ELSEVIER

Information Sciences 132 (2001) 179–194

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

A connectionist model for selection of cases

Rajat K. De *, Sankar K. Pal

Machine Intelligence Unit, Indian Statistical Institute, Calcutta 700 035, India

Received 1 January 1999; received in revised form 7 September 2000; accepted 30 November 2000

Abstract

The present article describes a method of designing a connectionist model for selection of cases for decision-making problems. The notion of fuzzy similarity is used for selecting the same from overlapping regions. Cases are stored as network parameters. The architecture of the network is adaptively determined through *growing* and *pruning* of hidden nodes under supervised training. The effectiveness of the cases, thus selected by the network, is demonstrated for pattern classification problem using 1-NN rule with the cases as the prototypes. Results, along with comparisons, are presented for various artificial and real life data for different parameter values of the similarity function, controlling the number of cases.

Keywords: Case-based reasoning; Fuzzy similarity; Classification; Node growing; Node pruning

1. Introduction

A case-based system adapts old solutions to meet new demands, explains and critiques new situations using old instances (called cases), and performs reasoning from precedents to interpret new problems [1]. A case may be defined as a contextualized piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goals of the system. The system learns as a byproduct of its reasoning activity. It becomes more efficient and more

* Corresponding author.
E-mail addresses: rajat@isical.ac.in (R.K. De), sankar@isical.ac.in (S.K. Pal).

0020-0255/01/$ - see front matter
PII: S0020-0255(01)00070-6

competent as a result of storing the experience of the system and referring to them in later reasoning. Case-based system, in contrast to the traditional knowledge-based system, operates through a process of remembering one or a small set of concrete instances or cases and basing decisions on comparisons between the new situation and the old one. The task of selection of cases constitutes an important and integral part of a case-based system, particularly when the size of the data set is large.

Artificial neural networks (ANNs), having the capability of fault tolerance, adaptivity and generalization, and scope for massive parallelism, are widely used in dealing with learning and recognition tasks. For the last few years, attempts are being made for developing methodologies integrating case-based reasoning and ANNs. The problems include, among others, designing hybrid case-based connectionist systems [2–4], formulating connectionist indexing approaches [5,6], retrieval of cases using neural network [7], and learning of cases in connectionist framework [8].

This article is an attempt in this regard for developing a connectionist model for selecting cases in pattern recognition problems. Cases are viewed as typically labeled patterns which represent different regions of the classes. A notion of fuzzy similarity, using π-type membership function, is incorporated together with repeated *insertion* and *deletion* of cases in order to determine a stable case base. The architecture of the connectionist model is determined adaptively through *growing* and *pruning* of hidden nodes under supervised mode of training. In order to demonstrate the effectiveness of the network (methodology) for pattern classification, we have considered the principle of k-NN classifier with $k = 1$ and the cases as the prototypes. Results of the algorithm, along with comparisons, are adequately demonstrated on an artificial data, and real life speech [9] and medical data [10].

2. Selection of cases and class representation

In this section, we describe how the task of *selection of few samples from each class as cases*, is performed. (For the sake of convenience, the samples which are not selected as cases, are referred to as patterns in the subsequent discussion.) For performing this task, let us, first of all, define a similarity function for measuring the degree of similarity between a pattern and a case. The function is such that the higher its value, the higher is the degree of similarity between a pattern and a case.

Let $\mathbf{x} = [x_1, x_2, \ldots, x_i, \ldots, x_n]$ be a pattern vector of known classification in an n-dimensional feature space containing M classes. $\boldsymbol{\xi}_{l_k} = [\xi_{l_k1}, \xi_{l_k2}, \ldots, \xi_{l_ki}, \ldots, \xi_{l_kn}]$ denotes l_kth case from kth class C_k. $\mu_{l_k}(\mathbf{x})$ represents the degree of similarity of $\mathbf{x}$ to a case $\boldsymbol{\xi}_{l_k}$. $d_{l_k}(\mathbf{x})$ stands for the distance between $\mathbf{x}$ and $\boldsymbol{\xi}_{l_k}$.

The degree of similarity between a pattern $\mathbf{x}$ and a case $\boldsymbol{\xi}_{l_k}$ is defined as

$$\begin{aligned}\mu_{l_k}(\mathbf{x}) &= 1-2\left(\frac{d_{l_k}(\mathbf{x})}{\lambda}\right)^2, \quad 0\leqslant d_{l_k}(\mathbf{x})<\tfrac{\lambda}{2},\\ &= 2\left[1-\frac{d_{l_k}(\mathbf{x})}{\lambda}\right]^2, \quad \tfrac{\lambda}{2}\leqslant d_{l_k}(\mathbf{x})<\lambda,\\ &= 0, \quad \text{otherwise},\end{aligned} \tag{1}$$

where λ is the bandwidth of $\mu_{l_k}(\mathbf{x})$, i.e., the separation between its two (cross-over) points where $\mu_{l_k}=0.5$. Note that $\mu_{l_k}(\mathbf{x})$ can be viewed as a π-type membership function characterizing a fuzzy set of points representing a region R_{l_k} with $\boldsymbol{\xi}_{l_k}$ as its center [11].

The distance $d_{l_k}(\mathbf{x})$ may be expressed in many ways. Considering Euclidian norm, we have

$$d_{l_k}(\mathbf{x})=\left[\sum_{i=1}^{n}(x_i-\xi_{l_k i})^2\right]^{1/2}. \tag{2}$$

It is clear from Eq. (1) that $\mu_{l_k}(\mathbf{x})$ decreases with the increase in $d_{l_k}(\mathbf{x})$ and vice-versa. It is maximum ($=1.0$), if $d_{l_k}(\mathbf{x})$ is zero (i.e., if a pattern $\mathbf{x}$ and the l_kth case are identical). The value of $\mu_{l_k}(\mathbf{x})$ is minimum ($=0.0$), if $d_{l_k}(\mathbf{x})\geqslant\lambda$. When $d_{l_k}(\mathbf{x})=\frac{\lambda}{2}$, $\mu_{l_k}(\mathbf{x})$ is 0.5, i.e., an ambiguous situation arises. $\mu_{l_k}(\mathbf{x})$ implies that there is a crisp region R_{l_k} centered around a case $\boldsymbol{\xi}_{l_k}$, beyond which a pattern $\mathbf{x}$ is said to be dissimilar to $\boldsymbol{\xi}_{l_k}$. Note that, one may define $\mu_{l_k}(\mathbf{x})$ in a different way satisfying the above mentioned characteristics.

A pattern $\mathbf{x}$ is selected randomly from any class C_k. $\mathbf{x}$ is considered as the first case if the case-base B_k corresponding to class C_k is empty. Otherwise, $\mu_{l_k}(\mathbf{x})$ (Eq. (1)) corresponding to the the cases $\boldsymbol{\xi}_{l_k}$ in the case-base B_k, are computed. $\mathbf{x}$ is selected as a new case, if

$$\mu_{l_k}(\mathbf{x})\leqslant 0.5 \quad \forall\ l_k.$$

When a case is selected, it is *inserted* into the case-base. After repeating this process over all the training patterns, a set of cases constituting the case-base for each class is obtained. The case-base B for the entire training set is the union of all B_ks, i.e., $B=\cup_{k=1}^{M}B_k$.

After the formation of this case-base B, a case $\boldsymbol{\xi}_{l_k}$ for which $\mu_{l_k}(\mathbf{x})\leqslant 0.5$ is minimum, is *deleted* from B, if the number of patterns with $\mu_{l_k}(\mathbf{x})>0.5$ (or with $d_{l_k}(\mathbf{x})<\frac{\lambda}{2}$). The processes of *insertion* and *deletion* are repeated until the case-base becomes stable, i.e., the set of cases does not change further. This *deletion* process reduces the possibility of a spurious pattern being considered as a case.

Therefore, the class C_k can be viewed as a union of all the crisp regions R_{l_k} around its different cases, i.e.,

$$C_k=\cup_{l_k=1}^{s_k}R_{l_k},$$

where s_k is the number of cases in class C_k. Note that as the value of λ increases, the extent of R_{l_k}s representing different regions around ξ_{l_k}s increases, and therefore, the number of cases s_k decreases.

Effect of λ: As λ increases, the extent of the region around a case increases, and therefore the number of cases required for representing a class decreases. This implies that the generalization capability of an individual case increases with increase in λ. Initially, although the number of cases decreases with the increase in λ, the generalization capability of individual cases dominates. For further increase in λ, the number of cases becomes so low that the generalization capability of the individual cases may not cope with the proper representation of the class structures.

3. Formulation of the network

Let us describe here the design of the network model, based on the methodology of case selection described in Section 2. Its architecture is determined adaptively through *growing* and *pruning* of hidden nodes. Note that these *growing* and *pruning* phenomena correspond to the tasks of *insertion* and *deletion* of cases.

3.1. Architecture

The connectionist model (Fig. 1) consists of three layers: input, hidden and class. The input layer represents the set of input features, i.e., for each feature there is a node (called input node) in the input layer. Similarly, for each case there is a node in the hidden layer. For each hidden node, there is an auxiliary node which makes the hidden node ON or OFF. An auxiliary node corresponding to a hidden node sends back signal to the input layer only when it sends a signal to the hidden node for making it ON. The hidden nodes are made ON one at a time keeping the remaining OFF. For the purpose of keeping class information of the cases, we have considered class layer consisting of several nodes; each node (class node) representing a class.

The input nodes are connected to the hidden and auxiliary nodes by feedforward and feedback links, respectively. The weight of a feedforward link connecting ith input node and l_kth hidden node is

$$w_{l_k i}^{(0)} = 1 \quad \forall\ l_k, i. \tag{3}$$

The weight $w_{l_k i}^{(\mathrm{fb})}$ of a feedback link connecting the auxiliary node corresponding to l_kth hidden node and ith input node is the same as the ith feature value of the l_kth case ($\xi_{l_k i}$). That is,

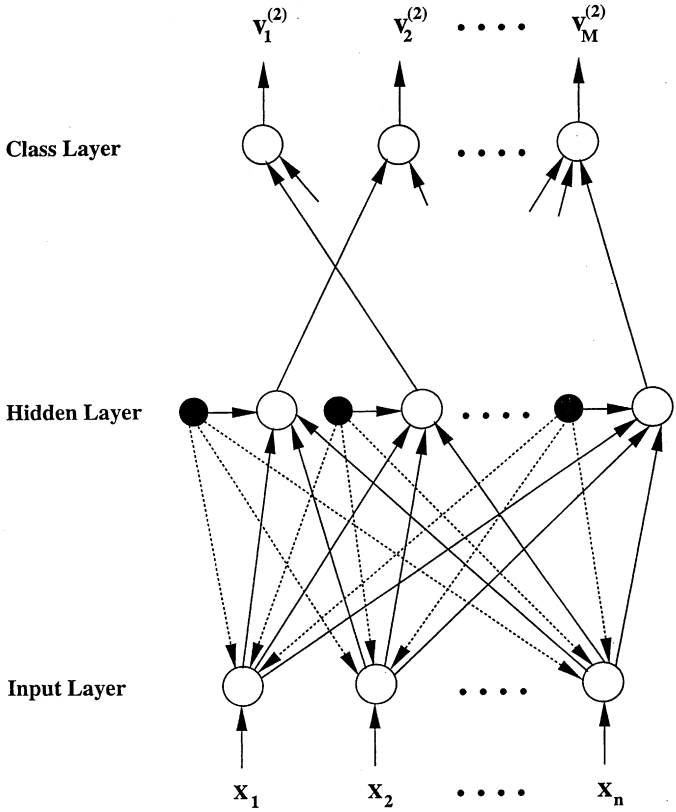

Fig. 1. A schematic diagram of the neural network model. Black circles represent the auxiliary nodes, and white circles represent input, hidden and class nodes.

$$w_{l_k i}^{(\mathrm{fb})} = \xi_{l_k i}. \tag{4}$$

The hidden layer is connected to the class layer via feedforward links. The weight ($w_{kl_k}^{(1)}$) of the link connecting l_kth hidden node and kth class node is 1, iff the case corresponding to the hidden node belongs to class C_k. Otherwise, there is no such links between hidden nodes and class nodes. That is,

$$\begin{aligned} w_{kl_k}^{(1)} &= 1 \quad \text{if } \boldsymbol{\xi}_{l_k} \in C_k, \\ &= 0, \quad \text{otherwise.} \end{aligned} \tag{5}$$

At the beginning, since the case-base is empty, there is no hidden node. Hence, the connectivity between the layers is not established. When there is at least one hidden node, a pattern $\mathbf{x}$ is presented to the input layer of the network. The activation of ith input node when l_kth hidden node is ON, is given by

$$v_{l_k i}^{(0)} = (u_{l_k i}^{(0)})^2. \tag{6}$$

$u_{l_k i}^{(0)}$ is the total input received by the ith input node when the l_kth hidden node is ON, and is given by

$$u_{l_k i}^{(0)} = x_{pi} - u_{l_k i}^{(\mathrm{fb})}, \tag{7}$$

where $u_{l_k i}^{(\mathrm{fb})} = (-1) * w_{l_k i}^{(\mathrm{fb})}$ (-1 being the feedback activation value of the auxiliary node corresponding to the l_kth hidden node) is the feedback input received by the input node. The total input received by the l_kth hidden node when it is made ON, is

$$u_{l_k}^{(1)} = \sum_i v_{l_k i}^{(0)} * w_{l_k i}^{(0)}. \tag{8}$$

The activation function of an l_kth hidden node is the same as $\mu_{l_k}(\mathbf{x})$ (Eq. (1)). Thus, the activation ($v_{l_k}^{(1)}$) of l_kth hidden node is given by

$$\begin{aligned} v_{l_k}^{(1)} &= 1 - 2\left(\frac{(u_{l_k}^{(1)})^{1/2}}{\lambda}\right)^2, && 0 \leqslant (u_{l_k}^{(1)})^{1/2} < \lambda/2, \\ &= 2\left[1 - \frac{(u_{l_k}^{(1)})^{1/2}}{\lambda}\right]^2, && \tfrac{\lambda}{2} \leqslant (u_{l_k}^{(1)})^{1/2} < \lambda, \\ &= 0, && \text{otherwise.} \end{aligned} \tag{9}$$

Here the value of λ is stored in all the hidden nodes.

3.2. Training and formation of the network

The network described in Section 3.1 is formed through *growing* and *pruning* of the hidden nodes during the training phase under supervised mode. Initially there is only input and class layers. The patterns are presented in a random sequence to the input layer of the network. The first pattern presented to the network is considered as a case. A hidden node along with its auxiliary node representing this case is added to the network. The connections of these auxiliary and hidden nodes with the input and class layers are established as described by Eqs. (3)–(5).

For the remaining patterns, their degrees of similarity with the cases represented by existing hidden nodes are computed, and if they are decided to be new cases (Section 2), hidden nodes are added through *growing* operation. After the process of addition is over, it is checked if there is any redundant hidden node. This is done through *pruning* operation depending on the criterion mentioned in Section 2. In this connection, one may note that as λ increases, the number of cases and hence the number of hidden nodes decreases. These two operations, which together constitute a single iteration, are continued until the structure of the network becomes stable, i.e., until

$$\sum_k \sum_{l_k i} |w_{l_k i}^{(\mathrm{fb})}(t)| = \sum_k \sum_{l_k i} |w_{l_k i}^{(\mathrm{fb})}(t-1)|, \tag{10}$$

where t is the number of iterations.

The aforesaid *growing* and *pruning* operations are described below.

Growing of hidden nodes: For a pattern $\mathbf{x} \in C_k$, if $v_{l_k}^{(1)} \leqslant 0.5$ and $\mathbf{w}_{l_k}^{(\mathrm{fb})} = \boldsymbol{\xi}_{l_k} \in C_k$ for all the hidden nodes, $\mathbf{x}$ is selected as a case. A hidden node along with its auxiliary node is added to the network for representing this case and the links are established accordingly, using Eqs. (3)–(5). This process is called *growing of hidden nodes*. Note that the task 'insertion' of cases described in Section 2, is performed through this process.

Pruning of hidden nodes: An l_kth hidden node is deleted, if

$$v_{l_k}^{(1)} = \min_{\boldsymbol{\xi}_{l_k} = \mathbf{w}_{l_k}^{(\mathrm{fb})} \in C_k} v_{l_k}^{(1)} \leqslant 0.5$$

and the number of training samples for which $v_{l_k}^{(1)} > 0.5$ is less than a pre-defined value. In this way, the network is pruned. Note that the task 'deletion' of cases described in Section 2, is performed through this process.

4. 1-NN classification using the cases

In order to demonstrate the effectiveness of the network model (i.e., the capability of the cases in representing respective classes) for pattern classification, we have considered the principle of 1-NN rule with the cases as the prototypes. According to this rule, an unknown sample $\mathbf{x}$ is said to be in class C_j if for an L_jth case

$$v_{L_j}^{(1)} = \max_{k,l_k} \{v_{l_k}^{(1)}\}, \quad j,k = 1,2,\ldots,M.$$

For performing this task, each node in the class layer (Fig. 1) is considered to function as a Winner-Take-All network. A kth class node receives activations only from the hidden nodes corresponding to the cases in C_k. That is, the activation received by the kth class node from the l_kth hidden node is

$$u_{kl_k}^{(2)} = v_{l_k}^{(1)} * w_{kl_k}^{(1)}. \tag{11}$$

The output of kth class node is

$$v_k^{(2)} = \max_{l_k} \{u_{kl_k}^{(2)}\}, \tag{12}$$

where $v_k^{(2)}$ represents the degree of belongingness of $\mathbf{x}$ to class C_k. Therefore, decide $\mathbf{x} \in C_j$ if

$$v_j^{(2)} > v_k^{(2)}, \quad j,k = 1,2,\ldots,M, \ \ j \neq k.$$

5. Experimental results

In this section, the effectiveness of the network (methodology) for automatic selection of cases is demonstrated by making the cases function as prototypes for a 1-NN classifier. Artificially generated data Pat1, and the real life vowel [9] and a medical data [10] are considered as input. In all the cases, the data set has been divided into two subsets – *training* and *testing*. While perc% samples are considered during training, the remaining (100 – perc)% is used for testing. The synthetic data Pat1 (Fig. 2) has two input features, two classes and 557 pattern points.

The vowel data [9] consists of a set of 871 Indian Telugu vowel sounds. These were uttered in a consonant–vowel–consonant context by three male speakers in the age group of 30–35 years. The data set has three features, F_1, F_2 and F_3 corresponding to the first, second and third vowel formant frequencies obtained through spectrum analysis of the speech data. Fig. 3 shows the

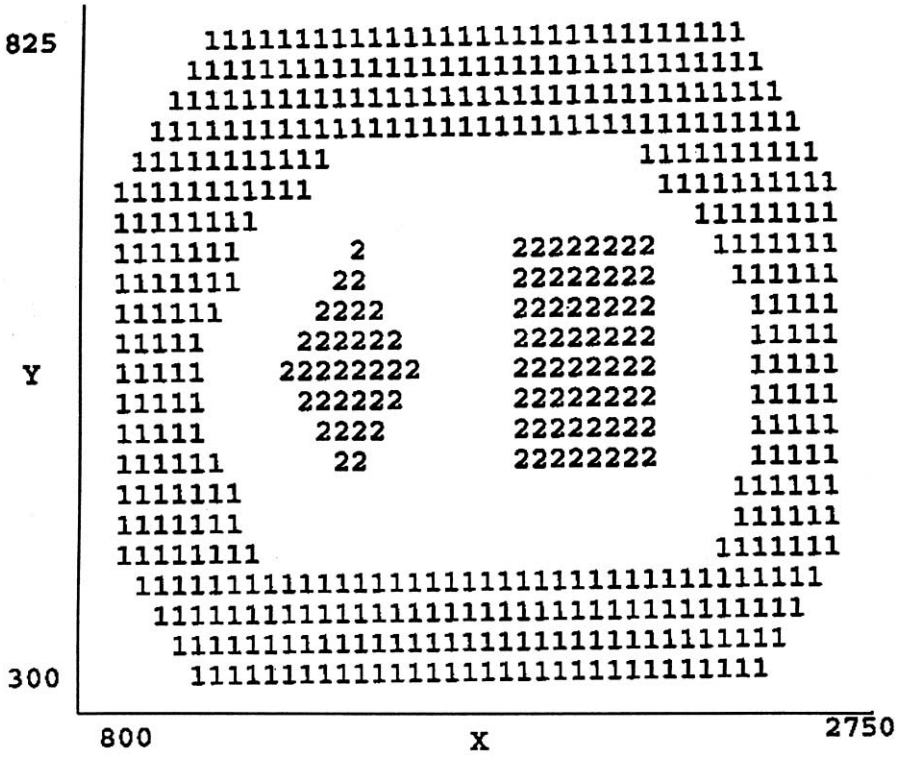

Fig. 2. Scatter plot of the artificially generated Pat1 data. Here '1' and '2' represent patterns in classes 1 and 2, respectively.

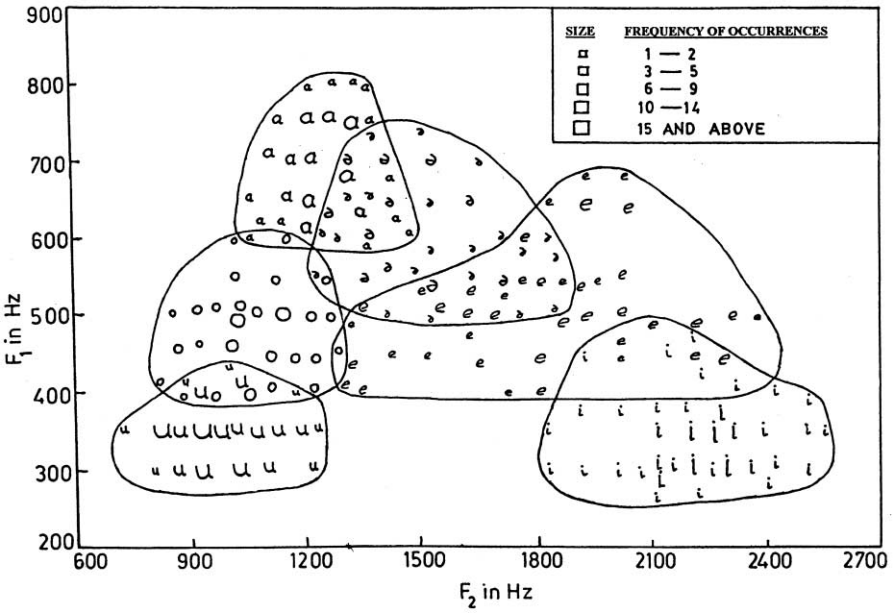

Fig. 3. Scatter plot of the vowel data in F_1–F_2 plane.

overlapping nature of the six vowel classes (viz., ∂, a, i, u, e, o) in the F_1–F_2 plane (for ease of depiction). The details of the data and its extraction procedure are available in [9]. This vowel data is being extensively used for more than two decades in the area of pattern recognition.

The medical data consisting of nine input features and four pattern classes, deals with various *Hepatobiliary disorders* [10] of 536 patient cases. The input features are the results of different biochemical tests, viz., Glutamic Oxalacetic Transaminate (GOT, Karmen unit), Glutamic Pyruvic Transaminase (GPT, Karmen Unit), Lactate Dehydrase (LDH, iu/l), Gamma Glutamyl Transpeptidase (GGT, mu/ml), Blood Urea Nitrogen (BUN, mg/dl), Mean Corpuscular Volume of red blood cell (MCV, fl), Mean Corpuscular Hemoglobin (MCH, pg), Total Bilirubin (TBil, mg/dl) and Creatinine (CRTNN, mg/dl). The hepatobiliary disorders Alcoholic Liver Damage (ALD), Primary Hepatoma (PH), Liver Cirrhosis (LC) and Cholelithiasis (C), constitute the four classes.

Tables 1–3 depict some of the results obtained with the above data sets for different values of λ when perc = 30 is considered. The first column of these tables indicates the number of iteration(s) required by the network until it stabilizes during training. It is found from these tables that the recognition scores on the training set, as expected, are higher than those on the test set. The recognition score during training decreases with the increase in the value of λ.

Table 1
Classification performance for different λ using Pat1 for perc = 30.

Number of iterations	λ	Class	Number of hidden nodes	Recognition score (%)	
				Training set	Testing set
1	150.0	1	54	100.0	100.0
		2	12	100.0	100.0
		Overall	66	100.0	100.0
1	200.0	1	45	100.0	100.0
		2	11	100.0	100.0
		Overall	56	100.0	100.0
1	250.0	1	29	100.0	100.0
		2	7	100.0	100.0
		Overall	36	100.0	100.0
1	300.0	1	23	100.0	99.69
		2	6	100.0	94.12
		Overall	29	100.0	98.72
1	350.0	1	16	98.91	97.55
		2	5	89.47	87.18
		Overall	21	97.30	95.74

On the other hand, for the test data, the recognition score increases with λ up to a certain value, beyond which it decreases. This can be explained as follows.

During training, the recognition score increases with decrease in λ due to better abstraction capability. While for the test data, as λ decreases, the modeling of class structures improves because of the increase in the number of cases, and therefore, the recognition score increases up to a certain value of λ. Beyond that, as mentioned in Section 2, the number of cases with poor generalization capability (i.e., similarity functions with very small bandwidth) increases. As a result, the recognition score decreases due to *overlearning*.

As mentioned in Section 3.2, the number of hidden nodes of the network decreases with the increase in λ, for all the cases (Tables 1–3). Since class 1 of Pat1 is more sparse than class 2 (Fig. 2), it needs more cases (and hence more hidden nodes) for its representation. This is reflected in Table 1. Similar observations hold good for vowel and medical data where class 'e' for vowel and PH for medical data, being most sparse, have the maximum number of hidden nodes. Note from Tables 1–3 that, for all the data sets, the stability of the architecture of the networks is achieved with a very few iterations.

Table 2
Classification performance for different λ using vowel data for perc = 30

Number of iterations	λ	Class	Number of hidden nodes	Recognition score (%)	
				Training set	Testing set
3	100.0	∂	21	95.24	41.18
		a	22	100.0	84.13
		i	42	98.04	72.73
		u	31	97.78	81.13
		e	53	98.39	67.59
		o	38	94.44	89.68
		Overall	207	97.30	75.0
3	150.0	∂	18	95.24	64.71
		a	13	96.15	93.65
		i	23	96.08	86.78
		u	20	88.89	86.79
		e	37	96.77	80.0
		o	26	92.59	85.71
		Overall	137	94.21	83.82
3	200.0	∂	16	80.95	64.71
		a	13	92.31	90.48
		i	21	98.04	87.60
		u	19	91.11	85.85
		e	36	93.55	81.38
		o	25	90.74	86.51
		Overall	130	92.28	83.99
1	250.0	∂	12	71.43	58.82
		a	9	88.46	80.95
		i	11	92.16	85.95
		u	9	84.44	72.64
		e	20	91.94	80.69
		o	14	81.48	74.60
		Overall	75	86.49	77.29
1	300.0	∂	10	57.14	52.94
		a	8	92.31	80.95
		i	10	92.16	86.78
		u	8	97.78	83.96
		e	20	88.71	80.69
		o	11	64.81	59.52
		Overall	67	83.78	75.82
3	350.0	∂	8	52.38	52.94
		a	7	92.31	95.24

Table 2 (Continued)

Number of iterations	λ	Class	Number of hidden nodes	Recognition score (%)	
				Training set	Testing set
		i	9	94.12	90.08
		u	8	97.78	89.62
		e	13	70.97	66.21
		o	8	46.30	42.86
		Overall	53	75.68	72.06
1	400.0	∂	8	57.14	56.86
		a	7	96.15	95.24
		i	7	88.24	86.78
		u	6	97.78	84.91
		e	10	69.35	65.52
		o	8	72.22	64.29
		Overall	46	80.31	75.16
1	450.0	∂	7	71.43	70.59
		a	5	84.62	68.25
		i	5	58.82	61.16
		u	6	93.33	83.02
		e	9	83.87	76.55
		o	6	68.52	67.46
		Overall	38	76.45	71.41

In order to demonstrate the effect of the size of a training set on the performance of the network, we have considered only the vowel data. Different values of perc considered are 10, 20, 30, 40, 50, 60 and 70 with $\lambda = 150.0, 200.0$ and 250.0. (Note that the network achieves the best generalization capability for $\lambda = 200.0$ (Table 2).) Table 4 shows that the recognition score on the test set, as expected, increases in general with the size of the training set.

Comparison: In a part of the experiment, we have compared the performance of the said classifier (where the cases are considered as prototypes) with that of the following ones:

(i) A standard k-NN classifier with $k = \sqrt{s}$ (s being the number of training samples) where all the perc% samples, selected randomly, are considered as prototypes. (It is known that as s goes to infinity, if the values of k and k/s can be made to approach infinity and zero, respectively, then the performance of k-NN classifier approaches that of the (optimal) Bayes classifier [12]. One such value of k for which the limiting conditions are satisfied is $\sqrt{s}$.)

(ii) Bayes maximum likelihood classifier where a multivariate normal distribution of samples with different class dispersion matrices and a priori prob-

Table 3
Classification performance for different λ using the medical data for perc = 30

Number of iterations	λ	Class	Number of hidden nodes	Recognition score (%)	
				Training set	Testing set
1	150.0	ALD	17	61.76	32.93
		PH	30	81.13	48.80
		LC	19	91.89	73.56
		C	13	42.86	27.71
		Overall	79	71.07	46.42
1	160.0	ALD	20	71.43	56.79
		PH	35	70.37	29.84
		LC	17	78.95	66.28
		C	8	58.33	57.32
		Overall	80	69.94	50.13
1	170.0	ALD	20	71.43	58.02
		PH	34	68.52	29.03
		LC	17	78.95	66.28
		C	8	55.56	54.88
		Overall	79	68.71	49.60
1	180.0	ALD	20	68.57	56.79
		PH	34	72.22	31.45
		LC	16	71.05	61.63
		C	8	55.56	54.88
		Overall	78	67.48	49.06
1	190.0	ALD	19	80.00	61.73
		PH	33	77.78	36.29
		LC	14	76.32	68.60
		C	6	8.33	12.20
		Overall	72	62.58	43.97
7	200.0	ALD	15	76.47	58.54
		PH	25	71.70	45.60
		LC	14	72.97	68.97
		C	11	25.71	9.64
		Overall	137	62.89	45.89

abilities ($= s_j/s$, for s_j patterns from class C_j) are assumed, and all the perc% samples are used to compute the mean vectors and the covariance matrix. Table 5 depicts that the network (CBNN) performs better than k-NN ($k = \sqrt{s}$) and Bayes maximum likelihood classifiers for Pat1 and vowel data. In the case

Table 4
Classification performance for different λ and perc on vowel data

λ	Recognition score (%)						
	perc = 10	perc = 20	perc = 30	perc = 40	perc = 50	perc = 60	perc = 70
150.0	70.99	81.55	83.82	80.42	84.67	86.29	87.01
200.0	75.70	82.12	83.99	83.27	84.67	85.71	86.20
250.0	75.06	80.11	77.29	80.23	82.38	83.43	84.03

Table 5
Comparative recognition score of various classifiers on different data sets

Data set	Class	Recognition score (%)					
		CBNN		Bayes		k-NN	
		Training	Testing	Training	Testing	Training	Testing
Pat1	1	100.0	100.0	100.0	100.0	100.0	99.38
	2	100.0	100.0	34.48	19.12	96.55	100.0
	Overall	100.0	100.0	88.62	85.90	99.40	99.49
Vowel	∂	80.95	64.71	38.10	43.14	23.81	33.33
	a	92.31	90.48	88.46	85.71	80.77	85.71
	i	98.04	87.60	90.20	85.12	88.24	85.12
	u	91.11	85.85	91.11	90.57	86.67	76.42
	e	93.55	81.38	75.81	80.69	75.81	77.93
	o	90.74	86.51	92.59	85.71	92.59	88.89
	Overall	92.28	83.99	83.01	81.70	79.92	78.43
Medical	ALD	71.43	56.79	61.76	50.00	52.94	46.34
	PH	70.37	29.84	54.72	64.80	69.81	77.60
	LC	78.95	66.28	51.35	36.78	21.62	29.89
	C	58.33	57.32	91.43	75.90	54.29	61.45
	Overall	69.94	50.13	63.52	57.56	51.57	56.23

of medical data, while the performance of CBNN on the training set is better than those obtained by the others, the reverse is true on the test samples.

6. Conclusions

We have described a method of designing a connectionist model (CBNN) for the selection of cases. A notion of fuzzy similarity, using π-type membership function, is incorporated together with the process of repeated *insertion* and *deletion* of cases in order to determine a stable case base. Cases are stored as network parameters during its supervised training for pattern recognition

problems. The architecture of the network is determined adaptively through *growing* and *pruning* of hidden nodes. The effectiveness of the cases, thus selected by the network, has been demonstrated for pattern classification problems by considering them as prototypes of a 1-NN classifier.

Experimental results demonstrate that the number of hidden nodes increases with the decrease in extent λ of the π-function (Eq. 1). As λ decreases, the performance during training increases because of the higher number of representative cases. On the other hand, during testing, it increases with the decrease in λ up to a certain value, beyond which the performance deteriorates because of *overlearning* (poor generalization capability of the cases). Note that here λ is the same for all the classes and is chosen empirically. However, λ might be distinct for individual classes. Its value could either be estimated from densities of individual classes, or be determined adaptively from the data set.

It has been found that CBNN performs better than k-NN, with $k = \sqrt{s}$, and Bayes maximum likelihood classifiers for Pat1 and vowel data. In other words, the class representation capability of the fewer cases selected by the CBNN is seen to be superior to those of the entire perc% samples selected randomly for k-NN, with $k = \sqrt{s}$, and Bayes classifiers. However, for medical data, the generalization capability of CBNN is seen to be poorer.

One may further note a drawback of the k-NN classifier that it needs to store all the prototypes (and hence to compute the distances from all of them). In this respect, the merit of the CBNN in selecting cases out of all the training samples is evident from the point of space and time complexity.

References

[1] J.L. Kolodner, Case-Based Reasoning, Morgan Kaufmann, San Mateo, 1993.
[2] P. Thrift, A neural network model for case-based reasoning, in: K.J. Hammond (Ed.), Proceedings of the DARPA Case-Based Reasoning Workshop, Morgan Kaufmann, California, 1989.
[3] B. Yao, Y. He, A hybrid system for case-based reasoning, in: Proceedings of the World Congress on Neural Networks, San Diego, USA, 1994, pp. 442–446.
[4] P.B. Musgrove, J. Davis, D. Izzard, A comparison of nearest neighbor, rule induction and neural networks for the recommendation of treatment at anticoagulant out-patient clinics, in: I. Watson (Ed.), Proceedings of the Second UK Workshop on Case-Based Reasoning, Salford, UK, 1996.
[5] E. Domeshek, A case study of case indexing: designing index feature sets to suit task demands and support parallelism, in: J. Barnden, K. Holyoah (Eds.), Advances in Connectionist and Neural Computation Theory, vol. 2: Analogical Connections, Ablex, Norwood, 1993.
[6] M. Malek, A connectionist indexing approach for CBR systems, in: M. Veloso, A. Aamodt (Eds.), Case-Based Reasoning Research and Development, Springer, Berlin, 1995.
[7] C. Milare, A. de Carvalho, Using a neural network in a CBR system, in: Proceedings of the International Conference on Computational Intelligence and Multimedia Applications, ICCIMA'98, Victoria, Australia, 1998, pp. 43–48.

[8] E. Reategui, J.A. Campbell, S. Borghetti, Using a neural network to learn general knowledge in a case-based system, in: M. Veloso, A. Aamodt (Eds.), Case-Based Reasoning Research and Development, Springer, Berlin, 1995.
[9] S.K. Pal, D. DuttaMajumder, Fuzzy Mathematical Approach to Pattern Recognition, Wiley (Halsted Press), New York, 1986.
[10] Y. Hayashi, A neural expert system with automated extraction of fuzzy if–then rules and its application to medical diagnosis, in: R.P. Lippmann, J.E. Moody, D.S. Touretzky (Eds.), Advances in Neural Information Processing Systems, Morgan Kaufmann, Los Altos, 1991, pp. 578–584.
[11] S.K. Pal, P.K. Pramanik, Fuzzy measures in determining seed points in clustering, Pattern Recognition Letters 4 (1986) 159–164.
[12] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, New York, 1972.