Soft Comput (2006) 10: 796–804
DOI 10.1007/s00500-005-0009-7

FOCUS

Ning Xiong · Peter Funk

Construction of fuzzy knowledge bases incorporating feature selection

Published online: 12 October 2005

Abstract Constructing concise fuzzy rule bases from databases containing many features present an important yet challenging goal in the current researches of fuzzy rule-based systems. Utilization of all available attributes is not realistic due to the "curse of dimensionality" with respect to the rule number as well as the overwhelming computational costs. This paper proposes a general framework to treat this issue, which is composed of feature selection as the first stage and fuzzy modeling as the second stage. Feature selection serves to identify significant attributes to be employed as inputs of the fuzzy system. The choice of key features for inclusion is equivalent to the problem of searching for hypotheses that can be numerically assessed by means of case-based reasoning. In fuzzy modeling, the genetic algorithm is applied to explore general premise structure and optimize fuzzy set membership functions at the same time. Finally, the merits of this work have been demonstrated by the experiment results on a real data set.

Keywords Fuzzy rule-based systems · Feature selection · Case-based reasoning · Fuzzy modelling · Genetic algorithm

1 Introduction

Fuzzy rule-based systems have been widely applied to control [9], modeling [12] and classification [3] problems. Rules based on fuzzy logic provide an effective means to describe uncertain, ill-defined systems using vague and imprecise statements. The major merits of fuzzy linguistic rules [19] lie in the close relationship to human reasoning and the facility of robust and flexible representation of knowledge in countless practical situations.

Generating fuzzy if-then rules from numerical examples gains great importance when no explicit human expertise is available. The purpose is to extract useful knowledge and information hidden somewhere from historical data. Data based knowledge discovery demonstrates various application potentials, including behavioral cloning of human operation skills, gaining insights into complex processes, identifying prospective customers in e-commerce, as well as understanding behavior of internet surfers to mention only a few cases.

Frequently, real-world scenarios produce great amounts of data with ten or even hundreds of features. Such high input dimension would make decision borders very complex and thus numerous rules required. An over-sized knowledge base is undesired due to interpretation difficulty apart from heavy computational burden. How to deal with large databases for deriving compact and comprehensible fuzzy rule sets turns out to be a challenging issue for intensive research.

Some attempts were made to determine the inputs of fuzzy-rule based systems by seeking the best feature combination relative to the performance of fuzzy modeling. Sugeno and Yasukawa [14] proposed to select fuzzy model inputs according to a regularity criterion (which entails fuzzy modeling on two separate data subsets). Grid-based premise structure was adopted in [11] for evaluating usefulness of feature subsets in fuzzy classification. Takagi and Hayashi [15] employed fuzzy neural networks to identify significant input variables by eliminating irrelevant features successively. A common property of these methods is that they all entail training fuzzy/neural network models for performance assessments. They are computationally expensive as repeatedly requesting to build system models for many feature subsets under consideration. A different method which enables assessing individual features in parallel was given in [13] where an information measure was defined to quantify the impacts of input features on the decision process, leading to a possible input dimension reduction. However, it is a sort of posteriori data analysis requiring the establishment of a fuzzy model with full features in advance.

This paper advocates incorporating feature selection into fuzzy knowledge base construction directly. The general framework is shown in Fig. 1 in which feature selection is treated as the first step to filter out irrelevant and unnecessary features and fuzzy modeling as the second step to build fuzzy

N. Xiong · P. Funk
Department of Computer Science and Electronics,
Mälardalen University, SE-72123 Västerås, Sweden
E-mail: ning.xiong@mdh.se

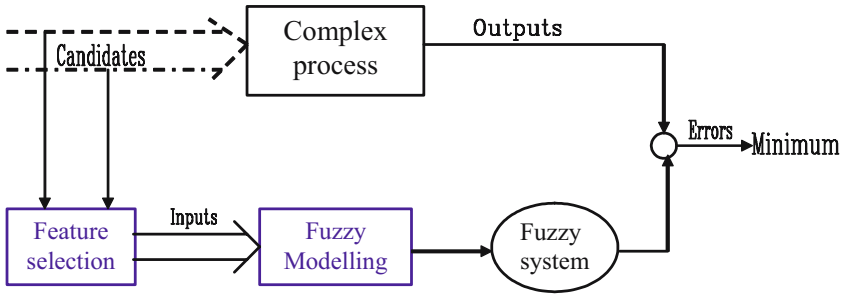

Fig. 1 Construction of fuzzy knowledge bases incorporating feature selection

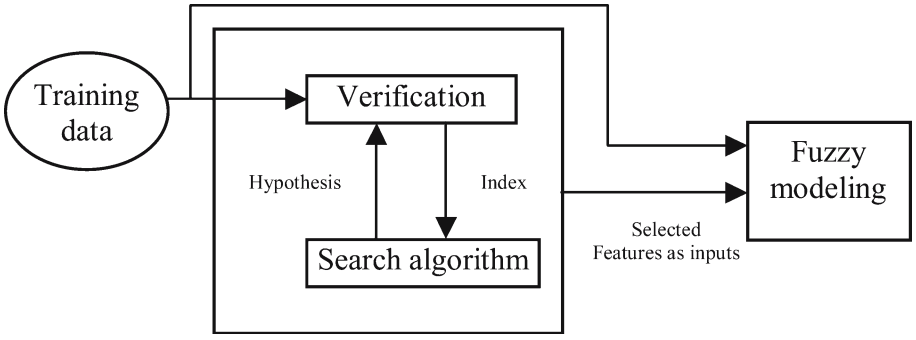

Fig. 2 The proposed sketch for feature selection

rule-based systems using selected features as inputs. Feature selection prior to fuzzy modeling brings the following benefits:

1. System accuracy and reliability can be improved by discarding irrelevant features that intrinsically have no influences on process outcomes at all;
2. The use of a smaller set of features reduces the problem space of fuzzy modeling as well as the complexity of the generated fuzzy system;
3. A reduced input dimension leads to requirement of fewer training data and helps to avoid the risk of over-fitting.

Several related but independent works have been reported concerning selection of significant features as inputs to fuzzy rule-based systems. Casillas et al. [2] presented a genetic feature selection process which is integrated into a multistage genetic learning method for fuzzy rule-based classification systems. Such a feature selection process uses the so-called k-NN rule to evaluate feature subsets and specifies, a priori, the number of features to be selected. Sorting features prior to building the final fuzzy classifier was addressed in both [6] and [10]. A heuristic method was introduced in [6] to assess the fitness degrees of individual features, while [10] calculated fuzzy entropy for every underlying attribute as a measure of its discriminating power. Evaluating features in isolation of each other simplifies the problem on one side whereas ignores possible interdependence among certain features or their joint effects on the other side.

The rest of the paper is organized as follows. Sect. 2 explains the details of feature selection including hypothesis testing and search algorithms. Fuzzy modeling using selected features as inputs is described in Sect. 3. In Sect. 4, we show the experimental results with Wine database. Finally, the concluding remarks are given in Sect. 5.

2 Feature selection as input identification

The task of feature selection for deciding system inputs can be considered as a problem of finding a correct hypothesis about key features which must be accepted. As every attribute will be either accepted or rejected, the total number of hypotheses for K features is 2^K . Evidently, the number of hypotheses will become too large for us to check every of them exhaustively when K is of high magnitude.

We aim at learning reliable hypotheses through a systematic mechanism. The basic sketch proposed is outlined in Fig. 2, where a certain search algorithm is applied to search in the hypothesis space for optimal or reliable solutions. The result from the search process appears as a feature subset containing selected attributes to be used as system inputs in the second step of fuzzy modeling.

The purpose of the "verification" block in Fig. 2 is to provide hypothesis evaluation. It is required to give a numerical index about how well an underlying hypothesis behaves. The intention is to verify the hypothesis based on the training data to see how well it is consistent with the available examples. As the verification procedure serves as trial evaluation for the search algorithm, it should not only be simple in nature but also guarantee a rough equivalence between (reliable) hypotheses and (good) performance indexes. In other words, the verification procedure must meet the following requirements:

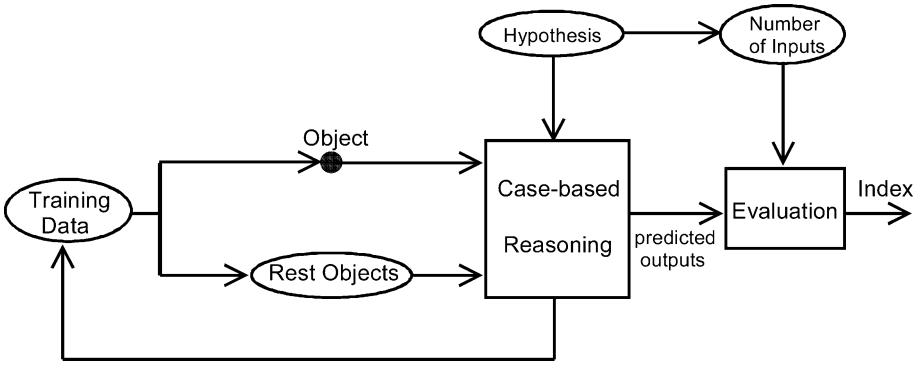

Fig. 3 Verification of a hypothesis through case-based reasoning

1. replying good performance indexes for reliable hypotheses;
2. replying bad performance indexes for absurd hypotheses;
3. being computationally simple for implementation.

The answer of how to verify hypotheses might not be unique. Any data analysis scheme is adequate to serve this purpose as long as it satisfies the above three requirements. Here in this paper, we suggest one alternative via case-based reasoning [1,8,18] as shown in Fig. 3. The main idea is to transform the problem of hypothesis verification to the one of checking the quality of case-based reasoning, which relies on attributes adopted by the hypothesis to reason about the outcome of an object from a set of known examples. Undoubtedly, a reliable hypothesis will give rise to high accuracy of case-based reasoning and vice versa. The performance of case-based reasoning under a hypothesis is estimated using the training data and by means of a "leave-one-out" procedure. This means that the outcome of an object from the training data is derived in terms of other cases in the same data set and this calculation is repeated for every training example. In this way, the performance estimation of the case-based reasoning can be expected to be both accurate and unbiased, since all samples in the training data are used for testing. On the other hand, in order to keep the input-dimension of a fuzzy system as low as possible, we should consider the number of attributes selected by the hypothesis as well. Hence the number of inputs is incorporated as the second criterion into the evaluation function to penalize any feature included.

2.1 Hypothesis testing via case-based reasoning

The general idea of verifying hypotheses by means of case-based reasoning has been highlighted previously. In this subsection we first give necessary details of the case-based reasoning method used for our purpose and then present an unbiased error estimate for hypotheses by checking results from the case-based reasoning.

Central component of case-based reasoning systems is a library of recorded cases. Each case in the library is identified by an index of features and an associated output. More formally we let $(a_1, a_2, \ldots, a_k)$ be a group of attributes used to index the cases and y be the outcome variable. A special case i in library can be represented by a $(K+1)$- tuple $(a_{i1}, a_{i2}, \ldots, a_{iK}, y_i)$ where a_{ij} corresponds to the value of the jth attribute for this case and y_i is the value in the outcome.

Generally speaking, case-based reasoning is tasked to determine the output of a new case based upon the library of historical cases. This new case can be represented by a probe consisting of a K-tuple $P=(u_1, u_{2,\ldots}, u_K)$ in which u_j corresponds to the value of the jth attribute in the probe. According to the given probe we are required to search the case library for neighboring cases. The outcomes of these neighboring cases are then fused to estimate the output for the current new case.

A neighborhood can be found from the library for the current new case via two essential steps. In the first step, the individual features of a case in the library are compared with the corresponding attributes in the probe to see how they are compatible with each other. This step is called attribute or feature matching and it results in a collection $[m_{i1}, m_{i2}, \ldots, m_{iK}]$, where $m_{ij} \in [0, 1]$ is a measure of compatibility on the jth attribute between the probe and the ith case in the library. The second step is aggregation of compatibility measures for individual attributes to acquire an overall matching value of a library case to the probe. Such a matching value can be regarded as a similarity degree between the new case and a remembered case. These similarity degrees are then utilized to establish a fuzzy neighborhood from the library for the new case being considered. Finally, the outcome for the current case is estimated according to its neighboring cases in history.

In the feature matching stage we must distinguish between relevant and irrelevant attributes. For an irrelevant attribute the measure of compatibility is always unity regardless of both attribute values. On the other side, the compatibility measure for a relevant attribute depends on the difference between two values concerned. The bigger this difference appears, the lower the compatibility value should be. A fuzzy set 'small' shown in Fig. 4 is adopted to define the measure of compatibility for a relevant feature. This means that the compatibility measure for a relevant feature is equal to the membership grade of the attribute difference to the fuzzy set 'small'. The parameter w in this membership

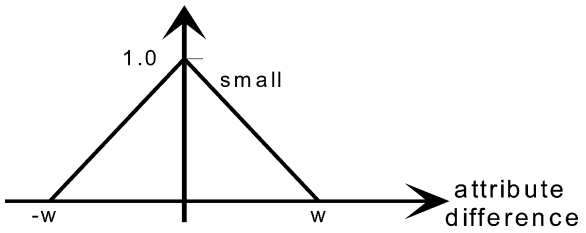

Fig. 4 The fuzzy set 'small' for defining compatibility measure

function is a coefficient to be specified by human operators. Since the purpose here is to examine the relative quality of hypotheses, the testing result is expected to be insensitive to the variations of w.

Obviously the measure of compatibility gets its maximal value of unity when there is no difference at all. In other cases, this measure decreases with the increment of the difference between two feature values until a zero membership degree is reached. Generally, the measure of compatibility m_{ij} for the jth attribute can be written as:

$$m_{ij} = \begin{cases} \mu_{small}(a_{ij} - u_j) & if\ a_j\ is\ relevant, \\ 1 & otherwise. \end{cases} \tag{1}$$

In the context of this article, case-based reasoning is carried out for hypothesis verification, such that it has to accept hypotheses as 'prior relevance information' for guiding the feature matching procedure, i.e. to determine the calculation of compatibility measure in (1). The accuracy of the case-based reasoning reflects the reliability of the hypothesis guiding it.

As a result of feature matching, we obtain a vector [m_{i1}, $m_{i2}, \ldots, m_{ik}$] for a given case in the library. Each element m_{ij} in this vector is a number in the unit interval indicating the truth value that the jth probe attribute is compatible with the jth attribute of the case being matched. The next problem is how to aggregate these individual measures to get an overall score, namely the similarity degree between the probe and the library case. Since similarity between case and probe is conditioned upon compatibility on all attributes, the similarity degree between library case and probe P should be a t-norm of the measures of feature compatibility. Using the operator of algebraic product as implementation of the t-norm, we have:

$$S_p(e_i, P) = m_{i1} \times m_{i2} \times \ \ldots \ \times m_{iK} \tag{2}$$

Given probe P its neighborhood can be built as a fuzzy subset upon examples in the library. A library case belongs to this neighbor set with a degree equal to its similarity measure with the probe. Let N be the number of examples in the library and denote e_i as the ith known case, the neighborhood of probe P can be notated as:

$$Neigh(P) = \left\{ e_1 \Big/ S_p(e_1, P),\ e_2 \Big/ S_p(e_2, P),\ , \right. \\ \left. \ldots, e_N \Big/ S_p(e_N, P) \right\} \tag{3}$$

Furthermore, since each case e_i has an associated outcome y_i, the score $S_P(e_i, P)$ provides a measure of appropriateness of solution y_i to the current case. The final stage of the case-based reasoning is to use this information to fuse solutions of neighboring cases to yield an estimate for the new case being inspected. Suppose that the output space is discrete: $\{C_1, C_2, \ldots, C_W\}$, we just need to pick one of the values as the estimation. For this purpose, we assign a voting strength (VS) to each discrete value $c \in \{C_1, C_2, \ldots, C_W\}$ by

$$VS(c) = \sum_{i=1}^{N} \begin{cases} S_p(e_i, P) & if\ y_i = c, \\ 0 & if\ y_i \neq c. \end{cases} \tag{4}$$

The outcome for probe P is estimated to be the value with the largest voting strength, i.e.,

$$Out(P) = \arg \max_{c \in \{C_1, C_2, \ldots C_W\}} VS(c) \tag{5}$$

The case-based reasoning technique stated above is applied for evaluating the performance of a hypothesis through a "leave-one-out" procedure. In this procedure one object from the training data set containing M samples is used for testing and the other $M-1$ samples constitute a library for case-based reasoning. We employ this library to reason about the outcome of the sample that is withheld for testing. Such a calculation is repeated for every training example and the portion of wrong judgments is treated herein as the error estimate for the hypothesis being evaluated.

More concretely, the output of every sample e_i in the training data is estimated based on the case library: $(\bigcup_{k=1}^{M} e_k) \backslash e_i$. Thus we can write:

$$\hat{y}_i = CBR\left\{ \left(\bigcup_{k=1}^{M} \mathrm{e_k} \right) \backslash \mathrm{e_i},\ (a_{i1}, a_{i2}, \ldots, a_{iK}) \right\} \tag{6}$$

where $\hat{y}_i$ is the estimated output for the sample e_i and *CBR* denotes a case-based reasoning function which accepts a library of $M-1$ training examples and a probe as its input elements to make classification about the output value. Finally by defining the function $dis(\cdot)$ as

$$dis(y_i, \hat{y}_i) = \begin{cases} 0 & if\ \hat{y}_i = y_i, \\ 1 & otherwise. \end{cases} \tag{7}$$

we acquire the error estimate for the hypothesis (under which the case-based reasoning is performed) as:

$$err = \left[\sum_{M}^{i=1} dis(\hat{y}_i, y_i) \right] \Big/ M \tag{8}$$

Usually this error estimate is combined with the penalty of selected features, producing a cost function of hypotheses in support of the search algorithms discussed in the next subsection.

2.2 Search algorithms for hypotheses

Having defined the assessment function for hypotheses, we now turn to discussing search engines for finding desirable ones. The search space is a state space (as exemplified in Fig. 5 with blackened circles denoting selected features) in

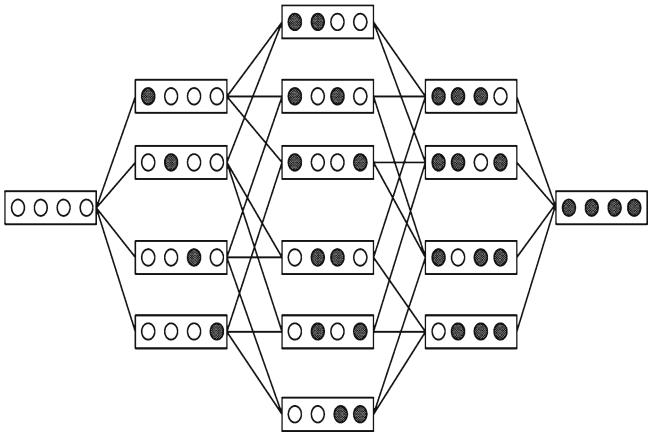

Fig. 5 State space and its operators for a four-attribute problem

which each state represents a hypothesis in the form of a feature subset and operators determine the connectivity between the states, i.e., adding or deleting a single feature with respect to a state. Two heuristic search algorithms (hill climbing and best-first), powerful for finding feature combinations, will be presented here. In principle, a search procedure might start with nothing and successively add attributes or begin with full attributes and successively remove part of them. However, considering the fact that case-based reasoning will become much faster under a hypothesis with few attributes selected, an empty feature subset is suggested being the initial state for our purpose.

2.2.1 Hill-climbing search engine

Hill-climbing is the simplest search technique, also called greedy search or steepest descent. It expands the current node and moves to the child with the lowest cost value, terminating when no child improves over the current node. A formal description of this algorithm is given below:

The hill-climbing search algorithm

1. Let $S \leftarrow$ initial state.
2. Expand S: apply all operators to S to produce its children.
3. Apply the cost function f to each child c of S.
4. Let S^* = the child with the lowest cost value $f(c)$.
5. If $f(S^*) < f(S)$ then $S \leftarrow S^*$, go to Step2.
6. Return S.

2.2.2 The best-first search engine

Best-first search is a more robust method than hill-climbing. The main idea is to choose the most promising node generated so far that has not already been expanded. It is worthy noting that the best-first search engine used for input identification varies slightly from the standard version, since there is no explicit goal condition in our problem. Best-first search usually terminates upon reaching the goal. Our goal is to find an optimal hypothesis about relevance of attributes, so the search can be stopped at any point and the solution found so far can be returned, thus making it an anytime algorithm. In practice, the search process must be stopped at some point and we can use here what is called a stale strategy: if an improved node has not been found in the last k expansions, the search is terminated. The best-first search engine devised for our purpose is formally described as:

The best-first search algorithm

1. Put the initial state in the OPEN list, CLOSED list $\leftarrow \emptyset$, Best $\leftarrow$ initial state.
2. Let $S = \arg\min_{c \in OPEN} f(c)$ (get the state from OPEN with minimal cost value).
3. Remove S from OPEN, and add S to CLOSED.
4. If $f(S) < f$(BEST), then BEST $\leftarrow S$.
5. If BEST changed in the last k expansions, go to the next step, otherwise return BEST.
6. Expand S: apply all operators to S to produce its children.
7. For each child neither in the OPEN list nor in the CLOSED list, evaluate and add it to the OPEN list.
8. Go to Step 2.

3 Fuzzy modeling using selected features

Given selected features $x_1, x_2, \ldots, x_n$, we ought to accept them as inputs in fuzzy modeling. The fuzzy sets for input x_j are represented by $A(j,1), A(j,2), \ldots, A(j, q[j])$ and $q[j]$ is the number of its linguistic terms. By $p(\bullet)$ we denote an integer function mapping from $\{1,2,..., s(s \leq n)\}$ to $\{1,2,...., n\}$ satisfying $\forall x \neq y, p(x) \neq p(y)$. Fuzzy rules that are to be generated herein are of the general form as:

$$
\begin{aligned}
&If\ [X_{p(1)} is \bigcup_{k \in D(1)} A(p(1), k)] AND \\
&\quad [X_{p(2)} is \bigcup_{k \in D(2)} A(p(2), k)] \\
&\quad AND \cdots AND [X_{p(s)} is \bigcup_{k \in D(s)} A(p(s), k)] \\
&\quad Then\ Conclusion\ B
\end{aligned}
\tag{9}
$$

where $D(i) \subset \{1, 2, \ldots, q[p(i)]\}$ for i=1 ,. . ., s, and $B \in \{C_1, C_2, \ldots, C_W\}$

If a premise includes all input variables in it (*e. g.* s=n), we say that its rule has complete structure, otherwise its structure is incomplete. Another important property of the rules in the form of (9) is that a union operation of input fuzzy sets is allowed in their premises. Rules having incomplete structure or containing OR connections of input fuzzy sets can achieve larger coverage of input domain, leading to substantial reduction of the number of rules [16].

3.1 Learning rule premises by genetic algorithms

The genetic algorithm (GA) introduced by Goldberg [5] is applied to search for general premises of rules. The approach is based on the work in [17], with the goal of taking advantage of the strength of genetic search to reach a set of suitable premise structures together with parameters of membership functions. The upper limit of the rule number needs to be

given by user in advance. It can be considered as an estimation of the sufficient amount of rules to achieve a satisfactory accuracy. During the running of GA the actual rule number can be adjusted automatically within this specified limit. In the following we state briefly about coding scheme, genetic operators and fitness function which present key points for the genetic learning.

3.1.1 Genetic coding scheme

The information concerning structure of rule premises can be considered as a set of discrete parameters, while the information about membership functions is described by a set of continuous parameters. Owing to the different natures between the information about rule structure and about fuzzy set membership functions, a hybrid string consisting of two substrings is used here as the coding scheme. The first substring is a binary code representing premise structure of a knowledge base, and the second substring is an integer code corresponding to parameters of fuzzy sets used by rules.

Usually membership functions of an attribute selected as input are characterized by a set of parameters. Each of these parameters can further be mapped by an integer through discretization. The resulting integers are then combined to form an integer-vector depicting the fuzzy partition of that input. Merging together integer-vectors for all inputs gives rise to the integer code as one part of the hybrid string. To ensure meaningful fuzzy partitions every newly generated integer vector in the integer code must be rearranged into an ascending or descending order.

From (9), we can see that premise structure of general rules is indeed decided by integer sets $D(i)$ (i=1, 2, ...s). This fact suggests that a binary code be a suitable scheme for encoding structure of premises, since an integer from $\{1, 2, \ldots, q[p(i)]\}$ is either included in the set $D(i)$ or excluded from it. For attribute x_j which is included in the premise (i.e. $p^{-1}(j) \neq \emptyset$), $q[j]$ binary bits need to be introduced to depict the set $D(p^{-1}(j)) \subset \{1, 2, ..., q[j]\}$, with bit "1" representing the presence of the corresponding fuzzy set in the OR-connection and vice versa. If attribute x_j does not appear in the premise, i.e. $p^{-1}(j)=\emptyset$, we use $q[j]$ one-bits to describe the wildcard of "don't care". For instance, the condition "*if* [x_1 =(*small* or *large)] AND* [x_3 = *middle*] *AND* [x_4= (*middle* or *large)*]" can be coded by the binary group (101; 111; 010; 011).

Further, the whole substring for the premise structure of the rule base is a merge of bit groups for all individual rule premises. It is worthy noting that the following two cases by a binary group lead to an *invalid premise* encoded: (1) All the bits in the group are equal to unity, meaning that no inputs are considered in the premise; (2) All the bits for an input are zero; this input thus takes no linguistic term in the premise resulting in an empty fuzzy set for the condition part of that input. Rules with invalid premises are meaningless, play no role in the fuzzy reasoning and therefore should be discarded. Through elimination of invalid rule premises, the actual rule number can be reduced from the upper limit given by man. This implies an opportunity to adjust the size of the rule base within certain constraint by GA.

3.1.2 Crossover

Owing to the distinct nature between the two substrings, it is preferable that the information in both substrings be mixed and exchanged separately. Here a three-point crossover is used. One breakpoint of this operation is fixed to be the splitting point between both substrings, and the other two breakpoints can be randomly selected within the two substrings, respectively. At breakpoints the parent bits are alternatively passed on to the offspring. This means that offspring get bits from one of the parents until a breakpoint is encountered, at which they switch and take bits from the other parent. The crossover rate used is around 0.867.

3.1.3 Mutation

Because of the distinct substrings used, different mutation schemes are needed. Since parameters of input membership functions are essentially continuous, a small mutation with high probability is more meaningful. Therefore it is so designed that each bit in the substrings for membership functions undergo a disturbance. The magnitude of this disturbance is determined by a Gaussian distribution function. For the binary substring representing structure of rule premises, mutation is simply to inverse a bit, replace '1' with '0' and vice versa. Every bit in this substring undergoes a mutation with a quite low probability around 0.033.

3.1.4 Fitness function

An individual in the population is evaluated according to both the performance and simplicity of the fuzzy system it yields. The simplicity here refers to the actual size of the rule base. In fact the number of valid rules is very likely to be smaller than the upper limit prescribed due to possible invalid premises encoded. The fitness function to evaluate a (hybrid) string can be constructed by combining the above two criteria in a simple linear way as:

$$Fit(HS) = Acc(HS) - \beta \cdot R_n(HS) \tag{10}$$

where $Acc(HS)$, $R_n(HS)$ denote respectively the classification accuracy on the training set and the actual rule number of the fuzzy system which is decoded from string HS. β is a coefficient determined by the user. Because the size of the rule base is now directly incorporated into the fitness function, GA searches for solutions with not only best performances but also minimal complexity.

In order to acquire the value of $Acc(HS)$ for fitness evaluation, we must first acquire the whole rule base for making classification of training examples. This entails determination of the appropriate rule conclusion under every valid premise after the hybrid string has been decoded. The next subsection describes such a procedure as an internal loop embedded in the genetic learning.

3.2 Determining consequence under a given premise

Given a rule premise we need to choose an appropriate consequent class from the finite set of possible classes. The idea is to choose such a consequence so that the resulting fuzzy if-then rule will become most truthful, i.e., obtaining the most evidences and support from the training examples. The technical details of this procedure are explained below.

By substituting the premise description of the rule in (9) with symbol $\tilde{A}$ under the definition as

$$\tilde{A} = [x_{p(1)} \underset{k\in D(1)}{\cup} A(p(1),k)] \; and \; [x_{p(2)} is \underset{k\in D(2)}{\cup} A(p(2),k)] \; AND \ldots AND \; [x_{p(s)} \; is \underset{k\in D(s)}{\cup} A(p(s),k)] \tag{11}$$

the rule can be abbreviated as " *If* $\tilde{A}$ *Then B*". The condition $\tilde{A}$, on the other hand, can be regarded as a fuzzy subset on the training set U_T=$\{e_1, e_2, \ldots, e_M\}$. The membership value of a training example to this fuzzy subset is equal to the degree to which $\tilde{A}$ is satisfied by its vector of selected features (inputs). Thus we write:

$$\tilde{A} = \left\{ \frac{e_1}{\mu_{\tilde{A}}(e_1)}, \frac{e_2}{\mu_{\tilde{A}}(e_2)}, \ldots, \frac{e_M}{\mu_{\tilde{A}}(e_M)} \right\} \tag{12}$$

$$\mu_{\tilde{A}} = \mu_{\tilde{A}}(x_{i1}, x_{i2}, \ldots, x_{in}) \quad i = 1, \ldots, M \tag{13}$$

Here $(x_{i1}, x_{i2}, \ldots, x_{iM})$ is the input vector of the example e_i in the training set. Similarly the conclusion B is treated as a crisp subset on the training set. An example belongs to this crisp set, if and only if its outcome is the same as B. Therefore the membership function of the subset for B is defined as:

$$\mu_B(e_i) = \begin{cases} 1 & if\, Out(e_i) = B \\ 0 & otherwise \end{cases} \qquad i = 1, \ldots, M \tag{14}$$

The rule "*If* $\tilde{A}$ *Then B*" corresponds to the implication of $\tilde{A} \Rightarrow B$, which is equivalent to the proposition that $\tilde{A}$ is a subset of B, i.e., $\tilde{A} \subseteq B$. In this view, the measure of subsethood of $\tilde{A}$ in B is utilized as the truth value of the rule. So, we obtain

$$truth(\tilde{A} \Rightarrow B) = \frac{M(\tilde{A} \cap B)}{M(\tilde{A})} = \frac{\sum\limits_{e\in U_T} (\mu_{\tilde{A}}(e) \wedge \mu_B(e))}{\sum\limits_{e\in U_T} \mu_{\tilde{A}}(e)} = \frac{\sum\limits_{\mathrm{Out}(e)=B} \mu_{\tilde{A}}(e)}{\sum\limits_{e\in U_T} \mu_{\tilde{A}}(e)} \tag{15}$$

where $M(\tilde{A})$ and $M(\tilde{A}, \cap B)$ indicate the cardinality measures of the sets $\tilde{A}$ and $\tilde{A}\cap$B , respectively.

Given a condition $\tilde{A}$, we choose the conclusion from the finite candidates such that the truth value of the considered rule reaches its maximum. This means that the rule consequence can be decided with the following two steps:

Step 1: Calculate the competition strength for each candidate as

$$\alpha(c) = \sum_{Out(e)=c} \mu_{\tilde{A}}(e), \quad c = C_1, C_2, \ldots, C_W \tag{16}$$

Step 2: Decide on conclusion B by maximizing the competition strength, i.e.,

$$B = \arg \max_{c\in\{C_1, C_2, \ldots, C_W\}} \alpha(c) \tag{17}$$

4 Experiments and results

In order to verify the performance of the method of hybridizing feature selection and premise learning, we made a case study on the well-known benchmark problem of wine classification. The wine classification data can be downloaded from the address as ftp.ics.uci.edu/pub/machine-learning-databases. It consists of 178 samples with 13 continuous attributes (a_i: $1 \le i \le 13$) from three classes. The task was to generate fuzzy classification rules based upon this data set.

4.1 Selecting features for wine classification

We first attempted to select significant variables from the 13 features. Heuristic search algorithms were applied to seek reliable hypotheses about feature relevance. The cost function utilized in experiments to evaluate hypotheses was defined as

$$f_c(Hp) = err(Hp) + 0.01 \cdot |Hp| \tag{18}$$

where $err(Hp)$ is the error estimate for hypothesis Hp and $|Hp|$ denotes the number of attributes selected by it. Clearly the second part of this cost function serves to encourage choosing as few attributes as possible for complexity reduction.

The procedure with the hill-climbing search is illustrated in Table 1 from which we can see that a feature subset containing a_1, a_7, a_{11}, and a_{13} was returned as the best hypothesis. We also utilized the more robust best-first algorithm

Table 1 The hill-climbing search procedure for feature selection

	The state S for expansion	f_c (S)	The best child S*	f_c (S*)
Step 1	Ø	—	(a_7)	0.290899
Step 2	(a_7)	0.290899	(a_1, a_7)	0.160449
Step 3	(a_1, a_7)	0.160449	(a_1, a_7, a_{11})	0.091798
Step 4	(a_1, a_7, a_{11})	0.091798	(a_1, a_7, a_{11}, a_{13})	0.062472
Step 5	(a_1, a_7, a_{11}, a_{13})	0.062472	(a_1, a_3, a_7, a_{11}, a_{13})	0.072472

Table 2 Performance of the rule sets from independent tests

Tests	Classification accuracy(%)	Number of rules
1	99.4382	5
2	98.8764	4
3	98.3146	5
4	98.3146	3
5	97.7528	3
6	99.4382	4
7	98.3146	4
8	98.3146	4
9	97.7528	3
10	98.8764	5

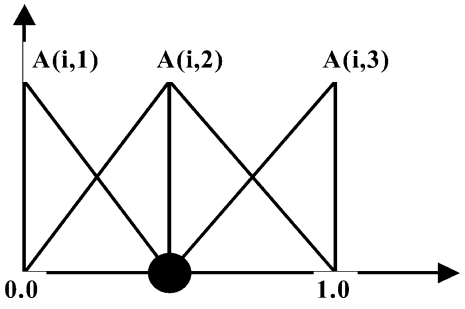

Fig. 6 The membership functions for input i

for investigation and the result turned out to be identical. As this is a problem of relatively small scale, heuristic schemes (hill-climbing and best-first) are believed to be adequate for arriving at satisfying solutions.

4.2 Modeling wine classification rules

The result of feature selection was then passed on to the second step of fuzzy modeling. That is to say that only attributes a_1, a_7, a_{11}, and a_{13} were accepted as inputs in building fuzzy classification rules. Each input i is associated with fuzzy sets $A(i, 1)$, $A(i, 2)$ and $A(i, 3)$. By normalizing the values of each input into the unit interval, the membership functions of input fuzzy sets are outlined as depicted in Fig. 6. The upper limit of the rule number in the rule base was set to 16, meaning that 16 rules were supposed to be sufficient to achieve a good classification accuracy. GA was then employed to search for the premise structure of possible rules and to optimize the parameters (corresponding to the circle in Fig. 6) of the input fuzzy sets at the same time. The coefficient β in the fitness function (10) for GA was fixed to be 0.005. In this way, we conducted 10 independent experiments and the performance of the rule sets learned are illustrated in Table 2.

The wine classification data were used by Corcoran and Sen [4] and Ishibuchi et al. [7] as well for verifying their genetic-based machine learning algorithms. However, both works accepted all the 13 attributes as inputs in the rule construction. The results reported therein are compared with those of this paper in Table 3. From this table, we can see that the size of the rule base from our work is surprisingly much lower than that from the others and the accuracy is roughly similar. This is very beneficial since the rule set has now become much more compact but high accuracy still remains. Besides, we should also note that in [4] a total number of 45,000 trials (with population size as 1,500 and 300 generations) were called for by GA, whereas in this paper only 10,000 individuals were visited with both the population size and the generation number being 100. This clearly shows the other advantage of reduction of computational complexity by our work.

It should be noted that all the results given in Tables 2 and 3 were derived from using the whole data base as the training set. Our purpose here is just to show the enhanced learning ability to acquire more compact fuzzy knowledge base without undermining modeling accuracy. Currently it seems not possible to make a fair comparison with [4] and [7] regarding classification accuracy on test data since such information is absent from both of them. However, we believe that, with a much lower number of inputs, our obtained fuzzy system should be more robust against the risk of over-fitting compared with those with all 13 features as inputs.

5 Conclusions

This paper deals with high-dimensional feature space in data-based fuzzy rule construction. The presented framework consists of two separate stages: feature selection upon known examples and fuzzy modeling using selected features as inputs. Identifying inputs as priori analysis is strongly suggested in that it can not only reduce computational complexity in fuzzy modeling but also result in simpler fuzzy systems with easier understanding. In the second stage of fuzzy modeling, GA-based premise learning is triggered enabling exploration of general premise structure and optimization of input membership functions simultaneously. The experimental studies on a real data set demonstrates that the roadmap of combined feature selection and premise learning can give rise to an excellently compact knowledge base yet still retaining very high accuracy.

Table 3 Comparison with other works

	Best Accuracy (%)	Worst Accuracy(%)	Average Accuracy(%)	Rule Number
Corcoran and Sen [4]	100	98.3	99.5	60
Ishibuchi et al. [7]	99.4	97.8	98.5	60
Xiong and Funk	99.4382	97.7528	98.5390	4 (average)

References

1. Aamodt A, Plaza E (1994) Case-based reasoning: foundational issues, methodological variations and system approaches. Artif Intell Commun 7(1):39–59
2. Casillas J et al. (2001) Genetic feature selection in a fuzzy rule-based classification system learning process for high-dimensional problems. Inform Sci 136:135–157
3. Chi Z, Yan H, Pham T (1996) Fuzzy algorithms with application to image processing and pattern recognition. World Scientific, Singapore
4. Corcoran AL, Sen S (1994) Using real-valued genetic algorithms to evolve rule sets for classification. In: Proceeding of 1st IEEE international conference on evolutionary computation, Orlando, FL, June 27–29, 1994, pp 120–124
5. Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, New York
6. Hong T-P, Chen J-B (1999) Finding relevant attributes and membership functions. Fuzzy Sets and Systems 103:389–404
7. Ishibuchi H, Nakashima T, Murata T (1999) Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. IEEE Trans Syst Man Cybern B 29(5):601–618
8. Kolodner J (1993) Case-based reasoning. Morgan Kaufmann, San Francisco, CA
9. Lee CC (1990) Fuzzy logic in control systems: Fuzzy logic controller. IEEE Trans Syst Man Cybern 20:404–435
10. Lee H-M et al. (2001) An efficient fuzzy classifier with feature selection based on fuzzy entropy. IEEE Trans Syst Man Cybern B 31(3):426–432
11. Nakashima T et al. (1997) Input selection in fuzzy rule-based classification systems. In: Proceeding of IEEE International Conference Fuzzy Systems, pp 1457–1462
12. Pedrycz W (ed) (1996) Fuzzy modeling: paradigms and practice. Kluwer, Norwell, MA, USA
13. Silipo R, Berthold M (2000) Input features' impact on fuzzy decision processes. IEEE Trans Syst Man Cybern B 30(6):821–834
14. Sugeno M, Yasukawa T (1993) A fuzzy-logic-based approach to qualitative modeling. IEEE Trans Fuzzy Syst 1:7–31
15. Takagi H, Hayashi I (1991) NN-driven fuzzy reasoning. Int J Approx Reason 5:191–212
16. Xiong N, Litz L (2002) Reduction of fuzzy control rules by means of premise learning – method and case study. Fuzzy Sets Syst 132:217–231
17. Xiong N, Litz L, Ressom H (2002) Learning premises of fuzzy rules for knowledge acquisition in classification problems. Knowledge Inform Syst 4(1):96–111
18. Yager RR (1996) A unified view of case based reasoning and fuzzy modeling. In: Fuzzy logic foundations and industrial applications, (ed) Ruan D Kluwer, Boston pp 5–26
19. Zadeh LA (1973) Outline of a new approach to the analysis of complex systems and decision processes. IEEE Trans Syst Man Cybern 3:28–44