TIME SERIES FORECASTING IN SOFTWARE RELIABILITY BY FUZZY NEAREST NEIGHBOUR METHOD

BURTSCHY, Bernard
ENST, INFRES, Paris, France
burtschy@enst.fr

ALBEANU, Grigore
Spiru Haret University, Bucharest, Romania
g.albeanu.mi@spiruharet.ro

POPENȚIU-VLĂDICESCU, Florin
University of Oradea, UNESCO IT Chair
popentiu@imm.dtu.dk

Abstract

Time series approach is used to predict future values based on the history of past values. This paper considers the usage of a fuzzy nearest neighbour method for time series forecasting. Not only sequential patterns in time series data are considered, but also the subsequences with some optimal size.

Keywords: *fuzzy logic, time-series, software reliability*

AMS classification: 91B99, 62P30

1. Introduction

Fuzzy logic has proven to be capable of modelling highly nonlinear and multidimensional processes. Nonlinear model identification is used in a probabilistic framework for data analysis during the software life cycle. However the uncertainty and imprecision can be captured in a fuzzy manner. The structure identification of the fuzzy model requires the following elements: the model type; the number of rules (the base of rules should satisfy all quality requirements consistencies, completion, and continuity), the positioning of the fuzzy sets in the domain of each variable, adequate conversion methods from crisp to fuzzy, and fuzzy to crisp, and an inspired inference operator.

The model structure can be described by the number of arrangements of membership functions associated to the fuzzy model. The simplest possible model structure uses one membership function per input variable. The correlation coefficient is computed for the testing data set, and then the model complexity is incrementally increased by adding membership functions. The model structure having the highest correlation coefficient will be selected as an optimal one. In this manner, an association between model parameters and the conclusions of the rules is established. The parameter estimation is based on the least square approach, and

from this point of view, fuzzy model identification is close to classical non-linear model identification.

The most difficult part consists of structure identification. For this task, different approaches were already proposed. When appriori knowledge is available the fuzzy partition can be established by nonlinear optimization, otherwise a preliminary data clustering is necessary as stated in [3]. Such considerations are useful when software architecture for reliability monitoring will be implemented in a soft computing framework. Fuzzy modelling, as described above, can be used in order to extend the EV approach introduced in [7]. For the software engineering field, another usage of fuzzy logic deals with software fault diagnosis consisting of the following tasks: fault detection and isolation, and fault analysis. For an efficient fault detection and isolation the software engineer has to perform the following tasks: the generation of messages which reflect the faults, and the decision making concerning the type and time of occurrence of the fault. As mentioned in [6], three categories of methods for fault detection and isolation can be used. These are based on: messages, analytical models, and knowledge. The messaging approach is not a portable one. A change in operation environment may cause false alarm, or masking of faults. Developing analytical models is not an easy task. Only an already known behavior, that is a deterministic process, can be monitored. The knowledge-based approach is a fuzzy logic one. This strategy offers the opportunity to combine heuristic knowledge with any model knowledge which may be available. However, implementing a fuzzy-inference system for fault detection and isolation is a great provocation. The rule base will be determined from inspection of the data according to all users' preferences, or software requirements as defined in the first phase of the software life cycle, based on developments presented in [1].

The detection of faults can also be realized by testing and debugging. According to [6], current SRGMs that incorporating the debugging process assume that when a failure occurs a debugging effort takes place which may or may not remove the fault with some unknown but estimable probabilities. However, there are a large variety of bugs in software and the debugging activity should be treated as a fuzzy process rather than crisp. This is why a fault is debugged according to a set of membership functions labeling the debugging performance as: total imperfect debugging, more or less imperfect debugging, less than perfect debugging, …, total perfect debugging.

In this context some software reliability measures can be proposed: the total fuzzy number of faults; the total fuzzy number of faults that remains in the software, and the expected proportion of faults removed from the software.

The next section will describe the fuzzy principles used for time series predictions. Firstly, a look-up table scheme is defined, the fuzzy nearest neighbour method, developed in [9], is modified for software engineering tasks. Some practical considerations, from experience, will be presented in the third section.

2. **Fuzzy principles**

It is well known that the main approaches in time series forecasting are based on: statistical models fitting [8], neural networks [6, 9], and fuzzy modelling [9].

One approach in time series prediction uses all available input-output data pairs in order to build a rule base [10]. Let us consider a fuzzy system with inputs

(x_1, x_2, ..., x_m) and an output y, and there are n data points in the training set. For systems having more inputs, the procedure will work in a similar manner. In order to create fuzzy rules with fixed membership functions, five steps are necessary to be followed [10]:

[**The first step**] – Define the fuzzy partitions for the input and output variables.

[**The second step**] – Generate one fuzzy rule for each input-output pair, and obtain an initial fuzzy rule-base.

[**The third step**] – Calculate the membership degree (D) for each fuzzy rule belonging to the fuzzy rule-base created in the previous step.

[**The fourth step**] – Removing inconsistent and redundant rules [10] and, create the final fuzzy rule-base. A reliability factor (RF) [4] can be computed for every set of k rules with the same antecedent part by the ratio k_1/k, where k_1 is the number of redundant rules. In this manner, an effective degree (ED) can be computed for each rule degree as ED = D*RF. In the final rule-base will contain fuzzy rules with the largest effective degrees.

[**The fifth step**] – Select the inference scheme and perform the fuzzy inference procedure [10]. Select a defuzzification scheme to provide a crisp value as output. A variety of defuzzification schemes can be found in [10].

The simplest fuzzy rule-base has rules of the form **IF** X(t) **IS** A THEN X(t+1) **IS** B, where A and B are fuzzy sets. However, the above approach can include more information in the antecedent part and complex rules can be created.

Another approach consists of exploring a nearest neighbour pattern for time-series forecasting in a fuzzy manner, called the fuzzy nearest neighbour method (FNNM). Let us consider the time series as a vector $y = (y_1, y_2, ..., y_n)$, where n is the total number of points in the series. An encoding procedure can be applied according to the rule $y_i ::= IF(y_{i+1} < y_i, 0, 1)$.

The main objective when dealing with time-series is to forecast y_{n+1} if the time-series vector y is given. The FNNM considers a total of k neighbours ($x_{t[1]}$, $x_{t[2]}$, ..., $x_{t[k]}$), where $x_{t[k]}$ is the value at time $t[k]$. According to [9], the forecast y_{n+1} is based on either the averaging or extrapolation of values $x_{t[1]+1}$, $x_{t[2]+1}$, ..., $x_{t[k]+1}$. In the following, the membership function can be seen as a proximity function. The function value of 1 denotes the nearest of all neighbours, instead poor neighbours obtained when the function values are near zero. An optimal membership threshold λ an be used to select the neighbours allowed to contribute in the analysis. The steps of the FNNM are the following:

[*The first step*] – Compute the proximity of y_n and past values: $\varphi(y_i) = (1+d(y_i, y_n))-1$, i = 1, 2, ..., n-1, n; $\varphi(y_i)$ shows the fuzzy proximity of y_i and y_n, and d is the Euclidian distance between y_i and y_n.

[*The second step*] – Scale the membership (proximity) values within the [0,1] interval:

$$\varphi'(y_i) = (\varphi(y_i) - a) / (b - a),$$

where

$$a = \min \{\varphi(y_i), i = 1, 2, ..., n-1\}, b = \max \{\varphi(y_i), i = 1, 2, ..., n-1\}.$$

[*The third step*] – Select the nearest neighbours: $\varphi'(y_i) \geq \lambda$. A total of k selected neighbours are denoted as xt[1], xt[2], ..., xt[k].

[*The fourth step*] – Forecast y_{n+1} by averaging the k nearest neighbours.

[*The fifth step*] – Optimize the method (select an optimal λ) for minimal error in prediction, considering the actual value y_{n+1}.

The data analyst has some possibilities to modify this skeleton. He or she can change the distance function and the procedure to compute the membership function. Also the selection of the neighbours can be easy modified.

3. Practical considerations and conclusions

As mentioned in the introductory section, an analysis concerning the TJMES software [5] was done both in a statistical manner, according to the methodology provided by [2, 7], and using fuzzy techniques [6].

It is easy to see that the number of rules increases when a large number of data it is considered. This is why it is not recommended to use the look-up table approach for long series. However, the FNNM approach can be used for large time-series. For the experiment we select $\lambda = 0.5$ and a variable k. We found that the results are interesting enough.

Finally, let us remark, that the above mentioned approaches are general enough for different types of time series and can be used in different fields of activity, not only in software reliability engineering.

References

1. Albeanu, G., Popenţiu Fl., *Total Quality for Software Engineering Management*, In H. Pham (ed.), Handbook of Reliability Engineering, 567-584, London: Springer, 2003.
2. Burtschy, B., Albeanu, G., Boroş, D.N., Popenţiu, Fl., *Improving Software Reliability Forecasting*, Microelectronics and Reliability, 37(6), 901-907, 1997.
3. Evsukoff, A., Branco A.C.S, Galichet, S., *Structure Identification and Parameter Optimization for Non-linear Fuzzy Modeling*, Fuzzy Sets and Systems, 132, 173-188, 2002.
4. Liu X., Kwan B. K., S. Y. Foo, *Time Series Prediction Based on Fuzzy Principles*, Preprint, Department of Electrical & Computer Engineering, Florida State University, 2003.
5. TJMES, *Time Series Analyzer Based on a Java API*, 2008.
6. Madsen, H., Thyregod, P., Burtschy, B., Fl. Popenţiu, Albeanu, G., *On Using Soft Computing Techniques in Software Reliability Engineering*, Proceedings of the ESREL 2005 Conference, 27-30 June 2005, Tri City Poland, Taylor & Francis Group plc, 2005.
7. Popenţiu-Vlădicescu, F., Burtschy, B., Albeanu, G., *Time Series Methods for Modeling Software Quality*. Proceedings of ESREL 2001, Italy, 1: 9-15, 2001.
8. Popenţiu-Vlădicescu, F., Albeanu, G., Sens P., Thyregod, P,. *Software Architecture for Distributed Systems (SADS): NN and EV Approaches*. Proceedings of ESREL 2001, Italy, 1031-1037, 2001.
9. Singh, S, *Fuzzy Nearest Neighbour Method for Time-Series Forecasting*, Proc. 6th European Congress on Intelligent Techniques and Soft Computing, Germany, vol. 3: 1901-1905, 1998.
10. Văduva I., Albeanu, G., *Introducere în modelarea fuzzy*, Editura Universităţii Bucureşti, 2003.