CLASSIFICATION USING NETWORKS OF NORMALIZED RADIAL BASIS FUNCTIONS.

Guido Bugmann, Centre for Neural and Adaptive Systems,
School of Computing, University of Plymouth, Plymouth PL4 8AA.

ABSTRACT

Normalized Radial Basis Function Networks (NRBF) were invented at the same time as standard RBF nets, in 1989, but went unnoticed until recently, when it was found that they constitute a very interesting tool, especially for pattern classification. NRBF classifiers behave as Nearest Neighbour classifiers and have a functionality similar to Fuzzy Inference Systems but without the Curse of Dimensionality problem. NRBF nets are easy to use, with performances largely insensitive to model parameters and with a very fast training. In this paper the functionality of NRBF nets is compared with those of standard RBF nets. The applications to classification will be illustrated with benchmark problems. For the IRIS classification problem, NRBF match the performances of the best published techniques.

1. INTRODUCTION.

Normalized Radial Basis Functions (NRBF) differ from standard Radial Basis Functions (RBF) by a seemingly minor modification of their equation (section 2). This results in novel computational properties which have attracted little attention in the neural network community. Moody and Darken (1989)[16] were first to mention Normalised RBF nets without elaborating on their functional significance. However. Servin and Cuevas (1993)[20] noted that normalization gave RBF nets the "same classification properties as nets using sigmoid functions". Cha and Kassam (1995)[9] proposed that "a normalized Gaussian basis function features either localized behavior similar to that of a Gaussian or nonlocalized behavior like that of a sigmoid, depending on the location of its centre". Rao et al. (1997)[19] interpreted NRBF nets as mixture of expert models and Jang & Sun (1993)[12] saw similarities with fuzzy inference systems. These multiple views reflect the fact that NRBF nodes in the hidden layer behave more like case indicators rather than basis functions proper, as is elaborated in section 2. This property leads to excellent performances in classification tasks, as shown in section 4. One of the key features of NRBF nets is their excellent generalization, a property that can be exploited to reduce the number of hidden nodes in classification tasks. A new learning rule was proposed to that effect by Bugmann (1998)[5] and is summarized in section 3.

NRBF nets have also given very good results in other classes of application, trajectory learning in robotics (Althoefer and Bugmann, 1995[1], Bugmann et al, 1998[6]) and function mapping (Bugmann, 1998[5]).

2. NORMALIZED RADIAL BASIS FUNCTION NETWORKS

Standard Radial Basis Functions (RBF) nets comprise a hidden layer of RBF nodes and an output layer with linear nodes (Broomhead and Lowe, 1988[7], Moody and Darken, 1989[16]). The function of these nets is given by:

$$y_i(x) = \sum_{j=1}^{n} w_{ij}\, \phi(x - x_j) \qquad (1)$$

where y_i is the activity of the output node i, $\phi(\boldsymbol{x}-\boldsymbol{x}_j)$is the activity of the hidden node j, with a RBF function centred on the vector $\boldsymbol{x}_j$, $\boldsymbol{x}$ is the actual input vector and w_{ij} are the weights from the RBF nodes in the hidden layer to the linear output node. Such a net is a universal function approximator according to Powell (1987)[17].

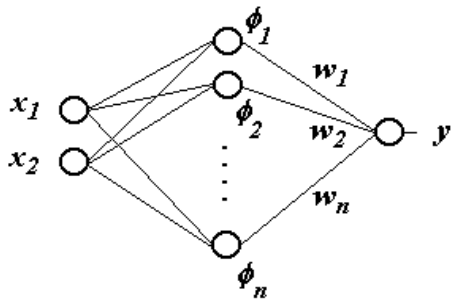


Figure 1: Network architecture for standard RBF nets and Normalized RBF nets (two inputs and one output shown).

The RBF function $\phi(\boldsymbol{x}-\boldsymbol{x}_j)$ of a hidden node j used here is the Gaussian Radial Basis Function:

$$\phi(x - x_j) = \exp\left(-\frac{\sum_{k=1}^{K}(x_k - w_{jk})^2}{2\sigma^2}\right)$$

where σ is the width of the Gaussian and K is the dimension of the input space. The "weights" w_{jk} between node k in the input layer and node j in the hidden layer do not act multiplicatively as in other neuron models, but define the input vector $\boldsymbol{x}_j = (w_{j1},...,w_{jK})$ eliciting the maximum response of node j ($\boldsymbol{x}_j$ *is* the "centre of the receptive field").

In normalised RBF nets, the output activity is normalised by the total input activity in the hidden layer:

$$y_i(x) = \frac{\sum_j W_{ij}\, \phi(x - x_j)}{\sum_j \phi(x - x_j)} \qquad (2)$$

Moody and Darken (1989)[16] proposed that normalisation be performed by the hidden nodes before the summation stage in the output node. In their approach, normalization is a non-local operation, requiring each hidden node to "know" about the outputs of the other hidden nodes. Hence a computationally costly convergence process is required. A similar view is taken in (Rao et al., 1997[19]). In contrast, in our implementation the normalisation is done in the output layer. As it receives already information from all hidden units, the locality of the computational processes is preserved.

Equation 2 shows that, as a result of the normalization, the output activity becomes an activity-weighted average of the input weights in which the weights from the most active inputs contribute most to the value of the output activity. In other words, the roles of output weights and hidden nodes activities are switched. In standard RBF nets, the weights determine how much each hidden node contributes to the output. In NRBF nets, the activity of the hidden nodes determine which weights contribute most to the output. For instance, in the

extreme case where only one of the hidden nodes is active, then the output of the net becomes equal to the weight corresponding to that hidden node, whatever its level of activity.

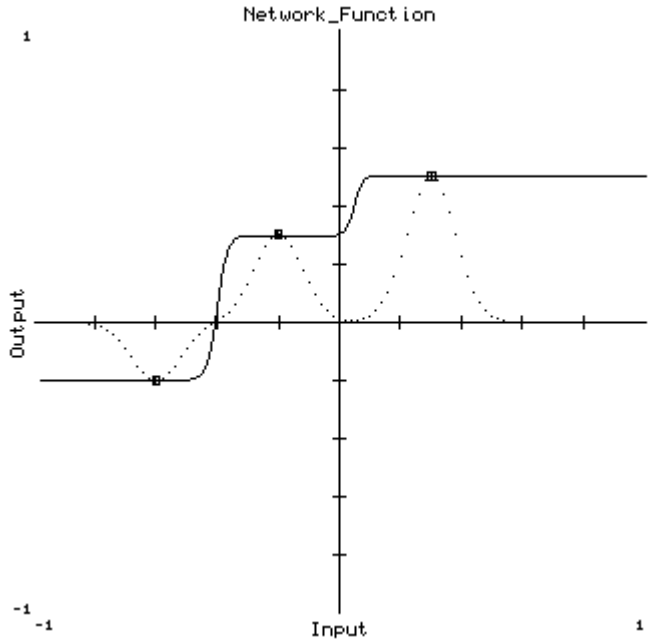


Figure 2. Illustration of the effects of normalization in RBF nets. *Full line:* output *y* of a NRBF net. *Dotted line:* Output of a standard RBF net. Both nets use three hidden nodes centred on the three training data indicated in the Figure.

Figure 2 shows that hidden nodes have a domain of influence in the input space in which they determine the output of the net. This domain is only limited by conflicts with other hidden nodes and has no limits outside of the domain covered by training data. This explains the "sigmoid-like" behaviour noted by Servin and Cuevas (1993)[20] and Cha and Kassam (1995)[9].

When the size of the Gaussians is small, the decay of the hidden nodes activity with distance is so fast that, for most point of the input space, there is usually only one hidden node that contributes significantly to equation 2. As a result, hidden nodes perform a parcelation of the input space similar to a Voronoi tessellation. This is also illustrated by a two-dimensional example in section 4. Due to normalization, RBF nodes become here case indicators rather than basis functions proper. In that sense, NRBF nets are similar to fuzzy inference systems, as discussed in (Jang & Sun, 1993[12], Andersen et al., 1998[2]). In these systems normalization is a key element of the "centre of gravity defuzzification method (Brown and Harris, 1994, pp 388-404[8]).

In equation 2, normalization is a way to select which weight becomes the output of the net. This is a special case of a more general approach where whole functions are selected. For instance in (Shao et al., 1993[21]) the output y_i is a combination of linear functions $L_{ij}(\mathbf{x})$ weighted by the activity of their respective hidden nodes *j*.

Two types of networks are compared in this paper: a standard RBF net, as in equation (1) and a net with a normalizing output node as in equation (2). In both nets the hidden nodes have Gaussian receptive fields, with a width σ indicated in figure captions. The networks (Figure 1) have two inputs, one output and a number of hidden nodes determined by the

recruitment procedures during training (section 3). Simulations are done on a PC with the neural network development package CORTEX-PRO[1].

3. LEARNING PROCEDURES.

3.1 STANDARD TRAINING PROCEDURE.

For the standard RBF nets, training is done in a standard way (Bishop, 1995, p. 170[4]), by recruiting hidden nodes in the first epoch, then, in subsequent epochs, adjusting the positions of the centres of the nodes and the weights to the output node to minimize the output error, as described below:

i) recruiting a new hidden node centred on an input vector that was beyond a radius of 0.5σ from the centre of an existing node, or slowly shifting the centre $\boldsymbol{x}_j$ of an existing hidden node towards the new vector $\boldsymbol{x}$ using: $\boldsymbol{x}_j(t+1) = 0.8\boldsymbol{x}_j(t) + 0.2\boldsymbol{x}$,

ii) modifying the output weight of the hidden nodes j within a radius 0.5σ of the current input vector $\boldsymbol{x}$ so that the output y_i becomes closer to the desired output y_{id} , using: $w_{ij}(t+1) = w_{ij}(t) + learnrate\ (y_{id} - y)$ (typically, $learnrate = 0.5$).

iii) showing the input vectors repeatedly to the net.

In this procedure, the recruitment of hidden node is purely input-driven, as it depends only on the distribution of training data in the input space. It is a simple procedure that converges rapidly. The number of epoch is indicated in figure captions.

3.2 MODIFIED TRAINING PROCEDURE.

For training the normalized RBF nets, the modified procedure (Bugmann, 1998[5]) is used. Taking advantage of the good interpolation and extrapolation properties of NRBF nets, new nodes are recruited only in crucial points, close to boundaries between two classes. This is achieved by recruiting no new nodes if the network indicates the correct class by using existing nodes. The steps of the modified procedure are:

i). Check if the output vector $\boldsymbol{y}$ of the net is correct, i.e. $|\boldsymbol{y}-\boldsymbol{y}_{desired}| < tolerance$, where a $tolerance = 0.4$ is adequate for classification problems (it may be reduced to much smaller values for mapping problems, where it represents the desired accuracy).

ii). If the output is correct. Do nothing, go to the next training data.

iii). If the output is incorrect and there is no node with its centre close to the current data point, recruit a new node centred on the data point and set its output weights to the desired output vector.

iv). If the output is incorrect and there is a node with centre close to input vector $\boldsymbol{x}$, move its centre $\boldsymbol{x}_j$ closer to the new vector $\boldsymbol{x}$ using: $\boldsymbol{x}_j(t+1) = 0.8\boldsymbol{x}_j(t) + 0.2\boldsymbol{x}$, and modify its output weights using: $w_{ij}(t+1) = w_{ij}(t) + learnrate \cdot (y_{id} - y_i)$.

---

[1]Unistat Ltd, Unistat House, 4 Shirland Mews, London W9 3DY, UK. http://www.unistat.com

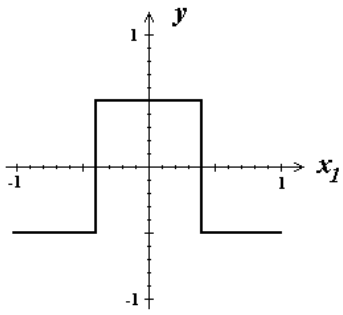
The main difference of this procedure with the standard one is that the recruitment of hidden nodes is also output-driven. It results in progressively less new nodes being recruited at each epoch, as the coverage of the class domains is refined. Generally, after 3 epochs, recruitment ceases and performance improvements are solely due to weights and centre modifications. It is not a optimal procedure, because the location of the centres of the recruited nodes depends on the order of presentation of the training data, but this is of no consequences in most applications.

The optimal width of the Gaussian basis functions differs for Normalized and standard RBF nets, and can be estimated by a simple calculation discussed in section 4.1.

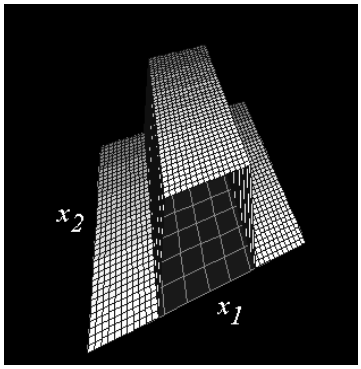
4. CLASSIFICATION WITH STANDARD AND NORMALIZED RBF NETS.

4.1 THE PLATEAU-VALLEY CLASSIFICATION PROBLEM

A simple example is used here to illustrate the difference between normalized RBF net and standard RBF nets. In this example, a function $f(x_1,x_2)$ is used to divide the input space into two regions: a "plateau" where the $f(x_1,x_2) = 0.5$ and "valleys" where $f(x_1,x_2) = -0.5$ (Figure 3). The problem is to train a network that classifies the regions of the input space by using 70 data points picked at random from the function $f(x_1,x_2)$. The location of most of these points can be inferred from Figure 4.



A)



B)

Figure 3. "Plateau" function $y = f(x_1,x_2)$ used in this example. The data are sampled in the domain $-1<x_1<1$, $-1<x_2<1$. The output $y = -0.5$ everywhere except for $-0.4<x_1<0.4$, where $y = 0.5$. **A)** Cross-section of the function along the plane $\{x_1, y\}$. **B)** Perspective view of the function in the sampling domain. The base grid is placed at y = -0.5.

Figures 4B and 5A show the functions learned by the standard RBF net and the NRBF net using a similar same small width for the basis functions. A striking effect of normalization is the improved interpolation. Even in regions of the input space where no RBF hidden node produces a strong response, NRBF nodes with receptive fields in surrounding regions can generate a large output value. In contrast, in standard RBF nets, a significant output requires a hidden node with its centre close to the input vector, or using hidden nodes with wider receptive fields (larger σ).

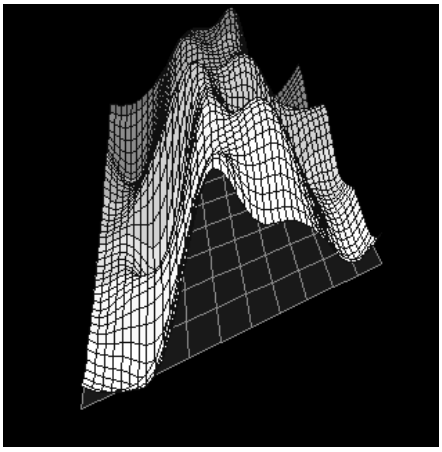
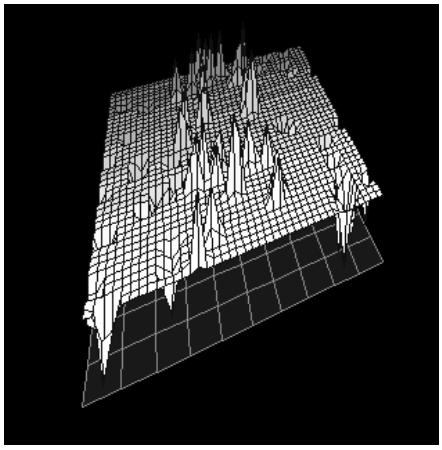
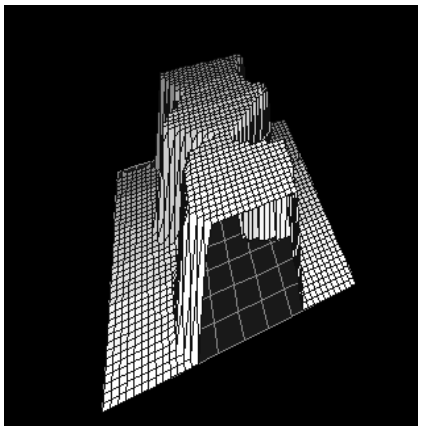
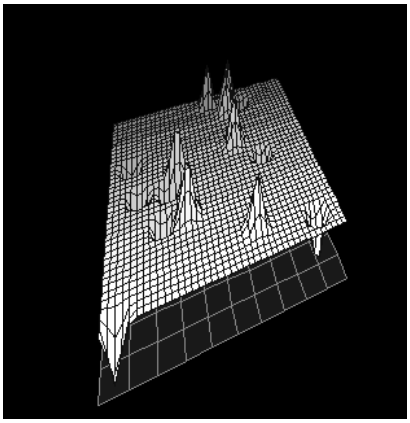
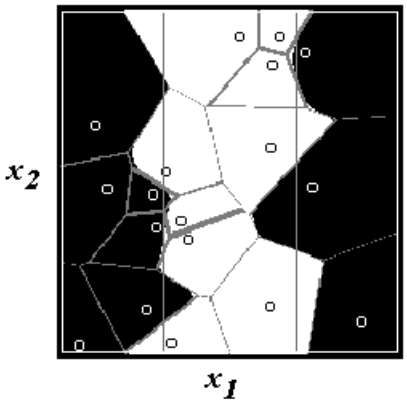

A)

B)

Figure 4: A) Function of a standard RBF net (equation 1) trained on 70 points sampled from the function shown in Figure 3. Parameters: $\sigma = 0.21$, Average RMS error < 0.04 after 30 epochs. 50 hidden nodes were recruited by the standard training procedure (section 3.1). **B)** Function of a standard RBF net with $\sigma = 0.03$. Figure produced for the purpose of indicating the positions of the 70 training data.


A)

B)



C)

Figure 5: A) Function of the Normalized RBF net trained with the modified procedure (section 3.2). *Parameters:* $\sigma = 0.05$, Average RMS error = 0.0041 after 2 epochs, 15 hidden nodes recruited. **B)** Non normalized output of the net, plotted to indicate the positions of the recruited nodes. **C)** Partitions of the input space generated by the recruited hidden nodes. The two vertical grey lines indicate the boundaries of the "plateau". Other grey lines indicate regions where neighbouring nodes have similar activities within a 20% margin.

It was shown in Bugmann (1998)[5] that for best interpolation, standard RBF nets require values of σ close to one half of the *largest* distance between nearest neighbours (which is here 0.46), reflecting the need to ensure large enough output values over the largest empty space between training data. Performances are better for larger receptive field sizes. In contrast, NRBF nets trained with the standard procedure show a good performance over a wide range of sizes, with better performances for small sizes. Interestingly, the best performance is achieved for a size $\sigma = 0.12$ which is exactly the *average* distance between a data point and its nearest neighbour in this training set. This size leads also to good generalization in mapping problems (Bugmann, 1998[5]). This points to a simple rule of thumb for an initial selection of the sizes of basis functions for NRBF nets.

Figure 5B shows the location of the recruited nodes. Comparison with Figure 4B indicates that recruited nodes are now mainly located at class boundaries, tending to form polarised pairs, with one member on each side of the boundary. This suggests that anti-symmetric basis functions reminiscent of the Gabor functions observed in visual cortex (Marcelja, 1980[15]; Jones & Palmer, 1987[13]), may constitute ideal oriented basis functions for classification problems with normalizing nets.

4.2 THE TWO-SPIRAL CLASSIFICATION PROBLEM.

Figure 7 illustrates the extrapolation properties of NRBF networks with the two-spirals benchmark problem. The network was trained with all 194 data points shown in Figure 7. It can be seen that areas outside of the immediate neighbourhood of training data are classified according to their nearest neighbour in the training data, or more precisely their nearest hidden node centre. This results in a function very similar to the one obtained with Multi-Layer Perceptrons using sigmoid functions in the hidden layers (Lang & Witbrock, 1988[14]). Indeed training is much faster here, with no classification errors after 4 epochs, and good generalisation may be inferred from Figure 7.

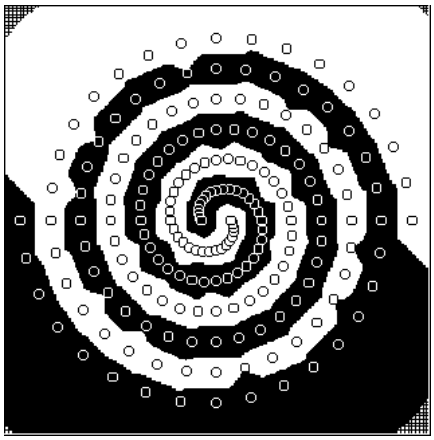


Figure 6: Two spirals classification. White and black areas indicate the two classes assigned to these areas by a Normalised RBF net. The hashed areas indicate unclassified areas due to activities of hidden nodes smaller than the precision of the computer. The 194 training data[2] are indicated by filled and empty symbols . Network parameters: $\sigma = 0.28$, Average RMS error = 0.057 after 4 epochs, 96 nodes recruited using the modified training procedure (section 4.1). Figure range: $-7 < x_1 < 7$, $-7 < x_2 < 7$.

[2] Data from the Carnegie Mellon AI repository: http://www-cgi.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/neural/bench/cmu/

4.3 THE IRIS BENCHMARK PROBLEM

NRBF nets have also been applied to the IRIS benchmark problem, matching the best published performances (Grant, 1997[10]). These results are summarized below.

Despite there being a huge range of different irises world-wide they can all be classified into one of three distinct classes. The classes are: Iris-Setosa, Iris-Versicolour and Iris-Virginica. The standard method of taxonomy is carried out by measuring four attributes of the flower in centimetres, that is sepal length and width, and petal length and width. Using these measures only one of the classes is linearly separable from the others. The Iris-setosa class to be distinct from the other two. Iris-versicolor and Iris-viginica, however show a high degree of overlap. The database used consisted of 150 samples, 50 for each class. Half of the data in each class were used for training and the other half for testing. A binary output target representation was used where each one of 3 output nodes was true for a particular class of iris and false for all other. The results for the best RBF networks and NRBF network are compared in table 1 with those of other methods. As expected, standard RBF nets exhibit poorer generalisation on the test set.

Classifier	Learning	Testing	Combined
3 Output Weighted RBF	***100%***	***95%***	***97%***
Fuzzy SOFM	100%	95%	97%
Genetic Programming	98%	95%	97%
Optical Thin Film	98%	93%	97%
3 Output Standard RBF	***100%***	***85%***	***93%***

Table 1. Summary of results for various classifier techniques[3].

The results for the Fuzzy Self Organised map are taken from (Hokirawa, 1997[11]). Genetic programming was a technique used by Backer (1996)[3]. Purvis and Li (1995)[18] describe the learning method using an Optical Thin-Film Model. It can be seen that the results achieved using NRBF nets match those of these techniques well.

5. CONCLUSION

These simple examples results suggest that Normalized RBF nets have very good generalisation properties that are beneficial in classification tasks. This is due to the property of normalised RBF nets to produce a significant output even for input vectors far from the centre of the receptive field of any of the hidden nodes.

Taking advantage of this, a modified learning procedure allows NRBF nets to operate with significantly less hidden nodes in classification tasks.

When applied to classification problems, the modified learning procedure results in nodes being recruited mainly along class boundaries. This obviates the need for a dense coverage of class domains, in contrast to standard RBF nets. Thus NRBF nets may contribute to soften the curse of dimensionality associated with networks of localized basis functions.

---

[3]It should be noted that the percentages have been rounded to the nearest percentage point.

It should be pointed out that the modified learning rule also performs well in a function mapping example where the NRBF net outperforms the standard RBF network, in terms of training error and generalization error. Interestingly, it leads to a more uniform performance over the training domain (Bugmann, 1998[5]).

A very positive aspect of NRBF nets is that the size of basis functions is relatively uncritical for the performance in classification and mapping. Further, learning is extremely fast. That makes this type of network very easy to use.

ACKNOWLEDGEMENTS:

Stimulating discussions with Kaspar Althoefer are gratefully acknowledged. The result on the Iris benchmark problem were produced during a student project by David Grant.

REFERENCES

1. Althoefer K. and Bugmann G.(1995) "Planning and Learning Goal-Directed Sequences of Robot-Arm movements", in Fogelman-Soulié F. and Gallinari P. (eds), Proc. of ICANN'95, Paris, Vol. 1, 449-454.
2. Andersen H.C., Lotfi A. & Westphal L. (1998) "Comments on 'Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems' Submitted to IEEE Trans. on Neural Networks, June, 1997 (http://www.elec.uq.edu.au/~andersen/).
3. Backer G. (1996) Learning with missing data using Genetic Programming. 1st Online Workshop on Soft Computing, http://www.tu-bs.de/institute/psych/gb/gb.html
4. Bishop C.M. (1995) "Neural networks for pattern recognition". Clarendon Press, Oxford, UK.
5. Bugmann G. (1998) "Normalized Radial Basis Function Networks", Neurocomputing, 20, Special Issue on Radial Basis Functions, in press. (Preprint downloadable from: http://www.tech.plym.ac.uk/soc/staff/GuidBugm/Bugmann.html).
6. Bugmann G., Koay K.L., Barlow N., Phillips M. and Rodney D. (1998) "Stable Encoding of Robot Trajectories using Normalised Radial Basis Functions: Application to an Autonomous Wheelchair", Proc. 29th Intl. Symp. Robotics (ISR'98), 27-30 April, Birmingham, UK, pp. 232-235. (ISBN 0 9524454 7 6). (Downloadable from: http://www.tech.plym.ac.uk/soc/staff/GuidBugm/Bugmann.html).
7. Broomhead D.S. and Lowe D. (1988) "Multivariable Functional Interpolation and Adaptive Networks", Complex Systems, 2, pp. 321-355.
8. Brown M. and Harris C. (1994) "Neurofuzzy Adaptive Modelling and Control", Prentice Hall, Hemel Hempstead, UK
9. Cha I. & Kassam S.A. (1995) "Interference Cancelation Using Radial Basis Function Networks", Signal Processing, 47, pp. 247-268.
10. Grant D. (1997) "Iris Classification", Student project, Postgrade Course in Computational Intelligence, accessible at http://freespace.virgin.net/david.grant7/iriscont.htm.
11. Horikawa S. (1997) "Fuzzy Classification System Using Self-Organising Feature Map". Oki Technical Review. Number 159. Vol 63. July 1997.. http://www.obd.com/oki/otr/html/nf/otr-159-05.html
12. Jang J.-S. R. & Sun C.-T. (1993) "Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems," IEEE Trans. on Neural Networks, vol. 4, pp. 156-159, Jan. 1993. (http://www.cs.nthu.edu.tw/~jang/publication.htm)

13. Jones J.P. & Palmer L.A. (1987) "The two-dimensional spatial structure of simple receptive fields in cat striate cortex". Journal of Neurophysiology, 58, pp. 1187-1211.
14. Lang K.J. & Witbrock M.J. (1988) "Learning to tell two spirals apart". In Touretzky D.S., Hinton G.E. & Sejnowski T.J. (eds). Proc. of the 1988 Connectionist Models Summer School, Morgan Kaufmann Publishers, San Mateo, CA.
15. Marcelja S. (1980) "Mathematical description of the response of simple cortical cells". J. Opt. Soc. Am., 70, pp. 1297-1300.
16. Moody, J. and Darken, Ch. (1989) "Fast learning in networks of locally-tuned processing units.", Neural Computation, 1, 281-294.
17. Powell M.J.D. (1987) "Radial Basis Functions for Multivariate Interpolation: A Review", in Mason J.C. and Cox M.G. (eds) "Algorithms for Approximation", Clarendon Press, Oxford, pp. 143-167.
18. Purvis M. & Li X. (1995) "Connectionist Learning Using an Optical Thin-Film Model". Computer and Information Science. University of Otago, Dunedin. New Zealand. http://divcom.otago.ac.nz:800/COM/INFOSCI/SECML/lab/publications/pub.html
19. Rao A.V., Miller D., Rose K. & Gersho A. (1997) "Mixture of experts regression modeling by deterministic annealing", IEEE Trans. on Signal Processing, 45, pp. 2811-2820.
20. Servin M. & Cuevas F.J. (1993) "A New Kind of Neural Network Based on Radial Basis Functions", Revista Mexicana de Fisica, 39:2, pp. 235-249.
21. Shao J., Kee Y.C. and Jones R. (1993) "Orthogonal Projection Method for Fast On-Line Learning Algorithm of Radial Basis Function Neural Networks", Proc. INNS World Congress on Neural Networks, Portland Oregon, USA, Vol. 3, pp. 520-535.