Learning Algorithms for Radial Basis Function Networks: Synthesis, Experiments and Cognitive Modelling

by

Enrico Blanzieri

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

Center of Cognitive Science
University and Polytechnic of Turin

1998

'Forty–two!' yelled Loonquawl. 'Is that all you've got to show for seven and a half million years' work?'

'I checked it very thoroughly,' said the computer, 'and that quite definitely is the answer. I think the problem, to be quite honest with you, is that you've never actually known what the question is.'

'But it was the Great Question! The Ultimate Question of Life, the Universe and Everything,' howled Loonquawl.

'Yes,' said Deep Thought with the air of one who suffers fools gladly, 'but what actually is *it?'*

A slow stupefied silence crept over the men as they stared at the computer and then at each other.

'Well, you know, it's just Eveything ... Everything ...' offered Phouchg weakly.

'Exactly!' said Deep Thought. 'So once you do know what the question actually is, you'll know what the answer means.'

Douglas Adams

The Hitch Hikers's Guide to the Galaxy

Center of Cognitive Science
University and Polytechnic of Turin

Abstract

Learning Algorithms for Radial Basis Function Networks: Synthesis, Experiments and Cognitive Modelling

by Enrico Blanzieri

Supervisor: *prof. Attilio Giordana*
Department of Computer Science
University of Turin

The dissertation presents structure and properties of the Radial Basis Function Networks (RBFN). The work contains a survey of the existing learning algorithms for RBFN and of the different interpretations that RBFN can have (Fuzzy Controller, Propositional Theory, Probabilistic Neural Networks, Regularization Networks). The central topic is the presentation of a formal framework for facing the problem of structural modifications. Two original algorithms that solve the problem are presented. In order to test the effectiveness of the methods we present and discuss some applications of the RBFNs. More specifically the possibility of using RBFN as cognitive models is explored. Finally, a case study of modeling of human communication phenomena is presented and discussed.

ACKNOWLEDGMENTS

The author wishes to thank Bruno Bara, Cristina Baroglio, Marco Botta, Monica Bucciarelli, Attilio Giordana, Anna Goy, Patrick Katenkamp, Filippo Neri, Lorenza Saitta, Maurizio Tirassa, Daniele Theseider Duprè and Pierpaolo Peretti.

TABLE OF CONTENTS

INTRODUCTION

The main topic of this thesis is a class of function approximators called Radial Basis Function Networks (RBFNs) that can also be seen as a particular class of Artificial Neural Networks (ANNs). The thesis presents two different kinds of results. On one hand we present the general architecture and the learning algorithms for RBFNs. In particular original technical improvements of the dynamic learning algorithms are presented. On the other hand this work discusses the general possibility of using RBFNs as cognitive models, and it presents a model of human communicative phenomena. This introduction outlines the original contributions of the thesis, placing both aspects of the work in the general framework of neural networks research.

The thesis organizes already–published results achieved during the Ph.D. program and also presents new results still unpublished. The first part aims at providing a technical contribution by presenting some original dynamic learning algorithms (i.e. capable of dealing with structural changes), and by proposing a powerful formal framework for modelling the algorithms themselves. A further technical goal is to show how different interpretations of RBFNs can be useful for designing hybrid systems and make it possible for results achieved with different approaches, to be synthesized in a unifying framework. The dissertation begins considering the existing results related to RBFNs, emphasizing the convergence of different approaches on the same network architecture. The attention is foccused on the symbolic interpretation of RBFNs [Blanzieri and Giordana, 1995] and on the description of two dynamic algorithms Dynamic Competitive Learning and Dynamic Regression Tree [Blanzieri and Katenkamp, 1996, Blanzieri and A.Giordana, 1996]. Finally, a formal

framework for dynamic algorithms analysis is presented.

In the second part of the thesis the possibility and the consequences of using RBFNs as cognitive models are discussed. The problem is posed in the general framework of cognitive modelling with neural networks by emphasizing the RBFNs' properties that prove to be the most useful. Specifically, the symbolic interpretation of RBFNs can be exploited in order to bridge the gap between symbolic and subsymbolic representations that characterized the introduction of connectionist models at the beginning. The latter feature is applied to an analysis of the mental processes involved in the comprehension of the effect achieved on the partner during human communication [Blanzieri and Bucciarelli, 1996a, Blanzieri and Bucciarelli, 1996b, Blanzieri et al., 1996].

Now let us briefly sketch the areas involved in the thesis: Artificial Neural Networks (ANNs), Machine Learning and Cognitive Modelling. ANNs are basically mathematical objects used for function approximation or dynamical system identification, which have been intensively used as models of neurophysiology and cognition since their appearance. For instance the first published mathematical model was intended to be a formalization of neural activity in the brain [McCulloch and Pitts, 1943], whereas the perceptron was inspired by the features of the visual system [Rosenblatt, 1962]. However, important technical and mathematical results on ANNs have been achieved as intermediate steps towards the goal of modelling cognitive phenomena. The development of the Backpropagation learning algorithm presented by the Parallel Distributed Processing group [Rumelhart et al., 1986, McClelland et al., 1986] ,for instance, gave rise to ten years of connectionist renaissance. The subtitle of both most representative books was "Explorations in the Microstructure of Cognition", and it shows how strongly the attention of the authors foccused on the cognitive implications of their models. Therefore, technical and cognitive aspects of ANNs are closely linked together as well as in the present work.

ANNs solve a sub–class of what can be called Inductive Learning problems. Learning is the complex activity performed by systems that adapt depending on their own history and experience and it is a natural phenomenon. Learning systems are the animals and, on a philogenetical scale, all living beings. A branch of Artificial Intelligence, called Machine Learning, has the goal of building artificial systems that can adapt to solve different tasks or at least specific instances of the same task.

The basic idea that underlies learning is induction: the ability to infer general rules from particular observations. From an abstract point of view Inductive Learning means constructing a general model of a phenomenon given a set of specific instances of it. Depending on the case a model can correspond to a concept, a mathematical function, an automata and so on.

A learning system adapts using the available data with the general goal of improving its performance in a related task. In order to achieve this goal, the system builds a knowledge structure that permits generalization of new facts.

Different kinds of learning are distinguished in the Machine Learning literature. A first distinction is between supervised and unsupervised learning. In the case of supervised learning, information on the target function is available. In the simplest case, a set of instances of the function (a mapping between a domain and codomain) is known and the task consists in reconstructing the function. If the function is boolean, it corresponds to a concept and the task corresponds to a classification problem whereas, if the function is continuous, the task is posed as a regression problem. In both cases, the accuracy of the response of the system to never–seen–before data, is commonly taken as a measure of learning process effectiveness. With unsupervised learning no target function is available, so the problem is to organize the data (conceptual learning, concept formation), or, to discover the functional relationship among the data. An example of partially supervised learning is reinforcement learning: no correct answer is known but the system receives, from the environment or from the teacher, either reward or punishment signals, depending on its own be-

haviour. The reward and punishment contains information on the target function. Furthermore, both supervised and unsupervised learning, can be classified into off–line and on–line learning. In the case of off–line learning, the system is completely focussed on the learning activity and the set of samples is given completely at the beginning. In case of on–line there is no a strict separation between the learning phase and the execution of the particular task. As a consequence, the system can learn while executing the task and so the data are only available step–by–step, during the training. As noted by [Mitchell, 1997] there is no a general technique, among the ones developed by the Machine Learning community, that outperforms all the others for every task and every domain. The choice of the most effective one, on a particular learning task, has to take into account the nature of the data, of the problem and of the domain. ANNs' architectures and the associated learning algorithms, offer solutions both off–line and on–line for supervised or unsupervised learning problems (for an extensive review see [Haykin, 1994]). The solution provided by ANNs to the inductive learning problem is achieved by numerical computation. Thus ANNs just learn from a set of numerical examples. This property limits the problems that ANNs solve to the sub–class of learning problems that can be represented as numerical problems. As a consequence symbolic problems faced using ANNs, need to be encoded into a numerical representation and then decoded back to symbols.

The major primary goal of Artificial Intelligence is to design machines exhibiting an intelligent behaviour. Thus cognitive scientists started to use AI systems as models of cognition. In other cases the explicit goal of an AI machine has been to implement a cognitive model, like in the case of SOAR [Laird et al., 1987]. Cognitive modelling is one of the central research activities in the field of Cognitive Science. It is in modelling cognitive phenomenon, that the synthesis of the psychological data and the Artificial Intelligence technology becomes an effective scientific method. Cognitive modelling has a long history. In fact there is a sort of continuity between Descartes' machine metaphor and, for instance, the recent computer metaphor of

the cognitive psychology. Although Neural Networks are only one of the possible techniques that Artificial Intelligence provides, they have gained a lot of popularity as cognitive models. There are many reasons for the success of connectionist modelling. One of the most relevant, is the sub–symbolism of these models, i.e. the fact that the level of the representation is different from the one of the computation. This property leads to the formation of distributed representations which are useful to explain task interference and content-dependent cognitive phenomena. It also permits soft degradation of the performance in case of failures. A good example of how these properties can be exploited for modelling a non–trivial cognitive phenomenon can be found in [Plaut and Shallice, 1993] where a damaged network with attractor semantics is used to model the neurological syndrome of deep dyslexia. Implementing ANNs is considerably simpler than other Machine Learning systems and so ANNs have been widely used as models of the natural learning phenomena themselves, for instance as in [Rumelhart and McClelland, 1986] where an ANN has been used for modelling language acquisition. Moreover, the resemblance of ANNs with biologic neural systems leads the supporters of the connectionist paradigm to affirm that ANNs provide –or at least they are a part of– an unifying framework of all sciences of the mind. For all these reasons the neural models of cognition gained a wide popularity but have also attracted criticisms. The biological plausibility of neural models is controversial. There is, in fact, a wide gap in terms of complexity between biological and artificial neural systems. A more philosophical criticism was presented by Fodor and Phylyshyn [Fodor and Pylyshyn, 1988] who addressed the capacity of these models to account for sistematicity and compositionality. This critical position, somehow favourable to the symbolic approach, led to a violent debate [Smolensky, 1988, Fodor and McMaughlin, 1990] that showed how the cognitive science community was divided on this topic.

Radial Basis Function Networks are a particular kind of Neural Network which until now, are little exploited in Cognitive Modelling. They are characterized by

having a transfer function in the hidden units layer, having radial symmetry with respect to a centre. From this characteristic it cames the name RBFNs. Usually the function is also bell-shaped, thus the activation of the unit has a maximum in the centre and is almost equal to zero far from it. This feature entails the possibility to modify a unit of the network without affecting the overall behaviour and turns out to be very useful in order to implement incremental learning strategies. Moreover, they exhibit nice mathematical properties that exploit the regularization theory, and they are suitable to statistical and symbolic interpretations.

RBFNs can be used to model cognitive phenomena. More specifically, we will focus on communicative phenomena. The reason why a speaker communicates is to achieve an effect on a listener. However, psychological literature is not overly concerned with an analysis of the communicative effect, i.e. perlocutionary effect. Our research is an attempt to analyze the perlocutionary effect from the point of view of the speaker. In particular, we are interested in what is relevant to the speaker in order to evaluate the effect had on the listener, and how the evaluation process is carried on. This evaluation process, we argue, consists of use of evidence to strengthen or weaken the belief concerning the effect achieved on the listener.

Chapter 1

RADIAL BASIS FUNCTION NETWORKS OVERVIEW

This chapter and the following one extensively survey the scientific literature related to Radial Basis Function Networks (in the following simply referred to as RBFNs). These chapters are complementary. The present one contains a basic overview while the Chapter 2 addresses the connections between RBFNs and other methods.

In the following, we describe the basic RBFN architecture, their approximation properties, i.e. the characterization of the problems that the RBFNs can solve, and the basic learning algorithms. Finally, the last section is devoted to a brief introduction to the recurrent version of the RBFNs.

1.1 Architecture and Approximation Properties

The RBFNs correspond to a particular class of function approximators which can be trained, using a set of samples. RBFNs have been receiving a growing amount of attention since their initial proposal [Broomhead and Lowe, 1988, Moody and Darken, 1988], and now a great deal of theoretical and empirical results are available.

1.1.1 Radial Basis Function Networks Architecture

The approximation strategy used in RBFNs consists of approximating an unknown function with a linear combination of non–linear functions, called basis functions. The basis functions are radial functions, *i.e.* they have radial symmetry with respect

to a centre. Let X be a vectorial space, representing the domain of the function $f(\bar{x})$ to approximate, and $\bar{x}$ a point in X. The general form for an RBFN $\mathcal{N}$ is given by the following expression:

$$\mathcal{N}(\bar{x}) = \sum_{i=1}^{n} w_i e(\| \bar{x} - \bar{c}_i \|_i) \tag{1.1}$$

where $e(z)$ is a non-linear radial function with centre in $\bar{c}_i$ and $\| \bar{x} - \bar{c}_i \|_i$ denotes the distance of $\bar{x}$ from the centre and w_i are real numbers (weights). Each basis function is radial because its dependence from $\bar{x}$ is only through the term $\| \bar{x} - \bar{c}_i \|_i$.

Many alternative choices are possible for the function $e(z)$: triangular, car-box, gaussian. Anyhow it is usual to choose $e(z)$ in such a way that the following conditions hold:

$$e(-z) = e(z)$$

$$\lim_{z \to \pm\infty} e(z) = 0$$

A common choice for the distance function $\| \, . \, \|_i$ is a biquadratic form:

$$\| \bar{x} \|_i = \bar{x} Q_i \bar{x}^T$$

where Q_i is a positive definite matrix, often constrained to be diagonal:

$$Q_i = \begin{bmatrix} q_{i,11} & 0 & \dots & 0 \\ 0 & q_{i,22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & q_{i,nn} \end{bmatrix}$$

In the simplest case all diagonal elements of Q_I are equal $q_{i,jj} = q_i$ so that $Q_i = q_i I$. In this case the radiality of the basis functions is proper and if function $e(z)$ fades to infinity, $\frac{1}{q_i}$ can be interpreted as the width of the i-th basis function.

From the point of view of the notation is also common to write:

$$e(\| \bar{x} - \bar{c}_i \|_i) = e_i(\| \bar{x} - \bar{c}_i \|)$$

where the information about the distance function $\| \, . \, \|_i$ is contained into the function $e_i(\bar{x})$.

It is also possible to define a normalized version of the RBFN:

$$\mathcal{N}(\bar{x}) = \frac{\sum_{i=1}^{n} w_i e(\| \bar{x} - \bar{c}_i \|_i)}{\sum_{i=1}^{n} e(\| \bar{x} - \bar{c}_i \|_i)}$$

Different type of output, continuous or boolean, may be needed depending on the type of the target function. In order to obtain a boolean output $\mathcal{N}_B$ we need to compose function N and a derivable threshold function σ:

$$\mathcal{N}_B(\bar{x}) = \sigma(\mathcal{N}(\bar{x}))$$

usually $\sigma(x)$ is the sigmoid (logistic function):

$$\sigma(x) = \frac{1}{1 + e^{-kx}}$$

whose derivative is:

$$\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$$

The positive constant k expresses the steepness of the threshold.

1.1.2 Universal Approximators

A relevant property usually required for a class of approximators is the the universal approximation. Given a family of function approximators, it is important to characterize the class of functions which can be effectively approximated. In general, an approximator is said to be universal if it can asymptotically approximate any integrable function to a desired degree of precision.

Hornik *et al.* [Hornik et al., 1989] proved that an any network with at least three layers (input, hidden and output layers) is an universal approximator provided that the activation function of the hidden layer is nonlinear. In the Multi-Layer Perceptron (MLP), traditionally trained by means of the backpropagation algorithm, the most frequently used activation function is the sigmoid. RBFNs are similar to MLPs from the point of view of the topological structure but they adopt activation functions having axial symmetry.

Universal approximation capability for RBFNs was presented in [Park and Sandberg, 1991, Park and Sandberg, 1993], where the problem of characterizing the kinds of radial function that entail the property of universal approximation was addressed by Chen and Chen [Chen and Chen, 1995] who found that a necessary and sufficient condition is that function $e(z)$ does not to be an even polynomial.

From the mathematical point of view the universal approximation property is usually asserted by demonstrating the density of the family of approximators into the set of the target functions. This guarantees the existence of an approximator that with a high, but finite number of units, can achieve an approximation with every degree of precision. The result states only that this approximator exists. It does not, however, suggest any direct method for constructing it. In general this assertion is true, even when the function is explicitly given. In other words, it is not always possible to find the best approximation within a specified class of approximators, even when the analytical expression of the function is given.

1.1.3 The Function Approximation Problem

Whether the target function is boolean or continuous, the learning task of a feed–forward RBFN can be stated as a classification or regression problem. In both cases the problem can be stated in the general framework of the function approximation problem, formally expressed as: given an unknown target function $f : \mathcal{R}^n \rightarrow \mathcal{D}$ and a

set S of samples (x_i, y_i) such that $f(x_i) = y_i$ for $i = 1 \ldots N$. find an approximator $\hat{f}$ of f that minimizes a cost function $E(f, \hat{f})$. Function f is a mapping from a continuous multidimensional domain X to a codomain $\mathcal{D} \subset \mathcal{R}$ (regression) or $\mathcal{D} = \mathcal{B} = \{0, 1\}$ (classification). The approximation accuracy is measured by the cost function $E(f, \hat{f})$ also said error function (or approximation criterion) and which depends on the set of examples S. In general the solution depends upon S, upon the choice of the approximation criterion and upon the class of functions in which we approximator $\hat{f}$ is searched. Many choices for the approximation criterion are possible. Statistics provides several alternative definitions of the error measures (see Appendix C).

However the error measures reported in the Appendix are useful only for theoretical reasons because the target function is usually given only in form of a set S. In practice, the common choice for the cost function is the empirical square error:

$$SE_{emp} = \sum_{i=0}^{N} \left(y_i - \hat{f}(x_i)\right)^2 \tag{1.2}$$

Under some restrictive hypothesis it can be shown that minimizing (1.2) is equivalent to finding the approximator that maximizes the likelihood of S, i.e. the probability of observing S given the a priori hypothesis $f = \hat{f}$ ($P(S/f = \hat{f})$ [Mitchell, 1997].

Given a family of approximators the optimal one minimizes the error in (1.2). Finding the optimal approximator is thus equivalent to solving a least squared error problem.

It is worth noting, that the problem definition and the considerations about the errors, can be extended to the case in which a subset of the dimension of the domain is boolean and so the domain is the Cartesian product of a n-dimensional continuous space to a m-dimensional boolean space $\mathcal{R}^n \times \mathcal{B}^m$. The boolean inputs can be viewed as continuous inputs that receive only the boolean values 0 and 1.

1.2 Learning Algorithms for RBFNs

The universal approximation property states that an optimal solution to the approximation problem exists: finding the corresponding minimum of the cost function is the goal of the learning algorithms. This section introduces some of the basic facts about learning algorithms for RBFNs. More advanced learning methods are presented in Chapters 2 and 3.

In the following we will assume that the choice of the radial basis function $e(z)$ has already been made. In order to find the minimum of the cost function a learning algorithms must accomplish the following steps:

1. select a search space (i.e. a subset of the parameter space);

2. select a starting point in the space (initialization);

3. search for the minimum (refining).

An RBFN is completely specified by choosing the following parameters:

- The number n of radial basis functions;

- The centres c_i and the distances $\| \, . \, \|_i$, i.e. the matrixes Q_i $(i = 1 \ldots n)$;

- The weights w_i.

The number n of radial functions is a critical choice and depending on the approach can be made a priori or determined incrementally. In fact, both the dimensions of the parameter space and, consequently, the size of the family of approximators depend on the value of n. We will call an algorithm that starts with a fixed number n of radial functions determined a priori 'static' , and an algorithm that during the computation is able to add or delete one or more basis functions 'dynamic'.

A static learning algorithm is also parametric because the search for the optimal approximator corresponds to a search in the parameter space defined by the fixed number of radial basis functions. On the contrary, a dynamic learning algorithms changes the parameter space in which it operates, while adding or deleting radial basis functions. The learning algorithms are also very different depending on whether the sample set S is completely available before the learning process or if it is given, sample by sample, during it. In the former case, off–line learning is possible while in the latter, an on-line learning approach is needed as was shown in the Introduction. While some of the static algorithms can be adapted for both learning type, the application of the dynamic one makes sense only in the case of on-line learning.

The static methods for learning RBFNs from examples are based on a two–step learning procedure: first the centres and the widths of the basis functions are determined and then in a second step the weights are determined. Each one of the steps can be done by means of several different strategies. A usual training procedure uses a statistical clustering algorithm, such as *k-Means* for determining the centres and the widths associated to the basis functions and then estimates the weights by computing the coefficients of the pseudo-inverse matrix or, alternatively, by performing the error gradient descent.

Given n, the corresponding parameter space is defined by the parameters that characterize each one of the radial basis functions, *i.e.* the centres c_i and the matrixes Q_i, and the weights w_i $(i = 1 \ldots n)$.

The search space can be restricted, by limiting the possible choices of the parameters, or imposing constraints on the values. Several basic techniques are available for initializing and refining an RBFN. Some of them apply to all kinds of parameter in the network, some not.

Gradient Descent

Using continuous radial functions derivable in the all parameter space, it is immediate to apply the classical error gradient descent technique, in order to finely tune not only the weights w_i but also the centres c_i and the elements of the matrix Q_i in the first hidden layer units. More specifically, let SE_{emp} be the quadratic error evaluated on the learning set and let, moreover, λ_k indicate a generic parameter in the network, such as, a weight w_i on a link, or an element of the matrix of the width Q_i or a component of the centre $\bar{c}_i$ of a radial function, all the necessary derivatives can be easely computed, and the learning rule takes the form:

$$\Delta\lambda_k = -\eta \frac{\partial SE_{emp}}{\partial \lambda_k} \tag{1.3}$$

where η is the learning rate.

The method based on the pseudo-inverse matrix is usually faster than the gradient descent. However, this last one is sometimes preferable, because it is simpler to implement and suitable to on-line learning. Optimized versions of the gradient descent technique such as conjugate gradient, momentum or others are possible. The main problems with gradient descent are that the convergence is slow and depends on the choice of the initial point. Although [Bianchini et al., 1995] demonstrated that, in the case of classification, a wide class of RBFNs has a unique minimum, (*i.e.* no local minima exists in the cost function) it is not possible to reach this optimal point, in a short time, from every starting point of the parameter space. Hence the initialization of the network is critical.

Instance Based Initialization

In the first formulation of RBFNs [Moody and Darken, 1988] all instances of the sample set S were used as centres of the basis functions and the width of the basis function were the same for all the functions of the network.

Centre Clustering

The technique of assigning a radial function to every sample is very simple but produces excessively large networks which are inefficient and sensitive to overfitting and exhibit poor performances. A partial solution to these problems is to cluster similar samples together, adopting a well–known technique (clustering), widely used in Pattern Recognition. To every cluster corresponds a centroid, i.e. a real or artificial sample that appears to be prototypical for the cluster. The centroid is then chosen as the centre for a radial basis function. The resulting network is remarkably smaller than when Instance Based Learning is used. Moreover in this case the centres can also be tuned via gradient descent. This basic technique also permits the radial condition to be relaxed, adopting different widths along different dimensions of the domain X. The parameter space contains all the centre values and the width values so it is larger than in the other case. The price to pay for better performance is an increase in the training time.

Symbolic Initialization

An alternative method for constructing the layout, *i.e.* choosing the centres and the correspondent widths of an RBFN is to use a symbolic learning algorithm [Baroglio et al., 1996, Tresp et al., 1993]. This becomes particularly simple in the case of Factorizable RBFNs (F-RBFNs). In this case the factorization permits seeing each factor of the radial function as a fuzzy membership of widths A_{ij} determined by the width of the basis function and the product as a logical *AND*, so that F-RBFN can be approximated by a set of propositional clauses of the type:

$$R_j = member(x_1, A_{1j}) \wedge member(x_2, A_{2j}) \wedge \ldots \\ \ldots \wedge member(x_n, A_{nj}) \rightarrow w_j \tag{1.4}$$

Rules of this type can be easely learned using an algorithm for inducing decision trees, such as ID3 [Quinlan, 1979, Sammut et al., 1992] or, better an algorithm for

inducing regression trees, such as CART [Breiman et al., 1984].

Weights Linear Optimization

Both equations (1.1) and (1.1.1) are linear in the weights w_i then, given the parameters c_i and Q_i of the basis functions it is possible to use a linear optimization method for finding the values of the w_i, that minimize the cost function computed on the sample set. This is the learning strategy adopted in the regularization theory approach [Poggio and Girosi, 1990]. This method relies on the computation of the pseudo-inverse matrix. This point will be addressed into details in Section 2.1 where the links between RBFNs and regularization theory are discussed. Further optimizations of the method have been presented [Chen et al., 1991] [Orr, 1995].

Evolutionary Computation

Evolutionary computation can be applied to the learning algorithms of the RBFNs [Whitehead and Choate, 1994]. The presentation of the details of this kind of system is far beyond the goals of this thesis, so we will briefly report, only the general principles of these methods. Evolutionary computation is a search strategy based on the maintenance of a succession of populations of solutions to a given problem. Every population, called a generation, is obtained from the previous one via the the selection of the best solution and their mutation or cross–over. The succession converges to a population of locally optimal solutions that depends on the operator used for the selection (fitness function) and the operators of mutation and cross–over. The definition of other operators is possible and leads to variants of the basic search strategy. In general evolutionary computation is a powerful search strategy that is particularly well suited to application in combinatory domains, where the cross–over of locally good solutions can lead to better global solutions. That seems to be the case for RBFNs. In fact the RBFNs architecture is based on local strategy of

approximation: the different functions interact poorly each other and so that allows their combinations to be significantly better than the original networks. As long as the intermediate solutions have a variable number of basis functions the methods could be classified as dynamic, static otherwise.

1.3 Recurrent Radial Basis Function Networks

Most attention in the ANNs literature is foccused on the feed–forward networks. Nevertheless there is a growing interest in networks provided with feedbacks called Recurrent Networks [Elman, 1990]. A recurrent network is characterized by having some output units connected with some units of the other layers. This apparently simple modification causes heavy changes in behaviour and computational properties of a network. Owing to the presence of feedback archs, a network becomes an approximator of dynamical systems. The reports related to recurrent RBFNs is limited to the work of [Frasconi et al., 1996]. Which introduces a second order RBFNs where the feedback connections are obtained via a product. It is shown how these networks can be forced to work as finite state automata. The authors report examples where a recurrent RBFN learns a Tomita grammar and provide an algorithm for extracting symbolic description of the corresponding automaton. Further investigation is still required to test possible relations with other formalisms like Feature Grammars and Markov Chains that could emerge from a recurrent generalization of the symbolic and statistical interpretations (see Chapter 2).

Chapter 2

DIFFERENT APPROACHES TO THE RADIAL BASIS FUNCTION NETWORKS

The architecture underlying Radial Basis Function Networks has been defined independently from their mathematical representation and presented several times under different names. Initially presented in the neural network framework [Moody and Darken, 1988] RBFNs were reintroduced as a particular case of regularization networks [Poggio and Girosi, 1990]. Independently the fuzzy logic community developed the fuzzy controllers [Berenji, 1992] whose effectiveness relies on the same approximation principles. Very related to the fuzzy approach some works [Blanzieri and Giordana, 1995, Tresp et al., 1993, Tresp et al., 1997] proposed to use the RBFN for mapping and refining propositional knowledge. With a very different approach in the mainstream of applied statistics [Scott, 1992], the problem of regression and classification, and more generally density estimation, were faced by means of kernel estimators that were strongly related to RBFN. Finally, RBFN can also be placed in the framework of instance based learning. As a consequence Radial Basis Function Networks can be viewed from several different points of view.

2.1 Regularization Networks Approach

In a paper that appears to be fundamental for the Radial Basis Function Networks theory Poggio and Girosi [Poggio and Girosi, 1990] provided an elegant connection with Kolmogorov regularization theory. The basic idea of regularization consists of reducing an approximation problem to the minimization of a functional. The

functional contains prior information about the nature of the unknown function, like constraints on its smoothness. The structure of the approximator is not initially given so in the regularization framework the function approximation problem is stated as:

Find the function $F(x)$ that minimize:

$$E(F) = \frac{1}{2}\sum_{i=1}^{n}(d_i - F(x_i))^2 + \lambda \| PF \|^2 = E_s(F) + \lambda E_c(F) \tag{2.1}$$

Where $E_S(F)$ is the standard error term, $E_R(F)$ is the regularization term, λ is a regularization parameter and P is a differential operator.

By differentiating equation (2.1) we obtain

$$P^*PF(x) = \frac{1}{\lambda}\sum_{i=1}^{n}(d_i - F(x_i))^2\delta(x - x_i) \tag{2.2}$$

where $\delta(.)$ is the Dirac's function. The solution F of the (2.2) is finally:

$$F(x) = \frac{1}{\lambda}\sum_{i=1}^{n}(d_i - F(x_i))^2 G(x, x_i) \tag{2.3}$$

The regularization theory leads to an approximator that is an expansion on a set of Green's function $G(x, x_i)$ of the operator P^*P. By definition the Green's function of the operator A centred in x_i is

$$AG(x, x_i) = \delta(x - x_i)$$

The shape of these functions depends only on the differential operator P, i.e. on the former assumptions about the characteristics of the mapping between input and output space. Thus the choice of P completely determines the basis functions of the approximator. In particular if P is invariant for rotation and translation the Green's function are:

$$G(x, x_i) = G(\| x - x_i \|)$$

so they depend only on the distance $\| x - x_i \|$ and so they are Radial Functions.

The points x_i are the centres of the expansion and the terms $\frac{1}{\lambda}(d_i - F(x_i)$ of the equation (2.3) are the coefficients.

The approximator is

$$w_i = \frac{1}{\lambda}(d_i - F(x)F(x) = \sum_{i=1}^{n} w_i G(x, x_i) \tag{2.4}$$

the equation (2.4) evaluated in the point x_j leads to

$$F(x_j) = \sum_{i=1}^{n} w_i G(x_j, x_i) \tag{2.5}$$

In order to determine the w_i let us define the matrixes:

$$F = \begin{bmatrix} F(x_1) \\ F(x_2) \\ \vdots \\ F(x_N) \end{bmatrix}$$

$$d = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix}$$

$$W = \begin{bmatrix} w_i \\ w_2 \\ \vdots \\ w_N \end{bmatrix}$$

$$G = \begin{bmatrix} G(x_1, x_1) & G(x_1, x_2) & \vdots & G(x_1, x_N) \\ G(x_2, x_1) & G(x_2, x_2) & \vdots & G(x_2, x_N) \\ \vdots & \vdots & \vdots & \vdots \\ G(x_N, x_1) & G(x_N, x_2) & \vdots & G(x_N, x_N) \end{bmatrix}$$

Then the equations (2.4) can be represented in the form of matrixes:

$$W = \frac{1}{\lambda}(d - F)$$

$$F = GW$$

Eliminating F from both expressions, we obtain:

$$(G + \lambda I)W = d$$

The matrix G is symmetric and for some operator is positive definite. It is always possible to choose a proper value of λ such that $G + \lambda I$ is invertible, that leads to:

$$W = (G + \lambda I)^{-1} d$$

It is not necessary to expand the approximator over the the whole data set, in fact the point x_j on which the equation (2.4) was evaluated is arbitrarily chosen. If we consider a more general case in which the centres of the basis functions c_i with $i = 1 \ldots n$ are distinct from the data the matrix G is rectangular. Defining two new matrixes as:

$$G_0 = (G(c_i, c_j))i, j = 1 \ldots n$$

$$G = (G(c_i, c_j))i = 1 \ldots N j = 1 \ldots n$$

the optimal weights are:

$$w = \left(G^T G + \lambda G_0\right)^{-1} G^T d$$

if $\lambda = 0$

$$w = \left(G^T G\right)^{-1} G^T d = G^{+} d$$

where $G^{+} = \left(G^T G\right)^{-1} G^T d$ is the pseudoinverse matrix.

In the regularization framework the choice of the differential operator P determines the shape of the basis function. In [Haykin, 1994] a formal description of the operator that leads to the Gaussian RBFN is reported. The operator expresses conditions on the absolute value of the derivatives of the approximator. Hence the minimization of the regularization term $E_R(F)$ causes a smoothing of the function encoded by the approximator.

In an analogous way Girosi *et al.* [Girosi et al., 1995] presented an extension of the regularization networks. The regularization functional is mathematically expressed as a condition on the Fourier transform of the approximator. In their work they set the constraint that the bandwidth be small. Such an approximator, they argue, oscillates less so it presents a smoother behaviour. They obtain the class of the generalized regularization networks strongly connected to what they called Hyper Basis Functions (HBF) that approximates the function with:

$$f(x) = \sum_{i=1}^{n} c_i(\| x - x_i \|_W) \tag{2.6}$$

where the weighted norm is defined as: $\| x \|_W = xW^TWx$, with W vector of weights.

The RBFN described by equation (2.6) is not radial. The surface with the same value of the function are not spheric any more but hyper–ellipsoidal. That is the case of the network described by the equation (1.1).

Finally, Orr [Orr, 1995] exploited the regularization framework for determining a criterion for the selecting the position of the centres. As a conclusion, regularization theory provides a strong mathematical framework which allows an optimal estimate of the weights and, at the same time, allows the smoothing of the function encoded in the RBFN, to be controlled via the regularization term.

Concluding Remarks

By applying the regularization theory to RBFNs we obtain an off-line learning method: The centres of the radial basis function are initialized with the samples or with a clustering technique, the widths are usually a fixed parameter and the weights are computed via pseudoinversion. The number of basis functions is fixed, so the method is static. The main feature of the framework is to provide guidelines for an optimal choice of the type of basis functions, depending on the regularization term used for expressing the differential smoothing properties of the approximator.

2.2 Fuzzy Controller Approach

The RBFNs can also be interpreted as fuzzy controllers. In general, a controller of this kind is a software or hardware implementation of a control function, defined from the state–space of the system to its input–space. In this way, the control function maps a set of information about the state of the system we want to control, to the actions the controller has to apply to the system. Typically, the state and the actions are continuous vectors and the controller is fully described by a set of input variables X , a set of output variables Y, and the set of elements implementing the control function. In the case of fuzzy controllers, the internal elements are defined by means of a fuzzy logic propositional theory.

Fuzzy Logics

Fuzzy logics are based on a generalization of the characteristic function of a set that is the boolean function, associated to the presence of an element in the set itself. Formally, let f_A the characteristic function of the a set A:

$$f_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

The fuzzy set theory [Zadeh, 1965] generalizes the notion of presence of an element in a set and consequently the notion of characteristic function, by introducing fuzzy values. This approach is equivalent to introduce uncertainty in the presence of an element in a set. In this context the fuzzy characteristic function, that is called membership, can assume any value in the interval $[0, 1]$. A set in which the membership function is restricted to assume the values in the set $\{0, 1\}$, is said to be crisp. The introduction of a fuzzy membership has deep implications concerning the logics which can be built on it. The first one is the possibility of having fuzzy truth values for predicates. A predicate is no longer simply false (0) or true (1) but can assume any value between. Consequently, the definitions of the basic connectives (disjunction, conjunction and negation) have to deal with fuzzy values. Fuzzy logics is typically used for expressing uncertain or approximated knowledge in the form of rules. The theory can be partially contradictory, causing fuzzy memberships to overlap each other. Many different shapes for the membership functions have been proposed (triangular, trapezoidal, gaussian) (see [Berenji, 1992]).

Fuzzy Controllers and RBFNs

Usually a fuzzy controller is organized on three layers. The first one implements the so–called fuzzyfication operation and maps every dimension of the input space via the memberships, to one or more linguistic variables, in a fuzzy logic language. The linguistic variables are then combined with the fuzzy connectives to form the fuzzy theory. Typically the theory is propositional and it can be flat or not, e.g. expressed as a sum of minterms. Finally, the last layer implements the defuzzyfication trasforming back the continuous truth values into points in the output space.

Therefore, Factorized Radial Basis Function Networks (F–RBFNs), that were initially introduced in [Poggio and Girosi, 1990] can be interpreted as fuzzy controllers. The architecture is also similar to the fuzzy/neural networks introduced by Berenji [Berenji, 1992] for implementing fuzzy controllers capable of learning from a rein-

forcement signal and to the architecture proposed by Tresp *et al.* [Tresp et al., 1993]. Figure 2.1 describes the basic network topology.

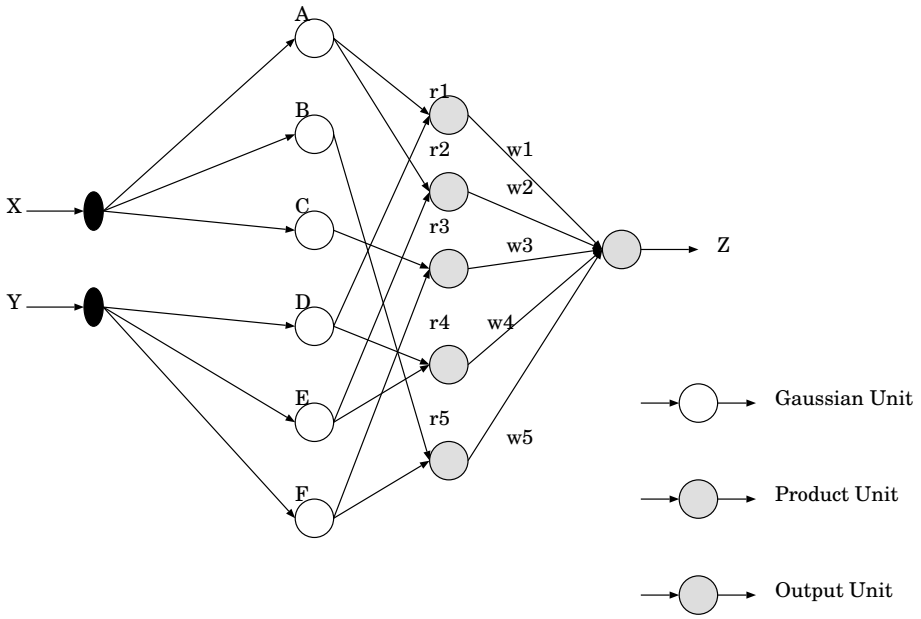


Figure 2.1: Reference F–RBFN architecture. The first layer hidden units have a one–dimensional Gaussian activation function. The second layer hidden units compose the input values using arithmetic product. An average sum unit performs the weighted sum of the activation values received from the product units.

The activation function used in an F-RBFN with n input units is defined as the product of n one–dimensional radial functions, each one associated to one of the input features. Therefore an F-RBFN can be described as a network with two hidden layers. The neurons in the first hidden layer are feature detectors, each associated to a single one–dimensional activation function and are connected to a single input only. For example if we choose to use Gaussian functions, the neuron r_{ij} (the i-th component

of the j-th activation area) computes the output:

$$\mu_{ij} = e^{-\left(\frac{I_i - C_{ij}}{\sigma_{ij}}\right)^2} \tag{2.7}$$

The neurons in the second hidden layer simply compute a product and construct multi-dimensional radial functions:

$$r_j = \prod_i \mu_{ij} = e_j \tag{2.8}$$

where e_j was introduced in the Chapter 1.

Finally, the output neuron combines the contributes of the composite functions computed in the second hidden layer. In our architecture, a choice of four different activation functions is possible for the output neuron, in order to adapt the network to different needs. The output function, normally adopted for RBFNs, is a weighted sum

$$Y = \sum_j w_j r_j \tag{2.9}$$

The same function can be followed by a sigmoid, thus realizing a perceptron, when the network is used for a classification task. Using this function the network tends to produce an output value close to '0' everywhere the input vector I falls in a point of the domain which is far from every activation area. The consequence is under-generalization in the classification tasks.

This problem can be avoided by introducing a normalization term in the output activation function:

$$\hat{Y} = \frac{\sum_j w_j r_j}{\sum_j r_j} \tag{2.10}$$

This function is frequently used for fuzzy controller architectures [Berenji, 1992]. In this case, one obtains a network biased toward over-generalization in a similar way as it happen for the multi-layer perceptron. Depending on the application, under-generalization or over-generalization can be preferable.

Concluding Remarks

Traditionally fuzzy controllers were designed by hand, expressing the domain knowledge in a set of fuzzy rules. When the membership functions are derivable, gradient descent techniques can be applied [Jang, 1993].

2.3 Symbolic Interpretation

An important property, directly related to the fuzzy controller interpretation of the F–RBFNs is the possibility of giving an immediate, symbolic interpretation of the hidden neuron semantics [Blanzieri and Giordana, 1995, Tresp et al., 1993, Tresp et al., 1997]. In fact, the closed regions corresponding to neuron activation areas can be labelled with a symbol and interpreted as elementary concepts. In this section we will define a more precise network architecture, which has straightforward interpretation in terms of propositional logics.

Associating a Symbolic Interpretation to an F-RBFN

Factorized RBFNs have an immediate symbolic interpretation. In fact, defining the activation area A_j of a neuron r_j as the region in the input domain where the output of r_j is greater than a given threshold T, we obtain an ellipse with the axis parallel to the input features. Moreover, A_j is inscribed into a rectangle R_j having the edges parallel to the input features (see Figure 2.2).

Then, every edge r_{ij} of R_j can be seen as a pair of conditions on the input I_i and then the whole rectangle can be read as a conjunctive condition on the input. A variant of this symbolic interpretation, could be to assign a symbol to every edge, interpreted as an atomic proposition. In this way, the one–dimensional activation functions can be seen as a "fuzzy" semantics of the atomic propositions.

Finally, links from the second hidden layer to the output neuron can be seen as

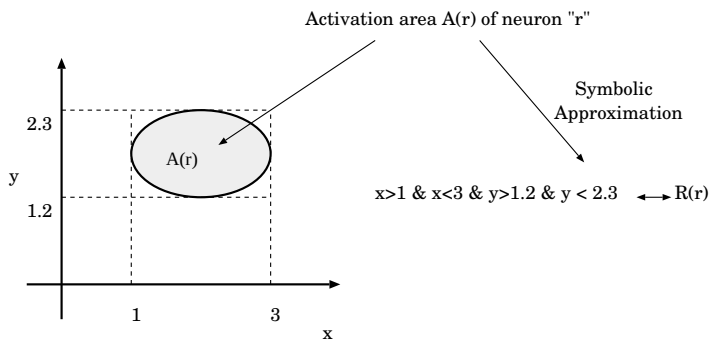


Figure 2.2: The closed region where a factorizable radial function is dominant can be roughly approximated using a conjunctive logical assertion.

implication rule of the type:

$$R_j \rightarrow w_j \tag{2.11}$$

being R_j the logical description of the rectangle R_j. In other words, the meaning of (2.11) is: "if conditions R_j hold then the value of the output is w_j". Then, the activation function associated to the output neuron implements a kind of *evidential reasoning* taking into account the different pieces of evidence coming from the rules encoded in the network.

In any case we must be conscious that this symbolic interpretation has to be considered only as a heuristic approximation of the knowledge encoded in the network and so we cannot expect a full logical equivalence. Moreover, this logical interpretation is plausible when the activation areas are not too overlapped in the input space.

Mapping a Propositional Theory into an F-RBFN

Given an n-dimensional continuous domain D, a classification theory in propositional calculus, defined on D, can always be reduced to a set of one–step implication rules of the type: $C_1 \wedge C_2 \wedge ... \wedge C_n \rightarrow H$ with $C_1, C_2, ..., C_n$ representing conditions

(thresholds) on the dimensions of D. Moreover, we observe that real continuous domains always have finite boundaries (defined by the range of the input sensors); then, the precondition of every classification rule always defines a rectangle in the space D.

Therefore, mapping such a theory into the network structure of Figure 2.1, is immediate according to the fuzzy interpretation we established in the previous section. The antecedent of a rule R_i will be represented by a proper set of neurons in the first hidden layer and by a single neuron in the second one, connected to the output neuron, representing class H. The weight on the link will be set to the numeric value (say 1) representing the concept of "true", if the rule is a positive one (implies H) or to the numeric value representing the "false" (say 0 or -1) if the rule is a negative one (i.e implies $\neg H$).

In order to preserve the theory's semantics, the activation function in the first hidden layer, should be rectangular. In fact, using Gaussian functions or other kinds of continuous functions, the logical conditions of the classification theory will become blurred and then, the performance will degrade, owing to the translation into the network. On the other hand, the use of continuous activation functions allows the nework to be refined by performing the error gradient descent, as will be described later on. Then, the initial performance decay can be recovered in a few learning steps and in general, the final performances can go far beyond the ones of the initial logical theory.

Gaussian functions having width (at half height) coincident with the length of the rectangle edges were used in the experiments.

Using activation functions having a value greater than zero on all the domain D, such as Gaussians do, the choice of function (2.9) or (2.10) for the output neuron is not so obvious and deserves some more attention. In logics, it is quite common to have the Closed World Assumption (CWA) so that, anything which is not explicitly asserted is assumed to be false. Under this assumption, a classification theory can

contain only positive rules, because the negation of a class follows from the CWA.

If function (2.9) is used, the CWA can be automatically embedded in the network semantics by using a threshold function (a sigmoid in our case) in order to split the output co–domain into two regions: one, above the threshold where the output value is high and the target *class* is asserted, and another, below the threshold, where the output value is low and the *class* is negated. As a consequence we can only model positive rules on the network.

On the contrary, using function (2.10) the output value tends to alaways be "1", if the theory contains only positive rules, because of the normalization factor. Then, the CWA doesn't hold anymore and negative rules must be explicitly inserted in the network in order to cover the whole domain D either with a positive or with a negative rule.

The considered F-RBFN architecture is able to approximate continuous functions as well as classification functions and, also in this case, it is possible to give them a *qualitative* symbolic interpretation, as is done for fuzzy controllers. In this case, both function (2.9) or (2.10) can be used for the output neuron.

When the function (2.10) is used, our network is similar to one of the networks described by the general architecture proposed by Tresp *et al.*. The main difference is that we use factorizable activation functions, whereas Tresp uses directly Basis Functions. In this way we can connect a first hidden layer neuron that has an one-dimensional activation function, with different product units. This leads to a more compact networks and, from a symbolic point of view, permits handling one–dimensional conditions that may appear in more than one rule.

Concluding Remarks

The symbolic interpretation of an RBFN allows a wide range of symbolic learning algorithms to be applied in order to initialize the basis functions. Decision trees [Breiman et al., 1984] or symbolic induction systems such as SMART+

[Botta and Giordana, 1993] can be used to construct the layout from a sample set of data. Alternatively, if domain knowledge is available, *e. g.* from an expert, it can be directly inserted into the network. Gradient descent provides a technique for refining knowledge with data. Finally, it is possible to exploit symbolic semantics for mapping knowledge back [Blanzieri and Giordana, 1995].

2.4 Neural Networks Interpretation

RBFN can be described as three layers neural networks where hidden units have a radial activation function. Although some of the results of the neural networks can be extended to RBFN, exploiting this interpretation (e.g. approximation capabilities [Hornik et al., 1989] and the existence of a unique minimum [Bianchini et al., 1995], substantial differences still remain with respect to the other feed–forward networks. In fact, RBFNs exhibit properties substantially different with respect to both learning properties and semantic interpretation. In order to understand the different behaviours of the two network types, suppose modifying a weight between two nodes in the Multi Layer Perceptron (MLP), as is done by the backpropagation updating rule, during the training phase. The effect involves an *infinite* region of the input space and can affect a large part of the co-domain of the target function. On the contrary, changing the amplitude of the region of the input space in which the activation function of a neuron in an RBFN fires, or shifting its position, will have an effect *local* to the region dominated by that neuron. More in general, this *locality property* of RBFNs allows the network layout to be incrementally constructed (see for instance [Millán, 1994]), adjusting the existing neurons, and/or adding new ones. As every change has a local effect, the knowledge encoded in the other parts of the network is not lost; so, it will not be necessary to go through a global revision process.

Considering the sigmoidal activation function $O_i = \sigma(\sum_{j=1}^{N} w_{ij} I_j)$ of the MLP's hidden units, we see that each neuron, apart from a narrow region where the sigmoid

transient occurs, splits the input domain into two semi-spaces where its output is significantly close to 1 or to 0. The whole semispace where the output is close to 1 contributes to the value of the target function. On the contrary, in an RBFN, each hidden neuron is associated to a convex closed region in the input domain (*activation area*), where its response is significantly greater than zero, and dominates over every other neuron. The greatest contribution of a neuron to the output value Y comes essentially from this region. On the contrary, RBFNs, while similar in the topological structure, make use of activation functions having axial symmetry. For instance, multidimensional Gaussian functions are used in Probabilistic Neural Networks (PNN) [Specht, 1990] and in Radial Basis Function Networks (RBFNs) [Poggio and Girosi, 1990], pyramidal or trapezoidal functions are used for fuzzy controllers, and cylindric functions in the Restricted Coulomb Energy model [D.L. Reilly and Elbaum, 1982]. As a consequence, MLP and RBFNs encode the target function in a totally different way.

Concluding Remarks

In this framework, the basic learning technique is gradient descent. Using an F-RBFN with factors that appear in more than one product, the technique can be called Back-Propagation as is usually done for MLPs. As was noted in the Chapter 1, the initialization is critical and is usually achieved using the whole data set or a clustering technique. The convergence of gradient descent algorithms is guaranteed only in the case of off-line learning. Empirically, an on-line version show to converge reasonably well.

Dynamic versions were presented by [Platt, 1991] and, combined with a on-line clustering algorithm by [Fritzke, 1994b].

2.5 Statistical Approach

The architecture of the RBFNs presents a strong similarity with the regression techniques, based on non–parametric estimation of an unknown density function [Scott, 1992] and with the Probabilistic Neural Networks [Specht, 1988, Specht, 1990].

Kernel Regression Estimators

This method is known as kernel regression. The basic idea is that an unknown random function $f(\bar{x}) = y$ can be constructed by estimating the joint probability density function $g(\bar{x}, y)$:

$$f(\bar{x}) = E(Y|\bar{X} = \bar{x}) = \int_{\mathcal{R}^n} yf(y|\bar{x})dy = \frac{\int_{\mathcal{R}^n} yf(\bar{x}, y)dy}{\int_{\mathcal{R}^n} f(\bar{x}, y)dy}$$

The technique used for estimating g, is the kernel smoothing of which the Parzen windows technique is a particular case. The general form of a kernel estimator of a density function $h(\bar{z})$ defined on a space $\mathcal{R}^d$ is:

$$\hat{h}(\bar{z}) = \frac{1}{N|H|} \sum_{i=1}^{N} K_{n+1}(H^{-1}(\bar{z} - \bar{z}_i))$$

where H is a $d \times d$ nonsingular matrix and $K_d : \mathcal{R}^d \rightarrow \mathcal{R}$ is a multivariate kernel density that satisfies the conditions:

$$\int_{\mathcal{R}^d} K_d(\bar{w})d\bar{w} = 1_d$$

$$\int_{\mathcal{R}^d} \bar{w}K_d(\bar{w})d\bar{w} = 0_d$$

$$\int_{\mathcal{R}^d} \bar{w}\bar{w}^T K_d(\bar{w})d\bar{w} = I_d$$

The constant N is the number of kernels that in the statistical literature usually coincides with the number of examples.

Let us consider $\bar{z} = (\bar{x}, y)$ and a product kernel of the form

$$K_{n+1}(\bar{z}) = K_n(\bar{x})K_1(y)$$

The estimation of g becomes:

$$\hat{g}(\bar{x}, y) = \frac{1}{N|H_x|h_y} \sum_{i=1}^{N} K_n(H_x^{-1}(\bar{x} - \bar{x}_i))K_1(h_y^{-1}(y - y_i))$$

Remembering that

$$\frac{1}{h_y} \int_{\mathcal{R}^n} K_1(h_y^{-1}(y - y_i))dy = 1$$

$$\frac{1}{h_y} \int_{\mathcal{R}^n} yK_1(h_y^{-1}(y - y_i))dy = y_i$$

and substituting the estimate of g in the denominator and numerator of the (2.5)

$$\int_{\mathcal{R}^n} f(\bar{x}, y)dy = \frac{1}{N|H_x|h_y} \sum_{i=1}^{N} K_n(H_x^{-1}(\bar{x} - \bar{x}_i)) \int_{\mathcal{R}^n} K_1(h_y^{-1}(y - y_i))dy =$$

$$\frac{1}{N|H_x|} \sum_{i=1}^{N} K_n(H_x^{-1}(\bar{x} - \bar{x}_i))$$

$$\int_{\mathcal{R}^n} yf(\bar{x}, y)dy = \frac{1}{N|H_x|} \sum_{i=1}^{N} y_i K_n(H_x^{-1}(\bar{x} - \bar{x}_i))$$

finally we obtain the approximation of the f:

$$\hat{f}(\bar{x}) = \frac{\sum_{i=1}^{N} y_i K_n(H_x^{-1}(\bar{x} - \bar{x}_i))}{\sum_{i=1}^{N} K_n(H_x^{-1}(\bar{x} - \bar{x}_i))} \tag{2.12}$$

In the univariate case the (2.12) is called Nadaraya-Watson Estimator. It is easy to see by comparing the equation (2.12) to the normalized RBFN;

$$N(x) = \frac{\sum_{i=1}^{n} w_i e_i(\| x - c_i \|)}{\sum_{i=1}^{n} e_i(\| x - c_i \|)}$$

that this kind of network has the same structure. The only difference relies on the fact that no kernel-like conditions are usually stated on the radial functions. Among

others this connection was noted by Xu *et al.* [Xu et al., 1994], who exploited it for extending some results of kernel estimators like consistency and convergence rate to RBFNs.

Probabilistic Neural Networks

Probabilistic Neural Networks (PNN) originate in a pattern recognition framework as tools for building up classifiers. In that framework the examples of a classification problem are points in a continuous space and they belong to two different classes conventionally named 0 and 1. PNN were first proposed by Specht [Specht, 1988, Specht, 1990], who proposed to approximate, separately, the density distributions $g_1(\bar{x})$ and $g_0(\bar{x})$ of the two classes and use a Bayes strategy for predicting the class.

$$\hat{f}(\bar{x}) = \begin{cases} 1 & \text{if } p_1 l_1 g_1(\bar{x}) > p_0 l_0 g_0(\bar{x}) \\ 0 & \text{if } p_1 l_1 g_1(\bar{x}) < p_0 l_0 g_0(\bar{x}) \end{cases}$$

where p_1 and p_0 are the a priori probabilities for the classes to separate and l_1 and l_0 are the losses associated with their misclassification (l_1 loss associated with the decision $\hat{f}(\bar{x}) = 0$ when $f(\bar{x}) = 1$).

Then the decision surface is described by the equation:

$$g_1(\bar{x}) = k g_0(\bar{x})$$

where,

$$k = \frac{p_0 l_0}{p_1 l_1}$$

and defining $\sigma(x)$ as a threshold function the estimate of the target function is:

$$\hat{f}(\bar{x}) = \sigma(g_1(\bar{x}) - k g_0(\bar{x}))$$

Again the density approximations are made using the kernel estimations

$$g_1(\bar{x}) = \frac{1}{N_1 |H|} \sum_{i=1}^{N_1} K_{n+1}(H^{-1}(\bar{z} - \bar{z}_i))$$

with the extension of the sum limited to the N_1 instances belonging to class 1 and analogously for the class 0.

$$\hat{f}(\bar{x}) = \sigma(\frac{1}{|H|}\sum_{i=1}^{N} C(z_i)K_{n+1}(H^{-1}(\bar{z} - \bar{z}_i))$$

where,

$$C(z_i) = \begin{cases} 1 & \text{if } f(z_i) = 1 \\ -k\frac{N_1}{N_0} & \text{if } f(z_i) = 0 \end{cases}$$

The equation (2.5) is a particular case of the RBFN described in Chapter 1 for approximating boolean functions.

Concluding Remarks

In the statistical framework it is common to use all the data as centres of the kernels. In the case of a large data set it is possible to limit the initialization to an extracted sample of data. It is worth noting, that no computation is needed to find the values of the weights. In fact, as an effect of the normalization terms contained in the kernels, the weights are equal to the output values, or set to an estimate of the a priori probability of the class. This method can be applied in an incremental way, but like any other method which uses all the data, it suffers for the overgrowing of the approximator.

2.6 Instance–Based Learning Approach

Instance–based learning or *lazy* learning can be defined as a learning method that delays some, or even all, the computation efforts until the prediction phase, limiting the learning to the simple memorization of samples (see [Mitchell, 1997]). Since little or no computation at all is done during the learning phase, the approach is suitable for on–line learning. Some of the most used algorithms in this context, are the well-known, k–Nearest Neighbour family. During the prediction phase, the computation

of a distance defined on the input space, leads to the determination of the k nearest neighbours to be used for predicting the value of the target variable. RBFNs can be viewed as a particular k-NN with $k = n$ number of the basis function and a local distance function defined by $e(\| \bar{x} - \bar{x}_i \|)$. In this framework, training the network can be interpreted as the determination of the n local metrics, associated to the centres of the basis functions. Moreover, RBFN centres can be viewed as instances of the unknown function, in fact the earlier proposal of the RBFN with no centre clustering was a completely instance–based approach. When the centres are determined by a training algorithm, they can be interpreted in association with the respective weight as prototypes of the unknown function. An application of this is the *shrinking technique* [Katenkamp, 1995] based on the principle that centres and weights of an RBFN can be seen as pairs (c_i, w_i) belonging to the space $X \times Y$, as the examples of the target function do. Hence, they can be interpreted as prototypes of the target function and be processed by the same techniques used for the data. In particular, Katenkamp clustered the (c_i, w_i) of a trained network for initializing a new smaller network. That can be useful for implementing a simple knowledge transfer between different –but similar– learning tasks. Katenkamp exploited this technique successfully on the simple cart-pole control task, transferring knowledge from tasks that differ only for the numeric values of the parameters.

Chapter 3

DYNAMIC LEARNING ALGORITHMS

In the Chapter 1 the distinction between static and dynamic learning algorithms for RBFNs was introduced. The dynamic algorithms modify the number of basis functions of the network, integrating the actions occurring in the initialization and in the refining phases in an incremental learning algorithm. The first work on this direction is the Resource Allocation Networks proposed by Platt [Platt, 1991]. Other works that introduced structural changes [Fritzke, 1994a, Fritzke, 1995] or structural selection criteria [Kadirkamanathan and Niranjan, 1993, Orr, 1995] were subsequently proposed. In this chapter, the algorithms mentioned above, will be briefly reviewed. More attention is devoted to two algorithms: Dynamic Competitive Learning (DCL) and Dynamic Regression Tree (DRT) [Blanzieri and Katenkamp, 1996]. Finally, a framework based on Hilbert functional spaces will be presented and its utility on the description of the dynamic algorithms will be emphasized.

3.1 Learning with Structural Changes

In Chapter 2 we noted that some of the presented methods were immediately suitable for working incrementally. In particular, both statistical and instance–based learning frameworks, adopt the technique of initializing the centres with the samples of the training set and performing almost no other computation during the training phase. Hence, it is straightforward to add a new function when a new sample is available. Unfortunately, as noted by Platt [Platt, 1991] referring to Parzen Windows and k–NN, the two mentioned algorithms present a drawback. The resulting RBFN grows

linearly with the number of the samples. Platt also proposed a neural architecture, called Resource Allocation Networks, which combines an on–line gradient descent with a method for adding new radial functions. A new function is added when a new sample falls far from the already existing centres, or causes an error which is greater than the average. Fritske [Fritzke, 1994a] noted that RANs suffer from noise sensitivity because they add a new function for each poorly mapped sample. He therefore proposed an interesting algorithm, called GCS (see also [Fritzke, 1995]). At a very abstract level, GCS works as follows: The weights in an RBFN are continuously trained on-line by means of the Δ-rule. When the network is trapped in a local error minimum, the algorithm locates the radial function r which exhibits the highest error rate. Then a new radial function is inserted in between r and the one, among its neighbourhoods in the space $X^{(n)}$, which exhibits the highest error rate. As the neuron insertion strategy is not the optimum with respect to the distribution of the radial functions on the input space, a competitive learning algorithm similar to the Kohonen's clustering techniques based on self–organizing maps is used in order to continuously adjust the centre position. The neuron neighbourhood is evaluated by means of a topological map that is encoded as a graph where the nodes correspond to rules and the edges define the adjacency relationship among them. The algorithm GCS continuously updates this map using a hebbian competitive rule as proposed by Martinez in [Martinetz and Schulten, 1991]. Finally, Fritske's algorithm also includes a mechanism for cancelling redundant neurons inserted owing to noise or a too hastily decision. Cyclically, the algorithm searches for a pair of rules, strongly overlapped and connected to the output neuron with a similar weight value, and collapses them into a single one. Blanzieri and Katenkamp [Blanzieri and Katenkamp, 1996] reported that, when experimented in its original formulation, Fritske's algorithm showed two major drawbacks. First, instead of reaching a stable structure, it indefinitely continues to alternate neuron insertions and neuron deletions, while the error rate remains quite high. Second, GCS turned out to be very keen on unlearning. Due to the on-line

clustering algorithm, the radial functions always tend to drift towards the input space region where the most recent inputs are located. Blanzieri and Katenkamp proposed an improved version of GCS called DCL (Dynamic Competitive Learning) which fixes the first drawback but is not yet able to avoid the unlearning problem. In order to cope with this last case, a new algorithm (Dynamic Regression Tree, DRT), which adopts a different strategy for constructing the network, has been proposed. In particular, DRT uses a more accurate strategy for inserting new radial function so that the on-line clustering algorithm is not required. This is done by explicitly considering the *symbolic interpretation* (1.5) associated to the radial functions. We observe that the body of the approximation R of a radial function r defines a hyper-rectangle A_r in the space $X^{(n)}$, which will be said the activation area of r. When an activation area A_r accumulates an error which cannot be reduced any further by the Δ-rule, DRT split it along a dimension, using a method similar to the one used by CART [Breiman et al., 1984]. In order to select the dimension and the split point, DRT keeps a window on the learning events as is done in some approaches to incremental construction of decision trees [Utgoff, 1988]. Moreover, DRT is not prone to unlearning. Both DCL and DRT will be presented in detail in the next three sections.

A regularization–based approach to solve some of the problems of the RAN is proposed by Orr [Orr, 1995]. The method is named Regularization Forward Selection and selects the centres among the samples of the basis functions, that mostly contribute to the output variance. The selection is performed in the regularization framework. Its major drawback is that the fast version of the algorithm is not suitable for on–line learning.

A completely different approach was pursued by [Kadirkamanathan and Niranjan, 1993], which set the problem of centre selection within the framework of functional Hilbert spaces. Introducing the notion of scalar product between two basis functions, the angle they form can be used

as a criterion for inserting new functions. Moreover the authors substituted the gradient descent used in the RAN with an extended Kalman filter. In Section (3.5) a similar formal approach is exploited for introducing a general framework for dynamic algorithms.

3.2 Constructing RBFNs Incrementally

Algorithms DCL and DRT [Blanzieri and Katenkamp, 1996] share a common strategy for the construction of the network. Given an RBFN, the main problem to face, in order to extend the network structure on-line, is to decide when and where in the input space $X^{(n)}$ a new basis function must be inserted.

In order to analyze the nature of the problem, let us consider the two error distributions reported in Figure 3.1 (a) and (b), respectively. In both cases, the errors for a set of input vectors, falling in a region of the space $X^{(n)}$ (for simplicity it is supposed $n = 1$) covered by a bell shape basis function e, are reported. In the case of Figure 3.1 (a), the global error can still be reduced by properly adjusting the weight w associated to e. On the contrary, in the case of Figure 3.1 (b), the error cannot be reduced because, every change in w aimed at reducing the positive errors will inevitably increase the size of the negative errors, and vice–versa.

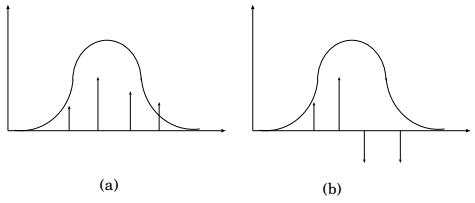


Figure 3.1: Example of two different error patterns for a one–dimensional radial function; arrows represent the error ascribed to the basis function. (a) The error pattern can still be reduced by following the gradient descent. (b) The error pattern cannot be reduced any further.

In other words, the basis function e, is obliged to supply an output value w for too wide a region, so that a good approximation cannot be reached. Of course, there is still hope of reducing the error cumulated on r, by shifting the centre in another region of the input space, but this will probably make the error distribution worse, on the adjacent radial functions. Therefore, if in a region of $X^{(n)}$, many basis functions exhibit an error pattern like the one in Figure 3.1 (b), the only solution is to increase their number, in order to allow a better fit to the target function.

We will now introduce analytic criteria in order to identify this error pattern. We will start supposing that by a learning set LS of samples from the unknown function, acquired off-line, is available. Then we will extend the results to the on-line case. Let M be the cardinality of LS, for a basis function e_j we define the following two error measures:

$$\begin{aligned} E_j &= \sum_{k=1}^{M} E_k e_j(\bar{x}) \\ E_j^{(abs)} &= \sum_{k=1}^{M} |E_k| e_j(\bar{x}) \end{aligned} \tag{3.1}$$

where $E_k = \hat{f}(\bar{x}_k) - f(\bar{x}_k)$. E_j and $E_j^{(abs)}$ are measures of the contribution to the error of the network of the input vector x_k due to the basis function e_j.

When the errors for r_j are all positive or all negative, the condition

$$\frac{|E_j|}{E_j^{(abs)}} = 1$$

holds. On the contrary, when the error distribution is of the type described in Figure 3.1 (b), the relation:

$$\rho_j = \frac{|E_j|}{E_j^{(abs)}} << 1. \tag{3.2}$$

holds. More specifically, when the relation (3.2) holds for several basis functions in a same region of $X^{(n)}$ together with the condition:

$$E_j^{(abs)} >> 0 \tag{3.3}$$

We have found the necessary conditions for inserting new basis functions in the network. The condition (3.2) and (3.3) can be combined in the condition:

$$E_j^{(abs)} - |E_j| = E_j^{(abs)}(1 - \rho_j) >> 0. \tag{3.4}$$

that is the one we use in the following.

However, conditions (3.2) and (3.3) need to be expressed in a form suitable for online learning. A first immediate extension used in this paper, consists in considering not a fixed learning set, but a window on the least M learning events, as is done in some incremental learning algorithms [Utgoff, 1988]. A second technique consists of using an iterative evaluation of E_j and $E_j^{(abs)}$, where the contribution due to events in the past, decays exponentially with time:

$$\begin{aligned} E_j(t+1) &= (1-\alpha)E_j(t) + \\ &\qquad \alpha r_j(\bar{x})(\hat{f}(\bar{x}(t)) - f(\bar{x}(t))) \\ E_j^{(abs)}(t+1) &= (1-\alpha)E_j^{(abs)}(t) + \\ &\qquad \alpha r_j(\bar{x})|\hat{f}(\bar{x}(t)) - f(\bar{x}(t))| \end{aligned} \tag{3.5}$$

In the following, algorithm DCL, will use the relations (3.6) which are totally online, whereas algorithm DRT, which makes use of a window on the learning examples makes use of relations (3.1).

3.3 Dynamic Competitive Learning (DCL)

DCL [Blanzieri and Katenkamp, 1996] is, for the most part, an improvement of the GCS algorithm [Fritzke, 1994a, Fritzke, 1995]. Therefore, we will insist only on the details which are a novelty with respect to the original method. The main novelty is the criterion for deciding when and where to insert a new basis function, which for DCL is stated by the condition (3.4) evaluated according to (3.6), whereas in GCS the decision is based on the square error where it is not possible to distinguish between the two error patterns described in Figure 3.1 (a) and (b), respectively.

The algorithm iterates through a cycle in which the network is first activated, feed–forward on a new input vector $\bar{x}$, then the weights and the radial function widths are updated using the Δ-rule, the centres are updated by the on-line clustering algorithm and the neighbourhood topology is updated by the competitive hebbian learning procedure. The algorithm for extending the network size by adding new basis functions is activated, with a longer period in order to grant the time for adjusting the network parameters between two insertion phases.

The abstract structure of the algorithm is as in the following:

Algorithm DCL

1. Activate the network on a new input vector $\bar{x}$

2. Update the Neighbourhood Topology using the *Competitive Hebbian Learning* procedure

3. Update basis function's centres, using the *Clustering Algorithm*

4. Update basis function's widths and weights using the $\Delta - rule$

5. Every T cycles *do*:

   (a) $E_c = 0$

   (b) Let $\mathbf{E}$ be the set of basis functions satisfying Criterion (3.4) with respect to the cumulated error

   (c) *repeat*

      i. Search the basis function $r_1 \in \mathbf{E}$ with highest value for the expression $E^{(abs)}(1-\rho)$

      ii. Search basis function r_2 among the neighbourhood of r_1 with the highest $E^{(abs)}(1-\rho)$

iii. Insert a new basis function in between r_1 and r_2

iv. $E_c = E_c + E_{r_1}^{abs}$

v. Remove r_1 from $\mathbf{E}$

(d) *until* ($E_c = 0.85 \sum_{j=1}^{N} E_j^{abs}$) or $\mathbf{E} = \emptyset$

The insertion procedure considers only a subset of the basis functions, satisfying the criterion (3.4): the ones, which having the highest value for ρ, contribute to the 85% of the global $E^{(abs)}$ of the network. When a new basis function R_3 is inserted in between two rules r_1 and r_2, for each one of them the width σ is set to the value of the average distance from its neighbours. Then the weight w for r_3 is set $w_{r_3} = \frac{1}{2}(w_{r_1} + w_{r_2})$, aiming at reducing as much as possible the disturbance caused by the new insertion.

The method for learning the neighbourhood topology is basically the one proposed for the 'Neural Gas' algorithm by Martinetz [Martinetz, 1993] [Martinetz and Schulten, 1991] and extended for incremental use from Fritzke [Fritzke, 1995]. The abstract scheme of the algorithm is as follows:

Algorithm MakeTopology

1. Determine the two nearest basis functions for the input signal $\bar{x}$.

2. If an edge exists between this two basis functions, set the age of this edge to zero. If no edge exists, connect them with a new edge and initialize the age of the edge with zero.

3. Increase the age of all edges which are connected to the nearest basis function by one.

4. Remove all edges which have an age higher than $MAXAGE$

The positioning of the gaussian centres is done using an algorithm similar to the *self-organizing feature maps* proposed by Kohonen [Kohonen, 1982][Martinetz and Schulten, 1994]:

- Determine the best matching basis function for the current input signal.
- Increase the matching of the best basis function and its topological neighbours.

Fritzke [Fritzke, 1994a] suggested that for on-line approaches there are some small differences to the Kohonen approach. In the Kohonen's model the strength of the adaptation is decreasing according to a cooling schedule. Moreover, also the topological neighbourhood involved in the training process decreases over time. In Fritzke's approach there are two main differences:

- The adaptation strength is constant over time. We use constant adaptation parameters ϵ_b and ϵ_n for the best matching basis function and the neighbouring basis functions.
- Only the best matching basis function and its direct topological neighbours are adapted.

The changes eliminate the need of a cooling schedule and are more suitable for an on–line approach.

3.4 Dynamic Regression Trees (DRT)

This algorithm [Blanzieri and Katenkamp, 1996] was inspired by Breiman's CART algorithm [Breiman et al., 1984] for inducing regression trees. The basic idea consists in recursively splitting the basis function activation area until a granularity sufficient to fit the target function is obtained.

As in DCL, the Δ-rule and new rule, insertion are interleaved, according to the following abstract scheme:

Procedure Split

1. Activate the network on a new input vector $\bar{x}$
2. Update all the network parameters using the Δ-rule
3. *if* the condition for trying to split holds, *then*
   (a) Select the set **R** of basis functions candidate for split
   (b) For each selected basis function, choose the dimension and the split point; then, if the split point falls inside the activation area split the basis function.

However, the criterion for deciding when to apply the split procedure and the split procedure are specific for DRT. As DRT tries to be very accurate when splitting a basis function, in order to reduce the need of rearranging the basis function position in the input space, the criteria for deciding when and how to split are more sophisticated than the ones used by DCL. In fact, working on-line, the time for splitting a basis function is critical, because the condition (3.4) is valid only when the ρ ratio is estimated on a proper window on the learning events. From a theoretical point of view a split should be created when the gradient descent reaches a local minimum, on the other hand this condition is too heavy, because of the computing time. Moreover with on-line Δ-rule it is not so easy to recognize when a local minimum is reached. Our choice is to update a global error variable according to the basis function:

$$E_g(t+1) = E_g(t)\beta + (1-\beta)|E| \tag{3.6}$$

E being the measure of the current error and β, a factor which weights the contribution of the past history and of E. When E_g reaches a threshold Th_{E_g} the split procedure is activated. In this way, it is likely to activate the split procedure in a proper window.

A second problem, is how many basis functions to split at each time. In fact, adjacent radial functions strongly interact because of overlapping. Therefore, splitting two adjacent basis functions at the same time could be unnecessary, because a single split could be sufficient to reduce the error on both basis functions.

The strategy of DRT is to split simultaneously, only the basis functions which do not interact among themselves. This leads to an improvement in the learning speed and efficiency but introduces the problem of defining when two basis functions are far enough from each other to be considered as independent. One possibility is to use topological information as it is done in the algorithm presented in Section 3.3. In order to tackle this problem in a simpler way the integral of the product of activation function of two basis functions is taken as a measure of their overlapping (see Section B.3.1)

Finally, let r be a basis function selected for splitting, The strategy is to split the activation area of the basis function into two regions, where the accumulated error has an opposite sign, so that it become possible to continue the refinement by performing the gradient descent.

The dimension x and the split point on x can be located by reviewing the learning events recorded in the window LS. Let x_i be the dimension of the input space $X^{(n)}$ currently searched for a split point. First, the instances in LS are sorted according to the increasing value of their projection on x_i and, then, they are presented to the network computing the output error, again. The value on x_i corresponding to the maximum of the absolute value of the accumulated error is selected as a split point.

It is easy to see that this point is the one that split the basis functions the best along the considered input dimension, in fact it divides the basis function in the two parts that have the greatest difference between the accumulated error on the different sides.

The split point is searched for in each one of the input dimensions and the one having the greatest value of accumulated error on both sides of the split point is

selected.

After a basis function has been split, the parameters of the new basis functions are updated according to the following rule:

$$w_L = w_{old} - \frac{E_{tmax}}{\sum_{k \in Left} \pi_k} \tag{3.7}$$

$$w_R = w_{old} - \frac{(E_t - E_{tmax})}{\sum_{k \in Right} \pi_k} \tag{3.8}$$

being w_{old}, w_L and w_R, the original weight and the weights for the basis functions to the left and to the right hand sides of the split point, respectively. Moreover, $Left$ and $Right$ denotes the set of input data falling in the region to the left and to the right of the split point, respectively. The expressions (3.8) set the weights in the direction of the error, this is almost a bet and it can produce a temporary decreasing of the network performance. Anyhow, the Δ-rule usually quickly recovers the loss by adjusting both the weights and the widths of the new basis functions. Moreover, it has been verified, through experiments that even the centre position can still be locally adjusted by the Δ-rule (using a low learning rate), without incurring in the unlearning effect.

3.5 A Formal Characterization of the Structural Changes

In this section we introduce a formal framework for characterizing structural changes. Let us consider the target function f and two approximators h and g. We will assume that h and g belong to a function space in which is defined a scalar product $\langle ., . \rangle$ and the associated norm $\| f \| = \langle f, f \rangle$ are defined.

For instance, it can be $f, g, h \in \mathcal{H}_2$ and

$$\langle f, g \rangle = \int_{\mathcal{R}^n} f(\bar{x}) g(\bar{x}) \, d\bar{x}$$

From this scalar product the L_2 norm can be defined:

$$\| f \| = \langle f, f \rangle = \int_{\mathcal{R}^n} f^2(\bar{x})\, d\bar{x}$$

The norm can be a measure of the quality of the approximation

$$E(f,g) = \| f - g \| = \| f \| + \| g \| - 2\langle f, g \rangle$$

In this case it will be $E(f,g) = ISE$ (see Appendix C).

Then, the error generated by replacing approximator h with the approximator g can be expressed as:

$$\Delta E = E(f,g) - E(f,h) = \| f - g \| - \| f - h \|$$

In order to characterize the error variation, we will use the following result expressed in the form of theorem.

Theorem 1 *Let f, g and h be elements of a vector space V with scalar product $\langle ., . \rangle$ and associated norm $\| . \|$ then the following equality holds:*

$$\| f - g \| - \| f - h \| = \| g - h \| - 2\langle f - h, g - h \rangle$$

Proof.

$$\begin{aligned}
\| g - h \| &= \| (f - h) - (f - g) \| = \| f - h \| + \| f - g \| - 2\langle f - h, f - g \rangle \\
&= \| f - h \| + \| f - g \| - 2\langle f - h, (f - h) - (g - h) \rangle \\
&= \| f - h \| + \| f - g \| - 2 \| f - h \| + 2\langle f - h, g - h \rangle \\
&= \| f - g \| - \| f - h \| + 2\langle f - h, g - h \rangle
\end{aligned}$$

From theorem 1 we obtain:

$$\Delta E = \| g - h \| - 2\langle f - h, g - h \rangle$$

Let us suppose that both approximators h and g are RBFNs and that $h = \mathcal{N}$ and $g = \mathcal{N} + \mathcal{N}_a - \mathcal{N}_d$ is obtained from $\mathcal{N}$ by adding a subnet $\mathcal{N}_a$ and deleting a subnet $\mathcal{N}_d$. The variation of the error becames:

$$\Delta E = \| \mathcal{N}_a - \mathcal{N}_d \| - 2\langle f - \mathcal{N}, \mathcal{N}_a - \mathcal{N}_d \rangle \tag{3.9}$$

$$= \sum_{i,j} \delta(i)\delta(j) w_i w_j \langle e_i, e_j \rangle + 2 \sum_i \delta(i) w_i \langle f - \mathcal{N}, e_i \rangle \tag{3.10}$$

With the extension of the sums limited to the deleted and added functions and

$$\delta(i) = \begin{cases} 1 & e_i \text{ deleted} \\ -1 & e_1 \text{ added} \end{cases}$$

In the Appendix B.1 the result is obtained directly in the particular case of the variation of the Integrated Square Error (ISE corresponding to the norm L_2) associated to the scalar product $\langle e_i, e_j \rangle = S(e_i, e_j) = \int_{\mathcal{R}^n} e_i * e_j \, dx$ obtaining:

$$\Delta E = \sum_{i,j} \delta(i)\delta(j) w_i w_j S(e_i, e_j) + 2 \sum_i \delta(i) w_i \int_{\mathcal{R}^n} (f - \mathcal{N}) * e_i \, dx \tag{3.11}$$

3.5.1 Locality as Orthogonality

In this framework the RBFN's locality property described in Section 3.2 can be expressed via the scalar product between the basis functions. The locality is then enforced by orthogonality: two basis functions are local, one with respect to the other, if they do not overlap more than a given extension, i.e. they are almost orthogonal, their scalar product being close to zero. In the case of Gaussian Basis Function the overlap function is itself a Gaussian (see Appendix B.3.1) and so it is not zero in any point of the domain. Then two Gaussians can be only quasi-orthogonal. During the execution of a dynamic learning algorithm, the approximator obtained after the application of a structural modification, is only partially different from the previous one. In fact functions that are not added or deleted are still the same. Let h and g be

the approximators before and after the structural modification respectively; g can be expressed as the sum of a component parallel to h and of a component perpendicular to h.

Let $g_{\|h}$ and g_{-h} denote the parallel and the perpendicular component respectively, we can write:

$$g = g_{-h} + g_{\|h} = g_{-h} + \frac{\langle g, h \rangle}{\| h \|} h$$

$$g - h = g_{-h} + (\frac{\langle g, h \rangle}{\| h \|} - 1)h$$

Notice that an analogous approach has been used by [Kadirkamanathan and Niranjan, 1993] who defined the angle between two radial function and using it as a locality measure.

3.5.2 Minimization

With the goal of increasing the accuracy of the approximation of the function f an approximator g is considered better than h only if the corresponding ΔE is negative. Thus a good structural change tends to minimize the (3.9). Given function f, this minimization is equivalent to solving the problem of projecting f onto a set of basis functions.

Given the centres and the widths of the basis functions added and deleted during the learning process, the problem of estimating incrementally, the optimum weights associated to the added functions is completely solved (see Appendix B.1.2); equation (B.4) describes the solution. It is worth noting, that adopting a scalar product computed on a finite set (see Appendix B.2), the equation is a generalization of the formula of the pseudoinversion presented in the regularization framework (2.1).

The general problem of determining all the parameters (centres, widths and weights) that minimize equation (3.11) is hard and partially addressed in the Ap-

pendix B.1.2. Moreover, some mathematical results that are needed even for determining approximated solutions (e.g. derivatives of the superposition function) are reported in Appendix B.3 in the case of Gaussian basis function.

3.5.3 Splitting

The case of splitting adopted in the DRT algorithm is a particular case, in which only one function is deleted and two functions are added. In general, the functions are not completely orthogonal, so none of the simplifying hypotheses introduced in Appendix B.1.2 are completely true. We show with a qualitative argument that the splitting technique can be considered an heuristic for minimizing the (3.11). In fact, the splitting technique tries to keep the overlap between the two functions as low as possible (almost orthogonal) and so tries to keep the first term of the equation (3.11) almost at zero. The second term is a sum. If we assume that the weights are always positive, the splitting technique presented in Section 3.4, tends to minimize the positive contributions and maximize the negative ones.

3.5.4 Generality of the approach

It is worth noting that (3.9) and (3.11) are derived from very weak hypothesis: essentially the integrability over $\mathcal{R}^n$. Hence the presented framework is very general: different choices of the basis function lead to the descriptions of different algorithms. For instance, if the basis function is a factorizable car-box function, it can be used for describing a histogram approximator or a regression tree. Moreover, choosing another definition for the scalar product, this framework can be used for obtaining the results on the Least Square Minimization again. In fact, given the learning set S introduced in Section 1.1.3, it is possible to define a scalar product as the sum of the product of the functions for each point $x_k \in S$ with $k = 1 \dots N$.

$$\langle f, g \rangle = \sum_{k=1}^{k=N} f(x_k) g(x_k)$$

With this definition the above argument comes down to the usual problem of minimizing the Empirical Square Error and, in the case of not deleting of basis function, the expression for the computation of the weights w_a, reduced to the computation of the pseudoinverse. In the Appendix B.2 this case is analyzed.

Chapter 4

MODELLING OF COMMUNICATIVE PHENOMENA

The cognitive modelling part of this thesis, is concerned with communicative phenomena. The aim is an analysis of the mental processes, involved in the attribution of intentions to the partner, in a communicative exchange. A main assumption of the research is that such an attribution process, involves weighing up a particular sort of belief, as well as the meaning of the sentence uttered by the partner. We refer to this process as cognitive balance. The hypothesis of the research is two–fold. First, it is possible to model the cognitive balance among beliefs and sentences by a connectionist network. Second, it is possible to find their specific weights by extracting symbolic rules from the network. The properties of Radial Basis Function Networks (i.e. symbolic and statistical interpretations), that are more relevant in this context, are briefly reported and the modelling of the communicative phenomena, is placed in the framework of cognitive pragmatics. Two experiments, as well as the results of the simulations, are described.

4.1 Cognitive Modelling with Radial Basis Function Networks

The symbolic tradition has been the dominant approach to cognitive modelling for a long time. Fodor and Pylyshyn [Fodor and Pylyshyn, 1988] - two of the most representative theorists - claim that only a system with symbolic representations, possessing constituent structure, can adequately model cognitive processes. Only the analysis at the level of symbolic processing, they maintain, is relevant to cognitive theorizing, and this level is nonconnectionist. In a totally different perspective,

[Rumelhart et al., 1986] argue that the cognitive system displays variability, flexibility, and subtlety in its behaviour which are not adequately captured in traditional rule-based systems; to model the actual mechanisms of cognition, more detailed, less brittle models are needed. Cognitive modelling with RBFNs places itself in the mainstream of the latter approach, and it adopts all the hypothesis of the connectionist cognitive modelling, such as, the existence of a level of computation, underlying the distributed representations, and the idea that continuous dynamical systems can describe cognitive phenomena better than discrete symbol manipulating system. Like other connectionist systems and as was presented in the Chapter 1 RBFNs are a class of networks that, given a set of samples, approximate an unknown target function. However, they have also properties quite different from the other Artificial Neural Networks that lead to relevant difference, when RBFNs are used as cognitive models. For instance, they naturally exhibit symbolic interpretation (Section 2.3) and statistical properties (Section 2.5). The RBFN architecture approximates a target function with a weighted sum of receptive field functions (hypergaussians). Such functions are local in that the points of the input space which activate each of them belong to a local area. The weight of each receptive field can be seen as an output value associated to its own local area. The associations between the local areas and the output values can be translated into a production rule of the type:

if $<$ the input is inside the local area of activation $>$,
then $<$ the output is the one suggested$>$.

Via this property it is easy to extract symbolic knowledge from the network. The network computes the average of all the activated rules. Thus, the rules are not completely equivalent to the network, rather, they are symbolic representations of the information encoded into it. This property permits partially bridging the gap between connectionist and nonconnectionist models, allowing the description by means of a set of rules, of what is going on inside the network.

From the statistical point of view, using an RBFN is explicitly equivalent to estimating the underlying density distribution in order to predict one of the variables (Section 2.5). Hence data collected from experiments with human subjects can be analyzed directly.

4.2 Cognitive Modelling of Communication

The study of communication presents some difficulties. On one hand it is necessary to take into account non–linguistic aspects of communication. More specifically, we ought to recognize the role of non–linguistic factors, commonly neglected by computational linguistics. In our perspective, both linguistic and extra–linguistic factors are the expression of a general pragmatic competence. This competence expresses itself through both language and gestures, tone and many other modalities of non–linguistic interaction that are aimed at modifying the mental states of the partner in the communicative exchange.

On the other hand, it is necessary to put the study of communication inside the general framework provided by studies on language and by those achieved from a computational linguistics perspective. According to the latter, the meaning of a phrase is compositionally built, via a cascade process, that starts from the recognition of the syntactical structures of the phrase, which is performed by a parser, and continues with the assembling of the basic semantic components. Only at the end of the entire process, does the system face the problem of solving the meaning ambiguities of the phrase by modelling the context of the communicative interaction, and exploiting the information concerning the pragmatic setting.

Cognitive Pragmatics reconciles the approaches mentioned above: indeed, it is concerned with analyzing the use of language in context. Furthermore, Cognitive Pragmatics focus on the mental processes underlying communication. Given the assumption that the goal of the communication is to modify the mental states of the

partner, the basic idea is that an actor involved in a communicative exchange has mental states and mental representations of the mental states of the partner. At this level, there is no difference between the modification achieved via a linguistic act or via an extralinguistic act. Modelling a communicative phenomena is a great enterprise. It requires making many assumptions and to specify the goals and the methods. So it is therefore necessary to explicitly state the particular framework adopted.

4.2.1 Cognitive Pragmatics

Cognitive Pragmatics Theory is concerned with an analysis of the cognitive processes underlying human communication. The theory, advanced by Airenti, Bara and Colombetti [Airenti et al., 1993], is presented inside the framework of Speech Acts' Theory and, consistently, claims that communication must be considered part of action ([Austin, 1962, Searle, 1969, Searle, 1979]). Indeed, when actor A communicates, either verbally or not, she aims at achieving an effect on partner P, by means of changing his mental states or inducing him to perform an action. Given the assumption that the same analysis holds, for both verbal and nonverbal communication, the terms actor and partner are commonly used instead of the terms speaker and listener. Following this convention, we will also refer to actor A as a female and to partner P as a male.

One of the major assumptions of Cognitive Pragmatics is that, in order to co–operate from the behavioural point of view, the actor and the partner must act on the basis of a plan at least partially shared, that is called a behaviour game between A and P. Consider, for instance, the following example (context: a customer enters in a shoe-shop).

[1] A: I'm looking for a pair of green shoes.

P: Sorry, but they were all sold out last week.

A: Well, I'll have a look elsewhere. Thanks.

An oversimplification of the behaviour game shared by A and P in [1] 4.2.1i is the following:

[2] [BUY-SOMETHING]

1. P gives an object x to A
2. A gives an amount of money y to P

A behaviour game is a stereo–typed pattern of interaction, where the moves of A and P are specified and indicate the type of contribution that each of them is expected to provide, at a certain point in the game, in order to be co–operative. The moves need not be logically necessary, as they just describe typical interactions involving the agents. Besides, the game specifies the situation in which it can be played by the agents, namely its validity conditions. In our example, the game will also specify that one must ask for an article in the proper shop, e.g. A will not ask for a pair of shoes in a bakery, and will pay the amount required for the article, e.g. A will not pay $5 if the shop assistant mentions a price of $60.

The relevance of the notion of behaviour games relies on the fact that, according to Cognitive Pragmatics Theory, a speech act accomplishes a move in a behaviour game. Therefore, it is claimed that, in order to deeply understand the communicative intention of an actor, the partner must realize what game she proposes by means of the utterance. In particular, Airenti and colleagues analyze the process of comprehension of a communicative act and theoretically decompose it in five phases:

1. Literal meaning. The partner reconstructs the mental states literally expressed by the actor.

2. Speaker's meaning. The partner reconstructs the communicative intentions of the actor.

3. Communicative effect. The partner possibly modifies his own beliefs and intentions.

4. Reaction. The intentions for the generation of the response are produced.

5. Response. An overt response is constructed.

In phase 2, all the relevant communicative intentions of the actor are reconstructed by the partner. Their relevance is established by the fact that they manifest the actor's intention to participate in a behaviour game with the partner. Thus, the utterance proffered by A is really understood when it is referred to the behaviour game proposed by A. For instance, an utterance like :

[3] A: Can you raise your arm?

may be interpreted by P as a request to raise his arm or as a request about his possibility to raise the arm. If the behaviour game suggested by the context is [AT THE TAILOR'S], the intended meaning might be a request. On the other hand, if the game suggested by the context is a [MEDICAL EXAMINATION], the actor's meaning might be a request concerning the physical possibility that P is able to raise his arm.

4.3 Evaluation of the Communicative Effect

The aim of this section is an analysis of the inferential processes involved in a speaker's evaluation of the communicative effect achieved on a listener. We present a computational model where such an evaluation process relies on two main factors which may vary according to their strength:

- The verbal commitment of the listener to play his role in the behavioural game actually proposed by the speaker,

- the personal beliefs of the speaker concerning listener's beliefs.

The hypothesis was tested as follows. First, we devised a questionnaire in order to collect human subjects evaluations of communicative effects. Subjects were required

to consider some scenarios and to identify themselves with a speaker. Their task was to evaluate, for each scenario, the communicative effect they had achieved on the listener (acceptance to play the game, refusal, or indecision). Then, we implemented our computational model in a connectionist network; we chose a set of input variables whose combination describes all the scenarios, and we used part of the experimental data to train the network. Finally, we compared the outputs of the network with the evaluations performed by the human subjects. The results are satisfactory.

Our research attempts to model part of the communicative process as it is described by Airenti and colleagues, namely A's evaluation of the communicative effect reached on P. In terms of proposing a game and acceptance of this, the communicative process can be analyzed as follows :

1. A invites P to play a game

2. P responds

3. A evaluates if P adheres to the game.

Indeed, A's comprehension of the response produced by P, can in turn be theoretically analyzed in the phases outlined by Airenti and colleagues. P's response is a starting point for A's reconstruction of his intention to play the game.

The evaluation of the perlocutionary effect consists in a reasoning process, where, according to Cognitive Pragmatics, two factors play a major role. First, A has to take into account the commitment of P to play the game. In particular, the commitment to play a move of the game may be considered an acceptance of the game itself. But, obviously, this is not sufficient to account for the evaluation of the perlocutionary effect. It is still possible, for instance, that P lacks of confidence in A. If A believes that this is the case, she may think she has not achieved the desired effect, even when P expresses the intention to play the game. Possibilities of this type may account

for deceits in communication. Thus, the second factor involved in evaluating the perlocutionary effect is the set of beliefs concerning the mental states of the partner. In our computational model the beliefs of A concerning P's beliefs play a major role in the evaluation of the communicative effect. Note that the notion of shared knowledge we borrow from Cognitive Pragmatics, is a one-sided definition. Since we are concerned with the mental representations of A, it could be the case that:

1. A may take as shared, the knowledge that P does not believes he is sharing with her. In particular, A's and P's representations of the behaviour game that A is proposing may be different.

2. A may erroneously attribute certain beliefs to P.

In our computational model, the attribution of beliefs to the partner, heavily influences the evaluation of the perlocutionary effect, as much as the verbal or non–verbal commitment of the partner to play the game. We refer to the evaluation process as evidential reasoning. Indeed, each of the possible evaluations (P accepts the game/P does not accept the game/it is not clear whether or not P accepts the game) is strengthened or weakened on the basis of the evidence. In our model, beliefs concerning the partner's beliefs and the partner's commitment to play a game are the evidence sought by the actor to evaluate the acceptance of a game. The weight of a given piece of evidence determines how much it should strengthen or weaken the belief that a partner has accepted to play the behaviour game. The cognitive science paradigm dictates three steps for the validation of a computational model. First, the implementation of the model in a program; second, an experiment carried out on human subjects and, third, the comparison between the performance of the program and those of the human subjects. As we implement our model into a connectionist network (i.e. an RBFN), we face the problem of how to use the experimental data. Indeed, in a classical Artificial Intelligence program, the computation

is completely defined from the beginning, and the experimental data are used just for the comparison with the outputs of the program. On the contrary, a connectionist network is, in principle, general purpose and needs to be trained on the experimental data to adapt to a specific task. Thus, we collected human subjects' evaluations of perlocutionary effects both to train the network to evaluate perlocutionary effects, and to observe the fitting of the evaluations of the network with those of the human subjects. The next section is devoted to the questionnaire we administered on the experimental subjects. The questionnaire aimed to collect human subjects' evaluation of communicative effects achieved on an hypothetical partner.

4.3.1 The questionnaire

Subjects

Twenty-four undergraduate students of the University of Turin. They were balanced according to their gender.

Materials and Procedure

Subjects were presented individually with the questionnaire, in a quiet room. At the very beginning they were told to read the following instructions: "Read carefully the story I'll give you, and try to identify yourself with the actor. After reading the story, your task will be to evaluate possible courses of a specific situation. In particular, for each course, you are asked to evaluate whether your partner accepts your proposal to have dinner at home tonight. Possible evaluations are: 'yes' (YES), 'no' (NO) or 'It is not clear' (?). Your evaluations must take into account the story and the information concerning the specific course of the situation." The story tells of the relationship of the actor A (the experimental subject) with her partner P. In particular, it specifies one of the following relationships involving the actor and the partner: confidence, mistrust and uncertainty about confidence. The questionnaires

were balanced according to the three types. Besides, the story tells of a particular situation of everyday life, involving the actor and the partner. For instance, let us consider the story as it is presented to a female experimental subject, where the name of the partner is Paul: "You and Paul usually have dinner together, sometimes at home, sometimes at a restaurant. When you decide to have dinner at home you are not satisfied with a hot-dog in front of the television; dinner is a rite for you and the table must be laid in the appropriate manner. Paul is very good at cooking, whereas you are a disaster: you usually attend to buying food. Now it's 7.00 pm and Paul cannot go out to buy food since he is waiting for a phone call. The table is not laid and you tell him: "I'm going shopping, but I will not lay the table". Information concerning possible courses of the situation specify:

1. the engagement of the partner to cook and to lay the table (engagement, no engagement, refusal to engage). E.g., 'Paul says that he will cook, but he does not intend to lay the table'.

2. the beliefs of the partner concerning the effective intentions of the actor to buy food and not lay the table (sincerity, uncertainty on sincerity, insincerity). E.g., 'Paul believes you are sincere when you say that you will buy food and not lay the table.

4.3.2 The network

We implemented the model in a feed–forward RBFN. In our model, the target function is the actor's evaluation of the adherence of the partner to the game. The function depends on the response of the partner and on his beliefs about the sincerity of the actor. Our network has three layers: five input units, one output unit, and five hidden units, i.e receptive fields. The five input units correspond to the input variables that describe the scenarios presented to the subjects (Figure 4.1).

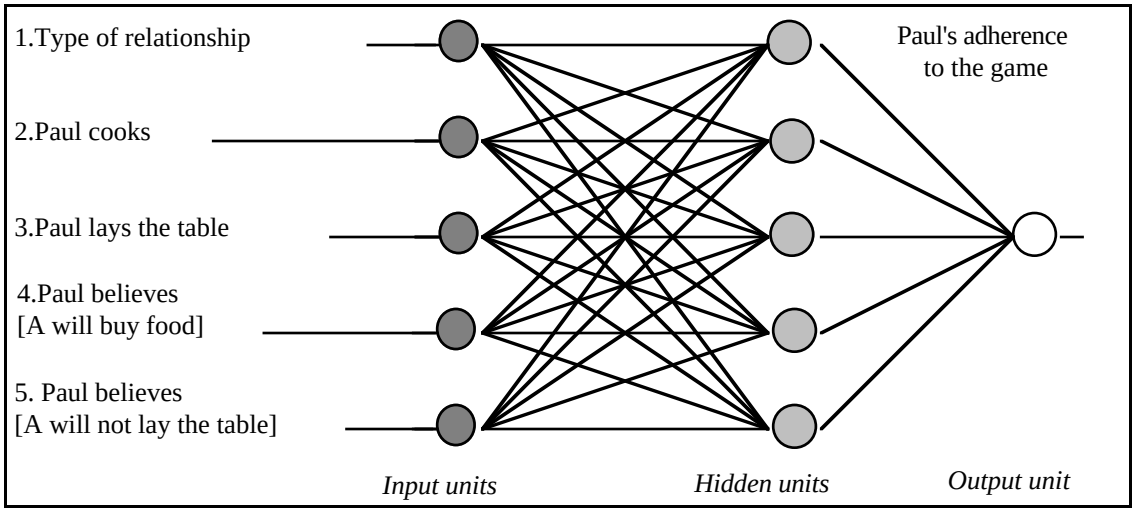


Figure 4.1: The architecture of the network

The value of input 1 refers to the relationship between A and P: confidence (1), mistrust (0) and uncertainty about confidence (0.5). The values of inputs 2 and 3 refer to the fact that P has expressed the intention to play a specific move (value 1), not to play the move (value 0), or he is uncertain (value 0.5). The value of inputs 4 and 5 refer to the reliability of A's engagement to play his move from P's point of view, as represented by A. Value 1 means that A believes that P believes that A intends to perform the declared move (sincerity). Value 0 means that A believes that P believes that A does not intend to perform the declared move (insincerity). Value 0.5 means that A believes that P believes that A is not reliable, namely A may or not perform the declared move (uncertainty on sincerity). The output unit represents the actor's evaluation of the degree of adhesion of the partner to the game; he adheres (value 1), he does not adhere (value 0) or his adhesion is uncertain (value 0.5). The number of units of the hidden layer coincides with the number of production rules. A number of hidden units higher than necessary leads to a poor generalization, i.e. overfitting. If this is the case, the network performs well on the training data set, but has poor performances on the remaining data. On the other

hand, a poor number of hidden units prevents the learning. The network with the optimal number of hidden units performs better than the others. Our experimental data base contains 486 couples scenario/actor's evaluations. We adopted the method of randomly select a population of 20 pairs of training and test sets (2/3 and 1/3 of the couples respectively). Then, we verified the generalization performance for each test set on networks containing different numbers of hidden units (ranging from 4 to 8). The results show that the network with 5 hidden units performs better than the others. The difference is statistically significant (Wilcoxon Test, $p < .05$). Finally, we trained a network with 5 hidden units on the overall data.

4.3.3 Comparison between subjects' evaluations and network's outputs

The trained RBFN is a functional expression of part of the actor's knowledge of the behaviour game. The three different types of evaluation occur as follows: adhesion to the game 31.7%, no adhesion 38.5% and uncertain adhesion 29.8%. We presented the network with the 81 scenarios which were presented to the experimental subjects, and we collected 81 different outputs. The output of the network ranges from 0 to 1. We considered as evaluations of adherence to the game, the outputs with a value greater than 0.66; as no adhesion the outputs with a value lower than 0.33; as uncertain adhesion, the outputs with a value ranging from 0.33 to 0.66.

First, we present the global results concerning the comparison between overall subjects' evaluations with respect to all the scenarios and the evaluations performed by the network. As each evaluation involves selection from three alternatives, the probability of chance-guessing the evaluation of the experimental subjects is .33. As a matter of fact, the correct prevision rate of the network is 63.7 %. The chart in Figure 4.2 visualizes the approximation capability of the network. Second, we compare the evaluations of each subject with those of the network (see Figure 4.3). Results show that the network reproduces more than 50% of the evaluations of 20 subjects. The network does not predict the evaluations of the remaining 4 subjects.

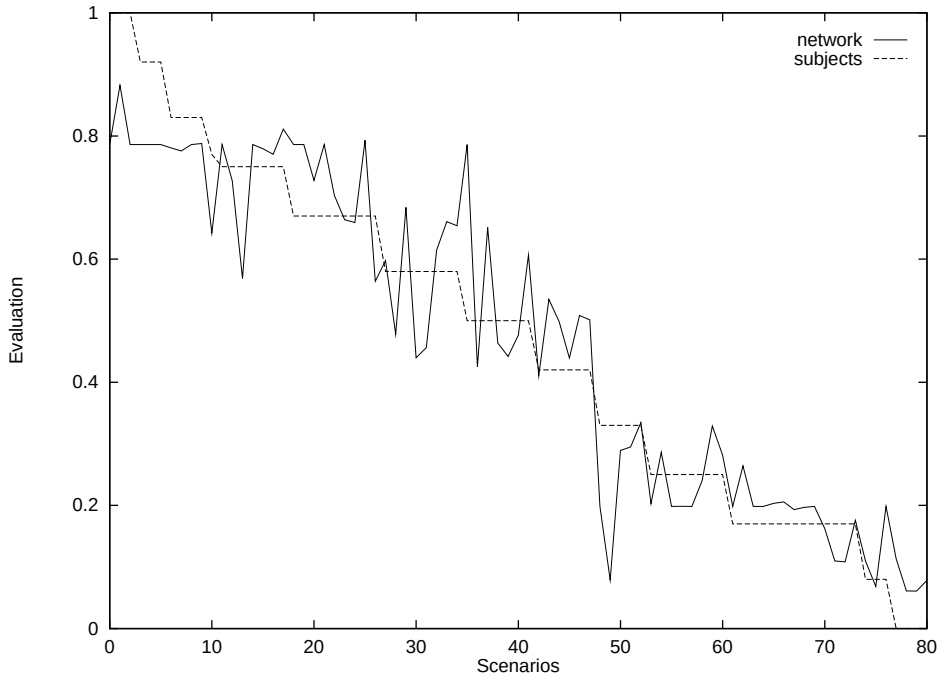


Figure 4.2: The chart shows, for the 81 scenarios, the outputs of the network (continuous line) and the average values of the subjects' evaluations (dashed line).

Finally, the rules extracted from the network are summarized in the Table 4.3.3. As an example, the 5th rule reads as follows:

If < the type of relationship is confidence,
P says that he will not cook,
it is uncertain if P will lay the table,
P believes that A will not buy the food, >
then < P does not accept to play the game.>

Note that in the 5th rule it is irrelevant whether P believes that A will lay the table or not. The five rules are psychologically plausible. Besides, as predicted by our model, the third and the fourth rules suggest that the kind of relationship and the beliefs about partner's beliefs, do influence the evaluation of the communicative

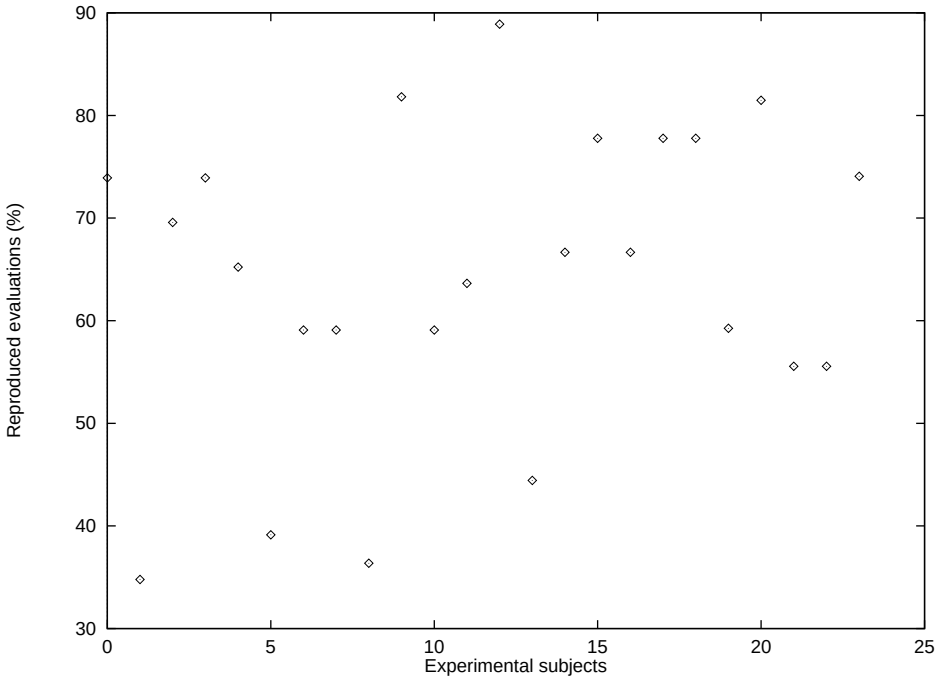


Figure 4.3: Percentages of evaluations reproduced by the network for each of the 24 experimental subjects.

effect, even if the partner declares that he intends to play his role in the game, i.e. to cook.

4.4 Balancing of Sentences and Mental States

In order to deal with some of the difficulties raised during the first experiment, we devised a second questionnaire to collect human data, to train a connectionist network. The scenarios were presented in a more controlled way. While in the first experiment the context information was only described, in the second one it was presented by means of pictures (see Figure 4.4). In fact, we think that visual material favours analogical mental representation of the context information; such representations are built by the reasoners when it is necessary to draw inferences (see [Johnson-Laird, 1983]). Moreover, there was no explicit reference to the kind of rela-

		Rule 1	Rule 2	Rule 3
1	Relationship	Confidence	Mistrust	About confidence
2	P cooks	About yes	Yes	Yes
3	P lays the table	Yes	Irrelevant	No
4	P believes [A buys food]	No	Uncertain	Irrelevant
5	P believes [A lays the table]	Irrelevant	Uncertain	About yes
	Paul's adhesion to the game	About yes	Yes	About no

		Rule 4	Rule 5
1	Relationship	Mistrust	Confidence
2	P cooks	Irrelevant	No
3	P lays the table	No/uncertain	Uncertain
4	P believes [A buys food]	Uncertain	No
5	P believes [A lays the table]	About no	Irrelevant
	Paul's adhesion to the game	No	About no

Table 4.1: The rules extracted from the network. The term 'irrelevant' indicates that the value of a variable does not affect the activation of the rule.

tionship between the actor and the partner.

4.4.1 The Questionnaire

First, we devised a questionnaire to collect human data to train a connectionist network. It was submitted to

Subjects

Twenty-two undergraduate students of University of Turin.

Material and procedure

The task was to figure out whether a character, B, sketched in a series of strip cartoons, would accept an invitation to the opera, even though he had already entered into an engagement. The experimental subject played the role of the inviter A — for the sake of simplicity let A be a female and B, a male. Each subject was presented with a set of information, i.e., a scenario:

1. B's saying whether he will give up his previous engagement (i.e., B's verbal commitment to do a)

2. B's saying whether he will collect the tickets for the opera (i.e., B's verbal commitment to do b)

3. A's attribution of a belief to B concerning the intention of A to book the tickets (i.e., A's attribution of a belief to B concerning the intention of A to do g)

4. B's enablement of giving up his previous engagement (i.e., B's CANDO to do a)

5. B's enablement to collect the tickets for the opera (i.e., B's CANDO to do b)

6. A's enablement to collect the tickets for the opera (i.e., A's CANDO to do b)

The experimental subject was required to evaluate – for each scenario – whether the character B accepts the invitation. Five possible evaluations exists: yes, about yes, perhaps, about no, no.

4.4.2 The Network

Once the data were collected, part of the subjects' evaluations were used to train a Radial Basis Function Network (RBFN). Our network has three layers: six input

units, two output units, and seven hidden units, i.e receptive fields. The six input units correspond to the input variables that describe the scenarios presented to the subjects (Figure 4.5).

In our model, the target function is the evaluation that subject A gives about B's acceptance of the invitation. The function depends on the series of information mentioned above. From A's point of view, we classify such information as follows:

- the sentence uttered by character B (input units 1 and 2);
- B's belief about whether A intends to do g (input unit 3);
- A's and B's mutual beliefs on their enablement to do a or b (input units 4, 5, 6).

The value of an input is set to 1, if the associated condition (e.g. B's verbal commitment to do b) holds, and it is set to 0, if it does not.

The two output units are associated to B's acceptance and B's refusal of the invitation respectively. They represent A's evaluation of B's degree of acceptance: B accepts (value 1), perhaps he accepts (values 0.75, 0.25), he does not accept (value 0.1), perhaps he does not accept (values 0.25, 0.75) or his acceptance is uncertain (values 0.5, 0.5).

The number of units of the hidden layer coincides with the number of production rules. As in the previous experiment we tested networks with different numbers of rules (from 3 up to 10 rules) and there was not any statistically significant difference between their generalization performances. That means that the number of rules is not critical and it can be freely chosen among the ones tested. Thus we trained a network with seven rules on the overall data.

4.4.3 Comparison between Subjects' Evaluations and networks' output

Let us now present the results of the comparison between the performances of the network and the evaluations of the experimental subjects. The data concerning 3 of the 22 experimental subjects were discarded, as their protocols were partially incomplete. Figure 4.6 shows the percentages of subjects' evaluations that are reproduced by the network.

Not considering as an error, the misclassification between two adjacent classes, the trained network reproduces the 79.03% of subjects' evaluations. The average error rate is of 20.97% (standard deviation 10.19). Such a result is satisfactory, as the expected error by guessing is 48.00%. The rules extracted from the network are illustrated in Tables 4.4.3 and 4.4.3. We deem that these rules are really useful to grasp the cognitive balance among sentences and mental states, even though they are not supposed to be followed by actor A. First, the rules allow to describe the specific relevance of beliefs and sentences in the attribution of intention process. Second, each rule allows to understand what gives rise to the subjects' evaluations. In other words, a rule detects the inputs that are indifferent to produce a specific evaluation and, so doing, it detects the inputs that are necessary. Moreover, by comparing the constituents of each antecedent, with the evaluation resulting in the consequent, we discover which inputs have more weight in determining the consequent itself. Now, let us provide some comments for a couple of rules, in order to make clear how they might describe the cognitive balance underlying the attribution of intention.

Consider, for instance, rules 2 and 4. They show that, even though character B is verbally committed to accept the invitation, A may not evaluate the utterance as acceptance. In particular, if B is supposed not to believe in A, then A will evaluate the acceptance as uncertain. In other words, A might be uncertain whether to attribute to B, the intention to accept her invitation. Further, if B cannot complete an action he committed himself to (in our case, to give up his previous engagement), the acceptance

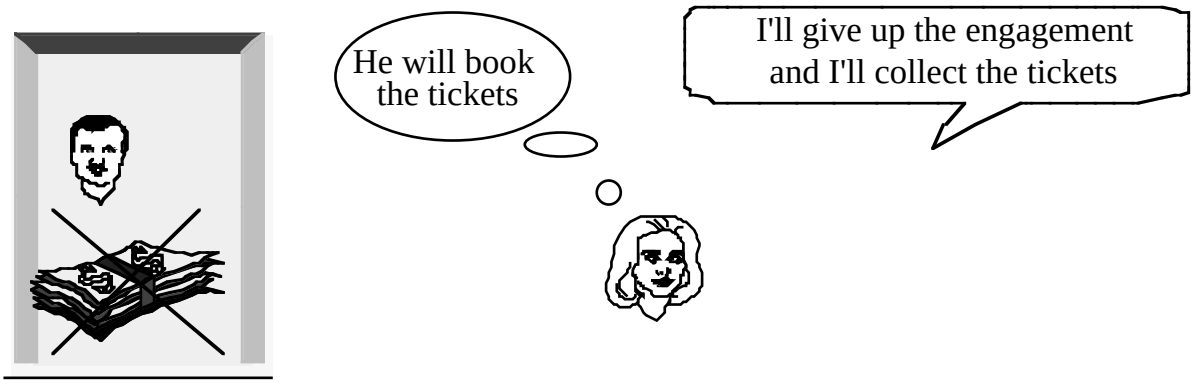
1	If	< B says that he will give up his previous engagement and
		A attributes to B the belief that A will book the tickets and
		A and B share the belief that B can collect the tickets >
	Then	< B accepts the invitation >
2	If	< B says that he will give up his previous engagement and
		B says that he will collect the tickets and
		A attributes to B the belief that A will not book the tickets and
		A and B shared the belief that B can collect the tickets >
	Then	< it is uncertain whether B accepts the invitation or not >
3	If	< B says that he will not give up his previous engagement and
		B says that he will collect the tickets and
		A and B share the belief that B cannot give up his previous engagement and
		A and B shared the belief that B can collect the tickets >
	Then	< Perhaps B does not accept the invitation >
4	If	< B says that he will give up his previous engagement and
		A and B shared the belief that B cannot give up his previous engagement >
	Then	< Perhaps B does not accept the invitation >

Table 4.2: The rules extracted from the network

will be highly improbable. Namely, A might consider as highly improbable that B intends to accept her invitation.

5	If	< B says that he will not give up his previous engagement and
		B says that he will collect the tickets and
		A attributes to B the belief that A will book the tickets and
		A and B share the belief that B can give up his previous engagement and
		A and B share the belief that B cannot collect the tickets >
	Then	< Perhaps B does not accept the invitation >
6	If	< B says that he will not give up his previous engagement and
		B says that he will book the tickets and
		A attributes to B the belief that A will book the tickets and
		A and B share the belief that B cannot give up his previous engagement and
		A and B share the belief that B cannot collect the tickets and
		A and B share the belief that A cannot collect the tickets >
	Then	< B does not accept the invitation >
7	If	B says that he will not give up his previous engagement and
		A attributes to B the belief that A will book the tickets and
		A and B share the belief that B cannot collect the tickets >
	Then	< B does not accept the invitation >

Table 4.3: The rules extracted from the network



Does she accept your invitation?

- ☐ Yes
- ☐ About yes
- ☐ Perhaps
- ☐ About no
- ☐ No

Figure 4.4: The questionnaire

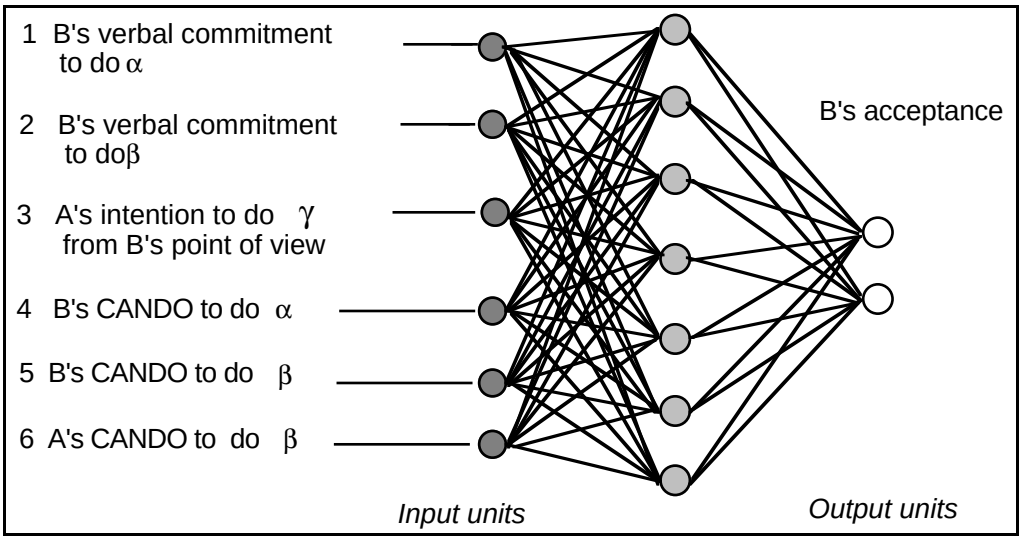


Figure 4.5: The architecture of the network

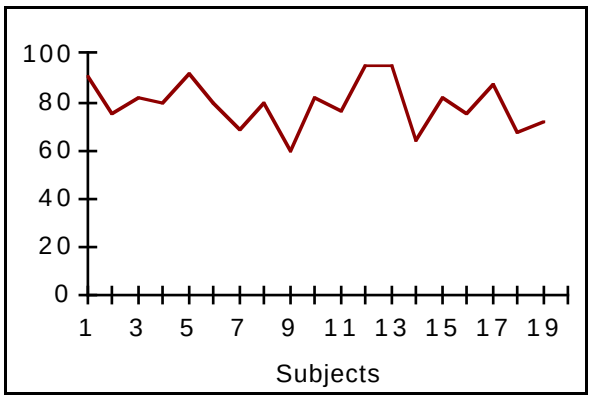


Figure 4.6: Percentages of subjects' evaluations that are reproduced by the network

Chapter 5

APPROXIMATION PROBLEMS

The proposed methodologies have been tested on some applications: i.e. a medical prognosis problem, a time–series prediction task and a regression task on an artificial data set. In general a medical domain combines the need to deal with large amounts of data with the need to provide knowledge understandable to human experts. Therefore, it is well suited for testing the methodologies based on the symbolic interpretation approach, presented in Section 2.3. In particular, the application consists of a classification task (prognosis) developed in a real medical domain where a data–base collected from the Italian National Research Council was available. What appears from the experiments, is that excellent F-RBFNs can be learned from examples, while preserving the symbolic readability, especially when they are synthesized using the symbolic approaches based on CART. A comparison with the well-known Multilayer Perceptron is also presented.

A time–series prediction task is used for testing the validity of the algorithms DCL and DRT presented in Chapter 3. Finally, an experiment with an artificial data set has been designed in order to test DRT, DCL and MLP against the unlearning effect.

5.1 A Medical Prognosis Case

5.1.1 Description of the application

The problem is to estimate the risk of death for patients in intensive care units, starting from a a set of physiological variables (see Table 5.1) measured on a patient

at their arrival in the hospital.

The state-of-the-art method currently used for predicting the outcome, according to these physiological variables, is based on the Simplified Acute Physiology Score (SAPS-II) [J.R. Le Gall, 1993], a statistical method for evaluating the severity of the patient's status. SAPS-II has been developed and validated using logistic regression analysis on more then 8,000 patients and it can be converted to a probability of death.

The problem presents itself as a classification task and it is well suited for our present purposes. In fact a-priori knowledge is available from domain experts (not used here in the learning process) and learned rules can be interesting for them. We used a data–base collected in the context of an epidemiological study on infections diseases in intensive care units, led by CNR on a sample of Italian centres. The research provided the acquisition of physiological and clinical data acquired at the admission of the patient in the unit and the outcome.

The data set available up to now consists of 2,020 patient records, for which the outcome is known. In order to compare the results, only the variables that are involved in the SAPS-II computation were used in the experimentation. Two missing variables have been substituted by others with the same informative content, thereby not affecting the SAPS-II evaluation. The variables (attributes) actually used for learning are shown in Table 5.1.

The data set (2020 records) was randomly split in a learning set (1020 records) and in a validation set (1000 records) and all the data were normalized.

Two groups of experiments were performed. In the first group an F-RBFN was constructed, using the *k-Means* algorithm, trying with different numbers of clusters. In the second group, a version of CART was used. Several alternative rules sets were obtained by running CART with different accuracy requirements (minimum impurity in a leaf). A pruning set of 300 examples was selected from the learning set.

The major difference between the two methods is in the feature selection operated by CART, which systematically leads to more compact and general rules. The best

Parameter	Meaning	Type
AGE	Age	continuous
HR	Heart rate	continuous
SBP	Systolic blood pressure	continuous
TEMP	Body temperature	continuous
V-ROFS	Ventilation, Respiratory Organ Failure Score*	discrete
URO	Urinary flow	continuous
N	Serum urea nitrogen level	continuous
WBC	White Blood Corpuscles count	continuous
NA	Serum sodium level	continuous
K	Serum potassium level	continuous
HCO3	Serum bicarbonate level	continuous
GCS	Glascow Coma Score	continuous
HOFS	Hepatic Organ Failure Score*	discrete
ADM	Type of admission	discrete
CD	Chronic diseases	discrete
OUT	Outcome	discrete (target)

Table 5.1: Attributes used in the medical prognosis problem. (*) Parameters that differs from the ones originally used for SAPS-II computation

results achieved are summarized in Table 5.2, where a comparison with the well–known Multilayer Perceptron and with SAPS-II method is also presented. However, SAPS-II clearly emerges as the most accurate predictor, with an advantage of around 1% over the best result obtained with the neural networks. However, we must take into account that it has been developed using a database much larger and with a low degree of noise. So, its good performance is not surprising. On the other hand MLP and F-RBFN performances are very comparable in terms of accuracy. The

Method	N of rules	$Err_{initial}$	Epochs	Err_{final}
KM+GD	15	24.90	100	23.30
KM+GD	18	26.10	100	22.70
KM+GD	20	24.00	100	22.30
KM+GD	22	24.70	100	24.20
KM+GD	25	24.70	100	25.70
KM+GD	35	24.80	100	24.60
CART+GD	13	22.70	300	22.60
CART+GD	18	22.70	200	22.60
CART+GD	24	22.70	200	22.10
CART+GD	31	24.70	200	21.40
CART+GD	46	24.09	100	22.30
CART+GD	52	24.30	100	21.00
CART+GD	60	25.60	200	22.60
MLP	7	—	6000	21.60
MLP	10	—	2000	21.20
MLP	5+5	—	2000	20.60
SAPS-II	–	—	—	19.30

Table 5.2: Comparative results on the Medical Prognosis Problem. The third column reports the error before performing the gradient descent. In the case of *k-Means* (KM) it corresponds to the untrained network error whereas in the case of CART, it corresponds to the error of the classification tree itself. The fifth column reports the final error after training.

MLP, that shows the best performance, has two hidden layers with a feed–forward fully–connected topology.

Considering the results for F-RBFNs, it is evident that the synthesis procedure based on CART is better than the one based on /em k-Means. Moreover, it is interesting to see how the gradient descent can still significantly increase the performances of the classification theories produced by CART, in a relatively small number of epochs. We see that, even very small set of rules (13 rules) can still reach reasonable performances, thank to the continuous valued semantics and the evidential reasoning performed by the network architecture. However, we wondered whether the gradient descent did blur or not the original symbolic structure. Therefore we tried to map the networks back into rules which were tested again on the data. The result was that, in general, the rules extracted from the network were better than the original ones. For instance, referring to the rule set of 52 rules (12th row in Table 5.2) the extracted rules set had an error rate of 22.60 original 24.30 a rule generated by CART with the corresponding one extracted from the network after refinement.

$$R12(BeforeRefinement) : SBP > 127.5 \wedge V - ROFS > 1.00 \wedge 27.09 < HCO3$$
$$\wedge HCO3 < 29.96 \wedge 4.50 < GCS \wedge GCS < 9.00$$
$$\wedge ADM < 3.00 \rightarrow OUT = 0$$

$$R12(AfterRefinement) : SBP > 130.9 \wedge V - ROFS > 0.97 \wedge 26.90 < HCO3$$
$$\wedge HCO3 < 29.47 \wedge 4.58 < GCS \wedge GCS < 8.65$$
$$\wedge ADM < 3.04 \rightarrow OUT = 0$$

5.2 Test on Mackey-Glass Chaotic Time Series

This case study has been taken from the literature, in order to have a direct comparison with other methods [Crowder, 1990, Jones et al., 1990, Moody and Darken, 1989, Sanger, 1991], and consists of predicting the value 84 time–steps ahead in the Mackey-Glass chaotic series, which are described by the following differential equation:

$$\dot{x} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t)$$

with $x(t<0)=0$, $x(0)=1.2$ and $\tau=17$; then, the task is to predict $x(t+84)$ given $x(t-18)$, $x(t-12)$, $x(t-6)$ and $x(t)$.

In order to correctly predict the function value 84 steps ahead it necessary to capture the generative model of the phenomenon.

The experiment was organized as follows. First, a sequence of 1500 time steps was generated. Then the learning set was obtained by taking the first 1000 whereas the remaining 500 were used as a test set.

The results reported in Table 5.3 compare DCL and DRT with two F-RBFNs having the same number of rules but generated off-line using k-Means and CART, respectively. In all cases, the gradient descent on-line (10.000.000 learning steps) was used to tune the network parameters. For DRT, a window of 100 learning events on the learning set was used. The Non-Dimensional Error Index (NDEI) is defined as the Root Mean Square Error divided by the Standard Deviation of the target series. The performances of the networks constructed incrementally look very similar to the ones of the networks generated off-line. Moreover, DCL reaches an accuracy higher than DRT.

In table 5.4, the comparison is made using the gradient descent off-line (i.e. updating the weights only after all the learning set has been seen), both for the network generated by DCL and DRT and the ones generated by CART and k-Means. Finally, Table 5.5 reports a comparison with some of the best results reported in the literature, such as ANFIS [Jang, 1993], a fuzzy neural network de-

signed by Jang and an F-RBFN generated using a symbolic learner called SMART+ [Botta and Giordana, 1993] which allows a domain theory to be take into account. However, considering that on-line learning is inherently suboptimal with respect to off-line learning, the results of DCL and DRT appear satisfactory in comparison to the ones reported in the literature.

Table 5.3: Performance after training for 1.000.000 learning events with on-line gradient descent. DCL = Dynamical Competitive Learning; DRT = Dynamical Regression Trees; KMEANS = Layout generated using the k-means algorithm; CART = Layout generated using CART algorithm.

Type	Rules	NDEI
DCL	80	0.051
	100	0.042
	120	0.047
DRT	80	0.074
	100	0.064
	120	0.057
KMEANS+GD	80	0.040
	100	0.037
	120	0.037
CART+GD	78	0.059
	103	0.071
	118	0.074

5.3 Comparison with Respect to the Unlearning Effect

In order to check the capability of DRT and of DCL of escaping to the unlearning effect the following test has been designed. A data set F was created by randomly sampling

Table 5.4: Performance after training with off-line gradient descent(Batch Learning) for 6.000 epochs.

Type	Rules	NDEI
DCL	80	0.043
	100	0.036
	120	0.045
DRT	80	0.051
	100	0.049
	120	0.047
KMEANS+GD	80	0.023
	100	0.023
	120	0.023
CART+GD	78	0.034
	103	0.031
	118	0.032

the function $F(x,y) = sin(x)cos(y)$ in the domain $D = \{0 \leq x \leq \pi, 0 \leq y \leq \pi\}$. Then F was partitioned into two disjoint data sets $F_A = \{F(x,y) \in F | (x \leq 1.8, y \leq 1.8) \vee (x \geq 1.4, y \geq 1.4)\}$ and $F_B = \{F(x,y) \in F | (x > 1.8, y < 1.4) \vee (x < 1.4, y > 1.8)\}$. The three data sets are described in Figure 5.1.

Afterwards, both DCL and DRT underwent a training cycle repeated four times in which a network was trained on-line for 500 times on the set F_A and then 500 times on the set F_B. For a comparison, the same procedure was repeated both with a multi-layer perceptron (trained off-line) and with an F-RBFN, initialized by applying one–step, the k-means clustering algorithm on the whole learning set F.

The descent of the Non-Dimensional Error Index (NDEI) along the training phases is reported in Figure 5.2, where it appears evident that the DRT algorithm is quite robust with respect to the unlearning effect, while DCL is very unstable, even more

Table 5.5: Comparison with some results reported in the literature

Type	NDEI	Comment
ANFIS	0.036	Fuzzy controller
MLP	0.050	Multilayer perceptron
KMEANS+GD	0.023	Standard k-means
CART+GD	0.031	CART + gradient descent
SMART+GD	0.032	Symbolic Learner
DRT_{online}	0.057	Dynamic Regr. Trees
DRT_{batch}	0.047	Dynamic Regression trees
$DCL + GD_{online}$	0.042	Dynamic Comp. Learning
$DCL + GD_{batch}$	0.036	Dynamic Comp. Learning

than the multi-layer perceptron. Finally, it is worth noting that the classical RBFN, learned using k-Means in one step, is extremely stable with respect to the gradient descent, even if when Δ-rule is allowed to tune the centre position of the Gauss's functions as was done in this case.

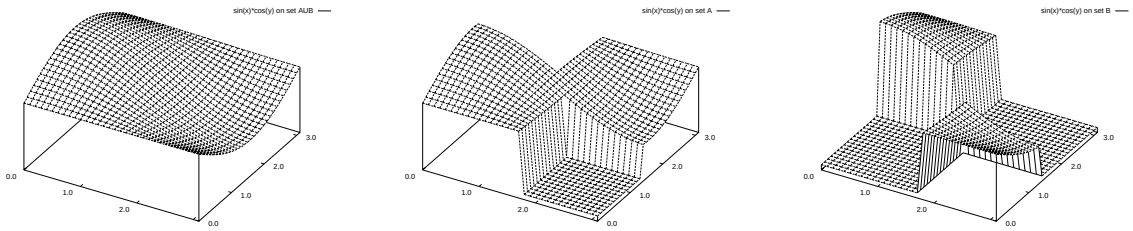


Figure 5.1: Learning set

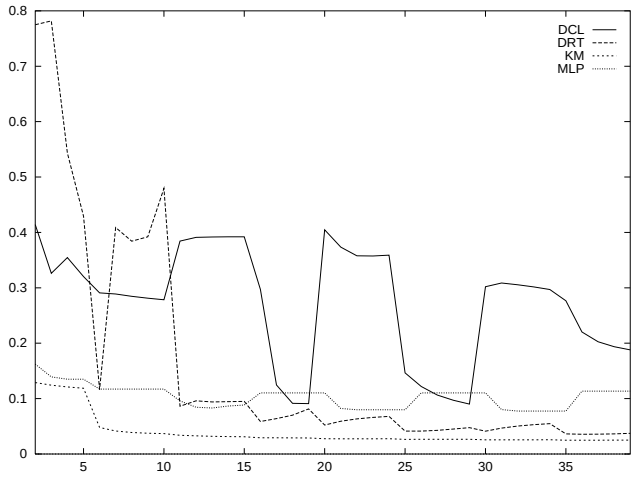


Figure 5.2: Non-dimensional error index time evolution obtained by alternating 500 epochs on set A and 500 on set B

CONCLUSIONS

This thesis is concerned with Radial Basis Function Networks (RBFNs) and it addresses the problem of using them as cognitive models. We have introduced the feed–forward version of the network and placed it in the framework of function approximation problems. From the technical point of view, the main topic has been the basic architecture of the RBFNs as well as the different approaches (Artificial Neural Networks, Regularization Theory, Fuzzy Controllers, Symbolic Interpretation, Statistical Approach, Instanced Based Learning) and their correspondent algorithms. After having introduced the distinction between static and dynamic algorithms, attention has been focussed on Dynamic Competitive Learning (DCL) and Dynamic Regression Tree (DRT) algorithms. Moreover, a formal framework for describing static as well as dynamic algorithms has been presented.

From the cognitive point of view, attention has been focussed on how it is possible to exploit some of the properties of RBFNs in order to develop cognitive models. More specifically, the thesis has presented a model of the cognitive processes involved in the evaluation of the communicative effect and in the balancing between sentences and mental states in the attribution of intentions process. The symbolic interpretation property of RBFNs is exploited to give a rule-base description of the simulated phenomenon. Finally, the applications of these techniques to some approximation problems have been presented.

An important point of the work is the unifying view, that have been proposed in order to compare and to integrate all the different approaches. RBFNs are particularly suitable for integrating the symbolic and connectionist paradigms in the line draw by Towell and Shavlik [Towell and Shavlik, 1994] for the usual ANN's architec-

ture. With respect to this work the symbolic interpretation emerges from RBFNs in a more natural way and so knowledge extraction requires simpler algorithms. More specifically, this thesis has presented a technique for using classical symbolic learning algorithms in order to synthesize a network, which can be refined, using other methods like gradient descent. After refinement, the network tends to preserve its symbolic readability. Another basic property of the RBFN is the locality that permits the synthesis of incremental dynamic algorithms like DCL and DRT. The DCL algorithm uses a kind of incremental clustering, resembling Kohonen's maps, in order to distribute the hidden unit over the domain of the target function. DRT uses a strategy derived from the regression tree learners. Both algorithms proved to be reasonably accurate although approximations learned on–line do not reach the performances of the ones learned off–line (as it is logical to expect). However, DCL produced more accurated approximation than DRT and moreover has the big advantage that it does not need to remember the past history. On the other hand, the incremental clustering algorithm embedded into DCL, is subject to unlearning because the hidden units can move anywhere in the input space following the data. This problem does not occur with DRT which always guarantees the full coverage of the input space, owing to the splitting strategy it uses. The formal framework proposed for dynamic algorithms, describes the locality in terms of orthogonality between functions. The variation of the approximation error, due to a structural change has been analytically expressed, and the problem of its minimization partially solved. Moreover, the partial optimality of DRT is explained within this framework in a qualitative way.

The simulation of the evaluation of the effect achieved by an actor to a communicative partner exploited some the properties of the presented architecture. The model is implemented by an RBFN, whose performances are compared with those of human subjects. The results show that the network reproduces most of the experimental subjects' evaluations. We argue that the rules extracted from the network shed light on which kind of evidence is relevant in the evaluation of the communicative effect.

Our model is particularly concerned with the notion of thinking about beliefs, advanced by Baron (1988). Indeed, the author suggests that thinking is involved in the formation of belief, namely when people consider how strongly to believe something, or, which of several competing beliefs is true. In this process, evidence consists of any object that helps them to determine the extent to which a possibility achieves some goal. We argue that the concept of evidence is particularly suited to describing the inferential processes involved in communication, where inferential bounds are built quickly and effortlessly. Whereas symbol manipulation seems to account for children and adults' ability to draw formal inferences (see, for instance [Bara et al., 1995]), it is possible that explicit knowledge is not required in dealing with the evaluation of the perlocutionary effect. The brain may use implicit representations, which are based on a parallel distributed process. In our view, cognitive phenomena such as reasoning and problem solving are inexplicable, except by positing symbols that are systematically manipulated. On the other hand, we claim that to model some of the mental processes underlying communication, the connectionist networks are more suitable. In particular, the assumption underlying our research, is that the attribution of intentions in a communicative exchange is not rule-governed, but rather it can only (approximately) be described in a rule-based way. Connectionism postulates distributed representations, very different from static symbolic representations, and the dynamics of the system owes more to statistical mechanics, than to logic. Nevertheless, it may be that these representations and the dynamics which transform one such representation into another, can form the basis of a theory of inference ([Levesque, 1988]). As far as our aims are concerned, a relevant feature of connectionist networks is flexibility: our model deals with flexibility as it claims that the actor's beliefs, may vary according to their strength, therefore resulting in possible different evaluations of the communicative effect. As a conclusion, a connectionist model may give an account of the cognitive processes involved in the evaluation of the perlocutionary effect. It is plausible that human beings are endowed with mechanisms

that allow evidence to be weighted, which has been collected in the light of specific behaviour goals. Such mechanisms would underlie the ability to draw inferences in communicative exchanges.

Although the simulative approach assumes cognitive pragmatics [Airenti et al., 1993] as the reference theory, it shares some aspects with the notion of relevance introduced by Sperber and Wilson [Sperber and Wilson, 1986]. In fact, the function that we implicitly have tried to approximate is really similar to the relevance of a sentence in the particular context and in the same way it is hard to be described and formalized.

The potential of the RBFN architecture is not yet been exploited completely. The directions of future work are various. From the technical point of view, the formal framework for describing the structural modification deserves further attention in order to obtain effective new dynamic algorithms. Recurrent versions of RBFNs, i.e. networks that are able to approximate dynamical systems, require analysis to verify the possibility of extending their multiple interpretability to more expressive systems (Finite State Automata, Feature Grammars, Markov Chains and more complex logical theories) They can also provide a powerful system for cognitive modelling with dynamical systems in the direction pointed to by Giunti [Giunti, 1996], enhancing the cognitive modelling practise with the ideas of observability, controllability and stability that can be critical for modelling complex cognitive phenomena, as was done for conscious perception [Mathis and Mozer, 1996].

From the cognitive point of view, RBFNs, provided with dynamic learning algorithms, are relevant in the present debate between constructivism and nativism [Blanzieri, 1998]. In fact, this kind of algorithm is constructive exactly in the sense expressed by Quartz and Sejnowski [Quartz and Sejnowski, 1998]: new units, typically the hidden ones, are added to the network changing both its representational power and the search space for the optimal or suboptimal parameters. Quartz and Sejnowski present biological and psychological evidence in favour of the growing of

the neural system during the developmental phases; their data support the RBFNs' approach. In particular the considerations about the dendritic growing and the speculations about their nonlinear computational capabilities show that even nonstandard networks, i.e. those not based on the multilayer perceptron, can be biologically plausible. Moreover, RBFNs verify the locality and stability conditions and also the clustering property suggested by Quartz and Sejnowski. Hence dynamic learning algorithms for RBFNs can be useful for modelling cognitive development.

Appendix A

A COLLECTION OF MATHEMATICAL RESULTS

A.1 Matrix Derivatives

In this section we collected some results on the derivatives of matrixes that we will need in the following. Let A be a real matrix with elements a_{ij} we define the derivative of A with respect to α, and call it $\frac{\partial A}{\partial \alpha}$, as the matrix with elements $\frac{\partial a_{ij}}{\partial \alpha}$.

$$\frac{\partial A}{\partial \alpha} = \left(\frac{\partial a_{ij}}{\partial \alpha} \right)$$

Proposition 1 *Let A, B be real matrixes then*

$$\frac{\partial AB}{\partial \alpha} = \frac{\partial A}{\partial \alpha} B + A \frac{\partial B}{\partial \alpha}$$

Proof.

$$\begin{aligned} \frac{\partial AB}{\partial \alpha} &= \left(\frac{\partial}{\partial \alpha} \sum_q a_{iq} b_{qj} \right) = \left(\sum_q \frac{\partial a_{iq}}{\partial \alpha} b_{qj} + a_{iq} \sum_q \frac{\partial b_{qj}}{\partial \alpha} \right) \\ &= \left(\sum_q \frac{\partial a_{iq}}{\partial \alpha} b_{qj} \right) + \left(a_{iq} \sum_q \frac{\partial b_{qj}}{\partial \alpha} \right) = \frac{\partial A}{\partial \alpha} B + A \frac{\partial B}{\partial \alpha} \end{aligned}$$

Proposition 1 *Let A be a real matrix and the inverse matrix A^{-1} exists then*

$$\frac{\partial A^{-1}}{\partial \alpha} = -A^{-1} \frac{\partial A}{\partial \alpha} A^{-1}$$

Proof.

$$\begin{aligned} & AA^{-1} = I \\ & \frac{\partial AA^{-1}}{\partial \alpha} = \frac{\partial I}{\partial \alpha} = 0 \\ & \frac{\partial A}{\partial \alpha} A^{-1} + A \frac{\partial A^{-1}}{\partial \alpha} = 0 \\ & \frac{\partial A^{-1}}{\partial \alpha} = -A^{-1} \frac{\partial A}{\partial \alpha} A^{-1} \end{aligned}$$

Appendix B

STRUCTURAL MODIFICATION

B.1 Integrated Square Error

Let $f : R^n \rightarrow R$ be the unknown function and $\mathcal{N} : R^n \rightarrow R$ the approximator expressed by

$$N(\bar{x}) = \sum_{i=1}^{n} w_i e(\| \bar{x} - \bar{c}_i \|_i) = \sum_{i=1}^{n} w_i e_i(\bar{x})$$

where e_i contains all the parameters that identify the distance $\| \, . \, \|_i$ and the centre c_i.

In the following, we will omit the dependence of e_i from $\bar{x}$.

Considering the L_2 norm, the distance between the f and $\mathcal{N}$ can be written as

$$\begin{aligned} \| f - \mathcal{N} \|_{\mathcal{R}^n} &= \int_{\mathcal{R}^n} (f - \mathcal{N})^2 \, dx \\ &= \int_{\mathcal{R}^n} f^2 \, dx + \int_{\mathcal{R}^n} \mathcal{N}^2 \, dx - 2 \int_{\mathcal{R}^n} f * \mathcal{N} \, dx \\ &= \| f \|_{\mathcal{R}^n} + \| \mathcal{N} \|_{\mathcal{R}^n} - 2 \int_{\mathcal{R}^n} f * \mathcal{N} \, dx \end{aligned}$$

$$\begin{aligned} \| \mathcal{N} \|_{\mathcal{R}^n} &= \int_{\mathcal{R}^n} \mathcal{N}^2 \, dx \\ &= \int_{\mathcal{R}^n} \sum_i w_i e_i \sum_j w_j e_j \, dx \\ &= \sum_{i,j} w_i w_j \int_{\mathcal{R}^n} e_i * e_j \, dx \\ &= \sum_{i,j} w_i w_j S(e_i, e_j) \end{aligned}$$

Where $S(e_i, e_j) = \int_{\mathcal{R}^n} e_i * e_j \, dx$

$$\begin{aligned}\int_{\mathcal{R}^n} f * \mathcal{N}\, dx &= \int_{\mathcal{R}^n} f * \sum_i w_i e_i\, dx \\ &= \sum_i w_i \int_{\mathcal{R}^n} f * e_i\, dx\end{aligned}$$

$$\| f - \mathcal{N} \|_{\mathcal{R}^n} = \| f \|_{\mathcal{R}^n} + \sum_{i,j} w_i w_j S(e_i, e_j) - 2 \sum_i w_i \int_{\mathcal{R}^n} f * e_i\, dx \qquad \text{(B.1)}$$

If we remember that the error function is defined as $\mathcal{E} = f - \mathcal{N} = f - \sum_j w_j e_j$ the equation B.1 is:

$$\| f - \mathcal{N} \|_{\mathcal{R}^n} = \| f \|_{\mathcal{R}^n} - \sum_{i,j} w_i w_j S(e_i, e_j) - 2 \sum_i w_i \int_{\mathcal{R}^n} \mathcal{E} * e_i\, dx \qquad \text{(B.2)}$$

B.1.1 Structural modifications

Let us suppose that the initial network $\mathcal{N}_n$ has n basis functions and we add n_a functions and delete n_d functions so that the final network $\mathcal{N}_m$ has $m = n + n_a - n_d$ basis functions. Without loss of generality we can suppose that the functions that we delete are the first n_d. The treatment for each function is shown below:

$$i = \begin{cases} 1, \ldots, n_d & \text{delete } e_i \\ n_d + 1, \ldots, n & \text{do nothing to } e_i \\ n + 1, \ldots, n + n_a & \text{add } e_i \end{cases}$$

Let us express the total error E_m of the network $\mathcal{N}_m$ in terms of the error function $\mathcal{E}$ of the network $\mathcal{N}_n$.

$$\begin{aligned}\| f - \mathcal{N}_m \|_{\mathcal{R}^n} &= \| f \|_{\mathcal{R}^n} + \sum_{i=n_d+1}^{i=n+n_a} \sum_{j=n_d+1}^{j=n+n_a} w_i w_j S(e_i, e_j) - 2 \sum_{i=n_d+1}^{i=n+n_a} w_i \int_{\mathcal{R}^n} f * e_i\, dx \\ &= \| f \|_{\mathcal{R}^n} + \sum_{i=n_d+1}^{i=n+n_a} \sum_{j=n_d+1}^{j=n+n_a} w_i w_j S(e_i, e_j) - 2 \sum_{i=n_d+1}^{i=n+n_a} \sum_{j=1}^{j=n} w_i w_j S(e_i, e_j) \\ &\quad -2 \sum_{i=n_d+1}^{i=n+n_a} w_i \int_{\mathcal{R}^n} \mathcal{E} * e_i\, dx\end{aligned}$$

The difference between the errors of $\mathcal{N}_m$ and $\mathcal{N}_n$ using the equation B.2 is:

$$\begin{aligned}
\Delta E &= \| f - \mathcal{N}_m \|_{\mathcal{R}^n} - \| f - \mathcal{N}_n \|_{\mathcal{R}^n} \\
&= \sum_{i=n_d+1}^{i=n+n_a} \sum_{j=n_d+1}^{j=n+n_a} w_i w_j S(e_i, e_j) - 2 \sum_{i=n_d+1}^{i=n+n_a} \sum_{j=1}^{j=n} w_i w_j S(e_i, e_j) \\
&\quad -2 \sum_{i=n_d+1}^{i=n+n_a} w_i \int_{\mathcal{R}^n} \mathcal{E} * e_i \, dx \\
&\quad + \sum_{i=1}^{i=n} \sum_{j=1}^{j=n} w_i w_j S(e_i, e_j) + 2 \sum_{i=1}^{i=n} w_i \int_{\mathcal{R}^n} \mathcal{E} * e_i \, dx
\end{aligned}$$

Each term with a double sum can be decomposed:

$$\begin{aligned}
&\sum_{i=n_d+1}^{i=n+n_a} \sum_{j=n_d+1}^{j=n+n_a} w_i w_j S(e_i, e_j) = \sum_{i=n_d+1}^{i=n} \sum_{j=n_d+1}^{j=n} w_i w_j S(e_i, e_j) \\
&+ \sum_{i=n+1}^{i=n+n_a} \sum_{j=n+1}^{j=n+n_a} w_i w_j S(e_i, e_j) + 2 \sum_{i=n_d+1}^{i=n} \sum_{j=n+1}^{j=n+n_a} w_i w_j S(e_i, e_j)
\end{aligned}$$

$$\begin{aligned}
&\sum_{i=n_d+1}^{i=n+n_a} \sum_{j=1}^{j=n} w_i w_j S(e_i, e_j) = \\
&+ \sum_{i=n_d+1}^{i=n} \sum_{j=1}^{j=n_d} w_i w_j S(e_i, e_j) + \sum_{i=n+1}^{i=n+n_a} \sum_{j=1}^{j=n_d} w_i w_j S(e_i, e_j) \\
&+ \sum_{i=n_d+1}^{i=n} \sum_{j=n_d+1}^{j=n} w_i w_j S(e_i, e_j) + \sum_{i=n+1}^{i=n+n_a} \sum_{j=n_d+1}^{j=n} w_i w_j S(e_i, e_j)
\end{aligned}$$

$$\begin{aligned}
&\sum_{i=1}^{i=n} \sum_{j=1}^{j=n} w_i w_j S(e_i, e_j) = \sum_{i=1}^{i=n_d} \sum_{j=1}^{j=n_d} w_i w_j S(e_i, e_j) \\
&+ \sum_{i=n_d+1}^{i=n} \sum_{j=n_d+1}^{j=n} w_i w_j S(e_i, e_j) + 2 \sum_{i=n_d+1}^{i=n} \sum_{j=1}^{j=n_d} w_i w_j S(e_i, e_j)
\end{aligned}$$

Substituting and simplifying:

$$\begin{aligned}\Delta E &= \sum_{i=n+1}^{i=n+n_a}\sum_{j=n+1}^{j=n+n_a} w_i w_j S(e_i,e_j) - 2\sum_{i=n+1}^{i=n+n_a}\sum_{j=1}^{j=n_d} w_i w_j S(e_i,e_j) \\ &\quad + \sum_{i=1}^{i=n_d}\sum_{j=1}^{j=n_d} w_i w_j S(e_i,e_j) - 2\sum_{i=n+1}^{i=n+n_a} w_i \int_{\mathcal{R}^n} \mathcal{E} * e_i \, dx + 2\sum_{i=1}^{i=n_d} w_i \int_{\mathcal{R}^n} \mathcal{E} * e_i \, dx\end{aligned}$$

We call $\mathcal{N}_a$, the network that we obtain, considering only the functions that we add and in a similar way $\mathcal{N}_d$, the network composed by the functions that we delete. Moreover we define the function δ as

$$\delta(i) = \begin{cases} 1 & e_i \text{ deleted} \\ -1 & e_1 \text{ added} \end{cases}$$

With the extension of the sums limited to the functions that we delete or add, the difference of the error can be written as:

$$\begin{aligned}\Delta E &= \sum_{i,j} \delta(i)\delta(j) w_i w_j S(e_i,e_j) + 2\sum_i \delta(i) w_i \int_{\mathcal{R}^n} \mathcal{E} * e_i \, dx \\ &= \| \mathcal{N}_d - \mathcal{N}_a \|_{\mathcal{R}^n} + 2\sum_i \delta(i) w_i \int_{\mathcal{R}^n} \mathcal{E} * e_i \, dx\end{aligned}$$

B.1.2 Minimization of ΔE

Without loss of generality we can reorder the set of basis functions so that

$$i = \begin{cases} 1, \ldots, n_d & \text{delete } e_i \\ n_d + 1, \ldots, n_d + n_a & \text{add } e_i \\ n_d + n_a + 1, \ldots, n + n_a & \text{do nothing to } e_i \end{cases}$$

The ΔE can be expressed as:

$$\Delta E = \sum_{i=1}^{n_d+n_a}\sum_{j=1}^{n_d+n_a} \delta(i)\delta(j) w_i w_j S(e_i,e_j) + 2\sum_{i=1}^{n_d+n_a} \delta(i) w_i \int_{\mathcal{R}^n} \mathcal{E} * e_i \, dx \qquad \text{(B.3)}$$

In order to find the optimum of the increment of error caused by a structural change we have to minimize the second term of the equation B.3.

We define the two $1 \times (n_d + n_a)$ vectors $\bar{w}$ and $\bar{\xi}$ and the $(n_d + n_a) \times (n_d + n_a)$ matrix S as:

$$\begin{aligned} \bar{w} &= (\delta(i) w_i) \\ \bar{\xi} &= \left(\int_{\mathcal{R}^n} \mathcal{E} * e_i \, dx \right) = (\xi_i) \\ S &= (S(e_i, e_j)) = (s_{ij}) \end{aligned}$$

The vectors $\bar{w}$ and $\bar{\xi}$ and the matrix S can be partitioned, depending on wether the function is added or deleted, i.e. depending on the sign of $\delta(i)$:

$$\begin{aligned} \bar{w} &= \begin{bmatrix} \bar{w}_d & -\bar{w}_a \end{bmatrix} \\ \bar{\xi} &= \begin{bmatrix} \bar{\xi}_d & \bar{\xi}_a \end{bmatrix} \\ S &= \begin{bmatrix} S_{dd} & S_{da} \\ S_{ad} & S_{aa} \end{bmatrix} \end{aligned}$$

It is possible to write the ΔE as:

$$\begin{aligned} \Delta E &= \bar{w} S \bar{w}^T + 2 \bar{w} \bar{\xi}^T \\ &= \bar{w}_d S_{dd} \bar{w}_d^T - 2 \bar{w}_d S_{da} \bar{w}_a^T + \bar{w}_a S_{aa} \bar{w}_a^T + 2 \bar{w}_d \bar{\xi}_d^T - 2 \bar{w}_a \bar{\xi}_a^T \end{aligned}$$

Let us derive ΔE with respect to the vector $\bar{w}_a$

$$\frac{\partial \Delta E}{\partial \bar{w}_a} = -2 \bar{w}_d S_{da} + 2 \bar{w}_a S_{aa} - 2 \bar{\xi}_a$$

The condition for the minimization is:

$$\bar{w}_d S_{da} + \bar{w}_a S_{aa} - \bar{\xi}_a = 0$$

$$\bar{w}_a = (\bar{w}_d S_{da} + \bar{\xi}_a) S_{aa}^{-1} \tag{B.4}$$

Let us substitute the B.4 into the expression of ΔE.

$$\begin{aligned}
\Delta E &= \bar{w}_d S_{dd} \bar{w}_d^T - 2\bar{w}_d S_{da} S_{aa}^{-1} (\bar{w}_d S_{da} + \bar{\xi}_a)^T + (\bar{w}_d S_{da} + \bar{\xi}_a) S_{aa}^{-1} (\bar{w}_d S_{da} + \bar{\xi}_a)^T \\
&\quad + 2\bar{w}_d \bar{\xi}_d^T - 2(\bar{w}_d S_{da} + \bar{\xi}_a) S_{aa}^{-1} \bar{\xi}_a^T = \\
&= \bar{w}_d S_{dd} \bar{w}_d^T + 2\bar{w}_d \bar{\xi}_d^T - (\bar{w}_d S_{da} + \bar{\xi}_a) S_{aa}^{-1} (\bar{w}_d S_{da} + \bar{\xi}_a)^T
\end{aligned}$$

The ΔE depends on the added functions only through a biquadratic term. Deriving it with respect to a generic parameter λ_k of the added basis function e_k.

$$\begin{aligned}
\frac{\partial \Delta E}{\partial \lambda_k} &= -\frac{\partial}{\partial \lambda_k} \left[(\bar{w}_d S_{da} + \bar{\xi}_a) S_{aa}^{-1} (\bar{w}_d S_{da} + \bar{\xi}_a)^T \right] = \\
&= (\bar{w}_d S_{da} + \bar{\xi}_a) S_{aa}^{-1} \left[\frac{\partial S_{aa}}{\partial \lambda_k} S_{aa}^{-1} (\bar{w}_d S_{da} + \bar{\xi}_a)^T - 2\frac{\partial}{\partial \lambda_k} (\bar{w}_d S_{da} + \bar{\xi}_a)^T \right]
\end{aligned}$$

By remembering (Appendix A) that:

$$\frac{\partial S_{aa}^{-1}}{\partial \lambda_k} = -S_{aa}^{-1} \frac{\partial S_{aa}}{\partial \lambda_k} S_{aa}^{-1}$$

The derivative of ΔE is zero for $(\bar{w}_d S_{da} + \bar{\xi}_a) S_{aa}^{-1} = \bar{0}$ but this is not a minimum, in fact it maximizes the biquadratic term of ΔE.

Adding Without Deleting

In the case the structural change of the network is obtained by adding some basis functions without deleting any $\bar{w}_d S_{da} = 0$ and the condition for the minimization results:

$$\bar{\xi}_a S_{aa}^{-1} \left[\frac{\partial S_{aa}}{\partial \lambda_k} S_{aa}^{-1} \bar{\xi}_a^T - 2\frac{\partial \bar{\xi}_a^T}{\partial \lambda_k} \right] = 0$$

Adding Orthogonal Functions

In the case of adding a set of orthogonal basis functions $S_{aa} = I$ the condition is:

$$(\bar{w}_d S_{da} + \bar{\xi}_a)\frac{\partial}{\partial \lambda_k}(\bar{w}_d S_{da} + \bar{\xi}_a)^T = 0$$

The condition can be written as:

$$\sum_{l=1}^{n_a}\left(\sum_{q=1}^{n_d} w_q s_{ql} + \xi_l\right)\left(\sum_{r=1}^{n_d} w_r \frac{\partial s_{rl}}{\partial \lambda_k} + \frac{\partial \xi_l}{\partial \lambda_k}\right) = 0$$

Only the term of the sum with $l = k$ is not zero.

$$\left(\sum_{q=1}^{n_d} w_q s_{qk} + \xi_k\right)\left(\sum_{r=1}^{n_d} w_r \frac{\partial s_{rk}}{\partial \lambda_k} + \frac{\partial \xi_k}{\partial \lambda_k}\right) = 0$$

The possibility that $\sum_{q=1}^{n_d} w_q s_{qk} + \xi_k = 0$ has already been examined, so it is:

$$\sum_{r=1}^{n_d} w_r \frac{\partial s_{rk}}{\partial \lambda_k} + \frac{\partial \xi_k}{\partial \lambda_k} = 0$$

Remembering the definition of s_{rk} and ξ_k:

$$\int_{\mathcal{R}^n}\left(\sum_{r=1}^{n_d} w_r e_r + \mathcal{E}\right)\frac{\partial e_k}{\partial \lambda_k}\,dx = 0 \tag{B.5}$$

Adding Orthogonal Functions Without Deleting

In the simplest case $S_{aa} = I$ and $w_d S_{da} = 0$ and the minimization condition is:

$$\bar{\xi}_a \frac{\partial \bar{\xi}_a^T}{\partial \lambda_k} = 0$$

hence:

$$\int_{\mathcal{R}^n} \mathcal{E}\frac{\partial e_k}{\partial \lambda_k}\,dx = 0$$

It is interesting to note, that this simple case is actually the case in which only one function is added without changing the parameters of the others.

B.2 Finite Set of Data Error

In the case of interest the function f is unknown. The only thing that we have is a set S of N sample (x_i, y_i) with $i = 1 \ldots N$. In this case instead of the norm $\| \, . \, \|_{\mathcal{R}^n}$ of the functional space we define the functional:

$$\| \, f \, \|_S = \sum_{k=1}^{k=N} f^2(x_k)$$

And a consequent definition of a new superposition function

$$S_S(e_i, e_j) = \sum_{k=1}^{k=N} (e_i(x_k) e_j(x_k))$$

In the identical way in which we have derived the equation B.1 we obtain:

$$\| \, f - \mathcal{N} \, \|_S = \| \, f \, \|_S + \sum_{i,j} w_i w_j S_S(e_i, e_j) - 2 \sum_i w_i \sum_{k=1}^{k=N} f(x_k) * e_i(x_k) \qquad \text{(B.6)}$$

and

$$\begin{aligned} \Delta E &= \| \, \mathcal{N}_d - \mathcal{N}_a \, \|_S + 2 \sum_i \delta(i) w_i \sum_{k=1}^{k=N} \mathcal{E}(x_k) * e_i(x_k) \\ &= \sum_{k=1}^{k=N} (\sum_{i,j} \delta(i)\delta(j) w_i w_j e_i(x_k) e_j(x_k) + 2 \sum_i \delta(i) w_i \mathcal{E}(x_k) * e_i(x_k)) \\ &\quad \sum_{k=1}^{k=N} \Delta E_k \end{aligned}$$

Let us consider the case in which we delete one function e_d and we add two functions e_r and e_l and omitting the dependency from x_k

$$\Delta E_k = (w_d e_d - w_l e_l - w_r e_r)^2 + 2\mathcal{E}(w_d e_d - w_l e_l - w_r e_r) \quad \text{(B.7)}$$

We have to use the equations above for choosing the function e_d that we want to delete and the functions e_l and e_r that we want to add.

We have to search for the minimum of the ΔE

$$\begin{aligned}
\frac{\partial \Delta E_k}{\partial w_l} &= -(w_d e_d - w_l e_l - w_r e_r)e_l - 2\mathcal{E}e_l \\
\frac{\partial \Delta E_k}{\partial w_r} &= -(w_d e_d - w_l e_l - w_r e_r)e_r - 2\mathcal{E}e_r \\
\frac{\partial \Delta E_k}{\partial \lambda_{kl}} &= -(w_d e_d - w_l e_l - w_r e_r)w_l \frac{\partial e_l}{\partial \lambda_{kl}} - 2\mathcal{E}w_l \frac{\partial e_l}{\partial \lambda_{kl}} \\
\frac{\partial \Delta E_k}{\partial \lambda_{kr}} &= -(w_d e_d - w_l e_l - w_r e_r)w_r \frac{\partial e_r}{\partial \lambda_{kr}} - 2\mathcal{E}w_r \frac{\partial e_r}{\partial \lambda_{kr}}
\end{aligned}$$

In the case of the splitting $e_d = e_r + e_l$ and the equation B.7 becomes:

$$\Delta E_k = e_r^2(w_d - w_r)^2 + e_l^2(w_d - w_l)^2 + 2e_r e_l(w_d - w_r)(w_d - w_l) \quad \text{(B.8)}$$

$$+2\mathcal{E}(e_r(w_d - w_r) + e_l(w_d - w_l)) \quad \text{(B.9)}$$

It is easy to show that the equation (B.4) generalizes the pseudoinversion formula obtained in Section 2.1 to the case of structural modifications.

B.3 Gaussian Basis Function

A common choice for the radial basis function is the Gaussian Basis Function:

$$e_i(x) = \gamma_i^{-1} e^{-\frac{1}{2}(x_i - x)Q_i(x_i - x)^T}$$

$$\text{if } \int_{\mathcal{R}^n} e_i \, dx = 1 \text{ then } \gamma_i^2 = (2\pi)^n |Q_i|^{-1}$$

and

$$e_i(x) = \sqrt{\frac{|Q_i|}{(2\pi)^n}} e^{-\frac{1}{2}(x_i - x)Q_i(x_i - x)^T}$$

The parameter of the radial function e_i are the matrix Q_i and the centre x_i:

$$Q_i = \begin{bmatrix} q_{i,11} & q_{i,12} & \dots & q_{i,1n} \\ q_{i,21} & q_{i,22} & \dots & q_{i,2n} \\ \dots & \dots & \dots & \dots \\ q_{i,n1} & q_{i,n2} & \dots & q_{i,nn} \end{bmatrix}$$

$$x_i = \begin{bmatrix} x_{i,1} & x_{i,2} & \dots & x_{i,n} \end{bmatrix}$$

The matrix Q_i can be written as $Q_i = HH^T$ with H non–singular, so that Q_i is a positive definite.

The derivative with respect to the l-th component $x_{i,l}$ of the centre x_i is:

$$\begin{aligned} \frac{\partial e_i}{\partial x_{i,l}} &= e_i \frac{\partial}{\partial x_{i,l}} (-\frac{1}{2}(x_i - x)Q_i(x_i - x)^T) \\ &= -\frac{1}{2} e_i \left(\frac{\partial (x_i - x)}{\partial x_{i,l}} Q_i (x_i - x)^T + (x_i - x) Q_i \frac{\partial (x_i - x)^T}{\partial x_{i,l}} \right) \\ &= -\frac{1}{2} e_i \left(\left(\frac{\partial (x_i - x)}{\partial x_{i,l}} Q_i (x_i - x)^T \right)^T + (x_i - x) Q_i \frac{\partial (x_i - x)^T}{\partial x_{i,l}} \right) \\ &= -\frac{1}{2} e_i 2 (x_i - x) Q_i \frac{\partial (x_i - x)^T}{\partial x_{i,l}} \\ &= e_i (x - x_i) Q_i I_l^T = e_i \sum_r (x_r - x_{i,r}) q_{i,rl} \end{aligned}$$

where I_l is the l-th row vector of the identity matrix I.

The gradient computed with respect to the vector $\bar{x}_i$ results:

$$\frac{\partial e_i}{\partial \bar{x}_i} = e_i(\bar{x} - \bar{x}_i)Q_i \tag{B.10}$$

The partial derivative with respect to the generic element $q_{i,lm}$ of the matrix Q_i is:

$$\begin{aligned}
\frac{\partial e_i}{\partial q_{i,lm}} &= \frac{1}{\sqrt{(2\pi)^n}} e^{-\frac{1}{2}(x_i - x)Q_i(x_i - x)^T} \left(\frac{\partial \sqrt{|Q_i|}}{\partial q_{i,lm}} - \frac{1}{2}\sqrt{|Q_i|}\frac{\partial}{\partial q_{i,lm}}(x_i - x)Q_i(x_i - x)^T \right) \\
&= e_i \left(\frac{1}{|Q_i|}\frac{\partial |Q_i|}{\partial q_{i,lm}} - \frac{1}{2}(x_i - x)\frac{\partial Q_i}{\partial q_{i,lm}}(x_i - x)^T \right) \\
&= e_i \left(\frac{Q_i^{(lm)}}{|Q_i|} - \frac{1}{2}(x_l - x_{i,l})(x_m - x_{i,m}) \right)
\end{aligned}$$

where $Q_i^{(lm)}$ is the algebraic complement of the element $q_{i,lm}$. If $m = l$ the equation above becomes:

$$\frac{\partial e_i}{\partial q_{i,ll}} = e_i \left(\frac{Q_i^{(ll)}}{|Q_i|} - \frac{1}{2}(x_l - x_{i,l})^2 \right)$$

The matrix Q_i is symmetric so that the case $l \neq m$ leads to:

$$\frac{\partial e_i}{\partial q_{i,lm}} = e_i \left(2\frac{Q_i^{(lm)}}{|Q_i|} - (x_l - x_{i,l})(x_m - x_{i,m}) \right)$$

Defining

$$\frac{\partial e_i}{\partial Q_i} = \left(\frac{\partial e_i}{\partial q_{i,lm}} \right)$$

it results:

$$\frac{\partial e_i}{\partial Q_i} = e_i(Q_i^{-1} - \frac{1}{2}(\bar{x} - \bar{x}_i)^T(\bar{x} - \bar{x}_i)) \tag{B.11}$$

B.3.1 Superposition

With Gaussian Basis Functions the scalar product, i.e. the superposition between two basis function is:

$$\begin{aligned} S(e_i, e_j) &= \int_{\mathcal{R}^n} e_i e_j \, dx = \\ &= \gamma_i^{-1}\gamma_j^{-1} \int_{\mathcal{R}^n} e^{-\frac{1}{2}(x_i-x)Q_i(x_i-x)^T - \frac{1}{2}(x_j-x)Q_j(x_j-x)^T} \, dx \\ &= \gamma_i^{-1}\gamma_j^{-1} \int_{\mathcal{R}^n} e^{-\frac{1}{2}A_{ij}} \, dx \end{aligned}$$

with the change of variables $y = x - x_j$ and $w = x_i - x_j$ the exponent can be written as:

$$\begin{aligned} A_{ij} &= (x_i - x)Q_i(x_i - x)^T + (x_j - x)Q_j(x_j - x)^T \\ &= (w - y)Q_i(w - y)^T + yQ_jy^T \end{aligned}$$

With the change of variables $y = zQ_i^{-1}$ and remembering that $(Q_i^{-1})^T = Q_i^{-1}$

$$\begin{aligned} A_{ij} &= (w - zQ_i^{-1})Q_i(w - zQ_i^{-1})^T + zQ_i^{-1}Q_jQ_i^{-1}z^T = \\ &= wQ_iw^T - zIw^T - wIz^T + zQ_i^{-1}Q_iQ_i^{-1}z^T + zQ_i^{-1}Q_jQ_i^{-1}z^T = \\ &= wQ_iw^T - zIw^T - wIz^T + zQ_i^{-1}(Q_i + Q_j)Q_i^{-1}z^T = \end{aligned}$$

Adding and subtracting the term $wQ_i(Q_i + Q_j)^{-1}Q_iw^T$

$$\begin{aligned} A_{ij} &= w(Q_i - Q_i(Q_i + Q_j)^{-1}Q_i)w^T + \\ &\quad (wQ_i(Q_i + Q_j)^{-1}Q_i - z)Q_i^{-1}(Q_i + Q_j)Q_i^{-1}(wQ_i(Q_i + Q_j)^{-1}Q_i - z)^T \end{aligned}$$

Remembering that $Q_i^T = Q_i$ it is easy to show that:

$$Q_i - Q_i(Q_i + Q_j)^{-1}Q_i = Q_iQ_j(Q_i + Q_j)^{-1} \tag{B.12}$$

Considering the changes of variables:

$$\begin{aligned}
\int_{\mathcal{R}^n} e^{-\frac{1}{2}A_{ij}}\,dx &= |Q_i|^{-1} e^{-\frac{1}{2}(x_j-x_i)Q_iQ_j(Q_i+Q_j)^{-1}(x_j-x_i)^T} \\
&\quad \int_{\mathcal{R}^n} e^{-\frac{1}{2}(wQ_i(Q_i+Q_j)^{-1}Q_i-z)Q_i^{-1}(Q_i+Q_j)Q_i^{-1}(wQ_i(Q_i+Q_j)^{-1}Q_i-z)^T}\,dz \\
&= |Q_i|^{-1}\sqrt{(2\pi)^n|Q_i||Q_i+Q_j|^{-1}|Q_i|}e^{-\frac{1}{2}(x_j-x_i)Q_iQ_j(Q_i+Q_j)^{-1}(x_j-x_i)^T} \\
&= \sqrt{(2\pi)^n|Q_i+Q_j|^{-1}}e^{-\frac{1}{2}(x_j-x_i)Q_iQ_j(Q_i+Q_j)^{-1}(x_j-x_i)^T}
\end{aligned}$$

The superposition is now expressed by:

$$\begin{aligned}
S(e_i,e_j) &= \gamma_i^{-1}\gamma_j^{-1}\sqrt{(2\pi)^n|Q_i+Q_j|^{-1}}e^{-\frac{1}{2}(x_j-x_i)Q_iQ_j(Q_i+Q_j)^{-1}(x_j-x_i)^T} \\
&= \sqrt{\frac{|Q_i||Q_j||Q_i+Q_j|^{-1}}{(2\pi)^n}}e^{-\frac{1}{2}(x_j-x_i)Q_iQ_j(Q_i+Q_j)^{-1}(x_j-x_i)^T}
\end{aligned}$$

This result is related to the well-known fact that the class of the gaussian distributions is close under convolution.

We write the derivatives of the superposition with respect to a component $x_{k,l}$ of the centre x_k and a component $q_{k,lm}$ of the matrix Q_k remembering equation B.12.

$$\begin{aligned}
\frac{\partial S(e_i,e_k)}{\partial x_{k,l}} &= S(e_i,e_k)(x_i-x_k)Q_iQ_k(Q_i+Q_k)^{-1}I_l \\
&= S(e_i,e_k)\sum_s\sum_r\sum_p (x_{i,p}-x_{k,p})q_{i,pr}q_{k,rs}\frac{(Q_i+Q_k)^{(ls)}}{|Q_i+Q_k|}
\end{aligned}$$

The gradient is:

$$\frac{\partial S(e_i,e_k)}{\partial x_k} = S(e_i,e_k)(\bar{x}_i-\bar{x}_k)Q_iQ_k(Q_i+Q_k)^{-1} \tag{B.13}$$

$$\begin{aligned}
\frac{\partial S(e_i,e_k)}{\partial q_{k,lm}} &= S(e_i,e_k)\left(\frac{1}{|Q_iQ_k(Q_i+Q_k)^{-1}|}\frac{\partial|Q_iQ_k(Q_i+Q_k)^{-1}|}{\partial q_{k,lm}}\right. \\
&\quad \left.-\frac{1}{2}(x_k-x_i)\frac{\partial}{\partial q_{k,lm}}\left(Q_i-Q_i(Q_i+Q_j)^{-1}Q_i\right)(x_k-x_i)^T\right)
\end{aligned}$$

$$= S(e_i, e_k)\left(\frac{Q_k^{(lm)}}{|Q_k|} + \frac{(Q_i+Q_k)^{(lm)}}{|Q_i+Q_k|} + \frac{1}{2}(x_k - x_i)Q_i\frac{\partial(Q_i+Q_k)^{-1}}{\partial q_{k,lm}}Q_i(x_k-x_i)^T\right)$$

$$(x_k - x_i)Q_i\frac{\partial(Q_i+Q_k)^{-1}}{\partial q_{k,lm}}Q_i(x_k-x_i)^T =$$

$$-(x_k - x_i)Q_i(Q_i+Q_k)^{-1}\frac{\partial Q_k}{\partial q_{k,lm}}(Q_i+Q_k)^{-1}Q_i(x_k-x_i)^T =$$

$$-\frac{1}{|Q_k+Q_i|^2}\sum_r (x_{k,r}-x_{i,r})\sum_s q_{rs}(Q_k+Q_i)^{(sl)}\sum_u (x_{k,u}-x_{i,u})\sum_v q_{uv}(Q_k+Q_i)^{(vm)}$$

B.3.2 Minima of the Error Function

Adding Orthogonal Functions

With a Gaussian radial function the equation B.5 and deriving with respect to each component $x_k^{(i)}$ of the centre vector x_k of the e_k leads to the vectorial equation:

$$\int_{\mathcal{R}^n}\left(\sum_{r=1}^{n_d} w_r e_r + \mathcal{E}\right) e_k(\bar{x}-\bar{x}_k)Q_k\,dx = 0$$

The condition $|Q_k| > 0$ so there is only one solution that leads to:

$$\bar{x}_k = \frac{\int_{\mathcal{R}^n}\left(\sum_{r=1}^{n_d} w_r e_r + \mathcal{E}\right) e_k \bar{x}\,d\bar{x}}{\int_{\mathcal{R}^n}\left(\sum_{r=1}^{n_d} w_r e_r + \mathcal{E}\right) e_k\,d\bar{x}} = \mathcal{H}(\bar{x}_k) \tag{B.14}$$

The second term of the equation depends on $\bar{x}_k$. Let us examine the possibility that the succession $\bar{x}_k^{(n+1)} = \mathcal{H}(\bar{x}_k^{(n)})$ converges to a solution $\bar{\gamma}$.

Adding Orthogonal function without deleting

B.3.3 Factorizable Gaussian Functions

If the gaussians are factorizable that means that the matrixes are expressed by:

$$Q_i = \mathrm{diag}(q_{i,1}, \ldots, q_{i,l}, \ldots, q_{i,n})$$

and the gaussian radial function can be written as:

$$\begin{aligned} e_i(x) &= \gamma_i^{-1} e^{-\frac{1}{2}(x_i - x) Q_i (x_i - x)^T} \\ &= \prod_{l=1}^{n} \gamma_{i,l}^{-1} e^{-\frac{1}{2} q_{i,l} (x_{i,l} - x_l)^2} \\ &= \prod_{l=1}^{n} \frac{1}{\sigma_{i,l}\sqrt{2\pi}} e^{-\frac{1}{2\sigma_{i,l}{}^2}(x_{i,l} - x_l)^2} \end{aligned}$$

where $q_{i,l} = \frac{1}{\sigma_{i,l}{}^2}$ and $\gamma_{i,l} = \frac{1}{\sigma_{i,l}\sqrt{2\pi}}$

The precedent equation becomes

$$\begin{aligned} S(e_i, e_j) &= \int_{\mathcal{R}^n} e_i e_j \, dx = \\ &= \prod_{l=1}^{n} \frac{1}{\sigma_{i,l}\sigma_{j,l} 2\pi} \int_{\mathcal{R}^n} e^{-\frac{1}{2\sigma_{i,l}{}^2}(x_{i,l} - x_l)^2 - \frac{1}{2\sigma_{j,l}{}^2}(x_{j,l} - x_l)^2} \, dx \\ &= \prod_{l=1}^{n} \frac{1}{\sqrt{2\pi}} \frac{1}{\sqrt{\sigma_{i,l}{}^2 + \sigma_{j,l}{}^2}} e^{-\frac{1}{2}\frac{(x_{i,l} - x_{j,l})^2}{\sigma_{i,l}{}^2 + \sigma_{j,l}{}^2}} \end{aligned}$$

Using the result:

$$\begin{aligned} &\int_{\mathcal{R}^n} e^{-\frac{1}{2\sigma_{i,l}{}^2}(x_{i,l} - x_l)^2 - \frac{1}{2\sigma_{j,l}{}^2}(x_{j,l} - x_l)^2} \, dx \\ &= \sigma_{i,l}\sigma_{j,l}\sqrt{\frac{2\pi}{\sigma_{i,l}{}^2 + \sigma_{j,l}{}^2}} e^{-\frac{1}{2}\frac{(x_{i,l} - x_{j,l})^2}{\sigma_{i,l}{}^2 + \sigma_{j,l}{}^2}} \end{aligned}$$

Appendix C

ERROR MEASURES FOR THE APPROXIMATORS

C.1 Errors

There are a lot of proposals of different criteria in statistical literature that lead to the synthesis of different optimal or suboptimal approximators. In this framework is common to define different measures for the precision of the approximation. If f has continuous values and calling $\hat{f}$ the approximation it is possible to define the Mean Square Error (MSE), which is the sum of the variance and squared bias(see [Geman et al., 1992]):

$$MSE\{\hat{f}(x))\} = E[(\hat{f}(x) - f(x))^2] = Var\{\hat{f}(x)\} + Bias^2\{\hat{f}(x)\}$$

where $Bias\{\hat{f}(x)\} = E[\hat{f}(x)] - f(x)$ and $Var\{\hat{f}(x)\} = E[(\hat{f}(x) - E[\hat{f}(x)])^2]$

Another very usual form of error measure, is the L_2 norm, which is called Integrated Squared Error (ISE):

$$ISE\{\hat{f}(x))\} = \int_{\mathcal{R}^n} (\hat{f}(x) - f(x))^2 dx$$

By averaging the ISE over the sets of samples we obtain the Mean Integrated Square Error (MISE):

$$MISE\{\hat{f}(x))\} = E[ISE\{\hat{f}(x))\}] = E\left[\int_{\mathcal{R}^n} (\hat{f}(x) - f(x))^2 dx\right] =$$

$$\int_{\mathcal{R}^n} E[(\hat{f}(x) - f(x))^2] dx = \int_{\mathcal{R}^n} MSE\{\hat{f}(x))\} dx = IMSE\{\hat{f}(x))\}$$

These measures are useful for theoretical reasons, such as proving the consistency of an approximator, i.e the property that the approximator asymptotically approximates the target function with no error. So in practice, it is not possible to use error measures such as MSE, ISE or MISE that are, in fact, more useful for general consideration about the nature and properties of the particular class of approximators that we study.

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

ANN: Artificial Neural Network

BACKPROPAGATION: the gradient descent applied to multilayers networks

DCL: Dynamic Competitive Learning

DRT: Dynamic Regression Tree

F–RBFN: Factorizable RBFN

FEED–FORWARD NETWORK: a network without feedback from the outputs to the inputs.

MLP: Multi–Layer Perceptron

NDEI: Non-Dimensional Error Index. It is defined as the Root Square Error divided by the Standard Deviation

RBFN: Radial Basis Function Network

RECURRENT NETWORK: a network with feedback from the outputs to the inputs

BIBLIOGRAPHY

[Airenti et al., 1993] Airenti, G., Bara, B., and Colombetti, M. (1993). Conversation and behaviour games in the pragmatics of dialogue. *Cognitive Science*, 17:197–253.

[Austin, 1962] Austin, J. A. (1962). *How to Do Things with Words*. Oxford University Press, Oxford.

[Bara et al., 1995] Bara, B., Bucciarelli, M., and Johnson-Laird, P. (1995). Development of syllogistic reasoning. *American Journal of Psychology*, 108(2):157–193.

[Baroglio et al., 1996] Baroglio, C., Giordana, A., Kaiser, M., Nuttin, M., and Piola, R. (1996). Learning controllers for industrial robots. *Machine Learning*, 23:221–249.

[Berenji, 1992] Berenji, H. (1992). Fuzzy logic controllers. In Yager, R. and Zadeh, L., editors, *An Introduction to Fuzzy Logic Applications in Intelligent Systems*. Kluver Academic Publishers.

[Bianchini et al., 1995] Bianchini, M., Frasconi, P., and Gori, M. (1995). Learning without local minima in radial basis function networks. *IEEE Transactions on Neural Networks*, 6(3):749–756.

[Blanzieri, 1998] Blanzieri, E. (1998). In [Quartz and Sejnowski, 1998]. Commentary to the target article (to appear).

[Blanzieri and A.Giordana, 1996] Blanzieri, E. and A.Giordana (1996). An incremental algorithm for learning radial basis function networks. In *Proceedings of the International Conference on Fuzzy Systems*, New Orleans.

[Blanzieri and Bucciarelli, 1996a] Blanzieri, E. and Bucciarelli, M. (1996a). The evaluation of the communicative effect. In *XVIII Cognitive Science Conference*, pages 501–506, San Diego, California.

[Blanzieri and Bucciarelli, 1996b] Blanzieri, E. and Bucciarelli, M. (1996b). Reasoning processes underlying communication: Extracting the rules of the game from a connectionist network. In *Ninth Conference of the European Society for Cognitive Psychology*, Wurzburg.

[Blanzieri et al., 1996] Blanzieri, E., Bucciarelli, M., and Peretti, P. (1996). Modeling human communication. In *1st European Workshop on Cognitive Modeling*, Berlin.

[Blanzieri and Giordana, 1995] Blanzieri, E. and Giordana, A. (1995). Mapping symbolic knowledge into locally receptive field networks. In Gori, M. and Soda, G., editors, *Topics in Artificial Intelligence*, Lectures Notes in Artificial Intelligence. Springer-Verlag.

[Blanzieri and Katenkamp, 1996] Blanzieri, E. and Katenkamp, P. (1996). Learning radial basis function networks on-line. In *13-th International Conference on Machine Learning*, pages 37–45, Bari.

[Botta and Giordana, 1993] Botta, M. and Giordana, A. (1993). SMART+: A multi-strategy learning tool. In *IJCAI-93, Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, volume 2, Chambéry, France.

[Breiman et al., 1984] Breiman, L., Friedman, J., Ohlsen, R., and Stone, C. (1984). *Classification And Regression Trees*. Wadsworth & Brooks, Pacific Grove, CA.

[Broomhead and Lowe, 1988] Broomhead, D. S. and Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355.

[Chen et al., 1991] Chen, S., Cowan, C. F. N., and Grant, P. M. (1991). Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302–309.

[Chen and Chen, 1995] Chen, T. and Chen, H. (1995). Approximation capability to functions of several variables nonlinear functionals and operators by radial basis function neural networks. *IEEE Transactions on Neural Networks*, 6(4):904–910.

[Crowder, 1990] Crowder, R. (1990). Predicting the Mackey-Glass time series with cascade-correlation learning. In D. Touretzky, G. H. and T.Sejnovsky, editors, *Proceedings of the 1990 Connectionist Models Summer School*, pages 117–123. Carnegie Mellon University.

[D.L. Reilly and Elbaum, 1982] D.L. Reilly, L. C. and Elbaum, C. (1982). A neural model for category learning. *Biological Cybernatics*, 45:35–41.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

[Fodor and McMaughlin, 1990] Fodor, J. and McMaughlin, B. P. (1990). Connectionism and the problem of systematicity: why smolensky's solution doen't work. *Cognition*, 35:183–204.

[Fodor and Pylyshyn, 1988] Fodor, J. and Pylyshyn, Z. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28:3–71.

[Frasconi et al., 1996] Frasconi, P., Gori, M., Maggini, M., and Soda, G. (1996). Representation of finite state automata in recurrent radial basis function networks. *Machine Learning*, 23:5.

[Fritzke, 1994a] Fritzke, B. (1994a). Growing cell structure - a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460.

[Fritzke, 1994b] Fritzke, B. (1994b). Growing cells structures - a self–organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460.

[Fritzke, 1995] Fritzke, B. (1995). A growing neural gas network learns topologies. In G.Tesauro, D.S.Touretzky, and T.K.Leen, editors, *Advances in Neural Information Processing Systems*. MIT Press.

[Geman et al., 1992] Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58.

[Girosi et al., 1995] Girosi, F., Jones, M., and Poggio, T. (1995). Regularization theory and neural networks architecture. *Neural Computation*, 7:219–269.

[Giunti, 1996] Giunti, M. (1996). Beyond computationalism. In *XVIII Cognitive Science Conference*, pages 501–506, San Diego, California.

[Haykin, 1994] Haykin, S. (1994). *Neural Networks, a Comprehensive Foundation*. IEEE Computer Society Press.

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feed-forward networks are universal approximators. *Neural Networks*, 2:359–366.

[Jang, 1993] Jang, J. (1993). ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Transactions on Systems, Men and Cybernetics*, SMC-23(3):665–687.

[Johnson-Laird, 1983] Johnson-Laird, P. N. (1983). *Mental Models*. Cambrdige University Press, Cambridge, UK.

[Jones et al., 1990] Jones, R., Lee, Y., Barnes, C., Flake, G., Lee, K., and Lewis, P. (1990). Function approximation and time series prediction with neural networks. In *Proceedings of IEEE International Joint Conference on Neural Networks*, pages I–649–665.

[J.R. Le Gall, 1993] J.R. Le Gall, S. Lemeshow, F. S. (1993). A new simplified acute physiology score (SAPS II) based on a European/North American multicenter study. *JAMA*, 270(24):2957–2963.

[Kadirkamanathan and Niranjan, 1993] Kadirkamanathan, V. and Niranjan, M. (1993). A function estimation approach to sequential learning with neural netoworks. *Neural Computation*, 5:954–975.

[Katenkamp, 1995] Katenkamp, P. (1995). Beispielbasierte Konstruktion von Reglerstrukturen. Master's thesis, Universität Karlsruhe, Fakultät für Informatik, Institut für Prozeßrechentechnik und Robotik.

[Kohonen, 1982] Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, pages 59–69.

[Laird et al., 1987] Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64.

[Levesque, 1988] Levesque, H. J. (1988). Logic and complexity of reasoning. *Journal of Philosophical Logic*, 17:355–389.

[Martinetz, 1993] Martinetz, T. (1993). Competitive hebbian learning rule forms perfectly topology preseving maps. *International Conference on Artificial Neural Networks*, pages 427–434.

[Martinetz and Schulten, 1991] Martinetz, T. and Schulten, K. (1991). A neural-gas network learns topologies. In Kohonen, T., Kisara, K. M., Simula, O., and Kangas, J., editors, *Artificial Neural Networks*, pages 397–402. North Holland, Amsterdam.

[Martinetz and Schulten, 1994] Martinetz, T. and Schulten, K. (1994). Topology representing networks. *Neural Networks*, 7(3):507–522.

[Mathis and Mozer, 1996] Mathis, D. W. and Mozer, N. C. (1996). Conscious and unconscious perception: a computational theory. In *XVIII Cognitive Science Conference*, pages 501–506, San Diego, California.

[McClelland et al., 1986] McClelland, J. L., Rumelhart, D. E., and the PDP research group, editors (1986). *Parallel distributed processing: Explorations in the microstructure of cognition, Volume 2: Psychological and Biological Models.* MIT Press.

[McCulloch and Pitts, 1943] McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

[Millán, 1994] Millán, J. (1994). Learning efficient reactive behavioral sequences from basic reflexes in a goal-directed autonomous robot. In *Proceedings of the third International Conference on Simulation of Adaptive Behavior.*

[Mitchell, 1997] Mitchell, T. (1997). *Machine Learning.* McGraw-Hill.

[Moody and Darken, 1988] Moody, J. and Darken, C. (1988). Learning with localized receptive fields. In Sejnowski, T., Touretzky, D., and Hinton, G., editors, *Connectionist Models Summer School*, Carnegie Mellon University.

[Moody and Darken, 1989] Moody, J. and Darken, C. (1989). Fast learning in networks of locally tuned units. *Neural Computations*, 1(2):281–294.

[Orr, 1995] Orr, M. J. L. (1995). Regularization in the selection of radial basis function centers. *Neural Computation*, 7(3):606–623.

[Park and Sandberg, 1993] Park, J. and Sandberg, I. W. (1993). Approximation and radial-basis-function networks. *Neural Computation*, 5(3):305–316.

[Park and Sandberg, 1991] Park, J. and Sandberg, W. (1991). Universal approximation using radial-basis functions. *Neural Computation*, 3:246–257.

[Platt, 1991] Platt, J. (1991). A resource-allocating network for function interpolation. *Neural Computation*, 3:213–225.

[Plaut and Shallice, 1993] Plaut, D. C. and Shallice, T. (1993). Deep dyslexia: a case study of connectionist neuropsychology. *Cognitive Neuropsychology*, 10(5):377–500.

[Poggio and Girosi, 1990] Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497.

[Quartz and Sejnowski, 1998] Quartz, S. R. and Sejnowski, T. J. (1998). The neural basis of cognitive development: a costructivist manifesto. *Behavioural and Brain Science*. (To appear).

[Quinlan, 1979] Quinlan, J. (1979). Induction over large data bases. Technical Report HPP-79-14, Heuristic Programming Project, Stanford University.

[Rosenblatt, 1962] Rosenblatt, F. (1962). *Principles of Neurodynamics*. Spartan Book, New York.

[Rumelhart and McClelland, 1986] Rumelhart, D. E. and McClelland, J. L. (1986). On learning the past tense of english verbs. In [McClelland et al., 1986].

[Rumelhart et al., 1986] Rumelhart, D. E., McClelland, J. L., and the PDP research group., editors (1986). *Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundations*. MIT Press.

[Sammut et al., 1992] Sammut, C., Hurst, S., Kedzier, D., and Michie, D. (1992). Learning to fly. In Sleeman, D. and Edwards, P., editors, *Machine Learning - Proceedings of the Ninth International Workshop (ML92)*, pages 385–393. Morgan Kaufmann.

[Sanger, 1991] Sanger, T. (1991). A tree-structured adaptive network for function approximate in high-dimensional spaces. *IEEE Transactions on Neural Networks*, 2(2):285–293.

[Scott, 1992] Scott, D. W. (1992). *Multivariate Density Estimation*. Wiley.

[Searle, 1969] Searle, J. R. (1969). *Speech Acts*. Cambridge University Press, Cambridge.

[Searle, 1979] Searle, J. R. (1979). *Expressions and Meaning*. Cambridge University Press, Cambridge.

[Smolensky, 1988] Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11:1–74.

[Specht, 1988] Specht, D. (1988). Probabilistic neural networks for classification mapping, or associative memory. In *IEEE International Conference on Neural Networks*, volume 1, pages 525–532.

[Specht, 1990] Specht, D. (1990). Probabilistic neural networks. *Neural Networks*, 3:109–118.

[Sperber and Wilson, 1986] Sperber, D. and Wilson, D. (1986). *Relevance. Communication and Cognition*. Harvard University Press, Cambridge, Mass.

[Towell and Shavlik, 1994] Towell, G. and Shavlik, J. (1994). Knowledge based artificial neural networks. *Artficial Intelligence*, 70(4):119–166.

[Tresp et al., 1993] Tresp, V., Hollatz, J., and Ahmad, S. (1993). Network structuring and training using rule-based knowledge. In *Advances in Neural Information Processing Systems 5 (NIPS-5)*.

[Tresp et al., 1997] Tresp, V., Hollatz, J., and Ahmad, S. (1997). Representing probabilistic rules with networks of gaussian basis functions. *Machine Learning*, 27:173—200.

[Utgoff, 1988] Utgoff, P. (1988). ID5: an incremental ID3. In *Proceedings of the 5th Machine Learning Conference MLC-88*, pages 107–120, Ann Arbor, MI.

[Whitehead and Choate, 1994] Whitehead, B. A. and Choate, T. D. (1994). Evolving space-filling curves to distribute radial basis functions over an input space. *IEEE Transactions on Neural Networks*, 5(1):15–23.

[Xu et al., 1994] Xu, L., Kryzak, A., and Yuille, A. (1994). On radial basis function nets and kernel regression: statistical consistency, convergence rates, and receptive field sizes. *Neural Networks*, 7:609–628.

[Zadeh, 1965] Zadeh, L. (1965). Fuzzy sets. *Information Control*, 8:338–353.