Generating Linguistic Fuzzy Rules for Pattern Classification with Genetic Algorithms

N. Xiong and L. Litz

Institute of Process Automation, University of Kaiserslautern, Postfach 3049, D-67653
Kaiserslautern, Germany
{xiong,Litz}@e-technik.uni-kl.de

Abstract. This paper presents a new genetic-based approach to automatically extracting classification knowledge from numerical data by means of premise learning. A genetic algorithm is utilized to search for premise structure in combination with parameters of membership functions of input fuzzy sets to yield optimal conditions of classification rules. The consequence under a specific condition is determined by choosing from all possible candidates the class which lead to a maximal truth value of the rule. The major advantage of our work is that a parsimonious knowledge base with a low number of classification rules is made possible. The effectiveness of the proposed method is demonstrated by the simulation results on the Iris data.

1. Introduction

Fuzzy logic based systems have found various applications to solve pattern classification problems [8]. The task of building a fuzzy classification system is to find a set of linguistic rules, based on which the class of an unknown object can be inferred through fuzzy reasoning. The premise determines the structure of a rule and it corresponds to a fuzzy subspace in the input domain. A simple way to define fuzzy subspaces as rule conditions is to partition the pattern space by a simple fuzzy grid [7, 9]. This is, however, not suitable in the cases of high attribute dimension because the rule number increases exponentially with the number of attributes.

This paper aims at learning general premises of rules by a genetic algorithm (GA) to treat problems of multiple attribute inputs. The upper limit of the number of rules is predetermined by man in advance. It can be considered as an estimation of the sufficient amount of rules to achieve a satisfactory classification. The proposed modeling procedure for a fuzzy classifier consists of two loops. In the extern loop GA is utilized to search for optimal premise structure of the rule base and to optimize parameters of input fuzzy sets at the same time. The task of the intern loop is to determine the classes in the conclusions by maximizing the truth values of the rules. The effectiveness of our method is examined by the well known example of Iris Data [3] classification using linguistic fuzzy rules.

2. Fuzzy Classification System

Let us consider a K -class classification problem as follows. The objects or cases in the universe are described by a collection of attributes $x_1, x_2, \dots$, and x_n . The fuzzy sets of the attribute x_j ($j=1 \dots n$) are represented by $A(j,1), A(j,2), \dots, A(j, q[j])$, and $q[j]$ is the number of linguistic terms for x_j . Denote $p(\bullet)$ as an integer function mapping from $\{1,2,\dots,s(s \leq n)\}$ to $\{1,2,\dots,n\}$ satisfying $\forall x \neq y, p(x) \neq p(y)$. Fuzzy rules to be generated for classification can be formulated as:

$$\text{if } [x_{p(1)} = \bigcup_{j \in D(1)} A(p(1), j)] \text{ and } \dots \text{ and } [x_{p(s)} = \bigcup_{j \in D(s)} A(p(s), j)] \text{ then Class } B \quad (1)$$

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

If the rule premise includes all input variables in it (e.g. $s=n$), we say that this rule has a complete structure, otherwise its structure is incomplete. An important feature of the rules in form (1) is that an union operation of input fuzzy sets is allowed in their premises. Rules containing such OR connections in conditions can cover a group of related rules which use complete AND connections of single linguistic terms as rule premises. By substituting the premise description of the rule in (1) with symbol A , i.e.

$$A = [x_{p(1)} = \bigcup_{j \in D(1)} A(p(1), j)] \text{ and } \dots \text{ and } [x_{p(s)} = \bigcup_{j \in D(s)} A(p(s), j)] \quad (2)$$

the rule can be abbreviated as “If A Then B ”. The condition A , on the other hand, can be regarded as a fuzzy subset on the training set $U_T = \{u_1, u_2, \dots, u_m\}$. The membership value of an object to this fuzzy subset is equal to the degree to which A is satisfied by its attribute vector. Thus we write:

$$A = \left\{ \frac{u_1}{\mu_A(u_1)}, \frac{u_2}{\mu_A(u_2)}, \dots, \frac{u_m}{\mu_A(u_m)} \right\} \quad (3)$$

$$\mu_A(u_i) = \mu_A(x_{i1}, x_{i2}, \dots, x_{in}) \quad i = 1 \dots m \quad (4)$$

Here $(x_{i1}, x_{i2}, \dots, x_{in})$ is the attribute vector of the object u_i in the training set. Similarly the conclusion B is treated as a crisp subset on the training set. An object belongs to this crisp set, if and only if its class is the same as B . Therefore the subset for B is represented as:

$$\mu_B(u_i) = \begin{cases} 1 & \text{if } \text{class}(u_i) = B \\ 0 & \text{otherwise} \end{cases} \quad i = 1 \dots m \quad (5)$$

The rule “If A Then B ” corresponds to an implication of $A \Rightarrow B$, which is equivalent to the proposition that A is B ’s subset, i.e. $A \subseteq B$. In this view, the measure of subsethood of A in B is utilized as the truth value of the rule. So we obtain

$$truth(A \Rightarrow B) = \frac{M(A \cap B)}{M(A)} = \frac{\sum_{u_i \in U_T} (\mu_A(u_i) \wedge \mu_B(u_i))}{\sum_{u_i \in U_T} \mu_A(u_i)} = \frac{\sum_{class(u_i)=B} \mu_A(u_i)}{\sum_{u_i \in U_T} \mu_A(u_i)} \quad (6)$$

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

Given a condition A , we choose the consequent class B from the finite candidates such that the truth value of the considered rule reaches its maximum. This means that the rule consequent can be selected with the following two steps:

Step 1: Calculate the competition strength for each class as

$$\alpha(c) = \sum_{class(u_i)=c} \mu_A(u_i) \quad c = C_1, C_2, \dots, C_K \quad (7)$$

Step 2: Determine the consequent class B as c^* that has the maximal competition strength, i.e.

$$\alpha(c^*) = \max(\alpha(C_1), \alpha(C_2), \dots, \alpha(C_K)) \quad (8)$$

3. Genetic Learning of Rule Premises

Genetic algorithms [4] are global algorithms based on mechanics of natural genetics and selection. They are theoretically and empirically proven to be an effective and robust means to find optimal or desirable solutions in complex spaces. A GA starts with a population of randomly or heuristically generated solutions (chromosomes). New offspring are created by applying genetic operators on selected parents. The selection of parents is performed based on the principle of ‘‘survive of the fittest’’. In this manner, a gradual improvement of the quality of individuals in the population can be achieved. Suitable coding scheme, genetic operators and fitness function must be defined for GA to search for optimal premise structure and membership functions of input fuzzy sets simultaneously.

The information concerning structure of rule premises can be considered as a set of discrete parameters, while the information about input fuzzy sets is described by a set of continuous parameters. Owing to the different nature between the information about rule structure and about fuzzy set membership functions, a hybrid string consisting of two substrings is proposed here. The first substring codes premise structure of a knowledge base and the second substring corresponds to parameters of fuzzy sets used by rules.

For coding parameters of membership functions the classical ‘‘concatenated binary mapping’’ method is certainly feasible. But for the sake of a shorter string length, an integer-substring is used in this paper. Let a parameter for fuzzy sets be quantised in the integer interval $\{0, 1, \dots, K_{\max}\}$ and denote $V_{\min}$, $V_{\max}$ as user-determined minimal and maximal parameter values, then the relationship between the corresponding integer K in the substring and the parameter value V is as follows:

$$V = V_{\min} + \frac{K}{K_{\max}}(V_{\max} - V_{\min}) \quad (9)$$

From (1), we can see that premise structure of such rules is characterized by integer sets $D(i)$ ($i=1, 2, \dots, s$). This fact suggests that a binary code be a suitable scheme for encoding structure of premises, since an integer from $\{1, 2, \dots, q[p(i)]\}$ 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 must be introduced to depict the set $D(p^{-1}(j)) \subseteq \{1, 2, \dots, q[j]\}$, 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 $[x_1=(small \text{ or } large)]$ and $[x_3=middle]$ and $[x_4=(middle \text{ or } large)]$ " can be coded by the binary group (101; 111; 010; 011).

Further, the whole substring for the premise structure of a rule base is a merging 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 one, meaning that all attributes are not considered in the premise; 2) All the bits for an attribute is zero. This attribute therefore takes no linguistic term in the premise resulting in an empty fuzzy set for the condition part of that attribute. Through elimination of invalid rule premises, the actual rule number can be reduced from the upper limit given by man. This also provides a possibility to adjust the size of the rule base by GA.

By the operation of crossover, parent strings mix and exchange their characters through a random process with expected improvement of fitness in the next generation. 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.

Mutation is a random alteration of a bit in a string so as to increase the variability of population. 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 small probability.

An individual in the population is evaluated according to the performance of the fuzzy classifier coded by it. Therefore the following four steps must be done to get an appropriate numerical fitness value of a string: 1) Decode the binary substring into premise structure of rules and integer-substring into membership functions of input

fuzzy sets; 2) Determine consequent classes under rule conditions by maximizing truth values of the rules; 3) Classify the training patterns with the rules generated from the above two steps; 4) Compute the rate of correctly classified training patterns as fitness value.

4. Simulation Results

We applied the proposed approach to build classification rules according to the Iris Data [3]. The task is to classify three species of iris (setosa, versicolor and virginica) by four-dimensional attribute vectors consisting of sepal length (x_1), sepal width(x_2), petal length (x_3) and petal width(x_4). There are 50 samples of each class in this data set. By randomly taking 10 patterns from each class as test data, the total data set was divided into training set (80%) and test set (20%). We built the fuzzy classifier based on the training data and then verified its performance according to the test data that were not used for learning.

Fig. 1. The membership functions of the attributes

Each attribute of the classifier was assigned with three linguistic terms: *short*, *middle* and *long*. By normalizing attribute values of each attribute into the interval between zero and one, the membership functions of input fuzzy sets are depicted in Figure 1. 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 desirable classification accuracy. GA was put into work to search for the premise structure of possible rules and to optimize the parameters (corresponding to the circle in Fig. 1) of the input fuzzy sets at the same time. As a result of the search process, 10 rule premises were identified as invalid, so that the rule base in fact contains only six rules in it. Clearly, the size of the rule base in this example was not prescribed exactly by man. Instead it was automatically adjusted by the learning algorithm within an upper limit.

The fuzzy classifier learned from the GA has very good performances despite its relatively small number of rules. On the training set it correctly classifies 119 patterns (i.e. 99.2% of the 120 patterns). On the test data 30 patterns (i.e. 100% of the 30 patterns) are properly classified.

The results of some other machine learning algorithms on the Iris Flower Problem are given in Table 1 for comparison. It is evident that our method outperforms the other learning algorithms on this well known benchmark problem.

Table 1. Accuracy of the other five learning algorithms on the Iris Data

<i>Algorithms</i>	<i>Setosa</i>	<i>Viginica</i>	<i>Versicolor</i>	<i>Average</i>
Hirsh [5]	100%	93.33%	94.00%	95.78%
Aha [1]	100%	93.50%	91.13%	94.87%
Dasarathy [2]	100%	98%	86%	94.67%
C4 [10]	100%	91.07%	90.61%	93.89%
Hong [6]	100%	94%	94%	96.00%

5. Conclusions

The method proposed in this paper provides an effective means for automatically acquiring classification knowledge from sample examples. GA is used to derive appropriate premise structure of classification rules as well as to optimize membership functions of fuzzy sets at the same time. The size of the rule base is adapted by the search algorithm within an upper limit of the rule amount assumed by man in advance. The resulted knowledge base is compact in the sense that it usually contains a much smaller number of rules compared with that enumerating all canonical AND connections of linguistic terms as input situations. A small number of rules makes it simple for human users to check and interpret the contents of the knowledge base.

References

1. Aha, D.W. and Kibler, D.: Noise-tolerant instance-based learning algorithms. In Proc. 11th Internat. Joint Conf. On Artificial Intelligence, Detroit, MI (1989), 794-799
2. Dasarathy, B.V.: Noising around the neighbourhood: a new system structure and classification rule for recognition in partially exposed environments. IEEE Trans. Pattern Analysis and Machine Intelligence 2 (1980) 67-71
3. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Ann. Eugen 7 (1936) 179-188
4. Goldberg, D.E.: Genetic algorithms in search, optimization and machine learning. Addison-Wesley, New York (1989).
5. Hirsh, H.: Generalizing version spaces. Machine Learning 17 (1994) 5-46
6. Hong, T.P. and Tseng, S.S.: A generalized version space learning algorithm for noisy and uncertain data. IEEE Trans. Knowledge Data Eng. 9 (1997) 336-340
7. Ishibuchi, H. et al.: Distributed representation of fuzzy rules and its application to pattern classification. Fuzzy Sets and Systems 52 (1992) 21-32
8. Meier, W., Weber, R. and Zimmermann, H.-J.: Fuzzy data analysis – methods and industrial application. Fuzzy Sets and Systems 61 (1994) 19-28
9. Nozaki, K. et al.: Adaptive fuzzy rule-based classification systems. IEEE Trans. Fuzzy Systems 4 (1996) 238-250
10. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, (1993).