Chapter 1

Introduction

1.1 Introduction

Many claims have been made concerning the importance of neural networks (NNs) as a paradigm shift in modeling both nature and the central processes of Information Technology including, most directly, Artificial Intelligence problem domains. To this stage, NNs have been applied to many and varied areas of inquiry from the control of chemical plants, through to pattern recognition, and many biological domains. Further, most proponents do not claim that NNs literally correspond to what actually occurs in the human brain, but there is a general belief of a loose correspondence between the actual computational units used and the response properties of individual neurons. Added to this, there is a belief that the inherent parallelism of NNs is consistent with brain function and that the use of fundamental numerical computations, in contrast to symbolic or declarative representations, is representative of the "hardware" of intelligent behaviour. This thesis is not aimed at challenging these claims, but at considering fundamental computational problems of NNs related to their usual representational dimensionality and model adequacy. In particular, this thesis is concerned with the problem of integrating domain- or task-specific knowledge into the very architecture of NNs.

It is important to note that most past NN formulations have a few central features in common. They are:

1. Problems are solved by the determination of weights or states in an extremely high-dimensional state space.

2. Learning, parameter estimation, and information processing are all inherently parallel.

3. No further constraints on the system are required, since the NN learns weights which are necessary and sufficient to predict behaviour.

This indicates that most applications of NNs are based upon the assumption that solutions to problems can be obtained by using generic technologies which essentially search high-dimensional state spaces, and so require *no additional knowledge* about the system under analysis. Examples of this abound in the literature, where the input and output level responses are defined by discrete nodes and their transducer functions, and at least one hidden layer is introduced. Further, NNs usually employ Supervised Learning which, in one sense, is a form of constraining the system and weight estimation processes. However, it is usually "black-box" in so far as it, *per se*, makes no assumptions about the processing characteristics and desired properties of the system not explicit in the training samples. For example, most traditional NN approaches to pattern recognition lack explicit shift, rotation and scale invariances, as the NNs are not modeled with such characteristics in mind. If such invariances arise it is due to the presence of appropriately shifted, rotated, or scaled example patterns in the training data set.

The aim of this thesis is to formalize an emerging perspective to NNs [e.g Tebelski and Waibel, 1990; Waibel, Jain, McNair, Saito, Hauptmann and Tebelski, 1991] which, whilst retaining the essentials of the NN approach, enables explicit modeling of the target system. This can result in dramatically improved generalization of classification performance to patterns not present in the training data, and representation in considerably lower-dimensional state spaces. Perhaps most importantly, model-based NNs can be constructed so that they are guaranteed to have responses which are invariant under certain transformations of the input data. Such networks can be trained with very much smaller training sets, since it is no longer necessary to provide examples of transformed versions of the input prototypes. This, coupled with the reduction in the dimensionality of the parameter space, means that training such model-based NNs is often much less computationally-expensive than the traditional alternative.

To attain these goals we first define the classical formulation, compare it to past technologies, and then develop the model-based formulation.

1.1.1 The Classical Feed-forward Neural Network

For a classical feed-forward NN, the input x_j to a given "neuron", j, is defined by

$$x_j = \sum_i y_i w_{ij} \tag{1.1}$$

and the output is defined by the logistic function

$$y = \frac{1}{1 + e^{-x}}. \tag{1.2}$$

This process is implemented in NNs with at least three layers: an input layer, and output layer and one or more hidden layers. These are connected in the form of a feed-forward architecture, as shown in Figure 1.1 (as, for example, employed by Plaut and Hinton (1987)). In classification or recognition problems, the input layer is defined by an array of nodes which constitute a sampled version of the input signal. The output layer is defined by a set of nodes each corresponding to a class, pattern type, or category. Connections $\{w_{ij}\}$ between nodes are usually restricted to layers above and below a given layer (see Figure 1.1). For a 3-layer NN with 100, 20, and 10 nodes for the three layers, respectively, this results in $100 \times 20 + 20 \times 10 = 2200$ connections. Each such connection has a weight to be estimated in accordance with the desired input-output (I-O) characteristics (Equations 1.1 and 1.2 above).

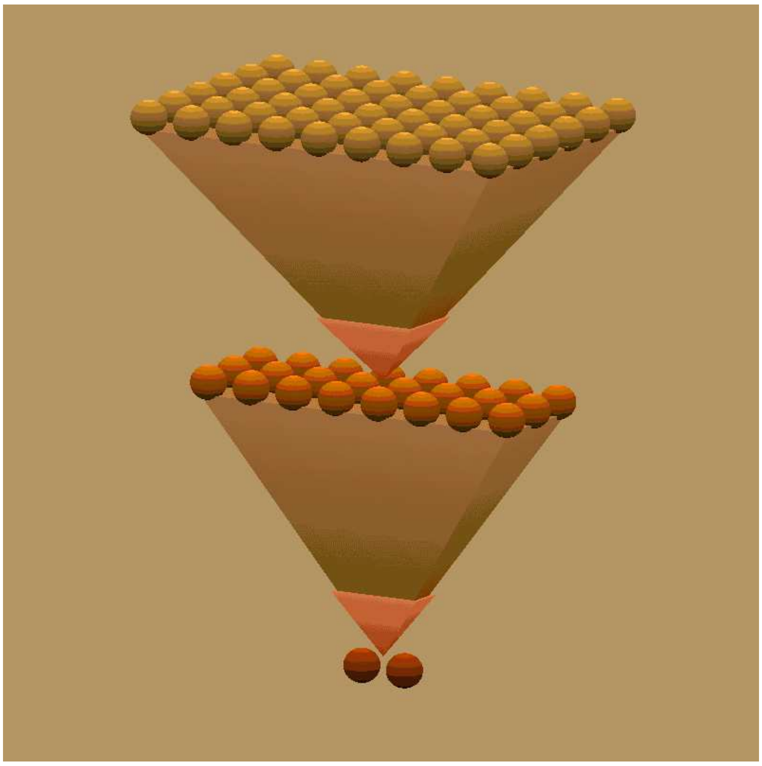

Figure 1.1: A classical feed-forward network. The weighting function between all pairs of levels is of type M.1.

For recognition or classification problems using Supervised Learning, the NN is trained to reproduce given outputs from known input examples as accurately as possible. That is, the learning problem is defined by estimating the $\{w_{ij}\}$ such that a given output error, or "cost", function E is minimized. The most commonly used cost function is the sum of the squares of the errors of the output nodes:

$$\min_{w_{ij}} \left\{ E = \frac{1}{2} \sum_{l} \sum_{c} (y_{lc} - t_{lc})^2 \right\} \tag{1.3}$$

where w_{ij} refers to all connection weights over all levels of the network, c indexes the input-output exemplar pairs, l indexes the output nodes, y_{lc} is the actual output, and t_{lc} is the desired output.

Though described in a variety of ways, learning techniques use either deterministic gradient descent (in particular, backpropagation [Rumelhart, Hinton and Williams, 1986b; Rumelhart, Hinton and Williams, 1986a; Plaut and Hinton, 1987] or stochastic relaxation methods [*e.g.* Metropolis, Rosenbluth, Rosenbluth, Teller and Teller, 1953; Kirkpatrick, Jr. and Vecchi, 1983; Aarts and Korst, 1989] to find a set of weights (a state of the network) that satisfies Equation 1.3. Solution or convergence times can be improved with various acceleration techniques [*e.g.* Fahlman, 1988].

We are not so much concerned with these search techniques but, rather, the definition of the state space being searched. The state space dimensionality is very high since it is defined by the total number of independent connection weights, which here includes every connection. Consequently, there are likely to be many local minima available to the solution technique that satisfy Equation 1.3 for the training examples, but do not characterize the problem in general. The problems associated with search time (number of iterations for convergence, *etc.*) are also significant. Further, under the classical NN perspective, no further constraints are added to such estimation problems as that would challenge the idea that relations and rules can be "discovered" automatically through the minimization or search for global minima of Equation 1.3. Here, we propose to retain the notion that states can indeed be discovered by the learning procedure, but to constrain the search procedure by modeling the network to explicitly include our knowledge of the important features of the data, or desired invariant properties.

This modeling can take a variety of forms. The network is frequently divided into a number of *modules.* These modules can be designed to perform component sub-tasks of the overall classification problem. Within a module, the weights may be constrained in some way: some weights may be set by hand; others may be constrained by a functional relationship; the module may be trained to perform a mapping independent of the classification training data. First, however, this view must be put in perspective. To this end, we will first relate the parameter estimation problem in NNs to similar problems in Signal Processing and Principal Components Analysis.

1.1.2 A Matrix View of Neural Networks

We first note that, for a given set of connections, Equations 1.1 and 1.2 can be written in matrix form as:

$$X_n = f\left\{\mathbf{W}Y_m\right\} \tag{1.4}$$

where f corresponds to the logistic nonlinear transducer (Equation 1.2), $\mathbf{W}$ to the connection matrix of size $n \times m$ (for m "source" and n "destination" nodes), and X and Y to the source and destination vectors corresponding to the pair-wise NN node layers. This form points to the essential idea that the flow of information from one level to the next corresponds to a transformation, defined by the connection matrix, which maps m-dimensional signals into n-dimensional ones, where, usually, $n < m$. The nonlinear transducer is used to map input values monotonically to the range $(0, 1)$. The net effect is that the NN procedure endeavours to discover a mapping which satisfies an overt constraint like least squares (Equation 1.3) or others.

Consequently, the best form of $\mathbf{W}$ is that of an orthogonal mapping of rank equal to the true dimensionality of the input - as measured, for example, by the rank of the input signal correlation matrix. Hidden units (units in neither the input nor output layers) correspond to components of the eigenstructure in the mappings, albeit constrained by the optimization goal. Except for the nonlinear transducer function, the use of hidden units in NNs is a way of determining the eigenstructure, or principal components, of the data. There must be at least as many hidden units (spread throughout the hidden layers) as there are non-zero eigenvalues in the input data correlation matrix, since it is necessary to span a vector space of the same dimensionality as the signal samples. The hidden units, by definition, extract the important features in the signals which, in more traditional signal processing, are extracted through the eigenvectors of the signal autocorrelation or autocovariance matrices [Ade, 1983; Ahmed and Rao, 1975].

Since many applications of NNs lie in the area of classification, it is interesting to note that, when the final output layer of such a cascaded NN is a set of nodes corresponding to classes or categories, we can interpret the system as a form of Discriminant Function Analysis. The Discriminant Analysis model is based upon the determination of decision hyperplanes which lie between sample class means and which maximize the between-class and minimize the within-class variance from their projections onto such planes. Discriminant functions are linear equations of the form:

$$z = \sum_{i=1}^{m} a_i y_i + a_0 \tag{1.5}$$

and class membership is determined by the value of the function. For example, for a two-class classification problem, class membership is determined by the sign of z. For the n-class problem Equation 1.5 generalizes to the matrix form:

$$Z = \mathbf{D}Y \tag{1.6}$$

where the n-class vector Z determines the weights associated with the data, Y, from each of the classes. $\mathbf{D}$ corresponds to the set of discriminant function coefficients which define each class. The crucial difference between NNs and classical Discriminant Function Analysis is that the decision boundries of NNs are non-linear, due to the transducer f.

Similar formulations for classification have occurred in the Pattern Recognition literature where, for example, the Least Squares Minimum Distance Classifier [Ahmed and Rao, 1975] attempts to find a mapping, in matrix form, which transforms samples in feature space into points in "decision space" (whose dimensionality corresponds to the number of classes) and satisfies the constraint that samples from the same class should be mapped as close as possible to each other, and as far away as possible from other class samples, in decision space.

However, NNs differ from these well-known past techniques, even in their traditional form, by binding the representation and processing characteristics together. Although all these techniques have similar aims and structures, the NN formulation integrates the processing characteristics with the decision processes in one network, which is represented, in general, by a set of cascaded transformations. The use of nonlinear transducers and layers of differing sizes has the disadvantage that analytic solutions are difficult, particularly since the dimensionality of the representation is of high order, but permits the network to distinguish between classes which are not linearly separable.

Our aim is to preserve this binding of *process* with *representation* (feature space), but to extend the NN philosophy to include more explicit constraints on the network geometry and connection weights. The resulting systems behave similarly to traditional NNs, but have two main advantages. First, it is possible to construct networks that are constrained to respond to features of the input data that are known *a priori* to characterize the task or to have desired invariances, rather than hoping that the training data will cause the optimization technique to find a set of weights with these properties. Secondly, this allows the dimensionality of the system to be reduced, which can reduce the chance of finding a local minimum that characterizes the training data but not the general task. We call this extension Model-Based Neural Networks (MBNN).

The use of MBNNs allows a network to be constructed in which the supervisor's knowledge of the task to be performed is used to specify, partially or completely, the roles of some hidden units, or of whole hidden layers or modules, in advance. Thus the supervisor's knowledge of which features of the training data are significant for the task is incorporated into the network geometry and connection weighting functions, serving as a constraint on the state space searched.

In this thesis, the aim is to demonstrate the utility of the MBNN approach for the specific problem of invariant pattern recognition. To situate this work, a review of existing general techniques for invariant pattern recognition is presented in Chapter 2, and of previous neural network-based approaches in Chapter 3. Later chapters present a variety of different forms of MBNN, and demonstrate their utility on a range of invariant pattern recognition problems, both real and synthetic.

Chapter 5 introduces a new invariant feature of two-dimensional contours, the Invariance Signature. It is shown that a MBNN can be constructed which calculates this (multi-dimensional) feature. This MBNN can be equipped with a classification module which uses the Invariance Signature as the basis for classification. Chapter 7 demonstrates that such Invariance Signature-based MBNNs can be employed successfully for shift-, rotation- and scale-invariant optical character recognition.