Multi-dimensional Data Construction Method with Its Application to Learning from Small-sample-sets

*Hsiao-Fan Wang, Chun-Jung Huang
Department of Industrial Engineering and Engineering Management,
National Tsing Hua University.
No. 101, Section 2 Kuang Fu Road, Hsinchu, 30013 Taiwan, ROC.
Tel: +886-3-5742654; Fax: +886-3-5722685
*hfwang@ie.nthu.edu.tw

Abstract

Insufficient training data is one of the major problems in neural network learning, because it leads to poor learning performance. In order to enhance an intelligent learning process, it is necessary to exploit the features of the problem from the available information even with limited scale. Due to the shortcomings of the existing methods for data generation; and also in general, a problem is described by multiple attributes, this study has first extended the developed one-dimensional Data Construction Method (DCM) for virtual data generation to multidimensional continuous space as denoted by *m*-DCM. Then, sensitivity analysis and numerical illustration have been carried out. By incorporating *m*-DCM into a supervised neural network learning process, we have shown to overcome the existing unbounded and immeasurable problems and provided a better learning performance in a comparative manner.

Keywords: Small Sample Set, Virtual Data Generation, Data Construction Method, multiple dimensions, Supervised Neural Network Learning

1. Introduction

In the framework of supervised neural network learning, the lack of reference data is very often responsible for poor performances. In fact, it has been widely acknowledged that a neural network model causes non-negligible errors when the amount of training data is limited. To avoid this, searching for more data has been necessary for training. However, due to either time (money) limit in collecting more data, or even impossible to obtain additional data, developing an effective method to fill the information gaps caused by insufficient training data cannot be undervalued.

1.1 Related Works

While it is difficult, even impossible, to gain more training data, there is evidence that generating more resembling data from original data set can make learning tools perform better. Evidence for this comes from two areas. First, a considerable number of studies were shown to demonstrate that the way to use bootstrap samples as a training data set, as so-called bagging (bootstrap aggregating), can significantly reduce the prediction error in the framework of neuro-learning. This idea can be traced back to Breiman (1996) who proposed the concept of bagging, applied bagging to reduce the prediction error of learning algorithms, and suggested decision trees with the most effectiveness. Then, Zhang (1999[a, b]) presented a bagging-based neural network (termed BAGNET) to build non-linear models with the data re-sampled by using bootstrap techniques. The BAGNET was also discussed and applied in

Jia and Culver (2006), and it has been reported that neural network models built from bootstrap re-sampling data are more accurate and robust than those built from original training data. The foregoing judgment may be accepted, but it must be noted that variance and mean square error (MSE) reduction effect of bagging is not necessarily true for all problems (Buja and Stuetzle, 2000), and that sample sizes of at least 12 are required to ensure stable results (Chen et al., 1997).

In addition to bagging-based neural network models, generating "virtual (training) data" based on a function derived from original training data appeared to show the effectiveness in training a neural network model. Such a process was the so-called Virtual Data Generation (VDG). The concept related to VDG can be referred to Abu-Mostafa (1993) who developed a methodology for integrating different kinds of "hints" (otherwise referred to as prior knowledge) into usual learning-from-example procedures. In this methodology, the "hints" can be represented by new examples generated from the existing data set through the applications of transformations that are known to leave the function as the learned invariant. Then, Girosi and Chan (1995) and Niyogi et al. (1998) subsequently developed the idea from "hints" to virtual data, and both reported that the improvement of learning accuracies was significant when using virtual data to train the models of back-propagation neural network (BPNN) and radial basis function networks (RBFN). Besides, Huang (2002) proposed to apply Information Diffusion Theory to VDG, which was further improved by Huang and

Moraga (2004) with a Diffusion-Neural-Network (DNN). This DNN was trained by using the deriving data instead of original training data, and the result showed that the method is effective when the target function has a strong non-linearity or the amount of original training data is less than ten.

Although numerous studies have demonstrated the success of applying bagging and VDG to supervised neural network learning, the amount of virtual data or bootstrap samples required to provide more information for the training system and reach higher learning accuracies remains in an ad-hoc manner. In their recent survey on VDG, Li and Lin (2006) have taken some important steps in the foregoing issues. In order to build up the management knowledge during the early manufacturing stages from a small training data set, Li and Lin (2006) developed a method of Intervalized Kernel Density Estimation (IKDE) to create virtual data for the improvement of scheduling effectiveness. In brief, the method states that if a set of observations, $\{X_1, \ldots, X_n\}$ is given, a function should firstly be estimated as equation (1) by using a kernel function, $K_j(\cdot)$, with its central location, c_j, and defining m different smoothing parameters, h_j:

$$\hat{f}(x) = \sum_{j=1}^{m} \frac{1}{n_j} \frac{1}{h_j} K_j\left(\frac{x - c_j}{h_j}\right), x \in \mathbb{R}, n = \sum_{j=1}^{m} n_j \ . \tag{1}$$

The randomly generated data from the function derived from eq.(1) are the so-called virtual data. Li and Lin (2006) concluded that the virtual data generated through IKDE were able to improve accuracies using BPNN as the learning tool. Nevertheless, having too many virtual

data may result in a steeply opposite effect. They explained this phenomenon by stating that this shortcoming stems from the fact that the virtual data may not be bounded by the domain values of data population (see Figure 1). This unbounded property on the domain values of virtual data may induce noise data in the procedure of VDG; the more are the amount of the virtual data generated, the worse is the accuracy. This induced a question whether creating virtual data bounded by the domain value of the collected data will improve the accuracy and robustness of the results, and this is a primal inspiration to this study.

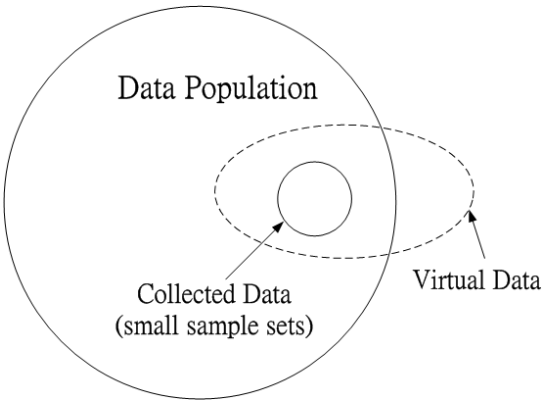

Figure 1 The relationship among the population, collected data and virtual data.
(Li and Lin, 2006)

1.2 Motivation

Based on the pros and cons of VDG, an alternative approach to create virtual data was proposed for one dimensional sample, namely, Data Construction Method (DCM) which has not only shown to have the bounded property, but also shown its estimation ability to predict the severe earthquakes in Taiwan along the spatial distribution with reasonable accuracy (Wang and Huang, 2009). Since we have observed that most events are caused by multiple

factors with multidimensional data description, in this study we shall extend the one-dimensional DCM to multi-dimension as *m*-DCM and demonstrate its learning ability to apply the developed *m*-DCM to the learning process in the supervised NN framework. Herein, "virtual data" and "DCM data" are used interchangeably for the smoothness of the logics.

The rest of this paper is organized as follows. First, the basic concepts of DCM and *m*-DCM are introduced. Then, the algorithm of *m*-DCM, which is explained through an example of classification, is presented in Section 2. A simulation experiment, including the sensitivity analysis of the *m*-DCM algorithm and the comparison between *m*-DCM and IKDE, is presented in Section 3. Finally, the discussion and conclusions are drawn in Section 4.

2. Methodology

Before presenting the algorithm of multi-dimensional Data Construction Method (*m*-DCM), let us first give a review of DCM as a step-stone of developing *m*-DCM. Then, in Sec. 2.2 after the properties of *m*-DCM are investigated, the solution procedure of *m*-DCM is proposed, for which a numerical example is presented in Sec. 2.3. Finally, discussion and summary are given in Sec. 2.4.

2.1 Review of DCM

This section is referred to Wang and Huang (2009). Suppose a sample set with size *n* are collected. To present all multiplicities of the elements, this sample can be described by

$$A_o = \{(a_{o_i}, \lambda_i) \mid a_{o_i} \in \mathbb{R}, \lambda_i \in \mathbb{N}, 1 \le i \le n, \sum \lambda_i = n\} \tag{2}$$

where A_o is a multiset, a_{o_i} is the element value, and λ_i is the corresponding frequency of a_{o_i} in A_o.

Then, a representative mode of A_o is uniquely defined by

$$\text{mode}(A_o) = \{\min_{a_{o_i}}\{\max_{\lambda_i}(a_{o_i}, \lambda_i)\} \mid 1 \le i \le n\}. \tag{3}$$

Based on eq.(3), we can obtain a multiset

$$A = \{(a_i, \lambda_i) \mid a_i \in \mathbb{R}, \lambda_i \in \mathbb{N}, 1 \le i \le n\} \tag{4}$$

where $a_i = a_{o_i} - \text{mode}(A_o)$ and $a_{i+1} > a_i$.

Now, let us consider a multiset $C=\{(1, 1), (c, 1)\}$ with $c>1$, then $\forall t \in \mathbb{N}$ a multiset division of A by C is defined by

$$Z^1 \equiv A(:)C = \{(a_i / 1, \lambda_i), (a_i / c, \lambda_i) \mid a_{o_i} \in \mathbb{R}, \lambda_i \in \mathbb{N}, 1 \le i \le n, \sum \lambda_i = n\}, \text{ and}$$

$$Z^t \equiv A(:)C^t = \{(z_{q^t}, f_{q^t}) \mid z_{q^t} \in [a_1, a_n], f_{q^t} \in \mathbb{N}, 1 \le q \le Q\}, Q \le 2^t * n \tag{5}$$

where t is a variable to indicate the number of divisions and Z^t is the resultant data after doing a specific number of divisions.

When $t=T$, let $z'_{q^T} = z_{q^T} + \text{mode}(A_o)$ and we have

$$Z^{T'} = \{(z'_{q^T}, f_{q^T}) \mid z'_{q^T} \in [a_{o_1}, a_{o_n}], f_{q^T} \in \mathbb{N}, 1 \le q \le Q\} \tag{6}$$

Then, $Z^{T'}$ is a so-called DCM data set, and it has some proven properties as follows:

Property: Measurable DCM Data Size and Conservations of the mode and bounds

$\sum f_{q^T} = 2^T * n$, $\text{mode}(Z^{T'}) = \text{mode}(A_o)$, $\inf(Z^{T'}) = a_{o_1}$, and $\sup(Z^{T'}) = a_{o_n}$. ■

For example, a sample set including eleven elements of 16, 16, 16, 17, 19, 19, 19, 19, 19, 20, and 20. This sample can be described by a multiset of A_o={(16, 3), (17, 1), (19, 5), (20, 2)}. By mode definition in eq.(3), mode(A_o)=19 is assigned for its maximal frequency, and A={(-3, 3), (-2, 1), (0, 5), (1, 2)}can be obtained from eq.(4). When c=2 and t=1 and 2, based on multiset division of eq.(5) the computation of Z^1 and Z^2 is shown as follows:

Z^1=A(:)C={(-3, 3), (-2, 1), (0, 5), (1, 2)} (:) {(1, 1), (2, 1)}

={(-3, 3), (-2, 1), (-1.5, 3), (-1, 1), (0, 10), (0.5, 2), (1, 2)};

Z^2= A(:)C^2= Z^1(:)C={(-3, 3), (-1.5, 3), (-2, 1), (-1, 1), (0, 10), (0.5, 2), (1, 2)} (:){(1, 1), (2, 1)}

={(-3, 3), (-2, 1), (-1.5, 6), (-1, 2), (-0.75, 3), (-0.5, 1), (0, 20), (0.25, 2), (0.5, 4), (1, 2)}.

Let $z'_{q^t} = z_{q^t} + 19$ for t=1 and 2 to replace back to the original axis. We have

$Z^{1'}$ = {(16, 3), (17, 1), (17.5, 3), (18, 1), (19, 10), (19.5, 2), (20, 2)}, and

$Z^{2'}$ = {(16, 3), (17, 1), (17.5, 6), (18, 2), (18.25, 3), (18.5, 1), (19, 20), (19.25, 2), (19.5, 4), (20, 2)}.

The foregoing result shows that $Z^{1'}$ and $Z^{2'}$ comprise 22 (=2^1*11) and 44 (=2^2*11) DCM data, respectively, and that $\text{mode}(Z^{1'}) = \text{mode}(Z^{2'}) = 19$, $\text{inf}(Z^{1'}) = \text{inf}(Z^{2'}) = 16$, and $\sup(Z^{1'}) = \sup(Z^{2'}) = 20$. This obviously links with the mentioned property, say, when t=T the amount of DCM data is 2^T-fold of the amount of the collected sample, and that DCM data will always be bound by the domain values of the collected sample. Therefore, the above definitions enable us to create virtual data through one dimensional sample with the bounded

property. In the next section, we shall extend the developed one-dimensional DCM to multi-dimension (*m*-DCM) for a sample with *m* independent attributes.

2.2 *m*-DCM with Its Solution Procedure

Extending to *m*-dimensional data set, let us first consider a sample of size *n* in *m*-dimensional attribute space by an *n* x *m* matrix as below

$$A_o = \begin{bmatrix} a_{o_1}^1 & \cdots & a_{o_1}^m \\ \vdots & \ddots & \vdots \\ a_{o_n}^1 & \cdots & a_{o_n}^m \end{bmatrix}_{n \text{ x } m} \tag{7}$$

where $n>1$ and $m\geq 1$ are assumed. Then, each datum is shown in a row with *m* attributes. For each attribute *j*, the data can be presented by a (column) vector $A_o^j = \begin{bmatrix} a_{o_1}^{\ j} & \cdots, a_{o_i}^j, \cdots, & a_{o_n}^{\ j} \end{bmatrix}^{\mathrm{T}}$, j=1, 2, …, *m*. Since there are *n* couples of elements in each attribute vector, it is the same as eq.(2) to describe A_o^j by the multiset form as

$$A_o^j \equiv \{(a_{o_i}, \lambda_i) \,|\, a_{o_i} \in \mathbb{R}, \lambda_i \in \mathbb{N}, 1 \le i \le n, \sum \lambda_i = n\}^j,\ j = 1, 2, ..., m \tag{8}$$

where a_{o_i} is a element value and λ_i is the corresponding frequency of a_{o_i} in A_o^j.

For instance, assuming $A_o = \begin{bmatrix} 4 & 5 \\ 1 & 5 \\ 2 & 7 \\ 4 & 7 \end{bmatrix}_{4x2}$ where $A_o^1 = \begin{bmatrix} 4 & 1 & 2 & 4 \end{bmatrix}^{\mathrm{T}}$ and $A_o^2 = \begin{bmatrix} 5 & 5 & 7 & 7 \end{bmatrix}^{\mathrm{T}}$, in terms of multiset presentation, two attribute vectors can be presented by {(4, 2), (1, 1), (2, 1)}^{1} and {(5, 2), (7, 2)}^{2}.

Following the one-dimensional definition, i.e., eq.(3), the mode of each attribute of an *m*-dimensional multiset was defined as below

$$\text{mode}^j = \left\{ \min_{a_{o_i}} \left\{ \max_{\lambda_i} \left\{ (a_{o_i}, \lambda_i) \right\} \right\} \right\}^j . \tag{9}$$

Taking the above example of {(4, 2), (1, 1), (2, 1}^{1} and {(5, 2), (7, 2)}^{2}: because in both attributes, λ=2 is the largest frequency, the corresponding data of 4, 5, and 7 are candidates of the mode. To select a unique mode for each attribute, eq.(9) says that while 4 represents the mode of attribute 1; due to 5<7, 5 should be chosen as the mode of attribute 2.

Based on the mode of each attribute defined in eq.(9), A_o is translated into

$$A = \begin{bmatrix} a^1_{1_o} - \text{mode}^1 & \cdots & a^m_{1_o} - \text{mode}^m \\ \vdots & \ddots & \vdots \\ a^1_{n_o} - \text{mode}^1 & \cdots & a^m_{n_o} - \text{mode}^m \end{bmatrix} = \begin{bmatrix} a^1_1 & \cdots & a^m_1 \\ \vdots & \ddots & \vdots \\ a^1_n & \cdots & a^m_n \end{bmatrix}_{n \text{ x } m} . \tag{10}$$

In each attribute-vector, therefore, eq.(10) ensures that there exists at least a zero value in each column. Using the same example as above, because of mode^1=4 and mode^2=5, A^1_o and A^2_o can be translated into $A^1 = \begin{bmatrix} 0 & -3 & -2 & 0 \end{bmatrix}^{\mathrm{T}}$ and $A^2 = \begin{bmatrix} 0 & 0 & 2 & 2 \end{bmatrix}^{\mathrm{T}}$.

2.2.1 Analysis of *m*-DCM

Before introducing our approach to virtual data generation (VDG), there are some definitions to be provided in the following.

Definition 1 Matrix Multiplication with Kronecker Product (Graham, 1981)

Given an n x m matrix A and an p x r matrix C, their Kronecker product, denoted by $A \otimes C$, is an np x mr matrix with the block structure

$$A \otimes C = \begin{bmatrix} a^1_1 C & \cdots & a^m_1 C \\ \vdots & \ddots & \vdots \\ a^1_n C & \cdots & a^m_n C \end{bmatrix}_{np \text{ x } mr} . \tag{11}$$

Note that this multiplication is not commutative.

Each multiplication operated on two single values results in a product, but the above multiplication on two matrices will result in a new matrix with (*np***mr*) elements. For example, given $A=\begin{bmatrix}1 & 2\\ 0 & -1\end{bmatrix}_{2x2}$, and $C=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}_{2x3}$, the Kronecker product $A\otimes C$ is

$$A\otimes C=\begin{bmatrix}1 & 2 & 3 & 2 & 4 & 6\\ 4 & 5 & 6 & 8 & 10 & 12\\ 0 & 0 & 0 & -1 & -2 & -3\\ 0 & 0 & 0 & -4 & -5 & -6\end{bmatrix}_{4x6}.$$

Therefore, if *A* is an *n* x *m* matrix and *C* is a *p* x 1 matrix, $A\otimes C$ will lead to an *np* x *m* matrix. This statement tells us that the data size will be expanded from *n* to *np* in the same *m*-dimensional attribute space by doing once product-operation. This concludes the following property:

Property 1: Conservation of Dimension

Given *n* data in a *m*-dimensional attribute space as denoted by an *n* x *m* matrix *A* and a multiplier *p* x 1 matrix *C*, Kronecker product of $A\otimes C$ will produce an *np* x *m* matrix with *np* data in the same *m*-dimensional space. ■

This paves a way for VDG in *m*-dimensional space as below:

Lemma 1 Matrix Multiplication to DDG

Given an *n* x *m* matrix *A* and a 2 x 1 matrix *C* with $C=[1\ 1/c]^T$ and $c>1$, the Kronecker product of *A* and *C* is defined by

$$A \otimes C = \begin{bmatrix} a_1^1 & a_1^2 & \cdots & a_1^m \\ c^{-1}a_1^1 & c^{-1}a_1^2 & \cdots & c^{-1}a_1^m \\ \vdots & \vdots & \ddots & \vdots \\ a_n^1 & a_n^2 & \cdots & a_n^m \\ c^{-1}a_n^1 & c^{-1}a_n^2 & \cdots & c^{-1}a_n^m \end{bmatrix}_{2n \text{ x } m} . \quad (12)$$

After doing Kronecker product by t times, it results in $A \otimes C^t$ denoted by Z^t as

$$Z^t \equiv [[[A \otimes C] \otimes C] \otimes ... \otimes C]$$

$$= \begin{bmatrix} z_1^1 & \cdots & z_1^m \\ \vdots & \ddots & \vdots \\ z_N^1 & \cdots & z_N^m \end{bmatrix}_{N \text{ x } m}, N = 2^t * n \quad \text{with } Z^{t,j} = \begin{bmatrix} z_1^j \\ \vdots \\ z_N^j \end{bmatrix}, j = 1, 2, \cdots, m , \quad (13)$$

Then, the final matrix is obtained by translating eq.(13) into original axis as

$$Z^{t'} = \begin{bmatrix} z_1^1 + \text{mode}^1 & \cdots & z_1^m + \text{mode}^m \\ \vdots & \ddots & \vdots \\ z_N^1 + \text{mode}^1 & \cdots & z_N^m + \text{mode}^m \end{bmatrix}_{N \text{ x } m} \text{ with } Z^{t,j'} = \begin{bmatrix} z_1^j + \text{mode}^j \\ \vdots \\ z_N^j + \text{mode}^j \end{bmatrix}, j = 1, 2, \cdots, m . \quad (14)$$

■

Property 2: Conservation of Original Training Data

Let eq.(14) be rewritten into

$$Z^{t'} = \begin{bmatrix} a_1^1 + \text{mode}^1 & a_1^2 + \text{mode}^2 & \cdots & a_1^m + \text{mode}^m \\ \vdots & \vdots & \ddots & \vdots \\ a_n^1 + \text{mode}^1 & a_n^2 + \text{mode}^2 & \cdots & a_n^m + \text{mode}^m \\ c^{-1}a_1^1 + \text{mode}^1 & c^{-1}a_1^2 + \text{mode}^2 & \cdots & c^{-1}a_1^m + \text{mode}^m \\ \vdots & \vdots & \ddots & \vdots \\ c^{-t}a_n^1 + \text{mode}^1 & c^{-t}a_n^2 + \text{mode}^2 & \cdots & c^{-t}a_n^m + \text{mode}^m \end{bmatrix}_{N \text{ x } m}, N = 2^t * n . \quad (15)$$

Then, it is obvious to see $A_o \subseteq Z^{1'} \subseteq Z^{2'} \subseteq Z^{3'} \subseteq \cdots$. ■

Property 2 says that A_o is always a part of $Z^{t'}$. The ratio of DCM data size to the amount of original training data will increase if there are more matrix multiplications with Kronecker product to be conducted. This property also holds in one dimensional sample such as the

example shown in Section 2.1, and it makes the information provided by original training data remain in the resultant DCM data.

Furthermore, let the numbers of columns and rows in a matrix be denoted by column(·) and row(·), respectively, then column(A)=m, row(A)=n, column(C)=1, row(C)=2, column($Z^{t'}$)=m and row($Z^{t'}$)=N. Let $Z^{t'}$ be seen as a data set of size N with m attributes, which possesses the following properties:

Property 3: Measurable Virtual Data Size

Given $A_{n\times m}$ (n data in m-attribute space), and $C=[1\ 1/c]^{\mathrm{T}}$ with $c>1$, after doing Kronecker product by t times, there are 2^t*n DCM data to be generated, that is, row($Z^{t'}$)=(row(C))$^t*$ row(A). ■

Property 3 is consistent with the result of doing multiset division, i.e., eq.(5), which says that if there are two elements in C, when applying t times of the multiplication procedure as eq.(13), 2^t-fold of the original sample size will be generated. Therefore, this property provides a measure of the amount of the obtained DCM data.

Furthermore, with a reference point of mode for each attribute j, each column matrix of $A_{n\times m}$ defined in eq.(10) contains at least one zero element. This contributes to the following property for each Kronecker product:

Property 4: Conservation of Mode

$\forall t \in \mathbb{N}$, mode($Z^{t,j'}$)=mode(A_o^j),$1 \le j \le m$. ■

In addition, with the defined matrix with $C=[1\ 1/c]^T$ for any arbitrary $c>1$, the following property can be derived for each Kronecker product with respect to the original data set A_o:

Property 5: Conservation of the Bounds

Let the minimal and maximal values of the elements in an attribute vector be defined by inf(.) and sup(.) respectively. Then, inf(A_o^j)=inf($Z^{t,j'}$) and sup(A_o^j)=sup($Z^{t,j'}$) for all t. ■

Both Property 4 and Property 5, reserve the properties of one-dimensional DCM under the conditions of $0 \in A^j$ and sup(C)=1 (Wang and Huang, 2008), and says that the domain value of each attribute will always be bounded by the value of the given data no matter how many times matrix multiplication with Kronecker product is conducted. The shortcomings of IKDE, therefore, resulted from the unbounded virtual data will be overcome by this property.

Property 6: Convergence

Let the variance of elements in an attribute vector be defined by var(·). Then,

var(A_o^j) > var($Z^{t,j'}$) > var($Z^{t+1,j'}$) for t=1, 2, 3, …. ■

Property 6 says that if there are more matrix multiplications with Kronecker product to be conducted, the variance of elements in an attribute vector will become smaller. Consequently, this property enables us to verify that when t value increases, the newly DCM data will converge to the point of $(\text{mode}^1, \text{mode}^2, \cdots, \text{mode}^m)$ in the m-dimensional attribute space.

Based on the above definitions and properties, the procedure of VDG is summarized in the following:

2.2.2 The Algorithm of *m*-DCM

Step 0: Set t=1. Input n training patterns and Q testing patterns;

Step 1: Set a 2x1 matrix $C=[1\ \ 1/c]^T$ with an arbitrary $c>1$. Denote n training patterns in m-dimensional attribute space by an n x m matrix A_o, i.e., eq.(7);

Step 2: Find the modes of all attribute vectors in A_o based on eq.(9) and denote them by mode^j, $j = 1 \sim m$;

Step 3: Translate A_o into $A = \begin{bmatrix} a^1_{1_o} - \text{mode}^1 & \cdots & a^m_{1_o} - \text{mode}^m \\ \vdots & \ddots & \vdots \\ a^1_{n_o} - \text{mode}^1 & \cdots & a^m_{n_o} - \text{mode}^m \end{bmatrix} = \begin{bmatrix} a^1_1 & \cdots & a^m_1 \\ \vdots & \ddots & \vdots \\ a^1_n & \cdots & a^m_n \end{bmatrix}_{n \text{ x } m}$;

Step 4: Do $Z^t \equiv A \otimes C^t = \begin{bmatrix} z^1_1 & \cdots & z^m_1 \\ \vdots & \ddots & \vdots \\ z^1_N & \cdots & z^m_N \end{bmatrix}_{N \text{ x } m}$ with $N = 2^t * n$.

Compute $Z^{t'} = \begin{bmatrix} z^1_1 + \text{mode}^1 & \cdots & z^m_1 + \text{mode}^m \\ \vdots & \ddots & \vdots \\ z^1_N + \text{mode}^1 & \cdots & z^m_N + \text{mode}^m \end{bmatrix}_{N \text{ x } m}$.

Step 5: Input $Z^{t'}$ to a supervised neural network; and define $f(t)$ be a measure of the learning performance evaluated by Q testing patterns. If $f(t-1) \le f(t)$, set $t = t+1$ and go to Step 4; otherwise, go to Step 6.

Step 6: Given that $f(t=T) > f(t=T+1)$, set $t = T$ and output $Z^{T'}$ as the final DCM data.

The above algorithm enables us to create DCM data so that the performance of supervised neural network learning can be improved. Note that Figure 2 presents the flowchart of *m*-DCM. To explain the algorithm in detail, an example is presented in the following section.

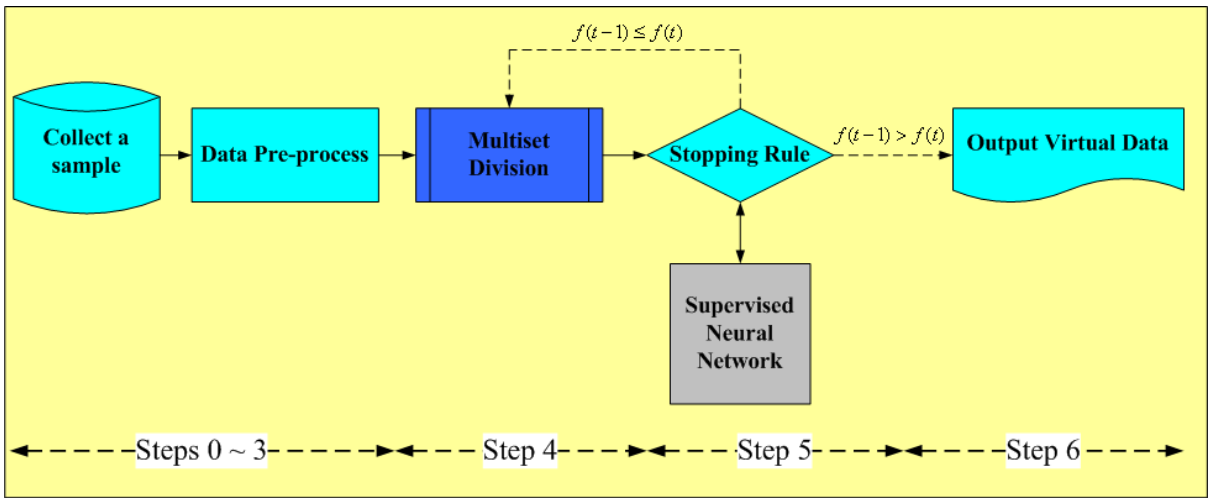

Figure 2 The Flowchart of Multi-dimensional Data Construction Method

2.3 Numerical Illustration of the *m*-DCM Algorithm

Here, an example provided by Li and Lin (2006) was used to explain how to create DCM data for small sample classification problems. The data in Table 1 were the source of *m*-DCM, and those in Table 2 were the testing data. The meaning of each datum was described as follows. The notations of Y=1 (First come, first served), Y=2 (Short Processing Time), and Y=3 (Earlier Due Date) indicate three scheduling rules where each rule is determined through three controlling factors (X_1=buffer size, X_2=arrival rate of parts, and X_3=speed of automobile general motors) on the production line.

Table 1 Twenty Training Data

No.	X_1	X_2	X_3	***Y***	***No.***	X_1	X_2	X_3	***Y***
1	7	20	110	1	11	9	40	60	2
2	8	20	50	1	12	10	20	60	2
3	8	30	70	1	13	10	30	70	2
4	10	30	110	1	14	10	20	70	2
5	10	40	70	1	15	10	30	70	2
6	7	20	60	2	16	11	30	90	2
7	7	30	90	2	17	7	30	60	3
8	8	30	50	2	18	9	20	100	3
9	9	30	50	2	19	10	30	120	3
10	9	30	60	2	20	13	20	90	3

Table 2 Twenty Testing Data (Q)

No.	X_1	X_2	X_3	***Y***	***No.***	X_1	X_2	X_3	***Y***
1	7	20	110	1	11	10	20	70	2
2	8	30	80	1	12	10	20	80	2
3	7	20	60	2	13	11	30	90	2
4	7	20	60	2	14	7	30	50	3
5	8	30	50	2	15	7	30	70	3
6	8	40	90	2	16	7	30	60	3
7	9	20	60	2	17	7	20	70	3
8	10	30	50	2	18	7	20	70	3
9	10	30	60	2	19	9	20	100	3
10	10	20	60	2	20	9	20	100	3

A BPNN with 3-4-1 structure (see Figure 3) was suggested by Li and Lin (2006) for training of which a delta learning rule was adopted in the hidden layer with a sigmoid function of $g(x)=1/[1+\exp(-x)]$. Besides, learning rate=0.3 and momentum=0.8 were set in the network; and $f(t)=q/Q$ was defined to measure the learning accuracy when Q is the amount of total testing data and q are the accurately classified data.

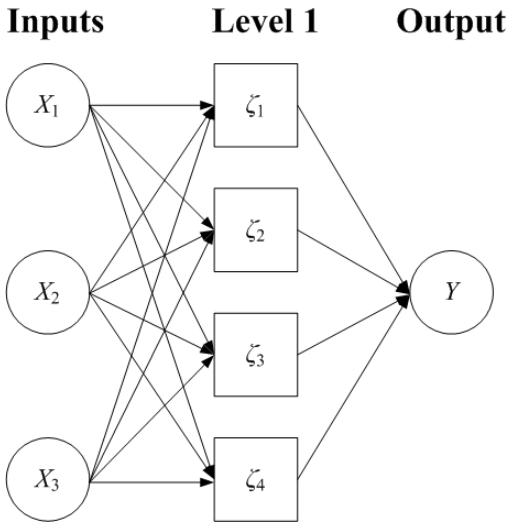

Figure 3 A 3-4-1 Neural Network Structure

Using the data sets above, the algorithm of *m*-DCM is described step by step as follows:

Step 1. Due to Y=1, 2, and 3, the 20 data in Table 1 were divided into:

$$A_{o,Y=1}=\begin{bmatrix}8&20&50\\10&30&110\\7&20&110\\8&30&70\\10&40&70\end{bmatrix},\ A_{o,Y=2}=\begin{bmatrix}9&30&50\\9&30&60\\8&30&50\\10&20&60\\10&30&70\\7&20&60\\9&40&60\\11&30&90\\10&20&70\\10&30&70\\7&30&90\end{bmatrix},\ A_{o,Y=3}=\begin{bmatrix}7&30&60\\13&20&90\\10&30&120\\9&20&100\end{bmatrix}.$$

Then, set matrix C=[1 1/5]^T with c=5.

Step 2. The modes of three attribute vectors, belonging Y=1, 2, and 3, were determined as {8, 20, 70}, {10, 30, 60}, and {7, 20, 60}, respectively.

Step 3. Based on the foregoing modes, $A_{o,Y}$ was translated into

$$A_{Y=1}=\begin{bmatrix}0&0&-20\\2&10&40\\-1&0&40\\0&10&0\\2&20&0\end{bmatrix},\ A_{Y=2}=\begin{bmatrix}-1&0&-10\\-1&0&0\\-2&0&-10\\0&-10&0\\0&0&10\\-3&-10&0\\-1&10&0\\1&0&30\\0&-10&10\\0&0&10\\-3&0&30\end{bmatrix},\ A_{Y=3}=\begin{bmatrix}0&10&0\\6&0&30\\3&10&60\\2&0&40\end{bmatrix}.$$

Step 4. Beginning with t=1, after doing $Z_Y^t \equiv A_Y \otimes C^t$ for Y=1, 2, and 3, the components of Z_Y^1 were listed as below:

$$Z^1_{Y=1} = \begin{bmatrix} 0 & 0 & -20 \\ 2 & 10 & 40 \\ -1 & 0 & 40 \\ 0 & 10 & 0 \\ 2 & 20 & 0 \\ 0 & 0 & -4 \\ 0.4 & 2 & 8 \\ -0.2 & 0 & 8 \\ 0 & 2 & 0 \\ 0.4 & 4 & 0 \end{bmatrix}, Z^1_{Y=2} = \begin{bmatrix} -1 & 0 & -10 \\ -1 & 0 & 0 \\ -2 & 0 & -10 \\ 0 & -10 & 0 \\ 0 & 0 & 10 \\ -3 & -10 & 0 \\ -1 & 10 & 0 \\ 1 & 0 & 30 \\ 0 & -10 & 10 \\ 0 & 0 & 10 \\ -3 & 0 & 30 \\ -0.2 & 0 & -2 \\ -0.2 & 0 & 0 \\ -0.4 & 0 & -2 \\ 0 & -2 & 0 \\ 0 & 0 & 2 \\ -0.6 & -2 & 0 \\ -0.2 & 2 & 0 \\ 0.2 & 0 & 6 \\ 0 & -2 & 2 \\ 0 & 0 & 2 \\ -0.6 & 0 & 6 \end{bmatrix}, Z^1_{Y=3} = \begin{bmatrix} 0 & 10 & 0 \\ 6 & 0 & 30 \\ 3 & 10 & 60 \\ 2 & 0 & 40 \\ 0 & 2 & 0 \\ 1.2 & 0 & 6 \\ 0.6 & 2 & 12 \\ 0.4 & 0 & 8 \end{bmatrix}.$$

By translated Z^1_Y back to the original axis by adding the values of the modes of {8, 20, 70}, {10, 30, 60}, and {7, 20, 60}, $Z^{1'}_Y$ for Y=1, 2, and 3 are obtained as below:

$$Z_{Y=1}^{1'}=\begin{bmatrix}8&20&50\\10&30&110\\7&20&110\\8&30&70\\10&40&70\\8&20&66\\8.4&22&78\\7.8&20&78\\8&22&70\\8.4&24&70\end{bmatrix}, Z_{Y=2}^{1'}=\begin{bmatrix}9&30&50\\9&30&60\\8&30&50\\10&20&60\\10&30&70\\7&20&60\\9&40&60\\11&30&90\\10&20&70\\10&30&70\\7&30&90\\9.8&30&58\\9.8&30&60\\9.6&30&58\\10&28&60\\10&30&62\\9.4&28&60\\9.8&32&60\\10.2&30&66\\10&28&62\\10&30&62\\9.4&30&66\end{bmatrix}, Z_{Y=3}^{1'}=\begin{bmatrix}7&30&60\\13&20&90\\10&30&120\\9&20&100\\7&22&60\\8.2&20&66\\7.6&22&72\\7.4&20&68\end{bmatrix}.$$

That is, $Z_Y^{1'}$ has 40 DCM data as 10 in $Z_{Y=1}^{1'}$, 22 in $Z_{Y=2}^{1'}$, and 8 in $Z_{Y=3}^{1'}$.

Step 5. Using the 20 real data ($A_{o,Y}$) and the 40 DCM data ($Z_Y^{1'}$) to train the 3-4-1 BPNN, respectively, we obtained $f(0)=0.53$ and $f(1)=0.78$ based on the definition of the accuracy. Then, we computed Z_Y^2 to obtain the 80 DCM data ($Z_Y^{2'}$). Due to $f(2)=0.91>f(1)=0.78$, set $t=2+1$ and computed Z_Y^3 to obtain the 160 DCM data ($Z_Y^{3'}$). Because $f(3)=0.88<f(2)=0.91$, the procedure of $A_Y \otimes C^t$ terminated at $t=3$ as shown in Table 3, and go to Step 6.

Table 3 Performance of Using $A_{o,Y}$, $Z_Y^{1'}$, $Z_Y^{2'}$, and $Z_Y^{3'}$ to Train the Network

Training Data	$A_{o,Y}$	$Z_Y^{1'}$	$Z_Y^{2'}$	$Z_Y^{3'}$
Training Data Size	20	40	80	160
Accuracy	0.53	0.78	0.91	0.88

Step 6. Output $Z_Y^{2'}$ as the final DCM data.

As the example given above, it clearly explained the *m*-DCM algorithm. Comparing the 20 real data ($A_{o,Y}$) and the 80 DCM data ($Z_Y^{2'}$), the 6 properties stated in Section 2.2.1 can be verified as below:

(1) Property 1: For all *Y*, $\text{column}(A_{o,Y}) = 3 = \text{column}(Z_Y^{2'})$.

(2) Property 2: For all *Y*, $A_{o,Y} \subseteq Z_Y^{2'}$.

(3) Property 3: For all *Y*, $\text{row}(Z_Y^{2'}) = 80 = 4*20 = 2^2*\text{row}(A_{o,Y})$.

(4) Property 4: For all *Y*, $\text{mode}(A_{o,Y}^j) = \text{mode}(Z_Y^{2,j'}),\ j = 1 \sim 3$, which was supported by $\text{mode}(A_{o,1}^j) = \text{mode}(Z_1^{2,j'}) = \{8,\ 20,\ 70\}$ for *Y*=1, $\text{mode}(A_{o,1}^j) = \text{mode}(Z_1^{2,j'}) = \{10,\ 30,\ 60\}$ for *Y*=2, and $\text{mode}(A_{o,1}^j) = \text{mode}(Z_1^{2,j'}) = \{7,\ 20,\ 60\}$ for *Y*=3.

(5) Property 5: For all *Y*, $\inf(A_{o,Y}^j) = \inf(Z_Y^{2,j'})$ and $\sup(A_{o,Y}^j) = \sup(Z_Y^{2,j'}),\ j = 1 \sim 3$, which was supported by $\inf(A_{o,1}^j) = \inf(Z_1^{2,j'}) = \{7, 20, 50\}$ and $\sup(A_{o,1}^j) = \sup(Z_1^{2,j'}) = \{10, 40, 110\}$ for *Y*=1, $\inf(A_{o,2}^j) = \inf(Z_2^{2,j'}) = \{7, 20, 50\}$ and $\sup(A_{o,2}^j) = \sup(Z_2^{2,j'}) = \{10, 40, 90\}$ for *Y*=2, and $\inf(A_{o,3}^j) = \inf(Z_3^{2,j'}) = \{7, 20, 60\}$ and $\sup(A_{o,3}^j) = \sup(Z_3^{2,j'}) = \{13, 30, 120\}$ for *Y*=3.

(6) Property 6: For all *Y*, $\text{var}(A_{o,Y}^j) < \text{var}(Z_Y^{2,j'}),\ j = 1 \sim 3$, which was supported by $\text{var}(A_{o,1}^j) = \{1.8, 70, 720\}$ and $\text{var}(Z_1^{2,j'}) = \{0.2, 14.2, 96.9\}$, $\text{var}(A_{o,2}^j) = \{1.7, 36.4, 185.5\}$ and $\text{var}(Z_2^{2,j'}) = \{0.3, 5, 27.8\}$, and $\text{var}(A_{o,3}^j) = \{6.3, 33.3, 625\}$ and $\text{var}(Z_3^{2,j'}) = \{1.4, 6.1, 170.5\}$.

As it has been demonstrated in Section 2.2.1, the DCM data shall keep these properties all the time. Moreover, it is noteworthy to state that the improvement in accuracies reached 72%, i.e., ((0.91-0.53)/0.53) when the 80 DCM data ($Z_Y^{2'}$) were used to train the network instead of the

20 real data ($A_{o,Y}$). To explain the foregoing success of applying the obtained DCM data to train the 3-4-1 BPNN, $A_{o,Y}$, $Z_Y^{1'}$, $Z_Y^{2'}$, and $Z_Y^{3'}$ were further discussed in the following section.

2.4 Discussion and Summary

This section discusses why the algorithm of *m*-DCM is able to generate useful virtual data for effectively training a BPNN. For illustration, we then drew $A_{o,Y}$, $Z_Y^{1'}$, $Z_Y^{2'}$, and $Z_Y^{3'}$ on the 3D space as shown in Figure 4.

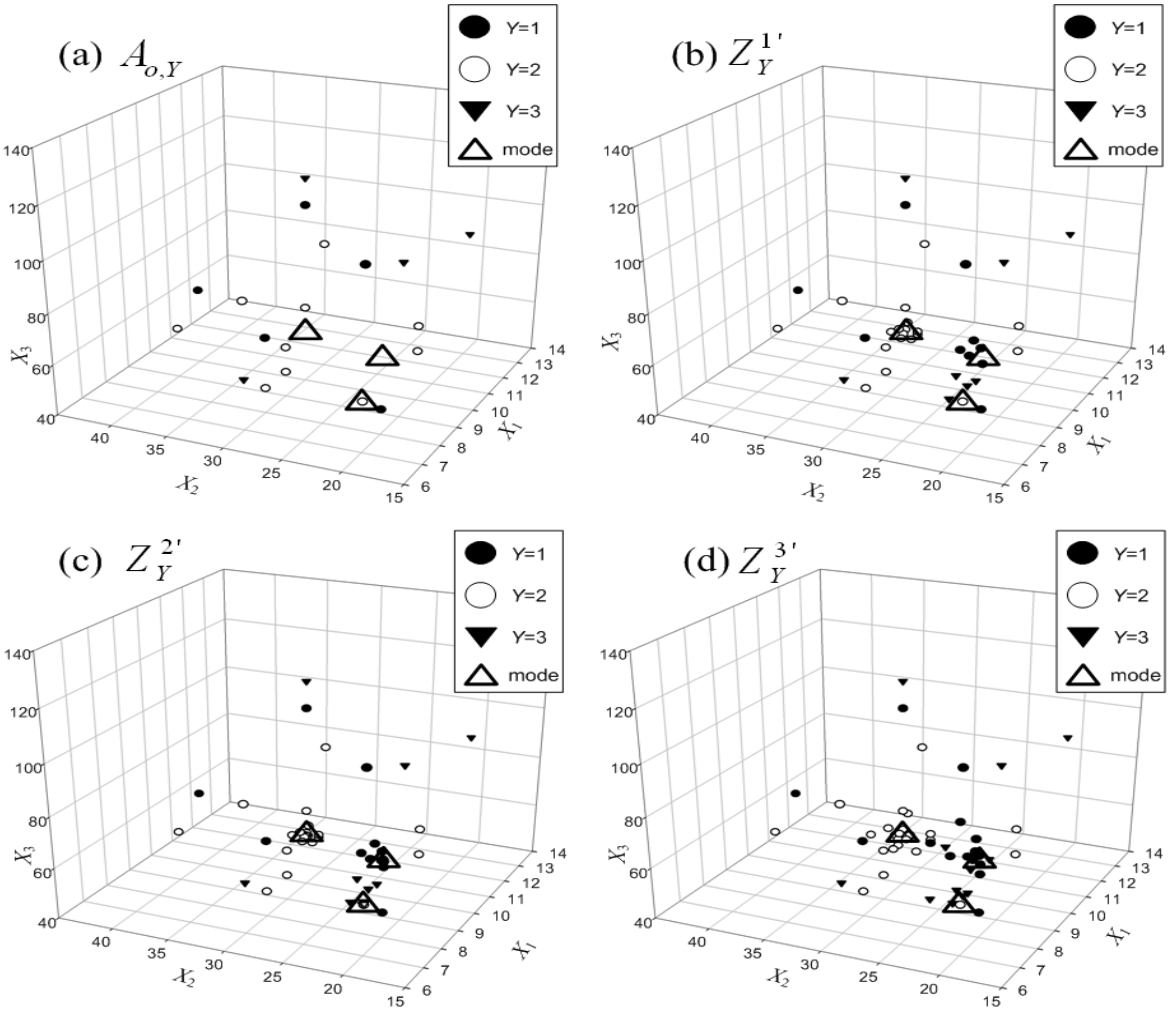

Figure 4 The Patterns of $A_{o,Y}$, $Z_Y^{1'}$, $Z_Y^{2'}$, and $Z_Y^{3'}$

From a series of four diagrams above, the similarity between real and DCM data can be

observed. First, let us mark three classes of $A_{o,Y}$, $Z_Y^{1'}$, $Z_Y^{2'}$, and $Z_Y^{3'}$ by different symbols (● for *Y*=1, ○ for *Y*=2, and ▼ for *Y*=3). Second, the universal discourses of $A_{o,Y}$, $Z_Y^{1'}$, $Z_Y^{2'}$, and $Z_Y^{3'}$ are the same as each other. Third, the newly generated DCM data get closer towards three points of (8, 20, 70), (10, 30, 60), and (7, 20, 60), indicated by △ which are the modes determined at Step 2 of the *m*-DCM algorithm. These phenomena obviously link with Property 5 and 6 previously mentioned. When we compare Figure 4-c and 4-d, we may notice that excessive data being generated as in $Z_Y^{3'}$ will cause more over-laps which seemed to result in a worse learning performance as $f(2)=0.91>f(3)=0.88$. However, by comparing Figure 3-a to the rest three diagrams of 4-b, 4-c, and 4-d, it can be observed that the patterns of 3 classes revealed while more DCM data were generated from our method. The observation is consistent with the results summarized in Table 3.

From this illustrative example and analysis, we may conclude that the proposed *m*-DCM is not only able to generate additional data for training; but also the obtained DCM data are able to reveal the patterns accurately and efficiently in supervised NN trainings.

To give an insight into practical applications of *m*-DCM, sensitivity analysis is carried out in the following section and the comparison with IKDE proposed by Li and Lin (2006) is also provided.

3. Sensitivity Analysis

By integrating Tables 1 and 2 into Table 4 below, a data set including 40 data was used for sensitivity analyses where each datum is described by X_1, X_2, X_3, and Y.

Table 4 The Source Data for the Sensitivity Analyses

No.	X_1	X_2	X_3	Y	***No.***	X_1	X_2	X_3	Y
1	9	30	50	2	21	9	20	100	3
2	8	20	50	1	22	10	30	110	1
3	9	30	60	2	23	7	20	110	1
4	8	30	50	2	24	7	20	60	2
5	10	20	60	2	25	8	30	70	1
6	10	30	50	2	26	8	30	80	1
7	10	30	60	2	27	9	40	60	2
8	10	30	70	2	28	13	20	90	3
9	8	30	50	2	29	11	30	90	2
10	7	20	60	2	30	10	40	70	1
11	7	20	60	2	31	10	30	120	3
12	7	30	50	3	32	9	20	100	3
13	7	30	60	3	33	7	20	110	1
14	7	30	70	3	34	10	20	70	2
15	9	20	60	2	35	10	30	70	2
16	7	30	60	3	36	10	20	80	2
17	7	20	70	3	37	11	30	90	2
18	7	20	70	3	38	8	40	90	2
19	10	20	60	2	39	7	30	90	2
20	10	20	70	2	40	9	20	100	3

3.1 Sensitivity Analyses on *c* and *t* Values

The 40 data above were randomly split into two groups: n training data and (40-n) testing data. The input of m-DCM was the n training data and the examining target was the corresponding (40-n) testing data. To represent the situation of few training data, two circumstances of n=10 and n=20 were under consideration. For simulation purposes, 10 trials were made at each circumstance. By setting c=2~10 for each of t=1~4, changes of accuracies with respect to n=10 and n=20 were shown in Table 5 and Table 6.

Table 5 Results of Sensitivity Analyses at the Circumstance of n=10

c		2	3	4	5	6	7	8	9	10
Trial no.	t	Accuracy								
1	1	0.59	0.55	0.69	0.73	0.73	0.64	0.64	0.51	0.51
	2	0.55	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47
	3	0.49	0.40	0.54	0.54	0.54	0.54	0.60	0.60	0.60
	4	0.40	0.40	0.54	0.54	0.54	0.54	0.54	0.54	0.54
2	1	0.55	0.51	0.51	0.51	0.55	0.55	0.55	0.55	0.51
	2	0.47	0.47	0.51	0.47	0.51	0.51	0.51	0.51	0.47
	3	0.47	0.51	0.47	0.47	0.47	0.51	0.51	0.55	0.47
	4	0.51	0.51	0.47	0.47	0.43	0.43	0.43	0.43	0.47
3	1	0.36	0.32	0.35	0.35	0.35	0.35	0.35	0.35	0.35
	2	0.39	0.44	0.41	0.41	0.41	0.41	0.41	0.41	0.41
	3	0.44	0.45	0.44	0.44	0.40	0.40	0.40	0.40	0.44
	4	0.37	0.37	0.40	0.37	0.37	0.37	0.37	0.37	0.37
4	1	0.51	0.59	0.45	0.45	0.50	0.50	0.54	0.54	0.54
	2	0.51	0.48	0.48	0.48	0.48	0.40	0.44	0.44	0.44
	3	0.58	0.49	0.40	0.36	0.36	0.40	0.34	0.34	0.43
	4	0.39	0.56	0.52	0.56	0.56	0.56	0.56	0.56	0.56
5	1	0.54	0.60	0.58	0.54	0.46	0.46	0.46	0.46	0.42
	2	0.38	0.54	0.50	0.48	0.44	0.44	0.44	0.44	0.44
	3	0.38	0.52	0.50	0.52	0.44	0.44	0.44	0.44	0.44
	4	0.44	0.46	0.46	0.46	0.46	0.46	0.48	0.48	0.48
6	1	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
	2	0.39	0.53	0.53	0.28	0.59	0.59	0.59	0.59	0.59
	3	0.59	0.54	0.52	0.52	0.52	0.52	0.54	0.54	0.54
	4	0.54	0.54	0.54	0.56	0.56	0.56	0.56	0.56	0.56
7	1	0.44	0.44	0.51	0.46	0.46	0.53	0.53	0.53	0.51
	2	0.40	0.37	0.37	0.37	0.37	0.37	0.37	0.37	0.37
	3	0.37	0.41	0.37	0.37	0.37	0.37	0.37	0.37	0.37
	4	0.44	0.41	0.41	0.41	0.41	0.41	0.41	0.41	0.41
8	1	0.48	0.54	0.52	0.52	0.44	0.44	0.44	0.44	0.44
	2	0.58	0.54	0.40	0.38	0.38	0.42	0.42	0.42	0.42
	3	0.51	0.44	0.42	0.42	0.42	0.42	0.42	0.42	0.42
	4	0.49	0.49	0.49	0.49	0.49	0.49	0.49	0.49	0.49
9	1	0.49	0.49	0.49	0.49	0.49	0.49	0.49	0.49	0.49
	2	0.49	0.49	0.57	0.57	0.53	0.53	0.53	0.53	0.53
	3	0.57	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.53
	4	0.53	0.53	0.53	0.57	0.57	0.57	0.57	0.57	0.57
10	1	0.39	0.39	0.34	0.34	0.34	0.32	0.32	0.32	0.34
	2	0.36	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40
	3	0.51	0.43	0.31	0.31	0.31	0.31	0.31	0.31	0.31
	4	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43

Table 6 Results of Sensitivity Analyses at the Circumstance of n=20

c		2	3	4	5	6	7	8	9	10
Trial no.	t	Accuracy								
1	1	0.57	0.62	0.78	0.78	0.62	0.62	0.62	0.62	0.62
	2	0.78	0.91	0.91	0.91	0.91	0.91	0.91	0.94	0.94
	3	0.88	0.91	0.94	0.88	0.85	0.85	0.80	0.80	0.83
	4	0.88	0.78	0.80	0.56	0.56	0.56	0.56	0.56	0.56
2	1	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35
	2	0.31	0.36	0.36	0.47	0.47	0.31	0.31	0.31	0.31
	3	0.31	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36
	4	0.31	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36
3	1	0.63	0.7	0.50	0.50	0.50	0.47	0.47	0.47	0.47
	2	0.60	0.47	0.50	0.50	0.37	0.37	0.37	0.37	0.37
	3	0.40	0.37	0.43	0.43	0.40	0.43	0.43	0.43	0.43
	4	0.50	0.37	0.40	0.40	0.40	0.40	0.40	0.40	0.40
4	1	0.61	0.61	0.61	0.69	0.69	0.69	0.69	0.69	0.69
	2	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58
	3	0.58	0.58	0.61	0.58	0.58	0.58	0.58	0.56	0.56
	4	0.58	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.58
5	1	0.55	0.42	0.42	0.42	0.42	0.42	0.42	0.42	0.42
	2	0.47	0.47	0.47	0.63	0.47	0.36	0.44	0.44	0.38
	3	0.47	0.46	0.49	0.49	0.49	0.49	0.49	0.49	0.49
	4	0.46	0.49	0.46	0.46	0.46	0.38	0.38	0.38	0.38
6	1	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
	2	0.47	0.53	0.53	0.60	0.60	0.60	0.60	0.60	0.60
	3	0.50	0.60	0.57	0.57	0.57	0.57	0.57	0.57	0.57
	4	0.47	0.43	0.43	0.43	0.43	0.43	0.50	0.50	0.5
7	1	0.57	0.57	0.57	0.64	0.64	0.64	0.64	0.64	0.64
	2	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.61	0.54
	3	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54
	4	0.54	0.47	0.47	0.47	0.54	0.47	0.62	0.62	0.62
8	1	0.58	0.68	0.68	0.68	0.68	0.68	0.63	0.63	0.63
	2	0.65	0.68	0.68	0.65	0.65	0.65	0.65	0.70	0.70
	3	0.58	0.58	0.63	0.63	0.63	0.63	0.63	0.63	0.63
	4	0.58	0.63	0.63	0.63	0.63	0.63	0.63	0.63	0.63
9	1	0.44	0.41	0.37	0.54	0.54	0.54	0.54	0.54	0.57
	2	0.37	0.37	0.31	0.31	0.31	0.31	0.31	0.31	0.31
	3	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.31
	4	0.31	0.27	0.24	0.24	0.24	0.24	0.24	0.24	0.24
10	1	0.58	0.61	0.55	0.55	0.55	0.55	0.52	0.52	0.52
	2	0.58	0.58	0.52	0.52	0.52	0.52	0.52	0.52	0.52
	3	0.46	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36
	4	0.36	0.36	0.29	0.29	0.29	0.29	0.29	0.29	0.29

3.1.2 Sensitivity Analysis on *c* Values

Since *c* value is a given parameter in the algorithm of *m*-DCM, it is essential to examine its impact on the learning performance.

Taking the trial no.1 of *n*=20 for example, 36 values of accuracy were listed. Figure 5 shows that the trends of the 9 plots with respect to *c*=2~10 are similar to each other. This was supported by taking Kruskal-Wallis *H* tests with *P*-value=0.962, degree of freedom=8, the statistic=2.485, and the confidence level (α) =0.05.

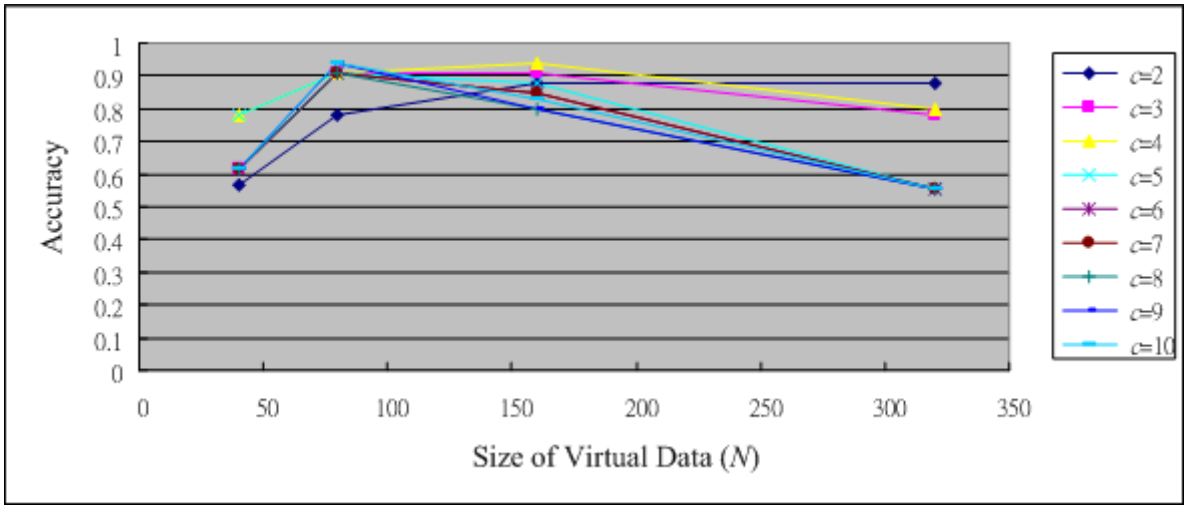

Figure 5 The Relationship between *c* Value and the Accuracy at the Trial no.1 of *n*=20

In the same way, the statistical results of all trials based on Kruskal-Wallis *H* tests were shown in Table 7. Since out of the 20 trials there is only one, i.e., the trial no.5 of *n*=10, to reveal statistical significance, this confirms the robustness of *c* value.

Table 7 Statistical Tests to the Difference in Accuracies among c=2~10

The Circumstance of n=10					
Trial no.	1	2	3	4	5*
Kruskal-Wallis H tests	4.354	3.413	1.161	3.006	16.560
P-value	0.824	0.906	0.997	0.934	0.035
Trial no.	6	7	8	9	10
Kruskal-Wallis H tests	4.312	0.381	9.812	2.420	3.676
P-value	0.828	1.000	0.278	0.965	0.885
The Circumstance of n=20					
Trial no.	1	2	3	4	5
Kruskal-Wallis H tests	2.485	11.405	7.194	0.737	7.754
P-value	0.962	0.180	0.516	0.999	0.458
Trial no.	6	7	8	9	10
Kruskal-Wallis H tests	4.102	6.254	5.960	1.181	3.374
P-value	0.848	0.619	0.652	0.997	0.909

Due to nine different values of c, i.e., c=2~10, the degree of freedom of Kruskal-Wallis H tests is 8 in each trial. Based on Kruskal-Wallis H tests with α=0.05, the sign of * indicates that there is a significant difference in accuracies among c=2~10.

3.1.2 Sensitivity Analysis on Value *t*

As the evidence previously mentioned, there may be an over-lap effect while we attempt to create more DCM data. This cause-effect analysis will be formally carried out to confirm such relations.

Taking the trial no.1 of n=20 for example again, while the learning performance after t=1 is considered the best, Figure 6 shows that $f(2)$ and $f(3)$ dominate $f(1)$ and $f(4)$ respectively. The observation was tested by using Kruskal-Wallis H tests with α=0.05. The results show that there is a significant difference among $f(1)$, $f(2)$, $f(3)$, and $f(4)$ with P-value<0.001. Based on Mann-Whitney U tests with α=0.05, it further reveals $f(2)>f(1)$ with P-value<0.001, $f(2)>f(3)$ with P-value=0.037, and $f(3)>f(4)$ with P-value=0.001.

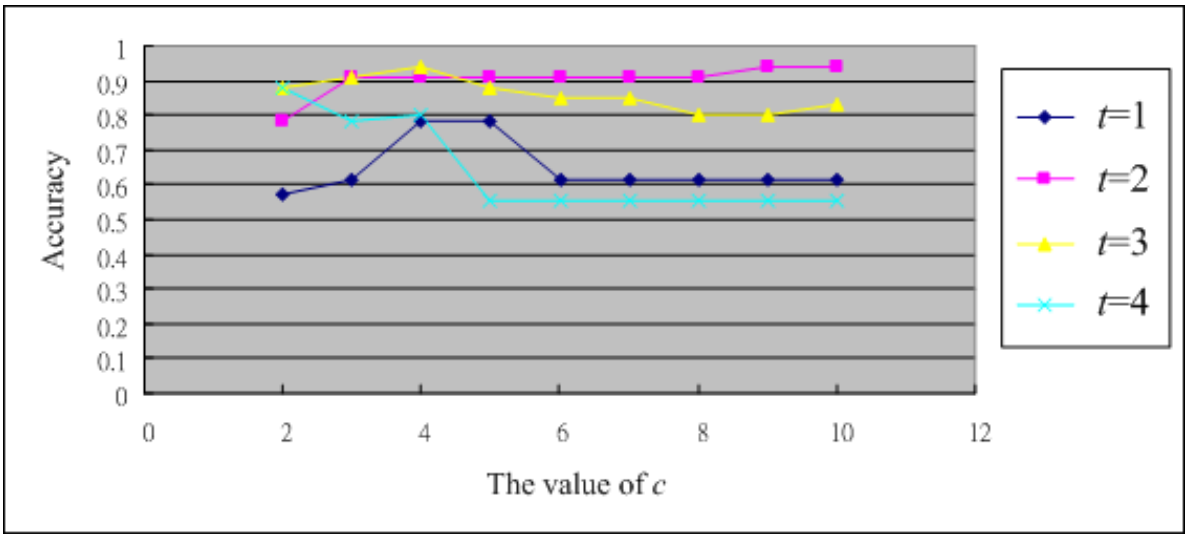

Figure 6 The Relationship between t Value and Accuracies at the Trial no.1 of n=20

Table 8 Statistical Tests to the Difference in Accuracies among t=1~4

The Circumstance of n=10					
Trial no.	$1^{*,+}$	$2^{*,+}$	$3^{*,+}$	$4^{*,-}$	5
Kruskal-Wallis H tests	15.065	16.992	29.647	15.712	4.895
P-value	0.002	0.001	<0.001	0.001	0.180
Trial no.	$6^{*,+}$	$7^{*,+}$	$8^{*,-}$	$9^{*,-}$	$10^{*,-}$
Kruskal-Wallis H tests	16.448	31.760	12.857	23.303	21.361
P-value	0.001	<0.001	0.005	<0.001	<0.001
The Circumstance of n=20					
Trial no.	$1^{*,+}$	$2^{*,-}$	$3^{*,+}$	$4^{*,+}$	$5^{*,+}$
Kruskal-Wallis H tests	24.742	8.768	13.147	27.427	12.681
P-value	<0.001	0.033	0.004	<0.001	0.005
Trial no.	$6^{*,+}$	$7^{*,+}$	$8^{*,+}$	$9^{*,+}$	$10^{*,+}$
Kruskal-Wallis H tests	24.296	19.050	19.681	31.259	30.400
P-value	<0.001	<0.001	<0.001	<0.001	<0.001

Due to t=1~4, the degree of freedom of Kruskal-Wallis H tests is 3 in each trial. Based on Kruskal-Wallis H tests with α=0.05, the sign of * indicates that there is a significant difference in accuracies among t=1~4. Based on Mann-Whitney U tests with α=0.05, the sign of + indicates that $f(4)$ is the worst level of accuracy, and the sign of - indicates that $f(4)$ is the best level of accuracy.

Similarly, Kruskal-Wallis H tests with α=0.05 were used to examine the impact of t value to accuracies among the rest of the trials at the circumstances of n=10 and n=20. The results

of statistical tests were shown in Table 8, which corroborates a great impact of t value to accuracies. On one hand, most of the statistical results of the 20 trials show that there is a significant different among $f(t)$ with t=1~4 based on Kruskal-Wallis H tests with α=0.05. Since virtual data size (N) depends on the number of multiplications (t) with the rule of $N=n*2^t$, the statistical results also indicate that when using m-DCM virtual data size has a great effect on accuracies. On the other hand, further examinations based on Mann-Whitney U tests with α=0.05 show that $f(4)$ is superior to the others among $f(t)$ with t=1~4 in five trials of n=10 while $f(4)$ is inferior to the others among $f(t)$ with t=1~4 in nine trials of n=20. Inferring from the above, we may need a larger t value to pursue a higher level of accuracy when the size of original training data is really small. On the other hand, further examinations based on Mann-Whitney U tests with α=0.05 reveal that one of $f(1)$, $f(2)$, and $f(3)$ is superior to $f(4)$ in the case of n=20.

Based on the sensitivity analysis above, we confirmed that factor c is robotic, yet t value is sensitive in influencing the learning performance when using m-DCM to create virtual data. In the next section, a comparative study will be employed to demonstrate the capacity of m-DCM for the improvement of the learning performance of the supervised neural network.

4. A Comparative Study between IKDE and *m*-DCM

As in the introduction, the method of IKDE, which was proposed by Li and Lin (2006), has been able to improve accuracies by creating virtual data. The main difference between IKDE and *m*-DCM is that the data generated by IKDE are not bounded by the universal discourse of the data population; in contrast those generated by *m*-DCM are. As solving unbounded issue has been one of our primary motivations of this study, we shall show that *m*-DCM has overcome this shortcoming and improved the learning performance from the comparative results with IKDE.

4.1 The Design of the Comparative Analysis

As it was shown in Table 4, a sample of 40 data has been prepared for analysis. By randomly dividing the given 40 data into two groups: *n* training data and (40-*n*) testing data, two circumstances of n=10 and n=20 were considered of which 10 trials were made for each circumstance. For each trial, the *n* training data are the source of individual IKDE and *m*-DCM to create virtual data; and the corresponding (40-*n*) testing data are the examining target of the 3-4-1 BPNN. The accuracy was used again to evaluate the learning performance.

Since *c* values have been shown to have no significant association with the learning performance, we arbitrarily set c=5 in the algorithm of *m*-DCM. Virtual (training) data sizes (N) of 2^t*n with t=1~ 4 for n=10 and 20 are considered to evaluate the learning performance

under the same amount of virtual data generated through IKDE and *m*-DCM. Hence, each trial has four sets of virtual data, that is, *N*=20, 40, 80, and 160 when *n*=10 and *N*=40, 80, 160, and 320 when *n*=20.

Let the corresponding outcomes of accuracy be

$$\forall t = 1 \sim 4,\ f(t) = \left\{ f_{trial}(t) \mid trial = 1 \sim 10 \right\}. \tag{16}$$

Then, the average of accuracies (ACR) defined by $\frac{1}{10}\left(\sum_{trial=1}^{10} f_{trial}(t) \right), t = 1 \sim 4$ is used as the comparison index between IKDE and *m*-DCM.

4.2 The Result of the Comparative Analysis

Without virtual data, the average accuracies of two concerned circumstances are 0.32 when *n*=10 and 0.47 when *n*=20. It is undeniable that these accuracies are obviously too low to be accepted. Applying the methods of IKDE and *m*-DCM individually to create virtual training data, the computational results were tabulated in Table 9.

Table 9 The Result of the Comparative Analysis

n	10				20			
N	20	40	80	160	40	80	160	320
	ACR							
m-DCM	0.49	0.43	0.45	0.48	0.57	0.57	0.52	0.44
IKDE	0.51	0.41	0.20	0.18	0.56	0.50	0.25	0.20

Table 9 reveals that both IKDE and *m*-DCM are able to improve accuracies by creating virtual data. In the case of *n*=10, the best performance of IKDE and *m*-DCM both appeared at

20 virtual data with ACR =0.51 for IKDE and 0.49 for *m*-DCM. In the case of *n*=20, the best performance for both IKDE and *m*-DCM were obtained from 40 virtual data, and the corresponding ACR values were 0.56 and 0.57. Therefore, there was no significant difference between IKDE and *m*-DCM in terms of the highest ACR value.

In terms of the impact of virtual data sizes on the learning performance, Table 9 also shows that there is a rapid drop in the ACR value no matter it is *n*=10 or *n*=20. When using IKDE, the lowest ACR values with respect to *n*=10 and *n*=20 are 0.18 and 0.20. Both are much worse than those without any virtual data. This defect cannot be neglected especially there was no specific stopping rule to be designed in IKDE and thus a worse result is likely to be adopted in applications. In contrast, Table 9 shows the stability of *m*-DCM for VDG. In the case of *n*=10, the ACR value rises from the beginning of using virtual data for training, falls down to the lowest point of 0.43, and then gradually increases to the level of 0.48. At *n*=20 cases, the ACR value also rises at the start of using virtual data for training and then has a gradually decreasing trend as the number of virtual data increases. Judging from the above, it can be seen that *m*-DCM provides more stable learning performance than that of IKDE.

4.3 Conclusions

A case adopted from Li and Lin (2006) was used to show the applicability of the proposed *m*-DCM. The simulation experiment which employed 40 given data was done by two cases of

different amount of data groups. To present small sample sets, the given data were randomly divided into two designed cases of 10 and 20 training data with 30 and 20 corresponding testing data. Then, sensitivity analyses on the major parameters of *m*-DCM and a comparative study between *m*-DCM and IKDE (Li and Lin, 2006) were carried out.

On the basis of the result of our simulation experiment, we can provide the following conclusions: First, virtual data generation is able to improve the level of accuracy when there is only a small sample in hand; nevertheless, too many virtual data will devalue the level of accuracy. Therefore, providing a proper stopping rule as in *m*-DCM is necessary. Second, generating virtual data by *m*-DCM provides more stable learning performance than those by IKDE. This also confirms the contention that the bounded property of *m*-DCM is able to avoid the prediction error embedded in the method of IKDE as shown in Figure 1.

5. Discussion

Lack of sufficient training data is indeed responsible for the performance of supervised neural network learning. Although the direct way to overcome this difficulty is to collect sufficient amount of data for training, in many cases, this cannot be possible. Several methods that may generate additional data to train the network have been reported in the literature and do help improving the learning performance. However, as we have mentioned in the introduction, existing methods possess certain shortcomings. Since IKDE proposed by

Li and Lin (2006) has shown its capability on pattern recognition; yet remaining some issues related to unbounded and immeasurable data set. Therefore, in this study, we not only extended our developed one-dimensional Data Construction Method (DCM) to multi-dimensional DCM (*m*-DCM); but also applied this method to improve the learning performance of a BPNN as a comparative study with IKDE.

The result of our simulation experiment shows that *m*-DCM does not only improve the learning performance and its robustness but also processes several promising properties: (1) it is implemented through a simple procedure, i.e., matrix multiplication with Kronecker product; (2) it performs well for sample size as small as 10; (3) It provides a measure of the required size of the generated data to avoid excessive training; (4) It generates sufficient data within a very few iterations. Therefore, using *m*-DCM to generate virtual data for BPNN training is an efficient and effective tool. Further study would be placed on the possibility of applying *m*-DCM to the system with nominal variables.

Finally, although the parameter c in the multiplier C has been shown to be robust, the size of C matrix remains an interesting issue. The property of 2^t-fold generation is established on $C=[1 \;\; 1/c]^T$ with $c>1$, which includes only two elements. This is assumed in this study. Nonetheless, whether it is helpful to increase the efficiency of *m*-DCM by increasing the size of the matrix C will be another interesting issue to be studied in the future.

Acknowledgement

The authors appreciate constructive suggestions from anonymous referees and acknowledge the financial support from the National Science Council, ROC with project number NSC96-2221-E-543-001-MY3.

References

[1] Abu-Mostafa Y. S., Hints and the VC-dimension, Neural Computation 5 (1993), 278-288.

[2] Breiman, L., Bagging predictors, Machine Learning 24 (1996), 123-140.

[3] Buja, A., and Stuetzle W., The effect of bagging on variance, bias, and mean squared error. Preprint, AT& T Labs-Research (2000).

[4] Chen L.-W., Zuo Z., Medoff D., Holcomb H.H., Lahti A.C., and Tamminga C.A., How many subjects? A Monte Carlo bootstrap simulation for functional imaging, Schizophrenia Research 24 (1997), 164-164.

[5] Girosi Federico, and Nicholas Tung Chan, Prior knowledge and the creation of "virtual" examples for RBF networks, Proceedings of the IEEE Workshop on Neural Networks for Signal Processing, September 1995, Cambridge, MA.

[6] Graham Alexander, Kronecker Products and Matrix Calculus with Applications, Halsted Press, John Wiley & Sons, Inc., 1981.

[7] Huang Chongfu, Information diffusion techniques and small-sample problem,

International Journal of Information Technology and Decision Making 1(2002), 229-249.

[8] Huang Chongfu and Claudio Moraga, A diffusion-neural-network for learning from small samples, International Journal of Approximate Reasoning 35 (2004) 137-161.

[9] Huang Chun-Jung and Hsiao-Fan Wang, Virtual Sampling with Data Construction Method, Intelligent Data Analysis: Developing New Methodologies through Pattern Discovery and Recovery, Chapter ⅩⅧ, Idea Group Inc., USA, 2008.

[10] Jia Yanbing and Teresa B. Culver, Bootstrapped artificial neural networks for synthetic flow generation with a small data sample, Journal of Hydrology 33 (2006), 580-590.

[11] Li Der-Chang and Yao-San Lin, Using virtual sample generation to build up management knowledge in the early manufacturing stages, European Journal of Operational Research 175 (2006), 413-434.

[12] Niyogi, P., Girosi, F., and Tomaso, P., Incorporating prior information in machine learning by creating virtual examples, Proceedings of the IEEE 86 (1998), 275-298.

[13] NeuralWorks Professional Ⅱ/PLUS (tm) Win32, NeuralWare, Inc.

[14] Wang Hsiao-Fan and Chun-Jung Huang, Data Construction Method for the analysis of the spatial distribution of disastrous earthquakes in Taiwan, International Transactions in Operational Research 16 (2009), 189-212.

[15] Zhang Jie, Developing robust non-linear models through bootstrap aggregated neural networks, Neurocomputing 25 (1999), 93-113.

[16] Zhang J., Inferential estimation of polymer quality using bootstrap aggregated neural networks, Neural Networks 12 (1999), 927-938.