A LOCAL MODEL NETWORK APPROACH TO NONLINEAR MODELLING

A dissertation submitted to
the Department of Computer Science
of the University of Strathclyde
in partial fulfilment of the requirements
for the degree of
Doctor of Philosophy

By
Roderick Murray-Smith
November, 1994

Abstract

This thesis describes practical learning systems able to model unknown nonlinear dynamic processes from their observed input-output behaviour. Local Model Networks use a number of simple, locally accurate models to represent a globally complex process, and provide a powerful, flexible framework for the integration of different model structures and learning algorithms.

A major difficulty with Local Model Nets is the optimisation of the model structure. A novel Multi-Resolution Constructive (MRC) structure identification algorithm for local model networks is developed. The algorithm gradually adds to the model structure by searching for 'complexity' at ever decreasing scales of 'locality'. Reliable error estimates are useful during development and use of models. New methods are described which use the local basis function structure to provide interpolated state-dependent estimates of model accuracy. Active learning methods which automatically construct a training set for a given Local Model structure are developed, letting the training set grow in step with the model structure – the learning system 'explores' its data set looking for useful information.

Local Learning methods developed in this work are explicitly linked to the local nature of the basis functions and provide a more computationally efficient method, more interpretable models and, due to the poor conditioning of the parameter estimation problem, often lead to an improvement in generalisation, compared to global optimisation methods. Important side-effects of normalisation of the basis functions are examined.

A new hierarchical extension of Local Model Nets is presented: the Learning Hierarchy of Models (LHM), where local models can be sub-networks, leading to a tree-like hierarchy of softly interpolated local models. Constructive model structure identification algorithms are described, and the advantages of hierarchical 'divide-and-conquer' methods for modelling, especially in high dimensional spaces are discussed.

The structures and algorithms are illustrated using several synthetic examples of nonlinear multivariable systems (dynamic and static), and applied to real world examples. Two nonlinear dynamic applications are described: predicting the strip thickness in an aluminium rolling mill from observed process data, and modelling robot actuator nonlinearities from measured data. The Local Model Nets reliably constructed models which provided the best results to date on the Rolling Mill application.

Acknowledgements

On the technical side, I would like to thank my employers Daimler-Benz AG for providing a stimulating research environment, and for letting me include research carried out for them in the thesis. I'd especially like to thank Frieder Lohnert for giving me the freedom to explore the areas I thought useful.

My other colleagues provided a great deal of advice and support, directly and indirectly: Jeff Donne for his insight into rolling mills (and cocktails) and for preparing the data for the mill modelling work; Henrik Gollee for his assistance in the vagaries of MATLAB, performing experiments and proof-reading the thesis; Ken Hunt for reading the thesis and trying to improve my mathematical precision; Dietmar Neumerkel for his organisational support in the mill work and for helping with the development of tools and ideas; Robert Shorten for co-operation with the normalisation work and for doing the dishes. Thanks to the other members of the group not directly involved in this work. Wise old Greeks, Indians and Persians also managed to put the ups and downs of research into perspective.

Cooperating in research has fewer geographical limitations than ever before, so I'd like to thank Tor Arne Johansen in Norway and Daniel Sbarbaro-Hofer in Chile for the many long distance E-mail discussions we had (thanks again to Dan for the help given at the start of the work, for reading the resulting thesis and for making insightful suggestions). I'd also like to thank Tom Kavli in Oslo for letting me use his robot actuator data. My supervisor Douglas McGregor was also, unusually for a supervisor, one of the long distance colleagues, but still managed to provide guidance in the right direction at the right time.

On the personal side, I thank my parents again for giving me the best start in life I could have hoped for.

I would like to thank history for making Berlin a fascinating place in the early 90's, and the Berliners (*Ossis*, *Wessis* and *Ausländer*) for being a lot of fun to be with.

Contents

List of Tables

List of Figures

Chapter 1

Learning Systems for Empirical Modelling

1.1 Learning

The ability to interact with the environment and to learn from the effects of these interactions is one of the defining features of intelligence. A system with the ability to learn from observation thus compensates for an initial lack of *a priori* knowledge about its given task. Such flexibility is becoming increasingly important in automatic systems, as the physical, technological and economical environments in which systems are operating are changing faster than ever before. Higher levels of autonomous action are desired from our robots, washing machines, cars and computers. Improved flexibility and adaptability is a major asset, whether the adaptability is in the product development process, or in the products themselves.

This thesis targets the task of modelling complex nonlinear, dynamic processes by allowing models to learn from the processes' observed behaviour. A goal of the work was to develop methods which not only had the required performance, but were also relatively interpretable, to support validation of models, and able to integrate models and methods from other paradigms. The introduction of explicit *a priori* knowledge about the target process is also an important element of applied learning systems.

Obtaining an accurate computer-based model of the physical process is the first step towards the creation of high performance diagnosis systems, supervisory systems, controllers and filters. In many practical systems, however, the process is still poorly understood, or new variations of the system are being constantly created, leading to a slightly different problem each time a controller is to be developed. The ability to use learning systems to cope with the uncertainties would allow developers to produce high performance systems faster and more cheaply than with conventional techniques.

The intuitive concept of 'learning' can be interpreted in a number of ways when trying to emulate it on a machine:

A typical dictionary definition of learning is:

- 1. to gain knowledge of something or acquire skill in some art or practice, 2. to commit to memory, 3. to gain by experience, example etc., 4. to become informed.

Other definitions from the researchers investigating machine learning include:

- Learning systems belong to the class of systems which show a gradual improvement of performance due to the improvement of the estimated unknown information (Fu, 1970).
- Learning is optimisation under conditions of insufficient *a priori* information (Tsypkin, 1971).
- Learning is the process by which one entity acquires knowledge (Rich, 1988).
- Learning can be regarded as synthesising an approximation of a multi-dimensional function, that is solving the problem of hypersurface reconstruction (Poggio and Girosi, 1990).
- ...modifying patterns of behaviour on the basis of past experience so as to achieve specific anti-entropic ends. In these higher forms of communicative organisms the environment, considered as the past experience of the individual, can modify the pattern of behaviour into one which in some sense or other will deal more effectively with the future environment (Wiener, 1948).
- Behaviour is primarily adaptation to the environment under sensory guidance. It takes the organism away from harmful events and toward favourable ones, or introduces changes in the immediate environment that make survival more likely (Hebb, 1949).

This work develops learning algorithms and model structures which gain knowledge about a process from observed input-output example by synthesising a suitable non-linear multi-dimensional function to fit the training data.

1.2 Learning & Engineering – Where is the Engineering?

Much research in recent years has been carried out within the artificial neural network paradigm, using simplified formal models of physiological systems. The neural network research in empirical modelling has been very experimentally oriented, so the learning algorithms and structures

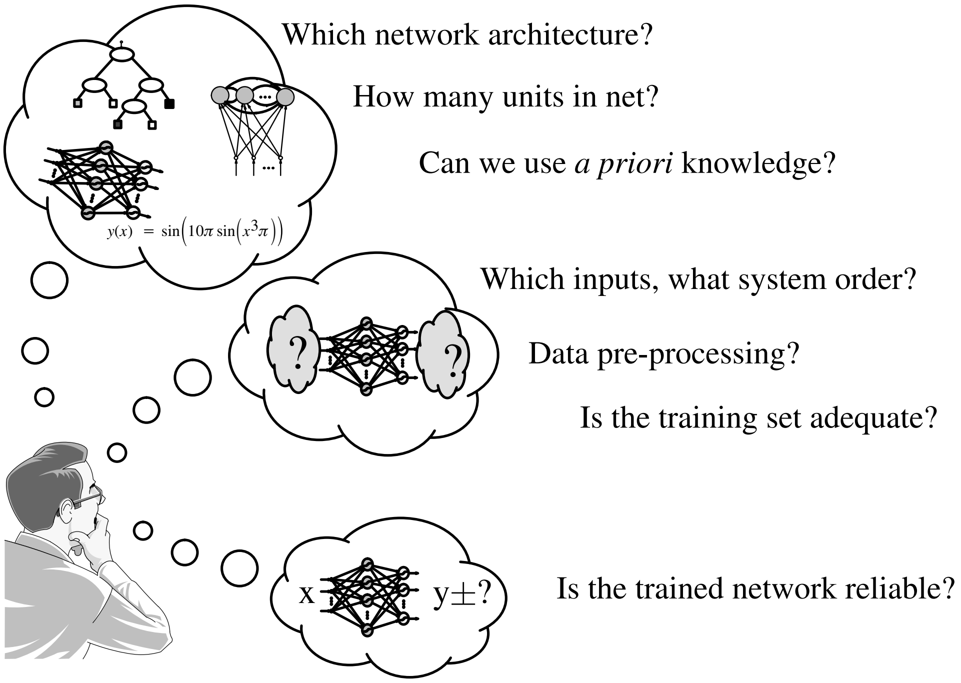


Figure 1.1: The trouble with neural nets for engineering ... An engineer faces a number of difficult design decisions when using neural networks in practical projects.

which have become popular in this community have been successfully applied to a variety of challenging applications. However, few workers in the area have analysed the deeper theoretical issues, or exploited results and experience from closely related fields such as *function approximation*, *system identification* or *statistics*. This ignores decades of relevant work, and has made the scientific output less accessible to a broader community, leading to criticism of the neural network area by scientists and engineers from other fields.

The lack of a theoretical approach to the work has, unfortunately, also had the consequence that there is no clearly defined engineering process in which a model can reliably be created from measurements taken from a physical system, as shown in Figure 1.1. The impressive results quoted in the literature usually come after months of 'tweaking' the parameters of learning algorithms, implicitly using *a priori* knowledge by pre-processing the data, and in selective testing of the trained system. The length of time taken for training is often prohibitive, often without the guarantee of an optimal solution.

The difficulty in determining whether a good solution has been found stems from the fact that most networks need to have their structure initialised in advance using guesswork, with little guarantee against over-fitting the data, or under-specifying the model structure. It is vital for the successful application of the ideas, that robust (in terms of reliably finding a good model which generalises well to new inputs) learning algorithms be developed which can adaptively find a parsimonious model structure and optimise its parameters to fit a given problem and training set.

The problems for learning systems in industry are not, however, limited to poor learning algorithms. The creation of a training set for a nonlinear, multi-variable dynamic system can be an extremely complex task. The data acquisition process may also be very expensive, time-consuming, costly, and in some cases dangerous (if you don't already have a good controller) and time-consuming. It is also possible to unwittingly include undesired side-effects in the data specific to a particular run/day/setting which limit the usefulness of the final model.

A further problem with much of the current reported research is that the methods used to evaluate performance of the trained system are usually too simplistic. The reliability of the resulting model is also usually not clearly understood, partially because of the poor interpretability of the model architectures. In conventional neural networks it is often difficult to introduce *a priori* knowledge. This is highly important in practical applications, where there is usually a great deal of such knowledge available about the system in question. In most publications, the creation of the training set is implicitly affected by *a priori* knowledge about the system, and certain architectures may be more suited to certain system types. The assumption underlying the research here is that it is very important that prior knowledge can be explicitly built in to the system's architecture and optimisation algorithms. Methods to do this have been described, but are still not present in much of the current research. This work attempts to bring these various facets together within a single framework.

1.3 Thesis Contributions

The methods described in this thesis have been chosen for their suitability for integration with conventional engineering techniques, as well as their powerful representations and learning algorithms. The resulting interpretability and robustness with respect to sparse or noisy data was also an important aspect of the work.

- The Local Model Network, an existing generalisation of Basis Function[1] networks is analysed, and the theoretical and practical suitability of the architecture for practical modelling applications is demonstrated. The Local Model framework allows the integration of *a priori* knowledge, is more transparent than other architectures, and by pre-structuring the model structure, can better cope with high-dimensional, or high order processes.
- It was found that the commonly used global parameter optimisation in Local Model networks is computationally expensive, and in some cases poorly conditioned. A new Local Learning algorithm for the optimisation of the parameters in a local model net has been developed. This is significantly faster, produces more interpretable models and can

[1] Networks consisting of a single nonlinear layer, linearly weighted to the output, as described in Section 2.3.1.

have a regularisation effect on the optimisation process producing models with a better generalisation ability.

- Locally interpolated error estimates for Basis Function and Local Model networks are developed. These produce state-dependent error estimates for a trained local model network, which help validate the trained model, and can be used by constructive algorithms, or on-line when the model is in use.

- The effect of normalisation of the basis functions in Local model and Basis Function networks is analysed. Side-effects are described which reduce the interpretability, and have serious effects on the smoothness and robustness of the final model.

- The new Multi-Resolution Constructive (MRC) structure identification algorithm for local model nets is developed. This automatically fits the model structure (number, location and size of the basis functions, and the complexity of the local models) to the available training data. The algorithm uses a 'complexity heuristic' to gradually improve the model's structure.

- The local model network is extended to a hierarchy of local model networks – the Learning Hierarchy of Models (LHM) architecture. Parameter optimisation and structure identification algorithms are described which utilise the hierarchical nature of the structure to make learning more efficient.

- The algorithms and model structures are applied to model data from a real industrial process – an aluminium rolling mill. The algorithms performed better than competing learning systems such as MARS (Friedman, 1991), and Multi-Layer Perceptrons[2], and it is intended to implement the model in on-line tests. A further example, modelling robot actuator dynamics is also analysed.

1.4 Thesis Structure

- Chapter 2 is a review of the existing empirical modelling theory. The empirical modelling process is introduced and the importance of the various stages discussed: *Experiment design*, *construction of a suitable representation*, *optimisation of the parameters* and *validation of the trained models* are all vital stages, and the lack of support for these phases from conventional neural networks is criticised. The Local Model network architecture is reviewed and its suitability as a network for practical modelling applications discussed, as is the use of hierarchical learning structures. An overview of related work and theory is given.

[2] See Section 2.1.4.

- Chapter 3 investigates several aspects of local model networks. The problems with ill-conditioning of the parameter estimation problem in local model nets are analysed and a new local learning algorithm is described which is faster and in noisy or sparsely populated problems can be more robust than global methods. The trade-off between global and local training methods is discussed.

  A flexible, straightforward extension to local model nets is given which allows interpolation of local estimates of accuracy to provide a state-dependent error statistic for a trained network.

  The effect of normalisation of the basis functions on the network's representational properties is analysed. Several side-effects other than the desired partition of unity are described which can have serious consequences for the interpretability and robustness of basis function networks.

- In Chapter 4 the new MRC algorithm for constructive creation of Local Model Basis Function Nets is described. This development iteratively allocates units to the areas of greatest complexity in the system. This is repeated at ever increasing resolution of complexity, gradually creating an ever more accurate model, given the limitations of the available training data.

  Constructive techniques for active selection of the most important training data from a large training set are given. This allows the learning system to maintain the training set at the size and completeness needed for a given model structure. The strengths and weaknesses of the methods developed in this chapter are illustrated using several artificial static and dynamic modelling tasks.

- Chapter 5 introduces the Learning Hierarchy of Models architecture. This can be viewed as a hierarchical local model net structure. The advantage of this structure is that more efficient models can be produced, and that the hierarchical nature of the structure allows more efficient learning algorithms, especially for high-dimensional problems. Parameter optimisation algorithms and structure identification algorithms are defined. The confidence estimation and active learning components described in previous chapters for local model nets are extended to the LHM architecture.

- Chapter 6 demonstrates the practical application of the new methods with the modelling aspects of an aluminium rolling mill. The local model and LHM architectures were successfully applied to model the output thickness deviation of the strip, given the current state of the system.

  A further example, the modelling of robot actuator nonlinearities, is given. The results are compared with three other methods, MARS, ASMOD (Kavli, 1992) and LSA (Johansen and Foss, 1994b).

- Chapter 7 summarises and discusses the significance of the results and analysis in the thesis.

Chapter 2

Methods for training models

The scope of the use of learning techniques for empirical modelling in this thesis is outlined, and the classes of learning systems are described in general terms. The various research paradigms associated with machine learning are reviewed, with special attention paid to the practical problems with artificial neural networks in an engineering environment. To clarify the requirements of a learning framework for modelling, the modelling process is analysed, each phase of which is then associated with desirable features in a learning system. The Local Model Network, a generalisation of Basis Function nets, is proposed as an architecture suited to real applications. It provides the ability to introduce a priori knowledge and model complex systems robustly, while allowing estimates of accuracy and enhancing the training set. The relevant literature is reviewed and the close connections to conventional statistical methods, system identification and fuzzy logic are emphasised.

2.1 Using Learning Techniques for Empirical Modelling

The 'modelling problem' as discussed here is to try to robustly approximate the behaviour of a given complex system from observation data, where complex implies that the system can be non-linear, time-invariant, multi-variable and dynamic. This can be interpreted as a learning task, where the learning system has to learn a suitable representation for the process in question. The model should obviously be a good representation of the target system, for the purposes intended of it, but in an industrial or scientific environment it should ideally also be interpretable, so that the engineer gains improved understanding about the system. The learning systems used in this thesis can therefore be viewed as computationally intensive tools which are used to support the modelling process by attempting to induce a parsimonious representation of the process from the behaviour described by the observed input-output data. The basic assumption underlying the use of learning systems for modelling purposes is therefore that the behaviour of the process can be described in terms of its observed inputs (ψ) and outputs (y). The process can therefore be modelled as a function $f(\psi)$ of the inputs ψ, i.e. $\hat{y} = f(\psi)$, subject to a measurement error e, so that the true outputs are

$$y = f(\psi) + e. \tag{2.1}$$

Such methods are basically black-box modelling techniques, i.e. techniques which attempt to describe a system by finding relationships between the system's inputs, internal states and outputs using general model structures ($f(\psi)$ performs a nonlinear mapping from n dimensional inputs to m dimensional outputs, $\mathcal{R}^n \rightarrow \mathcal{R}^m$) to represent a fit to the given data, irrespective of physical meaning of the parameters.

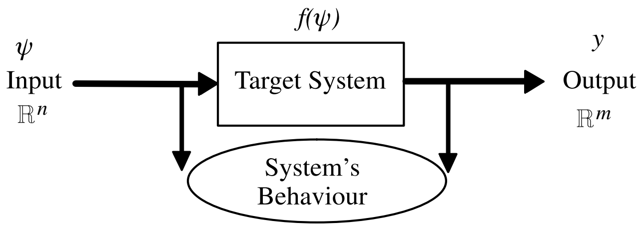


Figure 2.1: Systems can often be described by their input-output behaviour

In practice it will usually be impossible to find an absolutely correct representation of the underlying process because of the effects of unmeasured inputs and states, which can be generalised to be treated as noise on the data, leading to the need to use a stochastic framework for the modelling process. It will also be rare for the training data to be uniformly distributed around the input space, and there may be areas with insufficient data available for a good approximation. This uncertainty means that there is no general method which can be applied to all modelling problems. To reduce the effect of the uncertainty in the data, constraints on the form of the possible solutions must be used. Such constraints are basically any existing knowledge about the process, or its environment. The framework should therefore be able to include such knowledge wherever possible. If *a priori* knowledge of model structures or parameters is included in the estimation process, the model can be called a *grey-box* model. The task for the tools developed in this thesis is therefore to support the process

$$\text{Observed Data} + \textit{A priori} \text{ Knowledge} \rightarrow \text{Model}$$

as well as possible.

2.1.1 Empirical models of dynamic systems

Any real physical system's reactions to a given input are not likely to be instantaneous, and will depend on previous inputs. We are therefore dealing with dynamic systems, which are processes described by difference or differential equations, where the current output of the system depends not only on the current external stimuli, but also on previous stimuli and internal states. This brings a new dimension to the modelling process, as the optimal sampling rate of the unknown system must be estimated, the order of the system must be taken into account, and the model's dynamics must be validated. To represent the dynamic aspects of the system, it is necessary to have memory elements to store past inputs and model states which are passed to a static nonlinear model.

Most of the work in this thesis is based on the assumption that the process can be represented by a non-linear auto-regressive model with exogenous inputs (NARX) over the whole operation envelope. As the implementation of the learning system will be on a digital computer, we consider discrete-time non-linear systems having the general form

$$y(t) = f(y(t-1), \ldots y(t-n_y), \boldsymbol{u}(t-k), \ldots \boldsymbol{u}(t-k-n_u)) + e(t). \tag{2.2}$$

Here, $y(t)$ is the system output, and $\boldsymbol{u}(t)$ the input. The analysis is limited to multiple-input single-output (MISO) systems so that $y(t) \in Y \subset \mathcal{R}$ and $\boldsymbol{u}(t) \in U \subset \mathcal{R}^{n_{in}}$. Here k represents a time delay. The *information vector* passed to the nonlinear model is therefore defined as

$$\boldsymbol{\psi}(t-1) = [y(t-1), \ldots y(t-n_y), \boldsymbol{u}(t-k), \ldots \boldsymbol{u}(t-k-n_u)]^T \tag{2.3}$$

or, if a zero-mean disturbance term $e(t)$ is to be taken into account on-line, the widely studied NARMAX (Non-linear ARMAX) framework (Leontaritis and Billings, 1985) from system identification can be used,

$$\boldsymbol{\psi}(t-1) = [y(t-1), \ldots y(t-n_y), \boldsymbol{u}(t-k), \ldots \boldsymbol{u}(t-k-n_u), e(t-1), \ldots e(t-n_e)]^T, \tag{2.4}$$

where $e(t) \in E \subset \mathcal{R}$.

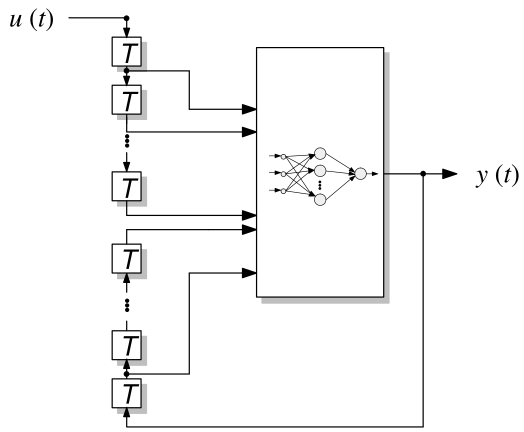


Figure 2.2: Using a tapped delay line (the T's represent delay elements) and static neural net to represent nonlinear dynamic systems.

The NARX, or tapped delay line approach (see the graphical representation of equation (2.2) in Figure 2.2) is fairly pragmatic, as it brings the neural network into the world of conventional system identification theory, where the static neural net can be viewed simply as a technique for function approximation. One disadvantage is that the input dimension becomes very large for even simple systems. Also, if the sampling rate has been correctly set, the data from a dynamic system will be highly correlated on the delayed inputs from the output state – i.e. it is impossible for the data to fill the input space, as shown in Figure 2.3, which shows a slice through the input space of the rolling mill data discussed in Chapter 6. This can

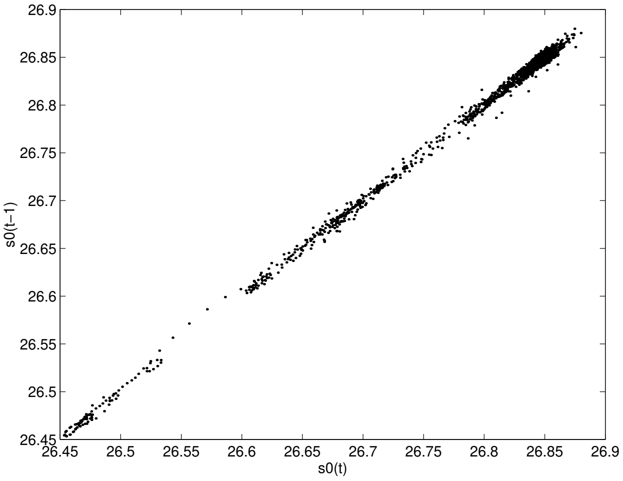


Figure 2.3: A slice through the input space of the rolling mill's training set, showing a variable plotted against the delayed version of itself.

have serious consequences for some learning algorithms, and model structures. If the model structure is such that the nonlinearity results from partitions which are orthogonal to the axes of the input space (an underlying assumption in learning systems such as MARS, ASMOD, ABBMOD, ID3), the model will be less suited to modelling such dynamic processes than an algorithm which can partition the input space more freely, e.g. by placing basis functions on data points from the training set, or by allowing axis-oblique partitions of the input space.

An alternative to an external tapped delay line is to have memory and feedback within the network itself, where the dynamics are learned by the network. Such recurrent networks, which contain local internal feedback connections have received a great deal of attention in the literature (see, for example (Hopfield, 1984, Żbikowski, 1994)). These networks are mathematically elegant structures which are, however, often more difficult to understand and train than static networks for practical problems. The local model network mixed order systems described in Section 2.5.4 are less susceptible to the dimensionality problems inherent to the NARX representation.

A further alternative to the NARX representation is to pass the derivatives of the variables to the model, which makes the data distribution more even and reduces the conditioning problems often caused by the highly correlated inputs found in the training data for NARX systems. Because of the sensitivity of such systems to noise, it is important to filter the data beforehand.

2.1.2 Classes of learning systems

As can be seen from the wide variety of definitions in Section 1.1, machine learning can be studied from a number of widely differing viewpoints. This is largely because the use of observed data to provide the information needed to better fulfil a given goal, whether on-line, or off-line, is a major aspect of almost all areas of science and engineering. This makes it important to try and describe the basic features of learning systems in as general a form as possible, so that the various branches of research can be more easily integrated. The systems shown in Figures 2.4-2.6 describe the range of learning ability for automatic model creation.

Systems which cannot learn

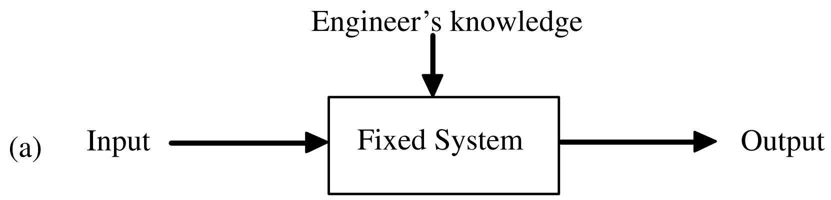


Figure 2.4: System without learning ability – behaviour is pre-programmed

System (a) is the traditional method of solving a problem. The problem is analysed, decomposed into subsystems, and an explicit solution is developed by human engineers using their knowledge of the problem, its constraints and its environment (note that much of this knowledge is however based on earlier empirical work). The resulting program/expert system/controller/classifier is then tested with experimental data and put into use. *No automatic learning is used* in the development, making the development process sensitive to changes in the environment, the system or the development goals.

Systems which can be trained

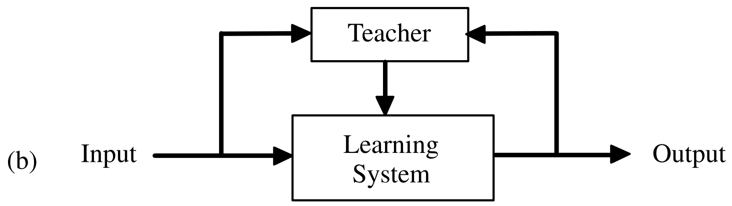


Figure 2.5: Supervised learning system

System (b) is a *supervised learning* system. The machine's task is therefore described by the 'teacher' using examples of what should be done, rather than instructions about how to do it, and criteria for the evaluation of the learning system's performance, compared to the output the 'teacher' expected. The learning system then learns the optimal mapping from input to output with the help of an external 'teacher'. The task of adjusting the learning system's parameters and structure to achieve this is not a trivial matter. For any learning structure the optimisation task usually becomes more difficult, the more flexible the representation used.[1]

Systems which can self-organise

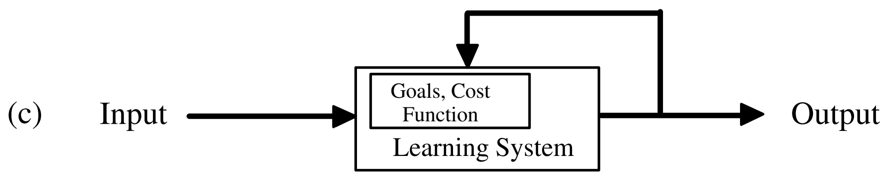


Figure 2.6: Self-organising system

System (c) is an *unsupervised learning* or *self-organising* system. There is no external teacher and the system adjusts its parameters and structure to optimise some predetermined cost function without instructive feedback from an external body. This very general description of an unsupervised system is extremely far-reaching, and should not be confused with the limited implementation of the current generation of unsupervised learning algorithms described in the literature, e.g. the *Self-Organising Maps* (Kohonen, 1990), which are basically clustering algorithms. The important feature of self-organising systems is that the goals and cost functions are built into the system, allowing them to autonomously adapt their behaviour to better achieve their given goals, rather than learning to imitate an existing solution.

The unsupervised learning system can provide a more general form of learning system, depending on the complexity of implementation. The teacher has, in effect, been 'hard-wired' into the learning algorithm in the form of goals and a quality functional which is dependent on the inputs and resulting outputs, taking system constraints into account. The methods described in this thesis are not directed towards the direct implementation of Self-organising learning systems, but will involve some of the self-organising principles to provide the structure for the solutions within the supervised learning schemes, so the desired output is described explicitly in the training set, although the structure of the learning system is identified from the training data in an unsupervised manner – i.e. the 'teacher' does not have to tell the learning system exactly what its structure should be and which training data it should use to train its parameters.

[1] A more abstract form of this style of learning can be seen in *reinforcement learning* systems, where the system is no longer told exactly what to do, but is given graded feedback about the system's performance.

2.1.3 Research paradigms for automatic empirical modelling

Cybernetics, control and adaptive systems

The idea of artificial systems capable of learning is not as new as some people imagine. Many of the basic ideas present in modern research have been in print since the 1940's, when Norbert Wiener initiated the field of *Cybernetics* (Wiener, 1948) – a field which, like neural networks, brought together control engineers, biologists, mathematicians, sociologists and computer scientists. The concepts of learning systems, especially with ideas from biological systems, were examined throughout the world, the popularity of the field shown in review papers such as (Sklansky, 1966) and (Fu, 1970). Steinbuch's work was often ahead of its time (note the rolling mill application of neural nets in (Steinbuch, 1963)!), while Tsypkin's work (Tsypkin, 1971, Tsypkin, 1973) still clarifies many of the basics involved and describes the application of learning and adaptation to a variety of technical systems.

Although cybernetics, like neural networks, is still seen as an elegant framework for promoting communication and interaction between various fields of research, it could not stand the strain brought by the explosion in progress and knowledge in its various sub-fields, which lead to the specialisation and more insular behaviour now common in the fields originating from cybernetics.

System identification (Ljung, 1987) is the area of modern control theory with the closest similarity to the learning systems philosophy described in this thesis, although the majority of the identification work has been based on the basic assumption of linear representations of the systems. Important contributions to the area of non-linear system identification can be found in the NARMAX modelling methods (Billings, 1980, Chen and Billings, 1989), based first on polynomial representations, then later on neural network implementations with multi-layer perceptrons and RBF (Radial Basis Function) Networks (Chen et al., 1990, Chen and Billings, 1992).

On-line adaptation of the model can be used to cope with slow variations in the parameters of a given system, or of a change in operating point. This adaptation allowed the use of linear models for the control of non-linear systems and environments, and was also a significant, if restricted, step towards learning systems. Adaptive signal processing is covered in (Haykin, 1991) and a review of the adaptive control field is given in (Åström, 1987), where he defines adaptive control as: *a special type of nonlinear feedback control, where the states of the process can be separated into two categories, which change at different rates. The slowly changing states are viewed as parameters.*

Most of the practical industrial applications of adaptive control systems have, however, been limited to self-tuning control, where linear controllers are adjusted automatically for a given system. In many cases, an accurate time-invariant nonlinear model of a system will remove the need for on-line adaptation.

Statistics

Many of the tasks routinely ascribed to learning systems (i.e. classification or modelling ability derived from data, as opposed to *a priori* knowledge about the problem) can be viewed as regression problems, and many of the tools for their solution have thus been a standard part of the statistician's toolkit for decades. (Barron and Barron, 1988) discusses the similarities between statistical methods and more recent developments. Fisher's simple linear discriminant algorithm is functionally similar to the Perceptron (Fisher, 1936), and has led to the more powerful quadratic and polynomial discriminant algorithms. Nearest-Neighbour algorithms are also powerful techniques, despite their simplicity (Dasarathy, 1990, Duda and Hart, 1973). Spline and Kernel based modelling methods have much in common with Basis Function Nets (to be described in Section 2.3.1) (Wahba, 1990, Wahba, 1992, Kavli, 1992) and local approximation ideas (to be described in Section 2.3.2) have also been used in statistics. The statisticians also developed tree based regression systems independently from the AI world (e.g. CART (Breiman et al., 1984) and MARS (Friedman, 1991)). A further important contribution from statistics is the wealth of evaluation techniques developed to validate the models and classifiers derived from observed data (e.g. cross-validation techniques, robust cost functions, sampling techniques, additive models etc.). *Projection pursuit* methods are also closely related to Multi-Layer Perceptron neural networks (Zhao, 1992).

Machine learning – the symbolic view

Non-neural machine learning has also a relatively long history, dating back to Samuel's checkers work (Samuel, 1959, Samuel, 1967) and although the neural network research faded away at the end of the 1960's, the Artificial Intelligence community kept working on learning systems. Most of this work was in the symbolic domain, as opposed to the numerical environment of neural networks, statistics and system identification. Despite this, some of the work on Inductive Learning is relevant to modelling nonlinear dynamic systems and, in general, the examination of different viewpoints often provide new insight into the learning mechanisms, improving understanding for both groups of researchers. Techniques such as decision trees have been used as models of dynamic systems (Isaksson et al., 1991). Omohundro describes a variety of standard algorithms which can be applied in many situations with more success than neural networks (Omohundro, 1987). *Case-Based Learning* also has applications in modelling (Kolodner, 1993), and we describe the overlap of Case-Based Learning with nearest neighbour methods and RBF neural networks in (Murray-Smith and Thakar, 1993), dealing with aspects of customer modelling from sales information, where the basis functions performed interpolation between cases. A general, easy to read overview of machine learning is given in (Weiss and Kulikowski, 1991), and (Shavlik and Ditterich, 1990) is a collection of the most significant papers in the field's history. (Michalski et al., 1983, Michalski et al., 1986) and (Kodratoff and Michalski, 1990) also provide collections of significant work.

2.1.4 Empirical modelling with neural networks

The basic philosophy of the *neural network* approach to machine learning is to use the style of information processing evident in the physiology of living beings as the inspiration for a computational paradigm. A large number of simple processing elements, based on highly simplified mathematical models of neurons, are densely interconnected by weighted connections which roughly represent the axons and synapses of the biological model. These structures are then optimised using a variety of algorithms, some aspects of which – although by no means the majority – are based on biologically plausible techniques.

The resulting networks can represent complex mappings from their input nodes to output nodes, making them interesting for engineering applications. The neural network architectures used in this thesis are examples of flexible black-box models which are created on the basis of input-output data pairs. There are several advantages of using neural networks to model systems, the most compelling of which is their ability to model continuous, non-linear, multivariable systems. As mentioned in the previous section, this ability is not unique to neural networks, but the progress made in recent years, due to the increase in available computing power, has shown that the new algorithms often perform well compared to previous methods of nonlinear system identification. More important, perhaps, is the multidisciplinary nature of the research, which has brought new ideas from artificial intelligence, mathematics and biology together with application-oriented engineering and computing practice.

A bit of history

The roots of the neural network approach to learning can be traced back to 1943, when McCulloch and Pitts (McCulloch and Pitts, 1943) made some proposals about how simple neural-like networks could compute. Hebb then suggested a biologically plausible learning rule for such networks (Hebb, 1949). The beginning of the field as it is known today can be found in Frank Rosenblatt's work on *Perceptrons* (Rosenblatt, 1962). He pioneered the use of formal mathematical analysis and the use of digital computers for simulation. He also made some over-enthusiastic claims about the power of perceptrons compared to normal computers which would spark off a debate which is still running to this day, but which then led to the field being laid dormant for twenty years. This was a combination of the limitations of the contemporary hardware, and the irritation caused to other scientists by Rosenblatt's claims. It was this irritation which lead Minsky and Papert to publish the famous *Perceptrons* book (Minsky and Papert, 1969) where they rigorously analysed the existing techniques and pointed out the limitations of the structures and their inability to scale up to larger problems.

(Widrow and Hoff, 1960), coming from a more engineering background also proposed a similar network called an *Adaline*, trained using the *Delta-rule*, which eventually found widespread use in telecommunications systems as an adaptive filter. Kurt Steinbuch was another engineer who did significant work in learning control systems, including his *Lernmatrix* learning system

(Steinbuch, 1961, Steinbuch, 1963). Grossberg's work in the 1970's introduced new ideas from biology and psychology to create the ART series of non-linear dynamic architectures. Albus made important contributions with the CMAC architecture in the field of learning in robotics, using many biologically motivated methods (Albus, 1972, Albus, 1975b, Albus, 1975a). Holland examined adaptation in (Holland, 1975), introducing *Genetic algorithms* and *classifier systems* for learning.

The main reason for the current renaissance in machine learning in general was the boom in 'artificial neural networks' in the eighties, precipitated by Hopfield's work (Hopfield, 1982, Hopfield, 1984) and the PDP Group's books (Rumelhart and McClelland, 1986). This was combined with a general frustration with the lack of progress in 'conventional' AI for tasks such as speech recognition, vision and pattern recognition, and the fact that the powerful computing hardware needed by these numerically intensive algorithms was now widely available.

The neural network 'label', as used in the current literature, seems to be applicable to almost any modelling or classification scheme, as networks can be used as convenient graphic representations for many mathematically described systems. (Barron and Barron, 1988) provides a clear description of the similarities between neural nets and statistical methods. General introductory books include (Haykin, 1994, Hertz et al., 1991, Wassermann, 1993). Historically important papers are reprinted in the *Neurocomputing* collections (Anderson and Rosenfeld, 1988, Anderson and Rosenfeld, 1990). Reviews of the neural network field for modelling and control include (Miller et al., 1990, Hunt et al., 1992, Warwick et al., 1992).

The Multi-Layer Perceptron

As discussed earlier, the first neural architecture to be implemented and studied in detail was Rosenblatt's Perceptron (Rosenblatt, 1958). This is a very simple system, which was suggested as a simple model of biological neurons present in the human visual system. The output is a function (usually non-linear, such as a hard limiter or sigmoidal function) of the sum of the weighted inputs.

$$\mathbf{y} = f(\mathbf{x}\mathbf{W} + bias), \tag{2.5}$$

where $f(\cdot)$ is the activation function, and $\mathbf{W}$ is the weight matrix connecting inputs $\mathbf{x}$ to outputs $\mathbf{y}$.

The extension of the perceptron to allow multiple layers – the *multi-layer perceptron* in equation (2.6) (see Figure 2.7 for a graphical representation), became widespread in the eighties after suitable learning algorithms, such as *back-propagation*, became widely known.

$$\mathbf{y} = f(\mathbf{W}_2 f(\mathbf{x}\mathbf{W}_1 + bias_1) + bias_2), \tag{2.6}$$

where $\mathbf{W}_1$ is the weight matrix connecting inputs $\mathbf{x}$ to the hidden layer neurons, and $\mathbf{W}_2$ connects the hidden layer neurons to the outputs $\mathbf{y}$. Back-propagation was developed independently by various workers (Werbos, 1974, Parker, 1985) with best known version being

(Rumelhart et al., 1986), bringing the neural networks field back to life. The Multi-Layer Perceptron soon became, for better or worse, the most commonly used (and abused) network architecture in the history of neurocomputing. The widespread use came about because of the

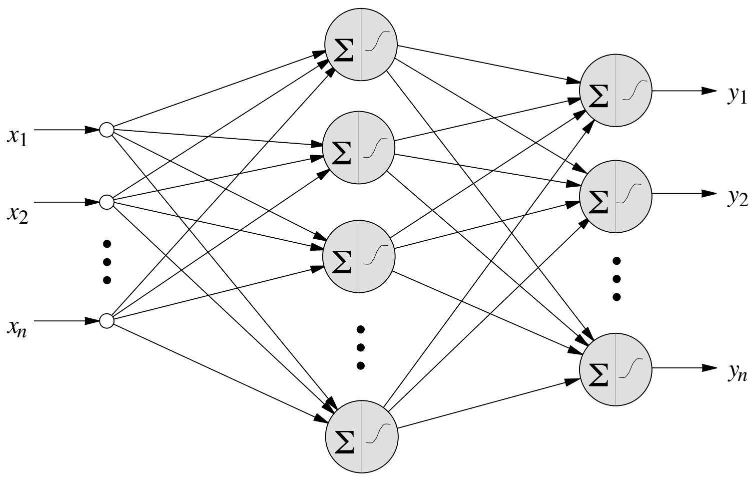


Figure 2.7: Multi-layer Perceptron

real flexibility of the structure in coping with complex high-dimensional problems, and because it managed to produce often excellent results compared to competing methods. The disadvantages of the structure are the slow training times associated with the back-propagation learning algorithm and poor transparency of the algorithm and of the trained networks. This led to an alchemy-like approach to training taken by the majority of researchers, producing a huge variety of heuristically motivated 'new improved' learning rules.

2.1.5 What is wrong with modelling with neural nets?

Empirical modelling as a methodology, even in its 'hard' form of System Identification, has its disadvantages. Often, the models produced in empirical modelling, even if they have achieved a useful representation of the process being modelled, give little physical insight into the process, since they are rarely based on detailed knowledge about the process structure. This is closely related to another disadvantage, which is that the product of a purely data based modelling process usually has limited validity in situations different to that in which the data was collected (e.g. at different working points, given different inputs, or a change in environment). The identification methods in existence are also still very much highly interactive art forms. They depend to a great extent on utilising as much knowledge about the process in question as possible, to simplify the modelling problem, and on an intuitive feel for the techniques used, as well as an understanding of the complex statistical tools available for the analysis of the data and estimation of the parameters.

While things have improved slightly in recent years, a characteristic of much work in the field of neural networks is the lack of rigour, compared to more established 'competing' areas such as statistics and system identification (see (Sjöberg et al., 1994) for a good discussion of the overlap between the fields, with the intention of 'removing the mystique' surrounding neural nets). This had certain advantages initially, as the fact that the area was linked to human intelligence often made it more appealing to young researchers, as well as to the various bodies which sponsor research and development, than the more highly mathematical established fields. Although the field has been rightly criticised for the wild claims made initially, it did bring in fresh ideas from other fields such as AI, psychology and physiology. These insights have enriched the scope and methods of the research into learning and adaptive systems, as well as a very application oriented style of research which got the method applied to a wide variety of problems.

The problem facing researchers at the start of the 1990's was that much of the experience gained in the existing fields had been ignored, and the field was too strongly linked to one architecture – the multi-layer perceptron. Initially much emphasis was laid on the fact that the multi-layer perceptron architecture is theoretically powerful enough to represent arbitrary nonlinear mappings, but such results did not help produce efficient training algorithms. A large proportion of the research was invested in a variety of problem specific 'fiddle factors' designed to speed up the optimisation process–which often took days of computing time–and to improve generalisation. The multi-layer perceptron can often be highly successful at modelling a given system, but despite this it is not ideal for many modelling tasks. The long training times are not suited to the iterative and interactive nature of the modelling process, especially as it was unclear when learning should be stopped, or how many units or layers a net should have for a particular problem. The poor interpretability limits the user's ability to validate a trained network, and limits the networks' applicability to safety critical situations. There is no straightforward way of introducing prior knowledge directly into the network structure, even though for many tasks the ability to simplify the problem using such knowledge is critical to reaching the desired level of accuracy.

Combining the different paradigms

While none of them is ideally suited to the task, the research paradigms for learning models described in this section can all contribute significantly to the goal of having a flexible, interactive learning system which robustly forms a model of a complex process, given a mixture of observed data and *a priori* knowledge. Systems theory and System Identification provide the user with a mathematically well founded base of theory and experience for modelling dynamic systems from observed data. Much of the work has been restricted to linear systems, but the basics remain relevant, and many of the algorithms can be directly applied to the hybrid local model architectures described in Section 2.3.2. Similarly with statistics, where the theory and experience developed by statisticians in areas such as experiment design, model optimisation, model validation and function approximation is an invaluable aid to a better understanding and evaluation of newer systems such as neural networks.

The models used in this thesis can benefit not only from numerically oriented techniques, but also from the integration of fuzzy logic aspects and symbolic machine learning systems which then allows the incorporation of linguistic or rule-based knowledge into the learning system. Section 2.3.2 presents an architecture suitable for the integration of the various paradigms.

2.2 The Modelling Process

The work in this thesis is aimed at improving techniques for the design of models of non-linear dynamic systems (the $f(\psi(t))$ in equation (2.1)), with the aid of computationally intensive data-driven techniques. The goal is to produce a mapping from the input space to the output space which best fits the observed data and meets the specified constraints. This involves integrating knowledge about the system with data observed from identification experiments. This allows the developer to produce better model structures and identify their parameters, as well as being able to validate the accuracy of the final model. Modelling from data and knowledge is often viewed as an art form, mixing 'expert' insight with the information in observed data, while using *ad hoc* simplifications to make the problem solvable. The typical

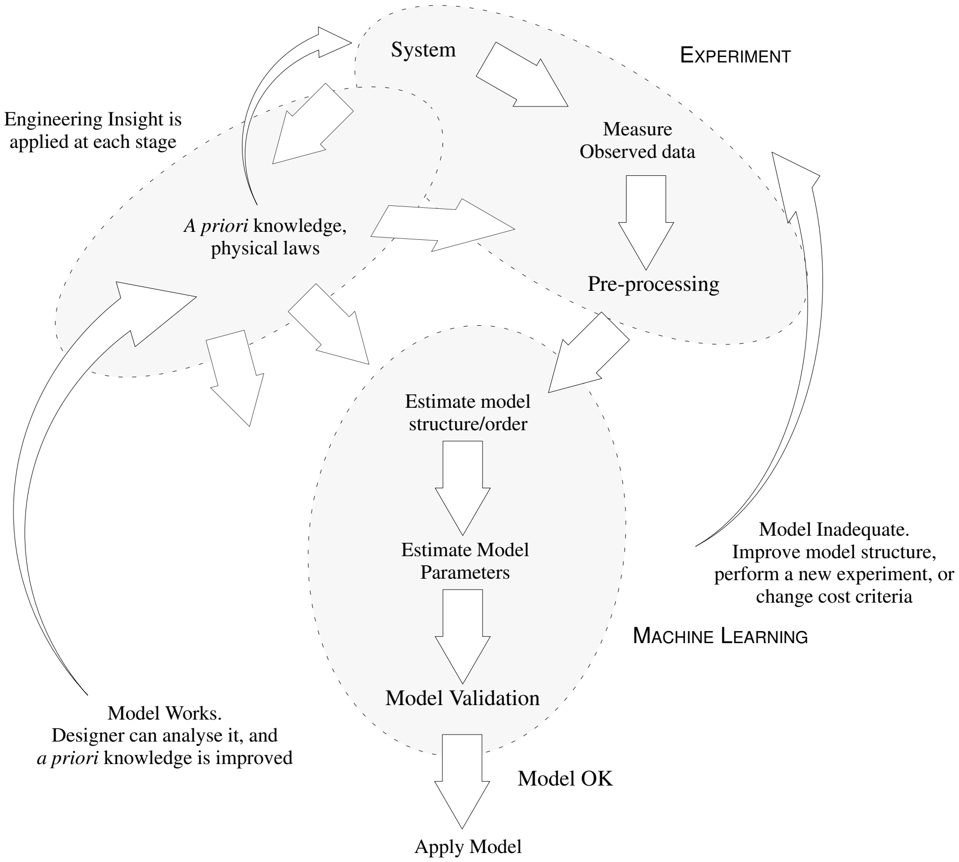


Figure 2.8: The Engineering Cycle for Training a Model

modelling cycle is shown in Figure 2.8, showing the interaction of *a priori engineering insight*, *experiment design*, *data acquisition* and *pre-processing*, followed by *machine modelling* and *validation*. Each of these phases will now be looked at in more detail.

2.2.1 Using *a priori* information

Modelling from observed data supported by learning systems is supposed to reduce the need to understand the detailed physical relationships within the system under investigation, but as can be seen in Figure 2.8, a major feature of the cycle is the important role of *a priori* knowledge at each stage of the modelling process. *A priori* information is initial knowledge about the system, or problem in question. This includes aspects such as the goals of the problem, the characteristics of the process, its parameters, the effect of the environment (expected noise, disturbances), and the robustness requirements for different situations. Learning or adaptive systems are usually used because of the insufficiency of the *a priori* information (few complex processes in reality can be described completely by *a priori* knowledge, due to the effects of noise, disturbances and unmeasured states). Such systems try to compensate for this by the continuous use of current information about the system:

> '*A priori* information is the basis for the formulation of an optimisation problem, but the current information provides the solution for the problem' (Tsypkin, 1971).

This describes the basis of Adaptive Control. Although the goal of learning systems is to reduce the need for *a priori* knowledge, its use almost invariably makes the learning problem more tractable. *A priori* knowledge can be used both *implicitly* and *explicitly*. It is used *implicitly* when framing the problem, in the act of creating a representative training set, deciding which learning algorithms and structures are best suited to the problem and which inputs and pre-processing algorithms are likely to make the learning task easiest.

The *explicit* use of *a priori* knowledge involves the direct integration of models or rule-bases into the learning system to reduce the learning effort. Knowledge about variable interaction can also be used to decouple the inputs, and reduce the dimensionality problems. Model structure, dynamic order and sampling rate are dependent on *a priori* knowledge. Knowledge about physical constraints can be used to limit the generalisation in areas with insufficient data. Existing models or controllers (e.g. human operators) can also be copied, where valid. Models of the environmental disturbances expected can also be included, as can knowledge about the relative importance of various areas of the input space.

Another important aspect of modelling from observed data is that the machine learning involved also usually results in the human engineer gaining a better understanding of the system, and the *a priori* knowledge about the system in question therefore being improved. The human designer is still a vital part of the process, and the goal should be to enhance the power of the interactive software by automating the learning process wherever possible, but by giving the engineer the freedom to intervene at each stage. As knowledge about the system being modelled increases, more possibilities of simplifying the machine learning should become clearer, allowing it to be used more successfully, closing the loop in the modelling cycle seen in Figure 2.8, where the *a priori* knowledge is improved by the availability of the validated model.

The final product, a working model, can therefore be seen as a contribution to the more general pool of engineering and scientific insight. The laws, rules or models we take as *a priori* today were also once poorly understood observed behaviour, which was then measured, analysed and turned into some simpler law or model. Kepler's laws of planetary motion were found only after painstaking acquisition of observed data, and the application of a variety of model structures to the data, estimation of the model parameters and validation of the models on new data!

2.2.2 Creating the training set – design & pre-processing

The *training set* $\mathcal{D}$ is the data set used for optimising the parameters and structure of the learning system. It is generally necessary to devote a great deal of attention to the process of extracting preprocessing and representing the data for training.

Experiment design

It is vitally important that the training set represents the task in hand correctly and adequately over the whole input space, but it is important to consider the relative importance of the various areas of the input space. In many situations a system spends most of its time in a particular operating region. It may make sense to weight this more heavily in the learning process. In others, a particular aspect of the model must be very accurate, for example the reaction to a step change in the inputs or disturbances is important for many processes. In others it is important to have very accurate models in relatively stable areas, as this is where the process spends most of its time. In many applications certain areas of the system are associated with danger – how should these be treated in the model? Constructing a representative training set is often far more difficult than learning the task from the training set. The general process of training set creation is shown in Figure 2.9[2]. A significant aspect of the diagram is the existence of important disturbances which can not be measured. The ability of the training procedure to cope robustly with such disturbances will often determine to a great extent the procedure's usefulness in real world applications. It is not only important

[2]In many cases the model is to be developed in order to produce a controller for the process in question. A valid point is to ask how this controller should be developed, without an adequate model to design it with? In critically unstable cases this will be a major problem, but in many cases the task is to improve on an existing controller which can be used for the purposes of data acquisition. There may, however, have to be a series of acquisition and modelling runs, as some aspects of the processes may not be apparent with less powerful controllers. In many cases, however, the learning system will have to make do with whatever data is available, because of the costs of extensive experimentation both in terms of money, as well as organisational effort.

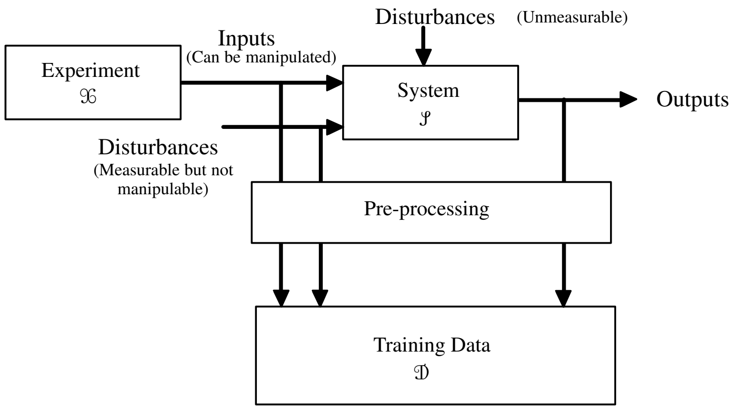


Figure 2.9: Acquiring and preparing the training data

to have training data covering the input space, but to have larger amounts of data where the decision surface is most complex, and to have some information about the relative significance of individual training data. This could be in the form of probability distribution functions, showing the relative frequency of particular situations, or the definition of areas which are particularly important. The science of creating an optimal sampling of the input space is called *Experiment Design* (Fedorov, 1972, Goodwin and Payne, 1977, Ljung, 1987), and has been picked up by workers in machine learning within the *active learning* framework.

Active learning

The neural network community traditionally threw every available training example at the network during the learning phase, irrespective of redundancy, noise levels, or local complexity in the process being modelled. The disadvantages of this procedure are being increasingly recognised, leading to the introduction of *active learning* methods which enhance the training set used during the learning process, as shown in Figure 2.10.

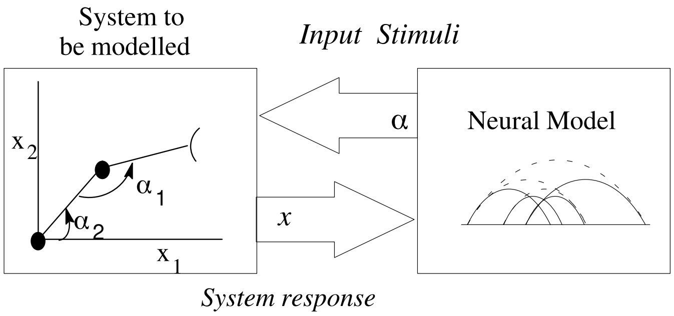


Figure 2.10: Active learning – exploring the input space. The learning system can interact with its environment to obtain new training data.

The research in *active learning* has gained momentum in recent years, see (Cohn et al., 1990, Cohn, 1994) and (Cohn et al., 1994). (Plutowski, 1994) is a recent thesis in the area. The use of local basis functions to guide active learning is described in (Murray-Smith, 1992). As described in (Thrun, 1992), active learning can be viewed as either *directed* guided search or *undirected*, random search. The undirected active learning methods described in the literature have the disadvantage that the effort needed to learn a given process increases exponentially with the dimension of the input space. Active learning can include *active sampling* and *active selection*,techniques:

- *Active sampling* is associated with the experiment design phase – in which regions of the input space should training data be acquired to best minimise the uncertainty in the model? The uncertainty estimate in the model is based on the information in the existing training set.[3]
- *Active selection*, which involves the most efficient use of a large existing training set uses closely related techniques to active sampling, without the connection to the environment.

The *active* label is due to the fact that the selection or sampling processes are dependent on the learning process and model structure, so that the learning process can be seen more broadly as actively *exploring* its environment or training data, then *exploiting* that data to optimise its performance.The use of complexity-based active learning algorithm in local model networks is described in Section 4.3, and an active sampling routine is used to improve the learning process in the rolling mill application in Chapter 6.

Pre-processing

A vital factor in all empirically driven model building approaches is that the training set should be pre-processed before learning. Any way in which the data can be transformed to make learning easier will lead to a significant improvement in performance. Pre-processing includes all action applied to the measured data, including which sensor information to use, how to sample it, how the amount of data can best be reduced, what transformations should be applied, and how can it be best encoded to make the learning task easier. The data may have to be pre-filtered to remove noise effects, anti-aliasing techniques will be necessary when continuous signals are sampled, outliers can be removed, or extra information can be used to reduce the distorting effect of measurable disturbances. In many cases non-linear characteristics in sensors or actuators are well known in advance, and the nonlinearity can be counteracted before learning commences. Somewhere between pre-processing and representation lies the aspect of minimising variable interaction. The data can be transformed to a lower

[3]The concept of giving a learning system a 'sense of curiosity', based on internal estimates of its own accuracy, and giving it the ability to search the most promising areas of the input space for new information is of major importance for the future of autonomously learning systems, and the techniques used are closely related to those needed for the recursive identification of time-varying processes, where it is important to be able to 'forget' the training examples in the right areas, while learning from those describing the new behaviour of the process.

dimension using principal components analysis to reduce the problems related to the '*curse of dimensionality*'. It may be known that particular variables interact in a given way, whether additively, multiplicatively or nonlinearly (Hastie and Tibshirani, 1990). This information can be used to limit the freedom of the learning system, forcing it to learn more efficiently and generalise more robustly (see Section 2.5.4 for more details on how this information can be used). In general, the pre-processing applied to the raw data is often a critical stage in the development of models from observed data, and its importance should not be underestimated for learning systems.

2.2.3 Learning algorithms and knowledge representation

Once the training data has been acquired and pre-processed, a *knowledge representation* ability is required for the model to be able to learn the data and a *learning algorithm* which can be used to go from the information presented to the learning system, in the form of inputs, outputs and feedback from the environment, to the desired representation. In simple cases this can be viewed as an optimisation process, where the optimal set of parameters for a given structure is found, or it can also involve the *construction* of the representation itself. Learning systems involve a wide variety of knowledge representation techniques, from simple numerical parameters or symbolic features to complex specialised structures. Each style of knowledge representation is biased towards a particular class of problems, a fact which can be seen in the classical fields of statistics and system identification, as well as the newer areas of neural networks and machine learning, each of which tend to use structures suited to the problems faced in that field

When considering learning systems in general, for any given data set, a variety of possible *model structures* usually compete for the best representation of the data. It is important that the most suitable style of representation is chosen for a given problem, as the right choice of model structure will be a major factor in producing a model which is able to 'generalise' correctly. *Generalisation* is the ability of a learning system to give a 'correct' output to inputs on which it has never been trained (see Figure 2.11). These could be new, noisy or incomplete inputs. Memorisation, or learning the training set to perfection is *not* the goal of learning – a random access memory (RAM) can do this adequately!

The resulting quality of generalisation for a given problem will depend on the problem definition, encoding of the features, the quality of the training set, the power and suitability of the representational structure $\mathcal{M}$ and the learning algorithm used. To analyse the trade-off between learning the training set $\mathcal{D}$, and generalising to unseen inputs, it can be helpful to decompose the modelling error $J(\mathcal{M},\mathcal{D})$ into two aspects, the *bias* $J_B(\mathcal{M},\mathcal{D})$ and the *variance* $J_V(\mathcal{M},\mathcal{D})$ (Geman et al., 1992).

$$J(\mathcal{M},\mathcal{D}) = J_V(\mathcal{M},\mathcal{D}) + J_B(\mathcal{M},\mathcal{D}), \tag{2.7}$$

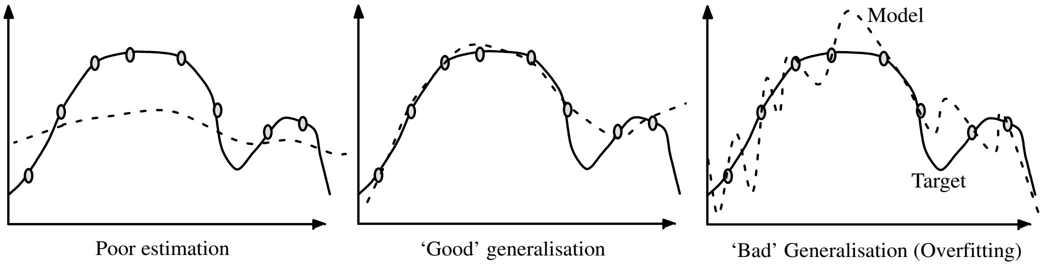


Figure 2.11: The generalisation/overfitting dilemma.

where the bias is

$$J_B(\mathcal{M},\mathcal{D}) = \Big(E_{\mathcal{D}}[\hat{f}(\mathbf{x};\mathcal{M},\mathcal{D})] - E\left[y(\mathbf{x})|\mathbf{x}\right]\Big)^2, \tag{2.8}$$

where $E_{\mathcal{D}}$ is the expectation over the training set $\mathcal{D}$, and the variance

$$J_V(\mathcal{M},\mathcal{D}) = E_{\mathcal{D}}\left[\left(\hat{f}(\mathbf{x};\mathcal{M},\mathcal{D}) - E_{\mathcal{D}}\left[\hat{f}(\mathbf{x},\mathcal{M},\mathcal{D})\right]\right)^2\right]. \tag{2.9}$$

The *bias* of a model is the average difference from the real system of models trained on a number of training sets, thus indicating what the system could not learn, even when given the information. The *variance* of a model is the average variation of the model estimates from all trained models to the 'average' model over all data sets, and can be used as an indicator of how robust the learning process was.

The flexibility required of the model to be able to model an arbitrary data set (reducing the bias part of the error) conflicts with the desire to reduce the variance of the resulting estimate. Use of a large flexible representation will reduce the bias, but without a correspondingly large data set is likely to lead to the *overfitting effect*, where the training set is learned adequately, but generalisation is poor – equivalent to high variance. The system has either learned the noise in the data, or has learned the data correctly, but interpolates between data points poorly. It is therefore important to use a suitably sized model structure for the given training data – the system must be over-determined (i.e. more training data than parameters, and – importantly for nonlinear systems – it must be locally over-determined in the complex areas of the input space). *One of the most important features of a learning algorithm is that it reliably finds a solution which generalises robustly to new data.* The regularisation methods described in Section 2.5.1 are methods which artificially increase the bias to improve the variance and therefore create more robust networks. In practice, algorithms which produce reliable models will be far better suited to engineering problems than algorithms which sometimes produce excellent results on one problem but then fail dismally on the next application.

2.2.4 Model validation

Once the model structure and parameters have been identified it is necessary to validate the accuracy of the final product. This is obviously a very important stage in a process which by its very nature has relatively little in the way of 'common sense' intuition about the behaviour of the target system. It is therefore important to combine data-driven validation – is it adequately accurate and robust for its purpose? – with more subjective validation, i.e. does the model behave in a way which seems physically plausible? Can the final machine learned model be interpreted to give the human engineer a better understanding of the system in question? Model validation is an issue which has often been neglected in the literature on learning systems, but one which is very important in industrial situations. There will usually be a trade-off between flexibility and interpretability, the outcome of which will depend on their relative importance for a given application.

Use of n-fold cross-validation

The most commonly applied method of predicting the accuracy of a neural network is that of measuring the quality of the system's response on the training and test sets, assuming that the data set was complete enough to have encountered all of the important areas of the input space. The general technique is called cross-validation. Generalisation ability is closely tied up with the concept of expected error prediction, so this is of fundamental importance to learning systems (Stone, 1974).

Resampling methods such as cross-validation aim to give unbiased estimates of the error rate of a learning system. They make minimal assumptions about the statistics of the training data. The simplest form of this is to use two sets of data (training and test sets), where the first is used to train the system and the test set is used to validate the results. A more general form of this is n-fold cross-validation, where the available data are partitioned into n subsamples. Each subsample is tested by training the net with the other $(n-1)$ subsamples and the error rate is then the average of these n sub-samples. This reduces the bias present in the error estimate, but is often a time consuming process and the error rates have a high variance. (Weiss and Kulikowski, 1991) *Leave-one-out validation* is a special case of cross-validation, where one example of the training set is left out to provide a test example for the model trained on all the other examples. This is then repeated until each member of the training set has been used as a test example. It is therefore very computationally expensive, and although it provides the most accurate prediction of error, is only suitable for problems with small training sets, or few parameters.

Cross-validation was initially not used with neural networks because of the computational expense of running the system several times, which for MLPs trained with back-propagation, can take several days, was fairly unrealistic. Less computationally expensive methods have been developed (e.g. Generalised Cross Validation GCV (Wahba, 1990), Aikaike's Final Prediction Error (FPE) have been used to optimise the size of multi-layer networks).

Cross-validation is therefore useful for automatically finding the optimal model structures or parameter estimates for a given data set, as well as supporting the human designer in validation and interpretation of models. The model can start small and increase its size until the cross-validation results start to show too much variance in the test errors.

The reliability of the accuracy estimates achieved by methods such as cross-validation depends strongly on the adequacy of the training data. If the training data is present in sufficient quantity throughout the input space (complex parts of the model will need a related number of data to train and test the parameters in that area) this provides a good estimate of the predictive power of the model. The amount of training data needed is also related to the noise on the training data. As the noise level increases, the amount of data points needed to train and validate a particular model increases.

2.2.5 Organisational aspects in empirical modelling projects

The procedure for the successful development of a model is often a major undertaking, involving a variety of experts from various fields. The data must be acquired somehow, usually from an experiment carried out by an experienced operator. The experiment design should ideally be an interactive process, involving several steps, including information from earlier modelling attempts. Initial data processing will involve signal processing engineers, the goals of the modelling are set by a mixture of business and engineering constraints, and the validation of how well the goals were achieved involves every link of the chain. For most realistic problems there will also have to be many iterations towards the goal of an accurate, useful, reliable model, so the representation used must be promote co-operation between specialists with very different backgrounds, and be able to integrate different types of *a priori* knowledge either directly as model structure, or in the optimisation process as cost-functions and constraints.

2.3 Local Methods in Modelling

The previous section described the aspects of the modelling process which should be supported by a learning architecture. This thesis discusses variations on one main class of network, the *Basis Function Network (BF Net)* because of its suitability for modelling continuous nonlinear systems. The BF nets used in this thesis are basically local structures, which inherently involve modularisation in representation and allow the easy integration of ideas and structures from other modelling paradigms.

2.3.1 Basis Function Networks for modelling

The basic Basis Function Network described in equation (2.10) is shown in Figure 2.12. The output[4] y is a weighted (by parameters $\boldsymbol{\theta}_i$) linear combination of the activations of the many

[4]The models discussed in this report are all Multi-Input/Single Output models. The extension to multi-output systems is mathematically straightforward. The optimisation of the units' weights is unchanged, other

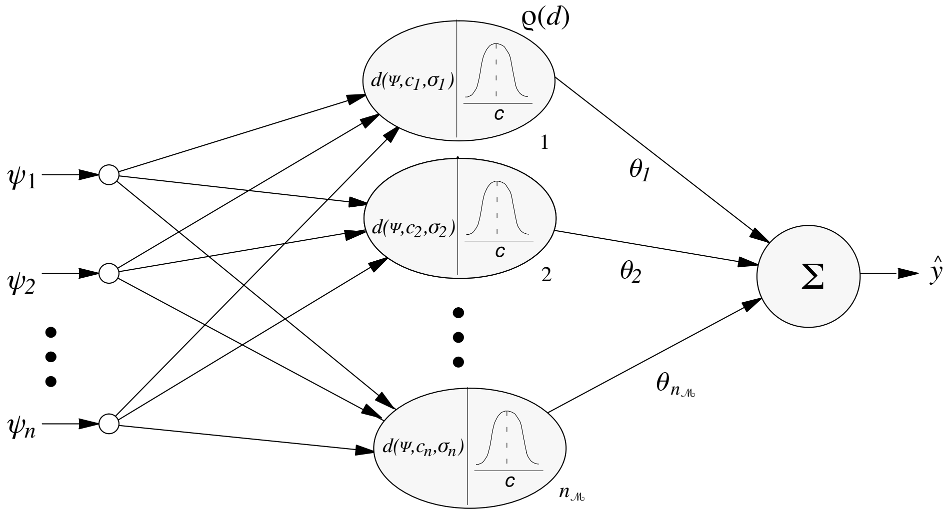


Figure 2.12: Radial Basis Function network

($n_{\mathcal{M}}$) locally active non-linear *basis functions* $\rho_i(\cdot)$ which react to the input vector $\boldsymbol{\psi}$,

$$y = f(\boldsymbol{\psi}) = \sum_{i=1}^{n_{\mathcal{M}}} \theta_i \rho_i(\boldsymbol{\psi}). \tag{2.10}$$

The nonlinear basis functions basically map inputs into a higher dimensional space, where it is easier to learn the mapping to the outputs than from the original input space, so that a linear connection to the output suffices. The optimisation of the weights then becomes a linear process, meaning that the optimal solution can be found using the standard tools of linear optimisation theory, including a variety of powerful methods for coping with *ill-posed* problems (see Section 2.5.1), and methods for analysing the covariance of the parameter estimates (see Section 3.2.3), from which aspects such as experiment design criteria can be obtained.

Basis functions

Each unit's centre is a point in the input space, and the *receptive field* of the unit (the *support*, or volume of the input space to which it reacts) is defined by its distance metric $d(\boldsymbol{\psi}; \mathbf{c}, \sigma)$. The *basis* or *activation function* (similar to the *membership function* of a fuzzy set) of the unit is usually designed so that the activation monotonically decreases towards zero as the input point moves away from the unit's centre ($\mathbf{c}_i$), e.g. B-Splines or Gaussian bells are common choices.

than that a matrix of output values is used instead of a vector. The optimisation of the model structure is obviously more difficult for multi-output problems, because the nonlinearity and complexity for the various output spaces will not always be in the same areas of the input space. The use of a single model structure with multiple outputs would mean that for the total model there are fewer parameters to optimise, which would suggest a lower variance than for a decomposed model. It may, however make more sense to decompose the problem into several single output problems, because each sub-problem will then have the required complexity and basis function locations for the complexity of the output in question, and will not produce an increase in variance in the other output variables, which happens in the multi-output case.

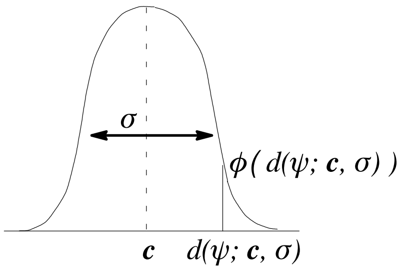


Figure 2.13: A typical locally active, smooth basis function

In Radial Basis Function (RBF) nets, the basis functions are composed of two elements. The distance metric $d(\boldsymbol{\psi}; \mathbf{c}_i, \sigma_i)$ for basis function i, defined in equation (2.11), can scale and shape the spread of the basis function relative to its centre $\mathbf{c}_i$, depending on its width σ_i, and the basis function itself $\rho(\cdot)$, which takes the distance metric as its input

$$d(\boldsymbol{\psi}; \mathbf{c}_i, \sigma_i) = \frac{(\boldsymbol{\psi} - \mathbf{c}_i)^2}{\sigma_i^2}, \tag{2.11}$$

and can be generalised by using a matrix $\boldsymbol{\sigma}_i$ instead of a scalar, giving an ellipsoidal basis function (equation (2.12)). Training algorithms for such distance functions are given in Section 4.2.3.

$$d(\boldsymbol{\psi}; \mathbf{c}_i, \sigma_i) = |\boldsymbol{\psi} - \mathbf{c}_i|^T \boldsymbol{\sigma}_i^{-1} |\boldsymbol{\psi} - \mathbf{c}_i| \tag{2.12}$$

There is a wide variety of possible basis functions, e.g. the Gaussian bells used in this thesis, as in equation (2.13), B-Splines, thin plate splines, linear functions etc.

$$\rho_i(\boldsymbol{\psi}) = \rho(d(\boldsymbol{\psi}; \mathbf{c}_i, \sigma_i)) = \exp(-d(\boldsymbol{\psi}; \mathbf{c}_i, \sigma_i)), \tag{2.13}$$

(Carlin, 1992) compares the effectiveness of a variety of basis functions for modelling and control purposes. The basis functions used in this report will be assumed to have local properties[5], i.e. they are active in a limited area of the input space because of the improvement in transparency achieved.

If the units have localised receptive fields, *and a limited degree of overlap with their neighbours*, the unit's weights can be viewed as locally accurate piecewise constant models (in more complex networks, more general local models can be used, see Section 2.3.2), whose validity for a given input is indicated by their unit's own activation functions for a given input.

For modelling and control of continuously differentiable processes, the basis function should be smooth, and if the basis functions are to be local, they must decrease monotonically from a maximum at $\boldsymbol{\psi} = \mathbf{c}$ (distance metric = 0) towards zero, according to the distance metric $d(\tilde{\boldsymbol{\phi}}; \mathbf{c}, \sigma)$. This forces the influence of the local model associated with the basis function to decrease as the inputs move away from its centre (where the basis function's local model is the most accurate representation of the system).[6]

[5] Note that strictly speaking the Gaussian is not a local basis function, as it does not have compact support.
[6] Given relevant *a priori* knowledge, more complex basis functions can be used which do not adhere to the

Partition of unity

For modelling tasks the basis functions should form a *partition of unity* for the input space, i.e. at any point in the input space, the sum of all basis function activations should be 1. This is a necessary requirement for the network to be able to globally approximate systems as complex as the basis functions' local models, e.g. in the straightforward single weight case, so that constant areas of the input space can be modelled exactly. The partition of unity ensures that every point in the input space has an equal weighting, so that any variation in output over the input space is due only to the parameters $\boldsymbol{\theta}$ weighting the basis functions' activation. In many applications the network's basis functions are normalised to achieve the partition of unity, i.e.

$$\rho_k(\boldsymbol{\psi}) = \frac{\rho\left(d\left(\boldsymbol{\psi}, \mathbf{c}_k, \sigma_k\right)\right)}{\sum_{i=1}^{n_{\mathcal{M}}} \rho\left(d\left(\boldsymbol{\psi}, \mathbf{c}_i, \sigma_i\right)\right)} \tag{2.14}$$

where $\rho(\cdot)$ is the general *unnormalised* basis function, so that the *normalised* basis functions $\rho_k(\cdot)$ sum to unity,

$$\sum_{i=1}^{n_{\mathcal{M}}} \rho_i\left(d\left(\boldsymbol{\psi}, \mathbf{c}_i, \sigma_i\right)\right) = 1. \tag{2.15}$$

(Werntges, 1993) discusses the advantages of normalisation in RBF nets, promoting somewhat simplistically the advantages of a partition of unity produced by normalisation. Normalisation can be important for basis function nets, often making the model less sensitive to poor choice of basis functions, but it also has a number of side-effects which are discussed in detail in Section 3.3. These side-effects make the argument for or against the use of normalisation far more complicated than is often assumed.

Literature of Basis Function nets for modelling

Basis Function Networks and their equivalents have been used for function approximation and modelling in various forms for many years. The original Radial Basis Function Nets came from *Interpolation theory* and are described in (Powell, 1987), where a basis function is associated with each training point, as in (Specht, 1991). *Potential Functions* (Aizermann et al., 1964), *Kernels* (Wahba, 1992) and *Spline Models* (Wahba, 1990) are all similar structures. The literature of *local learning* methods in statistics is reviewed in (Atkeson, 1990). These methods store the training data and for a given input point form a locally weighted representation of the system from the related training points. *Smoothing methods* such as Gaussian Kernel methods, as described in (Hastie and Tibshirani, 1990) like other local averaging methods, suffer from the curse of dimensionality (Friedman, 1991), and are computationally expensive.

Aldus's CMAC ideas have a great deal of overlap with BF Nets with uniformly distributed local basis functions (Albus, 1975b, Lane et al., 1991, Brown and Harris, 1994). The close

features above, but which are specially relevant to a particular application, e.g. sinusoidal basis functions are a good choice if it is known that oscillatory components play an important role in the system being modelled.

relation of basis function nets to classes of *fuzzy logic systems* has also been discussed in (Jang and Sun, 1993), (Haas and Murray-Smith, 1993) and (Brown and Harris, 1994), where the similarity between membership functions and basis functions is pointed out. *Mixture Models* used in statistics are created by mixing a number of probability distributions, and have many similarities to RBF nets (Bishop, 1994) and (Xu et al., 1994).

Recently BF neural networks have received a growing amount of attention from the neural network community, starting with the early papers (Moody and Darken, 1989, Jones et al., 1989, Broomhead and Lowe, 1988). (Hlavácková and Neruda, 1993) gives a brief review of the use of RBF nets. (Poggio and Girosi, 1990) (Girosi et al., 1993) and (Mason and Parks, 1992) describe the networks within the mathematical framework of *Regularisation Theory* for function approximation. (Hutchinson, 1994) describes the use of RBF nets for financial time series modelling. (Park and Sandberg, 1991) and (Park and Sandberg, 1993) proved the universal approximation abilities of RBF nets.

Use of RBF nets for modelling and control purposes is described in (Barnes et al., 1991) (Sanner and Slotine, 1992) (Sbarbaro, 1992a) and (Pantaleón-Prieto et al., 1993). The methods were applied in (Röscheisen et al., 1992) to control a rolling mill. Early work related to this thesis can be found in (Murray-Smith et al., 1992) and (Neumerkel et al., 1993). RBF networks which use the local nature of the basis functions to give a local prediction of their own accuracy throughout the input space are examined in (Leonard et al., 1992) (this idea is extended to the local model nets in Section 3.2).

The *Gaussian Bar* nets (Hartman and Keeler, 1991, Kurcova, 1992) are also closely related, although less powerful. These are nets with 'semilocal' units formed by taking one-dimensional local basis functions and forming their tensor product to approximate a multi-variable function, as with tensor product spline models such as the ASMOD system (Kavli, 1992). The advantage of these units is that they can better cope with some classes of high dimensional problems, as they do not need to cover 'uninteresting' dimensions. A disadvantage of this style of network is that the representation does not cope well with processes where the nonlinearity depends on several variables, such as the mars1 benchmark used in Chapter 4.

Pao's *Functional Link Network* (Pao, 1992), and Billing's closely related *Extended Model Set* ideas (Billings and Chen, 1989), use links with a fixed non-linear function built in to expand the input vector, resulting in the production of extra 'higher-order' inputs. These are linearly weighted by the networks parameters. The well known *Polynomial methods* can also be viewed as Basis Function systems, as there is a single layer of nonlinear functions, and the parameter optimisation is a linear process. The problem here is to find the suitable model structure – i.e. which set of basis functions can approximate the system adequately. Some off-line structure identification algorithms are described in (Chen and Billings, 1994, Ivakhnenko, 1971). Holden describes a general framework for basis function nets, calling them *Phi-nets* in (Holden, 1994).

Are Basis Function nets too local?

An intrinsic feature of the Basis Function networks is the concept of 'locality'. In linear systems the data, optimisation and validation are all considered to be globally relevant, i.e. any results obtained are valid over the entire input space, whereas in nonlinear systems the process complexity varies throughout the input space. For nonlinear systems, however, (especially when multivariable) the problem can be simplified by partitioning the input space into multiple subspaces. This can involve a reduction of the problem's dimensionality by decomposing the problem, discarding irrelevant interactions, or of simply partitioning the input spaces into subspaces which are easier to handle – the traditional 'divide and conquer' strategy inherent to local modelling techniques.

The concept of 'locality' is obviously relative, depending on the complexity of the system, the availability of training data, the importance of the given area of the input space, and *a priori* knowledge of internal structures within the given system. The form of 'locality' utilised depends on the representation used; in decision trees, locality is introduced by partitioning the input space into hypercubes, in RBF nets locality is hyperspherical and in multi-layer perceptrons or Projection Pursuit nets locality is a projection of the input space. This locality can be used to make networks more transparent and computationally efficient, which can then make learning algorithms which utilise the locality more efficiently than alternative algorithms.

The problem with standard basis function networks is that the crudeness of the local approximation (weighted piecewise constant models), suffers like other local methods from the 'curse of dimensionality' (Bellman, 1961). This forces the system to use exponentially increasing numbers of basis function units to approximate a given system, as the input dimension increases. This leads to computational, transparency and robustness problems (the training data to train all of the units has to exist!). It is therefore important to be able to profit from the local nature of the basis functions while not having to have too many units[7]. This implies that the basis functions should be associated with more powerful representations than piecewise constant models, so that a smaller number of them could cover larger areas of the input space while achieving the desired modelling accuracy.

2.3.2 Local Model basis function nets

Linear models, although very restricted in their representational ability have proved to be very useful for a large range of problems. This is due to their simple representation, their easy interpretability, and their robustness to noisy or missing data. It makes sense therefore to include the ability to at least be able to form a linear model within the network, as in

$$\hat{y} = \hat{f}(\boldsymbol{\psi}) = \theta_1 + \sum_{i=2}^{n_\psi+1} \theta_i \psi_i + \sum_{i=n_\psi+2}^{n_{\mathcal{M}}+n_\psi+1} \theta_i \rho_i(\boldsymbol{\psi}), \tag{2.16}$$

[7] see (Lowe, 1994) for a discussion of further arguments against local basis functions.

where n_ψ is the dimension of the input vector $\boldsymbol{\psi}$. The basic architecture (Poggio and Girosi, 1990) is a simple improvement which is highly relevant for practical applications, e.g. used in (Hutchinson, 1994).

The ability to use linear (or other) models, can also be introduced in a more general way. Standard basis function networks can be generalised to allow not just a constant weight to be associated with each basis function, but a more general function of the inputs, so that the network can be described in the form

$$\hat{y} = \hat{f}(\boldsymbol{\psi}) = \sum_{i=1}^{n_{\mathcal{M}}} \hat{f}_i(\boldsymbol{\psi})\rho_i(\tilde{\boldsymbol{\phi}}), \tag{2.17}$$

where $\tilde{\boldsymbol{\phi}}$ defines the *operating point* of the system. This is a vector which can be defined on a lower dimensional subspace of the input space which is covered by the basis functions. These can be seen as *scheduling* or *gating* functions for the local models which are defined on the full input space.

The basis, or *model validity functions* used in this thesis are radial, i.e. they use a *distance metric* $d(\tilde{\boldsymbol{\phi}}; \mathbf{c}_i, \sigma_i)$ which measures the distance of the current operating point $\boldsymbol{\psi}$ from the basis function's centre $\mathbf{c}_i$, relative to the width variable σ_i, as in equation (2.12). They are also *normalised*, so that they sum to unity, as in equation (2.14). See Figure 2.14 for a simple representation of operating regimes in a two dimensional operating space. The overlapping operating regimes allow the basis functions to smooth the transfer from one region of the model structure to the next.

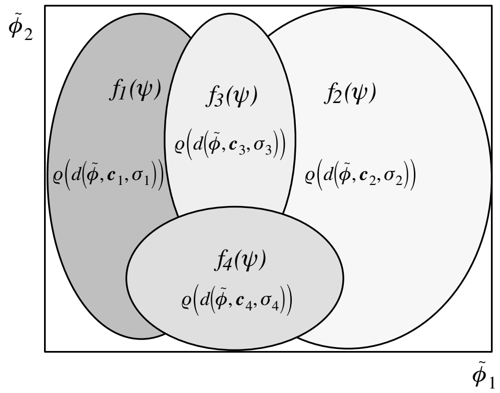


Figure 2.14: Local Model Operating Regimes. Each local model is associated with an operating regime. These regimes overlap, and the gradual decay of 'validity' provides interpolation between models.

This means that the structure has the advantages inherent to the local nature of the basis functions while, because of the more powerful local models associated with the basis functions, not requiring as many basis functions as before to achieve the desired accuracy. The improvement is more significant in higher dimensional problems. This generalisation of the

BF network to the *Local Model Network* described in equation (2.17), where the weights have been generalised to allow not just a constant weight to be associated with each parameter, but to have a function of the inputs weighted by the relevant basis function, has been applied by a number of authors. The $n_{\mathcal{M}}$ local models used are represented by general functions of the inputs $f_i(\boldsymbol{\psi})$, but in many cases simple linear models are chosen

$$\mathcal{M}_i(\boldsymbol{\theta}) = \hat{f}_i(\boldsymbol{\psi}) = \boldsymbol{\theta}_i^T \, [1 \; \boldsymbol{\psi}] \, . \tag{2.18}$$

The network form of equation (2.17) is shown in Figure 2.15. The trained network structure can be viewed as a decomposition of the complex, nonlinear system into a set of locally active sub-models, which are then smoothly integrated by their associated basis functions.

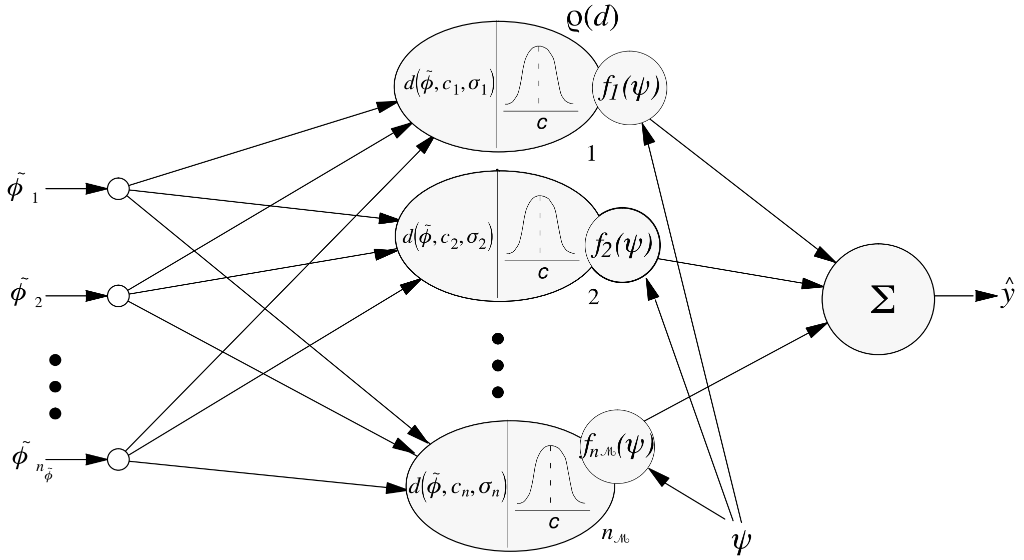


Figure 2.15: Local Model Basis Function network

To illustrate the workings of a local model network, a one dimensional function is mapped using local models in Figure 2.16. The top plot shows the target function and the model's approximation, while the basis functions and associated local models are shown below.

Literature of local model methods in learning and modelling

The representational ability of the normal Basis Function (BF) net can be extended to a generalised form of BF network, where the basis functions are used to weight other functions of the inputs as opposed to straightforward weights. This was suggested in (Jones et al., 1989), followed up by (Stokbro et al., 1990) and (Barnes et al., 1991). The Adaptive Expert networks in (Jacobs et al., 1991) are essentially local model systems, where the local models are called *expert networks* and the integration of the various *experts* is made by *gating networks.* These were developed into hierarchical models in (Jordan and Jacobs, 1991, Jordan and Jacobs, 1993).

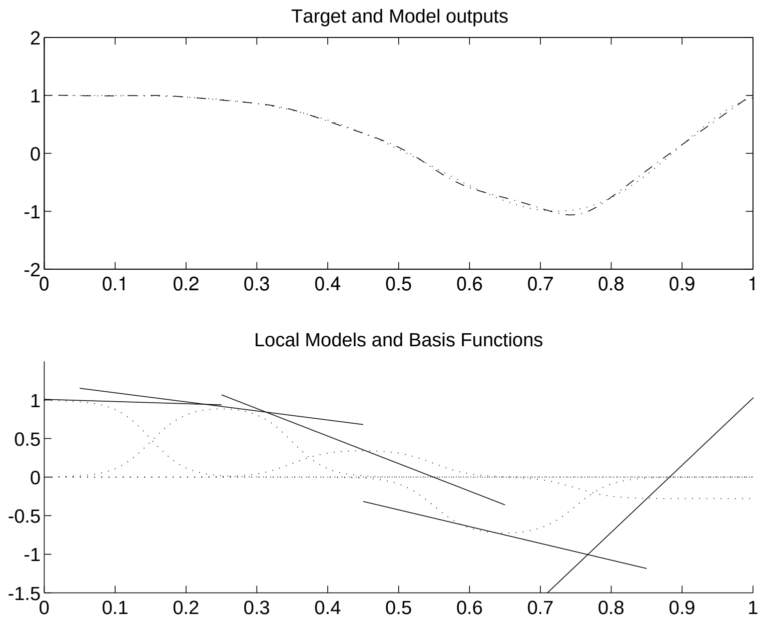


Figure 2.16: An example of local models representing a one dimensional function

The advantages of local representations are discussed in (Bottou and Vapnik, 1992), where they suggest that a proper compromise between local and global methods will usually prove most effective as varying levels of complexity are required throughout the input space, although they claim that the 'local capacity' should match the data density, which is not necessarily true, as this would not place units where the system is complex, but rather where there was most data. The more general goal of allocating local capacity is that the learning system should match the *'local complexity'* of the target system.

The idea of using locally accurate models is also described in the statistical literature in (Cleveland et al., 1988), where local linear or quadratic models are weighted by smoothing functions. (Priestley, 1988) describes *State Dependent Models* for non-linear time series which are basically linear models where the parameters depend on the operating vector $\tilde{\boldsymbol{\phi}}$ (corresponding to the 'state' in Priestley's terminology)

$$\hat{y}(t) = \boldsymbol{\psi}^T(t-1)\sum_{i=1}^{n_{\mathcal{M}}} \theta_i \rho_i(\tilde{\boldsymbol{\phi}}). \tag{2.19}$$

Local Model nets could be viewed as a finite parameterisation of the state-dependent model. Tong's *Smooth Threshold Autoregressive* (STAR) models (Tong, 1990) are also structurally equivalent to local model nets. In neither Priestly's or Tong's case, however, is much detail given about how to find the smooth weighting functions. (Billings and Voon, 1987) also uses a number of linear models to approximate a nonlinear system, but does not have smooth interpolation between basis functions.

Local model systems for diagnosis, modelling and control of dynamic systems have been applied extensively by Johansen and Foss in Trondheim, e.g. (Johansen and Foss, 1992c), (Johansen and Foss, 1992b) and (Johansen and Foss, 1993). A development of the ideas to a state-space implementation of local models is described in (Johansen and Foss, 1993).

Some *fuzzy logic* systems can also be viewed as Local Model networks, e.g. the methods used in (Takagi and Sugeno, 1985) are effectively overlapping piecewise linear models, with the interpolation between models provided by the membership functions. Similar applications are reported in (Sugeno and Kang, 1988, Foss and Johansen, 1993), (Wang, 1994) and (Harris et al., 1993). In (Haas and Murray-Smith, 1993) we discuss the similarity between fuzzy and basis function systems in more detail. (Back and Tsoi, 1991) describes the use of dynamic models as nodes in Multi-Layer Perceptrons – this could be seen as an MLP implementation of Local Model methods.

(Skeppstedt et al., 1992) describes the use of local dynamic models for modelling and control purposes, but with hard transfers from one model regime to the next. (Pottmann et al., 1993) describes a multi-model approach where although the local models overlap there is still a sharp transition from one model to the next, and to minimise model switching a heuristic criterion is introduced which only allows switching after three consecutive steps in the direction of the new model.

2.4 Hierarchical Approaches to Learning Models

We have discussed the use of locality in basis function networks for the limitation of complexity, but how can we decide on the suitable level of locality for any given area of the input space? How can we reduce the effect of high dimensionality by only concentrating on the areas of interest? Can we produce efficient training algorithms which take points outside a given local model into account, while not reducing the system to a global optimisation technique?

A common technique for the control of complexity, found in nature, society and technical systems, is *hierarchy* (Mesarovic et al., 1970). In the flat local model networks used so far, the structure identification phase tends to be computationally expensive, because any alteration to one unit affects many others. The effect of normalisation also alters the local properties of the basis functions, sometimes leading to unexpected results, as described in Section 3.3. Hierarchical methods offer, due to their structure, the ability to more effectively hide local complexity, in the traditional 'divide and conquer' manner, making the learning process more efficient and robust. The goal is truly hierarchical learning, where a network can grow to fit the data, and as the representation of the model improves decisions made earlier in the learning process can be reevaluated, leading to gradual changes to the higher levels. The local model framework is well suited for the creation of such hierarchies, as the local models in a given network can be further local model networks producing a hierarchy of local model nets. This is the Learning Hierarchy of Models (LHM) structure described in Chapter 5.

2.4.1 Hierarchical learning methods

The use of hierarchy in learning algorithms breaks down into two general camps. The decision tree methods started in the 1970's with *k-d trees* (Bently, 1975), *Classification And Regression Trees* (CART) (Breiman et al., 1984), and *ID3* and *C4.5* (Quinlan, 1993), and involve hierarchies of sharp partitions, dividing the input space into ever smaller areas. (Isaksson et al., 1991) and (Strömberg et al., 1991) describe the use of such trees with dynamic models in the leaf nodes to model nonlinear dynamic systems. Because of the sharp partitions, however, these methods are poorly suited to modelling continuous systems. An alternative approach is the

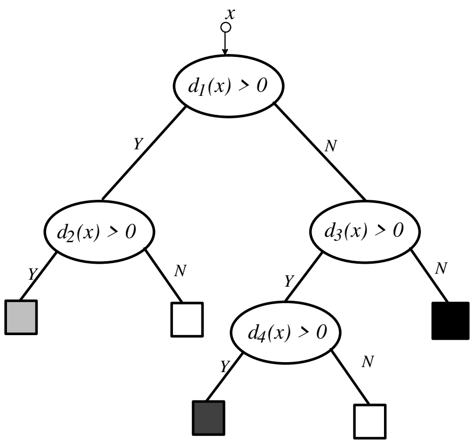


Figure 2.17: Decision tree structure

use of *soft splits*, where the input space is no longer sharply partitioned, but there is a gradual transfer from one local model space to the other, allowing smooth interpolation between the models. This also means that a point in the input space can activate models in several leaves of the tree, with a differing level of membership to each.

Examples of this type of structure include *Basis-Function Trees* (Sanger, 1991b) and the related paper (Sanger, 1991a). The *Hierarchical Mixtures of Experts* (HME) structure, is a hierarchical structure (Jordan and Jacobs, 1993) trained using Expectation Maximisation techniques (EM). Friedman used spline-based techniques to extend the CART ideas to soft splits for his *Multiple Adaptive Regression Splines* (MARS) algorithm (Friedman, 1991). Quinlan also started to work with continuous systems modelling using *Model Trees* (Quinlan, 1992). Links between wavelets and hierarchical networks (Bakshi and Stephanopoulos, 1993) have also been investigated. Banan describes a constructive hierarchical method, with local linear models, which trains many times, and partitions the input space *randomly* in the areas producing errors. The 'average' network produced by the random splits is then the result of training (Banan and Hjelmstad, 1992). (Omohundro, 1991) describes Bump- and *Balltrees* which have linear classifiers in the leaves of the tree. The first work from this thesis with hierarchical networks was the development of *Fractal Radial Basis Function Nets*, described in (Murray-Smith, 1992). The extension of this method is integrated into the local model paradigm in Chapter 5.

2.5 Learning in Local Model Basis Function Networks

The learning task for Local Model Networks is to try to adapt their structure and parameters to minimise a cost functional related to the deviation of the model from the target system. The optimisation of the parameters[8] is a linear process and relatively straightforward, while the optimisation of the number of basis functions and their position and determining a suitable level of 'locality' is a difficult non-convex problem, where *ad hoc* methods of reducing the complexity play an important role.

The description of the learning process is therefore split into the two highly interdependent stages of *structure identification* (Section 2.5.4 describes methods for introducing *a priori* knowledge into the model structure, Section 2.5.3 reviews the literature of structure identification methods) and *parameter estimation* (Section 2.5.1), which are repeated until the desired structure and parameters are found.

2.5.1 Parameter estimation in Local Model Nets

The assumption made during the parameter estimation stage is that the model structure (i.e. the basis functions and local model structures) already exists and remains fixed during the estimation phase. This thesis only deals with time-invariant processes, so the methods used are all *off-line* methods, where the entire training set is assumed to be available simultaneously for the estimation phase, as opposed to on-line or recursive methods, which assume a constant stream of new information[9]. The assumption made here is that the local models are linear in the parameters, as in equation (2.18).

The problem of parameter identification within such a framework is reasonably well understood, with a variety of efficient optimisation algorithms existing to solve the problem of optimising the parameters $\boldsymbol{\theta}$ of local models $\hat{f}_i(\cdot)$ in equation (2.17) to minimise the cost functional $J(\boldsymbol{\theta}, \mathcal{M}, \mathcal{D})$ for a given local model structure $\mathcal{M}$, where $\mathcal{M} = (\mathbf{c}, \boldsymbol{\sigma}, n_{\mathcal{M}}, \mathcal{M}_{1..n_{\mathcal{M}}})$ (i.e. the basis functions' centre locations and basis function sizes, as well as local model types) and training set $\mathcal{D} = (\psi(t-1), y(t)), t = 1..N$. Parameter optimisation for a given model structure finds the optimal cost J^*

$$J^*(\mathcal{M}, \mathcal{D}) = \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \mathcal{M}, \mathcal{D}). \tag{2.20}$$

[8] In this work the term *parameters* refers to the parameters of the local models and does not include the basis function parameters, the centres and widths, which are deemed to define the *model structure*.

[9] Section 4.3 discusses methods for iteratively extending the training set used, but for any given estimation stage the optimisation is seen as a batch process.

Weighted Least Squares estimation

In many learning situations the relative importance of the training data varies throughout the input space, either because the system spends most of its time in one particular operating regime, or because a particular aspect of the system is more interesting than others, and it is also common to have varying measurement accuracy in different areas of the input space. It is therefore important to be able to weight points in the training set to have more or less significance. The global criterion for estimation of the parameters of the model in equation (2.17) is then the *weighted least squares* cost functional,

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \alpha(\boldsymbol{\psi}_i)(y_i - \hat{y}_i)^2, \tag{2.21}$$

and the vector containing all the estimated model parameters is that which minimises J. In equation (2.21) $\alpha(\boldsymbol{\psi}_i)$ are observation weights which can be attached to each measurement. For the case where the measurement noise estimate for each training point is available, this cost functional is the chi-squared function (where $\alpha(\boldsymbol{\psi}_i) = \frac{1}{\sigma(\boldsymbol{\psi}_i)}$, where $\sigma(\boldsymbol{\psi}_i)$ is the measurement error (standard deviation) of the ith training vector). The model obtained using this functional is known as a *Markov Estimator* or a *Best Linear Unbiased Estimator (BLUE).*

Regularisation methods for complexity penalisation

The parameter optimisation for local model nets is an *ill-posed problem* as described in (Tikhonov and Arsenin, 1977). This is because there is insufficient data in the training set to reconstruct the input-output mapping uniquely, the data is usually corrupted by noise, meaning that a unique solution is impossible, and the continuity conditions are violated. To make the problem *well-posed* it is necessary to make assumptions about the smoothness of the underlying process being modelled, and the training set must have redundancy in an information-theoretic sense.

A variety of methods can be used to regularise the optimisation problem, to reduce the variance of the solution. Many neural network learning algorithms have implicitly (often unplanned!) had a regularisation effect, in that they have not found the 'optimal' (in the least squares sense) solution to the posed optimisation problem. Methods such as weight decay, stopping learning early (Sjöberg and Ljung, 1992), network pruning, learning with noise (Bishop, 1994) are all examples of *ad hoc* attempts to produce a regularisation effect. The classic regularisation method as defined in the *regularisation theory* proposed by (Tikhonov and Arsenin, 1977) is to extend the simple quadratic error cost functional to become a cost-complexity operator,

$$J(\boldsymbol{\theta}, \mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} \alpha(\boldsymbol{\psi}_i)\left((y_i - \hat{y}_i)^2 + \lambda R(\hat{f}(\boldsymbol{\psi}_i, \boldsymbol{\theta}))\right), \tag{2.22}$$

including penalising nonnegative functional $R(\hat{y})$ which includes *a priori* information such as smoothness constraints which makes the optimisation problem well-posed. This forces the optimisation to find a 'smoother' solution (λ is a small weighting, which defines the relative cost of the complexity compared to accuracy), which is likely to improve the generalisation of the network on new examples, at the cost of a slightly worse performance on the training data e.g. (Bishop, 1991) (Bishop, 1994) (Poggio and Girosi, 1990) (Girosi et al., 1993). These are, unless analytical solutions exist, too computationally expensive for problems of more than a few dimensions. More practical regularisation methods are described in Chapter 3. (Sjöberg et al., 1993) describes the use of regularisation methods in system identification, and the earlier paper (Sjöberg and Ljung, 1992) links the regularisation work to stopping training early, and discusses the number of important parameters in multi-layer perceptrons.

A priori knowledge of physical constraints on models can also be used to improve generalisation, as in (Kramer et al., 1992), and by (Röscheisen et al., 1992) who demonstrate the use of *a priori* models in training an RBF model of a rolling mill.

Using Singular Value Decomposition to estimate the local model parameters

The optimisation of the weights in RBF and Local Model Networks is theoretically a straightforward application of Linear Regression techniques, and as the optimisation problem is a linear one, the 'optimal' solution should always be found (assuming uncorrelated zero-mean noise). The regression problem can be viewed as finding the local model parameters $\boldsymbol{\theta}$ which satisfy the equation,

$$\mathbf{Y} = \mathbf{\Phi}\boldsymbol{\theta} \tag{2.23}$$

where $\mathbf{\Phi}$ is the design matrix, where the rows are defined by

$$\boldsymbol{\phi}_i = \left[\rho_1(\tilde{\boldsymbol{\phi}}_i)[1\ \boldsymbol{\psi}_{i_1} \ldots \boldsymbol{\psi}_{i_{n_\psi}}] \ldots \rho_{n_\mathcal{M}}(\tilde{\boldsymbol{\phi}}_i)[1\ \boldsymbol{\psi}_{i_1} \ldots \boldsymbol{\psi}_{i_{n_\psi}}]\right], \tag{2.24}$$

so that the design matrix $\mathbf{\Phi}$, and vector of output measurements $\mathbf{Y}$ are

$$\mathbf{\Phi} = \begin{pmatrix} \boldsymbol{\phi}_1 \\ \cdot \\ \cdot \\ \boldsymbol{\phi}_N \end{pmatrix}, \quad \mathbf{Y} = \begin{pmatrix} y_1 \\ \cdot \\ \cdot \\ y_N \end{pmatrix} \tag{2.25}$$

The task of matrix inversion is important for the optimisation process. In practical situations, however, the straightforward inversion of an information matrix is of little use, as the matrices are not square, and even if they are, the poor condition or singularity of the matrix in question makes inversion impossible. To avoid these problems, the Moore-Penrose pseudoinverse of $\mathbf{\Phi}$, $\mathbf{\Phi}^+$ is used to estimate the weights.

$$\hat{\boldsymbol{\theta}} = \mathbf{\Phi}^+\mathbf{Y} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{Y} \tag{2.26}$$

so for weighted least squares as described in equation (2.21), with $\mathbf{Q}$ a diagonal matrix with $Q_{ii} = \alpha(x_i)$, the optimal weights are:

$$\hat{\boldsymbol{\theta}} = (\mathbf{\Phi}^T\mathbf{Q}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{Q}\mathbf{Y} \tag{2.27}$$

The algorithm used in this work[10], as in many other papers, to calculate the pseudoinverse is *Singular Value Decomposition* (SVD) of a matrix of observed input data. The SVD algorithm decomposes any $N \times n_\phi$ matrix $\mathbf{\Phi}$ to matrices $\mathbf{U}$ ($N \times n_\phi$ column orthogonal), $\mathbf{\Sigma}$ ($n_\phi \times n_\phi$ diagonal) and $\mathbf{V}$ ($n_\phi \times n_\phi$ orthogonal), such that $\mathbf{\Phi} = \mathbf{U}\mathbf{S}\mathbf{V}^T$. Due to the orthogonality of the matrices $\mathbf{U}$ and $\mathbf{V}$, $\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = 1$. $\mathbf{V}$ is the matrix containing the eigenvectors of $\mathbf{\Phi}^T\mathbf{\Phi}$. $\mathbf{U}$ is made up of the eigenvectors of $\mathbf{\Phi}\mathbf{\Phi}^T$. The associated eigenvalues (σ_i) are the same in both cases, and are the squares of the singular values (s_i), i.e. $\sigma_i = {s_i}^2$ As $\mathbf{U}$ and $\mathbf{V}$ are orthogonal, their inverses are equal to their transposes. $\mathbf{\Sigma}$ is diagonal, so its inverse is the diagonal matrix containing the reciprocals of its diagonal elements (the singular values). The advantage of this decomposition[11] is that the inverse of $\mathbf{\Phi}$ is now trivial to compute, giving the pseudoinverse:

$$\mathbf{\Phi}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^T \tag{2.28}$$

where

$$\mathbf{\Sigma} = \begin{pmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & s_{n_\phi} \end{pmatrix}. \tag{2.29}$$

The method is robust because it can, within limits, cope with singular or poorly conditioned matrices. The *condition number* of a matrix gives an indication of the rank of the matrix.[12] If this is infinite the matrix is singular, and if the reciprocal of the condition number approaches the machine precision, the matrix is said to be ill-conditioned. In such cases the singular values are so small that the result is corrupted by the round off effects caused by finite accuracy arithmetic. Their corresponding columns in $\mathbf{V}$ are linear combinations of $\mathbf{x}$'s which are insensitive to the data. The $\frac{1}{s}$ elements, where s is less than a preset tolerance are zeroed, reducing the number of free parameters in the fit. Once the singular values have been zeroed, the parameters $\boldsymbol{\theta}$ solving the regression problem in equation (2.23) can be calculated,

$$\hat{\boldsymbol{\theta}} = \mathbf{V}\mathbf{S}^{-1}\mathbf{U}^T\mathbf{Y}. \tag{2.30}$$

This method of optimising the parameters is not without its disadvantages, however, which are described in Section 3.1.1. For more details on SVD see the general treatment in books such as (Golub and van Loan, 1989), or (Press et al., 1988). The use of the method in system identification applications is described in (Söderström and Stoica, 1989). The review article (van der Veen et al., 1993), and papers on the general application of the method in (Deprettre, 1988) and (Vaccaro, 1991) give further background.

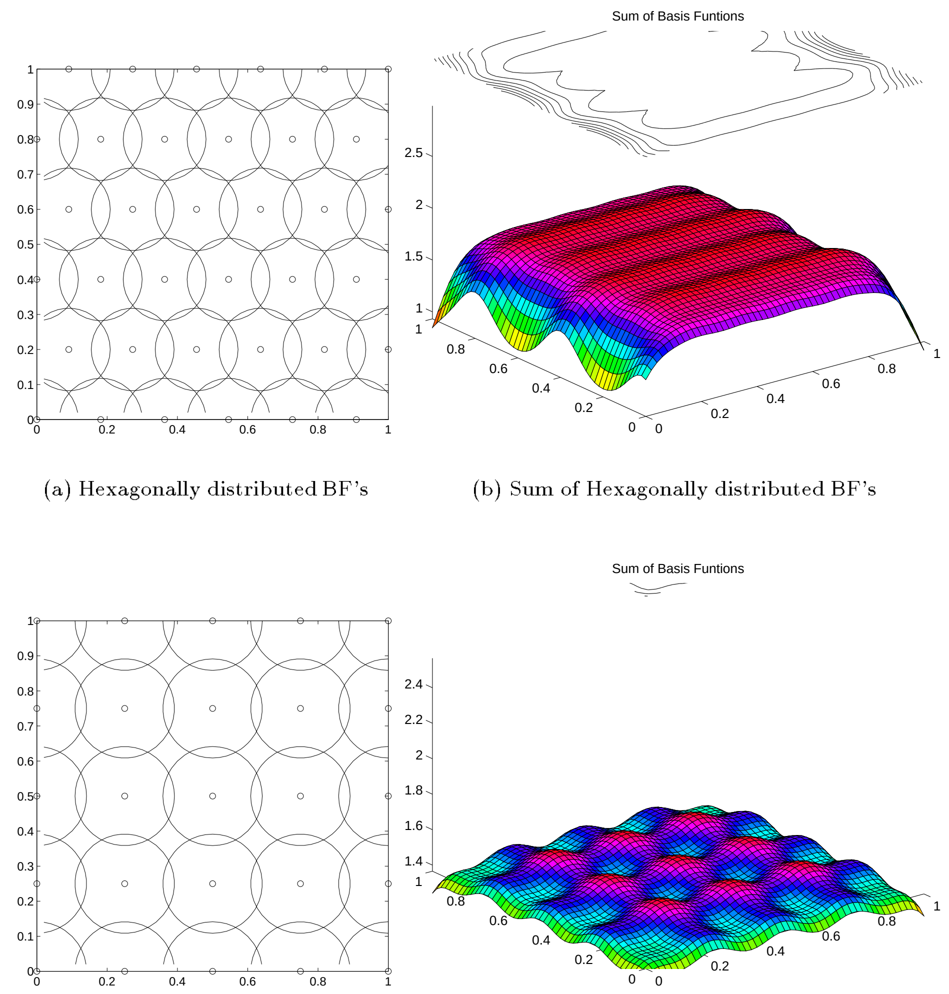
[10] The implementation of SVD used in this thesis is the MATLAB function svd().

[11] To better understand the behaviour of the original transformation $\mathbf{Y} = \mathbf{\Phi}\boldsymbol{\theta}$, the SVD of $\mathbf{\Phi}$ is a rotation of $\boldsymbol{\theta}$ in n_ϕ-space by $\mathbf{V}^T$, the components are then scaled by $\mathbf{S}$ and then rotated again by $\mathbf{U}$ to give $\mathbf{Y}$. If a vector $\mathbf{Y}$ lies in the range of $\mathbf{\Phi}$, there is a solution to $\boldsymbol{\theta}$. The solution is actually a set of solutions, as any vector in the nullspace can be added to $\boldsymbol{\theta}$ in any linear combination. SVD, however, finds the solution $\boldsymbol{\theta}$ with the smallest magnitude (see (Press et al., 1988) for details).

[12] See Section 3.1.1 for more details.

2.5.2 Uniformly distributed basis functions

The simplest method of determining the position and widths of the basis functions in a local model net is to fit a regular mesh or lattice of basis functions over the input space, as shown in Figure 2.18. This also allows the use of computationally simple methods to find active units, as



(a) Hexagonally distributed BF's

(b) Sum of Hexagonally distributed BF's

(c) Grid-like distribution of BF's

(d) Sum of grid-like distributed BF's

Figure 2.18: A lattice style distribution of basis functions in square and hexagonal forms

the net is effectively an interpolating memory. The total number of units required will rise – as the 'curse of dimensionality' would have us expect – exponentially with the input dimension. Many non-linear systems in the real world, however, have smooth nonlinearities which can be represented by relatively few units (this becomes even more relevant when the individual units are associated with more powerful local models). Also, as the system being modelled is often only active or of interest in a small region of the input space, a further saving of redundancy can be found by only placing basis functions in regions where the system operates. This demands more flexibility in the model structures than is possible with mesh-like networks, and methods for optimising the flexible structures.

2.5.3 Structure identification

The use of existing *a priori* knowledge, as described in Section 2.5.4, to define the model structure is important, but as many problems are not well enough understood for the model structure to be fully specified in advance, it will often be necessary to adapt the structure for a given problem, based on information in the training data. The optimisation of the network structure $\mathcal{M}$ is, however, a difficult non-convex optimisation problem, and is probably the most important area of research for basis function networks, if they are to be applied to demanding modelling problems where little is known about the model structure in advance.

The goal of the structure identification procedure is to provide a problem-adaptive learning scheme which automatically relates the density of basis functions and the size of their receptive fields to the local complexity and importance of the system being modelled. The desirable features of a structure identification algorithm are:

- *Consistency* – as the number of training points increases the algorithm should produce models which approximate the real process more accurately.
- *Parsimony* – the model structure produced by the algorithm should be the simplest possible which can represent the process to the required accuracy.
- *Robustness* – the model structures produced should be as robust as possible with regards to noisy data or missing data.
- *Interpretability* – the model structure produced should ideally be as interpretable as possible, given the available data, local models and basis functions.

The aim is therefore to find a model structure $\mathcal{M}$ which allows the network to best minimise the given cost function in a robust manner, taking the above points into consideration. Minimising $J^*(\mathcal{M}, \mathcal{D})$, from equation (2.20), over the possible model structures leads to the 'super-optimal' cost, using *a priori* knowledge about the process structure $\mathcal{K}_{\mathcal{S}}$,

$$J^{**}(\mathcal{D}, \mathcal{K}_{\mathcal{S}}) = \min_{\mathcal{M}} J^*(\mathcal{M}, \mathcal{D}). \tag{2.31}$$

The robustness is an important aspect, as constructive structure identification algorithms can obviously be very powerful, enabling the network to represent the training data very accurately by using a large number of parameters, but usually then leading to a high variance. The choice of model structure plays a major role in the *bias-variance trade-off* (see (Geman et al., 1992) for details about the trade-off), and this should be reflected in the cost functions $J$and J^* (from equation (2.20)) in the form of regularisation terms for J and terms which penalise over-parameterisation in the structure functional J^*.

Algorithms for structure identification from data should take into account the complexity of the target mapping, the representational ability of the local models associated with the basis functions, and the availability of data. This is a general non-convex optimisation problem, and in practice it is not possible to guarantee a general method which will provide an optimal solution to the problem for all possible learning tasks. The methods described here are optimisation techniques implicitly suited to the basis function optimisation problem.

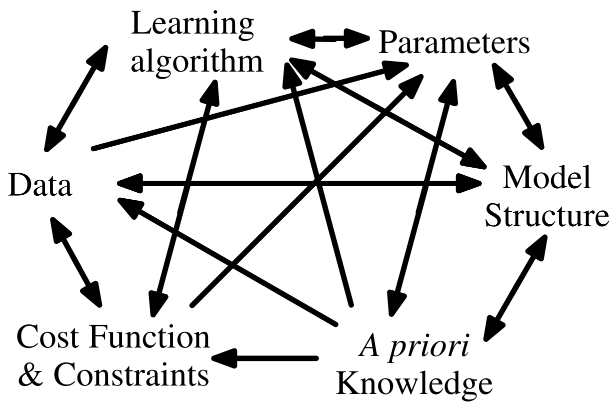


Figure 2.19: The structure learning process involves a number of complex interactions.

Structure through parameterised optimisation

A gradient descent optimisation technique for moving the centres and adapting the widths is described in (Poggio and Girosi, 1990). This is, however, reported to be a slow and unreliable technique. There is no guarantee of convergence to a global minimum and there is therefore likely to lead to variance in performance between runs on similar data. This method was applied in (Röscheisen et al., 1992) and also in (Hutchinson, 1994), where some practical guidelines for clustering are given.

Clustering basis functions

RBF researchers used methods where a fixed number of basis functions was assumed, and the structure identification task was seen as the optimisation of the centres and widths. In the papers (Moody and Darken, 1989, Sbarbaro, 1992b) clustering algorithms such as self-organising maps or k-means clustering were used to place the centres. A disadvantage of such algorithms is that they do not relate the location of the basis functions to the complexity of the function being mapped, only to the location of data in the input space. Pantaleón-Prieto *et al* (Pantaleón-Prieto et al., 1993) use a clustering routine where only the nearest unit to a given input is adapted, in order to reduce computation.

Placing centres on training data points

The disadvantage of the techniques described above is that the user must still define how many units the network should have before learning starts, but as the complexity of the system is usually not fully understood, the optimal number of units is also unknown. Some researchers developed methods which estimate the number of units from the position of the training data in the input space. Specht describes a method which has a basis function centred on every example in the training set (Specht, 1991). This is a simple technique, but one which scales up very poorly, and is not particularly robust when faced with noisy or sparse data. More promising methods use the redundancy in the training set to reduce the number of units needed to learn the desired training data. A hierarchical clustering technique based on a binary tree approach to recursively partition the input space is used in (Stokbro et al., 1990). The resulting network is a single layer net, only the partitioning process is hierarchical. The partitioning is not related to the local complexity of the system, but simply to the presence of data – a drawback of other similar algorithms. (Raipala and Koivo, 1992) describe a similar simple method for constructing a network, which is to insert a new unit whenever an input occurs which is not near the centre of any of the units' receptive fields. This is repeated in (Roberts and Tarassenko, 1994). The algorithms we used in (Murray-Smith et al., 1992) and (Neumerkel et al., 1993) can be seen as extending this type of technique by including the system complexity in the distance metric for the clustering process.

Iterative constructive techniques for gradual approximation

Another option is to start off with a simple model, to estimate its parameters, determine where the representation is still unsatisfactory and to dynamically add new models to the network. This leads to a sequence of model structures $\mathcal{M}_1 \rightarrow \mathcal{M}_2 \rightarrow \ldots \rightarrow \mathcal{M}_{n_{\mathcal{M}}}$, where $\mathcal{M}_i \rightarrow \mathcal{M}_{i+1}$ indicates an increase in the representational ability (more degrees of freedom) in the model structure followed by a parameter identification and confidence estimation stage. Constructive techniques which gradually enhance the model representation in this manner have a number of advantages. They automate the learning process by letting the network grow to fit the complexity of the target system, but they do this robustly, by forcing growth to be guided by the availability of data and the complexity of the local models. This automatically determines the size of the network needed to approximate the function adequately, while preventing overfitting. Two such constructive algorithms are described in Chapters 4 and 5, coming from the work published in (Murray-Smith and Gollee, 1994) and (Murray-Smith, 1992).

Chen *et al* (Chen et al., 1991) used orthogonal least squares for the clustering task. Their algorithm is a constructive one, which uses every training point as a candidate centre. Each time a unit is added to the network, it chooses the best candidate centre by attempting to minimise the variance in the model output due to the network parameters. The *Resource Allocation Net* described in (Platt, 1991) is a constructive algorithm, where when a pattern is presented which causes an error larger than a given threshold a new unit would be added at that point. Wynne-Jones suggests a constructive method where existing units in the network are split into two. The *Hierarchical Self-Organising Learning* (HSOL) algorithm, a hierarchical strategy for the construction of single layered basis function networks for classification is described in (Lee and Kil, 1991). Units are added to the network in a coarse to fine strategy. (Carlin, 1992) applied HSOL to modelling problems and similar coarse-to-fine ideas have been used for spline-based modelling applications (e.g. ASMOD in (Kavli, 1992)). (Fritzke, 1994) describes a constructive method based on a self-organising map framework. The LSA algorithm which splits the input space orthogonally to the axes of the input space is described in (Johansen and Foss, 1994b). The spline-based MARS algorithm (Friedman, 1991) mentioned earlier is also a gradual constructive method.[13]

2.5.4 Pre-structuring the local model net

The straightforward local model network, where the local models are simple linear models, can be viewed as a general structure which is well suited for use in modelling dynamic systems. A major advantage of the local model nets is, however, their ability to allow the introduction of *a priori* knowledge to define the model structure for a particular problem. This leads to more interpretable models which can be more reliably identified from a limited amount of observed data.

Incorporating local models based on *a priori* knowledge

The most general form of information is the expected order of the system, and the form of model to be identified (e.g. simple linear ARX models etc). If more knowledge is available, the local models could be physically oriented models, possibly with only a subset of their variables to be identified, thus allowing the engineer to easily create *grey-box* models. A generalised form would allow the designer to specify a pool of feasible local models, which could be locally tested for suitability in the various operating regimes defined by the basis functions.

In many cases, there will not be sufficient data to train the model throughout the input space. This is especially true in areas outside normal desired operation, where the model may have

[13] The *Model Merging algorithm* described in (Omohundro, 1991) attacks the problem in a different way, using a fine-to-coarse learning algorithm, where each training point is initially viewed as a model, and increasingly global models are created by merging the existing models. This has the disadvantage of being more computationally intensive than the top down methods, and will tend to overtrain where coarse-to-fine methods tend to over-generalise. Other workers have produced constructive algorithms for a variety of network types, e.g. *Cascade-correlation* (Fahlmann and Lebiere, 1990) and GAL (Alpaydin, 1991), but these structures lose the locality advantages of the basis function networks, and cannot easily introduce *a priori* knowledge, unlike local model nets.

to be very robust, and well understood. These situations can be covered by fixing *a priori* models in the given areas, and applying learning techniques only where the data is available and reliable.

Incorporating *a priori* knowledge of non-linearity into the basis functions

Locality of representation provides advantages for learning efficiency, generalisation and transparency. It is, however, very difficult to automatically find the 'correct' level of locality for a given subspace of an arbitrary problem. The problems of dimensionality can also be reduced in many systems with a large number of inputs, as there are often combinations of input dimensions which are of no interest, or which are additively or linearly related. The problem can then be decomposed, if the user already has *a priori* knowledge about the system being modelled, thus allowing the user to treat the system as an additive combination of lower dimensional sub-models. (The use of such physically based knowledge makes on-line adaptation of the system's parameters much more feasible). The statisticians have developed the theory of additive modelling techniques, e.g. (Hastie and Tibshirani, 1990, Friedman, 1991) to support such decompositions. (Hrycej, 1992) also describes similar methods for the modularisation of neural networks. Also, because of the strong links between Fuzzy membership functions and Basis Functions (see our review in (Haas and Murray-Smith, 1993), or the books (Harris et al., 1993, Brown and Harris, 1994)), the *a priori* knowledge of how best to decompose the problem could be expressed as linguistic rules with accompanying basis functions. (Bridgett et al., 1994) analyses the functionality of the MARS and ASMOD algorithms and relates them to fuzzy systems.

High dimensional Local Models, low dimensional Basis Functions

The decomposition of the input space is especially interesting for nonlinear, high order dynamic systems, as the input space is very large (and in practice such high dimensional input spaces are often impossible to fill with data), but the nonlinearity may only be dependent on a small number of the inputs. Local model nets are well suited for modelling such systems because although the system may be globally strongly nonlinearly dependent on the inputs,it may be possible to use the most important subset of the inputs to partition the input space. The system can then be locally approximated sufficiently accurately by simple (possibly linear) models which use the entire input vector, so

$$\tilde{\boldsymbol{\phi}} \subset \boldsymbol{\psi}, \dim \tilde{\boldsymbol{\phi}} < \dim \boldsymbol{\psi}. \tag{2.32}$$

In the dynamic systems' case, a dramatic reduction in the input space used for the nonlinear partition could be achieved by including only a subset of the delayed values of the inputs and state in $\tilde{\boldsymbol{\phi}}$, while all are present in $\boldsymbol{\psi}$ (i.e. the dimension of $\tilde{\boldsymbol{\phi}}$ is usually smaller than that of $\boldsymbol{\psi}$).

The well-known consequences of the 'curse of dimensionality' can be greatly reduced by defining a lower dimensional projection of the input space for the location of the basis functions and the evaluation of the distance metric (shown in Figure 2.20). This greatly simplifies the scale of task facing the structure identification algorithm.

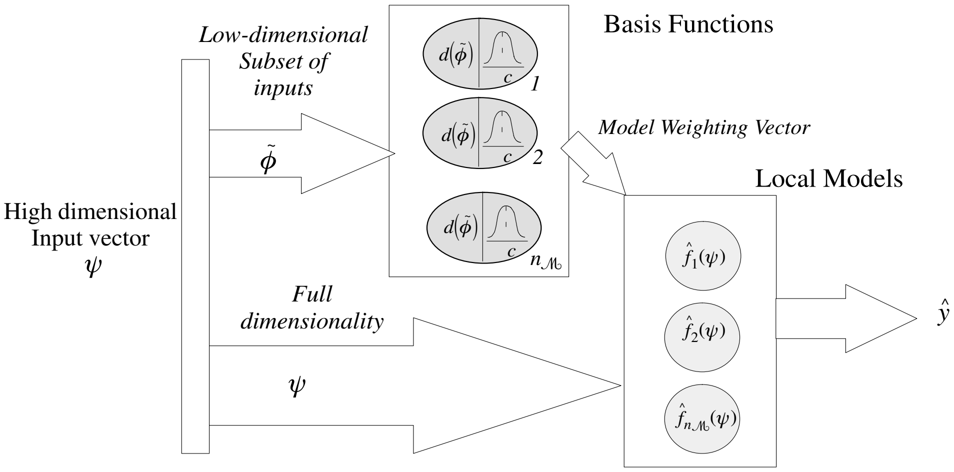


Figure 2.20: A mixed order hybrid Local Model Net system, where the operating point $\tilde{\boldsymbol{\phi}}$ has a lower dimension than the model inputs $\boldsymbol{\psi}$.

2.6 Conclusions

2.6.1 Engineering deficits of neural net solutions

The idea of creating a model of a given system by examining its behaviour, as opposed to gaining an understanding of the physical processes within the system, was not an innovation of the neural network field. The related fields of Statistics, System Identification, Cybernetics and Machine Learning all offer much support in both theoretical and practical aspects of the modelling task. Although the last decade has seen the publication of thousands of papers on neural networks, the deficiencies of many artificial neural networks for reliable use in practical applications are now becoming obvious to many working in the field. While networks like the multi-layer perceptron have been applied with success in a number of real applications, the lack of clear methods for training, analysis and validation lessen their applicability to difficult, or safety-critical projects. The 'curse of dimensionality' is a basic fact of life when producing models from data, making it necessary to introduce *a priori* knowledge–a procedure which is not well supported in multi-layer perceptrons. The lack of an engineering methodology also makes project administration in any such work more difficult, due to the unpredictability of success or failure, the variation in time needed to achieve a solution and the uncertainty about the quality of the final trained model.

2.6.2 Local Model Basis Function nets for practical problems

Local Model nets can be seen as a more general implementation of the more widely used basis function net. Local Model Nets have fewer of the engineering problems described above. The two main advantages of the architecture are:

- given local basis functions, i.e. limited overlap between models, a significantly higher level of transparency is achieved. This allows the easy local introduction of tools from other modelling paradigms (system identification, statistics etc.), and makes it easier to build *a priori* knowledge into the architecture in the form of partially or fully parameterised physical models.
- the partition also simplifies the evaluation of local confidence limits, and therefore leads to more efficient parameter and structure identification algorithms. The localised confidence estimates and constructive structure identification methods have a further advantage, as it becomes possible to automatically determine local sparsity in the training data so that more data can be demanded in active learning systems, if necessary.
- an enhanced representation, making it more suitable for modelling high dimensional and dynamic systems. Although the basis functions are still local, the more powerful local models associated with them allow the representation to be significantly more

global, without the problems seen in more powerful fully global representations such as polynomial approximations, or the curse of dimensionality in fully local methods such as RBF nets.

The local model nets seem therefore to be a promising framework for the improvement of the 'learning engineering' problems which this thesis set out to attack. The local nature of the basis functions makes it easier to develop constructive structure identification algorithms. The Local Model Nets are more interpretable than other neural network architectures, and thus allow existing modelling techniques to be more easily integrated. This makes the framework potentially very powerful, as it can benefit from the wealth of theory and experience in domains such as statistics and system identification.