Nonlinear Feature Extraction Using a Neuro Genetic Hybrid

Yung-Keun Kwon
School of Computer Science & Engineering
Seoul National University
Sillim-dong, Gwanak-gu, Seoul, 151-744 Korea
kwon@soar.snu.ac.kr

Byung-Ro Moon
School of Computer Science & Engineering
Seoul National University
Sillim-dong, Gwanak-gu, Seoul, 151-744 Korea
moon@soar.snu.ac.kr

ABSTRACT

Feature extraction is a process that extracts salient features from observed variables. It is considered a promising alternative to overcome the problems of weight and structure optimization in artificial neural networks. There were many nonlinear feature extraction methods using neural networks but they still have the same difficulties arisen from the fixed network topology. In this paper, we propose a novel combination of genetic algorithm and feedforward neural networks for nonlinear feature extraction. The genetic algorithm evolves the feature space by utilizing characteristics of hidden neurons. It improved remarkably the performance of neural networks on a number of real world regression and classification problems.

Categories and Subject Descriptors

I.5.1 [**Computing Methodologies**]: Pattern Recognition—*Models*

General Terms

Performance

Keywords

Function approximation, neuro-genetic hybrid, feature extraction

1. INTRODUCTION

Artificial neural networks (ANNs) have emerged as one of the most powerful tool for regression and classification problems. The effectiveness of ANNs has been empirically tested in a variety of real world applications such as bankruptcy prediction [1], stock forecasting [2], handwriting recognition [3], speech recognition [4], medical diagnosis [5], and protein structure prediction [6]. The recent successful studies have established neural network as an alternative to various conventional methods and they are supported by the universal approximator theorem; a three-layer feedforward neural network can approximate any nonlinear continuous function to an arbitrary accuracy [7] [8]. However, the theorem assumes just in theory that a hidden layer of unlimited size is available. It provides the necessary mathematical tool for the viability of feedforward networks but does not specify how to determine a multilayer perceptron. Therefore, there are two basic problems on neural networks' performance. One is to specify the weights of a network that minimizes its error. The backpropagation algorithm [9], a local gradient search method, is the most widely used one. Unfortunately, it is prone to get stuck in local minima and highly depends on the initial weights. The other problem is to determine the structure of a network. Most networks have one or two fully connected hidden layers but it could as well be appropriate to use more hidden layers, partially connected hidden layers, or direct connections from input to output [10]. Structure design is crucial in the successful applications of ANNs since the structure has significant impact on a network's information processing capabilities. Too small a network may not provide good information processing power. On the other hand, a large network may have redundant connections and the implementation cost is high. Up to now, ANN structures are still determined by a human expert's experience and a tedious trial-and-error process.

Feature extraction is an attractive process to overcome the difficulties in ANNs' weight and structure optimization. Appropriate feature extraction avoids the curse of dimensionality, improves the generalization, and reduces the computational cost. Many procedures have been devised for that purpose. Most popular methods are linear statistical projection such as the principal component analysis [11] and Fisher's discriminant analysis [12]. However, they are not appropriate for complex problems with nonlinear correlation since they are linear dimension-reduction techniques. The linear limitation can be overcome by directly using neural networks. The representative examples include Kohonen's self-organizing maps [13], the nonlinear projection method [14], and nonlinear discriminant analysis based on the functionality of hidden units in feedforward network [15]. They exhibit some nice properties that are different from classical methods for problems which may not be linearly separable. However, the neural network-based approaches still have the limitation since the network is learned using a local gradient learning algorithm in a fixed architecture which may not be optimal. In other words, the feature space extracted by the

GECCO'05, June 25–29, 2005, Washington, DC, USA.

neural networks still highly depends on their structure and the learning algorithm.

In this paper, we try to extract good feature space from multilayer feedforward neural networks. It makes distinction from the traditional feature extraction using neural networks since it does not assume a topology with a fixed input space by the observed input variables. To produce diverse input space, genetic algorithm (GA) is used as a search technique. In fact, GAs have been considered to have potential to reinforce the performance of neural networks. However, the combinations of GAs and ANNs were mostly designed to optimize the networks' weights or to find a good topology. The hybrid approach of GA and ANN in this paper is different from the traditional one. It starts from the fact that the hidden nodes in feedforward neural network play a critical role in the learning as feature detectors. In other words, each hidden neuron may provide a nonlinear combination of the original input variables and it can be used as a salient feature. A novel idea in this paper is that the features generated by the hidden nodes in the network are used as candidate input features for the offspring in the genetic search.

This paper is organized as follows. In Section 2, we provide brief explanation about feature extraction and survey on the combinations of genetic algorithms and artificial neural networks. In Section 3, we describe our novel neurogenetic hybrid for nonlinear feature extraction. In Section 4, we provide our experimental results and compare them against existing ones. Finally, conclusions and discussions are given in Section 5.

2. PRELIMINARIES

2.1 Feature Extraction and Neural Networks

Feature extraction is formulated as a mapping ψ from a D-dimensional input space to an F-dimensional feature space, $\psi : \Re^D \rightarrow \Re^F$(in general, $F \leq D$). There were a number of feature extraction approaches in the pattern recognition literature [16] [17]. The mapping function ψ can be either linear or nonlinear, and can be learned through either supervised or unsupervised methods. Four categories are made from the two criterion: unsupervised linear, supervised linear, unsupervised nonlinear, and supervised nonlinear. Table 1 shows some example approaches belonging to each category. Generally speaking, supervised methods have better performance than unsupervised ones if the category information is available. Linear methods are attractive since they require less computation than nonlinear ones and analytical solution is often available. However, nonlinear methods are more powerful than linear ones because of the mapping function's flexibility, especially in the nonlinear problems. Nonlinear and supervised methods including multilayer feedforward neural networks have large potential in performance if the situation is possible.

A large number of ANNs for feature extraction have been proposed [18] [13] [14]. (For a good survey, see [17].) They can also be grouped into the above four categories. ANNs for feature extraction have been investigated with two aims. One is to link between neural networks and the classical approaches such as principal component analysis or discriminant analysis. As a result, the networks actually perform the well-known feature extraction algorithms. The other is to design a new neural network or to apply existing neural network models for feature extraction. Examples include Kohonen's self-organizing maps [13], Kraaijveld *et al.*'s nonlinear project method [14]. They showed some nice properties which are different from the classical methods. On the other hand, there were some other studies using the functionality of hidden units in feedforward network classifiers. It can be viewed as an implementation of nonlinear discriminant functions which are learned from the training data. Webb and Lowe have investigated a specific class of feedforward networks with nonlinear hidden units and linear output units [15]. They showed that the role of hidden layers is to implement a nonlinear transformation which projects input patterns from the original space to a space where patterns are easily separated by the output layer. Similarly, a nonlinear discriminant analysis network based on the multilayer feedforward network was proposed in [19]. They specified the number of input nodes to be the number of the observed variables and the number of neurons in the output layer to be the number of categories. The number of neurons in the last hidden layer is set to the dimension of the projected space. Thus, it implemented a nonlinear feature extraction from the space in the input layer to a new space in the last hidden layer.

2.2 Combinations of Genetic Algorithms and Neural Networks

Various schemes for combining genetic algorithms and neural networks were proposed or compared [20] [21] [22]. This paper focuses on how GAs can be used to assist neural networks. Combinations can be collaborative where they are used simultaneously, or supportive where they are used sequentially.

Collaborative combinations typically involve using genetic algorithms to determine the neural network weights or the network topology, or both. The backpropagation algorithm is the most widely used method to train the weights. But, it is prone to get stuck in local minima and needs gradient information. On the other hand, GA usually avoids local minima by searching in several regions simultaneously and needs no gradient information. A drawback of GA is that it is weak in fine-tuning; thus, the hybrid genetic approach that uses the backpropagation algorithm for local improvement has been popular. A straightforward genetic representation of a neural network is a simple enumeration of the weights in a string. Some variations were studied to place functional units closely together [23] [24]. Recently, two-dimensional encoding has proven to perform favorably [25] [2]. The second type of collaborative combinations is to determine the network topology. Genetic algorithms seem to be an effective approach for finding good topologies [26] [27] [28]. In a direct encoding scheme, each connection of an architecture is directly specified by its linear binary representation.

Supportive combinations typically use genetic algorithms to prepare data for neural networks. They achieved some success on real world tasks, specially classification problems. Feature selection is a representative example. A typical approach of genetic algorithms for feature selection uses binary vectors, where each bit of a vector means whether the corresponding feature is included or not [29]. It was later expanded to allow linear feature extraction where a real-value vector is used for scaling of each feature [30].

Table 1: Categories of Feature Extraction Methods

	Linear	Nonlinear
Unsupervised	Principal component analysis (PCA) Projection Pursuit Independent component analysis (ICA)	Kohonen's map Sammon's projection Nonlinear PCA network
Supervised	Linear discriminant analysis	Nonlinear discriminant analysis

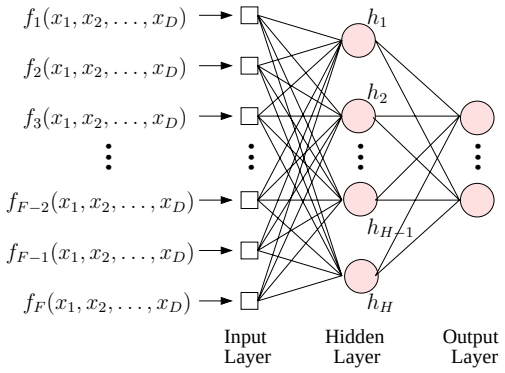


Figure 1: The ANN's Architecture Used in This Paper

2.3 Feature Space in Feedforward ANN

In this work, we propose a feature extracting genetic algorithm using multilayer feedforward ANNs having one hidden layer as shown in Figure 1. The feedforward neural network can be viewed as a composite map, $\phi \circ \psi : \Re^D \rightarrow \Re^C$ where $\psi : \Re^D \rightarrow \Re^F$ is a feature extraction map and $\phi : \Re^F \rightarrow \Re^C$ is a feed-forward map as shown in Figure 2. D-dimensional input $\mathbf{x} = (x_1, x_2, \ldots, x_D)$, the set of observed independent variables, is transformed by ψ into $\mathbf{f(x)} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_F(\mathbf{x}))$. Then, C-vectored network output $\mathbf{m} = (m_1, m_2, \ldots, m_C)$ is obtained by ψ. In general approach, $F := D$ and $f_i(\mathbf{x}) := x_i (i = 1, 2, \ldots, D)$. The network is typically built such that the mean squared errors (MSE)

$$E[\mathbf{y} - \mathbf{m}]^2$$

is minimized. The desired output $\mathbf{y}$ is one-dimensional real vector of a dependent variable in the regression problem, and it is a vector of binary values and is the j^{th} basis vector $e_j = (0, \ldots, 0, 1, 0, \ldots, 0)^t$ if $\mathbf{x}$ is in group j in the classification problem.

In the above, the neural network is trained on *input space* $S(f_1, \ldots, f_F)$ where $f_i \in \mathcal{F}_0 := \{f | f(\mathbf{x}) = x_i\}$ and generates *hidden space* $S(h_1, \ldots, h_H)$ where $h_i \in \mathcal{F}_1 = \{\varphi(f_1, f_2, \ldots, f_F; \mathbf{w})\}$. Input space transformation is one of the most important objects of feature extraction and selection because it affects the performance and learning time. In our approach, we do not use the input vector $\mathbf{x}$ directly for input nodes in the neural network. Instead, we evolve input space by utilizing the functions in hidden space.

3. FEATURE EXTRACTING GA (FGA)

3.1 Motivation

As mentioned, feature extraction is a promising process to overcome the drawbacks of feedforward neural networks with a fixed topology. It is also related to the studies about the need for multilayer perceptron. In the elementary perceptron which has no hidden neurons, it cannot classify input patterns that are not linearly separable. Figure 3 shows the elementary perceptron and its geometrical representation. In the example, m is the output of the output neuron and w_i's are weights. We assume that there are two input variables $(x_1, x_2) \in [0, 1] \times [0, 1]$ and the neuron is represented by a McCulloch-Pitts model, which uses a threshold function for its activation function. We first recognize that the use of a single neuron with two inputs results in a straight line for a decision boundary in the input space. In [31], they used a single hidden layer with two neurons to solve the XOR problem which cannot be solved by the elementary perceptron. Figure 4 shows an example solution using a hidden layer where g_1 and g_2 are the outputs of the hidden neurons. It can be explained from a different view. In the above two examples, the original input variables, x_1 and x_2, are used directly for input neurons. On the other hand, we can recognize that if g_1 and g_2 are used for input neurons the XOR problem can be solved in the elementary perceptron's topology as shown in Figure 5. This simple example shows an

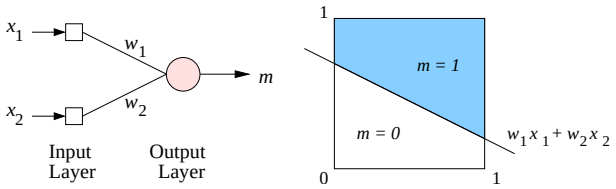


Figure 3: An Elementary Perceptron

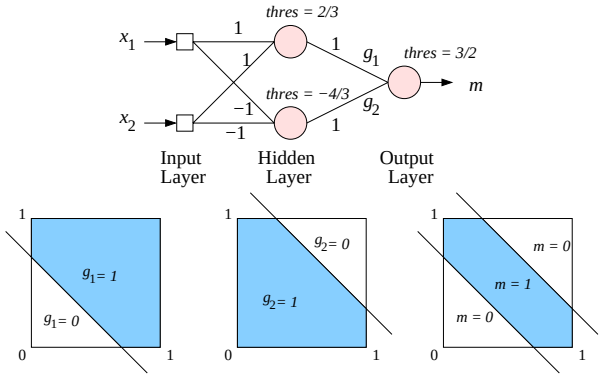


Figure 4: A Feedforward NN with a Single Hidden Layer for XOR Problem

[1]A logistic function, $\varphi[f](\mathbf{x}) = \frac{1}{1+\exp(-af(\mathbf{x}))}$ where a is the slope parameter of the sigmoid function is used in this paper.

$$m_k(\mathbf{x}) = \varphi[h_1, h_2, \ldots, h_H; \mathbf{w}](\mathbf{x}) = \varphi[\sum_{i=1}^{H} w_{k,i}^{(2)} \cdot h_i](\mathbf{x})^1 \qquad (\mathbf{w} = (w_{k,1}^{(2)}, w_{k,2}^{(2)}, \ldots, w_{k,H}^{(2)}))$$

$$h_k(\mathbf{x}) = \varphi[f_1, f_2, \ldots, f_F; \mathbf{w}](\mathbf{x}) = \varphi[\sum_{i=1}^{F} w_{k,i}^{(1)} \cdot f_i](\mathbf{x}) \qquad (\mathbf{w} = (w_{k,i}^{(1)}, w_{k,2}^{(1)}, \ldots, w_{k,F}^{(1)}))$$

where
$m_k(\mathbf{x})$: k^{th} output node's output function
$h_k(\mathbf{x})$: k^{th} hidden node's output function
$f_k(\mathbf{x})$: k^{th} input node's output function
$w_{i,j}^{(1)}$: weight between i^{th} hidden node and j^{th} input node
$w_{i,j}^{(2)}$: weight between i^{th} output node and j^{th} hidden node
φ : activation function
H : the number of hidden nodes
F : the number of input nodes

Figure 2: The Feedforward Process

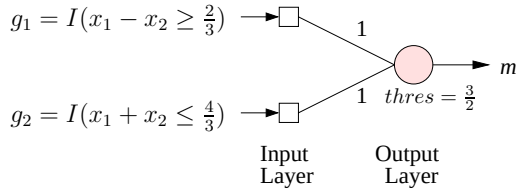


Figure 5: An Elementary Perceptron Using Feature Extraction for XOR Problem

```
create initial population of fixed size;
do {
    choose parent1 and parent2 from population;
    offspring = crossover(parent1, parent2);
    evaluation(offspring);
    replace(population, offspring);
}until(stopping condition);
report the best answer;
```

Figure 6: A Typical Steady-State Genetic Algorithm Used in FGA

effectiveness of feature extraction which can overcome the limitation of a fixed topology in ANNs.

Feature extraction GA was motivated by the hidden neurons' characteristic as *feature detectors*. If a feedforward neural network is trained, each hidden neuron in the network can be viewed as a feature extractor. FGA tries to evolve the input space using the hidden neurons. Figure 6 shows the flows of the GA we used. It is a steady-state GA. In the following, we describe each part of the GA.

3.2 FGA Frameworks

3.2.1 Problem Representation and Evaluation

A chromosome represents the input function of each input node for a feedforward neural network. It consists of F functions of the original independent variables, $f_1, f_2, \cdots, f_F$. It is evaluated using a fully-connected feedforward ANN with a single hidden layer as shown in Figure 1. The weights in the ANN are randomly initialized and updated by the backpropagation algorithms with a training dataset. The chromosome's fitness is specified with 1/MSE in a regression problem. In a classification problem, it is defined with the accuracy computed as $\frac{N_R}{N_R+N_W}$ where N_R and N_W mean the numbers of right classification and wrong classification, respectively. All the chromosomes in the population initialization have the same set of functions because f_i is set to x_i $(i = 1, 2, \cdots, F(= D))$ as the general approach described in Section 2.3. However, their fitness may be different since the weights in each ANN are randomly initialized for the evaluation .

3.2.2 Selection and Crossover

Roulette-wheel selection is used for parent selection. A crossover operator creates an offspring chromosome by choosing some functions from two parent chromosomes. Figure 7 shows the crossover process. Parents have F_1 and F_2 feature functions, $(f_1, f_2, \cdots, f_{F_1})$ and $(f_1^{'}, f_2^{'}, \cdots, f_{F_2}^{'})$, respectively. Using these feature functions, two ANNs with H hidden neurons are learned. Then, each ANN produces H nonlinear functions, $(h_1, h_2, \cdots, h_H)$ and $(h_1^{'}, h_2^{'}, \cdots, h_H^{'})$, respectively. In the example, there are totally $(F_1 + F_2 + 2H)$ candidate feature functions to be chosen for offspring. Crossover randomly chooses F_0 functions from the set of candidates (F_0 is randomly specified at every generation.) Figure 8 shows an example about the topological change in the neural networks by the crossover. In the figure, F_1, F_2 and H are set to 3 and the number of output nodes is 2. Two parents are trained by the backpropagation algorithm and produce 12 functions, f_i, h_i, $f_i^{'}$, and $h_i^{'}$ $(i = 1, 2, 3)$. If the crossover chooses f_1, h_2, and $h_3^{'}$ for the offspring, the network topology represented by the offspring is different from those of the parents. We note that the dotted connections are not updated by the backpropagation algorithm.

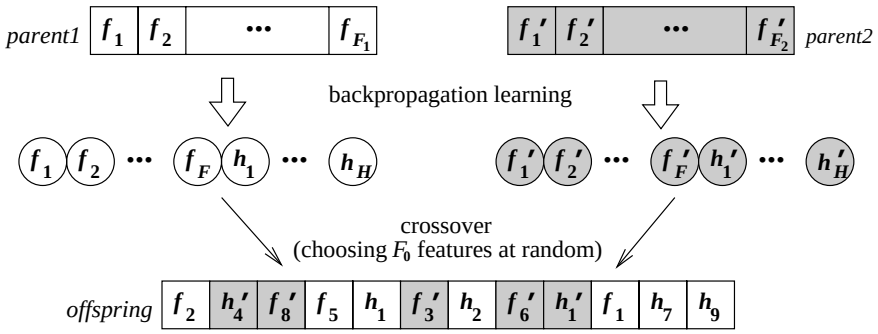


Figure 7: Crossover in FGA

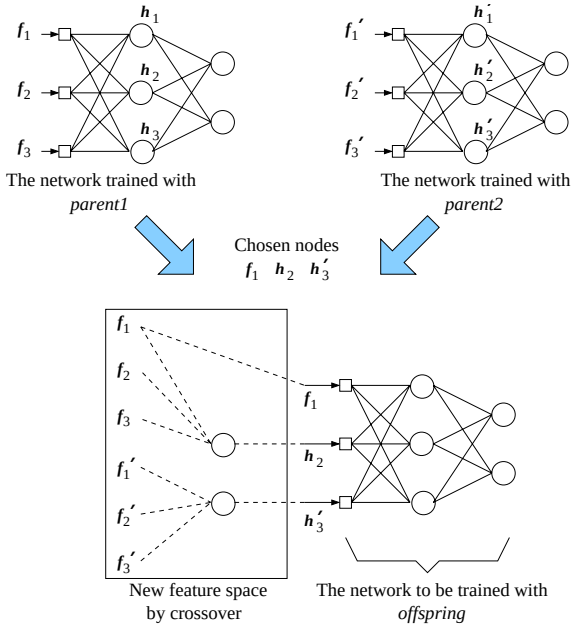


Figure 8: Topological Change by FGA's Crossover

3.2.3 *Replacement and Stopping Criterion*

The offspring first attempts to replace the inferior out of the two parents. If it fails, it attempts to replace the most inferior member of the population. It stops when there is no improvement during a given number of generations.

3.3 Feature Space in FGA

In the general approach explained in Section 2.3, the input space is fixed to $\mathcal{F}_0$. However, FGA evolves the input space by utilizing the output function of hidden nodes in ANNs with the crossover operation described in Section 3.2.2. As FGA evolves, the input space on which the ANN is trained is as follows:

$$\mathcal{F} := \bigcup_{i=0}^{n} \mathcal{F}_i$$

where $\mathcal{F}_k = \{\varphi[f_1, f_2, \ldots, f_F; \mathbf{w}] \mid f_i \in \mathcal{F}_l,\ 0 \leq l < k\}$ if $k \geq 2$ and n is a proper constant.

Table 2: Characteristics of The Datasets

Problem Type	Dataset	# of records	# of classes	# of indep. variables
Classification	BHC†	506	3	13
	BUP†	345	2	6
	SAW	500	2	2
	VEH†	846	4	18
	WIN†	178	3	13
Regression	FRD	500	real	10
	SER†	167	real	4
	BHR†	506	real	13
	OZN	330	real	8
	CHF	1607	real	8

† Available from UCI Repository (http://www.ics.uci.edu/~mlearn/MLRepository.html)

4. EXPERIMENTAL RESULTS

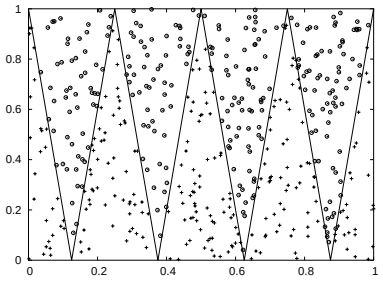
4.1 The Datasets

In the following, we briefly describe the datasets for experiment and summarize the characteristics of them in the Table 2.

Boston housing classification (BHC) This gives housing values in Boston suburbs [32]. The classes are created from the attribute median value of owner-occupied homes as follows: class = 1 if log(median value) $\leq$ 9.84, class = 2 if 9.84 $<$ log(median value) $\leq$ 10.075, class = 3 otherwise.

BUPA liver disorders (BUP) The problem is to predict whether or not a male patient has a liver disorder based on blood tests and alcohol consumption.

Artificial saw-shape data (SAW) The decision boundary is like the lines of the teeth of a saw as shown in the following figure. Samples are randomly generated in $[0, 1] \times [0, 1]$.



STATLOG Vehicle silhouette (VEH) The problem is to classify a given silhouette as one of four types of vehicles using a set feature extracted from the silhouette.

Wine recognition (WIN) The problem is to classify the type of wines from a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars.

Friedman#1(FRD) This is a synthetic benchmark dataset proposed in [33]. The formula for data generation is $y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \sum_{i=6}^{10} 0x_i + \epsilon$ where ϵ is a Gaussian random noise $N(0, 1)$, and $x_1, \ldots, x_5$ are uniformly distributed over the domain [0,1].

Servo(SER) This interesting collection of data refers to an extremely non-linear phenomenon – predicting the rise time of a servomechanism in terms of two continuous gain settings and two discrete choices of mechanical linkages.

Boston housing regression (BHR) This is the same dataset as BHC except that the dependent variable is continuous.

Ozone (OZN) This dataset was obtained from University of California at Berkeley[2]. The independent variables comprised meteorological information such as humidity and temperature. The target value is the maximum daily ozone at a location in the Los Angeles basin.

Critical Heat Flux (CHF) This dataset was obtained from Korea Atomic Energy Research Institute. It predicts the critical heat flux phenomenon which happens in operations of nuclear power plants [34].

For robust experiments, we also used a 5-fold cross-validation to estimate the accuracy. Each dataset in 5-fold cross validation is randomly divided into five disjoint subsets, $D_1, D_2, \ldots, D_5$, each containing approximately the same number of records. Each run undergoes five pairs of training and test; the k^{th} experiment was trained with $D \setminus D_k$ and tested with D_k.

4.2 Performance Comparison

We compared our approach with four other general approaches: WGA, mWNN, TGA, and mTNN. WGA and TGA are hybrid GAs combined with ANNs to optimize the ANN's weights and topology, respectively. As mentioned, they are traditional combinations of genetic algorithms and neural networks. Their representations are straight and similar to each other; if the ANN has N input nodes, H hidden nodes, and M output nodes, a chromosome in both GAs is represented by a linear array of $H \times (N + M)$ elements. However, the meaning of each element's value is different. The value in WGA means the weight of the corresponding connection; the chromosome is a real array [35] [36]. On the other hand, the value in TGA means the validity of the corresponding connection; the chromosome is a binary array [37] [38]. They also use the same GA framework as in Figure 6 except that the offspring is mutated after crossover. As crossover, 5-point crossover is used. Mutation operator replaces each weight with a low probability in WGA. On the other hand, the mutation in TGA flips each bit. Meantime, mWNN and mTNN use multi-start framework instead of GA to find optimal weights and topology; the ANNs are learned on a number of random weights and connections, respectively, and return the best result out of them. In this work, mWNN and mTNN took almost the same running time as WGA and TGA, respectively, by controlling the number of random initial points.

Table 3 shows the performance comparison result. They are the average over 20 trials. We tested them in the ANNs of various numbers of hidden nodes (H = 2, 6, 10). FGA showed the best performance on the average; it showed the best results in 32 out of 36 cases.

Figure 9 compares the transformed input space evolved by FGA with the original input space by visualizing them. In the figure, C1, C2, C3 and C4 mean the classes of the record. The two input spaces are projected to two-dimension space by Sammon's mapping. Euclidean distance was used in the projection algorithm. One can observe that the input spaces evolved by FGA are more separable.

5. CONCLUSIONS

In this paper, we proposed FGA, a neuro-genetic hybrid, for the feature extraction in the regression and classification problems. It evolves the input space in feedforward neural networks by utilizing characteristics of hidden nodes. It is a novel combination of genetic algorithm and artificial neural network; this is different from traditional neuro-genetic hybrid combinations where genetic algorithms were mainly used to optimize networks' weights and topologies. It showed notably consistent performance in regression and classification real world problems. Moreover, FGA generates considerably separable input space which may be useful in learning the ANNs.

Acknowledgments

This work was supported by grant No. (R01-2003-000-10879-0) from the Basic Research Program of the Korea Science and Engineering Foundation. This was also partly supported by the Brain Korea 21 Project. The ICT at Seoul National University provided research facilities for this study.

6. REFERENCES

[1] R. C. Lacher, P. K. Coats, S. C. Sharma, and L. F. Fant. A neural network for classifying the financial health of a firm. *European Journal of Operational Research*, 85:53–65, 1995.

[2] Y. K. Kwon and B. R. Moon. Daily stock prediction using neuro-genetic hybrids. In *Proceedings of the*

[2]ftp://ftp.stat.berkeley.edu/pub/users/breiman

Table 3: Performance Comparison
(1) Accuracy in Classification Problems

Data	H	ANN	mWNN	WGA	mTNN	TGA	FGA
BHC	2	0.728	0.731	0.729	0.718	0.713	**0.734**
	6	0.742	0.770	0.762	0.749	0.761	**0.781**
	10	0.744	**0.764**	0.757	**0.766**	0.750	0.762
BUP	2	0.703	0.701	0.688	0.697	**0.707**	**0.706**
	6	0.695	0.707	0.714	0.707	0.710	**0.726**
	10	0.698	0.715	0.710	0.700	**0.723**	0.718
SAW	2	0.743	0.742	0.740	**0.748**	**0.748**	**0.749**
	6	0.749	0.742	0.750	0.743	0.742	**0.758**
	10	0.754	0.746	0.752	0.741	0.736	**0.764**
VEH	2	0.577	0.681	0.657	0.675	0.670	**0.700**
	6	0.803	0.823	0.821	0.804	0.809	**0.830**
	10	0.803	0.824	0.829	0.820	0.820	**0.833**
WIN	2	0.868	0.933	0.884	0.831	0.876	**0.949**
	6	0.968	0.962	0.972	0.976	0.969	**0.982**
	10	0.970	0.970	0.974	0.974	**0.980**	**0.981**

(2) MSE in Regression Problems

Data	H	ANN	mWNN	WGA	mTNN	TGA	FGA
FRD	2	17.43	17.14	17.66	16.66	16.56	**14.52**
	6	17.02	16.89	15.33	16.13	15.94	**13.88**
	10	17.09	16.97	15.08	16.07	15.96	**13.90**
SER	2	0.487	0.394	0.478	0.446	0.436	**0.357**
	6	0.383	0.399	0.410	0.373	0.388	**0.324**
	10	0.384	0.381	0.452	0.369	0.386	**0.347**
BHR	2	18.19	17.76	18.18	17.01	17.36	**13.02**
	6	17.57	14.96	15.21	**14.34**	15.18	15.54
	10	16.75	15.77	14.31	15.25	**13.56**	14.80
OZN	2	23.35	19.16	19.29	18.57	**18.36**	**18.30**
	6	17.64	**17.43**	17.59	17.63	17.67	**17.46**
	10	18.03	17.68	**17.49**	**17.47**	17.76	**17.48**
CHF ($\times 10^{-2}$)	2	0.421	0.395	0.384	0.390	0.392	**0.307**
	6	0.351	0.325	0.311	0.322	0.319	**0.306**
	10	0.330	0.319	0.310	0.314	0.314	**0.305**

Genetic and Evolutionary Computation Conference, pages 2203–2214, 2003.

[3] S. Knerr, L. Personnaz, and G. Dreyfus. Handwritten digit recognition by neural networks with single-layer training. *IEEE Transactions on Neural Networks*, 3:962–968, 1992.

[4] H. Bourlard and N. Morgan. Continuous speech recognition by connectionist statistical methods. *IEEE Transactions on Neural Networks*, 4:893–909, 1992.

[5] H. B. Burke. Artificial neural networks for cancer research: Outcome prediction. *Seminars in Surgical Oncology*, 10:73–79, 1994.

[6] N. Qian and T. J. Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *Journal of Molecular Biology*, 202:865–884, 1988.

[7] M. Brown and C. Harris. *Neural Fuzzy Adaptive Modeling and Control*. Prentice-Hall, 1994.

[8] G. Cybendo. Approximations by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[10] D. Whitley, T. Starkweather, and C. Bogart. Genetic algorithms and neural networks: optimizing connections and connectivity. *Parallel Computing*, 14:347–361, 1990.

[11] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1996.

[12] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[13] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[14] M. A. Kraaijveld, J. Mao, and A. K. Jain. A nonlinear projection method based on kohonen's topology preserving maps. *IEEE Transactions on Neural Networks*, 6:548–559, 1995.

[15] A. R. Webb and D. Lowe. The optimized internal representation of multilayer classifier networks performs nonlinear discriminant analysis. *Neural Networks*, 3:367–375, 1990.

[16] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs, 1988.

[17] J. Mao and A. K. Jain. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks*, 6:296–317, 1995.

[18] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58, 1989.

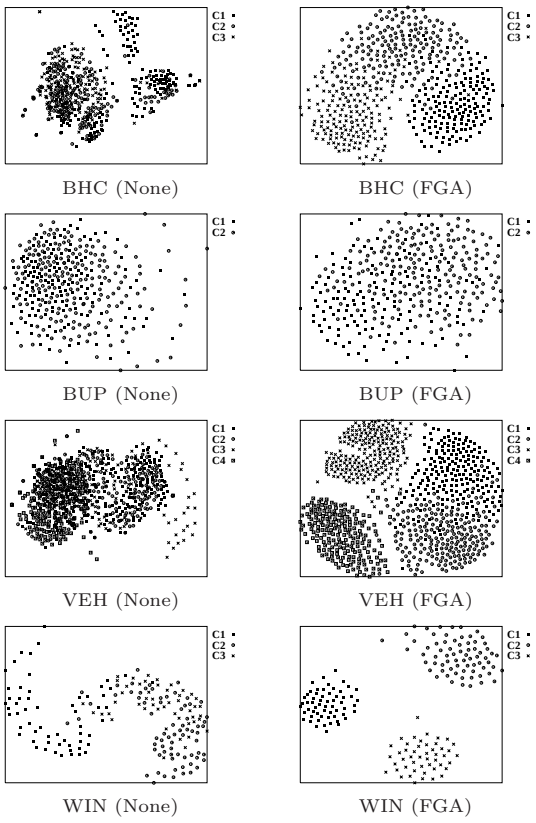


Figure 9: Visualization of Input Spaces

[19] J. Mao and A. K. Jain. Discriminant analysis neural networks. In *IEEE International Conference on Neural Networks*, pages 300–305, 1993.

[20] J. Branke. *Evolutionary Algorithms for Neural Network Design and Training*. Technical Report No.322, University of Karlsruhe, Institute AIFB., 1995.

[21] D. Schaffer, D. Whitley, and L. Eshelman. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In *Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pages 1–37, 1992.

[22] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87:1423–1447, 1999.

[23] D. Thierens, J. Suykens, J. Vandewalle, and B. De Moor. Genetic weight optimization of a feedforward neural network controller. In *Proceedings of the Conference on Artificial Neural Nets and Genetic Algorithms*, pages 658–663, 1993.

[24] B. Yoon, D. J. Holmes, G. Langholz, and A. Kandel. Efficient genetic algorithms for training layered feedforward neural networks. *Information Science*, 76:67–85, 1994.

[25] J. H. Kim and B. R. Moon. Neuron reordering for better neuro-genetic hybrids. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 407–414, 2002.

[26] H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:461–476, 1990.

[27] G. F. Miller, P. M. Todd, and S. U. Hedge. Designing neural networks using genetic algorithms. In *International Conference on Genetic Algorithm*, pages 379–384, 1989.

[28] S. A. Harp, T. Samad, and A. Guha. Towards the genetic synthesis of neural networks. In *International Conference on Genetic Algorithm*, pages 360–369, 1989.

[29] W. Siedlecki and J. Sklansky. A note on genetic algorithms for large-scale feature selection. 10:335–347, 1989.

[30] W. F. Punch, E. D. Goodman, M. Pei, L. Chia-Shun, P. Hovland, and R. Enbody. Further research on feature selection and classification using genetic algorithms. In *International Conference on Genetic Algorithm*, pages 557–564, 1993.

[31] D. S. Touretzky and D. A. Pomerleau. What's hidden in the hidden layers? *Byte*, 14:227–233, 1989.

[32] D. Harrison and D. L. Rubinfeld. Hedonic prices and the demand for clean air. *JEEM*, 5:81–102, 1978.

[33] J. H. Friedman. Multivariate adaptive regression splines with discussion. *Annals of Statistics*, 19:1–141, 1991.

[34] Y. K. Kwon and B. R. Moon. A genetic hybrid for critical heat flux function approximation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1119–1125, 2002.

[35] V. W. Porto, D. B. Fogel, and L. J. Fogel. Alternative neural network training methods. *IEEE Expert*, 10:16–22, 1995.

[36] G. W. Greenwood. Training partially recurrent neural networks using evolutionary strategies. *IEEE Transactions on Speech Audio Processing*, 5:192–194, 1997.

[37] S. W. Wilson. Perceptron redux: Emergence of structure. *Physics D*, 42:249–256, 1990.

[38] S. Oliker, M. Furst, and O. Maimon. A distributed genetic algorithms for neural network design and training. *Complex Systems*, 6(5):459–477, 1992.