2009 Ninth International Conference on Intelligent Systems Design and Applications

A Sensitivity Clustering Method for Memetic Training of Radial Basis Function Neural Networks

F. Fernández-Navarro, P.A. Gutiérrez
C. Hervás-Martínez
Department of Computer Science and Numerical Analysis, University of Córdoba,
Campus de Rabanales, C2 building,1004
i22fenaf@uco.es

Abstract

In this paper, we propose a Memetic Algorithm (MA) for classifier optimization based on a clustering method that applies the k-means algorithm over a specific derived space. In this space, each classifier or individual is represented by the set of the accuracies of the classifier for each class of the problem. The proposed sensitivity clustering is able to obtain groups of individuals that perform similarly for the different classes. Then, a representative of each group is selected and it is improved by a local search procedure. This method is applied in specific stages of the evolutionary process. The sensitivity clustering process is compared to a clustering process applied over the n-dimensional space that represent the behaviour of the classifier over each training pattern, where n is the number of patterns. This second method clearly results in a higher computational cost. The comparison is performed in ten imbalanced datasets, including the minimun sensitivity results (i.e. the accuracy for the worst classified class). The results indicate that, although in general the differences are not significant, the sensitivity clustering obtains the best perfomance for almost all datasets both in accuracy and minimum sensitivity, involving a lower computational demand.

1. Introduction

Evolutionary Algorithms (EAs) [2], generally require a great number of iterations and they converge slowly, especially in the neighbourhood of the global optimum [5]. It thus makes sense to incorporate a faster Local Search (LS) algorithm into the EA in order to overcome this lack of efficiency while keeping advantages of both optimization methods. During last years new methods have been developed in order to improve the lack of precision of EAs using local optimization algorithms as part of the EA cycle, what is known as Memetic Algorithms (MAs) [10]. However the use of the LS in every generation and for every individual of the EA would result in a prohibitive computational cost. Clustering methods can help to select the individuals of an EA which the LS is applied. They are a class of global optimization methods of which an important part includes a cluster-analysis technique [4]. These methods create groups (clusters) of mutually close points that could correspond to relevant regions of attraction. The use of a clustering algorithm allows the selection of individuals representing similar behaviours in the problem. In this way, the optimized individuals are more likely to converge towards different local optima.

In this paper, we present a MA which combines the ideas of genetic algorithms, LS and clustering techniques to solve the complex problem of finding the optimal structure and weights of Radial Basis Function Neural Networks (RBFNNs) for multi-classification problems. This MA is based on a previously Hybrid Algorithm for regression problems and Product Unit Neural Networks [8]. We propose a specific clustering method (a sensitivity clustering) applied over a derived space, where each classifier or individual is represented by the set of the accuracies of the classifier for each class of the problem. The sensitivity clustering process is compared to the clustering process applied over the n-dimensional space that represent the behaviour of the classifier over each training pattern, where n is the number of patterns. This second method was the clustering method used in [8], which clearly resulted in a higher computational cost than the sensitivity clustering proposed in this paper. The results indicate that, although in general the differences are not significant, the sensitivity clustering obtains the best perfomance for almost all datasets both in accuracy and minimum sensitivity, involving also a lower computational demand.

This paper is organized as follows: Section 2 presents

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.209

the sensitivity measure; Section 3 describes the two clustering techniques used to select the individuals which the LS will be applied to; Section 4 is dedicated to a short description of the RBFNN model; Section 5 states the most relevant aspects of the MA used for training the RBFNNs; Section 6 explains the experiments carried out; and Section 7 summarizes the conclusions of the paper.

2. Sensitivity Measure

A classification problem with Q classes and N training or testing patterns is considered with g as a classifier obtaining a $Q \times Q$ contingency or confusion matrix $M(g)$:

$$M(g) = \left\{ n_{ij};\ \sum_{i,j=1}^{Q} n_{ij} = N \right\} \quad (1)$$

where n_{ij} represents the number of times the patterns are predicted by classifier g to be in class j when they really belong to class i. The diagonal corresponds to correctly classified patterns and the off-diagonal to mistakes in the classification task, as shown in Table 1.

Table 1. Confusion matrix of a classifier

Class	1	2	...	Q	Priors
1	n_{11}	n_{12}	...	n_{1Q}	f_1
2	n_{21}	n_{22}	...	n_{2Q}	f_2
...	...	...	...	...	...
Q	n_{Q1}	n_{Q2}	...	n_{QQ}	f_Q

Let us denote the number of patterns associated with class i by $f_i = \sum_{j=1}^{Q} n_{ij}$, $i = 1, \ldots, Q$. Different scalar measures can be defined in order to take the elements of the confusion matrix into consideration from different points of view. Let $S_i = n_{ii}/f_i$ be the number of patterns correctly predicted to be in class i with respect to the total number of patterns in class i (sensitivity for class i). Therefore, the sensitivity for class i independently estimates the probability of correctly predicting a pattern of class i.

3. Clustering Techniques

3.1. Sensitivity Clustering

In this paper, a clustering process for obtaining groups of RBFNN models is proposed based on the performance of each RBFNN for each class. This method could be used for any type of classifier. In this way, given a classifier g, the following application to the space $\mathbb{R}^Q$ is considered:

$$\mathbf{s}_g = \{S_{g1}, S_{g2}, ..., S_{gQ}\} \quad (2)$$

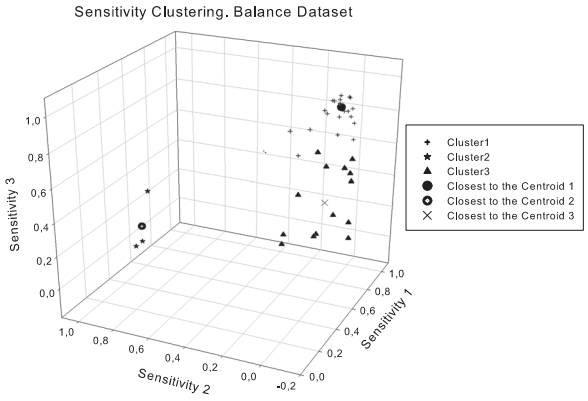

Figure 1. Sensitivity Clustering: 40 RBFNNs trained for Balance dataset, and a k-means process with $k = 3$.

where S_{gi} is the sensitivity of the classifier g for the i-th class obtained as indicated in the previous section. Then, we can define the distance between two classifiers g_1 and g_2 as the Euclidean distance between the associated vectors $\mathbf{s}_{g_1}$ and $\mathbf{s}_{g_2}$:

$$d(g_1, g_2) = ||\mathbf{s}_{g_1} - \mathbf{s}_{g_2}|| = \sqrt{\sum_{i=1}^{Q} (S_{g_1 i} - S_{g_2 i})^2} \quad (3)$$

Now considering a group of M classifiers $G = \{g_1, ..., g_M\}$, the associated vectors are derived, $\{\mathbf{s}_{g_1}, ..., \mathbf{s}_{g_M}\}$. With this distance measure, we apply the standard k-means [4] algorithm for obtaining a partition $P = \{C_1, ..., C_k\}$. The obtained groups represent classifiers that have a similar behaviour for the different classes of the problem. The classifier that is closest to the centroid of the i-th cluster is represented by $\overline{\mathbf{s}_i}$ and the members of the set $\{\overline{\mathbf{s}_1}, ..., \overline{\mathbf{s}_k}\}$ are selected as the most representative classifiers of the set G. An example for a group of 40 classifiers for the Balance dataset of 3 classes is given in Figure 1, where we apply k-means with $k = 3$ clusters.

3.2. Training Behavior Clustering

This is the clustering process proposed in [8]. In this case, given a classifier g, the following application to the space $\mathbb{R}^N$ is considered:

$$\mathbf{c}_g = \{c_{g1}, c_{g2}, ..., c_{gN}\} \quad (4)$$

where N is the number of training patterns and c_{gi} is a binary value indicating if the i-th pattern has been correctly

classified by the g classifier. After that, to obtain the most representative models in the search space, the same process as the Sensitivity Clustering is applied.

4. Architecture of the RBFNNs

We consider RBFNNs [3, 6] as the base classification model. An RBFNN is a three-layer feed-forward neural network. Let the number of nodes in the input layer, in the hidden layer and in the output layer be p, m and Q respectively. For any sample $\mathbf{x} = (x_1, x_2, \ldots, x_p)$, the output of the RBFNN is $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_Q(\mathbf{x}))$. The model of a RBFNN can be described with the following equation:

$$f_j(\mathbf{x}) = \beta_{0j} + \sum_{i=1}^{m} \beta_{ij} \cdot \phi_i(\|\mathbf{x} - \mathbf{c_i}\|),\ j = 1, 2, \ldots, Q, \quad (5)$$

where $\phi_i(\mathbf{x})$ is a non-linear mapping from the input layer to the hidden layer, while the mapping from hidden layer to output layer (i.e. $\phi_i(x)$ to $f_j(\mathbf{x})$) is linear. $\boldsymbol{\beta}_j = (\beta_{1j}, \beta_{2j}, \ldots, \beta_{mj})^T$, where $j = 1, 2, ..., Q$, β_{ij} is the connection weight between the i-th hidden node and the j-th output node and β_{0j} is the bias value for class j. Finally, $\|.\|$ represents the Euclidean norm and $\phi(.)$ is the Gaussian function:

$$\phi(\|\mathbf{x} - \mathbf{c}_i\|) = e^{-\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{2 \cdot \sigma_i^2}}, \quad (6)$$

where $\mathbf{c_i} = (c_i^1, c_i^2, \ldots, c_i^m) \in \mathbb{R}^m$ represents the i-th centre in the hidden layer, $i = 1, 2, \ldots, m$ and σ_i controls the attenuation speed of Gaussian function [13].

The activation function of each output node is the softmax function given by:

$$p_j(\mathbf{x}) = \frac{e^{f_j(\mathbf{x})}}{\sum_{i=1}^{Q} e^{f_i(\mathbf{x})}}; \quad j = 1, ..., Q \quad (7)$$

where $p_j(\mathbf{x})$ is the probability that pattern $\mathbf{x}$ belongs to class j.

Under this probabilistic interpretation of the model outputs, it is possible to evaluate the model using the cross-entropy error function, given by:

$$l(g) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{Q} (y_j(\mathbf{x_i}) ln(p_j(\mathbf{x}_i))) \quad (8)$$

where $y_j(\mathbf{x_i})$ is the expected value for the class j and pattern i (i.e. $y_j(\mathbf{x_i})$ is 1 if $\mathbf{x_i}$ belongs to the class j and 0 otherwise), g is the evaluated RBFNN and N is the number of training patterns.

5. Memetic Algorithm

A MA has been combined with the previously presented clustering processes for optimizing the structure and the weigths of RBFNNs. The basic framework of the MA is the following: the search begins with an initial population of RBFNNs and, in each iteration, the population is updated using a population-update algorithm which evolves both its structure and weights. The population is subject to the operations of replication, mutation and recombination. The main characteristics of the algorithm are the following:

1. *Representation of the Individuals*. The algorithm evolves architectures and connection weights simultaneously, each individual being a fully specified RBFNN. RBFNNs are represented using an object-oriented approach and the algorithm deals directly with the RBFNN phenotype. Each connection is specified by a binary value indicating if the connection exists and a real value representing its weights.

2. *Error and Fitness Functions*. We consider $l(g)$ (see Eq. 8) as the error function of an individual g of the population. The fitness measure needed for evaluating the individuals is a strictly decreasing transformation of $l(g)$ given by $A(g) = 1/(1 + l(g))$, where $0 < A(g) \leq 1$.

3. *Initialization of the Population*. The initial population is generated trying to obtain RBFNNs with the maximun possible fitness. First, 5.000 random RBFNNs are generated. The number of connections between all RBFs of an individual and the input layer is a random value in the interval $[1, k]$, and all of them are connected with the same randomly chosen input variables. In this way, all the RBFs of each individual are initialized in the same random subspace of the input variables. A pattern of the training set is randomly selected for assigning the weights between the input layer and the hidden layer, considering the value of the selected input variables for the corresponding pattern. The connections between the hidden layer and the output layer are initialized in the $[-O, O]$ interval. The obtained individuals are evaluated using the fitness function and the initial population is finally obtained by selecting the best 500 RBFNNs.

   In order to improve the randomly generated centres, the standard k-means clustering algorithm is applied using these random centres as the initial centroids for the algorithm and a maximun number of 100 iterations.

4. *Recombination*. Different crossover operators are applied:

Table 2. Characteristics of the ten datasets used for the experiments: number of instances (Size), number of Real (R), Binary (B) and Nominal (N) input variables, total number of inputs (#In.), number of classes (#Out.), per-class distribution of the instances (Distribution) and minimum and maximum number of hidden nodes used for each dataset ($[M_{\min}, M_{\max}]$)

Dataset	Size	R	B	N	#In.	#Out.	Distribution	$[M_{\min}, M_{\max}]$
Hepatitis	155	6	13	–	19	2	$(32, 123)$	$[1, 3]$
Breast-Cancer	286	4	3	2	15	2	$(201, 85)$	$[1, 3]$
Haberman	306	3	–	–	3	2	$(225, 81)$	$[1, 3]$
YeastCYTvsPOX	482	8	–	–	8	2	$(463, 19)$	$[4, 9]$
Newthyroid	215	5	–	–	5	3	$(150, 35, 30)$	$[4, 9]$
Balance	625	4	–	–	4	3	$(288, 49, 288)$	$[4, 9]$
Glass	214	9	–	–	9	6	$(70, 76, 17, 13, 9, 29)$	$[9, 12]$
Zoo	101	1	15	–	16	7	$(41, 20, 5, 13, 4, 8, 10)$	$[4, 9]$
Ecoli	336	7	–	–	7	8	$(143, 77, 52, 35, 20, 5, 2, 2)$	$[4, 9]$
Yeast	1484	8	–	–	8	10	$(463, 429, 244, 163, 51, 44, 35, 30, 20, 5)$	$[9, 12]$

All nominal variables are transformed to binary variables.

- *Binary Crossover Operator*. This binary-operator needs two RBFNNs to be applied, although it only changes one of them. The operator takes an uniformly randomly chosen number of consecutive hidden neurons from the first network, and another random sequence from the second. Then it replaces the first of these sequences by the second one, so that the second individual remains unchanged.
- *Multipoint Crossover Operator*. This operator replaces with probability 0.2 every hidden neuron of the first RBFNN by a randomly chosen neuron coming from the second net.

5. *Structural Mutation*. Structural mutation implies a modification in the structure of the RBFNNs and allows the exploration of different regions in the search space, helping to keep the diversity of the population. There are four different structural mutations: hidden node addition, hidden node deletion, connection addition and connection deletion. These four mutations are applied sequentially to each network.

6. *Parametric Mutation*. Different weight mutations are applied:

   - *Centre and Radii Mutation*. These parameters are modified in the following way:
     - Centre creep. It changes the values for the centres applying a Normal noise. For each dimension of the centre point, the Normal distribution is centred on the current value and it is as wide as the radius.
     - Radius creep. It changes the values for the radii applying another Normal noise. The Normal distribution is centred on the current value and is as wide as the range of each dimension.
     - Randomize centres. Changes the values of the centres of the hidden neurons to random values in the range allowed for each dimension of the input space.
     - Randomize radii. It changes radius values randomly, always with values in the corresponding range of each input space dimension.
   - *Output-to-Hidden Node Connection Mutations* [9]. These connections are modified by adding a Normal noise, $w(t+1) = w(t) + \xi(t)$, where $\xi(t) \in N(0, T(g))$ and $N(0, T(g))$ represents a one-dimensional normally distributed random variable with mean 0 and with variance the network temperature ($T(g) = 1 - A(g)$).

7. *Local Search*. The proposed methodology is based on the combination of the EA, a clustering process, and a local improvement procedure. The local optimization algorithm used in our paper is the $iRprop+$ [7] optimization method.

Two different versions of the MA are presented in this paper, depending of the clustering technique we carry out. Both methodologies apply the clustering process presented on Section 3.1 and 3.2 on the complete set of the individuals every G_0 generations. After that, we apply $iRprop+$ algorithm to the individual closest to the centroid obtained in each cluster and they are returned to the population with its fitness and values updated since our MA is based on the Lamarckian model [12].

6. Experiments

The two proposed methodologies are applied to ten datasets taken from the UCI repository [1], to test its overall performance when compared each other. Since we are interested on improving the sensitivity for the worst classified class in each dataset, we have selected ten imbalanced dataset. The selected datasets include both binary problems and multi-class problems and present different numbers of instances, features and classes (see Table 2). The YeastCYTvsPOX dataset is the Yeast dataset considering only patterns from CYT and POX classes. The minimum and maximum number of hidden nodes have been obtained as the best result of a preliminary experimental design, considering a small, medium and high value: $[M_{\min}, M_{\max}] \in \{[1,3],[4,9],[9,12]\}$. This value is also included in Table 2. The next subsection defines the experimental design and the following shows the results obtained.

6.1. Experimental Design

The experimental design was conducted using a 10-fold cross validation, with 10 repetitions per each fold. The performance of each method has been evaluated using the correct classification rate (C) and the Minimun Sensitivity (MS) value for the generalization set, i.e. the accuracy for the class that is worst classified.

All the parameters used in the evolutionary algorithm except the maximun and minimun number of RBFs in the hidden layer have the same values for all problems considered. We have done a simple linear rescaling of the input variables in the interval $[-2,2]$, X_i^* being the transformed variables. The connection between hidden and output layer are initialized in the $[-5,5]$ interval (i.e. $[-O,O] = [-5,5]$). The initial value of the radii r_j is obtained in the interval $(0, d_{max}]$, where d_{max} is the maximun distance between two training input examples.

The size of the population is $N = 500$. For the structural mutation, the number of nodes that can be added or removed is within the $[1,2]$ interval, and the number of connections to add or delete in the hidden and the output layer during structural mutations is within the $[1,7]$ interval. The number of the clusters is $k = 6$ for the k-mean algorithm. The $iRprop+$ local improvement procedure is performed every 50 generations (i.e. $G_0 = 50$), 8 times during the evolution, considering a maximum of 75 cicles. In this way, the algorithm stops when 400 generations are completed.

6.2. Analysis of the Results

In Table 3, the mean and the standard deviation of the correct classification rate and minimun sensitivity in the generalization set (C_G and MS_G) are shown for each dataset and a total of 100 executions. From the analysis of the results, it can be concluded, from a purely descriptive point of view, that the Sensitivity Clustering obtained the best C_G results for eight datasets, and the Training Behavior Clustering yielded the higher performance for the other two datasets. For the MS_G measure, the Sensitivity Clustering got the higher performance for seven datasets, the Training Behavior only for two datasets and both methods obtained the same value for Yeast dataset.

To ascertain the statistical significance of the differences observed in each dataset performance, we have carried out an t-test (for those cases in which a normal distribution could be considered) or a Mann-Whitney test [11] (for those cases in which a normal distribution could not be considered), with the C_G and the MS_G of the best models as the test variable (previously evaluating if the C_G and MS_G values follow or not a normal distribution, using a Kolmogorov-Smirnov test). In Table 4, we have included the statistical results obtained, concluding that the differences obtained for almost all datasets are not significant, except for C_G and MS_G values of YeastCYTvsPOX datasets and MS_G values of Glass dataset.

Table 4. p-values of the t-test or the Mann-Whitney test considering the C_G and MS_G values as the test variable. Sensitivity Clustering vs Training Behaviour Clustering

	p-values			
	C_G		MS_G	
Dataset	t-test	M-W	t-test	M-W
Hepatitis	–	0.622	–	0.941
Breast-Cancer	0.331	–	–	0.143
Haberman	–	0.984	–	0.324
YeastCYTvsPOX		$0.000_{(*)}$	–	$0.000_{(*)}$
Newthyroid	–	0.617	–	0.966
Balance	–	0.866	–	0.857
Glass	0.510	–	–	$0.004_{(*)}$
Zoo	–	0.538	–	0.504
Ecoli	0.704	–	–	0.838
Yeast	0.397	–	–	–

$(*)$: Statistically significant differences with $\alpha = 0.05$

Table 3. Mean and Standard Deviation of the Accuracy ($C_G(\%)$) and Minimun Sensitivity ($MS_G(\%)$) Results From 100 Executions of a 10-Fold Cross Validation

	$Method(C_G(\%)$ and $M_G(\%))$			
	SensitivityClustering		TrainingBehaviorClustering	
Dataset	$C_G(\%)$	$MS_G(\%)$	$C_G(\%)$	$MS_G(\%)$
Hepatitis	$\mathbf{81.82 \pm 8.05}$	$\mathbf{51.00 \pm 18.85}$	81.57 ± 7.00	49.78 ± 21.27
Breast-Cancer	$\mathbf{74.07 \pm 8.13}$	$\mathbf{38.79 \pm 13.90}$	73.87 ± 7.54	35.89 ± 13.56
Haberman	$\mathbf{73.28 \pm 5.80}$	27.12 ± 11.60	73.24 ± 6.38	$\mathbf{28.60 \pm 13.16}$
YeastCYTvsPOX	99.14 ± 0.41	$\mathbf{79.00 \pm 10.14}$	$\mathbf{97.26 \pm 1.64}$	45.91 ± 28.86
Newthyroid	96.54 ± 2.71	$\mathbf{86.73 \pm 14.41}$	$\mathbf{96.91 \pm 2.40}$	86.21 ± 14.82
Balance	$\mathbf{95.70 \pm 2.37}$	$\mathbf{73.58 \pm 21.14}$	95.61 ± 2.57	73.51 ± 21.23
Glass	$\mathbf{68.40 \pm 8.92}$	$\mathbf{6.97 \pm 17.53}$	67.59 ± 8.40	0.96 ± 5.67
Zoo	$\mathbf{95.33 \pm 6.56}$	$\mathbf{62.50 \pm 48.39}$	94.84 ± 6.56	58.00 ± 49.09
Ecoli	$\mathbf{84.82 \pm 8.55}$	26.18 ± 31.01	84.46 ± 8.80	$\mathbf{27.75 \pm 32.42}$
Yeast	$\mathbf{57.85 \pm 4.65}$	$\mathbf{0.00 \pm 0.00}$	57.82 ± 3.79	$\mathbf{0.00 \pm 0.00}$
Mean	83.84	41.91	83.72	41.11

The best result is in bold face

7. Conclusions

In this paper, we have proposed a MA to solve multi-classification problems. This algorithm is based on the combination of an EA, a new sensitivity clustering process, and a local-search procedure. The proposed clustering process allows the selection of individuals representing different regions in the search space. These selected individuals are the ones subject to local optimization. In this way, the optimized individuals are more likely to converge towards different local optima. We have compared this algorithm to the same algorithm using the clustering process proposed in [8], where each of the clustered individuals is represented by an N-dimensional vector, N being the number of training patterns.

The MA proposed was applied to ten imbalanced classification problems of the UCI repository [1]. The results show that the proposed sensitivity clustering process results in better C_G and MS_G performance, although these differences are not in general statistically significant. These results present the sensitivity clustering as a very advisable alternative for selecting the individuals to apply the LS process in MAs when applied to classification problems, given that the computational cost of this process is lower than that of the originally proposed clustering technique.

References

[1] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

[2] T. Back. *Evolutionary Algorithms in Theory and Practice*. Oxford, 1996.

[3] J. A. S. Freeman and D. Saad. Learning and generalization in radial basis function networks. *Neural Computation*, 7(5):1000–1020, 1995.

[4] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2nd edition, 1999.

[5] C. R. Houck, J. A. Joines, M. G. Kay, and J. R. Wilson. Empirical investigation of the benefits of partial lamarckianism. *Evol. Comput.*, 5(1):31–60, 1997. 1326739.

[6] Y. Hwang and S. Bang. An efficient method lo construct radial basis function neural network classifier. *Neural Networks*, 10(8):1495–1503, 1997.

[7] C. Igel and M. Hsken. Empirical evaluation of the improved rprop learning algorithms. *Neurocomputing*, 50(6):105–123, 2003.

[8] A. C. Martinez-Estudillo, C. Hervas-Martnez, F. J. Martinez-Estudillo, and N. Garcia-Pedrajas. Hybridization of evolutionary algorithms and local search by means of a clustering method. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, 36(3):534–545, 2006.

[9] F. J. Martnez-Estudillo, C. Hervs-Martnez, P. A. Gutirrez, and A. C. Martnez-Estudillo. Evolutionary product-unit neural networks classifiers. *Neurocomputing*, 72(1-2):548–561, 2008.

[10] P. Moscato and C. Cotta. A gentle introduction to memetic algorithms. In *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research and Management Science*, pages 105–144. Springer New York, 2003.

[11] A. C. Tamhane and D. D. Dunlop. *Statistics and Data Analysis*. Prentice Hall, 2000.

[12] D. L. Whitley, V. S. Gordon, and K. E. Mathias. Lamarckian evolution, the baldwin effect and function optimization. In Y. Davidor, H. P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature – PPSN III*, pages 6–15, Berlin, 1994. Springer.

[13] Z. R. Yang. A novel radial basis function neural network for discriminant analysis. *IEEE Transactions on neural networks*, 17(3):604–612, 2006.