International Journal of Computational Intelligence and Applications

THE EXTRACTION AND COMPARISON OF KNOWLEDGE 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, England
ken.mcgarry@sunderland.ac.uk

Received (received date)
Revised (revised date)

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.

Keywords: Rule Extraction, Radial Basis Functions, Neural Networks, Clustering, Local Functions

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 [3]. 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 [15].

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 [16,17]. 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.

This paper is a revised and extended version of our IJCAI-01 paper [12]. The remainder of 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 [13] 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 [18], Kohonen networks [19] and recurrent networks [14]. 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 [4].

In addition to providing an explanation facility, rule extraction is recognised as a powerful technique for neuro-symbolic integration within hybrid systems [11]. For a general discussion on the practicalities and issues of rule extraction see Andrews [1].

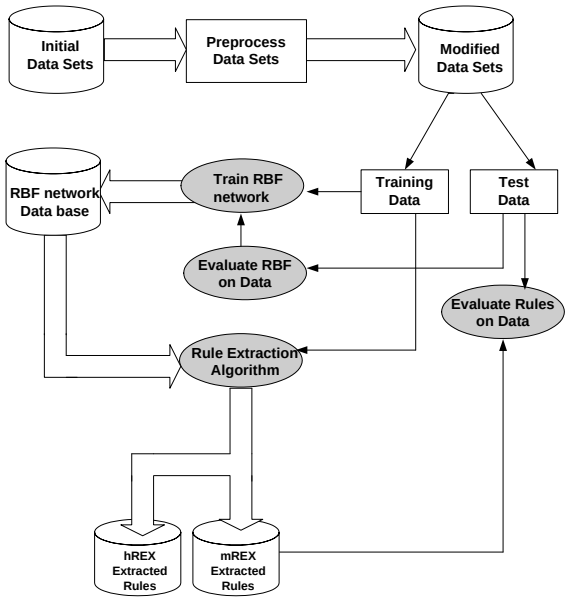

Fig. 1. Knowledge extraction system data flow and data transformation

3. Radial Basis Function Networks

Radial basis function (RBF) neural networks are a class of 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 [13]. 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 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 calculation of 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.

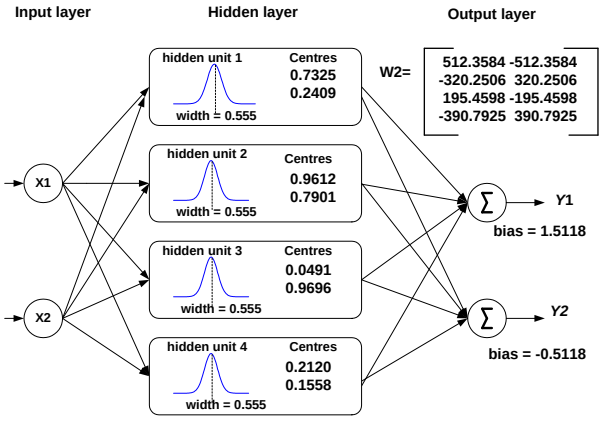

Fig. 2. Parameters for RBF network trained on noisy XOR

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

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

50/50. The training algorithm added a hidden unit at every epoch to reduce the global error. The location of the hidden unit corresponded to the training vector with the largest error. This technique avoided the need to specify the number of hidden units in advance as required by alternative techniques such as Expectation Maximization (EM). The number of hidden units necessary for similar accuracy levels to EM was not noticeably larger. The only disadvantage is the requirement for several training runs to obtain a good accuracy. We do not use any techniques for pruning the RBF networks prior to extracting the symbolic rules.

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	Examples	Outputs	Inputs	Cont	Discrete	Missing
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 [8]. 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. 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 next two Sections describe how the *m*REX and *h*REX modules provide the user with complementary types of extracted rules that explain the internal operation of the original RBF network.

4.1. *mREX: Input-to-output mapping*

The functionality of the *m*REX algorithm is shown in Figure 3. The first stage of

---

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

Fig. 3. *m*REX rule-extraction algorithm

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

Fig. 4. *m*REX extracted rules from Iris domain

4.2. *hREX: 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. Figure 5 shows the algorithm in detail. 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.

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

Fig. 5. hREX rule-extraction algorithm

This is achieved by three important parameters:

- α which determines the minimum weight value (positive) 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.

The quantization process is intended to simplify the hidden unit rule extraction process. Quantization results in a "yes" or "no" decision that will either include or exclude a weight value from the rule extraction process. Negative weights can be removed as they indicate that an input feature or hidden unit does not actively

participate in the classification process i.e. they do not provide useful information.

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. 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.

```
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
```

Fig. 6. *h*REX extracted rules from the Ecoli domain

*h*REX rules are useful for identifying the internal structural relationships formed by the hidden units. This is demonstrated 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 data set after the optimum cluster size has been reached produces an oscillating effect where the accuracy does not level off.

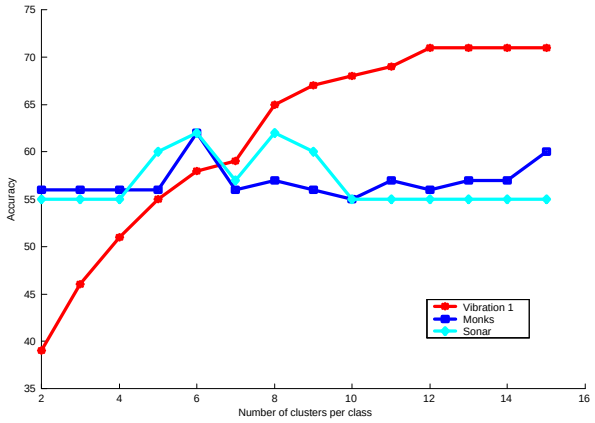

Fig. 7. hREX rule size and complexity

5. Analysis Of The 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 [2]. 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 as 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

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.

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	–

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 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 are associated with those RBF networks trained on data which has caused a high degree of overlap to occur between the hidden units. Overlap prevents the unique assignment of a hidden unit to a particular output class and is the major cause of rule inaccuracy.

The experimental results also revealed a number of disadvantages of rule extraction from networks with local representations: Sensitivity to non-discriminatory input features, which may lead to a high degree of overlap occurring with the hidden units. Figure 8 shows how in the left hand diagram the input space is composed of

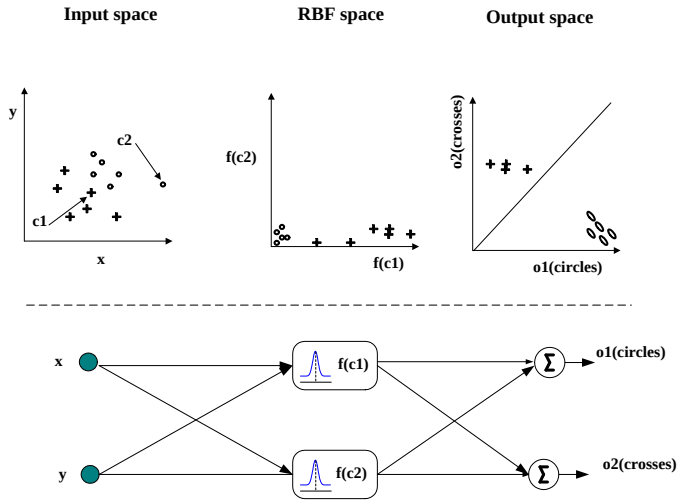

Fig. 8. Relevance of attributes in input space

two dimensions with two classes. The middle diagram shows the input space transformed nonlinearly into RBF space which is normally of a higher dimension but is approximately linearly separable. It can be seen that RBF unit f(c2) shows little response to the input data while the first RBF unit f(c1) is very responsive. The weights from f(c2) to the output unit will tend to be very small or even negative. The right hand diagram shows the output unit space where the class separation is given by 01=02 line i.e the class label is decided by the largest output unit value. The RBF network is positioned under the three graphs to highlight the functionality of the transformations. See Kubat for a further explantion [7].

The lack of accuracy in certain domains has prevented the data mining potential of the extracted rules from being fully realized. However, the providers of the Vibration data set found the extracted rules to be quite illuminating in terms of highlighting input to output mappings. To be sure, many of the rules simply confirmed well understood relationships in the original data. However, the rules provided "interesting" information on the less well known input parameters. Prior to the rule extraction experiments these parameters were simply extra input features incorporated into the neural network training data. A manual examination of the rule set by the domain experts indicated some useful ratios between the lower and upper antecedent values [10].

6. Conclusions

The work described in this paper has tackled the difficult issue of knowledge extraction from RBF networks. 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.

In terms of novelty and new knowledge, the rule extraction algorithms have enabled a symbolic analysis to be performed on standard Gaussian RBF networks. Previous work has concentrated on specialized architectures or have modified the hidden units to avoid overlap such as the rectangular basis functions of Huber [6]. The *m*REX rules provide an input to output mapping which can be used to determine why a particular input vector will be assigned to a given class. Also, the internal representation formed by the hidden units is made transparent. The activity of the hidden layer is revealed as the hidden units can be identified as either cooperating or working singly to form a classification. We are currently investigating techniques to manage the effects of overlapping hidden units. Fuzzy logic appears to be a suitable candidate as it offers the benefits of maintaining rule comprehensibility and accuracy while maintaining a similar format. In addition, it will be useful to understand how rules can be extracted from basis functions other than the Gaussian. The thin-plate spline functions for example have characteristics between the local Gaussian and the global sigmoid function [9]. Such functions may provide the platform to extract rules that are more descriptive, suffer less from large input dimensionality and from superfluous inputs. In addition, the extracted rule sets would certainly benefit from some means of providing a summary of the global trends. Work by Gaines [5] suggests the use of exception-directed acyclic graphs (EDAGs) which he developed for simplifying decision trees and production rules.

References

1. R. Andrews, J. Diederich, and A. Tickle. The truth will come to light:directions and challenges in extracting the knowledge embedded within trained artificial neural networks. *IEEE Transactions on Neural Networks*, 9(6):1057–1068, 1998.

2. 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.
3. C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
4. M. Craven and J. Shavlik. Using neural networks for data mining. *Future Generation Computer Systems*, 1997.
5. B. Gaines. Transforming rules and trees. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthursamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 205–226. AAAI-Press, 1996.
6. K. P. Huber and M. R. Berthold. Building precise classifiers with automatic rule rule extraction. In *Proceedings of the International Conference on Neural Networks*, volume 2, pages 117–120, 1995.
7. M. Kubat. Decision trees can initialize radial basis function networks. *IEEE Transactions on Neural Networks*, 9(5):813–821, 1998.
8. 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.
9. D. Lowe. On the use of nonlocal and non positive definite basis functions in radial basis function networks. In *Proceedings of the 3rd International Conference on Artificial Neural Networks*, pages 206–211, Cambridge, UK, 1995.
10. K. McGarry and J. MacIntyre. Data mining in a vibration analysis domain by extracting symbolic rules from RBF neural networks. In *Proceedings of 14th International Congress on Condition Monitoring and Engineering Management*, pages 553–560, Manchester, UK, 4th-6th September 2001.
11. 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.
12. K. McGarry, S. Wermter, and J. MacIntyre. Knowledge extraction from local function networks. In *Seventeenth International Joint Conference on Artificial Intelligence*, Seattle, USA, August 4th-10th 2001.
13. J. Moody and C. J. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, pages 281–294, 1989.
14. 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.
15. A. Roy, S. Govil, and R. Miranda. An algorithm to generate radial basis function (RBF)-like nets for classification problems. *IEEE Neural Networks*, 8(2):179–201, 1995.
16. J. Shavlik. A framework for combining symbolic and neural learning. *Machine Learning*, 14:321–331, 1994.
17. 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*, pages 105–110, 2000.
18. 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*, pages 505–512. MIT Press, San Mateo, CA, 1995.
19. A. Ultsch, R. Mantyk, and G. Halmans. Connectionist knowledge acquisition tool: CONKAT. In J. Hand, editor, *Artificial Intelligence Frontiers in Statistics: AI and statistics III*, pages 256–263. Chapman and Hall, 1993.