Preface

The foreign exchange market is one of the most complex dynamic markets with the characteristics of high volatility, nonlinearity and irregularity. Since the Bretton Woods System collapsed in 1970s, the fluctuations in the foreign exchange market are more volatile than ever. Furthermore, some important factors, such as economic growth, trade development, interest rates and inflation rates, have significant impacts on the exchange rate fluctuation. Meantime, these characteristics also make it extremely difficult to predict foreign exchange rates. Therefore, exchange rates forecasting has become a very important and challenge research issue for both academic and industrial communities.

In this monograph, the authors try to apply artificial neural networks (ANNs) to exchange rates forecasting. Selection of the ANN approach for exchange rates forecasting is because of ANNs' unique features and powerful pattern recognition capability. Unlike most of the traditional model-based forecasting techniques, ANNs are a class of data-driven, self-adaptive, and nonlinear methods that do not require specific assumptions on the underlying data generating process. These features are particularly appealing for practical forecasting situations where data are abundant or easily available, even though the theoretical model or the underlying relationship is unknown. Furthermore, ANNs have been successfully applied to a wide range of forecasting problems in almost all areas of business, industry and engineering. In addition, ANNs have been proved to be a universal functional approximator that can capture any type of complex relationships. Since the number of possible nonlinear relationships in foreign exchange rates data is typically large due to the high variability, ANNs have the advantage in approximating them well.

The main motivation of this monograph is to provide academic researchers and business practitioners recent developments in forecasting foreign exchange rates with ANNs. Therefore, some of the most important progress in foreign exchange rates forecasting with neural networks are first surveyed and then a few fully novel neural network models for exchange rates forecasting are presented. This monograph consists of six parts which are briefly described as follows.

Part I presents a survey on ANNs in foreign exchange rates forecasting. Particularly, this survey discusses the factors of affecting foreign exchange rates predictability with ANNs. Through a literature review and analysis, some implications and research issues are presented. According to the results and implications of this survey, the sequel parts will discuss those research issues respectively and provide the corresponding solutions.

In Part II, we provide a data preparation scheme for neural network data analysis. Some basic learning principles of ANNs are presented before an integrated data preparation scheme is proposed to remedy the literature gap.

In terms of the non-optimal choice of the learning rate and momentum factor in neural network learning algorithms in the existing literature, Part III constructs three single neural network models through deriving optimally adaptive learning rates and momentum factors. In the first single neural network model, optimal instantaneous learning rates and momentum factors are derived from the back-propagation (BP) algorithm. Using adaptive forgetting factors, an online BP learning model with optimally adaptive learning rates is developed to realize the online prediction of foreign exchange rates. In the third single neural network model, adaptive smoothing techniques are used to determine the momentum factor of neural network algorithm. Meantime, the proposed neural network model with adaptive smoothing momentum factors is applied to exchange rate index prediction.

In accordance with the analysis in the survey, the hybrid and ensemble models usually achieve better prediction performance than that of individual ANN model, Part IV and Part V present three hybrid neural network models and three ensemble neural network models, respectively.

In the first hybrid neural network model of Part IV, neural network and exponential smoothing model are hybridized into a synergetic model for foreign exchange rate prediction. Subsequently, the generalized linear autoregression (GLAR) and neural network models are fused by another neural network model in a nonlinear way. This model is applied to three typical foreign exchange rate forecasting problems and obtained good prediction performance. In the third model, a hybrid intelligent data mining approach integrating a novel ANN model — support vector machine (SVM) with genetic algorithm (GA) for exploring foreign exchange market movement tendency. In this model, a standard GA is first used to search through the possible combination of input features. The input features selected by GA are used to train SVM. The trained SVM is then used to predict exchange rate movement direction.

In the three ensemble neural network models of Part V, the first model presents a multistage ensemble framework to formulate an ensemble neural network model. In these stages of formulating ensemble models, some crucial issues are addressed. The second ensemble model introduces a meta-learning strategy to construct an exchange rate ensemble forecasting model. In some sense, an ensemble model is actually a meta-model. In the third ensemble model, a confidence-based neural network ensemble model is used to predict the exchange rate movement direction. In the last chapter of Part V, we propose a double-phase-processing procedure to solve the following two dilemmas, i.e., (1) whether should we select an appropriate modeling approach for the prediction purpose or should combine these different individual approaches into an ensemble forecast for the different/dissimilar models? (2) whether should we select the best candidate model for forecasting or to mix the various candidate models with different parameters into a new forecast for the same/similar modeling approaches?

Depended upon the previous methodology framework, an intelligent foreign exchange rates forecasting support system is developed by using client/server model and popular web technologies in Part VI. The description of this intelligent system is composed of two chapters. First of all, system conceptual framework, modeling techniques and system implementations are illustrated in details. Then an empirical and comprehensive assessment is performed. Empirical and comprehensive assessment results reveal that the intelligent exchange rate forecasting support system is one of the best forecasting systems by evaluating the performance of system implementation and comparing with the existing similar systems.

We would like to thank many friends and colleagues for their help and support in preparing this monograph. First, we thank Professor Wayne W. Huang of Ohio University, Professor Y. Nakamori of Japan Advanced Institute of Science and Technology and Professor Wei Huang of Huazhong University of Science and Technology for their contributions to this book. Three chapters are based on the results that we achieved jointly with them. We would like to thank several other scientists for their helpful suggestions and valuable comments on our research in this area, among them are Professor Kate A. Simth of Monash University in Australia, Professor Wuyi Yue of Konan University in Japan and Professor M. Makowski of International Institute of Applied Systems Analysis in Austria.

Finally, we would like to thank the National Natural Science Foundation of China (NSFC), Chinese Academy of Sciences (CAS), Academy of Mathematics and System Science (AMSS) of CAS, Hong Kong Research Granting Committee (HKRGC), City University of Hong Kong and University of Tsukuba in Japan for their financial support to our research in this promising area.

Lean YU
Institute of Systems Science
Academy of Mathematics and Systems Science
Chinese Academy of Sciences
Beijing, 100080, China

and

Department of Management Sciences
City University of Hong Kong
83 Tat Chee Avenue, Kowloon, Hong Kong
Email: yulean@amss.ac.cn

Shouyang WANG
Institute of Systems Science
Academy of Mathematics and Systems Science
Chinese Academy of Sciences
Beijing, 100080, China

and

Graduate School of Systems and Information Engineering
University of Tsukuba
Tsukuba, Ibaraki 305-8573, Japan
Email: sywang@amss.ac.cn

Kin Keung LAI
Department of Management Sciences
City University of Hong Kong
83 Tat Chee Avenue, Kowloon, Hong Kong
Email: mskklai@cityu.edu.hk

2 Basic Learning Principles of Artificial Neural Networks

2.1 Introduction

Artificial neural networks (ANNs), as an emerging discipline, studies or emulates the information processing capabilities of neurons of the human brain. It uses a distributed representation of the information stored in the network, and thus resulting in robustness against damage and corresponding fault tolerance (Shadbolt and Taylor, 2002). Usually, a neural network model takes an input vector ***X*** and produces output vector ***Y***. The relationship between ***X*** and ***Y*** is determined by the network architecture. There are many forms of network architecture inspired by the neural architecture of the human brain. The neural network generally consists of one input layer, one output layer, and one or more hidden layers, as illustrated in Fig. 2.1.

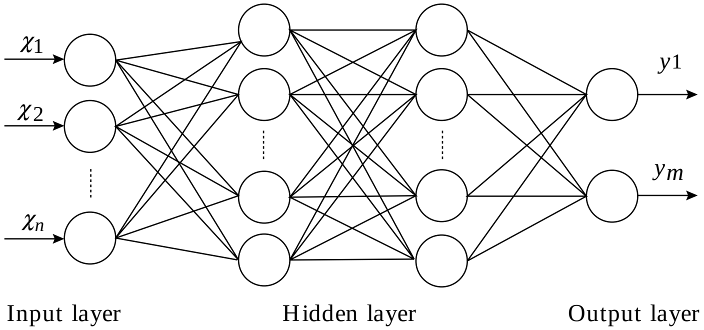

Fig. 2.1. The basic architecture of neural network

In the neural network model, it is widely accepted that a three-layer back propagation neural network (BPNN) with an identity transfer function in the output unit and logistic functions in the middle-layer units can approximate any continuous function arbitrarily well given a sufficient amount of middle-layer units (White, 1990). Furthermore, in the practical applications, about 70 percent of all problems are usually trained on a three-layer back-propagation network, as revealed by Chapter 1. The back-propagation learning algorithm, designed to train a feed-forward network, is an effective learning technique used to exploit the regularities and exceptions in the training sample.

A major advantage of neural networks is their ability to provide flexible mapping between inputs and outputs. The arrangement of the simple units into a multilayer framework produces a map between inputs and outputs that is consistent with any underlying functional relationship regardless of its “true” functional form. Having a general map between the input and output vectors eliminates the need for unjustified priori restrictions that are needed in conventional statistical and econometric modeling. Therefore, a neural network is often viewed as a “universal approximator”, i.e. a flexible functional form that can approximate any arbitrary function arbitrarily well, given sufficient middle-layer units and properly adjusted weights (Hornik et al., 1989; White, 1990). Both theoretical proof and empirical applications have confirmed that a three-layer BP neural network (BPNN) model with an identity transfer function in the output unit and logistic functions in the middle-layer units is adequate for foreign exchange rates forecasting, which is our research focus in this book. Therefore, a three-layer BP neural network model with identity activation function in the output unit and logistic function in the middle-layer units is used throughout this book except specially specified.

2.2 Basic Structure of the BPNN Model

Consider a three-layer BPNN, which has p nodes of the input layer, q nodes of the hidden layer and k nodes of the output layer. Mathematically, the basic structure of the BPNN model is described by (see derivation later)

$$\hat{Y}(t+1) = F_2[V^T(t)F_1(W(t)X(t))] \tag{2.1}$$

where $X=(x_0,x_1,\cdots,x_p)^T \in R^{(p+1)\times 1}$ are the inputs of BPNN, $\hat{Y}=(\hat{y}_0,\hat{y}_1,\cdots,\hat{y}_k)^T \in R^{k\times 1}$ is the outputs of the BPNN,

$$W = \begin{pmatrix} w_{10} & w_{11} & \cdots & w_{1p} \\ w_{20} & w_{21} & \cdots & w_{2p} \\ \cdots & \cdots & \cdots & \cdots \\ w_{q0} & w_{q1} & \cdots & w_{qp} \end{pmatrix} = (W_0, W_1, \cdots, W_p) \in R^{q\times(p+1)},$$

$$V = \begin{pmatrix} v_{10} & v_{20} & \cdots & v_{k0} \\ v_{11} & v_{21} & \cdots & v_{k1} \\ \cdots & \cdots & \cdots & \cdots \\ v_{1q} & v_{2q} & \cdots & v_{kq} \end{pmatrix} = (V_1, V_2, \cdots, V_k) \in R^{(q+1)\times k},$$

$F_1(W(t)X(t)) = \left(F_1(net_0(t))\ F_1(net_1(t))\ \cdots\ F_1(net_q(t))\right)^T \in R^{(q+1)\times 1}$, $net_i(t) = \sum_{j=0}^{p} w_{ij}(t)x_j(t)$, $i = 0, 1, \ldots, q$, is the output of the i-th hidden node. $w_{i0}(t)$, $i = 1, \ldots, q$, are the bias of the hidden nodes; $v_{ij}(t)$, $i = 1, \ldots, q$, $j = 1, \ldots, k$, are the weights form the hidden node i to the output node j; $v_{i0}(t)$ is the bias of the output node; $F_1(\bullet)$ and $F_2(\bullet)$ are the activation function, which can be any nonlinear function as long as they are continuous, bounded and differentiable. Typically, $F_1(\bullet)$ is a sigmoidal or hyperbolic tangent function and t is a time factor. For convenience, a symmetric hyperbolic tangent function (i.e., $f_1(x) = \tanh(u_0^{-1}x)$ where u_0 is the shape factor of the activation function) is used as the activation function of the hidden layer (Yu et al., 2005a, b).

Derivation of Equation (2.1):

$$\hat{Y}(t+1) = \begin{bmatrix} \hat{y}_1(t+1) \\ \hat{y}_2(t+1) \\ \cdots \\ \hat{y}_k(t+1) \end{bmatrix} = \begin{bmatrix} f_2[\sum_{i=1}^{q} f_1(\sum_{j=1}^{p} w_{ij}(t)x_j(t) + w_{i0}(t))v_{1i}(t) + v_{10}(t)] \\ f_2[\sum_{i=1}^{q} f_1(\sum_{j=1}^{p} w_{ij}(t)x_j(t) + w_{i0}(t))v_{2i}(t) + v_{20}(t)] \\ \cdots \\ f_2[\sum_{i=1}^{q} f_1(\sum_{j=1}^{p} w_{ij}(t)x_j(t) + w_{i0}(t))v_{ki}(t) + v_{k0}(t)] \end{bmatrix}$$

$$= \begin{bmatrix} f_2[\sum_{i=0}^{q} f_1(\sum_{j=0}^{p} w_{ij}(t)x_j(t))v_{1i}(t)] \\ f_2[\sum_{i=0}^{q} f_1(\sum_{j=0}^{p} w_{ij}(t)x_j(t))v_{2i}(t)] \\ \cdots \\ f_2[\sum_{i=0}^{q} f_1(\sum_{j=0}^{p} w_{ij}(t)x_j(t))v_{ki}(t)] \end{bmatrix} = \begin{bmatrix} f_2[\sum_{i=0}^{q} f_1(net_i)v_{1i}(t)] \\ f_2[\sum_{i=0}^{q} f_1(net_i)v_{2i}(t)] \\ \cdots \\ f_2[\sum_{i=0}^{q} f_1(net_i)v_{ki}(t)] \end{bmatrix}$$

$$= \begin{bmatrix} f_2[V_1^T F_1(W(t)X(t))] \\ f_2[V_2^T F_1(W(t)X(t))] \\ \cdots \\ f_2[V_k^T F_1(W(t)X(t))] \end{bmatrix} = F_2[V^T(t)F_1(W(t)X(t))]$$

where f_1 is the activation function of the hidden nodes and f_2 is the activation function of output nodes and t is a time factor. ■

Through estimating model parameter vector or network connection weights (W, V) via BPNN training and learning, we can realize the corresponding modeling tasks, such as function approximation, pattern recognition, and time series prediction.

2.3 Learning Process of the BPNN Algorithm

The ability to learn and improve its performance from examples is the neural network's fundamental trait. For BPNN, it is a class of supervised learning algorithm in the form of the neural network associative memory. Usually, the back-propagation learning mechanism consists of two phases: forward-propagation and back-propagation phase, as Tam and Kiang (1992) reported.

Suppose we have n samples. Each is described by $X_i = (x_{i1}, x_{i2}, \cdots, x_{ip})$ and $Y_i = (y_{i1}, y_{i2}, \cdots, y_{ik})$ where X_i is an input vector, Y_i is a target output vector and $1 \le i \le n$.

In the first phase (forward-propagation phase), X_i is fed into the input layer, and an output $\hat{Y}_i = (\hat{y}_{i1}, \hat{y}_{i2}, \cdots, \hat{y}_{ik})$ is generated based on the current weight vector W. The objective is to minimize an error function E defined as

$$E = \sum_{i=1}^{n} \sum_{j=1}^{k} \frac{(y_{ij} - \hat{y}_{ij})^2}{2} \tag{2.2}$$

Through changing W so that all input vectors are correctly mapped to their corresponding output vectors.

In the second phase (back-propagation phase), a gradient descent in the weight space, W, is performed to locate the optimal solution. The direction and magnitude change Δw_{ij} can be computed as

$$\Delta w_{ij} = -\frac{\partial E}{\partial w_{ij}} \varepsilon \tag{2.3}$$

where $0 < \varepsilon < 1$ is a learning parameter controlling the algorithm's convergence rate.

The total squared error calculated by Equation (2.1) is propagated back, layer by layer, from the output units to the input units in the second phase. Weight adjustments are determined on the way of propagation at each level. The two phases are executed during each iteration of the back-propagation algorithm until E converges.

2.4 Weight Update Formulae of the BPNN Algorithm

Actually, the model parameter vector or neural network weights (W, V) (as defined by Section 2.2) can be obtained by minimizing iteratively a cost function, $E(X\text{: } W, V)$. In general, $E(X\text{: } W, V)$ is a sum of the error squares cost function with k output nodes and n training pairs or patterns, that is,

$$\begin{aligned} E(X:W,V) &= \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}^2 = \frac{1}{2}\sum_{j=1}^{n} e_j^T e_j \\ &= \frac{1}{2}\sum_{j=1}^{n}[y_j - \hat{y}_j(X:W,V)]^T[y_j - \hat{y}_j(X:W,V)] \end{aligned} \tag{2.4}$$

where y_j is the jth actual value and $y_j(X\text{: } W,V)$ is the jth estimated value. Given the time factor t, Equation (2.4) can be rewritten as

$$\begin{aligned} E(t) &= \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}^2(t) = \frac{1}{2}\sum_{j=1}^{n} e_j^T(t) e_j(t) \\ &= \frac{1}{2}\sum_{j=1}^{n}[y_j(t) - \hat{y}_j(t)]^T[y_j(t) - \hat{y}_j(t)] \end{aligned} \tag{2.5}$$

where $e_j(t) = [e_{1j}(t) \quad e_{2j}(t) \quad \cdots \quad e_{kj}(t)]^T \in R^{k\times 1}$, $j = 1,2,\cdots,n$, $y_j(t)$ and $\hat{y}_j(t)$ are the jth actual value predicted value at time t, respectively.

By applying the gradient descent rule to the cost function $E(t)$ (as shown in Equation (2.5)) and considering Equation (2.1), we can obtain the weight increment formulae with respect to W and V, respectively (see proof later).

$$\Delta W(t) = -\eta(t)\nabla_W E(t) = \eta(t)\sum_{j=1}^{n} F'_{1(j)} V F'_{2(j)} e_j x_j^T \tag{2.6}$$

$$\Delta V(t) = -\eta(t)\nabla_V E(t) = \eta(t)\sum_{j=1}^{n} F_{1(j)} e_j^T F'_{2(j)} \tag{2.7}$$

where η is the learning rate, ∇ is the gradient operator, $\Delta W(t)$ and $\Delta V(t)$ are the weight adjustment increments at iteration t, respectively.

Suppose $\Delta W(t) = W(t) - W(t-1)$ and $\Delta V(t) = V(t) - V(t-1)$, then the weights update formulae for standard BP learning algorithm with respect to W and V are given by, respectively

$$W(t) = W(t-1) + \eta(t)\sum_{j=1}^{n} F'_{1(j)} V F'_{2(j)} e_j x_j^T \tag{2.8}$$

$$V(t) = V(t-1) + \eta(t)\sum_{j=1}^{n} F_{1(j)} e_j^T F'_{2(j)} \tag{2.9}$$

where Δ is the incremental operator, $F'_{1(j)} = \mathrm{diag}[f'_{1(1)} \ \cdots \ f'_{1(q)}] \in R^{q\times q}$,

$F'_{2(j)} = \mathrm{diag}[f'_{2(1)} \quad f'_{2(2)} \quad \cdots \quad f'_{2(k)}] \in R^{k\times k}$,

$$f'_{1(i)} = f'_1(net_i) = \frac{\partial f_1(net_i)}{\partial net_i}, i = 1,2,\cdots,q,$$

$$f'_{2(i)} = f'_2[v_i^T F_1(WX)] = \frac{\partial f_2[v_i^T F_1(WX)]}{\partial[v_i^T F_1(WX)]}, i = 1,2,\cdots k,$$

$$V = \begin{bmatrix} v_{11} & v_{21} & \cdots & v_{k1} \\ v_{12} & v_{22} & \cdots & v_{k2} \\ \cdots & \cdots & \cdots & \cdots \\ v_{1q} & v_{2q} & \cdots & v_{kq} \end{bmatrix} = [v_1 \ \ v_2 \ \ \cdots \ v_k] \in R^{q\times p},$$

$v_i = [v_{i1} \ \ \cdots \ \ v_{iq}]^T \in R^{q\times 1}, i = 1,2,\cdots,k$.

In order to prove Equations (2.8) and (2.9), three lemmas must be firstly introduced (Sha and Bajic, 2002).

Lemma 2.1

The derivative of activation function $F_1(WX)$ in hidden layer with respect to the vector *Net* or *WX* is given by

$$F_1'(WX) = \mathrm{diag}\,[\, f_1'(net_1) \quad f_1'(net_2) \quad \cdots \quad f_1'(net_q)]$$
$$= \frac{1}{u_0}\begin{bmatrix} 1 - f_1^2(net_1) & 0 & \cdots & 0 \\ 0 & 1 - f_1^2(net_2) & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 - f_1^2(net_q) \end{bmatrix}$$
$$= \frac{1}{u_0}\begin{bmatrix} 1 - f_1^2(W_1X) & 0 & \cdots & 0 \\ 0 & 1 - f_1^2(W_2X) & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 - f_1^2(W_qX) \end{bmatrix}$$

where $F_1(WX) = \left[f_1(net_0) \quad f_1(net_1) \quad \cdots \quad f_1(net_q)\right]^T$, $F_1'(WX)$ is the Jacobian matrix of $F_1(WX)$, $f_1'(net_i) = \partial f_1(net_i)/\partial net_i$, i = 0, 1, …, q, $f_1(x) = \tanh(u_0^{-1}x)$ and its derivative $f_1'(x) = u_0^{-1}[1 - f_1^2(x)]$.

Proof:

Due to $f_1(x) = \tanh(u_0^{-1}x)$ and its derivative $f_1'(x) = u_0^{-1}[1 - f_1^2(x)]$, $net_i = W_i X$, $F_1(WX) = \left[f_1(net_0)\ f_1(net_1)\ \cdots\ f_1(net_q) \right]^T$, the derivative of activation function $F_1(WX)$ in hidden layer with respect to the vector *Net* or *WX* can be calculated as

$$F_1'(WX) = \begin{bmatrix} \frac{\partial f_1(net_1)}{\partial net_1} & \cdots & \frac{\partial f_1(net_1)}{\partial net_q} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_1(net_q)}{\partial net_1} & \cdots & \frac{\partial f_1(net_q)}{\partial net_q} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1(net_1)}{\partial net_1} & \cdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \cdots & \frac{\partial f_1(net_q)}{\partial net_q} \end{bmatrix}$$

$$= \begin{bmatrix} f_1'(net_1) & \cdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \cdots & f_1'(net_q) \end{bmatrix} = diag\left[f_1'(net_1) \quad \cdots \quad f_1'(net_q) \right]$$

$$= \begin{bmatrix} u_0^{-1}\left[1 - f_1^2(net_1)\right] & 0 & \cdots & 0 \\ 0 & u_0^{-1}\left[1 - f_1^2(net_2)\right] & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & u_0^{-1}\left[1 - f_1^2(net_q)\right] \end{bmatrix}$$

$$= \frac{1}{u_0} \begin{bmatrix} 1 - f_1^2(W_1 X) & 0 & \cdots & 0 \\ 0 & 1 - f_1^2(W_2 X) & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 - f_1^2(W_q X) \end{bmatrix} \qquad \blacksquare$$

Lemma 2.2

The partial derivative of the single hidden output $V^T F_1(WX)$ with respect to the weight matrix *W* is given by

$$\frac{\partial [V^T F_1(WX)]}{\partial W} = F_1'(WX) V X^T$$

Proof:

$$\frac{\partial [V^T F_1(WX)]}{\partial W} = \left[\frac{\partial [\sum_{i=0}^{q} v_i f_1(\sum_{j=0}^{p} w_{ij} x_j)]}{\partial w_{ij}} \right]_{q\times(p+1)} = [v_i f_1'(net_i) x_j]_{q\times(p+1)}$$

$$= \begin{bmatrix} v_1 f_1'(net_1)x_0 & v_1 f_1'(net_1)x_1 & \cdots & v_1 f_1'(net_1)x_p \\ v_2 f_1'(net_2)x_0 & v_2 f_1'(net_2)x_1 & \cdots & v_2 f_1'(net_2)x_p \\ \cdots & \cdots & \cdots & \cdots \\ v_q f_1'(net_q)x_0 & v_q f_1'(net_q)x_1 & \cdots & v_q f_1'(net_q)x_p \end{bmatrix}$$

$$= \begin{bmatrix} f_1'(net_1) & \cdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \cdots & f_1'(net_q) \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ \cdots \\ v_q \end{bmatrix} \cdot \begin{bmatrix} x_0 & x_1 & \cdots & x_p \end{bmatrix}$$

$$= F_1'(WX)VX^T \qquad \blacksquare$$

Lemma 2.3

The partial derivative of the single hidden output $V^T F_1(WX)$ with respect to the weight vector V is given by

$$\frac{\partial[V^T F_1(WX)]}{\partial V} = F_1(WX)$$

Proof:

$$\frac{\partial[V^T F_1(WX)]}{\partial V} = \left[\frac{\partial[\sum_{i=0}^{q} v_i f_1(net_i)]}{\partial v_i} \right]_{(q+1)\times 1}$$

$$= [f_1(net_i)]_{(q+1)\times 1} = F_1(WX) \qquad \blacksquare$$

Using Lemmas 2.1-2.3, we can prove the weight update formula in the following. First of all, we derive the gradient of $E(t)$ with respect to weights W and V.

With reference to Equation (2.5) and Lemmas 2.1 and 2.2, the gradient of $E(t)$ with respect to W can be obtained by applying the steepest descent method to $E(t)$,

$$\nabla_W E(t) = \frac{\partial E(t)}{\partial W(t)} = \sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}(t)\frac{\partial e_{ij}(t)}{\partial W(t)} = -\sum_{i=1}^{k} e_{ij}(t)\frac{\partial \hat{y}_{ij}(t)}{\partial W(t)}$$

$$= -\sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}(t) F'_{2(j)}\left[v_i^T F_{1(j)}(Wx_j)\right]\frac{\partial[v_i^T F_1(Wx_j)]}{\partial W(t)}$$

$$= -\sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij} F'_{2(j)} F'_{1(j)} VX^T = -\sum_{j=1}^{n} F'_{1(j)}\left(\sum_{i=1}^{k} e_{ij} F'_{2(j)} v_i\right) x_j^T$$

$$= -\sum_{j=1}^{n} F'_{1(j)} \begin{bmatrix} e_{1j} f'_{2(1j)} v_{11} + e_{2j} f'_{2(2j)} v_{21} + \cdots e_{kj} f'_{2(kj)} v_{k1} \\ e_{1j} f'_{2(1j)} v_{12} + e_{2j} f'_{2(2j)} v_{22} + \cdots e_{kj} f'_{2(kj)} v_{k2} \\ \cdots \\ e_{1j} f'_{2(1j)} v_{1q} + e_{2j} f'_{2(2j)} v_{2q} + \cdots e_{kj} f'_{2(kj)} v_{kq} \end{bmatrix} x_j^T$$

$$= -\sum_{j=1}^{n} F'_{1(j)} \begin{bmatrix} v_{11} & v_{21} & \cdots & v_{k1} \\ v_{12} & v_{22} & \cdots & v_{k2} \\ \cdots & \cdots & \cdots & \cdots \\ v_{1q} & v_{2q} & \cdots & v_{kq} \end{bmatrix} \cdot \begin{bmatrix} e_{1j} f'_{2(1j)} \\ e_{2j} f'_{2(2j)} \\ \cdots \\ e_{kj} f'_{2(kj)} \end{bmatrix} x_j^T$$

$$= -\sum_{j=1}^{n} F'_{1(j)} \begin{bmatrix} v_{11} & v_{21} & \cdots & v_{k1} \\ v_{12} & v_{22} & \cdots & v_{k2} \\ \cdots & \cdots & \cdots & \cdots \\ v_{1q} & v_{2q} & \cdots & v_{kq} \end{bmatrix}$$

$$\times \begin{bmatrix} f'_{2(1j)} & 0 & \cdots & 0 \\ 0 & f'_{2(2j)} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & f'_{2(kj)} \end{bmatrix} \cdot \begin{bmatrix} e_{1j} \\ e_{2j} \\ \cdots \\ e_{kj} \end{bmatrix} \cdot x_j^T$$

$$= -\sum_{j=1}^{n} F'_{1(j)} V F'_{2(j)} e_j x_j^T$$

Similarly, the gradient of $E(t)$ with respect to V can also be obtained

$$\nabla_V E(t) = \frac{\partial E(t)}{\partial V(t)} = \sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}(t) \frac{\partial e_{ij}(t)}{\partial V(t)} = -\sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}(t) \frac{\partial \hat{y}_{ij}(t)}{\partial V(t)}$$

$$= -\sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}(t) \begin{bmatrix} \frac{\partial \hat{y}_{ij}}{\partial v_{10}} & \frac{\partial \hat{y}_{ij}}{\partial v_{20}} & \cdots & \frac{\partial \hat{y}_{ij}}{\partial v_{i0}} & \cdots & \frac{\partial \hat{y}_{ij}}{\partial v_{k0}} \\ \frac{\partial \hat{y}_{ij}}{\partial v_{11}} & \frac{\partial \hat{y}_{ij}}{\partial v_{21}} & \cdots & \frac{\partial \hat{y}_{ij}}{\partial v_{i1}} & \cdots & \frac{\partial \hat{y}_{ij}}{\partial v_{k1}} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial \hat{y}_{ij}}{\partial v_{1q}} & \frac{\partial \hat{y}_{ij}}{\partial v_{2q}} & \cdots & \frac{\partial \hat{y}_{ij}}{\partial v_{iq}} & \cdots & \frac{\partial \hat{y}_{ij}}{\partial v_{kq}} \end{bmatrix}$$

$$
= -\sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}(t) F_2'[V^T F_1(WX)] \begin{bmatrix} 0 & 0 & \cdots & f_{1(0j)} & \cdots & 0 \\ 0 & 0 & \cdots & f_{1(1j)} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & f_{1(qj)} & \cdots & 0 \end{bmatrix}
$$

$$
= -\sum_{j=1}^{n} \begin{bmatrix} e_{1j} f'_{2(1j)} f_{1(0j)} & \cdots & e_{ij} f'_{2(ij)} f_{1(0j)} & \cdots & e_{kj} f'_{2(kj)} f_{1(0j)} \\ e_{1j} f'_{2(1j)} f_{1(1j)} & \cdots & e_{ij} f'_{2(ij)} f_{1(1j)} & \cdots & e_{kj} f'_{2(kj)} f_{1(1j)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ e_{1j} f'_{2(1j)} f_{1(qj)} & \cdots & e_{ij} f'_{2(ij)} f_{1(qj)} & \cdots & e_{kj} f'_{2(kj)} f_{1(qj)} \end{bmatrix}
$$

$$
= -\sum_{j=1}^{n} \begin{bmatrix} f_{1(0j)} \\ f_{1(1j)} \\ \cdots \\ f_{1(qj)} \end{bmatrix} \begin{bmatrix} e_{1j} f'_{2(1j)} & e_{2j} f'_{2(2j)} & \cdots & e_{kj} f'_{2(kj)} \end{bmatrix}
$$

$$
= -\sum_{j=1}^{n} F_{1(j)} \begin{bmatrix} e_{1j} & e_{2j} & \cdots & e_{kj} \end{bmatrix} \begin{bmatrix} f'_{2(1j)} & 0 & \cdots & 0 \\ 0 & f'_{2(2j)} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & f'_{2(kj)} \end{bmatrix}
$$

$$
= -\sum_{j=1}^{n} F_{1(j)} e_j^T F'_{2(j)}
$$

Therefore, the weight increments $\Delta W(t)$ and $\Delta V(t)$ can be obtained from the above derivation processes, i.e.,

$$
\Delta W(t) = -\eta(t)\nabla_W E(t) = \eta(t)\sum_{j=1}^{N} F'_{1(j)} V F'_{2(j)} e_j x_j^T
$$

$$
\Delta V(t) = -\eta(t)\nabla_V E(t) = \eta(t)\sum_{j=1}^{n} F_{1(j)} e_j^T F'_{2(j)}
$$

Assume $\Delta W(t) = W(t) - W(t-1)$ and $\Delta V(t) = V(t) - V(t-1)$, then the weights update rule (i.e., Equations (2.8) and (2.9)) for standard BP learning algorithm with respect to W and V can be obtained, i.e.,

$$
W(t) = W(t-1) + \eta(t)\sum_{j=1}^{n} F'_{1(j)} V F'_{2(j)} e_j x_j^T
$$

$$
V(t) = V(t-1) + \eta(t)\sum_{j=1}^{n} F_{1(j)} e_j^T F'_{2(j)}
$$

Using the weight update formulae (i.e., Equations (2.8) and (2.9)), we can train BPNN to perform the corresponding tasks, such as data mining, function approximation and financial time series forecasting.

2.5 Conclusions

In this chapter, some preliminaries about back-propagation neural networks are presented. First of all, a basic architecture of three-layer BPNN model is described in the form of matrix. Then we briefly introduce a basic learning process including forward propagation phase and back-propagation phase for BPNN. Based upon the basic structure and learning process, the weight update rules are derived in terms of steepest gradient descent algorithm. Using the weight update rules, some data analysis tasks are performed.

However, in the neural network applications, an important process, data preparation process, is often neglected by researchers and business users. Although data preparation in neural network data analysis is important, some existing literature about the neural network data preparation are scattered, and there is no systematic study about data preparation for neural network data analysis (Yu et al., 2006a). Therefore, this book tries to develop an integrated data preparation scheme for neural network data analysis, which will be described in the next chapter.