CHAPTER 5 Pattern Recognition with Neural Network

5.1 Introduction

While feature analysis is an essential and important step towards designing effective clustering and classification algorithms as discussed in Chapter 3. This chapter deals with clustering method that looks for substructures present in a data set and partitions this data set into homogeneous groups. A classifier, on the other hand, partitions the feature space into appropriate classes as will be explained in Chapter 6.

First, we review some of the theory of neural network in Section 5.2 that is necessary for us to understand how to perform pattern recognition using neural computation. In section 5.3 and 5.4 we present the development of Kalman Filter and Extended Kalman Filter respectively for the weight optimization of the neural networks. The training procedures for the neural networks are explained in Section 5.5.

5.2 Theory of Neural Network

If there is sufficient information available about the physical laws of the system, then a mathematical approach may be suitable, where equations describing the underlying mechanism responsible for time series generation are derived based on first principles. In many real problems of interest, we do not have enough a priori information to attempt such an approach. In fact, although systems have several dynamical variables that govern their behaviour, often, only measurements of a single variable are available in the form of a series of time-dependent data. In the absence of enough information to derive equations mathematically, it is more attractive to use a model-based approach, where a network is trained to mimic the generating equations of a time series. Therefore, instead of explicitly deducing equations describing the underlying dynamics of the system, an implicit neural model is built that approximates the ideal equations. Neural networks are attractive for

modelling nonlinear dynamic systems because they are well established function approximators. It has been shown by the universal approximation theorem that a neural network, with an arbitrary number of units and with two or more layers could approximate any uniformly continuous function [68]. In the proposed modelling scheme, Multi Layer Perceptron Network (MLPN) [69] is used as the infrastructure.

We present the MLPN model as shown in Figure 5.1, where each circle denotes a neuron. The leftmost column is called input layer, the column in the middle is called hidden layer and the rightmost column is called output layer.

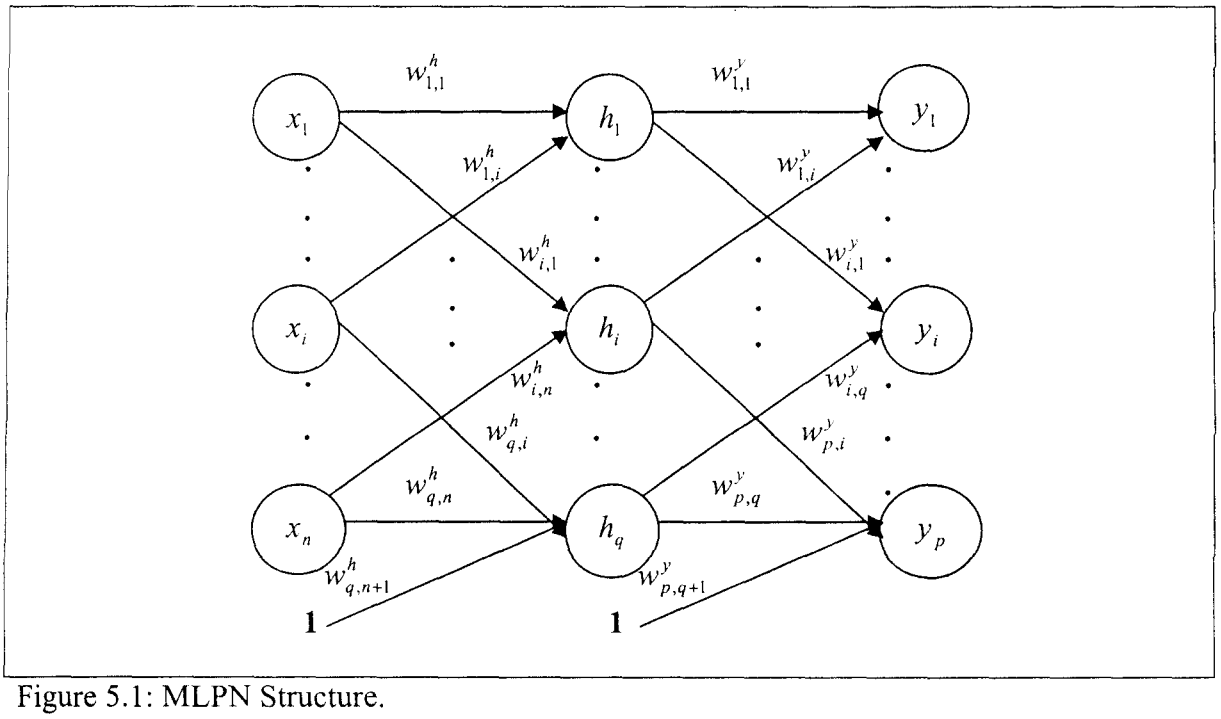

Figure 5.1: MLPN Structure.

The weight, $w^h_{j,i}$ represents strength of the synaptic connection between the ith neuron of the input layer and the jth neuron of the hidden layer whereas the weight, $w^y_{j,i}$ represents strength of the synaptic connection between the ith neuron of the hidden layer and the jth neuron of the output layer. In order to model the neural activity, we denote the weight matrices as described in the following.

$$W^h = \begin{bmatrix} W_1^h \\ . \\ . \\ . \\ W_q^h \end{bmatrix} = \begin{bmatrix} w_{1,1}^h & . & . & . & w_{1,n+1}^h \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ w_{q,1}^h & . & . & . & w_{q,n+1}^h \end{bmatrix} \tag{5.1}$$

$$W^y = \begin{bmatrix} W_1^y \\ . \\ . \\ . \\ W_p^y \end{bmatrix} = \begin{bmatrix} w_{1,1}^y & . & . & . & w_{1,q+1}^y \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ w_{p,1}^y & . & . & . & w_{p,q+1}^y \end{bmatrix} \tag{5.2}$$

$$\theta = \begin{bmatrix} \theta^h \\ \theta^y \end{bmatrix} = \begin{bmatrix} (W_1^h)^T \\ . \\ . \\ . \\ (W_q^h)^T \\ (W_1^y)^T \\ . \\ . \\ . \\ (W_p^y)^T \end{bmatrix} = \begin{bmatrix} w_{1,1}^h \\ . \\ . \\ w_{1,n+1}^h \\ . \\ . \\ w_{q,1}^h \\ . \\ . \\ w_{q,n+1}^h \\ w_{1,1}^y \\ . \\ . \\ w_{1,q+1}^y \\ . \\ . \\ w_{p,1}^y \\ . \\ . \\ w_{p,q+1}^y \end{bmatrix} \tag{5.3}$$

where (W_i^h) is the ith row vector in (W^h) and (W_i^y) is the ith row vector in (W^y).

The MLPN model with the optimum parameter vector, θ can be described by the following equations,

$$\theta(k+1) = \theta(k) \tag{5.4}$$

$$y(k) = \hat{y}(k) + e_m(k) \tag{5.5}$$

where

$$\hat{y}(k) = \hat{g}[\theta(k), x(k)] = W^y\left[\frac{1}{1+\exp(-W^h x(k))}\right] \tag{5.6}$$

and $e_m(k) \in \Re^p$ is assumed to be a white noise vector and regarded as a modelling error.

5.3 Development of Kalman Filter (KF)

The Extended Kalman Filter (EKF) is used to perform supervised training of Multi Layer Perceptron Network (MLPN). To describe how to proceed with this approach, the Kalman Filter (KF) method is first described in detail. After the foundation is laid, the application of the Extended Kalman Filter (EKF) to neural network training is then described.

Since the Kalman Filter (KF) is well suited for vector processes, we begin by assuming that the random process to be estimated can be modelled in the form

$$\theta(n) = A(n-1)\theta(n-1) + e_1(n) \tag{5.7}$$

which is known as the signal (or state vector) model where

$\theta(n)$ = p x 1 signal state vector at time n

$A(n-1)$ = p x p matrix that relates $\theta(n-1)$ to $\theta(n)$ in absence of a forcing function

$e_1(n)$ = p x 1 zero-mean white noise sequence with covariance matrix $E_1(n)$

The matrix $A(n-1)$ is known as the state-transition matrix while $e_1(n)$ is also known as the modelling error vector.

The observation (or measurement) model is described using the linear relationship

$$y(n) = C(n)\theta(n) + e_2(n) \tag{5.8}$$

where

$y(n)$ = q x 1 signal state vector at time n

$C(n)$ = q x p matrix that gives ideal linear relationship between $\theta(n)$ and $y(n)$

$e_2(n)$ = p x 1 zero-mean white noise sequence with covariance matrix $E_2(n)$

The matrix $C(n)$ is known as the output matrix, and the sequence $e_2(n)$ is known as the observation error.

In the development of the discrete Kalman Filter (KF), we will let $\hat{\theta}(n|n)$ denote the best linear estimate of $\theta(n)$ at time n given the observations $y(i)$ for $i = 1,2,...,n$, and we will let $\hat{\theta}(n|n-1)$ denote the best estimate given the observations up to time $n-1$. With $e_k(n|n)$ and $e_k(n|n-1)$ the corresponding state estimation errors,

$$e_k(n|n) = \theta(n) - \hat{\theta}(n|n) \tag{5.9}$$

$$e_k(n|n-1) = \theta(n) - \hat{\theta}(n|n-1) \tag{5.10}$$

and $E_k(n|n)$ and $E_k(n|n-1)$ the error covariance matrices,

Assuming that the system is linear and that the noise is white and Gaussian, the Kalman Filter (KF) is optimal in the sense that the mean squared error, $E\left\{\left\|e_k(n|n)\right\|^2\right\}$ is minimized. In the first step, since no new measurements are used to estimate $\theta(n)$, all that is known is that $\theta(n)$ evolves according to the state equation

$$\theta(n) = A(n-1)\theta(n-1) + e_1(n) \tag{5.11}$$

Since $e_1(n)$ is a zero mean white noise process (and the values of $e_1(n)$ are unknown), then we may predict $\theta(n)$ as follows,

$$\hat{\theta}(n|n-1) = A(n-1)\hat{\theta}(n-1|n-1) \tag{5.12}$$

which has an estimation error given by

$$e_k(n|n-1) = \theta(n) - \hat{\theta}(n|n-1) \tag{5.13}$$

$$= A(n-1)\theta(n-1) + e_1(n) - A(n-1)\hat{\theta}(n-1|n-1) \tag{5.14}$$

$$= A(n-1)e_k(n-1|n-1) + e_1(n) \tag{5.15}$$

Note that since $e_1(n)$ has zero mean, if $\hat{\theta}(n-1|n-1)$ is an unbiased estimate of $\theta(n-1)$, i.e.,

$$E\{e_k(n-1|n-1)\} = 0 \tag{5.16}$$

then $\hat{\theta}(n|n-1)$ will be an unbiased estimate of $\theta(n)$,

$$E\{e_k(n|n-1)\} = 0 \tag{5.17}$$

Finally, since the estimation error $e_k(n-1|n-1)$ is uncorrelated with $e_1(n)$ (a consequence of the fact that $e_1(n)$ is a white noise sequence), then

$$E_k(n|n-1) = A(n-1)E_k(n-1|n-1)A^H(n-1) + E_1(n) \tag{5.18}$$

where $E_1(n)$ is the covariance matrix for the noise process $e_1(n)$. This completes the first step of the Kalman Filter (KF).

In the second step, we incorporate the new measurement $y(n)$ into the estimate $\hat{\theta}(n|n-1)$. A linear estimate of $\theta(n)$ that is based on $\hat{\theta}(n|n-1)$ and $y(n)$ is of the form

$$\hat{\theta}(n|n) = K'(n)\hat{\theta}(n|n-1) + K(n)y(n) \tag{5.19}$$

$$\theta(n) - \hat{\theta}(n|n) = \theta(n) - \{K'(n)\hat{\theta}(n|n-1) + K(n)y(n)\} \tag{5.20}$$

$$e_k(n|n) = \theta(n) - K'(n)[\theta(n) - e_k(n|n-1)] + K(n)[C(n)\theta(n) + e_2(n)] \tag{5.21}$$

$$e_k(n|n) = [I - K'(n) - K(n)C(n)]\theta(n) + K'(n)e_k(n|n-1) - K(n)e_2(n) \tag{5.22}$$

The requirement that is imposed on $\hat{\theta}(n|n)$ is that it be unbiased, $E\{e_k(n|n)\} = 0$,

since $E\{e_2(n)\} = 0$ & $E\{e_k(n|n-1)\} = 0$, then $\hat{\theta}(n|n)$ will be unbiased for any $\theta(n)$ only if the term in brackets is zero,

$$K'(n) = I - K(n)C(n) \tag{5.23}$$

With this constraint, it follows from Eq. (4.19) that $\hat{\theta}(n|n)$ has the form

$$\hat{\theta}(n|n) = [I - K(n)C(n)]\,\hat{\theta}(n|n-1) + K(n)y(n) \tag{5.24}$$

or

$$\hat{\theta}(n|n) = \hat{\theta}(n|n-1) + K(n)[y(n) - C(n)\hat{\theta}(n|n-1)] \tag{5.25}$$

and the error is

$$e_k(n|n) = K'(n)e_k(n|n-1) - K(n)e_2(n) \tag{5.26}$$

$$= [I - K(n)C(n)]e_k(n|n-1) - K(n)e_2(n) \tag{5.27}$$

The error covariance matrix for $e_k(n|n)$ is

$$E_k(n|n) = [I - K(n)C(n)]E_k(n|n-1) \tag{5.28}$$

The Kalman gain is

$$K(n) = E_k(n|n-1)C^H(n)[C(n)E_k(n|n-1)C^H(n) + E_2(n)]^{-1} \tag{5.29}$$

Derivation of the Discrete Kalman Filter (KF) algorithm is described in Appendix B.

5.4 Extended Kalman Filter (EKF) based Multi Layer Perceptron Network (MLPN)

There are two fundamental assumptions in the derivation of the Kalman Filter (KF). The first is that the system is described by a linear state-space model and the second is that the noise terms are white and Gaussian with zero-mean, and they are also assumed to be uncorrelated with each other and with the initial state. When these assumptions are

satisfied, the Kalman Filter (KF) is optimal in the mean-squared error sense. When the system under consideration is nonlinear such as the Multi Layer Perceptron Network (MLPN) model, the first condition is violated and the Extended Kalman Filter (EKF) is applied as a sub-optimal filter and employed in the training algorithm for the neural model. In essence, the neural network's training problem is cast into the framework of a classic state estimation problem. In such a scenario, the neural network is described as a nonlinear dynamical system in a state-space model where the weights are treated as the state of the system and the goal is to use a series of measurements to determine the optimal estimate of the state variables. In the EKF method, the output error, is scaled by the Kalman gain and then used to update the weights according to the EKF equations. Training continues in this fashion for several epochs through the training data until satisfactory neural model is achieved i.e. In this study, the EKF training method is used due to its well-known superior convergence speed and lower tendency to get stuck in local minima as compared to other gradient descent methods i.e. backpropagation (BP) algorithm. BP is a gradient decent algorithm that tries to minimize the average squared error of the MLPN. BP was first developed by P. Werbos in 1974 [70]. After it is popularized by Rummelhart et al. in 1986 [71], many variations of this algorithm is developed, such as BP with momentum and variable learning rate, to improve its convergence speed and performance in avoiding getting stuck into local minima. There are also many advanced learning algorithms for the MLPN such as Fletcher-Powell, Polak-Ribire, Powell-Beale, One-Step-Secant, BFGS quasi-Newton and Levenberg-Marquardt. These functions are available for batch training in Matlab [72]. They are listed in Table 5.1 together with the relative time required to reach convergence [73]. The EKF algorithm is applied in this research because it has superior online learning capability over all such algorithms as demonstrated in Section 7.1. The EKF is considered one of the most effective methods for both nonlinear state estimation and

parameter estimation (e.g., learning the weights of MLPN). The EKF has been applied extensively to the field of nonlinear estimation, i.e. the training of feed-forward MLPN for process fault-tolerant control based on neural networks by T.K. Chang [68, 69].

Matlab Function	Technique	Time	Epochs	Mflops
traingdx	BP(Variable Learning Rate)	57.71	980	2.50
traincgf	Fletcher-Powell	16.40	81	0.99
traincgp	Polak-Ribire	19.16	89	0.75
traincgb	Powell-Beale	15.03	74	0.59
trainoss	One-Step-Secant	18.46	101	0.75
trainbfg	BFGS quasi-Newton	10.86	44	1.02
trainlm	Levenberg-Marquardt	1.87	6	0.46

Table 5.1: Learning algorithms for MLPN with a relative time to reach convergence [73].

System:

$$\theta(k+1) = A(k)\theta(k) + B(k)u(k) + e_1(k) \tag{5.30}$$

$$y(k) = C(k)\theta(k) + e_2(k) \tag{5.31}$$

Prediction:

$$\hat{\theta}(k) = A(k)\hat{\theta}(k) + B(k)u(k) \tag{5.32}$$

$$E_0(k+1) = A(k)E_w(k)A(k)^T + E_1(k) \tag{5.33}$$

Correction:

$$\hat{\theta}(k) = \hat{\theta}(k-1) + K_k(k)[y(k) - C(k)\hat{\theta}(k-1)] \tag{5.34}$$

$$E_w(k) = [I - K_k(k)C(k)]E_0(k) \tag{5.35}$$

$$K_k(k) = E_0(k)C(k)^T[E_2(k) + C(k)E_0(k)C(k)^T]^{-1} \tag{5.36}$$

The EKF-based training algorithm for the MLPN is presented in the following whereby the original vectors of $K_k(k), C(k), E_0(k)$ are replaced by $K_w(k), C_w(k), E_w(k-1)$. The observation error covariance matrix, $E_2(k)$ cannot be determined in this case and hence it is replaced by an unknown priori error covariance matrix, $R|_{k|k-1}$.

$$\hat{\theta}(k) = \hat{\theta}(k-1) + K_w(k)[y(k) - \hat{y}|_{k|k-1}] \tag{5.37}$$

$$E_w(k) = [I - K_w(k)C_w(k)]E_w(k-1) \tag{5.38}$$

$$K_w(k) = E_w(k-1)C_w(k)^T[R|_{k|k-1} + C_w(k)E_w(k-1)C_w(k)^T]^{-1} \tag{5.39}$$

where

$\hat{y}|_{k|k-1} = \hat{g}[\hat{\theta}(k-1), x(k)]$ and $x(k)$ is used to denote the input vector to MLPN.

$$C_w(k) = \left.\frac{\partial \hat{y}(k)}{\partial \theta}\right|_{\theta=\hat{\theta}(k-1)} \tag{5.40}$$

$R|_{k|k-1}$ is an unknown priori error covariance matrix.

$$R|_{k|k-1} = R(k-1) + \frac{1}{k}\{[y(k) - \hat{y}|_{k|k-1}][y(k) - \hat{y}|_{k|k-1}]^T - R(k-1)\} \tag{5.41}$$

$$R(k) = R(k-1) + \frac{1}{k}\{[y(k) - \hat{y}(k)|_{\theta=\hat{\theta}(k)}][y(k) - \hat{y}(k)|_{\theta=\hat{\theta}(k)}]^T - R(k-1)\} \tag{5.42}$$

The $C_w(k) \in \Re^{px[q(n+1)+p(q+1)]}$ is the Jacobian matrix of model output, $\hat{y}(k)$ with respect to θ, and can be expressed as

$$C_w(k) = \left.\frac{\partial \hat{y}(k)}{\partial \theta}\right|_{\theta=\hat{\theta}(k-1)} = \left[\left.\frac{\partial \hat{y}(k)}{\partial \theta^h}\right|_{\theta=\hat{\theta}(k-1)} \left.\frac{\partial \hat{y}(k)}{\partial \theta^y}\right|_{\theta=\hat{\theta}(k-1)}\right] \tag{5.43}$$

From Eq. (5.43) the term $\left.\frac{\partial \hat{y}(k)}{\partial \theta^h}\right|_{\theta=\hat{\theta}(k-1)}$ can be further expressed as

$$\left.\frac{\partial \hat{y}(k)}{\partial \theta^h}\right|_{\theta=\hat{\theta}(k-1)} = \left[\frac{\partial \hat{y}(k)}{\partial h(k)}\frac{\partial h(k)}{\partial z(k)}\frac{\partial z(k)}{\partial \theta^h}\right]_{\theta=\hat{\theta}(k-1)} \tag{5.44}$$

by using the chain rule.

Where

$$\left.\frac{\partial \hat{y}(k)}{\partial h(k)}\right|_{\theta=\hat{\theta}(k-1)} = \begin{bmatrix} \hat{w}^{y}_{1,1}(k-1) & \cdots & \hat{w}^{y}_{1,q}(k-1) \\ \vdots & & \vdots \\ \hat{w}^{y}_{p,1}(k-1) & \cdots & \hat{w}^{y}_{p,q}(k-1) \end{bmatrix} \tag{5.45}$$

and

$$\frac{\partial h(k)}{\partial z(k)} = \left[\begin{matrix} \frac{\partial h_1(k)}{\partial z_1(k)} & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & \frac{\partial h_q(k)}{\partial z_q(k)} \end{matrix}\right]_{\theta=\hat{\theta}(k-1)} \tag{5.46}$$

This is a diagonal matrix.

Where

$$\left.\frac{\partial h_i(k)}{\partial z_i(k)}\right|_{\theta=\hat{\theta}(k-1)} = h_i(k)\big|_{\theta=\hat{\theta}(k-1)}\left[1-h_i(k)\big|_{\theta=\hat{\theta}(k-1)}\right], i=1,\ldots,q \tag{5.47}$$

$$h_i(k)\big|_{\theta=\hat{\theta}(k-1)} = \frac{1}{1+\exp[-z_i(k)]\big|_{\theta=\hat{\theta}(k-1)}} \tag{5.48}$$

$$z_i(k)\big|_{\theta=\hat{\theta}(k-1)} = \hat{W}_i^{h}(k-1)\begin{bmatrix} x(k) \\ 1 \end{bmatrix} \tag{5.49}$$

From Eq. (5.44) the term $\left.\frac{\partial \hat{y}(k)}{\partial \theta^h}\right|_{\theta=\hat{\theta}(k-1)}$ can be further expressed as

$$\left.\frac{\partial z(k)}{\partial \theta^h}\right|_{\theta=\hat{\theta}(k-1)} = \begin{bmatrix} [x(k)^T & 1] & \cdots & 0_{1x(n+1)} \\ \vdots & & & \vdots \\ 0_{1x(n+1)} & & \cdots & [x(k)^T \quad 1] \end{bmatrix}_{qxq(n+1)} \tag{5.50}$$

From Eq. (5.43), the term $\left.\frac{\partial \hat{y}(k)}{\partial \theta^y}\right|_{\theta=\hat{\theta}(k-1)}$ can be expressed as

$$\left.\frac{\partial \hat{y}(k)}{\partial \theta^y}\right|_{\theta=\hat{\theta}(k-1)} = \frac{\partial \hat{W}^y(k-1)\begin{bmatrix} h(k)|_{\theta=\hat{\theta}(k-1)} \\ 1 \end{bmatrix}}{\partial \theta^y} \tag{5.51}$$

$$\left.\frac{\partial \hat{y}(k)}{\partial \theta^y}\right|_{\theta=\hat{\theta}(k-1)}$$

$$= \begin{bmatrix} [h^T(k)|_{\theta=\hat{\theta}(k-1)} & 1] & \cdots & 0_{1x(q+1)} \\ \vdots & & & \vdots \\ 0_{1x(q+1)} & & \cdots & [h^T(k)|_{\theta=\hat{\theta}(k-1)} \quad 1] \end{bmatrix}_{pxp(q+1)} \tag{5.52}$$

where

$0_{1x(q+1)} = [0 \quad \cdots \quad 0]$ is a row vector of $(q+1)$ zeros.

Derivation of the EKF-based on-line learning algorithm for MLPN is described in detail in Appendix B.

5.5 Neural Network Training Procedure

A neural network is an information processing system that is non-algorithmic, non-digital and intensely parallel. Neural networks learn to solve problem; they are not programmed to do so. Hence, neural networks have been applied to the problems where domain knowledge is scarce but numerical data can be acquired easily. Multilayer neural network can learn from data. However, if the trained network does not perform as we have expected, we need to retrain the network changing the network size, the number of hidden neurons or adding new training data. In addition, since the training is slow, repetition of training is prohibitive especially for a large size network with a large number of training data. In order to speed up training and avoid repetition on the same training data, we explore the use of neural network ensemble method where each network is trained dynamically, pattern by pattern, and has the ability to learn new patterns even when it is already in use by clinicians. The ensemble method integrated with the use of online learning provides great benefits not only in terms of classification performance but also in providing clinicians with added flexibility in the way they use the tool. Neural network ensembles adopt the divide-and-conquer strategy. Instead of using a single network to solve a task, a neural network ensemble combines a set of neural networks which learn to subdivide a task and thereby solve it more efficiently and elegantly. An ECG signal is considered as NSR as long as all the P, Q, R, S, T waves are present in the waveform and the heart rate is normal i.e. between 60-80 cycle per minute for an adult. Hence, the NSR class covers a broad range of ECG signals. Thus, we utilise an ensemble of neural networks consisting of three neural networks cascaded in parallel to distinguish the various NSR signals from the various abnormal ECG signals such as STDP, TINV and SVT. The first MLPN with be trained to distinguish between the first form of NSR signal from STDP, the second MLPN to distinguish between the second form of NSR signal from TINV, and the

third MLPN to distinguish between the third form of NSR signal from SVT. After these initial clustering of ECG data, a fuzzy inference system is used to further classify these substructures into the appropriate classes as will be described in Section 7.1 with Figure 7.1.

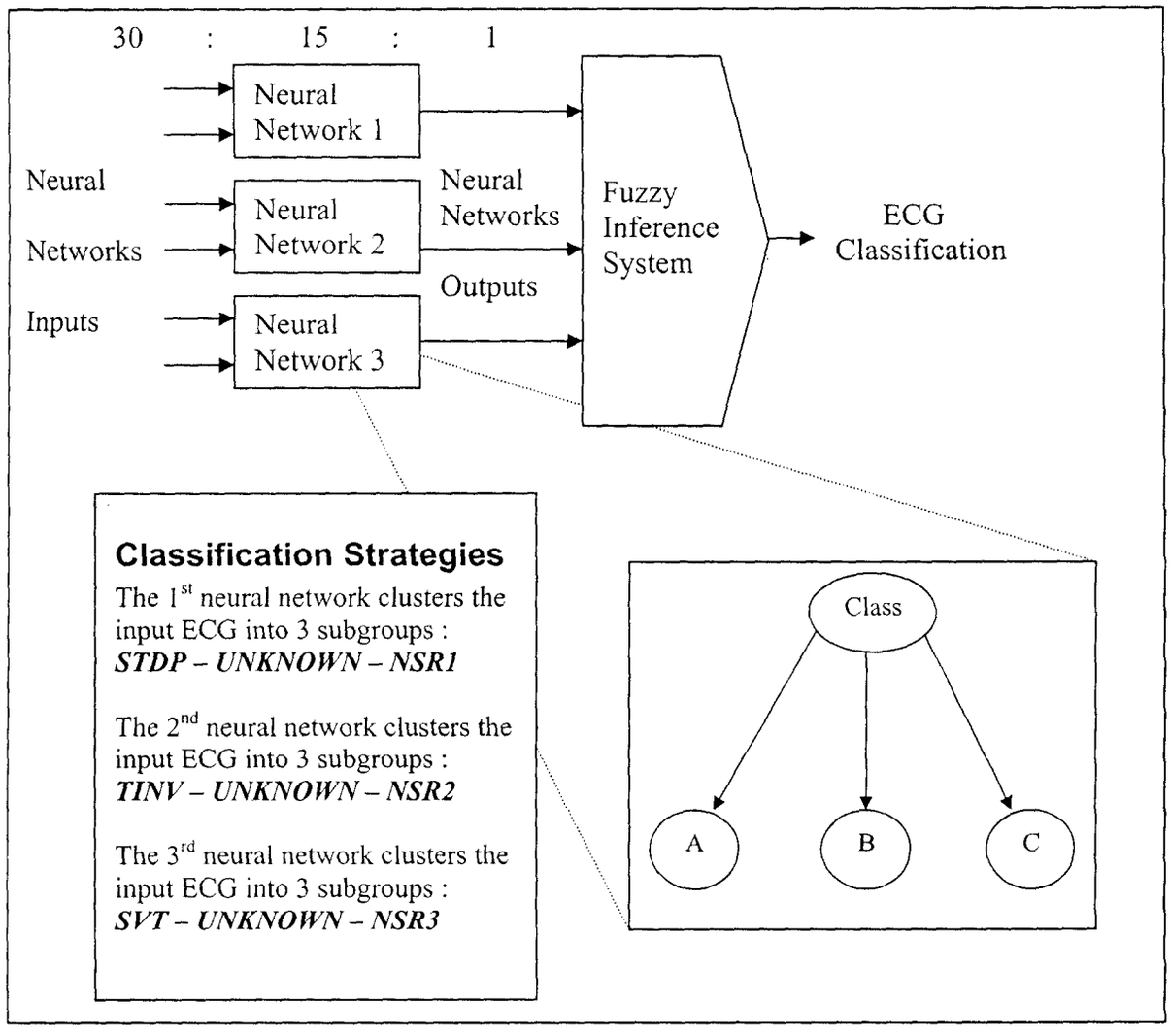

Figure 5.2: Fuzzy inference of the neural networks outputs.

As shown in Figure 5.2, each neural network has a structure of 30-15-1, i.e. it has three layers of neurons: the 1st layer which is called the input layer has 30 neurons, the 2nd layer which is called the hidden layer has 15 neurons and the 3rd layer which is called the output layer has single neuron. The input raw signal was initially compressed using DWT [8] since the digitized ECG data were no longer continuous variables. The main objective

behind using wavelet-compression was to increase computational speed. The actual input to the network was obtained by passing a sampling window with a length of 30 time steps over the signal. At each new iteration, the window will be shifted forward by one time-step. The number of hidden neurons that maximized accuracy and generalization were determined experimentally. The experimental study begins with 5 hidden neurons as the starting point. Since the extra neuron will increase the non-linearity of the solution, therefore a larger number of hidden neurons can be used to extract a more complicated feature. However, if too many neurons are used, the resulting minimized mean square error may be very small compared to what a smaller number of hidden neurons gives, but this also means that the noise has been included in the solution and that is undesirable. Therefore, we increase the number of hidden neurons by 5 units each time the solution diverges, until finally the network converges [72-75]. A well-designed neural network will exhibit good generalisation when a correct input-output mapping is obtained even when the input is slightly different from the examples used to train the network [74, 75]. In designing a neural network, i.e. the MLPN, the designer must make a number of choices with regard to the system architecture: what is the appropriate number of hidden layers to be included? How many nodes should each layer have? Unfortunately, there is a shortage of evidence available to designers that would enable them to make specific design choices based on a clear scientific rationale and as a consequence, many designs are made on the basis of trial and error. There are other design issues associated with the level and extent of training required for such a network, in particular locating the point at which the network is considered to be sufficiently trained. Conventional methods of training MLPN involve a process whereby the network is trained to the point of minimum error based upon the training data. Subsequently, the neural network's internal parameters are fixed and it is tested with unseen data to evaluate its performance. This has been the most common

approach in the development of neural network based ECG classifiers [76-78]. A danger exists with this approach in that the neural network, during training, may memorise the training data. If this becomes the case, then the neural network may be biased towards the training data and hence not fully represent the underlying function that is to be modelled. In such instances, poor generalisation is attained when unseen data is presented as input to the neural network. Such a phenomenon is referred to as over-fitting. Hence, it is possible to over-fit the neural network if training is not stopped at the correct point.

We utilize an ensemble of neural network consisting of three MLPN that can only learn from numerical data, there is no utilization of linguistic rules as developed by human experts. Hence, the fuzzy inference system plays the role of extracting rules through 'observation' of the outputs from the ensemble of neural network. This kind of learning through 'observation' is accomplished using the table-lookup training scheme as described in Chapter 6. This learning scheme is simple and straightforward in the sense that it is a one-pass build-up procedure that does not require time-consuming training as the MLPN.

5.6 Conclusions

A neural network ensemble offers several advantages over a monolithic neural network. First, it can perform more complex tasks than any of its components (i.e., individual neural networks in the ensemble). Secondly, it can make an overall system easier to understand and modify. Finally, it is more robust than a monolithic neural network and can show graceful performance degradation in situations where only a subset of neural networks in the ensemble is performing correctly. Given the advantages of neural network ensembles and the complexity of the problems that are beginning to be investigated, it is clear that the neural network ensemble method will be an important and pervasive problem-solving technique [74, 75].