Rule Visualization based on Multi-Dimensional Scaling

Thomas R. Gabriel, Kilian Thiel, and Michael R. Berthold

ALTANA Chair for Bioinformatics and Information Mining
Department of Computer and Information Science
Konstanz University, Box M 712, 78457 Konstanz, Germany
{gabriel,thiel,berthold}@inf.uni-konstanz.de

Abstract—This paper presents an approach to visualize and explore high-dimensional rules in two dimensional views. The proposed method uses multi-dimensional scaling to place the rule centers and then extends the rules’ regions to depict their overlap. This results not only in a visualization of the rules’ distribution but also allows their vicinity neighborhood to be judged. The proposed technique is illustrated and discussed based on a number of well-known benchmark data sets.

I. INTRODUCTION

Extracting rules from data is not a new area of research. In [13] and [16], to name just two examples, algorithms were described that construct hyper-rectangles in feature space. The resulting set of rules encapsulates regions in feature space that contain patterns of the same class. Other approaches, which construct fuzzy rules instead of crisp rules, were presented, for example, in [1], [10], [14] and [15].

What all of these approaches have in common is that they tend to build very complex rule systems for large data sets originating from a complicated underlying system. In addition, high-dimensional feature spaces result in even more complex rules relying on many attributes and increase the number of required rules to cover the solution space. An approach that aims to reduce the number of constraints on each individual rule was recently presented in [3]. The generated fuzzy rules only constrain a few of the available attributes and hence remain readable even in the case of high-dimensional spaces. However, this algorithm also tends to produce many rules for large, complicated data sets.

Multi-dimensional scaling [11] methods are widely used to map high-dimensional data onto lower dimensions. However, models—especially rule models, are much harder to visualize in lower feature spaces. The main challenge is the mapping of the rules’ antecedences, which are usually not only points but hyper-rectangles in the original higher dimensional feature space. One approach [5] that deals with this problem maps fuzzy points using fuzzy distance measure onto two dimensions using the MDS algorithm. Here, the rules’ coverage is displayed by a quadrate in two dimensions giving only a restricted feeling for the original rule-spread in higher dimensions. Another idea applies the information to all features by visualizing fuzzy points as bands in Parallel Coordinates [4], but still—in the case of highly overlapping rules, exploration is hardly possible. Therefore, in order to attain a useful view it is essential that the spread of rules is shown independently and in addition to their location, while preserving the distance between vicinity rules in the original space. The method proposed here makes use of this information, while keeping track of the relationship and organization between neighboring rules. The resulting two dimensional view can then be used for further analysis.

The paper is organized as follows: In the next section we briefly introduce the used fuzzy rule induction algorithm, before we describe the multi-dimensional scaling method and how it is extended to visualize rule systems in the section that follows. We then illustrate the proposed method using the Iris data set and show how larger rule sets can be visualized and explored for a number of real-world data sets.

II. FUZZY RULE INDUCTION

The applied fuzzy rule learning algorithm [3] constructs a set of fuzzy rules from given training data and generates individual fuzzy rules defined by independent membership functions in the feature space. The training algorithm usually only constrains a few attributes, that is, most support regions remain infinite, leaving the rules interpretable even in the case of high-dimensional input spaces. The final set of fuzzy rules can be described as a n-dimensional IF clause as antecedence and one assigned class in THEN rule’s conclusion:

\[
\mathcal{R}_c^k : \text{IF } \mu_{1,c} \wedge \ldots \wedge \mu_{n,c} \text{ THEN class } c
\]

where \(\mathcal{R}_c^k\) represents rule \(j\) for class \(k\). The rule base contains rules for \(c\) classes and \(r_k\) indicates the number of rules for class \(k\) (\(1 \leq j \leq r_k\) and \(1 \leq k \leq c\)). The fuzzy sets \(\mu_{i,j}^k : \mathbb{R} \rightarrow [0,1]\) are defined for every feature \(i\) (\(1 \leq i \leq n\)) and the overall degree of fulfillment of a specific rule for an input pattern \(\mathbf{x} = (x_1, \ldots, x_n)\) can be computed using the fuzzy set operator for conjunction, \(\wedge\)-norm:

\[
\mu_{i,j}^k(\mathbf{x}) = \min_{i=1,\ldots,n} \{\mu_{i,j}^k(x_i)\}. \tag{1}
\]

Note, however, that any algorithm generating rules from data can be used as a basis for the visualization method presented here.
The combined degree of membership for all rules of class \( k \) can be calculated using the fuzzy set operator for disjunction, \( \cap \)-norm:

\[
\mu^k(\vec{x}) = \max_{j=1, \ldots, r_k} \{ \mu_j^k(\vec{x}) \}.
\]  

(2)

From these membership values the predicted class \( k_{\text{best}} \) for an input pattern \( \vec{x} \) is derived then as:

\[
k_{\text{best}}(\vec{x}) = \arg \max_{k=1, \ldots, c} \{ \mu^k(\vec{x}) \}.
\]

(3)

The algorithm uses trapezoid membership functions, which can be described with four parameters \( \langle a_i, b_i, c_i, d_i \rangle \), where \( a_i \) and \( d_i \) define the fuzzy rule’s support-, and \( b_i \) and \( c_i \) its core-region for each attribute \( i \) of the input dimension.

The fuzzy rule induction method is based on an iterative algorithm. During each learning epoch, i.e., presentation of all training patterns, new fuzzy rules are introduced when necessary and existing ones are adjusted whenever a conflict occurs. For each pattern three main steps are executed:

- **Cover**: If a new training pattern lies inside the support-region of an already existing fuzzy rule of the correct class, its core-region is extended to cover the new pattern. In addition, the weight of this rule is incremented.

- **Commit**: If the new pattern is not yet covered, a new fuzzy rule belonging to the corresponding class is created. The new example is assigned to its core-region, whereas the overall rule’s support-region is initialized “infinite”, that is, the new fuzzy rule is unconstrained and covers the entire domain.

- **Shrink**: If a new pattern is incorrectly covered by an existing fuzzy rule of conflicting class, this fuzzy rule’s core- and/or support-region is reduced, so that the conflict with the new pattern is avoided. The underlying heuristic of this step aims to minimize the loss in volume. For details see [8].

The algorithm usually terminates after only a few iterations over the training data. The final set of fuzzy rules can be used to compute a degree of class activation for new input patterns.

In the next section, we show how the underlying multi-dimensional scaling method can be used to map these fuzzy rules’ core regions onto two dimensions.

### III. Multi-Dimensional Scaling

Multi-dimensional scaling is used to map objects of high-dimensional spaces to lower dimensions, usually two or three dimensions. The method tries to preserve the pairwise distances between objects by minimizing an appropriate error function. This dimension reduction enables visualizations of high-dimensional points in lower-dimensional space. One technique to compute this mapping is called Sammons MDS algorithm which finds a spatial representation for each object \( x \) of the high-dimensional space \( \mathbb{R}^L \). The distance between two objects \( x_i, x_j \in \mathbb{R}^L \) of the high- \( (D_{ij}) \) and lower-dimensional \( (d_{ij}) \) space has to be approximated, that is \( \forall i \neq j D_{ij} \approx d_{ij} \) which can be e.g. the Euclidean distance:

\[
D_{ij}^2 = \sum_{q} (x_{i,q} - x_{j,q})^2
\]

(4)

with \( q \in \{1, \ldots, L\} \) and \( i, j \in \{1, \ldots, N\} \). To minimize the error \( E \) called stress function, the differences of the distance values need to be aggregated to:

\[
E(\{x_i\}) = \sum_{i=1}^{N} \sum_{j>i}^{N} w_{ij}(d_{ij} - D_{ij})^2
\]

(5)

where the weight \( w_{ij} \) is used to normalize the stress function \( E \) in order to be independent from the absolute values \( D_{ij} \).

The Sammons algorithm uses a gradient descent-based method to iteratively minimize the stress function \( E \) for each point \( x_i \). The position for each point \( \hat{x}_i \) in the lower-dimensional space is computed using learning rate \( \eta \). The points are initialized randomly or via PCA at the beginning. The update equation for each point \( \hat{x}_i \) then appears as follows:

\[
\hat{x}_i^{(\text{new})} = \hat{x}_i^{(\text{old})} + \eta \Delta_i
\]

(6)
to which the gradient decent-based method is applied for \( \Delta_i \):

\[
\Delta_i = \frac{\partial E}{\partial \hat{x}_{i,q}} = \frac{\partial E^2}{\partial \hat{x}_{i,q}^2}
\]

(7)

A. Mapping the Rule Centers

To map \( n \)-dimensional fuzzy rules of the form

\[
\text{IF } x_1 \text{ IS } \mu_1 \land \cdots \land x_n \text{ IS } \mu_n \text{ THEN class } c
\]

where each dimension is described by a trapezoid membership function with the four parameter \( \langle a, b, c, d \rangle \). The center point \( \vec{p} \) within the core-region \( [b, c] \) is then approximated as the center between the left and right core boundary \( \vec{c} = (c + b)/2 \).

This set of points \( p \) is then used to be positioned onto two dimensions using the MDS method. For each high-dimensional point, a two dimensional point is initialized either randomly, or by its first two principle components. The algorithm then re-positions the points in two dimensions to approximate the distances in the original space. The update value is computed based on the learning rate \( \eta \), \( \Delta_i \), and is repeated for a given number of cycles. The learning rate is usually initialized with 1.0 and decreases after each epoch.

B. Visualizing the Rule-Spread

Since the MDS algorithm only maps the center points of each rule, the next step is to additionally position the spread of the core-region for each rule. This spread can be expressed by the distance between each corner point of the hyper-rectangle to the center of the rule, which is the same for all corner points. This distance can be used to display an approximation of the rules’ spread in the original space, the Delaunay Triangulation (dual structure of the Voronoi Diagram) is used to compute all direct neighbors of each rule. Each rule now has a well-defined number (at least two) of neighbors that span a triangle in the two dimensional space. The distances in the input space between
two neighboring rules are then used to calculate the spread of each rule towards these neighbors. The fraction of this spread compared to the overall distance between the rule centers is calculated in the next step to determine the outer envelope of each rule region. In the special case of rule center points on the convex hull, the area of interest is mirrored to the other side of the hull assuming the same spread of the core regions towards all dimensions. This is necessary to equally model rule coverages to rules in the inner part of the hull and convey a better impression of the actual rule coverage.

Given a rule \( R^A \), its center point \( p^A \) in the lower-dimensional space, and the neighbor’s center point \( p^B \) of rule \( R^B \), a point \( p^A_{\text{core}} \) also in the lower-dimensional space needs to be computed, that is, the spread of the core-region between rule \( R^A \) and \( R^B \). This point \( p^A_{\text{core}} \) is computed using the spread of the core-region in the original space:

\[
p^A_{\text{core}} = p^A + (p^B - p^A) \cdot C^{A,B}_{\text{core}}/D^{A,B}_{\text{core}},
\]

where \( C^{A,B}_{\text{core}} \) represents the spread between the rules’ \( (R^A) \) core-region and the rules’ \( (R^B) \) center in the original space:

\[
(C^{A,B}_{\text{core}})^2 = \sum_{q} (p^A_q - S^A_{q})^2,
\]

where \( P^A_q \) is center point and \( S^A_{q} \) the core boundary of rule \( R^A \) in high-dimensional space. \( D^{A,B}_{\text{core}} \) is the distance between the rules’ center in the original space.

Rules which do not overlap in any dimensions are non-overlapping (2). Obstructing rules may overlap either in one or more dimensions, but not in all dimensions (3). Figure 1 (1) shows two rules that partially overlap in all dimensions, the distance between the core boundary points is used to determine the overlapping values. The same holds for rules which do not overlap in any dimension, see Figure 1 (2).

The spread for obstructing rules is computed based on all non-overlapping dimensions, see Figure 1 (3).

![Fig. 2. Two unbalanced rules do overlap in dimension \( x_1 \), but \( f \) approximates to real spread.](image)

To address only one problem, in cases where the overlap is marginal in one dimension, this dimension is not taken into account when approximating the spread value. Figure 2 illustrates this problem, where \( f \) would be the better approximation, but \( s \) is computed as a spread value due to the small overlap in the \( x_1 \) dimension.

C. Rule Visualization: An Example

Before showing results on real benchmarks we demonstrate the proposed visualization scheme on the well-known Iris data set [7]. The Iris data consists of 150 four-dimensional patterns describing three classes of Iris plants: Iris-setosa (red), Iris-versicolor (green), and Iris-virginica (blue). The four dimensions consist of measurements for the petal and sepal, length and width. One class (Iris-setosa) can be separated from the other two along axes-parallel lines.

The mapping of rules’ center points (with 11 rules) is shown in Figure 3. In addition, for each rule the direct neighbors as determined by the Delaunay Triangulation are connected by lines for each rule. It can already be seen that the red class is covered by only one rule whose center is relatively far away from all others. As expected, for the other two classes a higher number of rules was introduced. Still, this picture does not reveal any information about possible overlaps between rules or even hint at how close they are to each other.

To gain more insights into the rule model, it is desirable to see how each rule’s spread can be displayed in relationship to their direct neighbors. Figure 4 illustrates how the proposed method displays this property for the rules generated for the Iris data. Here, we clearly see which areas are covered by rules and where rules overlap. Again, the red class is separated from the other two. It becomes a lot more obvious from this view, however, how the blue and green classes are modeled using rules that partially overlap and often only cover very small areas. The underlying algorithm had to

![Fig. 1. Shows the three cases of partially overlapping (1), non-overlapping (2), and obstructing rules (3).](image)
Fig. 3. Visualization of the rule set (center points only) generated for the Iris data. The Delaunay Triangulation indicates direct neighbors for each rule.

Fig. 4. Visualization of the Iris rule set, including its spread and possible overlaps.

Fig. 5. Visualization of the rule set (center points only) generated on the Ocean Satellite Image data.

IV. EXPERIMENTAL RESULTS ON REAL-WORLD DATA

Experiments are conducted on two benchmark sets, namely Ocean Satellite Images [17], [19] and Shuttle Control Database from the StatLog–Project [12].

A. Ocean Satellite Images

The first data set stems from a satellite used primarily to examine the ocean. The images are from the Coastal Zone Color Scanner (CZCS) and are of the West Florida shelf. The CZCS was a scanning radiometer aboard the Nimbus-7 satellite, which viewed the ocean in six co-registered spectral bands 443, 520, 550, 670, 750 nm, and a thermal IR band. It operated from 1979-1986.

The features used were the 443, 520, 550, 670 nm bands; the pigment concentration value was derived from the lowest 3 bands. Atmospheric correction was applied to each image [9] before the features were extracted. A fast fuzzy clustering algorithm, mrFCM [6], was applied to obtain 12 clusters per image. There were five regions of interest in each image. These consist of red tide, green river, other phytoplankton blooms, case I (deep) water and case II (shallow) water. Twenty-five images were ground-truthed by oceanographers [18] and eighteen of these were used for training. The eighteen training images were clustered into 12 classes. Each class or cluster was labeled by the ground-truth image as its majority class.

The rules’ center vectors generated from the training images are depicted in Figure 5. These 44 rules are mapped onto two dimensions showing low distortion, e.g. classes red and yellow almost form one coherent cluster. All other classes are distributed across the two dimensional domain.
The mapping of the rules gives more detailed insights into the structure of the rule system by showing larger inter-related areas. For instance, it is clearly visible that a number of rules of class yellow cover bigger areas of the space. Furthermore, a number of smaller rules, almost randomly distributed, only cover one or a small number of instances each of which can be an indication for outliers or artifacts in the data. The larger areas of interest reflect the distribution of the input patterns. These rules can easily be identified and used for further investigation.

B. Shuttle Control Database

This data originated from NASA and concerns the position of radiators within the Space Shuttle. The data was divided into a training set with 43,500 and a test set with 14,500. The shuttle data has 9 attributes which are assigned to three classes\(^2\). This data set can be perfectly described with only 15 rules), see Figure 7 and 8.

In this example the relationships between rules and their organization can be seen clearly. For example, rules for class blue are split due to intermediate instances of conflicting classes. Other rules cover bigger areas and are separated relatively from the other. A number of smaller rules, red and green, can be found in the middle of the plot which again are caused by areas with low evidence in the data, but which may, for some application, be exactly what the user is interested in.

\(^2\)The remaining classes of the original data set only occur less than 1% and were left out for these experiments.

V. CONCLUSIONS

In this paper a methodology was presented to visualize a set of rules built in a high-dimensional feature space. The mapping onto two dimensional space maintains the pairwise distances between the rules as much as possible and additionally displays an approximation of the rules' spread and overlap in the original feature space. The presented methodology not only provides a way to visualize the model
at once but also shows potential for user interaction with the model and hence offers an interesting addition to intelligent data analysis [2] in the area of model exploration.

VI. ACKNOWLEDGMENTS

This work was partially supported by the DFG Research Training Group GK-1042 “Explorative Analysis and Visualization of Large Information Spaces”.

REFERENCES