A First Approach to Birth Weight Prediction Using RBFNNs

A. Guillén[1],, I. Rojas[2], J. González[2], H. Pomares[2], and L. J. Herrera[2]

[1] Department of Informatics
University of Jaen. Spain

[2] Department of Computer Architecture and Computer Technology
Universidad de Granada. Spain

Abstract. This paper presents a first approach to try to determine the weight of a newborn using a set of variables determined uniquely by the mother. The proposed model to approximate the weight is a Radial Basis Function Neural Network (RBFNN) because it has been successfully applied to many real world problems. The problem of determining the weight of a newborn could be very useful by the time of diagnosing the gestational diabetes mellitus, since it can be a risk factor, and also to determine if the newborn is macrosomic. However, the design of RBFNNs is another issue which still remains as a challenge since there is no perfect methodology to design an RBFNN using a reduced data set, keeping the generalization capabilities of the network. Within the many design techniques existing in the literature, the use of clustering algorithms as a first initialization step for the RBF centers is a quite common solution and many approaches have been proposed. The following work presents a comparative of RBFNNs generated using several algorithms recently developed concluding that, although RBFNNs that can approximate a training data set with an acceptable error, further work must be done in order to adapt RBFNN to large dimensional spaces where the generalization capabilities might be lost.

1 Introduction

The problem of predicting the weight of a newborn using some parameters measured from the mother translates into the problem of approximating a function. Formally, a function approximation problem can be formulated as, given a set of observations $\{(\boldsymbol{x}_k; y_k); k = 1, ..., n\}$ with $y_k = F(\boldsymbol{x}_k) \in \mathrm{I\!R}$ and $\boldsymbol{x}_k \in \mathrm{I\!R}^d$, it is desired to obtain a function $\mathcal{G}$ so $y_k = \mathcal{G}\ (\boldsymbol{x}_k) \in \mathrm{I\!R}$ with $\boldsymbol{x}_k \in \mathrm{I\!R}^d$.

Designing an RBF Neural Network (RBFNN) to approximate a function from a set of input-output data pairs, is a common solution since this kind of networks are able to approximate any function [4, 12]. Once this function is learned, it will be possible to generate new outputs from input data that were not specified in the original data set, making possible to predict the weight of a newborn.

The most important information that could be obtained is the fetal macrosomia, this is, a a birth weight of more than 4,000 g. The macrosomia is difficult to predict and clinical and ultrasonographic estimates tend to have errors [15]. Furthermore, the weight of the fetus is a risk factor for several diseases such us gestational diabetes mellitus [3], therefore, if we are able to approximate the weight of the newborn, we will know in advance one of the many elements that are used to identify diseases.

The rest of the paper is organized as follows, Section 2 describes briefly the RBFNN model, Section 3 introduces the algorithms used to design the RBFNNs to predict the newborn weight and Section 4 shows the results. Finally, in Section 5, conclusions are drawn.

2 RBFNN Description

An RBFNN (Figure 1) $\mathcal{F}$ with fixed structure to approximate an unknown function F with n entries and one output starting from a set of values $\{(\boldsymbol{x}_k; y_k); k = 1, ..., n\}$ with $y_k = F(\boldsymbol{x}_k) \in \mathbb{R}$ and $\boldsymbol{x}_k \in \mathbb{R}^d$, has a set of parameters that have to be optimized:

$$\mathcal{F}(\boldsymbol{x}_k; C, R, \Omega) = \sum_{j=1}^{m} \phi(\boldsymbol{x}_k; \boldsymbol{c}_j, r_j) \cdot \Omega_j \qquad (1)$$

where $C = \{\boldsymbol{c}_1, ..., \boldsymbol{c}_m\}$ is the set of RBF centers, $R = \{r_1, ..., r_m\}$ is the set of values for each RBF radius, $\Omega = \{\Omega_1, ..., \Omega_m\}$ is the set of weights and $\phi(\boldsymbol{x}_k; \boldsymbol{c}_j, r_j)$ represents an RBF. The activation function most commonly used for classification and regression problems is the Gaussian function because it is continuous, differentiable, it provides a softer output and improves the interpolation capabilities [2, 14].

The procedure to design an RBFNN starts by setting the number of RBFs in the hidden layer, then the RBF centers $\boldsymbol{c}_j$ must be placed and a radius r_j has to be set for each of them. Finally, the weights Ω_j can be calculated optimally by solving a linear equation system [5].

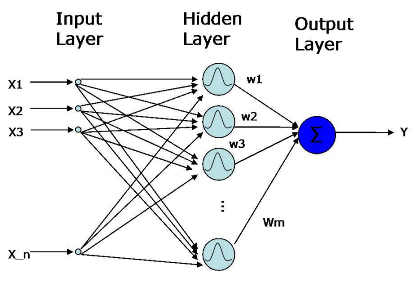

Fig. 1. A Radial Basis Function Neural Network

3 Algorithms for designing RFBNNs

This section presents the algorithms used to design the RBFNNs that predict the newborn weight. Some of these algorithms have been recently developed showing a better performance than classical algorithms used up to date.

3.1 Fuzzy C-means (FCM)

This algorithm presented in [1] uses a fuzzy partition of the data where an input vector belongs to several clusters with a membership value. It defines an objective distortion function to be minimized is:

$$J_h(U,C;X) = \sum_{k=1}^{n} \sum_{i=1}^{m} u_{ik}^h \|\boldsymbol{x}_k - \boldsymbol{c}_i\|^2 \tag{2}$$

where $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ are the input vectors, $C = \{\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_m\}$ are the centers of the clusters, $U = [u_{ik}]$ is the matrix where the degree of membership is established by the input vector to the cluster, and h is a parameter to control the degree of the partition fuzziness. After applying the least square method to minimize the function in Equation 2, we get the equations to reach the solution trough an iterative process.

3.2 Improved Clustering for Function Approximation Algorithm: ICFA

This algorithm uses the information provided by the objective function output in such a way that the algorithm will place more centers where the variability of the output is higher instead of where there are more input vectors.

In order to make the centers closer to the areas where the target function is more variable, a change in the similarity criteria used in the clustering process it is needed. To consider these situations, the parameter w is introduced (4) to modify the values of the distance between a center and an input vector. w will measure the difference between the estimated output of a center and the output value of an input vector. The smaller w is, the more the distance between the center and the vector will be reduced. This distance is calculated now by modifying the norm in the euclidean distance:

$$d_{kj} = \|\boldsymbol{x}_k - \boldsymbol{c}_j\|^2 \cdot w_{kj}^2. \tag{3}$$

To fulfill this task, the CFA algorithm defines a set $O = \{o_1, ..., o_m\}$ that represents a hypothetic output for each center.

$$w_{kj} = \frac{|F(\boldsymbol{x}_k) - o_j|}{\max\limits_{i=1}^{n} \{F(\boldsymbol{x}_i)\} - \min\limits_{i=1}^{n} \{F(\boldsymbol{x}_i)\}} \tag{4}$$

where $F(\boldsymbol{x})$ is the function output and o_j is the estimated output of $\boldsymbol{c}_j$.

Thus, the objective function to be minimize is redefined as:

$$J_h(U,C,W) = \sum_{k=1}^{n}\sum_{i=1}^{m} u_{ik}^h \|\boldsymbol{x}_k - \boldsymbol{c}_i\|^2 w_{ik}^2 \tag{5}$$

This function is minimized using an alternating optimization procedure in the same way as in the FCM algorithm, although new equations are needed to calculate the positions of the centers, the membership values and the expected output values:

$$u_{ik} = \left(\sum_{j=1}^{m}\left(\frac{d_{ik}}{d_{jk}}\right)^{\frac{2}{h-1}}\right)^{-1} \qquad \boldsymbol{c}_i = \frac{\sum_{k=1}^{n} u_{ik}^h \boldsymbol{x}_k w_{ik}^2}{\sum_{k=1}^{n} u_{ik}^h w_{ik}^2}$$

$$\boldsymbol{o}_i = \frac{\sum_{k=1}^{n} u_{ik}^h Y_k d_{ik}^2}{\sum_{k=1}^{n} u_{ik}^h d_{ik}^2} \tag{6}$$

where d_{ij} is the weighted euclidean distance between center i and input vector j, and $h > 1$ is a parameter that allow us to control how fuzzy will be the partition and usually is equal to 2.

The ICFA algorithm performs a migration step with the objective of reducing the global distortion of the partition by putting closer two centers. It performs a pre-selection of the centers, to decide what centers will be migrated, it is used a fuzzy rule that selects centers that have a distortion value above the average. By doing this, centers that do not add a significant error to the objective function are excluded because their placement is correct and they do not need help from other center. The center to be migrated will be the one that has assigned the smallest value of distortion and the destination of the migration will be the center that has the biggest value of distortion. If the total distortion of the partition has nor decreased after the migration, the centers remain at their original positions. The idea of a migration step was introduced in [13] as an extension of Hard C-means.

3.3 Fuzzy Possibilistic CFA

The algorithm that is used in the design is an adaptation of the one presented in [9] but modifying the way the input data is partitioned. As classical clustering algorithms, the proposed algorithm defines a distortion function that has to be minimized. The distortion function is based in a fuzzy-possibilistic approach as it was presented in [6], although the migration step remains the same as for ICFA. The function is:

$$J_h(U,C,T,W;X) = \sum_{k=1}^{n}\sum_{i=1}^{m} (u_{ik}^{h_f} + t_{ik}^{h_p}) D_{ikW}^2 \tag{7}$$

restricted to the constraints: $\sum_{i=1}^{m} u_{ik} = 1 \; \forall k = 1...n$ and $\sum_{k=1}^{n} t_{ik} = 1 \; \forall i = 1...m$.

As the previous approaches, the final position of the centers is reached by an alternating optimization approach where all the elements defined in the function to be minimized are updated iteratively using the equations obtained by differentiating $J_h(U, T, C, W; X)$ with u_{ik}, t_{ik}, $\boldsymbol{c}_i$ and o_i.

3.4 Possibilistic Centers Initializer (PCI)

This algorithm [8] adapts the algorithm proposed in [9] using a mixed approach between a possibilistic and a fuzzy partition, combining both approach as it was done in [16]. The objective function to be minimized is defined as:

$$J_h(U^{(p)}, U^{(f)}, C, W; X) = \sum_{k=1}^{n}\sum_{i=1}^{m} (u_{ik}^{(f)})^{h_f} (u_{ik}^{(p)})^{h_p} D_{ikW}^2 + \sum_{i=1}^{m} \eta_i \sum_{k=1}^{n} (u_{ik}^{(f)})^{h_f} (1 - u_{ik}^{(p)})^{h_p} \tag{8}$$

where $u_{ik}^{(p)}$ is the possibilistic membership of x_k in the cluster i, $u_{ik}^{(f)}$ is the fuzzy membership of $\boldsymbol{x}_k$ in the cluster i, D_{ikW} is the weighted euclidean distance, η_i is a scale parameter that is calculated by: $\eta_i = \frac{\sum_{k=1}^{n} (u_{ik}^{(f)})^{h_f} \|\boldsymbol{x}_k - \boldsymbol{c}_i\|^2}{(u_{ik}^{(f)})^{h_f}}$

This function is obtained by replacing de distance measure in the FCM algorithm by the objective function of the PCM algorithm, obtaining a mixed approach. The scale parameter determines the relative degree to which the second term in the objective function is compared with the first. This second term forces to make the possibilistic membership degree as big as possible, thus, choosing this value for η_i will keep a balance between the fuzzy and the possibilistic memberships.

3.5 Output Value-Based Initializer (OVI)

This algorithm [7] changes the perspective of the previous ones, the idea is to think the output space as a flat surface ($Y(\boldsymbol{x}_k) = 0$), where some of the values of this surface have been modified by an n-dimensional element, obtaining y_k. From this, it can be assumed that the most common value of the target function is constant and equal 0. The preprocessing of the output is performed by making the most frequent output value equal 0. This can be easily performed by calculating the fuzzy mode of the output values and subtracting it to each y_k. Once this situation is achieved, all the most common values that have an output around 0 will not affect the distortion significantly so the centers will be mostly influenced by the input vectors with high output values.

This distortion function combines the information provided by a coordinate in the input vector space and its corresponding output in such a way that, if a neuron is near an input vector and the output of the input vector is high, the activation value of that neuron respect that input vector will be high. The distortion function is defined as:

$$\delta = \sum_{k=1}^{n} \sum_{i=1}^{m} D_{ik}^2 a_{ik}^l |Y_k^p| \tag{9}$$

where D_{ik} represents the euclidean distance from a center c_i to an input vector $\boldsymbol{x}_k$, a_{ik} is the activation value that determines how important the input vector $\boldsymbol{x}_k$ is for the center $\boldsymbol{c}_i$, l is a parameter to control the degree of overlapping between the neurons, Y_k is the preprocessed output of the input vector $\boldsymbol{x}_k$, and p allows the influence of the output when initializing the centers to increase or decrease.

The OVI algorithm calculates how much an input vector will activate a neuron in function of its output value. From this, the value of the radius can be set as the distance to the farthest input vector that activates a center. In order to do that, a threshold has to be established to decide when an input vector activates or does not activate a center.

Using the values of the A matrix, an activation threshold ($\vartheta_{overlap}$) that allows us to calculate the distance to the farthest input vector that activates a neuron can be established. The proposed algorithm selects the radius for each center independently of the positions of the other centers, unlike in the KNN heuristic [11], and it allows to maintain an overlap between the RBFs, unlike in the CIV heuristic [10].

Each radius is defined as:

$$r_i = \max\{\ D_{ik} \ / \ a_{ik} > \vartheta_{overlap} \ , \ 1 \leq i \leq m \ , \ 1 \leq k \leq n \ \} \tag{10}$$

The selection of a threshold makes the algorithm more flexible, because it can increase or decrease the degree of overlap between the RBFs.

4 Experimental Results

The data used for the experiments was provided by the Preventative Medicine Department at the University of Granada and consists in a cohort of 969 pregnant women considering the following parameters: number of cigarettes smoked during the pregnancy, mother's weight at the beginning and at the end of the pregnancy, gestation days and the mother's age. A set of 500 randomly chosen elements from the original set was used for training and the rest for test.

To compare the results provided by the different algorithms, it will be used the normalized root mean squared error (NRMSE) which is defined as:

$$\textbf{NRMSE} = \sqrt{\frac{\sum_{k=1}^{n} \left(y_k - \mathcal{F}(\boldsymbol{x}_k; C, R, \Omega)\right)^2}{\sum_{k=1}^{n} \left(y_k - \bar{Y}\right)^2}} \tag{11}$$

where $\bar{Y}$ is the average of the outputs of the target function, in this case, the final weight of the newborn.

The radii of the RBFs were calculated using the k-neighbors algorithm with k=1, except for the OVI algorithm which uses its own technique as described above. The weights were calculated optimally by solving a linear equation system.

Table 1 shows the approximation errors for the training and test data sets. As the results show, the performance of the algorithms is quite similar although the OVI algorithm seems to perform better than the rest probably as a consequence of its own method to calculate the radii. All the algorithms are able to fit the training set with a reasonable error for any number of centers with no improvement of error when increasing the number of RBFs. Unfortunately, the test errors are unacceptable for all the algorithms showing how the RBFNNs loose their generalization capabilities in high dimensional spaces using a reduced amount of data.

Table 1. Mean of the approximation error (NRMSE) for the training and test sets.

Training

Clusters	**FCM**	**ICFA**	**FPCFA**	**PCI**	**OVI**
5	0.639	0.652	0.642	0.638	0.635
6	0.680	0.642	0.641	0.644	0.640
7	0.675	0.633	0.629	0.636	0.625
8	0.674	0.632	0.653	0.669	0.617
9	0.644	0.655	0.631	0.629	0.619
10	0.644	0.623	0.645	0.650	0.631

Test

Clusters	**FCM**	**ICFA**	**FPCFA**	**PCI**	**OVI**
5	4.583	4.461	4.518	4.575	4.545
6	4.569	4.606	4.499	4.625	4.619
7	4.639	4.550	4.634	4.479	4.579
8	4.593	4.573	4.575	4.451	4.621
9	4.580	4.570	4.577	4.614	4.607
10	4.628	4.588	4.597	4.601	4.591

5 Conclusions

This work has presented an application of RBFNNs to a real world problem: the prediction of a newborn's weight. This parameter could be quite useful in the diagnosis of macrosomia which can lead to complications at the childbirth and also to be considered in the diagnosis of other diseases which has the newborn weight as a risk factor. The RBFNNs were designed using a classical methodology where the centers of the RBFs are initialized using clustering techniques and applying local search algorithms. The results showed how, for the training set, the RBFNNs were able to approximate reasonably well the weights although test errors become unacceptable, independently of the algorithm used. This results cheer to keep on researching on this subject although aiming at other aspects such us how the Radial Basis Function behave in high dimensional spaces.

References

1. J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, New York, 1981.
2. A. G. Bors. Introduction of the Radial Basis Function (RBF) networks. *OnLine Symposium for Electronics Engineers*, 1:1–7, February 2001.
3. R. Dyck, H. Klomp, L.K. Tan, R.W. Turner, and M.A. Boctor. A comparison of rates, risk factors, and outcomes of gestational diabetes between aboriginal and non-aboriginal women in the Saskatoon Health District. *Diabetes Care*, 25:487–493, 2002.
4. A. Gersho. Asymptotically Optimal Block Quantization. *IEEE Transanctions on Information Theory*, 25(4):373–380, July 1979.
5. J. González, I. Rojas, J. Ortega, H. Pomares, F.J. Fernández, and A. Díaz. Multi-objective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation. *IEEE Transactions on Neural Networks*, 14(6):1478–1495, November 2003.
6. A. Guillén, I. Rojas, J. González, H. Pomares, L. J. Herrera, and A. Prieto. A Fuzzy-Possibilistic Fuzzy Ruled Clustering Algorithm for RBFNNs Design. *Lecture Notes in Computer Science*, 4259:647–656, 2006.
7. A. Guillén, I. Rojas, J. González, H. Pomares, L.J. Herrera, and A. Prieto. Supervised RBFNN Centers and Radii Initialization for Function Approximation Problems. In *International Joint Conference on Neural Networks, 2006. IJCNN '06*, pages 5814–5819, July 2006.
8. A. Guillén, I. Rojas, J. González, H. Pomares, L.J. Herrera, O. Valenzuela, and A. Prieto. A Possibilistic Approach to RBFN Centers Initialization. *Lecture Notes in Computer Science*, 3642:174–183, 2005.
9. A. Guillén, I. Rojas, J. González, H. Pomares, L.J. Herrera, O. Valenzuela, and A. Prieto. Improving Clustering Technique for Functional Approximation Problem Using Fuzzy Logic: ICFA algorithm. *Lecture Notes in Computer Science*, 3512:272–280, June 2005.
10. N. B. Karayiannis and G. W. Mi. Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques. *IEEE Transactions on Neural Networks*, 8:1492–1506, November 1997.
11. J. Moody and C.J. Darken. Fast learning in networks of locally-tunned processing units. *Neural Computation*, 1(2):281–294, 1989.
12. J. Park and J. W. Sandberg. Universal approximation using radial basis functions network. *Neural Computation*, 3:246–257, 1991.
13. G. Patanè and M. Russo. The Enhanced-LBG algorithm. *Neural Networks*, 14(9):1219–1237, 2001.

14. I. Rojas, M. Anguita, A. Prieto, and O. Valenzuela. Analysis of the operators involved in the definition of the implication functions and in the fuzzy inference proccess. *International Journal of Approximate Reasoning*, 19:367–389, 1998.
15. M. A. Zamorski and W.S. Biggs. Management of Suspected Fetal Macrosomia. *American Family Physician*, 63(2), January 2001.
16. J. Zhang and Y. Leung. Improved possibilistic C–means clustering algorithms. *IEEE Transactions on Fuzzy Systems*, 12:209–217, 2004.