Memetic Pareto Differential Evolution for Designing Artificial Neural Networks in Multiclassification Problems Using Cross-Entropy Versus Sensitivity

Juan Carlos Fernández[1], César Hervás[1], Francisco José Martínez[2], Pedro Antonio Gutiérrez[1], and Manuel Cruz[1]

[1] Department of Computer Science and Numerical Analysis of the University of Cordoba, Campus de Rabanales, 14071, Cordoba, Spain
fernandezcaballero@gmail.com, {chervas,i02gupep,i42crram}@uco.es
[2] Department of Management and Quantitative Methods, ETEA, Escritor Castilla Aguayo 4, 14005, Cordoba, Spain
fjmestud@etea.com

Abstract. This work proposes a Multiobjective Differential Evolution algorithm based on dominance Pareto concept for multiclassification problems using multilayer perceptron neural network models. The algorithm include a local search procedure and optimizes two conflicting objectives of multiclassifiers, a high correct classification rate and a high classification rate for each class, of which the latter is not usually optimized in classification. Once the Pareto front is built, we use two automatic selection methodologies of individuals: the best model with respect to accuracy and the best model with respect to sensitivity (extremes in the Pareto front). These strategies are applied to solve six classification benchmark problems obtained from the UCI repository. The models obtained show a high accuracy and a high classification rate for each class.

Keywords: Accuracy, Differential Evolution, Local Search, Multiclassification, Multiobjective, Neural Networks, Pareto, Sensitivity.

1 Introduction

Pattern classification occurs when an object needs to be assigned into a predefined class based on a number of observed attributes related to that object. Different methods for pattern classification [1] are shown in the literature, but in recent years Artificial Neural Networks (ANNs) have been an important tool for it [2].

Training Artificial Neural Networks by Evolutionary Pareto-based algorithms [3] is known as Multiobjective Evolutionary Artificial Neural Networks (MOEANNs), and it has been used in recent years to solve classification tasks, having some of its main exponents in H. Abbass [4] and Y. Jin [3].

In this paper we present a Memetic Pareto Differential Evolution (MPDE) algorithm, which is, a MultiObjective Evolutionary Algorithm (MOEA) [5] based on Differential Evolution (DE) [6] and on the Pareto dominance concept for solving multiclass classification problems. MPDE is improved with a local search algorithm, specifically with the *improved Resilient Backpropagation* (*iRprop+*) [7].

E. Corchado et al. (Eds.): HAIS 2009, LNAI 5572, pp. 433–441, 2009.
© Springer-Verlag Berlin Heidelberg 2009

Many techniques to improve the overall generalization capability for the classifier designed have been proposed, but a few maintain the classification capability in all classes (correctly classified rate per class), something that, in some datasets, is essential to ensure the benefits of a classifier against another. The objective pursued when using MOEAs in classifications with ANNs is mainly designing classifiers with the biggest possible accuracy and with a small structural complexity [3], [8]. Our proposal aims to achieve a high classification rate in the testing dataset with a good classification for each class. There are multi-objective works for classification that optimize the Accuraccy and the Sensitivity or Specificity, but only work with two classes or compare one of the classes with the rest.

The rest of the paper is organized as follows: in section 2 the accuracy and sensitivity measures are proposed and their properties are briefly discussed. Section 3 presents a brief overview about DE in Multiobjective Evolutionary Neural Networks. Section 4 describes the MPDE algorithm. Section 5 shows the experimental design, and finally the conclusions are drawn in Section 6.

2 Accuracy Versus Sensitivity

Accuracy cannot capture all the different behavioral aspects found in two different classifiers [9] so, in this section, we present two measures to evaluate a classifier:

- Accuracy $C = (1/N)\sum_{j=1}^{Q} n_{jj}$, that represents the number of times that the patterns are correctly predicted by a classifier with Q classes and N training or testing patterns, and where n_{jj} is the number of patterns from class *j-th* that are correctly classified.
- Minimum Sensitivity $S = \min\{S_i; i = 1,\ldots,Q\}$ given by the minimum of the classification rate per class, where $S_i = n_{ii} / \sum_{j=1}^{Q} n_{ij}$, being $\sum_{i,j=1}^{Q} n_{ij} = N.$ Henceforth when we talk about sensitivity, we refer to the minimal sensitivity of all classes.

Assuming that all misclassifications are equally costly and there is no profit for a correct classification, we understand that a good classifier should obtain a high accuracy level as well as an acceptable level for each class, the two-dimensional measure (S,C) is considered in this work for this reason.

Let us consider a Q –class classification problem. Let C and S be respectively the accuracy and the sensitivity associated with a classifier g , then $S \le C \le 1-(1-S)p^*$, where p^* is the minimum of the estimated prior probabilities. Therefore, each classifier will be represented as a point in the triangular region in Fig. 1 part B. Simultaneously minimize the $Q(Q-1)$ misclassification rates given by the off-diagonal elements of the confusion matrix has a main shortcoming, the dimension of the Pareto optimal front grows at the rate of the square of the number of classes, making the resolution of the problem extremely complex.

The feasible region within the (S,C) space is reduced considerably as we approach to the $(1,1)$ point; not taking the Pareto front obtained by multiobjective techniques a

great diversity in terms of number of elements. It should be noted that for a fixed value of Accuracy C, a classifier will be better when it corresponds to a point nearer to the diagonal of the square. In general, accuracy and sensitivity could be cooperative, but as we approach the $(1,1)$ point or optimum, the objectives become competitive and an improvement in one objective tends to involve a decrease in the other one, which justifies the use of a MOEA.

3 Differential Evolution in Multiobjective Evolutionary Artificial Neural Networks

A particular and simple yet powerful Evolutionary Algorithm (EA) that has been used for multiobjective optimization on ANNs is the Differential Evolution (DE) algorithm proposed by Price and Storn [6]. The main idea in DE with respect to EAs is to use vector differences in the creation of new candidate solutions $C[i]$ as one of the i elements in a population of size N. All applications of DE are distinguished by the s trategy used to create and insert new individuals in the population and by the self-adaptation of the parameters of crossover and mutation [10].

DE is used in the literature for multiobjective optimization and applications, and to a lesser extent, for the design of ANNs in classification. DE works well when the objective function has features such as nonlinearity, high dimensionality, the existence of multiple local optimal, undifferentiated or noise. For these reasons and because the article by Abbass [4] has been widely cited and used we have done an improved version of their algorithm.

To the best of our knowledge, sensitivity is nowhere used for improving the capability of generalization, quality and comparison between classifiers. Abbass [4] was one of the first authors in apply DE in Multiobjective Problems with ANNs and, in several works, he employs DE with/within local search procedures to create new individuals and to keep only the nondominated ones as the basis for the next generation, but the objectives to optimize are the accuracy and the net complexity. Ning [11] uses a Modified Differential Evolution algorithm introducing the reorganization of Evolution Strategies during the mutation and optimizing the weights of the feed-forward multilayer neural network, but only uses the mean square error as objective function.

4 The Memetic Pareto Multiobjective Evolutionary Differential Evolution Algorithm (MPMEDE)

4.1 Base Classifier Framework and Objective Functions

We consider standard feed forward MLP neural networks with one input layer with k inputs variables of the problem, one hidden layer with m maximum sigmoidal basis functions which depends on the problem, and one linear output layer with J outputs, one for each class in the problem. In this way, the functional model considered is the

following: $f_l(\mathbf{x},\boldsymbol{\theta}_l) = \beta_0^l + \sum_{j=1}^{M} \beta_j^l \sigma_j(\mathbf{x},\mathbf{w_j}),\ \ l = 1,2,...,J$, where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1,...,\boldsymbol{\theta}_J)^T$ is the transpose matrix containing all the neural net weights, $\boldsymbol{\theta}_l = (\beta_0^l, \beta_1^l,...,\beta_M^l, \mathbf{w}_1,...,\mathbf{w}_M)$ is the vector of weights of the l output node, $\mathbf{w}_j = (w_{1j},...,w_{Kj})$ is the vector of weights of the connections between input layer and the j hidden node, $\mathbf{x}$ is the input pattern and s the sigmoidal basis function.

We interpret the outputs of neurons on the output layer from a probability point of view, which considers the softmax activation function given by the following expression: $g_l(\mathbf{x},\boldsymbol{\theta}_l) = \dfrac{\exp f_l(\mathbf{x},\boldsymbol{\theta}_l)}{\sum_{l=1}^{J} \exp f_l(\mathbf{x},\boldsymbol{\theta}_l)}, l = 1,2,...,J$, where $g_l(\mathbf{x},\boldsymbol{\theta}_l)$ represents the probability of pattern $\mathbf{x}$ belonging to class *j*. Taking this consideration into account, it can be seen that the class predicted by the neuron net corresponds to the neuron on the output layer whose output value is the greatest.

In this multiobjective context we consider two multiobjective functions to maximize, where the first function is cross-entropy error and is given by the following expression for J classes: $l(\boldsymbol{\theta}) = -\dfrac{1}{N}\sum_{n=1}^{N}\sum_{l=1}^{J} y_n^{(l)} \log g_l(x_n,\boldsymbol{\theta}_l)$, where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1,...,\boldsymbol{\theta}_J)$. The advantage of using the error function $l(\boldsymbol{\theta})$ instead of $(1-C)$ is that it is a continuous function, then small changes in network parameters produce small changes in the fitness function, which allows improve the convergence. Then, the first fitness measure to maximize is a strictly decreasing transformation of the entropy error $l(\boldsymbol{\theta})$ given by $A(g) = \dfrac{1}{1+l(\boldsymbol{\theta})}$, where g is a sigmoidal basis function neural network model represented by the multivaluated function $g(\mathbf{x},\boldsymbol{\theta}) = \left(g_1(\mathbf{x},\boldsymbol{\theta}_1),...,g_J(\mathbf{x},\boldsymbol{\theta}_J)\right)$. The second objective to maximize is the sensitivity $S(g)$ of the classifier as the minimum value of the sensitivities for each class. Both $A(g)$ and $S(g)$ fitness functions, are necessary for the evaluation of the individuals in Algorithm 1 (see step 3).

4.2 MPDE Algorithm

In Algorithm 1 we describe our Memetic Pareto Differential Evolution (MPDE) algorithm. The approach evolves architectures and connection weights simultaneously, each individual being a fully specified ANN. The ANNs are represented using an object-oriented approach and the algorithm deals directly with the ANN phenotype. The fundamental characteristics are the following:

- The maximum number of non-dominated solutions in each generation was set to $(populationSize / 2)$. If it is exceeded, a nearest neighbor distance function [12] is adopted by preventing a agglomerative structure of the Pareto front (step 9-13).
- *Crossover* operator is proposed from step 17, where three parents have been previously selected randomly; being the child a perturbation of the main parent. First, with some probability P_c for each hidden neuron, h , if the probability is met, the

Algorithm 1. Memetic Multiobjective Differential Evolution (MPDE)

1: Create a random initial population of potential solutions.
2: **Repeat**
3: Evaluate the individuals in the population and label those who are non-dominated.
4: **If** the number of non-dominated individuals is less than 3 **then**
5: **Repeat**
6: Find a non-dominated solution among those who are not labeled.
7: Label the solution as non-dominated.
8: **Until** the number of non-dominated individuals is greater than or equal to 3.
9: **Else If** number of non–dominated solutions is greater than $(populationSize / 2)$ **then**
10: **Repeat**
11: Calculate the distance of each individual with its nearest neighbor.
12: Delete the individual with smaller distance.
13: **Until** the number of non-dominated individuals is equal to $(populationSize / 2)$.
14: Delete all dominated solutions from the population.
15: **Repeat**
16: Select at random an individual as the main parent $a1$ and two individuals, $a2$, $a3$ as supporting parents.
17: **Crossover:** with a crossover probability P_c for each hidden neuron, do
18: $$r_h^{child} \leftarrow \begin{cases} 1 & if \left(r_h^{a1} + N(0,1)\left(r_h^{a2} - r_h^{a3}\right)\right) \geq 0.5 \\ 0 & otherwise \end{cases}$$
19: $$w_{ih}^{child} \leftarrow w_{ih}^{a1} + N(0,1)\left(w_{ih}^{a2} - w_{ih}^{a3}\right)$$
20: otherwise
21: $$w_{ih}^{child} \leftarrow w_{ih}^{a1}$$
22: $$r_h^{child} \leftarrow r_h^{a1}$$
23: and with crossover probability P_c for each output neuron, do
24: $$w_{ho}^{child} \leftarrow w_{ho}^{a1} + N(0,1)\left(w_{ho}^{a2} - w_{ho}^{a3}\right)$$
25: otherwise
26: $$w_{ho}^{child} \leftarrow w_{ho}^{a1}$$
27: **If** the child is equal to the main parent **then**
28: A random link is perturbed by adding a Gaussian distribution $N(0,1)$.
29: **Mutation:** with a mutation probability P_m for each neuron do
30: $$r_h^{child} \leftarrow \begin{cases} 1 & if\ r_h^{child} = 0 \\ 0 & otherwise \end{cases}$$
31: A child has been created. Store the best child so far. NumCreated ← NumCreated + 1
32: **If** the candidate dominates the parent **then**
33: Apply *iRprop+* local search to the child.
34: Add the candidate to the population.
35: **Else If** there is no dominance relation between main parent and child **then**
Add the candidate to the population.
36: **Else If** NumCreated = 100 (here the main parent dominates to the child) **then**
37: Add the best of these 100 children to the population.
38: NumCreated ← 0
39: **Else** The candidate is discarded and to go to step 15 (No child is added).
40: **Until** the population size is N.
41: **Until** termination conditions are satisfied, go to 2 above.

neuron selected in the child will be maintained ($r = 1$) or deleted ($r = 0$), depending of the value of the expression that is shown in step 18. In the first case the weight w_{ih} between the *i-th* input variable and the *h-th* hidden node will be modified by the expression proposed in step 19. If the crossover probability is not met then the structure of the main parent is inherited by the child (steps 21-22). Third, a similar weight modification is reached with a P_c probability for each output neuron, o , in the output layer (steps 23-26).

- *Mutation* operator consists on adding or deleting neurons in the hidden layer depending on a P_m probability for each them. Taking into account the maximum number of hidden neurons that may exist in an individual in a specific problem, the probability will be used as many times as number of neuron has the classifier. If the neuron exists, is deleted, but if it does not exist, then it is created and the weights are established randomly, see step 29.
- Local search, steps 32-34, has been carried out based in the adaptation of a version of the *Resilient Backpropagation* (*Rprop)*, the improved Rprop or *iRprop+* [7]. The adaptation is made using a backtracking strategy to the *softmax* activation function and to the cross-entropy error function, modifying the gradient vector. The local search is applied only to the child that dominates to the main parent, after the crossover and mutation have been applied, decreasing in this way the computational cost. Other works perform this operation for each child created before checking if the child dominates or not to the main parent.
- There are significant differences with the Abbass' algorithm proposed in [4]. First, in the crossover we used a P_c probability for each neuron and not for each layer as Abbass does, being our algorithm less aggressive with the changes in the ANNs. The mutator probability also is used in independent way for each neuron and not for the hidden layer; because we believe that the changes proposed by Abbass produce such drastic changes in the ANNs, in which their generalization capability can be reduced significantly. Third, the way in which individuals are added to the population, Abbass adds to the population only those children who dominate the main parent and this decision may leave the algorithm running between the steps 15-40 for a long time, because when the number of generations increase is more difficult to improve the main parent. In our case, the way individuals are added in steps 31-39 is more relaxed, so children that dominates or not to the main parent can be added. In this way the computational time is reduced.

5 Experiments

For the experimental design we consider 6 datasets taken from the UCI repository [16], Autos, Balance, Breast-Cancer, Newthyroid, Pima and HeartStatlog, with 6, 3, 2, 3, 2 and 2 classes respectively. The design was conducted using a stratified holdout procedure with 30 runs, where 75% of the patterns were randomly selected for the training set and the remaining 25% for the test set. The population size is established to $N_p = 100$. The crossover probability is established to 0.8 and the mutation

probability to 0.1. For *iRprop+* the adopted parameters are $h^- = 0.5$, $h^+ = 1.2$, $D_0 = 0.0125$ (the initial value of the D_{ij}), $D_{max} = 50$, $D_{min} = 0$ and $Epochs = 5$.

Once the Pareto front is built in each run we use two automatic selection methodologies of individuals: First, the extreme values in training are chosen, that is, the best individual on Entropy, EI, and the best individual on Sensitivity, SI (see Fig 1 A). Once this is done, we get the values of Accuracy C and Sensitivity S in testing of EI, $EI_{testing} = (C_{EI\_testing}, S_{EI\_testing})$ and SI, $SI_{testing} = (C_{SI\_testing}, S_{SI\_testing})$. This is repeated for each run and the average and standard deviation from the EI and SI individuals are estimated obtaining $\overline{EI}_{testing} = (\overline{C}_{EI\_testing}, \overline{S}_{EI\_testing})$ and $\overline{SI}_{testing} = (\overline{C}_{SI\_testing}, \overline{S}_{SI\_testing})$. Therefore, the first expression $\overline{EI}_{testing}$ is the average obtained taking into account the Entropy as primary objective when we choose an individual from the first Pareto front, and the second $\overline{SI}_{testing}$ taking into account the Sensitivity, getting two automatic methodologies called MPDE-E and MPDE-S respectively.

We compare our algorithm with a modified and memetic version of NSGA2 (for details see [13]), which we also have used for designing ANNs models in the same framework shown in this work, using *iRprop+* and a mutation operator, although other implementations can be found in the framework Paradiseo-MOEO [14] . Also, we compare with the SVM methodology from the SMO algorithm with the defaults values that provides Weka [15].

In Table 1 we present the values of the average and the standard deviation for C and S obtained for the best models in each run over the testing set. We can observe that in Balance and Breast-Cancer, MPDE-S obtains the best values in S, and very close to those modified NSGA2 in C. In Autos, the best result in C is obtained by MPDE-E but the best value in S is achieved by MNSGA2-S. In Newthyroid MPDE obtains the best values in S and C, and in Pima and HeartStatlog, MPDE-S obtains the best values in S and very similar to those obtain by MNSGA2-E in C.

Table 1. Statistical results for MPDE and the modified NSGA2 version, MNSGA2, in testing. In **bold** the best result and in *Italic* the second best result.

Dataset	Algorithm	C(%)	S(%)	Dataset	Algorithm	C(%)	S(%)
	MPDE-E	**68.79±5.59**	28.75±21.40		MPDE-E	91.43±1.01	54.36±26.25
	MNSGA2-E	*66.67±4.07*	*39.64±14.92*		MNSGA2-E	**94.01**±1.52	42.66±17.00
Autos	MPDE-S	64.15±5.63	12.26±20.54	Balance	MPDE-S	91.41±1.53	**87.42±4.32**
	MNSGA2-S	66.04±4.78	**42.28±10.98**		MNSGA2-S	*92.47±2.16*	*83.72±8.19*
	SVM	67.92	0.00		SVM	88.46	0.00
	MPDE-E	*67.27±2.71*	38.09±11.59		MPDE-E	*96.66±2.02*	*81.42±10.74*
	MNSGA2-E	**69.34±2.30**	28.88±9.09		MNSGA2-E	95.12±2.30	74.81±10.07
BreastC	MPDE-S	65.39±3.40	**57.04±7.01**	Newthy	MPDE-S	**96.66±1.84**	**81.64±9.76**
	MNSGA2-S	63.99±3.10	*53.08±6.57*		MNSGA2-S	95.55±2.15	75.07±10.66
	SVM	64.79	23.81		SVM	88.89	55.56
	MPDE-E	*78.59±1.59*	61.94±4.10		MPDE-E	76.17±1.41	61.11±2.20
	MNSGA2-E	**78.99±1.80**	60.44±2.59		MNSGA2-E	**78.28±1.75**	61.88±2.08
Pima	MPDE-S	77.11±2.20	**73.12±2.98**	HeartStlg	MPDE-S	*76.27±1.57*	**63.66±2.37**
	MNSGA2-S	76.96±2.08	*72.68±3.06*		MNSGA2-S	77.5±1.73	*62.66±2.38*
	SVM	78.13	50.75		SVM	76.47	60.00

In Fig. 1 we can see the results obtained by MPDE for Balance dataset in the (S,C) space in one specific run, which presents the best individual on Entropy in training. Observe (Fig 1. A) that the (S,C) values do not form Pareto fronts in testing (Fig 1. B), and the individuals which in the training graphics were in the first Pareto front, can now be located within the (S,C) space in a worst region, since there is no direct relation between training Entropy and testing Accuracy C.

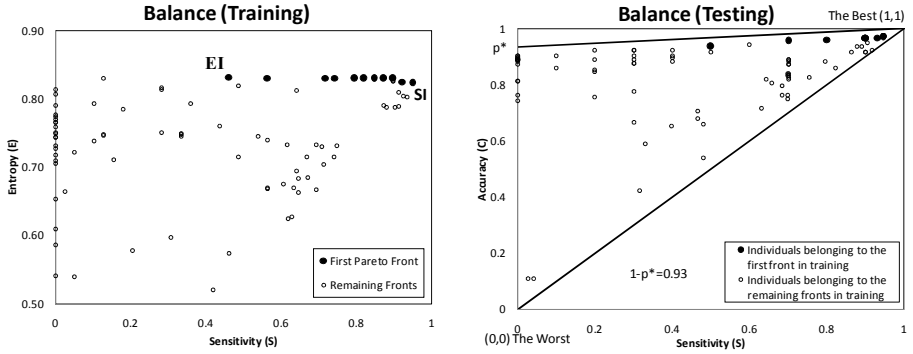

Fig. 1. A) Pareto front in training. B) Feasible region in the (S,C) space for testing.

6 Conclusions

The methodology uses a MOEA which tries to boost Accuracy and Sensitivity as conflicting objectives. A memetic version of DE with the *iRprop+* algorithm as local optimizer, designs the ANNs architecture finding the adequate number of neurons in the hidden layer and the optimal number of connections along with the corresponding weights. The features of the Pareto optimal front allowed us to consider two automatic selection methodologies of individuals: the best model in accuracy and the best model in sensitivity (extremes in the Pareto front). Through optimizing both measures, as is shown in the results, it is possible to obtain classifiers that combine a high classification level with a very good classification rate for each class. In our opinion, the perspective and the memetic DE approach reveal a new point of view for dealing with multi-classification problems.

Acknowledgements

This work has been partially subsidized by TIN 2008-06681-C06-03 project of the Spanish Ministerial Commission of Science and Technology (MICYT), FEDER funds and the P08-TIC-3745 project of the "Junta de Andalucía" (Spain). The research done by J.C. Fernández and P.A. Gutiérrez has been financed respectively by the FPI (grant reference BES-2006-12543) and FPU (grant reference AP-2006- 01746) Predoctoral Programs (Spanish Ministry of Education and Science).

References

1. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley Interscience, New York (2000)
2. Zhang, G.P.: Neural Networks for Classification: A Survey. IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews 30, 451–462 (2000)
3. Jin, Y., Sendhoff, B.: Pareto-Based Multiobjective Machine Learning: An Overview and Case Studies. IEEE Transaction on Systems, Man and Cybernetics, Part. C: Applications and reviews 38, 397–415 (2008)
4. Abbass, H.: An Evolutionary Artificial Neural Networks Approach for Breast Cancer Diagnosis. Artificial Intelligence in Medicine 25, 265–281 (2002)
5. Coello, C.A., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-Objective Problems. Springer, Heidelberg (2007)
6. Storn, R., Price, K.: Differential Evolution. A fast and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359 (1997)
7. Igel, C., Hüsken, M.: Empirical evaluation of the improved Rprop learning algorithms. Neurocomputing 50, 105–123 (2003)
8. Braga, A.P., Takahashi, R.H.C., Costa, M.A., Teixeira, R.A.: Multi-objective Algorithms for Neural Networks Learning. Studies in Computational Intelligence 16 (2006)
9. Provost, F., Fawcett, T.: Robust classification system for imprecise environments. In: Proccedings of the Fithteenth National Conference on Artificial Intelligence, pp. 706–713 (1998)
10. Zielinski, K., Laur, R.: Variants of Differential Evolution for Multi-Objective Optimization. In: Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multicriteria Decision Making (MCDM 2007), pp. 91–98 (2007)
11. Ning, G., Zhou, Y.: A Modified Differential Evolution Algorithm for Optimization Neural Network
12. Abbass, H.A., Sarker, R., Newton, C.: PDE: a Pareto-frontier differential evolution approach formulti-objective optimization problems. In: Proceedings of the 2001 Congress on Evolutionary Computation, Seoul, South Korea, vol. 2 (2001)
13. Fernández, J.C., Gutiérrez, P.A., Hervás, C., Martínez, F.J.: Memetic Pareto Evolutionary Artificial Neural Networks for the determination of growth limits of Listeria Monocytogenes. In: Hybrid Intelligent Systems Conference, HIS 2008, pp. 631–636. IEEE, Barcelona (2008)
14. Liefooghe, A., Basseur, M., Jourdan, L., Talbi, E.: ParadisEO-MOEO: A Framework for Evolutionary Multi-objective Optimization. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 386–400. Springer, Heidelberg (2007)
15. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)