retain phase of the CBR cycle, i.e., the learning of new cases [1]. However, according to [8], the case base is not the only important *knowledge container* for a CBR system. Another very important knowledge carrier is the similarity measure, of course. Unfortunately, most approaches to learning the content of this knowledge container suffer from limited applicability because they strictly assume a CBR system used for classification [12,1,13,6,5,4]. Basically, the functionality of these learning algorithms requires a set of pre-classified cases. Due to this fact they are unable to handle cases with a non-classification solution part. Nevertheless, many currently successful commercial CBR systems do not perform pure classification, for example, the recommendation systems applied in e-Commerce [11].

In the following we will introduce a new framework for learning similarity measures, avoiding the requirement of the availability of pre-classified cases. The need for such a more general framework was already discussed in [2].

## 2.1 Similarity versus Utility

As discussed in [3], one can notice a difference in the semantics of the two concepts *similarity* and *utility* of cases. Basically, the similarity is only defined between the problem parts of two cases. In contrast, the utility is defined between a problem (usually called query) and the (known) solution part of a case, i.e., the utility is a measure for the usefulness of a solution for solving a given problem. Because the utility function is an a-posteriori criterion and often only partially known, in CBR the similarity function is used to approximate the utility function a-priori. However, a pure syntactical similarity considering only the problem part of a case is often not sufficient to obtain optimal retrieval results. This leads to the need of utility-oriented similarity measures that take the solution part of the case into account, too.

## 2.2 The Similarity Teacher

Consider the typical situation when using a CBR system as illustrated in Fig. 1. Given is a *query* representing a new problem of the application environment to be solved. Then, the retrieval function retrieves a set of cases from the *case base* by using the defined *similarity measure*. The *retrieval result*, i.e., a set of cases ordered by their similarity to the query, is finally presented to the user.

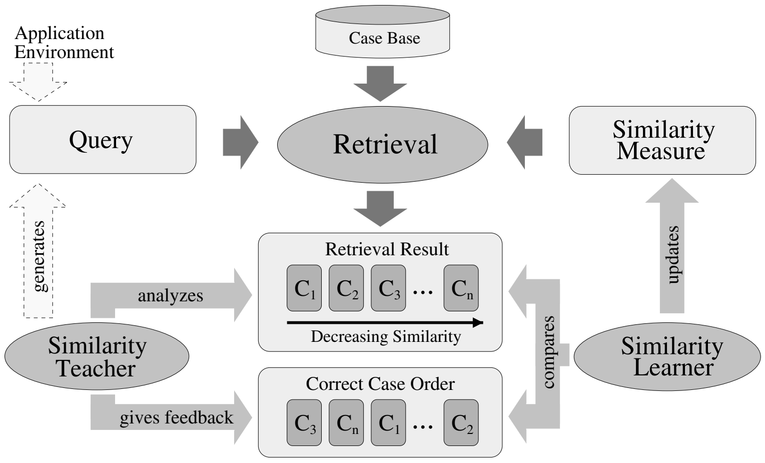

**Fig. 1.** A Framework for Learning Similarity Measures

The central element of our framework for learning similarity measures is a concept that we call *similarity teacher*. The job of this teacher is the evaluation of the determined retrieval result. Therefore, we assume that he possesses some implicit knowledge about the utility function described in Section 2.1. Due to this knowledge he can determine a kind of correct retrieval result, given through the *correct case order*. This means, we do not assume that he is able to correct the 'wrong' similarity values calculated by the similarity measure directly. Nevertheless, he should be able to order the retrieved cases correctly with respect to their utility for the query. This feedback information is then used by the *similarity learner* to update the previously used similarity measure. Therefore, the similarity learner has to compare the wrong retrieval result with the correct case order given by the teacher. During a similarity training phase, one can consider that the teacher generates special queries in order to obtain an optimal learning result. Depending on the concrete application scenario the similarity teacher can be realized differently. In the following we describe three typical examples.

**Human domain expert.** An obvious way to realize the similarity teacher is a human domain expert. As already mentioned before, a crucial problem during the development of a CBR system is the acquisition of the knowledge to be encoded in the similarity measure. The reason is the inability of many domain experts to formulate their knowledge in a formal way. However, they are usually able to give examples and to compare them. Therefore, we can assume that they can give the required feedback on the retrieval result, represented by the correct case order.

**Customer in e-Commerce.** Another situation occurs in many case-based e-Commerce applications. Here, it is often impossible to define an optimal similarity measure during the development of the application. The reason is that 'the optimal' similarity measure is usually determined by the preferences of the customers that are not known in advance by the developers of the e-Commerce system. This means that the system has to learn these preferences online to improve a pre-defined initial similarity measure. Of course, it is not very realistic to expect a customer to give explicit feedback about the complete retrieval result. However, due to his buying patterns he gives the system feedback in an implicit way. For example, if he finally buys a product with a lower similarity score than other recommended products, he partially defines the correct case order.

**Adaptation procedure.** The similarity teachers described in the two previous sections are both represented by human beings. However, the role of the similarity teacher can also be taken by an arbitrary adaptation procedure. As already mentioned in the introduction, the application of adaptation in CBR requires a special similarity measure because the utility of cases also depends on the adaptation method used. Thus utility can also be characterized as 'adaptability' [9]. Unfortunately, this complicates the definition of 'good' similarity measures enormously. However, the presented framework can help to overcome this problem. Therefore, we have to apply the given adaptation procedure on a set of retrieved cases. After that, the results of these adaptation attempts have to be evaluated, i.e., they have to be ordered respecting their utility for solving the problem specified by the query. The resulting case list can then again be interpreted as the correct case order on the originally retrieved cases. If we assume two different similarity measures, one for the basic case retrieval and one for the evaluation of the adaptation results, it is even possible to learn the first similarity measure in an automated way. In this way, the adaptation procedure coupled with the *evaluation similarity measure* that has not to consider the adaptability of cases, can be regarded as the similarity teacher. Here, the basic idea is the assumption that it is easier to define the evaluation measure than the adaptation-dependent retrieval measure. Of course, our approach can be applied without such an evaluation measure if the correct case order is determined by a human domain expert again. Nevertheless, one can imagine within our framework an approach in which the evaluation measure is first learned from a domain expert.

### 2.3 The Similarity Learner

Concerning the similarity learner, we suppose an arbitrary learning algorithm that is able to perform the kind of supervised learning described before. Depending on the structure of the similarity measures to be used, this algorithm may apply different machine learning techniques. In the following we assume attribute-value based case representations and a similarity measure consisting of three major parts:

1. a number of *local similarity measures* used to compare the values of single attributes
2. a number of *feature weights* representing the relative importance of each attribute
3. a *global similarity measure* responsible for the computation of a final similarity value base on the local similarities and feature weights

In the next section we will describe a first refinement of our similarity learning framework. Due to the fact that the learning of feature weights is an established area in machine learning, we will focus on this part of the similarity measure in the remainder of this paper. Therefore, we show how to implement an algorithm using the conjugate gradient method respecting the introduced learning framework.

## 3 Learning Feature Weights

### 3.1 Basics

To apply our feature weight learning approach we assume a flat attribute-value based case representation, i.e., a case $c = (v_1^c, \ldots, v_n^c)$ can be described as a vector with $n$ values where $v_i^c$ is the value of feature $f_i$. With respect to the similarity computation between a query and a case we introduce the following definition:

**Definition 1 (Similarity).** *Given a query $q = (v_1^q, \ldots, v_n^q)$ and a case $c = (v_1^c, \ldots, v_n^c)$, the similarity between $q$ and $c$ is defined as follows:*

$$Sim(q,c) = \sum_{i=1}^{n} w_i \cdot sim_i(v_i^q, v_i^c) = \sum_{i=1}^{n} w_i \cdot sim_i(q,c)$$

*where $w_i \in [0,1]$ is the weight, and $sim_i$ is the local similarity measure of feature $f_i$. It holds $\sum_{i=1}^{n} w_i = 1$ and $sim_i(v_i^q, v_i^c) \in [0,1]$, and therefore, it holds $Sim(q,c) \in [0,1]$, too. To simplify the notation we write $sim_i(q,c)$ instead of $sim_i(v_i^q, v_i^c)$ in the following.*

Because we are mainly interested in the result of a retrieval, i.e., the similarity computation between a query and a set of cases, we now give a definition for the *retrieval result.*

**Definition 2 (Retrieval Result).** *Given a case base $CB = \{c_1, \ldots, c_m\}$, a query $q$ and a similarity function $Sim$, we define the corresponding retrieval result as*

$$C_r^{Sim}(q, CB) = (c_1, \ldots, c_r)$$

*where $c_i \in CB$ and $Sim(q, c_i) \geq Sim(q, c_j)$ $\forall i, j$ with $1 \leq i < j \leq r$.*

This means, the retrieval result $C_r^{Sim}(q, CB)$ consists of a subset of $r$ cases out of the case base $CB$ (partially) ordered by their similarity to the given query $q$. We do not explicitly define which cases have to appear in the retrieval result, i.e., the retrieval result can contain an arbitrary case selection out of the case base. A possible selection may consist of the $r$ most similar cases with $r \leq m$.

The case order in the retrieval result is determined by the similarity measure $Sim$ representing an approximation of the utility function (see Section 2.1). Because the core of our interest is the difference between these two functions, we now introduce the concept of the *feedback function*:

**Definition 3 (Feedback Function).** *Let $Q$ be the set of all queries. The function $fb : Q \times (CB \times CB) \rightarrow \{0,1\}$ defined as*

$$fb(q, (c_1, c_2)) := \begin{cases} 1 & if \quad U(q, c_2) > U(q, c_1) \\ 0 & otherwise \end{cases}$$

*is called the feedback function. $U(q, c_i)$ is supposed to be a (possibly informal) measure of the utility of a case $c_i$ with respect to a query $q$.*

The feedback function $fb$ evaluates the utility of two cases given by the tuple $(c_1, c_2)$ with respect to a query $q$. If $c_2$ has a higher utility than $c_1$ it returns a 1, otherwise it returns a 0. Because the utility function $U$ is possibly an informal measure we do not necessarily suppose that $U(q, c_i)$ is computable. However, we assume the existence of a 'teacher' who is able to compare two cases $(c_1, c_2)$ with respect to their utility for a given query $q$. This means, the teacher can give feedback regarding which of the two cases has the higher utility for $q$.

By using the feedback function $fb$ we are now able to give a definition for the *correct case order*:

**Definition 4 (Correct Case Order).** *Let $CB = \{c_1, \ldots, c_m\}$ be a case base and $q$ be a query. The correct case order is defined as*

$$C^U(q, CB) = (c_1, \ldots, c_m)$$

*where $c_i \in CB$ and $fb(q, (c_i, c_j)) = 0 \; \forall i, j$ with $1 \le i < j \le m$.*

Because in the following we suppose a constant case base $CB$, we simply write $C_r^{Sim}(q)$ instead of $C_r^{Sim}(q, CB)$ and $C^U(q)$ instead of $C^U(q, CB)$.

After the introduction of these basic definitions, we will now discuss the core aspect of our weight learning approach, namely the error function used by the conjugate gradient learning algorithm to be described in Section 3.3.

## 3.2 Definition of the Error Function

As already described in Section 2, the goal of our similarity learner is to optimize the similarity measure in such a way that the case order in the retrieval result is equal to the correct case order given by the teacher. To be able to measure the difference between the correct case order $C^U$ and the retrieval result $C_r^{Sim}$ determined by the similarity measure $Sim$, we can define a special error function. A first attempt for the definition of such an error function is the *index error*:

**Definition 5 (Index Error).** *Consider a query $q$, a similarity function $Sim$ and the corresponding retrieval result $C_r^{Sim}(q)$. We define the index error as*

$$E_I(q, C_r^{Sim}(q)) = \sum_{k=1}^{r-1} \sum_{l=k+1}^{r} fb(q, (c_k, c_l))$$

The index error can be interpreted as a measure for the 'disorder' in the retrieval result $C_r^{Sim}$ in comparison to the correct case order $C^U$. If the two partial orders $C_r^{Sim}(q)$ and $C^U(q)$ are equal, it can be seen that $E_I = 0$. The maximum possible index error is obtained if $C_r^{Sim}(q)$ is the reverse order of $C^U(q)$.

However, if we want to apply a conjugate gradient algorithm for learning feature weights, the previously introduced error function is not usable. The reason for this is that the index error $E_I$ is not partially derivable w.r.t. a weight $w_i$. In the following we define an alternative error function in order to overcome this problem. First, we give a definition for the *similarity error for a case pair*:

**Definition 6 (Similarity Error for a Case Pair).** *Let $q$ be a query, $Sim$ be a similarity measure, and $(c_1, c_2)$ be a case pair with $Sim(q, c_1) \ge Sim(q, c_2)$. We define the similarity error for the case pair as*

$$\begin{aligned} E_{Sim}(q, (c_1, c_2)) &= (Sim(q, c_1) - Sim(q, c_2)) \cdot fb(q, (c_1, c_2)) \\ &= (\sum_{i=1}^{n} w_i \cdot (sim_i(q, c_1) - sim_i(q, c_2))) \cdot fb(q, (c_1, c_2)) \end{aligned}$$

In the next step we extend this definition to a *similarity error for a retrieval result*:

**Definition 7 (Similarity Error for a Retrieval Result).** *Let $q$ be a query and $C_r^{Sim}(q)$ be the corresponding retrieval result. The similarity error of the retrieval result is defined as*

$$E_{Sim}^{\alpha}(q, C_r^{Sim}(q)) = \sum_{k=1}^{r-1} \sum_{l=k+1}^{r} E_{Sim}(q, (c_k, c_l)) \cdot (l-k)^{\alpha}$$

*where $(l-k)^{\alpha}$ is called the index-distance weight which can be influenced by the parameter $\alpha \geq 0$.*

The index-distance weight influenced by the parameter $\alpha$ can be used to manipulate the similarity error depending on the degree of 'disorder' in the retrieval result compared with the correct case order. Generally, a greater $\alpha$ leads to an increased impact of the disorder on the similarity error.

If we are able to change the similarity measure $Sim$ in such a way that the similarity error $E_{Sim}^{\alpha}(q, C_r^{Sim}(q))$ for a query $q$ becomes 0, then we have reached our goal - but only for this particular query $q$. However, we want to optimize the similarity measure $Sim$ in a more global manner, i.e., we are searching for a similarity measure $Sim$ leading to a similarity error $E_{Sim}^{\alpha}(q_i, C_r^{Sim}(q_i)) = 0$ for any arbitrary query $q_i$. This leads us to the following definitions:

**Definition 8 (Retrieval Collection).** *Let $Q_s = \{q_1, \ldots, q_s\}$ be a set of queries. The set*

$$\hat{C}^{Sim}(Q_s) = \{(q_1, C_{r_1}^{Sim}(q_1)), \ldots, (q_s, C_{r_s}^{Sim}(q_s))\}$$

*is called the retrieval collection for $Q_s$ where $C_{r_j}^{Sim}(q_j) = \{c_{1_j}, \ldots, c_{r_j}\}$ is the corresponding retrieval result for query $q_j$.*

**Definition 9 (Average Similarity Error).** *The average similarity error for a retrieval collection $\hat{C}^{Sim}(Q_s)$ is defined as*

$$\begin{aligned} \hat{E}_{Sim}^{\alpha}(\hat{C}^{Sim}(Q_s)) &= \frac{1}{s} \cdot \sum_{j=1}^{s} E_{Sim}^{\alpha}(q_j, C_{r_j}^{Sim}(q_j)) \\ &= \frac{1}{s} \cdot \sum_{j=1}^{s} \sum_{k=1}^{r_j-1} \sum_{l=k+1}^{r_j} E_{Sim}(q_j, (c_{k_j}, c_{l_j})) \cdot (l-k)^{\alpha} \qquad (1) \end{aligned}$$

Now we are able to describe our learning goal by using the average similarity error $\hat{E}_{Sim}^{\alpha}$. Suppose that we have a set of queries $Q_s = \{q_1, \ldots, q_s\}$. To get an adequate similarity measure $Sim$ with respect to the utility function given by the teacher, we have to minimize the average similarity error $\hat{E}_{Sim}^{\alpha}(\hat{C}^{Sim}(Q_s))$. If we are able to construct a $Sim$ leading to an average similarity error of $\hat{E}_{Sim}^{\alpha} = 0$, we can call $Sim$ an *optimal similarity measure* respecting $Q_s$.

From the machine learning point of view we can characterize the set of queries $Q_s$ together with the corresponding correct case orders $C^U(q_i)$ given by the teacher as the *training data* for our learning algorithm.

## 3.3 The Learning Algorithm

After the introduction of the average similarity error $\hat{E}^{\alpha}_{Sim}$ we will now describe an algorithm that tries to minimize this error function by adjusting the feature weights $w_i$ as part of the similarity measure $Sim$. Therefore, we use a conjugate gradient algorithm performing an iterative search for a local minimum of our error function $\hat{E}^{\alpha}_{Sim}$. To apply this algorithm we assume a starting situation given by:

- initial similarity measure $Sim$ with feature weight-vector $w = (w_1, \ldots, w_n)$
- query set $Q_s$
- case base $CB$
- a 'similarity teacher' (see Section 2.2)
- a fixed value $\alpha \geq 0$

We assume that the similarity teacher is able to give the algorithm his knowledge about the utility function $U$ by realizing the feedback function $fb$ introduced in definition 3. Assuming this starting situation we can describe the basic learning algorithm as follows:

```
1. Initialize weight-vector $w$
2. Start retrieval procedure for $Q_s$ to determine $\hat{C}^{Sim}(Q_s)$
3. Compute average similarity error $\hat{E}^{\alpha}_{Sim}(\hat{C}^{Sim}(Q_s))$
4. Initialize learning rate $\lambda$
5. **While** stop-predicate = false **do**
   a) Generate new $Sim'$: $\forall i \; w'_i := w_i - \frac{\partial \hat{E}^{\alpha}_{Sim}(\hat{C}^{Sim}(Q_s))}{\partial w_i} \cdot \lambda$
   b) Normalize $w'_i := \frac{w_i}{\sum_{j=1}^{n} w_j}$
   c) Start retrieval procedure for $Q_s$ to determine $\hat{C}^{Sim'}(Q_s)$
   d) Compute average similarity error $\hat{E}^{\alpha}_{Sim'}(\hat{C}^{Sim'}(Q_s))$
   e) **If** $\hat{E}^{\alpha}_{Sim'}(\hat{C}^{Sim'}(Q_s)) < \hat{E}^{\alpha}_{Sim}(\hat{C}^{Sim}(Q_s))$
      **then** $Sim := Sim'$
      **else** $\lambda := \frac{\lambda}{2}$
6. Output: New similarity measure $Sim$
```

Basically, the algorithm can be separated into two phases which are iteratively repeated until the stop-predicate becomes true. In the first phase the algorithm performs the retrievals for the given query set $Q_s$ by using the actual similarity measure $Sim$ including the actual weight-vector $w$. In the second phase the algorithm uses the feedback of the teacher to calculate the error function $\hat{E}^{\alpha}_{Sim}$. Depending on the result of this error function a new similarity measure $Sim'$ with a new weight-vector $w'$ is computed. If the next iteration shows an improvement in the error function, $Sim'$ is accepted, i.e., $Sim := Sim'$. Else the learning rate is decreased by $\lambda := \frac{\lambda}{2}$.

As typical for a conjugate gradient algorithm, it uses the derivation of the error function to update the feature weights. According to equation 1, the partial derivation of the average similarity error $\hat{E}^{\alpha}_{Sim}$ w.r.t. the weight $w_i$ is as follows:

$$\frac{\partial \hat{E}^{\alpha}_{Sim}(\hat{C}^{Sim}(Q_s))}{\partial w_i} = \frac{1}{s} \cdot \sum_{j=1}^{s} \sum_{k=1}^{r_j-1} \sum_{l=k+1}^{r_j} (sim_i(q_j, c_{k_j}) - sim_i(q_j, c_{l_j})) \cdot$$
$$\cdot fb(q, (c_{k_j}, c_{l_j})) \cdot (l-k)^{\alpha}$$

**Initialization of feature weights.** As in other algorithms that apply a conjugate gradient method, the initialization of the start point, here given by the initial weights, is important for the success of the approach. In general, it cannot be guaranteed that the algorithm is able to find the global minimum of the error function. However, a 'good' initialization of $w$ can enable the algorithm to find the global minimum or at least a relatively low local minimum. Basically, one can distinguish between three possible approaches to initialize $w$:

1. use an uniform weight-vector, i.e., $\forall i \; w_i = \frac{1}{n}$
2. initialize the weights randomly
3. a domain expert defines the initial weights

Of course, the third approach is usually the best choice because a domain expert should be able to initialize the weights in a more 'intelligent' way by using his domain knowledge.

**The role of the learning rate.** Similar to other machine learning approaches, the choice of the learning rate $\lambda$ is crucial for the outcome of the algorithm. Generally, we can notice a tradeoff between a very small and a too large $\lambda$. A quite small $\lambda$ leads to a poor convergence speed, however, the advantage of a small $\lambda$ is a higher probability of finding the next local minimum with respect to $w$. This is important if the weights were initialized by a domain expert because this local minimum should normally correspond to a very accurate weight-vector. On the other hand, if $\lambda$ is too large, the algorithm will 'jump over' this nearby minimum. This leads to the risk that the algorithm will 'get caught' in another local minimum corresponding to a much higher error value.

To simplify the selection of an appropriate initial learning rate, we have introduced a normalization with respect to the resulting change of the weights in the first learning step. We allow the user of the learning algorithm to determine a maximal change in one of the weights $w_i$ represented by $\Delta_{max}w$. The corresponding initial learning rate $\lambda$ is then computed as follows:

$$\lambda = \frac{\Delta_{max}w}{max\{\Delta w_i\}} \quad \text{where} \quad \Delta w_i = \frac{\partial \hat{E}^{\alpha}_{Sim}}{\partial w_i}$$

Consider the example of a domain with four features and an initial weight-vector $w = (0.25, 0.25, 0.25, 0.25)$. If the user determines $\Delta_{max}w = 0.05$, the new weight-vector computed in the first learning step will at least contain one weight $w_i$ with $w_i = 0.2$ or $w_i = 0.3$. For all other weights $w_j$ holds $w_j \in [0.2, 0.3]$.

A further improvement with respect to the determination of an appropriate learning rate is the introduction of a *dynamic learning rate* adapted for every single learning step. Up to now, we have assumed that the initial learning rate is only changed if a learning step increases the value of the error function, i.e., $\hat{E}^{\alpha}_{Sim'} > \hat{E}^{\alpha}_{Sim}$. Then the learning rate is halved to guarantee the convergence of the learning algorithm. During the experiments we have made the observation that we can improve the learning efficiency by using a dynamic learning rate that depends on the actual value of $\hat{E}^{\alpha}_{Sim}$. The basic idea of this approach is the following heuristic: the larger the error $\hat{E}^{\alpha}_{Sim}$, the more the weights have to be changed. This heuristic can be realized by replacing $\lambda$ by $\lambda \cdot \hat{E}^{\alpha}_{Sim}$ in step 5.a of the algorithm. Of course, then we have also to update the normalization of the initial learning rate with respect to the additional factor $\hat{E}^{\alpha}_{Sim}$.

**The stop-predicate.** To guarantee the termination of the algorithm we have to introduce a stop-predicate. Generally, various possibilities to realize this stop-predicate exist [13]. For our experiments we have selected a minimal change of the error function.

## 4 Experiments

The primary motivation for the development of the introduced approach for learning similarity measures was the perception that any sophisticated adaptation approach requires a special similarity measure to retrieve the best adaptable cases. In [10] for example, we have presented a compositional adaptation method used to perform case-based configuration of personal computers. However, for a first evaluation of our learning algorithm we have decided to use a very simple artificial test domain, due to the huge complexity of the PC-domain. Nevertheless, we think that the results obtained from this test domain are sufficient to demonstrate the basic capability of our approach.

### 4.1 An Artificial Test Domain

The used case structure consists of 5 attributes, each of which has a value range of [0,100]. The local similarity measure for each attribute is a quite simple distance-based function:

$$sim_i(q,c) = sim_i(v_i^q, v_i^c) := 1 - \frac{|v_i^q - v_i^c|}{100}$$

The global similarity measure is a weighted sum as defined in Definition 1 with the particular weight-vector $w = (w_1, w_2, w_3, w_4, w_5)$. For the initial weights we have chosen a uniform $w^I$, i.e., $\forall i \; w_i^I = 0.2$. To get the necessary case base $CB$ we have generated 100 cases randomly. All cases are completely filled, i.e., every attribute has been assigned a random value between 0 and 100.

## 4.2 The Test Scenario

For our experiments we have realized the similarity teacher in the form of a pre-defined *target weight-vector* $w^T$. This means, the assumed similarity measure with this special weight-vector determines the correct case order for a given query. Of course, this situation will not appear in real world application scenarios due to the not explicitly known learning target. However, it enabled us to perform the experiments automatically and without using a complex adaptation procedure. Another advantage of this approach is the possibility to compare the learned weight-vectors with the correct target weights.

To get an impression of the amount of training data needed to obtain satisfactory learning results, we have performed a large number of learning cycles with a varying size $s$ of the query sets $Q_s$. Further, we have varied the size $r$ of the retrieval results $C_r^{Sim}(q, CB)$ used by the learning algorithm. Therefore, we have selected some cases out of the retrieval result (complete $CB$) returned by CBR-Works, the CBR tool used in our experiments. To simulate a realistic application scenario, we have restricted the retrieval result to between 5 and 25 cases. This corresponds to the necessary procedure if one wants to apply our learning approach in a real world application. On the one hand, a domain expert could not be expected to determine the correct case order for all cases in a large case base. On the other hand, the application of a sophisticated adaptation procedure on all cases would perhaps be limited by computational complexity. The concrete selection of the cases to be used in our experiments was again realized randomly with the restriction that the most and the least similar cases are always selected.

For the different combinations of $s$ and $r$ we recorded the finally learned weight-vectors $w^L$, i.e., the weight-vectors calculated so far when the stop-predicate became true. To evaluate the quality of these weight-vectors, we used two measures:

1. the *average index error* $\hat{E}_I(Q_{1000}^T) = \frac{1}{1000}\sum_{j=1}^{1000} E_I(q_j, C_m^{Sim}(q_j))$ (see definition 5) for a fixed test query set $Q_{1000}^T$ consisting of 1000 randomly generated queries and $m = |CB|$. To get meaningful evaluation results, the test query set is different from the training query sets used during the learning phase.
2. an additional *weight error* $E_W(w^L, w^T) = \sum_{i=1}^{n} |w_i^T - w_i^L|$

Because the quality of the results varies for different query sets (especially for small $s$) we have repeated our experiments several times with different randomly generated $Q_s$. Finally, we determined the *best*, the *average* and the *worst learning result* for each combination of $s$ and $r$. Due to the lower variance for large $s$, we decreased the number of iterations while increasing $s$. Concerning the different learning parameters introduced in section 3.3, we selected the following values: $\Delta_{max} w = 0.1$ and $\alpha = 1$.

## 4.3 Results

Fig. 2 shows the resulting average index error $\hat{E}_I(Q_{1000}^T)$ for three different values of $r$ and four different values of $s$. A legend like '100x10' states that the results were obtained by repeating the learning procedure for 100 different query sets $Q_s$ with $s = 10$, i.e., the first number represents the number of experiments while the second number represents the size of the query set used. Additionally, we have shown the value of $\hat{E}_I(Q^T_{1000})$ for the initial weights $w^I$.

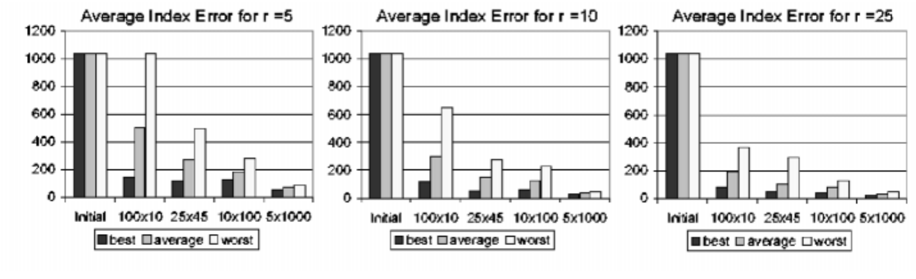

**Fig. 2.** Evaluation Results

One expected result is the fact that larger query sets and larger retrieval results lead to much better learning results because then the algorithm gets more information to find a globally valid weight-vector.

Another very important observation is the wide variation in the results of different identical experiments especially for small $s$. For example, the best index error for the '100x10/$r$=5' experiment is 144 while the worst index error is 1041. This suggests that the efficiency of the learning algorithm could significantly be improved by using 'good' $Q_s$ instead of randomly generated ones.

The results for the three different values for $r$ show that smaller retrieval results require a significant larger number of queries to obtain comparable results as with large retrieval results. In principle, a domain expert has to perform the same number of case-pair comparisons for the '$s$=10/$r$=10' and the '$s$=45/$r$=5' experiment. And in fact, the observed learning results are very similar. Therefore, dependent on the particular application, it is possible to obtain the necessary training data either by a large number of queries or by a large number of analyzed cases, i.e., by the determination of the correct case order for many cases.

Finally, in Table 1 we have summarized some of the learned weight-vectors in comparison to the target weights $w^T$ and the initial weights $w^I$. The last two rows of the table contain the weight errors $E_W$ for the particular learned weights or the average weight error $\emptyset E_W$ for all weight-vectors of one experiment respectively. Here, we can notice similar results as in the diagrams discussed before. Again, the size and the quality of the query set $Q_s$ and the value of $r$ are the major criteria for obtaining weight-vectors that are close to the target weights. Another observation is the lower precision in the learning results for the relative small weights. This can be explained with their minor impact on the error function.

**Table 1.** Comparison of Learned Weight-Vectors

| | | | **Learned Weight-Vectors** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $r$=5 | | | | $r$=10 | | $r$=25 | | | |
| | | | 100×10 | | 25×45 | | 100×10 | | 25×45 | | 5×1000 | |
| $w_i$ | $w^T$ | $w^I$ | best | worst | best | worst | best | worst | best | worst | best | worst |
| $w_1$ | **0.033** | 0.2 | 0.046 | 0.2 | 0.045 | 0.124 | 0.046 | 0.082 | 0.028 | 0.07 | 0.03 | 0.032 |
| $w_2$ | **0.33** | 0.2 | 0.36 | 0.2 | 0.299 | 0.258 | 0.316 | 0.325 | 0.327 | 0.296 | 0.334 | 0.334 |
| $w_3$ | **0.033** | 0.2 | 0.05 | 0.2 | 0.056 | 0.087 | 0.04 | 0.073 | 0.047 | 0.107 | 0.04 | 0.046 |
| $w_4$ | **0.2** | 0.2 | 0.169 | 0.2 | 0.208 | 0.241 | 0.204 | 0.175 | 0.197 | 0.19 | 0.199 | 0.189 |
| $w_5$ | **0.4** | 0.2 | 0.375 | 0.2 | 0.388 | 0.291 | 0.393 | 0.345 | 0.4 | 0.336 | 0.396 | 0.399 |
| $E_W$ | 0.0 | 0.667 | 0.11 | 0.667 | 0.092 | 0.369 | 0.087 | 0.45 | 0.029 | 0.221 | 0.015 | 0.027 |
| $\emptyset E_W$ | - | - | 0.345 | | 0.193 | | 0.206 | | 0.076 | | 0.02 | |

## 5 Related Work and Outlook

Some basic considerations concerning the learning of similarity measures can be found in [7]. Particularly, the learning of feature weights is a classic topic in the area of machine learning. A very good overview of different approaches used in CBR is given in [12]. Another very interesting approach that learns asymmetric similarity measures by adjusting two different weight-vectors (one for every 'side' of each local measure) is described in [6]. An algorithm that considers the costs of the decisions made with the learned weights is proposed in [13]. An approach that uses only 'boolean' feedback, i.e. the information whether the solution of a retrieved case is the same as the target solution or not, is described in [4]. However, all these approaches are restricted to classification tasks.

A more general approach is presented in [14]. Here, the teacher is able to give the system feedback about the desired score of a case. Nevertheless, the use of a special two-layer network architecture to model a case-base leads to some limitations. For example, only discrete feature value ranges can be represented and no local similarity measures to compare feature values are supported.

In this paper we have presented a framework for learning similarity measures independent of the application task. Further, we have described a first application of our framework for learning feature weights. Due to the evaluation results, we have seen that 'good' query sets are one crucial factor affecting the quality of the obtained learning results. Therefore, we are planning to investigate how to determine such 'good' queries. Additionally, it will be helpful to apply our algorithm to real-world problems and to perform more sophisticated evaluation experiments. Further, it is our goal to apply the framework to learning local similarity measures instead of learning feature weights only.

In our view, the ability to learn similarity measures more or less automatically will be a significant advance in the application of Case-Based Reasoning in real world applications. However, due to the lack of generally applicable approaches, we see an urgent need for further research in this area.

**Acknowledgements.** The author would like to thank Ralph Bergmann for the helpful discussions during the development of the presented approach. This work was funded by the European Commission (GROWTH programme, contract number GRD1-1999-10385, the FIORES-II project: Character Preservation and Modelling in Aesthetic and Engineering Design).

## References

1. D. Aha. Case-based learning algorithms. In *Proceedings of the DARPA Case-Based Reasoning Workshop*, pages 147–158. Morgan Kaufmann, 1991.
2. D. W. Aha and D Wettschereck. Case-based learning: Beyond classification of feature vectors. In *Proceedings of the 9th European Conference on Machine Learning (ECML'97)*. Springer, 1997.
3. R. Bergmann, M. Michael Richter, S. Schmitt, A. Stahl, and I. Vollrath. Utility-oriented matching: A new research direction for Case-Based Reasoning. In *Professionelles Wissensmanagement: Erfahrungen und Visionen. Proceedings of the 1st Conference on Professional Knowledge Management.* Shaker, 2001.
4. A. Bonzano, P. Cunningham, and B. Smyth. Using introspective learning to improve retrieval in CBR: A case study in air traffic control. In *Proceedings of the 2nd International Conference on Case-Based Reasoning (ICCBR-97).* Springer, 1997.
5. Igor Kononenko. Estimating attributes: Analysis and extensions of RELIEF. In *Proceedings of the European Conference on Machine Learning*, pages 171–182, 1994.
6. F. Ricci and P. Avesani. Learning a local similarity metric for case-based reasoning. In *Proceeding of the 1st International Conference on Case-Based Reasoning (ICCBR'95)*, pages 301–312. Springer, 1995.
7. Michael M. Richter. Classification and learning of similarity measures. Technical Report SR-92-18, 1992.
8. Michael M. Richter. The knowledge contained in similarity measures. Invited Talk at ICCBR-95, 1995.
9. B. Smyth and M. T. Keane. Retrieving adaptable cases: The role of adaptation knowledge in case retrieval. In *Proceedings of the 1st European Workshop on Case-Based Reasoning.* Springer, 1993.
10. A. Stahl and R. Bergmann. Applying recursive CBR for the customization of structured products in an electronic shop. In *Proceedings of the 5th European Workshop on Case-Based Reasoning.* Springer, 2000.
11. M. Stolpmann and S. Wess. *Optimierung der Kundenbeziehung mit CBR-Systemen.* Business and Computing. Addison-Wesley, 1999.
12. Dietrich Wettschereck and David W. Aha. Weighting features. In *Proceeding of the 1st International Conference on Case-Based Reasoning (ICCBR'95)*, pages 347–358. Springer Verlag, 1995.
13. W. Wilke and R. Bergmann. Considering decision cost during learning of feature weights. In *Proceedings of the 3rd European Workshop on Case-Based Reasoning.* Springer, 1996.
14. Z. Zhang and Q. Yang. Dynamic refiniement of feature weights using quantitative introspective learning. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI 99)*, 1999.