Abstract—The algorithm of on-line predictor from input-output data pairs will be proposed. In this paper, it proposed an approach to generate fuzzy rules of predictor from real-time input-output data by means of ARMA model concept for unknown system. It includes two phase: (1), Generating fuzzy rules phase, (2), On-Line Learning phase; If the error between the real output and the predictor's output is larger than the desired error, it means that the lack of the fuzzy rules. Thus, it will generate some new fuzzy rules for the fuzzy predictor or adding an output term in the premise part of fuzzy rules. From the generating fuzzy rules phase, it can on-line generate the fuzzy predictor. In another word, some redundant rules may be generated from bad information after learning. They may be incoming data include outliers, noises or uncertainties. Such bad rules will be discarded by a usage degree constant. To achieve good performance for this fuzzy predictor, the parameters of each fuzzy rule may be adjusted by on-line learning, when the prediction error into a pre-defined bound. In the simulation example, a nonlinear time-varying process operating in open loop is illustrated. Simulations and real-time results show the advantages of the proposed method.

I. INTRODUCTION

Model-based predictive controller (MBPC) has received increased attention in both the control engineering and industry communities [1-2], and much effort has been made to extend MBPC from linear system to nonlinear systems [8]. The fuzzy control has been proved to be a good choice of controlling complex dynamic systems [9-10]. Fuzzy models are able to provide the required nonlinear mapping and also provide, under suitable constraints, a linguistic interpretation for the relevant dynamic behavior of the plant [11-13]. The predictive fuzzy control has also been successes in an ATO application [14]. Recently, nonlinear plant descriptions have been introduced in this class of fuzzy predictive control strategy. For instance, nonlinear auto-regression with eXogenous inputs models [15], artificial neural networks [16], and fuzzy models [17-18] have been exploited. A comparison of those fuzzy model-based adaptive-predictive control strategies have been provided in [19]. In [35] proposed a fuzzy predictive filter for obtaining at each time instance a suitable set of control actions. The actions are based on the available control space and on simple fuzzy criteria regarding the current and predicted error. Also, in [20] that proposed a hierarchical intelligent controller for an unknown low-level system. The high-level controller consists of two parts; one is a fuzzy predictor and the other is a fuzzy controller. The fuzzy predictor is used to predict the system output and then modify the control signal so that the undesirable effects of system time delay can be eliminated.

Traditional, the methods for learning fuzzy models from data are based on the idea of consecutive structure and parameter identification. Traditional learning algorithms [21-28] for this kind of approaches are to tune system parameters according to modeling error, and without considering the training data with outlier. They may try to fit those improper data in the training process and thus, the learned systems are corrupted. In other words, if there exist outliers and the training process lasts long enough, the obtained systems may have overfitting phenomena. In [29-31] also proposed the algorithms for TSK fuzzy modeling with outlier. However, most above approaches their cost function is only based on modeling error through learning algorithm. The approach termed as RPSFR (Robust Proper Structure Fuzzy Regression Algorithm) is proposed in [3-5] which is iterative between these both agglomeration algorithms. Hence, it can not only simultaneously identify fuzzy subspaces and the functions in consequent parts without knowing the cluster number but also can have robust learning effects against outliers and with a proper clustering structure. These methods however, suppose that all the data is available at the start of the process of training. Therefore, they are appropriate for off-line applications only. Their use in on-line algorithms is only possible the price of re-training the parameters with iterative and time-consuming procedure. For on-line learning of the fuzzy model, it develops the algorithm of on-line clustering and recursive parameter estimation [32-33].

Another way is to generate fuzzy rules from numerical data pairs, collect these fuzzy rules into a common rule base [34]. It is a one-pass build-up procedure that does not require time-consuming training. The disadvantage for this approach is the fuzzy rules may be increase growing to extra numbers, especially; the data pairs contain noise or outlier. In [20] modify the approach in [34] to generate fuzzy rule of predictor from numerical data. Those approaches need some priori-knowledge of the model, not suitable for unknown system. In this paper, we also modify this approach to generate fuzzy rules of predictor from real-time input-output data by means of ARMA model concept. It includes two phase: (1), Generating fuzzy rules phase, (2), On-Line Learning phase; If the error between the real output and the predictor's output is larger than the desired error, it means that the lack of the fuzzy rules. Thus, it will generate some new fuzzy rules for the fuzzy predictor. From the generating fuzzy rules phase, it can on-line generate the fuzzy predictor. To achieve good performance for this fuzzy predictor, the
parameters of each fuzzy rule may be adjusted by on-line learning, when the prediction error into a pre-defined bound.

This paper is organized as follows: Section II describes the constructing on-line fuzzy predictor procedures. There are two phases in the algorithm. They will be introduced separately. The Generating fuzzy rules phase and On-Line Learning phase. The simulation results of a nonlinear time-varying process example are in Section III. Finally, Section IV concludes the paper.

II. ON-LINE FUZZY PREDICTOR FROM REAL-TIME DATA

Consider a single-input/single-output model-free plant whose dynamics can be represented by a discrete time model of the form

\[ x[k] = Y(x[k-1], x[k-2], \ldots, x[k-p], u[k], u[k-1], \ldots, u[k-q]) \]

where \( y[k] \) and \( u[k] \) represent the value of discrete time sequences of the input and output, respectively indexed by the discrete time \( k \). \( p \) and \( q \) are positive integer constants referring to the order of the plant such that \( 0 < q \leq p \). A well-known formula is the ARMA model, it can represent as a linear difference equation as follow

\[ x[k] = \sum_{n=1}^{p} a[n] x[k-n] + \sum_{n=0}^{q} b[n] u[k-n] \tag{1} \]

where \( a[k] \) and \( b[k] \) is the coefficients of AR(p) and MA(q), respectively. Thus the sets of the input/output pairs for each sample are

\[ (X[k], U[k], x[k]), (X[k+1], U[k+1], x[k+1]), \ldots \tag{2} \]

where \( X[k] = [x[k-1], \ldots, x[k-p]]_{p+1}, U[k] = [u[k], \ldots, u[k-q]]_{(q+1)x1} \).

We can use the sequences of these data sets to generate or learn the fuzzy rules of the predictor. The \( i \)-th rule of a fuzzy predictor for Eq.(1) is described as

\[
R^i: \text{IF} \ x[k-1] \text{ is } \mu^i_{x1}(\overline{\theta}^i_{x1}) \text{ and } \ldots \text{ and } x[k-p] \text{ is } \mu^i_{xp}(\overline{\theta}^i_{xp}) \text{ and } u[k] \text{ is } \mu^i_{u1}(\overline{\theta}^i_{u1}) \text{ and } \ldots \text{ and } u[k-q] \text{ is } \mu^i_{uq}(\overline{\theta}^i_{uq}) \text{ THEN } x[k] = y^i \tag{3}
\]

for \( j = 1, 2, \ldots, C, C \) is the number of rules, where \( \mu^i_{j}(\overline{\theta}^i_{j}) \) is the fuzzy set of the \( j \)-th rule with the parameter set \( \overline{\theta}^i_{j} \). The consequent part is a fuzzy singleton represented by real output value. Thus

\[ y^i = x^i_m \tag{4} \]

where \( x^i_m \) is the real output value at \( i \)-th time generated rule.

The algorithm of on-line predictor includes two phase:

(1). Generating fuzzy rules phase:

In [20] modify the approach in [34] to generate fuzzy rule of predictor from numerical data. Those approaches need some priori-knowledge of the model, not suitable for unknown system. We also modify this approach to generate fuzzy rules of predictor from on-line data pairs in Eq.(2). It describes as follow five steps.

Step 1. Define the domain intervals of input and output.

Define the domain intervals of \( x \) and \( u \) are \([x^-, x^+], [u^-, u^+]\), respectively, where “+” “-” denote the lower bound and upper bound of the variables. Initialize, we can assume a narrow region of those bounds. If the incoming data exceed those bound, then refine those bound as the maximal value of the data.

Step 2. Divide the input and output spaces into fuzzy region.

Divide each domain interval into \( N \) regions, and assign each region a fuzzy membership function. The Gaussian membership functions are used in the premise parts (i.e., \( \exp \left( \frac{(x_{j} - \theta_{j})}{2\sigma_{j}} \right) \)). Similarly, if the incoming data exceed those bound, then refine those fuzzy partitions based on the maximal value of the data.

Step 3. Select the \( p \) and \( q \) for output and input.

The \( (p+q) \) is the number of premise part for fuzzy rules. It is reasonable that the more \( (p+q) \) the more accuracy of the predictor. But the generating rules number will be huge, when the \( (p+q) \) is large. There is a problem will arise, too much more firing rule in the inference process may be overfitting. Hence, we can select a suitable \( (p+q) \) at the initiate. After a long term learning, if the error is still exceed the prescribe bound, then adding an output term in the premise part. The pseudo code is

If (long term learning) and (predict Error > pre-defined Error) Then (Add a output term in premise part)

Step 4. Generate fuzzy rules from on-line data pairs.

For an incoming data pair, the predicted output of the fuzzy predictor is inferred as

\[ \hat{x}[k] = \frac{\sum_{j=1}^{C} \mu^i \cdot y^i}{\sum_{j=1}^{C} \mu^i} \tag{5} \]
If the error between the actual output and predict output is
greater than the desired error, it means that the lack of the
fuzzy rules. Thus, it will generate some new fuzzy rules for
the fuzzy predictor. First, determine the degree of each
variable in difference region. Second, assign a Confidence
Degree (CD) for each possible new rule. The CD of each rule
is defined as

\[ CD^j = \prod_{i=1}^{q} m_i^j \]  \hspace{1cm} (6)

where \( m_i^j \) denote the membership degree of each variable in
premise part for the \( j \) rule. It represents our belief of its
usefulness. A new rule generates only its CD bigger than a
prescribed threshold degree (confidence constant). Thus,

\[ IF \ CD^j > \alpha \ THEN \ " \ Generate \ this \ new \ rule \" \]

where \( \alpha \) is a user defined confidence constant. It is probable
that there will be some conflicting rules, i.e., rules that have
the same IF part but a different THEN part. To resolve this
conflict is to combine those rules by CD for the consequent
part. For fuzzy singleton type in Equ.(4)

\[ y^j = \frac{\sum CD_i^j \cdot y_i^j}{\sum CD_i^j} \]  \hspace{1cm} (7)

\( i = 1, \cdots, n \) is conflicting rules.

\textbf{Step 5). Reduce the redundant fuzzy rules.}

It is probable that there will be some redundant rules
may be generated from bad information after long term
learning. They may be incoming data include outliers, noises
or uncertainties. Thus, it will generate some unnecessary
fuzzy rules, we call it is outlier rule. For those rules, the
probability of firing is very low for each fuzzy inference
process. Thus, we assign a Usage Degree (UD) for each
generated rule. For the reason, the more recent rule more
important. The UD of each rule is defined as

\[ UD^j = \frac{k^j + f_s}{k} \]  \hspace{1cm} (8)

where \( k \) is current time, \( k^j \) is the generated time for this rule
and \( f_s \) is firing counter. Thus,

\[ IF \ UD^j < \eta \ THEN \ " \ Discard \ this \ rule \"

where \( \eta \) is an user-defined usage constant.

(2). \textbf{On-Line Learning phase:}

From above five steps procedure, it can on-line generate
the fuzzy predictor. To achieve good performance for this
fuzzy predictor, the parameters of each fuzzy rule may be
adjusted by on-line learning, when the prediction error into a
pre-defined bound.

\[ IF \ error > \xi \ THEN \ " \ Generate \ Fuzzy \ Rule" \]

\[ ELSE \ " \ On-Line \ Learning" \]

\( \xi \) is a pre-defined error bound. Thus the parameters of
premise parts are updated as

\[ \Delta \theta_i = \zeta^{a} (x[k] - \hat{x}[k]) \sum_{j=1}^{p} \frac{\partial m_i^j}{\partial \theta_i^j}, \]  \hspace{1cm} (9)

where \( \zeta^{a} \) is the adaptive learning constant for premise parts.

In summarizing the above discussion, the algorithm of
on-line fuzzy predictor is described as follow:

[Step.1] Initialize. Define the domain intervals of input
and output. Divide their domain into fuzzy
region and selecting the \( p \) and \( q \).

[Step.2] Generate the rules of fuzzy predictor from
on-line data pairs sequence.

[Step.3] Discard the redundant fuzzy rules by UD.

[Step.4] Check the error, if bigger than an error bound
then goto Step.2

[Step.5] Apply the on-line adaptive learning law.

Based on the above procedures, that can build the fuzzy
predictor from real-time input/output data. Since the
Gaussian membership functions are used in the premise parts,
its characteristic is that all rules are firing, when there are too
many rules in fuzzy predictor, seem inefficient. So in the
actual inference, we can utilize Confidence Degree (CD) as
threshold value, when it greater than the preserving value,
this rule is firing. It can reduce the complexity of the
inference.

III. \textbf{Simulation Example}

In this section, a nonlinear time-varying process
operating in open loop [17] is illustrated. The process is

\[ x(k+1) = 0.1x(k)e^{-0.25x^2(k)} + 0.5u(k) + a(k)x(k) \]  \hspace{1cm} (10)

where \( a(k) = 0.5\sin(k/100), u(k) = 5\sin(k/20) \). The fuzzy
rule is singleton type such as Equ.(4). The actual process
output and the on-line one-step ahead predict output is shown
in Fig.1. We can find that the outlier effect will be
reduced after long term learning. The generated rules and
firing rules number for each sample are shown in Fig.2. Also,
the generated rule's number is reduced within a small variation range.

IV. CONCLUSIONS

MPC needs a model of the process under control to predict the future control actions. In this paper, we proposed an algorithm for on-line fuzzy predictor from real-time data. It is a one-pass build-up procedure that does not require time-consuming training. The outlier effect will be reduced after long-term learning by the discard the redundant fuzzy rule's strategy. Also, the generated rule's number will be reduced within a small variation range. It should be noted that the consequent part is a fuzzy singleton for the generated fuzzy rules. We believe that the consequent instead of TSK type that will reduce the generated rule's number.

ACKNOWLEDGMENT

This work was supported in part by National Science Council of Taiwan under Grant NSC 95-2221-E-244-004.

REFERENCES

Fig. 1. The actual and predict output for the nonlinear process.

Fig. 2. The generated rules and firing rules number for each sample.