Outline

1. Motivation

2. Extracting Grid-Based Fuzzy Rules

3. Extracting Individual Fuzzy Rules

4. Rule Generation by Fuzzy Clustering

5. Different Approaches
Extracting Fuzzy Systems from Data

How can fuzzy systems automatically be derived from example data?

- suppose, input space $\mathcal{X}$ and output space $\mathcal{Y}$
- we observe $n$ training patterns $(x_i, y_i) \in S \subseteq \mathcal{X} \times \mathcal{Y}, \ 1 \leq i \leq n$
- given numerical input, $\mathcal{X} = (X_1, \ldots, X_p) \subset \mathbb{R}^p$ and thus $x_i \mapsto x_i$
- fuzzy rule base shall approximate $S$

- classical two-step approach
  1. find fuzzy sets by
     - either predefining them on input and output variables
     - or constructing them throughout learning procedure
  2. find fuzzy rules
     - either directly
     - or iteratively
Outline

1. Motivation

2. Extracting Grid-Based Fuzzy Rules
   - Wang & Mendel Algorithm
   - Higgins & Goodman Algorithm

3. Extracting Individual Fuzzy Rules

4. Rule Generation by Fuzzy Clustering

5. Different Approaches
Fixed Grid-Based Algorithms

• each $X_j$ is partitioned into small set of linguistic variables
• rules use all or subset of possible combinations

⇒ global granulation of $X$ into *tiles*

\[ R_{1,\ldots,1} : \text{if } x_1 \text{ is } \mu_{1,1} \text{ and } \ldots \text{ and } x_p \text{ is } \mu_{1,p} \text{ then } \ldots \]

\[ \ldots \]

\[ R_{l_1,\ldots,l_p} : \text{if } x_1 \text{ is } \mu_{l_1,1} \text{ and } \ldots \text{ and } x_p \text{ is } \mu_{l_p,p} \text{ then } \ldots \]

• $l_j$ ($1 \leq j \leq p$): number of linguistic values for $X_j$

• problems
  • exponentially many rules in high-dimensional spaces
  • fine grid causes very high computational costs
  • wrong choice of grid may skip extrema

R. Kruse, C. Moewes
Fuzzy Systems – Fuzzy Rule Generation
2010/01/27 5 / 61
Wang & Mendel Algorithm

- basic fuzzy rule learning method
  1. predefine global grid for $X$ and $Y$
  2. determine best possible output fuzzy set for each rule

- one run through $S$ determines closest $x$ to geometrical rule center
- closest output fuzzy value $\mapsto$ corresponding fuzzy rule

- problems if
  1. function extrema far from away center point
  2. number of rules gets huge
1. Granulate $\mathcal{X}$ and $\mathcal{Y}$

- divide each $X_j$ into $l_j$ equidistant triangular fuzzy sets
- similarly, $\mathcal{Y}$ is granulated into $l_y$ triangular fuzzy sets
2. Determine best consequence for each rule

- for each \((x, y) = (x_1, \ldots, x_p, y) \in S\)
  compute membership degree to each possible tile

\[
\min \left\{ \mu_{k_1,1}(x_1), \ldots, \mu_{k_p,p}(x_p), \mu_{k_y}(y) \right\}
\]

with \(1 \leq k_j \leq l_j\) and \(1 \leq k_y \leq l_y\)

- \(\mu_{k_j,j}\) = membership function of \(k_j\)-th linguistic value of \(X_j\)

- \(\forall k_1, \ldots, k_p\) tile arg max \(1 \leq k_y \leq l_y\) \(\mu(k_1, \ldots, k_p, k_y)\) is 1 rule

\(R(k_1, \ldots, k_p) : \text{if } x_1 \text{ is } \mu_{k_1,1} \text{ and } \ldots \text{ and } x_p \text{ is } \mu_{k_p,p} \text{ then } y \text{ is } \mu_{k_y}\)

- membership degree is used as rule weight \(\beta(k_1, \ldots, k_p)\)
Determining $\hat{y}$ based on new $x$

- given arbitrary $x$, rules determine crisp output $\hat{y}$

1. for each rule compute degree of fulfillment

$$\mu_{(k_1,\ldots,k_p)}(x) = \min \{\mu_{k_1,1}(x_1), \ldots, \mu_{k_p,p}(x_p)\}$$

2. compute $\hat{y}$ by some kind of COG defuzzification

$$\hat{y} = \sum_{k_1=1,\ldots,k_p=1}^{l_1,\ldots,l_p} \frac{\beta_{(k_1,\ldots,k_p)} \cdot \mu_{(k_1,\ldots,k_p)}(x) \cdot \bar{y}_{(k_1,\ldots,k_p)}}{\beta_{(k_1,\ldots,k_p)} \cdot \mu_{(k_1,\ldots,k_p)}(x)}$$

- $\bar{y}_{(k_1,\ldots,k_p)} = \text{center of output region of } R_{(k_1,\ldots,k_p)}$
Example: Wang & Mendel Algorithm

- example data set with one input and one output
- note that closest points to corresponding rules are red
Example: Wang & Mendel Algorithm (cont.)

step 2: generate rules

resulting crisp approximation

- fuzzy rules are shown by their $\alpha = 0.5$-cuts
- learned model misses extrema far away from rule centers
Example: Wang & Mendel Algorithm (cont.)

• generated rule base:

\[
R_1 : \text{if } x \text{ is zero}_x \text{ then } y \text{ is medium}_y \\
R_2 : \text{if } x \text{ is small}_x \text{ then } y \text{ is medium}_y \\
R_3 : \text{if } x \text{ is medium}_x \text{ then } y \text{ is large}_y \\
R_4 : \text{if } x \text{ is large}_x \text{ then } y \text{ is medium}_y
\]

• intuitively, rule \( R_2 \) should probably be used to describe minimum

\[
R_2' : \text{if } x \text{ is small}_x \text{ then } y \text{ is small}_y
\]
Summary

• one pattern per rule is used to compute rule’s outcome

• high variance would lead to model failures

• predefined fixed grid yields to fuzzy model which
  • either does not fit underlying function very well
  • or consists of large number of rules

⇒ wish to automatically determine granulations of $\mathcal{X}$ and $\mathcal{Y}$
Higgins & Goodman Algorithm

- extension of Wang & Mendel algorithm

1. only one membership function is used for each $X_j$ and $Y$
   ⇒ one large rule covering entire feature space

2. new membership functions are placed at points of maximum error

- both steps are repeated until
  - maximum number of divisions is reached or
  - approximation error remains below threshold
1. Initialization

- create membership function for each $X_j$ covering entire domain range
- create membership function for $Y$ at corner points of $X$
- at corner point, each $X_j$ is maximal or minimal of its domain range
- for each corner point, closest example from $S$ is used to add membership function at its output value
2. Adding new Membership Functions

- find point within $S$ with maximum error
- defuzzification equals Wang & Mendel
- for each $X_j$, add new membership function at corresponding value of “maximal error point”
  \[ \Rightarrow \] perfectly described by model
3. Create new Cell-based Rule Set

- new rules: associate output membership functions with newly created cells
  
  ⇒ take closest point to all membership functions of $\mathcal{X}$ (equals Wang & Mendel)

- associated output membership function is closest one to output value of “closest point”

- if output value of “closest point” is far away, new output function is created
4. Termination Detection

- if error is below threshold (or if certain number of iterations have been done), then stop algorithm
- otherwise continue at step 2
Summary

• this approach can model extrema better than Wang & Mendel

• it favors extrema

⇒ strong tendency to outliers

• data driven granulation: difficult to interpret

• greedy algorithm: grid is often suboptimal

• it is possible to simplify learned rule base, e.g.,

  by rule ranking and search for best rules
Outline

1. Motivation

2. Extracting Grid-Based Fuzzy Rules

3. Extracting Individual Fuzzy Rules
   Individual Membership Functions
   Berthold & Huber Algorithm

4. Rule Generation by Fuzzy Clustering

5. Different Approaches
Extensions avoiding Global Grids

- in high-dimensional $\mathcal{X}$, global granulation leads to many rules

$\Rightarrow$ now, no global dependence on granulation
- individual membership functions for each rule
- better modeling of local properties

$R_1: \quad \text{if } x_1 \text{ is } \mu_{1,1} \text{ and } \ldots \text{ and } x_p \text{ is } \mu_{1,p} \text{ then } \ldots$

$\ldots$

$R_r: \quad \text{if } x_1 \text{ is } \mu_{r,1} \text{ and } \ldots \text{ and } x_p \text{ is } \mu_{r,p} \text{ then } \ldots$

- not all attributes will be used for all rules
  - individual choice of constraints on few attributes per rule
  - better interpretability in high dimensions
  - no exponential number of rules with increasing dimensionality
Local Granulation

- 3 rules in 2 dimensions
- compare with global granulation à la Wang & Mendel
- possible disadvantage
  - potential loss of interpretation
  - projection of all fuzzy sets onto one $X_j$ is usually not meaningful
Berthold & Huber Algorithm

- constructs rule base with individual fuzzy sets per rule
- parameters that must be specified
  - granulation of $\mathcal{Y}$, i.e., number and shape of membership functions
  - $c$ fuzzy sets defined by $\mu^k_y$ with $1 \leq k \leq c$

- algorithm iterates over $S$ and fine-tunes evolving model
- final rule base consists of fuzzy rules $R^k_d$, $1 \leq d \leq r_k$
- $r_k =$ number of rules for output region $k$
- output for $k$–th region and some $x$ equals Mamdani controller

$$\mu^k(x) = \max_{1 \leq d \leq r_k} \left\{ \min_{1 \leq j \leq p} \{ \mu^k_{d,j}(x_j) \} \right\}$$
Form of Rules

Rules

- all rules rely on trapezoidal membership functions

⇒ each rule can be described by 4 parameters per $X_j$
  
  if $x_1$ is $\langle a_1, b_1, c_1, d_1 \rangle$ and ... and $x_p$ is $\langle a_p, b_p, c_p, d_p \rangle$
  
  then $y$ is $\mu_y^k$

- however, if some trapezoids cover entire domain of an $X_j$,

  then rule’s degree of fulfillment is independent from of this $X_j$
Algorithm 1 Berthold & Huber

Input: $S = \{(x_i, c_i) \mid 1 \leq i \leq n\}$ with $x \in \mathbb{R}^p$ and $c \in \mathbb{N}$ class of $x$
do { for each training example $(x, c) \in S$ { if correct rule of class $c$ exists { increase weight by one // COVER adjust core region of rule to cover $x$ } else { insert new rule with core equals $x$ // COMMIT support equals $\infty$ (i.e., rule is not constrained) } reduce support of all rules of conflicting class that cover $x$ // SHRINK } }

- COVER and COMMIT are easy to implement
- SHRINK is based on heuristics (e.g., volume-based)
Some Remarks on Core and Support Regions

- algorithm finds rule base that completely describes data

- each rule is partial hypothesis for subset $\bar{S} \subset S$
  - core = most specific hypothesis covering $\bar{S}$
  - support = (one of the) most general hypotheses covering $\bar{S}$
  \[ \Rightarrow \text{ support is more general than core} \]

- both core and support regions can be seen as
  - smallest area with highest degree of confidence (evidence)
  - largest area without conflict (no counter-example)
Example: Berthold & Huber Algorithm

- given two-dimensional $\mathcal{X}$ and training data $\mathcal{S}$ where $|\mathcal{Y}| = 2$

- task: fuzzy binary classification

- first, start with empty rule base for each region/class

- Java applet is available that demonstrates algorithm
Example: Berthold & Huber Algorithm (cont.)

- insert general rule for first example pattern
Example: Berthold & Huber Algorithm (cont.)

- suppose that 2nd pattern is from different class

⇒ new rule is inserted for 2nd pattern

- also adjust 1st (conflicting) rule
Example: Berthold & Huber Algorithm (cont.)

• suppose that 3rd pattern is from same class as 2nd one

⇒ adjust free feature to avoid conflict with 3rd pattern

• and so on...
Choosing the Right Feature to Shrink

- in $p$-dimensional feature space, there are $p$ choices
- algorithm uses several heuristics:
  1. maximize remaining volume $\Rightarrow$ low rule numbers, good coverage
  2. minimize number of constrained attributes $\Rightarrow$ feature reduction
  3. minimize number of constraints on free features $\Rightarrow$ interpretability
  4. use information theoretic measures $\Rightarrow$ feature importance
Outline

1. Motivation

2. Extracting Grid-Based Fuzzy Rules

3. Extracting Individual Fuzzy Rules

4. Rule Generation by Fuzzy Clustering
   - Extend Membership Values to Continuous Membership Functions
   - Example: The Iris Data
   - Information Loss from Projection
   - Example: Transfer Passenger Analysis

5. Different Approaches
Rule Generation by Fuzzy Clustering

1. apply fuzzy clustering to $\mathcal{X} \Rightarrow$ fuzzy partition matrix $U = [u_{ij}]$

2. use obtained $U = [u_{ij}]$ to define membership functions

• usually $\mathcal{X}$ is multidimensional

⇒ How to specify meaningful labels for multidim. membership functions?
Extend $u_{ij}$ to Continuous Membership Functions

- assigning labels for one-dimensional domains is easier $\Rightarrow$
  
  1. project $U$ down to $X_1, \ldots, X_p$ axis, respectively
  2. only consider upper envelope of membership degrees
  3. linear interpolate membership values $\Rightarrow$ membership functions
  4. cylindrically extend membership functions

- original clusters are interpreted as conjunction of cyl. extensions
  
  - e.g., cylindrical extensions “$x_1$ is low”, “$x_2$ is high”
    $\Rightarrow$ multidimensional cluster label “$x_1$ is low and $x_2$ is high”

- labeled clusters = classes characterized by labels

- every cluster = one fuzzy rule
Convex Completion [Höppner et al., 1999]

• problem of this approach: non-convex fuzzy sets

⇒ having upper envelope, compute convex completion

• we denote $p_1, \ldots, p_n$, $p_1 \leq \ldots, p_k$ as ordered projections of $x_1, \ldots, x_n$ and $\mu_{i1}, \ldots, \mu_{in}$ as respective membership values

• eliminate each point $(p_t, \mu_{it})$, $t = 1, \ldots, n$, for which two limit indices $t_l, t_r = 1, \ldots, n$, $t_l < t < t_r$, exist s.t.

\[
\mu_{it} < \min\{\mu_{it_l}, \mu_{it_r}\}
\]

• after that: apply linear interpolation of remaining points
Example: The Iris Data

© Iris Species Database http://www.badbear.com/signa/

- Iris setosa
- Iris versicolor
- Iris virginica

- collected by Ronald Aylmer Fischer (famous statistician)
- 150 cases in total, 50 cases per Iris flower type
- measurements: sepal length/width, petal length/width (in cm)
- most famous dataset in pattern recognition and data analysis
Example: The Iris Data

- shown: sepal length and petal length
- Iris setosa (red), Iris versicolor (green), Iris virginica (blue)
1. Membership Degrees from FCM

- raw, unmodified membership degrees
2. Upper Envelope

- for every attribute value and cluster center, only consider maximum membership degree

![Graph showing upper envelopes for sepal length and pedal length.](image-url)
• convex completion removes spikes [Höppner et al., 1999]
4. Linear Interpolation

- interpolation for missing values (needed for normalization)
5. Stretched and Normalized Fuzzy Sets

- every $\mu_i(x_j) \mapsto \mu_i(x_j)^5$ (extends core and support)
- normalization has been performed finally
Information Loss from Projection

- rule derived from fuzzy cluster represents approximation of cluster
- information gets lost by projection
  - cluster shape of FCM is spherical
  - cluster projection leads to hypercube
  - hypercube contains hypersphere
- loss of information can be kept small using axes-parallel clusters
Example: Transfer Passenger Analysis
[Keller and Kruse, 2002]

- German Aerospace Center (DLR) developed macroscopic passenger flow model for simulating passenger movements on airport’s land side

- for passenger movements in terminal areas: distribution functions are used today

- goal: build fuzzy rule base describing transfer passenger amount between aircrafts

- these rules can be used to improve macroscopic simulation

- idea: find rules based on probabilistic fuzzy c-means (FCM)
Attributes for Passenger Analysis

- maximal amount of passengers in certain aircraft (depending on type of aircraft)
- distance between airport of departure and airport of destination (in three categories: short-, medium-, and long-haul)
- time of departure
- percentage of transfer passengers in aircraft
General Clustering Procedure

- Identification of outliers
- Scale adaption
- Clustering technique
- Number of clusters or validity measure
- Similarity measure

Preparation:
- Preprocessing
- Parameter selection
- Initialization

Calculation:
- Sufficient classification?
- Calculation of prototypes
- Calculation of membership degrees

Evaluation:
- Extraction of fuzzy rules

No

Yes
Distance Measure

- distance between $\mathbf{x} = (x_1, x_2)$ and $\mathbf{c} = (0, 0)$

$$d^2(\mathbf{c}, \mathbf{x}) = \| \mathbf{c} - \mathbf{x} \|^2$$

$$d^2_T(\mathbf{c}, \mathbf{x}) = \frac{1}{\tau^p} \| \mathbf{c} - \mathbf{x} \|^2$$
Distance Measure with Size Adaption

\[ d_{ij}^2 = \frac{1}{\tau_i^p} \cdot \|c_i - x_j\|^2 \]

\[ c_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m} \]

\[ \tau_i = \frac{\left( \sum_{j=1}^{n} u_{ij}^m d_{ij}^2 \right)^{\frac{1}{p+1}}}{\sum_{k=1}^{c} \left( \sum_{j=1}^{n} u_{kj}^m d_{kj}^2 \right)^{\frac{1}{p+1}}} \cdot \tau \]

\[ \tau = \sum_{i=1}^{c} \tau_i \]

- \( p \) determines emphasis put on size adaption during clustering
Constraints for the Objective function

- probabilistic clustering
- noise clustering
- influence of outliers
Probabilistic and Noise Clustering

![Diagram of membership degrees with data and prototypes marked with circles and crosses. The membership degrees range from 0.40 to 0.90.]

- Data points are marked with crosses.
- Prototypes are marked with circles.
- Membership degrees are indicated by different line styles and values.
Influence of Outliers

• A weighting factor $\omega_j$ is attached to each datum $x_j$

• weighting factors are adapted during clustering

• using concept of weighting factors:
  • outliers in data set can be identified and
  • outliers’ influence on partition is reduced
Membership Degrees and Weighting Factors

![Graphs showing membership degrees and weights with data and prototypes marked]
Influence of Outliers

- minimize objective function

\[
J(X, U, C) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m \cdot \frac{1}{\omega_j^q} \cdot d_{ij}^2
\]

subject to

\[
\forall j \in [n] : \sum_{i=1}^{c} u_{ij} = 1, \quad \forall i \in [c] : \sum_{j=1}^{n} u_{ij} > 0, \quad \sum_{j=1}^{n} \omega_j = \omega
\]

- \( q \) determines emphasis put on weight adaption during clustering
- update equations for memberships and weights, resp.

\[
u_{ij} = \frac{d_{ij}^{2(1-m)}}{\sum_{k=1}^{c} d_{kj}^{2(1-m)}}, \quad \omega_j = \frac{\left( \sum_{i=1}^{c} u_{ij}^m d_{ij}^2 \right)^{\frac{1}{q+1}}}{\sum_{k=1}^{n} \left( \sum_{i=1}^{c} u_{ik}^m d_{ik}^2 \right)^{\frac{1}{q+1}}} \cdot \omega
\]
Determining the Number of Clusters

- here, validity measures evaluating whole partition of data
  - global validity measures

- clustering is run for varying number of clusters

- validity of resulting partitions is compared
Fuzzy Rules and Induced Vague Areas

- intensity of color indicates firing strength of specific rule
- vague areas = fuzzy clusters where color intensity indicates membership degree
- tips of fuzzy partitions in single domains = projections of multidimensional cluster centers
Simplification of Fuzzy Rules

- similar fuzzy sets are combined to one fuzzy set
- fuzzy sets similar to universal fuzzy set are removed
- rules with same input sets are
  - combined if they also have same output set(s) or
  - otherwise removed from rule set
Results

- FCM with $c = 18$, outlier and size adaptation, Euclidean distance:

resulting fuzzy sets

simplified fuzzy sets
## Evaluation of the Rule Base

<table>
<thead>
<tr>
<th>rule</th>
<th>max. no. of pax</th>
<th>De st.</th>
<th>depart.</th>
<th>% transfer pax</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>paxmax1</td>
<td>R1</td>
<td>time1</td>
<td>tpax1</td>
</tr>
<tr>
<td>2</td>
<td>paxmax2</td>
<td>R1</td>
<td>time2</td>
<td>tpax2</td>
</tr>
<tr>
<td>3</td>
<td>paxmax3</td>
<td>R1</td>
<td>time3</td>
<td>tpax3</td>
</tr>
<tr>
<td>4</td>
<td>paxmax4</td>
<td>R1</td>
<td>time4</td>
<td>tpax4</td>
</tr>
<tr>
<td>5</td>
<td>paxmax5</td>
<td>R5</td>
<td>time1</td>
<td>tpax5</td>
</tr>
</tbody>
</table>

- **rules 1 and 5**: aircraft with relatively small amount of maximal passengers (80-200), short- to medium-haul destination, and departing late at night usually have high amount of transfer passengers (80-90%).

- **rule 2**: flights with medium-haul destination and small aircraft (about 150 passengers), starting about noon, carry relatively high amount of transfer passengers (ca. 70%).
Outline

1. Motivation

2. Extracting Grid-Based Fuzzy Rules

3. Extracting Individual Fuzzy Rules

4. Rule Generation by Fuzzy Clustering

5. Different Approaches
Different Approaches

- **constructive**: find fuzzy rules by growing singletons
- **hierarchical**: merge grid cells if no points are covered or same class is predicted
- **adaptive**:
  - initialize rules randomly (*e.g.*, with expert knowledge) and iteratively optimize rule parameters (*e.g.*, location, number of fuzzy sets)
  - based on, *e.g.*, gradient descent, neural networks, ... 
- **evolutionary**: find rules by mutation/crossover over generations
- **neuro-fuzzy**: inject fuzzy rules into ANN, use its learning algorithm
Literature about Fuzzy Rule Generation


