Fuzzy Systems

Fuzzy Rule Generation

Prof. Dr. Rudolf Kruse    Christian Moewes

{kruse,cmoewes}@iws.cs.uni-magdeburg.de

Working Group on Computational Intelligence
Department of Knowledge and Language Processing
School of Computer Science
Otto-von-Guericke University of Magdeburg
Outline of Today’s Lecture

1. Motivation

2. Extracting Grid-Based Fuzzy Rules

3. Extracting Individual Fuzzy Rules

4. Rule Generation by Fuzzy Clustering

5. Different Approaches
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 $(x_i, y_i) \in S \subseteq \mathcal{X} \times \mathcal{Y}$ where $1 \leq i \leq n$
- 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 of Today’s Lecture

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 input variable is partitioned into small set of linguistic variables
- resulting rule base uses all or subset of possible combinations of linguistic values

⇒ global granulation of $\mathcal{X}$ into tiles
  
  $R_{1,\ldots,1} : \text{ if } x_1 \text{ is } \mu_{1,1} \text{ and } \ldots \text{ and } x_p \text{ is } \mu_{1,p} \text{ then } \ldots$
  
  $\ldots$

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

- $l_j (1 \leq j \leq p)$ indicates number of linguistic values for $j$-th variable

- problems
  - exponentially many rules in high-dimensional spaces
  - if grid is chosen fine enough, arbitrarily good approximation (but at very high computational costs)
  - wrong choice of grid may skip extrema
• basic fuzzy rule learning method for function approximation
• based on predefined global grid for both $X$ and $Y$
• after definition of grid, it determines best possible output fuzzy set for each rule, i.e., each grid position in the input space

• one run through entire data set determines closest example to geometrical center of each rule
• closest output fuzzy value is assigned to corresponding fuzzy

• algorithm encounters problems
  1. when function extrema lie far from center point of grid positions
  2. and it generates huge number of rules for reasonable accuracy
Wang & Mendel Algorithm

1. Granulate $X$ and $Y$

- divide each variable $X_j$ into $l_j$ equidistant triangular fuzzy sets
- similarly, $Y$ is granulated into $l_Y$ triangular fuzzy sets
Wang & Mendel Algorithm

2. Determine best consequence for each rule

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

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

- \(1 \leq k_j \leq l_j\) and \(1 \leq k_y \leq l_y\)
- \(\mu_{k_j,j}\) indicates membership function of \(k_j\)-th linguistic value of \(X_j\)
- tile \((k_1, \ldots, k_p, k_y)\) with max. membership degree generates one rule

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

- degree of membership will be assigned to each rule as weight \(\beta_{(k_1,\ldots,k_p)}\)
Wang & Mendel Algorithm

Determining $\hat{y}$ based on new $x$

- given an input $x$ ...
- ... learned rule base can be used to compute crisp output $\hat{y}$
- first, degree of fulfillment for each rule is computed

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

- then, output $\hat{y}$ is computed by some kind of COG defuzzification

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

- $\bar{y}(k_1,\ldots,k_p)$ denotes center of output region of corresponding $R(k_1,\ldots,k_p)$
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.)**

- 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

- intuitively, rule $R_2$ should probably be used to describe minimum

$$R'_2 : \text{if } x \text{ is small}_x \quad \text{then } y \text{ is small}_y$$
Wang & Mendel Algorithm

Summary

• only one pattern per rule is used to compute this rule’s outcome
• if function has high variance, resulting rule base will fail to model system’s behavior

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

• therefore, we are interested in approaches that fine-tune or even automatically determine granulations of both input and output variables
Higgins & Goodman Algorithm

Overview

- approach that builds upon Wang & Mendel’s fixed-grid algorithm

- initially, only one membership function is used to describe each input variable and output variable

  ⇒ one large rule covering entire feature space

- then, new membership functions are introduced at points of maximum error

- this is repeated until
  - maximum number of divisions is reached or
  - approximation error remains below certain threshold
Higgins & Goodman Algorithm

1. Initialization

- create two membership functions for each $X_j$ at their extremal data domain range positions
- create membership function for $Y$ at corner points of input space
- at corner point, each $X_j$ is at maximum or minimum of its domain range
- for each corner point, closest example from $S$ is used to add membership function at its output value
Higgins & Goodman Algorithm

2. Add new membership functions into input space

- find point within $S$ with maximum error according to currently predicted output value
- defuzzification for output is done same way as for Wang & Mendel algorithm
- for each input variable, add new membership function at corresponding value of "maximal error point"

$\Rightarrow$ this point is described perfectly by generated model
Higgins & Goodman Algorithm

3. Create new cell-based rule set and insert new output fuzzy sets

- constructing new rules is done by associating output membership functions with newly created cells

⇒ so, taking point which is closest to all membership functions of \( \mathcal{X} \) (as done in Wang & Mendel)

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

- if output value of this “closest point” is too far away from closest membership function, new output function is created
Higgins & Goodman Algorithm

4. Termination detection

- if detected error is below given threshold (or if certain number of iterations have been done)...
- ... then stop algorithm
- otherwise continue at step number 2
• obviously, this approach is able to model extrema much better than Wang & Mendel algorithm

• however, it has definite preference to favor extrema

⇒ strong tendency to concentrate on outliers

• interpretation is difficult since granulation is solely data driven

• grid is often suboptimal due to greedy algorithm

• it also contains method to simplify learned rule base

• done by rule ranking system and afterwards searching for best rules
Outline of Today’s Lecture

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
Extracting Individual Fuzzy Rules
Extensions avoiding Global Grids

• in high-dimensional $\mathcal{X}$, global granulation leads to big number of rules
  ⇒ now, no global dependence on granulation
  • individual membership functions for each rule
  • better modeling of local properties

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

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

• 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 growth of number of rules with increasing dimensionality
Local Granulation

Individual membership functions for each rule

- example for three rules in two dimensions
- compare with global granulation à la Wang & Mendel
- possible disadvantage of individual fuzzy sets
  - potential loss of interpretation
  - projecting all fuzzy sets onto one variable will usually not lead to meaningful linguistic values
Berthold & Huber Algorithm

Overview

- algorithm that 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 of \( \mathcal{Y} \)
  - \( c \) fuzzy sets are defined by \( \mu^k_y \) whereas \( 1 \leq k \leq c \)

- algorithm iterates over examples \( S \) and fine-tunes evolving model
- resulting rule base consists of fuzzy rules \( R^k_d, 1 \leq d \leq r_k \)
- \( r_k \) represents number of rules for output region \( k \)
- output for \( k \)-th region and new \( x \) equals Mamdani controller output

\[
\mu^k(x) = \max_{1 \leq d \leq r_k} \left\{ \min_{1 \leq j \leq p} \{ \mu^k_{d,j}(x_j) \} \right\}
\]
Berthold & Huber Algorithm

Rules

- all rules rely on trapezoidal membership functions

⇒ each rule can be described by four parameters per dimension

\[
\text{if } x_1 \text{ is } \langle a_1, b_1, c_1, d_1 \rangle \text{ and } \ldots \text{ and } x_p \text{ is } \langle a_p, b_p, c_p, d_p \rangle \\
\text{then } y \text{ is } \mu^k_y
\]

- however, if some trapezoids cover entire domain of an $X_j$...
- ... then rule's degree of fulfillment is independent from of this $X_j$
Berthold & Huber Algorithm

Procedure

0. given $S = \{(x_i, c_i) \mid 1 \leq i \leq n\}$ with $x \in \mathbb{R}^p$ and $c \in \mathbb{N}$ class of $x$
1. for each training example $(x, c) \in S$ do
   1.1 if correct rule of class $c$ exists, then COVER
      • increase weight by one
      • adjust core region of rule to cover $x$
   else COMMIT
      • insert new rule with core equals $x$
      • support equals $\infty$ (i.e., rule is not constrained)
   1.2 SHRINK
      • reduce support of all rules of conflicting class that cover $x$
2. repeat last step until no more changes occur

• COVER and COMMIT are easy to implement
• SHRINK is based on heuristics (e.g., volume-based)
• algorithm finds rule base that completely describes data (as long as $S$ is conflict-free)

• each rule is partial hypothesis for subset of $S$
  • core = most specific hypothesis covering subset of $S$
  • support = (one of the) most general hypotheses covering subset of $S$
  $\Rightarrow$ support is more general than core

• both core and support regions can be seen as
  • smallest area with highest degree of confidence (we have evidence)
  • largest area without conflict (we haven’t seen any counter-example)
Example: Berthold & Huber Algorithm

- given two-dimensional \( \mathcal{X} \) and training data \( \mathcal{S} \) where \(|\mathcal{Y}| = 2\)
- task is binary fuzzy classification
- first, start with empty rule base for each region/class
- Java applet is available that demonstrates algorithm
• insert general rule for first example pattern
• suppose, second pattern from different class

⇒ insert new rule for second example pattern

• also adjust first (conflicting) rule
• suppose, third pattern from same class as second one

⇒ adjust free feature to avoid conflict with third pattern

• and so on...
• in $p$-dimensional feature space, there are $p$ choices

• algorithm uses heuristics
  • maximize remaining volume
    ⇒ low number of rules, good coverage
  • minimize number of constrained attributes
    ⇒ feature reduction
  • minimize number of constraints on free features
    ⇒ interpretability
  • use information theoretic measures
    ⇒ generalization, feature importance
Outline of Today’s Lecture

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: Traffic Jam Detection System
   
   Information Loss from Projection

5. Different Approaches
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

$\Rightarrow$ how to specify meaningful labels for multidimensional membership functions?
Extend $u_{ij}$ to Continuous Membership Functions

• assigning labels for one-dimensional domains is easier, thus
  1. project $U$ down to $X_1, \ldots, X_p$ axis, respectively
  2. linear interpolate membership values to obtain membership functions
  3. cylindrically extend membership functions to multidimensional arguments

• with labels for all attributes, original clusters can be interpreted as conjunction of cylindrical 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 can be represented as classes characterized by labels
• every cluster thus symbolized one fuzzy rule
Example: Traffic Jam Detection System

- an induction loop in the road is used as metal detector
- serves as sensor for cars moving across or standing on it
- each passing car induces pulse from sensor
- frequency $f$ of pulse equals number of cars per minute
- pulse width is inversely proportional to current velocity $v$ of car

- after clustering
  - determine clusters for both classes, i.e., traffic jam and no traffic jam
  - output values of classifier are $\mu_r(f, v)$ with $1 \leq r \leq c$

  \[
  \begin{align*}
  \text{if } (f, v) \text{ is } &\mu_1 \quad \text{then “traffic jam”} \\
  \text{if } f \text{ is } &\mu_1^{(f)} \text{ and } v \text{ is } \mu_1^{(v)} \quad \text{then “traffic jam”}
  \end{align*}
  \]
Information Loss from Projection

- rule derived from fuzzy cluster represents approximation of cluster
- information gets lost by projection
  - ideal cluster shape of fuzzy \( c \)-means (FCM) is spherical
  - projecting an FCM cluster leads to hypercube that contains hypersphere
- loss of information can be kept small using axes-parallel clusters
Outline of Today’s Lecture

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 (to be discussed)**
  - find good rules by mutation/crossover over many generations

- **neuro-fuzzy (to be discussed)**
  - inject fuzzy rules into neural network and use its learning algorithm
Literature about Fuzzy Rule Generation

*Intelligent Data Analysis: An Introduction.*

Vieweg, Wiesbaden, Germany, 3rd edition.

*Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition.*
John Wiley & Sons Ltd, New York, NY, USA.