International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems
Vol. 6, No. 2 (1998) 171–178
© World Scientific Publishing Company

MISSING VALUES AND LEARNING OF FUZZY RULES

MICHAEL R. BERTHOLD
Computer Science Division, Department of EECS
University of California, Berkeley, CA 94720, USA
eMail: `Michael.Berthold@Informatik.Uni-Karlsruhe.DE`

and

KLAUS–PETER HUBER
Institute for Computer Design and Fault Tolerance (Prof. D. Schmid)
University of Karlsruhe, Am Zirkel 2, 76128 Karlsruhe, Germany
eMail: `KlausPeter.Huber@Informatik.Uni-Karlsruhe.DE`

Received 12 September 1997
Revised

In this paper a technique is proposed to tolerate missing values based on a system of fuzzy rules for classification. The presented method is mathematically solid but nevertheless easy and efficient to implement. Three possible applications of this methodology are outlined: the classification of patterns with an incomplete feature vector, the completion of the input vector when a certain class is desired, and the training or automatic construction of a fuzzy rule set based on incomplete training data. In contrast to a static replacement of the missing values, here the evolving model is used to predict the most possible values for the missing attributes. Benchmark datasets are used to demonstrate the capability of the presented approach in a fuzzy learning environment.

Keywords: Fuzzy Rules; Classification; Learning; Missing Values

1. Introduction

Pattern classification applications often involve incomplete information; that is, some parts of the input vector are missing. This may be due to unrecorded information or occlusions of features, for example in vision applications. Therefore the classifier f has to deal with an input vector $\vec{x} = (\vec{v}, ?)$ with some known part $\vec{v}$. In this context three different scenarios can be of interest:

- classification with missing values; that is, the model is used to predict the most likely class of the incomplete input vector:

$$f(\vec{v}, ?) = \ldots \tag{1}$$

  Liu et al[4] discuss several methods to deal with this problem, where the focus lies on techniques that are based on decision trees. These methods are, however, time consuming to use.

- completion of input vectors. If the set of fuzzy rules is used to describe the behaviour of a system, then it is often of interest to find good values for all or part of the input features when a certain outcome is desired. This is similar to finding an inverse function of the model:

$$f(\vec{v}, \ldots) = k \tag{2}$$

- training with incomplete data. The last issue concerns training of a set of fuzzy rules when some or all of the training data is incomplete. To use as much of the available information as possible during training, it is desirable to make use of incomplete training vectors as well:

$$\ldots(\vec{v}, ?) = k \tag{3}$$

In this paper we concentrate on classifiers which are based on fuzzy rules[10,11], in contrast to the work presented by Tresp and his co-workers[1,8]. This simplifies the resulting math and allows to estimate the best guess for the missing values efficiently. Few restrictions on the type of fuzzy system are assumed; that is, rules have to be normalized and standard min/max-inference is required. For most practical applications these restrictions are of no relevance.

This paper is organized as follows. In the next section the used terminology will be established. Thereafter a way to classify patterns with incomplete features will be presented. Then this method will be extended to find the most likely value for these features. After that it is described how automatic learning of fuzzy rules can be adapted to tolerate missing values in the training data, followed by a section describing experimental results.

2. Fuzzy Rule Based Classifiers

A fuzzy rule based classifier[11] consists of a set of rules R_i^k for each class k with $1 \leq k \leq c$, $1 \leq i \leq m_k$, where c denotes the number of classes and m_k indicates the number of rules for class k:

$$R_i^k : \quad \texttt{IF} \quad \mathrm{x_1} \texttt{ is } \mathrm{A_{i,1}^k} \texttt{ and } \cdots \texttt{ and } \mathrm{x_n} \texttt{ is } \mathrm{A_{i,n}^k} \quad \texttt{THEN class } \mathrm{k} \tag{4}$$

The feature space is thus of dimensionality n; each feature vector $\vec{x}$ consists of n attributes: $\vec{x} = (x_1, \cdots, x_j, \cdots, x_n)$, $x_j \in \mathcal{R}$ ($\mathcal{R}$ denotes the domain of the attributes, usually the real numbers). A rule's activity $\mu_i^k(\vec{x})$ $(1 \leq i \leq m_k)$ indicates the degree of membership of the pattern $\vec{x}$ to the corresponding rule R_i^k:

$$\mu_i^k(\vec{x}) = \min_{1 \leq j \leq n} \left\{\mu_{i,j}^k(x_j)\right\} \tag{5}$$

Using the normal notation of fuzzy rules, where each rule can be decomposed into n individual one-dimensional membership functions, corresponding to the fuzzy sets $A_{i,j}^k$. Thus $\mu_{i,j}^k$ represents the projection of μ_i^k onto the j-th attribute. In addition, we assume normalized, one-dimensional membership functions for all rules i of all classes k:

$$\forall x_j \in \mathcal{R} \ : \ 0 \leq \mu^k_{i,j}(x_j) \leq 1 \quad \text{and} \quad \exists x_j \in \mathcal{R} \ : \ \mu^k_{i,j}(x_j) = 1 \tag{6}$$

The degree of membership for the entire class is then computed using:

$$\mu^k(\vec{x}) = \max_{1 \leq i \leq m_k} \left\{ \mu^k_i(\vec{x}) \right\} \tag{7}$$

3. Classification with Missing Values

For the case of missing input values; that is, attributes where no information about their true value is available, the feature vector can be written as: $\vec{x} = (\vec{v}, ?)$, where $\vec{v} = (v_1, ..., v_{n_v})$ represents the part of the vector with the known values and ? replaces $\vec{u} = (u_1, ..., u_{n_u})$, the vector of unknown attributes ($n_v + n_u = n$). The "best possible" activation of one rule can then be obtained through:

$$\mu^k_i(\vec{v}, ?) = \sup_{\vec{u} \in \mathcal{R}^{n_u}} \left\{ \mu^k_i(\vec{v}, \vec{u}) \right\} \overset{\text{eq.5}}{=} \min_{\substack{1 \leq j \leq n_v \\ 1 \leq l \leq n_u}} \left\{ \mu^k_{i,j}(v_j), \sup_{u_l \in \mathcal{R}} \{ \mu^k_{i,l}(u_l) \} \right\} \tag{8}$$

Since we are dealing with normalized fuzzy rules (equation 6), $\mu^k_{i,l}$ will have a maximum value of 1 somewhere on $\mathcal{R}$, and this equation can be simplified to:

$$\mu^k_i(\vec{v}, ?) = \min_{1 \leq j \leq n_v} \left\{ \mu^k_{i,j}(v_j), 1 \right\} = \min_{1 \leq j \leq n_v} \left\{ \mu^k_{i,j}(v_j) \right\} \tag{9}$$

independent of $\vec{u}$; that is, it is sufficient to compute the activation of each rule in the projection of $\mathcal{R}^n$ on $\mathcal{R}^{n_v}$. This means that only the activation for each one-dimensional membership function of the known attributes has to be computed.

The results obtained here are similar to the results obtained by Ahmad and Tresp[1] for Gaussian Basis Functions but are more efficient to compute. In the case of fuzzy rules the only assumption made is the fact that the rule's activity is composed of one-dimensional normalized activation functions.

4. Input Vector Completion

If the input-output pattern $(\vec{x}, k)$ (input $\vec{x}$, class k) has missing values the same notation as above can be used: $\vec{x} = (\vec{v}, \vec{u})$. Usually finding the best guess for $\vec{u}$ is computationally expensive since the maximum of the model-function has to be found in $\mathcal{R}^u$. In the case of fuzzy rules it becomes easier, the maximum of $\mu^k(\vec{v}, \vec{u})$ in respect to $\vec{u}$ has to be found:

$$\begin{aligned}
\sup_{\vec{u} \in \mathcal{R}^{n_u}} \left\{ \mu^k(\vec{v}, \vec{u}) \right\} &\overset{\text{eq.7}}{=} \sup_{\vec{u} \in \mathcal{R}^{n_u}} \left\{ \max_{1 \leq i \leq m_k} \left\{ \mu^k_i(\vec{v}, \vec{u}) \right\} \right\} \\
&= \max_{1 \leq i \leq m_k} \left\{ \sup_{\vec{u} \in \mathcal{R}^{n_u}} \left\{ \mu^k_i(\vec{v}, \vec{u}) \right\} \right\} \\
&\overset{\text{eq.8}}{=} \max_{1 \leq i \leq m_k} \left\{ \mu^k_i(\vec{v}, ?) \right\}
\end{aligned} \tag{10}$$

This results in the rule with the highest degree of membership being selected:

$$i_{\max} = \underset{1 \le i \le m_k}{\operatorname{argmax}} \left\{ \mu_i^k(\vec{v}, ?) \right\} \overset{\text{eq.9}}{=} \underset{1 \le i \le m_k}{\operatorname{argmax}} \left\{ \min_{1 \le j \le n_v} \left\{ \mu_{i,j}^k(v_j) \right\} \right\} \tag{11}$$

In the case of several maxima one of them can be chosen arbitrarily. The restriction $\vec{v}$ on the best rule leads to a not necessarily normalized fuzzy rule in $\mathcal{R}^{n_u}$ with a subspace $A_{\max}(\vec{u}|\vec{v})$ of maximal degree of membership under the condition $\vec{v}$:

$$A_{\max}(\vec{u}|\vec{v}) = \left\{ \vec{u} \mid \forall \vec{w} \in \mathcal{R}^{n_u} \ : \ \mu_{i_{\max}}^k(\vec{v}, \vec{u}) \ge \mu_{i_{\max}}^k(\vec{v}, \vec{w}) \right\} \tag{12}$$

Using equation 8 this can be simplified to:

$$A_{\max}(\vec{u}|\vec{v}) = \left\{ \vec{u} \mid \mu_{i_{\max}}^k(\vec{v}, \vec{u}) \ge \mu_{i_{\max}}^k(\vec{v}, ?) \right\} \tag{13}$$

and $\mu_i^k(\vec{v}, ?)$ is the activation of the rule using the missing values (see equation 9). $A_{\max}(\vec{u}|\vec{v})$ describes a hyper-rectangle in n_u-dimensional space that is equal to the α-cut of $\min_{1 \le l \le n_u}\{\mu_{i_{\max}}^k(u_l)\}$ using $\alpha = \mu_{i_{\max}}^k(\vec{v}, ?)$ and therefore easy to obtain.

Now the "best" vector $(\vec{v}, \vec{u}_{\max})$ can be computed, using a typical defuzzification technique, for example the Center of Gravity:

$$\vec{u}_{\max} = \frac{\int\limits_{\vec{u} \in A_{\max}} \vec{u} \cdot \mu_{i_{\max}}^k(\vec{v}, \vec{u}) \, d\vec{u}}{\int\limits_{\vec{u} \in A_{\max}} \mu_{i_{\max}}^k(\vec{v}, \vec{u}) \, d\vec{u}} \tag{14}$$

Figure 1 illustrates this procedure using a two dimensional example with one known and one unknown attribute. For the activation function a trapezoidal membership function is used, featuring a core- $(\mu = 1)$ and a support-region $(0 < \mu < 1)^2$.

5. Training with Missing Values

Training with missing values is usually being done by substituting the mean or another fixed value for each missing attribute. Tresp et al[8] demonstrate how this can lead to severe problems during training. More sophisticated methodologies can be used, but for practical applications an efficient implementation is also required, especially in the case of large datasets. The approach presented here uses the rule model that evolves during training to predict the most possible value for each of the missing attributes. As was shown above this is computationally uncritical, only the rule with the highest activation has to be found and from its core-region the value of $\vec{u}_{\max}$ can be quickly extracted.

A training vector with missing values is thus handeled in two steps, first the best guess $\vec{u}_{\max}$ for the vector of missing attributes $\vec{u}$ is determined, based on the known values for $\vec{v}$. Then normal training is executed using the best guess for $\vec{u}$ together with the known values of the input $\vec{v}$ resulting in a complete $\vec{x} = (\vec{v}, \vec{u}_{\max})$.

In practice there are two possible ways to implement this type of training, depending on the underlying rule learning algorithm. If the algorithm operates iteratively and the evolving rule set is meaningful during intermediate presentations of

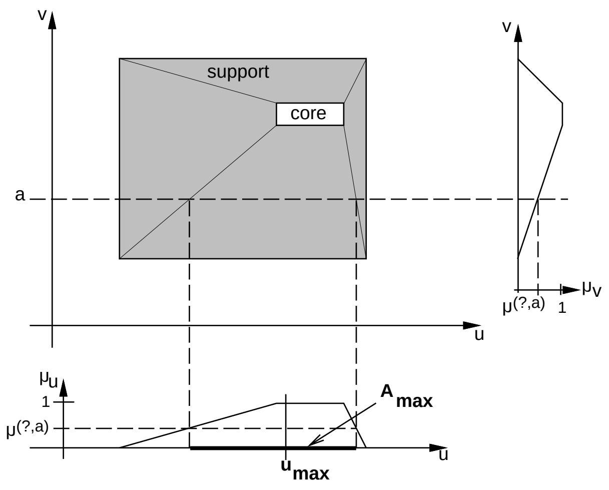

Figure 1: The proposed way to find a best guess for a missing value. The attribute v has the value a, u is unknown. Using the described method $\mu(?, a)$ can be obtained resulting in an area of maximum activity $A_{\max}$. The "best guess" for u is found using the Center of Gravity method resulting in $u_{\max}$.

training patterns, the current rule set can be used to replace the missing values. If the algorithm on the other hand is batch oriented; that is, the rule set is only meaningful after training is finished, the rule set of the previous epoch is used to replace the missing values. In the following the later approach has been used. At the beginning all missing values were replaced with the mean of the corresponding attribute (which is the "best guess" when no rule set is present). Then, after training using this dataset was finished, the constructed rule set was used to replace the missing values in the training data. With this new dataset the next training was performed, resulting in a new rule set, etc.

6. Experimental Results

To demonstrate the usefulness of the proposed method, an existing algorithm to construct a set of fuzzy rules from training data was adapted using the described technique to tolerate missing values during training. The underlying training algorithm[2] uses a set of training data points $(\vec{x}, k)$ to iteratively construct a set of locally independent fuzzy rules, in contrast to numerous other algorithms that build a global grid of rules[3,6,7,9]. The resulting rules only depend on relevant attributes, which is extremely beneficial when using a high-dimensional feature space. The algorithm is easy to use and trains fast. In addition its termination can be proven.

Two datasets from the StatLog-archive[5] have been used. This archive evolved during a european ESPRIT-project and contains a number of real world datasets together with results of over 20 different classifiers. Both used datasets are based on real data and contain a large number of examples. The SatImage dataset was chosen because of its complexity (high number of attributes, high remaining generalization error), the Shuttle dataset because of its size (about $60,000$ examples).

Since the used training algorithm does not require a validation dataset, training could be performed on the entire training set and the testing data was used to estimate the generalization capability of the constructed fuzzy rule set. Different levels of distortion were used to erase randomly chosen values from the training data, thus simulating different amounts of missing values. To estimate the quality of the resulting models, the remaining test data was used without distortion. In addition to the presented fuzzy based replacement of missing values, a static substitution using mean and zero was also used to compare the obtained results.

6.1. *Results on the SatImage Benchmark*

The SatImage dataset contains $4,435$ training cases divided into 6 classes in a 36-dimensional feature space. Testing is done on $2,000$ patterns. This dataset was used first to demonstrate how using the evolving ruleset to replace the missing values leads to a new rule set with better generalization. Table 1 shows the obtained results.

percentage of	proposed method		replace w. mean		replace w. zero	
missing values	error	#rules	error	#rules	error	#rules
0%	12.96%	506	12.96%	506	12.96%	506
1%	12.41%	526	12.86%	529	13.86%	645
5%	14.11%	532	14.51%	645	17.26%	966
10%	13.76%	501	15.86%	745	27.51%	1285
20%	15.31%	400	15.41%	843	43.12%	1444
40%	16.11%	303	16.66%	950	51.33%	1353
60%	16.96%	165	18.01%	1121	41.98%	1314

Table 1: Results on the SatImage dataset with different levels of missing values.

Obviously for this dataset the replacement with zero is disastrous, the replacement with the mean produces much better results. This illustrates clearly how the static substitution using a constant value can be harmful. The proposed method, however, finds the "best guess" dynamically, using the evolving rule set. This way to compute individual estimates for the missing values leads to rule sets with better generalization. Additionally the size of the constructed rule set grows slower with an increase in distortion. After a certain level of distortion ($> 5\%$ missing values) the size of the rule set even decreases, due to the increasing amount of freedom to choose a replacement for the missing values.

6.2. *Results on the Shuttle Benchmark*

The Shuttle data was used because of its size: $43,314$ training cases, 3 classes[a], 9

[a]Three other classes from the original dataset were ignored because their occurrence was $< 1\%$.

dimensions, and 14,442 patterns to test generalization. Table 2 shows results for the Shuttle dataset, where the difference in required number of rules between the different methodologies is even more drastic.

percentage of missing values	proposed method error	proposed method #rules	replace w. mean error	replace w. mean #rules	replace w. zero error	replace w. zero #rules
0%	0%	14	0%	14	0%	14
10%	0%	28	0%	183	0%	347
20%	0%	31	0%	449	0%	971
30%	0%	25	0.035%	934	0%	1730
40%	0%	23	0.035%	1616	0.160%	2704
60%	0%	6	0.160%	3481		

Table 2: Results on the Shuttle dataset with different levels of missing values.

At a level of 60% missing values, only 6 rules are required, while both other approaches need several thousand rules. With this dataset it is even more appearant how the rule-based, dynamic replacement of missing values reduces the number of required rules without loosing performance. In contrast, compared to the other methods the generalization performance is even marginally better.

7. Conclusions

A methodology was proposed to deal with missing values for classification, completion of inputs, and training using sets of fuzzy rules under the usual assumptions; that is, max/min-inference and normalized rules are used. A computationally very efficient way was derived, to compute the output of a fuzzy rule set when parts of the input vector are unknown. In addition the input vector can be completed, computing the so-called "best guess". During incremental training, the evolving set of fuzzy rules can also be used to compute the completed input vector, followed by a conventional training algorithm. Experimental results show that the presented methodology outperforms standard static replacement techniques.

8. Acknowledgments

Thanks go to Prof. D. Schmid for his support and the opportunity to work on this project. We also thank Ingrid Fischer and Frank Mayer for many helpful remarks.

9. References

1. S. Ahmad and V. Tresp. Some Solutions to the Missing Feature Problem in Vision. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems*, 5, pages 393–400, California, 1993. Morgan Kaufmann.
2. K.-P. Huber and M. R. Berthold. Building Precise Classifiers with Automatic Rule Extraction. In *IEEE International Conference on Neural Networks*, 3, pages 1263–1268, 1995.

3. C. M. Higgins and R. M. Goodman. Learning Fuzzy rule-based Neural Networks for Control. In *Advances in Neural Information Processing Systems*, 5, pages 350–357, California, 1993. Morgan Kaufmann.
4. W. Z. Liu, A. P. White, S. G. Thompson, and M. A. Bramer. Techniques for Dealing with Missing Values in Classification. In X. Liu, P. Cohen, and M. R. Berthold, editors, *Advances in Intelligent Data Analysis*, Lecture Notes in Computer Science, LNCS1280, pages 527–536, Springer Verlag, 1997.
5. D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood Limited, 1994.
6. Patrick K. Simpson. Fuzzy min-max Neural Networks – Part 1: Classification. *IEEE Transactions on Neural Networks*, 3(5):776–786, September 1992.
7. Patrick K. Simpson. Fuzzy min-max Neural Networks – Part 2: Clustering. *IEEE Transactions on Fuzzy Systems*, 1(1):32–45, January 1993.
8. V. Tresp, S. Ahmad, and R. Neuneier. Training Neural Networks with Deficient Data. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, 6, pages 128–135, California, 1994. Morgan Kaufmann.
9. L.-X. Wang and J. M. Mendel. Generating Rules by Learning from Examples. In *International Symposium on Intelligent Control*, pages 263–268. IEEE, 1991.
10. Lotfi A. Zadeh. Soft computing and Fuzzy Logic. *IEEE Software*, pages 48–56, November 1994.
11. Lotfi A. Zadeh. Fuzzy Logic and the Calculi of Fuzzy Rules and Fuzzy Graphs: A Precis. *Multi. Val. Logic*, 1, pages 1–38, 1996.