A
References

[1] S. Abe and R. Thawonmas. A fuzzy classifier with ellipsoidal regions. *IEEE Trans. on fuzzy systems*, vol. 5, num. 3, pp. 358-368, 1997

[2] C.B. Allendoerfer and C.O. Oakley. *Principles of mathematics*, McGraw-Hill, 1955

[3] R. Babuska, H. Bersini, D.A. Linkens, D. Nauck, G. Tselentis and O. Wolkenhauer. Future prospects for fuzzy systems and technology. *ERUDIT newsletter*, vol. 6, num. 1, 2000

[4] J.C. Bezdek. *Patter recognition with fuzzy objective function algorithms*, Plenum press, 1981

[5] U. Bondenhofer and P. Bauer. A formal model of interpretability of linguistic variables. *Interpretability issues in fuzzy modeling*, Studies in fuzziness and soft computing, vol. 128, Springer, pp. 524-545, 2003

[6] B. Bouchon-Meunier. Linguistic hedges and fuzzy logic. *Proceedings of the 1st IEEE international conference on fuzzy systems*, pp. 247-254, 1992

[7] G.E.P. Box and G.M. Jenkins. *Time series analysis, forecasting and control*, Holden Day, 1970

[8] M. Brown and C. Harris. *Neurofuzzy adaptive modeling and control*, Prentice-Hall, 1994

[9] J. Casillas, O. Cordón, M.J. del Jesús and F. Herrera. Genetic tuning of fuzzy rule-based systems integrating linguistic hedges. *Proceedings of the 9th IFSA world congress*, pp. 1570-1574, 2001

[10] J. Casillas, O. Cordón, M.J. del Jesús and F. Herrera. Genetic tuning of fuzzy rule deeps structures preserving intepretability and its interaction with fuzzy rule set reduction. *IEEE Trans. on fuzzy systems*, vol. 13, num. 1, pp. 13-29, 2005

[11] J. Casillas, O. Cordón and F. Herrera. COR methodology: a simple way to obtain linguistic fuzzy models with good interpretability and accuracy. *Accuracy improvements in linguistic fuzzy modeling*, Studies in fuzziness and soft computing, vol. 129, Springer, pp. 27-45, 2003

[12] J. Casillas, O. Cordón, F. Herrera and L. Magdalena. Interpretability issues in fuzzy modeling. *Studies in fuzziness and soft computing*, vol. 128, Springer, 2003

[13] J. Casillas, O. Cordón, F. Herrera and L. Magdalena. Accuracy improvements to find the balance interpretability-accuracy in linguistic fuzzy modeling: an overview. *Studies in fuzziness and soft computing*, vol. 129, Springer, pp. 3-24, 2003

[14] O. Cordón and F. Herrera. A proposal for improving the accuracy of linguistic modeling. *IEEE Trans. on fuzzy systems*, vol. 8, num. 3, pp. 335-344, 2000

[15] O. Cordón, F. Herrera and I. Zwir. Linguistic modeling by hierarchical systems of linguistic rules. *IEEE Trans. on fuzzy systems*, vol. 10, num. 1, pp. 2-20, 2002

[16] K. Chellapilla. Evolving nonlinear controllers for backing up a truck-and-trailer using evolutionary-programming. *Proceedings of the evolutionary programming 7th international conference*, pp. 417-426, 1998

[17] C.Y. Chen and B.D. Liu. Linguistic hedges and fuzzy rule based systems. *Accuracy improvements in linguistic fuzzy modeling*, Studies in fuzziness and soft computing, vol. 129, pp. 165-192, 2003

[18] S.L. Chiu. A cluster estimation method with extension to fuzzy model identification. *Proceedings of the third IEEE conference on fuzzy systems*, pp. 1240-1245, 1994

[19] P. Delicado. Another look at principal curves and surfaces. *Journals of multivariable analysis*, num. 77, pp. 84-116, 2001

[20] M. Delgado, F. Gómez-Skarmeta and F. Martín. A fuzzy clustering-based rapid prototyping for fuzzy rule-based modeling. *IEEE Trans. on fuzzy systems*, vol. 5, num. 2, pp. 223-232, 1997

[21] D. Driankow, H. Hellendoorn and M. Reinfrank. *An introduction to fuzzy control - Second edition*, Springer-Verlag, 1996

[22] J. Dickerson and B. Kosko. Fuzzy function approximation with ellipsoidal rules. *IEEE Trans. on system, man and cybernetics*, vol. 26, num. 4, pp. 542-560, 1996

[23] N. R. Draper and H. Smith. *Applied regression analysis*, Wiley-interscience, 1998

[24] J.C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of cybernetics*, vol. 3, pp. 32-57, 1973

[25] D. Dubois and H. Prade. *Fuzzy sets and systems: theory and applications*, Academic Press, 1980

[26] D. Dubois and H. Prade. Basic issues on fuzzy rules and their application to fuzzy control. *Proceedings of the IJCAI-91 workshop on fuzzy control*, pp. 5-17, 1991

[27] J. Espinosa and J. Vanderwalle. Constructing fuzzy models with linguistic integrity from numerical data - AFRELI algorithm. *IEEE Trans. on fuzzy systems*, vol. 8, num. 5, pp. 591-600, 2000

[28] J. Espinosa and J. Vanderwalle. Extracting linguistic fuzzy fuzzy models from numerical data - AFRELI algorithm. *Interpretability issues in fuzzy modeling*, Studies in fuzziness and soft computing, vol. 128, Springer, pp. 100-125, 2003

[29] G. Farin. *Curves and surfaces for computed aided geometric design. A practical guide*, Computer science and scientific computing, Academic Press Inc., 1990

[30] D.S. Feldman. Fuzzy logic synthesis with genetic algorithms. *Proceedings of the 5th international conference on genetic algorithms*, pp. 312-317, 1993

[31] M.J. Fuente, G.I. Sainz-Palmero and D. Pintado. Automatic fuzzy rule generation in a biotechnological process. *15th Triennial IFAC World Congress*, 2002

[32] C. Garriga-Berga. Min-square error optimization of fuzzy curves. *Proceedings of the international fuzzy systems association world congress IFSA'03*, Istanbul, 2003

[33] A.E. Gaweda and J.M. Zurada. Data-driven linguistic modeling using relational fuzzy rules. *IEEE Trans. on fuzzy systems*, vol. 11, num. 1, pp. 121-134, 2003

[34] A. González and R. Pérez. A fuzzy theory refinement algorithm. *International journal of approximate reasoning*, vol. 19, num. 3 and 4, pp. 193-220, 1998

[35] A. González and R. Pérez. A study about the inclusion of linguistic hedges in a fuzzy rule learning algorithm. *International journal of uncertainty, fuzziness and knowledge-based systems*, vol. 7, num. 3, pp. 257-266, 1999

[36] S. Guillaume. Designing fuzzy inference systems from data: an interpretability-oriented review. *IEEE Trans. on fuzzy systems*, vol. 9, num. 3, pp. 426-443, 2001

[37] S. Guillaume and B. Charnomordic. A new method for inducing a set of interpretable fuzzy partitions and fuzzy inference systems from data. *Interpretability issues in fuzzy modeling*, Studies in fuzziness and soft computing, vol. 128, Springer, pp. 148-175, 2003

[38] S. Guillaume and B. Charnomordic. Generating an interpretable family of fuzzy partitions from data. *IEEE Trans. on fuzzy systems*, vol. 12, num. 3, pp. 324-385, 2004

[39] J. Hauser, S. Sastry and P. Kokotović. Nonlinear control via approximate input-output linearization: the ball and beam example. *IEEE Trans. on automatic control*, vol. 37, num. 3, pp. 392-398, 1992

[40] C.M. Higgins and R.M. Goodman. Fuzzy rule-based networks for control. *IEEE Trans. on fuzzy systems*, vol. 2, num. 1, 1994

[41] F. Herrera, M. Lozano and J.L. Verdegay. Tuning fuzzy controllers by genetic algorithms. *International journal of approximate reasoning*, num. 12, pp. 299-315, 1995

[42] F. Herrera, M. Lozano and J.L. Verdegay. A learning process for fuzzy control rulmees using genetic algorithms. *Fuzzy sets and systems*, vol. 100, pp. 143-148, 1998

[43] H. Ishibuchi, T. Nakashima and T. Murata. Three-objective genetic-based machine learning for linguistic rule extraction. *Information sciences*, vol. 136, num. 1-4, pp. 109-133, 2001

[44] H. Ishibuchi, K. Nozaki, N. Yamamoto and H. Tanaka. Selecting fuzzy if-then rules for classification problems using genetic algorithms. *IEEE Trans. on fuzzy systems*, vol. 3, num. 3, pp. 260-270, 1995

[45] H. Ishibuchi, T. Murata and I.B. Turksen. Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. *Fuzzy sets and systems*, vol. 89, num. 2, pp. 135-150, 1997

[46] R. Jager. Fuzzy logic in control. *Ph.D. dissertation*, Technical University of Delft, 1995

[47] J.R. Jang. ANFIS: adaptive-network-based fuzzy inference system. *IEEE Trans. on systems, man and cybernetics*, vol. 23, num. 3, pp. 665-885, 1993

[48] J.R. Jang. Structure determination in fuzzy modeling: a fuzzy CART approach. *Proceedings of the third IEEE international conference on fuzzy systems*, vol. 1, pp. 480-485, 1994

[49] J.R. Jang. Input selection for ANFIS learning. *Proceedings of the fifth IEEE international conference on fuzzy systems*, vol. 2, pp. 1493-1499, 1996

[50] J.R. Jang and C.T. Sun. Neuro-fuzzy modeling and control. *Proceedings of the IEEE*, vol. 83, num. 3, pp. 378-406, 1995

[51] J.R. Jang and C. Sun. Neuro-fuzzy modeling and control. *Proceedings of the IEEE*, vol. 83, num. 3, pp. 378-406, 1995

[52] R.E. Jenkins and B. Yuhas. A simplified neural network solution through problem decomposition: the case of the truck backer-upper. *IEEE Trans. on neural networks*, vol. 4, num. 4, pp. 718-720, 1993

[53] Y. Jin. Fuzzy modeling of high dimensional systems: complexity reduction and interpretability improvement. *IEEE Trans. on fuzzy systems*, vol. 8, num. 2, pp. 212-221, 2000

[54] Y. Jin. Generating distinguishable, complete, consistent and compact fuzzy systems using evolutionary algorithms. *Accuracy improvements in linguistic fuzzy modeling*, Studies in fuzziness and soft computing, vol. 129, Springer, pp. 100-118, 2003

[55] Y. Jin, W. von Seelen and B. Sendhoff. On generating FC^3 fuzzy rule systems from data using evolution strategies. *IEEE Trans. on systems, man and cybernetics - Part B: cybernetics*, vol. 29, num. 4, pp. 829-845, 1999

[56] B. Kégl, A. Krzyzak, T. Linder and K. Zager. Learning and design of principal curves. *IEEE Trans. on pattern analysis and machine intelligence*, vol. 22, num. 3, pp. 281-297, 2000

[57] C.J. Kim. An algorithmic approach for fuzzy inference. *IEEE Trans. on fuzzy systems*, vol. 5, num. 4, pp. 585-598, 1997

[58] E. Kim, M. Park, S. Ji and M.Park. A new approach to fuzzy modeling. *IEEE Trans. on fuzzy systems*, vol. 5, num. 3, pp. 328-337, 1997

[59] G.J. Klir and T.A. Folger. *Fuzzy sets, uncertainty and information*, Prentice Hall, 1988

[60] L.T. Koczy and K. Hirota. Size reduction by interpolation in fuzzy rules. *IEEE Trans. on systems, man and cybernetics*, vol. 27, num. 1, pp. 14-25, 1997

[61] S. Kong and B. Kosko. Comparison of fuzzy and neural truck backer-upper control systems. *Proceedings of the international joint conference on neural networks*, vol. III, pp. 349-358, 1990

[62] S. Kong and B. Kosko. Adaptive fuzzy systems for backing up a truck-and-trailer. *IEEE Trans. on neural networks*, vol. 3, num. 2, pp. 211-223, 1992

[63] J.R. Koza. A genetic approach to the truck backer upper problem and the inter-twined spiral problem. *Proceedings of the international joint conference on neural networks*, vol. 4, pp. 310-318, 1992

[64] R. Krishnapuram and J.M. Keller. The possibilistic c-means algorithm: insights and recommendations. *IEEE Trans. on fuzzy systems*, vol. 4, num. 3, pp. 385-395, 1996

[65] A. Krone, P. Krause and A. Slawinski. A new rule reduction method for finding interpretable and small rule bases in high dimensional search spaces. *Proceedings of the 9th IEEE international conference on fuzzy systems*, pp. 693-699, 2000

[66] G. Lakoff. Hedges: a study in meaning criteria and the logic of fuzzy modeling. *Fuzzy sets and systems*, vol. 123, num. 3, pp. 343-358, 2001

[67] P.M. Larsen. Industrial applications of fuzzy logic control. *Int. Journal in man, machine studies*, vol. 12, num. 1, pp. 3-10, 1980

[68] W. van Leekwicjk and E.E. Kerre. Defuzzification: criteria and classification. *Fuzzy sets and systems*, vol. 108, num. 2, pp. 159-178, 1999

[69] Y. Lin and G.A. Cunningham III. A new approach to fuzzy-neural system modeling. *IEEE Trans. on fuzzy systems*, vol. 3, num. 2, pp. 190-197, 1995

[70] Y. Lin and G.A. Cunningham III and S.V. Coggeshall. Input variable identification - Fuzzy curves and fuzzy surfaces. *Fuzzy sets and systems*, num. 82, pp. 65-71, 1996

[71] B.D. Liu, C.Y. Chen and J.Y. Tsao. Design of adaptive fuzzy logic controller based on linguistic-hedge concepts and genetic algorithms. *IEEE Trans. on systems, man and cybernetics - Part B: cybernetics*, vol. 31, num. 1, pp. 32-53, 2001

[72] M. Mackey and L. Glass. Oscillation and chaos in a physiological control system. *Science*, vol. 197, pp. 287-289, 1977

[73] S.G. Makridakis, S.C. Wheelwright and R.J. Hyndman. *Forecasting: methods and applications - 3rd edition*, Wiley, 1998

[74] E.H. Mamdani and S. Assilian. An experiment in linguistic synthesis with fuzzy logic controler. *International journal of man-machine studies*, vol. 7, pp. 1-13, 1975

[75] E.H. Mamdani. Applications of fuzzy algorithms for control a simple dynamic plant. *Proceedings of the IEE*, vol. 12, pp. 1585-1588, 1974

[76] J.G. Marín-Blázquez and Q. Shen. Regaining comprehensibility of approximate fuzzy models via the use of linguistic hedges. *Interpretability issues in fuzzy modeling*, Studies in fuzziness and soft computing, vol. 128, Springer, pp. 25-53, 2003

[77] S. Marsili-Libelli, G. Pacini and C. Barresi. Fuzzy prediction of the algal blooms in the Orbetello lagoon. *2002 iEMSs International Meeting Proceedings*, pp. 438-443, 2002

[78] J.M. Mendel. Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE*, vol. 83, num. 3, pp. 345-377, 1995

[79] G.A. Miller. The magic number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, vol 63, pp. 81-97, 1956

[80] S. Mohaghegh, K. Mohamad, A. Popa and S. Ameri. Performance drivers in restimulation of Gas Storage Wells. *SPE eastern regional conference and exhibition*, num. 57453, 1999

[81] J.E. Nash and J.V. Sutcliffe. River flow forecasting through conceptual models: a discussion of principles. *Journal of Hydrology*, num. 10, pp. 282-290, 1970

[82] D.H. Nguyen and B. Widrow. The truck backer-upper: an example of self-learning in neural network. *Proceedings of the international joint conference on neural networks*, vol. II, pp. 357-363, 1989

[83] D.H. Nguyen and B. Widrow. Neural networks for self-learning control systems. *IEEE control systems magazine*, vol. 10, num. 2, pp. 18-23, 1990

[84] V. Novák. A horizon shifting model of linguistic hedges for approximate reasoning. *Proceedings of the 5th IEEE international conference on fuzzy systems*, pp. 423-427, 1996

[85] N.R. Pal and J.C. Bezdek. On cluster validity for the fuzzy c-means model. *IEEE Trans. on fuzzy systems*, vol. 3, num. 3, pp. 370-379, 1995

[86] W. Pedrycz. An identification algorithm in fuzzy relational systems. *Fuzzy sets and systems*, vol. 13, pp. 153-167, 1984

[87] W. Pedrycz. Fuzzy equalization in the construction of fuzzy sets. *Fuzzy sets and systems*, vol. 119, num. 2, pp. 329-335, 2001

[88] W. Pedrycz. Expressing relevance and interpretability of rule-based systems. *Interpretability issues in fuzzy modeling*, Studies in fuzziness and soft computing, vol. 128, Springer, pp. 546-567, 2003

[89] W. Pedrycz and J. Valiente de Oliveira. Optimization of fuzzy models. *IEEE Trans. on systems, man and cybernetics - Part B: cybernetics*, vol. 26, num. 4, pp. 627-637, 1996

[90] X. Peng and P. Wang. On generating linguistic rules for fuzzy models. *Lectures notes comput. sci. uncertainty intell. syst.*, vol. 313, pp. 185-192, 1988

[91] C.A. Peña-Reyes and M. Sipper. Fuzzy CoCo: a cooperative-coevolutionary approach to fuzzy modeling. *IEEE Trans. on fuzzy systems*, vol. 9, num. 5, pp. 727-737, 2001

[92] C.A. Peña-Reyes and M. Sipper. Fuzzy CoCo: balancing accuracy and interpretability of fuzzy models by means of coevolution. *Accuracy improvements in linguistic fuzzy modeling*, Studies in fuzziness and soft computing, vol. 129, Springer, pp. 119-146, 2003

[93] W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling. *Numerical recipes in C*, Cambridge University Press, 1993

[94] H. Pomares, I. Rojas, J. Ortega, J. González and A. Prieto. A systematic approach to a self-generating fuzzy rule-table for function approximation. *IEEE Trans. on systems, man and cybernetics - Part B: cybernetics*, vol. 30, num. 3, pp. 431-337, 2000

[95] H. Pomares, I. Rojas and J. González. Automatic construction of fuzzy rule-based systems: a trade-off between complexity and accuracy maintaining interpretability. *Accuracy improvements in linguistic fuzzy modeling*, Studies in fuzziness and soft computing, vol. 129, Springer, pp. 193-219, 2003

[96] M.A. Potter and K.A. De Jong. Cooperative coevolution: an arquitecture for evolving coadapted subcomponents. *Evolutionary computation*, vol. 8, num. 1, pp. 1-29, 2000

[97] P.A. Ramamoorthy and S. Huang. Fuzzy expert systems vs neural networks - Truck backer-upper control revisited. *Proceedings. of the IEEE Intl. Conf. on Systems Engineering*, pp. 221-224, 1991

[98] A. Riid. Transparent fuzzy systems: modeling and control. *Ph.D. dissertation*, Tallin Technical University, 2002

[99] A. Riid and E. Rüstern. Transparent fuzzy systems in modeling and control. *Interpretability issues in fuzzy modeling*, Studies in fuzziness and soft computing, vol. 128, Springer, pp. 452-476, 2003

[100] I. Rojas, H. Pomares, J. Ortega and A. Prieto. Self-organized fuzzy systems generation from training examples. *IEEE Trans. on fuzzy systems*, vol. 8, num. 1, pp. 23-36, 2000

[101] H. Roubos and M. Setnes. Compact and transparent fuzzy models and classifiers through iterative complexity reduction. *IEEE Trans. on fuzzy systems*, vol. 9, num. 4, pp. 516-524, 2001

[102] R. Rovatti, R. Guerrieri and G. Baccarani. An enhanced two-level boolean synthesis methodology for fuzzy rules minimization. *IEEE Trans. on fuzzy systems*, vol. 3, pp. 288-299, 1995

[103] M. Schoenauer and E. Ronald. Neuro-genetic truck backer-upper controller. *Proceedings of the IEEE conference on computational intelligence*, pp. 720-723, 1994

[104] M. Setnes. Simplification and reduction of fuzzy rules. *Interpretability issues in fuzzy modeling*, Studies in fuzziness and soft computing, vol. 128, Springer, pp. 278-302, 2003

[105] M. Setnes, R. Babuska, U. Kaymak and H.R. van Nauta Lemke. Similarity measures in fuzzy rule base simplification. *IEEE Trans. on systems, man and cybernetics - Part B: cybernetics*, vol. 28, num. 3, pp. 376-386, 1998

[106] M. Setnes, R. Babuska and H.B. Verbruggen. Rule-based modeling: precision and transparency. *IEEE Trans. on systems, man and cybernetics - Part C: applications and reviews*, vol. 28, num. 1, pp. 165-169, 1998

[107] M. Setnes and H. Roubos. GA-based modeling and classification: complexity and performance. *IEEE Trans. on fuzzy systems*, vol. 8, num. 5, pp. 509-522, 2000

[108] T. Sudkamp, A. Knapp and J. Knapp. Model generation by domain refinement and rule reduction. *IEEE Trans. on systems, man and cybernetics - Part B: cybernetics*, vol. 33, num. 1, pp. 45-55, 2003

[109] M. Sugeno and G. Kang. Structure identification of fuzzy model. *Fuzzy sets and systems*, num. 28, pp. 15-33, 1988

[110] M. Sugeno and K. Tanaka. Successive identification of a fuzzy model and its applications to prediction of a complex system. *Fuzzy sets and systems*, vol. 42, pp. 315-334, 1991

[111] M. Sugeno and T. Yasukawa. A fuzzy-logic-based approach to qualitative modeling. *IEEE Trans. on fuzzy systems*, vol. 1, num. 1, pp. 7-31, 1993

[112] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. on system, man and cybernetics*, vol. 15, pp. 115-132, 1985

[113] R.M. Tong. The evaluation of fuzzy models derived from experimental data. *Fuzzy sets and systems*, vol. 4, pp. 1-12, 1980

[114] J. Valiente de Oliveira. Semantic constraints for membership function optimization. *IEEE Trans. on systems, man and cybernetics - Part A: systems and humans*, vol. 29, num. 1, pp. 128-138, 1999

[115] J. Valiente de Oliveira. Towards neuro-linguistic modeling: constraints for optimization of membership functions. *Fuzzy sets and systems*, vol. 106, num. 3, pp. 357-380, 1999

[116] J. Valiente de Oliveira and P. Fazendeiro. On the achievement of both accurate and interpretable fuzzy systems using data-driven design processes. *Accuracy improvements in linguistic fuzzy modeling*, Studies in fuzziness and soft computing, vol. 129, Springer, pp. 147-162, 2003

[117] L.X. Wang. Fuzzy systems are universal approximators. *IEEE International Conference on Fuzzy Systems*, pp. 1163-1170, 1992

[118] L. Wang and R. Langari. Building Sugeno-type models using fuzzy discretization and orthogonal parameter estimation techniques. *IEEE Trans. on fuzzy systems*, vol. 3, pp. 454-458, 1995

[119] L. Wang and R. Langari. Complex systems modeling via fuzzy logic. *IEEE Trans. on system, man and cybernetics*, vol. 26, pp. 100-106, 1996

[120] L.X. Wang and J.M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Trans. on system, man and cybernetics*, vol. 22, num. 6, pp. 1414-1427, 1992

[121] W. Weiss. Risk reduction with fuzzy expert exploration tool. *Second annual technical progress report of the petroleum recovery research center*, New Mexico, 2001

[122] C.W. Xu and Y.Z. Lu. Fuzzy model identification and self-learning for dynamic systems. *IEEE Trans. on system, man and cybernetics*, vol. SMC-17, pp. 683-689, 1987

[123] J. Yen and L. Wang. Simplifying fuzzy rule-based models using orthogonal transformation methods. *IEEE Trans. on systems, man and cybernetics - Part B: cybernetics*, vol. 29, num. 1, pp. 13-24, 1999

[124] Y. Yoshinari, W. Pedrycz and K. Hirota. Construction of fuzzy models through clustering techniques. *Fuzzy sets and systems*, vol. 54, pp. 157-165, 1993

[125] L.A. Zadeh. Fuzzy sets. *Information and control*, vol. 8, pp. 338-353, 1965

[126] L.A. Zadeh. A fuzzy-set theoretic interpretation of linguistic hedges. *Journal of cybernetics*, vol. 2, num. 2, pp. 4-34, 1972

[127] L.A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. on systems, man and cybernetics*, vol. 3, pp. 28-44, 1973

B

List of publications

This thesis has summarized the author's research experience in the field of fuzzy modeling for the last seven years. Obviously, during its development we have published several articles related to this work. Here we enumerate them and we also include their abstracts. They are sorted by its publication date in descending order.

1. **An Intelligible Approach for the Synthesis of Intelligible Fuzzy Models**
   Submitted to Fuzzy Sets and Systems (but not accepted yet)
   International Fuzzy Systems Association & Elsevier
   Elsevier, 2005
   We present an heuristic methodology devised to address the problems encountered in designing an intelligible fuzzy model to fit a set of input-output data. In particular, we are able to determine the number of fuzzy sets, place them in the universe of scope and propose a set of linguistic rules that relate them.

   The resulting method is very simple and also intelligible. Therefore, it performs the final models with a low computational cost but furthermore, if helps the tuning of its different options based on the nature of the problem and the nature of the users. Thus, observe that we will focus this work not only on the intelligibility of the model but also on the intelligibility of the method itself.

   We do not seek to conclude that our method is better than others but to obtain an acceptable error in comparison while keeping the linguistic capacities of the fuzzy model. In fact with this methodology we will be able to choose the precision of the model and consequently its degree of intelligibility.

2. **A Fast Approach to Synthesize Intelligible Fuzzy Systems from Input-Output Data**
   Proceedings of the IFSA'05 Word Congress
   International Fuzzy Systems Association
   Beijing, 2005
   We present a set of heuristic criteria devised to address the problems encountered in designing a fuzzy controller to fit a set of input-output data. The objective is to obtain in a fast and simple manner an intelligible model able to undergo further refinements. We detail the method, compare it with other alternatives and finally some examples are given.

3. **Intelligible Fuzzy Models Applied to Time Series Prediction and Control**
   Poster for the 7-th CCIA
   Catalan Association for Artificial Intelligence
   Barcelona, 2004
   A simple and fast method to build fuzzy systems from input-output data consists in computing optimal fuzzy curves whose linearization can define the necessary fuzzy sets. In this paper we review this method and show its capacities to predict the popular Box and Jenkins' time series and to control the ball and beam system.

4. **Building Controllers from Optimal Fuzzy-Curves: A Simple and Intelligible Approach**
   Proceedings of the 2004-EUROFUSE Workshop on Data and Knowledge Engineering
   European Chapter of the International Fuzzy Systems Association
   Warsaw, 2004
   In this paper we present a set of heuristic criteria devised to address the problems encountered in designing a fuzzy controller to fit a set of input-output data. The objective is to obtain an intelligible starting control able to undergo further refinements. The method is exemplified by means of two popular cases: the truck backer-upper and the ball and beam problem.

5. **An Approach of Fuzzy Modeling towards Intelligible Modeling**
   Proceedings of the 5th WSEAS Int.Conf. on fuzzy sets and fuzzy systems
   World Scientific and Engineering Academy and Society
   Udine, 2004
   In this paper we present a set of heuristic criteria devised to address the

problems encountered in designing a fuzzy system to fit a set of input-output data. The objective is to obtain in a simple and fast manner a good starting model to undergo further refinements. The result is a simple algorithm with a similar performance than other techniques but with a low computational cost.

6. **Min-square Fitting of Fuzzy Curves**
   Proceedings of the IFSA'03 World Congress
   International Fuzzy Systems Association
   Istanbul, 2003
   Fuzzy curves proposed by Lin *et al.* deliver a smooth representation of the relation between two variables from the weighted average of near samples. This average is taken from samples inside a window of adjustable size by means of a parameter defined in the fuzzy curves, the b or β parameter, which is adjusted empirically until the moment. This paper proposes a method to fit this parameter so that the fuzzy curve presents the minimum square error between its result and the samples.

7. **Extracting Relevant Information from Input-Output Data**
   Poster for the 4-th CCIA
   Catalan Association for Artificial Intelligence
   Barcelona, 2001
   In this paper a set of heuristic criteria devised to address the problems encountered in designing a fuzzy system from input-output data is presented. In particular, we show how to discriminate unsignificant linguistic variables, determine the number of fuzzy sets, place them in the universe of scope, and propose a set of linguistic rules. The objective is to obtain in a simple and fast manner an algorithm simpler than the standard ones with minimum computational cost but still similar performance and more intelligibility in most cases.

8. **Automatic Process for the Synthesis of Fuzzy Systems from Input-Output Data**
   Proceedings of the 1999 EUSFLAT-ESTYLF Joint Conference
   European Society for Fuzzy Logic and Technology
   Palma de Mallorca, 1999
   In this paper we present a set heuristic criteria devised to address the problems encountered in designing a fuzzy system to fit a set of input-output data. In particular, we discriminate unsignificant linguistic variables, determine the number of fuzzy sets, place them in the universe of scope, propose a set of linguistic rules and give the necessary number

of bits to represent each variable. The objective is to obtain in a simple and fast manner a good starting model to undergo further refinements.

C
Algorithms

Here we provide the algorithm of the methodology that we have proposed in this work with the Wang&Mendel's option and the Chiu's clustering option. We detail all the steps except for some few computations that we have considered in the real program in order to diminish the elapsed time.

In order to facilitate its understanding we have considered for every variable the same name in all procedures. Every variable is treated as a global variable and thus, the whole algorithm can be obtained by joining directly all the procedures. Furthermore we have included some comments where the methodology could be difficult to understand.

Every variable is identified with an intelligible name and in most cases they have the same name that we have used when we have explained the methodology.

Anyway, among all the variables there is one of them, `fuzzy_curve`, probably one of the most significant, which should be introduced here because it has not been explained before. This variable is a 3-dimensional array where the first number indicates the input variable, the second number indicates the fuzzy set of this variable and the third number indicates the type of information which may be a *1* if it indicates the value of the universe of scope (UoS), a *2* if it indicates the value of the fuzzy curve in this point, a *3* if it indicates the value of the linearized fuzzy curve in this point, a *4* if it indicates the error between the real value and the linearized value in this point or a *5* if it indicates the rounded value of this error.

Furthermore in every call to a function we include with a subscript the number of the algorithm in order to locate it quickly. For example the call $\texttt{ComputeOptimalBeta}_4$ means that this procedure can be found in the algorithm number 4.

Nevertheless, some generic functions are not detailed here. These are the following:

- `Abs` → Absolute value.
- `All` → True if all elements of a vector are nonzero.
- `Any` → True if any element of a vector is nonzero.
- `Bisection` → Bisection method.
- `Ceil` → Round towards plus infinity.
- `Exp` → Exponential.
- `Find` → Find indices of nonzero elements.
- `Floor` → Round towards minus infinity.
- `FuzzySystem` → Perform fuzzy inference calculations.
- `Length` → Length of a vector or matrix.
- `Log` → Natural logarithm.
- `LogCommon` → Common base 10 logarithm.
- `LogDivision` → Logarithmically spaced vector.
- `Max` → Largest value.
- `Mean` → Mean value.
- `Min` → Smallest value.
- `Prod` → Product of values.
- `Rand` → Uniformly distributed random numbers.
- `Round` → Round towards nearest integer.
- `Sort` → Sort in ascending order.
- `Sqrt` → Square root.
- `Std` → Standard deviation.
- `Sum` → Sum of values.
- `TStudentInv` → Inverse of Student's T cumulative distribution function.

```
input  : the samples in a matrix samples of size #_samples × #_variables
         the desired error in a variable desired_error between 0 and 100
output: the fuzzy sets in a matrix sets_final of size #_variables × #_sets
         the rules in a matrix rules_final of size #_rules × #_variables
         the RMSE in a variable model_rmserror_final
         the NRMSE in a variable model_nrmserror_final
begin
    num_samples ← number of rows of samples
    num_variables ← number of columns of samples
    num_inputs ← num_variables-1
    ComputeRoundValues 2
    ComputeUoSPoints 3
    ComputeOptimalBeta 4
    ComputeFuzzyCurves 7
    EvaluateOddFuzzyCurve 8
    ExtremaFuzzySets 9
    best_model_nrmserror ← ∞
    repeat
        ComputeLinearFuzzyCurve 10
        PossibleOutSetsWithWangMendel 11
        ClusteringOutSetsWithChiu 12
        EvaluateCurrentModel 13
        EvaluateStopDecision 14
    until end_of_process =TRUE
end
```

Algorithm 1: Main procedure

```
begin
    for i ← 1 to num_variables do
        round_value[i] ← (Max(samples)-Min(samples))×(desired_error/100)
        round_value_rounded[i] ← 1
        if round_value[i] < 0.5 then
            while round_value[i] < 0.5 do
                round_value[i] ← round_value[i] × 10
                round_value_rounded[i] ← round_value_rounded[i]/10
            endw
        else
            while round_value[i] > 0.5 do
                round_value[i] ← round_value[i]/10
                round_value_rounded[i] ← round_value_rounded[i] × 10
            endw
        endif
        round_value[i] ← round_value_rounded
    endfor
end
```

Algorithm 2: ComputeRoundValues

```
begin
    for i ← 1 to num_inputs do
        accepted_points ← NULL
        min_point ← Floor(Min(samples[:, i])/round_value[i])×round_value[i]
        step_point ← round_value[i]
        max_point ← Ceil(Max(samples[:, i])/round_value[i])×round_value[i]
        current_point ← min_point
        repeat
            accepted_points ← [accepted_points current_point]
            current_point ← current_point + step_point
        until current_point > max_point
        num_points[i] ← 0
        for j ← 1 to Length(accepted_points) do
            if Any(Abs(samples[:, i] − possible_points[j])≤ round_value[i]) then
                accepted_points ← [accepted_points possible_points[j]]
                num_points[i] ← num_points[i] + 1
            endif
        endfor
        fuzzy_curve[i, :, 1] ← accepted_points
    endfor
end
```

Algorithm 3: ComputeUoSPoints

```
begin
    for i ← 1 to num_inputs do
        round_input_value ← round_value[i]
        GroupedSamplesToComputeOptimalBeta 5
        optimal_beta_mean ← NULL
        optimal_beta_std ← NULL
        current_error ← ∞
        current_iteration ← 1
        while (current_error > desired_error)OR(current_iteration ≤ 21) do
            TestTrainPointsToComputeOptimalBeta 6
            possible_min ← NULL
            for j ← 2 to Length(derivative_square_error) do
                if (derivative_square_error[j, 2] ≥
                0)AND(derivative_square_error[j − 1, 2] ≤ 0) then
                    possible_min[:, 1] ←
                    [possible_min[:, 1] derivative_square_error[j − 1, 1]]
                    possible_min[:, 2] ← [possible_min[:, 2] derivative_square_error[j, 1]]
                endif
            endfor
            local_min ← Bisection(possible_min)
            if Length(local_min)=1 then
                optimal_beta[current_iteration] ← local_min
            else
                Comment: Choosing the global minimum.
                square_error ← NULL
                for k ← 1 to Length(local_min) do
                    square_error[k] ← 0
                    for q ← 1 to Length(grouped_samples) do
                        numerator ← Sum(Exp(−((train_points[:
                        , 1] − test_points[q, 1])/(local_min[k])))×train_points[:, 2])
                        denominator ← Sum(Exp(−((train_points[:
                        , 1] − test_points[q, 1])/(local_min[k]))))
                        square_error[k] ← square_error[k] + 0.5 × (test_points[q, 2] −
                        (numerator/denominator))^2
                    endfor
                endfor
                optimal_beta[current_iteration] ←
                Mean(local_min[Find(square_error = Min(square_error))])
            endif
            optimal_beta_mean[current_iteration] ← Mean(optimal_beta)
            optimal_beta_std[current_iteration] ←
            Std(optimal_beta/Sqrt(current_iteration))
            current_error ←
            100 × TStudentInv((200-desired_error)/200,current_iteration-1) ×
            optimal_beta_std[current_iteration]/optimal_beta_mean[current_iteration]
            current_iteration ← current_iteration + 1
        endw
        optimal_beta[i] ← optimal_beta_mean[current_iteration − 1]
    endfor
end
```

Algorithm 4: ComputeOptimalBeta

```
begin
    satisfactory_group_of_samples ← FALSE
    while satisfactory_group_of_samples = FALSE do
        min_point ← Floor(Min(samples[:, i])/round_input_value)×round_input_value
        step_point ← round_value[i]
        max_point ← Ceil(Max(samples[:, i])/round_input_value)×round_input_value
        current_point ← min_point
        repeat
            possible_points ← [possible_points current_point]
            current_point ← current_point + step_point
        until current_point > max_point
        grouped_samples ← NULL
        for j ← 1 to Length(possible_points) do
            grouped_samples[Length(grouped_samples) + 1, :, 1] ← samples
            [Find(Abs(samples[:, i] − possible_points[j])≤(round_input_value/2)),1]
            grouped_samples[Length(grouped_samples), :, 2] ← samples
            [Find(Abs(samples[:, i] − possible_points[j])≤(round_input_value/2)),2]
        endfor
        if Length(grouped_samples)>(0.5×Length(possible_points)) then
            satisfactory_group_of_samples ← TRUE
        else
            round_input_value ← round_input_value × 2
        endif
    endw
end
```

Algorithm 5: GroupedSamplesToComputeOptimalBeta

```
begin
    satisfactory_partition ← FALSE
    while satisfactory_partition = FALSE do
        test_points ← NULL
        train_points ← NULL
        for j ← 1 to Length(grouped_samples) do
            k ← Rand(1 to Length(grouped_samples))
            test_points[:, 1] ← [test_points[:, 1] grouped_samples[j, k[1], 1]]
            test_points[:, 2] ← [test_points[:, 2] grouped_samples[j, k[1], 2]]
            train_points[:, 1] ← [train_points[:, 1] grouped_samples[j, k[2], 1]]
            train_points[:, 2] ← [train_points[:, 2] grouped_samples[j, k[2], 2]]
        endfor
        beta_min ← ∞
        beta_max ← 0
        for j ← 1 to Length(grouped_samples) do
            current_dist ← Sort(Abs(test_points[i, 1] − train_points[:, 1]))
            min_dist ← current_dist
            [Min(Find(((current_dist-Min(current_dist))>0)))]^2−Min(current_dist)^2
            beta_min ← Min([beta_min Sqrt(−(min_dist/Log(0.01 × desired_error)))])
            max_dist ← Max(current_dist)^2−Min(current_dist)^2
            beta_max ← Max([beta_max Sqrt(−(max_dist
            /Log(0.01×(100−desired_error))))])
        endfor
        num_decades ← LogCommon(beta_max)-LogCommon(beta_min)
        points_per_decade ← 3
        square_error[:, 1] ← LogDivision(beta_min to
        beta_max,num_decades,points_per_decade)
        derivative_square_error[:, 1] ← square_error[:, 1]
        for j ← 1 to Length(derivative_square_error) do
            derivative_square_error[j, 2] ← 0
            for k ← 1 to Length(grouped_samples) do
                A ← Sum(Exp(−((test_points[k, 1] − train_points[:
                , 1])/derivative_square_error[j, 1])^2)×(test_points[k, 2] − train_points[:
                , 2]))
                B ← Sum(Exp(−((test_points[k, 1] − train_points[:
                , 1])/derivative_square_error[j, 1])^2)×((test_points[k, 1] − train_points[:
                , 1])^2) × train_points[:, 2])
                C ← Sum(Exp(−((test_points[k, 1] − train_points[:
                , 1])/derivative_square_error[j, 1])^2))
                D ← Sum(Exp(−((test_points[k, 1] − train_points[:
                , 1])/derivative_square_error[j, 1])^2)×((test_points[k, 1] − train_points[:
                , 1])^2))
                E ← Sum(Exp(−((test_points[k, 1] − train_points[:
                , 1])/derivative_square_error[j, 1])^2)×train_points[:, 2])
                derivative_square_error[j, 2] ←
                derivative_square_error[j, 2] + (A × (−B × C + D × E))/(C^3)
            endfor
            derivative_square_error[j, 2] ←
            derivative_square_error[j, 2]/(derivative_square_error[j, 1]^2)
        endfor
        for j ← 2 to Length(derivative_square_error) do
            if (derivative_square_error[j, 2] ≥ 0)AND(derivative_square_error[j − 1, 2] ≤ 0)
            then
                satisfactory_partition ← TRUE
            endif
        endfor
    endw
end
```

Algorithm 6: TestTrainPointsToComputeOptimalBeta

```
begin
    for i ← 1 to num_inputs do
        for j ← 1 to num_points[i] do
            numerator ← Sum(Exp(−((fuzzy_curve[i, j, 1] − samples[:
            , i])/optimal_beta[i])^2)×samples[:, num_variables])
            denominator ←
            Sum(Exp(−((fuzzy_curve[i, j, 1] − samples[:, i])/optimal_beta[i])^2))
            fuzzy_curve[i, j, 2] ← numerator/denominator
        endfor
    endfor
end
```

Algorithm 7: ComputeFuzzyCurve

```
begin
    for i ← 1 to num_inputs do
        is_odd_function ← TRUE
        global_mid_point ← Sum(fuzzy_curve[i, 1 to num_points[i], 2])/num_points[i]
        for j ← 2 to num_points[i]/2 do
            current_mid_point ←
            (fuzzy_curve[i, j, 2] + fuzzy_curve[i, num_points[i] − j + 1, 2])/2
            if ((current_mid_point < global_mid_point − ((desired_error/100) ×
            round_value(num_variables)))OR(current_mid_point >
            global_mid_point + ((desired_error/100) × round_value(num_variables)))) then
                is_odd_function ← FALSE
            endif
        endfor
    endfor
end
```

Algorithm 8: EvaluateOddFuzzyCurve

```
begin
    input_sets ← NULL
    if Any(is_odd_function)=FALSE then
        for i ← 1 to num_inputs do
            input_sets[i, :, 1] ← [fuzzy_curve[i, 1, 1] fuzzy_curve[i, num_points[i], 1]]
            input_sets[i, :, 2] ← [fuzzy_curve[i, 1, 2] fuzzy_curve[i, num_points[i], 2]]
            num_sets[i] = 2
        endfor
    else
        for i ← 1 to num_inputs do
            if is_odd_function[i]=FALSE then
                input_sets[i, :, 1] ← [fuzzy_curve[i, 1, 1] fuzzy_curve[i, num_points[i], 1]]
                input_sets[i, :, 2] ← [fuzzy_curve[i, 1, 2] fuzzy_curve[i, num_points[i], 2]]
                num_sets[i] = 2
            else
                if (num_points[i]/2)−Floor(num_points[i]/2))=0 then
                    input_sets[i, :, 1] ←
                    [fuzzy_curve[i, 1, 1] (fuzzy_curve[i, num_points[i]/2, 1] +
                    fuzzy_curve[i, num_points[i]/2 + 1, 1])/2 fuzzy_curve[i, num_points[i], 1]]
                    input_sets[i, :, 2] ←
                    [fuzzy_curve[i, 1, 2] (fuzzy_curve[i, num_points[i]/2, 2] +
                    fuzzy_curve[i, num_points[i]/2 + 1, 2])/2 fuzzy_curve[i, num_points[i], 2]]
                else
                    input_sets[i, :, 1] ← [fuzzy_curve[i, 1, 1]
                    fuzzy_curve[i,Ceil(num_points[i]/2),1] fuzzy_curve[i, num_points[i], 1]]
                    input_sets[i, :, 2] ← [fuzzy_curve[i, 1, 2]
                    fuzzy_curve[i,Ceil(num_points[i]/2),2] fuzzy_curve[i, num_points[i], 2]]
                endif
                num_sets[i] = 3
            endif
        endfor
    endif
end
```

Algorithm 9: ExtremaFuzzySets

```
begin
    for i ← 1 to num_inputs do
        if fuzzy_curve[:, :, 3]=NULL then
            Comment: 1^st iteration. Compute every point of the Univ. of Scope.
            for j ← 1 to num_points[i] do
                edge_low ← Sum(input_sets[i, :, 1] ≤ fuzzy_curve[i, j, 1])
                edge_low ← num_sets[i]−Sum(input_sets[i, :, 1] ≥ fuzzy_curve[i, j, 1])+1
                if edge_low <edge_high then
                    fuzzy_curve[i, j, 3] ← (input_sets[i, edge_high, 2] −
                    input_sets[i, edge_low, 2])/(input_sets[i, edge_high, 1] −
                    input_sets[i, edge_low, 1]) × (fuzzy_curve[i, j, 1] −
                    input_sets[i, edge_low, 1]) + input_sets[i, edge_low, 2]
                else
                    fuzzy_curve[i, j, 3] ← input_sets[i, edge_low, 2]
                endif
                fuzzy_curve[i, j, 4] ← Abs(fuzzy_curve[i, j, 2] − fuzzy_curve[i, j, 3])
                fuzzy_curve[i, j, 5] ← Round(fuzzy_curve[i, j, 4])
            endfor
        else
            Comment: Compute only those which may change.
            for j ← 2 to num_sets[i] − 1 do
                if All(old_sets[i, :, 1] ≠ input_sets[i, j, , 1]) then
                    points_to_modify ← Find((fuzzy_curve[i, :, 1] >
                    input_sets[i, j − 1])AND(fuzzy_curve[i, :, 1] < input_sets[i, J + 1]))
                    for k ← 1 to Length(points_to_modify) do
                        edge_low ←
                        Sum(input_sets[i, :, 1] ≤ fuzzy_curve[i, points_to_modify[k], 1])
                        edge_high ← num_sets[i]−Sum(input_sets[i, :, 1] ≥
                        fuzzy_curve[i, points_to_modify[k], 1])+1
                        if edge_low < edge_high then
                            fuzzy_curve[i, points_to_modify[k], 3] ←
                            (input_sets[i, edge_high, 2] −
                            input_sets[i, edge_low, 2])/(input_sets[i, edge_high, 1] −
                            input_sets[i, edge_low, 1]) ×
                            (fuzzy_curve[i, points_to_modify[k], 1] −
                            input_sets[i, edge_low, 1]) + input_sets[i, edge_low, 2]
                        else
                            fuzzy_curve[i, points_to_modify[k], 3] ←
                            input_sets[i, edge_low, 2]
                        endif
                        fuzzy_curve[i, points_to_modify[k], 4] ←
                        Abs(fuzzy_curve[i, points_to_modify[k], 2] −
                        fuzzy_curve[i, points_to_modify[k], 3])
                        fuzzy_curve[i, points_to_modify[k], 5] ←
                        Round(fuzzy_curve[i, points_to_modify[k], 4]/((desired_error/100)×
                        round_value[num_variables]))×((desired_error/100) ×
                        round_value[num_variables])
                    endfor
                endif
            endfor
        endif
    endfor
end
```

Algorithm 10: ComputeLinearFuzzyCurve

```
begin
    num_rules ← Prod(num_sets)
    rules ← NULL
    for r ← 1 to num_rules do
        for i ← 1 to num_inputs do
            n ← Prod(num_sets[i + 1 to num_inputs])
            rules[r, i] ← Ceil(r/n)
            rules[r, i] ← rules[r, i] − num_sets[i]×Ceil(rules[r, i]/num_sets[i] − 1)
            rules[r, i] ← input_sets[i, rules[r, i], 1]
        endfor
        is_new_rule ← TRUE
        for old_r ← 1 to Length(old_rules) do
            if All(rules[r, 1 to num_inputs] = old_rules[old_r, 1 to num_inputs]) then
                rules[r, num_variables] ← old_rules[old_r, num_variables]
                is_new_rule ← FALSE
            endif
        endfor
        if is_new_rule = TRUE then
            fuzzyfication ← NULL
            for i ← 1 to num_inputs do
                c ← Find(input_sets[i, :, 1] = rules[r, i])
                if c =1 then
                    fuzzyfication_left ← (input_sets[i, c + 1] − samples[:
                    , 1])/(input_sets[i, c + 1, 1] − input_sets[i, c, 1])
                    fuzzyfication_left[Find(fuzzyfication_left > 1)] ← 1
                    fuzzyfication_left[Find(fuzzyfication_left < 0)] ← 0
                    fuzzyfication_right ← NULL
                else if c < num_sets[i] then
                    fuzzyfication_left ← (input_sets[i, c + 1, 1] − samples[:
                    , 1])/(input_sets[i, c + 1, 1] − input_sets[i, c, 1])
                    fuzzyfication_left[Find((fuzzyfication_left >
                    1)OR(fuzzyfication_left < 0))] ← 0
                    fuzzyfication_right ← (samples[:
                    , 1] − input_sets[i, c − 1, 1])/(input_sets[i, c, 1] − input_sets[i, c − 1, 1])
                    fuzzyfication_right[Find((fuzzyfication_right >
                    1)OR(fuzzyfication_right < 0))] ← 0
                endif
                else
                    fuzzyfication_right ← (samples[:
                    , 1] − input_sets[i, c − 1, 1])/(input_sets[i, c, 1] − input_sets[i, c − 1, 1])
                    fuzzyfication_right[Find(fuzzyfication_right > 1)] ← 1
                    fuzzyfication_right[Find(fuzzyfication_right < 0)] ← 0
                    fuzzyfication_left ← NULL
                endif
                fuzzyfication[:, i] ← Max(fuzzyfication_left fuzzyfication_right)
            endfor
            fuzzyfication ← Prod(fuzzyfication)
            if Max(fuzzyfication)>0 then
                rules[r, num_variables] ←
                Mean(samples[Find(fuzzyfication =Max(fuzzyfication)) num_variables])
            endif
            old_rules ← [old_rules rules[r, :]]
        endif
    endfor
    num_rules ← Length(rules)
end
```

Algorithm 11: PossibleOutSetsWithWangMendel

```
begin
    sets_to_cluster ← Sort(rules[:, num_variables])
    for i ← 1 to num_inputs do
        possible_points ← NULL
        min_point ←
        Round(Min(sets_to_cluster)/round_value[num_variables])×round_value[num_variables]
        step_point ← round_value[num_variables]
        max_point ←
        Round(Max(sets_to_cluster)/round_value[num_variables])×round_value[num_variables]
        current_point ← min_point
        repeat
            possible_points ← [possible_points current_point]
            current_point ← current_point + step_point
        until current_point > max_point
    endfor
    for i ← 1 to Length(possible_points) do
        if Any(Abs(points − possible_points[i])≤ (round_value/2)) then
            points ← [points possible_points[i]]
        endif
    endfor
    n ← Length(points)
    radius_a ← (desired_error/100) × (Max(sets_to_cluster)−Min(sets_to_cluster))
    radius_b ← radius_a
    alpha_parameter ← −Log(desired_error/100)/(radius_a^2)
    beta_parameter ← −Log(desired_error/100)/(radius_b^2)
    for i ← 1 to n do
        p_factor[i] ← Sum(Exp(−alpha_parameter × (points[i] − points)^2))
    endfor
    last_max_p_factor ← Max(p_factor)
    max_p_factor ← last_max_p_factor
    Comment: Initial cluster.
    possible_cluster ← sets_to_cluster[Find(p_factor = max_p_factor)]
    cluster ← possible_cluster[1]
    added_cluster ← possible_cluster[1]
    for i ← 2 to Length(possible_cluster) do
        if All(cluster ≠ possible_cluster[i]) then
            cluster ← [cluster possible_cluster[i]]
            added_cluster ← [added_cluster possible_cluster[i]]
        endif
    endfor
    Comment: Next clusters.
    no_more_clusters ← FALSE
    while no_more_clusters = FALSE do
        for i ← 1 to Length(added_cluster) do
            p_factor ← p_factor − last_max_p_factor×Exp(−beta_parameter × (points −
            added_cluster[i])^2)
        endfor
        last_max_p_factor ← Max(p_factor)
        if last_max_p_factor > ((desired_error/100) × max_p_factor) then
            possible_cluster ← points[Find(p_factor = last_max_p_factor)]
            added_cluster ← NULL
            for i ← 1 to Length(possible_cluster) do
                if All(cluster ≠ possible_cluster[i]) then
                    cluster ← [cluster possible_cluster[i]]
                    added_cluster ← [added_cluster possible_cluster[i]]
                endif
            endfor
        else
            no_more_clusters ← TRUE
        endif
    endw
    possible_output_sets ← Sort(cluster)
end
```

Algorithm 12: ClusteringOutSetsWithChiu

```
begin
    Comment: Adding the boundaries of the UoS in possible_output_sets.
    if All(possible_output_sets ≠min(rules[:, num_variables])) then
        possible_output_sets ← [possible_output_sets min(rules[:, num_variables])]
    endif
    if All(possible_output_sets ≠max(rules[:, num_variables])) then
        possible_output_sets ← [possible_output_sets max(rules[:, num_variables])]
    endif
    Comment: Choosing the closest possible output set to the singleton of each rule.
    for r ← 1 to num_rules do
        rules[r, num_variables] ←
        round(mean(possible_output_sets[Find(Abs(possible_output_sets −
        rules[r, num_variables])=min(Abs(possible_output_sets −
        rules[r, num_variables])))]))
    endfor
    Comment: We have the model.
    output_sets ← rules[:, num_variables]
    for i ← 1 to num_inputs do
        sets[i, :] ← input_sets[i, :, 1]
    endfor
    sets[num_variables, :] ← output_sets
    Comment: Computing the error of the model.
    output_with_model ← FuzzySystem(sets,rules,samples)
    model_nrmserror ← Sqrt(mean((output_with_model − samples[:
    , num_variables])^2)/mean((samples[:, num_variables]−mean(samples[:, num_variables]))^2))
    model_rmserror ← Sqrt(mean((output_with_model − samples[:, num_variables])^2))
    for i ← 1 to num_inputs do
        fuzzy_curve_nrmserror[i] ← Sqrt(mean(fuzzy_curve[i, :, 4]^2)/mean((samples[:
        , num_variables]−mean(samples[:, num_variables]))^2))
    endfor
    if model_nrmserror < best_model_nrmserror then
        sets_final ← sets
        rules_final ← rules
        model_nrmserror_final ← model_nrmserror
        model_rmserror_final ← model_rmserror
        best_model_nrmserror ← model_nrmserror
    endif
end
```

Algorithm 13: EvaluateCurrentModel

```
begin
    if (model_nrmserror ≤ desired_error)OR(All(fuzzy_curve_nrmserror ≤ desired_error))
    then
        Comment: Successful end of process.
        end_of_process ← TRUE
    else if Sum(Sum(fuzzy_curve[:, :, 5]=0)=num_points)=num_inputs then
        Comment: No more sets are considered due to a high desired error.
        end_of_process ← TRUE
    else
        Comment: Search the variable with highest error in order to diminish it.
        old_sets ← input_sets
        highest_error ← Max(fuzzy_curve[:, :, 5])
        input_with_highest_error ← Find(highest_error =Max(highest_error))
        for i ← 1 to Length(input_with_highest_error) do
            new_set ← NULL
            possible_new_set ← NULL
            max_error_found ← −∞
            for j ← 1 to num_points[input_with_highest_error[i]] do
                if fuzzy_curve[input_with_highest_error[i], j, 5] =
                highest_error[input_with_highest_error[i] then
                    if fuzzy_curve[input_with_highest_error[i], j, 4] > max_error_found
                    then
                        possible_new_set ←
                        [fuzzy_curve[input_with_highest_error[i], j, 1]
                        fuzzy_curve[input_with_highest_error[i], j, 2]]
                        max_error_found ← fuzzy_curve[input_with_highest_error[i], j, 4]
                    else if
                    fuzzy_curve[input_with_highest_error[i], j, 4] = max_error_found then
                        possible_new_set[1] ←
                        [possible_new_set[1] fuzzy_curve[input_with_highest_error[i], j, 1]
                        possible_new_set[2] ←
                        [possible_new_set[2] fuzzy_curve[input_with_highest_error[i], j, 2]
                    endif
                endif
                if (fuzzy_curve[input_with_highest_error[i], j, 5] <
                highest_error[input_with_highest_error[i]])AND(Length(possible_new_set)>
                0) then
                    new_set[1] ← [new_set[1] Mean(possible_new_set[1])]
                    new_set[2] ← [new_set[2] Mean(possible_new_set[2])]
                    possible_new_set ← NULL
                    max_error_found ← −∞
                endif
            endfor
            if Length(possible_new_set) then
                new_set[1] ← [new_set[1] Mean(possible_new_set[1])]
                new_set[2] ← [new_set[2] Mean(possible_new_set[2])]
            endif
            sets_current_variable[:, 1] ← [input_sets[input_with_highest_error[i], 1 to
            num_sets[input_with_highest_error[i], 1] new_set[1]]
            sets_current_variable[:, 2] ← [input_sets[input_with_highest_error[i], 1 to
            num_sets[input_with_highest_error[i], 2] new_set[2]]
            sets_current_variable ← Sort(sets_current_variable)
            input_sets[input_with_highest_error[i], :, 1] ← sets_current_variable[:, 1]
            input_sets[input_with_highest_error[i], :, 2] ← sets_current_variable[:, 2]
            num_sets[input_with_highest_error[i]] ←
            Sum(input_sets[input_with_highest_error[i], :, 1]] > −∞)
        endfor
    endif
end
```

Algorithm 14: EvaluateStopDecision