Journal of Alloys and Compounds 279 (1998) 30–38

Materials structure-property prediction using a self-architecting neural network

C.L. Philip Chen[a,*], Yang Cao[a], Steven R. LeClair[b]

[a]Department of Computer Science and Engineering, Wright State University, Dayton, OH 45435, USA
[b]Materials and Manufacturing Directorate, U.S. Air Force Research Laboratory, Wright-Patterson Air Force Base, Dayton, OH 45433, USA

Abstract

An important trend in materials research is to predict properties for a new material before committing experimental resources. Often the prediction is motivated by the search for a material with a unique combination of properties. The selection of a property or feature is crucial to the plausibility of the prediction. This paper proposes the use of a self-architecting neural network to model the relation between materials structure and properties for the purpose of predicting the properties of new materials, i.e. to predict properties for an unknown compound. In this paper, we summarize the prediction attained with the proposed neural network structure referred to as the Orthogonal Functional Basis Neural Network (OFBNN). The OFBNN, which combines a new basis selection process and a regularization technique, not only gives us a more computationally tractable method, but better generalization performance. Simulation studies presented here demonstrate the performance, behavior and advantages of the proposed network. © 1998 Elsevier Science S.A. All rights reserved.

Keywords: Neural network; Materials; Property prediction; Unsupervised learning; Supervised learning

1. Introduction

The objective of this research is to extend the capability of discovery methods materials research using unsupervised and supervised learning to predict material properties. In order to predict properties for a new material, it is generally acknowledged that classes or clusters of compounds which have similarities properties must be formed [14]. Unsupervised learning is a well recognized paradigm [15,16] where there is no feedback from the environment to indicate what the desired output should be or whether the output is correct. Using unsupervised learning, the system must discover for itself any relationships of interest, such as patterns, features, property values, correlations, or categories in the input and translate the discovered relationship into outputs. We first introduce a conventional unsupervised learning algorithm, K-medoid partitioning algorithm, for grouping and clustering. To complement the unsupervised algorithm, an innovative supervised functional learning algorithm and structure, Orthogonal Functional Basis Neural Network (OFBNN), is proposed to find functional relationship within the learned clusters.

Traditional methods of function approximation (learning) or regression involve a linear combination of the product of single variable or fixed basis functions (e.g., polynomial, spline, and/or trigonometric expansions). From Barron [10], the problem with traditional methods is that there are exponentially many orthonormal functions, but unless all of these orthonormal functions are used in the fixed basis, there will remain functions that are not well approximated, i.e., the order of the squared approximation error is $1/n^{(2/d)}$, where n is the number of basis functions and d is the number of input variables. This problem is avoided by tuning or adapting the parameters of multivariable basis functions to fit the target function as in the case of neural networks, wherein the order of the squared approximation error is $1/n$.

The biological origins of neural networks [11], as chronicled by Pao [12], established the multi-variable sigmoid as 'the' basis function for neural networks. Today the suite of multivariable basis functions employed in neural networks is without bound, but the most commonly used are the sigmoid and radial basis functions. Radial basis function neural networks typically employ subset selection to identify a set of Gaussian basis functions. Broomhead and Lowe [2] have shown how to choose such a subset randomly from the entire given set. In lieu of random selection, Rawlings [1] proposed a systematic approach that employs forward selection to choose the

[*]Corresponding author.

Table 1
The four-cluster result using self-clustering K-medoid algorithm

Clusters	Compounds										
Cluster#1:	5	9	11	14	15	17	18	19	20	23	$\underline{24}$
	25	31	32	33	34	38	40	41	46		
Cluster #2:	26	28	43	44	45	47	48	$\underline{49}$	50	52	53
Cluster#3:	1	2	3	4	6	7	8	10	12	$\underline{13}$	16
	29	30	35	36	37	39	42				
Cluster#4:	$\underline{21}$	22	27	51							

subset that best explains the variation in the dependent variable incrementally.

Based on this concept, Chen et al. presented an efficient implementation of forward selection using the orthogonal least square method (OLS) [3]. Subset selection can also be used to avoid overfitting by limiting the complexity of the network. From the literature, overfitting may be avoided when combining subset selection with other methods such as regularization [5,6], and as contributed by Mark Orr, combining of OLS and regularization [4]. We propose the combined use of an unsupervised learning method and the OFBNN as applied to electro-optic (EO) data of several semiconductor compounds to demonstrate the advantage of the OFBNN over other neurocomputing methods.

2. Unsupervised learning

Unsupervised learning is an important major learning paradigm where there is no feedback from the environment to indicate what the desired output should be or whether the output is correct. Using unsupervised learning, the system must discover for itself any relationships of interest, such as patterns, features, property values, correlations, or categories in the input, and translate the discovered relationship into outputs. Conventional unsupervised learning algorithms vary from sundry data clustering algorithms to more recent neurocomputing approaches employing Hebbian [15], Kohonen's [16] and Grossberg's [17] learning rules.

Because most neurocomputing approaches to unsupervised learning suffer from varying results across multiple runs, in this research we use a conventional unsupervised K-medoid partitioning learning algorithm (PAM), for grouping and clustering [13]. Given the same data set and the same measurement, K-medoid partitioning algorithm guarantees giving us the same result every time. Tables 1 and 2 show the clustering result for partitioning the EO data (see Appendix A) to four and eight clusters, respectively. The underlined compound within each cluster is the medoid of that cluster. According to the definition of a K-medoid silhouette, eight-cluster partitioning is preferred as it provides better results.

In the next section, a supervised learning method referred to as the OFBNN is used to predict the property or feature value of interest from a set of compounds which have been organized into a class of similar compounds using the above identified K-medoid method for generating clusters.

3. 'Functional' learning via OFBNN

3.1. Supervised learning

The goal of function approximation, heretofore referred to as regression, is to learn the mapping or function relating the input $\mathbf{x}$, where $\mathbf{x} \in \mathbf{R}^{\mathbf{m}}$ and the output $\mathbf{y}$. Herein, the neurocomputing approach distinguishes from most by: (1) emphasizing the use of multi-variable 'squashing' functions for the basis set, and (2) achieving a mapping such that

$$\boldsymbol{y} = f(\boldsymbol{x}) = \sum_{j=1}^{N} w_j f_j(\boldsymbol{x}),$$

Table 2
The eight-cluster result using self-clustering K-medoid algorithm

Clusters	Compounds											
Cluster#1:	5	9	15	$\underline{17}$	18	19	24	25	31	34	38	41
Cluster #2:	44	48	$\underline{49}$	50	52	53						
Cluster#3:	1	2	3	6	12	$\underline{13}$	16	29	30	35	36	
	37	39	40	42								
Cluster #4:	$\underline{21}$	22										
Cluster #5:	4	7	8	$\underline{10}$	14	20	26	32				
Cluster#6:	11	$\underline{23}$	33									
Cluster #7:	27	$\underline{51}$										
Cluster #8:	28	43	45	46	$\underline{47}$							

wherein only the linear coefficients $\{w_j\}_1^N$ need to be learned [6]. Yet despite their inherent ease of use and their advantageous approximation error, neurocomputing approaches are subject to the same failings as are all other methods of function approximation – a 'mapping' as opposed to a 'functional' mapping, and therein, little or no ability to discern signal from signal-plus-noise.

Today the suite of multi-variable basis functions employed in neural networks is without bound, but the most commonly used are the sigmoid and radial basis functions. Radial basis function neural networks typically employ subset selection to identify a set of Gaussian basis functions. Broomhead and Lowe [4] have tried to choose such a subset randomly from the entire given set. In lieu of random selection, Rawlings [1] has proposed a systematic approach that employs forward selection to choose the subset that best explains the variation in the dependent variable incrementally.

One approach for selecting RBF centers is subset selection, in which a possible subset is chosen to form the regressors from a given regressor set. This method has the advantage of producing an efficacious network [1,2]. With data sets from noisy environments, functional mapping is expressed as, $y_i = f(\mathbf{x}_i) + \varepsilon_i$, where ε_i represents the error and is assumed to be uncorrelated with $f(\mathbf{x}_i)$. From the literature, the general approach to mapping noisy data sets is the regularization approach suggested by Tikhonov and Arsenin [7]. For example, the zero-order regularization is to minimize the following objective function: $E = \mathbf{e}^T\mathbf{e} + \lambda\mathbf{g}^\mathbf{T}\mathbf{g}$ where $\mathbf{e} = \mathbf{y} - f(\mathbf{X})$ and $\mathbf{g}$ is the linear weight vectors. In general, regularization results in small weights in the final 'smoothed' functional form. Mark Orr further enhanced OLS and reported on the benefits of regularization involving the selection of RBFs [4]. OLS can be viewed as a more efficient implementation of forward selection in the context of subset selection. The detailed procedure is described in reference [3]. During the OLS process, each orthogonal basis $\mathbf{h}_i^{\mathrm{OLS}}$ is obtained in the following way. The computational procedure can be simply represented as:

$$\mathbf{h}_1^{\mathrm{OLS}} = \mathbf{f}_1, a_{ik} = \frac{(\mathbf{h}_i^{\mathrm{OLS}})^T\mathbf{f}_k}{(\mathbf{h}_i^{\mathrm{OLS}})^T\mathbf{h}_i^{\mathrm{OLS}}}, 1 \le i \le k, \mathbf{h}_k^{\mathrm{OLS}} = \mathbf{f}_k - \sum_{i=1}^{k-1} a_{ik}\mathbf{h}_i^{\mathrm{OLS}} \tag{1}$$

By combining OLS and forward selection, the regressors can be selected [3]. However, if too many regressors are used in the functional mapping, it will cause the network to be overly sensitive to the training data which often results in poor generalization. To avoid this situation, the zero-order regularization can be employed. The OLS has been shown to be an effective method and has been fundamental in the implementation of forward selection for both ROLS [8] and regularization in the selection of RBF centers [4]. However, the OLS is insufficient to use in forward selection [9]. In the next section, we will discuss a new approach for selecting basis.

In the next section, we will summarize basis selection process by a simple transformation, Orthogonal Function Transformation (OFT). We will also propose a new neural network architecture, Orthogonal Functional Basis Neural Network (OFBNN) based on this transformation.

3.2. Orthogonal Functional Transformation (OFT) and Orthogonal Functional Basis Neural Network (OFBNN)

Based on the above discussed approach, we will construct k orthogonal basis functions, $\mathbf{h_1}, \mathbf{h_2}, ..., \mathbf{h}_l$, from the original $\mathbf{F}$ set, where l is the rank of $\mathbf{F}$. Each orthogonal basis, $\mathbf{h_i}$, is a linear combination of orthogonal components of each member in the given set $\mathbf{F}$. We also show that based on this decomposition. a new neural network structure, OFBNN, is proposed.

The OFT can be obtained by rearranging the columns of the $\mathbf{F} \equiv [\mathbf{f}_1^{(1)}, \mathbf{f}_2^{(1)}, ..., \mathbf{f}_N^{(1)}]$ matrix through an iterative procedure, where the superscript denotes the iteration number. In order to explain the OFT procedure, let us define $\Psi \equiv [\varphi_1, \varphi_2, ..., \varphi_N] =$ rearrangement ($\mathbf{F}$). The rearrangement is based on the following procedure. Define

$$\alpha(k) = \arg(\max\{\|\mathbf{f}_i^{(k)}\|^2, i = 1,2,...,N\}),$$

and the index, $\mathbf{t} = \alpha(1)$ at the first step ($k = 1$, denote as the superscript). The first column of the Ψ matrix is, $\varphi_1 \equiv \mathbf{f}_\mathbf{t}^{(1)}$, and the first orthogonal basis is:

$$\mathbf{h}_1 = \sum_{i=1}^{N} \frac{\langle \mathbf{f}_\mathbf{t}^{(1)}, \mathbf{f}_i^{(1)} \rangle}{\|\mathbf{f}_\mathbf{t}^{(1)}\|^2} \mathbf{f}_\mathbf{t}^{(1)}$$

At the kth step, $k \ge 2$, calculate $\mathbf{f}_i^{(k)}$ and $\mathbf{h}_k$ as

$$\mathbf{f}_i^{(k)} = \mathbf{f}_i^{(k-1)} - \frac{\langle \mathbf{f}_i^{(k-1)}, \mathbf{f}_\mathbf{t}^{(k-1)} \rangle}{\|\mathbf{f}_\mathbf{t}^{(k-1)}\|^2} \mathbf{f}_\mathbf{t}^{(k-1)}, i = 1,2,...,N. \tag{2}$$

$$\mathbf{h}_k = \sum_{i=1}^{N} \frac{\langle \mathbf{f}_\mathbf{t}^{(k)}, \mathbf{f}_i^{(k)} \rangle}{\|\mathbf{f}_\mathbf{t}^{(k)}\|^2} \mathbf{f}_\mathbf{t}^{(k)} \tag{3}$$

$$\varphi_k = \mathbf{f}_\mathbf{t}^{(k)} \tag{4}$$

This process continues until $\|\mathbf{f}_\mathbf{t}^{(k)}\|^2 \le \varepsilon$, where $\varepsilon = 10^{-6}$.

In this way, we have the Ψ matrix obtained from the rearrangement of the $\mathbf{F}$ matrix. The columns in the Ψ matrix are rearranged in descending order of the norm of the orthogonal basis contributed by the original $\mathbf{f}$ vectors.

We also show that the new orthogonal basis function set has better convergence when combined with the regularization process [9].

Based on the above discussion, we obtain a new neural network architecture, OFBNN, shown in Fig. 1(a), in which an extra layer for orthogonal transformation is added.

Here we summarize the learning algorithm for OFBNN below.

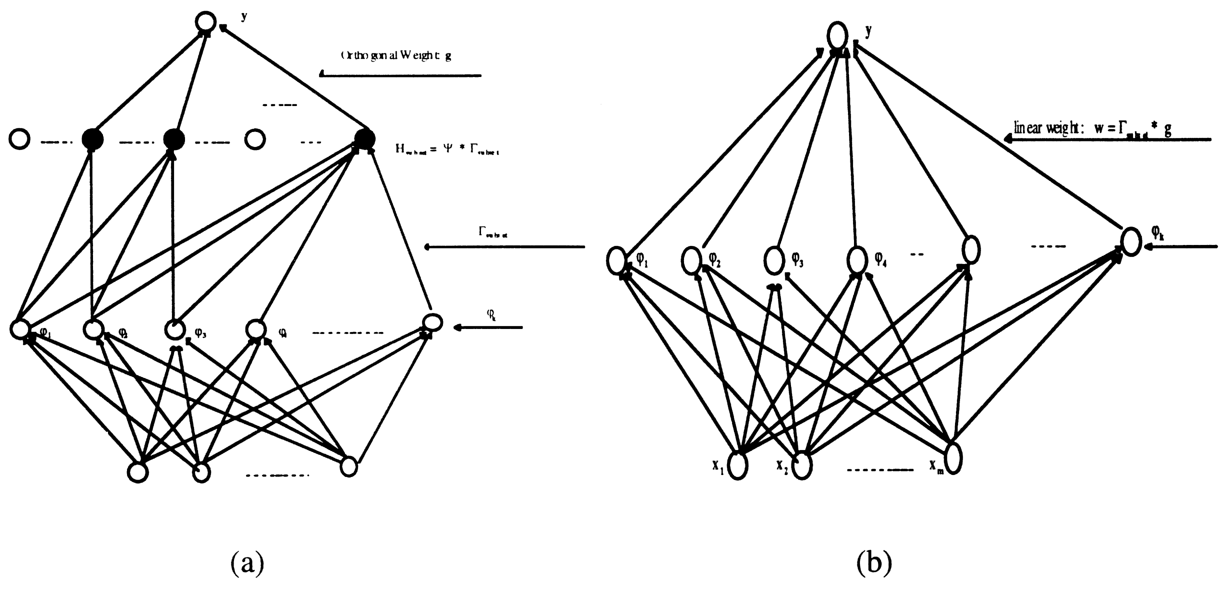

Fig. 1. Architecture of OFBNN.

3.2.1. Algorithm Orthogonal Functional Basis Functional Mapping (OFBFM)

Input: The training patterns $\{\mathbf{x}_t, y_t\}_{t=1}^{P}$.

Output: The connection weights, **g**, the orthogonal basis set, **H**, the subset, $\mathbf{H}_{\text{subset}}$, the linear weight, **w**, connected from the nodes in Ψ layer to the output nodes, and the mapping.

Step 1. Construct a Heterogeneous Regressor Set:: $\mathbf{F} = \{\mathbf{f}_i\}_{i=1}^{N}$. Build the regressor matrix **F**.

We use a combination of heterogeneous functions.

Step 2: Build orthogonal basis matrix **H** according to Eq. (3).

Step 3: Initialization: Let $k=1$, and $\mathbf{H}_{\text{subset}}^{(k)} = \phi$, where ϕ is an empty set.

Step 4 Subset selection, regularization, and generalized cross validation (GCV):

The original mapping problem can be transformed to $\mathbf{y} = \mathbf{Hg} + \mathbf{e}$. With zero-order regularization employed, the objective function is $E = \mathbf{e}^T\mathbf{e} + \lambda \mathbf{g}^T\mathbf{g}$.

We can use a similar approach to find the most efficacious subset of **H** [4]. The GCV is used as the stopping criterion. Find $\mathbf{h}_i^{(k)}$ such that

$$\max_i \left\{ \frac{(\mathbf{y}^{\mathrm{T}}\mathbf{h}_i^{(k)})^2}{\lambda + (\mathbf{h}_i^{(k)})^{\mathrm{T}}\mathbf{h}_i^{(k)}} \right\} \tag{5}$$

The coefficient is

$$g_i = \frac{\mathbf{y}^T\mathbf{h}_t^{(k)}}{\lambda + \mathbf{h}_i^{(k)^T}\mathbf{h}_i^{(k)}} \quad 1 \le i \le k$$

Include $\mathbf{h}_i^{(k)}$ as an element of the $\mathbf{H}_{\text{subset}}^{(k)}$, i.e., $\mathbf{H}_{\text{subset}}^{(k)} \leftarrow \mathbf{H}_{\text{subset}}^{(k)} \cup \mathbf{h}_i^{(k)}$.

Because all the columns in **H** are orthogonal to each other, in implementation of selecting $\mathbf{H}_{\text{subset}}$ we can set the corresponding selected column to 0, and the next $\mathbf{h}_1$ that satisfies Eq. (5) can be selected easily. Compared with the computational cost of order $\mathbf{o}(P^2)$–$\mathbf{o}(P^3)$ for $\mathbf{h}_i^{(k)}$ shown in [4] where

$$\mathbf{H} \leftarrow \mathbf{H} - \frac{\mathbf{h}_i^{(k)^T}\mathbf{h}_i^{(k)}\mathbf{H}}{\mathbf{h}_i^{(k)^T}\mathbf{h}_i^{(k)}},$$

our process is much more efficient. In general, the computational complexity to generate the **H** matrix using our OFT method described above is the order of $\mathbf{O}(k^*N)$, where k is the rank of **F** and N is the number of the given regressors. In functional mapping, where $P \gg N$, our approach takes advantage of replacing the complicated process of updating orthogonal basis with a pre-selection process.

For regularization, we modify λ based upon the GCV derivation as described by [4]. Stop if GCV value reaches its minimum point; otherwise $k=k+1$, repeat this step.

3.2.2. End of OFBFM learning algorithm

The OFBFM finds the $\mathbf{H}_{\text{subset}}$ with d basis, and the orthogonal weights $\mathbf{g} = [\mathbf{g}_1, \ldots, \mathbf{g}_d]$. Since $\mathbf{H} = \Psi\Gamma_{\text{subset}}$ and $\mathbf{H}_{\text{subset}}$ is the subset of **H**, we have $\mathbf{H}_{\text{subset}} = \Psi\Gamma_{\text{subset}}$. Finally, the system equation is: $\hat{\mathbf{y}} = \mathbf{H}_{\text{subset}}\ \mathbf{g} = \Psi\Gamma_{\text{subset}}\ \mathbf{g} = \Psi\mathbf{w}$, where $\mathbf{w} = \Gamma_{\text{subset}}\mathbf{g}$ is the equivalent final weight from the original basis nodes to the output nodes. The final architecture is shown in Fig. 1(b).

4. Simulation results

In this section, we apply the proposed OFBNN to the EO data listed in Appendix A. The OFBNN is applied to predict feature values of unknown compounds for the four-cluster result, shown in Fig. 2 and 3. We apply OFBNN to learn the functional relationship between

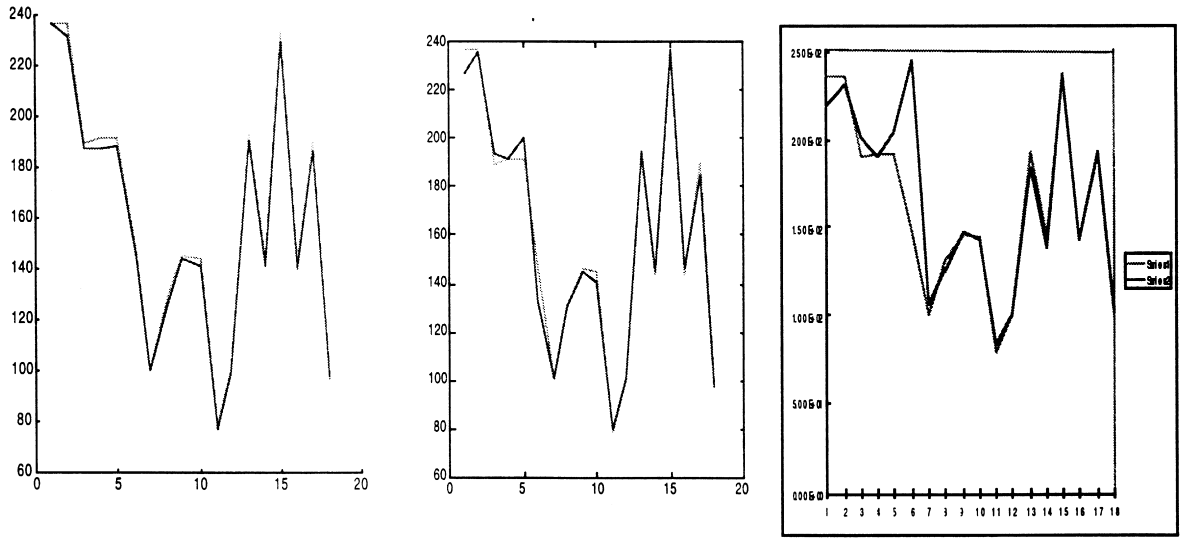

Fig. 2. Predict a feature value of a compound from Table 1, Cluster #1.

attributes of 18 compounds in Cluster #3 from Table 1. In Fig. 2(a) the training and testing results are shown. The training set is comprised of 17 compounds, while one compound (compound number seven) is used for testing. The feature value number four (weight) is the independent variable and is predicted by the OFBNN.

In this simulation, the desired feature value is 146.7700 and the predicted value is 145.0495 for OFBNN. The elapsed training time is 3.4400 s on a Macintosh PowerPC 8100/175 machine. The same data set is tested on a Functional-Link neural network (FLNN) using conjugate-gradient search method. The predicted result is 132.4781, and it takes 9.0690 s on the same machine. The result is shown in Fig. 2(b). The same data set is also tested on a 5-3-1 multi-layer neural network (MLNN) using the BP algrorithm [18]. The result is shown in Fig. 2(c), where the training MSE is 0.041 running on 2075 epochs, the testing MSE is 576.7140, the predicted result is 244.424. The BP algorithm gives the worst generalized result.

We also learn the function of compounds in Cluster #2 from Table 1, which consists of 11 compounds. Among them, ten compounds are used for training, another compound (compound number 45) is used for testing. In Fig. 3(a) the training and testing plot for the OFBNN is shown. In this simulation, the desired feature value is 181.8360 and the predicted value is 181.8360. The elapsed training time is 0.3590 s. We also apply FLNN to the same data set, the predicted result using this method is 181.8580, and it takes 5.8650 s on the same machine. The result is shown in Fig. 3(b). A 5-3-1 MLNN takes 780 epochs to reach training MSE 0.047, testing MSE is 0.8212, the predicted value is 188.9185.

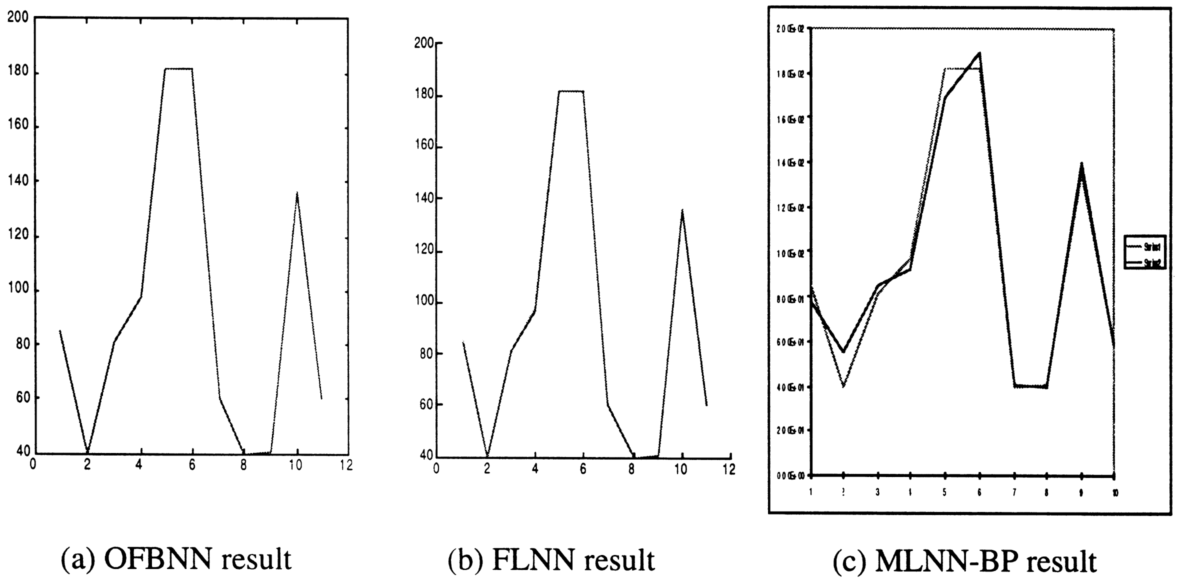

Fig. 3. Predict a feature value of a compound from Table 1, Cluster #2.

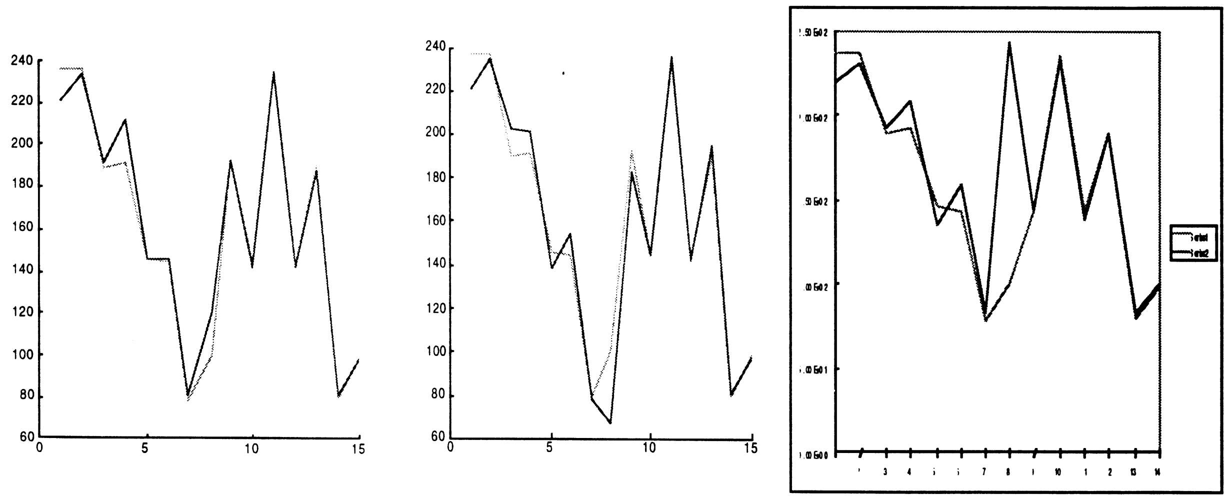

Fig. 4. Predict a feature value of a compound from eight-cluster result.

The OFBNN is again applied to predict feature values for compounds in Table 2 which involve eight clusters. The training and testing results are shown in Fig. 4. Among the compounds in Cluster #3, 14 are used for training, one compound (compound number 29) is used for testing. The feature value number four (weight) is the independent variable and will be predicted by the networks. In this simulation, the desired feature is 100.690 and the predicted value is 114.0455. The elapsed time is 3.0540 s. The FLNN gives the predicted result 66.6705, and it takes 7.6470 s on the same machine. A 5-3-1 MLNN takes 1176 epochs to reach training MSE 0.049, testing MSE is 1441, the predicted value is 44.42. Again, the BP algorithm gives a very bad prediction.

We also learn the function of compounds in Cluster #2 from Table 2, which consists of 12 compounds. Among them, 11 compounds are used for training, another compound (compound #24) is used for testing. In Fig. 5 the training and testing plot is shown. In this simulation, the desired feature is 199.9000 and the predict value is 191.5589. The elapsed time is 2.1760 s. The FLNN gives a predicted result of 360.5574, and it takes 6.9630 s on the same machine. A 5-3-1 MLNN takes 1208 epochs to reach training MSE 0.046, testing MSE is 86.76, the predicted value is 203.78.

A more complicated data set with 357 compounds and 26 dimensions is used to test the effectiveness of the OFBNN. This data set is provided by A. Jackson at Wright Lab and being furnished by S. Thaler for the missing values [19]. In order to predict local property correctly, we use PAM clustering algorithm working on five to 25 clusters. The best clustering result is 15 clusters. We select the largest cluster that has most compounds (52 compounds) for prediction. The feature value density is the independent variable and is predicted by the OFBNN. We train the network using 45 compounds and test on 52 compounds. The training MSE is 0.0296, the testing MSE is 0.0765, the training time is 8.265 s. The OFBNN only uses 33 basis. The result is shown in Fig. 6(a). For the same data set, the FLNN gives training MSE 0.1676, the

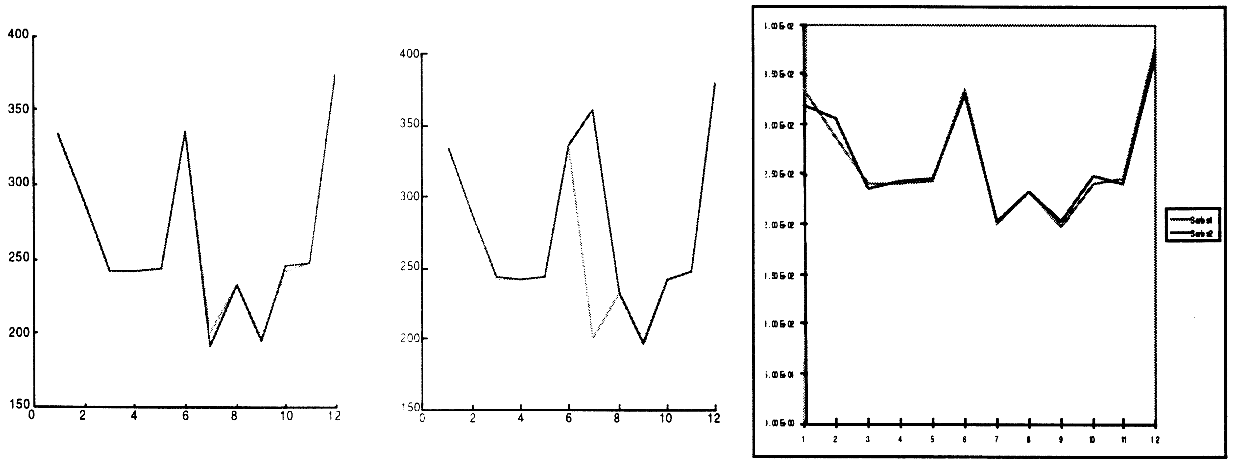

Fig. 5. Predict a feature value of a compound from eight-cluster result.

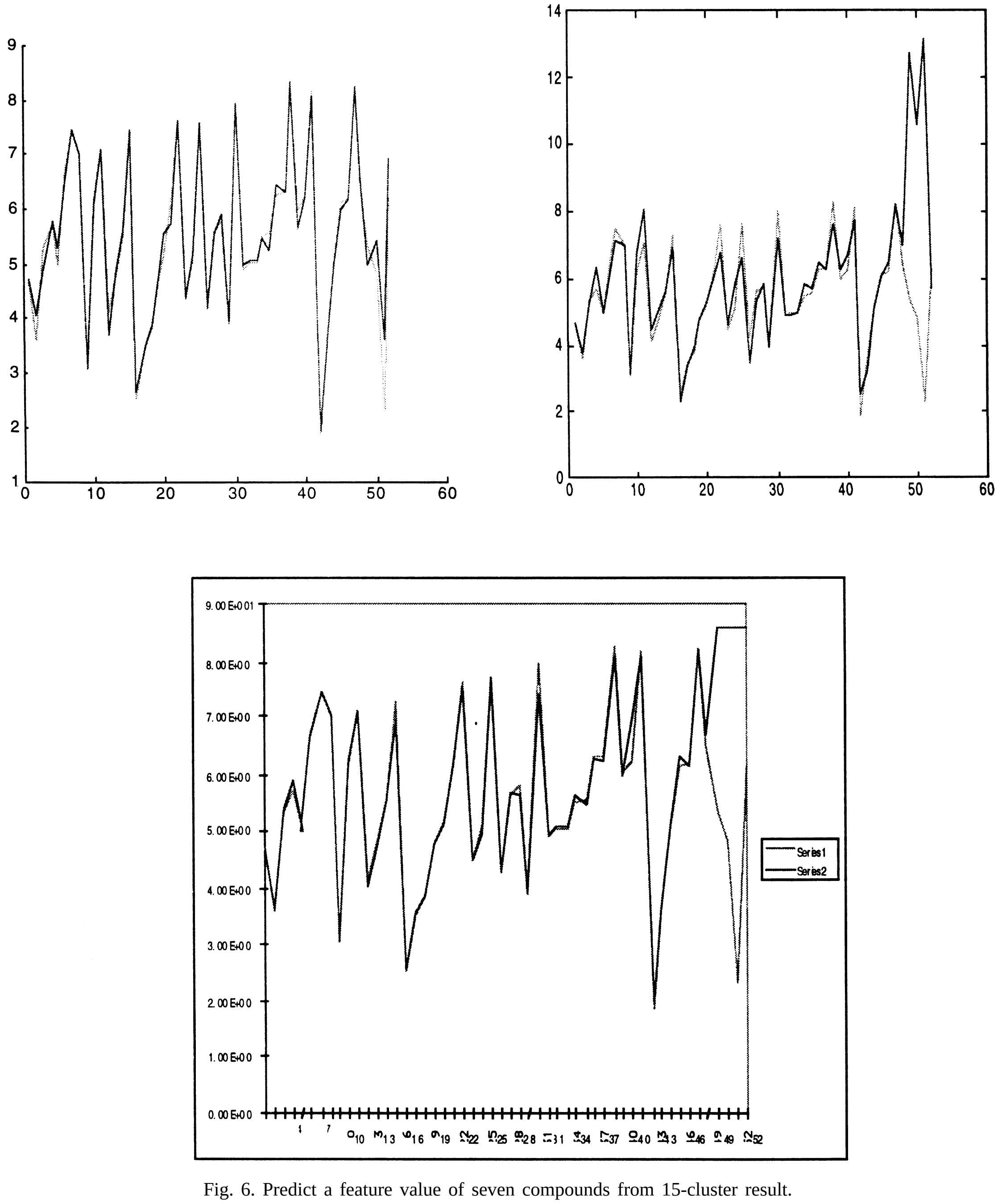

Fig. 6. Predict a feature value of seven compounds from 15-cluster result.

testing MSE, 4.1133, and the training time 96.4950 s. The FLNN result is given in Fig. 6(b). The BP algorithm, result shown in Fig. 6(c), gives us a lower training MSE, 0.0225, running 8963 epochs on a 25-5-1 network, while the testing MSE is 1.3543.

5. Conclusions

We have presented a new neural network functional approximation model, OFBNN, in which, specifically, the model combines regularization and generalized cross validation. The OFBNN not only gives us a more computationally tractable method, but also gives us better generalization performance. Clusters of EO data of semiconductor compounds are formed using an unsupervised learning algorithm. The OFBNN model is used to learn a function relating attributes of compounds which form a cluster based on similar attribute values. We are able to learn the function and predict the feature value of compounds in a cluster. The simulation result of the EO data gives us a very promising results.

Appendix A

Name		Gap	*a*	*c*	at. wt	Radiuslon	Density
1.	InSb	0.23	6.479	6.479	236.55	89	5.777
2.	Te	0.33	4.457	5.939	236.55	82	6.25
3.	InAs	0.36	6.268	6.479	189.79	71	5.72
4.	GeSn	0.3	6.07	6.07	191.28	96	0.0001
5.	$CdGeAs_2$	0.57	5.943	11.217	334.97	71	5.6
6.	GaSb	0.72	6.095	6.095	191.47	89	5.615
7.	SiSn	0.84	5.96	5.96	146.77	96	0.0001
8.	SiGe	0.9	5.54	5.54	100.67	76	0.0001
9.	$AgInSe_2$	1.2	6.099	11.691	286.798	66	5.808
10.	SnO	1.2	5.03	5.03	130.7	77	0.0001
11.	InSe	1.25	4.002	24.946	193.76	66	5.55
12.	InP	1.35	5.868	5.868	145.77	59	4.798
13.	GaAs	1.4	5.653	5.653	144.71	71	5.316
14.	CdTe	1.5	6.488	6.488	240	82	0.0001
15.	$CuInS_2$	1.53	5.489	11.101	242.468	66	4.73
16.	Se	1.7	4.361	4.954	78.96	66	4.819
17.	$CuGaSe_2$	1.7	5.606	11.006	242.468	66	4.73
18.	$ZnSiAs_2$	1.74	5.606	10.88	243.43	71	4.7
19.	$AgGaSe_2$	1.8	5.981	10.865	335.51	66	5.759
20.	CdSe	1.8	4.3	7.01	191.36	66	0.0001
21.	Ag_3SbS_3	1.93	11	8.7	541.552	53	0.0001
22.	Ag_3AsS_3	2	10.8	8.69	494.792	53	5.6
23.	GaSe	2.021	3.747	23.91	148.68	66	5.03
24.	$ZnGeP_2$	2.05	5.463	10.731	199.9	59	4.105
25.	HgS	2.1	4.145	9.496	232.654	53	7.101
26.	GeC	2.1	4.61	4.61	84.6	29	0.0001
27.	$AgAsS_2$	2.14	17.23	15.19	246.988	53	0.0001
28.	b-SiC	2.26	4.359	4.359	40.09	29	3.191
29.	GaP	2.3	5.45	5.45	100.69	59	4.135
30.	ZnTe	2.3	6.101	6.101	192.97	82	5.924
31.	$CuGaS_2$	2.43	5.351	10.47	197.388	53	4.332
32.	CdS	2.485	4.16	6.756	144.464	53	0.0001
33.	GaS	2.5	3.586	15.496	101.784	53	3.86
34.	$AgGaS_2$	2.638	5.751	10.238	241.718	53	4.66
35.	ZnSe	2.7	5.667	5.668	144.33	66	5.318
36.	AgI	2.8	6.473	6.473	234.77	126	6
37.	CuBr	2.91	5.69	5.69	143.449	82	4.72
38.	$CdGeP_2$	2.91	5.74	10.776	246.93	59	4.549
39.	CuI	2.95	6.042	6.042	190.44	96	5.667
40.	$CdGa_2S_4$	3.05	5.568	10.04	80.096	53	3.97
41.	$CdGa_2S_4$	3.05	5.568	10.04	380.096	53	3.97
42.	CuCl	3.17	5.405	5.405	98.993	77	4.137
43.	ZnO	3.3	3.251	5.209	81.369	22	5.651
44.	ZnS	3.9	3.823	6.261	97.434	53	3.536
45.	$LiIO_3$	4	5.481	5.171	181.836	22	4.502
46.	$LiNbO_3$	4	5.148	13.86	147.842	22	4.64
47.	$LiIO_3$	4	5.481	5.171	181.836	22	4.502
48.	NH_2−2CO	5.9	5.58	4.69	60.069	22	0.0001
49.	SiC	6	4.359	4.359	40.09	29	3.191
50.	AlN	6.2	3.11	4.98	40.99	25	3.255
51.	BaB_2O_4	6.3	12.5316	12.7285	222.958	22	0.0001
52.	KH_2PO_4	7	7.45	6.97	136.086	59	0.0001
53.	SiO_2	8.4	4.9134	5.4052	60.078	22	2.65

References

[1] J.O. Rawlings, Applied Regression Analysis, Wadsworth and Brooks/Cole, Pacific Grove, CA, 1988.
[2] A.R. Barron, X. Xiao, Ann. Stat. 19 (1991) 67–82.
[3] L. Breiman, Stacked Regression, Tech. Rep. TR-367, Department of Statistics, University of California, Berkeley, 1992.
[4] D.S. Broomhead, D. Lowe, Complex Sys. 2 (1988) 321–355.
[5] J. Park, I.W. Sandberg, Neural Computation 3(2) (1991) 246–257..
[6] B. Igelnik, Y.H. Pao, IEEE Trans. Neural Networks 6(6) (1995) 1320–1328.
[7] K. Funahashi, Neural Networks 2 (1989) 183–192.
[8] S. Chen, C.F.N. Cowan, P.M. Grant, IEEE Trans. Neural Networks 2(2) (1991) 302–309.

[9] M.J.L. Orr, Neural Computation 7 (1995) 606–623.
[10] A.R. Barron, IEEE Trans. Inform. Theory 39(3) (1993) 930–945.
[11] W.S. McCulloch, W. Pitts, Bull. Math. Biophys. 5 (1943) 115–133.
[12] Y.-H. Pao, Memory based computational intelligence for materials processing and design, Wright Laboratory Technical Report WL-TR-96-4062, Wright-Patterson AFB, OH, 1996, pp. 1–14.
[13] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data, Academic Press, New York, 1994.
[14] M.F. Ashby, Materials Selection in Mechanical Design, Pergamon Press, New York, 1992.
[15] D.O. Hebb, The Organization of Behavior, Wiley, New York, 1949.
[16] T. Kohonen, Biol. Cybern. 43 (1982) 59–69.
[17] G.A. Carpenter, S. Grossberg, Appl. Opt. 26 (1992) 4919–4930.
[18] Neuralyst, Cheshire Engineering Corporation, Parsadena, CA 91107, 1994.
[19] S. Thaler, Network-based creativity machines for materials discovery, Proceedings of the Australasia-Pacific Forum on Intelligent Processing and Manufacturing of Materials, 1997, 14–17, July.