ENSEMBLE OF FLEXIBLE NEURAL TREES FOR BREAST CANCER DETECTION

Yuehui CHEN[1], **Ajith ABRAHAM**[2] and **Yong ZHANG**[3]

[1]*School of Information Science and Engineering,*
Jinan University, Jinan 250022, P.R. China
E-mail: yhchen@ujn.edu.cn

[2]*IITA Professorship Program,*
School of Computer Science and Engineering,
Chung-Ang University, Seoul, Republic of Korea
E-mail: ajith.abraham@ieee.org

[3]*School of Control Science and Engineering,*
Jinan University, Jinan 250022, P.R. China
E-mail: cse_zhangy@ujn.edu.cn

Abstract

Breast cancer is one of the major tumor related cause of death in women. Various artificial intelligence techniques have been used to improve the diagnoses procedures and to aid the physician's efforts. In this paper, we summarize our study to detect breast cancer using a Flexible Neural Tree (FNT) and an ensemble of FNTs. For the FNT model, a tree-structure based evolutionary algorithm and the Particle Swarm Optimization (PSO) are used to find an optimal FNT. The performance of each approach is evaluated using the breast cancer data set. Simulation results show that the obtained FNT model has a fewer number of variables with reduced number of input features and without significant reduction in the detection accuracy. The overall accuracy could be improved by using an generalized ensemble method.

Keywords: Flexible neural tree, probabilistic incremental program evolution, particle swarm optimization, ensemble learning, breast cancer detection

1 Introduction

Breast cancer is the most common cancer in women in many countries. Most breast cancers are detected as a lump/mass on the breast, or through self-examination or mammography [1]. Screening mammography is the best tool available for detecting cancerous lesions before clinical symptoms appear [7]. Surgery through a biopsy or lumpectomy have been also been the most common methods of removal. Fine needle aspiration (FNA) of breast masses is a cost-effective, non-traumatic, and mostly invasive diagnostic test that obtains information needed to evaluate malignancy. Recently, a new less invasive technique, which uses super-cooled nitrogen to freeze and shrink a non-cancerous tumor and destroy the blood vessels feeding the growth of the tumor, has been developed [2] in the USA.

Various artificial intelligence techniques have been used to improve the diagnoses procedures and to aid the physician's efforts [3], [4], [5], [6]. Fogel et al. [3] used evolutionary programming to train artificial neural networks to detect breast cancer using radiographic features and patient age. Results from 112 suspicious breast masses (63 malignant, 49 benign, biopsy proven) indicated that a significant probability of detecting malignancies can be achieved using simple neural architectures at the risk of a small percentage of false positives.

One of the most difficult problems for neural network modeling is selection of proper neural network structure. Usually single network fails to capture all the intricacy present in the data. Ensemble uses many neural network outputs to jointly solve a problem and at the same time improves the generalization ability of the network significantly. Therefore, in most of the cases, combination of neural networks improves accuracy of estimation since different networks capture different pattern of the data. Hence, classification error of a combined network (ensemble) is less as compared to individual networks. It was found that the use of the ensembles resulted in an improvement of the classifications and predictions over the individual neural network (NN) models [14], [15], [16], [17].

In this paper, we evaluate the performance of Flexible Neural Trees and an ensemble of FNTs to detect breast-cancer. For FNT model, a tree-structure based evolutionary algorithm and the Particle Swarm Optimization (PSO) are used to find an optimal FNT. Simulation studies shown the effectiveness of the proposed method. The paper is organized as follows. The PSO algorithm is given in Section 2. The FNTs and their ensemble classifiers are described in Section 3. Section 4 gives the simulation results. Finally in Section 5 we present some concluding remarks.

2 The PSO Algorithm

The PSO [8] conducts searches using a population of particles which correspond to individuals in an evolutionary algorithm (EA). A population of particles is randomly generated initially. Each particle represents a potential solution and has a position represented by a position vector $\mathbf{x_i}$. A swarm of particles moves through the problem space, with the moving velocity of each particle represented by a velocity vector $\mathbf{v_i}$. At each time step, a function f_i representing a quality measure is calculated by using $\mathbf{x_i}$ as input. Each particle keeps track of its own best position, which is associated with the best fitness it has achieved so far in a vector $\mathbf{p_i}$. Furthermore, the best position among all the particles obtained so far in the population is kept track of as $\mathbf{p_g}$. In addition to this global version, another version of PSO keeps track of the best position among all the topological neighbors of a particle.

At each time step t, by using the individual best position, $\mathbf{p_i(t)}$, and the global best position, $\mathbf{p_g(t)}$, a new velocity for particle i is updated by

$$\mathbf{v_i(t+1)} = \mathbf{v_i(t)} + c_1\phi_1(\mathbf{p_i(t)} - \mathbf{x_i(t)}) + c_2\phi_2(\mathbf{p_g(t)} - \mathbf{x_i(t)}) \quad (1)$$

where c_1 and c_2 are positive constant and ϕ_1 and ϕ_2 are uniformly distributed random number in [0,1]. The term $\mathbf{v_i}$ is limited to the range of $\pm\mathbf{v_{max}}$. If the velocity violates this limit, it is set to its proper limit. Changing velocity this way enables the particle i to search around its individual best position, $\mathbf{p_i}$, and global best position, $\mathbf{p_g}$. Based on the updated velocities, each particle changes its position according to the following equation:

$$\mathbf{x_i(t+1)} = \mathbf{x_i(t)} + \mathbf{v_i(t+1)}. \quad (2)$$

We deployed a PSO algorithm to optimize the parameter vectors of the FNT.

3 Flexible Neural Tree Classifier

In this research, a tree-structural based encoding method with specific instruction set is selected for representing a FNT model [9], [10].

3.1 Flexible Neuron Instructor and FNT Model

The function set F and terminal instruction set T used for generating a FNT model are described as follows:

$$S = F \bigcup T = \{+_2, +_3, \ldots, +_N\} \bigcup \{x_1, \ldots, x_n\}, \quad (3)$$

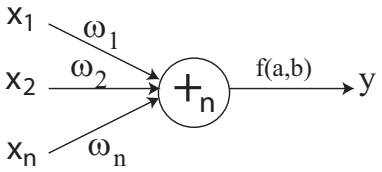

Figure 1. A flexible neuron operator

where $+_i (i = 2, 3, \ldots, N)$ denote non-leaf nodes' instructions and taking i arguments. x_1,x_2,. . .,x_n are leaf nodes' instructions and taking no other arguments. The output of a non-leaf node is calculated as a flexible neuron model (see Fig.1). From this point of view, the instruction $+_i$ is also called a flexible neuron operator with i inputs.

In the creation process of neural tree, if a nonterminal instruction, i.e., $+_i (i = 2, 3, 4, \ldots, N)$ is selected, i real values are randomly generated and used for representing the connection strength between the node $+_i$ and its children. In addition, two adjustable parameters a_i and b_i are randomly created as flexible activation function parameters. Some examples of flexible activation functions are shown in Table 1.

For developing the FNT classifier, the following flexible activation function is used.

$$f(a_i, b_i, x) = e^{-\frac{1}{2}(\frac{x-a_i}{b_i})^2} \tag{4}$$

The output of a flexible neuron $+_n$ can be calculated as follows. The total excitation of $+_n$ is

$$net_n = \sum_{j=1}^{n} w_j * x_j \tag{5}$$

where $x_j (j = 1, 2, \ldots, n)$ are the inputs to node $+_n$. The output of the node $+_n$ is then calculated by

$$out_n = f(a_n, b_n, net_n) = e^{-\frac{1}{2}(\frac{net_n - a_n}{b_n})^2}. \tag{6}$$

Table 1. The activation functions

Gaussian function	$f(x, a, b) = e^{-\frac{1}{2}(\frac{x-a}{b})^2}$
Flexible unipolar sigmoid function	$f(x, a) = \frac{2\|a\|}{1+e^{-2\|a\|x}}$
Flexible bipolar sigmoid function	$f(x, a) = \frac{1-e^{-2xa}}{a(1+e^{-2xa})}$

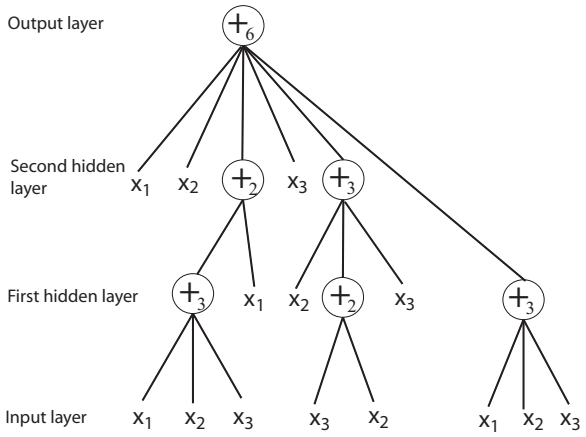

Figure 2. A typical representation of neural tree with function instruction set $F = \{+_2, +_3, +_4, +_5, +_6\}$, and terminal instruction set $T = \{x_1, x_2, x_3\}$

A typical flexible neural tree model is shown as Fig.2. The overall output of flexible neural tree can be computed from left to right by depth-first method, recursively.

3.2 Fitness function

A fitness function maps FNT to scalar, real-valued fitness values that reflect the FNT' performances on a given task. Firstly the fitness functions should be seen as error measures, i.e., MSE or $RMSE$. A secondary non-user-defined objective for which algorithm always optimizes FNTs is the size of FNT usually measured by number of nodes. Among FNTs having equal fitness values smaller FNTs are always preferred. The fitness function used for the FNT algorithm is given by the mean square error (MSE):

$$Fit(i) = \frac{1}{P}\sum_{j=1}^{P}(y_1^j - y_2^j)^2 \tag{7}$$

where P is the total number of samples, y_1^j and y_2^j are the actual required output and the FNT model output of j-th sample. $Fit(i)$ denotes the fitness value of i-th individual.

3.3 The Optimization of FNT Model

The optimization of FNT includes the tree-structure and parameter optimization. Finding an optimal or near-optimal neural tree is formulated is ac-

complished by using probabilistic incremental program evolution algorithm (PIPE) algorithm [18] and the parameters embedded in a FNT are optimized using a PSO algorithm.

Evolving an optimal or near-optimal neural tree structure. PIPE combines probability vector coding of program instructions, population-based incremental learning, and tree-coded programs. PIPE iteratively generates successive populations of functional programs according to an adaptive probability distribution, represented as a probabilistic prototype tree (PPT), over all possible programs. Each iteration uses the best program to refine the distribution. Thus, the structures of promising individuals are learned and encoded in the PPT.

The PPT stores the knowledge gained from experiences with programs (trees) and guides the evolutionary search. It holds the probability distribution over all possible programs that can be constructed from a predefined instruction set. The PPT is generally a complete n-ary tree with infinitely many nodes, where n is the maximum number of function arguments.

Each node N_j in PPT, with $j \geq 0$ contains a variable probability vector $\overrightarrow{P_j}$. Each $\overrightarrow{P_j}$ has n components, where n is the number of instructions in instruction set S. Each component $P_j(I)$ of $\overrightarrow{P_j}$ denotes the probability of choosing instruction $I \in S$ at node N_j. Each vector $\overrightarrow{P_j}$ is initialized as follows:

$$P_j(I) = \frac{P_T}{l}, \forall I : I \in T \tag{8}$$

$$P_j(I) = \frac{1 - P_T}{k}, \forall I : I \in F, \tag{9}$$

PIPE combines two forms of learning: generation-based learning (GBL) and elitist learning (EL). GBL is PIPE's main learning algorithm. The purpose of EL is to use the best program found so far as an attractor. PIPE executes as follows:

GBL

REPEAT

with probability P_{el} DO EL

otherwise DO GBL

UNTIL termination criterion is reached

Here P_{el} is a user-defined constant in [0,1].

Generation-Based Learning

Step 1. **Creation of Program Population.** A population of programs $P_{ROG\,j}$ ($0 < j \leq PS$; PS is population size) is generated using the prototype tree PPT.

Step 2. **Population Evaluation.** Each program P_{ROG_j} of the current population is evaluated on the given task and assigned a fitness value $FIT(P_{ROG_j})$ according to the predefined fitness function (Equations(5)and (6)). The best program of the current population (the one with the smallest fitness value) is denoted P_{ROG_b}. The best program found so far (elitist) is preserved in ${P_{ROG}}^{el}$.

Step 3. **Learning from Population.** Prototype tree probabilities are modified such that the probability $P(P_{ROG_b})$ of creating P_{ROG_b} increases. This procedure is called adaptting PPT towards (Progb). This is implemented as follows. First $P(P_{ROG_b})$ is computed by looking at all PPT nodes N_j used to generate P_{ROG_b}:

$$P\left(P_{ROG_b}\right) = \prod_{j:N_j \text{ used to generate } P_{ROG_b}} P_j\left(I_j(P_{ROG_b})\right) \tag{10}$$

where $I_j(P_{ROG_b})$ denotes the instruction of program P_{ROG_b} at node position j. Then a target probability P_{TARGET} for P_{ROG_b} is calculated:

$$P_{TARGET} = P(P_{ROG_b}) + (1 - P(P_{ROG_b})) \cdot lr \cdot \frac{\varepsilon + FIT({P_{ROG}}^{el})}{\varepsilon + FIT(P_{ROG_b})} \tag{11}$$

Here lr is a constant learning rate and ε a positive user-defined constant. Given P_{TARGET}, all single node probabilities $P_j(I_j(P_{ROG_b}))$ are increased iteratively:

REPEAT:

$$P_j\left(I_j(P_{ROG_b})\right) = P_j\left(I_j(P_{ROG_b})\right) + c^{lr} \cdot lr \cdot \left(1 - P_j\left(I_j(P_{ROG_b})\right)\right) \tag{12}$$

UNTIL $P\left(P_{ROG_b}\right) \geq P_{TARGET}$

where c^{lr} is a constant influencing the number of iterations. The smaller c^{lr} the higher the approximation precision of P_{TARGET} and the number of required iterations. Setting $c^{lr} = 0.1$ turned out to be a good compromise between precision and speed. And then all adapted vectors $\overrightarrow{P_j}$ are re-normalized.

Step 4. **Mutation and Crossover of Prototype Tree.** All probabilities $P_j(I)$ stored in nodes N_j that were accessed to generate program P_{ROG_b} are mutated with probability P_{M_p}:

$$P_{M_p} = \frac{P_M}{n \cdot \sqrt{|P_{ROG_b}|}} \tag{13}$$

where the user-defined parameter P_M defines the overall mutation probability, n is the number of instructions in instruction set S and $|P_{ROG_b}|$ denotes the

number of nodes in program P_{ROG_b}. Selected probability vector components are then mutated as follows:

$$P_j(I) = P_j(I) + mr \cdot \left(1 - P_j(I)\right) \tag{14}$$

where mr is the mutation rate, another user-defined parameter. Also all mutated vectors $\overrightarrow{P_j}$ are re-normalized.

Step 5. **Prototype Tree Pruning.** At the end of each generation the prototype tree is pruned. PPT subtrees attached to nodes that contain at least one probability vector component above a threshold T_P can be pruned.

Step 6. **Termination Criteria.** Repeat above procedure until a fixed number of program evaluations is reached or a satisfactory solution is found.

Elitist Learning

Elitist learning focuses search on previously discovered promising parts of the search space. The PPT is adapted towards the elitist program ${P_{ROG}}^{el}$. This is realized by replacing the P_{ROG_b} with ${P_{ROG}}^{el}$ in learning from population in Step 3. It is particularly useful with small population sizes and works efficiently in the case of noise-free problems.

In order to learn the structure and parameters of a FNT simultaneously there is a tradeoff between the structure optimization and parameter learning. In fact, if the structure of the evolved model is not appropriate, it is not useful to pay much attention to the parameter optimization. On the contrary, if the best structure has been found, further structure optimization may destroy the best structure. In this paper, a technique for balancing the structure optimization and parameter learning is proposed. If the better structure is found then do local search (PSO) for a number of steps (maximum allowed steps) or stop in case no better parameter vector is found for a significantly long time (say 100 to 2000 in our experiments). The criterion of better structure is distinguished as follows: if the fitness value of the best program is smaller than the fitness value of the elitist program, or the fitness values of two programs are equal but the nodes of the former is lower than the later, then we say that the better structure is found.

Parameter Optimization by PSO. Parameter optimization is achieved by the PSO algorithm as described in Section 2. In this stage, the architecture of FNT model is fixed, and it is the best tree developed during the end of run of the structure search. The parameters (weights and flexible activation function parameters) encoded in the best tree formulate a particle. The PSO algorithm works as follows:

(a) Initial population is generated randomly. The learning parameters c_1 and c_2 in PSO should be assigned in advance.

(b) The objective function value is calculated for each particle.

(c) Modification of search point. The current search point of each particle is changed using Eqn.(2) and Eqn.(1).

(d) If maximum number of generations is reached or no better parameter vector is found for a significantly long time (100 steps), then stop, otherwise goto step (b).

The General Learning Algorithm. The general learning procedure for designing a FNT model may be described as follows.

1) Set the initial values of parameters used in the PIPE and PSO algorithms. Set the elitist program as NULL and its fitness value as a biggest positive real number of the computer at hand. Create the initial population (flexible neural trees and their corresponding parameters);
2) Structure optimization by PIPE algorithm as described in subsection 3.1, in which the fitness function is calculated by mean square error (MSE) or root mean square error(RMSE).
3) If the better structure is found, then go to step 4), otherwise go to step 2);
4) Parameter optimization is achieved by the degraded ceiling algorithm as described in subsection 3.3. In this stage, the tree structure or architecture of flexible neural tree model is fixed, and the best tree is taken from the end of run of the PIPE search. All the parameters used in the best tree formulated a parameter vector to be optimized by local search;
5) If the maximum number of iterations of PSO algorithm is reached, or no better parameter vector is found for a significantly long time (100 steps) then go to step 6); otherwise go to step 4);
6) If satisfactory solution is found, then stop; otherwise go to step 2.

3.4 The Ensemble Classifier

For most regression and classification problems, combining the outputs of several predictors improves the performance of a single generic one [19]. Formal support to this property is provided by the so-called bias/variance dilemma [20], based on a suitable decomposition of the prediction error. According to these ideas, good ensemble members must be both accurate and diverse, which poses the problem of generating a set of predictors with reasonably good individual performances and independently distributed predictions for the test

points. Diverse individual predictors can be obtained in several ways. These include: (i) using different algorithms to learn from the data (classification and regression trees, artificial neural networks, support vector machines, etc.), (ii) changing the internal structure of a given algorithm (for instance, number of nodes/depth in trees or architecture in neural networks), and (iii) learning from different adequately-chosen subsets of the data set. The probability of success in strategy (iii), the most frequently used, is directly tied to the instability of the learning algorithm. That is, the method must be very sensitive to small changes in the structure of the data and/or in the parameters defining the learning process. Again, classical examples in this sense are classification and regression trees and artificial neural networks (ANNs). In particular, in the case of ANNs the instability comes naturally from the inherent data and training process randomness, and also from the intrinsic non-identifiability of the model. In what follows, three ensemble methods are employed for the stock index forecasting problems.

3.4.1 The Basic Ensemble Method

A simple approach for combining several classifier outputs is to simply average them together. The basic ensemble method (BEM) output is defined as:

$$f_{BEM} = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \tag{15}$$

This approach by itself can lead to improved performance [12], but doesn't take into account the fact that some FNTs may be more accurate than others. It has the advantage that it is easy to understand and implement [13] and can be shown not to increase the expected error [13].

3.4.2 The Generalized Ensemble Method

A generalization to the BEM method is to find weights for each output that minimize the positive and negative classification rates of the ensemble. The general ensemble method (GEM) is defined:

$$f_{BEM} = \frac{1}{n} \sum_{i=1}^{n} \alpha_i f_i(x) \tag{16}$$

where the $\alpha_i' s$ are chosen to minimize the positive and negative classification rates with respect to the target class index and sum to 1. In this work, the optimal weights of the ensemble classifier are optimized by using PSO algorithm.

Table 2. Comparative results of the FNTs and ensemble of FNTs classification methods for the detection of breast cancer

Cancer type	FNT1	FNT2	FNT3	FNT4	FNT5	FNT6
Benign	93.31%	91.55%	91.20%	94.71%	92.61%	93.66%
Malignant	93.31%	91.90%	91.20%	91.55%	90.85%	91.91%
Cancer type	BEM	GEM				
Benign	95.43%	97.18%				
Malignant	94.72%	97.19%				

Table 3. Important variables selected by FNT algorithm automatically for Benign detection

FNTs	Important variables
FNT1	x_{12},x_{14},x_{15},x_{16},x_{17},x_{18},x_{23},x_{24},x_{26},x_{27},x_{29}
FNT2	x_0,x_1,x_3,x_5,x_6,x_7,x_9,x_{13},x_{14},x_{18},x_{19}, x_{20},x_{23},x_{24},x_{27},x_{29}
FNT3	x_5,x_6,x_7,x_{10},x_{12},x_{13},x_{18},x_{19},x_{22},x_{25},x_{29}
FNT4	x_0,x_4,x_5,x_6,x_8,x_{11},x_{13},x_{22},x_{23},x_{24},x_{27},x_{28},x_{29}
FNT5	x_0,x_2,x_3,x_4,x_6,x_7,x_9,x_{10},x_{14},x_{15},x_{19},x_{23},x_{24},x_{29}
FNT6	x_1,x_2,x_5,x_7,x_9,x_{10},x_{11},x_{13},x_{16},x_{18},x_{20},x_{22},x_{26},x_{27}

4 Results

As a preliminary study, we made use of the Wisconsin breast cancer data set from the UCI machine-learning database repository [11]. This data set has 30 attributes (30 real valued input features) and 569 instances of which 357 are of benign and 212 are of malignant type. We randomly divided the training and test data sets. The first 285 data is used for training and the remaining 284 data is used for testing the performance of the different models.

All the models were trained and tested with the same set of data. As the data set has two different classes we performed a 2-class binary classification. The classification results for testing data set are shown in Table 2. It should be noted that the obtained FNT classifier has smaller size and reduced features and without a significant reduce in the accuracy. The important features for constructing the FNT models are shown in Table 3 for Benign detection and Table 4 for Malignant detection. In general, the ensemble of FNTs shows the best classification rate. Receiver Operating Characteristics (ROC) analysis of the FNTs, SEM and GEM models is shown in Table 5.

Table 4. Important variables selected by FNT algorithm automatically for Malignant detection

FNTs	Important variables
FNT1	x_1,x_5,x_6,x_7,x_{11},x_{12},x_{14},x_{16},x_{17},x_{20},x_{21},x_{29}
FNT2	x_2,x_3,x_6,x_9,x_{15},x_{17},x_{23},x_{26},x_{27}
FNT3	x_4,x_5,x_6,x_7,x_{10},x_{15},x_{16},x_{20},x_{21},x_{23},x_{24},x_{25},x_{26},x_{27},x_{28},x_{29}
FNT4	x_3,x_4,x_6,x_{12},x_{13},x_{17},x_{20},x_{22},x_{24},x_{26},x_{27},x_{29}
FNT5	x_7,x_{10},x_{13},x_{18},x_{23},x_{24},x_{26},x_{29}
FNT6	x_0,x_7,x_9,x_{14},x_{16},x_{19},x_{21},x_{22},x_{26},x_{27}

Table 5. Comparison of false positive rate (fp) and true positive rate (tp) for FNT, NN, WNN and ensemble classifiers

Cancer	FNT1		FNT2		FNT3		FNT4	
Type	fp(%)	tp(%)	fp(%)	tp(%)	fp(%)	tp(%)	fp(%)	tp(%)
Benign	3.84	91.71	14.56	95.03	9.71	91.72	6.80	95.58
Malignant	2.76	84.41	4.970	86.42	9.94	93.20	6.08	87.38
Cancer	FNT5		FNT6		BEM		GEM	
Type	fp(%)	tp(%)	fp(%)	tp(%)	fp(%)	tp(%)	fp(%)	tp(%)
Benign	16.51	97.80	8.74	95.03	4.85	95.58	5.83	98.90
Malignant	7.730	88.35	8.29	92.23	5.52	95.15	2.21	96.12

5 Conclusion

In this paper, we presented the flexible neural tree models and their ensemble models for the detection of breast cancer. As depicted in Table 2, the empirical results are very encouraging. The best accuracy was offered by the generalized ensemble method followed by the basic ensemble method. An important advantage of the FNT model is the ability to reduce the number of input variables as presented in Table 3 and Table 4. Because of this reason, it is easy to visualize the variance of the individual FNT classifiers. ROC analysis (Table 5) illustrates that FNT5 model has the highest false positive rate and the FNT1 model has the lowest false positive rates for detecting benign and the FNT3 model has the highest false positive rate and the GEM model has the lowest false positive rates for detecting malignant cancer. The time required to construct these models are not very much and hope these tools would assist the physician's effort to improve the currently available automated ways to diagnose breast cancer.

Acknowledgments

This research was partially supported by the Natural Science Foundation of China grant 60573065, and The Provincial Science and Technology Development Program of Shandong grant SDSP2004-0720-03.

References

[1] DeSilva C.J.S., Choong P.L. and Attikiouzel Y., 1994, *Artificial Neural networks and Breast Cancer Prognosis*, The Australian Computer Journal, Vol. 26, No. 1, pp. 78-81.

[2] The Weekend Australia, 2002, Health Section, pp. 13-14.

[3] David B.F., Eugene C. W., 1997, Edward M. B. and Vincent W. P., *A step toward computer-assisted mammography using evolutionary programming and neural networks*, Cancer Letters, Vol. 119, No. 1, pp. 93-97.

[4] Charles E. K. J., Linda M. R., Katherine A. S. and Peter H., 1997, *Construction of a Bayesian network for mammographic diagnosis of breast cancer*, Computers in Biology and Medicine, Vol. 27, No. 1, pp. 19-29.

[5] Shinsuke M., Satoru K., Naoyuki O., Tadao S., Takashi O., Masatoshi H. and Akio S., 1989, *An expert system for early detection of cancer of the breast*, Computers in Biology and Medicine, Vol. 19, No. 5, pp. 295-305.

[6] Barbara S. H. and Patricia G. M., 2001, *Breast Cancer: Hormones and Other Risk Factors*, Maturitas, Vol. 38, No. 1, pp. 103-113.

[7] Jain R. and Abraham A., 2004, *A Comparative Study of Fuzzy Classifiers on Breast Cancer Data*, Australiasian Physical And Engineering Sciences in Medicine, Australia, Vol. 27, No. 4, pp. 147-152.

[8] Kennedy J. and Eberhart R.C., 1995, *Particle Swarm Optimization*, Proc. of IEEE International Conference on Neural Networks, IV, pp. 1942-1948.

[9] Chen Y., Yang B., Dong J., 2004, *Nonlinear systems modelling via optimal design of neural trees*, International Journal of Neural ystems, Vol. 14, No. 2, pp. 125-137.

[10] Chen Y., Yang B., Dong J., Abraham A., 2005, *Time-series forcasting using flexible neural tree model*, Information Science, Vol. 174, No. 3-4, pp. 219-235.

[11] Merz J., and Murphy P.M., 1996, *UCI repository of machine learning databases*, http://www.ics.uci.edu/- learn/MLRepository.html.

[12] Perrone M.P., and Cooper L.N., 1993, *When networks disagree: ensemble methods for hybrid neural networks*, Neural Networks for Speech and Image Processing by R.J. Mammone, ed., Chapman-Hall.

[13] Bishop C.M., 1995, *Neural Networks for Pattern Recognition*, Oxford University Press, pp. 364-369.

[14] Ramasubramanian P. and Kannan A., 2004, *Quickprop Neural Network Ensemble Forecasting Framework for a Database Intrusion Prediction System*, Neural Information Processing, Vol. 5, No. 1, pp. 9-18.

[15] Zhou Z.-H., Wu J.-X., Tang W., 2002, *Ensembling Neural Networks: Many Could Be Better Than All*, Artificial Intelligence, Vol. 137, No. 1-2, pp. 239-263.

[16] Granitto P.M., Verdes P.F., Ceccatto H.A., 2005, *Neural network ensembles: evaluation of aggregation algorithms*, Artificial Intelligence, Vol. 163, No. 2, pp. 139-162.

[17] Maqsood I., Khan R., Abraham A., 2004, *An ensemble of neural networks for weather forecasting*, Neural Comput & Applic, Vol. 13, No. 1, pp. 112-122.

[18] Salustowicz R.P., Schmidhuber J., 1997, *Probabilistic Incremental Program Evolution*, Evolutionary Computation, Vol. 2, No. 5, pp. 123-141.

[19] Sharkey A.J.C. (Ed.): Combining Artificial Neural Nets, Springer, London, (1999)

[20] Geman S., Bienenstock E., Doursat R., 1992, *Neural networks and the bias/variance dilemma*, Neural Computation, Vol. 4, No. 1, pp. 1-58.