Pareto Neuro-Evolution: Constructing Ensemble of Neural Networks Using Multi-objective Optimization

Hussein A. Abbass
Artificial Life and Adaptive Robotics (A.L.A.R.) Lab,
School of Information Technology and Electrical Engineering,
Australian Defence Force Academy,
University of New South Wales, Canberra, ACT 2600, Australia,
h.abbass@adfa.edu.au

Abstract- In this paper, we present a comparison between two multi-objective formulations to the formation of neuro-ensembles. The first formulation splits the training set into two non-overlapping stratified subsets and form an objective to minimize the training error on each subset, while the second formulation adds random noise to the training set to form a second objective. A variation of the *memetic Pareto artificial neural network* **(MPANN) algorithm is used. MPANN is based on differential evolution for continuous optimization. The ensemble is formed from all networks on the Pareto frontier. It is found that the first formulation outperformed the second. The first formulation is also found to be competitive to other methods in the literature.**

1 Introduction

A mixture of predictors/classifiers (ensemble) provides a better bias–variance trade-off which usually results in an improvement in the prediction accuracy [14]. Many studies [17, 18, 19, 20, 26] focused on the construction of ensembles using *artificial neural networks* (ANNs), more specifically multi-layer feed-forward artificial neural networks. The field is usually called neuro-ensemble, to denote the use of ANN to form an ensemble.

In this paper, we have chosen to compare our work against the *negative correlation learning* (NCL) method of [19] because the latter can be seen as one of the recent well-tested work that was also combined with evolutionary computations to evolve an ensemble of neural networks [20] and has also been used successfully for feature selection tasks [7].

Liu and Yao [19] proposed the NCL algorithm for training an ensemble of ANNs using Backpropagation [18]. A mathematical penalty term describing the negative correlation between the networks is added to the conventional mean square error of each network and traditional *backpropagation* (BP) [23] is used for network training. In a study presented in [21], an analysis of NCL revealed that the penalty function in NCL acts to maximize the average difference between each network and the mean of the population; while the intended aim of anti-correlation mechanisms is to maximize the average difference between pairs of population members, which is not necessarily the same thing.

In NCL, the ensemble is trained using the training set, where the output of the ensemble F^p is given by the following equation:

$$F^p = \frac{1}{M}\sum_{m=1}^{M}\hat{Y}^p(m) \tag{1}$$

The expected error of pattern p is given by the average error of the ensemble, $Error^p$, as defined by the following two Equations:

$$Error^p = \frac{1}{P}\sum_{p=1}^{P}Error^p(m) \tag{2}$$

$$Error^p(m) = \frac{1}{P}\sum_{p=1}^{P}\frac{1}{2}(\hat{Y}^p(m) - Y^p)^2 + \frac{1}{P}\sum_{p=1}^{P}\lambda\Phi^p(m) \tag{3}$$

Here, $\Phi^p(m)$ is the penalty function of network m and pattern p. This function represents the correlation term that we need to minimize. In NCL, the following is used as the anticorrelation measure:

$$\Phi^p(m) = (\hat{Y}^p(m) - F^p)\sum_{l\neq m}(\hat{Y}^p(l) - F^p) \tag{4}$$

The previous function has some desirable characteristics, as when it is combined with the mean square error, the result will be a nice tradeoff between the bias, variance and co-variance. Moreover, it does not change the BP algorithm much as it only adds a simple term to the derivative of the error as follows:

$$\frac{\partial Error^p(m)}{\partial \hat{Y}^p(m)} = (\hat{Y}^p(m) - Y^p) - \lambda(\hat{Y}^p(m) - F^p) \quad (5)$$

NCL was then combined with an evolutionary approach [20] to evolve the ensemble. The k-means algorithm was used to cluster the individuals in the population. The fittest individual in each cluster is used as a member in the ensemble. The authors found no statistical difference between the use of the population as a whole in the ensemble and the use of a subset. It is not clear however how to choose the value of k.

Three key open research questions remain in the literature:

1. On what basis should a network be included in, or excluded from the ensemble?
2. How should the ensemble size be determined?
3. How to ensure that the networks in an ensemble are different?

The objective of this paper is to attempt to answer these three questions. This is a challenging task and this paper should be seen as only an attempt to provide a theoretically-sound answer to these questions. The rest of the paper is organized as follows: In Section 2, background materials are covered followed by an explanation of the methods in Section 3. Results are discussed in Section 4 and conclusions are drawn in Section 5.

2 Multiobjective optimization

Consider a MOP as presented below:-

$$\textbf{Optimize } F(\vec{x} \in \Upsilon) \quad (6)$$

$$\textbf{Subject to: } \Upsilon = \{\vec{x} \in R^n | G(\vec{x}) \leq 0\} \quad (7)$$

where $\vec{x}$ is a vector of decision variables $(x_1, \ldots, x_n)$ and $F(\vec{x} \in \Upsilon)$ is a vector of objective functions $(f_1(\vec{x} \in \Upsilon), \ldots, f_K(\vec{x} \in \Upsilon))$. Here $f_1(\vec{x} \in \Upsilon), \ldots, f_K(\vec{x} \in \Upsilon)$, are functions on R^n and Υ is a nonempty set in R^n. The vector $G(\vec{x})$ represents a set of constraints.

The aim is to find the vector $\vec{x}^* \in \Upsilon$ which optimizes $F(\vec{x} \in \Upsilon)$. Without any loss of generality, we assume that all objectives are to be minimized. We note that any maximization problem can be transformed to a minimization one by multiplying the former by -1.

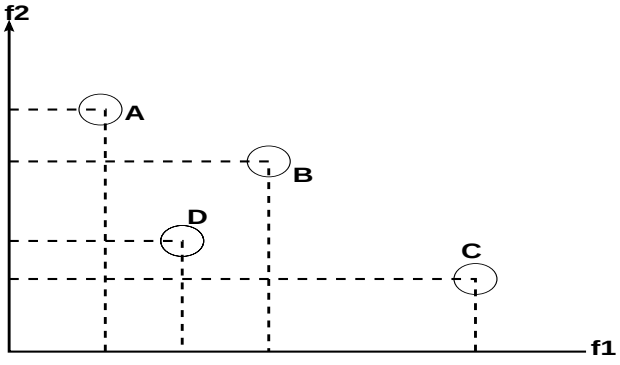

Figure 1: The concept of dominance in multiobjective optimization. Assuming that both f_1 and f_2 are to be minimized, B is dominated by D since D is better than B when measured on all objectives. However, A, C and D are non–dominated since none of them is better than the other two when measured on all objectives.

The core aspect in the discussion of a *multiobjective optimization problem* (MOP) is the possible conflict that arises in the case of optimizing the objectives simultaneously. At a certain point, one objective can just be improved at the expense of at least another objective. The principle of dominance (Figure 1) in MOP allows a partial order relation that works as follows: a solution does not have an advantage to be included in the set of optimal solutions unless there is no solution that is better than the former when measured on all objectives. A non–dominated solution is called Pareto. To formally define the concept of non–dominated solutions in MOPs, we need to define two operators, $\ngeq$ and $\precnsim$ and then assume two vectors, $\vec{x}$ and $\vec{y}$. $\vec{x} \ngeq \vec{y}$ iff $\exists\ x_i \in \vec{x}$ and $y_i \in \vec{y}$ such that $x_i \neq y_i$. And, $\vec{x} \precnsim \vec{y}$ iff $\forall\ x_i \in \vec{x}$ and $y_i \in \vec{y}, x_i \leq y_i$, and $\vec{x} \ngeq \vec{y}$. $\ngeq$ and $\precnsim$ can be seen as the "not equal to" and "less than" operators over two vectors, respectively. We can now define the concepts of local and global optimality in MOPs.

Definition 1 Global efficient (non–inferior/ pareto–optimal) solution *A vector $\vec{x}^* \in \Upsilon$ is said to be a global efficient solution of MOP iff $\nexists\ \vec{x} \in \Upsilon$ such that $F(\vec{x}) \precnsim F(\vec{x}^*)$.*

Definition 2 Global non–dominated solution *A vector $\vec{y}* \in F(\vec{x})$ is said to be global non–dominated solution of MOP iff its projection onto the decision space, $\vec{x}^*$, is a global efficient solution of MOP.*

A MOP can be solved in different ways. One method is

by taking a weighted sum of the objectives. However, this is not an efficient way to solve the problem [24]. First, the weighted sum method could only generate a single non-dominated solution at a time. Second, it assumes convexity of the Pareto front. Third, the question of determining the appropriate values for the weights remains unsolved; that is, we will need to run the algorithm a number of times with different weights. Another method for solving MOPs is through the use of goal programming, where an aspiration level is set for each objective, transforming each objective to an equality after adding underachievement and overachievement deviations, then a single objective is constructed through the minimization of a prioritized and/or weighted sum of the relevant deviations. This method is biased to the choice of the aspiration levels.

A third method for solving MOP, ϵ constraint, is to sequentially optimize each objective. The way it works is as follows: select one of the objective and construct a single objective optimization problem ignoring the other objectives. The optimum of the single objective problem in conjunction with the corresponding objective function are used to form a constraint while ϵ is the right hand side of this constraint representing the amount the decision maker is willing to sacrifice in the corresponding optimal objective value. After adding this constraint to the constraint set of the problem, another objective is selected and the process is repeated until either all objectives are selected or the solution does not change. An obvious drawback to this approach is that it is sensitive to the order of optimizing the objectives and the value of ϵ.

A fourth group of methods for solving MOPs generate a single specific Pareto solution. Compromise programming and Benson's method are representatives of this class. A fifth group of methods are called interactive methods, where an interactive session is established with the decision maker to find satisfactory solutions.

The last group of methods is evolutionary algorithms (EAs) [8, 9]. EAs offered something different from all other methods. Being population based, they are able to generate a set of near-Pareto solutions in a single run. In addition, they do not require assumptions of convexity, differentiability, and/or continuity as traditional optimization problems do. EAs with local search are usually used to improve the performance of EAs to get closer to the actual optimal or, the Pareto set in case of MOPs.

In EANN, an evolutionary algorithm is used for training the ANN. A major advantage to the evolutionary approach over BP alone is the ability to escape a local optimum. The major disadvantage of the EANN approach is that it is computationally expensive, as the evolutionary approach is normally slow.

Recently, the problem of simultaneous optimization of the network architecture and the corresponding training error has been casted as a multiobjective optimization problem [4]. It was found that by combining BP with an EMO algorithm, a considerable reduction in the computational cost can be achieved [4]. The multiobjective optimization outputs a set of solutions, where no solution in the set is better than all others when compared on all objectives. The Pareto set provided a new insight into the learning problem, where an obvious question that emerged from the study is how to utilize the information embedded in the set as a whole. One answer to this question is to form an ensemble, which is the focus of this paper.

3 Formation of Neural Networks ensembles

3.1 Evolutionary artificial neural networks

The following notations will be used for a single hidden layer ANN:

- I and H are the number of input and hidden units respectively.
- $\tilde{\mathbf{X}}^p \in \mathbf{X} = (x_1^p, x_2^p, \ldots, x_I^p), p = 1, \ldots P$, is the pth pattern in the input feature space $\mathbf{X}$ of dimension I, and P is the total number of patterns.
- Without any loss of generality, $\mathbf{Y}_o^p \in \mathbf{Y}_o$ is the corresponding scalar of pattern $\mathbf{X}^p$ in the hypothesis space $\mathbf{Y}_o$.
- w_{ih} and w_{ho}, are the weights connecting input unit $i,\ i = 1 \ldots I$, to hidden unit $h,\ h = 1 \ldots H$, and hidden unit h to the output unit o respectively. The number of outputs is assumed to be 1 in this paper.
- $\Theta_h(\tilde{\mathbf{X}}^p) = \sigma(a_h)$; $a_h = \sum_{i=1}^{I} w_{ih} x_i^p,\ \ h = 1 \ldots H$, is the h^{th} hidden unit's output corresponding to the input pattern $\mathbf{X}^p$, where a_h is the activation of hidden unit h, and $\sigma(.)$ is the activation function taken in this paper to be the logistic function $\sigma(z) = \frac{1}{1+e^{-Dz}}$, with D the function's sharpness or steepness and is taken to be 1.
- $\hat{Y}_o^p = \sigma(a_o)$; $a_o = \sum_{h=1}^{H} w_{ho} \Theta_h(\tilde{\mathbf{X}}^p)$ is the network's output and a_o is the activation of output unit o corresponding to the input pattern $\tilde{\mathbf{X}}^p$.

In this paper, we use the quadratic error function by squaring the difference between the predicted and actual

output. We will also make use of the traditional BP algorithm as a local search method. For a complete description of BP, see for example [12].

3.2 The multiobjective learning problem

In casting the learning problem as a multiobjective problem, a successfully-tested formulation was presented in [4]. We will compare in this paper two different formulations more suitable for neuro-ensemble. In the first formulation, the training set is divided into two subsets using stratified sampling, where each objective corresponds to the minimization of error for each of these subsets. In the second formulation, the first objective is to minimize the error on the overall training set while the second is similar to the first with a Gaussian error being added.

Let $p1$ and $p2$ be the number of patterns/instances in the two subsets $Sub1$ and $Sub2$ respectively. The learning problem can be formulated as

Prob1

$$\textbf{Minimize } f_1 = \sum_o \sum_{p1} (\hat{Y}_o^{p1} - Y_o^{p1})^2 \tag{8}$$

$$\textbf{Minimize } f_2 = \sum_o \sum_{p2} (\hat{Y}_o^{p2} - Y_o^{p2})^2 \tag{9}$$

The second alternative is

Prob2

$$\textbf{Minimize } f_3 = \sum_o \sum_{p} (\hat{Y}_o^{p} - Y_o^{p})^2 \tag{10}$$

$$\textbf{Minimize } f_4 = \sum_o \sum_{p} (\hat{Y}_o^{p} - Y_o^{p})^2 + N(0, 0.2) \tag{11}$$

In $Prob1$, there is no guarantee that the ensemble size would be more than 1. Take the case where a network is large enough to over-fit on both $Sub1$ and $Sub2$, where the training error on each of them is zero. In this case, the Pareto set would have a single solution. It is therefore our assumption that we choose a network size small enough not to over-fit the union of $Sub1$ and $Sub2$, while it is large enough to learn each of them independently. In $Prob2$, it is unlikely that the ensemble size will be 1 since it is impossible to find a network which satisfies both objectives and achieve error on both equal to 0.

Our second assumption relates to the algorithm. Getting to the actual Pareto frontier is always a challenge for any algorithm. We do not assume that we have to get to the actual Pareto frontier for this method to work. The networks on the Pareto set returned by the algorithm will represent still a bias-variance trade-off and therefore, will be potentially good.

The ensemble is formed from all networks on the Pareto frontier found by the evolutionary method. We have used three methods for forming the ensemble's gate. In voting, the predicted class is the one where the majority of networks agreed. In winner-take-all, the network with the largest activation in the ensemble is used for prediction. In simple averaging, the activations for all networks in the ensemble are averaged and the class is determined using this average.

3.3 The MPANN algorithm

Recently, we developed the *Pareto differential evolution* (PDE) method [5]. The algorithm is an adaptation of the original *differential evolution* (DE) introduced by Storn and Price [28] for optimization problems over continuous domains. There is a large number of *evolutionary multi-objective optimization* (EMO) algorithms in the literature [8, 9, 10, 11, 13, 27, 29, 29, 29, 25, 15, 16], but we selected PDE as it has been tested successfully in the area of neural networks [2, 4].

PDE generally improves over traditional differential evolution algorithms because of the use of Gaussian distribution, which spreads the children around the main parent in both directions of the other two supporting parents. Since the main parent is a non-dominated solution in the current population, it is found [3] that the Gaussian distribution helps to generate more solutions on the Pareto-front; which is a desirable characteristic in EMO. PDE was also tested successfully for evolving neural networks, where it is called *Memetic Pareto Artificial Neural Network* (MPANN) algorithm [1, 2, 4].

1. Create a random initial population of potential solutions. The elements of the weight matrix Ω are assigned uniformally distributed random values $U(0, 1)$.
2. Apply BP to all individuals in the population.
3. Repeat
   (a) Evaluate the individuals in the population and label the non-dominated ones.
   (b) If the number of non-dominated individuals is less than 3 repeat the following until the number of non-dominated individuals is greater than or

equal to 3 so that we have the minimum number of parents needed for crossover:

i. Find a non-dominated solution among those who are not marked.

ii. Mark the solution as non-dominated.

(c) Delete all dominated solutions from the population.

(d) Repeat

i. Select at random an individual as the main parent α_1, and two individuals, α_2, α_3 as supporting parents.

ii. **Crossover:** For all weights do

$$\omega_{ih}^{child} \leftarrow \omega_{ih}^{\alpha_1} + N(0,1)(\omega_{ih}^{\alpha_2} - \omega_{ih}^{\alpha_3}) \quad (12)$$

$$\omega_{ho}^{child} \leftarrow \omega_{ho}^{\alpha_1} + N(0,1)(\omega_{ho}^{\alpha_2} - \omega_{ho}^{\alpha_3}) \quad (13)$$

iii. Apply BP to the child then add the child to the population.

(e) Until the population size is M

4. Until termination conditions are satisfied.

4 Experiments

Similar to [20], we have tested MPANN on two benchmark problems; the Australian credit card assessment problem and the diabetes problem, available by anonymous ftp from ice.uci.edu [6].

The Australian credit card assessment dataset contains 690 patterns with 14 attributes; 6 numeric and 8 discrete (with 2 to 14 possible values). The predicted class is binary - 1 for awarding the credit and 0 for not. To be consistent with the literature [22], the dataset is divided into 10 folds where class distribution is maintained in each fold. Cross–validation is used where we run the algorithm with 9 out of the 10 folds for each data set, then we test with the remaining one. Similar to [20], the number of generations is 200, the population size 25, the learning rate for BP 0.003, the number of hidden units is set to 5, and the number of epochs BP was applied to an individual is set to 5 for each subset incase of Prob1.

The diabetes dataset has 768 patterns; 500 belonging to the first class and 268 to the second. It contains 8 attributes. The classification problem is difficult as the class value is a binarized form of another attribute that is highly indicative of a certain type of diabetes without having a one-to-one correspondence with the medical condition of being diabetic [22]. To be consistent with the literature [22], the dataset was divided into 12 folds where class distribution is maintained in each fold. Cross–validation is used where we run the algorithm with 11 out of the 12 folds for each dataset and then we test with the remaining one. Similar to [20], the number of generations is 200, the population size 25, the learning rate for BP 0.003, the number of hidden units is set to 5, and the number of epochs BP was applied to an individual is set to 5.

Table 1: Accuracy rates of MPANN2 for the Australian Credit Card and the diabetes datasets.

Australian credit card dataset				
	Prob1		Prob2	
	Simple averaging			
	Training	Testing	Training	Testing
Mean	0.849	0.865	0.850	0.844
SD	0.019	0.043	0.017	0.058
Min	0.803	0.797	0.809	0.724
Max	0.877	0.928	0.869	0.913
	Majority voting			
	Training	Testing	Training	Testing
Mean	0.854	0.862	0.852	0.844
SD	0.018	0.049	0.015	0.056
Min	0.812	0.783	0.821	0.724
Max	0.879	0.928	0.871	0.913
	Winner-rakes-all			
	Training	Testing	Training	Testing
Mean	0.847	0.858	0.834	0.824
SD	0.018	0.044	0.027	0.053
Min	0.805	0.797	0.768	0.724
Max	0.876	0.913	0.866	0.9
Diabetes dataset				
	Prob1		Prob2	
	Simple averaging			
	Training	Testing	Training	Testing
Mean	0.769	0.777	0.753	0.744
SD	0.023	0.032	0.019	0.034
Min	0.737	0.723	0.738	0.690
Max	0.808	0.84	0.795	0.806
	Majority voting			
	Training	Testing	Training	Testing
Mean	0.771	0.779	0.755	0.744
SD	0.023	0.033	0.019	0.034
Min	0.737	0.723	0.738	0.690
Max	0.81	0.84	0.795	0.806
	Winner-rakes-all			
	Training	Testing	Training	Testing
Mean	0.767	0.777	0.753	0.746
SD	0.022	0.037	0.020	0.032
Min	0.737	0.723	0.738	0.690
Max	0.807	0.84	0.795	0.802

The results of this experiment are presented in Table 1. It is interesting to see that the training error is slightly worse than the test error for $Prob1$. In a normal situation, this might be interpreted as under-training. The situation here is very different indeed. Let us first explain how this training error is calculated. Recall that we have divided the training set into two subsets using stratified samples. By virtue of the Pareto definition, the Pareto frontier would have networks which are over-fitting on the first subset but not on the second and other networks which are over-fitting on the second subset but not on the first. However, the sum of the bias of a machine trained on the first subset and another machine trained on the second subset would be less than a machine trained on the union of the two subsets. Therefore, in a conventional situation, where the whole training set is used as a single training set, a biased machine would usually perform better on the training and worse on the test. In $Prob1$, this is unlikely to occur since the bias and the variance are averaged over the entire Pareto-set. For $Prob2$, however, it is clear that the performance on the training is better than on testing as would be expected since the whole training set is used for training the networks.

We can also observe in Table 1 that the standard deviation on the training set is always smaller than on the test. This should be expected given that the networks are trained on two subsets which constitute the entire training set for $Prob1$ or on the entire training set as in $Prob2$.

4.1 Comparisons with other work

We compare our results against backprob[22] and Liu et.al. [20]. In Table 2, we find that the Pareto-based ensemble is better than BP and equivalent to Liu et.al. with smaller ensemble size. Although it is desirable in a new method to perform better than other methods on some dataset, the main contribution from our perspective here is that the decision on which network to include in, or exclude from, the ensemble is determined automatically by the definition of the Pareto frontier. Also, the ensemble's size is automatically determined.

5 Conclusion

In this paper, we cast the problem of training artificial neural networks as a multiobjective optimization problem and use the resultant Pareto frontier to form an ensemble. The method provides a theoretically-sound approach for formation of neuro-ensembles while answering three main questions in the neural network ensemble regarding the criteria for including a network in, or excluding it from, the ensemble, the Pareto set defines the size of the ensemble, and the Pareto concept ensures that the network in the ensemble are different.

Table 2: Comparing MPANN against other work for the Australian credit card and diabetes data sets. The three rates for MPANN and Liu et. al. represent the performance using simple average, vote, winner-takes-all strategies, respectively.

	Accuracy Rate on Test Set	
Algorithm	Australian	Diabetes
This paper	0.865,0.862,0.858	0.777,0.779,0.777
Backprob[22]	0.846	0.749
Liu et.al. [20]	0.855,0.857,0.865	0.766,0.764,0.779

Bibliography

[1] H.A. Abbass. A memetic pareto evolutionary approach to artificial neural networks. In M. Stumptner, D. Corbett, and M. Brooks, editors, *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence (AI'01)*, pages 1–12, Berlin, 2001. Springer-Verlag.

[2] H.A. Abbass. An evolutionary artificial neural network approach for breast cancer diagnosis. *Artificial Intelligence in Medicine*, 25(3):265–281, 2002.

[3] H.A. Abbass. The self-adaptive pareto differential evolution algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC2002)*, volume 1, pages 831–836, Piscataway, NJ, 2002. IEEE Press.

[4] H.A. Abbass. Speeding up back-propagation using multiobjective evolutionary algorithms. *Neural Computation*, to appear, 2003.

[5] H.A. Abbass, R.A. Sarker, and C.S. Newton. PDE: A pareto-frontier differential evolution approach for multi-objective optimization problems. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC2001)*, volume 2, pages 971–978, Piscataway, NJ, 2001. IEEE Press.

[6] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, http://www.ics.uci.edu/∼mlearn/mlrepository.html. *University of California, Irvine, Dept. of Information and Computer Sciences*, 1998.

[7] G. Brown, X. Yao, J. Wyatt, H. Wersing, and B. Sendhoff. Exploiting ensemble diversity for automatic feature extraction. In L. Wang, J.C. Rajapakse,

K. Fukushima, S.Y. Lee, and X. Yao, editors, *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'02)*, pages 1786–1790, 2002.

[8] C.A. Coello, D.A. Van Veldhuizen, and G.B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic, New York, 2002.

[9] K. Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons,, New York, 2001.

[10] C.M. Fonseca and P.J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. *Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California*, pages 416–423, 1993.

[11] P. Hajela and C.Y. Lin. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, 4:99–107, 1992.

[12] S. Haykin. *Neural networks - a comprehensive foundation*. Printice Hall, USA, 2 edition, 1999.

[13] J. Horn, N. Nafpliotis, and D.E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. *Proceedings of the First IEEE Conference on Evolutionary Computation*, 1:82–87, 1994.

[14] H. Ishibuchi and T. Yamamoto. Evolutionary multiobjective optimization for generating an ensemble of fuzzy rule-based classifiers. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, Lecture Notes in Computer Science (LNCS), pages 1077–1088, Chicago, IL, 2003. Springer-Verlag.

[15] J. Knowles and D. Corne. The pareto archived evolution strategy: a new baseline algorithm for multiobjective optimization. *In 1999 Congress on Evolutionary Computation, Washington D.C., IEEE Service Centre*, pages 98–105, 1999.

[16] J. Knowles and D. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[17] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D.S. Touretzky, and T.K. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. MIT Press, 1995.

[18] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999.

[19] Y. Liu and X. Yao. Simultaneous training of negatively correlated neural networks in an ensemble. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29(6):716–725, 1999.

[20] Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.

[21] R. McKay and H.A. Abbass. Anti-correlation: A diversity promoting mechanisms in ensemble learning. *The Australian Journal of Intelligent Information Processing Systems (AJIIPS)*, 7(3/4):139–149, 2001.

[22] D. Michie, D.J. Spiegelhalter, and C.C. Taylor. *Machine learning, neural and statistical classification*. Ellis Horwood, 1994.

[23] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In J.L. McClelland D.E. Rumelhart and the PDP Research Group Eds, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition., Foundations, 1, 318,*. MIT Press Cambridge, 1986.

[24] Y. Sawaragi, H. Nakayama, and T. Tanino. *Theory of multiobjective optimization*, volume 176 of *Mathematics in science and engineering*. Academic Press Inc, Harcourt Brace Jovanovich, 1985.

[25] J.D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100, 1985.

[26] A.J.C. Sharkey. On combining artificial neural nets. *Connection Science*, 8:299–313, 1996.

[27] N. Srinivas and K. Dev. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[28] R. Storn and K. Price. Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley, 1995.

[29] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.