Neural Comput & Applic (2001)10:264–276

Neural
Computing
& Applications

Case-Based Reasoning System and Artificial Neural Networks: A Review

Daqing Chen and Phillip Burrell

Knowledge-Based Systems Centre, School of Computing, Information System and Mathematics, South Bank University, London, UK

In this survey paper, the-state-of-art of the connectionist model (i.e. Artificial Neural Network (ANN)) based methodology for a Case-Based Reasoning (CBR) system design is discussed. Special emphasis is laid on how the ANN can advance CBR technology by building an ANN-based CBR system, or integrating itself as a component within a CBR system. Several ANN models proposed for constructing a CBR system and for solving some special issues involved in a CBR process are described. The main characteristics of each model are analysed, and the advantages and limitations of different models are compared. Also, future research directions are outlined.

Keywords: Artificial intelligence; Artificial neural networks; Case-based reasoning; Decision-making system; Expert system; Machine learning

1. Introduction

As a significant branch of Artificial Intelligence (AI), Case-Based Reasoning (CBR) has received more and more research attention. In the last few decades, CBR has grown from a quite new area to a subject of major influence. Much work has been dedicated to this topic, including its basic principle [1,2], methodologies [3,4] and applications [5]. CBR systems have also been used to solve a wide range of problems. Examples of the applications of the CBR systems include medical diagnosis [6,7], time series prediction [8], product design [9,10] and planning [11,12], etc. The basic idea behind CBR is that reasoning and problem-solving are based on the most specific experiences available instead of general abstract knowledge or rules. In this way, case-based reasoning provides a new method for building intelligent systems.

However, although CBR is simple in principle, and has been successfully used in engineering problem-solving, it still lacks a generally theoretically sound framework. A systematically mathematical model has not been established for CBR system design, analysis, comparison and test. Consequently, most CBR systems could not complete their reasoning process, and propose a solution to a given task without intervention of domain experts or system managers [5]. Many CBR systems, for example, actually act as case retrieval and proposal systems [4], while case adaptation and case update are performed by human experts. On the other hand, based on specific application backgrounds of interest, different CBR systems usually use different approaches for case representation, case-base construction, case indexing, and criteria for case matching and adaptation: some are quantitative and algorithm-based; and the others are descriptive, constructive and experience-based. As such, although all the CBR systems share the same basic principle in common, it may be difficult to compare and contrast one CBR system with another, so that the performances of different CBR systems can be estimated and verified on a common benchmark.

Recently, several methodologies have been put forward for using the connectionist approach (i.e. Artificial Neural Networks (ANN)) in CBR system

Correspondence and offprint requests to: D. Chen, Knowledge-Based Systems Centre, School of Computing, Information Systems and Mathematics, South Bank University, 103 Borough Road, London SE1 OAA, UK. E-mail: chend@sbu.ac.uk

design [13,14], and some relating examples in engineering applications have been reported [9,15–17]. In this survey paper, the-state-of-art of connectionist model-based CBR system design is discussed. Special emphasis is laid on how the ANN can advance CBR technology by building an ANN-based CBR system, or integrating itself as a component within a CBR system. A comprehensively comparative study on the main feature of each method is presented. Furthermore, some other network models that are potentially useful in developing CBR systems are described.

The rest of this paper is organised as follows. Section 2 presents some basic concepts concerning CBR. The nature of CBR and several key components involved in the reasoning process of CBR are introduced. Also, some problems and challenges met in a CBR system design are described. The importance of using an ANN technique in a CBR system is then discussed. In Section 3, the CBR is further discussed from the instance-based learning point of view, which provides an interesting bridge between CBR and ANN, as well as a basis for comparing and contrasting these two methods. Several ANN models proposed for constructing a CBR system and for solving some specific issues involved in the CBR process are summarised in Section 4. The main characteristics of each model are analysed, and the advantages and limitations of different models compared. On this basis, future research directions and other network models are suggested and discussed in Section 5. Finally, concluding remarks are given in Section 6.

2. Case-Based Reasoning System

Classical knowledge- or rule-based decision support systems draw conclusions by applying generalised rules, step-by-step, starting from scratch. Although successful in many application areas, such systems have met several problems in knowledge acquisition and system implementation.

Inspired by the role of reminding in human reasoning, CBR systems have been proposed as an alternative to rule-based systems, where knowledge or rule elicitation are a difficult or intractable process. In principle, the CBR approach takes a different view, in that reasoning and problem-solving are performed by applying the experience that a system has acquired. This approach focuses on how to exploit human experience, instead of rules, in problem-solving, and thus improving the performance of decision support systems. In brief, reasoning in CBR is based on experience or remembering.

In CBR, a primary knowledge, stored in memory, is not compiled from rules, but a set of structured cases. These cases represent an experience or lesson that, under which situations and to what problems, some specific solutions have been derived in the past to achieve the goals of the reasoner, and the effects after the solutions have been applied. When a new problem is presented to the system, a new solution is proposed by retrieving the most relevant old cases and adapting them to fit new situations, which is then introduced as a new case into the system in order to enrich the case base. Therefore, a case-based reasoner solves new problems by adapting solutions that were used to solve old problems [1], and with efficient adaptation strategies and a proper complement of new cases, a CBR reasoner is expected to work more and more effectively.

Generally speaking, successful use of CBR depends upon addressing issues of how to acquire, represent, index, retrieve and adapt existing cases. However, the most important operations involved in a primary reasoning process of CBR are: case retrieval and case adaptation, together with case similarity assessment and case adaptation criteria. For a new problem or situation, a set of the most similar cases is retrieved by the CBR, based on similarity assessment criteria, and the solution, corresponding to these previous cases, is adapted to propose a solution for the given problem based on the adaptation criteria.

It is obvious that the criteria for both case similarity assessment and case adaptation are inextricably linked, which means that the solution suggested is intimately connected to the similarities and differences between new and old cases. More precisely, such a statement may be described as

$$\textit{Suggested Solution} = \boldsymbol{A}[\boldsymbol{S}(\{C^{Old}\}, C^{New}] \qquad (1)$$

where $\{C^{Old}\}$ and C^{New} denote a set of old cases and a new case, respectively. $\boldsymbol{S}(\cdot,\cdot)$ is the similarity assessment criterion, and $\boldsymbol{A}$, an operator, is defined by the adaptation criteria. Therefore, in essence, the CBR reasoning process is a pattern matching and classification process.

Developing case similarity assessment criteria and case adaptation criteria is a central open challenge for CBR.

In a symbolic description model-based CBR system, the case adaptation process is often performed by a rule-based system, which makes the CBR system design re-confront the knowledge acquisition problem for rule-based systems, and consequently, contradicts the important motivation for using CBR in problem-solving. As a result, many CBR systems currently work primarily as a case retrieval and

proposal system, leaving the case adaptation to be undertaken by field experts. Some new methods have been proposed to overcome the case adaptation problems [2,4]. Basically, however, these approaches are not easy to handle without the help of human experts or system designers.

On the other hand, for a quantitative description model-based CBR system, the system design may become more flexible compared with symbolic description model-based systems. This is because a lot of mathematical approaches and optimisation techniques are available for defining, synthesising and analysing the case similarity assessment and case adaptation criteria. In particular, the connectionist approach and several related ANN models have been suggested for the CBR system design.

As is well known, ANN approaches have many appealing characteristics, such as parallelism, robustness, adaptability and generalisation. Neural networks have the potential to provide some human characteristics of problem-solving that are difficult to describe and analyse using the logical approach of expert systems. More importantly, learning and problem-solving can be incorporated naturally and effectively by using a network. In a pattern classification problem, for example, classifiers based on a Multilayer Feedforward Neural Network (MFNN) [18,19] and a Radial Basis Function (RBF) Network [20,21] can perform not only the classification itself, but also internally extract input pattern features. For such networks, the feature extraction and classification can be considered as being merged in the single classifier network.

In addition, some methods have been put forward to extract symbolic representations (i.e. rules) from a trained network [22–25], which makes it possible to combine the network with the symbolic description model-based CBR system.

In the following sections, the connectionist model-based approaches for CBR system design are discussed in detail. As an introduction in the next section, a comparison between CBR and ANN is made from the machine learning viewpoint [26].

3. Instance-Based Learning, CBR and ANN

The discussion made in this section is based on the work in Mitchell [26].

Instance-based learning is an important method in machine learning theory. A key property of such a method is that it simply stores some typical example instances. When a new instance is presented, its relationship to these previously stored instances is analysed, and a set of similar example instances is retrieved from memory. These instances are then used to classify the new instance according to some given criteria. In other words, classification of a new instance is based on its similarity to the known example instances, and is performed at the instance query time. Such a method is a *lazy* learning method.

Several representative examples of instance-based learning approaches include the k-nearest neighbour algorithm and locally-weighted linear regression. In these algorithms, an instance $\boldsymbol{x}$ is represented as a point in an m-dimensional space, $\boldsymbol{x} = (x_1 \ldots x_m)^T$, and the criteria consist of the standard Euclidean distance norm as a similarity metric and a discrete-valued or real-valued target function for the classification assignment. When a new query instance is encountered, a set of example instances, near the query point, is selected based on the distance norm. An approximation to the target function is constructed over all these instances surrounding the query instance, which is then used as the estimated target value for the new instance, and classification of the new query instance is made accordingly.

In the locally-weighted linear regression algorithm, for example, a real-valued target function $F(\boldsymbol{x})$ is approximated near the new instance $\boldsymbol{x}_{new}$ using a linear function $\hat{F}(\boldsymbol{w},\boldsymbol{x})$, expressed as

$$\begin{aligned}\hat{F}(\boldsymbol{w},\boldsymbol{x}) &= w_0x_0 + w_1x_1 + \ldots + w_mx_m \\ &= \sum_{i=0}^{m} w_ix_i\end{aligned} \tag{2}$$

where $\boldsymbol{w} = (w_0 \; w_1 \; \ldots \; w_m)^T$ is the regression coefficient vector, and $x_0 \equiv 1$. Let $\boldsymbol{S}$ denote the set of k nearest instances corresponding to $\boldsymbol{x}_{new}$, and $d(\boldsymbol{x},\boldsymbol{x}_{new})$ represent the distance between $\boldsymbol{x}$ and $\boldsymbol{x}_{new}$, respectively. The regression coefficient vector $\boldsymbol{w}$ is then specified in order to minimise the squared error summed over the set $\boldsymbol{S}$ and weighted by the distance between each instance in $\boldsymbol{S}$ and $\boldsymbol{x}_{new}$, as

$$\begin{aligned}E(\boldsymbol{w},\boldsymbol{x}) = \frac{1}{2}\sum_{x\in S} &(F(\boldsymbol{x}) \\ &- \hat{F}(\boldsymbol{w},\boldsymbol{x}))^2 h(d(\boldsymbol{x},\boldsymbol{x}_{new}))\end{aligned} \tag{3}$$

where $h(\cdot)$ is some decreasing function of $d(\cdot,\cdot)$ and is used as weighting the error of each instance.

Obviously, instance-based approaches never construct an explicit general description regarding the target function. Instead, they can form a different local approximation to the target function, which is based on each distinct query instance and its relationship to the known example instances. A different set of example instances may be retrieved

when a new instance is observed, and this then needs to be classified.

Case-based reasoning is also an important approach of instance-based learning. In contrast to the relatively simple *k*-nearest neighbour algorithm and locally-weighted linear regression, CBR methods may use more general and rich symbolic descriptions to represent instances or cases. Also, more complex similarity metrics, such as the syntactic similarity measures, may be employed, and the target function to be 'approximated' may be a functional and logical relationship, or description, rather than the discrete-valued or real-valued analytical functions. Therefore, CBR is a more complicated instance-based learning process. Its accomplishment may be more knowledge-intensive and rely on an effective combination of statistical, knowledge-based reasoning and other learning approaches.

In the ANN research area, several types of network models exist, which are closely related to instance-based learning methods, and therefore may be used as a heuristic example to link and compare CBR to ANN approaches. The Radial Basis Function network [20,21], for example, is a typical ANN learning paradigm which can synthesise local messages (locally-weighted distances) to construct a description to a given target function.

Usually, the regression function expressed by RBF may be described as

$$\hat{F}(\boldsymbol{w},\boldsymbol{x}) = w_0 + w_1 h_1[d(\boldsymbol{x}_1,\boldsymbol{x})] + \ldots + w_k h_k[d(\boldsymbol{x}_k,\boldsymbol{x})] = w_0 + \sum_{i=1}^{k} w_i h_i[d(\boldsymbol{x}_i,\boldsymbol{x})] \quad (4)$$

where $\boldsymbol{w}$ is the regression coefficient vector, $\{h_i[d(\boldsymbol{x}_i, \boldsymbol{x})]\}$ is a set of kernel functions, and $\{\boldsymbol{x}_i\}$ is a set of example instances or training patterns chosen from the entire set of sample patterns. k is an integral constant that specifies the number of kernel functions to be included. Typically, the kernel function is specified as an n-dimensional Gaussian function centred at the point $\boldsymbol{x}_i$ with some variance σ_i^2. This may be shown as

$$h_i[d(\boldsymbol{x}_i,\boldsymbol{x})] = \exp\left\{-\frac{d^2(\boldsymbol{x}_i,\boldsymbol{x})}{2\sigma_i^2}\right\} = \exp\left\{-\frac{\left\|\boldsymbol{x}-\boldsymbol{x}_i\right\|^2}{2\sigma_i^2}\right\}, \; i = 1,2,\ldots,k \quad (5)$$

Thus, $h_i[d(\boldsymbol{x}_i,\boldsymbol{x})]$ measures similarity: the larger the radial distance between $\boldsymbol{x}$ and $\boldsymbol{x}_i$, the smaller the value of $h_i[d(\boldsymbol{x}_i,\boldsymbol{x})]$.

The topology of the RBF network is depicted in Fig. 1. The network is a two-layer feedforward network with k hidden nodes. The nonlinear transfer function of each hidden node is described by a kernel function. The network output is obtained by a linear combination of the outputs of all hidden nodes with $\{w_i\}$ as the parameters.

Let $\boldsymbol{X}$ denote the entire set of N sample patterns. Each element $\boldsymbol{x}_i$ in $\boldsymbol{X}$ is used as a training pattern and centralises a kernel function. Using the same criterion as Eq. (3), the optimal set of regression coefficient $\{w_i\}$ is to be found to minimises the error

$$E(\boldsymbol{w},\boldsymbol{x}) = \frac{1}{2}\sum_{\mathbf{x}\in\mathbf{X}}\left(F(\boldsymbol{x}) - \sum_{i=1}^{N} w_i \exp\left\{-\frac{\left\|\boldsymbol{x}-\boldsymbol{x}_i\right\|^2}{2\sigma_i^2}\right\}\right) \quad (6)$$

In such a way, a RBF network provides a global approximation to the target function over the entire sample space. In particular, the approximation is formed by the linear combination of many local approximations. Each local approximation is represented by a kernel function localised to a region near its particular centre with a certain radical width. The sign and magnitude of each regression coefficient describes the contribution from each local approximation.

On the other hand, it should be noticed that the multiple local approximations that the RBF constructs are only specifically targeted to the known learning patterns. When a new query instance is observed, the trained RBF determines a global approximation for the instance directly, by using its generalisation capability obtained through previous

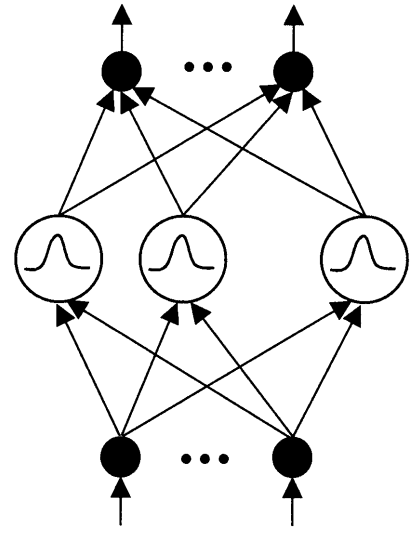

Fig. 1. The topology of a RBF network.

learning procedure. In other words, for any unknown future query pattern, the network has already formed its approximation at training time rather than at query time. For this reason, learning conducted in the RBF is called an *eager* learning method, and in this same sense, most learning methods in ANN are eager methods.

It is still an open problem as to whether there are essential differences in system performance and generalisation capability that can be achieved by lazy versus eager learning methods. Such a fundamental question concerns the effectiveness of applying an ANN in CBR system design. A further discussion will be included in Section 5.2.

To summarise, CBR is a more general representation of the instance-based learning process. In essence, CBR is a lazy learning paradigm. A RBF network can be viewed as an inspiring example of the relation and difference between CBR and ANN learning approaches. To apply ANN to CBR more effectively, some basic problems concerning the learning systems, based on the lazy versus eager methods, need to be addressed.

4. Connectionist Approaches for CBR System Design

In this section, some representative network models and methods employed for building CBR systems are introduced and compared. A common feature among these approaches is then summarised.

In general, the parallel computing structures and neural network models used for CBR system design can be either deterministic or non-deterministic; the connection weights of the network can be learned through either supervised or unsupervised training procedures. Based on the combination of these two factors, a taxonomy of neural network based approach for CBR system design is given in Fig. 2.

4.1. Some Basic Frameworks

4.1.1. Deterministic Model. A parallel computing network structure has been proposed for case-based reasoning [13]. As shown in Fig. 3, the network is a feedforward network with one hidden layer, and for a new query, is designed to make a specific choice, or a design decision, based on a set of stored cases. A problem under consideration is viewed as a point in an m-dimensional feature space which determines the number of input nodes in the net-

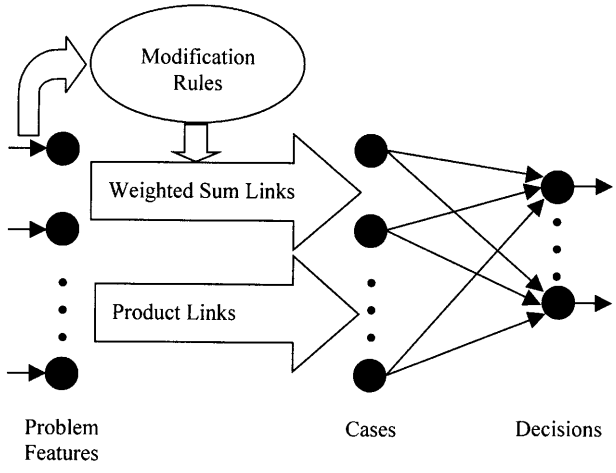

Fig. 3. A layered neural network model for CBR.

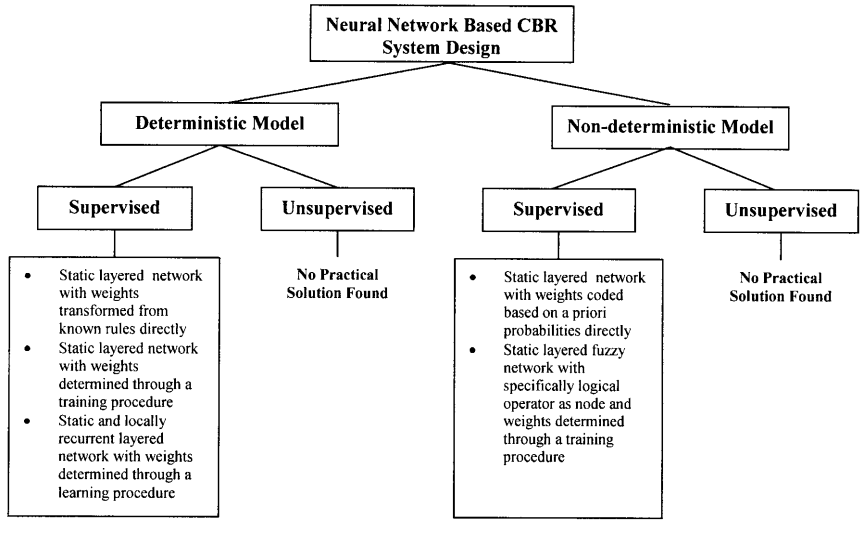

Fig. 2. A taxonomy of neural network based CBR system design.

work. Each hidden node and output node represent an old case and a specific decision choice, respectively.

The components or attributes of the problem feature vector are characterised by two types which are related to the activation level of hidden nodes. One is absolutely required for exciting a set of hidden nodes, whilst inhibiting other hidden nodes; the other is not absolutely required, but has a certain degree of importance. Consequently, the links between input and hidden layers are divided into two types: *product* and *weighted sum* links.

The product links connect the input nodes, which represent the absolutely necessary attributes for hidden node activation, to the corresponding hidden nodes. These links provide a pruning function that eliminates some hidden nodes from the decision-making process. The weighted sum links connect the input nodes, which possess a certain degree of importance for hidden node activation, to the relating hidden nodes. The importance is the weight on the link. Thus, for a given network input, the output of a specific activated hidden node is the linear combination of the attributes multiplied by the weighted sum links.

In addition, when an attribute takes some specific values, the importance (i.e. the weight on the weighted sum link) of some other attributes should be adjusted by using a Modification Rule and Conjunctive Antecedent.

In the output layer of the network, each node is specified by a simple linear function. With a predetermined weight between each two nodes, the hidden and output layers are fully connected. Such a connecting weight suggests a measure to which a stored case is relevant to a specific design choice.

When a new query problem is presented to the network, the value of each attribute in the given problem feature vector is detected. On this basis, a set of hidden nodes is excited (i.e. a set of old cases is retrieved), and then relating design choices are suggested. Note that the higher the activation level of an activated hidden node, the higher the degree of similarity between the new problem and the old problem contained in the retrieved old cases. Meanwhile, the old cases vote for different choices in proportion to their similarity degree to the new problem.

In such a way, the network defines a mapping between problem and design choice, which is based on the experience and knowledge concerning the known design cases. Additionally, by constructing different mapping forms, the network can be used as direct feature-based reasoning, case-based reasoning, and combined reasoning, respectively.

The idea here is simple: transforming known rules and relations among problems, cases and decision choices into a parallel computing structure. To realise such a transformation, this method uses a set of rules to determine the connecting weights and node states (especially the links between input and hidden nodes, and the activation level of hidden nodes). Consequently, it is difficult to apply the method to more complicated situations, where problems are expressed in a high dimensional vector space, and problem–problem and problem–case relationships are too complex to describe analytically.

To make the method more effective, the network learning property should be utilised for the transforming process instead of the rules. In fact, the above network can be easily realised by training a standard three-layer feedforward network [18,19,21]. Also, by using different transfer functions for hidden nodes, the trained network can be used for combined reasoning [27].

A more general multilayered network model with hybrid connections (feedforward and backward connections) has been described [14]. In this model, the input layer, hidden layer and output layer represent factors (features), stored cases and actions (decisions), respectively. When necessary, an additional hidden layer may be added between the factor and case layers. This hidden layer can be used to describe the complex dependence of cases on factors, and its output is viewed as *hidden* or *computed* factors. In addition, there are additional hidden units called *adaptons* between the factor and action units. Connections associated with these units are from actions to adaptons, factors to adaptons, and adaptons to actions. The structure of this network model is illustrated in Fig. 4.

Two stages are included in the network training process. First, with a set of factor-case patterns, the network is trained to determine the connecting weights between factor and case units by using the

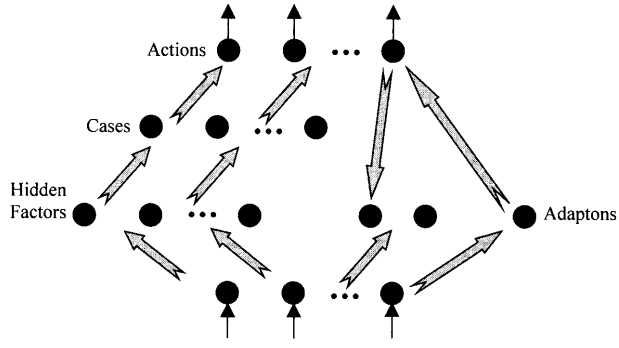

Fig. 4. A layered neural network model with hybrid connection for CBR.

error backpropagation algorithm [18]. The trained network is then able to perform case retrieval when a factor input is given. In the next stage, an action is to be associated with a particular case by directly setting connections from a case unit to action units.

When a new case, with a new set of factors, is presented to the network, the retrieved cases and associated actions may not fit the current case appropriately. For such a situation, a case adaptation process is introduced, where the adaptons units are used to modify the activity of the action units, based on factor values.

Compared with Becker and Jazayeri [13], this method properly integrates network learning with CBR system design. More importantly, a strategy for case adaptation is suggested. In essence, such a strategy aims at building a mapping with an appropriate form (linear, non-linear, static and dynamic, etc.) between case and action units. Related issues need to be addressed further.

4.1.2. Non-Deterministic Model. A probability theory-based network with probabilistic similarity metrics for case matching and adaptation has been presented [15]. The network is a hierarchical network with bottom and top layers. First, the network has been used for a deterministic situation, where a case is described as an m-dimensional real-valued feature vector. Suppose that l case feature vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l$ are selected as sample cases from a case base, where $\boldsymbol{x}_k = (x_{k1} \ldots x_{km})^T \in R^m$, $k = 1,2, \ldots, l$. Accordingly, the bottom layer contains m nodes, one for each component of a case vector, and the top layer consists of l nodes, one for each of the l cases, respectively. Furthermore, the connection weight from a bottom node to a case node is set to the value of the corresponding component of the case that the top node is associated with. In such a way, each sample case $\boldsymbol{x}_k$ is stored (coded) into the network as a set of weights attached to the connections linking all the bottom units to the corresponding case unit.

When an input case vector $\boldsymbol{x}_0 = (x_{01} \ldots x_{0m})^T$ is presented to the network at the bottom layer, a case matching process is performed to provide each case $\boldsymbol{x}_k$ with a similarity rank $S(\boldsymbol{x}_k)$, representing the similarity between the vectors $\boldsymbol{x}_k$ and $\boldsymbol{x}_0$. Usually, the similarity rank is a function depending upon the inner product of the case and input vectors, i.e. $S(\boldsymbol{x}_k) = F(\boldsymbol{x}_k^T \boldsymbol{x}_0)$. The case adaptation process is then conducted, based on the ranking of cases, which sends an output vector $\hat{\boldsymbol{x}}_0 = (\hat{x}_{01} \ldots \hat{x}_{0m})^T$ to the bottom nodes, as the result. In its simplest form, for example, the stored case vector with the highest ranking in $\{S(\boldsymbol{x}_k)\}$ is retrieved as the output vector. In addition, if an input vector is incomplete, its missing values can be filled in with values of the output vector.

To apply the network in a Bayesian framework, it is assumed that the database of problem domain knowledge is represented as m discrete attributes $A_1, \ldots, A_m$. Each attribute A_i is described as a random variable with n_i possible values from the set $\{a_{i1}, \ldots, a_{in_i}\}$. Additionally, all the cases $X = \{X_1, \ldots, X_l\}$ are regarded as binary random variables, with $X_k = 1$ denoting the fact that X_k is in question, $X_k = 0$ the opposite situation. A case feature vector is then coded as a vector:

$$\begin{aligned}\boldsymbol{x}_k &= [P_k(A_1),\ldots,P_k(A_m)] \\ &= [P_k(a_{11}),\ldots,P_k(a_{1n_1}),\ldots,P_k(a_{m1}),\ldots P_k(a_{mn_m})] \quad (7)\end{aligned}$$

where $P_k(A_i)$ is the conditional probability distribution for attribute A_i when case X_k is in question, and $P_k(a_{ij}) = P(A_i = a_{ij}|X_k = 1)$. Similarly, given initial probability distribution $\{P_0(a_{ij})\}$ for all the attributes, the input case feature vector $\boldsymbol{x}_0$ to the network can be expressed as

$$\begin{aligned}\boldsymbol{x}_0 = [&P_0(a_{11}),\ldots,P_0(a_{ln_l}),\ldots, \\ &P_0(a_{m1}),\ldots P_0(a_{mn_m})]\end{aligned} \quad (8)$$

Let X_0 denote the input case random variable; the following metrics are defined for the case matching and adaptation in the Bayesian network:

- *Bayesian case matching*: the rank score for a case X_k is the conditional probability $P(X_k = 1|X_0{=}\boldsymbol{x}_0)$.
- *Bayesian case adaptation*: the components of an output vector are the conditional probability $\{P(A_i = a_{ij}|X_0 = \boldsymbol{x}_0)\}$.

To realise the above CBR process, relevant probabilities $\{P_k(a_{ij})\}$ and $\{P(X_k) = 1\}$ for all the variables. $\{A_i\}$ and all the cases $\{X_k\}$ must be provided. Usually, these probabilities can be estimated based on the database at hand. Furthermore, to code all the case attributes $\{a_{ij}\}$ into the network, an additional intermediate layer between the bottom and top layers is added. This middle layer contains m groups of nodes, one for each attribute A_i. Each group has l nodes, one for each of the l cases. In addition, each node in the bottom and top layers is associated with one attribute of the attributes $\{a_{ij}\}$ and one case of cases $\{X_k\}$, respectively. The total number of nodes in the resulting three-layer network is $\Sigma_{i=1}^{m} n_i + lm + 1$. Let w_{ij}^k denote the weight between a unit a_{ij} in the bottom layer and a unit A_{ik} in the middle layer, $k = 1, \ldots, n_i, i = 1, \ldots, m$. Thus, by setting $w_{ij}^k = P_k(a_{ij})$, all the case feature vectors in the case base can be integrated and stored in the network.

When an input case feature vector $\boldsymbol{x}_0$ is presented,

a process for probability computation and propagation is conducted in the network with $P(X = X_k)$ as the initial values of the case nodes. The process consists of bottom-up and top-down phases. The bottom-up phase corresponds to the case matching process, which makes each case node excited with the matching score as its activity level. Based on the ranking of the stored cases, the following top-down phase performs the case adaptation process, where the computed probabilities are propagated from the top back down to the attribute values, and then used as the output of the network. This output can be understood as the complement or correction to the input case feature vector.

The importance of this method lies in that it introduces a parallel probability computation for the case matching and adaptation process, which allows all stored cases to contribute to the adaptation by the amount justified by their original matching. Also, when a given case feature vector is incomplete, i.e. values of one or more components are missing, or unknown, this approach can be used to retrieve, recover or predict these values. A main limitation of this method is that various *a priori* probabilities must be provided.

4.2. Case Indexing and Case Retrieval

Case indexing involves assigning a case's indexes that describe its features and distinguish it from other cases. Case indexing is important for case matching, retrieving and updating. An effective case indexing can facilitate case retrieval and case update in a case base.

In a connectionist framework, a case is usually described by a vector with features or attributes as its components. Therefore, several known approaches for case indexing and retrieval are mainly based on, to a certain extent, investigating and utilising the relationship between cases and features. As a result, case indexing and case retrieval can be integrated into a single network.

Perhaps the simplest method for case indexing and retrieval is to code or store a case as the connecting weights between two adjacent layers in a feedforward layered network, coupled with a specific choice of a node transfer function. Each node in the lower and higher layers represents one attribute and one case, respectively. The linked weight from each attribute node to a case node is set to the corresponding component of the case that the case node is associated with, whilst the function of each case node is specified by a simple linear function. When a new case is presented to the lower layer, a dot product of this new case vector and each old case, expressed by the weight vector, is calculated, respectively. The result is then shown as the activity level of the corresponding case node, and indicates the similarity between the new and old cases. Consequently, a set of cases with higher activity levels is retrieved. An example of this method can be found in Myllymaki and Tirri [15].

The main strength of this approach lies in its computational simplicity, and that it can naturally be integrated into the parallel structure of the network. On the other hand, however, it is not practical to store all the cases at hand into a network, especially when the cases are corrupted with noise. Therefore, an optimal strategy should be addressed for the proper choice of cases.

A more general and complex approach for case indexing and retrieval is to construct a mapping from the attribute domain (feature space) to the case space, based on a set of chosen attribute-case sample patterns. Various methods and network structures have been used for this purpose. In Becker and Jazayeri [13], knowledge about the relation between features and cases, and the dependencies among the features, are inducted to a set of rules. These rules are then transformed into a set of connecting weights between feature and case nodes, as well as a strategy to control the activity of case nodes, based on the activity of feature nodes. Thus, a non-analytical mapping from feature to case is established.

The multilayer feedforward neural network was introduced for case indexing and retrieval [9,16,17]. Because of its powerful non-linear mapping ability, the MFNN is particularly well suited for describing complicated feature-case relationships. To fulfil desired input-output requirements, the network is trained with a set of feature-case pairs. In particular the network can perform a multi-level case retrieval, which is more useful when features, sub-classes and general cases are structured in hierarchial levels. As shown in Main and Dillon [9], a number of MFNNs are used to describe a hierarchy of interwoven levels of cases, where the lowest level (i.e. the most fundamental level) corresponds to a set of features, and the highest level represents a set of general cases. The relationship between each two levels is normally non-linear. Beginning from the lowest level, one network is trained to perform a mapping from the chosen features to a set of sub-classes. A further network is then trained to map these sub-classes to a set of sub-classes at the following higher level. This is repeated upward for each two levels until the highest level classes are reached. When a current case with a set of features is input to the trained hierarchial network, output at each level of

the network can be used to aid in retrieving of a set of cases or sub-classes.

In addition, a feedforward three-layer fuzzy neural network has been suggested for case retrieval [28], where the transfer functions of hidden and output nodes are specified as logical operators, such as logical OR and AND operations. The connecting weights between each two adjacent layers represent the influence of nodes in the lower layer on the output of nodes in the upper layer. Essentially, such a method views the case retrieving as a multiple criteria decision making process.

4.3. Hybrid System

In CBR systems, general domain knowledge and particular knowledge represented by instances play an equally important role in problem solving. A hybrid architecture integrating a CBR system with a neural network component has been developed to use general and specific knowledge in the reasoning process [29,30]. In such a hybrid system, the neural network itself is not involved in case matching, case retrieval and the reasoning process. Instead, the network is trained to learn general domain knowledge and, as a result, the trained network can be used as a source of general knowledge. Furthermore, knowledge stored in the network is interpreted as a symbolic representation, which is combined with specific cases to support the reasoning.

The Combinational Neural Model (CNM) [31,32] was employed in the system. As shown in Fig. 5, the CNM has a feedforward topology with three layers. Each node represents a symbolic concept. The activation state of a node is described by a value in the interval [0,1], which can be interpreted as the degree of confidence that the network places in the concept represented by the specific node. Nodes in each two adjacent layers are connected by a set of weights, which can be understood as memberships possessed by the concepts described by the nodes in the lower layer to that described in the higher layer. When an input, represented by a set of fuzzy numbers, is presented, a logical operation is performed, layer-by-layer, in the network. Each input node takes either an excitatory or inhibitory state. An excitatory node sends the arriving signal x, multiplied by its weight. An inhibitory node preforms the fuzzy negation on the arriving signal, transforming it in $1\text{-}x$, before propagating it, using its weight as an attenuating factor. The combinational layer is formed by hidden fuzzy AND-nodes, which perform different combinations of the input signals. The output layer contains a set of OR-nodes. These nodes implement a competitive mechanism among the all different arriving signals.

The main advantage of using the CNM for learning general domain knowledge is that, because of the specific choices of transfer functions of nodes, the connecting weights have a clear meaning, and can easily be explained and justified. Therefore, knowledge embodied in the connecting weights can be further extracted and interpreted as a symbolic representation.

In contrast to Reategui and Campbell [29] and Reategui et al. [30], a hybrid system proposed in Lees et al. [33] uses a RBF network in the aid of the CBR adaptation process. Some similar hybrid systems can be found, (see, for example, Yao and He [34]).

In addition, some other work concerning eliciting symbolic knowledge representation from a trained neural network has been reported [35–37]. By using these methods, the neural network can be incorporated with a CBR system more effectively for problem-solving.

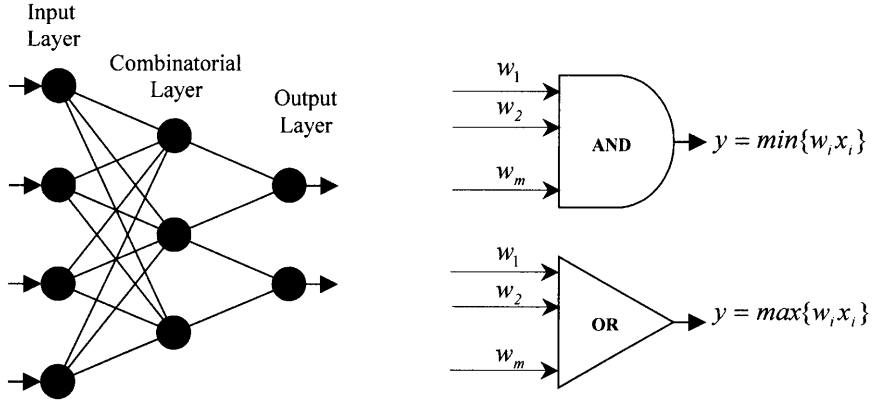

Fig. 5. The structure of the Combinational Neural Model (CNM) and node characters.

4.4. Summary

In this section, several connectionist-based models for general CBR system design, and for some specific aspects of CBR, are discussed. It has been shown that the applications of ANN models in the CBR study is fruitful. A common feature among these models is that a network is constructed to form a mapping from problem domain to solution space. The mapping can be linear, non-linear, static, dynamic, deterministic and probabilistic, depending on the specific problem domain. Such a mapping defines case matching and case adaptation metrics in a direct or indirect way. For this purpose, specific network structure and node transfer functions are used. Various knowledge and rules are coded or stored in the network by the transformation of rules, a network training process, or proper choices of connection weights. In such a way, the ANN approach integrates case indexing, retrieval and adaptation into a single network framework.

5. Future Research Directions

5.1. Fuzzy Neural Network

The fuzzy-set-theory-based neural networks have been used efficiently for aggregation of information in multi-criteria decision making systems, such as financial management and medical diagnosis [38]. In general, a fuzzy network can be trained to describe complex relationships among symbolic concepts.

As discussed in Section 4.3, the main advantage of employing the fuzzy neural networks for a CBR system design, or integrating the network into a CBR system as a component, is that information contained in the network can be easily understood and integrated. Such a property is extremely important for a reasoning system, so as to explain its reasoning results. Furthermore, it is possible to transform the knowledge embodied in the connecting weights into a set of rules or symbolic representations. In addition, using the fuzzy network can be an aid when handling complex uncertainty in engineering practice, which is the usual situation a CBR system encounters in real-world applications.

Compared with earlier work [28–30], a generalised logical operator was used as a node transfer function to construct a fuzzy-set-based hierarchical network model [39]. The operator can behave both as a union operator and an intersection operator, in addition to being a compensatory operator. Such a generalised hybrid operator is expressed by the so-called γ-model, as

$$y = \left(\prod_{i=1}^{m} x_i^{\delta_i} \right)^{1-\gamma} \left(1 - \prod_{i=1}^{m} (1 - x_i)^{\delta_i} \right)^{\gamma}, \quad 0 \leqslant \gamma \leqslant 1 \tag{9}$$

where variables $\{x_i | x_i \in [0,1]\}$ are m inputs to a node in the network, δ_i represents the connecting weight associated with x_i, $\Sigma_{i=1}^{m} \delta_i = m$, and variable y is the node output that depends on $\{x_i\}$ and $\{\delta_i\}$. γ is a parameter that controls the degree of compensation between the union and intersection operation parts.

Based on the γ-model, a more complex and effective hierarchical network can be constructed for a CBR system, in which case, further research should be focused on the following three aspects:

- Appropriate network structure for a given problem of interest, which includes the number of hidden layers and the number of hidden nodes in each hidden layer.
- Efficient network learning strategy for speeding up the network training. Also, the control parameter γ may be considered as a parameter to be determined through the network training process.
- Appropriate criteria for pruning some connections in the network, where the dependences among the nodes linked by these pruned connections are weak.

5.2. Radial Basis Function Network

The RBF network uses a linear combination (i.e. a linear additive) of many local approximations to approximate a given target function. These local approximations are formed by a set of nonlinear kernel functions based on given sample patterns. As usual, a supervised learning paradigm is used in the network training process, and for a pattern classification and recognition problem, the 1-*of*-*n* output encoding scheme is employed to determine the desired network output for each given network input pattern, where n is the number of output nodes in the network.

In addition, it should be noted that, for a trained network, the sign and magnitude of a connecting weight between each pair of hidden and output nodes represents the local contribution made by the associated hidden node to the given target function. Accordingly, the sample patterns which are used to centralise the kernel functions of hidden nodes can be divided into two groups: positive and negative patterns. It is obvious that, because of the linear combination, a local contribution made by some hidden nodes, associated with a subset of positive

patterns, may be counteracted by the other hidden nodes, associated with a subset of negative patterns. Therefore, identifying the role of each connecting weight and its associated hidden node is important for using the network effectively.

To apply the RBF network in a CBR system, some issues concerning the network re-training and learning sample updating should be addressed.

As analysed in Section 3, learning conducted in a RBF network is an eager learning approach, which is different from the lazy approach that the CBR system uses. Therefore, a theoretic investigation should be made on whether it is necessary to make the network 'lazy' instead of using the network's generalisation capability, or in other words, whether it is necessary to retrain the network when a new query pattern is presented to a trained network, even thought the corresponding network output matches a known output pattern for a predefined threshold? If yes, how to retain the network? How to define the desired network output corresponding to the new query pattern, and whether the new training can begin with the known weights obtained through the previous learning process?

On the other hand, if a new pattern fails to produce a desired output pattern under a given threshold, a strategy should be given concerning how to use it as a new learning pattern: assigning it to a known class of the training sample set, or considering it as a sample from a new patterns class, and accordingly, a new hidden node and output node should be added to the network?

5.3. Self-Organising and Competitive Learning Neural Networks

As far as we know, networks which use an unsupervised learning approach have not been employed for CBR system design (see Fig. 2 for reference). In fact, self-organising and competitive learning neural networks have been widely used in pattern feature extraction and classification. Such networks can create a meaningful coordinate system for different input features through an unsupervised learning paradigm and a competitive process among the network nodes [19,21]. Two representative examples of this type of network paradigm are the Self-Organising Map (SOM) network [40] and the Adaptive Resonance Theory (ART) network [41].

The SOM consists of an input layer and an output layer, with feedforward connections from input to output and lateral connections between nodes in the output layer. Usually, nodes in the input and output layers are arranged in a two-dimensional array, respectively. For a given input pattern, the output nodes compete among themselves to be activated or fired, with the result that only one output node is on. Through a competitive learning course, the SOM can capture the underlying structure of the input patterns, and perform a topology preserving map in which the spatial location of the activated nodes in the output array correspond to intrinsic topographic features of the input patterns.

Using the SOM in CBR system design, the high dimensional case feature vectors can be visualised (i.e. projected linearly) in a two-dimensional space, which is very helpful for understanding the underlying structure of the cases and analysing the relationship among cases. Further, based on some pattern matching criteria, for example, the k-nearest neighbour algorithm, case retrieval can be easily performed in the low dimensional space.

The ART network is specified by a two-layer network, where the input and output layers are labelled as comparison and recognition layers, respectively. There are two sets of connections, bottom-up and top-down connections, between the nodes in the two layers. A bottom-up connection associated with a recognition node respects a stored pattern. A binary version of the same pattern is also stored in a corresponding connection in the comparison layer, which is used to control the network training process, when necessary. In addition, three functional modules are used to provide control functions needed for the network training and classification phases.

When a new input vector is applied, only the output node with a weight vector best matching the input pattern is activated, and all other nodes are inhibited. Once this occurs, both the bottom-up and top-down connecting weights associated with the fired node are modified to make the stored patterns more like the input pattern. If no stored pattern matches the input pattern, within a predefined vigilance parameter, a new pattern category is created by adding a new node in the recognition layer and setting its associated weights to match the input pattern.

In this way, the ART resolves the stability-plasticity dilemma for a learning process: adding new additional pattern classes whilst keeping the stored patterns obtained through previous learning. Obviously, such a property is important for a CBR system. Therefore, applying ART in CBR systems is helpful in developing an effective strategy for case retrieval and case updating.

6. Concluding Remarks

The artificial neural network has established itself as a fruitful approach for developing an intelligent information processing system. In particular, ANNs have been viewed as a powerful tool in modern AI techniques. Over the last few years, a number of hybrid systems that combine the strengths of symbolic and connectionist approaches have been successfully developed.

In this paper, the ANN approaches used for CBR system design are reviewed. Several basic frameworks and network models for constructing a CBR system, and for some specific aspects in CBR, are introduced and analysed in detail. Also, further research directions based on some typical neural network paradigms are outlined. The most appealing property of the ANN-based approach for CBR is that the process for case index, matching, retrieval and adaptation can be integrated and conducted by a single network. Furthermore, because of the parallel computing structure and powerful learning ability, the specific information of cases can be compressed, then distributed into the network and represented indirectly by a set of connection weight vectors. As a result, the time taken to perform case matching and retrieval, and to produce a reasoning result, is much shorter. In addition, the network can be combined as a component with a general symbolic-based CBR system.

No doubt research on this important area will produce fruitful results which will facilitate CBR system design and promote the application of CBR in real-world problem-solving.

Acknowledgements. The authors would like to thank the anonymous reviewers for their constructive comments on an earlier version of this paper.

References

1. Reisbeck CK, Schank RC. Inside Case Based Reasoning. Lawrence Erlbaum, 1989
2. Kolodner J. Case-Based Reasoning. Morgan Kaufmann, 1993
3. Aamodt A. Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Communication 1994; 7(1):39–59
4. Leake DB (Eds). Case-Based Reasoning: Experiences, Lessons, and Future Directions. AAAI Press hMIT Press, 1996
5. Watson I. Applying Case-Based Reasoning: Techniques for Enterprise Systems. Morgan Kaufmann, 1997
6. Bichindaritz I, Kansu E, Sullivan KM. Case-based reasoning in CARE-PARTNER: gathering evidence for evidence-based medical practice. Advances in Case-Based Reasoning: Proc 4th European Workshop on Case-Based Reasoning, Springer-Verlag, 1998; 334–345
7. Schmidt R, Gierl L. Experiences with prototype designs and retrieval methods in medical case-based reasoning systems. Advances in Case-Based Reasoning: Proc 4th European Workshop on Case-Based Reasoning, Springer-Verlag, 1998; 370–381
8. Nakhaeizadeh G. Learning prediction of time series: a theoretical and empirical comparison of CBR with some other approaches. Topics in Case-Based Reasoning: Proc 1st European Workshop on Case-Based Reasoning, Springer-Verlag, 1993; 65–76
9. Main J, Dillon TS. A hybrid case-based reasoner for footwear design. Case-Based Reasoning Research and Development: Proc 3rd International Conference on Case-Based Reasoning, Springer-Verlag 1999; 497–509
10. Krampe D, Lusti M. Case-based reasoning for information system design. Proc 2nd International Conference on Case-Based Reasoning 1997; 63–73
11. Bradburn C, Zeleznikow J. The application of case-based reasoning to the tasks of health care planning. Topics in Case-Based Reasoning: Proc 1st European Workshop on Case-Based Reasoning, Springer-Verlag, 1993; 365–378
12. Arts RJ, Rousu J. Towards CBR for bioprocess planning. Advances in Case-Based Reasoning: Proc 3rd European Workshop on Case-Based Reasoning, Springer-Verlag 1996; 16–27
13. Becker L, Jazayeri K. A connectionist approach to case-based reasoning. Proc Case-Based Reasoning Workshop, Morgan Kaufmann, 1989; 213–217
14. Thrift P. A neural network model for case-based reasoning. Proc Case-Based Reasoning Working, Morgan kaufmann, 1989; 334–337
15. Myllymaki P, Tirri H. Massively parallel case-based reasoning with probabilistic similarly metrics. Proc 1st European workshop on Case-Based Reasoning, Springer-Verlag, 1993; 144–154
16. Micarelli A, Sciarrone F. A case-based systyem for adaptive hypermedia navigation. Proc European Workshop on Case-Based Reasoning, Springer-Verlag, 1996; 266–279
17. Micarelli A, Sciarrone F, Ambrosini L, Cirillo V. A case-based approach to user modeling. Proc European Workshop on Case-Based Reasoning, Springer-Verlag, 1998; 310–321
18. Rumelheart D, Hinton G, Williams R. Learning internal representation by error propagation. In: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol 1. MIT Press, 1986
19. Lippmann RP. An introduction to computing with neural nets. IEEE Trans Acoustics, Speech and Signal Processing 1987; 4: 4–22
20. Powell M. Radial basis function for multivariable interpolation: A review. In: Mason J, Cox M. (Eds), Algorithms for Approximation. Clarendon Press, 1987; 143–167
21. Haykin S. Neural Networks: A Comprehensive Foundation. Macmillan College Publishing, 1994
22. Gallant SI. Neural Network Learning and Expert Systems. MIT Press, 1993
23. Honavar V, Uhr L (Eds). Artificial Intelligence and

Neural Networks: Steps towards Principled Integration. Academic Press, 1994
24. Goonatilake S, Khebbal S (Eds). Intelligent Hybrid Systems. Wiley, 1995
25. McGarry K, Wermter S, MacIntyre J. Hybird neural systems: from simple coupling to fully integrated neural networks. Neural Comput Surv 1999; 2:62–93
26. Mitchell TM. Machine Learning. MIT Press/McGraw-Hill, 1997
27. Chen D. Studies and Applications of the Nonlinear Mechanism of Multilayer Feedforward Neural Networks. PhD Thesis, Northwestern Polytechnical University, Xian, China, 1993
28. Kraslawski A, Pedrycz W, Nystrm L. Fuzzy neural network as instance generator for case-based reasoning system: an example of selection of heat exchange equipment in mixing tanks. Neural Comput & Applic 1999; 8:106–113
29. Reategui EB, Campbell J. A classification system for credit card transactions. Proc 2nd European Workshop on Case-Based Reasoning, Springer-Verlag, 1994; 280–291
30. Reategui EB, Campbell J, Borghetti S. Using a neural network to learn general knowledge in a case-based system. Proc International Conference on Case-Based Reasoning, 1995; 528–537
31. Machado RJ, Rocha AFD. The combinational neural network: a connectionist model for knowledge based systems. Proc 3rd International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, 1990; 578–587
32. Beategui E, Leao B. Integrating neural networks with the formalism of frames. Proc World Congress on Neural Networks, 1993; 2:172–175
33. Lees B, Kumar B, Mathew A, Corchado J, Shiha B, Pedreschi R. A hybrid case-based neural network approach to scientific and engineering data analysis. Proc Eighteenth Annual International Conference of the British Computer Society Specialist Group on Expert Systems, Springer-Verlag, 1998; 245–260
34. Yao B, He Y. A hybrid system for case-based reasoning. Proc World Congress on Neural Networks, 1994; 4:442–446
35. Egri PA, Underwood PF. HILDA: Knoweldge extraction from neural networks in legal rule based and case based reasoning. Proc IEEE International Conference on Neural Networks, 1995; 1:1800–1805
36. Sestito S, Dillion T. Knowledge acquisition of conjunctive rules using multilayered neural networks. Int J Intelligent Systems, 1993; 8:779–805
37. Towell GG, Shavlik JW. Extracting refined rules from knowledge-based neural network. Machine Learning, 1993; 13:71–101
38. Kosco B. Neural Networks and Fuzzy Systems. Prentice Hall, 1992
39. Krishnapuram R, Lee J. Fuzzy-set-based hierarchical networks for information fusion in computer vision. Neural Networks 1992; 5:335–350
40. Kohonen T. The self-organizing map. Proc IEEE, 1990; 78:1464–1480
41. Carpenter G, Grossberg S. A massively parallel architecture for a self-organizing neural pattern recognition machine. Computer Vision, Graphics and Image Process 1987; 37:54–115