

Explanation from Neural Networks

by

Timothy A. Corbett-Clark

St Edmund Hall

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy.

Michaelmas Term, 1998

Timothy A. Corbett-Clark
St Edmund Hall

Doctor of Philosophy
Michaelmas Term, 1998

Explanation from Neural Networks

Abstract

Neural networks have frequently been found to give accurate solutions to hard classification problems. However neural networks do not make explained classifications because the class boundaries are implicitly defined by the network weights, and these weights do not lend themselves to simple analysis. Explanation is desirable because it gives problem insight both to the designer and to the user of the classifier.

Many methods have been suggested for explaining the classification given by a neural network, but they all suffer from one or more of the following disadvantages:

- a lack of equivalence between the network and the explanation;
- the absence of a probability framework required to express the uncertainty present in the data;
- a restriction to problems with binary or coarsely discretised features;
- reliance on axis-aligned rules, which are intrinsically poor at describing the boundaries generated by neural networks.

The structure of the solution presented in this thesis rests on the following steps:

1. Train a standard neural network to estimate the class conditional probabilities. Bayes' rule then defines the optimal class boundaries.
2. Obtain an explicit representation of these class boundaries using a piece-wise linearisation technique. Note that the class boundaries are otherwise only implicitly defined by the network weights.
3. Obtain a safe but possibly partial description of this explicit representation using rules based upon the city-block distance to a prototype pattern.

The methods required to achieve the last two represent novel work which seeks to explain the answers given by a proven neural network solution to the classification problem.

Acknowledgements

I would like to thank my supervisor, Professor Lionel Tarassenko, for all his help and support during the last three years. Thanks must also go to several members of the research group for lively and insightful discussions. Generous financial support was provided jointly by the Engineering and Physical Sciences Research Council and Rolls Royce under a CASE studentship.

Contents

1	Introduction	1
1.1	Feed-forward networks	2
1.2	Approaches to extracting explanation from feed-forward networks	4
1.2.1	Decompositional approaches	5
1.2.2	Pedagogical approaches	10
1.2.3	Functional approaches	12
1.2.4	Rule refinement approaches	14
1.2.5	Sensitivity analysis based approaches	18
1.2.6	Fuzzy approaches	20
1.2.7	Others	22
1.3	Discussion	23
1.4	Conclusions	25
1.5	An overview of this thesis	26
2	Principled classification	27
2.1	Classification	27
2.2	Using probabilities to classify	30
2.2.1	Classification	30
2.2.2	Novelty detection	33
2.3	Using a network to estimate class probabilities	34
2.3.1	Maximum likelihood	36
2.3.2	Overfitting, regularisation, and validation	37
2.3.3	Pre-processing	39
2.3.4	Types of variables	40
2.4	An explicitly decoupled framework	42
2.4.1	Justification for decoupling the probability estimates	43
2.4.2	Justification for decoupling probabilities from decisions	46
2.5	Summary	48
3	Making explained classifications	49
3.1	The form of an explanation	49
3.1.1	Rules based on prototypes	50

3.1.2	A user friendly format	56
3.1.3	Some examples	57
3.1.4	Incorporating categorical variables	60
3.1.5	Intersecting rules	60
3.2	Explanation vs optimal classification	61
3.2.1	The fundamental problem of explanation	62
3.2.2	Extending the decoupled framework	69
3.3	Measuring explanation	73
3.3.1	Parsimony	73
3.3.2	Effectiveness	73
3.3.3	Comparative accuracy	74
3.3.4	Comparative safety	76
3.3.5	Summary	80
3.4	The value of the explanation measures	82
3.4.1	The inherent compromises of explanation	82
3.4.2	Specifying an explaining classifier	84
3.5	Exploiting freedom in novel regions	84
3.5.1	The Karnaugh map analogy	86
3.6	Comparative safety regions	88
3.7	Summary	90
4	Linearising an MLP	91
4.1	Introduction	91
4.2	Specification	92
4.3	Solution	93
4.3.1	Main derivation	94
4.3.2	Summary of main derivation	96
4.3.3	Some preliminaries to sigmoid linearisation	98
4.3.4	Sigmoid linearisation	100
4.4	Properties and interpretation	113
4.5	Summary	118
5	Extracting explanation	119
5.1	Introduction	119
5.2	Translating a safety region into \mathcal{R}_k^1	121
5.2.1	Converting to a single linear output MLP	121
5.2.2	Simplifying to problems with only two classes	122
5.2.3	Derivation of \mathcal{R}_k^1	123
5.2.4	Discussion	125
5.3	Translating \mathcal{R}_k^1 into \mathcal{R}_k^2	127
5.3.1	Introduction	128
5.3.2	An algorithm to produce a valid \mathcal{R}_k^2	129

5.3.3	Cell/half-plane operations	131
5.4	Translating \mathcal{R}_k^2 into a rule-base	140
5.4.1	Introduction	140
5.4.2	Obtaining an effective (but unparsimonious) rule-base	141
5.4.3	Reducing the number of rules	148
5.4.4	Reducing the complexity of the rules	150
5.5	Summary	151
6	Results	154
6.1	Datasets	154
6.2	Preamble to results	155
6.2.1	Implementation details	156
6.3	Dataset 1: Heart Disease	158
6.3.1	Training the feed-forward network	160
6.3.2	Extracting a rule-base	161
6.3.3	The explanation	163
6.3.4	The effects of decreasing comparative safety	169
6.4	Dataset 2: Pima Indian	172
6.4.1	Training the feedforward network	173
6.4.2	Extracting a rule-base	176
6.4.3	The explanation	180
6.4.4	The effects of decreasing comparative safety	183
6.5	Dataset 3: Breast Cancer	183
6.5.1	Training the feed-forward network	185
6.5.2	Extracting a rule-base	189
6.5.3	The explanation	190
6.5.4	The effects of decreasing comparative safety	195
7	Conclusions and future work	197
7.1	Overview of thesis and its contributions	197
7.2	Discussion	201
7.3	Future work	202
A	Encoding categorical variables	203
B	Density estimation	205
C	Justifying a novelty threshold	207
D	Proofs of the rule fraction formulae	210

List of Figures

1.1	A typical feed-forward network classifier	3
1.2	The sigmoid activation function	3
2.1	The nature of mapping objects or events to classes	29
2.2	The structure of an MLP to estimate $P(k \mathbf{x})$	36
2.3	An explicitly decoupled classification framework	43
2.4	The relationship between $p(\mathbf{x} k)$ and $p(\mathbf{x})$	44
2.5	$P(k \mathbf{x})$ is often simpler than $p(\mathbf{x} k)$	45
2.6	A problem with partially unsupervised training a classifier	46
2.7	A problem with defining novelty with a single threshold	47
3.1	Three symmetric rule antecedent regions based on prototypes	52
3.2	Three asymmetric rule antecedent regions based on prototypes	55
3.3	The generic format of a rule for ordered variables	56
3.4	Some examples of city-block rules	58
3.5	Some examples of euclidean rules	59
3.6	Some examples of axis-aligned rules	59
3.7	The generic format of a rule for categorical variables	60
3.8	An example city-block rule-base	63
3.9	An example euclidean rule-base	64
3.10	An example axis-aligned rule-base	65
3.11	Comparing the different rule types on a hyperplane boundary	68
3.12	Comparing the different rule types on a spherical boundary	69
3.13	The benefit of allowing the rules to cross the boundary	70
3.14	An explicitly decoupled explaining classification framework	71
3.15	The rule-base is generated from probability estimates (not data)	71
3.16	The dead-band interpretation of comparative safety	81
3.17	The compromises of explanation	85
3.18	The classification boundary is meaningless in regions of novelty	87
3.19	An example of a Karnaugh map	88
4.1	Illustration of piece-wise linear bounds	92
4.2	Piece-wise linearisation of a sigmoid function	94

4.3	Strictly one-sided sigmoid linearisation configuration	105
4.4	Configuration of sigmoid linearisation through $(0, 0.5)$	107
4.5	Structure of the piece-wise linearisation algorithm	109
4.6	Example results of sigmoid piece-wise linearisation	111
4.7	Graph of linearisation error verses number of segments	112
4.8	Illustration of cells in 2D y -space	114
4.9	The cells in 3D x -space resulting from figure 4.8	115
4.10	Illustration of cells in 3D y -space	116
4.11	Different detail to figure 4.10	117
4.12	The cells in 2D x -space resulting from figure 4.10	117
5.1	Illustration of the key terms in the definitions of \mathcal{R}_k and \mathcal{R}_k^1	125
5.2	The three combinations of cell and half-plane	126
5.3	An example <i>not</i> satisfying the connected property	127
5.4	Simple algorithm to produce simplexes	130
5.5	Example of four connected cell/half-planes	130
5.6	Modified algorithm to produce simplexes	132
5.7	Numerical imprecision with connected boundary cells	137
5.8	A 2D illustration of the construction of \mathbf{y}_0	139
5.9	A summary of the rule extraction process	153
6.1	Heart disease: network output	161
6.2	Heart disease: contributions to each rule from each variable	167
6.3	Pima Indian: graph of score verses number of hidden nodes	175
6.4	Pima Indian: network output	176
6.5	Pima Indian: contributions to each rule from each variable	181
6.6	Breast cancer: graph of score verses number of hidden nodes	187
6.7	Breast cancer: network output	187
6.8	Breast cancer: contributions to each rule from each variable	193

List of Tables

3.1	Comparing the different rule types on a hyperplane boundary	67
3.2	Comparing the different rule types on a hypersphere boundary	67
3.3	Summary of the measures used to describe an explaining classifier	83
4.1	Summary of the linearisation derivation	97
5.1	Conditions for each type of rule to be in a half-plane	145
5.2	Maximum isotropic growth for each type of rule	146
5.3	Maximum anisotropic growth for each type of rule, case 1	148
5.4	Maximum anisotropic growth for each type of rule, case 2	148
6.1	Heart disease: description of the variables	159
6.2	Heart disease: 10-fold cross-validation results	160
6.3	Heart disease: effectiveness of the three large rule-bases	162
6.4	Heart disease: increasing the parsimony of the city-block rule-base	163
6.5	Heart disease: the final explanation using two rules	164
6.6	Heart disease: the performance of the two rules	165
6.7	Heart disease: prototype patterns for the two rules	168
6.8	Heart disease: explaining why a pattern is heart disease	170
6.9	Heart disease: explaining why a pattern is not heart disease	171
6.10	Heart disease: effectiveness of the three large rule-bases ($r = 0.43$)	172
6.11	Heart disease: the performance of two rules ($r = 0.43$)	172
6.12	Pima Indian: description of the variables	173
6.13	Pima Indian: 10-fold cross-validation results	174
6.14	Pima Indian: the weights and offsets in the final MLP	177
6.15	Pima Indian: effectiveness of the three large rule-bases	178
6.16	Pima Indian: increasing the parsimony of the city-block rule-base	178
6.17	Pima Indian: the final explanation using two rules	179
6.18	Pima Indian: the performance of the two rules	179
6.19	Pima Indian: explaining why a pattern is diabetes	182
6.20	Pima Indian: effectiveness of the three large rule-bases ($r = 0.67$)	183
6.21	Pima Indian: the performance of two rules ($r = 0.67$)	184
6.22	Breast cancer: description of the variables	184

6.23 Breast cancer: 10-fold cross-validation results	186
6.24 Breast cancer: the weights and offsets in the final MLP	188
6.25 Breast cancer: effectiveness of the three large rule-bases	189
6.26 Breast cancer: increasing the parsimony of the city-block rule-base . . .	190
6.27 Breast cancer: the final explanation using five rules	191
6.28 Breast cancer: the performance of the five rules	192
6.29 Breast cancer: the final explanation using two rules	192
6.30 Breast cancer: the performance of the two rules	192
6.31 Breast cancer: explaining why a pattern is cancer	194
6.32 Breast cancer: effectiveness of the three large rule-bases ($r = 0.43$) . .	196
6.33 Breast cancer: the performance of two rules ($r = 0.43$)	196

Chapter 1

Introduction

Artificial neural networks have been successfully applied to many problems in many domains. The evidence for this statement can be found in the multitude of journals and books on the subject, but example applications are to be found in finance [Ref95] (*e.g.* assessing risk for insurance and credit purposes), medicine [Bax93, TWGH96] (*e.g.* prognosis after treatment and diagnosis of disease), and industry [Leo97] (*e.g.* detection of faults and general automation). Although the language of artificial neural networks suggests biological analogies, this historical legacy is largely irrelevant to the solution of the problems mentioned above. Trying to model and understand the brain is a goal quite distinct from trying to solve specific commercial, medical, or industrial problems.

An artificial neural network (henceforth a feed-forward network) is a general-purpose parameterised function. These parameters are chosen by optimising some criterion based on data. A particularly common use of feed-forward networks is to classify objects. Example classification problems include deciding whether credit should be given to a client; diagnosing a patient with breast cancer; or deciding if an aircraft engine is faulty. In these classification applications, the input to the network is a set of features representing an object, and the output of the network is the predicted class. Data is used to optimise the network parameters such that future class predictions are as accurate as possible.

A frequent criticism of feed-forward networks is their lack of transparency (see for ex-

ample, [Tar98] page 49, [Hay99] pages 34–37, [MST94] page 221, [Wya95], [Bax95], [CHK95]). In some sense a feed-forward network is a “black box” containing uninterpretable parameters offering very little insight as to why the output should take on one value rather than another. In terms of classification problems, this means that the predicted class of a particular object has not been intuitively justified. This may be unsatisfactory for safety critical applications in aircraft, power stations, and medicine, and in those applications such as credit assessment where the refused applicant has the legal right to demand an explanation. Explanation can also help both the designer and the user to understand the data. Data visualisation techniques are frequently used for this purpose, and explanation in the form of *data description* can provide a complementary solution. Lastly, producing an explanation satisfies a basic psychological need.

This thesis attempts to address the issue of explaining the classifications produced by a suitably trained feed-forward network. The purpose of this chapter is to provide a review of the various approaches which have been suggested for explaining feed-forward networks. This leads naturally into a discussion which culminates in a list of ideal criteria. The chapter ends by providing an overview of the remainder of this thesis.

1.1 Feed-forward networks

It is useful to provide a brief description of a typical feed-forward network. Comprehensive discussions can be found for example in [Bis95], [Tar98], [Rip96], and [Hay99].

Figure 1.1 shows the structure of a common feed-forward network known as a *multi-layer perceptron* (MLP). The connections between the nodes represent the parameters or *weights* of the network. Evaluating the network proceeds from left to right. The input nodes serve only to take on the values of the features making up an input *pattern*. The hidden nodes pass a linear combination¹ of the input nodes through an *activation function*. The most common activation function is the sigmoid (or logistic) function, shown in figure 1.2.

¹This linear combination includes a additive constant, which can be absorbed into the figure by making one input node and one hidden node always assume the value of 1.0.

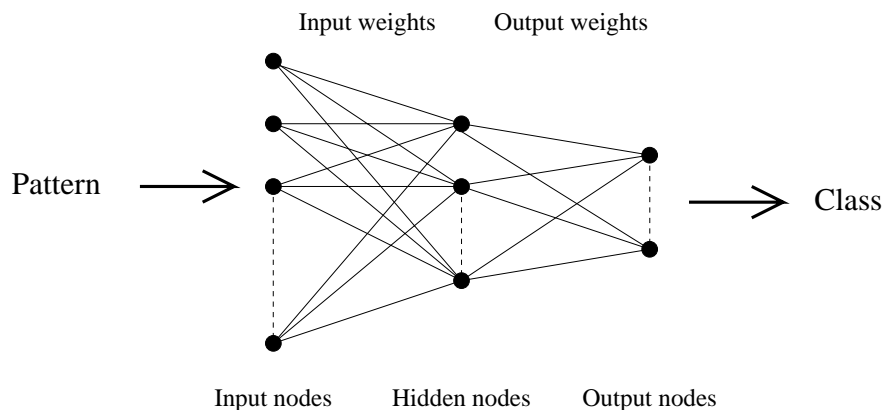


Figure 1.1: A typical feed-forward network classifier. Each solid line represents a numerical parameter (or weight) of the model, and the variability in the number of input, hidden, and output nodes is indicated by the dashed lines.

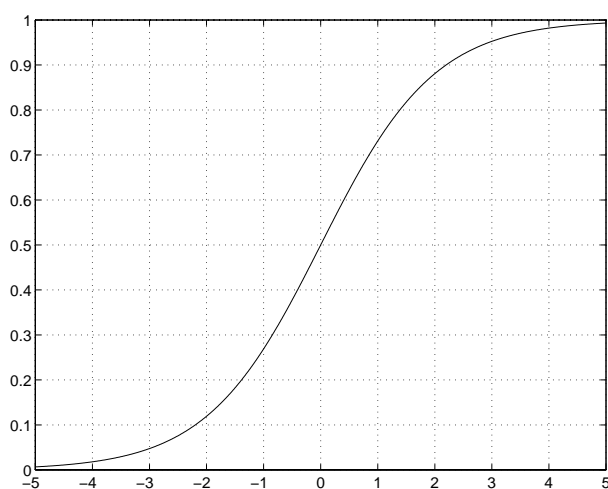


Figure 1.2: The sigmoid activation function, $f(x) = \frac{1}{1+e^{-x}}$.

Note that some feed-forward networks have more than one layer of hidden nodes, in which case each layer is evaluated in turn. Evaluating the output nodes is the same as evaluating the hidden nodes, except that a different activation function may be used. Finally, in a classification network, the values of the output nodes are interpreted as the class of the input pattern.

The weights of the network are obtained by *training*, which involves minimising some error function based on the *training data*. Such training data must be *labelled*, which means that each pattern is labelled with a corresponding class. Finally, a technique called *error back-propagation* is a means of computing the partial derivatives of the error func-

tion with respect to the weights. These derivatives are required by many optimisation (training) algorithms, the most basic being gradient descent.

1.2 Approaches to extracting explanation from feed-forward networks

Andrews *et al* have written several partial surveys of the field, [AJD95], [ACD⁺96], [AG96a], which include a taxonomy of approaches extending [CS94]. The two most commonly used dimensions are *decompositional* — where the structure of the network is used to structure a rule-base, and *pedagogical* — where extracting an explanation is seen as a learning process. Several other dimensions are mentioned, such as the need for specialised training procedures and the expressive power of the extracted rules. However these have only rarely been adopted in the literature. Listed below are the dimensions which will be used here to classify the various approaches to explanation from feed-forward networks.

1. *Decompositional*: the structure of the network is mapped directly into rules.
2. *Pedagogical*: rules are generated by observing the output of the network for different inputs.
3. *Functional*: rules are generated by observing the range of values at the output of the network in response to a range of input values.
4. *Rule refinement*: prior knowledge is first incorporated into a network structure, which is then optimised using data before the knowledge is re-extracted.
5. *Sensitivity analysis*: a means of summarising the magnitude and sign of changes in the network output due to various (small) changes in the input.
6. *Fuzzy*: fuzzy rules are extracted from the network.

There is some overlap between these dimensions. In particular, both rule refinement and fuzzy logic approaches tend to use decompositional ideas, and many researchers report

improved explanation when prior knowledge is first inserted into the network (rule refinement). In spite of this, the above categorisation provides a useful framework for the literature review presented below. The following descriptions indicate the ideas present in the field, and thus provide a background against which the research described in the later chapters of this thesis can be evaluated.

Two useful on-line resources for rule extraction from neural networks are [And98] and [Boz98].

1.2.1 Decompositional approaches

Decompositional approaches to rule extraction are characterised by a literal mapping between the structure of network and the structure of the rule-base. This invariably means that each hidden and output node becomes a single rule, although this rule may later be eliminated through combination with others. Decompositional algorithms invariably approximate each sigmoid in the network with a threshold function.

Fu

Fu, [Fu94], describes a method called *knowledgeatron* (KT), for extracting rules from an MLP type network. The algorithm works by analysing the weights connected to each hidden node in order to find combinations of inputs which make the output of a hidden node sigmoid greater (or less than) a threshold. These combinations are then written as a rule from input space to a temporary symbol representing the output of the hidden node. Repeating the process for all layers of the network produces a set of rules from input space to output space via temporary symbols. Substitution is then used to eliminate the temporary symbols. Finally, heuristics are used to increase the comprehensibility of the rules; this involves dropping rule conditions until the accuracy on the training data changes significantly.

Note that the algorithm is restricted to problems with discrete² network inputs, and makes the usual decompositional assumption of approximating sigmoid functions with step functions. Fu attempts to force the sigmoids to be saturated by effectively setting the input layer weights to be large prior to training.

Fu compared KT with Quinlan's decision tree algorithm C4.5, [Qui93]. Results on three datasets were reported: Fisher iris, hepatitis prognosis, and hypothyroid diagnosis. Classification accuracy of KT was greater than C4.5 only on the iris data, and equal on the hepatitis and hypothyroid. 5 rules explained the straightforward iris data, but the number of rules on the other two problems was not reported.

Krishnan

Krishnan, [Kri96], describes a decompositional method of rule extraction from neural networks called COMBO. This method is a systematic alternative to the heuristics used by other decompositional methods to search for input combinations resulting in saturated activation functions. Particular comparison is made to Fu's KT algorithm, to which this algorithm is very similar. However COMBO is more efficient because it sorts the weights, considers the combinations in lexicographic order, and structures the search space using a combination tree. The search is exhaustive, and so the complexity of the algorithm is exponential.

The basic algorithm is limited to discrete input variables, but Krishnan extends it to continuous inputs by using a heuristic for the input layer and the COMBO search algorithm for all other layers. Results showed that applying the method to the Fisher iris data produced four rules with the same classification error as the network.

²The differences between discrete, continuous, categorical, and ordered variables will be discussed in chapter 2. However the terminology used by the reviewed paper will be used in this chapter.

Blasig

Blasig, [Bla93], describes a back-propagation-based training scheme called Gradient Descent Symbolic (GDS) rule generation that produces networks which can be transformed into a concise set of rules. This is achieved by constraining network training with a term which tries to force as many weights as possible to zero, whilst simultaneously saturating the hidden node sigmoid functions so that they can be well approximated by step functions. Hence this approach may be interpreted as a means of constraining classification performance with explanation.

The method was tested on two problems: splice-junction recognition and prediction of interest rates. The results for the first of these compared most favourably with several other classification methods, producing a small percentage of errors with only a small number of rules. The second problem has continuous values which had to be discretised into a thermometer code representation because the algorithm only works for binary inputs. The results were incomplete since neither the final number of rules nor the class priors were reported.

Abe, Kayama, Takenaga, and Kitamura

Abe *et al*, [AKTK93], describe an algorithm for simplifying the decision process of an MLP network. Their approach is based upon the concept of extracting discriminatory information from the hidden nodes. The weights of the input layer of an MLP define a set of hyperplanes, the distances from which are assumed to provide more concise information regarding class membership than the raw data. The process attempts to eliminate the output layer by iteratively digitising the input hyperplanes and re-tuning the weights. Digitising the input hyperplanes consists of assigning a 1 (or a 0) to each hidden node such that majority of training patterns cause that hidden node to take on the value 1 (or 0). Re-tuning consists of adapting the weights to achieve these 1 (or 0) values. The net result is a set of hyperplanes which map input patterns to “digitised class vectors”. A decision tree is then used to classify the outputs of these hyperplanes (no details given).

The results on a number recognition problem showed very little change in accuracy from the original network to the extracted algorithm. No details of the explanatory power of the method were given.

Taha and Ghosh

Taha and Ghosh describe three methods of rule extraction, [TG96b, TG96a, DAC⁺96]. The first approach is strictly of the pedagogical type (see below) and uses a brute-force search over all input combinations to form the rules. It requires the input variables to be binary.

The second approach is decompositional and is based on ordering both the positive and the negative weights of each hidden node into descending order. If-Then rules with “certainty factors” are generated for each node, where the certainty factor indicates the magnitude of the output for a given input. These rules are then combined to eliminate intermediate (hidden) nodes.

The third approach is also decompositional but is not restricted to binary inputs. This is achieved by discretising each continuous variable into a set of intervals. Linear programming techniques (see for example [Fou98]) are used to find combinations of inputs which cause the output of each hidden node to be greater than a threshold. This threshold is used to set the certainty factor for the corresponding rule. Rules are then combined to eliminate intermediate nodes. Note that this third approach has similarities with the method described by Filer and Austin (see below), although the latter requires neither discretisation nor linear programming.

After extraction using any of the three methods, the rules are ordered to maximise rule-base performance on the training data.

Results are presented on three problems: an artificial dataset involving binary values *generated* from six rules; the Fisher iris problem; and the Wisconsin breast cancer problem. The first algorithm suffered from discretisation of input space, but the second and third approaches produced a small number of accurate rules which compared favourably with

other algorithms (Setiono and Liu's NeuroRule [SL96], Quinlan's decision tree approach C4.5 [Qui93], and Fu's KT [Fu94]).

Setiono

Setiono, [Set97], describes an algorithm for extracting rules from neural networks by pruning and hidden-unit splitting. First the network is pruned of any redundant weights. Then the activation values of the hidden nodes are clustered into discrete values. This enables rules to be used to describe the mapping from combinations of these discretised values to network output nodes. However the rules from the input nodes to each hidden node will be complex unless the number of non-zero weights is small. To achieve the latter condition, any such offending hidden node is split into a sub-network which is trained to be identical to the functionality of the hidden node. Rules are then obtained from this sub-network by a recursive application of the complete rule extraction process.

Results are presented on a splice-junction problem and a sonar-return problem. The first has discrete inputs, and the 16 reasonably comprehensive rules obtained have the same accuracy as the network. The second problem has 208 examples of 60 continuous inputs representing sonar returns from either a metal cylinder or a cylindrically shaped rock. In order to facilitate rule extraction, these numeric attributes were discretised into bins and then entered into the network using a thermometer coding scheme. Again, the accuracy of the network was approximately the same as that given by the 8 extracted rules.

It should be noted that network pruning was a key component of rule extraction because it dramatically reduced the number of weights in both networks: from 1220 to 16 in the splice-junction problem, and from 186 to 15 in the sonar-return problem.

Unlike the other decompositional methods described above, this algorithm does not approximate each sigmoid activation function with a step function. It also ensures an approximate equivalence between the rules and the original network by looking at the performance on the training data, both during pruning and during splitting.

Bologna

Bologna, [Bol96], describes a method of rule extraction from a type of network called the Interpretable Multi-Layer Perceptron (IMLP). This network differs from a standard multi-layer perceptron in two ways. First, the hidden layer activation functions are unit step functions rather than sigmoids. This implies that the error function is non-differentiable, so simulated annealing is used to optimise the network. Second, each hidden node is only connected to a single input node. However, each input may be connected to several hidden nodes.

It is straightforward to extract rules from such an architecture, although the number of rules can be large. To solve this problem, Bologna uses Quinlan's C4.5 algorithm directly on the output of the hidden nodes. This has the advantage of allowing the number of rules to be limited, but also means that the rules will no longer make the same classification as the network. Note that this method is similar to the one described by Abe *et al* (see above).

Results on three datasets are presented: breast cancer diagnosis (discrete inputs); Australian credit card (binary and continuous inputs); and image segmentation (continuous input variables). These required between 5 and 30 rules to achieve accuracies comparable with a standard MLP and C4.5.

1.2.2 Pedagogical approaches

Pedagogical methods of extracting rules work by observing the output of the network for different input patterns. They may also consider the weights of the network, but unlike the decompositional approach do not match the structure of the rule-base to that of the network.

Craven and Shavlik

Craven and Shavlik, [CS94] consider rule extraction as a learning problem. Two *oracles*, called EXAMPLES and SUBSET, are used to generate training examples for the rule-learning algorithm and to test the validity of hypothesised rules. The SUBSET oracle may be implemented using either Thrun's VIA algorithm, or a more efficient version of Towell and Shavlick's M-of-N algorithm. Both of these algorithms are described later.

The emphasis of the paper is in comparing the efficiency of exploring rule space using oracles, with the efficiency of an exhaustive search-based approach. To make the exhaustive search computationally feasible they restrict their experiments to networks with only a single layer (no hidden layer). In addition their algorithm only works on domains with discrete input features.

A DNA promoter dataset is used to assess the following aspects: the similarity between the network and the rules; the number and complexity of the rules to achieve a given similarity; and the number of operations required. Two different types of rule are considered: the standard conjunctive rules,

if (conjunction of categories) **then** (particular class),

and the "M-of-N" style rules used by Towell and Shavlik,

if (M of these N categories) **then** (particular class).

Various graphs show that significantly fewer M-of-N rules than conjunction rules are required to achieve a given similarity with the original network. In addition, the number of operations required by their learning approach is an order of magnitude smaller than the exhaustive search.

Tickle, Orlowski, and Diederich

Tickle *et al* describe what they call the DEDEC methodology, which is applicable to a broad class of MLP like networks. The algorithm is similar to Craven and Shavlik described above, but uses the network weights to rank and then cluster the inputs so as to further increase the efficiency of the search over possible rules.

Results are presented on two discrete problems (the MONKs problem and a mushroom classification problem) using three different network architectures (including the standard MLP). Very simple and accurate rules were generated for all problems and networks. However, both datasets are artificially generated from rules and so are intrinsically simple.

1.2.3 Functional approaches

The functional approach to rule extraction maps sets of values through the network. In the two methods described below, each set of values is the region of input space described by a rule. Thus for the rule to give the same classification as the network, the corresponding set of values at the output of the network must all indicate the same class.

Thrun

Thrun, [Thr94, Thr95], describes a method of extracting rules from multi-layer perceptrons which are guaranteed to give the same classification as the network. The method is founded on *validity interval analysis* (VIA), which employs linear programming (see for example, [Fou98]) to determine the consistency of a set of constraints on the activation values of the network. Thrun uses VIA to check if a hypothesised rule gives the same classification as the network. Such a rule is said to be *valid*. Note that VIA is only guaranteed to correctly identify all invalid rules, and may reject rules which are in fact valid.

Hypothesised rules are generated in one of two ways, referred to as specific-to-general and general-to-specific. The former starts with a degenerate rule which only classifies a single pattern. The volume of input space covered by this rule is then incrementally

enlarged until it fails the validity test (using VIA). The general-to-specific approach to rule hypothesis is to start with the most general rule possible, and to iteratively sub-divide the volume of input space covered by the rule until it passes the validity test.

Thrun gives results for his algorithm on two problems: the so-called MONKs problems, and a robot arm manipulator problem. The three MONKs problems all use discrete inputs, and the data is artificially generated from different rules. To test the VIA approach, rules were extracted from a network trained on this data. Results were satisfactory, showing that a small number of effective rules could be (re-)extracted. For the robot arm manipulator problem, the task is to find the mapping from 5 input variables (describing the configuration of the arm) to an output which indicates whether the tip of the arm is above a fixed height (the table on which the arm sits). 10 rules covered 30.2% of the volume of space within the arm's reach, whilst 10,000 rules were required to cover 84.4% of the volume.

Filer and Austin

Filer and Austin, [FA96, FSA96], describe an algorithm to extract rules from networks trained on data with continuous values. These rules take the form of axis-aligned hypercubes in input space, where an axis-aligned hypercube is the obvious extension to many dimensions of the two dimensional example of a square with sides parallel to the axis. A recursive search procedure is used to generate a set of these hypercubes which describe the region of input space assigned to each class by the network. A technique called *interval analysis* is used to map hypercubes from input space to output space and thus verify the correctness of each rule. Interval analysis is a method of mapping intervals through functions. To be more precise, the output interval thus obtained is guaranteed to be a superset of the true image of the input interval.

Results of applying this approach on two datasets are compared with two other methods. The first dataset (USER) has two continuous dimensions with 49 examples from two classes. The second dataset (PAC) has four continuous dimensions with 150 examples from each of three classes. The first of the two methods is a Multiple-Valued Logic (MVL)

approach developed from the symbolic mapping method described in [SY96]. It requires continuous data to be quantised. The second method is the decision tree algorithm ID3, [Qui79].

A large number of rules (947) were required to achieve 88% accuracy on the USER data. This compares with an accuracy of 96% achieved by the original network. On the PAC dataset, approximately 2000 rules were required to achieve close to network accuracy; reducing the number of rules to about 60 reduced the accuracy by approximately 20%. The number of rules generated by the MVL and ID3 methods was not reported.

Overall, their algorithm proved much more robust to noisy data than either the ID3 or the MVL approach. It is reasonable to suppose that this is because the neural network solution (from which the rules were extracted) was less affected by the noise.

1.2.4 Rule refinement approaches

Rule refinement approaches are characterised by the following three stages: prior knowledge in the form of rules is encoded in a network structure; this network is then optimised using labelled data; finally the rules are re-extracted from the network. Typically these approaches use the decompositional idea of forming a correspondence between the structure of the rule-base and structure of the network.

Andrews and Geva

Andrews and Geva, [AG96b, AG96a], describe a method of rule extraction (RULEX) which works on Constrained Error Back Propagation (CEBP) networks. These networks are constructed from activation functions called Locally Responsive Units (LRU), which are themselves constructed from the difference between two offset sigmoid functions. Each LRU forms a “ridge” in input space, and the weights of the network are constrained such that these ridges are axis-aligned. CEBP networks are designed for incremental constructive training, which means that new local response units are added when no further

significant decrease in error is possible. Andrews and Geva explain that this allows semantics to be attached to the hidden nodes and knowledge to be inserted into the network. After training, the RULEX algorithm is used to determine the active range of each ridge and map this range into the constraints of a rule describing an axis-aligned hypercube in input space. Heuristics are used to remove redundant rule conditions and redundant rules. Results are given from applying the RULEX algorithm to three datasets: the Fisher IRIS data, Mushroom, and a Heart disease dataset. Small numbers of rules with a high accuracy are reported.

Towell and Shavlik

Towell and Shavlik, [TS91, TS92], describe an algorithm for extracting rules from Knowledge-Based Artificial Neural Networks (KBANN). The rule extraction stage of the refinement algorithm obtains so called “M-of-N” style rules from the network. These rules are of the form

if (M of the following N antecedents) **then** (particular class)

which have several advantages over the more usual conjunctive rules

if (specified antecedents) **then** (particular class).

The first advantage is of compactness, since M-of-N style rules describe several cases simultaneously. The second advantage is that these type of rules have an intuitive correspondence with the network weights: the combination of any of the M antecedents ensures a particular activation function is relatively saturated. The method for discovering these combinations is based on grouping similar valued weights into clusters, replacing these clusters of weights by their mean, eliminating insignificant groups, re-optimising the off-

sets³, approximating the sigmoids by step functions, and finally expressing the result as M-of-N rules. Note that only problems with binary variables are considered.

Two key assumptions are made to justify the rule-base-network correspondence. First, the semantics of the hidden nodes are assumed to remain unchanged by training. Second, the process makes unquantified changes to the functionality of the original network, which means that there is no guaranteed relationship between the explanation and the neural network.

On two molecular biology problems, the authors demonstrate that their algorithm produces more accurate and comprehensible rules than other methods (C4.5, LINUS, EITHER, and SUBSET). However they observe that their algorithm is limited to networks which have been pre-structured with rules, and hence to problems where such *a priori* knowledge is available.

Tresp, Hollatz, and Ahmad

Tresp *et al.*, [THA92] describe a type of network which consists of normalised basis functions. These basis functions may be of any of a number of types, but Gaussian functions are used in their study. The network is pre-structured by combining a set of logical if-then expressions obtained from an expert. The network has a probabilistic interpretation which enables the likelihood function to be used as a criterion for training the network from data. This training proceeds in one of four modes: *forget*, where gradient descent or Expectation Maximisation (EM) is used to optimise the parameters; *freeze*, where the initial configuration is frozen, but new units are added if a large discrepancy between prediction and data occurs; *correct*, where each parameter is penalised for deviating from its initial value; and *internal teacher*, where a penalty is added in terms of the function mapping rather than in terms of the function parameters. Thus these four possibilities preserve the rule-based knowledge in different ways and by differing amounts.

³The offsets of the network are sometimes called *bias weights*.

After training, the network is pruned of all unnecessary weights and then re-interpreted as rules. Pruning is vital to ensuring that the final rules are as concise as possible.

Results are given on some data where the task is to predict the housing price in a Boston neighbourhood. A network of 4 basis units corresponding to 4 rules gives results which are approximately 10% better than a decision tree method (CART, [BFOS84]). They conclude:

Rule extraction provides a quantitative interpretation of what is “going on” in the network, although, in general, it is difficult to define the domain where a give rule “dominates” the network response and along which boundaries the rules partition the input space.

Goodman, Higgins, Miller, and Smyth

Goodman *et al* [GHM92], describe a network architecture which estimates the posterior probability of classes as a function of discrete input data. The network consists of 3 layers of nodes: an input layer with a node for every value of every variable; a hidden layer where each hidden node forms the binary conjunction of some subset of the input nodes; and an output layer with one node per class, which forms a function of the weighted sum of all hidden layer nodes and gives an estimate of the posterior probability of that class.

The effect of this structure is a collection of rules of the form

$$Rule_j: \mathbf{if} (x_1 = X_1^a, \dots, x_n = X_n^z) \mathbf{then} y = Y^i \mathbf{with strength} w_{ij}$$

The connections between the input and hidden layers are formed using a combination of information theoretic and minimal descriptive length⁴ concepts to balance the trade-offs between complexity and accuracy. The weights between the hidden and output layers are then estimated by assuming that values of the hidden nodes are conditionally independent given the class (often known as *naive* or *idiot's Bayes*).

⁴The method of minimal descriptive length provides a trade-off between the complexity of a solution and its performance.

The performance of the above model is compared with that of a standard multi-layer perceptron (MLP) and a first order Bayes classifier⁵. Five datasets were evaluated: LED digits, a boolean function, congressional voting, diagnosis of breast cancer, and protein secondary structure. All three classifiers had comparable accuracy except the Bayes classifier which was significantly worse on the Boolean function.

It is not clear whether the rules for the first three problems are comprehensible, but 11 rules were produced for the breast cancer problem and 194 for the protein data.

1.2.5 Sensitivity analysis based approaches

Sensitivity analysis of a network is a means of assessing the magnitude (and direction) of change in the output for various (small) changes to the input. The idea has been used by a number of researchers, both to produce graphs which provide a qualitative explanation of the network, and to produce rules. For continuous input variables sensitivity analysis is a means of summarising the partial derivatives of the network.

Saito and Nakano

An early use of sensitivity analysis was the PDP diagnostic system, [SN88], which uses a method called *relation factors* to assess the effects on the network output (disease) from individually changing different inputs (symptoms). Two types of relation factor are described: the first considers the effects of switching a single symptom on when all other symptoms are off; the second averages the effect of changing a particular symptom in all patterns. It is shown that these relation factors agree with the relation factors used by doctors.

Saito and Nakano then describe a related method for extracting rules which is based on searching over the different combinations of (non-continuous) input variables. The search is a greedy procedure which considers the effects of changing individual input compo-

⁵A first order Bayes classifier makes the assumption that the distribution of the input components are conditionally independent given the class.

nents in turn. In addition, only those symbol combinations which are meaningful to the problem domain are considered. Thus the method extracts rules guaranteed to give the same classification as the network, although not all rules will be found because of the limited depth of the search.

Applying the system to a network trained on a muscle-contraction headache problem generated 443 rules.

Goh and Wong

Goh and Wong, [GW96], have used sensitivity analysis to extract rules from networks trained on patterns with continuous valued input features. They define a Point Sensitivity Index (PSI) for of each input variable i ,

$$\text{psi}_i(x_i) = \sum_n |f(x_1^n, \dots, x_i, \dots, x_d^n) - f(x_1^n, \dots, x_i + \delta, \dots, x_d^n)|$$

where the sum is over all patterns and δ is some small value. Peaks in the graphs of each $\text{psi}_i(x)$ are then (loosely) used to write down rules. The authors use some artificial datasets to demonstrate their approach.

Baxt

Baxt, [Bax93, Bax92] describes a method similar to the relation factors used by Saito and Nakano. Histograms are used to show the quantitative effects on network output resulting from the single-variable perturbation of training patterns. On the myocardial infarction problem studied by Baxt, these histograms were strongly bi-modal. This suggests that changing certain input variables has two characteristically different effects, depending on the value of the other variables. Baxt comments that these relationships are quite different from the traditional linear manner by which clinical decisions are made, and although they “often do not appear to make a great deal of clinical sense, they must impart significant

power to network performance in view of the high degree of accuracy with which the network has been shown to function”.

Plate, Bert, Grace, and Band

Plate *et al*, [PBGB98], describe some methods for visualising the function computed by a neural network. This visualisation assists in identifying interactions in the fitted model, and extends the work of Baxt to continuous inputs by adapting the graphical plots of generalised additive models⁶.

1.2.6 Fuzzy approaches

Numerous papers have been published on using Neuro-Fuzzy hybrids to form classifiers. For example, [Hay90] describes a method which maps hidden nodes to fuzzy rules (and so is another example of the decompositional approach), and [MP95] describes a connectionist expert system model based on a fuzzy version of the multilayer perceptron (another example of using a network for refinement). Two further methods of interest are described below.

Buckley, Hayashi, and Czogala

Buckley *et al* [BHC92] prove that feed-forward neural networks and fuzzy expert systems are functionally equivalent. In other words, a feed-forward network can always be constructed to approximate a fuzzy expert system with arbitrary accuracy, and conversely, a fuzzy expert system can always be constructed to approximate a feed-forward network with arbitrary accuracy. Note that to achieve this arbitrary accuracy may require an arbitrary number of nodes or rules.

However this equivalence does not imply (and the authors do not attempt to show) that a neural network can be explained by translating it into a fuzzy expert system.

⁶A generalised additive model (GAM) takes a linear combination of non-linear functions of each input component.

Benítez, Castro, and Requena

Benítez *et al* [BCR97] show that by using a particular membership function and fuzzy logic operator, a neural network can be made exactly *equal* to a fuzzy logic system. In some respects this is a decompositional approach because each hidden node is mapped into a single fuzzy rule. However these fuzzy rules are not approximations but *interpretations* of the functionality of each hidden node.

The key steps in this interpretation are as follows. Each hidden node computes the value $f(\sum_i w_i x_i)$, and this may be interpreted as the fuzzy rule

$$\mathbf{if} \sum_i w_i x_i \text{ is } A \mathbf{ then } y \text{ is } 1.0, \quad (1.1)$$

where A is the fuzzy membership function formed from the sigmoid function f . In the notation of fuzzy logic, this is written $\mu_A(a) = f(a)$. By defining a suitable fuzzy logic operator, $*$, equation (1.1) may be identically expressed

$$\mathbf{if} w_1 x_1 \text{ is } A * \dots * w_n x_n \text{ is } A \mathbf{ then } y \text{ is } 1.0. \quad (1.2)$$

The authors discuss the semantics of this notation, describing the symbol $*$ as an interactive-or operator.

Defuzzification consists of using the network output weights to form a linear combination of the degrees with which each rule fires. This output is used to determine the class of any pattern. Note that the output node is assumed to be linear.

Results are presented on the Fisher iris data, where a network with a single hidden node is described with the corresponding rule. Note that the interpretation of these rules relies critically on an intuitive understanding of the semantics of the interactive-or operator.

1.2.7 Others

Sejnowski and Rosenberg

Sejnowski and Rosenberg, [SR87], attempted to understand the functioning of a network trained to convert english text into speech (NETtalk). In particular they aimed to understand the coding methods used by the network by studying the patterns of activation amongst the hidden nodes. Their network had 80 hidden nodes and they discovered that on average about 20% of the hidden nodes were highly activated for any given input. A simple plot of these activations proved uninformative, but a hierarchical clustering technique applied to the average activations over letter-to-sound correspondences was more successful: the vowels and consonants were completely separated, and for the vowels the next most important variable was the letter whereas the consonants were clustered more according to the similarity of the sounds. Note that this analysis did not explain why the network produced certain outputs. However it did give some insight into the problem domain.

Sanger

Sanger, [San89], describes a technique called *contribution analysis* for deriving the responsibilities of individual hidden units in implementing the input-output mapping of the network. The contribution from a hidden unit is defined as the product of the weight connecting that unit to an output unit multiplied by the activation of that unit when a particular pattern is applied to the network. Thus there is a contribution associated with every hidden unit, output unit, and input pattern. This forms a three-dimensional space which was visualised by finding the principle component of two-dimensional slices. Sanger explains why this is more informative than looking at either the weights or the activations separately.

The idea was used to explain a network trained to convert english text into speech (NETtalk). Only very general conclusions were reached, such as “redundant output units are handled

by identical patterns of hidden units, and the amount of responsibility taken on by a hidden unit is inversely proportional to the number of hidden units.”.

1.3 Discussion

An immediate conclusion from the above review is that almost all algorithms produce an explanation consisting of if-then rules. Those approaches which do not (*e.g.* sensitivity analysis) are useful for gaining insight into the solution, but do not explain why a particular pattern was assigned to a particular class. Hence it will be assumed that rules are the most suitable means of explaining a network.

A vital aspect of explanation using rules is the size of the rule-base. Clearly hundreds of rules make the explanation incomprehensible, and a dozen or so is probably an acceptable maximum. However several of the algorithms described above required many more rules than this in order to cover a reasonable proportion of the data. Related to the size of the rule-base is the expressive power of individual rules, and one would expect a small number of more expressive rules to be equivalent to a larger number of less expressive rules. For example, M-of-N style rules are efficient at describing many conditions on categorical variables in an intuitively acceptable manner, and Craven and Shavlik [CS94] demonstrated that fewer M-of-N rules were required than ordinary conjunctive rules.

It is noticeable that most of these rule extraction methods do not guarantee a fixed relationship between the generated rule-base and the network. In this sense they are *new* learning algorithms which are *based* on a feed-forward network, and cannot therefore be said to explain the original network. This is particularly true of the decompositional approaches because they approximate sigmoids by step functions and modify the weights. These transformations make unknown changes to the functionality of the network. Indeed [MSS91] has shown that there are Boolean functions which can be computed by a network of sigmoid functions but not by a network of threshold functions with the same topology.

Rule refinement algorithms do not need to guarantee an equivalence between the network and the rules. This is because their remit is to produce a rule-based classifier which uses prior rule knowledge and labelled data. Feed-forward network ideas may be used, but this does not imply that this approach can be used to explain how a standard feed-forward network classifies. There are many problems for which prior rule-based knowledge is unavailable.

The two functional approaches (Thrun, and Filer and Austin) both produced rules which were guaranteed to make the same classification as the network. However they both needed a large number of rules in order to classify a significant fraction of the data. In contrast, those decompositional approaches which extracted axis-aligned rules all generated far smaller rule-bases which still covered most of the data. This is not a coincidence, and it will be shown later in this thesis that there is an inherent trade-off between ensuring that the rules make the same classification as the network, and maximising the fraction of data classified by the rules.

Many of the decompositional and pedagogical algorithms relied on searching over combinations of inputs. This is clearly a computational burden since there are $\prod_i^N n_i$ combinations for N variables where the i^{th} variable can take one of n_i values. Most of the algorithms overcome this problem by trimming the search, but this has the disadvantage of either producing rules which do not cover all the patterns, or of producing rules which do not necessarily make the same classification as the network.

Many of the algorithms do not work on datasets with continuous variables. Solving this problem by discretisation has at least two disadvantages: first, it constrains the network to work with quantised values which have not been optimally chosen for classification performance; and second, it may (depending on the coding required by the algorithm) significantly increase the number of input dimensions. This has significant consequences on the algorithms which use a search-based approach.

Lastly, probability theory is not part of the framework for most of the reviewed papers. This is surprising given its suitability both for describing the uncertainty in mapping pat-

terns to classes, and for understanding the problems of learning a mapping using a sample of data from some underlying population.

1.4 Conclusions

The conclusion from the above discussion is that there is a clear need for a rule extraction method which:

1. is expressed within a probability framework,
2. generates a rule-base which has a known relationship with the network,
3. produces a small number of comprehensible rules which cover a significant fraction of all data,
4. operates on a feed-forward network of a standard type which has been trained using a standard procedure,
5. is not restricted to datasets with only categorical variables.

This thesis describes such a method. Note that some problems do not require all of these properties. For example, a statistical model is not required if the data is fundamentally non-random and can be enumerated in entirety. Alternatively, the data may not contain any continuous variables, or a standard feed-forward network technique may be sub-optimal because it does not take advantage of prior rule-based knowledge. However there is a significant class of problems which would benefit from rule extraction algorithms satisfying the above properties.

In terms of the classification scheme adopted in this chapter's literature review, the method is closest to the functional approach. A probabilistic framework (property 1) together with the piece-wise linearisation of a standard network function (property 4) are used to overcome the problem of generating a known relationship with the rule-base (property 2). An expressive type of rule, which is not restricted to categorical variables (property 5), then enables a small rule-base to classify most of the data (property 3).

1.5 An overview of this thesis

Chapter 2 provides a summary of how feed-forward networks can be used for classification in a probabilistic framework. Chapter 3 then describes how explanation can be added within this framework, and how such an approach can be used to understand and control the compromises inherent in explaining a feed-forward network classifier. This chapter also introduces the expressive and general family of rules based on prototypes which will be used later. Chapter 4 describes a technique for efficiently obtaining a piecewise linearisation of a very common network called a multi-layer perceptron (MLP). This linearisation is the starting point for the sequence of methods, described in chapter 5, which ultimately lead to the generation of a rule-base with the properties defined above. The results of applying the complete procedure to three real datasets are given in chapter 6. Finally, chapter 7 contains a discussion and critique, and suggests some directions of research which may lead to further improvement.

Chapter 2

Principled classification

This chapter provides an introduction to classification using feed-forward networks, the purpose of which is to give the background to the concepts presented in later chapters. As a consequence of this, the amount of detail given is inherently slightly uneven. A thorough treatment can be found in many books, for example [Fuk90, Bis95, Rip96, Tar98].

In addition to providing background, this chapter also argues for the explicit decoupling of a classification system into separate components. The effect of this decoupling is to give each component an independent specification, thus eliminating or at least making explicit the interactions which may be concealed within a more homogeneous framework. This explicit decoupling also underpins the approach described in this thesis to producing an explanation from a feed-forward network classifier.

2.1 Classification

Classification problems have the following generic properties. A set of measurements or *features* are gathered on an object or event. These features are concatenated to form a feature vector, and so if there are d features then this feature vector or *pattern* belongs to an d dimensional *input space*. Objects or events are therefore represented by corresponding patterns. Depending on the discriminatory quality of the features however, different

objects may or may not be mapped into significantly different patterns. It is assumed that each object or event can be assigned (by an expert) to a discrete class, although it is not assumed that all experts will agree. This association between objects and classes is carried through into an association between patterns and classes, and so classified objects are represented by classified or *labelled* patterns.

The generic classification problem can now be stated as follows: replace or pre-empt the expert by finding the best function to predict the class of an object or event from its corresponding (unlabelled) pattern. This function is estimated from *training data*, which is simply a set of expertly labelled patterns.

There are two primary factors which make the classification problem ill-posed. The first factor is due to the non-invertible mapping from objects to patterns. Different objects can map to the same or very similar patterns, implying that information is lost in the encoding of objects as patterns. The second factor derives from the multivalent concept of grouping objects into classes. Objects cannot always be unambiguously assigned to a single class. Hence different experts, or the same expert at different times, may assign different classes to the same object. How these primary factors relate to the ill-posed nature of the classification problem is summarised in figure 2.1.

The effect of these factors is that many patterns are not associated with a single class. One way to visualise the problem is to see all the training set patterns as coloured points in input space, with colour used to indicate the expert labels. Those patterns with a definite class will form clusters of points of the same colour. The effect of patterns belonging to ambiguous classes will be to cause these regions to blur into each other, forming a less uniformly coloured overlap. Fundamentally difficult problems are characterised by considerable overlap; easy problems have well separated regions.

There is a third way in which the classification problem can be said to be ill-posed. The classification function is estimated from a set of labelled patterns populating input space. It is highly likely that new (test) patterns will be different from all those in the training set, and so the classification function must be capable of interpolation from the training

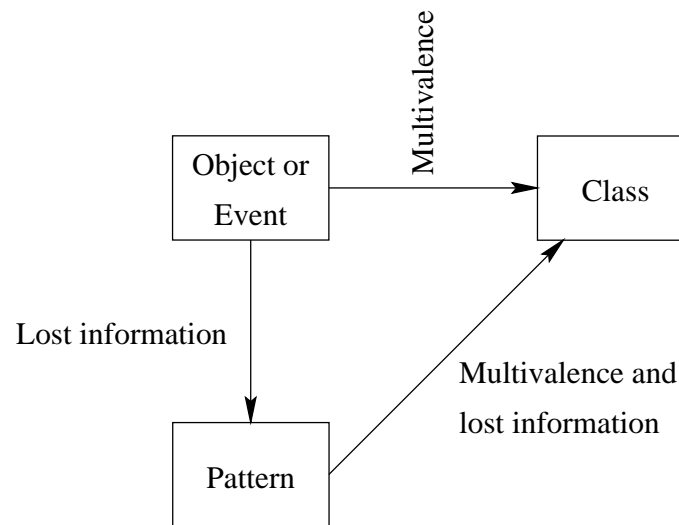


Figure 2.1: The classification problem of predicting a single class from a pattern is made difficult both by the multivalent nature of assignment of objects to classes, and by the information lost in mapping objects to patterns.

data. However the classification function should not necessarily reproduce the class label of every training pattern because, as explained above, these labels are not necessarily definitive; an expert re-labelling a pattern may not always decide on the same class. Note that there are also other factors such as measurement noise and labelling errors which increase class ambiguity.

In its most simple form, a classifier takes a pattern and returns a class. However there are possibly two types of pattern which should not be assigned to any class. First, a test pattern might be so different from all the training patterns that no reliable classification can be given. There is not usually any reason to believe that the classification function can extrapolate into regions of input space which are unpopulated by training data. Second, a test pattern may not definitely belong to any single class. The reasons for this are the same as the factors which make the problem ill-posed.

It may be argued that the first of these reasons not to classify is much more important than the second. Whilst it may be acceptable to classify patterns arbitrarily when the model indicates they do not clearly belong to any particular class, it is definitely not acceptable to classify completely new patterns for which the model is invalid. Thus the approach

used in this thesis is only to classify test patterns which are sufficiently similar to those seen during training; all other patterns are flagged as *novel* by a *novelty detector*.

2.2 Using probabilities to classify

It should be clear from the previous description of the classification problem that probability and statistics provide the most appropriate framework within which to express the uncertainty between objects, patterns and classes. This field is based on sound theoretical concepts and has many powerful techniques which have been successfully applied across a variety of problem domains. Neural networks should and can build on these techniques.

2.2.1 Classification

From a statistical point of view, the optimal classification is to assign the most probable class to an object. This will be referred to as *Bayes classification*, although strictly speaking the Bayes classification is more general and includes both a loss function and an expression of doubt [Rip96]. Before any information on a particular object is known, the probability of that object belonging to class k is given by the *prior* probability $P(k)$. However, given the feature vector \mathbf{x} representing an object, the probability of that object belonging to class k is given by the *posterior* probability $P(k | \mathbf{x})$. Suppose a classifier $d(\mathbf{x}) \in \{1, \dots, K\}$ is required to solve a K class problem, then the probability that $d(\mathbf{x})$ gives the correct class is

$$\alpha(d, \mathbf{x}) = P(d(\mathbf{x}) | \mathbf{x}). \quad (2.1)$$

The Bayes classifier d_B maximises this probability¹, and can be defined according to:

$$d_B(\mathbf{x}) = \operatorname{argmax}_k P(k | \mathbf{x}) \quad (2.2)$$

¹In the event of any ties this definition of the Bayes classification rule is not unique. However this is generally unimportant and ties can be broken arbitrarily.

The *accuracy* of a classifier d is equal to the expected probability of making the correct classification,

$$\begin{aligned}\alpha(d) &= \int \alpha(d, \mathbf{x})p(\mathbf{x})d\mathbf{x} \\ &= \int P(d(\mathbf{x}) | \mathbf{x})p(\mathbf{x})d\mathbf{x},\end{aligned}\tag{2.3}$$

where $p(\mathbf{x})$ is the unconditional probability density for pattern \mathbf{x} . The Bayes classification rule also optimises accuracy:

$$\max_d \alpha(d) = \alpha(d_B).\tag{2.4}$$

However the Bayes classifier is independent of the unconditional density distribution of patterns because it assigns the most probable class to each pattern irrespective of the probability of that pattern occurring. Hence as expressed by equation (2.2), the optimal classification of *individual* patterns should be considered the defining property.

Accuracy is a useful measure which summarises one quality of a classifier. However it is an *average* measure, and averaging can hide important effects. Consider a two-class problem for which one class is relatively rare: let the prior probabilities of patterns from each class occurring be $P(1) = 0.9$ and $P(2) = 0.1$. A classifier which assigns every pattern to class 1 will have an accuracy of 0.9, but this mis-represents the quality of such classifications. One possible solution is to quantify the per-class accuracy. The probability of the classifier d making the correct classification of pattern \mathbf{x} associated with an object labelled as class k can be written

$$\alpha_k(d, \mathbf{x}) = P(d(\mathbf{x}) | \mathbf{x}, k).\tag{2.5}$$

The *class conditional accuracy* is then defined to be the expected probability of making the correct classification of objects labelled as belonging to class k .

$$\alpha_k(d) = \int P(d(\mathbf{x}) | \mathbf{x}, k)p(\mathbf{x} | k)d\mathbf{x}.\tag{2.6}$$

The class conditional accuracies of the example classifier given above will give a good indication of the problem since $\alpha_1(d) = 1.0$ and $\alpha_2(d) = 0.0$.

The accuracy and class conditional accuracy of the Bayes classifier are maximal; they give an upper bound for the performance of a classifier. This limit depends on the difficulty of the problem, which is primarily a function of the amount of overlap between the classes in input space.

Practical issues

Optimal classification can only be made if the posterior class probabilities are known. Unfortunately these probabilities are rarely available and most practical classifiers use, either explicitly or implicitly, *estimates* of the posterior class probabilities. Neural networks are no exception and can be used either to classify directly or to estimate the probabilities from which a classification can be decided. It will be seen that the latter approach has many advantages and is therefore adopted in this thesis. The estimated posterior probabilities are written $\hat{P}(k | \mathbf{x})$, and equation (2.2) is then used with these estimated probabilities to make the classifier $\hat{d}_B(\mathbf{x})$.

The accuracy of an estimated Bayes classifier $\hat{d}_B(\mathbf{x})$ is unlikely to equal the accuracy of the true Bayes classifier $d_B(\mathbf{x})$. Even with an estimate of the unconditional probability density, $\hat{p}(\mathbf{x})$, it is unlikely that the corresponding versions of equations (2.3) and (2.6) will have analytic solutions. Therefore in practice the integrals are replaced by a sum over patterns which have been implicitly and independently sampled from $p(\mathbf{x})$ and labelled with samples from the posterior class distribution $P(k | \mathbf{x})$. In other words, the integrals are replaced by a sum over independent test patterns.

$$\begin{aligned}
 \alpha(d) &\approx \frac{1}{N} \sum_{n=1}^N \alpha(d, x^n) \\
 &= \frac{1}{N} \sum_{n=1}^N P(d(x^n) | x^n) \\
 &\approx \frac{\#\text{“correct” classifications}}{N} = \hat{\alpha}(d). \tag{2.7}
 \end{aligned}$$

Similarly,

$$\begin{aligned}
 \alpha_k(d) &\approx \frac{1}{N_k} \sum_{n=1}^{N_k} \alpha_k(d, x_k^n) \\
 &= \frac{1}{N_k} \sum_{n=1}^{N_k} P(d(x_k^n) | x_k^n, k) \\
 &\approx \frac{\#\text{“correctly” classified patterns labelled class } k}{\#\text{class } k \text{ patterns}} = \hat{\alpha}_k(d). \quad (2.8)
 \end{aligned}$$

Note that in both of these derivations there are two approximations, and that each of these approximations is due to the substitution of an unbiased estimator². Hence the expected value of $\hat{\alpha}(d)$ is $\alpha(d)$ and the expected value of $\hat{\alpha}_k(d)$ is $\alpha_k(d)$.

2.2.2 Novelty detection

A Bayes classifier assigns a class to every pattern. However there are strong reasons for not attempting to classify test patterns which are significantly different from those seen previously. A novelty detector is a function which takes a pattern \mathbf{x} and returns a verdict of normal or novel; novel patterns should not be classified since by definition the system has not seen a sufficient number of such patterns before. Note that there are alternative uses for novelty detectors which will not be considered here. For example if one class occurs extremely rarely, it might only be detected by dissimilarity from the class which represents normality.

The relevant probability for novelty detection is the unconditional density function, $p(\mathbf{x})$, which models the distribution of patterns. One possible definition of normality is the *smallest* region of input space which encloses all possible patterns from the distribution $p(\mathbf{x})$. However such a definition of normality is only satisfactory if the unconditional distribution has tails which quickly reach zero. Most real-world distributions have tails which only gradually decrease to zero, and this causes the region of normality as defined above to include patterns which only rarely occur. This is a problem because the purpose of the novelty detector is to filter out rare patterns.

²By definition, the expected value of an unbiased estimator is the true value.

One possible solution is to define normality as the smallest region of input space which encloses $100r\%$ of patterns, where r is typically close to 1.0. Appendix C shows that this is equivalent to the region of space for which $p(\mathbf{x})$ is larger than a single threshold, t . Thus novelty can be defined according to

$$p(\mathbf{x}) < t \quad \equiv \quad \text{pattern } \mathbf{x} \text{ is novel} \quad (2.9)$$

where t typically has a very small value. Normality is simply the converse of novelty, and so the set of normal patterns \mathcal{N} may be defined

$$\mathcal{N} = \{\mathbf{x} \mid p(\mathbf{x}) > t\}. \quad (2.10)$$

Note that thresholding $p(\mathbf{x})$ has commonly been used to differentiate novel from normal patterns, [Sil96].

Practical issues

In practice, the true distribution $p(\mathbf{x})$ is unknown and so an estimate $\hat{p}(\mathbf{x})$ is substituted into equation (2.9). In addition, the threshold t cannot usually be found by analytical means, but it may be estimated using order statistics [Ric95]. In short, N patterns are sampled from $\hat{p}(\mathbf{x})$ and sorted into decreasing values of $\hat{p}(\mathbf{x})$. The expected value of the $(N \times r)^{th}$ order statistic is an estimate of t , and the variance of this estimate decreases with increasing N . See also [NCCR⁺97].

2.3 Using a network to estimate class probabilities

The previous two sections have shown how classification and novelty detection may be decided from estimates of posterior class probabilities and unconditional densities. This section provides a very brief summary of how feed-forward networks may be used to estimate class probabilities. Similar concepts also apply to estimating unconditional density

but analogy will be sufficient and no details will be required (see [Sil96] for a thorough introduction to density estimation). Note that only one approach to feed-forward network based probability estimation is described here; there are several alternative methods which are not mentioned.

Of the several different types of feed-forward network, the two most common are multi-layer perceptrons (MLP) and radial basis function (RBF) networks. Both of these networks can be used to estimate posterior probabilities, but it will be seen that there are good reasons for preferring the MLP when implementing the approach to making “explained classifications” described in this thesis.

Feed-forward networks are conveniently parameterised general purpose functions which have many applications other than classification. Different applications have different requirements and this has led to several variations in the form of the output of the network function. It is clear that if a function is to estimate class probabilities, then the outputs should always lie in the interval $[0, 1]$. In addition, if the classes are mutually exclusive then these outputs should also sum to unity. These two conditions are guaranteed by the so called *softmax* output function. The equation for the class probability estimates provided by a two layer³ MLP using a softmax output function is given below.

$$\begin{aligned}
 y_j &= \sum_k W_{jk}^{\text{IN}} x_k + c_j^{\text{IN}} \\
 z_i &= \sum_j W_{ij}^{\text{OUT}} \text{sig}(y_j) + c_i^{\text{OUT}} \\
 \hat{P}(k | \mathbf{x}) &= \frac{e^{z_i}}{\sum_i e^{z_i}}, \tag{2.11}
 \end{aligned}$$

where

$$\text{sig}(y_j) = \frac{1}{1 + e^{-y_j}}. \tag{2.12}$$

This may be visualised as the structure shown in figure 2.2. The matrix \mathbf{W}^{IN} and vector

³Only networks with two layers will be considered in this thesis. Many texts report that two layers are sufficient, both in theory and in practice. See for example [Rip96, Bis95]

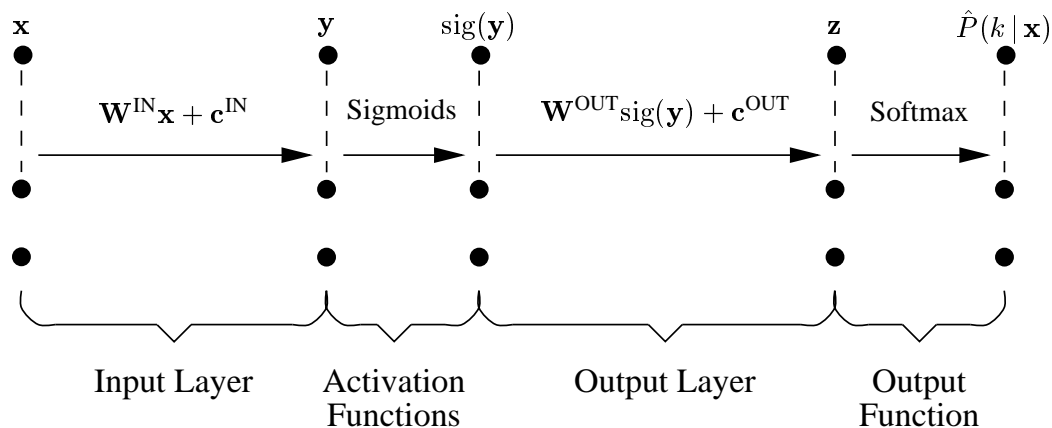


Figure 2.2: The structure of a multi-layer perceptron (MLP) which takes a feature vector \mathbf{x} and estimates the class probabilities $P(k | \mathbf{x})$.

\mathbf{c}^{IN} are called the input weights and offsets⁴ respectively. These are used to form a linear combination y_j of the features x_k . These y_j form the inputs to a number of *hidden nodes*; the output from hidden node j is a non-linear *sigmoid* function $\text{sig}(\cdot)$ of its input y_j . The output from all the hidden nodes are then linearly combined using the output weights and output offsets, \mathbf{W}^{OUT} and \mathbf{c}^{OUT} , to form the inputs z_i to the output nodes. Finally, these z_i are normalised using a softmax output function so that the outputs are in the $[0, 1]$ interval and sum to unity.

The input and output weights and offsets are the parameters of the network, and hence the parameters of the posterior probability distribution. In other words, the underlying model for the posterior probability distribution is the MLP function. Given an arbitrary number of hidden nodes, this function can be shown to be a universal approximator [Jon90, BL91]. This suggests that MLP networks impose relatively few constraints, and in this sense can be described as data-driven model-free estimators.

2.3.1 Maximum likelihood

The task of training a feed-forward network is as follows. Given a number of labelled patterns, find the parameter values which result in the MLP providing the best estimate

⁴The offsets are sometimes referred to as bias weights.

of the underlying posterior distribution $P(k | \mathbf{x})$. This problem can be solved using the statistical method of *maximum likelihood* [Ric95], which in this context seeks to maximise the probability of observing the labels on the training data as a function of the network parameters,

$$L = \prod_n \hat{P}(k^n | \mathbf{x}^n), \quad (2.13)$$

where k^n is the class label of the n^{th} training pattern \mathbf{x}^n . Since the logarithm function is monotonically increasing, maximising this likelihood is equivalent to minimising the following error function:

$$E = -\frac{1}{N} \sum_n \ln \hat{P}(k | \mathbf{x}^n). \quad (2.14)$$

This is a non-linear optimisation problem in the parameters of the network, for which there exist many possible algorithms [PTVF92]. Many of these algorithms require the first derivatives of the error function to be calculated, and another advantage of many feed-forward networks (including MLP's) is the efficiency with which these first derivatives can be computed. The standard derivations of these algorithms may be found in many standard texts, for example [Bis95].

2.3.2 Overfitting, regularisation, and validation

An overfitting network is one which has captured details of the training data which are not characteristic of the whole population; it is a consequence of an overly flexible model. Hence a symptom of overfitting is a network which performs well on the training data but considerably less well on new data.

The problem of overfitting can be reduced either by increasing the amount of training data, or by limiting the complexity of the network function. The latter can be achieved either by reducing the total number of network parameters or by constraining the freedom of these parameters. The total number of parameters is a function of the number of input

and hidden nodes, but note that reducing the number of input nodes involves reducing the number of features in a pattern. The freedom of the parameters can be constrained or *regularised* by adding a penalty term, Ω , to the error function which becomes large when the network function becomes complex. Hence training a regularised network involves minimising the *penalised* error,

$$E_{\text{penalised}} = E + \Omega. \quad (2.15)$$

A typical penalty term is the average squared second derivatives of the parameterised function [Hay99]. However when the function is a feed-forward network such penalty terms are analytically intractable, and a popular and simple alternative is called *weight decay*:

$$\Omega = \mu_1 \sum_{ij} (W_{ij}^{OUT})^2 + \mu_2 \sum_{jk} (W_{jk}^{IN})^2 + \mu_3 \sum_i (c_i^{OUT})^2 + \mu_4 \sum_k (c_k^{IN})^2, \quad (2.16)$$

where the μ_i may be all equal. Weight decay has a probabilistic interpretation of a parameterised (in μ_i) prior belief on the complexity of the function. These parameters therefore allow the amount of regularisation to be controlled.

Testing for overfitting is equivalent to assessing whether the complexity of the network function is justified by its performance on unseen data. This is called *validation* and is based on randomly splitting the training data into a *training* set and a *validation* set. For a given number of network parameters and amount of regularisation, the training set is used to minimise the error function and the validation set to assess the performance. A set of parameters which causes the training set to be overfitted will give a poor performance on the validation set, and the complexity of the network function can then be reduced by decreasing the number of network parameters and/or increasing the amount of regularisation.

Thus it can be seen that obtaining a final solution involves an optimisation within an optimisation. Network complexity is optimised by maximising the performance on a

validation set, but this optimisation is over a set of networks whose parameters have all been optimised by minimising penalised error on a training set.

Note that using validation has not completely eliminated the problem of overfitting because the final network could be overfitting the validation set. Variations on the basic validation scheme can reduce this possibility even further. For example, v -fold validation averages the results from v different training and validation sets. This also has the advantage of making more efficient use of the data available for training.

Finally, validation sets are not independent because they have been used in the optimisation process. This implies that the performance on validation sets is *not* necessarily indicative of the performance on future unseen patterns. An independent *test* set is required to provide an unbiased estimate of how well the network will perform on new patterns.

2.3.3 Pre-processing

Data pre-processing means applying a transformation to all patterns before they are passed to the feed-forward network. This transformation can be used to encode problem domain knowledge, but it is also often used simply to reduce the number of input dimensions. This dimension reduction reduces the number of parameters in the network, which helps to prevent the network function from overfitting the training data. However the disadvantage is that any dimension reducing transformation is a type of projection, and projections lose information. If useful information is lost then the overlap between classes will increase, and this must necessarily decrease the accuracy of the class predictions. There is therefore a potential trade-off between preventing overfitting and reducing the accuracy of the network.

Pre-processing can also be given a regularisation interpretation, and this is particularly clear if the pre-processing transformation is linear. The most common linear pre-processing transformation is called *principal component analysis* (PCA), which projects all patterns onto a linear subspace containing as much of the data's variance as possi-

ble [Bis95]. After the network has been trained on the projected data, the linear pre-processing transformation can be incorporated into the MLP input weights and offsets. This produces a new MLP which can be considered to have been trained with constrained (*i.e.* regularised) parameters. More practically, composing the linear pre-processing transformation with the input layer of the MLP has the advantage that the network will work directly on (new) un-projected patterns.

Training data is often normalised to have zero mean and unit variance in order to avoid problems with the non-linear optimisation algorithm used to train the feed-forward network. This normalisation is also a linear transformation and so can be incorporated into the input layer of an MLP after training. As above, this has the advantage that the network will work directly on (new) unnormalised patterns.

2.3.4 Types of variables

There are primarily two types of variables which make up feature vectors. Variables such as temperature and frequency are naturally *ordered* since it makes sense to say one value is larger than another. In contrast, *categorical* variables have no such ordering. Examples include eye colour (which might be a factor in predicting skin cancer), house location (a factor in predicting insurance risk), occupation (for targeting mail shots), type of purchase (for fraudulent use of credit cards), previous medical history *etc.*

Ordered variables are naturally encoded as a single number, but a different approach must be used with categorical variables. Suppose that a single number was used to denote one of three categories as 0, 1, or 2. This coding implies an ordering where none exists, and this affects the network by requiring a more complex network function if, for example, the first and third categories would ideally have very different estimated posterior probabilities from the second category. Since complexity needs to be penalised to prevent overfitting, a better solution would be obtained by re-ordering the codes for the first, second and third categories as 0, 2, and 1 respectively. However the implied ordering effect can be completely eliminated by using codes which are equally distant from each other.

Encoding categorical variables

The most common method of encoding categorical variables for input into a feed-forward network is called 1-of- n coding. For n categories, a vector of n components is used to represent category i by a single 1.0 in the i^{th} position, with all the remaining components of the vector equal to 0.0. As required by the defining property of categorical variables, this coding does not imply any ordering on the categories since all codes are the same distance from each other.

However this scheme has the disadvantage that all the 1-of- n codes lie in the hyperplane, $\mathbf{1}^T \mathbf{x} = 1$. This hyperplane is an $n - 1$ dimensional sub-space, which indicates that $n - 1$ dimensions are sufficient and probably necessary to encode n categories. Using n dimensions increases the number of inputs to the network, which requires more weight parameters. These extra parameters increase the complexity of the network, an effect which should be avoided since one of the causes of overfitting is using too complex a model.

In the statistical literature [VR97], 1-of- n coding with an additional code at the origin (called *contrasts*) has been used to encode $n+1$ categories in just n dimensions⁵. However the codes are then no longer equi-distant from each other.

The ideal solution requires that n codes of $n - 1$ dimensions should be the same distance from each other. Appendix A shows that the columns of the following $n - 1$ by n matrix have such a property.

$$\mathbf{T}_n = \begin{pmatrix} -\frac{\alpha_2}{1} & \alpha_2 & 0 & 0 & \dots & 0 \\ -\frac{\alpha_3}{2} & -\frac{\alpha_3}{2} & \alpha_3 & 0 & & 0 \\ -\frac{\alpha_4}{3} & -\frac{\alpha_4}{3} & -\frac{\alpha_4}{3} & \alpha_4 & & 0 \\ \vdots & & & & \ddots & \vdots \\ -\frac{\alpha_n}{n-1} & -\frac{\alpha_n}{n-1} & -\frac{\alpha_n}{n-1} & -\frac{\alpha_n}{n-1} & -\frac{\alpha_n}{n-1} & \alpha_n \end{pmatrix} \quad (2.17)$$

⁵The motivation for using 1-of- n and a code at the origin comes from solving linear regression problems. Using just 1-of- n encoding means that the linear regression cannot be solved using a matrix inversion because the 1-of- n codes lie in an $n - 1$ dimensional sub-space.

where

$$\alpha_k = \sqrt{\frac{k-1}{2k}}. \quad (2.18)$$

To be precise, the Euclidean distance between any two columns is 1.0, the magnitude of every column is α_n , and the columns sum to zero.

The motivations to use this solution are as follows: it is simple to implement; it has no disadvantages over 1-of- n encoding; and it may have some of the advantages given above. Note that as with any pre-processing, possible advantages are only gained during the *training* of the network. It may be convenient for the test patterns to use the more usual 1-of- n encoding, in which case a \mathbf{T} matrix for each categorical variable can be applied to the input of a network. Since the input to the network is via the matrix \mathbf{W} , applying an appropriate \mathbf{T} matrix for each categorical variable simply produces a new input matrix. This hides the fact that a 1-of- n encoding was not used during training from all interfacing software which uses the trained network.

2.4 An explicitly decoupled framework

The previous sections have described the essential aspects of the classification problem from a statistical perspective. This has led naturally to a system of four components: estimating unconditional density, detecting novelty, estimating posterior class probabilities, and making a decision on class. Figure 2.3 shows how these components are used at run-time to classify test patterns. Note that each of these components has a specification which is independent or *decoupled* from the others.

However feed-forward networks are commonly used as classifiers in such a way that some (or even all) of these components become merged. This has several disadvantages such as reducing system flexibility and obscuring specifications. However the biggest disadvantage is the lack of posterior probability estimates which are so useful in understanding the problem. The final two sections of this chapter justify the explicit decoupling of these

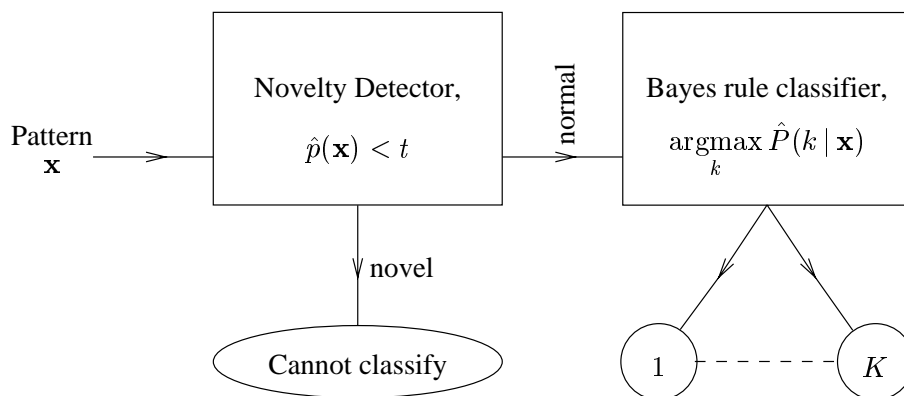


Figure 2.3: An explicitly decoupled classification framework.

components. The problems caused by merging the estimation of the unconditional density $\hat{p}(\mathbf{x})$ with the estimation of the posterior class probability $\hat{P}(k|\mathbf{x})$ are reviewed first, followed by the problems caused by merging these probability estimates with the classification/novelty decisions.

2.4.1 Justification for decoupling the probability estimates

In general, the class posterior probability $P(k | \mathbf{x})$ and the unconditional density $p(\mathbf{x})$ are independent in the sense that knowledge of one does not necessarily imply knowledge of the other. In terms of an individual pattern, knowledge of the unconditional density for that pattern does not necessarily indicate the class to which it is most likely to belong. Similarly, knowledge that a particular pattern almost certainly belongs to class 1 does not necessarily indicate anything about the probability of that pattern occurring. Of course, there is a relationship between $P(k | \mathbf{x})$ and $p(\mathbf{x})$ via the class conditional densities $p(\mathbf{x} | k)$ and prior probabilities $P(k)$,

$$P(k | \mathbf{x}) = \frac{p(\mathbf{x} | k)P(k)}{p(\mathbf{x})},$$

$$p(\mathbf{x}) = \sum_i p(\mathbf{x} | k)P(k). \quad (2.19)$$

However this only forms a single constraint because $P(k | \mathbf{x})$ equals $p(\mathbf{x} | k)P(k)$ normalised by $p(\mathbf{x})$, and $p(\mathbf{x})$ is just the sum of these $p(\mathbf{x} | k)P(k)$ terms. It is this normalisa-

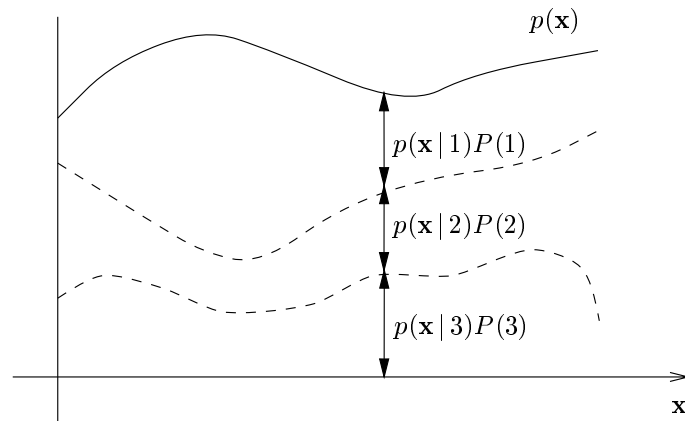


Figure 2.4: Illustration of the single constraint between $p(\mathbf{x})$ and $P(k|\mathbf{x})$.

tion which connects $P(k|\mathbf{x})$ with $p(\mathbf{x})$. Figure 2.4 demonstrates this single normalisation constraint graphically.

Given the above relationship it can be argued that estimating the class conditional probability densities $p(\mathbf{x}|k)$ directly would have the advantage of only requiring a single probability estimator. Together with the prior probabilities $P(k)$ which are usually estimated with simple methods, such a model can be used to provide estimates of both $P(k|\mathbf{x})$ and $p(\mathbf{x})$. However this approach does not often work well in practice simply because estimating density is so much harder than estimating class probability. Part of the reason for this is that the class conditional densities $p(\mathbf{x}|k)$ are usually more complex than the posterior class probabilities $P(k|\mathbf{x})$, as shown in figure 2.5. Hence more flexibility is required by an accurate density estimator than an accurate class probability estimator, and this flexibility makes overfitting extremely likely. As usual, the strength of this effect increases rapidly with increasing number of dimensions.

Overall, a system constructed from the solutions to the two decoupled problems (estimates of $P(k|\mathbf{x})$ and $p(\mathbf{x})$) is likely to perform better than the classifier constructed from the solution to the single class conditional density problem (estimates of $p(\mathbf{x}|k)$). With such a strategy, the difficulties of density estimation are restricted to novelty detection rather than being applied to both novelty detection and classification.

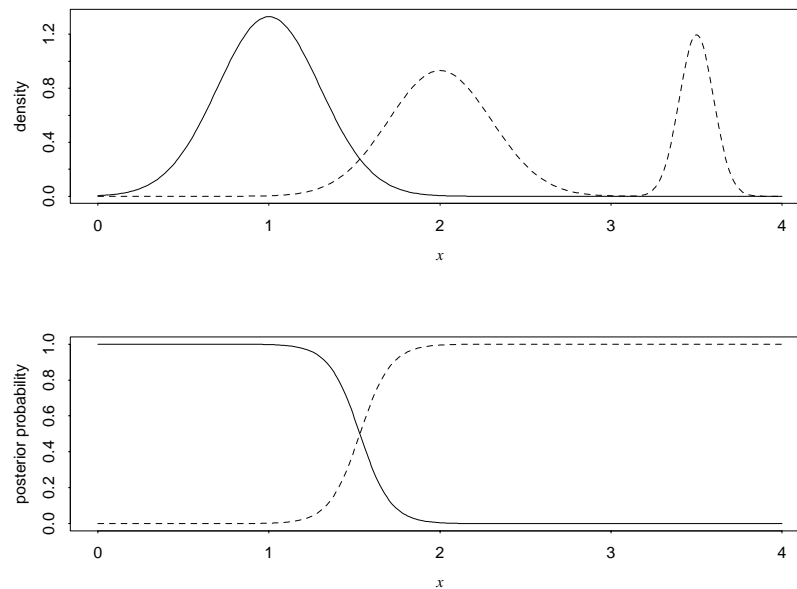


Figure 2.5: One dimensional example showing how the posterior class probabilities can be much simpler than the corresponding class conditional densities. Reproduced from [Rip96] with kind permission from the author.

Unsupervised learning

Although there is no necessary connection between $P(k | \mathbf{x})$ and $p(\mathbf{x})$, so called unsupervised learning based on cluster analysis effectively assumes the existence of a helpful link. For example, the input layer of a RBF network is often trained using the training data without using the class labels [Hay99]. In effect the input layer is chosen for unconditional density estimation. However the output layer is still optimised to estimate the class probabilities. This has the advantage that large amounts of unlabelled data can be used to train the input layer, but only a small number of labelled patterns are required to optimise an output layer of relatively few parameters.

However this advantage is gained at the cost of assuming that the data is clustered in such a way that the resulting input layer is helpful in the estimation of posterior class probabilities⁶. This is an assumption which may not hold: consider a simple example of

⁶One interpretation of the RBF network architecture is to consider the input layer as a pre-processing or feature extraction layer, and the output layer as a single-layer network estimating posterior class probabilities from these derived features.

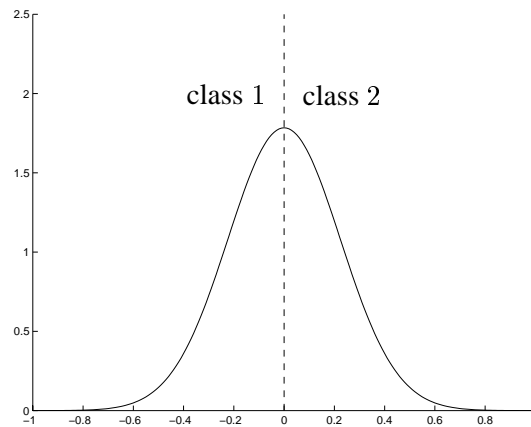


Figure 2.6: A one dimensional Gaussian centred at the origin. All patterns to the left of the origin belong to class 1, and all patterns to the right belong to class 2.

data distributed as a single one dimensional Gaussian centred at the origin (figure 2.6). Let all patterns to the left of the origin belong to class 1 and all patterns to the right belong to class 2. The optimal unconditional density function is obviously a single Gaussian, and this can be optimally implemented using an RBF network with only a single (Gaussian) basis function. However the output from this basis function gives no information about class membership, hence such a network would be useless for classification. In contrast, any RBF network with two basis functions equi-distant from the origin can be used to obtain perfect classification, but then the input layer only forms a very poor estimate of the unconditional density distribution. Note that it is not intended for the above to imply that an RBF network is an inferior classifier in comparison to other types of feed-forward network; rather that interpreting the network as an estimator of both unconditional density and posterior class probability may involve making certain assumptions about the underlying distributions.

2.4.2 Justification for decoupling probabilities from decisions

In the statistical approach to classification, decisions are separated from probability estimation: Bayes' classification rule defines an optimal classification given the class probabilities, and a single threshold defines an optimal novelty detector given the unconditional density. However, these optimal decisions will only be optimal with respect to the chosen

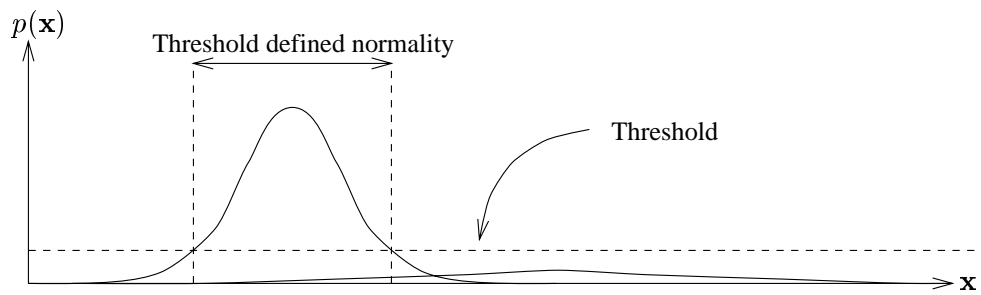


Figure 2.7: Illustration showing that despite satisfying an optimality criterion, a single threshold may not be optimal for detecting novelty.

criterion. In other words a different criterion should sometimes be optimised. For example, a single threshold may give rise to a poor novelty decision if one class has a much larger prior probability and is localised in input space, and the other class has a small prior probability with a more spread out distribution (see figure 2.7). A single threshold will therefore incorrectly flag many more patterns from the second class as novel, [THCB95].

In terms of classification *per se*, the Bayes classification is always the optimal decision. However, in the next chapter it will be seen that there are compelling reasons to make sub-optimal (in terms of Bayes classification) classifications because only then may it be possible to “explain” a reasonable number of classification decisions. The control of this sub-optimality is only possible if estimation of the class probabilities has been decoupled from the classification decision.

Finally, there is one more advantage in decoupling probability estimates from decisions: the probability estimates may have uses other than deciding novelty or class. For example, the unconditional density function can be used to fill in or *impute* missing values if some patterns are incomplete; and the class probabilities can be used to indicate the certainty of individual classifications and perhaps flag particularly ambiguous patterns (when the posterior class probabilities are very similar).

2.5 Summary

This chapter has described the statistical framework within which classification problems can be solved using feed-forward networks. It was shown how the optimal classification decision can be made from estimates of posterior probability, and how these estimates may be obtained from the output of a suitably trained multi-layer perceptron. The need for novelty detection was identified to avoid classifying data outside the boundaries of the training data, and it was shown how novelty can be assessed using an estimate of the unconditional probability density function. Thus the four basic components of a classification system are: estimating the posterior class probabilities, deciding on a classification, estimating the unconditional density, and deciding on novelty.

It was then explained why these four components should remain decoupled. Firstly, the posterior probabilities and the unconditional probability density function need to be decoupled because they are not equally difficult to estimate and they are used for different purposes. Secondly, although feed-forward networks may be used to classify without explicitly estimating probabilities, such a scheme suffers because these probability estimates are so intrinsically useful in understanding and solving classification problems.

Chapter 3

Making explained classifications

It was shown in chapter 1 that there are important advantages in making *explained* classifications. Chapter 2 described how a statistical approach could be used to provide an excellent framework within which to understand and build principled feed-forward network classifiers. However no mention was made as to whether these classifiers can provide any explanation. This chapter shows how explanation can be fitted into the statistical framework, and how such an approach can be used to understand and control the compromises inherent in making explained classifications. Note that this chapter does *not* describe the methods which actually provide an explanation; these will be covered in chapters 4 and 5.

3.1 The form of an explanation

A classifier is a function of input space which returns a class. An *explaining* classifier is a *comprehensible* function of input space which returns a class. Such a comprehensible function is parameterised in such a way that it is immediately and intuitively obvious which regions of input space will be mapped to which classes. The type of explanation considered in this thesis consists of a set of **if-then** rules called a rule-base. The *consequent* of each rule is a class, and the conditional part of each rule, the *antecedent*, is a simple description of a region of input space. If a test pattern is in the antecedent re-

gion then it is classified as belonging to the consequent class. The rule is then said to be activated by or *cover* the pattern.

Note that the use here of the word “explanation” is not intended to imply the formulation of an argument or the hypothesis of any causation. Neither of these tasks are attempted in this thesis and so “explanation” and “rule-base” should be taken as being synonymous.

Assume initially that input space consists only of ordered variables (no categorical variables). One possible antecedent region is a hyper-rectangle with sides parallel to the axes. For a d -dimensional input pattern, a rule using such an antecedent may be written

$$\mathbf{if } a_1 \leq x_1 \leq b_1 \mathbf{ and } \dots \mathbf{ and } a_d \leq x_d \leq b_d \mathbf{ then class } k. \quad (3.1)$$

This rule provides an explained classification because the classification criterion is explicit and the rule parameters (a_i and b_i) are comprehensible. Note that some of the rule conditions can be omitted by setting the corresponding parameters to plus or minus infinity. For example, the i^{th} dimension (variable x_i) can be effectively removed from the rule by setting $a_i = -\infty$ and $b_i = \infty$.

Rules based on axis-aligned hyper-rectangles are an obvious form of explanation. However it will be seen later that they have a limiting property which becomes severe for problems of more than a few dimensions. A more general form of rule is therefore defined below which includes the axis-aligned rule as a special case.

3.1.1 Rules based on prototypes

Let a rule antecedent region be defined by a threshold on the distance from a prototype pattern. Such a rule may be written

$$\mathbf{if } D(\mathbf{x}, \boldsymbol{\mu}) \leq 1 \mathbf{ then class } k \quad (3.2)$$

where $D(\mathbf{x}, \boldsymbol{\mu})$ is a metric which measures the proximity of the test pattern \mathbf{x} to the prototype pattern $\boldsymbol{\mu}$. A simple way of interpreting such a rule is to state that pattern \mathbf{x}

belongs to class k because it is sufficiently close to a prototype pattern μ belonging to that class. Note that different rules may have different prototype patterns μ and different distance metrics D . Thus both the positions and the shapes of the antecedent regions are different for different rules.

A convenient family of distance metrics are based upon the so called L -norms [Kre88] or p -norms [TI97],

$$D_{\mathbf{u},p}(\mathbf{x}, \mu) = \left(\sum_i \left| \frac{x_i - \mu_i}{u_i} \right|^p \right)^{1/p}, \quad (3.3)$$

where the u_i weight the importance of each dimension. Of particular interest are the three cases of $p = 1$, $p = 2$, and $p = \infty$ because they correspond to intuitively understood distances. The first metric, $p = 1$, is called a weighted city block or Manhattan distance because it is the weighted sum of the distances along a rectilinear grid. The region defined by $D_{\mathbf{u},1}(\mathbf{x}, \mu) \leq 1$ is a diamond shape centred at μ , and figure 3.1 shows such a region in two dimensions. However a two dimensional picture is potentially misleading because it suggests a shape with the same number of corners and sides as a hyper-rectangle. This is not the case since in d dimensions a hyper-rectangle has 2^d corners and $2d$ sides, whereas a hyper-diamond has only $2d$ corners but 2^d sides.

The second metric, $p = 2$, is the weighted Euclidean distance. The region defined by $D_{\mathbf{u},2}(\mathbf{x}, \mu) \leq 1$ is an axis-aligned hyper-ellipse centred at μ . Figure 3.1 shows such a region in two dimensions.

The third metric, $p = \infty$, is an axis-aligned hyper-rectangle which is slightly more constrained than the one described by the antecedent of (3.1). To see this, consider the fol-

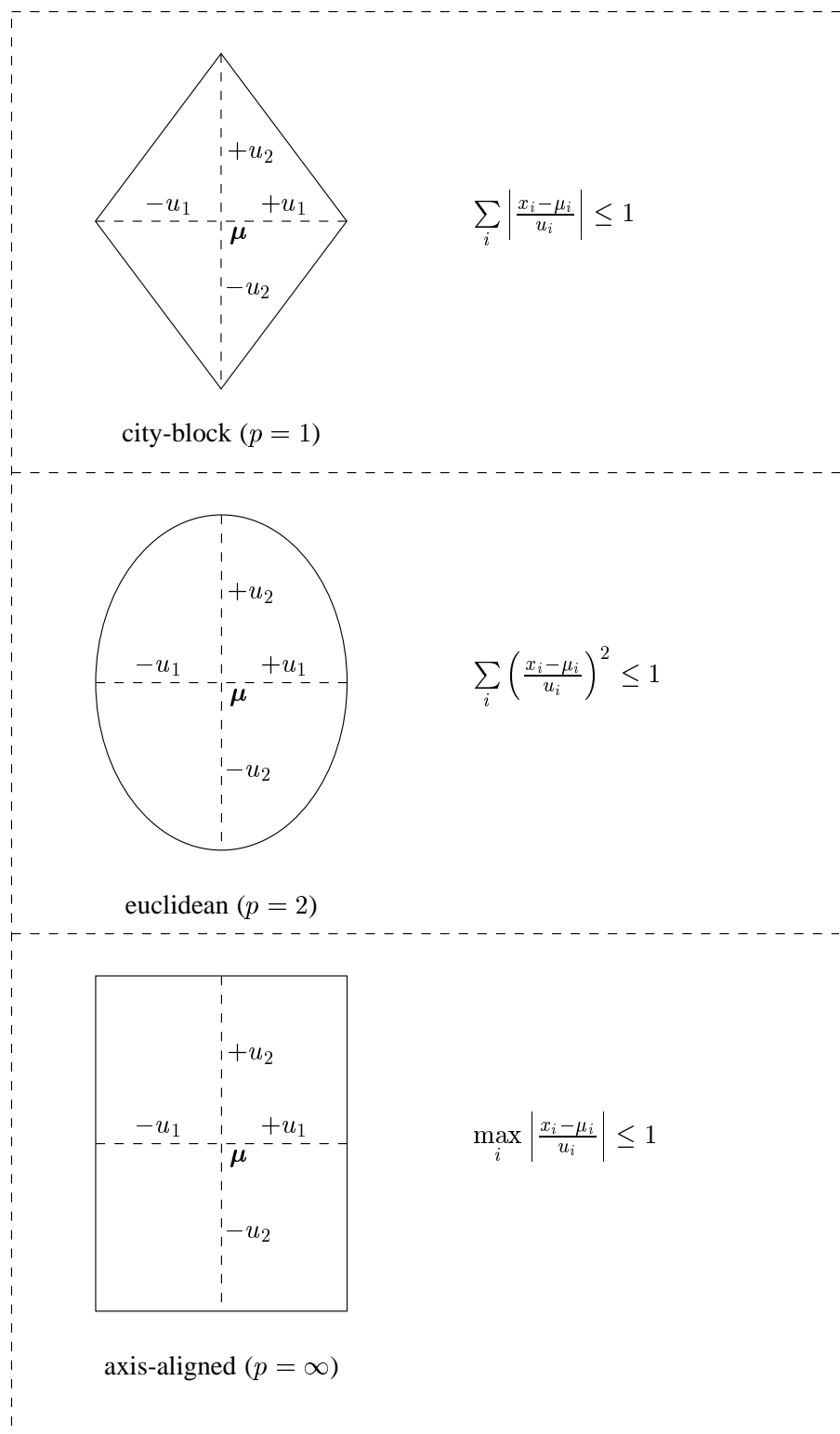


Figure 3.1: Two dimension illustrations of the three types of rule antecedent formed from the distance metric defined by equation (3.3)

lowing steps:

$$\begin{aligned}
\left(\sum_i \left| \frac{x_i - \mu_i}{u_i} \right|^\infty \right)^{1/\infty} &\leq 1 \equiv \max_i \left(\left| \frac{x_i - \mu_i}{u_i} \right| \right) \leq 1 \\
&\equiv \bigwedge_i \left(\left| \frac{x_i - \mu_i}{u_i} \right| \leq 1 \right) \\
&\equiv \bigwedge_i -u_i \leq (x_i - \mu_i) \leq u_i \\
&\equiv \bigwedge_i (\mu_i - u_i) \leq x_i \leq (\mu_i + u_i) \quad (3.4)
\end{aligned}$$

where \bigwedge is the conjunction operator. Thus if μ_i and u_i are chosen such that

$$\begin{aligned}
a_i &= \mu_i - u_i \\
b_i &= \mu_i + u_i \quad (3.5)
\end{aligned}$$

then the antecedent of (3.1) and equation (3.4) are equivalent. Figure 3.1 shows such a region in two dimensions.

However it is not possible to find μ_i and u_i such that only one of a_i or b_i is infinite. This problem is a symptom of a more significant constraint: the position of the prototype pattern must be midway between the extremes of the antecedent region. In other words, the prototype must be at the centre of the region. To see the disadvantage of this constraint, consider trying to use such rules to describe a region which is infinite in extent in one or more directions (a common occurrence). The centre of the region is therefore not well defined, so neither is the position of the prototype.

A distance function which uses different weighting coefficients depending on the sign of $(x_i - \mu_i)$ can be used to solve this problem. Hence the following family of functions can be defined

$$D_{\mathbf{u}, \mathbf{v}, p}(\mathbf{x}, \boldsymbol{\mu}) = \left(\sum_i U(x_i - \mu_i) \left| \frac{x_i - \mu_i}{u_i} \right|^p + U(\mu_i - x_i) \left| \frac{\mu_i - x_i}{v_i} \right|^p \right)^{1/p} \quad (3.6)$$

where $U(\cdot)$ is the step function,

$$U(t) = \begin{cases} 0 & \text{if } t < 0 \\ 1 & \text{if } t \geq 0 \end{cases}. \quad (3.7)$$

Note that this function does not define a distance metric because in general it lacks symmetry. In other words $D_{\mathbf{u},\mathbf{v},p}(\boldsymbol{\mu}, \mathbf{x}) = D_{\mathbf{u},\mathbf{v},p}(\mathbf{x}, \boldsymbol{\mu})$ if and only if $\mathbf{u} = \mathbf{v}$. Ambiguity in the ordering of the arguments to D can be avoided by adhering to the order used in equation (3.6).

The shape of the antecedent regions defined by rules based on D_p for $p = 1, 2, \infty$ are shown for two dimensions in figure 3.2. Note that the prototype is no longer constrained to be at the centre of the antecedent region; ultimately this extra freedom will allow the prototype to be placed in a more representative position.

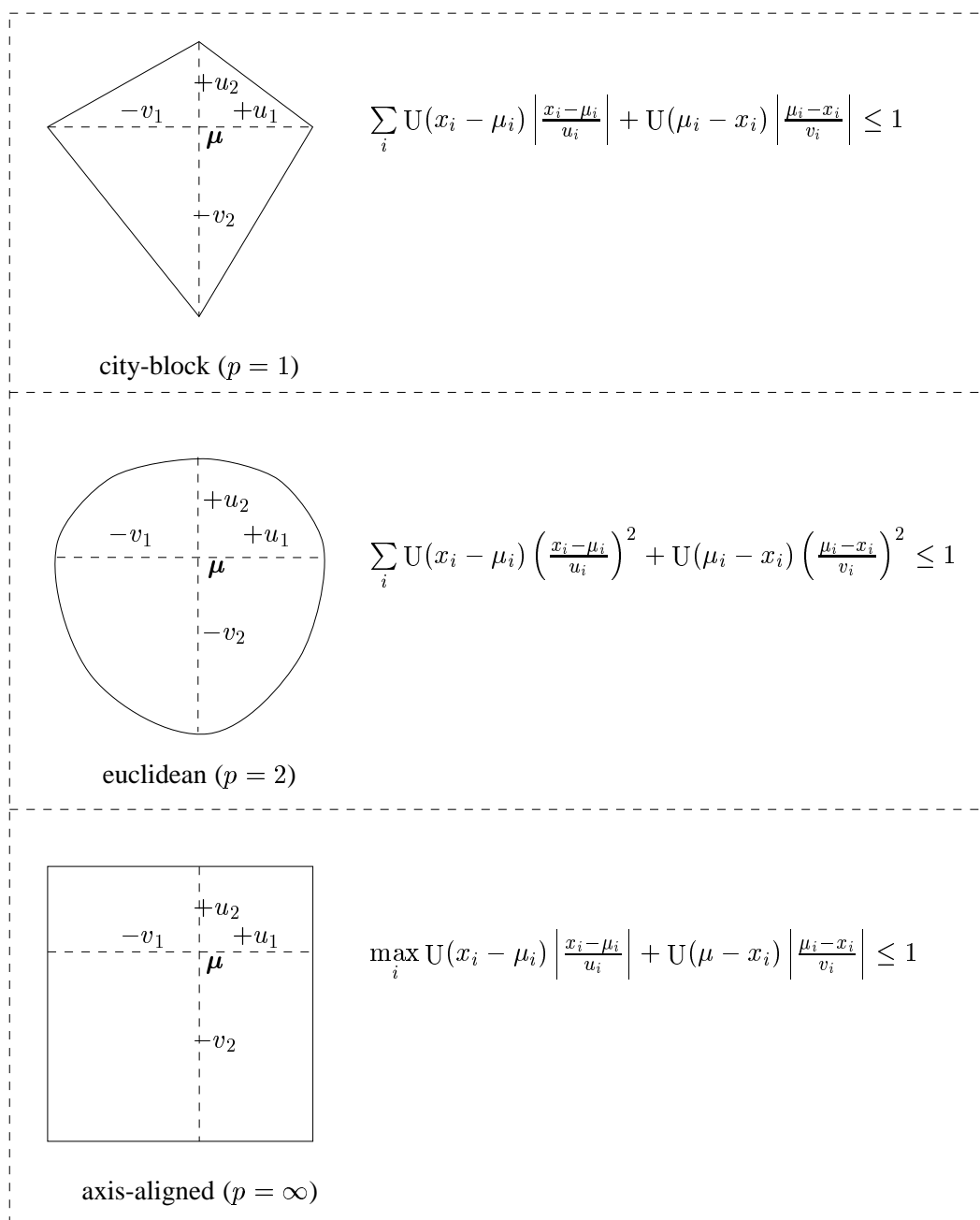


Figure 3.2: Two dimension illustration of the three types of rule antecedent formed from the function defined by equation (3.6).

Class k if			
x_0 :	$-v_0$	μ_0	u_0
x_1 :	$-v_1$	μ_1	u_1
\vdots	\vdots	\vdots	\vdots
x_d :	$-v_d$	μ_d	u_d

Figure 3.3: The generic format of a rule for ordered variables.

3.1.2 A user friendly format

It is clear that equation (3.6) is too cumbersome to be presented to the user as part of a rule. A much more user friendly format is required, and one such possibility is shown in figure 3.3. For each particular rule the x_i 's are replaced by the names of the variables and the variables μ_i, u_i, v_i are replaced by their numerical value. The rule is read differently depending on the value of p .

For the city-block rules ($p = 1$), the condition for rule activation may be written

$$\sum_i U(x_i - \mu_i) \frac{x_i - \mu_i}{u_i} + U(\mu_i - x_i) \frac{\mu_i - x_i}{v_i} \leq 1, \quad (3.8)$$

which suggests an intuitive way of reading such a city-block rule. Observe that only one of the two terms in the summation is non-zero, and that this term then *contributes* part of the “distance” from the pattern relative to either u_i or v_i . Thus the rule is activated if the sum of the contributions is less than unity. Although this is slightly cumbersome to describe, it is simple and intuitive to use.

Euclidean rules ($p = 2$) have a similar condition for rule activation,

$$\sum_i U(x_i - \mu_i) \left(\frac{x_i - \mu_i}{u_i} \right)^2 + U(\mu_i - x_i) \left(\frac{\mu_i - x_i}{v_i} \right)^2 \leq 1, \quad (3.9)$$

and so may be interpreted in much the same way as the city-block rule. However the squaring makes it more difficult to mentally estimate the contributions from each term.

The condition for activation of an axis-aligned rule is

$$\bigwedge_i (\mu_i - u_i) \leq x_i \leq (\mu_i + v_i) \quad (3.10)$$

which is extremely easy to interpret because the conditions on the variables are independent of each other. Note that the format of (3.1) is probably simpler and could be used instead. However it is convenient to use the same generic format for all three types of rule.

Finally note that the effect of infinite u_i or v_i is to place no constraint on x_i when it is on the appropriate side of the μ_i . Hence a sensible convention is to represent these infinities by leaving the corresponding positions in the rule blank.

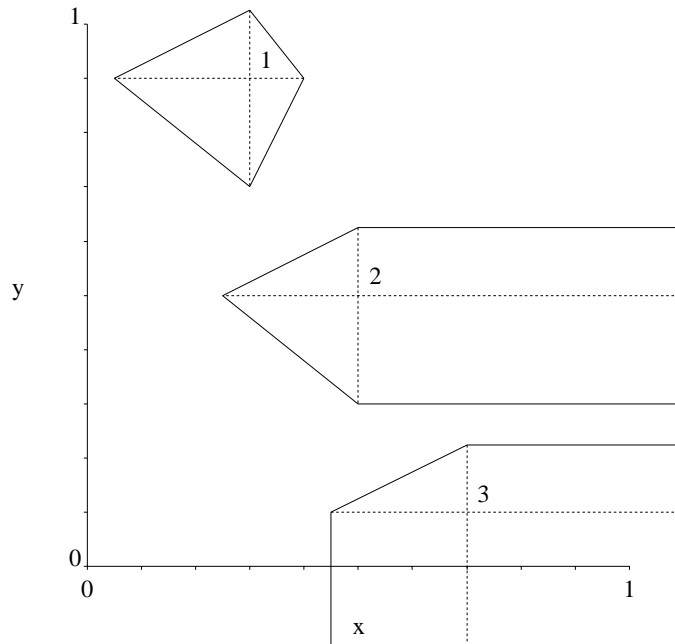
3.1.3 Some examples

Figure 3.4 shows three pedagogical city-block rules both graphically and in the user friendly format of figure 3.3. The class 1 rule is shown in the top left corner. It has a prototype at $(0.3, 0.9)$ and is bounded on all four sides because all the u_i and v_i are finite. For example, the point $(0.25, 0.8)$ activates this rule because the two contributions sum to less than unity,

$$\frac{0.25 - 0.3}{-0.25} + \frac{0.8 - 0.9}{-0.2} = 0.7 \leq 1. \quad (3.11)$$

The class 2 rule is shown in the middle and is unbounded in the positive x-direction because $u_1 = \infty$. Note that this ∞ is represented as a blank in the rule format. Similarly the class 3 rule is unbounded in two directions.

Figures 3.5 and 3.6 shows the appearance of these rules if instead of being city-block they are of the euclidean and axis-aligned type. The text for these rules is omitted because it is identical to the text shown in figure 3.4. Obviously the interpretation of the text is dependent on the rule type.



Class 1 if

x	-0.25	0.3	+0.1
y	-0.2	0.9	+0.125

Class 2 if

x	-0.25	0.5	
y	-0.2	0.5	+0.125

Class 3 if

x	-0.25	0.7	
y		0.1	+0.125

Figure 3.4: Three city-block rules shown both graphically and in the format of figure 3.3.

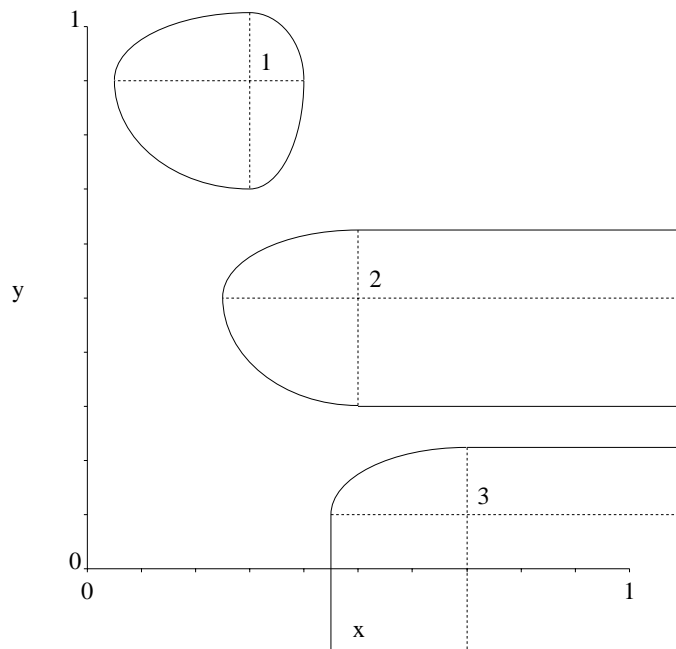


Figure 3.5: The same rules as in figure 3.4 but of euclidean rather than city-block type.

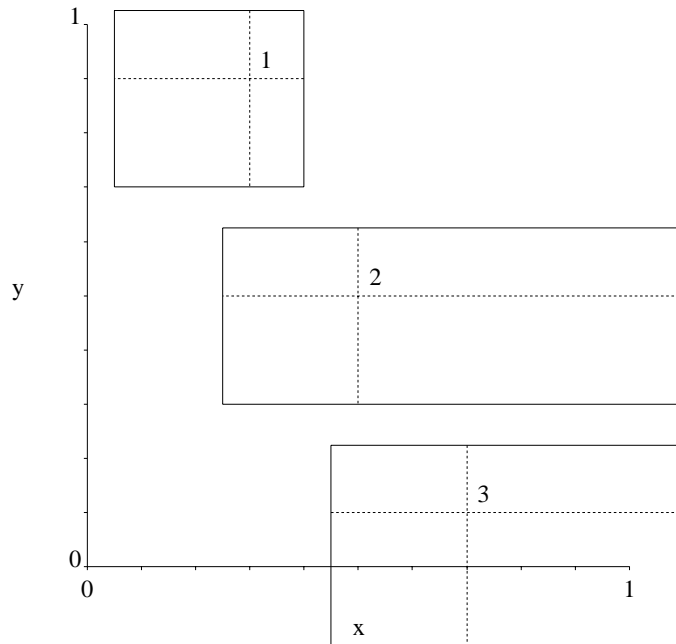


Figure 3.6: The same rules as in figure 3.4 but of axis-aligned rather than city-block type.

Class k if			
x_0 :	a_{01}	a_{02}	\dots
x_1 :	a_{11}	a_{12}	\dots
\vdots		\vdots	
x_d :	a_{d1}	a_{d2}	\dots

Figure 3.7: The generic format of a rule for categorical variables.

3.1.4 Incorporating categorical variables

Categorical variables require a rule antecedent which does not rely on an ordering of values to describe a subset of input space. An expressive way of writing an all-categorical rule is shown in figure 3.7. Variable x_i contributes a_{ij} when x_i is equal to its j^{th} category¹, and the rule is activated if the sum of the contributions is less than unity.

In fact this format arises naturally from the prototype rules for ordered variables given previously. To see this, consider encoding categorical variables using a scheme such as the one described in section 2.3.4. The important property is that the categories of each variable are encoded using a set of vectors in a space separate from the other variables. Thus a prototype rule acting on these encodings may be re-written in the form of figure 3.7 simply by calculating the contribution made from each encoded category.

Obviously input spaces which consist of a mixture of ordered and categorical variables require rules which consist of a combination of the formats shown in figures 3.3 and 3.7.

3.1.5 Intersecting rules

It is possible for the antecedent regions of different rules to intersect. This can be an advantage when the intersecting rules indicate the same class since this allows each rule antecedent to be larger and therefore cover more patterns. However intersecting rules which indicate different classes give conflicting classifications for a common set of patterns. There are two solutions to this problem: either avoid having conflicting rules in the rule-base; or order the rules and assign every test pattern to the class given by the first ac-

¹The categories are enumerated *only* so that they may be listed; clearly this ordering is arbitrary and unimportant.

tivated rule. One possible criterion used to order the rules is to position the most effective rules first.

3.2 Explanation vs optimal classification

The previous chapter described how a Bayes classifier makes the optimal classification decision using estimates of class probabilities. Clearly a Bayes classifier which uses a feed-forward network to estimate these probabilities does not provide an explanation in the form of the rules described above. In fact it does not provide an explanation of any form because the criterion used to make each classification is implicitly encoded in the weights and offsets of the network. More precisely, each pattern is classified individually and there is no indication of which other patterns would also be assigned to the same class.

The reason for this non-explaining property of the Bayes classifier $\hat{d}_B(\mathbf{x})$ is straightforward. $\hat{d}_B(\mathbf{x})$ is made from two non-explaining components: the Bayes classification decision, which is defined by an optimality criterion which uses the posterior probabilities to assign the most likely class to each pattern; and the feed-forward network, which is optimised to estimate these class probabilities. Neither of these two components has a specification which *involves* explanation, and so the classifier formed from their combination does not *optimise* explanation.

There are two approaches to solving this problem: either insert the explanation at the level of the class decision, or insert the explanation at the level of posterior probability estimation. Examples of the latter include classification trees [BFOS84, Qui93] and the rule extraction methods of [Bla93, GHM92] (see chapter 1), but models which estimate both posterior probabilities and give explained classification are making an implicit compromise. The probability estimates of such models are effectively regularised by the need to make explained classifications, and this regularisation may not be optimal. Conversely, the need to estimate the posterior probabilities must have an affect on the quality of the explained classifications, but this effect may be detrimental. In other words, combining probability estimation with explanation has prevented their separate optimisation.

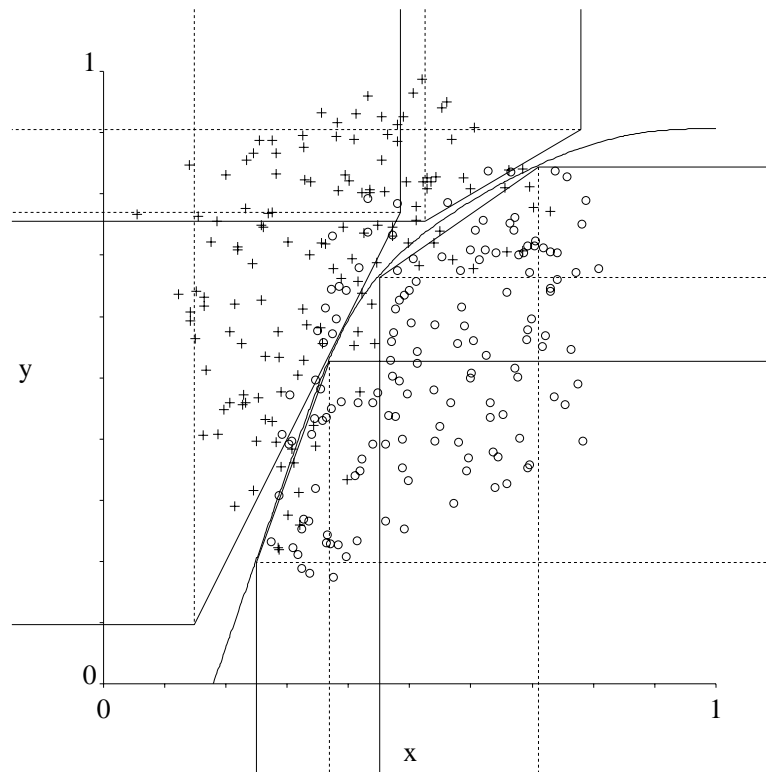
This argument against merging posterior probability estimation with explanation suggests that a better alternative may be to try to insert the explanation at the level of the class decision. In other words, the aim is to explain the classifications made by a system which is not influenced by the need to explain. This is the approach taken in this thesis because it makes explicit the compromises which are only implicit within a combined model. This enables the compromises to be better understood and balanced in an appropriate trade-off.

3.2.1 The fundamental problem of explanation

As justified above, an explaining classifier should ideally provide a transparent description of the classification given by the Bayes classifier $\hat{d}_B(\mathbf{x})$. The following argument first shows why this ideal criterion is impossible to achieve, and then suggests two compromises which will enable a satisfactory criterion to be met.

Recall that the classifier $\hat{d}_B(\mathbf{x})$ divides input space into class regions separated by boundaries. Recall also that a rule assigns all patterns within its antecedent region to a single class. Therefore if a rule is to make the same classifications as the Bayes classifier, then the antecedent region of a class k rule must lie entirely within the class k region defined by $\hat{d}_B(\mathbf{x})$. Figures 3.8, 3.9, 3.10 illustrate how a small number of each type of rule can begin to describe the classification provided by $\hat{d}_B(\mathbf{x})$.

There is now an obvious problem: the incompatibility of shape between the rule antecedents and the class boundaries implies that an infinite number of rules is required to describe the Bayes classifier completely. An infinite number of rules does not form an explanation; a dozen rules is probably a generous upper limit. This limit implies that some patterns cannot be given an explained classification. The above examples were two dimensional and had a simple classification boundary, which makes it possible for a small rule-base to cover most of the patterns. However *the* fundamental problem of explanation is that the number of rules required to explain a significant fraction of patterns increases *exponentially* with the number of dimensions.



```

Class 1 if
  x   -0.2592  0.7104
  y           0.6629  +0.1805

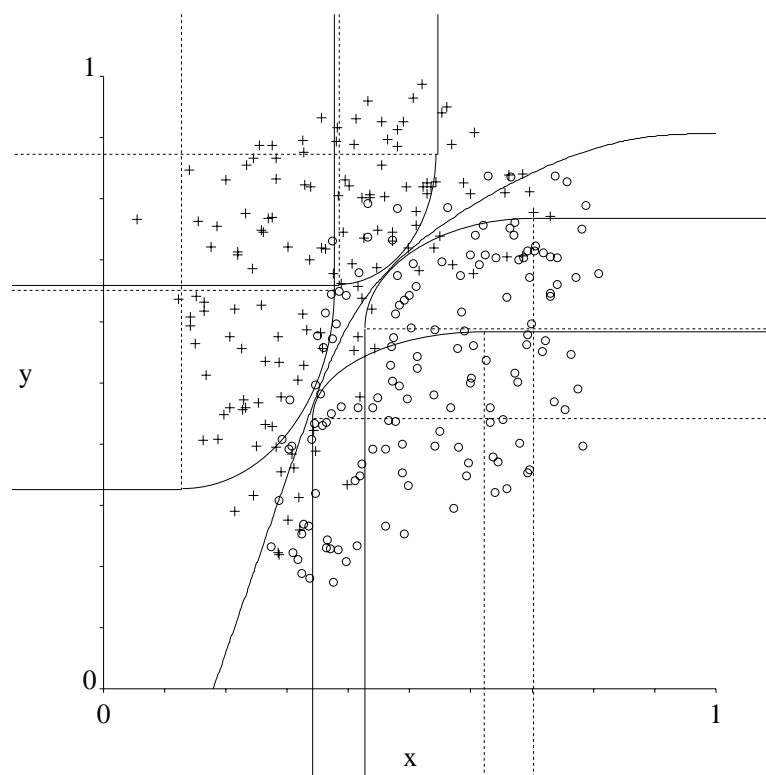
Class 1 if
  x   -0.1189  0.3692
  y           0.1986  +0.3286

Class 2 if
  x           0.5260  +0.2537
  y   -0.1502  0.9052

Class 2 if
  x           0.1491  +0.3360
  y   -0.6737  0.7704

```

Figure 3.8: Four city-block rules used to describe the estimated Bayes classification boundary on an artificial dataset. These rules cover approximately 94% of the patterns.



```

Class 1 if
  x   -0.2750  0.7027
  y           0.5888  +0.1789

```

```

Class 1 if
  x   -0.2811  0.6223
  y           0.4418  +0.1422

```

```

Class 2 if
  x           0.1286  +0.2488
  y   -0.3257  0.6514

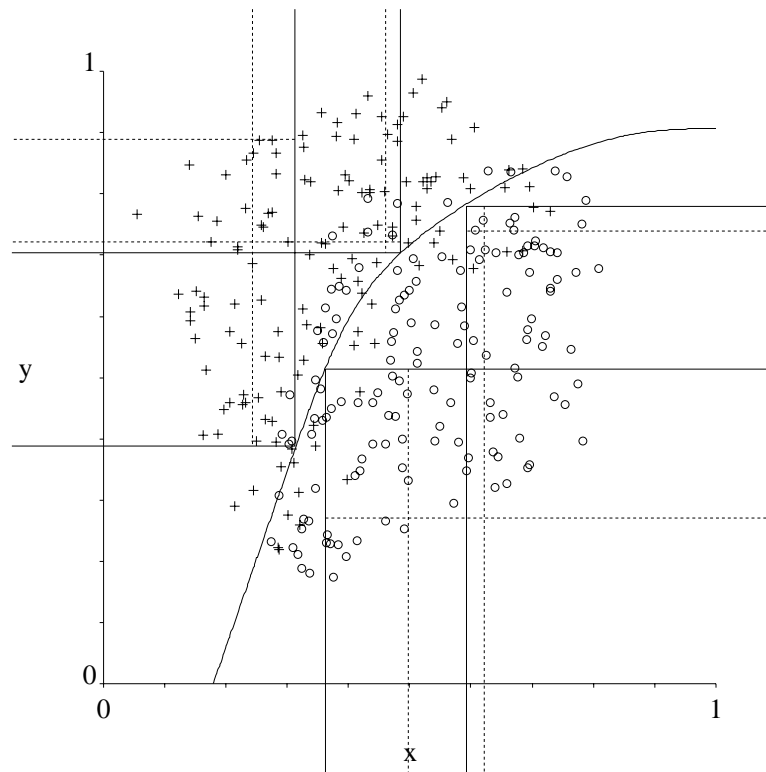
```

```

Class 2 if
  x           0.3855  +0.1605
  y   -0.2131  0.8722

```

Figure 3.9: Four euclidean rules used to describe the estimated Bayes classification boundary on an artificial dataset. These rules cover approximately 82% of the patterns.



```

Class 1 if
  x   -0.0288  0.6210
  y                   0.7389  +0.0408

Class 1 if
  x   -0.1353  0.4977
  y                   0.2711  +0.2425

Class 2 if
  x                   0.2437  +0.0686
  y   -0.5006  0.8887

Class 2 if
  x                   0.4611  +0.0237
  y   -0.0182  0.7219

```

Figure 3.10: Four axis-aligned rules used to describe the estimated Bayes classification boundary on an artificial dataset. These rules cover approximately 64% of the patterns.

This important effect can be demonstrated by calculating the optimal fraction of space covered by a single rule when it is used to describe different boundaries. The two most analytically convenient and realistic boundaries are those formed from a hyperplane and a hypersphere. Table 3.1 gives the formulae for the optimal fraction of space covered by a rule of each type when describing a hyperplane boundary. Table 3.2 gives the same information for a hypersphere boundary. See appendix D for the derivations of the formulae in both of these tables. Figures 3.11 and 3.12 show how these fractions decrease with increasing number of dimensions.

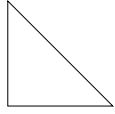
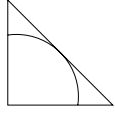
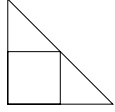
Rule type	Picture	Fraction
City-block		1
Euclidean		$(\frac{\pi}{4d})^{d/2} \frac{d!}{(d/2)!}$
Axis-aligned		$\frac{d!}{d^d}$

Table 3.1: The fraction of hyperplane bounded space covered by each of the three types of rule; d is the number of dimensions.

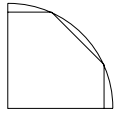
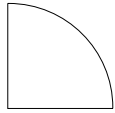
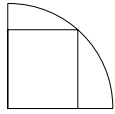
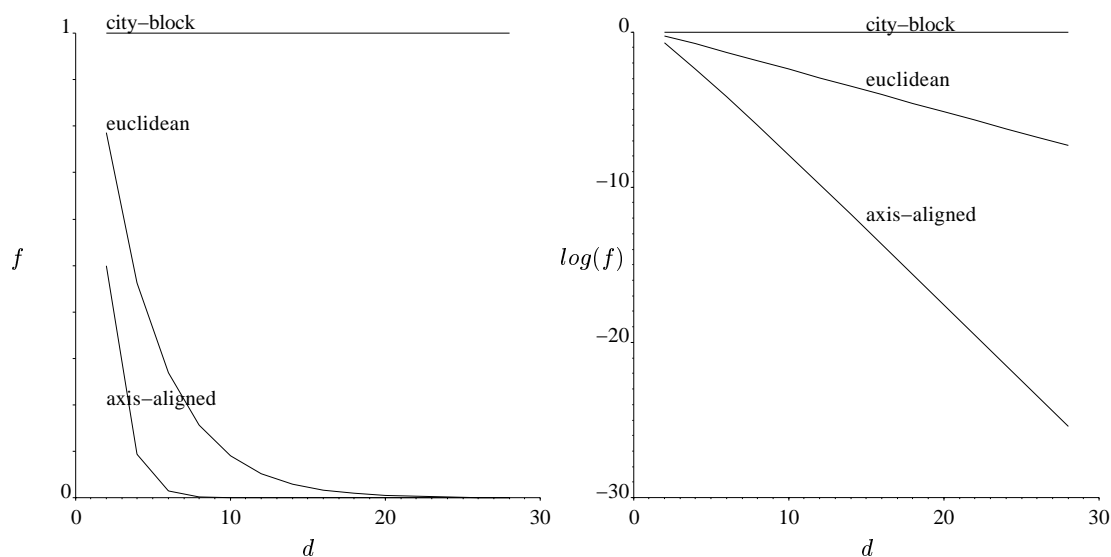
Rule type	Picture	Fraction
City-block		—
Euclidean		1
Axis-aligned		$\frac{(d/2)!}{(\frac{\pi d}{4})^{d/2}}$

Table 3.2: The fraction of hypersphere bounded space covered by each of the three types of rule; d is the number of dimensions. No closed form solution could be found for the city block fraction.

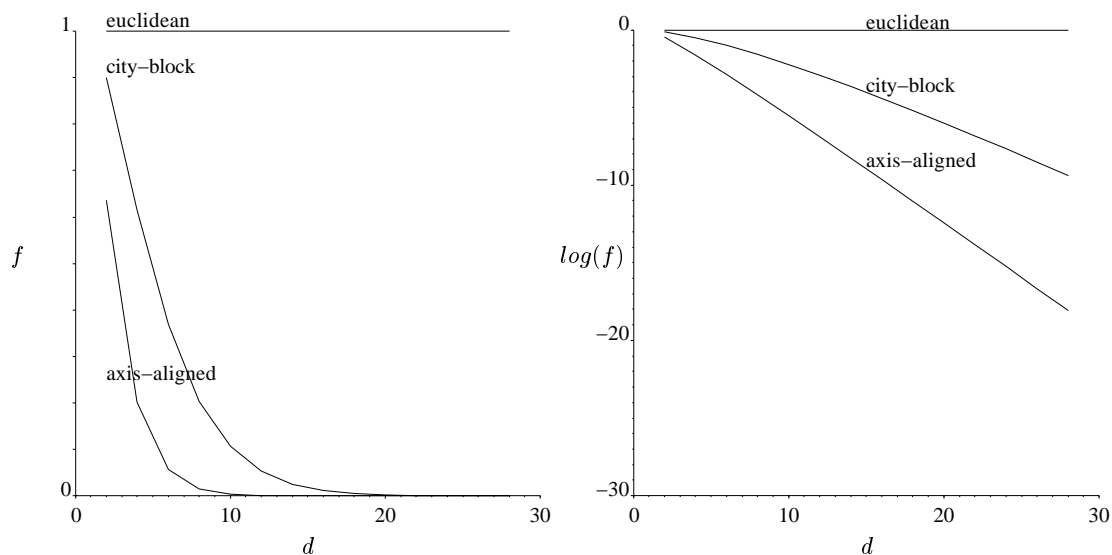


f	$d = 2$	$d = 4$	$d = 10$	$d = 20$
city-block	1	1	1	1
euclidean	0.78	0.46	0.090	0.0058
axis-aligned	0.5	0.093	3.6×10^{-4}	2.3×10^{-8}

Figure 3.11: Fraction of space f (as a function of dimension d) which each of the three types of rules cover when pushed up against a hyperplane boundary.

One partial solution to this fundamental problem is to allow the rule antecedent regions to cross the class boundary, in which case the rules make sub-optimal (with respect to $\hat{d}_B(\mathbf{x})$) classifications. Figure 3.13 shows the effect of allowing axis-aligned rules to cross over the classification boundary. Note that the percentage of patterns covered has increased from 64% (in figure 3.10) to 85%. Thus relaxing the restriction that rules must make the same classification as $\hat{d}_B(\mathbf{x})$ enables a rule-base to classify more patterns. Obviously this relaxation must be carefully controlled if the rule-base is not to make improbable classifications. On an intuitive level, allowing rules to be sub-optimal is only acceptable if there is sufficient overlap between the patterns from each class. This important point will be described more precisely in section 3.3.4.

There are therefore two ways in which an explaining classifier can be sub-optimal with respect to the Bayes classifier $\hat{d}_B(\mathbf{x})$. First, the rule antecedent regions may cover only part of input space, resulting in some patterns not being classified by the rule-base. Second, the rule antecedent regions could be allowed to cross the class boundary, resulting



f	$d = 2$	$d = 4$	$d = 10$	$d = 20$
city-block	0.90	0.62	0.11	0.0025
euclidean	1	1	1	1
axis-aligned	0.64	0.20	0.004	4.0×10^{-6}

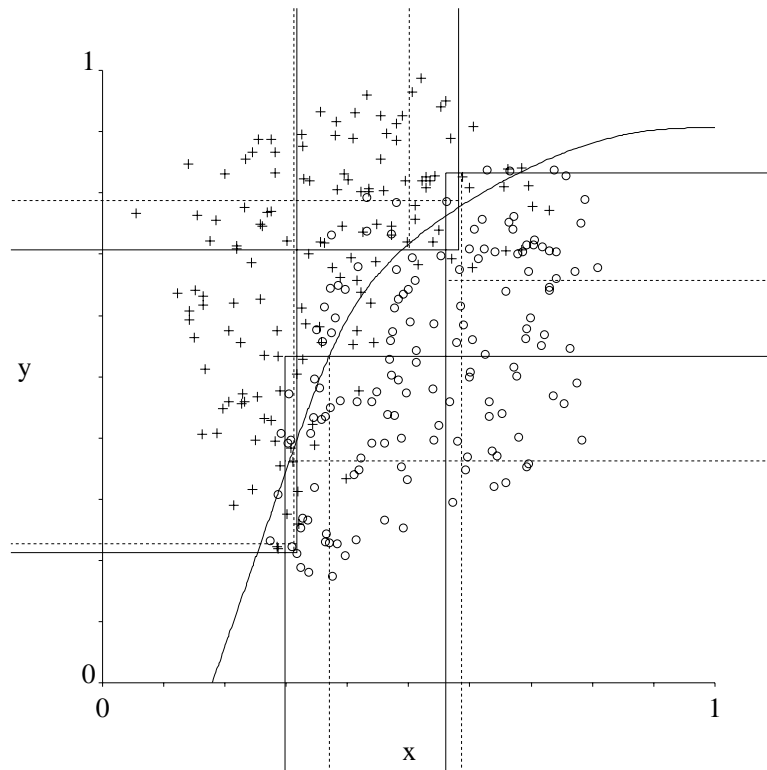
Figure 3.12: Fraction of space f (as a function of dimension d) which each of the three types of rules cover when pushed up against a spherical boundary.

in the rule-base assigning some patterns to classes different from those given by $\hat{d}_B(\mathbf{x})$. Any practical explaining classifier must balance these sub-optimality if it is to assign a reasonable fraction of patterns to a reasonably probable class using only a small number of rules.

3.2.2 Extending the decoupled framework

The decoupled framework described in section 2.4 can now be extended to incorporate a component which makes explained classifications. The use of such an explanation component to classify new patterns is shown by the run-time flow diagram of figure 3.14.

As a justification for placing the explaining classifier *after* the novelty detector, note that the explaining classifier uses the complete posterior probability function to decide the position of the rule antecedent regions. However the posterior probability estimate is only



```

Class 1 if
  x   -0.0720  0.3712
  y           0.3632  +0.1702

Class 1 if
  x   -0.0260  0.5871
  y           0.6569  +0.1752

Class 2 if
  x           0.3134  +0.0037
  y   -0.0151  0.2279

Class 2 if
  x           0.5009  +0.0801
  y   -0.0795  0.7873

```

Figure 3.13: Four axis-aligned rules used to describe the estimated Bayes classification boundary on an artificial dataset. Unlike figure 3.10, these rules make sub-optimal classifications (they cross the boundary) and so cover more patterns – approximately 85%.

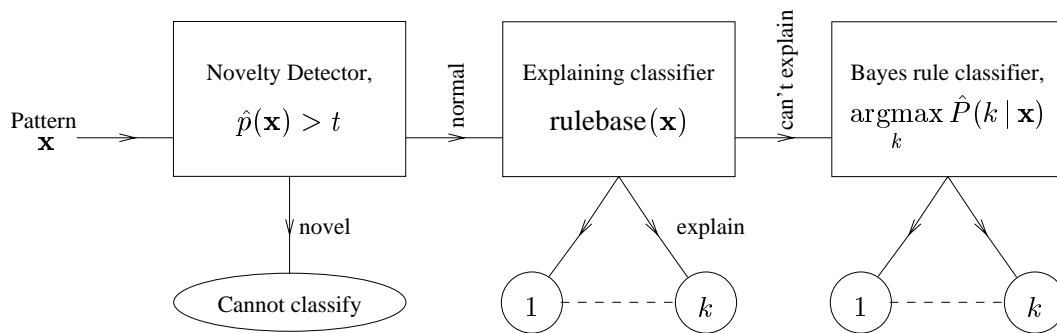


Figure 3.14: An explicitly decoupled explaining classification framework.

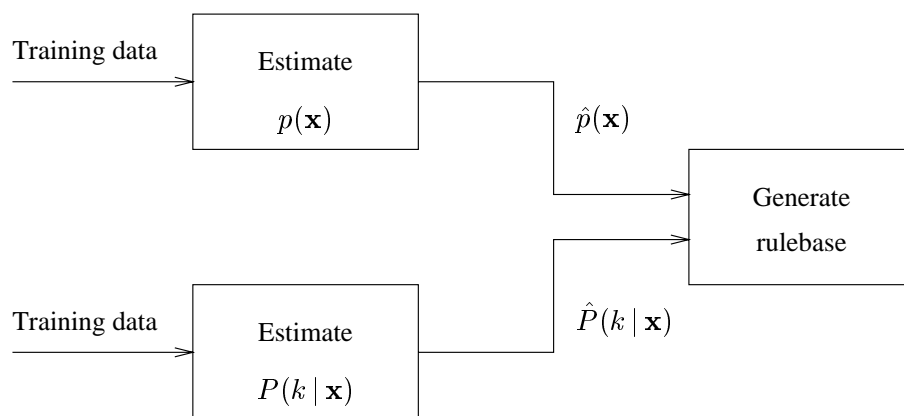


Figure 3.15: A rule-base is generated from probability estimates, which are in turn estimated from the training data.

valid for normal patterns, and so the explaining classifier is also only valid for normal patterns. Hence the explaining classifier fits into the pipeline after the novelty detector.

To justify placing the explaining classifier *before* the Bayes classifier $\hat{d}_B(\mathbf{x})$, note that if one cannot classify a pattern with explanation, then the best alternative is to classify without explanation. It is therefore natural to use $\hat{d}_B(\mathbf{x})$ for those patterns not classified by the rule-base. Hence the explaining classifier fits into the pipeline before the Bayes classifier.

It is important to emphasize that the Bayes classifier and the explanation classifier use probability estimates in quite different ways. The Bayes classifier uses the estimated class posterior probabilities of a pattern to decide its most likely class. However it is clear that an explaining classifier cannot do this because the class posterior probabilities are estimated by the MLP, and a MLP is a “black box” producing numbers without explanation. Hence the explaining classifier is generated from estimates of the class posterior probabilities without reference to any of the patterns to be explained. In addition, to ensure that the rules cover regions of space likely to be occupied by test patterns, the generation of the rule-base also requires an estimate of the unconditional density. Thus figure 3.15 summarises the training-time flow at a similar level of detail to that shown in the run-time flow given in figure 3.14

Recall that the purpose of a decoupled framework is to give a clean specification for the optimisation of each component. In particular, optimising the class posterior probability estimates is decoupled from the optimisation of the rule-base. Note that this does not imply that the two are unconnected, but that the need to explain should not affect the estimation of the posterior probabilities. A corollary of this principle is that even though the rules within the explaining classifier are in terms of distances to prototypes, this does not imply that the posterior probabilities should necessarily be estimated using prototype-based networks such as radial basis function (RBF) networks, [Hay99]. As explained in chapter 2, it will be assumed in this thesis that the class posterior probabilities are estimated using a multi-layer perceptron (MLP). It will be seen later that this choice does not in any way handicap the generation of prototype based rules.

Finally, no attempt will be made to explain the novelty detector even though it is used on every test pattern. This is reasonable since only a very small number of patterns are ordinarily flagged as novel; prolonged occurrence of “novel” patterns would either indicate a change in the underlying density distribution or a poorly trained density model.

3.3 Measuring explanation

It is clearly important to be able to monitor the type and degree of sub-optimality of an explaining classifier. This section defines a set of *measures* which indicate the closeness of the explaining classifier $\hat{d}_E(\mathbf{x})$ to the Bayes classifier $\hat{d}_B(\mathbf{x})$.

3.3.1 Parsimony

It is obvious that a rule-base consisting of a large number of complex rules does not constitute an explanation; both the number and complexity of the rules must be limited for the rule-base to have any explanatory ability. Hence one of the desirable properties of an explaining classifier is for it to be *parsimonious* (consists of only a small number of simple rules). Parsimony is the only purely qualitative measure used to describe an explaining classifier.

3.3.2 Effectiveness

One of the ways to make a rule-base more parsimonious is not to classify every pattern. In contrast to this, the Bayes classifier always classifies every pattern. This difference can be described by the *effectiveness* of an explaining classifier $\hat{d}_E(\mathbf{x})$, which is defined as the probability that it makes a classification. If \mathcal{C} is the set of all patterns which are classified by $\hat{d}_E(\mathbf{x})$, then the effectiveness $\eta(\hat{d}_E)$ can be written

$$\begin{aligned}\eta(\hat{d}_E) &= P(\mathbf{X} \in \mathcal{C}) \\ &= \int_{\mathcal{C}} p(\mathbf{x}) d\mathbf{x}.\end{aligned}\tag{3.12}$$

Clearly an explaining classifier with high effectiveness has a high probability of making an explained classification. However it is also useful to know how effective the explaining classifier is at classifying patterns from each class. Hence the *conditional effectiveness* is defined as the probability of the explaining classifier assigning a pattern to any class given that the pattern most probably belongs to class k .

$$\begin{aligned}\eta_k(\hat{d}_E) &= P(\mathbf{X} \in \mathcal{C} \mid \hat{d}_B(\mathbf{X}) = k) \\ &= \int_{\mathcal{C}} p(\mathbf{x} \mid \hat{d}_B(\mathbf{x}) = k) d\mathbf{x}\end{aligned}\tag{3.13}$$

Estimating effectiveness

In practice the underlying unconditional distribution $p(\mathbf{x})$ is unknown. However there are several ways of estimating effectiveness. The first possibility is to approximate $p(\mathbf{x})$ by $\hat{p}(\mathbf{x})$ in equations (3.12) and (3.13). Monte Carlo integration can be used if the subsequent integrals cannot be evaluated analytically. This amounts to using the following formulae on a sample from $\hat{p}(\mathbf{x})$:

$$\hat{\eta}(\hat{d}_E) = \frac{\text{\#explained classifications}}{\text{\#patterns}},\tag{3.14}$$

$$\hat{\eta}_k(\hat{d}_E) = \frac{\text{\#explained classifications which are optimally class-}k}{\text{\#classifications which are optimally class-}k}.\tag{3.15}$$

Another possibility is to use these formulae on a sample from $p(\mathbf{x})$ rather than $\hat{p}(\mathbf{x})$. Clearly the training or test data are such samples, but note that the expert class labels are not used.

3.3.3 Comparative accuracy

Of the two ways in which an explaining classifier can be said to be sub-optimal, the first (not classifying every pattern) is measured by effectiveness. The second (not assigning the optimal class to every pattern) is partially measured by what will be known as *comparative accuracy*.

The comparative accuracy of an explaining classifier $\hat{d}_E(\mathbf{x})$ is defined as the probability of $\hat{d}_E(\mathbf{x})$ making the same classification as the Bayes classifier $\hat{d}_B(\mathbf{x})$.

$$\begin{aligned}\gamma(\hat{d}_E, \hat{d}_B) &= P(\hat{d}_E(\mathbf{X}) = \hat{d}_B(\mathbf{X}) \mid \mathbf{X} \in \mathcal{C}) \\ &= \int_{\hat{d}_E(\mathbf{x})=\hat{d}_B(\mathbf{x})} p(\mathbf{x} \mid \mathbf{x} \in \mathcal{C}) d\mathbf{x}.\end{aligned}\quad (3.16)$$

Thus a classifier with an effectiveness of 0.8 and a comparative accuracy of 0.5 fails to classify 20% of all patterns, and half of those patterns which are classified will be assigned to a class which is different from the Bayes classifier.

Comparative accuracy suffers from the same insensitivity to unbalanced class priors as was shown previously to be a problem with accuracy (see section 2.2.1). For example, if the Bayes classifier assigns 90% of all patterns in \mathcal{C} to class 1 then an explaining classifier can achieve a comparative accuracy of 0.9 by classifying all patterns (within \mathcal{C}) as class 1. As before, the problem is solved by defining a conditional measure. The *conditional comparative accuracy* of $\hat{d}_E(\mathbf{x})$ with respect to class k is the probability of $\hat{d}_E(\mathbf{x}) = k$ given that $\hat{d}_B(\mathbf{x}) = k$.

$$\begin{aligned}\gamma_k(\hat{d}_E, \hat{d}_B) &= P(\hat{d}_E(\mathbf{X}) = k \mid \hat{d}_B(\mathbf{X}) = k, \mathbf{X} \in \mathcal{C}) \\ &= \int_{\hat{d}_E(\mathbf{x})=k} p(\mathbf{x} \mid \hat{d}_B(\mathbf{x}) = k, \mathbf{x} \in \mathcal{C}) d\mathbf{x}.\end{aligned}\quad (3.17)$$

Thus in the above example, the poor classification of class 2 patterns is flagged by $\gamma_1 = 1.0$ and $\gamma_2 = 0.0$.

Estimating comparative accuracy

As with effectiveness, the defining integrals for comparative accuracy are in terms of an unknown probability density. However the following equations provide an estimate using

a sample:

$$\hat{\gamma}(\hat{d}_E, \hat{d}_B) = \frac{\text{\#explained optimal classifications}}{\text{\#explained classifications}}, \quad (3.18)$$

$$\hat{\gamma}_k(\hat{d}_E, \hat{d}_B) = \frac{\text{\#explained class-}k \text{ classifications which are optimally class-}k}{\text{\#explained classifications which are optimally class } k}. \quad (3.19)$$

The sample can be generated in one of two ways: from the estimated probability distribution $\hat{p}(\mathbf{x})$; or from the assumed underlying distribution $p(\mathbf{x})$. In the latter case the training or test data may be used, but note that as before the expert class labels are not used.

3.3.4 Comparative safety

The previous two sections defined the two measures of (conditional) effectiveness and (conditional) comparative accuracy. Both are necessary because neither alone would be sensitive to the two ways in which an explaining classifier can be sub-optimal (not explaining every pattern and sub-optimally classifying some patterns). However this does not imply that these two measures are sufficient to characterise an explaining classifier. This section will first define a measure called *comparative safety*, and then explain how it complements comparative accuracy and effectiveness.

Defining comparative safety

Consider first the ratio of the accuracy² of the explaining classifier $\hat{d}_E(\mathbf{x})$ to the accuracy of the Bayes classifier $\hat{d}_B(\mathbf{x})$. Note that the integrals are over the region of input space \mathcal{C} in which the explaining classifier is activated.

$$\rho_1(\hat{d}_E, \hat{d}_B) = \frac{\int_{\mathcal{C}} P(\hat{d}_E(\mathbf{x}) | \mathbf{x})p(\mathbf{x} | \mathbf{x} \in \mathcal{C})d\mathbf{x}}{\int_{\mathcal{C}} P(\hat{d}_B(\mathbf{x}) | \mathbf{x})p(\mathbf{x} | \mathbf{x} \in \mathcal{C})d\mathbf{x}}. \quad (3.20)$$

This measure has the disadvantage of not comparing the probability of the explained classification of a pattern with the probability of the Bayes classification *of the same pattern*.

²This use of accuracy refers to the standard definition given on page 31, and not the *comparative safety* described above.

This observation suggests the following alternative:

$$\rho_2(\hat{d}_E, \hat{d}_B) = \int_{\mathcal{C}} \frac{P(\hat{d}_E(\mathbf{x}) | \mathbf{x})}{P(\hat{d}_B(\mathbf{x}) | \mathbf{x})} p(\mathbf{x} | \mathbf{x} \in \mathcal{C}) d\mathbf{x}. \quad (3.21)$$

However this is still an *average* measure and so will never indicate the safety of an individual classification. Of primary interest is the class of an individual pattern, and so it is unhelpful to use a weighting based on the unconditional probability density. Hence the final modification removes the averaging effect by replacing the integral by a minimisation. The *comparative safety* of a classifier $\hat{d}_E(\mathbf{x})$ is defined,

$$\rho(\hat{d}_E, \hat{d}_B) = \min_{\mathbf{x} \in \mathcal{C}} \frac{P(\hat{d}_E(\mathbf{x}) | \mathbf{x})}{P(\hat{d}_B(\mathbf{x}) | \mathbf{x})}, \quad (3.22)$$

and thus can be described as a measure which indicates the severity of the worst sub-optimal classification.

Estimating comparative safety

The definition of comparative safety involves the underlying posterior class probability $P(k | \mathbf{x})$. This is unknown and so the obvious solution is to replace it with the estimated probability $\hat{P}(k | \mathbf{x})$,

$$\hat{\rho}(\hat{d}_E, \hat{d}_B) = \min_{\mathbf{x} \in \mathcal{C}} \frac{\hat{P}(\hat{d}_E(\mathbf{x}) | \mathbf{x})}{\hat{P}(\hat{d}_B(\mathbf{x}) | \mathbf{x})}. \quad (3.23)$$

Thus $\hat{\rho}$ is found by minimising a function over \mathcal{C} . The solution to this minimisation problem is a central part of this thesis, although the approach will be an indirect one. This will become clearer, particularly in section 3.6 and in chapter 5.

Note that any method to find $\hat{\rho}$ based upon a sample of patterns is completely inappropriate. The reason for this is simple: comparative safety attempts to quantify *the most* sub-optimal classification, and the point at which this occurs will never be found by chance in a high dimensional space. This is in strong contrast to effectiveness and comparative accuracy which are both essentially *average* measures.

Estimated comparative safety $\hat{\rho}$ has the property that it is always in $[0, 1]$. This is because the ratio of probabilities will always be greater than or equal to zero, and $\hat{d}_B(\mathbf{x})$ is always the class estimated to be the most probable. For this reason, and because there is no alternative, the rest of this thesis will work with the *estimated* comparative safety defined by equation (3.23).

Comparative safety vs comparative accuracy

Comparative safety and comparative accuracy are similar measures which are nevertheless sensitive to different effects. Observe that a comparative accuracy of 1.0 means that all explained classifications are optimal (in the sense that they are identical to those given by the Bayes classifier). This implies that comparative safety also equals 1.0. The reverse is also true: a comparative safety of 1.0 implies a comparative accuracy of 1.0. However, a high comparative accuracy does not necessarily imply high comparative safety. In fact a comparative accuracy only slightly below 1.0 can have a comparative safety of 0.0 since it only takes a single pattern to be assigned to a class with zero probability for the comparative safety to drop to zero. Note that this is independent of the probability of that pattern occurring, and is therefore an example of the importance of decoupling unconditional density from conditional class probability estimates in the model (see section 2.4 on page 42).

Conversely, a comparative safety close to but not equal to 1.0 does not necessarily imply a high comparative accuracy. Consider the following degenerate two class problem for which there is total overlap between the two classes. Suppose that $P(1 | \mathbf{x}) = P(1) = 0.55$ and $P(2 | \mathbf{x}) = P(2) = 0.45$, and that the explaining classifier assigns every pattern to the second class. The comparative accuracy is therefore 0.0 but the comparative safety is quite high at $0.45/0.55 \approx 0.82$.

It is clear from the above two arguments that comparative safety and comparative accuracy measure different effects. Both are therefore useful measures to describe an explaining classifier.

The principle difference between comparative safety and comparative accuracy is that the latter is a function of the unconditional density but the former is not. However comparative safety is also specifically sensitive to how improbable the sub-optimal classifications are. Assigning a pattern \mathbf{x} to class 2 when $P(1 | \mathbf{x}) = 0.55$ is more acceptable than assigning a pattern to class 2 when $P(1 | \mathbf{x}) = 0.95$. Preventing improbable classifications is particularly important in safety critical applications for which it would be far safer to give no classification at all rather than one which is extremely unlikely. In addition, user confidence in such a system could be reduced considerably by the presence of obvious mistakes.

The special case of two classes

Note that if there are only two classes then

$$\frac{\hat{P}(\hat{d}_E(\mathbf{x}) | \mathbf{x})}{\hat{P}(\hat{d}_B(\mathbf{x}) | \mathbf{x})} = \begin{cases} 1 & \text{if } \mathbf{x} \text{ is optimally classified by } \hat{d}_E \\ \frac{1 - \hat{P}(\hat{d}_B(\mathbf{x}) | \mathbf{x})}{\hat{P}(\hat{d}_B(\mathbf{x}) | \mathbf{x})} & \text{if } \mathbf{x} \text{ is suboptimally classified by } \hat{d}_E \end{cases}.$$

From this it follows that

$$\hat{\rho}(\hat{d}_E, \hat{d}_B) = \min_{\hat{d}_E(\mathbf{x}) \neq \hat{d}_B(\mathbf{x})} \frac{1 - \hat{P}(\hat{d}_B(\mathbf{x}) | \mathbf{x})}{\hat{P}(\hat{d}_B(\mathbf{x}) | \mathbf{x})}. \quad (3.24)$$

This implies that a two-class classifier with a comparative safety of r must optimally classify the pattern \mathbf{x} if

$$\hat{P}(\hat{d}_B(\mathbf{x}) | \mathbf{x}) > \frac{1}{1 + r}. \quad (3.25)$$

However $\hat{P}(\hat{d}_B(\mathbf{x}) | \mathbf{x}) = \max_k \hat{P}(k | \mathbf{x})$, so the explaining classifier must satisfy the condition

$$\hat{P}(k | \mathbf{x}) > \frac{1}{1 + r} \implies \text{assign } \mathbf{x} \text{ to class } k. \quad (3.26)$$

Consider the following examples. For maximum comparative safety ($r = 1.0$), the

classifier \hat{d}_E must assign pattern \mathbf{x} to class k if $\hat{P}(k | \mathbf{x}) > 0.5$. In other words, with the exception of arbitrary ties, the classifier \hat{d}_E must make the Bayes classification and $\hat{d}_E(\mathbf{x}) = \hat{d}_B(\mathbf{x})$. Conversely for minimum comparative safety ($r = 0.0$), the classifier \hat{d}_E is only required to assign pattern \mathbf{x} to class k if $\hat{P}(k | \mathbf{x}) > 1.0$. Since this condition never occurs, such a classifier can make completely unconstrained classifications.

Intermediate values of r correspond to intermediate cases of sub-optimal classification. As r is decreased from 1.0, the number of patterns which may be sub-optimally classified increases. However these extra patterns are those which belong to decreasingly ambiguous classes. To clarify this point, observe that a two class classifier with a comparative safety of r can assign to either class when

$$\max \left[\hat{P}(1 | \mathbf{x}), \hat{P}(2 | \mathbf{x}) \right] \leq \frac{1}{1+r}, \quad (3.27)$$

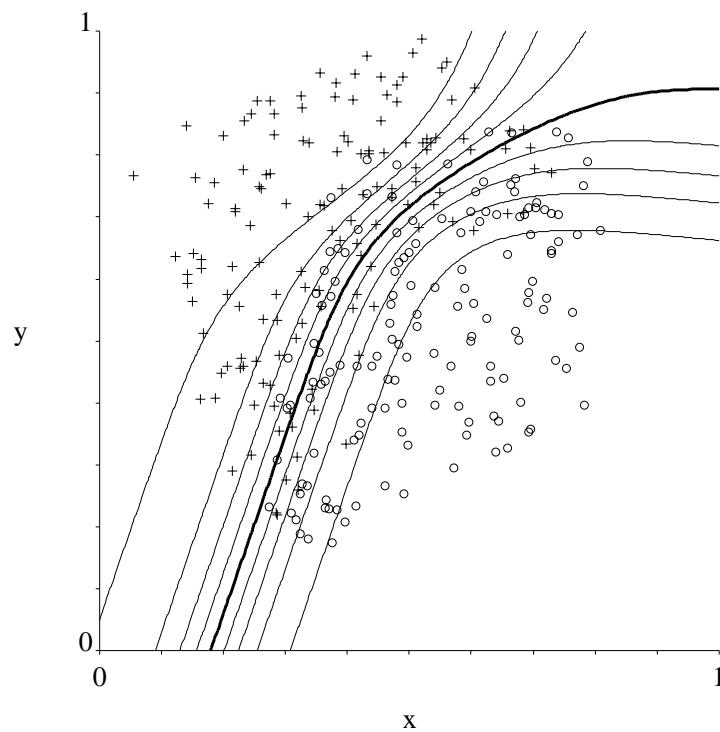
but since $\hat{P}(1 | \mathbf{x}) + \hat{P}(2 | \mathbf{x}) = 1$ this is equivalent to

$$1 - \frac{1}{1+r} \leq \hat{P}(1 | \mathbf{x}) \leq \frac{1}{1+r}. \quad (3.28)$$

This clearly demonstrates that decreasing comparative safety increases the width of a *dead-band*, centred on the optimal classification boundary. Patterns within this dead-band can be assigned to either class, and it is this freedom which may allow an explaining classifier to be more parsimonious and effective whilst still making safe classifications. Figure 3.16 illustrates the effect of an increasing dead-band in two dimensions.

3.3.5 Summary

The measures which will be used to describe an explaining classifier are summarised in table 3.3: parsimony measures the comprehensibility of the explanation; effectiveness measures the probability of being able to explain the classification; comparative accuracy measures the probability that the explained classification will be the same as the optimal



r	class 2 definite	either class	class 1 definite
1.00	$\hat{P}(1 \mathbf{x}) < 0.5$	$0.5 \leq \hat{P}(1 \mathbf{x}) \leq 0.5$	$\hat{P}(1 \mathbf{x}) > 0.5$
0.67	$\hat{P}(1 \mathbf{x}) < 0.4$	$0.4 \leq \hat{P}(1 \mathbf{x}) \leq 0.6$	$\hat{P}(1 \mathbf{x}) > 0.6$
0.43	$\hat{P}(1 \mathbf{x}) < 0.3$	$0.3 \leq \hat{P}(1 \mathbf{x}) \leq 0.7$	$\hat{P}(1 \mathbf{x}) > 0.7$
0.25	$\hat{P}(1 \mathbf{x}) < 0.2$	$0.2 \leq \hat{P}(1 \mathbf{x}) \leq 0.8$	$\hat{P}(1 \mathbf{x}) > 0.8$

Figure 3.16: Illustration of the effect of an increasing dead-band for decreasing comparative safety r . The graph shows contours of $\hat{P}(1|\mathbf{x}) = (0.1, 0.2, \dots, 0.9)$, and the table shows how the region which may be assigned to either class (the dead-band) increase in width as r is decreased. Patterns from class 1 are represented as crosses and patterns from class 2 are represented as circles.

Bayes classification; and comparative safety measures the severity of the most sub-optimal explained classification.

The accuracy (and conditional accuracy) of an explaining classifier is also of interest, but unlike the measures above it is not normalised with respect to the optimal non-explaining classification given by the Bayes classifier $\hat{d}_B(\mathbf{x})$. This normalisation is useful because it is the sub-optimality (in the above sense) of the explaining classifier which needs to be measured.

An alternative justification for not considering accuracy as a primary measure of explanation is based on the observation that a low classification accuracy does not imply (and is not implied by) a low quality of explanation. Problems with significant overlap between the classes are fundamentally difficult to classify accurately, but this does not mean that the class boundaries are difficult to describe with rules. Conversely, a problem which is extremely difficult to explain because of awkward class boundaries can have an extremely accurate Bayes classifier because the classes are well separated.

Note that parsimony is the only measure which has no formal definition, and that with the exception of comparative safety, all the others are defined by integrals over input space.

3.4 The value of the explanation measures

The definition of a set of explanation measures makes the problems of explanation much easier to describe. In particular it enables the compromises which are inherent within an explaining classification system to be better quantified and understood. This leads naturally to a specification.

3.4.1 The inherent compromises of explanation

The fundamental difficulty in explaining classifications is the fact that the class boundaries defined by the Bayes classifier are of a form which usually makes the class regions difficult to describe with only a small number of rules. This constraint prevents the si-

Measure	Symbol	Definition	Integral
parsimony	—	—	—
effectiveness	$\eta(\hat{d}_E)$	$P(\mathbf{X} \in \mathcal{C})$	$\int_{\mathcal{C}} p(\mathbf{x}) d\mathbf{x}$
conditional effectiveness	$\eta_k(\hat{d}_E)$	$P(\mathbf{X} \in \mathcal{C} \mid \hat{d}_B(\mathbf{X}) = k)$	$\int_{\mathcal{C}} p(\mathbf{x} \mid \hat{d}_B(\mathbf{x}) = k) d\mathbf{x}$
comparative accuracy	$\gamma(\hat{d}_E, \hat{d}_B)$	$P(\hat{d}_E(\mathbf{X}) = \hat{d}_B(\mathbf{X}) \mid \mathbf{X} \in \mathcal{C})$	$\int_{\hat{d}_E(\mathbf{x})=\hat{d}_B(\mathbf{x})} p(\mathbf{x} \mid \mathbf{x} \in \mathcal{C}) d\mathbf{x}$
conditional comparative accuracy	$\gamma_k(\hat{d}_E, \hat{d}_B)$	$P(\hat{d}_E(\mathbf{X}) = k \mid \hat{d}_B(\mathbf{X}) = k, \mathbf{X} \in \mathcal{C})$	$\int_{\hat{d}_E(\mathbf{x})=k} p(\mathbf{x} \mid \hat{d}_B(\mathbf{x}) = k, \mathbf{x} \in \mathcal{C}) d\mathbf{x}$
comparative safety	$\rho(\hat{d}_E, \hat{d}_B)$	$\min_{\mathbf{x} \in \mathcal{C}} \frac{P(\hat{d}_E(\mathbf{x}) \mid \mathbf{x})}{P(\hat{d}_B(\mathbf{x}) \mid \mathbf{x})}$	—

Table 3.3: The measures used to describe an explaining classifier which classifies all patterns $\mathbf{x} \in \mathcal{C}$.

multaneous maximisation of the measures. For example, high comparative safety and effectiveness can usually only be achieved with many rules, thus reducing parsimony; high comparative safety and parsimony can usually only be achieved by reducing effectiveness; and achieving high effectiveness and parsimony usually results in a reduced comparative safety. Figure 3.17 demonstrates all three effects.

3.4.2 Specifying an explaining classifier

A simple approach to specifying an explaining classifier is to maximise all the measures. Clearly these maximands need to be weighted since it will not in general be possible simultaneously to achieve the optimal value for them all. The choice of this weighting will depend on the application. For example, acceptable values of conditional effectiveness are a function of the importance of explaining patterns from each class. However in many classification problems it is of primary importance that all patterns are guaranteed to be assigned to a class which is reasonably likely. This places a strong weighting on the comparative safety, which in turn motivates the change of role of comparative safety from a weighted *measure* to a *constraint* in the optimisation of an explaining classifier. Hence in this thesis the problem of obtaining an explaining classifier will involve the weighted maximisation of parsimony, effectiveness and comparative accuracy, *constrained* by a pre-specified comparative safety.

3.5 Exploiting freedom in novel regions

Recall the modified run-time flow diagram of figure 3.14 and observe that the novelty detector is a filter which prevents all novel patterns from reaching either classifier. One consequence of this filtering is that the explaining classifier is free to output an arbitrary class for all novel patterns. This freedom can be exploited to increase the parsimony of the rule-base.

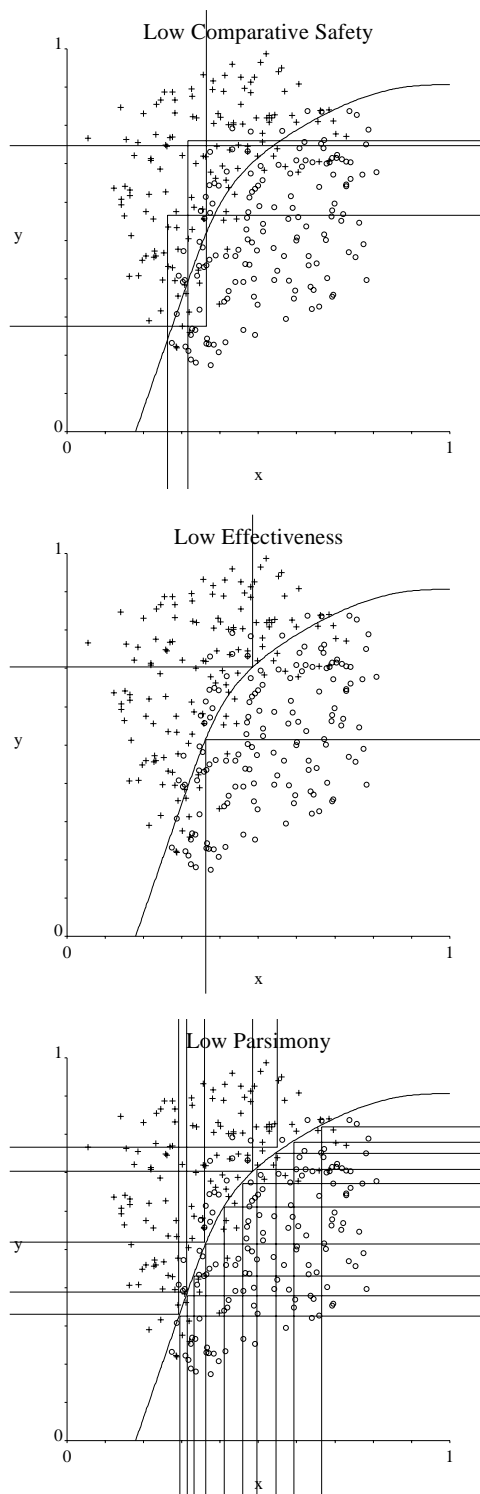


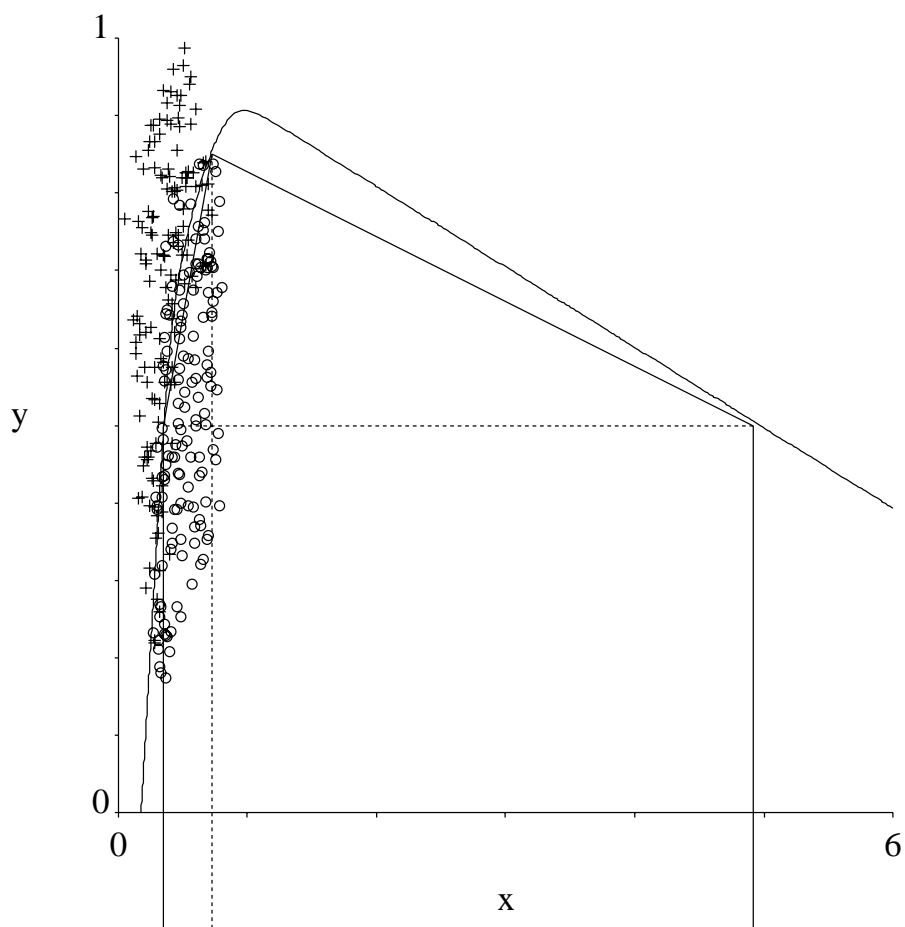
Figure 3.17: Three plots showing the difficulties in simultaneously achieving high parsimony, high effectiveness, and high comparative safety. For example, the first plot shows an attempt to achieve a high parsimony and effectiveness, at the expense of a low comparative safety. Axis-aligned rules are shown here (omitting the position of the prototypes for clarity), but similar effects also occur with city-block and euclidean rules.

It may be argued that this exploitation of freedom in regions of novelty is unacceptable because it could lead to misleading explanations. There are two counter-arguments to this. First, explaining classifiers which are more directly driven by data already make implicit use of this property. The second and more important reply is as follows. Extremely few training patterns are found in regions of novelty, so there is virtually no evidence to suggest that one classification of a novel pattern is better than another. Figure 3.18 provides a two dimensional example of this, showing a zoomed out view of the same data and optimal Bayes boundary which has been used for many of the examples of this chapter. Observe that the boundary curves round far away from the data. It is extremely likely that this curve is an irrelevant artifact of the shaping of the boundary within the cluster of *normal* data. There is no advantage in forcing rules to adhere to the correct side of this boundary outside the region of normality. Indeed there is a definite disadvantage: the rules will be less parsimonious due to an irrelevant constraint. For example, the single city-block rule shown in the figure can be made more parsimonious by removing (replacing by ∞) the rule weighting $u_0 = 4.192$.

3.5.1 The Karnaugh map analogy

The freedom of class assignment in novel and dead-band regions is closely analogous to the “don’t care” states in the Karnaugh maps used to simplify the boolean logic of digital circuits [Kar53]. Karnaugh maps can be used to design digital circuits so that they satisfy a given input/output function. However it is often the case that the output of the circuit for some inputs is unspecified. Choosing convenient outputs for these “don’t care” inputs makes it possible to use a simpler circuit than might otherwise have been achieved if these outputs had been chosen arbitrarily.

Figure 3.19 shows a simple example of a Karnaugh map for three binary variables A,B and C. The task is to design a digital circuit which implements the design constraints of providing a 0 when all of A,B and C are 0 (written $ABC = [000]$), and a 1 when $ABC \in \{[110], [011], [111], [101]\}$. The 0 and 1’s are shown in the map, and the X’s indicate those inputs which will never occur. The designer is free to choose different



Class 1 if (city block)

x	-0.3712	0.7282	+4.1920
y		0.4992	+0.3504

Figure 3.18: Illustration showing that the classification boundary is effectively arbitrary when far away from normal data. Thus there is no advantage in constraining rules in novelty, and this may be exploited in order to increase parsimony.

		AB			
		00	01	11	10
C	0	0	X	1	X
	1	X	1	1	1

Figure 3.19: An example showing how a Karnaugh map may be used to simplify the implementation of a digital circuit. See text for details.

values for each X so as to minimise the number of logic gates needed to implement the circuit. In this example, if the X in the top right corner corresponding to $ABC = [100]$ is chosen to be a 0 and the other two X's chosen to be 1's, then the function is simply $B + C$ and the circuit can be implemented with a single OR gate. If however all the X's are arbitrarily assigned to 0 then the simplest realisation of the function would be $A(B + C) + BC$. This non-parsimonious realisation therefore requires four gates.

So the analogy between a Karnaugh map and an explaining classifier is as follows: with the Karnaugh map, freedom to choose the output of the digital circuit for the “don't care” inputs enables the implementation of a more parsimonious circuit; with the explaining classifier, freedom to choose the class of novel patterns and those patterns in the dead-bands enables the implementation of a more parsimonious explanation.

3.6 Comparative safety regions

Section 3.3.4 defined and justified comparative safety as a *measured* quantity. However the specification given in section 3.4.2 used comparative safety as a fundamental *constraint* in the optimisation of an explaining classifier. It is therefore advantageous to express the comparative safety concept as a set of comparative safety *regions*. A class- k comparative safety- r region, written $\mathcal{R}_k(r)$, is defined as the largest set of patterns which can all be assigned to class k with a comparative safety greater than or equal to r . Comparative safety regions have the obvious property that any class k rule with an antecedent region entirely within $\mathcal{R}_k(r)$ must have a comparative safety greater than or equal to r .

Thus the problem of obtaining an explanation becomes one of finding a parsimonious and effective set of rules which lie completely within the appropriately classed comparative safety regions. It will be seen in chapters 4 and 5 how to achieve this when the class probabilities are estimated using a multi-layer perceptron.

From the above description, and the definition of (estimated) comparative safety given by equation (3.23), the region $\mathcal{R}_k(r)$ is the largest region satisfying

$$\min_{\mathbf{x} \in \mathcal{R}_k(r)} \frac{\hat{P}(k | \mathbf{x})}{\hat{P}(\hat{d}_B(\mathbf{x}) | \mathbf{x})} = r. \quad (3.29)$$

This is equivalent to the more direct and intuitive definition,

$$\mathcal{R}_k(r) = \left\{ \mathbf{x} \mid \frac{\hat{P}(k | \mathbf{x})}{\hat{P}(\hat{d}_B(\mathbf{x}) | \mathbf{x})} \geq r \right\}. \quad (3.30)$$

Note that the regions $\mathcal{R}_k(r)$ for different classes k will necessarily intersect unless $r = 1.0$. It is these intersections which give the explaining classifier the freedom to be optimised for parsimony in only those regions which can be assigned to a sub-optimal class with comparative safety. In other words, the intersection between the safety regions correspond to the dead-bands which were described in section 3.3.4.

It was justified in section 3.5 why a rule-base can assign any novel pattern to any class. It is advantageous to include this idea into the working definition of each comparative safety region, and so a more complete definition is actually given by

$$\mathcal{R}_k(r) = \bar{\mathcal{N}} \cup \mathcal{N} \cap \left\{ \mathbf{x} \mid \frac{\hat{P}(k | \mathbf{x})}{\hat{P}(\hat{d}_B(\mathbf{x}) | \mathbf{x})} \geq r \right\}, \quad (3.31)$$

where \mathcal{N} is the set of all normal patterns. Note that with this modification, comparative safety regions intersect in two different ways. The first type of intersection is in regions of novelty (very low unconditional density), and the second type of intersection is in regions of ambiguous class (no dominating class posterior probability). Both of these overlaps can be used to increase the parsimony and effectiveness of the rule-base.

3.7 Summary

This chapter began by defining an explanation as a small number of rules, each of which describes a simple region of input space. Three types of simple regions were considered, each based upon a different (asymmetric) distance to a prototype pattern.

It was then argued that the need for explanation should not be used to constrain the class posterior probability estimates. Instead a decoupled approach was proposed: an explanation should describe the optimal Bayes classification, where the Bayes classification is based upon explanation-unbiased posterior probability estimates.

This decoupling allows the fundamental problem of explanation to be made explicit: it is not generally possible to describe the Bayes classification using only a small number of simple rules because the rule antecedent regions cannot be made to fit neatly against the classification boundary. One partial solution given was to allow the rules to make sub-optimal classifications. However it was pointed out that this sub-optimality must be carefully controlled if the rule-base is to cover a reasonable number of patterns without making improbable classifications. Hence a set of measures (parsimony, effectiveness, comparative accuracy, and comparative safety) were defined to indicate the closeness between the Bayes classifier and the explaining classifier.

The explanation measures enable a more precise understanding of the issues involved in obtaining explaining classifiers. Hence they were used to specify the ideal characteristics of an explaining classifier. It was argued that the concept of comparative safety was so fundamental to explaining classifiers that it should be used to form a constraint in their optimisation. Thus the comparative safety measure was re-expressed as comparative safety regions.

Finally, it was explained that rules need only be on the correct side of the Bayes classification boundary within the region of normality. This provides an additional freedom which can be exploited to increase the parsimony of the rule-base. The definition of the comparative safety regions were modified to incorporate this idea.

Chapter 4

Linearising an MLP

This chapter and the next together describe a method of extracting rules under the explanation framework previously introduced. This division into two chapters reflects the top level structure of the solution: first generate a piece-wise linearisation of the trained multi-layer perceptron (MLP); then use this piece-wise linearisation to obtain rules. The justification for this approach is that no method could be found which obtained satisfactory rules without using this intermediate linearisation step. Expressed simply, piece-wise linearisation of the MLP enables an explicit description of the classification boundaries to be obtained (with respect to a given comparative safety); rule extraction is made easier by starting from this basis.

To maintain a modular structure, this chapter details the linearisation method in a completely self-contained manner.

4.1 Introduction

The piece-wise linearisation of a one dimensional function $y = f(x)$ is obtained by partitioning an interval of x into segments and approximating the function by a straight line within each segment. Similarly, multi-dimensional functions such as MLPs can be piece-wise linearised by partitioning input space into *cells* and approximating the function by a

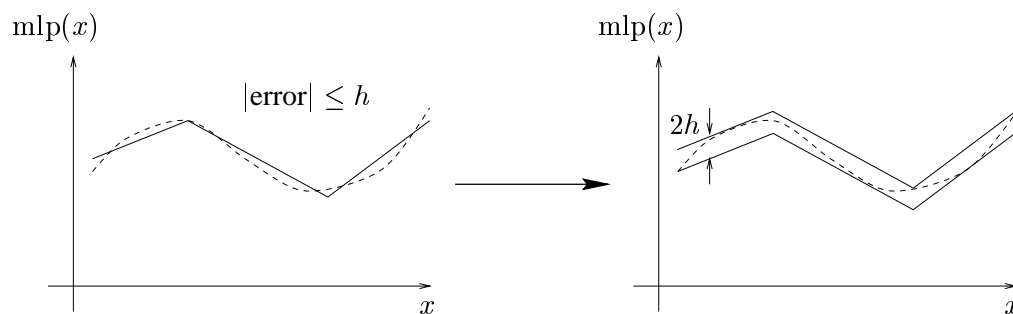


Figure 4.1: Piece-wise linearisation with error bounds can be interpreted as a piece-wise linear bounds on the function.

linear equation within each cell. For this process to be useful it is necessary to be able to compute the largest error caused by the linear approximation. In other words, the piece-wise linearisation must be accompanied by bounds on the error. A useful interpretation of a piece-wise linearisation with error bounds is a piece-wise linear bound on the function. A one-dimensional illustration is shown in figure 4.1.

The solution for the piece-wise linearisation of a MLP given in this chapter will assume the network has a single hidden layer and a single linear output. Recall that an MLP trained for (multiple-class) classification has softmax outputs, so it might appear insufficient only to consider the piece-wise linearisation of an MLP with a single linear output. However this is not the case, although the justification will be postponed until chapter 5.

4.2 Specification

Let \mathbf{s} be a vector of integers, called the cell coordinates, such that each \mathbf{s} refers to a unique cell, $\mathcal{X}_{\mathbf{s}}$. The pattern \mathbf{x} is said to be in cell \mathbf{s} if and only if $\mathbf{x} \in \mathcal{X}_{\mathbf{s}}$. Clearly the cells should not intersect, and should cover all normal patterns, \mathcal{N} . Hence

$$\mathcal{X}_{\mathbf{s}_1} \cap \mathcal{X}_{\mathbf{s}_2} = \emptyset \quad \text{if } \mathbf{s}_1 \neq \mathbf{s}_2, \quad (4.1)$$

$$\bigcup_{\mathbf{s}} \mathcal{X}_{\mathbf{s}} \supseteq \mathcal{N}. \quad (4.2)$$

A multi-layer perceptron with a single hidden layer and a single linear output may be

written

$$\text{mlp}(\mathbf{x}) = \sum_i w_i^{\text{OUT}} \text{sig}(y_i) + c^{\text{OUT}} \quad (4.3)$$

$$\mathbf{y} = \mathbf{W}^{\text{IN}} \mathbf{x} + \mathbf{c}^{\text{IN}} \quad (4.4)$$

where the sigmoid function is defined

$$\text{sig}(y_i) = \frac{1}{1 + \exp(-y_i)}. \quad (4.5)$$

The linearisation of a cell s may then be written

$$\left. \begin{aligned} \text{mlp}(\mathbf{x}) &= \mathbf{a}_s^T \mathbf{x} + b_s + e_s(\mathbf{x}) \\ |e_s(\mathbf{x})| &\leq h_s \end{aligned} \right\} \quad \text{for all } \mathbf{x} \in \mathcal{X}_s, \quad (4.6)$$

which is equivalent to saying that for all $\mathbf{x} \in \mathcal{X}_s$ the mlp function is bounded:

$$\mathbf{a}_s^T \mathbf{x} + b_s - h_s \leq \text{mlp}(\mathbf{x}) \leq \mathbf{a}_s^T \mathbf{x} + b_s + h_s. \quad (4.7)$$

This can also be expressed more concisely using the notation,

$$\text{mlp}(\mathbf{x}) \in \mathbf{a}_s^T \mathbf{x} + b_s + [-h_s, h_s] \quad \text{for all } \mathbf{x} \in \mathcal{X}_s. \quad (4.8)$$

Hence the problem of piece-wise linearising the MLP is to obtain a set of cells \mathcal{X}_s which satisfy (4.1) & (4.2), and a set of parameters \mathbf{a}_s, b_s, h_s which satisfy equation (4.8).

4.3 Solution

The key observation which enables the MLP to be linearised efficiently is that the separate piece-wise linearisation of each sigmoid (one per hidden node) *implies* a piece-wise linearisation of the whole MLP function.

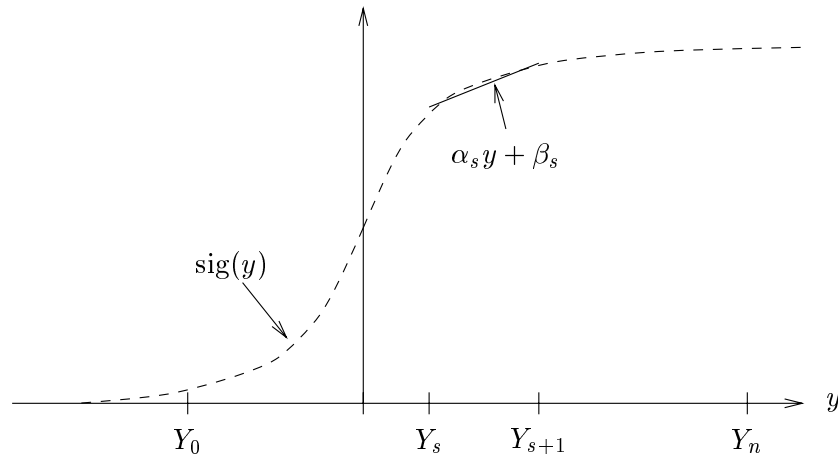


Figure 4.2: The piece-wise linearisation of a sigmoid function. For clarity only one of the n segments used to linearise the sigmoid is shown.

4.3.1 Main derivation

To maintain the flow of the derivation, assume for the moment that a method to obtain a piece-wise linearisation of a sigmoid function has been found. In other words assume that it is known how to determine α_s , β_s , h_s and Y_s such that

$$\left. \begin{aligned} \text{sig}(y) &= \alpha_s y + \beta_s + \epsilon_s(y) \\ |\epsilon_s(y)| &\leq h_s \end{aligned} \right\} \text{ for all } y \in [Y_s, Y_{s+1}], \quad (4.9)$$

where the integer s indicates the segment being linearised. This linearisation is illustrated in figure 4.2.

To keep track of the linearisation of each sigmoid the parameters needs to be indexed by i , the hidden unit index:

$$\left. \begin{aligned} \text{sig}(y_i) &= \alpha_{i,s_i} y_i + \beta_{i,s_i} + \epsilon_{i,s_i}(y_i) \\ |\epsilon_{i,s_i}(y_i)| &\leq h_{i,s_i} \end{aligned} \right\} \text{ for all } y_i \in [Y_{i,s_i}, Y_{i,s_i+1}]. \quad (4.10)$$

Substituting equation (4.10) into (4.3) gives the following:

$$\begin{aligned}
\text{mlp}(\mathbf{x}) &= \sum_i w_i^{\text{OUT}} \text{sig}(y_i) + c^{\text{OUT}} \\
&= \sum_i w_i^{\text{OUT}} [\alpha_{i,s_i} y_i + \beta_{i,s_i} + \epsilon_{i,s_i}(y_i)] + c^{\text{OUT}} \\
&= \left[\sum_i (w_i^{\text{OUT}} \alpha_{i,s_i}) y_i \right] + \left[\sum_i w_i^{\text{OUT}} \beta_{i,s_i} + c^{\text{OUT}} \right] + \left[\sum_i w_i^{\text{OUT}} \epsilon_{i,s_i}(y_i) \right] \\
&= \boldsymbol{\alpha}_s^T \mathbf{y} + \beta_s + \epsilon_s(\mathbf{y}), \tag{4.11}
\end{aligned}$$

where $\boldsymbol{\alpha}_s$, β_s , and $\epsilon_s(\mathbf{y})$ are defined

$$\begin{aligned}
(\boldsymbol{\alpha}_s)_i &= w_i^{\text{OUT}} \alpha_{i,s_i} \\
\beta_s &= \sum_i w_i^{\text{OUT}} \beta_{i,s_i} + c^{\text{OUT}} \\
\epsilon_s(\mathbf{y}) &= \sum_i w_i^{\text{OUT}} \epsilon_{i,s_i}(y_i).
\end{aligned}$$

Using equation (4.4), this may be re-written,

$$\begin{aligned}
\text{mlp}(\mathbf{x}) &= \boldsymbol{\alpha}_s^T (\mathbf{W}^{\text{IN}} \mathbf{x} + \mathbf{c}^{\text{IN}}) + \beta_s + \epsilon_s(\mathbf{y}) \\
&= (\boldsymbol{\alpha}_s^T \mathbf{W}^{\text{IN}}) \mathbf{x} + (\boldsymbol{\alpha}_s^T \mathbf{c}^{\text{IN}} + \beta_s) + \epsilon_s(\mathbf{y}) \\
&= \mathbf{a}_s^T \mathbf{x} + b_s + \epsilon_s(\mathbf{y}), \tag{4.12}
\end{aligned}$$

where \mathbf{a}_s and b_s are defined

$$\begin{aligned}
\mathbf{a}_s &= \boldsymbol{\alpha}_s^T \mathbf{W}^{\text{IN}}, \\
b_s &= \boldsymbol{\alpha}_s^T \mathbf{c}^{\text{IN}} + \beta_s,
\end{aligned}$$

thus giving an expression for the linearisation of the MLP within cell s . To obtain bounds

on the error term $\epsilon_s(\mathbf{y})$, first define \mathcal{Y}_s by

$$\mathbf{y} \in \mathcal{Y}_s \equiv \bigwedge_i y_i \in [Y_{i,s_i}, Y_{i,s_i+1}] \quad (4.13)$$

and then observe

$$\begin{aligned} \max_{\mathbf{y} \in \mathcal{Y}_s} |\epsilon_s(\mathbf{y})| &= \max_{\mathbf{y} \in \mathcal{Y}_s} \left| \sum_i w_i^{\text{OUT}} \epsilon_{i,s_i}(y_i) \right| \\ &\leq \sum_i \max_{y_i \in [Y_{i,s_i}, Y_{i,s_i+1}]} |w_i^{\text{OUT}} \epsilon_{i,s_i}(y_i)| \\ &= \sum_i |w_i^{\text{OUT}}| h_{i,s_i} \end{aligned} \quad (4.14)$$

Hence for all $\mathbf{y} \in \mathcal{Y}_s$,

$$|\epsilon_s(\mathbf{y})| \leq h_s, \quad \text{where } h_s = \sum_i |w_i^{\text{OUT}}| h_{i,s_i}. \quad (4.15)$$

Finally equation (4.4) allows the cell constraint to be transformed from \mathbf{y} -space to \mathbf{x} -space. Define \mathcal{X}_s by

$$\mathbf{x} \in \mathcal{X}_s \equiv \mathbf{W}^{\text{IN}} \mathbf{x} + \mathbf{c}^{\text{IN}} \in \mathcal{Y}_s. \quad (4.16)$$

Observe that because the \mathcal{Y}_s 's are mutually exclusive, it follows from this definition that the \mathcal{X}_s 's are also mutually exclusive.

4.3.2 Summary of main derivation

The linearisation method is summarised in table 4.1. Note that the more compact notation of equation (4.8) has been used to avoid the need for the error terms $\epsilon_s(\mathbf{x})$, $\epsilon_{i,s_i}(y_i)$ and $\epsilon_s(\mathbf{y})$. It is very clear from this summary that the key equation is the piece-wise linearisation of the sigmoid function. This equation ultimately determines the linear coefficients \mathbf{a}_s & b_s , the bounds h_s , the cells \mathcal{X}_s , and the cell coordinate system \mathbf{s} .

<p>if</p> $\text{mlp}(\mathbf{x}) = \sum_i w_i^{\text{OUT}} \text{sig}(y_i) + c^{\text{OUT}}$ $\mathbf{y} = \mathbf{W}^{\text{IN}} \mathbf{x} + \mathbf{c}^{\text{IN}}$ <p>then</p> $\text{mlp}(\mathbf{x}) \in \mathbf{a}_s^T \mathbf{x} + b_s + [-h_s, h_s] \quad \text{for all } \mathbf{x} \in \mathcal{X}_s$ <p>where</p> $\mathbf{a}_s^T = \boldsymbol{\alpha}_s^T \mathbf{W}^{\text{IN}}$ $b_s = \boldsymbol{\alpha}_s^T \mathbf{c}^{\text{IN}} + \beta_s$ $h_s = \sum_i w_i^{\text{OUT}} h_{i,s_i}$ $(\boldsymbol{\alpha}_s)_i = w_i^{\text{OUT}} \alpha_{i,s_i}$ $\beta_s = \sum_i w_i^{\text{OUT}} \beta_{i,s_i} + c^{\text{OUT}}$ $\text{sig}(y_i) \in \alpha_{i,s_i} y + \beta_{i,s_i} + [-h_{i,s_i}, h_{i,s_i}] \quad \text{for all } y_i \in [Y_{i,s_i}, Y_{i,s_i+1}]$ $\mathbf{y} \in \mathcal{Y}_s \equiv \bigwedge_i y_i \in [Y_{i,s_i}, Y_{i,s_i+1}]$ $\mathbf{x} \in \mathcal{X}_s \equiv \mathbf{W}^{\text{IN}} \mathbf{x} + \mathbf{c}^{\text{IN}} \in \mathcal{Y}_s$

Variable	Description
i	Integer indexing a hidden unit or its sigmoid.
\mathbf{s}	Vector of segment indexes denoting a cell.
\mathbf{a}_s	Linear coefficients of MLP approximation in cell \mathbf{s} .
b_s	Linear offset of MLP approximation in cell \mathbf{s} .
h_s	Error bound of MLP approximation in cell \mathbf{s} .
α_{i,s_i}	Line slope for sigmoid i in segment s_i .
β_{i,s_i}	Line offset for sigmoid i in segment s_i .
h_{i,s_i}	Error bound for sigmoid i in segment s_i .
$[Y_{i,s_i}, Y_{i,s_i+1}]$	Linearisation interval for sigmoid i in segment s_i .
\mathcal{Y}_s	Set of points defining cell \mathbf{s} in y -space.
\mathcal{X}_s	Set of points defining cell \mathbf{s} in x -space.

Table 4.1: Obtaining piece-wise linear bounds of an MLP from the piece-wise linear bounds of each sigmoid.

4.3.3 Some preliminaries to sigmoid linearisation

The piece-wise linearisation of the MLP should cover the whole of normal space with a linearisation error less than some maximum specified amount. From this specification, the maximum linearisation error and linearisation interval of each sigmoid can be derived. The calculation of each is explained below.

Using cells efficiently . . .

A natural definition of MLP-linearisation error is the largest error bound $h_{\mathbf{s}}$, evaluated over all cells:

$$h = \max_{\mathbf{s}} h_{\mathbf{s}}. \quad (4.17)$$

It is intuitively obvious that equalising all the $h_{\mathbf{s}}$'s will make the most efficient use of cells. Note (table 4.1) that each error bound $h_{\mathbf{s}}$ is the weighted sum of an error bound h_{i,s_i} from each sigmoid i ,

$$h_{\mathbf{s}} = \sum_i |w_i^{\text{OUT}}| h_{i,s_i}. \quad (4.18)$$

Hence the $h_{\mathbf{s}}$'s can be made equal by making the contributions from the linearisation of each sigmoid all equal and independent of the cell coordinate s_j . In other words,

$$h_{i,s_i} = \frac{h}{|w_i^{\text{OUT}}|} \quad \text{for all } s_i. \quad (4.19)$$

Hence if the linearisation error is required to be no more than a given value H , then the optimal solution will be made from the smallest number of cells which satisfy

$$h_{i,s_i} \leq H_i \quad \text{for all } s_i \quad (4.20)$$

where H_i is the maximum linearisation error for each sigmoid,

$$H_i = \frac{H}{|w_i^{\text{OUT}}|}. \quad (4.21)$$

... to cover all of normal space

The interval over which each sigmoid is linearised must be such that the resulting MLP-linearisation covers the whole of normality, \mathcal{N} . Observe that the input to the i^{th} sigmoid, y_i , is proportional to the distance from a hyperplane defined by the i^{th} row of \mathbf{W}^{IN} , written $(\mathbf{w}_i^{\text{IN}})^T$, and the i^{th} element of \mathbf{c}^{IN} , written c_i^{IN} :

$$y_i = (\mathbf{w}_i^{\text{IN}})^T \mathbf{x} + c_i^{\text{IN}}. \quad (4.22)$$

Hence the interval over which it is necessary for the i^{th} sigmoid to be linearised can be expressed

$$y_i \in \left[\min_{\mathbf{x} \in \mathcal{N}} (\mathbf{w}_i^{\text{IN}})^T \mathbf{x} + c_i^{\text{IN}}, \max_{\mathbf{x} \in \mathcal{N}} (\mathbf{w}_i^{\text{IN}})^T \mathbf{x} + c_i^{\text{IN}} \right]. \quad (4.23)$$

Recall (section 2.2.2) that normality is defined in terms of a threshold on the estimated unconditional density function, $\hat{p}(\mathbf{x})$. This function is often complex, which makes the interval of equation (4.23) difficult to compute. A simple and sufficient alternative is to linearise over a superset of normality, $\mathcal{N}_{\square} \supseteq \mathcal{N}$, where this superset is an axis-aligned hyper-rectangle in input space,

$$\mathcal{N}_{\square} = \left\{ \mathbf{x} \mid \bigwedge_j p_j \leq x_j \leq q_j \right\}. \quad (4.24)$$

The parameters p_j, q_j of this superset are easy to find by considering the marginal distributions of \mathbf{x} .

Thus a sufficient linearisation interval for the i^{th} sigmoid may be written

$$y_i \in \left[\min_{\mathbf{x} \in \mathcal{N}_\square} (\mathbf{w}_i^{\text{IN}})^T \mathbf{x} + c_i^{\text{IN}}, \max_{\mathbf{x} \in \mathcal{N}_\square} (\mathbf{w}_i^{\text{IN}})^T \mathbf{x} + c_i^{\text{IN}} \right]. \quad (4.25)$$

The advantage of this form is that the maximisation and minimisation are trivial to compute because the \mathcal{N}_\square constraint allows each x_i to be optimised independently. Hence the interval can be expressed as $y_i \in [\min_i, \max_i]$ where \min_i and \max_i are defined

$$\begin{aligned} \min_i &= \sum_j M((\mathbf{w}_i^{\text{IN}})_j, p_j, q_j) + c_i^{\text{IN}} \\ \max_i &= \sum_j M((\mathbf{w}_i^{\text{IN}})_j, q_j, p_j) + c_i^{\text{IN}} \end{aligned} \quad (4.26)$$

and where the function M is defined

$$M(\gamma, a, b) = \begin{cases} \gamma a & \text{if } \gamma \geq 0 \\ \gamma b & \text{if } \gamma < 0 \end{cases}. \quad (4.27)$$

4.3.4 Sigmoid linearisation

This section describes a method of generating a piece-wise linearisation of a sigmoid function within a specified interval to within a specified linearisation error. Many alternative methods were considered, often consisting of minor variations which had a significant effect on performance. Hence in order to make the work in this thesis reproducible it is important to describe the algorithm in some detail. It was decided that a description using pseudo-code would achieve the greatest clarity.

The first two sections below summarise the notation and give the complete pseudo-code. The next 5 sections give a bottom-up explanation of the algorithm followed by a summary of the algorithm structure. Finally the last two sections present an assessment of the method and also discuss an important property.

Notation

The input to the algorithm consists of the linearisation interval $[\min_i, \max_i]$ and the maximum linearisation error H_i . These would be calculated using equations (4.26) and (4.21). The output from the algorithm is the piece-wise linearisation described by the set of vectors $\{\mathbf{Y}_i, \boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \mathbf{h}_i\}$. This set consists of the segment intervals \mathbf{Y}_i , the line coefficients $\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i$, and the linearisation error achieved, \mathbf{h}_i . These variables will satisfy the following property for all segments s_i :

$$\text{sig}(y_i) \in \alpha_{i,s_i}y + \beta_{i,s_i} + [-h_{i,s_i}, h_{i,s_i}] \quad \text{for all } y \in [Y_{i,s_i}, Y_{i,s_i+1}]. \quad (4.28)$$

For clarity, explicit reference to each sigmoid (using the index i) will now be dropped.

Pseudo-code for the complete algorithm

```
function linearise1(min, max, H)
    // Linearise sigmoid over interval [min, max] with an error bound less than H.
    // Assume min < max.

    if (min ≥ 0)
        {Y, α, β, h} = linearise2(min, max, H)
    else if (max ≤ 0)
        {Y, α, β, h} = rotate(linearise2(-max, -min, H))
    else if (min < 0 and max > 0)
        {Y, α, β, h} = merge(rotate(linearise2(0, -min, H)),
                             linearise2(0, max, H))

return {Y, α, β, h}
```

```
function rotate({Y, α, β, h})
    // Rotate a piece-wise linearisation about (0, 0.5).
    // Assume Y0 ≥ 0.

    Y = reverse(Y) // reverse reverses the order of the elements.
    α = reverse(α)
    β = reverse(β)
    h = reverse(h)
```

```

Y = -Y
for  $s = 1$  to  $n$ 
     $\beta_s = 1 - \beta_s$ 

return {Y,  $\alpha$ ,  $\beta$ , h}

```

```

function merge({Y1,  $\alpha_1$ ,  $\beta_1$ , h1}, {Y2,  $\alpha_2$ ,  $\beta_2$ , h2})
    // Merge two piece-wise linearisations which are assumed to join at (0, 0.5).

     $\alpha^T = (\alpha_1^T \ \alpha_2^T)$ 
     $\beta^T = (\beta_1^T \ \beta_2^T)$ 
     $\mathbf{h}^T = (\mathbf{h}_1^T \ \mathbf{h}_2^T)$ 

    // The last element of Y1 and the first element of Y2 are both 0; only keep one of them.
    YT = (Y1T tail(Y2T)) // tail discards the first element of a vector.
return {Y,  $\alpha$ ,  $\beta$ , h}

```

```

function linearise2(min, max,  $H$ )
    // Linearise a sigmoid over the interval [min, max] with an error bound less than  $H$ .
    // Assume  $0 \leq \min < \max$ .

     $n = 0$ 
do
    increment  $n$ 
    {Y,  $\alpha$ ,  $\beta$ , h} = linearise3(min, max,  $n$ )
while (max  $h_i > H$ )
return {Y,  $\alpha$ ,  $\beta$ , h}

```

```

function linearise3(min, max,  $n$ )
    // Linearise a sigmoid over the interval [min, max] using  $n$  segments.
    // Assume  $0 \leq \min < \max$ .

    // ***** local variables *****
vector     $\alpha$ ,  $\beta$ 
vector    h
vector    Y
constant  tolerance =  $1 \times 10^{-6}$ 

```



```

// ***** local functions *****
function initialise()
    // Initialise segment spacings  $Y$  with sensible (but sub-optimal) values.

     $Y_0 = \min$ 
     $Y_{n+1} = \max$ 
    for  $i = 1$  to  $n$ 
         $Y_i = \text{sig}((\text{sig}^{-1}(\max) - \text{sig}^{-1}(\min)) \times \frac{i}{n+1})$ 

    // Optimise  $\alpha, \beta$ , and  $h$  for these spacings.
    for  $s = 1$  to  $n$ 
        optimiseSegment( $s$ )
return

function balance( $s$ )
    // Use a binary search on  $Y_s$  to equalise the error bounds  $h_s$  and  $h_{s+1}$ .

    left =  $Y_s$ 
    right =  $Y_{s+2}$ 
    while ( $|h_s - h_{s+1}| > \text{tolerance} \times \min_s h_s$ )
        if ( $h_{s+1} > h_s$ )
            left =  $Y_{s+1}$ 
        else
            right =  $Y_{s+1}$ 
         $Y_{s+1} = (\text{left} + \text{right})/2$ 
        optimiseSegment( $s$ )
        optimiseSegment( $s + 1$ )
return

function optimiseSegment( $s$ )
    // Compute optimal  $\alpha_s, \beta_s$  and  $h_s$  for segment  $[Y_s, Y_{s+1}]$ .

    if ( $Y_s > 0$ )
        if ( $Y_s > 10$ )
             $\alpha_s = 0$ 
             $\beta_s = (\text{sig}(Y_s) + \text{sig}(Y_{s+1}))/2$ 
             $h_s = \beta_s - \text{sig}(Y_s)$ 
        else
             $\alpha_s = (\text{sig}(Y_{s+1}) - \text{sig}(Y_s))/(Y_{s+1} - Y_s)$ 
             $t = \text{sig}^{-1}((1 + \sqrt{1 - 4\alpha_s})/2)$ 
             $\beta_s = (\text{sig}(Y_s) + \text{sig}(t) - \alpha_s(Y_s + t))/2$ 
             $h_s = \alpha_s Y_s + \beta_s - \text{sig}(Y_s)$ 
    else
        //  $Y_s == 0$ 
         $t = 0$ 
        for  $i = 1$  to 20
             $\alpha_s = (\text{sig}(Y_{s+1} + \text{sig}(t)) - 1)/(Y_{s+1} + t)$ 

```

```

        t = sig-1((1 + √(1 - 4αs))/2)
        βs = 0.5
        hs = sig(t) - αst - βs
return

// ***** main loop *****
initialise()
do
    s = argmaxs |hs - hs+1|
    difference = |hs - hs+1|

    balance(s)
while (difference > tolerance × mins hs)

return {Y, α, β, h}

```

The core function, `optimiseSegment`

The most basic unit of the algorithm is a function which computes the line which optimally approximates the sigmoid within a single segment. This function is called `optimiseSegment(s)` where s indicates the segment to be linearised. The callees of this function will ensure that $Y_s \geq 0$. The two cases $Y_s > 0$ and $Y_s = 0$ will be considered separately.

Assume first that the interval is strictly positive, $Y_s > 0$. The optimal configuration for a line which minimises the maximum linearisation error is shown in figure 4.3. By considering the error at the end points Y_s and Y_{s+1} ,

$$h_s = \alpha_s Y_s + \beta_s - \text{sig}(Y_s), \quad (4.29)$$

$$h_s = \alpha_s Y_{s+1} + \beta_s - \text{sig}(Y_{s+1}). \quad (4.30)$$

Subtracting one equation from the other gives an expression for the slope,

$$\alpha_s = \frac{\text{sig}(Y_{s+1}) - \text{sig}(Y_s)}{Y_{s+1} - Y_s}. \quad (4.31)$$

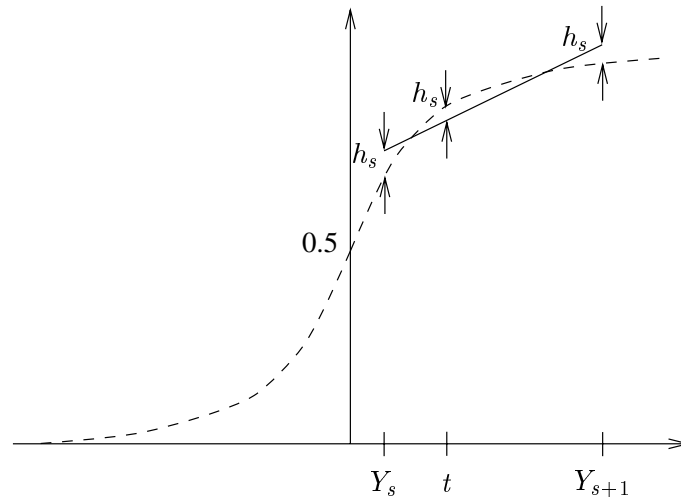


Figure 4.3: Optimal configuration for minimising the maximum difference between a line and sigmoid when $Y_s > 0$.

The maximum linearisation error over the segment occurs at the turning point t , defined by

$$\frac{\partial}{\partial t} (\alpha_s t + \beta_s - \text{sig}(t)) = 0, \quad (4.32)$$

which can be solved to give

$$t = \text{sig}^{-1} \left(\frac{1 + \sqrt{1 - 4\alpha_s}}{2} \right). \quad (4.33)$$

The condition for the magnitude of the error to equal h_s at this turning point is

$$-h_s = \alpha_s + \beta_s - \text{sig}(t), \quad (4.34)$$

which when added to equation (4.29) gives the following expression for β_s :

$$\beta_s = \frac{\text{sig}(Y_s) + \text{sig}(t) - \alpha_s(Y_s + t)}{2}. \quad (4.35)$$

Finally, substitution back into equation (4.29) can be used to determine h_s , thus completing the calculation of the optimum single line segment. The whole calculation is well conditioned so long as the slope α_s is not too small, in which case the calculation of the

turning point t becomes extremely sensitive to round-off error. This is most easily seen by considering the absolute condition number,

$$\kappa = \left| \frac{\partial t}{\partial \alpha_s} \right| = \frac{1}{\alpha_s \sqrt{1 - 4\alpha_s}}, \quad (4.36)$$

from which it can be seen that determining t is very sensitive to changes in α_s when α_s is small.

Equation 4.31 indicates that the problem of small α occurs when $\text{sig}(Y_s)$ and $\text{sig}(Y_{s+1})$ are both saturated to unity, and this occurs when Y_s is large. Hence the problem can be avoided by defining α_s to equal zero for sufficiently large Y_s . This implies that the turning point of error no longer exists, and so the equations become

$$\begin{aligned} \alpha_s &= 0 \\ \beta_s &= \frac{\text{sig}(Y_s) + \text{sig}(Y_{s+1})}{2} \\ h_s &= \beta_s - \text{sig}(Y_s). \end{aligned} \quad (4.37)$$

Since $\text{sig}(10) = 0.9999546$, sufficiently large Y_s can be taken to mean $Y_s > 10$.

Now consider the case when the interval begins at the origin, $Y_s = 0$. It is convenient to impose the constraint that the linearisation of a sigmoid intersects the centre of rotational symmetry of the sigmoid, $(0, 0.5)$. Hence when $Y_s = 0$ the configuration shown in figure 4.3 is replaced by the configuration shown in figure 4.4. The relevant equations are

$$\begin{aligned} \alpha_s &= \frac{\text{sig}(Y_{s+1}) + h_s - 0.5}{Y_{s+1}} \\ \beta_s &= 0.5 \\ t &= \text{sig}^{-1} \left(\frac{1 + \sqrt{1 - 4\alpha_s}}{2} \right) \\ -h_s &= \alpha_s t + \beta_s - \text{sig}(t), \end{aligned} \quad (4.38)$$

but unfortunately these do not have a closed-form solution. However it is straightforward

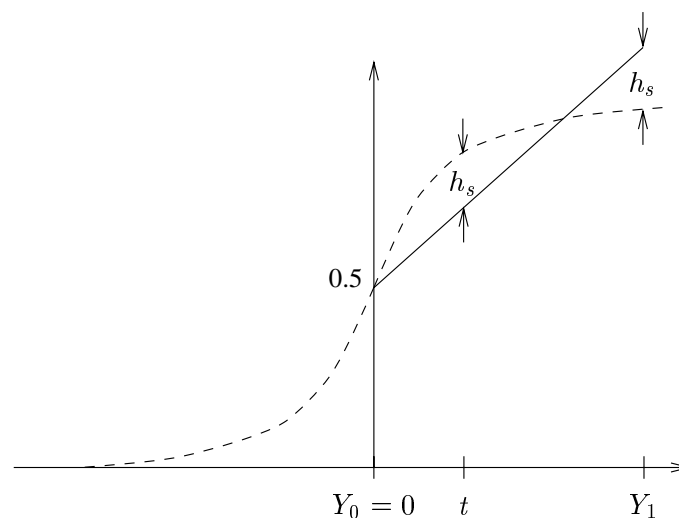


Figure 4.4: Optimal configuration for minimising the maximum difference between a line and a sigmoid when the line is required to pass through the point $(0, 0.5)$.

to derive the following numerical solution: initialise t to 0, and iteratively evaluate

$$\alpha_s = \frac{\text{sig}(Y_{s+1}) + \text{sig}(t) - 1}{Y_{s+1} + t},$$

$$t = \text{sig}^{-1}\left(\frac{1 + \sqrt{1 - 4\alpha_s}}{2}\right). \quad (4.39)$$

In practice this scheme always converges within a dozen or so iterations.

The function `linearise3`

Optimal segment spacing is achieved when all the segments have the same linearisation error. The approach taken here is to find the two adjacent segments, $[Y_s, Y_{s+1}]$ and $[Y_{s+1}, Y_{s+2}]$, which have the largest difference in linearisation error, $\max_s |h_s - h_{s+1}|$, and to use a binary search to find the mid-point, Y'_s , which equalises h_s and h_{s+1} . The function which implements this binary search is called `balance(s)`.

Before the binary search can be called for the first time, the segment spacings need to be initialised. This is achieved by the function `initialise` which positions the segments such that $\text{sig}(Y_s)$ are equi-spaced over s .

The function `linearise3` is the only caller of the functions `initialise`, `balance`, and `optimiseSegment`. It takes as arguments the interval over which the sigmoid is to be linearised, $[\min, \max]$, and the number of segments to be used, n . Note that because `optimiseSegment` only works on intervals which are entirely positive, `linearise3` assumes $\min \geq 0$.

The function `linearise2`

The key to understanding the function `linearise2` is to compare its arguments with the arguments of the function `linearise3`. The former takes an interval to be linearised and a maximum error bound, whereas the latter takes an interval to be linearised and the number of segments to be used. Translating maximum error to the minimum number of segments is the sole purpose of `linearise2`. The function works by calling `linearise3` with an increasing number of segments until the resulting linearisation error becomes less than the specified amount.

Note that the linearisation interval is still assumed to be entirely positive.

The function `linearise1`

Finally the function `linearise1` removes the constraint that the linearisation interval must be entirely positive. There are three distinct cases: the linearisation interval is entirely positive ($\min \geq 0$); the linearisation interval is entirely negative ($\max \leq 0$); and the linearisation interval straddles the origin ($\min < 0$ and $\max > 0$). Two functions are used to enable all three cases to be solved by routines which require $\min \geq 0$. The first of these is called `rotate` which rotates a linearisation about the point of rotational symmetry of a sigmoid function, $(0, 0.5)$. The second is called `merge` which merges a linearisation over an interval from some negative value to zero with a linearisation over an interval from zero to some positive value.

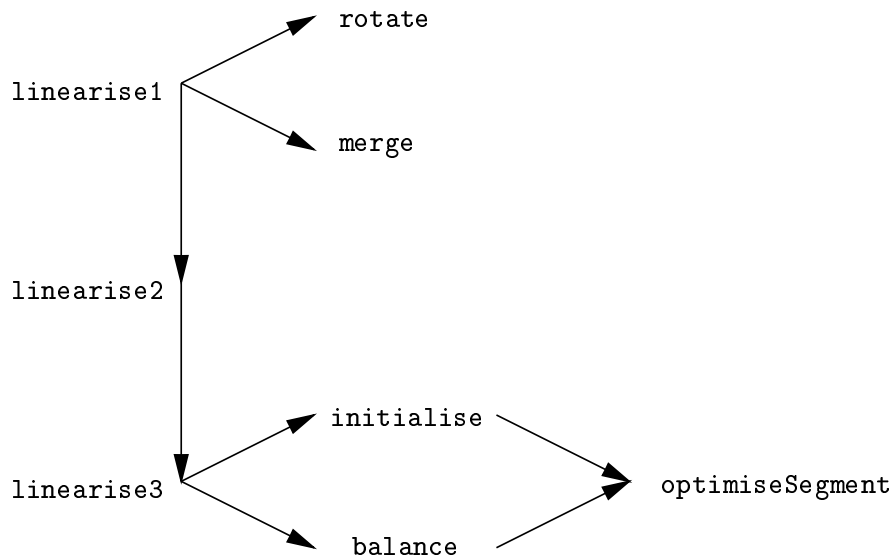


Figure 4.5: Structure of the piece-wise linearisation algorithm.

The structure of the algorithm

It is now possible to summarise the structure of the algorithm. Figure 4.5 shows the graph of important function calls used by the complete algorithm: `linearise1` uses `merge` and `rotate` to break the problem down to the $\min \geq 0$ case handled by `linearise2`; `linearise2` converts from maximum permitted linearisation error to the number of segments case handled by `linearise3`; `linearise3` initialises the segment spacing with `initialise` and then uses `balance` to balance up the linearisation errors in adjacent segments. Both `initialise` and `balance` call `optimiseSegment` to find the optimal straight line fit to the sigmoid within a single segment.

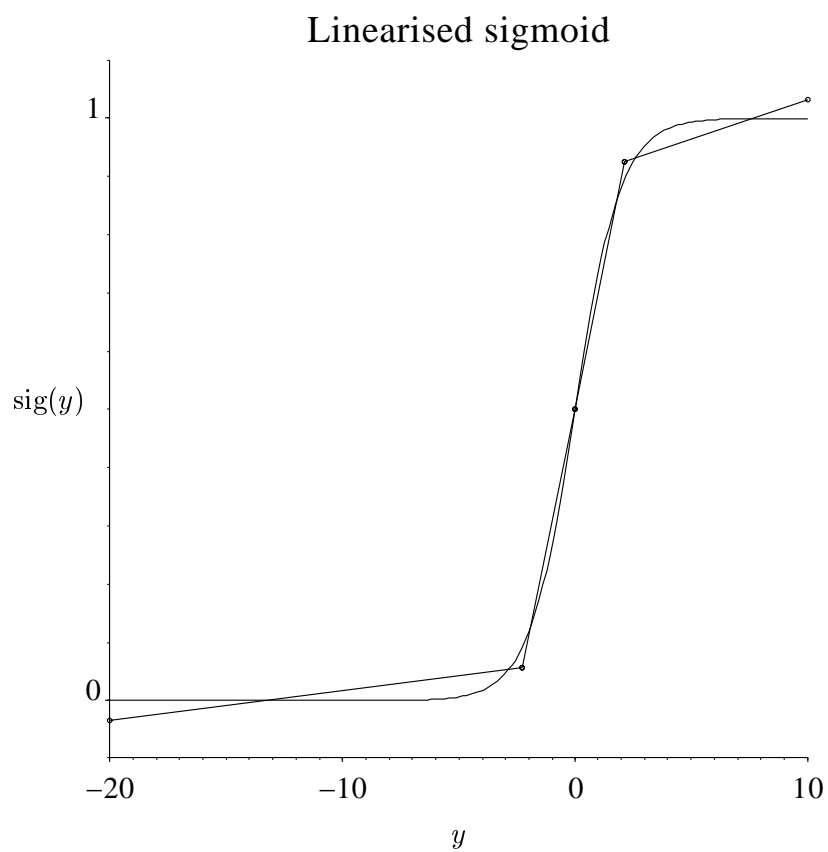
Assessment

Figure 4.6 shows the result of a linearisation over the interval $[-20, 10]$ with a specified maximum linearisation error of 0.05. Note that the smallest number of segments needed to meet this specification produce errors which are smaller than 0.05. Note also that the error for those segments to the left of $y = 0$ is different from those to the right. This is a result of merging two independent linearisations, one for each side of $y = 0$.

It is also interesting to see how accurately a sigmoid can be approximated by a piece-wise linearisation. Figure 4.7 shows how the linearisation error decreases as the number of segments used to linearise the interval $[0, 10]$ is increased. Note that there is a diminishing return from using more segments. This clearly has an affect on the choice of maximum linearisation error for each sigmoid, equation (4.21). In particular, specifying a very small value of overall linearisation error H may result in a very large number of cells. This is a computationally undesirable and unavoidable problem with the piece-wise linearisation method. Depending on the functional complexity of the MLP, some compromise between linearisation error and number of cells may be required.

The continuous property

Observe that the linearisation of figure 4.6 has the property that the lines from each segment connect to form a continuous approximation to the sigmoid. This property is true of every linearisation obtained with the algorithm described above. To understand why, note that the function `linearise3` (and hence `linearise2`) always returns a piece-wise linearisation which is continuous. This property is guaranteed because the line configurations within each segment are such that they connect when the error bounds are all made equal, and balancing the error bounds is exactly what the function achieves. It follows that every piece-wise linearisation obtained by `linearise1` over intervals which do not straddle $y = 0$ must be continuous. For intervals which *do* straddle $y = 0$, observe that the complete linearisation is made by merging two `linearise2`-linearisations, one for each side of $y = 0$. Thus the linearisation on each side is certainly continuous. However these linearisations also connect with each other because they are both constrained to pass through the point of rotational symmetry, $(0, 0.5)$. Thus the linearisation over the whole interval is continuous.



Segment, s	Y_s	Y_{s+1}	α_s	β_s	h_s
0	-20.0	-2.273	0.0691	0.0362	0.0362
1	-2.273	0.0	0.195	0.5	0.0362
2	0.0	2.135	0.200	0.5	0.0316
3	2.135	10.0	0.0134	0.897	0.0316

Figure 4.6: Results of the piece-wise linearisation of a sigmoid over the interval $[-20, 10]$ with a specified maximum linearisation error of 0.05.

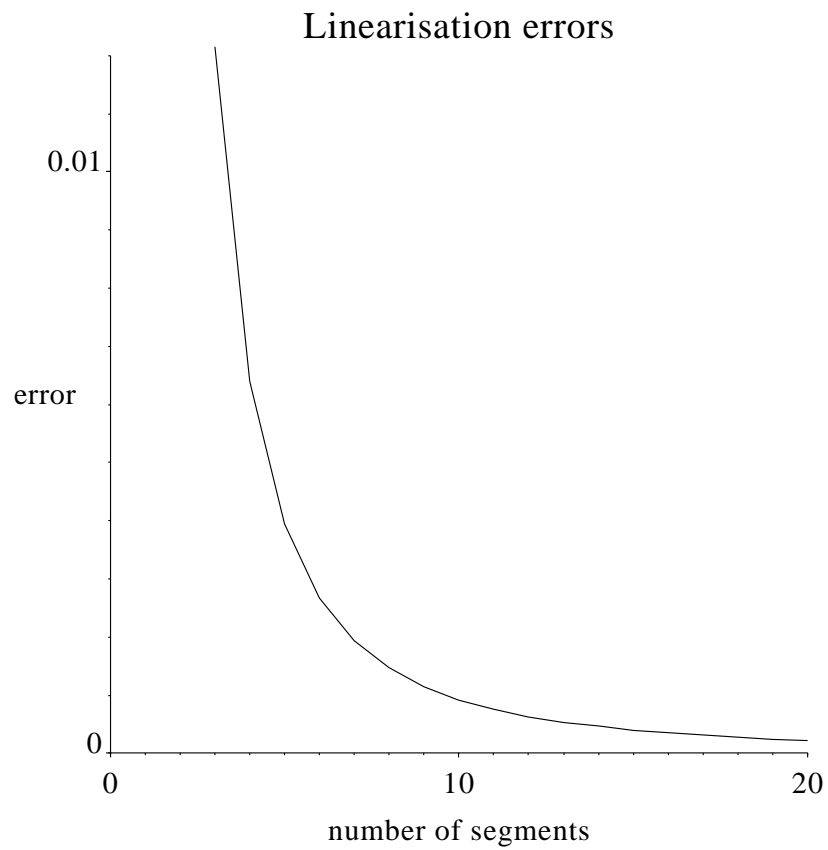


Figure 4.7: Linearisation error *versus* number of segments when approximating a sigmoid over the interval $[0, 10]$. Note how the reduction in linearisation error diminishes with increasing number of segments.

4.4 Properties and interpretation

One of the most important properties of the MLP-linearisation algorithm is continuity. Observe that the piece-wise linearisation of each sigmoid is continuous, and that these linearisations are linearly combined by the weights of the network. This implies that the piece-wise linearisation of the whole network is also continuous. In particular, the function

$$\widehat{\text{mlp}}(\mathbf{x}) = \mathbf{a}_s^T \mathbf{x} + b_s \quad \text{for all } \mathbf{x} \in \mathcal{X}_s \quad (4.40)$$

is linear within each cell s and continuous everywhere.

An immediate corollary of this result is that piece-wise linear and continuous *bounds* to the network are given by

$$\widehat{\text{mlp}}(\mathbf{x}) - h \leq \text{mlp}(\mathbf{x}) \leq \widehat{\text{mlp}}(\mathbf{x}) + h \quad (4.41)$$

where $h = \max_s(h_s)$. This property will prove invaluable in the next chapter.

Another important feature of the linearisation method is the emphasis on the hidden n -node representation, y -space. Cells are initially defined in y -space as axis-aligned hyper-rectangles, and each x -space cell is then defined as the set of points which map to a corresponding y -space cell.

$$\begin{aligned} \mathbf{y} \in \mathcal{Y}_s &\equiv \bigwedge_i y_i \in [Y_{i,s_i}, Y_{i,s_i+1}], \\ \mathbf{y} &= \mathbf{W}^{\text{IN}} \mathbf{x} + \mathbf{c}^{\text{IN}}, \\ \mathbf{x} \in \mathcal{X}_s &\equiv \mathbf{y} \in \mathcal{Y}_s. \end{aligned} \quad (4.42)$$

The characteristics of the mapping of cells from y -space to x -space is dictated by whether the input matrix, \mathbf{W}^{IN} , has more rows than columns. Let n_{in} be the number of input nodes (the number of columns of \mathbf{W}^{IN}) and n_{hid} be the number of hidden nodes (the number of rows of \mathbf{W}^{IN}). The two cases of $n_{\text{in}} \geq n_{\text{hid}}$ and $n_{\text{in}} < n_{\text{hid}}$ will now be considered.

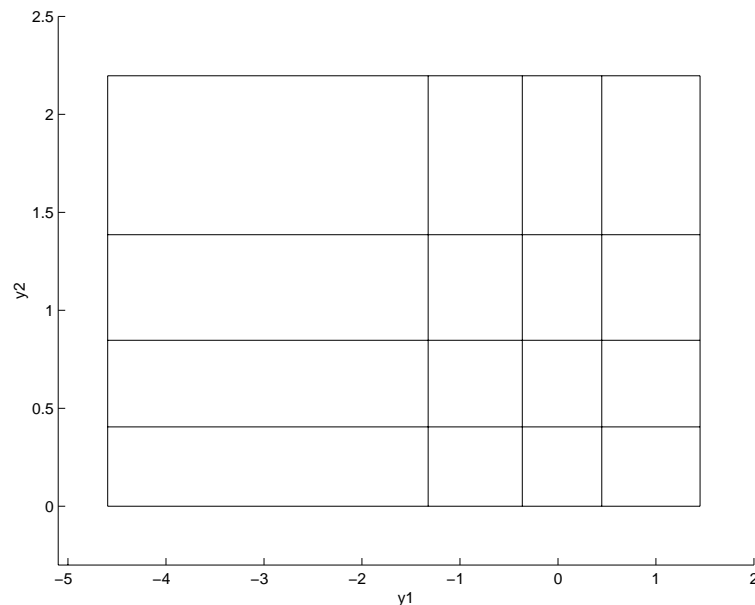


Figure 4.8: Illustration of the cells in a 2 dimensional y -space. The corresponding x -space cells for a network with 3 input nodes is shown in figure 4.9.

If $n_{\text{in}} \geq n_{\text{hid}}$ then the whole of x -space is projected onto y -space. Hence the cells in x -space will be hyper-parallelograms with no constraint in directions in the kernel of \mathbf{W}^{IN} . Figures 4.8 and 4.9 provide an illustration for the case when $n_{\text{in}} = 3$ and $n_{\text{hid}} = 2$.

However, if $n_{\text{in}} < n_{\text{hid}}$ then the whole of x -space is mapped into a linear subspace of y -space. This subspace will “slice” the axis-aligned hyper-rectangles defining the y -space cells, and so the cells in x -space will not be any convenient geometric shape. Indeed some x -space cells will be empty because the corresponding y -space cell has no members in the range of \mathbf{W}^{IN} . Figures 4.10, 4.11, 4.12 gives an illustration for the case when $n_{\text{in}} = 2$ and $n_{\text{hid}} = 3$.

An important conclusion to be drawn from this discussion on cell mapping is that the resulting cells in x -space are analytically much more complex than the original cells in y -space. Hence applying an operation on an x -space cell may well be considerably more complex than applying a similar operation on a y -space cell. Thus instead of mapping the cells from y -space to x -space and then applying a complex operation, mapping the *operation* from x -space to y -space may be much easier to implement. Exactly this idea will be used in chapter 5.

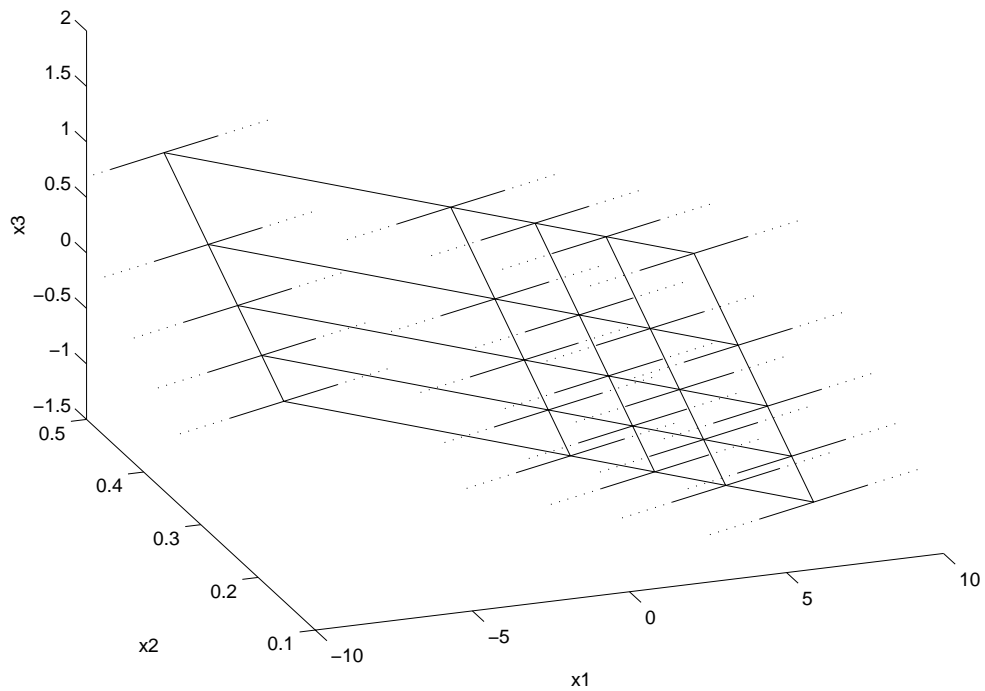


Figure 4.9: Illustration of the cells in a 3 dimensional x -space which result from the y -space cells shown in figure 4.8. Note that the cells are no longer axis-aligned hyper-rectangles, but have become arbitrarily aligned hyper-parallelograms. The dotted lines are a direction in the kernel of \mathbf{W}^{IN} ; the cells are unconstrained in this direction.

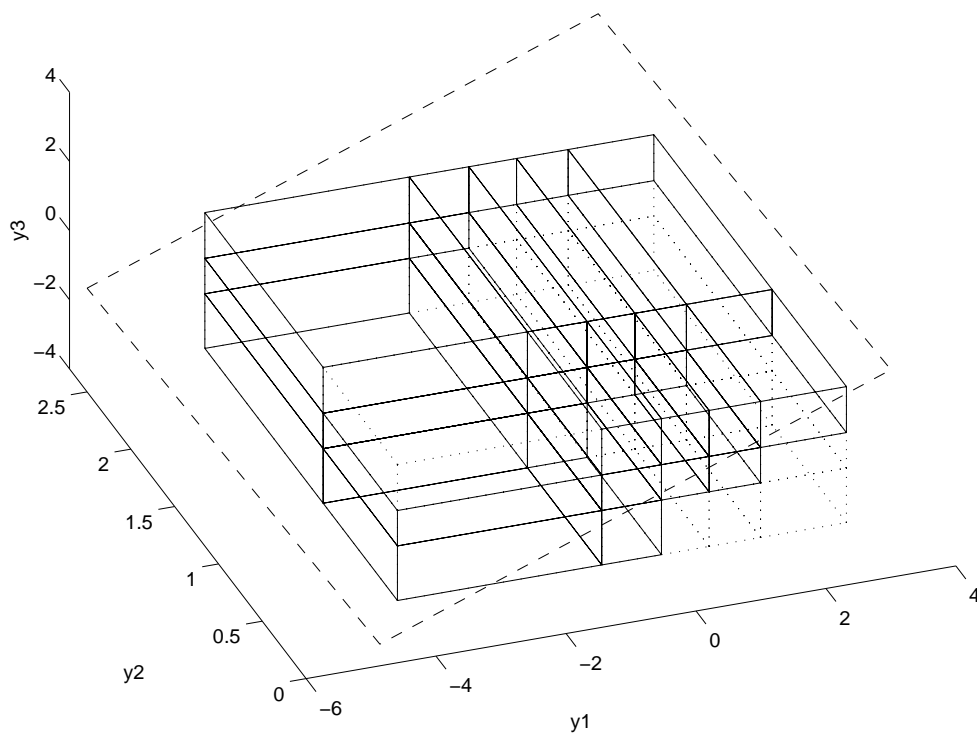


Figure 4.10: Illustration of the cells in a 3 dimensional y -space. The dashed rectangle is the image of a 2 dimensional x -space (the range of $\mathbf{W}^{\mathbf{N}}$). This image forms a subspace which “slices” some of the y -space cells. Sliced cells are shown solid; the dotted cells are not sliced and are therefore unreachable from x -space — see the companion figure 4.11.

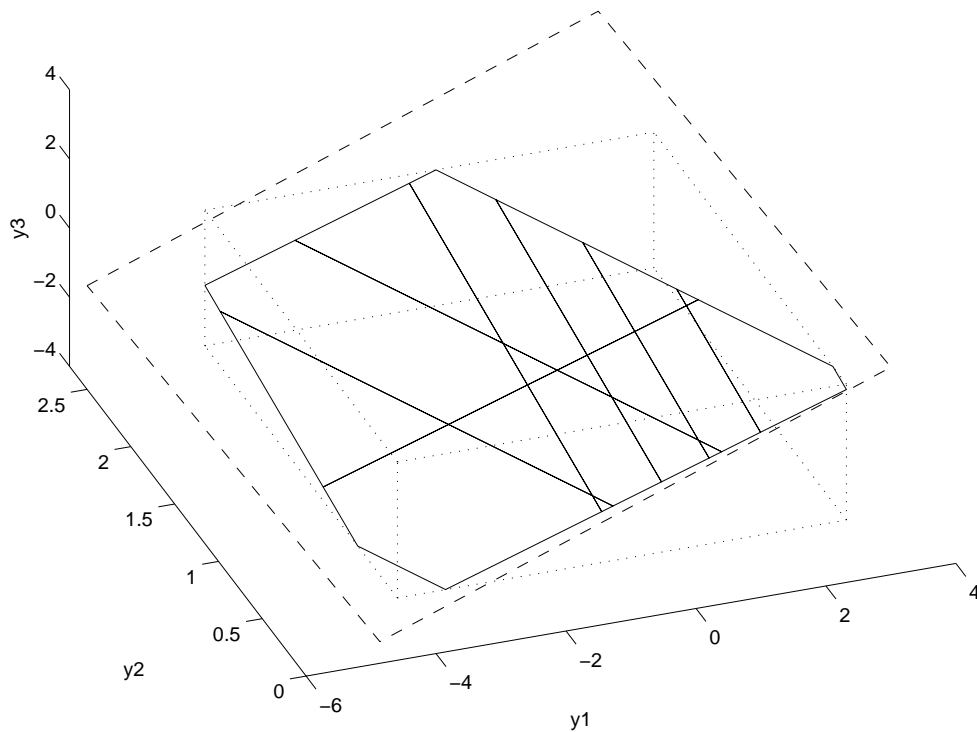


Figure 4.11: A companion to figure 4.10. The solid lines show the regions of y -space cells which are reachable from x -space.

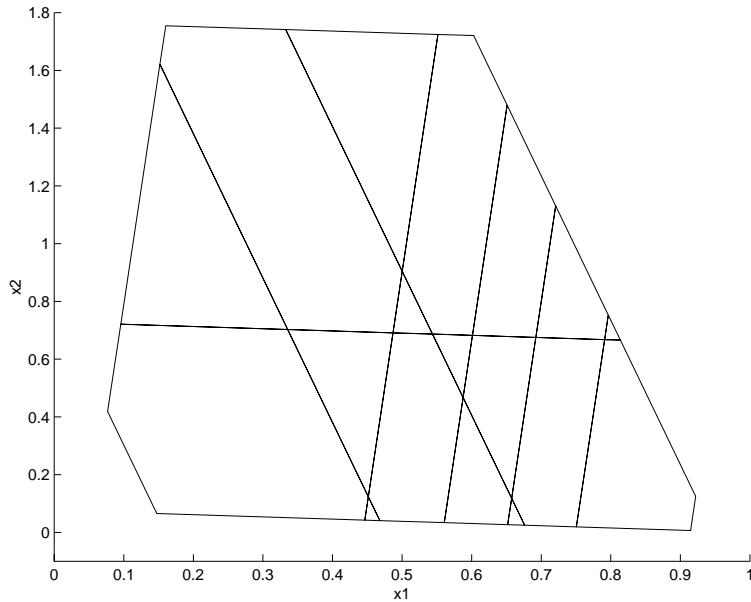


Figure 4.12: The x -space cells induced by the y -space cells of figure 4.10. Note the irregular shape of the cells when compared with the original axis-aligned hyper-rectangular cells in y -space.

4.5 Summary

This chapter has presented an algorithm for dividing the input space of a single linear output MLP into cells such that the network is accurately approximated by a linear function within each cell. These linear functions have known error bounds, and also join up to form a continuous piece-wise linearisation. Hence the result can be interpreted as a piece-wise linear and continuous *bound* on the MLP function.

The algorithm is based on the observation that the piece-wise linearisation of each sigmoid *implies* a piece-wise linearisation of the whole network. This property is used to generate an efficient MLP-linearisation algorithm which does not require operations in high dimensional space. Hence the complexity of this algorithm is approximately linear in the number of hidden nodes, and is virtually unaffected by the number of input dimensions. This is well suited to many classification problems which have a relatively large number of input dimensions but insufficient data to justify a very complex function and hence large number of hidden nodes.

Chapter 5

Extracting explanation

This chapter describes an algorithm to generate an explanation of the form described in chapter 3 from the feed-forward network classifier described in chapter 2. The algorithm critically relies on the bounded and continuous piece-wise linear approximation to an MLP described in chapter 4.

5.1 Introduction

It is useful at this stage to summarise the argument so far. An MLP is trained to estimate posterior class probabilities, and a Bayes classifier then uses these to make the optimal classification. However such classifications are unexplained, and this motivates the need to describe the classifications using a rule-base. Unfortunately a rule-base description has limitations which grow exponentially with the number of input dimensions. Compromises are therefore unavoidable. However it is essential that these compromises be controlled if there is to be a known relationship between the rule-base and the original Bayes classifier/feed-forward network system. The primary means of controlling the degree of compromise is through the *comparative safety* parameter, which is a measure of the greatest difference between the optimal and the explained classification. Thresholding safety gives rise to the concept of *comparative safety regions*. A class- k safety- r region

is defined to be the largest region of input space within which all patterns can be assigned to class k with a comparative safety greater than or equal to r .

Explaining the network/Bayes rule classifier means describing these safety regions using rules. Obviously a class k rule must not have a comparative safety less than that of its corresponding class k safety region. Thus every class k rule with comparative safety r must have an antecedent region which is a subset of the corresponding safety region, $\mathcal{R}_k(r)$,

$$\text{rule}_k(r) \subseteq \mathcal{R}_k(r). \quad (5.1)$$

In terms of the rule-base this becomes

$$\text{rulebase}_k(r) = \bigcup_i \text{rule}_{k,i}(r) \subseteq \mathcal{R}_k(r). \quad (5.2)$$

The primary tool used to produce a rule-base with the above subset property is the piece-wise linearisation method of chapter 4. It will be shown that using this method enables a “tight” subset of a safety region $\mathcal{R}_k(r)$ to be expressed using intersections and unions of cells and half-planes. Although this new representation, $\mathcal{R}_k^1(r)$, is much more explicit than $\mathcal{R}_k(r)$, it is still an awkward representation from which to extract rules. The solution is to use a further translation, expressing $\mathcal{R}_k^1(r)$ with an even simpler representation, $\mathcal{R}_k^2(r)$, consisting only of the union of intersections of half-planes. In order to guarantee (5.2), each new representation is made a subset of the previous representation. Hence the complete class k extraction process is characterised by the *subset condition*:

$$\text{rulebase}_k(r) \subseteq \mathcal{R}_k^2(r) \subseteq \mathcal{R}_k^1(r) \subseteq \mathcal{R}_k(r). \quad (5.3)$$

The process can be summarised as follows: a non-explicit representation of the class- k region of space, $\mathcal{R}_k(r)$, is made more and more explicit via a series of simplifying translations. This process terminates with a union of class k rules forming the class k part

of a rule-base. The ultimate aim is then to maximise the effectiveness and parsimony of a rule-base containing rules for all classes.

The three translations represented by each subset symbol in equation (5.3) are described in each of the following three sections. Note that a simplification will be made in the first section which constrains the solution to two class problems; the sections that follow will continue to work under this restriction. However there are no fundamental reasons why multi-class explanations cannot be obtained using these methods.

5.2 Translating a safety region into \mathcal{R}_k^1

This section shows how the piece-wise linearisation method of chapter 4 can be used to generate the first intermediate representation, \mathcal{R}_k^1 .

5.2.1 Converting to a single linear output MLP

Recall the equations (2.11) on page 35 for a multi-layer perceptron used to estimate posterior probabilities of mutually exclusive classes:

$$\begin{aligned} y_j &= \sum_k W_{jk}^{\text{IN}} x_k + c_j^{\text{IN}} \\ z_i &= \sum_j W_{ij}^{\text{OUT}} \text{sig}(y_j) + c_i^{\text{OUT}} \\ \hat{P}(k | \mathbf{x}) &= \frac{e^{z_k}}{\sum_i e^{z_i}}. \end{aligned} \tag{5.4}$$

Also recall from chapter 3 that a safety region, equation (3.31) on page 89, is the largest region of space which an explaining classifier can assign to a particular class whilst guaranteeing a specified comparative safety, r :

$$\mathcal{R}_k(r) = \tilde{\mathcal{N}} \cup \mathcal{N} \cap \left\{ \mathbf{x} \left| \frac{\hat{P}(k | \mathbf{x})}{\hat{P}(\hat{d}_B(\mathbf{x}) | \mathbf{x})} \geq r \right. \right\}. \tag{5.5}$$

This expression can be simplified by using the property that the Bayes classifier $\hat{d}_B(\mathbf{x})$ assigns each pattern to the class which is estimated to be the most probable:

$$\begin{aligned} \mathcal{R}_k(r) &= \bar{\mathcal{N}} \cup \mathcal{N} \cap \left\{ \mathbf{x} \mid \frac{\hat{P}(k|\mathbf{x})}{\hat{P}(\hat{d}_B(\mathbf{x})|\mathbf{x})} \geq r \right\} \\ &= \bar{\mathcal{N}} \cup \mathcal{N} \cap \left\{ \mathbf{x} \mid \bigwedge_{i \neq k} \frac{\hat{P}(k|\mathbf{x})}{\hat{P}(i|\mathbf{x})} \geq r \right\} \end{aligned}$$

Further simplification is achieved using equation (5.4), and the property that a set of conjunction of conditionals is equal to the intersection of sets of conditionals:

$$\begin{aligned} &= \bar{\mathcal{N}} \cup \mathcal{N} \cap \bigcap_{i \neq k} \left\{ \mathbf{x} \mid \frac{e^{z_k}}{e^{z_i}} \geq r \right\} \\ &= \bar{\mathcal{N}} \cup \mathcal{N} \cap \bigcap_{i \neq k} \{ \mathbf{x} \mid z_k - z_i \geq \ln(r) \} \\ &= \bar{\mathcal{N}} \cup \mathcal{N} \cap \bigcap_{i \neq k} \{ \mathbf{x} \mid f_{k,i}(\mathbf{x}) \geq 0 \} \end{aligned} \quad (5.6)$$

where

$$f_{k,i}(\mathbf{x}) = z_k - z_i - \ln(r). \quad (5.7)$$

Note that $f_{k,i}(\mathbf{x})$ is effectively a single linear output MLP. Thus each safety region may be expressed in terms of the intersections of regions defined by thresholds on single linear output MLP's. The advantage of this property is that the method of chapter 4 may be used on each $f_{k,i}(\mathbf{x})$ to obtain an explicit representation of the safety region consisting only of cells and half-planes.

5.2.2 Simplifying to problems with only two classes

The rest of this thesis will be limited to problems which have only two classes, *i.e.* $k \in \{1, 2\}$. The following few lines demonstrate how this simplification results in the removal of the intersection quantifier in equation (5.6).

An MLP estimating two posterior class probabilities only needs a single output because the two probabilities must sum to unity.

$$\begin{aligned}
 y_j &= \sum_k W_{jk}^{\text{IN}} x_k + c_k^{\text{IN}} \\
 z &= \sum_j w_j^{\text{OUT}} \text{sig}(y_j) + c^{\text{OUT}} \\
 \hat{P}(1 | \mathbf{x}) &= \text{sig}(z) \\
 \hat{P}(2 | \mathbf{x}) &= 1 - \text{sig}(z)
 \end{aligned} \tag{5.8}$$

Following the same steps as in (5.6) it is straightforward to show that

$$\mathcal{R}_k = \bar{\mathcal{N}} \cup \mathcal{N} \cap \{\mathbf{x} | f_k(\mathbf{x}) \geq 0\} \tag{5.9}$$

where

$$\begin{aligned}
 f_1(\mathbf{x}) &= +z - \ln(r) \\
 f_2(\mathbf{x}) &= -z - \ln(r).
 \end{aligned} \tag{5.10}$$

5.2.3 Derivation of \mathcal{R}_k^1

Observe that $f_k(\mathbf{x})$ is a linear combination of the hidden node sigmoids of the MLP. In other words, $f_k(\mathbf{x})$ is a single linear output MLP which can be piece-wise linearised using the method of chapter 4. Thus coefficients $\mathbf{a}_{k,s}$, $b_{k,s}$, $h_{k,s}$ and cells $\mathcal{X}_{k,s}$ can be obtained which satisfy

$$f_k(\mathbf{x}) \in \mathbf{a}_{k,s}^T \mathbf{x} + b_{k,s} + [-h_{k,s}, h_{k,s}] \quad \text{for all } \mathbf{x} \in \mathcal{X}_{k,s}, \tag{5.11}$$

and where the cells $\mathcal{X}_{k,s}$ are mutually exclusive regions which cover a superset of normality,

$$\bigcup_s \mathcal{X}_{k,s} \supseteq \mathcal{N}. \quad (5.12)$$

It follows that when \mathbf{x} is in $\mathcal{X}_{k,s}$, $f_k(\mathbf{x})$ is guaranteed to be greater than 0 if $\mathbf{a}_{k,s}^T \mathbf{x} + b_{k,s} - h_k \geq 0$, where $h_k = \max_s h_{k,s}$. The intuitive explanation is clear: within each cell, use the worst case linearisation error to decide the half-plane which is *guaranteed* to be on the correct (comparatively safe) side of the boundary. Expressing this more consisely,

$$\mathbf{a}_{k,s}^T \mathbf{x} + b_{k,s} - h_k \geq 0 \quad \Rightarrow \quad f_k(\mathbf{x}) \geq 0 \quad \text{for all } \mathbf{x} \in \mathcal{X}_{k,s} \quad (5.13)$$

where $h_k = \max_s h_{k,s}$. Therefore

$$\mathcal{X}_{k,s} \cap \mathcal{H}_{k,s} \subseteq \mathcal{X}_{k,s} \cap \mathcal{R}_k, \quad (5.14)$$

where the half-plane $\mathcal{H}_{k,s}$ is defined

$$\mathcal{H}_{k,s} = \{ \mathbf{x} \mid \mathbf{a}_{k,s}^T \mathbf{x} + b_{k,s} - h_k \geq 0 \}. \quad (5.15)$$

Note that equation (5.14) is a subset relation and not an equality because $\mathcal{H}_{k,s}$ is formed from a linear *bound* on the MLP function within cell s .

Taking the union of equation (5.14) over all cells shows that

$$\bigcup_s \mathcal{X}_{k,s} \cap \mathcal{H}_{k,s} \subseteq \bigcup_s \mathcal{X}_{k,s} \cap \mathcal{R}_k, \quad (5.16)$$

and so if \mathcal{R}_k^1 is defined

$$\mathcal{R}_k^1 = \bar{\mathcal{N}} \cup \mathcal{N} \cap \left(\bigcup_s \mathcal{X}_{k,s} \cap \mathcal{H}_{k,s} \right), \quad (5.17)$$

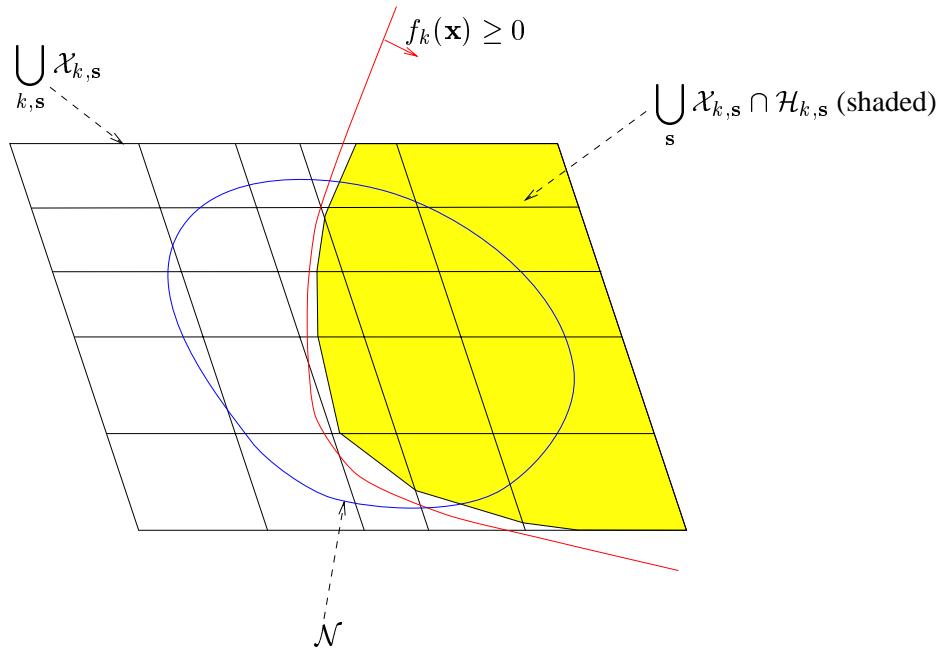


Figure 5.1: Illustration of the key terms in the definitions of \mathcal{R}_k and \mathcal{R}_k^1 . Note that here $n_{\text{hid}} \leq n_{\text{in}}$, thus making all the cells similar in shape.

then equations (5.16) and (5.12) imply the subset condition:

$$\begin{aligned}
 \mathcal{R}_k^1 &= \bar{\mathcal{N}} \cup \mathcal{N} \cap \left(\bigcup_s \mathcal{X}_{k,s} \cap \mathcal{H}_{k,s} \right) \\
 &\subseteq \bar{\mathcal{N}} \cup \mathcal{N} \cap \bigcup_s \mathcal{X}_{k,s} \cap \mathcal{R}_k \\
 &= \mathcal{R}_k.
 \end{aligned} \tag{5.18}$$

A two dimensional illustration of this result is shown in figure 5.1.

5.2.4 Discussion

The key property of \mathcal{R}_k^1 defined by equation (5.17) is that it expresses a subset of \mathcal{R}_k in terms of simple geometric regions $\mathcal{X}_{k,s}$ and $\mathcal{H}_{k,s}$. In particular, both $\mathcal{X}_{k,s}$ and $\mathcal{H}_{k,s}$ are simplexes¹ with particular properties: $\mathcal{H}_{k,s}$ is a degenerate simplex defined by a single

¹Recall that a *simplex* is a region defined by the intersection of half-planes.

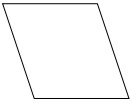
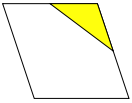
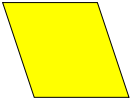
Empty cell		$\mathcal{X}_{k,s} \cap \mathcal{H}_{k,s} = \emptyset$
Boundary cell		$\mathcal{X}_{k,s} \cap \mathcal{H}_{k,s} \neq \emptyset$ and $\mathcal{X}_{k,s} \cap \bar{\mathcal{H}}_{k,s} \neq \emptyset$
Full cell		$\mathcal{X}_{k,s} \cap \bar{\mathcal{H}}_{k,s} = \emptyset$

Figure 5.2: The three combinations of a cell and a half-plane.

half-plane; $\mathcal{X}_{k,s}$ is a simplex which maps to an axis-aligned hyper-rectangle $\mathcal{Y}_{k,s}$ in y -space. The value of these properties will become clear in the next section.

Note that each cell can be categorised into one of three types: *empty cells* are cells which do not intersect with their half-planes; *full cells* are cells which are a subset of their half-planes; and *boundary cells* are cells which are “sliced” by their half-planes. These three types are illustrated in figure 5.2.

The connected property

Every boundary cell contains a hyper-plane boundary, and these boundaries have the important property that they are *connected*. This is a direct result of the linearisation method producing a continuous function; any threshold defined over a piece-wise linear and continuous function must produce a connected set of constraints. This connected property can be defined

$$\text{if } \mathbf{x}_1 \in \bigcup_s \mathcal{X}_{k,s} \cap \mathcal{H}_{k,s} \quad \text{and} \quad \mathbf{x}_2 \notin \bigcup_s \mathcal{X}_{k,s} \cap \mathcal{H}_{k,s}$$

$$\text{then there exists an } \mathcal{H}_{k,s} \text{ such that } \mathbf{x}_1 \in \mathcal{H}_{k,s} \quad \text{and} \quad \mathbf{x}_2 \notin \mathcal{H}_{k,s}. \quad (5.19)$$

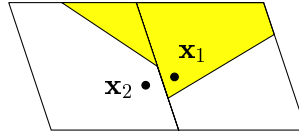


Figure 5.3: An example of two cells/half-planes *not* satisfying the connected property.

Figure 5.3 shows an example which does *not* satisfy the connected property, and therefore will not theoretically be produced by the above methods. This point will be discussed again later.

5.3 Translating \mathcal{R}_k^1 into \mathcal{R}_k^2

This section describes how \mathcal{R}_k^1 can be simplified into a representation denoted \mathcal{R}_k^2 . The difference between these two representations is that \mathcal{R}_k^1 is expressed in terms of a union of cells intersecting with half-planes, whilst \mathcal{R}_k^2 is expressed as a union of simplexes where each simplex is formed only from the half-plane boundaries. Thus one of the effects of the translation is to eliminate the cells from the formula. This has the advantage of describing the comparative safety regions using a union of simplexes, where each simplex is no longer restricted to being no larger than the size of an individual cell. Later in section 5.4 it will be seen that the method of obtaining rules guarantees that each rule is contained within an individual simplex. Hence it is clearly advantageous to make these simplexes as large as possible.

The first section below introduces the notation and provides a specification for an ideal solution. The next section describes an algorithm which although not ideal, still produces a satisfactory solution. This algorithm requires certain operations on cells and half-planes, and the detailed implementation of these operations are described in the last section.

5.3.1 Introduction

Define the regions of space \mathcal{P}_k^1 and \mathcal{P}_k^2 ,

$$\mathcal{P}_k^1 = \bigcup_{\mathbf{s}} \mathcal{X}_{k,\mathbf{s}} \cap \mathcal{H}_{k,\mathbf{s}}, \quad (5.20)$$

$$\mathcal{P}_k^2 = \bigcup_i \mathcal{S}_{k,i} \cap \bigcup_{\mathbf{s}} \mathcal{X}_{k,\mathbf{s}}, \quad (5.21)$$

where the simplex $\mathcal{S}_{k,i}$ is defined using a set of half-plane coordinates, $A_{k,i}$:

$$\mathcal{S}_{k,i} = \bigcap_{\mathbf{s} \in A_{k,i}} \mathcal{H}_{k,\mathbf{s}}. \quad (5.22)$$

Thus \mathcal{R}_k^1 and \mathcal{P}_k^1 are related by $\mathcal{R}_k^1 = \bar{\mathcal{N}} \cup \mathcal{N} \cap \mathcal{P}_k^1$ and we will define \mathcal{R}_k^2 in terms of \mathcal{P}_k^2 ,

$$\mathcal{R}_k^2 = \bar{\mathcal{N}} \cup \mathcal{N} \cap \mathcal{P}_k^2. \quad (5.23)$$

Note that the left hand side of the \mathcal{P}_k^2 definition can be interpreted as the union of simplexes, where the i^{th} simplex is formed from the intersection of half-planes $\mathcal{H}_{k,\mathbf{s}}$ with indexes \mathbf{s} taken from $A_{k,i}$. The right hand side simply restricts the whole region to the space which has been linearised.

Define validity and completeness,

$$\text{completeness} \equiv \mathcal{P}_k^2 \supseteq \mathcal{P}_k^1 \quad (5.24)$$

$$\text{validity} \equiv \mathcal{P}_k^2 \subseteq \mathcal{P}_k^1. \quad (5.25)$$

Thus if both validity and completeness are satisfied then $\mathcal{P}_k^2 = \mathcal{P}_k^1$. This would constitute an ideal solution because it implies $\mathcal{R}_k^2 = \mathcal{R}_k^1$.

The special form of \mathcal{P}_k^1 and \mathcal{P}_k^2 give rise to the following equivalent expressions for

completeness and validity:

$$\text{completeness} \equiv \bigwedge_{\mathbf{s}} (\mathcal{X}_{k,\mathbf{s}} \cap \mathcal{H}_{k,\mathbf{s}} \subseteq \mathcal{P}_k^2) \quad (5.26)$$

$$\text{validity} \equiv \bigwedge_{\mathbf{s}} (\mathcal{X}_{k,\mathbf{s}} \cap \bar{\mathcal{H}}_{k,\mathbf{s}} \cap \mathcal{P}_k^2 = \emptyset). \quad (5.27)$$

Thus completeness ensures that all cell / half-plane intersections are in \mathcal{P}_k^2 , and validity ensures that there does not exist a cell / half-plane intersection in \mathcal{P}_k^2 which should not be in \mathcal{P}_k^2 .

5.3.2 An algorithm to produce a valid \mathcal{R}_k^2

Unfortunately no algorithm could be found which guarantees both validity and completeness. However the algorithm described below is guaranteed to produce a valid \mathcal{P}_k^2 . This solution is sub-optimal but acceptable because validity implies the important subset condition, $\mathcal{R}_k^2 \subseteq \mathcal{R}_k^1$.

First note that validity can be usefully re-expressed:

$$\begin{aligned} \text{validity} &\equiv \bigwedge_{\mathbf{s}} \left(\mathcal{X}_{k,\mathbf{s}} \cap \bar{\mathcal{H}}_{k,\mathbf{s}} \cap \left(\bigcup_i \mathcal{S}_{k,i} \cap \bigcup_{\mathbf{s}'} \mathcal{X}_{k,\mathbf{s}'} \right) = \emptyset \right) \\ &\equiv \bigwedge_{\mathbf{s}} \left(\mathcal{X}_{k,\mathbf{s}} \cap \bar{\mathcal{H}}_{k,\mathbf{s}} \cap \bigcup_i \mathcal{S}_{k,i} = \emptyset \right) \\ &\equiv \bigwedge_i \left[\bigwedge_{\mathbf{s}} (\mathcal{X}_{k,\mathbf{s}} \cap \bar{\mathcal{H}}_{k,\mathbf{s}} \cap \mathcal{S}_{k,i} = \emptyset) \right]. \end{aligned} \quad (5.28)$$

The structure of this equation suggests the method of figure 5.4. For each simplex, each half-plane/cell term is checked in turn, and the half-plane added if necessary. As an example, consider figure 5.5 which shows four adjacent boundary cells with shading used to represent the four $\mathcal{X}_{k,\mathbf{s}} \cap \mathcal{H}_{k,\mathbf{s}}$ intersections. We would like to describe this shaded region using the \mathcal{R}_k^2 representation. It is easily verified that if the coordinates \mathbf{s} are by chance chosen in the order $[1, 4, 2, 3]$ then the half-plane indexes $\{1, 4, 2\}$ will be added to $A_{k,1}$.

1. For $i=1$ to n
2. Let C be the set of coordinates of all cells
3. Initialise $A_{k,i} = \{\}$
4. Until C is empty
5. Pick at random an s from C
6. If $\mathcal{X}_{k,s} \cap \bar{\mathcal{H}}_{k,s} \cap \mathcal{S}_{k,i} \neq \emptyset$ then add s to $A_{k,i}$
7. Remove s from C

Figure 5.4: Simple algorithm for producing the simplexes $\mathcal{S}_{k,i} = \bigcap_{s \in A_{k,i}} \mathcal{H}_{k,s}$ which make up a valid \mathcal{R}_k^2 .

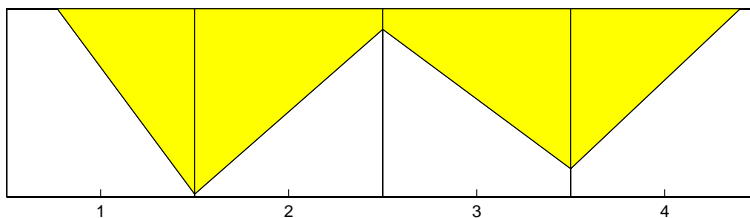


Figure 5.5: Simple 4 cell configuration of half-planes. The shaded region denotes $\bigcup_s \mathcal{X}_{k,s} \cap \mathcal{H}_{k,s}$.

In shorthand,

$$[1, 4, 2, 3] \longrightarrow A_{k,1} = \{1, 4, 2\} \quad (5.29)$$

where the left hand side indicates the order in which the coordinates are chosen, and the right hand side indicates the set $A_{k,i}$ after the simplex has been formed. Similarly,

$$[1, 4, 3, 2] \longrightarrow A_{k,2} = \{1, 4, 3\} \quad (5.30)$$

These examples can be used to illustrate two points. First, the outputs can be simplified because they represent simplexes with redundant half-planes. In particular, the simplex described by $A_{k,i} = \{1, 4, 2\}$ is the same as the simplex described by $A_{k,i} = \{1, 2\}$, and the simplex described by $A_{k,i} = \{1, 4, 3\}$ is the same as the simplex described by $A_{k,i} = \{3, 4\}$. Second, different simplexes will be formed depending on the order in

which cells are considered. By considering cells in a random order (line 5), each simplex can be different. The union of all such generated simplexes may then satisfy completeness.

Improving the algorithm

The above algorithm can be improved upon in a number of ways. The first improvement relies upon the connected property of the half-planes within the boundary cells. This property implies that if a simplex does not intersect any “forbidden” region $\mathcal{X}_{k,s} \cap \bar{\mathcal{H}}_{k,s}$ for all *boundary cells* s , then the simplex does not intersect any forbidden region $\mathcal{X}_{k,s} \cap \bar{\mathcal{H}}_{k,s}$ for *all cells* s . Thus only boundary cells need be considered.

The second improvement involves reducing the number of half-planes which are added to the simplex but later found to be redundant. Intuition suggest that adjacent boundary cells should be considered first. This approach builds a maximally local simplex before checking the remaining boundary cells. Using this idea on the example shown in figure 5.5 prevents redundant half-planes from being formed.

The third improvement attempts to encourage completeness by beginning each new simplex with a half-plane which is not in any of the simplexes generated so far.

These three modifications are embodied in the final algorithm, summarised in figure 5.6. The first loop (lines 5 to 9) builds a locally valid simplex; the set C contains all the coordinates of all connected boundary cells which need to be checked. At the end of this loop, B contains the coordinates of all the boundary cells which have not yet been checked. Hence the second loop (lines 10 to 13) checks these remaining boundary cells.

5.3.3 Cell/half-plane operations

The following operations are required by the algorithm of figure 5.6.

1. Identify all boundary cells (line 2).
2. Identify all boundary cells connected to a given boundary cell (line 9).

1. For $i=1$ to n
2. Let B be the set of coordinates of all boundary cells
3. Let $A_{k,i} = \{s\}$ where $s \notin \bigcup_j^{i-1} A_{k,j}$; remove s from B
4. Let $C =$ set of coordinates of all boundary cells connected to s
5. Until C is empty
6. Pick at random an s from C
7. If $\mathcal{X}_{k,s} \cap \bar{\mathcal{H}}_{k,s} \cap \mathcal{S}_{k,i} \neq \emptyset$ then add s to $A_{k,i}$
8. Remove s from both B and C
9. Add to C the coordinates of all boundary cells joined to s and in B .
10. Until B is empty
11. Pick at random an s from B
12. If $\mathcal{X}_{k,s} \cap \bar{\mathcal{H}}_{k,s} \cap \mathcal{S}_{k,i} \neq \emptyset$ then add s to $A_{k,i}$
13. Remove s from B
14. Remove all redundant half-planes from the simplex $A_{k,i}$

Figure 5.6: Algorithm to produce simplexes $\mathcal{S}_{k,i} = \bigcap_{s \in A_{k,i}} \mathcal{H}_{k,s}$ which make up a valid \mathcal{R}_k^2 .

3. Test for $\mathcal{X}_{k,s} \cap \bar{\mathcal{H}}_{k,s} \cap \mathcal{S}_{k,i} \neq \emptyset$ (lines 7 and 12).
4. Eliminate all redundant half-planes from a simplex (line 14).

The field of *linear programming*, [Chv80, IC94, Fou98], provides techniques which can be used to perform all these operations. Linear programming is a means of optimising a linear equation constrained by a set of linear equalities and inequalities. All of the above operations are of this form. However they also have a particular structure which general purpose linear programming techniques do not (by definition) take advantage of. This structure can be exploited to obtain more efficient implementations of operations 1 and 2, and sometimes operation 3. Operation 4 does not have sufficient structure to make dedicated solutions worthwhile, and so the general linear programming method called *the simplex method* is used for this operation.

It was discussed in section 4.4 on page 113 that cells in x -space are analytically more complex than cells in y -space. Thus it is not surprising that the key to solving operations 1, 2 and 3 efficiently is *not* to consider them to be problems in x -space, but rather to consider the equivalent problems in y -space. The first section below discusses this mapping between x -space and y -space. The succeeding sections provide implementations of each operation in y -space.

All of the methods below are used on cells / half-planes originating from the safety region for a single class k . Hence the k subscript will be dropped to increase clarity.

Mapping operations from x -space into y -space

Recall (figure 4.1 on page 97) that the equation linking x -space to y -space is that of the input layer of the MLP,

$$\mathbf{y} = \mathbf{W}^{\text{IN}} \mathbf{x} + \mathbf{c}^{\text{IN}}, \quad (5.31)$$

and that cells are originally defined as axis aligned hyper-rectangles in y -space,

$$\mathbf{y} \in \mathcal{Y}_s \equiv \bigwedge_i y_i \in [Y_{i,s_i}, Y_{i,s_i+1}] \quad (5.32)$$

$$\mathbf{x} \in \mathcal{X}_s \equiv \mathbf{y} = \mathbf{W}^{\text{IN}} \mathbf{x} + \mathbf{c}^{\text{IN}} \in \mathcal{Y}_s. \quad (5.33)$$

Recall also (equation 5.15) that the half-planes in x -space are defined,

$$\mathcal{H}_s = \{ \mathbf{x} \mid \mathbf{a}_s^T \mathbf{x} + b_s - h \geq 0 \} \quad (5.34)$$

where (figure 4.1 on page 97) $\mathbf{a}_s^T = \boldsymbol{\alpha}_s^T \mathbf{W}^{\text{IN}}$ and $b_s = \boldsymbol{\alpha}_s^T \mathbf{c}^{\text{IN}} + \beta_s$. Hence \mathcal{H}_s has a corresponding half-plane in y -space,

$$\mathcal{I}_s = \{ \mathbf{y} \mid \boldsymbol{\alpha}_s^T \mathbf{y} + \beta_s - h \geq 0 \}. \quad (5.35)$$

Taking equations (5.32), (5.33), (5.34), and (5.35) together, it is obvious that there is a correspondence between cells and half-planes in x -space ($\mathcal{X}_s, \mathcal{H}_s$), and cells and half-planes in y -space ($\mathcal{Y}_s, \mathcal{I}_s$). Given that x -space cells can have awkward shapes (section 4.4 on page 113), it is clearly much simpler to perform all the operations in y -space. In other words, instead of mapping each cell and half-plane from y -space to x -space and then forming \mathcal{R}^2 , form an analogous \mathcal{R}^2 in y -space and then map this region into x -space. This is remarkably simple to achieve: take the algorithm of figure 5.6 and re-write every \mathcal{X}_s as

\mathcal{Y}_s and every \mathcal{H}_s as \mathcal{I}_s . The output from such an algorithm will be a union of simplexes in y -space. To map these simplexes into x -space, apply equation (5.31) to change every half-plane of each simplex from a constraint in y -space to a constraint in x -space.

If there are more hidden nodes than input nodes then one extra task is required: all redundant constraints must be identified and removed. Redundant constraints may have been introduced because x -space is a “slice” through the constraints in y -space (figure 4.10 on page 116). Note that this redundancy removal was required by the original x -space algorithm where it was solved without relying on any special properties using linear programming. Thus the same method may be used here.

Identifying all boundary cells

A boundary cell (see figure 5.2 and section 5.2.4) is a cell with an associated half-plane which “slices” the cell region. In other words (in y -space),

$$\begin{aligned} & \mathcal{Y}_s \cap \mathcal{I}_s \neq \emptyset \\ \text{and} \quad & \mathcal{Y}_s \cap \bar{\mathcal{I}}_s \neq \emptyset. \end{aligned} \quad (5.36)$$

This is equivalent to

$$\begin{aligned} & \max_{\mathbf{y} \in \mathcal{Y}_s} [\boldsymbol{\alpha}_s^T \mathbf{y} + \beta_s - h] > 0 \\ \text{and} \quad & \min_{\mathbf{y} \in \mathcal{Y}_s} [\boldsymbol{\alpha}_s^T \mathbf{y} + \beta_s - h] < 0. \end{aligned} \quad (5.37)$$

Computing the above constrained maximum and minimum is trivial because in y -space the constraints are axis-aligned, and this implies that each variable may be chosen independently. Thus using the fact that $Y_{i,s_i} < Y_{i,s_i+1}$, the solution may be written

$$\max_{\mathbf{y} \in \mathcal{Y}_s} [\boldsymbol{\alpha}_s^T \mathbf{y} + \beta_s - h] = \sum_i M((\boldsymbol{\alpha}_s)_i, Y_{i,s_i+1}, Y_{i,s_i}) + \beta_s - h \quad (5.38)$$

$$\min_{\mathbf{y} \in \mathcal{Y}_s} [\boldsymbol{\alpha}_s^T \mathbf{y} + \beta_s - h] = \sum_i M((\boldsymbol{\alpha}_s)_i, Y_{i,s_i}, Y_{i,s_i+1}) + \beta_s - h \quad (5.39)$$

where

$$M(\gamma, a, b) = \begin{cases} \gamma a & \text{if } \gamma \geq 0 \\ \gamma b & \text{if } \gamma < 0 \end{cases}. \quad (5.40)$$

This method provide an extremely quick way of determining which cells are boundary cells.

Identifying all connected boundary cells

This section describes a method of determining all the boundary cells which are connected to a given boundary cell, \mathcal{Y}_s . Observe that if the half-plane \mathcal{I}_s intersects a particular side of the cell, then the cell on the other side must be a boundary cell connected to \mathcal{Y}_s . Observe also that checking if a half-plane intersects a side is the same problem as checking if a cell is a boundary cell, but constrained to the sub-space defined by the side. These two observations lead to the following method which can be used to test each adjacent cell in turn.

Let the cell to be tested have coordinates \mathbf{t} satisfying $\mathbf{t} = \mathbf{s} + \mathbf{e}_j$ where $\mathbf{e}_j = (0, \dots, 0, 1, 0, \dots, 0)^T$ and the 1 is in the j^{th} position. Then cell \mathbf{t} is connected to cell \mathbf{s} if and only if

$$\begin{aligned} & \mathcal{Y}_s \cap \mathcal{I}_s \cap \{\mathbf{y} \mid y_j = Y_{j,s_{j+1}}\} \neq \emptyset \\ \text{and} \quad & \mathcal{Y}_s \cap \bar{\mathcal{I}}_s \cap \{\mathbf{y} \mid y_j = Y_{j,s_{j+1}}\} \neq \emptyset. \end{aligned} \quad (5.41)$$

It is easy to show that this condition is equivalent to the following two computable inequalities:

$$\begin{aligned} & \sum_{i \neq j} M((\boldsymbol{\alpha}_s)_i, Y_{i,s_{i+1}}, Y_{i,s_i}) + \alpha_{s_j} Y_{j,s_{j+1}} + \beta_s - h > 0 \\ \text{and} \quad & \sum_{i \neq j} M((\boldsymbol{\alpha}_s)_i, Y_{i,s_i}, Y_{i,s_{i+1}}) + \alpha_{s_j} Y_{j,s_{j+1}} + \beta_s - h < 0. \end{aligned} \quad (5.42)$$

If the cell to be tested has coordinates $\mathbf{t} = \mathbf{s} - \mathbf{e}_j$ then these conditions become

$$\begin{aligned} & \sum_{i \neq j} M((\boldsymbol{\alpha}_{\mathbf{s}})_i, Y_{i,s_i+1}, Y_{i,s_i}) + \alpha_{s_j} Y_{j,s_j} + \beta_{\mathbf{s}} - h > 0 \\ \text{and} \quad & \sum_{i \neq j} M((\boldsymbol{\alpha}_{\mathbf{s}})_i, Y_{i,s_i}, Y_{i,s_i+1}) + \alpha_{s_j} Y_{j,s_j} + \beta_{\mathbf{s}} - h < 0. \end{aligned} \quad (5.43)$$

Testing the validity condition - Linear programming

The test for validity is used on lines 7 and 12 of the algorithm in figure 5.6. In y -space the condition may be written

$$\mathcal{Y}_{\mathbf{s}} \cap \bar{\mathcal{I}}_{\mathbf{s}} \cap \left(\bigcap_{\mathbf{s}' \in A_i} \mathcal{I}_{\mathbf{s}'} \right) = \emptyset. \quad (5.44)$$

When this relationship holds, the half-plane $\mathcal{I}_{\mathbf{s}}$ does not need to be added to the simplex, $\bigcap_{\mathbf{s}' \in A_i} \mathcal{I}_{\mathbf{s}'}$.

Recall (figure 4.1 on page 97) that a cell is defined by two inequalities on each y -coordinate (these two inequalities form two opposite sides of the cell). Recall also that half-planes are defined by inequalities on a linear mixture of y -space coordinates. Hence determining whether condition (5.44) holds is equivalent to determining whether there is a solution to a set of linear inequalities. This is a linear programming problem, and in particular it is the so called *auxiliary problem* of the simplex algorithm. Details of how to solve the auxiliary problem with the simplex method can be found in [Chv80].

However, before applying the simplex method to test condition (5.44), it is worthwhile trying to simplify the problem. One possible simplification is to determine any coordinates $\mathbf{s}' \in A_i$ which satisfy $\mathcal{Y}_{\mathbf{s}} \cap \bar{\mathcal{I}}_{\mathbf{s}'} = \emptyset$. All such \mathbf{s}' 's may be removed from the intersection over $\mathcal{I}_{\mathbf{s}'}$'s in equation (5.44). The proof of this follows immediately from the identity

$$\mathcal{Y}_{\mathbf{s}} \cap \bar{\mathcal{I}}_{\mathbf{s}'} = \emptyset \quad \equiv \quad \mathcal{Y}_{\mathbf{s}} \cap \mathcal{I}_{\mathbf{s}'} = \mathcal{Y}_{\mathbf{s}}. \quad (5.45)$$

A second possible simplification is to determine whether there exists an $\mathbf{s}' \in A_i$ which



Figure 5.7: Two illustrations of the numerical problem between connected boundary cells. Only the left example causes a problem with the simplex algorithm.

satisfies $\mathcal{Y}_s \cap \bar{\mathcal{I}}_{s'} = \emptyset$. If so, then condition (5.44) must hold and no further testing using the simplex method is required.

The test for both of these simplifications follows the now familiar pattern:

$$\begin{aligned}
 \mathcal{Y}_s \cap \bar{\mathcal{I}}_{s'} = \emptyset &\equiv \min_{\mathbf{y} \in \mathcal{Y}_s} [\boldsymbol{\alpha}_{s'}^T \mathbf{y} + \beta_{s'} - h] > 0 \\
 &\equiv \sum_i M((\boldsymbol{\alpha}_{s'})_i, Y_{i,s_i}, Y_{i,s_i+1}) + \beta_{s'} - h > 0 \quad (5.46)
 \end{aligned}$$

and

$$\begin{aligned}
 \mathcal{Y}_s \cap \mathcal{I}_{s'} = \emptyset &\equiv \max_{\mathbf{y} \in \mathcal{Y}_s} [\boldsymbol{\alpha}_{s'}^T \mathbf{y} + \beta_{s'} - h] < 0 \\
 &\equiv \sum_i M((\boldsymbol{\alpha}_{s'})_i, Y_{i,s_i+1}, Y_{i,s_i}) + \beta_{s'} - h < 0. \quad (5.47)
 \end{aligned}$$

Testing the validity condition for connected boundary cells

There is a numerical problem with the linear programming method of testing the validity condition for connected boundary cells. The problem arises because the representation used to encode the cells and half-planes does not enforce the connected property. This implies that the half-planes of two connected boundary cells may not exactly intersect on the common side. The left hand side of figure 5.7 gives an example where numerical error has made $\mathcal{Y}_s \cap \bar{\mathcal{I}}_s \cap \mathcal{I}_{s'}$ not quite empty, although the connected property implies that it should be. Note that the problem only occurs when the intersection (the shaded region) is non-convex. Inspection of the right hand side of figure 5.7 should make this clear.

The solution to this numerical problem is based on the following observation: let \mathbf{y}_0 be a point in cell \mathcal{Y}_s and on the border between being in \mathcal{I}_s and being in $\bar{\mathcal{I}}_s$. If \mathbf{y}_0 is in $\mathcal{I}_{s'}$ then

clearly $\mathcal{Y}_s \cap \bar{\mathcal{I}}_s \cap \mathcal{I}_{s'} \neq \emptyset$. Less obviously, if \mathbf{y}_0 is not in $\mathcal{I}_{s'}$ then $\mathcal{Y}_s \cap \bar{\mathcal{I}}_s \cap \mathcal{I}_{s'} = \emptyset$. This follows because the half-planes from two connected cells should only intersect on the cell side. Combining these two implications gives the following equivalence:

$$\mathcal{Y}_s \cap \bar{\mathcal{I}}_s \cap \mathcal{I}_{s'} = \emptyset \quad \equiv \quad \mathbf{y}_0 \notin \mathcal{I}_{s'} \quad (5.48)$$

where cells s and s' are connected and \mathbf{y}_0 satisfies $\boldsymbol{\alpha}_s^T \mathbf{y}_0 + \beta_s - h = 0$ and $\mathbf{y}_0 \in \mathcal{Y}_s$.

It is easy to avoid the numerical problem by using equation (5.48) with a \mathbf{y}_0 which is not close to the side common to both cells. Such a \mathbf{y}_0 can be found as follows. Define \mathbf{y}_{\max} and \mathbf{y}_{\min}

$$\begin{aligned} \mathbf{y}_{\max} &= \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}_s} [\boldsymbol{\alpha}_s^T \mathbf{y} + \beta_s - h] \\ \mathbf{y}_{\min} &= \operatorname{argmin}_{\mathbf{y} \in \mathcal{Y}_s} [\boldsymbol{\alpha}_s^T \mathbf{y} + \beta_s - h], \end{aligned} \quad (5.49)$$

which can be found in the usual way. Note that if all the components of $\boldsymbol{\alpha}_s$ are non-zero, then \mathbf{y}_{\max} and \mathbf{y}_{\min} are guaranteed to be at opposite corners of the cell \mathcal{Y}_s . Furthermore, for those components of $\boldsymbol{\alpha}_s$ which are zero, opposite corners can be chosen arbitrarily. Let \mathbf{y}_0 take the form

$$\mathbf{y}_0 = \lambda \mathbf{y}_{\min} + (1 - \lambda) \mathbf{y}_{\max}. \quad (5.50)$$

Solving for the λ which satisfies $\boldsymbol{\alpha}_s^T \mathbf{y}_0 + \beta_s - h = 0$ gives

$$\lambda = \frac{\boldsymbol{\alpha}_s^T \mathbf{y}_{\max} + \beta_s - h}{\boldsymbol{\alpha}_s^T (\mathbf{y}_{\max} - \mathbf{y}_{\min})}. \quad (5.51)$$

Note that by the construction of \mathbf{y}_{\max} and \mathbf{y}_{\min} , and because a cell is a convex region, a solution must exist with $0 \leq \lambda \leq 1$. A two dimensional illustration of the process is shown in figure 5.8.

The above method should be used to check each cell $s' \in A_i$ which is connected to s . In effect, this is a third possible simplification of condition (5.44) which can be tested for

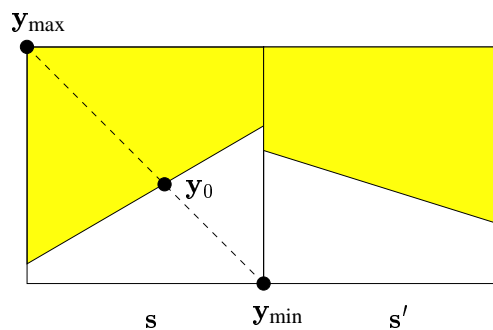


Figure 5.8: A two dimensional illustration of the construction of \mathbf{y}_0 .

before applying the simplex algorithm. However this simplification does more than just make the test more efficient, it also solves an important numerical problem.

Removing redundant half-planes

The last operation required by the algorithm of figure 5.6 is the removal of redundant half-planes from a simplex. It is actually more useful to do this in x -space because all the half-planes will eventually be mapped into x -space, and if the number of hidden nodes is greater than the number of input nodes then this mapping will introduce additional redundancies.

Thus the problem can be stated as follows: remove all the redundant half-planes from the simplex $\mathbf{S}_i = \bigcap_{\mathbf{s} \in A_i} \mathcal{H}_{\mathbf{s}}$. This can be achieved by testing each half-plane for redundancy in turn. Obviously the half-plane $\mathcal{H}_{\mathbf{s}'}$ is redundant if

$$\bigcap_{\mathbf{s} \in A_i - \mathbf{s}'} \mathcal{H}_{\mathbf{s}} = \bigcap_{\mathbf{s} \in A_i} \mathcal{H}_{\mathbf{s}}. \quad (5.52)$$

Intersecting both sides with $\bar{\mathcal{H}}_{\mathbf{s}'}$ gives the following equivalent condition for redundancy

$$\left(\bigcap_{\mathbf{s} \in A_i - \mathbf{s}'} \mathcal{H}_{\mathbf{s}} \right) \cap \bar{\mathcal{H}}_{\mathbf{s}'} = \emptyset. \quad (5.53)$$

Since every half-plane is defined with a linear inequality, a straightforward application of

the auxiliary problem of the simplex method can be used to determine whether condition (5.53) holds.

5.4 Translating \mathcal{R}_k^2 into a rule-base

This section presents the final stage of the rule extraction process. The method described below will generate a rule-base which contains rules of all classes and of any of the three types described in chapter 3. Parsimony and effectiveness will be heuristically maximised, but comparative safety will be guaranteed because every rule of class k will be a subset of \mathcal{R}_k^2 .

5.4.1 Introduction

Assume that \mathcal{R}_k^2 regions have been found for all classes k . We would like to describe these regions using a parsimonious and effective rule-base which satisfies the rule condition. This can be achieved by first generating a large number of rules within each \mathcal{R}_k^2 , and then putting all these rules into a single rule-base. This rule-base is likely to have a high effectiveness but extremely low parsimony. However this imbalance can then be rectified by finding a maximally effective subset of these rules, where the number of rules in the subset is a specified quantity which controls the parsimony of the final rule-base. Thus the whole process has two phases: first obtain a large number of rules which form an effective but unparsimonious rule-base; then reduce the size of this rule-base to increase parsimony without significantly reducing effectiveness.

The advantage of this two phase approach is that the individual optimisation of each rule is separated from the optimisation of the rule-base. Note that since a deterministic rule optimisation algorithm will always produce the same rule from a given \mathcal{R}_k^2 , the above approach requires a random component. This randomness ensures that each rule is different, providing a variety of rules from which to pick an effective rule-base.

Although the descriptions which follow will all use the family of rules for ordered variables described in chapter 3, this does not imply that the rules for categorical variables cannot be extracted. In fact the process is identical to the extraction of rules for ordered variables except for an additional final re-writing phase. This re-writing simply involves evaluating the rules on the codes for each categorical value (see section 3.1.4 on page 60).

5.4.2 Obtaining an effective (but unparsimonious) rule-base

The first part of the algorithm generates N class k rules which are all subsets of \mathcal{R}_k^2 . The approach will involve *seeding* each rule with a sample from the unconditional density² and then *growing* a rule antecedent region centred on this seed. The steps required to achieve this are given below:

1. Randomly select a simplex from the \mathcal{R}_k^2 representation. Write this simplex $\mathcal{S}_i = \bigcap_{\mathbf{s} \in A_i} \mathcal{H}_s$ where $\mathcal{H}_s = \{ \mathbf{x} \mid \mathbf{a}_s^T \mathbf{x} + b_s - h \geq 0 \}$.
2. Sample a pattern from $\hat{p}(\mathbf{x})$ which falls within \mathcal{S}_i and use this pattern as the prototype $\boldsymbol{\mu}$ of a new class k rule.
3. Maximally grow a rule antecedent region \mathbf{r} about the prototype, ensuring that \mathbf{r} remains a subset of the simplex, $\mathbf{r} \subseteq \mathcal{S}_i$. This is done in two stages: isotropic growth followed by anisotropic growth.
4. Add the rule to the rule-base, and return to step 1 if the number of rules is less than N .

This method produces rules which satisfy the subset condition because

$$\mathbf{r} \subseteq \mathcal{S}_i \quad \Rightarrow \quad \mathbf{r} \subseteq \bigcup_i \mathcal{S}_i \quad \Rightarrow \quad \mathbf{r} \subseteq \mathcal{R}_k^2. \quad (5.54)$$

A key part of the above method is the isotropic and anisotropic growth. Roughly speaking, isotropic growth involves expanding all sides of the rule antecedent region simultaneously,

²It is straightforward to sample patterns from the kernel based density model described in appendix B. See for example [Sil96].

whereas anisotropic growth only expands a single side of the rule antecedent region at a time. Both are required because isotropic growth alone may produce a rule antecedent region with sides which can be further expanded, and anisotropic growth alone tends to produce antecedent regions which are very thin in some directions. The latter effect results in rules with a very low effectiveness, which is clearly undesirable.

The first section below derives the conditions which must be satisfied for each of the three types of rule antecedent regions to be a subset of the simplex \mathcal{S}_i . The two subsequent sections show how these conditions can be used to grow rules both isotropically and anisotropically.

Conditions for rule validity

It is useful to rewrite the condition that the rule antecedent region is a subset of the simplex,

$$\begin{aligned} \mathbf{r} \subseteq \mathcal{S}_i &\equiv \mathbf{r} \subseteq \bigcap_{\mathbf{s} \in A_i} \mathcal{H}_{\mathbf{s}} \\ &\equiv \bigwedge_{\mathbf{s} \in A_i} \mathbf{r} \subseteq \mathcal{H}_{\mathbf{s}}. \end{aligned} \quad (5.55)$$

Hence a valid rule must be entirely on the positive side of each hyper-plane forming the simplex. Conditions for each of the three rule antecedent regions to be on the positive side of a given hyperplane will now be derived.

First consider the city-block rule ($p = 1$ case of figure 3.2 on page 55),

$$\mathbf{r} = \left\{ \mathbf{x} \left| \sum_j \left(\frac{x_j - \mu_j}{u_j} \right) U(x_j - \mu_j) + \left(\frac{\mu_j - x_j}{v_j} \right) U(\mu_j - x_j) \leq 1 \right. \right\} \quad (5.56)$$

Note that both \mathbf{r} and the half-plane $\mathcal{H}_{\mathbf{s}}$ are simplexes, hence $\mathbf{r} \subseteq \mathcal{H}_{\mathbf{s}}$ if and only if every

corner of \mathbf{r} is in \mathcal{H}_s . There are two corners on each axis, and so $\mathbf{r} \subseteq \mathcal{H}_s$ if and only if

$$\begin{aligned} & \bigwedge_j \mathbf{a}_s^T \boldsymbol{\mu} + b_s - h + (\mathbf{a}_s)_j u_j \geq 0 \\ \text{and} \quad & \bigwedge_j \mathbf{a}_s^T \boldsymbol{\mu} + b_s - h - (\mathbf{a}_s)_j v_j \geq 0. \end{aligned} \quad (5.57)$$

Note that if $(\mathbf{a}_s)_j \geq 0$ then the condition with v_j implies the condition with u_j , and conversely if $(\mathbf{a}_s)_j \leq 0$ then the condition with u_j implies the condition with v_j . Hence equation (5.57) simplifies to

$$\begin{aligned} & \bigwedge_j \mathbf{a}_s^T \boldsymbol{\mu} + b_s - h + M((\mathbf{a}_s)_j, -v_j, u_j) \geq 0 \\ \equiv & \mathbf{a}_s^T \boldsymbol{\mu} + b_s - h + \min_j M((\mathbf{a}_s)_j, -v_j, u_j) \geq 0. \end{aligned} \quad (5.58)$$

Next consider the axis aligned rule ($p = \infty$ case of figure 3.2 on page 55),

$$\mathbf{r} = \left\{ \mathbf{x} \left| \bigwedge_j \left(\frac{x_j - \mu_j}{u_j} \right) U(x_j - \mu_j) + \left(\frac{\mu_j - x_j}{v_j} \right) U(\mu_j - x_j) \leq 1 \right. \right\}. \quad (5.59)$$

As with the case above, both the axis aligned region and the half-plane are simplexes, so $\mathbf{r} \subseteq \mathcal{H}_s$ if and only if every corner of \mathbf{r} is in \mathcal{H}_s . These corners are at $\mu_j + M(c_j, u_j, v_j)$ where $c_j = \pm 1$. Thus $\mathbf{r} \subseteq \mathcal{H}_s$ if and only if

$$\bigwedge_{\mathbf{c} \in \mathbf{C}} \mathbf{a}_s^T \boldsymbol{\mu} + b_s - h + \sum_j (\mathbf{a}_s)_j M(c_j, u_j, v_j) \geq 0. \quad (5.60)$$

where $\mathbf{C} = \{\mathbf{c} | c_j = \pm 1\}$. This simplifies in a similar way to the city-block rule condition,

$$\begin{aligned} \equiv & \mathbf{a}_s^T \boldsymbol{\mu} + b_s - h + \min_{\mathbf{c} \in \mathbf{C}} \sum_j (\mathbf{a}_s)_j M(c_j, u_j, v_j) \geq 0 \\ \equiv & \mathbf{a}_s^T \boldsymbol{\mu} + b_s - h + \sum_j M((\mathbf{a}_s)_j, -v_j, u_j) \geq 0. \end{aligned} \quad (5.61)$$

Lastly, consider the euclidean rule ($p = 2$ case of figure 3.2 on page 55),

$$\mathbf{r} = \left\{ \mathbf{x} \mid \bigwedge_j \left(\frac{x_j - \mu_j}{u_j} \right)^2 U(x_j - \mu_j) + \left(\frac{\mu_j - x_j}{v_j} \right)^2 U(\mu_j - x_j) \leq 1 \right\}. \quad (5.62)$$

This case is different from the other two because the rule region does not have any corners. Consider minimising $\mathbf{a}_s^T \mathbf{x} + b_s - h$ subject to the constraint that \mathbf{x} is on the boundary of the rule antecedent region,

$$\sum_j \left(\frac{x_j - \mu_j}{t_j} \right)^2 = 1, \quad (5.63)$$

where the t_j will be chosen later. It is easy to show that the minimum is at \mathbf{x}_0 where

$$(\mathbf{x}_0)_j = \mu_j - \frac{(\mathbf{a}_s)_j t_j^2}{\sqrt{\sum_j ((\mathbf{a}_s)_j t_j)}}. \quad (5.64)$$

This implies that $\text{sign}((\mathbf{x}_0)_j - \mu_j) = -\text{sign}((\mathbf{a}_s)_j)$, which is independent of t_j . Hence the t_j can be set to the appropriate u_j or v_j , $(\mathbf{a}_s)_j t_j = M((\mathbf{a}_s)_j, v_j, u_j)$. Clearly $\mathbf{a}_s^T \mathbf{x}_0 + b_s - h$ must be positive for $\mathbf{r} \subseteq \mathcal{H}_s$, so

$$\begin{aligned} & \mathbf{a}_s^T \mathbf{x}_0 + b_s - h \geq 0 \\ \equiv & \mathbf{a}_s^T \boldsymbol{\mu} + b_s - h - \sqrt{\sum_j M((\mathbf{a}_s)_j, -v_j, u_j)^2} \geq 0. \end{aligned} \quad (5.65)$$

The conditions for all three types of rule are summarised in table 5.1. There are two intuitive interpretations of these conditions. First, they all imply that the rule prototype must be in the half-plane, $\boldsymbol{\mu} \in \mathcal{H}_s$. Second, they all use the same combination of u_j & v_j , and this combination depends only on the signs of the components of the normal to the hyperplane, $(\mathbf{a}_s)_j$, defining the half-plane.

Rule type	Condition for $\mathbf{r} \subseteq \mathcal{H}_s$
$p = 1$	$\mathbf{a}_s^T \boldsymbol{\mu} + b_s - h + \min_j M((\mathbf{a}_s)_j, -v_j, u_j) \geq 0$
$p = 2$	$\mathbf{a}_s^T \boldsymbol{\mu} + b_s - h - \sqrt{\sum_j M((\mathbf{a}_s)_j, -v_j, u_j)^2} \geq 0$
$p = \infty$	$\mathbf{a}_s^T \boldsymbol{\mu} + b_s - h + \sum_j M((\mathbf{a}_s)_j, -v_j, u_j) \geq 0$

Table 5.1: Conditions for each of the three types of rule to be on the positive side of a hyper-plane.

Isotropic rule growth

Isotropic rule growth involves expanding all sides of the rule antecedent region simultaneously. Let the expansion be controlled by a positive parameter α and a vector \mathbf{t} of positive elements such that

$$u_i = v_i = \alpha t_i. \quad (5.66)$$

Thus for a given \mathbf{t} , maximal isotropic growth is achieved by the largest α which ensures that the rule region is still a subset of the simplex \mathcal{S}_i .

Consider a city-block rule \mathbf{r} being isotropically expanded within the simplex \mathcal{S}_i . Using equation (5.55), the isotropic constraint equation (5.66), and the ($p = 1$) condition of figure 5.1,

$$\begin{aligned}
\mathbf{r} \subseteq \mathcal{S}_i &\equiv \bigwedge_{s \in A_i} \mathbf{a}_s^T \boldsymbol{\mu} + b_s - h + \min_j M((\mathbf{a}_s)_j, -\alpha t_j, \alpha t_j) \geq 0 \\
&\equiv \bigwedge_{s \in A_i} \mathbf{a}_s^T \boldsymbol{\mu} + b_s - h - \alpha \max_j |(\mathbf{a}_s)_j t_j| \geq 0 \\
&\equiv \bigwedge_{s \in A_i} \alpha \leq \frac{\mathbf{a}_s^T \boldsymbol{\mu} + b_s - h}{\max_j |(\mathbf{a}_s)_j t_j|} \\
&\equiv \alpha \leq \min_{s \in A_i} \left[\frac{\mathbf{a}_s^T \boldsymbol{\mu} + b_s - h}{\max_j |(\mathbf{a}_s)_j t_j|} \right]. \quad (5.67)
\end{aligned}$$

The maximum isotropic growth is given by changing the inequality into an equality. A

Rule type	Maximum isotropic growth
$p = 1$	$\alpha = \min_{s \in A_i} \frac{\mathbf{a}_s^T \boldsymbol{\mu} + b_s - h}{\max_j (\mathbf{a}_s)_j t_j }$
$p = 2$	$\alpha = \min_{s \in A_i} \frac{\mathbf{a}_s^T \boldsymbol{\mu} + b_s - h}{\sqrt{\sum_j ((\mathbf{a}_s)_j t_j)^2}}$
$p = \infty$	$\alpha = \min_{s \in A_i} \frac{\mathbf{a}_s^T \boldsymbol{\mu} + b_s - h}{\sum_j (\mathbf{a}_s)_j t_j }$

Table 5.2: Maximum isotropic growth, α , for each of the three types of rule such that the rule antecedent region is within the simplex, $\mathbf{r} \subseteq \mathcal{S}_i$.

very similar process gives the maximum α for the other two types of rule. The results of maximum isotropic growth for all three types of rule are summarised in table 5.2.

There are several possible choices for the isotropic growth “direction” \mathbf{t} . One possibility is to set \mathbf{t} to be a vector of one’s. However this does not take into account that different dimensions will have different units and so should be scaled by different amounts. Another possibility is to use the component-wise standard deviation of a sample of patterns from $\hat{p}(\mathbf{s})$ which fall within the simplex \mathcal{S}_i . However the simplest solution which works well is to obtain, by sampling, one other pattern \mathbf{x} which is in \mathcal{S}_i . The “direction” of isotropic growth can then be set by

$$t_j = |\mu_j - x_j|. \quad (5.68)$$

Anisotropic rule growth

An isotropically grown rule is unlikely to be optimal because it is possible that directions can be independently expanded further. This second expansion phase is called anisotropic rule growth.

The idea is to maximise each u_j & v_j in turn. The order in which this is done will clearly have an effect on the final antecedent region. Since it is difficult to determine the optimal

order, the u_j 's & v_j 's are chosen in a random order. This is a heuristic which works well in practice.

Suppose the maximum value of u_j is required which ensures the rule remains within the simplex. Define

$$\mathbf{u}' = \mathbf{u} + (\alpha - u_j)\mathbf{e}_j \quad (5.69)$$

where \mathbf{e}_j is the j^{th} unit vector. Then the condition for the corresponding city-block rule \mathbf{r} to be a subset of \mathcal{S}_i is given by

$$\bigwedge_{\mathbf{s} \in A_i} \mathbf{a}_{\mathbf{s}}^T \boldsymbol{\mu} + b_{\mathbf{s}} - h + \min_{j'} M((\mathbf{a}_{\mathbf{s}})_{j'}, -v_{j'}, u'_{j'}) \geq 0. \quad (5.70)$$

Since the rule is assumed to satisfy the condition for $\alpha = 0$, only the conjunctions and min terms which use u'_j need to be considered. Hence only those half-planes with $(\mathbf{a}_{\mathbf{s}})_j < 0$ need to be considered,

$$\bigwedge_{\mathbf{s} \in A_i \wedge (\mathbf{a}_{\mathbf{s}})_j < 0} \mathbf{a}_{\mathbf{s}}^T \boldsymbol{\mu} + b_{\mathbf{s}} - h + (\mathbf{a}_{\mathbf{s}})_j u'_j \geq 0, \quad (5.71)$$

from which it follows that

$$\alpha \leq \min_{\mathbf{s} \in A_i \wedge (\mathbf{a}_{\mathbf{s}})_j < 0} \left[\frac{\mathbf{a}_{\mathbf{s}}^T \boldsymbol{\mu} + b_{\mathbf{s}} - h}{-(\mathbf{a}_{\mathbf{s}})_j} \right]. \quad (5.72)$$

If instead of maximising u_j we wish to maximise v_j , then with \mathbf{v}' defined

$$\mathbf{v}' = \mathbf{v} + (\beta - v_j)\mathbf{e}_j \quad (5.73)$$

it is straight forward to show that the condition becomes

$$\beta \leq \min_{\mathbf{s} \in A_i \wedge (\mathbf{a}_{\mathbf{s}})_j > 0} \left[\frac{\mathbf{a}_{\mathbf{s}}^T \boldsymbol{\mu} + b_{\mathbf{s}} - h}{(\mathbf{a}_{\mathbf{s}})_j} \right]. \quad (5.74)$$

Rule type	Maximum anisotropic growth	
$p = 1$	$\alpha = \min_{\mathbf{s} \in A_i \wedge (\mathbf{a}_s)_j < 0}$	$\left[\frac{\mathbf{a}_s^T \boldsymbol{\mu} + b_s - h}{-(\mathbf{a}_s)_j} \right]$
$p = 2$	$\alpha = \min_{\mathbf{s} \in A_i \wedge (\mathbf{a}_s)_j < 0}$	$\left[\frac{(\mathbf{a}_s^T \boldsymbol{\mu} + b_s - h)^2 - \sum_{j' \neq j} M((\mathbf{a}_s)_{j'}, -v_{j'}, u_{j'})^2}{(\mathbf{a}_s)_j^2} \right]^{0.5}$
$p = \infty$	$\alpha = \min_{\mathbf{s} \in A_i \wedge (\mathbf{a}_s)_j < 0}$	$\left[\frac{\mathbf{a}_s^T \boldsymbol{\mu} + b_s - h + \sum_{j' \neq j} M((\mathbf{a}_s)_{j'}, -v_{j'}, u_{j'})}{-(\mathbf{a}_s)_j} \right]$

Table 5.3: Maximum anisotropic growth, α where $\mathbf{u}' = \mathbf{u} + (\alpha - u_j)\mathbf{e}_j$, for each of the three types of rule such that the rule antecedent region is within the simplex, $\mathbf{r} \subseteq \mathcal{S}$.

Rule type	Maximum anisotropic growth	
$p = 1$	$\beta = \min_{\mathbf{s} \in A_i \wedge (\mathbf{a}_s)_j > 0}$	$\left[\frac{\mathbf{a}_s^T \boldsymbol{\mu} + b_s - h}{(\mathbf{a}_s)_j} \right]$
$p = 2$	$\beta = \min_{\mathbf{s} \in A_i \wedge (\mathbf{a}_s)_j > 0}$	$\left[\frac{(\mathbf{a}_s^T \boldsymbol{\mu} + b_s - h)^2 - \sum_{j' \neq j} M((\mathbf{a}_s)_{j'}, -v_{j'}, u_{j'})^2}{(\mathbf{a}_s)_j^2} \right]^{0.5}$
$p = \infty$	$\beta = \min_{\mathbf{s} \in A_i \wedge (\mathbf{a}_s)_j > 0}$	$\left[\frac{\mathbf{a}_s^T \boldsymbol{\mu} + b_s - h + \sum_{j' \neq j} M((\mathbf{a}_s)_{j'}, -v_{j'}, u_{j'})}{(\mathbf{a}_s)_j} \right]$

Table 5.4: Maximum anisotropic growth, β where $\mathbf{v}' = \mathbf{v} + (\beta - v_j)\mathbf{e}_j$, for each of the three types of rule such that the rule antecedent region is within the simplex, $\mathbf{r} \subseteq \mathcal{S}$.

Very similar results may be obtained for the other two types of rules. These are given in tables 5.3 and 5.4.

5.4.3 Reducing the number of rules

The procedure described above can be used to obtain a large number of rules within each \mathcal{R}_k^2 region for each class k . Merging these rules into a single rule-base produces a large rule-base which is likely to have a high effectiveness at the expense of being extremely unparsimonious. Obviously the number of rules must be reduced. The optimum solution involves searching over all subsets of size $n < N$, and selecting the subset which is the most effective. However using this approach alone is impractical because there are $\binom{N}{n}$

subsets, and typical values are $N = 10000$, $n = 5$, giving $\binom{N}{n} \approx 8 \times 10^{17}$. The solution is therefore firstly to reduce the number of rules using sub-optimal but reasonably fast methods before applying a search over all combinations. Two such algorithms, a greedy forward selection and a greedy backward pruning, are described below. Obviously all rules with an individual effectiveness less than some threshold could be removed prior to employing these algorithms.

Greedy forward selection

This algorithm works by accumulating rules in a new rule-base, adding each time the rule which gives rise to the largest increase in effectiveness of the new rule-base. A reasonably efficient implementation is given below:

1. Let \mathbf{R}_0 be the rule-base of many rules. Initialise $\mathbf{R} = \{\}$
2. Initialise \mathbf{S} = a large sample of patterns from $\hat{p}(\mathbf{x})$
3. Find the rule $\mathbf{r} \in \mathbf{R}_0$ which covers the largest number of patterns in \mathbf{S}
4. Remove from \mathbf{S} all patterns which activate \mathbf{r}
5. Add \mathbf{r} to \mathbf{R} and remove \mathbf{r} from \mathbf{R}_0
6. Return to step 3 until \mathbf{R} contains the desired number of rules.

Greedy backward pruning

This algorithm works by repeatedly eliminating rules from the rule-base, each time removing the rule which gives rise to the smallest reduction in effectiveness. A possible implementation is given below:

1. Let \mathbf{R} be the rule-base to be reduced
2. Initialise \mathbf{S} = a large sample of patterns from $\hat{p}(\mathbf{x})$
3. Find the rule $\mathbf{r} \in \mathbf{R}$ such that $(\mathbf{R} - \mathbf{r})$ covers the largest number of patterns in \mathbf{S}
4. Remove \mathbf{r} from \mathbf{R}
5. Return to step 3 until \mathbf{R} contains the desired number of rules.

Illustrative example

To clarify the use of these two greedy algorithms, consider the example of wanting to reduce a rulebase of 1000 rules to a rulebase of 5 rules. Using some large sample of patterns from $\hat{p}(\mathbf{x})$, remove from the rulebase all rules which cover fewer than 100 patterns. Then use the greedy forward selection procedure to reduce the rulebase to 50 rules, followed by greedy backward pruning to reduce this further to 15 rules. Finally, use an all combinations search to find the most effective subset of 5 rules. Although this is a very simple procedure which could probably be improved in a number of ways, it was found to be an effective and practical solution to the problem of increasing parsimony without avoidably decreasing effectiveness.

5.4.4 Reducing the complexity of the rules

The above algorithm produces a rule-base of rules, where each class k rule is a subset of one of the simplexes from \mathcal{R}_k^2 ,

$$\text{each class } k \text{ rule } \mathbf{r} \text{ satisfies } \mathbf{r} \subseteq \mathcal{S}_{k,i} \text{ for some } i \quad (5.75)$$

where

$$\mathcal{R}_k^2 = \bar{\mathcal{N}} \cup \mathcal{N} \cap \bigcup_i \mathcal{S}_{k,i}. \quad (5.76)$$

Observe that

$$\begin{aligned} \mathbf{r} \subseteq \mathcal{R}_k^2 &\equiv \mathbf{r} \cap \mathcal{R}_k^2 = \mathbf{r} \\ &\equiv (\mathbf{r} \cap \bar{\mathcal{N}}) \cup (\mathbf{r} \cap \mathcal{N} \cap \bigcup_i \mathcal{S}_{k,i}) = (\mathbf{r} \cap \bar{\mathcal{N}}) \cup (\mathbf{r} \cap \mathcal{N}) \\ &\equiv \mathbf{r} \cap \mathcal{N} \cap \bigcup_i \mathcal{S}_{k,i} = \mathbf{r} \cap \mathcal{N} \\ &\equiv \mathbf{r} \cap \mathcal{N} \subseteq \bigcup_i \mathcal{S}_{k,i}. \end{aligned} \quad (5.77)$$

This makes it obvious that

$$\mathbf{r} \subseteq \mathcal{S}_{k,i} \Rightarrow \mathbf{r} \subseteq \mathcal{R}_k^2 \quad (5.78)$$

thus verifying that such rules satisfy the subset condition (equation 5.3). However equation (5.77) also demonstrates that rules are only constrained in regions of normality. Exploiting freedom in regions of novelty may make it possible to simplify some of the rules. Essentially, rules can be grown further so long as the newly covered region is entirely novel. An example of this was given at the end of chapter 3. In practice it is difficult to achieve such simplifications because the definition of normality, $\mathcal{N} = \{\mathbf{x} | \hat{p}(\mathbf{x}) > t\}$ is an inconveniently defined region of space.

Note that it is dangerous to rely on a novelty test which is based on a sample of normal patterns. This is because high-dimensional space cannot be adequately filled by a sample of points, and so there is no guarantee that the extra space covered by growing the rule further does not intersect with normality. If it does, then it may also cross the safety threshold, thus losing the rule's validity.

In conclusion, reducing the complexity of the rules in this way requires further research (see chapter 7).

5.5 Summary

This chapter has described a series of methods which combine to extract a rule based explanation of a feed-forward network classifier. These rules have the vital property that they have a guaranteed comparative safety, which essentially specifies a bound on the difference between the rule-based explanation and the network's unexplained classification. In addition to ensuring comparative safety, several algorithms have been suggested which attempt to optimise the trade-offs between effectiveness and parsimony, and thus provide a rule-base which can reasonably be said to explain.

Figure 5.9 provides a summary of the content of this chapter, and how it combines with the contents of the previous chapters to form a complete rule extraction process. Many of the details have consisted of techniques for simplifying representations of regions of input space. This began with an application of the MLP piece-wise linearisation method to translate an implicit description of each safety region into an explicit representation in terms of cells and half-planes. Rather than attempting to translate this representation directly into rules, it was found to be expedient to use an intermediate representation consisting of a union of simplexes. Thus it was explained how a rule-base of prototype based rules could be created by “seeding” and “growing” antecedent regions within randomly chosen simplexes. These rules could be of any of the three p -norm types: city-block ($p = 1$), euclidean ($p = 2$), and axis-aligned ($p = \infty$). Finally, several algorithms were suggested for finding an effective subset of a large number of such rules, thus producing the end product: a rule-base with a guaranteed comparative safety and heuristically optimised parsimony and effectiveness.

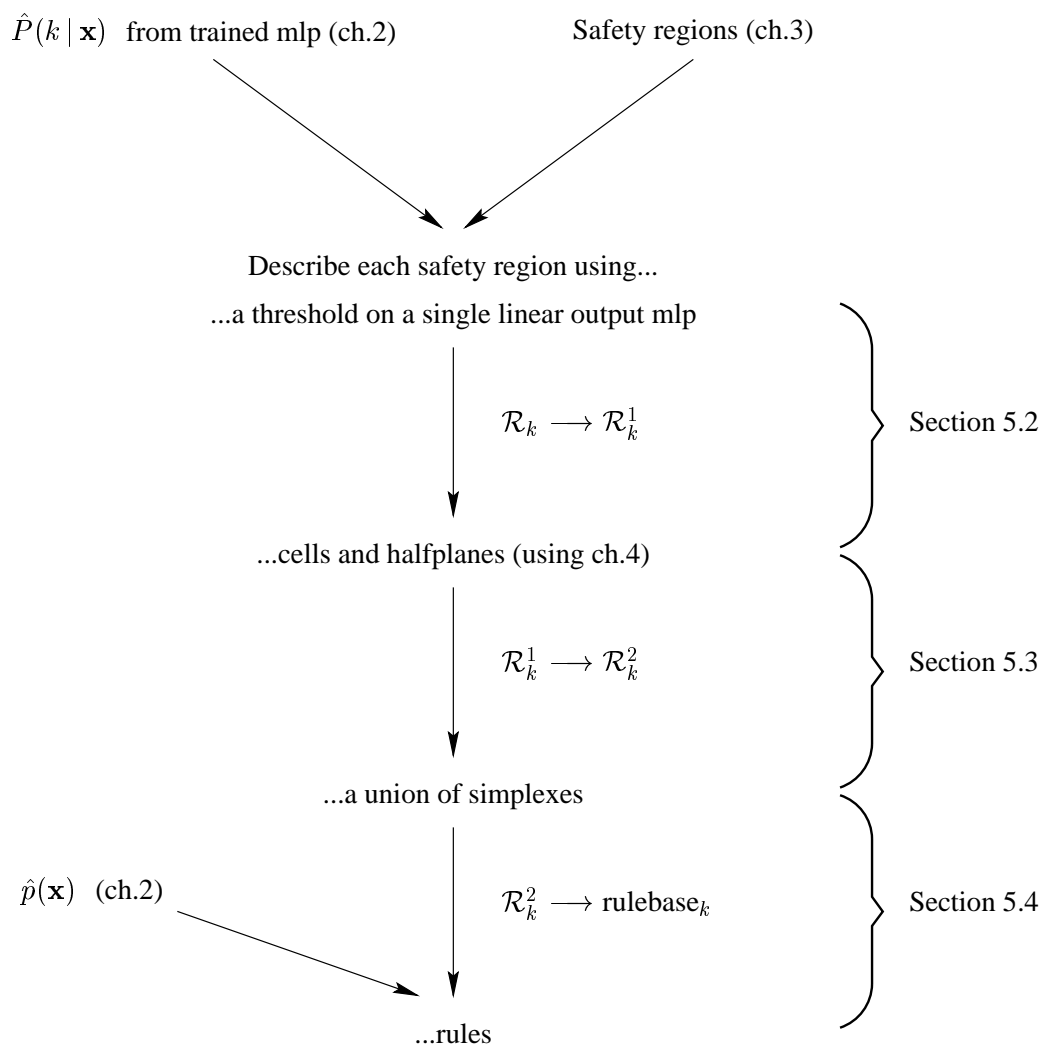


Figure 5.9: A summary of the rule extraction process.

Chapter 6

Results

This chapter presents the results of applying the rule extraction method on three non-artificial datasets. The first section describes the criteria used to select suitable datasets, and the second section discusses some general issues relating to implementation and experimental procedure. The final three sections present the results for each of the three datasets.

6.1 Datasets

There are numerous repositories of datasets, such as the UCI machine learning databases archive [MA95], the CMU nnbench [WF98], Proben1 [Pre94], StatLog [MST94], ELENA [JGDAC⁺98], and DELVE [RNH⁺98]. Two very useful sources are StatLib [Sta98], and the benchmarking of learning algorithms information repository page [Pre98]. There have also been several attempts to provide standard benchmarks so that different algorithms can be compared under reproducible conditions, [MST94, Pre94, Zhe93, RNH⁺98, WF98].

The principal criteria used to select the test problems were as follows:

1. The dataset must come from a real problem. Several datasets such as the MONKS problems are artificially generated and were not therefore considered a realistic test.

2. The variables in the dataset must have semantics. Clearly if the variables do not mean anything then neither will an explanation in terms of these variables. Some semantics are withheld for reasons of confidentiality (*e.g.* the German credit dataset from StatLog), and some variables are not intuitively interpretable, for example auto-regressive or reflection coefficients [Mak75].
3. The problem must require a statistical approach. Some datasets are essentially non-stochastic (such as the exclusive-or problem) and hence an inappropriate test for the method described in this thesis.
4. The classification problem must be non-trivial. Datasets such as the Fisher iris or the crabs data are very easy to classify.

A secondary criteria is that the problem must have only two classes. This is due to a limitation of the implementation (See section 5.2.2 on page 122).

The above considerations led to three datasets being chosen to assess the rule extraction method developed in this thesis. The first is a heart disease dataset for which the task is to predict the presence or absence of heart disease from a variety of medical tests carried out on patients (section 6.3). The task in the second problem is to predict the presence or absence of diabetes in women of Pima Indian descent (section 6.4). Lastly, the third problem is a breast cancer dataset for which the task is predict whether a breast tissue sample is cancerous or non-cancerous (section 6.5).

6.2 Preamble to results

Central to the results is an evaluation of the similarities and differences between the extracted rule-base and the network. It is this comparison which should be used to judge whether a satisfactory explanation of the network-based classifier has been obtained.

A comparison with other algorithms described in the literature would only have limited value because in most cases there is no tight coupling between the feed-forward network and the rules. It is this coupling, controlled via comparative safety, which is a primary

focus of this thesis. Those algorithms which do have a guaranteed equivalence between the explanation and network also require a large number of rules to cover a reasonable fraction of all data. Thus the parsimony and effectiveness of the final rule-based explanations are also of vital importance. For these reasons, no other rule extraction algorithm was implemented for comparison, although reference is made to any published results (classification performance, explanation *etc*) on the datasets studied.

6.2.1 Implementation details

The complete classification system was implemented in the Java programming language, [Jav98]. The three sub-sections below describe some of the practical details which were common to all three datasets.

Estimating conditional class probabilities, $P(k | \mathbf{x})$

Obtaining an estimate of the conditional class probabilities was achieved using an MLP as follows. First, the optimal network *architecture* (number of hidden nodes and amount of weight decay) was determined using 10-fold cross-validation. This involved training a network with a given architecture on data from 9 of the 10 partitions and evaluating the performance on the data from the 10th partition. This was repeated for all 10 partitions and the results averaged. Average negative log probability scores were used to select the optimal architecture, where the negative logarithm probability score is the value of the error function (without weight decay) evaluated on data not used for training. This measure was preferred over average classification rate for two reasons. First, at this stage the aim is to obtain the best estimate of the probability function. The accuracy of this estimate is given by the likelihood, which is one reason why the negative log likelihood is the error function to be minimised by the whole process. Second, classification rate is estimated by counting patterns and is therefore quantised into discrete values. The resolution of this discretisation is low for relatively small datasets.

Once the optimal number of hidden nodes and weight decay coefficient had been determined, all the training data was used in a second phase to train the final network.

Each network was trained using the scaled conjugate gradient algorithm [Mø193]. This is a far superior method than gradient descent, although like all “hill climbing” algorithms, it suffers from the possibility of converging to a local minimum. Hence the best result from three random initial conditions was chosen to reduce the effects of this problem.

Estimating unconditional density, $p(\mathbf{x})$

Estimating the unconditional probability density function was achieved using the kernel based method described in appendix B. Recall that there are two reasons for estimating unconditional density. First, the novelty detector uses a threshold on the unconditional density estimate to decide the normality of new patterns. Second, the unconditional density estimate is used to generate sample patterns required both to seed rules and to assess the effectiveness of these rules. However it was decided that a novelty detector was not required on any of the three test problems because there was no reason to believe that any of the patterns were novel. Note that novelty detection *would* be required to detect changes in the distribution of the data if the system was being used on-line to classify new patterns.

Extracting a rule-base

The rule extraction algorithm described in this thesis has several user controlled parameters, chosen by a combination of experimentation, an intuitive balancing of parsimony, effectiveness, and comparative safety, and bearing in mind the requirement for the algorithm to finish in a reasonable length of time. Listed below are the more important details.

No MLP was linearised to an accuracy requiring more than a few tens of thousands of cells. Initially comparative safety was always set to 1.0, but if the effectiveness/parsimony of the final rule-base was unacceptable then a smaller figure was tried.

Once a piecewise linear representation of the MLP had been obtained, a large rule-base of 1000 rules per class was extracted. This rule-base was then optimised for parsimony as follows. Based on a sample of 10000 patterns generated from the unconditional density estimate, all rules which covered fewer than 100 patterns were removed from the rule-base. The greedy forward selection procedure was then used to reduce the rule-base to 50 rules, followed by greedy backward selection to reduce this further to 15 rules. Finally, the most effective sets of 6, 5, 4, 3 and 2 rules were found by searching over all combinations of these 15 rules. Note that this procedure places no preference on rules of any particular class. If it is more important to obtain rules for a particular class then the obvious approach is to optimise the parsimony of the rules for each class separately.

The simplex linear programming method described in [Chv80] was used to implement the various cell/hyperplane operations required by the algorithm.

6.3 Dataset 1: Heart Disease

The purpose of this dataset is to predict the presence or absence of heart disease given the results of various medical tests carried out on each patient. The dataset comes from the Cleveland Clinic Foundation and was originally supplied by Dr Detrano of the V.A. Medical Center, Long Beach, CA. The results from the tests were encoded as patterns with 6 categorical variables and 7 ordered variables, see table 6.1. There are a total of 270 patterns; 120 of these are from patients with heart disease, and 150 from patients without heart disease. Plotting every variable against every other (pairwise scatter plots) showed very little separation between the two classes; neither did it reveal any obvious outliers.

This data was previously used in the Statlog project [MST94] and may be downloaded from their web site. It was also used by [AG96a].

Categorical variable	Code	Values
sex	sex	female, male
chest pain type	chest	typical angina, atypical angina, non-anginal, asymptomatic
fasting blood sugar > 120 mg/dl	sugar	low, high
resting electrocardiographic results	ecg	normal, wave abnormal, hypertrophy
exercise induced angina	angina	no, yes
thal	thal	normal, fixed, reversible

Ordered variable	Code	Values
age	age	29 - 77
resting blood pressure in mmHg	bp	94 - 200
serum cholesterol in mg/dl	cholest	126 - 564
maximum heart rate achieved in bpm	heart	71 - 202
exercise induced ST depression relative to rest	st	0 - 6.2
the slope of the peak exercise ST segment	slope	upsloping(1), flat(2), downsloping(3)
number of major vessels coloured by fluoroscopy	vessel	0,1,2,3

Table 6.1: (Heart disease problem) Description of the variables in the dataset.

#hidden nodes	weight decay	score	# correct per class		
			absent/150	present/120	total/270
1	0.005	0.3798	135	96	231
1	0.001	0.3752	134	97	231
1	0.0005	0.3871	133	98	231
2	0.01	0.3911	134	96	230
2	0.005	0.3795	135	96	231
2	0.001	0.4387	134	91	225
3	0.01	0.3870	134	95	229
3	0.005	0.3785	134	95	229
3	0.001	0.4495	127	92	219
4	0.01	0.3851	134	94	228
4	0.005	0.3769	135	95	230
4	0.001	0.4656	130	95	225

Table 6.2: (Heart disease problem) 10-fold cross-validation results from training different feed-forward networks.

6.3.1 Training the feed-forward network

The categorical variables were encoded using the scheme given in section 2.3.4 on page 40. This maps each categorical variable with k categories into k vectors in $k - 1$ dimensions. Since each ordered variable is mapped to a single network input, it follows from table 6.1 that the MLP has a total of 17 input nodes. A single output was used to estimate the conditional probability of heart disease, $P(\text{heart disease} | \mathbf{x})$.

It was decided to use all the data to train the network because there were so few patterns relative to the number of input dimensions. Thus there was no independent test set on which to estimate generalisation performance.

Table 6.2 shows the results of applying the 10-fold cross-validation procedure described in section 6.2.1. The “score” column gives the negative logarithm conditional class probabilities, averaged both over patterns within each partition and over all 10 partitions. This provides a relative measure of how accurately the network has estimated the conditional class probabilities. The next 3 columns give the average number of correct classifications over the 10 sets of partitions.

The negative logarithm probability scores were used to determine the optimal number of hidden nodes and amount of weight decay. Thus a single final network with 1 hidden

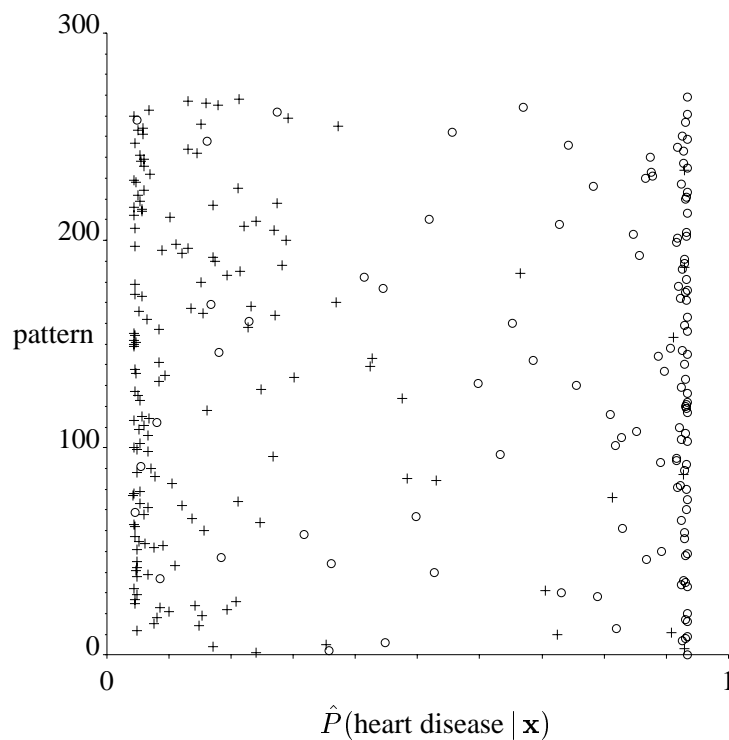


Figure 6.1: (Heart disease problem) Output from the final network. The circles/crosses are patterns labelled with/without heart disease.

node was selected and trained on all the data using a weight decay coefficient of 0.001. Figure 6.1 shows the output from this network evaluated on all the data. This plot gives an indication of the class separation achieved.

6.3.2 Extracting a rule-base

The MLP was linearised to a maximum error of $H = 0.005$. This produced a one dimensional division of hidden node space into 22 cells with a linearisation error of 0.00458. As expected, only one of these cells was a boundary cell (see figure 5.2 on page 126 for a reminder of the definition of a boundary cell). This completed the formation of the \mathcal{R}^1 representation.

To translate this into the \mathcal{R}^2 representation, comparative safety was set to 1.0 and one simplex (containing a single half-plane) was obtained for both classes. These two simplexes covered all but 3 patterns from a sample of 10000 patterns, thus demonstrating the

	Axis-aligned	Euclidean	City-block
Seed sample (10000)	30%	45%	96%
New sample (10000)	14%	32%	95%
Original data (270)	27%	51%	99%

Table 6.3: (Heart disease problem) Effectiveness of the three different rule-bases, each containing 1000 rules per class.

effect of a small but non-zero linearisation error. The geometric interpretation is that the two simplexes do not quite touch.

Using a sample of 10000 patterns, 1000 rules for each class were then generated to describe the space occupied by the two simplexes. This was repeated for each of the three types of rule. Table 6.3 gives the effectivenesses of these rule-bases on the sample used to seed the rules, a new sample of 10000 patterns, and the original data. The effectiveness on the sample used to seed the rules is greater than the effectiveness on an independent sample because there are rules which are so small that they are only likely to be activated by the pattern used to seed it. This may be considered to be a form of overfitting. The reason for the effectiveness being higher on the original data than on the new sample was suspected to be due to the difficulties in estimating density. In particular, the density estimate was probably rather “broad” in comparison with the true underlying density distribution.

Note that axis-aligned rules are the least effective, followed by euclidean, and that city-block rules are the most effective. This ordering is in agreement with the theoretical result shown in figure 3.11 on page 68. Due to the low effectiveness of axis-aligned and euclidean rules it was decided only to retain the city-block rule-base. This was optimised for parsimony using the method described in section 6.2.1. Table 6.4 gives the effectiveness of each of these rule-bases. Note that a new sample of 10000 patterns was used for each optimisation, and so the table shows the effectiveness both before and after each optimisation.

Placing a strong weight on maximising parsimony, the rule-base with only 2 rules was selected as the final explanation of the network. These 2 rules are shown in table 6.5, and their performance is shown in table 6.6. The latter table shows the effectiveness of the rule-base, the accuracy of the rule-base, the accuracy of the network, and the comparative

Number of rules		Effectiveness	
Before	After	Before	After
2000	1517	95%	95%
1517	50	95%	91%
50	15	91%	89%
15	6	89%	85%
15	5	89%	83%
15	4	89%	82%
15	3	88%	78%
15	2	89%	75%

Table 6.4: (Heart disease problem) Table showing the reduction in effectiveness (estimated on samples of 10000 patterns) as the parsimony of the city-block rule-base is increased.

accuracy. Naturally the comparative accuracy is equal to unity because the comparative safety was set to 1.0. Note that these results were computed using the 270 examples in the dataset, so the network and rule-base accuracy figures may contain significant bias.

Before discussing this explanation in detail, the following is a quick reminder of the format of a city-block rule (see section 3.1.3 on page 57). A city-block rule is activated if the sum of the contributions (one per variable) is less than unity, and the rule tableau simply indicates how each contribution (one per row) is calculated. For categorical variables the contribution simply depends upon the category, and so the rule simply lists the possible contributions. For ordered variables the contribution is the difference between the value of the variable and some prototype value, divided by one of two possible weighting coefficients. Which weighting coefficient is used depends upon the sign of the difference between the value and the prototype value. In principle the rule tableau gives two weights and a prototype value for each ordered variable, although for clarity an infinite weight is represented by a blank. Several numerical examples will be given in the following section.

6.3.3 The explanation

This section is divided into two parts: a basic interpretation of the two heart disease rules; followed by a demonstration of how these rules explain the classification of individual patterns.

```

Categories template
sex      : female   male
chest   : typical  atypical  non-angina  asymptomatic
sugar   : low      high
ecg     : normal   abnormal  hypertrophy
angina  : no       yes
thal    : normal   fixed     reversible

```

```

Rule 1: HEART DISEASE if
sex      : 0.134    0
chest   : 0.242    0.1637  0.202    0
sugar   : 0        0.00206
ecg     : 0        0        0
angina  : 0.0509   0
thal    : 0.1833   0.1904   0

age     :          17          +321.4
bp      :    -360.6    167
cholest :    -2091    419
heart   :          122          +344.5
st      :    -17.84    2.1
slope   :    -13.09    3
vessel  :    -6.807    2

```

```

Rule 2: NO HEART DISEASE if
sex      : 0        0.1413
chest   : 0.06059   0.1431   0        0.2131
sugar   : 0        0
ecg     : 0        0.02561   0.04988
angina  : 0        0.05368
thal    : 0        0        0.1933

age     :    -304.77    89
bp      :          98          +342.0
cholest :          314          +1983
heart   :    -326.64    194
st      :          0          +16.92
slope   :          1          +12.41
vessel  :          -1          +6.455

```

Table 6.5: (Heart disease problem) The final two city-block rules extracted from the feed-forward network. These rules are guaranteed to give the same classification as the network, and cover $245/270 \approx 91\%$ of the patterns in the dataset.

	No heart disease	Heart disease	Total
rule-base effectiveness	138/150 \approx 92%	107/120 \approx 89%	245/270 \approx 91%
rule-base accuracy	128/138 \approx 93%	94/107 \approx 88%	222/245 \approx 91%
network accuracy	139/150 \approx 70%	102/120 \approx 95%	241/270 \approx 89%
comparative accuracy	141/141 \approx 100%	104/104 \approx 100%	245/245 \approx 100%

Table 6.6: (Heart disease problem) The performance of the two city-block rules shown in table 6.5 on the 270 examples in the dataset.

Basic interpretation

A basic interpretation of the two rules may be obtained by examining the minimum and maximum contributions from each row of the rule. For categorical variables these contributions are immediately apparent from the *categories template*, which indicates which contribution corresponds to which category. Clearly the value of `sugar` is irrelevant to the activation of Rule 2 since either value contributes zero to the activation. In addition, the value of `sugar` is only slightly relevant to the activation of Rule 1. This suggests that the variable is unimportant. Similarly, the value of the `ecg` variable is irrelevant to the activation of Rule 1, and is also unimportant to the activation of Rule 2 because it makes no contribution when it is `normal` and only a very small contribution when `ecg` is not `normal`. Taking a third example, the effect of the gender of the patient on the rules is easily understood. The heart disease rule is more likely to be activated if the patient is male because then there is no contribution from the `SEX` row; similarly, the no heart disease rule is more likely to be activated if the patient is female.

Reading the contributions for ordered variables is only slightly more complicated. Consider for example blood pressure (`bp`) and Rule 1. If the blood pressure of the test patient is less than 167 mmHg then the contribution from this row is $\frac{167-bp}{360.6}$. Thus lower blood pressure gives rise to a larger contribution from this row, and therefore decreases the chance of this rule being activated. Note that if the blood pressure is greater than 167 mmHg then there is no contribution from this row because the right-most column is infinite (represented as a blank).

A basic interpretation of the conditions on the remaining ordered variables is as follows:

high blood pressure, high cholesterol, low maximum heart rate are all factors which reduce the total contribution of rule 1, and thus increase the possibility of assignment to the heart disease class. Perhaps curiously, the effects of age are counter-intuitive: older does not mean more likely to have heart disease. However two points must be remembered: firstly, this conclusion is based upon a single variable and thus ignores possible correlations between the variables; second, all these conclusions ultimately derive from a sample of only 270 patients and so may not be statistically significant. Of course it is only by extracting explanation that these kinds of effects can be detected.

The contributions from each row in each rule are shown in figure 6.2. These plots illustrate graphically which variables make significant contributions and hence may prevent the activation of a rule. Thus `sugar` is confirmed to be irrelevant, followed by `ecg` and `angina`. `cholesterol` mostly contributes very little, and this can also be seen directly from the rules because the divisors (-2091 in rule 1 and 1983 in rule 2) are relatively large. Note that these plots are useful for assessing the contributions of the ordered variables because they normalise out the effects of scale. Of particular interest are those variables which contribute a wide range of values.

A further simplification of the above approach is to consider the prototype pattern(s) for each rule. A prototype pattern may be defined as any pattern giving rise to the minimum contribution to the activation of the rule. In the case of ordered variables the coordinates are the middle column, and in the case of categorical variables the prototype categories are those which make the smallest contribution. Hence the prototype patterns for the two rules are shown in table 6.7. The crudest explanation of any new pattern is to say that it is most similar to one of these two prototype patterns and thus activates the corresponding rule.

Explaining individual patterns

Though constructive, the conclusions which can be drawn from analysing variables individually is limited. To gain an understanding of the interactions between variables it is

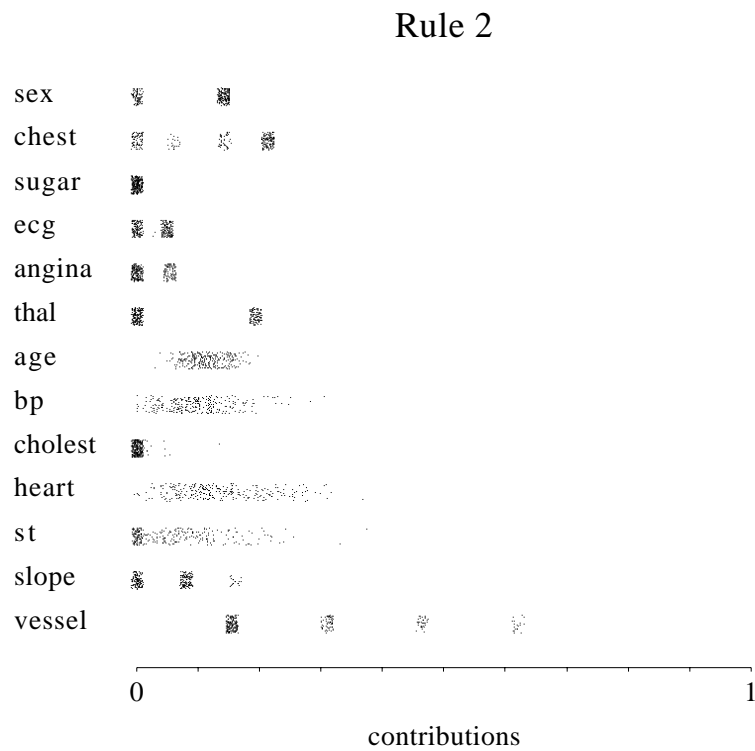
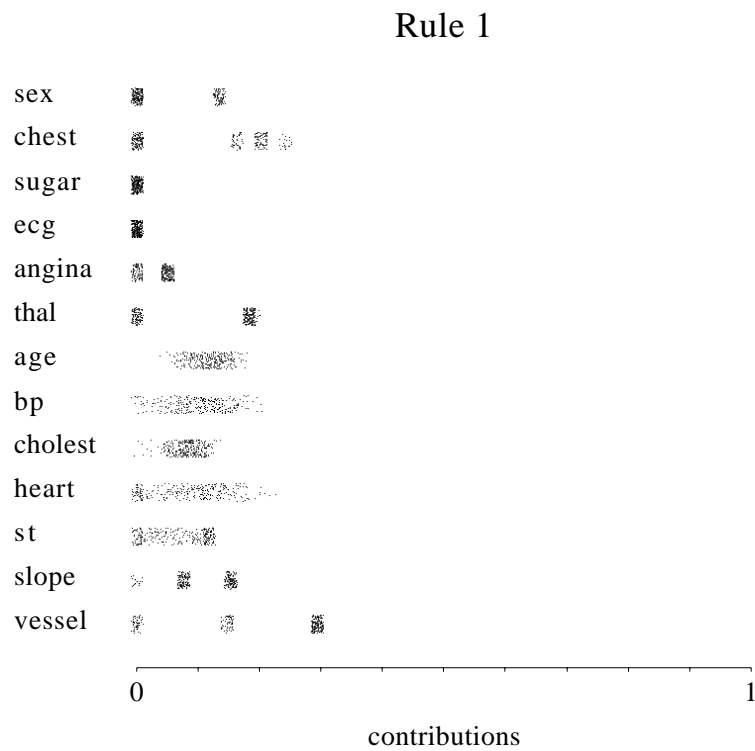


Figure 6.2: (Heart disease problem) Contributions from each variable in each rule over all 270 patterns in the dataset. Random jitter has been added both vertically and horizontally to separate co-incident points.

Rule 1: HEART DISEASE prototype pattern

sex	:	male
chest	:	asymptomatic
sugar	:	low
ecg	:	normal, abnormal or hypertrophy (ie any)
angina	:	yes
thal	:	reversible
age	:	17 and below
bp	:	167 and above
cholest	:	419 and above
heart	:	122 and below
st	:	2.1
slope	:	3
vessel	:	2 or 3

Rule 2: NO HEART DISEASE prototype pattern

sex	:	female
chest	:	non-angina
sugar	:	low or high (ie any)
ecg	:	normal
angina	:	no
thal	:	normal or fixed
age	:	89 and above
bp	:	98 and below
cholest	:	314 and below
heart	:	194 and above
st	:	0
slope	:	1
vessel	:	-1

Table 6.7: (Heart disease problem) Prototype patterns for the two rules shown in table 6.5.

useful to look at the explained classification of individual patterns. Two such explanations are shown in tables 6.8 and 6.9. The first shows why a particular pattern should be classified as corresponding to a patient with heart disease. In addition, this explanation indicates the large number of similar patterns which would also be classified as the heart disease class. For example, this pattern would be given the same classification even if the CHEST variable was TYPICAL. Similarly, even if the maximum achieved heart rate was increased to 180 beats per minute and the blood pressure reduced to 80 mmHg, then this pattern would still be classified as the heart disease class.

The second explanation, shown in table 6.9, justifies assigning a different pattern to the no heart disease class. Exactly the same kind of reasoning applies, and the effect on the classification from making changes to the input pattern is immediately apparent.

6.3.4 The effects of decreasing comparative safety

The effectiveness of the final two rules is relatively high (covering 91% of the data), but it is constructive to see if this figure can be increased by decreasing comparative safety. Recall from figure 3.16 on page 81 that reducing comparative safety increases the width of a “dead-band”, within which patterns may be assigned to either class. Clearly it is not possible to produce a similar plot for problems in more than two dimensions, however the probability plot shown in figure 6.1 may be used to view the distribution of the classes for problems of any number of dimensions. This plot was therefore used to decide that an acceptable comparative safety would be 0.43 because it produces a dead-band between $\hat{P}(\text{heart disease} | \mathbf{x}) = 0.3$ and $\hat{P}(\text{heart disease} | \mathbf{x}) = 0.7$.

Thus the rule extraction procedure was repeated with comparative safety set to 0.43, and this produced the results shown in table 6.10. The effectiveness of these rules is only marginally higher than the effectiveness of rules extracted with a comparative safety of unity (see table 6.3). Certainly the effectiveness of the axis-aligned and euclidean rules are still very low. Optimising the city-block rule-base for parsimony produced two rules with the performance shown in table 6.11. Thus reducing comparative safety from 1.0

Categories template

```
sex      : female   male
chest    : typical  atypical  non-angina  asymptomatic
sugar    : low      high
ecg      : normal   abnormal  hypertrophy
angina   : no       yes
thal     : normal   fixed     reversible
```

Rule 1: HEART DISEASE ** activated **

	pattern	contrbn	condition		
sex	: male	0	0.134	0	
chest	: asymptomatic	0	0.242	0.1637	0.202 0
sugar	: low	0	0	0.002056	
ecg	: hypertrophy	0	0	0	0
angina	: no	0.0509	0.0509	0	
thal	: normal	0.1833	0.1833	0.1904	0
age	: 70	0.1649		17	+321.4
bp	: 130	0.1026	-360.6	167	
cholest	: 322	0.0464	-2091	419	
heart	: 109	0		122	+344.5
st	: 2.4	0	-17.84	2.1	
slope	: 2	0.0764	-13.09	3	
vessel	: 3	0	-6.808	2	

TOTAL CONTRIBUTION = 0.625

Rule 2: NO HEART DISEASE ** not activated **

	pattern	contrbn	condition		
sex	: male	0.1413	0	0.1413	
chest	: asymptomatic	0.2131	0.06059	0.1431	0 0.2131
sugar	: low	0	0	0	
ecg	: hypertrophy	0.0499	0	0.02561	0.04988
angina	: no	0	0	0.05368	
thal	: normal	0	0	0	0.1933
age	: 70	0.0623	-304.8	89	
bp	: 130	0.0936		98	+342.0
cholest	: 322	0.0040		314	+1983
heart	: 109	0.2602	-326.6	194	
st	: 2.4	0.1418		0	+16.92
slope	: 2	0.0806		1	+12.41
vessel	: 3	0.6196		-1	+6.455

TOTAL CONTRIBUTION = 1.67

Table 6.8: (Heart disease problem) Explanation of why a given pattern is assigned to the heart disease class.

Categories template

```
sex      : female   male
chest    : typical  atypical  non-angina  asymptomatic
sugar    : low      high
ecg      : normal   abnormal  hypertrophy
angina   : no       yes
thal     : normal   fixed     reversible
```

Rule 1: HEART DISEASE ** not activated **

	pattern	contrbn	condition			
sex	: male	0	0.134	0		
chest	: non-angina	0.2020	0.242	0.1637	0.202	0
sugar	: low	0	0	0.002056		
ecg	: hypertrophy	0	0	0	0	
angina	: no	0.0509	0.0509	0		
thal	: normal	0.1833	0.1833	0.1904	0	
age	: 44	0.0840		17	+321.4	
bp	: 140	0.0749	-360.6	167		
cholest	: 235	0.0880	-2091	419		
heart	: 180	0.1684		122	+344.5	
st	: 0	0.1177	-17.84	2.1		
slope	: 1	0.1528	-13.09	3		
vessel	: 0	0.2938	-6.808	2		

TOTAL CONTRIBUTION = 1.42

Rule 2: NO HEART DISEASE ** activated **

	pattern	contrbn	condition			
sex	: male	0.1413	0	0.1413		
chest	: non-angina	0	0.06059	0.1431	0	0.2131
sugar	: low	0	0	0		
ecg	: hypertrophy	0.0499	0	0.02561	0.04988	
angina	: no	0	0	0.05368		
thal	: normal	0	0	0	0.1933	
age	: 44	0.1477	-304.77	89		
bp	: 140	0.1228		98	+342.0	
cholest	: 235	0		314	+1983	
heart	: 180	0.0429	-326.64	194		
st	: 0	0		0	+16.92	
slope	: 1	0		1	+12.41	
vessel	: 0	0.1549		-1	+6.455	

TOTAL CONTRIBUTION = 0.659

Table 6.9: (Heart disease problem) Explanation of why a given pattern is assigned to the no heart disease class.

	Axis-aligned	Euclidean	City-block
New sample (10000)	17%	39%	100%
Original data (270)	31%	57%	100%

Table 6.10: (Heart disease problem) Effectiveness of the three different rule-bases, each containing 1000 rules per class, extracted with a comparative safety of 0.43.

	No heart disease	Heart disease	Total
rule-base effectiveness	145/150 \approx 97%	116/120 \approx 97%	261/270 \approx 97%
rule-base accuracy	134/145 \approx 92%	98/116 \approx 84%	232/261 \approx 89%
network accuracy	139/150 \approx 70%	102/120 \approx 95%	241/270 \approx 89%
comparative accuracy	150/151 \approx 99%	108/110 \approx 98%	258/261 \approx 99%

Table 6.11: (Heart disease problem) The performance of two city-block rules extracted with comparative safety of 0.43, estimated using the 270 examples in the heart disease dataset.

to 0.43 has increased the effectiveness of two rules from 91% to 97%. Naturally this means that comparative accuracy is also reduced, albeit by a very small amount. This was perhaps to be expected since there are only a small number of patterns in the 0.3 to 0.7 dead-band. Note that as before, the network and rule-base accuracies are biased because they are estimated on the same data as was used to train the original network.

6.4 Dataset 2: Pima Indian

The purpose of this dataset is to predict the presence or absence of diabetes in women of Pima Indian descent living near Phoenix, Arizona, USA. The data was collected by the US National Institute of Diabetes and Digestive and Kidney Diseases, and the subjects were tested for diabetes according to World Health Organization criteria. The results of various physiological measurements and medical tests on each subject were encoded as a pattern consisting of 8 ordered variables. There was originally 768 observations of these 8 variables, however many of the patterns contained zero values which are physically impossible. Following [Rip96], these patterns and the serum insulin variable were removed from the data. This left a total of 532 patterns with 7 variables (see table 6.12). 355 of these patterns were from subjects without diabetes, and 177 from subjects with

Ordered variable	Code	Values
number of pregnancies	npreg	0 - 17
plasma glucose concentration	glu	56 - 199
diastolic blood pressure (mm Hg)	bp	24 - 110
triceps skin fold thickness (mm)	skin	7 - 99
body mass index (kg/m ²)	bmi	18.2 - 67.1
diabetes pedigree function	ped	0.085 - 2.42
age	age	21 - 81

Table 6.12: (Pima Indian problem) Description of the variables in the dataset.

diabetes. Plotting every variable against every other (pairwise scatter plots) showed very little separation between the two classes; neither did it reveal any obvious outliers.

The dataset has been used previously by both [Rip96] and [Tar98]. It may be downloaded from the UCI machine-learning database collection [MA95], or from Proben1 [Pre94], or alternatively from [Rip96]. The latter was used here because it was already split into training and test partitions of 200 and 332 patterns respectively.

6.4.1 Training the feedforward network

An MLP network with the 7 ordered variables as inputs and a single output to estimate the conditional probability of diabetes, $P(\text{diabetes} \mid \mathbf{x})$, was trained as before.

The 10-fold cross-validation method described in section 6.2.1 was used with the 200 training patterns in order to optimise the number of hidden nodes and weight decay coefficient. Table 6.13 summarises the results of these experiments.

Figure 6.3 shows a graph of the minimum scores obtained by networks of increasing number of hidden nodes. According to this plot, the optimal network has at least 20 hidden nodes. However the benefits of choosing the best network with 20 hidden nodes over the best network with 5 hidden nodes is marginal, as can be seen by considering that 75% of the decrease in the value of the score occurs when the number of hidden nodes is increased from 1 to 5. However there is also a pragmatic motive for preferring a network with a relatively small number of hidden nodes. The amount of computer memory and execution time required by the rule extraction algorithm increases with the number of

#hidden nodes	weight decay	score	# correct per class		total/200
			non/132	diabetic/68	
1	0.05	0.5908	132	0	132
1	0.01	0.5151	112	35	147
1	0.005	0.5129	109	38	147
1	0.001	0.5291	106	39	145
2	0.05	0.5729	131	2	133
2	0.01	0.5047	112	36	148
2	0.005	0.5098	111	36	147
2	0.001	0.5762	110	32	142
3	0.05	0.5640	130	9	139
3	0.01	0.5005	112	36	148
3	0.005	0.5017	109	38	147
3	0.001	0.6515	107	32	139
4	0.05	0.5587	130	12	142
4	0.01	0.4967	113	35	148
4	0.005	0.4966	111	37	148
4	0.001	0.6768	109	34	143
5	0.05	0.5552	130	14	144
5	0.01	0.4946	113	35	148
5	0.005	0.4968	110	37	147
5	0.001	0.6685	106	31	137
6	0.05	0.5526	130	15	145
6	0.01	0.4931	114	36	150
6	0.005	0.4963	111	37	148
6	0.001	0.7096	101	35	136
7	0.05	0.5508	130	16	146
7	0.01	0.4920	114	36	150
7	0.005	0.4953	111	37	148
7	0.001	0.7498	106	34	140
8	0.05	0.5493	130	16	146
8	0.01	0.4911	114	36	150
8	0.005	0.4960	111	36	147
8	0.001	0.7079	103	35	138
9	0.05	0.5481	130	17	147
9	0.01	0.4904	114	36	150
9	0.005	0.4948	111	37	148
9	0.001	0.7977	102	31	133
10	0.05	0.5471	130	17	147
10	0.01	0.4899	114	35	149
10	0.005	0.4949	111	37	148
10	0.001	0.7613	105	34	139
15	0.05	0.5441	129	19	148
15	0.01	0.4884	114	35	149
15	0.005	0.4942	111	37	148
15	0.001	0.7086	108	36	144

Table 6.13: (Pima Indian problem) 10-fold cross-validation results of training different feedforward networks.

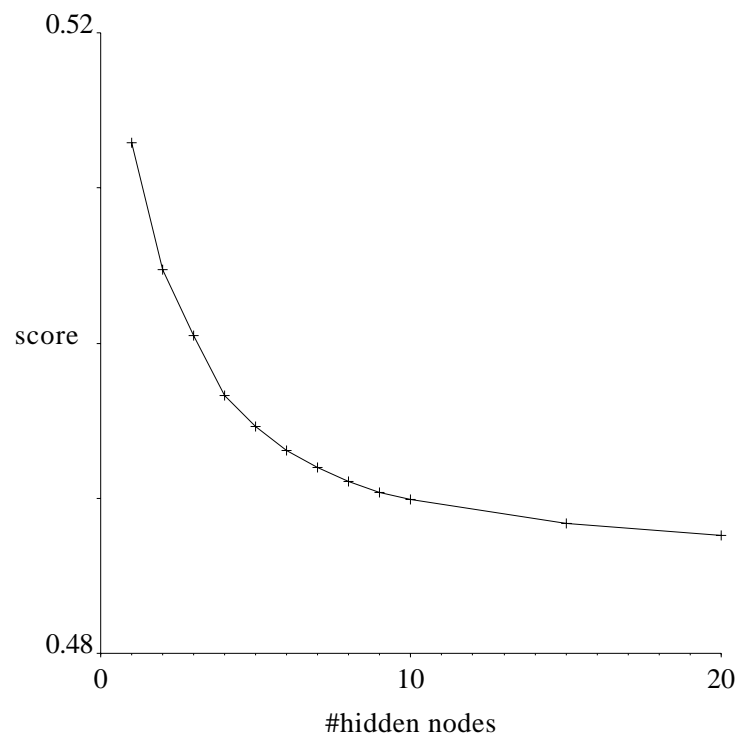


Figure 6.3: (Pima Indian problem) A graph showing how the minimum cross-validation score decreases with increasing number of hidden nodes.

hidden nodes. Thus the choice of a near-optimal network with fewer hidden nodes is a reasonable compromise.

The final network was obtained by training a 5 hidden node MLP on the 200 training patterns using a weight decay coefficient of 0.01. Out of the 332 test patterns, 201 out of the 223 non-diabetic patterns ($\approx 90\%$) were classified correctly, and 66 out of the 109 diabetic patterns ($\approx 61\%$) were classified correctly. This is a total of 267/332, so the final classification performance of the network was estimated to be $\approx 80\%$. Figure 6.4 shows a plot of the output of the network for each pattern in the test set. This indicates the amount of overlap between the two classes.

In previous attempts with this dataset, [Rip96] found that no MLP of any number of hidden nodes had a classification error rate less than that obtained using logistic regression, and [Tar98] found that 6 hidden nodes were optimal. However the latter used a different partitioning of the dataset. Despite the differences in the number of hidden nodes used by

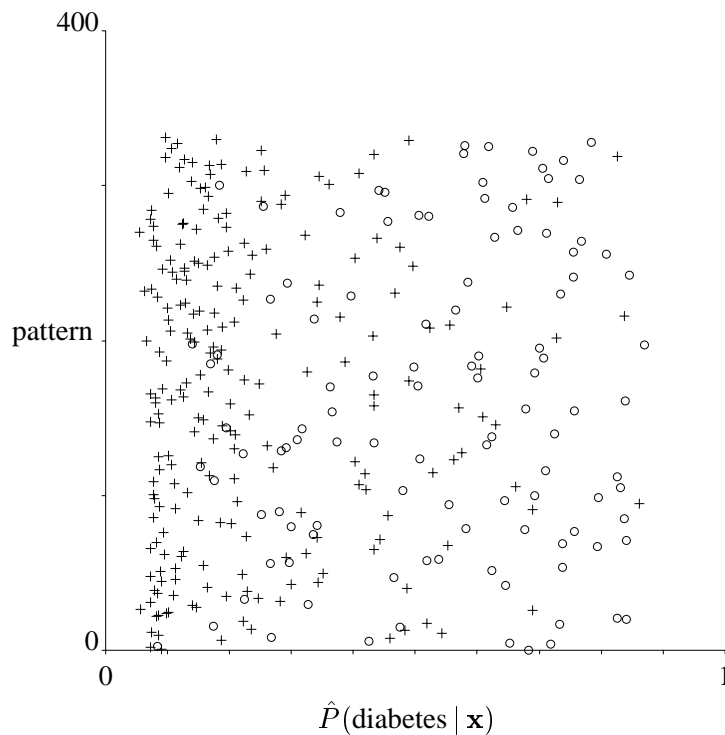


Figure 6.4: (Pima Indian problem) Output from the final network. The circles/crosses are patterns from the test set labelled with/without diabetes.

[Rip96], [Tar98], and here, the test error rates are almost identical, being 19.8%, 22.7%, and 19.6% respectively.

It is interesting to note that the weights associated with each hidden node in the final network were almost identical. Table 6.14 shows the weights and offsets. The similarity between the values suggests that there is only a slight curve in the classification boundary (if the weights associated with each hidden node were exactly the same then the classification boundary would be a hyperplane).

6.4.2 Extracting a rule-base

The MLP was linearised to a maximum error of $H = 0.05$. This divided each hidden node sigmoid into 7 segments with a linearisation error of 0.0282. The total number of cells was $7^5 = 16807$, but only 5806 of these were found to be boundary cells. These boundary cells form the \mathcal{R}^1 representation. Comparative safety was then set to 1.0, and the algorithm to

Input weights		Input offsets					
-.066193	-.0195000	-0.0422054	-0.003722342	-0.4452875	-1.0092886	-0.0269058	6.01921926
-.066201	-.0194990	-0.0421619	-0.003722757	-0.4452056	-1.0094054	-0.0269042	6.01876114
-.066199	-.0194989	-0.0421567	-0.003723655	-0.4452385	-1.0094235	-0.0269026	6.01879478
-.066195	-.0195002	-0.0422257	-0.003720833	-0.4452526	-1.0092286	-0.0269086	6.01925470
-.066203	-.0194996	-0.0422025	-0.003720169	-0.4451626	-1.0092859	-0.0269092	6.01891021

Output weights		Output offset			
-1.04758128	-1.04751708	-1.04753306	-1.04757258	-1.04751323	2.00389063

Table 6.14: (Pima Indian problem) Weights and offsets in the final MLP. The similarity between the values suggests that the function is gently curved.

	Axis-aligned	Euclidean	City-block
Seed sample (10000)	59%	72%	95%
New sample (10000)	52%	69%	94%
Test data (332)	55%	70%	94%

Table 6.15: (Pima Indian problem) Effectivenesses of 2000 rules of each of the three rule types.

Number of rules		Effectiveness	
Before	After	Before	After
2000	1759	94%	94%
1759	50	95%	95%
50	15	94%	94%
15	6	94%	93%
15	5	94%	92%
15	4	93%	91%
15	3	94%	90%
15	2	94%	87%

Table 6.16: (Pima Indian problem) Optimising the parsimony of the city-block rule-base. The table shows the effectiveness on a sample of 10000 patterns both before and after each optimisation.

form the \mathcal{R}^2 representations was iterated 200 times to obtain simplexes describing both the diabetes and non-diabetes regions. Only 1101 out of the possible 5806 boundary cells were used, and the simplexes were found to cover 9690 patterns from a sample of 10000. There are two reasons for the incomplete coverage: non-zero linearisation error; and the fact that the two \mathcal{R}^2 representations did not use all the boundary cells. The first of these is a limitation of the linearisation method (chapter 4), and the second is a drawback of the \mathcal{R}^1 to \mathcal{R}^2 conversion algorithm (section 5.3).

Using a sample of 10000 patterns, 1000 rules of each type and for each class were generated to form three rule-bases. Table 6.15 gives the effectiveness of these rule-bases on the sample used to seed the rules, a new sample of 10000 patterns, and the test data. As with the heart disease dataset, the city-block rules are much more effective than either euclidean or axis-aligned rules. Hence only the city-block rule-base was further optimised for parsimony. This was achieved using the same method as before (described in 6.2.1). Table 6.16 gives the effectiveness of each increasingly parsimonious rule-base.

Placing a strong weight on maximising parsimony, the rule-base with only 2 rules was

Rule 1: DIABETES if			
npreg	-27.617	10	
glu	-93.757	176	
bp	-433.36	86	
skin	-491.18	23	
bmi	-41.062	35.9	
ped	-1.8113	0.883	
age	-67.949	50	
Rule 2: NO DIABETES if			
npreg		0	+39.166
glu		82	+132.96
bp		69	+614.63
skin		21	+696.5
bmi		33.4	+58.232
ped		-0.127	+2.5687
age		17	+96.365

Table 6.17: (Pima Indian problem) The final two city-block rules extracted from the feed-forward network. They are guaranteed to give the same classification as the network and cover approximately 86% of the test patterns (see table 6.18).

	Not diabetic	Diabetic	Total
rule-base effectiveness	200/223 \approx 90%	87/109 \approx 80%	287/332 \approx 86%
rule-base accuracy	189/200 \approx 95%	51/87 \approx 59%	240/287 \approx 84%
network accuracy	201/223 \approx 90%	66/109 \approx 61%	267/332 \approx 80%
comparative accuracy	225/225 \approx 100%	62/62 \approx 100%	287/287 \approx 100%

Table 6.18: (Pima Indian problem) The performance of the two city-block rules shown in table 6.17 on the 332 examples in the test set.

selected as the final explanation of the network. These 2 rules are shown in table 6.17, and their performance is shown in table 6.18. The latter table shows the effectiveness of the rule-base, the accuracy of the rule-base, the accuracy of the network, and the comparative accuracy, all computed using the 332 examples in the test set. Note that the accuracy of the rule-base is higher than the accuracy of the feed-forward network because the rules do not classify every test pattern. Note also that the comparative accuracy is necessarily 100% because the comparative safety was set to 1.0.

Comment on prototype patterns

Although the prototype patterns in the city-block rules belong to a very definite class, they may not be very likely to occur. There is a simple geometric reason for this property: the further the rule prototype pattern is from the class boundary, the more effective the rule antecedent region will be. This explains certain characteristics of the prototypes in the results given above. For example, the prototype value of `ped` in the second rule shown in table 6.17 is negative, even though the value of `ped` is always positive in all of the available data (see table 6.12). Similarly, the prototype `age` is 17, compared with the smallest value in the dataset of 21. Thus this prototype “no diabetes” pattern is on the edge of normality. Similar examples can be found in the heart disease problem (the prototype value of `vessel` in rule 2 of table 6.5 is -1, compared with a minimum value of 0 found in the dataset), and in the next problem to be considered. In short, the prototype patterns have a large conditional class probability, but tend to occur in regions of low unconditional density.

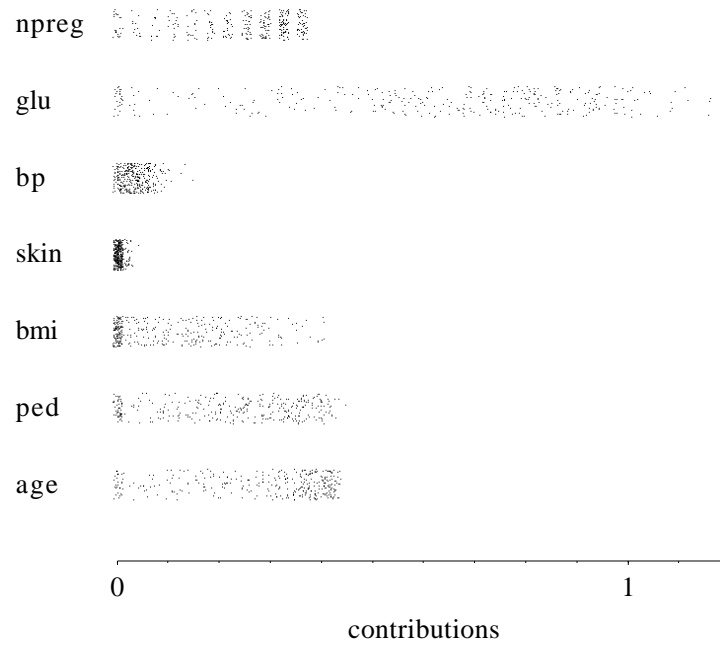
6.4.3 The explanation

The final two rules are intuitively satisfying. The total contribution for the diabetes rule will be smaller for subjects with a large number of pregnancies, large glucose concentration, high blood pressure, large skin fold thickness, high body mass index, high diabetes pedigree function, and of a more mature age. The converse is true of the no diabetes rule.

The contributions plot of the two rules are shown in figure 6.5. These suggest that neither blood pressure nor skin fold thickness are critical in determining diabetes. They also suggest that glucose is an important variable because it has a large spread of contributions.

Finally, an explanation of why a particular pattern should be classified as belonging to the diabetes class is given in table 6.19. This shows the contribution arising from each variable, and how the pattern could change such that the same or the opposite classification would result.

Rule 1



Rule 2

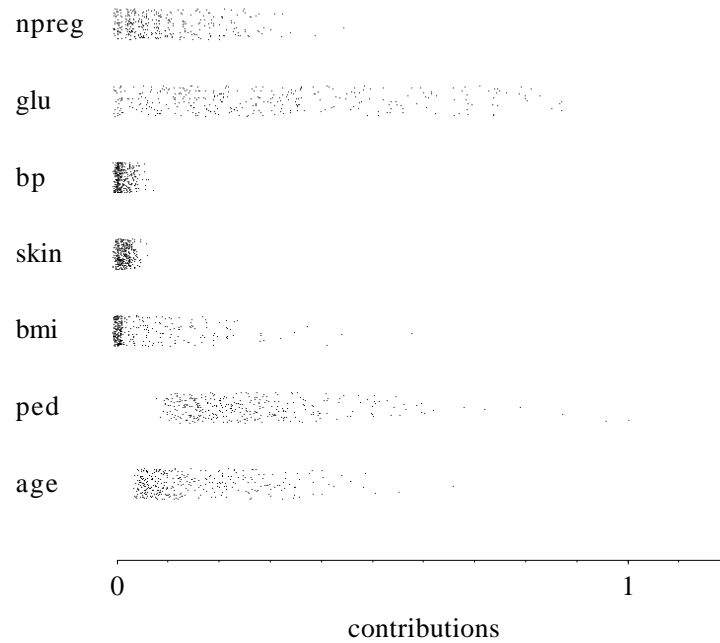


Figure 6.5: (Pima Indian problem) Contributions from each variable in each rule over all 332 patterns in the test set. Random jitter has been added both vertically and horizontally to separate co-incident points.

Rule 1: DIABETES if ** activated **				
	pattern	contribution	condition	
npreg	6	0.1448	-27.617	10
glu	148	0.2986	-93.757	176
bp	72	0.0323	-433.36	86
skin	35	0	-491.18	23
bmi	33.6	0.0560	-41.062	35.9
ped	0.627	0.1413	-1.8113	0.883
age	50	0	-67.949	50
TOTAL CONTRIBUTION = 0.6731				
Rule 2: NO DIABETES ** not activated **				
	pattern	contribution	condition	
npreg	6	0.1532		0 +39.166
glu	148	0.4964		82 +132.96
bp	72	0.0049		69 +614.63
skin	35	0.0201		21 +696.5
bmi	33.6	0.0034		33.4 +58.232
ped	0.627	0.2935		-0.127 +2.5687
age	50	0.3424		17 +96.365
TOTAL CONTRIBUTION = 1.3140				

Table 6.19: (Pima Indian problem) Explanation of why a given pattern is assigned to the diabetes class.

	Axis-aligned	Euclidean	City-block
New sample (10000)	67%	85%	100%
Original data (270)	68%	83%	100%

Table 6.20: (Pima Indian problem) Effectiveness of the three different rule-bases, each containing 1000 rules per class, extracted with a comparative safety of 0.67.

6.4.4 The effects of decreasing comparative safety

In an attempt to obtain more effective rules, the extraction procedure was repeated with a comparative safety less than unity. Figure 6.4 was used to decide by how much comparative safety could be acceptably reduced. Notice how the distribution of classes is different from those of the heart disease dataset, shown in figure 6.1. In particular, there is more overlap between the classes and also fewer patterns at the extremes with probabilities close to 0 or 1. Hence it was decided to use a narrower dead-band of 0.4 to 0.6. This is achieved (see figure 3.16 on page 81) by a comparative safety of 0.67.

The results are shown in table 6.20. These compare favourably with the previous results obtained using unity comparative safety (table 6.15). The city-block rulebase was further optimised for parsimony, and the performance of the resulting two city-block rules is shown in table 6.21. Compared with the original two city-block rules (table 6.18), the effectiveness has increased from 86% to 93%, the accuracy has dropped from 84% to 80%, and the comparative accuracy has dropped from 100% to 96%.

The same procedure was also used to obtain the most effective two axis-aligned and euclidean rules with a comparative safety of 0.67. The two axis-aligned rules were found to have an effectiveness of 33% and the two euclidean rules an effectiveness of 58%. Thus on this dataset, reducing comparative safety does not make these type of rules viable alternatives to city-block rules.

6.5 Dataset 3: Breast Cancer

The purpose of this dataset is to predict whether a tissue sample taken from a patient's breast is malignant or benign. The data was collected at the University of Wisconsin

	No heart disease	Heart disease	Total
rule-base effectiveness	214/223 \approx 96%	95/109 \approx 87%	309/332 \approx 93%
rule-base accuracy	203/214 \approx 95%	45/95 \approx 47%	248/309 \approx 80%
network accuracy	201/223 \approx 90%	66/109 \approx 61%	267/332 \approx 80%
comparative accuracy	242/242 \approx 100%	56/67 \approx 84%	298/309 \approx 96%

Table 6.21: (Pima Indian problem) The performance of two city-block rules extracted with comparative safety of 0.67, estimated using the 322 examples in the test set.

Ordered variable	Code	Values
Clump Thickness	thickness	1 - 10
Uniformity of Cell Size	size	1 - 10
Uniformity of Cell Shape	shape	1 - 10
Marginal Adhesion	adhesion	1 - 10
Single Epithelial Cell Size	epithelial	1 - 10
Bare Nuclei	bare	1 - 10
Bland Chromatin	chromatin	1 - 10
Normal Nucleoli	normal	1 - 10
Mitosis	mitosis	1 - 10

Table 6.22: (Breast cancer problem) Description of the variables in the dataset.

Hospitals, Madison. Each pattern consists of the results from 9 visual assessments of nuclear features of fine needle aspirates, see table 6.22. Each assessment is scored with a number in the interval 1 to 10, with value 1 referring to a normal state and 10 to a most abnormal state. Malignancy is determined by taking a tissue sample from the patient's breast and performing a biopsy on it. A benign diagnosis is confirmed by biopsy or by periodic examination, depending on the patient's choice. There are a total of 699 patterns; 458 of these correspond to benign tissue samples and 241 to malignant tissue samples. In 16 patterns, the results from the bare nuclei test were missing. The simple strategy of substituting the mean (evaluated over the remaining 683 patterns) was used to eliminate these missing values.

An early subset of the data was analysed using the FACT classifier, [WTLV87], [WTL88], [WTL89], with linear programming methods, [MW90], and using a multisurface method of pattern separation, [WM90]. In addition this data formed one of the datasets in an empirical comparison of decision trees and other classification methods, [LLS97], and

is also found in the Proben1 database, [Pre94]. The data may be obtained electronically from [Pre94] or the UCI machine-learning database collection, [MA95].

Three of the rule extraction papers reviewed in chapter 1 used this dataset: Taha and Ghosh (page 8) produced 5 rules; Bologna (page 10) produced 8 rules; and Goodman *et al* (page 17) produced 11 rules. All three had approximately the same accuracy as a standard MLP ($\approx 96\%$). Note that the rules produced by Taha and Ghosh classify all patterns in the dataset but are not guaranteed to give the same classification as the network, and that both Bologna and Goodman *et al* used non-standard networks.

6.5.1 Training the feed-forward network

An MLP with the 9 ordered variables as inputs and a single output to estimate the conditional probability of cancer, $P(\text{cancer} \mid \mathbf{x})$, was trained as before.

The 10-fold cross-validation method to select network architecture was used on 690 of the 699 patterns (9 of the patterns were not used during cross-validation). Table 6.23 summarises the results of these experiments. Figure 6.6 shows a graph of the minimum scores obtained by networks with increasing numbers of hidden nodes. According to this plot, approximately 75% of the total decrease (over the range tested) occurs when the number of hidden nodes is increased from 1 to 5.

The final network was obtained by training a 5-hidden node MLP on all the data using a weight decay coefficient of 0.0005. Figure 6.7 shows a plot of the output of the network for each pattern, thus indicating the distribution of class probabilities and the amount of class separation achieved. The weights associated with each hidden node in the network are shown in table 6.24. Note that the weights are all different, which suggests that the function is different from that which would be produced by a network with only a single hidden node.

#hidden nodes	weight decay	score	# correct per class		total/690
			non/438	cancer/237	
1	0.0005	0.1121	437	227	664
1	0.001	0.1116	438	228	666
1	0.005	0.1339	440	227	667
2	0.0005	0.0964	437	227	664
2	0.001	0.0954	440	227	667
2	0.005	0.1157	440	227	667
3	0.0005	0.1001	437	227	664
3	0.001	0.0960	439	226	665
3	0.005	0.1088	440	227	667
4	0.0005	0.0975	439	225	664
4	0.001	0.0951	441	226	667
4	0.005	0.1051	441	227	668
5	0.0001	0.1609	436	217	653
5	0.0005	0.0907	439	227	666
5	0.001	0.0938	441	226	667
6	0.0001	0.1644	438	214	652
6	0.0005	0.0883	438	227	665
6	0.001	0.0935	441	226	667
7	0.0001	0.1355	439	216	655
7	0.0005	0.0865	439	226	665
7	0.001	0.0926	441	226	667
8	0.0001	0.1454	440	216	656
8	0.0005	0.0867	438	227	665
8	0.001	0.0925	441	226	667
9	0.0001	0.1497	442	219	661
9	0.0005	0.0858	438	227	665
9	0.001	0.0924	441	226	667
10	0.0001	0.1369	440	219	659
10	0.0005	0.0852	438	227	665
10	0.001	0.0928	441	226	667
15	0.0001	0.1312	441	219	660
15	0.0005	0.0836	438	226	664
15	0.001	0.0924	441	226	667

Table 6.23: (Breast cancer problem) 10-fold cross-validation results of training different feed-forward networks.

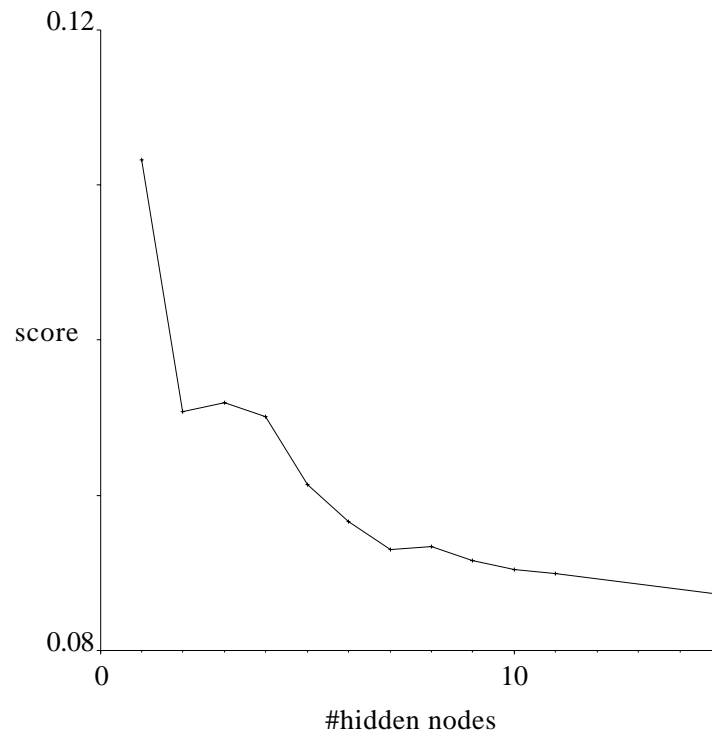


Figure 6.6: (Breast cancer problem) A graph showing how the minimum cross-validation score decreases with increasing number of hidden nodes.

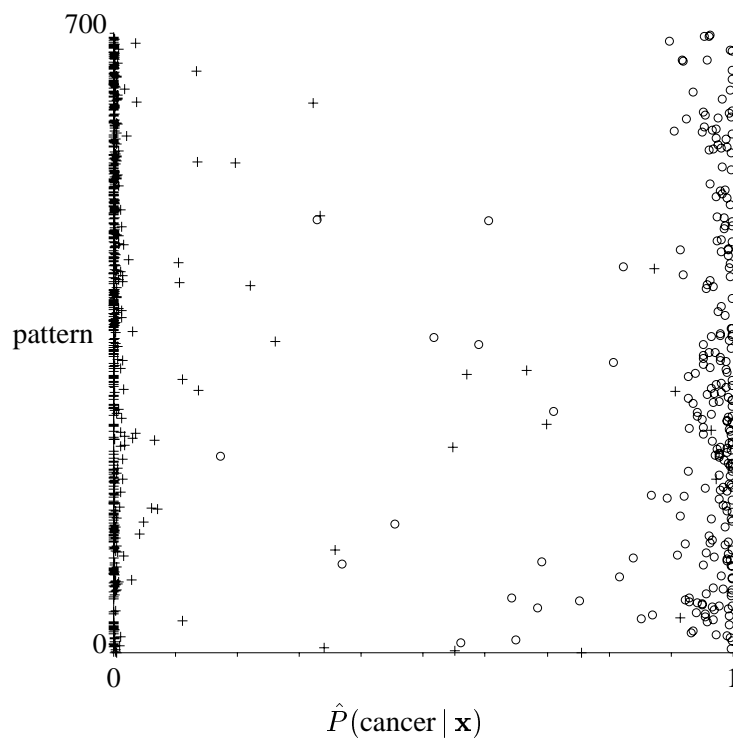


Figure 6.7: (Breast cancer problem) Output from the final network. The circles/crosses are patterns labelled with/without cancer.

Input weights											Input offsets	
-0.3725	-0.1983	-0.3065	-0.3011	-0.07477	-0.3736	-0.02311	-0.1305	-0.1389	5.050		-0.1389	
-0.2579	-0.3288	-0.4353	-0.08009	-0.1009	-0.2339	-0.06337	-0.1537	-0.07568	4.633		-0.07568	
-0.1973	-0.2423	-0.3992	-0.3236	-0.2385	0.1463	-0.1055	0.2184	-0.5143	7.914		-0.5143	
-0.3600	0.3342	0.2832	0.2882	0.6734	-0.6489	-0.5063	-0.4555	-0.9057	3.668		-0.9057	
-0.3921	0.1067	0.02008	-0.5751	-0.2079	-0.3165	-0.2352	-0.1151	-0.2494	5.305		-0.2494	

Output weights		Output offset			
-2.604	-2.418	-3.398	-3.626	-2.715	6.6600

Table 6.24: (Breast cancer problem) Weights and offsets in the final MLP. The difference between the values suggests that the function is significantly different from the function produced by a network with a single hidden node.

	Axis-aligned	Euclidean	City-block
Seed sample (10000)	65%	82%	92%
New sample (10000)	56%	78%	92%
Original data (699)	67%	82%	93%

Table 6.25: (Breast cancer problem) Effectivenesses of 2000 rules of each of the three rule types, estimated on the seed sample, a new sample, and the original data.

6.5.2 Extracting a rule-base

The MLP was linearised to a maximum error of $H = 0.1$. This divided the hidden node sigmoids into 7, 7, 10, 10, 8 segments, producing a linearisation error of 0.0815. The total number of cells was 39200, but only 10580 of these were found to be boundary cells (the \mathcal{R}^1 representation). Comparative safety was set to 1.0, and the algorithm to form the \mathcal{R}^2 representations (see section 5.3) was iterated 100 times to obtain simplexes describing both the cancer and non-cancer regions. Only 5294 out of the possible 10580 boundary cells were used, and the simplexes were found to cover 9866 patterns from a sample of 10000.

Rule-bases of each type, each containing 1000 rules per class were then generated using samples of 10000 patterns. Table 6.25 gives the effectiveness of these rule-bases on three different samples: the sample used to seed the rules, a new sample of 10000 patterns, and the original data.

As with the heart disease and Pima Indian datasets, the city-block rules are much more effective than either euclidean or axis-aligned rules. Hence only the city-block rule-base was further optimised for parsimony. This was achieved with the same method as used on the previous two datasets. Table 6.26 gives the effectiveness of each intermediate rule-base.

The five most effective city-block rules are shown in table 6.27, and their performance is shown in table 6.28. In comparison, the two most effective city-block rules are shown in table 6.29, and their performance is shown in table 6.30. Notice that the two most effective rules are the also included amongst the five most effective rules. Selecting one of these rule-bases for the final explanation depends simply on the degree of parsimony required.

Number of rules		Effectiveness	
Before	After	Before	After
2000	1773	91%	91%
1773	50	91%	88%
50	15	88%	87%
15	6	86%	83%
15	5	87%	83%
15	4	86%	81%
15	3	86%	79%
15	2	86%	74%

Table 6.26: (Breast cancer problem) Optimising the parsimony of the city-block rule-base. The table shows the effectiveness on a sample of 10000 patterns both before and after each optimisation.

The effectiveness gained from the extra three rules is not insignificant, and increases from $\approx 80\%$ for the two rules to $\approx 86\%$ for the five rules. However for the purposes of illustration it was decided to select the two rule explanation for further discussion.

6.5.3 The explanation

Recall that all variables take values between 1 and 10, with 1 referring to a normal state and 10 to the most abnormal state. It is not surprising therefore that the no cancer prototype consists entirely of small values. Notice also that there are bounds on both sides of some of the components of the rule prototype pattern. This implies that the rule antecedent region is bounded on both sides, and so the classification boundary is most definitely curved.

The contributions plots are shown in figure 6.8. The presence of some large contributions in rule 2 implies that it only requires a couple of variables with large values to prevent the rule from being activated. This agrees with the crude intuitive belief that abnormal states are unlikely to be non-cancerous.

Finally, table 6.31 gives an example explanation of why a pattern should be assigned to the cancer class.

Rule 1 : CANCER if			
thickness	-15.691	9	
size	-28.063	6	+28.928
shape	-18.633	5	+38.89
adhesion	-12.113	4	+38.117
epithelial	-31.512	3	+15.056
bare	-13.446	10.2	
chromatin	-16.685	5	
normal	-20.107	9	
mitosis	-8.4274	2	
Rule 2 : NO CANCER if			
thickness		4	+8.1906
size	-74.826	1	+22.025
shape		0	+12.383
adhesion		2	+8.6354
epithelial	-15.375	3	+17.269
bare	-138.55	0.5	+7.2099
chromatin		1	+12.837
normal	-37.533	1	+12.247
mitosis		1	+6.8058
Rule 3 : CANCER if			
thickness	-12.89	5	
size	-13.823	9	+53.713
shape	-8.394	9	+163.93
adhesion	-8.4318	9	
epithelial	-14.287	6	+46.75
bare	-14.074	8.6	+25.389
chromatin	-17.528	8	
normal	-26.58	8	+16.03
mitosis	-5.9996	1	
Rule 4 : CANCER if			
thickness	-7.8866	8	
size	-18.933	4	+24.734
shape	-8.2821	6	+45.501
adhesion	-10.842	3	+39.925
epithelial	-29.016	6	+10.384
bare	-8.6321	10	
chromatin	-9.4427	6	
normal	-14.101	4	+43.79
mitosis	-3.2004	1	
Rule 5 : NO CANCER			
thickness		1	+11.681
size		0	+15.746
shape		1	+10.678
adhesion		1	+10.958
epithelial	-582.2	2	+22.819
bare		-0.8	+13.246
chromatin		1	+25.425
normal	-431.24	-1	+28.939
mitosis		1	+12.385

Table 6.27: (Breast cancer problem) The five most effective city-block rules extracted from the feed-forward network. They are guaranteed to give the same classification as the network and cover approximately 86% of the patterns in the dataset.

	No cancer	Cancer	Total
rule-base effectiveness	438/458 \approx 96%	165/241 \approx 68%	603/699 \approx 86%
rule-base accuracy	435/438 \approx 99%	165/165 \approx 100%	600/603 \approx 100%
network accuracy	447/458 \approx 98%	237/241 \approx 98%	684/699 \approx 98%
comparative accuracy	435/435 \approx 100%	168/168 \approx 100%	603/603 \approx 100%

Table 6.28: (Breast cancer problem) The performance of the five most effective city-block rules shown in table 6.27 on the 699 examples in the dataset.

```

Rule 1 :  CANCER if
thickness  -15.691    9
size      -28.063    6          +28.928
shape     -18.633    5          +38.89
adhesion  -12.113    4          +38.117
epithelial -31.512    3          +15.056
bare      -13.446   10.2
chromatin -16.685    5
normal    -20.107    9
mitosis   -8.4274    2

Rule 2 :  NO CANCER if
thickness          4          +8.1906
size              -74.826    1          +22.025
shape              0          +12.383
adhesion           2          +8.6354
epithelial        -15.375    3          +17.269
bare              -138.55    0.5        +7.2099
chromatin          1          +12.837
normal            -37.533    1          +12.247
mitosis            1          +6.8058

```

Table 6.29: (Breast cancer problem) The final two city-block rules extracted from the feed-forward network. They are guaranteed to give the same classification as the network and cover approximately 80% of the patterns in the dataset.

	No cancer	Cancer	Total
rule-base effectiveness	430/458 \approx 94%	126/241 \approx 52%	556/699 \approx 80%
rule-base accuracy	428/430 \approx 100%	126/126 \approx 100%	554/556 \approx 100%
network accuracy	447/458 \approx 98%	237/241 \approx 98%	684/699 \approx 98%
comparative accuracy	428/428 \approx 100%	128/128 \approx 100%	556/556 \approx 100%

Table 6.30: (Breast cancer problem) The performance of the two city-block rules shown in table 6.29 on the 699 examples in the dataset.

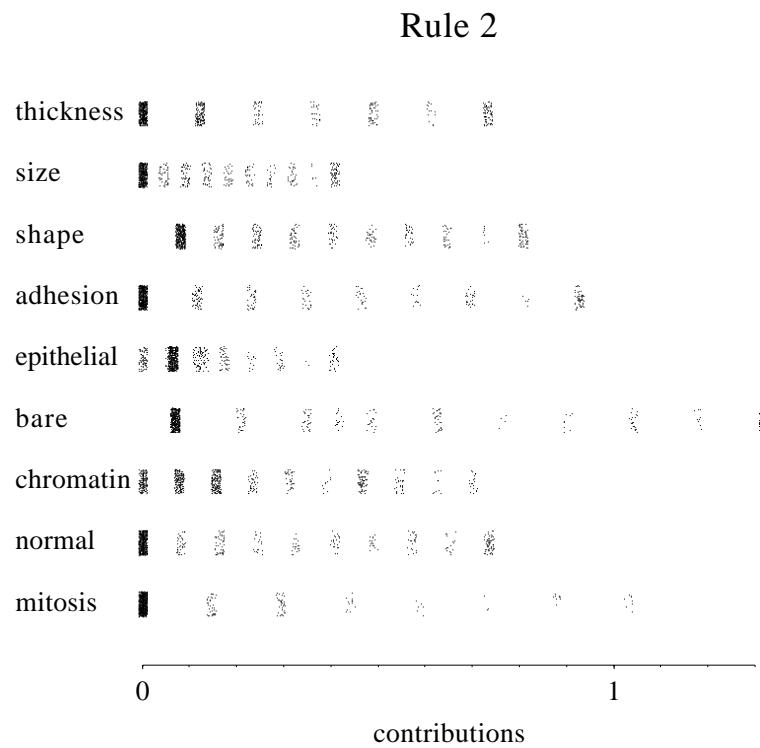
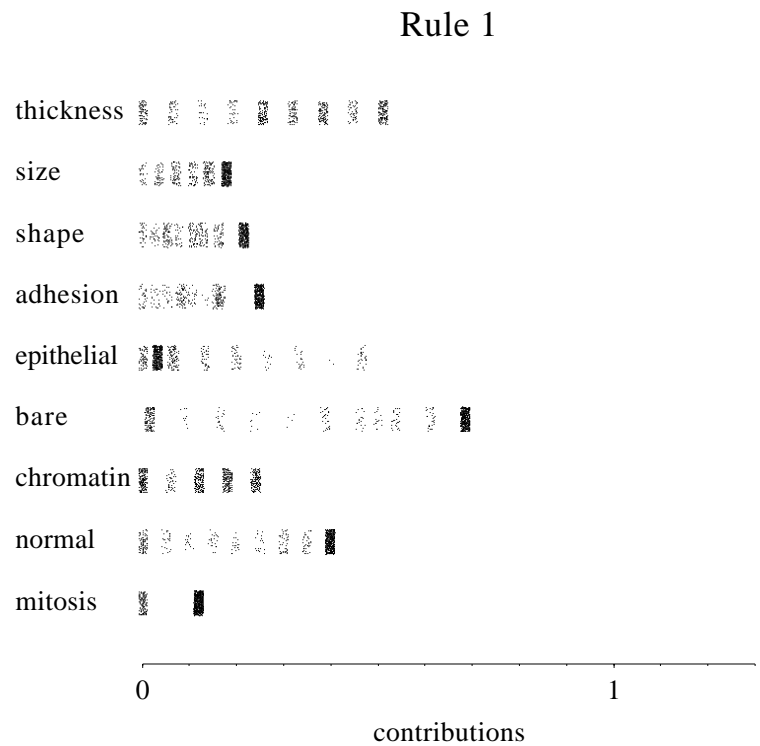


Figure 6.8: (Breast cancer problem) Contributions from each variable in each rule over all 699 patterns in the dataset. Random jitter has been added both vertically and horizontally to separate co-incident points.

Rule 1: CANCER if ** activated **

	pattern	contribution		condition	
thickness	8	0.0637	-15.691	9	
size	7	0.0346	-28.063	6	+28.928
shape	5	0	-18.633	5	+38.89
adhesion	10	0.1574	-12.113	4	+38.117
epithelial	7	0.2657	-31.512	3	+15.056
bare	9	0.0892	-13.446	10.2	
chromatin	5	0	-16.685	5	
normal	5	0.1989	-20.107	9	
mitosis	4	0	-8.4274	2	

TOTAL CONTRIBUTION = 0.8096

Rule 2: NO CANCER if ** not activated **

	pattern	contribution		condition	
thickness	8	0.4884		4	+8.1906
size	7	0.2724	-74.826	1	+22.025
shape	5	0.4038		0	+12.383
adhesion	10	0.9264		2	+8.6354
epithelial	7	0.2316	-15.375	3	+17.269
bare	9	1.1789	-138.55	0.5	+7.2099
chromatin	5	0.3116		1	+12.837
normal	5	0.3266	-37.533	1	+12.247
mitosis	4	0.4408		1	+6.8058

TOTAL CONTRIBUTION = 4.5806

Table 6.31: (Breast cancer problem) Explanation of why a given pattern is assigned to the cancer class.

6.5.4 The effects of decreasing comparative safety

The rule extraction procedure was repeated using a comparative safety less than unity to see whether rules with a greater effectiveness could be obtained. Notice from figure 6.7 that there is considerable separation between the classes. However it was not thought acceptable to classify patterns such that the estimated probability of the class is less than 0.3. Hence it was decided to use a dead-band of 0.3 to 0.7. This is achieved (see figure 3.16 on page 81) by a comparative safety of 0.43.

Table 6.32 gives the results from extracting three large rule-bases with a comparative safety of 0.43, and table 6.33 gives the results of the best two city-block rules. Comparing with the corresponding results for unity comparative safety (tables 6.25 and 6.33), it can be seen that the performance of the city-block rules hardly changes. Hence there was probably insufficient overlap between the two classes of this dataset for the reduction of comparative safety to have an appreciable effect.

Finally, the effectiveness of the large rule-bases of axis-aligned and euclidean rules does increase as a result of decreasing comparative safety. However the optimal two axis-aligned and euclidean rules were still found to have relatively low values of effectiveness (51% and 64% respectively). Thus the city-block rules are again undoubtedly the most effective.

	Axis-aligned	Euclidean	City-block
New sample (10000)	69%	84%	95%
Original data (699)	78%	88%	95%

Table 6.32: (Breast cancer problem) Effectiveness of the three different rule-bases, each containing 1000 rules per class, extracted with a comparative safety of 0.43.

	No cancer	Cancer	Total
rule-base effectiveness	440/458 \approx 96%	140/241 \approx 58%	580/699 \approx 83%
rulebase accuracy	435/440 \approx 99%	139/140 \approx 99%	574/580 \approx 99%
network accuracy	447/458 \approx 98%	237/241 \approx 98%	684/699 \approx 98%
comparative accuracy	436/436 \approx 100%	144/144 \approx 100%	580/580 \approx 100%

Table 6.33: (Breast cancer problem) The performance of two city-block rules extracted with a comparative safety of 0.43, estimated using the 699 examples in the dataset.

Chapter 7

Conclusions and future work

The first two sections of this final chapter provide a complete summary and discussion of the work in this thesis. The last section suggests some directions for future work.

7.1 Overview of thesis and its contributions

The work in this thesis can be divided into two parts: a framework for explanation from feed-forward network classifiers (chapters 2 and 3); and an implementation within this framework (chapters 4 and 5). The former is a natural extension of the probabilistic approach to classification. The latter is an algorithm based upon the well known principle of linearising analytically intractable non-linear functions.

The comprehensive survey of the literature on explanation from neural networks given at the beginning of the thesis showed that none of the published methods met all of the following criteria:

1. a probabilistic framework;
2. an explanation produced from a standard feed-forward network trained using standard procedures;

3. an explanation with a known relationship with the unexplained network-based classification;
4. an explanation consisting of only a small number of easily comprehensible rules which cover a significant fraction of all data;
5. no restriction to datasets with only categorical variables.

The third property is particularly important; without it, the algorithm producing the explanation is simply just another machine learning algorithm. This property is an essential element to any claim that a network has been explained. A solution which meets the above specification is the main contribution of this thesis.

A framework for explanation from feed-forward networks

The statistical perspective for solving the classification problem using feed-forward networks was outlined in chapter 2. It was shown that the decoupling of the probability estimates from the making of decisions is important. Not only does this lead to a better understanding of the classification problem, but it also paves the way for a better understanding of the problems of explaining the classifications given by a feed-forward network.

Chapter 3 provided the core framework for explanation. It began with a generalisation of the popular axis-aligned rules into three types of rule, all based upon different distance metrics with respect to a prototype pattern. These three distances are *city-block* (the weighted absolute sum of the differences between components), *Euclidean* (the weighted sum of differences squared), and *axis-aligned* (the largest weighted difference between component values). Of these three types of rule, the city-block rule was shown to have the most suitable properties for describing the type of classifications provided by a feed-forward network. This conclusion was subsequently much strengthened by the results presented in chapter 6. A key assertion of this thesis is that city-block rules are best for overcoming the fundamental problem of explanation, namely that it is not generally possible exactly to describe the type of classification given by a feed-forward network using only a

small number of simple rules. The reason for this is best understood geometrically: rules in general describe simple regions of input space which cannot be used to reproduce the relatively complex classification boundary defined by the network.

The first part of chapter 3 also showed how the rules for ordered variables can be rewritten as rules for categorical variables. A process of evaluating the “distances” (or contributions) on the numerical codes used for each category within each categorical variable was described. Since this re-writing is a simple process which can be carried out at the end of rule generation, only the extraction of prototype-based rules for ordered variables need be considered.

The second part of chapter 3 described a set of measures for assessing, understanding, and specifying the differences between the rule-based explanation and the original network-based classifier. These measures are *parsimony* (the simplicity of the rule-base), *effectiveness* (the probability that the rule-base is used to classify a pattern), *comparative accuracy* (the probability that the classification given by the rule-base is the same as that given by the network), and *comparative safety* (the severity of the worst sub-optimal classification). The last of these is arguably the most important because it indicates the most extreme difference between the explanation and the network *for any individual classification*. Indeed this importance is reflected in the fact that comparative safety was translated from a measure of an explanation into a constraint in the optimisation of explanation. Thus *comparative safety regions* were defined as the largest regions of input space containing patterns which can all be assigned to a particular class within some specified comparative safety. An approach to explanation was therefore constructed as follows: describe the network-defined comparative safety regions using a small number of the prototype-based rules such that the largest fraction of patterns are explained; in other words, ensure a given comparative safety whilst maximising parsimony and effectiveness.

An implementation

A possible implementation for extracting rules under the explanation framework presented in chapters 2 and 3 was then constructed in chapters 4 and 5. This implementation was based upon the piece-wise linearisation of the function defined by the feed-forward network (an MLP). It was shown in chapter 4 that such a linearisation could easily be achieved by linearising the sigmoid function associated with each hidden node. The method described in chapter 5 took advantage of this approach to enable the comparative safety regions to be described using the union of the intersections of cells and half-planes. Although this cell/half-plane representation does not constitute an explanation, it is an *explicit* representation of the classifications made by the network. A simplifying translation process applied to this first explicit representation produces a second representation consisting of the union of intersections of the half-planes. It was then shown that it is a relatively simple process to describe this second representation using a large number of any of the three prototype-based rules. Simple heuristics were proposed to reduce the number of rules in order to increase the parsimony of the explanation. The final number of rules is then determined by the chosen parsimony/effectiveness trade-off. Clearly this is a function of the particular classification problem being solved.

The new method of making explained classifications presented in this thesis was evaluated on three non-artificial problems, the results of which are given in chapter 6. With all three problems, parsimonious explanations (2 city-block rules) can describe a significant fraction of all patterns (at least 80%) whilst ensuring that the network and the explanation always give the same classification (maximum comparative safety). The application of *reduced* comparative safety was not found to be necessary to generate explanations of the three classifiers. This is almost certainly due to the appropriateness of the city-block rules for describing the type of classification boundaries generated by multi-layer perceptron networks.

7.2 Discussion

This thesis has presented a solution to the problem of explaining a feed-forward network classifier which has several significant advantages over the algorithms reviewed in the opening chapter. Not least is the fact that the explanation has a known and controllable relationship with the network. However another important reason for the success of the extracted explanations is the fact that they use rules based upon the city-block “distance” to a prototype. It has been shown, both in theory and in practice, that these city-block rules are capable of indicating in a compact and intuitive way a significant fraction of those patterns which the network assigns to each class. In addition, it has been shown that the popular axis-aligned rules are fundamentally poor at describing the types of classification boundaries produced by a feed-forward network classifier. Although the emphasis in this thesis has been on explaining networks trained on datasets with ordered variables, this has not prevented explanations from being generated with categorical variables.

It is now possible to be precise about the difference between the network explanation algorithm presented in this thesis and the algorithms described in the literature review of chapter 1. Decompositional and some pedagogical algorithms trade an unknown quantity of comparative safety in order to maximise effectiveness and parsimony. In contrast, the functional methods maximise comparative safety but use a type of rule which is inefficient at describing the types of class boundaries generated by feed-forward networks. The latter results in either an explanation of low parsimony or one of low effectiveness.

The main disadvantages of the new algorithm are that the implementation is currently restricted to two classes, and that the linearisation approach prevents the application to committees of networks. However these disadvantages need to be evaluated against the fact that the system is fully integrated into a probabilistic framework, and that the explanation has a controlled equivalence with the classifications given by the original network. Finally, networks trained on problems containing both ordered and categorical variables can be explained.

7.3 Future work

The following list provides some possible directions of research which would extend and improve the work described in the previous chapters:

1. adapt the algorithm so that it can explain networks trained on problems involving more than two classes;
2. assess the comparative safety of explanations extracted using other algorithms;
3. investigate the computational time required by the algorithm (suspected to be approximately exponential in the number of hidden nodes);
4. explore the possibilities of mixing different types of rule and targeting variables with particularly strong semantics (equivalently, avoiding variables with less intuitive semantics, such as the pedigree function `ped` in the Pima Indian problem of section 6.4 on page 172);
5. improve the piece-wise linearisation algorithm by taking into account the correlations between the error terms ($\epsilon_s(\mathbf{y})$ in equation (4.14) on page 96), and thus obtain a tighter error bound h_s on the MLP function;
6. increase the parsimony of the explanation by exploiting freedom in regions of novelty (see the Karnaugh-map analogy in section 3.5 on page 84, and also section 5.4.4 on page 150).

Appendix A

Encoding categorical variables

This appendix proves that the matrix \mathbf{T}_n ,

$$\mathbf{T}_n = \begin{pmatrix} -\frac{\alpha_2}{1} & \alpha_2 & 0 & 0 & \dots & 0 \\ -\frac{\alpha_3}{2} & -\frac{\alpha_3}{2} & \alpha_3 & 0 & & 0 \\ -\frac{\alpha_4}{3} & -\frac{\alpha_4}{3} & -\frac{\alpha_4}{3} & \alpha_4 & & 0 \\ \vdots & & & & \ddots & \vdots \\ -\frac{\alpha_n}{n-1} & -\frac{\alpha_n}{n-1} & -\frac{\alpha_n}{n-1} & -\frac{\alpha_n}{n-1} & -\frac{\alpha_n}{n-1} & \alpha_n \end{pmatrix} \quad (\text{A.1})$$

where

$$\alpha_k = \sqrt{\frac{k-1}{2k}}, \quad (\text{A.2})$$

has the following properties:

1. the columns sum to zero,
2. the magnitude of every column is α_n ,
3. the Euclidean distance between any two columns is 1.0.

Proposition 1 is immediately obvious from the construction of the matrix. Induction over n is used to prove proposition 2. Assume that the magnitude of all the columns of \mathbf{T}_k

equal α_k , then the (magnitude)² of any of the first k columns of \mathbf{T}_{k+1} equals

$$\begin{aligned}
 \alpha_k^2 + \left(\frac{\alpha_{k+1}}{k^2}\right)^2 &= \frac{k-1}{2k} + \frac{1}{k^2} \frac{k}{2(k+1)} \\
 &= \frac{(k-1)(k+1) + 1}{2k(k+1)} \\
 &= \frac{k}{2(k+1)} \\
 &= \alpha_{k+1}^2.
 \end{aligned} \tag{A.3}$$

Clearly the (magnitude)² of the $(k+1)$ th column is also α_{k+1}^2 , thus completing the inductive step. The base case of \mathbf{T}_2 is trivially true, thus proposition 2 is true by induction for all $k \geq 2$.

Induction is also used to prove proposition 3. Assume that the euclidean distance between all distinct columns of \mathbf{T}_k is equal to unity, then clearly the euclidean distance between any of the first k columns of \mathbf{T}_{k+1} is also equal to unity. Using the fact that all columns have a magnitude α_{k+1}^2 , the (distance)² from the $(k+1)$ th column to any other column equals

$$\begin{aligned}
 \alpha_{k+1}^2 - \left(\frac{\alpha_{k+1}}{k}\right)^2 + \left(\alpha_{k+1} + \frac{\alpha_{k+1}}{k}\right)^2 &= \alpha_{k+1}^2 \left(1 - \frac{1}{k^2} + \left(\frac{1+k}{k}\right)^2\right) \\
 &= \frac{k}{2(k+1)} \left(\frac{k^2 - 1 + (1+k)^2}{k^2}\right) \\
 &= \frac{1}{2(k+1)} \frac{2k^2 + 2k}{k} \\
 &= 1.
 \end{aligned} \tag{A.4}$$

This completes the inductive step. The base case of \mathbf{T}_2 is trivially true because $\alpha_2 = 0.5$, hence proposition 3 is true by induction for all $k \geq 2$.

Appendix B

Density estimation

A method similar to the suggestions given in Silverman [Sil96] was used to obtain a kernel based estimate of the unconditional probability density. Each kernel consists of a product of one-dimensional distributions, where a Gaussian distribution is used for each ordered variable and an impulse distribution is used for each categorical variable. The following equations make the structure of the model precise.

$$p(\mathbf{x}) = \sum_i p(\mathbf{x} | i) P(i) \quad (\text{B.1})$$

where the distribution of each kernel is given by

$$p(\mathbf{x} | i) = \prod_j p(x_j | i) \quad (\text{B.2})$$

and the 1D kernel distributions are given by

$$p(x_j | i) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} \exp\left(-\frac{(x_j - \mu_{ij})^2}{2\sigma_{ij}^2}\right) & \text{if } x_j \text{ is an ordered variable} \\ \delta(x_j, \mu_{ij}) P_{ij} + (1 - \delta(x_j, \mu_{ij})) \frac{1 - P_{ij}}{n_j - 1} & \text{if } x_j \text{ is a categorical variable} \end{cases} \quad (\text{B.3})$$

where n_j is the number of categories in (categorical) variable x_j .

The parameters of the model are the kernel priors $P(i)$, the ordered variances σ_{ij}^2 , the categorical impulses P_{ij} , and the centres μ_{ij} . These parameters are selected in the following way. The training data is partitioned into v folds and the patterns from the first partition are used to position the centres of the kernels in the first model. Keeping these centres fixed, the likelihood of $p(\mathbf{x})$ is maximised (using expectation maximisation (EM) [DLR77]) on the data in the remaining partitions, but constraining the kernel widths on each dimension to be equal *i.e.* $\sigma_{ij}^2 = \sigma_{kj}^2$ and $P_{ij} = P_{kj}$ for all i, k . This procedure is repeated over all partitions to produce v models, and the output from these models is averaged to produce the final density estimate.

This density model will have a kernel for every pattern in the training set. Note that increasing the number of partitions tends to make the final distribution “broader”. This is because each of the v intermediate models tries to account for a relatively large amount of data with only a small number of kernels. The number of partitions may be selected by plotting the marginal distributions and comparing with histograms of the original data. Obviously a good density model will produce marginal distributions which are very similar to the marginal histograms; unfortunately the reverse is not true.

Appendix C

Justifying a novelty threshold

Normality can be defined as the smallest volume which encloses some fraction r of the unconditional distribution $p(\mathbf{x})$. The following proof shows that this volume \mathcal{N} is defined by some threshold t on the unconditional distribution.

By definition,

$$\int_{\mathcal{N}} p(\mathbf{x}) d\mathbf{x} = r, \quad (\text{C.1})$$

and we wish to show that the minimum volumed \mathcal{N} is defined by

$$\mathcal{N} = \{\mathbf{x} \mid p(\mathbf{x}) \geq t\}, \quad (\text{C.2})$$

for some t .

Consider another volume \mathcal{N}_0 which also covers 100% of the distribution but which contains some points (A)bove the threshold and some points (B)elow the threshold,

$$\int_{\mathcal{N}_0} p(\mathbf{x}) d\mathbf{x} = r, \quad (\text{C.3})$$

$$\mathcal{N}_0 = \mathcal{A} \cup \mathcal{B} \quad (\text{C.4})$$

where

$$p(\mathbf{x}) \geq t \quad \text{for every } \mathbf{x} \in \mathcal{A} \quad (\text{C.5})$$

$$p(\mathbf{x}) < t \quad \text{for every } \mathbf{x} \in \mathcal{B}. \quad (\text{C.6})$$

Note that $\mathcal{A} = \mathcal{N} \cap \overline{\mathcal{N} \cap \overline{\mathcal{A}}}$ because $\mathcal{A} \subseteq \mathcal{N}$, and also that \mathcal{A} and \mathcal{B} are disjoint, $\mathcal{A} \cap \mathcal{B} = \emptyset$.

Hence

$$\begin{aligned} \int_{\mathcal{N}_0} d\mathbf{x} &= \int_{\mathcal{N} \cap \overline{\mathcal{N} \cap \overline{\mathcal{A}}}} d\mathbf{x} + \int_{\mathcal{B}} d\mathbf{x} \\ &= \int_{\mathcal{N}} d\mathbf{x} - \int_{\mathcal{N} \cap \overline{\mathcal{A}}} d\mathbf{x} + \int_{\mathcal{B}} d\mathbf{x}. \end{aligned} \quad (\text{C.7})$$

However we will now show that

$$- \int_{\mathcal{N} \cap \overline{\mathcal{A}}} d\mathbf{x} + \int_{\mathcal{B}} d\mathbf{x} \geq 0. \quad (\text{C.8})$$

This clearly implies the desired result that \mathcal{N} has the smallest volume covering 100% of the unconditional distribution.

Using the same identities as before,

$$\int_{\mathcal{N}_0} p(\mathbf{x}) d\mathbf{x} = \int_{\mathcal{N}} p(\mathbf{x}) d\mathbf{x} - \int_{\mathcal{N} \cap \overline{\mathcal{A}}} p(\mathbf{x}) d\mathbf{x} + \int_{\mathcal{B}} p(\mathbf{x}) d\mathbf{x}. \quad (\text{C.9})$$

However

$$\begin{aligned} \int_{\mathcal{N}_0} p(\mathbf{x}) d\mathbf{x} &= r \\ \int_{\mathcal{N}} p(\mathbf{x}) d\mathbf{x} &= r \end{aligned} \quad (\text{C.10})$$

and because of the properties of \mathcal{A} and \mathcal{B} ,

$$\begin{aligned} \int_{\mathcal{N} \cap \bar{\mathcal{A}}} p(\mathbf{x}) d\mathbf{x} &\geq \int_{\mathcal{N} \cap \bar{\mathcal{A}}} r d\mathbf{x} \\ \int_{\mathcal{B}} p(\mathbf{x}) d\mathbf{x} &< \int_{\mathcal{B}} r d\mathbf{x}. \end{aligned} \tag{C.11}$$

So equation (C.9) becomes

$$-\int_{\mathcal{N} \cap \bar{\mathcal{A}}} r d\mathbf{x} + \int_{\mathcal{B}} r d\mathbf{x} \geq 0. \tag{C.12}$$

Since r is non-zero, this implies the required condition, equation (C.8)

Appendix D

Proofs of the rule fraction formulae

This appendix outlines the proofs of the formulae used to calculate the fraction of space covered by each of the three types of rule when pushed up against different types of boundary. These formulae are quoted in tables 3.1 and 3.2 on page 67.

The basic volumes

This section gives the volumes of each of the three types of rule antecedent region.

Volume of a city-block region

First consider the volume defined by the positive “quadrant” and

$$\sum_i U(x_i) x_i \leq l, \tag{D.1}$$

where U is the unit step function. Let this volume be $V_d(l)$ in d dimensions, and define I_d by

$$V_d'(l) = I_d l^d. \tag{D.2}$$

Observe that

$$\begin{aligned} V'_{d+1}(l) &= \int_0^l I_d x^d dx \\ &= \frac{I_d}{d+1} l^{d+1} + c_d. \end{aligned} \quad (\text{D.3})$$

where the constant integration c_d is found to be zero by evaluating at $l = 0$. Since $I(2) = 0.5$ it is clear that $I_d = 1/d!$ and hence

$$V'_d(l) = \frac{l^d}{d!}. \quad (\text{D.4})$$

Now consider the volume of a city-block antecedent region defined by the positive quadrant and the condition

$$\sum_i U(x_i - \mu)(x_i - \mu) \leq l, \quad (\text{D.5})$$

where $0 \leq \mu \leq l$. Observe that each of the combinations for which the different terms in this condition are non-zero represent disjoint volumes. Hence using the result given by equation (D.4),

$$V_d(l) = \sum_{i=0}^d \binom{d}{i} \mu^i \frac{l^{(d-i)}}{(d-i)!}. \quad (\text{D.6})$$

Volume of a Euclidean region

It is shown in [Bis95] that the volume of a hyper-sphere of radius r in an even number of dimensions, d , is given by

$$\frac{(2\pi)^{d/2} r^d}{d(d-2)\dots 2}.$$

Hence the volume of a hyper-sphere in only the positive “quadrant” is given by

$$V_d(r) = \frac{(\frac{\pi}{4})^{d/2} r^d}{(\frac{d}{2})!}. \quad (\text{D.7})$$

Volume of an axis-aligned region

The volume of an axis-aligned region with dimension l is given by

$$V_d(l) = l^d. \quad (\text{D.8})$$

The relative volumes

This section gives the fraction of space covered by each of the three types of region when pushed up against either a “45°” hyper-plane corner or hyper-spherical corner. The trivial cases are omitted.

Axis-aligned region against a hyper-plane corner

The space to be covered is given by

$$\begin{aligned} x_i &\geq 0 \quad \text{for all } i \\ \sum_i x_i &\leq l_1. \end{aligned} \quad (\text{D.9})$$

Hence the dimension of the largest axis-aligned region within this space is

$$l_2 = \frac{l_1}{d}. \quad (\text{D.10})$$

So using results (D.4) and (D.8) the fraction of space covered is given by

$$\rho = \frac{l_2^d}{l_1^d/d!} = \frac{d!}{d^d}. \quad (\text{D.11})$$

Euclidean region against a hyper-plane corner

The space to be covered is given by

$$\begin{aligned} x_i &\geq 0 \quad \text{for all } i \\ \sum_i x_i &\leq l, \end{aligned} \tag{D.12}$$

and so the radius r of the largest hyper-sphere ‘‘quadrant’’ within this space is given by

$$r = \frac{l}{\sqrt{d}}. \tag{D.13}$$

Hence using results (D.4) and (D.7) the fraction of space covered is given by

$$\rho = \frac{\frac{(\pi/4)^{d/2}}{(d/2)!} r^d}{\frac{l^d}{d!}} = \left(\frac{\pi}{4d}\right)^{d/2} \frac{d!}{(d/2)!}. \tag{D.14}$$

City-block region against a hyper-sphere corner

The space to be covered is given by

$$\begin{aligned} x_i &\geq 0 \quad \text{for all } i \\ \sum_i x_i^2 &\leq r^2, \end{aligned} \tag{D.15}$$

and define the city-block region by

$$\begin{aligned} x_i &\geq 0 \quad \text{for all } i \\ \sum_i \text{U}(x_i - \mu)(x_i - \mu) &\leq 1.0. \end{aligned} \tag{D.16}$$

Note that for convenience the last inequality is normalised to be less than 1.0 rather than l .

The configuration of the largest hyper-sphere just touching the city-block region is when

all the x_i equal μ , except for one x_i which equals $\mu + 1$. Hence

$$\begin{aligned} r^2 &= (d-1)\mu^2 + (\mu+1)^2 \\ &= d\mu^2 + 2\mu + 1. \end{aligned} \tag{D.17}$$

Thus using results (D.6) and (D.7), the fraction of space covered is given by

$$\begin{aligned} \rho &= \frac{\sum_{i=0}^d d \binom{d}{i} \mu^i \frac{1}{(d-i)!}}{\frac{(\pi/4)^{d/2} r^d}{(d/2)!}} \\ &= \frac{d!(d/2)!}{(\pi/4)^{d/2} r^d} \sum_i \frac{1}{((d-i)!)^2 i!} \mu^i. \end{aligned} \tag{D.18}$$

Maximising this with respect to μ gives

$$dr^{d-1} \frac{d\mu+1}{r} \sum_i B_i \mu^i = r^d \sum_i B_i i \mu^{i-1} \tag{D.19}$$

where

$$B_i = \frac{1}{((d-i)!)^2 i!}. \tag{D.20}$$

This may be re-written

$$\mu = \frac{d\mu^2 + 2\mu + 1}{d^2} \frac{\sum_i B_i i \mu^{i-1}}{\sum_i B_i \mu^i} - \frac{1}{d} \tag{D.21}$$

which may be iterated to obtain a numerical solution for the optimal μ . This may then be substituted into equations (D.17) and (D.18) to obtain a numerical value for the fraction of space covered.

Axis-aligned region against a hyper-sphere corner

The space to be covered is given by

$$\begin{aligned} x_i &\geq 0 \quad \text{for all } i \\ \sum_i x_i^2 &\leq r^2, \end{aligned} \tag{D.22}$$

and so the dimension of the largest axis-aligned region within this space is

$$l = \frac{r}{\sqrt{d}}. \tag{D.23}$$

Using the results (D.8) and (D.7), the fraction of space covered is

$$\rho = \frac{l^d}{\frac{(\pi/4)^{d/2} r^d}{(d/2)!}} = \frac{(d/2)!}{(d\pi/4)^{d/2}}. \tag{D.24}$$

Bibliography

- [ACD⁺96] Robert Andrews, Russell Cable, Joachim Diederich, Shlomo Geva, Mostefa Golea, Ross Hayward, Chirs Ho-Stuart, and Alan B Tickle. An evaluation and comparison of techniques for extracting and refining rules from artificial neural networks. Technical report, Neurocomputing Research Centre, Queensland University of Technology, Box 2434 GPO, Brisbane 4001, Queensland, Australia, March 1996. <http://www.fit.qut.edu.au/>.
- [AG96a] Robert Andrews and Shlomo Geva. Rule refinement and local function networks. *Proceedings of the NIPS*96 Rule Extraction From Trained Artificial Neural Network Workshop*, 1996.
- [AG96b] Robert Andrews and Shlomo Geva. Rulex and cebp networks as the basis for a rule refinement system. Technical report, Neurocomputing Research Centre, Faculty of Information Technology, Queensland University of Technology, GPO Box 2434 Brisbane 4001, Queensland, Australia, 1996. <http://www.fit.qut.edu.au/>.
- [AJD95] Robert Andrews and Alan B Tickle Joachim Diederich. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8(6):373–389, December 1995.
- [AKTK93] Shigeo Abe, Masahiro Kayama, Hiroshi Takenaga, and Tadaaki Kitamra. Extracting algorithms from pattern classification neural networks. *Neural Networks*, 6:729–735, 1993.
- [And98] Robert Andrews. Rule extraction from trained artificial neural networks, 1998. <http://www.fit.qut.edu.au/~robert/rulex.html>.
- [Bax92] William G Baxt. Analysis of the clinical variables driving decision in an artificial neural network trained to identify the presence of myocardial infarction. *Annals of Emergency Medicine*, 21:1439–1444, 1992.
- [Bax93] William G Baxt. A neural network trained to identify the presence of myocardial infarction bases diagnostic decision on nonlinear relationship between input variables. *Neural Computing & Applications*, 1:176–182, 1993.
- [Bax95] William G Baxt. Application of artificial neural networks to clinical medicine. *The Lancet*, 346:1135–1138, October 28 1995.

- [BCR97] J M Benítez, J L Castro, and I Requena. Are artificial neural networks black boxes ? *IEEE Transactions on Neural Networks*, 8(5), September 1997.
- [BFOS84] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and Regression Trees*. Chapman & Hall, 1984.
- [BHC92] James J Buckley, Yoichi Hayashi, and Ernest Czogala. On the equivalence of neural networks and fuzzy expert systems. *Advances in Neural Information Processing Systems*, 1992.
- [Bis95] Christopher M Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
- [BL91] E K Blum and L K Li. Approximation theory and feed-forward networks. *Neural Networks*, 4(4):511–515, 1991.
- [Bla93] Reinhard Blasig. Gds: Gradient descent generation of symbolic classification rules. *Advances in Neural Information Processing Systems*, 6:1093–1100, 1993.
- [Bol96] Guido Bologna. Rule extraction from the imlp neural network: a comparative study. *Proceedings of the NIPS*96 Rule Extraction From Trained Artificial Neural Network Workshop*, 1996.
- [Boz98] Olcay Boz. Bibliography on integration of symbolism with connectionism, and rule integration and extraction in neural networks, 1998. <http://www.lehigh.edu/~ob00/integrated/references-new.html>.
- [CHK95] Simon S Cross, Robert F Harrison, and R Lee Kennedy. Introduction to neural networks. *The Lancet*, 346:1075–1079, October 21 1995.
- [Chv80] Vašek Chvátal. *Linear Programming*. W H Freeman, 1980.
- [CS94] Mark W Craven and Jude W Shavlik. Using sampling and queries to extract rules from trained neural networks. *Machine Learning*, 4(3):37–45, 1994.
- [DAC⁺96] Dagli, Akay, Chen, Fernandez, and Ghosh, editors. *Intelligent Engineering Systems through Artificial Neural Networks*, volume 6. ASME Press, St. Louis, November 1996.
- [DLR77] A P Dempster, N M Laird, and D B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39(1):1–38, 1977.
- [FA96] R J H Filer and J Austin. A neural network interrogation tool for the intensive therapy unit. *Proceedings of the 2nd International Conference on Neural Networks and Expert Systems in Medicine and Health Care*, 1996.
- [Fou98] Robert Fourer. Linear programming frequently asked questions. Optimisation Technology Centre of Northwestern University and Argonne National Laboratory, 1998. <http://www-c.mcs.anl.gov/home/otc/Guide/faq/linear-programming-faq.html%>.

- [FSA96] Richard J H Filer, Ishwar K Sethi, and James Austin. A comparison between two rule extraction methods for continuous input data. *Proceedings of the NIPS*96 Rule Extraction From Trained Artificial Neural Network Workshop*, 1996.
- [Fu94] LiMin Fu. Rule generation from neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(8):1114–1124, August 1994.
- [Fuk90] K Fukunaga. *Introduction to Statistical Pattern Recognition*. San Diego: Academic Press, 2 edition, 1990.
- [GHM92] Rodney M Goodman, Charles M Higgins, and John W Miller. Rule-based neural networks for classification and probability estimation. *Neural Computation*, 4:781–804, 1992.
- [GW96] TH Goh and Francis Wong. Semantic extraction using neural network modelling and sensitivity analysis. Technical report, Institute of Systems Science, National University of Singapore, Heng Mui Keng Terrace, Kent Ridge, Singapore 0511, 1996. <http://iss.nus.sg>.
- [Hay90] Yoichi Hayashi. A neural expert system with automated extraction of fuzzy if-then rules and its applications to medical diagnosis. *Advances in Neural Information Processing Systems*, 3:578–584, 1990.
- [Hay99] Simon Haykin. *Neural Networks, A Comprehensive Foundation. Second edition*. Macmillan, 1999.
- [IC94] James P Ignizio and Tom M Cavalier. *Linear Programming*. Prentice Hall, 1994.
- [Jav98] The source for java technology, 1998. <http://java.sun.com/>.
- [JGDAC⁺98] C Jutten, A Guérin-Dugué, C Aviles-Cruz, J L Voz, and D Van Cappel. Basic research esprit project number 6891: Enhanced learning for evolutive neural architectures, 1998. <http://www.dice.ucl.ac.be/neural-nets/ELENA/ELENA.html>.
- [Jon90] L K Jones. Constructive approximations for neural networks by sigmoidal functions. *Proceedings of the IEEE*, 78(10):1589–1589, 1990.
- [Kar53] M Karnaugh. A map method for synthesis of combinatorial logic circuits. *Transactions AIEE, Communications and Electronics*, 72(part I):593–599, 1953.
- [Kre88] Erwin Kreyszig. *Advanced Engineering Mathematics*. John Wiley & Sons, 1988.
- [Kri96] R Krishnan. A systematic method for decompositional rule extraction from neural networks. *Proceedings of the NIPS*96 Rule Extraction From Trained Artificial Neural Network Workshop*, 1996.
- [Leo97] Cornelius T Leondes, editor. *Industrial and Manufacturing Systems (Neural Network Systems Techniques and Applications)*, volume 4. Academic Press Ltd, 1997.

- [LLS97] Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. An empirical comparison of decision trees and other classification methods. Technical Report 979, Department of Statistics, University of Wisconsin, Madison, <http://www.stat.wisc.edu/~limt/>, January 1997.
- [MA95] P M Murphy and D W Aha. *UCI Repository of Machine Learning Databases*. Irvine, CA: University of California, Dept of Information and Computer Science., 1995. <ftp://ftp.ics.uci.edu/pub/machine-learning-databases>.
- [Mak75] John Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, apr 1975.
- [Mø193] M Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993.
- [MP95] Sushmita Mitra and Sankar K Pal. Fuzzy multi-layer perceptron, inferencing and rule generation. *IEEE Transactions on Neural Networks*, 6(1):51–63, January 1995.
- [MSS91] W Maass, G Schnitger, and E D Sontag. On the computational power of sigmoids versus boolean threshold circuits. *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science*, pages 767–776, 1991.
- [MST94] D Michie, D J Spiegelhalter, and C C Taylor, editors. *Machine Learning, Neural and Statistical classification*. Ellis Horwood, 1994. <http://www.ncc.up.pt/liacc/ML/statlog/>.
- [MW90] O L Mangasarian and W H Wolberg. Cancer diagnosis via linear programming. *Siam News*, 23(5):1–18, September 1990.
- [NCCR⁺97] Alexandre Nairac, Timothy Corbett-Clark, Ruth Ripley, Neil Townsend, and Lionel Tarassenko. Choosing an appropriate model for novelty detection. *Proceedings 5th IEE Int. Conf. on Artificial Neural Networks. Cambridge 1997*, pages 117–122, 1997.
- [PBGB98] Tony Plate, Joel Bert, John Grace, and Pierre Band. Visualising the function computed by a feed-forward neural network. Technical Report CS-TR-98-5, School of Mathematical and Computing Sciences, Victoria University of Wellington, PO Box 600, Wellington, New Zealand, July 1998. <http://www.mcs.vuw.ac.nz/~tap/publications.html>.
- [Pre94] Lutz Prechelt. Proben1 - a set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, Information Faculty, Karlsruhe University, Fakultät für Informatik, Universität Karlsruhe, 76128 Karlsruhe, Germany, September 1994. <ftp://ftp.ira.uka.de/pub/neuron/proben1.tar.gz>.
- [Pre98] Lutz Prechelt. Benchmarking of learning algorithms, 1998. http://www.ipd.ira.uka.de/~prechelt/NIPS_bench.html.
- [PTVF92] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.

- [Qui79] J Quinlan. *Discovering Rules from Large Collections of Examples: A Case Study*. Edinburgh University Press, 1979. In *Expert Systems in the Micro Electronic Age*.
- [Qui93] J R Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.
- [Ref95] A P Refenes. *Neural Networks in the Capital Markets*. Wiley, Chichester, 1995.
- [Ric95] John A Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, 1995.
- [Rip96] B D Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.
- [RNH⁺98] C E Rasmussen, R M Neal, G E Hinton, D van Camp, M Revow, Z Ghahramani, R Kustra, and R Tibshirani. Delve: Data for evaluating learning in valid experiments, 1998. <http://www.cs.utoronto.ca/~delve/>.
- [San89] Dennis Sanger. Contribution analysis: A technique for assigning responsibilities to hidden units in connectionist networks. *Connection Science*, 1(2):115–138, 1989.
- [Set97] Rudy Setiono. Extracting rules from neural networks by pruning and hidden-unit splitting. *Neural Computation*, 9(1):205–225, 1997.
- [Sil96] B W Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, 1996.
- [SL96] R Setiono and H Liu. Symbolic representation of neural networks. *IEEE Computer*, 29(3):71–77, March 1996.
- [SN88] Kazumi Saito and Ryohei Nakano. Medical diagnostic expert system based on pdp model. *Proceedings of the ICNN*, 4(1):255–262, 1988.
- [SR87] Terrence J Sejnowski and Charles R Rosenberg. Parallel networks that learn to pronounce english text. *Complex Systems*, 1:145–168, 1987.
- [Sta98] Statlib, a system for distributing statistical software, datasets, and information by electronic mail, ftp and www., 1998. <http://lib.stat.cmu.edu/>.
- [SY96] I K Sethi and J H Yoo. Symbolic mapping of neurons in feed-forward networks. *Pattern Recognition Letters*, 17:1035–1046, 1996.
- [Tar98] L Tarassenko. *A Guide to Neural computing applications*. Arnold, 1998.
- [TG96a] Ismail Taha and Joydeep Ghosh. Symbolic interpretation of artificial neural networks. Technical report, Department of Electrical and Computer Engineering, University of Texas, University of Texas, Austin, TX 78712-1084, September 1996. <http://pegasus.ece.utexas.edu:80/~ismail>.

- [TG96b] Ismail Taha and Joydeep Ghosh. Three techniques for extracting rules from feed-forward networks. Technical report, Department of Electrical and Computer Engineering, University of Texas, University of Texas, Austin, TX 78712-1084, August 1996. <http://pegasus.ece.utexas.edu:80/~ismail>.
- [THA92] Volker Tresp, Jurgen Hollatz, and Subutai Ahmad. Network structuring and training using rule-based knowledge. *Advances in Neural Information Processing Systems*, 5:871–878, 1992.
- [THCB95] L Tarassenko, P Hayton, N Cerneaz, and M Brady. Novelty detection for the identification of masses in mammograms. *Proceedings of the Fourth International IEE Conference on Artificial Neural Networks (Cambridge, 1995)*, 1995.
- [Thr94] Sebastian Thrun. Extracting symbolic knowledge from artificial neural networks. Technical report, Dept. of Computer Science III, University of Bonn, Rumerstr: 164, D-5300 Bonn1, Germany, 1994.
- [Thr95] Sebastian Thrun. Extracting rules from artificial neural networks with distributed representations. *Advances in Neural Information Processing Systems*, 7, 1995.
- [TI97] Lloyd N Trefethen and David Bau III. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, 1997.
- [TS91] Geoffrey Towell and Jude W Shavlik. Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules. *Advances in Neural Information Processing Systems*, 4:977–984, 1991.
- [TS92] Geoffrey G Towell and Jude W Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13(1):71–101, 1992.
- [TWGH96] L Tarassenko, R Whitehouse, G Gasparini, and A L Harris. Neural network prediction of relapse in breast cancer patients. *Neural Computing and Applications*, 4:105–113, 1996.
- [VR97] W N Venables and B D Ripley. *Statistics and Computing*. Springer-Verlag New York, 2 edition, 1997.
- [WF98] Matt White and Scott E Fahlman. Cmu repository of neural network benchmarks, 1998. <http://www.boltz.cs.cmu.edu/>.
- [WM90] W H Wolberg and O L Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Sciences, USA*, 87:9193–9196, December 1990.
- [WTL88] W H Wolberg, M A Tanner, and W-Y Loh. Diagnostic schemes for fine needle aspirates of breast masses. *Analytical and Quantitative Cytology and Histology*, 10:225–228, 1988.
- [WTL89] W H Wolberg, M A Tanner, and W-Y Loh. Fine needle aspiration for breast mass diagnosis. *Archives of Surgery*, 124:814–818, 1989.

- [WTLV87] W H Wolberg, M A Tanner, W-Y Loh, and N Vanichsetakul. Statistical approach to fine needle aspiration diagnosis of breast masses. *Acta Cytologica*, 31:737–741, 1987.
- [Wya95] Jeremy Wyatt. Nervous about artificial neural networks ? *The Lancet*, 346:1175–1176, November 4 1995.
- [Zhe93] Zijian Zheng. A benchmark for classifier learning. Technical Report 474, Basser Department of Computer Science, The University of Sydney, N.S.W. Australia 2006, November 1993. <ftp://ftp.cs.su.oz.au/pub/tr/TR93.474.ps.Z>.