Learning Similarity with Fuzzy Functions of Adaptable Complexity

Giorgos Mountrakis   Peggy Agouris

Dept. of Spatial Information Science and Engineering,
National Center for Geographic Information & Analysis,
University of Maine
348 Boardman Hall, Orono, ME, USA 04469-5711
{giorgos, peggy}@spatial.maine.edu

Abstract. A common approach in database queries involves the multidimensional representation of objects by a set of features. These features are compared to the query representation and then combined together to produce a total similarity metric. In this paper we introduce a novel technique for similarity learning within features (attributes) by manipulating fuzzy membership functions (FMFs) of different complexity. Our approach is based on a gradual complexity increase adaptable to problem requirements. The underlying idea is that less adaptable functions will act as approximations for more complex ones. We begin by interpolating a set of planes in the training dataset and due to linearity we get a fast first impression of the underlying complexity. We proceed to interpolate two asymmetrical sigmoidal functions whose initial approximations are calculated from the plane properties. If satisfactory accuracy is not achieved we provide advanced modeling capabilities by investigating FMFs parameters and convolving their output with additional functions.

1 Introduction

In recent years there is a significant increase in geospatial information availability. New sensor technologies together with enhanced data capturing techniques have created extensive data collections. Users that access these collections have diversified needs based on their past experience and/or task at hand. In such complex environments a communication process should be established with the ability to encapsulate user similarity. Similarity parameters should not be predetermined but rather adaptive to different scenarios and requirements even for the same datasets and/or users.

Relational operations like equality or inequality have been used in the past, but in complex database applications with quantitative features a similarity matching approach is applied. Such examples include similarity search within multimedia databases containing images [1], stock extraction based on temporal similarity [15] and retrieval from CAD databases [4]. In order to perform the similarity match efficiently an object O is represented by a set of features $O(F^1,F^2,\ldots,F^m)$ that describe its main characteristics. This way a multidimensional representation of the object's content is created. Each of these features may be represented by feature measures, so for feature *i* we might have *k* measures and represent it as $F^i = (f_{i1}, f_{i2}, \ldots, f_{ik})$. For example a GIS source might be described by two features, qualitative and quantitative attributes ($O_{GISsource}$ (Quantitative, Qualitative)). Each feature can have its own measures, for example Quantitative = (Timestamp, Scale,…).

A database object O is compared to a query description O^q by using matching functions to produce a similarity metric S as follows:

$$S = g\left[h_1\left(t_{11}(f_{11}, f_{11}^q), \ldots, t_{1k}(f_{1k}, f_{1k}^q) \right), \ldots, h_i\left(t_{i1}(f_{i1}, f_{i1}^q), \ldots, t_{ik}(f_{ik}, f_{ik}^q) \right) \right] \quad \mathbf{(1)}$$

In the above equation t_{ik} expresses the similarity between each feature measure, h_i combines similarity results from each feature measure to provide a metric for each feature F^i and g is the overall similarity measure. Value f^q reflects the query value for a feature measure presented to the system. The goal is to define functions t_{ik}, h_i and g so they express user perception of similarity.

Our focus in this paper is function t_{ik}. It is often described by a Euclidean distance (difference). Functions h_i and g receive more attention and are sometimes modeled through complex non-linear functions. However, if function t_{ik} fails to describe the corresponding similarity relationships adequately, its errors propagate in the overall solution and cannot be recovered. This may be the case if function t_{ik} attempts to model asymmetric, non-linear behavior that involves the direct comparison of feature measures. For example, let us consider a geospatial database and a user request for an aerial image of specific ground pixel size for building extraction. User interest decreases gradually (but not necessarily linearly) as pixel size increases to the degree that buildings would not be identifiable. Furthermore the user may have cost considerations (e.g. cost, storage and processing time) associated to a higher resolution acquisition. This translates to a similarity relation that can also be non-linear as resolution improves. So it is easily understood that we need asymmetrical, non-linear relations to model similarity *within each feature measure comparison* (function t_{ik}). Note that this does not necessarily hold true in the combined metric of the feature level (function h_i) or combination of features (function g).

In this paper we present a learning algorithm that attempts to express similarity within dimensions. We concentrate on quantitative attributes that are ordered (e.g. not postal code). We make use of fuzzy membership functions to capture similarity. Fuzzy methods have been used in the past for similarity assessment. For a general review see [14] and for an application in spatial configuration similarity queries [13]. Our contribution is of dual nature, namely:

i. Support of asymmetrical, non-linear similarity learning within dimensions while the complexity of similarity functions is dynamically adaptable to problem requirements. We do so by gradually increasing the complexity of the underlying function, evaluate results, and if further advanced modeling is required we use the less complex functions as approximations for the next more complex ones. Furthermore, similarity dependence on input parameters is identified, thus accelerating the process. Also, user confidence and prioritization of parts of the input and the output (similarity) space are incorporated by manipulating the weight matrix accordingly.

ii. Establishment of a theoretical framework for function combination through mathematical convolution. This is extremely useful in cases where either internal (by users) constraints are imposed (e.g. periodicity) or external (e.g. by system administrators/designers) restrictions are applied (e.g. real time information availability based on work load).

Our system's similarity results can be incorporated into algorithms that perform multi-dimensional similarity aggregation (e.g. quadratic function).

The rest of the paper is organized as follows. First we present background information on distance functions used previously. Section 3 introduces our approach and the two general similarity functions used, the piecewise linear and the sigmoidal one. The similarity dependence on the input values is also investigated. For cases where sufficient accuracy is not met we developed advanced fuzzy functions as described in section 4. Parameters that were previously constant are now adaptable through the training process. Also the theoretical framework for similarity function convolution is discussed. GIS information retrieval examples showing the functionality of our approach are presented in the next section. Finally we summarize our contributions in the last section (Section 6).

2 Related Work

Our algorithm falls in the general category of lazy learning methods otherwise known as nearest neighbor techniques [6]. Similarity learning is performed by storing examples as points in the feature space and using a distance metric to measure correlation to these points [2]. Usually a Minkowskian p-distance [3] is used to define the similarity measure and is defined as:

$$L_p(\bar{x},\bar{y}) = \left(\sum_{i=1}^{n} \left| x_i - y_i \right|^p \right)^{1/p} \qquad \textbf{(2)}$$

In the case that p=2 we have the traditional Euclidean distance metric. If p=1 then the Minkowskian distance expresses the Manhattan distance function. There is also the Quadratic distance that is a weighted form of the multi-dimensional Euclidean. Other functions used are the Mahalanobis [12], Camberra, Chebychen and others. A good overview and corresponding mathematical expressions can be found in [18].

The above functions provide a simple model that allows efficient indexing by dimensionality reduction techniques [7,19]. On the other hand though the simplicity of these functions does not support high adaptability to complex queries. Researchers have come to realize these limitations and have addressed them by a variety of non-linear, complex algorithms. Some of them involve for example neural networks with applications in content-based [9], information [10], and image [5] retrieval. Fuzzy sets have been used in visual retrieval systems [16] and fuzzy integrals approximate similarity in genetic algorithms [8]. Neuro-fuzzy approaches include supervised learning of agent profiles using if-then rules in remote sensing applications [11].

The operation of the above methodologies is concentrated on multi-dimensional similarity aggregation to provide an overall similarity metric. In some cases though the complexity of the problem relies on the similarity calculation within each dimension separately rather than on their combined aggregation. This is frequently the case when querying for GIS datasets. The data mining process might fail because the individual similarity metrics in every dimension were not able to capture user similarity preferences. So the need for more complex similarity functions within dimensions is evident, but what is really the computational cost associated with such an approach? Usually these database collections do not exhibit high dimensionality (e.g. 5-20 dimensions are typically used to index geospatial databases). This leads us to the conclusion that a more computationally intensive algorithm is feasible for improved accuracy. As we will show later, similarity is expressed through continuous functions of not very high complexity order leading to fast computations.

3 Fuzzy Similarity Functions

In this section we present the two function classes we use, the piecewise planar and sigmoidal ones. We discuss how the resulting planes of the piecewise solution can provide approximations for the non-linear sigmoidal functions. We also investigate similarity dependence on inputs by examining plane parameters.

3.1 Approach Overview

In order to adapt similarity to user preferences we developed a relevance feedback algorithm. Users are presented with a variety of pairs of requested and returned values and are asked to provide a similarity metric for each pair. They can also provide a similarity confidence level for the similarity response and prioritize parts of the input and output space. The corresponding training dataset is created and used as input for our similarity learning method.

For our training we make use of several similarity models as expressed through a variety of fuzzy membership functions (FMFs). Our approach is simple yet effective: gradually increase the complexity of the underlying function until an acceptable solution is reached. We begin the process by interpolating a set of planes to the training dataset (fig. 1). We examine the resulting accuracy and if it is within the predefined specifications we end the process. If not, we examine the obtained plane parameters. This analysis leads to a decision whether similarity is dependent on the query value, their difference metric or the actual database and query values. Building on that we interpolate two asymmetrical sigmoidal functions whose initial approximations are calculated from the plane properties. If required accuracy is not achieved we provide further modeling capabilities by parameterizing the FMFs parameters. At the last stage we obtain the best possible set of FMFs expressing user similarity as presented through the training set.

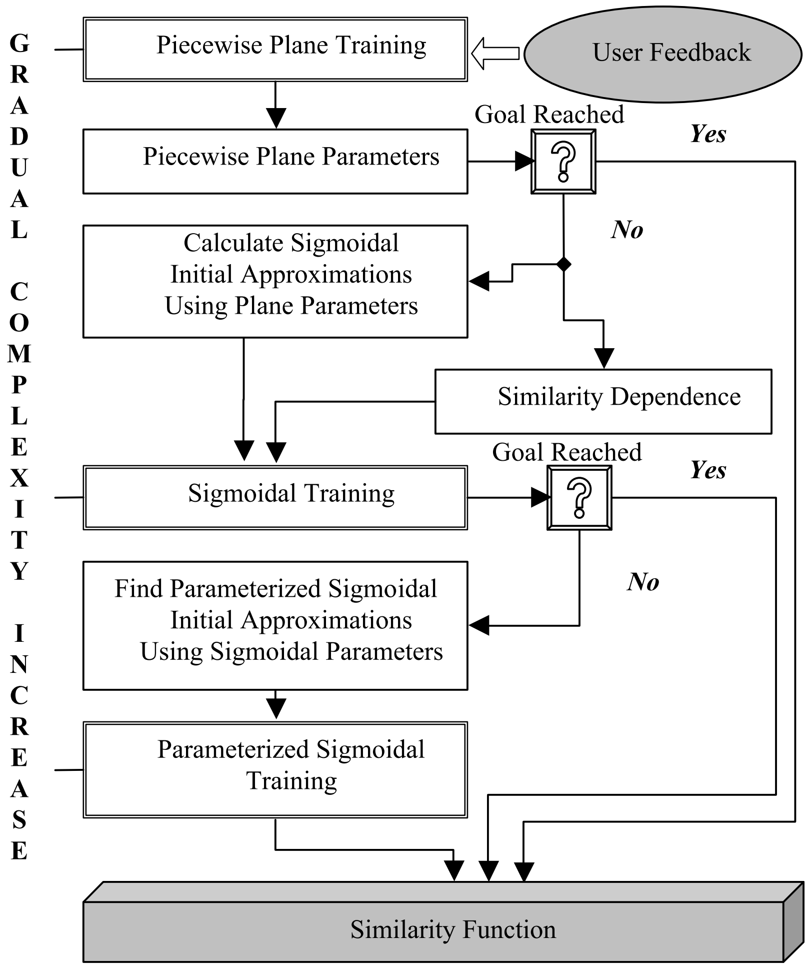
GRADUAL COMPLEXITY INCREASE
Piecewise Plane Training
User Feedback
Piecewise Plane Parameters
Goal Reached
Yes
No
Calculate Sigmoidal Initial Approximations Using Plane Parameters
Similarity Dependence
Sigmoidal Training
Goal Reached
Yes
No
Find Parameterized Sigmoidal Initial Approximations Using Sigmoidal Parameters
Parameterized Sigmoidal Training
Similarity Function

Fig. 1. Fuzzy functions training flow

3.2 Piecewise Planar Similarity Function

Mathematical formulation. Our simplest set of functions is composed by two piecewise planar solutions to support asymmetrical cases. Let function Sim$_{\text{Planar}}$ (•) represent an FMF mapping of the two-dimensional input space to the one-dimensional similarity space:

$$\mathbf{Sim}_{\mathbf{Planar}}\colon \Re^2 \rightarrow \Re^1 \tag{3}$$

The function inputs are query and database values [X_Q, X_{DB}] where $X_Q \in \mathbf{X_Q}$, $X_{DB} \in \mathbf{X_{DB}}$. Depending on which half plane the input parameters rely on ($X_Q > X_{DB}$ or $X_Q \leq X_{DB}$), two separate training datasets are created. Each half plane solution is independent from the other. The similarity function $\mathbf{Sim_{Planar}}$ (•) expressing the relation between a database value X_{DB} compared to a query value X_Q is:

$$Sim_{Planar}(X_Q, X_{DB}) = \begin{Bmatrix} a_1^R(i)X_Q + a_2^R(i)X_{DB} + a_3^R(i) & \text{if } X_Q \leq X_{DB} \\ \\ a_1^L(i)X_Q + a_2^L(i)X_{DB} + a_3^L(i) & \text{if } X_Q > X_{DB} \end{Bmatrix} \qquad \textbf{(4)}$$

Parameters a_1, a_2 and a_3 define the planes used for the corresponding half (left and right). Index i specifies the current plane under examination for each half. Our solution is composed by a collection of planes. Specifically, five planes are used to model similarity in each half plane (i.e. i = [1,2,3,4,5]). Each plane expresses similarity within certain range, for example plane #5 represents similarity for $X_Q > X_{DB}$ where $\mathbf{Sim_{Planar}} \in [1\text{-tb}, 1]$. An example of the plane configuration for the left half is shown in figure 2. Axes X and Y correspond to the inputs of our process, namely X_Q and X_{DB}. The Z axis represents the similarity output and is calculated based on the plane similarity function. A 2D section of the 3D function is presented in figure 3. This section shows the similarity function for a specific query value X_Q (the white line of figure 2). Such sections of the planes are used after the system is trained to calculate similarity of candidate database values to a specific user query.

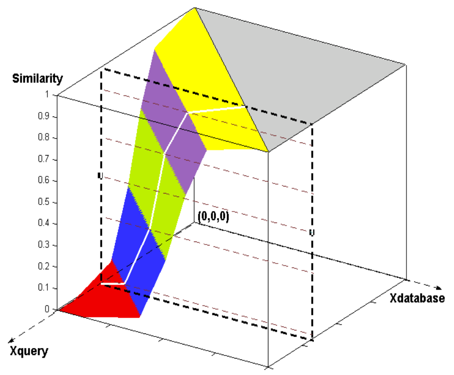

Fig. 2. Piecewise planar similarity function

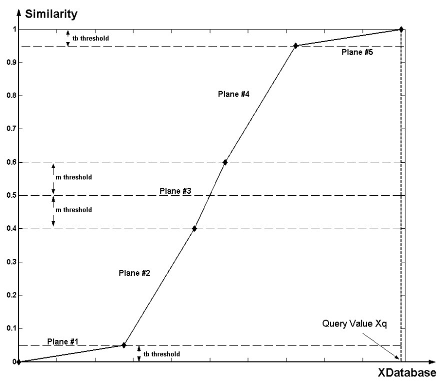

Fig. 3. 2D section for specific query request

Within our process we use exactly five planes in each half for two reasons:

i) *To address ranges where similarity function is almost parallel to the X_QX_{DB} plane.* The *tb* threshold expresses a similarity range of values close to 1 and 0 that will be mapped on planes #5 and #1 respectively. The use of *tb* allows the exclusion of non-active [X_Q, X_{DB}] pairs in terms of similarity gradient. Slight variations that might exist in similarity values close to 0 or 1 could lead the terms a_1 and a_2 to become very small with an unstable solution. In essence we use this threshold as a backup for cases where we might not obtain a solution so we assign a direct value. Furthermore, we want to be able to handle cases where expected similarity might follow a linear behavior but be active only in portion of the [X_Q, X_{DB}] space. The *tb* value defines the starting and ending point of planes #2 and #4 respectively (fig. 3).

ii) *To provide approximations for more complex functions that follow.* The *m* threshold defines the range above and below the 0.5 similarity value that is used to define the similarity modeling range of plane #3. The role of this threshold will be explained in the sigmoidal function family, where it is used as a parameter approximation. Planes #2 and #4 are used to model similarity in-between planes #1, #3 and #5, bringing the total number of planes to five.

Mathematical solution. The solution of this system can be found by using least squares. Our planes have some specific properties that we would like to propagate in the solution. These properties result from enforcing continuity between successive planes so there would be no similarity discontinuities. The continuity requirement provides the following constraints for each half of the solution:

i. The footprints of each plane on the $X_Q X_{DB}$ plane should be parallel to each other. In other words the slope should be the same, which is expressed as a constant ratio between parameters a_1(i), a_2(i).
ii. Successive planes should intersect at the specific similarity value as defined by thresholds *tb* and *m*. This is enforced mathematically by having the two successive planes and a third plane with [$\mathbf{Sim_{Planar}}$= constant] intersect at a common line.

In order to make our system efficient first we perform a fast linear interpolation for each plane separately. Then we add the constraints to the solution and after a few iterations the new planes with the imposed continuity are obtained.

Another interesting modification involves the *formulation of the weight matrix* **W** in the least squares solution. If we assume independency between the samples of the training dataset then all non-diagonal elements of **W** would be zero. Each diagonal element of **W** corresponds to a specific training sample that is presented as a [X_Q, X_{DB}, Sim] point. This element can express one or more of the following:

· $\mathbf{W_{confidence}}$: User confidence to the specified response [Sim] of the presented inputs [X_Q, X_{DB}]. For example, users might return a similarity value of 40% while being 80% sure for their response.

· $\mathbf{W_{input}}$: Users/database designers desire the capability to prioritize the training set based on how important a part of the input space is. In essence, based on the [X_Q, X_{DB}] value a metric is assigned showing the influence/significance of that section of the input space to the overall solution. For example, if users are requesting satellite imagery they might want the system to adapt more accurately to years close to 2000 than 1985 due to information availability.

· $\mathbf{W_{output}}$: Users/database designers might also want to guide the solution to be more accurate in specific parts of the output (similarity) space. This weight metric is solely dependent on the output value provided. For example a better fitting might be desired to higher similarity values (e.g. >70%) rather than lower ones (e.g. <30%).
The overall effect of the three cases is expressed in the calculation of matrix **W** as:

$$\mathbf{W} = \mathbf{W}_{\mathbf{confidence}} * \mathbf{W}_{\mathbf{input}} * \mathbf{W}_{\mathbf{output}} \tag{5}$$

If any of the three intermediate weight matrices is not a factor then it can be substituted by the identity matrix. If none of them is specified, **W** will be omitted from the least squares solution.

3.3 Similarity Dependence on Input Values

After the plane parameters are calculated we compute the average rotation angle (j_P) over the Z (similarity) axis. For each plane it is given by:

$$j_p = \arctan(\frac{-a_1(i)}{a_2(i)}) \tag{6}$$

Angle j_P should be the same for all planes in every half because we enforced the condition of having parallel footprints on the X_QX_{DB} plane. In figure 5 we show a piecewise planar similarity function. A contour plot representing similarity isolines is presented in figure 6. The calculating angle j_P is the angle between the footprints (or the isolines since they are parallel) and the $X_{database}$ axis.

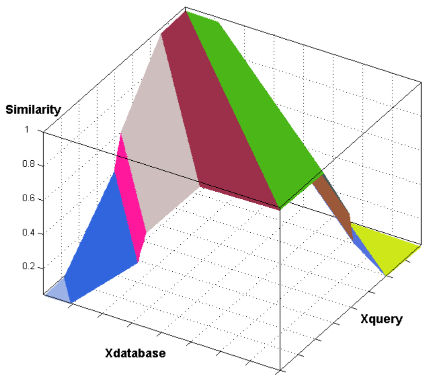

Fig. 5. Asymmetric planar similarity function

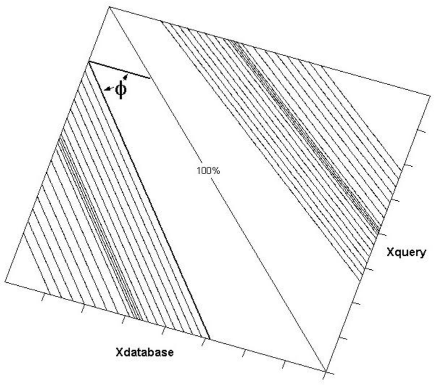

Fig. 6. Planar function contour plot

Our interest in this angle comes from the fact that based on its value similarity dependency on the input values can be extracted. Two special cases are identified and presented hereafter.

· Angle $j_P \approx 45^o$

In some cases the calculated angle might be close to the 45 degrees (fig. 7). This translates in the plane equation as $a_1(i) \approx -a_2(i)$. By substituting that to the plane equation we have for each half plane:

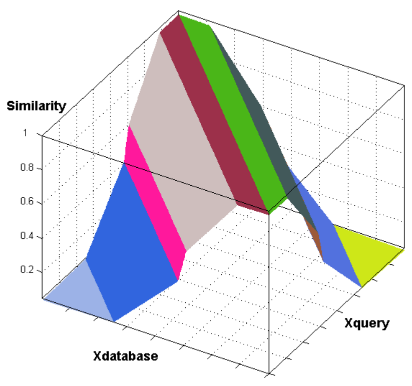

Fig. 7. Rotation angle $j_P \approx 45^o$

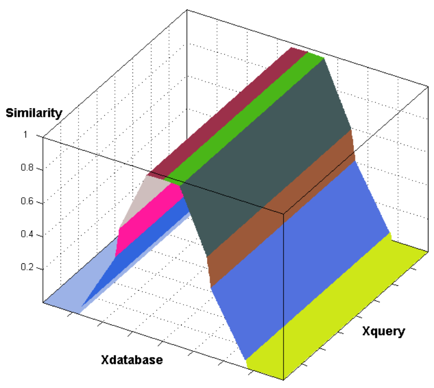

Fig. 8. Rotation angle $j_P \approx 90^o$

$$Sim_{Planar}(X_Q, X_{DB}) = \left\{ \begin{matrix} a_1(i)X_Q + a_2(i)X_{DB} + a_3(i) \\ \\ a_1(i) \approx -a_2(i) \end{matrix} \right\} \Rightarrow \quad \textbf{(7)}$$

$$Sim_{Planar}(X_Q, X_{DB}) = \left\{ \begin{matrix} a_1(i)X_Q - a_1(i)X_{DB} + a_3(i) \\ \\ Dist = (X_Q - X_{DB}) \end{matrix} \right\} \Rightarrow \quad \textbf{(8)}$$

$$Sim_{Planar}(X_Q, X_{DB}) = a_1(i) * Dist + a_3(i) \quad \textbf{(9)}$$

So we come to the conclusion that similarity is not dependent on the actual values of [X_Q, X_{DB}] but only on their Euclidean distance [Dist = X_Q - X_{DB}]. Equation 9 shows that the planes can be replaced by lines providing a significant computational gain since a three-dimensional problem is downsized to a two-dimensional one.

- Angle $j_P \approx 90^o$

For the case that angle j_P approaches the 90 degrees mark (fig. 8) the calculation formula of the angle provides $a_1(i) << a_2(i)$. With proper substitution in the plane equation we have:

$$Sim_{Planar}(X_Q, X_{DB}) = \left\{ \begin{matrix} a_1(i)X_Q + a_2(i)X_{DB} + a_3(i) \\ \\ a_1(i) << a_2(i) \end{matrix} \right\} \Rightarrow \quad \textbf{(10)}$$

$$Sim_{Planar}(X_Q, X_{DB}) \approx a_2(i)X_{DB} + a_3(i) \quad \textbf{(11)}$$

This translates into similarity dependence only on the X_{DB} value. So our system has the ability to recognize that user preference is not dependent on the actual request. But they still have a preference to the returned dataset which is expressed by equation 11.

An example of this nature might involve cases where different users access the same dataset and only their combined knowledge of the problem could express similarity in a comprehensive manner. Each user based on his/her expertise might provide part of the solution without necessarily being able to identify neither the overall similarity trend nor the "ideal" dataset that they might want. This might be the case in a remote-sensing application. Different experts might examine several images of different wavelengths. They would all be looking be a specific temporal instance of the phenomenon under investigation (e.g. iceberg separation). The problem would be that none of them knows the exact time and they all express their preference based on their expertise on the training datasets.

Our system design will overcome this problem based on a combined training dataset from a variety of users. If all users are looking for the same dataset the plane angle has a high possibility of being close to 90^{o}. In this case their similarity behavior should be expressed by a 3D surface similar to the one of figure 8. Since their preference revolves around a specific query value and only that, a 2D line can replace the 3D planes, which is consistent with the formulation of equation 11.

3.4 Sigmoidal Similarity Function

Mathematical Formulation. After the plane interpolation is performed, an accuracy assessment through a fitting error is done. If the results are not as desired a more complex function is necessary. To capture *non-linear* similarity relations between a query and a stored metric attribute we use a modified sigmoidal fuzzy relationship function. Sigmoidal functions are popular in the neural network community and have been used in the GIS field as predefined similarity functions for spatiotemporal trajectory matching [17]. Our similarity function is composed by two separate sigmoidal functions to compensate for asymmetrical cases. The similarity function Sim(•) for a database value X_{DB} compared to a query value X_Q is:

$$Sim_{Sigmoidal}(X_Q, X_{DB}) = \left\{ \begin{bmatrix} \dfrac{1}{1+e^{-a_R(X_R)}} \quad \mathit{if}\ X_Q \le X_{DB} \\ X_R = (X_Q - X_{DB} - c_R)\cos j_R + (X_Q + X_{DB} + c_R)\sin j_R \\ \\ \dfrac{1}{1+e^{-a_L(X_L)}} \quad \mathit{if}\ X_Q > X_{DB} \\ X_L = (X_Q - X_{DB} - c_L)\cos j_L + (X_Q + X_{DB} + c_L)\sin j_L \end{bmatrix} \right\} \quad \textbf{(12)}$$

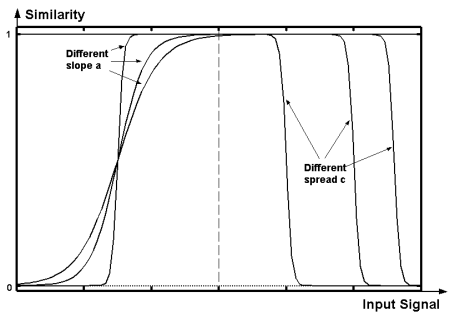

Fig. 9. Slope and spread influence on sigmoidal's shape

The parameters c_R and c_L specify the translation along the database axis. The slope of each sigmoidal function is expressed through a_R and a_L respectively. An important characteristic of the sigmoidal function is the large range of modeling capabilities. Efficient manipulation of the slope can result in representing a variety of cases, ranging from a linear up to a step-like behavior (fig. 9). This diversified capability together with the large operational range on the input space and the mathematical continuity of the function (first derivative exists everywhere) establishes the sigmoidal as appropriate solution from a variety of available fuzzy membership functions.

Initial approximations and parameter calculation. In a non-linear solution such as this there is always the problem of initial approximations. This is where the fast plane interpolation becomes multipurpose. We use the angle j as calculated before for the initial value of the rotation angle of the sigmoidal (for computational consistency $j = j_P - 45^o$). Also from the mathematical properties of our sigmoidal we know that spread c corresponds to the value where the sigmoidal similarity function will return 0.5 as output (fig. 10).

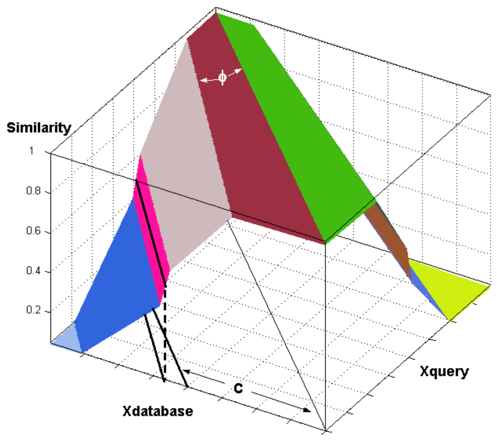

Fig. 10. Calculating sigmoidal initial parameters based on planar solution

That is the main reason we introduced plane #3 earlier and the threshold *m*. We want *m* to be as small as possible but at the same time include enough samples to have an accurate result. So using the properties of plane #3 we calculate:

$$c = \frac{-(0.5 - a_3)}{a_2} \tag{13}$$

The slope parameter cannot be calculated accurately directly from the planes. So in order to get an initial value we use the temporary values for j and *c*, and an equal (in number) random subset of the training data for each of the five planes. A least square solution gives an approximation for *a* with j and *c* being fixed.

After all three temporary values are calculated a final refinement takes place with the whole training set. In order to calculate the sigmoidal parameters a least squares solution is implemented through an iterative process. We use the A*δX=L formula where A is the matrix containing the partial derivatives with respect to the unknowns, δX contains the unknowns and L is the observation matrix. The solution is given by $dX = (A^T WA)^{-1} A^T WL$. W is the weight matrix as defined previously.

4 Advanced Fuzzy Similarity Functions

When the underlying complexity of the problem is high the already presented similarity functions might not be able to adequately model it. For these cases we present a more adaptable set of functions with higher modeling capabilities. We do so by introducing functions with higher input dependency. Later in this section we also present the theoretical framework for similarity function convolution. Even though this is not part of our training process its significant applicability is noticeable.

4.1 Parameter Substitution by Input-dependent Functions

In more complex behavior function parameters might not be independent of the query and/or database value. For example spread parameter *c* of a sigmoidal FMF may depend on the query value X_Q so *c* will not be constant throughout the input space (e.g. $c=c_o + tX_Q^2$, c_o and t are constants). Such a case might exist when users are more tolerant (in a non-linear fashion) towards query deviations as the query value increases. A detailed example is presented in section 5.1.

Mathematically this is expressed as follows: Let function **F(•)** represent an FMF mapping of the two-dimensional input space to the one-dimensional similarity space:

$$\mathbf{F}\colon \Re^2 \to \Re^1 \tag{14}$$

The function inputs are query and database values $[X_Q, X_{DB}]$ where $X_Q \in \mathbf{X_Q}$, $X_{DB} \in \mathbf{X_{DB}}$. Let **P** be the set of the *n* parameters that formulate this function:

$$\mathbf{P} = [p_1, p_2, \ldots, p_n] \qquad \mathbf{(15)}$$

In this case function **F(•)** can be expressed as:

$$\mathbf{F}\ (X_Q, X_{DB}|\ \mathbf{P}) \qquad \mathbf{(16)}$$

Now if we assume that each parameter p_i is not constant but is expressed by function $\mathbf{P}_i(\bullet)$ and is dependent on values $[X_Q, X_{DB}]$. Also function $\mathbf{P}_i(\bullet)$ with i = [1,2,…, n] has its own set of parameters K_i. This leads to the general expression of function **F(•)** which is :

$$\mathbf{F}\ (X_Q, X_{DB}|\ [\mathbf{P}_1 (X_Q, X_{DB}|\ K_1),\ \mathbf{P}_2(X_Q, X_{DB}|\ K_2), \ldots, \mathbf{P}_n (X_Q, X_{DB}|\ K_n)\ ]\ ) \qquad \mathbf{(17)}$$

For example let's examine the sigmoidal function of equation 12. Similarity function **F(•)** will be represented as:

$$\mathbf{F}\ (X_Q, X_{DB}|\ a, c,\ j\ ) = \frac{1}{1+e^{-a((X_Q-X_{DB}-c\ )\cos j\ +(X_Q+X_{DB}+c\ )\sin j\ )}} \qquad \mathbf{(18)}$$

The number of parameters is three, namely a, c, and j (n=3). Let $\mathbf{P} = [p_1, p_2, p_3]$ be the corresponding functions of these three parameters. For simplicity let's assume that $[p_1, p_3]$ are constants and only p_2 is substituted by function $\mathbf{P}_2\ (X_Q, X_{DB}|\ K_2)$. An example of such function could be:

$$P_2\ (X_Q, X_{DB}|\ c_o, c_1, c_2) = c_o + c_1 X_Q^2 + c_2 X_{DB} \qquad \mathbf{(19)}$$

In this case we would have $K_2 = [c_o, c_1, c_2]$. So instead of trying to solve for parameters [a, c, j] our new more complex system would have higher modeling capabilities and would be expressed by a new set of parameters [a, c_o, c_1, c_2, j]. The new set of parameters would be approximated initially by the solution obtained in the previous less complex solution, which in this example would happen if we set as initial approximations $[a^{new}, c_o^{new}, c_1^{new}, c_2^{new}, j^{\ new}] = [a^{old}, c^{old}, 0, 0, j^{\ old}]$.

4.2 Convoluting Function Output

We further enhance the operational range of our FMFs by introducing another important theorem. This time we do not alternate the properties of a function. Instead we combine more than one function to compose the underlying similarity signal. Such cases facilitate more complex user preferences that a single function could not express. An example would be periodicity combined with gradual decreasing interest (example 5.2). Combination of functions has another potential application that does not necessarily coincide with user perception of similarity. It rather expresses database system requirements and/or constrains that might exist. They can be static or adjust in real time depending on system sources. They can also vary depending on user position in the hierarchy (e.g. restricted access systems).

We allow the combination of functions by convolving their signals in the input space. Let function **F**(•) represent an FMF mapping and function **G**(•) be for instance an administrative constrain. We have:

$$\mathbf{F}\text{: } \Re^2 \rightarrow \Re^1, \ \mathbf{G}\text{: } \Re^2 \rightarrow \Re^1 \qquad \textbf{(20)}$$

Inputs for these functions are query and database values [X_Q, X_{DB}] where $X_Q \in \mathbf{X_Q}$, $X_{DB} \in \mathbf{X_{DB}}$. We define the convolution of functions **F**(•), **G**(•) as their multiplication throughout the input space $\Re^2$. If function **H**(•) is the resulting new function it can be represented as:

$$\mathbf{H}\ (X_Q, X_{DB}) = \mathbf{F}\ (X_Q, X_{DB}) \ \mathbf{*}\ \mathbf{G}\ (X_Q, X_{DB}) \qquad \textbf{(21)}$$

This new mapping function would also project the two-dimensional input space into the one-dimensional similarity space:

$$\mathbf{H}\text{: } \Re^2 \rightarrow \Re^1 \qquad \textbf{(22)}$$

Here we should note that we do not support training of function **G**(•) and/or retrain function **F**(•). Such a task would be extremely difficult due to the higher amount of parameters and the correlations in-between them. At this point we present the theoretical framework behind it and we investigate possible applications of it. Such training is reserved for future work.

5 Functionality Examples of our Model

In the following sections we introduce a series of examples using our method. Increasingly challenging similarity tasks are presented showing our model adaptability. Our applications are inspired by common but complex user preferences within geospatial environments. We begin with similarity learning in the scale dimension, continue in the temporal one and conclude with a connection speed example.

5.1 Scale Similarity

Let's consider users who are querying a GIS database and specifically request imagery of a particular scale (pixel ground size) that will be used for airport traffic monitoring. Their interest decreases gradually (but not necessarily linearly) as scale decreases to the degree that planes would not be identifiable. This expectation holds true for $X_Q < X_{DB}$. Such a case is presented in the right half of figures 11 and 12.

After training, the resulting function in the right half is a sigmoidal one with a variable slope a. The variable slope is able to express user alterable tolerance based on

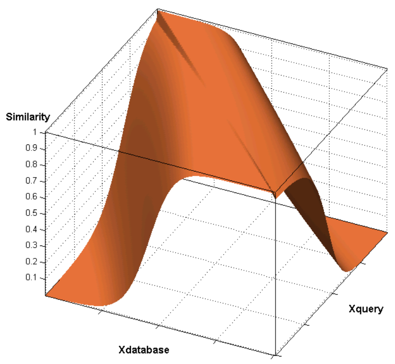

Fig. 11. Sigmoidal fuzzy similarity function

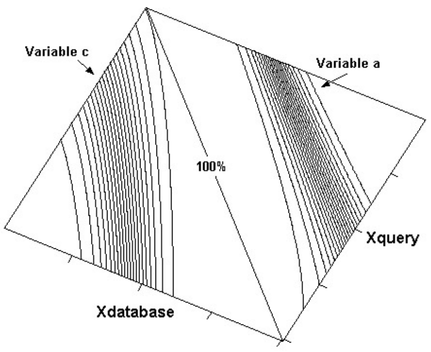

Fig. 12. Sigmoidal similarity isolines

their asking pixel size. The larger the requested pixel size the more flexible they are about additional pixel size. That does not happen in a linear fashion so a gradual decrease of *a* (slope) can model that.

Furthermore the user may have a cost function in mind associated with a better than requested resolution (e.g. price, storage, and processing time). This translates to a similarity relation that can also be non-linear as pixel size improves ($X_Q > X_{DB}$). The rate of similarity decrease does not change so slope remains the same (i.e. the isoline distance). But depending on the query pixel size the user expresses the associated cost by allowing a larger range of 100% similarity as query value increases. In other words, if they ask for 50m resolution they will consider that a 40m resolution does not add any additional cost so they assign similarity to be 100%. But if they ask for a much finer resolution this tolerance range will be much smaller due to for example higher price or manipulation cost.

5.2 Temporal Similarity

Another typical request involves the temporal footprint of a GIS source. One task might require the investigation of periodical phenomena. Such scenarios can incorporate dual preference. For example the main focus might be a specific year, but years close by would be acceptable too. This can be expressed by a sigmoidal function for each half (fig. 13 dotted line). Another preference could result from the specifics of the problem which might require information only during specific months (seasons). A sinusoidal function is used to model such preference (fig. 13 solid line).

Users would like though to combine both of these two requirements in the overall similarity computation. In order to do so we convolve the sigmoidals with the sinusoidal function which results in the similarity surface of figure 13. A specific example is shown in figure 14. A user wants to study a disease associated with the leaves of deciduous trees that appeared in 2000. The two asymmetrical sigmoidal functions express his/her preference to datasets before and after 2000, respectively. Note the different similarity gradient that shows datasets of earlier dates would be more suitable than datasets of later dates. Also this query requires datasets only through the summer

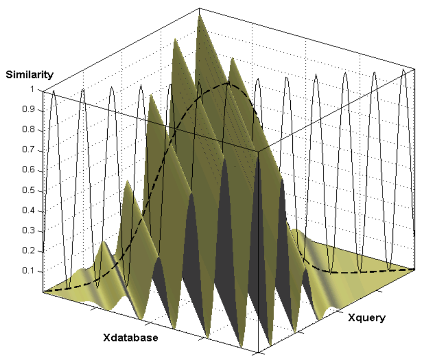

Fig. 13. Sigmoidal with sinusoidal similarity functions convolution

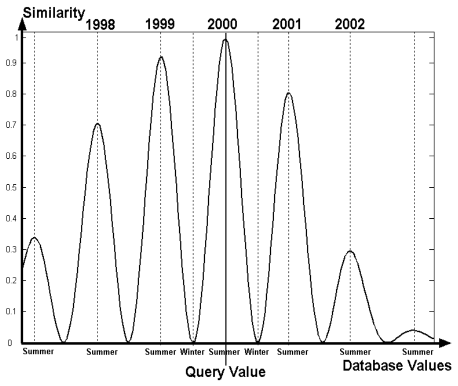

Fig. 14. Time periodicity example

months since in the winter time the trees lose their leaves. This is expressed through the sinusoidal function with a periodicity of a year. The combined result of the sigmoidal and sinusoidal functions shows user preferences for this task.

5.3 Server Connection Similarity

In this example we examine the application of constraints to user similarity functions. These constraints can be imposed by the database administrator or by system design, and can be fixed or adaptable to real-time monitoring of the database system.

To accommodate the vast volume of GIS datasets, data warehouses might be created leading to dataset availability through a variety of servers. Depending on their connection speed users might query for servers of analogous speed. Due to high demand all fast servers at some point might be overloaded. Then the system administrator might exclude these servers from the candidate ones being afraid that this might result into denial of service. So he/she creates a function such as the sigmoidal of figure 15 with solid black line. Then the user similarity preference (dotted line) will be convolved with the system constrain and would provide a new similarity surface, the one of figure 15. In figure 16 a contour plot shows the exact effect of the filtering function that was imposed. Servers with connection much faster than 10Mbit would not be accessed even though they would have been under normal circumstances.

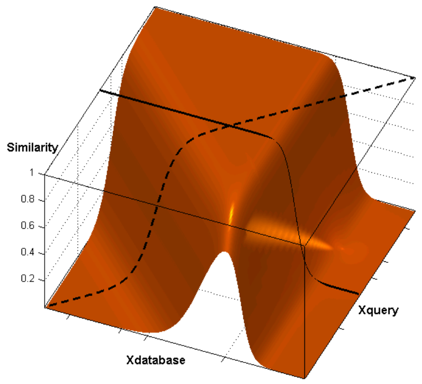

Fig. 15. Sigmoidal with sigmoidal similarity functions convolution

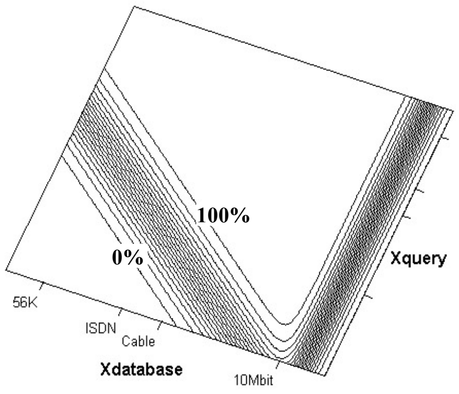

Fig. 16. Connection speed example with similarity isolines

6 Conclusion

In this paper we have proposed a novel method for similarity learning within dimensions. Our approach is based on a gradual complexity increase based on problem requirements. We use a variety of fuzzy membership functions to model the similarity signal. Back-propagation is used to train the fuzzy similarity functions. Initially, planes are interpolated and an analysis of the similarity dependence on the input space is performed. Based on the plane angle the system identifies cases where similarity is independent of the query value or dependent only on the distance metric between query and database values. This way the process is simplified and a significant computational gain is achieved. Also formulation of the weight matrix can enforce more accurate fitting in specific areas of the input space or specific similarity outputs, as well as support incorporation of user confidence in the provided response.

Our gradual complexity increase is expressed through different sets of functions. Their specific design allows the use of properties from less complex functions as approximations for the following more complex ones. By doing so a high convergence rate is achieved in the least squares solution. In the advanced functions we enhance complexity by adding non-constant behavior to the function parameters. We also presented a theoretical framework for similarity function combination through mathematical convolution. Incorporation of the convoluted functions in the gradual training flow is a future challenge. However, overall function adjustability as expressed through GIS query examples is indicative of superior modeling capabilities.

Acknowledgments

This work is supported by the National Science Foundation through grants DG-9983445 and ITR-0121269.

References

1. Agrawal, R., Faloutsos, C., Swami, A.: Efficient similarity search in sequence databases. Proc. of the Fourth Intl Conference on Foundations of Data Organization and Algorithms. (1993) 69-84
2. Aha, D.W., Kibler, D. F., Albert, M. K.: Instance-Based Learning Algorithms. Machine Learning. 6 (1991) 37-66
3. Batchelor, B.G.: Pattern Recognition: Ideas in Practice. New York: Plenum Press. (1978) 71- 72
4. Berchtold, S., Kriegel, H.-P.: S3: Similarity Search in CAD Database Systems. Proc. ACM SIGMOD Conf. (1997) 564-567
5. Carkacioglu, A., Fatos, Y.-V.: Learning Similarity Space. Proc. of Intl Conference in Image Processing. (2002) 405-408
6. Cover, T. M., Hart, P. E.: Nearest neighbor Pattern Classification. Institute of Electrical and Electronics Engineers Trans. on Information Theory. 13 (1) (1967) 21-27
7. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. Proc. of the 25th Intl. Conf. on Very Large Data Bases (VLDB). (1999) 518–529
8. Ishii, N., Wang, Y.: Learning Feature Weights for Similarity Measures using Genetic Algorithms. Proc. of IEEE Intl Joint Symp. on Intelligence and Systems. (1998) 27-33
9. Lim, J.-H., Wu, J.-K., Singh, S., Narasimhalu, A. D.: Learning Similarity Matching in Multimedia Content-Based Retrieval. IEEE Transactions on Knowledge and Data Engineering. 13 (5) (2001) 846-850
10. Mandl, T.: Tolerant Information Retrieval with Backpropagation Networks. Neural Computing & Applications. 9 (4) (2000) 280-289
11. Mitaim, S., Kosko, B.: Neural Fuzzy Agents that Learn a User's Preference Map. Proc. of 4th International Forum on Research and Technology Advances in Digital Libraries (1997) 25-35
12. Nadler, M., Smith, E. P.: Pattern Recognition Engineering. New York: Wiley (1993)
13. Papadias, D., Karacapilidis, N., Arkoumanis, N.: Processing Fuzzy Spatial Queries: A Configuration Similarity Approach. International Journal of Geographic Information Science (IJGIS). 13 (2) (1999) 93-128
14. Pappis, C.P., Karacapilidis, N.I.: A comparative assessment of measures of similarity of fuzzy values. Fuzzy Sets and Systems. 56 (21) (1993) 171-174
15. Rafiei, D., Mendelzon, O.: Similarity-Based Queries for Time Series Data. Proc. ACM SIGMOD Conf. (1997) 13-25
16. Santini, S., Jain, R.: Similarity Measures. IEEE Transactions on Pattern Analysis and Machine Intelligence. 21 (9) (1999) 871-883
17. Vlachos, M., Gunopulos, D., Kollios, G.: Robust Similarity Measures for Mobile Object Trajectories. Proc. of DEXA Workshops. (2002) 721-728
18. Wilson, D. R., Martinez, T. R.: An Integrated Instance-Based Learning Algorithm. Computational Intelligence. 16 (1) (2000) 1-28
19. Yi, B.-K., Faloutsos, C.: Fast Time Sequence Indexing for Arbitrary Lp Norms. Proc. of the 26th Intl. Conf. on Very Large Data Bases (VLDB). (2000) 385-394