4.4 Introduction to special neural network architectures

4.4.1 Determining the optimal Structure

❶ Empirical Formula

The number of nodes in the hidden layer:

$$k = \sqrt{0.43sn + 0.12n^2 + 2.54s + 0.77n + 0.35} + 0.5$$

s: No. of nodes in the input layer

n: No. of nodes in the output layer

❷ Identifying redundant node

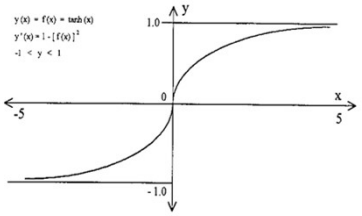
$\delta_i = E_{\text{without node i}} - E_{\text{with node i}}$

❸ **Structure optimization**

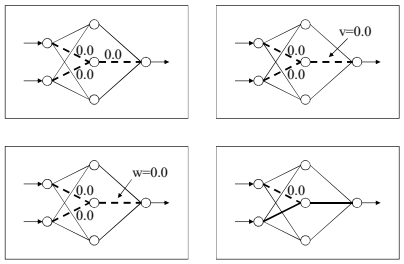
1. Selecting transfer function

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$f(x) = \frac{1}{1+e^{-x}} - 0.5$$



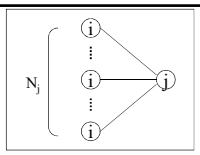
Redundant nodes and connections



2. Objective function

$$E_T = E_{BPN} + \lambda E_{COM}$$

$$E_{COM} = \sum_j N_j \sum_{i=1}^{N_j} w_{ij}^2$$



$$\Delta w_{ij} = -\eta \frac{\partial E_T}{\partial w_{ij}} = -\eta \frac{\partial E_{BPN}}{\partial w_{ij}} - \eta\lambda \frac{\partial E_{COM}}{\partial w_{ij}} = \eta \delta_j^k v_i^{k-1} - 2\eta\lambda N_j w_{ij}$$

$$\Delta w_{ij} = \eta \delta_j^k v_i^{k-1} - \gamma N_j w_{ij}$$

$$where, \quad \gamma = 2\eta\lambda$$

$$\gamma = \gamma_0 e^{-E_{BPN}} e^{-|w_{ij}|}$$

Adding momentum item

$$\Delta w_{ij}(n+1) = \eta \delta_j^k v_i^{k-1} + \alpha \Delta w_{ij}(n) - \gamma N_j w_{ij}(n)$$

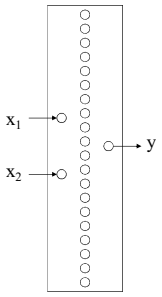
Example 1

$x_1 = 0.5x + 4.57$

$x_2 = 2.5x + 1.23$

$y = x_1 + x_2$

① Structure 2-20-1

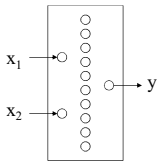


Record of deleting the redundant node

Iterations	900	901	912	913	918	920	924	996
The number of nodes in hidden layer	20	18	17	16	14	13	11	10
No. of deleted nodes		Mid 19,20	Mid 13	Mid 4	Mid 14,17	Mid 18	Mid 15,16	Mid 7

Iterations	1273	1747	2183	2269	2406	2900	3507
The number of nodes in hidden layer	9	8	7	6	5	3	2
No. of deleted nodes	Mid 8	Mid 12	Mid 11	Mid 10	Mid 9	Mid 1,6	Mid 2

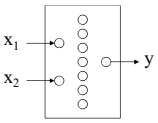
② Structure 2-10-1



Record of deleting the redundant node

Iterations	500	528	554	724	756	953	1075	1839	3618
The number of nodes in hidden layer	10	9	8	7	6	5	4	3	2
No. of deleted nodes		Mid 8	Mid 9	Mid 6	Mid 5	Mid 7	Mid 3	Mid 10	Mid 4

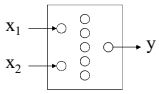
③ Structure 2-7-1



Record of deleting the redundant node

Iterations	1000	1042	1105	1114	1143	1159
The number of nodes in hidden layer	7	6	5	4	3	2
No. of deleted nodes		Mid 5	Mid 4	Mid 6	Mid 7	Mid 2

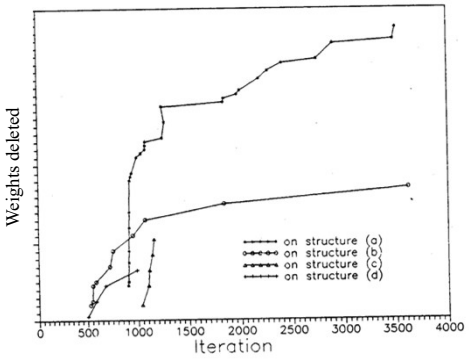
④ Structure 2-5-1



Record of deleting the redundant node

Iterations	500	582	680	992
The number of nodes in hidden layer	5	4	3	2
No. of deleted nodes		Mid 3	Mid 5	Mid 4

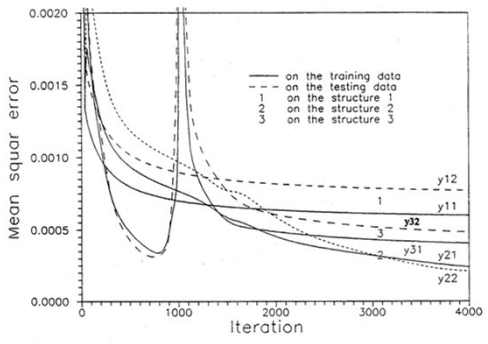
Weights deleted for four structures

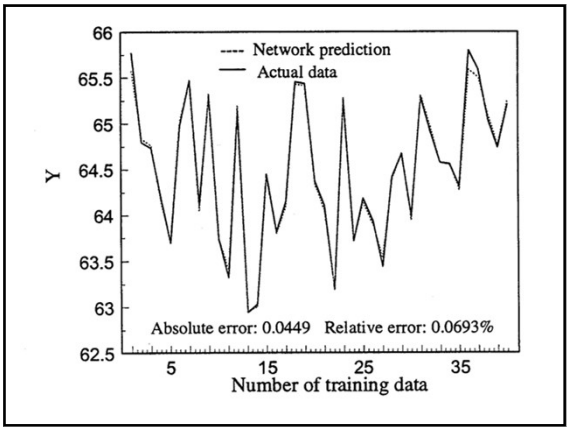


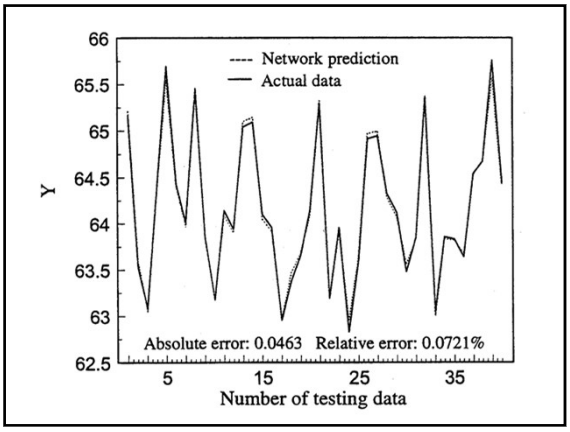
Compare results with three structures (2-1-1, 2-2-1, 2-3-1)

No. of nodes in hidden layer		1	2	3
Iteration		8000	4000	4000
Training set	MSE	0.000578	0.000240	0.000404
	Relative error	0.1143	0.0693	0.0885
	Absolute error	0.0736	0.0449	0.0571
Testing set	MSE	0.000744	0.000211	0.000483
	Relative error	0.1364	0.0721	0.1023
	Absolute error	0.0872	0.0463	0.0655

Training process







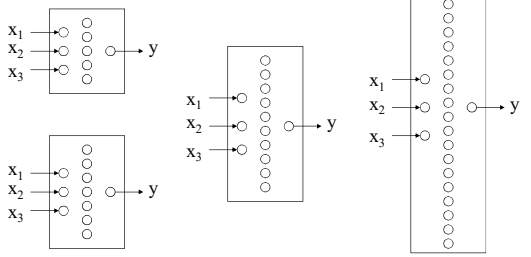
Example 2

x_1=0.5x+4.57

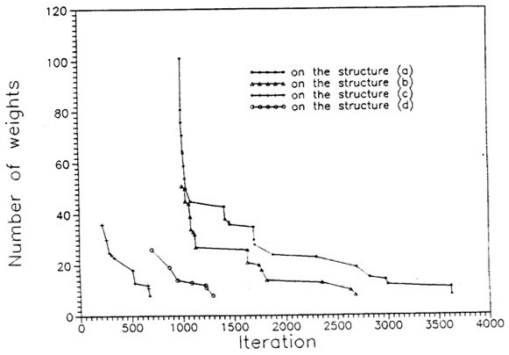
x_2=2.5x+1.23

x_3= noise (random value)

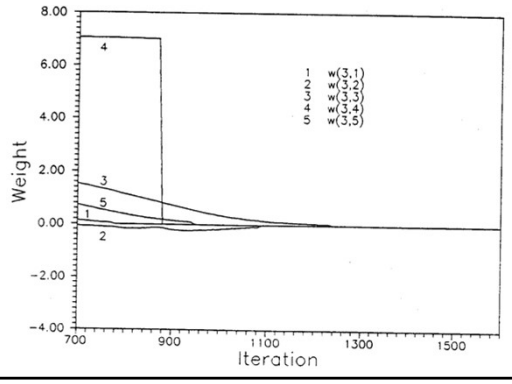
$y=x_1+x_2$



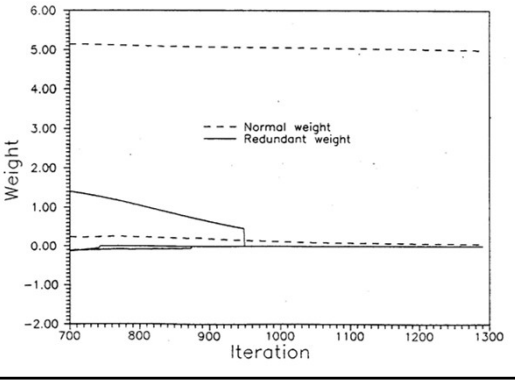
Weights deleted for four structures



Changing of the weights connected to input node 3 (structure 3-5-1)

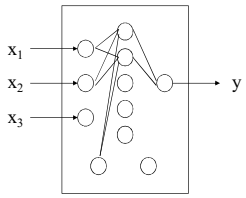


Weights changing curve



Example

Final structure 1



Record for redundant weights deleted with different weight deadlines

W_0=±0.001		W_0=±0.005		W_0=±0.01		W_0=±0.05		W_0=±0.1	
Iteration	Weights deleted	Iteration	Weights deleted	Iteration	Weights deleted	Iteration	Weights deleted	Iteration	Weights deleted
771	1	768	1	765	1	742	1	700	1
904	2	886	2	866	2	775	2	716	2
956	3	953	3	949	3	874	7	723	3
1292	4	1083	4	1027	4	948	12	780	8
1339	5	1167	5	1109	5	1088	13	855	13
1378	8	1206	6	1147	6	1223	14	1002	18
1460	9	1223	9	1150	9	1231	15		
1523	14	1410	10	1287	13	1290	18		
1532	18	1440	14	1372	14				
		1515	18	1492	18				

Record for redundant weights deleted with different γ_0

γ_0=0.005		γ_0=0.0005		γ_0=0.0001		γ_0=0.00005	
Iteration	Weights deleted	Iteration	Weights deleted	Iteration	Weights deleted	Iteration	Weights deleted
700	1	704	1	742	1	766	1
701	8	713	3	775	2	901	2
703	14	738	8	874	7	1097	7
1922	15	740	9	948	12	1114	12
2880	16	901	10	1088	13	1281	13
2883	17	1480	11	1223	14	1629	14
3305	18	1509	12	1231	15	1654	15
		2082	15	1290	18	5083	18
		2517	16				
		3064	18				

Record for redundant weights deleted with different E_0

E_0=0.005		E_0=0.0005		E_0=0.0001		E_0=0.00005	
Iteration	Weights deleted	Iteration	Weights deleted	Iteration	Weights deleted	Iteration	Weights deleted
765	1	765	1	765	1	765	1
866	2	866	2	866	2	866	2
949	3	949	3	949	3	949	3
1140	4	1027	4	1027	4	1027	4
1231	5	1109	5	1109	5	1109	5
1533	6	1147	6	1147	6	1147	6
3020	7	1887	9	1150	9	1150	9
3166	8	1903	10	1287	13	1287	13
3968	9	1904	11	1372	14	1372	14
4127	11	1981	14	1492	18	1492	18
6422	14	2025	18				
6597	18						
10000	**18**	**4000**	**18**	**4000**	**18**	**4000**	**18**

4.4.2 Autoassociative Networks

Two main applications:

(1) Data compression and filtering

(2) Dimensionality reduction of an input vector

Two main categories:

(1) Signal-processing networks

(2) Image-processing networks

The most commonly used NN for signal processing is the autoassociative backpropagation network

- reduce the amount of random noise in the input signal
- filter gross errors in data resulting from sensor-malfunctions, drift and bias.

The input signal has three distinct components:

(1) A pure signal (p)

(2) Measurement noise (n)

(3) Measurement error (e)

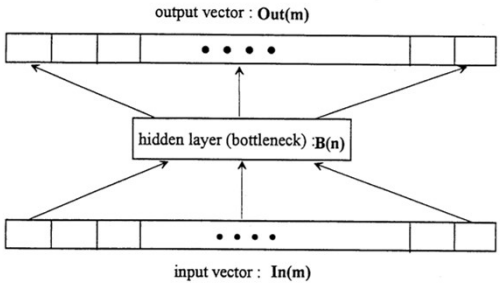
Thus, we may represent the input vector as a linear combination of the three components:

$$In(m)=In_p(m)+In_n(m)+In_e(m)$$

The objective of filtering a process signal is to map only the pure signal ($In_p(m)$) onto the output vector, while removing all measurement noise ($In_n(m)$) and measurement errors ($In_e(m)$).

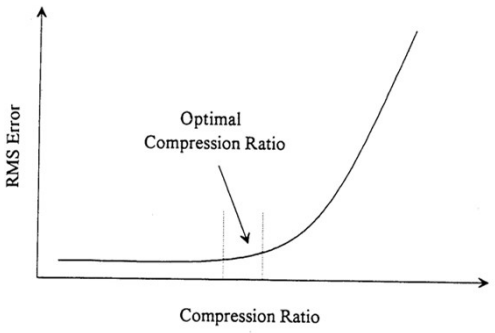
$$In(m)=Out(m)$$

A three-layer autoassociative network



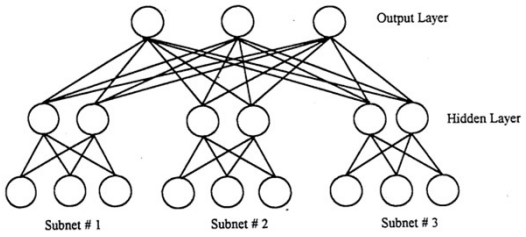
Compression ratio = $\frac{m}{n}$

Typical ratios are normally set between 2 and 8, depending on the signal type and noise in the system



4.4.3 Hierarchical Neural Networks

The input vector is divided into groups that have similar effects on the output responses



A hierarchical NN architecture

Advantages:

(1) use more nodes with fewer weight factors, improving efficiency. Therefore, network training requires fewer examples and less time.

(2) well-defined subnetworks of related variables provides hints that help the network learn in the right direction.

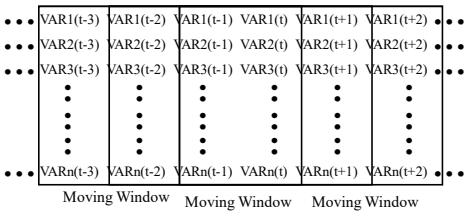
(3) isomorphic to expert systems or model-based algorithms for the same task. We can map each useful subnetwork to a rule (or small set of rules) or a model-based local analysis.

The following sections describe two significant types of hierarchical networks used in process system engineering.

(1) moving-window networks for time-dependent processes

(2) input-compression networks for working with large input variable sets

1: Moving-Window Networks



two types of applications:

- model dynamic systems
- model unsteady-state processes

Key challenge problems:

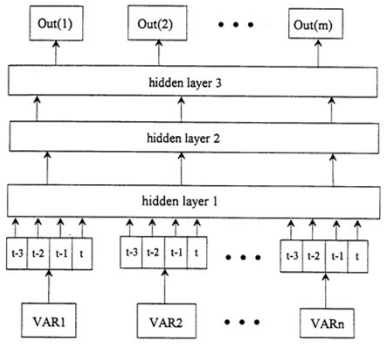
(1) Choose the " time-window width "

(2) Choose the time between data sampling point

(3) Use average moving-window values as input to the network

$$\overline{VAR_j}(t) = \frac{\sum_{n=0}^{3} VAR_j(t - n\Delta t)}{4}$$

A moving-window network with a process-trend scanning window including 4 time increments



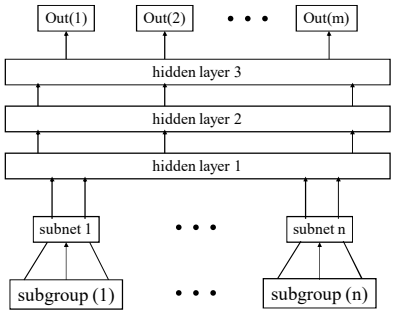
2: Input-Compression Networks

Neural networks can become very large and difficult to train if they have a large number of input variables that influence the output responses.

Auto-assoicate hierarchical neural network

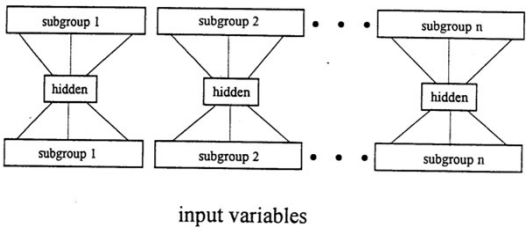
The input layer in the input-compression network consists of n property groups, where each group includes input variables with similar effects on the output responses. Each property group is compressed into a corresponding subnetwork to create a more compact representation of the input variables without losing any essential information.

A input-compression network



An input-compression network divided into a lower section containing a dimensionality-reduction network and an upper section containing a standard backpropagation network.

* Attaching a Dimensionality-Reduction Network to a Backpropagation Network

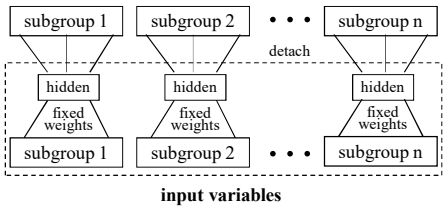


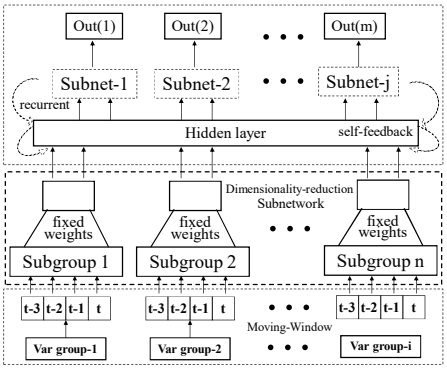
Developing an input-compression network involving two steps.

(1) use an autoassociative network to reduce the dimensionality of the input vector.

(2) attach the input and hidden layers of the dimensionality-reduction network to a standard backpropagation network.

* Dimensionality Reduction of Input Vector





Example:

Feature Clustering-FC

Feature Target Correlation -FTC

Data Attributes Decompostion method

DATASETS FROM UCI RESPOSITORY AND PRODUCTION

Dataset Name	Dataset Character Quality			
	Instances	***Attributes***	***Outputs***	***Default Task***
Ionosphere	351	34	1	Classification
Breast Cancers	569	30	1	Regression /Classification
PTA	260	17	1	Regression

FEATURE SETS ENSEMBLES FROM TEST DATA SETS

Dataset Name	Feature Sets Ensemble Steps	
	Feature Target Correlation	***Feature Clustering***
Ionosphere	{22,27,30,32,34} {1,2…20,21,23,24,25,26,28,29,31,33}	{2,18,20,24,26,28} {1,3,5,7,9,29,31} {22,27,30,32,34} {4,6,8,10,12,14,16,33} {11,13,15,17,19,21,23,25}
Breast Cancers	{10,12,15,19} {1,2,3,4,5,6,7,8,9,11,13,14,16,17,18,20…30}	{3,23} {4,24} {10,12,15,19} {1,2,5,7,8…22,25…30}
PTA	{3,4,8,10,13} {1,2,5,6,7,9,11,12,14,15,16,17}	{7,9,11,12,14} {15,16,17} {1,2,5,6} {3,4,8,10,13}

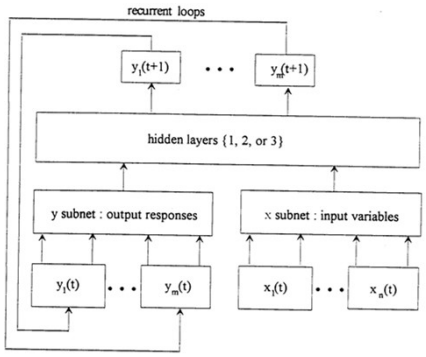
Comparison results

Dataset Name	Methods Comparison(Err/Sta)		
	BP	***DAD-AHNN***	***DAD-THNN***
Ionosphere	8.1468	3.9423	4.4179
	0.2669	0.1239	0.1168
Breast Cancers	5.1059	2.4075	4.0621
	0.1665	0.1348	0.1568
PTA	0.4426	0.4247	0.3609
	0.0037	0.0033	0.0030

AHNN--Auto-assoicate Hierarchical Neural Network

THNN--Treetyped Hierarchical Neural Network

4.4.4 Recurrent Networks



Definition:

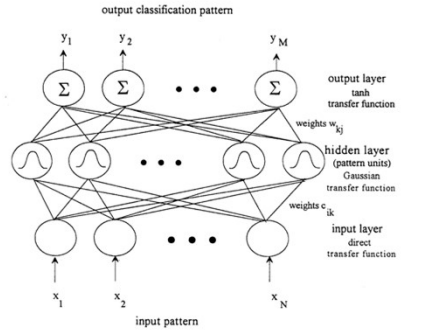
A recurrent network combines the feedback and feedforward connections of neural networks. It is simply a neural network with loops connecting the output responses to the input layer.

Applications:

The recurrent network provides a modeling technique that can be used for process optimization and adaptive process control of time-dependent processes.

4.4.5 Radial-Basis-Function Networks

1. Network Architecture



(1) Input Layer

The output of the node equals the input.

(2) Hidden Layer

Its nodes satisfy the unique property of being ***radially symmetric.***

(a) A center vector $\mathbf{c_k}$ in the input space, made up of cluster center, with elements c_{ik}(i=1 to N). The vector is typically stored as the weight factors from the input layer to the hidden layer.

(b) A distance measure to determine how far an input vector **x**, with elements x_i(i=1 to N), is from the center vector c_k. We use the standard Euclidean distance measure between x and $\mathbf{c_k}$ to define a Euclidean summation, I_k(k=1 to L), where L is the number of nodes in hidden layer.

$$I_k = \|x - c_k\| = \sqrt{\sum_{i=1}^{N} (x_i - c_{ik})^2}$$

(c) A transfer function which transforms the Euclidean summation I_k (k=1 to L) to give an output for each node.

$$v_k = e^{\frac{\|x - c_k\|^2}{2\sigma_k^2}}$$

Where Sigma σ_k is a width of the Gaussian function

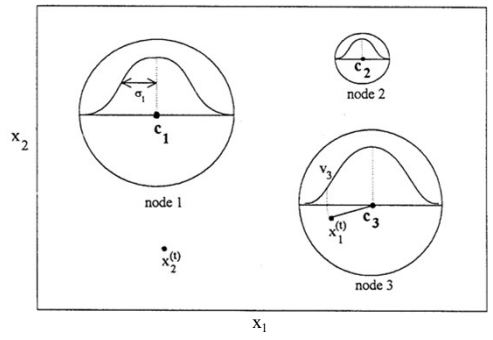
(3) Output Layer

There are weight factors w_{kj} (k=1 to L; j=1 to M) between the k^{th} node in the hidden layer and the j^{th} node in the output layer.

$$y_j = \sum_{k=1}^{L} w_{kj} v_k \qquad j = 1 \ \ to \ \ M$$

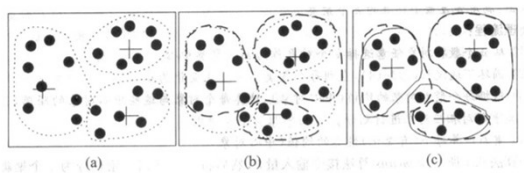
2. Network Development

(1) Cluster Centers $\mathbf{c_k}$



Using K-means clustering algorithm

- We start by assuming an initial set of L cluster centers, i.e.,center vector $\mathbf{c_k}$, which corresponds to the L nodes in the hidden layer.
- The elements, c_{ik}(i=1 to N; k=1 to L) of $\mathbf{c_k}$ are stored as the weight factors between the input and hidden layers.
- Assume that there are T training examples available to the input layer with N nodes, and represent them by training vectors $\mathbf{x^{(t)}}$ with $x_i^{(t)}$ (i=1 to N; t=1 to T).
- The adaptive K-means clustering algorithm then iteratively finds a desirable set of L center vectors $\mathbf{c_k}$ that will minimize the sum of the squares of the distance between T training vectors $\mathbf{x^{(t)}}$ and their nearest L centers $\mathbf{c_k}$(k=1 to L).



Example of K-means clustering algorithm

a. We arbitrily choose three objects as the three initial cluster centers, where clusters are marked by a "+". Each object is distributed to a cluster based on the cluster center to which it is the nearest.

b. The mean value of each cluster is recalculated based on the objects in the cluster. Relative to these new centers, objects are redistributed to the cluster domains based on which cluster center is the nearest.

c. This process iterate, leading to Figure ©. Eventually, no redistribution of the objects in any cluster occurs and so the process terminates.

(a) Initialization step

- Assume a set of cluster centers, $\mathbf{c_k}$, for the L nodes in the hidden layer, with the elements c_{ik} (i=1 to N; k=1 to L) stored as the weight factors between the input and hidden layers.

(b) Iterative Steps:

- Read the next training vector $\mathbf{x^{(t)}}$ with elements $x_i^{(t)}$ (i=1 to N) into the input layer as t increases from 1 to T.
- Modify only the cluster center $\mathbf{c_k}$ (k=1 to L) closest to the training vector $x^{(t)}$ in the Euclidean distance:

$$c_k^{(new)} = c_k^{(old)} + \alpha(x^{(t)} - c_k^{(old)})$$

where α is a learning rate which decreases as t increases from 1 to T.

- Continue this process for a fixed number of iterations.

(2) Gaussian Function, Sigma σ_k

Using P-nearest neighboring heuristic.

- consider a given center vector $\mathbf{c_k}$ (k=1 to L)
- assume that $c_{k1}, c_{k2}, \ldots, c_{kp}$ ($1 \le k_1, k_2, \ldots k_p \le L$) are the P nearest neighboring centers.

$$\sigma_k = \sqrt{\frac{1}{P}\sum_{p=1}^{P}\left\|c_k - c_{kp}\right\|^2}$$

(3) The resulting output from the k^{th} node in the hidden layer, v_k, is:

$$v_k = e^{\frac{\|x-c_k\|^2}{2\sigma_k^2}}$$

(4) Weight factors, w_{ki}

$$y_j = \sum w_{kj} v_k - T_j$$

$$y_j = f\left(\sum w_{kj} v_k - T_j\right)$$

Using BP algorithm:

$$w_{kj}(t+1) = w_{kj}(t) + \beta(y_j^d - y_j)v_k$$

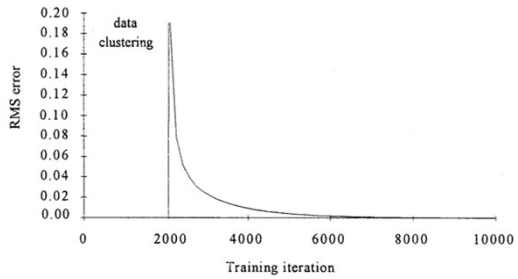
(5) Network training

First, do not use any desired output to train the input connects to the hidden layer.

Second, for any given training example expressed by the t^{th} training vector $x^{(t)}$ with elements $x_i^{(t)}$ (i=1 to N), we iteratively modify only one cluster center c_k (k=1,2,…, or L) closest to the t^{th} training vector $x^{(t)}$ in the Euclidean distance.

Over this same time period, no training of the weight factors, w_{kj} , from hidden layer to output layer, occurs. We call this initial period the ***data-clustering phase***.

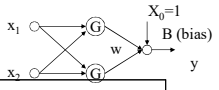
For example, after the 2000^{th} iteration, the weight factors, c_{ik}, are fixed for the remainder of the training procedure. The weight factors w_{kj} are then trained using a conventional backpropagation algorithm.



XOR Problem

Gaussian Function

$$G(\|x-c_i\|) = \exp(-\|x-c_i\|^2) \quad i=1,2$$



Assuming $c_1=[1,1]^T$, $c_2=[0,0]^T$, b=1

$$v_{ik} = G(\|x_i - c_k\|) \quad i=1,2,3,4; \quad k=1,2$$

$$v_{11} = e^{-(x_1^1-c_{11})^2-(x_2^1-c_{21})^2}$$

$$= e^{-(1-1)^2-(1-1)^2} = 1$$

$$v_{12} = e^{-(x_1^1-c_{12})^2-(x_2^1-c_{22})^2}$$

$$= e^{-(1-0)^2-(1-0)^2} = 0.1353$$

Input	Desired output
(1,1)	1
(0,1)	0
(0,0)	1
(1,0)	0

$$G = \begin{bmatrix} v_{11} & v_{12} & x_0 \\ v_{21} & v_{22} & x_0 \\ v_{31} & v_{32} & x_0 \\ v_{41} & v_{42} & x_0 \end{bmatrix} = \begin{bmatrix} 1 & 0.1353 & 1 \\ 0.3678 & 0.3678 & 1 \\ 0.1353 & 1 & 1 \\ 0.3678 & 0.3678 & 1 \end{bmatrix}$$

Assuming $\mathbf{w}=[w,w,b]^T$

$\mathbf{G}\mathbf{w}=\mathbf{d}$

$d=[1,0,1,0]^T$

$$w = \begin{bmatrix} 2.284 \\ 2.284 \\ -1.692 \end{bmatrix}$$

Input	Desired output	Network output
(1,1)	1	0.901
(0,1)	0	0.01
(0,0)	1	0.901
(1,0)	0	-0.01

4.4.6 Input Training Neural Network

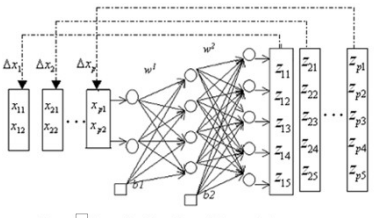


Figure Input Training Neural Network Structure

Let: t_{pk} is the value of the kth observed variable in the pth sample,
z_{pk} is the corresponding output of IT-NN.

The aim of training IT-NN is
to minimize the following objective function:

$$\min E(X,W) = \min \sum_p \sum_k (Z_{pk} - t_{pk})^2$$

The steepest descent direction for optimizing network inputs ΔX_{pi}

$$\Delta X_{pi} = -\frac{\partial E}{\partial X_{pi}} = \sum_k (t_{pk} - Z_{pk}) \frac{\partial Z_{pk}}{\partial X_{pi}}$$

The steepest descent direction for optimizing network weights ΔW_{ji} is

$$\Delta W_{ji} = -\frac{\partial E}{\partial W} = \sum_k (t_{pk} - Z_{pk}) \frac{\partial Z_{pk}}{\partial W_{ji}}$$

The network output is given by

$$Z_{pk} = \sigma(f_k + \sum_j w_{kj}^2 \sigma(b_j + \sum_i w_{ji}^1 X_{pi}))$$

where $\sigma(.)$ is sigmoidal function,
b_j, f_k are the bias of the jth hidden node and kth output node,
w_{kj}, w_{ji} are the weights of IT-NN.

The steepest descent direction for training network is

$$\Delta X_{pi} = \sum_j w_{ji}^1 \delta_{pj}$$

where δ_{pj} is the propagated error at the hidden layer.

$$\delta_{pj} = \sigma'(b_j + \sum_i w_{ji}^1 X_{pi}) \bullet$$
$$\sum_k w_{kj}^2 (t_{pk} - Z_{pk}) \sigma'[f_k + \sum_j w_{kj}^2 \sigma(\sum_i w_{ji}^1 X_{pi})]$$

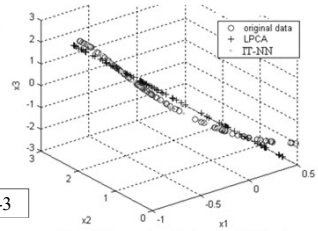
Weight adjusting is given by

$$\Delta w_{ji} = \sum_p X_{pi} \delta_{pj}$$

Effectiveness of IT-NN

Consider a nonlinear system of flowing,

$$x_1 = 0.5t^2 - 2t + 0.5 + e_1$$
$$x_2 = t^2 + t + \sin \pi t + e_2$$
$$x_3 = 2t^2 - t - 2\cos \pi t + e_3$$



网络结构为1-7-3

Figure 3. Compare the effectiveness of LPCA and IT-NN when analyzing nonlinear system

Nonlinear System Modeling with IT-NN & RBF-NN

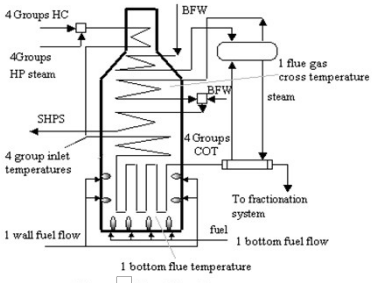


Figure Cracking Furnace

There are 20 factors that directly or indirectly affect the yields of ethylene and propylene

Combining IT-NN with RBF

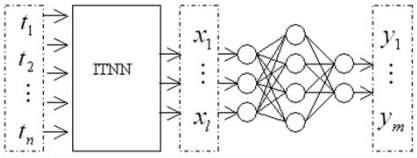


Figure Combine ITNN with RBF to model nonlinear system

Results

Num. Of Input nodes	10	9	8	7	6	5
e_{max} (%)	2.1432	3.4599	2.7845	3.6290	4.3221	5.8745
e_{min} (%)	0.0007	0.0030	0.0051	0.0033	0.0065	0.0103
e_{mse} (%)	0.0923	0.1065	0.1565	0.1903	0.5329	1.3785

Table 1. Reconstruction errors with different input layer nodes of IT-NN but same hidden layer nodes

Num. of nonlinear PCs			10	9	8	7	6	5
Training period	C_2H_4	e_{max}	0.1055	0.1771	0.1730	0.2036	0.7814	0.9631
		e_{min}	0.0000	0.0001	0.0007	0.0010	0.0105	0.0489
		e_{mse}	0.0106	0.0130	0.0144	0.0555	0.1001	0.3901
	C_3H_6	e_{max}	0.2373	0.3672	0.3829	0.6300	0.8100	1.485
		e_{min}	0.0002	0.0001	0.0005	0.0023	0.0090	0.1068
		e_{mse}	0.0835	0.0902	0.0995	0.1300	0.1689	0.2155
Predicting period	C_2H_4	e_{max}	2.7501	2.2456	2.3780	2.4998	2.7599	5.1613
		e_{min}	0.0095	0.0092	0.0089	0.0104	0.0196	0.1079
		e_{mse}	0.5933	0.6318	0.6770	0.8317	0.8965	1.4086
	C_3H_6	e_{max}	3.0477	3.7034	3.9861	4.1960	4.3435	6.7936
		e_{min}	0.0104	0.0108	0.0153	0.0230	0.0267	0.023
		e_{mse}	0.6053	0.6539	0.6900	0.90	1.09	1.7540

Table 2. Training and predicting errors with different PCs in IT-NN+RBF-NN model

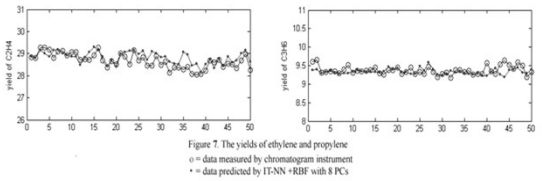
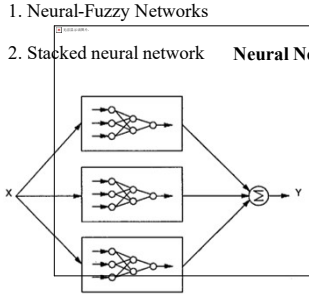


Figure 7. The yields of ethylene and propylene
o = data measured by chromatogram instrument
• = data predicted by IT-NN +RBF with 8 PCs

We can see that the IT-NN with 8 nonlinear PCs will be the best choice. Fewer PCs (7, 6 and 5) cause information lost, and more PCs (10 and 9) increase the input nodes of RBF.

4.4.7 other Networks

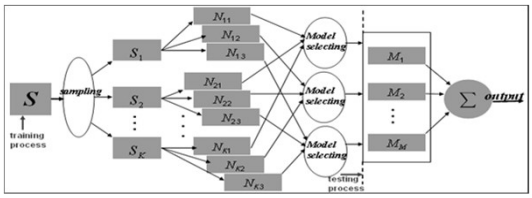
1. Neural-Fuzzy Networks
2. Stacked neural network **Neural Network Ensemble**



$$\hat{Y} = [\hat{y}_1, \hat{y}_2 \cdots \hat{y}_n]$$

$$\hat{Y} = \hat{Y} w = w_1 \hat{y}_1 + w_2 \hat{y}_2 + \cdots + w_n \hat{y}_n$$

Diversity measurement using network **W**eight based **S**elective **EN**semble



Network structure of DWSEN

Weight Vector

$\vec{D}_i = (w_{i1}, w_{i2}, \ldots, w_{iH})$ All weights in the ith network

Distance between Ni and Nj:

$$d(N_i, N_j) = d(\vec{D}_i, \vec{D}_j) = \left\| \vec{D}_i - \vec{D}_j \right\|_2 = \sqrt{\sum_{h=1}^{H} (w_{ih} - w_{jh})^2}$$

Weight Matrix

$$\begin{bmatrix} \vec{D}_1 \\ \vec{D}_2 \\ \cdots \\ \vec{D}_N \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1H} \\ w_{21} & w_{22} & \cdots & w_{2H} \\ \cdots & \cdots & \cdots & \cdots \\ w_{N1} & w_{N2} & \cdots & w_{NH} \end{bmatrix}_{N*H}$$

Definition of the diversity

$$c = dif(N_i, N_j) = dif(\vec{D}_i, \vec{D}_j) = \frac{\left\| \vec{D}_i - \vec{D}_j \right\|_2}{\left\| \vec{D}_i \right\|_2 + \left\| \vec{D}_j \right\|_2}$$

（0≤c≤1）

Diversity Matrix

$$\begin{bmatrix} \vec{dif}_1 \\ \vec{dif}_2 \\ \cdots \\ \vec{dif}_N \end{bmatrix} = \begin{bmatrix} 0 & dif(N_1, N_2) & \ldots & dif(N_1, N_N) \\ dif(N_2, N_1) & 0 & \ldots & \ldots \\ \ldots & \ldots & 0 & \ldots \\ dif(N_N, N_1) & \ldots & \ldots & 0 \end{bmatrix}_{N*N}$$

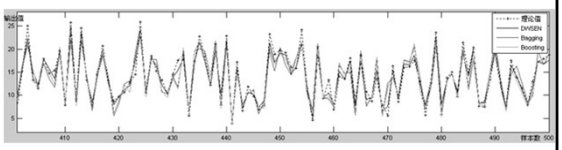
UCI Standard Data

Freidman#1 $t = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \varepsilon$

$$\begin{array}{c} \\ \overrightarrow{Dif_1} \\ \overrightarrow{Dif_2} \\ \overrightarrow{Dif_3} \\ \overrightarrow{Dif_4} \\ \overrightarrow{Dif_5} \end{array} \begin{array}{c} \begin{array}{ccccc} N_1 & N_2 & N_3 & N_4 & N_5 \end{array} \\ \begin{bmatrix} 0 & 0.3278 & 0.1671 & 0.2466 & 0.1750 \\ 0.3278 & 0 & 0.2700 & 0.2558 & 0.2249 \\ 0.1671 & 0.2700 & 0 & 0.2131 & 0.1547 \\ 0.2466 & 0.2558 & 0.2131 & 0 & 0.1214 \\ 0.1750 & 0.2249 & 0.1547 & 0.1214 & 0 \end{bmatrix} \end{array} \begin{array}{c} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \end{array}$$

$$\begin{array}{c} \\ \overrightarrow{Dif_1} \\ \overrightarrow{Dif_2} \\ \overrightarrow{Dif_3} \\ \overrightarrow{Dif_4} \\ \overrightarrow{Dif_5} \\ \overrightarrow{Dif_6} \end{array} \begin{array}{c} \begin{array}{cccccc} N_1 & N_2 & N_3 & N_4 & N_5 & N_6 \end{array} \\ \begin{bmatrix} 0 & 0.1733 & 0.3091 & 0.1011 & 0.2298 & 0.2584 \\ 0.1733 & 0 & 0.2836 & 0.1334 & 0.3638 & 0.1988 \\ 0.3091 & 0.2836 & 0 & 0.2292 & 0.4028 & 0.2597 \\ 0.1011 & 0.1334 & 0.2292 & 0 & 0.2606 & 0.2273 \\ 0.2298 & 0.3638 & 0.4028 & 0.2606 & 0 & 0.4189 \\ 0.2584 & 0.1988 & 0.2597 & 0.2273 & 0.4189 & 0 \end{bmatrix} \end{array} \begin{array}{c} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{array}$$

$$\begin{array}{c} \\ \overrightarrow{Dif_1} \\ \overrightarrow{Dif_2} \\ \overrightarrow{Dif_3} \\ \overrightarrow{Dif_4} \\ \overrightarrow{Dif_5} \\ \overrightarrow{Dif_6} \end{array} \begin{array}{c} \begin{array}{cccccc} N_1 & N_2 & N_3 & N_4 & N_5 & N_6 \end{array} \\ \begin{bmatrix} 0 & 0.3639 & 0.4141 & 0.3785 & 0.3995 & 0.3629 \\ 0.3639 & 0 & 0.3393 & 0.1723 & 0.1460 & 0.2602 \\ 0.4141 & 0.3393 & 0 & 0.2562 & 0.2947 & 0.3768 \\ 0.3785 & 0.1723 & 0.2562 & 0 & 0.0873 & 0.2603 \\ 0.3995 & 0.1460 & 0.2947 & 0.0873 & 0 & 0.2852 \\ 0.3629 & 0.2602 & 0.3768 & 0.2603 & 0.2852 & 0 \end{bmatrix} \end{array} \begin{array}{c} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{array}$$



	数据集	Bagging	Boosting	ALL-ENN	SEL-ENN	DWSEN
Freidman#1	泛化误差(%)	12.31	11.51	13.84	11.16	**10.83**
	标准偏差	0.1217	0.1264	0.1362	0.1073	**0.0922**

4.5 Applications

4.5.1 Classification

Two major areas:

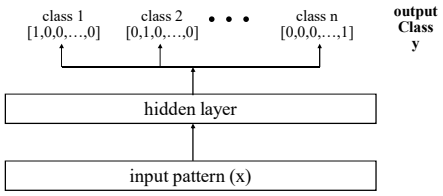
(1) Identification of process faults based on the operating conditions of given process

(2) Prediction of the most likely categorical group for a given input pattern

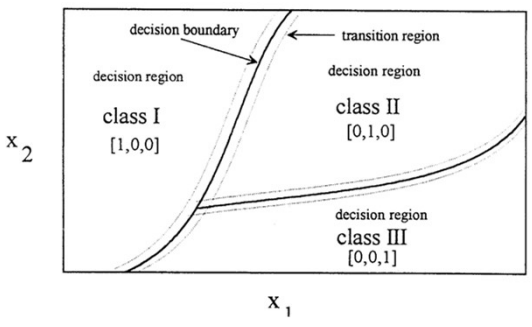
Two basic types of pattern classifiers:

Parametric and nonparametric

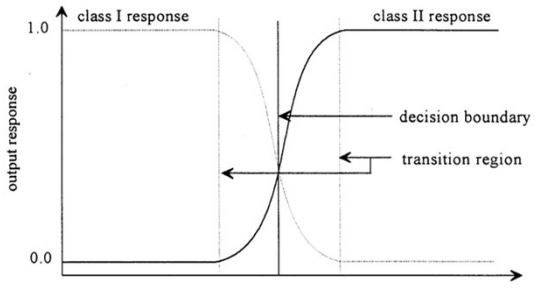
A standard neural network architecture for pattern-classification network



An illustration of a two-dimensional input space,decision regions, decision boundaries, and transition regions for a classification problem

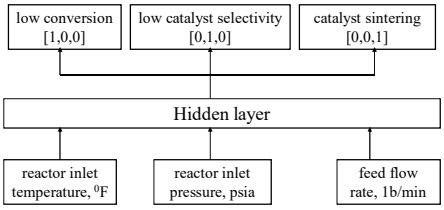


A classification network's predictions moving from the class I decision region (y=[1,0,0]) to the II decision (y=[0,1,0]).



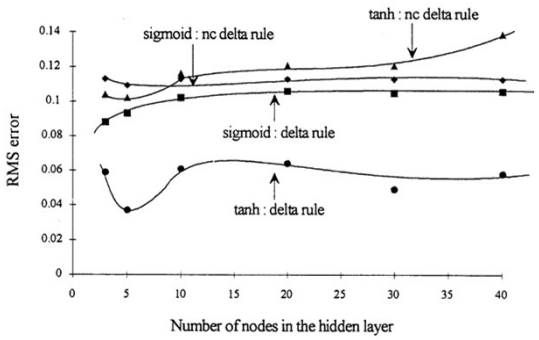
The two most commonly used network architectures for classification problems are the BP network and the RBF network.

Illustrative example of fault diagnosis of a chemical reactor

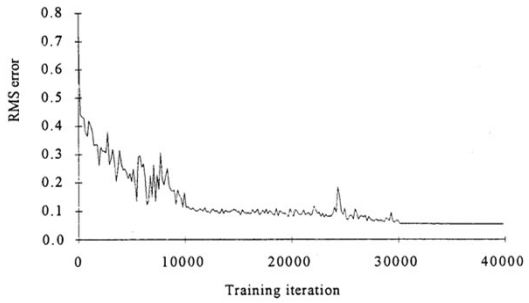


Next, we use a example to compare the BP network with the RBF network. In addition, we compare different transfer functions(e.g. sigmoid and hyperbolic tangent) and network training rules(e.g., delta rule and normalized cumulative delta rule, nc-delta).

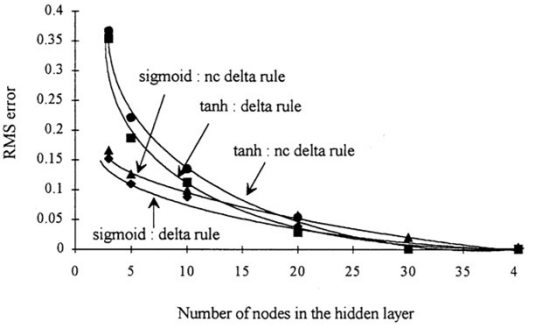
Using BP network



Using BP network with delta-learning rule and the hyperbolic tangent transfer function and **5** nodes in the hidden layer



Using the RBF network



Comparison of the BP network and RBF network

(1) RBF networks perform better than BP networks for classification problems

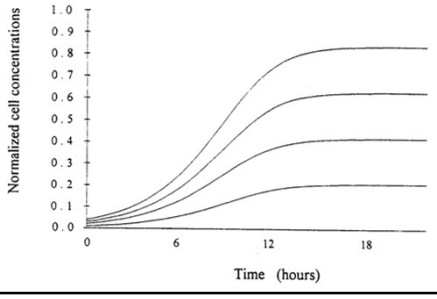
(2) The RBF network's RMS error approaches 0 as the number of nodes increases, while the BP network has a much larger RMS error of 0.037.

(3) BP network has a much more compact system representation than the RBF network, and has significantly lower RMS error for smaller networks(e.g. less than 5 nodes).

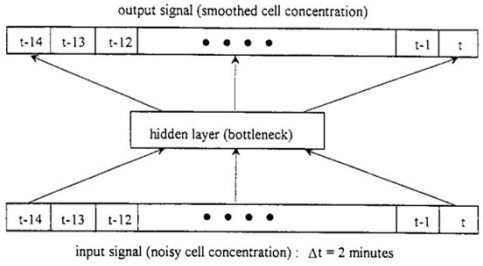
(4) The RBF network also trains faster requiring only 7000 iterations, versus 30000 for the BP network.

4.5.2 Data compression

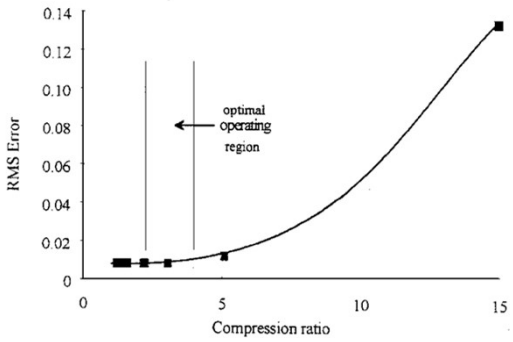
Using four simulated cell-growth curves spanning the entire range of possible cell-growth profiles, having maximum normalized cell concentrations of 0.2, 0.4, 0.6, and 0.8,respectively.



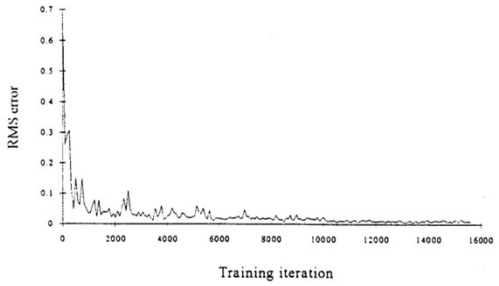
Each curve consist of 675 data points, divided into 45 training examples consisting of 15 data points each, producing a total training set of 180 examples(45*4).



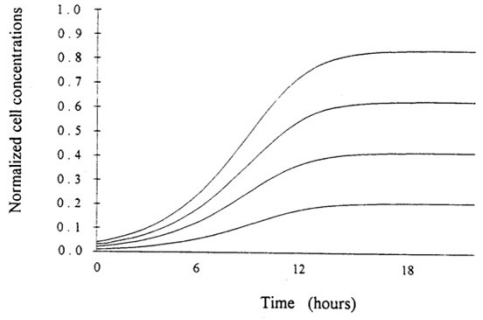
A plot of RMS error versus compression ratio used for the determination of the optimal compression ratio.



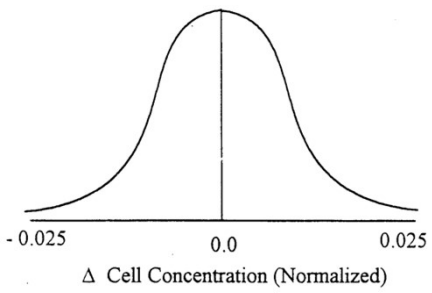
Select a 3:1 compression ratio(I.e. 15 nodes in both the input and output layers being compressed to 5 nodes in the hidden layer) for the network.



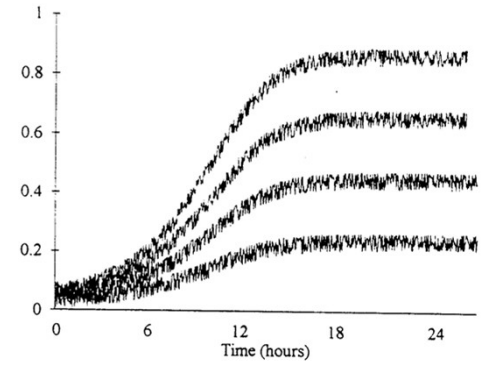
Following figure shows the four cell-growth curves predicted.
The average error of the normalized cell concentration for the training examples is 0.0027.



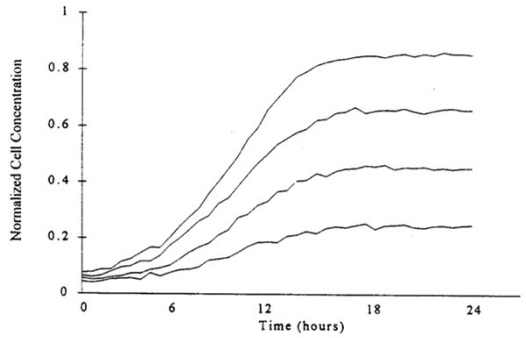
To demonstrate using the network to reduce noise in the cell-growth signal, we now add random noise to the four signals used for training.



The four cell-growth curves with noise



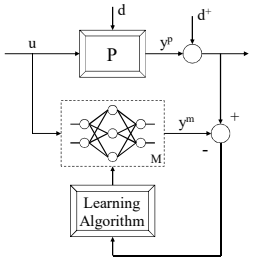
corresponding cell-growth curves after data compression with a 3:1 compression ratio.



4.5.3 Identification

1. Forward modelling

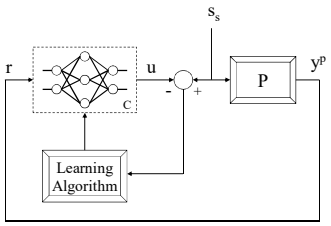
$$y^p(t+1) = f\left[y^p(t),\ldots y^p(t-n+1); u(t),\ldots u(t-m+1)\right]$$



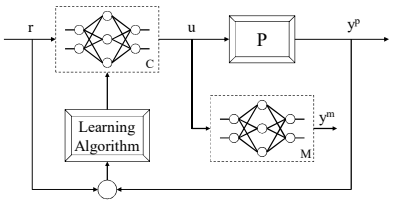
2. Inverse modelling

$$u(t) = g\left[y^p(t),\ldots, y^p(t-n+1), r(t+1); u(t-1),\ldots, u(t-m+1)\right]$$

Direct method

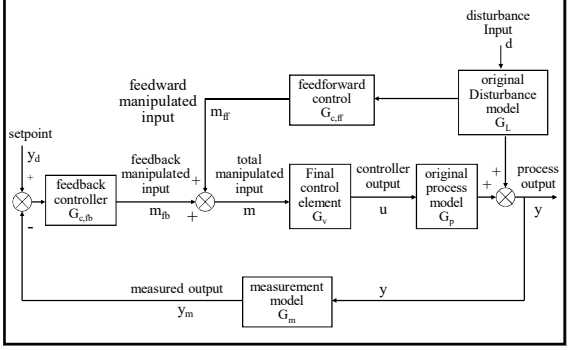


Specialized method



4.5.4 Control

1. A simplified block diagram for a process control system



2. Basic Digital PID control Algorithm

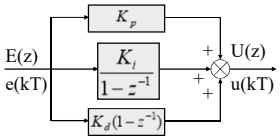
$$u(t) = K_p\left[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}\right]$$

$$u(kT) = K_p\left\{e(kT) + \frac{T}{T_I}\sum_{j=0}^{k} e(jT) + \frac{T_d}{T}[e(kT) - e(kT-T)]\right\}$$

$$U(z) = K_p\left\{E(z) + \frac{T}{T_I(1-z^{-1})}E(z) + \frac{T_d}{T}(1-z^{-1})E(z)\right\}$$

$$D(z) = \frac{K_p(1-z^{-1}) + K_i + K_d(1-z^{-1})^2}{1-z^{-1}}$$

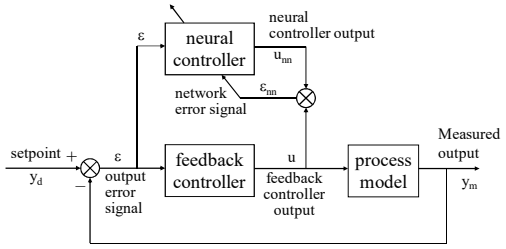
Digital PID Block diagram



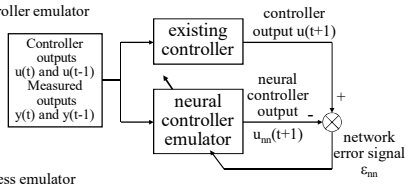
3. Neural Networks for Process Control

(1) Direct Network Control

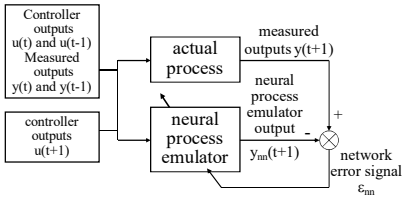
training a neural network as the controller determining the controller output directly



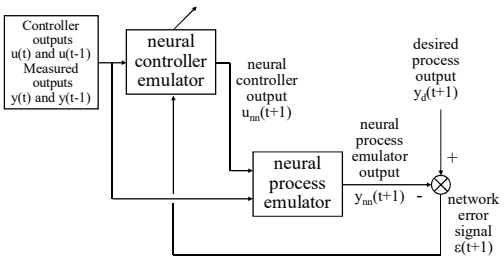
Neural controller emulator



Neural process emulator

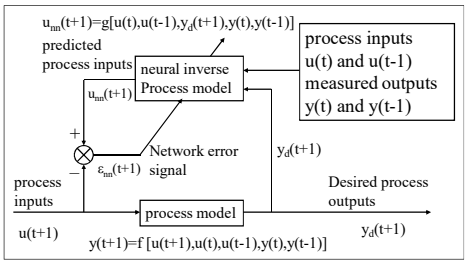


Integrated Neural Controller/Process System

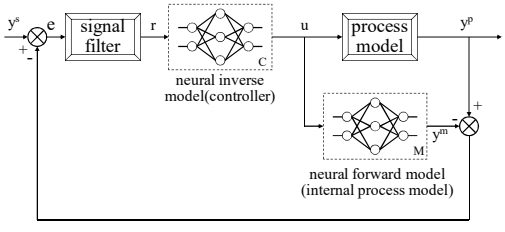


(2) Inverse Network Control

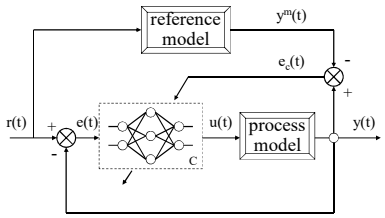
training a neural network as an "inverse" model of a process, predicting the process input necessary to produce the desired process outputs(setpoints).



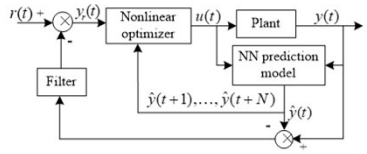
(3) Internal Model Control (IMC)



(4) Model Reference Control



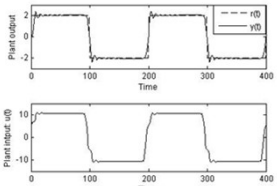
(5) Neural Network Nonlinear Predictive Control based upon Tent-map Chaos Optimization



The neural network is used as multi-step predictive model and the TCOA is applied to performing the nonlinear rolling optimization to enhance the convergence and accuracy in the NNPC.

Example 1

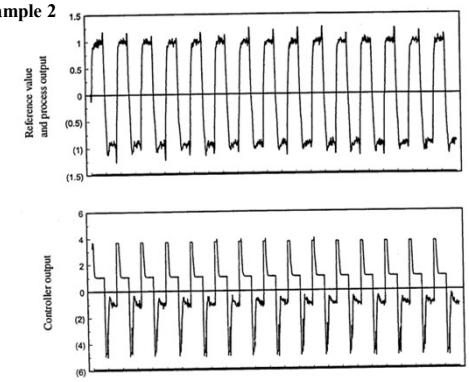
$$y(t) = 0.9722y(t-1) + 0.3578u(t-1)$$
$$-0.1295u(t-2) - 0.3103y(t-1)u(t-1)$$
$$-0.04228y^2(t-2) + 0.1663y(t-2)u(t-2)$$
$$-0.03259y^2(t-1)y(t-2)$$
$$-0.3513y^2(t-1)u(t-2)$$
$$+0.3084y(t-1)y(t-2)u(t-2)$$
$$+0.1087y(t-2)u(t-1)u(t-2)$$
$$+0.2573y(t-2)e(t-1)$$
$$+0.2939y^2(t-2)e(t-1)$$
$$+0.4770y(t-2)u(t-1)e(t-1)$$



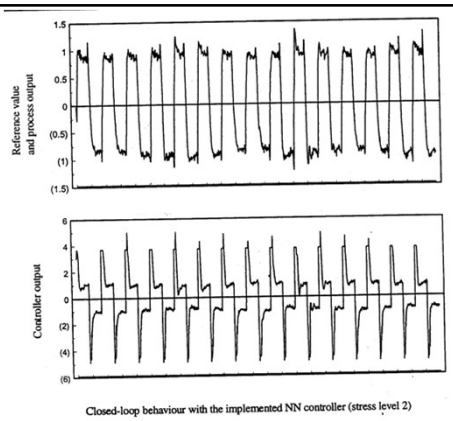
The process control input is calculated to minimize a criterion at each sampling instant

$$J = \sum_{j=1}^{N}\left[y_r(t+j) - \hat{y}(t+j)\right]^2 + \lambda\sum_{j=1}^{N_c}\left[\Delta u(t+j-1)\right]^2$$

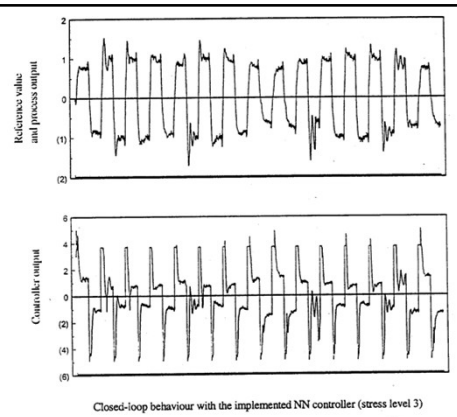
Example 2



Closed-loop behaviour with the implemented NN controller (stress level 1)



Closed-loop behaviour with the implemented NN controller (stress level 2)



Closed-loop behaviour with the implemented NN controller (stress level 3)