


12-2004

Context-Specific Preference Learning of One Dimensional Quantitative Geospatial Attributes Using a Neuro-Fuzzy Approach

Georgios Mountrakis

Follow this and additional works at: <http://digitalcommons.library.umaine.edu/etd>

 Part of the [Databases and Information Systems Commons](#), and the [Geographic Information Sciences Commons](#)

Recommended Citation

Mountrakis, Georgios, "Context-Specific Preference Learning of One Dimensional Quantitative Geospatial Attributes Using a Neuro-Fuzzy Approach" (2004). *Electronic Theses and Dissertations*. 569.
<http://digitalcommons.library.umaine.edu/etd/569>

This Open-Access Dissertation is brought to you for free and open access by DigitalCommons@UMaine. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of DigitalCommons@UMaine.

**CONTEXT-SPECIFIC PREFERENCE LEARNING OF ONE-
DIMENSIONAL QUANTITATIVE GEOSPATIAL ATTRIBUTES
USING A NEURO-FUZZY APPROACH**

By

Georgios Mountrakis

Diploma in Engineering, National Technical University of Athens, Greece, 1998

M.S., University of Maine, 2000

A THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

(in Spatial Information Science and Engineering)

The University of Maine

December, 2004

Advisory Committee:

Peggy Agouris, Associate Professor of Spatial Information Science and Engineering,
Advisor

Anthony Stefanidis, Assistant Professor of Spatial Information Science and
Engineering

Mary Kate Beard-Tisdale, Professor of Spatial Information Science and Engineering

Mohamad Musavi, Professor of Electrical and Computer Engineering

Silvia Nittel, Assistant Professor of Spatial Information Science and Engineering

Michael Worboys, Professor of Spatial Information Science and Engineering

Copyright 2004: Georgios Mountrakis

All Rights Reserved

**CONTEXT-SPECIFIC PREFERENCE LEARNING OF ONE-
DIMENSIONAL QUANTITATIVE GEOSPATIAL ATTRIBUTES
USING A NEURO-FUZZY APPROACH**

By Georgios Mountrakis

Thesis Advisor: Dr. Peggy Agouris

An Abstract of the Thesis Presented
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
(in Spatial Information Science and Engineering)
December, 2004

With the recent explosion of information availability in geospatial datasets, query complexity has increased. Multiple users access the same data collections with highly diversified needs. Information retrieval goals can vary significantly due to the large number of potential scenarios/applications, a common problem in geospatial data collections. The current approaches are deterministic and do not allow the incorporation of user preferences in the query process. The approach developed in this thesis adjusts query returns using a preference-based similarity modeling and therefore expresses more accurately user anticipation of results.

In this thesis we present a machine learning approach to express user preferences within one-dimensional, quantitative attributes. Training is performed in multiple stages and is based on a training dataset provided by the user. Depending on the provided preference complexity our algorithm adjusts the learning process. Several families of functions are used progressively, from simple planar to complex sigmoidal functions. The design of the

algorithm allows previously interpolated functions to act as approximations for more complex ones that follow, thereby decreasing training time and increasing robustness.

A customized neural network, a Multi-Scale Radial Basis Function (MSRBF) network, is also developed specifically to express the characteristics of the problem. We model potential errors that result from the interpolation of the fuzzy functions; we do not want our neural network to expand to portions of the input space without significant evidence. Therefore, our network design forces the network to operate in a localized manner and only where necessary. At the last training stage fuzzy functions are combined with the MSRBF into one solution and if found appropriate, the fuzzy functions go through a self-organizing process, where they adjust further to the overwhelming preference.

The proposed neuro-fuzzy system outperforms the currently used distance-based nearest neighbor methods. It does so by design because it recognizes and supports distance dependent user preferences, while simultaneously offering advanced modeling capabilities. Our system also exhibits high robustness as statistical simulations demonstrate. This is partially due to the ability of the algorithm to adjust its complexity as the user preference complexity increases.

Acknowledgements

This work was partially supported by the National Science Foundation through grants DG-9983445 and ITR-0121269 for the National Center for Geographic Information and Analysis and a University of Maine Graduate Assistant Award.

For their support, collaboration, and assistance I would like to thank my advisor Peggy Agouris and my research co-advisor Anthony Stefanidis. They were there for me when I needed them the most inside and outside the academic environment. I would also like to thank the reviewing committee members:

- Mohamed Musavi for introducing me into the neural network world and being always accessible with valuable comments.
- Kate Beard for her supervision and helping me express my thoughts in paper.
- Silvia Nittel for her guidance and moral/scientific support during gray days.
- Michael Worboys for his promptness and putting the time into our discussions.

I am still grateful to my previous advisor, Giorgos Karras, for showing me the right way for my graduate studies, and Gillian Avruskin for her support, patience and encouragement. Also, I should not forget my friends at UMaine for making this experience unforgettable.

Finally, special thanks go to my family for their support and guidance. Simply put, without them this work would not have been possible.

Table of Contents

Acknowledgements	iii
List of Tables	xi
List of Figures	xii
Chapter 1. Introduction	1
1.1 Problem statement.....	2
1.1.1 In simple words.....	2
1.1.2 In mathematical terms.....	3
1.2 Motivation and applications.....	6
1.3 Scope of thesis.....	7
1.3.1 Focus within the general picture.....	8
1.3.2 Approach specifics.....	10
1.4 Research questions and objectives.....	15
1.5 Major results and contribution of thesis.....	17
1.6 Intended audience.....	18
1.7 Thesis organization.....	18
Chapter 2. Literature review	20
2.1 Data mining and knowledge discovery in databases.....	21
2.2 Data mining multi-disciplinary history.....	23

2.3 Data mining tasks and similarity learning.....	25
2.3.1 Predictive tasks.....	26
2.3.2 Descriptive tasks.....	27
2.3.3 Similarity learning within data mining.....	29
2.4 Geographic data mining.....	30
2.4.1 Special characteristics of geographic data mining.....	30
2.4.2 Tasks within geographic data mining.....	34
2.5 Machine learning and similarity learning.....	36
2.5.1 Positioning similarity learning in the general machine learning categories.....	36
2.5.2 Machine learning methods and their applicability for similarity learning..	38
2.5.2.1 Decision trees.....	38
2.5.2.2 Genetic algorithms.....	40
2.5.2.3 Case-based reasoning.....	41
2.5.2.4 Regression.....	42
2.6 Nearest neighbor in databases and machine learning.....	43
2.6.1 Nearest neighbor and K-nearest neighbor.....	44
2.6.2 Nearest neighbor similarity functions used in databases.....	46
2.6.3 Nearest neighbor in machine learning (instance-based systems).....	48
2.7 Preference learning approaches.....	52
2.8 Summary.....	56

Chapter 3. System design	57
3.1 Problem revisited.....	58
3.1.1 Nearest neighbor shortcomings.....	58
3.1.2 Issues addressed in this thesis.....	59
3.2 Machine learning using neural networks and fuzzy logic.....	61
3.2.1 Neural networks.....	62
3.2.2 Fuzzy methods.....	63
3.2.3 Neuro-fuzzy techniques.....	64
3.2.3.1 Neuro-fuzzy models.....	65
3.2.3.2 Mapping fuzzy rules to networks.....	67
3.2.3.3 Learning process.....	67
3.3 System design.....	69
3.3.1 Applicability assumptions.....	69
3.3.1.1 Class-based vs. instance-based similarity.....	69
3.3.1.2 Attributes under consideration.....	70
3.3.1.3 Database design independence.....	71
3.3.2 General framework.....	72
3.3.3 Architecture.....	73
3.4 Characteristics and uniqueness.....	74
3.5 Comparable work.....	76
3.5.1 Applications of neural networks and fuzzy logic in similarity learning....	77
3.5.2 Neuro-fuzzy architectures.....	78
3.5.3 Explicit related work.....	80

3.6 Summary.....	81
Chapter 4. Preference learning in one-dimensional attributes using fuzzy functions of adaptable complexity.....	82
4.1 Approach overview.....	83
4.1.1 Introduction.....	83
4.1.2 Process flow.....	85
4.2 Piecewise planar similarity function.....	87
4.2.1 Mathematical formulation.....	87
4.2.2 Mathematical solution.....	91
4.2.3 Weight manipulation to express training samples prioritization.....	91
4.3 Similarity dependence on input values.....	93
4.3.1 Distance dependent case.....	95
4.3.2 Database value dependent case.....	96
4.4 Sigmoidal similarity function.....	97
4.4.1 Mathematical formulation.....	97
4.4.2 Initial approximations and parameter calculation.....	99
4.5 Advanced fuzzy similarity functions.....	101
4.5.1 Parameter substitution by input-dependent functions.....	101
4.5.2 Convoluting function output.....	103
4.6 Functionality examples.....	104
4.6.1 Temporal similarity.....	104
4.6.2 Server connection similarity.....	106
4.7 Conclusion.....	108

Chapter 5. Preference refinement using a multi-scale radial basis neural network.....	109
5.1 Why neural networks?.....	110
5.2 Why neural networks with radial basis functions?.....	111
5.2.1 Multilayer perceptron networks.....	111
5.2.2 Radial basis function networks.....	112
5.2.3 Similarities and differences.....	113
5.2.4 Chosen network type.....	114
5.3 Radial basis network for the non-expert.....	117
5.4 Multi-scale RBF network.....	119
5.4.1 Selection of node centers.....	119
5.4.2 Selection of activation functions.....	122
5.4.2.1 Gaussian activation functions.....	122
5.4.2.2 Sigmoidal activation functions.....	124
5.4.3 Supporting a multi-scale analysis.....	126
5.4.3.1 Spread influence in RBFs.....	127
5.4.3.2 Spread assessment in our MSRBF.....	128
5.5 Customized network training.....	130
5.5.1 Traditional RBF training.....	130
5.5.2 The multi-scale overlapping problem.....	132

5.5.3 Incorporate local behavior into node evaluation.....	134
5.5.3.1 Customized node selection statistics.....	134
5.5.3.2 Blocking successfully mapped neighborhoods.....	140
5.5.3.3 Keeping blocked points as center candidates.....	142
5.5.3.4 Checking for local density.....	143
5.6 Mathematical formulation.....	146
5.7 Network pseudo code.....	150
5.8 Conclusion.....	152
Chapter 6. Combining fuzzy functions and neural network into a global solution.....	153
6.1 Self-organization of global similarity functions.....	154
6.2 Architecture.....	157
6.3 Mathematical solution.....	159
6.4 Conclusion.....	161
Chapter 7. Training and system evaluation.....	163
7.1 Training.....	164
7.1.1 Similarity input/output discussion.....	164
7.1.1.1 Class identification after simulation.....	165
7.1.1.2 Influence of total class number.....	168
7.1.2 Progressive training.....	169

7.2 Functionality examples.....	179
7.2.1 Walk-through training example for the resolution attribute.....	179
7.2.2 Additional example using the fuzzy functions.....	183
7.2.3 Neuro-fuzzy example.....	185
7.3 Statistical testing.....	187
7.3.1 Fuzzy functions assessment.....	187
7.3.1.1 Influence of initial approximations.....	187
7.3.1.2 Influence of noise and training sample size for convergence.....	191
7.3.1.3 Influence of noise and training sample size for convergence.....	193
7.3.1.4 Influence of noise and training sample size for iteration number.....	194
7.3.2 Neural network assessment.....	195
7.3.2.1 Simulation training dataset.....	195
7.3.2.2 Configuration of compared neural networks.....	197
7.3.2.3 Interpretation of simulation results.....	199
7.4 Conclusion.....	204
Chapter 8. Conclusions and future work.....	205
8.1 Thesis synopsis.....	206
8.2 Research contributions.....	207
8.3 Future work.....	209
8.3.1 Extensions of our learning system.....	209
8.3.2 Application of our learning methods to other domains.....	210
Bibliography.....	211
Appendix: Statistical simulations results.....	222
Biography of the Author.....	225

List of Tables

Table 2.1: Time line of data mining development.....	24
Table 2.2: Popular distance functions and their equations.....	47
Table 5.1: Example of a training set.....	131
Table 5.2: Node selection with MSE.....	132
Table 5.3: MSRBF inputs and parameters.....	150
Table 5.4: MSRBF Network training process.....	151
Table 5.5: Summary of the modifications on the traditional RBF.....	152
Table 7.1: Median and median values for MSE Training, MSE Testing and Nodes..	199
Table 7.2: Improvement percentage of the mean values.....	199
Table 7.3: Improvement percentage of the median values.....	199
Table 7.4: MSE differences between training and testing.....	200
Table A1: Detailed results of statistical simulations	222

List of Figures

Figure 1.1: Hierarchical object attribute representation.....	3
Figure 1.2: Query process of a geospatial source collection.....	8
Figure 2.1: Knowledge discovery process.....	22
Figure 2.2: Data mining primary goals and tasks.....	26
Figure 2.3: Types of inference: induction and deduction.....	36
Figure 2.4: Supervised learning.....	37
Figure 2.5: Unsupervised learning.....	38
Figure 3.1: Fuzzy inference system.....	64
Figure 3.2: Geospatial attributes organization.....	70
Figure 3.3: General similarity framework.....	72
Figure 3.4: Neuro-fuzzy training flow.....	73
Figure 3.5: System architecture.....	74
Figure 4.1: Fuzzy functions training flow.....	86
Figure 4.2: Piecewise planar similarity function.....	88
Figure 4.3: 2D section for specific query request.....	89
Figure 4.4: Partially active linear similarity function.....	90
Figure 4.5: Asymmetric planar similarity function.....	94
Figure 4.6: Planar function contour plot.....	94
Figure 4.7: Rotation angle $\varphi_p \approx 45^\circ$	95
Figure 4.8: Rotation angle $\varphi_p \approx 90^\circ$	96

Figure 4.9: Slope and spread influence on sigmoidal's shape.....	98
Figure 4.10: Calculating sigmoidal initial parameters based on planar solution.....	99
Figure 4.11: Sigmoidal with sinusoidal similarity functions convolution.....	105
Figure 4.12: Time periodicity example.....	106
Figure 4.13: Sigmoidal with sigmoidal similarity functions convolution.....	107
Figure 4.14: Connection speed example with similarity isolines.....	107
Figure 5.1: Multilayer perceptron network structure.....	111
Figure 5.2: Radial basis function network structure.....	113
Figure 5.3: Fuzzy membership function interpolation.....	115
Figure 5.4: Resulting error from fuzzy membership function.....	115
Figure 5.5: Fuzzy membership function with RBF.....	117
Figure 5.6: Simple RBF example – 1 input, 3 hidden nodes and 1 output.....	117
Figure 5.7: Inside look at RBF operation.....	118
Figure 5.8: Node center selection.....	121
Figure 5.9: Node center selection generalization example.....	121
Figure 5.10: Gaussian function.....	122
Figure 5.11: Different 3D Gaussian activation functions.....	124
Figure 5.12: Sigmoidal function.....	125
Figure 5.13: Different 3D Sigmoidal activation functions.....	126
Figure 5.14: Spread influence in a RBF network.....	127
Figure 5.15: Gaussian overlapping problem.....	133
Figure 5.16: Combination of global and local MSE.....	136
Figure 5.17: Maximum global MSE.....	137

Figure 5.18: Chosen activation function after first iteration selected a local-fitting one.....	141
Figure 5.19: Chosen activation function after blocking part of input space.....	141
Figure 5.20: Using an inverted symmetrical Sigmoidal as a blocking function.....	141
Figure 5.21: 3D blocking example.....	142
Figure 5.22: Degrading signal problem.....	143
Figure 5.23: Local distribution problem.....	144
Figure 5.24: Density zones at the output.....	145
Figure 5.25: Multi-scale RBF network architecture.....	146
Figure 6.1: Outlier effect on fuzzy membership function.....	154
Figure 6.2: Residuals of original and weighted-distance fuzzy functions.....	155
Figure 6.3: Weights manipulation.....	156
Figure 6.4: Neuro-fuzzy network architecture.....	158
Figure 7.1: System return values.....	166
Figure 7.2: Classification translation to quantitative values during training and vice-versa during simulation.....	167
Figure 7.3: Effect of number of classes on training set.....	168
Figure 7.4: Progressive training.....	169
Figure 7.5: Calculating dependency angle.....	172
Figure 7.7: Training points for planar solution.....	174
Figure 7.8: Calculation of plane parameters.....	175
Figure 7.9: Assignment of classes to intersecting lines.....	176
Figure 7.10: Resolution attribute profile training.....	180

Figure 7.11: Advanced sigmoidal fuzzy similarity function.....	182
Figure 7.12: Profile example.....	182
Figure 7.13: Example of a user preference function.....	183
Figure 7.14: Contour plot and query examples of this preference function.....	184
Figure 7.15: Similarity preference captures with our neuro-fuzzy system.....	186
Figure 7.16: Influence of shift's initial approximations to convergence.....	188
Figure 7.17: Influence of angle's initial approximations to convergence.....	189
Figure 7.18: Influence of slope's initial approximations to convergence.....	190
Figure 7.19: Influence of slope to sigmoidal's shape.....	190
Figure 7.20: The influence of noise in slope calculation.....	192
Figure 7.21: The influence of noise in shift calculation.....	192
Figure 7.22: The influence of noise in angle calculation.....	192
Figure 7.23: Influence of noise and training sample size for convergence rate.....	193
Figure 7.24: Influence of noise and training sample size for iteration number.....	194
Figure 7.25: Creation of the simulations training dataset.....	196
Figure 7.26: Training points of the simulations dataset.....	197
Figure 7.27: Convergence success rates for training and testing.....	201
Figure 7.28: Local gaussians effect in the testing MSE.....	203
Figure 7.29: Summarized local gaussians effect in the testing MSE.....	203

Chapter 1

Introduction

The development of novel sensors and innovative data acquisition methods, and advancements in computer storage and access capabilities are resulting in tremendous increases in the amount of geospatial datasets that are currently available to the corresponding user community. Parallel to these developments, the user community itself is undergoing an expansion and transformation, with growing numbers of users and applications that need access to geospatial information. This increased usage is affecting geospatial information retrieval (IR) as the same geospatial data collections may be accessed by users with diverse needs and interests. Advanced communication processes should be established to capture and express user preferences in such environments as basis for similarity models. This preference can result from a variety of scenarios/applications, a common problem in geospatial data collections. It is context-specific, therefore many users can exhibit comparable preference, and the same user can demonstrate various preferences based on task requirements. Consequently, the goal of this work is to develop a novel similarity model that is preference-based and therefore improving appropriateness accuracy of the retrieved geospatial information in the query process.

In this chapter we provide a description of the problem and the motivation behind it. We discuss how our algorithm fits in the overall query process framework. A general idea of the expected contributions follows along with the thesis organization.

1.1 Problem statement

1.1.1 In simple words

With the recent explosion of information availability, query complexity has increased. Multiple users access the same data collections with highly diverse needs. A deterministic approach cannot facilitate the varying scenarios and applications. Establishing and expressing user information requests requires formulation of advanced communication processes. A learning component should be added to capture user preference and incorporate it into a similarity model. This is the goal of the dissertation.

In some cases relational operations like equality or inequality can handle a query request, like for instance “return all aerial photographs taken after 1999”. In many advanced database applications though, users would expect a more detailed answer, one that will not just return the results but will rank them based on some similarity metrics. Such a request might be “find me the 10 *most appropriate* aerial photographs taken at 11/12/1999”. In our work we attempt to model these *most appropriate* criteria and express them through a mathematical model so the returned answers are adjusted to user preferences. A user preference example would be “I would prefer aerial photographs from 1999, but 1998 will be OK too, but I do not want anything after 2000”.

The above example is a fairly simple case. In more complex preference expressions, non-linearity and non-monotonic behaviors might exist. Non-linearity refers to the way user interest degrades as the candidates are further away from the query request. Non-monotonic decline means that if candidate A is further away from the query than candidate B that does not necessarily translate into candidate B being more suitable than A. Examples of such cases are described in the next section.

1.1.2 In mathematical terms

In the context of this work, a geospatial information object O is an autonomous entity with a specific database record. Examples may include a map, a DEM, a satellite image, or the record of a building in a cadastre database. Within a database, such objects are typically described by a set of attributes. For example, a satellite image may be described through its coverage, resolution, time, and type of sensor, among others. Certain attributes may be conceptually related and thus may form distinct conceptual groups, for example metric and qualitative attributes may be grouped separately. This hierarchical arrangement is visualized in figure 1.1.

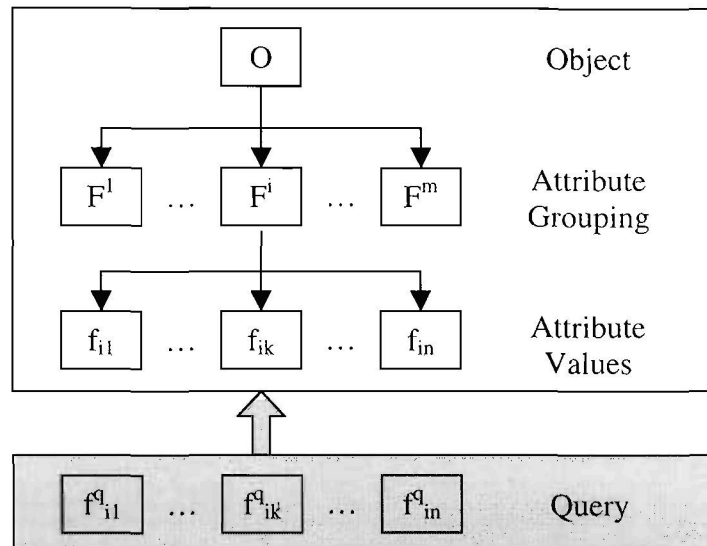


Figure 1.1: Hierarchical object attribute representation

The comparison of an information object O stored in a database to a query request O^q entails the comparison of their corresponding attribute values. This may be a straight forward issue if the attributes used to describe both O and O^q are the same, or may require advanced translation methods (e.g. using ontologies) to establish correspondences among two different sets of attributes. Assuming similar representations, the comparison

of a stored object to a query request involves the use of matching functions to produce a similarity metric S as:

$$S = g \left[h_1(t_{11}(f_{11}, f_{11}^q), t_{12}(f_{12}, f_{12}^q), \dots), \dots, h_i(t_{i1}(f_{i1}, f_{i1}^q), \dots, t_{ik}(f_{ik}, f_{ik}^q), \dots, t_{in}(f_{in}, f_{in}^q)), \dots \right] \quad (1.1)$$

In the above equation:

- Function t_{ik} expresses the similarity between the *database attribute value* f_{ik} and the corresponding *query value* request f_{ik}^q .
- Function h_i combines similarity results from each separate attribute to provide a similarity metric for each conceptual attribute grouping.
- Function g is the overall similarity measure combining similarity from each conceptual group to one total metric.

The overall goal of intelligent database queries is to define functions t_{ik} , h_i and g so they express user perceptions of similarity.

For example, let us consider a query of a geospatial database. Assume that global similarity is calculated based on three attribute groups, namely $F = [\text{Metric attributes, Qualitative attributes, Dataset accessibility}]$. The “Metric attributes” group may be represented by two attributes, namely *time* and *scale* ($F^1 = [\text{Time, Scale}]$). An example of a query request may be $F^{1q} = (f_{11}^q, f_{12}^q) = [10\text{am}, 50\text{cm}]$, aiming for the recovery of datasets depicting an area at 10am, with a scale (resolution) of 50cm. After this query is presented to the system, similarity within each attribute is calculated using functions t_{ik} that explain how similar stored values are to the query request. For example, function t_{11} would calculate the similarity in time between query request and a database candidate, and function t_{12} would calculate the similarity in the scale attribute.

At the next step, function h_i aggregates these similarity results from different

attributes to provide a similarity metric for the corresponding attribute group (e.g. producing an aggregate similarity value for “Metric attributes” group from the similarity results obtained from attributes Time and Scale). In the last step, function g integrates similarity values from different attribute groups to derive an overall similarity metric expressing similarity between query request and a database record. In many cases functions h_i and g are treated as one function depending on the attribute organization.

Now let us examine similarity preference users can exhibit in the above example $F^{1q} = [10\text{am}, 50\text{cm}]$, and assume a request for aerial photography for parking load estimation. We focus on the first step of the process, i.e. calculation of similarity preference within each attribute (time, scale) using functions t_{ik} (in this case functions t_{11} , t_{12}). User experience from the area suggests that photographs between 7-9am would not be appropriate due to early morning fog that diminishes the analysis potential of aerial photography. Also the user might not want photographs taken during lunch break as they may provide misleading information. In addition to that even if the initial target is 10 am, any other daylight photograph would be acceptable but as the sun goes down, visibility decreases rapidly and the temporal preferences of the user.

The scale of the imagery is also important. User interest may decrease gradually (but not necessarily linearly) as scale decreases to the degree that cars would not be identifiable. Furthermore, the user may have additional considerations (such as cost, storage and processing time) associated to a higher resolution acquisition. This translates to a preference expression that can also be non-linear as resolution improves.

The above example is typical in geospatial applications and offers evidence that currently used distance-based linear similarity functions do not describe adequately user

preferences. Asymmetrical, non-linear, non-monotonically decreasing functions that support user preference can model similarity more precisely in each attribute comparison (function t_{ik}). Currently, the focus of database queries has been on the development of complex non-linear models for functions h_i and g . In contrast, functions t_{ik} have received little attention and are typically modeled in relatively simple manner. It is easily understood though that if functions t_{ik} fail to describe adequately the corresponding similarity relationships, the resulting metrics of similarity will be significantly compromised. In geospatial queries user preferences may be much more complex than general queries (e.g. text queries), while the diversity of users and applications is further emphasizing the need for efficient modeling of t_{ik} functions. Thus, modeling user similarity preference within each attribute can substantially help geospatial queries. Motivated by these observations, the focus of our work is to investigate the application of complex functions for user preference within each attribute.

1.2 Motivation and applications

The major motivation behind this research is the lack of a query model for geospatial environments that has the ability to adjust results based on varying user preference. Nowadays, information volumes increase at high rates. This information makes users more demanding in their information requirements and information retrieval expectations in the query process.

Unfortunately, there is a disproportionate amount of research done on adaptable systems in information retrieval from geospatial databases than in other areas (e.g. text retrieval). Specifically, large distributed information source repositories are created and several issues related to storing and accessing these databases are investigated.

Ontologies are created to compensate for different field descriptions, as well as multi-node architectures and theoretical database models to support them. Query languages and indexing mechanisms for faster information retrieval are developed.

Surprisingly, similarity learning has not yet received significant attention. The similarity measures used to rank potential candidates are defined by the system designer and remain the same for all scenarios and applications. These models represent a deterministic approach, which might or might not facilitate user needs. We believe that the query process should be adaptable to user preferences in order to achieve high ranking accuracy. Consequently, this is the issue we address in this thesis.

Our method interacts with but is independent of the query process itself, and thus a variety of GIS applications can benefit from it. Repositories of GIS source information for environmental, remote sensing, transportation, multimedia and monitoring applications could experience a significant information retrieval accuracy improvement. The improvement would be especially evident in cases where the same source collection is accessed by highly diverse groups of users, where diversity translates to different similarity preferences. Even though our method is designed to facilitate geospatial needs, either parts or our whole methodology could be applied on other data collections with similar characteristics, depending on the problem at hand.

1.3 Scope of thesis

We have already presented the problem that we focus on in this thesis. Here we provide the general framework of the query process in order to make it easier for the reader to understand our contribution. We also explain in more detail what we do and do not address in our work.

1.3.1 Focus within the general picture

In a geospatial environment the following steps take place in a query process (fig 1.2):

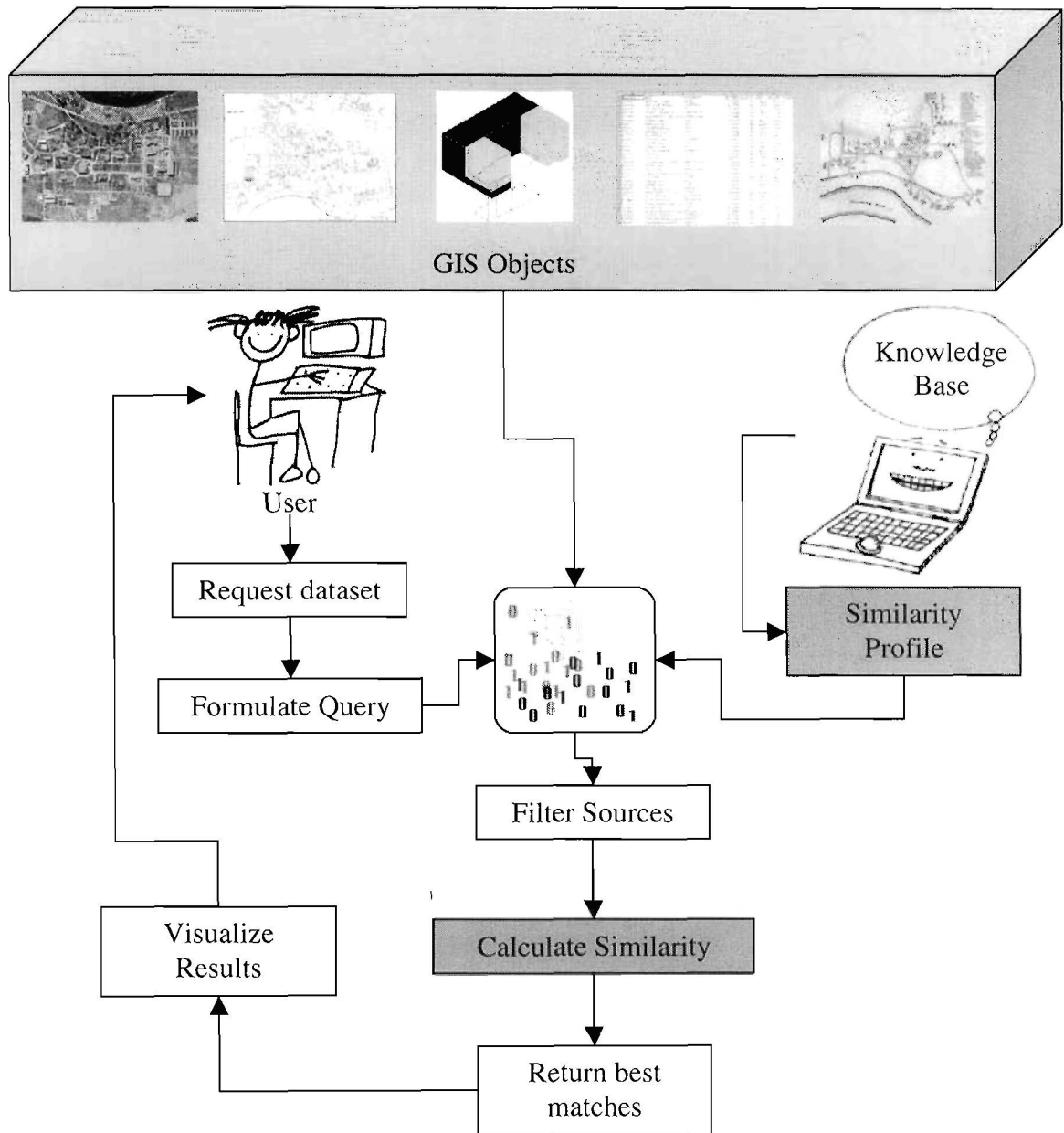


Figure 1.2: Query process of a geospatial source collection

- i. Users request an information object from the database (or more than one).
- ii. Their request is translated into a structured query that the system understands and that is compatible with the database collection (e.g. using ontologies).
- iii. A query language is used as a mediator between user and database (e.g. SQL).
- iv. Based on their request an indexing mechanism is used to return all potentially similar objects, in essence filtering dissimilar ones to speed up the process.
- v. *On this filtered object collection a similarity algorithm with properties extracted from a knowledge base is applied. The output is either a certain number of best answers (e.g. 10 best datasets) or answers within a specific similarity range (e.g. higher than 80%).*
- vi. The results are presented to the users to assess their similarity accuracy.

In the above information flow there are several areas of interest that the database community is working on. Various disciplines are involved in the process and many different approaches have been proposed. Our work concentrates on step number v on the previous list. Our goal is to develop a similarity algorithm that will rank the results in an accurate way. Therefore our main focus is information retrieval accuracy. Retrieval speed is not a primary target even though we optimize our system whenever possible. Issues such as multi-dimensional indexing that are related to the process in the general sense are not addressed.

1.3.2 Approach specifics

As mentioned above our goal is to develop a preference-based similarity learning system. This issue has been previously tackled by a variety of disciplines like artificial intelligence, statistics, computer science, cognitive science and psychology. Several machine learning general methodologies have been used like genetic algorithms, decision trees, neural networks and non-parametric algorithms.

In this thesis we develop a learning system to optimize query returns of geospatial information. Some clarifications leading to the focus of this work follow.

- **Applicability within the geospatial domain**

Geospatial information has certain characteristics that differentiate it from other types of information when it comes to user preferences. Of primary interest to this thesis is the fact that (one-dimensional) geospatial parameters are quantifiable and continuous. Thus a user may easily express his/her preference as functions of such quantifiable and continuous parameters. Considering for example resolution, a user may state and quantify preferences along the lines of the following: *I am interested in imagery with a resolution of 50cm, and my interest drops linearly/exponentially as resolution decreases.* In this manner, a GIS user can quantify expressions of preference in terms of resolution, making e.g. an aerial photograph with 2m resolution twice as suitable for his/her application than another with 4m resolution. Relations among these properties are not only ordinal, but metric as well.

This richness and metric structure of geospatial information (and corresponding user preference patterns) is a very important aspect that differentiates geospatial from other types of information collections (e.g. text databases, stock prices). Of course, there still exist a number of properties that may be contained in a GIS but do not have such

structure (e.g. land use, ownership), but these attributes are beyond the scope of this thesis. Our focus is on one-dimensional quantitative attributes (e.g. scale, resolution, area, azimuth) and on handling complex user preferences that deviate from the above mentioned linear or exponential models.

Furthermore, our intention is not to develop a global technique that would be applicable to any domain as a traditional artificial intelligence approach is expected to do. We focus on the characteristics of queries performed on geospatial information and optimize performance based on these. Subsequently, when evaluating the performance of our approach we do so by comparing our methodology to existing solutions in geospatial information retrieval and we do not extend to additional algorithms developed for other machine-learning tasks. Of course, certain ideas and approaches from this dissertation may eventually find applications in other domains.

- **Support but not investigation of cognitive assumptions**

Similarity assessment has attracted attention from cognitive scientists due to the human factor presence. In our work, we support some basic assumptions that scientists have determined (e.g. exponential decrease of similarity the further away from the target value) and we build our mathematical model on that. We do not question these principles, therefore no human testing is performed. We see our work as a regression optimization issue and we address it within that context. Evaluation is based on statistical error measures to show good generalization, solution stability and high adaptability capabilities.

- **Object similarity vs. scene similarity**

We address the issue of similarity learning in queries requesting specific geospatial information objects and not a combination of them. Spatial relations expressing topology,

cardinality and distance measures between objects are outside the scope of this work.

Another important distinction should be made as to the types of attributes that our learning algorithm supports, leading to three additional clarifications:

- **Measurement scale**

Many taxonomies have been introduced based on different measurement scales. Such examples include counts versus measurements, qualitative versus quantitative, and metrical versus categorical measurements (Hand et al., 2001). For our categorization we make use of the four scales of measurement as introduced by Stevens (1946), namely nominal, ordinal, interval and ratio measurement types.

- Nominal: Data that do not have a natural ordering fall in this category. They can be numbers or text and they are used as labels or names, such as for instance owner names of land parcels.
- Ordinal: These attribute types are ordered but do not express information about the differences between the ordered values. An example would be the values of “black”, “gray”, and “white” for color.
- Interval: An interval attribute has numerical distances between any two levels of the scale. However, they do not have a measurement origin though. A typical example would be a temperature reading (Fahrenheit, Centigrade, etc.).
- Ratios: When attribute values have an origin in addition to being interval, then they belong to the ratio type. An example would be distances.

From the similarity perspective, interval and ratio scales of data require similar learning techniques. Their only difference relies on having an origin of measurement or not. This can be rectified through appropriate normalization, a common preprocessing

step in data preparation for machine learning algorithms. It is important to mention that even though the type of data in this category is common with other non-geographic data collections preference might not be. For example, the “Ground Pixel Size” thematic attribute can demonstrate diverse preference based on the application usage. Photogrammetrists may be less flexible than oceanographers with respect to larger pixel sizes, just as non-profit organizations might be more stringent towards smaller pixel size due to acquisition costs.

Ordinal data do have some relative order, but because this distance between ordered values is not quantifiable, regression techniques (e.g. neural networks) are not easy to apply. Other methods such as decision trees might be more appropriate. As for the last scale category, the nominal, it is a textual matching process. Learning involves identification of possible relations between nominal values (e.g. synonyms, same root). A thesaurus is often used and learned domain knowledge is incorporated to represent these relations. Many methods used in this category overlap similarity learning in textual databases.

The methodologies developed in this thesis apply to measurements in the interval and ratio scales. The proposed learning algorithm focuses on user preference modeling in quantitative attributes that describe geospatial information (e.g. time, scale, azimuth). Qualitative attributes require a different treatment for similarity assessment and this issue is reserved for future work. It should be noted that within the context of this work, quantitative attributes should fall into the interval or ratio scale. Even if an attribute is represented by a metric that does not necessarily mean that it is quantitative (e.g. postal code is treated as a qualitative one).

- **One-dimensional vs. multi-dimensional attributes**

Our approach concentrates on one-dimensional attributes. It does not extend to multi-dimensional attributes where dependencies might exist among dimensions. For example, we do not examine the spatial attribute where X and Y are heavily dependent, and color attribute where Red, Green and Blue also exhibit high correlation. We should clarify that even in one-dimensional attributes independence should exist among attribute values. An attribute value should not be a combination of two or more original attributes, stored for instance in the same field for indexing or storage purposes (e.g. using a Peano-Hilbert transformation). We also do not access the content of the geospatial objects, so one-dimensional queries that cannot be answered by a metadata descriptor are beyond the scope of this work. Such an example would be a query for an aerial photograph with a building having an area of 100 m². If that information is pre-extracted and supplied with the object then our approach is applicable, but if this information requires an object-extraction operation on the photograph this is not supported in this work.

- **Single attribute vs. combination of attributes**

A last remark involves the applicability level of our algorithm. We calculate similarity within each attribute but do not aggregate similarity results between attributes into one total metric. Nonetheless, our approach can be easily incorporated into a multi-dimensional similarity assessment approach (e.g. a weighted nearest neighbor), which in fact is the general framework under which our method operates. Correlation between attributes mandates a complex approach that is the next logical extension of our work in the future.

1.4 Research questions and objectives

The goal of this work is to enhance the existing information retrieval methods in geospatial applications. Within the context of our preference-based similarity learning algorithm several issues arise, namely:

Model adaptability/accuracy. First and foremost, the investigated technique has to show high adaptability to various cases that may be present. A flexible model is required with high degree of freedom to compensate for unusual behavior. The model should present good generalization over a variety of training sample cases. The results should outperform existing techniques in terms of retrieval accuracy currently used for geospatial information retrieval.

Model convergence/robustness. An important goal of any learning system is not only to have the potential to behave well, but also to do so in a consistent manner. This is an extremely challenging task when it comes to complex non-linear systems. Optimization is achieved through minimization of an appropriately chosen statistical error measure. Sometimes local minima misguide the solution to undesired results. Therefore, caution should be exercised when designing and training the system.

Model control. Another issue that relates to complex systems is that as complexity increases it becomes more difficult to control system behavior, and contribution of each processing element to the overall solution might be hard to identify. Thus, erroneous results are difficult to investigate and correct. Furthermore, this “transparency” of the system would allow user interpretation of the behavior of each processing element. So our system should be complex enough to model the underlying problem but simple enough to train and analyze.

In accordance to the above the hypothesis of this thesis is:

“The proposed neuro-fuzzy preference learning system outperforms the distance-based nearest neighbor algorithm in geospatial attributes”.

In the remaining parts of the thesis we demonstrate the benefits of our approach over existing methods. We also show how our system identifies the nearest neighbor as a sub-case during the learning process and supports it. At the same time more complicated user preferences are modeled if necessary. Therefore, our approach by design is superior to the nearest neighbor.

An important factor in our evaluation process is the robustness of our method. A more complex method than the nearest neighbor can be proposed but it would be a mediocre one without ensuring a good, consistent performance. Our statistical simulations will confirm our method’s robustness.

We should mention here that “performance” is measured in terms of relevancy accuracy in the obtained results and not associated with computational retrieval speed. Naturally our approach is more computationally expensive than the nearest neighbor. The exact retrieval speed cost depends on underlying complexity of our identified similarity preference. It should be kept in mind though that because of the gradual complexity increase in our training process, complex functions are used only where necessary. Also, the more complex the preference model is, the more complex user preference is (that created that complex model), and consequently, the less appropriate the distance-based nearest neighbor is. So eventually a database designer will evaluate based on task requirements and computational resources the trade-off between our more accurate and computationally expensive approach and the faster but not so accurate nearest neighbor.

1.5 Major results and contribution of thesis

The major contribution of this thesis is the development of a learning system to express user preference of similarity within geospatial environments. Experiments on simulated datasets demonstrate robustness and advanced modeling capabilities of the proposed technique. Our system has the ability to adapt to different scenarios and express them successfully through its mathematical model. It supports a variety of cases, and modeling accuracy through statistical simulations was found to be high and consistent. Therefore, users accessing geospatial environments will have the option of a query system that adjusts to their preference.

In order to achieve our goal, some desired characteristics of our algorithm were outlined in the previous section, namely high accuracy and convergence rate, and also transparency in the system design. Here is how these issues are addressed from the perspective of our system and the novel methodologies used leading to that.

Development of a novel learning system architecture. To achieve our goal to model user similarity preference we developed a neuro-fuzzy system. Its modular design allows a variety of knowledge rules to be incorporated and shows flexibility in the complexity addressed within each processing element. Our system supports advanced modeling capabilities due to its ability to distinguish expected similarity behavior from localized unusual similarity preference. The expected behavior is captured by a global fuzzy membership function whose complexity grows with the problem difficulty, and unexpected behavior is described by a customized multi-scale radial basis function (MSRBF) neural network. Our multi-stage training ensures adaptability and control in our system performance.

Development of a novel multi-scale RBF neural network. Our innovative neural network design combines local (i.e. within node receptive fields) and global fit accuracy metrics to produce an architecture that can identify and model various trends in user preferences without expanding to undesired areas of the input space. Highly localized trends are identified first and this exposes larger scale trends that may otherwise have remained hidden. In doing so, our MSRBF outperforms traditional RBF architectures in generalization accuracy, and by using less number of nodes it accelerates simulation.

1.6 Intended audience

The intended audience of this thesis includes researchers and developers working on database systems, especially in the communication process, and on intelligent systems. Fields like computer science and GIS can benefit from the implementation of our model. Also techniques developed for our learning process can be extracted and implemented in other tasks of similar requirements and constraints. Therefore, scientists in the artificial intelligence discipline concerned with machine learning can find interest in this work.

1.7 Thesis organization

In this chapter we provided a brief introduction of the problem and the motivation behind it. Characteristics of the proposed solution were identified and a preliminary discussion of our contribution took place. A short description of the remaining chapters follows:

Chapter 2. Background work related to this thesis is introduced. The general framework of data mining and some challenging tasks within its community are described. As the chapter progresses so does the depth of analysis focusing on methodologies closely related to our approach. Throughout the literature review the reader can see where our work fits with respect to existing methods and categories.

Chapter 3. The purpose of this chapter is twofold, namely to discuss the concerns and limitations of the research presented in chapter 2, and to build our model based on those. Some additional existing research such as neuro-fuzzy techniques is reserved for this chapter rather than the previous one, due to its direct relation to our model. Theoretical justification of our model is provided, as expressed through its potential modeling and control capabilities.

Chapter 4. This chapter presents the fuzzy membership functions used to model similarity within various geographic information dimensions (attributes). Explanation of the chosen functions takes place. Also, the training process based on progressively increasing complexity is discussed. Furthermore, the framework where a more simple function acts as approximation for more complex ones is described.

Chapter 5. Fuzzy functions can model user similarity to some degree, but cannot adjust to local unexpected behavior. Therefore, we developed a customized radial basis neural network to capture errors from the previous process. Specific properties of the networks are discussed in this chapter, limitations of current networks are presented and solutions to address them are shown. Thus, a novel multi-scale network is developed to facilitate our needs.

Chapter 6. Following the fuzzy functions and the neural network of the previous two chapters, chapter 6 shows how these are combined to form our neuro-fuzzy system.

Chapter 7. This chapter provides evidence of the benefits produced by our approach. Statistical testing, functionality examples, and accuracy assessment are presented in this section, demonstrating that our system outperforms existing techniques.

Chapter 8. Major findings, a brief summary and future directions are outlined.

Chapter 2

Literature review

The approach employed in this thesis is the result of a convergence of influences from a number of different fields. The goal of this chapter is to review the pertinent literature from these diverse fields and to provide the necessary background for the remainder of the work.

Traditionally analysts have performed the task of extracting useful information from recorded data. As datasets have grown in size and complexity there has been an inevitable shift away from direct hands-on data analysis towards indirect, automatic techniques using more complex, sophisticated tools. Modern technologies of computers, networks, and sensors have made data collection and organization an almost effortless task. However, the captured data needs to be converted into information and knowledge to eventually become useful. Data mining is the entire process of applying computer-based methodologies, including new techniques for knowledge discovery, on data.

Here we should mention that there is no clear difference between mining and information retrieval when multimedia data is dealt with (Boca Raton, 1999). As cited in (Natsev et al., 2004) “applications requiring content-based querying and searching of images abound and can be found in a number of different domains that include data mining...”. So from the above we can see that the traditional line separating data mining and information retrieval does not exist any more. For the purposes of this review we use the term “data mining” as an umbrella incorporating some well-known traditional mining tasks such as pattern identification, and our task of intelligent information extraction.

Therefore we begin this review with a short introduction to data mining and its multi-disciplinary history. Major tasks performed within data mining are introduced and our method is categorized among them. A discussion on data mining progress on geographical datasets is offered. Key algorithms in machine learning are outlined and a detailed description of the inductive and deductive learning approaches is presented followed by a classification of our algorithm among these approaches. We focus on nearest neighbor techniques and its variants and we discuss the variety of similarity functions they employ. We also give a brief literature review on user preference learning.

This chapter does not provide details on all the background material used but discusses broad-spectrum research that supports most of the later work. Background material specific to particular proposed methods will be presented in later chapters. Examples include a detailed comparison of selected techniques that led to the design of our system (Chapter 3), and an in depth look of Radial Basis Function neural networks (Chapter 5).

2.1 Data mining and knowledge discovery in databases

In recent years an explosive growth of many business, government and scientific databases is notable. This increase of data availability has far outpaced the ability to interpret and digest this data creating the need for advanced tools and techniques for automated and intelligent analysis. Development of such tools and methods is the subject of the rapidly growing field of knowledge discovery in databases (KDD).

The terms KDD and data mining are often used interchangeably. Additional terms used include knowledge extraction, information discovery, exploratory data analysis,

information harvesting and unsupervised pattern recognition. These terms can be characterized by the following (Dunham, 2002):

- Knowledge discovery in databases is the process of finding useful information and patterns in data.
- Data mining is the use of algorithms to extract the information and patterns derived by the KDD process.

According to (Fayyad et al., 1996) the KDD process is composed of the following five steps shown in figure 2.1.

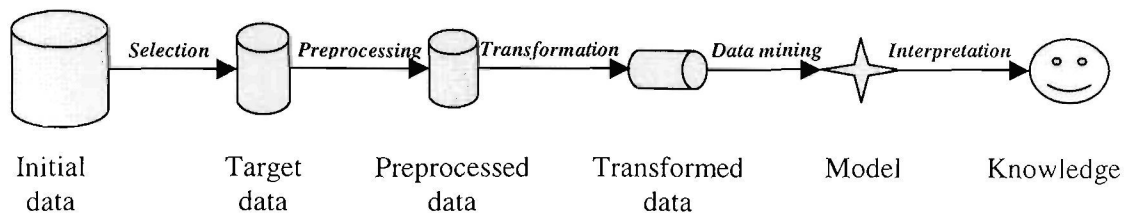


Figure 2.1: Knowledge discovery process

Selection: In this first step the data needed for the data mining process is obtained from many different and heterogeneous sources. These sources might collect data from various databases, files, and non-electronic sources.

Preprocessing: The data to be used by the process may have incorrect or missing data. There may be anomalous data from multiple sources involving different data types and metrics. Many different activities might be performed at this stage. Erroneous data may be identified and removed, whereas missing data must be supplied or predicted.

Transformation: At this step, data from different sources are converted into a common format for processing. Some data may be encoded or transformed into more usable formats. Data reduction may be used to decrease the number of possible data values under examination.

Data Mining: This step applies algorithms to the transformed data to generate the desired results. Our algorithm falls in this category.

Interpretation/evaluation: Various visualization and GUI strategies are used at this last step. This is an important step because the usefulness of the results obtained through the data mining is dependent on it.

2.2 Data mining multi-disciplinary history

The current evolution of data mining algorithms is the result of years of influence from different disciplines. A major trend in the database community is to combine results from these seemingly different disciplines into one unifying algorithmic approach. This is the underlying idea of our approach as well so it is interesting to examine how data mining evolved through the years. The extensive variety of data mining problems combined with different research fields often leads to different perspectives based on the background of the researcher. We may find that similar problems and sometimes even similar solutions are described differently. For example, statisticians often raise concerns over the use of approximations with results being generalized where they should not be. Database researchers may doubt the efficiency of AI algorithms, especially in large datasets. Information retrieval scientists may complain about the lack of applicability of data mining algorithms in textual databases since they concentrate on numeric values.

Table 2.1 (Dunham, 2002) shows developments in the areas of Artificial Intelligence (AI), Information Retrieval (IR), Databases (DB) and Statistics (Stat) leading to the current view of data mining. For an extended review of the statistical methods developed over the past 40 years and their contribution to KDD the reader is advised to check (Elder and Pregibon, 1996).

Time	Area	Contribution	Reference
Late 1700s	Stat	Bayes theorem of probability	(Bayes, 1763)
Early 1900s	Stat	Regression analysis	
Early 1920s	Stat	Maximum likelihood estimate	(Fisher, 1921)
Early 1940s	AI	Neural networks	(McCulloch and Pitts, 1943)
Early 1950s		Nearest neighbor	(Fix and Hodges, 1951)
Early 1950s		Single link	(Florek et al., 1951)
Late 1950s	AI	Perceptron	(Rosenblatt, 1958)
Late 1950s	Stat	Resampling, bias reduction, jackknife estimating	
Early 1960s	AI	ML started	(Feigenbaum and Feldman, 1963)
Early 1960s	DB	Batch reports	
Mid 1960s		Decision trees	(Hunt et al., 1966)
Mid 1960s	Stat	Linear models for classification	(Nilsson, 1965)
	IR	Similarity measures	
	IR	Clustering	
	Stat	Exploratory data analysis	
Late 1960s	DB	Relational data model	(Codd, 1970)
Early 1970s	IR	SMART IR system	(Salton, 1971)
Mid 1970s	AI	Genetic algorithms	(Holland, 1975)
Late 1970s	Stat	Estimation with incomplete data (EM algorithm)	(Dempster et al., 1977)
Late 1970s	Stat	K-means clustering	
Early 1980s	AI	Kohonen self-organizing map	(Kohonen, 1982)
Mid 1980s	AI	Decision tree algorithms	(Quinlan, 1986)
Early 1990s	DB	Association rule algorithms	
		Web and search engines	
1990s	DB	Data warehousing	
1990s	DB	Online analytic processing (OLAP)	

Table 2.1: Time line of data mining development

2.3 Data mining tasks and similarity learning

Data mining is one of the fastest growing fields in the computer industry. Once a small interest area within computer science and statistics, it has quickly expanded into a field of its own. One of the great strengths of data mining is reflected in the wide range of methodologies and techniques that can be applied to a host of problem sets. It is a cooperative effort of humans and computers. Best results are achieved by balancing the knowledge of human experts in describing problems and goals with the search capabilities of computers.

In practice, the two “high-level” primary goals of data mining tend to be prediction and description (Fayyad et al., 1996; Dunham, 2002; Kantardzic, 2002). *Prediction* involves using some variables or fields in the dataset to predict unknown or future values of other variables of interest. *Description* focuses on finding human-interpretable patterns or relationships in the data. Thus we can categorize data mining activities into one of the two categories:

- Predictive data mining which produces the model of the system described by the given dataset, or
- Descriptive data mining that produces new, non-trivial information based on the available dataset.

Several grouping schemas have been proposed in the literature, especially in books and introductory tutorials on data mining. There is a significant overlap between them and sometimes their distinction is based solely on terminology. For our review we use the rather complete task representation of figure 2.2 as presented in (Dunham, 2002).

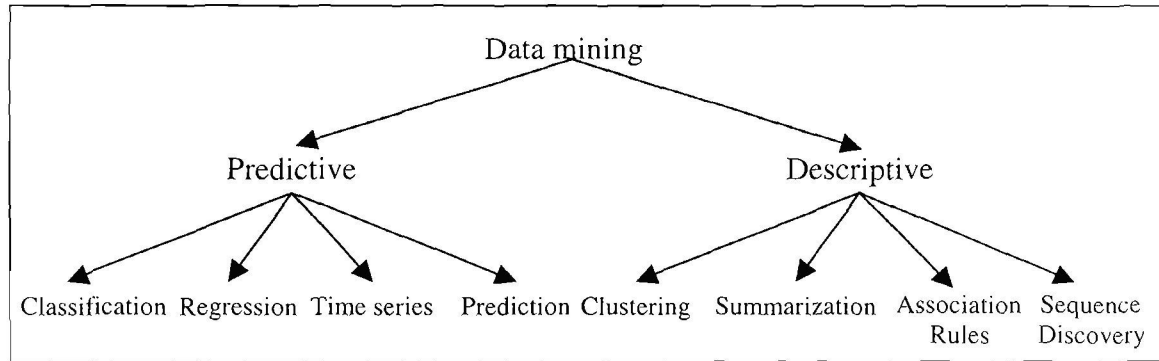


Figure 2.2: Data mining primary goals and tasks

2.3.1 Predictive tasks

The following tasks are categorized as predictive based on their functionality: classification, regression, time series analysis and prediction. Here we should mention that borderlines along these tasks are not crisp since one task might borrow techniques developed for another. Nonetheless, each of the four tasks has its distinct methodologies so this categorization can hold true.

Classification is learning a function that maps a data item into one of several classes (Hand, 1981; Weiss and Kulikowski, 1991; McLachlan, 1992). It is often referred to as supervised learning because the classes are determined before examining the data. Pattern recognition is a type of classification where an input pattern is classified into one out of several classes based on its similarity to these predefined classes. For example in face recognition a feature vector is produced describing facial characteristics (distance between eyes, size and shape of mouth, shape of head, etc.). This is then compared to the entries in a database to see if there is a successful match.

Regression is used to map a data item to a real-valued prediction value. This is done by learning a function that does this mapping. Regression assumes that the target

data fit into some known type of function (e.g. linear, logistic, etc.) and attempts to identify the best function that models the given data. This identification is done using error analysis techniques. An example of regression is the calculation of recovery probability of a patient based on a set of diagnostics.

Time series analysis examines the value of an attribute as it varies over time. There are three basic functions performed. Distance measures can be used to evaluate how similar different time series are. Furthermore, the time series can be examined to determine its behavior based on its structure. Finally data from historical time series can be used to allow prediction of future values.

Prediction allows the estimation of future states based on past and current data. It has many real-world data mining applications such as flooding, speech recognition and earthquake prediction. Even though prediction can be seen as a type of classification or sometimes time series analysis or application of regression methods, it should be recognized as a distinct task since other techniques may be used as well.

2.3.2 Descriptive tasks

In the descriptive tasks the properties of the data are examined as a way to explore the properties themselves and not to predict new properties. Clustering, summarization, association rules and sequence discovery are usually viewed as descriptive in nature.

Clustering is a common descriptive task where one seeks to identify a finite set of categories or clusters to describe the data. It is similar to classification except that the groups are not predefined but rather defined by the data alone. Therefore, clustering is alternatively portrayed as unsupervised learning or segmentation. Clustering is usually accomplished by determining the similarity among the data based on predefined

attributes. The most similar data are grouped together. The clusters may be mutually exclusive and exhaustive or consist of a richer representation such as hierarchical or overlapping categories. A clustering example in KDD would be the identification of sub-populations of consumers in marketing applications.

Summarization involves methods for finding a compact description for a subset of data. It is also called characterization or generalization. It extracts or derives representative information from a database. Summarization techniques are often applied to interactive exploratory data analysis and automated report generation. A simple example would be the representation of fields based on their mean and standard deviation.

Association rules refer to the data mining task of uncovering relationships among data. It is also called link analysis, affinity analysis or dependency modeling. They try to find a local model that describes significant dependencies between variables or between the values of a feature in the dataset or in parts of it. A frequent application of this task involves its use in the retail sales community, for example to identify items that are purchased together.

Sequence discovery is used to determine sequential patterns in data. These patterns are based on a time sequence of actions. They are similar to association rules in the sense that data are found to be related, but the relationship is based on time. An example would be the discovery of sequence within which goods are purchased (e.g. people who purchase CD players may purchase Audio CDs within one week).

2.3.3 Similarity learning within data mining

The similarity learning task can utilize methods from the wide range of data mining tasks. Similarity learning involves the *classification* of an input into one of several classes, based on its similarity to these predefined classes. Several methodologies can be borrowed from classification, when the output of the similarity learning algorithm is discrete (i.e. categorical). If the output is continuous then *regression* techniques are more appropriate. These techniques are used to map a data item to a real-valued prediction value by learning a function that does this mapping. An important trend in recent years is the incorporation of temporal information in GIS. *Time series analysis* examines the value of an attribute as it varies over time, therefore useful techniques can be borrowed.

The above tasks explicitly help in a similarity learning process. In addition to these there are some others that can optimize the learning process, without affecting the similarity learning per se. *Association rules* is one such example, where relationships are uncovered among data. Such analysis can help for example to learn dependencies between successive similarity queries, in other words project future queries. *Clustering* can also contribute by determining the similarity among the data based on predefined attributes. It can be used as a pre-processing step. *Summarization* techniques are often applied to interactive exploratory data analysis and automated report generation and can be integrated with the input/output of a similarity learning algorithm, but not the learning process itself.

2.4 Geographic data mining

Data mining techniques have been applied to a variety of real-life applications and new applications continue to drive research in the area. To date most data mining research concentrates on relational and transactional data. Despite the importance and proliferation of geospatial datasets, work in this field has appeared only recently (Gunopulos, 2001). Nonetheless, temporal and spatial data mining continue to grow rapidly as exciting subfields of data mining. There are many reasons for this, including the following (Roddick et al., 2001):

- Growth in the volume of data being collected and requiring analysis.
- Increase in data availability through the Internet and as a result of electronic commerce and inter-enterprise applications.
- Recognition of the value and commercial advantage that the results of geographic data mining can provide.
- The realization that temporal and spatial data are special and need to be explicitly accommodated.

In the next section we examine the distinction of geospatial data from others for data mining purposes. This distinction is important as it propagates to our similarity learning algorithm datasets and requirements. We also provide a brief summary of current geospatial mining issues under investigation, where the lack of research in similarity learning algorithms tuned for geospatial data is notable.

2.4.1 Special characteristics of geographic data mining

The recent digital geographic data explosion is not different from other areas such as marketing, biology and astronomy. But is there a difference between geographic data

mining and data mining in other fields? Several papers were recently published addressing this issue (Yuan et al., 2001; Miller and Jiawei, 2001; Gahegan, 2001) and are used in the following review.

Many of the challenging issues arise from the fact that geography is an integrative discipline. Geographic data span a wide range of perspectives and interests from the social to the physical aspects of the problem. This mixture of perspectives coupled with the growing infrastructure for gathering information pose the following obstacles:

Complexities associated with data volume. Like many disciplines where data mining is applied, geography is rich in data. Many of the large consumer, medical and financial transaction databases now being constructed contain spatial and temporal attributes and hence offer the possibility of discovering or confirming geographical knowledge (Miller and Jiawei, 2001). Explicitly, geographical datasets of terabyte proportions are now in existence and traditional retrieval methods have a hard time to keep up.

Complexities associated with the domain itself. Interesting and relevant signals in data are often entirely hidden by stronger patterns that must first be removed. Many of these complexities originate from spatial and temporal codependence that occurs across a variety of scales and from a variety of causes (Roddick and Lees, 2001). For example, the cyclic nature of many geographical systems (daily, seasonal, annual, circulatory, El-Nino, sunspot) imposes a heavy signal on data that will overshadow more localized variance.

Complexities caused by local variation. Earth systems are so intrinsically interconnected that it is difficult to isolate an analysis conducted on some part of a system from the affects of other unmodeled aspects. Measured geographic attributes often exhibit the properties of spatial dependency and spatial heterogeneity. The former refers to the

tendency of attributes at some locations in space to be related; typically, these are proximal locations. The latter refers to the non-stationarity of most geographic processes, meaning that global parameters do not reflect well the process occurring at a particular location. While these properties have been traditionally treated as nuisances, contemporary research fueled by advances in geographic information technology provides tools that can exploit these properties for new insights into geographic phenomena (e.g. Anselin, 1995; Brunson et al., 1996; Fotheringham et al., 1997; Getis and Ord, 1992; Getis and Ord, 1996). Some preliminary research in geographic knowledge discovery suggests that ignoring these properties affects the patterns derived from data mining techniques (Chawla et al., 2001). More research is required on scalable techniques for capturing spatial dependency and heterogeneity in geographic knowledge discovery.

Complexity of spatiotemporal objects and patterns. Another unique aspect of geographic information in knowledge discovery is the complexity of spatiotemporal objects and patterns. In most non-geographic domains, data objects can be meaningfully represented discretely within the information space without losing important properties. This is often not the case with geographic objects: size, shape and boundaries can affect geographic processes, meaning that geographic objects cannot necessarily be reduced to points or simple line features without information loss. Relationships such as distance, direction and connectivity are also more complex with dimensional objects (Egenhofer and Herring, 1994; Okabe and Miller, 1996; Peuquet and Zhang, 1987). Transformations among these objects over time are complex but information-bearing (Egenhofer and Hornsby, 2000). The scales and granularities for measuring time can also

be complex, preventing a simple "dimensioning up" of space to include time (Roddick and Lees, 2001). Developing scalable tools for extracting patterns from collections of diverse spatiotemporal objects is a critical research challenge. Also, since the complexity of derived spatiotemporal patterns and rules can be daunting, a related challenge is making sense of these derived patterns, perhaps through "meta-mining" of the derived rules and patterns (Roddick and Lees, 2001).

Complexities caused by data gathering and sampling. Although data are available in increasing volume, it is still often the case that we must resort to surrogates for the phenomena of interest, rather than direct measurements. Furthermore, data are often provided in spatially and temporally aggregated forms that themselves give rise to many interpretation problems (e.g. in cluster detection algorithms).

Difficulty in formalizing the geographic domain. One of the main difficulties with knowledge discovery activities within the geographical domain is the complex conceptualization necessary. There is, as yet, no universally accepted conceptual model of geography (e.g. Goodchild, 1992), and the models that are currently implemented in commercial GIS vary significantly one from another, often in quite fundamental, philosophical ways. This leads to three distinct problems (Yuan et al., 2001):

- a. Data are often intrinsically non-commensurate; they cannot be directly compared or combined.
- b. It is difficult to apply formal geographical knowledge to the process of knowledge discovery, since such knowledge is not readily available.
- c. When new knowledge is uncovered it is difficult to represent that knowledge formally.

To summarize, the development of data mining and knowledge discovery tools must be supported by a solid geographic foundation that accommodates the unique characteristics and challenges presented by geospatial data. The emergence of national and global geospatial data infrastructures to date has been ad-hoc. Contributed data has not been coupled with contributed tools for data analysis and modeling. Data mining and knowledge discovery methods have not been implemented to deal effectively with geospatial data, whose sensitivities are known widely to geographers. As our understanding of the nature of geographic information and its sensitivities to spatial, temporal and spectral measurement improve, it is probable that refinement of data mining algorithms will prove insufficient; therefore design of new procedures and knowledge validation procedures will begin to emerge. We see intelligent information extraction as a new important issue within geospatial data mining. Context-specific knowledge expressing user/application preferences should be incorporated using intelligent systems.

2.4.2 Tasks within geographic data mining

Over the past four years there has been a substantial increase in temporal, spatial and spatio-temporal data mining applications and a variety of papers have been published. In this section we introduce some general categories of tasks performed within geographic data mining. Despite much research stretching across these categories, their categorization has been retained for continuity reasons. This section provides a useful guide rather than an exhaustive classification. According to (Roddick and Spiliopoulou, 1999; Roddick et al., 2001) there are some general areas of interest:

- **Frameworks.** This category includes research dealing primarily with models for spatial and temporal knowledge discovery.

- **Temporal and spatial association rule mining.** This category combines contributions to the problem of discovering association rules from temporal or spatial data.
- **Discovery of temporal patterns.** This research is concerned with the discovery of patterns or trends over time. The data itself need not be temporal but ordering is required.
- **Time series mining.** This category includes research on occurrence of events over time.
- **Discovery of causal and/or temporal rules.** This category contains works that search for temporal relationships between (sets of) events.
- **Spatial data mining.** Relevant work to spatial and geo-referenced data mining is the subject of this general task category.
- **Spatial and spatio-temporal clustering techniques.** This category includes research on algorithms or frameworks for spatial and spatio-temporal clustering.
- **Spatio-temporal data mining.** This category contains works that accommodate the special semantics of both space and time.

An extensive list of papers classified in the above categories is presented in (Roddick et al., 2001). Similarity learning falls into the last category, that of spatiotemporal data mining. We do not discover patterns from data in the strict sense, but we discover user preference patterns when requesting data. Spatial similarity preference tied together with temporal and other thematic attributes is a challenging task that a complete preference learning algorithm for geospatial information should address.

2.5 Machine learning and similarity learning

Machine learning as a combination of artificial intelligence and statistics has proven to be a fruitful area of research, spawning a number of different problems as well as algorithms for their solution. These algorithms vary in their goals, training datasets, learning strategies and representation of data (Kantardzic, 2002). Similarity learning is one of many machine learning applications in databases. Before we discuss some representative algorithms we provide an insight to a machine learning approach based on the different types of learning systems and we position our similarity learning task within them.

2.5.1 Positioning similarity learning in the general machine learning categories

If we relate the problem of learning from data to the general notion of inference in classical philosophy, two main phases are identified:

- i. *Induction*: Learn or estimate unknown dependencies in the system from a given training set.
- ii. *Deduction*: Use the above dependencies to predict new outputs for future input values in the system.

The two phases are shown graphically in the next figure.

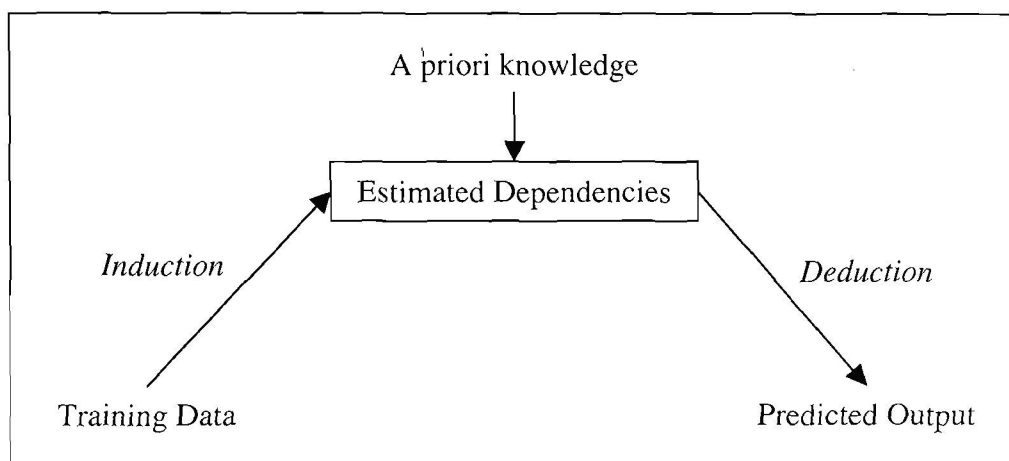


Figure 2.3: Types of inference: induction and deduction

Induction can be seen as the progress from particular cases (training data) to a general mapping or model. On the other hand, deduction starts with a general model and using given input values it progresses to particular cases of output data. Clearly, similarity learning is an inductive task since the model is not known in advance; it is identified through the training process.

There are two types of inductive learning methods, the supervised and the unsupervised approaches. *Supervised learning* is used to estimate an unknown dependency from known input-output samples. A supervised approach learns by example. A training input should be provided together with some correct answers (output). The term “supervised” denotes that the output values are known, in essence provided by a teacher. Based on its ability to handle the provided input-output dataset the goal for the model is to learn the correct behavior and be able to expand (generalize) to any potential entry. A schema of supervised technique is shown in figure 2.4.

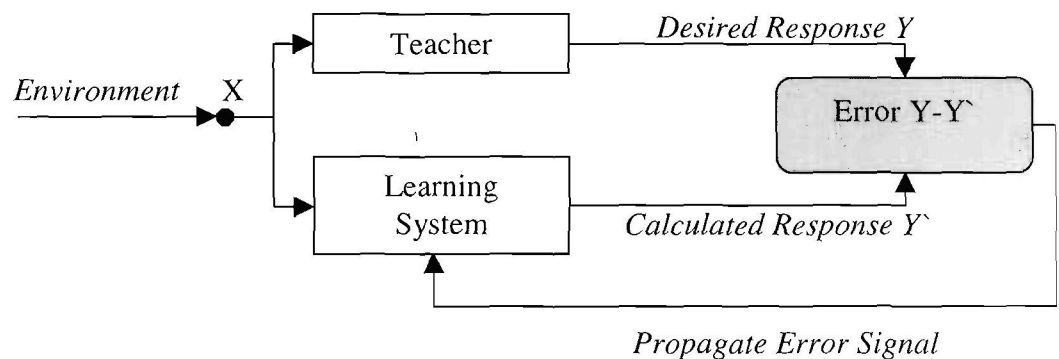


Figure 2.4: Supervised learning

Under the *unsupervised learning* scheme only input values are provided to the learning system. There is no notion of the output during the learning process. Unsupervised learning does not require a teacher; the learner forms and evaluates the

model on its own. The goal is to uncover the structure of the input data. This happens after the system adapts to the regularities of the input data. It forms internal representations for encoding features of the input examples either in a local or a global level. Clustering, summarization and association rules are typical unsupervised tasks. Figure 2.5 shows a simplified version of an unsupervised learner.

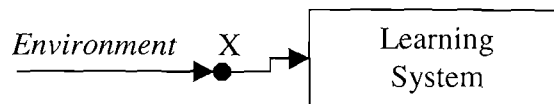


Figure 2.5: Unsupervised learning

Since most similarity learning algorithms learn from example they can be categorized as a supervised inductive task. The user is required to provide a similarity evaluation to a presented example, acting as a teacher for the algorithm. Popular machine learning methods that could be used for supervised learning include neural networks, decision trees, instance-learners, genetic algorithms and others.

2.5.2 Machine learning methods and their applicability for similarity learning

In this section we briefly present some popular machine learning methods and discuss their applicability for our similarity learning task.

2.5.2.1 Decision trees

An efficient method for predicting classifiers from data is to generate decision trees. The decision tree representation is a widely used logic method. In the machine learning and applied statistics literature a large number of decision-tree induction algorithms can be found. They belong to the supervised learning category since they create trees from a set of input-output samples.

A typical decision tree learning system adopts a top-down strategy that searches for a solution in a part of the search space. It is a flow-chart type structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and leaf nodes represent classes or class distributions (Han and Kamber, 2001). In order to classify an unknown sample the attribute values of the sample are tested against the decision tree. A path is traced from the root to a leaf node that holds the class prediction of that sample. Therefore decision trees can be easily converted to classification rules.

There are many advantages to the use of decision trees for classification. They are easy to use and efficient. Generated rules are easy to interpret and understand. They scale well for large databases because the size of the tree is independent of the database size (Dunham, 2002). On the other hand disadvantages also exist, with the most important one from the similarity learning perspective being their inability to handle easily continuous data. In order to do so these attribute domains must be divided into categories. The domain space is divided into hyper-rectangles. Handling missing data is difficult and overfitting may occur. The latter can be overcome via tree pruning. Finally another important drawback is the lack of incorporation of correlations among attributes in the decision tree process.

Some well-known examples of decision trees include the ID3 algorithm (Quinlan, 1986), the C4.5 (Quinlan, 1993), which extends the domain classification from categorical attributes to numeric ones, and PRISM (Cendrowska, 1987) that allows individual testing of each attribute to support an attribute importance ranking. Other types of trees are the classification and regression trees (CART) and the scalable parallelizable induction of decision trees algorithm, also known as SPRINT.

2.5.2.2 Genetic algorithms

Genetic algorithms are derivative free, stochastic optimization methods based loosely on the concepts of natural selection and evolutionary processes. The basic idea of genetic algorithms was developed by a number of biologists that used computers to perform simulations of natural genetic systems. In general, genetic learning follows these steps. An initial population is created consisting of randomly generated rules. Each rule can be represented by a string of bits. As a simple example, suppose that samples in a given training set are described by two boolean attributes A_1 and A_2 , and that there are two classes, C_1 and C_2 . The rule *"IF A_1 AND NOT A_2 THEN C_2 "* can be encoded as the bit string "100", where the two leftmost bits represent attributes A_1 and A_2 , respectively, and the rightmost bit represents the class. Similarly, the rule *"IF NOT A_1 AND NOT A_2 THEN C_1 "* can be encoded as "001". If an attribute has k values where $k > 2$, then k bits may be used to encode the attribute's values. Classes can be encoded in a similar fashion.

Based on the notion of survival of the fittest, a new population is formed to consist of the fittest rules in the current population. Typically, the fitness of a rule is assessed by its classification accuracy on a set of training samples. The new population (or generation) is created by applying genetic operators such as crossover and mutation. In crossover, substrings from pairs of rules are swapped to form new pairs of rules. In mutation, randomly selected bits in a rule's string are inverted.

Genetic algorithms have been used in data mining for classification, clustering and generating association rules as well as other optimization problems. Other research areas include scheduling, robotics, economics, biology and pattern recognition. Genetic algorithms are popular because they do not depend on functional derivatives, they are easily parallelizable and are applicable to both continuous and discrete data. However,

they suffer some shortcomings that would significantly impact their selection for a similarity learning task:

- They are complicated to understand and to explain to end users, therefore it is difficult to do an error assessment when the model is not performing well.
- Crossover and mutation are hard to determine exactly how to perform them.
- Best fitness function is another difficult and challenging task.
- The coding of the problem often moves the algorithm to operate in a different space than the one of the problem.

For more information on genetic algorithms a comprehensive book is (Goldberg, 1989).

2.5.2.3 Case-based reasoning

Case-based reasoning methods do not measure similarity between cases numerically. Instead they form a model in memory of the relationships between examples. These relationships may either be induced (Kolodner, 1984; Lebowitz, 1987) or supplied by expert users. New examples are compared to stored ones by determining how closely they match these relationships. Case-based reasoning has been used extensively in the cognitive science community. Their underlying basis is that humans try to recall past cases when solving new problems. Therefore case-based reasoning is considered a plausible model for this process (Bareiss and Porter, 1988). Examples of this method include CYRUS (Kolodner, 1984), UNIMEM (Lebowitz, 1987) and PROTOS (Bareiss and Porter, 1988). The first two infer generalization hierarchies from examples. The third one maintains a complex set of user-supplied relationships that are continually refined as new examples are added. Even though we did not choose to follow a case-based reasoning approach it would be interesting in the future to investigate possible applications to similarity learning.

2.5.2.4 Regression

The prediction of output values can be modeled by a statistical technique called regression. The objective of regression analysis is to determine the best model that can relate the output variable to the various input variables. If we would like to formalize that then the regression analysis is the process of determining the dependency of the output Y to a set of inputs X . Y is usually called the response output or the dependent variable and X s are called inputs, regressors, explanatory variables or independent variables. Common reasons for using regression techniques include the following (Kantardzic, 2002):

- The inputs are less expensive to measure than the output.
- The values of the inputs are known before the output.
- Controlling of the input can predict behavior of outputs.
- Need for identification of casual links between some inputs and the output.

Linear regression with one input variable is the simplest form of regression. It models a random variable Y as a linear function of another random variable X . Mathematically the regression function is expressed as:

$$Y = a + bX \quad (2.1)$$

Parameters a , b are the regression coefficients and are usually calculated based on some given points of X, Y . A least squares solution takes place and tries to minimize the difference between the actual data points as given by the sample set and the calculated ones from the regression function.

Multiple linear regression takes place when the output variable is related to more than one input. In that case the regression function would be:

$$Y = a + b_1X_1 + b_2X_2 + \dots + b_nX_n \quad (2.2)$$

for n number of input variables. The solution follows the same least squares minimization process as before. However, the solution is going to be a (hyper) plane instead of a line.

Linear regression techniques, while easy to understand and implement they are not applicable to most of the complex data mining applications. They do not work well with non-numeric data. They also assume that the underlying relationship between input(s) and output is linear which of course might not be the case.

Non-linear regression takes place if the inputs in the regression function are modified by a nonlinear function (e.g. exponential). The regression function would be:

$$Y = a + f_1(X_1) + f_2(X_2) + \dots + f_n(X_n) \quad (2.3)$$

where f_i is the function being used to transform the predictor. Examples of these advanced regression models include polynomials functions and neural network techniques. The latter is the method of choice for our similarity learning algorithm. We treat our problem as a function-approximation one based on the training input provided by the user and we follow regression-based training methods. For a more detailed discussion on the selection of neural networks and other functions (large scale fuzzy membership ones) please see chapter 3 and chapter 5 where a direct comparison with our proposed method is presented. If the reader is interested in an in depth analysis of regression models a valuable starting book would be (Hastie et al., 2001).

2.6 Nearest neighbor in databases and machine learning

The existing methodology used to retrieve information from geospatial databases is the nearest neighbor (NN) one. In the following sections we explain how the NN operates and present popular distance functions used in the database community for information extraction. We also discuss some advanced NN methodologies in machine learning, the

Instance-Based (IB) family of algorithms. The main distinction between NN as applied in databases and IB is that the NN algorithm keeps all the examples in memory, since each example can be seen as a unique class. In IB approaches each class is described by more than one sample, therefore some generalization methods can be applied.

2.6.1 Nearest neighbor and K-nearest neighbor

The nearest neighbor method originated in statistics. It was first considered for rule production by Fix and Hodges (1951), who performed an initial analysis of the properties of k -nearest neighbor systems, and established the consistency of the method as k varies from one to infinity. They also numerically evaluated the performance of k -nearest neighbor for small samples, under the assumptions of normal distribution statistics (Fix and Hodges, 1952). It was subsequently adopted as a Bayesian approach to non-parametric classification for two-class problems (Johns, 1961), and has been widely used in the field of pattern recognition since 1963 (Kanal, 1963).

A nearest neighbor learner uses a metric that measures the distance between a new example and a set of exemplars in memory. The new example is then classified according to the class of its nearest neighbor. A pure nearest neighbor system stores all examples in memory verbatim, which is the case for database information retrieval. It then classifies new examples by finding the most similar case in memory and adopting its class. A distance function is used to determine similarity. For numeric attributes this is usually based on Euclidean distance, where each example is treated as a point in an n -dimensional feature space. It assumes that for a given point in the feature space the surrounding area will share the same class. The Euclidean function further assumes that

all features are equally important, and so share the same scale in feature space, and that this scale is linear along each axis.

Symbolic features are more problematic as they do not fit the Euclidean feature space model. To overcome this, similarity between symbolic features is determined by counting the matching features. This is a much weaker function as there may be several concepts based on entirely different features, all of which match the current example to the same degree. For domains containing a mixture of numeric and symbolic features the Euclidean distance function is adopted, with the distance between two symbolic values trivialized to zero if the features are the same and one if they are not. This mismatch between Euclidean feature space and symbolic features means that pure nearest neighbor systems usually perform better in numeric domains than in symbolic ones.

A successful variation of nearest neighbor is k -nearest neighbor (Kibler and Aha, 1987). This is an alternative method mostly developed to address noise. The most popular class of the k nearest examples is used for prediction. This prevents a single noisy example from incorrectly classifying the new one, and has met with much success. Note that this variation of the NN does not apply in information retrieval from databases, due to the fact that each example is considered a separate class as we discussed previously. A problem, however, is determining the value of k ; the more noise included in the input set, the larger k should be. As the system does not have this information *a priori*, a popular method is to use cross validation. The user trains and then tests the system using a variety of k values, and the one that produces the best result is subsequently adopted.

2.6.2 Nearest neighbor similarity functions used in databases

The approach described in this thesis can be seen as a nearest neighbor approach with the distance functions substituted by a complex neuro-fuzzy system. Therefore the following review focuses on existing NN techniques and their corresponding distance functions and is based on previous reviews (Martin, 1995; Kibler and Aha, 1987; Kantardzic, 2002).

The choice of distance function to calculate the NN can influence the bias of a learning algorithm. Bias is defined as a rule or method that causes an algorithm to choose one generalized output over another (Mitchell, 1980). In order for a learning algorithm to be able to generalize, a bias must exist. The problem is that there is no algorithm that would generalize better than others on all possible problems (Schaffer, 1994). Subsequently no distance function is going to be better than others in all applicable problems. This led to a variety of functions over the years, some more general, some others more application specific. The most popular ones are presented in table 2.2.

Distances are often normalized by dividing the distance for each attribute by the range (i.e. maximum - minimum). To avoid outliers it is also common to use standard deviation or to reduce the range by removing the highest and lowest percent of the data under consideration for defining the range. In addition to normalization, attribute weights (e.g. see quadratic distance metric) and other weighting schemes have been incorporated in the learning process (e.g. Wettschereck et al., 1995; Atkeson et al., 1997).

Additional distance functions include the context-similarity measure (Biberman, 1994), the contrast model (Tversky, 1977) and the hyper-rectangle distance functions (Salzberg, 1991; Domingos, 1995). Popular similarity measures for documents are the cosine measure, the Pearson correlation and the Jaccard similarity function. For nominal attributes the Value Difference Metric (VDM) was introduced by Stanfill and Waltz

(1986) and variants were developed later (Cost and Salzberg, 1993; Rachlin et al., 1994; Domingos, 1995). In (Wilson and Martinez, 1997) three more functions were introduced, namely the Heterogeneous VDM, the Interpolated VDM and the Windowed VDM.

Name	Equation
<i>Minkowsky</i> (Batchelor, 1978)	$D(x, y) = \left(\sum_{i=1}^m x_i - y_i ^r \right)^{1/r}$
<i>Euclidean</i> (Created for $r=2$ at Minkowsky function)	$D(x, y) = \sqrt{\left(\sum_{i=1}^m x_i - y_i ^2 \right)}$
<i>Manhattan or city-block</i> (Created for $r=1$ at Minkowsky function)	$D(x, y) = \left(\sum_{i=1}^m x_i - y_i \right)$
<i>Quadratic</i> (Created by adding a weight matrix to the Euclidean function)	$D(x, y) = (\bar{x} - \bar{y})^T Q (\bar{x} - \bar{y})$
<i>Mahalanobis</i> (Nadler and Smith, 1993)	$D(x, y) = (\det V)^{1/m} (\bar{x} - \bar{y})^T V^{-1} (\bar{x} - \bar{y})$
<i>Camberra</i>	$D(x, y) = \left(\sum_{i=1}^m \frac{ x_i - y_i }{ x_i + y_i } \right)$
<i>Chebyshev</i>	$D(x, y) = \max_{i=1}^m x_i - y_i $
<i>Correlation</i>	$D(x, y) = \frac{\sum_{i=1}^m (x_i - \bar{x}_i)(y_i - \bar{y}_i)}{\sqrt{\sum_{i=1}^m (x_i - \bar{x}_i)^2 \sum_{i=1}^m (y_i - \bar{y}_i)^2}}$
<i>Chi-square</i> (Diday, 1974)	$D(x, y) = \sum_{i=1}^m \left(\frac{1}{sum_i} \left(\frac{x_i}{size_x} - \frac{y_i}{size_y} \right)^2 \right)$

Table 2.2: Popular distance functions and their equations

The \bar{x}, \bar{y} are two input vectors, one from the stored instance and the other from the one waiting to be classified and m is the number of input variables. Matrices Q and V are of size $m \times m$ and they represent a problem-specific positive definite weight matrix and a

covariance matrix, respectively. Variables \bar{x}_i, \bar{y}_i are the average values for attribute i occurring in the training set. Variable sum_i is the sum of all values for attribute i occurring in the training set and $size_x$ is the sum of all values in the vector \vec{x} .

2.6.3 Nearest neighbor in machine learning (instance-based systems)

Nearest neighbor algorithms, also known as instance-based (IB) in machine learning literature, have as their goal to learn new concepts by storing past cases in such a way that new examples can be directly compared with them. On the basis of this comparison the system decides the class of the new example. Once again, we should emphasize here that the methods presented below go beyond the applicability to information extraction from databases due to multi-sample existence per class. It is interesting though to see the broader picture for NN methods and more specifically some drawbacks that are identified through this more detailed analysis that propagate to the typical NN approach. These drawbacks are further discussed in chapter 3, where we propose our novel model.

Nearest neighbor methods regained popularity after Kibler and Aha (1987) showed that simple nearest neighbor models could produce excellent results for a variety of domains. They tested three simple algorithms, named PROXIMITY, GROWTH, and SHRINK. All three used a normalized Euclidean distance function to classify each new example, with the class being decided according to that of the single nearest neighbor.

PROXIMITY is a pure nearest neighbor algorithm, retaining all examples and using an unweighted Euclidean distance function to perform classification. This system gave the best performance of the three. GROWTH accepts examples incrementally, and only stores those that the current exemplar database misclassifies. This reduces the number of examples stored by up to 80% with only a small reduction in classification

accuracy. SHRINK accepts all exemplars at first, and then weeds out those that the rest of the database classifies correctly. This algorithm produces impressive compression of the exemplar database (up to 80%), but at the expense of classification accuracy.

Kibler and Aha (1987) tested the above algorithms on several benchmark domains, and reported very good results for all of them. In particular, GROWTH produced excellent results for a relatively small final database. However, the results were misleading. A later study shows that the choice of domains for the initial study was fortuitous, and that in general performance of the three algorithms is much poorer than other classification methods (Aha, 1992). In comparisons with C4.5 (Quinlan, 1993), PROXIMITY performs quite well for four domains but very badly for another two, showing that the simple nearest neighbor approach has problems to overcome. A series of improvements was introduced in the algorithms IB1 to IB5, showing how the standard Euclidean distance metric is inadequate in many domains. The aim of the study was to overcome five objections to nearest neighbor systems (Brieman et al., 1984), namely that:

- they are expensive due to their large storage requirements;
- they are sensitive to the choice of similarity function;
- they cannot easily work with missing attribute values;
- they cannot easily work with nominal attributes;
- they do not yield concise summaries of concepts.

Below follows a brief discussion on these experimental systems (IB1 through IB5).

IB1: nearest neighbor. IB1 uses an Euclidean distance function that classifies according to the nearest neighboring example, saving all examples as they are introduced to it. The only variations from a pure nearest neighbor system such as PROXIMITY are

that attribute values are linearly normalized before examples are processed, and that missing values are handled by assuming that a missing feature is maximally different to that feature in all other examples. Like PROXIMITY, IB1 performs well in four out of six of the domains tested, and very poorly on the other two. These two are characterized by noisy values, missing values, and irrelevant features.

IB2: save only misclassified instances. IB2 differs from IB1 in that it saves only instances that it misclassifies. This reduces the number of exemplars required by storing only a single exemplar for each important region of feature space, and proves to be an effective way to prune the exemplar database. Accuracy decreases because early in the learning process important examples may be discarded because there were not enough examples of conflicting classes to accurately portray the differences between the new example and the nearest neighbor. As the number of stored exemplars increases, the accuracy of the model improves, and so the system makes fewer mistakes. There are problems though when the input data is noisy. Because the classification of noisy examples is poor, IB2 is more likely to store them, leading to an exemplar database where a disproportionate number of the examples contain noise. Aha (1992) observed that the performance of IB2 degraded more sharply with increased noise than IB1, and that the amount of noise in the exemplar database containing noise was higher than the percentage of noise in the input data. This confirms that IB2's method of choosing which examples to store leads to a bias towards noisy examples.

IB3: retain only good classifiers. Noisy exemplars will impact the performance of any system that does not detect them, because they will repeatedly misclassify new examples. IB3 overcomes this by pruning bad classifiers. It monitors the classification

performance of each exemplar to determine whether or not they should be used. A record is kept of the number of correct and incorrect classifications each exemplar makes. If the closest exemplar does not have an acceptable performance record, its statistics are updated but it is ignored in favor of the closest acceptable neighbor. IB3 bases this decision on the exemplar's performance relative to the observed frequency of its class. If an exemplar correctly classifies new examples to a significantly higher degree of accuracy than the observed frequency of its class, it is accepted for classification. If it classifies to a significantly lower degree, it is deleted from the database. This modification dramatically improves the performance of IB3, resulting in comparable performance to IB1 in most domains, and improvements in two. Both storage requirements and the amount of noise in the database are substantially reduced compared to IB1 and IB2.

IB4: weight attribute values. The Euclidean distance function works well for numeric domains where all attributes have similar relevance. In most domains this is not the case. The relevance of each attribute may be learned incrementally by dynamically updating feature weights. Aha (1992) proposes that these weights should be concept-specific, in that an attribute may be important to one class but not to the others. IB4 weights attributes dynamically, and performs much better than IB1, IB2 and IB3. In particular, the introduction of irrelevant attributes has very little effect on IB4 while for IB3 the exemplar database grows exponentially as the number of irrelevant attributes increases.

IB5: handle novel attributes. IB1 handles missing values by assuming maximal distance for an attribute if it is missing in either the new example or the exemplar being

tested. However, sometimes an attribute is missing because it is not relevant to the current example. Also, the value of an attribute may not be available when the first examples are being collected, but become so later. IB1 will incorrectly penalize those examples for which the attribute value was not available or not relevant. IB5 overcomes this problem by assuming that the distance between a missing and present attribute is zero. Therefore, sometimes-present attributes affect the distance function only when the value of that attribute is known for both examples. IB5 was tested using a domain that contained six boolean attributes, of which only one is relevant to classification. For the first 100 examples, the relevant attribute is missing, and so classification accuracy is approximately 50%. When the relevant attribute is added, IB5 quickly adapts to the new situation and classification accuracy improves. In contrast, IB4 reacts very slowly to the introduction of the new attribute.

2.7 Preference learning approaches

Since the goal of decision support systems is to assist users with making decision, it is especially important for them to accurately model the user preferences. The preferences of a user can be expressed in a variety of ways, either explicitly, for example in the form of preference statements, or implicitly through the way of acting in different situations. The problem of finding out about an individual's preferences, or about those of a group of individuals, is referred to as *preference elicitation*. This requires, among other things, models for the formal representation and methods for the (automatic) acquisition of preferences. Touching on various aspects of Artificial Intelligence, both theoretical and practical, preference elicitation is a recent and interesting research topic.

If no information is available at the start of interaction, preference elicitation techniques must attempt to collect as much information on user preferences as possible so that the systems can help users working toward their goals. Because user preferences are always incomplete initially, and tend to change in different contexts, in addition to user's cognitive and emotional limitations of information processing, preference elicitation methods must also be able to accommodate preference reversals, discover hidden preferences, and assist users making tradeoffs when confronting competing objectives.

Traditional elicitation methods include the *value* or *utility* functions (Chen and Pu, 2004). The *value* function reflects the preferences on a particular outcome (Keeney and Raiffa, 1976). In case of uncertain decision scenarios, where the outcomes are characterized by probabilities, a more complex function, *utility* function, is need to evaluate the "utility" of a decision. The *utility* function represents the user's attitudes about risk as well as the value of outcomes, so it induces a preference ordering on the probability distributions over the outcome space. The Analytic Hierarchy Process (AHP) (Satty, 1994) is an example of a decision support tool to solve multi-criteria decision problems. It uses a multi-level hierarchical structure of objectives, criteria, subcriteria, and alternatives.

Since the traditional methods are sometimes too time consuming and tedious, the computer aided decision support systems have appeared to simplify the task by making the assumption of additive preferential independence (mutually preferentially independence exists among attributes). Several representative systems were described, including:

- FindMe (Burke et al., 1997) that uses knowledge about users and products to provide advice to users about items they might wish to purchase or examine.
- Automated Travel Assistant (ATA) (Linden et al., 1997) is a recommender system that focuses on the problem of flight selection. In ATA, user preferences are described in terms of soft constraints on the values of attributes.
- The Apt Decision agent (Shearin and Lieberman, 2001) learns user preferences in the domain of rental apartments by observing user's critique of apartment features.
- The ExpertClerk (Shimazu, 2001) is an agent system imitating a human salesclerk. It interacts with shoppers in natural language and narrows down matching goods by asking effective questions.

In order to improve the accuracy of preferences elicited as well as save decision maker's effort, another research branch on preference elicitation has aimed at releasing all assumptions on preference structure by matching new user preferences to other users preference models (Chen and Pu, 2004). Typical research works are (Chajewska et al., 1998) and (Ha and Haddawy, 1998), where the concrete procedure can be summarized as follows (Carenini and Poole, 2002):

- Using "complete and reliable" elicitation techniques to elicit a sufficient number of users preference models.
- Grouping these models into qualitatively different clusters.
- Given the clusters, a new user's preference model is elicited by two sub-processes: find the cluster to which the new user more likely belongs and refine the preference model associated with that cluster for the new user.

The rationale is that finding and refining a matching cluster would require significantly less elicitation steps than building a preference model from scratch. The Video Advisor (Nguyen and Haddawy, 1999) is a representative system applying this methodology, which uses the case-based technique described in (Ha and Haddawy, 1998) to elicit the value function representing the user's long term preferences.

Collaborative filtering is based on an analogous idea, but the preferences matched are item ratings provided by different users. The recommender system will match these ratings against ratings submitted by all other users of the system, find the "most similar" users based on some criterion of similarity, and recommend items that similar users rated highly but the user has not rated (presumably not familiar with). The user can further rate the recommended items. Therefore, over time, the system can acquire an increasingly accurate representation of user preferences.

Examples of this methodology include the MovieLens (Rashid et al., 2002) that collects movie preferences from users and groups users with similar tastes. Based on the movie ratings expressed by all the users in a group, it attempts to predict for each individual his opinion on movies he has not yet seen. Other seminal collaborative filtering systems include GroupLens (Resnick et al., 1994), Bellcore Video Recommender (Hill et al., 1995) and Ringo (Shardanand and Maes, 1995). The systems vary in how they weighted the ratings of different users (i.e., determined who the similar users were and how close they were) and how they combine the ratings. There are also many applications of collaborative filtering on the web (Schafer et al., 2001).

2.8 Summary

In this chapter we presented literature review related to our work. We positioned our task within general data mining tasks. Subsequently, we focused on geographic data mining and discussed challenges within that field. Representative methods in machine learning followed, tied together with a discussion of appropriateness for similarity learning. We then focused on nearest neighbor techniques and variants and we discussed preference learning methods. This discussion serves as the basis for our system design motivation. Additional review material is reserved for the next chapter for direct comparison to our chosen approach.

Chapter 3

System design

In previous chapters we provided a definition of the problem at hand and the goals of our approach. In addition to that we presented literature related to the general framework of this thesis. This chapter introduces the architecture of our system that will be employed in the remaining chapters. Our system is used to prevail over the traditional, non-adaptable similarity models leading to a preference-based model that supports adaptable and context-specific information retrieval. We should note that the term “system” encapsulates both theoretical and applied investigation performed in this thesis. We provide an in-depth analysis of the problem and we investigate specific solutions proposed in the literature. We concentrate on neuro-fuzzy techniques due to their relevance to our system. Thereafter, we identify the desired characteristics of our model and we proceed to implementing them. The overall design including the corresponding information flow that will act as the basis for the chapters to follow is also shown.

3.1 Problem revisited

3.1.1 Nearest neighbor shortcomings

The nearest neighbor (NN) algorithm is intuitive and easy to understand. It has been successfully applied to a variety of real world problems. However, in its basic form the NN method has several weaknesses, with the most important ones presented below.

- *Dimensionality curse.* As the number of dimensions grows the data become so sparse that local neighborhoods are empty and non-empty neighborhoods are not local any more (Scott, 1992). Also nearly every point behaves as an outlier with respect to the rest of the training set and becomes closer to an outer boundary than to its next nearest neighbor (Friedman, 1995). Thus techniques and experience gained in low dimensions are hard to propagate in high-dimensional cases.
- *Irrelevant attributes influence.* With the introduction of irrelevant attributes in the dataset retrieval accuracy declines.
- *Large storage requirements.* The whole training set has to be stored in the model in order to obtain an answer.
- *Noise intolerance.* Accuracy decreases with the introduction of noise in the model.
- *Slow execution.* All the training instances must be searched to classify a new vector.
- *Distance functions selection.* The choice of appropriate functions to calculate the nearest neighbor to a query vector has been an important drawback. Some functions work well with some datasets but fail on others. Also, quantitative approaches are hard to extend to nominal data. Furthermore, the assumption that similarity between a training set and a query solely depends on their distance and not the actual values does not hold true in every domain.

- *User preference.* Once the distance functions are chosen they remain fixed throughout the process. There are not able to adapt to specific users/scenarios that might require fine-tuning of the function properties. Therefore similarity results are not adaptable to user needs.

Here we should mention that the above list is not exhaustive but representative of the past and current research challenges to improve the NN performance.

3.1.2 Issues addressed in this thesis

Some efforts have focused on one or more of the above problems without addressing them all in a comprehensive system. Others have presented solutions to some of the problems that were not as robust as those by others. In addition to that, techniques applied to one domain do not guarantee success in other domains.

From the perspective of this work the NN algorithm can be seen as a classifier, where every data belongs to its own class. This is not though the traditional classification task where the number of classes is significantly lower than the training sets. Therefore two drawbacks are not usually addressed in communication processes like ours, namely:

- The storage requirements cannot be avoided since the elimination of a training set would correspond to elimination of a class/possible answer to our problem as well. So all training sets have to be kept and be available.
- The problematic existence of noise does not apply in our approach because every class is represented by a single training set so the notion of noise does not exist. Noisy data would be related to imprecision issues resulting from the creation of the source metadata (e.g. temporal footprint). Imprecision issues are beyond the scope of this thesis.

The other issues are directly related to the formulation of a complete environment for intelligent geospatial information extraction. Dimensionality is an important issue that has attracted a lot of attention since storage limitations have decreased and databases are able to store more and more information. Higher dimensionality translates into higher number of possible dependencies between dimensions, and potentially different influence (attribute relevance) of each dimension to the overall similarity calculation. These are two issues that have to be addressed in the context of aggregating similarity results between dimensions. Our work concentrates on similarity assessment within each dimension therefore such investigation is not performed. Keep in mind though that in the future this analysis will be necessary for the next logical extension of our work, the similarity aggregation between dimensions.

A significant performance boost can be achieved by selecting an appropriate subset of the candidate objects instead of the whole set. This is a particularly active area in the database community often referred to as multi-dimensional indexing. These approaches can be grouped in two general categories, the space-partitioning and data-partitioning ones. In the first one the data space is divided in predefined subspaces regardless of data clusters. Some examples include the grid-file, the K-D-B-tree and the quadtree. In the latter category, index trees such as the B-trees, R-trees and their variants divide the data space according to the distribution of data objects inserted or loaded into the tree. A variety of surveys and papers have been published on indexing methods. A comparative study can be found in (Weber et al., 1998) and up-to-date references in (Korn et al., 2001). Indexing issues are not examined in our work since we propose enhancements on retrieval accuracy rather than retrieval speed.

This thesis focuses on the last two drawbacks of the traditional NN by aiming at the improvement of distance functions through incorporation of user preference in the query process. Thus we substitute these functions with a machine learning system that adapts to user/problem requirements, therefore achieving higher retrieval accuracy.

3.2 Machine learning using neural networks and fuzzy logic

Several machine learning methods were presented in chapter 2. We intentionally omitted discussion of neuro-fuzzy methods. We chose to present them here since our system falls into that category and some benefits of our system structure are tied together with the underlying general methodologies.

Over the last few decades neural networks and fuzzy systems have established their reputation as successful approaches to information processing (Klose et al., 2001). They both offer certain advantages especially when vague data or prior knowledge is involved in the process. Neural networks incorporate learning capabilities in their process. They also provide the developed system with “memory”, which allows the system to store the results of learning. On the other hand fuzzy inference systems can provide a structured knowledge representation in the form of if-then rules. They are easy to interpret and analyze.

However, each of these two methods suffers from several weaknesses. To overcome such limitations several systems were proposed where both models complement each other. These so called neuro-fuzzy systems address some of the individual weaknesses and offer some appealing features. In the following sections we discuss briefly each method, present the reasoning behind the combination of them and show some applicable examples.

3.2.1 Neural networks

Artificial neural networks also known as connectionist models are systems that try to mimic the organization of the human brain. Initially, research in this area was driven by neurobiological interests. Modern interest from the data mining perspective considers the development of architectures and learning algorithms that will be applied in information processing tasks.

In the literature, a large amount of information exists on the network types, learning methodologies and applications of neural networks. Good starting books would be (Haykin, 1994; Bishop, 1995). At this stage we provide a brief description of the most popular architecture, the feedforward multilayer one. A more detailed examination can be found in chapter 5. Neural networks consist of a number of independent, simple processors called neurons. Neurons communicate with each other through weighted connections, the synaptic weights. After the design of the network is chosen a dataset is presented to the network and the learning process begins. The goal is to optimize the network behavior by adjusting the weights appropriately so the network output is close to the expected output. In other words the network creates a mapping of the input data to the desired output using the presented examples.

The most important advantage of neural networks is that they are universal approximators, meaning they have the ability to approximate any arbitrary function (Haykin, 1994). They do not need a mathematical model describing the problem and no prior knowledge is necessary. On the other hand neural networks are black boxes, meaning they are hard to interpret in terms of rules. Also prior knowledge is hard to incorporate in the process because they usually learn from scratch. The learning process can take a long time and the success is not guaranteed. The author's experience suggests

that this type of network might work where every other technique fails, but unfortunately it could also fail where most techniques would succeed. And even if the modeling is successful, it is really hard to figure out why it worked.

3.2.2 Fuzzy methods

Fuzzy set theory supports a large number of applications and has become a popular method for dealing with complexity, uncertainty and imprecision in various systems. Fuzzy sets arise from an extension of the classical (Boolean) sets for representing concepts that exhibit a gradual transition from membership to non-membership. There is a large number of concepts in which an element can have partial membership in a set. Fuzzy (and crisp) sets may be represented by a mathematical formulation known as the membership function. This function gives a degree or grade of membership within a fuzzy set. Interpretations of membership degrees include similarity, preference and uncertainty (Dubois et al., 1996). They can show how similar an object is to a prototype, they can indicate preferences between sub optimal solutions to a problem or they can model uncertainty by using imprecise terms.

Based on the fuzzy sets theory fuzzy inference systems are created using if-then rules and fuzzy reasoning. Their goal is to derive conclusions from a given set of fuzzy rules. A typical system is composed of five functional blocks (fig. 3.1). The input is transformed from crisp outputs to degrees of match in the Fuzzification block. Then it propagates in the Decision Making module where the inference operations take place. The decision is based on rules from the Rule Base and membership functions retrieved from the Function Database. In the last stage Defuzzification transforms the fuzzy results of the inference into a crisp output. A more detailed example of this fuzzification-

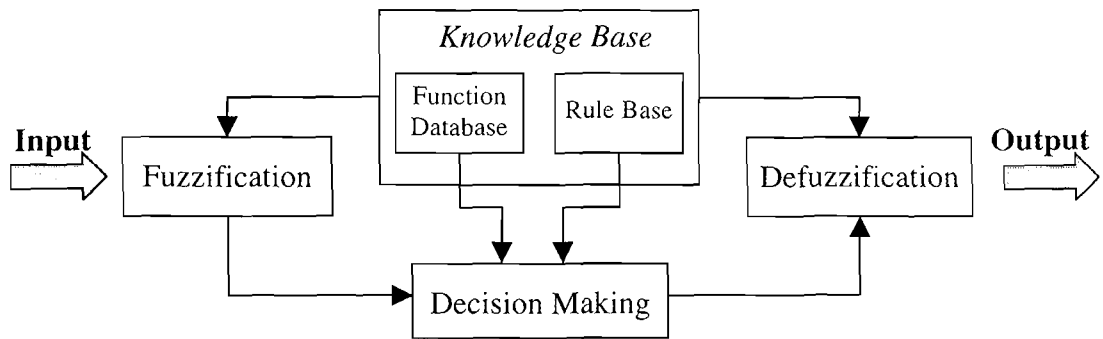


Figure 3.1: Fuzzy inference system

defuzzification process can be found in chapter 7, where we explain the training of our algorithm. There is extended literature on fuzzy sets applications; close to our task we can mention visual retrieval systems (Santini and Jain, 1999) and fuzzy integrals for similarity approximation (Ishii and Wang, 1998).

3.2.3 Neuro-fuzzy techniques

The description provided in the previous sections of neural networks and fuzzy methods can lead to an intuitive combination of the approaches. The fuzzy system can be used to represent knowledge and the neural network techniques can provide a learning mechanism to determine membership values. The drawbacks of both individual methods, the black box behavior and the non-adaptable membership functions, could thus be avoided. The combination can constitute an interpretable model, with learning and prior knowledge incorporation capabilities (Klose et al., 2001). Therefore, neuro-fuzzy methods are especially suited for applications, where user interaction in model design or interpretation is desired.

Most of the existing neuro-fuzzy models are motivated by fuzzy control systems. In fuzzy control the main idea is to build a model of human control expert that will control the system without thinking in terms of a mathematical model. The control

actions should rather be specified in a linguistic form of rules. However, due to uncertainties in specifying the fuzzy controllers, a manual tuning is often necessary to overcome initial design errors. The role of a neural component in this case would be to automate this tuning process.

In the data analysis field there is also a number of neuro-fuzzy techniques. However, there are some differences in the characteristics of the problem. The learning can be done off-line as generally the data are available in a database. Efficient learning from scratch is more frequent than in control applications. Also the interpretability of the resulting rule base is often more important than in control where a working controller is often satisfactory enough. Nonetheless, the underlying motivation to combine human accessible fuzzy rule approach and learning capabilities of neural networks still exists in every neuro-fuzzy approach.

3.2.3.1 Neuro-fuzzy models

The term neuro-fuzzy systems is often used to describe all kinds of combinations of fuzzy systems and neural networks. In order to make it more specific we present the categorization presented in (Nauck et al., 1997), where the following categories are introduced:

Fuzzy neural networks: In some cases fuzzy methods have been used to enhance learning capabilities or performance of a neural network. Examples in this category can be found in fuzzy rule based adaptation of the learning rate (Halgamuge et al., 1994) or by fuzzy additions to allow support for fuzzy inputs (Ishibuchi et al., 1995; Narazaki and Ralescu, 1991). These approaches should not be confused with neuro-fuzzy ones, at least in their narrow sense.

Concurrent "neural/fuzzy systems": A neural network and a fuzzy system work together on the same task but without influencing each other, in other words neither system is used to determine the parameters of the other. A frequent example is the application of a neural network in order to pre-process the inputs or post-process the outputs. These kinds of models are not strictly speaking either real neuro-fuzzy approaches or fuzzy neural networks.

Cooperative neuro-fuzzy models: A simple form of neuro-fuzzy systems, this term is used when a neural network is adopted to determine the parameters of a fuzzy system. The neural network is involved only in the training of the fuzzy system. After the training is done the fuzzy system works independently. Cooperative models can be further divided into approaches that learn fuzzy sets offline, fuzzy rules offline, fuzzy sets online and rule weights. The last one is quite popular in commercial fuzzy development tools.

Hybrid neuro-fuzzy models: This architecture describes a homogenous blend of a neural network and a fuzzy system. Depending on the characteristics of the design the system can be seen as a special neural network with fuzzy parameters or as a fuzzy system implemented in a parallel distributed form.

The majority of neuro-fuzzy approaches fall in the hybrid category. Especially in data analysis, this architecture is predominant due to the characteristics of the relevant problems. Our proposed neuro-fuzzy system falls in the hybrid category as well. The fuzzy functions are blended together with the neural network and even if training happens independently at some stages the final training stage readjusts parameters from both the fuzzy functions and the neural network.

3.2.3.2 Mapping fuzzy rules to networks

In order to convert the fuzzy rules to a neural network and vice versa a mapping scheme should be developed between the rule elements and the network elements. The usual approach is a feed-forward network with at least three layers. The first layer represents the inputs from the domain attributes. The second layer represents the fuzzy rules. It is the hidden layer and each node corresponds to a specific rule. The antecedents of the rules are modeled as connections from the input to the hidden (rule) layer, the consequents as connections from the rule layer to the output one. Depending on the model the membership functions are represented as either fuzzy valued weights or as additional layers with special activation functions and parameterized weights.

Assuming the above structure, the propagation of a stimulus would be as follows (Klose et al., 2001). The input values would be selected according to the tuple values. Then the membership values of the fuzzy sets would be calculated. This can be done either in the hidden layer or by applying fuzzy weights. The membership values are then introduced in the rule layer. The rule nodes combine their participating inputs to activate the rules. This can be seen as conjunction or disjunction of the antecedents. Finally in the output layer an aggregation of the corresponding rules for each class takes place and the highest activation unit provides the result (winner-takes-all method).

3.2.3.3 Learning process

The learning process can be divided into two major tasks. The first one involves the choice of the structure and the second the optimization of the given structure. The choice of the structure reflects the rules we want to impose on the system. This has been a challenging task for neural networks and a lot of work remains to be done. In our work

we do not address this issue, namely to find heuristic solutions to modify the structure, so further discussion is omitted.

What we focus on, as most of the learning algorithms for neuro-fuzzy systems do, is the optimization of the membership functions. The membership functions can be easily described by the parameters of their mathematical formulation. In this case optimization would involve calculation of these parameters with respect to a global error measure.

There are some issues involved in the learning phase. First of all, the model can be only as good as the membership functions used. For example, if we would try to interpolate a non-linear problem with a linear approach, the degree of adaptation freedom of the function would not allow us to model the problem in an appropriate manner. Therefore the mathematical formulation of the membership functions should be chosen with caution based on the specific rule requirements.

Another problem comes from the fact that a standard fuzzy system uses non-differentiable functions (like min, max) in the inference process. Differentiation though is a prerequisite for the gradient descent learning methods in neural networks like error backpropagation. So these functions would need to be replaced by differentiable functions with similar properties.

In addition to the above challenges, the traditional learning techniques do not take under consideration the semantics of the underlying fuzzy system. Suitable constraints should be imposed to assure that certain properties would remain and therefore their semantic meaning is kept throughout the learning and consequently the implementation process. Further discussion and an overview of current approaches can be found in (Lin and Lee, 1996; Nauck et al., 1997; Klose et al., 2001; Liu and Miyamoto, 2000;

Tettamanzi and Tomassini, 2001). Neuro-fuzzy approaches directly relevant to our system design and functionality will be presented later in this chapter after we portray our own system to facilitate comparisons.

3.3 System design

In this section we begin with some assumptions for our system applicability. Then we present the proposed system architecture and identify some important characteristics of it. Thus, we compare it with other neuro-fuzzy systems and directly related methodologies to distinguish our approach from them.

3.3.1 Applicability assumptions

In the following section we discuss assumptions related to the use of our method.

3.3.1.1 Class-based vs. instance-based similarity

Each information object stored in a database is described by a set of attributes. The *class* of an attribute is a template definition of the variables for a particular kind of object. Thus, an attribute *instance* is a specific value of a class; it contains real values consistent with the class definition. Similarity assessment can be performed in these two levels, the *class* and the *instance* level. Attribute class level similarity assessment provides an evaluation of the degree to which two different classes resemble each other semantically. For example if a user queries on “Image Scale” and the database has an attribute class described as “Ground Pixel Size”, the question to be addressed is whether these two terms are similar, and if so by how much. Similarity assessment at the class level is not addressed in this thesis.

Attribute instance similarity assessment aims at the evaluation of the degree to which two different values of the same attribute are similar. We assume that there is a

one-to-one mapping relation at the class level of the query attribute and the corresponding database attribute class. This is the similarity level we concentrate on.

3.3.1.2 Attributes under consideration

Before we proceed and describe the architecture of our similarity learning algorithm, we should identify the kind of attributes that our algorithm supports. In general, a geospatial information object can be represented by quantitative and qualitative attributes (fig. 3.2). For example “Time” is a quantitative attribute as opposed to “Owner Name” that is a qualitative one. This work focuses on quantitative attributes, therefore qualitative attributes are not analyzed any further. Some examples of qualitative similarity functions can be found in (Wilson and Martinez, 1997). A very active field of research addressing a similar problem is the one of document retrieval. Correlation between terms is established through a combination of thesaurus and ontologies.

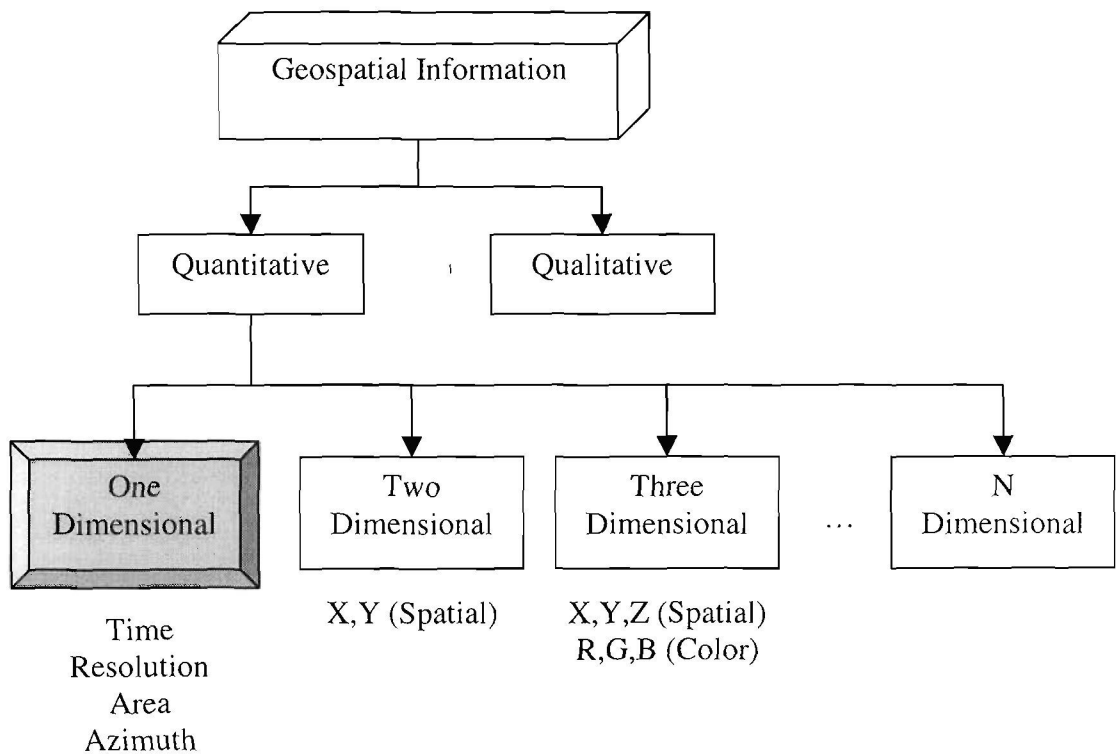


Figure 3.2: Geospatial attributes organization

Quantitative attributes can be further classified based on the number of dimensions that represent each attribute. For example, “Time” is represented by a single dimension, in contrast to “Space” that is composed of a set of 2 (X,Y) or 3 (X,Y,Z) potentially highly dependent dimensions. The algorithm focuses on one-dimensional, quantitative attributes. Attributes with more than one dimension are not examined. For similarity in space (shape similarity) a good overview is presented in (Veltkamp and Hagedoorn, 2001). Similarity in other correlated dimensions such as color has attracted attention from computer and cognitive scientists, with an overview of many color spaces, their definition and usability given in Chapter 1 of (Plataniotis and Venetsanopoulos, 2000).

It should also be mentioned that we do not aggregate similarity metrics from each attribute to produce a total similarity metric. This is left for future work, which is the next logical extension of this thesis. In addition to that, we deal with single objects and not scenes, where multiple objects exist and possibly interact with each other.

3.3.1.3 Database design independence

An important characteristic of our learning system (as most of database learning systems) is that it is independent of the chosen database design. In order to assess similarity, objects can be stored in any structured information format, where specific attribute values for each object can be easily extracted. Therefore our algorithm supports Object-Oriented designs (e.g. XML) or Relational Database designs. In addition to that, the user query can be in any language and format as long as the required “translator” exists so that the attribute values for the requested object can be extracted and a one-to-one mapping can be produced between the query request and the database value.

3.3.2 General framework

The general system framework is described in figure 3.3. The inputs to the process are two multi-dimensional vectors, one representing the request to the database (query vector) and another expressing an existing object in the database (database vector). These two vectors are compared to each other within the system and the output shows the degree of similarity between them. As we mentioned before, we assume that a one-to-one mapping can be established between the query and the database vector, in other words the classes of the corresponding attributes are considered identical. Also a common measurement system and scale is assumed for instance values in each vector.

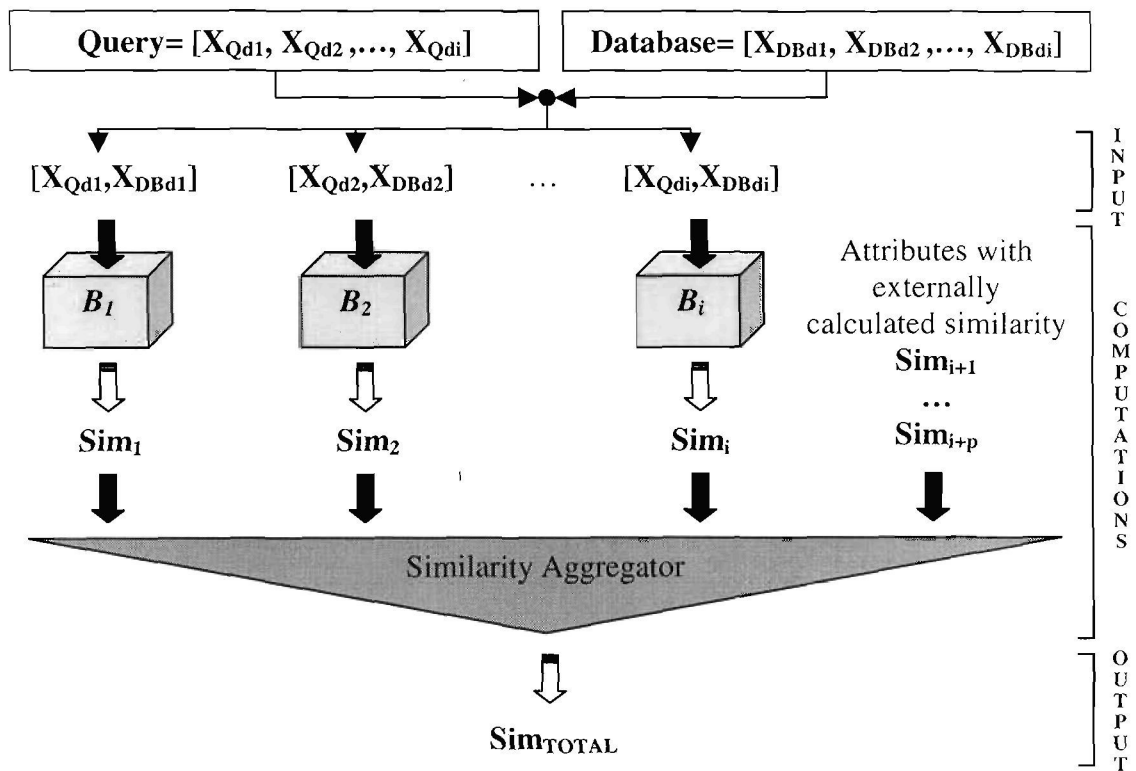


Figure 3.3: General similarity framework

The information flow is as follows. First each attribute from the query vector is paired with the corresponding attribute from the database vector. Then for each pair a

In the second step, we capture highly localized deviations from this (statistically) average behavior that could not be represented by the global function. For these cases we developed a custom multi-scale RBF network (MSRBF). The training sample of our network contains the similarity error vectors as obtained from the FMFs training.

After the training is complete the module will have the structure of figure 3.5 that acts as a guide for the remainder of the thesis. This feedforward neuro-fuzzy network will calculate the similarity metric based on a cumulative contribution from the fuzzy function and the neural network.

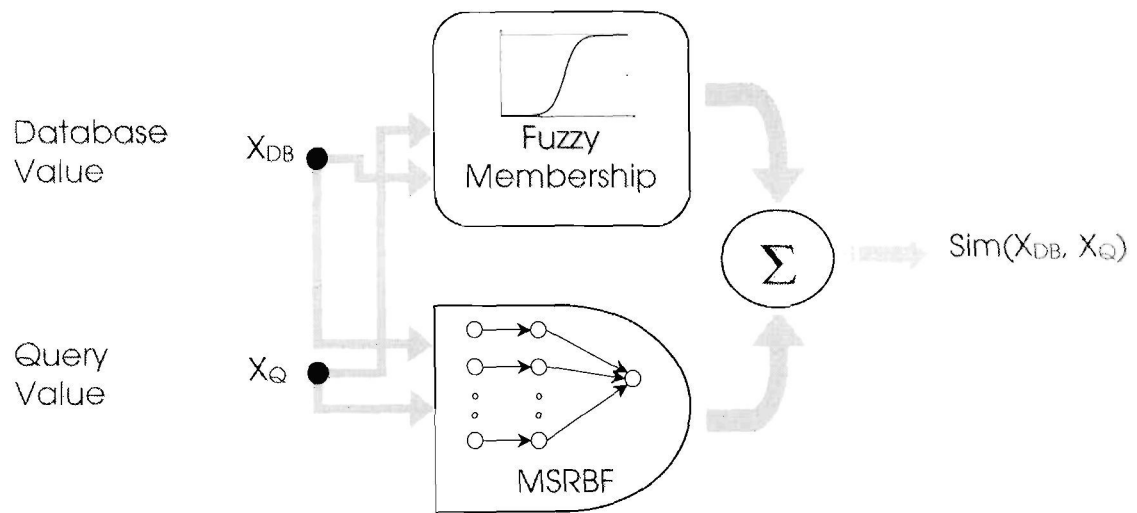


Figure 3.5: System architecture

3.4 Characteristics and uniqueness

Based on the architecture presented previously we can identify some system characteristics. These specifics constitute our novel approach as a competitive solution in the theoretical aspect that involves the design process. A comprehensive evaluation through functionality examples and statistical simulations is presented in chapter 7.

Learning adaptability. The nearest neighbor techniques do not incorporate learning techniques in most of their applications. They usually utilize distance-based similarity

functions that cannot compensate either for non-linearity or for more complex cases where a distance is not a representative metric for comparison within dimensions. In our approach, by expressing similarity through fuzzy functions we support asymmetrical, non-linear similarity metrics that express user preference, in other words they adapt to the specifications of the problem, and do not remain constant. In addition to that, they are not distance dependent but they are able to identify that dependency and model it if necessary.

Modular design – independent training. The system might be composed by a large number of processing nodes, but there is a conceptual and practical organization in the system design. This offers a significant advantage, the easy assessment of the contribution of every node. In doing so, nodes can be trained easier, error can be captured and interpreted more successfully, and the overall transparency of our system makes it intuitive for non-expert users.

Expected and unexpected similarity modeling. We identify two kinds of similarity behavior, expected and unexpected. This is a major contribution of our system since this separation is not supported in current similarity algorithms. This distinction is supported by the choice of a neuro-fuzzy architecture.

The fuzzy part, as expressed through the fuzzy similarity functions, has the ability to adapt to some “expected” similarity rules (e.g. exponentially monotonically decreasing behavior the further away from the target value) as investigated by cognitive scientists. The rules are incorporated in the method and are adaptable through a back-propagation training that adjusts the fuzzy functions expressing these rules to the given training set. This way our system incorporates prior knowledge in the process.

The neural network acts as an error correction function. It is only triggered when “unexpected” behavior is identified. Since no explicit general behavior rules apply in such a case, the choice of a neural network for modeling something “unknown” comes naturally. A considerable drawback of some neuro-fuzzy architectures is their inability to control the influence of the neural network to the overall input space. To compensate for that, as we will show in chapter 5, a novel multi-scale radial basis function network is developed. By doing so the fuzzy functions act as global approximators of the similarity signal and the neural part is restricted to controlled localized areas.

3.5 Comparable work

Now that the design of our system is presented we proceed to examine theoretical support and comparable work in the literature. To the best of our knowledge no method exists that supports similarity learning within attributes in geographic object queries, where the objects are presented through a feature vector that does not describe the actual content of the object. On the other hand, methodologies for similarity learning that query the content of objects, especially when these are images, exist in a disproportional ratio.

Probably one of the most representative works in content-based retrieval is the one in (Ma and Manjunath, 1996). A learning based approach is presented to retrieve similar image patterns (textures) from aerial photographs. They use self-organizing maps to achieve coarse labeling and learning vector quantization to fine-tune their process. A variety of other techniques exists, techniques that lately have gathered a lot of attention due to their application in security and surveillance, especially those addressing face recognition issues.

3.5.1 Applications of neural networks and fuzzy logic in similarity learning

Our approach is based on a neuro-fuzzy network, therefore we should first discuss applications of neural networks and fuzzy logic specifically for similarity learning. Later we will expand to neuro-fuzzy systems where a larger variety of applications will be introduced.

Neural networks have been used effectively in content-based retrieval systems (e.g. Carkacioglu and Vural, 2002; Lim et al., 2001). Another example would be the NeuroRule data mining system (Lu et al., 1995) where several classification problems are solved using neural methods. In categorical perception the goal is to find how similar things look depending on whether they are in the same or different categories. An example of backpropagation network applied for these purposes is (Tijsseling and Harnad, 1997). Also another active field that has drawn a lot of attention as the Internet has exploded is information retrieval (IR). The name is much more general than the actual content of the databases under investigation. Similarity is calculated in textual databases (e.g. databases containing documents). Neural networks have been successfully applied there as well (e.g. Rumelhart and Todd, 1993; Mandl, 2000). For an extended review the reader is referred to (Chen, 1995).

Fuzzy methods have also been employed for similarity learning in databases. A fuzzy integral has been proposed in (Wang and Ishii, 1997) to act as a non-linear similarity function that will later be incorporated in a genetic algorithm. In (Klawonn and Keller, 1997) the authors have provided a modified version of the fuzzy-c means algorithm to substitute for Euclidean distance. It has higher modeling capabilities but it is still a distance-based similarity algorithm. Montesi and Trombetta (1999) have developed a fuzzy relational algebra to model queries in multimedia and web applications and

provide a framework to study similarity issues. In (Santini and Jain, 1999) a similarity measure was presented based on fuzzy logic. Their model is based on the Feature Contrast model (Tversky, 1977) from the psychology domain and is mostly concerned with identifying dependencies among properties. In the IR field an overview on fuzzy methods and neural network applications can be found in (Crestani and Pasi, 1999).

3.5.2 Neuro-fuzzy architectures

In section 3.2.3 we discussed the underlying idea behind a neuro-fuzzy (NF) approach, namely to combine the best of both techniques (neural and fuzzy). Here we present some popular neuro-fuzzy architectures as developed in the last decade.

FuNe network. This NF network is based on an architecture of five layers. The first layer consists of the inputs. The second one contains sigmoidal functions expressing membership. In the next layer specialized units exist to represent fuzzy sets and the forth layer consists of the units that express the fuzzy rules. The last layer is the output layer. This kind of network is special mostly because it supports rules with only one variable as antecedent. More information can be found in (Halgamuge and Glesner, 1994).

Sugeno-type. In this case the NF system simulates a Sugeno-type system of weighted rules. It can be interpreted as a special RBF network. The only difference would be the expansion of activation functions from gaussians to logistic ones. This network was employed to predict the German DAX stock index (Siekmann et al., 1997).

NEFCLASS. For the NEFCLASS NF network (Nauck and Kruse, 1995) the main characteristic is the incorporation of linguistic rules in the hidden layer. The membership functions are represented by fuzzy valued weights on the connections between the input and the hidden layer. Instead of a summation of all inputs to create the output of each

hidden node the membership values are kept. An application of this network was on object reconstruction from lines previously extracted from remotely sensed imagery.

ANFIS. ANFIS stands for Adaptive Network-based Fuzzy Inference System and it is presented in detail in (Jang, 1993). This network can also be seen as a Sugeno-type fuzzy system with learning capabilities. All functions should be differentiable to accommodate the backpropagation learning algorithm. Its learning is based on a combination of gradient descent and least squares. ANFIS has been applied successfully in non-linear function modeling and time series prediction.

Fuzzy ARTMAP. This network is based on a combination of modules called Fuzzy ART (Adaptive Resonance Theory). It is a self-organizing neural network capable of clustering collections of arbitrarily complex patterns via unsupervised learning (Carpenter et al., 1991). Approaches of Fuzzy ART have been used for autonomous robot guidance and navigation.

Fuzzy Gated neural networks. In (Chandrasekaran et al., 1995) another approach to neuro-fuzzy computing was proposed. It is a topology-constraint free feature map with a controlled input-output “gate”. It requires only one epoch for learning. This property makes it suitable for real-time applications where low levels of noise exist. The authors have demonstrated its applicability on a pattern recognition example.

None of the above mentioned NF architectures has been applied to similarity learning to the best of our knowledge. We chose to develop our own system to facilitate the specific requirements of our problem and to have a better understanding of the underlying complexity.

3.5.3 Explicit related work

A NF network built for similarity learning is presented in (Mitaim and Kosko, 1997). Although they calculate similarity in content-based applications (images) their general approach is the same: they use a neuro-fuzzy system to create a profile of user preferences based on a relevance feedback training. This profile should be based on specific rules that will be fine-tuned by the training process. After training the profile can be called by an agent so query results will be adaptable to specific user preference. Our general approach might look similar to theirs, but in reality the two architectures are significantly different to accommodate different specifications of similarity.

Another application similar to ours is shown in (Frayman et al., 1999). Their dynamically constructed fuzzy neural networks are used to correct the problem of small disjuncts in decision trees. The problem of small disjuncts relates to the fact that rules which cover very small portion of the training set can cause large classification errors. While these rules might represent individually only a small portion of the input set, they may collectively account for a much higher percentage of errors (Clarke and Niblett, 1987). Our system design addresses this issue with the introduction of a multi-scale RBF network to absorb this localized unexpected behavior. Additional work on small disjuncts can be found in (Weiss, and Hirsh, 2000).

From the cognitive science point of view our model supports similarity behavior as identified through human experiments. In (Shepard, 1987) it was shown that similarity decreases exponentially as we get further away from the target. Therefore previously performed human testing supports the choice of sigmoidal functions as fuzzy memberships. This realization is also discussed in (Santini and Jain, 1997).

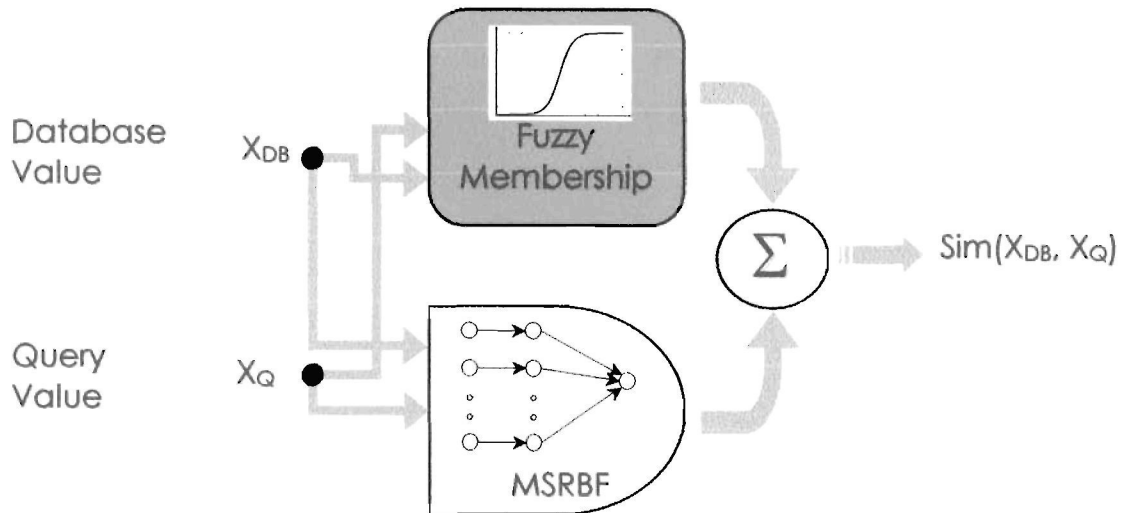
In other related work to our research, the user can examine (Griffiths and Bridge, 1997) for a nice discussion on fixed and adaptive similarity functions. Local use of weights can be found in (Howe and Cardie, 1997) and a good review on local versus global weights in (Wettschereck et al., 1995). An example of error correction technique for classification improvement is shown in (Dietterich and Bakiri, 1995).

3.6 Summary

The focus of this chapter was to provide an overview of the system that we designed for similarity learning. In order to justify the proposed architecture, an overview of nearest neighbor weaknesses was presented. We built our system to address some of these issues. Neuro-fuzzy methods were discussed in depth and our system design was introduced. Some significant characteristics of our system are presented, namely its learning adaptability and modular design, and ability to distinguish expected from unexpected similarity behavior and model them in separate processes. Works similar to ours are outlined after the system is established to facilitate comparisons.

Chapter 4

Preference learning in one-dimensional attributes using fuzzy functions of adaptable complexity



In the next chapters we will examine how our system learns preference behavior within each dimension. Our method follows conceptually a wavelet paradigm. A variety of mathematical functions are used corresponding to different frequencies with different operational range on the input space. By doing so we generate two families of similarity functions, the *global* and the *localized* ones.

Global functions attempt to model the similarity signal throughout the input space. They can also be presented as the mother-type functions introduced in the wavelets literature. Their application range is defined by the limits of the input space. These functions increase gradually in complexity until a specified goal (e.g. accepted error) is

reached. In order to do so we make use of fuzzy membership functions of adaptable complexity, which is the focus of this chapter.

While the global functions attempt to model a fairly expected preference behavior, there are cases where user-feedback will divert locally due to personal preferences and/or application specifics. To model this unexpected local deviations a second type of functions is introduced, namely the *localized functions*. This family of functions is used to capture the high frequency components of the similarity preference signal. These functions are applied on the whole input space but their active/operational segments are only a subset of this space. They are represented by a customized multi-scale radial basis neural network, which is the subject of chapter 5.

4.1 Approach overview

In this section we present a short description of the problem and we discuss the process flow of the developed system using fuzzy functions.

4.1.1 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 geospatial collections. Users that access these collections have diversified information 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 preferences. Similarity parameters should not be predetermined but rather adaptive to different scenarios and requirements even for the same dataset collections and/or users.

Relational operations like equality or inequality have been used in the past, but in complex database applications a similarity matching approach is incorporated. In order to perform the similarity match efficiently an information object O stored in a database is compared to a query request O^q using their corresponding *database attribute values* f_{ik} and the *query value* request f_{ik}^q . A database object O is compared to a query description O^q by using matching functions to produce a similarity metric S as follows:

$$S = g \left[\begin{array}{c} h_1 \left(t_{11}(f_{11}, f_{11}^q), t_{12}(f_{12}, f_{12}^q), \dots, t_{1p}(f_{1p}, f_{1p}^q) \right), \\ \dots \\ h_i \left(t_{i1}(f_{i1}, f_{i1}^q), t_{i2}(f_{i2}, f_{i2}^q), \dots, t_{ik}(f_{ik}, f_{ik}^q) \right), \\ \dots \\ h_m \left(t_{m1}(f_{m1}, f_{m1}^q), t_{m2}(f_{m2}, f_{m2}^q), \dots, t_{mj}(f_{mj}, f_{mj}^q) \right) \end{array} \right] \quad (4.1)$$

In the above equation function t_{ik} expresses the similarity between each attribute, h_i combines similarity results from each attribute to provide a metric for each conceptual attribute grouping and g is the overall similarity measure (see section 1.1.2 for more details). The focus of this work is function t_{ik} . It is often described by a Euclidean distance (difference). Functions h_i and g have so far received more attention in the literature and numerous models have been proposed through the use of complex non-linear functions. However, if function t_{ik} fails to describe the corresponding similarity relationships adequately, its errors propagate in the overall solution making it hard, if not impossible, for the aggregation functions h_i and g to correct this.

A common example of such similarity preference in GIS is when asymmetric, non-linear user behavior is exhibited during the direct comparison of attributes. 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 with 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 user preference within each attribute comparison (function t_{ik}).

4.1.2 Process flow

In order to adapt similarity models 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 preference metric for each pair. The corresponding training dataset is created and used as input for our preference learning method. An example of this dataset creation is described in chapter 7.

For our training we make use of several preference models as expressed through a variety of fuzzy membership functions (FMFs). Our approach is simple yet effective: gradually increase the complexity of the underlying FMF until an acceptable solution is reached. We begin the process by interpolating a set of planes to the training dataset (fig. 4.1). We examine the resulting accuracy and if it is within the predefined specifications we end the process. These predefined specifications are in essence thresholds describing the maximum acceptable error between the interpolated functions and the training points. They can be preset by the database designer or adjusted in real-time by the user. If the results are not within these thresholds, we examine the obtained plane parameters. This analysis leads to a decision whether similarity is dependent on the query value, their

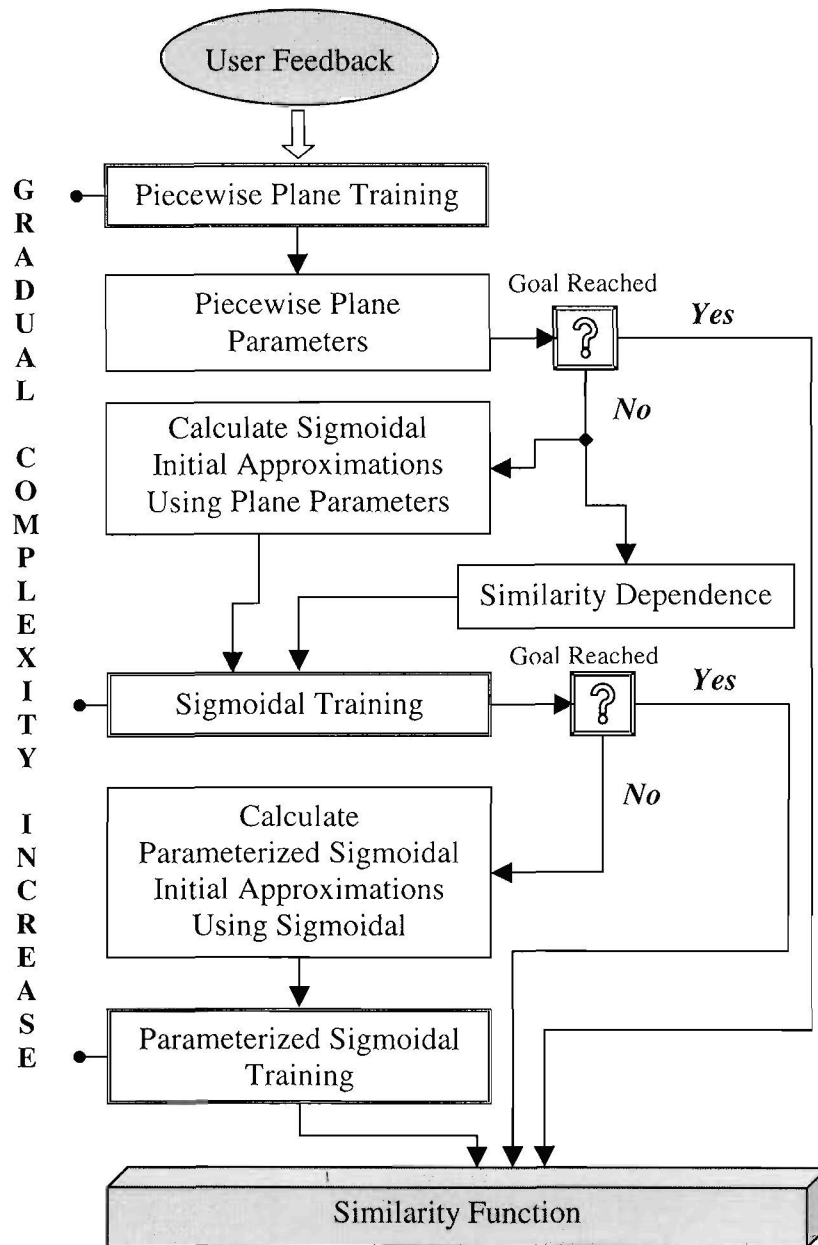


Figure 4.1: Fuzzy functions training flow

difference metric or the actual database and query values. We continue by interpolating two sigmoidal functions whose initial approximations are calculated from the plane properties. If required accuracy is not achieved, we provide further modeling capabilities by parameterizing further the FMFs parameters. At the last stage we obtain the best possible set of FMFs that express user preference as presented through the training set.

One common characteristic of our FMFs is that they take into consideration *asymmetric* similarity behavior that might exist. Asymmetry refers to different preference expression when two candidate values are equally away from the target value but from different sides (i.e. \pm same distance). For simplicity we examine one side in the rest of the chapter but the same process is applied for the other side (half) as well. In the next sections we introduce the training of different functions, how one acts as a basis for the next more complex one and their corresponding modeling capabilities.

4.2 Piecewise planar similarity function

4.2.1 Mathematical formulation

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

$$\mathbf{Sim}_{\text{Planar}}: \mathfrak{R}^2 \rightarrow [0,1] \quad (4.2)$$

The function inputs are query and database values $[X_Q, X_{DB}]$. 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 of the other one. The similarity function $\mathbf{Sim}_{\text{Planar}}(\bullet)$ expressing the relation between a database value X_{DB} compared to a query value X_Q is:

$$\mathbf{Sim}_{\text{Planar}}(X_Q, X_{DB}) = \begin{cases} a_1^R(i)X_Q + a_2^R(i)X_{DB} + a_3^R(i) & \text{if } X_Q \leq X_{DB} \\ a_1^L(i)X_Q + a_2^L(i)X_{DB} + a_3^L(i) & \text{if } X_Q > X_{DB} \end{cases} \quad (4.3)$$

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 of a number of planes in each half plane. Specifically, five planes are used to model similarity in each half plane (i.e. $i \in \{1,2,3,4,5\}$). Each plane expresses similarity within a certain similarity (output) range. An example of the plane configuration for the left half is shown in figure 4.2. 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 4.3. This section shows the similarity function for a specific query value X_Q (the white line of figure 4.3). Such sections of the planes are used after the system is trained to calculate similarity of candidate database values to a specific user query.

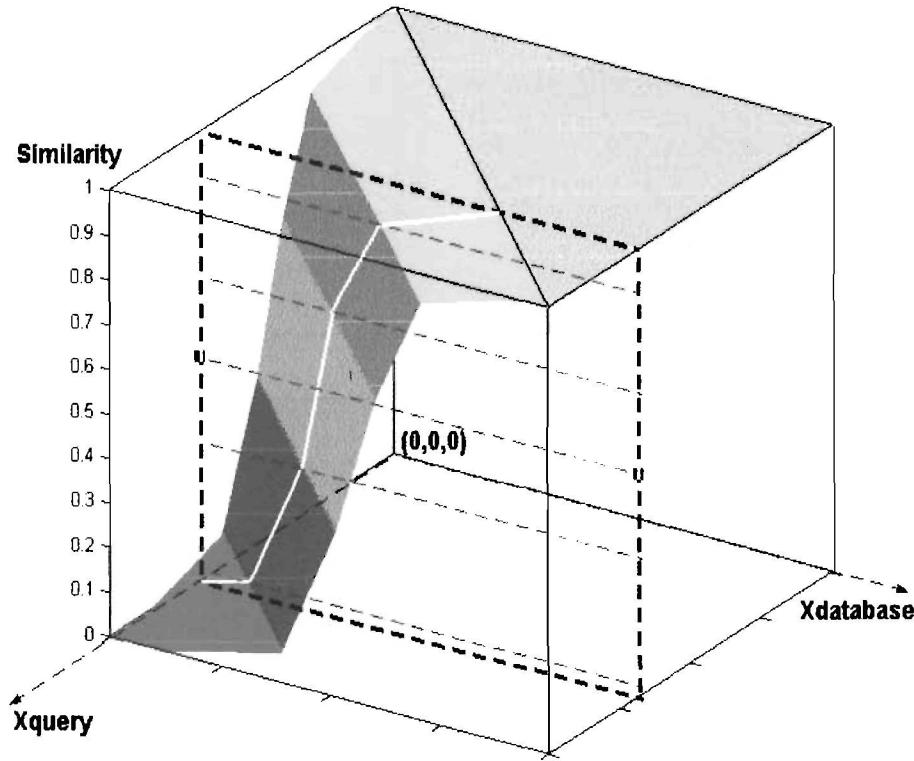


Figure 4.2: Piecewise planar similarity function

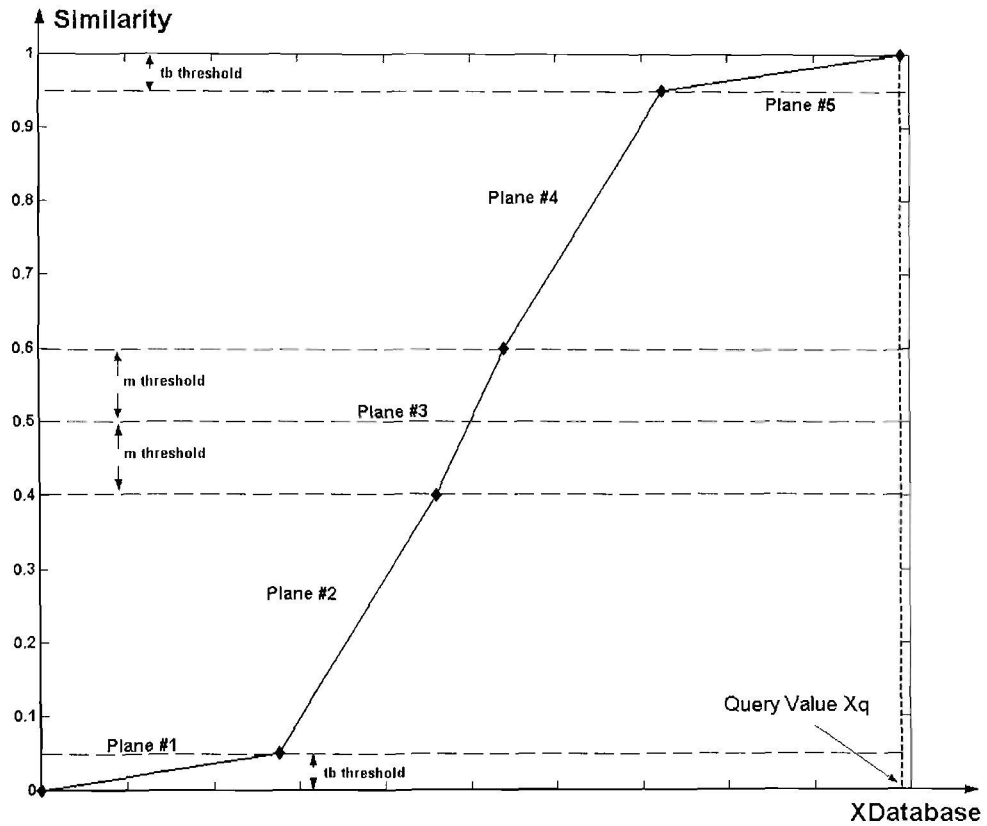


Figure 4.3: 2D section for specific query request

Within our process we use exactly five planes in each half of the input space. This is done for two reasons, namely:

- **To address ranges where similarity function is almost parallel to the $X_Q X_{DB}$ 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. Very small 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. We also use this threshold as a backup for cases where we might not obtain a solution so we can assign a direct value. Furthermore, we want to be able to handle cases where the

expected similarity might follow a linear behavior but be active only in a portion of the $[X_Q, X_{DB}]$ space (fig. 4.4). The tb value defines the starting and ending point of planes #2 and #4 respectively (fig. 4.2).

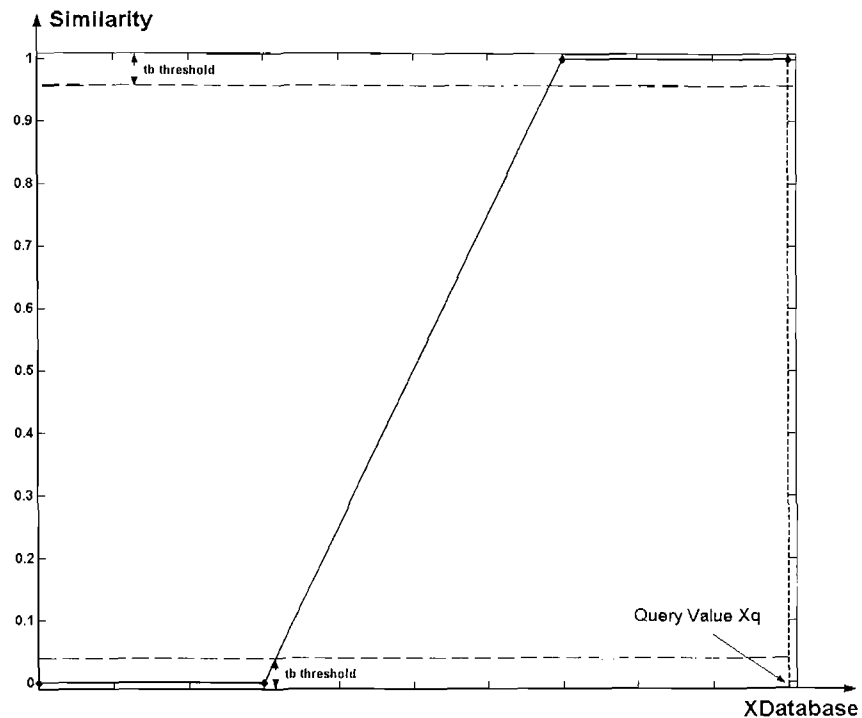


Figure 4.4: Partially active linear similarity function

- **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.

4.2.2 Mathematical solution

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

- 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 $a_1(i)$, $a_2(i)$.
- Successive planes should intersect at the specific similarity value as defined by thresholds tb and m . This is performed by using specifically targeted training samples and through a 3D line interpolation as we will further explain in section 7.1.2.

In order to make our system efficient first we perform a fast linear interpolation for each plane separately. To calculate the linear solution we use the $\mathbf{A}^* \mathbf{X} = \mathbf{L}$ formula where \mathbf{A} is the matrix containing the partial derivatives with respect to the unknowns, \mathbf{X} contains the unknowns and \mathbf{L} is the observation matrix. The solution is given by $\mathbf{X} = (\mathbf{A}^T \mathbf{W} \mathbf{A})^{-1} \mathbf{A}^T \mathbf{W} \mathbf{L}$, where \mathbf{W} is the weight matrix as explained below.

4.2.3 Weight manipulation to express training samples prioritization

Another interesting modification involves the formulation of the weight matrix \mathbf{W} in the least squares solution. If we assume independence between the samples of the training dataset then all non-diagonal elements of \mathbf{W} would be zero. Each diagonal element of \mathbf{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:

$\mathbf{W}_{\text{confidence}}$: User confidence in 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.

$\mathbf{W}_{\text{input}}$: 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.

$\mathbf{W}_{\text{output}}$: 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 the higher range of similarity (close to 100%) rather than the lower one (close to 0%).

The overall effect of the above three cases is expressed in the calculation of the \mathbf{W} matrix by:

$$\mathbf{W} = \mathbf{W}_{\text{confidence}} * \mathbf{W}_{\text{input}} * \mathbf{W}_{\text{output}} \quad (4.4)$$

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, \mathbf{W} can be omitted from the least squares solution.

4.3 Similarity dependence on input values

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

$$\varphi_p = \arctan\left(\frac{-a_1(i)}{a_2(i)}\right) \quad (4.5)$$

Angle φ_p should be the same for all planes in every half because we enforced the condition of having parallel footprints on the $X_Q X_{DB}$ plane. In figure 4.5 we show a piecewise planar similarity function. A contour plot representing similarity isolines is presented in figure 4.6. The calculated angle φ_p is the angle between the footprints (or the isolines since they are parallel) and the X_{database} axis. Here we should mention that we also examine the error associated with the calculated angle. This way we avoid situations where the angle might be expressing an average of highly deviating values. Therefore, we proceed with the method described next only if the associated error is within a predefined margin (e.g. 1-3 degrees). 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.

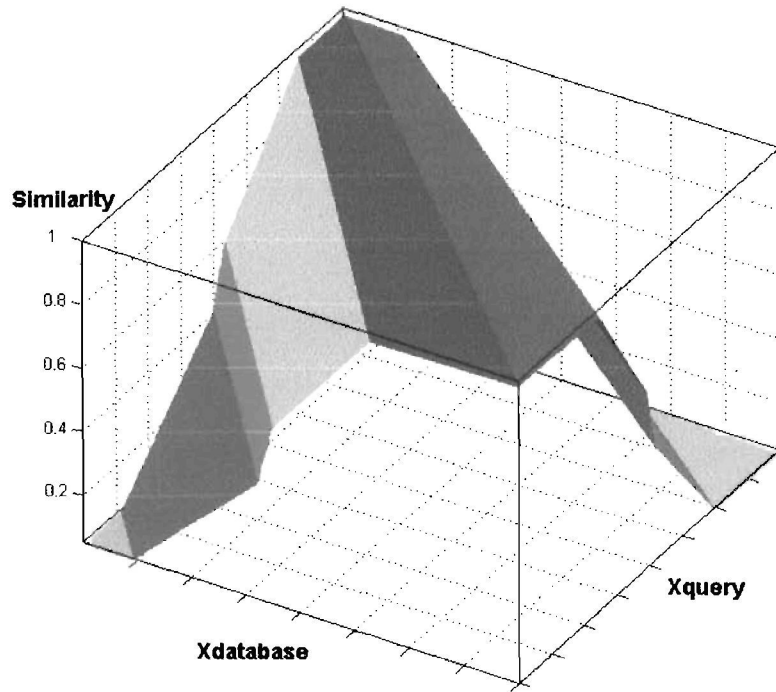


Figure 4.5: Asymmetric planar similarity function

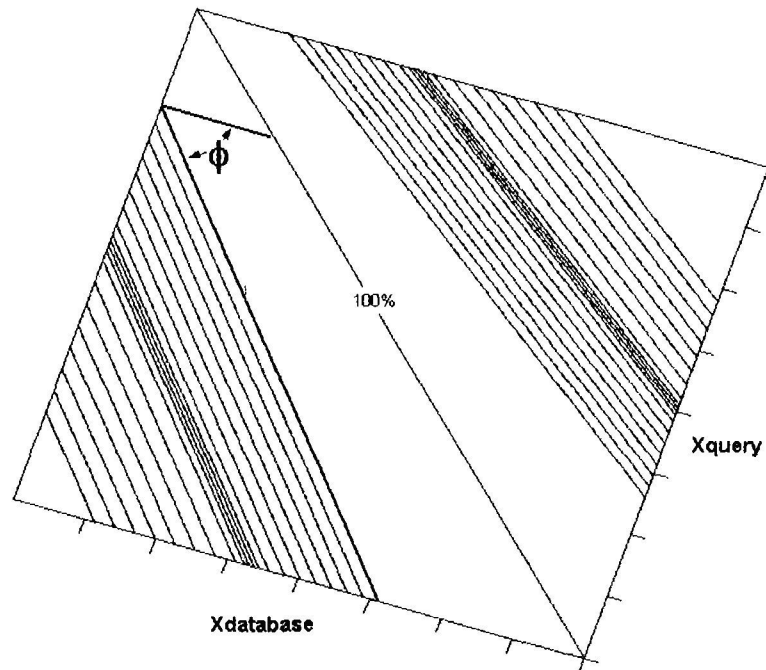


Figure 4.6: Planar function contour plot

4.3.1 Distance dependent case

In some cases the calculated angle might be close to the 45 degrees ($\varphi_p \approx 45^\circ$) (fig. 4.7).

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:

$$Sim_{Planar}(X_Q, X_{DB}) = \begin{cases} a_1(i)X_Q + a_2(i)X_{DB} + a_3(i) \\ a_1(i) \approx -a_2(i) \end{cases} \Rightarrow \quad (4.6)$$

$$Sim_{Planar}(X_Q, X_{DB}) = \begin{cases} a_1(i)X_Q - a_1(i)X_{DB} + a_3(i) \\ Dist = (X_Q - X_{DB}) \end{cases} \Rightarrow \quad (4.7)$$

$$Sim_{Planar}(X_Q, X_{DB}) = a_1(i) * Dist + a_3(i) \quad (4.8)$$

So we can conclude that similarity is not dependent on the actual values of X_Q, X_{DB} but only on their Euclidean distance [$Dist = X_Q - X_{DB}$]. This is a significant advantage of our design since our algorithm recognizes the currently used distance-based nearest neighbor case and provides support for it. Equation 4.8 shows that the planes can be replaced by lines providing a significant computational gain since a 3D problem is reduced to a 2D.

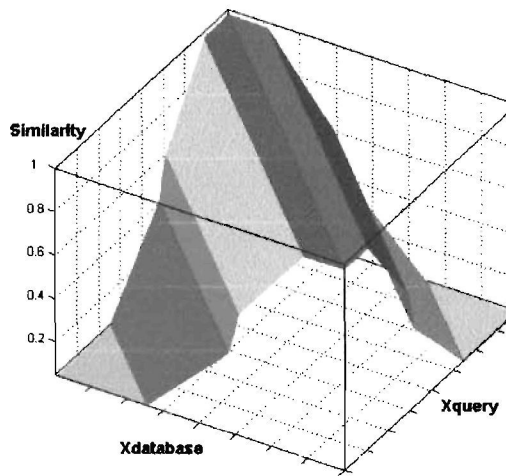


Figure 4.7: Rotation angle $\varphi_p \approx 45^\circ$

4.3.2 Database value dependent case

For the case that angle φ_P approaches the 90 degrees mark $\varphi_P \approx 90^\circ$ (fig. 4.8) the calculation formula of the angle provides $a_1(i) \ll a_2(i)$. With proper substitution in the

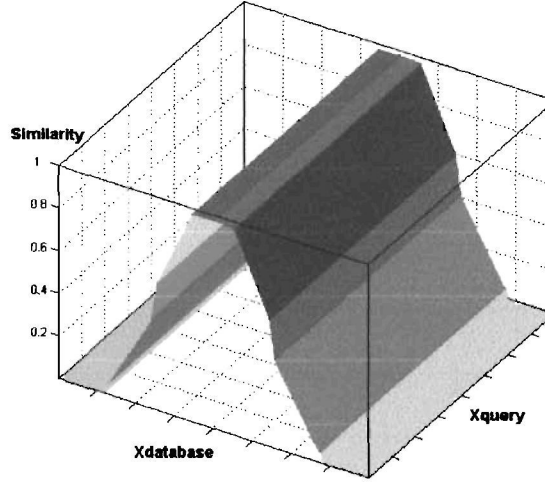


Figure 4.8: Rotation angle $\varphi_P \approx 90^\circ$

plane equation we have:

$$Sim_{planar}(X_Q, X_{DB}) = \begin{cases} a_1(i)X_Q + a_2(i)X_{DB} + a_3(i) \\ a_1(i) \ll a_2(i) \end{cases} \Rightarrow \quad (4.9)$$

$$Sim_{planar}(X_Q, X_{DB}) \approx a_2(i)X_{DB} + a_3(i) \quad (4.10)$$

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 that is expressed by equation 4.10.

An example of preference 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 either the overall similarity trend or 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 looking for a specific temporal instance of the phenomenon under investigation (e.g. iceberg separation). None of them knows the exact time and they all express their temporal preference based on their expertise on the training datasets.

Our system design overcomes 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° . In this case their similarity behavior should be expressed by a 3D surface similar to the one of figure 4.8. 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 4.10.

4.4 Sigmoidal similarity function

4.4.1 Mathematical formulation

After the plane interpolation is performed, an accuracy assessment through a fitting error takes place. For our application we use the Root Mean Square Error (R.M.S.E.). If the error is high a more complex function is triggered. 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 (Vlachos et al., 2002). Our similarity function is composed of 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:

$$Sim_{Sigmoidal}(X_Q, X_{DB}) = \begin{cases} \frac{1}{1 + e^{-a_R(X_R)}} & \text{if } X_Q \leq X_{DB} \\ X_R = (X_Q - X_{DB} - c_R)\cos\varphi_R + (X_Q + X_{DB} + c_R)\sin\varphi_R \\ \frac{1}{1 + e^{-a_L(X_L)}} & \text{if } X_Q > X_{DB} \\ X_L = (X_Q - X_{DB} - c_L)\cos\varphi_L + (X_Q + X_{DB} + c_L)\sin\varphi_L \end{cases} \quad (4.11)$$

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. 4.9). 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 the appropriate solution to express preferences from a variety of fuzzy membership functions.

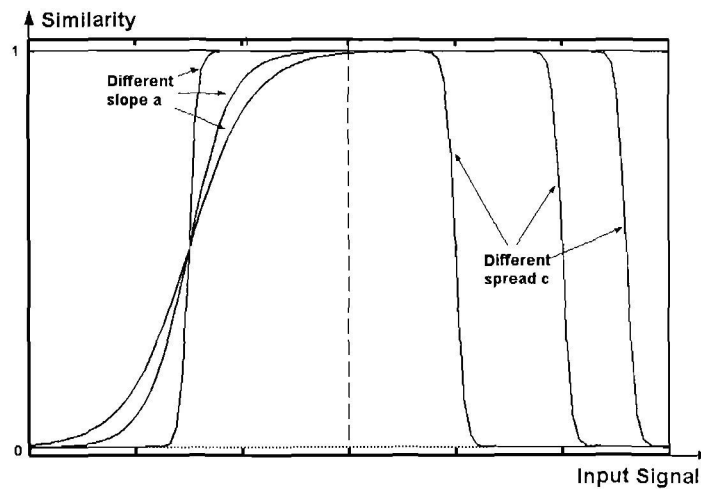


Figure 4.9: Slope and spread influence on sigmoidal's shape

4.4.2 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 φ as calculated before for the initial value of the rotation angle of the sigmoidal (for computational consistency $\varphi = \varphi_p - 45^\circ$). Also from the mathematical properties of our sigmoidal function we know that spread c corresponds to the value where the sigmoidal similarity function will return 0.5 as output (fig. 4.10). That is the main reason we introduced plane #3 earlier and the threshold m . We want m to be as small as possible but

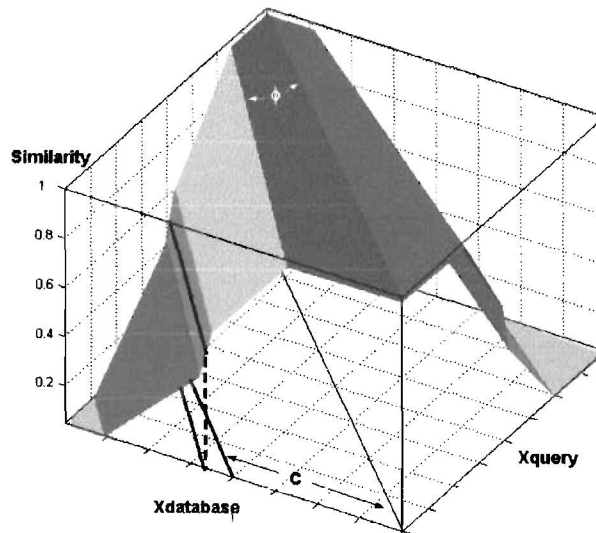


Figure 4.10: Calculating sigmoidal initial parameters based on planar solution

at the same time include enough samples to have an accurate result. So using the properties of plane #3 we calculate:

$$c = \frac{-(0.5 - a_3)}{a_2} \quad (4.12)$$

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 ϕ 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 ϕ 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 $\mathbf{A} \cdot \delta \mathbf{X} = \mathbf{L}$ formula where \mathbf{A} is the matrix containing the partial derivatives with respect to the unknowns, $\delta \mathbf{X}$ contains the unknowns and \mathbf{L} is the observation matrix.

Specifically if we have n training points $[X_Q, X_{DB}, Sim]$ to calculate the sigmoidal parameters a, ϕ and c the formulation of the matrices would be:

$$A(n \times 3) = \begin{bmatrix} \frac{\partial Sim^1}{\partial a} & \frac{\partial Sim^1}{\partial c} & \frac{\partial Sim^1}{\partial \phi} \\ \dots & \dots & \dots \\ \frac{\partial Sim^n}{\partial a} & \frac{\partial Sim^n}{\partial c} & \frac{\partial Sim^n}{\partial \phi} \end{bmatrix}, \delta \mathbf{X} (3 \times 1) = \begin{bmatrix} \delta a \\ \delta c \\ \delta \phi \end{bmatrix}, L(n \times 1) = \begin{bmatrix} Sim^1 \\ \dots \\ Sim^n \end{bmatrix} \quad (4.13)$$

$$\text{where:} \quad K = (X_Q - X_{DB} - c) \cos \phi + (X_Q + X_{DB} + c) \sin \phi \quad (4.14)$$

$$\frac{\partial Sim^n}{\partial a} = \frac{K e^{(-a K)}}{(1 + e^{(-a K)})^2} \quad (4.15)$$

$$\frac{\partial Sim^n}{\partial c} = \frac{a (-\cos \phi + \sin \phi) e^{(-a K)}}{(1 + e^{(-a K)})^2} \quad (4.16)$$

$$\frac{\partial Sim^n}{\partial \phi} = \frac{a (-(X_Q - X_{DB} - c) \sin \phi + (X_Q + X_{DB} + c) \cos \phi) e^{(-a K)}}{(1 + e^{(-a K)})^2} \quad (4.17)$$

The solution is given by $\delta \mathbf{X} = (\mathbf{A}^T \mathbf{W} \mathbf{A})^{-1} \mathbf{A}^T \mathbf{W} \mathbf{L}$. \mathbf{W} is the weight matrix as defined previously in the planar solution (equation 4.4).

4.5 Advanced fuzzy similarity functions

When the underlying complexity of preference is high, the already presented similarity functions might not be able to model it adequately. 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.

4.5.1 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 fuzzy membership function 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.

Mathematically this can be expressed as follows: Let function $\mathbf{F}(\bullet)$ represent a FMF mapping of our two-dimensional input space to one-dimensional similarity space:

$$\mathbf{F}: \mathfrak{R}^2 \rightarrow [0,1] \quad (4.18)$$

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

$$\mathbf{P} = \{p_1, p_2, \dots, p_n\} \quad (4.19)$$

In this case the arguments of function $\mathbf{F}(\bullet)$ can be expressed as:

$$\mathbf{F}(X_Q, X_{DB} | \mathbf{P}) \quad (4.20)$$

Now let us assume that each parameter p_i in equation 4.19 is not constant. Instead, it is expressed through a function $\mathbf{P}_i(\bullet)$ and it is dependent on values X_Q, X_{DB} . Also function $\mathbf{P}_i(\bullet)$ with $i = \{1, 2, \dots, n\}$ has its own set of parameters K_i . This leads to the general expression for inputs to function $\mathbf{F}(\bullet)$ which is :

$$\mathbf{F}(X_Q, X_{DB} | [\mathbf{P}_1(X_Q, X_{DB} | K_1), \mathbf{P}_2(X_Q, X_{DB} | K_2), \dots, \mathbf{P}_n(X_Q, X_{DB} | K_n)]) \quad (4.21)$$

For example, let us examine the sigmoidal function of equation 4.12. Similarity function $\mathbf{F}(\bullet)$ will be represented as:

$$\mathbf{F}(X_Q, X_{DB} | a, c, \varphi) = \frac{1}{1 + e^{-a((X_Q - X_{DB} - c) \cos \varphi + (X_Q + X_{DB} + c) \sin \varphi)}} \quad (4.22)$$

The number of parameters is three, namely a, c , and φ ($n=3$). 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 a function could be:

$$\mathbf{P}_2(X_Q, X_{DB} | c_0, c_1, c_2) = c_0 + c_1 X_Q^2 + c_2 X_{DB} \quad (4.23)$$

In this case we would have $K_2 = \{c_0, c_1, c_2\}$. So instead of trying to solve for parameters $\{a, c, \varphi\}$ our new more complex system would have higher modeling capabilities and would be expressed by a new set of parameters $[a, c_0, c_1, c_2, \varphi]$. 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 $[a^{new}, c_0^{new}, c_1^{new}, c_2^{new}, \varphi^{new}] = [a^{old}, c_0^{old}, 0, 0, \varphi^{old}]$.

4.5.2 Convoluting function output

We further enhance the operational range of our FMFs by introducing another important operation. 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 than a single function could express. An example would be periodicity combined with gradual decreasing interest (section 4.6.1). 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 constraints 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 $\mathbf{F}(\bullet)$ represent an FMF mapping and function $\mathbf{G}(\bullet)$ be for instance an administrative constraint. We have:

$$\mathbf{F}: \mathfrak{R}^2 \rightarrow [0,1], \mathbf{G}: \mathfrak{R}^2 \rightarrow [0,1] \quad (4.24)$$

Inputs for these functions are query and database values X_Q, X_{DB} . We define the convolution of functions $\mathbf{F}(\bullet), \mathbf{G}(\bullet)$ as their multiplication throughout the input space \mathfrak{R}^2 .

If function $\mathbf{H}(\bullet)$ is the resulting new function it can be represented as:

$$\mathbf{H}(X_Q, X_{DB}) = \mathbf{F}(X_Q, X_{DB}) * \mathbf{G}(X_Q, X_{DB}) \quad (4.25)$$

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

$$\mathbf{H}: \mathfrak{R}^2 \rightarrow [0,1] \quad (4.26)$$

Here we should note that we do not support training of function $\mathbf{G}(\bullet)$ and/or retrain function $\mathbf{F}(\bullet)$. 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.

4.6 Functionality examples

In this section we introduce a few 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 provide two examples of similarity preference in the temporal dimension and the connection speed dimension. A more comprehensive example with a step by step explanation of the algorithm is presented later in chapter 7.

4.6.1 Temporal similarity

A typical geospatial request can involve the temporal footprint of a geospatial object. 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. 4.11 dotted line). Another preference could result from the specifics of the problem which might require information only during specific months (seasons). A sinusoidal function can be used to model such preference (fig. 4.11 solid line).

Users would like though to combine both of the above 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 4.11. A specific example is

shown in figure 4.12. 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 for 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 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 models user preference for this task (fig. 4.12).

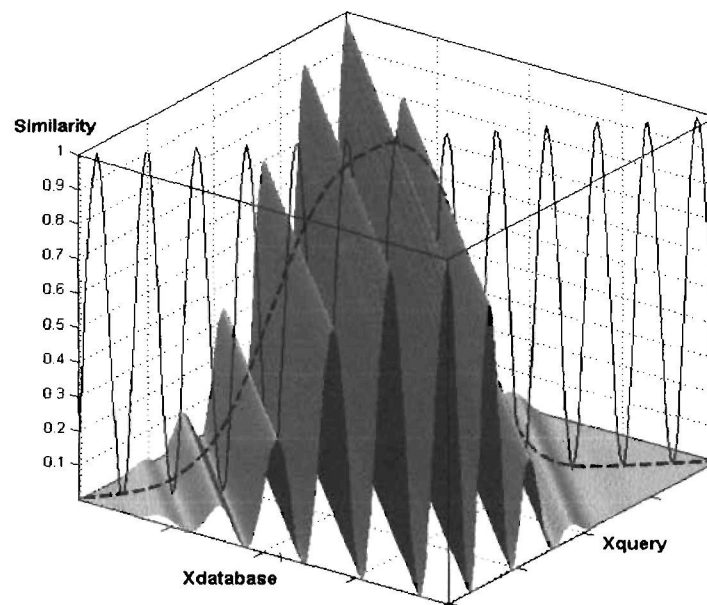


Figure 4.11: Sigmoidal with sinusoidal similarity functions convolution

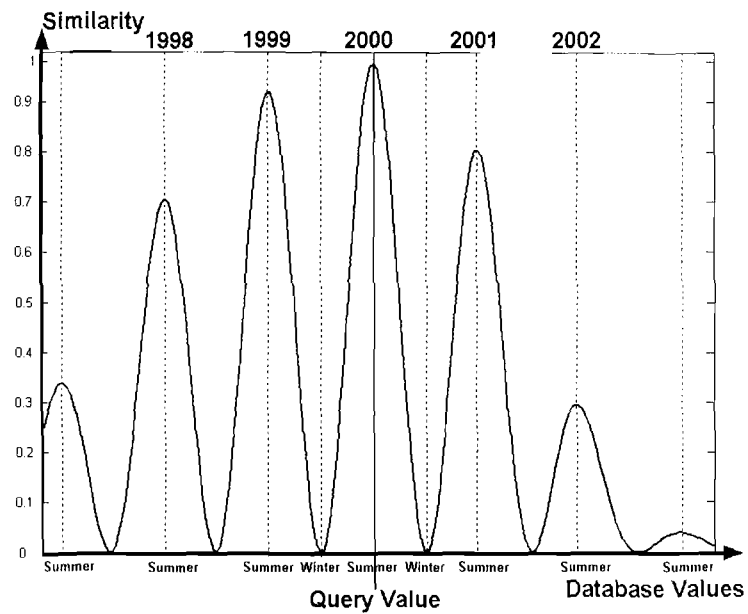


Figure 4.12: Time periodicity example

4.6.2 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 the system design, and they can be fixed or adaptable based on real-time monitoring of the database system.

Due to the large volume of geospatial datasets, data warehouses might be created leading to dataset availability through a variety of connected servers. Let us assume that GIS users request information in such a distributed environment. Depending on their connection speed they might query for servers of analogous speed. Because of high demand at some point all fast servers 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 4.13 with solid black line. Then the user similarity preference (dotted line) will be convolved with the system constraint and would provide a new similarity surface, the one of figure

4.13. In figure 4.14 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 while they would have been under normal circumstances.

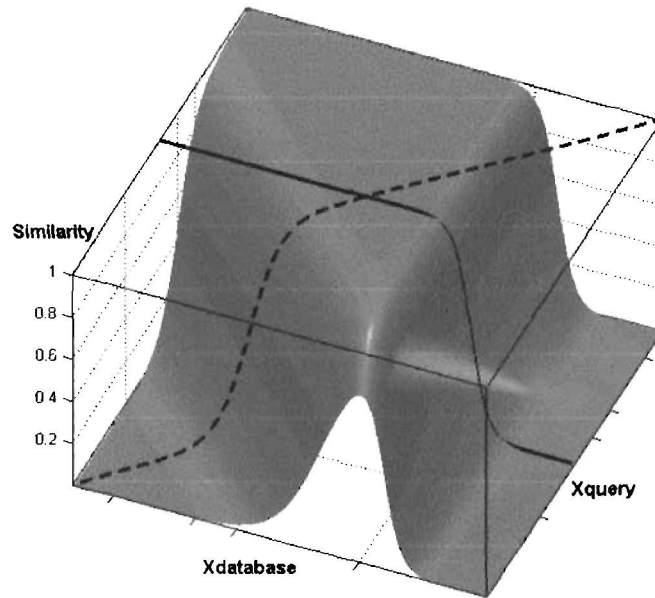


Figure 4.13: Sigmoidal with sigmoidal similarity functions convolution

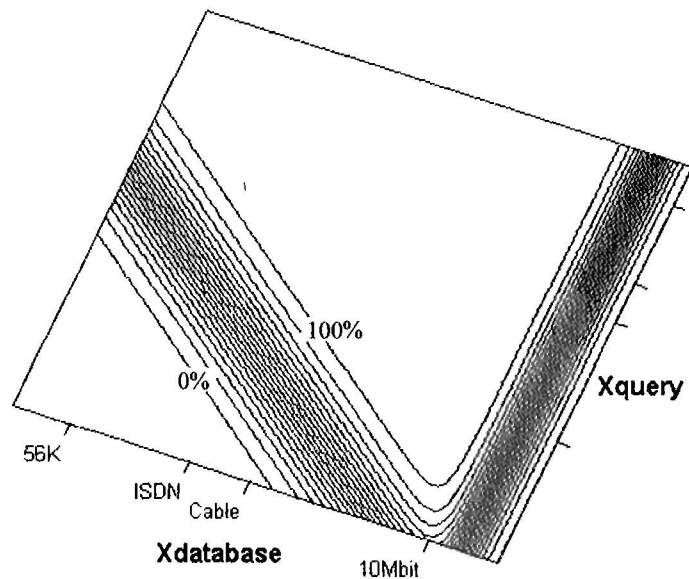


Figure 4.14: Connection speed example with similarity isolines

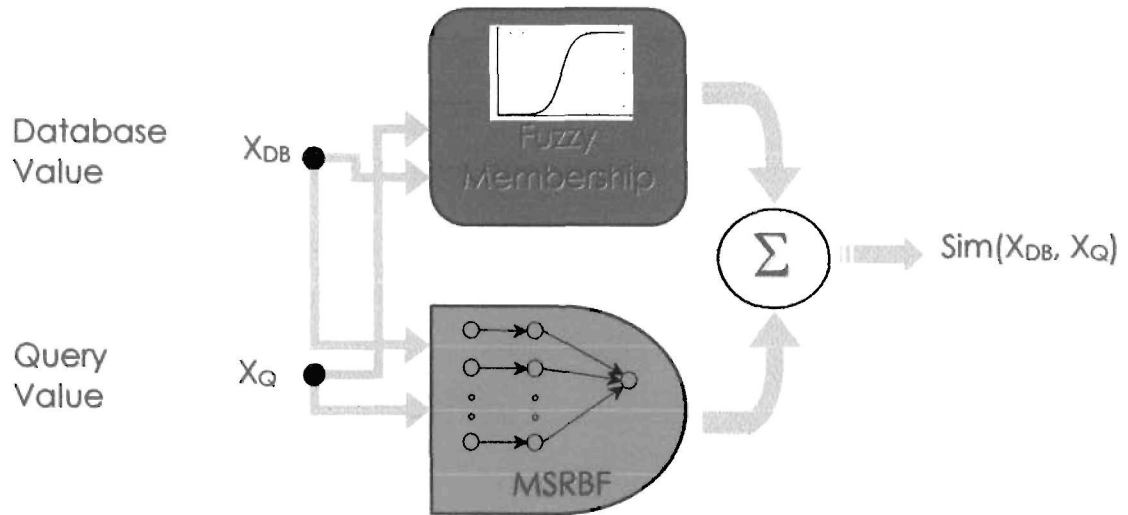
4.7 Conclusion

In this chapter we proposed a novel method for preference modeling within dimensions. Our approach accommodates gradual complexity increases in response to increased preference complexity. We use a variety of fuzzy membership functions to model preference expression. The training of the fuzzy similarity functions is performed using a backpropagation algorithm. 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 incorporation of user confidence in the provided response.

The gradual complexity increase is expressed through different sets of functions. Their specific design allows the use of properties from initial 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. Through advanced functions we enhance complexity by adding non-constant behavior to the function parameters. In addition to the gradual training, we presented a theoretical framework for preference function combination through mathematical convolution. We also describe examples preferences in geospatial information retrieval and how modeling these preferences is achieved using this mathematical convolution.

Chapter 5

Preference refinement using a multi-scale radial basis neural network



In this chapter we examine the application of a customized neural network to capture erroneous preference results obtained by the fuzzy method described in the previous chapter. Our approach is inspired by function approximation techniques. Preference modeling is achieved through a multi-scale neural network developed especially to accommodate the characteristics of our problem.

After providing a brief overview on neural networks and their applications, we justify the chosen general model that is based on radial basis functions. We investigate possible weaknesses of the general case and progressively build our novel network architecture by finding solutions. The final architecture is presented, accompanied by an information flow description during the algorithm training.

5.1 Why neural networks?

The basic idea of artificial intelligence (AI) is to plant human reasoning in a computer so the computer can behave intelligently. Artificial neural network (ANN) is a generic form of artificial intelligence for emulation of human thinking and tends to mimic biological neural networks with the help of computational electronics or software. It is often defined as emerging technology that mostly depends on soft or approximate computing. So would an ANN be appropriate for our problem?

The answer is yes. The investigation presented in this chapter will provide sufficient evidence. But what makes ANNs such a prominent tool to use? First of all, the common aspect of our task and the ANN definition: “emulation of human thinking that tends to mimic biological neural networks with the help of computational electronics or software”. We deal with a signal reconstruction issue where the signal expresses how humans perceive similarity among database values, in other words it encapsulates human thinking. Without getting too explicit, biological and AI researchers have identified a common parallel processing structure in the human brain and ANNs. Some might even claim that ANNs resulted from the study of the human brain. There exists extensive literature from different disciplines on the connection of human brain and ANNs. From an engineer’s point of view a ANN simulates a human brain to some degree by trying to:

- Examine inputs (stimulus) to provide a response in a forward-transmission manner.
- Extract salient features and model them through processing nodes (neurons in the human brain).

For further reading on that relation from the perspective of the neural network community the reader is referred to (Haykin, 1999).

5.2 Why neural networks with radial basis functions?

The analysis in this section justifies the choice of the general ANN model category based on our problem characteristics. Among dozen of different networks, there are two general classes: ones that provide function approximation and ones that simulate pattern classification problems. In the function approximation category, two are the commonly used types of feedforward networks, the Multilayer perceptrons (MLPs) and Radial basis function (RBF) networks. In the following sub-sections, first we provide a brief introduction on each network, and then a comparison through our task viewpoint is presented that leads naturally to our network type selection.

5.2.1 Multilayer perceptron networks

This kind of network typically consists of a set of source nodes that create the input layer, one or more hidden layers that represent the computation nodes and an output layer of these computation nodes (fig. 5.1). The input signal propagates through the network in a forward direction on a layer-by-layer basis. A MLP network has three distinctive characteristics (Haykin, 1999):

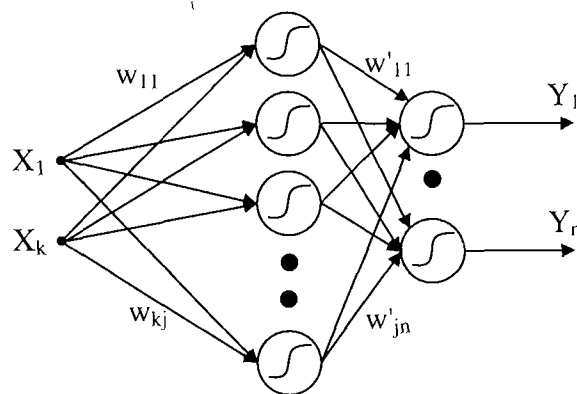


Figure 5.1: Multilayer perceptron network structure

- The activation functions of the computation nodes are nonlinear and differentiable throughout the input space. Typically the activation function is a logistic or tangent function. Hence the formula for the activation is:

$$y_j = \frac{1}{1 + e^{(-u_j)}} \quad (5.1)$$

where u_j is the weighted sum of all synaptic units connected to node j plus the bias.

- The network contains one or more layers of hidden neurons that are not part of the network input or output. By using this structure, the network is able to learn progressively more meaningful features from the input patterns that are presented.
- There is a high degree of connectivity determined by the synapses of the network. Change in the connectivity requires the network to be trained back from scratch.

5.2.2 Radial basis function networks

The MLP network described above can be seen as the application of a recursive technique that in statistics is defined as stochastic approximation. An alternative approach would be to design an ANN that would act as a curve-fitting hypersurface of the high-dimensional space, in essence attempting to find the best fit to the training dataset. The hidden units of such a network would be an arbitrary basis for the input patterns when they are expanded in the hidden space (Haykin, 1999). These functions are called radial basis function and the corresponding network a Radial basis function (RBF) network. There are many types of radial basis functions. Gaussian RBFs seem to be the most popular in the ANN literature. In the statistical literature, thin plate splines are also used (Green and Silverman 1994).

RBFs are usually a combination of an input layer, a single hidden layer with the radial basis functions and a linear output layer (fig. 5.2). The activation functions of the hidden layer are based on the Euclidean distance between the input vector and the weight vector. Then a nonlinear transformation is applied from the input space to the hidden space. Finally the output is calculated by a weighted summation of the hidden layer response.

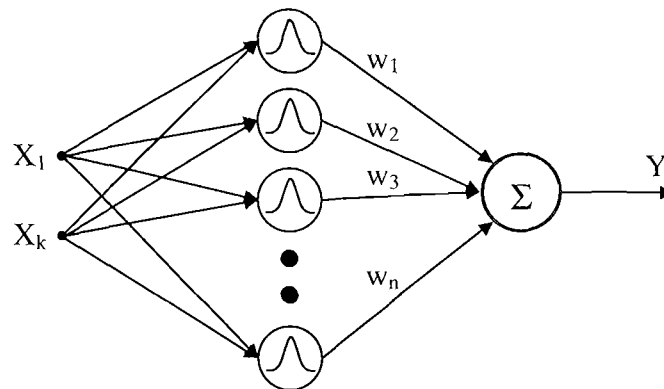


Figure 5.2: Radial basis function network structure

5.2.3 Similarities and differences

First let us examine the similarities between MLPs and RBFs. Similarities are not as important though because the chosen model is based on the exclusive characteristics of the winning ANN type. Keeping that in mind, we can summarize as follows:

- The RBF Networks and the Multilayer Perceptrons are layered feedforward networks that produce nonlinear function mappings.
- They are both proven to be universal approximators. By using this theorem we can mathematically prove that they have the ability to approximate an arbitrary continuous function.

On the other hand these are the main differences between them:

- The way in which hidden units combine values coming from preceding layers in the network: MLPs use inner products, while RBFs use Euclidean distance.
- The methods for training MLPs and RBF networks, although most methods for training MLPs can also be applied to RBF networks.
- The nodes in the hidden and output layers of MLP use the same activation function, while RBFs use different activation functions at each node (Gaussians parameterized by different centers and variances).
- The hidden and output layers of MLP are both nonlinear, while only the hidden layer of RBFs is nonlinear (the output layer is linear). Also an RBF network has only one hidden layer, while MLP networks have one or more hidden layers depending on the application task.
- MLPs construct global approximations while RBF construct local approximations.
- MLP is harder to train but RBF might require more computational nodes to achieve the same accuracy.

5.2.4 Chosen network type

In the previous sections we introduced the available options for our neural network structure. To support our final decision we revisit our problem and using the problem specifications and requirements the selected model is justified.

Our neuro-fuzzy system builds a preference model based on user feedback. We use a backpropagation algorithm to train the fuzzy membership functions that act as global function approximations as presented in the previous chapter (fig. 5.3). In other words fuzzy functions describe the “anticipated” behavior of the preference expression throughout the input space. But there are also localized behaviors that a global function

cannot represent. And there is also some inherent noise that should be decreased. In figure 5.4 the resulting error from the fuzzy function of figure 5.3 is presented. If we examine the error function we can see that there is one high value in the middle where the user expressed a highly unexpected similarity choice. There are also errors of smaller amplitude that need to be modeled.

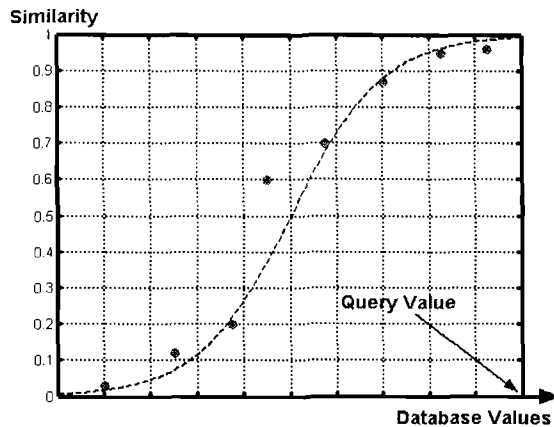


Figure 5.3: Fuzzy membership function interpolation

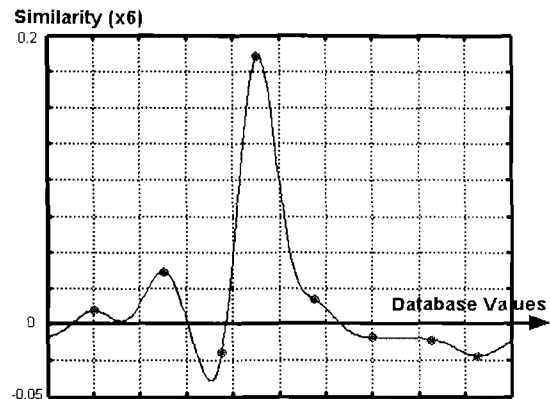


Figure 5.4: Resulting error from fuzzy membership function

We claim that our solution should be:

“A local-nature artificial neural network”

The rationale behind this comes from the fact that we expect the majority of the preference expression to be modeled by the fuzzy function. In other words, this is the behavior that results by training on specific rules/knowledge. We trust this fuzzy function to carry on the preference where not enough training data exist - a good generalization is anticipated. So our neural network should be limited to localized neighborhoods of the input space - appear only where it is necessary.

This leads us to the choice of a Radial basis function network. RBF architectures have local receptive fields, meaning that changing the hidden-to-output weights of a

given unit will affect the output of the network only in a neighborhood around the center of the hidden unit, where the size of the neighborhood is determined by the spread of the hidden unit. This satisfies the requirement of a localized solution.

Local receptive fields have an advantage compared to the distributed architecture of MLPs, since local units can adapt to local patterns in the data without causing unwanted side effects in other regions. In a distributed architecture such as a MLP, adapting the network to fit a local pattern in the data can cause spurious side effects in other parts of the input space. For our application this means that a modular node structure can be implemented. By adding or dropping a node we know the neighborhood of the input space that will be affected. So as new training data might arrive, our network can be updated without having to retrain it (assuming no significant changes).

Another desirable feature comes from the training of the network. The dimensionality of the input space is low (2D) so visualization is an option. Problematic areas can be identified and improvements should be made in the network structure. If a MLP would be chosen it would be extremely hard to identify the nodes causing errors because MLP activation functions have a broad activation range in the input space. On the other hand RBFs can be easily analyzed and corrected due to their local receptive fields. In figure 5.4 an RBF is applied on the error function. The combined result of the fuzzy function and the RBF is shown in figure 5.5.

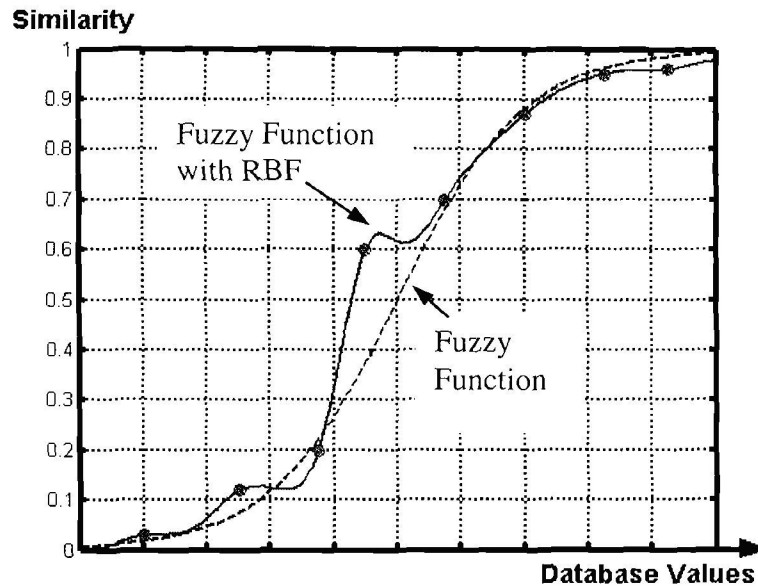


Figure 5.5: Fuzzy membership function with RBF

5.3 Radial basis network for the non-expert

In this section we provide an inside look on how RBFs work. Special consideration is given to non-expert users through a step-by-step walk through. We also examine the simplest version of RBF, an exact one (number of nodes = number of training points). Advanced users should proceed to the next section.

Let's assume that we have a function approximation problem. The input to our network is one-dimensional and so is the output. We also choose to use 3 hidden nodes and that these nodes are Gaussian distributions (fig. 5.6). Each Gaussian has a pre-defined width (sigma) of value s and specific mean as well.

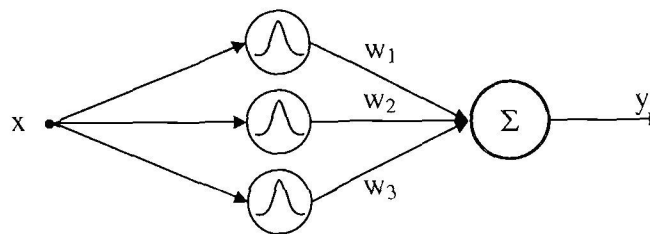


Figure 5.6: Simple RBF example – 1 input, 3 hidden nodes and 1 output

Now let's attempt with the above network to approximate the three points of figure 5.7. First here is how the network works (information flow is shown with connectivity arrows in figure 5.6). Each point is presented to the network. The signal goes from the input layer to the hidden one. Over there the output of each Gaussian node is calculated based on the distance between the point and the center of each Gaussian. These are the bottom three Gaussians in figure 5.7 represented with dark shadowed fillings. Then the output is multiplied with a weight, in other words scaled up or down. The corresponding three new Gaussians are shown with dotted line. Finally the weighted output of each node is added in the linear node. That is the output of the network as presented with a solid black line.

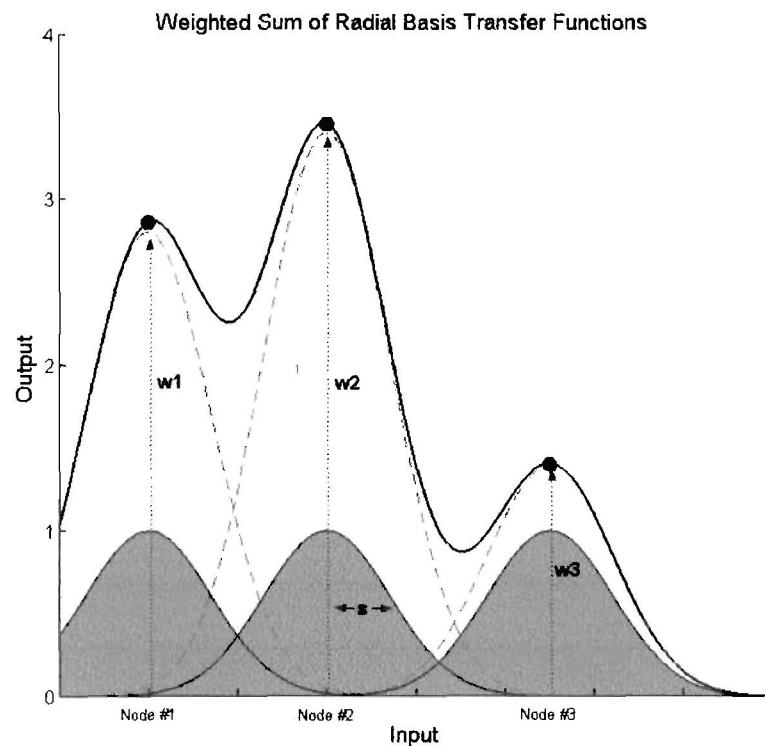


Figure 5.7: Inside look at RBF operation

5.4 Multi-scale RBF network

In the previous section we discussed the merits for our application coming from an RBF-type network. In this part we address issues related to the built-in properties of a multi-scale RBF that facilitates our needs. First, we investigate the selection of node centers and activation functions. A discussion follows on how a variable width (multi-scale) approach would improve performance and affect the formulation of the activation functions. To facilitate visualization purposes the investigation is based on an one-dimensional input even though our task has a 2D input. Only where it is necessary, a reference to the second dimension will be provided.

5.4.1 Selection of node centers

In the literature two general approaches exist for center selection of RBF nodes. They can either be fixed or calculated during the training process of the network. We provide a brief introduction for each and then assess their possible benefits for our network.

The first category requires the centers to be fixed throughout the training process.

The centers are preselected:

- in a random fashion from the training dataset, or
- based on a self-organized learning technique.

The first method is simple to implement, just center the nodes on the training dataset points and then train the network from there. The second choice requires a clustering algorithm that would divide the dataset into homogenous subgroups and use those subgroups as centers. Examples of such approaches include the k-means clustering algorithm, the enhanced k-means by (Chen, 1995) and the self-organizing maps (Kohonen,1990).

In the second category, the RBF takes its most generalized form where centers are not predefined but are computed as additional parameters during training. A study that compared fixed and parameterized solutions performed by (Wettschereck and Dietterich, 1992) showed that the parameterized solution outperforms the non-adaptable one, which is expected since the model has higher flexibility with added parameters.

Projecting the above techniques to our task we would have to choose a fixed-center approach with the centers based on a subset of the training dataset. This results from the following constraint we apply on our network:

- Activate only at neighborhoods when there is substantial evidence of error and do not generalize outside these neighborhoods of the input space.
- Trust the fuzzy membership function to generalize outside the neighborhoods.

Two interrelated reasons can cause the network to expand beyond the desired limits: the selection of centers and the selection of width. Here we examine the former, the latter is presented afterwards in the chapter. The main problem with the self-organized and parameterized methods is that there is not much control on node center location. Neighborhoods and other constraints can be applied but there is no warranty that the network will generalize in the desired fashion. In most cases this lack of control would be disregarded, it would be interpreted as higher adaptability of the system, but that is not our case. An example is shown in figure 5.8. The training points can be seen with black diamonds. The former two methods would probably lead to the choice of the high-amplitude Gaussian at the center (dotted line). Our choice would result in the two low-amplitude Gaussian left and right (solid line). During the training process the high-amplitude solution would provide a better error estimation. But when the generalization

would take place the results would be erroneous. An example is presented in figure 5.9.

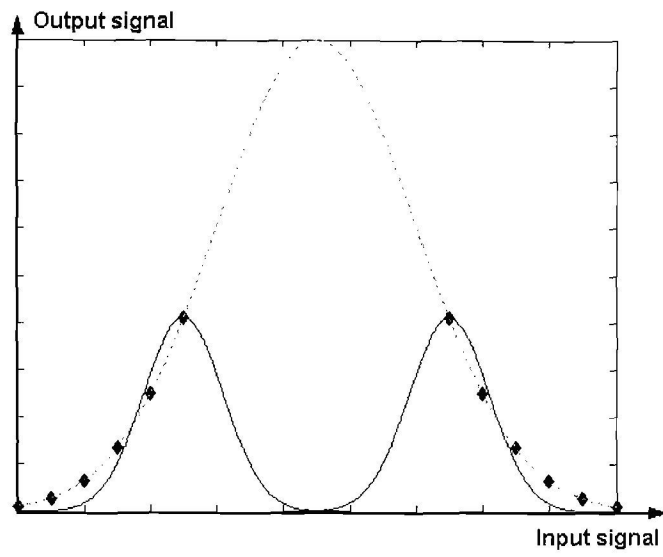


Figure 5.8: Node center selection

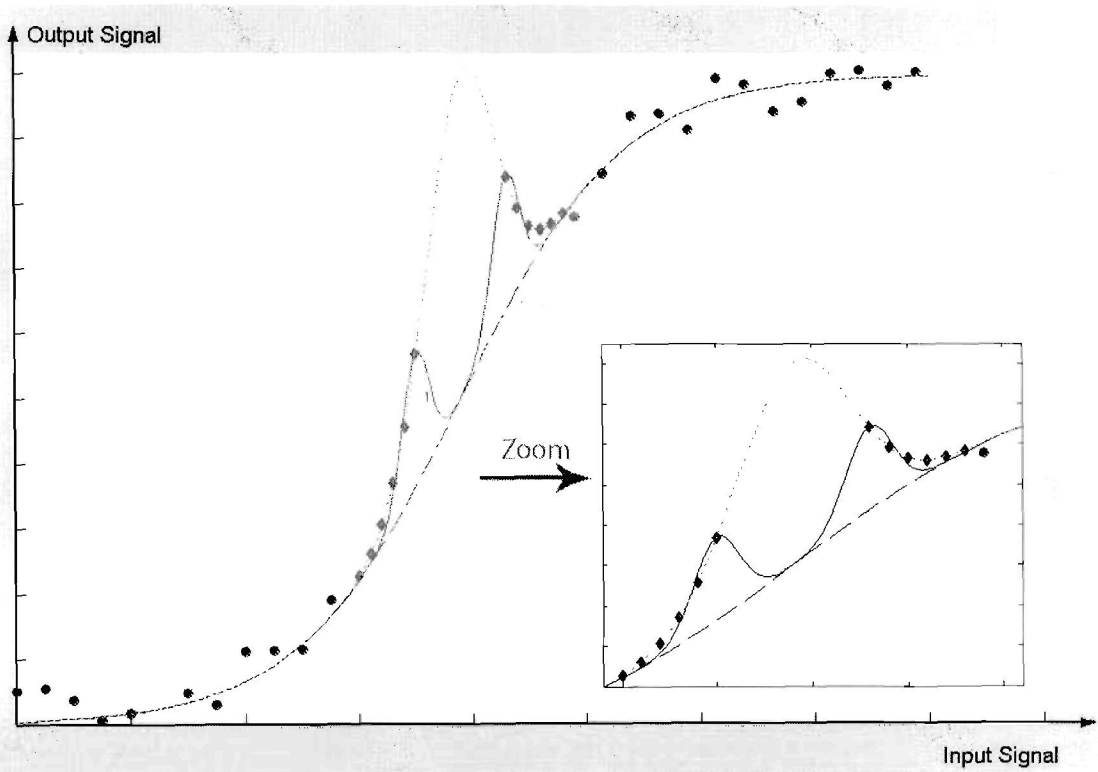


Figure 5.9: Node center selection generalization example

5.4.2 Selection of activation functions

Inside every node of an RBF network there is an activation function, a function that maps an input signal to an output one. This function is non-linear and should belong to a Green's function category (Haykin, 1999). Such a function has two properties:

- Its linear differential operator is invariant to translation and rotation.
- Its response depends only on the Euclidean norm of the difference of the center vector and the presented input pattern.

Under these conditions a chosen function is a radial-basis function. A large class of radial-basis functions has been introduced in the past, with Gaussian ones dominating the literature. Other types include thin-plate splines, logistic basis functions, multiquadrics and inverse multiquadrics (Hardy, 1971). In our approach we use two non-linear function classes, the predominant Gaussian and a symmetrical Sigmoidal class.

5.4.2.1 Gaussian activation functions

The first class of activation functions used in our MSRBF is based on a Gaussian distribution. Gaussian functions are popular activation functions because they satisfy the radial-basis function conditions mentioned above, and at the same time they support traditional statistical analysis, a consequence of their Bayesian form.

For a uniform standard deviation σ_a activation function $\Phi(\bullet)$ of node j (fig. 5.10)

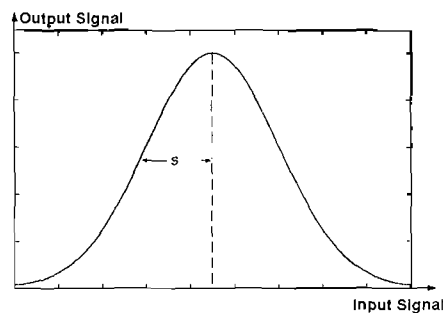


Figure 5.10: Gaussian function

can be expressed as:

$$\phi_j(\vec{x}) = \exp\left\{-\frac{\|\vec{x} - \vec{\mu}_j\|^2}{2\sigma_a^2}\right\} \quad (5.2)$$

where \vec{x} is the d-dimensional input vector with elements x_i and $\vec{\mu}_j$ is the vector determining the center of basis function ϕ_j and has elements μ_{ji} .

The Gaussian radial-basis functions can be generalized to allow for arbitrary covariance matrices Σ_j . In this case the equation 5.2 takes the following form:

$$\phi_j(\vec{x}) = \exp\left\{-\frac{1}{2}(\vec{x} - \vec{\mu}_j)^T \Sigma_j^{-1} (\vec{x} - \vec{\mu}_j)\right\} \quad (5.3)$$

The current problem we examine involves two inputs and one output. The inputs are the Database value and the Query value and the output is the similarity metric. Depending on the formulation of the covariance matrix Σ_j the resulting Gaussian activation function can have equal axes scale (fig. 5.11a,d) or unequal axes scale (fig. 5.11b,e) and can be rotated (fig. 5.11c,f).

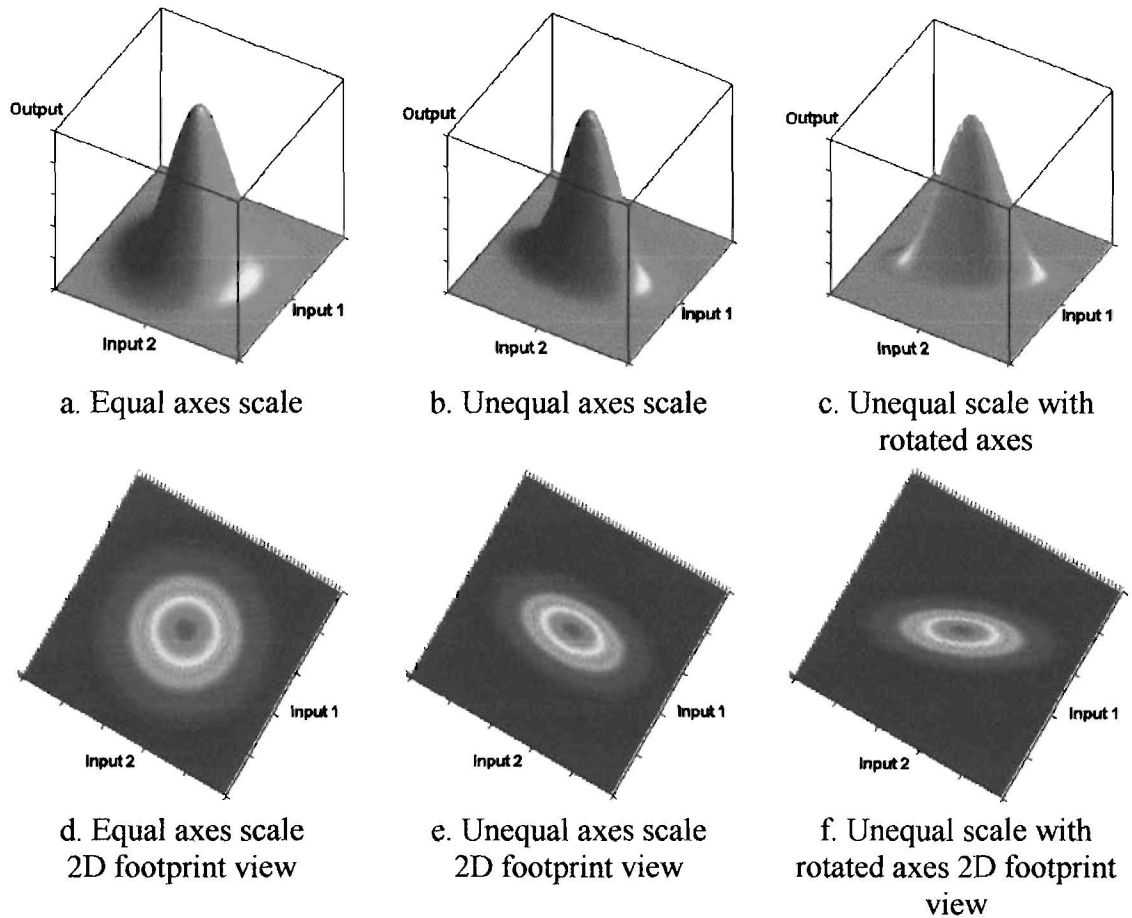


Figure 5.11: Different 3D Gaussian activation functions

5.4.2.2 Sigmoidal activation functions

During the initial development of the MSRBF, we realized a limitation of using Gaussian activation functions. Their highly active receptive field is rather small compared to the total receptive field of the function as seen in figure 5.11. This results into limited modeling capabilities when a uniformly highly stimulated area exists. To overcome such a problem we would have to use a large number of nodes. Another approach that traditional RBF networks follow is the introduction of an external bias. But in our case we cannot use a global bias because that would overturn the goal of this network as defined previously:

Allow network activation only where there is substantial evidence of error.

The role of a global bias is placed upon the non-linear fuzzy membership function. So we introduce another class of radial-basis functions that act as localized bias, namely the symmetric Sigmoidal one. The activation function $\Phi(\bullet)$ of node j can be expressed as:

$$\phi_j(\vec{x}) = 1 / (1 + \exp\{-a_a[\text{abs}\|\vec{x} - \vec{\mu}_j\| - c_a]\}) \quad (5.4)$$

where \vec{x} is the d -dimensional input vector with elements x_i and $\vec{\mu}_j$ is the vector determining the center of basis function ϕ_j and has elements μ_{ji} . Parameters a_a and c_a define slope and spread characteristics of the function (fig. 5.12). The Sigmoidal radial-basis functions can be also generalized to allow for arbitrary covariance matrices Σ_j . In

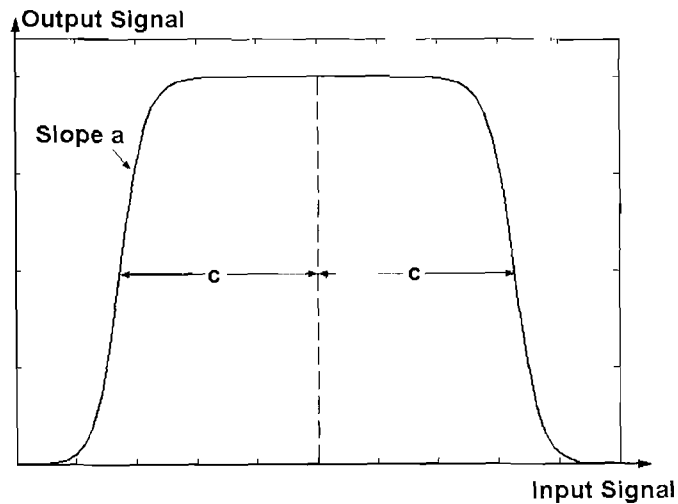


Figure 5.12: Sigmoidal function

this case the equation 5.4 takes the following form:

$$\phi_j(\vec{x}) = 1 / (1 + \exp\{-a_a \left([(\vec{x} - \vec{\mu}_j)^T \Sigma_j^2 (\vec{x} - \vec{\mu}_j)]^{1/2} - c_a \right)\}) \quad (5.5)$$

By manipulation of the covariance matrix Σ_j we can introduce different scales along each axis and introduce rotation along the output axis (fig. 5.13).

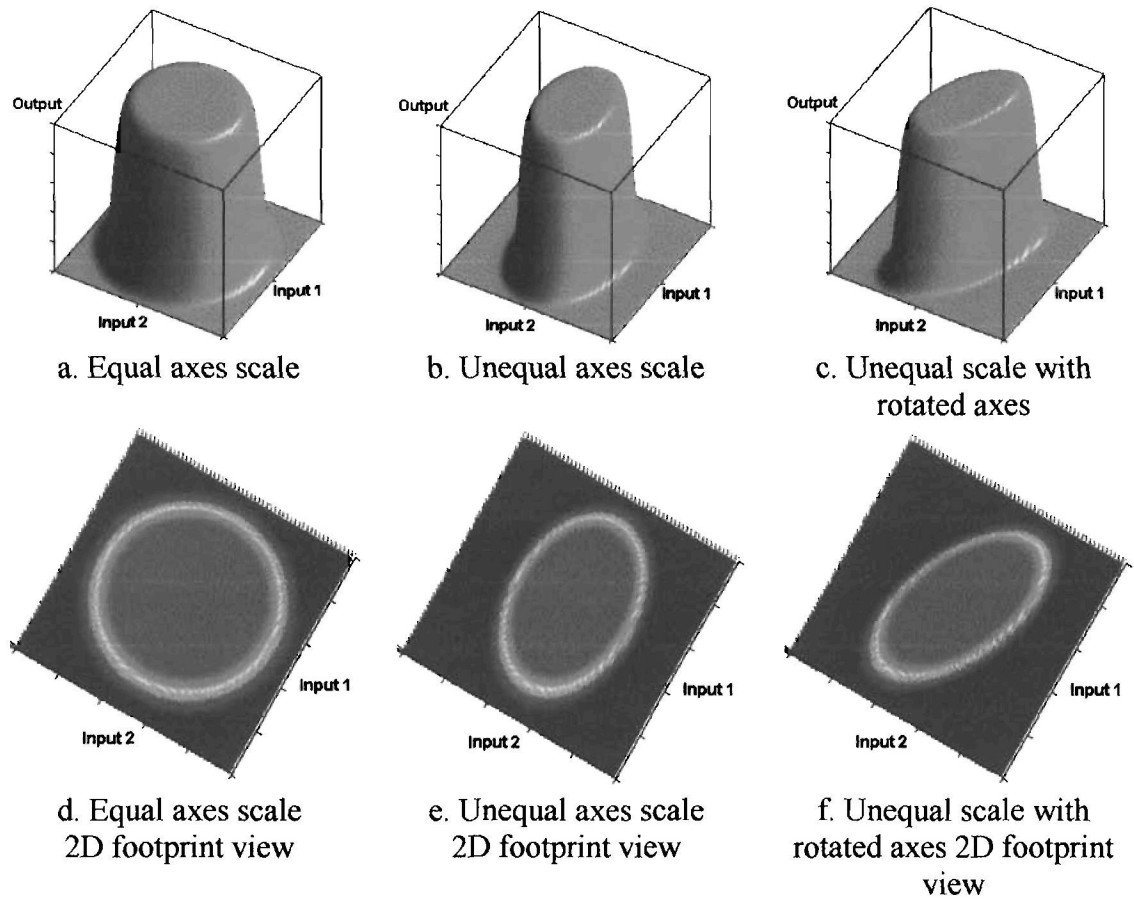


Figure 5.13: Different 3D Sigmoidal activation functions

5.4.3 Supporting a multi-scale analysis

This section introduces issues related to spread definition within an RBF. This is essential to our model since we want to incorporate variable spreads and produce a multi-scale RBF. First, we explain how spread affects the outcome of a network using a simple one-dimensional input example. Building on that, we discuss available techniques to overcome arbitrary spread selection. Our chosen method concludes the assessment. Note that spread refers to the definition of parameters σ_a and c_a of the Gaussian and Sigmoidal activation functions, respectively.

5.4.3.1 Spread influence in RBFs

In most RBF networks the node functions are predefined and the goal is to find the optimal set of weights. This implies that the parameters of each activation function will be constant throughout the iterations. In the case of a Gaussian activation function standard deviation σ_a and center vector $\vec{\mu}_j$ are predefined in equation 5.2. But how does the choice of these two parameters affect the result? The node center selection was discussed in section 5.4.1. The influence of standard deviation (spread) is our focus here. To help us with the investigation a set of training points is presented to three identical networks. Their only difference is the spread used in the Gaussian activation functions. The resulting mapping is displayed for $\sigma_a = \{0.03, 10, 12\}$ in figure 5.14.

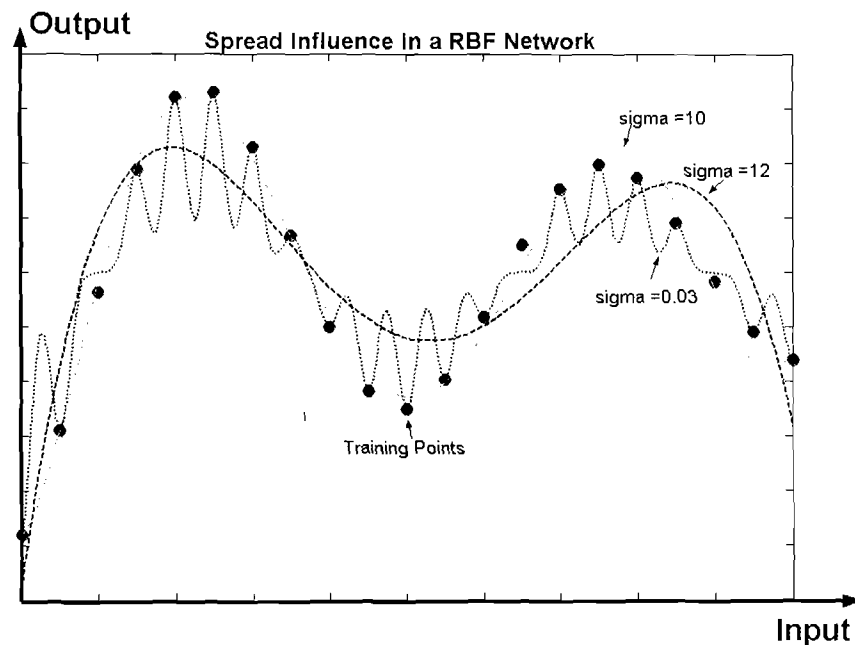


Figure 5.14: Spread influence in a RBF network

For $\sigma_a = 0.03$ the network output is relatively accurate on the training data. But when we try to generalize it we obtain poor results. This is due to the choice of a small spread that results in a narrow receptive field. This leads to a less “smooth” mapping since the

functions are too localized. On the other hand, when σ_a was set to 12 the mapping was more “smooth” than desired. This is caused by large receptive fields used for the Gaussian functions, constraining their ability to capture details in an appropriate manner.

The last value tested was for $\sigma_a=10$. This value seems to describe the problem adequately. It provides good training accuracy and good generalization at the same time. Judging from the above an important conclusion is that the spread parameter should be: *“large enough so neurons respond to overlapping regions of the input space, but not so large that all the neurons respond in essentially the same manner”*. Thus, depending on the problem small deviations in the spread value can have a significant effect to the overall outcome of the training.

5.4.3.2 Spread assessment in our MSRBF

So the big question now is how we define the optimal spread value for the task at hand. There are two general methodologies; one requires the spread pre-defined and another that treats it as an additional network parameter.

The first one defines the spread before the training of the network, then during the training process an evaluation of the chosen spread is provided. A common approach is to follow a trial-and-error technique by presenting the same training input to a variety of RBF networks with different standard deviations each time, and then evaluate the resulting accuracy after all spread candidates are presented. The area around the best width is then tested in more detail, and so on until the desired accuracy is achieved. Another solution would be to use the generalized cross-validation (Craven and Wahba, 1979). But this approach does not work in all cases as shown in (Wahba, 1990).

In the second category, the spread is not pre-defined but it acts as an additional parameter for the network in addition to the weights solution. Theoretically this is a more powerful solution due to the higher adaptability that comes from a flexible spread. But in practice this also increases the complexity of the training. Convergence of the network is not guaranteed due to the non-linear solution that is required and sometimes generalization issues appear. More specifically, in our application we impose some constraints on the network, on the receptive fields of the nodes. We desire a more “controlled” solution that would converge in a frequent manner, without any unexpected behavior. For example, if spread is an independent variable, this could generate large values that would contradict with our motivation of having the network act locally only when necessary and trust the fuzzy membership function to generalize beyond that.

Based on the above analysis, there is a trade-off between model adaptability and convergence/control degree. Since our network is a performance improvement and not necessarily the only means to successfully model the similarity signal we chose convergence and control at the spread specification. So the selected method is a trial-and-error approach where several spreads are tested and the best “statistical” solution is retrieved. At the same time though, model adaptability is significantly improved by using different “statistical” criteria as introduced hereafter.

5.5 Customized network training

Initial statistical experiments demonstrated that the use of traditional RBF networks increase accuracy. At the same time several problematic scenarios were identified that needed attention leading to new techniques and a customized multi-scale RBF. This section discusses limitations and concerns that rose during testing and solutions to overcome them. We should point out that throughout this section theory and applications will be used interchangeably. We follow a problem-solution approach while we increase the complexity rather than using the final algorithm flow as a guide. Information flow is presented in a subsequent section (5.7).

5.5.1 Traditional RBF training

Up to this stage we have established the network type, node activation functions and center/spread selection methodology. The next step concentrates on the RBF training, which takes place with a sample dataset. The dataset is divided randomly into two sets; one having 60-80% of the points formulating the training dataset and another set with the rest of the points that is used for evaluation purposes. Training and evaluation sets are different sets to ensure good generalization of the network, otherwise overfitting cases like fig 5.14 for $\sigma_a=0.03$ might appear.

During the network training we try to minimize error, the difference between the network actual and desired response. First, we have to make the network adjust (fit) to the training set and then estimate its final performance with the evaluation set. At this stage the evaluation set is ignored, we will assume that the training set is sufficient enough to provide good generalization.

Let's assume that a training set of n points is provided. The input dimension of the training set is 2 (x_1, x_2). There is also a one-dimensional output (y) describing the desired response for each input vector. An example of such a training set is given in table 5.1.

x_1	x_2	y
1.3	8.3	0.14
2.5	5.2	0.12
...
9.7	5.3	0.19
17.3	-11.4	0.05

Table 5.1: Example of a training set

In a multi-scale approach we would like to test several different spread values. A range $[\sigma_{min}, \sigma_{max}]$ is defined showing the minimum and the maximum spread value. Based on that a set of candidate spread values is produced. Also, a set of candidate centers μ_j is created. Since all points can act as activation function centers the μ_j set is the same as the input part of the training set.

After the parameters of the activation functions are defined all possible combinations of spread and centers are created. RBFs can be treated as high-dimensional curve-fitting approximations. In order to compute the best solution some statistical measures should be used. The most popular one is the Mean Square Error (*MSE*). The *MSE* is given by the following equation:

$$MSE = \frac{\sum_{i=1}^n (y - \hat{y})^2}{n} \tag{5.6}$$

where \hat{y} is the network response, y is the expected response from the training set and n is the number of points in the dataset. The *MSE* is calculated for every combination of spread/center as shown in table 5.2.

Spread σ_a	Center $\mu_j (x_1, x_2)$	MSE
4	(2,4)	0.12
4	(3,5)	0.17
4
4	(9,12)	0.09
6	(2,4)	0.07
6	(3,5)	0.03
6
6	(9,12)	0.08
...
14	(2,4)	0.07
14	(3,5)	0.10
14
14	(9,12)	0.22

Table 5.2: Node selection with MSE

The activation function with the lowest *MSE* is selected. If this *MSE* is better than a predefined threshold (that expresses the desired accuracy of the fitting) then the network stops there. If not, then all the training inputs are corrected based on the last activation function and the process is repeated. The training stops when either a predefined number of nodes is reached or the desired accuracy is achieved.

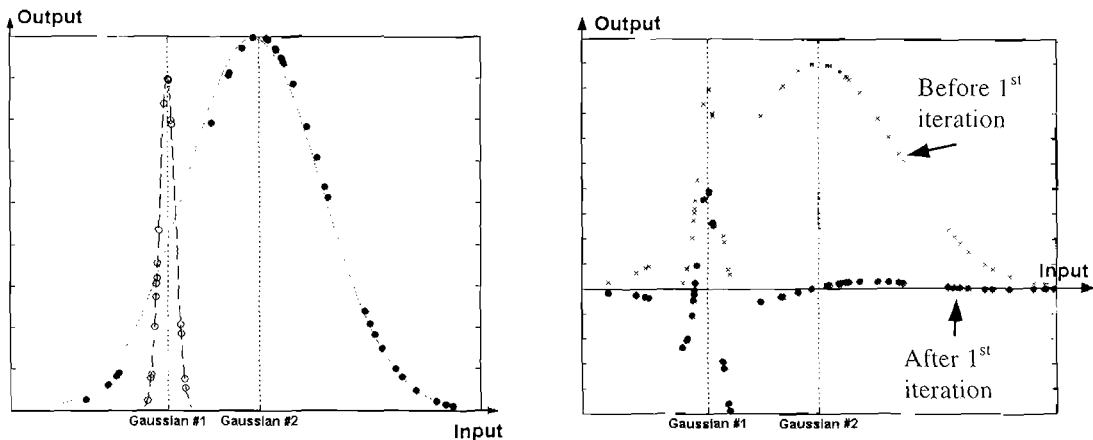
5.5.2 The multi-scale overlapping problem

A multi-scale approach is significantly more powerful since it has the ability to model the similarity signal at multiple resolutions without the drawbacks of a pre-defined single spread. In existing multi-scale RBF networks a multi-scale capability might exist but is hindered when there is an overlap of receptive fields. They can still capture the signal in multiple resolutions but they lack a fundamental functionality, namely:

Distinguish and isolate signals of different resolution in overlapping receptive fields.

To better understand the significance of this, let's examine the training case of figure 5.15a. All points in the figure belong to the same dataset. In order to visually

distinguish them, the solid circle points belong to a Gaussian with a large spread and the open circles belong to a smaller spread Gaussian. If these two Gaussians would not overlap then existing approaches would be able to model them appropriately. But in this case the receptive fields overlap each other.



a. Original dataset

b. Dataset after first iteration

Figure 5.15: Gaussian overlapping problem

In order to train the network we apply the traditional *MSE* criterion. We use a variable set of spreads including the ones that formulated this statistical training set so we would expect the model to identify them. The minimum *MSE* would identify Gaussian #2 as the winner in the first iteration. This is logical since that Gaussian would minimize the *overall* error. The problem comes thereafter. After the first iteration the training dataset is corrected (basically subtract dataset's response to the first Gaussian node). The resulting dataset is shown in figure 5.15b. Two remarks can be made after the first iteration:

- Gaussian #2 is correctly modeled since all its points are absorbed providing a close to zero network output.
- BUT Gaussian #1 has lost its original form now.

The latter is a constraint that traditional RBFs could not overcome with the addition of one node. If a bias would be used that would not work because it would have to operate in the whole input space. In order to operate within a specific range the linearity of the second layer would be resolved. But even if a linear bias could be introduced it would not always be able to capture the non-linear output modification (in our figure the up or down non-linear shifting). So traditional RBFs have to use a significant number of additional nodes with relatively small widths and it would be questionable if the sufficient accuracy would be achieved.

5.5.3 Incorporate local behavior into node evaluation

To compensate for the problematic case mentioned above we perform some modifications in the node selection criteria used through the iterations. First, we introduce the influence of a local statistic in the node selection process in addition to the traditional *MSE*. Then we present a method to allow multi-scale overlapping receptive fields. We do so by blocking parts of the input space that are successfully mapped. A discussion on some related issues such as node center selection (revisited) and local density requirement concludes this section.

5.5.3.1 Customized node selection statistics

In section 5.5.2 we defined the problem. Here we provide an answer to the overlapping issue of figure 5.15a. The underlying idea is to create a technique that we will choose these two Gaussian distributions over other candidates. With this network we argue that the answer is:

Examine local behavior in addition to global MSE.

By global MSE we mean the traditional MSE, the one that results by using *all* the

points in the dataset. Local behavior reflects how well the chosen activation function fits the dataset points that are within its' receptive field.

Since a prerequisite for a RBF activation function is to be localized the corresponding receptive field within which the function is active can be easily calculated. During each candidate examination a subset of the dataset is created including only points that fall within this activation field. A local MSE is calculated using this subset according to the formula:

$$MSE_{local} = \frac{\sum_{i=1}^k (y - \hat{y})^2}{k} \quad (5.7)$$

where \hat{y} is the network response, y is the expected response from the training set and k is the number of points in the subset dataset ($k < n$) representing the points that fall within the node's receptive field.

Our next step in the investigation is to find a method that combines both local and global MSE in an appropriate manner. The selection of a node should be based on minimizing both global and local MSE s. But this minimization usually does not happen for the same activation function. An activation function with smaller spread might fit better the data locally (e.g. Gaussian #1 of figure 5.15a) while another one with larger spread/receptive field could provide a better global error reduction (e.g. Gaussian #2 of figure 5.15a).

So is there a way to combine both in one solution? The answer relies on creating a membership function to choose all possible candidates. This function is presented in figure 5.16. The X axis is the local MSE for all the candidate functions and the Y axis represents their global MSE . By mapping functions on the $[MSE_{local}, MSE_{global}]$ 2D space

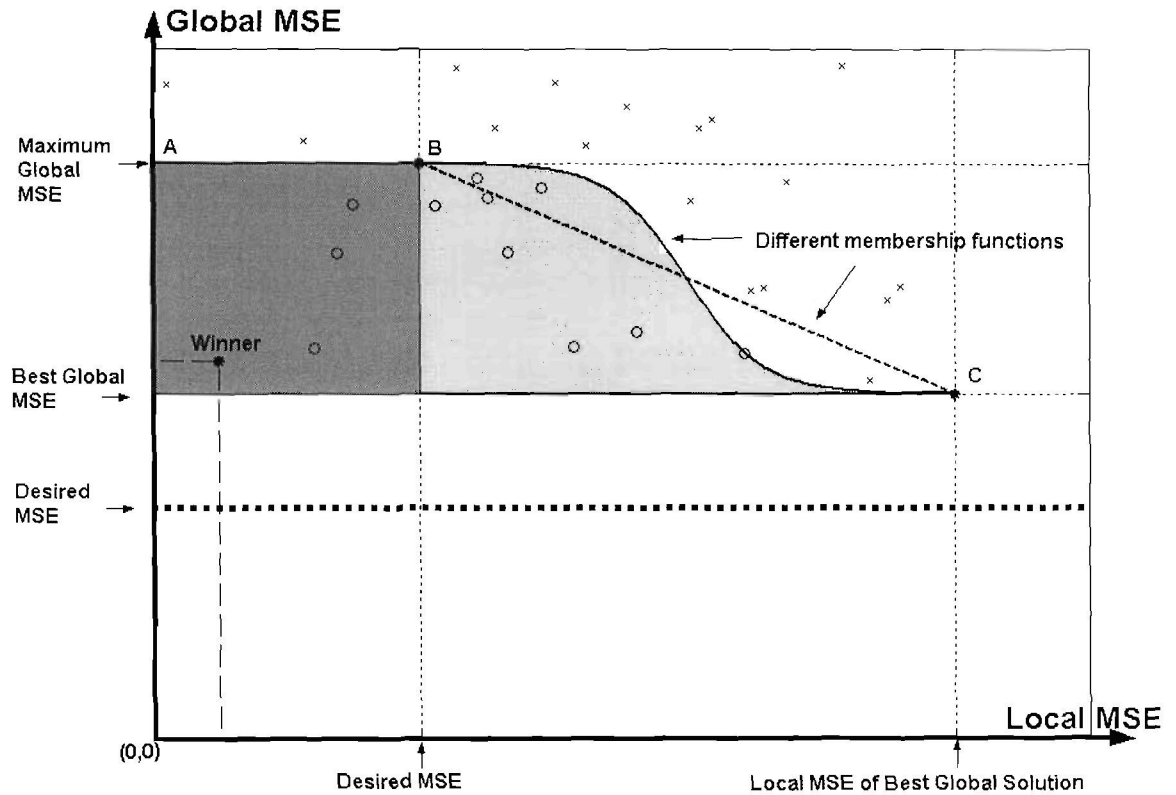


Figure 5.16: Combination of global and local MSE

we allow a ranking process based on these two values. There are four parameters that are essential to define the graph:

- i. **Best global MSE:** All the MSE_{global} values are calculated and ranked. The one with the minimum value is assigned here.
- ii. **Local MSE of best global solution:** This value corresponds to the MSE_{local} for the candidate function that provided the minimum MSE_{global} .
- iii. **Desired MSE:** This is the cut-off, the threshold value for the MSE_{global} . If this value is reached then the iterations stop because the desired accuracy is achieved.

- iv. **Maximum Global MSE:** This value expresses the maximum MSE_{global} allowed for a specific iteration number. It is calculated based on the membership function of figure 5.17. The equation for the membership function is the following:

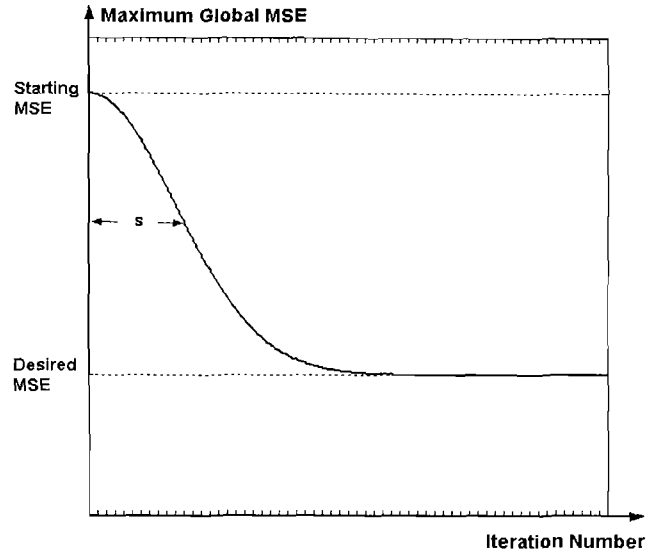


Figure 5.17: Maximum global MSE

$$MSE_{global_max}(iter) = MSE_{Desired} + \left[(MSE_{Starting} - MSE_{Desired}) * \exp\left\{-\frac{iter^2}{2\sigma_g^2}\right\} \right] \quad (5.8)$$

where $MSE_{Starting}$ is the MSE_{global} before the iterations. The role of this parameter is to ensure that if the best MSE_{global} candidate is not chosen, the selected activation function will capture a significant amount of global error. In other words, it expresses how flexible we can be between the chosen and the minimum value of the MSE_{global} . This flexibility decreases though as we progress through the iterations. The decrease rate is presented through the spread σ_g of equation 5.8. Larger σ_g allows potential acceptance of more activation functions with better local than global performance. On the other hand if that value is too high there is the risk of not achieving the desired final global accuracy. So there is a trade-off in the

definition of σ_g . A suggested value is $\sigma_g = \text{maximum_nodes_allowed} / (2 * 3)$. This way we allow potential candidates of high local accuracy to be included but only up to half the total number of total iterations (nodes). Then the analysis will only include MSE_{global} criterion to ensure high final accuracy.

After these four parameters are defined points A, B and C can be represented on the graph. We define a membership function $Q(\bullet)$ that connects these points. This membership function restricts the acceptable solutions by establishing a correlation between local and global MSE . It is given by the equation:

$$Q(MSE_{local}) = \left\{ \begin{array}{l} MSE_{global}^{max} \\ \quad \text{if } MSE_{local} \in [0, MSE_{Desired}] \\ \\ MSE_{global}^{best} + [(MSE_{global}^{max} - MSE_{global}^{best}) / \{1 + \exp[-a_Q(MSE_{local} - c_Q)]\}] \\ \quad \text{if } MSE_{local} \in (MSE_{Desired}, MSE_{local}^{of_best_global}) \\ \\ 0 \\ \quad \text{if } MSE_{local} \in [MSE_{local}^{of_best_global}, +\infty) \end{array} \right\} \quad (5.9)$$

All the candidate points have a calculated MSE_{global} . If their MSE_{global} is smaller than the membership $Q(MSE_{local})$ based on their MSE_{local} value then these points are accepted (points with a circle on the graph). Otherwise they are rejected (points represented with a cross). If no accepted points are found then the solution with the best MSE_{global} remains. Also, if the MSE_{local} value of the best MSE_{global} is smaller than the $MSE_{Desired}$ then it automatically accepted without going through this process. From all the accepted points (m) the one with the minimum MSE_{local} value is the winner. Formally:

$$MSE_{global}^{winner} \in [MSE_{global}^{i=1}, \dots, MSE_{global}^{i=m}] \quad , \quad \text{where } MSE_{global}^{i=[1..m]} \text{ satisfies the condition:}$$

$$MSE_{global}^{i=\{1...m\}} < Q(MSE_{local}^{i=\{1...m\}}) \quad \text{and the winner is:} \quad MSE_{local}^{winner} = \min[MSE_{local}^{i=1}, \dots, MSE_{local}^{i=m}].$$

Let's examine the reasoning behind each of the three segments of $Q(\bullet)$. Keep in mind that for a candidate to be accepted its' global error (i.e. the Y axis value) should be smaller than $Q(\bullet)$.

- $MSE_{local} \in [0, MSE_{Desired}]$. In this segment we chose a linear constant function. This way we accept any candidates that have local accuracy better than the desired one and global accuracy better than the one allowed based on the iteration number (fig. 5.16, dark shaded area). This makes sense because any candidate with really good local behavior should be included as long as it contributes acceptably in the global error minimization.

- $MSE_{local} \in (MSE_{Desired}, MSE_{local}^{of_best_global})$. For this segment we use a transition function from point B to C. This function expresses the *rate* at which we are willing to sacrifice global accuracy for a better local solution. The magnitude is defined by the maximum MSE_{global} allowed as calculated earlier. The last constant together with the $MSE_{Desired}$ provide a scaling on the global MSE axis. In the X axis, the local MSE one, scaling is adjusted based on the MSE_{local} value of the best MSE_{global} solution and the $MSE_{Desired}$ value. In this example we chose a Sigmoidal transition function. Constants a_Q and c_Q are predefined. Constant a_Q expresses the slope of change. Constant c_Q makes sure that the Sigmoidal response in the middle of the segment is half between points B and C. The selected points range can be seen in fig. 5.16 with a light shaded area. Alternatively a different transition function can be used if another rate of change is preferred (e.g. linear).

- $MSE_{local} \in [MSE_{local}^{of_best_global}, +\infty)$. This value is set to zero because we do not want to accept any candidates that have a worse MSE_{local} than the one of the initially chosen candidate with the best MSE_{global} .

5.5.3.2 Blocking successfully mapped neighborhoods

In the previous section we introduced a training criterion that would allow the system to choose a better local-fitting activation function (fig. 5.15a Gaussian #1) with good MSE_{global} over one with a better MSE_{global} but not so good MSE_{local} . But this criterion alone does not make sure that Gaussian #2 of fig 5.15a will be eventually chosen. Let's examine again the same example. But assume that our local criterion is applied and the Gaussian with a smaller spread is selected. In the next iteration all the original points of Gaussian #1 (open circle on the graph) will be downscaled to almost zero. The new dataset for the next iteration will have the form of figure 5.18 with the solid circle points. If we would then apply any MSE criterion (traditional or ours) the Gaussian with a solid line (fig. 5.18) instead of Gaussian #2 would be selected. This happens because the corrected points of Gaussian #1 remain in the solution influencing the result.

To overcome this we introduce the notion of *blocking functions*. These functions are applied on the input space of the RBF network when the previously chosen activation function has a $MSE_{local} < MSE_{Desired}$. By doing so we exclude local (small-scale) signals that are sufficiently captured. This way *we allow the multi-scale network to unfold over a larger input space neighborhood without being misguided by localized noise*. If we apply a blocking function in figure 5.19 we exclude points in the shaded area so in the next iteration the correct large-scale Gaussian is selected. Another advantage of this exclusion is that the original dataset is getting smaller speeding up the process for coming iterations.

The blocking function can be a binary one or a close to binary. For our application we chose an inverted symmetrical Sigmoidal function $\mathbf{B}(\bullet)$ given by the equation:

$$B_j(\vec{x}) = 1 - \left[1 / 1 + \exp \left\{ -a_B [abs \|\vec{x} - \vec{\mu}_j\| - c_B] \right\} \right] \quad (5.10)$$

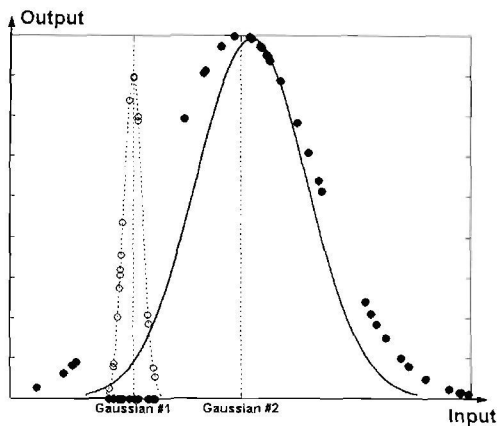


Figure 5.18: Chosen activation function after first iteration selected a local-fitting one

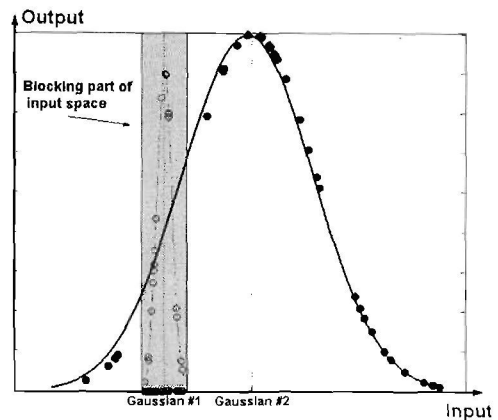


Figure 5.19: Chosen activation function after blocking part of input space

for node j . \vec{x} is the d -dimensional input vector with elements x_i and $\vec{\mu}_j$ is the vector determining the center of function's symmetry. Parameter c_B defines the spread and parameter a_B the slope (fig. 5.20). They are both directly related to the receptive field of

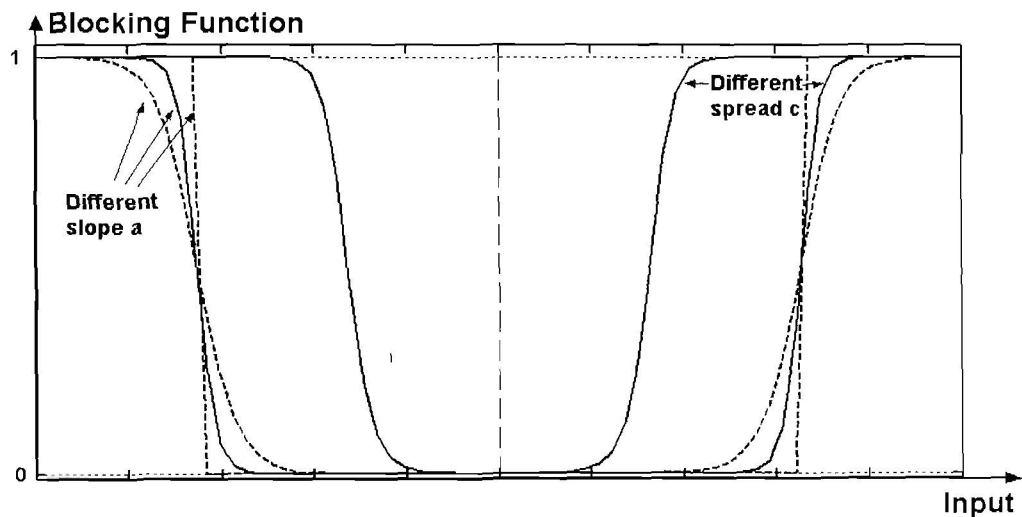


Figure 5.20: Using an inverted symmetrical Sigmoidal as a blocking function the activation function they are blocking. In the case of a Gaussian activation function spread c_B should be set approximately to $(3 * \text{Gaussian_spread})$ while in a case of a Sigmoidal activation function, c_B can have the same value as the spread of the activation Sigmoidal. Slope a_B should be set to a relatively small value to lead to a close binary

function. We do not want the blocking function to expand further than the receptive field of the activation one so attention should be showed. Also parameter a_B will effect the signal output smoothness so a too small value is not recommended either.

If we would follow the same notion as the one of equation 5.10. we can apply the above blocking function to a 2D input. An example of three such functions (A,B,C) is shown in figure 5.21.

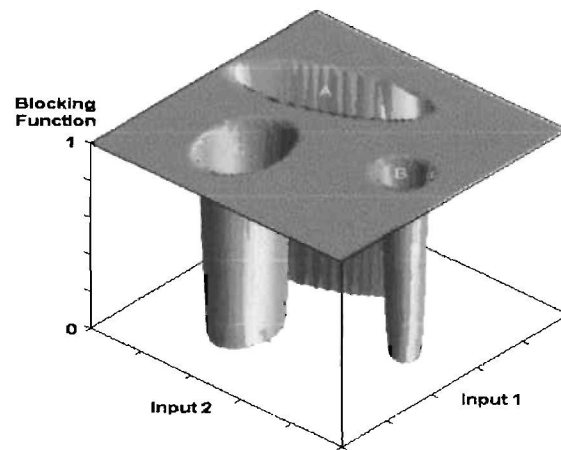


Figure 5.21: 3D blocking example

5.5.3.3 Keeping blocked points as center candidates

While an evaluation of the algorithm was taking place we noticed a drawback of the local-based solution. Localized functions might be able to reveal a global function by excluding local noise, but at the same time might degrade its signal so it is not detectable any more. This is illustrated in figure 5.22. The training set is represented with solid points. During our localized solution the algorithm chooses Gaussian #2 because the two neighboring points are not close enough to be in the receptive field and introduce errors. In addition to that the noise of Gaussian #1 is not strong enough to be chosen since it is represented by a single point.

So what would happen is that point A would be eliminated from the dataset by a blocking function. This way Gaussian #3 would not have the opportunity to be selected in the next iteration since its' center would be out. This can be corrected by eliminating points from solution but still use them as center candidates for the nodes to follow. At the end Gaussians #2 and #3 would coexist in the final solution. A simple node elimination process is triggered and the redundancy of Gaussian #2 will be picked up and substituted just by #3.

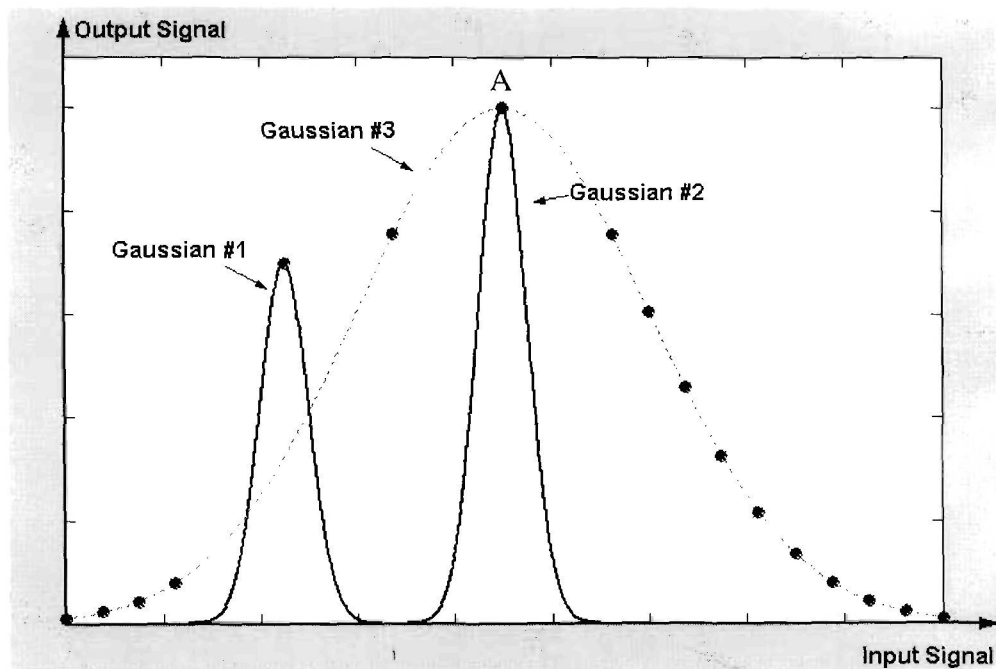


Figure 5.22: Degrading signal problem

5.5.3.4 Checking for local density

In previous sections we mentioned that an important characteristic of the MSRBF should be idleness unless sufficient evidence exists. The problem of using a multi-scale approach is that sometimes low-error points cause undesired extensive spread choice for the activation functions. Such an example is seen in figure 5.23. We have two input points and we try to model them with our RBF. If the maximum spread is large enough then the

network will pick the Gaussian with the larger spread (dashed gray line). That contradicts though the purpose of the MSRBF, since there is not sufficient evidence to justify such a large spread of the receptive field. Instead, we would rather have two nodes with much smaller spread (solid black lines).

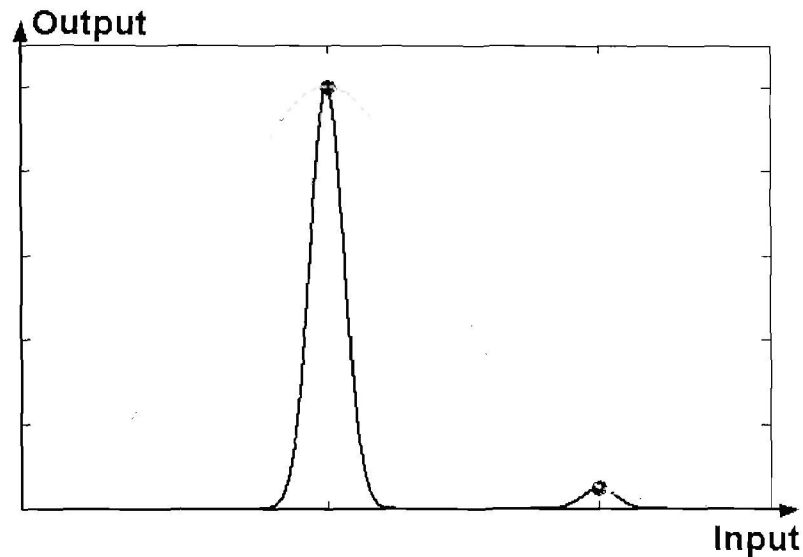


Figure 5.23: Local distribution problem

In order to assure that Gaussians with large receptive fields are only employed where necessary, we analyze the points that are included within the receptive field, the same points that we used to calculate the MSE_{local} . We establish buffer zones at the output values and count the points within that (fig. 5.24). When designing the MSRBF, the number of zones, their output threshold values, and the minimum number of points per zone should be defined. These parameters can also be spread-specific, to allow for example more strict criteria in large-spread nodes. Then this density criterion returns a positive response only if some (or all) of the zones restrictions are met. By doing so the density criterion makes sure that only when satisfied the candidate Gaussian is chosen.

For example, in figure 5.24 we have 3 zones. Their output threshold values (relative portion of the Gaussian amplitude) are $[0.8, 1]$ for zone #1, $(0.2, 0.8)$ for zone #2,

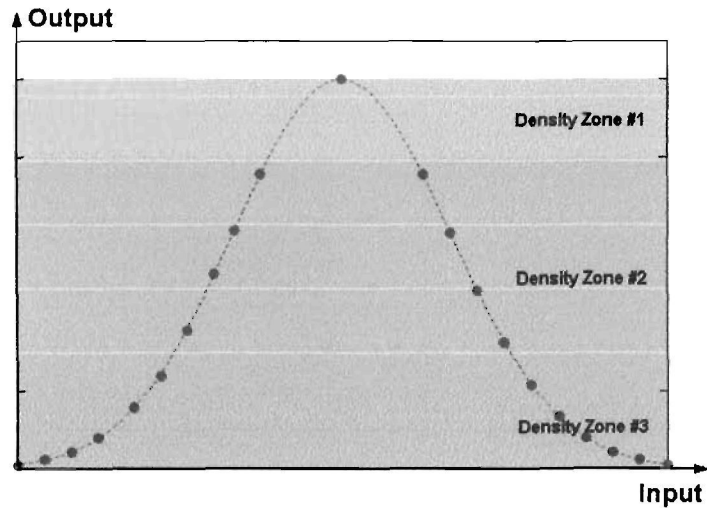


Figure 5.24: Density zones at the output

and $[0, 0.2]$ for zone #3. If we assume a point count per zone as $[2, 2, 2]$ for zones 1, 2, and 3 respectively the density criterion would fail because there is only one point in zone #1, when a minimum of two is required. If the point count per zone would be $[1, 3, 2]$ then we would have a successful density result.

This approach has the advantage that it is independent of the input dimensionality because the analysis is performed at the output level. Lack of analysis at the input level is highly desirable to achieve fast processing times. At the same time though there are some inherent restrictions in our method because we do not examine spatial distribution of points within zones. For example, there are no warranties that symmetry will exist. In figure 5.24 all points could have been at the left of the Gaussian's center and still give a successful density result. However, the trade-off in computational speed combined with our localized metrics for node evaluation compensate for these undesired cases.

5.6 Mathematical formulation

In the general case, an RBF network presents a mapping from an m -dimensional input space to a single-dimensional output space. If we assign s as the mapping function it can be described as:

$$s: \mathcal{R}^m \rightarrow \mathcal{R}^1 \quad (5.11)$$

Mapping s results in a hypersurface of dimensionality $m+1$.

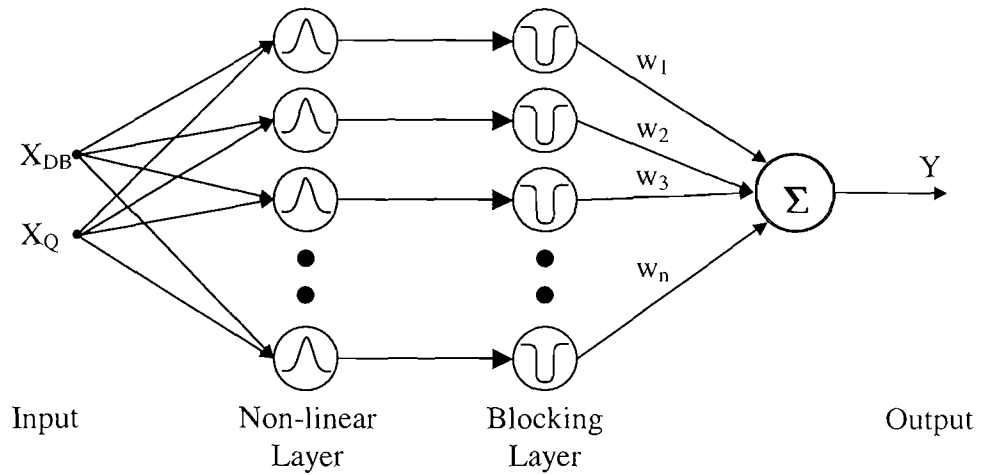


Figure 5.25: Multi-scale RBF network architecture

Because of the addition of blocking functions a change in the traditional network architecture takes place by adding another layer (fig. 5.25). This layer acts as a filtering to make sure that successfully mapped neighborhoods are excluded from later node influence. Each activation function is connected with the corresponding blocking function in a linear unweighted manner. Then the outcome of the blocking layer is weighted and the summation provides the network output.

For the general case let n be the number of data in the training set and k the final number of selected nodes. The value of k is usually required to be smaller than n for storage purposes. In this case the network does not have an exact solution but some

optimization technique should be used. One major advantage of RBF networks is the linear dependence of the network mapping function on the weight layer. This means that we can use a simple least-squares solution to calculate the weights without worrying about non-linear optimization and going through iterations.

The following matrices are involved in the least squares solution:

Matrix Φ ($n \times k$): It contains the response of the nodes to the training dataset. Each column describes the response of a specific node to the dataset. This means that the function remains the same for each column element, just the input changes. On the other hand each row presents the response of a specific training input to all the nodes. The response is calculated by applying the same input to the corresponding activation function for that node.

$$\Phi(n \times k) = \begin{bmatrix} \Phi_1^1 & \dots & \Phi_1^k \\ \vdots & \ddots & \vdots \\ \Phi_n^1 & \dots & \Phi_n^k \end{bmatrix} \quad (5.12)$$

Matrix \mathbf{B} ($n \times k$): This matrix presence allows or prevents the signal of a node to propagate in the final solution depending on the input pattern position. It shows the response of the input to the blocking functions used in the training process. Values of 1 allow propagation while 0 values do not. Each row corresponds to the blocking of a specific pattern to all nodes. Each column reflects the result of a specific blocking function to the input set.

$$\mathbf{B}(n \times k) = \begin{bmatrix} 1 & B_1^2 & \dots & B_1^k \\ \vdots & \vdots & \ddots & \vdots \\ 1 & B_n^2 & \dots & B_n^k \end{bmatrix} \quad (5.13)$$

It is important to clarify the purpose of a blocking function, which is to “secure” that

neighborhood for the next iterations. So a blocking function that is caused by node j , will not have any effect on itself but will interfere with all the next nodes $\{j+1, \dots, k\}$. Now if another blocking function is necessary for example caused by node $j+3$, then it should apply for nodes $\{j+4, \dots, k\}$. But at the same time the first blocking function is active. The form of the chosen functions is such that their accumulative action is calculated by multiplication. If we formalize multiple blocking functions for node r we have:

$$B_r(\bar{x}) = \prod_1^{j=r-1} B_j(\bar{x}) \quad (5.14)$$

with $B_j(\bar{x}) = 1 - \left[1 / 1 + \exp\{-a_B[\text{abs}\|\bar{x} - \bar{\mu}_j\| - c_B]\} \right]$ and \bar{x} is the input vector, and $\bar{\mu}_j, a_B, c_B$ are predefined parameters. If node j does not trigger blocking then we assign $B_j(\bar{x}) = 1$.

Matrix \mathbf{H} ($n \times k$): The matrix \mathbf{H} represents the filtered response to the nodes of the dataset (after blocking). It is calculated by an element-by-element multiplication of matrices Φ and \mathbf{B} .

Matrix \mathbf{P} ($n \times n$): The elements of this diagonal matrix express users confidence on the dataset they provided. Each element corresponds to a specific set of input and output values. A more detailed explanation of this confidence within the provided training set was provided in section 4.2.3.

$$\mathbf{P}(n \times n) = \begin{bmatrix} P_1 & \dots & 0 \\ \cdot & \cdot & \cdot \\ 0 & \dots & P_n \end{bmatrix} \quad (5.15)$$

Matrix \mathbf{Y} ($n \times 1$): Matrix \mathbf{Y} has the output values of the training set of the network.

$$\mathbf{Y}(n \times 1) = \begin{bmatrix} y_1 \\ \cdot \\ y_n \end{bmatrix} \quad (5.16)$$

Matrix $\mathbf{W}(k \times 1)$: This is the unknown matrix, the one containing the weights that we are solving for:

$$\mathbf{W}(k \times 1) = \begin{bmatrix} w_1 \\ . \\ w_k \end{bmatrix} \quad (5.17)$$

The solution to the problem is given by a least-squares minimization and the equation:

$$\mathbf{W} = (\mathbf{H}^T \mathbf{P} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{P} \mathbf{Y} \quad (5.18)$$

To avoid singularity problems due to the possible ill-conditioning of matrix \mathbf{H} the above equation is solved using singular value decomposition. Thus, we can see that the weights can be found by fast, linear matrix inversion techniques.

After the network weights are found a pruning method takes place. Every weight is set to zero and change in the error is evaluated (Reed, 1993). In some cases the other nodes will be able to pick up a specific nodes' contribution. Because of our network architecture this might happen if a large-scale signal is degraded by local solutions (fig. 5.22). Network pruning will eliminate localized nodes if the large-scale nodes can pick up their involvement and reveal the original signal.

5.7 Network pseudo code

In this section we provide the process flow for the MSRBF. A more detailed discussion on network parameters is presented in the training and evaluation part of chapter 7.

Network Inputs:
$[X_{Query}, X_{Database}, Similarity]$ // 2D input pattern, output vector
Network Parameters:
$[\min(\sigma_a), \max(\sigma_a)]$ // Gaussian Activation Functions (GAF)
$[\min(c_a), \max(c_a), a_a]$ // Sigmoidal Activation Functions (SAF)
[Total number of GAF, Total number of SAF]
$[a_Q, c_Q]$ // MSE local/global weight transfer function parameters
$[a_B, c_B]$ // Blocking function parameters
[Zone_output_limits, Minimum_points_per_zone] // Density criterion
[MSE _{Desired}] // Stopping criterion
[Min_Nodes, Nodes_Step, Max_Nodes] // Number of Nodes
$[\text{Min}_\sigma, \sigma_{Step}, \sigma_{Max}]$ // Maximum MSE global parameters

Table 5.3: MSRBF inputs and parameters

Our network training process can be seen in table 5.4. The process will create not a single RBF like the traditional method but a set of candidate RBF networks. The one that shows the best generalization based on a testing dataset will be chosen. The number of created RBF networks is pxd , where p is the number of candidate nodes, and d is the number of candidate spreads for the Maximum MSE global calculation.

Network Training:

```
[ $\sigma_a$ ] = calculate_GAF_spreads [min( $\sigma_a$ ), max( $\sigma_a$ ), Total number of GAF]
[ $c_a$ ] = calculate_SAF_spreads [min( $c_a$ ), max( $c_a$ ), Total number of SAF]
FOR i=[Min_Nodes, Nodes_Step, Max_Nodes] // Number of Nodes
    max_number_of_nodes_allowed = i
    FOR k=[Min_ $\sigma_g$ ,  $\sigma_g$ _Step,  $\sigma_g$ _Max] // Maximum MSE global parameters
        current_ $\sigma_g$  = k
        FOR j=[1,2,..., max_number_of_nodes_allowed]
            Calculate MSEglobal for every GAF and SAF
            IF min(MSEglobal) < MSEDesired then add node and exit loop ENDIF
            Calculate maximum MSEglobal allowed using current_ $\sigma_g$ 
            Select GAFs and SAFs with MSEglobal < maximum MSEglobal allowed
            Calculate MSElocal for selected AFs
            Compute Best_candidate GAF or SAF using MSE local/global transfer
            function Q(•)
            Check for local density of Best_candidate
            IF failed choose next best Q(•) winner, else Best_candidate = winner, ENDIF
            Simulate training set with winner AF and correct output vector y
            IF winner's MSElocal < MSEDesired //initiate blocking
                Add corresponding blocking function Bj(•)
                Eliminate training points within winner's receptive field from training set
            ENDIF
        END
    END
END
```

Table 5.4: MSRBF Network training process

5.8 Conclusion

In this chapter we concentrated on improving the accuracy of our fuzzy method by using a local-nature artificial neural network, the Radial Basis Function one. Our Multi-Scale RBF is modified accordingly to capture the errors of the fuzzy functions. There are two important characteristics in our network:

- Extensive multi-scale modeling capability, and
- Constrained application to highly active parts of input space.

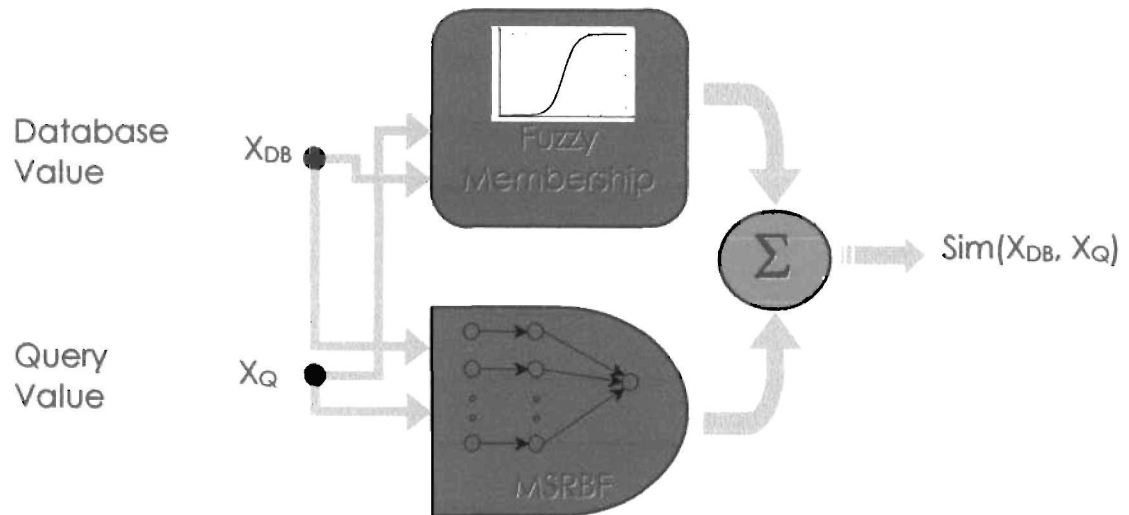
The following table summarizes the modifications made and their result in order to adapt to the above two requirements:

Adjustment from traditional RBF	Adjustment effect
Variable spread for activation functions	⇒ Support multi-scale analysis
Symmetric Sigmoidal activation functions	⇒ Introduction of localized bias
Customized training criteria that combine local (within receptive field) and global fitting accuracy metrics. <p style="text-align: center;">+</p> Blocking functions over successfully mapped regions <p style="text-align: center;">+</p> Additional layer in network architecture	⇒ Capture small-scale signals first which exposes large-scale signals that would otherwise remain hidden
Efficient training point elimination from the training set when successfully modeled	⇒ Computational gain
Density check metrics	⇒ Minimization of receptive field to high stimulus areas.

Table 5.5: Summary of the modifications on the traditional RBF

Chapter 6

Combining fuzzy functions and neural network into a global solution



Previous chapters discussed individually the fuzzy membership function and the MSRBF neural network. We also investigated the characteristics of our MSRBF while keeping in mind their combined result. Here we make our case for an optimization technique for the fuzzy membership function based on MSRBF properties. Their combined operation and corresponding mathematical solution are presented.

6.1 Self-organization of global similarity functions

Experiments on the fuzzy membership function (FMF) showed that the identification of the parameters in the solution is influenced by outliers. An example is shown in figure 6.1. After the FMF training is finalized the chosen sigmoidal function is calculated (solid line). This is the best solution in a statistical sense, based on least squares minimization. In the next step we take the residuals of that solution and try to capture them with the MSRBF (fig. 6.2 solid line). Unfortunately original outliers cause the FMF to adjust to them and introduce high residuals.

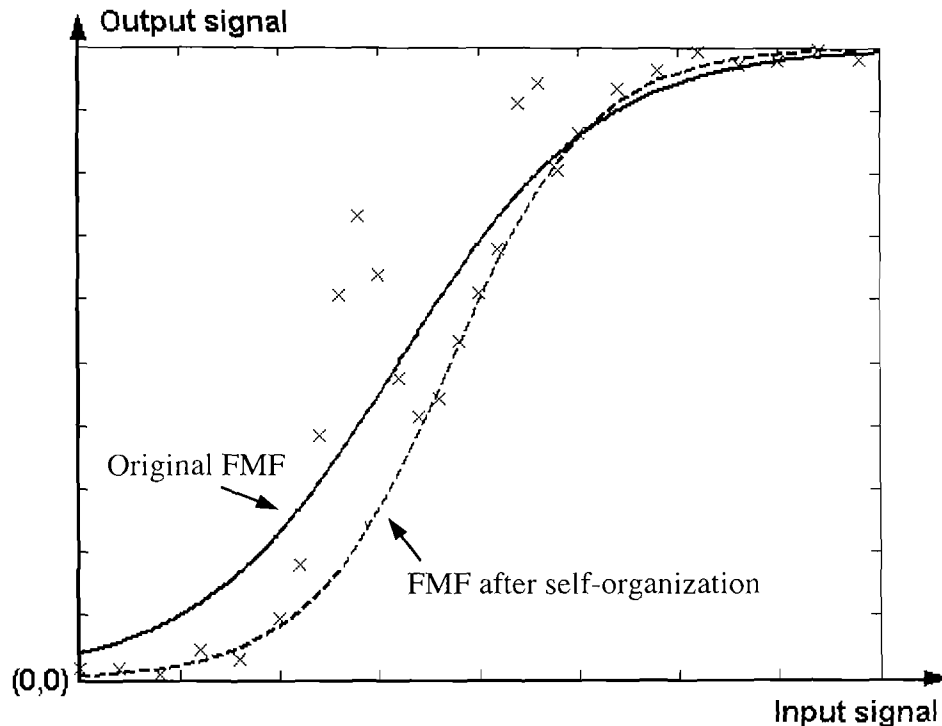


Figure 6.1: Outlier effect on fuzzy membership function

To correct this, we introduce an additional step in the training process for the FMF. The goal of this process is to uncover the dominant signal, even if this means a higher final error (MSE). The underlying idea is that whatever causes the additional error

would be an outlier that should be mapped with the MSRBF locally and not influence the whole input space. This assumption of course requires a successful initial training of the FMF, in other words we expect the FMF to have a reasonable interpolation error.

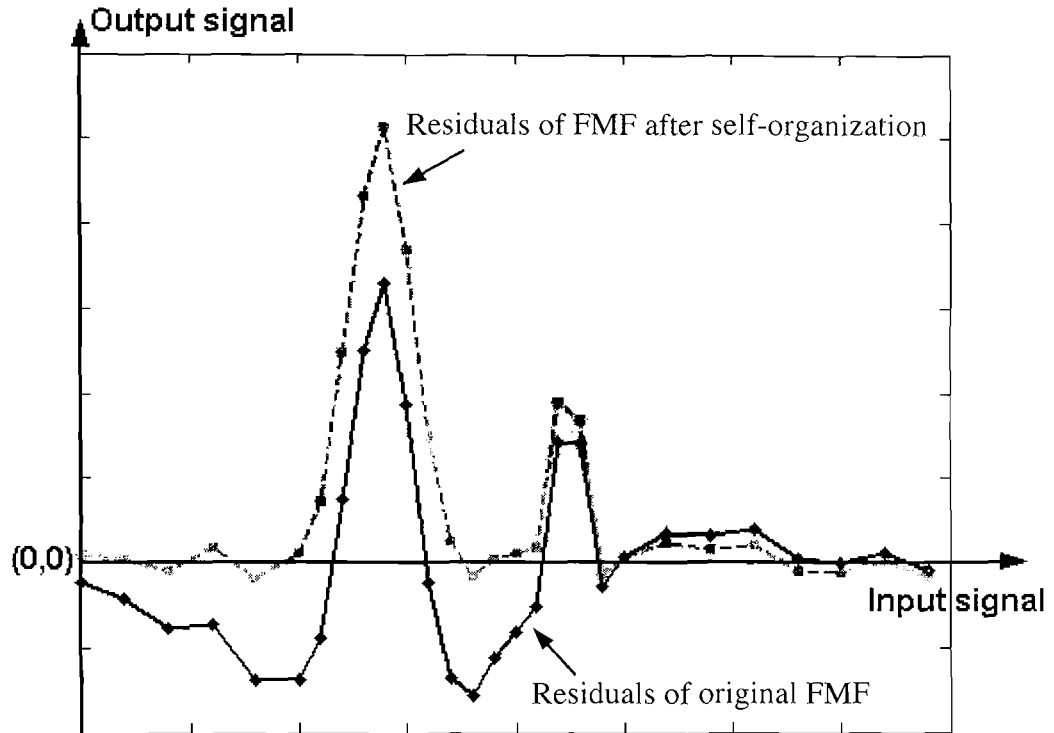


Figure 6.2: Residuals of original and weighted-distance fuzzy functions

The additional training is done the same way as the initial training with one difference. A weight manipulation is used to enforce the desired fitting. Assuming that points representing the majority signal will have a smaller output (similarity) error than possible outliers we adjust the weights of each point by using a linear and then gaussian transfer function (fig. 6.3). The transfer function is:

$$W_{FMF}(d) = \begin{cases} 1 & \text{if } d \leq \sqrt{MSE_{Desired}} \\ \exp\left\{-\frac{\|d - \sqrt{MSE_{Desired}}\|^2}{2\sigma_{wFMF}^2}\right\} & \text{if } d > \sqrt{MSE_{Desired}} \end{cases} \quad (6.1)$$

where d is the output error of the point at the previous iteration and W_{FMF} is the weight of that point in the current iteration. Variable σ_{wFMF} specifies the width of the gaussian function used. It is usually assigned the value of

$$\sigma_{wFMF} = (\text{Maximum Error} - (MSE_{Desired})^{0.5})/3 \quad (6.2)$$

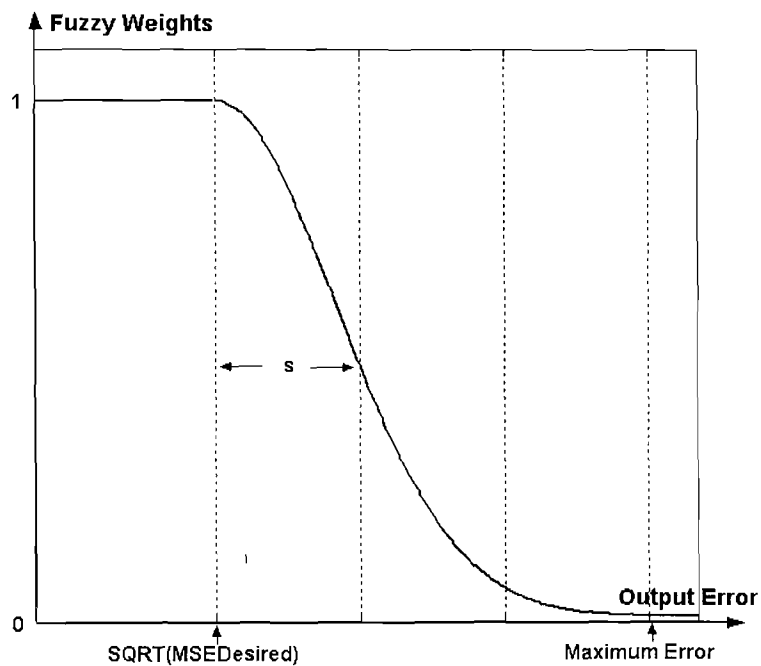


Figure 6.3: Weights manipulation

This way based on the maximum distance of outliers the high weights interval is stretched/shirked. The reason for having the linear segment for $d \leq \sqrt{MSE_{Desired}}$ relates to the expected result accuracy. If points are within that interval they should all have the maximum possible contribution in the solution.

The example of figure 6.1 is revisited. After the additional weights solution the resulting FMF is presented with a dashed line. There we can see that the overwhelming signal is captured leading to the conclusion that:

“the weight manipulation acts as a self-organizing optimization technique to the overwhelming signal.”

If we examine the residuals of the new FMF (fig 6.2 dashed line) we can comment that:

- The total MSE is higher.
- BUT the residuals are now easily captured by our MSRBF than before the self-organization technique.

So the combined effect of the FMF and the MSRBF can be facilitated better at times if a self-organization method is applied in the FMF before the MSRBF training starts.

6.2 Architecture

In figure 6.4 the final architecture of our system is presented. In addition to the custom neural network structure (as presented in the previous chapter) we have the fuzzy functions as an additional node added at the top. Two important characteristics of this node are:

- No blocking nodes interfere with the output of the fuzzy function. This is consistent with our motivation to have the fuzzy function carry the signal throughout the input space and we expect good generalization for it.
- The weight of the fuzzy function in the summation node is one. We do so because a choice of a different weight would scale the function in intervals smaller or larger than the desired similarity output $[0,1]$.

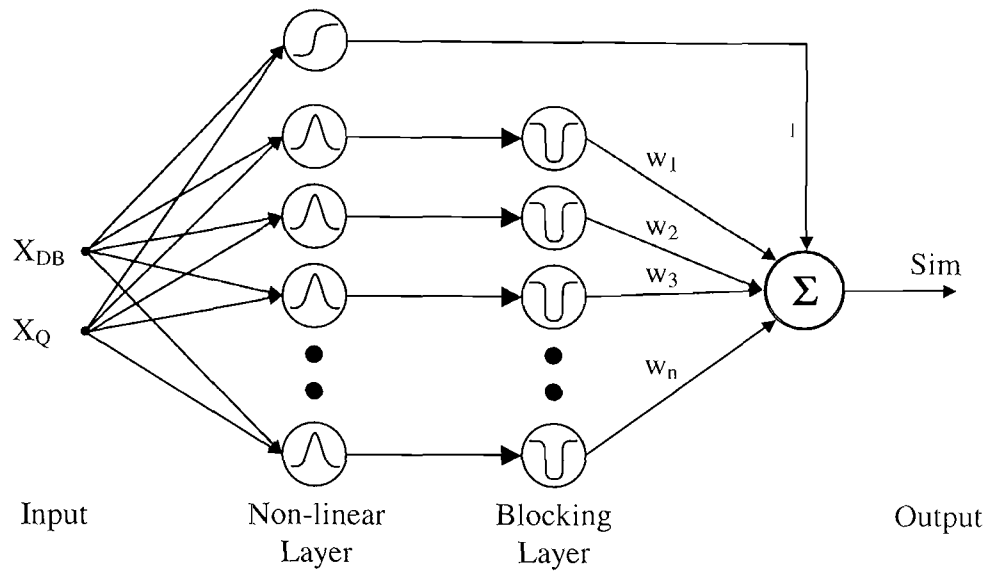


Figure 6.4: Neuro-fuzzy network architecture

After the establishment of the system parameters the information flow in our system is as follows:

- i. The two inputs of the process are a Database Value with a corresponding Query Value of a one-dimensional object attribute.
- ii. The inputs propagate to the non-linear layer. One node output is calculated from the fuzzy function and a response is also returned from every hidden layer of the neural network.
- iii. The outputs of the neural network hidden layer go through the blocking layer. Depending on the training some signals are down-scaled while others pass freely.
- iv. The outputs of the blocking layer together with the fuzzy function response are accumulated through a weighted summation at the last layer.
- v. The weighted summation is returned as the calculated similarity value between the two input values.

6.3 Mathematical solution

The training of the algorithm follows a progressive approach as higher complexity is identified. At the first stage fuzzy functions are interpolated to capture the similarity preference. Then, if necessary, our MSRBF network is used to increase accuracy. At this section we describe the last training stage that is triggered whenever the neural network is incorporated. The training, in this case, involves calculation of the parameters of the fuzzy functions and the neural network within the same solution.

The solution to the problem is given by a least-squares minimization and the equation:

$$\mathbf{W}=(\mathbf{C}^T\mathbf{P}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{P}\mathbf{Y} \quad (6.3)$$

For the general case let:

n be the number of data in the training set,

r the number of parameters describing the fuzzy function , and

k the final number of selected nodes in our MSRBF.

The matrices in equation 6.3 are formulated as follows:

Matrix \mathbf{P} ($n \times n$): This diagonal matrix expresses users confidence on the dataset. Each element corresponds to a specific set of input and output values. A more detailed explanation was provided in section 4.2.3.

$$\mathbf{P}(n \times n) = \begin{bmatrix} P_1 & \dots & 0 \\ \cdot & \cdot & \cdot \\ 0 & \dots & P_n \end{bmatrix} \quad (6.4)$$

Matrix \mathbf{Y} ($n \times 1$): Matrix \mathbf{Y} has the output values of the training set of the network.

$$\mathbf{Y}(nx1) = \begin{bmatrix} y_1 \\ \cdot \\ y_n \end{bmatrix} \quad (6.5)$$

Matrix $\mathbf{W}((r+k) \times 1)$: This is the unknown matrix, the one containing the parameters we are solving for:

$$\mathbf{W}((r+k) \times 1) = \begin{bmatrix} w_1 \\ \cdot \\ w_r \\ \cdot \\ w_{r+k} \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{Fuzzy} \\ \mathbf{W}_{MSRBF} \end{bmatrix} \quad (6.6)$$

Parameters for the fuzzy function are:

$$\mathbf{W}_{Fuzzy} = \{w_1, w_2, \dots, w_r\} \quad (6.7)$$

Parameters for the MSRBF neural network are:

$$\mathbf{W}_{MSRBF} = \{w_{r+1}, w_{r+2}, \dots, w_{r+k}\} \quad (6.8)$$

Matrix $\mathbf{C}((r+k) \times n)$: This matrix contains the derivatives of the unknowns. For simplicity let us decompose matrix \mathbf{C} to two matrices \mathbf{C}_{Fuzzy} and \mathbf{C}_{MSRBF} , where:

$$\mathbf{C} = [\mathbf{C}_{Fuzzy} \quad \mathbf{C}_{MSRBF}] \quad (6.9)$$

Matrix \mathbf{C}_{Fuzzy} expresses the derivatives of the parameters associated with the fuzzy function and has size $r \times n$. An explanation of the formulation of this matrix was described in chapter 4. A specific example of how this matrix gets the values assigned for a sigmoidal function is presented in equations 4.13 through 4.17. The corresponding derivative matrix in these equations is matrix \mathbf{A} .

Matrix \mathbf{C}_{MSRBF} contains the derivatives of the neural network solution with size $k \times n$. This matrix was investigated in detail in chapter 5. It is the product of an element-by-element multiplication of two matrices, the Φ and \mathbf{B} ones. Matrix Φ contains the

response of the nodes to the training dataset and an example of its formulation is given by equation 5.15. Matrix \mathbf{B} is a new matrix we have added to control whether or not the signal of a node propagates in the final solution depending on the input pattern position. It shows the response of the input to the blocking functions used in the training process. Equation 5.16 demonstrates the contents of such a matrix.

To avoid singularity problems due to the possible ill-conditioning of matrix \mathbf{C} equation 6.3 is solved using singular value decomposition. We should also mention that this solution is not always a linear one. Matrix $\mathbf{C}_{\text{MSRBF}}$ does not require any temporary values for the unknowns since there is a linear correlation of the MSRBF parameters and the overall solution. On the other hand $\mathbf{C}_{\text{Fuzzy}}$ matrix contains the derivatives of the fuzzy functions. If the planar solution gives acceptable results then the overall solution will be linear. But if more complex non-linear functions are used (e.g. sigmoidal) then formulation of matrix $\mathbf{C}_{\text{Fuzzy}}$ would require temporary values for the unknowns. In this case the solution would have to be an iterative non-linear one, but as we showed we resort into this more computationally expensive solution if more simple ones fail to capture the complexity of the similarity preference.

6.4 Conclusion

In this chapter we showed how to combine the fuzzy approach of chapter 4 with the MSRBF of chapter 5 creating a neuro-fuzzy system. We presented an optimization technique for the fuzzy functions through a self-organization process that adjusts to the overwhelming signal. This way our neural network can adapt easier to the remaining signal. We also showed how information propagates in our feed-forward neuro-fuzzy network. Using the fuzzy membership function the two inputs create an output describing

the global bias/overwhelming signal. Simultaneously, the same inputs produce a localized correction to the signal as they propagate through the MSRBF. At the final stage, both signals, the large-scale and the local one, are combined with a summation to provide the overall similarity result. The mathematical formulation for the system solution during training was also provided.

Our system incorporates high modeling capabilities using the combined neuro-fuzzy method because of an important semantic relationship between the system design and the similarity preference under investigation, which dictates that:

- Expected behavior is modeled by a global fuzzy membership function whose complexity grows as the problem.
- Unexpected behavior is captured using the MSRBF in a localized fashion.

Chapter 7

Training and system evaluation

This chapter provides evidence of the benefits produced by our approach. It will help proving our hypothesis that our neuro-fuzzy system returns more accurate results than the existing method used in geospatial queries, the distance-based nearest neighbor (NN). We should mention again that by design our system outperforms the distance-based nearest neighbor, since our system recognizes it as a subcase. The combination of the non-linear fuzzy functions with the neural network provides advanced modeling capabilities compared to the NN, something that results from the mathematical equations used and the fact that neural networks are universal approximators, meaning that they can describe any function to arbitrary accuracy. Furthermore, our system has the capability to recognize distance-based similarity preference and model it without proceeding to more complicated processes as shown in chapter 4 and further discussed later in this chapter.

Our focus here is twofold, namely to present our system's training and perform a system evaluation. First, we show how the training process takes place and demonstrate the idea behind the progressive training. By doing so, additional training samples are requested from the users only when necessary, starting from a low number of training samples and gradually requesting more information as modeling demands.

For our system evaluation we present *functionality examples* and *statistical simulations*. The first show the advanced modeling capabilities our similarity preference learning algorithm exhibits. The latter examines whether these modeling capabilities can be established in a consistent manner and without undesirable erroneous results.

7.1 Training

In this section we present issues regarding the training of our algorithm. First we examine the training input provided by the user and then we provide a detailed analysis of our progressive training methodology.

7.1.1 Similarity input/output discussion

In order to train our system we present users with a set of a Query and a Database values and request a similarity assessment. This assessment is performed in the form of a ranked classification by choosing a class describing their perception of similarity. User preference is expressed through a crisp set of n classes $\{class_1, class_2, \dots, class_n\}$ that represent similarity within the $[0,1]$ interval. These classes can be linguistic terms, also known as linguistic hedges (Zadeh, 1972) or expressed by a numerical descriptor. In order to facilitate users with various similarity assessment capabilities (novice to expert) numerous classification schemes can be used. Our algorithm's applicability is not dependent on the chosen classification scheme but its accuracy is. A linguistic classification scheme for a novice user could be:

Similarity Classes = {No Similarity, Very Low Similarity, Low Similarity, Average Similarity, High Similarity, Very High Similarity, Identical Similarity}.

A more expert user could use a higher number of classes (e.g. 13), following for example a university grading scheme such as this one:

Similarity Classes = {F, E, D⁻, D, D⁺, C⁻, C, C⁺, B⁻, B, B⁺, A⁻, A}.

For the purposes of our application we assume a linear ordering of these classes, since we are modeling similarity preference of each specific user (we do not combine preferences from different users). Partial ordering is not examined because all similarity values

correspond to a single attribute evaluation (we do not combine attributes), therefore all similarity values are directly comparable to each other. Of course there exists incomparable information in the real world within which there are linguistic terms that do not correspond to a linear ordering on the universe (Xu et al., 1999) but such research is outside the scope of this work because we constrain the user to choose from one of our predefined classes.

Our algorithm translates the ranking provided through the set of n classes into a quantitative similarity value. As we mentioned, we assume linear ordering between the classes chosen. If the nature of the class selection is such that linear ordering does not hold true, then we expect a conversion method to be provided externally to our algorithm. For linear ordering a similarity value is assigned to each class using the following equation:

$$V_i = 100*(i-1)/(n-1) \quad (7.1)$$

In the above equation n represents the total number of classes, i the current class under examination and V_i the quantitative similarity representation for class i . The algorithm performs the necessary training based on these quantitative values.

7.1.1.1 Class identification after simulation

After a successful solution is found the model is used for simulation, during which the system calculates the corresponding similarity value for a new case. We compare this calculated similarity value with the pre-defined quantitative similarity representations of each class (from equation 7.1). In most cases an exact match will not exist between these values to assign automatically a single class as a return. Our system then can follow one of these two approaches:

- 1- Use a minimum distance criterion to match the calculated similarity value to a single class.
- 2- Return not a single class, but both classes that the value is closest to.

An example can be seen in figure 7.1. Let us assume that the number of similarity classes is 5, namely *Classes = {No Similarity, Low Similarity, Average Similarity, High Similarity, Identical Similarity}* and their corresponding quantitative transformation using equation 7.1 is *Classes = {0%, 25%, 50%, 75%, 100%}*. Now let us examine the case where the system calculates a similarity value of 42% (or 0.42) for a set of inputs, which is a value that does not match any of the 5 quantitative values representing the classes.

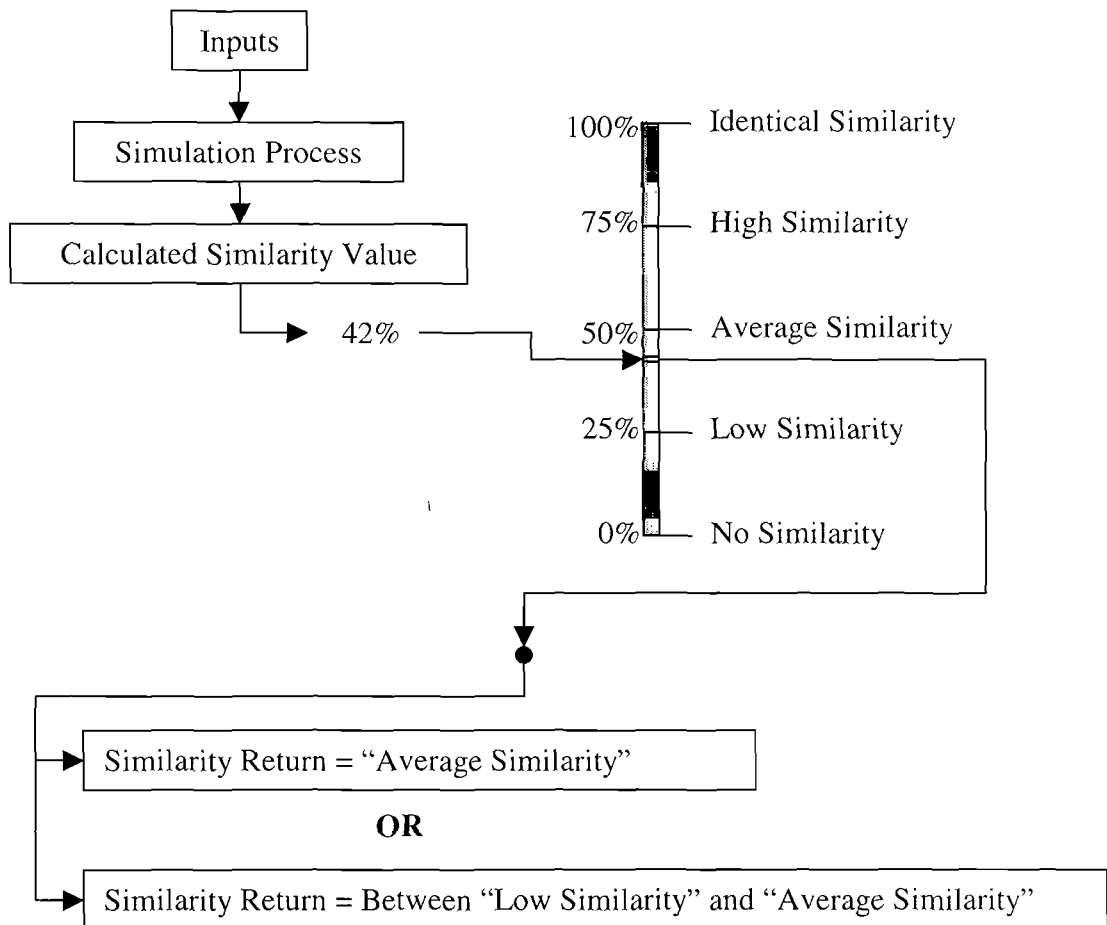


Figure 7.1: System return values

The system could return the class that is closest to the quantitative value, which in this case would be “*Average Similarity*” class. Alternatively, the system could return not a single class, but both classes that the quantitative value is inbetween, in this example classes “*Low Similarity*” and “*Average Similarity*”.

Here we should also point out that, strictly speaking, a direct class identification would be incorrect. The minimum distance criterion used to identify the winning class is based on the assumption that a metric distance has been established between classes during training. However, this is not the case; the user has provided only a ranking (ordering) of classes, not an exact metric relationship between them. For the above example, we know that class “*Low Similarity*” expresses a smaller similarity degree than the “*Average Similarity*” class, but we do not know that “*Low Similarity*” is twice worse than “*Average Similarity*” as the quantitative values might imply. So the distance criterion should be used with the understanding of the underlying assumptions involved.

The transformation from qualitative to quantitative values (and the inverse) is shown in figure 7.2. We should distinguish this fuzzification – defuzzification process from the fuzzy functions used within the system. The fuzzification – defuzzification

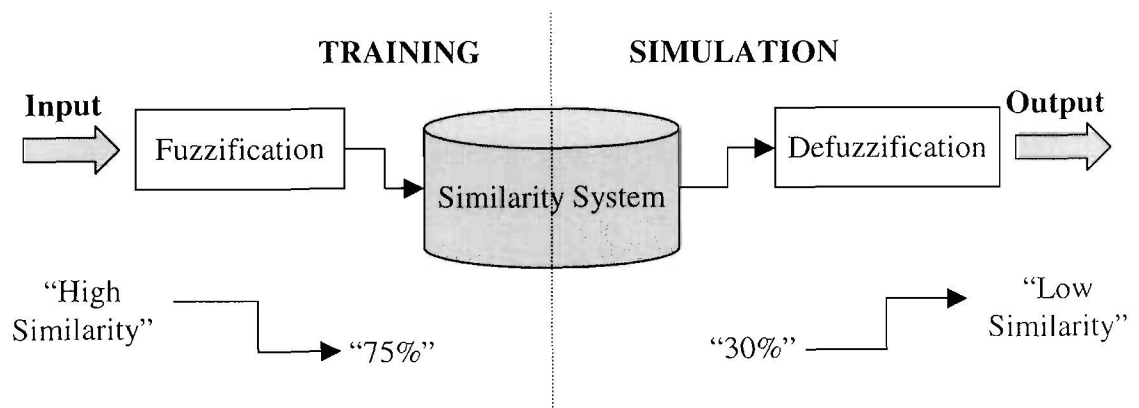


Figure 7.2: Classification translation to quantitative values during training and vice-versa during simulation

process acts as an additional pre-processing step for training and post-processing step for simulation. The fuzzy functions are part of the system to model the expressed preference.

7.1.1.2 Influence of total class number

The approach described earlier requests a classification from the user at the training stage and returns a classification at the simulation stage. The number of classes chosen does not affect the applicability of the algorithm. Based on user expertise and modeling accuracy demands, several classification schemes can be used. However, the higher the number of classes the more detailed the input is and more advanced modeling capabilities can be reached. This can be easily understood by examining figure 7.3. This graph represents a 2D section of inputs with different number of classes (0% and 100% are omitted). With blue squares we can see the training set that 3 classes would provide and with red circles the corresponding set from 10 classes. It is evident that the higher the number of classes, the more accurately similarity preference can be modeled. We can see that a very limited number of classes would lead to generalization at parts of the output space (similarity) that no information exists. Therefore a number of at least 7 classes is suggested.

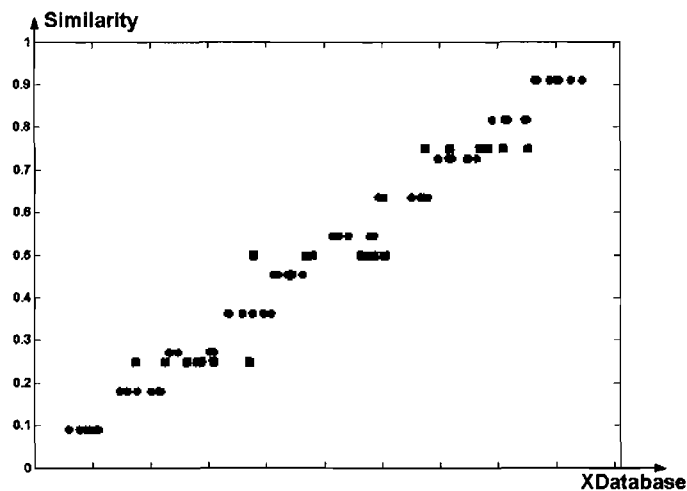


Figure 7.3: Effect of number of classes on training set

7.1.2 Progressive training

One of the major characteristics of our system is that modeling complexity increases as preference gets more complex. This propagates to our training of the algorithm as well. We do not want to overwhelm the user by requesting a large amount of training samples at the beginning of the process. Instead, we gradually increase the request for sample data as we carry on the modeling and we still identify high errors.

Figure 7.4 is a diagram representing the modeling process. In the first step we attempt to get an idea of the underlying complexity that might exist between Inputs and

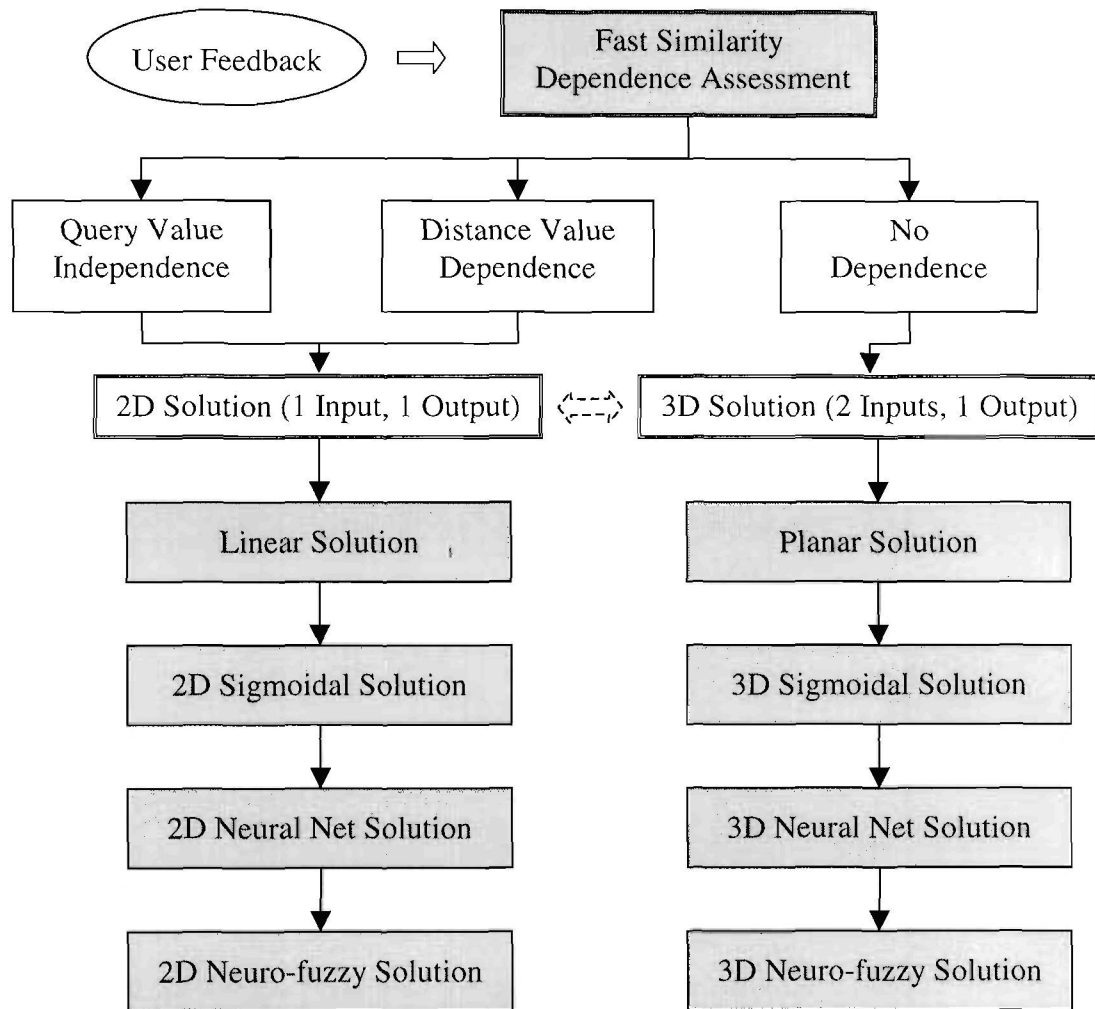


Figure 7.4: Progressive training

Output. Depending on the obtained dependency result we proceed to a 2D (1 Input, 1 Output) or 3D (2 Inputs, 1 Output) solution. In the 2D case interpolation of lines takes place at the next step (or planes for the 3D case). Then, if necessary a non-linear solution is performed using simple sigmoidal functions. If accuracy is still not achieved, more complex sigmoidal functions are implemented. As a last resort, our customized neural network is triggered together with a neuro-fuzzy solution. In the remainder of this section we examine how training takes place in each of these steps, focusing on the more complex 3D solution, since the 2D one is a simpler derivative of it. Throughout this section we examine only one half of the solution and before we start examining the other half we interpolate our developed model from the first half to see if symmetry exists and avoid further training.

User Feedback. Users are providing the input for the training process. Pairs of sample values within each dimension are presented and their similarity assessment is requested. For example, we ask them: “If you request a geospatial object from Time=11/12/2003, and we return an object from Time=02/04/2001, how similar is that?”. This way the training set is created, composed by a set of {Query, Database, Similarity} points (two inputs, 1 output). More information on the user similarity assessment is provided in section 7.1.1. We should also mention that our training is progressive, meaning that a small number of initial training points is required and based on the underlying complexity more samples are requested when necessary.

Similarity Dependence Assessment. The first processing step involves the identification of Input/Output dependencies that would simplify our solution. More specifically, we try to identify whether the given similarity preference is:

- Independent of the Query Value.
- Dependent solely on the distance between Query and Database Value and not the actual Query and Database Values.

In either of the above results we proceed with a two-dimensional solution (1 Input, 1 Output). If independence of the Query Value is recognized that would mean the input is only the Database Value. In the case that similarity is solely distance-dependent, then this distance is the only input to the algorithm. If no dependence is detected, the algorithm proceeds with the more computationally expensive (but necessary) three-dimensional solution with 2 Inputs (Query and Database Values) and 1 Output (Similarity Preference). An important aspect of the algorithm design is that it can switch from a two to three-dimensional solution and vice versa in real time, in other words identify or dismiss dependencies (therefore the dotted arrow in figure 7.4). The function equations are formulated such that based on the detected angle φ (see equation 7.3) the two dependency subcases are easily identified. On the other hand, if increased errors are detected, then the 3D solution is triggered.

In order to calculate the angle φ we interpolate a single plane in the active output space. We make this distinction between active and inactive output space in terms of similarity gradient. Areas with high similarity gradient are considered active and appropriate for measuring angle φ . Inactive areas are not included in the solution due to almost independence of the angle value since the interpolated planes are almost parallel to the XY plane. In figure 7.5 there is an example of an active area in green and the two inactive areas in red. Axes X and Y correspond to the inputs of our process, namely X_Q and X_{DB} . The Z axis represents the similarity preference output.

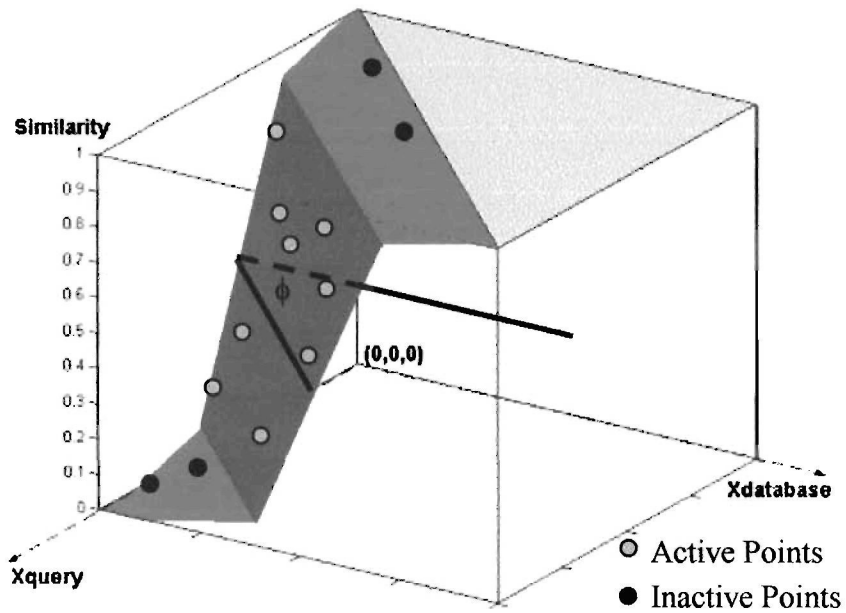


Figure 7.5: Calculating dependency angle

In practice, this is achieved during training by assigning part of the similarity classes to be inactive and the rest active. An example would be (fig. 7.6):

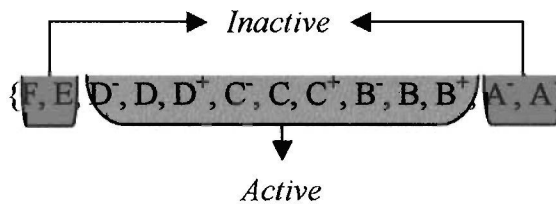


Figure 7.6: Active and inactive classes

This active/inactive categorization will vary based on the chosen classification scheme but the structure of having the inactive parts at the two ends of the categorization spectrum will remain the same.

The calculation of the angle is based on the solution obtained for the plane parameters (the green active plane in figure 7.5). If the database value is X_{DB} , the query value is X_Q , and the quantitative representation of the similarity value is Sim , then the plane equation would be:

$$a_1 X_Q + a_2 X_{DB} + a_3 = Sim \quad (7.2)$$

To calculate the three parameters a_1 , a_2 , and a_3 we need at least three training samples (assuming that all three samples do not belong to the same line). For higher stability in the results a higher number of points is suggested (>5) with the use of least squares for this overdetermined solution. The angle can then later be calculated using:

$$\varphi = \arctan\left(\frac{-a_1}{a_2}\right) \quad (7.3)$$

Based on the angle value we decide to proceed to a 2D or 3D solution as explained earlier.

Planar Solution. At the next step the simplest set of fuzzy membership functions (FMFs) is used, the one composed by a piecewise planar solution. The similarity function Sim_{Planar} expressing the relation between a database value X_{DB} compared to a query value X_Q is:

$$Sim_{Planar}(X_Q, X_{DB}) = a_1^R(i)X_Q + a_2^R(i)X_{DB} + a_3^R(i) \quad (7.4)$$

Parameters a_1 , a_2 and a_3 define the planes and index i specifies the current plane under examination. Our solution is composed by a collection of five planes as we explained in chapter 4. An example of the planes configuration is shown in figure 7.7.

During training and in order to identify the plane parameters an important constraint should be imposed, the one of continuity between planes. In order to avoid a computationally expensive least squares solution with this added constraint, we define the plane characteristics and operational output space in a specific manner:

- 1- All planes have the same angle φ as calculated before. In other words the ratio $a_1(i)/a_2(i)$ remains constant.

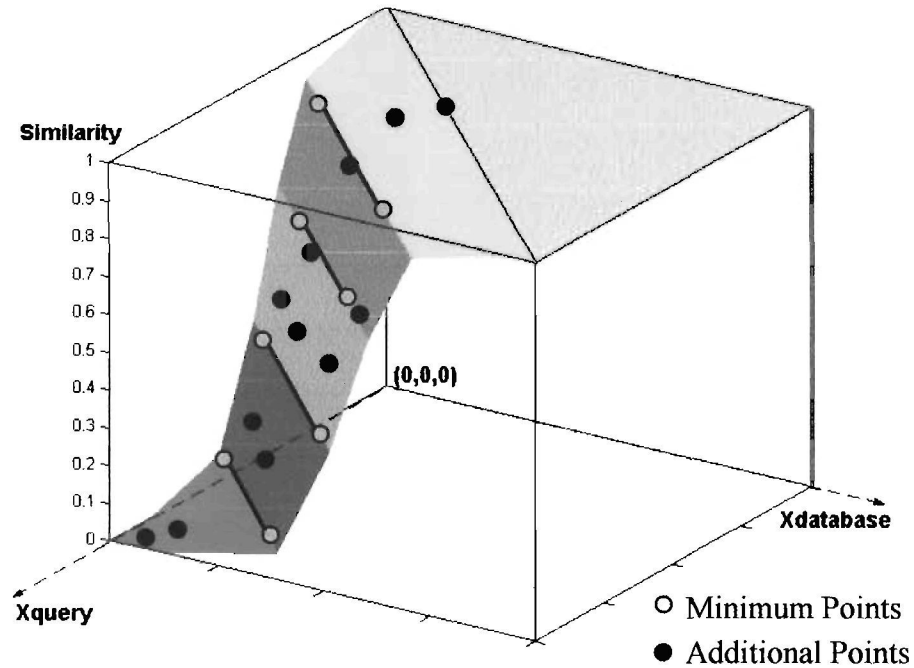


Figure 7.7: Training points for planar solution

- 2- The intersection of successive planes should correspond to an exact quantitative representation of a class.

Using the above the solution can be easily obtained by finding the equations of the four lines where the planes intersect (fig 7.7). Each line corresponds to a specific Z value, therefore it is a two dimensional solution. In addition to that, the angle remains constant, so the only parameter we are solving for is the shift d along the X_{DB} axis (fig. 7.8). Using the equations of the two lines that define the upper and lower bounds of a plane we can calculate the plane parameters as follows:

Assume that the two line equations are:

$$X_{DB} = kX_Q + d_A \quad \text{for } Z = Z_A \quad (7.5)$$

$$X_{DB} = kX_Q + d_B \quad \text{for } Z = Z_B \quad (7.6)$$

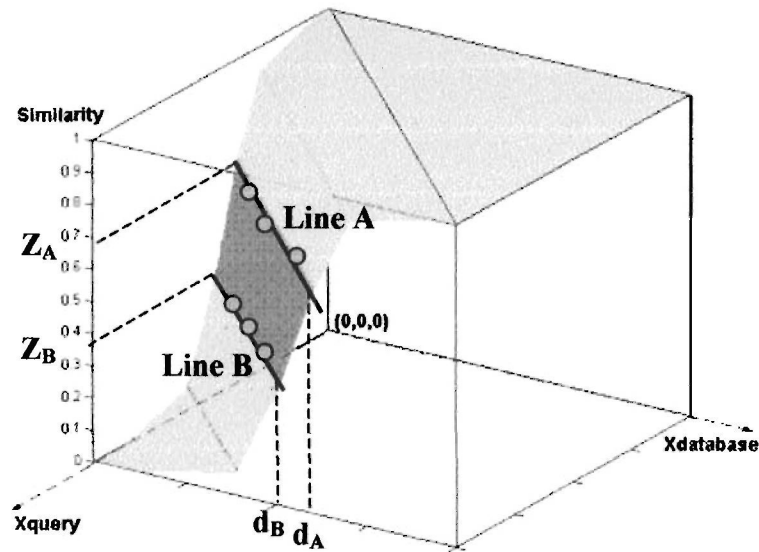


Figure 7.8: Calculation of plane parameters

Parameter k (slope of the line) is known from the angle φ ($k = \tan(\varphi)$). So for every line we solve with all the points that correspond to that specific class to calculate parameter d (shift of the line). When the parameters for the two lines are found (k, d_A, d_B) then the plane parameters are calculated using the following equations:

$$a_2 = \frac{Z_A - Z_B}{d_A - d_B} \quad (7.7)$$

$$a_1 = ka_2 \quad (7.8)$$

$$a_3 = Z_A - d_A a_2 \quad (7.9)$$

So to calculate the parameters for the five planes we need to calculate the parameters for the four intersecting lines. Since all lines have common slope k we only need a single training sample for each line, so four training samples in total. It is understandable though that a number of points larger than this minimum set (4) would provide a more robust solution.

An observation is that we are not using all the training points provided by the user, only the ones that are classified to one of the four classes that define the intersecting planes. For example in the classification used before we may use the following classes for our lines calculation:

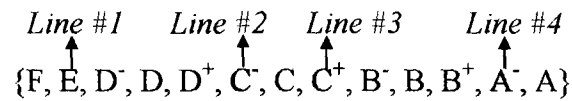


Figure 7.9: Assignment of classes to intersecting lines

The training points in the remaining classes will be used for accuracy evaluation of the interpolated planes. Here we should mention that the number of planes is not fixed but at least five. We need two planes to describe the inactive similarity ranges, one for the lower and another for the higher similarity values (close to 0 and 1). We also need another plane with a relatively short range describing the center of the similarity range (close to 0.5). Inbetween the inactive planes and the center plane we need at least one plane at each side of the center plane. So the number of interpolated planes is $2n+3$, the $2n$ is to ensure symmetry. For $n=1$ we get the minimum plane number which is 5.

Sigmoidal Solution. Following the plane interpolation, an accuracy assessment through a fitting error is performed. 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. The similarity function $\text{Sim}_{\text{Sigmoidal}}$ for a database value X_{DB} compared to a query value X_Q is:

$$Sim_{Sigmoidal}(X_Q, X_{DB}) = \left\{ \begin{array}{l} \frac{1}{1 + e^{-a(X)}} \\ X = (X_Q - X_{DB} - c) \cos \varphi + (X_Q + X_{DB} + c) \sin \varphi \end{array} \right\} \quad (7.10)$$

The parameters we solve for are the angle φ , the shift c , and the slope a . The initial approximations for the parameters are as follows:

- Angle φ is the same angle calculated from the dependency analysis.
- Shift c is calculated from the parameters of the center plane using $c = \frac{-(0.5 - a_3)}{a_2}$.
- Slope a cannot be calculated directly from the planes. Instead an indirect value is assigned through a fast least squares solution with four points each belonging to one of the lines used to calculate the planes.

The importance of the good initial approximations that our system offers is further examined in the statistical section of this chapter. After the initial parameters are set, a least squares solution follows to calculate the final values. If the underlying complexity is high, the simple sigmoidal similarity functions might not be able to adequately model it. For these cases we showed how a more adaptable set of functions with higher modeling capabilities can be incorporated. The solution for these functions will depend on the chosen functions, but the central idea remains the same: use the simple sigmoidal function solution as the initial approximation for the more complex ones to follow and perform an evaluation of the fitting error after each interpolation until you achieve maximum desired accuracy.

Neural network solution. A significant modeling constraint of the fuzzy functions is their monotonically decreasing behavior. To compensate for this, a novel neural network

solution was developed as introduced in chapter 5. Its purpose is to capture possible errors of the fuzzy functions and model them accurately. From the modeling perspective these errors are created by unexpected preference expression, therefore the applicability of a neural network is appropriate.

Before the input vectors are created for our multi-scale radial basis function (MSRBF) network, a self-organization of the fuzzy function to the dominant signal takes place. This process was explained in chapter 6. Then the MSRBF is trained based on the output errors of the fuzzy functions. A detailed description of the training is provided in section 5.7.

Neuro-fuzzy solution. At the final stage, and whenever the MSRBF is used, a merging of the fuzzy functions and the MSRBF is performed creating a neuro-fuzzy system. The weights are re-adjusted to achieve even higher modeling accuracy. The mathematical solution behind the least squares training is described in section 6.3.

7.2 Functionality examples

In this section we demonstrate the advanced modeling capabilities of our method. Our application is inspired by common but complex user preferences within geospatial environments. We begin with a walk-through example with similarity learning in the resolution dimension using the fuzzy membership functions. We continue with a cadastre example using GIS and conclude with temporal preference example to show applicability of our neural network with the fuzzy functions. Additional examples of our system's applicability can be found in section 4.6.

7.2.1 Walk-through training example for the resolution attribute

Let us consider users who query a geospatial data collection and specifically request imagery of a particular resolution (ground pixel size) that will be used for object extraction. Their interest decreases gradually (but not necessarily in a linear fashion) as resolution increases to the degree that objects would not be identifiable. This expectation holds true for $X_Q < X_{DB}$ (i.e. when the returned pixel size is larger than the requested one). Furthermore, the user may have a cost function in mind associated with a better (smaller pixel size) than requested resolution (e.g. due to price, storage, and processing time). This translates to a similarity relation that can also be non-linear as pixel size improves ($X_Q > X_{DB}$). 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 much finer resolution this tolerance range may be smaller due to for example higher price or manipulation cost.

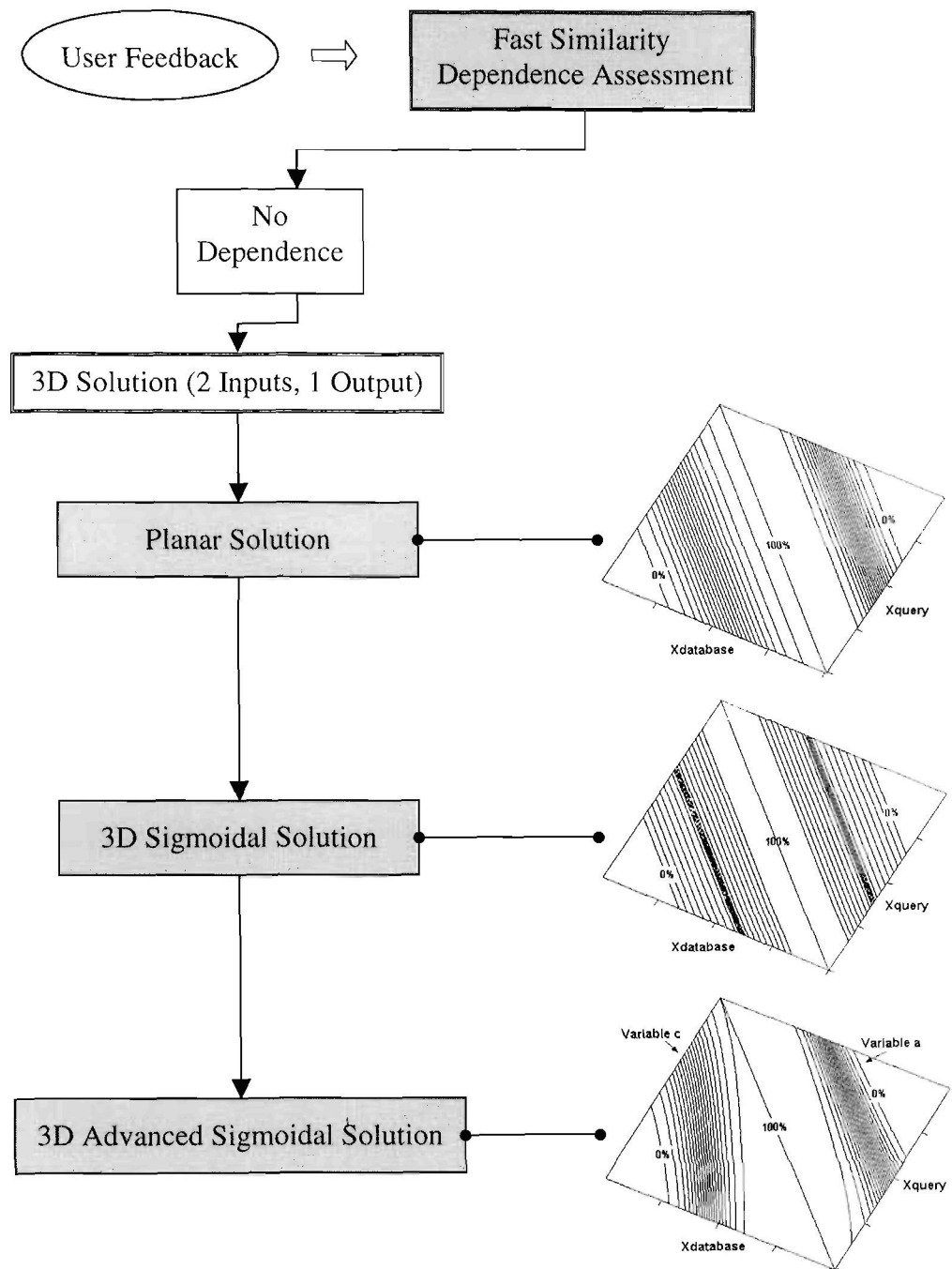


Figure 7.10: Resolution attribute profile training

Figure 7.10 shows the process followed to capture the above similarity preference with our fuzzy membership functions:

- Creation of training dataset by providing the user with different pairs of pixel size (query and return) and request a similarity assessment.

- Interpolate planes on each half ($X_Q > X_{DB}$ and $X_Q \leq X_{DB}$) and calculate the corresponding plane parameters.
- Examine if the interpolation error is better than the desired accuracy.
- If not, calculate angle φ and its associated error.
- If angle φ is close (closeness range predefined by user) to 45 or 90 degrees then eliminate the third dimension.
- Interpolate two sigmoidal functions each applied on the corresponding half. Calculate the initial approximations of the sigmoidals using the plane parameters.
- Examine if the interpolation error is again better than the desired accuracy.
- If not, allow the so far constant sigmoidal parameters to be expressed by more complicated functions as preselected by the user.
- Solve for the parameters including the additional variables.
- Return the best possible solution from the above functions.

In figure 7.10 the contour plots of the resulting similarity surface from each fuzzy function are presented. After training for our resolution example, the resulting function in the right half is a sigmoidal one with a variable slope a (fig. 7.10 bottom contour graph). The variable slope is able to express user alterable tolerance depending on requested 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.

In the left half, the rate of similarity decrease does not change so slope a remains the same (i.e. the isoline distance). What changes is the spread value c . By doing so, this complex fuzzy function has the ability to express user similarity tolerance as the query

value increases. In figure 7.11 the final similarity surface is presented.

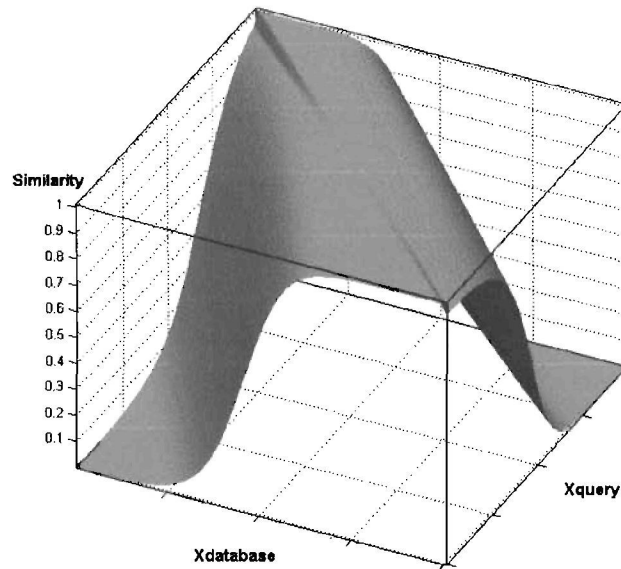


Figure 7.11: Advanced sigmoidal fuzzy similarity function

After successful training the mathematical parameters of the functions are stored to be used later during simulation, creating user preference profiles. A profile example describing user similarity preference as identified by the above functions is shown in figure 7.12. *Profile ID* is a unique key number to identify the profile. For each of the two asymmetrical solutions a separate function is provided. The *Class ID* refers to the type of function used and the relationship between the parameters and the function's output (i.e. the similarity equation). The calculated parameters after training are stored thereafter.

Profile ID: 52368	
Left Side – Class ID:12	Right Side – Class ID:17
$a = -0.07$	$c = 85.17$
$c_o = -60.04$	$a_o = -0.21$
$c_1 = -0.08$	$a_1 = -0.01$
$\phi = 1.57$	$\phi = 1.57$

Figure 7.12: Profile example

7.2.2 Additional example using the fuzzy functions

Here we demonstrate a representative example of their functionality on a cadastre/real estate application. Let's investigate user preference of a geospatial attribute expressing parcel value per square meter. The function is composed of two sub-functions, each one applicable in half of the input space (e.g. $X_Q > X_{DB}$) to compensate for asymmetrical cases. A major factor for choosing a sigmoidal function comes from its superior modeling capabilities. The parameters c_R and c_L specify the translation along the $X_{database}$ axis, which is especially useful in specifying the highly active portion of the function (close to 100%). Efficient manipulation of the slope of each sigmoidal function can result in representing a variety of cases, ranging from a linear up to a step-like behavior. A result of this trained function can be seen in figure 7.13.

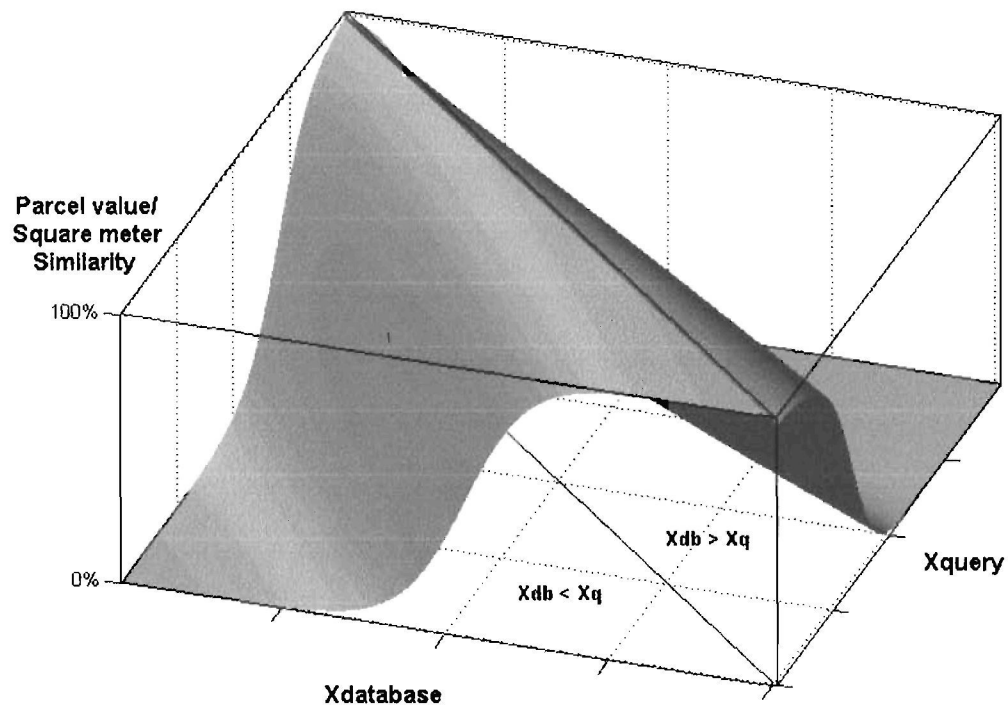


Figure 7.13: Example of a user preference function

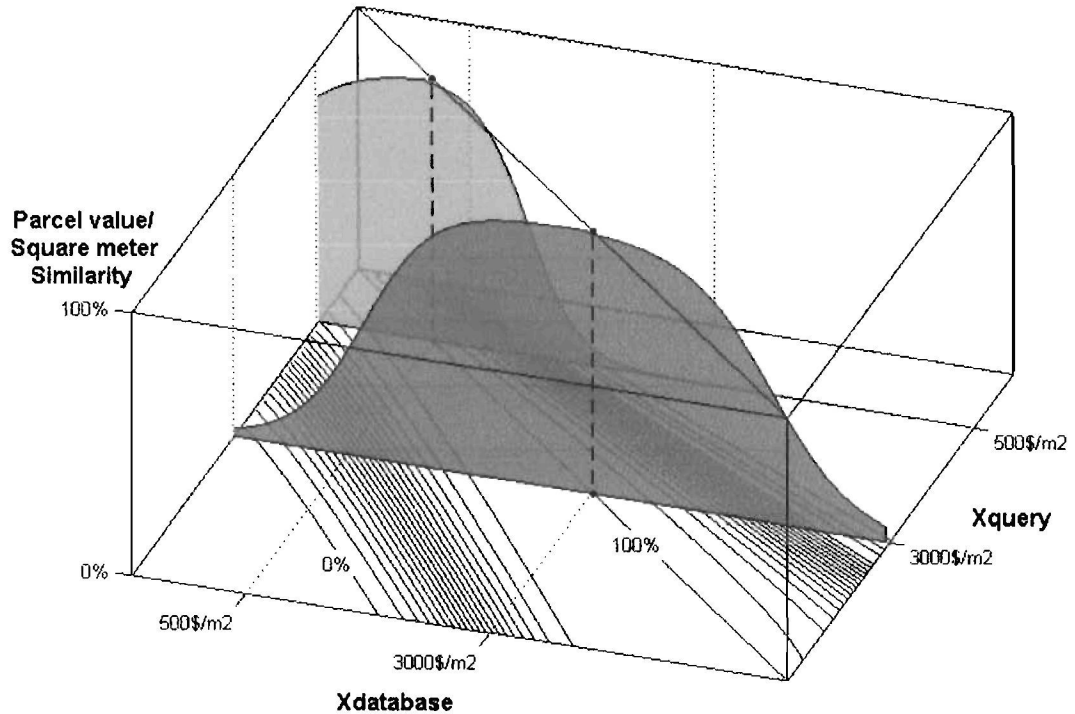


Figure 7.14: Contour plot and query examples of this preference function

In figure 7.14 we have the corresponding contour plot of the above figure. We also included two sections for specific user requests, for parcel value per square meter (PVSM) of $500\$/m^2$ and $3000\$/m^2$. By examining these two sections we can conclude the following:

- i) In the $X_Q > X_{DB}$ half (left side at the graph) the dependency of shift on the X_Q input is able to express the gradual decrease of user's interest as the returned PVSM is smaller than the requested. Note in figure 7.14 how user flexibility increases as the PVSM request X_{User} gets larger. No normalization could encapsulate this dependency. Analyzing the reasons of such a preference pattern could lead to the conclusion that the higher the requested PVSM, the more flexible the user is as to the range of highly similar results when $X_Q > X_{DB}$.

ii) At the right half more complicated modeling is necessary. A dependency on the X_Q input exists for both slope and shift. This powerful combination expresses user decrease of interest when the returned PVSM is larger than the requested one. It is dependent on the X_Q input since the larger the requested value is, the larger the range of highly acceptable values increases (through shift manipulation) and the interest decrease rate is smaller (done through slope modification). In other words, when users request 500\$/m² PVSM they are less flexible in accepting larger values than when querying for a 3000\$/m² one (for a return value higher than the requested) and this is what mainly we express with the displayed function.

7.2.3 Neuro-fuzzy example

In this example we demonstrate the combined application of the fuzzy functions and the MSRBF. Let's assume that a user is querying for satellite imagery from a specific temporal instance. At the initial stage, the fuzzy function captures the majority of the similarity preference using a sigmoidal function (we only examine the left side). Because the fitting error is still high the MSRBF is used. The fuzzy function is self-organized to the overwhelming signal and the errors of this new function are the inputs for the MSRBF. After the training of the neural network a global solution follows to fine-tune the weights.

The resulting user preference is shown in figure 7.15. We can see the effect of the MSRBF acting as localized error-corrector with the small amplitude variations. But also there is a large-scale correction when the returned value (X_{DB}) is close to January/2001. For some reason the user does not prefer results from that period of time and the MSRBF is able to express that. This preference can relate to additional knowledge that the user

might have, for example that the optics of a specific satellite were not calibrated correctly at the beginning of January, therefore the quality of the images might not be up to par.

The above example is indicative of our system's modeling capabilities when complex user similarity preference is present. Such preference is not currently supported in geospatial query systems, but the necessity of its incorporation is obvious..

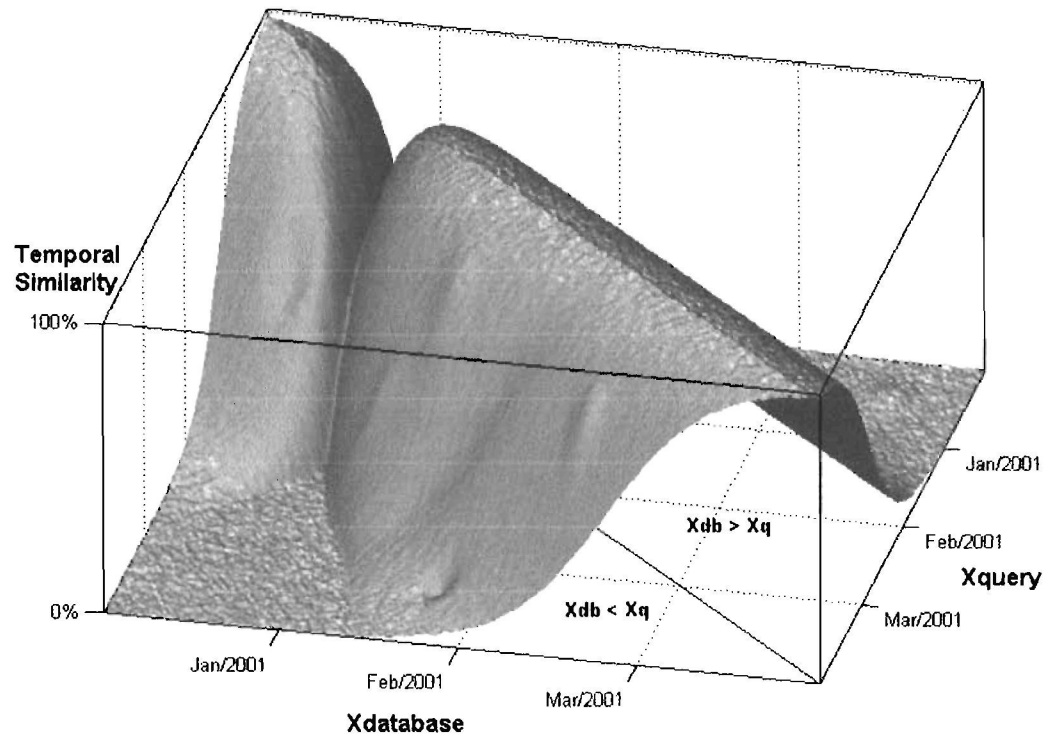


Figure 7.15: Similarity preference captures with our neuro-fuzzy system

7.3 Statistical testing

Our statistical tests aim at establishing the robustness of our neuro-fuzzy system. First we examine the fuzzy functions and then the MSRBF neural network performance.

7.3.1 Fuzzy functions assessment

Our complex functions have advanced modeling capabilities as we already showed. An important question that often rises when dealing with complex non-linear function approximations is the stability of the algorithm. In other words what is the influence of factors such as initial approximations, noise, and number of training samples to the least squares solution. A thorough investigation is offered to assess algorithm's performance.

7.3.1.1 Influence of initial approximations

A repetitive problem in non-linear least squares solutions is caused by the fact that if the initial approximations are far away from the target values there is the possibility that convergence to the desired solution will not always be achieved. To address this, within our system we have developed a method of calculating accurate initial approximations from previously interpolated less complex functions (Mountrakis and Agouris, 2003). In this section we investigate the relationship between convergence and the distance between initial and target values for each of the three parameters: shift, angle, and slope. All tests were performed with 10 training points, with the initial values of the other two parameters having the same value as their corresponding target values so they would not influence the solution. However, each solution was performed by solving for all three parameters simultaneously each time to ensure overall stability. Convergence is achieved when solution parameters are within these (strict) thresholds $\pm 1\%$ of target value for shift, ± 0.3 degrees for angle, and ± 0.005 for slope.

Shift parameter. The influence of shift initial approximations can be seen in figure 7.16. The X axis represents the value of the initial approximation and the Y axis shows the target value. We can see that for a specific range ($\pm 0.33 \cdot C$ for that graph) of difference between the initial and target value convergence is achieved consistently. On the other hand, solution is sparingly achieved beyond that range. This shows the importance of having good approximations in the shift parameter.

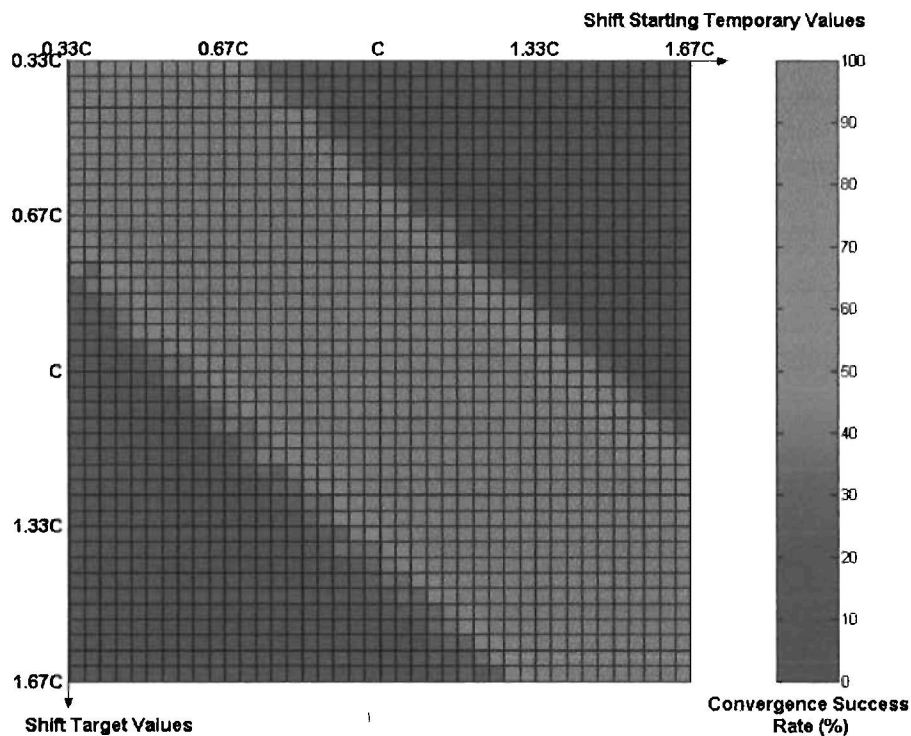


Figure 7.16: Influence of shift's initial approximations to convergence

Angle parameter. Another parameter we evaluated was the angle, with the results shown in figure 7.17. The X axis represents the value of the initial approximation in the angle parameter and the Y axis shows the target value. Angles are measures in degrees. The behavior is the same as the shift parameter. The range for convergence is approximately ± 8 degrees. Values further away from that range do not always ensure a successful solution, at least not within our strict thresholds as previously defined.

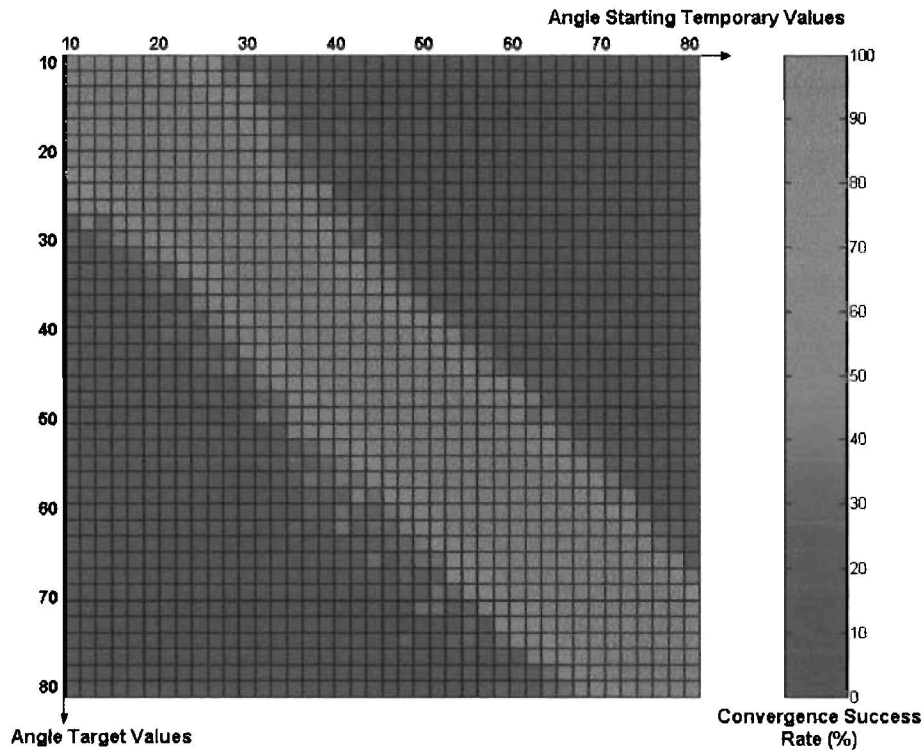


Figure 7.17: Influence of angle's initial approximations to convergence

Slope a. The most intriguing parameter in our assessment was the slope of the sigmoidal function. The obtained results are presented in figure 7.18. The X axis represents the value of the initial approximation in the slope parameter and the Y axis corresponds to the target value. The results do not exhibit the same behavior as the previous two parameters. For small slope target values (<0.15) there is a gradually increasing range of convergence. Beyond the 0.15 target value mark convergence is achieved consistently almost independently of the initial approximation. This is an exceptional result for our algorithm's design because the slope parameter is the only parameter that we cannot calculate a good initial approximation from previous fuzzy functions. So this higher tolerance on the initial values is desired and actually motivated part of the algorithm's design. Given the result of figure 7.18 we assign slope starting values close to zero to achieve high convergence rate independently of the target value.

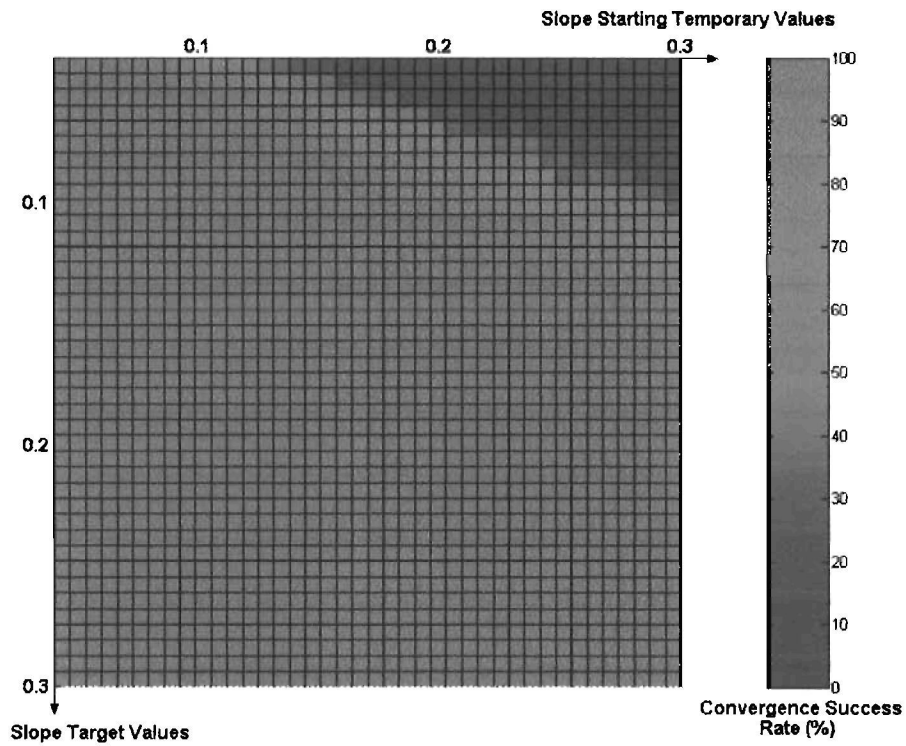


Figure 7.18: Influence of slope's initial approximations to convergence

The explanation for slope's unusual behavior is found in figure 7.19, where sigmoidals with different slope values are presented. We can see that a 0.01 change in the slope value will have a much more drastic influence as the slope gets closer to 0. Therefore changes beyond the 0.15 mark are almost insignificant, which explains the unusually high convergence rate and its independence from the initial value.

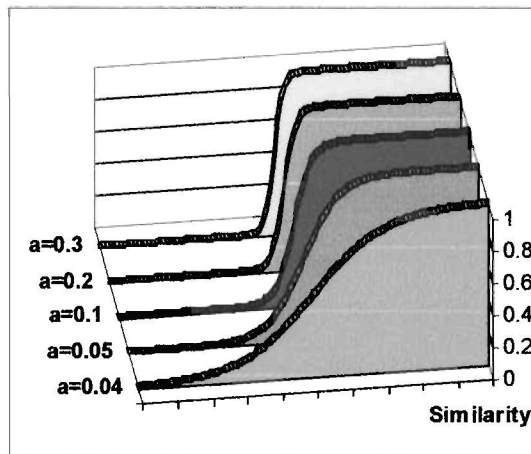


Figure 7.19: Influence of slope to sigmoidal's shape

7.3.1.2 Influence of noise and training sample size for convergence

The experiments of this section examine the influence of noise combined with the number of training points to a successful solution. Noise was inserted in the training dataset by altering the similarity value of a single point. This alteration varied from 0 to ± 0.5 and was imposed on a randomly selected point. The absolute value of similarity change is represented in the X axis of figures 7.20, 7.21, and 7.22. The Y axis shows the percentage difference from the desired target value in each parameter before noise was added. We should mention that all iterations started with a value 20% away from the target. Experiments were performed for a variety of training size $n = \{10, 20, 30, 40, 50\}$ to assess the stability of the algorithm. Also 1000 iterations took place for each result and their average is presented at the graphs.

The overall impression from the three figures is that the higher the number of training samples the more tolerant the solution is to noise. This was an expected outcome of our statistical simulations. Furthermore, slope parameter seems to be the one mostly affected with the introduction of noise. This does not necessarily translate to unsuccessful modeling, since as explained in figure 7.19, the influence of slope differences is in some parts significant and in others negligible. For this specific test, slope was set to 0.07. The other two parameters, shift and angle, appear to be more tolerant to noise with the shift showing a slightly better performance. There is a significant gain when the training size is increased from 10 to 20 points. We should mention of course that we want to keep this number as low as possible to avoid overwhelming the user with an excessive training set.

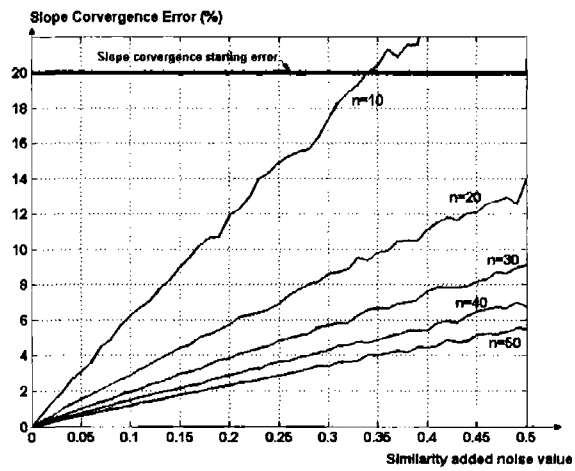


Figure 7.20: The influence of noise in slope calculation

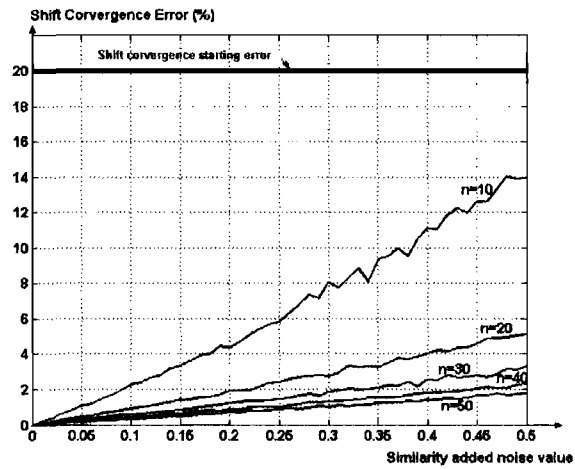


Figure 7.21: The influence of noise in shift calculation

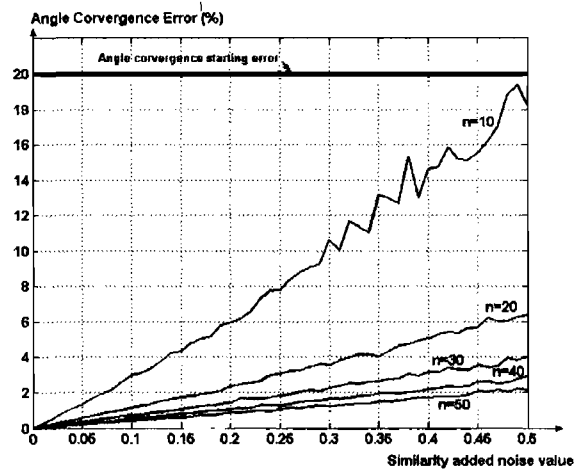


Figure 7.22: The influence of noise in angle calculation

7.3.1.3 Influence of noise and training sample size for convergence

An additional test we performed to test the stability of the algorithm is the one in figure 7.23. The X axis shows noise amplitude added and the Y axis represents the total number of training points used. We examine what effect noise and training size have on the convergence rate. From the graph we can see that in most cases low convergence failure exists. For the top right part where high failure is present, it is attributed to limiting thresholds combined with increased inserted noise. Especially for high noise values and low sample sizes the convergence failure rate reflects that values within thresholds cannot be achieved, not due to a least squares solution error but because of limited modeling capabilities of the sigmoidal function (e.g. it is a monotonically decreasing function). The variability within values is caused by the randomness of the selected point to which the noise was added.

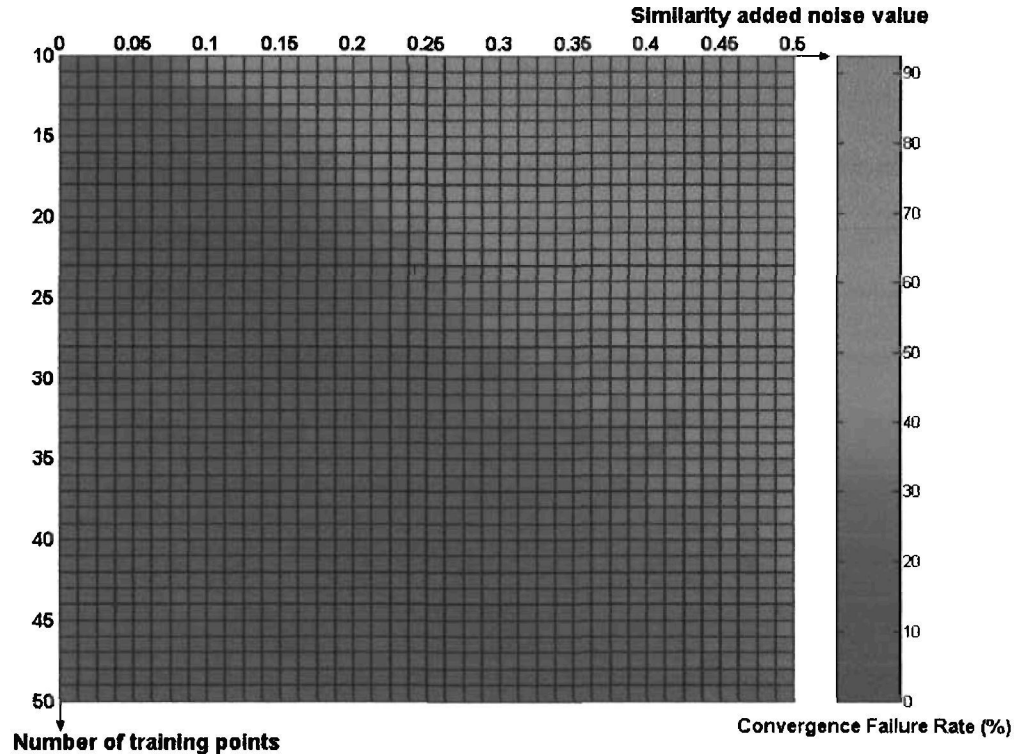


Figure 7.23: Influence of noise and training sample size for convergence rate

7.3.1.4 Influence of noise and training sample size for iteration number

Even though the overall approach aims at improving retrieval accuracy, the influence of noise and training sample size on the required iteration number was a concern. Therefore, we performed the experiment below to assess how the average iteration number for convergence changes as noise increases. The results are presented in figure 7.24. Noise is again inserted randomly to a single point of the training set as an error in the similarity value (X axis). We also included various training sizes (Y axis). We can see that the number of iterations increases when the number of training points decreases, as expected, but not to a prohibitive number, since the difference is only one additional iteration. The same conclusion applies to noise introduction, the single iteration difference, when going from noiseless to noisy training sets. The above remarks show that our algorithm's training speed is not significantly affected by noise.

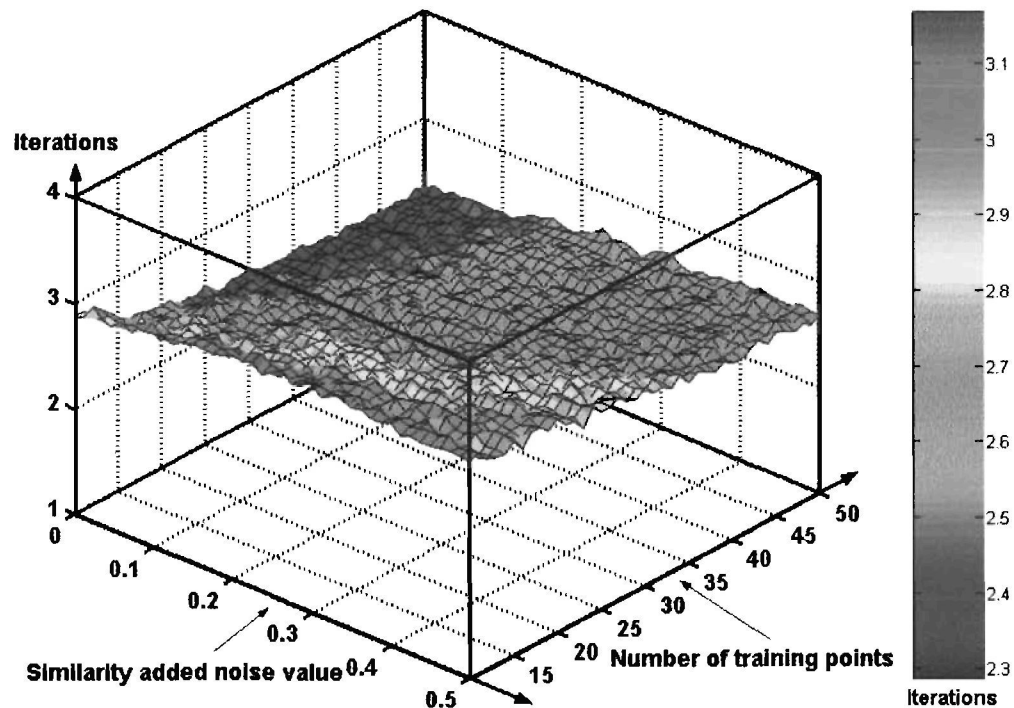


Figure 7.24: Influence of noise and training sample size for iteration number

7.3.2 Neural network assessment

In this section we present the statistical evaluation performed on our novel Multi-Scale Radial Basis Function (MSRBF) neural network. In order to exhibit the benefits of our algorithm, we compared it to the currently used solution. We begin the assessment with an explanation of the training sets used, and then we proceed with presentation and interpretation of the results.

7.3.2.1 Simulation training dataset

The testing dataset used for our evaluation process was created using a combination of gaussian distributions. An example of a training set can be seen in figure 7.25. We distinguish two kinds of gaussian distribution, the global and the local ones. The global gaussians do not overlap each other in the input space and are presented with orange color in figure 7.25. The local gaussians (green color of fig. 7.25) always overlap a global gaussian. Their purpose is to degrade the signal “clarity” of the global gaussian they overlap, therefore causing modeling errors. Some characteristics of the gaussians are:

- In our experiments the number of global gaussians varied from 2 to 40. The number of local gaussians was dependent on the global’s number ranging from 1 to twice the amount of global gaussians used, with a maximum of 2 local gaussian overlapping a single global gaussian.
- The centers for the global gaussians were chosen to cover completely the given input space. The centers of local gaussians were selected so the formulated local gaussians would overlap a randomly selected global gaussian but not within +/- one spread of the global’s center, otherwise globals would be unrecoverably modified.
- The spreads of the global gaussians were chosen to overlap completely the input

space and be within 10% of each other. The locals spreads were within 1/3-1/4 of the spread values for the global gaussians.

- Each global gaussian was represented by 14 points, and each local by 10 points. Higher number of points was used for the globals to ensure they are represented adequately within the whole input sample.

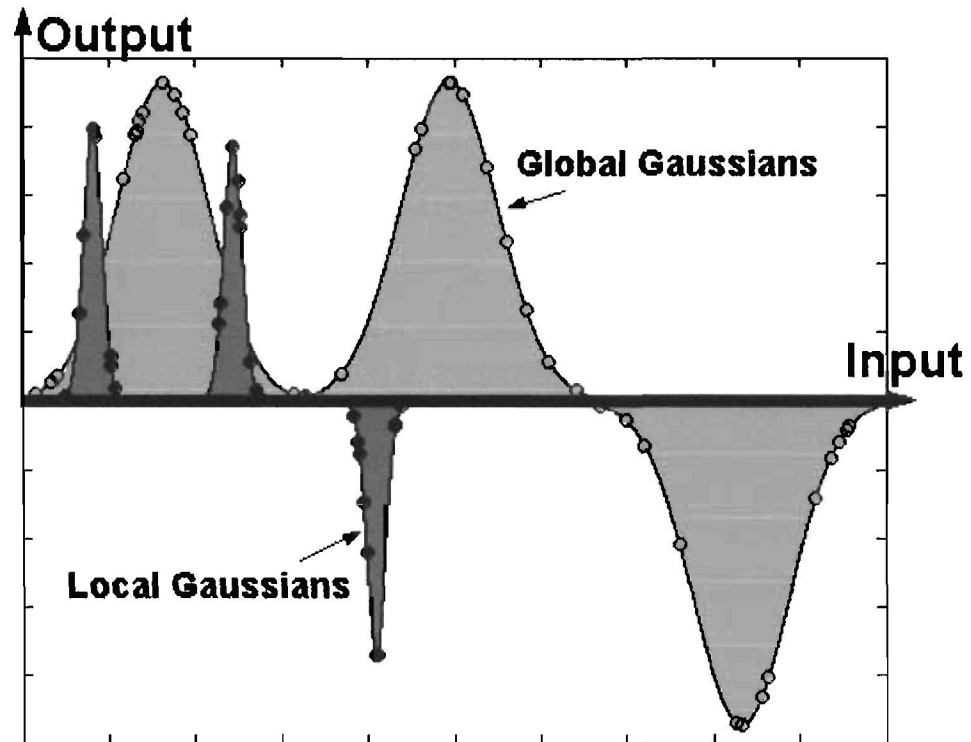


Figure 7.25: Creation of the simulations training dataset

For the error evaluation using the same gaussian distributions a testing sample was created with 6 times the number of points used for testing. A more clear representation of the training set of figure 7.25 is shown in figure 7.26. Black points shows samples resulting from local gaussians and light blue correspond to globals.

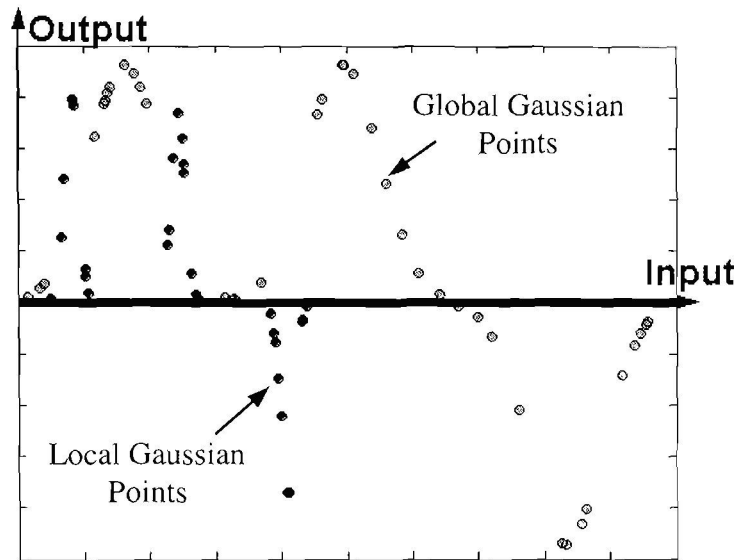


Figure 7.26: Training points of the simulations dataset

7.3.2.2 Configuration of compared neural networks

Three neural networks were evaluated based on the training set described above. We used a traditional single spread radial basis function network (RBF), a variable spread radial basis function network (VRBF), and our multi-scale radial basis function network (MSRBF). The properties of each network are as follows:

- **RBF.** The traditional RBF was tested. We used Matlab's code and spreads were assigned as $\{1:1:60\}$. Each spread resulted to an RBF network (60 in total). The one that had the best testing MSE (Mean Square Error) was reported. Iterations would stop if a maximum number of 100 nodes would be reached, except in the case of 30 and 40 global gaussians that a maximum of 200 nodes was allowed.
- **VRBF.** This network was different from the RBF. Instead of creating 60 network candidates based on a single spread that changed from one network to the other, we created a single network that tested variable spreads within each of its training iterations. The minimum spread used was 1 and the maximum spread value was

based on the spreads of the global gaussians used to create the training set. A step of 1 was assigned.

- **MSRBF.** Our novel neural network had the following input parameters: i) candidate spread values were the same as the VRBF to facilitate comparisons, and ii) for our density metrics we used three zones with $\{0.2, 0.8\}$ threshold values, requiring a minimum of at least one point per zone to accept a local MSE solution candidate. A minimum of one point only in zone one was required for global MSE solution candidate. The MSRBF training process is attempting to find the best balance between global and local candidates. In order to do so two variables were adjusted. The number of total nodes had a minimum of the summation of global and local gaussians used in the training set and such a step that a maximum of 5 total node values would be examined. For each of the total nodes value the network was setup to test different sigmas used to calculate the Maximum global MSE (see equation 5.8). The sigmas values varied from 0.3 to twice the number of total nodes, with a maximum number of 8 sigmas examined. So for every candidate dataset that was composed using a specific number of global and local gaussians, we would create a number of networks to test. To make this more clear, if we had 4 global gaussians and 3 locals, we would test for total nodes = $\{7, 10, 13, 16, 19\}$. For each of these nodes we would test for sigmas = $\{0.30, 2.30, 4.30, 6.30, 8.30, 10.30, 12.30\}$. Note that for $\sigma = 0$ our MSRBF turns into the VRBF since automatically all local solutions would be rejected.

All networks had their node centers chosen from the training sample values and the goal MSE was set to 0.001.

7.3.2.3 Interpretation of simulation results

A detailed representation of the results obtained from the comparison of the three networks is presented in the Appendix, at the end of the dissertation. Here we summarize the key points and discuss the results.

Overall Assessment. A summary of the three neural networks comparisons can be seen in table 7.1. In addition to the mean value, we calculated the median value, to compensate for the effect of possible outliers in the results. The mean and median values show

	MSE Training			MSE Testing			Nodes		
	RBF	VRBF	MSRBF	RBF	VRBF	MSRBF	RBF	VRBF	MSRBF
Mean	0.00323	0.02022	0.00066	0.01620	0.02210	0.00159	105.4	65.3	39.0
Median	0.00283	0.01934	$5 \cdot 10^{-6}$	0.01435	0.02369	0.00061	100	51	31

Table 7.1: Median and mean values for MSE Training, MSE Testing and Nodes

explicitly that our MSRBF outperformed RBF and VRBF consistently and by a large margin. Outperformance is mostly inferred through the MSE testing values. A more explicit comparison is displayed in tables 7.2 and 7.3.

	MSRBF vs. RBF	MSRBF vs. VRBF	RBF vs. VRBF
MSE Training	387%	2941%	525%
MSE Testing	915%	1285%	36%
Nodes	170%	67%	-38%

Table 7.2: Improvement percentage of the mean values

	MSRBF vs. RBF	MSRBF vs. VRBF	RBF vs. VRBF
MSE Training	56124%	384159%	583%
MSE Testing	2248%	3777%	65%
Nodes	223%	65%	-49%

Table 7.3: Improvement percentage of the median values

The MSRBF provided is at least 10 times more accurate results than the other two methods while at the same time it uses a significant less number of nodes, a significant advantage for faster simulation times. Here we should mention that we do not expect such a vast accuracy gap between the MSRBF and the rest of the methods when dealing with real data. Our dataset was biased to show cases that the existing modeling techniques fail and this experiment verified that. Also our network requires more training time, which might be a constraint for some applications. Nonetheless, our network's focus was accuracy and we have showed that explicitly through our simulations.

Generalization. Another important factor for the networks evaluation is how well they generalize from the training to the testing values. The testing size was 6 times larger than the training to make sure that the generalization evaluation was very detailed.

In order to quantify the generalization behavior of each network we compared the average difference between MSE training and MSE testing together with their standard deviation. The results are revealing: the MSRBF had 10 times better generalization than the VRBF and 100 times than the traditional RBF. This is mostly attributed to the complexity of the training set, that the other two methods were not able to capture.

	RBF MSE Testing - Training	VRBF MSE Testing - Training	MSRBF MSE Testing - Training
Mean	0.01296	0.00187	0.00093
Standard Deviation	0.01102	0.00986	0.00213

Table 7.4: MSE differences between training and testing

We also investigated the consistency of convergence of the networks and how it propagated from training to testing. Figure 7.27 shows the ratio between successful

solutions and unsuccessful, where success is measured in terms of having the MSE being

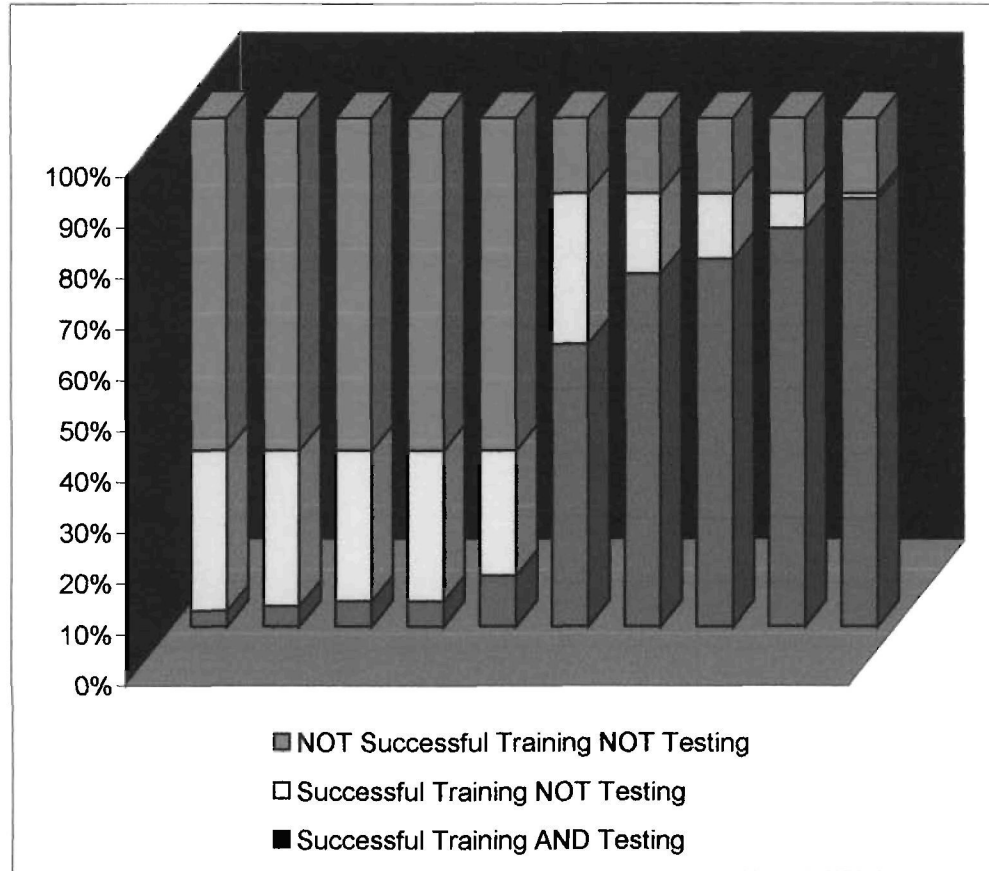


Figure 7.27: Convergence success rates for training and testing

less than a specific value. Each bar at the graph has three percentages that sum to 100%. The part in red shows the percentage of unsuccessful solutions in training, in essence all the times that the network did not converge to the desired MSE during training. The green and yellow parts correspond to the times it actually converged during training. The green parts show that not only training was below our MSE goal (0.001) but the testing was below this (or a multiplier of this) threshold as well. Yellow parts correspond to successful training but unsuccessful testing. The leftmost five bars correspond to the RBF solution and the five rightmost ones to the MSRBF. The VRBF converged in training only 3% of the time so we did not include it in this evaluation. From the five bars

for each network all of them were created using an MSE goal for training equal to the original MSE goal of 0.001. Going from left to right, the five bars show different convergence rates based on different testing MSE goal values. These five values were based on {1,2,3,5,10} times the original MSE target value (i.e. {0.001, 0.002, 0.003, 0.005, 0.010}). The different MSE testing values were used to show when even though strict convergence ($1 * \text{MSE original goal}$) was not achieved, how far away from it was the obtained solution.

The results show that a successful solution in training and testing for the RBF was achieved 3-5% of the time with the testing MSE goal ranging from 0.001 to 0.01. For the MSRBF convergence was achieved 56% of the time for MSE testing = 0.001, climbing fast to 70% for MSE testing = 0.002, end finishing to 85% for MSE testing = 0.01. This shows that even when strict convergence was not achieved the MSRBF was close to the desired MSE, while the RBF was never in that range.

Noise stability. A final analysis on the results was aiming at discovering the effect of noise in the achieved testing MSE. Noise is expressed in our dataset in the form of the local gaussians, overlapping the global ones. Figure 7.28 shows the effect of the number of local gaussians to the MSE testing for a global number of gaussians ranging from 20 to 30 and 40. All three networks were included in this comparison. The average of each network's MSE over the three global gaussian values is presented in figure 7.29.

The results verify the benefits of our approach. The MSRBF exhibited a small additional error as noise increased. In contrast to that, the RBF and VRBF showed to be significantly influenced by the introduction of noise. The MSRBF outperformed the others, even when minimal noise (i.e. low number of local gaussians) was present. Also,

this additional noise was not addressed in our MSRBF with an exponential increase in the number of nodes. There seems to be a more linear relationship between them instead.

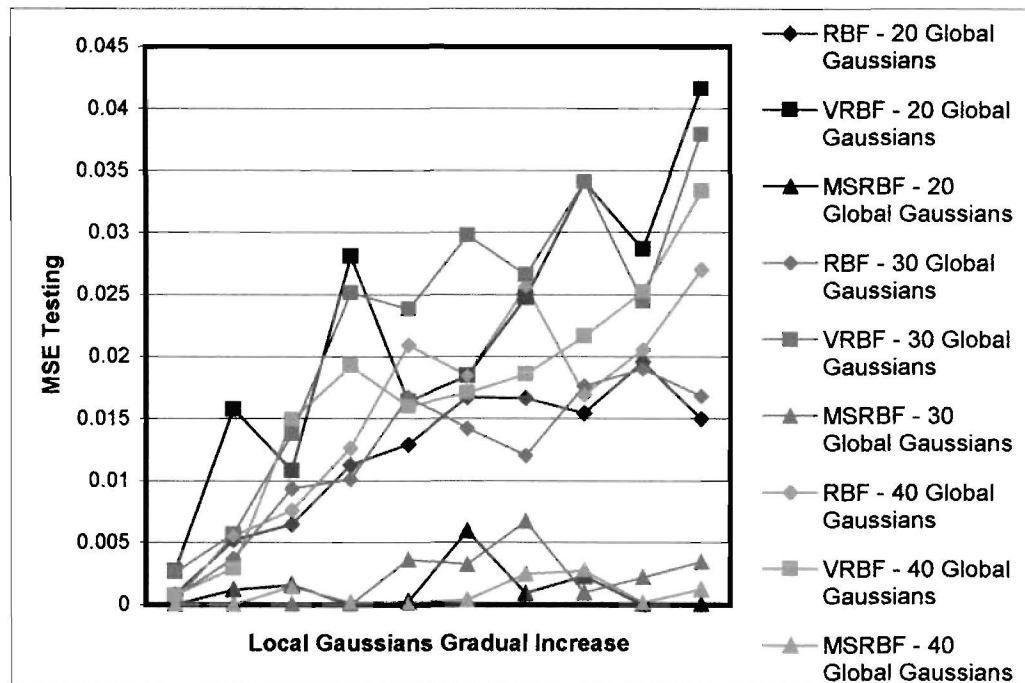


Figure 7.28: Local gaussians effect in the testing MSE

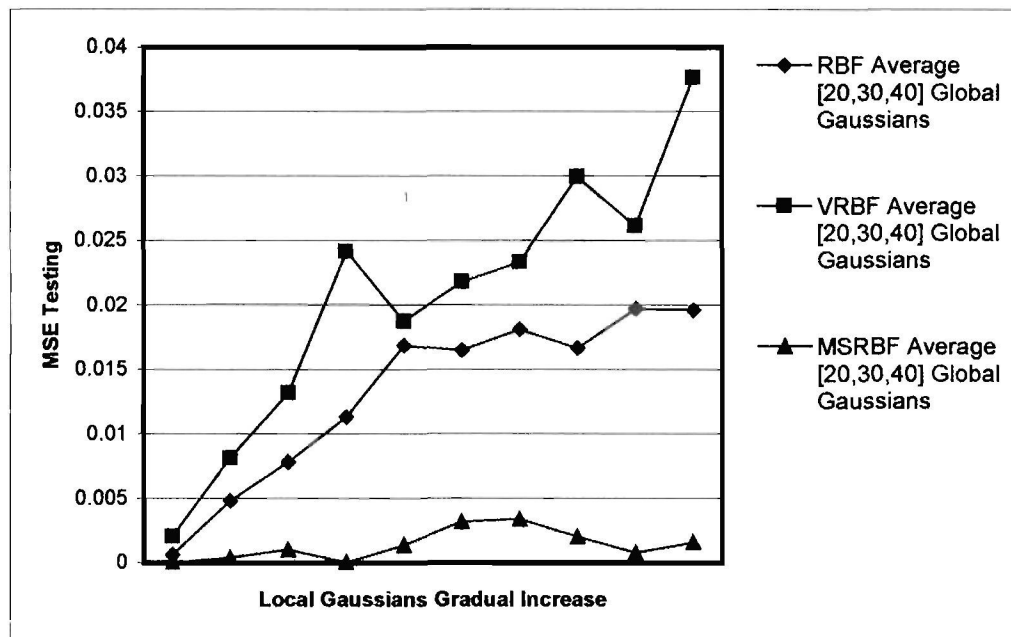


Figure 7.29: Summarized local gaussians effect in the testing MSE

7.4 Conclusion

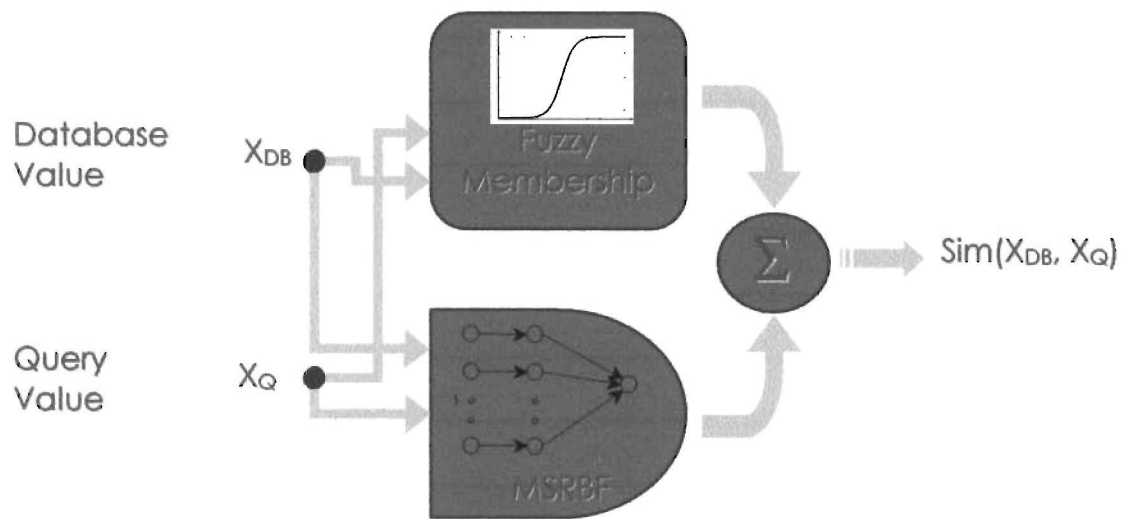
In this chapter we presented the training process of our system. We also discussed the idea behind the progressive training. The similarity input/output presentation aimed at facilitating users with various levels of expertise as they interact with our system at the training input and simulation output.

Our system evaluation was based on functionality examples and statistical simulations. The functionality examples showed the advanced modeling capabilities our similarity preference learning algorithm exhibits. The statistical simulations were performed in the two parts of our system, the fuzzy functions and the MSRBF neural network. The fuzzy functions showed stability in convergence and also exemplified the importance of good initial approximations that our system provides through the progressive training. The MSRBF neural network was compared with two other approaches, the traditional radial basis function (RBF) with single spread, and a variable spread RBF. The results showed that our network outperformed consistently the other two methods in terms of modeling accuracy, and also in simulation times as it requires less number of nodes.

Chapter 8

Conclusions and future work

This chapter provides a brief overview of the techniques developed to address our problem. A more specific discussion follows on the research contributions of this work and the overall benefit of this thesis. Future work that will enhance applicability and performance of our system is also presented.



8.1 Thesis synopsis

In this thesis we addressed a common problem in geospatial queries, the lack of relevance of results based on user/application needs. The current approaches are deterministic and do not allow the incorporation of user preference in the query process. In contrast to that, our proposed algorithm adjusts query returns based on similarity profiles that express more accurately user anticipation of relevant returns.

Our work focuses on learning preference within one-dimensional, quantitative attributes. The learning process is based on a training dataset provided by the user. Depending on the provided similarity preference complexity our algorithm adjusts the learning process. At the first stage, fuzzy membership functions are interpolated to capture the dominant preference (signal). Several families of functions are used progressively, from simple planar functions to complex sigmoidal ones. The design of the algorithm allows previously interpolated functions to act as approximations for more complex ones that follow, therefore decreasing training time and increasing robustness.

During the next stage of our training, a customized neural network is used, specifically developed to express the characteristics of the problem. We attempt to model potential errors that resulted from the interpolation of the fuzzy functions; we do not want our neural network to expand to portions of the input space without significant evidence. Therefore, our network design forces it to operate in a localized manner and only where it is necessary. The idea behind this is that we trust the fuzzy functions to carry the signal throughout the input space with a more predictable modeling, and let the neural network capture more unpredictable user behavior. Our neural network, called Multi-Scale Radial Basis Function (MSRBF) network, offers an innovative design and training by inserting local accuracy metrics (i.e. within node receptive fields) in the traditional global ones (i.e.

throughout the input space).

At the last training stage, the fuzzy functions are combined with MSRBF into one solution. Weights are readjusted to facilitate even higher accuracy in this neuro-fuzzy system. In some cases, if found appropriate, the fuzzy functions go through a self-organizing process, where they adjust even more to the overwhelming signal, expecting the MSRBF to provide the localized corrections.

The proposed neuro-fuzzy system outperforms the currently used distance-based nearest neighbor methods. It does so by design, since it recognizes and supports distance dependent preference, but at the same time, offers advanced modeling capabilities as we have seen in the examples of chapters 4 and 7. Also in chapter 7 we demonstrated the robustness of the system through statistical simulations and how our algorithm adjusts its complexity as the similarity preference is getting more complicated.

8.2 Research contributions

The neuro-fuzzy similarity learning system offers a significant improvement in the query process of geospatial information. By using our system, complex preference expression can be modeled and incorporated when users perform queries on geospatial information. The results are customized to specific user/application needs and restrictions, and thus, more accurate information retrieval is achieved.

During the development of the neuro-fuzzy system several novel methodologies were introduced:

- **A collection of fuzzy membership functions that not only express user preference in a successful way (as examples showed) but which also allow the interaction between successive functions.** The less complex functions act as an approximation

for more complex ones. By doing so the number of iterations is reduced, while the convergence rate is higher.

- **A novel customized version of the traditional Radial Basis Function neural network.** Even though our initial intention was to use the original RBF network, as the investigation on the desired characteristics of the network progressed, some significant modifications were made, leading to our Multi-Scale RBF. The distinction of our developed method is twofold. Firstly, the training process uses different criteria to accept node candidates. In addition to the traditional global error fit measure, we use a local error fit metric. This allows some smaller scale (but still strong) signals to be absorbed first and potential larger scale signals to reveal themselves in subsequent iterations. Secondly, the architecture of the MSRBF is different with the addition of a blocking layer to make sure that these local scale signals do not influence the overall solution.
- **The semantic basis of our neuro-fuzzy architecture for similarity preference.** The fuzzy functions are used to model “expected” user preference and they propagate it throughout the input space. The MSRBF neural network is used whenever “unexpected” behavior is identified during training and as an error improvement method. The idea behind this comes from the original motivation behind the combination of fuzzy and neural methods. The fuzzy methods are easy to interpret but not adaptable, and neural networks are adaptable but hard to interpret. Our adaptable fuzzy functions using a back-propagation algorithm and our easier to interpret MSRBF (due to its localized use) coupled together produce a powerful model.

8.3 Future work

Future directions of this research can be classified in two major categories. The first one includes the development of modules not treated in this thesis in order to have a complete similarity learning system for geospatial information retrieval. The second category involves the applicability of the methodologies used for similarity learning to various machine learning tasks.

8.3.1 Extensions of our learning system

Our algorithm calculates similarity in one-dimensional, quantitative attributes. A logical extension of this work would be the calculation of similarity in other non-supported attributes, and then, the combination of these techniques with ours to produce a total similarity metric. It is interesting to see whether our method can be scaled up to more than one-dimensional, quantitative attributes. For example, similarity learning in space would be a challenging task. Also other highly dependent attributes such as color could be a potential candidate for our system. Furthermore, similarity learning in qualitative attributes should be addressed. There are already successful examples of algorithms used for text retrieval that could be extended to facilitate geospatial qualitative attributes.

The most challenging task remains the combination of similarity results from each individual attribute to one total metric. This increased difficulty is attributed to two main reasons:

- i) The dimensionality can be high and most importantly not predetermined, which means that the algorithm should be able to easily scale up in dimensions without being prohibitively slow and require retraining when a new attribute is added.
- ii) Dependencies might exist between dimensions, making this task even harder.

8.3.2 Application of our learning methods to other domains

Another appealing future investigation would be to see whether our system can be applied towards machine learning tasks in different domains. The idea that you have a set of functions describing an expected signal and then another set to model unexpected behavior is easily extendable to other domains. Our neuro-fuzzy system does support that separation; it is actually built on that.

The application of the MSRBF to other signal modeling tasks can also yield some interesting results. We see our neural network as an improved version of the traditional RBF networks. It can be transformed into the traditional version with the appropriate selection of input parameters, or at the same time, it can investigate additional node selections based on their local errors in addition to the global ones. A good example of such implementation would be analysis of time-series data.

Bibliography

- Aha D. (1992) Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36, pp. 267-287.
- Anselin L. (1995) Local indicators of spatial association. *Geographical Analysis*, 27, pp. 93-115.
- Atkeson C. G., A. W. Moore, and S. Schaal (1997) Locally Weighted Learning. 11(1-5), pp. 11-73.
- Bareiss E. R. and B. W. Porter (1988) PROTOS: an exemplar-based learning apprentice. *International Journal of Man-Machine Studies*, 29, pp. 549-561.
- Batchelor B. G. (1978) *Pattern Recognition: Ideas in Practice*. Plenum Press, New York.
- Bayes T. (1763) An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53, pp. 370-418.
- Biberman Y. (1994) A Context Similarity Measure. In: *European Conference on Machine Learning*, Catalina, Italy, pp. 49-63.
- Bishop C. M. (1995) *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK.
- Boca Raton T. B. (1999) *Data Mining: Technologies, Techniques, Tools, and Trends*. CRC Press, pp. 270.
- Breiman L., J. H. Friedman, R. A. Olshen, and C. J. Stone (1984) *Classification and regression trees*. Wadsworth International Group, Belmont, CA.
- Brunsdon C., A. S. Fotheringham, and M. E. Charlton (1996) Geographically weighted regression: A method for exploring spatial nonstationarity. *Geographical Analysis*, 28, pp. 281-298.
- Burke R., K. Hammond, and R. Young (1997) The FindMe Approach to Assisted Browsing. *IEEE Expert*, 12(4), pp. 32-40.
- Carenini G. and D. Poole (2002) Constructed Preferences and Value-focused Thinking: Implications for AI research on Preference Elicitation. In: *AAAI-02 Workshop on Preferences in AI and CP: symbolic approaches*, Edmonton, Canada.
- Carkacioglu A. and F. Y. Vural (2002) Learning Percieved Similarity. In: *International Conference on Image Processing (ICIP)*, Rochester NY.

- Carpenter G. A., S. Grossberg, and D. B. Rosen (1991) Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4, pp. 759-771.
- Cendrowska J. (1987) PRISM: an algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27, pp. 349-370.
- Chajewska U., L. Getoor, J. Norman, and Y. Shahar (1998) Utility Elicitation as a Classification Problem. In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, Wisconsin, USA, pp. 79-88.
- Chandrasekaran V., Z. Q. Liu, and M. Palaniswami (1995) Fuzzy gated neuronal architecture for pattern recognition. In: *IEEE International Conference on Neural Networks*, Perth, Australia, pp. 1622-1627.
- Chawla S., S. Shekhar, W. L. Wu, and U. Ozesmi (2001) Modeling spatial dependencies for mining geospatial data: An introduction. In: *Geographic Data Mining and Knowledge Discovery*, H. J. Miller and J. Han, (Eds.), Taylor and Francis, London, pp. 131-159.
- Chen H. (1995) Machine Learning for Information Retrieval: Neural Networks, Symbolic Learning, and Genetic Algorithms. *Journal of the American Society for Information Science*, 46(3), pp. 194-216.
- Chen L. and P. Pu (2004) Survey of Preference Elicitation Methods, *Technical Report No. IC/200467*, Swiss Federal Institute of Technology in Lausanne (EPFL), Lausanne, Switzerland.
- Clarke P. and T. Niblett (1987) Induction in noisy domains. In: *Progress in Machine Learning*, I. Bratko and N. Lavrac, (Eds.), Sigma Press, Wilmslow, England, pp. 11-30.
- Codd E. F. (1970) A Relational Model of Data for Large Shared Data Banks. *Communications of ACM*, 13(6), pp. 377-387.
- Cost S. and S. Salzberg (1993) A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. *Machine Learning*, 10, pp. 57-78.
- Craven P. and G. Wahba (1979) Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik*, 31, pp. 377-403.
- Crestani F. and G. Pasi (1999) Soft Information Retrieval: applications of fuzzy sets theory and neural networks. In: *Neuro-fuzzy Techniques for Intelligent Information Systems*, N. Kasabov and R. Kozma, (Eds.), Physica Verlag, Heidelberg, Germany, pp. 287-315.

- Dempster A., N. Laird, and D. Rubin (1977) Maximum Likelihood from Incomplete Data via the EM algorithm. *Journal of the Royal Statistical Society*, 39, pp. 1-38.
- Diday E. (1974) Recent Progress in Distance and Similarity Measures in Pattern Recognition. In: *Second International Joint Conference on Pattern Recognition*, pp. 534-539.
- Dietterich T. G. and G. Bakiri (1995) Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research*, 2, pp. 263-286.
- Domingos P. (1995) Rule Induction and Instance-Based Learning: A Unified Approach. In: *Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Canada, pp. 1226-1232.
- Dubois D., H. Prade, and R. R. Yager (1996) Information Engineering and Fuzzy Logic. In: *5th IEEE International Conference on Fuzzy Systems*, New Orleans, LA, pp. 1525-1531.
- Dunham M. H. (2002) *Data Mining: Introductory and Advanced Topics*. Prentice Hall.
- Egenhofer M. J. and J. R. Herring (1994) Categorizing binary topological relations between regions, lines and points in geographic databases. In: *The 9-intersection: Formalism and its Use for Natural-language Spatial Predicates*, M. Egenhofer, D. M. Mark, and J. R. Herring, (Eds.), National Center for Geographic Information and Analysis Technical Report 94-1, Santa Barbara, CA, pp. 1-28.
- Egenhofer M. J. and K. Hornsby (2000) Identity-based change: A foundation for spatio-temporal knowledge representation. *International Journal of Geographical Information Science*, 14, pp. 207-224.
- Elder J. and D. Pregibon (1996) A statistical perspective on KDD. In: *Advances in Knowledge Discovery and Data Mining*, U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, (Eds.), AAAI/MIT Press, Cambridge, MA,
- Fayyad U. M., G. Piatetsky-Shapiro, and P. Smyth (1996) From data mining to knowledge discovery: An overview. In: *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. P.-. Shapiro, P. Smyth, and R. Uthurusamy, (Eds.), AAAI Press/MIT Press, Menlo Park, CA, pp. 1-34.
- Feigenbaum E. and J. Feldman (1963) *Computers and thought*. McGraw-Hill, New York.
- Fisher R. A. (1921) On the probable error of a coefficient of correlation deduced from a small sample. *Metron International Journal of Statistics*, 1(4), pp. 3-32.
- Fix E. and J. L. Hodges (1951) *Discriminating analysis: non-parametric distribution*. USAF School of Aviation Medicine, Randolph Field, TX, Technical Report 21-49-004(4).

- Fix E. and J. L. Hodges (1952) *Discriminating analysis: small sample performance*. USAF School of Aviation Medicine, Randolph Field, TX, Technical Report 21-49-004(11).
- Florek K., J. Lukaszewicz, J. Perka, H. Steinhaus, and S. Zubrzycki (1951) Taksonomia wroclawska. *Przegląd Antropologiczny*, 17(4), pp. 93-207.
- Fotheringham A. S., M. Charlton, and C. Brunsdon (1997) Two techniques for exploring non-stationarity in geographical data. *Geographical Systems*, 4, pp. 59-82.
- Frayman Y., K. M. Ting, and L. Wang (1999) A Fuzzy Neural Network for Data Mining: Dealing with the Problem of Small Disjuncts. In: *International Joint Conference on Neural Networks*, Washington, D.C.
- Friedman J. H. (1995) An overview of predictive learning and function approximation. In: *From Statistics to Neural Networks: Theory and Pattern recognition Applications*, V. Cherkassky, J. H. Friedman, and H. Wechsler, (Eds.), Springer, Berlin, pp. 1-61.
- Gahegan M. (2001) *Intersection of Geospatial Information and Information Technology*. National Academies white paper.
- Getis A. and J. K. Ord (1992) The analysis of spatial association by use of distance statistics. *Geographical Analysis*, 24, pp. 189-206.
- Getis A. and J. K. Ord (1996) Local spatial statistics: An overview. In: *Spatial Analysis: Modelling in a GIS Environment*, P. Longley and M. Batty, (Eds.), GeoInformation International, Cambridge, UK, pp. 261-277.
- Goldberg D. E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA.
- Goodchild M. F. (1992) Geographical data modeling. *Computers and Geosciences*, 18(4), pp. 401-408.
- Green P. J. and B. W. Silverman (1994) *Nonparametric Regression and Generalized Linear Models: A roughness penalty approach*. Chapman & Hall, London.
- Griffiths A. and D. G. Bridge (1997) Towards a Theory of Optimal Similarity Measures. In: *Third UK Workshop on Case-Based Reasoning*, Salford.
- Gunopulos D. (2001) *Data mining techniques for geospatial applications*. National Academies white paper.
- Ha V. and P. Haddawy (1998) Toward Case-Based Preference Elicitation: Similarity Measures on Preference Structures. In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, Wisconsin, USA, pp. 193-201.

- Halgamuge S. K. and M. Glesner (1994) Neural Networks in Designing Fuzzy Systems for Real World Applications. *Fuzzy Sets and Systems*, 65(1), pp. 1-12.
- Halgamuge S. K., A. Mari, and M. Glesner (1994) Fast Perceptron Learning by Fuzzy Controlled Dynamic Adaptation of Network Parameters. In: *Fuzzy Systems in Computer Science*, R. Kruse, J. Gebhardt, and R. Palm, (Eds.), Vieweg Verlag, Wiesbaden, pp. 129-139.
- Han J. and M. Kamber (2001) *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, CA.
- Hand D. J. (1981) *Discrimination and Classification*. Wiley, New York.
- Hand D., H. Mannila, and P. Smyth (2001) *Principles of Data Mining*. The MIT Press. pp. 425.
- Hardy R. L. (1971) Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysics Research*, 76, pp. 1905-1915.
- Hastie T., R. Tibshirani, and J. Friedman (2001) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York.
- Haykin S. (1994) *Neural Networks*. Prentice Hall, New Jersey, NJ.
- Haykin S. (1999) *Neural Networks. A Comprehensive Foundation*. Second ed., Prentice Hall, New Jersey, NJ.
- Hill W., L. Stead, M. Rosenstein, and G. Furnas (1995) Recommending and evaluating choices in a virtual community of use. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, Denver, Colorado, USA, ACM Press, pp. 194-201.
- Holland J. H. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.
- Howe N. and C. Cardie (1997) Examining locally varying weights for nearest neighbor algorithms. In: *Second International Conference on Case-Based Reasoning*, Berlin, pp. 455-466.
- Hunt E. B., J. Marin, and P. Stone (1966) *Experiments in Induction*. Academic Press, New York.
- Ishibuchi H., K. Morioka, and I. B. Turksen (1995) Learning by fuzzified neural networks. *International Journal of Approximate Reasoning*, 13(4), pp. 327-358.
- Ishii N. and Y. Wang (1998) Learning Feature Weights for Similarity Measures using Genetic Algorithms. In: *IEEE Intl Joint Symposia on Intelligence and Systems*, Rockville, Maryland, pp. 27-33.

- Jang J. S. (1993) ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Transactions on Systems, Man and Cybernetics*, 23(3), pp. 665-684.
- Johns M. V. (1961) An empirical Bayes approach to non-parametric two-way classification. In: *Studies in Item Analysis and Prediction*, H. Solomon, (Ed.) Stanford University Press,
- Kanal L. N. (1963) *Statistical methods for pattern classification*. U.S. Army Electronics Research and Development lab, Philco Report VO43-2 and VO43-3.
- Kantardzic M. (2002) *Data Mining: Concepts, Models, Methods, and Algorithms*. Wiley-IEEE Press.
- Keeney R. L. and H. Raiffa (1976) *Decisions with Multiple Objectives: Preferences and value tradeoffs*. John Wiley and Sons, New York, NY.
- Kibler D. and D. Aha (1987) Learning representative exemplars of concepts: an initial case study. In: *Fourth International Workshop on Machine Learning*, Irvine, CA, pp. 24-30.
- Klawonn F. and A. Keller (1997) Fuzzy Clustering and Fuzzy Rules. In: *7th International Fuzzy Systems Association World Congress*, Academia, Prague, pp. 193-198.
- Klose A., A. Nürnberger, D. Nauck, and R. Kruse (2001) Data Mining with Neuro-Fuzzy Models. In: *Data Mining and Computational Intelligence*, A. Kandel, H. Bunke, and M. Last, (Eds.), Physica-Verlag, Berlin, pp. 1-36.
- Kohonen T. (1982) Self-organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics*, 43, pp. 59-69.
- Kolodner J. L. (1984) *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model*. Lawrence Erlbaum Associates, Hillsdale, NJ.
- Korn F., B. U. Pagel, and C. Faloutsos (2001) On the Dimensionality Curse and the Self-Similarity Blessing. *IEEE Transactions on Knowledge and Data Engineering*, 13(1), pp. 96-111.
- Lebowitz M. (1987) Experiments with universal concept formation: UNIMEM. *Machine Learning*, 2, pp. 103-138.
- Lim J. H., J. K. Wu, S. Singh, and A. D. Narasimhalu (2001) Learning Similarity Matching in Multimedia Content-Based Retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 13(5), pp. 846-850.
- Lin C. T. and C. C. Lee (1996) *Neural Fuzzy Systems. A Neuro-Fuzzy Synergism to Intelligent Systems*. Prentice Hall, New York.

- Linden G., S. Hanks, and N. Lesh (1997) Interactive Assessment of User Preference Models: The Automated Travel Assistant. In *User Modeling: Proceedings of the Sixth International Conference*, A. Jameson, C. Paris, and C. Tasso (Eds.) Vienna, Austria, pp. 67-78.
- Liu Z. Q. and S. Miyamoto (2000) *Soft Computing and Human-Centered Machines*. Springer, Tokyo.
- Lu H., R. Setiono, and H. Liu (1995) Neurorule: A connectionist approach to data mining. In: *21st International Conference on Very Large Data Bases (VLDB)*, Zurich, Switzerland.
- Ma W. Y. and B. S. Manjunath (1996) Texture Features and Learning Similarity. In: *IEEE International Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, pp. 425-430.
- Mandl T. (2000) Tolerant Information Retrieval with Backpropagation Networks. *Neural Computing & Applications. Special Issue on Neural Computing in Human-Computer Interaction*, 9(4), pp. 280-289.
- Martin B. (1995) *Nearest neighbours with generalization*. M.Sc. Thesis, University of Waikato, Hamilton, New Zealand.
- McCulloch W. S. and W. Pitts (1943) A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, pp. 115-133.
- McLachlan G. J. (1992) *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley & Sons, New York.
- Miller H. J. and H. Jiawei (2001) Geographic data mining and knowledge discovery: An overview. In: *Geographic Data Mining and Knowledge Discovery*, H. J. Miller and J. Han, (Eds.), Taylor and Francis, London, pp. 3-32.
- Mitaim S. and B. Kosko (1997) Neural Fuzzy Agents that Learn a User's Preference Map. In: *4th International Forum on Research and Technology Advances in Digital Libraries*, Washington, DC, pp. 25-35.
- Mitchell T. M. (1980) The Need for Biases in Learning Generalizations. In: *Readings in Machine Learning*, J. W. Shavlik and T. G. Dietterich, (Eds.), Morgan Kaufmann, San Mateo, CA, pp. 184-191.
- Montesi D. and A. Trombetta (1999) Similarity Search through Fuzzy Relational Algebra. In: *International Workshop on Database and Expert Systems Applications (DEXA'99)*, Florence, Italy, pp. 235-239.
- Mountrakis G. and P. Agouris (2003) Learning Similarity with Fuzzy Functions of Adaptable Complexity. In: *8th International Symposium on Spatial and Temporal Databases*, Lecture Notes in Computer Science, Vol. 2750, pp. 412-429.

- Nadler M. and E. P. Smith (1993) *Pattern Recognition Engineering*. Wiley, New York.
- Narazaki H. and A. L. Ralescu (1991) A Synthesis Method for Multi-Layered Neural Network using Fuzzy Sets. In: *Workshop on Fuzzy Logic in Artificial Intelligence*, Sydney, pp. 54-66.
- Natsev A., R. Rastogi, and K. Shim (2004) WALRUS: A Similarity Retrieval Algorithm for Image Databases. *IEEE Transactions on Knowledge and Data Engineering*, 16 (3), pp. 301-316.
- Nauck D. and R. Kruse (1995) NEFCLASS - A Neuro-Fuzzy Approach for the Classification of Data. In: *ACM Symposium on Applied Computing*, Nashville, pp. 461-465.
- Nauck D., F. Klawonn, and R. Kruse (1997) *Foundations of Neuro-Fuzzy Systems*. Wiley, Chichester.
- Nguyen, H. and P. Haddawy (1999) The Decision-Theoretic Video Advisor. In: *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, Stockholm, Sweden, pp. 77-80.
- Nilsson N. J. (1965) *Learning Machines: Foundations of trainable pattern-classifying systems*. McgrawHill, New York.
- Okabe A. and H. J. Miller (1996) Exact computational methods for calculating distances between objects in a cartographic database. *Cartography and Geographic Information Systems*, 23, pp. 180-195.
- Peuquet D. J. and C.-X. Zhang (1987) An algorithm to determine the directional relationship between arbitrarily-shaped polygons in the plane. *Pattern Recognition*, 20, pp. 65-74.
- Plataniotis K. N., and A. N. Venetsanopoulos (2000) *Color Image Processing and Applications*. Springer Verlag, New York, NY, USA, pp. 335.
- Quinlan J. R. (1986) Induction of Decision Trees. *Machine Learning*, 1, pp. 81-106.
- Quinlan J. R. (1993) *C4.5: Program for Machine Learning*. Morgan Kaufmann, San Mateo, CA.
- Rachlin J., S. Kasif, S. Salzberg, and D. W. Aha (1994) Towards a Better Understanding of Memory-Based and Bayesian Classifiers. In: *Eleventh International Machine Learning Conference*, New Brunswick, NJ, pp. 242-250.
- Rashid A. M., I. Albert, D. Cosley, S .K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl (2002) Getting to know you: Learning new user preferences in recommender systems. In: *Proceedings of the International Conference on Intelligent User Interfaces*, San Francisco, California, USA, pp. 127-134.

- Reed R. (1993) Pruning algorithms: A survey. *IEEE Transactions on Neural Networks*, 4(5), pp. 740-747.
- Resnick P., N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl (1994) GroupLens: An open architecture for collaborative filtering of netnews. In: *Proceedings of 1994 Conference on Computer Supported Cooperative Work*, New York City, New York, USA, ACM Press, pp. 175-186.
- Roddick J. F. and B. Lees (2001) Paradigms for spatial and spatio-temporal data mining. In: *Geographic Data Mining and Knowledge Discovery*, H. J. Miller and J. Han, (Eds.), Taylor and Francis, London,
- Roddick J. F. and M. Spiliopoulou (1999) A bibliography of temporal, spatial and spatio-temporal data mining research. *SIGKDD Explorations*, 1(1), pp. 34-38.
- Roddick J., K. Hornsby, and M. Spiliopoulou (2001) An updated bibliography of temporal, spatial and spatio-temporal data mining research. In: *International Workshop on Temporal, Spatial, and Spatio-temporal Data Mining, TSDM2000*, Lyon, France, pp. 147-163.
- Rosenblatt F. (1958) The Perceptron : A Probabalistic Model for information Storage and Organization in the Brain. *Psychological Review*, 65, pp. 386-408.
- Rumelhart D. E. and P. M. Todd (1993) Learning and connectionist representations. In: *Attention and performance*, vol. XIV, D. E. Meyer and S. Kornblum, (Eds.), MIT Press/Bradford Books, Cambridge, MA, pp. 3-30.
- Salton G. (1971) *The Smart Retrieval System - Experiments in Automatic Document Processing*. Prentice-Hall, Englewood Cliffs, NJ.
- Salzberg S. (1991) A Nearest Hyperrectangle Learning Method. *Machine Learning*, 6, pp. 277-309.
- Santini S. and R. Jain (1997) Image databases are not databases with images. In: *9th International Conference on Image Analysis and Processing*, Florence, Italy, pp. 38-45.
- Santini S. and R. Jain (1999) Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Learning*, 21(9), pp. 871-883.
- Satty T. L. (1994) *Fundamentals of Decision Making and Priority Theory with the AHP*. RWS Publications, Pittsburgh, PA.
- Schafer J. B., J. A. Konstan, and J. Riedl (2001) E-Commerce Recommender Applications. *Data Mining and Knowledge Discovery*, 5, pp. 115-152.
- Schaffer C. (1994) A Conservation Law for Generalization Performance. In: *Eleventh International Conference on Machine Learning*, San Francisco, CA.

- Scott D. W. (1992) *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley, New York.
- Shardanand U. and P. Maes (1995) Social Information Filtering: Algorithms for automating “word of mouth”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, Denver, Colorado, USA, ACM Press, pp. 210-217.
- Shearin S. and H. Lieberman (2001) Intelligent Profiling by Example. In: *Proceedings of the Conference of Intelligent User Interfaces*, Santa Fe, New Mexico, USA, ACM Press, pp. 145-151.
- Shepard R. N. (1987) Toward a Universal Law of Generalization for Psychological Science. *Science*, 237, pp. 1317-1323.
- Shimazu H. (2001) ExpertClerk: Navigating Shoppers’ Buying Process with the Combination of Asking and Proposing. In: *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, Seattle, Washington, USA, pp. 1443-1448.
- Siekmann S., R. Kruse, R. Neuneier, and H. Zimmermann (1997) Advanced Neuro-Fuzzy Techniques Applied To The German Stock Index DAX. In: *2nd European Workshop on Fuzzy Decision Analysis and Neural Networks*, Dortmund, Germany, pp. 170-179.
- Stanfill C. and D. Waltz (1986) Toward memory-based reasoning. *Communications of the ACM*, 29, pp. 1213-1228.
- Stevens, S. S. (1946) On the Theory of Scales of Measurement. *Science*, 103(2684), pp. 677-680.
- Tettamanzi A. and M. Tomassini (2001) *Soft Computing: Integrating Evolutionary, Neural, and Fuzzy Systems*. Springer-Verlag, Heidelberg.
- Tijsseling A. and S. Harnad (1997) Warping Similarity Space in Category Learning by Backprop Nets. In: *Interdisciplinary Workshop on Similarity and Categorization*, Edinburgh, pp. 263-269.
- Tversky A. (1977) Features of Similarity. *Psychological Review*, 84(4), pp. 327-352.
- Veltkamp R. C., and M. Hagedoorn (2001) State of the art in shape matching. Michael S. Lew (Ed.), *Principles of Visual Information Retrieval*, Series in Advances in Pattern Recognition, Springer, pp. 87-119.
- Vlachos M., D. Gunopulos, and G. Kollios (2002) Robust Similarity Measures for Mobile Object Trajectories. In: *13th International Workshop on Database and Expert Systems Applications*, Aix-en-Provence, France, pp. 721-728.

- Wahba G. (1990) *Spline models for Observational Data*. vol. 59, SIAM Press, Philadelphia.
- Wang Y. and N. Ishii (1997) A genetic algorithm for learning weights in a similarity function. In: *International Conference on Artificial Neural Networks and Genetic Algorithms*, Norwich, UK, pp. 206-209.
- Weber R., H. Schek, and S. Blott (1998) A quantitative analysis and performance study for Similarity Search Methods in High Dimensional Spaces. In: *24th International Conference on Very Large Data Bases (VLDB)*, New York City, NY, pp. 194-205.
- Weiss G. M. and H. Hirsh (2000) A Quantitative Study of Small Disjuncts. In: *Seventeenth National Conference on Artificial Intelligence*, Austin, TX, pp. 665-670.
- Weiss S. M. and C. A. Kulikowshi (1991) *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Expert Systems*. Morgan Kaufmann, San Mateo, CA.
- Wettschereck D. and T.G. Dietterich (1992) Improving the performance of radial basis function networks by learning center locations. In Moody, J. E., Hanson, S. J., and Lippmann, R. P. (Eds.) *Advances in Neural Information Processing Systems*, 4, San Mateo, CA, Morgan Kaufmann, pp. 1133-1140.
- Wettschereck D., D. W. Aha, and T. Mohri (1995) *A Review and Comparative Evaluation of Feature Weighting Methods for Lazy Learning Algorithms*. Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, Washington, D.C, Technical Report AIC-95-012.
- Wilson D. and T. Martinez (1997) Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6, pp. 1-34.
- Xu Y., D. Ruan and J. Liu (1999) Uncertainty automated reasoning in intelligent learning of soft knowledge. In: *ICST Workshop Information and Communication Technology for Teaching and Training*, Gent, Belgium, pp. 29-45.
- Yuan M., B. Battenfield, M. Gahegan, and H. Miller (2001) *Geospatial Data Mining and Knowledge Discovery*. A UCGIS White Paper on Emergent Research Themes.
- Zadeh L. A. (1972) A fuzzy-set-theoretic interpretation of linguistic hedges. *Journal of Cybernetics*, 2(3), pp. 4-34.

Appendix: Statistical simulations results

G	L	MSE Training *10 ⁻⁶			MSE Testing *10 ⁻⁶			Nodes		
		RBF	VRBF	MSRBF	RBF	VRBF	MSRBF	RBF	VRBF	MSRBF
2	1	21.2	12097.0	2.3	46061.0	41803.0	47.2	19	3	3
2	2	15.4	3209.0	3.1	11148.0	6885.9	2579.4	34	12	4
2	3	3.1	23404.0	55.1	17917.0	37912.0	1731.2	44	13	5
2	4	13.7	50443.0	5.8	14957.0	35456.0	50.2	49	12	6
4	1	14.3	22848.0	0.6	31740.0	7539.3	2.0	45	6	5
4	2	9.5	78690.0	2.6	14963.0	23699.0	2.4	45	13	6
4	3	11.0	41339.0	0.6	24901.0	38242.0	3718.8	59	11	7
4	4	9.9	37015.0	4.6	19155.0	37100.0	4.0	64	12	8
4	5	8.0	8814.8	9.7	13977.0	12430.0	12.4	81	21	9
4	6	8.5	43383.0	9.8	44976.0	28314.0	6187.9	89	23	11
4	7	6.5	32179.0	8855.6	12118.0	33117.0	3286.9	90	26	31
4	8	2.0	28136.0	4.1	23356.0	27814.0	7.2	94	23	14
6	1	9.2	958.8	0.4	13414.0	1682.7	2.5	59	15	7
6	3	6.2	21529.0	1.0	12225.0	17392.0	3.3	72	17	9
6	5	7.3	11849.0	5.3	41988.0	16540.0	9.9	97	25	11
6	7	19.0	12351.0	8892.0	27894.0	16099.0	4014.5	99	13	13
6	9	6.4	15602.0	2227.8	13370.0	24515.0	761.4	99	45	29
6	11	2929.2	32344.0	7.5	18743.0	42417.0	3793.3	100	17	18
8	1	7.7	1467.4	1.2	12404.0	2635.7	2.3	95	16	9
8	3	58.8	11288.0	1.9	50911.0	19471.0	2.9	46	15	11
8	5	658.2	13358.0	2.2	12532.0	26213.0	6.7	57	25	14
8	7	161.7	6570.2	3.5	51496.0	17369.0	8.0	46	39	16
8	9	240.0	16121.0	6626.6	25066.0	30930.0	10400.0	49	45	24
8	11	958.6	34830.0	4.1	22085.0	35580.0	9.1	52	51	21
8	13	8060.8	16051.0	491.4	13122.0	27936.0	1224.2	100	57	24
8	15	2420.1	24920.0	2.7	10863.0	37446.0	10202.0	100	63	28
10	1	28.0	11750.0	0.5	30206.0	7720.5	2.6	41	12	11
10	3	6.2	3530.3	1.1	6615.0	8668.2	4.2	43	19	13
10	5	53.4	12993.0	2.8	8338.3	9609.8	4.1	48	45	15
10	7	2467.5	30219.0	2.1	13862.0	28894.0	5.7	100	31	18
10	9	4150.3	12154.0	3.0	13784.0	22586.0	611.2	100	51	22
10	11	3860.0	13507.0	5009.3	15230.0	16574.0	7812.7	100	57	39
10	13	1545.5	24730.0	4089.5	60113.0	22044.0	2725.8	100	53	53
10	15	1169.7	21411.0	1727.4	17254.0	20185.0	2451.4	100	75	35
10	17	6344.2	22689.0	2.8	14783.0	38798.0	14.9	100	49	32
10	19	6267.6	44040.0	4.9	14233.0	41089.0	9.8	100	51	35

Table A1: Detailed results of statistical simulations

G	L	MSE Training *10 ⁻⁶			MSE Testing *10 ⁻⁶			Nodes		
		RBF	VRBF	MSRBF	RBF	VRBF	MSRBF	RBF	VRBF	MSRBF
12	1	20.0	29382.0	1.3	2982.4	1681.0	5.7	33	17	13
12	4	206.1	17641.0	1.8	8497.6	21965.0	4.3	37	16	16
12	7	3694.1	20031.0	1.6	7319.7	14494.0	1106.0	100	35	21
12	10	5113.5	24815.0	7.3	10184.0	33187.0	11.2	100	49	23
12	13	4650.9	18278.0	867.0	11948.0	22179.0	1708.5	100	75	29
12	16	4112.2	19349.0	4268.5	18784.0	28614.0	7337.6	100	64	40
12	19	5371.7	22069.0	8.7	16899.0	28112.0	10.6	100	57	33
12	22	8493.8	26908.0	8.3	16226.0	32003.0	42.3	100	62	37
14	1	202.4	15598.0	0.1	18537.0	1030.3	2.4	33	31	15
14	4	1414.0	31421.0	1.8	4975.3	17654.0	4.2	100	28	18
14	7	7505.1	10712.0	353.8	10648.0	11488.0	1256.1	100	39	24
14	10	2271.6	11894.0	2.8	13237.0	25634.0	7210.0	100	34	28
14	13	3226.4	21726.0	3.3	12943.0	23643.0	7033.7	100	63	31
14	16	6674.1	14277.0	1333.8	12615.0	22261.0	1184.3	100	42	42
14	19	5583.1	22201.0	682.3	14353.0	29032.0	3981.4	100	61	38
14	22	2753.9	20514.0	946.6	17173.0	26643.0	1953.3	100	51	45
14	25	7091.7	25542.0	1529.4	19324.0	30575.0	1906.2	100	87	55
14	28	4790.4	33267.0	3.2	17336.0	42355.0	903.0	100	110	50
16	1	267.2	6647.3	0.6	8736.2	3960.6	3.1	38	23	17
16	5	663.2	1775.1	1.1	6616.9	6967.2	5.9	42	30	21
16	9	3540.4	16404.0	671.5	12219.0	15334.0	709.8	100	65	27
16	13	3878.1	18266.0	847.9	8016.9	27302.0	616.4	100	41	32
16	17	4824.6	24571.0	856.5	14319.0	30442.0	6031.0	100	61	36
16	21	3984.1	19797.0	7.2	14363.0	23807.0	320.4	100	52	40
16	25	7725.1	15049.0	1910.9	15920.0	20933.0	2924.0	100	92	109
16	29	4781.8	30451.0	1149.7	18293.0	32798.0	1521.8	100	63	63
18	1	407.6	843.4	0.4	1449.9	1253.1	2.6	48	20	19
18	5	3442.7	9927.9	1.0	4586.8	13599.0	4.3	100	33	23
18	9	2831.1	9675.5	1.7	10439.0	9453.9	4.3	100	38	28
18	13	2434.9	35800.0	2635.2	14197.0	13173.0	3571.9	100	73	70
18	17	6450.5	29675.0	1.3	11726.0	24953.0	1598.9	100	90	41
18	21	7365.3	29091.0	956.3	17624.0	30942.0	1594.5	100	95	48
18	25	3429.9	26532.0	3.3	18663.0	31907.0	12.4	100	79	49
18	29	5348.4	26018.0	994.2	17594.0	26128.0	1550.1	100	104	58
18	33	6905.0	31142.0	5.2	16202.0	29578.0	1980.2	100	93	64

Table A1: Continued

G	L	MSE Training *10 ⁻⁶			MSE Testing *10 ⁻⁶			Nodes		
		RBF	VRBF	MSRBF	RBF	VRBF	MSRBF	RBF	VRBF	MSRBF
20	1	568.5	6272.6	0.5	820.3	2737.1	2.6	56	27	21
20	5	2776.8	20260.0	862.9	5215.4	15778.0	1206.7	100	25	25
20	9	2170.8	6357.5	206.7	6492.9	10820.0	1598.1	100	41	31
20	13	2314.7	22815.0	2.2	11267.0	28082.0	8.1	100	75	36
20	17	5849.9	17577.0	1.7	12929.0	16398.0	271.0	100	82	42
20	21	4879.6	11201.0	5.9	16725.0	18470.0	5941.6	100	109	44
20	25	6338.6	24597.0	559.9	16644.0	24758.0	934.2	100	117	53
20	29	8970.3	26689.0	1055.7	15427.0	34086.0	2292.7	100	129	69
20	33	10287.0	17382.0	5.5	19571.0	28659.0	8.2	100	97	63
20	37	7973.1	56114.0	2.7	15005.0	41627.0	34.8	100	140	70
30	1	184.7	372.7	2.9	645.9	2642.9	3.3	172	31	31
30	7	2641.4	3223.7	0.6	3726.9	5706.0	3.7	200	37	39
30	13	1471.6	7282.3	1.7	9366.6	13768.0	5.4	200	61	46
30	19	3021.5	19910.0	2.7	10109.0	25141.0	8.0	200	69	53
30	25	5468.0	12846.0	997.7	16718.0	23839.0	3629.7	200	55	76
30	31	5976.4	26495.0	873.5	14233.0	29768.0	3244.2	200	161	74
30	37	6467.6	22811.0	3.1	12040.0	26602.0	6710.5	200	175	80
30	43	6993.6	26493.0	470.5	17624.0	34085.0	974.7	200	103	86
30	49	8224.3	18805.0	340.3	19024.0	24446.0	2185.3	200	207	94
30	55	7495.3	34381.0	341.3	16836.0	37931.0	3470.8	200	255	106
40	1	277.1	2402.6	0.1	387.5	738.3	3.2	168	51	41
40	9	1293.3	6481.3	1.2	5543.9	2943.7	4.4	200	85	50
40	17	1666.0	15266.0	2.6	7613.8	14895.0	1477.3	200	80	61
40	25	3204.9	16218.0	5.0	12627.0	19223.0	168.0	200	179	69
40	33	4673.8	11999.0	3.5	20950.0	15998.0	179.2	200	133	81
40	41	4087.1	14305.0	424.5	18466.0	17057.0	379.4	200	114	92
40	49	4973.5	12820.0	1823.4	25623.0	18593.0	2482.8	200	233	125
40	57	5693.4	15843.0	436.1	16966.0	21628.0	2773.2	200	253	110
40	65	5988.7	20522.0	6.4	20543.0	25221.0	150.0	200	147	124
40	73	12035.0	30038.0	597.3	27014.0	33340.0	1243.3	200	293	132

Table A1: Continued

In the above table, the first column titled G stands for the number of global gaussians and the second column named L shows the number of local gaussians. For comparison purposes we should mention that MSE target for stopping iterations was $1000 * 10^{-6}$.

Biography of the Author

Giorgos Mountrakis is a Ph.D. candidate in the Department of Spatial Information Science and Engineering and a research assistant for the National Center for Geographic Information and Analysis at the University of Maine. He holds a Dipl.Eng. degree from the National Technical Univ. of Athens, Greece in 1998 and a M.S. degree from the University of Maine in 2000. In the past he has worked on spatiotemporal GIS modeling and image processing applications. His current focus includes applications of machine learning and artificial intelligence in the GIS field. Mr. Mountrakis has published papers in journals, books and refereed conferences. More information about him is available at www.aboutgis.com. Giorgos is a candidate for the Doctor of Philosophy degree in Spatial Information Science and Engineering from The University of Maine in December, 2004.