Knowledge Extraction from Local Function Networks

Kenneth McGarry, Stefan Wermter and John MacIntyre
School of Computing, Engineering and Technology,
University of Sunderland, St Peters Campus,
St Peters Way, Sunderland, SR6 ODD, UK
ken.mcgarry@sunderland.ac.uk

Abstract

Extracting rules from RBFs is not a trivial task because of nonlinear functions or high input dimensionality. In such cases, some of the hidden units of the RBF network have a tendency to be "shared" across several output classes or even may not contribute to any output class. To address this we have developed an algorithm called LREX (for Local Rule EXtraction) which tackles these issues by extracting rules at two levels: *h*REX extracts rules by examining the *hidden* unit to class assignments while *m*REX extracts rules based on the input space to output space *mappings*. The rules extracted by our algorithm are compared and contrasted against a competing local rule extraction system. The central claim of this paper is that local function networks such as radial basis function (RBF) networks have a suitable architecture based on Gaussian functions that is amenable to rule extraction.

1 Introduction

Neural networks have been applied to many real-world, large-scale problems of considerable complexity. They are useful for pattern recognition and they are robust classifiers, with the ability to generalize in making decisions about imprecise input data [Bishop, 1995]. They offer robust solutions to a variety of classification problems such as speech, character and signal recognition, as well as functional prediction and system modeling where the physical processes are not understood or are highly nonlinear.

Although neural networks have gained acceptance in many industrial and scientific fields they have not been widely used by practitioners of mission critical applications such as those engaged in aerospace, military and medical systems. This is understandable since neural networks do not lend themselves to the normal software engineering development process. Knowledge extraction by forming symbolic rules from the internal parameters of neural networks is now becoming an accepted technique for overcoming some of their limitations [Shavlik, 1994; Sun, 2000].

In this paper we describe our method of extracting knowledge from an RBF network which is classed as a local type of neural network. That is, its internal parameters are limited to responding to a limited subset of the input space. We also compare and contrast our technique with a specialized local type neural architecture. The extracted rules are examined for comprehensibility, accuracy, number of rules generated and the number of antecedents contained in a rule.

The paper is structured as follows: section two describes the motivations for performing knowledge extraction. Section three describes why the architecture of the radial basis function network is particulary suitable for knowledge extraction. Section four outlines how our knowledge extraction algorithm produces rules from RBF networks and section five explains the results of the experimental work. Section six discusses the conclusions of the experimental work.

2 Knowledge Extraction

In this section we discuss motivations, techniques and methodology for knowledge extraction from RBF networks. RBF networks provide a localized solution [Moody and Darken, 1989] that is amenable to extraction, which section three discusses in more detail. It is possible to extract a series of IF..THEN rules that are able to state simply and accurately the knowledge contained in the neural network. In recent years there has been a great deal of interest in researching techniques for extracting symbolic rules from neural networks. Rule extraction has been carried out upon a variety of neural network types such as multi-layer perceptrons [Thrun, 1995], Kohonen networks and recurrent networks [Omlin and Giles, 1994]. The advantages of extracting rules from neural networks can be summarized as follows:

- The knowledge learned by a neural network is generally difficult to understand by humans. The provision of a mechanism that can interpret the networks input/output mappings in the form of rules would be very useful.

- Deficiencies in the original training set may be identified, thus the generalization of the network may be improved by the addition/enhancement of new classes. The identification of superfluous network parameters for removal would also enhance network performance.
- Analysis of previously unknown relationships in the data. This feature has a huge potential for knowledge discovery/data mining and possibilities may exist for scientific induction [Craven and Shavlik, 1997].

In addition to providing an explanation facility, rule extraction is recognised as a powerful technique for neuro-symbolic integration within hybrid systems [McGarry *et al.*, 1999].

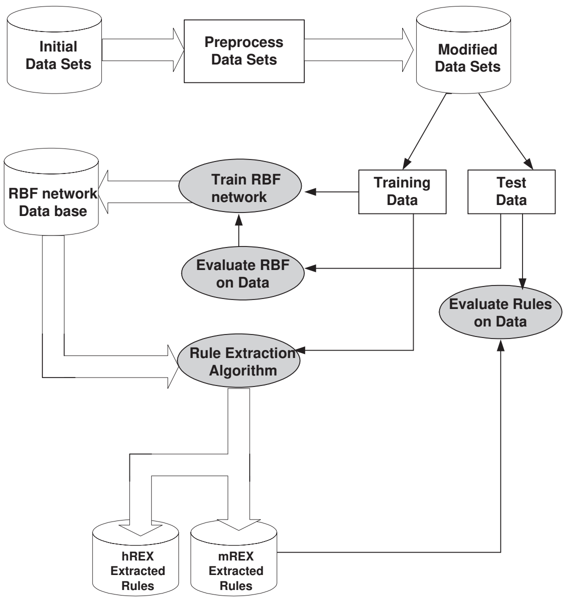

Figure 1: Knowledge extraction system data flow and data transformation

3 Radial Basis Function Networks

Radial basis function (RBF) neural networks are a model that has functional similarities found in many biological neurons. In biological nervous systems certain cells are responsive to a narrow range of input stimuli, for example in the ear there are cochlear stereocilla cells which are locally tuned to particular frequencies of sound [Moody and Darken, 1989]. Figure 2 shows a network trained on a noisy Xor data set for illustration. This network has two input features, two output classes and four hidden units.

The RBF network consists of a feedforward architecture with an input layer, a hidden layer of RBF "pattern" units and an output layer of linear units. The input layer simply

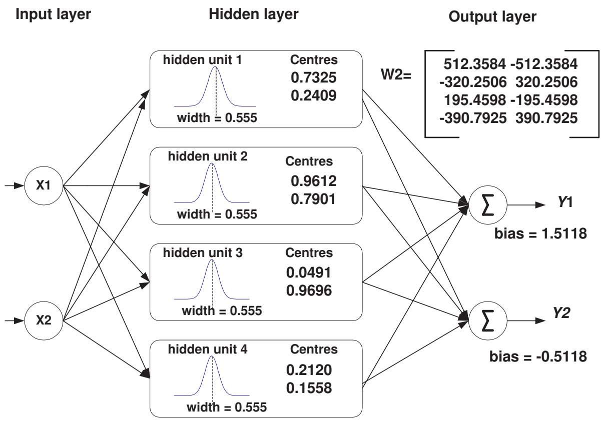

Figure 2: Parameters for RBF network trained on noisy Xor

transfers the input vector to the hidden units, which form a localized response to the input pattern. Learning is normally undertaken as a two-stage process. The first stage consists of an unsupervised process in which the RBF centres (hidden units) are positioned and the optimum field widths are determined in relation to the training samples.

The second stage of learning involves the calculating the hidden unit to output unit weights and is achieved quite easily through a simple matrix transformation.

The radial basis functions in the hidden layer are implemented by kernel functions, which operate over a localized area of input space. The effective range of the kernels is determined by the values allocated to the centre and width of the radial basis function. The Gaussian function has a response characteristic determined by equation 1.

$$Z_j(x) = exp\left(-\frac{||x-\mu||^2}{\sigma_j^2}\right) \tag{1}$$

The response of the output units is calculated quite simply using equation 2.

$$\sum_{j=l}^{J} W_{lj} Z_j(x) \tag{2}$$

where:
W = weight matrix, Z = hidden unit activations,
x = input vector, μ = n-dimensional parameter vector,
σ = width of receptive field.

3.1 RBF training

The first stage was to train RBF networks to an acceptable level of accuracy on all data sets. The specific level of accuracy varied with each data set, the literature was examined to provide guidance on what accuracy levels could be achieved. The accuracy levels stated in the tables are the best out of up to 10 test runs. Training of the RBF networks required the setting of three parameters, the global error, the spread or width of the basis function and the maximum number of hidden units. The value assigned to the global error setting may result in fewer hidden units being used than the maximum value. If the error value is not reached, training will terminate when the maximum number of hidden units has been assigned. The training and test data for the construction of the RBF networks were generally split 75/25.

3.2 Data Sets

In order to allow good benchmarking and comparison we used a mixture of well known benchmark data as well as two new vibration data sets for our tests. The data sets were selected from various sources but mainly obtained from the collection maintained by the University of California at Irvine (UCI). The vibration data sets were produced as part of two large projects which were concerned with the monitoring the health of industrial machinery. The data sets represent a variety of synthetic and real world problems of varying complexity (i.e. number of examples, input features and classes).

Table 1: Composition of data sets used in experimental work

Data Set	**Ex**	**O/P**	**I/P**	**C**	**D**	**M**
Xor(binary)	4	2	2	No	Yes	No
Xor(continuous)	100	2	2	Yes	No	No
Iris	150	3	4	Yes	No	No
Vowell(Peterson)	1520	10	5	Yes	Yes	No
Vowell(Deterding)	990	11	11	Yes	Yes	No
Protein(yeast)	1484	10	8	Yes	No	No
Protein(ecoli)	336	8	8	Yes	No	No
Credit(Japanese)	125	2	9	Yes	Yes	Yes
Credit(Australian)	690	2	15	Yes	Yes	Yes
Diabetes(Pima)	768	2	8	Yes	No	No
Monks1	556	2	6	No	Yes	No
Sonar	208	2	60	Yes	No	No
Vibration 1	1028	3	9	Yes	No	No
Vibration 2	1862	8	20	Yes	No	No

Table 1 gives details of the data sets. The columns indicate the number of examples, the number of ouput features or classes, the number of input features, whether the data set contains continuous data or discrete data and the last column indicates if any data is missing.

4 LREX: Rule Extraction Algorithm

The development of the LREX algorithm was motivated by the local architecture of RBF networks which suggested that rules with unique characteristics could be extracted. In addition, there was little published work on extracting rules from ordinary RBF networks [Lowe, 1991]. Therefore our work fills a substantial gap in rule extraction research.

The LREX algorithm is composed of two modules: the *m*REX module extracts IF..THEN type rules based on the premise that a hidden unit can be uniquely assigned to a specific output class. Therefore, by using the centre locations of the hidden units an input vector could be directly mapped to an output class. Experimental work performed on the simpler data sets tended to reinforce this belief. However, hidden unit sharing occurs within networks trained on non-linear or complex data. This phenomena reduces rule accuracy as several hidden units may be shared amongst several classes. The second module, *h*REX was developed to identify which hidden units are shared between classes. Analysis of how each hidden unit contributes provides information to determine a class. The extracted rules are IF..THEN type rules where any given hidden may appear across several classes. The next two sections describe how the *m*REX and *h*REX modules provide the user with complimentary types of extracted rules that explain the internal operation of the original RBF network.

4.1 *m*REX: Input-to-output mapping

The functionality of *m*REX algorithm is shown in figure 3.

Input:
    Hidden weights μ (centre positions)
    Gaussian radius spread σ
    Output weights $W2$
    Statistical measure S
    Training patterns
Output:
    One rule per hidden unit
Procedure:
    Train RBF network on data set
    Collate training pattern *"hits"* for each hidden unit
    For each hidden unit
        Use $W2$ correlation to determine Class label
        Use *"hits"* to determine S
        Select S format $\{min, max, std, mean, med\}$
        For each μ_i
            $X_lower = \mu_i - \sigma_i * S$
            $X_upper = \mu_i + \sigma_i * S$
    Build rule by:
        antecedent $= [X_lower; X_upper]$
        Join antecedents with AND
        Add Class label
    Write rule to file

Figure 3: *m*REX rule-extraction algorithm

The first stage of the *m*REX algorithm is to use the W2 weight matrix (see figure 2) to identify the class allocation of each hidden unit. The next stage is to calculate the lower and upper bounds of each antecedent by adjusting the centre weights μ using the Gaussian spread σ . The lower and upper limits are further adjusted using a statistical measure S gained from the training patterns classified by each hidden unit. S is used empirically to either contract or expand each antecedents range in relation to the particular characteristics of these training patterns.

The entire rule set for the Iris domain is presented in figure 4. Note that there are four extracted rules, one for each RBF hidden unit.

Rule 1 :
IF (SepalLength $\geq$ 4.1674 AND $\leq$ 5.8326) AND
IF (SepalWidth $\geq$ 2.6674 AND $\leq$ 4.3326) AND
IF (PetalLength $\geq$ 0.46745 AND $\leq$ 2.1326) AND
IF (PetalWidth $\geq$ 0.53255 AND $\leq$ 1.1326)
THEN..Setosa

Rule 2 :
IF (SepalLength $\geq$ 5.2674 AND $\leq$ 6.9326) AND
IF (SepalWidth $\geq$ 1.9674 AND $\leq$ 3.6326) AND
IF (PetalLength $\geq$ 3.1674 AND $\leq$ 4.8326) AND
IF (PetalWidth $\geq$ 0.46745 AND $\leq$ 2.1326)
THEN..Versicolor

Rule 3 :
IF (SepalLength $\geq$ 5.9674 AND $\leq$ 7.6326) AND
IF (SepalWidth $\geq$ 2.3674 AND $\leq$ 4.0326) AND
IF (PetalLength $\geq$ 5.0674 AND $\leq$ 6.7326) AND
IF (PetalWidth $\geq$ 1.4674 AND $\leq$ 3.1326)
THEN..Virginica

Rule 4 :
IF (SepalLength $\geq$ 4.8674 AND $\leq$ 6.5326) AND
IF (SepalWidth $\geq$ 1.6674 AND $\leq$ 3.3326) AND
IF (PetalLength $\geq$ 4.1674 AND $\leq$ 5.8326) AND
IF (PetalWidth $\geq$ 1.1674 AND $\leq$ 2.8326)
THEN..Virginica

Figure 4: *m*REX extracted rules from Iris domain

4.2 *h*REX: Hidden unit analysis

A different approach to rule extraction is taken by the *h*REX algorithm which uses quantization and clustering on the network parameters (weights and activation levels) to form an abstraction of its operation. The number of extracted rules is determined by the user who can place an upper limit on the rules extracted for each class. This is a useful feature since it enables a tradeoff to be made between rule size and rule comprehensibility. This is achieved by three important parameters:

- α which determines the minimum weight value (postive) to be quantized as a "one", weights below this cutoff point are quantized to -1 and do not participate in rule extraction.
- β which determines the minimum hidden unit activation level. Hidden units with activation levels below this cutoff point will not be quantized and will play no further part in rule extraction.
- N determines the maximum number of clusters that the training set (for each class) is divided into. This process abstracts the input space into a number of distinct regions which will require a separate rule to identify.

These parameters are determined empirically for a satisfactory arrangement. Figure 5 shows the algorithm in detail. Note that valid rules consist of both a positive quantized weight (QW2) and a positive quantized activation (AQZ) level. A rule consists of one or more hidden units which must all be active for the class lable to be satisified.

Input:
    Output weights $W2$
    Hidden unit activations Z (training data)
    Output weights quantization modifier α
    Hidden unit activation quantization modifier β
    Maximum Cluster number N
    Training patterns by sorted by class T
Intermediate information:
    Quantized W2 weights $QW2$
    Quantized hidden unit activations QZ
    Average Quantized hidden unit activations AQZ
Output:
    One rule per cluster
Procedure:
    Quantize W2 weights with α
    Quantize hidden unit activations Z with β
    Separate training patterns by class T
    For each class
        Partition QZ up to NC Clusters
        For each N Cluster
            Identify Positive QZ activations
            Calculate Average AQZ value for cluster
            Identify Positive $QW2$ weights attached to QZ
        Build rule by:
            IF AQZ==Positive AND QW2 ==Positive
                Hidden unit H belongs to rule
                Join Hidden Units with AND
                Add Class label
        Write rule to file

Figure 5: *h*REX rule-extraction algorithm

Some *h*Rules rules extracted from the ecoli domain are presented in figure 6. For instance, for Rule 4 to "fire", each antecedent must be satisfied so hidden units 5, 19, 20, 24, 25, 26, 28 and 31 must all be active. It can be seen that hidden unit 20 participates in both class 3 and class 4.

*h*REX rules are useful for identifying the internal structural relationships formed by the hidden units. This is demon-

Rule ♯9 Class: 3
IF((H5 == TRUE) AND
(H19 == TRUE) AND
(H20 == TRUE) AND
(H24 == TRUE) AND
(H25 == TRUE) AND
(H26 == TRUE) AND
(H28 == TRUE) AND
(H31 == TRUE))
THEN
Class: 3

Rule ♯10 Class: 4
IF((H12 == TRUE) AND
(H20 == TRUE) AND
(H22 == TRUE) AND
(H23 == TRUE) AND
(H32 == TRUE))
THEN
Class: 4

Figure 6: *h*REX extracted rules from ecoli domain

strated on those RBF networks that have a poor performance on certain classes. These RBF networks produce *h*REX rules which exhibit a large degree of hidden unit sharing or in the worse cases fail to generate any *h*REX rules for these classes.

Figure 7 shows the accuracy of the *h*REX rules against the rule size (comprehensibility) for RBF networks trained on the Vibration 1, Monks and Sonar data sets. The Vibration shows a steady increase in accuracy with each additional rule until it levels off at a cluster size of 12. The rules extracted from Sonar actually lose accuracy beyond a certain point before the accuracy reaches a steady value. Generating additional rules for the Monks after the optimum cluster size is reached produces an oscillating effect where the accuracy does not level off.

5 Analysis of Results

The performance of the RBF rule extraction algorithm was compared with a related system called MCRBP/RULEX which was developed by Andrews and Geva [Andrews and Geva, 1999]. MCRBP builds RBF-like networks with specialized activation functions. Once the networks are trained, the RULEX algorithm can then be used to extract IF..THEN rules with boundaries. The rules extracted by RULEX are in a very similar format to those produced by the author's system. Table 2 shows the results of the experimental work. The first column identifies the data set. The second column presents the *mREX* accuracy alongside the original RBF accuracy. The third column details the *h*REX accuracy next to the original RBF accuracy and the fourth column shows the Rulex accuracy

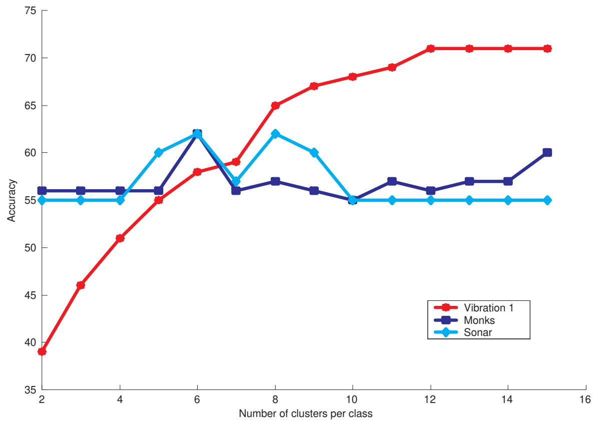

Figure 7: hREX rule size and complexity

Table 2: Comparison between RBF net, mREX, hREX and Rulex accuracy

Data set	***m*REX**	***h*REX**	**Rulex**
Xor(binary)	100/100	100/100	100
Xor(continuous)	96/100	100/100	100
Iris	93/96	93/96	100
Vowell(Peterson)	43/86	22/86	–
Vowell(Deterding)	9/62	20/62	38
Protein(yeast)	26/57	66/57	28
Protein(ecoli)	49/87	72/87	88
Credit(Japanese)	73/93	66/93	93
Credit(Australian)	66/71	64/71	88
Diabetes(Pima)	65/76	70/76	69
Monks1	79/83	60/83	72
Sonar	57/95	58/95	–
Vibration 1	56/73	69/73	61
Vibration 2	73/94	72/94	–

Table 3 shows the number of rules generated by the three systems. The rule set size quoted for LREX is based on the unmodified basic version.

RULEX extracts highly compact rule sets compared with LREX. The majority of the domains can be represented with as few as 3-5 rules. Unfortunately, RULEX completely failed to generate rules for three of the domains. This problem was tracked down to the initial MCRBP network, as it was unable to form a viable classifier on the training data. Therefore, any rules extracted would be invalid. RULEX also failed to provide rules to cover a specific class in the vibration 1 domain. Training the MCRBP networks took fewer attempts to reach acceptable accuracies than the equivalent RBF networks (typically 2-3 runs).

MCRBP/RULEX could not form a viable network on the vowel, sonar and vibration 2 domains. It is likely that the specialized architecture cannot cope with the large number of

Table 3: Comparison between rule set size of mREX, hREX and Rulex

Data set	mREX	hREX	Rulex
Xor(binary)	4	4	4
Xor(continuous)	4	4	4
Iris	4	4	5
Vowell(Peterson)	30	80	–
Vowell(Deterding)	200	110	11
Protein(yeast)	120	24	9
Protein(ecoli)	35	24	9
Credit(Japanese)	50	6	2
Credit(Australian)	50	20	5
Diabetes(Pima)	300	11	3
Monks1	20	24	3
Sonar	20	10	–
Vibration 1	30	25	2
Vibration 2	100	32	–

input features present in these data sets. However, by using non-overlapping local functions the MCRBP/RULEX algorithm can form a rule from each function that is specific to a class. This requires fewer rules to form a classifier.

The *h*REX algorithm produces fewer rules than the *m*REX algorithm and are generally more accurate. A smaller rule set enables a better understanding of the internal operation of the RBF network. further analysis of the *h*REX rules proved to be interesting as several of the RBF networks have up to 35-40% of their hidden units shared between the various output classes. Such results tend to occur with those RBF networks that have lower accuracies and may implie that the original settings of the internal parameters during training were not optimal e.g. a badly chosen value for the width of the basis function can be a source of error.

6 Conclusions

The work described in this paper has tackled the difficult issue of knowledge extraction from RBF networks which has been avoided in the literature because of the problems with overlapping neurons. The rules extracted by the LREX algorithm provide information about the original RBF network in two forms; an input to output mapping and information regarding those hidden units that participate in classification. The knowledge extracted by the *m*REX algorithm transforms the original RBF network into a rule based classifier. This makes the input to output mapping of the RBF network transparent and open to scrutiny. However, the number of rules produced is dependent on the number of hidden units and therefore a large number of rules may obscure the comprehensibility. This problem is partially solved by the *h*REX algorithm which can generate a maximum number of rules determined in advance by the user. The tradeoff is rule size (and generally accuracy) versus comprehensibility. Some RBF networks may naturally be described by small rule sets that are accurate but still allow a good understanding of their internal structure. Other RBF networks may have modeled complex functions and their hidden units are used by several classes, in which case the *h*REX algorithm will provide useful information regarding the extent of this activity.

References

[Andrews and Geva, 1999] R. Andrews and S. Geva. On the effects of initialising a neural network with prior knowledge. In *Proceedings of the International Conference on Neural Information Processing (ICONIP'99)*, pages 251–256, Perth, Western Australia, 1999.

[Bishop, 1995] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[Craven and Shavlik, 1997] M. Craven and J. Shavlik. Using neural networks for data mining. *Future Generation Computer Systems*, 1997.

[Lowe, 1991] D. Lowe. On the iterative inversion of RBF networks: a statistical interpretation. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 29–33, Bournemouth, UK, 1991.

[McGarry *et al.*, 1999] K. McGarry, S. Wermter, and J. MacIntyre. Hybrid neural systems: from simple coupling to fully integrated neural networks. *Neural Computing Surveys*, 2(1):62–93, 1999.

[Moody and Darken, 1989] J. Moody and C. J. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, pages 281–294, 1989.

[Omlin and Giles, 1994] C. W. Omlin and C. L. Giles. Extraction and insertion of symbolic information in recurrent neural networks. In V.Honavar and L.Uhr, editors, *Artificial Intelligence and Neural Networks:Steps Towards principled Integration*, pages 271–299. Academic Press, San Diego, 1994.

[Shavlik, 1994] J. Shavlik. A framework for combining symbolic and neural learning. *Machine Learning*, 14:321–331, 1994.

[Sun, 2000] R. Sun. Beyond simple rule extraction: the extraction of planning knowledge from reinforcement learners. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, Lake Como, Italy, 2000.

[Thrun, 1995] S. Thrun. Extracting rules from artificial neural networks with distributed representations. In G.Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*. MIT Press, San Mateo, CA, 1995.