

Preliminary Experiments with Ensembles of Neurally Diverse Artificial Neural Networks for Pattern Recognition

Abdullahi Adamu¹, Tomas Maul¹, Andrzej Bargiela²,
and Christopher Roadknight²

¹ University of Nottingham - Malaysia Campus,
Malaysia

² University of Nottingham - UK Campus,
United Kingdom

Abstract. Although there have been a few approaches to achieve the goal of fault tolerance by diversifying redundancy of the individual networks that make up a neural network ensemble, some of which include ensembles of neural networks of different sizes, and ensembles of different models of neural networks such as Radial Basis Function Networks and Multilayer Perceptron, there is yet to be an empirical study on hybrid neural networks that makes use of a diverse set of transfer functions, which we would expect to be able to exhibit diverse network architectures, and thus possibly more diverse error patterns. In this paper, we present an approach that uses transfer function diversity to achieve significant results on ensembles. The results show that even with relatively small networks having 5 hidden nodes, and a relatively small ensemble size of just 10 members, the ensemble is able to get competitive results on the Iris data set. It also capable of obtaining competitive results with 20 ensemble members of relatively small networks on other popular data sets such as the Diabetes, Sonar, Hepatitis, and Australian Credit Card problems. In addition to that, it is shown that these results can be achieved with a simple sorting and selection of the Top N solutions of the population, in contrast to other methods of selecting ensemble members that can be computationally expensive, such as selection of the Pareto-front, or hill climbing methods of selection.

Keywords: Hybrid Neural Networks, Artificial Neural Networks, Transfer function Optimization, Pattern Recognition.

1 Introduction

Fault tolerance through redundancy is a concept that is used in many fields, for example in software engineering where one of the methods for achieving reliability is through redundant diverse implementations[19]. The aim is to minimize the failure of a system by combining several redundant, but diverse components designed for a single task in such a way that they are unlikely to all fail at

once. This concept has also gained interest in the field of Artificial Neural Networks as Ensembles of Artificial Neural networks, where Neural Networks have diverse designs resulting in an improved generalization ability when combined. The research findings from earlier studies [6,19,14,20,10,15,18] show that neural networks combined as ensembles exhibit an improved generalization ability provided there is diversity in the bias of its members. In other words, if there are differences in the limitations of the learners, then an ensemble of such learners should result in increased learning accuracy. In addition to these empirical findings, it is also proven [6] that the error of an ensemble is guaranteed to be the same or better than the average error of its components.

The importance of diversity in ensembles is quite intuitive. If the individual solutions in an ensemble were all identical, the performance of the ensemble would not differ from any of the members of the ensemble. However, if all the members of the ensemble were different from each other such that the decision boundaries that they project onto the input-space are varied yet accurate, then one can expect that the averaged decision boundary of the learners is likely to be significantly more accurate, or at the very least the same. The significance also depends on some other factors such as the method used to combine the outputs of the members, the number of members in the ensemble, and the accuracy of the members in the ensemble.

Generally, most of the promising methods for creating diversity in ensembles could be categorized into three according to their area of focus [6]: data set, model, and training algorithm. In the first case, some approaches found in the literature use re-sampling and pattern distortion methods to achieve some variations in the training data set. These variations then implicitly cause behavioral differences in members of the ensemble. Popular re-sampling methods include bagging and boosting. In bagging, random samples of the data set are used to train each member. Boosting is similar to bagging, however it considers the distribution of the subsets while sampling. Another method used by some studies that was highlighted by Brown et. al. [6] is to re-sample the features of each pattern in the training data set. On the other hand, distortion methods used include the addition of Gaussian noise, or non-linear transformations of the training patterns in the data set. One of the non-linear transformation approaches found [6,19] to be effective was to stimulate a randomly generated neural network with the training pattern and then use its output as the distorted pattern. Gaussian noise was also found to be helpful [6,19]. In the case of diversity creation methods focusing on models [6], some methods use a mixture of models. The most popular includes the use of homogeneous models with varied parameters [6], such as neural networks of different sizes within the ensemble, or neural networks with different types of architectures such as Multilayer Perceptrons and Radial Basis Function networks [6]. Other methods [6] use heterogeneous models such as the use of decision trees and neural network within an ensemble. In the final category are methods which focus on the training algorithm, some of which include the use of different training algorithms [19], and the introduction

of an additional term in the objective function [15,18], such as in neural network ensembles trained by evolutionary algorithms [6].

There are some areas that have been deficient with regards to experiments. One such area is the topic of model diversification approaches for neural network ensembles. Intuitively, it makes sense that if we are aiming for error diversity within our neural network ensemble, an equally likely approach to the others that could yield significant diversity is an approach that is explicit. By *explicit*, we mean an approach that takes a direct method, such as the combination of diverse architectural models of neural networks. According to our knowledge, there has yet to be experiments with ensembles using hybrid artificial neural networks implementing a diverse transfer functions set; which we would expect to increase error diversity in ensembles. This was also highlighted in the thorough survey of ensembles by Brown [6]. The only partially related work done so far was by Partridge who used pure models of Multilayer Perceptron’s (MLP) and Radial Basis Functions (RBF) in an ensemble to achieve diversity. However, even that work was suggested to be a preliminary study by [6].

In this paper, we experiment with hybrid neural network ensembles implementing a diverse transfer function set, also known as Neural Diversity Machine Ensembles (NeuDiME). This is unlike other approaches found in the literature which have used a mixture of pure models as reported by Brown et. al [6]. We explore the performance of this approach in different circumstances; specifically, we test it on popular pattern classification problems such as the Iris, Sonar, Hepatitis, Diabetes, and the Australian credit card problem, which have been used as benchmarks of choice in most literature. We also compare this approach with other approaches found in the literature, and also analyze two accurate yet different strategies evolved for some of the problems (i.e. diabetes and Iris).

The contributions of this paper are as follows: firstly, the paper presents the application of hybrid neural networks in ensembles and provides a study on some of the effects of transfer function diversity in neural network ensembles. Secondly, it shows that this neural network framework is able to develop different strategies for a problem that can be used in ensembles without explicit diversity maintenance that can be expensive, such as selection of the Pareto front for use as ensembles, or fitness sharing. It also shows that this approach can evolve a relatively smaller ensemble of compact networks that has a competitive performance. Finally, it shows how neural diversity can result in diverse classifiers by analyzing two computational strategies evolved for the diabetes problem.

It should be noted that the term “weight function” is used interchangeably with the term “input combination function” and activation function. Likewise, the term, “node function” is used interchangeably with the term “output function”. Additionally, the term “node” is used interchangeably with the term “neuron” and “unit”.

2 Neural Diversity Machines

In this section, the Neural Diversity Machines approach proposed by Maul [16] is explained, starting from transfer function diversity - the core of the framework, then followed by weight and architectural evolution.

2.1 Transfer Function Diversity

A transfer function of a neural network is composed of the input combination function and an output function; the input combination function computes the input values being transferred into a neuron by other neurons in the network connected to it using weighted connections, while the output function can be considered as a mathematical model of its biological counterpart, which determines the output value of the neuron.

Traditional and well established Artificial Neural Networks such as Radial Basis Functions (RBF) and Multilayer Perceptrons (MLP) use predetermined transfer functions for the nodes of each layer as part of their distinct architectural properties. Often, the nodes of each layer are homogeneous in their transfer function. A Radial Basis Function uses a combination of a distance-based input combination function (e.g. Euclidean distance) and a radial basis output function, which is typically a Gaussian, Multiquadratic, thin-plate-spline, or inverse Multiquadratic. In the case of a Multilayer Perceptron, its typically an inner-product input combination function accompanied with a sigmoid output function. Though it's proven that both RBF networks (RBFN) and Multilayer Perceptrons are able to approximate any function, provided that the complexity of the network's model is matched with the complexity of the problem [4,8], it's not proven that the model will be optimal or practical. By *optimal*, we are specifically emphasizing the use of a minimal number of hidden units to learn a problem.

There is no one-size-fits-all in the choice of transfer functions [9,13,16]. A certain problem might be more suited for an MLP unit, whereas another may require RBF units. Thus, there is a need for adapting the choice of transfer function.

Several studies have approached this problem using hybrid models [16,9,13] which implement different basis functions, either as a single hybrid layer, or as several pure layers and have been found to perform better than their canonical models (i.e. RBF or MLP).

This work differs because it is particularly motivated from the biological standpoint of the benefits of neural diversity found in biological neural networks, which includes increased representational capacity that contributes significantly to their efficiency [5]. This was replicated by having different classes of functions in the transfer functions pool, such as radial-basis units, projection-basis units and higher-order units. However, due to the flexibility allowed in the combination of activation and output functions during optimization, other unconventional transfer functions were also evolved.

Table 1. List of some input combination and output functions used by Neural Diversity Machines and their visualization color codes

Input Combination Functions	Color Code
Inner-Product ($ap_i = \sum_{i=0}^n w_i i_i + w_{bias}$)	Red Solid Edge
Higher-Order Product ($ap_i = \prod_{i=0}^n cw_i * i_i$)	Yellow Solid Edge
Higher-Order Subtractive ($ap_i = \sum_{i=1}^n x_0 - x_i $)	Yellow Dashed Edge
Euclidean Distance ($ap_i = \sqrt{\sum_{i=0}^n (w_i - i_i)^2}$)	Magenta Dashed Edge
Standard Deviation ($ap_i = stdDev(w_i i_i, w_{i+1} i_{i+1} \dots w_n i_n)$)	Blue Solid Edge
Min ($ap_i = min(w_i i_i, w_{i+1} i_{i+1} \dots w_n i_n)$)	Gray Dashed Edge
Max ($ap_i = max(w_i i_i, w_{i+1} i_{i+1} \dots w_n i_n)$)	Black Dashed Edge

Table 2. List of Output functions for Neural Diversity Machines and their visualization color codes

Output functions	Color Codes
Linear ($O_i = \alpha * ap_i$)	Yellow Node Outline
Hyperbolic tangent ($O_i = \frac{1 - e^{-\alpha * ap_i}}{1 + e^{-\alpha * ap_i}}$)	Cyan Node Outline
Sigmoid ($O_i = \frac{c}{1 + e^{-\alpha * ap_i}}$)	Red Node Outline
Gaussian ($O_i = e^{-\frac{(ap)^2}{width}}$)	Blue Node Outline
Gaussian II ($O_i = e^{-\frac{(ap)^2}{width}}$ if $O_i > \theta$ then $O_i = 1.0$)	Dark-Blue Node Outline

3 Neural Diversity Machine Ensembles (NeuDiME)

Neural Diversity Machine Ensembles is an ensemble made up of Neural Networks that conforms to the framework of Neural Diversity Machines [16]. It uses a set of diverse input combination functions (see input combination set in table 1) and output functions (see output function set in table 2) which enables the optimization algorithm to find near optimal transfer functions for each node from the transfer function set, thus increasing the likelihood of having accurate ensemble members. In addition to that, the diverse transfer function set should result in relatively more diverse computational strategies [6] for a problem as compared to an approach using pure models. Subsequently, this is expected to result in the desired balance between bias and variance. After all, it is intuitive that an ensemble made up of different yet accurate models will likely yield more *useful* diversity. By *useful*, we are referring to the diversity that results in significant improvements in the generalization ability of the ensemble. This differs from other approaches which have attempted the use of neural networks of different sizes within an ensemble or a mixture of pure models [6]. The stages for the optimization algorithm are as follows:

1. Solutions (of various hidden node sizes) are created.
2. Solutions are evaluated.
3. Differential Evolution is applied.
4. Other Evolutionary Operators are applied (i.e. Mutation and Cross-over).
5. Finally, ensemble members are selected from the population.

6. Ensemble is evaluated.
7. Weak solutions are eliminated.
8. Stopping condition is checked.
9. If the stopping condition is met, the ensemble is evaluated on the test set and the error is used as its generalization performance.

3.1 Ensemble Member Selection

In this study, we use the Top N solutions for convenience. It lacks the relatively higher computational cost of other selection methods such as the use of the Pareto-front [2], or selection of members using a hill climbing approach while maintaining diversity with fitness sharing [7,12]. In addition to that, it's relatively simple and has been used in later works by Opitz & Shavlik [17].

The method of selection selects the Top N solutions from the population after sorting by fitness, where N is the desired size of the ensemble. The sorting prior to selection helps in improving the chances of picking the N solutions with the most generalization ability.

However, this selection method assumes there is already diversity in the population such that the top N solutions are diverse after sorting by accuracy. In the case of NeuDiME, there is already diversity introduced by using a diverse set of transfer function combinations to pick from for all the neural networks. In addition to that, there is also the added diversity of neural networks of different sizes.

4 Experiments

In this section of the report, we present the benchmarks, performance measures, and finally the experimental setup.

4.1 Experimental Setup

The benchmarks used consisted of some problems commonly used in the ensemble literature: Iris, Sonar, Australian credit card, Hepatitis and Diabetes retrieved from the machine learning repository [3]. In order to conform to the common measure of generalization ability found in the literature [8,14,10], 10 fold cross-validation was used.

The optimization parameters for the Global Stochastic Optimization (GSO) algorithm used to optimize the neural network ensembles is given as in Table 5.

5 Results and Discussion

In this section we present the results of the neural network ensembles, NeuDiME, on the popular data sets and a comparison with various other methods that include: Memetic Pareto Artificial Neural Networks(MPANN), Cooperative Neural Network Ensemble(CNNE), Ensemble with Negative Correlation Learning (EENCL), Diverse and Accurate Ensembles(DIVACE).

Table 3. List of some of the benchmarks retrieved from the UCI Machine Learning Repository[3]

Benchmarks	Features	Classes	Examples
Iris	4	3	150
Sonar	60	2	208
Card (Australian)	51	2	690
Diabetes	8	2	768
Hepatitis	19	2	155

Table 4. Experimental setup for the data sets showing the maximum number of hidden units allowed per ensemble member, the number of members in the ensemble and the number of folds used for K-fold cross-validation

Benchmarks	Max Hidden units	Members(Ensemble)	Folds (K-fold CV)
Iris	5	10	10
Sonar	5	10	10
Diabetes	5	20	10
Hepatitis	5	20	10
Card (Australian)	5	20	12

5.1 Performance on Some Popular Benchmarks

The results produced by the Neural Diversity Machine Ensemble were competitive. It achieved the best performance in 3 out of 5 of the benchmarks when compared with the 5 other methods. However, it did not perform as well in the Australian Credit Card data set. This could be for a variety of reasons, which include the relatively higher dimensionality of the Australian Credit Card Data set (i.e input space being $\{x\}^{51}$, and sample size being 690). Typically, problems of higher dimensionality in terms of their input space require relatively complex solutions. As the dimensions of the problem increases, solutions have to account for these new dimensions by making the decision of which tuple of dimensions have more information gain over others, and form decision boundaries through these dimensions that separate the classes accurately. As one can imagine, as the dimensionality increases, this problems gets consistently harder. In the case of NeuDiME, its increased access to a wider variety of solutions can be mixed news. The advantage is that it increases its chances of finding computational strategies with creative hypotheses that describes the problem with simplicity and usually better generalization [8][13]. The disadvantage is that this can present more local minima. In the case of these results for the Australian Credit card and Sonar, the dimensionality of the benchmarks played some role in the slight performance difference; while the Card benchmark had 51 inputs, the Sonar had 60. In addition to having relatively more local minima, one can also expect that convergence might also be relatively slower as a result of the increased computational capacity.

Table 5. The optimization parameters used for the neuroevolution of the ensemble members

Optimization Parameter(s)	Value(s)
Max Iterations	100
Population size	30
Percent to eliminate	0.3
Min cost (elimination)	0.66
Min age (elimination)	3
Cross Over	True
Probability of Cross Over	0.2
Differential Evolution (DE) Iterations	3
DE alpha	0.2
Gene range	[-0.9, 0.9]
Probability of Mutation	0.2
Gaussian Mutation (Mean,Std)	(0.0, 0.2)

Table 6. Comparison of test error with other learning methods - Results of MPANN, EENCL, CNNE, DIVACE and Optiz and Shavlik as reported in their findings. In the case of Opitz et. al the best results of the five techniques experimented with were used. The Testing Error of EENCL was used as reported in [11]. As for MPANN, the best of the three results as reported in [2], while results of CNNE and DIVACE were retrieved as reported in [12,7], respectively.

	Australian Credit Card	Iris	Sonar	Diabetes	Hepatitis
MPANN	0.135	-	-	0.23	-
EENCL	0.138	-	-	0.221	-
CNNE	0.092	-	-	0.198	-
DIVACE	0.138	-	-	0.226	-
Opitz & Maclin [17]	0.137	0.039	0.129	0.233	0.178
NeuDiME	0.221	0.027	0.181	0.174	0.115

5.2 Interesting Computational Strategies Evolved for Prediction

In this subsection of the results and discussion, we present statistics of transfer function use for some of the data sets used in the experiments. These include the probabilities of combining the possible input combination functions with the possible output functions in the fittest members of the ensemble, and the associated error with each combination. In addition to that, we also reveal some of the diverse strategies used by the members of the ensemble towards solving the diabetes problem by looking at their choice of transfer function, connectivity and weights.

Diabetes. The diabetes problem showed an emphasis on strategies that relied on using standard deviation and output function such as the identity function, and hyperbolic tangent. The least error was associated with the combination of Euclidean distance and Gaussian II, which is essentially a variant of a radial

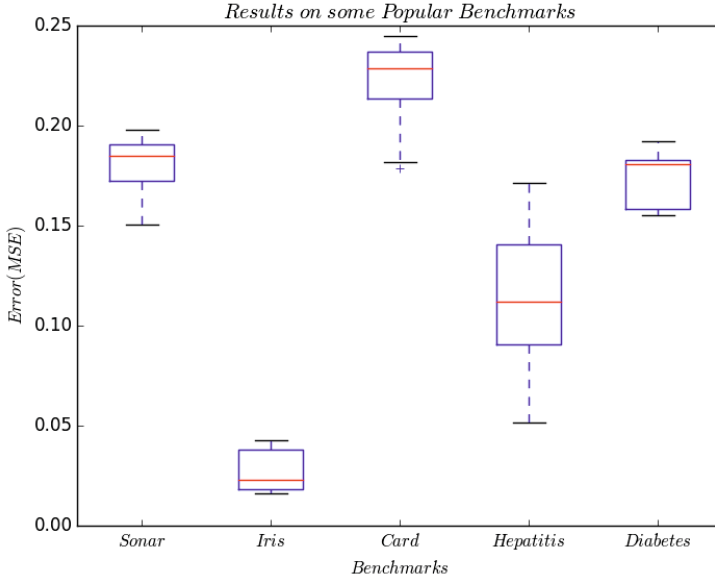


Fig. 1. Results on the popular data sets of NeuDiME - Test error averaged over 10-fold cross-validation, except in the case of Australian credit card problem which was set to 12-fold cross-validation for the sake of comparability with published results

basis function unit. Higher-order product combined with identity function also showed a relatively lower associated error. However, considering these were some of the least likely to be evolved for the problem, they should be expected to have lower errors. The tables below present the statistics.

In the following discussion, we will present a study of two diverse strategies evolved for NeuDiME on the diabetes problem. This reveals the ability of NeuDiME to exhibit diverse neural computation strategies that are accurate. One of the most accurate strategies evolved for the diabetes problem was a fully connected network consisting of four hidden units, implementing the following transfer functions found in Figure 2.

It seems that the range of the feature values (min and max) is somehow important in this strategy. This probably explains why min and max functions were transferring information to the projection unit (i.e. the perceptron output unit). In an effort to understand why this is essential for this strategy, we compared the min and max of the raw data set: most of the time, the max value corresponds to the 2nd feature of the data set, i.e. *glucose concentration reading*. While the min value usually corresponded to either the 4th or the 5th feature, i.e. *skin fold thickness* and *serum insulin reading*, respectively. Based on the connection weights of these features to the hidden layers using min and max as a relay; the 4th feature (skin fold thickness) was given more weight as compared to the rest. The 2nd feature (glucose concentration) and 5th feature (serum insulin reading) were both given a medium weight, perhaps to normalize its values with the rest

Table 7. Likelihood(%) of using combinations of input combination and output functions for the Diabetes problem - The most likely combination was standard deviation and identity, and standard deviation and hyperbolic tangent

	Identity	Sigmoid	Gaussian	Hyperbolic Tangent	Gaussian II
Inner Product	0	0	0	3.57143	3.57143
Euclidean Distance	0	0	0	0.0	3.57143
Higher Order Product	3.57143	3.57143	0	0.0	3.57143
Higher Order Subtractive	0	0	0	0.0	7.14286
Standard Deviation	17.85714	0	0	14.28572	3.57143
Min	3.57143	7.14286	0	7.14286	0.0
Max	3.57143	7.14286	3.57143	3.57143	0.0

Table 8. Associated error of using combinations of input combination and output functions for the Diabetes problem

	Identity	Sigmoid	Gaussian	Hyperbolic Tangent	Gaussian II
Inner Product	-	-	-	0.01709	0.01161
Euclidean Distance	-	-	-	-	0.00125
Higher Order Product	0.00210	0.00644	-	-	0.01319
Higher Order Subtractive	-	-	-	-	0.02434
Standard Deviation	0.03476	-	-	0.05334	0.00405
Min	0.01357	0.00539	-	0.01066	-
Max	0.00500	0.02989	0.02224	0.00543	-

of the features, as they usually have the highest values. The most important feature relayed by these relay units based on their weights to them was *age*.

Interestingly, *age* and *skin fold thickness* are actually regarded as highly correlated to diabetes. The American Diabetes Association for example regards age as one of the leading contributing factors that increases the risk of a person having type-2 diabetes[1].

In general, this strategy seems to be taking advantage of the variety of input combination function and output functions to extract important features using unusual combinations of transfer functions such as the minimum feature value, weighted variance of between features, proximity of the feature vector to the centre of the RBF unit, and maximum feature value, and finally used these features for training a simple perceptron in the output layer. In other words, its using the hidden layer for feature selection, then taking advantage of the reduced dimensionality to train a simple hyperbolic-tangent perceptron.

Another strategy evolved was a fully connected network with two hidden units where one adopted a min input combination function and a sigmoid output function, while the other adopted a standard deviation output function with a hyperbolic tangent output function. The output unit differed from the other strategy, consisting of a max input combination which has a winner-take-all effect on the hidden nodes. Its hyperbolic tangent output function has another norming effect.

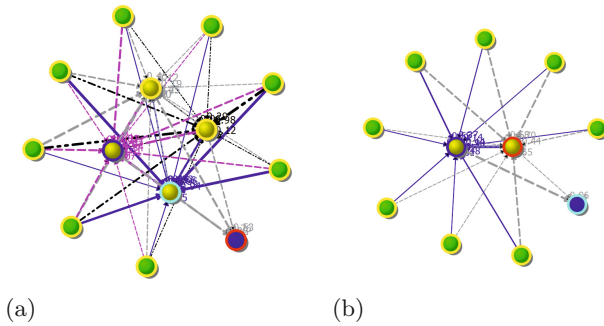


Fig. 2. Visualizations of two models that evolved different strategies for the Diabetes problem

Once again, this strategy uses the min input combination function, however in this case, it is coupled with a sigmoid output function, which has a normalizing effect on the output value - restricting it between 0.0 and 1.0. In addition to that, in this case it seems to act as a threshold for the other hidden node using the weighted variance. This is because of the choice of using max as the input combination function by the output unit.

In general, the normalizing effect of the output functions of both hidden nodes allows the hidden node using the min input combination to essentially resemble the role of a bias node that is dependent on the features.

6 Conclusion

In conclusion, this paper has presented the application of hybrid neural networks in the field of ensembles and has shown that neural diversity in an artificial neural network framework is able to exhibit different computational strategies for a problem that can be used in ensembles without the need of other explicit, and usually computationally expensive diversity maintenance. This is shown by the two different strategies for the diabetes problem. It has also been shown that this approach can evolve relatively small ensembles of compact networks that have a competitive performance. The limitations include the increased local minima, as a result of the increased access to the search space. In addition, relatively slower convergence is also a concern. However, considering the increased possibility for gain of generalization and efficiency, it is arguable that these properties are significant motivations for further research.

References

1. Lower Your Risks: Age, Race, Gender & Family History (2013), <http://www.diabetes.org/are-you-at-risk/lower-your-risk/nonmodifiabiles.html>

2. Abbass, H.: Pareto neuro-evolution: constructing ensemble of neural networks using multi-objective optimization. In: *The 2003 Congress on Evolutionary Computation, CEC 2003*, vol. 3, pp. 2074–2080. IEEE, Cancun (2003)
3. Bache, K., Lichman, M.: *UCI Machine Learning Repository* (2013), <http://archive.ics.uci.edu/ml>
4. Bishop, C.M.: *Neural Networks for Pattern Recognition*. Oxford University Press (1995)
5. Briggman, K., Kristan, W.: Multifunctional pattern-generating circuits. *Annual Review of Neuroscience* 31, 2710294 (2008)
6. Brown, G., Wyatt, J., Harris, R., Yao, X.: Diversity creation methods: a survey and categorisation. *Information Fusion* 6(1), 5–20 (2005)
7. Chandra, A., Yao, X.: Ensemble Learning Using Multi-Objective Evolutionary Algorithms. *Journal of Mathematical Modelling and Algorithms* 5(4), 417–445 (2006)
8. Gutierrez, P., Hervás, C., Carbonero, M., Fernandez, J.: Combined projection and kernel basis functions for classification in evolutionary neural networks. *Neurocomputing* 72(13–15), 2731–2742 (2009)
9. Gutiérrez, P.A., Hervás-Martínez, C.: Hybrid Artificial Neural Networks: Models, Algorithms and Data. In: Cabestany, J., Rojas, I., Joya, G. (eds.) *IWANN 2011, Part II. LNCS*, vol. 6692, pp. 177–184. Springer, Heidelberg (2011)
10. Hansen, L., Salamon, P.: Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(10), 993–1001 (1990)
11. Higuchi, T.: Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation* 4(4), 380–387 (2000)
12. Islam, M.M., Yao, X., Murase, K.: A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on Neural Networks* 14(4), 820–834 (2003)
13. Jankowski, N., Duch, W.: Optimal transfer function neural networks. In: *9th European Symposium on Artificial Neural Networks, ESANN 2001, Bruges*, vol. (I), pp. 101–106 (2001)
14. Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. In: *Advances in Neural Information Processing Systems*, pp. 231–238 (1995)
15. Liu, Y., Yao, X.: Ensemble learning via negative correlation. *Neural Networks: The Official Journal of the International Neural Network Society* 12(10), 1399–1404 (1999)
16. Maul, T.: Early experiments with neural diversity machines. *Neurocomputing* 113, 36–48 (2013)
17. Opitz, D., Shavlik, J.: Generating Accurate and Diverse Members of a Neural-Network Ensemble. In: *Advances in Neural Information Processing Systems*, vol. 8, pp. 535–541 (1996)
18. Perrone, M., Cooper, L.: When networks disagree: Ensemble methods for hybrid neural networks. In: *Neural Networks for Speech and Image Processing*, pp. 126–142 (1993)
19. Sharkey, A., Sharkey, N.: Combining diverse neural nets. *The Knowledge Engineering Review* 12(3), 231–247 (1997)
20. Wu, Z., Chen, Y.: Genetic algorithm based selective neural network ensemble. In: *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001*, vol. 1, pp. 797–802 (2001)