Erschienen in: Fuzzy neuro systems '99 : [computational intelligence] / Gerhard Brewka ... (eds.). - Leipzig : Leipziger Univ.-Verl., 1999. - S. 41-53. - ISBN 3-933240-75-1

Two Stage Fuzzy Models and Potential Outliers

Michael Berthold*

Berkeley Initiative in Soft Computing (BISC)
Department of EECS, Computer Science Division
University of California at Berkeley
329 Soda Hall, Berkeley, CA 94720, USA
Email: berthold@cs.berkeley.edu

Abstract

Outliers or distorted attributes very often severely interfere with data analysis algorithms that try to extract few meaningful rules. Most methods to deal with outliers try to completely ignore them. This can be potentially harmful since the very outlier that was ignored might have described a rare but still extremely interesting phenomena. In this paper we describe an approach that tries to build an interpretable model while still maintaining all the information in the data. This is achieved through a two stage process. A first phase builds an outlier-model for data points of low relevance, followed by a second stage which uses this model as filter and generates a simpler model, describing only examples with higher relevance, thus representing a more general concept. The outlier-model on the other hand may point out potential areas of interest to the user. Preliminary experiments indicate that the two models in fact have lower complexity and sometimes even offer superior performance.

1 Introduction

Many datasets obtained from real-world systems contain distorted elements, for example due to errors in measurements, sensor-failures, or simple recording problems. If the resulting data is to be analyzed by means of extracting an interpretable model from this data, these so-called "outliers" are often difficult to ignore. Many existing methodologies to build models from data

*M. Berthold was in part supported by a stipend of the "Gemeinsame Hochschulsonderprogramm III von Bund und Ländern" through the DAAD.

will try to incorporate the outliers, making the resulting model more complex and harder to interpret [2, 3, 6, 7, 8].

In this paper an approach is discussed which aims to model an evolving set of potential outliers through an additional outlier-model. This results in two models, one representing a more general concept which offers better understandability, and the other concentrating on potential outliers or regions with low evidence in the observed data. The presented approach is based on sets of fuzzy rules as a way to model imprecise relationships [9, 10]. Based on an existing, fast algorithm that constructs fuzzy rule sets from data [4], the presented approach builds a model for the entire data in a first phase. After completion this model is analyzed automatically and the parts with low relevance in respect to the training data are moved to the outlier model. The outlier model is then used as a filter for the second phase. The second phase generates a more general model, representing the data points with higher evidence. The final pair of models consists of one model for potential outliers and one model which represents the more general behavior. The resulting general model is less complex and therefore easier to interpret whereas the outlier-model may point out potential areas of interest to the user.

We will start out by describing a method to build fuzzy models based on example data, followed by a description of the two-stage fuzzy model generation. Thereafter we present results on two datasets, demonstrating the effects of the proposed methodology.

2 Learning Fuzzy Models from Data

2.1 Fuzzy Models

In the following we will concentrate on n-dimensional feature spaces with one dependent variable which is either continuous or divided into a set of classes. The goal of the discussed methods is to generate a set of rules which describe some available example data. The used rules describe an implication:

$$\mathcal{R}: \texttt{if "antecedent" then "consequent"}$$

and we will focus on the most commonly used form of rules, consisting of an antecedent in form of a constraint on the input variables:

$$x_1 \texttt{ is } A_1 \texttt{ and } \cdots \texttt{ and } x_n \texttt{ is } A_n \qquad \text{or short:} \quad \vec{x} \texttt{ is } \vec{A}$$

where the consequent in case of a classifier assigns a certain class or, in case of a fuzzy graph describing a functional relationship, a granule of the dependent or output variable: $\cdots y$ is B.

In the n-dimensional feature space the area of influence of rule $\mathcal{R}$ is specified by the vector of membership functions $(A_1, \cdots, A_n)$. Of interest is sometimes also a weight parameter which usually represents the percentage of training patterns explained by this rule.

2.2 Specializing Fuzzy Rules

In both cases the resulting training data consists of an input vector $\vec{x}$, together with a vector of membership values $\vec{\mu}^{\text{target}}$ for the desired classes or output granules. The algorithm described here also allows for some – but not necessarily all – features having predefined granulation and is based on two methods to build fuzzy rules and fuzzy graphs [4, 1]. As a-priori information we therefore have the following:
for the output:

- either the number C^{out} of classes in case of a classifier, or in case of a continuous output variable its granulation into a set of C^{out} linguistic values, described through membership functions $\mu_c^{\text{out}} : \mathbb{R} \rightarrow [0, 1]$, with $1 \leq c \leq C^{\text{out}}$.

and for the inputs:

- a number n_{granul} of features x_i $(1 \leq i \leq n_{\text{granul}})$ with a predefined granulation, defined through a set of C_i^{in} linguistic values, described through membership functions $\mu_i^{\text{in},c} : \mathbb{R} \rightarrow [0, 1]$, with $1 \leq c \leq C_i^{\text{in}}$ and $1 \leq i \leq n_{\text{granul}}$.
- and a number n_{free} of features x_i $(n_{\text{granul}} + 1 \leq i \leq n = n_{\text{granul}} + n_{\text{free}})$ without a-priori defined granulation[1].

Example 2.1 *In case of a three-dimensional feature space* $(n = 3)$ *with a continuous output variable "pressure", one could define* $C^{\text{out}} = 4$ *granules, where the linguistic values "very low", "low", "medium", and "high" are defined through the membership functions* μ_c^{out} $(c = 1, \ldots, 4)$. *The three input features could consist of one granulated feature* $(n_{\text{granul}} = 1)$ *temperature, with three linguistic values "cold", "warm", and "hot"* $(C_1^{\text{in}} = 3)$ *and another two free features* x_2 *and* x_3 $(n_{\text{free}} = 2)$ *for two sensor values.*

In addition to this information a set of training examples exists:

$$\mathbf{T} = \left\{ \left(\vec{x}^{(m)}, \vec{\mu}^{\text{target},(m)} \right) \mid 1 \leq m \leq M \right\}$$

[1] For clarity of representation we assume that the features are ordered with respect to being granulated or not. This does, however, not affect generality of the discussed approach.

where M indicates the number of examples in the set $\mathbf{T}$. In case of a classifier the values $\mu_c^{\text{target},(m)}$ indicate the degree of membership to class c for training example m. For a continuous output $\mu_c^{\text{target},(m)}$ specifies the desired degree of membership to granule c ($1 \le c \le C^{\text{out}}$).

The goal of the algorithm is thus to generate a set of rules $\mathbf{R}$, which describe the training data $\mathbf{T}$.

$$\mathbf{R} = \left\{ \mathcal{R}^{(r)} \,|\, 1 \le r \le R \right\}$$

Each of the R rules is specified through a set of constraints on the input domain and the index of the associated granule or class $g^{(r)}$ ($1 \le g^{(r)} \le C^{\text{out}}$):

$$\mathcal{R}^{(r)} : \texttt{if} \bigwedge_{i=1}^{n_{\text{granul}}} \text{cond}_{\text{granul},i}^{(r)} \text{ and } \bigwedge_{i=n_{\text{granul}}+1}^{n} \text{cond}_{\text{free},i}^{(r)} \texttt{ then } g^{(r)} \texttt{ with } w^{(r)}$$

where the constraint consists of two parts, $\text{cond}_{\text{granul},i}^{(r)}$ and $\text{cond}_{\text{free},i}^{(r)}$.

- The first part describes the restrictions on the granulated part of the feature space:

$$\text{cond}_{\text{granul},i}^{(r)} = x_i \texttt{ is } \bigvee_{c=1}^{C_i^{\text{in}}} \left(s_i^c \wedge \mu_i^{\text{in},c} \right)$$

  and s_i^c specifies the applicable linguistic values for that specific feature. If $s_i^c = 1$, value c is part of the condition, whereas $s_i^c = 0$ indicates its absence in $\text{cond}_{\text{granul},i}^{(r)}$.

  Example 2.2 *Assuming a linguistic variable "temperature" (index i) for rule r, the condition* $\text{cond}_{\text{granul},i}^{(r)}$ *could for example contain two linguistic values "cold" and "warm" out of the set of three possible values "cold"* ($\mu_i^{\text{in},1}$), *"warm"* ($\mu_i^{\text{in},2}$), *and "hot"* ($\mu_i^{\text{in},3}$). *Then* $C_i^{\text{in}} = 3$ *and* $s_i^1 = 1$, $s_i^2 = 1$, *and* $s_i^3 = 0$. *Rule* $\mathcal{R}^{(r)}$ *would thus have the following constraint (among others):*
  $\mathcal{R}^{(r)}$: if $\cdots$temperature is (cold or warm)$\cdots$ then $\cdots$

  In most cases $\text{cond}_{\text{granul},i}^{(r)}$ will actually contain all linguistic values ($s_i^* = 1$), indicating that this feature is not restricted at all.

- The second part of the constraints $\text{cond}_{\text{free},i}^{(r)}$, are specified through:

$$\text{cond}_{\text{free},i}^{(r)} = \begin{cases} \text{true} \\ x_i \texttt{ is } < a_i^{(r)}, b_i^{(r)}, c_i^{(r)}, d_i^{(r)} > \end{cases}$$

  which either specify no constraint at all, or a trapezoidal membership function in case a constraint was derived by the training algorithm.

The algorithm to derive such a set of rules $\mathbf{R}$ from a set of examples $\mathbf{T}$ is described in the following section.

2.3 Algorithmic Details

The training algorithm operates sequentially, considering one example pattern after the other. At start no rules are existent, during training new rules will be introduced and existing rules will be fine tuned. In order to be able to guarantee termination of the algorithm, each rule keeps track of the pattern that originally triggered its creation. This so-called "anchor" is denoted $\vec{\lambda}^{(r)}$ for rule $\mathcal{R}^{(r)}$. In addition we will use the following abbreviation:

$$\begin{aligned} \text{vol}(\mathcal{R}) &:= \int \mu_{\mathcal{R}}(\vec{x})\, \mathrm{d}\vec{x} \\ &= \int \cdots \int \prod_{i=1}^{n_{\text{granul}}} \left(\bigvee_{c=1}^{C_i^{\text{in},c}} s_i^c \cdot \mu_{\text{granul},i}^{\text{in},c}(x_i) \right) \cdot \prod_{i=n_{\text{granul}}+1}^{n} \left(\mu_{\text{free},i}^{\text{in},c}(x_i) \right) \mathrm{d}x_1 \cdots \mathrm{d}x_n \end{aligned}$$

which represents the volume rule $\mathcal{R}$ covers in the feature space,

$$\mu_{\mathcal{R}}(\vec{x}) := \bigwedge_{i=1}^{n_{\text{granul}}} \left(\bigvee_{c=1}^{C_i^{\text{in},c}} s_i^c \wedge \mu_{\text{granul},i}^{\text{in},c}(x_i) \right) \wedge \bigwedge_{i=n_{\text{granul}}+1}^{n} \mu_{\text{free},i}^{\text{in},c}(x_i)$$

which represents the degree of membership of input $\vec{x}$ to the antecedent of rule $\mathcal{R}$, that is its degree of fulfillment, and finally

$$\text{maxindex}(\vec{\mu}) := \arg\max \left\{ \mu_c \,|\, 1 \le c \le C^{\text{out}} \right\}$$

which determines the index of the component in $\vec{\mu}$ with the maximum degree of membership.

During training for each example pattern $(\vec{x}, \vec{\mu}^{\text{target}}) \in \mathbf{T}$ two different scenario can now occur:

- COVERED: in this case a rule index r_{win} exists which already describes the new pattern, and also has the largest degree of membership among all rules:

$$\begin{aligned} &\exists r_{\text{win}} : 1 \le r_{\text{win}} \le R : g^{(r_{\text{win}})} = \text{maxindex}(\vec{\mu}^{\text{target}}) \wedge \mu_{\mathcal{R}^{r_{\text{win}}}}(\vec{x}) > 0.0 \\ &\quad \wedge \forall r' \ne r_{\text{win}} : g^{(r')} = \text{maxindex}(\vec{\mu}^{\text{target}}) \rightarrow (\mu_{\mathcal{R}^{r_{\text{win}}}}(\vec{x}) \ge \mu_{\mathcal{R}^{r'}}(\vec{x})) \end{aligned}$$

  The core region of this rule will be increased along the non-granulated features to make sure it covers the new pattern (non-restricted free features do not need to be modified since they cover the pattern anyway):

$$\begin{aligned} &\forall i : n_{\text{granul}} + 1 \le i \le n : \\ &\quad \text{cond}_{\text{free},i}^{r_{\text{win}}} \ne \text{TRUE} \Rightarrow b_i^{\prime r_{\text{win}}} := \min\{b_i^{r_{\text{win}}}, x_i\} \wedge c_i^{\prime r_{\text{win}}} := \max\{c_i^{r_{\text{win}}}, x_i\} \end{aligned}$$

  and the weight of the rule will be incremented, thus keeping track of the number of patterns this rule explains: $w^{\prime r_{\text{win}}} := w^{r_{\text{win}}} + 1$.

- COMMIT: in contrast to the above case no rule exists which describes the new pattern. In this case a new rule will be introduced (committed), describing the new example:

  $\mathcal{R}^{(R+1)}$: if TRUE then $g^{(R+1)} := \text{maxindex}(\vec{\mu}^{\text{target}})$ with $w^{(R+1)} := 1$

  Obviously this rule is too general and will need to be specialized subsequently, through adaptation of the antecedent. (Note the weight $w^{(R+1)}$ which is initialized to 1, indicating that this rule explains one example pattern so far.) We also need to remember the pattern which triggered creation of this rule:

  $$\vec{\lambda}^{(R+1)} := \vec{x}.$$

In addition to the modification and creation of rules describing patterns it is also necessary to actively avoid conflicts, that is, rules describing wrong relations need to be modified. This is done through a third step, called SHRINK, which modifies rules that incorrectly cover a pattern. More precisely, all rules $\mathcal{R}^{(r)}$ will be investigated if they fulfill

$$1 \leq r \leq R \wedge \mu^{\text{target}}_{g^{(r)}} = 0 \rightarrow \mu_{\mathcal{R}^{(r)}}(\vec{x}) = 0$$

that is, if the rule's consequent has a degree of membership equal to zero for this particular target $\vec{\mu}^{\text{target}}$ then it also generates a degree of membership equal to zero for the corresponding input vector $\vec{x}$. For each rule which violates this condition the following SHRINK-procedure will be performed:

- SHRINK: The goal is to specialize the conflicting rule $\mathcal{R}^{(r)}$ in a manner which inhibits the conflict. This is done through specialization of the rule, that is, the area of influence will be shrunk to exclude the new example pattern. To achieve this several alternatives exist, which are applied in the following order:

  S1 : Restrict an already constrained granulated feature. In this scenario it is assumed that the conflict can be resolved by tightening an already existing constraint of a granulated feature. Hence we restrict ourselves to:

  $$\begin{aligned} &\forall i_{S1} : 1 \leq i_{S1} \leq n_{\text{granul}} \\ &\quad \wedge \exists c : 1 \leq c \leq C^{\text{in}}_{i_{S1}} \wedge s^c_{i_{S1}} = 0 \\ &\quad \wedge \exists c : 1 \leq c \leq C^{\text{in}}_{i_{S1}} \wedge s^c_{i_{S1}} = 1 \wedge \mu^{\text{in},c}_{i_{S1}}(\vec{x}) = 0 \end{aligned}$$

  that is, all granulated features for which at least one linguistic value is excluded from $\text{cond}^{(r)}_{\text{granul},i_{S1}}$ and at least one of the linguistic values in $\text{cond}^{(r)}_{\text{granul},i_{S1}}$ results in a degree of membership

equal to zero. The last requirement makes sure that we can restrict this feature without completely erasing rule $\mathcal{R}^{(r)}$. If no such feature can be found, alternative **S2** is probed.

Otherwise we compute the loss in rule r's volume, assuming that feature i_{S1} is used to eliminate the conflict, resulting in a revised rule $\mathcal{R}'^{(r)}_{i_{S1}}$ which is similar to rule $\mathcal{R}^{(r)}$, the only difference being the constraint on the granulated feature i_{S1}:

$$\text{cond}^{(r)}_{\text{granul},i_{S1}} := x_{i_{S1}} \text{ is } \bigvee_{c=1}^{C^{\text{in}}_{i_{S1}}} \underbrace{\left(s^c_{i_{S1}} \wedge \left(\mu^{\text{in},c}_{i_{S1}}(\vec{x}) = 0\right)\right)}_{=:s'^c_{i_{S1}}} \mu^{\text{in},c}_{i_{S1}}$$

all other constraints remain unchanged. From there we can compute the respective loss in rule r's volume:

$$\text{loss}_{i_{S1}} = \text{vol}\left(\mathcal{R}^{(r)}\right) - \text{vol}\left(\mathcal{R}'^{(r)}_{i_{S1}}\right)$$

We then choose the one feature $i_{S1,\text{best}}$ which minimizes this loss in volume, thus keeping the restricted rule as large as possible:

$$i_{S1,\text{best}} = \arg\min\{\text{loss}_{i_{S1}}\}$$

and replace rule $\mathcal{R}$ in the new set of rules:

$$\mathbf{R}' := \mathbf{R} \cup \{\mathcal{R}'^{(r)}_{i_{S1,\text{best}}}\} \backslash \{\mathcal{R}^{(r)}\}.$$

Example 2.3 *A rule $\mathcal{R}$ describes the fact that one buys oil when the temperature is either "cold" or "warm":*

$\mathcal{R}$: if *temperature* is *cold* or *warm* then *buy oil*

This rule might result in a conflict because an example household did not buy oil during "warm" temperatures. The revised rule would then be (through further restriction of feature temperature):

$\mathcal{R}'$: if *temperature* is *cold* then *buy oil*

S2 : Restrict an unconstrained granulated feature. If **S1** could not be applied we will try to resolve the conflict through restricting a previously unconstrained, granulated feature. We are therefore looking at the following features:

$$\begin{aligned} &\forall i_{S2} : 1 \leq i_{S2} \leq n_{\text{granul}} \\ &\quad \wedge \forall c : 1 \leq c \leq C^{\text{in}}_{i_{S2}} \wedge s^c_{i_{S2}} = 1 \\ &\quad \wedge \exists c : 1 \leq c \leq C^{\text{in}}_{i_{S2}} \wedge \mu^{\text{in},c}_{i_{S2}}(\vec{x}) = 0 \end{aligned}$$

that is, all linguistic are contained in the constraint and we again make also sure that at least one linguistic value will remain in the modified constraint. Similar to above, if no such feature can be found, alternative **S3** is probed.

Otherwise we proceed analogous to scenario **S1**, that is, we compute the possible losses in volume and choose the one feature for shrinkage which results in the minimum loss in volume.

Example 2.4 *The feature "temperature" of the rule from above can now not be restricted any further without eliminating the rule entirely.*

$$\mathcal{R}: \texttt{if } \textit{temperature} \texttt{ is } \textit{cold} \texttt{ then } \textit{buy oil}$$

Therefore the previously unconstrained feature "oil price" will be restricted and $\mathcal{R}$ might become:

$$\mathcal{R}': \texttt{if } \textit{temperature} \texttt{ is } \textit{cold} \texttt{ and } \textit{oilprice} \texttt{ is } \textit{low} \texttt{ then } \textit{buy oil}$$

S3 : Restrict an already constrained free feature. If none of the granulated features can be restricted to avoid the conflict, one of the remaining free features needs to be used. We will first try to tighten an already existing constraint on a free feature, that is we restrict ourselves to:

$$\begin{aligned} &\forall i_{S3} : n_{\text{free}} + 1 \leq i_{S3} \leq n \\ &\quad \wedge \text{cond}^{(r)}_{\text{free},i_{S3}} = x_{i_{S3}} \text{ is } < a^{(r)}_{i_{S3}}, b^{(r)}_{i_{S3}}, c^{(r)}_{i_{S3}}, d^{(r)}_{i_{S3}} > \end{aligned}$$

that is, all free features i for which a constraining trapezoidal membership function is already defined. If no such feature can be found, alternative **S4** is probed.

Otherwise – similar to cases **S1** and **S2** – we determine which of these features results in a minimum loss of rule-coverage. First for each feature i_{S3} we compute a modified trapezoid which eliminates the conflict:

$$\begin{aligned} &< a'^{(r)}_{i_{S3}}, b'^{(r)}_{i_{S3}}, c'^{(r)}_{i_{S3}}, d'^{(r)}_{i_{S3}} > \\ &\quad := \begin{cases} < x_{i_{S3}}, \max\{x_{i_{S3}}, b^{(r)}_{i_{S3}}\}, c^{(r)}_{i_{S3}}, d^{(r)}_{i_{S3}} > & : \quad x_{i_{S3}} < \lambda^{(r)}_{i_{S3}} \\ < a^{(r)}_{i_{S3}}, b^{(r)}_{i_{S3}}, \min\{x_{i_{S3}}, c^{(r)}_{i_{S3}}\}, x_{i_{S3}} > & : \quad x_{i_{S3}} > \lambda^{(r)}_{i_{S3}} \end{cases} \end{aligned}$$

that is, depending on the position of the conflict with respect to the anchor $\lambda_{i_{S3}}$ the left or right side of the trapezoid is modified. This ensures that the anchor will always remain inside this rule but the conflict will be moved to the border of the support area.

Similar to **S1** we replace $\text{cond}^{(r)}_{\text{free},i_{S3}}$ using the new trapezoid and compute the loss in volume for the modified rule. One Feature $i_{S3,\text{best}}$ will then be selected which minimizes this loss and the corresponding rule will be replaced in the set of rules $\mathbf{R}$.

Example 2.5

$$\mathcal{R}: \texttt{if } \textit{temperature} \texttt{ is } \textit{cold} \texttt{ and } x_{42} \texttt{ is } < 2,3,7,9 > \texttt{ then } \textit{buy oil}$$

might become, through restriction of feature x_{42} to avoid a conflict at $x_{42} = 8$:

$$\mathcal{R}': \texttt{if } \textit{temperature} \texttt{ is } \textit{cold} \texttt{ and } x_{42} \texttt{ is } < 2,3,7,8 > \texttt{ then } \textit{buy oil}$$

S4 : Restrict an unconstrained free feature. If no constrained free feature can be found, one of the unconstrained free feature needs to be constrained. That is we restrict ourselves to:

$$\begin{aligned} &\forall i_{S4} : n_{\text{free}} + 1 \leq i_{S4} \leq n \\ &\quad \wedge \text{cond}^{(r)}_{\text{free},i_{S4}} = \text{TRUE} \end{aligned}$$

that is, all free features i_{S4} for which a constraining trapezoidal membership function has not been defined. If no such feature can be found, a serious conflict has been encountered, that is, two example points have the same input vector $\vec{x}$ but conflicting targets $\vec{\mu}$. Usually this will result in a feedback to the user, pointing out this inconsistency in the data set.

Otherwise – similar to case **S3** – we determine which of these features results in a minimum loss of rule-coverage. First for each feature i_{S4} we compute a new trapezoid which eliminates the conflict:

$$\begin{aligned} &< a'^{(r)}_{i_{S4}}, b'^{(r)}_{i_{S4}}, c'^{(r)}_{i_{S4}}, d'^{(r)}_{i_{S4}} > \\ &\quad := \begin{cases} < x_{i_{S4}}, \lambda^{(r)}_{i_{S4}}, \lambda^{(r)}_{i_{S4}}, \sup\{x_{i_{S4}}\} > & : \quad x_{i_{S4}} < \lambda^{(r)}_{i_{S4}} \\ < \inf\{x_{i_{S4}}\}, \lambda^{(r)}_{i_{S4}}, \lambda^{(r)}_{i_{S4}}, x_{i_{S4}} >, & : \quad x_{i_{S4}} > \lambda^{(r)}_{i_{S4}} \end{cases} \end{aligned}$$

that is, depending on the position of the conflict with respect to the anchor $\lambda_{i_{S4}}$ the left or right side of the trapezoid is restricted. This ensures that the anchor will always remain inside this rule but the conflict x_i will be moved to the border of the support area. The core itself is set equal to the anchor λ_{S4} and will later be enlarged if this rule covers other patterns.

Similar to **S3** we then replace $\text{cond}^{(r)}_{\text{free},i_{S4}}$ and compute the loss in volume for the modified rule. Feature $i_{S4,\text{best}}$ will then be selected to minimize this loss and the respective rule will be replaced in the set of rule **R**.

Example 2.6

$$\mathcal{R}: \texttt{if } \textit{temperature} \texttt{ is } \textit{cold} \texttt{ then } \textit{buy oil}$$

might become (through restriction of the previously unconstrained feature x_{42}):

$$\mathcal{R}': \texttt{if } \textit{temperature} \texttt{ is } \textit{cold} \texttt{ and } x_{42} \texttt{ is } <0,3,3,7> \texttt{ then } \textit{buy oil}$$

if $\sup\{x_{42}\} = 0$, $\lambda_{42} = 3$, *and the conflict occurred at* $x_{42} = 7$.

3 Two Stage Fuzzy Models

Most existing algorithms to construct fuzzy rule sets from data have tremendous problems with noisy data or data containing outliers. Usually an excessive number of rules is being introduced simply to model noise and/or outliers. This also applies to the algorithm described in the previous section. This is due to the fact that these algorithms aim to generate conflict free rules, that is, examples encountered during training will result in a degree of membership > 0 only for those rules of the correct class (or granule). Unfortunately in case of spares outliers such an approach will, especially in high-dimensional feature spaces, result in an enormous amount of rules to avoid these conflicts. Figure 1 demonstrates this effect.

Using the already existing model we can, however, in many cases easily determine parts that have low relevance, based on their weight or another parameter which denotes individual relevance. In the approach here we will focus on the weight parameter $w^{(r)}$ which represents the number of training patterns covered by rule r. From this a measure for the importance

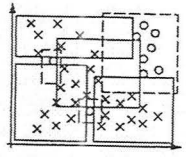

Figure 1: An example how two outliers (o) produce a high number of rules for the competing class ×.

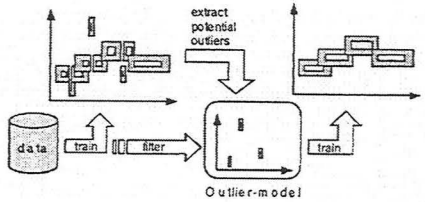

Figure 2: The role of the two models during training.

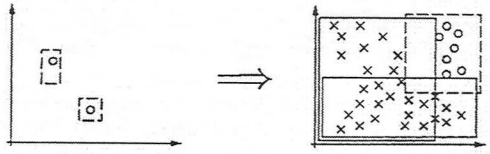

Figure 3: The same example, but here the additional outlier-model (left) reduces the number of generated rules in the second stage model (right).

or relevance of each rule can be derived, frequently we will simply use the percentage of patterns covered by a rule:

$$\Phi(R^{(r)}) = \frac{w^{(r)}}{|\mathbf{T}|}.$$

Using such a measure of relevance, we can now extract rules with low relevance from this model, assuming that they describe points in the data which are outliers or spares points. Using this "outlier"-model as filter for a second training phase will then generate a new fuzzy model which has less rules with higher significance. Figure 2 shows the flow of this procedure. Note, how the initial model is being used to extract the outlier-model. This model is then in turn used as a filter for the existing training data to generate the final model. In Figure 3 (left) the outlier model is shown, covering the two single points of class o. Now the second phase can ignore these points and generate two large rules describing the remaining data points of class × (right). In the following section we will show how this effects the size of the rule sets on two real-world datasets.

4 Experiments

Experiments on two datasets from the Statlog-archive [5] were performed to demonstrate the effect of the proposed methodology. The relevance function

Table 1: Results on the Satimage (left) and Segment (right) dataset

level of distort.	used model	error on test data	#rules
0.0%	Stand.	15.9%	393
	$\mathcal{H}_1(\mathcal{H}_0)$	13.5%	270 (60)
1.0%	Stand.	17.1%	394
	$\mathcal{H}_1(\mathcal{H}_0)$	13.5%	313 (81)
2.0%	Stand.	18.1%	404
	$\mathcal{H}_1(\mathcal{H}_0)$	12.9%	295 (109)
5.0%	Stand.	18.1%	479
	$\mathcal{H}_1(\mathcal{H}_0)$	12.4%	334 (145)
10.0%	Stand.	22.3%	578
	$\mathcal{H}_1(\mathcal{H}_0)$	15.2%	379 (199)

level of distort.	used model	error on test data	#rules
0.0%	Stand.	3.5%	96
	$\mathcal{H}_1(\mathcal{H}_0)$	3.0%	80 (12)
1.0%	Stand.	5.2%	108
	$\mathcal{H}_1(\mathcal{H}_0)$	4.3%	86 (22)
2.0%	Stand.	6.9%	113
	$\mathcal{H}_1(\mathcal{H}_0)$	5.6%	83 (30)
5.0%	Stand.	6.1%	144
	$\mathcal{H}_1(\mathcal{H}_0)$	3.8%	107 (37)
10.0%	Stand.	6.5%	151
	$\mathcal{H}_1(\mathcal{H}_0)$	6.5%	106 (45)

$\Phi(R^{(j)}) = w^{(j)}$ with a threshold of $\theta = 5$ was used, that is rules which cover less then five patterns were considered irrelevant.

The first dataset contains images from Satellites (Satimage-dataset), patterns with 36 attributes have to be separated into 6 different classes. All together 4435 training and 2000 test patterns were used. Table 1 (left) shows the results. Here "Standard" stands for the normal algorithm which generates fuzzy rules in one run. $\mathcal{H}_1$ indicates the general model generated through the algorithm explained above. The number of rules for both models is shown in the last column, the number before the brackets indicates the size of the rule-set of the general model $\mathcal{H}_1$, whereas the number in brackets denotes the number of rules of the outlier model $\mathcal{H}_0$. It is interesting to see how already without any additional distortion (0.0%) the two-stage model shows slightly better performance using a considerable smaller number of rules (270 vs. 393). Note also how the error rate on the unseen test data increases much slower with increases in distortion for the two-stage model. The gap between the sizes of the two models widens as well.

The second dataset is the Segment data from the same archive. Here 19 inputs and 7 classes are used with 2079 training and 220 test patterns. Table 1 (right) shows the results on this dataset. Here the effect in performance is not as obvious. Still noticeable, however, is the difference in model size. While the size of the separate outlier-model increases with increasing distortion, the size of the model representing the more general behavior stays almost the same.

5 Conclusions

In this paper we discussed a strategy to model potential outliers through an additional outlier-model. This results in two models, one representing a more general concept which offers better understandability, and the other concentrating on potential outliers or regions with low evidence in the observed data. In the future an entire hierarchy of fuzzy models seems to be a promising approach to model large amounts of data and enable the user to investigate the underlying behavior at various levels of granularity, following Lotfi Zadeh's concept of "information granulation".

References

[1] M. R. Berthold and K.-P. Huber. Constructing fuzzy graphs from examples. *Intelligent Data Analysis*, 3(1), 1999. (http://www.elsevier.nl/locate/ida).

[2] S. K. Halamuge and M. Glesner. FuNe Deluxe: A group of fuzzy-neural methods for complex data analysis problems. In *Proceedings of the EUFIT'95*, Aug. 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] 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.

[5] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood Limited, 1994.

[6] P. K. Simpson. Fuzzy min-max neural networks – part 1: Classification. *IEEE Transactions on Neural Networks*, 3(5):776–786, Sept. 1992.

[7] P. K. Simpson. Fuzzy min-max neural networks – part 2: Clustering. *IEEE Transactions on Fuzzy Systems*, 1(1):32–45, Jan. 1993.

[8] L.-X. Wang and J. M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6):1313–1427, 1992.

[9] L. A. Zadeh. Soft computing and fuzzy logic. *IEEE Software*, pages 48–56, Nov. 1994.

[10] L. A. Zadeh. Fuzzy logic and the calculi of fuzzy rules and fuzzy graphs: A precis. *Multi. Val. Logic*, 1:1–38, 1996.