Brief Papers

Two-Phase Construction of Multilayer Perceptrons Using Information Theory

Hong-Jie Xing and Bao-Gang Hu

***Abstract*—This brief presents a two-phase construction approach for pruning both input and hidden units of multilayer perceptrons (MLPs) based on mutual information (MI). First, all features of input vectors are ranked according to their relevance to target outputs through a forward strategy. The salient input units of an MLP are thus determined according to the order of the ranking result and by considering their contributions to the network's performance. Then, the irrelevant features of input vectors can be identified and eliminated. Second, the redundant hidden units are removed from the trained MLP one after another according to a novel relevance measure. Compared with its related work, the proposed strategy exhibits better performance. Moreover, experimental results show that the proposed method is comparable or even superior to support vector machine (SVM) and support vector regression (SVR). Finally, the advantages of the MI-based method are investigated in comparison with the sensitivity analysis (SA)-based method.**

***Index Terms*—Hidden units pruning, input units selection, multilayer perceptron (MLP), mutual information (MI) criterion.**

I. Introduction

On the issue of architecture selection in the field of neural network, Engelbrecht [1] provided a comprehensive review, while Zeng and Yeung [2] classified the approaches in the literature into two main categories, i.e., constructive and pruning approaches. Among the latter, the sensitivity analysis (SA) based method [1] plays an important role. Engelbrecht combined both the input units pruning and the hidden units pruning of multilayer perceptrons (MLPs) in a single formula and achieved satisfying results [1].

Because the selection of input units corresponds to the task of feature selection, the method for feature selection can be applied to input units selection directly. Among the commonly used feature selection algorithms, Oh *et al.* declared [3] that the sequential floating forward search (SFFS) algorithm is the best among the sequential search algorithms without considering time complexity, while their proposed hybrid genetic algorithm (HGA) could outperform the SFFS algorithm. Nevertheless, Van Dijck and Van Hulle suggested [4] that the sequential forward search (SFS) algorithm was encouraged to be utilized for the following two reasons.

- The SFS algorithm has the lowest time complexity of $O(d^2)$, where d is the number of features. However, the SFFS has time complexity in the order of $O(2^d)$, which nearly equals to that of the exhaustive search method, i.e., 2^d. In addition, the genetic algorithms need more time than the SFS algorithm for convergence [3].
- It is found from literature [3] that the performances of SFS on the standard data sets are often only a few percent smaller than those of the SFFS and HGA algorithms.

Therefore, in the first phase of the proposed method, to construct the input layer of an MLP effectively, the relevances of the feature subsets are ranked according to their dependencies to their target outputs, while the ranking process strictly obeys the SFS algorithm. Then, the relevant input units are determined according to the order of the ranking result until the best generalization ability of the MLP with a certain number of input units is obtained. In the above procedure, to determine the dependency of a certain feature subset of a data set to its corresponding target outputs, mutual information (MI) is utilized in this brief.

Once the input units with the optimal number have been determined, the MLP with sufficient number of hidden units can be trained. Then, in the second phase of the present method, the redundant hidden units in the MLP can be identified by a novel relevance measure, which is based on the MI and the connection weights between the hidden layer and the output layer (for a three-layer MLP with one hidden layer). Therefore, pruning the redundant hidden units away from the MLP, we can finally get a compact network.

II. Mutual Information

A. Definitions

According to Shannon's information theory [5], the uncertainty of a discrete random variable C can be measured by the entropy, which is defined in terms of probability as

$$H(C) = -\sum_{c} P(c)\log(P(c)) \tag{1}$$

where the base of the logarithm in (1) that is taken is 2. After giving a continuous random variable X, the remaining uncertainty of C when knowing X is now the conditional entropy

$$H(C|X) = -\int_{\mathbf{x}} p(\mathbf{x}) \left(\sum_{c} p(c|\mathbf{x}) \log(p(c|\mathbf{x})) \right) d\mathbf{x}. \tag{2}$$

By definition, the amount by which the uncertainty is decreased is the MI between the two variables C and X

$$I(C;X) = H(C) - H(C|X). \tag{3}$$

For two continuous random variables X and Y, the MI is written as

$$I(Y;X) = \int_{\mathbf{x}} \int_{\mathbf{y}} p(\mathbf{x},\mathbf{y}) \log \frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{x})p(\mathbf{y})} d\mathbf{x}d\mathbf{y}. \tag{4}$$

B. Estimation of Mutual Information

1) For Discrete Case: To estimate the MI in (3), the Parzen-window-based approach [6] is utilized. Given N input vectors $\{\mathbf{x}^{(p)}\}_{p=1}^{N}$ in N_c classes, the conditional probability density function $p(\mathbf{x}|c)$ of class $c \in \{1, 2, \ldots, N_c\}$ can be estimated by Parzen window estimation as

$$\hat{p}(\mathbf{x}|c) = \frac{1}{J_c} \sum_{i \in I_c} G(\mathbf{x} - \mathbf{x}^{(i)}, \Sigma_{\mathbf{x}_c}) \tag{5}$$

Manuscript received November 28, 2007; revised April 07, 2008; accepted July 08, 2008. First published February 27, 2009; current version published April 03, 2009. This work was supported in part by the Natural Science Foundation of China under Grants 60275025 and 60121302, by the China Postdoctoral Science Foundation under Grant 20080440820, and by the Foundation of Hebei University under Grant 2008123.

H.-J. Xing is with the College of Mathematics and Computer Science, Hebei University, Baoding 071002, China (e-mail: hjxing@hbu.edu.cn).

B.-G. Hu is with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Science, Beijing 1000910, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2008.2005604

where $\mathbf{x}^{(i)} \in \mathcal{R}^d$, J_c is the number of occurrences of c, I_c is the set of input vectors belonging to class c, $\Sigma_{\mathbf{x}_c}$ is the covariance matrix of the input vectors in class c, and the Gaussian window function is defined as

$$G(\mathbf{x}-\mathbf{x}^{(i)}, \Sigma_{\mathbf{x}_c}) = \frac{\exp\left\{-\dfrac{(\mathbf{x}-\mathbf{x}^{(i)})^T \Sigma_{\mathbf{x}_c}^{-1}(\mathbf{x}-\mathbf{x}^{(i)})}{2h^2}\right\}}{(2\pi)^{d/2} h^d |\Sigma_{\mathbf{x}_c}|^{1/2}} \quad (6)$$

with $h = (4/(d+2))^{1/(d+4)} N^{-1/(d+4)}$.

The probability of C in (3) taken as c can be estimated by

$$\hat{P}(c) = \Pr\{C = c\} = \frac{J_c}{N}. \quad (7)$$

Thus, the first term in the right-hand side of (3) can be computed by combining (1) and (7).

Similar to [6], suppose that $\hat{p}(\mathbf{x}^{(j)}) = 1/N$ with $j = 1, 2, \ldots, N$; the second term in the right-hand side of (3) can be estimated by

$$\hat{H}(C|X) = -\sum_{j=1}^{N} \frac{1}{N} \sum_{c=1}^{N_{\mathcal{C}}} \hat{p}(c|\mathbf{x}^{(j)}) \log \hat{p}(c|\mathbf{x}^{(j)}) \quad (8)$$

with

$$\hat{p}(c|\mathbf{x}^{(j)}) = \frac{\sum\limits_{i \in I_c} G(\mathbf{x}^{(j)} - \mathbf{x}^{(i)}, \Sigma_{\mathbf{x}_c})}{\sum\limits_{l=1}^{N_{\mathcal{C}}} \sum\limits_{k \in I_l} G(\mathbf{x}^{(j)} - \mathbf{x}^{(k)}, \Sigma_{\mathbf{x}_l})}. \quad (9)$$

2) For Continuous Case: The kth-nearest neighbor statistics-based approach [7] is used to estimate the MI in (4). Given N input–output pairs, $\mathbf{z}^{(p)} = [(\mathbf{x}^{(p)})^T, (\mathbf{y}^{(p)})^T]^T$ with index $p = 1, 2, \ldots, N$, where $\mathbf{x}^{(p)} \in \mathcal{R}^d$ and $\mathbf{y}^{(p)} \in \mathcal{R}^m$.

Let $\mathbf{z}^{(k(p))} = [(\mathbf{x}^{(k(p))})^T, (\mathbf{y}^{(k(p))})^T]^T$ denote the k-nearest neighbor of $\mathbf{z}^{(p)}$, while $\epsilon^p = \|\mathbf{z}^{(p)} - \mathbf{z}^{(k(p))}\| = \max\limits_{i=1,\ldots,d+m} |z_i^{(p)} - z_i^{(k(p))}|$, $\epsilon_{\mathbf{x}}^p = \|\mathbf{x}^{(p)} - \mathbf{x}^{(k(p))}\| = \max\limits_{s=1,\ldots,d} |x_s^{(p)} - x_s^{(k(p))}|$, and $\epsilon_{\mathbf{y}}^p = \|\mathbf{y}^{(p)} - \mathbf{y}^{(k(p))}\| = \max\limits_{t=1,\ldots,m} |y_t^{(p)} - y_t^{(k(p))}|$. Then, we denote $N_{x_s}^{(p)}$ as the number of points whose distance from $x_s^{(p)}$ is no more than ϵ^p, while $N_{y_t}^{(p)}$ as the number of points whose distance from $y_t^{(p)}$ is no more than ϵ^p.

Thus, the estimation of MI can be deduced from [8] as

$$\hat{I}(Y;X) = \psi(k) - \frac{d+m-1}{k} + (d+m-1)\psi(N) - \frac{1}{N}\sum_{p=1}^{N}\left[\sum_{s=1}^{d}\psi(N_{x_s}^{(p)}) + \sum_{t=1}^{m}\psi(N_{y_t}^{(p)})\right] \quad (10)$$

where the digamma function $\psi(\cdot)$ satisfies the recursion $\psi(\tau+1) = \psi(\tau) + 1/\tau$ with $\psi(1) \approx -0.5772156$.

III. Construction Strategy

This section expands on the proposed two-phase construction strategy for MLPs.

A. Phase I: Input Units Selection

Given a data set $\mathbf{D} = \{(\mathbf{x}^{(p)}, \mathbf{t}^{(p)})\}_{p=1}^{N}$, the relevance of target vector $\mathbf{t}$ with respect to the feature subsets of input vector $\mathbf{x}$ can be determined by (3) or (10) with respect to the different learning types. That is to say, for a classification task, the relevance can be obtained from (3), while from (10), for a function approximation problem. In the following, they are not specified without confusion. Hence, let $\mathbf{x}_s$ be the subset of input vector $\mathbf{x}$, which means that $\mathbf{x}_{\mathcal{S}} \subseteq \mathbf{x}$ with $\mathbf{x} \in \mathcal{R}^d$ and

$$\mathbf{x}_{\mathcal{S}} \in \{\{x_1\}, \ldots, \{x_d\}, \{x_1, x_2\}, \ldots, \{x_{d-1}, x_d\}, \ldots, \{x_1, \ldots, x_d\}\}.$$

Then, the MI of a target output vector $\mathbf{t}$ with respect to $\mathbf{x}_s$ can be estimated from (3) or (10).

Before selecting input units, the feature subsets of input vector $\mathbf{x}$ need to be ranked. For the subsets with one element, we need to sort them according to their values of MI with respect to the target vector $\mathbf{t}$ and choose the one with the maximum value. Without loss of generality, let the selected one-element subset be $\{x_{i_1}\}$. Then, fixing element x_{i_1}, we can choose one two-element subset possessing maximum value of MI with respect to $\mathbf{t}$, and let it be $\{x_{i_1}, x_{i_2}\}$. Similarly, we can get d subsets, and finally obtain a d-element set as $\{x_{i_1}, x_{i_2}, \ldots, x_{i_d}\}$, where indices $i_j \in \{1, 2, \ldots, d\}$ with $j = 1, 2, \ldots, d$. Hence, the features of input vector $\mathbf{x}$ can be ranked as order $i_1 \succ i_2 \succ \cdots \succ i_d$.

Once the ranking of the elements of feature vectors is obtained, the MLP with one input unit fed by the i_1th feature of the input vectors $\{\mathbf{x}^{(p)}\}_{p=1}^{N_{\text{train}}}$ can be trained, where N_{train} is the number of training samples. Then, a new input unit fed by the i_2th feature of the input vectors $\{\mathbf{x}^{(p)}\}_{p=1}^{N_{\text{train}}}$ is added into the MLP, and the new MLP is retrained. Similarly, we can get d MLPs by adding input units in turn according to the above order. The optimal one with the best generalization ability can be finally chosen from these d MLPs.

The whole procedure of input units selection is summarized in Algorithm 1.

Algorithm 1: The input units selection algorithm

Input: Feature vector matrix $\mathbf{X} = (x_{p,i})_{N \times d}$, target vector matrix $\mathbf{T} = (t_{p,t})_{N \times m}$

Output: The optimal MLP with best performance

Initialization: Selected feature vector matrix $\mathbf{X}_S = \phi$, ordinal feature index set $S = \phi$, remain feature index set $F = \{1, 2, \ldots, d\}$,

repeat

  for $\text{epoch} = 1, 2, \ldots, d$ **do**

    $h^* = \arg\max\limits_{h \in F} \hat{I}(\mathbf{T}; \mathbf{X}_s, (x_h^{(1)}, x_h^{(2)}, \ldots, x_h^{(N)})^T)$,

    $\mathbf{X}_S = [\mathbf{X}_S, (x_{h^*}^{(1)}, x_{h^*}^{(2)}, \ldots, x_{h^*}^{(N)})^T]$,

    $F = F \setminus \{h^*\}, S = S \cup \{h^*\}$

  end for

until $F = \phi$

for $i = 1, 2, \ldots, d$ **do**

  Train the MLPs with i input units, which are fed by the training data with the indices of their features strictly corresponding to the first i elements of index set S.

end for

B. Phase II: Hidden Units Pruning

In this brief, we focus on three-layer perceptrons with one hidden layer. However, this approach can be easily extended to MLPs with

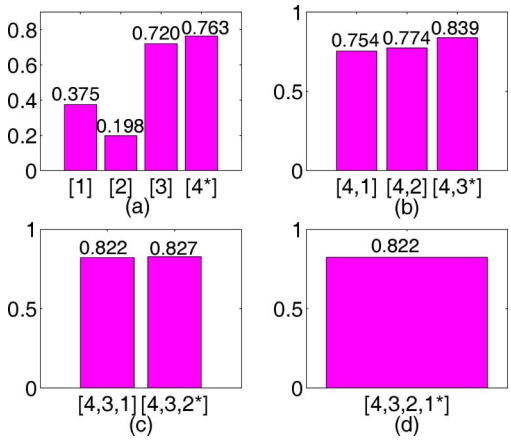

Fig. 1. Values of mutual information of target output with respect to the different feature subsets of the *Iris* data set. (a) The result of the first epoch in Algorithm 1. (b) The result of the second epoch. (c) The result of the third epoch. (d) The result of the final epoch.

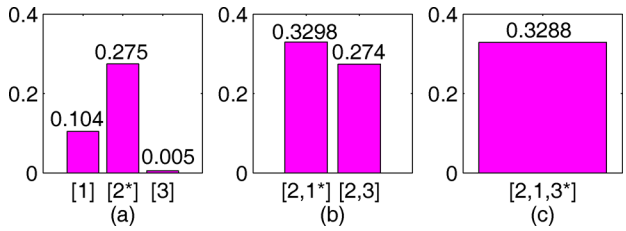

Fig. 2. Values of mutual information of target output with respect to the different feature subsets of the synthetic data set. (a) The result of the first epoch. (b) The result of the second epoch. (c) The result of the final epoch.

more than one hidden layer. The multiple-input–multiple-output parametric model for a three-layer perceptron is as follows:

$$o_k(\mathbf{x}) = \sum_{j=1}^{J} \frac{w_{kj}}{1+\exp\left\{-\sum_{i=1}^{d'} \mu_{ji}x_i - \mu_{j0}\right\}} + w_{k0} \qquad (11)$$

where w_{kj} $(k = 1,2,\ldots,m, j = 1,2,\ldots,J)$ and μ_{ji} $(i = 1,2,\ldots,d')$ are the connection weights. Moreover, $\mathbf{x} \in \mathcal{R}^{d'}$ and $\mathbf{o} \in \mathcal{R}^m$ are the input and output vectors of the MLP, and w_{k0} together with μ_{j0} are both the bias terms.

Given N_{train} training samples, we denote their input vectors as $\{\mathbf{x}^{(p)}\}_{p=1}^{N_{\text{train}}}$ with $\mathbf{x}^{(p)} = (x_1^{(p)}, x_2^{(p)}, \ldots, x_{d'}^{(p)})^T \in \mathcal{R}^{d'}$, and their output vectors as $\{\mathbf{t}^{(p)}\}_{p=1}^{N_{\text{train}}}$ with $\mathbf{t}^{(p)} = (t_1^{(p)}, t_2^{(p)}, \ldots, t_m^{(p)})^T \in \mathcal{R}^m$. Once the training of the three-layer MLP completes, the output of the jth hidden unit for input vector $\mathbf{x}^{(p)}$ can be obtained as

$$y_j^{(p)} = \frac{1}{1+\exp\left\{-\sum_{i=1}^{d'} \mu_{ji}x_i^{(p)} - \mu_{j0}\right\}}. \qquad (12)$$

Thus, the output vector of the hidden layer for $\mathbf{x}^{(p)}$ is given by $\mathbf{y}^{(p)} = (y_1^{(p)}, y_2^{(p)}, \ldots, y_J^{(p)})^T$. Moreover, the output of the kth unit in the output layer on $\mathbf{x}^{(p)}$ can be obtained as

$$o_k^{(p)} = \sum_{j=1}^{J} w_{kj}y_j^{(p)} + w_{k0}. \qquad (13)$$

Then, the output vector of the output layer for $\mathbf{x}^{(p)}$ is denoted by $\mathbf{o}^{(p)} = (o_1^{(p)}, o_2^{(p)}, \ldots, o_m^{(p)})^T$.

TABLE I
RESULTS OF THE FOUR MLPS ON THE IRIS DATA SET (IN PERCENT)

Architecture of network	Acc_{train}	Acc_{test}
1-10-3	97.33%	94.77%
2-10-3	96.00%	96.00%
3-10-3	97.33%	97.33%
4-10-3	98.67%	96.00%

Note: Acc_{train}–Accuracy rate on training set; Acc_{test}–Accuracy rate on test set.

TABLE II
RESULTS OF THE THREE MLPS ON THE SYNTHETIC DATA SET (IN PERCENT)

Architecture of network	Acc_{train}	Acc_{test}
1-15-2	83.06%	86.51%
2-15-2	88.71%	87.30%
3-15-2	94.35%	86.51%

TABLE III
RESULTS OF THE THREE MLPS ON THE IRIS DATA SET (IN PERCENT)

Model	Architecture of network	Acc_{train}	Acc_{test}
Original network	4-10-3	98.67%	96.00%
Directly trained network	3-2-3	97.33%	97.33%
Pruned network	3-2-3	97.33%	98.67%

TABLE IV
RESULTS OF THE THREE MLPS ON THE SYNTHETIC DATA SET (IN PERCENT)

Model	Architecture of network	Acc_{train}	Acc_{test}
Original network	3-15-2	94.35%	86.51%
Directly trained network	2-9-2	88.71%	87.30%
Pruned network	2-9-2	90.32%	88.10%

Without loss of generality, for the jth hidden unit, its outputs for the N_{train} input vectors can be expressed as $\mathbf{y}_j = (y_j^{(1)}, y_j^{(2)}, \ldots, y_j^{(N_{\text{train}})})^T$. Let the output matrix for the N_{train} input vectors be $\mathbf{O} = (o_{p,t})_{N_{\text{train}} \times m}$. Because the elements of $\mathbf{y}_j$ and $\mathbf{O}$ are all continuous, the MI of $\mathbf{O}$ with respect to $\mathbf{y}_j$ can be estimated by (10). However, it may be inaccurate to directly use $\hat{I}(\mathbf{O};\mathbf{y}_j)$ as the relevance of $\mathbf{y}_j$ for the following reason.

- Besides $I(\mathbf{O};\mathbf{y}_j)$ $(j = 1,2,\ldots,J)$, the connection weights w_{kj} $(k = 1,2,\ldots,m)$ are also possessing considerable contributions to estimate the relevance of $\mathbf{y}_j$ $(j = 1,2,\ldots,J)$. For the jth hidden unit, if $I(\mathbf{O},\mathbf{y}_j)$ is large while w_{kj} $(k = 1,2,\ldots,m)$ are extremely small, the real contribution of the output $\mathbf{y}_j$ to the overall network output will be also small. Therefore, if only $I(\mathbf{O},\mathbf{y}_j)$ is taken as the relevance of $\mathbf{y}_j$, it may not reflect the real contribution of it to the overall network output. On the contrary, if the MI is small while the weights are very large, the relevance of $\mathbf{y}_j$ determined only by the MI may also not reflect the real contribution because the relevance of it may be amplified by the large weights.

Hence, the relevance of the jth unit in the hidden layer of the MLP can be defined as

$$r_j = \hat{I}(\mathbf{O};\mathbf{y}_j) \times \sum_{k=1}^{m} |w_{kj}|. \qquad (14)$$

The whole process of redundant hidden units pruning of MLPs can be summarized in three steps as follows.

Step 1) Inherit the MLP constructed by the procedure of input units selection.

Step 2) Compute the values of relevance for all the hidden units according to (14) and prune the hidden unit with the least value of relevance. If the number of hidden units becomes

TABLE V
AVERAGE TESTING ACCURACY RATES (IN PERCENT) AND THEIR STANDARD DEVIATIONS (IN PERCENT) OF THE THREE MLPS ON THE IRIS AND SYNTHETIC DATA SETS IN TEN TRIALS

Date set	Original network			Directly trained network			Pruned network		
	NN	Mean	Std	NN	Mean	Std	NN	Mean	Std
Iris	4-10-3	94.80	2.63	3-3-3	95.33	2.37	3-3.3-3	96.00	1.99
Synthetic	3-15-2	83.10	2.92	2-9-2	84.29	2.45	2-9.5-2	86.67	1.66

Note: NN–Architecture of network; Mean–Average testing accuracy rate; Std–Standard deviation.

TABLE VI
SETTINGS OF PARAMETERS FOR TRAINING DIFFERENT MLPS

Data set	Original network	Pruned network
	N_H;Goal;Epoch;η;α	Goal;Epoch;η;α
Func1	10;0.01;2e+4;0.001;0.8	0.01;2000;0.001;0.8
Friedman #1	15;0.01;5e+3;0.001;0.8	0.01;2000;0.001;0.8
Ozone	15;0.01;1e+4;0.0001;0.8	0.01;2000;0.0001;0.8
Boston Housing	25;0.01;2e+4;0.0001;0.8	0.01;1000;0.0001;0.8

Note: N_H–Number of hidden units; η–Learning rate; α–Momentum constant.

zero, stop the pruning operation and obtain the final MLP with the only one hidden unit, else go to Step 3).

Step 3) Train the pruned MLP. If the performance of the pruned MLP is equal or better than that of the former one, return to Step 2), else stop the pruning operation and export the MLP before pruning in this iteration as the final MLP.

Note that for Step 3), the initial values of weights for each pruned MLP are all inherited from their corresponding values of weights in the trained MLPs in Step 2).

IV. EXPERIMENTAL RESULTS

In the experiments, all the MLPs are trained by the backpropagation algorithm with a momentum term, while the error functions are root mean square error (RMSE).

A. Classification

In this section, the performance of the present method is demonstrated on the *Iris* data set [9] and a synthetic one. The synthetic data set consists of the Ripley's data set [10] plus a noise feature satisfying $N(0, 0.01)$. We randomly split each of the two data sets into two equally sized sets for training and testing. Moreover, to make fair comparisons between different networks, ten Monte Carlo trials of each model upon the two data sets are also included.

According to Algorithm 1, the orders of features for the two data sets need to be obtained. It is shown in Figs. 1 and 2 that the obtained orders of features for the *Iris* data set and the synthetic data set are $4 \succ 3 \succ 2 \succ 1$ and $2 \succ 1 \succ 3$, respectively. The number of hidden units, the maximum number of iterations, the learning rate, the momentum constant, and the tolerance of termination for training MLPs in the first phase of the proposed method for the *Iris* data set are set to be 10, 10 000, 0.0001, 0.9, and 0.01, respectively. The settings of parameters for the synthetic data set are completely the same with the case for *Iris* except that the number of hidden units is taken as 15. The results for the two data sets are summarized in Tables I and II. It can be easily concluded from the two tables that the optimal MLPs selected in the first phase of our strategy upon the two data sets are the MLPs of 3-10-3 and 2-15-2, respectively.

For the second phase, the settings of parameters of the pruned MLPs are all the same with their corresponding values in the first phase except that the maximum number of iterations is 1000. Finally, the results of the final pruned MLP for the *Iris* and synthetic data sets are summarized in Tables III and IV. It can be concluded from Tables I–IV that our two-phase construction strategy can enhance the generalization ability of the MLP trained using all the features, which is denoted by the term "original network" in Tables III and IV. On the other hand, we need to check that the final pruned network trained with its initial weights inherited from the network obtained from the first phase outperforms the MLP trained directly with its weights randomly initialized. To make this further comparison fair, all the parameters of the directly trained MLPs of 3-2-3 and 2-9-2 for the two data sets are all the same with the pruned networks except that the maximum numbers of iterations are 19 000 and 17 000, which summarize their corresponding roles in the original and pruned networks. The results of the directly trained network on the two data sets are summarized in Tables III and IV. It can be found from the two tables that the generalization ability of the pruned network is better than that of its corresponding directly trained network. Moreover, it is shown in Table IV that the proposed method can avoid the high redundancy in the input units selection procedure.

Among the above experiments, the training and test sets of the *Iris* and synthetic data sets are all chosen randomly. Thus, to make the comparisons more convincing, ten Monte Carlo trials for each of the aforementioned networks are performed. Nonetheless, the numbers of features chosen by the present method may be different for different training sets. Therefore, the number of the subset of features can be taken as the number with the most frequency in the ten trials. Based on our simulations, it can be found that the number of input units for the *Iris* data set that can be taken is 3 because it is with the most frequency (7 out of 10), while it is 2 for the synthetic data set (8 out of 10). For the second phase, we report the mean of numbers of hidden units among the pruned networks, while the numbers of hidden units for the directly trained networks are all the integral part of this mean in the ten trials.

It is shown in Table V that the generalization ability of the proposed method is the best among the three types of networks. Furthermore, the proposed method achieves the smallest standard deviation of testing. This means that the performance of the present method on the two data sets is the most stable.

B. Function Approximation

The following problems in this section are all function approximation tasks. The description of the four data sets is as follows.

1) Func1: This synthetic data set is generated by the function $y(x_1, x_2) = (1/2)\exp\{\cos(4x_1 + 4x_2)\}$. The training and test sets are constructed by evenly spaced 8×8 and 40×40 grids on $[0, 1] \times [0, 1]$, respectively. Moreover, the training data are contaminated by a Gaussian noise $N(0, 0.01)$.

2) Friedman #1: This synthetic data set is taken from [11]. Two hundred forty samples contaminated by a Gaussian noise $N(0, 1)$ are randomly chosen for training, while 1000 noise-free samples are chosen for testing.

3) Ozone: This real-world data set [12] consists of 366 samples. The feature with the largest number of missing values is eliminated and the samples containing missing values are removed. Then, 250 randomly chosen samples are used for training and the remaining 80 samples are used for testing.

4) Boston Housing: This real-world data set [9] consists of 506 samples with 14 variables. Four hundred eighty one samples of the data set are chosen for training and the remaining 25 samples are used for testing.

TABLE VII
EXPERIMENTAL RESULTS OF THE THREE DIFFERENT MLPS ON THE FOUR FUNCTION APPROXIMATION PROBLEMS

Data set	Original network			Directly trained network			pruned network		
	NN	E_{train}	E_{test}	NN	E_{train}	E_{test}	NN	Etrain	E_{test}
Func1	2-10-1	0.0832	0.0576	2-4-1	0.0950	0.0593	2-4-1	0.0890	0.0359
Friedman #1	10-15-1	1.1967	2.5526	5-10-1	0.4381	0.7841	5-10-1	0.4312	0.7117
Ozone	8-15-1	3.5987	4.5812	3-7-1	4.0724	4.0213	3-7-1	4.0470	3.9434
Boston Housing	13-25-1	2.4354	3.2052	5-18-1	3.1781	3.2576	5-18-1	3.0834	2.7728

Note: NN–Architecture of network; E_{train}–Training error; E_{test}–Test error.

TABLE VIII
EXPERIMENTAL RESULTS OF THE TWO DIFFERENT METHODS ON THE FIVE DATA SETS

Data set	Engelbrecht's method [1]						Our method					
	Original network			Pruned network			Original network			Pruned network		
	NN	Acc_{train}	Acc_{test}	NN	Acc_{train}	Acc_{test}	NN	Acc_{train}	Acc_{test}	NN	Acc_{train}	Acc_{test}
Iris	4-10-3	97.1%	97.7%	2-2-3	96.2%	97.7%	4-10-3	98.67%	96.00%	3-2-3	97.33%	98.67%
Wine	13-10-1	100%	98%	6-3-3	96.1%	95.9%	13-10-3	100%	97.78%	7-6-3	100%	98.89%
Hepatitis	19-25-1	94.3%	80%	4-4-2	78.9%	83.3%	19-25-2	100%	79.49%	3-8-2	85.71%	84.62%
Diabetes	8-40-1	72%	68%	6-8-2	70.5%	69.1%	8-40-2	94.27%	70.05%	6-8-2	83.07%	74.22%
Cancer	9-10-1	96.4%	98.1%	3-1-2	96.2%	97.8%	9-10-2	98.53%	96.49%	3-3-2	96.77%	96.78%

TABLE IX
EXPERIMENTAL RESULTS OF SVMS WITH DIFFERENT TYPES OF KERNEL FUNCTIONS AND THE PROPOSED METHODS ON THE FIVE DATA SETS

Data set	SVM				Our method	
	Linear kernel		Gaussian RBF kernel			
	Acc_{train}	Acc_{test}	Acc_{train}	Acc_{test}	Acc_{train}	Acc_{test}
	λ		(λ, γ)			
Iris	95.95%	96.05%	95.95%	97.37%	97.33%	98.67%
	10		$(10^3, 2^{-10})$			
Wine	100%	96.67%	97.73%	98.89%	100%	98.89%
	10^{-2}		$(1, 2^{-10})$			
Hepatitis	98.70%	88.46%	98.70%	89.74%	85.71%	84.62%
	1		$(1, 2^{-4})$			
Diabetes	80.73%	74.74%	95.57%	69.01%	83. 07%	74.22%
	10^{-2}		$(1, 2^{-1})$			
Cancer	96.48%	98.25%	100%	96.78%	96.77%	96.78%
	10^{-1}		$(1, 1)$			

The settings of parameters for the original and the pruned MLPs are all included in Table VI. The settings of the parameters of the directly trained MLPs for the four problems are all the same with their corresponding roles of the original networks except that the maximum numbers of iterations are 32 000, 15 000, 26 000, and 21 000, respectively. Table VII depicts the results for the four problems. It can be concluded from Table VII that the pruned MLPs can achieve better generalization ability in comparison with the original MLPs and the directly trained MLPs.

C. Comparison With Related Work

In the following, the proposed method is compared with Engelbrecht's work [1]. The learning rate, the momentum constant, and the tolerance of termination for the original MLPs are all 0.0001, 0.9, and 0.01, respectively. The maximum numbers of iterations for the original MLPs upon the five data sets, i.e., Iris, Wine, Hepatitis, Diabetes, and Cancer, are 10 000, 5000, 5000, 1000, and 1000, respectively. The numbers of hidden units of the original MLPs are completely the same with those used in [1]. For the pruned MLPs, the settings of parameters over the five data sets are all the same as their corresponding original MLPs except that the maximum numbers of iteration for the five data sets are 1000, 100, 200, 100, and 100, respectively.

In comparison with Engelbrecht's method, it can be easily found from the results in Table VIII that the proposed method can achieve better generalization ability on all data sets except *Cancer*. However, for the *Cancer* data set, though with nearly 1% less testing accuracy rate by comparing to Engelbrecht's result, the final pruned MLP of the proposed method can improve the generalization ability of its original network while Engelbrecht's method cannot fulfill.

TABLE X
EXPERIMENTAL RESULTS OF SVRS WITH DIFFERENT TYPES OF KERNEL FUNCTIONS AND THE PROPOSED METHODS ON THE FOUR DATA SETS

Data set	SVR				Our method	
	Linear kernel		Gaussian RBF kernel			
	E_{train}	E_{test}	E_{train}	E_{test}	E_{train}	E_{test}
	λ		(λ, γ)			
Func1	0.4180	0.3519	0.0832	0.0739	0.0890	0.0359
	10		$(1, 1)$			
Friedman #1	2.5210	2.6931	2.5271×10^{-6}	1.1605	0.3354	0.6563
	10^{-2}		$(10^{6}, 2^{-6})$			
Ozone	4.5727	4.4124	3.6025	3.7478	4.0747	3.7856
	10^{-2}		$(10^{-1}, 2^{-4})$			
Boston Housing	4.9383	4.9756	4.4984×10^{-6}	3.6870	2.6703	2.8499
	10^{6}		$(10^{2}, 2^{-1})$			

TABLE XI
VALUES OF SENSITIVITY AND MI OVER THE THREE DIFFERENT INTERVALS AND THE TWO DIFFERENT DISTRIBUTIONS FOR $y(x) = 1/(1 + \exp\{-2x\})$

Distribution of Samples	SA based method			MI based method		
	$[-6, -2]$	$[-2, 2]$	$[2, 6]$	$[-6, -2]$	$[-2, 2]$	$[2, 6]$
Uniform distribution	0.0969	0.2002	0.0965	4.1174	4.1174	4.1174
Normal distribution	0.0799	0.1682	0.0809	4.0969	4.1062	4.1018
$N(\mu, \sigma^2)$	$N(-4, 0.7)$	$N(0, 0.7)$	$N(4, 0.7)$	$N(-4, 0.7)$	$N(0, 0.7)$	$N(4, 0.7)$

TABLE XII
VALUES OF SENSITIVITY AND MI OVER THE CASES OF INDEPENDENT AND DEPENDENT INPUTS FOR $y(x_1, x_2) = \sin(6x_1) + \sin(6x_2)$

Function dependency	SA based method		MI based method	
between inputs	Input 1	Input 2	Input 1	Input 2
Input independent	1.0223	1.0206	0.2038	0.2017
$x_2 = x_1^3$	0.5111	0.3575	0.3277	0.3505
$x_2 = \sin(2x_1)$	1.2494	2.9172	0.3169	0.3277
$x_2 = \sin(4x_1) + x_1^2(x_1 + 1)$	1.7287	3.5010	0.4920	0.4096

D. Comparisons With SVMs and SVRs

In this section, we compare the proposed method with the state-of-the-art methods, i.e., support vector machine (SVM) and support vector regression (SVR). The linear kernel function $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}$ and the Gaussian kernel function $K(\mathbf{x}, \mathbf{x}') = \exp\{-\gamma\|\mathbf{x} - \mathbf{x}'\|^2\}$ are both utilized to construct SVMs and SVRs. The parameters in the kernel functions and the penalty parameter λ, which control the tradeoff between the margin and the error, are all tuned by grid search. Moreover, the training and test sets for the SVMs and SVRs are completely the same with those utilized by our proposed approach. The insensitivity parameter ϵ of the following SVRs is set to be 10^{-7}.

Table IX shows the results obtained by the SVMs. The results of our proposed method are all directly taken from Table VIII. Table X lists the experimental results obtained by the SVRs. Thereafter, based on the results in Table IX, one can find that our proposed method is comparable to the SVMs. Moreover, it turns out from Table X that the generality ability of the proposed method is superior to the SVRs.

E. Discussion

In this section, a comparative study between the SA- and MI-based methods is conducted. Towards the SA-based approach, the architectures of MLPs are, respectively, 1-3-1 and 2-10-1 for the following two cases, while the learning rates, the momentum constants, the maximum numbers of iterations and the tolerances of terminations for all the MLPs are 0.001, 0.8, 10 000, and 0.001, respectively.

1) Interval and Distribution of Input: For the same underlying function, different values of sensitivity can be obtained for the samples in different intervals and for the samples with different distributions. In these two cases, the relevance of input units may become inaccurate according to the SA-based method. However, the MI-based method can provide the satisfying results.

In the following, an experiment is constructed to validate the above description. The function used to generate samples is $y(x) = 1/(1 + \exp\{-2x\})$ $(x \in [-6, 6])$. To set up the experiment, the whole interval $[-6, 6]$ is divided into three intervals with the same length. First, the three sets of 400 samples in each are generated uniformly with the same step length on the three intervals. The MI and the sensitivities upon the three intervals can be obtained as shown in Table XI.

Second, to observe the influences produced by the distribution of samples on the two methods, the normal distribution with mean of $-4, 0$, and 4 and variances of 0.7 are used to generate 400 samples each on the three intervals. The results of the MI-based method and the SA-based method are both summarized in Table XI. Table XI shows that with the same distribution, either uniform or normal, the sensitivity on the interval $[-2, 2]$ is obviously different from those on the

other two intervals. However, the values of MI on the three intervals are completely or approximately same. It is also shown in Table XI that for the different distributions, the percentage of change for the MI-based method is much less than that for the SA-based method. Therefore, for the above two cases, the results in Table XI indicate that the SA-based method cannot provide an exactly or approximately fixed value while the MI-based method can.

2) Dependency Between Inputs: As shown in [13], the SA-based method for neural networks may be inefficient when the inputs of the networks are dependent. However, the MI-based method can overcome this limitation successfully. To validate this assertion, one function in [13], $y(x_1, x_2) = \sin(6x_1) + \sin(6x_2)$ $(x_1, x_2 \in [0, 1])$, is utilized to generate the samples. For the case without dependency between the two inputs, 400 samples are constructed by evenly spaced 20×20 grids on $[0, 1] \times [0, 1]$. For the other three cases with dependencies between the two inputs, the values of the first input x_1 are uniformly distributed over the interval $[0, 1]$, while the values of the second input x_2 are determined by those of x_1 using the functional relationships depicted in Table XII.

The values of MI and sensitivity for the two inputs are summarized in Table XII. Therefore, for each of the three cases with dependencies between the two inputs, it is shown in the table that the values of sensitivities for the two inputs are quite different, while the values of MI are approximately the same. Thus, this investigation confirms the findings in [13] towards the SA-based method. Moreover, from Table XII, one can find that the MI-based method can provide a solution to overcome this problem.

V. Conclusion

To construct a compact MLP together with better generalization ability, a novel two-phase construction method is proposed for pruning both input and hidden units. In the first phase, the order of features can be ranked by the MI of target outputs with respect to certain feature subsets together with a forward strategy. Then, the salient features can be identified and taken as the input units of an MLP according to the ranking results and by considering their contributions to the network's performance. In the second phase, the redundant hidden units can be removed using a novel relevance measure. In our experiments, the proposed method shows its efficiency towards both classification and function approximation problems. Compared with representative results of the SA-based pruning strategy, the present method shows its better performance. Moreover, the proposed method demonstrates the comparable performance in comparison with the SVMs, and superior generalization ability compared to the SVRs. The investigation shows that, for the samples on the different intervals or the samples with different distributions on the same interval, the values of sensitivity provided by the SA-based method are not exactly or approximately the same, while the values of MI given by the MI-based method are. In addition, the investigation also exhibits that when there are dependencies between inputs, the SA-based method can be ineffective while the MI-based method can successfully avoid this limitation.

References

[1] A. P. Engelbrecht, "New pruning heuristics based on variance analysis of sensitivity information," *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1386–1399, Nov. 2001.

[2] X. Zeng and D. S. Yeung, "Hidden neuron pruning of multilayer perceptrons using a quantified sensitivity measure," *Neurocomputing*, vol. 69, pp. 825–837, 2006.

[3] I.-S. Oh, J.-S. Lee, and B.-R. Moon, "Hybrid genetic algorithms for feature selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 11, pp. 1424–1437, Nov. 2004.

[4] G. Van Dijck and M. M. Van Hulle, "Speeding up feature subset selection through mutual information relevance filtering," in *Lecture Notes in Artificial Intelligence*. Berlin, Germany: Springer-Verlag, 2007, vol. 4702, pp. 277–287.

[5] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[6] N. Kwak and C.-H. Choi, "Input feature selection by mutual information based on Parzen window," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 12, pp. 1667–1671, Dec. 2002.

[7] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Phys. Rev. E, Stat. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 69, 2004, 0661138.

[8] H. Stögbauer, A. Kraskov, S. A. Astakhov, and P. Grassberger, "Least dependent component analysis based on mutual information," *Phys. Rev. E, Stat. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 70, no. 6, 2004, 066123.

[9] C. Blake and C. Merz, UCI Repository of Machine Learning Datasets, 1998 [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[10] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1996.

[11] J. H. Friedman, "Multivariate adaptive regression splines," *Ann. Statist.*, vol. 19, no. 1, pp. 1–141, 1991.

[12] L. Breiman and J. Friedman, "Estimating optimal transformations in multiple regression and correlation," *J. Amer. Statist. Assoc.*, vol. 80, pp. 580–619, 1985.

[13] M. A. Mazurowski and P. M. Szecówka, "Limitations of sensitivity analysis for neural networks in cases with dependent inputs," in *Proc. 4th IEEE Int. Conf. Comput. Cybern.*, Tallinn, Estonia, 2006, pp. 299–303.

Constructing Sparse Kernel Machines Using Attractors

Daewon Lee, Kyu-Hwan Jung, and Jaewook Lee

***Abstract*—In this brief, a novel method that constructs a sparse kernel machine is proposed. The proposed method generates attractors as sparse solutions from a built-in kernel machine via a dynamical system framework. By readjusting the corresponding coefficients and bias terms, a sparse kernel machine that approximates a conventional kernel machine is constructed. The simulation results show that the constructed sparse kernel machine improves the efficiency of testing phase while maintaining comparable test error.**

***Index Terms*—Attractors, dynamical systems, kernel method, sparse kernel machines, support vector domain description (SVDD), support vector machine (SVM).**

I. Introduction

Kernel-based learning methods [1]–[3] have been successfully applied to a variety of pattern recognition problems. Despite their outstanding generalization performance, kernel methods are computationally expensive in the test phase since their computational complexity is proportional to the number of training points needed to represent the

Manuscript received October 23, 2007; revised August 03, 2008 and August 03, 2008; accepted January 02, 2009. First published February 27, 2009; current version published April 03, 2009. This work was supported by the Korea Research Foundation under Grant KRF-2007-313-D00918.

D. Lee is with the Max Planck Institute for Biological Cybernetics, 72076 Tübingen, Germany.

K.-H. Jung and J. Lee are with the Department of Industrial and Management Engineering, Pohang University of Science and Technology, Pohang, Kyungbuk 790-784, Korea (e-mail: jaewookl@postech.ac.kr).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2009.2014059