Top-Down Induction of Similarity Measures Using Similarity Clouds

Thomas Gabel(✉) and Eicke Godehardt

Faculty of Computer Science and Engineering,
Frankfurt University of Applied Sciences, 60318 Frankfurt am Main, Germany
{tgabel,godehardt}@fb2.fra-uas.de

Abstract. The automatic acquisition of a similarity measure for a CBR system is appealing as it frees the system designer from the tedious task of defining it manually. However, acquiring similarity measures with some machine learning approach typically results in some black box representation of similarity whose magic-like combination of high precision and low explainability may decrease a human user's trust in the system. In this paper, we target this problem by suggesting a method to induce a human-readable and easily understandable – and thus potentially trustworthy – representation of similarity from a previously learned black box-like representation of similarity measures. Our experimental evaluations support the claim that, given some highly precise learned similarity measure, we can induce a less powerful, but human-understandable representation of it while its corresponding level of accuracy is only marginally impaired.

1 Introduction

Similarity measures represent an integral part of a CBR system, but providing an accurate and suitable definition of these functions represents a difficult task for any designer of a case-based application. A natural way out of this problem is to cast the task as a function learning problem and, instead of defining similarity measures by hand, to apply some machine learning algorithm to generate useful similarity measures. Different such learning techniques exist, each of which comes up with its own model for representing the learned function. A common characteristic of various machine learning methods is, however, that the models learned represent black boxes from the user's perspective.

A substantial portion of the success and the acceptance of CBR systems by users can be tributed to the fact that case-based systems do inherently generate trust. The solutions suggested by a CBR system always relate to previous experience and the reasons why a specific solution is proposed to a user are, in general, easily explainable and, hence, perceived to be trustworthy.

If machine learning techniques are employed for the acquisition of highly accurate similarity measures and if, in doing so, these measures become represented by some black box machine learning model, then understandability and traceability of the CBR inference process are reduced and, as a consequence, the trustworthiness of the system is impaired. This is exactly the point we want to

E. Hüllermeier and M. Minor (Eds.): ICCBR 2015, LNAI 9343, pp. 1–16, 2015.
DOI: 10.1007/978-3-319-24586-7_11

address in the context of this paper. We first propose a powerful machine learning approach based on neural networks for learning high-quality, but black box-like similarity measures (Sect. 2). In a second step, we capture the essence of these neural network-based similarity measures in so-called similarity clouds and utilize these to induce easily interpretable similarity measures that are represented using some established human-readable formalism (Sect. 3). Finally (Sect. 4), we evaluate our approach in the context of a large number of benchmark application domains and, in so doing, analyze the trade-off made between a highly accurate black-box representation and a mapping to a human-readable, but probably less powerful representation of similarity measures.

2 Neural Similarity Measures

Artificial neural networks are known for their excellent performance in different areas of machine learning. Specifically, multi-layer perceptron neural networks have been shown to be universal function approximators [12]. Recent advances in the training of so-called "deep architectures" have once more boosted the attention to neural network-based learning architectures for high-dimensional input data [11]. Thus, employing neural networks for the representation of similarity measures seems to be a natural choice. While neighboring research communities have frequently addressed the topic of representing distance or similarity measures with neural networks (cf. Sect. 2.4) the core CBR community and conferences have paid comparatively little attention to that topic so far.

2.1 Multi-layer Perceptron Neural Networks

A multi-layer perceptron is an artificial neural network whose units (perceptrons) are connected in an acyclic graph. All of its neurons are arranged in layers that are disjoint from one another in that there are no connections among units within the same layer and that two successive layers are fully connected with one another. Data is propagated through the network (forward propagation) by providing inputs to the network's first layer (input layer) and, subsequently, calculating the activations of all neurons in all successive layers (hidden layers) till the final, so-called output layer. For a given training set

$$\mathbb{P} = \{(x^p, t^p) | p \in \{1, \dots, |\mathbb{P}|\}\} \tag{1}$$

of training patterns (x^p, t^p) with input vectors $x^p = (x_1^p, \dots, x_m^p) \in \mathbb{R}^m$ and target values $t^p = (t_1^p, \dots, t_n^p) \in \mathbb{R}^n$, a multi-layer perceptron can be trained using the back-propagation algorithm which essentially performs a gradient descent-based adaptation of the net's connection weights such that the error

$$E = \sum\nolimits_{p=1}^{|\mathbb{P}|} \sum\nolimits_{i=1}^{n} (t_i^p - o_i^p)^2$$

is minimized where o^p denotes the net's output under input of pattern x^p [19].

When training neural networks, the resilient propagation update rule (Rprop [17]) has frequently been shown to provide robust and convincing results. Rprop is a batch method which calculates the gradient of the error as in standard back-propagation, but which does not use the magnitude of the gradient, but its direction for determining the weight change. To this end, the step length of a weight change is calculated by a simple heuristic that is stored separately for each connection weight. If the direction of the gradient has been the same in successive update steps, then the step width is incremented, if the sign of the gradient changes, however, the step width is decremented. An appealing feature of Rprop is that it introduces relatively few parameters and that the setting of these parameters has been shown to be quite robust with respect to the results obtained. When training neural networks within this work, we stick to the use of Rprop with its default parameter setting published in [17].

2.2 Supervised Training of Neural Net-Based Similarity Functions

In the remainder of this paper we focus on case characterizations with m describing attributes $A_1, \ldots, A_m$ and an additional solution attribute A_s. While the methods we present are generic enough to accommodate complex solutions (e.g. object-oriented or multi-dimensional ones), we will specifically focus on the case where the solution can be represented by a single value, e.g. by a class label from a finite set $D_{A_s} = \{g_1, \ldots, g_k\}$ in classification tasks or by a numeric value in regression tasks. More importantly, in what follows we will use the notation $c = (c_p, c_s)$ when speaking about case c and, in doing so, emphasize the distinction between the case's problem and solution part.

2.2.1 Utility Feedback and Case Order Feedback

In [21], a framework for learning similarity measures is described which we have used for our research repeatedly in the previous years. Its core ideas are that

1. some "similarity teacher" provides information (utility feedback) about the desired order of cases as it should result from a retrieval for a given query
2. and some machine learning module employs that information in order to learn an improved similarity measure – ideally one which matches perfectly the feedback the similarity teacher has specified.

Concerning 2, we have worked with gradient-based feature weighting techniques [21] as well as with evolutionary algorithms [20]. Within this paper, we focus on neural methods to learn and represent the similarity measure.

2.2.2 Solution Similarity

During the retrieval phase of a CBR application, the similarity between a query q and several cases c must be determined in order to find the most similar cases. To this end, the similarity is calculated between q and the problem part c_p of cases $c = (c_p, c_s)$ using some function $Sim : \mathcal{M} \times \mathcal{M} \to [0, 1]$ where $\mathcal{M}$ denotes the set of all problem descriptions (in our case $\mathcal{M} \subset \mathbb{R}^m$). One established way for obtaining utility feedback in the form of the desired case retrieval order for some query is to employ a solution similarity measure [22]. The idea here is to define an additional similarity measures $sim_s : A_s \times A_s \to [0,1]$ for the cases' solution parts, i.e. for the solution attribute A_s. Then, the similarity assessments received from sim_s can be used to generate the training data for optimizing the similarity measure $sim_p := Sim$ for the cases' problem parts. If, for example, the similarity between the solutions of two cases is very different from the similarity between their problem parts, then this may indicate that sim_p is poorly defined.

The solution similarity measure may either be a rather simple, distance-based syntactical similarity measure or a more sophisticated one defined by an expert. The fundamental assumption, however, is that it is in general by far easier to settle on a similarity measure for the cases' solution part than to define an appropriate similarity measure for the problem part on the basis of which the retrieval will be carried out. Instead, the former one ought to be used to learn the latter one. Based on these assumptions, we can rewrite our definition of a training set (cf. Eq. 1) such that it contains pairs of cases as input values and the solution similarities of those case pairs as targets.

Definition 1 (Case-Based Training Pattern). *Given two cases c and d each of which consists of a problem and solution part ($c = (c_p, c_s)$ and $d = (d_p, d_s)$), we define a* case-based training pattern *as a triple $(c_p, d_p, sim_s(c_s, d_s))$ where sim_s is the solution similarity measure and $sim_s(c_s, d_s)$ the target value.*

Matching this definition with the notion of Eq. 1, where the training set is $\mathbb{P} = \{(x^p, t^p) | p = 1, \ldots, |\mathbb{P}|\}$ we can say that each x^p corresponds to a pair of case problem parts (c_p, d_p) and the target value t^p corresponds to a solution similarity value $sim_s(c_s, d_s)$. On top of this, we define the full training pattern set $\mathbb{P}_{CB}$ for a given case base CB as

$$\mathbb{P}_{CB} = \{((c_p, d_p), sim_s(c_s, d_s)) | \forall c, d \in CB, c \neq d\}. \tag{2}$$

If we assume case problem parts to be made up of m (numerically represented) attributes, then the space of the supervised learning problem is $2m$-dimensional.

2.2.3 Classification Tasks

A special case arises in classification domains. Here, the utility of a case c for a given query q is either zero or one, depending on the real class membership of q, i.e. on whether it matches c's class or not. The solution similarity is then

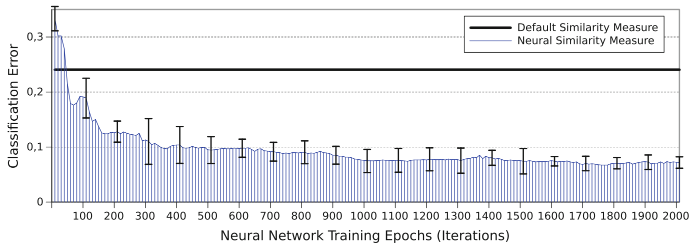
$$sim_s(c_s, d_s) = \begin{cases} 1 & if\ c_s = d_s \\ 0 & else. \end{cases} \tag{3}$$

Therefore, the resulting utility feedback cannot be said to be very "substantial", since for any case $c \in CB$ two sets of training examples can be generated – one with maximal utility, the rest with zero. In previous work [7], we found that learning similarity measures given such “knowledge-poor” training information is only of limited success, when using evolutionary optimization techniques. In this paper, we demonstrate, however, that this kind of feedback is suitable when learning and representing similarity measures with artificial neural networks.

2.3 Exemplary Results

Throughout this paper, we employ the data set on the evaluation of car values, taken from the UCI Machine Learning Repository [14], as explanatory example. In Sect. 4 we will present results for a larger selection of application domains.

The car data set consists of 1728 cases describing six features of cars like their maintenance costs, number of doors, and others (see Fig. 1, right). The solution attribute denotes the index for one out of four classes denoting how good (or acceptable) the car is. We split the data set into a train and independent test set using 5-fold cross-validation. Each time we trained a multi-layer perceptron neural network on the basis of a training set which was made up of $|\mathbb{P}_{CB}| \approx$ 1.9 million training examples according to Eq. 2 where we employed a solution similarity as given by Eq. 3. The network was made up of an input layer wit 12 inputs, two hidden layers with 13 neurons each using sigmoidal activation functions and a single output neuron with linear activation. We trained the network for 2000 epochs of Rprop and tested its performance on the independent test data set). The results shown in Fig. 1 correspond to the progress of the average classification error that results from a nearest neighbor classification on the basis of some knowledge-poor default similarity measure Sim_{def} (its accurate definition will be given in Sect. 3.1) as well as the neural network-based similarity measure (Sim_{NN}). Using Sim_{def} an average error of $24.1 \pm 2.4\,\%$ is obtained, whereas Sim_{NN} yields an average error of $7.2\,\%$ with a standard deviation of $1.0\,\%$ within a 5-fold cross-validation.



Car Domain

Attr.	Name / Range of Values
A0	buying price {low,medium,high,veryhigh}
A1	maintenance costs {low,medium,high,veryhigh}
A2	number of doors {2,3,4,5+}
A3	persons to carry {2,4,more}
A4	trunk capacity {small,medium,big}
A5	safety {low,medium,high}

Fig. 1. Classification error of the neural similarity measure Sim_{NN} compared to the knowledge-poor default similarity measure for the car evaluation domain.

2.4 Related Work on and Discussion of Neural Similarities

Neural networks are an established tool within machine learning. In this section, we point to related work that is of highest relevance to the first part of this paper insofar as it focuses on the combination of neural net-based and case-based approaches and on the use of neural networks for assessing similarity.

In [6], Dieterle and Bergmann target the regression task of internet domain appraisal. They employ knowledge-intensive similarity measures in conjunction with a neural network for a form of feature weighting. Interestingly, their trained networks receive (feature-specific) local similarity values as inputs and produce an estimate of the target value (numeric solution attribute) as output. While their approach to obtaining target values from some kind of solution similarity measure is similar to what we described in Sect. 2.2, a core difference is that in our approach the network represents a similarity measure in itself, i.e. a function that produces a similarity value from $[0, 1]$, whereas in [6] the network generates a prediction of the solution.

The approach described in Sect. 2.2 is highly related to the work of Maggini et al. on similarity neural networks [15]. These authors train a multi-layer perceptron that is used as a similarity measure for a k-nearest neighbor retrieval. Input to the net are the two full case representations of query and case, output is a single scalar similarity value from $[0, 1]$ which is similar to our approach. This work differs from ours in that they do not use the notion of utility feedback and solution similarity, but instead employ so-called pairwise constraints to generate the training data set. This is also related to the work of Hüllermeier et al. [13] who, however, do not focus on neural network-based architectures when learning or representing similarity measures. Additionally, Maggini et al. make use of a specialized, sophisticated network topology which, for example, also ensures the resulting similarity measure to be symmetric (a restriction we do not desire). By contrast, in this work we understand the utilization of neural networks as a useful and easy-to-use standard tool and therefore stick with established standards and defaults to the largest degree possible.

While the papers mentioned so far are of high relevance to our work, there exists also a number of further pieces of work on hybrid approaches using CBR methods and neural networks. These include applications for case adaptation [9], sequential case-based decision-making [25], as well as for case retrieval [16].

Discussion: Given the well-known generalization and approximation capabilities of neural networks, their usage for representing similarity measures in CBR appears highly attractive, but it misses two important facts. First, neural networks are black boxes and as those are completely untransparent to the user. While this issue may often seem acceptable to researchers, it quite as often represents a no-go for industrial applications where decision-makers and stakeholders critically scrutinize the decisions or recommendations of any AI-based system, specifically those involving subsymbolic approaches like neural networks. Second, the solutions and outputs of a neural network are not at all self-explaining. So, while a similar case or a set of cases represent actual episodical experience and, in doing so, generate trust by a potential (re-)user, neural networks completely fail to do so. These issues are, however, not new and have been acknowledged by other authors as well, e.g. [6] complain about the lack of transparency of neural similarity measures.

3 Similarity Measure Induction with Similarity Clouds

The discussion at the end of the preceding section represents the point of departure for the remainder of this paper. We proceed on the assumption that some neural similarity Sim_{NN} has been trained on the basis of a training data set $\mathbb{P}_{CB}$ with the method described in Sect. 2. While Sim_{NN} may yield excellent performance (in terms, for example, of classification or regression accuracy), it is a black box. To this end, it is our goal to extract the essence of the knowledge encoded in the trained network Sim_{NN} into a human-readable and understandable similarity measure Sim_{HR}. In so doing, of course, we want to lose as little of Sim_{NN}'s capabilities as possible.

3.1 The Local-Global Principle

In what follows, we are going to model similarity measures using the so-called *local-global principle* [4] which disassembles the overall similarity calculation into

1. local similarity measures $sim_i : D_{A_i} \times D_{A_i} \rightarrow [0,1]$ used to compute similarities between values of individual attributes A_i with domain D_{A_i},
2. feature weights w_i used to express the importance of individual attributes,
3. an amalgamation function used to combine local similarities and feature weights. Here, we stick to a weighted average calculation of similarity

$$Sim(q,c) = \frac{\sum_{i=1}^{n} w_i \cdot sim_i(q_i, c_i)}{\sum_{i=1}^{n} w_i} \tag{4}$$

With respect to local similarity measures, we focus on the following two commonly used representation formalisms for numeric and symbolic data types.

Definition 2 (Similarity Table and Similarity Function). *Let S be a symbolic attribute with a defined list of allowed values $D_S = \{v_1, \ldots, v_d\}$. A $d \times d$-matrix with entries $x_{i,j} \in [0,1]$ representing the similarity between the query value $q_S = v_i$ and the case value $c_S = v_j$ is called* similarity table *for D_S.*

Let N be a numeric attribute with a value range of $D_N = [D_N^{min}, D_N^{max}]$. A difference-based similarity function *$sim_N : D_N \times D_N \rightarrow [0,1]$ is defined as to compute a similarity value based on the difference between the case value $c_N = d_x$ and query value $q_N = d_y$ with $d_x, d_y \in D_N$, i.e. it calculates $sim_N(q_N, c_N) = f(q_N - c_N)$ for some function $f : [D_N^{min} - D_N^{max}, D_N^{max} - D_N^{min}] \rightarrow [0,1]$.*

So, a knowledge-poor default similarity measure Sim_{def} is defined to be made up of identical weights for all features ($w_i = 1$ for $1 \le i \le n$) as well as *default* local similarity measures. These are defined as

$$\begin{aligned} sim_{N,def} : D_N \times D_N &\rightarrow [0,1] \\ (q,c) &\mapsto f(q-c) \text{ with } f(x) = 1 - \tfrac{|x|}{D_N^{max} - D_N^{min}} \end{aligned}$$

for a numeric attribute N with domain range $[D_N^{min}, D_N^{max}]$, and as

$$sim_{S,def} : D_S \times D_S \rightarrow [0,1] \ with \ (q,c) \mapsto \begin{cases} 1 & if \ q = c \\ 0 & else \end{cases}$$

for a symbolic attribute S with domain D_S.

3.2 Similarity Clouds

The central concept on the basis of which our similarity induction procedure is based is the *similarity cloud*.

Definition 3 (Similarity Cloud). *Let a case c be described by m attributes A_i ($1 \leq i \leq m$) such that $c = (c_1, \dots, c_m)$. For a given case base CB and neural similarity measure Sim_{NN} a* similarity cloud *SC_i for attribute A_i is defined as*

$$SC_i : CB \times CB \times D_{A_i} \rightarrow [0,1] \ with \ (q,c,v) \mapsto sim_{NN}(q, c^v)$$

where case $c^v = (c_1^v, \dots, c_n^v)$ is defined such that $c_j^v = \begin{cases} v & if \ j = i \\ c_j & else \end{cases}$.

So, a similarity cloud is defined for a given set of cases and a specific attribute. It provides access to the neural net-based similarity for any combination of a query (from CB) and a full range of cases (also from CB) where the latter, however, have been "modified" such they take all possible values from domain D_{A_i}. The similarity cloud thus captures not just the similarities between any pair of cases from CB, but also the variations in similarity along the domain of attribute A_i (i.e. fluctuations, if that attribute value is altered).

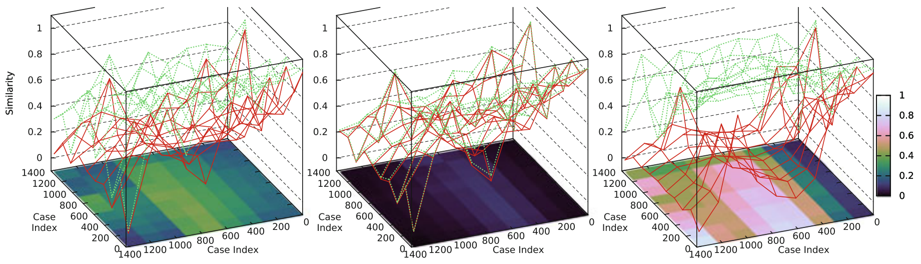


Fig. 2. Visualization of an excerpt of the similarity clouds for three of the attributes of the car domain (left A_0, middle A_2, and right A_5) which show average, little and high variability and yield corresponding feature weights. See the text for a full explanation.

In Fig. 2, we attempt to visualize parts of similarity clouds for the car evaluation domain introduced in Sect. 2.3. While the x and y axes correspond to queries and cases taken from the case base, respectively, we use the z values to plot the minimal (solid, μ_i) as well as maximal (dashed, ν_i) similarity values from the cloud for three exemplary attributes, where

$$\mu_i(q,c) = \min_{v \in D_{A_i}} SC_i(q,c,v) \textit{ and } \nu_i(q,c) = \max_{v \in D_{A_i}} SC_i(q,c,v).$$

The volume between μ_i and ν_i corresponds to the variability in neural similarity, when varying the value of attribute A_i. This volume is actually filled with a large number of similarity data points which, when watched in 3D, conveys the impression of a cloud (hence, the name "similarity cloud"). For clarity, however, the visualization shows only min and max values for each query-case combination.

3.3 Feature Weighting Using Similarity Clouds

Within this section, our goal is to derive feature weights w_i for Sim_{HR}. We employ a fixed amalgamation scheme (according to Eq. 4) as well as fixed local *default* similarity measures as specified in Sect. 3.1. Thus, the method introduced subsequently relies solely on the modification of the feature weights.

The variability of the similarity within the similarity cloud mentioned and visualized above represents changes in the *overall* (global) similarity between queries and cases. Therefore, it is an indicator of the respective feature's importance for the overall similarity assessment. Based on this observation we derive feature weights from the variability in the similarity cloud as follows.

Definition 4 (Similarity Cloud-Based Feature Weight). *Let SC_i be the similarity cloud for case base CB and attribute A_i. If A_i is a discrete attribute, we let $D := D_{A_i}$, otherwise (i.e. if A_i is numeric) we discretize D_{A_i} equidistantly according to $D := \{\frac{j}{S}(D_{max} - D_{min}) + D_{min} | j = 0, \dots, S\}$. The* similarity cloud-based feature weight *w_i is then defined as*

$$w_i = \frac{1}{|CB|^2} \sum_{c \in CB} \sum_{q \in CB} \sqrt{\sum_{v \in D} \left(\left(\sum_{u \in D} \frac{SC_i(q,c,u)}{|D|} \right) - SC_i(q,c,v) \right)^2}$$

The induced feature weight of an attribute essentially corresponds to the average standard deviation of neural similarity when altering the value of the attribute considered. In Fig. 2, samples of the standard deviation of neural similarity are visualized at the bottom of the plots using colored map views. As can be seen, for attribute A_2 there is very little variability (almost all samples are black) corresponding to a low feature weight, whereas for A_5 there are much higher variations which correspond to a significantly higher value of w_5.

3.4 Induction of Local Similarity Measures Using Similarity Clouds

In the following, we aim at the induction of local similarity measures from similarity clouds. We cover both types of local measures specified in Definition 2.

Symbolic Attributes: For a symbolic attribute with $D_{A_i} = \{v_1, \dots, v_s\}$ we fill the similarity table sim_i which contains an entry for each combination (v_q, v_c) of the query's and case's ith attribute value. For a specific combination of $v, w \in D_{A_i}$ we assume that $sim_i(v, w)$ is low, if the neural similarity $Sim_{NN}(q, c)$ for some query q and case c with $c_i = v$ changes "a lot", if we modify c_i's value and set it from v to w. Clearly, the notion "a lot" needs to be formalized. To this end, we utilize the concept of a similarity cloud, which already captures this information, and define the following scoring function.

Definition 5 (Cloud-Based Local Distance Scoring). *Given a symbolic attribute A_i and two values $v, w \in D_{A_i}$ as well as the attribute's similarity cloud SC_i, the* cloud-based local distance scoring *δ is defined as*

$$\delta(v, w) = \sum_{q \in CB} \sum_{c \in CB} \left(SC_i(q, c, v) - SC_i(q, c, w)\right)^2.$$

For given $v, w \in D_{A_i}$, the scoring function from Definition 5 considers all pairs of cases from CB. In so doing, the value of attribute i is, however, altered to be v and w, respectively, and the squared differences in the corresponding neural network-based similarities are summed up.

While the scores we obtain from Definition 5 represent a local distance function and may be utilized for determining a query's nearest neighbor, our emphasis is on inducing an easily human-interpretable similarity measure representation from the similarity cloud. Therefore, we apply the following normalization and transformation from a distance to a compatible similarity measure.

Definition 6 (Cloud-Based Local Similarity Table). *Given a cloud-based local distance scoring function $\delta : D_{A_i} \times D_{A_i} \to \mathbb{R}$ we induce a* local similarity table *for attribute A_i according to*

$$sim_i : D_{A_i} \times D_{A_i} \to [0, 1] \textit{ with } (v, w) \mapsto 1.0 - \frac{\delta(v, w) - \delta_{min}(v)}{\delta_{max}(v) - \delta_{min}(v)}$$

where $\delta_{min}(v) = \min_{w \in D_{A_i}} \delta(v, w)$ and $\delta_{max}(v) = \max_{w \in D_{A_i}} \delta(v, w)$.

When normalizing the induced local similarity measure according to Definition 6 we gain better interpretability, but lose information about the relevance of individual attributes. So, the induction of local similarity measures should be used in conjunction with the feature weighting method presented in Sect. 3.3.

Numeric Attributes: For attributes A_i with a numeric domain $D_{A_i} = [D_{A_i}^{min}, D_{A_i}^{max}]$ and, thus, for the induction of a difference-based similarity function we proceed in a similar manner. Essentially, we iterate over all query-case combinations from the case base, i.e. over all $q, c \in CB$, and measure the variability of the neural similarity, if we alter the value of the ith attribute in q and c, respectively, such that the difference $q_i - c_i$ takes some specific difference value s.

Of course, handling that infinite number of possible real-valued differences is numerically infeasible, which is why our goal is to come up with a similarity function that is represented by a set $\mathcal{S}$ of sampling points (i.e. a finite set of possible real-valued differences) that are distributed equidistantly over $[D_{A_i}^{min} - D_{A_i}^{max}, D_{A_i}^{max} - D_{A_i}^{min}]$. This approach is not new and has been successfully applied in [20]. When calculating the local similarity with such a sampled similarity function, the similarity is interpolated linearly between the two neighboring sampling points of $q_i - c_i$. For the purpose of representing sim_i it is therefore sufficient to provide a mapping from $\mathcal{S}$ to $[0,1]$, i.e. it is sufficient to specify sim_i as $sim_i : \mathcal{S} \rightarrow [0,1]$. Based on this observation we define:

Definition 7 (Cloud-Based Sampled Distance Scoring). *Let A_i denote a numeric attribute, SC_i the attribute's similarity cloud, and $\mathcal{S}$ denote the set of sampling points (with $|\mathcal{S}|$ assumed to be odd). Then,*

$$D'_{A_i} := \left\{ D_{A_i}^{min} + 2k \frac{D_{A_i}^{max} - D_{A_i}^{min}}{|\mathcal{S}| - 1} \mid 0 \leq k \leq \frac{|\mathcal{S}| - 1}{2} \right\}$$

represents a discretization of D_{A_i} on the basis of which we compute the cloud-based sampled distance scoring ε *according to*

$$\varepsilon : \mathcal{S} \rightarrow \mathbb{R} \; with \; s \mapsto \sum_{v \in D'_{A_i}} \sum_{w \in D'_{A_i}} \begin{cases} \delta(v,w) & if \; v - w = s \\ 0 & else \end{cases}$$

where $\delta(v,w)$ is calculated according to Definition 5.

Similarly to the case of symbolic attributes we induce the sampled similarity function by applying a normalization and transformation from a distance to a compatible similarity measure.

Definition 8 (Cloud-Based Sampled Similarity Function). *Given a cloud-based sampled distance scoring function $\varepsilon : \mathcal{S} \rightarrow \mathbb{R}$ we induce a local sampled similarity function for an attribute A_i with numeric domain according to*

$$sim_i : \mathcal{S} \rightarrow [0,1] \; with \; s \mapsto 1.0 - \frac{\varepsilon(s) - \min_{x \in \mathcal{S}} \varepsilon(x)}{\max_{x \in \mathcal{S}} \varepsilon(x) - \min_{x \in \mathcal{S}} \varepsilon(x)}.$$

3.5 Exemplary Results

We return to our explanatory example introduced in Sect. 2.3 and apply the similarity measure induction procedure presented above to induce both, feature weights and local similarity measures for the car evaluation domain. The left chart of Fig. 3 compares the classification errors of Sim_{def}, Sim_{NN} as well as induced similarity measures Sim_{HR} (with only weights and only local measures induced from Sim_{NN} as well as the combination of both). The numbers indicate the general effectiveness of the approach, since the high accuracy of the neural similarity can be preserved when inducing a similarity measure that is represented according to the local-global principle.

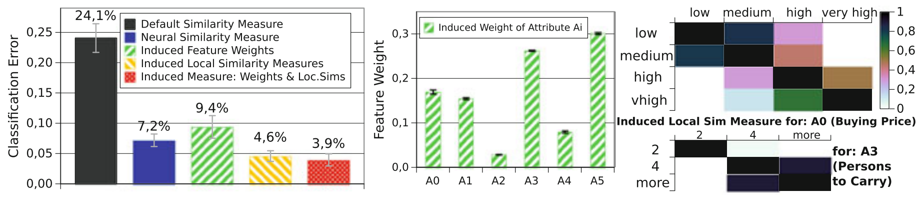


Fig. 3. Results of the similarity measure induction procedure applied to the car domain.

Another interesting question is whether the resulting measure Sim_{HR} is easy to read and understand. To this end, the central chart shows the normalized feature weights induced for the car evaluation domain. Please note how the weight of w_0, w_2 and w_5 (buying price, number of doors, car safety) correspond to the visualization of the attribute's corresponding similarity clouds in Fig. 2. Additionally, the right part of Fig. 3 visualizes the local similarity tables for attributes A_0 and A_3 (buying price, persons to carry).

3.6 Related Work on Feature Weighting and Similarity Learning

Feature weighting and similarity measure learning have a long history in case-based reasoning. Wettschereck and Aha [24] provide an overview on various early approaches. Recent approaches that go beyond pure feature weighting and instead aim at learning full similarity measure representations include neighborhood component analysis [8], large margin nearest neighbor classifiers [23] and various similarity-based classifiers reviewed by Chen et al. [5]. Another line of research on similarity learning focuses on exploiting binary pairwise information about similarity or dissimilarity, so-called pairwise constraints or preferences. This stream of research includes work on kernel-based learning methods [2], the boosting variant DistBoost [10], solution similarity learning in preference-based CBR [1], relevant component analysis [3] as well as similarity learning from case-order feedback [21]. A contrasting feature of the work presented in this paper compared to the pieces of related work mentioned here as well as in Sect. 2.4 is that we aim at learning a human-understandable and easily interpretable similarity measure which may increase trust by the users of the system. To this end, our work bears also some relatedness to the research field on explanations and case-based reasoning [18].

4 Experimental Results

In order to empirically evaluate the approach proposed in this paper we selected a batch of data sets from the UCI Machine Learning Repository. These included both, 14 classification and 5 regression tasks. For each domain, we split the data randomly into a training set S_{train} and a test set S_{test} within a 5-fold cross validation. In each learning run we successively applied the following steps:

1. Create a case base CB from S_{train}.
2. Evaluate the performance of the default similarity measure Sim_{def} (cf. Sect. 3.1) for all queries from S_{test}.
3. Train a neural network-based similarity measure Sim_{NN} as described in Sect. 2 and evaluate it on all queries from S_{test}.
4. Induce feature weights w_i from Sim_{NN} according to the procedure presented in Sect. 3.3, create a similarity measure $Sim_{HR,w}$ which employs the induced weights in combination with default local similarity measures.
5. Induce local similarity measures from Sim_{NN} according to the techniques described in Sect. 3.4, create a similarity measure $Sim_{HR,l}$ which uses those induced local measures in combination with default weights ($w_i = 1 \forall i$).
6. Create a similarity measure Sim_{HR} which uses both, induced feature weights as well as local measures $(4+5)$ and evaluate it on all queries from S_{test}.

Speaking about the evaluation of similarity measures, we perform k-nearest neighbor classification/regression and report the average classification or regression accuracy on S_{test} and the corresponding standard deviation. Throughout all experiments we set $k = 1$ for better comparability. This value might be optimized for each application domain and thus yield superior overall results.

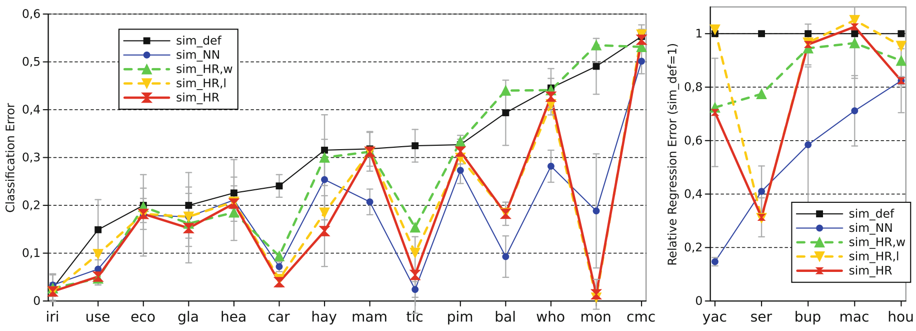


Fig. 4. Visualization of the performance of the considered similarity measures when used for classification/regression in various domains.

Results: As can be seen in Fig. 4, the quality of the learned neural similarity measure varies substantially over the domains considered. Averaging over all those domains the classification/regression error of the neural similarity measure is 39.2 % lower than the error made by default similarity measures (Table 1).

The neural measure, Sim_{NN}, as well as the knowledge-poor default similarity measure Sim_{def} represent, informally speaking, the upper and lower limit of what we can expect from the similarity induction technique proposed in this paper. On the one hand, we can observe that each induced measure is at least as performant as the default measure – which is a minimal goal from an evaluation perspective, because otherwise the entire approach would be rendered pointless. On the other hand, we can observe that in almost all cases the induced

Table 1. Overview of experimental results per domain. The task column distinguishes classification (C, with number of classes in brackets) from regression (R) tasks.

Domain	$\|CB\|$	#Attributes		Task	Classification/Regression error on S_{test} using				
		discr	num		Sim_{def}	Sim_{NN}	$Sim_{HR,w}$	$Sim_{HR,l}$	Sim_{HR}
balancescale	625	4	0	C(3)	.394 ± .068	.093 ± .043	.440 ± .022	.184 ± .024	.182 ± .024
car	1728	6	0	C(4)	.241 ± .024	.072 ± .010	094. ± .019	.046 ± .009	.039 ± .010
cmc	1473	9	0	C(3)	.553 ± .024	.501 ± .026	.531 ± .015	.557 ± .033	.546 ± .022
ecoli	336	0	7	C(8)	.200 ± .036	.179 ± .085	.197 ± .022	.182 ± .029	.182 ± .032
glass	214	0	9	C(2)	.200 ± .069	.162 ± .049	.176 ± .082	.152 ± .073	.176 ± .062
hayes-roth	132	4	0	C(3)	.315 ± .074	.254 ± .084	.300 ± .063	.185 ± .079	.146 ± .074
heart	270	8	5	C(2)	.226 ± .033	.211 ± .084	.185 ± .029	.207 ± .033	.204 ± .037
iris	150	0	4	C(3)	.027 ± .028	.033 ± .024	.027 ± .028	.020 ± .018	.020 ± .018
mammograph	830	4	0	C(2)	.318 ± .036	.207 ± .027	.312 ± .039	.312 ± .041	.312 ± .041
monks2	432	6	0	C(3)	.491 ± .058	.188 ± .119	.535 ± .039	.007 ± .016	.014 ± .031
pima	768	0	8	C(2)	.327 ± .020	.273 ± .027	.333 ± .024	.299 ± .033	.312 ± .026
tictactoe	958	9	0	C(2)	.325 ± .034	.024 ± .016	.154 ± .034	.101 ± .078	.054 ± .080
userknowl	258	0	5	C(4)	.149 ± .063	.067 ± .033	.047 ± .022	.098 ± .024	.051 ± .018
wholesale	440	0	6	C(3)	.445 ± .040	.282 ± .034	.441 ± .029	.411 ± .044	.427 ± .038
bupa	345	0	5	R	2.26 ± .27	1.32 ± .66	2.13 ± .26	2.18 ± .18	2.17 ± .17
housing	506	2	11	R	2.58 ± .11	2.13 ± .31	2.32 ± .16	2.47 ± .20	2.13 ± .04
machines	209	0	6	R	49.5 ± 10.1	35.2 ± 6.5	47.7 ± 10.6	51.9 ± 10.9	50.7 ± 9.5.
servo	167	4	0	R	.661 ± .139	.271 ± .063	.511 ± .199	.205 ± .046	.207 ± .048
yacht	308	0	6	R	3.78 ± .310	0.55 ± .059	2.73 ± .367	3.83 ± .421	2.66 ± .765
Average of Error Relative to Sim_{def}					100.00 %	60.84 %	82.44 %	69.98 %	63.86 %

human-readable similarity measure brings about almost as good results as the neural ones with only minor impairments, which are actually to be expected since Sim_{NN} is the input to the similarity clouds and, thus, to the entire induction procedure. Despite this, there are also some notable exceptions where the induced measures even outperform Sim_{NN}. Summarizing, from the 19 domains considered, there are 11 domains in which the resulting system's performance using Sim_{HR} is approximately equivalent or even superior to Sim_{NN}, 4 domains where the performance of the induced measures is somewhere in between Sim_{NN} and Sim_{def}, and finally 4 domains where the induced measures' performance is rather close the performance of the default measure. Averaging over all domains, the average classification/regression error of the induced measures, when compared to the error made by Sim_{def}, is reduced by 17.6 % if only weights are induced, by 30.0 % if only local measures are induced, and by 36.1 % if both approaches are combined, whereas Sim_{NN} lowered that error by 39.2 % as stated above. This, in conjunction with the fact, that the induced similarity measures are easily understandable for humans and domain experts, clearly stresses the usability of our approach.

5 Conclusion

In this paper, we have presented an approach that induces a similarity measure according to the local-global principle from a previously learned neural network-based representation of similarity. In so doing, we aimed at generating a human-readable representation of similarity which is likely to be accepted and trusted more by human users of the CBR system. We evaluated this approach on a large set of classification and regression domains from the UCI Repository and found that the induction procedure generates similarity measures that yield levels of accuracy that are only slightly inferior to the accuracy of the neural network-based templates.

References

1. Abdel-Aziz, A., Strickert, M., Hüllermeier, E.: Learning solution similarity in preference-based CBR. In: Lamontagne, L., Plaza, E. (eds.) ICCBR 2014. LNCS, vol. 8765, pp. 17–31. Springer, Heidelberg (2014)
2. Baghshah, M., Shouraki, S.: Semi-supervised metric learning using pairwise constraints. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI), San Francisco, USA, pp. 1217–1222 (2009)
3. Bar-Hillel, A., Hertz, T.: Shental, weinshall: learning a mahalanobis metric from equivalence constraints. J. Mach. Learn. Res. **6**, 937–965 (2005)
4. Bergmann, R., Richter, M., Schmitt, S., Stahl, A., Vollrath, I.: Utility-oriented matching: a new research direction for case-based reasoning. In: Proceedings of the 9th German Workshop on Case-Based Reasoning (GWCBR) (2001)
5. Chen, Y., Garcia, E., Gupta, M., Rahimi, A., Cazzanti, L.: Similarity-based classification: concepts & algorithms. J. Mach. Learn. Res. **10**, 747–776 (2009)
6. Dieterle, S., Bergmann, R.: A hybrid CBR-ANN approach to the appraisal of internet domain names. In: Lamontagne, L., Plaza, E. (eds.) ICCBR 2014. LNCS, vol. 8765, pp. 95–109. Springer, Heidelberg (2014)
7. Gabel, T., Stahl, A.: Exploiting background knowledge when learning similarity measures. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 169–183. Springer, Heidelberg (2004)
8. Goldberger, J., Roweis, S., Hinton, G.: Salakhutdinov: Neighborhood Component Analysis. In: Neural Information Processing Systems 18 (NIPS), pp. 513–520 (2005)
9. Henriet, J., Leni, P.-E., Laurent, R., Roxin, A., Chebel-Morello, B., Salomon, M., Farah, J., Broggio, D., Franck, D., Makovicka, L.: Adapting numerical representations of lung contours using case-based reasoning and artificial neural networks. In: Agudo, B.D., Watson, I. (eds.) ICCBR 2012. LNCS, vol. 7466, pp. 137–151. Springer, Heidelberg (2012)
10. Hertz, T., Bar-Hillel, A., Weinshall, D.: Boosting margin-based distance functions for clustering. In: Proceedings of the International Conference on Machine Learning (ICML), New York, USA, pp. 393–400 (2004)
11. Hinton, G., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. Science **313**, 504–507 (2006)
12. Hornick, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. **2**, 359–366 (1989)

13. Hüllermeier, E., Cheng, W.: Preference-based CBR: general ideas and basic principles. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI), Beijing, China, pp. 3012–3016 (2013)
14. Lichman, M.: UCI Machine Learning Repository (2013). archive.ics.uci.edu/ml
15. Maggini, M., Melacci, S., Sarti, L.: Learning from pairwise constraints by similarity neural networks. Neural Netw. **26**, 141–158 (2012)
16. Main, J., Dillon, T.S.: A hybrid case-based reasoner for footwear design. In: Althoff, K.-D., Bergmann, R., Branting, L.K. (eds.) ICCBR 1999. LNCS (LNAI), vol. 1650, pp. 497–509. Springer, Heidelberg (1999)
17. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In: Proceedings of the IEEE International Conference on Neural Networks (ICNN), San Francisco, USA, pp. 586–591 (1993)
18. Roth-Berghofer, T.R.: Explanations and case-based reasoning: foundational issues. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 389–403. Springer, Heidelberg (2004)
19. Rumelhart, D., Hinton, G.: Learning representations by back-propagating errors. Nature **323**, 533–536 (1986)
20. Stahl, A., Gabel, T.: Using evolution programs to learn local similarity measures. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS, vol. 2689, pp. 537–551. Springer, Heidelberg (2003)
21. Stahl, A., Gabel, T.: Optimizing similarity assessment in case-based reasoning. In: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006). AAAI Press, Boston (2006)
22. Stahl, A., Schmitt, S.: Optimizing retrieval in CBR by introducing solution similarity. In: Proceedings of the International Conference on Artificial Intelligence (IC-AI 2002). CSREA Press, Las Vegas (2002)
23. Weinberger, K., Saul, L.: Distance metric learning for large margin nearest neighbor classification. J. Mach. Learn. Res. **10**, 207–244 (2009)
24. Wettschereck, D., Aha, D.: Weighting features. In: Proceedings of the 1st International on Case-Based Reasoning (ICCBR), London, UK, pp. 347–358 (1995)
25. Zehraoui, F., Kanawati, R., Salotti, S.: CASEP2: hybrid case-based reasoning system for sequence processing. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 449–463. Springer, Heidelberg (2004)