

Li Fanzhang, Zhang Li, Zhang Zhao
Dynamic Fuzzy Machine Learning

Also of Interest



Lie Group Machine Learning

F. Li, 2019

ISBN 978-3-11-050068-4, e-ISBN (PDF) 978-3-11-049950-6,
e-ISBN (EPUB) 978-3-11-049807-3, Set-ISBN 978-3-11-049955-1



Cloud Computing Architecture

J. Gu, 2018

ISBN 978-3-11-053784-0, e-ISBN (PDF) 978-3-11-054067-3,
e-ISBN (EPUB) 978-3-11-053998-1, Set-ISBN 978-3-11-054068-0



Trusted Computing

D. Feng, 2017

ISBN 978-3-11-047604-0, e-ISBN (PDF) 978-3-11-047759-7,
e-ISBN (EPUB) 978-3-11-047609-5, Set-ISBN 978-3-11-047760-3



Chaotic Secure Communication

K. Sun, 2016

ISBN 978-3-11-042688-5, e-ISBN (PDF) 978-3-11-043406-4,
e-ISBN (EPUB) 978-3-11-043326-5, Set-ISBN 978-3-11-043407-1

Li Fanzhang, Zhang Li, Zhang Zhao

Dynamic Fuzzy Machine Learning

DE GRUYTER

Author

Prof. Fan Zhang Li
Soochow University
School of Computer Science
and Technology
No. 1 Shizi Road
215006 Suzhou, China
lfzh@suda.edu.cn

ISBN 978-3-11-051870-2
e-ISBN (PDF) 978-3-11-052065-1
e-ISBN (EPUB) 978-3-11-051875-7
Set-ISBN 978-3-11-052066-8

Library of Congress Cataloging-in-Publication Data

A CIP catalog record for this book has been applied for at the Library of Congress.

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie;
detailed bibliographic data are available on the Internet at <http://dnb.dnb.de>.

© 2018 Walter de Gruyter GmbH, Berlin/Boston

Typesetting: Compuscript Ltd., Shannon, Ireland

Printing and binding: CPI books GmbH, Leck

Cover image: JIRAROJ PRADITCHAROENKUL / iStock / Getty Images

♾ Printed on acid-free paper

Printed in Germany

www.degruyter.com

Preface

As an important method of data analysis, machine learning technology has been widely used in academic and industrial circles. In particular, the development of cloud computing, logistics networks, big data, and quantum information has played an unprecedented role in promoting the globalization of machine learning. In modern data analysis, studying the technical level of computing may limit the whole process of scientific research. As an analogy, analyzing the physical properties of a material or the physical properties of molecular structures is quite hard to achieve through calculations alone. The fundamental reason is the deep relationship between these data structures and the semantic uncertainty, which makes it difficult to complete certain tasks using only computing technology.

Dynamic fuzzy data (DFD) is one of the most difficult data types in the field of big data, cloud computing, the Internet of Things, and quantum information. To investigate the deep structure of relations and the data semantic uncertainty of DFD, our research group has been studying this field since 1994, and we have proposed various set, logic, and model system theories for uncertain datasets. To effectively handle DFD, dynamic fuzzy sets, and dynamic fuzzy logic (DFL) are introduced into the machine learning field, and the dynamic fuzzy machine learning theory framework is proposed.

Our work has been published in international journals and presented at international conferences. The first draft of "*Dynamic Fuzzy Machine Learning*" has been taught to master's and doctoral students at Soochow University as an independent 54-hour course. Based on this, several revisions have been made. To meet the requirements of the readers of this book, the course is published here over a total of seven chapters.

In the first chapter, the dynamic fuzzy machine learning model is discussed. This chapter is divided into six sections. In the first section, we define the problem. In the second section, we introduce the dynamic fuzzy machine learning model. In the third section, we study some related work. The fourth section presents the algorithm of the dynamic fuzzy machine learning system and the related process control model. In the fifth section, we introduce the dynamic fuzzy relational learning algorithm. The sixth section summarizes the chapter.

The second chapter describes the dynamic fuzzy autonomous learning subspace learning algorithm. This chapter is divided into four sections. In the first section, we analyse the current state of autonomous learning. In the second section, we present an autonomous learning subspace theoretical system based on DFL. In the third section, we introduce the autonomic subspace learning algorithm based on DFL. In the fourth section, we summarize this chapter.

The third chapter is devoted to fuzzy decision tree learning. This chapter is divided into six sections. In the first section, we examine the current state of decision tree

learning. In the second section, we study the dynamic fuzzy lattice decision tree method. In the third section, we discuss special attribute processing for dynamic fuzzy decision trees. In the fourth section, we study the pruning strategy for dynamic fuzzy decision trees. In the fifth section, we describe some applications of dynamic fuzzy decision trees. The sixth section summarizes this chapter.

In the fourth chapter, we consider dynamic concepts based on dynamic fuzzy sets. This chapter is divided into seven sections. In the first section, we analyse the relation between dynamic fuzzy sets (DFS) and concept learning. In the second section, we introduce the model of DF concept representation. In the third section, we model the DF concept learning space, and the fourth section describes the concept learning model based on the DF lattice. In the fifth section, we present a concept learning model based on dynamic fuzzy decision tree. In the sixth section, we discuss some applications of the dynamic concept based on dynamic fuzzy sets and analyse their performance. In the seventh section, we summarize this chapter.

The fifth chapter concentrates on semi-supervised multi-task learning based on dynamic fuzzy learning. This chapter is divided into six sections. The first section introduces the notion of multi-task learning. In the second section, we describe the semi-supervised multi-task learning model. In the third section, we introduce a semi-supervised multi-task learning model based on DFS. In the fourth section, we introduce a dynamic fuzzy semi-supervised multi-task matching algorithm. The fifth section extends this to a dynamic fuzzy semi-supervised multi-task adaptive learning algorithm. The sixth section summarizes this chapter.

In the sixth chapter, we study dynamic fuzzy hierarchy learning. This chapter is divided into seven sections. The first section introduces the idea of hierarchical learning. In the second section, we describe the design of an inductive logic program. The third section discusses dynamic fuzzy hierarchical relational learning (HRL). In the fourth section, we study dynamic fuzzy tree HRL, and the fifth section discusses dynamic fuzzy graph HRL. In the sixth section, we give some applications of dynamic concepts based on dynamic fuzzy sets and analyse their performance. The seventh section summarizes this chapter.

In Chapter 7, we consider a multi-agent learning model based on DFL. This chapter is divided into five sections. The first section introduces multi-agent learning. In the second section, we introduce the agent mental model based on DFL. In the third section, we introduce the single agent learning algorithm based on DFL. The fourth section extends this idea to a multi-agent learning model based on DFL. In the fifth section, we summarize this chapter. This book systematically introduces the relevant content of dynamic fuzzy learning. It can be used as a reference book for senior college students and graduate students as well as college teachers and scientific and technical personnel involved in computer science, artificial intelligence, machine learning, automation, mathematics, management science, cognitive science, financial management, and data analysis. The text can also be used as the basis for a lecture course on dynamic fuzzy learning.

This book was designed and revised by Professors Li Fanzhang, Zhang Li, and Zhang Zhao. Huang Shuning, Cui Jingmei, Zoupeng, Luo Xiaohui, Wu Xinjian, Li Meixuan, Yin Hongwei, and Xu Xiaoxiang also assisted with the writing of the book. We are grateful for the wisdom and work of all teachers and students involved in the process. This book has also cited a large number of references, and it is our honour to express our deep gratitude to the authors of all references. Thanks to the National Natural Science Foundation (61033013, 60775045, 61672364, 61672365), Soochow scholar program (14317360, 58320007) and the key subjects of Jiangsu province “Technology of computer science” and “Software engineering” for the support of this book.

Finally, I wish the book can bring happiness and inspiration to readers! Because this work is the first attempt, combined with the author's experience and knowledge being limited, if there is any improper place, please contact us. Contact method: E-mail: lfzh@suda.edu.cn, Tel.: 13962116494.

Li Fanzhang
December 12, 2016
Soochow University

Contents

Preface — v

1 Dynamic fuzzy machine learning model — 1

- 1.1 Problem statement — 1
- 1.2 DFML model — 1
- 1.2.1 Basic concept of DFMLs — 2
- 1.2.2 DFML algorithm — 4
- 1.2.3 DFML geometric model description — 13
- 1.2.4 Simulation examples — 14
- 1.3 Relative algorithm of DFMLS — 16
 - 1.3.1 Parameter learning algorithm for DFMLS — 16
 - 1.3.2 Maximum likelihood estimation algorithm in DFMLS — 21
- 1.4 Process control model of DFMLS — 29
 - 1.4.1 Process control model of DFMLS — 29
 - 1.4.2 Stability analysis — 30
 - 1.4.3 Design of dynamic fuzzy learning controller — 34
 - 1.4.4 Simulation examples — 36
- 1.5 Dynamic fuzzy relational learning algorithm — 39
 - 1.5.1 An outline of relational learning — 40
 - 1.5.2 Problem introduction — 43
 - 1.5.3 DFRL algorithm — 44
 - 1.5.4 Algorithm analysis — 47
- 1.6 Summary — 48

References — 48

2 Dynamic fuzzy autonomic learning subspace algorithm — 51

- 2.1 Research status of autonomic learning — 51
- 2.2 Theoretical system of autonomous learning subspace based on DFL — 54
 - 2.2.1 Characteristics of AL — 54
 - 2.2.2 Axiom system of AL subspace — 56
- 2.3 Algorithm of ALSS based on DFL — 57
- 2.3.1 Preparation of algorithm — 58
- 2.3.2 Algorithm of ALSS based on DFL — 60
- 2.3.3 Case analysis — 63
- 2.4 Summary — 66

References — 66

3	Dynamic fuzzy decision tree learning — 69
3.1	Research status of decision trees — 69
3.1.1	Overseas research status — 69
3.1.2	Domestic research status — 70
3.2	Decision tree methods for a dynamic fuzzy lattice — 72
3.2.1	ID3 algorithm and examples — 72
3.2.2	Characteristics of dynamic fuzzy analysis of decision trees — 74
3.2.3	Representation methods for dynamic fuzzy problems in decision trees — 74
3.2.4	DFDT classification attribute selection algorithm — 77
3.2.5	Dynamic fuzzy binary decision tree — 82
3.3	DFDT special attribute processing technique — 86
3.3.1	Classification of attributes — 87
3.3.2	Process used for enumerated attributes by DFDT — 87
3.3.3	Process used for numeric attributes by DFDT — 88
3.3.4	Methods to process missing value attributes in DFDT — 94
3.4	Pruning strategy of DFDT — 98
3.4.1	Reasons for pruning — 98
3.4.2	Methods of pruning — 100
3.4.3	DFDT pruning strategy — 101
3.5	Application — 104
3.5.1	Comparison of algorithm execution — 104
3.5.2	Comparison of training accuracy — 105
3.5.3	Comprehensibility comparisons — 109
3.6	Summary — 110
	References — 110
4	Concept learning based on dynamic fuzzy sets — 115
4.1	Relationship between dynamic fuzzy sets and concept learning — 115
4.2	Representation model of dynamic fuzzy concepts — 115
4.3	DF concept learning space model — 117
4.3.1	Order model of DF concept learning — 117
4.3.2	DF concept learning calculation model — 120
4.3.3	Dimensionality reduction model of DF instances — 125
4.3.4	Dimensionality reduction model of DF attribute space — 126
4.4	Concept learning model based on DF lattice — 129
4.4.1	Construction of classical concept lattice — 129
4.4.2	Constructing lattice algorithm based on DFS — 132
4.4.3	DF Concept Lattice Reduction — 135
4.4.4	Extraction of DF concept rules — 137
4.4.5	Examples of algorithms and experimental analysis — 139
4.5	Concept learning model based on DFDT — 142

4.5.1	DF concept tree and generating strategy — 142
4.5.2	Generation of DF Concepts — 143
4.5.3	DF concept rule extraction and matching algorithm — 151
4.6	Application examples and analysis — 152
4.6.1	Face recognition experiment based on DF concept lattice — 152
4.6.2	Data classification experiments on UCI datasets — 156
4.7	Summary — 159
References	— 159

5	Semi-supervised multi-task learning based on dynamic fuzzy sets — 161
5.1	Introduction — 161
5.1.1	Review of semi-supervised multi-task learning — 161
5.1.2	Problem statement — 166
5.2	Semi-supervised multi-task learning model — 167
5.2.1	Semi-supervised learning — 167
5.2.2	Multi-task learning — 172
5.3	Semi-supervised multi-task learning model based on DFS — 178
5.3.1	Dynamic fuzzy machine learning model — 179
5.3.2	Dynamic fuzzy semi-supervised learning model — 180
5.3.3	DFSSMTL model — 180
5.4	Dynamic fuzzy semi-supervised multi-task matching algorithm — 182
5.4.1	Dynamic fuzzy random probability — 183
5.4.2	Dynamic fuzzy semi-supervised multi-task matching algorithm — 184
5.4.3	Case analysis — 189
5.5	DFSSMTAL algorithm — 192
5.5.1	Mahalanobis distance metric — 192
5.5.2	Dynamic fuzzy K-nearest neighbour algorithm — 193
5.5.3	Dynamic fuzzy semi-supervised adaptive learning algorithm — 196
5.6	Summary — 205
References	— 206

6	Dynamic fuzzy hierarchical relationships — 209
6.1	Introduction — 209
6.1.1	Research progress of relationship learning — 209
6.1.2	Questions proposed — 214
6.1.3	Chapter structure — 215
6.2	Inductive logic programming — 215
6.3	Dynamic fuzzy HRL — 217
6.3.1	DFL relation learning algorithm (DFLR) — 217
6.3.2	Sample analysis — 222
6.3.3	Dynamic fuzzy matrix HRL algorithm — 226
6.3.4	Sample analysis — 232

6.4	Dynamic fuzzy tree hierarchical relation learning — 235
6.4.1	Dynamic fuzzy tree — 235
6.4.2	Dynamic fuzzy tree hierarchy relationship learning algorithm — 238
6.4.3	Sample analysis — 246
6.5	Dynamic fuzzy graph hierarchical relationship learning — 249
6.5.1	Basic concept of dynamic fuzzy graph — 249
6.5.2	Dynamic fuzzy graph hierarchical relationship learning algorithm — 253
6.5.3	Sample analysis — 255
6.6	Sample application and analysis — 255
6.6.1	Question description — 256
6.6.2	Sample analysis — 260
6.7	Summary — 262
	References — 262
7	Multi-agent learning model based on dynamic fuzzy logic — 267
7.1	Introduction — 267
7.1.1	Strategic classification of the agent learning method — 267
7.1.2	Characteristics of agent learning — 267
7.1.3	Related work — 268
7.2	Agent mental model based on DFL — 269
7.2.1	Model structure — 269
7.2.2	Related axioms — 274
7.2.3	Working mechanism — 275
7.3	Single-agent learning algorithm based on DFL — 277
7.3.1	Learning task — 277
7.3.2	Immediate return single-agent learning algorithm based on DFL — 277
7.3.3	Q-learning function based on DFL — 279
7.3.4	Q-learning algorithm based on DFL — 280
7.4	Multi-agent learning algorithm based on DFL — 282
7.4.1	Multi-agent learning model based on DFL — 282
7.4.2	Cooperative multi-agent learning algorithm based on DFL — 282
7.4.3	Competitive multi-agent learning algorithm based on DFL — 298
7.5	Summary — 299
	References — 299
8	Appendix — 301
8.1	Dynamic fuzzy sets — 301
8.1.1	Definition of dynamic fuzzy sets — 301
8.1.2	Operation of dynamic fuzzy sets — 301
8.1.3	Cut set of dynamic fuzzy sets — 304
8.1.4	Dynamic fuzzy sets decomposition theorem — 305

8.2	Dynamic fuzzy relations — 308
8.2.1	The conception dynamic fuzzy relations — 308
8.2.2	Property of dynamic fuzzy relations — 309
8.2.3	Dynamic fuzzy matrix — 310
8.3	Dynamic fuzzy logic — 312
8.3.1	Dynamic fuzzy Boolean variable — 312
8.3.2	DF proposition logic formation — 313
8.4	Dynamic fuzzy lattice and its property — 316

Index — 321

1 Dynamic fuzzy machine learning model

This chapter is divided into six sections. Section 1.1 proposes the problem, Section 1.2 introduces the dynamic fuzzy machine learning (DFML) model, and Section 1.3 describes the dynamic fuzzy machine learning system (DFMLS)-related algorithms. Section 1.4 discusses a process control model for DFML, and Section 1.5 presents a dynamic fuzzy relation learning algorithm. Finally, Section 1.6 gives a summary of this chapter.

1.1 Problem statement

Although machine learning theory has been widely used in various learning paradigms, the currently accepted concept of machine learning is as follows: “If a system can improve its performance by performing a certain process, it is learning”. The point of this argument is, first, that learning is a process; second, that learning considers the whole system; and third, that learning can change the performance of the system. “Process”, “system” and “performance change” are the three main points of learning. Obviously, if the system is considered human, the same argument can be established. According to this statement, existing machine learning methods could be considered somewhat limited. The reason is that these three points have a dynamic fuzzy character: the learning process is essentially dynamic and fuzzy; changes in the system (i.e. whether a system is “good” or “bad” and so on) are essentially dynamic and fuzzy; changes in system performance, results, and so on are all dynamic and fuzzy. These dynamic fuzzy characteristics are ubiquitous in the whole of machine learning and machine learning systems. Therefore, we believe that to make progress and meet the above definition of machine learning, the key question is to be able to effectively solve the dynamic fuzzy problems arising from learning activities.

However, existing machine learning methods are not sufficient for dynamic fuzzy problems. For example, Rough Set theory is based on fuzzy sets and can solve fuzziness problems, but it cannot handle dynamic problems; statistical machine learning is based on small-sample statistics and can solve static problems but not dynamic problems; Reinforcement learning is based on Markov processes and can solve dynamic problems but not fuzzy problems. Therefore, choosing dynamic fuzzy sets (DFSs) to study machine learning is a kind of inevitable choice. For the basic concepts of DFSs, see Appendix 8.1 and References [1, 2–4].

1.2 DFML model

To effectively deal with the dynamic fuzzy problems that arise in machine learning activities, a coordination machine learning model based on dynamic fuzzy sets (DFS)

and related algorithms has been proposed [9, 10]. Further work on this foundation has put forward the method of DFML and described a DFMLM and related algorithms in terms of their algebra and geometry [5–8, 11]. This section introduces the basic concepts of DFML, DFML algorithms and their stability analysis, and the geometry of DFML.

1.2.1 Basic concept of DFMLs

The process of system learning can be considered as self-adjusting, reflecting a series of changes in the system structure or parameters. Using mathematical language, learning can be defined as a mapping from one set to another set.

Definition 1.1 Dynamic fuzzy machine learning space: The space used to describe the learning process, which consists of all the DFML elements, is called the DFML space. It consists of five elements: {learning sample, learning algorithm, input data, output data, representation theory}, and can be expressed as $(\overleftarrow{S}, \overrightarrow{S}) = \{(\overleftarrow{Ex}, \overrightarrow{Ex}), ER, (\overleftarrow{X}, \overrightarrow{X}), (\overleftarrow{Y}, \overrightarrow{Y}), ET\}$.

Definition 1.2 Dynamic fuzzy machine learning: DFML $(\overleftarrow{I}, \overrightarrow{I})$ is the mapping of an input dataset $(\overleftarrow{X}, \overrightarrow{X})$ to an output dataset $(\overleftarrow{Y}, \overrightarrow{Y})$ in the DFML space $(\overleftarrow{S}, \overrightarrow{S})$. This can be expressed as $(\overleftarrow{I}, \overrightarrow{I}): (\overleftarrow{X}, \overrightarrow{X}) \rightarrow (\overleftarrow{Y}, \overrightarrow{Y})$.

Definition 1.3 Dynamic fuzzy machine learning system (DFMLS): The five elements of the DFMLS $(\overleftarrow{S}, \overrightarrow{S})$ can be combined with a certain learning mechanism to form a computer system with learning ability. This is called a Dynamic Fuzzy Machine Learning System.

Definition 1.4 Dynamic fuzzy machine learning model (DFMLM): $DFMLM = \{(\overleftarrow{S}, \overrightarrow{S}), (\overleftarrow{L}, \overrightarrow{L}), (\overleftarrow{u}, \overrightarrow{u}), (\overleftarrow{y}, \overrightarrow{y}), \dots, (\overleftarrow{I}, \overrightarrow{I}), (\overleftarrow{O}, \overrightarrow{O})\}$, where $(\overleftarrow{S}, \overrightarrow{S})$ is the part to be learned (dynamic environment/dynamic fuzzy learning space), $(\overleftarrow{L}, \overrightarrow{L})$ is the part to be dynamically learned, $(\overleftarrow{u}, \overrightarrow{u})$ is the output of $(\overleftarrow{S}, \overrightarrow{S})$ to $(\overleftarrow{L}, \overrightarrow{L})$, $(\overleftarrow{y}, \overrightarrow{y})$ is the dynamic feedback from $(\overleftarrow{L}, \overrightarrow{L})$ to $(\overleftarrow{S}, \overrightarrow{S})$, $(\overleftarrow{p}, \overrightarrow{p})$ is the system learning performance index, $(\overleftarrow{I}, \overrightarrow{I})$ is the input to the DFMLS from the external environment, and $(\overleftarrow{O}, \overrightarrow{O})$ is the output of the system to the outside world. A schematic diagram is given in Fig. 1.1.

If we discretize the processing system, $(\overleftarrow{S}, \overrightarrow{S})$ and $(\overleftarrow{L}, \overrightarrow{L})$ are represented by the state space, so we have the following definition.

Definition 1.5 DFMLM can be described as

$$(\overleftarrow{x}, \overrightarrow{x})(k+1) = G_1((\overleftarrow{x}, \overrightarrow{x})(k), (\overleftarrow{u}, \overrightarrow{u})(k), \xi_1(k)), \quad (1.1)$$

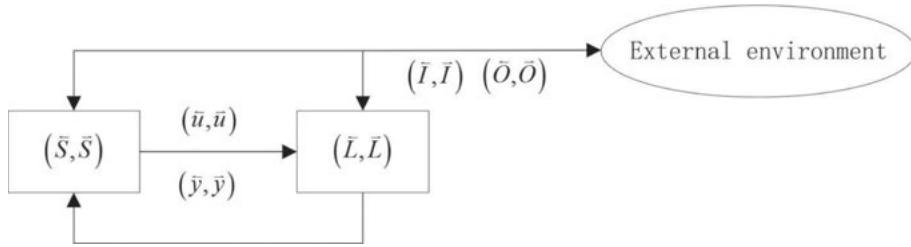


Fig. 1.1: DFMLM block diagram.

$$(\bar{y}, \bar{y})(k) = G_2(\bar{x}, \bar{x})(k), (\bar{u}, \bar{u})(k), \xi_2(k), \text{ and} \quad (1.2)$$

$$(\bar{p}, \bar{p})(k) = \sum_{i=1}^k P((\bar{y}, \bar{y})(i)), \quad (1.3)$$

where $(\bar{x}, \bar{x})(k)$ is the state variable of (\bar{S}, \bar{S}) at time k , $(\bar{u}, \bar{u})(k)$ is the dynamic output of (\bar{S}, \bar{S}) , $\xi_1(k)$ is a random disturbance in the state equation, $(\bar{y}, \bar{y})(k)$ is the dynamic feedback input of (\bar{L}, \bar{L}) , and $\xi_2(k)$ is observational random error. k represents the time period; only integer values are taken. We assume that all of the vectors are finite-dimensional state variables. (\bar{p}, \bar{p}) represents the system learning performance index, and $P(i)$ is a scalar function that represents the system learning performance at time i .

According to the definition, the following three propositions can be obtained:

Proposition 1.1 DFMLS is a random system.

Proposition 1.2 DFMLS is an open system.

Proposition 1.3 DFMLS is a nonlinear system.

Definition 1.6 The model of DFML process can be described as

$$DFMLP = \{(\bar{So}, \bar{So}), (\bar{Y}, \bar{Y}), (\bar{Op}, \bar{Op}), (\bar{V}, \bar{V}), ER, (\bar{G}, \bar{G})\},$$

where:

(\bar{So}, \bar{So}) is the source field, which refers to the DFMLS facing the learning material set. Its elements are (\bar{s}, \bar{o}) .

(\bar{Y}, \bar{Y}) is the target domain, which is the set of knowledge acquired by the DFMLS. Its elements are (\bar{y}, \bar{y}) .

(\bar{Op}, \bar{Op}) is an operational mechanism, which refers to a set of actions from the source field to the target domain.

$(\overleftarrow{V}, \overrightarrow{V})$ is an evaluation mechanism, which is a set for evaluating and correcting or deleting target domain elements.

ER is an execution algorithm, which is the implementation of the algorithm to verify the source field elements and provide executive information.

$(\overleftarrow{G}, \overrightarrow{G})$ is an incentive mechanism, which refers to the control and coordination of environmental communication.

This definition leads to the following proposition:

Proposition 1.4 DFML is an orderly process controlled by an incentive mechanism. Its general procedure is as follows:

- (1) For a relevant subset $(\overleftarrow{So}^+, \overrightarrow{So}^+)$ in the source field $(\overleftarrow{So}, \overrightarrow{So})$, under incentive mechanism $(\overleftarrow{G}, \overrightarrow{G})$, a subset $(\overleftarrow{Op}, \overrightarrow{Op})$ of the operating mechanism is activated according to the common characteristics of $(\overleftarrow{So}^+, \overrightarrow{So}^+)$ in the role of $(\overleftarrow{Op}^+, \overrightarrow{Op}^+)$, forming a subset $(\overleftarrow{Y}^+, \overrightarrow{Y}^+)$ of the target domain $(\overleftarrow{Y}, \overrightarrow{Y})$. That is, $(\overleftarrow{Op}^+, \overrightarrow{Op}^+)_{{(\overleftarrow{So}^+, \overrightarrow{So}^+)} \Rightarrow (\overleftarrow{Y}^+, \overrightarrow{Y}^+) \subseteq (\overleftarrow{Y}, \overrightarrow{Y})}$.
- (2) Consider the elements $(\overleftarrow{y}_0, \overrightarrow{y}_0)$ in the target domain $(\overleftarrow{Y}, \overrightarrow{Y})$. In the execution under the action of algorithm ER , there is some deviation information $(\overleftarrow{y}_0, \overrightarrow{y}_0)$. The incentive mechanism $(\overleftarrow{G}, \overrightarrow{G})$ is based on a subset $(\overleftarrow{V}^+, \overrightarrow{V}^+)$ of the activation evaluation mechanisms $(\overleftarrow{V}, \overrightarrow{V})$ and act on the environment and the environmental response $N(\overleftarrow{y}_0, \overrightarrow{y}_0)$ and $E(\overleftarrow{y}_0, \overrightarrow{y}_0)$ in the role of $(\overleftarrow{V}^+, \overrightarrow{V}^+)$. We then revise $(\overleftarrow{y}_0, \overrightarrow{y}_0)$. That is,

$$(\overleftarrow{V}^+, \overrightarrow{V}^+)_{{(N(\overleftarrow{y}_0, \overrightarrow{y}_0) \cup E(\overleftarrow{y}_0, \overrightarrow{y}_0))} \Rightarrow (\overleftarrow{y}_0', \overrightarrow{y}_0') \subseteq (\overleftarrow{Y}, \overrightarrow{Y})}.$$

- (3) For a subset $(\overleftarrow{So}^+, \overrightarrow{So}^+)$ of the source field $(\overleftarrow{So}, \overrightarrow{So})$, the results of its operation and evaluation are

$$(\overleftarrow{V}, \overrightarrow{V})[(\overleftarrow{Op}, \overrightarrow{Op})(\overleftarrow{So}, \overrightarrow{So})] \Rightarrow (\overleftarrow{Y}^+, \overrightarrow{Y}^+) \subseteq (\overleftarrow{Y}, \overrightarrow{Y}).$$

- (4) Take the above target as a new source field.
- (5) Repeat the above steps until the learning process meets the accuracy requirements. A schematic diagram is shown in Fig. 1.2.

1.2.2 DFML algorithm

In DFML, the curse of dimensionality can be a serious problem. In machine learning, many of the data are nonlinear and high-dimensional, which brings further difficulties to data processing in machine learning. Therefore, we need to reduce the dimension (i.e. dimensionality reduction) of the high-dimensional data.

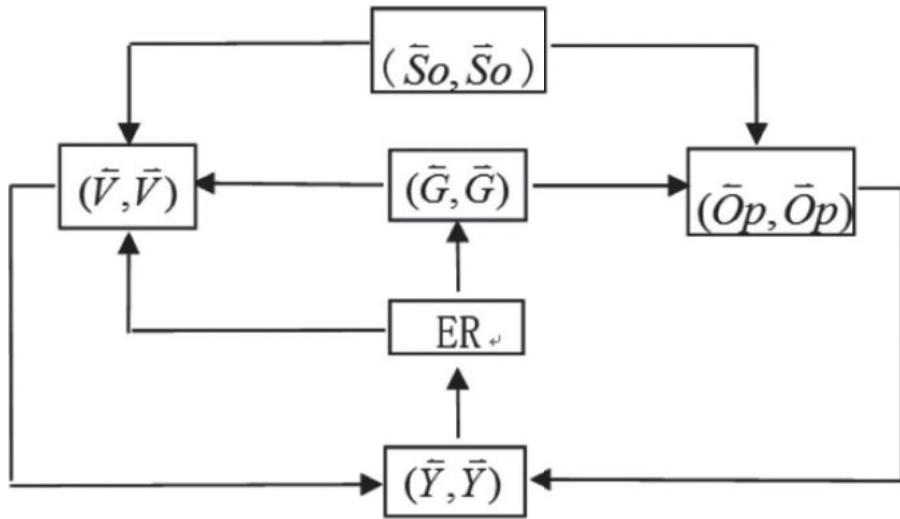


Fig. 1.2: Dynamic fuzzy machine learning process description model.

1.2.2.1 Locally linear embedding (LLE)

LLE is a well-known nonlinear dimension reduction method proposed by Saul and Roweis in 2000 [12]. Its basic idea is to transform the global nonlinearity into local linearity, with the overlapping local neighbourhoods providing the global structure of the information. In accordance with certain rules, each part of the linear dimension reduction is then combined with the results to give a low-dimensional global coordinate representation.

In the dynamic fuzzy high-dimensional space (\vec{R}^D, \vec{R}^D) , there is a dynamic fuzzy dataset $(\vec{X}, \vec{X}) = \{(\vec{x}_1, \vec{x}_1), (\vec{x}_2, \vec{x}_2), \dots, (\vec{x}_n, \vec{x}_n)\}$. LLE reduces the dimension (\vec{F}, \vec{F}) to map (\vec{X}, \vec{X}) to a relatively low dimension dynamic fuzzy space (\vec{R}^d, \vec{R}^d) and obtains the corresponding low-dimensional dynamic fuzzy vector $(\vec{Y}, \vec{Y}) = \{(\vec{y}_1, \vec{y}_1), (\vec{y}_2, \vec{y}_2), \dots, (\vec{y}_n, \vec{y}_n)\}$ while retaining as much information as possible from the original dynamic fuzzy data (DFD). That is, the topology of the dynamic fuzzy dataset (as determined by the neighbourhood relation of each point) is maintained. An effective approach is to replace the linear approximation with a linear structure in the local approximation, that is, to reduce the nonlinear dimensionality (\vec{F}, \vec{F}) by approximating the dimensional reduction locally, so that the problem is transformed into one of local linear dimension reduction.

Definition 1.7 The model for the dynamic fuzzy dimensionality reduction problem is $((\vec{X}, \vec{X}), (\vec{F}, \vec{F}))$, where the mapping $(\vec{F}, \vec{F}): (\vec{X}, \vec{X}) \rightarrow (\vec{Y}, \vec{Y})$. Let (\vec{F}, \vec{F}) be the dimension reduction from (\vec{X}, \vec{X}) to (\vec{Y}, \vec{Y}) for the dynamic fuzzy datasets.

Definition 1.8 The mapping $(\overleftarrow{f}, \overrightarrow{f}) : (\overleftarrow{Y}, \overrightarrow{Y}) \rightarrow (\overleftarrow{T}, \overrightarrow{T}) \subset (\overleftarrow{R}^D, \overrightarrow{R}^D)$ is called an embedded mapping.

Consider the arbitrary point $(\overleftarrow{x}_i, \overrightarrow{x}_i) \in (\overleftarrow{X}, \overrightarrow{X})$. Let $\{(\overleftarrow{x}_i^l, \overrightarrow{x}_i^l), l = 1, 2, \dots, k\}$ represent the k -nearest neighbourhood of $(\overleftarrow{x}_i, \overrightarrow{x}_i)$ [the DFS of k points nearest to $(\overleftarrow{x}_i, \overrightarrow{x}_i)$ in $(\overleftarrow{X}, \overrightarrow{X})$] and let the linear approximation of the local $(\overleftarrow{F}, \overrightarrow{F})$ be $(\overleftarrow{F}^i, \overrightarrow{F}^i)$. Let $(\overleftarrow{y}_i, \overrightarrow{y}_i) = (\overleftarrow{F}^i, \overrightarrow{F}^i)(\overleftarrow{x}_i, \overrightarrow{x}_i)$, $(\overleftarrow{y}_i^l, \overrightarrow{y}_i^l) = (\overleftarrow{F}^i, \overrightarrow{F}^i)(\overleftarrow{x}_i^l, \overrightarrow{x}_i^l)$. Then, we have the following theorem:

Theorem 1.1 If $(\overleftarrow{x}_i, \overrightarrow{x}_i) = \sum_{l=1}^k (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)(\overleftarrow{x}_i^l, \overrightarrow{x}_i^l)$, then [13]

$$(\overleftarrow{y}_i, \overrightarrow{y}_i) = \sum_{l=1}^k (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)(\overleftarrow{y}_i^l, \overrightarrow{y}_i^l),$$

where $(\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)$ ($i = 1, 2, \dots, n; l = 1, 2, \dots, k$) is a linear combination of dynamic fuzzy coefficients satisfying $\sum_{l=1}^k (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l) = (\overleftarrow{1}, \overrightarrow{1})$.

Proof: Because $(\overleftarrow{F}^i, \overrightarrow{F}^i)$ is a linear dimensional reduction, there exists a $d \times D$ matrix $(\overleftarrow{A}, \overrightarrow{A})$ and a d -dimensional vector $(\overleftarrow{b}, \overrightarrow{b})$ such that

$$(\overleftarrow{Y}, \overrightarrow{Y}) = (\overleftarrow{F}^i, \overrightarrow{F}^i)(\overleftarrow{X}, \overrightarrow{X}) = (\overleftarrow{A}, \overrightarrow{A})(\overleftarrow{X}, \overrightarrow{X}) + (\overleftarrow{b}, \overrightarrow{b}).$$

We have

$$\begin{aligned} (\overleftarrow{y}_i, \overrightarrow{y}_i) &= (\overleftarrow{F}^i, \overrightarrow{F}^i)(\overleftarrow{x}_i, \overrightarrow{x}_i) = (\overleftarrow{A}, \overrightarrow{A})(\overleftarrow{x}_i, \overrightarrow{x}_i) + (\overleftarrow{b}, \overrightarrow{b}) \\ &= (\overleftarrow{A}, \overrightarrow{A}) \sum_l (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)(\overleftarrow{x}_i^l, \overrightarrow{x}_i^l) + (\overleftarrow{b}, \overrightarrow{b}) \\ &= \sum_l ((\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)(\overleftarrow{A}, \overrightarrow{A})(\overleftarrow{x}_i^l, \overrightarrow{x}_i^l)) + \sum_l (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)(\overleftarrow{b}, \overrightarrow{b}) \\ &= \sum_l (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)((\overleftarrow{A}, \overrightarrow{A})(\overleftarrow{x}_i^l, \overrightarrow{x}_i^l) + (\overleftarrow{b}, \overrightarrow{b})) \\ &= \sum_l (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)(\overleftarrow{F}^i, \overrightarrow{F}^i)(\overleftarrow{x}_i^l, \overrightarrow{x}_i^l) \\ &= \sum_l (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)(\overleftarrow{y}_i^l, \overrightarrow{y}_i^l). \end{aligned}$$

This suggests that the local linear dimension reduction is a constant. $(\overleftarrow{x}_i, \overrightarrow{x}_i)$ is approximated by a linear combination of $\{(\overleftarrow{x}_i^l, \overrightarrow{x}_i^l) \mid l = 1, 2, \dots, k\}$: $(\overleftarrow{x}_i, \overrightarrow{x}_i) \sim \sum_{l=1}^k (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)(\overleftarrow{x}_i^l, \overrightarrow{x}_i^l)$, so the dynamic fuzzy coefficient $\{(\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)\}$ depicts the invariant features of the data structure under the reduced dimension. Then, according

$$\text{to } (\overleftarrow{y}_i, \overrightarrow{y}_i) = (\overleftarrow{F}^i, \overrightarrow{F}^i)(\overleftarrow{x}_i, \overrightarrow{x}_i) \sim (\overleftarrow{F}^i, \overrightarrow{F}^i) \sum_{l=1}^k (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)(\overleftarrow{x}_i^l, \overrightarrow{x}_i^l) = \sum_{l=1}^k (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)(\overleftarrow{y}_i^l, \overrightarrow{y}_i^l),$$

allowing us to identify $(\overleftarrow{y}_i, \overrightarrow{y}_i)$ through the information provided by $\{(\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)\}$.

The core of the LLE method is to find the k -dimensional dynamic fuzzy vector $(\overleftarrow{W}_i, \overrightarrow{W}_i) = ((\overleftarrow{w}_i^1, \overrightarrow{w}_i^1), (\overleftarrow{w}_i^2, \overrightarrow{w}_i^2), \dots, (\overleftarrow{w}_i^k, \overrightarrow{w}_i^k))$ and to minimize

$$(\overleftarrow{Q}, \overrightarrow{Q})(\overleftarrow{W}_i, \overrightarrow{W}_i) = \left\| (\overleftarrow{x}_i, \overrightarrow{x}_i) - \sum_{l=1}^k (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)(\overleftarrow{x}_i^l, \overrightarrow{x}_i^l) \right\|^2, \quad (1.4)$$

where $\sum_{l=1}^k (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l) = (\overleftarrow{1}, \overrightarrow{1})$, $(\overleftarrow{w}_i^l, \overrightarrow{w}_i^l) = (\overleftarrow{0}, \overrightarrow{0})$, and, when $(\overleftarrow{x}_j, \overrightarrow{x}_j) \notin \{(\overleftarrow{x}_i, \overrightarrow{x}_i) | l = 1, 2, \dots, k\}$, and $(\overleftarrow{w}_i^j, \overrightarrow{w}_i^j) = (\overleftarrow{0}, \overrightarrow{0})$. $(\overleftarrow{W}_i, \overrightarrow{W}_i)$ records the neighbourhood information of point $(\overleftarrow{x}_i, \overrightarrow{x}_i)$, that is, the local topological structure of each point, and the dynamic fuzzy matrix of all the components is denoted by $(\overleftarrow{W}, \overrightarrow{W})$. At this point, choose the appropriate dimension d and perform R^l d -dimensional embeddings, that is, seek $(\overleftarrow{Y}, \overrightarrow{Y}) = ((\overleftarrow{y}_1, \overrightarrow{y}_1), (\overleftarrow{y}_2, \overrightarrow{y}_2), \dots, (\overleftarrow{y}_n, \overrightarrow{y}_n)) \in (\overleftarrow{R}^{d \times n}, \overrightarrow{R}^{d \times n})$ to satisfy

$$\begin{aligned} (\overleftarrow{L}, \overrightarrow{L})((\overleftarrow{y}_1^*, \overrightarrow{y}_1^*), (\overleftarrow{y}_2^*, \overrightarrow{y}_2^*), \dots, (\overleftarrow{y}_n^*, \overrightarrow{y}_n^*)) &= \min_{(Y, Y)} \sum_{i=1}^n \left\| (\overleftarrow{y}_i, \overrightarrow{y}_i) - (\overleftarrow{Y}, \overrightarrow{Y})(\overleftarrow{W}_i, \overrightarrow{W}_i)^T \right\|^2 \\ &= \min_{(Y, Y)} \text{tr}((\overleftarrow{Y}, \overrightarrow{Y})(\overleftarrow{M}, \overrightarrow{M})(\overleftarrow{Y}, \overrightarrow{Y})^T) \\ (\overleftarrow{M}, \overrightarrow{M}) &= ((\overleftarrow{1}, \overrightarrow{1}) - (\overleftarrow{W}, \overrightarrow{W})^T)((\overleftarrow{1}, \overrightarrow{1}) - (\overleftarrow{W}, \overrightarrow{W})^T) \end{aligned}$$

because $(\overleftarrow{W}, \overrightarrow{W})$ is solved by the dynamic fuzzy dataset $(\overleftarrow{X}, \overrightarrow{X})$, and the computation is very sensitive to noise, especially when the eigenvalues of $(\overleftarrow{X}, \overrightarrow{X})^T(\overleftarrow{X}, \overrightarrow{X})$ are small. In this case, the desired result may not be obtained.

1.2.2.2 Analysis of noise interference

Let $(\overleftarrow{x}_i^l, \overrightarrow{x}_i^l)' = (\overleftarrow{x}_i^l, \overrightarrow{x}_i^l) + (\overleftarrow{\varepsilon}_i^l, \overrightarrow{\varepsilon}_i^l)$, $(i = 1, 2, \dots, k)$ denote the corresponding noise-affected point and $(\overleftarrow{x}_i, \overrightarrow{x}_i)' = \sum_{l=1}^k (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)'(\overleftarrow{x}_i^l, \overrightarrow{x}_i^l)', \sum_{l=1}^k (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l)' = (\overleftarrow{1}, \overrightarrow{1})$; $(\overleftarrow{U}(\overleftarrow{x}_i), \overrightarrow{U}(\overrightarrow{x}_i)) = ((\overleftarrow{x}_i^1, \overrightarrow{x}_i^1), (\overleftarrow{x}_i^2, \overrightarrow{x}_i^2), \dots, (\overleftarrow{x}_i^k, \overrightarrow{x}_i^k))$ is the k -neighbourhood DFS of $(\overleftarrow{x}_i, \overrightarrow{x}_i)$, $(\overleftarrow{U}(\overleftarrow{x}_i), \overrightarrow{U}(\overrightarrow{x}_i))' = ((\overleftarrow{x}_i^1, \overrightarrow{x}_i^1)', (\overleftarrow{x}_i^2, \overrightarrow{x}_i^2)', \dots, (\overleftarrow{x}_i^k, \overrightarrow{x}_i^k)')$ is the k -neighbourhood DFS of $(\overleftarrow{x}_i, \overrightarrow{x}_i)'$, and

$$(\overleftarrow{x}_i, \overrightarrow{x}_i) = (\overleftarrow{U}(\overleftarrow{x}_i), \overrightarrow{U}(\overrightarrow{x}_i))(\overleftarrow{W}_i, \overrightarrow{W}_i),$$

$$(\overleftarrow{x}_i, \overrightarrow{x}_i)' = (\overleftarrow{U}(\overleftarrow{x}_i), \overrightarrow{U}(\overrightarrow{x}_i))'(\overleftarrow{W}_i, \overrightarrow{W}_i)'.$$

Theorem 1.2 If each point between the noise and between different dimensions, and between $(\bar{W}_i, \bar{W}_i)'$ and the noise are independent of each other, the noise has a mean value with the same variance [14]. Then, $\delta(\bar{W}, \bar{W}) = (\bar{W}_i, \bar{W}_i)' - (\bar{W}_i, \bar{W}_i)$ can be estimated as

$$E\|\delta(\bar{W}, \bar{W})\|^2 \leq \frac{k(k+1)(\bar{\delta}, \bar{\delta})^2}{(\bar{\lambda}, \bar{\lambda})_{\min}} E\|(\bar{W}, \bar{W})'\|^2,$$

where $\|\cdot\|$ is the Euclidean norm; $(\bar{\delta}, \bar{\delta})^2 = \sum_{i=1}^D (\bar{\delta}_i, \bar{\delta}_i)^2$, $(\bar{\delta}_i, \bar{\delta}_i) = \text{Var}(\bar{\varepsilon}_i, \bar{\varepsilon}_i)$ ($i = 1, 2, \dots, D$), $(\bar{\varepsilon}_i, \bar{\varepsilon}_i)$ represents the i th component of $(\bar{\varepsilon}, \bar{\varepsilon})$, $l = \text{rank}(\bar{U}(\bar{x}_i), \bar{U}(\bar{x}_i))$, and $(\bar{\lambda}, \bar{\lambda})_{\min}$ is the smallest nonzero eigenvalue of $(\bar{U}(\bar{x}_i), \bar{U}(\bar{x}_i))^T(\bar{U}(\bar{x}_i), \bar{U}(\bar{x}_i))$.

From this theorem, it can be concluded that when the neighbourhood size k is known, the error of (\bar{W}, \bar{W}) is mainly determined by three factors: (1) the influence of noise interference, that is, the size of $(\bar{\delta}, \bar{\delta})$; (2) neighbourhood effects, that is, the size of $(\bar{\lambda}, \bar{\lambda})_{\min}$ and the rank l ; and (3) the impact of weight energy, that is, the size of $\|(\bar{W}, \bar{W})'\|$.

1.2.2.3 Improved locally linear embedding (ILLE)

Determining the size of the neighbourhood k is a very important problem. If k is too small, the mapping will cause the sub-manifolds to break away from the continuous manifold and not reflect any global properties. If k is too large, the entire dataset will be considered as the local neighbourhood, and the map will lose its nonlinear characteristic [15]. In this section, we propose an ILLE method that uses the diffusion and growing self-organizing map (DGSOM) [16] to define neighbourhood relations. For example, if a point has five associated points in the DGSOM map, the five data points are considered neighbours of that point. Thus, the size of the neighbourhood of each sample point, k , is different, which overcomes the shortcoming of the traditional LLE method that the neighbourhood size of each point is fixed. In addition, the ILLE reduces the influence of noise by reducing the noise interference and weighting energy. By way of example, we show that ILLE can adapt to the neighbourhood and overcome the sensitivity problem, enabling it to solve the error problem caused by the three factors identified above.

The core of the ILLE method is to seek the k -dimensional dynamic fuzzy vector $(\bar{W}_i, \bar{W}_i) = ((\bar{w}_i^1, \bar{w}_i^1), (\bar{w}_i^2, \bar{w}_i^2), \dots, (\bar{w}_i^k, \bar{w}_i^k))$. The optimization problem is

$$\begin{aligned} & \text{Min } (\bar{Q}, \bar{Q})(\bar{W}_i, \bar{W}_i) \\ &= \left\| (\bar{x}_i, \bar{x}_i) - \sum_{l=1}^k (\bar{w}_i^l, \bar{w}_i^l) (\bar{x}_i^l, \bar{x}_i^l) \right\|^2 + (\bar{\eta}, \bar{\eta})(\bar{g}, \bar{g}) \left(\left\| \sum_{l=1}^k (\bar{w}_i^l, \bar{w}_i^l) \right\| \right) \end{aligned} \quad (1.5)$$

$$\text{s.t. } ((\overleftarrow{\eta}, \overrightarrow{\eta}) \geq (\overleftarrow{0}, \overrightarrow{0}), \sum_{l=1}^k (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l) = (\overleftarrow{1}, \overrightarrow{1})),$$

which is to find the corresponding dynamic fuzzy topology matrix $(\overleftarrow{W}, \overrightarrow{W})$, and finally get $(\overleftarrow{M}, \overrightarrow{M})$.

ILLE avoids the ill-posedness of LLE by introducing the dynamic fuzzy regularization term $(\overleftarrow{g}, \overrightarrow{g}) \left(\left\| \sum_{l=1}^k (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l) \right\| \right)$, which reduces the influence of noise interference. This term must satisfy two points: (1) (1.5) must have a steady solution; and (2) $\left\| \sum_{l=1}^k (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l) \right\|$ in (1.5) must be minimized, typically such that $(\overleftarrow{g}, \overrightarrow{g})(\overleftarrow{x}, \overrightarrow{x}) = (\overleftarrow{x}, \overrightarrow{x})^2$. $(\overleftarrow{\eta}, \overrightarrow{\eta})$ is the improvement factor, which is related to the neighbourhood number k , the dimension of the dynamic fuzzy dataset \mathbf{D} , and the dimension of the dynamic fuzzy dataset. We can see from the calculation that the choice of $(\overleftarrow{\eta}, \overrightarrow{\eta})$ does not have a large effect on the outcome. The description of ILLE is shown in Algorithm 1.1.

Algorithm 1.1 ILLE algorithm

Input: $(\overleftarrow{X}, \overrightarrow{X}) = \{(\overleftarrow{x}_1, \overrightarrow{x}_1), (\overleftarrow{x}_2, \overrightarrow{x}_2), \dots, (\overleftarrow{x}_n, \overrightarrow{x}_n)\} \subset (\overleftarrow{R}^{D \times n}, \overrightarrow{R}^{D \times n})$,
 $(\overleftarrow{x}_i, \overrightarrow{x}_i) \in (\overleftarrow{R}^D, \overrightarrow{R}^D) (i = 1, 2, \dots, n)$

Output: $(\overleftarrow{Y}, \overrightarrow{Y}) = \{(\overleftarrow{y}_1, \overrightarrow{y}_1), (\overleftarrow{y}_2, \overrightarrow{y}_2), \dots, (\overleftarrow{y}_n, \overrightarrow{y}_n)\} \subset (\overleftarrow{R}^{d \times n}, \overrightarrow{R}^{d \times n})$,
 $(\overleftarrow{y}_i, \overrightarrow{y}_i) \in (\overleftarrow{R}^d, \overrightarrow{R}^d) (i = 1, 2, \dots, n)$

For each $(\overleftarrow{x}_i, \overrightarrow{x}_i)$ **do**

1. **Choose neighbourhood:** The k neighbourhoods $\{(\overleftarrow{x}_i^l, \overrightarrow{x}_i^l) | l = 1, 2, \dots, k\}$ of each point $(\overleftarrow{x}_i, \overrightarrow{x}_i)$ are determined according to the DGSMOM method, where k may be different for each neighbourhood. A linear combination of $\{(\overleftarrow{x}_i^l, \overrightarrow{x}_i^l) | l = 1, 2, \dots, k\}$ is used to denote $(\overleftarrow{x}_i, \overrightarrow{x}_i)$, that is, the hyperboloid passing through $\{(\overleftarrow{x}_i^l, \overrightarrow{x}_i^l) | l = 1, 2, \dots, k\}$ is used to approximate the local block near $(\overleftarrow{x}_i, \overrightarrow{x}_i)$ in the manifold.
2. **Get weight:** For each $(\overleftarrow{x}_i, \overrightarrow{x}_i)$, find $(\overleftarrow{W}_i, \overrightarrow{W}_i) = ((\overleftarrow{w}_i^1, \overrightarrow{w}_i^1), (\overleftarrow{w}_i^2, \overrightarrow{w}_i^2), \dots, (\overleftarrow{w}_i^k, \overrightarrow{w}_i^k))$ that minimizes $(\overleftarrow{Q}, \overrightarrow{Q})(\overleftarrow{W}_i, \overrightarrow{W}_i) = \left\| (\overleftarrow{x}_i, \overrightarrow{x}_i) - \sum_{l=1}^k (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l) (\overleftarrow{x}_i^l, \overrightarrow{x}_i^l) \right\|^2 + (\overleftarrow{\eta}, \overrightarrow{\eta}) \left\| \sum_{l=1}^k (\overleftarrow{w}_i^l, \overrightarrow{w}_i^l) \right\|^2$, and finally obtain the dynamic fuzzy topology matrix $(\overleftarrow{W}, \overrightarrow{W})$.
3. **Reduce dimension:** Minimize the loss function

$$\begin{aligned} (\overleftarrow{L}, \overrightarrow{L})((\overleftarrow{y}_1^*, \overrightarrow{y}_1^*), (\overleftarrow{y}_2^*, \overrightarrow{y}_2^*), \dots, (\overleftarrow{y}_n^*, \overrightarrow{y}_n^*)) &= \min_{(\overleftarrow{Y}, \overrightarrow{Y})} \sum_{i=1}^n \left\| (\overleftarrow{y}_i, \overrightarrow{y}_i) - (\overleftarrow{Y}, \overrightarrow{Y})(\overleftarrow{W}_i, \overrightarrow{W}_i)^T \right\|^2 \\ &= \min_{(\overleftarrow{Y}, \overrightarrow{Y})} (tr((\overleftarrow{Y}, \overrightarrow{Y})(\overleftarrow{M}, \overrightarrow{M})(\overleftarrow{Y}, \overrightarrow{Y})^T)) \\ (\overleftarrow{M}, \overrightarrow{M}) &= ((\overleftarrow{1}, \overrightarrow{1}) - (\overleftarrow{W}, \overrightarrow{W})^T)(\overleftarrow{1}, \overrightarrow{1}) - (\overleftarrow{W}, \overrightarrow{W})^T \end{aligned}$$

where $(\overleftarrow{Y}, \overrightarrow{Y}) = ((\overleftarrow{y}_1, \overrightarrow{y}_1), (\overleftarrow{y}_2, \overrightarrow{y}_2), \dots, (\overleftarrow{y}_n, \overrightarrow{y}_n)) \in (\overleftarrow{R}^{d \times n}, \overrightarrow{R}^{d \times n})$, $(\overleftarrow{y}_i, \overrightarrow{y}_i) \in (\overleftarrow{R}^d, \overrightarrow{R}^d) (i = 1, 2, \dots, n)$. To make the solution unique, we add the constraint to $(\overleftarrow{y}_i, \overrightarrow{y}_i) \in (\overleftarrow{R}^d, \overrightarrow{R}^d) (i = 1, 2, \dots, n)$:

$$\sum_{i=1}^n (\vec{y}_i, \vec{y}_i) = (\vec{0}, \vec{0}) \text{ and } (\vec{Y}, \vec{Y})^T (\vec{Y}, \vec{Y}) = (\vec{I}, \vec{I})_{d \times d}$$

where $(\vec{I}, \vec{I})_{d \times d}$ is the dynamic fuzzy unit matrix.

The computational complexity of each step of ILLE is as follows. The number of computations in steps 1–3 is $O(Dn^2)$, $O(Dnk^3)$, and $O(dn^2)$, respectively. The computational complexity of ILLE is similar to that of LLE, but ILLE is more robust against noise interference.

1.2.2.4 DFML algorithm and its stability analysis

A. DFML Algorithm [5, 7, 8, 11]

Consider the DFMLS set up by the following dynamic fuzzy rules:

R^l : IF $(\vec{x}, \vec{x})(k)$ is $(\vec{A}_1^l, \vec{A}_1^l)$ and $(\vec{x}, \vec{x})(k-1)$ is $(\vec{A}_2^l, \vec{A}_2^l)$ and ... and $(\vec{x}, \vec{x})(k-n+1)$ is $(\vec{A}_n^l, \vec{A}_n^l)$ and $(\vec{u}, \vec{u})(k)$ is (\vec{B}_l, \vec{B}_l) THEN

$$(\vec{x}, \vec{x})^l(k+1) = (\vec{A}_l, \vec{A}_l)(\vec{X}, \vec{X})(k) + (\vec{b}_l, \vec{b}_l)(\vec{u}, \vec{u})(k) \quad (l = 1, 2, \dots, m), \quad (1.6)$$

where $(\vec{X}, \vec{X})(k) = ((\vec{x}, \vec{x})(k), (\vec{x}, \vec{x})(k-1), \dots, (\vec{x}, \vec{x})(k-n+1))^T \in (\vec{U}, \vec{U})$ is the system state vector, $(\vec{A}_l, \vec{A}_l) = ((\vec{a}_1^l, \vec{a}_1^l), (\vec{a}_2^l, \vec{a}_2^l), \dots, (\vec{a}_n^l, \vec{a}_n^l))$, $(\vec{a}_i^l, \vec{a}_i^l) \quad (i = 1, 2, \dots, n)$ is the corresponding dynamic fuzzy constant, and $(\vec{A}_i^l, \vec{A}_i^l) \in (\vec{R}, \vec{R})^{n \times n}$, $(\vec{B}_l, \vec{B}_l) \in (\vec{R}, \vec{R})^{n \times n}$ is the dynamic fuzzy constant matrix. The linear state equation corresponding to each dynamic fuzzy rule is called the DFML subsystem.

For the problem of noise in the learning process, we use the robust factor proposed in [17] to reduce the interference:

$$(\vec{\phi}, \vec{\phi})((\vec{e}_i, \vec{e}_i), (\vec{c}_i, \vec{c}_i)) = \frac{\partial(\vec{\varphi}, \vec{\varphi})((\vec{e}_i, \vec{e}_i), (\vec{c}_i, \vec{c}_i))}{\partial((\vec{e}_i, \vec{e}_i))} \Big/ (\vec{e}_i, \vec{e}_i), \quad (1.7)$$

where $(\vec{e}_i, \vec{e}_i) = (\vec{y}_d, \vec{y}_d) - (\vec{y}_i, \vec{y}_i)$, (\vec{y}_d, \vec{y}_d) is the expectation output, (\vec{y}_i, \vec{y}_i) is the actual output, and (\vec{c}_i, \vec{c}_i) is the central value of the confidence interval for the statistical estimation of the deviation (\vec{e}_i, \vec{e}_i) near (\vec{x}_i, \vec{x}_i) . $(\vec{\phi}, \vec{\phi})((\vec{e}_i, \vec{e}_i), (\vec{c}_i, \vec{c}_i)) \geq (\vec{0}, \vec{0})$ is a nonnegative integrable function of (\vec{e}_i, \vec{e}_i) , and

$$\begin{aligned} \max \quad & (\vec{\phi}, \vec{\phi})((\vec{e}_i, \vec{e}_i), (\vec{c}_i, \vec{c}_i)) = (\vec{\phi}, \vec{\phi})((\vec{c}_i, \vec{c}_i), (\vec{c}_i, \vec{c}_i)) \\ \text{s.t.} \quad & (\vec{\varphi}, \vec{\varphi})((\vec{0}, \vec{0}), (\vec{c}_i, \vec{c}_i)) = (\vec{0}, \vec{0}). \end{aligned}$$

The learning algorithm of the DFMLS is shown in Algorithm 1.2.

Algorithm 1.2 Dynamic fuzzy machine learning

Input: $(\overrightarrow{x}, \overrightarrow{x})(k), (\overrightarrow{x}, \overrightarrow{x})(k-1), \dots, (\overrightarrow{x}, \overrightarrow{x})(k-n+1)^T \in (\overrightarrow{U}, \overrightarrow{U})$

Output: $(\overrightarrow{y}_k, \overrightarrow{y}_k)$

$((\overrightarrow{u}_{\overrightarrow{A}_l}), (\overrightarrow{u}_{\overrightarrow{A}_l}), (\overrightarrow{u}_{\overrightarrow{b}_l}), (\overrightarrow{u}_{\overrightarrow{b}_l}))$ is the corresponding dynamic fuzzy membership, $(\overrightarrow{y}_d, \overrightarrow{y}_d)$ is the desired output //

(1) Use the ILLE method to reduce the dimension of the DFD;

(2)

$$(\overrightarrow{x}, \overrightarrow{x})(k+1) = \frac{\sum_{l=1}^m (\overrightarrow{x}, \overrightarrow{x})^l (k+1) (\overrightarrow{v}, \overrightarrow{v})^l}{\sum_{l=1}^m (\overrightarrow{v}, \overrightarrow{v})^l} \quad (1.8)$$

(3) $(\overrightarrow{v}, \overrightarrow{v})^l = \prod_{i=1}^n ((\overrightarrow{u}_{\overrightarrow{A}_l}), (\overrightarrow{u}_{\overrightarrow{A}_l})) [(\overrightarrow{x}, \overrightarrow{x})(k-i+1)] ((\overrightarrow{u}_{\overrightarrow{b}_l}), (\overrightarrow{u}_{\overrightarrow{b}_l})) [(\overrightarrow{u}, \overrightarrow{u})(k)];$

(4) $(\overrightarrow{y}_k, \overrightarrow{y}_k) = \frac{\sum_{j=1}^k (\overrightarrow{h}_j, \overrightarrow{h}_j) (\overrightarrow{x}, \overrightarrow{x})_j}{\sum_{j=1}^k (\overrightarrow{h}_j, \overrightarrow{h}_j)};$

(5) $(\overrightarrow{e}_k, \overrightarrow{e}_k) = (\overrightarrow{y}_d, \overrightarrow{y}_d) - (\overrightarrow{y}_k, \overrightarrow{y}_k);$

(6) $(\overrightarrow{\phi}, \overrightarrow{\phi}) ((\overrightarrow{e}_k, \overrightarrow{e}_k), (\overrightarrow{c}_k, \overrightarrow{c}_k)) = 2 \cdot \exp(-(\overrightarrow{\lambda}_k, \overrightarrow{\lambda}_k)) |(\overrightarrow{e}_k, \overrightarrow{e}_k) - (\overrightarrow{c}_k, \overrightarrow{c}_k)|;$
// $(\overrightarrow{\lambda}_k, \overrightarrow{\lambda}_k)$ and $(\overrightarrow{c}_k, \overrightarrow{c}_k)$ are determined as in [17] //

(7) $(\overrightarrow{p}, \overrightarrow{p})(k) = \frac{1}{2} \sum_{i=1}^k [(\overrightarrow{e}_i, \overrightarrow{e}_i)]^2; // \text{Performance} //$

(8) **If** $|(\overrightarrow{e}_k, \overrightarrow{e}_k)| > (\overrightarrow{\epsilon}, \overrightarrow{\epsilon})|,$

// $(\overrightarrow{\epsilon}, \overrightarrow{\epsilon})$ is the maximum tolerance error, its value can be based on system requirements or experience. If the learning error is greater than the maximum tolerance, then the learning is invalid//

then Discard $(\overrightarrow{y}_k, \overrightarrow{y}_k)$, feedback to the rule base, notify the adjustment rules of the parameters, perform a parametric learning algorithm.

$(\overrightarrow{u}, \overrightarrow{u})(k) = (\overrightarrow{u}, \overrightarrow{u})(k) + \alpha (\overrightarrow{e}_k, \overrightarrow{e}_k) // \alpha$ is the gain learning coefficient

Continue from scratch

else if $|(\overrightarrow{e}_k, \overrightarrow{e}_k)| \leq (\overrightarrow{\delta}, \overrightarrow{\delta})|$

// $(\overrightarrow{\delta}, \overrightarrow{\delta})$ is an acceptable error, much smaller than $(\overrightarrow{\epsilon}, \overrightarrow{\epsilon})$, its value can be given according to system requirements or experience. If the error after learning is less than (or equal to) this acceptable level, it is considered to have reached the accuracy of the learning requirements //

then The learning process is finished, corrected, and output

$(\overrightarrow{y}_k, \overrightarrow{y}_k) = (\overrightarrow{y}_k, \overrightarrow{y}_k) + \beta (\overrightarrow{\phi}, \overrightarrow{\phi}) ((\overrightarrow{e}_k, \overrightarrow{e}_k), (\overrightarrow{c}_k, \overrightarrow{c}_k)) = (\overrightarrow{y}_k, \overrightarrow{y}_k) // \beta$ is the correction factor

else Feedback to the rule base, notify the adjustment rules in the parameters, and implement the parameter learning algorithm.

$$(\overrightarrow{u}, \overrightarrow{u})(k) = (\overrightarrow{u}, \overrightarrow{u})(k) + \alpha (\overrightarrow{e}_k, \overrightarrow{e}_k)$$

Continue from scratch;

end if

end if

Note: $(\overleftarrow{h}_j, \overrightarrow{h}_j)$ is the corresponding weight coefficient, and the membership function is generally taken as the weight coefficient.

B. Stability analysis

Denoting $(\overleftarrow{X}, \overrightarrow{X})(k)$ as $((\overleftarrow{x}, \overrightarrow{x})(k), (\overleftarrow{x}, \overrightarrow{x})(k-1), \dots, (\overleftarrow{x}, \overrightarrow{x})(k-n+1))^T \in (\overleftarrow{U}, \overrightarrow{U})$ and

$$(\overleftarrow{A}_p, \overrightarrow{A}_p) = \begin{bmatrix} (\overleftarrow{a}_1^p, \overrightarrow{a}_1^p) & (\overleftarrow{a}_2^p, \overrightarrow{a}_2^p) & \dots & (\overleftarrow{a}_{n-1}^p, \overrightarrow{a}_{n-1}^p) & (\overleftarrow{a}_n^p, \overrightarrow{a}_n^p) \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix},$$

(1.8) can be rewritten as

$$(\overleftarrow{X}, \overrightarrow{X})(k+1) = \frac{\sum_{p=1}^m (\overleftarrow{A}_p, \overrightarrow{A}_p) (\overleftarrow{X}, \overrightarrow{X})(k) (\overleftarrow{v}, \overrightarrow{v})^p}{\sum_{p=1}^m (\overleftarrow{v}, \overrightarrow{v})^p}. \quad (1.9)$$

Since the right side of (1.9) is equal to $(\overleftarrow{0}, \overrightarrow{0})$ when $(\overleftarrow{X}, \overrightarrow{X})(k) = (\overleftarrow{0}, \overrightarrow{0})$, the origin $(\overleftarrow{0}, \overrightarrow{0})$ in the domain is the equilibrium point of (1.9).

Theorem 1.3 The equilibrium point $(\overleftarrow{0}, \overrightarrow{0})$ of (1.9) of the DFMLS is globally asymptotically stable. If there exists a positive dynamic fuzzy matrix $(\overleftarrow{P}, \overrightarrow{P})$, then [18]

$$(\overleftarrow{A}_p, \overrightarrow{A}_p)^T (\overleftarrow{P}, \overrightarrow{P}) (\overleftarrow{A}_p, \overrightarrow{A}_p) - (\overleftarrow{P}, \overrightarrow{P}) < (\overleftarrow{0}, \overrightarrow{0}) \quad (p = 1, 2, \dots, n). \quad (1.10)$$

Proof: Consider the following Lyapunov function:

$$V[(\overleftarrow{X}, \overrightarrow{X})(k)] = (\overleftarrow{X}, \overrightarrow{X})^T(k) (\overleftarrow{P}, \overrightarrow{P}) (\overleftarrow{X}, \overrightarrow{X})(k),$$

where $(\overleftarrow{P}, \overrightarrow{P})$ is a positive dynamic fuzzy matrix. Then, there is

$$\begin{aligned} \Delta V[(\overleftarrow{X}, \overrightarrow{X})(k)] \\ = (\overleftarrow{X}, \overrightarrow{X})^T(k+1) (\overleftarrow{P}, \overrightarrow{P}) (\overleftarrow{X}, \overrightarrow{X})(k+1) - (\overleftarrow{X}, \overrightarrow{X})^T(k) (\overleftarrow{P}, \overrightarrow{P}) (\overleftarrow{X}, \overrightarrow{X})(k) \\ = (\overleftarrow{X}, \overrightarrow{X})^T(k) \left[\left(\frac{\sum_{p=1}^m (\overleftarrow{A}_p, \overrightarrow{A}_p)^T (\overleftarrow{v}, \overrightarrow{v})^p}{\sum_{p=1}^m (\overleftarrow{v}, \overrightarrow{v})^p} \right) (\overleftarrow{P}, \overrightarrow{P}) \left(\frac{\sum_{p=1}^m (\overleftarrow{A}_p, \overrightarrow{A}_p) (\overleftarrow{v}, \overrightarrow{v})^p}{\sum_{p=1}^m (\overleftarrow{v}, \overrightarrow{v})^p} \right) - (\overleftarrow{P}, \overrightarrow{P}) \right] \\ = (\overleftarrow{X}, \overrightarrow{X})(k) \end{aligned}$$

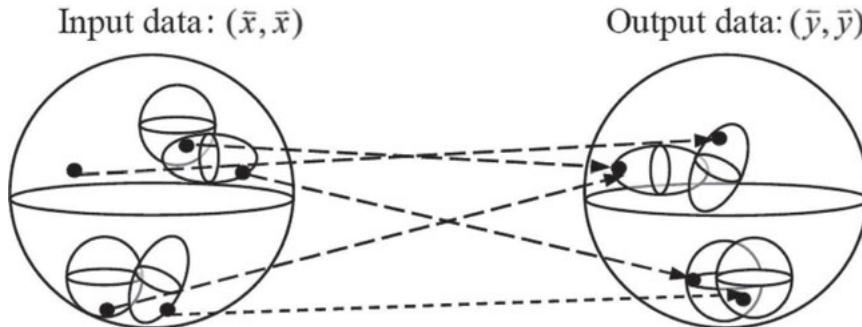


Fig. 1.3: Geometric model of dynamic fuzzy machine learning system.

$$= \frac{\sum_{p=1}^n \sum_{q=1}^n \left\{ (\bar{v}, \bar{v})^p (\bar{v}, \bar{v})^q (\bar{X}, \bar{X})^T(k) [(\bar{A}_p, \bar{A}_p)^T (\bar{P}, \bar{P}) (\bar{A}_q, \bar{A}_q)^T (\bar{P}, \bar{P})] (\bar{X}, \bar{X})(k) \right\}}{\sum_{p=1}^n \sum_{q=1}^n (\bar{v}, \bar{v})^p (\bar{v}, \bar{v})^q}.$$

$\Delta V[(\bar{X}, \bar{X})(k)] < (\bar{0}, \bar{0})$ is obtained from (1.10) and $(\bar{v}, \bar{v})^p \geq (\bar{0}, \bar{0})$. Using the Lyapunov stability theorem, the proof is complete.

1.2.3 DFML geometric model description

1.2.3.1 Geometric model of DFMLS [8, 11]

As shown in Fig. 1.3, we define the universe as a dynamic fuzzy sphere (large sphere), in which some small spheres represent the DFSs in the universe. Each dynamic fuzzy number is defined as a point in the DFS (small sphere). The membership degree of each dynamic fuzzy number is determined by the position and radius of the DFS (small sphere) in the domain (large sphere) and the position in the DFS (small sphere).

1.2.3.2 Geometric model of DFML algorithm

In Fig. 1.4, the centre of the two balls is the expected value (\bar{y}_d, \bar{y}_d) , the radius of the large sphere is $(\bar{\varepsilon}, \bar{\varepsilon})$, and the radius of the sphere is $(\bar{\delta}, \bar{\delta})$ [here, $(\bar{\varepsilon}, \bar{\varepsilon})$ and $(\bar{\delta}, \bar{\delta})$ are the same as in Algorithm 1.2].

The geometry model can be described as follows:

- (1) If the value (\bar{y}_k, \bar{y}_k) of the learning algorithm falls outside the ball, then this learning is invalid. Discard (\bar{y}_k, \bar{y}_k) and feed the information back to the rules of the system library and learning part, then begin the process again;
- (2) If the value (\bar{y}_k, \bar{y}_k) of the learning algorithm falls between the big sphere and the small sphere, the information is fed back to the system rules and learning part of the library so that they can be amended. Proceed to the next step;

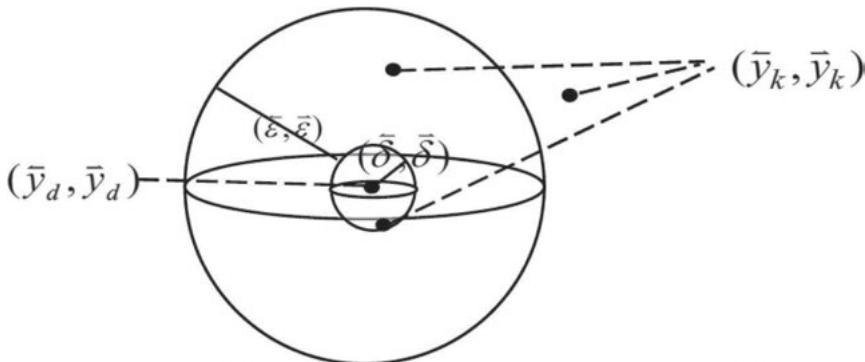


Fig. 1.4: Geometric model of dynamic fuzzy machine learning algorithm.

- (3) If the value (\bar{y}_k, \bar{y}_k) of the learning algorithm falls within the small sphere, it is considered that the precision requirement has been reached. Terminate the learning process and output (\bar{y}_k, \bar{y}_k) .

1.2.4 Simulation examples

Example 1.1 The sunspot problem: Data from 1749–1924 were extracted from [19]. This gave a total of 176 data, of which the first 100 were used as training samples and the remainder were used as test data.

Figure 1.5 shows the error and iterative steps of the algorithm described in this section, the neural network elastic Back Propagation (BP) algorithm resilient propagation (RPROP) [20], and a Q-learning algorithm. When the error $(\overrightarrow{e}_k, \overrightarrow{e}_k) = (0.4, 0.4)$, RPROP falls into a local minimum and requires approximately 1660 iterations to find the optimum solution; in the Q-learning algorithm, 1080 iterations are needed to reach an error of $(\overrightarrow{e}_k, \overrightarrow{e}_k) = (0.334, 0.334)$; for our algorithm, the number of iterations required for an error of $(\overrightarrow{e}_k, \overrightarrow{e}_k) < (0.4, 0.4)$ is 455. When the error $(\overrightarrow{e}_k, \overrightarrow{e}_k) = (0.034, 0.034)$, the training is basically stable, and the required number of iterations is about 1050. Figure 1.6 compares the actual values with the predicted values obtained by the algorithm presented in this section for an initial value $(\overrightarrow{u}_0, \overrightarrow{u}_0)(t) = (0, 0)$, gain learning coefficient of $\alpha = 0.3$, correction coefficient of $\beta = 0.2$, maximum tolerance error of $(\overrightarrow{e}, \overrightarrow{e}) = (0.5, 0.5)$, acceptable error of $(\overrightarrow{\delta}, \overrightarrow{\delta}) = (0.005, 0.005)$, and error $(\overrightarrow{e}_k, \overrightarrow{e}_k) = (0.034, 0.034)$.

Example 1.2 Time series forecast of daily closing price of a company over a period of time.

The data used in this example are again from [19]. Of the 250 data used, the first 150 data were taken as training samples, and the remaining 100 were used as test data.

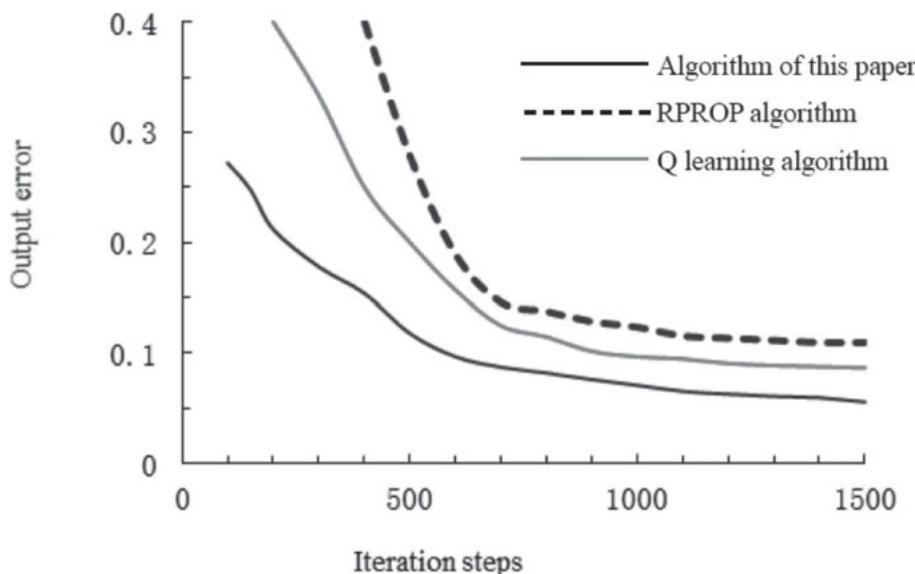


Fig. 1.5: Comparison of errors of three algorithms and iterative steps.

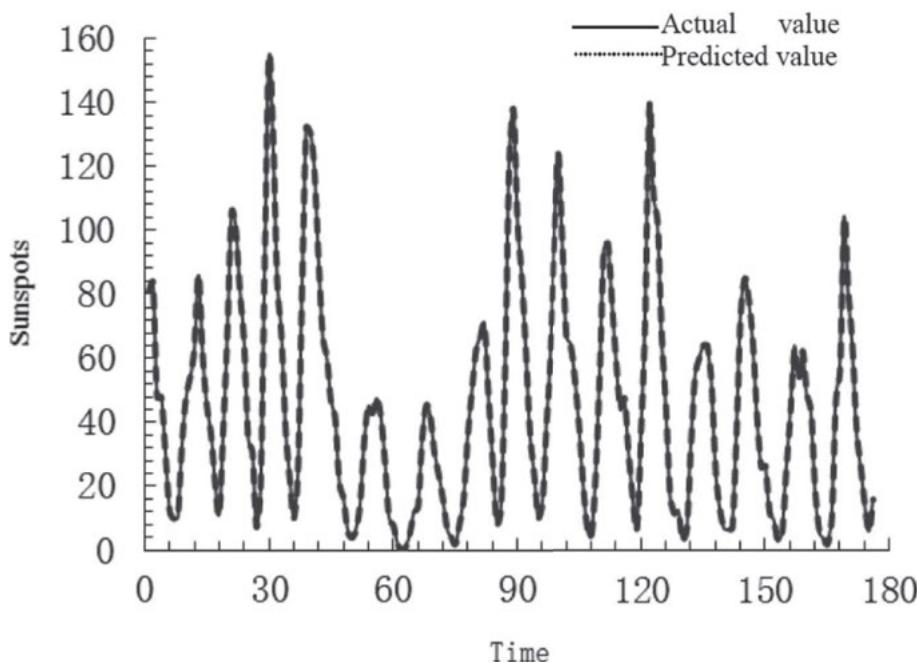


Fig. 1.6: Comparison between predictive value and actual value of this algorithm value.

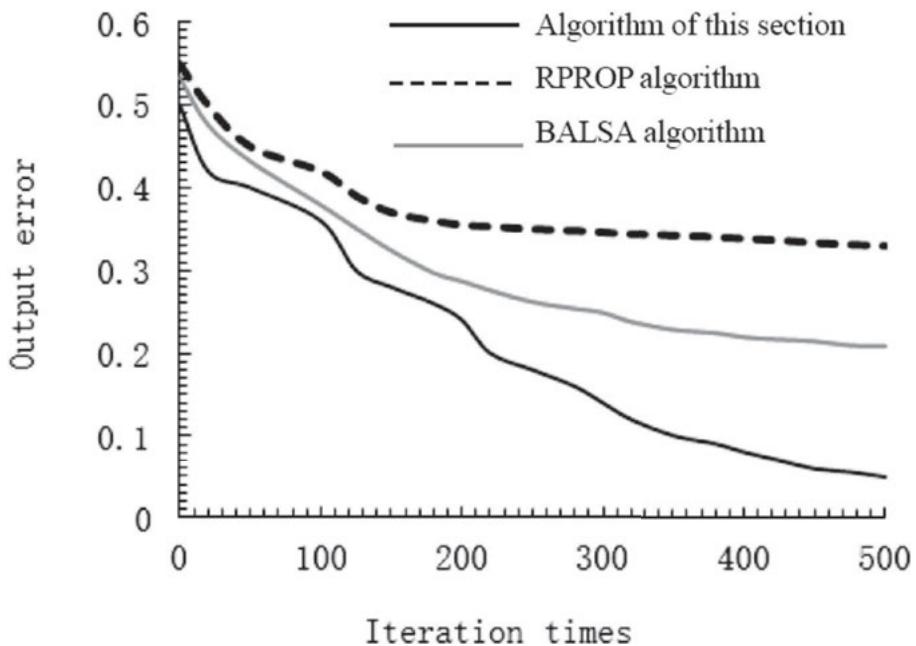


Fig. 1.7: Comparison between errors of three algorithms and iterative steps.

We set the initial value to $(\overleftarrow{u}_0, \overrightarrow{u}_0)(t) = (\overleftarrow{0}, \overrightarrow{0})$, gain learning coefficient to $\alpha = 0.3$, correction coefficient to $\beta = 0.3$, maximum tolerance error to $(\overleftarrow{\varepsilon}, \overrightarrow{\varepsilon}) = (0.5, 0.5)$, and acceptable error to $(\overleftarrow{\delta}, \overrightarrow{\delta}) = (0.005, 0.005)$.

Figure 1.7 shows the error and iterative steps of this algorithm compared with the elastic BP algorithm RPROP and the BALSA algorithm, which is based on a Bayesian algorithm [21]. When the performance index is $(\overleftarrow{p}, \overrightarrow{p})(k) = (0.013, 0.013)$, RPROP falls into a local minimum after approximately 146 iterations; for the BALSA algorithm, when the performance index is $(\overleftarrow{p}, \overrightarrow{p})(k) = (0.013, 0.013)$, the number of iterations required is 114; for our algorithm, when the performance index is less than $(0.013, 0.013)$, the number of iterations required is 82. The number of iterations required to satisfy the accuracy requirement is about 500. Figure 1.8 compares the actual values and the learning results of the proposed algorithm, where k is the number of iterations.

1.3 Relative algorithm of DFMLS [5]

1.3.1 Parameter learning algorithm for DFMLS

1.3.1.1 Problem statement

According to Algorithm 1.2, DFMLS adjusts and modifies the rules in the rule base according to the results of each learning process. The adjustment and modification

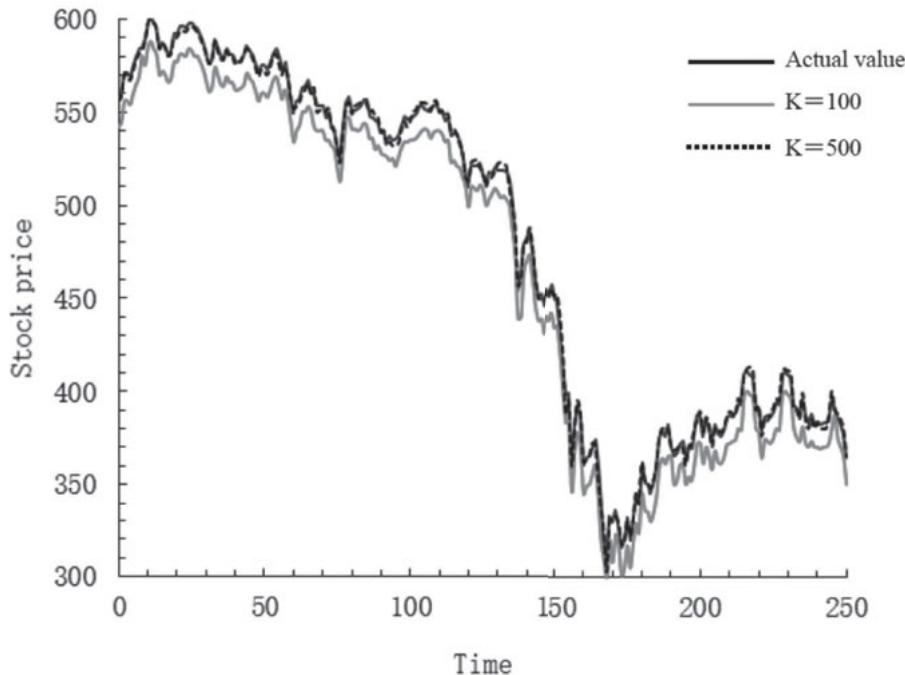


Fig. 1.8: Comparison between actual value and learning result.

of rules are mainly reflected in the adjustment of parameters in the rules. For this problem, this section derives a DFML algorithm that identifies the optimal system parameters.

Consider the DFMLS described in the previous section, which is formalized as $(\vec{y}, \vec{y}) = f((\vec{X}, \vec{X}), (\vec{\theta}, \vec{\theta}))$. Each element in $(\vec{\theta}, \vec{\theta})$ is

$$(\vec{\theta}_i^l, \vec{\theta}_i^l) = [(\vec{m}, \vec{m})_{(\vec{A}_i^l, \vec{A}_i^l)}, (\vec{\delta}, \vec{\delta})_{(\vec{A}_i^l, \vec{A}_i^l)}, (\vec{\delta}, \vec{\delta})_{(\vec{b}_l, \vec{b}_l)}]^T$$

$$(i = 1, 2, \dots, n; l = 1, 2, \dots, m),$$

where $(\vec{m}, \vec{m})_{(\vec{A}_i^l, \vec{A}_i^l)}$, $(\vec{\delta}, \vec{\delta})_{(\vec{A}_i^l, \vec{A}_i^l)}$ and $(\vec{\delta}, \vec{\delta})_{(\vec{b}_l, \vec{b}_l)}$ are the mean and variance of the corresponding membership functions.

According to the given input and output data pairs $((\vec{X}_k, \vec{X}_k), (\vec{y}_k, \vec{y}_k)) (k = 1, 2, \dots, N)$, the system parameters can be learnt using the least-squares error (LSE) objective function, which minimizes the output error of the system:

$$E_{LSE} = \frac{1}{2} \sum_{k=1}^N [(\vec{y}_k, \vec{y}_k) - f((\vec{X}_k, \vec{X}_k), (\vec{\theta}, \vec{\theta}))]^2 = \frac{1}{2} \sum_{k=1}^N (\vec{e}_k, \vec{e}_k)^2, \quad (1.11)$$

and modifies the parameters along the direction of steepest gradient descent:

$$\Delta(\overleftarrow{\theta}, \overrightarrow{\theta}) = -\eta \frac{\partial E_{LSE}}{\partial(\overleftarrow{\theta}, \overrightarrow{\theta})} = -\eta \sum_{k=1}^N \frac{\partial E_{LSE}}{\partial(\overleftarrow{e}_k, \overrightarrow{e}_k)} \frac{\partial(\overleftarrow{e}_k, \overrightarrow{e}_k)}{\partial(\overleftarrow{\theta}, \overrightarrow{\theta})}, \quad (1.12)$$

where η is the training step length. Thus, the iterative optimization equation of parameter $(\overleftarrow{m}, \overrightarrow{m})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}$, $(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}$ and $(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{b}_l, \overrightarrow{b}_l)}$ can be obtained so as to minimize the sum of squares of the error in (1.11).

There are two issues worth considering:

- (1) The choice of step size: If only a single fixed training step is used, it is difficult to take into account the convergence of different error variations, sometimes resulting in oscillations in the learning of the parameters, especially near the minimum point. Fixed training steps will reduce the convergence speed. In practical applications, there is often no universal, fixed training step for different parameter learning problems. Therefore, the following optimization steps are proposed to solve this problem [22]:

$$\begin{aligned} \eta(t) &= [\eta_{(\overleftarrow{m}, \overrightarrow{m})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}}(t), \eta_{(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}}(t), \eta_{(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{b}_l, \overrightarrow{b}_l)}}(t)]^T \\ \eta_{(\overleftarrow{m}, \overrightarrow{m})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}}(t) &\approx \frac{1}{\sum_{l=1}^m \sum_{i=1}^n \left\{ \frac{\partial f((\overleftarrow{\theta}_i, \overrightarrow{\theta}_i), (\overrightarrow{X}_i, \overrightarrow{X}_i))}{\partial(\overleftarrow{m}, \overrightarrow{m})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}^{(t)}} \right\}} \\ \eta_{(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{b}_l, \overrightarrow{b}_l)}}(t) &\approx \frac{1}{\sum_{l=1}^m \sum_{i=1}^n \left\{ \frac{\partial f((\overleftarrow{\theta}_i, \overrightarrow{\theta}_i), (\overrightarrow{X}_i, \overrightarrow{X}_i))}{\partial(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{b}_l, \overrightarrow{b}_l)}^{(t)}} \right\}} \\ \eta_{(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}}(t) &\approx \frac{1}{\sum_{l=1}^m \sum_{i=1}^n \left\{ \frac{\partial f((\overleftarrow{\theta}_i, \overrightarrow{\theta}_i), (\overrightarrow{X}_i, \overrightarrow{X}_i))}{\partial(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}^{(t)}} \right\}} \end{aligned}$$

- (2) When the input and output data contain measurement noise, the learning parameter cannot converge to the true value $(\overleftarrow{\theta}, \overrightarrow{\theta})^*$ using the LSE objective function.

When the input and output data contain noise, the input and output data pairs can be expressed as:

$$\begin{cases} (\overrightarrow{X}_k, \overrightarrow{X}_k) = (\overrightarrow{X}_k, \overrightarrow{X}_k)^* + n_{(\overrightarrow{X}_k, \overrightarrow{X}_k)} \\ (\overrightarrow{y}_k, \overrightarrow{y}_k) = (\overrightarrow{y}_k, \overrightarrow{y}_k)^* + n_{(\overrightarrow{y}_k, \overrightarrow{y}_k)} \end{cases} \quad (1.13)$$

where $(\overrightarrow{X}_k, \overrightarrow{X}_k)^*$, $(\overrightarrow{y}_k, \overrightarrow{y}_k)^*$ are the true values of the input and output data, respectively, $n_{(\overrightarrow{X}_k, \overrightarrow{X}_k)}$, $n_{(\overrightarrow{y}_k, \overrightarrow{y}_k)}$ are the input and output data contained in the noise, respectively, and the variance is $\{(\overrightarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{x}_k^i, \overrightarrow{x}_k^i)}, i = 1, 2, \dots, n\}$, $(\overrightarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{y}_k, \overrightarrow{y}_k)}$.

When learning the parameter of a DFMLS that includes input/output noise, the error cost function [30] is defined as

$$\begin{aligned} E_{EIV} &= \frac{1}{2} \sum_{k=1}^N \left[\frac{((\overrightarrow{y}_k, \overrightarrow{y}_k) - f((\overrightarrow{\theta}, \overrightarrow{\theta}), (\overrightarrow{X}_k, \overrightarrow{X}_k)'))^2}{(\overrightarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{y}_k, \overrightarrow{y}_k)}^2} + \sum_{i=1}^n \frac{((\overrightarrow{x}_k^i, \overrightarrow{x}_k^i) - (\overrightarrow{x}_k^i, \overrightarrow{x}_k^i)')^2}{(\overrightarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{x}_k^i, \overrightarrow{x}_k^i)}^2} \right], \\ &= \frac{1}{2} \sum_{k=1}^N \left(\frac{(\overrightarrow{e}, \overrightarrow{e})_{(\overrightarrow{y}_k, \overrightarrow{y}_k)}^2}{(\overrightarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{y}_k, \overrightarrow{y}_k)}^2} + \sum_{i=1}^n \frac{(\overrightarrow{e}, \overrightarrow{e})_{(\overrightarrow{x}_k^i, \overrightarrow{x}_k^i)}^2}{(\overrightarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{x}_k^i, \overrightarrow{x}_k^i)}^2} \right) \end{aligned} \quad (1.14)$$

where $(\overrightarrow{X}_k, \overrightarrow{X}_k)'$ is the estimate of the true value $(\overrightarrow{X}_k, \overrightarrow{X}_k)^*$.

Theorem 1.4 When the input and output data contain noise, the learning parameter $(\overrightarrow{\theta}, \overrightarrow{\theta})$ converges strongly to the true value $(\overrightarrow{\theta}, \overrightarrow{\theta})^*$ of the parameters using the error cost function of (1.14), which is

$$\lim_{N \rightarrow \infty} \left[\arg \min_{(\overrightarrow{\theta}, \overrightarrow{\theta})} (E_{EIV}) \right] = (\overrightarrow{\theta}, \overrightarrow{\theta})^*.$$

Proof: The proof is easy using the proof of Lemma 2 in [23].

1.3.1.2 Parameter learning algorithm

According to the analysis in the previous section, we know that the parameters $(\overrightarrow{\theta}_i^l, \overrightarrow{\theta}_i^l) = [(\overrightarrow{m}, \overrightarrow{m})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}, (\overrightarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}, (\overrightarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{b}_i, \overrightarrow{b}_i)}]^T$ that need to be learned in the DFMLS are further modified according to the iterative equation of the parameter-learning algorithm described in (1.12) and the adaptive training step provided by (1.13). These give the corresponding parameter Learning Algorithm Description:

$$(\overrightarrow{m}, \overrightarrow{m})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}(t+1) = (\overrightarrow{m}, \overrightarrow{m})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}(t) + \frac{\eta_{(\overrightarrow{m}, \overrightarrow{m})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}(t)}}{\sum_{l=1}^m \sum_{i=1}^n \left\{ \frac{\partial f((\overrightarrow{\theta}, \overrightarrow{\theta}), (\overrightarrow{X}_k, \overrightarrow{X}_k))}{\partial (\overrightarrow{m}, \overrightarrow{m})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}(t)} \right\}_2}.$$

$$\sum_{k=1}^N (\overrightarrow{e}_k, \overrightarrow{e}_k)(t) \cdot \frac{\partial f((\overrightarrow{\theta}, \overrightarrow{\theta}), (\overrightarrow{X}_k, \overrightarrow{X}_k))}{\partial (\overrightarrow{m}, \overrightarrow{m})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}(t)},$$

$$(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}(t+1) = (\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}(t) + \frac{\eta_{(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}}(t)}{\sum_{l=1}^m \sum_{i=1}^n \left\{ \frac{\partial f((\overleftarrow{\theta}, \overrightarrow{\theta}), (\overleftarrow{X}_k, \overrightarrow{X}_k))}{\partial (\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}(t)} \right\}_2}. \quad (1.16)$$

$$\sum_{k=1}^N (\overleftarrow{e}_k, \overrightarrow{e}_k)(t) \cdot \frac{\partial f((\overleftarrow{\theta}, \overrightarrow{\theta}), (\overleftarrow{X}_k, \overrightarrow{X}_k))}{\partial (\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{A}_i^l, \overrightarrow{A}_i^l)}(t)}, \text{ and}$$

$$(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{b}_l, \overrightarrow{b}_l)}(t+1) = (\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{b}_l, \overrightarrow{b}_l)}(t) + \frac{\eta_{(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{b}_l, \overrightarrow{b}_l)}}(t)}{\sum_{l=1}^m \sum_{i=1}^n \left\{ \frac{\partial f((\overleftarrow{\theta}, \overrightarrow{\theta}), (\overleftarrow{X}_k, \overrightarrow{X}_k))}{\partial (\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{b}_l, \overrightarrow{b}_l)}(t)} \right\}_2}. \quad (1.17)$$

$$\sum_{k=1}^N (\overleftarrow{e}_k, \overrightarrow{e}_k)(t) \cdot \frac{\partial f((\overleftarrow{\theta}, \overrightarrow{\theta}), (\overleftarrow{X}_k, \overrightarrow{X}_k))}{\partial (\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{b}_l, \overrightarrow{b}_l)}(t)}.$$

For the noise variance $(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{x}_k^i, \overrightarrow{x}_k^i)}, (\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{y}_k, \overrightarrow{y}_k)}$ included in the input and output data of the error cost function and the estimated value $(\overleftarrow{X}_k, \overrightarrow{X}_k)'$ of the true value $(\overleftarrow{X}_k, \overrightarrow{X}_k)^*$, we use the direction of steepest gradient descent correction parameters to obtain the learning algorithm:

$$\Delta(\overleftarrow{x}_k^i, \overrightarrow{x}_k^i)' = -\eta(t) \left[\frac{(\overleftarrow{e}, \overrightarrow{e})_{(\overleftarrow{y}_k, \overrightarrow{y}_k)} \cdot \frac{\partial f((\overleftarrow{\theta}, \overrightarrow{\theta}), (\overleftarrow{X}_k, \overrightarrow{X}_k)')}{\partial (\overleftarrow{x}_k^i, \overrightarrow{x}_k^i)'}}{(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{y}_k, \overrightarrow{y}_k)}^2} + \frac{(\overleftarrow{e}, \overrightarrow{e})_{(\overleftarrow{x}_k^i, \overrightarrow{x}_k^i)}}{(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{x}_k^i, \overrightarrow{x}_k^i)}^2} \right], \quad (1.18)$$

$$\Delta(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{x}_k^i, \overrightarrow{x}_k^i)} = -\eta(t) (\overleftarrow{e}, \overrightarrow{e})_{(\overleftarrow{x}_k^i, \overrightarrow{x}_k^i)}^2 / (\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{x}_k^i, \overrightarrow{x}_k^i)}^3, \text{ and} \quad (1.19)$$

$$\Delta(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{y}_k, \overrightarrow{y}_k)} = -\eta(t) (\overleftarrow{e}, \overrightarrow{e})_{(\overleftarrow{y}_k, \overrightarrow{y}_k)}^2 / (\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{y}_k, \overrightarrow{y}_k)}^3. \quad (1.20)$$

Equations (1.15)–(1.20) constitute the parameter-learning algorithm of the whole DFMLS.

1.3.1.3 Examples

To verify the validity of the parameter-learning algorithm for the DFMLS proposed in this section, and to illustrate the superiority of this algorithm, we simulate the

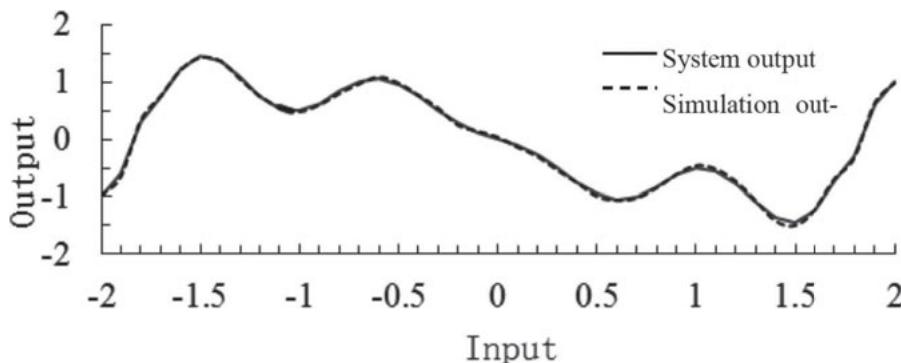


Fig. 1.9: Simulation results of the adaptive algorithm.

nonlinear function proposed in [22]. Finally, the algorithm is compared with the algorithm in [22]:

$$(\vec{y}, \vec{y})(\vec{x}, \vec{x}) = -\sin\left(\frac{\pi}{2}(\vec{x}, \vec{x})\right) + 0.5(\vec{x}, \vec{x}) \cdot \cos(2\pi \cdot (\vec{x}, \vec{x})) + (\vec{v}, \vec{v})_n,$$

where $(\vec{x}, \vec{x}) \in [(-\vec{2}, -\vec{2}), (\vec{2}, \vec{2})]$, and $(\vec{v}, \vec{v})_n$ is random interference noise with mean $(\vec{0}, \vec{0})$ and mean square error $(0.032, 0.032)$.

A total of 200 input and output points $((\vec{X}_k, \vec{X}_k), (\vec{y}_k, \vec{y}_k))$ were uniformly selected as training samples. The noise variance of the input and output samples was set to $(\vec{\delta}, \vec{\delta})^2 = (0.25, 0.25)$, and the subtraction clustering algorithm [24] was used to cluster the sample data. Thirty-two dynamic fuzzy logic (DFL) rules were extracted, and the corresponding initial parameters were determined. Using the adaptive parameter-learning algorithm proposed in [22] and the method proposed in this section, the algorithm trained 100 steps for the relevant parameters, where $\eta_{(\vec{m}, \vec{m})_{(\vec{A}_i^l, \vec{A}_i^l)}}(0)$, $\eta_{(\vec{\delta}, \vec{\delta})_{(\vec{A}_i^l, \vec{A}_i^l)}}(0)$, $\eta_{(\vec{\delta}, \vec{\delta})_{(\vec{b}_i^l, \vec{b}_i^l)}}(0)$ was set to 0.01. The results are shown in Figs. 1.9 and 1.10. Figure 1.9 shows the simulated output of the adaptive parameter-learning algorithm, which has a root mean square error of $RMSE = (0.0560\vec{9}, 0.0560\vec{9})$; Fig. 1.10 shows the parametric-learning algorithm described in this section, which has a root mean square error of $RMSE = (0.0417\vec{3}, 0.0417\vec{3})$. Obviously, our algorithm is superior.

1.3.2 Maximum likelihood estimation algorithm in DFMLS

Incomplete observations and unpredictable data losses in the process of learning mean that many real systems have a lot of missing data. Thus, studies on incomplete data have very important theoretical and practical value in terms of how to use the

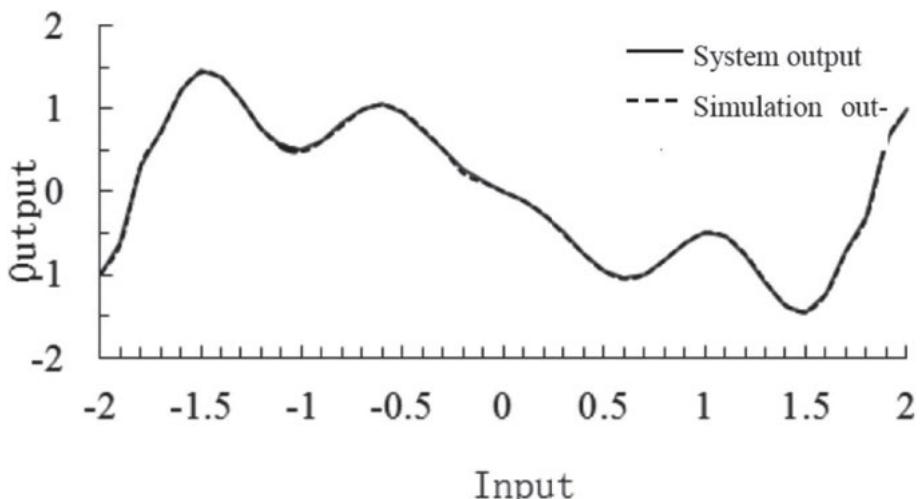


Fig. 1.10: Simulation results of the algorithm in this section.

observed data to estimate the missing data in a certain range. Earlier solutions include the Expectation-Maximization (EM) algorithm [25] proposed by Dempster, Laird, and Rubin in 1977; the Gibbs sampling proposed by S. Geman and D. Geman in 1984; and the Metropolis-Hastings method proposed by Metropolis in 1953 and improved by Hastings. The EM algorithm has been extended into a series of algorithms, such as the EM algorithm proposed by Meng and Rubin in 1993 and the Monte Carlo EM algorithm. The EM algorithm is an iterative procedure based on the maximum likelihood estimation (MLE) and is a powerful tool. The advantage of the EM algorithm is obvious, as it simplifies a problem through division, but its shortcomings cannot be ignored: its convergence rate is slow, and local convergence and pseudo-convergence phenomena [26] may also occur. The GEM algorithm and Monte Carlo EM algorithm offer various improvements, but the slow convergence rate has not been adequately dealt with. The α -EM algorithm for calculating the entropy rate in information theory can overcome this defect. Therefore, this paper proposes a dynamic fuzzy α -EM algorithm (DF α -EM) for the statistical inference problem of incomplete DFD in a DFMLS.

1.3.2.1 DF α -EM algorithm

Definition 1.9 Let $L^{(\alpha)}(r) = \frac{2}{1+\alpha}(r^{\frac{1+\alpha}{2}} - 1)$ be the α - logarithm, where $\alpha \in (-\infty, +\infty)$, $r \in (0, +\infty)$.

Theorem 1.5 For $\alpha \in (-\infty, +\infty)$, $r \in (0, +\infty)$, the α - logarithm has the following properties [27]:

$$(1) \quad L^{(-1)}(r) = lbr.$$

- (2) $L^{(\alpha)}(1) = 0$.
- (3) $L^{(\alpha)}(r)$ on r is monotonically increasing.
- (4) When $\alpha < 1$, $L^{(\alpha)}(r)$ is a strictly concave function; when $\alpha = 1$, $L^{(\alpha)}(r)$ is a linear function; when $\alpha > 1$, $L^{(\alpha)}(r)$ is a strictly convex function.
- (5) When $\alpha < \beta$, $L^{(\alpha)}(r) \leq L^{(\beta)}(r)$, and if and only if $r = 1$ equal sign.

The nonlinear DFMLM (see Definition 1.5) can be expressed as

$$\begin{cases} (\bar{y}_i, \bar{y}_i) \sim a((\bar{y}_i, \bar{y}_i), (\bar{\sigma}, \bar{\sigma})^2) \exp \left\{ -\frac{1}{2(\bar{\sigma}, \bar{\sigma})^2} d_i((\bar{y}_i, \bar{y}_i); (\bar{\mu}_i, \bar{\mu}_i)) \right\} \\ (\bar{\mu}_i, \bar{\mu}_i) = f((\bar{x}_i, \bar{x}_i), (\bar{\beta}, \bar{\beta})) = (\bar{\mu}_i, \bar{\mu}_i)(\bar{\beta}, \bar{\beta}) \quad (i = 1, 2, \dots, n), \end{cases} \quad (1.21)$$

where $((\bar{x}_i, \bar{x}_i)^T, (\bar{y}_i, \bar{y}_i)) (i = 1, 2, \dots, n)$ are independent observational data points; (\bar{y}_i, \bar{y}_i) is the response variable of the i th data point; $(\bar{x}_i, \bar{x}_i) = ((\bar{x}_{i1}, \bar{x}_{i1}), (\bar{x}_{i2}, \bar{x}_{i2}), \dots, (\bar{x}_{ik}, \bar{x}_{ik}))^T$ is the k th known explanatory variable vector of the i th data point; $a((\bar{y}_i, \bar{y}_i), (\bar{\sigma}, \bar{\sigma})^2) \geq 0$ and $a(\bullet, \bullet)$ is an appropriate function; $(\bar{\beta}, \bar{\beta}) = ((\bar{\beta}_1, \bar{\beta}_1), (\bar{\beta}_2, \bar{\beta}_2), \dots, (\bar{\beta}_p, \bar{\beta}_p))^T$ are regression coefficients; $(\bar{\mu}_i, \bar{\mu}_i)$ is the mean of (\bar{y}_i, \bar{y}_i) and belongs to the dynamic fuzzy open interval $(\bar{\Omega}, \bar{\Omega})$; $d_i(\bullet, \bullet)$ is the known deviation function defined on $C \times \Omega$ satisfying $d_i((\bar{y}_i, \bar{y}_i); (\bar{\mu}_i, \bar{\mu}_i)) \geq 0$, when $(\bar{\mu}_i, \bar{\mu}_i) = (\bar{y}_i, \bar{y}_i)$ the equality holds.

Assumptions: (1) Response variables are fully observed, with $(\bar{y}_i, \bar{y}_i) | (\bar{x}_i, \bar{x}_i) (i = 1, 2, \dots, n)$ independent mutual conditions. The conditional density function is then $p((\bar{y}_i, \bar{y}_i) | (\bar{x}_i, \bar{x}_i)) = a((\bar{y}_i, \bar{y}_i), (\bar{\sigma}, \bar{\sigma})^2) \exp \left\{ -\frac{1}{2(\bar{\sigma}, \bar{\sigma})^2} d_i((\bar{y}_i, \bar{y}_i); (\bar{\mu}_i, \bar{\mu}_i)) \right\} (i = 1, 2, \dots, n)$;

(2) The vector (\bar{x}_i, \bar{x}_i) , which is composed of k explanatory variables, is a discrete random vector whose distribution density function is $p((\bar{x}_i, \bar{x}_i) | (\bar{\gamma}, \bar{\gamma})) = ((\bar{\gamma}_1, \bar{\gamma}_1), (\bar{\gamma}_2, \bar{\gamma}_2), \dots, (\bar{\gamma}_r, \bar{\gamma}_r))$. Some of the explanatory variable values are missing in the partially observed data points, and this loss is random; $(\bar{x}_i, \bar{x}_i) (i = 1, 2, \dots, n)$ are independent of each other.

Set $(\bar{\theta}, \bar{\theta}) = ((\bar{\beta}, \bar{\beta})^T, (\bar{\gamma}, \bar{\gamma})^T)$. Then, $((\bar{x}_i, \bar{x}_i), (\bar{y}_i, \bar{y}_i))$ is the joint probability density $p((\bar{x}_i, \bar{x}_i), (\bar{y}_i, \bar{y}_i)) = p((\bar{y}_i, \bar{y}_i) | (\bar{x}_i, \bar{x}_i)) p((\bar{x}_i, \bar{x}_i) | (\bar{\gamma}, \bar{\gamma}))$. The joint α -log likelihood function of $((\bar{x}_i, \bar{x}_i), (\bar{y}_i, \bar{y}_i))$ is

$$\begin{aligned} L_i^{(\alpha)}((\bar{\theta}, \bar{\theta}); (\bar{x}_i, \bar{x}_i), (\bar{y}_i, \bar{y}_i)) &= L^{(\alpha)}(p((\bar{y}_i, \bar{y}_i) | (\bar{x}_i, \bar{x}_i))) \\ &\quad + L^{(\alpha)}(p((\bar{x}_i, \bar{x}_i) | (\bar{\gamma}, \bar{\gamma}))) \end{aligned}$$

if we define $L_{(\bar{y}_i, \bar{y}_i) | (\bar{x}_i, \bar{x}_i)}^{(\alpha)}(\bar{\beta}, \bar{\beta}) = L^{(\alpha)}(p((\bar{y}_i, \bar{y}_i) | (\bar{x}_i, \bar{x}_i)))$, then $L_{(\bar{x}_i, \bar{x}_i)}^{(\alpha)}(\bar{\gamma}, \bar{\gamma}) = L^{(\alpha)}(p((\bar{x}_i, \bar{x}_i) | (\bar{\gamma}, \bar{\gamma})))$ is the edge α -log likelihood function of (\bar{x}_i, \bar{x}_i) . Thus, the combined α -log likelihood function of $(\bar{y}, \bar{y}) = ((\bar{y}_1, \bar{y}_1),$

$(\overrightarrow{y}_2, \overrightarrow{y}_2), \dots, (\overrightarrow{y}_n, \overrightarrow{y}_n))^T$ and $(\overrightarrow{x}, \overrightarrow{x}) = ((\overrightarrow{x}_1, \overrightarrow{x}_1), (\overrightarrow{x}_2, \overrightarrow{x}_2), \dots, (\overrightarrow{x}_n, \overrightarrow{x}_n))^T$ can be expressed as

$$\begin{aligned} L_i^{(\alpha)}((\overrightarrow{\theta}, \overrightarrow{\theta}); (\overrightarrow{x}, \overrightarrow{x}), (\overrightarrow{y}, \overrightarrow{y})) &= \sum_{i=1}^n L_i^{(\alpha)}((\overrightarrow{\theta}, \overrightarrow{\theta}); (\overrightarrow{x}_i, \overrightarrow{x}_i), (\overrightarrow{y}_i, \overrightarrow{y}_i)) \\ &= \sum_{i=1}^n L_{(\overrightarrow{y}_i, \overrightarrow{y}_i) | (\overrightarrow{x}_i, \overrightarrow{x}_i)}^{(\alpha)}(\overrightarrow{\beta}, \overrightarrow{\beta}) + \sum_{i=1}^n L_{(\overrightarrow{x}_i, \overrightarrow{x}_i)}^{(\alpha)}(\overrightarrow{\gamma}, \overrightarrow{\gamma}). \end{aligned}$$

Set the complete data $(\overrightarrow{x}_i, \overrightarrow{x}_i) = ((\overrightarrow{x}_{obs,i}, \overrightarrow{x}_{obs,i}), (\overrightarrow{x}_{mis,i}, \overrightarrow{x}_{mis,i}))$, where $(\overrightarrow{x}_{obs,i}, \overrightarrow{x}_{obs,i})$ is the observed data of $(\overrightarrow{x}_i, \overrightarrow{x}_i)$, $(\overrightarrow{x}_{mis,i}, \overrightarrow{x}_{mis,i})$ is the missing data of $(\overrightarrow{x}_i, \overrightarrow{x}_i)$, $p((\overrightarrow{x}_{mis,i}, \overrightarrow{x}_{mis,i}) | (\overrightarrow{x}_{obs,i}, \overrightarrow{x}_{obs,i}), (\overrightarrow{y}_i, \overrightarrow{y}_i), (\overrightarrow{\theta}, \overrightarrow{\theta}))$ is the conditional distribution density function of the missing explanatory variable $(\overrightarrow{x}_{mis,i}, \overrightarrow{x}_{mis,i})$ under the condition of $(\overrightarrow{\theta}, \overrightarrow{\theta})$, and the observed data $(\overrightarrow{x}_{obs,i}, \overrightarrow{x}_{obs,i})$ are given.

Thus, the dynamic fuzzy MLE algorithm (dynamic fuzzy α -EM algorithm) is shown in Algorithm 1.3.

Algorithm 1.3 DF α -EM algorithm

Dynamic fuzzy α -E step:

$$\begin{aligned} Q^{(\alpha)}((\overrightarrow{\theta}, \overrightarrow{\theta}) | (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)}) &= \sum_{i=1}^n E(L_i^{(\alpha)}((\overrightarrow{\theta}, \overrightarrow{\theta}); (\overrightarrow{x}_i, \overrightarrow{x}_i), (\overrightarrow{y}_i, \overrightarrow{y}_i)) | (\overrightarrow{x}_{obs,i}, \overrightarrow{x}_{obs,i}), (\overrightarrow{y}_i, \overrightarrow{y}_i), (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)})) \\ &= \sum_{i=1}^n \sum_{(\overrightarrow{x}_{mis,i}, \overrightarrow{x}_{mis,i})} p((\overrightarrow{x}_{mis,i}, \overrightarrow{x}_{mis,i}) | (\overrightarrow{x}_{obs,i}, \overrightarrow{x}_{obs,i}), (\overrightarrow{y}_i, \overrightarrow{y}_i), (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)}) a \\ &\quad \times L_i^{(\alpha)}((\overrightarrow{\theta}, \overrightarrow{\theta}); (\overrightarrow{x}_i, \overrightarrow{x}_i), (\overrightarrow{y}_i, \overrightarrow{y}_i)) \\ &= \sum_{i=1}^n \sum_{(\overrightarrow{x}_{mis,i}, \overrightarrow{x}_{mis,i})} p((\overrightarrow{x}_{mis,i}, \overrightarrow{x}_{mis,i}) | (\overrightarrow{x}_{obs,i}, \overrightarrow{x}_{obs,i}), (\overrightarrow{y}_i, \overrightarrow{y}_i), (\overrightarrow{\theta}, \overrightarrow{\theta})) \\ &\quad \times L_{(\overrightarrow{y}_i, \overrightarrow{y}_i) | (\overrightarrow{x}_i, \overrightarrow{x}_i)}^{(\alpha)}(\overrightarrow{\beta}, \overrightarrow{\beta}) \\ &\quad + \sum_{i=1}^n \sum_{(\overrightarrow{x}_{mis,i}, \overrightarrow{x}_{mis,i})} p((\overrightarrow{x}_{mis,i}, \overrightarrow{x}_{mis,i}) | (\overrightarrow{x}_{obs,i}, \overrightarrow{x}_{obs,i}), (\overrightarrow{y}_i, \overrightarrow{y}_i), (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)}) \times L_{(\overrightarrow{x}_i, \overrightarrow{x}_i)}^{(\alpha)}(\overrightarrow{\gamma}, \overrightarrow{\gamma}) \end{aligned} \tag{1.22}$$

where $(\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)}$ denotes the iteration value of $(\overrightarrow{\theta}, \overrightarrow{\theta})$ obtained by the m th dynamic fuzzy α -EM iteration, E represents the expectation with respect to the distribution $p((\overrightarrow{x}_{mis,i}, \overrightarrow{x}_{mis,i}) | (\overrightarrow{x}_{obs,i}, \overrightarrow{x}_{obs,i}), (\overrightarrow{y}_i, \overrightarrow{y}_i), (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)})$, $\sum_{(\overrightarrow{x}_{mis,i}, \overrightarrow{x}_{mis,i})}$ represents the sum of all possible values of the missing explanatory variable $(\overrightarrow{x}_{mis,i}, \overrightarrow{x}_{mis,i})$, and the conditional distribution density function for the missing explanatory variable $(\overrightarrow{x}_{mis,i}, \overrightarrow{x}_{mis,i})$ under the condition of $(\overrightarrow{\theta}, \overrightarrow{\theta})^m$ and the observed data $(\overrightarrow{x}_{obs,i}, \overrightarrow{x}_{obs,i})$ are given. The Bayesian formula can be directly applied to get:

$$\begin{aligned}
& p((\overrightarrow{x}_{mis,i}, \overrightarrow{x}_{mis,i}) | (\overrightarrow{x}_{obs,i}, \overrightarrow{x}_{obs,i}), (\overrightarrow{y}_i, \overrightarrow{y}_i), (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)}) \\
&= \frac{p((\overrightarrow{y}_i, \overrightarrow{y}_i) | (\overrightarrow{x}_i, \overrightarrow{x}_i), (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)}) p((\overrightarrow{x}_i, \overrightarrow{x}_i) | (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)})}{\sum_{(\overrightarrow{x}_{mis,i}, \overrightarrow{x}_{mis,i})} p((\overrightarrow{y}_i, \overrightarrow{y}_i) | (\overrightarrow{x}_i, \overrightarrow{x}_i), (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)}) p((\overrightarrow{x}_i, \overrightarrow{x}_i) | (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)})} p((\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)})
\end{aligned}$$

Dynamic fuzzy α -E step:

Find $(\overrightarrow{\theta}, \overrightarrow{\theta})^{(m+1)} = ((\overrightarrow{\beta}, \overrightarrow{\beta})^{(m+1)T}, (\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(m+1)T})$, maximize $Q^{(\alpha)}((\overrightarrow{\theta}, \overrightarrow{\theta}) | (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)})$, i.e.

$$(\overrightarrow{\theta}, \overrightarrow{\theta})^{(m+1)} = \arg \max Q^{(\alpha)}((\overrightarrow{\theta}, \overrightarrow{\theta}) | (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)}),$$

and set the sum of the two terms on the right of (1.22) as $Q_1^{(\alpha)}((\overrightarrow{\beta}, \overrightarrow{\beta}) | (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)})$, $Q_2^{(\alpha)}((\overrightarrow{\gamma}, \overrightarrow{\gamma}) | (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)})$, respectively. Because $Q_1^{(\alpha)}((\overrightarrow{\beta}, \overrightarrow{\beta}) | (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)})$, $Q_2^{(\alpha)}((\overrightarrow{\gamma}, \overrightarrow{\gamma}) | (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)})$ are non-negative, the $(m+1)$ th dynamic ambiguity α -EM iteration can be transformed to:

$$\begin{aligned}
& (\overrightarrow{\beta}, \overrightarrow{\beta})^{(m+1)} = \arg \max Q_1^{(\alpha)}((\overrightarrow{\beta}, \overrightarrow{\beta}) | (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)}) \\
& (\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(m+1)} = \arg \max Q_2^{(\alpha)}((\overrightarrow{\gamma}, \overrightarrow{\gamma}) | (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)})
\end{aligned}$$

The method of solving $(\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(m+1)}$ is related to the distribution of the explanatory variables. The iterative formula for solving $(\overrightarrow{\beta}, \overrightarrow{\beta})^{(m+1)}$ is [32]:

$$(\overrightarrow{\beta}, \overrightarrow{\beta})^{(t+1)} = (\overrightarrow{\beta}, \overrightarrow{\beta})^{(t)} + (\tilde{D}^T \tilde{G} W^{(m)} \tilde{D})^{-1} \tilde{D}^T \tilde{G} W^{(m)} \tilde{e};$$

the definition of each variable is described in [32].

The above two steps are repeated until convergence.

1.3.2.2 Convergence of algorithms

We will describe the convergence behaviour of $L_{(\overrightarrow{x}_i, \overrightarrow{x}_i)}^{(\alpha)}((\overrightarrow{\gamma}, \overrightarrow{\gamma}))$; the convergence of $L_{(\overrightarrow{y}_i, \overrightarrow{y}_i) | (\overrightarrow{x}_i, \overrightarrow{x}_i)}^{(\alpha)}((\overrightarrow{\beta}, \overrightarrow{\beta}))$ is similar.

Theorem 1.6 $L_{(\overrightarrow{x}_i, \overrightarrow{x}_i)}^{(\alpha)}((\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(m+1)} | (\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(m)}) > 0 \Rightarrow p((\overrightarrow{x}_i, \overrightarrow{x}_i) | (\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(m+1)}) > p((\overrightarrow{x}_i, \overrightarrow{x}_i) | (\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(m)})$.

Theorem 1.7 $L_{(\overrightarrow{x}_i, \overrightarrow{x}_i)}^{(\alpha)}((\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(m)} | (\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(0)})$ on $(\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(m)}$ is monotonically increasing [28].

Theorem 1.8 Let $L_{(\overrightarrow{x}_i, \overrightarrow{x}_i)}^{(\alpha)}((\overrightarrow{\gamma}, \overrightarrow{\gamma}) | (\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(m)})$ and $Q_2^{(\alpha)}((\overrightarrow{\gamma}, \overrightarrow{\gamma}) | (\overrightarrow{\theta}, \overrightarrow{\theta})^{(m)})$ be continuous in $(\overrightarrow{R}, \overrightarrow{R}) \times (\overrightarrow{R}, \overrightarrow{R})$ and differentiable in its interior [29]. For any $p((\overrightarrow{x}_i, \overrightarrow{x}_i) | (\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(m)}) > 0$, $(\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(m)} \in (\overrightarrow{R}, \overrightarrow{R})$, if the set $(\overrightarrow{R}, \overrightarrow{R})_{(\overrightarrow{\gamma}^{(m)}, \overrightarrow{r}^{(m)})} = \{(\overrightarrow{\gamma}, \overrightarrow{\gamma}) \in (\overrightarrow{R}, \overrightarrow{R}) | L_{(\overrightarrow{x}_i, \overrightarrow{x}_i)}^{(\alpha)}((\overrightarrow{\gamma}, \overrightarrow{\gamma}) | (\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(m)}) \geq 0\}$ is dense and $(\overrightarrow{R}, \overrightarrow{R})_{(\overrightarrow{r}^{(m)}, \overrightarrow{r}^{(m)})}$ is inside $(\overrightarrow{R}, \overrightarrow{R})$, $L_{(\overrightarrow{x}_i, \overrightarrow{x}_i)}^{(\alpha)}((\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(m+1)} | (\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(m)}) > 0$ holds for all $(\overrightarrow{\gamma}, \overrightarrow{\gamma})^{(m)} \in (\overrightarrow{R}, \overrightarrow{R})$ and m . Then, we have the following conclusions:

- (1) The limit point of the sequence $\{(\bar{\gamma}, \bar{\gamma})^{(m)}\}$ is the fixed point of $L_{(\bar{x}_i, \bar{x}_i)}^{(\alpha)}((\bar{\gamma}, \bar{\gamma})|(\bar{\gamma}, \bar{\gamma})^{(0)})$ on $(\bar{\gamma}, \bar{\gamma})$.
- (2) $L_{(\bar{x}_i, \bar{x}_i)}^{(\alpha)}((\bar{\gamma}, \bar{\gamma})^{(m)}|(\bar{\gamma}, \bar{\gamma})^{(0)})$ converges monotonically to $L_{(\bar{x}_i, \bar{x}_i)}^{(\alpha)}((\bar{\gamma}, \bar{\gamma})^{(m+1)}|(\bar{\gamma}, \bar{\gamma})^{(0)})$, where $(\bar{\gamma}, \bar{\gamma})^{(m+1)} \in (\bar{S}, \bar{S})$.

Proof: Define the iterative operator:

Under the conditions of the theorem, the operator $\Lambda^{(\alpha)}$ is closed.

$(\bar{R}, \bar{R})_{(\bar{r}^{(m)}, \bar{r}^{(m)})}$ inside (\bar{R}, \bar{R}) guarantees that $Q_2^{(\alpha)}((\bar{\gamma}, \bar{\gamma})|(\bar{\theta}, \bar{\theta})^{(m)})$ and $L_{(\bar{x}_i, \bar{x}_i)}^{(\alpha)}((\bar{\gamma}, \bar{\gamma})|(\bar{\gamma}, \bar{\gamma})^{(m)})$ are differentiable for any $(\bar{\gamma}, \bar{\gamma}) \in (\bar{R}, \bar{R})$ such that the global convergence conditions in [30] and [31] are satisfied:

- (1) Sequence $\{(\bar{\gamma}, \bar{\gamma})^{(m)}\} (m \geq 0)$ is generated by $(\bar{\gamma}, \bar{\gamma})^{(m)} = \Lambda^{(\alpha)}((\bar{\gamma}, \bar{\gamma})^{(m-1)})$;
- (2) The fixed point set (\bar{S}, \bar{S}) of the iterative operator $\Lambda^{(\alpha)}$ is contained in a dense set $(\bar{R}, \bar{R})_{(\bar{r}^{(0)}, \bar{r}^{(0)})}$;
- (3) $L_{(\bar{x}_i, \bar{x}_i)}^{(\alpha)}((\bar{\gamma}, \bar{\gamma})|(\bar{\gamma}, \bar{\gamma})^{(m)})$ is continuous. Thus, for $(\bar{\gamma}, \bar{\gamma})^{(m)} \in (\bar{R}, \bar{R})$, there is

$$L_{(\bar{x}_i, \bar{x}_i)}^{(\alpha)}((\bar{\gamma}, \bar{\gamma})^{(m+1)}|(\bar{\gamma}, \bar{\gamma})^{(0)}) > L_{(\bar{x}_i, \bar{x}_i)}^{(\alpha)}((\bar{\gamma}, \bar{\gamma})^{(m)}|(\bar{\gamma}, \bar{\gamma})^{(0)}).$$

Therefore, by the global convergence theorem in [30] and [31], the proof is complete.

1.3.2.3 Examples

Table 1.1 provides complete data for 32 patients with haematological malignancy, from Ibrahim (1900). Assume that the 3rd, 8th, 19th, 27th, and 32nd data points are not fully observed, see Tab. 1.2, which is taken from [32]. In the table, t represents the patient's duration (weeks); $(\bar{x}_1, \bar{x}_1) = (\bar{1}, \bar{1})$ indicates that a biological characteristic test for leukocytes is positive, $(\bar{x}_1, \bar{x}_1) = (\bar{0}, \bar{0})$ indicates that the test was negative; $z = 0$ indicates that there are less than 11,000 white blood cells in the blood to be tested, $z = 1$ represents $11,000 \leq WBC \leq 40,000$, and $z = 2$ represents $WBC > 40,000$.

Variable z has three states. Therefore, two dummy dynamic fuzzy variables (\bar{x}_2, \bar{x}_2) and (\bar{x}_3, \bar{x}_3) are introduced. When $z = 0$, $((\bar{x}_2, \bar{x}_2), (\bar{x}_3, \bar{x}_3)) = ((\bar{0}, \bar{0}), (\bar{0}, \bar{0}))$; when $z = 1$, $((\bar{x}_2, \bar{x}_2), (\bar{x}_3, \bar{x}_3)) = ((\bar{1}, \bar{1}), (\bar{0}, \bar{0}))$; when $z = 2$, $((\bar{x}_2, \bar{x}_2), (\bar{x}_3, \bar{x}_3)) = ((\bar{0}, \bar{0}), (\bar{1}, \bar{1}))$.

Assuming that (\bar{y}, \bar{y}) follows the parameter $(\bar{\sigma}, \bar{\sigma})^2 = (\bar{1}, \bar{1})$ and the logarithmic gamma distribution of the mean $(\bar{\mu}, \bar{\mu}) = ((\bar{\beta}_0, \bar{\beta}_0) + (\bar{\beta}_1, \bar{\beta}_1)(\bar{x}_1, \bar{x}_1) + (\bar{\beta}_2, \bar{\beta}_2)(\bar{x}_2, \bar{x}_2)(\bar{\beta}_3, \bar{\beta}_3)(\bar{x}_3, \bar{x}_3))^{-1}$, then

$$d((\bar{y}, \bar{y}); (\bar{\mu}, \bar{\mu})) = -2 \left\{ (\bar{1}, \bar{1}) + (\bar{y}, \bar{y}) - (\bar{\mu}, \bar{\mu}) - \exp((\bar{y}, \bar{y}) - (\bar{\mu}, \bar{\mu})) \right\}.$$

Suppose the joint probability distribution of $(\bar{x}, \bar{x}) = ((\bar{x}_1, \bar{x}_1), (\bar{x}_2, \bar{x}_2), (\bar{x}_3, \bar{x}_3))^T$ is $p((\bar{x}, \bar{x})|(\bar{\gamma}, \bar{\gamma})) = (\bar{\gamma}_j, \bar{\gamma}_j)^{I_j(\bar{x}, \bar{x})}$, where $I_j(\bar{x}, \bar{x}) (j = 1, 2, \dots, 6)$ denotes the function of the j th result of $x_1, x_1, x_2, x_2, x_3, x_3, \dots, \bar{y}$ ending

Tab.1.1: Blood cancer patient data.

Numbering	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
(\bar{x}_1, \bar{x}_2)	$(\bar{1}, \bar{1})$															
z	0	0	0	0	0	0	0	0	0	0	0	0	1	1	2	2
t	65	156	100	134	16	108	121	4	39	143	56	26	22	1	5	65
Numbering	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
(\bar{x}_1, \bar{x}_2)	$(\bar{0}, \bar{0})$															
z	0	0	0	0	0	0	0	0	1	1	1	1	1	2	2	2
t	56	65	17	7	16	22	3	4	2	3	8	4	3	30	4	43

Tab. 1.2: Partially incomplete data.

Numbering	3	8	19	27	32
$(\tilde{x}_1, \tilde{x}_1)$	$(\tilde{1}, \tilde{1})$	$(\tilde{1}, \tilde{1})$	—	$(\tilde{0}, \tilde{0})$	—
Z	—	—	—	—	—
T	100	4	17	8	43

to the j th result of $((\tilde{x}_1, \tilde{x}_1), (\tilde{x}_2, \tilde{x}_2), (\tilde{x}_3, \tilde{x}_3))$, $\sum_{j=1}^6 (\tilde{\gamma}_j, \tilde{\gamma}_j) = (\tilde{1}, \tilde{1})$. Setting $(\tilde{\beta}, \tilde{\beta}) = ((\tilde{\beta}_0, \tilde{\beta}_0), (\tilde{\beta}_1, \tilde{\beta}_1), (\tilde{\beta}_2, \tilde{\beta}_2), (\tilde{\beta}_3, \tilde{\beta}_3))^T$, the probability density of the complete data based on $n = 32$ observation points of $(\tilde{\theta}, \tilde{\theta}) = ((\tilde{\beta}, \tilde{\beta})^T, (\tilde{\gamma}, \tilde{\gamma})^T)$ can be expressed as

$$p((\tilde{x}, \tilde{x}), (\tilde{y}, \tilde{y}) | (\tilde{\theta}, \tilde{\theta})) = \prod_{i=1}^n \prod_{j=1}^6 -2 \{ (\tilde{1}, \tilde{1}) + (\tilde{y}_i, \tilde{y}_i) - (\tilde{\mu}_i, \tilde{\mu}_i) \\ - \exp((\tilde{y}_i, \tilde{y}_i) - (\tilde{\mu}_i, \tilde{\mu}_i)) \} (\tilde{\gamma}_j, \tilde{\gamma}_j)^{I_j(\tilde{x}, \tilde{x})}.$$

The probability density of the incomplete data is

$$p((\tilde{x}_{obs}, \tilde{x}_{obs}), (\tilde{y}, \tilde{y}) | (\tilde{\theta}, \tilde{\theta})) = \prod_{i=1}^n \prod_{j=1}^6 -2 \{ (\tilde{1}, \tilde{1}) + (\tilde{y}_i, \tilde{y}_i) - (\tilde{\mu}_i, \tilde{\mu}_i) \\ - \exp((\tilde{y}_i, \tilde{y}_i) - (\tilde{\mu}_i, \tilde{\mu}_i)) \} (\tilde{\gamma}_j, \tilde{\gamma}_j).$$

The α – log likelihood ratio of the complete data is

$$L^{(\alpha)}((\tilde{x}, \tilde{x}), (\tilde{y}, \tilde{y}) | (\tilde{\theta}, \tilde{\theta})^{(m)}, (\tilde{\theta}, \tilde{\theta})) \\ = \frac{2}{1+\alpha} \left[\left\{ \frac{p((\tilde{x}, \tilde{x}), (\tilde{y}, \tilde{y}) | (\tilde{\theta}, \tilde{\theta})^{(m)})}{p((\tilde{x}, \tilde{x}), (\tilde{y}, \tilde{y}) | (\tilde{\theta}, \tilde{\theta}))} \right\}^{\frac{1+\alpha}{2}} - 1 \right].$$

Our basic task is to estimate the parameter $(\tilde{\beta}, \tilde{\beta})$. Table 1.3 gives only the maximum likelihood estimate $(\tilde{\beta}, \tilde{\beta})$ of the parameter $(\tilde{\beta}, \tilde{\beta})$ and the estimated standard error, which is the same as the EM algorithm in [32]. However, the convergence speed is different. The influence of different α on the convergence rate indicator can be ascertained from Tab. 1.4. When $\alpha = -1$, we have the EM algorithm. Obviously, obtaining the appropriate α will greatly improve the convergence rate of the algorithm, overcoming the shortcoming of the EM algorithm [29]; of course, which value of α gives the fastest convergence speed requires further exploration.

Note: In Tab. 1.4, the CPU time ratio is the ratio of the CPU consumed by the algorithm to the time taken to execute the algorithm; the CPU time rate is the number of time cycles of the algorithm that the CPU executes per unit of time and the number of time periods of the CPU itself.

Tab. 1.3: Numerical test results.

Complete data		This paper algorithm	
$(\tilde{\beta}, \tilde{\beta})$	Estimation error	$(\tilde{\beta}, \tilde{\beta})$	Estimation error
$(0.303\overrightarrow{4}, 0.303\overrightarrow{4})$	$(0.031\overrightarrow{8}, 0.031\overrightarrow{8})$	$(0.308\overrightarrow{6}, 0.308\overrightarrow{6})$	$(0.033\overrightarrow{6}, 0.033\overrightarrow{6})$
$(-0.079\overrightarrow{1}, -0.079\overrightarrow{1})$	$(0.034\overrightarrow{0}, 0.034\overrightarrow{0})$	$(-0.089\overrightarrow{7}, -0.089\overrightarrow{7})$	$(0.035\overrightarrow{3}, 0.035\overrightarrow{3})$
$(0.180\overrightarrow{3}, 0.180\overrightarrow{3})$	$(0.076\overrightarrow{6}, 0.076\overrightarrow{6})$	$(0.087\overrightarrow{4}, 0.087\overrightarrow{4})$	$(0.040\overrightarrow{5}, 0.040\overrightarrow{5})$
$(0.053\overrightarrow{8}, 0.053\overrightarrow{8})$	$(0.041\overrightarrow{0}, 0.041\overrightarrow{0})$	$(0.067\overrightarrow{2}, 0.067\overrightarrow{2})$	$(0.043\overrightarrow{6}, 0.043\overrightarrow{6})$

Tab. 1.4: Effect of α on the convergence rate index.

α	Convergence speed (steps/second)	CPU time ratio	CPU time rate
-1	15	1.0	1.32
0.6	37	1.4	1.8
0.98	37	1.5	1.68

1.4 Process control model of DFMLS [8]

Studying the visualization and process control of machine learning is a fundamental problem. This section attempts to examine this issue from the perspective of process control to study the problem of learning. The results are summarized below.

1.4.1 Process control model of DFMLS

For a DFMLS composed of m rules $R^l (l = 1, 2, \dots, m)$, the global model of the system is

$$\begin{aligned} (\overrightarrow{X}, \overrightarrow{X})(k+1) &= \sum_{l=1}^m (\overrightarrow{w}_l, \overrightarrow{w}_l)(k) ((\overrightarrow{A}_l, \overrightarrow{A}_l)(\overrightarrow{X}, \overrightarrow{X})(k) + (\overrightarrow{b}_l, \overrightarrow{b}_l)(\overrightarrow{u}, \overrightarrow{u})(k)) \\ &= (\overrightarrow{A}, \overrightarrow{A})(\overrightarrow{X}, \overrightarrow{X})(k) + (\overrightarrow{B}, \overrightarrow{B})(\overrightarrow{u}, \overrightarrow{u})(k), \end{aligned} \quad (1.23)$$

$$\text{where } (\overrightarrow{A}, \overrightarrow{A}) = \sum_{l=1}^m (\overrightarrow{w}_l, \overrightarrow{w}_l)(k) (\overrightarrow{A}_l, \overrightarrow{A}_l), (\overrightarrow{B}, \overrightarrow{B}) = \sum_{l=1}^m (\overrightarrow{w}_l, \overrightarrow{w}_l)(k) (\overrightarrow{b}_l, \overrightarrow{b}_l),$$

$$(\overrightarrow{w}_l, \overrightarrow{w}_l)(k) = \frac{\prod_{i=1}^n (\overrightarrow{\mu}, \overrightarrow{\mu})_{(\overrightarrow{A}_i^l(k), \overrightarrow{A}_i^l(k))}}{\sum_{i=1}^n \prod_{j=1}^n (\overrightarrow{\mu}, \overrightarrow{\mu})_{(\overrightarrow{A}_j^l(k), \overrightarrow{A}_j^l(k))}},$$

$$(\overrightarrow{0}, \overrightarrow{0}) \leq (\overrightarrow{w}_l, \overrightarrow{w}_l)(k) \leq (\overrightarrow{1}, \overrightarrow{1}), \text{ and } \sum_{l=1}^m (\overrightarrow{w}_l, \overrightarrow{w}_l)(k) = (\overrightarrow{1}, \overrightarrow{1}).$$

As each dynamic fuzzy subsystem is linearly descriptive, when we choose the global dynamic fuzzy control law, we first use linear system theory to stabilize the subsystem and obtain the local dynamic fuzzy control law that satisfies the subsystem design requirements. The global dynamic fuzzy control law is a weighted combination of subsystem control laws.

The dynamic fuzzy control rules are as follows:

R_c^l : If $(\vec{x}, \vec{x})(k)$ is $(\vec{A}_1^l, \vec{A}_1^l)$ and $(\vec{x}, \vec{x})(k-1)$ is $(\vec{A}_2^l, \vec{A}_2^l)$ and ... and $(\vec{x}, \vec{x})(k-n+1)$ is $(\vec{A}_n^l, \vec{A}_n^l)$, then

$$(\vec{u}, \vec{u})(k) = (\vec{K}_l, \vec{K}_l)(\vec{X}, \vec{X})(k), (l = 1, 2, \dots, m), \quad (1.24)$$

and the global control is

$$(\vec{u}, \vec{u})(k) = \sum_{l=1}^m (\vec{w}_l, \vec{w}_l)(k)(\vec{K}_l, \vec{K}_l)(\vec{X}, \vec{X})(k). \quad (1.25)$$

Thus, the global model of the DFML process control system (hereinafter referred to as dynamic fuzzy control system) is

$$(\vec{X}, \vec{X})(k+1) = \sum_{i=1}^m \sum_{j=1}^m ((\vec{w}_i, \vec{w}_i)(k)(\vec{w}_j, \vec{w}_j)(k)((\vec{A}_i, \vec{A}_i) + (\vec{b}_i, \vec{b}_i)(\vec{K}_j, \vec{K}_j)))(\vec{X}, \vec{X})(k). \quad (1.26)$$

For ease of analysis, note that

$$(\vec{C}, \vec{C}) = \sum_{i=1}^m \sum_{j=1}^m (\vec{w}_i, \vec{w}_i)(k)(\vec{w}_j, \vec{w}_j)(k)((\vec{A}_i, \vec{A}_i) + (\vec{b}_i, \vec{b}_i)(\vec{K}_j, \vec{K}_j)). \quad (1.27)$$

1.4.2 Stability analysis

The most important characteristic of the control system is its stability, because an unstable system is unable to complete the expected control tasks. This section presents a stability analysis of the dynamic fuzzy control system.

Theorem 1.9 For the dynamic fuzzy control system shown in (1.26), if there exists a common dynamic fuzzy positive definite matrix such that [33]

$$((\vec{A}_i, \vec{A}_i) + (\vec{b}_i, \vec{b}_i)(\vec{K}_j, \vec{K}_j))^T (\vec{P}, \vec{P}) ((\vec{A}_i, \vec{A}_i) + (\vec{b}_i, \vec{b}_i)(\vec{K}_j, \vec{K}_j)) - (\vec{P}, \vec{P}) < (\vec{0}, \vec{0}), (i, j = 1, 2, \dots, m) \quad (1.28)$$

is valid, then the dynamic fuzzy control system (1.26) is globally asymptotically stable.

In general, even if all the subsystems are stable, there is a possibility that the public dynamic fuzzy positive definite matrix $(\overline{P}, \overline{P})$ does not exist, so that (1.28) holds, especially when dynamic fuzzy control rules are more difficult to apply or are invalid [34]. To avoid the difficulty of finding $(\overline{P}, \overline{P})$, we propose a simple and effective method to judge the stability of closed-loop dynamic fuzzy control systems based on the interval matrix and robust control theory.

$(\overline{C}, \overline{C})$ changes with the state value $(\overline{X}, \overline{X})(k)$ at each time, but any element $(\overline{c}_{ij}, \overline{c}_{ij})$ in $(\overline{C}, \overline{C})$ is the weighted sum of the corresponding elements of the subsystem. Thus, we know that the elements of $(\overline{C}, \overline{C})$ are in the determined closed interval given by $(\overline{C}, \overline{C})_{\max} = \max_l (\overline{c}_{ij}^l, \overline{c}_{ij}^l)$, $(\overline{C}, \overline{C})_{\min} = \min_l (\overline{c}_{ij}^l, \overline{c}_{ij}^l)$. Then, $(\overline{C}, \overline{C}) \in [(\overline{C}, \overline{C})_{\min}, (\overline{C}, \overline{C})_{\max}]$. Write $(\overline{C}_0, \overline{C}_0) = ((\overline{C}, \overline{C})_{\max} + (\overline{C}, \overline{C})_{\min})/2$, $(\overline{H}, \overline{H}) = ((\overline{C}, \overline{C})_{\max} - (\overline{C}, \overline{C})_{\min})/2$, $(\overline{h}_{ij}, \overline{h}_{ij}) = ((\overline{h}_{ij}, \overline{h}_{ij}))$.

Using the properties of interval matrices, $(\overline{C}, \overline{C})$ can be equivalently expressed as follows:

$$\begin{aligned} (\overline{C}, \overline{C}) &= (\overline{C}_0, \overline{C}_0) + (\overline{M}, \overline{M})(\overline{\Sigma}, \overline{\Sigma})(k)(\overline{N}, \overline{N}) \\ (\overline{\Sigma}, \overline{\Sigma}) &= \text{diag}[(\overline{\varepsilon}_{11}, \overline{\varepsilon}_{11})(k), \dots, (\overline{\varepsilon}_{1n}, \overline{\varepsilon}_{1n})(k), \dots, (\overline{\varepsilon}_{n1}, \overline{\varepsilon}_{n1})(k), \dots, (\overline{\varepsilon}_{nn}, \overline{\varepsilon}_{nn})(k)], \end{aligned} \quad (1.29)$$

where

$$|(\overline{\varepsilon}_{ij}, \overline{\varepsilon}_{ij})(k)| \leq (\overline{1}, \overline{1}), (i, j = 1, 2, \dots, n);$$

$$\begin{aligned} (\overline{M}, \overline{M}) &= \left[\sqrt{(\overline{h}_{11}, \overline{h}_{11})}(\overline{e}_1, \overline{e}_1) \dots \sqrt{(\overline{h}_{1n}, \overline{h}_{1n})}(\overline{e}_1, \overline{e}_1) \dots \sqrt{(\overline{h}_{n1}, \overline{h}_{n1})}(\overline{e}_n, \overline{e}_n) \dots \right. \\ &\quad \left. \sqrt{(\overline{h}_{nn}, \overline{h}_{nn})}(\overline{e}_n, \overline{e}_n) \right] \end{aligned}$$

$$\begin{aligned} (\overline{N}, \overline{N}) &= \left[\sqrt{(\overline{h}_{11}, \overline{h}_{11})}(\overline{e}_1, \overline{e}_1) \dots \sqrt{(\overline{h}_{1n}, \overline{h}_{1n})}(\overline{e}_n, \overline{e}_n) \dots \sqrt{(\overline{h}_{n1}, \overline{h}_{n1})}(\overline{e}_1, \overline{e}_1) \dots \right. \\ &\quad \left. \sqrt{(\overline{h}_{nn}, \overline{h}_{nn})}(\overline{e}_n, \overline{e}_n) \right]^T. \end{aligned}$$

Here, $(\overline{e}_i, \overline{e}_i)$ is the unit dynamic fuzzy column vector whose i th element is $(\overline{1}, \overline{1})$, and the remaining elements are $(\overline{0}, \overline{0})$; $(\overline{M}, \overline{M}) \in (\overline{R}, \overline{R})^{n \times n^2}$, $(\overline{N}, \overline{N}) \in (\overline{R}, \overline{R})^{n^2 \times n}$, $(\overline{\Sigma}, \overline{\Sigma}) \in (\overline{R}, \overline{R})^{n^2 \times n^2}$ is the dynamic fuzzy diagonal matrix, and the value of $(\overline{\Sigma}, \overline{\Sigma})$ changes with the rule weight value at each time, but $(\overline{\Sigma}, \overline{\Sigma})^T(k)(\overline{\Sigma}, \overline{\Sigma})(k) \leq (\overline{I}, \overline{I})$. Then, the global model of the closed-loop dynamic fuzzy control system can be

expressed as

$$(\overleftarrow{X}, \overrightarrow{X})(k+1) = ((\overleftarrow{C}_0, \overrightarrow{C}_0) + (\overleftarrow{M}, \overrightarrow{M})(\overleftarrow{\Sigma}, \overrightarrow{\Sigma})(k)(\overleftarrow{N}, \overrightarrow{N}))(\overleftarrow{X}, \overrightarrow{X})(k). \quad (1.30)$$

Definition 1.10 For an uncertain dynamic fuzzy autonomous system [35],

$$\dot{(\overleftarrow{X}, \overrightarrow{X})}(t) = ((\overleftarrow{A}_0, \overrightarrow{A}_0) + \Delta(\overleftarrow{A}, \overrightarrow{A}))(\overleftarrow{X}, \overrightarrow{X})(t),$$

if there exists an n -order dynamic fuzzy positive definite matrix $(\overleftarrow{P}, \overrightarrow{P})$ and a dynamic fuzzy constant $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) > (\overleftarrow{0}, \overrightarrow{0})$ such that, for any permissible uncertainty $\Delta(\overleftarrow{A}, \overrightarrow{A})$,

$$\begin{aligned} & (\overleftarrow{X}, \overrightarrow{X})^T(t)((\overleftarrow{A}_0, \overrightarrow{A}_0)^T(\overleftarrow{P}, \overrightarrow{P}) + (\overleftarrow{P}, \overrightarrow{P})(\overleftarrow{A}_0, \overrightarrow{A}_0))(\overleftarrow{X}, \overrightarrow{X})(t) \\ & + 2(\overleftarrow{X}, \overrightarrow{X})^T(t)\Delta(\overleftarrow{A}, \overrightarrow{A})^T(t)(\overleftarrow{P}, \overrightarrow{P})(\overleftarrow{X}, \overrightarrow{X})(t) \leq -(\overleftarrow{\alpha}, \overrightarrow{\alpha})\|(\overleftarrow{X}, \overrightarrow{X})(t)\|^2 \end{aligned}$$

holds for any solution $(\overleftarrow{X}, \overrightarrow{X})(t)$ and t , then the system is said to be quadratic stable. In the above expression, $(\overleftarrow{A}_0, \overrightarrow{A}_0) = ((\overleftarrow{A}, \overrightarrow{A})_{\max} + (\overleftarrow{A}, \overrightarrow{A})_{\min})/2$, $(\overleftarrow{A}, \overrightarrow{A})_{\max} = \max_l((\overleftarrow{\alpha}_{ij}^l, \overrightarrow{\alpha}_{ij}^l))$, $(\overleftarrow{A}, \overrightarrow{A})_{\min} = \min_l((\overleftarrow{\alpha}_{ij}^l, \overrightarrow{\alpha}_{ij}^l))$.

Theorem 1.10 If the disturbance system (1.30) is quadratic stable, the closed-loop dynamic fuzzy control system (1.26) is asymptotically stable.

Proof: The perturbed system (1.30) is quadratic stable according to definition (1.10):

$$\begin{aligned} & (\overleftarrow{X}, \overrightarrow{X})^T(k)\{((\overleftarrow{C}_0, \overrightarrow{C}_0) + (\overleftarrow{M}, \overrightarrow{M})(\overleftarrow{\Sigma}, \overrightarrow{\Sigma})(k)(\overleftarrow{N}, \overrightarrow{N}))^T(\overleftarrow{P}, \overrightarrow{P})(\overleftarrow{C}_0, \overrightarrow{C}_0) \\ & + (\overleftarrow{M}, \overrightarrow{M})(\overleftarrow{\Sigma}, \overrightarrow{\Sigma})(k)(\overleftarrow{N}, \overrightarrow{N}) - (\overleftarrow{P}, \overrightarrow{P})\}(\overleftarrow{X}, \overrightarrow{X})(k) \leq -(\overleftarrow{\alpha}, \overrightarrow{\alpha})\|(\overleftarrow{X}, \overrightarrow{X})(t)\|^2. \end{aligned}$$

For the closed-loop dynamic fuzzy control system (1.26), we have the Lyapunov function

$$V[(\overleftarrow{X}, \overrightarrow{X})(k)] = (\overleftarrow{X}, \overrightarrow{X})^T(k)(\overleftarrow{P}, \overrightarrow{P})(\overleftarrow{X}, \overrightarrow{X})(k),$$

where $(\overleftarrow{P}, \overrightarrow{P})$ is a dynamic fuzzy symmetric positive definite matrix. As the system global representation matrix $(\overleftarrow{C}, \overrightarrow{C})$ is an interval matrix, and $(\overleftarrow{C}, \overrightarrow{C}) = ((\overleftarrow{C}_0, \overrightarrow{C}_0) + (\overleftarrow{M}, \overrightarrow{M})(\overleftarrow{\Sigma}, \overrightarrow{\Sigma})(k)(\overleftarrow{N}, \overrightarrow{N}), (\overleftarrow{\Sigma}, \overrightarrow{\Sigma})^T(k)(\overleftarrow{N}, \overrightarrow{N}) \leq (\overleftarrow{I}, \overrightarrow{I})_{n^2}$, we have

$$\begin{aligned} \Delta V[(\overleftarrow{X}, \overrightarrow{X})(k)] &= V[(\overleftarrow{X}, \overrightarrow{X})(k+1)] - V[(\overleftarrow{X}, \overrightarrow{X})(k)] = (\overleftarrow{X}, \overrightarrow{X})^T(k)\{((\overleftarrow{C}_0, \overrightarrow{C}_0) \\ & + (\overleftarrow{M}, \overrightarrow{M})(\overleftarrow{\Sigma}, \overrightarrow{\Sigma})(k)(\overleftarrow{N}, \overrightarrow{N}))^T \end{aligned}$$

$$\begin{aligned}
& (\overleftarrow{P}, \overrightarrow{P})((\overleftarrow{C}_0, \overrightarrow{C}_0) + (\overleftarrow{M}, \overrightarrow{M})(\overleftarrow{\Sigma}, \overrightarrow{\Sigma})(k)(\overleftarrow{N}, \overrightarrow{N}) - (\overleftarrow{P}, \overrightarrow{P}))(\overleftarrow{X}, \overrightarrow{X})(k) \\
& \leq -(\overrightarrow{\alpha}, \overrightarrow{\alpha})\|(\overleftarrow{X}, \overrightarrow{X})(t)\|^2 < (\overrightarrow{0}, \overrightarrow{0})
\end{aligned}$$

and the closed loop dynamic fuzzy control system (1.26) is asymptotically stable.

Lemma 1.11 $(\overleftarrow{A}, \overrightarrow{A})$, $(\overleftarrow{D}, \overrightarrow{D})$, $(\overleftarrow{E}, \overrightarrow{E})$ and $(\overleftarrow{F}, \overrightarrow{F})$ are dynamic fuzzy positive definite matrices and, if $(\overleftarrow{P}, \overrightarrow{P}) - (\overrightarrow{\varepsilon}, \overrightarrow{\varepsilon})(\overleftarrow{D}, \overrightarrow{D})(\overleftarrow{D}, \overrightarrow{D})^T > (\overrightarrow{0}, \overrightarrow{0})$ holds for any dynamic fuzzy symmetric matrix $(\overleftarrow{P}, \overrightarrow{P})$ and scalar $(\overrightarrow{\varepsilon}, \overrightarrow{\varepsilon}) > (\overrightarrow{0}, \overrightarrow{0})$, then [35]

$$\begin{aligned}
& ((\overleftarrow{A}, \overrightarrow{A}) + (\overleftarrow{D}, \overrightarrow{D})(\overleftarrow{F}, \overrightarrow{F})(\overleftarrow{E}, \overrightarrow{E}))^T(P, P)^{-1}((\overleftarrow{A}, \overrightarrow{A}) + (\overleftarrow{D}, \overrightarrow{D})(\overleftarrow{F}, \overrightarrow{F})(\overleftarrow{E}, \overrightarrow{E})) \\
& \leq (\overleftarrow{A}, \overrightarrow{A})^T((\overleftarrow{P}, \overrightarrow{P}) - (\overrightarrow{\varepsilon}, \overrightarrow{\varepsilon})(\overleftarrow{D}, \overrightarrow{D})(\overleftarrow{D}, \overrightarrow{D})^T)^{-1}(\overleftarrow{A}, \overrightarrow{A}) + (\overrightarrow{\varepsilon}, \overrightarrow{\varepsilon})^{-1}(\overleftarrow{E}, \overrightarrow{E})^T(\overleftarrow{E}, \overrightarrow{E}).
\end{aligned}$$

Theorem 1.11 If there exist a scalar $(\overrightarrow{\alpha}, \overrightarrow{\alpha}) > (\overrightarrow{0}, \overrightarrow{0})$ and dynamic fuzzy symmetric positive definite matrix $(\overleftarrow{P}, \overrightarrow{P})$, the following holds:

$$\begin{aligned}
& (\overleftarrow{C}_0, \overrightarrow{C}_0)^T((\overleftarrow{P}, \overrightarrow{P})^{-1} - (\overrightarrow{\alpha}, \overrightarrow{\alpha})(\overleftarrow{M}, \overrightarrow{M})(\overrightarrow{M}, \overrightarrow{M})^T)^{-1}(\overleftarrow{C}_0, \overrightarrow{C}_0) + (\overrightarrow{\alpha}, \overrightarrow{\alpha})^{-1}(\overleftarrow{N}, \overrightarrow{N})^T(\overleftarrow{N}, \overrightarrow{N}) \\
& - (\overleftarrow{P}, \overrightarrow{P}) < (\overrightarrow{0}, \overrightarrow{0}) \\
& (\overleftarrow{P}, \overrightarrow{P})^{-1} - (\overrightarrow{\alpha}, \overrightarrow{\alpha})(\overleftarrow{M}, \overrightarrow{M})(\overrightarrow{M}, \overrightarrow{M})^T > (\overrightarrow{0}, \overrightarrow{0})
\end{aligned}$$

and the closed-loop dynamic fuzzy control system (1.26) is asymptotically stable.

Proof: Let the Lyapunov function of the closed-loop dynamic fuzzy control system (1.26) be

$$V[(\overleftarrow{x}, \overrightarrow{x})(k)] = (\overleftarrow{x}, \overrightarrow{x})^T(k)(\overleftarrow{P}, \overrightarrow{P})(\overleftarrow{x}, \overrightarrow{x})(k),$$

where $(\overleftarrow{P}, \overrightarrow{P})$ is a dynamic fuzzy symmetric positive definite matrix. Then,

$$\begin{aligned}
& \Delta V[(\overleftarrow{x}, \overrightarrow{x})(k)] \\
& = V[(\overleftarrow{x}, \overrightarrow{x})(k+1)] - V[(\overleftarrow{x}, \overrightarrow{x})(k)] \\
& = (\overleftarrow{x}, \overrightarrow{x})^T(k) \left\{ \left[\sum_{i=1}^m \sum_{j=1}^m (\overleftarrow{w}_i, \overrightarrow{w}_i)(k)(\overleftarrow{w}_j, \overrightarrow{w}_j)(k)((\overleftarrow{A}_i, \overrightarrow{A}_i) + (\overleftarrow{B}_i, \overrightarrow{B}_i)(\overleftarrow{K}_j, \overrightarrow{K}_j)) \right]^T \right. \\
& \quad \times (\overleftarrow{P}, \overrightarrow{P}) \left[\sum_{i=1}^m \sum_{j=1}^m (\overleftarrow{w}_i, \overrightarrow{w}_i)(k)(\overleftarrow{w}_j, \overrightarrow{w}_j)(k)((\overleftarrow{A}_i, \overrightarrow{A}_i) + (\overleftarrow{B}_i, \overrightarrow{B}_i)(\overleftarrow{K}_j, \overrightarrow{K}_j)) \right] \\
& \quad \left. - (\overleftarrow{P}, \overrightarrow{P}) \right\} \times (\overleftarrow{x}, \overrightarrow{x})(k)
\end{aligned}$$

according to (1.30), and

$$\begin{aligned}\Delta V[(\overleftarrow{x}, \overrightarrow{x})(k)] &= (\overleftarrow{x}, \overrightarrow{x})^T(k) \left\{ ((\overleftarrow{C}_0, \overrightarrow{C}_0) + (\overleftarrow{M}, \overrightarrow{M})(\overleftarrow{\Sigma}, \overrightarrow{\Sigma})(k)(\overleftarrow{N}, \overrightarrow{N}))^T(\overleftarrow{P}, \overrightarrow{P}) \right. \\ &\quad \left. - ((\overleftarrow{C}_0, \overrightarrow{C}_0) + (\overleftarrow{M}, \overrightarrow{M})(\overleftarrow{\Sigma}, \overrightarrow{\Sigma})(k)(\overleftarrow{N}, \overrightarrow{N})) - (\overleftarrow{P}, \overrightarrow{P}) \right\} (\overleftarrow{x}, \overrightarrow{x})(k).\end{aligned}$$

Using Lemma 1.1, if there exists a scalar $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) > (\overleftarrow{0}, \overrightarrow{0})$ and a dynamic fuzzy symmetric positive definite matrix $(\overleftarrow{P}, \overrightarrow{P})$ satisfying $(\overleftarrow{P}, \overrightarrow{P})^{-1} - (\overleftarrow{\alpha}, \overrightarrow{\alpha})(\overleftarrow{M}, \overrightarrow{M})$ $(\overleftarrow{M}, \overrightarrow{M})^T > (\overleftarrow{0}, \overrightarrow{0})$, then

$$\begin{aligned}\Delta V[(\overleftarrow{x}, \overrightarrow{x})(k)] &\leq (\overleftarrow{x}, \overrightarrow{x})^T(k) \left\{ (\overleftarrow{C}_0, \overrightarrow{C}_0)^T((\overleftarrow{P}, \overrightarrow{P})^{-1} - (\overleftarrow{\alpha}, \overrightarrow{\alpha})(\overleftarrow{M}, \overrightarrow{M})(\overleftarrow{M}, \overrightarrow{M})^T)^{-1} \right. \\ &\quad \left. - (\overleftarrow{C}_0, \overrightarrow{C}_0) + (\overleftarrow{\alpha}, \overrightarrow{\alpha})^{-1}(\overleftarrow{N}, \overrightarrow{N})^T(\overleftarrow{N}, \overrightarrow{N}) - (\overleftarrow{P}, \overrightarrow{P}) \right\} (\overleftarrow{x}, \overrightarrow{x})(k) < (\overleftarrow{0}, \overrightarrow{0}).\end{aligned}$$

According to the Lyapunov stability theorem, the closed-loop dynamic fuzzy system (1.26) is asymptotically stable.

1.4.3 Design of dynamic fuzzy learning controller

Designing the dynamic fuzzy learning controller has the following considerations:

- (1) Determine the input variables and output variables (i.e. the control variables) of the dynamic fuzzy learning controller;
- (2) Design the control rules of the dynamic fuzzy learning controller;
- (3) Choose the domain of the input variables and output variables of the dynamic fuzzy learning controller;
- (4) Prepare the application of the dynamic fuzzy learning controller design algorithm;
- (5) Choose a reasonable dynamic fuzzy learning controller design algorithm sampling time.

This section discusses the following.

1.4.3.1 Input variables and output variables of dynamic fuzzy learning controller

Choosing which variables constitute the information for the dynamic fuzzy learning controller is a problem worthy of further study. Because this control system services the DFMLS, its input data come from the DFMLS itself.

In a manual control process, the amount of information that people can obtain is based on three factors:

- (1) Error;
- (2) The variation of error; and
- (3) The rate of change in error.

Tab. 1.5: Dynamic fuzzy control rules table.

E	EC						
	NB	NM	NS	ZO	PS	PM	PB
NB	PB	PB	PB	PB	PM	0	0
NM	PB	PB	PB	PB	PM	0	0
NS	PM	PM	PM	PM	0	NS	NS
ZO	PM	PM	PS	O	NS	NM	NM
PS	PS	PS	O	NM	NM	NM	NM
PM	0	O	NM	NB	NB	NB	NB
PB	O	O	NM	NB	NB	NB	NB

Therefore, there are three input variables to the dynamic fuzzy learning controller, namely, the error of the output data from the DFMLS, the variation of the error, and the change in the error variation. The output variables of the dynamic fuzzy learning controller generally determine the variation of the control variable.

1.4.3.2 Design of control rules for dynamic fuzzy learning controller

The control rules of the dynamic fuzzy learning controller can be described in the following linguistic form:

- (1) If $(\overleftarrow{A}, \overrightarrow{A})$ then $(\overleftarrow{B}, \overrightarrow{B})$;
- (2) If $(\overleftarrow{A}, \overrightarrow{A})$ then $(\overleftarrow{B}, \overrightarrow{B})$; else $(\overleftarrow{C}, \overrightarrow{C})$;
- (3) If $(\overleftarrow{A}, \overrightarrow{A})$ and $(\overleftarrow{B}, \overrightarrow{B})$, then $(\overleftarrow{C}, \overrightarrow{C})$.

These rules can be expressed as: if $(\overleftarrow{A}, \overrightarrow{A})$ then if $(\overleftarrow{B}, \overrightarrow{B})$ then $(\overleftarrow{C}, \overrightarrow{C})$.

The corresponding control strategies of the operator that may be encountered in the operation process are summarized in Tab. 1.5.

The basic idea of establishing a dynamic fuzzy learning control rule table is as follows:

- (1) When the error is negative (large), then the error has an increasing trend. To eliminate the existing negative large errors and prevent the error from becoming larger, the control volume change is positive (large);
- (2) When the error is negative and the change in error is positive, the system itself has a tendency to reduce the error. Thus, to eliminate the error as soon as possible, a smaller control amount should be used. As can be seen from Tab. 1.5, when the error is negative (large) and the change in error is positive (small), the control variable is taken as positive (middle). When the error is positive (large) or positive (middle), the control volume should not be increased, otherwise an overshoot may produce a positive error. Hence, the change in the control quantity is set to the 0 level;
- (3) When the error is negative (middle), the change in the control quantity should eliminate the error as soon as possible. Based on this principle, the variation of the control variable is the same as when the error is negative (large).

- (4) When the error is negative (small), the system is close to the steady state. If the change in error is negative, select a positive (middle) control variable to suppress the change in error in the negative direction. If the error variation is positive, the system itself has a tendency to eliminate negative (small) errors; thus, select the control variable to be positive (small);
- (5) The situation when the error is positive is similar to that when the error is negative with the corresponding change of symbol.

Therefore, the principle of selecting the control variable can be summarized as follows: when the error is large, select the control to eliminate the error as soon as possible; when the error is small, the choice of control should prevent an overshoot to ensure the stability of the system.

1.4.3.3 Design algorithm of dynamic fuzzy learning controller

To design a stable dynamic fuzzy learning controller according to Theorem (1.28) [36], the design algorithm is as follows:

- (1) For the closed-loop dynamic fuzzy control system (1.26), the object parameters $(\vec{A}_l, \vec{A}_l), (\vec{B}_l, \vec{B}_l)$ and their corresponding membership function are known, whereas the dynamic fuzzy learning controller parameter (\vec{K}_l, \vec{K}_l) and its membership function $(\vec{\mu}, \vec{\mu})_{(\vec{K}_l, \vec{K}_l)}$ are to be designed. Usually, the membership function $(\vec{\mu}, \vec{\mu})_{(\vec{K}_l, \vec{K}_l)}$ is selected and the parameter (\vec{K}_l, \vec{K}_l) is designed according to Theorem (1.28).
- (2) The parameter (\vec{K}_l, \vec{K}_l) is selected so that the approximate linear subsystem is stable. According to (1.4), the approximate linear subsystem is

$$(\vec{x}, \vec{x})(k+1) = (\vec{w}_i, \vec{w}_i)(\vec{w}_j, \vec{w}_j)(\vec{Q}_{ij}, \vec{Q}_{ij})(\vec{x}, \vec{x})(k) \\ (\vec{Q}_{ij}, \vec{Q}_{ij}) = (\vec{A}_i, \vec{A}_i) + (\vec{B}_i, \vec{B}_i)(\vec{K}_j, \vec{K}_j).$$

- (3) Find the dynamic fuzzy positive definite matrix (\vec{P}_i, \vec{P}_i) such that $(\vec{Q}_{ij}, \vec{Q}_{ij})^T(\vec{P}_i, \vec{P}_i)(\vec{Q}_{ij}, \vec{Q}_{ij}) - (\vec{P}_i, \vec{P}_i) < (\vec{0}, \vec{0})$. For a given $i^* \in \{1, 2, \dots, m\}$, if there exists some $(\vec{P}_{i^*}, \vec{P}_{i^*})$ such that $(\vec{Q}_{ij}, \vec{Q}_{ij})^T(\vec{P}_{i^*}, \vec{P}_{i^*})(\vec{Q}_{ij}, \vec{Q}_{ij}) - (\vec{P}_{i^*}, \vec{P}_{i^*}) < (\vec{0}, \vec{0})$, then select $(\vec{P}, \vec{P}) = (\vec{P}_i, \vec{P}_i)$; otherwise, return to (2), and redesign the parameter (\vec{K}_l, \vec{K}_l) until $(\vec{P}, \vec{P}) = (\vec{P}_i, \vec{P}_i)$.

1.4.4 Simulation examples

Example 1.3 Consider an inverted pendulum system:

$$(\vec{x}_1, \dot{\vec{x}}_1) = (\vec{x}_2, \vec{x}_2) \\ (\vec{x}_2, \dot{\vec{x}}_2) = \frac{g \sin(\vec{x}_1, \vec{x}_1) - (aml(\vec{x}_2, \vec{x}_2)^2 \sin(2\vec{x}_1, 2\vec{x}_1))/2 - a \cos(\vec{x}_1, \vec{x}_1)u}{4l^3 aml \cos^2 \vec{x}_1, \vec{x}_1},$$

where (\vec{x}_1, \vec{x}_1) is the angle (in radians) between the pendulum and the horizontal direction, (\vec{x}_2, \vec{x}_2) is the angular velocity (rad/s) of the pendulum, $g = 9.8m/s^2$ is the acceleration due to gravity, m is the mass of the pendulum, M is the mass of the trolley, $2l$ is the pendulum length, and u is the horizontal force exerted on the trolley. The parameters are as follows: $m = 2.0kg$, $M = 8.0kg$, $2l = 1.0m$, $a = 1/(m + M)$.

The following dynamic fuzzy model is adopted [37]:

R^1 : if $(\vec{x}_1, \vec{x}_1)(t)$ is $(\vec{0}, \vec{0})$, then $(\dot{\vec{x}}, \vec{x})(t) = (\vec{A}_1, \vec{A}_1)(\vec{x}, \vec{x})(t) + (\vec{B}_1, \vec{B}_1)u(t)$;

R^1 : if $(\vec{x}_1, \vec{x}_1)(t)$ is $(\frac{\pi}{2}, \frac{\pi}{2})$, then $(\dot{\vec{x}}, \vec{x})(t) = (\vec{A}_2, \vec{A}_2)(\vec{x}, \vec{x})(t) + (\vec{B}_2, \vec{B}_2)u(t)$;

where

$$(\vec{A}_1, \vec{A}_1) = \left[\begin{array}{cc} (\vec{0}, \vec{0}) & (\vec{1}, \vec{1}) \\ \left(\frac{\vec{g}}{4l/3 - aml}, \frac{\vec{g}}{4l/3 - aml} \right) & (\vec{0}, \vec{0}) \end{array} \right],$$

$$(\vec{B}_1, \vec{B}_1) = \left[\begin{array}{c} (\vec{0}, \vec{0}) \\ \left(\frac{\vec{a}}{4l/3 - aml}, \frac{\vec{a}}{4l/3 - aml} \right) \end{array} \right],$$

$$(\vec{A}_2, \vec{A}_2) = \left[\begin{array}{cc} (\vec{0}, \vec{0}) & (\vec{1}, \vec{1}) \\ \left(\frac{2\vec{g}}{\pi(4l/3 - aml\beta^2)}, \frac{2\vec{g}}{\pi(4l/3 - aml\beta^2)} \right) & (\vec{0}, \vec{0}) \end{array} \right],$$

$$(\vec{B}_2, \vec{B}_2) = \left[\begin{array}{c} (\vec{0}, \vec{0}) \\ \left(\frac{\vec{a}\beta}{(4l/3 - aml\beta^2)}, \frac{\vec{a}\beta}{(4l/3 - aml\beta^2)} \right) \end{array} \right],$$

$$\beta = \cos(88^\circ),$$

and the membership functions of the DFSs (\vec{G}_1, \vec{G}_1) and (\vec{G}_2, \vec{G}_2) are

$$[(\vec{\mu}_1, \vec{\mu}_1)(x) = ((1 - \frac{1}{1 + \exp(-7(x - \frac{\pi}{4}))}) \frac{1}{1 + \exp(-7(x + \frac{\pi}{4}))}, \leftarrow, \rightarrow),$$

$$(\vec{\mu}_2, \vec{\mu}_2)(x) = (\vec{1}, \vec{1}) - (\vec{\mu}_1, \vec{\mu}_1)(x).$$

We obtain the following interval representation of the coefficient matrix:

$$(\vec{A}, \vec{A})_{\max} = \left[\begin{array}{cc} (\vec{0}, \vec{0}) & (\vec{1}, \vec{1}) \\ (17.29, 17.29) & (\vec{0}, \vec{0}) \end{array} \right], (\vec{A}, \vec{A})_{\min} = \left[\begin{array}{cc} (\vec{0}, \vec{0}) & (\vec{1}, \vec{1}) \\ (9.36, 9.36) & (\vec{0}, \vec{0}) \end{array} \right],$$

$$(\overleftarrow{B}, \overrightarrow{B})_{\max} = \begin{bmatrix} (\overleftarrow{0}, \overrightarrow{0}) \\ (0.18, 0.18) \end{bmatrix}, (\overleftarrow{B}, \overrightarrow{B})_{\min} = \begin{bmatrix} (\overleftarrow{0}, \overrightarrow{0}) \\ (0.005, 0.005) \end{bmatrix}, \text{ and}$$

$$(\overleftarrow{C}, \overrightarrow{C})_{\max} = \begin{bmatrix} (\overleftarrow{0}, \overrightarrow{0}) \\ (-22.46, -22.46) \end{bmatrix}, (\overrightarrow{1.000\overleftarrow{2}}, \overrightarrow{1.000\overrightarrow{2}}) \\ (-8.68, -8.68)$$

$$(\overleftarrow{C}, \overrightarrow{C})_{\min} = \begin{bmatrix} (\overleftarrow{0}, \overrightarrow{0}) \\ (-30.08, -30.08) \end{bmatrix}, (\overrightarrow{1}, \overrightarrow{1}) \\ (-10.72, -10.72)$$

The state feedback gain of subsystems 1 and 2 is then:

$$(\overleftarrow{K}_1, \overrightarrow{K}_1) = [(-211.3, -211.3) \quad (-45.3, -45.3)]$$

$$(\overleftarrow{K}_2, \overrightarrow{K}_2) = [(-5607.5, -5607.5) \quad (-1527.9, -1527.9)]$$

and so:

$$(\overleftarrow{A}_0, \overrightarrow{A}_0) = \begin{bmatrix} (\overleftarrow{0}, \overrightarrow{0}) \\ (13.325, 13.325) \end{bmatrix}, (\overrightarrow{1}, \overrightarrow{1}) \\ (\overrightarrow{0}, \overrightarrow{0})$$

$$(\overleftarrow{C}_0, \overrightarrow{C}_0) = \begin{bmatrix} (\overleftarrow{0}, \overrightarrow{0}) \\ (-26.27, -26.27) \end{bmatrix}, (\overrightarrow{1}, \overrightarrow{1}) \\ (-9.7, -9.7)$$

$$(\overleftarrow{M}, \overrightarrow{M})(\overleftarrow{M}, \overrightarrow{M})^T = \begin{bmatrix} (0.000\overleftarrow{1}, 0.000\overrightarrow{1}) \\ (\overrightarrow{0}, \overrightarrow{0}) \end{bmatrix}, (\overrightarrow{0}, \overrightarrow{0}) \\ (4.83, 4.83)$$

$$(N, N) = \begin{bmatrix} (\overleftarrow{0}, \overrightarrow{0}) & (\overleftarrow{0}, \overrightarrow{0}) & (1.95\overleftarrow{2}, 1.95\overrightarrow{2}) & (\overrightarrow{0}, \overrightarrow{0}) \\ (\overrightarrow{0}, \overrightarrow{0}) & (0.01, 0.01) & (\overrightarrow{0}, \overrightarrow{0}) & (1.01, 1.01) \end{bmatrix}^T.$$

Finally, a dynamic fuzzy positive definite matrix is obtained:

$$(\overleftarrow{P}, \overrightarrow{P}) = \begin{bmatrix} (0.01\overleftarrow{3}, 0.01\overrightarrow{3}) & (1.01\overleftarrow{2}, 1.01\overrightarrow{2}) \\ (-27.46\overleftarrow{2}, -27.46\overrightarrow{2}) & (-10.05, -10.05) \end{bmatrix}.$$

Therefore, from Theorem (1.28), the inverted pendulum system is asymptotically stable under the control of the dynamic fuzzy learning controller $(\overleftarrow{u}, \overrightarrow{u})(k) = ((\overleftarrow{u}_1, \overrightarrow{u}_1)(k)(\overleftarrow{K}_1, \overrightarrow{K}_1) + (\overleftarrow{u}_2, \overrightarrow{u}_2)(k)(\overleftarrow{K}_2, \overrightarrow{K}_2))(\overleftarrow{x}, \overrightarrow{x})(k)$.

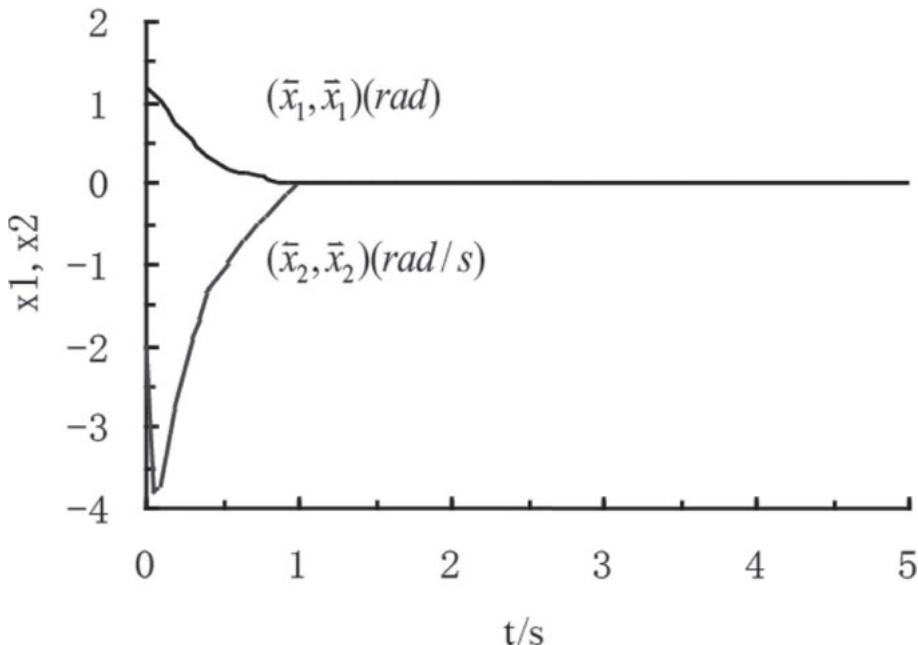


Fig. 1.11: $(\bar{x}_1, \bar{x}_1)(0) = (60^\circ, 60^\circ)$, $(\bar{x}_2, \bar{x}_2)(0) = (0, 0)$ simulation curve.

The initial points

$$(\bar{x}_1, \bar{x}_1)(0) = (60^\circ, 60^\circ), (\bar{x}_2, \bar{x}_2)(0) = (0, 0)$$

$$(\bar{x}_1, \bar{x}_1)(0) = (89^\circ, 89^\circ), (\bar{x}_2, \bar{x}_2)(0) = (0, 0)$$

give the simulation curves shown in Figs. 1.11 and 1.12, respectively. It can be seen from these figures that the system stabilizes at equilibrium point $(0, 0)$ after 1.01s; i.e. point $(0, 0)$ is the stable equilibrium of the closed-loop system. The inverted pendulum system in [37] becomes stable at $(0, 0)$ after 1.05s, so the performance of the proposed system is superior.

1.5 Dynamic fuzzy relational learning algorithm

In machine learning systems, data with dynamic fuzzy relations are universal. Dynamic fuzzy relations provide an effective method for describing and dealing with dynamic fuzzy phenomena. For further information on dynamic fuzzy relations, see Appendix 8.2 and References [1, 2, 4]. In this section, a learning algorithm for dynamic fuzzy relation (\bar{R}, \bar{R}) is described based on the theory of dynamic fuzzy relations.

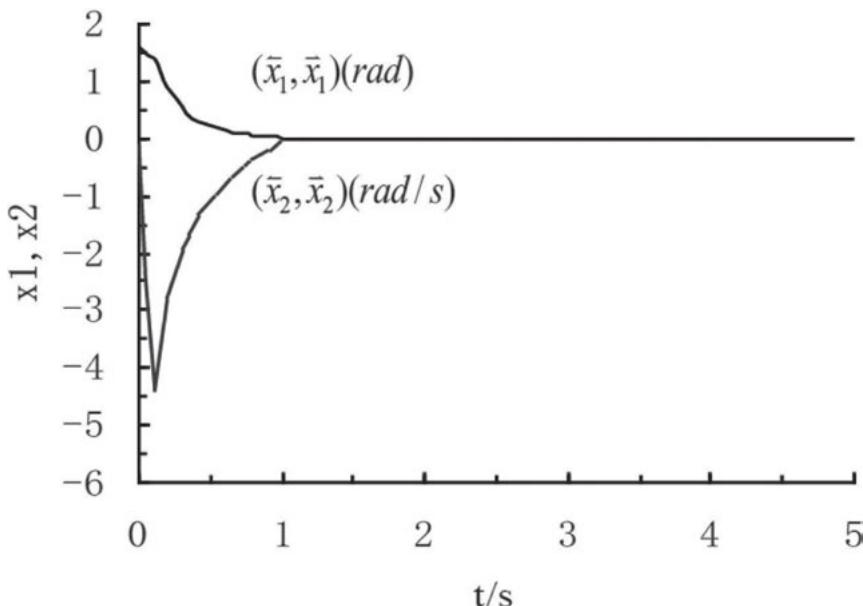


Fig. 1.12: $(\bar{x}_1, \bar{x}_1)(0) = (89^\circ, 89^\circ)$, $(\bar{x}_2, \bar{x}_2)(0) = (0^\circ, 0^\circ)$ simulation curve.

1.5.1 An outline of relational learning

Relational learning is an important paradigm in the field of machine learning. Its early work can be traced back to the famous Winston program of 1970, which could learn concepts such as arches. This program uses the semantic web as a descriptive language. It takes the arches with their components and properties and considers the description of the nature of properties as a node of the semantic network. In the same year, Codd published a landmark paper on “A Relational Model of Data for Large Shared Data Banks”, which was a prelude to the idea of relational databases. This paper is cited in many applications of relational learning, as it enables data with complex internal relations to be concisely expressed.

As we know, early studies on relational learning focus on inductive logic programming (ILP) [38, 39]. Although the logical representations that are presented in Baner’s “A Language for the Description of Concepts” and in Michalski’s AQ algorithm are first applied to learning problems, the foundation of this work is the first-order predicate formula presented by Plotkin in the 1970s, which laid the theoretical foundation for the development of ILP. The thrust of Plotkin’s work is to use the “special to general” approach to obtain a conclusion; this is strictly limited in practice, as a generalization of the smallest general conclusion is not finite. Thus, Shapiro used the general to special method to research the induction problem on the Horn clause set and proposed the MIS learning system in 1983. His work clarified the relationship between the logical program and inductive learning.

Early research on ILP focuses mainly on relationship rule induction, i.e. first-order logic concept learning and logic program composition. Representative systems include CIGOL (Muggleton, 1988) [54], FOIL (Quinlan, 1990) [55], GOLEM (Muggleton, 1990) [56], and LINUS (Lavarac, Dzeroski, and Grobelnik, 1991) [57]. In recent years, the research scope of ILP has been expanded to almost all learning tasks, such as classification, regression, clustering, and correlation analysis, and has generated relational classification rules, relational regression trees, and relational learning based on distance and relationship-related rules. Representative systems include PROLOG (Muggleton, 1995) [58] and ALEPH (Srinivasan, 2000) [59]. Later, the characteristics of statistical learning theory and gradual research on Statistical Relational Learning (SRL) led to many relational learning methods based on SRL, such as SRL based on Bayesian networks, SRL based on Markov networks, SRL based on hidden Markov models, and SRL based on random grammar (Popescul, Ungar, Lawrence, & Pennock, 2003) [40, 41].

(1) SRL based on Bayesian networks

Ng and Subrahmanian proposed the first framework for Probabilistic Logic Programs (PLPs) [42] in 1992. This framework enabled the description of probabilistic information within the fixed-point semantics of PLP, which is a valuable tool for both probabilistic information and common sense reasoning. In 1997, Ngo and Haddawy combined the logic with Bayesian networks by defining probabilities based on first-order logic or relational interpretation. Their work enriched the PLP content. In 1996, Debabrata and Sumit Sarkar proposed the Probabilistic Relational Model (PRM) [43], which deals with uncertain information. They systematically described the theory of probabilistic databases and defined a series of concepts such as super keys, primary keys, candidate keys, probabilistic patterns, probabilistic relations, and probability relational databases. They also defined a whole train of operations such as projection, selection, difference, and renaming, and discussed how to realize probabilistic databases and the NULL problem. Their work also converted the partial probability distribution to the interval estimation and point estimation. PRM extended the standard Bayesian network using an entity relationship model as a basic representation framework. Today, PRM has successfully solved many problems in relational clustering, hypertext classification, and so on. In 2000, Kersting and De Raedt developed Bayesian Logic Programs (BLPs) [44] by combining logic programs with Bayesian networks. BLPs not only handle Boolean variables but also take continuous-valued variables. They can solve a number of instances of clauses that have the same head using a simplified form of the combination of rules.

(2) SRL based on Markov networks

In 2002, Taskar, Abbeel, and Koller proposed the Relational Markov Network (RMN) [45] by introducing the relationship between entities into the Markov network. RMN extended the Markov network to relational databases and has been used to classify

relational data. RMN defined the groups and potential functions between attributes of related entities at the template level, thus giving an arbitrary set of examples a distribution. In 2004, Richardson and Domingos proposed a Markov logic network (MLN) [46], which is a closed Markov network generated by a first-order logic knowledge database with a weighted value as a template. The advantage of MLN is that it introduced domain knowledge into Markov networks while providing a simple description language for large-scale networks. Additionally, MLN increased the uncertainty processing capability for first-order logic and served as a unified framework for many SRL tasks.

(3) RSL based on hidden Markov models

In 2002, Anderson and Domingos proposed a Relational Markov Model (RMM) [47] by introducing relations (logic) into Markov models. RMM allows state variable values to be of multiple types and uses one predicate or relationship to represent the same type of states. Thus, it overcomes the shortcoming of state variables whereby the Markov model can have only one variable. In 2003, Kersting proposed a logic hidden Markov model (LOHMM) [48] by introducing first-order logic into the Markov model. The differences from conventional Markov models are that there are two variables for the state and observation and the parameters have occurrence probabilities for each observation in addition to the transition probabilities.

(4) SRL based on random grammar

SRL [49] was proposed by Muggleton in 1996 as an extension of SLP by attaching a probability value to each clause in a random context-free grammar. In recent years, there have been many studies on SLP, such as a two-stage algorithm that learns the SLP parameters and a basic logic program proposed by Muggleton in 2000 to maximize the Bayesian posterior probability. In 2001, Cussens proposed Failure-Adjusted Maximization (FAM), which can estimate the SLP parameters on the premise that the basic logic program is given. In 2002, Muggleton proposed a method that can learn parameters and structure simultaneously. Another method based on random grammar is Programming In Statistical Modeling (PRISM) [50]. PRISM is a symbolic statistical modelling language that not only extends the probability of logic programs but also studies from examples using EM algorithms. In 1995, Sato introduced the rationale for the PRISM program through distributed semantics. In 1997, Sato and Kameya described the PRISM language and system and also gave an example to describe a hidden Markov model and Bayesian network [50]. In recent years, the study of PRISM includes a general method given by Sato in 2005. This general method first uses PRISM to write field-related logic programs, then applies the EM algorithm to learn. In 2005, Sato and Kameya proposed a new learning model based on FAM and a program conversion technique proposed by Cussens. To improve the efficiency of learning, this model adds some restrictions to the PRISM program.

1.5.2 Problem introduction

The above relational learning algorithms cannot solve the learning problem of dynamic fuzzy systems. The study of relational learning has been further developed with the introduction of fuzzy theory. For example, relational learning has been enriched by fuzzy predicates [51] and a fuzzy learning algorithm called Autonomic Learning (AL) has been proposed based on fuzzy reasoning rules [52]. AL has two shortcomings: it does not consider possible noise interference in the observed data and is limited because it only considers the influence of the n th step on the $n + 1$ th step. To solve these problems, a dynamic fuzzy relational learning (DFRL) algorithm is presented in this section.

Consider the following dynamic fuzzy inference formula:

Dynamic fuzzy rules:

If $(\overleftarrow{A}_1, \overrightarrow{A}_1)$ and $(\overleftarrow{A}_2, \overrightarrow{A}_2)$ and ... and $(\overleftarrow{A}_n, \overrightarrow{A}_n)$ then $(\overleftarrow{B}, \overrightarrow{B})$

In fact $(\overleftarrow{A}_1, \overrightarrow{A}_1)'$ and $(\overleftarrow{A}_2, \overrightarrow{A}_2)'$ and ... and $(\overleftarrow{A}_n, \overrightarrow{A}_n)'$

Conclusion $(\overleftarrow{B}, \overrightarrow{B})' = (\overleftarrow{A}, \overrightarrow{A})' \circ (\overleftarrow{R}, \overrightarrow{R})((\overleftarrow{A}, \overrightarrow{A}), (\overleftarrow{B}, \overrightarrow{B}))$

where

$$(\overleftarrow{A}, \overrightarrow{A})' = (\overleftarrow{A}_1, \overrightarrow{A}_1)' \wedge (\overleftarrow{A}_2, \overrightarrow{A}_2)' \wedge \dots \wedge (\overleftarrow{A}_n, \overrightarrow{A}_n)',$$

$$(\overleftarrow{A}, \overrightarrow{A}) = (\overleftarrow{A}_1, \overrightarrow{A}_1) \wedge (\overleftarrow{A}_2, \overrightarrow{A}_2) \wedge \dots \wedge (\overleftarrow{A}_n, \overrightarrow{A}_n),$$

$(\overleftarrow{A}, \overrightarrow{A})$, $(\overleftarrow{A}, \overrightarrow{A})'$, $(\overleftarrow{B}, \overrightarrow{B})$ are the DFSs on the universe $(\overleftarrow{U}, \overrightarrow{U})$, $(\overleftarrow{U}, \overrightarrow{U})$, $(\overleftarrow{V}, \overrightarrow{V})$, respectively. $(\overleftarrow{R}, \overrightarrow{R})((\overleftarrow{A}, \overrightarrow{A}), (\overleftarrow{B}, \overrightarrow{B}))$ defines the dynamic fuzzy relation between $(\overleftarrow{A}, \overrightarrow{A})$ and $(\overleftarrow{B}, \overrightarrow{B})$, abbreviated as $(\overleftarrow{R}, \overrightarrow{R})$.

In a complex objective world, it is often difficult to determine $(\overleftarrow{R}, \overrightarrow{R})$ in dynamic fuzzy inference rules. We propose a DFRL algorithm to learn $(\overleftarrow{R}, \overrightarrow{R})$ based on existing samples or other data obtained by means such as expert experience or monitoring system behaviour.

Given the observed data $(\overleftarrow{a}, \overrightarrow{a})$ of $(\overleftarrow{A}, \overrightarrow{A})$, $(\overleftarrow{b}, \overrightarrow{b})$ of $(\overleftarrow{B}, \overrightarrow{B})$, we have:

$$(\overleftarrow{b}, \overrightarrow{b}) = (\overleftarrow{a}, \overrightarrow{a}) \circ (\overleftarrow{R}, \overrightarrow{R}) \quad (1.31)$$

or

$$(\overleftarrow{b}_j, \overrightarrow{b}_j) = \max_{i=1}^N (\overleftarrow{a}_i, \overrightarrow{a}_i) \wedge (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}) \quad (1.32)$$

where $j = 1, 2, \dots, M$, $(\overleftarrow{a}, \overrightarrow{a}) \in [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]^N$, $(\overleftarrow{b}, \overrightarrow{b}) \in [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]^M$,

$$(\overleftarrow{R}, \overrightarrow{R}) \in [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]^N \times [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]^M.$$

The above can also be expressed as

$$(\overleftarrow{b}_j, \overrightarrow{b}_j) = (\overleftarrow{r}_j, \overrightarrow{r}_j)(\overleftarrow{a}, \overrightarrow{a}), \quad (1.33)$$

where $(\overleftarrow{b}_j, \overrightarrow{b}_j)$ is regarded as a result of applying some nonlinear transformation $(\overleftarrow{r}_j, \overrightarrow{r}_j)$ to $(\overleftarrow{a}, \overrightarrow{a})$.

The fuzzy relational learning algorithm model proposed in [50] is as follows:

$$\begin{cases} \theta(n+1) = \theta(n) + \delta\theta(n) \\ \delta\theta(n) = \varepsilon \bar{c} \bar{d}(x_k, \theta(k)) \\ \bar{d}(\bar{x}, \theta) = -\nabla l(\bar{x}, \theta) \\ l(\bar{x}, \theta) = \sum_{j=1}^M l(b_j, \bar{r}_j(\bar{a})) = \frac{1}{2} \sum_{j=1}^M (\bar{r}_j(\bar{a}) - b_j)^2 \end{cases}, \quad (1.34)$$

where $\delta\theta(n)$ is a correctional term and $n, n + 1$ represent successive steps of learning. The correctional term depends on the loss function $l(\bar{x}, \theta)$. Let the input vector be $a^* = [a_1^*, a_2^*, \dots, a_N^*]$ and the expected output vector be $b^* = [b_1^*, b_2^*, \dots, b_M^*]$. Thus, $\bar{x} = [a^*, b^*]$, $\theta = [r_{ij}]_{N \times M}$, $i = 1, 2, \dots, N$, $j = 1, 2, \dots, M$, θ is defined as an element of all fuzzy relationships R in vector form. \bar{c} is a positive matrix of size $NM \times NM$, which usually is a unit matrix. ε is a small positive number. $\bar{d}(x_k, \theta(k))$ is a search direction vector computed from \bar{x} and θ .

The above model is very limited because it predicts the learning of step $n + 1$ from step n alone. Hence, it cannot effectively guarantee the correctness and convergence of learning. In this section, we present a solution that takes the previous k steps of learning into account and effectively deals with noise. The value of k can be determined according to the scale of the learning and the requirements of the system.

1.5.3 DFRL algorithm

From the previous analysis, we should adjust the learning recursive formula and loss function of model (1.34). Thus, we can define the follow learning recursive formula:

$$(\overleftarrow{\theta}, \overrightarrow{\theta})(n+1) = \sum_{i=n-k+1}^n (\overleftarrow{w}_i, \overrightarrow{w}_i)[(\overleftarrow{\theta}, \overrightarrow{\theta})(i) + \Delta(\overleftarrow{\theta}, \overrightarrow{\theta})(i)], \quad (1.35)$$

$$\Delta(\overrightarrow{\theta}, \overrightarrow{\theta})(i) = (\overrightarrow{\epsilon}, \overrightarrow{\epsilon})(\overrightarrow{c}, \overrightarrow{c})\bar{d}((\overrightarrow{x}_i, \overrightarrow{x}_i), (\overrightarrow{\theta}, \overrightarrow{\theta})(i)), \quad (1.36)$$

$$\bar{d}((\overrightarrow{x}_i, \overrightarrow{x}_i), (\overrightarrow{\theta}, \overrightarrow{\theta})(i)) = -\nabla(\overrightarrow{l}, \overrightarrow{l})((\overrightarrow{x}, \overrightarrow{x}), (\overrightarrow{\theta}, \overrightarrow{\theta})), \quad (1.37)$$

where $(\overrightarrow{w}_i, \overrightarrow{w}_i)$ is a coefficient representing the influence level of step i on step $n + 1$; its value depends on the error generated in step i . The smaller the coefficient, the greater the error, and the coefficient should satisfy $\sum_{i=n-k+1}^n (\overrightarrow{w}_i, \overrightarrow{w}_i) = (\overrightarrow{1}, \overrightarrow{1})$. Assuming that the error in the step i learning is $(\overrightarrow{e}_i, \overrightarrow{e}_i)$, we have

$$(\overrightarrow{w}_i, \overrightarrow{w}_i) = \frac{[\sum_{i=n-k+1}^n (\overrightarrow{e}_i, \overrightarrow{e}_i)] - (\overrightarrow{e}_i, \overrightarrow{e}_i)}{(k-1) \sum_{i=n-k+1}^n (\overrightarrow{e}_i, \overrightarrow{e}_i)}. \text{ The value of } k (1 \leq k \leq n) \text{ can be determined}$$

according to the scale of the learning and the requirements of the system. $(\overrightarrow{\theta}, \overrightarrow{\theta}) = [(\overrightarrow{r}_{ij}, \overrightarrow{r}_{ij})]_{N \times M}$, $i = 1, 2, \dots, N$, $j = 1, 2, \dots, M$. $(\overrightarrow{c}, \overrightarrow{c})$ is a positive matrix of size $NM \times NM$, which is usually a unit matrix. $(\overrightarrow{\epsilon}, \overrightarrow{\epsilon})$ is a small positive number. Other variables and functions are defined as in (1.34).

When the observed data contain noise, the model can be expressed as follows:

$$\begin{cases} (\overrightarrow{a}, \overrightarrow{a}) = (\overrightarrow{a}, \overrightarrow{a})^* + n(\overrightarrow{a}, \overrightarrow{a}) \\ (\overrightarrow{b}, \overrightarrow{b}) = (\overrightarrow{b}, \overrightarrow{b})^* + n(\overrightarrow{b}, \overrightarrow{b}) \end{cases},$$

where $(\overrightarrow{a}, \overrightarrow{a})^*$, $(\overrightarrow{b}, \overrightarrow{b})^*$ are the true values of observation data $(\overrightarrow{a}, \overrightarrow{a})$, $(\overrightarrow{b}, \overrightarrow{b})$, respectively. $n(\overrightarrow{a}, \overrightarrow{a})$, $n(\overrightarrow{b}, \overrightarrow{b})$ are the noise components contained in the observed data, respectively. The variances are $(\overrightarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{a}, \overrightarrow{a})}$, $(\overrightarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{b}, \overrightarrow{b})}$, respectively. Thus, we can define the following loss function:

$$(\overrightarrow{l}, \overrightarrow{l})((\overrightarrow{x}, \overrightarrow{x}), (\overrightarrow{\theta}, \overrightarrow{\theta})) = \sum_{j=1}^M (\overrightarrow{l}_j, \overrightarrow{l}_j)((\overrightarrow{b}_j, \overrightarrow{b}_j), (\overrightarrow{r}_j, \overrightarrow{r}_j)(\overrightarrow{a}, \overrightarrow{a})) \quad (1.38)$$

$$\begin{aligned} & (\overrightarrow{l}_j, \overrightarrow{l}_j)((\overrightarrow{b}_j, \overrightarrow{b}_j), (\overrightarrow{r}_j, \overrightarrow{r}_j)(\overrightarrow{a}, \overrightarrow{a})) \\ &= \frac{1}{2} \left[\left(\frac{(\overrightarrow{r}_j, \overrightarrow{r}_j)(\overrightarrow{a}, \overrightarrow{a})' - (\overrightarrow{b}_j, \overrightarrow{b}_j)}{(\overrightarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{b}_j, \overrightarrow{b}_j)}} \right)_2 + \left(\frac{(\overrightarrow{a}, \overrightarrow{a})' - (\overrightarrow{a}, \overrightarrow{a})}{(\overrightarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{a}, \overrightarrow{a})}} \right)_2 \right] \\ &= \frac{1}{2} \left[\left(\frac{\max_{i=1}^N (\overrightarrow{a}_i, \overrightarrow{a}_i)' \wedge (\overrightarrow{r}_{ij}, \overrightarrow{r}_{ij}) - (\overrightarrow{b}_j, \overrightarrow{b}_j)}{(\overrightarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{b}_j, \overrightarrow{b}_j)}} \right)_2 + \left(\frac{(\overrightarrow{a}, \overrightarrow{a})' - (\overrightarrow{a}, \overrightarrow{a})}{(\overrightarrow{\delta}, \overrightarrow{\delta})_{(\overrightarrow{a}, \overrightarrow{a})}} \right)_2 \right], \end{aligned}$$

where $(\overleftarrow{\alpha}, \overrightarrow{\alpha})'$ is an estimate of the true value $(\overleftarrow{\alpha}, \overrightarrow{\alpha})^*$.

$$\begin{aligned}
 & \nabla(\overleftarrow{l}_j, \overrightarrow{l}_j)((\overleftarrow{b}_j, \overrightarrow{b}_j), (\overleftarrow{r}_j, \overrightarrow{r}_j)(\overleftarrow{\alpha}, \overrightarrow{\alpha})) \\
 &= \frac{\partial}{\partial(\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})} \left\{ \frac{1}{2} \left[\left(\frac{\max_{i=1}^N (\overleftarrow{\alpha}_i, \overrightarrow{\alpha}_i)' \wedge (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}) - (\overleftarrow{b}_j, \overrightarrow{b}_j)}{(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{b}_j, \overrightarrow{b}_j)}} \right)_2 \right. \right. \\
 & \quad \left. \left. + \left(\frac{(\overleftarrow{\alpha}, \overrightarrow{\alpha})' - (\overleftarrow{\alpha}, \overrightarrow{\alpha})}{(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{\alpha}, \overrightarrow{\alpha})}} \right)_2 \right] \right\} \\
 &= \frac{1}{2} \left(\frac{\max_{i=1}^N (\overleftarrow{\alpha}_i, \overrightarrow{\alpha}_i)' \wedge (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}) - (\overleftarrow{b}_j, \overrightarrow{b}_j)}{(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{b}_j, \overrightarrow{b}_j)}} \right)^2 \\
 & \quad \frac{\partial}{\partial(\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})} \max_{i=1}^N (\overleftarrow{\alpha}_i, \overrightarrow{\alpha}_i)' \wedge (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}) \tag{1.39}
 \end{aligned}$$

Because the max function is non-continuous and differentiable, the partial derivative of the above equation is problematic. Here, we introduce an approximation function to overcome this problem.

We define a generalized p mean for the data $(\overleftarrow{\alpha}, \overrightarrow{\alpha})' = [(\overleftarrow{\alpha}_1, \overrightarrow{\alpha}_1)', (\overleftarrow{\alpha}_2, \overrightarrow{\alpha}_2)', \dots, (\overleftarrow{\alpha}_N, \overrightarrow{\alpha}_N)']$:

$$M_p(\overleftarrow{\alpha}, \overrightarrow{\alpha})' = \left(\frac{1}{N} \sum_{i=1}^N [(\overleftarrow{\alpha}_i, \overrightarrow{\alpha}_i)']^p \right)^{1/p}. \tag{1.40}$$

$M_p(\overleftarrow{\alpha}, \overrightarrow{\alpha})'$ is a good approximation function [53] for $\max_i (\overleftarrow{\alpha}_i, \overrightarrow{\alpha}_i)'$ and $\min_i (\overleftarrow{\alpha}_i, \overrightarrow{\alpha}_i)'$, and has the following properties.

Lemma 1.2 [53]

- (1) $M_0(\overleftarrow{\alpha}, \overrightarrow{\alpha})' = \lim_{p \rightarrow 0} M_p(\overleftarrow{\alpha}, \overrightarrow{\alpha})' = [(\overleftarrow{\alpha}_1, \overrightarrow{\alpha}_1)' \dots (\overleftarrow{\alpha}_N, \overrightarrow{\alpha}_N)']^{1/N}$
- (2) $p < q \Rightarrow M_p(\overleftarrow{\alpha}, \overrightarrow{\alpha})' < M_q(\overleftarrow{\alpha}, \overrightarrow{\alpha})'$
- (3) $\lim_{p \rightarrow \infty} M_p(\overleftarrow{\alpha}, \overrightarrow{\alpha})' = \max[(\overleftarrow{\alpha}_1, \overrightarrow{\alpha}_1)', \dots, (\overleftarrow{\alpha}_N, \overrightarrow{\alpha}_N)']$
- (4) $\lim_{p \rightarrow -\infty} M_p(\overleftarrow{\alpha}, \overrightarrow{\alpha})' = \min[(\overleftarrow{\alpha}_1, \overrightarrow{\alpha}_1)', \dots, (\overleftarrow{\alpha}_N, \overrightarrow{\alpha}_N)']$
- (5) For each variable $(\overleftarrow{\alpha}_i, \overrightarrow{\alpha}_i)'$, $M_p(\overleftarrow{\alpha}, \overrightarrow{\alpha})'$ is continuously differentiable. The partial derivative of $M_p(\overleftarrow{\alpha}, \overrightarrow{\alpha})'$ with respect to $(\overleftarrow{\alpha}_i, \overrightarrow{\alpha}_i)'$ is

$$\frac{\partial M_p(\overleftarrow{\alpha}, \overrightarrow{\alpha})'}{\partial(\overleftarrow{\alpha}_i, \overrightarrow{\alpha}_i)'} = \frac{1}{N} \left(\frac{(\overleftarrow{\alpha}_i, \overrightarrow{\alpha}_i)'}{M_p(\overleftarrow{\alpha}, \overrightarrow{\alpha})'} \right)^{p-1}. \tag{1.41}$$

Supposing $p > 0$ and approximating $\max_{i=1}^N (\overrightarrow{a}_i, \overrightarrow{a}_i)'$ with $M_p(\overrightarrow{a}, \overrightarrow{a})'$, (1.39) can be written as

$$\nabla(\overleftarrow{l}_j, \overrightarrow{l}_j)((\overleftarrow{b}_j, \overrightarrow{b}_j), (\overleftarrow{r}_j, \overrightarrow{r}_j)(\overrightarrow{a}, \overrightarrow{a})) = \frac{1}{2} \left(\frac{M_p(\overrightarrow{a}_i, \overrightarrow{a}_i)' \wedge (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}) - (\overleftarrow{b}_j, \overrightarrow{b}_j)}{(\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{b}_j, \overrightarrow{b}_j)}} \right)^2 \frac{\partial}{\partial(\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})} M_p(\overrightarrow{a}_i, \overrightarrow{a}_i)' \wedge (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}). \quad (1.42)$$

In summary, DFRL can be used to generate or modify dynamic fuzzy rules R^l in a DFMLS. The algorithm can be summarised in Algorithm 1.4.

Algorithm 1.4 DFRL algorithm

Input: $(\overrightarrow{a}, \overrightarrow{a}) = [(\overrightarrow{a}_1, \overrightarrow{a}_1), (\overrightarrow{a}_2, \overrightarrow{a}_2), \dots, (\overrightarrow{a}_N, \overrightarrow{a}_N)]$
 $(\overleftarrow{b}, \overrightarrow{b}) = [(\overleftarrow{b}_1, \overrightarrow{b}_1), (\overleftarrow{b}_2, \overrightarrow{b}_2), \dots, (\overleftarrow{b}_M, \overrightarrow{b}_M)]$

Output: $(\overleftarrow{R}, \overrightarrow{R}) = [(\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})]_{N \times M}$

- (1) Initialization: Select an initial dynamic fuzzy relationship $(\overleftarrow{R}, \overrightarrow{R})$ that can have a random initialization value;
- (2) Repeat:
 - (2a) Set k to an initial value of 0. This parameter can be determined according to the scale of the learning and the requirements of the system.
 - (2b) Calculate the influence coefficient of the step i learning on the step $n + 1$ learning;

$$(\overleftarrow{w}_i, \overrightarrow{w}_i) = \frac{[\sum_{i=n-k+1}^n (\overrightarrow{e}_i, \overrightarrow{e}_i)] - (\overrightarrow{e}_i, \overrightarrow{e}_i)}{(k-1) \sum_{i=n-k+1}^n (\overrightarrow{e}_i, \overrightarrow{e}_i)},$$

where $(\overrightarrow{e}_i, \overrightarrow{e}_i)$ is the error generated by step i learning;

- (2c) Calculate (1.38);
- (2d) Calculate (1.42) from (1.40) and (1.41);
- (2e) Modify the current dynamic fuzzy relationship matrix $(\overleftarrow{R}, \overrightarrow{R})$ according to (1.42), (1.15), and (1.16);

Until Convergence

1.5.4 Algorithm analysis

Comparing the DFRL algorithm with the AL algorithm [52], DFRL accounts for the influence of the k th step of learning on the $n + 1$ th step. Although this increases the time complexity, it ensures the learning results are more credible and the error is smaller. In addition, DFRL solves the problem of possible noise interference in the observed data and overcomes the disadvantages of the lateral method. In general,

the DFRL algorithm is superior to the latter. It is not difficult to prove that the DFRL algorithm is convergent.

1.6 Summary

In this chapter, we have presented a DFMLM and its related algorithms based on DFSs. We have discussed a DFMLM and DFML algorithm, DFML geometric model, DFMLS parameter learning algorithm and MLE algorithm, DFMLS process control model, and a dynamic fuzzy relation learning algorithm.

References

- [1] Li FZ. Dynamic fuzzy logic and its applications. NY, USA, Nova Science Publishers, 2008.
- [2] Li FZ, Liu GQ, Yu YM. An introduction to dynamic fuzzy logic. Yunnan, China, Yunnan Science & Technology Press, 2005.
- [3] Li FZ, Zhu WH. Dynamic fuzzy logic and its application. Yunnan, China, Yunnan Science & Technology Press, 1997.
- [4] Li FZ, Shen QZ, Zheng JL, Shen HF. Dynamic fuzzy sets and its application. Yunnan, China, Yunnan Science & Technology Press, 1997.
- [5] Xie L, Li FZ. Research on the kind of dynamic fuzzy machine learning algorithm. *Acta Electronica Sinica*, 2008, 36(12A): 114–116.
- [6] Xie L, Li FZ. Research on an new dynamic fuzzy parameter learning algorithm. *Microelectronics & Computer*, 2008, 25(9): 84–87.
- [7] Xie L, Li FZ. Research on a new parameter learning algorithm. In *Proceedings of 2008 Internal Conference on Advanced Intelligence*, Beijing, China, Oct 18–22, 2008, 112–116.
- [8] Zhang J. Dynamic Fuzzy Machine Learning Model and Its Applications. Master's thesis, Soochow University, Suzhou, China, 2007.
- [9] Li FZ. Research on a kind of coordination machine learning model based on the DFS. *Computer Engineering*, 2001, 27(3): 106–110.
- [10] Li FZ. Research on stability of coordinating machine learning. *Journal of Chinese MiniMicro Computer Systems*, 2002, 23(3): 314–317.
- [11] Zhang J, Li FZ. Machine learning model based on dynamic fuzzy sets (DFS) and its validation. *Journal of Computational Information Systems*, 2005, 1(4): 871–877.
- [12] Roweis ST, Lawrence KS. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 2000, 290(5500): 2323–2326.
- [13] Liu Z. Research on dimension reduction in high dimensional data analysis. Master's thesis, National University of Defense Technology, Changsha, China, 2002.
- [14] Tan L, Wu X, Yi DY. Robust locally linear embedding. *Journal of National University of Defense Technology*, 2004, 26(6): 91–95.
- [15] Xiao J, Zhou ZT, Hu DW, Yin JS, Chen S. Self-organized locally linear embedding for nonlinear dimensionality reduction. In *Proceedings of ICNC2005*, 2005, 3610: 101–109.

- [16] Chen S, Zhou ZT, Hu DW. Diffusion and growing self-organizing map: A nitric oxide based neural model. In Proceedings of ISNN2004, 2004, 1: 199–204.
- [17] Li JX, Zhang Y, Fu X. A robust learning algorithm for noise data. *Journal on Numerical Methods and Computer Applications*, 2000, 6(2): 112–120.
- [18] Khalil H K. *Nonlinear systems* (Third Edition). Beijing, China, Publishing House of Electronics Industry, 2005.
- [19] Yang SZ, Wu Y. *Time series analysis in engineering application*. Wuhan, China, Huazhong University of Science & Technology Press, 1994.
- [20] Martin R, Heinrich B. A direct adaptive method for faster back propagation learning: The RPROP algorithm. In Proceedings of the IEEE International Conference on Neural Networks, 1993, 586–591.
- [21] Webb JM, Liu JS, Lawrence CE. BALS: Bayesian algorithm for local sequence alignment. *Nucleic Acids Research*, 2002, 30(6): 1418–1426.
- [22] Zu JK, Zhao CS, Dai GZ. An adaptive parameters learning algorithm for fuzzy logic systems. *Journal of System Simulation*, 2004, 16(5): 1108–1110.
- [23] Van Gorp J, Schoukens J, Pintelon R. Learning neural networks with noisy inputs using the errors in variables approach. *IEEE Transactions on Neural Networks*, 2000, 11(2): 402–413.
- [24] Yager RR, Filev DP. Approximate clustering via the mountain method. *IEEE Transactions on Systems, Man and Cybernetics*, 1994, 15(8): 1274–1284.
- [25] Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of Royal Statistics Society B*, 1977, 39: 1–38.
- [26] Moon TK. The expectation maximization algorithm. *IEEE Signal Processing Magazine*, 1996, 13(6): 47–60.
- [27] Matsuyama Y. The α -EM algorithm and its applications[C]// IEEE International Conference on Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. IEEE, 2000, 1: 592–595.
- [28] Fang YB. α -EM algorithm and its some applications. Master's thesis, Shanghai Jiao Tong University, Shanghai, China, 2004.
- [29] Matsuyama Y. The α -EM algorithm: Surrogate likelihood maximization using alogarithmic information measures. *IEEE Transactions on Information Theory*, 2003, 49(3): 692–706.
- [30] Zangwill W I, Mond B. *Nonlinear programming: A unified approach*. NJ, USA, Prentice Hall, 1969.
- [31] Luenberger DG. *Introduction to linear and nonlinear programming*. 2th ed. MA, USA, Addison Wesley Publishing Company, 1984.
- [32] Zhang WZ, Wang LB, Wang XR. Maximum likelihood estimation of nonlinear reproductive divergence data model parameters new exploration of theory. *Statistics and Decision Making*, 2004, 7: 6–7.
- [33] Tanaka K, Sugeno M. Stability analysis and design of fuzzy control systems. *Fuzzy Sets and Systems*, 1992, 45: 135–156.
- [34] Xie L, Souza CE. Criteria for robust stabilization of uncertain linear systems with time-varying states delays. In Proceedings of IFAC 13th Triennial word congress, USA, 1996, 137–142.
- [35] Wu M, Gui WH. *Advanced robust control*. ChangSha, China, Central South University Press, 1998.
- [36] Cai ZX, Xiao XM. Robust stability analysis and controller design for a class of dynamic fuzzy system. *Journal of Central South University*, 1999, 30(4): 418–421.

- [37] He XQ. Stability analysis and application of a class of multivariable fuzzy systems. PhD Thesis, Northeastern University, Shenyang, China, 2000.
- [38] Lavrac N, Dzeroski S. Inductive logic programming. New York, USA, Horwood Press, 1994.
- [39] Dzeroski S. Multi-relational data mining: An introduction. ACM SIGKDD Explorations Newsletter, 2003, 5(1): 1–16.
- [40] Popescul A, Ungar L H. Statistical relational learning for link prediction. Working Notes of the IJCAI2003 Workshop on Learning Statistical Models from Relational Data, 2003, 109–115.
- [41] Popescul A. Statistical learning from relational databases. PhD Thesis, University of Pennsylvania, USA, 2004.
- [42] Ng R, Subrahmanian VS. Probabilistic logic programming. *Information and Computation*, 1992, 101(2): 150–201.
- [43] Dey D, Sarkar S. A probabilistic relational model and algebra. *ACM Transaction on Database System*, 1996, 21(3): 339–369.
- [44] Kersting K, De Raedt L. Bayesian logic programs. In Proceedings of the Work in Progress Track at the 10th International Conference on Inductive Logic Programming, 2000.
- [45] Taskar B, Abbeel P, Koller D. Discriminative probabilistic models for relational data. In Proceedings of Conference on Uncertainty in Artificial Intelligence, Edmonton, 2002.
- [46] Richardson M, Domingos P. Markov logic networks. *Machine Learning*, 2006, 62(1): 107–136.
- [47] Anderson C, et al. Relational Markov Models and their application to adaptive web navigation. In Proceeding of the Eighth International Conference on Knowledge Discovery and Data Mining, 2002, 143–152.
- [48] Kersting K, et al. Towards discovering structural signatures of protein folds based on logical hidden Markov models. In Proceedings of the 8th Pacific Symposium on Biocomputing, 2003, 192–203.
- [49] Muggleton S. Stochastic logic programs. *Advances in Inductive Logic Programming*, 1996, 254–264.
- [50] Sato T, Kameya Y. PRISM: A symbolic-statistical modeling language. In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, 1997, 1330–1339.
- [51] Prade H, Richard G, Serrurier M. Enriching relational learning with fuzzy predicates, In Proceedings of PKDD2003, LNAI 2838, 2003, 399–410.
- [52] Wang ST. Learning algorithm AL of fuzzy inference rules based on fuzzy relation. *Computer Engineering and Applications*, 1995, 4: 57–59.
- [53] Rivest RL. Game tree searching by min/max approximation. *Artificial Intelligence*, 1987, 34(1): 1–2.
- [54] Muggleton S, Buntine W. Machine Invention of First-order Predicates by Inverting Resolution. *Machine Learning Proceedings*, 1988, 339–352.
- [55] Quinlan JR. Learning logical definitions from relations. *Machine Learning*, 1990, 5(3): 239–266.
- [56] Muggleton S, Feng C. Efficient induction of logic programs. *New Generation Computing*, 1990, 38.
- [57] Lavrac N, Dzeroski S, Grobelnik M. Learning nonrecursive definitions of relations with LINUS// European Working Session on Learning. Springer Berlin Heidelberg, 1991, 265–281.
- [58] Muggleton S, Mizoguchi F, Furukawa K. Special issue on inductive logic programming. *New Generation Computing*, 1995, 13(3-4): 243–244.
- [59] Srinivasan A. A study of two probabilistic methods for searching large spaces with ILP. Technical Report PRG-TR-16-00, 2000, Oxford University Computing Laboratory, Oxford.

2 Dynamic fuzzy autonomic learning subspace algorithm

This chapter is divided into four sections. Section 2.1 analyses the research status of Autonomic Learning. In Section 2.2, we present the theoretical system of autonomic learning subspace based on DFL. In Section 2.3, we provide learning algorithm of Autonomic Learning Subspace based on DFL. The summary of this chapter is in Section 2.4.

2.1 Research status of autonomic learning

Autonomic Learning (AL), from the aspect of learning theory, is a kind of self-directed learning mechanism composed of a learner's attitude, ability, and learning strategies. For example, it can refer to the ability of individuals to guide and control their learning, set learning goals, choose different learning methods and learning activities according to different tasks, monitor the learning process, evaluate the learning results, and help others according to the situation. AL usually refers to active, conscious, and independent learning, as opposed to passive, mechanical, and receptive learning. It is the basis of individual lifelong learning and lifelong development. AL has always been an important issue in education and psychology, and it is also a hot topic in the field of machine learning [1].

A learning system must be able to detect, plan, experiment, adapt, and discover [2]. These activities, in an integrated way, set a framework for an autonomous system to learn and discover from the environment: LIVE. LIVE consists of three modules: a prediction sequence generator (model application), model construction/modification, and implementation/perception (environmental interface). Prediction is used as the evaluation criterion for model construction; model construction (modification) provides a tool to improve the predictive ability; problem solving using the approximation model detects where and when to improve the model; the creation of a new term provides more components for creating, predicting, and problem solving. This study shows that it is an effective way to discover the process of merging activity from the environment. LIVE has three main advantages: (1) it defines the problem of learning from the environment as that of constructing an environment model in the context of problem solving. This definition distinguishes the internal actions of the individual from the results of the action on the environment. The approximation model of the environment is extracted from the “rough” space determined by the internal ability and a priori knowledge of the individual. The extraction process is guided by the information gathered during the interaction with the environment. (2) The framework provides an integrated approach for coordinating multiple

activities such as perception, behaviour, inquiry, experimentation, problem solving, learning, discovering, and creating new terms. LIVE is an execution system that combines these activities. (3) The key to identifying and integrating the framework is the concept of the prediction sequence and learning methods using complementary discrimination. The prediction sequence provides a joint window for planning, exploration, and experimentation. However, there are some shortcomings and deficiencies in the framework. For example, the framework cannot handle uncertainty, meaning that a single noise prediction failure can cause the model to be completely modified. It does not react in real time, and all actions are produced by careful consideration. Thus, it takes a lot of time to make decisions. The biggest limitation of LIVE is the exhaustive search method. The new terminology is too time-consuming and depends on two kinds of biases: (1) the terminology must be set in advance given useful mental relations and functions and (2) behaviour-dependent terminology methods are limited to considering only the features and objects that are relevant to the current conditions and behaviours.

In [3], an autonomous machine learning model based on rough set theory is proposed. In this paper, the problem of uncertainty in decision tables and decision rules is studied and the process of machine learning is controlled by the law that knowledge uncertainty should not change during machine learning. Thus, autonomic machine learning proposes an AL algorithm for default rule knowledge acquisition and the pre-pruning of the decision tree. Rough set theory can solve the fuzzy problem, but the result of solving the dynamic problem is not satisfactory.

Real-world problems can be described in terms of learning spaces. Learning problems can be mapped to multiple learning spaces, and a learning space can be used to describe multiple learning problems [4]. According to the degree of autonomy of the learning individual, the study space is further divided into autonomous and non-autonomous learning spaces. In the actual learning situation, completely independent and completely involuntary learning is rare, with most of the learning occurring somewhere between these two poles. Therefore, instead of dividing the learning space into autonomous or involuntary regions, it is better to consider the degree of autonomy of learning and then distinguish what learning is autonomous and what is not autonomous. This is more conducive to independent learning toward a target, in order to achieve better completion of learning tasks [5].

Thus, the AL system is a dynamic fuzzy system. Therefore, it is necessary to choose a theoretical tool that can solve the problem of dynamic fuzziness. To deal with these complex, dynamic, ambiguous, and unstructured properties in application systems, DFL has been introduced into autonomous machine learning methods, and some basic concepts of fuzzy logic have been proposed [6–8].

This chapter focuses on the following aspects of self-learning:

(1) Learning is a process of reasoning.

As knowledge increases, learning becomes the process of exploring and acquiring information in the knowledge space to satisfy the purpose of learning. It is a continuous process involving all kinds of reasoning and memorizing methods. During each iteration, the system extracts new knowledge or expressions from the input information and basic knowledge. If the acquired knowledge meets the learning goal, it is stored in the knowledge base and becomes part of the basic knowledge. The input information can be observed data, facts, concrete concepts, abstract concepts, knowledge structures, and information about the authenticity of knowledge.

Therefore, learning can be seen as a reasoning process. The process can be summarized as follows: autonomous machine learning based on DFL = independent reasoning based on DFL + memory.

(2) Learning is a transformation operator.

Learning is a process of exploring the knowledge space. This search is done by a series of knowledge transformation operators. The learning ability of a system is determined by the type and complexity of the transformation. According to different types of operations, knowledge transformation can be divided into knowledge generation transformation and knowledge processing transformation. Knowledge generation transformation modifies the content of knowledge by basic logic transformations such as substitution, addition, deletion, expansion, combination, decomposition, and so on, as well as the logical reasoning methods of extension, specialization, abstraction, concretization, analogy, and inverse analogy. Knowledge processing transformation concerns the original change in organizational structure, physical location, and other operations but does not change the content of knowledge. Thus, a learning process can be defined as follows:

Given: input knowledge (I), learning objectives (G), basic knowledge (BK);

Determine: output knowledge (O) to meet G by determining the appropriate transformation T to apply to I and BK.

(3) Learning goals are variable.

The goal is the desired achievement and result. The goal is also variable. From a psychological point of view, the need is the starting point of human behaviour and the source of the target. When a goal is achieved, it means that people have met their material or psychological needs. However, new needs will inevitably arise, becoming new goals. The variability of goals is also manifested in a seemingly unattainable goal, often through a cleverly achieved target workaround.

In the existing control structure, the goal is usually set to be constant and have some limitations. In the existing control structure, taking into account the variability of the control target, a closed-loop control structure diagram illustrates the appropriate transformation: the impact of the target disturbance. In addition to noise disturbance, some degree of target disturbance may exist. The former must be filtered and smoothed out, and the latter must be retained and processed.

(4) Learning environment is variable.

At present, computer science mainly considers the environment through human-machine interface design. However, the human-machine interface cannot adapt the demands of the user to the computer's ability. To break through this limitation, Brooks presented an artificial intelligence system in the mid-1980s, and Minsky proposed a multi-agent system, which Picard later employed for emotional computing. These problems involve the representation and processing of the environment. In fact, the consideration of environmental representations in adaptive computation is completely different from that in traditional control theory. In traditional control theory, the environment is often characterized by a mathematical description. In fact, the most realistic description of the environment is its own, and the use of other tools to characterize it is only an approximation of the environment itself. Presently, in adaptive computing, the representation of the environment emphasizes the use of true and direct representations.

The direct representation of the natural environment is simple, whereas the direct representation of the social environment must consider the observed behaviour of members of society. Besides the direct expression of the human environment, there is almost no other means of expression; thus, we can only adopt a direct expression similar to the social environment.

In this chapter, the representation of the environment and its transformation will be represented by a dynamic fuzzy model. The basic concepts of DFL are given in Appendix 8.3.

2.2 Theoretical system of autonomous learning subspace based on DFL

2.2.1 Characteristics of AL

1. Auto-knowledge

An autonomic learner can only be highly autonomous if he or she is at a high level of auto-awareness. Therefore, auto-knowledge is the basis for AL. The learner must have "auto-knowledge" to determine whether the current behaviour is compatible with

the learning environment. New conditions and a new environment will give rise to new behaviour; therefore, auto-knowledge should be constantly updated and evolve according to the environment. In other words, auto-knowledge is the internal knowledge base of the AL system [9]. With such auto-knowledge, the learner's self-learning ability will become stronger.

2. Auto-monitoring

Self-directed learning is a feedback cycle. AL individuals can monitor the effectiveness of their learning methods or strategies and adjust their learning activities based on this feedback. In the process of learning, auto-monitoring refers to the process of self-learning individuals assessing their learning progress using certain criteria. Auto-monitoring is a key process of AL. Self-directed learning is only possible when learners are self-monitoring [8]. Planning can be used to monitor the problem-solving process. Auto-monitoring strengthens the learner's self-learning ability.

3. Auto-evaluating

The auto-evaluating process is the most important component of AL because it influences not only the ability judgment, task evaluation, goal setting, and expectation of learners but also the individual auto-monitoring of the learning process and the learning result of the self-strengthening [8].

Existing research suggests that auto-evaluating includes self-summary, self-assessment, self-attribution, self-strengthening, and other sub-processes. Self-assessment involves comparing the learning results with the established learning objectives, to confirm which goals have been completed, which goals have not yet been achieved, and then to judge the merits of self-learning. Self-attribution refers to a self-assessment based on the results of learning success or failure, reflecting the reasons for the follow-up learning providing experience or lessons. Self-reinforcement is the process of rewarding or stimulating oneself based on the results of self-assessment and self-attribution, and it often motivates subsequent learning [5].

Behavioural evaluation is a way of simulating the behaviour of AL individuals. When solving a problem, two simple evaluation methods can be selected: the first is to choose the best behaviour to perform, that is, in the current state, that which acts most toward the direction of the best move; the second is to choose a better behaviour, that is, choose an act that offers the superior direction.

When learning an action, the learner autonomously evaluates his or her behaviour using the current evaluation function. The evaluation function is described as follows.

The original objective function value of the learner's self-learning is that a new situation is obtained after the execution of an action. The corresponding new objective function is denoted by $[1, 2, \dots, n]$, where n is the number of objective functions.

The historical evaluation function is adjusted as follows:

$$VY'_i = \alpha \times VY_i + (1 - \alpha)D_i \quad (2.1)$$

where VY_i denotes the historical evaluation function, VY'_i denotes the new evaluation function, and α denotes the weighting coefficient.

Auto-evaluation enhances the AL ability of the learner.

2.2.2 Axiom system of AL subspace [7]

The various problems in the real world can be described by learning space. That is mapping learning problems into learning space. A learning problem can be mapped into multiple learning spaces, and a learning space can be used to describe multiple learning problems [4].

Definition 2.1 Learning space: The space used to describe the learning process is called the learning space. It consists of five parts: {learning sample, learning algorithm, input data, output data, representation theory}, which can be expressed as $S = \{EX, ER, X, Y, ET\}$.

Definition 2.2 Learning subspace: A spatial description of a specific learning problem, that is, a subset of the learning space, is called the learning subspace.

The axiom system of the AL subspace consists of two parts: (1) a set of initial formulas, namely axioms, and (2) a number of deformation rules, namely deduction rules.

First, on the basis of DFL, we define AL as follows:

Definition 2.3 Autonomic learning: $(\overleftarrow{AL}, \overrightarrow{AL})$ refers to the mapping of an input dataset $(\overleftarrow{X}, \overrightarrow{X})$ to an output dataset $(\overleftarrow{Y}, \overrightarrow{Y})$ in an AL space, which can be expressed as $(\overleftarrow{AL}, \overrightarrow{AL}) : (\overleftarrow{X}, \overrightarrow{X}) \rightarrow (\overleftarrow{Y}, \overrightarrow{Y})$. According to $(\overleftarrow{AL}, \overrightarrow{AL})$, the process can be divided into 1:1 AL, n:1 AL, 1:n AL, and m:n AL.

AL can be divided into the following modes according to the knowledge stored in the knowledge base (KB): When the KB is the empty set, there is no a priori knowledge, only 1:1 or n:1 AL. As the learning process continues, the KB grows and updates itself, enabling 1:n or m:n AL.

Definition 2.4 Autonomic learning space: The space used to describe the learning process is composed of all AL elements, called the AL Space. It consists of five elements, which can be expressed as $(\overleftarrow{S}, \overrightarrow{S}) = \{(\overleftarrow{Ex}, \overrightarrow{Ex}), ER, (\overleftarrow{X}, \overrightarrow{X}), (\overleftarrow{Y}, \overrightarrow{Y}), ET\}$, where $(\overleftarrow{S}, \overrightarrow{S})$ represents the AL space, $(\overleftarrow{Ex}, \overrightarrow{Ex})$ denotes the AL samples, ER is an AL algorithm, $(\overleftarrow{X}, \overrightarrow{X})$ and $(\overleftarrow{Y}, \overrightarrow{Y})$ are the input and output data, respectively, and ET is the learning theory.

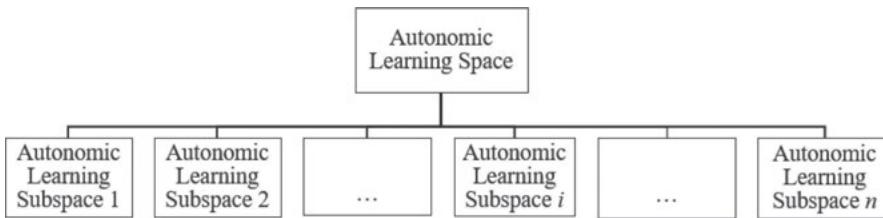


Fig. 2.1: Diagram of relation between Autonomic learning space and learning subspace.

Definition 2.5 Autonomic learning subspace (ALSS): According to some standard, we divide the AL space $(\overleftarrow{S}, \overrightarrow{S})$ into n subspaces: $(\overleftarrow{S}_1, \overrightarrow{S}_1)$, $(\overleftarrow{S}_2, \overrightarrow{S}_2)$, ..., $(\overleftarrow{S}_i, \overrightarrow{S}_i)$, ..., $(\overleftarrow{S}_n, \overrightarrow{S}_n)$, that is, $(\overleftarrow{S}, \overrightarrow{S}) = \{(\overleftarrow{S}_1, \overrightarrow{S}_1), (\overleftarrow{S}_2, \overrightarrow{S}_2), \dots, (\overleftarrow{S}_i, \overrightarrow{S}_i), \dots, (\overleftarrow{S}_n, \overrightarrow{S}_n)\}$, $i = 1, 2, \dots, n$.

ALSS is a learning subspace with the highest and lowest limits, namely, $S_{\min}, S_{\max} \in R^n$, $(\overleftarrow{S}_{\min}, \overrightarrow{S}_{\min}) \leq (\overleftarrow{S}_i, \overrightarrow{S}_i) \leq (\overleftarrow{S}_{\max}, \overrightarrow{S}_{\max})\}$. In the algorithm, it can be regarded as the subspace of algorithm optimization. In this subspace, some AL individuals are distributed, and the corresponding search ability of the individuals is given, so that they can carry out AL in the corresponding environment.

The relationship between the AL space and ALSS is depicted graphically in Fig. 2.1.

The partition given in Definition 2.5 is complete, and the axioms can be derived as follows:

Axiom 2.1 $(\overleftarrow{S}, \overrightarrow{S}) = \bigcup_{i=1,n} (\overleftarrow{S}_i, \overrightarrow{S}_i)$.

Axiom 2.2 $\forall (\overleftarrow{S}_i, \overrightarrow{S}_i) \in (\overleftarrow{S}, \overrightarrow{S}), i = 1, 2, \dots, n, (\overleftarrow{S}_i, \overrightarrow{S}_i) \subseteq (\overleftarrow{S}_i, \overrightarrow{S}_i)$.

Axiom 2.3 If $(\overleftarrow{S}_i, \overrightarrow{S}_i) \subseteq (\overleftarrow{S}_j, \overrightarrow{S}_j)$, $(\overleftarrow{S}_j, \overrightarrow{S}_j) \subseteq (\overleftarrow{S}_i, \overrightarrow{S}_i)$, $i, j = 1, 2, \dots, n$, then $(\overleftarrow{S}_i, \overrightarrow{S}_i) = (\overleftarrow{S}_j, \overrightarrow{S}_j)$.

Axiom 2.4 If $(\overleftarrow{S}_i, \overrightarrow{S}_i) \subseteq (\overleftarrow{S}_j, \overrightarrow{S}_j)$, $(\overleftarrow{S}_j, \overrightarrow{S}_j) \subseteq (\overleftarrow{S}_k, \overrightarrow{S}_k)$, $i, j, k = 1, 2, \dots, n$, then $(\overleftarrow{S}_i, \overrightarrow{S}_i) = (\overleftarrow{S}_k, \overrightarrow{S}_k)$.

2.3 Algorithm of ALSS based on DFL

ALSS is a learning subspace with the highest and lowest limits, which can be regarded as the subspace for optimization in the algorithm. Some AL individuals are scattered about this subspace, and as the AL progresses, their AL ability will become stronger and stronger.

2.3.1 Preparation of algorithm

AL individuals in the ALSS can be seen as vectors.

1. Definition of variables

The state of an AL individual can be expressed by the vector $(\overleftarrow{X}, \overrightarrow{X}) = \{(\overleftarrow{x}_1, \overrightarrow{x}_1), (\overleftarrow{x}_2, \overrightarrow{x}_2), \dots, (\overleftarrow{x}_n, \overrightarrow{x}_n)\}$, where $(\overleftarrow{x}_i, \overrightarrow{x}_i)$ ($i = 1, 2, \dots, n$) is the variable to be optimized; the objective function value $Y = f((\overleftarrow{X}, \overrightarrow{X}))$ represents the learner's current object of interest, *Learner_consistence*; $d_{ij} = \|(\overleftarrow{X}_i, \overrightarrow{X}_i) - (\overleftarrow{X}_j, \overrightarrow{X}_j)\|$ represents the distance between AL individuals; *Visual* represents the perceived distance of AL individuals; *Step* represents the maximum number of steps moved by AL individuals; and δ represents the congestion degree of AL individuals.

The neighbourhood of AL organization is defined as follows:

$$C = \{(\overleftarrow{x}, \overrightarrow{x}) \in (R^n, R^n) \mid \|(\overleftarrow{x}, \overrightarrow{x}) - (\overleftarrow{x}_s, \overrightarrow{x}_s)\|^2\} \leq D. \quad (2.2)$$

2. Description of behaviour

(1) Setting target behaviour

Let the current state of AL individual *Learner* be $(\overleftarrow{X}_i, \overrightarrow{X}_i)$, and choose a state $(\overleftarrow{X}_j, \overrightarrow{X}_j)$ at random in the corresponding sensing range. If Y_i is less than Y_j , go forward a step in that direction; otherwise, choose another state $(\overleftarrow{X}_j, \overrightarrow{X}_j)$ at random to judge whether or not the forward condition is satisfied. If the forward conditions are not met after a specified number of attempts, then perform some other behaviour (such as moving a step at random).

The pseudocode description of this process is shown in Algorithm 2.1, where $Random(N((\overleftarrow{X}_i, \overrightarrow{X}_i), Visual))$ represents a randomly selected neighbour in the neighbourhood of the visual distance of $(\overleftarrow{X}_i, \overrightarrow{X}_i)$.

Algorithm 2.1 Setting target behaviour

```
float Learner::Learner_setgoal()
{
    for (i = 0; i < try_number; i++)
    {
        (\overleftarrow{X}_j, \overrightarrow{X}_j) = Random(N((\overleftarrow{X}_i, \overrightarrow{X}_i), Visual));
        if (Y_i > Y_j)
            {(\overleftarrow{X}_{i|next}, \overrightarrow{X}_{i|next}) = (\overleftarrow{X}_j, \overrightarrow{X}_j);
            return Learner_consistence ((\overleftarrow{X}_{i|next}, \overrightarrow{X}_{i|next}));}
        }
    (\overleftarrow{X}_{i|next}, \overrightarrow{X}_{i|next}) = (\overleftarrow{X}_i, \overrightarrow{X}_i);
    return Learner_consistence ((\overleftarrow{X}_{i|next}, \overrightarrow{X}_{i|next}));}
```

(2) Aggregation behaviour

Let $(\overleftarrow{X}_i, \overrightarrow{X}_i)$ be the current state of AL individual *Learner*, and let n be the number of learning partners while exploring the current neighbourhood (which is $d_{ij} < \text{Visible}$). If $n_m / N < \delta$ ($0 < \delta < 1$), the learning partner centre has more learning objects but is not crowded. Under the condition $Y_i < Y_c$, move a step toward the centre of the learning partner; otherwise, perform other behaviours (such as setting the target behaviour).

The pseudocode description of this process is shown in Algorithm 2.2:

Algorithm 2.2 Aggregation behaviour

```
float Learner::Learner_gather()
{
     $n_f = N((\overleftarrow{X}_i, \overrightarrow{X}_i), \text{Visual});$ 
     $(\overleftarrow{X}_c, \overrightarrow{X}_c) = \text{Center}(N((\overleftarrow{X}_i, \overrightarrow{X}_i), \text{Visual}));$ 
    If ( $\frac{n_m}{N} < \delta \ \&\& \ Y_i < Y_c$ )
         $(\overleftarrow{X}_{i|next}, \overrightarrow{X}_{i|next}) = (\overleftarrow{X}_c, \overrightarrow{X}_c);$ 
    else
        Learner_setgoal();
    return Learner_consistence  $((\overleftarrow{X}_{i|next}, \overrightarrow{X}_{i|next}))$ ;
}
```

(3) Imitating behaviour

Let $(\overleftarrow{X}_i, \overrightarrow{X}_i)$ be the current state of AL individual *Learner*, and explore the neighbour $(\overleftarrow{X}_{\max}, \overrightarrow{X}_{\max})$ with the best state in the current neighbourhood.

Let n be the number of learning partners of the neighbour $(\overleftarrow{X}_{\max}, \overrightarrow{X}_{\max})$. If $Y_i < Y_{\max}$ with $\frac{n_m}{N} < \delta$ ($0 < \delta < 1$), then there is high *Learner_consistence* near the learning partner $(\overleftarrow{X}_{\max}, \overrightarrow{X}_{\max})$ without being crowded. Thus, the *Learner* moves a step toward the learning partner $(\overleftarrow{X}_{\max}, \overrightarrow{X}_{\max})$; otherwise, some target setting behaviour is performed.

The pseudocode description of this process is shown in Algorithm 2.3.

Algorithm 2.3 Imitating behaviour

```
float Learner::Learner_follow()
{
     $Y_{\max} = \text{MAX}(f((\overleftarrow{X}_{\max}, \overrightarrow{X}_{\max})), (\overleftarrow{X}_{\max}, \overrightarrow{X}_{\max}) \in N((\overleftarrow{X}_i, \overrightarrow{X}_i), \text{Visual});$ 
     $n_m = N((\overleftarrow{X}_{\max}, \overrightarrow{X}_{\max}), \text{Visual});$ 
    if ( $Y_i < Y_{\max} \ \&\& \ \frac{n_m}{N} < \delta$ )
         $(\overleftarrow{X}_{i|next}, \overrightarrow{X}_{i|next}) = (\overleftarrow{X}_{\max}, \overrightarrow{X}_{\max});$ 
    else
        Learner_setgoal();
    return Learner_consistence  $((\overleftarrow{X}_{i|next}, \overrightarrow{X}_{i|next}))$ ;
}
```

(4) Random behaviour

The realization of random behaviour is relatively simple. A state in a smaller neighbourhood is selected at random, and *Learner* moves in that direction. This ensures that the AL object will not be far away from the better ALSS in the blind moving process. Other autonomic learners who did not find the learning goal would conduct stochastic learning behaviour in their neighbourhood D.

In fact, the random learning behaviour is the default target setting behaviour.

(5) Auto-evaluating behaviour

After learning an action, the learner autonomously evaluates his or her behaviour using the current evaluation function. The evaluation function is described as follows.

Let Y_i be the original objective function value of *Learner*, and consider a new situation obtained after performing an action. The corresponding new objective function value is Y'_i , denoted as $D_i = Y_i - Y'_i$, where $i \in [1, 2, \dots, n]$ and n denotes the number of objective functions.

The historical evaluation function is adjusted as follows:

$$VY'_i = \alpha \times VY_i + (1 - \alpha)D_i, \quad (2.3)$$

where VY_i denotes the historical evaluation function, VY'_i denotes the new evaluation function, and α denotes the weighting coefficient.

This auto-evaluation process strengthens the AL ability of *Learner*. The pseudocode description of this process is shown in Algorithm 2.4.

Algorithm 2.4 Auto-evaluating behaviour

```
float Learner:: Learner_evaluate()
{
     $VY'_i = \alpha \times VY_i + (1 - \alpha)D_i;$ 
     $VY'_j = \alpha \times VY_j + (1 - \alpha)D_j;$ 
    if ( $VY'_i < VY'_j$ )
        ( $\overleftarrow{X}_i, \overrightarrow{X}_i$ ) = ( $\overleftarrow{X}_j, \overrightarrow{X}_j$ );
    else
        Learner_setgoal();
}
```

2.3.2 Algorithm of ALSS based on DFL

In the process of learning, the changing environment and lack of complete information make it difficult to predict the exact model of the environment. Traditional supervised learning methods cannot perform behaviour learning in an unknown

environment, so *Learner* must exhibit intelligent behaviour. Intensive learning, as an atypical learning method, provides agents with the ability to select the optimal action in the Markov environment, and obviates the need to establish the environment model. This approach has been widely used in the field of machine learning with a certain degree of success. However, intensive learning has some limitations: (1) single-step learning is inefficient and can only accept discrete state inputs and produce discrete actions, but the environment in which machine learning is usually spatially continuous. (2) Discretization of continuous state space and motion space often leads to information loss and dimension disaster. (3) The computational complexity increases exponentially with the increase of the state-action pairs. Trial-and-error interactions with the environment can bring about losses in the system.

In view of the above limitations of reinforcement learning, this section proposes an AL mechanism based on DFL. According to the learner's AL performance, the learning mechanism is decomposed into multi-step local learning processes. The state space is divided into a finite number of categories, which reduces the number of state-action pairs. After receiving an input state vector, the learner chooses an action to execute through a dynamic fuzzy reasoning system, and continuously updates the weight of the result in the rule base until convergence. Thus, a complete rule base can be obtained, which provides prior information for the learner's behaviour.

1. Reasoning model of DF

The reasoning process of DFL attempts to solve the problem of dynamic fuzziness. The model can be expressed as:

$$DF(A) \xrightarrow{DF(R)} DF(C), \quad (2.4)$$

$$DF(A_1) \wedge DF(A_2) \wedge \dots \wedge DF(A_n) \xrightarrow{DF(R)} DF(C), \quad (2.5)$$

$$DF(A_1) \vee DF(A_2) \vee \dots \vee DF(A_n) \xrightarrow{DF(R)} DF(C), \quad (2.6)$$

where $DF(A)$, $DF(A_1)$, $DF(A_2)$, ..., $DF(A_n)$ are the prerequisites or conditions of dynamic fuzzy reasoning, $DF(C)$ is the conclusion or posterior of dynamic fuzzy reasoning, and $DF(R)$ is the dynamic fuzzy relation between the premise and the conclusion satisfied by $DF(R) \in [0, 1] \times [\leftarrow, \rightarrow]$. The premises and conclusions are facts represented by DFL.

The rule base for a dynamic fuzzy inference system is generally composed of n rules in the following form:

$$R_j: \text{if } DF(A_1) \text{ and } DF(A_2) \text{ and } \dots \text{ and } DF(A_n) \text{ then } DF(C), DF(R), \quad (2.7)$$

where R_j represents the j th rule.

2. Description of algorithm

The goal of Q-learning is to learn how to choose the suboptimal or optimal action according to the external evaluation signal in the dynamic environment. The essence is of a learning process of dynamic decisions. However, the relevant environmental information may be lost when the learner does not have complete knowledge of the environment. Thus, a trial-and-error method must be used, and so the efficiency of the algorithm is not high. Sometimes, learning in an unknown environment can also involve some risks. One way to reduce these risks is to use an environmental model. The environmental model can be built using the experience gained by performing the relevant tasks before. Using environmental models can facilitate the results of the action without risking losses.

The environment model is a function from the state and action (S_{t+1}, a) to the next state and the enhanced value (S_{t+1}, r) . The model is established in the following two ways: first, the agent builds the model offline using the data provided in the initial stage of the learning; second, the agent establishes or perfects the environment model online in the process of interacting with the environment. We consider the second method in AL.

In this chapter, we add the function $E : S \times A \rightarrow R$, which represents the knowledge of experience. This function influences the learner's choice of action in the process of AL, thus accelerating the convergence speed and learning speed of the algorithm.

Definition 2.6 Experience: Experience is represented by a four-tuple $\{(\overleftarrow{S}_t, \overrightarrow{S}_t), a_t, (\overleftarrow{S}_{t+1}, \overrightarrow{S}_{t+1}), r_t\}$ indicating that an action a_t is performed at state $(\overleftarrow{S}_t, \overrightarrow{S}_t)$, generating a new state $(\overleftarrow{S}_{t+1}, \overrightarrow{S}_{t+1})$ and obtaining an enhanced signal r_t at the same time.

The empirical function $E((\overleftarrow{S}, \overrightarrow{S}), a)$ in the AL algorithm records the relevant empirical information about performing a in state $(\overleftarrow{S}, \overrightarrow{S})$. The most important problem when adding an empirical function to an algorithm is how to obtain empirical knowledge in the initial stage of learning, that is, how to define the empirical function $E((\overleftarrow{S}, \overrightarrow{S}), a)$. The answer mainly depends on the specific field to which the algorithm is applied. For example, in the case of agent-based routing optimization, when the agent collides with the wall, the corresponding knowledge is acquired. That is, the agent obtains empirical knowledge about the environmental model online in the process of interacting with the environment.

We assume that the rule base described in the previous section has been established. This rule base can be used by the AL individual to select an action. According to the output value and empirical function of the dynamic fuzzy reasoning system for action selection, the environment goes into the next state and returns a certain

value. According to the return value of this action, the result weights are updated in the KB.

The pseudocode of the autonomic subspace learning algorithm based on DFL is shown in Algorithm 2.5.

The demand for initial values in the algorithm is not high: usually, initializing a random distribution of *Learners* in the variable domain using ::Learner_init(); the algorithm can be terminated according to the actual situation, such as whether the variance between the values of successive results is less than the expected error.

The characteristics of our autonomic subspace learning algorithm based on DFL can be summarized as follows:

- (1) The autonomic subspace learning algorithm based on DFL takes the learning objective as the objective function of the algorithm in the learning process (equivalent to the optimization process). The algorithm only compares the objective function values and does not consider other specific problems of the traditional optimization algorithm (such as derivative information). The nature of the objective function is not computationally complex. Therefore, the ALSS algorithm based on DFL can be considered a type of “black box” algorithm.
- (2) Without high demands on the initial value, the algorithm only needs to initialize a number of AL individuals at random in the learning subspace to generate the initial value of the algorithm.
- (3) In the process of optimization, there are relatively few parameter settings, mainly related to giving the behaviour process of the individuals a larger allowable range.
- (4) The autonomic subspace learning algorithm based on DFL is very robust, which means that the algorithm gives similar results to the same problem under different run conditions.

2.3.3 Case analysis

1. Experiment description

Using Java as a development tool, the autonomic subspace learning algorithm was applied in a 10×10 grid environment. The experimental environment is shown in Fig. 2.2. Each square represents a state of the *Learner*. Box 2 is the *Learner*’s initial position, Box 3 represents the *Learner*’s target position, and Region 1 is the obstacle. The obstacles and targets in the environment are static. For the *Learner*, the environment (obstacles, boundaries, and the location of the target) is unknown. In the *Learner*-centred two-dimensional space, the four directions of motion are equally distributed. The *Learner*’s four directions of movement, East, West, South, and North, represent four optional actions. If the *Learner* encounters obstacles or boundaries, then he must return to the original state.

Algorithm 2.5 Autonomic subspace learning algorithm

```
procedure ASL
    Event-Handler(T);
    if receive(Event-Handler)
        Activate(Learner);
    else
        return;
    end if
    :: Learner_init();
    while the result is satisfied do
        switch (:: Learner_evaluate())
            case value1:
                :: Learner_follow();
            case value2:
                :: Learner_gather();
            default:
                :: Learner_setgoal();
        Decompose T: T = {T1, T2, ..., Tn};
        while (i<=n)
            Visit the ( $\overleftarrow{S}_t$ ,  $\overrightarrow{S}_t$ ) state;
            Set goal function Yi;
            Select an action a using the action choice rule of DFRS and Experience Function;
            Update state and goal function: ( $\overleftarrow{S}_{t+1}$ ,  $\overrightarrow{S}_{t+1}$ ), Y'i;
            if (Dj = 0)
                return;
            end if
        end while
    end switch
    :: Learner_move();
    get result();
end while
end ASL
```

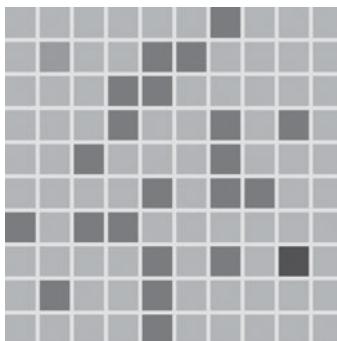


Fig. 2.2: Environment of AL experiment (10 × 10).

The parameters of the standard Q-learning algorithm and autonomic subspace learning algorithm based on DFL were set to $\alpha = 0.1$, $r = \{-100, 100, 0\}$, and the corresponding condition was {hit obstacles, reach the target, other}.

2. Experiment results

Using the autonomic subspace learning algorithm based on DFL (denoted as Method I) and Q-learning algorithm (denoted Method II), we compared the time cost and average number of search steps (L) required for the *Learner* to reach the target. The experimental results are shown in Figs. 2.3 and 2.4. The Episodes in Reinforcement Learning denote the learning cycle, i.e. the process in which the *Learner* moves from the initial position to the target position. Steps represents the number of steps taken by the *Learner* in moving from the initial position to the target position in each learning cycle.

In Fig. 2.3, it can be seen that, with the increase in the number of experiments, the autonomic subspace learning algorithm based on DFL required much less time than the Q-learning method.

Figure 2.4 shows that, with the increase in the number of experiments, the average number of search steps in method I is much smaller than that in method II.

In summary, the experiment shows that, in the 10×10 grid, the autonomic subspace learning algorithm based on DFL achieves a faster learning rate and better convergence ability than the ordinary Q-learning method.

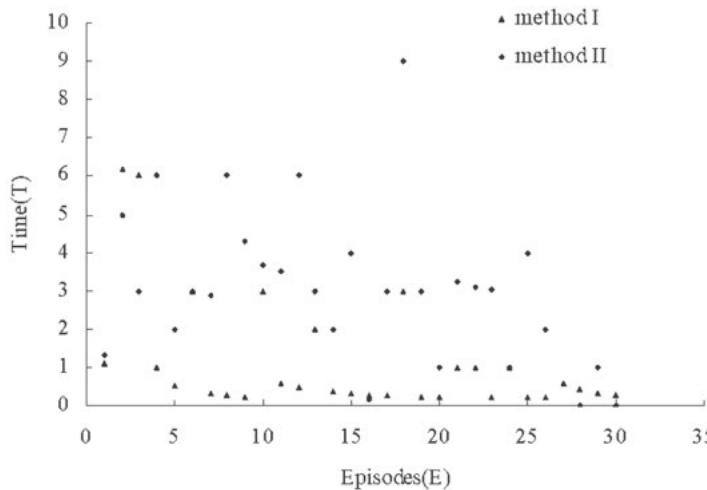


Fig. 2.3: Experimental environment results comparison 1.

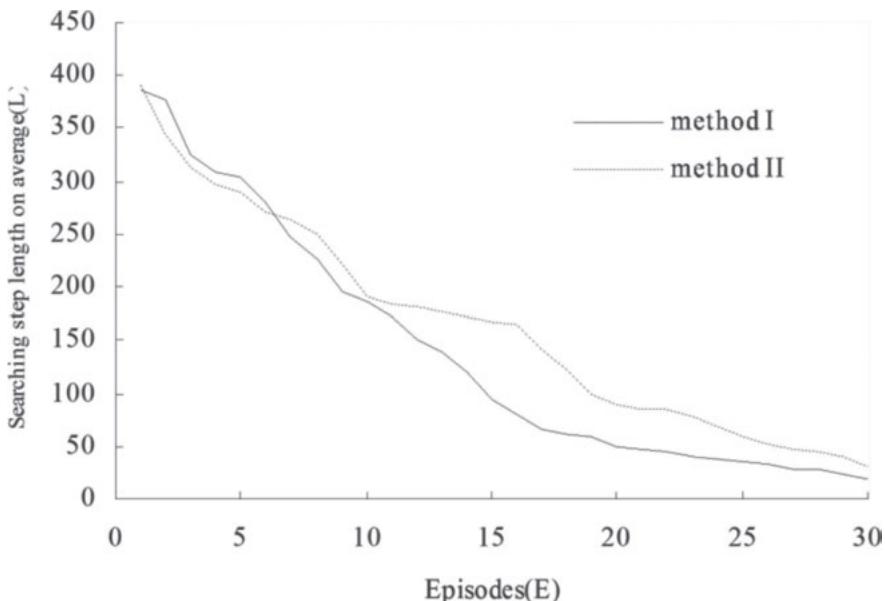


Fig. 2.4: Experimental environment results comparison 2.

2.4 Summary

In this chapter, we introduced a subspace learning algorithm for AL. The contribution of this chapter has two aspects. (1) The axiom system of ALSS uses DFL. The relevant axiom for an AL system was established, and a full axiom system and related content were derived. This provides the theoretical basis for solving the problem of machine learning activity and dynamic fuzziness in the system. (2) An autonomic subspace learning algorithm based on DFL was proposed.

References

- [1] Pang WG. Self-regulated learning. Shanghai, China, East China Normal University Edition, 2003.
- [2] Shen WM. Discovery as autonomous learning from the environment. Machine Learning, 1993, 12(8): 143–165.
- [3] Wang GY, He X. Initiative machine learning based on rough set. Computer Science, 2002, 29(9): 24–26.
- [4] Chen F, Li FZ. A Preliminary study on the theory and algorithm of learning subspace orbital generation for lie group machine learning. Journal of Soochow University (Nature Science Edition), 2007, 23(1): 61–66.
- [5] Wang J. Research and application on learning algorithm of autonomic learning subspace based on DFL. Master's thesis, Soochow University, Suzhou, China, 2008.

- [6] Wang J, Li FZ. Axiomatic system of autonomic learning subspace based on DFL. *Computer Science*, 2008, 35(12): 146–148.
- [7] Wang J, Li FZ. Autonomic learning model and algorithm based on DFL. In *Proceedings of 2007 IEEE International Conference on Granular Computing*, Silicon Valley, USA, 2007, 259–265.
- [8] Cofino T, Doganata Y, Drissi Y, et al. Towards knowledge management in autonomic systems. In *Proceedings of the Eighth IEEE International Symposium on Computers and Communication*, 2003, 2: 789–794.
- [9] Li FZ. *Dynamic fuzzy logic and its applications*. New York, USA, Nova Science Publishers, 2008.

3 Dynamic fuzzy decision tree learning

This chapter is divided into six sections. Section 3.1 analyses the research status of decision tree. In Section 3.2, we present the decision tree methods of dynamic fuzzy lattice. In Section 3.3, we provide dynamic fuzzy decision trees (DFDTs) special attribute processing technique. Section 3.4 presents the pruning strategy of DFDT. In Section 3.5, we introduce the application. The summary of this chapter is in Section 3.6.

3.1 Research status of decision trees

3.1.1 Overseas research status

1. Decision tree algorithms

An Attribute Value Taxonomies Decision Tree Learning algorithm [1] uses the accuracy of attribute values in instances to construct a decision tree across multiple layers. The algorithm has higher classification accuracy for multi-layer instances, and, when there are more missing values, the algorithm achieves better results than C4.5, which is more compact than the decision tree obtained by other methods. A learning algorithm for a Fuzzy Decision Tree [2] has been developed by constructing the tree based on the uncertainty of the decision boundary in terms of certain properties. This paper proposes a method to obtain classification confidence estimates as well as determine the location and associated uncertainty of each decision boundary. The method was applied to four machine learning datasets, and the results showed improved classification accuracy and tree size. A Soft Decision Tree (SDT) method using fuzzy sets has also been proposed [3]. SDT determines the structure of the tree by merging the growth and pruning processes of the tree and modifies the tree to improve its universality. The method proposed in [4] constructs a decision tree depending on the category of an attribute subset. Attribute selection is an important factor in improving the accuracy of the classifier, so this method retains the attribute classification information. A subset of the original attribute set is then obtained to construct the decision tree. As the information contained in the subset is more concentrated, the accuracy of the classifier is improved, and good results have been achieved in pattern recognition experiments. A method of constructing a decision tree using discrete continuous-valued attributes in the numerical domain has been proposed [5]. The continuous-valued attributes are first discretized, and then the decision tree is constructed with an estimated value for each interval. This method is used to construct a decision tree on the dataset in the numerical domain and offers significant improvements in speed and accuracy. At the same time, a practical pre-pruning method has been developed to deal with numerical attributes, and highly accurate results have been obtained.

2. Decision tree pruning

Decision tree pruning methods are summarized in [6, 7] in terms of theoretical basis, computational complexity, advantages and disadvantages. A large number of experimental data are used to objectively evaluate excessive and insufficient pruning methods with the aim of reducing the error inherent in current pruning methods [8]. presented the DI pruning method. Most pruning methods reduce the classification error rate, but in uncertain domains, many subtrees that cannot decrease the error rate may have some special effects. The DI pruning method takes into account the complexity of the subtree, and determines the corresponding decision rules from the leaf nodes of subtrees. At the same time, the DI method can be used to evaluate the quality of data for knowledge discovery. This method has been applied to the UnDeT software. [9] defined a unified framework based on the pruning method and maps the pruning process to an optimization problem. The framework can be used to learn the pruning method, leading to an experimental analysis of post-pruning in terms of prediction accuracy and tree size. In addition to construction and tree pruning methods, there have been many studies on attribute processing and applications in decision trees, such as IBM's processing schemes for missing values in data mining decision rules. Decision trees have also been applied to the classification of chemical and biological structures [10].

3.1.2 Domestic research status

1. Decision tree algorithms

The decision tree algorithms proposed by [11, 12] are based on rough set theory. Attribute reduction is first carried out [14], and then the branch property is selected by calculating the relative positive region. Yin et al. [13] proposed a reduced decision tree algorithm that can deal with changes in the data based on an incremental algorithm. Three basic theorems were proposed and proved, and the reliability of the algorithm was guaranteed. This algorithm can solve the problem of constructing decision trees on dynamic datasets, and the proposed Dynamic Decision Tree algorithm also promotes the development and improvement of online rule extraction. A two-stage decision tree construction algorithm that combines IBM's SLIQ algorithm with multi-attribute classification based on ECGA has been proposed [14]. The tree construction algorithm first constructs subtrees based on data from a coarse classification, then multi-attribute classification based on ECGA is applied to incomplete classification data on the leaf nodes. The rule set is directly obtained from the incomplete classification data of the leaf nodes, producing a decision tree that is a mixture of multiple attributes. Yang et al. [15] proposed a method to construct a decision tree by transforming a number of classes into two categories. Classes are transformed into positive and negative classes to produce the first level, and the positive class is then subdivided into positive and negative classes to generate the second level. In the same

way, the negative class of the first level is also subdivided into positive and negative classes to generate a second level. This process is repeated until the classification is complete. Wang et al. [16] presented a fast scalable parallel classification (FSPC) algorithm to deal with datasets with multiple attributes. FSPC not only reduces the cost of communication and I/O but also improves the parallelism of the algorithm by dividing the dataset vertically in the process of selecting the test attribute. There have been many studies on decision tree learning algorithms, such as parallel decision tree classification research based on SPRINT [17], decision tree classification research, and the latest development of decision tree algorithms in data mining [18, 19].

2. Other aspects

Wen et al. [20] reported a method to deal with missing attribute values in decision trees. A method is defined to fill in missing values that cannot be added to the decision tree during construction. A new method of information acquisition for semi-structured decision trees has been proposed from the perspective of applications, including a standardization, character, utility, and construction process [21]. This provides a new efficient retrieval tool for web information. The method of network intrusion detection based on multiple decision trees [22] divides a big dataset into several sub-datasets, to which the decision tree algorithm is applied for information retrieval. The results from multiple decision trees are then combined to form a global judgment by voting. The method can be applied to network intrusion detection, which not only improves the processing ability of mass data but also reduces the false-positive rate. The application of decision tree research includes the use of decision trees for data packet filtering [22], customer classification [16], and so on.

Although scholars have continued to research and improve decision tree algorithms (such as SLIQ, SPRINT, and PUBLIC), the shortcomings and limitations of each algorithm are very obvious. The ID3 algorithm is simple and easy to use, but it cannot deal with continuous attributes and the constructed decision tree overfits data; in this regard, the C4.5 algorithm offers two improvements. The CART algorithm not only deals with highly skewed or polymorphic numerical data but also handles sequential or unordered categorical data. However, it is not suitable for processing large-scale data because the training samples must reside in memory.

Decision tree construction algorithms are generated through an in-depth study of decision tree learning. Although the decision trees built by these methods are very useful in generating rules and knowledge, they are often compromised by uncertainties related to people's thinking and perception in terms of rule generation and knowledge expression. Thus, inductive learning with an exact description of features cannot adapt to the imprecise knowledge representation in a system. To handle imprecise knowledge representation in an uncertain (fuzzy) environment, a fuzzy decision tree learning algorithm is required.

The world and the problem domains are always changing. How to express the dynamic nature of decision trees is therefore a genuine problem. Current approaches

can only solve problems in which the decision tree node set and attribute set are static. In fact, it is hoped that the set of nodes in the decision tree can describe dynamic fuzzy uncertainty problems in the real world. Hence, we introduce a Dynamic Fuzzy Lattice to conduct exploratory research on this problem. This will be introduced in the following. The concept of the Dynamic Fuzzy Lattice is described in Appendix 8.4.

3.2 Decision tree methods for a dynamic fuzzy lattice

In this section, we first introduce the concept and classical algorithms related to decision trees and then outline the theoretical framework of decision trees based on the Dynamic Fuzzy Lattice. The following introduces the ID3 algorithm and related examples.

3.2.1 ID3 algorithm and examples

ID3 is a decision tree classification algorithm based on information entropy that uses instance categories based on an attribute set. ID3 is based on information theory, in terms of the mutual information (information gain), as a measure of attribute discrimination. The information gain rate is used as the criterion for attribute selection [24].

For a training instance set X , the number of training instances that belong to class i is N_i and the total number of instances in X is $|X|$. If the probability of an instance belonging to class i is p_i , then

$$p_i = \frac{N_i}{|X|}. \quad (3.1)$$

If the decision attribute has k different values, then the entropy of the training instance set X with respect to state k is

$$\text{Entropy}(X) = \sum_{i=1}^k -p_i \text{lb} p_i. \quad (3.2)$$

The information gain of attribute A related to instance set X is defined as

$$\text{Gain}(X, A) \equiv \text{Entropy}(X) - \sum_{v \in \text{Value}(A)} \frac{|X_v|}{|X|} \text{Entropy}(X_v), \quad (3.3)$$

where $\text{Value}(A)$ is the set of all possible values of attribute A , and X_v indicates the subset that the value of attribute A in X is v , that is

$$X_v = \{x \in X | A(x) = v\}.$$

Tab. 3.1: The training instance set of the students' scores.

	Effort	Family	Homework	Score
1	Hard	Good	Bad	P
2	Hard	Good	Good	P
3	Hard	Average	Bad	N
4	Normal	Poor	Bad	N
5	Hard	Average	Good	P
6	Normal	Good	Good	P
7	Hard	Poor	Bad	P
8	Hard	Poor	Good	N
9	Hard	Good	Bad	P
10	Normal	Average	Bad	N
11	Normal	Average	Good	N
12	Normal	Poor	Good	N

Consider student achievements as an example to illustrate the actual process of the ID3 algorithm. The training instance set of the students' scores is given in Tab. 3.1.

At the beginning of the algorithm, all instances are included in the root node. To find the best partition attribute, the entropy of the training example set is needed:

$$\text{Entropy}(X) = -\frac{6}{12} \text{lb} \frac{6}{12} - \frac{6}{12} \text{lb} \frac{6}{12} = 1.$$

The entropy of each attribute in the training instance is calculated as follows:

$$\text{Entropy}(X_{\text{effort}}, \text{hard}) = -\frac{4}{6} \text{lb} \frac{4}{6} - \frac{4}{6} \text{lb} \frac{4}{6} = 0.9183$$

and

$$\text{Entropy}(X_{\text{effort}}, \text{normal}) = -\frac{2}{6} \text{lb} \frac{2}{6} - \frac{2}{6} \text{lb} \frac{2}{6} = 0.9183.$$

Thus, the information gain for the “Effort” attribute is

$$\begin{aligned} \text{Gain}(X, \text{Effort}) &= \text{Entropy}(X) - \sum_{v \in \{\text{hard}, \text{normal}\}} \frac{|X_v|}{|X|} \text{Entropy}(X_v) \\ &= 1 - \left(\frac{1}{2} \times 0.9183 + \frac{1}{2} \times 0.9183 \right) = 0.0817. \end{aligned}$$

In the same way, $\text{Gain}(X, \text{Family}) = 0.4591$, $\text{Gain}(X, \text{Homework}) = 0$.

From this, we have that the information gain of the “Family” attribute is the largest, and the classification attribute of this node is “Family”. The final Decision Tree is shown in Fig. 3.1.

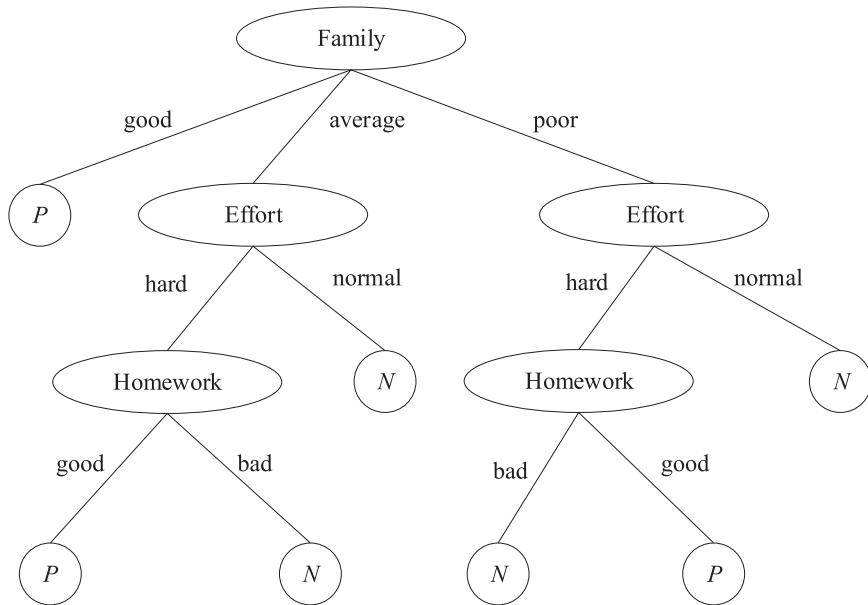


Fig. 3.1: Decision tree generated by Tab. 3.1.

3.2.2 Characteristics of dynamic fuzzy analysis of decision trees

The ID3 algorithm actually ends with the construction of the decision tree. It operates on some of the attributes of the instance set. The instance set has four attributes (see Tab. 3.1): “Effort”, “Family”, “Homework”, and “Score”. For example, the range of “Effort” is {hard, normal} and there is no clear boundary between the two values. That is, there is no exact standard to measure what kind of degree is “hard” and what kind of degree is “normal”. Thus, the condition for obtaining the attribute value of “Effort” is fuzzy. At the same time, the “Effort” is constantly changing and developing, which shows that it has the characteristics of DFD. As the decision tree learning completes the classification of instances through the attributes, the decision tree learning system uncovers dynamic and fuzzy characteristics.

3.2.3 Representation methods for dynamic fuzzy problems in decision trees [25, 26]

Dynamic fuzzy knowledge representation for decision tree learning can be summarized as follows:

- (1) For attribute x_i , an attribute value represented by (\bar{x}_i, \hat{x}_i) is used to replace x_i . This indicates dynamic fuzzy characteristics, and its value is a dynamic fuzzy

number. The attribute “Effort” in Tab. 3.1 is replaced by “Effort = hard” over the range $\{\overleftarrow{0.7}, \overrightarrow{0.3}\}$. This changes the attribute value represented by the dynamic fuzzy number, where $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$ indicates a higher degree or a general degree of effort.

- (2) For instance $\langle x, c(x) \rangle$, where $x = \langle x_1, x_2, \dots, x_i \rangle$ represents the training instance, x_i represents the attributes of the training instance. From (1), we can see that the attributes have the characteristics of DFD. Thus, x is represented by $(\overleftarrow{x}, \overrightarrow{x})$ and $c(x)$ is represented by the dynamic fuzzy number $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$, that is, $c(\overleftarrow{x}, \overrightarrow{x}) = (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \in [\overleftarrow{0}, \overrightarrow{0}] \times [\overleftarrow{1}, \overrightarrow{1}]$.
- (3) Each hypothesis h in the hypothesis space H is represented by the dynamic fuzzy number $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$. The hypothesis $h \in H$ represents a function defined on $X : h : X \rightarrow [0, 1] \times [\leftarrow, \rightarrow]$, where $h(\overleftarrow{x}, \overrightarrow{x}) = (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \in [\overleftarrow{0}, \overrightarrow{0}] \times [\overleftarrow{1}, \overrightarrow{1}]$.
- (4) The instance set U contains the instances, and the characteristics of DFD are represented by its attributes. Thus, the instance set U is represented by dynamic fuzzy sets (DFSs) $(\overleftarrow{U}, \overrightarrow{U})$, which means the instance set is a domain in a dynamic fuzzy space.

The representation of instance 9 in Tab. 3.1 after the above substitutions is presented in Tab. 3.2.

As DFS and DFL are used to represent instances, the decision tree constructed by those instances undergoes the corresponding transformation. For example, if we consider Fig. 3.1 in dynamic fuzzy terms, we obtain Fig. 3.2. The decision tree constructed by instances based on DFS and DFL is called a Dynamic Fuzzy Decision Tree.

To construct a decision tree using the Dynamic Fuzzy Lattice, we need to define the general-to-special order structure. Firstly, we give the following two hypotheses.

$$h_i = \langle (\alpha_1, \alpha_1), (\alpha_2, \alpha_2), \dots, (\alpha_m, \alpha_m) \rangle$$

$$h_j = \langle (\overleftarrow{\beta}_1, \overrightarrow{\beta}_1), (\overleftarrow{\beta}_2, \overrightarrow{\beta}_2), \dots, (\overleftarrow{\beta}_m, \overrightarrow{\beta}_m) \rangle$$

Definition 3.1 Let h_i, h_j be functions defined on $(\overleftarrow{X}, \overrightarrow{X})$. h_i is equal to or more general than h_j if and only if, $(\forall (\overleftarrow{x}, \overrightarrow{x})) \in (\overleftarrow{X}, \overrightarrow{X})$,

$$[(h_i(\overleftarrow{x}, \overrightarrow{x}) = (\overleftarrow{\alpha}, \overrightarrow{\alpha})) \rightarrow (h_j(\overleftarrow{x}, \overrightarrow{x}) = (\overleftarrow{\beta}, \overrightarrow{\beta}))], (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \leq (\overleftarrow{\beta}, \overrightarrow{\beta}).$$

This is denoted by $h_i \geq_g h_j$, where $(\overleftarrow{x}, \overrightarrow{x})$ represents the instance. If h_i is strictly more general than h_j , we write $h_i >_g h_j$.

Tab. 3.2: Representation of instance 9 in Tab. 3.1.

Effort = hard	Family = good	Homework = bad	Score = P
9	0.7	0.6	0.4

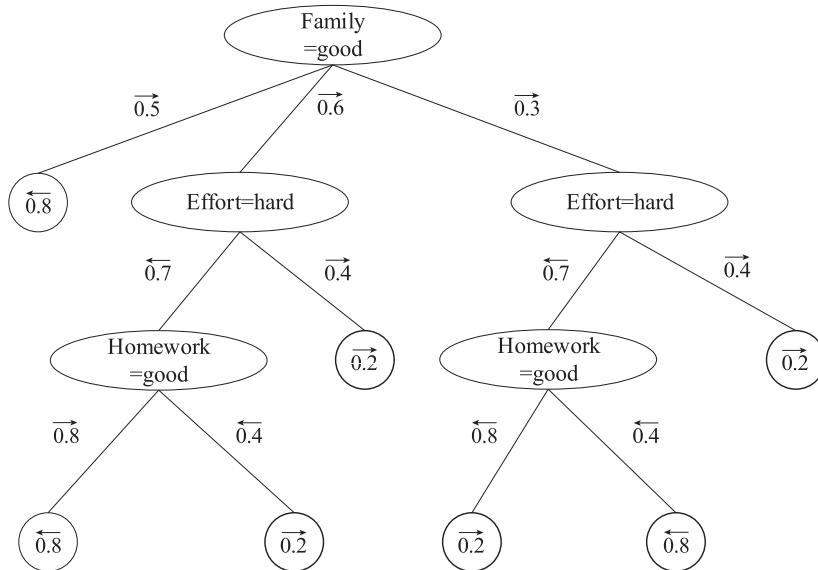


Fig. 3.2: Decision tree after Fig. 3.1 dynamic fuzzy.

The \geq_g operator defines a partial order relation on the instance space $(\overleftarrow{U}, \overrightarrow{U})$ (the relation is reflexive, antisymmetric, and transitive), which provides an effective structure for learning problems. A more general explanation for the \geq_g relation is that h_i contains fewer instance constraints, as any instance of a class that is divided by it will be divided into the same class by h_j . Because h_j has more constraints than h_i , h_i is more general than h_j . The attributes “Family = good”, “Effort = hard” and “Homework = good” in Fig. 3.3 divide the instances of the training set. As shown in Fig. 3.3, the left tree structure is a set of instances of the decision tree that correspond to each node, and the relationship between the upper set and the lower set can be

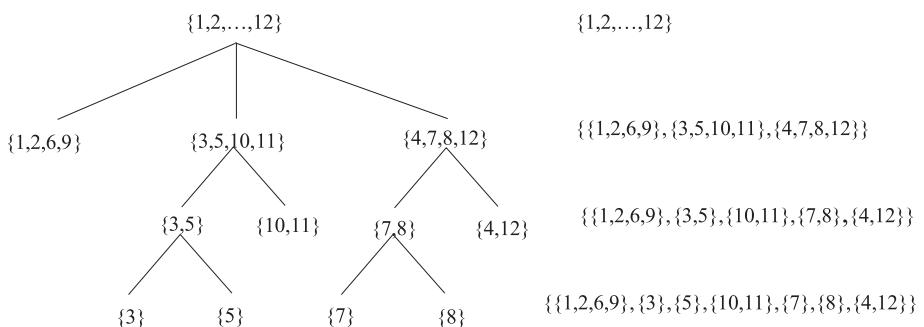


Fig. 3.3: Partition of training samples corresponding to the DFDT in Fig. 3.2.

defined by \geq_g , such as $\{3, 5, 10, 11\} \geq_g \{3, 5\}, \{3, 5, 10, 11\} \geq_g \{10, 11\}$. Therefore, the set of corresponding instances of the nodes of the decision tree consists of the partial order relation. The right side is the result of dividing the instance set at each layer of the decision tree. It can be seen from the partition result that the classification attributes have gradually divided the instance set.

3.2.4 DFDT classification attribute selection algorithm

The representation of the DFDT is given above. As in the construction of a decision tree, we need to determine how to divide the set of instances, that is, decide what attributes to involve in the next step of the classification. This gives the DFDT classification attribute selection algorithm.

In DFDT learning, $(\overleftarrow{a}_i, \overrightarrow{a}_i)$ denotes an attribute, the range of $(\overleftarrow{a}_i, \overrightarrow{a}_i)$ is $(\overleftarrow{V}_{\overleftarrow{a}_i}, \overrightarrow{V}_{\overrightarrow{a}_i}) = \{(\overleftarrow{v}_1, \overrightarrow{v}_1), (\overleftarrow{v}_2, \overrightarrow{v}_2), \dots, (\overleftarrow{v}_k, \overrightarrow{v}_k)\}$, and $|\{(\overleftarrow{V}_{\overleftarrow{a}_i}, \overrightarrow{V}_{\overrightarrow{a}_i})\}| = k$ denotes the number of attribute values of $(\overleftarrow{a}_i, \overrightarrow{a}_i)$. An instance can be denoted as

$$(\overleftarrow{I}_i, \overrightarrow{I}_i) = \left\langle (\overleftarrow{V}_{\overleftarrow{a}_1}^1, \overrightarrow{V}_{\overrightarrow{a}_1}^1), (\overleftarrow{V}_{\overleftarrow{a}_2}^2, \overrightarrow{V}_{\overrightarrow{a}_2}^2), \dots, (\overleftarrow{V}_{\overleftarrow{a}_m}^m, \overrightarrow{V}_{\overrightarrow{a}_m}^m) \right\rangle,$$

where $(\overleftarrow{v}_{\overleftarrow{a}_i}^i, \overrightarrow{v}_{\overrightarrow{a}_i}^i)$ denotes the i th attribute value of $(\overleftarrow{a}_i, \overrightarrow{a}_i)$.

The instance set can be denoted as $(\overleftarrow{U}, \overrightarrow{U}) = \{(\overleftarrow{I}_i, \overrightarrow{I}_i) | i = 1, 2, \dots, n\}$.

From the above, the instances in $(\overleftarrow{U}, \overrightarrow{U})$ whose attributes $(\overleftarrow{a}_i, \overrightarrow{a}_i)$ have values $(\overleftarrow{t}, \overrightarrow{t})$ can be denoted as

$$\begin{aligned} \text{Par}_{(\overleftarrow{a}_i, \overrightarrow{a}_i)}^{(\overleftarrow{t}, \overrightarrow{t})}(\overleftarrow{U}, \overrightarrow{U}) &= \{(\overleftarrow{S}, \overrightarrow{S}) | (\overleftarrow{S}, \overrightarrow{S}) \subseteq (\overleftarrow{U}, \overrightarrow{U}), \\ &(\overleftarrow{I}_j, \overrightarrow{I}_j) \in (\overleftarrow{S}, \overrightarrow{S}), (\overleftarrow{V}_{\overleftarrow{a}_i}, \overrightarrow{V}_{\overrightarrow{a}_i})(\overleftarrow{I}_j, \overrightarrow{I}_j) = (\overleftarrow{t}, \overrightarrow{t})\}, \end{aligned}$$

where $(\overleftarrow{t}, \overrightarrow{t}) = (\overleftarrow{v}_{\overleftarrow{a}_i}^k, \overrightarrow{v}_{\overrightarrow{a}_i}^k)$, that is, the k th ($k \leq |\{(\overleftarrow{V}_{\overleftarrow{a}_i}, \overrightarrow{V}_{\overrightarrow{a}_i})\}|$) attribute value of attribute $(\overleftarrow{a}_i, \overrightarrow{a}_i)$.

The instance set U can be divided according to the value of attribute $(\overleftarrow{a}_i, \overrightarrow{a}_i)$, that is

$$\begin{aligned} \text{Div}_{(\overleftarrow{a}_i, \overrightarrow{a}_i)}(\overleftarrow{U}, \overrightarrow{U}) &= \{(\overleftarrow{S}_1, \overrightarrow{S}_1), (\overleftarrow{S}_2, \overrightarrow{S}_2), \dots, \\ &(\overleftarrow{S}_n, \overrightarrow{S}_n) | (\overleftarrow{S}_i, \overrightarrow{S}_i) = \text{Par}_{(\overleftarrow{a}_i, \overrightarrow{a}_i)}^{(\overleftarrow{t}, \overrightarrow{t})}(\overleftarrow{U}, \overrightarrow{U})\}, \end{aligned}$$

where $n \leq |\{(\overleftarrow{V}_{\overleftarrow{a}_i}, \overrightarrow{V}_{\overrightarrow{a}_i})\}|$.

The condition attribute is $(\overleftarrow{A}, \overrightarrow{A}) = \{(\overleftarrow{A}_1, \overrightarrow{A}_1), (\overleftarrow{A}_2, \overrightarrow{A}_2), \dots, (\overleftarrow{A}_m, \overrightarrow{A}_m)\}$ and the decision attribute is $(\overleftarrow{A}_D, \overrightarrow{A}_D)$. As the decision attribute $(\overleftarrow{A}_D, \overrightarrow{A}_D)$ is dependent on

the condition attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$, there is a certain relationship between the condition attribute and decision attribute regarding the division of the instance set. It meets one of the following three relationships:

$\text{Par}_{(\overleftarrow{A}_D, \overrightarrow{A}_D)}^{(\overleftarrow{v}_j, \overrightarrow{v}_j)}(\overleftarrow{U}, \overrightarrow{U})$ is the partition of $(\overleftarrow{U}, \overrightarrow{U})$ by the j th attribute value of decision attribute $(\overleftarrow{A}_D, \overrightarrow{A}_D)$, $\exists(\overleftarrow{A}_i, \overrightarrow{A}_i)$, then $\text{Par}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U}) \subseteq \text{Par}_{(\overleftarrow{A}_D, \overrightarrow{A}_D)}^{(\overleftarrow{v}_j, \overrightarrow{v}_j)}(\overleftarrow{U}, \overrightarrow{U})$, $\text{Par}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U}) = \text{Par}_{(\overleftarrow{A}_D, \overrightarrow{A}_D)}^{(\overleftarrow{v}_j, \overrightarrow{v}_j)}(\overleftarrow{U}, \overrightarrow{U})$, or $\text{Par}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U}) \supseteq \text{Par}_{(\overleftarrow{A}_D, \overrightarrow{A}_D)}^{(\overleftarrow{v}_j, \overrightarrow{v}_j)}(\overleftarrow{U}, \overrightarrow{U})$. Because “ \supseteq ” and “ \subseteq ” are inverse operations, and “ $=$ ” is a special case of “ \supseteq ”, we take the first situation as follows:

$\text{Par}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U}) = \text{Par}_{(\overleftarrow{A}_D, \overrightarrow{A}_D)}^{(\overleftarrow{v}_j, \overrightarrow{v}_j)}(\overleftarrow{U}, \overrightarrow{U})$ represents the set of instances when the value of decision attribute $(\overleftarrow{A}_D, \overrightarrow{A}_D)$ is $(\overleftarrow{v}_j, \overrightarrow{v}_j)$ and contains those values for which condition attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ is $(\overleftarrow{v}_k, \overrightarrow{v}_k)$. That is, when the value of condition attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ is $(\overleftarrow{v}_k, \overrightarrow{v}_k)$, the partition instance set can be determined as class $(\overleftarrow{v}_j, \overrightarrow{v}_j)$. As $\text{Par}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U}) \subseteq \text{Par}_{(\overleftarrow{A}_D, \overrightarrow{A}_D)}^{(\overleftarrow{v}_j, \overrightarrow{v}_j)}(\overleftarrow{U}, \overrightarrow{U})$, we know the other attribute values besides $(\overleftarrow{v}_k, \overrightarrow{v}_k)$ of $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ can classify the instances into class $(\overleftarrow{v}_j, \overrightarrow{v}_j)$.

Simultaneously, $\exists n \leq |(\overleftarrow{V}_{\overleftarrow{A}_i}, \overrightarrow{V}_{\overrightarrow{A}_i})|$, $\text{Par}_{(\overleftarrow{A}_D, \overrightarrow{A}_D)}^{(\overleftarrow{v}_j, \overrightarrow{v}_j)}(\overleftarrow{U}, \overrightarrow{U}) \subseteq \bigcup_{k=1}^n \text{Par}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U})$, so the certain degree of attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ classifying the instances into class $(\overleftarrow{v}_j, \overrightarrow{v}_j)$ is

$$\text{Cer}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^j(\overleftarrow{U}, \overrightarrow{U}) = \max\left(\left|\text{Par}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(U)\right|\right) / \min\left(\left|\bigcup_{k=1}^n \text{Par}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U})\right|\right),$$

where

$$\text{Par}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U}) \subseteq \text{Par}_{(\overleftarrow{A}_D, \overrightarrow{A}_D)}^{(\overleftarrow{v}_j, \overrightarrow{v}_j)}(\overleftarrow{U}, \overrightarrow{U}) \subseteq \bigcup_{k=1}^n \text{Par}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U}), n \leq |(\overleftarrow{V}_{\overleftarrow{A}_i}, \overrightarrow{V}_{\overrightarrow{A}_i})|.$$

There are $n = |(\overleftarrow{V}_{\overleftarrow{A}_i}, \overrightarrow{V}_{\overrightarrow{A}_i})|$ classes in the instance set. The condition attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ has a certain degree of partitioning for each class. Thus, the degree of certainty of partitioning for the entire set of instances is denoted as

$$\text{Cer}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U}) = \sum_{k=1}^n \text{pro}_k \text{Cer}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^k(\overleftarrow{U}, \overrightarrow{U}),$$

where $\text{pro}_k = \left|\text{Par}_{(\overleftarrow{A}_D, \overrightarrow{A}_D)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U})\right| / |(\overleftarrow{U}, \overrightarrow{U})|$ denotes the ratio of the number of instances belonging to class $(\overleftarrow{v}_k, \overrightarrow{v}_k)$ in instance set $(\overleftarrow{U}, \overrightarrow{U})$ to the total set of instances.

We set $\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U}) = 1 - \text{Cer}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U}) = 1 - \sum_{k=1}^n \text{pro}_k \text{Cer}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^k(\overleftarrow{U}, \overrightarrow{U})$ as the attribute selection criteria in the decision tree classification. The range of $\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U})$ is $[0, 1]$.

If $\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U}) = 0$, the conditional attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ has a degree of certainty of 1 for the entire set of instances, indicating that the conditional attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ can definitely divide the instance set. If $\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U}) = 1$, we have a maximum indefinite partition of the instance set in conditional attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$. Therefore, the smaller the value of $\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U})$, the better the attributes divide the instance set. In the attribute selection algorithm, the condition attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$, which enables $\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U})$ to obtain the minimum value, is selected as the branch attribute.

1. Basic idea and description of the algorithm

In the process of constructing the decision tree, branch attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ should be able to cover as many instances as possible, reducing the number of instances to be classified. This effectively reduces the whole branch of the tree and achieves the purpose of simplification. The attributes of $\min(\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U}))$ satisfy the above conditions, and the contribution of each partition to the entire set of decision instances is considered more comprehensively by using pro_k . We select the best attribute from the available set of attributes to create the branch, and the attributes that have been used to branch can no longer be selected.

The basic description of the algorithm is as follows:

- (1) For each condition attribute, calculate $\text{Div}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U})$, $\text{Div}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U}) = \left\{ \bigcup_{k=1}^{\left| \overleftarrow{V}_{\overleftarrow{A}_i}, \overrightarrow{V}_{\overrightarrow{A}_i} \right|} \text{Par}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U}) \right\}$;
- (2) Compute the degree of certainty $\text{Cer}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^j(\overleftarrow{U}, \overrightarrow{U})$ of condition attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ classifying the instance into class $(\overleftarrow{v}_j, \overrightarrow{v}_j)$, and the ratio pro_j of the number of instances with decision attribute $(\overleftarrow{v}_j, \overrightarrow{v}_j)$ to the total number of instances;
- (3) Compute $\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U})$, select the attribute that satisfies $\min(\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U}))$ as the branch attribute, then divide the instance set as $\{\text{Par}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_1, \overrightarrow{v}_1)}(\overleftarrow{U}, \overrightarrow{U}), \text{Par}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_2, \overrightarrow{v}_2)}(\overleftarrow{U}, \overrightarrow{U}), \dots, \text{Par}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U})\}$;
- (4) Determine whether the instances contained in $\text{Par}_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{t}, \overrightarrow{t})}(\overleftarrow{U}, \overrightarrow{U})$ belong to the same category. If so, the corresponding node is a leaf node; otherwise, return to (1) and repeat the first three steps. The attributes that have been used to branch are no longer selected.

The pseudocode is described in Algorithm 3.1.

Algorithm 3.1 Procedure buildTree(TreeNode, U)

```

//Initialization TreeNode←{}, BA
for each condition attribute  $(\overleftarrow{A}_i, \overrightarrow{A}_i)$  do
  for  $j \leftarrow 1$  to  $|(\overleftarrow{V}_{\overleftarrow{A}_i}, \overrightarrow{V}_{\overrightarrow{A}_i})|$  do
    Par $_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{V}_j, \overrightarrow{V}_j)}(\overleftarrow{U}, \overrightarrow{U})$ 
  repeat
  Get Div $_{(\overleftarrow{a}_i, \overrightarrow{a}_i)}(\overleftarrow{U}, \overrightarrow{U})$ 
  repeat
  Apply Cer $_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^j(\overleftarrow{U}, \overrightarrow{U})$  to calculate the certain degree of U that is divided by  $A_i$ 
  Get the classifying attribute BA by calculating min( $\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U})$ )
  Use BA to split training set U into subset Split( $\overleftarrow{U}, \overrightarrow{U}$ ) =  $\{(\overleftarrow{U}_1, \overrightarrow{U}_1), (\overleftarrow{U}_2, \overrightarrow{U}_2), \dots, (\overleftarrow{U}_n, \overrightarrow{U}_n)\}$ 
  for each subset  $(\overleftarrow{U}_i, \overrightarrow{U}_i)$  in Split( $\overleftarrow{U}, \overrightarrow{U}$ )
    Create  $i$ th child of TreeNode
    if all records in  $(\overleftarrow{U}_i, \overrightarrow{U}_i)$  are in the same class
      then child is leaf, save  $(\overleftarrow{U}_i, \overrightarrow{U}_i)$  as LeafNode
      else save  $A_i$  as no-LeafNode, buildTree(Child i,  $(\overleftarrow{U}_i, \overrightarrow{U}_i)$ )
  repeat

```

2. Algorithm application examples

We use the data in [29] (see Tab. 3.3) to illustrate the flow of the proposed classification attribute selection algorithm. First, the data of the source data table are expressed by dynamic fuzzy mathematics theory. The DFDT is then generated by classifying the training set according to the steps of the algorithm.

The data in Tab. 3.4 are classified according to the algorithm described in Section 3.2.4.1. For convenience, $(\overleftarrow{a}, \overrightarrow{a})$ replaces Outlook = Sunny, $(\overleftarrow{b}, \overrightarrow{b})$ replaces Temperature = Hot, $(\overleftarrow{c}, \overrightarrow{c})$ replaces Humidity = High, $(\overleftarrow{d}, \overrightarrow{d})$ replaces Wind = Strong, $(\overleftarrow{e}, \overrightarrow{e})$ replaces PlayTennis = Yes, and $(\overleftarrow{U}, \overrightarrow{U})$ denotes the instance set in Tab. 3.4. We substitute instances with the numbers in the instance set to simplify the representation (same below). That is, $(\overleftarrow{U}, \overrightarrow{U}) = \{D1, D2, \dots, D14\}$. The process is as follows.

First, the attribute set $(\overleftarrow{U}, \overrightarrow{U})$ is divided:

$$\text{Div}_{(\overleftarrow{a}, \overrightarrow{a})}(\overleftarrow{U}, \overrightarrow{U}) = \{1, 2, 8, 9, 11\}, \{3, 7, 12, 13\}, \{4, 5, 6, 10, 14\}$$

$$\text{Div}_{(\overleftarrow{b}, \overrightarrow{b})}(\overleftarrow{U}, \overrightarrow{U}) = \{1, 2, 3, 13\}, \{4, 8, 10, 11, 12, 14\}, \{5, 6, 7, 9\}$$

$$\text{Div}_{(\overleftarrow{c}, \overrightarrow{c})}(\overleftarrow{U}, \overrightarrow{U}) = \{1, 2, 3, 4, 8, 12, 14\}, \{5, 6, 7, 9, 10, 11, 13\}$$

$$\text{Div}_{(\overleftarrow{d}, \overrightarrow{d})}(\overleftarrow{U}, \overrightarrow{U}) = \{2, 6, 7, 11, 12, 14\}, \{1, 3, 4, 5, 8, 9, 10, 13\}$$

$$\text{Div}_{(\overleftarrow{e}, \overrightarrow{e})}(\overleftarrow{U}, \overrightarrow{U}) = \{3, 4, 5, 7, 9, 10, 11, 12, 13\}, \{1, 2, 6, 8, 14\}$$

Tab. 3.3: Training instance set of PlayTennis.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

For the condition attribute Outlook, use $\overrightarrow{0.7}$, $\overleftarrow{0.4}$, $\overleftarrow{0.2}$ to indicate sunny, overcast, and rain.

For the condition attribute Temperature, use $\overleftarrow{0.8}$, $\overleftarrow{0.5}$, $\overrightarrow{0.4}$ to indicate hot, mild, and cool.

For the condition attribute Humidity, use $\overleftarrow{0.6}$, $\overrightarrow{0.4}$ to indicate high and normal.

For the condition attribute Windy, use $\overrightarrow{0.8}$, $\overleftarrow{0.1}$ to indicate strong and weak.

For the decision attribute PlayTennis, use $\overrightarrow{0.7}$, $\overleftarrow{0.3}$ to indicate yes and no.

Tab. 3.4: The training instance set PlayTennis after dynamic fuzzification.

Day	Outlook = sunny	Temperature = hot	Humidity = high	Wind = strong	PlayTennis = yes
D1	$\overrightarrow{0.7}$	$\overleftarrow{0.8}$	$\overleftarrow{0.6}$	$\overleftarrow{0.1}$	$\overleftarrow{0.3}$
D2	$\overrightarrow{0.7}$	$\overleftarrow{0.8}$	$\overleftarrow{0.6}$	$\overrightarrow{0.8}$	$\overleftarrow{0.3}$
D3	$\overrightarrow{0.4}$	$\overleftarrow{0.8}$	$\overleftarrow{0.6}$	$\overrightarrow{0.1}$	$\overrightarrow{0.7}$
D4	$\overleftarrow{0.2}$	$\overleftarrow{0.5}$	$\overleftarrow{0.6}$	$\overrightarrow{0.1}$	$\overrightarrow{0.7}$
D5	$\overleftarrow{0.2}$	$\overrightarrow{0.4}$	$\overrightarrow{0.4}$	$\overrightarrow{0.1}$	$\overrightarrow{0.7}$
D6	$\overleftarrow{0.2}$	$\overrightarrow{0.4}$	$\overrightarrow{0.4}$	$\overrightarrow{0.8}$	$\overrightarrow{0.3}$
D7	$\overrightarrow{0.4}$	$\overrightarrow{0.4}$	$\overrightarrow{0.4}$	$\overrightarrow{0.8}$	$\overrightarrow{0.7}$
D8	$\overrightarrow{0.7}$	$\overleftarrow{0.5}$	$\overleftarrow{0.6}$	$\overrightarrow{0.1}$	$\overleftarrow{0.3}$
D9	$\overrightarrow{0.7}$	$\overrightarrow{0.4}$	$\overrightarrow{0.4}$	$\overleftarrow{0.1}$	$\overrightarrow{0.7}$
D10	$\overleftarrow{0.2}$	$\overleftarrow{0.5}$	$\overrightarrow{0.4}$	$\overleftarrow{0.1}$	$\overrightarrow{0.7}$
D11	$\overrightarrow{0.7}$	$\overleftarrow{0.5}$	$\overrightarrow{0.4}$	$\overrightarrow{0.8}$	$\overrightarrow{0.7}$
D12	$\overrightarrow{0.4}$	$\overleftarrow{0.5}$	$\overrightarrow{0.6}$	$\overrightarrow{0.8}$	$\overrightarrow{0.7}$
D13	$\overrightarrow{0.4}$	$\overleftarrow{0.8}$	$\overrightarrow{0.4}$	$\overleftarrow{0.1}$	$\overrightarrow{0.7}$
D14	$\overleftarrow{0.2}$	$\overleftarrow{0.5}$	$\overleftarrow{0.6}$	$\overrightarrow{0.8}$	$\overleftarrow{0.3}$

The degree of certainty of the classification of each attribute on $(\overleftarrow{U}, \overrightarrow{U})$ is then calculated:

$$\begin{aligned}\Gamma_{(\overleftarrow{a}, \overrightarrow{a})}(\overleftarrow{U}, \overrightarrow{U}) &= 1 - \sum_{k=1}^2 \text{pro}_k \text{Cer}_{(\overleftarrow{a}, \overrightarrow{a})}^j(\overleftarrow{U}, \overrightarrow{U}) \\ &= 1 - \left(\frac{4}{14} \times \frac{9}{14} + 0 \right) = 0.8163.\end{aligned}$$

Similarly, $\Gamma_{(\overleftarrow{b}, \overrightarrow{b})}(\overleftarrow{U}, \overrightarrow{U}) = 1$, $\Gamma_{(\overleftarrow{c}, \overrightarrow{c})}(\overleftarrow{U}, \overrightarrow{U}) = 1$, $\Gamma_{(\overleftarrow{e}, \overrightarrow{e})}(\overleftarrow{U}, \overrightarrow{U}) = 1$.

We select attribute $(\overleftarrow{a}, \overrightarrow{a})$ as the first branch point, that is, the root node.

As the instances in $\text{Par}_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.4}}(\overleftarrow{U}, \overrightarrow{U}) = \{3, 7, 12, 13\}$ belong to the same class, the branch is ended. Look at the other classes.

When $(\overleftarrow{a}, \overrightarrow{a}) = \overrightarrow{0.7}$,

$$\text{Div}_{(\overleftarrow{b}, \overrightarrow{b})}(\text{Par}_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U})) = \{\{1, 2\}, \{8, 11\}, \{9\}\}$$

$$\text{Div}_{(\overleftarrow{c}, \overrightarrow{c})}(\text{Par}_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U})) = \{\{1, 2, 8\}, \{9, 11\}\}$$

$$\text{Div}_{(\overleftarrow{d}, \overrightarrow{d})}(\text{Par}_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U})) = \{\{1, 8, 9\}, \{2, 11\}\}$$

$$\text{Div}_{(\overleftarrow{e}, \overrightarrow{e})}(\text{Par}_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U})) = \{\{1, 2, 8\}, \{9, 11\}\}$$

$$\Gamma_{(\overleftarrow{b}, \overrightarrow{b})}(\text{Par}_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U})) = 0.5,$$

$$\Gamma_{(\overleftarrow{c}, \overrightarrow{c})}(\text{Par}_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U})) = 0, \Gamma_{(\overleftarrow{d}, \overrightarrow{d})}(\text{Par}_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U})) = 1.$$

Therefore, when $(\overleftarrow{a}, \overrightarrow{a}) = \overrightarrow{0.7}$, the branch attribute is $(\overleftarrow{c}, \overrightarrow{c})$ and the instances in $\text{Par}_{(\overleftarrow{c}, \overrightarrow{c})}^{\overrightarrow{0.4}}(\text{Par}_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U}))$, $\text{Par}_{(\overleftarrow{c}, \overrightarrow{c})}^{\overrightarrow{0.6}}(\text{Par}_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U}))$ are in the same class. Thus, the branch is ended.

Similarly, when $(\overleftarrow{a}, \overrightarrow{a}) = \overleftarrow{0.2}$, $\Gamma_{(\overleftarrow{b}, \overrightarrow{b})}(\text{Par}_{(\overleftarrow{a}, \overrightarrow{a})}^{\overleftarrow{0.2}}(\overleftarrow{U}, \overrightarrow{U})) = 1$, $\Gamma_{(\overleftarrow{c}, \overrightarrow{c})}(\text{Par}_{(\overleftarrow{a}, \overrightarrow{a})}^{\overleftarrow{0.2}}(\overleftarrow{U}, \overrightarrow{U})) = 1$, $\Gamma_{(\overleftarrow{d}, \overrightarrow{d})}(\text{Par}_{(\overleftarrow{a}, \overrightarrow{a})}^{\overleftarrow{0.2}}(\overleftarrow{U}, \overrightarrow{U})) = 0$.

Hence, when $(\overleftarrow{a}, \overrightarrow{a}) = \overleftarrow{0.2}$, the branch attribute is $(\overleftarrow{d}, \overrightarrow{d})$ and the instances in $\text{Par}_{(\overleftarrow{d}, \overrightarrow{d})}^{\overrightarrow{0.8}}(\text{Par}_{(\overleftarrow{a}, \overrightarrow{a})}^{\overleftarrow{0.2}}(\overleftarrow{U}, \overrightarrow{U}))$, $\text{Par}_{(\overleftarrow{d}, \overrightarrow{d})}^{\overrightarrow{0.1}}(\text{Par}_{(\overleftarrow{a}, \overrightarrow{a})}^{\overleftarrow{0.2}}(\overleftarrow{U}, \overrightarrow{U}))$ are in the same class. Again, the branch is ended.

After constructing the decision tree above, we can see that all leaf nodes corresponding to the instance set belong to the same class. This completes the tree structure. The whole instance set is classified and the constructed decision tree is shown in Fig. 3.4.

3.2.5 Dynamic fuzzy binary decision tree

A test attribute with multiple attribute values may have two or more branches per internal node. A DFDT with only two branches per internal node is called a Dynamic Fuzzy Binary Decision Tree (DFBDT). There are a number of algorithms that can

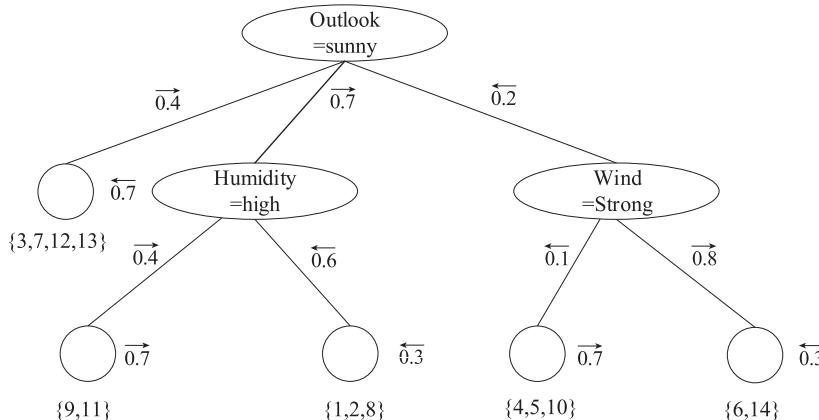


Fig. 3.4: Decision tree generated from Tab. 3.4.

create binary decision trees. For example, the CRAT algorithm uses entropy to select the best splitting attributes and criteria and produces only two child nodes where each subclass has children [30]. Sun and Zhang [31] described a binary decision tree generation algorithm based on attribute-value information gain optimization and a method of transforming multi-valued attributes into binary attributes.

DFBDT is a special form of DFDT and has a slightly different tree structure. By introducing, the relationship between the corresponding subset of the nodes of the decision tree can be analysed. At the same time, each layer of the decision tree corresponding to the instance set is divided according to a certain relationship. DFBDT is illustrated in Fig. 3.5.

In DFDT, there is a partial ordering relationship between the instance set of upper-level nodes and the directly lower nodes. In DFBDT, the instance set contained in Attribute $B = V_B$ and its two sub-nodes is $\{\{1, 4, 5, 6, 7, 8, 9, 10, 11\}, \{1, 4, 6, 8, 9, 10\}, \{5, 7, 11\}\}$. The elements not only meet the partial order relationship, but also the union of $\{1, 4, 6, 8, 9, 10\}$ and $\{5, 7, 11\}$ also equals $\{1, 4, 5, 6, 7, 8, 9, 10, 11\}$. This obeys the definition of a semi-lattice. As with DFDT, the branch attribute also divides the entire set of instances. To study the partition of the object domain, a partition lattice has been proposed [32] and was introduced to a decision tree to study the partitioning of the instance set.

A DFDT can be denoted as

$$(\overleftarrow{T}, \overrightarrow{T}) = \langle (\overleftarrow{U}, \overrightarrow{U}), (\overleftarrow{C}, \overrightarrow{C}), (\overleftarrow{D}, \overrightarrow{D}), (\overleftarrow{V}, \overrightarrow{V}), (\overleftarrow{F}, \overrightarrow{F}) \rangle,$$

where $(\overleftarrow{U}, \overrightarrow{U})$ denotes the instance set, $(\overleftarrow{C}, \overrightarrow{C}) \cup (\overleftarrow{D}, \overrightarrow{D}) = (\overleftarrow{A}, \overrightarrow{A})$ is the attribute set, subsets $(\overleftarrow{C}, \overrightarrow{C})$ and $(\overleftarrow{D}, \overrightarrow{D})$ are the condition attribute and decision attribute, respectively, and $(\overleftarrow{V}, \overrightarrow{V}) = \bigcup (\overleftarrow{V}_r, \overrightarrow{V}_r)$ denotes the range of attribute.

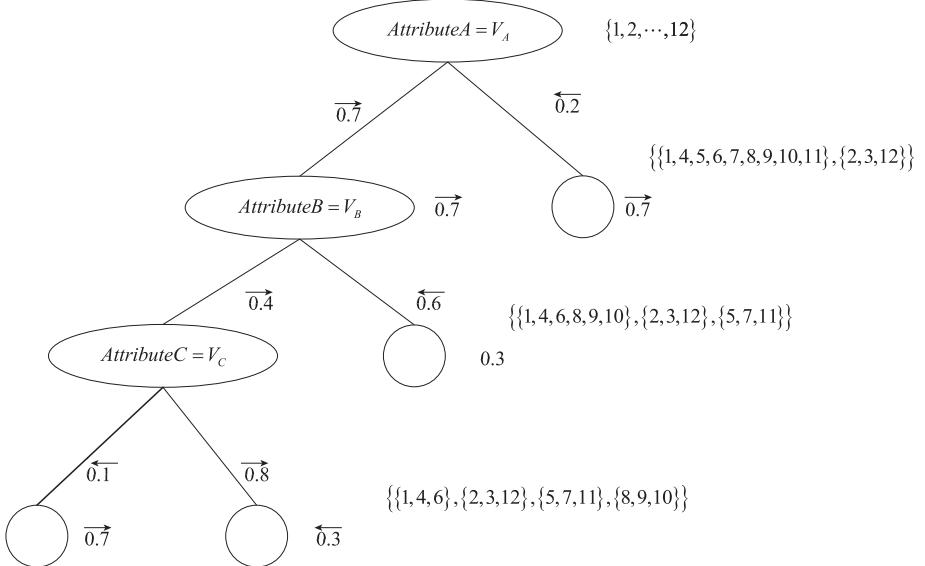


Fig. 3.5: DFBDT example.

$(\overleftarrow{A}, \overrightarrow{A})$. $(\overleftarrow{F}, \overrightarrow{F}) : (\overleftarrow{U}, \overrightarrow{U}) \times (\overleftarrow{A}, \overrightarrow{A}) \rightarrow (\overleftarrow{V}, \overrightarrow{V})$ is a mapping that specifies the attribute values for each object in instance set $(\overleftarrow{U}, \overrightarrow{U})$, and $(\overleftarrow{F}, \overrightarrow{F}) \subseteq [0, 1] \times [\leftarrow, \rightarrow]$.

Definition 3.2 Given a finite universe U , $\pi = \{X_i | 1 \leq i \leq m\}$ is a division of U if π satisfies the following conditions [33]:

- X_i is not empty;
- $X_i \cap X_j = \varnothing$, $i \neq j$;
- $\bigcup\{X_i | 1 \leq i \leq m\} = U$.

We can define a partial order relationship to represent the size between partitions. Suppose π_1, π_2 are two divisions of U . If $\pi_1 \leq \pi_2$, any partition of π_1 is contained in a certain partition of π_2 . That is, $\pi_1 \leq \pi_2$ if and only if, $\forall X_i \in \pi_1, \exists X_j \in \pi_2$ such that $X_i \subseteq X_j$.

The operator “ \leq ” defines a partial order relation on the partition of domain U (the relation \leq is reflexive, anti-symmetric, and transitive). If $\pi_1 \leq \pi_2$, then we say partition π_1 is smaller than π_2 .

Take the partition as a computing object, and define intersection and union operations:

- $\pi_1 \wedge \pi_2$ denotes the largest partition smaller than both π_1 and π_2 ;
- $\pi_1 \vee \pi_2$ denotes the smallest partition larger than both π_1 and π_2 .

The union operation of partitions is equivalent to the supremum in computing, and the intersection operation is equivalent to the infimum operation. At the same

time, the thickness of the partition can be defined so that we have a partition lattice. The partition lattice is introduced into the DFDT because the process of constructing a decision tree involves partitioning an instance set through branch attributes. The definition of the partition of the DFDT instance set is given so that we can get the Dynamic Fuzzy Partition Lattice.

Definition 3.3 Suppose $(\overleftarrow{U}, \overrightarrow{U})$ is the training instance set of DFDT. $(\overleftarrow{\pi}, \overrightarrow{\pi}) = \{\overleftarrow{X}_i, \overrightarrow{X}_i | 1 \leq i \leq m\}$ is a partition of $(\overleftarrow{U}, \overrightarrow{U})$ if $(\overleftarrow{\pi}, \overrightarrow{\pi})$ satisfies the following conditions:

- $(\overleftarrow{X}_i, \overrightarrow{X}_i)$ is not empty;
- $(\overleftarrow{X}_i, \overrightarrow{X}_i) \cap (\overleftarrow{X}_j, \overrightarrow{X}_j) = \emptyset (i \neq j)$;
- $\bigcup\{(\overleftarrow{X}_i, \overrightarrow{X}_i) | 1 \leq i \leq m\} = (\overleftarrow{U}, \overrightarrow{U})$.

Define the partial order relationship between partitions (the intersection and union operations) as follows:

$(\overleftarrow{\pi}_1, \overrightarrow{\pi}_1), (\overleftarrow{\pi}_2, \overrightarrow{\pi}_2)$ are partitions of $(\overleftarrow{U}, \overrightarrow{U}), (\overleftarrow{\pi}_1, \overrightarrow{\pi}_1) \leq (\overleftarrow{\pi}_2, \overrightarrow{\pi}_2)$, if and only if $\forall (\overleftarrow{X}_i, \overrightarrow{X}_i) \in (\overleftarrow{\pi}_1, \overrightarrow{\pi}_1) \exists (\overleftarrow{X}_j, \overrightarrow{X}_j) \in (\overleftarrow{\pi}_2, \overrightarrow{\pi}_2)$ such that $(\overleftarrow{X}_i, \overrightarrow{X}_i) \subseteq (\overleftarrow{X}_j, \overrightarrow{X}_j)$. If $(\overleftarrow{\pi}_1, \overrightarrow{\pi}_1)$ is strictly smaller than $(\overleftarrow{\pi}_2, \overrightarrow{\pi}_2)$, we write $(\overleftarrow{\pi}_1, \overrightarrow{\pi}_1) < (\overleftarrow{\pi}_2, \overrightarrow{\pi}_2)$.

At the same time, define the intersection and union operations as follows:

$$\begin{aligned} (\overleftarrow{\pi}_1, \overrightarrow{\pi}_1) \wedge (\overleftarrow{\pi}_2, \overrightarrow{\pi}_2) &= \{\sup((\overleftarrow{X}_i, \overrightarrow{X}_i) \wedge (\overleftarrow{X}_j, \overrightarrow{X}_j)) | (\overleftarrow{X}_i, \overrightarrow{X}_i) \in (\overleftarrow{\pi}_1, \overrightarrow{\pi}_1), \\ &\quad (\overleftarrow{X}_j, \overrightarrow{X}_j) \in (\overleftarrow{\pi}_2, \overrightarrow{\pi}_2)\} \\ (\overleftarrow{\pi}_1, \overrightarrow{\pi}_1) \vee (\overleftarrow{\pi}_2, \overrightarrow{\pi}_2) &= \{\inf((\overleftarrow{X}_i, \overrightarrow{X}_i) \vee (\overleftarrow{X}_j, \overrightarrow{X}_j)) | (\overleftarrow{X}_i, \overrightarrow{X}_i) \in (\overleftarrow{\pi}_1, \overrightarrow{\pi}_1), \\ &\quad (\overleftarrow{X}_j, \overrightarrow{X}_j) \in (\overleftarrow{\pi}_2, \overrightarrow{\pi}_2)\}. \end{aligned}$$

Through the previous analysis of DFDT, the set of instances that fall into each node constitutes a partial order relation. In DFBDT, a set of instances falling into each node of an adjacent layer forms a semi-lattice. After the dynamic fuzzy partition is given, there are some theorems about the relation between the two sets of decision trees:

Theorem 3.1 The set of DFDTs can be divided into a linear ordered set.

Proof: The process of constructing a DFDT uses branch attributes on the instances in the process of division. Thus, the lower set of examples of the division is finer than the upper instance set. $(\overleftarrow{\pi}_j, \overrightarrow{\pi}_j) < (\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$ (j is located at a lower level than i in the DFDT) is always established, even if the division is linear, as $<$ constitutes a partial order relationship. Therefore, the division of the set of instances is a linear ordered set.

Theorem 3.2 The set of DFDTs is composed of a set of dynamic fuzzy partitioning grids.

Proof: Let $(\overleftarrow{U}_\pi, \overrightarrow{U}_\pi)$ be the set of partitions of the set of instances in the DFDT. From Theorem 3.1, we know that $(\overleftarrow{U}_\pi, \overrightarrow{U}_\pi)$ is a poset. Let $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$, $(\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$ be an arbitrary division of the set of instances of DFDT in layers i and j ($j < i$). When $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) = \{(\overleftarrow{X}_i, \overrightarrow{X}_i) | 1 \leq i \leq m\}$, $(\overleftarrow{\pi}_j, \overrightarrow{\pi}_j) = \{(\overleftarrow{X}_j, \overrightarrow{X}_j) | 1 \leq j \leq m\}$, there are two cases:

(1) Layer j is directly below layer i . Then, there are two cases for $\forall (\overleftarrow{X}_i, \overrightarrow{X}_i) \in (\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$:

If $(\overleftarrow{X}_i, \overrightarrow{X}_i)$ is defined by the leaf node, then we should stop the division in the process of constructing the DFDT. Thus, $\exists (\overleftarrow{X}_j, \overrightarrow{X}_j) \in (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$ that makes $(\overleftarrow{X}_i, \overrightarrow{X}_i) = (\overleftarrow{X}_j, \overrightarrow{X}_j)$, that is, $(\overleftarrow{X}_i, \overrightarrow{X}_i) \supseteq (\overleftarrow{X}_j, \overrightarrow{X}_j)$.

If $(\overleftarrow{X}_i, \overrightarrow{X}_i)$ is not a terminal node, in the process of creating a tree, they are divided by selecting attributes, i.e. $(\overleftarrow{X}_i, \overrightarrow{X}_i) = (\overleftarrow{X}_j^1, \overrightarrow{X}_j^1) \cup (\overleftarrow{X}_j^2, \overrightarrow{X}_j^2) \cup \dots \cup (\overleftarrow{X}_j^{|\overleftarrow{V}_A, \overrightarrow{V}_A|}, \overrightarrow{X}_j^{|\overleftarrow{V}_A, \overrightarrow{V}_A|})$, where $(\overleftarrow{V}_A, \overrightarrow{V}_A)$ is the range of branch attribute $(\overleftarrow{A}, \overrightarrow{A})$. As j is directly below layer i , the choice of attribute $(\overleftarrow{V}_A, \overrightarrow{V}_A)$ for division $(\overleftarrow{X}_i, \overrightarrow{X}_i)$ falls into $(\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$, that is, $(\overleftarrow{X}_j^k, \overrightarrow{X}_j^k) \in (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$, where $k = 1, 2, \dots, |\overleftarrow{V}_A, \overrightarrow{V}_A|$.

It can be seen from the above that $(\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$ consists of a partition defined by the leaf node and a direct upper layer by selecting a partition of attributes. Thus, $(\overleftarrow{\pi}_j, \overrightarrow{\pi}_j) \prec (\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$ is established, that is:

$$\begin{aligned} (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j) \wedge (\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) &= (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j) \in (\overleftarrow{U}_\pi, \overrightarrow{U}_\pi) \\ (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j) \vee (\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) &= (\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \in (\overleftarrow{U}_\pi, \overrightarrow{U}_\pi). \end{aligned}$$

(2) Layer j is not directly below layer i .

In this case, there is a sequence $i, i+1, \dots, k, k+1, \dots, j$ between i and j . From Theorem 3.1, we know that $i, i+1, \dots, k, k+1, \dots, j$ is linear and orderly. Thus,

$$\begin{aligned} (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j) \wedge (\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) &= (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j) \in (\overleftarrow{U}_\pi, \overrightarrow{U}_\pi) \\ (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j) \vee (\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) &= (\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \in (\overleftarrow{U}_\pi, \overrightarrow{U}_\pi). \end{aligned}$$

From cases (1) and (2), we can determine that for $\forall (\overleftarrow{\pi}_i, \overrightarrow{\pi}_i), (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j) \in (\overleftarrow{U}_\pi, \overrightarrow{U}_\pi)$, both $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \wedge (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j) \in (\overleftarrow{U}_\pi, \overrightarrow{U}_\pi)$ and $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \vee (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j) \in (\overleftarrow{U}_\pi, \overrightarrow{U}_\pi)$ hold. From Theorem 3.1, we know that $(\overleftarrow{U}_\pi, \overrightarrow{U}_\pi)$ is a poset, so $(\overleftarrow{U}_\pi, \overrightarrow{U}_\pi)$ is a lattice. This is called a Dynamic Fuzzy Lattice, written as $\Pi(U)$.

3.3 DFDT special attribute processing technique

DFS is a good choice for describing the characteristics of DFD in the subjective and objective world. Therefore, in decision tree research, this theory is the first choice when processing the attributes of decision trees.

3.3.1 Classification of attributes

Knowledge representation is conducted through the knowledge expression system, the basic component of which is a collection of research objects with a description of their attributes and attribute values. The objects' attributes can be divided into four broad types: measure, symbol, switch, and degree [34], each of which has a value range. Thus, the objects' attributes can be divided into following three types by the range of attribute values.

Semantic attributes: Mostly refer to a degree or state, such as the strength of the wind, the level of the temperature, etc.

Numeric attributes: Generally a set on a real interval.

In the instance set corresponding to the data table that reflects the real problem, the attribute value can be semantic. However, there is an obvious drawback of semantic attributes: they cannot give a clear description of the attribute. Hence, some attributes are described as numeric attributes in which the value range is generally a real interval. For instance, the range of a height attribute may belong to the interval $[0, 300]$ (cm).

Enumerated attributes: A collection of unordered relations. The elements in the set are some symbols that represent a certain meaning.

The above three categories have no clear boundaries. When the range is {high, low}, it is a semantic attribute, and when its value range is $[-10, 40]$, it is a numeric attribute. Similarly, when the fractional attribute range is {1, 2, 3, 4, 5}, it is an enumerated attribute, and when its value range is $[0, 100]$, it is a numeric attribute.

The classification of attribute type is determined by the specific situation and the range of the attribute value. When the above three types of attributes correspond to a specific instance, the attribute value is determined. However, in the instance set, there are many reasons why an attribute may be missing; that is, the value of the attribute is unknown. Such attributes are called missing value attributes. Enumerated, numeric, and missing value attributes are commonly referred to as special attributes. The DFDT process for special attributes is described in the next section.

3.3.2 Process used for enumerated attributes by DFDT

In practical problems, some attribute values are limited to specific range. For example, there are only seven days in a week, 12 months in a year, and six classes in a week. When defining an enumerated type, all enumerated elements that are listed constitute the range (range of values) of this enumerated type, and the value of this enum type cannot exceed the defined range. The processing method for enumerated attributes is relatively simple. Take the gender attribute in an example set. The field of its attribute value is {male, female}. We replace these values so that, for example, male is and female is. The essence of this method is to map attributes to dynamic fuzzy

numbers. We can use another method known as discrete normalization to replace attribute values. In this case, we set male = 0, female = 1. The difference between the two methods is purely representational after the replacement of the attribute value.

3.3.3 Process used for numeric attributes by DFDT

1. Discretization and its meaning

In machine learning, many learning algorithms require the input attribute values to be discrete or can only handle semantic attributes rather than numeric attribute values. In actual problems, however, the data often contain a large number of numeric attributes, and current algorithms that can deal with numeric attributes sometimes cannot meet the requirements, which may lead to an unreasonable classification. Because of the existence of numerical attributes, many of the actual application processes run very slowly as the attribute values of the numeric dataset need to be sorted many times. To use this kind of algorithm to deal with numeric datasets, we need to process numeric attribute values in the dataset; that is, the value of the property domain (usually a real interval) is divided into several sub-intervals. This process is called numerical attribute discretization. In DFDT, numeric attributes are handled by a discretization operator.

Discretization of numerical attributes has the following three aspects:

(1) Improve the classification accuracy of the decision tree.

Through the discretization operation, the value range of the numerical attribute is mapped to a few intervals. This reduces the number of numerical attribute values, which is very useful for classification in decision trees. Taking ID3 as an example, the information entropy tends to take attributes with more values, and numerical attributes with more values are more likely to be selected for the classification in the next step. This ensures that the current node will export a lot of branches and that instance data in the lower nodes will be placed into the so-called “pure” state faster, that is, instances that fall into the node belong to the same category, but the number of instances is small (possibly only one instance). Thus, the final decision tree lacks adaptability, which means that the rule is not meaningful because of low support. That is, the number of data that can satisfy the rule is very small, and the performance of the classification is very poor. Hence, the discretization of numerical attributes can improve the accuracy of the decision tree.

(2) Because discretization reduces the value of the attribute, one effect of reducing and simplifying the data is to facilitate the storage of data. Additionally, the I/O operation in the process of creating the decision tree is reduced.

(3) The discretization of numerical attributes is more conducive to the transformation of data to knowledge, which is easily understood by experts and users.

2. Common discretization strategies

According to whether the classification information should be taken into consideration, discretization can be divided into unsupervised discretization and supervised discretization. Unsupervised discretization can be divided into equal intervals partitions and equal frequency partitions [36]. For the unsupervised discretization method, the completeness of the information may be undermined by the indiscriminate division of data in the discretization process, which may disrupt the usefulness of the information in the learning process [37]. Equal intervals partitioning often results in an imbalanced distribution of instances, with some intervals containing many instances and other intervals containing none. This can significantly impair the role of attributes in the process of building the decision tree. In contrast, equal frequency partitioning is insensitive to the class of the instance and may lead to useless thresholds. Division by unsupervised discretization is very rough, and the wrong selection of the threshold may result in useless partitions consisting of many different classification instances. Compared to unsupervised discretization, supervised discretization takes information of the class attribute into account during the discretization operation, so that a higher accuracy rate can be obtained in the classification.

The discretization algorithms have been roughly classified [38], and the commonly used discretization strategies are as follows:

The simplest partitioning strategy is bisection: the input space is divided into equal intervals, which are adapted to the distribution of values in the input space.

Adaptive method: First, the attribute value field is divided into two intervals, then the rules are extracted and the classification ability of the rule is evaluated. If the correct classification rate is below a certain threshold, one of the partitions is subdivided and the process is repeated.

Equal frequency interval method: After the user specifies the number of divided intervals, calculate the number of instances in each interval (the total number of objects divided by the number of intervals) according to the number of samples in each interval to determine the interval boundaries.

Class information entropy-based method: Using the principle of the smallest class of information entropy to find the sub-point q , the value of the range of numerical attributes is divided into k intervals.

In addition, there are many discretization methods, e.g. converting the discretization problem into a Boolean reasoning problem; applying the χ^2 distribution to continuous data; and the 1R method.

3. Discretization based on information entropy

The discretization method based on information entropy is a supervised discretization method. It introduces the concept of information entropy and makes full use of the information of class attributes, which makes it more likely to exactly locate the boundary of the partitioned interval. As a result, this approach is widely used.

Suppose U is the instance set and the number of classes of decision attributes is m . Let $x \in U$ and $V_A(x)$ be the value of x in attribute A . Attribute A is a numeric attribute, and so the value set of is

$$U_A = \{V_A(x) | x \in U\} = \{a_1, a_2, \dots, a_n\}.$$

The process of discretization for A is as follows:

- (1) Sort the values of attribute A into a sequence a_1, a_2, \dots, a_n .
- (2) Each $\text{Mid}_i = (a_i + a_{i+1})/2$ ($i = 1, 2, \dots, n-1$) is a possible interval boundary. Mid_i is called a candidate segmentation point, and divides the instance set into two sets:

$$U_i^{\triangleleft} = \{x \in U | V_A(x) \leq \text{Mid}_i\}, U_i^{\triangleright} = \{x \in U | V_A(x) > \text{Mid}_i\}.$$

Mid_i is chosen to minimize the entropy of U after partitioning, and the entropy is calculated as follows:

$$\text{Entropy}(U, \text{Mid}_i) = \frac{|U_i^{\triangleleft}|}{|U|} \text{Entropy}(U_i^{\triangleleft}) + \frac{|U_i^{\triangleright}|}{|U|} \text{Entropy}(U_i^{\triangleright}),$$

where the formula for $\text{Entropy}(U_i^{\triangleleft})$ and $\text{Entropy}(U_i^{\triangleright})$ is given in (3.2). The instance set U is divided into two subsets $U_i^{\triangleleft}, U_i^{\triangleright}$.

- (3) If $\text{Entropy}(U, \text{Mid}_i) \leq \omega$, then stop dividing; otherwise, $U_i^{\triangleleft}, U_i^{\triangleright}$ are divided according to steps (1) and (2). ω is the threshold for stopping the partitioning.

For example, we can discretize the temperature attribute values in the weather data, the value of which is given in Tab. 3.5.

These values correspond to 11 possible separation points, and the information entropy of each point can be calculated in the usual way. For instance, temperature < 71.5 divides the instance into four positive and two negative data, corresponding to five “yes” and three “no” answers. Thus, the information entropy is

$$\begin{aligned} \text{Entropy}(U, 71.5) &= (6/14) \times \text{Entropy}(U_{71.5}^{\triangleleft}) + (8/14) \text{Entropy}(U_{71.5}^{\triangleright}) \\ &= 0.939. \end{aligned}$$

This represents the information entropy required to distinguish between individual yes and no values by segmentation. In the process of discretization, the point at which

Tab. 3.5: Attribute values of temperature.

64	65	68	69	70	71	72	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	No	Yes	Yes	No

Tab. 3.6: Discretization of temperature.

64	65	68	69	70	71	72	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	No	Yes	Yes	No
F			E			D	C	B		A	

the information entropy is minimized in each operation is selected in order to make the instance class in the interval as simple as possible. The final results are presented in Tab. 3.6.

Although the discretization algorithm based on information entropy is quite successful in some applications, the threshold for the termination of the partitioning process must be set. Therefore, some problems will arise when the interval is partitioned [39]. If a uniform threshold is used for multiple numerical attributes, the number of intervals after attribute discretization will be very different, which may affect the generation of decision tree support rules. An improper choice of the threshold value may also cause the condition attribute value of the original instance after discretization to be the same, as well as giving conflicting data for different decision attribute values (if the condition attribute value of two instances is the same but the decision attribute values are different, the instance set is inconsistent). Therefore, the discretization method based on information entropy lacks operability, and a new method should be developed to make the discretization process simpler, more accurate and more consistent.

4. Discretization method based on dynamic fuzzy partition lattice

The Dynamic Fuzzy Partitioning Lattice was defined in Section 3.2 to discuss the relationship between the partitions of the instance set in the DFDT layers. The Dynamic Fuzzy Partition Lattice divides the finite definite region of the instance set. The range of the numerical attributes is also a definite region, the division of which is limited.

Consider a DFDT $(\overleftarrow{T}, \overrightarrow{T}) = \langle (\overleftarrow{U}, \overrightarrow{U}), (\overleftarrow{C}, \overrightarrow{C}), (\overleftarrow{D}, \overrightarrow{D}), (\overleftarrow{V}, \overrightarrow{V}), (\overleftarrow{F}, \overrightarrow{F}) \rangle$, where $|\overrightarrow{U}| = n$ and $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ are numeric attributes. To discretize the attributes $(\overleftarrow{A}_i, \overrightarrow{A}_i)$, we must first sort them. The values of $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ are arranged in ascending order, and a separation point is inserted at the interval of the data sequence to obtain a division of $(\overleftarrow{A}_i, \overrightarrow{A}_i)$:

$$\{[(\overleftarrow{v}_1, \overrightarrow{v}_1), (\overleftarrow{v}_2, \overrightarrow{v}_2)], [(\overleftarrow{v}_2, \overrightarrow{v}_2), (\overleftarrow{v}_3, \overrightarrow{v}_3)], \dots, [(\overleftarrow{v}_{k-1}, \overrightarrow{v}_{k-1}), (\overleftarrow{v}_k, \overrightarrow{v}_k)]\}.$$

Each interval is replaced with a dynamic fuzzy number $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$, thus completing the discretization of $(\overleftarrow{A}_i, \overrightarrow{A}_i)$. A division of the range of $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ is called a discretization scheme. If two discretization schemes have the same division, then the two discretization schemes are equivalent.

As $|\langle \overleftarrow{U}, \overrightarrow{U} \rangle| \triangleq n$, we have $\max(|\overleftarrow{V}_{\overrightarrow{A}_i}|, |\overrightarrow{V}_{\overrightarrow{A}_i}|) = n$. This is because there can be multiple separation points for the same value interval, and the resulting discretization schemes are equivalent. Therefore, it is possible to specify that a separation point can only exist at a value interval, and the number of separation points may be $0, 1, 2, \dots, n - 1$. When the number of separation points is j , there are C_{n-1}^j methods for setting the separation points. There are $\sum_{j=0}^{n-1} C_{n-1}^j$ kinds of discretization schemes in

which the attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ is not equivalent. If there are m numerical attributes in the instance set, the number of corresponding discretized sets is at most $2^{(n-1)m}$. If the given instance set is finite, regardless of which method has been used, the discretization has a limited range. That is, the discrete set of the instance set is a finite field.

Suppose the condition attribute of the instance set is $(\overleftarrow{C}, \overrightarrow{C}) = \{(\overleftarrow{A}_1, \overrightarrow{A}_1), (\overleftarrow{A}_2, \overrightarrow{A}_2), \dots, (\overleftarrow{A}_n, \overrightarrow{A}_n)\}$ and the range of attribute $(\overrightarrow{A}_i, \overrightarrow{A}_i) \in (\overleftarrow{C}, \overrightarrow{C})$ is $(\overleftarrow{V}_{\overrightarrow{A}_i}, \overrightarrow{V}_{\overrightarrow{A}_i})$.

A discretization scheme for a single numerical attribute $(\overrightarrow{A}_i, \overrightarrow{A}_i)$ is a partition $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$ of $(\overleftarrow{V}_{\overrightarrow{A}_i}, \overrightarrow{V}_{\overrightarrow{A}_i})$.

The instance set $(\overleftarrow{U}, \overrightarrow{U})$ filtered by attributes is denoted as $(\overleftarrow{U}_N, \overrightarrow{U}_N)$. $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ is the conditional attribute of $(\overleftarrow{U}_N, \overrightarrow{U}_N)$, and $(\overrightarrow{A}_i, \overrightarrow{A}_i)$ is numerical. Thus, the discretization of $(\overleftarrow{U}_N, \overrightarrow{U}_N)$ has multiple attributes.

Suppose that $(\overleftarrow{\pi}_1, \overrightarrow{\pi}_1) = \{(\overleftarrow{\pi}_1^1, \overrightarrow{\pi}_1^1), (\overleftarrow{\pi}_2^1, \overrightarrow{\pi}_2^1), \dots, (\overleftarrow{\pi}_n^1, \overrightarrow{\pi}_n^1)\}$ and $(\overleftarrow{\pi}_2, \overrightarrow{\pi}_2) = \{(\overleftarrow{\pi}_1^2, \overrightarrow{\pi}_1^2), (\overleftarrow{\pi}_2^2, \overrightarrow{\pi}_2^2), \dots, (\overleftarrow{\pi}_n^2, \overrightarrow{\pi}_n^2)\}$ are two discretization schemes for instance set $(\overleftarrow{U}_N, \overrightarrow{U}_N)$, where $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$ is the discretization scheme for attribute $(\overrightarrow{A}_i, \overrightarrow{A}_i)$ in $(\overleftarrow{\pi}_1, \overrightarrow{\pi}_1)$, $(\overleftarrow{\pi}_j^2, \overrightarrow{\pi}_j^2)$ is the discretization scheme for attribute $(\overrightarrow{A}_j, \overrightarrow{A}_j)$ in $(\overleftarrow{\pi}_2, \overrightarrow{\pi}_2)$.

As $(\overleftarrow{\pi}_j^i, \overrightarrow{\pi}_j^i)$ is the discretization scheme for attribute $(\overrightarrow{A}_j, \overrightarrow{A}_j)$, the partial order relation in the discretization is defined as follows:

Discretization scheme $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \leq (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$ if and only if $(\overleftarrow{\pi}_k^i, \overrightarrow{\pi}_k^i) \leq (\overleftarrow{\pi}_k^j, \overrightarrow{\pi}_k^j)$ ($k = 1, 2, \dots, n$) is satisfied for each attribute $(\overrightarrow{A}_k, \overrightarrow{A}_k)$ in instance set $(\overleftarrow{U}_N, \overrightarrow{U}_N)$, where $(\overleftarrow{\pi}_k^i, \overrightarrow{\pi}_k^i)$ is the division of $(\overleftarrow{V}_{\overrightarrow{A}_k}, \overrightarrow{V}_{\overrightarrow{A}_k})$ in $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$. The relation \leq defines the degree of discretization among the discretization schemes as a partial order relation.

$(\overleftarrow{\pi}_k^i, \overrightarrow{\pi}_k^i) \leq (\overleftarrow{\pi}_k^j, \overrightarrow{\pi}_k^j)$ indicates that $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$ divides the value of attribute $(\overrightarrow{A}_k, \overrightarrow{A}_k)$ into a range. Then, the division of the value of attribute $(\overrightarrow{A}_k, \overrightarrow{A}_k)$ in $(\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$ must belong to the same range. Two or more adjacent segmentation intervals of the value of attribute $(\overrightarrow{A}_k, \overrightarrow{A}_k)$ in $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$ are merged into one interval of attribute $(\overrightarrow{A}_k, \overrightarrow{A}_k)$ in $(\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$. As $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \leq (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$, $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$ is the refinement of $(\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$.

The intersection and union operations of the discretization scheme for instance set $(\overleftarrow{U}_N, \overrightarrow{U}_N)$ are

$$(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \bigcap (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j) = \{(\overleftarrow{\pi}_1^i, \overrightarrow{\pi}_1^i) \wedge (\overleftarrow{\pi}_1^j, \overrightarrow{\pi}_1^j), (\overleftarrow{\pi}_2^i, \overrightarrow{\pi}_2^i) \wedge (\overleftarrow{\pi}_2^j, \overrightarrow{\pi}_2^j), \dots, (\overleftarrow{\pi}_n^i, \overrightarrow{\pi}_n^i) \wedge (\overleftarrow{\pi}_n^j, \overrightarrow{\pi}_n^j)\}$$

$$(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \bigcup (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j) = \{(\overleftarrow{\pi}_1^i, \overrightarrow{\pi}_1^i) \vee (\overleftarrow{\pi}_1^j, \overrightarrow{\pi}_1^j), (\overleftarrow{\pi}_2^i, \overrightarrow{\pi}_2^i) \vee (\overleftarrow{\pi}_2^j, \overrightarrow{\pi}_2^j), \dots, (\overleftarrow{\pi}_n^i, \overrightarrow{\pi}_n^i) \vee (\overleftarrow{\pi}_n^j, \overrightarrow{\pi}_n^j)\}.$$

The partition of instance set $(\overleftarrow{U}_N, \overrightarrow{U}_N)$ divided by $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \cap (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$ is the biggest partition that is smaller than that divided by $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$ and $(\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$. Similarly, $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \cup (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$ is the smallest partition that is bigger than both $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$ and $(\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$.

Define the partial order \leq and the intersection and union operations of the discretization scheme. Then, the lattice structure with the discretization scheme of the instance set as an element can be obtained. This is called the Dynamic Fuzzy Discrete Lattice [36].

Because all kinds of discretization schemes are elements of the Dynamic Fuzzy Discrete Lattice, the discretization of instance set $(\overleftarrow{U}_N, \overrightarrow{U}_N)$ is transformed into a search problem on the Dynamic Fuzzy Discrete Lattice. When attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ is discretized, there are many separation points in $(\overleftarrow{V}_{\overleftarrow{A}_i}, \overrightarrow{V}_{\overrightarrow{A}_i})$. If there are some separation points that will not affect the discretization result after being deleted, these nodes are redundant separation points.

Definition 3.4 For attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$, there is a set of separation points $(\overleftarrow{S}, \overrightarrow{S}) = \{(\overleftarrow{S}_1, \overrightarrow{S}_1), (\overleftarrow{S}_2, \overrightarrow{S}_2), \dots, (\overleftarrow{S}_k, \overrightarrow{S}_k)\}$. If $(\overleftarrow{S}_i, \overrightarrow{S}_i) \in (\overleftarrow{S}, \overrightarrow{S})$ is deleted and its two separate intervals are merged, the attribute values in the two intervals are replaced by the same dynamic fuzzy number $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$. The resulting instance set is the same, and the separation point $(\overleftarrow{S}_i, \overrightarrow{S}_i) \in (\overleftarrow{S}, \overrightarrow{S})$ is called a redundant separation point.

Definition 3.5 The number of redundant separation points in a discretization scheme of attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ is called the separation point level.

Obviously, the higher the separation point level, the more corresponding redundant points exist for the discretization of this attribute.

The goal of the discretization of attributes is to keep the discretization results consistent and to minimize the number of redundant separation points [37]. The basic steps are as follows: calculate the separation point level of each attribute; sort attributes in ascending order of the separation point level; delete the redundant divider points one by one. The algorithm is described as:

- (1) Find the separation point level for each discretization attribute and the set of separable points that can be removed.
- (2) Each attribute is sorted in ascending order according to the separation point level, and a sequence of attributes is obtained.
- (3) Each attribute in the attribute sequence is processed one after the other. If there is a redundant separation point at the i th attribute, merge the interval adjacent to this separation point and judge the consistency of the instance set. The number

of redundant separation points for the i th attribute decreases by one, and the instance set is updated. Otherwise, the separation point is retained. Go to (3) and perform the next round of operations on redundant separation points.

- (4) If there are no redundant separation points for each attribute in the attribute sequence, the algorithm ends and a discretized instance set is obtained.

It can be seen from the algorithm that the operation of deleting redundant separation points enhances the discretization scheme, and a sequence of discretization schemes with decreasing separation points is obtained. The two adjacent discretization schemes in the sequence satisfy the partial order relation \leq . The last scheme in the sequence is the discretization scheme of instance set $(\overleftarrow{U}_N, \overrightarrow{U}_N)$.

In the discretization based on the Dynamic Fuzzy Partition Lattice, the first consideration is the consistency of the instance set. The consistency can limit the degree of approximation to the original set, avoiding the conflict of data caused by discretization based on information entropy. By operating according to the value of the attributes without artificially terminating the algorithm, the impact of human factors in the discretization of the instance set is reduced.

3.3.4 Methods to process missing value attributes in DFDT

1. Reasons for data loss

Missing attribute values are a frequent and inevitable occurrence in datasets that reflect practical problems. Therefore, in most cases, the information system is incomplete, or there is a certain degree of incompleteness. The reasons for missing data are multifaceted [33, 42] and concluded as follows:

- (1) Some information is temporarily unavailable. For example, in medical databases, not all of the clinical test results of all patients can be given within a certain time, resulting in some of the attribute values being empty. Furthermore, the response to certain questions depends on the answers to other questions in application form data.
- (2) Some information is missing. This may be because the data are considered unimportant when they are input, simply forgotten, or some misunderstanding occurs. Additionally, it is possible that the data acquisition equipment, storage media, or transmission media have failed, or some human factor has resulted in the data loss.
- (3) One or more properties of some object are not available. In other words, the attribute value does not exist for this object. Examples include an unmarried person asked for their spouse's name or a child's fixed income status.
- (4) Some information is (considered) unimportant.
- (5) The cost of obtaining this information is too high.

- (6) Real-time performance requirements of the system are high, meaning that judgments or decisions must be made before such information can be obtained.

2. Importance and complexity of processing missing values

Data loss is a complex problem in many fields of research. The existence of null values has the following effects [40]: First, the system has lost some useful information. Second, the uncertainty of the system increases, as deterministic elements in the system are more difficult to quantify. Finally, data that contain null values will lead to confusion in the learning process, resulting in unreliable output.

Decision tree algorithms are committed to avoiding the problem of overfitting the training set. This feature makes it difficult to deal with incomplete data. Therefore, missing values need to be handled through specialized methods such as filling to reduce the distance between the algorithms and reality.

3. Analysis and comparison of missing value processing methods

There are three main methods for dealing with incomplete instance sets:

A. Delete the tuple

That is, delete the instance that has missing information attribute values, leaving a complete set of instances. This method is simple, effective in cases where the instance includes multiple attribute missing values, and the number of deleted objects with missing values is very small compared to the whole set of instances. However, this approach has great limitations. It will result in wasted resources, as the information hidden in these instances will be discarded. If the set includes a small number of instances, deleting just a few objects could seriously affect the information objectivity and the correctness of the results. When the percentage of the value of each property changes significantly, performance becomes very poor. Thus, when a large percentage of missing data is available, particularly when missing data is not randomly distributed, this approach can lead to deviations in the data and erroneous conclusions [44].

B. Complete the data

This method completes the set of instances by filling in the missing values. The values used to fill in the missing data are usually based on statistical principles according to the distribution of the other objects in the dataset, e.g. average value of the remaining attributes. Table 3.7 lists some commonly used methods of filling in missing values.

C. Not handled

Create the decision tree directly based on the set of instances that contain missing values.

Tab. 3.7: Comparison of missing value filling methods [45–48].

Methods	Theory	Advantage	disadvantage
Filling manually	Filled by users	Least data deviation, best filling performance	Costs too much time, unworkable when the data scale is large May cause serious data deviation
Treating missing attribute values as Special values	Treat missing attribute as a special one which is different from all of other attributes	Guess the missing value using available data	
Mean/mode completer	When the missing value is a value, fill it using the mean of all other values; otherwise, use the one of the highest frequency	Simple conceptually, use the relationship between data to fill the missing value Good filling performance	Hard to formulate the standard, and too many subjective factors
Hot deck imputation	Find the most similar one to fill the missing value		Costs too much when missing a large number of values
Assigning all possible values of the Attribute K-means clustering	Fill the missing one using all possible values		Costs too much when missing a large number of values Get deviation when the values are not linearly dependent or the missing values are of high dependence
Combinatorial completer	Find the nearest K samples with Euclidean distance or correlation analysis, and the mean of the k values to fill the missing value, Try every possible value of missing value, and find the best one Establish regression equation to figure out the miss value	Can find a good and concise result	May trap in local extremum, convergence speed is not fast, hard to compute
Regression			Can handle only noun-type attributes
Expectation maximization	An iterative algorithm to compute maximum likelihood estimator or posterior distribution under situation of incompleteness data		
C4.5	Find the relationship between values to fill the missing one		

4. Classification of missing values in DFDT

The missing attribute values should be divided into the following three cases and handled separately in DFDT:

- (1) Attribute values that contribute no information to the classification;
- (2) Attribute values that partly contribute information to the classification;
- (3) Attribute values that play an important role in the classification, i.e. important sources of information.

An example is given below to illustrate how DFDT treats these three types of missing attribute values. Assume that the following attributes are present in the dataset that determines the incidence of influenza: name, gender, geography, age, onset time, temperature. We seek to identify those attribute and classification results that are not necessarily linked; i.e. for this category name, the attribute does not contribute any information. There are two types of treatment for such attributes:

- (1) Delete this attribute directly from the dataset.

This approach is essentially the first method of dealing with incomplete data, but it also inherits the shortcomings of the delete tuple method.

- (2) Conduct a type of discretization operation. For example, assign a dynamic fuzzy number to the name attribute of all instances (for instance, “name” = $\overrightarrow{0.5}$). The advantage of this approach is that it maintains the integrity of the dataset.

The time attribute contributes some information to the classification; e.g. from the “2005-10-10” value of the “onset time” property, we may infer that October is the high incidence period of influenza. We first simplify the attribute values and retain the useful information for classification. For the attributes that have been simplified, their values make a complete contribution to the classification; i.e. the contribution has increased from a partial contribution to an important contribution. When the simplified attribute value is missing, the same processing method as for important attributes is applied. The processing of important attributes is divided into techniques for the training instance set and the test set [37].

Processing methods for the training set:

- (1) Ignore the instance that contains the missing value attribute.
- (2) Assign values to the attribute using information from other instances.

Assume that the attribute of instance A belonging to class C is unknown. Values are assigned to A according to the other instances belonging to class C .

- (3) Obtain the attribute value using the decision tree method.

Assume that the training set S has a partially unknown value of attribute A . Select the instances that in which attribute A is known from S , and compose the subset S' using these selected instances. Create a decision tree in set S' by processing the original classification attribute as a general attribute. Let A be the classification attribute of S' , and generate the decision tree and rules that can be used to assign values to attribute A in set $S - S'$.

(4) Assign value “unknown” to the attribute.

These methods have advantages and disadvantages. Method (1), although simple, will inevitably lead to the loss of information and the destruction of information integrity. Method (3) is complicated and applies only to the case where relatively few attribute values are missing. Method (4) considers the value “unknown” as an attribute value in the dataset, which changes the domain of the value. Although method (2) may be affected by subjective factors, and the criterion for a correlation between instances is difficult to determine, it is known to be highly practical. Because of the obvious disadvantages of the other three methods, we use method (2) to deal with important missing value attributes in the training set in DFDT.

After processing the missing value attributes in the training set, we can create the decision tree. We should test the decision tree using the test instance set. When classifying instances of the test set, if the instance contains missing attribute values, the classification immediately becomes impossible because we do not know which branch this instance will go into in the decision tree. In this case, we need to deal with the missing attribute values in the test set. There are many ways to deal with missing attribute values in a test set, such as agent attribute segmentation (Breiman et al. 1984) [51], dynamic path generation (White 1987, Liu & White 1994) [52, 53] and the lazy decision tree method (Friedman et al. 1996) [54]. In DFDT, the action is similar to that taken for the training instance set: a value is assigned to the attribute using information from other instances.

3.4 Pruning strategy of DFDT

3.4.1 Reasons for pruning

After feature extraction and the processing of special features in the previous stage, we can now build the decision tree according to the training dataset. The decision tree is built on the basis of classifying the training data. Next, we classify the surplus instances in the instance space. At this point, we must determine whether the established decision tree achieves acceptable performance. For this, a validation process is needed. This may involve extra instances from the test set or the application of cross-validation. Here, we adopt algorithm C4.5 with cross-validation and conduct tests on several datasets from the UCI repository (further information can be found in Tab. 3.8). The test data are presented in Tab. 3.9.

From Tab. 3.9, it can be seen that if there is noise in the data or the number of training instances is too small to obtain typical samples, we will have problems in building the decision tree.

Tab. 3.8: Information of UCI dataset.

Data	Instance number	Properties number	Classifications number	Missing
Weather	14	5	2	No
Contact lenses	24	4	3	No
Iris plants dataset	150	5	3	No
Labor	57	16	2	Yes
Image segmentation training	810	19	7	No
Image segmentation test	2100	19	7	No
Soybean	683	35	19	yes

Tab. 3.9: Decision tree accuracy.

Data	Evaluation on training set	Stratified cross-validation	Number of fold
Weather	100%	68.2857%	3
Contact lenses	91.6667%	87.5%	3
Iris plants dataset	98%	96%	10
Labor	87.7193%	77.193%	3
Image segmentation training	98.7654%	93.4568%	10
Image segmentation test	99%	95.7333%	10
Soybean	96.3397%	91.5081%	10

For a given assumption, if there are other assumptions that fit the training instances worse but fit the whole instance space better, we say that the assumption overfits the training instances.

Definition 3.6 Giving an assumption space H and an assumption $h \in H$, if there is another $h' \in H$ such that the error rate of h is smaller than that of h' in the training instances but the error rate of h' is smaller across the whole set of instances, we say that h overfits the training instances.

What causes tree h to fit better than tree h' on the training instances but achieve worse performance on the other instances? One reason is that there is random error or noise in the training instances. For example, we add a positive training instance in PlayTennis (Tab. 3.3), but it is wrongly labelled as negative:

Outlook = Sunny, Temperature = Hot, Humidity = Normal

Wind = Strong, PlayTennis = No

We build the decision tree from positive data using algorithm ID3 and generate the decision tree (Fig. 3.4). However, when adding the wrong instance, it generates a more complex structure (Fig. 3.6).

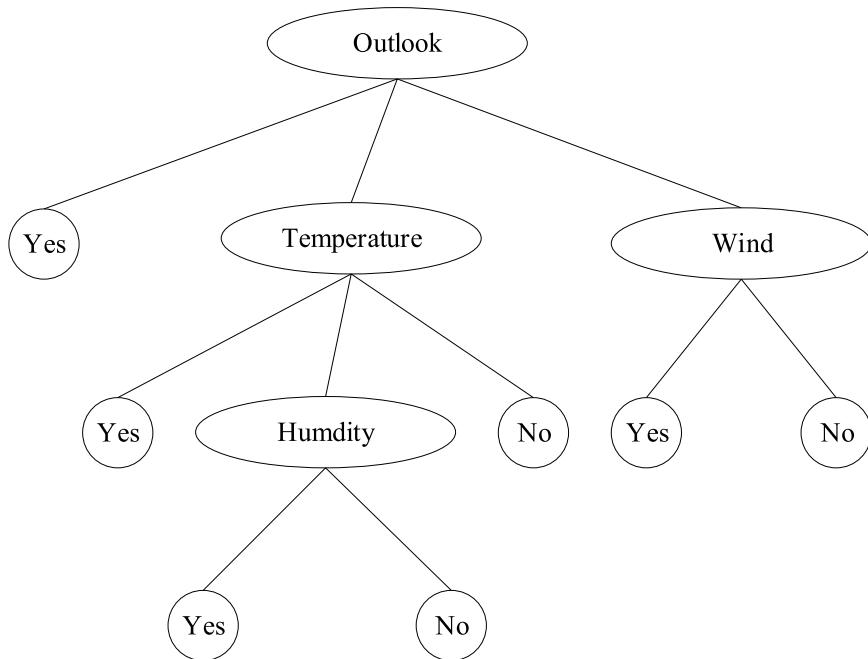


Fig. 3.6: Overfitting decision tree.

Certainly, h will fit the training instance set perfectly, whereas h' will not. However, the new decision node is produced by fitting the noise in the training instances. We can conclude that h' will achieve better performance than h in the data extracted from the same instance distribution.

From this example, we can see that random error or noise in the training instances will lead to overfitting. However, overfitting also occurs without noise, especially when fewer instances are associated.

To avoid a complex and huge decision tree caused by overfitting and too many branches, we apply a pruning process to the branches.

3.4.2 Methods of pruning

There are several methods to prevent overfitting in learning a decision tree. These methods are divided into two classes.

Pre-pruning: Stop the growth of the tree before correctly classifying the training set completely. There are many different methods to determine when to stop the growth of the decision tree. Pre-pruning is a direct strategy that prevents overfitting during tree building. At the same time, there is no need for pre-pruning to generate the whole decision tree, leading to high efficiency, which makes it a suitable strategy

for large-scale questions. However, there is a disadvantage. It is difficult to accurately identify when to stop the growth. That is, under the same standard, maybe the current extension is unable to meet the demand, but a further extension will. Thus, if the algorithm terminates too early, useful information will be missed.

Post-pruning: Overfitting is permitted, and then pruning is applied. Post-pruning, as proposed by Breiman et al. (1984), has been widely discussed and can achieve great success in applications such as Cost-Complexity Pruning, Minimum Error Pruning, Pessimistic Error Pruning, Critical Value Pruning, Reduced Error Pruning, and Error-Based Pruning (EBP). Table 3.10 gives a summary of several decision tree pruning methods.

3.4.3 DFDT pruning strategy

Unlike pre-pruning, which is suitable for large-scale problems, post-pruning is known to be successful in practice. In the preliminary stage of DFDT, the scale of the dataset is not too large. We prefer not to interpose when building DFDT, so that we obtain an unpruned original decision tree. Then, we can determine whether DFDT is good or bad from the classification error rate of the original decision tree. Thus, post-pruning is much more suitable for DFDT. There is no need for EBP to use an extra pruning instances set. Through the process of pruning, DFDT can attain low complexity under the premise of the original predictive power.

EBP checks each subtree from bottom to top, computes the estimated error rate for each subtree, and returns it as the error rate of a leaf node. If a branch is unable to reduce the error rate, then it will be pruned. In other words, if the error rate of the subtree is larger than that of the leaf node, the subtree is returned to a leaf node.

The estimated error rate of a leaf node is given by

$$e_t = \frac{f + \frac{Z^2}{2N} + Z \sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{Z^2}{4N}}}{1 + \frac{Z^2}{N}} \quad (3.4)$$

where N is the number of instances of the leaf node, $f = E/N$ is the classification error rate of the node, and E is the number of erroneous classifications. When the reliability is 25%, Z is 0.69.

The estimated error rate of a subtree is given by

$$e_T = \sum_{i=1}^k \frac{N_i}{N} e_i, \quad (3.5)$$

where N is the number of instances in the root node number of subtree T , and N_i is the number of instances of T in the i th node.

Tab. 3.10: Comparison of different pruning methods. [9, 10, 48, 49]

Methods	Property and strategy	Instance set of pruning	Advantages	Disadvantages and restrictions
Minimal cost-complexity pruning	Check all nodes from bottom to top, and use cross-validation or standard error to measure accuracy	No need in cross-validation	Generate a tree of suitable size	Unstable accuracy and over-pruning
Minimum error pruning	From bottom to top and based on estimate error rate	Need		Sensitive to class number, unstable under noise, low accuracy, and need extra instances when pruning
Laplacian error estimation	Use Laplace probability estimation as expected error rate	Need		
Pessimistic error pruning	From top to bottom and use continuous correction factor	No need	High efficiency, faster than Cost-Complexity and Reduced Error pruning, and no need of extra pruning instance set	Unstable under noise, level effect caused by top-bottom
Critical value pruning	Use a critical value as the standard for branch choosing to determine whether to hold the branch	Need		Weak accuracy, difficult to set critical value, and need extra instances to do pruning
Reduced error pruning	From bottom to top, find the most suitable pruning tree, and do pruning to instance set	Need	Good accuracy, suitable tree size	Need extra instances to do pruning, rare instances will be ignored
Error-based pruning	From bottom to top, estimate confidence interval, and use training instance set	No need	High accuracy, no need of extra pruning instance set, fast speed	

Next, we use an example to explain the pruning process. The system adopts a bottom-to-top strategy to check all subtrees in the decision tree and compute e_t and e_T . If $e_t < e_T$, then the pruning is conducted as shown in Fig. 3.7.

First, we find a subtree and compute e_{t4} using (3.4). That is, we return the error rate to that of the leaf node. Next, we use (3.5) to compute e_{T4} , saving the estimated error rate of the subtree. At this point, e_{T4} ($= 0.487$) is smaller than e_{t4} ($= 0.591$); thus, saving $T4$ can reduce the error rate, so we do not prune $T4$. The procedure is repeated for all subtrees. When we have checked the root node, the whole tree has been pruned.

$$e_{t4} = \frac{\frac{7}{14} + \frac{(0.69)^2}{2 \times 14} + (1.69) \sqrt{\frac{7/14}{14} - \frac{(7/14)^2}{14} + \frac{(7/14)^2}{4 \times (14)^2}}}{1 + \frac{(0.69)^2}{14}} = 0.590\ 676$$

$$e_{t5} = \frac{\frac{3}{8} + \frac{(0.69)^2}{2 \times 8} + (1.69) \sqrt{\frac{3/8}{8} - \frac{(3/8)^2}{8} + \frac{(3/8)^2}{4 \times (8)^2}}}{1 + \frac{(0.69)^2}{8}} = 0.496\ 973\ 6$$

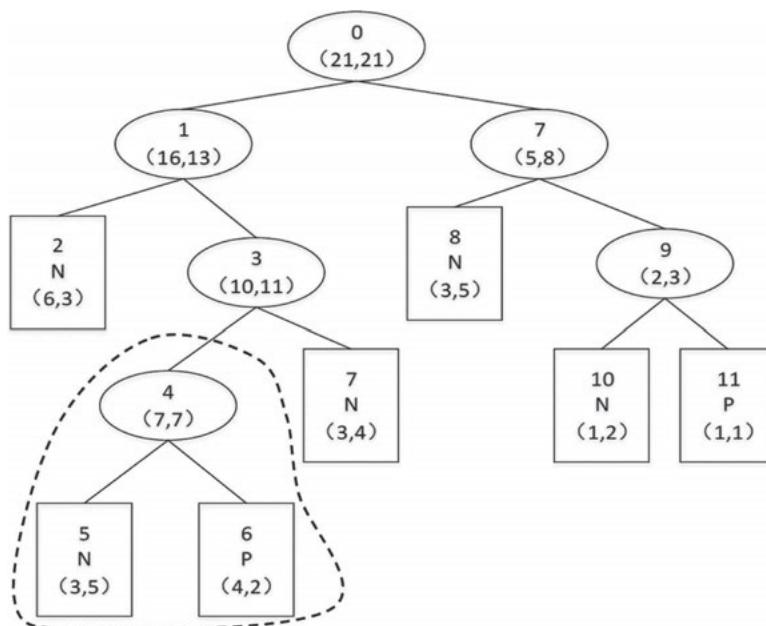


Fig. 3.7: Pruning of decision tree.

$$e_{t6} = \frac{\frac{2}{6} + \frac{(0.69)^2}{2 \times 6} + (1.69) \sqrt{\frac{2/6}{6} - \frac{(2/6)^2}{6} + \frac{(2/6)^2}{4 \times (6)^2}}}{1 + \frac{(0.69)^2}{6}} = 0.473\ 988\ 2$$

$$e_{T4} = \frac{8}{14} \times 0.496\ 973\ 6 + \frac{8}{14} \times 0.473\ 988\ 2 = 0.487\ 123$$

While pruning, EBP deletes subtrees and then grafts them to the father node of the deleted tree.

3.5 Application

Decision tree algorithms are an active area of research in the field of data mining. They incorporate multi-stage decision processes. Because of the rapidity of decision tree generation and the clarity and comprehensibility of their structures, it is easy to extract commercially valuable information that enables decision makers to make the right decisions. Thus, decision trees are widely used to good effect in the military, medicine, securities and investment, exploration, and corporate governance. In medicine, a decision tree model for pancreatic cancer analysis successfully concluded that a patient should first have an endoscopy; in securities, decision trees are applied more widely, such as the decision tree model used in the ISCRMS system [50] (ISCRMS is a customer intelligence analysis solution developed by applying data mining/online analytical techniques in the security field); and customer classification has been applied to data mining in the banking industry. In other respects, the decision tree model has been widely used in company management, exploration, and so on.

Although the domain of decision tree models varies, different applications have the same essence – using a decision tree algorithm to process and classify data and providing a friendly, easy-to-use data mining application environment. After datasets have been obtained for different problems, the attributes can be processed according to the DFDT processing method described in the preceding sections of this chapter, and the decision tree can be constructed and pruned. The following sections compare our DFDT with the C4.5 decision tree algorithm.

3.5.1 Comparison of algorithm execution

A decision tree algorithm consists of three steps: (1) data preprocessing, that is, data discretization (or dynamic fuzzification), (2) decision tree construction in accordance

with the attribute selection algorithm to build the decision tree, and (3) pruning to prevent overfitting.

The purpose of data preprocessing is to apply to the attribute selection algorithm. In the data preprocessing stage, DFDT uses different processing methods according to the situation and treats the attributes for discretization and dynamic fuzzification. The fuzzified attribute values are dynamic fuzzy numbers within the interval $[(\overleftarrow{0}, \overleftarrow{0}), (\overrightarrow{1}, \overleftarrow{1})]$, as presented in Tab. 3.1. The fuzzified attribute-value pairs are $\langle \text{Effort} = \text{high}, \overleftarrow{0.7} \rangle$.

As absolute clarity does not exist in the real world, it is more natural and reasonable to describe the characteristics of the instance after applying fuzzification to the attributes.

3.5.2 Comparison of training accuracy

A dataset from the UCI-Machine-Learning database was tested using DFDT and C4.5. The raw data with a test instance set was processed, the training instance set was combined with the test instance set, and cross-validation was used to test the constructed decision tree. The number of cross-validated directories was set to 10. The information in the data table is presented in Tab. 3.11.

The C4.5 algorithm and DFDT algorithm were applied to the UCI dataset, which was divided into a pruned decision tree and an unpruned decision tree. We recorded the number of leaf nodes generating the decision tree, the size of the tree, the number of instances of the correct classification on the training instance set, and the number of instances of correctly categorized cross-validation. Table 3.12 presents the results obtained using the C4.5 algorithm for the UCI dataset, and Tab. 3.13 presents those from the DFDT algorithm.

Tab. 3.11: Information of UCI.

Name	Instance number	Attribute number	Class number	Missing
Fitting contact lenses	24	5	3	no
Hepatitis	155	20	2	yes
Glass identification	214	10	6	no
BUPA liver disorders	345	7	2	no
Ionosphere	351	35	2	no
Balance-scale weight & distance	625	5	3	no
Pima Indians diabetes	768	9	2	no
Annealing	798	39	5	yes
Tic-tac-toe endgame	958	10	2	no
Pen-based recognition	10992	17	10	no
Connect-4	67557	43	3	no

Tab. 3.12: Results of C4.5 on UCI.

Name	Pruning	Leaf node number	Tree node number	Training instance		Cross-validation	
				Correct number	Accuracy	Correct number	Accuracy
Fitting	No	4	7	22	0.9167	20	0.8333
	Yes	4	7	22	0.9167	17	0.7083
Hepatitis	No	16	31	149	0.9613	125	0.8065
	Yes	11	21	143	0.9226	130	0.8387
Glass	No	30	59	206	0.9626	142	0.6636
	Yes	30	59	206	0.9626	141	0.6589
Liver	No	27	53	293	0.8493	238	0.6899
	Yes	26	51	292	0.8464	237	0.6870
Ionosphere	No	18	35	350	0.9972	321	0.9145
	Yes	18	35	350	0.9972	321	0.9145
Balance-scale	No	89	111	505	0.8080	433	0.6928
	Yes	33	41	469	0.7504	403	0.6448
Pima	No	22	43	648	0.8438	558	0.7266
	Yes	20	39	646	0.8411	567	0.7383
Annealing	No	106	151	769	0.9637	751	0.9411
	Yes	53	78	758	0.9499	739	0.9261
Tic-tac-toe	No	139	208	916	0.9562	828	0.8643
	Yes	95	142	898	0.9374	815	0.8507
Hand writing	No	215	429	10,926	0.9940	10,615	0.9657
	Yes	191	389	10,913	0.9928	10,614	0.9656
Connect-4	No	10,635	15,952	62,086	0.9190	54,174	0.8019
	Yes	4,297	6,445	59,123	0.8753	54,701	0.8097

Tab. 3.13: Results of DFDT on UCI.

Name	Pruning	Leaf node number	Tree node number	Training instance		Cross-validation Accuracy
				Correct	Accuracy	
Fitting	No	4	7	22	0.9167	20
	Yes	4	7	22	0.9167	17
Hepatitis	No	15	28	150	0.9677	127
	Yes	10	19	145	0.9355	132
Glass	No	28	53	207	0.9673	149
	Yes	28	53	207	0.9673	148
Liver	No	25	48	298	0.8638	244
	Yes	24	26	297	0.8609	243
Ionosphere	No	17	31	350	0.9972	323
	Yes	17	31	350	0.9972	323
Balance-scale	No	85	103	512	0.8192	445
	Yes	31	36	478	0.7648	416
Pima	No	20	39	655	0.8529	571
	Yes	18	35	653	0.8502	579
Annealing	No	103	143	771	0.9662	756
	Yes	50	73	762	0.9549	745
Tic-tac-toe	No	135	198	920	0.9603	836
	Yes	90	132	904	0.9436	824
Handwriting	No	211	418	10,933	0.9946	10,638
	Yes	189	370	10,921	0.9935	10,637
Connect-4	No	10,614	15,906	62,250	0.9214	54,575
	Yes	4,284	6,413	59,385	0.8790	55,087

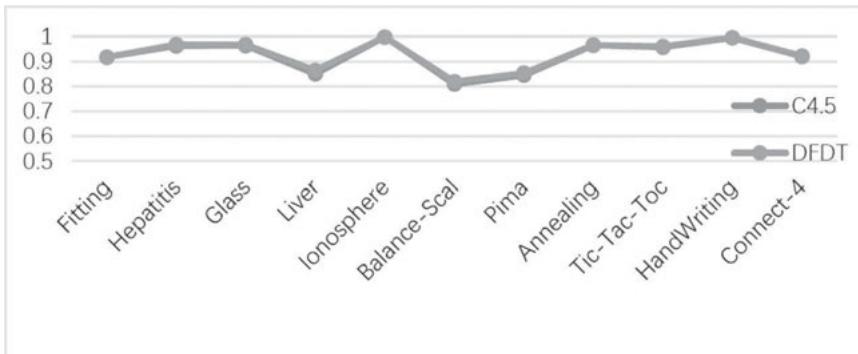


Fig. 3.8: Classification accuracy comparison of unpruned tree on training instance set.

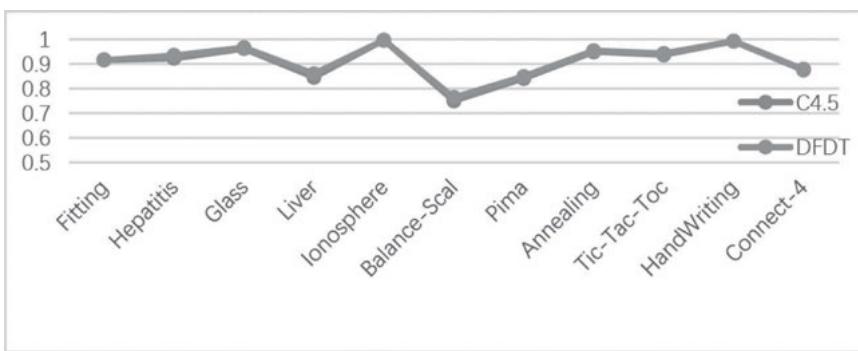


Fig. 3.9: Classification accuracy comparison of pruned tree on training instance set.

Figures 3.8–3.11 were obtained by comparing Tab. 3.11 with Tab. 3.12. Figure 3.8 compares the classification accuracy of the unpruned decision tree established by the C4.5 method and the DFDT with the training instance set. Figure 3.9 compares the classification accuracy of the pruned decision tree established by the C4.5 method and the DFDT and Fig. 3.10 shows a comparison of the accuracy of the cross-validation of the unpruned decision trees on the training instance set. Figure 3.11 compares the classification accuracy of the pruned decision trees in C4.5 and DFDT.

Analysis of experimental results:

- (1) DFDT uses the dynamic fuzzy preprocessing method, which fully considers the dynamics and fuzziness of the real problem. As a result, DFDT has better reasoning and learning abilities in the environment of dynamic fuzziness. Compared with the C4.5 method, the prediction ability is improved, and its improvement is reflected in the classification accuracy of the cross-validation of the instance set.
- (2) As the attribute selection algorithm considers the decision attribute's contribution to the whole decision-making instance set, the example sets achieve better

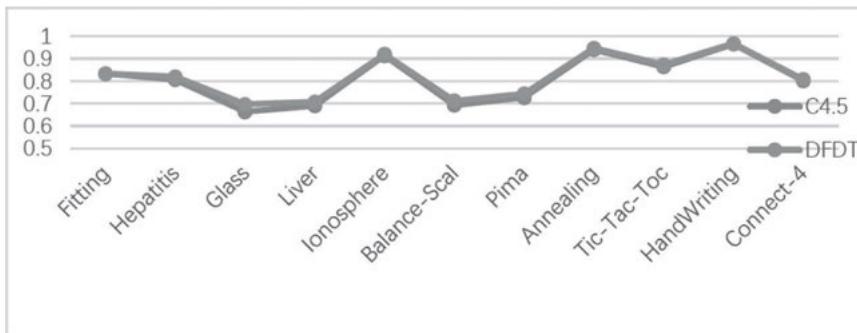


Fig. 3.10: Classification accuracy comparison of unpruned tree on cross-validation.

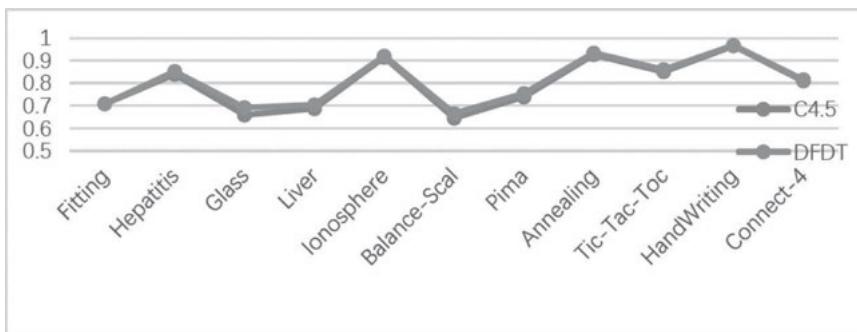


Fig. 3.11: Classification accuracy comparison of pruned tree on cross-validation.

classification with DFDT. In DFDT, the leaf node number and size of tree are less than those of the decision tree given by the C4.5 algorithm. Using the principle of Occam's razor, a small decision tree has better generalization ability. Thus, the DFDT instances in the training set have better classification accuracy, and so the DFDT has high generalization ability.

3.5.3 Comprehensibility comparisons

Real life is a dynamic and fuzzy, and dynamic data from fuzzy operations reflect this dynamic uncertainty. DFDT handles such properties, whether semantic or numerical. Handling attributes of different types is easier to understand than the discretization of numerical attributes in C4.5.

As the recognition of certain attributes, especially semantic attributes, is ambiguous in reality, the use of clear boundaries to describe attributes cannot describe the characteristics of data very well and is not consistent with people's thinking. Dynamic data blurring in DFDT not only contains the original attributes but also assigns a dynamic fuzzy number to describe the degree of each attribute in the tree. Therefore, DFDT contains more information than the C4.5 decision tree does, and this dynamic fuzzification enables the current level of development to be extrapolated to future trends. This is more in line with reality, and is easy to understand.

3.6 Summary

The decision tree method is derived from the concept learning system, which developed into ID3 and then the C4.5 algorithm, and can deal with continuous-valued attributes. In view of the dynamic fuzziness of reality, further studies are needed for the representation of dynamic fuzziness in decision tree learning and to identify whether there is a better attribute selection algorithm. Based on this, this chapter has studied the basic problem of decision tree learning based on a dynamic fuzzy lattice.

The innovations of this chapter are as follows:

- (1) We analysed the dynamic fuzziness in decision tree learning and proposed a DFDT and DFBDT based on a dynamic fuzzy lattice.
- (2) By studying the original decision tree learning algorithm, an attribute selection algorithm based on classification accuracy was proposed.
- (3) By introducing the partitioned lattice into the DFDT, the basic concept of a dynamic fuzzy partitioning grid was developed. Based on this, the relationship between the DFDT and the dynamic fuzzy binary decision tree was analysed. The relevant theorems and proofs were given. Using the dynamic fuzzy partitioning grid as a basis, the discretization of single attributes in the fuzzy decision tree and the discretization of multiple attributes was considered.
- (4) Based on the information contribution of attributes to classified information, a method of dealing with missing values in DFDTs was proposed.

References

- [1] Zhang J, Vasant H. Learning decision tree classifiers from attribute value taxonomies and partially specified data. In Proceedings of the Twentieth International Conference on Machine Learning, 2003.
- [2] Myles AJ, Brown SD. Induction of decision trees using fuzzy partitions. Journal of Chemometrics, 2003, 17: 531–536.

- [3] Olarn C, Wehenkel L. A complete fuzzy decision tree technique. *Fuzzy Sets and Systems*, 2003, 138: 221–254.
- [4] Aoki K, Watanabe T, Kudo M. Design of decision tree using class-dependent features subsets. *Systems and Computers*, 2005, 36(4): 37–47.
- [5] Kalles D, Papagelis A. Induction of decision trees in numeric domains using set-valued attributes. *Intelligence Data Analysis*, 2000, 4(4): 323–347.
- [6] Esposito F, Malerba D, Semeraro G. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1997, 19(5): 476–491.
- [7] Bolakova I. Pruning decision trees to reduce tree size. In *Proceedings of the International Conference Traditional and Innovations in Sustainable Development of Society*, 2002, 160–166.
- [8] Fournier D, Cremilleux B. A quality index for decision tree pruning. *Knowledge-Based System*, 2002, 15: 37–47.
- [9] Esposito F, Malerba D, Semeraro G, Tamma V. The effects of pruning methods on the predictive accuracy of induced decision trees. *Applied Stochastic Models in Business and Industry*, 1999, 15(4): 277–299.
- [10] Myles AJ, Feudale RN, Liu Y, Woody NA, Brown SD. An introduction to decision tree modeling. *Journal of Chemometrics*, 2004, 18: 275–285.
- [11] Zhu H. The research of the simplest decision tree production algorithm based on rough set. *Computer Engineering and Application*, 2003, 39(13): 129–131.
- [12] Jiang Y, Li ZH, Zhang Q, Liu Y. New method for constructing decision tree based on rough sets theory. *Journal of Computer Applications*, 2004, 24(8): 21–23.
- [13] Yin AD, Xie LQ, Long Y, Yang LD. Researches on dynamic algorithm of decision trees. *Computer Engineering and Application*, 2004, 40(33): 103–105.
- [14] Zhu YZ, Wu GF. A Two-phase-based construction method of decision tree and its application. *Computer Engineering*, 2004, 30(1): 82–84.
- [15] Yang HW, Zhao MH, Sun J, Wang XZ. A decision tree based on hierarchy decomposition. *Computer Engineering and Application*, 2003, 39(23): 108–110.
- [16] Wang XG, Hang SK, Zhu W, Li QY. Method of building decision trees of customer classification by using C4.5 algorithm. *Computer Engineering*, 2003, 29(14): 89–91.
- [17] Wei HN. Study on the parallelism of decision tree classification based on SPRINT. *Journal of Computer Applications*, 2003, 25(1): 39–41.
- [18] Luan LH, Ji GL. The study on decision tree classification techniques. *Computer Engineering*, 2004, 30(9): 94–96.
- [19] Han H, Wang F, Wang WY. Review of recent development in decision tree algorithm in data mining. *Application Research of Computers*, 2004, 12: 5–8.
- [20] Wen SP, Qiao SY, Chen CY, Li ZG. Decision-tree-based missing data filling and rules extraction in incomplete decision table. *Journal of Computer Applications*, 2003, 11(23): 17–19.
- [21] Zhang SY, Zhu ZY. The study of semi-structured information retrieval based on Web. *Computer Engineering and Application*, 2004, 40(13): 69–71.
- [22] Shi CQ, Yi A. Intrusion detection based on algorithm of multi-decision tree. *Computer Engineering and Design*, 2004, 25(4): 518–519.
- [23] Li CY, Yang YT. Key technology of packet filter using decision trees. *Computer Engineering*, 2004, 30(1): 45–47.
- [24] Shao FJ, Yu ZQ. Principle and algorithm of data mining. Beijing, China, China Water & Power Press, 2003.
- [25] Li ML. Research and application on theory of decision tree based on dynamic fuzzy lattice. Master's Thesis, Soochow University, Suzhou, China, 2008.

- [26] Li ML, Li FZ. Research on lattice structure of DFL and its application. *Computer Applications and Software*, 2007, 24(9): 145–150.
- [27] Li FZ, Liu GQ, She YM. An introduction to dynamic fuzzy logic. Kunming, China, Yunnan Science and Technology Press, 2005.
- [28] Cai C. Research and application on the dynamic fuzzy decision tree learning. Master's Thesis, Soochow University, Suzhou, China, 2008.
- [29] Mitchell, TM. Machine learning. NY, USA, McGraw-Hill, 1997.
- [30] Chen, ZB. Data warehouse and data mining. Beijing, China, Tsinghua University Press, 2009.
- [31] Sun CL, Zhang JF. The information gain algorithm based on attribute-value pairs. *Journal of Taiyuan University of Science and Technology*, 2005, 26(3): 199–202.
- [32] Yao YY, Yao JT. Granular computing as a basis for consistent classification problems. In *Proceedings of PAKDD Workshop on Foundations of Data Mining*, 2002.
- [33] Ghahramani Z, Jordan MI. Supervised learning from incomplete data via an em approach. *Advances in Neural Information Processing Systems*, 1993, 6: 120–127.
- [34] Li HX, Xu LD. Feature space theory – A mathematical foundation for data mining. *Knowledge-Based System*, 2001, 14: 253–257.
- [35] Shi ZZ. Knowledge discovery. Beijing, China, Tsinghua University Press, 2002.
- [36] Mao GJ, Duan LJ, Wang S, Shi Y. Principles and algorithms of data mining. Beijing, China, Tsinghua University Press, 2005.
- [37] Witten IH, Frank E. Data mining practical machine learning tools and techniques with java implementations. Beijing, China, China Machine Press, 2003.
- [38] Huan L, Farhad H, Manoranjan D. Discretization: A enabling technique. *Data Mining and Knowledge Discovery*, 2002, 6: 393–423.
- [39] Dan V, Martinez TR. An empirical comparisons of discretization methods. In *Proceedings of the 10th International Symposium on Computer and Information Science*, 1995, 443–450.
- [40] Wang LH, Wu GF. Study on the discretization lattice of information table. *Pattern Recognition and Artificial Intelligence*, 2004, 17(1): 11–15.
- [41] Wang LH, Wu Y, Wu GF. Heuristic algorithm for discretization lattice searching. *Journal of Computer Applications*, 2004, 24(8): 41–43.
- [42] Kettenring J, Lindsay B, Siegmund D. Statistics: Challenges and opportunities for the twenty-first century. *Transportation Research Record: Journal of the Transportation Research Board*, 2003, 1951(1): 44–51.
- [43] Jin YJ. Lack of data in the survey and processing (I)—Missing data and its impact. *Application of Statistics and Management*, 2001, 20(1): 59–62.
- [44] Quinlan JR. Unknown attribute values in induction. In *Proceedings of 6th International Workshop on Machine Learning*, 1989, 164–168.
- [45] Grzymala-Busse JW, Fu M. A comparison of several approaches to missing attribute values in data mining. In *Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing*, 2000, 378–385.
- [46] Liu P, Lei L, Zhang XF. A comparison study of missing value processing methods. *Computer Science*, 2004, 31(10): 155–156.
- [47] Dai WS, Xie BC. Treatment of missing values in experimental design. *Statistics and Decision*, 2004, (9): 6–7.
- [48] Huang XL. A pseudo-nearest-neighbor approach for missing data recovery on Gaussian random data sets. *Pattern Recognition Letters*, 2002, 23(13): 1613–1622.
- [49] Cai ZX, Xu GY. Artificial intelligence: Principles and applications. Beijing, China, Tsinghua University Press, 1996.
- [50] Li H, Zhu JQ, Zhu YY. ISCRMS: An Intelligent Solution of CRM in Stock. *Computer Engineering*, 2003, 29(z1): 138–140.

- [51] Breiman LI, Friedman JH, Olshen RA, et al. Classification and Regression Trees (CART)[J]. *Biometrics*, 1984, 40(3): 358.
- [52] White AP. Probabilistic induction by dynamic part generation in virtual trees[J]. *Journal of Petrology*, 1986, 45(9): 1747–1776.
- [53] Liu WZ, White AP. The importance of attribute selection measures in decision tree induction[J]. *Machine Learning*, 1994, 15(1): 25–41.
- [54] Friedman, Jerome H. “Lazy decision trees.” *Thirteenth National Conference on Artificial Intelligence AAAI Press*, 1996: 717–724.

4 Concept learning based on dynamic fuzzy sets

This chapter is divided into seven sections. Section 4.1 introduces the relationship between dynamic fuzzy sets and concept learning. In Section 4.2, we present the representation model of dynamic fuzzy concepts. In Section 4.3, we introduce the dynamic fuzzy (DF) concept learning space model. Section 4.4 presents the concept learning model based on DF lattice. In Section 4.5, we provide the concept learning model based on dynamic fuzzy decision tree (DFDT). In Section 4.6, we introduce the application examples and analysis. The summary of this chapter is in Section 4.7.

4.1 Relationship between dynamic fuzzy sets and concept learning

Concept learning is one of the main methods in the field of machine learning for generalizing and abstracting general rules or patterns from massive data sets. The key problem in concept learning is to give a set of samples, determine whether each sample belongs to a certain concept of membership, and automatically infer the general definition of the concept. In other words, the problem is to approximate the objective function of the concept from the sample set and effectively predict the concept labels of unknown concept categories in test samples to provide intelligent decision support.

Each concept can be thought of as a collection of objects or events. Categorization is the process of selecting a subset from a larger set, such as selecting birds from a collection of animals or defining the objective function from a larger set. Traditional concept learning is based on Boolean functions. However, the concept of human contact in the real world has the character of dynamic fuzzy universality. To truly reveal the laws of human understanding, scientists need to develop an actual theoretical model that provides a better approximation of the real-life objective function. Dynamic fuzzy sets are a very promising tool, so this chapter studies concept learning using dynamic fuzzy sets.

4.2 Representation model of dynamic fuzzy concepts

Dynamic fuzzy concepts are defined in an instance space. The concept or function to be learned is called the target concept, denoted as C . When learning the objective function, it is necessary to provide a set of training examples. Each sample is an instance (\vec{x}, \vec{x}) in (\vec{X}, \vec{X}) and its target concept value $C((\vec{x}, \vec{x}))$. When $C((\vec{x}, \vec{x})) \leq (\vec{\lambda}, \vec{\lambda})$, the sample is a positive example. When $C((\vec{x}, \vec{x})) \leq (\vec{\lambda}, \vec{\lambda})$, the sample is a counter example. $(\vec{\lambda}, \vec{\lambda})$ is a given dynamic fuzzy threshold. The two-tuple $<(\vec{x}, \vec{x}), C((\vec{x}, \vec{x}))>$ contains the target concept value $C((\vec{x}, \vec{x}))$ of instance (\vec{x}, \vec{x}) .

Tab. 4.1: Examples of dynamic fuzzy data background.

Example	Sky	Air Temp	Humidity	Wind	Water	Forecast
1	$(\overleftarrow{I}_{11}, \overrightarrow{I}_{11})$	$(\overleftarrow{I}_{12}, \overrightarrow{I}_{12})$	$(\overleftarrow{I}_{13}, \overrightarrow{I}_{13})$	$(\overleftarrow{I}_{14}, \overrightarrow{I}_{14})$	$(\overleftarrow{I}_{15}, \overrightarrow{I}_{15})$	$(\overleftarrow{I}_{16}, \overrightarrow{I}_{16})$
2	$(\overleftarrow{I}_{21}, \overrightarrow{I}_{21})$	$(\overleftarrow{I}_{22}, \overrightarrow{I}_{22})$	$(\overleftarrow{I}_{23}, \overrightarrow{I}_{23})$	$(\overleftarrow{I}_{24}, \overrightarrow{I}_{24})$	$(\overleftarrow{I}_{25}, \overrightarrow{I}_{25})$	$(\overleftarrow{I}_{26}, \overrightarrow{I}_{26})$
3	$(\overleftarrow{I}_{31}, \overrightarrow{I}_{31})$	$(\overleftarrow{I}_{32}, \overrightarrow{I}_{32})$	$(\overleftarrow{I}_{33}, \overrightarrow{I}_{33})$	$(\overleftarrow{I}_{34}, \overrightarrow{I}_{34})$	$(\overleftarrow{I}_{35}, \overrightarrow{I}_{35})$	$(\overleftarrow{I}_{36}, \overrightarrow{I}_{36})$

Definition 4.1 A dynamic fuzzy information system can be formalized as a dynamic fuzzy dyadic group $((\overleftarrow{O}, \overrightarrow{O}), (\overleftarrow{A}, \overrightarrow{A}))$, where $(\overleftarrow{O}, \overrightarrow{O})$ is a set of nonempty finite dynamic fuzzy instances (objects) and $(\overleftarrow{A}, \overrightarrow{A})$ is a set of finite dynamic fuzzy attributes with nonempty attributes. There is a mapping $f|_{(\overleftarrow{\alpha}, \overrightarrow{\alpha})} : (\overleftarrow{O}, \overrightarrow{O}) \rightarrow (\overleftarrow{V}, \overrightarrow{V})_{(\overleftarrow{\alpha}, \overrightarrow{\alpha})}$, where $\forall (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \in (\overleftarrow{A}, \overrightarrow{A})$, $(\overleftarrow{V}, \overrightarrow{V})_{(\overleftarrow{\alpha}, \overrightarrow{\alpha})}$ is the set of values of $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$.

Definition 4.2 It is known that the dynamic fuzzy form background is $B : (X, Y, I)$, where X is the set of dynamic fuzzy objects $X = \{\overleftarrow{x}_1, \overrightarrow{x}_1, \overleftarrow{x}_2, \overrightarrow{x}_2, \dots, \overleftarrow{x}_n, \overrightarrow{x}_n\}$, Y is the identification set of dynamic fuzzy attributes $Y = \{\overleftarrow{y}_1, \overrightarrow{y}_1, \overleftarrow{y}_2, \overrightarrow{y}_2, \dots, \overleftarrow{y}_n, \overrightarrow{y}_n\}$, $I = \phi(X \times Y)$ is defined as a dynamic fuzzy (DF) membership function on $X \times Y$, and $((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) \in I$ has the membership value $\mu((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y}))$. This value satisfies $\mu((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) \rightarrow [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]$ where $(\overleftarrow{x}, \overrightarrow{x}) \in (\overleftarrow{X}, \overrightarrow{X})$, $(\overleftarrow{y}, \overrightarrow{y}) \in (\overleftarrow{Y}, \overrightarrow{Y})$. The DF membership function can be a general dynamic fuzzy function.

There are many dynamic fuzzy background data in real life. Table 4.1 presents a dynamic fuzzy form of background data in table frame form.

Definition 4.3 In the dynamic fuzzy background, the truncation operation is defined as follows: for each $(\overleftarrow{y}, \overrightarrow{y}) \in D$, define $(\overleftarrow{\lambda}_{(\overleftarrow{y}, \overrightarrow{y})}, \overrightarrow{\lambda}_{(\overleftarrow{y}, \overrightarrow{y})})$ and $(\overleftarrow{0}, \overrightarrow{0}) \leq (\overleftarrow{\lambda}_{(\overleftarrow{y}, \overrightarrow{y})}, \overrightarrow{\lambda}_{(\overleftarrow{y}, \overrightarrow{y})}) \leq (\overleftarrow{1}, \overrightarrow{1})$. The truncation of the dynamic fuzzy background is $(\overleftarrow{\lambda}, \overrightarrow{\lambda}) K((\overleftarrow{X}, \overrightarrow{X}), (\overleftarrow{Y}, \overrightarrow{Y}), (\overleftarrow{I}_{(\overleftarrow{\lambda}, \overrightarrow{\lambda})}, \overrightarrow{I}_{(\overleftarrow{\lambda}, \overrightarrow{\lambda})}))$, where $(\overleftarrow{X}, \overrightarrow{X}), (\overleftarrow{Y}, \overrightarrow{Y})$ is the same as in Definition 4.2, namely:

$$(\overleftarrow{I}_{(\overleftarrow{\lambda}, \overrightarrow{\lambda})}, \overrightarrow{I}_{(\overleftarrow{\lambda}, \overrightarrow{\lambda})})((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) = \begin{cases} I((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) & \text{if } I((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) \geq (\overleftarrow{\lambda}_{(\overleftarrow{y}, \overrightarrow{y})}, \overrightarrow{\lambda}_{(\overleftarrow{y}, \overrightarrow{y})}) \\ (\overleftarrow{0}, \overrightarrow{0}) & \text{if } I((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) \geq (\overleftarrow{\lambda}_{(\overleftarrow{y}, \overrightarrow{y})}, \overrightarrow{\lambda}_{(\overleftarrow{y}, \overrightarrow{y})}) \end{cases} \quad (4.1)$$

Definition 4.4 In the dynamic fuzzy background B , the dynamic fuzzy concept $(\overleftarrow{C}_i, \overrightarrow{C}_i) = ((\overleftarrow{X}_i, \overrightarrow{X}_i), (\overleftarrow{Y}_i, \overrightarrow{Y}_i))$ is defined. $(\overleftarrow{X}_i, \overrightarrow{X}_i)$ and $(\overleftarrow{Y}_i, \overrightarrow{Y}_i)$ define the two maps f and g as follows:

$$\begin{aligned} \forall (\overleftarrow{X}_i, \overrightarrow{X}_i) \subseteq X: f((\overleftarrow{x}_i, \overrightarrow{x}_i)) &= \{(\overleftarrow{y}, \overrightarrow{y}) | \forall (\overleftarrow{x}_i, \overrightarrow{x}_i) \\ &\in (\overleftarrow{X}_i, \overrightarrow{X}_i), I_{(\overleftarrow{\lambda}, \overrightarrow{\lambda})}((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y}))\} \end{aligned}$$

$$\begin{aligned} \forall (\overleftarrow{Y}_i, \overrightarrow{Y}_i) \subseteq Y: g((\overleftarrow{Y}_i, \overrightarrow{Y}_i)) &= \{(\overleftarrow{y}, \overrightarrow{y}) | \forall (\overleftarrow{y}, \overrightarrow{y}) \\ &\in (\overleftarrow{Y}_i, \overrightarrow{Y}_i), I_{(\overleftarrow{\lambda}, \overrightarrow{\lambda})}((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y}))\} \end{aligned}$$

$I_{(\overleftarrow{\lambda}, \overrightarrow{\lambda})}((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y}))$ is read as the object $(\overleftarrow{x}, \overrightarrow{x})$ that has property $(\overleftarrow{y}, \overrightarrow{y})$. Functions f and g are called the DFGalois connection between the power set of X and the power set of Y .

Definition 4.5 If the two-tuple $(\overleftarrow{c}_i, \overrightarrow{c}_i) = ((\overleftarrow{X}_i, \overrightarrow{X}_i), (\overleftarrow{Y}_i, \overrightarrow{Y}_i))$ satisfies $(\overleftarrow{X}_i, \overrightarrow{X}_i) = g((\overleftarrow{Y}_i, \overrightarrow{Y}_i))$ and $(\overleftarrow{Y}_i, \overrightarrow{Y}_i) = f((\overleftarrow{X}_i, \overrightarrow{X}_i))$, then this two-tuple is called a dynamic fuzzy concept of B , where $(\overleftarrow{X}_i, \overrightarrow{X}_i)$ and $(\overleftarrow{Y}_i, \overrightarrow{Y}_i)$ are called the extension and connotation of dynamic fuzzy concept $(\overleftarrow{c}_i, \overrightarrow{c}_i)$, respectively. The set of all dynamic fuzzy concepts of B is denoted by $DFCB(B)$. The internal structure of $DFCB(B)$ is generated by generalization and specialization.

In dynamic fuzzy concept learning theory, the connotation of a concept refers to the common attribute set of all objects. The extension of a dynamic fuzzy concept refers to the largest dynamic fuzzy object set that can be determined by the connotation of a dynamic fuzzy concept. A dynamic fuzzy concept is a complete dynamic fuzzy sequence. A dynamic fuzzy concept that contains all objects is called a complete dynamic fuzzy concept. A concept that can contain all dynamic fuzzy attributes is called a dynamic fuzzy empty concept.

4.3 DF concept learning space model

In this section, we discuss the conceptual learning model and its learning algorithm based on dynamic fuzzy set (DFS) theory, which is based on the classical structure theory.

4.3.1 Order model of DF concept learning

With the increasing development of the machine learning domain, we study concept learning from different perspectives and have different model structures. Mitchell proposed a good framework for the study of concept learning theory, and its related theoretical algorithms include the FIND-S algorithm [1], list elimination algorithm, and candidate elimination algorithm [2].

The definition of dynamic fuzzy concept in an instance space $(\overleftarrow{X}, \overrightarrow{X})$: given a set of samples $(\overleftarrow{S}, \overrightarrow{S}) \subseteq (\overleftarrow{X}, \overrightarrow{X})$ and whether each sample $(\overleftarrow{x}, \overrightarrow{x}) \subseteq (\overleftarrow{X}, \overrightarrow{X})$ belongs to a concept, analysis tools can be used to summarize the definition of the target concept. This is the training process of DF concept learning. This process can also be viewed as a search process over a particular space, which is the entire space implied by the conceptual assumption. This ultimately leads to the highest degree of fit between the

hypothesis space of the search and the sample space. Under normal circumstances, the use of hypothetical space is a common structure of the natural order to conduct a quick search.

4.3.1.1 Sequence structure of DF hypothesis space

According to the “general to special” partial order structure, processing can occur in a limited hypothesis space.

Definition 4.6 For functions h_i, h_j on $(\overleftarrow{X}, \overrightarrow{X})$, we say that h_i is equal to or greater than h_j if and only if

$$(\forall (\overleftarrow{x}, \overrightarrow{x}) \in (\overleftarrow{X}, \overrightarrow{X}))[(h_i((\overleftarrow{x}, \overrightarrow{x})) = (\overleftarrow{\alpha}, \overrightarrow{\alpha})) \rightarrow (h_j((\overleftarrow{x}, \overrightarrow{x})) = (\overleftarrow{\beta}, \overrightarrow{\beta}))].$$

That is, $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \leq (\overleftarrow{\beta}, \overrightarrow{\beta})$ is recorded as $h_i \geq h_j$, where $(\overleftarrow{x}, \overrightarrow{x})$ is given as an example. We denote that h_i is strictly greater than h_j as $h_i > h_j$.

Consider the scenario depicted in Fig. 4.1: How do h_p , h_q , and h_r relate to the relationship defined by \geq_g ? By definition, h_q is more general than h_p , and so the instances that satisfied h_p also satisfied h_q , but there is no such relationship between h_p and h_r . The reason is that, although the subsets of instances satisfying the two hypotheses have intersections, neither instance subset exactly contains the other. Note here that the definitions \geq_g , $>_g$ are independent of the target concept and are relevant only to instances that satisfy these two hypotheses, regardless of the classification of instances according to the target concept. The relation \geq_g defines a partial order structure on the hypothesis space H.

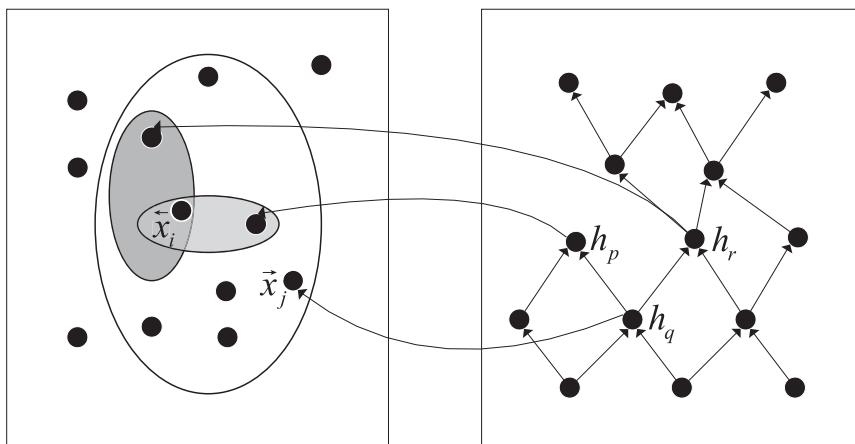


Fig. 4.1: DF concept learning space mapping diagram.

4.3.1.2 FIND-S algorithm and candidate elimination algorithm

Definition 4.7 Overlay: When a hypothesis can correctly divide a positive example, the positive case is said to be covered by the hypothesis.

FIND-S algorithm steps:

- (1) Initialize h as the most specific assumption in H .
- (2) For each positive case (\vec{x}, \vec{x}) and for each attribute constraint (\vec{a}_i, \vec{a}_i) of h , if (\vec{x}, \vec{x}) satisfies (\vec{a}_i, \vec{a}_i) , nothing is done; otherwise, replace (\vec{a}_i, \vec{a}_i) in h with the next most general constraint that satisfied by (\vec{x}, \vec{x}) .
- (3) Output hypothesis h .

Algorithm features: If the correct target concept is contained in H and the training data are correct, the FIND-S algorithm can guarantee that the output is the most specific hypothesis in H that is consistent with the positive examples. However, the convergence of the learning process is poor, and convergence to the correct objective function cannot be guaranteed. In addition, the robustness to noise is weak, and, for a number of special assumptions, the algorithm becomes powerless.

To overcome the deficiencies of the FIND-S algorithm, we extend it using the following definition.

Definition 4.8 A hypothesis h is consistent with the set D of training samples if and only if there is some $h((\vec{x}, \vec{x})) = c((\vec{x}, \vec{x}))$ for each sample $<(\vec{x}, \vec{x}), c((\vec{x}, \vec{x}))>$ in D .

$$DFConsistent(h, D) \equiv (\forall <(\vec{x}, \vec{x}), c((\vec{x}, \vec{x}))> \in D) h((\vec{x}, \vec{x})) = c((\vec{x}, \vec{x}))$$

A sample (\vec{x}, \vec{x}) satisfies hypothesis h at $h((\vec{x}, \vec{x})) \geq (\vec{\lambda}, \vec{\lambda})$, regardless of whether (\vec{x}, \vec{x}) is a positive or negative example of the DF target concept. However, whether this sample is consistent with hypothesis h is related to the concept of the target.

Definition 4.9 The general boundary G of the hypothesis space and the training data D is the set of maximal general members in D that correspond to D in H .

$$G \equiv \{g \in H \mid Consistent(g, D) \wedge (\neg \exists g' \in H) [(g' >_g g) \wedge Consistent(g', D)]\}$$

Definition 4.10 The special boundary S about the hypothesis space H and the training example D is the set of very large special members that coincide with D in H

$$S \equiv \{s \in H \mid Consistent(s, D) \wedge (\neg \exists s' \in H) [(s >_g s') \wedge Consistent(s', D)]\}$$

Theorem 4.1 Representation theorem for DF variant space: Let (\vec{X}, \vec{X}) be an arbitrary set of instances and H be a set of dynamic fuzzy hypotheses on (\vec{X}, \vec{X}) . Let $C : (\vec{X}, \vec{X}) \rightarrow [(\vec{0}, \vec{0}), (\vec{1}, \vec{1})]$ be any of the target concepts defined on (\vec{X}, \vec{X}) and

let D be any training set $\{<(\overline{x}, \overline{x}), c((\overline{x}, \overline{x}))>\}$. For all $(\overline{X}, \overline{X})$, there are some H , c , D , and well-defined S and G such that

$$VS_{H,D} = \{h \in H \mid (\exists s \in S)(\exists g \in G)(g \geq_g h \geq_g s)\}.$$

The candidate elimination algorithm proceeds as follows:

- (1) Initialize the G -set to the maximum general assumption in H ;
- (2) The S -set is initialized to the extreme special assumption in H ;
- (3) For each training example d :

If d is a positive example, remove the hypotheses that are not consistent with d ; for each hypothesis s in S that is not consistent with d , remove s from S and add all the minimal generalized h of s to S , where h is consistent with d and some member of G is more general than h ; all such hypotheses are removed from S ; this is more general than the other assumption in S .

If d is a counter example, all assumptions inconsistent with d are removed from S , g is assumed to be inconsistent with d in g , and g is removed from G . All the minimal specializations h of g are added to G , where h satisfies that h is consistent with d , and some member of S is more specialized than h . Remove all such hypotheses from G , which is more specialized than the other assumption in G .

Algorithm features: The algorithm will output a set of all hypotheses consistent with the training examples. The algorithm is biased and can be generalized to the training examples. As a set of general and special boundaries is formed, it is easy to compute the variational space formed by all the assumptions of the training examples in the entire partial order structure. If the training data are noisy, the two boundaries S and G converge to an empty variant space, so the robustness to noise is weak.

4.3.2 DF concept learning calculation model

To quantitatively analyse learning algorithms and evaluate learning performance, DF concept learning also follows commonly used computational learning theory. We must consider how many training examples are needed to successfully learn the target concept under the given learning model, how many errors will occur before learning the target concept, and so on. Two definitions of such problems are given below.

Definition 4.11 DF concept sample complexity: The number of training examples required for the learning model to converge successfully to a hypothesis.

Definition 4.12 DF concept error boundary: The number of error categorizations of the training model to the training sample before successful convergence to a hypothesis.

4.3.2.1 Dynamic fuzzy probably approximately correct (DFPAC) Learning Framework Theory

Prof. Valiant's probably approximately correct (PAC) learning theory is very important in the development of machine learning. The PAC learning framework, first proposed in 1984, states that the evaluation of machine learning should be based on "probability approximation correctness" [3], unlike traditional pattern recognition, which is based on probabilities of 1 [4].

The basic idea of this theory is that we do not need the absolutely correct learning algorithm. We use the probability language statement. The correctness of a learning algorithm can be established by the probability of the representation. This algorithm is required to satisfy polynomial complexity.

Let $(\vec{X}, \vec{X})_n$ be an instance set of learning problems of size n , let any $(\vec{x}, \vec{x}) \in (\vec{X}, \vec{X})$ be a training example, and let $(\vec{X}, \vec{X}) = (\vec{U}, \vec{U})_{n \geq 1}$ be an instance space. The task of learner L is to generate a hypothesis of the target concept so that it can correctly label each instance's category. For $n \geq 1$, $C_n \subseteq 2^{|\vec{X}, \vec{X}|_n}$ is a series of concepts on $(\vec{X}, \vec{X})_n$, $C = (\vec{U}, \vec{U})_{n \geq 1}$, and C_n is the concept space on (\vec{X}, \vec{X}) , i.e. the concept class. Suppose that the space is the set of hypotheses h that the algorithm can output.

Assume that the instances are randomly generated from X in a probability distribution. The general φ may be any distribution, unknown to learner L, which requires stability and does not change over time. The generation of the training example (\vec{x}, \vec{x}) is randomly selected according to the φ distribution, and then (\vec{x}, \vec{x}) and its target value $c((\vec{x}, \vec{x}))$ are supplied to L. Learner L considers the hypothesis space H in the learning target concept, and after observing a series of training examples of target concept c , L must output a hypothesis h from H, which is the probability estimate of c . We can evaluate the ability of learner L by evaluating the performance of the new instance extracted from (\vec{X}, \vec{X}) .

To depict the approximation degree of learner L's output hypothesis h and target concept c , the true error rate is defined as follows.

Definition 4.13 The true error rate of hypothesis h and target concept c obeys the distribution:

$$\text{error}_\varphi(h) \equiv \Pr_{(\vec{x}, \vec{x}) \in \varphi} [c((\vec{x}, \vec{x})) \neq h((\vec{x}, \vec{x}))]. \quad (4.2)$$

This states that h randomly distributes the probability that the instance of φ is mistakenly classified, where $\Pr_{(\vec{x}, \vec{x}) \in \varphi}$ is the probability distribution in the example calculated on φ .

Although we would like to know the true error rate $\text{error}_\varphi(h)$, which represents the error that learner L encounters when classifying unknown samples, it is not measurable for the size of the prediction sample space. What we can measure is the size of the sample. Therefore, we define the sample error rate.

Definition 4.14 The sample error rate of h for target concept c and sample space S is

$$\text{error}_S(h) \equiv \frac{1}{n} \sum_{(\vec{x}, \vec{x}) \in S} \sigma(c((\vec{x}, \vec{x})), h((\vec{x}, \vec{x}))), \quad (4.3)$$

where n is the length of the sample space S , $\sigma(c((\vec{x}, \vec{x})), h((\vec{x}, \vec{x})))$ is 1 if $c((\vec{x}, \vec{x})) \neq h((\vec{x}, \vec{x}))$ and 0 otherwise.

Definition 4.15 Consider the concept class C defined on the instance set (\vec{X}, \vec{X}) with length n . Learner L uses the hypothesis space H . For all $c \in C$, (\vec{X}, \vec{X}) distributions φ , $0 < \varepsilon < 1/2$ and $0 < \delta < 1/2$, learner L will output the hypothesis $h \in H$ with a probability of at least $1 - \delta$ such that $\text{error}_\varphi(h) \leq \varepsilon$. Concept C is then a DFPAC for L using H . The polynomial functions used are $1/\varepsilon$, $1/\delta$, n , and the size (c) .

4.3.2.2 Sample complexity of DF assumed space

According to the definition, the learning ability of DFPAC is largely determined by the number of training samples. Therefore, the problem of the sample complexity of DF learning arises.

(1) For a limited hypothesis space situation

First, we consider the sample complexity of a consistent learner. If and only if it is possible to output the hypothesis that best fits the training examples, this class of learners are said to be consistent.

Definition 4.16 Consider a set of training examples D of hypothesis space H , target concept c , and instance distributions φ . When each hypothesis h in $VS_{H,D}$ is less than ε for c and φ errors, the variant space is called c and the ε – space of φ is exhaustive.

The detailed deformation space of ε – indicates that the true error rate of all hypotheses [i.e. $\text{error}_S(h) = 0$] consistent with the training example is just less than ε . From the perspective of trainer L , these assumptions fit the training data equally well, and they all have zero training error rates. Only by knowing the exact concept of the observation can we determine whether the deformation of space ε – is detailed.

Theorem 4.2 ε – Extension of the Variant Space: If the hypothesis space H is finite, and D consists of $m \geq 1$ independent, randomly selected training examples of concept c , then for any $0 \leq \varepsilon \leq 1$, the variant space $VS_{H,D}$ is not ε – exhaustive (with respect to c). The probability is less than or equal to $|H|e^{-\varepsilon m}$.

We obtain the upper bound of the probability that the variant space is not ε –. On the other hand, we define the probability that m training samples fail to

reject all the “bad” assumptions (i.e. the assumption that the error rate is greater than ϵ) for any learner L using hypothesis space H . This theorem can be used to determine the number of training examples required to reduce the probability of this “undeleted” to a certain limit δ . This satisfies $|H|e^{-\epsilon m} < \delta$, from which we can conclude

$$m \geq \frac{1}{\epsilon}(\ln|H| + \ln(1/\delta)). \quad (4.4)$$

It can be seen that the number of training examples increases linearly with $1/\epsilon$ and increases logarithmically with $1/\delta$, but also increases logarithmically with the size of the hypothetical space H .

When the learner is not consistent, for example, for a given sample D , $\text{error}_D(h) \neq 0$, only the hypothetical h_{best} with the smallest training error rate can be considered at this time, thus ensuring that $\text{error}_{\S}(h_{best})$ will not exceed the number of samples of $\text{error}_{\S}(h_{best}) + \epsilon$ with a higher probability. When $\text{error}_D(h)$ contains m random samples on the measurement of D , this probability can be given by the Hoeffding bound:

$$P(\exists h \in H, s.t. \text{error}_{\wp}(h_{best}) > \text{error}_D(h_{best}) + \epsilon) \leq |H|e^{-2\epsilon^2 m}.$$

The boundary of the training samples is

$$m \geq \frac{1}{2\epsilon^2}(\ln|H| + \ln(1/\delta)). \quad (4.5)$$

(2) For infinite hypothesis space

For the infinite hypothesis space (concept class), we need to use the concept of VC-dimensionality to describe the complexity of the hypothesis space, which reflects the ability to scatter the (\vec{X}, \vec{X}) instance set size.

Definition 4.17 The set of instances S is scrambled by the hypothesis space H if and only if, for each partition of S , there exists an assumption in H that is consistent with this partition.

Assume that the ability of the spatial H to break up the set of instances is a measure of its ability to define the target concept on these instance sets. The larger the subset in the scattered instance space, the stronger the representation ability of H .

Definition 4.18 The VC dimension of a hypothetical space H , or $\text{VC}(H)$, defined on the instance space (\vec{X}, \vec{X}) is the size of the largest finite subset of (\vec{X}, \vec{X}) that can

be broken up by H . If any finite subset of $(\overleftarrow{X}, \overrightarrow{X})$ can be broken up by H , then $VC(H) \equiv \infty$.

Using $VC(H)$ as a measure of H complexity, we can derive a new boundary, that is,

$$m \geq \frac{1}{\varepsilon} (4 \log 2(2/\delta) + 8VC(H) \log 2(13/\varepsilon)).$$

The number of training examples m grows with the logarithm of $1/\delta$, and also linearly in the logarithm of $1/\varepsilon$ and in $1/\varepsilon$.

Theorem 4.3 Lower Bound of Sample Complexity: Consider any concept class C with $VC(C) > 2$, any learner L , and any $0 < \varepsilon < 1/8$, $0 < \delta < 1/100$. There is a target concept for the distribution φ and C , and when L observes that the number of samples is less than:

$$\max \left[\frac{1}{\varepsilon} \log(1/\delta), \frac{VC(C) - 1}{32\varepsilon} \right]. \quad (4.6)$$

L will output hypothesis h with a probability of at least δ and $error_{\varphi}(h) > \varepsilon$.

This theorem states that if there are few training examples, then no learner L can learn every target concept in C from any nontrivial PAC model.

4.3.2.3 DF error bound model

1) Learning algorithm error rate model

As there may be noise in the real data, it is inevitable that there may be errors in the learning process. Assume that the training sample S is randomly selected in the set of instances φ , and the true error rate $error_{\varphi}(h)$ is estimated from the statistical sample error rate $error_S(h)$. Because the test h on the sample set S obtained by random sampling to estimate $error_{\varphi}(h)$ is the same as the probability of estimating the occurrence of the positive or negative sides from n random samples of a tossed coin, it can be described by the binomial distribution $B(n, p)$.

Assuming that the number of instances of a set S is n , let r be the number of instances in S that are misclassified by h , and let p be the probability that one instance is randomly divided into other concepts from φ . Then, the moment estimator of p is $error_S(h) = r/n$, where $error_{\varphi}(h) = p$. It can be proved that $error_S(h) = r/n$ is an unbiased estimator. The binomial distribution property gives $error_S(h)$ a standard deviation of $\sigma_{error_S(h)} = \sqrt{p(1-p)/n}$, and if r/n is used to approximate p , we get

$$\sigma_{error_S(h)} \approx \sqrt{error_S(H)(1 - error_S(H))/n}. \quad (4.7)$$

2) Optimal error bound model

To define the error rate for any learning algorithm and any target concept learning process, we introduce the concept of optimal error bounds. Let $Magin_{Al}(C) \equiv \max_{c \in C} Magin_{Al}(c)$ denote the maximum value of errors in all possible training examples to ensure that concept c is learned.

Definition 4.19 Let C be any nonempty concept class. (C) is the minimum value of $Magin_{Al}(C)$ for all possible learning algorithms Al .

$$Opt(C) \equiv \min_{Al \in \text{Learning Algorithm}} Magin_{Al}(C) \quad (4.8)$$

Littlestone [5] proved that, for any concept C , there exists the following relation between the VC error of C and the optimal error bound of C :

$$VC(C) \leq Opt(C) \leq \log 2^{|C|}. \quad (4.9)$$

Therefore, it is very interesting to study the error boundary of DF concept learning. This is a hot topic and an integral part of DF concept learning.

4.3.3 Dimensionality reduction model of DF instances

For common actual dynamic fuzzy datasets (DFD), human factors such as data recording errors, missing records, too specialized data, and so on, mean it is necessary to pre-process the corresponding data set to improve the quality of data before applying a DF concept learning algorithm.

Because DF concept learning is a supervised learning process, the concept category of each instance in the sample set has been marked, which can reduce the number of cluster centres. Therefore, a clustering method is more feasible. Clustering methods can detect data that do not belong to the same concept of the class instance, while highly similar data can be attributed to the same concept class. Common clustering methods include density-based clustering, segmentation clustering, and hierarchical clustering. However, the Fuzzy C-means (FCM) algorithm theory in nonclassical applications is far away perfect. The following describes the DF data using an FCM algorithm.

- (1) According to the labelled DF concept class, specify the number of clusters $c (2 \leq c \leq n)$, where n is the number of DF data, set the iteration termination threshold $(\xi, \bar{\xi})$, initialize the cluster center point V^0 , and let the iteration counter $b = 0$.
- (2) The partition matrix $U^b = [(\bar{u}_{ij}, \vec{u}_{ij})]$ is calculated as

$$(\bar{u}_{ij}, \vec{u}_{ij}) = \left[\sum_{k=1}^c \left(\frac{\|(\bar{x}_j, \vec{x}_j) - (\bar{v}_i, \vec{v}_i)\|^2}{\|(\bar{x}_j, \vec{x}_j) - (\bar{v}_k, \vec{v}_k)\|^2} \right)^{1/(m-1)} \right]^{-1}, \quad 1 \leq i \leq c, 1 \leq j \leq n. \quad (4.10)$$

(3) Update the cluster centre according to

$$(\overleftarrow{v}_i, \overrightarrow{v}_i) = \frac{\sum_{j=1}^n (\overleftarrow{u}_{ij}, \overrightarrow{u}_{ij})^m (\overleftarrow{x}_j, \overrightarrow{x}_j)}{\sum_{j=1}^n (\overleftarrow{u}_{ij}, \overrightarrow{u}_{ij})^m}, 1 \leq i \leq c. \quad (4.11)$$

(4) If $\|V^b - V^{b+1}\| \leq (\overleftarrow{\xi}, \overrightarrow{\xi})$, the algorithm terminates and outputs the partition matrix U and the cluster centre V ; otherwise, $b = b + 1$ and the algorithm returns to step (3).

Through the optimization of the initial data set selection, the quality of the data is improved, which enhances the accuracy of data analysis.

4.3.4 Dimensionality reduction model of DF attribute space

DF data attribute space reduction is the main method for preprocessing massive datasets. Common methods are principal component analysis (PCA), linear discriminant analysis (LDA), maximum margin criterion, and so on. PCA is one of the most widely used methods for feature selection in large-scale data sets. The main idea is to extract the eigenvectors with expressive force from the high-dimensional data and remove the irrelevant and redundant features so as to improve the accuracy of the learning problem.

4.3.4.1 Dynamic fuzzy principal component analysis (DFPCA) dimensionality reduction model

DFPCA uses the PCA principle to reduce the dimension of the data. The basic idea is to analyse, process, and extract the covariance matrix of dynamic fuzzy variables according to the principle of variance maximization. The original DFD matrix is represented by a new set of linearly independent and orthogonal vectors. The variance is sorted by size, and the largest variance and eigenvalue are taken as the first principal component, the second-largest form the second principal component, and so on. The principal components are orthogonal to each other. The specific algorithm is described as follows:

Consider a known function $\{(\overleftarrow{X}, \overrightarrow{X})\}$ whose probability distribution function is given by n -dimensional random dynamic fuzzy variables, namely, $\{(\overleftarrow{X}, \overrightarrow{X})\} = [(\overleftarrow{x}_1, \overrightarrow{x}_1), (\overleftarrow{x}_2, \overrightarrow{x}_2), \dots, (\overleftarrow{x}_n, \overrightarrow{x}_n)]$, with mean $E((\overleftarrow{x}, \overrightarrow{x})) = (\overleftarrow{0}, \overrightarrow{0})$.

The DF covariance matrix $(\overleftarrow{C}, \overrightarrow{C}) = E[((\overleftarrow{X}, \overrightarrow{X}) - E[(\overleftarrow{X}, \overrightarrow{X})])((\overleftarrow{X}, \overrightarrow{X}) - E[(\overleftarrow{X}, \overrightarrow{X})])^T]$. From $E[(\overleftarrow{X}, \overrightarrow{X})] = (\overleftarrow{0}, \overrightarrow{0})$, this can be simplified as

$$(\overleftarrow{C}, \overrightarrow{C}) = E[(\overleftarrow{X}, \overrightarrow{X})(\overleftarrow{X}, \overrightarrow{X})^T]. \quad (4.12)$$

The characteristic value of $(\overleftarrow{C}, \overrightarrow{C})$ is $(\overleftarrow{\lambda}_1, \overrightarrow{\lambda}_1), (\overleftarrow{\lambda}_2, \overrightarrow{\lambda}_2), \dots, (\overleftarrow{\lambda}_n, \overrightarrow{\lambda}_n)$, and the characteristic value is normalized to obtain $(\overleftarrow{y}_1, \overrightarrow{y}_1), (\overleftarrow{y}_2, \overrightarrow{y}_2), \dots, (\overleftarrow{y}_n, \overrightarrow{y}_n)$ as

the input characteristic satisfying the condition. Thus, the projection of the eigenvectors is

$$(\overleftarrow{M}, \overrightarrow{M}) = (Y_i, Y_i)^T (\overleftarrow{X}, \overrightarrow{X}) (i = 1, 2, \dots, n). \quad (4.13)$$

This can be expressed in matrix form as the first principal component of the eigenvector:

$$(\overleftarrow{M}, \overrightarrow{M}) = (Y, Y)^T (\overleftarrow{X}, \overrightarrow{X}), \quad (4.14)$$

where $(\overleftarrow{M}, \overrightarrow{M})$ can be decomposed into $(\overleftarrow{M}, \overrightarrow{M}) = \{(\overleftarrow{M}_1 \overrightarrow{M}_1), (\overleftarrow{M}_2 \overrightarrow{M}_2), \dots, (\overleftarrow{M}_n \overrightarrow{M}_n)\}$ satisfying $(\overleftarrow{M}, \overrightarrow{M})(\overleftarrow{M}, \overrightarrow{M})^T = (\overleftarrow{I}, \overrightarrow{I})$ [$(\overleftarrow{I}, \overrightarrow{I})$ is the dynamic fuzzy unit matrix]. The arbitrary vector $(\overleftarrow{M}_i, \overrightarrow{M}_i)$ in the feature space is the i th principal component of the input n -dimensional vector $(\overleftarrow{X}, \overrightarrow{X})$.

The autocovariance matrix of the feature vector $(\overleftarrow{M}, \overrightarrow{M})$ is

$$\begin{aligned} (\overleftarrow{C}, \overrightarrow{C})_{(\overleftarrow{M}, \overrightarrow{M})} &= E((\overleftarrow{M}, \overrightarrow{M})(\overleftarrow{M}, \overrightarrow{M})^T) \\ &= E((\overleftarrow{Y}, \overrightarrow{Y})^T (\overleftarrow{X}, \overrightarrow{X}) (\overrightarrow{X}, \overrightarrow{X})^T (\overleftarrow{Y}, \overrightarrow{Y})) \\ &= (\overleftarrow{Y}, \overrightarrow{Y})^T E((\overrightarrow{X}, \overrightarrow{X}) (\overrightarrow{X}, \overrightarrow{X})^T) (\overleftarrow{Y}, \overrightarrow{Y}). \end{aligned} \quad (4.15)$$

At this time, $(\overleftarrow{Y}, \overrightarrow{Y})$ becomes the $(\overrightarrow{X}, \overrightarrow{X})$ eigenvector matrix, and then according to the eigenvalue of the principal component eigenvalues corresponding to $(\overleftarrow{\lambda}_1, \overrightarrow{\lambda}_1) \geq (\overleftarrow{\lambda}_2, \overrightarrow{\lambda}_2) \geq \dots \geq (\overleftarrow{\lambda}_n, \overrightarrow{\lambda}_n)$, the $(\overleftarrow{Y}, \overrightarrow{Y})$ cutoff dimension is reduced. If the EE value corresponding to the eigenvector is larger, bigger values of $(\overleftarrow{\lambda}, \overrightarrow{\lambda})$ correspond to smaller values in the original DFD set.

For the original data, the minimum error criterion should be satisfied in the process of dimension reduction. Suppose the first a feature values are considered. The estimated value of the corresponding reconstruction is

$$\overline{(\overrightarrow{X}, \overrightarrow{X})} = \sum_{i=1}^n (\overleftarrow{Y}_i, \overrightarrow{Y}_i) (\overleftarrow{M}_i, \overrightarrow{M}_i). \quad (4.16)$$

The average variance is

$$(e_a, e_a) = E[((\overrightarrow{X}, \overrightarrow{X}) - \overline{(\overrightarrow{X}, \overrightarrow{X})})^2] = \sum_{i=a+1}^n (\overleftarrow{\lambda}_i, \overrightarrow{\lambda}_i). \quad (4.17)$$

When the a -eigenvalues of the $(\overrightarrow{X}, \overrightarrow{X})$ autocovariance matrix become larger, the mean square error becomes smaller. Thus, it is easy to find principal component whereby the former achieves a reduced dimension.

4.3.4.2 Dynamic fuzzy linear discriminant analysis (DFLDA) dimensionality reduction model

DFLDA is based on LDA. The basic idea is to find some vectors that can best distinguish the classes in the DF concept space. That is, we seek to maximize the class-space metric and minimize the intra-class distance measure. The high-dimensional data can be classified into subspaces by mapping them into low-dimensional sub-spaces. Suppose that, given a dynamic fuzzy training set S , the total number of samples is N , the total number of concept classes is C , and the total number of samples in each category is N_i , $i = 1, 2, \dots, C$, $\sum_{i=1}^C N_i = N$. The dynamic fuzzy dispersion matrix in the DF concept class is

$$S_W = \frac{1}{N} \sum_{i=1}^C \sum_{j=1}^N \left((\overline{\overline{x}}_{ij}, \overline{x}_{ij}) - (\overline{\overline{x}}_i, \overline{x}_i) \right) \left((\overline{\overline{x}}_{ij}, \overline{x}_{ij}) - (\overline{\overline{x}}_i, \overline{x}_i) \right)^T. \quad (4.18)$$

The inter-class dispersion matrix for the DF concept is

$$S_B = \frac{1}{N} \sum_{i=1}^C N_i \left((\overline{\overline{x}}_i, \overline{x}_i) - (\overline{\overline{x}}, \overline{x}) \right) \left((\overline{\overline{x}}_i, \overline{x}_i) - (\overline{\overline{x}}, \overline{x}) \right)^T, \quad (4.19)$$

where $(\overline{\overline{x}}, \overline{x})$ is the dynamic fuzzy mean vector of all attributes, $(\overline{\overline{x}}_i, \overline{x}_i)$ is the dynamic fuzzy mean vector of the i th attribute, and $(\overline{\overline{x}}_{ij}, \overline{x}_{ij})$ is the j th sample of the i th dynamic fuzzy attribute.

Therefore, according to Fisher's linear discriminant method, get the discriminant function:

$$J(W) = \frac{|W^T S_B W|}{|W^T S_W W|}. \quad (4.20)$$

If S_W is a nonsingular matrix, then an orthogonal matrix can be obtained that maximizes the ratio of the matrix determinants of the interdisciplinary dispersion matrix to those of the intra-class dispersal distribution matrix:

$$W_{opt}^* = [w_1, w_2, \dots, w_p] = \arg \max_w \frac{|W^T S_B W|}{|W^T S_W W|}, \quad (4.21)$$

where $\{w_i | i = 1, 2, \dots, p\}$ is the generalized eigenvector of (4.20), corresponding to the p largest maximum eigenvalues $\{(\overline{\lambda}_i, \overline{\lambda}_i) | i = 1, 2, \dots, p\}$ of (4.20), which is

$$S_B w_i = \lambda_i S_W w_i. \quad (4.22)$$

As the rank of the inter-class scatter matrix S_B of the DF concept is less than or equal to $C - 1$, $p \leq C - 1$. Namely, there are at most $C - 1$ generalized nonzero eigenvalues. W_{opt}^* is the best projection transformation matrix.

According to the eigenvector corresponding to the largest eigenvalue (the number of elements in the eigenvector is the number of attributes), all the attributes in the eigenvectors with values above a certain threshold $(\overline{\delta}, \overline{\delta})((\overline{0}, \overline{0}) < (\overline{\delta}, \overline{\delta}) < (\overline{1}, \overline{1}))$ are selected as attributes of the new training samples.

4.4 Concept learning model based on DF lattice

The formal concept analysis theory was first applied to the concept of discovery, display, and sorting. One of the main data structures for formal concept analysis is the concept lattice. Each node in the concept lattice is a concept. An extension refers to the instance space (object) covered by the concept. The connotation is the common characteristic (attribute) of the covered instance. The structure reflects the relationship between objects and attributes and a complete concept hierarchy of conceptual generalizations and specialization relations. Burusco and others [6] first applied fuzzy theory to formal concept analysis, and constructed a lattice structure based on the L-fuzzy concept set. As a result of the interaction between the theory of order and lattice theory, the combination of the concept lattice with DFL has a very important practical value. In this section, we introduce four classic construction algorithms; then, we focus on the hierarchical DF concept lattice construction algorithm before discussing the DF concept lattice reduction technology. Finally, we introduce a DF containing rule generation strategy and verify the effectiveness of the algorithm by means of an example and an experiment.

4.4.1 Construction of classical concept lattice

At present, concept lattice construction algorithms are divided into two kinds: batch algorithms and increment algorithms. Batch-based construction algorithms are mainly applied to relatively small instance sets in the data space, e.g. the Bordat, Chein, Ganter, and Nourine algorithms. Incremental construction algorithms are applied to instance spaces where the data are constantly updated. Representative algorithms include those of Godin, Capinetto, Ho, and so on.

4.4.1.1 Batch algorithms

Batch processing algorithms to generate a concept lattice generally have two steps: First, the grid nodes are generated; second, the grid nodes are used to establish a direct precursor/direct successor relationship. According to the task order, there are two main generation methods, namely, the task partitioning model (as developed by Chein, Ganter and other algorithms) and the task cross-building model (such as Bordat's algorithm).

The Chein algorithm [7] constructs the concept lattice from bottom to top. The algorithm iteratively generates a collection of all concepts starting with the most

ubiquitous, such as the concept of an object. The algorithm starts at the L1 level and contains a set of pairs $(\{x\}, f(x))$ for all objects x in all object sets. It constructs a new set of pairs of $L_k + 1$ layers by combining any pair of L_k layers. The merging process considers the two pairs of intersection sets for the L_k layer and tests whether the intersection already exists, depending on the result. If the upper layer appears, it indicates that the upper layer is not completely correct and is marked as such, so that the end of this layer operation will be deleted. The algorithm is simple and clear, but the practical application will produce a large number of redundant pairs in the same layer and in lower layers. In the algorithm, the same concept will be generated at different times by different concepts. This is the main drawback of this algorithm. The algorithm itself does not generate a Hasse graph, but this layer-by-layer grid-node generation method is useful for identifying links between grid nodes.

In the Bordat algorithm [8], the concept lattice L is constructed from the topmost node. For each node, the child nodes are generated and the link between child nodes and parent node is completed. The process is then called recursively for the resulting child nodes. The basic idea of the algorithm is that if the current node is (O_1, D_1) , where O is the object set and D is the attribute set, we find the attribute subset $D_2 = D - D_1$, such that D_2 can maintain the complete binary property on O_1 . This is the largest expansion and constitutes the current node of the subnode's content. The algorithm is also very simple, intuitive, and easy to run in parallel. The disadvantage is that the same node is generated multiple times. In practical applications, each node is duplicated to generate its parent node several times. However, this is one of the most effective algorithms.

The Ganter algorithm [9] uses the eigenvectors representing X sets to enumerate X sets of cells. The length of each vector is the cardinality of the attribute set. The corresponding bit is set to 1 if an attribute value is present in the vector, and to 0 otherwise. The feature vectors are sorted in lexicographic order. Given a vector $X = \{x_1, x_2, \dots, x_m\}$, it finds the next X vector. The way to do this is to set the property bit in order and test whether it is a complete pair. The algorithm is initialized with $\{0, 0, \dots, 0\}$. The vectors produced in this way are ordered. The list is sorted according to the containing topology of the X set. The algorithm does not generate Hasse graphs.

In contrast to the other algorithms, the Nourine [10] algorithm generates a concept lattice using the concept of epitaxial complement as the basic operating units. The algorithm is divided into two parts. First, the extension set of all the individual attributes forms a base B , and the set of clusters generated in B is $F = \left\{ \bigcup_{D \subset K} U \mid K \subset B \right\}$. The purpose of this step is to generate F from the base as a lexical-order tree representation. Each path of the tree represents an element in F . Operations with the same prefix share the same partial path. The root is an empty set. The overlay of F is then calculated. The covering relation between two elements is defined in the algorithm, and the condition whereby the elements in the lexical-order tree satisfy the covering

relation is derived to determine the corresponding algorithm. It has been shown that the algorithm is less complex than those of Bordat and Ganter et al.

4.4.1.2 Incremental algorithms

The basic idea of increment algorithms is to present the object to be inserted and place all the concepts in the intersection of the lattice according to the results of different treatments. It is assumed that the first i objects in the formal context have generated subsets of the concept nodes and sublattices of the concept lattice. This step is repeated until the final cell structure is generated. Obviously, it is easy to apply this strategy to attributes, which can be generated incrementally based on attributes rather than objects. Typical algorithms include Godin's algorithm [11], Capinetto's algorithm [12], and Ho's algorithm [13]. The property-based incremental construction algorithm is similar to the basic idea of Godin's algorithm. The difference is that the algorithm adds to the concept lattice one by one.

Given the concept lattice L corresponding to the background $B : (X, Y, I)$ and the new object X^* , the concept lattice L^* corresponding to the formal context $B^* = (X \cup \{X^*\}, Y, I)$ is given. There are three main problems in the process of generating the concept lattice: (1) generation of all new nodes, (2) avoiding the repetition of existing lattice nodes, and (3) updating the edges. Therefore, for each node in the original concept lattice, different types are defined according to the relation between them and the connotation description of the new object.

Definition 4.20 If a lattice node C satisfies $Intent(C) \subseteq f(\{x^*\})$, C is called a replacement node. Obviously, if C is a replacement node, then C should be replaced by $(Extent(C) \cup \{X^*\}, Intent(C))$ in L^* .

Definition 4.21 If the given node $C_i = (O_i, A_i)$ satisfies the following: (1) the intersection of connotations $(f(\{x^*\}))$ between the concept node and new node is not equal to the connotation of any node in the lattice; (2) the intersection of connotations $(f(\{x^*\}))$ between the concept node and new node is also not equal to $Intent(C_j) \cap f(\{x^*\})$, where C_j is any node that satisfies $C_j > C$ in L ; then C_i is called an extended subgrid node.

Obviously, replaceable nodes cannot be generated with the same node because $D_i \cap f(\{x^*\}) = D_i$. If a node in lattice L is neither a replacement node nor a generatable node, it is called an invariant node.

For any node C in L^* : If there is no node C' in L satisfying $Intent(C') = Intent(C)$, C is called a new node; otherwise, it is called an inheritance node. If C' is an extended subgrid node, node $(Extent(C') \cup \{X^*\}, Intent(C') \cap \{X^*\})$ is obviously a new node in L^* . This is called a newborn node generated by C' .

Theorem 4.4 Given the concept lattice L and the new object x^* , the node C' in L is a generating subgrid node if and only if (1) $\neg(Intent(C_1) \subseteq f(\{X^*\}))$ and (2) $\neg(\exists C_2 \in Before(C_1)(Intent(C_2) = Intent(C_1)))$, where $Before(C_1)$ denotes the set of all lattice

nodes that are accessed before C_1 and all the newborn lattices generated before C_1 . The order of access here is given by the L grid nodes in accordance with the meaning of small to large access to the order. This ensures that the C_1 predecessor node is accessed first to satisfy the definition of generating the child nodes.

Theorem 4.5 If C_1 is a subgrid node and its corresponding newborn node is C_{new} , then L satisfies the concept of $Intent(C_2) \supseteq Intent(C_{new})$, and C_2 must satisfy $Intent(C_2) \supseteq Intent(C_1)$.

It can be seen from Theorem 4.5 that, for a new node, none of the old nodes can be its subnode except for its subgrid node.

Theorem 4.6 For two new nodes C_{new1} and C_{new2} , if $Intent(C_{new1}) \subset Intent(C_{new2})$, then C_{new1} must be generated before C_{new2} and cannot become its subgrid node.

4.4.2 Constructing lattice algorithm based on DFS

4.4.2.1 Delamination of DF concept lattice

In the concept lattice of DF, the hierarchical structure of a concept is defined by attributes. Therefore, the hierarchical concept lattice is defined by the length of the attribute set. We now introduce the DF concept lattice.

Definition 4.22 In the concept lattice of DF, if the longest distance between the concept of $((\overleftarrow{O}, \overrightarrow{O}), (\overleftarrow{A}, \overrightarrow{A}))$ and the largest element in the lattice is N, then we say that DF concept $((\overleftarrow{O}, \overrightarrow{O}), (\overleftarrow{A}, \overrightarrow{A}))$ is in the Nth layer of the DF concept lattice.

Definition 4.23 In the DF concept lattice, we assume that $((\overleftarrow{O}_i, \overrightarrow{O}_i), (\overleftarrow{A}_i, \overrightarrow{A}_i)) \geq ((\overleftarrow{O}_j, \overrightarrow{O}_j), (\overleftarrow{A}_j, \overrightarrow{A}_j))$, and if the longest chain length of $((\overleftarrow{O}_i, \overrightarrow{O}_i), (\overleftarrow{A}_i, \overrightarrow{A}_i))$ to $((\overleftarrow{O}_j, \overrightarrow{O}_j), (\overleftarrow{A}_j, \overrightarrow{A}_j))$ is $l(l \in \mathbb{Z}^+)$, then $((\overleftarrow{O}_i, \overrightarrow{O}_i), (\overleftarrow{A}_i, \overrightarrow{A}_i))$ is the l -layer ancestor of $((\overleftarrow{O}_j, \overrightarrow{O}_j), (\overleftarrow{A}_j, \overrightarrow{A}_j))$.

Obviously, there is no overlay between two nodes on the same layer and there is no connection line in the DF concept lattice. The high-level nodes at different layers are covered by at least one lower node.

Definition 4.24 If concepts $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ and $(\overleftarrow{C}_j, \overrightarrow{C}_j)$ satisfy $(\overleftarrow{C}_i, \overrightarrow{C}_i) \leq (\overleftarrow{C}_j, \overrightarrow{C}_j)$, then $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ is a subconcept of $(\overleftarrow{C}_j, \overrightarrow{C}_j)$, and $(\overleftarrow{C}_j, \overrightarrow{C}_j)$ is a superconcept of $(\overleftarrow{C}_i, \overrightarrow{C}_i)$.

If there is no concept $(\overleftarrow{C}_k, \overrightarrow{C}_k)$ in the concept lattice such that $(\overleftarrow{C}_i, \overrightarrow{C}_i) \leq (\overleftarrow{C}_k, \overrightarrow{C}_k) \leq (\overleftarrow{C}_j, \overrightarrow{C}_j)$ holds, then we call $(\overleftarrow{C}_j, \overrightarrow{C}_j)$ the parent node of $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ and $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ is the child node of $(\overleftarrow{C}_j, \overrightarrow{C}_j)$. There is an edge between $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ and $(\overleftarrow{C}_j, \overrightarrow{C}_j)$: $((\overleftarrow{C}_i, \overrightarrow{C}_i), (\overleftarrow{C}_j, \overrightarrow{C}_j))$.

Definition 4.25 Except for the minimum element, the DF concept lattice can be divided into $T_{\max} = \max_{(\overline{x}, \overline{x}) \in (\overline{X}, \overline{X})} \{ |(\overline{x}, \overline{x})| \}$ layers, among which $|(\overline{x}, \overline{x})|$ is the number of attributes of the object.

4.4.2.2 Algorithm for constructing the DF concept lattice layer by layer

Construction of the DF lattice algorithm from top to bottom (DFCL_Layer) is as follows:

All of the temporary children of the second layer are generated iteratively according to this algorithm.

Processing operations on temporary subnodes: The $(N + 1)$ th temporary subgrid node is $((\overline{O}_{N+1,1}, \overline{O}_{N+1,1}), (\overline{A}_{N+1,1}, \overline{A}_{N+1,1}))$, $((\overline{O}_{N+1,2}, \overline{O}_{N+1,2}), (\overline{A}_{N+1,2}, \overline{A}_{N+1,2}))$, \dots , $((\overline{O}_{N+1,q}, \overline{O}_{N+1,q}), (\overline{A}_{N+1,q}, \overline{A}_{N+1,q}))$. Sort these according to the number of objects covered by the node, i.e. \dots , $|(\overline{O}_{N+1,i}, \overline{O}_{N+1,i})| \geq |(\overline{O}_{N+1,j}, \overline{O}_{N+1,j})| \dots$, where $1 \leq i, j \leq q$, and, $i \neq j$. Finally, the neighbouring temporary sublattice nodes are examined in turn. If $(\overline{O}_{N+1,i}, \overline{O}_{N+1,i}) \supseteq (\overline{O}_{N+1,j}, \overline{O}_{N+1,j})$, then delete the temporary subnode $(\overline{O}_{N+1,j}, \overline{O}_{N+1,j}), (\overline{A}_{N+1,j}, \overline{A}_{N+1,j})$ and judge the sorted temporary subgrid nodes until all subgrid nodes of the $(N + 1)$ th layer have been determined.

The algorithm is simple and clear and is fully dependent on the dynamic fuzzy processing background. The construction of a dynamic fuzzy concept lattice is faster than in other algorithms.

4.4.2.3 A critical hierarchical construction algorithm for DF concept lattice

The DF concept lattice critical layer construction algorithm (DFCL_BorderLayer) is the first to find all critical concepts. According to a list of properties, the algorithm then constructs the DF concept lattice. The definition of the critical concept is realized by the intersection of the connotation and the union of the extension of the concept of DF. The algorithm is a bottom-up hierarchical construction algorithm.

Algorithm 4.1 Constructing the DF concept lattice layer by layer

First, according to the prior knowledge, we give the appropriate threshold to preprocess the dynamic fuzzy formalism background $(B: (X, Y, I))$. Then, we remove the meta-sets that satisfy the following two conditions.

- (1) $\{(\overline{x}, \overline{x}) \mid \exists (\overline{x}, \overline{x}) \in X, \text{ so that, } f((\overline{x}, \overline{x})) = Y\}$, which meets all the attributes of the sample collection.
- (2) $\{(\overline{y}, \overline{y}) \mid \exists (\overline{y}, \overline{y}) \in Y, \text{ so that, } g((\overline{y}, \overline{y})) = X\}$, which covers all instances of the property set.

- 1) Assuming that the first $j-1$ lattice $<(\overline{O}_{N,1}, \overline{O}_{N,1}), (\overline{A}_{N,1}, \overline{A}_{N,1})>, \dots, <(\overline{O}_{N,j-1}, \overline{O}_{N,j-1}), (\overline{A}_{N,j-1}, \overline{A}_{N,j-1})>$ temporary grid nodes of the N th layer have been generated, the following will generate the temporary grid nodes of $<(\overline{O}_{Nj}, \overline{O}_{Nj}), (\overline{A}_{Nj}, \overline{A}_{Nj})>$. Let $M = Y - \bigcup_{j=1}^r (\overline{A}_{Nj}, \overline{A}_{Nj})$. If $M \neq \emptyset$, then take $(\overline{y}_{j0}, \overline{y}_{j0})$ to make $|g((\overline{A}_{Nj}, \overline{A}_{Nj}) \cup (\overline{y}_{j0}, \overline{y}_{j0}))| = \max\{|g((\overline{A}_{Nj}, \overline{A}_{Nj})) \cup (\overline{y}, \overline{y})|\}$. Otherwise, go to (9).

- 2) Generate $\left(g((\overleftarrow{A}_{Nj}, \overrightarrow{A}_{Nj}) \cup (\overleftarrow{Y}_{j0}, \overrightarrow{Y}_{j0})), \bigcap_{(\overleftarrow{x}, \overrightarrow{x}) \in g((\overleftarrow{A}_{Nj}, \overrightarrow{A}_{Nj}) \cup (\overleftarrow{Y}_{j0}, \overrightarrow{Y}_{j0}))} f((\overleftarrow{x}, \overrightarrow{x}))\right)$, denoted as $(\alpha_{j0}, \beta_{j0})$.
- 3) Set $\varphi = \{\alpha_{j0}\}$, and let $i = 0$; $O = \{\alpha_{j0}\}$ (O is a collection of object sets).
- 4) Let $M = Y - \left(\bigcup_{j=1}^r (\overleftarrow{A}_{Nr}, \overrightarrow{A}_{Nr}) \cup \beta_{ji}\right)$. If $M \neq \emptyset$, take $(\overleftarrow{Y}_{j,i+1}, \overrightarrow{Y}_{j,i+1})$ to make $|g((\overleftarrow{A}_{Nj}, \overrightarrow{A}_{Nj}) \cup (\overleftarrow{Y}_{j,i+1}, \overrightarrow{Y}_{j,i+1}))| = \max_{(\overleftarrow{y}, \overrightarrow{y}) \in M} \{|g((\overleftarrow{A}_{Nj}, \overrightarrow{A}_{Nj}), (\overleftarrow{y}, \overrightarrow{y}))|\}$, and get $\left(g((\overleftarrow{A}_{Nj}, \overrightarrow{A}_{Nj}) \cup (\overleftarrow{Y}_{j,i+1}, \overrightarrow{Y}_{j,i+1})), \bigcap_{(\overleftarrow{x}, \overrightarrow{x}) \in g((\overleftarrow{A}_{Nj}, \overrightarrow{A}_{Nj}) \cup (\overleftarrow{Y}_{j,i+1}, \overrightarrow{Y}_{j,i+1}))} f((\overleftarrow{x}, \overrightarrow{x}))\right)$, denoted as $(\alpha'_{j,i+1}, \beta'_{j,i+1})$; otherwise, go to step (9).
- 5) Set $\beta_{j,i} = \beta_{j,i} \cup \beta'_{j,i+1}$.
- 6) If there is an element in φ that contains $\alpha'_{j,i+1}$ (i.e. $\alpha'_{j,i+1} \in \varphi$), go to step (5).
- 7) Produce and change $(\alpha'_{j,i+1}, \beta'_{j,i+1})$ to $(\alpha_{j,i+1}, \beta_{j,i+1})$.
- 8) Reset $\varphi = \varphi \cup \{\alpha_{j,i+1}\}$, $i++$; go to step (5).
- 9) If $M = \emptyset$, terminate the algorithm.

Definition 4.26 If DF concept $(\overleftarrow{C}, \overrightarrow{C}) = ((\overleftarrow{X}, \overrightarrow{X}), (\overleftarrow{Y}, \overrightarrow{Y}))$ has only one direct child node, DF concept $(\overleftarrow{C}, \overrightarrow{C})$ is the DF critical concept. Otherwise, it is called the DF extension concept.

According to the above definition, if the critical concept is $(\overleftarrow{C}_1, \overrightarrow{C}_1) = ((\overleftarrow{x}_1, \overrightarrow{x}_1), (\overleftarrow{y}_1, \overrightarrow{y}_1))$, $((\overleftarrow{x}_2, \overrightarrow{x}_2), (\overleftarrow{y}_2, \overrightarrow{y}_2)), \dots, ((\overleftarrow{x}_m, \overrightarrow{x}_m), (\overleftarrow{y}_m, \overrightarrow{y}_m))$ in dynamic fuzzy background $B : (X, Y, I)$, $(\overleftarrow{X}, \overrightarrow{X}) = (\overleftarrow{x}_1, \overrightarrow{x}_1) \cup (\overleftarrow{x}_2, \overrightarrow{x}_2) \cup \dots \cup (\overleftarrow{x}_m, \overrightarrow{x}_m)$ and $Y = (\overleftarrow{y}_1, \overrightarrow{y}_1) \cap (\overleftarrow{y}_2, \overrightarrow{y}_2) \cap \dots \cap (\overleftarrow{y}_m, \overrightarrow{y}_m)$ can be obtained.

For a conceptual $(\overleftarrow{C}', \overrightarrow{C}') = ((\overleftarrow{X}', \overrightarrow{X}'), (\overleftarrow{Y}', \overrightarrow{Y}'))$ in a dynamic fuzzy background $B : (X, Y, I)$, if $(\overleftarrow{C}', \overrightarrow{C}')$ has more than two direct subconcepts, it may be assumed that $(\overleftarrow{C}_1, \overrightarrow{C}_1) = ((\overleftarrow{x}_1, \overrightarrow{x}_1), (\overleftarrow{y}_1, \overrightarrow{y}_1))$, $(\overleftarrow{C}_2, \overrightarrow{C}_2) = ((\overleftarrow{x}_2, \overrightarrow{x}_2), (\overleftarrow{y}_2, \overrightarrow{y}_2))$, \dots , $(\overleftarrow{C}_m, \overrightarrow{C}_m) = ((\overleftarrow{x}_m, \overrightarrow{x}_m), (\overleftarrow{y}_m, \overrightarrow{y}_m))$ (where $m \geq 2$), and then $(\overleftarrow{X}', \overrightarrow{X}') = (\overleftarrow{x}_1, \overrightarrow{x}_1) \cup (\overleftarrow{x}_2, \overrightarrow{x}_2) \cup \dots \cup (\overleftarrow{x}_m, \overrightarrow{x}_m)$, $(\overleftarrow{Y}', \overrightarrow{Y}') = (\overleftarrow{y}_1, \overrightarrow{y}_1) \cap (\overleftarrow{y}_2, \overrightarrow{y}_2) \cap \dots \cap (\overleftarrow{y}_m, \overrightarrow{y}_m)$. The DF concept lattice can be generated as long as all DF critical concepts have been found.

The DF critical hierarchical construction algorithm proceeds as follows:

- (1) The dynamic fuzzy background is preprocessed and the dynamic fuzzy critical concept is obtained on the dynamic fuzzy background.
- (2) According to the hierarchical rule of the DF concept lattice, the critical concept of DF is placed on the corresponding level.
- (3) Starting from the bottom, perform the following operation layer by layer and generate the dynamic fuzzy extension concept:
 - 1 The same-layer DF critical concept determines the intersection of connotation and the union of extension with two combinations and finds the new concept of DF. Then, DF concepts with the same connotation are merged. The merging strategy leaves the intension unchanged and the epitaxy is the union operation.

- 2 For the new concept of DF, check whether its subconcept attribute difference is 1 to determine whether it is the DF direct super-concept. If so, determine its location and connection line; otherwise, consider this the DF temporary concept.
- 3 For the DF concept, the provisional concepts are divided into three cases:
 - i. If there is no inclusion relationship between the DF concept and the superconcepts of its subconcepts, the concept is the direct superconcept of its DF subconcept, and its position and connection line are determined.
 - ii. If there is an inclusion relationship between the DF concept and a superconcept of its subconcepts, then modify the concept of DF to be the intersection of connotation and the union of the extension of DF.
 - iii. If there is an inclusion relationship between the DF concept and two or more superconcepts of its subconcepts, then delete the DF temporary concept.

The algorithm is simple and clear. After all the DF boundary concepts have been obtained, we can break away from the background of DF data, which improves the efficiency of constructing the DF concept lattice. The disadvantage is that some redundant concept nodes may be generated as we break away from the dynamic fuzzy background. Compared with the DFCL_Layer algorithm, as the number of instances in the form of DF data grows, the efficiency of the DF concept lattice becomes higher, but there may be more redundant concept nodes in the DF data. Therefore, the two algorithms have their own advantages in specific applications, and we can choose one algorithm to construct a DF concept lattice in accordance with the specific case.

4.4.3 DF Concept Lattice Reduction

4.4.3.1 Background and status of concept lattice reduction

The DF concept lattice is a complex lattice structure, which places a serious obstacle in its specific application. Therefore, it is necessary to control the growth of the concept lattice nodes and to reduce the DF concept lattice. There are two common reasons for the expansion of the concept lattice. One is that there are too many attributes. In the worst case, the number of nodes in the DF concept lattice grows exponentially with the number of attributes. The second problem is the high degree of similarity between elements of the object; i.e. the difference between the different concepts is very small. If we allow nearly identical objects to be merged (similar to clustering problems), the size of the concept lattice will be reduced. This may also improve the application efficiency of the concept lattice and reduce the consumption of time and space. Therefore, reducing the DF concept lattice not only improves the application efficiency but also improves the application quality.

At this stage, there have been many studies on the reduction of the concept lattice. Oosthuizen proposed a method of pruning concept lattices [14], and Agrawal

proposed a method of clustering association rules in concept lattices [15]. Ho et al. used fuzzy set theory [16] to reduce the fuzzy concept lattice. Concept lattice reduction techniques are widely used, especially in classification systems. These prevent the use of a large concept lattice, reducing the processing time and improving the precision of the concept lattice.

4.4.3.2 Reduction based on clustering technique

In data analysis, conceptual objects with large similarities are often divided into different concepts. From a human point of view, however, this division is completely unnecessary. It is often considered that objects belong to the same concept. Based on this consideration, we can use some clustering techniques to classify similar concepts and form one concept class.

Definition 4.27 Consider a known dynamic fuzzy concept $C = (O, A)$ in the dynamic fuzzy form background $B: (X, Y, I)$ with node vector $E^{(\overleftarrow{C}, \overrightarrow{C})} = \left(E^{(\overleftarrow{C}, \overrightarrow{C})}_{(\overrightarrow{y}_1, \overrightarrow{y}_1)}, E^{(\overleftarrow{C}, \overrightarrow{C})}_{(\overrightarrow{y}_2, \overrightarrow{y}_2)}, \dots, E^{(\overleftarrow{C}, \overrightarrow{C})}_{(\overrightarrow{y}_3, \overrightarrow{y}_3)}, \dots, E^{(\overleftarrow{C}, \overrightarrow{C})}_{(\overrightarrow{y}_q, \overrightarrow{y}_q)} \right)$. For $(\overrightarrow{y}_i, \overrightarrow{y}_i) \in Y$:

$$E^{(\overleftarrow{C}, \overrightarrow{C})}_{(\overrightarrow{y}_i, \overrightarrow{y}_i)} = \begin{cases} \frac{1}{n} \sum_{j=1}^n I((\overrightarrow{o}_j, \overrightarrow{o}_j), (\overrightarrow{y}_i, \overrightarrow{y}_i))(\overrightarrow{y}_i, \overrightarrow{y}_i) \in A, (\overrightarrow{o}_j, \overrightarrow{o}_j) \in O \\ \frac{1}{k} \sum_{j=1}^k I((\overrightarrow{x}_j, \overrightarrow{x}_j), (\overrightarrow{y}_i, \overrightarrow{y}_i))(\overrightarrow{y}_i, \overrightarrow{y}_i) \notin A, (\overrightarrow{x}_j, \overrightarrow{x}_j) \in X \end{cases}, \quad (4.23)$$

where $|X| = k$, $|Y| = q$, $|O| = n$. The formula indicates that the value of $E^{(\overleftarrow{C}, \overrightarrow{C})}_{(\overrightarrow{y}_i, \overrightarrow{y}_i)}$ depends on whether attribute $(\overrightarrow{y}_i, \overrightarrow{y}_i)$ is included in connotation $(\overleftarrow{A}, \overrightarrow{A})$ of node $(\overleftarrow{C}, \overrightarrow{C})$. The dynamic fuzziness of the object is introduced into the node, which reflects the entire dynamic fuzzy characteristics of the object set on the basis of preserving the dynamic fuzzy features of a single object. Thus, the dynamic fuzzy lattice between the nodes has a theoretical basis for comparison. Each concept in DF is a node. Each node can be expressed in vector form, so that the similarity between nodes (or concepts) can be defined as a metric for concept clustering.

Definition 4.28 The distance between the dynamic fuzzy concept lattice nodes $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ and $(\overleftarrow{C}_j, \overrightarrow{C}_j)$ is defined as the Manhattan distance $d((\overleftarrow{C}_i, \overrightarrow{C}_i), (\overleftarrow{C}_j, \overrightarrow{C}_j)) = d_1 + d_2 + \dots + d_i + \dots + d_m$ between vectors $E^{(\overleftarrow{C}_i, \overrightarrow{C}_i)}$ and $E^{(\overleftarrow{C}_j, \overrightarrow{C}_j)}$, where $d_i = |E^{(\overleftarrow{C}_i, \overrightarrow{C}_i)}_{(\overrightarrow{y}_i, \overrightarrow{y}_i)} - E^{(\overleftarrow{C}_j, \overrightarrow{C}_j)}_{(\overrightarrow{y}_i, \overrightarrow{y}_i)}|$, and $E^{(\overleftarrow{C}_i, \overrightarrow{C}_i)}_{(\overrightarrow{y}_i, \overrightarrow{y}_i)} E^{C_i}_{(\overrightarrow{y}_i, \overrightarrow{y}_i)}, E^{(\overleftarrow{C}_j, \overrightarrow{C}_j)}_{(\overrightarrow{y}_i, \overrightarrow{y}_i)} E^{C_j}_{(\overrightarrow{y}_i, \overrightarrow{y}_i)}$, are the i th components of the dynamic fuzzy parameters of concepts $(\overleftarrow{C}_i, \overrightarrow{C}_i) = ((O_i, O_i), (\overleftarrow{A}_i, \overrightarrow{A}_i))$ and $(\overleftarrow{C}_j, \overrightarrow{C}_j) = ((\overleftarrow{O}_j, \overrightarrow{O}_j), (\overleftarrow{A}_j, \overrightarrow{A}_j))$, respectively.

λ_s is the similarity threshold for clustering. It can be set manually or calculated by other methods. If the distance between the two concepts is less than λ_s , the two

concepts are placed into the same concept category. If the distance between the two concepts is greater than λ_s , they belong to different concepts.

4.4.4 Extraction of DF concept rules

Presenting conceptual rules in a large dataset is a common application of knowledge discovery. This problem has been solved by generating association rules through the definition of the probability parameters [17]. However, the dynamic fuzzy concept lattice expresses the hierarchical relationship of concepts vividly. This level also implies that the rules contain a very high level of information. Mathematical parameters can be used to constrain and generate rules with greater application values.

4.4.4.1 Representation of dynamic fuzzy rules

Dynamic fuzzy association rules are the implication relations among dynamic rules: $(\overleftarrow{M}, \overrightarrow{M}) \Rightarrow (\overleftarrow{N}, \overrightarrow{N})$, where $(\overleftarrow{M}, \overrightarrow{M})$ is the premise of the dynamic association rules and $(\overleftarrow{N}, \overrightarrow{N})$ is the conclusion. The meaning of the rule is that the DF membership of each attribute in $(\overleftarrow{M}, \overrightarrow{M})$ is greater than the corresponding set of thresholds. The DF membership of each attribute in $(\overleftarrow{N}, \overrightarrow{N})$ is also greater than the corresponding threshold. DF classification rules are special association rules. When the conclusion of a rule is a decision attribute (i.e. label attribute of dynamic fuzzy concept), a DF classification rule is obtained.

To extract concept rules effectively, the DF minimum support and DF minimum confidence must be specified by the user. The main problem of dynamic fuzzy concept rule extraction is that, given an object database, finding all conception rules may require a support degree that is not lower than the user specified DF minimum support and DF minimum credibility.

Definition 4.29 Given the dynamic fuzzy concepts $(\overleftarrow{c}_i, \overrightarrow{c}_i)$ and $(\overleftarrow{c}_j, \overrightarrow{c}_j)$, and the dynamic fuzzy association rule $(\overleftarrow{M}, \overrightarrow{M}) \Rightarrow (\overleftarrow{N}, \overrightarrow{N})$, where $(\overleftarrow{M}, \overrightarrow{M}) = Intent((\overleftarrow{c}_i, \overrightarrow{c}_i))$, $(\overleftarrow{N}, \overrightarrow{N}) = Intent((\overleftarrow{c}_j, \overrightarrow{c}_j)) \setminus Intent((\overleftarrow{c}_i, \overrightarrow{c}_i))$, the dynamic fuzzy association rules support and confidence are

$$DFSUP((\overleftarrow{M}, \overrightarrow{M}) \Rightarrow (\overleftarrow{N}, \overrightarrow{N})) = \frac{|Extent((\overleftarrow{c}_j, \overrightarrow{c}_j))|}{|(\overleftarrow{O}, \overrightarrow{O})|} \quad (4.24)$$

and

$$DFCONF((\overleftarrow{M}, \overrightarrow{M}) \Rightarrow (\overleftarrow{N}, \overrightarrow{N})) = \frac{|Extent((\overleftarrow{c}_j, \overrightarrow{c}_j))|}{|Extent((\overleftarrow{c}_i, \overrightarrow{c}_i))|}. \quad (4.25)$$

The corresponding thresholds are dfs and $dfconf$, respectively.

4.4.4.2 Dynamic fuzzy rule extraction algorithm

To avoid generating redundant dynamic fuzzy association rules, two parameters are defined to control the generation of association rules. According to the definitions of mean and variance in statistics, the mean defines the fuzzy degree of the sample and the variance defines the degree of dispersion of the sample. These are introduced into dynamic fuzzy concept learning as follows.

Definition 4.30 Define two DF parameters: the dynamic fuzzy mean $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$ $((\overleftarrow{X}_i, \overrightarrow{X}_i), (\overleftarrow{Y}_i, \overrightarrow{Y}_i))$ and the dynamic fuzzy variance $(\overleftarrow{\delta}, \overrightarrow{\delta})$ $((\overleftarrow{X}_i, \overrightarrow{X}_i), (\overleftarrow{Y}_i, \overrightarrow{Y}_i))$:

$$(\overleftarrow{\alpha}, \overrightarrow{\alpha})_{(\overleftarrow{Y}_i, \overrightarrow{Y}_i)} = \frac{1}{|(\overleftarrow{X}_i, \overrightarrow{X}_i)|} \sum_{(\overleftarrow{x}_i, \overrightarrow{x}_i) \in (\overleftarrow{X}_i, \overrightarrow{X}_i)} I((\overleftarrow{x}_i, \overrightarrow{x}_i), (\overleftarrow{y}_i, \overrightarrow{y}_i)), \quad (4.26)$$

$$(\overleftarrow{\alpha}, \overrightarrow{\alpha}) = \frac{1}{|(\overleftarrow{Y}_i, \overrightarrow{Y}_i)|} \sum_{(\overleftarrow{y}_i, \overrightarrow{y}_i) \in (\overleftarrow{Y}_i, \overrightarrow{Y}_i)} (\overleftarrow{\alpha}, \overrightarrow{\alpha})_{(\overleftarrow{y}_i, \overrightarrow{y}_i)}, \quad (4.27)$$

$$\begin{aligned} (\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{Y}_i, \overrightarrow{Y}_i)} &= (\overleftarrow{\delta}, \overrightarrow{\delta}) (|I((\overleftarrow{x}_i, \overrightarrow{x}_i), (\overleftarrow{y}_i, \overrightarrow{y}_i))| |(\overleftarrow{x}_i, \overrightarrow{x}_i) \in (\overleftarrow{X}_i, \overrightarrow{X}_i)|) \\ &= \sqrt{\frac{\sum_{(\overleftarrow{x}_i, \overrightarrow{x}_i) \in (\overleftarrow{X}_i, \overrightarrow{X}_i)} (I((\overleftarrow{x}_i, \overrightarrow{x}_i), (\overleftarrow{y}_i, \overrightarrow{y}_i)) - (\overleftarrow{\alpha}, \overrightarrow{\alpha}))}{|(\overleftarrow{X}_i, \overrightarrow{X}_i)|}}, \end{aligned} \quad (4.28)$$

and

$$(\overleftarrow{\delta}, \overrightarrow{\delta})((\overleftarrow{x}_i, \overrightarrow{x}_i), (\overleftarrow{y}_i, \overrightarrow{y}_i)) = \frac{1}{|(\overleftarrow{Y}_i, \overrightarrow{Y}_i)|} \sum_{(\overleftarrow{y}_i, \overrightarrow{y}_i) \in (\overleftarrow{Y}_i, \overrightarrow{Y}_i)} (\overleftarrow{\delta}, \overrightarrow{\delta})_{(\overleftarrow{y}_i, \overrightarrow{y}_i)}, \quad (4.29)$$

where $|(\overleftarrow{X}_i, \overrightarrow{X}_i)|$ is the length of element $(\overleftarrow{X}_i, \overrightarrow{X}_i)$ and $|(\overleftarrow{Y}_i, \overrightarrow{Y}_i)|$ is the length of element $(\overleftarrow{Y}_i, \overrightarrow{Y}_i)$. $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$ denotes the dynamic average fuzzy degree of concept $(\overleftarrow{c}_i, \overrightarrow{c}_i)$, and $(\overleftarrow{\delta}, \overrightarrow{\delta})$ illustrates the discrete degree of dynamic fuzzy membership of concept $(\overleftarrow{c}_i, \overrightarrow{c}_i)$.

Definition 4.31 $\forall (\overleftarrow{c}_i, \overrightarrow{c}_i) = ((\overleftarrow{X}_i, \overrightarrow{X}_i), (\overleftarrow{Y}_i, \overrightarrow{Y}_i))$, if the parameter $(\overleftarrow{\alpha}, \overrightarrow{\alpha})((\overleftarrow{X}_i, \overrightarrow{X}_i), (\overleftarrow{Y}_i, \overrightarrow{Y}_i))$ is greater than or equal to the given threshold $(\overleftarrow{\theta}, \overrightarrow{\theta})$ and the parameter $(\overleftarrow{\delta}, \overrightarrow{\delta})$ is smaller than the threshold $(\overleftarrow{\psi}, \overrightarrow{\psi})$, then this is a candidate node.

Dynamic fuzzy association rule generation algorithm (DFCL_Rule):

- (1) The dynamic fuzzy formal background is preprocessed to generate the dynamic fuzzy processing background. The corresponding dynamic fuzzy concept lattice is generated according to the dynamic fuzzy processing background.

- (2) The breadth-first traversal of the corresponding dynamic fuzzy concept lattice is carried out, and each dynamic fuzzy concept node is judged. If $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$ is greater than the prior parameter $(\overleftarrow{\theta}, \overrightarrow{\theta})$ and $(\overleftarrow{\delta}, \overrightarrow{\delta})$ is less than the prior parameter $(\overleftarrow{\psi}, \overrightarrow{\psi})$, then the dynamic fuzzy concept nodes are dynamic fuzzy frequent nodes.
- (3) Enumerating all dynamic fuzzy node pairs $\langle (\overleftarrow{c}_i, \overrightarrow{c}_i), (\overleftarrow{c}_j, \overrightarrow{c}_j) \rangle$ in the dynamic fuzzy concept lattice, if $(\overleftarrow{c}_i, \overrightarrow{c}_i)$ and $(\overleftarrow{c}_j, \overrightarrow{c}_j)$ have a father child relationship and satisfy $\frac{\text{Min}\{(\overleftarrow{\alpha}, \overrightarrow{\alpha})|_{(\overleftarrow{c}_i, \overrightarrow{c}_i)}, (\overleftarrow{\alpha}, \overrightarrow{\alpha})|_{(\overleftarrow{c}_j, \overrightarrow{c}_i)}\}}{\text{Max}\{(\overleftarrow{\alpha}, \overrightarrow{\alpha})|_{(\overleftarrow{c}_i, \overrightarrow{c}_i)}, (\overleftarrow{\alpha}, \overrightarrow{\alpha})|_{(\overleftarrow{c}_j, \overrightarrow{c}_i)}\}} \geq (\overleftarrow{\varphi}, \overrightarrow{\varphi})$, $\langle (\overleftarrow{c}_i, \overrightarrow{c}_i), (\overleftarrow{c}_j, \overrightarrow{c}_j) \rangle$ is called a dynamic fuzzy candidate pair.
- (4) According to the generated multiple dynamic fuzzy tuples, the dynamic fuzzy association rule $(\overleftarrow{M}, \overrightarrow{M}) \Rightarrow (\overleftarrow{N}, \overrightarrow{N})$ is generated. The dynamic fuzzy association rule is generated with two different thresholds $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$ and $(\overleftarrow{\delta}, \overrightarrow{\delta})$ specified by $(\overleftarrow{M}, \overrightarrow{M}) = \text{Intent}((\overleftarrow{c}_i, \overrightarrow{c}_i)), (\overleftarrow{N}, \overrightarrow{N}) = \text{Intent}((\overleftarrow{c}_j, \overrightarrow{c}_j)) \setminus \text{Intent}((\overleftarrow{c}_i, \overrightarrow{c}_i))$, which can generate dynamic fuzzy classification rules.

4.4.5 Examples of algorithms and experimental analysis

The dynamic fuzzy processing background is obtained by preprocessing the DFD sets. Based on this, the DF concept lattice is generated by DFCL_BorderLayer. Then, the dynamic fuzzy concept rules are generated. Finally, the two algorithms in this section are compared with some common algorithms.

4.4.5.1 Examples of algorithms

For example, suppose that the dynamic fuzzy background after feature extraction is as presented in Tab. 4.2. According to BorderLayerDFCL algorithm, the dynamic fuzzy processing background is given in Tab. 4.3.

First, we calculate the threshold of the dynamic fuzzy attributes according to Tab. 4.2 and obtain the dynamic fuzzy processing background (Tab. 4.3) after reduction, which improves the precision of constructing the dynamic fuzzy concept lattice. Second, the dynamic fuzzy concept lattice is constructed according to the algorithm, as shown in Fig. 4.2.

For example, assume that the threshold $(\overleftarrow{\theta}, \overrightarrow{\theta})$ of $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$ is $(0.7, 0.7)$ and the threshold $(\overleftarrow{\delta}, \overrightarrow{\delta})$ of $(\overleftarrow{\gamma}, \overrightarrow{\gamma})$ is $(0.3, 0.3)$. Then, the frequent nodes in the dynamic fuzzy concept lattice are $(\{12345\}, \{d_1d_2d_7\})$, $(\{2345\}, \{d_1d_2d_7d_8\})$, $(\{78\}, \{d_1d_2d_4d_6\})$, $(\{8\}, \{d_1d_2d_3d_4d_6\})$. The dynamic fuzzy support degree is $(0.7, 0.7)$ and the dynamic fuzzy confidence degree is $(0.95, 0.95)$. The dynamic fuzzy candidate double tuples are $\langle (\{12345\}, \{d_1d_2d_7\}), (\{2345\}, \{d_1d_2d_7d_8\}) \rangle$ and $\langle (\{78\}, \{d_1d_2d_4d_6\}), (\{8\}, \{d_1d_2d_3d_4d_6\}) \rangle$. The dynamic fuzzy association rules can then be extracted. According to $d_i = (\overleftarrow{y}_i, \overrightarrow{y}_i) i = 1, 2 \dots 9$, the dynamic fuzzy association rules listed in Tab. 4.4 are obtained.

Tab. 4.2: Dynamic fuzzy form backgrounds.

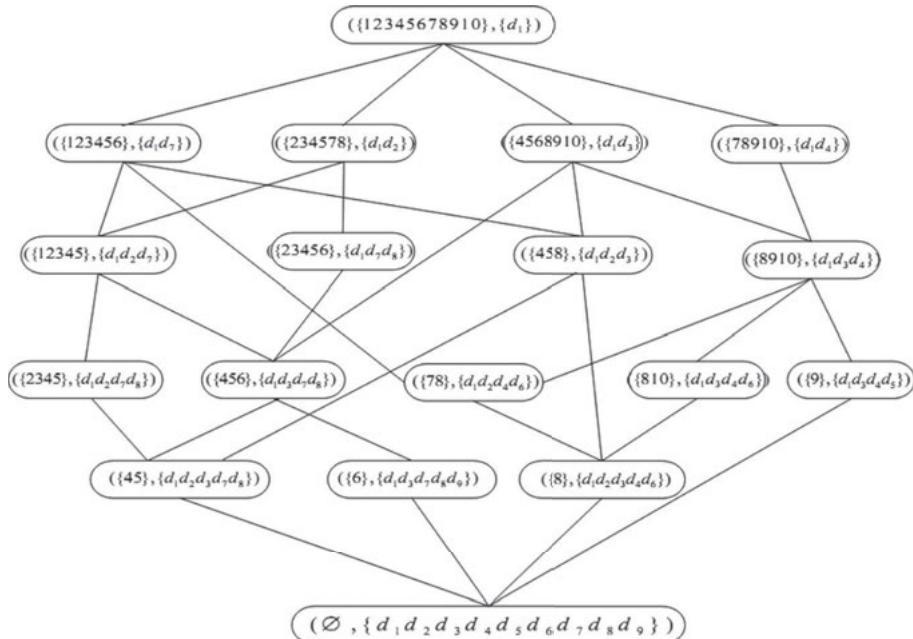
OB	Attributes					
	(\bar{y}_1, \bar{y}_1)	(\bar{y}_2, \bar{y}_2)	(\bar{y}_3, \bar{y}_3)	(\bar{y}_4, \bar{y}_4)	(\bar{y}_5, \bar{y}_5)	(\bar{y}_6, \bar{y}_6)
1	$(0.8, 0.8)$	$(0.9, 0.9)$	$(0.2, 0.2)$	$(0.1, 0.1)$	$(0.2, 0.2)$	$(0.1, 0.1)$
2	$(0.8, 0.8)$	$(0.9, 0.9)$	$(0.1, 0.1)$	$(0.2, 0.2)$	$(0.3, 0.3)$	$(0.2, 0.2)$
3	$(0.8, 0.8)$	$(0.7, 0.7)$	$(0.2, 0.2)$	$(0.1, 0.1)$	$(0.3, 0.3)$	$(0.1, 0.1)$
4	$(0.8, 0.8)$	$(0.8, 0.8)$	$(0.6, 0.6)$	$(0.1, 0.1)$	$(0.1, 0.1)$	$(0.9, 0.9)$
5	$(0.8, 0.8)$	$(0.6, 0.6)$	$(0.7, 0.7)$	$(0.2, 0.2)$	$(0.3, 0.3)$	$(0.2, 0.2)$
6	$(0.8, 0.8)$	$(0.7, 0.7)$	$(0.7, 0.7)$	$(0.1, 0.1)$	$(0.1, 0.1)$	$(0.9, 0.9)$
7	$(0.8, 0.8)$	$(0.9, 0.9)$	$(0.2, 0.2)$	$(0.6, 0.6)$	$(0.3, 0.3)$	$(0.6, 0.6)$
8	$(0.8, 0.8)$	$(0.8, 0.8)$	$(0.9, 0.9)$	$(0.7, 0.7)$	$(0.2, 0.2)$	$(0.6, 0.6)$
9	$(0.8, 0.8)$	$(0.2, 0.2)$	$(0.8, 0.8)$	$(0.8, 0.8)$	$(0.7, 0.7)$	$(0.6, 0.6)$
$(\bar{\lambda}, \bar{\lambda})$	$(0.8, 0.8)$	$(0.3, 0.3)$	$(0.7, 0.7)$	$(0.37, 0.37)$	$(0.1, 0.1)$	$(0.6, 0.6)$

Tab. 4.3: Dynamic fuzzy processing background.

OBS	Attributes sets	$(\overleftarrow{\alpha}, \overrightarrow{\alpha})$	$(\overleftarrow{\delta}, \overrightarrow{\delta})$
{1}	$\{(\overleftarrow{y}_1, \overrightarrow{y}_1), (\overleftarrow{y}_2, \overrightarrow{y}_2), (\overleftarrow{y}_7, \overrightarrow{y}_7)\}$	$(0.83, 0.83)$	$(0.04, 0.04)$
{2,3}	$\{(\overleftarrow{y}_1, \overrightarrow{y}_1), (\overleftarrow{y}_2, \overrightarrow{y}_2), (\overleftarrow{y}_7, \overrightarrow{y}_7), (\overleftarrow{y}_8, \overrightarrow{y}_8)\}$	$(0.79, 0.79)$	$(0.08, 0.08)$
{4,5}	$\{(\overleftarrow{y}_1, \overrightarrow{y}_1), (\overleftarrow{y}_2, \overrightarrow{y}_2), (\overleftarrow{y}_3, \overrightarrow{y}_3), (\overleftarrow{y}_7, \overrightarrow{y}_7), (\overleftarrow{y}_8, \overrightarrow{y}_8)\}$	$(0.77, 0.77)$	$(0.31, 0.31)$
{6}	$\{(\overleftarrow{y}_1, \overrightarrow{y}_1), (\overleftarrow{y}_3, \overrightarrow{y}_3), (\overleftarrow{y}_7, \overrightarrow{y}_7), (\overleftarrow{y}_8, \overrightarrow{y}_8), (\overleftarrow{y}_9, \overrightarrow{y}_9)\}$	$(0.78, 0.78)$	$(0.10, 0.10)$
{7}	$\{(\overleftarrow{y}_1, \overrightarrow{y}_1), (\overleftarrow{y}_2, \overrightarrow{y}_2), (\overleftarrow{y}_4, \overrightarrow{y}_4), (\overleftarrow{y}_6, \overrightarrow{y}_6)\}$	$(0.73, 0.73)$	$(0.13, 0.13)$
{8}	$\{(\overleftarrow{y}_1, \overrightarrow{y}_1), (\overleftarrow{y}_2, \overrightarrow{y}_2), (\overleftarrow{y}_3, \overrightarrow{y}_3), (\overleftarrow{y}_4, \overrightarrow{y}_4), (\overleftarrow{y}_6, \overrightarrow{y}_6)\}$	$(0.76, 0.76)$	$(0.09, 0.09)$
{9}	$\{(\overleftarrow{y}_1, \overrightarrow{y}_1), (\overleftarrow{y}_3, \overrightarrow{y}_3), (\overleftarrow{y}_4, \overrightarrow{y}_4), (\overleftarrow{y}_5, \overrightarrow{y}_5), (\overleftarrow{y}_6, \overrightarrow{y}_6)\}$	$(0.74, 0.74)$	$(0.07, 0.07)$
{10}	$\{(\overleftarrow{y}_1, \overrightarrow{y}_1), (\overleftarrow{y}_3, \overrightarrow{y}_3), (\overleftarrow{y}_4, \overrightarrow{y}_4), (\overleftarrow{y}_6, \overrightarrow{y}_6)\}$	$(0.67, 0.67)$	$(0.15, 0.15)$

Tab. 4.4: Generated partial DF rules.

Number	Rule	Confidence
1	$\{(\overleftarrow{y}_1, \overrightarrow{y}_1), (\overleftarrow{y}_2, \overrightarrow{y}_2), (\overleftarrow{y}_7, \overrightarrow{y}_7)\} \Rightarrow \{(\overleftarrow{y}_8, \overrightarrow{y}_8)\}$	$(0.951, 0.951)$
2	$\{(\overleftarrow{y}_1, \overrightarrow{y}_1), (\overleftarrow{y}_2, \overrightarrow{y}_2), (\overleftarrow{y}_4, \overrightarrow{y}_4), (\overleftarrow{y}_6, \overrightarrow{y}_6)\} \Rightarrow \{(\overleftarrow{y}_3, \overrightarrow{y}_3)\}$	$(0.961, 0.961)$

**Fig. 4.2:** Dynamic fuzzy concept lattice.

4.4.5.2 Experimental comparison

The following experiments compare the run times of DFCL_BorderLayer and DFCL_Layer algorithm with those of several algorithms with good performance, including Godin's algorithm, Bordat's algorithm, and Norris' algorithm. The test platform configuration used an Intel 2.79 GHz processor with 512 MB memory. The experiment used randomly generated data with a total of 100 attributes.

It can be seen from Fig. 4.3 that the performance of DFCL_Layer and DFCL_BorderLayer increases with an increase in the data size. This verifies the validity of the proposed method.

4.5 Concept learning model based on DFDT

Combining the knowledge of the DFDT, this section discusses a conceptual learning model based on DFDT. First, we introduce the DF concept tree and the generating strategy. Then, we analyse the generation process of the DF concept tree in detail before discussing the DF concept rule extraction and matching algorithm.

4.5.1 DF concept tree and generating strategy

Definition 4.32 The DF concept tree is a tree structure. The instances are arranged from the root node to a leaf node for concept classification. Each node on the tree specifies the value of a particular element of the concept to which the instance belongs. Its successor branch corresponds to the value of the connotation element, and the leaf node corresponds to a specific concept category.

Different from the traditional step-by-step search space method, the dynamic fuzzy concept tree is a decision tree method for dealing with DF data. The basic idea is to use a DF rule matching algorithm to make decisions. Because the learning

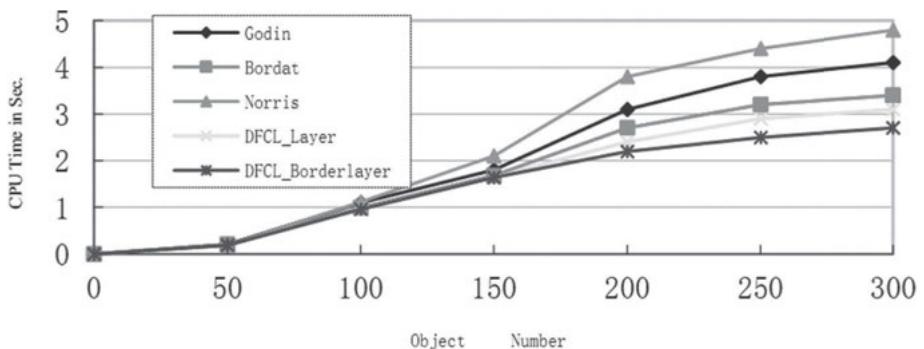


Fig. 4.3: Performance comparison of the five algorithms.

strategy requires the input values to be discrete, the datasets should be subjected to dimension reduction and discretization, which concerns the dynamic fuzzy domain and the processing of dynamic fuzzy attributes. We then consider how to choose test attributes and select a processing method for special attributes. Finally, we introduce the dynamic fuzzy concept tree construction algorithm, which incorporates dynamic fuzzy rule extraction and dynamic fuzzy concept tree pruning to improve the precision of the concept tree.

4.5.2 Generation of DF Concepts

4.5.2.1 DF attribute selection metric

1) Information gain standard

The key problem in constructing a DF concept tree algorithm is how to select the best classification attributes. The choice is usually based on “information gain”, which measures the ability of a given set of attributes.

Given the DF target concept training set S , the entropy in the classical Boolean type is defined as

$$\text{Entropy}(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}. \quad (4.30)$$

This can be written in the following form:

$$\text{Entropy}(S) \equiv \sum_{i=1}^c -p_i \log_2 p_i, \quad (4.31)$$

where p_i represents the ratio of the number of samples labelled as i in sample set S to the total number of samples in S ; c is the total number of labelled samples.

$$\text{Gain}(S, A) \equiv \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v), \quad (4.32)$$

where $\text{Values}(A)$ is a possible set of values for attribute A and S_v is a subset of the values v of attribute A in S .

Because the dynamic fuzzy attribute is a nonclassical data value, the relevant parameters in the definition of information entropy should be adjusted. We consider two options for this, and choose one of them to achieve.

The first scheme finds the DF membership in the DF concept sample by setting a dynamic fuzzy threshold $(\overleftarrow{\delta}, \overrightarrow{\delta})$. $((\overleftarrow{0}, \overrightarrow{0}) < (\overleftarrow{\delta}, \overrightarrow{\delta}) < (\overleftarrow{1}, \overrightarrow{1}))$. p_{\oplus} is the ratio of positive cases of DF membership degree in S that are greater than the threshold

$(\overleftarrow{\delta}, \overrightarrow{\delta})$. p_{\oplus} is the ratio of counter-cases of DF membership degree in S that are smaller than the threshold $(\overleftarrow{\delta}, \overrightarrow{\delta})$. The other does not change.

The second scheme takes the training sample S that generates the DF concept, and obtains the sum of the DF memberships of all samples in this description. This process is repeated for the membership of each description in the intersection of the DF concept and the positive and negative examples. p_{\oplus} is the ratio of the sum of positive cases of DF membership degree in S and DF membership degree in the DF concept in the sum of the two classes. p_{\oplus} is the ratio of the sum of counter-cases of DF membership degree in S and DF membership degree in the DF concept in the sum of the two classes. The other does not change.

The higher the information gain of the DF attribute, the better the partitioning ability. Therefore, the information gain is often applied to the candidate attribute set to select the best classification attributes.

2) Gini index standard

The Gini index standard is based on distance measurement. The distance refers to the ratio distribution distance of the concept class. Assuming there are m DF concept classes in the training sample set S, $\text{Gini}(S)$ is defined as follows:

$$\text{Gini}(S) = 1.0 - \sum_{i=1}^m p_i^2, \quad (4.33)$$

where p_i is the probability of the i th class appearing in the training sample set. Assuming that S is divided into two classes S_1 and S_2 , $\text{Gini}(S)$ is

$$\text{Gini}(S) = \frac{n_1}{n} \text{Gini}(S_1) + \frac{n_2}{n} \text{Gini}(S_2). \quad (4.34)$$

The smaller the Gini index of the DF attribute, the better the partitioning ability of the sample set. The advantage of this method is that the Gini index ensures the attribute contain a lot of useful information; the disadvantage is that the attribute selection becomes worse as the number of concept categories increases.

3) Relief standard

Kira and Rendell proposed the Relief standard in 1992. The basic idea of the algorithm is to randomly extract an instance and measure the dependency between attributes by the distance between the instance and the attribute value of the neighbour instance. The algorithm defines an attribute evaluation value $W[(\overleftarrow{A}, \overrightarrow{A})]$ of the sample set S. The initial state is 0. We randomly select m instances from the sample set S. For each record $(\overleftarrow{x}_{c_i}, \overrightarrow{x}_{c_i})$, select the same category $(\overleftarrow{x}'_{c_i}, \overrightarrow{x}'_{c_i})$ and the closest different categories $(\overleftarrow{x}''_{c_j}, \overrightarrow{x}''_{c_j})$. Check each attribute value $(\overleftarrow{A}, \overrightarrow{A})$. If the values of two instances

in different classes are the same, add one value; otherwise, subtract one value. The formula is as follows:

$$W[(\overleftarrow{A}, \overrightarrow{A})] = W[(\overleftarrow{A}, \overrightarrow{A})] - \text{diff}((\overleftarrow{A}, \overrightarrow{A}), (\overleftarrow{x}_{C_i}, \overrightarrow{x}_{C_i}), (\overleftarrow{x}'_{C_i}, \overrightarrow{x}'_{C_i})) \\ + \text{diff}((\overleftarrow{A}, \overrightarrow{A}), (\overleftarrow{x}_{C_i}, \overrightarrow{x}_{C_i}), (\overleftarrow{x}''_{C_j}, \overrightarrow{x}''_{C_j})), \quad (4.35)$$

where the attribute is a classification type, and the value of $\text{diff}((\overleftarrow{A}, \overrightarrow{A}), (\overleftarrow{I}_1, \overrightarrow{I}_1), (\overleftarrow{I}_2, \overrightarrow{I}_2))$ is

$$\text{diff}((\overleftarrow{A}, \overrightarrow{A}), (\overleftarrow{I}_1, \overrightarrow{I}_1), (\overleftarrow{I}_2, \overrightarrow{I}_2)) = \\ \begin{cases} 0 & |DFvalue((\overleftarrow{A}, \overrightarrow{A}), (\overleftarrow{I}_1, \overrightarrow{I}_1)) - DFvalue((\overleftarrow{A}, \overrightarrow{A}), (\overleftarrow{I}_2, \overrightarrow{I}_2))| < (\overleftarrow{\varepsilon}, \overrightarrow{\varepsilon}), \\ 1 & \text{otherwise} \end{cases}, \quad (4.36)$$

where $(\overleftarrow{\varepsilon}, \overrightarrow{\varepsilon})$ is the smaller positive dynamic fuzzy threshold that is divided into discrete attributes. The larger the value of $W[(\overleftarrow{A}, \overrightarrow{A})]$ between the attributes, the better the classification performance. We select the attributes that give the best classification.

Each of the three attribute selection strategies has its own characteristics. If the properties of the sample set S are very different, then the information gain or Gini criterion should be used. If there are dependencies among the attributes of the sample set S, the Relief criterion is the most appropriate.

4.5.2.2 Discretization of DFD

The dynamic fuzzy concept tree generation algorithm selects the value of the best classification property to expand the branch. In general, there are several values for each branch. If the value of the property is continuous, then the input data should be discretized. Namely, we need to divide the numerical values in the data set into several sub-intervals. This process is called the discretization of numerical data [19]. According to differences in the classification information, discretization is divided into unsupervised discretization and supervised discretization. Unsupervised discretization may destroy the completeness of the information and the usefulness of the data in the learning process. Supervised discretization takes the class attribute information into account and has a high accuracy rate in the learning process. Commonly used discretization strategies include the split strategy, adaptive strategy, equal frequency interval strategy, and class information entropy division strategy [20].

Suppose the condition attribute set of the instance set is $(\overleftarrow{C}, \overrightarrow{C}) = \{(\overleftarrow{A}_1, \overrightarrow{A}_1), (\overleftarrow{A}_2, \overrightarrow{A}_2), \dots, (\overleftarrow{A}_n, \overrightarrow{A}_n)\}$. The value range of attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i) \in (\overleftarrow{C}, \overrightarrow{C})$ is $(\overleftarrow{V}_{\overleftarrow{A}_i}, \overrightarrow{V}_{\overrightarrow{A}_i})$

and the discretization scheme for the single numeric attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ is a partition $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$ of $(\overleftarrow{V}_{\overleftarrow{A}_i}, \overrightarrow{V}_{\overrightarrow{A}_i})$.

The instance set $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$ after the attribute filter is recorded as $(\overleftarrow{U}_N, \overrightarrow{U}_N)$. $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ is any condition attribute of $(\overleftarrow{U}_N, \overrightarrow{U}_N)$, and is of numeric type. The discretization of $(\overleftarrow{U}_N, \overrightarrow{U}_N)$ is a multiple-attribute discretization.

Suppose that $(\overleftarrow{\pi}_1, \overrightarrow{\pi}_1) = \{(\overleftarrow{\pi}_1^1, \overrightarrow{\pi}_1^1), (\overleftarrow{\pi}_2^1, \overrightarrow{\pi}_2^1), \dots, (\overleftarrow{\pi}_n^1, \overrightarrow{\pi}_n^1)\}$ and $(\overleftarrow{\pi}_2, \overrightarrow{\pi}_2) = \{(\overleftarrow{\pi}_1^2, \overrightarrow{\pi}_1^2), (\overleftarrow{\pi}_2^2, \overrightarrow{\pi}_2^2), \dots, (\overleftarrow{\pi}_n^2, \overrightarrow{\pi}_n^2)\}$ are two discretization schemes of $(\overleftarrow{U}_N, \overrightarrow{U}_N)$, where $(\overleftarrow{\pi}_i^1, \overrightarrow{\pi}_i^1)$ is the discretization scheme of attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ in $(\overleftarrow{\pi}_1, \overrightarrow{\pi}_1)$ and $(\overleftarrow{\pi}_j^2, \overrightarrow{\pi}_j^2)$ is the discretization scheme of attribute $(\overleftarrow{A}_j, \overrightarrow{A}_j)$ in $(\overleftarrow{\pi}_j^2, \overrightarrow{\pi}_j^2)$.

As $\overleftarrow{\pi}_j^i$ is a discretization scheme for attribute $(\overleftarrow{A}_j, \overrightarrow{A}_j)$, the partial order relation in the discretization is defined as $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \leq (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$ if and only if each attribute $(\overleftarrow{A}_k, \overrightarrow{A}_k)$ in the instance set $(\overleftarrow{U}_N, \overrightarrow{U}_N)$ satisfies $(\overleftarrow{A}_k^i, \overrightarrow{A}_k^i) \leq (\overleftarrow{A}_k^j, \overrightarrow{A}_k^j)$, $k = 1, 2, \dots, n$, where $(\overleftarrow{\pi}_k^i, \overrightarrow{\pi}_k^i)$ is the division of $(\overleftarrow{V}_{\overleftarrow{A}_k}, \overrightarrow{V}_{\overrightarrow{A}_k})$ in $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$. \leq defines the degree between the discretization scheme as a partial order relationship.

In $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$, the value of attribute $(\overleftarrow{A}_k, \overrightarrow{A}_k)$ is divided into an interval. Then, the partition of attribute $(\overleftarrow{A}_k, \overrightarrow{A}_k)$ in $(\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$ must belong to the same interval. Two or more adjacent segmentation intervals of the value of attribute $(\overleftarrow{A}_k, \overrightarrow{A}_k)$ in $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$ are combined into one partition interval of the value of attribute $(\overleftarrow{A}_k, \overrightarrow{A}_k)$ in $(\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$. Because $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \leq (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$, $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$ is said to be the refinement of $(\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$.

The intersection and union operations of the discretization scheme of set $(\overleftarrow{U}_N, \overrightarrow{U}_N)$ are

$$\begin{aligned} (\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \cap (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j) &= \{(\pi_1^i, \pi_1^i) \wedge (\pi_1^j, \pi_1^j), (\pi_2^i, \pi_2^i) \wedge (\pi_2^j, \pi_2^j), \dots, (\pi_n^i, \pi_n^i) \wedge (\pi_n^j, \pi_n^j)\} \\ (\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \cup (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j) &= \{(\pi_1^i, \pi_1^i) \vee (\pi_1^j, \pi_1^j), (\pi_2^i, \pi_2^i) \vee (\pi_2^j, \pi_2^j), \dots, (\pi_n^i, \pi_n^i) \vee (\pi_n^j, \pi_n^j)\} \end{aligned}$$

The division of $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \cap (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$ to an instance set $(\overleftarrow{U}_N, \overrightarrow{U}_N)$ is the biggest one that is smaller than $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i)$, $(\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$. $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \cup (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$ is the smallest division that is bigger than $(\overleftarrow{\pi}_i, \overrightarrow{\pi}_i) \cap (\overleftarrow{\pi}_j, \overrightarrow{\pi}_j)$.

We define the partial order relation \leq and the intersection operation and union operation of the discretization scheme. Then, we can obtain a lattice structure that takes various discretization schemes of the instance set $(\overleftarrow{U}_N, \overrightarrow{U}_N)$ as elements. This is called a dynamic fuzzy discrete lattice [20].

As all kinds of discretization schemes are elements of dynamic fuzzy discrete lattices, the problem of discretizing the instance set $(\overleftarrow{U}_N, \overrightarrow{U}_N)$ is transformed to the search of dynamic fuzzy discrete lattices. When attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ is discretized, there are many separation points in $(\overleftarrow{V}_{\overleftarrow{A}_i}, \overrightarrow{V}_{\overrightarrow{A}_i})$. If some of these points are deleted, they will not affect the discretization result. Thus, the node is a redundant separation point.

Definition 4.33 For attribute $(\overrightarrow{A}_i, \overrightarrow{A}_i)$, there is a partition set $(\overleftarrow{S}, \overrightarrow{S}) = \{(\overleftarrow{S}_1, \overleftarrow{S}_1), (\overleftarrow{S}_2, \overleftarrow{S}_2), \dots, (\overleftarrow{S}_k, \overleftarrow{S}_k)\}$. If $(\overleftarrow{S}_i, \overrightarrow{S}_i) \in (\overleftarrow{S}, \overrightarrow{S})$ is deleted, merge the two separate intervals. The attribute values in the two intervals are replaced by the same dynamic fuzzy number $(\overline{\alpha}, \overline{\alpha})$. If the resulting instance sets are consistent, the separation point $(\overleftarrow{S}_i, \overrightarrow{S}_i) \in (\overleftarrow{S}, \overrightarrow{S})$ is called a redundant separation point.

Definition 4.34 The number of redundant separation points in a discretization scheme of attribute $(\overrightarrow{A}_i, \overrightarrow{A}_i)$ is called the separation point level of $(\overrightarrow{A}_i, \overrightarrow{A}_i)$.

Obviously, the higher the separation point level, the more redundant separation points correspond to the discretization of this attribute. The goal of the discretization is to minimize the number of redundant separation points while preserving the consistency of the discretization results [21]. The basic steps are as follows. First, calculate the separation point level of each attribute and sort the various attributes in accordance with the separation point level from small to large to obtain a property sequence containing redundant separation points. Then, remove the redundant separation points in turn. Each operation only deletes one redundant separation point. The algorithm is described as follows:

- (1) Find the separation point level for each discretization property and the set of separable points that can be removed.
- (2) Each attribute is sorted from small to large according to the separation point level to obtain a sequence of attributes.
- (3) Each attribute in the attribute sequence is processed in turn. If the attribute in the i th position has redundant separation points, the interval adjacent to this separation point is merged to judge the consistency of the instance set. If it is consistent, the number of redundant separation points of the i th attribute is decreased by 1 and we update the instance set. Otherwise, keep the separation point. Go to (3) to carry out the next round of deleting redundant separation points.
- (4) If there are no redundant separation points for each attribute in the attribute sequence, terminate the algorithm and obtain the discretized set of instances.

From this algorithm, it can be seen that the operation of deleting redundant separation points is a process of increasing the discretization scheme. Thus, we could obtain a sequence of discretization schemes with a decreasing number of separation points. The two adjacent discretization schemes in the sequence satisfy the partial ordering relation \leq , and the last scheme in the sequence is the discretization scheme of the instance set $(\overleftarrow{U}_N, \overrightarrow{U}_N)$.

In a discretization based on the dynamic fuzzy partitioning lattice, the consistency of the instance set is considered first. Retaining consistency could limit the approximate degree of the discretization to the original instance set. At the same time, we

must avoid the data conflict caused by discretization based on information entropy. According to the value of the property we are operating on, there may not be any guidelines for terminating the algorithm. Thus, we aim to reduce the influence of human factors in the process of discretizing the instance set.

4.5.2.3 Construction algorithm for the DF concept tree

To construct a dynamic fuzzy concept tree on a given data set, the following algorithm is performed:

- (1) Pre-treat the discretization of the input sample data to get the threshold set V . Select the appropriate attribute set as the candidate attribute set according to the actual situation.
- (2) For the candidate attribute set, use the information gain to select the best classification attribute.
- (3) According to the difference of the current decision attribute, the sample set is divided into several subsets.
Repeat steps (2) and (3) for each subset obtained in (3) until one of the following conditions is satisfied:
 - a) All tuples in a subset belong to the same concept (or class).
 - b) The subset has traversed all the attributes.
 - c) All candidate attributes in the subset have the same value and cannot be divided.
- (4) Annotate the leaf node concept category: Case a) is annotated according to the concept of data decision attributes. For the leaf node produced by b) or c), select the representative feature to label the concept. Normally, the concept with the largest number of elements in the sample set should be selected for labelling.

The DF concept tree obtained by the above algorithm is a good data structure for concept learning.

4.5.2.4 Pruning strategy for the DF concept tree

After the dynamic fuzzy concept tree has been constructed, some abnormal branches may be constructed because the sample itself may be noisy. This may lead to the overfitting of decision trees and training examples. Hence, we usually need to prune the generated tree. Common pruning strategies use statistical methods to remove the most unreliable branches. Pruning strategies are usually divided into two types: pre-pruning and post-pruning.

The basic idea of pre-pruning is to avoid the problem of overfitting by not building an entire decision tree. This strategy is realized by judging whether the current node wants to continue dividing the sample set contained in the node. If the decision is to stop the partition, the current node becomes the leaf node, and the leaf node may contain a subset of the category sample of different concepts. The judgment is mainly

based on χ^2 , information gain, and other methods. The disadvantage of pre-pruning is that it is difficult to estimate when to stop. It is possible that the decision tree will be too simplistic if the estimated threshold is large, whereas if the threshold is small, we may have redundant trunks that cannot be pruned.

The basic idea of post-pruning is to allow overfitting in the process of building a tree and then prune the data afterwards. This method was first proposed by Breiman and others, and has been successfully applied. Table 4.5 compares a series of methods derived from this strategy.

Pre-pruning is suitable for solving large-scale problems, although post-pruning is known to be more successful in practice. Datasets processed by the initial phase of the DFDT are not very large. In the process of constructing the DFDT, we do not want to intervene so that we obtain the original pruning decision tree and identify the pros and cons of the DFDT from the classification error rate of the original decision tree. Therefore, we use a post-pruning method in DFDT. EBP does not require an additional set of pruning examples and has good pruning speed and accuracy; the literature comparing pruning methods shows that it also has a good predictive effect. After pruning, not only does the tree retain its predictive ability, but also the complexity of the tree has also been reduced.

The EBP method described above uses a bottom-up approach to examine each sub-DF concept tree and calculate the error rate of the DF concept tree and the error rate of restoring it into leaf nodes. If the branch cannot reduce the error rate, it should be pruned. In other words, when the error rate of the sub-DF concept tree is greater than the error rate of the reduced leaf nodes, we reduce the DF concept tree to a leaf node.

Estimation error rate of leaf nodes:

$$e_t = \frac{f + \frac{Z^2}{2N} + Z \sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{Z^2}{4N}}}{1 + \frac{Z^2}{N}}, \quad (4.37)$$

where $f = \frac{E}{N}$ gives the error rate of node classification, E is the number of instances of misclassification, and N is the total number of instances falling into this leaf node. When the reliability is 25%, the value of Z is 0.69.

Estimation error rate e_T of sub-DF concept:

$$e_T = \sum_{i=1}^k \frac{N_i}{N} e_i, \quad (4.38)$$

where N is the total number of root nodes of the sub-DF concept tree T , N_i is the number of instances of the i th child of T , and T has k child nodes.

Tab. 4.5: Comparison of DF decision tree pruning methods.

Pruning algorithm	Main idea	Trim subset	Advantages	Disadvantages
Minimal cost-complexity pruning	Bottom-up (traversing all internal nodes). Use cross-validation or standard error to estimate accuracy	Not necessary	Obtaining a tree of suitable size	Accuracy is not stable and may lead to excessive pruning
Minimum error pruning	Bottom-up, based on probability estimation error rate	Necessary		Low anti-noise ability and accuracy, trim the sample
Laplacian error estimation	The Laplace probability estimate is used as the expected error rate	Necessary		High complexity, additional trim samples are required
Pessimistic error pruning	From top to bottom, a continuous correction factor is used	Not necessary	High efficiency, high speed, no need to trim samples	Low anti-noise ability, top to bottom level effect
Critical value pruning	The threshold is used as the branch selection criteria to determine whether to retain the tree branch	Necessary		The cutoff value should not be set and an additional trim sample is required
Reduced error pruning	Bottom-up, find the most suitable pruning tree, need to trim the set of examples	Necessary	High accuracy, moderate size of the tree	Additional trim samples are needed, rare cases are easily overlooked
Error-based pruning	Bottom-up, estimate the confidence interval using the training set	Not necessary	High speed and accuracy do not need trim samples	

4.5.3 DF concept rule extraction and matching algorithm

4.5.3.1 Dynamic fuzzy concept rule extraction

The dynamic fuzzy concept tree is constructed by discretizing the attribute values of the sample set. After the necessary pruning, a relatively high-precision dynamic fuzzy concept tree has been built. We can form a set of rules to classify the concept of the objective function, which transforms into a series of dynamic fuzzy rules. Assuming $(\overrightarrow{A}, \overrightarrow{A})$ and $(\overrightarrow{B}, \overrightarrow{B})$ are dynamic fuzzy subsets defined on a finite space $(\overrightarrow{X}, \overrightarrow{X})$, *IF* $(\overrightarrow{A}, \overrightarrow{A})$ *THEN* $(\overrightarrow{B}, \overrightarrow{B})$ is a dynamic fuzzy rule, denoted as $(\overrightarrow{A}, \overrightarrow{A}) \Rightarrow (\overrightarrow{B}, \overrightarrow{B})$. $(\overrightarrow{A}, \overrightarrow{A})$ is called the conditional dynamic fuzzy set and $(\overrightarrow{B}, \overrightarrow{B})$ is the conclusion dynamic fuzzy set. The dynamic fuzzy rules change dynamically in the attribute space, and the dynamic boundaries are fuzzy. The boundary of the concept is determined by the dynamic fuzzy inference mechanism.

The steps for dynamic fuzzy concept rule extraction are as follows:

- (1) According to the pruned DF concept tree structure, we use the root node as the originator of the traversal.
- (2) Every node in the concept tree is traversed depth-first, and then we traverse from the root node to each leaf node (concept class node), forming a conditional conjunctive rule (“..” part).
- (3) The rules of the same conceptual category form a disjunctive paradigm of dynamic fuzzy concepts.
- (4) When the entire dynamic fuzzy concept of the rules set has been generated, the algorithm terminates.

The algorithm may have the same rules as the classical set-based decision tree learning algorithm, but the domain of the attribute values is different. There is a substantial difference. For example, under the classical rules: 1) if the temperature tomorrow is greater than or equal to 10°C, then the relative humidity will be less than or equal to 15%; under the dynamic fuzzy rules: 2) if the temperature tomorrow continues to rise, then the relative humidity will gradually decline (confidence is 97%). As the real world is dynamic and fuzzy, our approach is closer to reality. We can learn the target concept better from this example from the point of view of concept learning.

4.5.3.2 Concept matching algorithm based on DF concept

The concept matching algorithm for the dynamic fuzzy concept is as follows:

- (1) Apply dynamic fuzzification to the test samples.
- (2) For each rule, calculate the membership degree after partial matching with the dynamic fuzzy condition of this test sample.
- (3) If there are multiple rules to divide the test sample into the same concept category, the concept rule with the largest membership degree is taken as the concept category.

- (4) If the test sample is divided into different categories with different membership degrees, the test sample with the highest membership degree is the corresponding concept category.

The algorithm takes the rule with the highest membership degree as factually credible. This reflects the mode of human thinking in understanding a problem and provides better information for mining. Compared with the traditional matching algorithm, the dynamic and uncertain characteristics of the development of the real problem are fully considered in the algorithm. The generated dynamic fuzzy concept tree has strong generality and the division of space is reasonable. The ability to express knowledge is more accurate, and the dynamic concept tree is robust to noise. The overfitting problem is also avoided to the greatest extent.

4.6 Application examples and analysis

This section presents a concrete application example of dynamic fuzzy concept learning theory.

4.6.1 Face recognition experiment based on DF concept lattice

4.6.1.1 Introduction to face recognition

Face recognition is an important part of biometrics and is an identity authentication technique based on human biometrics. The research on face recognition can be



Fig. 4.4: ORL face database (part).

divided into four stages. The first stage is to study the features of the human face, which is the primary stage of face recognition. In the second stage, the face image is represented by geometrical parameters which requires manual participation. This is a human-computer interaction stage. In the third stage, face recognition algorithms such as PCA, LDA, Elastic Graph Matching, independent component analysis, support vector machine (SVM), neural network (NN), hidden Markov model (HMM), and flexible model (FM) are applied. This stage is an active period in the face recognition domain, which realizes automatic machine identification. The fourth stage is based on the consideration of the robustness of face recognition, and considerable research has examined situations where the image is affected by changes in illumination, pose, expression, occlusion, low resolution, and so on. In summary, the main methods of face recognition can be divided into geometric feature-based methods, template-matching methods, Eigen-subspace methods, HMM methods, NN methods, elastic pattern-matching methods, and flexible matching methods [22].

4.6.1.2 ORL face database

In the first experiment, the ORL library is used as the training sample space. The ORL face database consists of 40 individuals, each with 10 images under different lighting, facial expressions, and gestures, giving a total of 400 images. All images have been normalized to 92×112 pixels. The grey level of each image is 256. Figure 4.4 shows part of the sample images from the ORL face database. Each one has three images, and 18 persons have a total of 54 images.

4.6.1.3 Experimental platform and steps

This experiment used the ORL face database and MATLAB 7.0 on a Pentium4 + Intel 2.79 GHz processor with 1 GB of memory. Firstly, the original images were preprocessed. The geometric features of each training sample image were extracted, and the corresponding facial feature vector was constructed. The dynamic fuzzy form background (DF matrix) was then obtained by dynamic fuzzy preprocessing. The dynamic fuzzy facial concept lattice was constructed using the dynamic fuzzy concept lattice hierarchical construction method. A face concept classifier was generated using dynamic fuzzy concept lattice association rule extraction to generate dynamic fuzzy concept rules for facial images. Finally, through the formation of the classifier, the faces were quickly identified. Figure 4.5 shows the flow of the algorithm.

1) Feature extraction

The algorithm selects seven feature points of the face (four corners of the eyes, tip of the nose, and two points at the corners of the mouth). Because the distribution of these feature points has the characteristics of angular invariance, they are relatively easy to collect and measure. According to a known feature extraction algorithm, the integral projection method is used to locate the human face. The grey integral projection is used to locate the face contours and the nose and to position the eyes. Then,

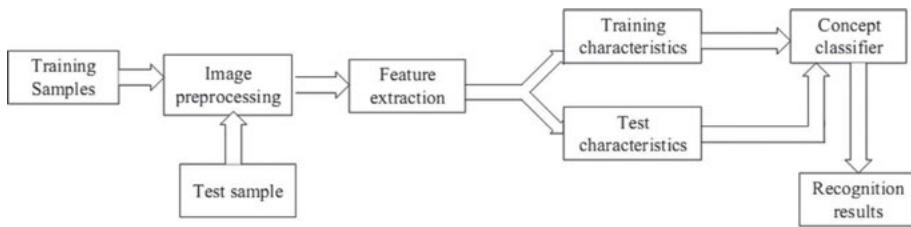


Fig. 4.5: Face recognition flow.

using a priori knowledge to determine the region of the mouth, a vertical integral projection is applied to determine the final position of the mouth. Finally, we compose 10 distance feature values with size, rotation, and displacement invariability that are easily recognized by the computer. They are (\vec{y}_1, \vec{y}_1) : left eye width, (\vec{y}_2, \vec{y}_2) : vertical distance between nose point and centre line of eyes, (\vec{y}_3, \vec{y}_3) : distance between left and right edge of face, (\vec{y}_4, \vec{y}_4) : mouth width, (\vec{y}_5, \vec{y}_5) : the distance between the centre of the eye and the left mouth angle, (\vec{y}_6, \vec{y}_6) : the horizontal distance between the outside of the eyes, (\vec{y}_7, \vec{y}_7) : the horizontal distance between the outside of the right eye and the top of the nose, (\vec{y}_8, \vec{y}_8) : the horizontal distance between the inner corner of the left eye and the top of the nose, (\vec{y}_9, \vec{y}_9) : the vertical distance between mouth and the tip of the middle point, $(\vec{y}_{10}, \vec{y}_{10})$: distance between nose and mouth.

2) Process and results of face recognition

According to the extracted geometric features, the corresponding eigenvectors are constructed and dynamically fuzzified and normalized. Each processed eigenvector is taken as an example of the DF formal background. The set of eigenvectors corresponding to all training examples constitutes a DF formal context. For example, Tab. 4.6 lists the background of some DF forms given by five different faces after feature extraction, dynamic fuzzification and normalization. Taking into account the process of building a dynamic fuzzy face concept lattice to fully rely on the formation of the dynamic fuzzy form background, the DFCL_Layer algorithm is used to construct a dynamic fuzzy face concept lattice, and the appropriate reduction is applied to the dynamic fuzzy face concept lattice. Finally, the dynamic fuzzy face recognition algorithm is extracted by the DFCL_Rule algorithm to generate high-precision the dynamic fuzzy face concept classifier.

When the face recognition classifier is generated, the first five face images of each person are selected as training samples, and the last five face images are the test samples. The recognition results of some face test samples are shown in Fig. 4.6.

The experiment also compared the other algorithms in terms of the recognition rate. The comparison results are shown in Fig. 4.7. It can be seen that, as the number of training samples increases, the recognition rate of the three algorithms

Tab. 4.6: DF form background of partial facial feature formation.

OB	Attributes									
	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
1	0.1961	0.0171	0.4878	0.0716	0.0325	0.2862	0.2016	0.2797	0.5009	0.5024
2	0.0207	0.0933	0.7567	0.0104	0.0104	0.0207	0.0311	0.0104	0.4561	0.1566
3	0.0142	0.0192	0.5126	0.0071	0.0071	0.0285	0.0214	0.0214	0.5909	0.5909
4	0.0116	0.0166	0.7113	0.0816	0.0350	0.2449	0.1749	0.1866	0.4081	0.4088
5	0.0160	0.0920	0.5281	0.0080	0.0080	0.0240	0.0240	0.0160	0.5841	0.5842

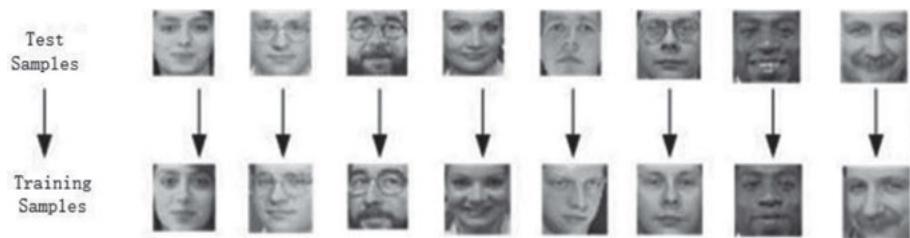


Fig. 4.6: Partial sample recognition result.

improves. PCA + LDA performs a PCA dimensionality reduction on the original image and then uses the LDA algorithm to identify the face image. This algorithm's recognition rate is higher than that of the simple LDA algorithm. The performance of this algorithm is equivalent to that of PCA + LDA algorithm. When the number of training samples exceeds a certain threshold, our algorithm achieves a better recognition rate.

4.6.2 Data classification experiments on UCI datasets

4.6.2.1 Data classification experiments on UCI datasets

The second experiment used some datasets from the UCI-Machine-Learning database (see Tab. 4.7). The test sample set and the original dataset were preprocessed, and the training sample set and the test sample set were mixed. Cross-validation was used to test the constructed DF concept tree and the number of cross-validation directories was set to 10.

Some of the UCI datasets were tested by the C4.5 and DFDT algorithms. Depending on the constructed DF concept tree, pruning may or may not be performed. The number of concept nodes in the generated concept tree (a concept can have multiple concept nodes), the number of tree nodes, the correct number and accuracy of training samples, and the correct number and accuracy of cross-validation were recorded as evaluation metrics. The experimental results of the two algorithms are given in

Tab. 4.7: UCI standard dataset information.

Name	Number of instances	Number of attributes	Number of categories	Whether there is a missing value
Balance-Sca	625	5	3	No
Dermatology	366	34	6	No
Hepatitis	155	19	2	Yes
Monk	124	7	2	Yes
Tic-tac-toe	958	9	2	No
Wine	178	17	3	Yes

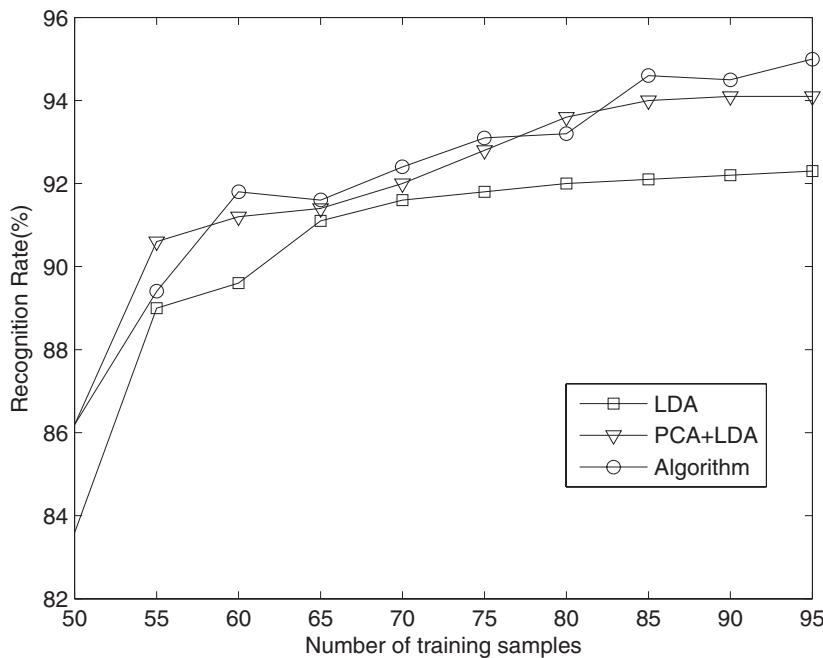


Fig. 4.7: Comparison of the recognition rates of the three algorithms.

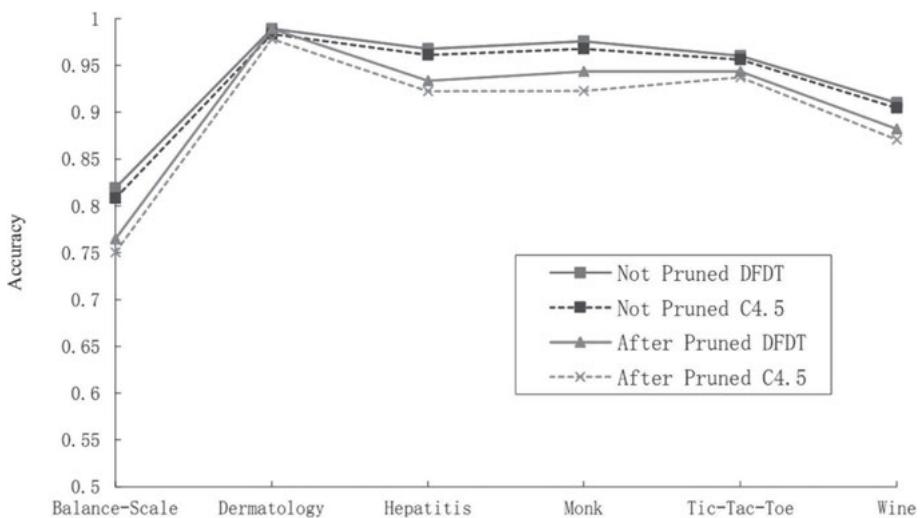
Tab. 4.8: Experimental results of the C4.5 algorithm on the UCI dataset.

Name	Whether pruning	Concept points	Number of tree nodes	Training examples		Cross-validation	
				Correct number	Accuracy	Correct number	Accuracy
Balance-Scale	No	89	111	505	0.8080	433	0.6928
	Yes	33	41	469	0.7504	403	0.6448
Dermatology	No	19	53	360	0.9836	332	0.9071
	Yes	19	37	358	0.9781	334	0.9126
Hepatitis	No	16	31	149	0.9613	125	0.8065
	Yes	11	21	143	0.9226	130	0.8387
Monk	No	15	28	120	0.9678	95	0.7661
	Yes	11	20	115	0.9274	99	0.7984
Tic-Tac-Toe	No	139	208	916	0.9562	828	0.8643
	Yes	95	142	898	0.9374	815	0.8507
Wine	No	15	33	161	0.9045	146	0.8202
	Yes	12	20	155	0.87007	151	0.8483

Tabs. 4.8 and 4.9. Figures 4.8 and 4.9 compare the accuracy of the algorithms based on the two results tables.

Tab. 4.9: Experimental results of DFDT algorithm on UCI dataset.

Name	Whether pruning	Concept points	Number of tree nodes	Training examples		Cross-validation	
				Correct number	Accuracy	Correct number	Accuracy
Balance-sca	No	85	103	512	0.8192	445	0.7120
	Yes	31	36	478	0.7648	416	0.6656
Dermatology	No	18	35	362	0.9891	346	0.9453
	Yes	18	35	362	0.9891	346	0.9453
Hepatitis	No	15	28	150	0.9677	127	0.8149
	Yes	10	19	145	0.9355	132	0.8156
Monk	No	14	28	121	0.9758	97	0.7823
	Yes	10	18	117	0.9435	101	0.8145
Tic-tac-toe	No	135	198	920	0.9603	836	0.8727
	Yes	90	132	904	0.9436	824	0.8601
Wine	No	14	31	162	0.9101	148	0.8315
	Yes	11	18	157	0.8820	153	0.8596

**Fig. 4.8:** Comparison of the classification accuracy of the classification algorithm on the training set.

4.6.2.2 Analysis of results

The concept learning algorithm based on DFDT adopts the dynamic fuzzy preprocessing technique, which fully considers the dynamic fuzziness in the real problem and offers learning ability in the dynamic fuzzy environment. From the cross-validation concept classification accuracy rate, we can see that the prediction performance of concept classification is further improved over that of the C4.5 algorithm. In addition,

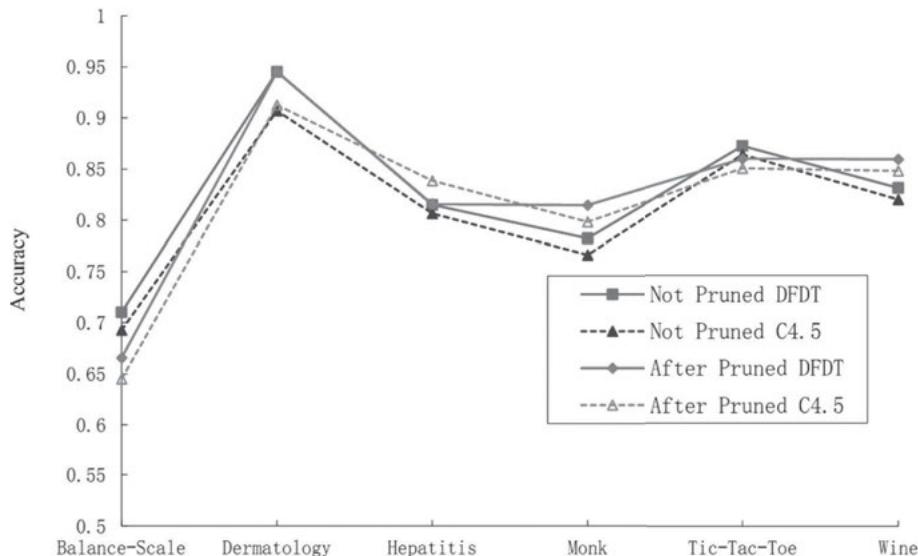


Fig. 4.9: Cross-validation comparison of classification algorithms.

when selecting the best classification attribute, the algorithm takes full account of the contribution of decision attributes to the whole set of decision instances, which can be used to classify the instances of unknown concept classes more precisely. Moreover, the size of the generated dynamic fuzzy leaf node (dynamic fuzzy concept node) and the tree size are smaller those that generated by the C4.5 algorithm. Therefore, the generalization ability of the constructed classifier is better, and the generalization ability has been enhanced.

4.7 Summary

This chapter has discussed the mechanism of a concept learning model based on DFS. According to the theory of DFS data modelling, a DF concept representation model was proposed. The formal representation of DF concepts was described by establishing the mapping relation between a DFD instance set and dynamic fuzzy attribute set (constituting the DFGalois connection).

References

- [1] Mitchell TM. Version space: A candidate elimination approach to rule learning. In Proceedings of the Fifth international joint conference on AI, 1977, 305–310.
- [2] Mitchell TM. Generalization as search. Artificial Intelligence, 1982, 18(2): 203–226.

- [3] Valiant LG. A theory of the learnable. *Communication of ACM*, 1984, 27(11): 1134–1142.
- [4] Duda R, Hart P. *Pattern classification and science analysis*. NY, USA, John Wiley and Sons, 1973.
- [5] Littlestone N. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 1988, 2(4): 285–318.
- [6] Burusco A, Gonzalez RF. The study of the L-fuzzy concept lattice. *Mathware and Soft Computing*, 1994, 1(3): 209–218.
- [7] Chein M. Algorithme de recherche des sou-matrices premières d'une matrice. *Bulletin mathématique de la Société des Sciences Mathématiques de la République Socialiste de Roumanie*, 1969, 13(1): 21–25.
- [8] Bordat J P. Calcul pratique du treillis de Galois d'une correspondance. *Mathématiques Informatiques Et Sciences Humaines*, 1986, 96(96): 31–47, 101.
- [9] Ganter B. Two basic algorithms in concept analysis. *Lecture Notes in Computer Science*, 2010, 5986: 312–340.
- [10] Nourine L, Raynaud O. A fast algorithm for building lattice. *Workshop on Computational Graph Theory and Combinations*, Victoria, Canada, 1999.
- [11] Godin R. Incremental concept formation algorithm bases on Galois (concept) lattices. *Computational Intelligence*, 1995, 11(2): 246–267.
- [12] Carpineto C, Galois RG. An order-theoretic approach to conceptual clustering. In *Proceedings of ICML-93*, 1993, 33–40.
- [13] Ho TB. Incremental conceptual clustering in the framework of Galois lattice. In Lu H, Motoda H, Liu H, eds. *KDD: Techniques and Applications*. Singapore: World Scientific, 1997, 49–64.
- [14] Oosthuizen GD. Rough Sets and Concept Lattices. *International Workshop on Rough Sets and Knowledge Discovery: Rough Sets, Fuzzy Sets and Knowledge Discovery*. London: Springer-Verlag, 1993, 24–31.
- [15] Agrawal R, Imielinski T, Swami AN. Mining Association Rules between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, 1993, 207–216.
- [16] Ho TB. An Approach to Concept Formation Based on Formal Concept Analysis. *Ieice Transactions on Information & Systems*, 1995, 78(5): 553–559.
- [17] Qiang Y, Liu ZT, Lin W, et al. Research on fuzzy concept lattice in knowledge discovery and a construction algorithm. *Acta Electronica Sinica*, 2005, 33(2): 350–353.
- [18] Shi ZZ. *Knowledge discovery*. Beijing, China, Tsinghua University Press, 2002.
- [19] Huan L, Farhad H, Manoranjan D. Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, 2002, 6(4): 393–423.
- [20] Mao GJ, Duan LJ, Wang S, et al. *Data mining principles and algorithms*. Beijing, China, Tsinghua University Press, 2005.
- [21] Witten IH, Frank E. *Data mining practical machine learning tools and techniques with Java implementations*. Beijing, China, China Machine Press, 2003.
- [22] Liu XG. Research on the concept learning based on DFS. *Master's Thesis*, Soochow University, Suzhou, China, 2011.

5 Semi-supervised multi-task learning based on dynamic fuzzy sets

This chapter is divided into six sections. Section 5.1 gives a brief introduction to this chapter. We present the semi-supervised multi-task learning model in Section 5.2 and the semi-supervised multi-task learning model based on DFS in Section 5.3. In Section 5.4, we introduce algorithm of dynamic fuzzy semi-supervised multi-task matching. The DFSSMTAL algorithm is described in Section 5.5. The last section is the summary of the chapter.

5.1 Introduction

5.1.1 Review of semi-supervised multi-task learning

It is well known that supervised learning is an effective technique for learning classifiers when there is sufficient mark-up data. However, there is often a high price to obtain the tags for data. Thus, it can be difficult to obtain sufficient tag data. Moreover, even if the classification error rate of supervised classifiers trained from a small amount of tag data is zero, the classifiers still lack good generalization performance. Ensuring the effective use of labelled and unlabelled data for learning, has become a major concern.

1. Overview of semi-supervised learning

In many machine learning problems, we need tag data to train a model. However, sometimes, the marked data are limited because manual marking is often tedious or the cost of acquiring data is high. As a result, the models that are being trained are unsatisfactory. In recent years, numerous scholars have attempted to improve the generalization performance of classifiers using information sources other than marker data. These studies can be divided into two categories according to different information sources: semi-supervised learning [1–5] and multi-tasking learning [6–10]. Additional information for the former comes from data samples, where sufficient sample information is included in the data without markers; the latter uses related tasks to obtain additional information. We can consider semi-supervised learning as an extension of the traditional supervised learning; that is, semi-supervised learning is mainly concerned with obtaining good generalization ability and performance in learning machines that lack partial information from the training data. Semi-supervised learning has aroused the interest of scholars in terms of building better classifiers with a lot of unlabelled data, reducing manual tasks and material consumption, and the performance of machine learning. Currently, semi-supervised learning methods include generative models, self-training, co-training,

multi-angle learning, low-density separation (such as transductive SVM), and graph-based approaches.

There are different advantages and disadvantages to these methods. For example, generative models have the longest history. Mixed models must be identified and correct, otherwise the learning will not be carried out smoothly, and the performance of the classifier will be damaged to a certain extent. Self-training, another familiar semi-supervised learning algorithm, is the simplest encapsulation algorithm but has the disadvantage of the packaging effect. The performance of the algorithm relies heavily on the supervisory method of its use, and errors early in the training stage will be enhanced in later periods, leading to a decrease in performance. Overall, in semi-supervised learning, good learning performance requires model assumptions that are consistent with the problem structure, and compared with general supervised learning, this approach often requires more energy to design features or similar functions.

2. Overview of multi-task learning

At present, most machine learning techniques are designed for a single problem or task. However, many problems are highly interrelated. For example, a tennis player can easily learn other sports such as badminton and table tennis; the future performance of students can be predicted according to their existing results; robots can be trained to identify the different characteristics of the face; and so on. In the field of machine learning, this is known as multi-task learning. Multi-task learning considers related tasks and shares information from relevant tasks to improve the learning effect of target tasks.

This production method is closely related to human learning. When people learn new knowledge, they will consciously or unconsciously learn from similar past learning experiences. Psychologists call this learning transfer. It is possible to improve the performance of the learning system by using valuable information from related tasks, which is the main reason that multi-task learning has become an important branch of machine learning.

Over the past decade, many multi-task learning algorithms have been proposed. For example, the multi-task feature learning in [11] uses regularization to learn the same representation of all tasks. The regularized multi-task SVM method in [12] requires the SVM parameters in all tasks to be similar and successfully introduces the traditional SVM method into the multi-task learning domain. The task clustering method in [13, 14] divides all tasks into different clusters and then learns the similar or common expressions of all tasks in the same cluster. The multitasking learning method in [15], [16], [17], [18] is a kind of learning method based on the Gaussian process, and the Gaussian process is the basic model of multitasking.

Most existing multi-task learning methods assume that tasks are interrelated. Thus, a subset of tasks has similar or identical data characteristics and model parameters. For example, both neural network-based approaches and multi-task

feature learning assume that all tasks have the same data characteristics. Regularized multi-task SVM [12] and [14] assume that all tasks or tasks in the same cluster have similar model parameters.

3. Overview of semi-supervised multi-task learning

The common goal of semi-supervised learning and multi-task learning is to use additional information to improve the learning performance of supervised learning tasks. This extra information may come from unlabelled data or other related tasks. Therefore, it is very reasonable to integrate both types of learning to constitute a new learning framework.

In fact, in recent years, some scholars have conducted studies in this regard. Although the method proposed by Ando and Zhang [19] is mainly intended to improve the performance of semi-supervised learning, they laid the foundations for the study of the relationship between semi-supervised learning and multi-task learning. Starting from a single target task and unlabelled data, they use unlabelled data to construct more unrelated tasks to assist in learning the target task. Liu et al. proposed a semi-supervised multi-task learning framework [20–22], which can be regarded as the first successful integration of semi-supervised learning and multi-task learning into a learning framework. The semi-supervised learning method used in this new framework is Parametric Neighbourhood Classification; it is composed of M semi-supervised classifiers and a joint prior distribution with all classifier parameters.

Each classification task has a corresponding classifier. In this integrated framework, M tasks are classified at the same time.

The semi-supervised multi-task learning framework proposed by Li et al. has led to a boom in semi-supervised multi-task learning in the field of machine learning. The current semi-supervised multi-task research has two main categories: parameter estimation methods and cost minimization techniques.

1) Parameter estimation method

The parameter estimation method is described in [20–24]. Liu et al. constructed a parameter classifier (called a basic classifier) for each classification task in [20, 21]. They use the joint distribution of these parameters and the posteriori maximization method to estimate the corresponding classifier parameters and then complete the multi-task learning. Because the classifier is linear, [22] extends the basic classifier (mainly using the kernel method) to make it suitable for nonlinear classification. Applying Dirichlet process (DP) in the context of multi-task learning is a complex task. Thus, a simple form of DP is selected that obtains a solution to using EM. This multi-task framework effectively integrates semi-supervised learning. However, this method randomly selects the marker data without validation. Using this idea, active learning is applied to the semi-supervised multi-task learning framework in [23], whereby active learning is employed to select the most informative marker data.

The general idea of the semi-supervised multi-task learning algorithm based on the parameter estimation method can be summarized as Algorithm 5.1.

Algorithm 5.1 Semi-supervised multi-task learning based on the parameter estimation method

Input: Training Set of M Tasks (include labeled data and unlabeled data);

Output: The parameters neighborhood basic classifier parameters of each task, take the m th basic classifier parameters as θ_m .

(1) Take θ as parameter in each task, the probability that takes the class of sample point x_i as y_i :

$$p^*(y_i | x_i, \theta)$$

(2) Take the probability of what take t steps neighborhood of the sample point x_i as $\mathcal{N}_t(x_i)$ and have

$$p(y_i | \mathcal{N}_t(x_i), \theta) = \sum_{j=1}^n b_{ij}^{(t)} p^*(y_i | x_i, \theta)$$

where $b_{ij}^{(t)}$ is the probability that x_i is x_j 's t steps neighborhood

(3) Joint likelihood function of M tasks:

$$\begin{aligned} & p(\{y_i^m, i \in \mathcal{L}_m\}_{m=1}^M | \{\mathcal{N}_t(x_i^m), i \in \mathcal{L}_m\}_{m=1}^M, \{\theta_m\}_{m=1}^M) \\ &= \prod_{m=1}^M p(y_i^m, i \in \mathcal{L}_m | \{\mathcal{N}_t(x_i^m), i \in \mathcal{L}_m\}, \theta_m) \\ &= \prod_{m=1}^M \prod_{i \in \mathcal{L}_m} \sum_{j=1}^{n_m} b_{ij}^{(t_m)} p^*(y_i^m | x_i^m, \theta_m) \end{aligned}$$

where \mathcal{L}_m is index collection of labeled data in the m -th task

(4) Use maximum a posteriori estimation to solve the M parameters θ

The classification problem considered in [20–23] does not involve regression. In fact, many examples can be designed as semi-supervised and multi-task regression problems. For instance, in the individual posture prediction application, each task corresponds to a posture prediction for individuals, and each person has a large number of pictures of unknown posture information. Thus, semi-supervised regression and multi-task regression are properly integrated in [24]. This forms a semi-supervised multi-task regression framework. On the basis of the parameter estimation method, a Supervised Multitasking Regression Framework is first proposed based on Gaussian processes, with the kernel parameters of all tasks assumed to have the same Gaussian prior. The Gauss prior kernel function is then modified to have a data-dependent kernel function; combined with the use of unlabelled data, this ultimately forms a semi-supervised multi-task regression framework.

2) Cost minimization method

Loeff et al. [25] and Wang et al. [26] described typical applications of cost minimization. In [25], multi-task learning and semi-supervised learning are integrated into a new model of extended semi-supervised multi-task learning through the addition of a regular term. The new provisions of the regularization encourage the

sharing of characteristics between the classifiers, thus improving the generalization performance; on the other hand, it helps spread neighbourhood tag information and the effective use of unlabelled samples to learn. The attributes of the original model are unchanged by the new regular items: the cost function is still convex. Therefore, the global optimal value can be obtained using the gradient descent method.

In general, multi-task learning studies assume that tasks are interrelated. In [26], the author proposes a different approach. He assumes that tasks can form different groups and that tasks in each group should have similar classification rules. In fact, [13] and [27] describe multi-task research under this assumption, but they are monitoring methods rather than semi-supervised approaches. In [26], the author constructs a linear classifier for each task according to the data manifold of the potential task and then integrates all the classification vectors with the K-means inter-class regularization term, which forms a semi-supervised multi-task learning system as in Algorithm 5.2.

Algorithm 5.2 Semi-supervised multi-task learning system

Input: Training Set of M Tasks (include labeled data and unlabeled data);

Output: Assume classifier of the t-th task is $f_t(x) = w_t^T x$, x is the sample of task t, $f_t(x)$ is the label of sample, w_t is the weight. Then the main need to determine the algorithm is the M tasks by the weight of the classifier matrix W .

(1) List the cost function of the problem. In general, the cost function consists of two parts: general cost and regular item. The general cost can be a common loss function, while the regular item is set to complete the semi-supervised multi-task learning. Thus, the cost function is

$C(W) = \mathcal{L}(W) + \mathcal{R}(W)$, $\mathcal{L}(W)$ is general cost function, $\mathcal{R}(W)$ is the regular item.

(2) The whole semi-supervised multi-task problem is the need to solve:

$$\arg \min_W C(W)$$

Because the cost function is a convex optimization problem, it can be solved by the conventional method.

3) Other semi-supervised multi-task learning research

In addition to the above two categories, the study of semi-supervised multi-task learning in theory and application has also made new progress in recent years. In application, Chen et al. broke through the usual supervised learning or unsupervised learning in [28]; they use semi-supervised model to integrate a multi-task learning method – rotation structure optimization algorithm – and successfully apply semi-supervised multi-task learning to semantic relations between phrases. On the theoretical side, Michelangelo et al. studied semi-supervised kernel machines in [29] and relied on multi-task learning patterns. Each task is related to individual predicates in feature space. And the deep abstraction is composed of the first-order logical clauses with these predicates, so that the semi-supervised multi-task learning framework can successfully integrate the first-order logical clauses theory with the nuclear machine.

5.1.2 Problem statement

In engineering applications, we often need to model a system, and we require a reliable model to describe the correct relationship between the input and output of the application system, such as performance monitoring, error detection, problem optimization, and multi-group classification. Time series prediction is an extremely important and practical problem that is related to economics, business decision making, signal processing, and control. Prediction is a dynamic fuzzy process that can be predicted as a dynamic fuzzy object. DFD are an important feature of this type of problem.

In contrast, the current accepted concept of machine learning is still Simon's explanation of learning: "If a system can improve its performance by performing a certain process, that is learning". This statement implies that learning is a process, that learning occurs in terms of a system, and that learning can change the performance of the system. "Process", "system", and "change performance" are the three main points of learning. If we analyse this carefully, we can see that these three points have dynamic ambiguity: the nature of the learning process has a dynamic fuzzy character; system changes may be good or bad, and so are essentially dynamic and fuzzy; and changes in system performance, results, etc., have dynamic fuzzy characteristics.

Therefore, it can be seen that dynamic fuzzy information is common in practical applications and theoretical research. To establish a machine learning method that offers substantial progress and can meet the main points of Simon's argument, the key problem is to solve the dynamic ambiguity problem arising from learning activities effectively.

Although adaptive model can predict the performance of a dynamic system, they cannot handle dynamic fuzzy problems. Rough set theory is based on fuzzy sets and can effectively solve fuzzy problems but cannot handle dynamic fuzzy problems. Statistical learning theory is based on small-sample statistics, which solve static problems but are insufficient for dynamic problems. Reinforcement learning based on the Markov process theory can solve dynamic problems but is unsuitable for fuzzy problems. Therefore, there is an urgent need for a theory that effectively solves dynamic fuzzy problems in the field of machine learning.

To solve dynamic fuzzy problems in engineering applications and machine learning activities, dynamic machine learning models have been developed based on dynamic fuzzy sets and related algorithms [30, 31]. Further work on this basis has proposed a dynamic fuzzy machine learning method, a dynamic fuzzy machine learning model, and related algorithms from the perspectives of algebra and geometry [32–34]. In [35], dynamic fuzzy set theory is applied to the semi-supervised multi-task learning domain. A dynamic fuzzy semi-supervised multi-task learning (DFSSMTL) algorithm provides a new theoretical method for solving dynamic fuzzy problems. The main work of this chapter is to apply the theory of dynamic fuzzy sets to

the semi-supervised multi-task learning neighbourhood. We will elaborate on the following aspects:

- (1) For dynamic fuzzy problems, how can we design a reasonable and effective semi-supervised multi-task learning model using dynamic fuzzy set theory?
- (2) Based on the theory of dynamic fuzzy sets, how can we integrate the existing semi-supervised approach and multi-task method more effectively to form a new semi-supervised multi-task learning framework?

5.2 Semi-supervised multi-task learning model

5.2.1 Semi-supervised learning

Traditional machine learning techniques typically use only labelled sample sets (supervised learning) or only unlabelled sample sets (unsupervised learning) for learning. In most practical problems, however, labelled samples coexist with unlabelled samples. To make better use of these data, semi-supervised learning was developed. In recent years, semi-supervised learning technology has become a hot topic of research in the machine learning field.

1. Concept of semi-supervised learning

Traditional machine learning techniques can be divided into two categories: unsupervised learning and supervised learning.

General unsupervised learning data are as follows:

Suppose the data sample set X contains n sample points $X = \{x_1, x_2, \dots, x_n\}$, where $x_i \in X, \forall i \in \{1, 2, \dots, n\}$. It is generally assumed that the data are independent and identically distributed. The purpose of unsupervised learning is to estimate the density distribution of X according to these sample points. Clustering and dimensionality reduction are typical representative techniques.

Supervised learning differs from unsupervised learning in that the data contains not only sample points but also category labels that can be used to describe a sample point (x, y) , where y is the class label of sample point x . The purpose of supervised learning is to learn a mapping from sample points to category labels from these data samples. When the category label is a real number, the supervised learning technique is called regression; otherwise, it is classification. SVM is a typical supervised learning technology.

Semi-supervised learning is a learning method for studying how computers and natural systems (e.g. people) learn from unlabelled data and labelled data. It is a learning framework between supervised learning and unsupervised learning. Its dataset $X = \{x_1, x_2, \dots, x_n\}$ ($n = l + u$) is divided into two blocks: one is labelled data $X_l = \{(x_1, y_1), \dots, (x_l, y_l)\}$, where the class label of the sample point x_i is y_i ; the other set consists of unlabelled data $X_u = \{x_{l+1}, \dots, x_{l+u}\}$, where the class label of the data is unknown. As there are generally more unlabelled data than labelled data,

we assume that $u \gg l$. Semi-supervised learning is widely studied, partly because of its practical value in constructing computer algorithms and partly because it can help people understand the theoretical value of machines and how they learn.

2. Unlabelled data in semi-supervised learning

We can indeed learn useful information from unmarked data. It may sound magical, but it is no miracle that the model assumptions are a good match for the structure of the problem. Although people do not need to spend the same effort to mark the training data in semi-supervised learning, it is important that excellent and reasonable models, features, cores, and similar functions are provided. These efforts are particularly critical in compensating for the lack of mark-up data compared to supervised learning.

However, there is no free lunch: unmarked data are not always useful. Elworthy realized that training HMMs with unlabelled data reduces the model accuracy under certain initial conditions [38]; Cozman analysed the semi-supervised learning performance of a mixed model and found that unlabelled data sometimes introduce an increase in categorical errors – the phenomenon of degeneration was analysed from a mathematical point of view [39]. The reason for the unhelpfulness of these unmarked data may lie in the fact that the model assumptions are not well matched with the problem structure. For example, many semi-supervised learning methods assume that decision boundaries should be far away from high-density regions, such as transductive SVMs (TSVMs), information regularization, noiseless Gaussian processes, and graph-based approaches (the weight of the graph is determined by the distance between data). If the data come from two highly covered Gaussian distributions, the decision boundary will be located in the densest place, and learning by the above method will give poor results. Other semi-supervised learning algorithms such as EM generate a hybrid model that can easily solve such problems, but the pre-detection of poor matches is difficult, and this field has many open problems.

Semi-supervised learning methods use unlabelled data primarily to modify or re-differentiate assumptions obtained from the marked data. So, how does semi-supervised learning use unlabelled data? To facilitate a description, we use a probabilistic method. Assuming that $p(y|x)$ and unmarked data $p(x)$ are known, the joint distribution $p(x, y)$ in the production model has the same parameters. Obviously, $p(x)$ can affect $p(y|x)$; the EM mixed model belongs to this category, and to some extent, self-training is the same. Many other methods are discriminant techniques, such as TSVMs, information regularization, noiseless Gaussian processes, and graph-based approaches. The original discriminant training method cannot be used for semi-supervised learning, because $p(x)$ is ignored in estimating $p(y|x)$. To this end, $p(x)$ dependencies are often introduced into the objective function, and these objective functions often assume that $p(y|x)$ has the same parameters as $p(x)$.

3. Basic assumption in semi-supervised learning

Most current machine learning techniques are based on the independent and identical distribution hypothesis; that is, the data samples must be independently sampled from the same distribution. Obviously, it is not possible to generalize from a finite training set to an unknown test set without making certain assumptions.

The following are the most common assumptions in semi-supervised learning.

1) Semi-supervised smoothness assumption

The semi-supervised smoothness assumption states that the labelled function is smoother in the high-density region than in the low-density region. That is, if two sample points in the high-density region are very similar, their corresponding marks are also very similar. This means that if two sample points are connected by a high-density path (for example, the two points belong to the same cluster), then their output signatures are likely to be similar. Otherwise, if they are separated by a low-density region, their outputs need not be close.

2) Cluster assumption

The Cluster Assumption states that sample points in the same cluster are likely to have the same category marker. However, this does not imply that each class forms a single, compact cluster; it simply means that there are no two different classes of samples in the same cluster. In fact, if the cluster is regarded as a set of sampling points that are connected by a short curve over a high-density region, it can be said that the clustering assumption is a special case of the semi-supervised smoothing assumption.

An equivalent argument to the cluster assumption is known as low-density separation, which argues that the decision boundary should be located in a sparsely populated data area. This is because, if the decision boundary crosses an area with more data points, it is possible to classify the sample points in a cluster into different categories.

3) Manifold assumption

A different and related assumption is the manifold hypothesis, which forms the basis of many semi-supervised learning algorithms. This states that high-dimensional data have low-dimensional properties.

The manifold assumption effectively avoids the curse of dimensionality suffered by many statistical methods and learning algorithms. As the dimension increases, the amount of data grows exponentially, and the number of samples required for statistical problems such as density estimation increases exponentially. A related problem at high dimensions is that the distance between the data tends to be similar and loses its expressiveness, which will heavily influence the discriminant method. If the data has low-dimensional characteristics, then the learning algorithm can be implemented in the corresponding low-dimensional space, thus avoiding the dimension disaster.

In fact, we can see from previous analysis that algorithms based on the manifold assumption can be regarded as an approximate realization of the semi-supervised

smoothness assumption, because these algorithms use the manifold metric to calculate the geodesic distance. Moreover, if we consider manifolds as approximations of high-density regions, the semi-supervised smoothing assumption at this point degrades to the smoothing assumption applied to manifolds.

4. Common methods of semi-supervised learning

Many of the above-mentioned assumptions are included in many approaches. According to the hypothesis, semi-supervised learning methods can be divided into the following categories: generative models, low-density division, and Graph-based approaches.

1) Generative models

Generative models are a common approach to semi-supervised learning. The method of generating the model involves estimating the conditional probability $p(x|y)$, where any additional information on $p(x)$ is useful. For example, suppose $p(x|y)$ is a Gaussian distribution. Then, the EM algorithm can be used to find Gaussian parameters for each category. In fact, the only difference between the EM algorithm for clustering and the standard EM algorithm is that the “hidden variables” of any labelled sample are not invisible but are known and are the class labels of the sample. As each given cluster belongs to only one class, this implements the clustering assumption.

One of the advantages of generative models is that they can contain structural knowledge of problems or data through modelling. However, studies have found that, when the model assumptions are wrong, unlabelled data will reduce the prediction accuracy. In statistical learning, a class of functions or a priori data is chosen prior to reasoning and can be selected according to knowledge of the known problems in advance. In semi-supervised learning, if some information about the objective function is known in advance, the a priori data can be more accurately selected after using the unmarked data. In general, higher prior probabilities can be assigned to functions that fit the clustering assumptions. In theory, this is a natural way for semi-supervised learning to obtain boundaries.

2) Low-density division

Some methods directly implement low-density partitioning assumptions by moving the decision boundaries away from unlabelled data. The most common way to achieve this goal is to use a maximum interval algorithm, such as an SVM. The maximum separation method for the labelled data and the unlabelled data is the TSVM. However, the problem is non-convex, so it is difficult to optimize.

Chapter 6 of Reference [1] provides an optimized algorithm for TSVM. The SVM is used to train the labelled data and mark the unlabelled data, and then it re-trains the SVM on all data samples. It repeatedly and gradually increases the weight of the unlabelled data. Another approach uses semi-definite relaxation procedures [1, Chapter 7]. Compared with TSVM, this method seems to implement the low-density partitioning hypothesis more directly because it takes advantage of an extra term that

reflects the density of neighbouring decision boundaries and the standard quadratic regularization term. For details, please see Chapter 10 of Reference [1].

3) Graph-based approaches

In recent years, graph-based approaches have become a hot topic in the field of semi-supervised learning. They all use graph nodes to represent the data, with the edges of the graph representing the distance between the nodes (where there is no edge between two nodes, the distance is infinite). The distance between two data samples is calculated by minimizing the path distance of all paths between two points. From the manifold data sampling point of view, this can be seen as the geodesic distance between two data points. Therefore, it can be said that graph-based methods are constructed on the basis of the manifold assumption.

4) Other methods

There are other methods that are not semi-supervised in nature and are generally learned in two steps.

First, an unsupervised learning method is performed on all data, ignoring existing data tags. This represents a change in the method or a limitation of the new metric or new kernel.

Second, it ignores the unlabelled data and uses the new distance as a new representation method or new kernel for general supervised learning. It can be said that these methods directly implement the semi-supervised smoothing assumption, because the representation method is changed by keeping the fine distance in the high-density region.

Obviously, it is confusing to choose from a number of semi-supervised methods. However, there is no clear way to solve this problem. Because the labelled data are limited, the semi-supervised learning method makes a strict model hypothesis.

In the ideal scenario, the methodological model should fit the structure of the problem, but this is not easy in practice. However, we can use the following method to determine which semi-supervised approach to use. If the data in the class is clustered, the EM-generated hybrid model is a good choice; if the features can be naturally divided into two sets, then co-training is a good choice; if two data samples with similar characteristics tend to be in the same class, then a graph-based approach may be used; if you have used SVM, then TSVM is preferred. The current supervised classifier is complex and difficult to modify, so self-training is a good encapsulation method.

5. Inductive semi-supervised learning and transductive semi-supervised learning

There are many semi-supervised learning methods based on different hypotheses, but these can be broadly divided into two blocks: inductive semi-supervised learning and transductive semi-supervised learning. In the supervised learning classification, the training samples are all labelled. Therefore, people are more concerned about

the classification performance of the model in the test data. In the semi-supervised learning classification, the training sample contains unlabelled data, so there are two different goals. We call the former inductive semi-supervised learning and call the latter transductive semi-supervised learning, or simply transductive learning.

1) Inductive semi-supervised learning

For a given training example $\{(x_i, y_i)\}_{i=1}^l, \{x_j\}_{j=l+1}^{l+u}$, inductive semi-supervised learning involves learning a function $f : \chi \rightarrow y$ such that it is possible to predict unknown data other than $\{x_j\}_{j=l+1}^{l+u}$. Similar to supervised learning, the performance with the unknown data is estimated using test samples $\{(x_k, y_k)\}_{k=1}^m$ that are not trained.

2) Transductive learning

For a given training example $\{(x_i, y_i)\}_{i=1}^l, \{x_j\}_{j=l+1}^{l+u}$, transductive learning involves training a function $f : \chi^{l+u} \rightarrow y^{l+u}$ such that f gives a good prediction for unlabelled data $\{x_j\}_{j=l+1}^{l+u}$. Note that f is only defined for a given training sample, and does not predict extra data. Therefore, it is just a simple function.

To facilitate a better understanding, inductive semi-supervised learning can be analogized into classroom tests. The test questions are unknown in advance, so the students need to prepare for all possible problems. Conversely, transductive learning is analogous to homework, where students know the test questions and do not need to prepare for other questions.

5.2.2 Multi-task learning

Traditional machine learning methods learn only one task at a time. If the problem is more complicated, it can be split into several relatively simple and logically independent subproblems. These subproblems can be learnt separately and then integrated to form the solution to the original problem. Sometimes, the effect of this approach will backfire because it ignores the many potential problems in real rich information. The information is included in the training signals of other tasks in the same field. In 1997, “Multitask Learning” (published in Machine Learning by Caruana) made “multi-task learning” a hot issue in machine learning.

1. Related tasks

The formal or heuristic description of what constitutes related tasks remains the most important open problem in the field of inductive transfer. Because of the lack of an accurate definition of task relatedness, inductive migration theory is progressing slowly. However, some of the characteristics of the theory of task relatedness are very clear.

- (1) If two tasks have the same function, but the noise introduced into the task is independent, then the two tasks are related.

- (2) If two tasks predict different aspects of the same person's health, these tasks are certainly more relevant than those that predict different aspects of human health.

It should be noted that this is not supposed to determine that tasks are relevant simply because they are able to cooperate when being trained at the same time. For example, noise is sometimes added to the extra output of a backpropagation network. Other outputs in the hidden layer serve as regular terms to improve the generalization performance, but this does not mean that the noise task is related to other tasks.

2. Concept of multi-task learning

The standard machine learning method is to learn one task at a time. If the scale of the problem is large, then the problem will be refined into a number of separate, simple questions. These simple problems are individually learnt and then merged to solve complex problems. However, this practice does not always work because it ignores the rich information about training signals for other problems in the same field.

Considering the number of known training patterns and training time, a network with 1000×1000 input retina cannot learn to recognize complex objects in the real world. If you learn a lot of things at the same time, however, then the learning effect will be improved. If tasks are allowed to share what they have learned, then people will feel that learning together is much easier than learning alone. Therefore, if we train a network to recognize contours, shapes, edges, regions, sub-regions, textures, reflections, highlights, shadows, orientations, sizes, and distances of objects, then the trained model is more likely to identify the real world of complex objects. This approach is multi-task learning. Multi-task learning is an inductive migration method that improves the generalization performance using domain information contained in the relevant task training signal as the inductive bias. It improves the generalization performance by learning multiple learning tasks in parallel, and these learning tasks have the same representation, that is, they are related. Each task learns information to help train other tasks. In practice, the training signal of other tasks is an inductive bias.

In short, the core idea of multi-task learning is that a core issue is accompanied by some related issues that help the core results of the problem.

In general, typical information migration methods include the sharing of a hidden model in neural networks [6, 40, 41], setting the same priors in a hierarchical Bayesian model [16, 42–44], sharing parameters in the Gaussian process [15], learning the best distance metric in the KNN [45], sharing the same structure in the prediction space [19], and the regularization of structures in the kernel method [46].

3. Common applications in multi-task learning

In the nine fields discussed below, it is often possible to obtain training signals for useful additional tasks. In general, most real problems belong to one or more of these nine areas [6]. Only a few of the resources in machine learning are multi-task learning

problems. This may sound surprising, but many problems in the field of machine learning are dealt with in terms of single-task learning, thus reducing the chances of using multi-task learning.

1) Use future information to predict the current

People often obtain some important features when they make predictions. However, if we cannot get these features at runtime, they cannot be used as input. If learning is conducted, we can collect these features for the training set and use them as an additional instance of multi-task learning. When using the system, the prediction made by the learner for these additional tasks is likely to be ignored. The primary role of prediction is to provide additional information for the learner during training.

One application that uses future information for learning is medical crisis prediction, such as in a pneumonia crisis [6]. In this problem, lab data from the training set are used as the additional output task. In fact, these laboratory data are not available when making predictions for patients. The useful information contained in these future measures makes the network biased towards hidden layers that are better able to predict crises from features that are available at runtime.

2) Multiple representations and metrics

Sometimes, it is difficult to obtain all the important information from an error metric or an output representation. The use of multi-tasking can benefit from these aspects when the optional measures or outputs represent different aspects of the problem and are useful.

3) Time sequence prediction

This type of application uses future information to predict a current subclass. The future task is the same as the current task, but occurs later. The simplest way to use multi-task learning for time series prediction is to use a single network with multiple outputs, each corresponding to the same task at different times. Figure 5.1 shows a multi-task learning network with four outputs. If output k represents the prediction at time T_k , then the network makes predictions for the same task at four different times. The output that is normally used for prediction is currently in the middle, so that earlier or later tasks can be trained on the network. When the input feature temporarily “skips” the input, we can collect and combine the output from a prediction sequence.

4) Use an inoperable feature

It is not practical to use certain features at runtime. This is either because they are computationally expensive or because they require human assistance that cannot be provided in time. We often incur high costs to obtain more features because the training set is too small. It is practical to compute inoperable eigenvalues for the training set, which can be used as an output for multi-task learning.

As in scene analysis, humans are often required to mark important features. Generally, when a learning system is used, there is no need for a manual annotation, but this does not mean that people cannot use the characteristics that have been marked.

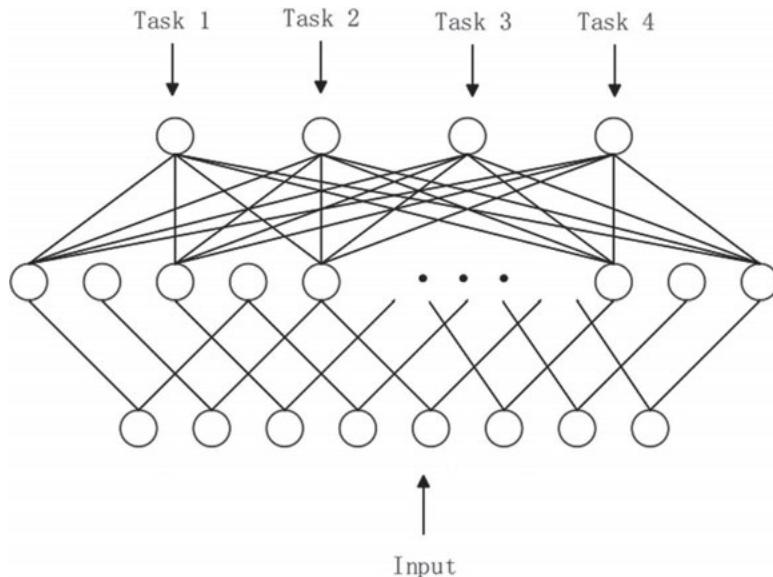


Fig. 5.1: A simple multi-output network.

If these markers are included in the training set, they can be used as additional tasks for the learner but are not required when using the system.

5) Use additional tasks to focus attention

Learners often learn to use large and common input patterns, while ignoring smaller or less common inputs that may also be useful. Multi-task learning can be used to force the learner to focus on patterns that are ignored in the input. The task of relying strictly on input patterns that may be overlooked is supported by forcing learners to learn internal representations. As in the road-following field [6], single-task learning networks often ignore road markings in learning to drive. This is because road marking are often a small part of the image, and do not always exist.

If a learning-to-drive network is required to identify road boundaries and serve additional output tasks, the network will learn to pay attention to the road boundaries in the picture. Road borders are learnable, so the network learns the internal representation to support them. As the network also learns to drive with the same hidden layer, the driving task can use any part of the data that hides the way the road is useful for driving.

6) Sequential transfer

Sometimes, we learn from the prior study of related tasks, but this cannot be used to train the model data. At the same time, multi-task learning can benefit from a priori learning without training data. People can use this model to generate simulation data and use the training signal in the simulation data for multi-task learning.

This approach of sequential transfer avoids catastrophic interference problems (i.e. forgetting the old task when learning a new task – “the grass is always greener”). Sequential transfer can also be used where analytical field theory is not applicable, and these field theories are used by other sequential transfer methods.

7) Multiple tasks appear naturally

Some related tasks are widely encountered in everyday life. The separation of these tasks into separate problems and the traditional way of training these issues alone is often counterproductive. If you train related tasks together, then they can benefit from each other. An early and accidental use of multi-tasking in a backpropagation network was NETtalk, which learned phonemes and stresses for a speech synthesizer to pronounce input words. NETtalk uses a network with multiple outputs, in part because the goal was to control the synthesizer, which requires both phonemes and pressure. Although the researchers did not analyse the contribution of multi-tasking migration to NETtalk, there is already evidence that it may be more difficult to use a separate learning network.

8) Quantitative smoothing

There is a lot of quantitative information in everyday life. For example, training signals may be assigned a level (superior, good, medium, or poor), or some natural process may quantify potentially smoother functions, such as finite predictive physical measurements leading to the patient’s end result – live or die. Although quantification sometimes makes the problem easier to learn, it makes learning more difficult.

These additional training signals may be used as additional tasks if the primary task is quantified better than the additional training signals, or if the two use different quantization methods. It is useful to learn something that is poorly quantified by the extra task because, sometimes, this is easier to learn (because it is smoother). The additional tasks that are not smooth but use different quantization processes are sometimes beneficial because they may better refine the roughness of the primary and extra tasks after combining the main tasks. In fact, each task can make up for the limitations of other tasks due to quantification.

9) Some of the inputs are better as output items

Multi-task learning is useful in many areas where it is impractical to include certain features as input. Multi-tasking provides a way to benefit from these features by using them as additional tasks rather than ignoring them. Using certain features as output items may be more useful than using them as input items because it is entirely possible to construct such problems. For further details on this point, see [6].

4. Transfer learning and multi-task learning

Inductive transfer or transfer learning [47, 48] is widely studied in the machine learning field. It focuses on the information stored in the solved problem that can

be used to solve different and related problems. For example, learning to walk helps in learning to run. Of course, research in this area is associated with the transfer of learning in psychology, although the formalization of this association is limited. Multi-task learning is also studied in the field of machine learning. It studies a certain problem while learning other related problems that have the same representation method. Thus, multi-task learning is an inductive transfer method.

Table 5.1 summarizes the relationship between traditional machine learning and various types of transfer learning. According to different settings, the transfer learning is divided into three categories: inductive transfer learning, transitive transfer learning, and unsupervised transfer learning. Figure 5.2 summarizes the different settings for transfer learning [47].

Tab. 5.1: Relationship between traditional machine learning and various transfers.

Learning background settings	Source area and target area	Source task and target task
Traditional machine learning	Same	Same
Transfer learning	Same	Different but related
Inductive transfer learning/unsupervised transfer learning	Different but related	Different but related
Transductive transfer learning	Different but related	Same

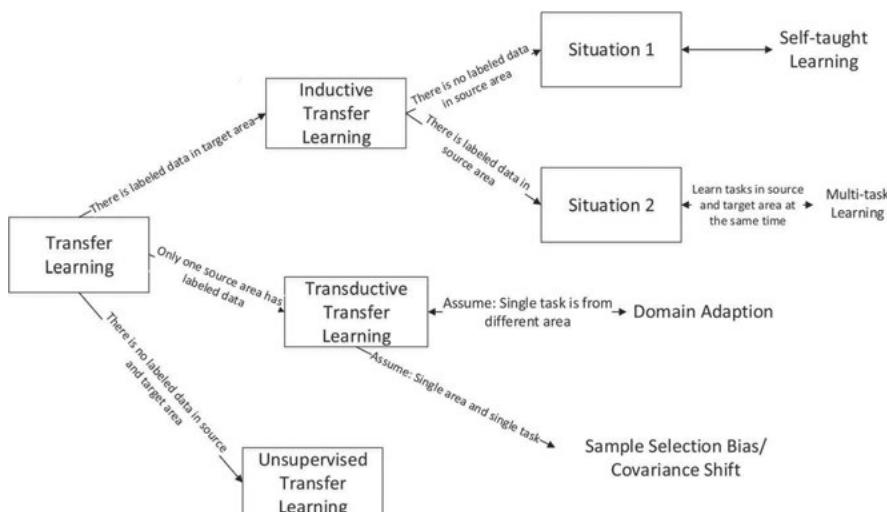


Fig. 5.2: The classification overview of different settings in transfer learning.

5. Make predictions for multiple tasks

Multi-task learning involves training multiple tasks in parallel within a model, but that does not mean that the model predicts the outcome of multiple tasks. The reason why multiple tasks are trained in a model is that each task can utilize useful information contained in other task training signals, rather than reducing the number of models that must be trained. The general performance of all tasks must be balanced with good performance on the task of interest. In general, it is best to optimize the performance of one task at a time, allowing the performance of other tasks to decline. When the need to forecast multiple tasks arises, it is particularly important for each task to be trained using an independent multi-task learning model.

6. Parallel transfer and sequential transfer

Multi-task learning is a type of parallel transfer. Though parallel tasks may seem easier, this is not the case. Parallel transfer has the following advantages:

- (1) Because all tasks are learned at the same time, all tasks have full details about the tasks being learned.
- (2) In many applications, additional tasks can be applied in parallel to the main task. Parallel migration does not require the definition of a training order, while the order of training in the sequential transfer is not the same.
- (3) Tasks can benefit from each other, which is not the case with a linear order. For example, if you learned Task 1 before Task 2, Task 2 does not help Task 1 to learn. This not only reduces the performance of Task 1 but also weakens the ability of Task 1 to assist the learning of Task 2.

When tasks occur naturally and continuously, the use of parallel transfer is easier than sequential transfer. If you can store training data, you can use existing tasks for multi-tasking and relearn after new tasks. If you cannot store training data, you can use the learned model to generate a priori simulation data. It is interesting to note that, while it is easy to migrate sequentially using parallel transfer, it is difficult to use sequential transfer to initiate a parallel transfer. However, it is worth mentioning that performing sequential transfer and parallel transfer at the same time is feasible.

5.3 Semi-supervised multi-task learning model based on DFS

Although semi-supervised multi-tasking is a hot topic at present and can deal with static problems effectively, dynamic fuzzy problems still cannot be solved. In 1965, Zadeh proposed the concept of fuzzy sets as a static concept. At this point, it was not possible to describe dynamic fuzzy phenomena, dynamic fuzzy events, and dynamic fuzzy concepts objectively. However, dynamic fuzzy set theory can describe dynamic fuzzy problems well, so it is an extremely natural thing to propose a semi-supervised multi-task learning model based on dynamic fuzzy sets.

5.3.1 Dynamic fuzzy machine learning model

In [34], dynamic fuzzy set theory was introduced into the machine learning domain, and a dynamic fuzzy machine learning model and related theories were proposed.

Definition 5.1 Dynamic fuzzy machine learning space: The space used to describe the learning process composed of all the dynamic fuzzy machine learning elements is called the dynamic fuzzy machine learning space. It consists of {learning sample, learning algorithm, input data, output data, representation theory} and is written as follows:

$$(\overleftarrow{S}, \overrightarrow{S}) = \{(\overleftarrow{Ex}, \overrightarrow{Ex}), ER, (\overleftarrow{X}, \overrightarrow{X}), (\overleftarrow{Y}, \overrightarrow{Y}), ET\}.$$

Definition 5.2 Dynamic fuzzy machine learning: Dynamic fuzzy machine learning $(\overleftarrow{l}, \overrightarrow{l})$ refers to the mapping of an input dataset $(\overleftarrow{X}, \overrightarrow{X})$ to an output dataset $(\overleftarrow{Y}, \overrightarrow{Y})$ in a dynamic fuzzy machine learning space $(\overleftarrow{S}, \overrightarrow{S})$, denoted by

$$(\overleftarrow{l}, \overrightarrow{l}) : (\overleftarrow{X}, \overrightarrow{X}) \rightarrow (\overleftarrow{Y}, \overrightarrow{Y}).$$

Definition 5.3 Dynamic fuzzy machine learning system: Five elements in the dynamic fuzzy machine learning space correspond with a certain learning mechanism to form the learning ability of a computer system, known as the dynamic fuzzy machine learning system.

Definition 5.4 Dynamic fuzzy machine learning model: $DFMLM = \{(\overleftarrow{S}, \overrightarrow{S}), (\overleftarrow{L}, \overrightarrow{L}), (\overleftarrow{u}, \overrightarrow{u}), (\overleftarrow{y}, \overrightarrow{y}), (\overleftarrow{p}, \overrightarrow{p}), (\overleftarrow{I}, \overrightarrow{I}), (\overleftarrow{O}, \overrightarrow{O})\}$, where $(\overleftarrow{S}, \overrightarrow{S})$ is the learning part (dynamic environment/dynamic fuzzy learning space), $(\overleftarrow{L}, \overrightarrow{L})$ is the dynamic learning part, $(\overleftarrow{u}, \overrightarrow{u})$ is the output of $(\overleftarrow{S}, \overrightarrow{S})$ to $(\overleftarrow{L}, \overrightarrow{L})$, $(\overleftarrow{y}, \overrightarrow{y})$ is the dynamic feedback of $(\overleftarrow{L}, \overrightarrow{L})$ to $(\overleftarrow{S}, \overrightarrow{S})$, $(\overleftarrow{p}, \overrightarrow{p})$ is the system learning performance index, $(\overleftarrow{I}, \overrightarrow{I})$ is the external environment of the dynamic fuzzy learning system input, and $(\overleftarrow{O}, \overrightarrow{O})$ is the system output to the outside world. See Fig. 5.3 for a schematic representation.

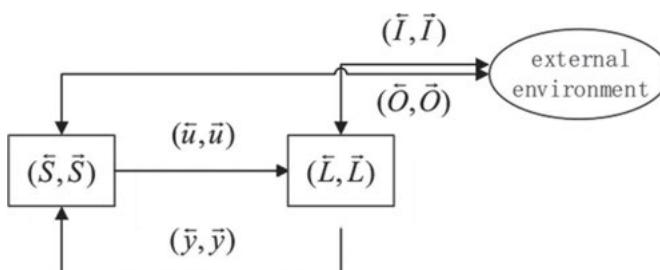


Fig. 5.3: Dynamic fuzzy machine learning model block diagram.

5.3.2 Dynamic fuzzy semi-supervised learning model

Definition 5.5 Dynamic fuzzy semi-supervised learning (DFSSL): Consider a set of samples $(\overleftarrow{X}, \overrightarrow{X}) = (\overleftarrow{L}, \overrightarrow{L})(\overleftarrow{U}, \overrightarrow{U})$ from an unknown distribution, where $(\overleftarrow{L}, \overrightarrow{L}) = \{((\overleftarrow{x}_1, \overrightarrow{x}_1), (\overleftarrow{y}_1, \overrightarrow{y}_1)), ((\overleftarrow{x}_2, \overrightarrow{x}_2), (\overleftarrow{y}_2, \overrightarrow{y}_2)), \dots, ((\overleftarrow{x}_{|\overleftarrow{L}, \overrightarrow{L}|}, \overrightarrow{x}_{|\overleftarrow{L}, \overrightarrow{L}|}), (\overleftarrow{y}_{|\overleftarrow{L}, \overrightarrow{L}|}, \overrightarrow{y}_{|\overleftarrow{L}, \overrightarrow{L}|}))\}$ is a marked sample set and $(\overleftarrow{U}, \overrightarrow{U}) = \{(\overleftarrow{x}'_1, \overrightarrow{x}'_1), (\overleftarrow{x}'_2, \overrightarrow{x}'_2), \dots, (\overleftarrow{x}'_{|\overleftarrow{U}, \overrightarrow{U}|}, \overrightarrow{x}'_{|\overleftarrow{U}, \overrightarrow{U}|})\}$ is the unlabelled sample set.

DFSSL seeks to obtain a function $(\overleftarrow{f}, \overrightarrow{f}) : (\overleftarrow{X}, \overrightarrow{X}) \rightarrow (\overleftarrow{Y}, \overrightarrow{Y})$ that can accurately predict the label $(\overleftarrow{y}, \overrightarrow{y})$ of sample $(\overleftarrow{x}, \overrightarrow{x})$ and classify it into class $(\overleftarrow{C}_y, \overrightarrow{C}_y)$ with membership degree $(\overleftarrow{C}_y(\overleftarrow{x}), \overrightarrow{C}_y(\overrightarrow{x}))$. Here, $(\overleftarrow{x}_i, \overrightarrow{x}_i)$ and $(\overleftarrow{x}'_i, \overrightarrow{x}'_i)$ are d -dimensional vectors, $(\overleftarrow{y}_i, \overrightarrow{y}_i) \in (\overleftarrow{Y}, \overrightarrow{Y})$ is the label of sample $(\overleftarrow{x}_i, \overrightarrow{x}_i)$, and $|\overleftarrow{L}|, |\overrightarrow{L}|, |\overleftarrow{U}|, |\overrightarrow{U}|$ are the sizes of $(\overleftarrow{L}, \overrightarrow{L})$ and $(\overleftarrow{U}, \overrightarrow{U})$, respectively (that is, the number of samples contained in the labelled and unlabelled sample sets).

Compared with the previous definitions, it is easy to find that DFSSL is a subclass of dynamic fuzzy machine learning. If the learning sample in the dynamic fuzzy machine learning space is refined into unlabelled and tagged data, dynamic fuzzy machine learning can be called DFSSL.

DFSSL is an attempt to find an optimal learner for the sample set $(\overleftarrow{X}, \overrightarrow{X})$. If $(\overleftarrow{X}, \overrightarrow{X}) = (\overleftarrow{L}, \overrightarrow{L})$, then the problem is transformed into dynamic fuzzy supervised learning; if $(\overleftarrow{X}, \overrightarrow{X}) = (\overleftarrow{U}, \overrightarrow{U})$, then the problem becomes dynamic fuzzy unsupervised learning. However, how to use the labelled samples and the unmarked samples effectively is one area of study for DFSSL. It is worth mentioning that the three hypotheses in traditional semi-supervised learning still exist in DFSSL.

5.3.3 DFSSMTL model

Definition 5.6 Dynamic fuzzy task (DFT): A problem or task with dynamic ambiguity is called a dynamic fuzzy task.

Obviously, descriptions of people's age, weight, and height are dynamic and fuzzy, so the description of these three tasks is a DFT. Figure 5.4 describes three ambiguous dynamic tasks. We assume that Task 1 describes the age of a person and is composed of three categories: old, young, and juvenile. Task 2 describes a person's weight and consists of obese, average, and slim. Task 3 describes the height of the person as either tall, medium, or short. Obviously, these three tasks are dynamic fuzzy characteristics because the age, weight, and height of the standard itself have some dynamic ambiguity.

Definition 5.7 Dynamic fuzzy multi-task learning (DFMTL): The learning model of dynamic fuzzy multi-task learning considers the sequential or simultaneous learning of multiple related DFTs.

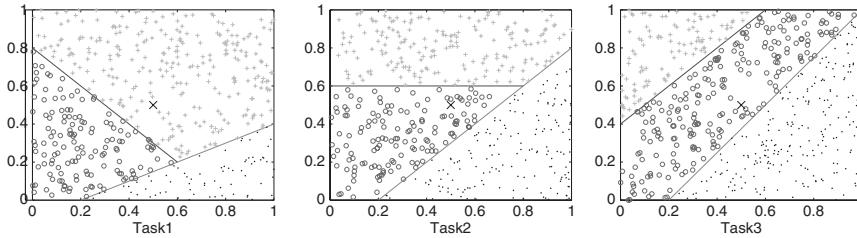


Fig. 5.4: Three examples of dynamic ambiguous tasks that consist of three classes.

It is easy to see that the different descriptive tasks describing different aspects of the same person from different angles are relevant; that is, the three dynamic ambiguity tasks are related. If we study these DFTs at the same time or sequentially, we are performing DFMTL.

Definition 5.8 Dynamic fuzzy semi-supervised multi-task learning: Given a set of M related DFTs, the sample set of each task is from an unknown distribution. The sample of the t -th task set is labelled as

$$(\overleftarrow{X}, \overrightarrow{X})_t = (\overleftarrow{L}, \overrightarrow{L})_t (\overleftarrow{U}, \overrightarrow{U})_t$$

where $(\overleftarrow{L}, \overrightarrow{L})_t = \{((\overleftarrow{x}_1, \overrightarrow{x}_1)_t, (\overleftarrow{y}_1, \overrightarrow{y}_1)_t), ((\overleftarrow{x}_2, \overrightarrow{x}_2)_t, (\overleftarrow{y}_2, \overrightarrow{y}_2)_t), \dots, ((\overleftarrow{x}_{|(\overleftarrow{L}, \overrightarrow{L})_t|}, \overrightarrow{x}_{|(\overleftarrow{L}, \overrightarrow{L})_t|})_t, (\overleftarrow{y}_{|(\overleftarrow{L}, \overrightarrow{L})_t|}, \overrightarrow{y}_{|(\overleftarrow{L}, \overrightarrow{L})_t|})_t\}$ is the tagged sample set for the t th task and $(\overleftarrow{U}, \overrightarrow{U})_t = \{(\overleftarrow{x}'_1, \overrightarrow{x}'_1)_t, (\overleftarrow{x}'_2, \overrightarrow{x}'_2)_t, \dots, (\overleftarrow{x}'_{|(\overleftarrow{U}, \overrightarrow{U})_t|}, \overrightarrow{x}'_{|(\overleftarrow{U}, \overrightarrow{U})_t|})_t\}$ is the unlabelled sample set for task t .

DFSS-MTL seeks to learn multiple DFTs simultaneously or sequentially and aims to learn a function for each task. The functions of the t th task are written as $(\overleftarrow{f}, \overrightarrow{f})_t : (\overleftarrow{X}, \overrightarrow{X})_t \rightarrow (\overleftarrow{Y}, \overrightarrow{Y})_t$, so that we can accurately predict the tag $(\overleftarrow{y}, \overrightarrow{y})_t$ of the samples $(\overleftarrow{x}, \overrightarrow{x})_t$ of the t th task and classify $(\overleftarrow{x}, \overrightarrow{x})_t$ into classes $(\overleftarrow{C}_y, \overrightarrow{C}_y)_t$ whose membership degree is denoted as $(\overleftarrow{C}_y(\overleftarrow{x}), \overrightarrow{C}_y(\overrightarrow{x}))_t$. Here, $(\overleftarrow{x}_i, \overrightarrow{x}_i)_t$ and $(\overleftarrow{x}'_i, \overrightarrow{x}'_i)_t$ are d -dimensional vectors, $(\overleftarrow{y}_i, \overrightarrow{y}_i)_t \in (\overleftarrow{Y}, \overrightarrow{Y})_t$ is the tag for the t th task samples $(\overleftarrow{x}_i, \overrightarrow{x}_i)_t$, and $|(\overleftarrow{L}, \overrightarrow{L})_t|, |(\overleftarrow{U}, \overrightarrow{U})_t|$ represent the size of the untagged sample set $(\overleftarrow{L}, \overrightarrow{L})_t$ of the t th task and the unlabelled sample set $(\overleftarrow{U}, \overrightarrow{U})_t$ of the t th task (that is, the number of samples in the labelled sample set and unlabelled sample set of the t th task).

For the former definition, it is easy to find that DFSS-MTL is also a subclass of DFML. If the learning sample in the dynamic fuzzy machine learning space is refined into unlabelled data and tagged data in a single DFT and multiple related DFTs are learned in sequence or simultaneously, it can be said that the dynamic fuzzy machine learning is DFSSMTL.

DFSS-MTL is mainly considered from two aspects. First, when the data in the task to be learned are extremely rare, the DFT can be learned using sample data from similar or related DFTs learned in the past, that is, learning in the context of DFMTL. Second, in the case of DFMTL, there are unlabelled sample data in addition to the labelled sample data, which belong to the DFSSL domain.

Figure 5.5 depicts a simple DFSS-MTL model. The learning sample sets $\{(\bar{X}, \bar{X})_1, (\bar{X}, \bar{X})_2, \dots, (\bar{X}, \bar{X})_M\}$ of M tasks are obtained under this model. The class label sets $\{(\bar{Y}, \bar{Y})_1, (\bar{Y}, \bar{Y})_2, \dots, (\bar{Y}, \bar{Y})_M\}$ of each task are obtained and the membership degree $(\bar{C}_y(\bar{x}), \bar{C}_y(\bar{x}))_t$ of a sample (\bar{x}, \bar{x}) in task t is represented by (\bar{C}_y, \bar{C}_y) .

5.4 Dynamic fuzzy semi-supervised multi-task matching algorithm

We can regard the performance of the dynamic fuzzy system as different operating states in different time periods or moments, and each state in the pattern recognition method represents a dynamic fuzzy set of similar patterns, which together form finite regions in the feature space. We call this the class. Thus, the membership function can be used to identify a state or class of the new model. In this chapter, we will introduce a DFSS-MTL algorithm for dynamic fuzzy semi-supervised multi-task matching. The main purpose of this algorithm is to learn the membership functions from finite initial datasets and update the membership functions of the corresponding categories with the addition of a new model. This can effectively identify each model and then solve the corresponding dynamic fuzzy problem.

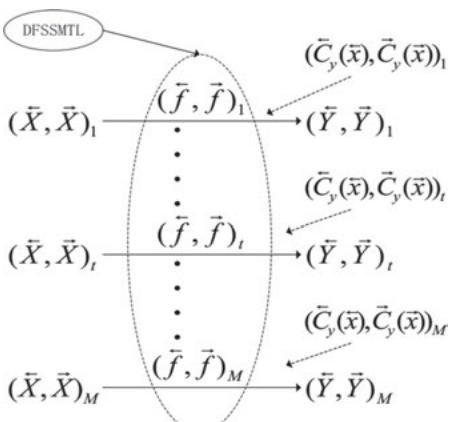


Fig. 5.5: A simple dynamic fuzzy semi-supervised multi-task learning model.

5.4.1 Dynamic fuzzy random probability

In this algorithm, dynamic fuzzy stochastic probabilities will be used as the main way to calculate the membership degree of a certain model. Therefore, the relevant dynamic fuzzy random probability is briefly introduced [36, 37].

Definition 5.9 Dynamic fuzzy data: $(\overleftarrow{A}, \overrightarrow{A}) \in DF(R)$ is DFD if

- (1) $(\overleftarrow{A}, \overrightarrow{A})$ is normal; that is, there exists some $(\overleftarrow{x}, \overrightarrow{x}) \in R$ such that $(\overleftarrow{A}, \overrightarrow{A})(\overleftarrow{x}, \overrightarrow{x}) = (\overleftarrow{1}, \overrightarrow{1})$.
- (2) $\forall (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \in [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]$, $(\overleftarrow{A}, \overrightarrow{A})_{(\overleftarrow{\alpha}, \overrightarrow{\alpha})}$ is bounded by a closed interval.

Definition 5.10 Distribution number: The map $DF : (\overleftarrow{R}, \overrightarrow{R}) \rightarrow [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]$ is a distribution number if

- (1) DF is monotone decreasing;
- (2) DF is left continuous;
- (3) $DF(-\infty) = (\overleftarrow{0}, \overrightarrow{0})$, $DF(+\infty) = (\overleftarrow{1}, \overrightarrow{1})$.

If the distribution number $DF(\overleftarrow{0}, \overrightarrow{0}) = (\overleftarrow{0}, \overrightarrow{0})$, it is called a positive distribution function.

If DF is a positive distribution number, when $(\overleftarrow{x}, \overrightarrow{x}) \leq (\overleftarrow{0}, \overrightarrow{0})$, $DF(\overleftarrow{x}, \overrightarrow{x}) = (\overleftarrow{0}, \overrightarrow{0})$.

Definition 5.11 Dynamic fuzzy variable: For some domain $(\overleftarrow{U}, \overrightarrow{U})$, if $(\overleftarrow{x}, \overrightarrow{x})$ is a dynamic fuzzy variable, then its value is in the domain, that is, $(\overleftarrow{x}, \overrightarrow{x}) = (\overleftarrow{u}, \overrightarrow{u})$, $(\overleftarrow{u}, \overrightarrow{u}) \in (\overleftarrow{U}, \overrightarrow{U})$.

As $(\overleftarrow{F}, \overrightarrow{F})$ is a dynamic fuzzy set on $(\overleftarrow{U}, \overrightarrow{U})$, $(\overleftarrow{F}(\overleftarrow{u}), \overrightarrow{F}(\overrightarrow{u}))$ is the membership degree of $(\overleftarrow{F}, \overrightarrow{F})$.

Definition 5.12 Dynamic fuzzy distribution: As $(\overleftarrow{x}, \overrightarrow{x})$ is a dynamic fuzzy variable, $(\overleftarrow{F}, \overrightarrow{F})$ is the dynamic fuzzy limit of $(\overleftarrow{x}, \overrightarrow{x})$, that is, $DR(\overleftarrow{x}, \overrightarrow{x})$. Then, the proposition that $(\overleftarrow{x}, \overrightarrow{x})$ is $(\overleftarrow{F}, \overrightarrow{F})$ can be expressed as $DR(\overleftarrow{x}, \overrightarrow{x}) = (\overleftarrow{F}, \overrightarrow{F})$. This is associated with a dynamic fuzzy distribution $D\Pi_{(\overleftarrow{x}, \overrightarrow{x})}$ that makes $D\Pi_{(\overleftarrow{x}, \overrightarrow{x})} = DR(\overleftarrow{x}, \overrightarrow{x})$.

If $D\Pi_{(\overleftarrow{x}, \overrightarrow{x})}$ is a set $(\overleftarrow{X}, \overrightarrow{X})$ that is composed of a dynamic ambiguity variable $(\overleftarrow{x}, \overrightarrow{x})$ and is mapped to an interval $[(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]$, we set $D\Pi_{(\overleftarrow{x}, \overrightarrow{x})} : (\overleftarrow{X}, \overrightarrow{X}) \rightarrow [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]$ so as to satisfy the following:

- (1) $D\Pi_{(\overleftarrow{x}, \overrightarrow{x})}$ is monotone decreasing;
- (2) $D\Pi_{(\overleftarrow{x}, \overrightarrow{x})}$ is left continuous; and
- (3) $D\Pi_{(\overleftarrow{x}, \overrightarrow{x})}(-\infty) = (\overleftarrow{0}, \overrightarrow{0})$, $D\Pi_{(\overleftarrow{x}, \overrightarrow{x})}(+\infty) = (\overleftarrow{1}, \overrightarrow{1})$.

Then, it can be said that the dynamic fuzzy distribution is the distribution number defined above.

In fact, if the distribution function of the dynamic fuzzy distribution $D\Pi_{(\overleftarrow{X}, \overrightarrow{X})}$ is denoted by $d\pi_{(\overleftarrow{X}, \overrightarrow{X})}$, then the dynamic fuzzy distribution function associated with $(\overleftarrow{X}, \overrightarrow{X})$ is numerically equal to the membership degree of $(\overleftarrow{F}, \overrightarrow{F})$. That is, the dynamic fuzzy probability of $(\overleftarrow{X}, \overrightarrow{X}) = (\overleftarrow{u}, \overrightarrow{u})$ can be approximated as $d\pi_{(\overleftarrow{X}, \overrightarrow{X})}((\overleftarrow{u}, \overrightarrow{u})) \triangleq (\overleftarrow{F}(\overleftarrow{u}), \overrightarrow{F}(\overrightarrow{u}))$, where $d\pi_{(\overleftarrow{X}, \overrightarrow{X})}((\overleftarrow{u}, \overrightarrow{u}))$ is abbreviated as $d\pi((\overleftarrow{u}, \overrightarrow{u}))$.

In [49], fuzzy sets form the basic theory of fuzzy stochastic probability, which solves fuzzy problems well. However, it cannot deal with dynamic fuzzy problems. Therefore, we can say that dynamic fuzzy variables correspond to classical random probability, fuzzy random probability, and dynamic fuzzy probability. In [50], the transition theory between classical stochastic probability and fuzzy stochastic probability is discussed. Similarly, to better deal with dynamic fuzzy problems, we transform the classical stochastic probability to dynamic fuzzy stochastic probability. If there are n dynamic fuzzy variables $(\overleftarrow{x}, \overrightarrow{x})$, and $p((\overleftarrow{x}_i, \overrightarrow{x}_i))$, $p((\overleftarrow{x}_j, \overrightarrow{x}_j))$ are the classical random probabilities of the dynamic fuzzy variables $(\overleftarrow{x}_i, \overrightarrow{x}_i)$ and $(\overleftarrow{x}_j, \overrightarrow{x}_j)$, then the dynamic fuzzy random probability $d\pi((\overleftarrow{x}_i, \overrightarrow{x}_i))$ can be expressed as

$$d\pi((\overleftarrow{x}_i, \overrightarrow{x}_i)) = \sum_{j=1, \dots, n} \min(p((\overleftarrow{x}_i, \overrightarrow{x}_i)), p((\overleftarrow{x}_j, \overrightarrow{x}_j))) \quad (5.1)$$

5.4.2 Dynamic fuzzy semi-supervised multi-task matching algorithm

1. Dynamic fuzzy semi-supervised matching algorithm

Firstly, the dynamic fuzzy semi-supervised matching algorithm is described in the context of single tasks, which is an improvement on semi-supervised fuzzy pattern matching [51]. Without knowing the number of classes in advance, a probabilistic histogram is used to estimate the edge probability density of each attribute in the class. As patterns are added, the dynamic fuzzy probability (or membership) for each class is gradually formed.

$\{(\overleftarrow{C}_1, \overrightarrow{C}_1), (\overleftarrow{C}_2, \overrightarrow{C}_2), \dots, (\overleftarrow{C}_c, \overrightarrow{C}_c)\}$ represents c categories, each of which can be regarded as a dynamic fuzzy set. Each model of each class consists of d attributes; $(\overleftarrow{x}, \overrightarrow{x}) = (\overleftarrow{x}^1, \overrightarrow{x}^1), (\overleftarrow{x}^2, \overrightarrow{x}^2), \dots, (\overleftarrow{x}^d, \overrightarrow{x}^d)$ denotes a dynamic fuzzy pattern consisting of d attributes in the feature space; $(\overleftarrow{X}_{\max}^j, \overrightarrow{X}_{\max}^j)$ and $(\overleftarrow{X}_{\min}^j, \overrightarrow{X}_{\min}^j)$ represent the property j histogram of the maximum and minimum values, respectively, and determine the histogram's size according to the actual situation; H represents the number of intervals in the histogram; and 1 denotes the k th interval of attribute j . When an attribute value of some new pattern 1 exceeds the maximum and minimum values of the attribute, the current histogram of the attribute needs to be updated. Let 1 , 2 , and h denote the maximum and minimum values of the updated attribute j and the number of intervals. These are updated by the following formula:

$$\begin{cases}
 \overleftarrow{x}^j < \overrightarrow{X}_{\min}^j - \frac{\Delta j}{2} \Rightarrow \overleftarrow{X}'^j_{\min} = \overrightarrow{X}_{\min}^j - \Delta j \\
 \overleftarrow{x}^j > \overrightarrow{X}_{\max}^j + \frac{\Delta j}{2} \Rightarrow \overleftarrow{X}'^j_{\max} = \overrightarrow{X}_{\max}^j + \Delta j , \quad \text{where } \Delta j = \frac{\overrightarrow{X}_{\max}^j - \overrightarrow{X}_{\min}^j}{h} \\
 h' = h + 1
 \end{cases} \quad (5.2)$$

or

$$\begin{cases}
 \overleftarrow{x}^j < \overleftarrow{X}_{\min}^j - \frac{\Delta j}{2} \Rightarrow \overleftarrow{X}'^j_{\min} = \overleftarrow{X}_{\min}^j - \Delta j \\
 \overleftarrow{x}^j > \overleftarrow{X}_{\max}^j + \frac{\Delta j}{2} \Rightarrow \overleftarrow{X}'^j_{\max} = \overleftarrow{X}_{\max}^j + \Delta j , \quad \text{where } \Delta j = \frac{\overleftarrow{X}_{\max}^j - \overleftarrow{X}_{\min}^j}{h} \\
 h' = h + 1
 \end{cases}$$

The number of patterns $n_{ib(j,k)}$ belonging to class $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ in the interval according to the attribute j pattern is represented by the height of the histogram in every interval $b(j, k)$, $k \in \{1, 2, \dots, h\}$. The ratio of the total number N_i of patterns of class $n_{ib(j,k)}$ and class $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ can be used to obtain probability distributions of some attribute j in class 6. In general, we use the probability of $(\overleftarrow{y}_{b(j,k)}, \overrightarrow{y}_{b(j,k)})$ at the midpoint of the interval to approximate the probability that attribute j belongs to class $(\overleftarrow{C}_i, \overrightarrow{C}_i)$; that is,

$$p((\overleftarrow{C}_i, \overrightarrow{C}_i)|(\overleftarrow{y}_{b(j,k)}, \overrightarrow{y}_{b(j,k)})) = \frac{n_{b(j,k)}}{N_i}, \quad (5.3)$$

which we write as $p_{ik}^j((\overleftarrow{y}, \overrightarrow{y}))$.

Combining this with (5.1), the random dynamic fuzzy probability is

$$d\pi((\overleftarrow{C}_i, \overrightarrow{C}_i)|(\overleftarrow{y}_{b(j,k)}, \overrightarrow{y}_{b(j,k)})) = \sum_{f=k}^h \min(p_{ik}^j((\overleftarrow{y}, \overrightarrow{y})), p_{if}^j((\overleftarrow{y}, \overrightarrow{y}))), \quad (5.4)$$

which is written as $d\pi_{ik}^j((\overleftarrow{y}, \overrightarrow{y}))$.

1) Classify new patterns

The membership degree of each attribute j in class $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ is approximated as a dynamic fuzzy random distribution. Thus, a new pattern $(\overleftarrow{x}, \overrightarrow{x})$ with a different attribute value of $(\overleftarrow{x}^1, \overrightarrow{x}^1), (\overleftarrow{x}^2, \overrightarrow{x}^2), \dots, (\overleftarrow{x}^d, \overrightarrow{x}^d)$ can be classified in two steps:

- (1) The membership degree $d\pi_i^j((\overleftarrow{x}, \overrightarrow{x}))$ of each attribute j in class $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ is determined by the dynamic fuzzy random distribution.
- (2) The membership degrees of all attributes j of class $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ are combined by an operator H ; that is,

$$d\pi_i((\overleftarrow{x}, \overrightarrow{x})) = H(d\pi_i^1((\overleftarrow{x}, \overrightarrow{x})), \dots, d\pi_i^d((\overleftarrow{x}, \overrightarrow{x}))). \quad (5.5)$$

Thus, the global membership value $d\pi_i((\overleftarrow{x}, \overrightarrow{x}))$ of pattern $(\overleftarrow{x}, \overrightarrow{x})$ belonging to class $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ is obtained. Because the minimum operation is simple and the result is good, we use the minimum operator as H in the proposed system. Finally, the pattern $(\overleftarrow{x}, \overrightarrow{x})$ is assigned to the class with the largest membership. If the membership of pattern $(\overleftarrow{x}, \overrightarrow{x})$ belongs to a class whose membership is not $(\overleftarrow{0}, \overrightarrow{0})$, then the pattern is classified into the class and the dynamic fuzzy distribution of each attribute of the class is updated.

2) Detect new classes

For an existing class c , the first pattern rejected by all classes is treated as a prototype of the new class and the membership degrees of the new class are calculated using (5.3) and (5.4). The number of categories is increased by 1. For the first pattern $(\overleftarrow{x}, \overrightarrow{x})$ of the new class c , if the attribute value of the pattern is in the interval $b(j, k)$, $k \in \{1, 2, \dots, h\}$, the probability histogram of attribute j in class c can be expressed as

$$p_c^j = \{p_{c1}^j = 0, p_{c2}^j = 0, \dots, p_{ck}^j = 1, \dots, p_{ch}^j = 0\}.$$

Then, the dynamic fuzzy random probability histogram is constructed using (5.4). Obviously, there is only one mode, so the dynamic fuzzy random probability histogram and probability histogram are the same. The specific process of detecting new classes is briefly described as follows:

$$\begin{aligned} d\pi_i((\overleftarrow{x}, \overrightarrow{x})) &= 0, \forall i \in \{1, 2, \dots, c\} \\ \Rightarrow c &\leftarrow c + 1, (\overleftarrow{C}_c, \overrightarrow{C}_c) = \{(\overleftarrow{x}, \overrightarrow{x})\}, d\pi_c = \{d\pi_c^1, \dots, d\pi_c^j, \dots, d\pi_c^d\}. \end{aligned}$$

3) Update the local membership degree

If pattern $(\overleftarrow{x}, \overrightarrow{x})$ belongs to class $(\overleftarrow{C}_i, \overrightarrow{C}_i)$, then the dynamic fuzzy distribution of class $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ needs to be locally updated. The probability distribution and the dynamic fuzzy distribution of the j th attribute of class $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ are denoted by $p_i^j = \{p_{i1}^j, p_{i2}^j, \dots, p_{ih}^j\}$ and $d\pi_i^j = \{d\pi_{i1}^j, d\pi_{i2}^j, \dots, d\pi_{ih}^j\}$, respectively, and the two probabilities are denoted by $p_i'^j = \{p_{i1}'^j, p_{i2}'^j, \dots, p_{ih}'^j\}$ and $d\pi_i'^j = \{d\pi_{i1}'^j, d\pi_{i2}'^j, \dots, d\pi_{ih}'^j\}$, respectively, after updating.

For simplicity of calculation, let $p_{ih}^j < p_{i(h-1)}^j < \dots < p_{i1}^j$. The new random probability can then be calculated according to the following formula [52]:

$$\begin{aligned} (\overleftarrow{x}^j, \overrightarrow{x}^j) \in b(j, k), \forall k \in \{1, \dots, h\} \Rightarrow p_{ik}'^j &= p_{ik}^j \times \frac{N_i}{N_i + 1} + \frac{1}{N_i + 1}, \\ p_{iz}'^j &= p_{iz}^j \times \frac{N_i}{N_i + 1} \quad \forall z \in \{1, \dots, h\}, z \neq k \end{aligned} \tag{5.6}$$

A new dynamic fuzzy stochastic probability can be obtained by combining this with (5.4). The detailed process of updating the local membership is briefly described as follows:

$$d\pi_i((\overleftarrow{x}, \overrightarrow{x})) = \max_{z \in [1, \dots, c]} (d\pi_z((\overleftarrow{x}, \overrightarrow{x})))$$

$$\Rightarrow (\overleftarrow{C}_i, \overrightarrow{C}_i) \leftarrow \{(\overleftarrow{C}_i, \overrightarrow{C}_i), (\overleftarrow{x}, \overrightarrow{x})\}, d\pi'_i = \{d\pi_i'^1, \dots, d\pi_i'^j, \dots, d\pi_i'^d\}.$$

4) Merge similar classes

If the patterns occur in different orders, then eventually there will be a number of different divisions or clusters of different numbers, and some clusters may represent the same pattern. Therefore, we need to merge clusters that represent the same pattern. The merged similarity measure involves the similarity defined in [53]. We define the similarity criterion as follows:

$$(\overleftarrow{\delta}_{iz}, \overrightarrow{\delta}_{iz}) = (\overleftarrow{1}, \overrightarrow{1}) - \frac{\sum_{(\overleftarrow{x}, \overrightarrow{x}) \in (\overleftarrow{C}_i, \overrightarrow{C}_i) \text{ or } (\overleftarrow{x}, \overrightarrow{x}) \in (\overleftarrow{C}_z, \overrightarrow{C}_z)} |d\pi_i((\overleftarrow{x}, \overrightarrow{x})) - d\pi_z((\overleftarrow{x}, \overrightarrow{x}))|}{\sum_{(\overleftarrow{x}, \overrightarrow{x}) \in (\overleftarrow{C}_i, \overrightarrow{C}_i)} d\pi_i((\overleftarrow{x}, \overrightarrow{x})) + \sum_{(\overleftarrow{x}, \overrightarrow{x}) \in (\overleftarrow{C}_z, \overrightarrow{C}_z)} d\pi_z((\overleftarrow{x}, \overrightarrow{x}))} \quad (5.7)$$

The membership functions are merged online in an incremental manner. Let $p_i^j = \{p_{i1}^j, \dots, p_{ih}^j\}$ and $p_z^j = \{p_{z1}^j, \dots, p_{zh}^j\}$ be the random probability distributions of cluster $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ and cluster $(\overleftarrow{C}_z, \overrightarrow{C}_z)$, respectively.

Equation (5.8) gives the new probabilities for interval $b(j, k)$ after the merging of the two clusters:

$$p_{izk}^j = \frac{n_{ib(j,k)} + n_{zb(j,k)}}{N_i + N_z} \quad (5.8)$$

Further simplifying (5.8) yields the following:

$$p_{izk}^j = \frac{n_{ib(j,k)}}{N_i} \times \frac{N_i}{N_i + N_z} + \frac{n_{zb(j,k)}}{N_z} \times \frac{N_z}{N_i + N_z}$$

$$= p_{ik}^j \times \frac{N_i}{N_i + N_z} + p_{zk}^j \times \frac{N_z}{N_i + N_z} \quad (5.9)$$

$$\forall j \in \{1, 2, \dots, d\}, \forall k \in \{1, 2, \dots, h\}$$

After calculating the random probabilities of the corresponding attributes of each class, we can calculate the membership of the corresponding attributes of each class and update the membership degree of each mode using (5.4).

2. Knowledge transfer

Generally speaking, if the current task being learned is associated with the past, then people can quickly learn this new task. In multi-task learning, we need to understand how knowledge transfer can be made more effective [54]. In this section, the tasks are related; that is, there are overlapping areas between any two tasks. When we learn new tasks, we can use some relevant information from the previous task to improve the learning of the current task. We assume that the pattern of each class in each task is representative; that is, similarity patterns with the same label in the previous task have the same label in the next task. As shown in Fig. 5.5, the neighbours of the same pattern x are marked with a square in Task 1, and these neighbours should also have the same label as pattern x in Task 2.

Figure 5.6 shows the basic idea of knowledge transfer. Suppose we have successfully learned the first task. As can be seen in Fig. 5.6(1), Task 1 has three types of patterns, labelled with a plus, an asterisk, and a square. When the second task occurs, we need to classify pattern x . First, x is placed on the first task and classified as a plus, and in the same class, x has five neighbours (circled in a dashed line). Then, in the current task, the labelled categories for these fields of pattern x should also be the same. Assuming that pattern x is classified in the asterisk category in the current task, we can also classify the marked category of the field of x as an asterisk, as shown in Fig. 5.6(2).

However, the quality of the classification depends on the size of the neighbourhood and the formation of the interface. In fact, some of the migrated neighbourhood patterns may mislead the learning of the current task, which is a natural result of the induction bias. We can use training samples to correct that misleading. That is, when a new training pattern appears, we first find the nearest neighbour of the pattern and check whether this is the same as the new training pattern. If the two are different,

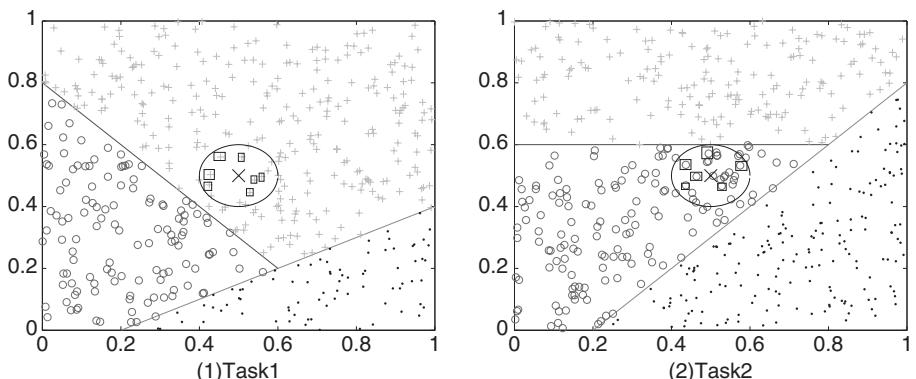


Fig. 5.6: Basic idea of knowledge transfer.

the marker of the nearest neighbour must be corrected to the label of the current new training pattern.

3. Description of dynamic fuzzy semi-supervised multi-task matching algorithm

The general dynamic fuzzy semi-supervised multi-task matching algorithm (DFSSMTPM) is described as follows.

For the first task, the DFSSL algorithm is used to classify the learning tasks. For other tasks, the neighbour and the nearest neighbour of the current training pattern are found in a copy of the previously learned task. We verify that the label of the current training pattern is the same as that of the nearest neighbour, or we correct the label of the nearest neighbour; in addition, we combine these neighbours to learn the current task.

5.4.3 Case analysis

1. Description of experimental data

To test the validity of our algorithm, we reintegrated the UCI Iris dataset to simulate several related tasks. The Iris dataset consists of three classes of data across 150 data points, with 50 data items and four attributes. Using the idea of constructing multi-tasking data [54], we reconstructed three tasks for the Iris data. Table 5.2 presents the integrated data. For the same dataset, different classification problems constitute related tasks. Task 1 is composed of raw data; that is, it contains three types of data. Task 2 contains two types of data, the original data from the first and second types of data constitute the first class, and the remaining data constitute the second type; Task 3 also contains two types of data, the original data from the first class and the combined data from the original second and third types.

2. Experimental results and analysis

We mainly investigate the following aspects. Effectiveness of knowledge transfer: To test the validity of knowledge transfer, we carried out a comparison of two kinds of experiments on the same task. First, we conducted a dynamic fuzzy semi-supervised matching experiment as a single task. Then, we carried out a dynamic fuzzy semi-supervised multi-task matching experiment with a multi-task background.

Tab. 5.2: Multi-task problem with iris data.

Original data	Task 1	Task 2	Task 3
1	1	4	6
2	2	4	7
3	3	5	7

At the same time, we compared our algorithm with that reported in [51] to test the validity of knowledge transfer in related tasks.

Semi-supervised classification vs. supervised classification: In [51], the authors demonstrate the superiority of fuzzy pattern matching for semi-supervised tasks under a single task without prior knowledge. In this chapter, we attempt to show that dynamic fuzzy semi-supervised pattern matching is better than supervised fuzzy pattern matching in the multi-task background when there are few training data for a task.

We mainly detect the classification error rate of the algorithms, i.e. the ratio of the number of erroneous predictions in the test samples to the total number of test samples. In the experiment, we considered Task 1 as the basic task; that is, we learned the remaining two tasks after learning Task 1. First, we need to understand the number of histograms in a single task, the relationship between the number of markers, and the error rate. As shown in Fig. 5.7, we used the dynamic fuzzy semi-supervised matching algorithm (DFSSPM) to learn Task 1. It can be seen that when the number of markers in the task is fixed, the number of intervals is related to the classification effect, so the number of intervals is one of the factors that affects the classification effect. When the number of intervals is fixed, the number of out-of-label data is in direct proportion to the effect of classification, which is consistent with our intuition.

It is also important to identify the neighbourhood radius of the relationship between the size and classification results when the number of intervals and labels is known. As shown in Fig. 5.8, we applied the fuzzy pattern matching algorithm [51] and the dynamic fuzzy semi-supervised matching algorithm to Task 3 in the multi-task background. These are abbreviated as MFPM and DFSSMTPM, respectively.

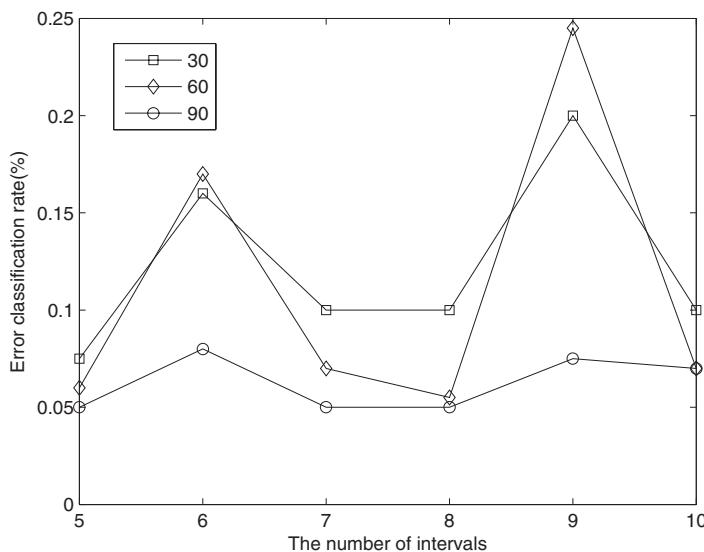


Fig. 5.7: The relationship between the number of intervals and the number of tags and the error rate.

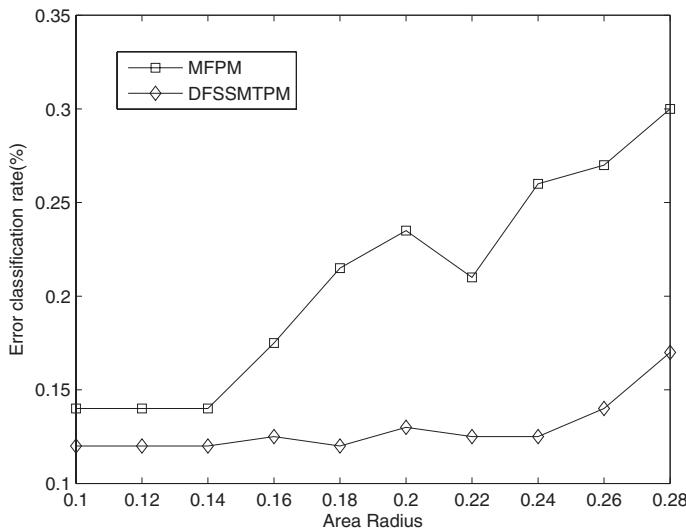


Fig. 5.8: The relationship between the size of the neighbourhood radius, the two methods, and the error rate.

There were 30 labelled training data and six histogram intervals. As can be seen from the figure, regardless of which method is used, the size of the neighbourhood radius affects the classification; as mentioned earlier, this is a natural result of the introduction of inductive bias. Overall, the DFSSMTPM algorithm gives a better classification than MFPM does. As MFPM is a monitoring algorithm, we can say that in the context of multi-tasking, our semi-supervised algorithm classification using tag-free data is effective. Using this new information to update the current category membership function produces a better classification than that given by MFPM [51].

To illustrate the validity of the knowledge transfer mechanism, dynamic fuzzy semi-supervised matching (DFSSPM) and dynamic fuzzy semi-supervised multi-task learning algorithm (DFSSMTPM) were applied to single tasks. Table 5.3 lists the classification performance on Task 2 with six histogram intervals; for the DFSSMTPM algorithm, the neighbourhood radius was set to 0.2. The table lists the classification error rates of the two algorithms for different numbers of marked training data. It can be seen from the table that the overall classification error rate of DFSSPM is higher than that of DFSSMTPM, which indicates that the knowledge migration mechanism can reduce the classification error rate after extending the number of labelled data.

Tab. 5.3

Methods	Number of tagged data		
	30	60	90
DFSSPM	0.358333	0.311111	0.2
DFSSMTPM	0.333333	0.3	0.116667

5.5 DFSSMTAL algorithm

Real-life problems are often dynamic and fuzzy. It is difficult to use traditional classification method to correctly study the corresponding classification model for these problems. Thus, an adaptive classifier is constructed in which the parameters are updated as the model discovers new data. In this section, we will introduce a DFSSMTAL algorithm, i.e. semi-supervised multi-task learning in the context of an adaptive classification method, in order to provide a new method to solve dynamic fuzzy problems.

5.5.1 Mahalanobis distance metric

Definition 5.13 Metric [55]: A mapping $D : \chi \times \chi \rightarrow \mathbb{R}_0^+$ in a vector space χ is called a metric if and only if, $\forall x_i, x_j, x_k \in \chi$ in the vector space, the following are satisfied:

- (1) Triangle inequality: $D(x_i, x_j) + D(x_j, x_k) \geq D(x_i, x_k)$
- (2) Nonnegativity: $D(x_i, x_j) \geq 0$
- (3) Symmetry: $D(x_i, x_j) = D(x_j, x_i)$
- (4) Distinguishability: $D(x_i, x_j) = 0 \Leftrightarrow x_i = x_j$

Strictly speaking, if a mapping satisfies the first three conditions (i.e. triangular inequality, nonnegativity, and symmetry) but does not satisfy the fourth condition (discernibility), it is called a pseudo-metric. In the absence of special circumstances, the latter measurement refers to the pseudo-metric.

After linearly transforming the vectors $x = Lx$ in the vector space χ , a set of metrics is obtained by calculating the Euclidean distance between the vectors. The square of the distance can be calculated using (5.10), where the matrix L is a parameter in the linear transformation. It is easily proved that if L is a full rank matrix, (5.10) is a valid metric; otherwise, it is a pseudo-metric.

$$D_L(x_i, x_j) = \|L(x_i - x_j)\|_2^2 \quad (5.10)$$

The quadratic matrix in (5.11) is often used to describe the square of the distance in (5.10).

$$M = L^T L \quad (5.11)$$

A matrix M of any real number matrix L using (5.11) must be a positive semidefinite matrix. In other words, the matrix M has no nonnegative eigenvalues.

Definition 5.14 Mahalanobis metric: For any matrix L of real numbers using the matrix M of (5.11), we denote the square of the distance as

$$D_M(x_i, x_j) = (x_i - x_j)^T M (x_i - x_j), \quad (5.12)$$

and the pseudo-metric of this form is called the Markov metric.

When the matrix M is the inverse of the covariance matrix, the Gaussian distribution of the quadratic form can be described using (5.12). If the matrix M is a semi-definite matrix, (5.10) and (5.12) can be thought of as a generalized form of the Euclidean distance, and when M is an identity matrix, it can be used to calculate the Euclidean distance.

As can be seen from (5.12), the Markov distance metric can be obtained using the matrix L or the matrix M , and L can uniquely determine M . Using a rotation (which does not affect the calculated distance), the matrix M gives the matrix L . This means that two different methods can be used to learn the distance metric – the first is to estimate the linear transformation matrix L , and the second is to estimate the positive semidefinite matrix M . Note that there is no restriction on the former method, but the latter requires a positive semidefinite matrix.

5.5.2 Dynamic fuzzy K-nearest neighbour algorithm

Although the K-nearest neighbour (KNN) algorithm [56] is the oldest and simplest method in pattern classification, it is more effective than some current methods when there is explicit prior knowledge [57, 58]. KNN classifies each unlabelled data point according to most of the KNNs in the training set, and the classification effect depends largely on the distance measure to identify similar neighbours. In the absence of prior knowledge, most KNN classification methods use the Euclidean distance to measure the similarity among the sample data, but the Euclidean distance cannot obtain any statistical information from the data.

One problem of KNN is that the usefulness of each training sample in the training set is treated the same when classifying the sample data. Thus, when the sample sets overlap, they are often difficult to classify; on the other hand, after classifying a kind of sample, the extent to which the sample belongs to the class is not specified. In this section, we set dynamic fuzzy theory and the Mahalanobis distance into the KNN algorithm to form a dynamic fuzzy KNN (DFKNN) algorithm. It is expected that this algorithm will be able to deal with dynamic fuzzy problems better.

For a given sample set $\{(\overleftarrow{x}_1, \overrightarrow{x}_1), (\overleftarrow{x}_2, \overrightarrow{x}_2), \dots, (\overleftarrow{x}_n, \overrightarrow{x}_n)\}$ consisting of n samples, if they are divided into c categories $\{(\overleftarrow{C}_1, \overrightarrow{C}_1), (\overleftarrow{C}_2, \overrightarrow{C}_2), (\overleftarrow{C}_c, \overrightarrow{C}_c)\}$, then the membership of the k th sample $(\overleftarrow{x}_k, \overrightarrow{x}_k)$ belonging to the i th class $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ is denoted by $(\overleftarrow{C}_{ik}, \overrightarrow{C}_{ik}) = (\overleftarrow{C}_i(\overleftarrow{x}_k), \overrightarrow{C}_i(\overrightarrow{x}_k))$, and the following conditions are satisfied:

$$\sum_{i=1}^c (\overleftarrow{C}_{ik}, \overrightarrow{C}_{ik}) = (\overleftarrow{1}, \overrightarrow{1}), (\overleftarrow{0}, \overrightarrow{0}) < \sum_{k=1}^m (\overleftarrow{C}_{ik}, \overrightarrow{C}_{ik}) < (\overleftarrow{n}, \overrightarrow{n}), \quad (\overleftarrow{C}_{ik}, \overrightarrow{C}_{ik}) \in [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})] \quad (5.13)$$

Rather than classify the samples directly, DFKNN is used to calculate the class membership of the sub-class samples. In addition, the membership of classified samples provides a certain degree of confidence in the classification results. For example, if

the membership with respect to a sample class is $(\overleftarrow{0.9}, \overrightarrow{0.9})$ and the membership of the other two categories is $(\overleftarrow{0.05}, \overrightarrow{0.05})$, then we can be reasonably certain that the sample belongs to the class of membership $(\overleftarrow{0.9}, \overrightarrow{0.9})$. However, if a sample belongs to one class with membership $(\overleftarrow{0.55}, \overrightarrow{0.55})$, to a second category with membership $(\overleftarrow{0.45}, \overrightarrow{0.45})$, and to a third category with membership $(\overleftarrow{0.1}, \overrightarrow{0.1})$, then, because membership to the first two categories is almost equivalent, we cannot determine the sample category. It is reasonable to believe that this sample does not belong to the third category, but the final determination of this sample's category requires further study. Obviously, we are confident that the introduction of membership is conducive to the improvement of classification results.

Assuming that the labelled sample set $(\overleftarrow{X}, \overrightarrow{X}) = \{(\overleftarrow{x}_1, \overrightarrow{x}_1), (\overleftarrow{x}_2, \overrightarrow{x}_2), \dots, (\overleftarrow{x}_n, \overrightarrow{x}_n)\}$ is composed of n labelled samples, that sample $(\overleftarrow{x}, \overrightarrow{x})$ has membership $(\overleftarrow{C}_i(\overleftarrow{x}), \overrightarrow{C}_i(\overrightarrow{x}))$ of the i th class $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ and that the j th sample in the sample tag set has membership $(\overleftarrow{C}_{ij}, \overrightarrow{C}_{ij})$ of the i th class, then the DFKNN algorithm is described as follows:

Algorithm 5.3 DFKNN

Enter an unknown class tag sample $(\overleftarrow{x}, \overrightarrow{x})$.
 Set the value of k , and k meets $1 \leq k \leq n$.
 Initialize $i = 1$.
Do until (find k neighbors of the sample $(\overleftarrow{x}, \overrightarrow{x})$ to be classified)
 Calculate the Mahalanobis distance of a sample $(\overleftarrow{x}, \overrightarrow{x})$ to be classified into a sample $(\overleftarrow{x}_i, \overrightarrow{x}_i)$.
If $(i \geq k)$ **then**
 The samples $(\overleftarrow{x}_i, \overrightarrow{x}_i)$ are added to the k -nearest neighbor set of the samples $(\overleftarrow{x}, \overrightarrow{x})$ to be classified.
Else if (The sample $(\overleftarrow{x}_i, \overrightarrow{x}_i)$ is closer to the nearest neighbor than the sample $(\overleftarrow{x}, \overrightarrow{x})$ to be classified) **then**
 Delete the farthest neighbor of the k nearest neighbor of the sample $(\overleftarrow{x}, \overrightarrow{x})$ to be classified;
 Add the samples $(\overleftarrow{x}_i, \overrightarrow{x}_i)$ to the k -nearest neighbor set of the samples $(\overleftarrow{x}, \overrightarrow{x})$ to be classified.
End if
End do until
 Initialize $i = 1$.
Do until (Samples $(\overleftarrow{x}, \overrightarrow{x})$ to be classified belonging to each category membership is completed)
 Using (5.14) to calculate $(\overleftarrow{C}_i(\overleftarrow{x}), \overrightarrow{C}_i(\overrightarrow{x}))$;
 Self-increment i .
End until

$$(\overleftarrow{C}_i(\overleftarrow{x}), \overrightarrow{C}_i(\overrightarrow{x})) = \frac{\sum_{j=1}^k ((\overleftarrow{C}_{ij}, \overrightarrow{C}_{ij})((\overleftarrow{1}, \overrightarrow{1})/d_M((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{x}_j, \overrightarrow{x}_j))^{2/m-1}))}{\sum_{j=1}^k ((\overleftarrow{1}, \overrightarrow{1})/d_M((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{x}_j, \overrightarrow{x}_j))^{2/m-1})} \quad (5.14)$$

$$d_M((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{z}, \overrightarrow{z})) = \sqrt{((\overleftarrow{x}, \overrightarrow{x}) - (\overleftarrow{z}, \overrightarrow{z}))^T M ((\overleftarrow{x}, \overrightarrow{x}) - (\overleftarrow{z}, \overrightarrow{z}))} \quad (5.15)$$

Among above, $d_M(\cdot, \cdot)$ is the Mahalanobis distance can be calculated using (5.15), and the matrix M is a positive semidefinite matrix, and how to get this positive definite matrix will be described in the following sections.

From (5.14), it can be seen that the membership degree of sample $(\overleftarrow{x}, \overrightarrow{x})$ is related to the reciprocal of the Mahalanobis distance to the nearest neighbour and the membership degree of these neighbours. If the sample to be classified is closer to its neighbours, then the reciprocal of the distance has a greater influence on the calculation of its membership; otherwise, its influence is slightly lower.

When calculating the membership, the variable m in (5.14) gives the weight of the distance directly. If $m = 2$, then the weight is the reciprocal of the distance; as m gradually becomes larger, the distance will gradually lose its influence; when m is close to 1, closer neighbourhood points are more important than farther neighbouring points, so the number of effective neighbourhood points will be greatly reduced when calculating the membership degree. In the following example, we use $m = 2$, but the experiment shows that the impact on the error rate is not large regardless of the value of m .

There are many ways to calculate the membership degree of labelled samples. In addition to using the common membership function introduced in Chapter 2 to calculate the membership degree, we have the following kinds of membership:

- (1) The membership degree of its known category is set $(\overleftarrow{1}, \overrightarrow{1})$, the membership of the remaining categories is set to $(\overleftarrow{0}, \overrightarrow{0})$;
- (2) Another method of assigning membership values to labelled samples is based on the procedure in [54]. This method is suitable for two types of problems. It is based on the distance of the mean value between the labelled sample and the labelled sample class, calculates the membership of the known class for the labelled sample, and calculates the subordinate degree $(\overleftarrow{1}, \overrightarrow{1})$ of the remaining categories to satisfy the membership of all categories of a sample. The details of the procedure can be found in [59].
- (3) According to the KNN rule for assigning the membership value of the labelled sample, where $K \neq k$ in classification k . Find k neighbours of the labelled sample $(\overleftarrow{x}, \overrightarrow{x})$ belonging to the i th category $(\overleftarrow{C}_i, \overrightarrow{C}_i)$. Then, using (5.16), we can calculate the membership of the labelled samples $(\overleftarrow{x}, \overrightarrow{x})$ belonging to each category:

$$(\overleftarrow{C}_j(\overleftarrow{x}), \overrightarrow{C}_j(\overrightarrow{x})) = \begin{cases} (\overleftarrow{0.51}, \overrightarrow{0.51}) + (n_j/K)^*(\overleftarrow{0.49}, \overrightarrow{0.49}), & \text{if } j = i \\ (n_j/K)^*(\overleftarrow{0.49}, \overrightarrow{0.49}), & \text{if } j \neq i \end{cases} \quad (5.16)$$

Among these, n_j is the number of neighbours of the marked samples $(\overleftarrow{x}, \overrightarrow{x})$ in the j th class. This approach attempts to dynamically fuzzify the membership of labelled samples located in the cross-section of the sample space and to classify the samples that are not in the cross-section of the sample space into a known class because of their high membership. Thus, an unknown sample located in the intersection region will be less affected by this type of labelled sample, which is located in the “dynamic blur” region of the category boundary.

5.5.3 Dynamic fuzzy semi-supervised adaptive learning algorithm

The categories in the dynamic deduction system are dynamically fuzzy, and the class characteristics change over time. They may change slowly or abruptly depending on the state of the system at that time. Some new models either further validate information in previous data or introduce new information, such as new categories, inter-class merging, splitting, and so on. This new information changes the operational state. Thus, in view of these small and transient changes, it is necessary to update the membership function of each category in order to better estimate the characteristics of the current system model. To identify these changes, we need an adaptive classifier that can adjust its parameters over time.

A dynamic fuzzy semi-supervised adaptive learning (DFSSAL) algorithm based on DFKNN is introduced in this section. DFSSAL uses the labelled information to estimate the class feature. The unlabelled data are used to detect the new class and learn the membership function of the class. Moreover, the class information can be learned in the feature space before taking into account the class deduction. The purpose of the algorithm is to judge whether the class can be deduced by considering the usefulness of the model and to estimate the new model of the system according to the adjusted class feature. As shown in Fig. 5.9, the main process of the algorithm can be described as follows.

- (1) Learning and classification: The DFKNN algorithm is used to learn and classify the new model.
- (2) Detect whether category can be deduced: After completing the classification of the new model, the algorithm gradually updates the category parameters in order to track the category deduction process. If the category is correctly deduced, and the corresponding deductive indication parameter confirms that the category has been deduced, the algorithm automatically creates a new category.
- (3) Determine the class validity: This stage validates the usefulness of the current category and retains useful categories. Categories that are similar are merged.

We now describe these stages in detail.

1) Learning and classification stage

The learning stage of the DFSSAL algorithm aims to learn all the marking patterns and class information. The training set is denoted by $(\vec{X}, \vec{X}) = \{((\vec{x}_1, \vec{x}_1), (\vec{y}_1, \vec{y}_1)), \dots, ((\vec{x}_l, \vec{x}_l), (\vec{y}_l, \vec{y}_l)), (\vec{x}_1', \vec{x}_1'), \dots, (\vec{x}_u', \vec{x}_u')\}$, where $((\vec{x}_i, \vec{x}_i), (\vec{y}_i, \vec{y}_i))$ is the i th labelled data point or pattern, (\vec{y}_i, \vec{y}_i) is the label, l is the number of labelled data, (\vec{x}_i', \vec{x}_i') is the i th unlabelled data point or pattern, and u is the number of unlabelled data.

The initial learning set consists of all the marked data in the training set. There must be at least two patterns in order to calculate the initial centre of gravity and standard deviation of each class. The DFSSAL algorithm uses the centre of gravity and standard deviation to evaluate the category deduction. The current centre of gravity

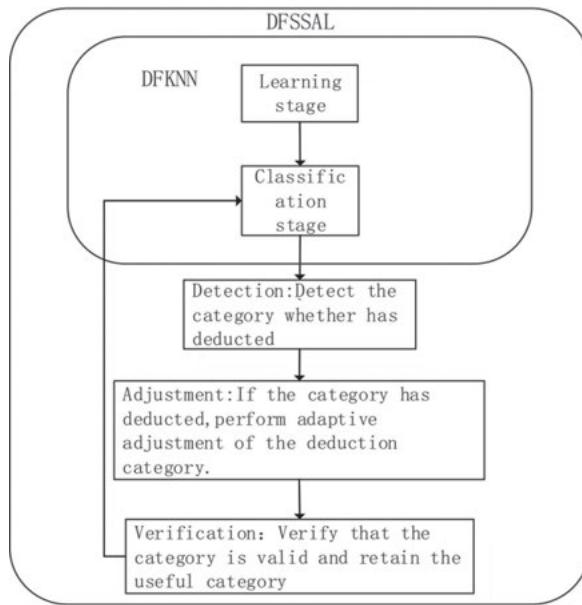


Fig. 5.9: Process of dynamic fuzzy semi-supervised adaptive learning algorithm.

$(\overleftarrow{G}_{k,cur}^a, \overrightarrow{G}_{k,cur}^a)$ and the initial standard deviation $(\overleftarrow{Std}_{k,ini}^a, \overrightarrow{Std}_{k,ini}^a)$ of the current category $(\overleftarrow{C}_k, \overrightarrow{C}_k)$ are calculated for each attribute $(\overleftarrow{a}, \overrightarrow{a})$ of each category.

DFKNN is used to classify each new model step by step. Therefore, we need to initialize K, classify a new pattern into a known category, and then use the category centre of gravity and standard deviation to determine whether the test category can be deduced.

2) Detect whether category can be deduced

In general, it is highly likely that deductions will be made for categories with new patterns. After classifying a pattern $(\overleftarrow{x}, \overrightarrow{x})$ into category $(\overleftarrow{C}_k, \overrightarrow{C}_k)$, only the parameters of category $(\overleftarrow{C}_k, \overrightarrow{C}_k)$ need to be updated. We calculate the new features of category $(\overleftarrow{C}_k, \overrightarrow{C}_k)$ (standard deviation and centre of gravity) to detect whether or not deduction has occurred in that category. Equation (5.17) is used to update the standard variance $(\overleftarrow{Std}_{k,cur}^a, \overrightarrow{Std}_{k,cur}^a)$ and the centre of gravity $(\overleftarrow{G}_{k,cur}^a, \overrightarrow{G}_{k,cur}^a)$ of category $(\overleftarrow{C}_k, \overrightarrow{C}_k)$:

$$\begin{aligned}
 & (\overleftarrow{Std}_{k,cur}^a, \overrightarrow{Std}_{k,cur}^a) \\
 &= \sqrt{\frac{n_k - 1}{n_k} \times ((\overleftarrow{Std}_{k,cur-1}^a)^2, (\overrightarrow{Std}_{k,cur-1}^a)^2) + \frac{((\overleftarrow{x}, \overrightarrow{x}) - (\overleftarrow{G}_{k,cur-1}^a, \overrightarrow{G}_{k,cur-1}^a))^2}{n_k + 1}} \quad (5.17) \\
 & (\overleftarrow{G}_{k,cur}^a, \overrightarrow{G}_{k,cur}^a) = \frac{(\overleftarrow{G}_{k,cur-1}^a, \overrightarrow{G}_{k,cur-1}^a) \times n_k}{n_k + 1} + \frac{(\overleftarrow{x}, \overrightarrow{x})}{n_k + 1}
 \end{aligned}$$

where n_k is the total number of new patterns $(\overleftarrow{x}, \overrightarrow{x})$ before entering category $(\overleftarrow{C}_k, \overrightarrow{C}_k)$. $(\overleftarrow{Std}_{k,cur-1}^a, \overrightarrow{Std}_{k,cur-1}^a)$ and $(\overleftarrow{G}_{k,cur-1}^a, \overrightarrow{G}_{k,cur-1}^a)$ are the standard variance and the centre of gravity, respectively, of the new pattern $(\overleftarrow{x}, \overrightarrow{x})$ according to attribute $(\overleftarrow{a}, \overrightarrow{a})$ before classification into category $(\overleftarrow{C}_k, \overrightarrow{C}_k)$.

Based on the calculated values of $(\overleftarrow{Std}_{k,cur}^a, \overrightarrow{Std}_{k,cur}^a)$ and $(\overleftarrow{G}_{k,cur}^a, \overrightarrow{G}_{k,cur}^a)$, a brief change in the system is monitored using two deductive indicators.

- (1) $i_{k,a}^1$ represents the change in compactness of each attribute $(\overleftarrow{a}, \overrightarrow{a})$ of class $(\overleftarrow{C}_k, \overrightarrow{C}_k)$ and is calculated using (5.18). If at least one attribute $(\overleftarrow{a}, \overrightarrow{a})$ has a value of $i_{k,a}^1$ greater than the threshold th_1 , class $(\overleftarrow{C}_k, \overrightarrow{C}_k)$ is considered to begin to change on attribute $(\overleftarrow{a}, \overrightarrow{a})$. To track small changes in the category, the threshold th_1 should be set to a smaller value; otherwise, to detect important deductions, th_1 should be set to a larger value. For example, when $th_1 = 5$, only 5% of the category characteristics can be deduced.

$$i_{k,a}^1 = \frac{(\overleftarrow{Std}_{k,cur}^a, \overrightarrow{Std}_{k,cur}^a) \times (\overleftarrow{100}, \overrightarrow{100})}{(\overleftarrow{Std}_{k,ini}^a, \overrightarrow{Std}_{k,ini}^a)} - (\overleftarrow{100}, \overrightarrow{100}) \quad (5.18)$$

- (2) $i_{k,a}^2$ represents the distance between the value $(x^a, \overrightarrow{x}^a)$ of pattern $(\overleftarrow{x}, \overrightarrow{x})$ on attribute $(\overleftarrow{a}, \overrightarrow{a})$ and the centre of gravity $(\overleftarrow{G}_{k,cur}^a, \overrightarrow{G}_{k,cur}^a)$ of the class according to the standard deviation $(\overleftarrow{Std}_{k,cur}^a, \overrightarrow{Std}_{k,cur}^a)$ of the current category $(\overleftarrow{C}_k, \overrightarrow{C}_k)$. This is calculated by (5.19). If at least one value of attribute $(\overleftarrow{a}, \overrightarrow{a})$ is greater than th_1 , then this pattern will not be in the same region as the other patterns in class $(\overleftarrow{C}_k, \overrightarrow{C}_k)$.

$$i_{k,a}^2 = \frac{|(\overleftarrow{x}^a, \overrightarrow{x}^a) - (\overleftarrow{G}_{k,cur}^a, \overrightarrow{G}_{k,cur}^a)|}{(\overleftarrow{Std}_{k,cur}^a, \overrightarrow{Std}_{k,cur}^a)} - (\overleftarrow{100}, \overrightarrow{100}) \quad (5.19)$$

Note that noise sometimes changes the centre of gravity of the category and the standard deviation, so we need to add an additional constraint: If at least $N_{k,\min}$ consecutive patterns in class $(\overleftarrow{C}_k, \overrightarrow{C}_k)$ are found to be deductive, then we can confirm that the category is deduced. If $N_{k,\min}$ is too large, the detection of the class evolution will be delayed, and we can determine the value of $N_{k,\min}$ according to the ratio of the noise in the pattern and the need to delay the evolution of the detection category. When two successive deductions indicate that the values of $i_{k,a}^1$ and $i_{k,a}^2$ are greater than th_1 , we ensure that the category has been deduced, then adjust the category characteristics as described below.

3) Adjustment of the category characteristics

When class changes are detected in the classification stage, the class characteristics may need to be adjusted. When the category is identified by deduction, the useful

model can be employed to create a new category. If there are now c categories, we adjust the class characteristics by the following steps.

- (1) Create a new category $(\overleftarrow{C}_{c+1}, \overrightarrow{C}_{c+1})$ and select the most representative model for deduction. $(\overleftarrow{x}, \overrightarrow{x})$ and its $k - 1$ neighbours are new members of the new class, as the last sample to be classified in class $(\overleftarrow{x}, \overrightarrow{x})$ is one of the patterns in which the class was deduced. Because the neighbours are already known at the time of class $(\overleftarrow{x}, \overrightarrow{x})$, there is no need to determine these neighbours by calculating the distance. In short, the k selected nodes constitute the new model of the new category.
- (2) The selected pattern is deleted from the original category $(\overleftarrow{C}_i, \overrightarrow{C}_i)$, and its new centre of gravity $(\overleftarrow{G}_{i,cur}^a, \overrightarrow{G}_{i,cur}^a)$ and current standard deviation $(\overleftarrow{Std}_{i,cur}^a, \overrightarrow{Std}_{i,cur}^a)$ are calculated.
- (3) Calculate the centre of gravity $(\overleftarrow{G}_{c+1,cur}^a, \overrightarrow{G}_{c+1,cur}^a)$ and standard deviation $(\overleftarrow{Std}_{c+1,cur}^a, \overrightarrow{Std}_{c+1,cur}^a)$ of the new category, and update the total number of categories.

4) Determine the class validity

Because noise is present in the training set, if a noise category is created, it should be removed during the validity phase. Categories that are deduced from one state to another are useless and should be removed to avoid excessive growth of the dataset. Category deletion is required when the following situations occur:

- (1) A category contains a small number of models, only slightly larger than k , that is, the number of categories is $n_1 > k$;
- (2) No new schema has been added to the category to be deleted, but there are enough n_2 patterns to appear in the other categories.

Sometimes, instead of deleting categories, similar categories should be merged. In fact, it is necessary to measure the coverage over time of existing categories and those that are created. If multiple categories have been created and the deductions for these new categories tend to be similar, then these categories should be merged. To determine the similarity between two categories for merging, a similarity measure based on fuzzy sets is used [53]. To deal with the dynamic fuzzy problem, we will use the similarity measure based on dynamic fuzzy sets, as shown in (5.20), which takes into account the degree of coverage between categories based on the membership of the classified model. Let $(\overleftarrow{C}_i(\overleftarrow{x}), \overrightarrow{C}_i(\overrightarrow{x}))$ and $(\overleftarrow{C}_z(\overleftarrow{x}), \overrightarrow{C}_z(\overrightarrow{x}))$ be the memberships of pattern $(\overleftarrow{x}, \overrightarrow{x})$, which belongs to category $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ and $(\overleftarrow{C}_z, \overrightarrow{C}_z)$, respectively, and let $(\overleftarrow{\delta}_{i,z}, \overrightarrow{\delta}_{i,z})$ be the similarity between the two categories. Values closer to $(\overleftarrow{1}, \overrightarrow{1})$ indicate that these two categories are more similar, but we do not need to wait until $(\overleftarrow{\delta}_{i,z}, \overrightarrow{\delta}_{i,z}) = (\overleftarrow{1}, \overrightarrow{1})$ before merging $(\overleftarrow{C}_i, \overrightarrow{C}_i)$ and $(\overleftarrow{C}_z, \overrightarrow{C}_z)$. We can merge two categories of operations when the similarity is above a certain threshold of $(\overleftarrow{\delta}_{i,z}, \overrightarrow{\delta}_{i,z})$.

$$(\overleftarrow{\delta}_{i,z}, \overrightarrow{\delta}_{i,z}) = (\overleftarrow{1}, \overrightarrow{1}) - \frac{\sum_{(\overleftarrow{x}, \overrightarrow{x}) \in (\overleftarrow{C}_i, \overrightarrow{C}_i) \text{ or } (\overleftarrow{x}, \overrightarrow{x}) \in (\overleftarrow{C}_z, \overrightarrow{C}_z)} |(\overleftarrow{C}_i(\overleftarrow{x}), \overrightarrow{C}_i(\overrightarrow{x})) - (\overleftarrow{C}_z(\overleftarrow{x}), \overrightarrow{C}_z(\overrightarrow{x}))|}{\sum_{(\overleftarrow{x}, \overrightarrow{x}) \in (\overleftarrow{C}_i, \overrightarrow{C}_i)} (\overleftarrow{C}_i(\overleftarrow{x}), \overrightarrow{C}_i(\overrightarrow{x})) + \sum_{(\overleftarrow{x}, \overrightarrow{x}) \in (\overleftarrow{C}_z, \overrightarrow{C}_z)} (\overleftarrow{C}_z(\overleftarrow{x}), \overrightarrow{C}_z(\overrightarrow{x}))} \quad (5.20)$$

1. DFSSMTAL algorithm

The Mahalanobis distance metric is an important parameter in both the DFKNN algorithm and the DFSSAL algorithm. Many scholars have proposed various methods of estimating the Mahalanobis distance in order to calculate the distance between samples in KNN classification. There are three main categories [55]: eigenvector methods, convex optimization methods, and neighbour component analysis methods.

Eigenvector methods are often used to solve the linear transformation of the input space. As mentioned in Section 4.1, the Mahalanobis distance metric can be obtained by a linear transformation. Principal component analysis, linear discriminant analysis, and correlation analysis are common feature vector methods. The convex optimization method formalizes distance metric learning as the convex optimization of the key positive semidefinite matrix M . Nearest neighbour component analysis [60] is a novel supervised learning algorithm proposed by Goldberger et al. [6]. To reduce the misclassification rate, the classifier uses the Mahalanobis distance metric as a parameter; that is, the expected misclassification rate is reduced by estimating the linear transformation matrix L . In this section, we discuss how to study the Mahalanobis distance metric from the perspective of DFSSMTL.

1) Mahalanobis distance measurement for single task learning

For a single task, the KNNs of a sample X are labelled with the same category using the trained Mahalanobis distance, and differently labelled samples are separated from X . We call the samples with different labels the pseudo-neighbours of X , and those with the same label as sample X are the target neighbours. Assuming that the sample is a target neighbourhood of the sample, it is a pseudo-neighbourhood of the sample, and the learned Markov metric is combined with a larger interval such that each input sample is closer to its target neighbourhood than its pseudo-neighbourhood. The following linear inequality describes this relationship:

$$d_M^2((\overleftarrow{x}_i, \overrightarrow{x}_i), (\overleftarrow{x}_k, \overrightarrow{x}_k)) - d_M^2((\overleftarrow{x}_i, \overrightarrow{x}_i), (\overleftarrow{x}_j, \overrightarrow{x}_j)) \geq 1 \quad (5.21)$$

As shown in Fig. 5.10, all samples on the circle are close to sample E and can be the KNNs of sample X. Under the influence of the Mahalanobis distance, the circle becomes elliptical, so that the pseudo-neighbour point and the target neighbourhood point of sample X become larger ($K = 3$) at the same time.

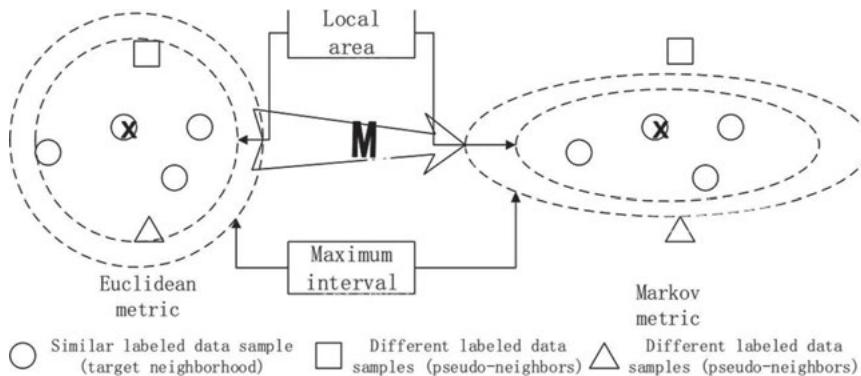


Fig. 5.10: The changes in neighbourhood of sample X before and after the calculation of Mahalanobis distance.

2) Mahalanobis distance measurement in multi-task background

Assuming that there are T tasks, the training data in each task consist of unlabelled data and tagged data, and the training dataset for the t th task is written as

$$(\overleftarrow{X}, \overrightarrow{X})_t = \{((\overleftarrow{x}_1, \overrightarrow{x}_1)_t, (\overleftarrow{y}_1, \overrightarrow{y}_1)_t), \dots, ((\overleftarrow{x}_l, \overrightarrow{x}_l)_t, (\overleftarrow{y}_l, \overrightarrow{y}_l)_t), (\overleftarrow{x}'_1, \overrightarrow{x}'_1)_t, \dots, (\overleftarrow{x}'_u, \overrightarrow{x}'_u)_t\},$$

where $((x_i, x_i)_t, (y_i, y_i)_t)$ denotes the i th tag data $(\overleftarrow{x}_i, \overrightarrow{x}_i)_t$, which is $(\overleftarrow{y}_i, \overrightarrow{y}_i)_t$ in the t th task, $(\overleftarrow{x}'_i, \overrightarrow{x}'_i)$ denotes the i th unmarked data of the t th task, l_t denotes the number of tag data of the t th task, and u_t denotes the number of untagged data for t tasks. The main goal of the DFSSMTAL algorithm is to learn a metric for each task (the metric of the Mahalanobis distance that is learned by the first task is denoted by $d_{M1}(\cdot, \cdot)$). Then, the task can be learned through DFSSAL. Similar points for all tasks are modelled using the Mahalanobis distance $d_{M0}(\cdot, \cdot)$, which is shared by all tasks, and the Mahalanobis distance $d_{M0}(\cdot, \cdot)$ is calculated from the matrix of semi-definite matrices M_0 . The Mahalanobis distance of each task preserves the individual properties of each task, with the Mahalanobis distance metric for the t th task denoted by $d_{Mt}(\cdot, \cdot)$ and calculated from the semi-definite matrix M_t . The Mahalanobis distance defining the t th task is given by the following equation:

$$d_{Mt}((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{z}, \overrightarrow{z})) = \sqrt{((\overleftarrow{x}, \overrightarrow{x}) - (\overleftarrow{z}, \overrightarrow{z}))^T (M_0 + M_t) ((\overleftarrow{x}, \overrightarrow{x}) - (\overleftarrow{z}, \overrightarrow{z}))} \quad (5.22)$$

As shown in Fig. 5.11, M_0 grasps the overall trend of the task data, whereas M_t reflects specific tasks.

Before the DFSSMTAL algorithm is presented, we introduce some notations:

- (1) $(j \rightarrow i)_t$ indicates that sample $(\overleftarrow{x}_j, \overrightarrow{x}_j)_t$ of the t th task is a target neighbour of sample $(\overleftarrow{x}_i, \overrightarrow{x}_i)_t$.

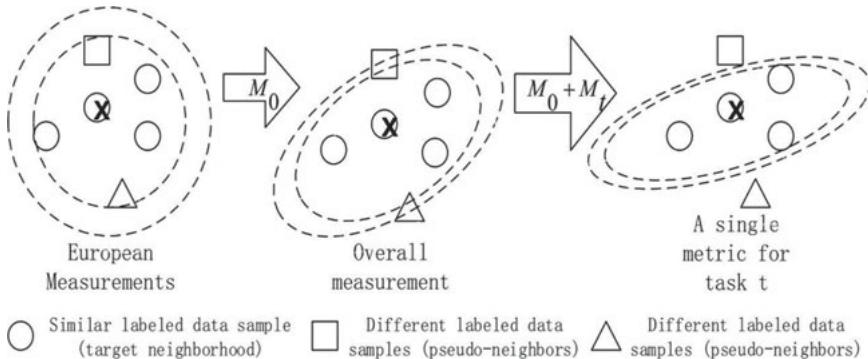


Fig. 5.11: Effects of Mahalanobis distance measurements on nearest neighbours of sample.

- (2) $(k \rightarrow i)_t$ indicates that sample $(\bar{x}_k, \vec{x}_k)_t$ of the t th task is in the pseudo-neighbourhood of sample $(\bar{x}_i, \vec{x}_i)_t$.
- (3) $S_t = \{(i, j, k)_t | (j \rightarrow i)_t, (k \rightarrow i)_t\}$ denotes the tuple consisting of subscripts of a sample and its neighbours and pseudo-neighbours.

The main task of the DFSSMTAL algorithm is to study the $T + 1$ positive semidefinite matrices M_0, M_1, \dots, M_T (M_0 contains the shared parameters of all tasks and M_1, \dots, M_T contain the respective parameters of the T tasks) with the aim of minimizing the overall cost of the algorithm. One of the key points in multi-task learning is the need to properly integrate multiple learning tasks, so we must ensure that the algorithm does not over-emphasize M_0 or M_t . To balance the two and minimize the cost of the problem, the following algorithm is used [12].

Algorithm 5.4 Dynamic fuzzy semi-supervised multi-task adaptive learning

First, solve the problem of (5.23), which can refer to the solution in [55]:

$$\min_{M_0, \dots, M_T} \gamma_0 \|M_0 - I\|_F^2 + \sum_{t=1}^T \left[\gamma_t \|M_t\|_F^2 + \sum_{(j \rightarrow i)_t} d_{M_t}^2((\bar{x}_i, \vec{x}_i)_t, (\bar{x}_j, \vec{x}_j)_t) + \sum_{(i, j, k)_t \in S_t} \xi_{ijk}^t \right] \quad (5.23)$$

where $\forall t, \forall (i, j, k) \in S_t$ has the following constraints:

- (1) $d_{M_t}^2((\bar{x}_i, \vec{x}_i)_t, (\bar{x}_k, \vec{x}_k)_t) - d_{M_t}^2((\bar{x}_i, \vec{x}_i)_t, (\bar{x}_j, \vec{x}_j)_t) \geq 1 - \xi_{ijk}^t$;
where ξ_{ijk}^t is a non-negative slack variable with a smaller value, that is $\xi_{ijk}^t > 0$. The learned matrices make a sample in the t -th task closer to its target neighborhood $(\bar{x}_i, \vec{x}_i)_t$ than its pseudo-neighborhood $(\bar{x}_k, \vec{x}_k)_t$.
- (2) $M_0, M_1, \dots, M_T \succeq 0$, that is the $T + 1$ matrices M are semi-definite matrices.

(5.23) is a convex optimization problem, the proof of which can be referred to [61].

Second, using (5.23) to solve the $T + 1$ positive semidefinite matrices M_0, M_1, \dots, M_T , the dynamic fuzzy semi-supervised adaptive learning is applied to the task, and finally a dynamic fuzzy semi-supervised multi-task adaptive learning algorithm is constructed.

2. Case analysis

In this paper, DFSSAL is presented in the context of a single task and multiple tasks. The experimental results are evaluated in terms of the classification error rate, i.e. the number of misclassified samples in the test data set as a percentage of the total number of test data.

1) DFSSAL under single task

DFSSAL of a single task was examined using the Balance Scale dataset from the UCI repository. To model the results of psychological experiments, Siegler and others created the Balance Scale dataset. The dataset consists of three categories (left-hand, balance, and right-hand side) and four attributes (left-side weight, left-hand side, right-side weight, and right-side distance) and contains 625 samples (288 in the partial left category, 49 in the balance category, and the remainder in the right-side category). The category is determined by the product of the left-hand weight and the left-hand side and the product of the right-hand weight and the right-hand side distance. If they are equal, they belong to the balanced category; otherwise, they belong to the larger product.

To illustrate the feasibility and effectiveness of DFSSAL, we compared our algorithm with the large-interval nearest neighbour algorithm (LMNN) [55, 62]. LMNN is a supervised learning method for learning the Mahalanobis distance metric, and its classification method uses KNN. Figure 5.12 shows the results for different distance

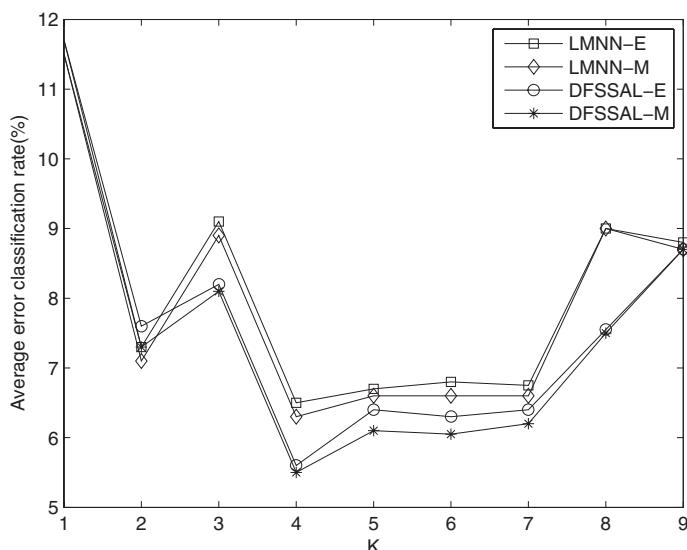


Fig. 5.12: Error classification rate when the K is a different integer and the LMNN and DFSSAL methods use Euclidean distance and Markov distance, where XXX-E and XXX-M represent respectively the XXX method which based on the Euclidean distance and Mahalanobis distance.

measures and different values of K , where the proportion of training data in the training set was set to 0.4. We randomly selected the average classification error rate of the labelled data to form the training set and ran the experiment 50 times. In the experiment, we used the third membership function given by (5.16). In fact, we found that the selection of the membership function has no significant effect on the experimental results. It is easy to see that the error classification rate of DFSSAL is lower than that of LMNN based on both the Euclidean distance and Mahalanobis distance. When K is set in the range 5–7, the error rate of classification changes is more stable.

Secondly, when the training set contains a different number of marked sample data, the average error classification rate of the two methods is different for the Balance Scale dataset, as shown in Fig. 5.13. In this experiment, K was set to 5, and the two algorithms were randomly selected and run 50 times. The average error rate of the two methods decreases as the proportion of training sample data increases, and the error rate of DFSSAL is again lower than that of LMNN.

In short, from the experimental comparison, it is clear that, if the DFSSAL algorithm parameters are appropriate, we can obtain good classification results.

2) Dynamic fuzzy semi-supervised multi-task adaptive learning

Dynamic fuzzy semi-supervised multi-task adaptive learning (DFSSMTAL) was investigated using the Isolet speech dataset from the UCI repository. The Isolet dataset contains 26 alphabetical pronunciation data for 150 people, each of whom spelled the letters twice. This means that there are 52 training samples per person, with a total

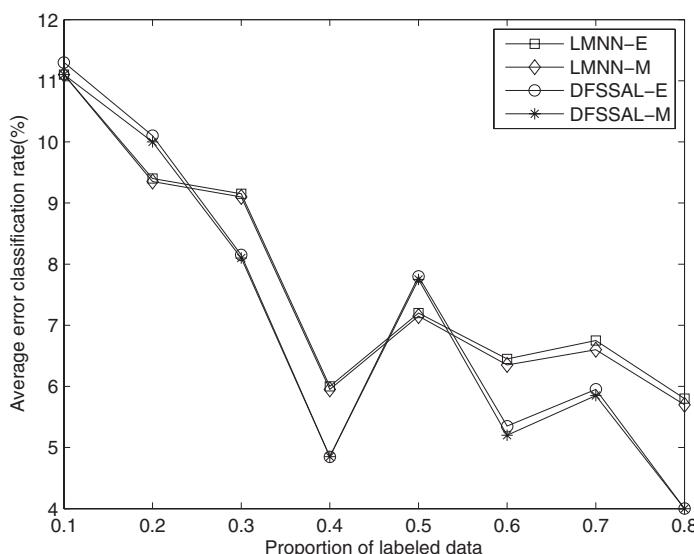


Fig. 5.13: The wrong classification rate of two methods when k is 5 and the number of marked data in the training set is scaled to different values.

Tab. 5.4: Average error classification rate for various algorithms in different backgrounds.

Number	Single task		Multiple tasks	
	LMNN (%)	DFSSAL (%)	MTLMNN (%)	DFSSMTAL (%)
1	13.3333	12.7778	12.5	11.667
2	11.4583	10	10.833	9.16667
3	17.5	15	15.833	13.5417
4	15.8333	14.0167	13.9276	12.2563
5	11.6667	10.833	8.95833	7.77778

of 7797 sample data (three of which are unavailable for historic reasons). In other words, Isolet contains 26 categories of sample data, with an average of 300 sample data for each class, and each class of sample data has 617 attribute characteristics. A description of each property can be found in [63]. Thirty of the pronunciation data constitute subsets of the sample data.

In fact, these five sample data subsets can be thought of as distinct but related sorting tasks, with each task being to identify which of the 26 categories the current utterance data belong to. Obviously, because each group of members has different pronunciations, the five tasks are different. However, the five tasks are related, mainly because the pronunciation data are from the same English alphabet. Therefore, this naturally constitutes multi-task learning.

To reduce the noise and improve the computing speed, we first applied PCA [64] to reduce the dimension of the Isolet data and map it to the principal component subspace. This enabled us to reduce the number of features from 617 to 169. In the experiment, similar to the previous single-task scenario, we randomly selected the sample data as a certain proportion of labelled data (that is, the training set may not contain some categories of sample data, which must be detected dynamically) and ran the experiment 10 times to obtain the average error rate. Table 5.4 summarizes the average error classification rate of the five tasks based on the Mahalanobis distance in the single task and multi-task context when the ratio of marked data is 0.4 and $K = 3$. In the table, MTLMNN is the LMNN in the multi-task learning background [6]. In the experiment, we used the third membership function given by (5.16). Again, we found that the selection of the membership function had no significant effect on the experimental results. As can be seen from the table, the DFSSAL algorithm and the DFSSMTAL algorithm are superior to the large-interval algorithms for single and multi-task learning, respectively.

5.6 Summary

Semi-supervised multi-task learning is mainly considered from two aspects: using information from a large amount of unlabelled data to learn the current task and

taking useful information from related tasks to study the whole problem. Dynamic fuzzy set theory is an effective theoretical tool for dealing with dynamic fuzzy problems. The combination of the two forms the semi-supervised multi-task learning model based on dynamic fuzzy sets. We have discussed the research status of semi-supervised learning, multi-task learning, and semi-supervised multi-task learning and classified the main semi-supervised multi-task learning algorithms. Combined with dynamic fuzzy set theory, this chapter introduced the dynamic fuzzy theory tool of the membership degree for classifying sample data. We proposed a semi-supervised multi-task learning model based on dynamic fuzzy sets, as well as dynamic fuzzy semi-supervised multi-task matching and DFSSMTAL algorithms and presented a simple experimental comparison to verify their feasibility and effectiveness.

References

- [1] Chapelle O, Schölkopf B, Zien A. *Semi-supervised learning*. London, UK, MIT Press, 2006.
- [2] Zhu XJ. *Semi-supervised learning reference survey*. Technical Report 1530, Department of Computer Science, University of Wisconsin at Madison, Madison, 2008.
- [3] Zhu XJ, Goldberg AB. *Introduction to semi-supervised learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool, 2009.
- [4] Shiga M, Mamitsuka H. Efficient semi-supervised learning on locally informative multiple graphs. *Pattern Recognition*, 2012, 45(3): 1035–1049.
- [5] Silva TC, Zhao L. Semi-supervised learning guided by the modularity measure in complex networks. *Neurocomputing*, 2012, 78(1):30–37.
- [6] Caruana R. Multitask learning. *Machine Learning*, 1997, 28: 41–75.
- [7] Baxter J. A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 1997, 28(1): 7–39.
- [8] Thrun S. Is learning the n-th thing any easier than learning the first?[C]// *International Conference on Neural Information Processing Systems*. MIT Press, 1995:640–646.
- [9] Chang Y, Bai J, Zhou K, et al. Multi-task learning to rank for web search. *Pattern Recognition Letters*, 2012, 33(2): 173–181.
- [10] Zhang DQ, Shen DG. Multi-modal multi-task learning for joint pre-diction of multiple regression and classification variables in Alzheimer’s disease. *NeuroImage*, 2012, 59(2): 895–907.
- [11] Schölkopf, B, Platt, J, Hofmann, T. *Multi-Task Feature Learning*[M]// *Advances in Neural Information Processing Systems 19:Proceedings of the 2006 Conference*. MIT Press, 2007:41–48.
- [12] Evgeniou T, Pontil M. Regularized multi-task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.
- [13] Bakker B, Heskes T. Task clustering and gating for Bayesian multi-task learning. *Journal of Machine Learning Research*, 2003, 4: 83–99.
- [14] Xue Y, Liao XJ, Carin L. Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research*, 2007, 8: 35–63.
- [15] Lawrence N, Platt J. Learning to learn with the informative vector machine. In *Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.
- [16] Yu K, Tresp V, Schwaighofer A. Learning Gaussian process from multiple tasks. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, 2005.

- [17] Schwaighofer A, Tresp V, Yu K. Learning Gaussian process kernels via hierarchical Bayes. In Advances in Neural Information Processing Systems, 2005.
- [18] Bonilla EV, Chai KMA, Williams CKI. Multi-task Gaussian Process Prediction[C]// Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December. DBLP, 2008.
- [19] Ando RK, Zhang T. A framework for learning predictive structures from multiple tasks and unlabeled data. Journal of Machine Learning Research, 2005, 6: 1817–1853.
- [20] Liu QH, Liao XJ, Carin L. Semi-supervised life-long learning with application to sensing. In Proceedings of the 2nd IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, 2007.
- [21] Liu Q, Liao X, Hui LC, et al. Semi-supervised Multitask Learning[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2009, 31(6):1074–1086.
- [22] Liu QH, Liao XJ, Li H, et al. Semi-supervised multitask learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2009, 31(6): 1074–1086.
- [23] Li H, Liao XJ, Carin L. Active learning for semi-supervised multi-task learning. In Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 2009.
- [24] Zhang Y, Yeung DY. Semi-supervised multi-task regression. In Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II, 2009.
- [25] Loeff N, Farhadi A, Endres I, et al. Unlabeled data improves word prediction. In Proceedings of IEEE 12th International Conference on Computer Vision, 2009.
- [26] Wang F, Wang X, Tao Li. Semi-supervised multi-task learning with task regularizations. In Proceedings of IEEE International Conference on Data Mining, 2009.
- [27] Jacob L, Bach F. Clustered multi-task learning: A convex formulation. In Advances in Neural Information Processing Systems, 2008.
- [28] Chen Y, Lu Y, Lan M, et al. A semi-supervised method for classification of semantic relation between nominals. In Proceedings of IEEE International Conference on Asian Language Processing, 2010.
- [29] Diligenti M, Gori M, Maggini M, et al. Bridging logic and kernel machines. Machine Learning, 2012, 86(1): 57–88.
- [30] Li FZ. Coordinated machine learning model based on DFS. Computer Engineering, 2001, 27(3): 106–110.
- [31] Li FZ. Research on the stability of coordinated machine learning. Journal of Chinese Mini-Micro Computer Systems, 2002, 23(3): 314–317.
- [32] Xie L, Li FZ. Study on a class of dynamic fuzzy machine learning algorithms. Acta Electronica Sinica, 2008, 36(12A): 114–119.
- [33] Xie L, Li FZ. Study on dynamic fuzzy parameter learning algorithm. Microelectronics and Computer Science, 2008, 25(9): 84–87.
- [34] Zhang J, Li FZ. Machine learning model based on dynamic fuzzy sets (DFS) and its validation. Journal of Computational Information Systems, 2005, 1(4): 871–877.
- [35] Dai MY, Li FZ. Dynamic fuzzy semisupervised multitask learning. In Proceedings of 2011 Seventh International Conference on Computational Intelligence and Security, 2011.
- [36] Li FZ, Qian XP, Xie L, He SP. Machine learning theory and its applications. Hefei, China, University of Science and Technology of China Press, 2004.
- [37] Elworthy D. Does Baum-Welch re-estimation help taggers?. In Proceedings of the 4th ACL Conference on Applied Natural Language Proceeding, 1994.
- [38] Cozman FG, Cohen I, Cirelo MC. Semi-supervised learning of mixture models. In Proceedings of the 20th International Conference on Machine Learning, 2003.
- [39] Baxter J. Learning internal representations. In Proceedings of the 8th Annual Conference On Computational Learning Theory, 1995.

- [40] Baxter J. A model of inductive bias learning. *Journal Of Artificial Intelligence Research*, 2000, 12: 149–198.
- [41] Yu K, Schwaighofer A, Tresp V. Collaborative ensemble learning: Combining collaborative and content-based information filtering via hierarchical Bayes. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, 2003.
- [42] Yu K, Tresp V, Yu S. A nonparametric hierarchical Bayesian framework for information filtering. In *Proceedings of the 27th Annual International ACM SIGIR Conference On Research And Development In Information Retrieval*, 2004.
- [43] Zhang J, Ghahramani Z, Yang YM. Learning multiple related tasks using latent independent component analysis. In *Advances in Neural Information Processing Systems*, 2006.
- [44] Thrun S, O'Sullivan J. Discovering structure in multiple learning tasks: The TC algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, 1996.
- [45] Evgeniou T, Micchelli CA, Massimiliano P. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 2005, 6: 615–637.
- [46] Pan SJ, Yang Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 2010, 22(10): 1345–1359.
- [47] Huang PP, Wang G, Qin SY. Boosting for transfer learning from multiple data sources. *Pattern Recognition Letters*, 2012, 33(5): 568–579.
- [48] Zadeh LA. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1999, 100: 9–34.
- [49] Dubois D, Prade H. On possibility/probability transformation. *Fuzzy Logic*, 1993: 103–112.
- [50] Mouchaweh MS. Semi-supervised classification method for dynamic applications. *Fuzzy Sets and Systems*, 2010, 161: 544–563.
- [51] Mouchaweh MS. Variable probability-possibility transformation for the diagnosis by pattern recognition. *International Journal of Computational Intelligence Theory and Practice*, 2006, 1(1): 9–21.
- [52] Frigui H, Krishnapuram R. A robust algorithm for automatic extraction of an unknown number of clusters from noisy data. *Pattern Recognition Letters*, 1996, 17: 1223–1232.
- [53] Ozawa S, Roy A, Roussinov D. A Multitask learning model for online pattern recognition. *IEEE Transactions on Neural Networks*, 2009, 20(3): 430–445.
- [54] Weinberger KQ, Saul LK. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 2009, 10: 207–244.
- [55] Cover T, Hart P. Nearest neighbor pattern classification. *IEEE Transactions in Information Theory*, 1967, 13: 21–27.
- [56] Belongie S, Malik J, Puzicha J. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, 24(4): 509–522.
- [57] Simard P, Lecun Y, Denker JS. Efficient Pattern Recognition Using a New Transformation Distance[M]// *Advances in Neural Information Processing Systems (NIPS 1992)*. 1992:50–58.
- [58] Keller JM, Hunt DJ. Incorporating fuzzy membership functions into the perceptron algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1985, 7(6): 693–699.
- [59] Goldberger J, Roweis S, Hinton G, et al. Neighborhood component analysis. In *Advances in Neural Information Processing System*, 2005:513–520.
- [60] Parameswaran S, Weinberger KQ. Large Margin Multi-Task Metric Learning.[J]. *Advances in Neural Information Processing Systems*, 2010:1867–1875.
- [61] Weinberger KQ, Saul LK. Distance Metric Learning for Large Margin Nearest Neighbor Classification[J]. *Journal of Machine Learning Research*, 2009, 10(1):207–244.
- [62] Fanty M, Cole R. Spoken Letter Recognition.[C]// *Advances in Neural Information Processing Systems*. DBLP, 1990:220–226.
- [63] Jolliffe IT. *Principal component analysis*. New York, USA, Springer Verlag, 2002.

6 Dynamic fuzzy hierarchical relationships

This chapter is divided into seven sections. Section 6.1 gives a brief introduction to this chapter. In Section 6.2, we present the ILP. In Section 6.3, we introduce the dynamic fuzzy hierarchical relational learning (HRL). Section 6.4 presents the dynamic fuzzy tree hierarchical relation learning. In Section 6.5, we provide the dynamic fuzzy graph hierarchical relationship learning. In Section 6.6, we introduce the sample application and analysis. The summary of this chapter is in Section 6.7.

6.1 Introduction

6.1.1 Research progress of relationship learning

Aiming at the limitations of traditional symbolic machine learning, relationship learning combined with machine learning and knowledge representation, using logic as a representation language to describe the data and generalization, is the study of learning problems including multi-entities and relationship. Logic has a strong ability to describe the complex relationship between objects, which also makes ILP as the main means of mining relational schema [1]. The goal is to obtain specific information in the relational data and then use the knowledge to make inference, prediction, and classification. At present, it has been widely used in the field of computational biology and chemistry and so on, which has been widely used in the field of data mining [2].

According to the different techniques used in the study, relationship learning can be divided into SRL, HRL, graph-based relational learning (GBRL), and DFRL. The main research includes the following.

6.1.1.1 SRL

The SRL method, which combines the likelihood relation representation theory and machine learning phase, can better understand the relationship between the data of complex problems in the real world. Due to the characteristics of statistical learning theory, the research of SRL has attracted much attention in recent years.

In 1995, Sato proposed the PRISM program [3]. The PRISM program is a symbolic statistical modelling language; it is not only the probability expansion of logic program but also uses EM algorithm to learn from examples; the Muggleton proposed stochastic logic programs (SLP) [4], through each clause in a stochastic context-free grammar with a probability value, directly to upgrade; Kramer proposed a new algorithm of SRTs tree structure degradation [5], combining degenerate statistical methods with ILP. In 1998, Blockeel and Raedt proposed the first-order logic framework, the logic decision tree FOLDT's top-down induction [6]; its expression ability was significantly enhanced and supports for the invention predicate, the existence

of classifiers, and the full name of the classifier mixture. According to the decision tree, we cannot consider the deficiency of the mining object structure. In 1999, Knobbe, Siebes, and Wallen proposed the MRDTI framework, which can efficiently find multi-relational decision tree is a domain knowledge mined by encoding [7].

In 1997, Ngo and Haddawy proposed an SRL method based on Bayesian network [8], combining logic and Bayesian network, extending the Bayesian net by the definition of probability in first-order logic or interpretation. On this basis, Kersting and De Raedt proposed a BLP [9] in 2000; the model, through the establishment of model base atoms and random variables by one-by-one mapping, combines Bayesian net and clause logic together and solves the problem of continuous valued variables, and the combination rule is used to solve the multiple instantiation clause with the same head. The same year, the two-stage [10] algorithm that Muggleton proposed to maximize Bayesian posterior probability can learn the basic parameters and the logic program of SLP. In 2001, Cussens proposed the FAM [11] algorithm, based on the premise of the given logic program, estimating the parameters of SLP. In 2002, ILP and Muggleton can efficiently calculated the best probability parameters of a single clause, on the assumption that the clause was independently constructed, based on statistical techniques, learning [12] method parameters, and SLP structure. In 2003, Neville et al. proposed a relational probability tree learning algorithm [13], which deals with heterogeneous and interdependent data for the relationship framework. In the same year, Bernstein, Clearwater, and Provost proposed a relational vector space model [14], which can abstract the link structure and use the weight vector to represent entities, which is mainly used to solve the classification problem of link data. In 2004, Angelopoulos and Cussens gave the implementation method of MCMC on SLP [15], which provided a general tool for Bayesian learning.

In order to flexibly represent the relationship characteristics of data in the real world, Taskar et al. and Matthew et al., in 2002 and 2004, respectively, put forward the RMN [16] and MLN [17], respectively; in the same year, Anderson, Domingos, and Weld put relationship (logic) into the Markov model and proposed RMM [18]. RMM allows the state variable value to have a variety of types, and the same type of state is represented by a predicate or relationship, which overcomes the shortcomings of the Markov model with only one variable of state variables. RMM is a probabilistic relation representation combining the probability and the predicate calculus. In 2003, Kersting et al. introduced first-order logic into the Markov model and put forward the LOHMM [19]. The difference between the model and the Markov model is that it has two variables, the state and the observed value, and the parameters not only has the transition probability but also has the occurrence probability of each observation value in each state.

In 2005, Davis et al. [20] and Lanwehr et al. [21] combined Bayesian network and ILP technology to develop the SAYU and nFOIL system. Sato and Kameya proposed a new learning model based on Cussens's FAM algorithm and program transformation technology; in this mode, we can add limit to the PRISM program, in order to improve

the efficiency of [22] learning. In 2006, Izumi et al. proposed a parallel EM algorithm for data [23], to reduce the computing time, and memory space can be extended to learn by using multiple computers. Xu et al. proposed the Wireless Hidden Relation model [24], which, through the introduction of infinite dimensional hidden variables for each entity, expands the relationship model, studies the inference algorithm of the model, and discusses the necessity that the dissemination of information in the hidden variables can reduce structure learning in large range. Singla and Domingos proposed an entity resolution method [25] based on Markov logic; the method can combine the existing methods and Markov logic, so as to effectively solve the problem of learning and reasoning. Kersting et al. improved the basic HMM inference algorithm, giving conversion probability estimation and choosing the appropriate distribution, so as to realize the reasoning algorithm of LOHMM [26], and verified in the field of biological information the effectiveness of the algorithm. Chen and Muggleton compared BLP and SLP [27], analysed the difference and relationship between the semantics from two BLP and SLP from the definition of probability, and discussed the conversion between the two methods and their learnability.

In 2007, Singla and Domingos used the Gibbs framework to extend the Markov logic to an infinite domain [28]. Aiming at MLN can only deal with the discrete variables and attributes of the problem. Wang and Domingos proposed the MLN hybrid method [29], supporting continuous attributes and functions, in 2008. In 2009, Petr Buryan et al. [30] put forward the representation of concept based on grammar framework which can be used in the relationship learning random search, in order to ensure the correct concept of the search operation. In 2011, Man Zhu used the SRL method to establish the potential of the task of learning algorithms, for the analysis of the noise of link data learning out of the OWL axioms [31].

The SRL method is widely used, but there are still some problems needing further research: a comparative study on the advantages and disadvantages of different methods, the unified standard, and the mutual relationship, then get the unified framework; the ability of the practical metadata and self-describing data, learning from a variety of sources; the expansion of application, as the application is still limited, and combining with other systems or technologies can improve the intelligence level of the whole system; and pay attention to the characteristics of relational data and how to use the characteristics of data association to improve the efficiency of data aggregation, learning, or reasoning.

6.1.1.2 HRL

Deep architecture [32] is a learning model that contains multiple levels of nonlinear operations, such as neural networks with multiple hidden layers. Neural networks with only one hidden layer, as well as SVMs and other nuclear methods, are classified as shallow structures. The deep structure of the training is a very difficult problem that has not been effectively resolved for many years.

In 2006, Hinton et al. [33] made a breakthrough in deep structure training. They proposed a training algorithm for the structure of deep belief networks (DBNs). The core idea of the training method is that DBN carries out hierarchical unsupervised learning to adjust the weights of each layer, so that each component of the training is essentially a restricted Boltzmann machine for unsupervised training. Starting from the lowest level, take the training samples as input to learn the layer weights, and then output the value of the node on this layer after training the upper layer. Finally, train the upper layer until the training of the highest level is complete. After training in the DBN, the weights of each layer are set as the initial weights of the corresponding structure in the deep neural network, and the BP algorithm is used for supervised training. Finally, a deep neural network that can be used for further classification is obtained. The deep neural network obtained by the hierarchical non-supervised pre-training achieved the highest recognition rate in the handwritten digit recognition problem. In 2007, Bengio et al. [34] analysed the results of DBN hierarchical non-supervised pre-training and found that this produced weights that may be located near some real local optimal value, compared with random initial weights, because of an improved starting position. Hence, the BP algorithm based on gradient descent has a good optimization effect. In addition, Bengio et al. proposed a deep structure of another type, named the Stacked Auto-encoder. This superposition of multiple auto-encoders uses the same hierarchical unsupervised learning to train the sequence of auto-encoders. Each auto-encoder's training goal is to minimize the reconstruction error after encoding and decoding the sample with the original input data and to adjust the weights by gradient descent optimization to give the internal representation of the samples through unsupervised learning.

In 2009, the research team of Bengio [35, 36] analysed and explored why DBN and Stacked Auto-encoders are successful and the effect of unsupervised hierarchical pre-training, considering that the main role of unsupervised hierarchical pre-training is to assign lower weights to the optimized initial values. Ranzato et al. proposed a coefficient auto-encoder [37] that gives a sparse internal representation of the forced automatic encoder. This is more conducive to the extraction of abstract features and has achieved good results. Other deep structures include the deep Boltzmann machine [38] proposed by Hinton et al., an embedded encoding deep structure [39], and a special deep structure [40]. Lee et al. combined a convolutional neural network with DBN to give a convolutional deep confidence network [41]. This shifts the variance of extracted features and has achieved very good results. These deep structures also use hierarchical unsupervised pre-training to learn. Ming et al. [42] proposed a relational learning method based on hierarchical association rules in the logarithmic domain and used this to discover the potential user access behaviour of Web usage records. Their method uses the partial and whole relation between composite and atomic time in the log ontology to determine the transaction size, extend the association rules to the time hierarchy, find candidate frequent user rules, and extract the potential field relations after pruning the redundant and invalid rules.

The rise of deep structure learning over a short period of four or five years has seen it successfully applied to many issues, such as identification, regression, dimensionality reduction, texture modelling, motion modelling, information retrieval, robotics, and natural language processing.

At present, the study of deep structure learning is still in its initial stages, but more and more researchers are paying attention to its potential. International research teams include those led by Hinton at the University of Montreal, University of Toronto, LeCun, New York University, Bengio, and other research groups. Deep learning was one of the hot topics at the 28th International Conference on Machine Learning held in Washington DC in 2011.

6.1.1.3 GBRL

GBRL, also known as graph-based multi-relational data mining, has a different style of logical relationship learning from ILP. With frequent sub-graph discovery using graph data mining technology, GBRL represents the pre-preparation of input and output mode languages in the form of graphs; i.e. examples, background knowledge, hypotheses, and the target concepts are expressed in this form. On the basis of frequent patterns, the main goal of GBRL is to find a novel, useful, and understandable pattern.

This kind of algorithm is not well studied; to date, the development of GBRL algorithms is mainly covered by Subdue [43] and graph-based induction (GBI) [44], which is based on a greedy search to find the sub-graph containing the maximum information. Both algorithms extract common sub-graphs from the graph through an incomplete search.

Subdue uses the principle of the minimum description length to assess the degree of interest of the sub-structures. The number of sub-structures generated by this algorithm is determined by the optimal compression of datasets, and the sub-structures provide more useful and important knowledge associated with the field. Subdue can be used for both unsupervised and supervised learning [43].

Subdue uses a cluster search algorithm in which each node is labelled as a sub-structure by adding a side and/or a node to engender a candidate. When all the sub-structures have been considered (or more than some pre-specified number), the DML principle is used to compress the structure.

GBI uses a similar information gain metric to evaluate the sub-graphs with three nodes and then identifies the frequent three-tuples, some of which are compressed by a pre-set ratio [44]. GBI uses the scale of the experience graph to define the size of the compression map and the extraction model. GBI can handle a directed or undirected tag graph with a closed path.

6.1.1.4 DFRL

The basic concept of dynamic fuzzy sets was described by Wang. In 1965, Zadeh proposed the concept of fuzzy sets and later developed fuzzy logic theory. On this

basis, Professor Li has thoroughly researched a broader perspective leading to a series of results on dynamic fuzzy sets and DFL. Landmark publications include “Dynamic fuzzy sets and its application” [46] in March 1997, “Of dynamic fuzzy logic and its application” [47] in December 1997, “Dynamic fuzzy logic theory” [48] in July 2005, and “Dynamic fuzzy logic and its applications” [49] in May 2008. This latter book was the first international book on DFL to be published in English and has been applied in the United States, Britain, France, Japan, and Canada.

In 2002, Professor Li established a dynamic fuzzy relational data model [50], gave the calculation method for DF data, and provided the theoretical basis for processing DFD. In 2007, Zhang researched DF relationships that are difficult to determine in the DF reasoning rules and presented a DFRL algorithm [51] to solve the problem of relationship learning in DF systems. The algorithm can be used to generate or modify the dynamic fuzzy rules of a dynamic fuzzy machine learning system and considers the effect of the first K steps in learning on learning step $n + 1$ using the time complexity to ensure the credibility of the results. In addition, the DFRL algorithm also solves the problem of noise interference in the observation data.

6.1.2 Questions proposed

Most classical learning methods (such as decision trees, naive Bayesian networks, and SVM) are attribute-value learning methods. In such methods, each sample is represented as an attribute-value tuple. The entire dataset can be viewed as a table or relationship in a relational database. To organize and access the data effectively, the relational database is organized in the form of multi-relationships. Attribute-value learning has the limitation that single forms and single relationships make it hard to find a more complex model hidden in the real-world data [1]. Machine learning and data mining technology clearly need to consider the relationship representation of learning tasks and the related search mechanism; that is, this should be considered in the complex structure of the multi-relational dataset learning design. In the field of machine learning, this kind of problem and its solution are called relationship learning. Relationship learning can give better solutions to complex real-world relationship data, but it is difficult to deal with uncertain information. Dynamic fuzzy systems have the ability to deal with the uncertainty of complex data [48, 49].

Most of the current learning algorithms have various data constraints, such as being independent, from the same distribution, and so on. As a result, we must focus on processing the flat data. Real-world data, except structured data, contain many semi-structural or unstructured data. To deal with the deep relationships of structured data, we must solve the problem of processing the multiple relationships between the data. Through the deep structure, we can relate complex data to the related categories to arrange and gradually process the hierarchical structure.

In the theoretical framework of dynamic fuzzy mathematics, classification and understanding are aided by the hierarchical structure of complex data, the discovery of hidden modes among data, and the learning of beneficial relationships between feature values.

In summary, hierarchical relationships are very important in relationship learning, but the current research on this topic is insufficient. From the view of DFL, this chapter studies dynamic fuzzy hierarchical relationship learning.

6.1.3 Chapter structure

The first section analyses the progress of relationship learning research and outlines the existing problems and research focus of this chapter.

The second section gives the basic theoretical knowledge of DFL; the third section describes a dynamic fuzzy hierarchical relationship learning algorithm based on the basic theory of dynamic fuzzy matrixes; and the fourth section discusses the dynamic fuzzy tree hierarchical relationship learning algorithm and uses examples to analyse the process of this mechanism. The fifth section gives a dynamic fuzzy graph HRL algorithm and presents representative examples. The sixth section focuses on applications and analysis, and the seventh section summarizes this chapter.

6.2 Inductive logic programming

For a given first-order language \mathcal{L} , let the set V contain the language variables. We establish an item set T , atoms, and general rules. Text l is atom a (positive text) or its negation $\neg a$ (negative text). The replacement σ is from V to an application of T and extends to an atom. Let Subset represent the basic replacement set. Clause is a finite disjunction of multiple languages, such as $l_1 \vee \dots \vee l_n$, denoted by $\{l_1 \vee \dots \vee l_n\}$. The Horn clause has at most one positive literal, such as $H \vee \neg l_1 \vee \dots \vee \neg l_n$, and can be denoted by $H \leftarrow \{l_1 \wedge \dots \wedge l_n\}$. Among them, H is called a Horn clause or a corollary. The text conjunctive $l_1 \wedge \dots \wedge l_n$ is called the body of the Horn clause or antecedent words. We now proceed with the concept of logical inference.

Definition 6.1 With the knowledge that X is an atom rule, $I, \sigma \models X$ represents $\sigma(X) \in I$. We extend this to a general rule F using semantic combination.

$I \models F$ represents $\forall \sigma, I, \sigma \models F$ (i.e. I is a model of F).

$\models F$ represents $\forall I \in \mathcal{I}, I \models F$.

$F \models G$ denotes that all of models of F are models of G .

Statement: Under the background of the first-order logic, the goal is to find the set of rule H , just like

$$B \bigcup H \models E. \quad (6.1)$$

Given a set of background theory and observation values (training set), let E , B , and H represent the set of clauses. The following set of rules are considered as the conjunctive elements and increase the two constraint conditions: 1 – $B \not\models E$: in this condition, H is must not be used to explain E ; 2 – $B \cup H \not\models \perp$: this illustrates that $B \cup H$ is a theory without contradiction.

In ILP, there are two ways to describe the sample. In the first case, the positive example set is represented as E^+ , and the negative example set is represented as E^- . Another approach only describes the positive cases in E and uses the closed world assumption. In this chapter, we use the second kind of hypothesis; that is, when learning rules, we only learn the rules of the positive cases. Each element of E is called a sample, and we say the counterexample that does not meet the target is the truth. In a relational database, ILP is usually restricted by the Horn clause and no-function rule, which is a basic fact set. In addition, the set E satisfies the previous requirements, but because it does not have the ability to speculate, it is generally not accepted as a solution.

In this chapter, we will use the algorithm FOIL [52], which is a top-down process. From top to bottom, the algorithm proceeds with the most general clause, which is specialized step by step. FOIL's goal is to generate rules that cover all examples. The following steps are used to obtain the rules that contain the conclusion and the target predicate:

1. Make the rule $X \rightarrow Y$ the most general clause, among them $X = T$;
2. Choose a language l , such as a clause $l \wedge X \rightarrow Y$, to maximize the gain function;
3. $X = l \wedge X$;
4. If the confidence rate $cf(X \rightarrow Y) <$ the critical value, then go to step (2);
5. Return $X \rightarrow Y$.

Here, $T = h_0, h_1 \dots h_n = h \in L_h$.

The gain function can be calculated by the following formula [53]:

$$gain(l \wedge X \rightarrow Y, X \rightarrow Y) = n * (\log_2(cf(l \wedge X \rightarrow Y)) - \log_2(cf(X \rightarrow Y))),$$

where n is the number of samples covered by $l \wedge X \rightarrow Y$. For a Horn clause $X \rightarrow Y$, its confidence rate is $cf(X \rightarrow Y) = \frac{P(X \wedge Y)}{P(X)}$. The confidence rate is also called the confidence level and can be calculated through the definition of probability [54]. Under the assumption of the closed world, ILP data should describe an explanation I_{ILP} . Thus, we can give a fact f :

$$I_{ILP} \models f \text{ iff } B \wedge E \models f,$$

where \mathcal{H} is the Herbrand field described by B and E . Record the average probability of P on \mathcal{H} . As a result, when \vec{t} is the vector based on n free variables, the confidence level of the clause $X \rightarrow Y$ is

$$cf(X(\vec{t}) \rightarrow Y(\vec{t}))_{ILP} = \frac{\left| \left\{ \vec{x} \in \mathcal{H}^n \mid I_{ILP} \models \sigma[\vec{t}/\vec{x}] (X(\vec{t}) \wedge Y(\vec{t})) \right\} \right|}{\left| \left\{ \vec{x} \in \mathcal{H}^n \mid I_{ILP} \models \sigma[\vec{t}/\vec{x}] (X(\vec{t})) \right\} \right|}, \quad (6.2)$$

where $| |$ represents the base. In relation to the number of all samples (positive and negative) covered by the rule, the confidence rate can be understood as the proportion of the number of positive cases covered by the rule. The confidence rate is the rate for which the fact is true as inferred from the rules. However, the definition does not consider the number of rules covered. As this is in the context of first-order logic, this quantity does not always cover the total number of samples.

In ILP, the goal is to learn the concepts represented by predicates. E is a collection of all the facts that are relevant to the target predicate. Hence, in the case of non-recursion, the rule learned by us is made up of the predicate in B and the object predicate, where the predicate is the condition part and the object predicate is the conclusion part.

6.3 Dynamic fuzzy HRL

This section is based on DFL and dynamic fuzzy matrix theory. The related algorithms are presented separately to describe the hierarchical relationship between data, how the rules are learnt, and how this is combined with the application of DFD.

6.3.1 DFL relation learning algorithm (DFLR)

1. Summary of the relationship between DFL

Definition 6.2 A mapping is defined on the domain U :

$$(\overleftarrow{A}, \overrightarrow{A}): (\overleftarrow{U}, \overrightarrow{U}) \rightarrow [0, 1] \times [\leftarrow, \rightarrow], (\overleftarrow{u}, \overrightarrow{u}) \mapsto (\overleftarrow{A}(\overleftarrow{u}), \overrightarrow{A}(\overrightarrow{u})).$$

We record this as $(\overleftarrow{A}, \overrightarrow{A}) = \overleftarrow{A}$ or \overrightarrow{A} and call $(\overleftarrow{A}, \overrightarrow{A})$ as the dynamic fuzzy set of $(\overleftarrow{U}, \overrightarrow{U})$. $(\overleftarrow{A}(\overleftarrow{u}), \overrightarrow{A}(\overrightarrow{u}))$ is the membership degree of $(\overleftarrow{A}, \overrightarrow{A})$ according to some the membership degree function [48].

Consider the first-order logic relation database RDB and the dynamic fuzzy logical predicate as the positive fact set. The field $[0, 1] \times [\leftarrow, \rightarrow]$, which is the dynamic fuzzy rate of $[\overleftarrow{0}, \overrightarrow{1}]$, marks these facts. For example, a database contains the fact $(weight(car, (light, light)), (0.9, 0.9))$. The fact states that the car is likely to represent a light truck. Thus, RDB is of the form $(A(\vec{x}, \vec{x}), (\vec{u}(A(\vec{x}, \vec{x})), \vec{u}(A(\vec{x}, \vec{x}))))$, where

$(\overleftarrow{x}, \overrightarrow{x}) \in \mathcal{H}^n$, $A(\overleftarrow{x}, \overrightarrow{x})$ is a fact, and $(\overleftarrow{u}(A(\overleftarrow{x}, \overrightarrow{x})), \overrightarrow{u}(A(\overleftarrow{x}, \overrightarrow{x})))$ is the coincidence rate, which is related with the dynamic fuzzy attribute of $(\overleftarrow{x}, \overrightarrow{x})$.

Dynamic fuzzy predicates make the rules more flexible and expressive. The features function of general predicates is the horizontal tangent function μ_F , which is related to the dynamic fuzzy membership function μ_F , that is, $\mu_{F_\alpha}(\overleftarrow{x}, \overrightarrow{x}) = 1$ iff $\mu(F(\overleftarrow{x}, \overrightarrow{x})) \geq \alpha$ or $\mu_{F_\alpha}(\overleftarrow{x}, \overrightarrow{x}) = 0$. Thus, the rule $A(\overleftarrow{t}, \overrightarrow{t}) \rightarrow C(\overleftarrow{t}, \overrightarrow{t})$ is naturally related to a clear rule $A_\alpha(\overleftarrow{t}, \overrightarrow{t}) \rightarrow C_\beta(\overleftarrow{t}, \overrightarrow{t})$. If $A_\beta(\overleftarrow{t}, \overrightarrow{t})$ is true, then $A_\alpha(\overleftarrow{t}, \overrightarrow{t})$ is true, because $\alpha \leq \beta$. Thus, we need only consider the clear approximation $A_\alpha(\overleftarrow{t}, \overrightarrow{t}) \rightarrow C_\alpha(\overleftarrow{t}, \overrightarrow{t})$.

In this way, if the flexibility is considered, one possible understanding of the dynamic fuzzy rule $A(\overleftarrow{t}, \overrightarrow{t}) \rightarrow C(\overleftarrow{t}, \overrightarrow{t})$ is

$$\forall(\overleftarrow{x}, \overrightarrow{x}), \exists \alpha A_\alpha(\overleftarrow{x}, \overrightarrow{x}) \rightarrow C_\alpha(\overleftarrow{x}, \overrightarrow{x}). \quad (6.3)$$

That is, there exists a clear understanding of the dynamic fuzzy rule of each sample [but there is no need to deal with every sample in the same way, because α does not rely on $(\overleftarrow{x}, \overrightarrow{x})$].

If expressiveness is being considered, we need to find the dynamic fuzzy rule that is true to each horizontal cut copy of it. This means

$$\forall(\overleftarrow{x}, \overrightarrow{x}), \forall \alpha A_\alpha(\overleftarrow{x}, \overrightarrow{x}) \rightarrow C_\alpha(\overleftarrow{x}, \overrightarrow{x}), \quad (6.4)$$

which is stricter than (6.3) because the dynamic fuzzy rule is just a general rule set. Dynamic fuzzy rules are nested predicates that summarize the only dynamic fuzzy rule. In fact, (6.4) is just a gradual rule: the meaning of the expression is “the more $(\overleftarrow{x}, \overrightarrow{x})$ accords with A , the more $(\overleftarrow{x}, \overrightarrow{x})$ accords with C ” [because it simulates $\mu(A(\overleftarrow{x}, \overrightarrow{x})) \geq \mu(C(\overleftarrow{x}, \overrightarrow{x}))$ by a conditional constraint].

This type of gradual rule is one of the four general dynamic fuzzy rules. The gradual rule and the certainty rule are based on implication connection and restrict the possible model of the world. The other two types, possibility rules and non-gradual rules, are more likely to express certain values that are guaranteed to be possible (i.e. they are present in the sample group). For example, the form “the more $(\overleftarrow{x}, \overrightarrow{x})$ accords with A , the greater the possibility that C can be true” (when C is dynamic and fuzzy, the true value becomes a problem of some rate). That is, when C is dynamic and fuzzy, for any explanations that are possible for C , the more samples exist. Note that the rule cannot contain any “classic” counterexamples because we are interested in the distribution of the membership rate in the database.

Next, we consider gradual and certainty rules. Compared with the possibility rule, the certainty rule expresses that “the more $(\overleftarrow{x}, \overrightarrow{x})$ accords with A , the more certain it is that $(\overleftarrow{x}, \overrightarrow{x})$ is C ”. First, consider A as a dynamic fuzzy predicate and C as a general predicate. This is represented as “the more $(\overleftarrow{x}, \overrightarrow{x})$ accords with A ,

which gives a bigger α according to $A(\overleftarrow{x}, \overrightarrow{x}) \geq \alpha$, the fewer the extra situations of the rule $A_\alpha(\overleftarrow{t}, \overrightarrow{t}) \rightarrow C(\overleftarrow{t}, \overrightarrow{t})$. In fact, when α becomes smaller, the number of extra situations can only increase, because the range of A_α becomes larger. When C is also a dynamic fuzzy predicate, to retain the comprehension, we need to search for rules of the form

$$\forall(\overleftarrow{x}, \overrightarrow{x}), \forall\alpha A_\alpha(\overleftarrow{x}, \overrightarrow{x}) \rightarrow C_{1-\alpha}(\overleftarrow{x}, \overrightarrow{x}), \quad (6.5)$$

because when α becomes bigger, $C_{1-\alpha}$ covers more samples.

It is known to be difficult to handle real-valued attributes when attempting to learn an algorithm of the rules, because numerical data can lead to an infinite number of hypotheses. In relationship learning, because the assumption space is large, this problem is more in-depth. When real numbers appear in the concept of learning, the difficulty will increase. Real numbers are generally treated by two methods: the introduction of constraints or the discretization of the real number.

In first-order logic, constraint conditions generally use a number of mathematical operations (inequalities, averages, etc.). The rules induced in this method may lack expression and generality. In addition, in the first-order logic, the algorithm to deal with the constraints is often beyond the scope of the standard solution.

The second method is to use discretization and clustering to convert the continuous information into qualitative information. Then, the information can be processed directly by classical logic and background knowledge. This method is the most widely used because it allows the processing of numerical data and improves the readability of the data. Because clusters are usually defined before induction, the generated rules are dependent on the quality of the cluster. These clusters are usually represented by means of imprecise and variable predicates. For example, among the auto-mpg data in the UCI database, the fuel consumption per mile can be represented by the three predicates “low consumption”, “medium consumption”, and “high consumption”. In this case, the dynamic fuzzy markers represented by the dynamic fuzzy set are more suitable for describing the consumption values, because the dynamic fuzzy markers can avoid any critical point between low and medium.

Finally, the use of dynamic fuzzy predicates allows us to relax the rigidity of the clustering clarity and maintain the readability of the resolution rule. At the same time, the different types of dynamic fuzzy rules can be better described and can also handle a new type of data generalization. The flexible rules are regarded as a dynamic fuzzy fitting of clear rules, in this sense, the high-level cut of some dynamic fuzzy expression corresponding to a dynamic fuzzy predicate. There is an understanding of the dynamic fuzzy predicate with respect to the data, which can produce meaningful rules. Gradual rules and certainty rules are the new rules that describe the new properties of the data. In ILP, the goal is to find a hypothesis that is reasonable and complete with respect to the sample. The background knowledge and sample define the interpretation, in

which there is a false target concept. If the assumption does not cover the fact of the concept of this false target, then the assumption is reasonable; there is a true target concept in the explanation, and, if it really covers all the facts in the true target, then the hypothesis is complete. In the dynamic fuzzy ILP, the definition of the sample covered by the rule depends on the dynamic fuzzy rules and the membership rate type of the verification rule fact.

2. DFL relation learning algorithm

In the FOIL algorithm, the measurement criteria of the procedures are the confidence, suspension conditions, and the number of samples covered by the rules. If a sample is covered by a regular copy of the rule, the sample is covered by the fuzzy rule. As a result, the following measures are described for each of these rules.

1) Flexible rule

For dynamic fuzzy rule $A(\overleftarrow{t}, \overrightarrow{t}) \rightarrow C(\overleftarrow{t}, \overrightarrow{t})$, the first meaning is close to the classical rule. The coincidence rate of $A(\overleftarrow{x}, \overrightarrow{x})$ and $C(\overleftarrow{x}, \overrightarrow{x})$ should be as high as possible. Thus, the classic explanations of each horizontal cut α are induced.

Definition 6.3 For a known fact f , the α -explanatory type I_α is defined as follows:

$$I_\alpha |= f \text{ iff } B \wedge E |= f \text{ and } \mu(f) \geq \alpha.$$

In this kind interpretation, only the fact that the coincidence rate is higher than α is true. Now, calculate the coincidence rate of each explanatory type of α in the classic way [55]. According to the original intention of the dynamic fuzzy rules, we must agree with the confidence level of the high α -explanatory-type calculation rules. In fact, it is more likely to be covered by a high degree of coincidence of the sample. A first-order logic is proposed in [55], and the next definition is the adjustment of the first-order logic:

$$cf_{flex}(A(\overleftarrow{t}, \overrightarrow{t}) \rightarrow C(\overleftarrow{t}, \overrightarrow{t})) = \sum_{\alpha_i} (\alpha_i - \alpha_{i+1}) * cf(A(\overleftarrow{t}, \overrightarrow{t}) \rightarrow C(\overleftarrow{t}, \overrightarrow{t}))_{I_{\alpha_i}},$$

where $\alpha_1 = 1, \dots, \alpha_t = 0$ is the decreasing sequence of coincidence rates that appears in the database. The coincidence is the discretization of confidence rates in the α -explanatory type. We induce the formula for calculating the number of different covered samples:

$$\begin{aligned} n_{flex}(A(\overleftarrow{t}, \overrightarrow{t}) \rightarrow C(\overleftarrow{t}, \overrightarrow{t})) &= \sum_{\alpha_i} (\alpha_i - \alpha_{i+1}) * |\{(\overleftarrow{x}_1, \overrightarrow{x}_1) \in \mathcal{H}^q, \exists (\overleftarrow{x}_2, \overrightarrow{x}_2) \in \mathcal{H}^r | I_{\alpha_i} \}| \\ &= \sigma \left[(\overleftarrow{t}_1, \overrightarrow{t}_1), (\overleftarrow{t}_2, \overrightarrow{t}_2) / (\overleftarrow{x}_1, \overrightarrow{x}_1), (\overleftarrow{x}_2, \overrightarrow{x}_2) \right] (A \wedge C). \end{aligned}$$

2) Gradual rule

In this case, the value of the degree of coincidence is only useful for the comparison conditions and the conclusion. As a result, it is no longer the same as the previous treatment of the coincidence rate of high α -explanatory type.

$$cf_{grad}(A(\overleftarrow{t}, \overrightarrow{t}) \rightarrow C(\overleftarrow{t}, \overrightarrow{t})) = \frac{\left| \{(\overleftarrow{x}, \overrightarrow{x}) \in \mathcal{H}^n | I_{ILP} = \sigma[(\overleftarrow{t}, \overrightarrow{t}) / (\overleftarrow{x}, \overrightarrow{x})] (A \wedge C), \mu(\sigma[(\overleftarrow{t}, \overrightarrow{t}) / (\overleftarrow{x}, \overrightarrow{x})] C) \geq \mu(\sigma[(\overleftarrow{t}, \overrightarrow{t}) / (\overleftarrow{x}, \overrightarrow{x})] A) \} \right|}{\left| \{(\overleftarrow{x}, \overrightarrow{x}) \in \mathcal{H}^n | I_{ILP} = \sigma[(\overleftarrow{t}, \overrightarrow{t}) / (\overleftarrow{x}, \overrightarrow{x})] (A) \} \right|}.$$

When the part values of the rules are a conjunction of the basic text, the coincidence rate of the conjunction is the minimum rate of each word. The formula for calculating the number of different covered samples is

$$n_{grad}(A(\overleftarrow{t}, \overrightarrow{t}) \rightarrow C(\overleftarrow{t}, \overrightarrow{t})) = \left| \{(\overleftarrow{x}_1, \overrightarrow{x}_1) \in \mathcal{H}^q, \exists (\overleftarrow{x}_2, \overrightarrow{x}_2) \in \mathcal{H}^r | I_{ILP} = \sigma[(\overleftarrow{t}_1, \overrightarrow{t}_1), (\overleftarrow{t}_2, \overrightarrow{t}_2) / (\overleftarrow{x}_1, \overrightarrow{x}_1), (\overleftarrow{x}_2, \overrightarrow{x}_2)] (A \wedge C, \mu(\sigma[(\overleftarrow{t}_1, \overrightarrow{t}_1) / (\overleftarrow{x}_1, \overrightarrow{x}_1)] C) \geq \mu(\sigma[(\overleftarrow{t}_1, \overrightarrow{t}_1), (\overleftarrow{t}_2, \overrightarrow{t}_2) / (\overleftarrow{x}_1, \overrightarrow{x}_1), (\overleftarrow{x}_2, \overrightarrow{x}_2)] A)) \} \right|.$$

3) Type-one certainty rule

The dynamic fuzzy rule $A(\overleftarrow{t}, \overrightarrow{t}) \rightarrow C(\overleftarrow{t}, \overrightarrow{t})$ means “the more $(\overleftarrow{x}, \overrightarrow{x})$ coincides with A , the more $(\overleftarrow{x}, \overrightarrow{x})$ ensures C ”. For these rules, the coincidence rate of the conclusion does not interest us. This is called a type-one certainty rule in the α -horizontal slice corresponding to the following classic explanation type:

Definition 6.4 For a known fact f , the α -certainty type explains the following:

$$I_{\alpha-cert} |= f \text{ iff } (B |= f \text{ and } \mu(f) \geq \alpha) \text{ or } E |= f.$$

Under the restriction of this rule and the explanation of high α -certainty, the respective coincidence rates are high. Thus, it will be more tolerant to exceptional cases. The rule $A(\overleftarrow{t}, \overrightarrow{t}) \rightarrow C(\overleftarrow{t}, \overrightarrow{t})$ coincides with small values of α , and exceptional cases appear in the classic copies of the rule. Thus, we induce the use of Choquet integration:

$$cf_{cert1}(A(\overrightarrow{t}) \rightarrow C(\overrightarrow{t})) = \sum_{\alpha_i}^t (\alpha_i - \alpha_{i+1}) * cf(A(\overrightarrow{t}) \rightarrow C(\overrightarrow{t}))_{I_{\alpha_i-cert}}.$$

The formula for calculating the number of different covered samples is

$$\begin{aligned} n_{cert1}(A(\overleftarrow{t}, \overrightarrow{t}) \rightarrow C(\overleftarrow{t}, \overrightarrow{t})) &= \sum_{\alpha_i} (\alpha_i - \alpha_{i+1}) * |\{(x_1, \vec{x}_1) \in \mathcal{H}^q, \exists(x_2, \vec{x}_2) \in \mathcal{H}^r | I_{\alpha cert}\} \\ &= \sigma[(\overleftarrow{t}_1, \overrightarrow{t}_1), (\overleftarrow{t}_2, \overrightarrow{t}_2) / (\overleftarrow{x}_1, \vec{x}_1), (\overleftarrow{x}_2, \vec{x}_2)](A \wedge C)\}|. \end{aligned}$$

4) Type-two certainty rule

In accordance with the coincidence rate of the results of the processing rules, the above definition is modified by the following methods. This is called a type-two certainty rule and is defined as shown below:

$$\begin{aligned} cf_{cert2}(A(\overleftarrow{t}, \overrightarrow{t}) \rightarrow C(\overleftarrow{t}, \overrightarrow{t})) &= \\ |\{(\overleftarrow{x}, \vec{x}) \in \mathcal{H}^n | I_{ILP}\}| &= \sigma[(\overleftarrow{t}, \overrightarrow{t}) / (\overleftarrow{x}, \vec{x})](A \wedge C), \mu(\sigma[(\overleftarrow{t}, \overrightarrow{t}) / (\overleftarrow{x}, \vec{x})]C) \\ &> 1 - \mu(\sigma[(\overleftarrow{t}, \overrightarrow{t}) / (\overleftarrow{x}, \vec{x})]A)\}| \\ \hline |\{(\overleftarrow{x}, \vec{x}) \in \mathcal{H}^n | I_{ILP}\}| &= \sigma[(\overleftarrow{t}, \overrightarrow{t}) / (\overleftarrow{x}, \vec{x})](A)\}| \end{aligned}.$$

The formula for calculating the number of different covered samples is

$$\begin{aligned} n_{cert2}(A(\overleftarrow{t}, \overrightarrow{t}) \rightarrow C(\overleftarrow{t}, \overrightarrow{t})) &= |\{(x_1, \vec{x}_1) \in \mathcal{H}^q, \exists(x_2, \vec{x}_2) \in \mathcal{H}^r|\} \\ I_{ILP} &= \sigma[(\overleftarrow{t}_1, \overrightarrow{t}_1), (\overleftarrow{t}_2, \overrightarrow{t}_2) / (\overleftarrow{x}_1, \vec{x}_1), (\overleftarrow{x}_2, \vec{x}_2)](A \wedge C), \mu(\sigma(\overleftarrow{t}_1, \overrightarrow{t}_1) / (\overleftarrow{x}_1, \vec{x}_1)C) \\ &> 1 - \mu(\sigma[(\overleftarrow{t}_1, \overrightarrow{t}_1), (\overleftarrow{t}_2, \overrightarrow{t}_2) / (\overleftarrow{x}_1, \vec{x}_1), (\overleftarrow{x}_2, \vec{x}_2)]A)\}|. \end{aligned}$$

Thus, we can use the FOIL algorithm to induce all kinds of first-order dynamic fuzzy rules by adjusting the rule to be learnt.

6.3.2 Sample analysis

1. Sample description

Consider a database that describes a town of 20 houses. First, there are some dynamic fuzzy relational predicates, such as $(close(x, y), (\overleftarrow{\alpha}, \overrightarrow{\alpha}))$ to describe that house x is close to house y with membership rate $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$ (0 represents strangers, 1 represents friends). Dynamic fuzzy predicates, some of which are similar to propositions such as $price(x, (\overleftarrow{expensive}, \overrightarrow{expensive}))$ or $size(x, (\overleftarrow{small}, \overrightarrow{small}))$, can be used to describe the houses.

Thus, in this case, we can attain good confidence values for each type of dynamic fuzzy rule. For instance, consider the flexible rule

$$\text{close}(x, y), \text{price}(y, (\overleftarrow{\text{expensive}}, \overrightarrow{\text{expensive}})) \rightarrow \text{price}(x, (\overleftarrow{\text{expensive}}, \overrightarrow{\text{expensive}})).$$

This has a confidence value of 0.81, so it is reasonable to predict that if a house is close to the price of an expensive house, the price of this house is also expensive (because expensive houses are usually located in the same area). A typical gradual rule is

$$\text{size}(x, (\overleftarrow{\text{large}}, \overrightarrow{\text{large}})) \rightarrow \text{price}(x, (\overleftarrow{\text{expensive}}, \overrightarrow{\text{expensive}})).$$

That is, the bigger the house, the more expensive it is, which explains the fact that price increases with size. The confidence of this rule is 0.80. A good example of certainty is

$$\text{close}(x, y) \rightarrow \text{know}(x, y).$$

If this is type-one certainty rule, the confidence level is 0.95. If it is a type-two certainty rule, the confidence level is 0.88. This rule means “the closer the two houses, the more likely it is that the owners know each other”. Obviously, the owners of houses close to each other have a greater probability of knowing one another. As the distance between the houses increases, the possibility of the owners knowing each other will drop. At the end of the study, we can observe that all of these rules are obviously affected by exceptions, and regardless of what they concern, they cannot be obtained by the classical ILP machine.

2. Results analysis and comparison

The data used in this chapter are taken from the auto-mpg database in the UCI repository (<http://www.ics.uci.edu/~mlearn/MLRepository.html>). The database consists of a car and a concept to learn. The concept to learn is the fuel consumption per mile of city driving. There are 398 car samples in the database. They are described by nine attributes, five of which are continuous, including the concept of miles per gallon (mpg). We conducted an experiment using three multi-valued discrete attributes and five continuous attributes to predict the future mpg properties, that is, the city cycle fuel consumption/mile. The database cannot be represented by propositional logic, but it is sufficient to illustrate the interesting areas of the method. First, the database can be “discretized” into dynamic fuzzy sets. Moreover, we can construct three distinct discrete correspondences:

1. A clear part of the property domain is most closely connected to the dynamic fuzzy part;
2. The support of the dynamic fuzzy sets; and
3. The core of the dynamic fuzzy sets.

Second, we learn the city cycle fuel consumption category for all types of dynamic fuzzy rules, according to clear and dynamic fuzzy sets. Table 6.1 presents the results of predicting the mpg using different rules.

The following describes an example for each type of rule combined into an algorithm. (For the clear part of the domain of the property that corresponds to the nearest dynamic fuzzy part, the classical rule is obtained by discretization.) According to different rules, the algorithm automatically generates the following example:

Classic rule: $cylinders(A, 8) \rightarrow mpg(A, low)$.

The classic rule predicted the attribute value of mpg to be lower than the low field when the automobile cylinder number is less than eight. The data are coincident with the known data in Fig. 6.1.

Tab. 6.1: Prediction result of different rules.

Rule type	Number	Cover	Average confidence
Classic rule	8	0.76	0.85
Classic rule with dynamic fuzzy set core	11	0.61	0.87
Classic rule supported by dynamic fuzzy set	10	0.87	0.81
Flexible rule	3	0.52	0.85
Gradual rule	4	0.48	0.91
Type one certainty rule	2	0.61	0.74
Type two certainty rule	2	0.58	0.77

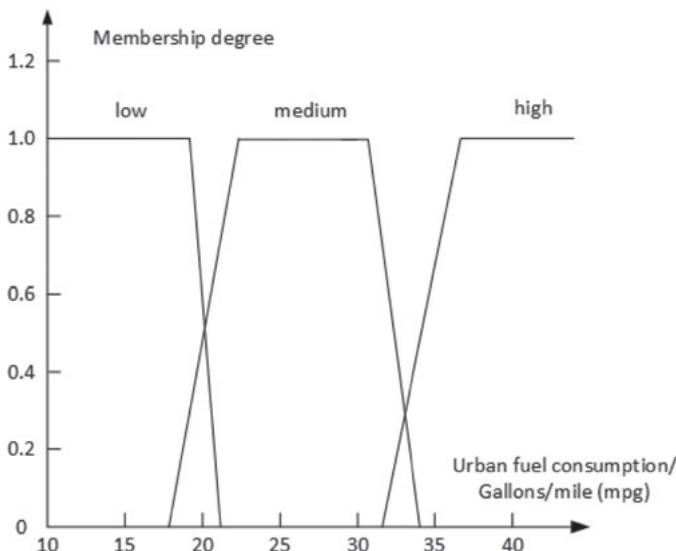


Fig. 6.1: Relationship between membership rate and mpg.

Flexible rule:

$$\text{displacement}(A, \text{low}), \text{weight}(A, \text{medium}) \rightarrow \text{mpg}(A, \text{medium}).$$

The rule explains that we can induce that mpg is in the low field when the number of cylinders is six, the weight is high, $\text{origin} = 1$, automobile acceleration is low, and horsepower is low.

Gradual rule: $\text{weight}(A, \text{high}) \rightarrow \text{mpg}(A, \text{low})$.

The rule explains that we can induce that attribute mpg is in the low field when weight is in the high field.

Type one certainty rule: $\text{cylinders}(A, 6), \text{weight}(A, \text{high}) \rightarrow \text{mpg}(A, \text{low})$.

The rule explains that we can induce that mpg is in the low field when the number of cylinders of automobile A is six and its weight is in the high field.

Type two certainty rule:

$$\begin{aligned} \text{cylinders}(A, 6), \text{weight}(A, \text{high}), \text{origin}(A, 1), \text{acceleration}(A, \text{low}), \\ \text{horsepower}(A, \text{low}) \rightarrow \text{mpg}(A, \text{low}). \end{aligned}$$

The rule explains that we can induce that mpg is in the low field when the number of cylinders of automobile is six, its weight is in the high field, $\text{origin} = 1$, automobile acceleration is low, and horsepower is low.

As expected, the coverage rate of the classical rules is between that of the classical rules with a dynamic fuzzy set kernel and the classical rules supported by dynamic fuzzy sets. If this is due to the dynamic fuzzy rules kernel, samples on the clear category boundaries will not be processed. In contrast, with the support of dynamic fuzzy sets, it is clear that samples on the class boundary will belong to two categories. Relative to the coverage, because the dynamic fuzzy rules are more difficult to find than the classical rules, the kernel of the dynamic fuzzy rules is smaller. This result is to be expected because the estimated value of each predicate must be considered, and the dynamic fuzzy rules will be subject to more conditions. In fact, the confidence level of the classical rule does not depend on the distance between the data on the discrete set boundary. For example, consider the classic rule F with good confidence, as well as its dynamic fuzzy flexible copy F' . Relative to the dynamic fuzzy sets, if many examples of F belong to the boundary line, the confidence of F' will be lower than the confidence of F . On the contrary, if many counterexamples of F lie in the border region, the confidence of F' will be higher than that of F . Thus, compared to the small change in the set boundary, the confidence of the dynamic fuzzy flexible copy is a good indicator of the robustness of the classical rule.

Type-one certainty rules concern the degree of membership of the rule condition. Gradual and certainty rules show how the conditions and conclusions evolve together.

The meanings of these rules are far different from those of the classical rules and need not have a clear copy or approximation. Dealing with certainty in dynamic fuzzy rules, it tends to be the non-dynamic fuzzy predicates that favour the conditional part. This is because the non-dynamic fuzzy predicate allows more freedom about the conformity of the sample to be covered. Note that there are some rules that can be described by propositional logic, but the examples are automatically generated by the algorithm. As some of the rules have shown, this algorithm can mix dynamic fuzzy predicates and non-dynamic fuzzy predicates.

6.3.3 Dynamic fuzzy matrix HRL algorithm

1. Dynamic fuzzy matrix

Definition 6.5 If $(\overleftarrow{R}, \overrightarrow{R}) \in DF_L((\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y}))$, we say that $(\overleftarrow{R}, \overrightarrow{R})$ is the type L relationship from $(\overleftarrow{X}, \overrightarrow{X})$ to $(\overleftarrow{Y}, \overrightarrow{Y})$. If

$L = [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]$, $(\overleftarrow{R}, \overrightarrow{R}) \in DF_L((\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y}))$, $(\overleftarrow{R}, \overrightarrow{R})$ is the DF relationship from $(\overleftarrow{X}, \overrightarrow{X})$ to $(\overleftarrow{Y}, \overrightarrow{Y})$.

Definition 6.6 If $(\overleftarrow{R}_1, \overrightarrow{R}_1), (\overleftarrow{R}_2, \overrightarrow{R}_2) \in DF_L((\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y}))$, write

- (1) $(\overleftarrow{R}_1, \overrightarrow{R}_1) \subset (\overleftarrow{R}_2, \overrightarrow{R}_2) \Leftrightarrow (\overleftarrow{R}_1, \overrightarrow{R}_1)((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) \leq (\overleftarrow{R}_2, \overrightarrow{R}_2)((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y}))$,
where $((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) \in ((\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y}))$
- (2) $(\overleftarrow{R}_1, \overrightarrow{R}_1) \cup (\overleftarrow{R}_2, \overrightarrow{R}_2)((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) = (\overleftarrow{R}_1, \overrightarrow{R}_1)((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) \vee (\overleftarrow{R}_2, \overrightarrow{R}_2)((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y}))$
 $(\overleftarrow{R}_1, \overrightarrow{R}_1) \cap (\overleftarrow{R}_2, \overrightarrow{R}_2)((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) = (\overleftarrow{R}_1, \overrightarrow{R}_1)((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) \wedge (\overleftarrow{R}_2, \overrightarrow{R}_2)((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y}))$
- (3) $(\overleftarrow{R}, \overrightarrow{R})^{-1}((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) = (\overleftarrow{R}, \overrightarrow{R})((\overleftarrow{y}, \overrightarrow{y}), (\overleftarrow{x}, \overrightarrow{x}))$

Assume $(\overleftarrow{X}, \overrightarrow{X}) = \{(\overleftarrow{x}_1, \overrightarrow{x}_1), \dots, (\overleftarrow{x}_n, \overrightarrow{x}_n)\}$, $(\overleftarrow{Y}, \overrightarrow{Y}) = \{(\overleftarrow{y}_1, \overrightarrow{y}_1), \dots, (\overleftarrow{y}_m, \overrightarrow{y}_m)\}$, $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ is a DF grid, and that 0 and 1 respectively represent the minimum and maximum elements of $(\overleftarrow{L}, \overrightarrow{L})$.

Definition 6.7 If $(\overleftarrow{R}, \overrightarrow{R}) : (\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y}) \rightarrow (\overleftarrow{L}, \overrightarrow{L})$, we say that $(\overleftarrow{R}, \overrightarrow{R})$ is an $(\overleftarrow{L}, \overrightarrow{L})$ -type DF matrix, written as

$$(\overleftarrow{R}, \overrightarrow{R}) = \begin{pmatrix} (\overleftarrow{r}_{11}, \overrightarrow{r}_{11}) & (\overleftarrow{r}_{12}, \overrightarrow{r}_{12}) & \dots & (\overleftarrow{r}_{1m}, \overrightarrow{r}_{1m}) \\ (\overleftarrow{r}_{21}, \overrightarrow{r}_{21}) & (\overleftarrow{r}_{22}, \overrightarrow{r}_{22}) & \vdots & (\overleftarrow{r}_{2m}, \overrightarrow{r}_{2m}) \\ \vdots & \vdots & \vdots & \vdots \\ (\overleftarrow{r}_{n1}, \overrightarrow{r}_{n1}) & (\overleftarrow{r}_{n2}, \overrightarrow{r}_{n2}) & \dots & (\overleftarrow{r}_{nm}, \overrightarrow{r}_{nm}) \end{pmatrix},$$

where $(\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}) = (\overleftarrow{R}, \overrightarrow{R})((\overleftarrow{x}_i, \overrightarrow{x}_i), (\overleftarrow{y}_j, \overrightarrow{y}_j))$ ($i \leq n, j \leq m$).

In particular, $(\overleftarrow{R}, \overrightarrow{R}) : (\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y}) \rightarrow [0, 1] \times [\leftarrow, \rightarrow]$ is a DF matrix.

Obviously, an L-type DF matrix is an L-type DF relationship, and a DF matrix is a DF relationship.

Write $DF_L((\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y})) = \{(\overleftarrow{R}, \overrightarrow{R}); (\overleftarrow{R}, \overrightarrow{R}) : (\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y}) \rightarrow (\overleftarrow{L}, \overrightarrow{L})\}$. If $(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{S}, \overrightarrow{S}) \in DF_L((\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y}))$, $(\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})$, $(\overleftarrow{S}, \overrightarrow{S}) = (\overleftarrow{s}_{ij}, \overrightarrow{s}_{ij})$. If

$\forall i \leq n, j \leq m, (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}) \leq (\overleftarrow{s}_{ij}, \overrightarrow{s}_{ij})$, $(\overleftarrow{R}, \overrightarrow{R})$ is contained in $(\overleftarrow{S}, \overrightarrow{S})$, which we write as $(\overleftarrow{R}, \overrightarrow{R}) \subset (\overleftarrow{S}, \overrightarrow{S})$.

Obviously, $DF_L((\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y}))$, \subset is a poset.

Assume (L, S, T, N) is a module system, and write

$$(\overleftarrow{R}, \overrightarrow{R}) \bigcup (\overleftarrow{S}, \overrightarrow{S}) = ((\overleftarrow{S}, \overrightarrow{S})((\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}), (\overleftarrow{s}_{ij}, \overrightarrow{s}_{ij})); i \leq n, j \leq m)$$

$$(\overleftarrow{R}, \overrightarrow{R}) \bigcap (\overleftarrow{S}, \overrightarrow{S}) = ((\overleftarrow{T}, \overrightarrow{T})((\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}), (\overleftarrow{s}_{ij}, \overrightarrow{s}_{ij})); i \leq n, j \leq m)$$

$$(\overleftarrow{R}, \overrightarrow{R})^c = ((\overleftarrow{N}, \overrightarrow{N})((\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})); i \leq n, j \leq m).$$

Then, $DF_L((\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y}))$, \bigcap , \bigcup , c is also a module system, $(\overleftarrow{O}, \overrightarrow{O}) = ((\overleftarrow{o}_{ij}, \overrightarrow{o}_{ij}) = (\overleftarrow{0}, \overrightarrow{0}); i \leq n, j \leq m)$ and $(\overleftarrow{E}, \overrightarrow{E}) = ((\overleftarrow{e}_{ij}, \overrightarrow{e}_{ij}) = (\overleftarrow{1}, \overrightarrow{1}); i \leq n, j \leq m)$ are the minimum element and maximum element of $DF_L((\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y}))$.

Definition 6.8 Assume $(\overleftarrow{R}, \overrightarrow{R})$ is an L-type DF matrix and $(\overleftarrow{R}, \overrightarrow{R}) = ((\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}); i \leq n, j \leq m)$. If there exists an L-type DF matrix $(\overleftarrow{Q}, \overrightarrow{Q}) = ((\overleftarrow{q}_{ij}, \overrightarrow{q}_{ij}); j \leq m, i \leq n)$ such that $(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R})$, we call $(\overleftarrow{R}, \overrightarrow{R})$ regular and say that $(\overleftarrow{Q}, \overrightarrow{Q})$ is the general type-T matrix of $(\overleftarrow{R}, \overrightarrow{R})$.

If there is some $(\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) = (\overleftarrow{Q}, \overrightarrow{Q})$, then $(\overleftarrow{Q}, \overrightarrow{Q})$ is the T inverse matrix of $(\overleftarrow{R}, \overrightarrow{R})$.

Theorem 6.1 Assume $(\overleftarrow{Q}, \overrightarrow{Q})$ is the general T inverse matrix of $(\overleftarrow{R}, \overrightarrow{R})$. Then, $(\overleftarrow{S}, \overrightarrow{S}) = (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q})$ is the T inverse matrix of $(\overleftarrow{R}, \overrightarrow{R})$, where T is a distribution grid.

Proof: According to Definition 6.8,

$$\begin{aligned} (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{S}, \overrightarrow{S}) \circ (\overleftarrow{R}, \overrightarrow{R}) &= (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \\ &= (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{R}, \overrightarrow{R}). \end{aligned}$$

Thus,

$$\begin{aligned} (\overleftarrow{S}, \overrightarrow{S}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{S}, \overrightarrow{S}) &= (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \\ &= (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) = (\overleftarrow{S}, \overrightarrow{S}), \end{aligned}$$

which completes the proof.

Theorem 6.2 Assume $(\overleftarrow{R}, \overrightarrow{R})$ is a DF matrix, and the sufficient and necessary condition for $(\overleftarrow{R}, \overrightarrow{R})$ to be regular is

$$(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{X}, \overrightarrow{X}) \circ (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R}).$$

Now, $(\overleftarrow{X}, \overrightarrow{X})$ is the biggest general DF inverse matrix. When $(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R})$, then $(\overleftarrow{Q}, \overrightarrow{Q}) \subset (\overleftarrow{X}, \overrightarrow{X})$.

Proof: The sufficient condition is obviously true, so we need to prove the necessary condition.

Assume $(\overleftarrow{R}, \overrightarrow{R})$ is regular and that there exists some $(\overleftarrow{Q}, \overrightarrow{Q})$ satisfying $(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R})$. Thus, there is

$$\bigvee_{k=1}^n \bigvee_{j=1}^m ((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{q}_{ik}, \overrightarrow{q}_{ik})) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj}) = (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}) \quad (i \leq n, j \leq m),$$

that is,

$$\bigvee_{k=1}^n \bigvee_{e=1}^m ((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{q}_{ek}, \overrightarrow{q}_{ek})) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj}) = (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}) \quad (i \leq n, j \leq m).$$

Thus, for $\forall k \leq n, e \leq m$, there is $((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{q}_{ik}, \overrightarrow{q}_{ik})) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj}) \leq (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})$ ($i \leq n, j \leq m$).

We can conclude the following:

1. $((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj})) \leq (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}) \Rightarrow 0 \leq (\overleftarrow{q}_{ek}, \overrightarrow{q}_{ek}) \leq 1$;
2. $((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj})) > (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}) \Rightarrow 0 \leq (\overleftarrow{q}_{ek}, \overrightarrow{q}_{ek}) \leq (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})$.

Hence, $0 \leq (\overleftarrow{q}_{ek}, \overrightarrow{q}_{ek}) \leq \wedge \{(\overleftarrow{r}_{st}, \overrightarrow{r}_{st}), (\overleftarrow{r}_{st}, \overrightarrow{r}_{st}) < ((\overleftarrow{r}_{st}, \overrightarrow{r}_{st}) \wedge (\overleftarrow{r}_{nt}, \overrightarrow{r}_{nt}))\} = (\overleftarrow{x}_{ek}, \overrightarrow{x}_{ek})$; that is, $(\overleftarrow{Q}, \overrightarrow{Q}) \subset (\overleftarrow{X}, \overrightarrow{X})$ and $(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \subset (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{X}, \overrightarrow{X}) \circ (\overleftarrow{R}, \overrightarrow{R})$.

Assuming that $(\overleftarrow{U}, \overrightarrow{U}) = (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{X}, \overrightarrow{X}) \circ (\overleftarrow{R}, \overrightarrow{R})$ and

$$u_{ij} = \bigvee_{k=1}^n \bigvee_{e=1}^m ((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{x}_{ek}, \overrightarrow{x}_{ek}) \wedge (\overleftarrow{r}_{ke}, \overrightarrow{r}_{ke})) \quad (i \leq n, j \leq m),$$

Thus, we can prove that $u_{ij} \leq r_{ij}$ ($i \leq n, j \leq m$) is true.

In fact, when $((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{r}_{ej}, \overrightarrow{r}_{ej})) \leq (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})$ is true, we have

$$((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{x}_{ik}, \overrightarrow{x}_{ik}) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj})) \leq ((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj})) \leq (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}),$$

and when $((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{r}_{ej}, \overrightarrow{r}_{ej})) > (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})$ is true, there is

$$((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{x}_{ek}, \overrightarrow{x}_{ek}) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj})) \leq (\overleftarrow{x}_{ek}, \overrightarrow{x}_{ek}) \leq (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}).$$

Hence, $(\overleftarrow{u}_{ij}, \overrightarrow{u}_{ij}) \leq (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})$ ($i \leq n, j \leq m$); that is,

$$(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{X}, \overrightarrow{X}) \circ (\overleftarrow{R}, \overrightarrow{R}) \subset (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}).$$

Thus, $(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{X}, \overrightarrow{X}) \circ (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R})$, which completes the proof.

2. Dynamic fuzzy matrix HRL algorithm

Learning problems can be decomposed in accordance with their level so as to construct a bottom-up hierarchy. This is more conducive to qualitative and quantitative analysis. The traditional absolute value is used to show the uncertainty that does not accord with the subjective judgment and the characteristics of the object, so dynamic fuzzy theory is used to analyse the learning problem. We use the dynamic fuzzy matrix $(\overleftarrow{R}, \overrightarrow{R})$ to represent the relationship rate between upper data and related data in this level. Assuming that upper element C and this level's elements c_1, c_2, \dots, c_n are related, then the dynamic fuzzy matrix $(\overleftarrow{R}, \overrightarrow{R})$ has the following form:

$$\begin{pmatrix} C & c_1 & c_2 & \dots & c_n \\ c_1 & (\overleftarrow{r}_{11}, \overrightarrow{r}_{11}) & (\overleftarrow{r}_{12}, \overrightarrow{r}_{12}) & \dots & (\overleftarrow{r}_{1n}, \overrightarrow{r}_{1n}) \\ c_2 & (\overleftarrow{r}_{21}, \overrightarrow{r}_{21}) & (\overleftarrow{r}_{22}, \overrightarrow{r}_{22}) & \dots & (\overleftarrow{r}_{2n}, \overrightarrow{r}_{2n}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ c_n & (\overleftarrow{r}_{n1}, \overrightarrow{r}_{n1}) & (\overleftarrow{r}_{n2}, \overrightarrow{r}_{n2}) & \dots & (\overleftarrow{r}_{nn}, \overrightarrow{r}_{nn}) \end{pmatrix},$$

where $(\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}) \in [0, 1] \times [\leftarrow, \rightarrow]$ ($i, j \leq n$) represents element c_i and c_j according to the relationship rate of element C and elements c_i and c_j contain the membership rates of the dynamic fuzzy relationship. According to the relationship between elements c_1, c_2, \dots, c_n and upper element C , we construct the dynamic fuzzy matrix of each level as

$$(\overleftarrow{R}, \overrightarrow{R}) = \begin{pmatrix} (\overleftarrow{r}_{11}, \overrightarrow{r}_{11}) & (\overleftarrow{r}_{12}, \overrightarrow{r}_{12}) & \dots & (\overleftarrow{r}_{1n}, \overrightarrow{r}_{1n}) \\ (\overleftarrow{r}_{21}, \overrightarrow{r}_{21}) & (\overleftarrow{r}_{22}, \overrightarrow{r}_{22}) & \vdots & (\overleftarrow{r}_{2n}, \overrightarrow{r}_{2n}) \\ \vdots & \vdots & \vdots & \vdots \\ (\overleftarrow{r}_{n1}, \overrightarrow{r}_{n1}) & (\overleftarrow{r}_{n2}, \overrightarrow{r}_{n2}) & \dots & (\overleftarrow{r}_{nn}, \overrightarrow{r}_{nn}) \end{pmatrix}.$$

Definition 6.9 For a dynamic fuzzy number M in dynamic fuzzy matrix $(\overleftarrow{R}, \overrightarrow{R})$, if and only if its membership rate $(\overleftarrow{\mu}_M, \overrightarrow{\mu}_M)_{(\overleftarrow{x}, \overrightarrow{x})} \in [0, 1] \times [\leftarrow, \rightarrow]$ satisfies

$$(\overleftarrow{\mu}_M, \overrightarrow{\mu}_M)_{(\overleftarrow{x}, \overrightarrow{x})} = \begin{cases} \frac{(\overleftarrow{l} - \overleftarrow{x}, \overrightarrow{l} - \overrightarrow{x})}{(\overleftarrow{l} - \overleftarrow{m}, \overrightarrow{l} - \overrightarrow{m})}, & (\overleftarrow{x}, \overrightarrow{x}) \in [l, m] \times [\leftarrow, \rightarrow] \\ \frac{(\overleftarrow{x} - \overleftarrow{n}, \overrightarrow{x} - \overrightarrow{n})}{(\overleftarrow{m} - \overleftarrow{n}, \overrightarrow{m} - \overrightarrow{n})}, & (\overleftarrow{x}, \overrightarrow{x}) \in [m, n] \times [\leftarrow, \rightarrow] \\ 0, & \text{otherwise} \end{cases}, \quad (6.6)$$

then M is a triangular dynamic fuzzy number, written as $M = ((\overleftarrow{l}, \overrightarrow{l}), (\overleftarrow{m}, \overrightarrow{m}), (\overleftarrow{n}, \overrightarrow{n}))$. Here, $(\overleftarrow{l}, \overrightarrow{l}) \leq (\overleftarrow{m}, \overrightarrow{m}) \leq (\overleftarrow{n}, \overrightarrow{n})$, and all M belong to $[0, 1] \times [\leftarrow, \rightarrow]$, where $(\overleftarrow{l}, \overrightarrow{l})$ and $(\overleftarrow{n}, \overrightarrow{n})$ denote the lower and upper bounds of M , respectively.

Consider two groups of triangular dynamic fuzzy number $M_1 = ((\overleftarrow{l}_1, \overrightarrow{l}_1), (\overleftarrow{m}_1, \overrightarrow{m}_1), (\overleftarrow{n}_1, \overrightarrow{n}_1))$ and $M_2 = ((\overleftarrow{l}_2, \overrightarrow{l}_2), (\overleftarrow{m}_2, \overrightarrow{m}_2), (\overleftarrow{n}_2, \overrightarrow{n}_2))$. Mathematical operations on these numbers are performed as follows:

$$\begin{aligned} & ((\overleftarrow{l}_1, \overrightarrow{l}_1), (\overleftarrow{m}_1, \overrightarrow{m}_1), (\overleftarrow{n}_1, \overrightarrow{n}_1)) \oplus ((\overleftarrow{l}_2, \overrightarrow{l}_2), (\overleftarrow{m}_2, \overrightarrow{m}_2), (\overleftarrow{n}_2, \overrightarrow{n}_2)) \\ & = ((\overleftarrow{l}_1, \overrightarrow{l}_1) + (\overleftarrow{l}_2, \overrightarrow{l}_2), (\overleftarrow{m}_1, \overrightarrow{m}_1) + (\overleftarrow{m}_2, \overrightarrow{m}_2), (\overleftarrow{n}_1, \overrightarrow{n}_1) + (\overleftarrow{n}_2, \overrightarrow{n}_2)) \end{aligned} \quad (6.7)$$

$$\begin{aligned} & ((\overleftarrow{l}_1, \overrightarrow{l}_1), (\overleftarrow{m}_1, \overrightarrow{m}_1), (\overleftarrow{n}_1, \overrightarrow{n}_1)) \otimes ((\overleftarrow{l}_2, \overrightarrow{l}_2), (\overleftarrow{m}_2, \overrightarrow{m}_2), (\overleftarrow{n}_2, \overrightarrow{n}_2)) \\ & \approx ((\overleftarrow{l}_1, \overrightarrow{l}_1)(\overleftarrow{l}_2, \overrightarrow{l}_2), (\overleftarrow{m}_1, \overrightarrow{m}_1)(\overleftarrow{m}_2, \overrightarrow{m}_2), (\overleftarrow{n}_1, \overrightarrow{n}_1)(\overleftarrow{n}_2, \overrightarrow{n}_2)) \end{aligned} \quad (6.8)$$

$$((\overleftarrow{l}_1, \overrightarrow{l}_1), (\overleftarrow{m}_1, \overrightarrow{m}_1), (\overleftarrow{n}_1, \overrightarrow{n}_1))^{-1} \approx \left(\frac{1}{(\overleftarrow{l}_1, \overrightarrow{l}_1)}, \frac{1}{(\overleftarrow{m}_1, \overrightarrow{m}_1)}, \frac{1}{(\overleftarrow{n}_1, \overrightarrow{n}_1)} \right). \quad (6.9)$$

Define the possibility rate of $M_1 \geq M_2$, then

$$V(M_1 \geq M_2) = \sup_{(\overleftarrow{x}, \overrightarrow{x}) \geq (\overleftarrow{y}, \overrightarrow{y})} (\wedge((\overleftarrow{\mu}_{M_1}, \overrightarrow{\mu}_{M_1})_{(\overleftarrow{x}, \overrightarrow{x})}, (\overleftarrow{\mu}_{M_2}, \overrightarrow{\mu}_{M_2})_{(\overleftarrow{y}, \overrightarrow{y})})), \quad (6.10)$$

in which $(\overleftarrow{m}_1, \overrightarrow{m}_1) \geq (\overleftarrow{m}_2, \overrightarrow{m}_2)$ is the sufficient and necessary condition of $V(M_1 \geq M_2) = 1$.

When $(\overleftarrow{m}_1, \overrightarrow{m}_1) < (\overleftarrow{m}_2, \overrightarrow{m}_2)$, define the possibility of $M_1 \geq M_2$ as follows:

$$V(M_1 \geq M_2) = hgt(M_1 \bigcap M_2) = (\overleftarrow{\mu}, \overrightarrow{\mu})_{(\overleftarrow{d}, \overrightarrow{d})}, \quad (6.11)$$

where $(\overleftarrow{d}, \overrightarrow{d})$ is the abscissa of the crossover between M_1, M_2 , as shown in Fig. 6.2.

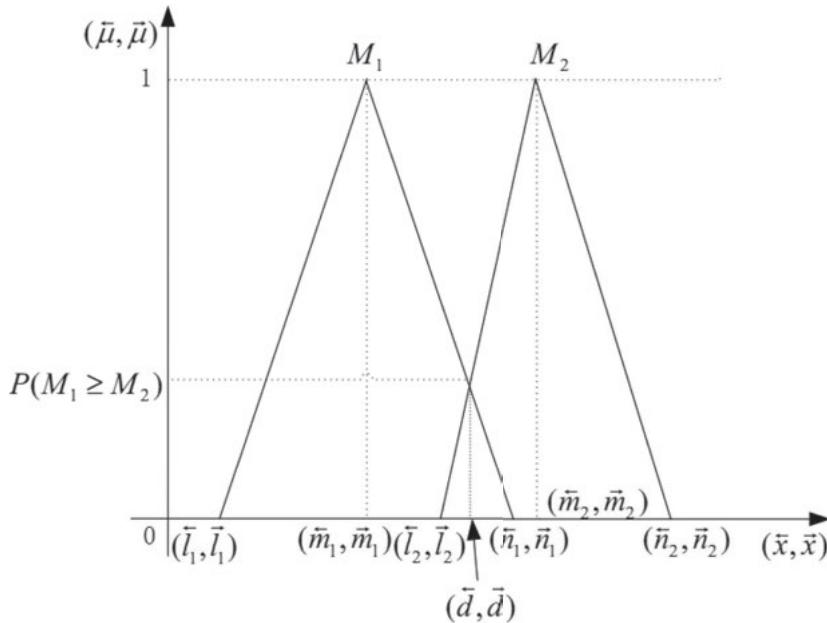


Fig. 6.2: Illustration for two triangular dynamic fuzzy numbers.

We can know,

$$(\overleftarrow{\mu}, \overrightarrow{\mu})_{(\overleftarrow{d}, \overrightarrow{d})} = \begin{cases} \frac{(\overleftarrow{l}_1, \overrightarrow{l}_1) - (\overleftarrow{n}_2, \overrightarrow{n}_2)}{((\overleftarrow{m}_2, \overrightarrow{m}_2) - (\overleftarrow{n}_2, \overrightarrow{n}_2)) - ((\overleftarrow{m}_1, \overrightarrow{m}_1) - (\overleftarrow{l}_1, \overrightarrow{l}_1))}, & (\overleftarrow{l}_1, \overrightarrow{l}_1) \leq (\overleftarrow{n}_2, \overrightarrow{n}_2) \\ 0, & \text{otherwise} \end{cases}. \quad (6.12)$$

Use $f = ((\overleftarrow{l}, \overrightarrow{l}), (\overleftarrow{m}, \overrightarrow{m}), (\overleftarrow{n}, \overrightarrow{n}))$ converse dynamic fuzzy matrix $(\overleftarrow{R}, \overrightarrow{R})$ into

$$(\overleftarrow{R}, \overrightarrow{R}) = \left(\begin{array}{ccc|ccc|ccc} ((\overleftarrow{r}_{11l}, \overrightarrow{r}_{11l}), & ((\overleftarrow{r}_{12l}, \overrightarrow{r}_{12l}), & ((\overleftarrow{r}_{1nl}, \overrightarrow{r}_{1nl}), & & & & & \\ (\overleftarrow{r}_{11m}, \overrightarrow{r}_{11m}), & (\overleftarrow{r}_{12m}, \overrightarrow{r}_{12m}), & (\overleftarrow{r}_{1nm}, \overrightarrow{r}_{1nm}), & & & & & \\ (\overleftarrow{r}_{11n}, \overrightarrow{r}_{11n})) & (\overleftarrow{r}_{12n}, \overrightarrow{r}_{12n})) & (\overleftarrow{r}_{1nn}, \overrightarrow{r}_{1nn})) & & & & & \\ \hline ((\overleftarrow{r}_{21l}, \overrightarrow{r}_{21l}), & ((\overleftarrow{r}_{22l}, \overrightarrow{r}_{22l}), & ((\overleftarrow{r}_{2nl}, \overrightarrow{r}_{2nl}), & & & & & \\ (\overleftarrow{r}_{21m}, \overrightarrow{r}_{21m}), & (\overleftarrow{r}_{22m}, \overrightarrow{r}_{22m}), & (\overleftarrow{r}_{2nm}, \overrightarrow{r}_{2nm}), & & & & & \\ (\overleftarrow{r}_{21n}, \overrightarrow{r}_{21n})) & (\overleftarrow{r}_{22n}, \overrightarrow{r}_{22n})) & (\overleftarrow{r}_{2nn}, \overrightarrow{r}_{2nn})) & & & & & \\ \vdots & \\ \hline ((\overleftarrow{r}_{n1l}, \overrightarrow{r}_{n1l}), & ((\overleftarrow{r}_{n2l}, \overrightarrow{r}_{n2l}), & ((\overleftarrow{r}_{nnl}, \overrightarrow{r}_{nnl}), & & & & & \\ (\overleftarrow{r}_{n1m}, \overrightarrow{r}_{n1m}), & (\overleftarrow{r}_{n2m}, \overrightarrow{r}_{n2m}), & (\overleftarrow{r}_{nnm}, \overrightarrow{r}_{nnm}), & & & & & \\ (\overleftarrow{r}_{n1n}, \overrightarrow{r}_{n1n})) & (\overleftarrow{r}_{n2n}, \overrightarrow{r}_{n2n})) & (\overleftarrow{r}_{nnn}, \overrightarrow{r}_{nnn})) & & & & & \end{array} \right).$$

Definition 6.10 Let $(\overleftarrow{r}_{ij}^t, \overrightarrow{r}_{ij}^t) = ((\overleftarrow{l}_{ij}^t, \overrightarrow{l}_{ij}^t), (\overleftarrow{m}_{ij}^t, \overrightarrow{m}_{ij}^t), (\overleftarrow{n}_{ij}^t, \overrightarrow{n}_{ij}^t))$ be the DFD given by the t th expert, where $i, j = 1, 2, \dots, n_k$, $t = 1, 2, \dots, N$. Then, the relationship matrix of all the elements in the k th level with respect to the h th element in the $(k-1)$ th level is given by the following formula:

$$M_{ij}^k = \frac{1}{N} \otimes ((\overleftarrow{r}_{ij}^1, \overrightarrow{r}_{ij}^1) + (\overleftarrow{r}_{ij}^2, \overrightarrow{r}_{ij}^2) + \dots + (\overleftarrow{r}_{ij}^N, \overrightarrow{r}_{ij}^N)). \quad (6.13)$$

The corresponding dynamic fuzzy comprehensive rate is

$$S_i^k = \sum_{j=1}^{n_k} M_{ij}^k \otimes \frac{1}{\sum_{i=1}^{n_k} \sum_{j=1}^{n_k} M_{ij}^k}. \quad (6.14)$$

Definition 6.11 Assume $(\overleftarrow{d}', \overrightarrow{d}')_{(\overleftarrow{R}_i, \overrightarrow{R}_i)} = \wedge V(S_i \geq S_k)$. The weight vector is written as

$$W' = ((\overleftarrow{d}', \overrightarrow{d}')_{(\overleftarrow{R}_1, \overrightarrow{R}_1)}, (\overleftarrow{d}', \overrightarrow{d}')_{(\overleftarrow{R}_2, \overrightarrow{R}_2)}, \dots, (\overleftarrow{d}', \overrightarrow{d}')_{(\overleftarrow{R}_n, \overrightarrow{R}_n)})^T, \quad (6.15)$$

where $k, i = 1, 2, \dots, n; k \neq i$.

After normalization, we get the normalized weight vector

$$W = ((\overleftarrow{d}, \overrightarrow{d})_{(\overleftarrow{R}_1, \overrightarrow{R}_1)}, (\overleftarrow{d}, \overrightarrow{d})_{(\overleftarrow{R}_2, \overrightarrow{R}_2)}, \dots, (\overleftarrow{d}, \overrightarrow{d})_{(\overleftarrow{R}_n, \overrightarrow{R}_n)})^T. \quad (6.16)$$

6.3.4 Sample analysis

Consider a customer who wishes to buy an automobile based mainly on the fuel consumption; that is, the learning goal is mpg. There are three types of automobile to choose from, C_1, C_2, C_3 , and four related evaluation attributes, c_1, c_2, c_3, c_4 . We wish to find the attribute that is most closely related to mpg, that is, the attribute that most affects city cycle fuel consumption. Attribute c_1 is the number of cylinders, c_2 is weight, c_3 is displacement, and c_4 is horsepower.

First, experts process the data and compare their opinions to set up the dynamic fuzzy matrix $(\overleftarrow{R}, \overrightarrow{R})$, as given in Tab. 6.2.

To improve reliability and stability, we calculate the average of Tab. 6.2 and gain Tab. 6.3. Using (6.10)–(6.14) in a MATLAB program, we calculate

$$W' = ((\overleftarrow{0.32}, \overrightarrow{0.32}), (\overleftarrow{0.96}, \overrightarrow{0.96}), (\overleftarrow{0.06}, \overrightarrow{0.06}), (\overleftarrow{1}, \overrightarrow{1}))^T.$$

After normalization, we obtain the weight vector for c_1, c_2, c_3, c_4 :

$$W = ((\overleftarrow{0.13}, \overrightarrow{0.13}), (\overleftarrow{0.41}, \overrightarrow{0.41}), (\overleftarrow{0.03}, \overrightarrow{0.03}), (\overleftarrow{0.43}, \overrightarrow{0.43}))^T.$$

Tab. 6.2: Dynamic fuzzy matrix (\vec{R}, \vec{R}) .

c_1	c_2	c_3	c_4
c_1 (1, 1, 1)	$\begin{pmatrix} (\overline{0.67}, \overline{0.67}) & (\overline{1}, \overline{1}) & (\overline{1.5}, \overline{1.5}) \\ (\overline{0.4}, \overline{0.4}) & (\overline{0.5}, \overline{0.5}) & (\overline{0.67}, \overline{0.67}) \\ (\overline{1.5}, \overline{1.5}) & (\overline{2}, \overline{2}) & (\overline{2.5}, \overline{2.5}) \end{pmatrix}$	$\left((\overline{0.67}, \overline{0.67}) (\overline{1}, \overline{1}) (\overline{1.5}, \overline{1.5}) \right)$	$\left((\overline{0.29}, \overline{0.29}) (\overline{0.33}, \overline{0.33}) (\overline{0.4}, \overline{0.4}) \right)$
c_2	$\begin{pmatrix} (\overline{0.67}, \overline{0.67}) & (\overline{1}, \overline{1}) & (\overline{1.5}, \overline{1.5}) \\ (\overline{1.5}, \overline{1.5}) & (\overline{2}, \overline{2}) & (\overline{2.5}, \overline{2.5}) \\ (\overline{0.4}, \overline{0.4}) & (\overline{0.5}, \overline{0.5}) & (\overline{0.67}, \overline{0.67}) \end{pmatrix}$	$\begin{pmatrix} (1, 1, 1) \\ (\overline{2.5}, \overline{2.5}) (\overline{3}, \overline{3}) (\overline{3.5}, \overline{3.5}) \\ (\overline{2.5}, \overline{2.5}) (\overline{3}, \overline{3}) (\overline{3.5}, \overline{3.5}) \end{pmatrix}$	$\left((\overline{0.4}, \overline{0.4}) (\overline{1}, \overline{1}) (\overline{1.5}, \overline{1.5}) \right)$
c_3	$\left((\overline{0.67}, \overline{0.67}) (\overline{1}, \overline{1}) (\overline{1.5}, \overline{1.5}) \right)$	$\left((\overline{0.29}, \overline{0.29}) (\overline{0.33}, \overline{0.33}) (\overline{0.4}, \overline{0.4}) \right)$	$(1, 1, 1)$
c_4	$\begin{pmatrix} (\overline{0.67}, \overline{0.67}) & (\overline{1}, \overline{1}) & (\overline{1.5}, \overline{1.5}) \\ (\overline{0.4}, \overline{0.4}) & (\overline{0.5}, \overline{0.5}) & (\overline{0.67}, \overline{0.67}) \\ (\overline{1.5}, \overline{1.5}) & (\overline{2}, \overline{2}) & (\overline{2.5}, \overline{2.5}) \end{pmatrix}$	$\begin{pmatrix} (\overline{0.67}, \overline{0.67}) (\overline{1}, \overline{1}) (\overline{1.5}, \overline{1.5}) \\ (\overline{0.67}, \overline{0.67}) (\overline{1}, \overline{1}) (\overline{1.5}, \overline{1.5}) \\ (\overline{0.4}, \overline{0.4}) (\overline{0.5}, \overline{0.5}) (\overline{0.67}, \overline{0.67}) \end{pmatrix}$	$(1, 1, 1)$

Tab. 6.3: Average value of dynamic fuzzy matrix $(\overleftarrow{R}, \overrightarrow{R})$.

c_1	c_2	c_3	c_4
$c_1 (1, 1, 1)$	$((\overleftarrow{0.86}, \overrightarrow{0.86}), (\overleftarrow{1.17}, \overrightarrow{1.17}), (\overleftarrow{1.56}, \overrightarrow{1.56}))$	$((\overleftarrow{0.67}, \overrightarrow{0.67}), (\overleftarrow{1}, \overrightarrow{1}), (\overleftarrow{1.5}, \overrightarrow{1.5}))$	$((\overleftarrow{0.33}, \overrightarrow{0.33}), (\overleftarrow{0.39}, \overrightarrow{0.39}), (\overleftarrow{0.49}, \overrightarrow{0.49}))$
$c_2 ((\overleftarrow{0.64}, \overrightarrow{0.64}), (\overleftarrow{0.85}, \overrightarrow{0.85}), (\overleftarrow{1.16}, \overrightarrow{1.16}))$	$(1, 1, 1)$	$((\overleftarrow{2.5}, \overrightarrow{2.5}), (\overleftarrow{3}, \overrightarrow{3}), (\overleftarrow{3.5}, \overrightarrow{3.5}))$	$((\overleftarrow{0.95}, \overrightarrow{0.95}), (\overleftarrow{1.33}, \overrightarrow{1.33}), (\overleftarrow{1.83}, \overrightarrow{1.83}))$
$c_3 ((\overleftarrow{0.87}, \overrightarrow{0.87}), (\overleftarrow{1}, \overrightarrow{1}), (\overleftarrow{1.49}, \overrightarrow{1.49}))$	$((\overleftarrow{0.29}, \overrightarrow{0.29}), (\overleftarrow{0.33}, \overrightarrow{0.33}), (\overleftarrow{0.40}, \overrightarrow{0.40}))$	$(1, 1, 1)$	$((\overleftarrow{0.4}, \overrightarrow{0.4}), (\overleftarrow{0.5}, \overrightarrow{0.5}), (\overleftarrow{0.67}, \overrightarrow{0.67}))$
$c_4 ((\overleftarrow{2.04}, \overrightarrow{2.04}), (\overleftarrow{2.56}, \overrightarrow{2.56}), (\overleftarrow{3.03}, \overrightarrow{3.03}))$	$((\overleftarrow{0.55}, \overrightarrow{0.55}), (\overleftarrow{0.75}, \overrightarrow{0.75}), (\overleftarrow{1.05}, \overrightarrow{1.05}))$	$((\overleftarrow{1.49}, \overrightarrow{1.49}), (\overleftarrow{2}, \overrightarrow{2}), (\overleftarrow{2.5}, \overrightarrow{2.5}))$	$(1, 1, 1)$

Tab. 6.4: C value of dynamic fuzzy matrix $(\overleftarrow{R}, \overrightarrow{R})$.

	C_1	C_2	C_3
c_1	$(\overleftarrow{0.28}, \overrightarrow{0.28})$	$(\overleftarrow{0.21}, \overrightarrow{0.21})$	$(\overleftarrow{0.51}, \overrightarrow{0.51})$
c_2	$(\overleftarrow{0.66}, \overrightarrow{0.66})$	$(\overleftarrow{0.16}, \overrightarrow{0.16})$	$(\overleftarrow{0.19}, \overrightarrow{0.19})$
c_3	$(\overleftarrow{0.35}, \overrightarrow{0.35})$	$(\overleftarrow{0.33}, \overrightarrow{0.33})$	$(\overleftarrow{0.32}, \overrightarrow{0.32})$
c_4	$(\overleftarrow{0.22}, \overrightarrow{0.22})$	$(\overleftarrow{0.42}, \overrightarrow{0.42})$	$(\overleftarrow{0.36}, \overrightarrow{0.36})$

Tab. 6.5: Final weight of dynamic fuzzy matrix $(\overleftarrow{R}, \overrightarrow{R})$.

	C_1	C_2	C_3
Final weight	$(\overleftarrow{0.41}, \overrightarrow{0.41})$	$(\overleftarrow{0.28}, \overrightarrow{0.28})$	$(\overleftarrow{0.25}, \overrightarrow{0.25})$

Thus, the influence rate on mpg, from high to low, is horsepower, weight, cylinders, and displacement. This result is the same as reported in [56], where SVM was used to extract features from the same database.

Next, in the same case, the results were calculated according to C for each attribute c in Tab. 6.4.

Finally, we added the weight of this level and the multiple upper level weights. The final results are presented in Tab. 6.5.

Above all, the customer should choose car C_1 .

6.4 Dynamic fuzzy tree hierarchical relation learning

The tree structure is an important kind of nonlinear data format that is widely observed in the real world, such as in the genealogy of human society and various social organization structures. Intuitively, the tree defines the hierarchical structure through the branch relation. Dynamic fuzzy theory is used to construct a dynamic fuzzy tree representation method, which is more suitable for processing uncertain data in the subjective and objective world.

6.4.1 Dynamic fuzzy tree

The following uses the DF formulation to explain the basic concept of dynamic fuzzy trees.

There is a complicated relationship between different things or knowledge in the objective world. The causal relation is the most common and simple form. Therefore,

production is a representation that is suitable for relationships. From a semantic perspective, this represents the causal inference “If A, then B”.

1) DF production definition [48]

The general form of a DF production is

$$P \leftarrow Q, CDF, I,$$

where the right-hand side represents a group of conditions or premises and the left-hand side represents various premises Q and the conclusion P of the conclusion action that can belong to DF. CDF is called the rule confidence, and $(\overleftarrow{0}, \overrightarrow{0}) \leq CDF \leq (\overleftarrow{1}, \overrightarrow{1})$. The above rule means that if premise Q can be satisfied, then we can induce whether conclusion P is true or execute action P with some degree of truth, where the confidence of the rule is CDF.

In the representation of production, premise Q is a template r , and “satisfied” means the template can be matched in some cases. Theoretically, we can use the form of a string to represent this. For example, production in formal language such as $\alpha BB \leftarrow \alpha AB$ means that, if there exists a substring in a string that matches αAB with r , we can exchange the substring with αAB . Thus, it is a more general expressive module. As a special string structure, the logical formula can be viewed as a special case of a module.

For example, the premise of implication $P(\overleftarrow{x}, \overrightarrow{x}) \leftarrow Q(\overleftarrow{x}, \overrightarrow{x})$ can be viewed as a string $Q(\overleftarrow{x}, \overrightarrow{x})$ made up of Q , “(”, “($\overleftarrow{x}, \overrightarrow{x}$)”, “)”. Only if there exists a string that matches r can the premise of the rule be satisfied, so that $P(\overleftarrow{x}, \overrightarrow{x})$ can be induced. Thus, we determine whether to use the rule by checking if the premise can be satisfied instead of calculating the truth value of the premise predicate in the production rule. This is the essential difference between production and implication. It is certain that if the appointment can be matched, the predicate is true; otherwise, it is false. In this sense, production contains implication.

Naturally, the match proposed above should define the DF match in some sense instead of giving completely the same meaning. Similar to using the semantic distance to measure the distance between DF data objects, we design a method to measure the degree of piecewise matching between any templates. When the matching degree reaches a certain level, it is considered that the match is successful and we can use a production rule.

In addition, the DF string sequence can be used as premises or conditions to make the premise and condition of production dynamic and fuzzy:

$$Q = a_1(\overleftarrow{u_1}, \overrightarrow{u_1})a_2(\overleftarrow{u_2}, \overrightarrow{u_2})\dots a_n(\overleftarrow{u_n}, \overrightarrow{u_n}),$$

where a_i is a character, $(\overleftarrow{u_i}, \overrightarrow{u_i})$ is the membership rate of a_i to string Q , and $(\overleftarrow{0}, \overrightarrow{0}) \leq (\overleftarrow{u_i}, \overrightarrow{u_i}) \leq (\overleftarrow{1}, \overrightarrow{1})$. When $(\overleftarrow{u_i}, \overrightarrow{u_i}) = (\overleftarrow{1}, \overrightarrow{1})$, the i th character of string Q is a_i .

When $(\overleftarrow{u}_i, \overrightarrow{u}_i) = (\overleftarrow{0}, \overrightarrow{0})$, the i th character of string Q is any character but a_i . When $(\overleftarrow{0}, \overrightarrow{0}) < (\overleftarrow{u}_i, \overrightarrow{u}_i) < (\overleftarrow{1}, \overrightarrow{1})$, the i th character of string Q is a_i with probability $(\overleftarrow{u}_i, \overrightarrow{u}_i)$. We use $*$ to represent the wildcard that can match any characters.

2) DF match

Assume the production

$$P \leftarrow Q, CDF, I.$$

When $Q = a_1(\overleftarrow{u}_1, \overrightarrow{u}_1)a_2(\overleftarrow{u}_2, \overrightarrow{u}_2)\dots a_n(\overleftarrow{u}_n, \overrightarrow{u}_n)$, if $Q' = a'_1(\overleftarrow{v}_1, \overrightarrow{v}_1)a'_2(\overleftarrow{v}_2, \overrightarrow{v}_2)\dots a'_n(\overleftarrow{v}_n, \overrightarrow{v}_n)$, $(\overleftarrow{t}', \overrightarrow{t'})$, $(\overleftarrow{0}, \overrightarrow{0}) < (\overleftarrow{t}', \overrightarrow{t'}) \leq (\overleftarrow{1}, \overrightarrow{1})$, where $(\overleftarrow{t}', \overrightarrow{t'})$ is the true value of knowledge Q' , $a'_i = a_i$ or $*$, $(\overleftarrow{0}, \overrightarrow{0}) \leq (\overleftarrow{u}_i, \overrightarrow{u}_i) \leq (\overleftarrow{v}_i, \overrightarrow{v}_i) \leq (\overleftarrow{1}, \overrightarrow{1})$ (or $(\overleftarrow{0}, \overrightarrow{0}) \leq (\overleftarrow{v}_i, \overrightarrow{v}_i) \leq (\overleftarrow{u}_i, \overrightarrow{u}_i) \leq (\overleftarrow{1}, \overrightarrow{1})$), $i = 1, 2, \dots, n$, then Q can be matched.

The distance between Q and Q' is

$$(\overleftarrow{d}, \overrightarrow{d}) = \max\{|\overleftarrow{(v_1, v_1)} - \overleftarrow{(u_1, u_1)}|, |\overleftarrow{(v_2, v_2)} - \overleftarrow{(u_2, u_2)}|, \dots, |\overleftarrow{(v_n, v_n)} - \overleftarrow{(u_n, u_n)}|\}.$$

The definition of the matching rate of premise Q is

$$(\overleftarrow{m}, \overrightarrow{m}) = \min\{1 - (\overleftarrow{d}, \overrightarrow{d}), (\overleftarrow{t}', \overrightarrow{t'})\} (\text{or } (\overleftarrow{m}, \overrightarrow{m}) = (\overleftarrow{t}', \overrightarrow{t'}) * (1 - (\overleftarrow{d}, \overrightarrow{d}))).$$

3) Representation and/or tree for the DF production rule

A group of production can be represented by an and/or tree. For example, the following production group [48] is known:

$$\begin{aligned} (\overleftarrow{A}, \overrightarrow{A}) &\leftarrow (\overleftarrow{B}_1, \overrightarrow{B}_1), (\overleftarrow{B}_2, \overrightarrow{B}_2) \quad (CDF_1, I_1) \\ (\overleftarrow{A}, \overrightarrow{A}) &\leftarrow (\overleftarrow{B}_3, \overrightarrow{B}_3), (\overleftarrow{B}_4, \overrightarrow{B}_4), (\overleftarrow{B}_5, \overrightarrow{B}_5) \quad (CDF_2, I_2) \\ (\overleftarrow{B}_1, \overrightarrow{B}_1) &\leftarrow (\overleftarrow{C}_1, \overrightarrow{C}_1), (\overleftarrow{C}_2, \overrightarrow{C}_2) \quad (CDF_3, I_3) \\ (\overleftarrow{B}_2, \overrightarrow{B}_2) &\leftarrow (\overleftarrow{C}_3, \overrightarrow{C}_3) \quad (CDF_4, I_4) \\ (\overleftarrow{B}_3, \overrightarrow{B}_3) &\leftarrow (\overleftarrow{C}_4, \overrightarrow{C}_4), (\overleftarrow{C}_5, \overrightarrow{C}_5), (\overleftarrow{C}_6, \overrightarrow{C}_6) \quad (CDF_5, I_5) \\ (\overleftarrow{B}_4, \overrightarrow{B}_4) &\leftarrow (\overleftarrow{C}_7, \overrightarrow{C}_7) \quad (CDF_6, I_6) \\ (\overleftarrow{B}_4, \overrightarrow{B}_4) &\leftarrow (\overleftarrow{C}_8, \overrightarrow{C}_8) \quad (CDF_7, I_7) \\ (\overleftarrow{B}_5, \overrightarrow{B}_5) &\leftarrow (\overleftarrow{C}_9, \overrightarrow{C}_9), (\overleftarrow{C}_{10}, \overrightarrow{C}_{10}) \quad (CDF_9, I_9) \\ (\overleftarrow{C}_5, \overrightarrow{C}_5) &\leftarrow (\overleftarrow{D}_1, \overrightarrow{D}_1), (\overleftarrow{D}_2, \overrightarrow{D}_2), (\overleftarrow{D}_3, \overrightarrow{D}_3) \quad (CDF_{10}, I_{10}) \\ (\overleftarrow{C}_8, \overrightarrow{C}_8) &\leftarrow (\overleftarrow{D}_4, \overrightarrow{D}_4), (\overleftarrow{D}_5, \overrightarrow{D}_5), (\overleftarrow{D}_6, \overrightarrow{D}_6) \quad (CDF_{11}, I_{11}). \end{aligned}$$

We represent this in the form of a dynamic fuzzy tree, as shown in Fig. 6.3.

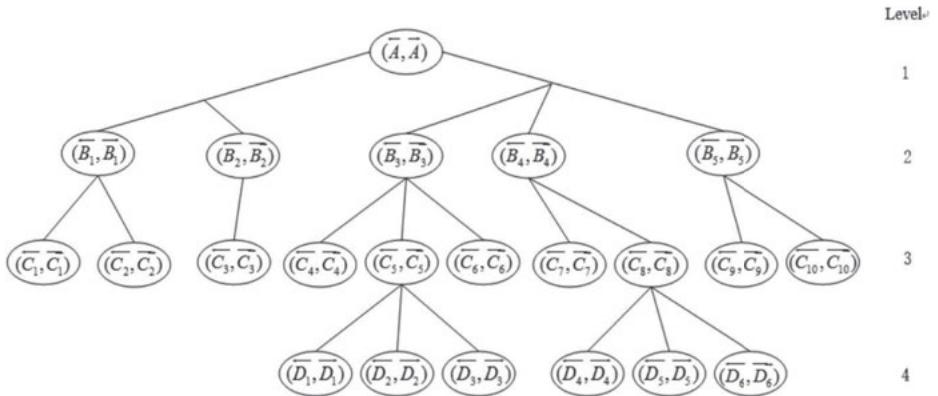


Fig. 6.3: DF production represented by dynamic fuzzy tree.

For the root node $(\overleftarrow{A}, \overrightarrow{A})$, the other nodes are divided into five non-overlapping limited subsets:

$$\begin{aligned}
 T_1 &= \{(\overleftarrow{B}_1, \overrightarrow{B}_1), (\overleftarrow{C}_1, \overrightarrow{C}_1), (\overleftarrow{C}_2, \overrightarrow{C}_2)\}, \\
 T_2 &= \{(\overleftarrow{B}_2, \overrightarrow{B}_2), (\overleftarrow{C}_3, \overrightarrow{C}_3)\}, \\
 T_3 &= \{(\overleftarrow{B}_3, \overrightarrow{B}_3), (\overleftarrow{C}_4, \overrightarrow{C}_4), (\overleftarrow{C}_5, \overrightarrow{C}_5), (\overleftarrow{C}_6, \overrightarrow{C}_6), (\overleftarrow{D}_1, \overrightarrow{D}_1), (\overleftarrow{D}_2, \overrightarrow{D}_2), (\overleftarrow{D}_3, \overrightarrow{D}_3)\}, \\
 T_4 &= \{(\overleftarrow{B}_4, \overrightarrow{B}_4), (\overleftarrow{C}_7, \overrightarrow{C}_7), (\overleftarrow{C}_8, \overrightarrow{C}_8), (\overleftarrow{D}_4, \overrightarrow{D}_4), (\overleftarrow{D}_5, \overrightarrow{D}_5), (\overleftarrow{D}_6, \overrightarrow{D}_6)\}, \text{ and} \\
 T_5 &= \{(\overleftarrow{B}_5, \overrightarrow{B}_5), (\overleftarrow{C}_9, \overrightarrow{C}_9), (\overleftarrow{C}_{10}, \overrightarrow{C}_{10})\}.
 \end{aligned}$$

Each set is also a dynamic fuzzy sub-tree of the root $(\overleftarrow{A}, \overrightarrow{A})$. For example, T_4 has the root $(\overleftarrow{B}_4, \overrightarrow{B}_4)$, and the other nodes are divided into two non-overlapping subsets $T_{41} = \{(\overleftarrow{C}_7, \overrightarrow{C}_7)\}$ and $T_{42} = \{(\overleftarrow{C}_8, \overrightarrow{C}_8), (\overleftarrow{D}_4, \overrightarrow{D}_4), (\overleftarrow{D}_5, \overrightarrow{D}_5), (\overleftarrow{D}_6, \overrightarrow{D}_6)\}$. T_{41} and T_{42} are both sub-trees of T_4 . $(\overleftarrow{C}_8, \overrightarrow{C}_8)$ is the root in T_{42} , and $(\overleftarrow{D}_4, \overrightarrow{D}_4)$, $(\overleftarrow{D}_5, \overrightarrow{D}_5)$, $(\overleftarrow{D}_6, \overrightarrow{D}_6)$ are three non-overlapping sub-trees of $(\overleftarrow{C}_8, \overrightarrow{C}_8)$, which itself is a tree with one node.

6.4.2 Dynamic fuzzy tree hierarchy relationship learning algorithm

1. ID3 algorithm and sample analysis

Decision tree learning is one of the most widely used inductive reasoning algorithms for learning disjunctive expressions. The decision tree classifies an instance from the root node to a leaf node, and the leaf nodes are the classification of instances. Each node in the tree specifies a test for a property of the instance, and each successive layer of the node corresponds to a possible value of the property. The method of

classification is to start from the root node of the tree, test the properties specified by the node, and then move down according to the branch corresponding to the value of the property for the given instance. This process is then repeated with the new node as the root of the sub-tree.

ID3 is based on the information entropy decision tree classification algorithm, using the top-down structure of the decision tree to learn [52]. According to the attribute value used to judge the category of samples, the statistical attribute of information gain is introduced to measure the ability of the given attribute to distinguish the training samples.

Consider a set of training samples X consisting of N_i training samples belonging to class i . The probability of a sample belonging to class i is p_i , and

$$p_i \equiv \frac{N_i}{|X|}, \quad (6.17)$$

where $|X|$ is the number of samples in X .

To precisely define the information gain, the entropy metric is widely used in information theory to describe the purity of any sample set [52].

Assume the objective attribute has k different values. The entropy of the training set X corresponding to k classification states is defined as

$$\text{Entropy}(X) \equiv \sum_{i=1}^k -p_i \text{lb } p_i. \quad (6.18)$$

The definition of some attribute A corresponding to the information gain $\text{Gain}(X, A)$ is

$$\text{Gain}(X, A) \equiv \text{Entropy}(X) - \sum_{v \in \text{Values}(A)} \frac{|X_v|}{|X|} \text{Entropy}(X_v), \quad (6.19)$$

where $\text{Values}(A)$ is the set of all possible values for attribute A , X_v is the subset of values v of attribute A ; that is, $X_v = \{x \in X | A(x) = v\}$. The first term of this formula is the entropy of the original set X , and the second term is the expectation of the entropy of attribute A classified by X .

The following uses the traffic situation of a city as an example to explain the actual procedure of the ID3 algorithm. The training samples are listed in Tab. 6.6.

When the algorithm starts, all samples are contained in the root node. To determine the best division attribute, calculate the entropy of the training samples according to (6.18):

$$\text{Entropy}(X) \equiv -\frac{6}{12} \text{lb } \frac{6}{12} - \frac{6}{12} \text{lb } \frac{6}{12} = 1.$$

Tab. 6.6: Sample set of traffic conditions in a city.

Day	Rushhour	Weather	Trafficjam	Annlate
D1	False	Good	False	Positive
D2	True	Good	True	Positive
D3	True	Normal	False	Negative
D4	False	Bad	False	Negative
D5	True	Normal	True	Positive
D6	False	Good	True	Positive
D7	True	Bad	False	Positive
D8	True	Bad	True	Negative
D9	True	Good	False	Positive
D10	False	Normal	False	Negative
D11	False	Normal	True	Negative
D12	False	Bad	True	Negative

Then, we can calculate the entropy of each attribute in the training sample set:

$$\text{Entropy}(X_{\text{rushhour}}, \text{true}) \equiv -\frac{4}{12} \text{lb} \frac{4}{12} - \frac{2}{6} \text{lb} \frac{2}{6} = 0.918$$

and

$$\text{Entropy}(X_{\text{rushhour}}, \text{false}) \equiv -\frac{2}{6} \text{lb} \frac{2}{6} - \frac{4}{6} \text{lb} \frac{4}{6} = 0.918.$$

Calculate the information gain of attribute “Rushhour” according to (6.19):

$$\begin{aligned} \text{Gain}(X, \text{Rushhour}) &\equiv \text{Entropy}(X) - \sum_{v \in \{\text{true, false}\}} \frac{|X_v|}{|X|} \text{Entropy}(X_v) \\ &= 1 - \left(\frac{1}{2} \times 0.918 + \frac{1}{2} \times 0.918 \right) = 0.082. \end{aligned}$$

In the same way, we can obtain information for attributes “Weather” and “Trafficjam”: $\text{Gain}(X, \text{Weather}) = 0.459$ and $\text{Gain}(X, \text{Trafficjam}) = 0$.

Overall, the information gain of “Weather” is the greatest, so the classification attribute on this node is “Weather”. This procedure is repeated to obtain the decision tree shown in Fig. 6.4.

2. Analysis and representation of dynamic fuzzy problems using decision trees

The ID3 decision tree algorithm eventually establishes the final decision tree, from which we identify information on the relevant attributes of the example set. Tab. 6.6 presents the sample set, including “Rushhour”, “Weather”, “Trafficjam”, and

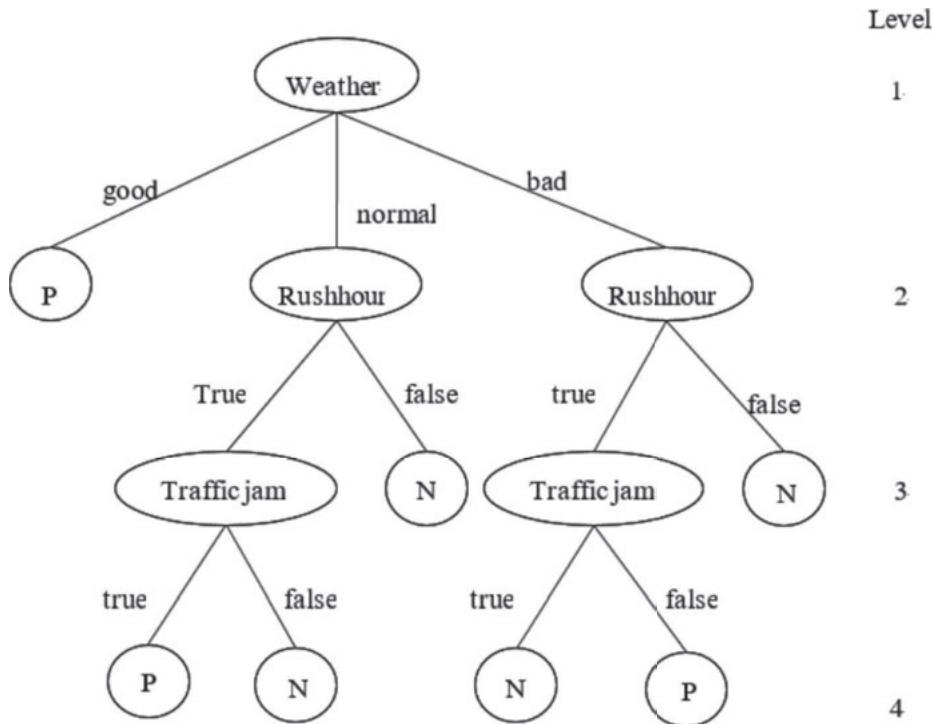


Fig. 6.4: Decision tree trained by ID3 algorithm.

“Annlate”. Taking ““Weather” as an example to illustrate the analysis, its range is $\{good, normal, bad\}$. These three properties are not clear – there is no exact standard to measure what is good, normal, or bad, that is, the condition of the attribute value of “Weather” is blurred; on the other hand, the “Weather” is constantly changing and developing, which indicates that it has a dynamic and fuzzy character. It can be seen that the decision tree learning system, which is based on the operation of the attribute, is a system with dynamic fuzziness.

Knowledge representation of dynamic fuzzy problems in decision trees is as follows.

- (1) For attribute x_i , use a value of attribute x_i to replace x_i . This is represented as $(\overleftarrow{x_i}, \overrightarrow{x_i})$, where the attribute has dynamic fuzziness and is a dynamic fuzzy number; in Tab. 6.6, the attribute “Rushhour” is replaced by “Rushhour=true”, and its range is $(0.7, 0.3)$, which is the attribute value after being transformed using the dynamic fuzzy number $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$ to represent the degree to which Rushhour is true or false.
- (2) For samples $\langle x, c(x) \rangle$, where $x = \langle x_1, x_2, \dots, x_i \rangle$, x represents the training samples, and x_i represents the attributes contained in the attribute. According

to (1), the attribute has dynamic fuzziness, and the dynamic fuzziness of the samples is reflected by its attribute value. Thus, we replace $(\overleftarrow{x}, \overrightarrow{x})$ by x and use the dynamic fuzzy number $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$ to represent $c(x)$, that is, $c(\overleftarrow{x}, \overrightarrow{x}) = (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \in [\overleftarrow{0}, \overrightarrow{0}] \times [\overleftarrow{1}, \overrightarrow{1}]$.

- (3) Every hypothesis h in the hypothesis space H is represented by the dynamic fuzzy Boolean quantity $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$, because $h \in H$ represents the function on $X : h : X \rightarrow [0, 1] \times [\leftarrow, \rightarrow]$. Now, $h(\overleftarrow{x}, \overrightarrow{x}) = (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \in [\overleftarrow{0}, \overrightarrow{0}] \times [\overleftarrow{1}, \overrightarrow{1}]$.
- (4) The sample set U contains samples whose dynamic fuzziness is represented by its attribute values. Thus, replace U with the dynamic fuzzy set $(\overleftarrow{U}, \overrightarrow{U})$ representing the range of the sample set in the dynamic fuzzy space.

Use the above knowledge to represent sample D_9 in Tab. 6.6 as given in Tab. 6.7.

After representing the sample with a dynamic fuzzy set and DFL, the decision tree constructed by the samples has also changed. After dynamic fuzzification, Fig. 6.4 becomes Fig. 6.5.

The tree produced by the samples based on DFS and DFL is called a dynamic fuzzy tree (DFT).

Definition 6.12 Assume $h_i = <(\overleftarrow{\alpha_1}, \overrightarrow{\alpha_1}), (\overleftarrow{\alpha_2}, \overrightarrow{\alpha_2}), \dots, (\overleftarrow{\alpha_m}, \overrightarrow{\alpha_m})>$, $h_j = <(\overleftarrow{\beta_1}, \overrightarrow{\beta_1}), (\overleftarrow{\beta_2}, \overrightarrow{\beta_2}), \dots, (\overleftarrow{\beta_m}, \overrightarrow{\beta_m})>$, and h_i, h_j are functions defined in $(\overleftarrow{X}, \overrightarrow{X})$. h_i is equal to or more general than h_j if and only if

$$\forall (\overleftarrow{x}, \overrightarrow{x}) \in (\overleftarrow{X}, \overrightarrow{X}), [(h_i(\overleftarrow{x}, \overrightarrow{x}) = (\overleftarrow{\alpha}, \overrightarrow{\alpha})) \rightarrow (h_j(\overleftarrow{x}, \overrightarrow{x}) = (\overleftarrow{\beta}, \overrightarrow{\beta}))], \quad (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \leq (\overleftarrow{\beta}, \overrightarrow{\beta}).$$

We write this as $h_i \geq_g h_j$. Here, $(\overleftarrow{x}, \overrightarrow{x})$ represents a sample. We write $h_i >_g h_j$ when h_i is strictly more general than h_j .

The relationship \geq_g represents a kind of partial ordering relation on the sample $(\overleftarrow{U}, \overrightarrow{U})$ (the relation is reflexive, anti-symmetric, and transitive). This provides an effective structure for learning problems. For the relationship \geq_g , a more general explanation for h_i contains fewer instances of constraints, any of which may be divided into a class of the samples placed into the same category by h_j . This is because h_j contains more constraints than h_i , so h_i is more general than h_j .

In Fig. 6.5, the attributes “Weather=good”, “Rushhour=true”, and “Trafficjam=true” divide the training samples. The left side of the tree structure

Tab. 6.7: D_9 knowledge representation after dynamic fuzzification.

Day	Rushhour=true	Weather=good	Trafficjam=false	Annlate=P
D9	$\overleftarrow{0.7}$	$\overrightarrow{0.6}$	$\overleftarrow{0.4}$	$\overleftarrow{0.8}$

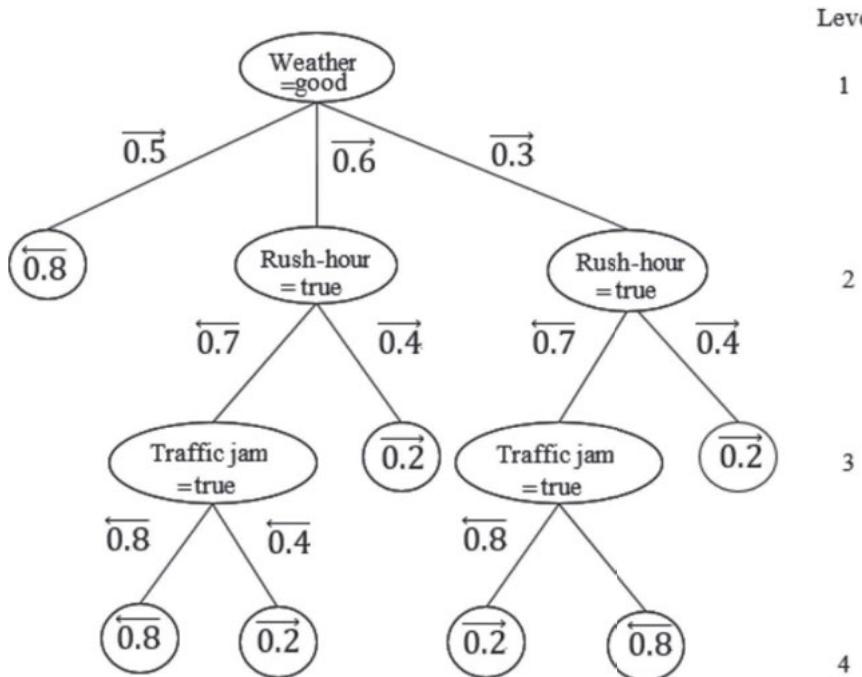


Fig. 6.5: Decision tree after dynamic fuzzification.

consists of samples in each node of the decision tree, where the relation between the upper set and lower set is \geq_g . For example, $\{D3, D5, D10, D11\} \geq_g \{D3, D5\}$, $\{D3, D5, D10, D11\} \geq_g \{D10, D11\}$. Therefore, the set consists of samples in the decision tree divided by a partial order relation. The right side is the result of dividing each layer of the sample set. From the division of each layer, the classification attribute progressively divides the sample set.

3. Dynamic fuzzy tree HRL algorithm

In dynamic fuzzy tree HRL, let $(\overleftarrow{a}_i, \overrightarrow{a}_i)$ represent an attribute with value range $(\overleftarrow{V}_{\overleftarrow{a}_i}, \overrightarrow{V}_{\overrightarrow{a}_i}) = \{(\overleftarrow{v}_1, \overrightarrow{v}_1), (\overleftarrow{v}_2, \overrightarrow{v}_2), \dots, (\overleftarrow{v}_k, \overrightarrow{v}_k)\}$. Then, $|(\overleftarrow{V}_{\overleftarrow{a}_i}, \overrightarrow{V}_{\overrightarrow{a}_i})| = k$ represents the number of attribute values of $(\overleftarrow{a}_i, \overrightarrow{a}_i)$. A sample can be represented by

$$(\overleftarrow{I}_i, \overrightarrow{I}_i) = <(\overleftarrow{v}_{\overleftarrow{a}_i}^1, \overrightarrow{v}_{\overrightarrow{a}_i}^1), (\overleftarrow{v}_{\overleftarrow{a}_i}^2, \overrightarrow{v}_{\overrightarrow{a}_i}^2), \dots, (\overleftarrow{v}_{\overleftarrow{a}_m}^m, \overrightarrow{v}_{\overrightarrow{a}_m}^m)>,$$

where $(\overleftarrow{v}_{\overleftarrow{a}_i}^i, \overrightarrow{v}_{\overrightarrow{a}_i}^i)$ represents the i th attribute of $(\overleftarrow{a}_i, \overrightarrow{a}_i)$ and the sample set can be represented as $(\overleftarrow{U}, \overrightarrow{U}) = \{(\overleftarrow{I}_i, \overrightarrow{I}_i) | i = 1, 2, \dots, n\}$.

Above all, the set consists of the value $(\overleftarrow{t}, \overrightarrow{t})$ of attribute $(\overleftarrow{a_i}, \overrightarrow{a_i})$ in sample set $(\overleftarrow{U}, \overrightarrow{U})$, which can be represented by

$$\begin{aligned} Par_{(\overleftarrow{a_i}, \overrightarrow{a_i})}^{(\overleftarrow{t}, \overrightarrow{t})}(\overleftarrow{U}, \overrightarrow{U}) &= \{(\overleftarrow{S}, \overrightarrow{S}) | (\overleftarrow{S}, \overrightarrow{S}) \subseteq (\overleftarrow{U}, \overrightarrow{U}), (\overleftarrow{I_j}, \overrightarrow{I_j}) \in (\overleftarrow{S}, \overrightarrow{S}), (\overleftarrow{V_{\overleftarrow{a_i}}}, \overrightarrow{V_{\overrightarrow{a_i}}})(\overleftarrow{I_j}, \overrightarrow{I_j}) \\ &= (\overleftarrow{t}, \overrightarrow{t})\}, \end{aligned}$$

where $(\overleftarrow{t}, \overrightarrow{t}) = (\overleftarrow{v_{\overleftarrow{a_i}}^k}, \overrightarrow{v_{\overrightarrow{a_i}}^k})$ represents the k th attribute value of attribute $(\overleftarrow{a_i}, \overrightarrow{a_i})$ and $k \leq |(\overleftarrow{V_{\overleftarrow{a_i}}}, \overrightarrow{V_{\overrightarrow{a_i}}})|$.

According to the difference among the values of attribute $(\overleftarrow{a_i}, \overrightarrow{a_i})$, we obtain the division of $(\overleftarrow{U}, \overrightarrow{U})$, written as

$$Div_{(\overleftarrow{a_i}, \overrightarrow{a_i})}(\overleftarrow{U}, \overrightarrow{U}) = \{(\overleftarrow{S_1}, \overrightarrow{S_1}), (\overleftarrow{S_2}, \overrightarrow{S_2}), \dots, (\overleftarrow{S_n}, \overrightarrow{S_n}) | (\overleftarrow{S_i}, \overrightarrow{S_i}) = Par_{(\overleftarrow{a_i}, \overrightarrow{a_i})}^{(\overleftarrow{t}, \overrightarrow{t})}(\overleftarrow{U}, \overrightarrow{U})\},$$

where $n \leq |(\overleftarrow{V_{\overleftarrow{a_i}}}, \overrightarrow{V_{\overrightarrow{a_i}}})|$.

We write the condition attribute as $(\overleftarrow{A}, \overrightarrow{A}) = \{(\overleftarrow{A_1}, \overrightarrow{A_1}), (\overleftarrow{A_2}, \overrightarrow{A_2}), \dots, (\overleftarrow{A_m}, \overrightarrow{A_m})\}$, and the decision attribute is $(\overleftarrow{A_d}, \overrightarrow{A_d})$. Because $(\overleftarrow{A_d}, \overrightarrow{A_d})$ relies on condition attribute $(\overleftarrow{A_i}, \overrightarrow{A_i})$, there is a relationship between the condition attribute and decision attribute that satisfies one of the following three relationships. For the division $Par_{(\overleftarrow{A_d}, \overrightarrow{A_d})}^{(\overleftarrow{v_j}, \overrightarrow{v_j})}(\overleftarrow{U}, \overrightarrow{U})$ given by dividing the sample set $(\overleftarrow{U}, \overrightarrow{U})$ by the j th value of the decision attribute $(\overleftarrow{A_d}, \overrightarrow{A_d})$, $\exists (\overleftarrow{A_i}, \overrightarrow{A_i})$ such that one of $Par_{(\overleftarrow{A_i}, \overrightarrow{A_i})}^{(\overleftarrow{v_k}, \overrightarrow{v_k})}(\overleftarrow{U}, \overrightarrow{U}) \subseteq Par_{(\overleftarrow{A_d}, \overrightarrow{A_d})}^{(\overleftarrow{v_j}, \overrightarrow{v_j})}(\overleftarrow{U}, \overrightarrow{U})$, $Par_{(\overleftarrow{A_i}, \overrightarrow{A_i})}^{(\overleftarrow{v_k}, \overrightarrow{v_k})}(\overleftarrow{U}, \overrightarrow{U}) = Par_{(\overleftarrow{A_d}, \overrightarrow{A_d})}^{(\overleftarrow{v_j}, \overrightarrow{v_j})}(\overleftarrow{U}, \overrightarrow{U})$, or $Par_{(\overleftarrow{A_i}, \overrightarrow{A_i})}^{(\overleftarrow{v_k}, \overrightarrow{v_k})}(\overleftarrow{U}, \overrightarrow{U}) \supseteq Par_{(\overleftarrow{A_d}, \overrightarrow{A_d})}^{(\overleftarrow{v_j}, \overrightarrow{v_j})}(\overleftarrow{U}, \overrightarrow{U})$ holds. Considering that “Include” and “Included” are reciprocal computations, and “=” can be viewed as a special situation of “Include”, we take the first case as an example.

$Par_{(\overleftarrow{A_i}, \overrightarrow{A_i})}^{(\overleftarrow{v_k}, \overrightarrow{v_k})}(\overleftarrow{U}, \overrightarrow{U}) \subseteq Par_{(\overleftarrow{A_d}, \overrightarrow{A_d})}^{(\overleftarrow{v_j}, \overrightarrow{v_j})}(\overleftarrow{U}, \overrightarrow{U})$ represents the case when the value of the decision attribute $(\overleftarrow{A_d}, \overrightarrow{A_d})$ is $(\overleftarrow{v_j}, \overrightarrow{v_j})$, and the sample set contains the set consisting of condition attribute $(\overleftarrow{A_i}, \overrightarrow{A_i})$ whose value is $(\overleftarrow{v_k}, \overrightarrow{v_k})$; that is, when the value of the decision attribute $(\overleftarrow{A_i}, \overrightarrow{A_i})$ is $(\overleftarrow{v_j}, \overrightarrow{v_j})$, the determined sample division is class $(\overleftarrow{v_j}, \overrightarrow{v_j})$. Because $Par_{(\overleftarrow{A_i}, \overrightarrow{A_i})}^{(\overleftarrow{v_k}, \overrightarrow{v_k})}(\overleftarrow{U}, \overrightarrow{U}) \subseteq Par_{(\overleftarrow{A_d}, \overrightarrow{A_d})}^{(\overleftarrow{v_j}, \overrightarrow{v_j})}(\overleftarrow{U}, \overrightarrow{U})$, it is known that the other attribute values of condition attribute $(\overleftarrow{A_i}, \overrightarrow{A_i})$ except $(\overleftarrow{v_k}, \overrightarrow{v_k})$ can divide the sample into class $(\overleftarrow{v_j}, \overrightarrow{v_j})$.

At the same time, $\exists n \leq |(\overleftarrow{V}_{\overleftarrow{A}_i}, \overrightarrow{V}_{\overrightarrow{A}_i})|$ such that $Par_{(\overleftarrow{A}_d, \overrightarrow{A}_d)}^{(\overleftarrow{v}_j, \overrightarrow{v}_j)}(\overleftarrow{U}, \overrightarrow{U}) \subseteq \bigcup_{k=1}^n Par_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U})$ holds, and the membership rate of sample class $(\overleftarrow{v}_j, \overrightarrow{v}_j)$ divided by condition attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ is written as

$$Cer_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^j(\overleftarrow{U}, \overrightarrow{U}) = \max(|Par_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U})|) / \min(|\bigcup_{k=1}^n Par_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U})|),$$

where $Par_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U}) \subseteq Par_{(\overleftarrow{A}_d, \overrightarrow{A}_d)}^{(\overleftarrow{v}_j, \overrightarrow{v}_j)}(\overleftarrow{U}, \overrightarrow{U}) \subseteq \bigcup_{k=1}^n Par_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U})$.

Because there are n categories ($n = |(\overleftarrow{V}_{\overleftarrow{A}_d}, \overrightarrow{V}_{\overrightarrow{A}_d})|$), condition attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ has a division membership rate of each category, and the division membership of condition attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ for the whole sample set can be written as

$$Cer_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U}) = \sum_{k=1}^n pro_k \ Cer_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^k(\overleftarrow{U}, \overrightarrow{U}),$$

where $pro_k = |Par_{(\overleftarrow{A}_d, \overrightarrow{A}_d)}^{(\overleftarrow{v}_k, \overrightarrow{v}_k)}(\overleftarrow{U}, \overrightarrow{U})| / |(\overleftarrow{U}, \overrightarrow{U})|$ represents the rate of class $(\overleftarrow{v}_k, \overrightarrow{v}_k)$ samples.

In decision tree classification, attribute selection conditions are written as $\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U}) = 1 - Cer_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U}) = 1 - \sum_{k=1}^n pro_k \ Cer_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}^k(\overleftarrow{U}, \overrightarrow{U})$. Here, the range of $\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U})$ is $[0, 1] \times [\leftarrow, \rightarrow]$. When $\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U}) = (\overleftarrow{0}, \overrightarrow{0})$, the membership rate of condition attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ for the whole sample set is 1. That is, condition attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ can determine the partition of the sample set without uncertain elements. When $\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U}) = (\overleftarrow{1}, \overrightarrow{1})$, the membership rate of condition attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ for the whole sample set is 0. That is, condition attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ can produce a most uncertain partition of the sample set. Therefore, the greater the value of $\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U})$, the more certain the partition. Otherwise, the partition is more uncertain. Thus, in the attribute choosing algorithm, the attribute that minimizes $\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U})$ should be chosen as the classification attribute.

In the process of constructing the dynamic fuzzy tree, condition attribute $(\overleftarrow{A}_i, \overrightarrow{A}_i)$ should cover as many samples as possible to decrease the number of samples waiting to be classified. The attributes that satisfy $\min(\Gamma_{(\overleftarrow{A}_i, \overrightarrow{A}_i)}(\overleftarrow{U}, \overrightarrow{U}))$ satisfy the above conditions. Thus, we consider the contribution of each division of the decision attribute to the partition of the whole sample set comprehensively using pro_k and choose the best attribute to construct the division; attributes that have already been chosen cannot be chosen again.

The pseudocode of the dynamic fuzzy tree hierarchical relationship learning algorithm can be described as Algorithm 6.1.

Algorithm 6.1 Procedure buildDFTree(DFTreeNode, U)

```

//Initialization TreeNode←{ },BA
for each condition attribute  $(\overleftarrow{A_i}, \overrightarrow{A_i})$  do
    for j ← 1 to  $|\overleftarrow{V_{\overleftarrow{A_i}}}, \overrightarrow{V_{\overrightarrow{A_i}}}|$  do
        for  $\overleftarrow{V_{\overleftarrow{A_i}}}, \overrightarrow{V_{\overrightarrow{A_i}}}$  do
             $Par_{\overleftarrow{(A_i, A_i)}}^{(\overleftarrow{V_j}, \overrightarrow{V_j})}(\overleftarrow{U}, \overrightarrow{U})$ 
            repeat
        Get  $Div_{\overleftarrow{(A_i, A_i)}}(\overleftarrow{U}, \overrightarrow{U})$ 
        repeat
        Apply  $Cer_{\overleftarrow{(A_i, A_i)}}^j(\overleftarrow{U}, \overrightarrow{U})$  to calculate the certain degree of  $(\overleftarrow{U}, \overrightarrow{U})$  that is divided by  $(\overleftarrow{A_i}, \overrightarrow{A_i})$ 
        Get the classifying attribute BA by calculating  $\min(\overleftarrow{T_{(A_i, A_i)}}(\overleftarrow{U}, \overrightarrow{U}))$ 
        Use BA to split training set  $(\overleftarrow{U}, \overrightarrow{U})$  into subset  $\text{Split}(\overleftarrow{U}, \overrightarrow{U}) = \{(\overleftarrow{U}_1, \overrightarrow{U}_1), (\overleftarrow{U}_2, \overrightarrow{U}_2), \dots, (\overleftarrow{U}_n, \overrightarrow{U}_n)\}$ 
        for each subset  $(\overleftarrow{U}_i, \overrightarrow{U}_i)$  in  $\text{Split}(\overleftarrow{U}, \overrightarrow{U})$ 
            Create  $i$ th child of DFTreeNode
            if all records in  $(\overleftarrow{U}_i, \overrightarrow{U}_i)$  are in the same class
            then child is leaf, save  $(\overleftarrow{U}_i, \overrightarrow{U}_i)$  as LeafNode
            else save  $(\overleftarrow{A_i}, \overrightarrow{A_i})$  as no-LeafNode, buildDFTree(Child i,  $(\overleftarrow{U}_i, \overrightarrow{U}_i)$ )
            repeat

```

6.4.3 Sample analysis

We used data from the literature [1] (see Tab. 6.8) to illustrate the process of dynamic fuzzy tree HRL algorithm. First, dynamic fuzzy theory was applied to the data source, and then the hierarchical learning algorithm was used to classify the dynamic fuzzy set and build a DFT.

For the Outlook condition attribute, $\overrightarrow{0.7}$, $\overrightarrow{0.4}$, and $\overleftarrow{0.2}$ represent the dynamic fuzzy degree of sunny, overcast, and rain.

For the Temperature condition attribute, $\overleftarrow{0.8}$, $\overleftarrow{0.5}$, and $\overrightarrow{0.4}$ represent the dynamic fuzzy degree of hot, mild, and cold.

For the Humidity condition attribute, $\overleftarrow{0.6}$ and $\overrightarrow{0.4}$ represent the dynamic fuzzy degree of high and normal.

For the Windy condition attribute, $\overrightarrow{0.8}$ and $\overleftarrow{0.1}$ represent the dynamic fuzzy degree of yes and no.

For the decision attribute of Class, $\overrightarrow{0.7}$ and $\overleftarrow{0.3}$ represent the dynamic fuzzy degree of positive and negative.

Thus, for Quinlan's "playtennis" training sample set, dynamic fuzzification produces the dataset in Tab. 6.9.

Tab. 6.8: Quinlan's playtennis example.

Example	Outlook	Temperature	Humidity	Windy	Class
1	Sunny	Hot	High	No	Negative
2	Sunny	Hot	High	Yes	Negative
3	Overcast	Hot	High	No	Positive
4	Rain	Mild	High	No	Positive
5	Rain	Cool	Normal	No	Positive
6	Rain	Cool	Normal	Yes	Negative
7	Overcast	Cool	Normal	Yes	Positive
8	Sunny	Mild	High	No	Negative
9	Sunny	Cool	Normal	No	Positive
10	Rain	Mild	Normal	No	Positive
11	Sunny	Mild	Normal	Yes	Positive
12	Overcast	Mild	High	Yes	Positive
13	Overcast	Hot	Normal	No	Positive
14	Rain	Mild	High	Yes	Negative

Tab. 6.9: Quinlan's playtennis training sample set after dynamic fuzzified.

Example	Outlook=sunny	Temperature=hot	Humidity=high	Windy=yes	Class=positive
1	$\overrightarrow{0.7}$	$\overleftarrow{0.8}$	$\overleftarrow{0.6}$	$\overleftarrow{0.1}$	$\overleftarrow{0.3}$
2	$\overrightarrow{0.7}$	$\overleftarrow{0.8}$	$\overleftarrow{0.6}$	$\overleftarrow{0.8}$	$\overleftarrow{0.3}$
3	$\overrightarrow{0.4}$	$\overleftarrow{0.8}$	$\overleftarrow{0.6}$	$\overleftarrow{0.1}$	$\overrightarrow{0.7}$
4	$\overleftarrow{0.2}$	$\overleftarrow{0.5}$	$\overleftarrow{0.6}$	$\overleftarrow{0.1}$	$\overrightarrow{0.7}$
5	$\overleftarrow{0.2}$	$\overrightarrow{0.4}$	$\overrightarrow{0.4}$	$\overleftarrow{0.1}$	$\overrightarrow{0.7}$
6	$\overleftarrow{0.2}$	$\overrightarrow{0.4}$	$\overrightarrow{0.4}$	$\overrightarrow{0.8}$	$\overleftarrow{0.3}$
7	$\overrightarrow{0.4}$	$\overrightarrow{0.4}$	$\overrightarrow{0.4}$	$\overleftarrow{0.8}$	$\overrightarrow{0.7}$
8	$\overrightarrow{0.7}$	$\overleftarrow{0.5}$	$\overleftarrow{0.6}$	$\overleftarrow{0.1}$	$\overleftarrow{0.3}$
9	$\overrightarrow{0.7}$	$\overrightarrow{0.4}$	$\overrightarrow{0.4}$	$\overleftarrow{0.1}$	$\overrightarrow{0.7}$
10	$\overleftarrow{0.2}$	$\overleftarrow{0.5}$	$\overrightarrow{0.4}$	$\overleftarrow{0.1}$	$\overrightarrow{0.7}$
11	$\overrightarrow{0.7}$	$\overleftarrow{0.5}$	$\overrightarrow{0.4}$	$\overrightarrow{0.8}$	$\overrightarrow{0.7}$
12	$\overrightarrow{0.4}$	$\overleftarrow{0.5}$	$\overleftarrow{0.6}$	$\overleftarrow{0.8}$	$\overrightarrow{0.7}$
13	$\overrightarrow{0.4}$	$\overleftarrow{0.8}$	$\overrightarrow{0.4}$	$\overleftarrow{0.1}$	$\overrightarrow{0.7}$
14	$\overleftarrow{0.2}$	$\overleftarrow{0.5}$	$\overrightarrow{0.6}$	$\overrightarrow{0.8}$	$\overleftarrow{0.3}$

According to the algorithm described in Section 6.4.2, we classified the data in Tab. 6.9; $(\overrightarrow{a}, \overleftarrow{a})$, $(\overrightarrow{b}, \overleftarrow{b})$, $(\overrightarrow{c}, \overleftarrow{c})$, $(\overrightarrow{d}, \overleftarrow{d})$, and $(\overrightarrow{e}, \overleftarrow{e})$ represent the dynamic fuzzy degree of Outlook = sunny, Temperature = hot, Humidity = high, Windy = yes, and Class = positive. $(\overrightarrow{U}, \overleftarrow{U})$ represents the sample set in Tab. 6.9, where the number in the sample set has replaced the samples to simplify the representation (similarly hereinafter); that is, $(\overrightarrow{U}, \overleftarrow{U}) = \{E1, E2, \dots, E14\}$. The process can be described as follows.

First, divide the sample set $(\overleftarrow{U}, \overrightarrow{U})$ according to by each attribute:

$$Div_{(\overleftarrow{a}, \overrightarrow{a})}(\overleftarrow{U}, \overrightarrow{U}) = \{\{1, 2, 8, 9, 11\}, \{3, 7, 12, 13\}, \{4, 5, 6, 10, 14\}\};$$

$$Div_{(\overleftarrow{b}, \overrightarrow{b})}(\overleftarrow{U}, \overrightarrow{U}) = \{\{1, 2, 3, 13\}, \{4, 8, 10, 11, 12, 14\}, \{5, 6, 7, 9\}\};$$

$$Div_{(\overleftarrow{c}, \overrightarrow{c})}(\overleftarrow{U}, \overrightarrow{U}) = \{\{1, 2, 3, 4, 8, 12, 14\}, \{5, 6, 7, 9, 10, 11, 13\}\};$$

$$Div_{(\overleftarrow{d}, \overrightarrow{d})}(\overleftarrow{U}, \overrightarrow{U}) = \{\{2, 6, 7, 11, 12, 14\}, \{1, 3, 4, 5, 8, 9, 10, 13\}\}; \text{ and}$$

$$Div_{(\overleftarrow{e}, \overrightarrow{e})}(\overleftarrow{U}, \overrightarrow{U}) = \{\{3, 4, 5, 7, 9, 10, 11, 12, 13\}, \{1, 2, 6, 8, 14\}\}.$$

Then, calculate the classification membership degree of $(\overleftarrow{U}, \overrightarrow{U})$ divided by each attribute:

$$\Gamma_{(\overleftarrow{a}, \overrightarrow{a})}(\overleftarrow{U}, \overrightarrow{U}) = 1 - \sum_{k=1}^2 pro_k \ Cer_{(\overleftarrow{a}, \overrightarrow{a})}^k(\overleftarrow{U}, \overrightarrow{U}) = 1 - (\frac{4}{14} \times \frac{9}{14} + 0) = 0.816.$$

In the same way, we can obtain $\Gamma_{(\overleftarrow{b}, \overrightarrow{b})}(\overleftarrow{U}, \overrightarrow{U}) = 1$, $\Gamma_{(\overleftarrow{c}, \overrightarrow{c})}(\overleftarrow{U}, \overrightarrow{U}) = 1$, $\Gamma_{(\overleftarrow{d}, \overrightarrow{d})}(\overleftarrow{U}, \overrightarrow{U}) = 1$, $\Gamma_{(\overleftarrow{e}, \overrightarrow{e})}(\overleftarrow{U}, \overrightarrow{U}) = 1$.

Thus, we choose attribute $(\overleftarrow{a}, \overrightarrow{a})$ as the branch node of the first level, which is the root node of the DFT.

When $(\overleftarrow{a}, \overrightarrow{a}) = \overrightarrow{0.7}$:

$$Div_{(\overleftarrow{b}, \overrightarrow{b})}(Par_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U})) = \{\{1, 2\}, \{8, 11\}, \{9\}\};$$

$$Div_{(\overleftarrow{c}, \overrightarrow{c})}(Par_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U})) = \{\{1, 2, 8\}, \{9, 11\}\};$$

$$Div_{(\overleftarrow{d}, \overrightarrow{d})}(Par_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U})) = \{\{1, 8, 9\}, \{2, 11\}\};$$

$$Div_{(\overleftarrow{e}, \overrightarrow{e})}(Par_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U})) = \{\{1, 2, 8\}, \{9, 11\}\}; \text{ and}$$

$$\Gamma_{(\overleftarrow{b}, \overrightarrow{b})}(Par_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U})) = 0.5, \Gamma_{(\overleftarrow{c}, \overrightarrow{c})}(Par_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U})) = 0, \Gamma_{(\overleftarrow{d}, \overrightarrow{d})}(Par_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U})) = 1.$$

Hence, the hierarchical attribute is $(\overleftarrow{c}, \overrightarrow{c})$, $Par_{(\overleftarrow{c}, \overrightarrow{c})}^{\overrightarrow{0.4}}(Par_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U}))$ and $Par_{(\overleftarrow{c}, \overrightarrow{c})}^{\overrightarrow{0.6}}(Par_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.7}}(\overleftarrow{U}, \overrightarrow{U}))$ belong to the same category, and we can determine the division and end of the branch.

In the same way, when $(\overleftarrow{a}, \overrightarrow{a}) = \overleftarrow{0.2}$,

$$\Gamma_{(\overleftarrow{b}, \overrightarrow{b})}(Par_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.2}}(\overleftarrow{U}, \overrightarrow{U})) = 1, \Gamma_{(\overleftarrow{c}, \overrightarrow{c})}(Par_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.2}}(\overleftarrow{U}, \overrightarrow{U})) = 1, \Gamma_{(\overleftarrow{d}, \overrightarrow{d})}(Par_{(\overleftarrow{a}, \overrightarrow{a})}^{\overrightarrow{0.2}}(\overleftarrow{U}, \overrightarrow{U})) = 0.$$

For this scenario, the hierarchical attribute is $(\overleftarrow{d}, \overrightarrow{d})$, $Par_{(\overleftarrow{d}, \overrightarrow{d})}^{0.8}(Par_{(\overleftarrow{a}, \overrightarrow{a})}^{0.2}(\overleftarrow{U}, \overrightarrow{U}))$ and $Par_{(\overleftarrow{d}, \overrightarrow{d})}^{0.1}(Par_{(\overleftarrow{a}, \overrightarrow{a})}^{0.2}(\overleftarrow{U}, \overrightarrow{U}))$ belong to the same category, and we can determine the division and end of the branch.

After establishing the DFT, the sample set corresponding to all the leaf nodes has the same classification. At this point, the algorithm terminates, and the tree building operations cease. The whole sample set classification is complete, and the final DFT hierarchical relationship learning results are shown in Fig. 6.7.

Figure 6.6 shows the value of the decision attribute for the classification, the value of the selected attribute of the branch node to the next layer, and the corresponding sample set for each layer. It can be seen that each layer is a partition of the attribute, which indicates the contribution of the attribute to the final classification result. For nodes that are no longer classified, the corresponding sample set belongs to the same classification. The representation of the attribute value using a dynamic fuzzy number increases the intuitiveness of the DFT.

6.5 Dynamic fuzzy graph hierarchical relationship learning

In the tree structure, there are obvious hierarchical relationships among data elements. Indeed, data elements in each layer may relate to multiple elements of the next layer, but can relate to only one element of the next layer. In the graph structure, however, the relationship between nodes can be arbitrary – any two data elements in the graph may be related. Dynamic fuzzy graph structure descriptions of the classification relationship can avoid the inconsistency caused by dynamic fuzzy hierarchy trees and are more in line with real-life descriptions. According to the dynamic fuzzy graph representation method, this chapter proposes a dynamic fuzzy graph hierarchical relation learning algorithm and presents the results of an example test and comparison.

6.5.1 Basic concept of dynamic fuzzy graph

1. Relational concept of graph

Definition 6.13 Let $G = (V_G, E_G)$ be a graph, where V_G represents the finite non-empty set of data elements in the graph that are nodes and E_G represents the set of relationships between two nodes, i.e. a set of edges.

Definition 6.14 For nodes v, w in a graph, if $\langle v, w \rangle \in E_G$, (v, w) represents a bow between an initial point v and an end point w , the graph is called a direct graph. If $\langle v, w \rangle \in E_G$ and $\langle w, v \rangle \in E_G$, then E_G is symmetric. We can use the unordered pair

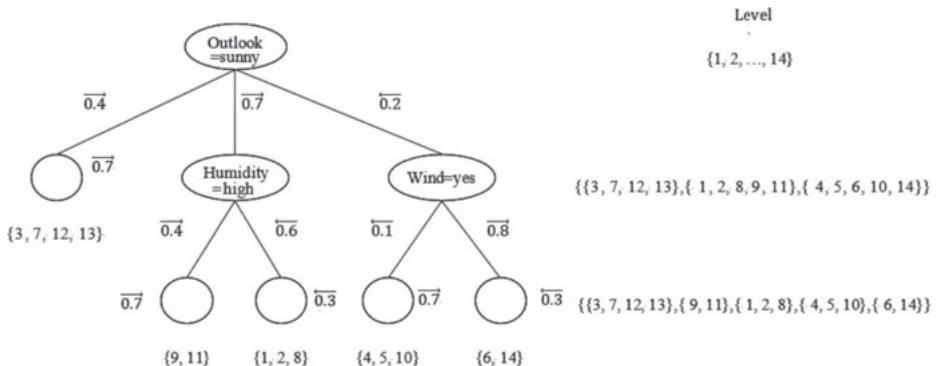


Fig. 6.6: DFT of Quinlan's playtennis example.

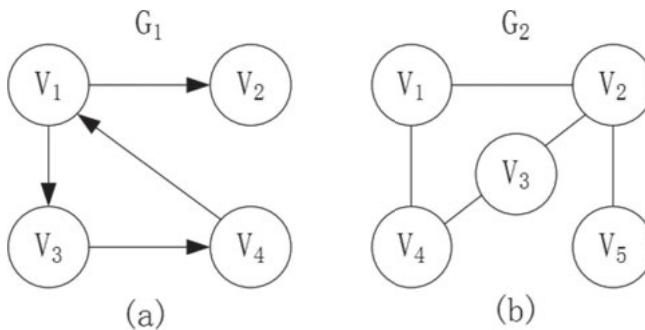


Fig. 6.7: Example of graph.

(v, w) to replace the two ordered pairs in representing that an edge exists between v and w ; this graph is called a non-direct graph. The predicate $P(v, w)$ represents the information or meaning of the bow $< v, w >$.

For example, G_1 in Fig. 6.7(a) is a direct graph, and the predicate $P(v, w)$ represents a one-way path from v to w :

$$G_1 = (V_1, E_1),$$

where $V_1 = \{v_1, v_2, v_3, v_4\}$ and $E_1 = \{< v_1, v_2 >, < v_1, v_3 >, < v_3, v_4 >, < v_4, v_1 >\}$.

G_2 in Fig 6.7(b) is a non-direct graph

$$G_2 = (V_2, E_2),$$

where $V_2 = \{v_1, v_2, v_3, v_4, v_5\}$ and $E_2 = \{(v_1, v_2), (v_1, v_4), (v_2, v_3), (v_2, v_5), (v_3, v_4), (v_3, v_5)\}$.

Definition 6.15 Assume there are two graphs $G = (V_G, E_G)$ and $G' = (V'_G, E'_G)$. If $V'_G \subseteq V_G$ and $E'_G \subseteq E_G$, then G' is a sub-graph of G , that is, G contains G' .

Definition 6.16 In a non-direct graph, if there is a path from node v to node v' , then v and v' are connected. If v_i and v_j are connected $\forall v_i, v_j \in V_G$, then G is a connected graph; otherwise, G is a disconnected graph.

Definition 6.17 If the two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ have the same topological structure, they are said to be isomorphic; that is, there exists a corresponding relationship from V_1 to V_2 such that each edge in E_1 corresponds to only one edge in E_2 , and vice-versa. Sub-graph isomorphic means that there exist sub-graphs of G_1 and G_2 that are isomorphic, which is to judge whether G_1 contains G_2 .

2. Dynamic fuzzy graph representation

Definition 6.18 Assume $G = (V_G, E_G)$ is a general graph and that $\sigma : V \rightarrow [0, 1]$, $\mu : E \rightarrow [0, 1]$ satisfy $\mu(e) \leq g(x) < g(y)$, where $e = xy$ [48]. Define $\underline{G} = (G, \sigma, \mu)$ as an F-graph. If G is a simple graph, then \underline{G} is a simple graph; if G is a direct graph, then \underline{G} is an F-direct graph.

Theorem 6.3 The sufficient and necessary condition for $\underline{G} = (G, \sigma, \mu)$ to be an F-graph is that, $\forall \lambda \in [0, 1]$, $(\sigma_\lambda, \mu_\lambda)$ is a sub-graph of G , where $\sigma_\lambda = \{x : \sigma(x) \geq \lambda\}$, $\mu_\lambda = \{e : \mu(e) \geq \lambda\}$.

Proof: If G is an F-graph, $\forall e = xy \in \mu_\lambda$, there is $\lambda \leq \mu(e) \leq \sigma(x) \wedge \sigma(y)$, so $x \in \mu_\lambda$, $y \in \mu_\lambda$, that is, $(\sigma_\lambda, \mu_\lambda)$ is a sub-graph of G .

Conversely, if $\forall \lambda \in [0, 1]$, $(\sigma_\lambda, \mu_\lambda)$ is a sub-graph of G , then $\forall e \in E$, $(\sigma_{\mu(e)}, \mu_{\mu(e)})$ is a sub-graph of G .

Because all $\forall e = xy \in \mu_{\mu(e)}$, then $\mu(e) \leq \sigma(x) \wedge \sigma(y)$, so \underline{G} is an F-graph. This completes the proof.

Based on the basic concept of F-graphs, with the help of DFL to make appropriate changes, the points and edges represented by the DF formula transform an F-figure into a graph containing a DFL formula.

For example, if the range field of some F-graph G is $U = \{v_1, v_2, v_3, v_4\}$, the F-matrix is

$$B = \begin{bmatrix} 0 & 0.3 & 0.2 & 0.8 \\ 0.3 & 0 & 0 & 0.5 \\ 0.2 & 0 & 0.3 & 0.7 \\ 0.8 & 0.5 & 0.7 & 0.7 \end{bmatrix}.$$

The node set of the F-graph is $V_G = \{0.9/v_1, 0.9/v_2, 0.6/v_3, 0.7/v_4\}$, and $G = (V, E)$ is shown in Fig. 6.8.

The DFL can be represented as $V_G = \{\overleftarrow{0.9}, \overrightarrow{0.9}\}/v_1, \{\overleftarrow{0.9}, \overrightarrow{0.9}\}/v_2, \{\overleftarrow{0.6}, \overrightarrow{0.6}\}/v_3, \{\overleftarrow{0.7}, \overrightarrow{0.7}\}/v_4\}$.

The above sample can be transformed into a direct graph, as shown in Fig. 6.9.

The DFL can be described as $V_G = \{\overleftarrow{0.9}, \overrightarrow{0.9}\}/v_1, \{\overleftarrow{0.9}, \overrightarrow{0.9}\}/v_2, \{\overleftarrow{0.6}, \overrightarrow{0.6}\}/v_3, \{\overleftarrow{0.7}, \overrightarrow{0.7}\}/v_4\}$

$E_G = \{\overrightarrow{0.2}/(v_1, v_3), \overrightarrow{0.3}/(v_1, v_2), \overrightarrow{0.8}/(v_1, v_4), \overrightarrow{0.7}/(v_3, v_4), \overrightarrow{0.5}/(v_2, v_4), \overrightarrow{0.3}/(v_3, v_3), \overrightarrow{0.3}/(v_1, v_2)\}$. In Fig. 6.9, there are seven circles, that is,

$$\{V_1V_2V_4V_1, V_1V_4V_3V_1, V_2V_4V_1V_2, V_3V_4V_1V_3, V_4V_1V_3V_4, V_3V_3, V_4V_1V_2V_4\}.$$

Taking $V_1V_2V_4V_1$ as an example, consider

$$\begin{aligned} & (\overleftarrow{0.9}, \overrightarrow{0.9})V_1 \wedge (\overleftarrow{0.9}, \overrightarrow{0.9})V_2 \wedge (\overleftarrow{0.7}, \overrightarrow{0.7})V_4 \\ & (\overleftarrow{0.9}, \overrightarrow{0.9})V_1 \vee (\overleftarrow{0.9}, \overrightarrow{0.9})V_2 \vee (\overleftarrow{0.7}, \overrightarrow{0.7})V_4 \\ & \overrightarrow{0.3}(V_1, V_2) \wedge \overrightarrow{0.5}(V_2, V_4) \wedge \overrightarrow{0.8}(V_4, V_1) \\ & \overrightarrow{0.3}(V_1, V_2) \vee \overrightarrow{0.5}(V_2, V_4) \vee \overrightarrow{0.8}(V_4, V_1). \end{aligned}$$

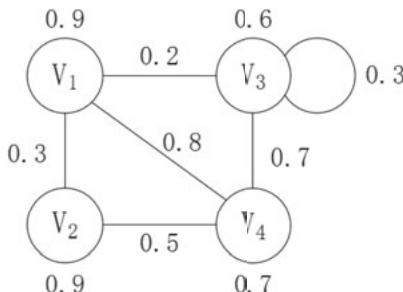


Fig. 6.8: Non-direct graph.

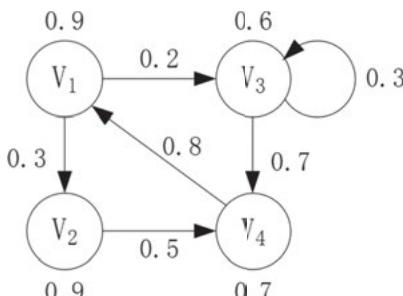


Fig. 6.9: Direct graph.

These situations will cause the results to change whenever an item in the formula changes.

More generally, in a direct circle, F-points and F-edges have the following DFL formulas:

$$(\overleftarrow{\lambda_1}, \overrightarrow{\lambda_1})V_1 \wedge (\overleftarrow{\lambda_2}, \overrightarrow{\lambda_2})V_2 \wedge \dots \wedge (\overleftarrow{\lambda_n}, \overrightarrow{\lambda_n})V_n \rightarrow (\overleftarrow{\lambda'_1}, \overrightarrow{\lambda'_1})V'_1 \wedge (\overleftarrow{\lambda'_2}, \overrightarrow{\lambda'_2})V'_2 \wedge \dots \wedge (\overleftarrow{\lambda'_n}, \overrightarrow{\lambda'_n})V'_n$$

$$(\overleftarrow{\lambda_1}, \overrightarrow{\lambda_1})V_1 \vee (\overleftarrow{\lambda_2}, \overrightarrow{\lambda_2})V_2 \vee \dots \vee (\overleftarrow{\lambda_n}, \overrightarrow{\lambda_n})V_n \rightarrow (\overleftarrow{\lambda'_1}, \overrightarrow{\lambda'_1})V'_1 \vee (\overleftarrow{\lambda'_2}, \overrightarrow{\lambda'_2})V'_2 \vee \dots \vee (\overleftarrow{\lambda'_n}, \overrightarrow{\lambda'_n})V'_n$$

$$\begin{aligned} & (\overleftarrow{\mu_1}, \overrightarrow{\mu_1})(V_1, V_1) \wedge (\overleftarrow{\mu_2}, \overrightarrow{\mu_2})(V_2, V_2) \wedge \dots \wedge (\overleftarrow{\mu_n}, \overrightarrow{\mu_n})(V_n, V_n) \\ & \rightarrow (\overleftarrow{\mu'_1}, \overrightarrow{\mu'_1})(V'_1, V'_1) \wedge (\overleftarrow{\mu'_2}, \overrightarrow{\mu'_2})(V'_2, V'_2) \wedge \dots \wedge (\overleftarrow{\mu'_n}, \overrightarrow{\mu'_n})(V'_n, V'_n) \end{aligned}$$

$$\begin{aligned} & (\overleftarrow{\mu_1}, \overrightarrow{\mu_1})(V_1, V_1) \vee (\overleftarrow{\mu_2}, \overrightarrow{\mu_2})(V_2, V_2) \vee \dots \vee (\overleftarrow{\mu_n}, \overrightarrow{\mu_n})(V_n, V_n) \\ & \rightarrow (\overleftarrow{\mu'_1}, \overrightarrow{\mu'_1})(V'_1, V'_1) \vee (\overleftarrow{\mu'_2}, \overrightarrow{\mu'_2})(V'_2, V'_2) \vee \dots \vee (\overleftarrow{\mu'_n}, \overrightarrow{\mu'_n})(V'_n, V'_n) \end{aligned}$$

These DFL formulas cause the results to change only if any item changes, a process that can be considered to have a learning action.

6.5.2 Dynamic fuzzy graph hierarchical relationship learning algorithm

A dynamic fuzzy directed acyclic graph (hereafter referred to as DF) can represent the relationships between categories according to levels, namely, the dynamic fuzzy dependency relation among the characters. This has the following meaning:

- (1) Each vertex represents a class, and the corresponding categories are not the same;
- (2) Each directed edge represents dynamic fuzzy dependencies between classes, with arrows pointing toward the subclass and away from the parent class;
- (3) There is only one vertex with an in-degree of 0, which represents the total class of all data at the highest level; a vertex with an out-degree of 0 denotes that the current category cannot be classified. See Fig. 6.10.

Theorem 6.4 There must be a topological order of nodes in the DF graph.

The topological order is the order of the vertices in the DF graph; i.e. the starting point of each directed edge in the DF graph is located at the front of the vertex pointed to by the edge in the ranking. Kahn developed an algorithm of complexity $O(|V| + |E|)$ to find the topological order of a non-directed graph $G = (V, E)$ [57]. As shown in Fig. 6.10, the application of this algorithm gives the topological order $a \ b \ c \ d \ e \ f \ g \ h$.

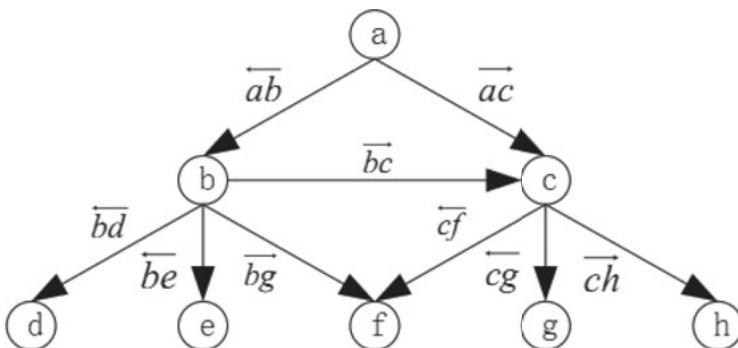


Fig. 6.10: Dynamic fuzzy directed acyclic graph that represents hierarchical structure.

The dynamic fuzzy graph is used to deal with uncertain information and the dependency relationship, which is more expressive and has good learning ability. The dynamic fuzzy membership function can be automatically generated and adjusted so as to be applied to complex processes and systems.

The dynamic fuzzy graph hierarchical relationship learning algorithm (DFGHR) is described in Algorithm 6.2. Here, x is an attribute feature, Y represents the class set, and c_i represents the class corresponding to x . The training process considers the vertex v_i (except the general category whose in-degree is 0) in the DF graph. Generate a two-element classifier $H_{c_i} : X \times Y \rightarrow R$, where X represents an attribute feature, Y represents the class set, and R represents the real numbers. $H_{c_i}(x, y) \in [0, 1] \times [-, +]$ represents the degree to which x belongs to y , that is, the membership rate. $|H_{c_i}(x, y)|$ represents the confirmation of the judgment.

In the topological order of the DF graph, nodes corresponding to the parent class must be in front of nodes corresponding to sub-classes. Step 4 in the following DFGHR algorithm ensures the high-level classification occurs before the low-level classification. In two-element classification, the judgment that the attribute belongs to a class uses the DF membership rate $(\overleftarrow{\theta}, \overrightarrow{\theta})$ (Step 6 in the following DFGHR algorithm). When a classifier return determines that some feature attribute does not belong to the category, according to the idea of hierarchical classification, the feature attribute does not belong to a parent class, and it also does not belong to the subclass. In this case, we remove the transfer path. That is, when a category node in-degree is reduced to 0, the feature attribute does not belong to all parent classes, and the node should be removed from the topological order in T . The class corresponding to a vertex in T that has not been removed is the class that the feature attribute is waiting to classify (step (7)).

The time required to find the topological order in a DF graph is $O(|V| + |E|)$, and the time complexity of the dynamic fuzzy graph hierarchical relationship learning algorithm is $\max \{O(|V| + |E|), O(|V|T(H_{c_i}))\}$, where $T(H_{c_i})$ represents the time cost of H_{c_i} .

The original topological order of the vertex in Fig. 6.10 is $a b c d e f g h$. Through dynamic adjustment, we evaluate $H_b(x) < (\overleftarrow{\theta}, \overrightarrow{\theta})$; if so, then $R^-(d) = R^-(e) = 0$, $R^-(f) = 1$, $R^-(f) = 1$. Therefore, the topological order becomes $a c f g h$.

Algorithm 6.2 DFGHR

- (1) The number of initial feature classes and the number of final classes determine the vertexes; initialize the DF figure as input;
 - (2) Find a topological order T in the DF graph;
 - (3) Calculate the in-degree $R^-(v_i)$ of vertex v_i ;
 - (4) According to T , deal with the vertexes v_i in the DF graph in turn;
 - (5) Call the classifier H_{c_i} of vertex v_i (the corresponding class c_i) to classify x , then obtain $H_{c_i}(x)$;
 - (6) Compare $H_{c_i}(x)$ and $(\overleftarrow{\theta}, \overrightarrow{\theta})$ to determine whether to remove v_i , and adjust the value of $R^-(v_i)$ accordingly; when $R^-(v_i) = 0$, remove the corresponding vertex from T , and adjust the in-degree of the adjacent vertex. Repeat until the in-degrees of all vertexes behind that vertex in T are not 0.
 - (7) Output the class c corresponding to the saved vertex in T , the DF membership rate, obtain the level division set Y of each attribute feature, and determine the hierarchical structure in the DF graph.
-

6.5.3 Sample analysis

An example analysis using the Weka dataset (see Tab. 6.10) was conducted to compare the J48 algorithm with DFGHR. After processing the original data in the test sample set and the combined training sample/test sample set, the constructed DF diagram was tested by 10-fold cross-validation. The vertex numbers of the corresponding figure obtained by the J48 and DFGHR algorithms are presented in Tabs. 6.11 and 6.12.

The classification accuracy of J48 and DFGHR was compared after 10-fold cross-validation. The dynamic fuzzy graph hierarchical relation learning algorithm uses the dynamic fuzzy pretreatment method to deal with data, which fully takes the uncertainty in the real problem into account. The algorithm constructs a hierarchical relationship, which considers the contribution of each layer's attribute features to the classification of instances and further improves the classification accuracy of the training example set. According to the experimental results, the classification accuracy of DFGHR indicates improved predictive ability over the J48 algorithm. For the datasets with missing information, its superiority is more obvious.

6.6 Sample application and analysis

As a kind of mathematical model, graphs are an active field of study in relational data mining and have a wide range of applications, including protein analysis, compound

Tab. 6.10: Date set information owned by weka.

Dataset	Number of instances	Number of attributes	Number of category	Value missing
Weather	14	5	2	Yes
Contact-lenses	24	5	3	No
Labor-neg-data	57	17	2	Yes
Soybean	683	36	19	Yes
Segment	1500	20	7	No

Tab. 6.11: J48 algorithm experiment result.

Dataset	Number of vertices	10 Cross-validations		Using training set	
		CCI	DOA	CCI	DOA
Weather*	8	9	64.29%	14	100%
Contact-lenses	7	20	83.33%	22	91.67%
Labor-neg-data*	5	42	73.68%	50	87.72%
Soybean*	93	625	91.51%	658	96.34%
Segment	67	1436	95.73%	1485	99%

*Note:** denotes a dataset with missing values, and CCI/DOA denote classified correctly (instances) and degree of accuracy, respectively.

Tab. 6.12: DFGHR algorithm experiment result.

Dataset	Number of vertices	10 Cross-validations		Using training set	
		CCI	DOA	CCI	DOA
Weather*	7	10	78.57%	14	100%
Contact-lenses	7	20	83.33%	22	91.67%
Labor-neg-data*	4	44	77.19%	53	92.98%
Soybean*	87	626	91.65%	679	99.41%
Segment	61	1454	96.53%	1500	100%

*Note:** denotes a dataset with missing values, and CCI/DOA denote classified correctly (instances) and degree of accuracy, respectively.

analysis, link mining, XML document analysis, and the detection of security threats [57]. This section mainly verifies the algorithms presented in this chapter.

6.6.1 Question description

First, we consider the verification of chemical molecular structure identification results. Data mining and knowledge representation of chemical molecular information is an important field of life sciences. By analysing the chemical molecular

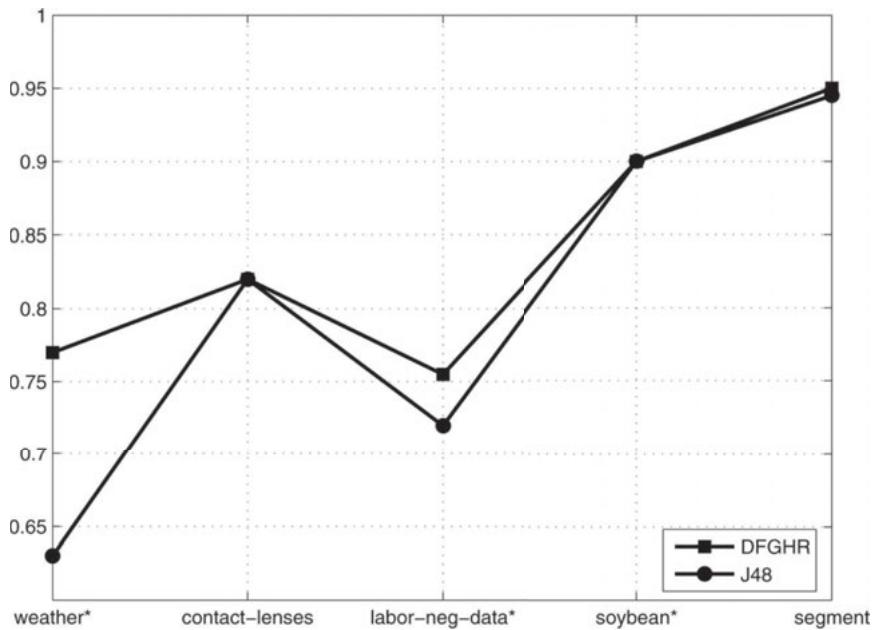


Fig. 6.11: J48 and DFGHR 10-fold cross validation classification accuracy comparison chart.

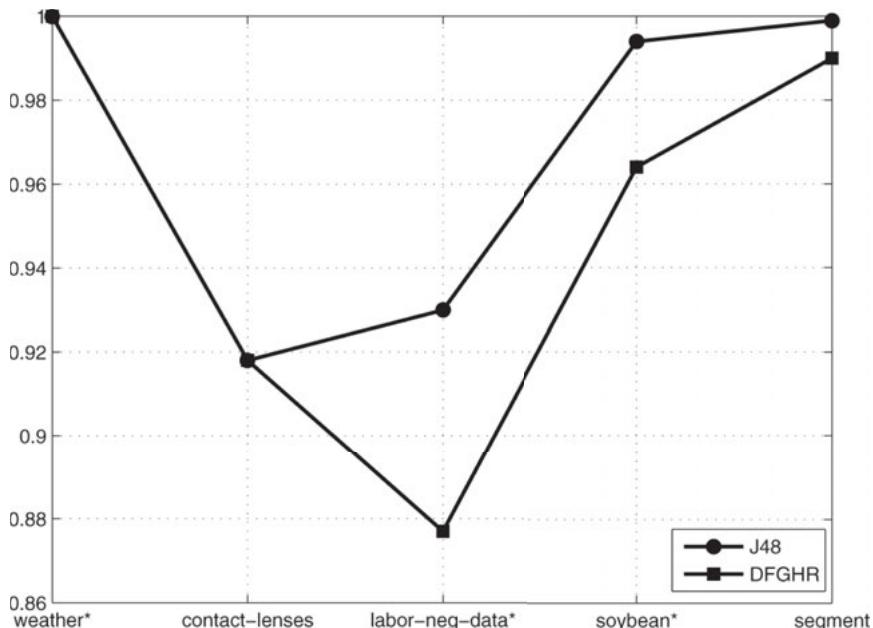


Fig. 6.12: Classification accuracy rate of J48 and DFGHR based on training set.

structure, we can abstract a given substance to an undirected graph, where the vertices in the graph denote the atoms and the relations between the vertices represent the chemical bonds between the atoms. A double sparse relational learning algorithm based on multiple kernel learning [2] has been used to test chemical information datasets, and experimental results show that the prediction accuracy is obviously improved, with fewer rule sets and more direct explanations. Zhao XF et al. [58] designed the OpenMolGRID system, which is used to mine chemical information in the grid environment for the design and construction of molecules. Qi et al. [59] proposed a two-layer classifier to identify the *Escherichia coli* promoter in DNA. The first-level classifier uses three Bayesian neural networks to learn three different feature sets, and the second layer is combined with the output of the first layer to give the final results. The method of Borgelt et al. [60] embeds the generated molecular fragments in parallel into all suitable molecular structures and prunes the search tree to help distinguish between different classes based on local atomic and bond sequences. Machine learning methods help chemists in two ways: finding frequently occurring chemical molecular fragments and finding fragments in the database that are frequent and infrequent in other parts of the database, which allows the database to be divided into active molecules and non-active molecules, as shown in the following example.

For the compounds shown in Fig. 6.14, two are active and two are inactive. The task of learning is to find a model that can distinguish active from non-active molecules. This type of learning task is important in computational chemistry, where it is usually called structure activity relationship prediction [1] and can be used for the design and development of drugs and toxicity detection.

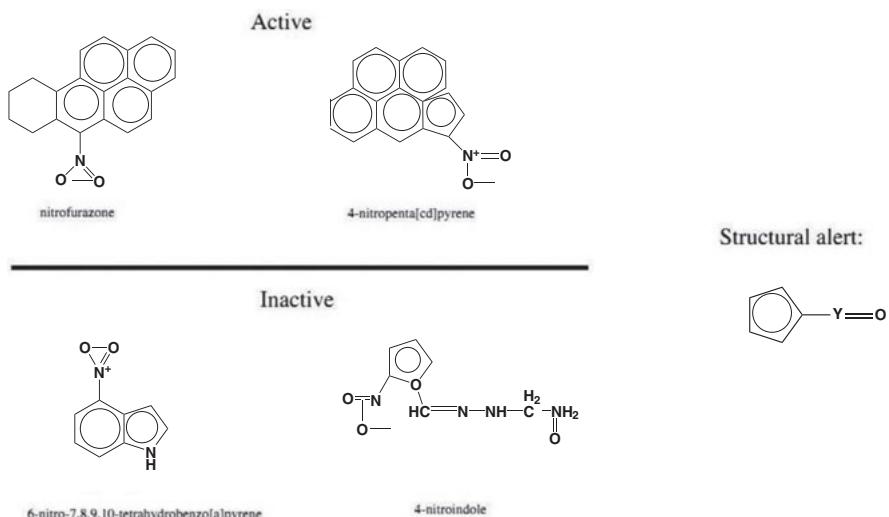


Fig. 6.13: Prediction of mutagenicity of compound.

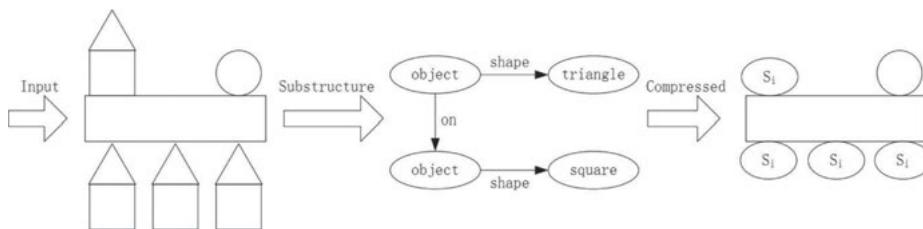


Fig. 6.14: Substructure discovery and compression of graphs.

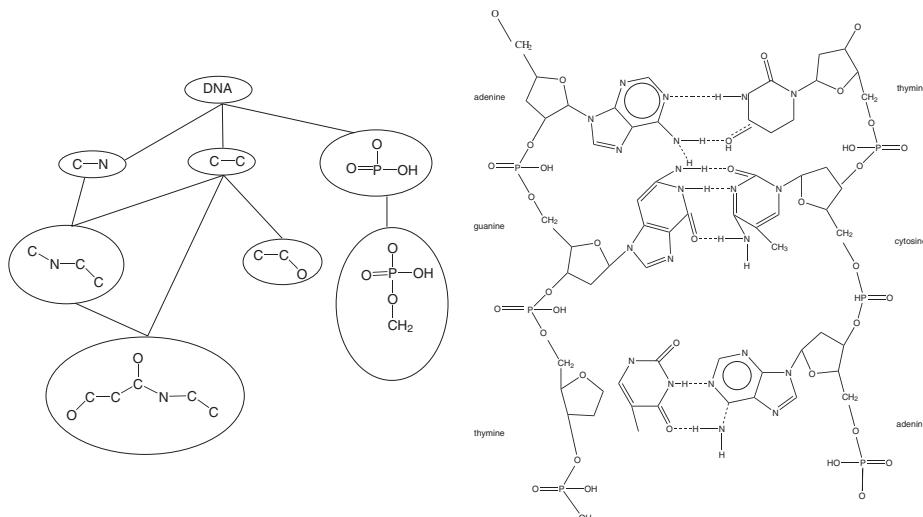


Fig. 6.15: A part of DNA Subdue clustering.

The pattern found in Fig. 6.13 is called a structural alarm and is used to separate active molecules from inactive molecules. Because the structure warning is a sub-structure (sub-graph), it is compatible with the active molecule, and there is no match with the non-active molecules. At the same time, the structural alarm is easy to understand and provides useful insights into the factors that determine molecular activity.

The DF map constructed by DFGHR is compressed using the Subdue method [57]. Because it can find common sub-graph patterns in the large image, we iterate the model to compress the original image until it can no longer be compressed. This produces a hierarchical conceptual clustering of the input data. In the i th iteration, the best sub-graph is used to compress the input graph and introduce the new node into the graph for the next iteration, as shown in Fig. 6.14.

Figure 6.16 hierarchy discovery process can be compressed as shown in Fig. 6.15.

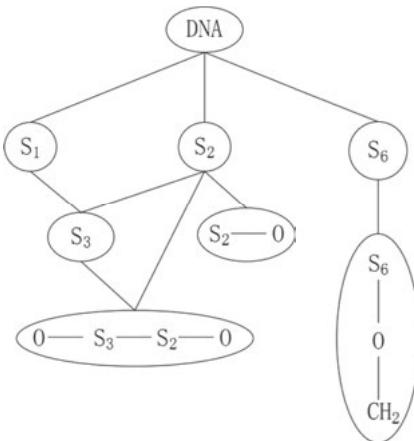


Fig. 6.16: Compressed representation of 6.15.

6.6.2 Sample analysis

We combined the DFGHR and DFLR algorithms to construct a DF graph and extract rules from the “promoter” dataset in the UCI knowledge base (<http://www.ics.uci.edu/~mlearn/MLRepository.html>). The input properties are 57 ordered DNA nucleotides (A, G, T, C), with a total sample number of 106 [53 positive cases (promoter sequence samples), 53 negative examples (non-promoter sequence samples)]. Each nucleotide sequence is aligned with a reference point, so that the n th attribute corresponds to the n th nucleotide. If the data are not aligned, standard classifiers such as C4.5 cannot handle the problem. In the form of a graph, the data need not be aligned with the reference points.

In this section, each sequence is transformed into a representation of a graph, as shown in Fig. 6.17. Each element is assumed to interact with at most 10 elements. Therefore, each sequence obtains 57 nodes and 515 connections.

After using the DFGHR algorithm to produce a sub-graph model of the promoter dataset, the DFLR algorithm was used to obtain the following classification rules:

```

IF T?T??????T?A = y THEN+
IF C??????T??T?A = y THEN+
IF ATTT = y THEN + IF C?AA = y THEN+
IF T?A???G?A = y THEN+
IF A?AT?A = y THEN+
IF T?T??????T?A = n ∧ A?T??T??????C = n
    ∧ ATTT = n ∧ C?AA = n ∧ A?AT?A = n ∧ T?A???G?A = n THEN -
  
```

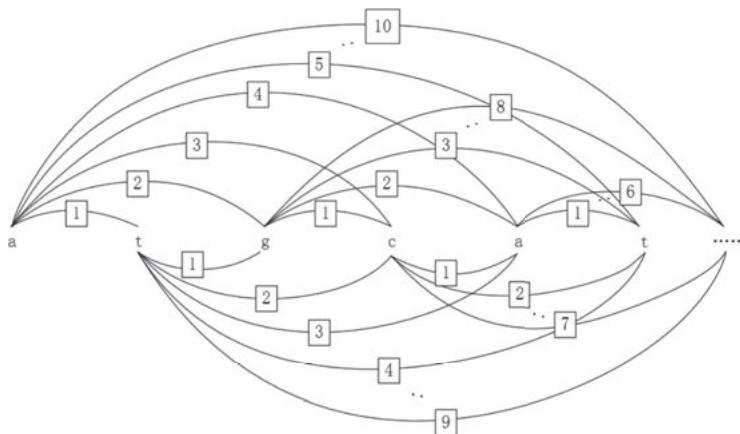


Fig. 6.17: DNA sequence data transformation diagram.

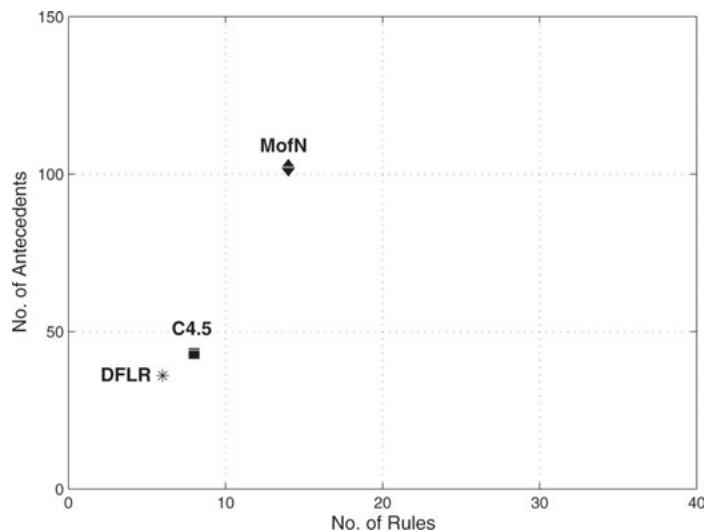


Fig. 6.18: Comparison of the rule size extracted by 3 algorithms.

where y represents the abstracted sub-structure, n represents the number of models abstracted, + represents a promoter, and - represents a non-promoter.

The resulting rule size is compared with those of the C4.5 and MofN algorithms in Fig. 6.18.

Intelligibility reflects the extent to which the rule is understood by the user. Any increase in the rule number and the antecedent number of each rule will increase the difficulty of understanding the rules. From the above graph, we can see that the rules obtained by DFLR are simpler to understand than those obtained by C4.5 and MofN.

It is also necessary to consider whether the rules have significance. Of the six rules obtained by DFLR, the sixth rule covers counterexamples. This has too many complex antecedents, and can be omitted. That is, the samples covered by the first five rules are positive, otherwise they are negative.

6.7 Summary

Relationship learning is a learning paradigm in the field of machine learning that considers knowledge representation to obtain specific information in relational data, and then uses the knowledge obtained to reason, predict, and classify.

It is difficult to deal with uncertain information, such as the bias in data caused by the subjectivity of an expert system, or the absence or fuzziness of data. Dynamic fuzzy theory can effectively express this kind of data.

This chapter has introduced the theory of DFD into relational learning. The main results include the following:

- (1) Based on DFL and a dynamic fuzzy matrix, we proposed a DFL learning algorithm and dynamic fuzzy relationship matrix hierarchy learning algorithm, and verified the effectiveness of the algorithms.
- (2) On the basis of dynamic fuzzy sets and the dynamic fuzzy production, combined with the C4.5 algorithm and a decision tree, a dynamic fuzzy tree hierarchical relationship learning algorithm was proposed, and samples combined with real data were analysed.
- (3) Based on dynamic fuzzy graph theory, we proposed a dynamic fuzzy direct acyclic graph hierarchy analysis, and presented a dynamic fuzzy HRL algorithm that was compared with the J48 algorithm.
- (4) By combining the DFGHR and DFLR algorithms, a dataset of chemical molecules with missing data was constructed and its rules extracted. Compared with the conventional method, our algorithm is more general.

References

- [1] Raedt LD. Logical and relational learning. New York, USA, Springer-Verlag Inc., 2008.
- [2] Han YJ, Wang YYJ. A Bi-sparse relational learning algorithm based on multiple kernel learning. *Journal of Computer Research and Development*, 2010, 47(8): 1400–1406.
- [3] Sato T, Kameya Y. PRISM: A symbolic-statistical modeling language. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*. San Francisco, Morgan Kaufmann, 1997, 1330–1339.
- [4] Muggleton S. Stochastic logic programs. In *Proceedings of the 5th International Workshop on Inductive Logic Programming*. Amsterdam, USA, IOS Press, 1996, 254–264.
- [5] Kramer S. Structural regression trees. In *Proceedings of the 13th National Conference on Artificial Intelligence*. Menlo Park, CA, AAAI Press, 1996, 812–819.

- [6] Blaauw B, Raedt L De. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 1998, 101(1/2): 285–297.
- [7] Knobbe A J, Siebes A, Der Wallen D Van. Multi-relational decision tree induction. In *Proceedings of the 3rd European Conference on Principles and Practice of KDD*, Berlin, Germany, Springer, 1999, 378–383.
- [8] Ngo L, Haddawy P. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 1997, 171(1/2): 147–177.
- [9] Kersting K, Raedt L De. Adaptive Bayesian logic programs. In *Proceedings of the 11th Conference on Inductive Logic Programming*. Berlin, Germany, Springer, 2001, 104–117.
- [10] Muggleton S. Learning stochastic logic programs. In *Proceedings of the AAAI2000 Workshop on Learning Statistical Models from Relational Data*. Menlo Park, CA, AAAI Press, 2000, 19–26.
- [11] Cussens J. Parameter estimation in stochastic logic programs. *Machine Learning*, 2001, 44(3): 245–271.
- [12] Muggleton S. Learning structure and parameters of stochastic logic programs //LNCS 2583: *Proceedings of the 12th International Conference on Inductive Logic Programming*. Berlin, Germany, Springer, 2002, 198–206.
- [13] Neville J, Jensen D, Friedland L, et al. Learning relational probability trees. In *Proceedings of the 9th International Conference on Knowledge Discovery & Data Mining*. New York, USA, ACM, 2003, 9–14.
- [14] Bernstein A, Clearwater S, Provost F. The relational vector-space model and industry classification. In *Proceedings of IJCAI2003, Workshop on Learning Statistical Models from Relational Data*. San Francisco, USA, Morgan Kaufmann, 2003, 8–18.
- [15] Angelopoulos N, Cussens J. On the implementation of MCMC proposals over stochastic logic programs. In *Proceedings of Colloquium on Implementation of Constraint and Logic Programming Systems. Satellite Workshop to ICLP'04*. Berlin, Germany, Springer, 2004.
- [16] Taskar B, Abbeel P, Koller D. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*. San Francisco, USA, Morgan Kaufmann, 2002, 485–492.
- [17] Richardson M, Domingos P. Markov logic networks. Seattle, WA, Department of Computer Science and Engineering, University of Washington, 2004.
- [18] Anderson C, Domingos P, Weld DS. Relational Markov models and their application to adaptive Web navigation. In *Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining (KDD-02)*. New York, ACM, 2002, 143–152.
- [19] Kersting K, Raiko T, Kramer S, et al. Towards discovering structural signatures of protein folds based on logical hidden Markov models. In *Proceedings of the 8th Pacific Symposium on Biocomputing*, 2003, 192–203.
- [20] Davis J, Burnside E, Dutra IC, et al. An integrated approach to learning Bayesian networks of rules. *Proceedings of the 16th European Conference on Machine Learning*. Berlin, Germany, Springer, 2005, 84–95.
- [21] Landwehr N, Kersting K, Raedt L De. nFOIL: Integrating naive Bayes and FOIL. In *Proceedings of AAAI*. Berlin, Germany, Springer, 2005, 795–800.
- [22] Sato T. A generic approach to EM learning for symbolic-statistical models. In *Proceedings of ICML05 Workshop on Learning Language in Logic*. New York, USA, ACM, 2005, 1130–1200.
- [23] Izumi Y, Kameya Y, Sato T. Parallel EM learning for symbolic-statistical models. In *Proceedings of the International Workshop on Data-Mining and Statistical Science*. Sapporo, Japan, 2006, 133–140.
- [24] Xu Z, Tresp V, Yu K, et al. Infinite hidden relational models. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*. Arlington, Virginia, AUAI Press, 2006.

- [25] Singla P, Domingos P. Entity resolution with Markov logic. In Proceedings of the 6th International Conference on Data Mining. Los Alamitos, CA, IEEE Computer Society, 2006, 572–582.
- [26] Kersting K, Raedt LD, Raiko T. Logical hidden Markov models. *Journal of Artificial Intelligence Research*, 2006, 25: 425–456.
- [27] Chen J, Muggleton SH. A revised comparison of Bayesian logic programs and stochastic logic programs. In Proceedings of the 16th International Conference on Inductive Logic Programming. Berlin, Germany, Springer, 2006.
- [28] Singla P, Domingos P. Markov logic in infinite domains. In Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence. Arlington, VA, AUAI Press, 2007, 368–375.
- [29] Wang J, Domingos P. Hybrid Markov logic networks. In Proceedings of the 23rd AAAI Conference on Artificial Intelligence. Menlo Park, CA, AAAI Press, 2008, 1106–1111.
- [30] Petr B, Jiri K, Katsumi I. Grammatical concept representation for randomised optimization algorithms in relational learning. In Proceedings of the 9th International Conference on Intelligent Systems Design and Applications, 2009, 156, 1450–1455.
- [31] Zhu M. DC Proposal: Ontology learning from noisy linked data. USA, Springer-Verlag, 2011, 2(4): 373–380.
- [32] Bengio Y. Learning Deep Architectures for AI[J]. *Foundations & Trends® in Machine Learning*, 2009, 2(1): 1–127.
- [33] Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. *Neural Computation*, 2014, 18(7): 1527–1554.
- [34] Bengio Y, Lamblin P, Dan P, et al. Greedy layer-wise training of deep networks[C]. International Conference on Neural Information Processing Systems. MIT Press, 2006:153–160.
Bengio Y, Lamblin P, Popovici D, Larochelle H. Greedy layer-wise training of deep networks. In Scholkopf B, Platt J, Hoffman T (Eds.). In *Advances in neural information processing systems* 19, 2006, 153–160.
- [35] Erhan D, Manzagol PA, Bengio Y, Bengio S, Vincent P. The difficulty of training deep architectures and the effect of unsupervised pre-training. In Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, 2009, 153–160.
- [36] Hugo L, Yoshua B, Jerme L, et al. Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, 2009, 10: 1–40.
- [37] Ranzato M, Boureau YL, LeCun Y. Sparse feature learning for deep belief networks. In Platt J, Koller D, Singer Y, & Roweis S (Eds.), Cambridge, MA, In *Advances in Neural Information Processing System*, 2007, 1185–1192.
- [38] Salakhutdinov R, Hinton GE. Deep Boltzmann machines. *Journal of Machine Learning Research*, 2009, 5(2): 1967–2006.
- [39] Jason W, Frederic R, Ronan C. Deep learning via semi-supervised embedding. In Proceedings of the 25th International Conference on Machine Learning, 2008, 307, 1168–1175.
- [40] Ranzato, Marc A, Huang, et al. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In Proceedings of the Computer Vision and Pattern Recognition Conference, 2007.
- [41] Lee H, Grosse R, Ranganath R, et al. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In Proceedings of the 26th International Conference on Machine Learning, 2009.
- [42] Sun M, Chen B, Zhou MT. Domain relation learning from events in log ontologies based on hierarchy association rules. *Application Research of Computers*, 2009, 26(10): 3683–3686.
- [43] Cook D, Holder L. Graph-based data mining. *IEEE Intelligent Systems*, 2000, 15(2): 32–41.
- [44] Yoshida K, Motoda H, Indurkhya N. Graph-based induction as a unified learning framework. *Journal of Applied Intelligence*, 1994, 4: 297–328.

- [45] Li FZ, et al. Dynamic fuzzy set and application. Kunming, China, Yunnan Science & Technology Press, 1997.
- [46] Li FZ, Zhu WH. Dynamic fuzzy logic and Application. Kunming, China, Yunnan Science & Technology Press, 1997.
- [47] Li FZ, Liu GQ, She YM. An introduction to dynamic fuzzy logic. Kunming, China, Yunnan Science & Technology Press, 2005.
- [48] Li FZ. Dynamic fuzzy logic and its application. America, Nova Science Publishers, 2008.
- [49] Li FZ. Research on a dynamic fuzzy relation data model. *Journal of Chinese Mini-Micro Computer Systems*, 2002, 23(9): 1107–1109.
- [50] Zhang J. Dynamic fuzzy machine learning model and its applications. Master's Thesis, Soochow University, Suzhou, China, 2007.
- [51] Mitchell, TM. Machine learning. NY, USA, McGraw-Hill, 1997.
- [52] Henri P, Gilles R, Mathieu S. Enriching relational learning with fuzzy predicates. In *Proceedings of PKDD*, 2003.
- [53] Sheng Z, Xie SQ, Pan CY. Probability theory and mathematical statistics. Beijing, China, Higher Education Press, 2001: 176–207.
- [54] Delgado M, Sanchez D, and Vila MA. Fuzzy cardinality based evaluation of quantified sentences. *International Journal of Approximate Reasoning*, 2000, 23(1): 23–66.
- [55] Frederic S, David M. Using kernel basis with relevance vector machine for feature selection. In *Proceedings of ICANN*, 2009, 5769: 255–264.
- [56] Kahn AB. Topological sorting of large networks. *Communications of the ACM*, 1962, 5(11): 558–562.
- [57] Maran U, Sild S, Kahn I, Takkis K. Mining of the chemical information in GRID environment. *Future Generation Computer Systems*, 2007, 23: 76–83.
- [58] Qicheng MA, Jason TL. Wang. Biological data mining using bayesian neural networks: a case study[J]. *International Journal on Artificial Intelligence Tools*, 1999, 8(04).
- [59] Borgelt C, Berthold MR. Mining molecular fragments: Finding relevant substructures of molecules. In *Proceeding of the 2002 international conference on data mining*, 2002, 2: 211–218.
- [60] Tammy RR, Nir S, Nahum K. On symmetry, perspectivity, and level set based segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 2009, 31(8): 1458–1471.
- [61] Holder L, Cook DJ, Coble J, and Mukherjee M. Graph-based relational learning with application to security. *Fundamental Informaticae Special Issue on Mining Graphs, Trees and Sequences*, 2005, 66(1–2): 83–101.
- [62] Hsu W, Joehanes R. Relational decision networks. In *Proceedings of the ICML-2004 Workshop on Statistical Relational Learning and Connections to Other Fields*, 2004, 61–67.
- [63] Li FF, Rob F, Antonio T. Recognizing and Learning Object Categories. In *Proceedings of ICCV*, 2009.
- [64] David GL. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004, 60(2), 91–11.
- [65] Ontanon S, Plaza E. Similarity measures over refinement graphs. *Machine Learning*, 2012, 87: 57–92.
- [66] Rettinger A, Nickles M, Tresp V. Statistical relational learning of trust. *Machine Learning*, 2011, 82: 191–209.
- [67] Mustafa D. The order-theoretic duality and relations between partial metrics and local equalities. *Fuzzy Sets and Systems*, 2012, 192: 5–57.
- [68] Ignjatović J, et al. Weakly linear systems of fuzzy relation inequalities: The heterogeneous case. *Fuzzy Sets and Systems*, 2011, 199(24): 64–91.

7 Multi-agent learning model based on dynamic fuzzy logic

This chapter is divided into five sections. Section 7.1 gives a brief introduction to this chapter. We present the agent mental model based on DFL in Section 7.2 and the single agent learning algorithm based on DFL in Section 7.3. In Section 7.4, we introduce algorithm of multi-agent learning model based on DFL. The last section is the summary of the chapter.

7.1 Introduction

By using a learning method, the knowledge and ability of an agent will be enhanced. This is known as agent learning. In other words, the knowledge base of the agent will be complemented, and the ability to execute will be improved or have a better fit with the interests and habits of the users.

7.1.1 Strategic classification of the agent learning method

Recently, many agent learning strategies have been developed, such as Rote Learning, Learning from Instructions and by Advice Taking, Learning from Examples, Inductive Learning, Learning by Analogy, Learning by Observation and Discovery, Case-Based Learning, Explanation-Based Learning, Completing Learning, Cooperating Learning, Game-theoretic Learning, Adaptive Learning (through the adaptive adjustment of all kinds of objects to stabilize a target neural network, which is then combined with a Genetic Algorithm), and Reinforcement Learning.

7.1.2 Characteristics of agent learning

The agent is an intellectual body that has the capacity for self-learning, as well as the following characteristics:

Self-control: during the learning procedure, the agent does not need external guidance and has the ability to control its own learning actions.

Reactivity: during the learning procedure, the agent will respond to changes in the outside world.

Inferential: according to its own current knowledge, which means the current information included in some repository, the agent will implement a continuing rational inquiry and update the repository.

Learnability: from unknown to known, the agent learns from everything around, much like a conscious human. After obtaining the learning results, the agent will store them in the repository. Thus, the agent can directly recall knowledge from the repository when faced with a similar situation.

According to these characteristics, the agent techniques have been applied in multiple areas, e.g. game development [1-2], mail routing [3], spoken dialogue [4], and robot soccer [5].

7.1.3 Related work

A typical case of single-agent learning is OpenML, which is a software development tool published by Intel Corporation in 2004. This tool enables systems built by software engineers to “learn” from the application. The key to this learning process is to use previous data to improve the accuracy and durability of the system. Moreover, this tool can predict the probability of events that have happened. OpenML is based on Bayes’ theorem. Its core philosophy is to study the frequency of an event that has taken place and predict the probability of this event happening in the future. An interesting application of OpenML is when Intel researchers used it to create an audio/video speech recognition system. This recognition system employs video cameras to detect a speaker’s facial expressions and mouth movements. This “read lips” method helps to improve the accuracy of speech recognition in noisy environments such as airports or supermarkets [6].

The ADE project at the University of Southern California is an adaptive learning system geared for education. The core of this project is to solve problems. In Stanford University, an adaptive system called “Fab” provides a web page recommendation service. All of these applications are representations of agent learning technology, collectively referred to as the interface agent technique.

A typical case of a learning agent has been developed by Chen and Sycara. Their WebMate can automatically extract resources online according to the user’s interests and can satisfy demands for the retrieval of related domain knowledge. Yang and Ning conducted a deep analysis of reinforcement learning. They proposed an ensemble method based on AODE that employs the BDI model and reinforcement learning to guide the learning process of the agent. Moreover, they studied the deviation of motivated learning. Using planning rules, they were able to overcome the shortcomings of reinforcement learning to some extent and improve the learning efficiency. The machine learning research group led by Professor Fanzhang at Soochow University has conducted a lot of theoretical work in the field of agent learning. For example, they have studied an agent self-learning model based on DFL [7] and an automated reasoning platform [8].

The ILS system of GTE [9] integrates heterogeneous agents through a centralized controller. In this way, the agents are able to judge one another. The MALE system

enables different agents to cooperate on an interactive board, similar to a blackboard. Guorui proposed an emotional body system that employs the features of the genetic system, nervous system, and endocrine system in its construction. In this system, the researchers proposed a novel action learning method using emotions. The machine learning research group led by Professor Li has also reported great achievements in this area, such as a multi-agent problem-solving system based on DFL and a combinatorial mathematics learning system.

7.2 Agent mental model based on DFL

The agent mental model is the foundation of agent learning, and an efficient and reasonable mental model is needed to construct the learning system [10]. In previous work, the agent system has been studied as a mental system [11–13], and many scholars believe that is reasonable and useful. Hu and Shi [14] described a representative system called A-BDI with Agent-BDI logic. In this section, we will construct the model structure and axiomatic system of the agent mental model based on DFL.

7.2.1 Model structure

Definition 7.1 The agent mental model based on DFL consists of an 11-tuple $(B, C, I, A, S, Sty, R, Brf, Acf, Irf, Stf)$, where B (Belief) represents the understanding of information and basic views of the environmental world within the range of cognitive ability of the agent model. $B(P_i, (\overleftarrow{x_i}, \overrightarrow{x_i}))$ represents the agent belief of event $(\overleftarrow{x_i}, \overrightarrow{x_i})$ at some point. B is the set of current beliefs, and $(\overleftarrow{B}, \overrightarrow{B}) = \{B(P_1, (\overleftarrow{x_1}, \overrightarrow{x_1})), B(P_2, (\overleftarrow{x_2}, \overrightarrow{x_2})), \dots, B(P_m, (\overleftarrow{x_m}, \overrightarrow{x_m}))\}$ form current events.

From the definition of Belief, we know that an event is actually a pseudo-proposition; therefore, the implication of $B(P_i, (\overleftarrow{x_i}, \overrightarrow{x_i}))$ is the extent to which the agent believes that proposition P_i is true, which we represent using $(\overleftarrow{x_i}, \overrightarrow{x_i})$. The structure and updating algorithm of our belief database are shown in Tab. 7.1 and Algorithm 7.1. In this model structure, we will update the belief database when the agent conducts operations relevant to the belief database as follows: first, contrast

Tab. 7.1: Structure of belief database.

Proposition	Event
P_1	$(\overleftarrow{x_1}, \overrightarrow{x_1})$
P_2	$(\overleftarrow{x_2}, \overrightarrow{x_2})$
.....
P_m	$(\overleftarrow{x_m}, \overrightarrow{x_m})$

the belief of the current learning process to query whether there is any information about the belief. If the belief already exists and its value is invariant, do nothing. If the value of the belief has changed, make the corresponding modifications. If the belief does not already exist, we will create and insert it at the appropriate location.

C (Capability) represents prerequisites for performing a task, i.e. the capacity of the current agent. $C(M_i, (\overleftarrow{x}_i, \overrightarrow{x}_i))$ represents the agent's capacity to process problem M_i at time $(\overleftarrow{x}_i, \overrightarrow{x}_i)$. C is the set of capacities after the learning process, and $(\overleftarrow{C}, \overrightarrow{C}) = \{C(M_1, (\overleftarrow{x}_1, \overrightarrow{x}_1)), C(M_2, (\overleftarrow{x}_2, \overrightarrow{x}_2)), \dots, C(M_m, (\overleftarrow{x}_m, \overrightarrow{x}_m))\}$, which assumes the current number of events is m .

Algorithm 7.1 The update strategy of the belief database

Repeat:

Query the belief database

If the belief has existed

If the value of the belief is invariable, do not do any conduction and turn to the next belief.

Else modify the value of the belief in the database and turn to the next belief.

Else insert the belief in the database and turn to the next belief.

Until: all the beliefs were conducted

Different problems result from the learning process of the agent. We need to solve these problems. First, the capacity of the agent to solve one problem may be zero. However, as the learning proceeds, the capacity of the agent is enhanced. When the agent accomplishes one step of the learning process, one of the following cases exists:

1. For a certain problem, the agent's capacity to solve it has not changed.
2. For a certain problem, the agent's capacity to solve it has changed.
3. For a certain problem, the agent's capacity to solve it does not exist in the database.

The structure of the capacity database is analogous to that of the belief database, and we present the updating strategy in Algorithm 7.2.

Algorithm 7.2 The update strategy of the capacity database

Repeated:

Query the capacity database

If the capacity to solve the problem has existed,

If the capacity value of the agent is invariable, do not do any conduction and turn to the next belief.

Else modify the value in the database and turn to the next belief.

Else insert the capacity to solve the problem in the database and turn to the next belief.

Until: all the problems were conducted

Tab. 7.2: Structure of intention database.

Intention	Strategy	Event
$i-1$
i	Sty_1	$(\overset{\leftarrow}{x}_i^1, \vec{x}_i^1)$
i	Sty_2	$(\overset{\leftarrow}{x}_i^2, \vec{x}_i^2)$
...
i	Sty_n	$(\overset{\leftarrow}{x}_i^n, \vec{x}_i^n)$
$i+1$

I (Intention) represents the prearrangements of ones behaviour in future events and represents the action orientation of the agent for future events. In the i th step of the learning process, we assume the number of next-time executable strategies of the agent is n , which means the strategy set is $\{Sty_1, Sty_2, \dots, Sty_n\}$. $I_i(Sty_j, (\overset{\leftarrow}{x}_i^j, \vec{x}_i^j))$ represents the intention of the executable strategy Sty_j in the i th learning step with respect to $(\overset{\leftarrow}{x}_i^j, \vec{x}_i^j)$. I_i is the intention set of optional strategies of the agent in the i th learning step, and the number of current optional strategies is n , or $(\overset{\leftarrow}{I}_i, \vec{I}_i) = \{I_i(Sty_1, (\overset{\leftarrow}{x}_i^1, \vec{x}_i^1)), I_i(Sty_2, (\overset{\leftarrow}{x}_i^2, \vec{x}_i^2)), \dots, I_i(Sty_n, (\overset{\leftarrow}{x}_i^n, \vec{x}_i^n))\}$. Moreover, the whole intention set can be represented by $(\overset{\leftarrow}{I}, \vec{I}) = \{(\overset{\leftarrow}{I}_1, \vec{I}_1), (\overset{\leftarrow}{I}_2, \vec{I}_2), \dots, (\overset{\leftarrow}{I}_m, \vec{I}_m)\}$.

According to the definition of intention, we know that if the agent learns according to the specific strategies and eliminates the possibility of backtracking, the intention database will be waiting for new intentions. However, if backtracking occurs, the intention database will be modified. The structure and updating algorithm for the intention database are shown in Tab. 7.2 and Algorithm 7.3.

Algorithm 7.3 The update strategy of the intention database

Repeated:

If the current strategy was effective

Insert the intention defined by the current strategy into the intention database

Else trace back to the last step and delete the corresponding intention information

A (Action) saves the information of all actions in every step of the learning process. If we set the agent to move into the i th step of the learning process, the current optional action sequence could be represented by $A_i = \{A_i^1, A_i^2, \dots, A_i^m\}$, where the number of current optional actions is m . When the agent moves into the n th step of the learning process, the set of optional actions can be defined by $A = \{A_1, A_2, \dots, A_n\}$. The structure and updating algorithm of the action database are shown in Tab. 7.3 and Algorithm 7.4.

Tab. 7.3: Structure of action database.

Index	Action
$i - 1$...
i	A_i^1
i	A_i^2
...	...
i	A_i^m
$i + 1$...
...	...

Algorithm 7.4 The update strategy of the action database**Repeated:**

If the current strategy was effective

Insert the action defined by the current strategy into the action database

Else trace back to the last step and delete the corresponding action information

S (State) is the set of current states of the agent. $(\overleftarrow{S}_i, \overrightarrow{S}_i)$ represents the state in the i th learning step. $(\overleftarrow{S}, \overrightarrow{S})$ represents the set of states that have happened after n learning steps, or $(\overleftarrow{S}, \overrightarrow{S}) = \{(\overleftarrow{S}_1, \overrightarrow{S}_1), (\overleftarrow{S}_2, \overrightarrow{S}_2), \dots, (\overleftarrow{S}_n, \overrightarrow{S}_n)\}$. The structure and updating algorithm of the state database are shown in Tab. 7.4 and Algorithm 7.5.

Sty (Strategy) records the learning path defined by the learning rules. These learning paths are recorded during the process of learning, and this set is actually a subset of the action set.

Tab. 7.4: Structure of state database.

Index	State
$i - 1$...
i	$(\overleftarrow{S}_1, \overrightarrow{S}_1)$
i	$(\overleftarrow{S}_2, \overrightarrow{S}_2)$
...	...
i	$(\overleftarrow{S}_n, \overrightarrow{S}_n)$
$i + 1$...

Algorithm 7.5 The update strategy of the state database**Repeated:**

If the current strategy was effective

Insert the state defined by the current strategy into the state database

Else trace back to the last step and delete the corresponding state information

Tab. 7.5: Action database of Example 7.1.

State	Path
1	Left
1	Right
1	Up
2	Down
2	Left
3	Up
3	Right

Tab. 7.6: Strategy database of Example 7.1.

Path	Strategy
Left	A_1^1
Down	A_2^1
Right	A_3^1

Example 7.1 After the agent has been through three steps of the learning process, the action database is as shown in Tab. 7.5

Based on the target, capability database, and intention database of the agent, fused with the rewards and punishments provided by the ambient environment, we choose the path Left → Down → Right: $A_1 = A_1^1 = \text{Left}$, $A_2 = A_2^1 = \text{Down}$, $A_3 = A_3^1 = \text{Right}$. Therefore, we obtain the strategy database shown in Tab. 7.6.

In the learning process, we find that the current strategy database is not optimal for deeper study. Thus, we trace back to the last step, delete the outdated strategies, and insert the new strategies.

Theorem 7.1 The current strategy is $Sty_i = \{A_1^{i_1}, A_2^{i_2}, \dots, A_n^{i_n}\}$. During the learning process, a better strategy $Sty_j = \{A_1^{j_1}, A_2^{j_2}, \dots, A_n^{j_n}\}$ has been found. Deleting Sty_i will not affect the optimal learning process of the agent.

Proof: The choice of strategy during the learning process follows certain rules. As the superiority of the new strategy has been demonstrated, the outdated strategy Sty_i , is obviously not the best, and deleting it will not cause an error.

R (Reward): After the agent takes actions on the environment, the environment will produce a value, named the reward. If the environment returns a positive value when the agent takes an action, the possibility of the action happened in the next time will increase, and vice versa. By the definition of the reward value, each action corresponds to a value. Thus, the reward value in the i th learning step of the agent is $(\overleftarrow{R}_i, \overrightarrow{R}_i) = \{(\overleftarrow{R}_i^1, \overrightarrow{R}_i^1), (\overleftarrow{R}_i^2, \overrightarrow{R}_i^2), \dots, (\overleftarrow{R}_i^m, \overrightarrow{R}_i^m)\}$. When the learning

Tab. 7.7: Structure of reward database.

Index	Reward
$i - 1$...
i	(R_i^1, R_i^1)
i	(R_i^2, R_i^2)
...	...
i	(R_i^m, R_i^m)
$i + 1$...
...	...

process moves into the n th step, the set of reward values is defined by $(\overleftarrow{R}, \overrightarrow{R}) = \{(\overleftarrow{R}_1, \overrightarrow{R}_1), (\overleftarrow{R}_2, \overrightarrow{R}_2), \dots, (\overleftarrow{R}_n, \overrightarrow{R}_n)\}$.

The structure and updating algorithm of the reward database are shown in Tab. 7.7 and Algorithm 7.6.

Algorithm 7.6 The update strategy of the reward database

Repeated:

If the current strategy was effective

Insert the reward value defined by the current strategy into the reward database

 Else trace back to the last step and delete the corresponding reward information

Brf (Belief Revision Function): As the environment changes, the state of the agent also changes. This kind change results in the beliefs of the agent being constantly modified, and the belief revision function is $Brf : (\overleftarrow{b}_i, \overrightarrow{b}_i) \times (\overleftarrow{s}_i, \overrightarrow{s}_i) \rightarrow (\overleftarrow{b}_{i+1}, \overrightarrow{b}_{i+1})$.

Acf (Action Choice Function): Each action performed by the agent depends on the agent's belief, capacity, intention, and its own external environment. Thus, the action choice function is $Acf : (\overleftarrow{b}_i, \overrightarrow{b}_i) \times (\overleftarrow{c}_i, \overrightarrow{c}_i) \times (\overleftarrow{i}_i, \overrightarrow{i}_i) \times (\overleftarrow{s}_i, \overrightarrow{s}_i) \rightarrow a_i$.

Irf (Intention Revision Function): As the environment changes, the state of the agent also constantly changes. This results in the intentions of the agent being constantly modified, and the intention revision function is $Irf : (\overleftarrow{i}_i, \overrightarrow{i}_i) \times (\overleftarrow{s}_i, \overrightarrow{s}_i) \rightarrow (\overleftarrow{i}_{i+1}, \overrightarrow{i}_{i+1})$.

Stf (State Transition Function): This function maps the Cartesian product between action sets A and B to the state set. Thus, the state transition function is $Stf : a_i \times (\overleftarrow{s}_i, \overrightarrow{s}_i) \rightarrow (\overleftarrow{s}_{i+1}, \overrightarrow{s}_{i+1})$.

7.2.2 Related axioms

There are some relationships between the mental state of each agent that we call the mutual generation and restriction principle. The mutual generation between mental

states refers to the ability of mental states to promote one another, and the mutual restriction refers to the ability of mental states to restrict one another.

a) Belief:

$$\text{Axiom 7.1 } B(P_i, \overline{(\overleftarrow{x}_i, \overrightarrow{x}_i)}) = \overline{B(P_i, (\overleftarrow{x}_i, \overrightarrow{x}_i))}$$

$$\text{Axiom 7.2 } B(P_i, (\overleftarrow{x}_{i1}, \overrightarrow{x}_{i1})) \wedge B(P_i, (\overleftarrow{x}_{i2}, \overrightarrow{x}_{i2})) = B(P_i, (\overleftarrow{x}_{i1}, \overrightarrow{x}_{i1}) \wedge (\overleftarrow{x}_{i2}, \overrightarrow{x}_{i2}))$$

$$\text{Axiom 7.3 } B(P_i, (\overleftarrow{x}_{i1}, \overrightarrow{x}_{i1})) \vee B(P_i, (\overleftarrow{x}_{i2}, \overrightarrow{x}_{i2})) = B(P_i, (\overleftarrow{x}_{i1}, \overrightarrow{x}_{i1}) \vee (\overleftarrow{x}_{i2}, \overrightarrow{x}_{i2}))$$

b) Capacity:

$$\text{Axiom 7.4 } C(M_i, \overline{(\overleftarrow{x}_i, \overrightarrow{x}_i)}) = \overline{C(M_i, (\overleftarrow{x}_i, \overrightarrow{x}_i))}$$

$$\text{Axiom 7.5 } C(M_i, (\overleftarrow{x}_{i1}, \overrightarrow{x}_{i1})) \wedge C(M_i, (\overleftarrow{x}_{i2}, \overrightarrow{x}_{i2})) = C(M_i, (\overleftarrow{x}_{i1}, \overrightarrow{x}_{i1}) \wedge (\overleftarrow{x}_{i2}, \overrightarrow{x}_{i2}))$$

$$\text{Axiom 7.6 } C(M_i, (\overleftarrow{x}_{i1}, \overrightarrow{x}_{i1})) \vee C(M_i, (\overleftarrow{x}_{i2}, \overrightarrow{x}_{i2})) = C(M_i, (\overleftarrow{x}_{i1}, \overrightarrow{x}_{i1}) \vee (\overleftarrow{x}_{i2}, \overrightarrow{x}_{i2}))$$

c) Intention:

$$\text{Axiom 7.7 } I_i(Sty_j, \overline{(\overleftarrow{x}_i^j, \overrightarrow{x}_i^j)}) = \overline{I_i(Sty_j, (\overleftarrow{x}_i^j, \overrightarrow{x}_i^j))}$$

$$\text{Axiom 7.8 } I_i(Sty_j, (\overleftarrow{x}_i^{j1}, \overrightarrow{x}_i^{j1})) \wedge I_i(Sty_j, (\overleftarrow{x}_i^{j2}, \overrightarrow{x}_i^{j2})) = I_i(Sty_j, (\overleftarrow{x}_i^{j1}, \overrightarrow{x}_i^{j1}) \wedge (\overleftarrow{x}_i^{j2}, \overrightarrow{x}_i^{j2}))$$

$$\text{Axiom 7.9 } I_i(Sty_j, (\overleftarrow{x}_i^{j1}, \overrightarrow{x}_i^{j1})) \vee I_i(Sty_j, (\overleftarrow{x}_i^{j2}, \overrightarrow{x}_i^{j2})) = I_i(Sty_j, (\overleftarrow{x}_i^{j1}, \overrightarrow{x}_i^{j1}) \vee (\overleftarrow{x}_i^{j2}, \overrightarrow{x}_i^{j2}))$$

The mental states of the agent are not independent but are correlated and mutually restrictive. From the two mapping relations $Acf : (\overleftarrow{b}_i, \overrightarrow{b}_i) \times (\overleftarrow{c}_i, \overrightarrow{c}_i) \times (\overleftarrow{i}_i, \overrightarrow{i}_i) \times (\overleftarrow{s}_i, \overrightarrow{s}_i) \rightarrow a_i$ and $Stf : a_i \times (\overleftarrow{s}_i, \overrightarrow{s}_i) \rightarrow (\overleftarrow{s}_{i+1}, \overrightarrow{s}_{i+1})$, we can see that there are some relationships between action a_i and state $(\overleftarrow{s}_i, \overrightarrow{s}_i)$. If the action makes the state move toward the target state, the relationship between them is one of facilitation. Conversely, the relationship may be restrictive. Obviously, under a certain state $(\overleftarrow{s}_i, \overrightarrow{s}_i)$, the action of the agent is determined by belief $(\overleftarrow{b}_i, \overrightarrow{b}_i)$, capacity $(\overleftarrow{c}_i, \overrightarrow{c}_i)$, and intention $(\overleftarrow{i}_i, \overrightarrow{i}_i)$. Consequently, there are certain relationships between each kind of attribute.

7.2.3 Working mechanism

Based on the above model, we now introduce the main working mechanism. Based on DFL, the agent can be divided into a responding layer, learning layer, and thinking layer. The responding layer mainly realizes the information exchange between the agent and the external environment. The learning layer handles the learning process

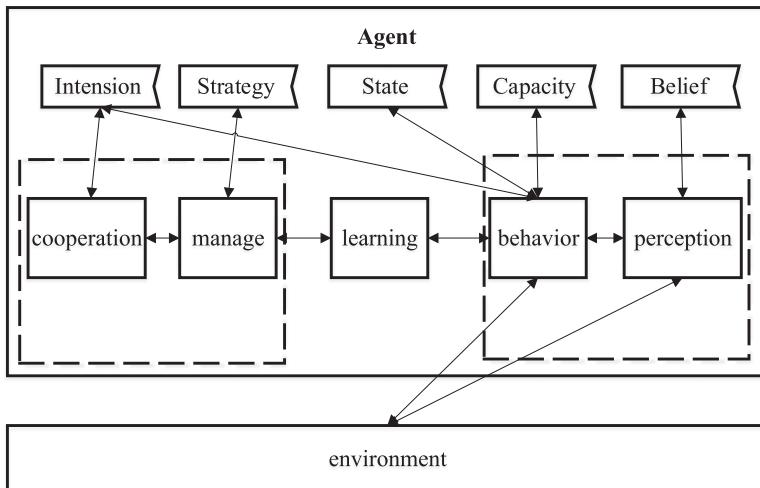


Fig. 7.1: The structure of the learning process of the agent.

of the agent. The thinking layer includes two parts, one to design the overall learning process of the agent and another to consider collaboration with other agents. The concrete working mechanism is shown in Fig. 7.1.

In a constantly changing environment, the agent is an entity with the following basic working mechanism. The agent perceives changes in the environment and transmits this information to the perception layer.

The perception layer conducts new perceptual behaviour based on belief and the result from the previous behaviour layer and transmits the new perceptual result to the current behaviour layer. Then, it updates the belief database.

The action taken by the behaviour layer is dependent on the current capacity, state, and intention of the agent and the result of the learning layer. When the behaviour layer takes a new action, it transmits the result of the new action to the learning layer and the perception layer. Then, it updates the corresponding capacity database, state database, and intention database.

The learning layer conducts new learning steps based on the new action transmitted from the behaviour layer and the new management plans transmitted from the management layer.

The management layer makes plans for the next learning step based on the result from the learning layer, cooperation information from the cooperation layer, and the strategies in the strategy database. If the management layer has made a new plan, it will transmit the new plan to the learning layer and cooperation layer at the same time and update the corresponding strategy database.

The cooperation layer decides how to collaborate with other agents based on the plan from the management layer and the current intention from the intention database and updates the intention database.

7.3 Single-agent learning algorithm based on DFL

This section discusses the single-agent learning algorithm based on the agent mental model described in the previous section.

7.3.1 Learning task

The learning task of the agent is defined as follows: when learning a strategy $\pi : S \rightarrow A$, the agent will select the state for the next action based on the reward value r .

Definition 7.2 Starting with any state s_i , $V^\pi(s_i)$ is acquired with any strategy π as

$$V^\pi(s_i) = r_i + \gamma r_{i+1} + \gamma^2 r_{i+2} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{i+k}, \quad (7.1)$$

where γ is a constant in the range $\{0 - 1\}$. If $\gamma = 0$, we obtain an immediate return from the agent. If $\gamma = 1$, we will obtain the result that the future return is more important than the immediate return. This section starts by considering the simplest case of an immediate return. Thus, we first present the immediate return single-agent learning algorithm.

7.3.2 Immediate return single-agent learning algorithm based on DFL

Definition 7.3 $(\overleftarrow{S}, \overrightarrow{S})$ is the set of state $(\overleftarrow{S}_i, \overrightarrow{S}_i)$ of the agent. A is the action set of the agent, named A_i , and $(\overleftarrow{R}, \overrightarrow{R})$ is the reward set of the agent, named $(\overleftarrow{R}_i, \overrightarrow{R}_i)$. At a particular time, if the agent is in a certain state $(\overleftarrow{S}_t, \overrightarrow{S}_t)$ and the actions that can be taken by the agent are represented by $a_t = \{a_t^1, a_t^2, \dots, a_t^m\}$, $m \geq 1$, the learning process of the agent will be defined as follows:

The agent would like to obtain the different reward values $(\overleftarrow{r}_t^i, \overrightarrow{r}_t^i)$ ($1 \leq i \leq m$) and the final reward value set $(\overleftarrow{r}_t, \overrightarrow{r}_t) = \{(\overleftarrow{r}_t^1, \overrightarrow{r}_t^1), (\overleftarrow{r}_t^2, \overrightarrow{r}_t^2), \dots, (\overleftarrow{r}_t^m, \overrightarrow{r}_t^m)\}$ by taking different actions a_t^i ($1 \leq i \leq m$). Thus, the agent will select the peak value in the set, defined as $(\overleftarrow{r}_t^{\max}, \overrightarrow{r}_t^{\max})$ ($1 \leq \max \leq m$). Based on the peak reward value, the agent will obtain the corresponding action a_t^{\max} ($1 \leq \max \leq m$). Therefore, the current state of the agent will be converted from $(\overleftarrow{S}_t, \overrightarrow{S}_t)$ to $(\overleftarrow{S}_{t+1}, \overrightarrow{S}_{t+1})$. Overall, the formulation of the learning process of the agent can be represented as $(\overleftarrow{R}_t, \overrightarrow{R}_t) : (\overleftarrow{S}_{t-1}, \overrightarrow{S}_{t-1}) \times A_t \rightarrow (\overleftarrow{S}_t, \overrightarrow{S}_t)$.

The immediate return single-agent learning is the simplest agent learning approach. This method has no need to consider whether the follow-up action exerts any influence on the current learning step. The key to this method is to select the

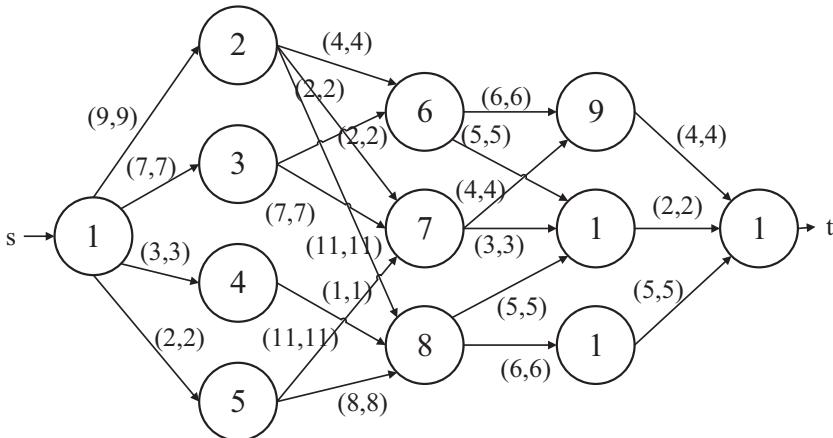


Fig. 7.2: Example 7.2.

maximal reward value during the learning process of the agent, which has a low computational complexity. It turns out that, if we always select the maximal reward value in every step, the final reward value will not be optimal.

Example 7.2 (See Fig. 7.2).

Let point 1 be the initial state S and point 12 be the target state T. Points 2–11 represent the states that are accessible, and the weights are the reward values.

Starting from state S, we obtain the following learning process under the computational method described by Definition 7.3.

- (1) $S = \{1\}$, $a_1 = \{1 \rightarrow 2, 1 \rightarrow 3, 1 \rightarrow 4, 1 \rightarrow 5\}$, $(\overleftarrow{r}_1, \overrightarrow{r}_1) = \{(\overleftarrow{9}, \overrightarrow{9}), (\overleftarrow{7}, \overrightarrow{7}), (\overleftarrow{3}, \overrightarrow{3}), (\overleftarrow{2}, \overrightarrow{2})\}$.
We know that $(\overleftarrow{R}_1, \overrightarrow{R}_1) = (\overleftarrow{9}, \overrightarrow{9})$, $A_1 = 1 \rightarrow 2$. As $(\overleftarrow{R}, \overrightarrow{R}) = \{(\overleftarrow{9}, \overrightarrow{9})\}$, $A = \{1 \rightarrow 2\}$ does not allow the agent to reach the target state; the learning process continues.
- (2) $S = \{1, 2\}$, $(\overleftarrow{r}_2, \overrightarrow{r}_2) = \{(\overleftarrow{4}, \overrightarrow{4}), (\overleftarrow{2}, \overrightarrow{2}), (\overleftarrow{1}, \overrightarrow{1})\}$.
We know that $(\overleftarrow{R}_2, \overrightarrow{R}_2) = (\overleftarrow{4}, \overrightarrow{4})$, $A_2 = 2 \rightarrow 6$. As $(\overleftarrow{R}, \overrightarrow{R}) = \{(\overleftarrow{9}, \overrightarrow{9}), (\overleftarrow{4}, \overrightarrow{4})\}$, $A = \{1 \rightarrow 2, 2 \rightarrow 6\}$ does not reach the target state; the learning process continues.
- (3) $S = \{1, 2, 6\}$, $a_3 = \{6 \rightarrow 9, 6 \rightarrow 10\}$, $(\overleftarrow{r}_3, \overrightarrow{r}_3) = \{(\overleftarrow{6}, \overrightarrow{6}), (\overleftarrow{5}, \overrightarrow{5})\}$.
We know that $(\overleftarrow{R}_3, \overrightarrow{R}_3) = (\overleftarrow{6}, \overrightarrow{6})$, $A_3 = 6 \rightarrow 9$. As $(\overleftarrow{R}, \overrightarrow{R}) = \{(\overleftarrow{9}, \overrightarrow{9}), (\overleftarrow{4}, \overrightarrow{4}), (\overleftarrow{6}, \overrightarrow{6})\}$, $A = \{1 \rightarrow 2, 2 \rightarrow 6, 6 \rightarrow 9\}$ does not reach the target state; the learning process continues.
- (4) $S = \{1, 2, 6, 9\}$, $a_4 = \{9 \rightarrow 12\}$, $(\overleftarrow{r}_4, \overrightarrow{r}_4) = \{(\overleftarrow{4}, \overrightarrow{4})\}$.

We know that $(\overleftarrow{R}_4, \overrightarrow{R}_4) = (\overleftarrow{4}, \overrightarrow{4})$, $A_3 = 9 \rightarrow 12$. As $(\overleftarrow{R}, \overrightarrow{R}) = \{(\overleftarrow{9}, \overrightarrow{9}), (\overleftarrow{4}, \overrightarrow{4}), (\overleftarrow{6}, \overrightarrow{6}), (\overleftarrow{4}, \overrightarrow{4})\}$, $A = \{1 \rightarrow 2, 2 \rightarrow 6, 6 \rightarrow 9, 9 \rightarrow 12\}$ has reached the target state; the learning process is terminated.

Therefore, the final strategy of the agent is $\pi = A = \{1 \rightarrow 2, 2 \rightarrow 6, 6 \rightarrow 9, 9 \rightarrow 12\}$. If the accumulated reward value of this example is defined as the sum of the reward values from the initial state to the target state, the result would be $(\overleftarrow{9}, \overrightarrow{9}) + (\overleftarrow{4}, \overrightarrow{4}) + (\overleftarrow{6}, \overrightarrow{6}) + (\overleftarrow{4}, \overrightarrow{4}) = (\overleftarrow{23}, \overrightarrow{23})$.

However, in Fig. 7.2, we can see that the accumulated reward value $(\overleftarrow{23}, \overrightarrow{23})$ is not the maximal value obtained from $\pi = \{1 \rightarrow 2, 2 \rightarrow 6, 6 \rightarrow 9, 9 \rightarrow 12\}$. There is an action set $\{1 \rightarrow 4, 4 \rightarrow 8, 8 \rightarrow 11, 11 \rightarrow 12\}$ whose corresponding rewarding value is $(\overleftarrow{3}, \overrightarrow{3}) + (\overleftarrow{11}, \overrightarrow{11}) + (\overleftarrow{6}, \overrightarrow{6}) + (\overleftarrow{5}, \overrightarrow{5}) = (\overleftarrow{25}, \overrightarrow{25}) > (\overleftarrow{23}, \overrightarrow{23})$.

How can we find the optimal strategy, in which the enumeration method is obviously clumsy? Baermann proposed using the principle of optimality, and the resulting recurrence relation provides a better method. By this method, we can easily obtain the optimal sequence of strategies and decrease the problem size. The principle of optimality indicates that the optimal sequence of strategies of the learning process has a specific property: no matter what the initial state and the target state, the intermediate strategies must construct an optimal strategy with respect to the initial strategy.

What follows is a non-immediate return reinforcement learning algorithm, Q-learning. This algorithm can obtain the optimal control strategy from the delayed retribution, even without prior knowledge.

7.3.3 Q-learning function based on DFL

Definition 7.4 The evaluation function $Q((\overleftarrow{s}, \overrightarrow{s}), a)$ of Q-learning based on DFL is defined as follows: the reward value of Q-learning is accumulated from conducting action A under state $(\overleftarrow{s}, \overrightarrow{s})$. When the agent obtains the accumulated reward value that has been added the value of the optimal strategy, we get

$$Q((\overleftarrow{s}, \overrightarrow{s}), a) \equiv (\overleftarrow{R}, \overrightarrow{R})((\overleftarrow{s}, \overrightarrow{s}), a) + \gamma V^*((\overleftarrow{s}', \overrightarrow{s}')). \quad (7.2)$$

Here, $(\overleftarrow{R}, \overrightarrow{R})((\overleftarrow{s}, \overrightarrow{s}), a)$ is the immediate return value after the agent has taken action A under state $(\overleftarrow{s}, \overrightarrow{s})$. State $(\overleftarrow{s}', \overrightarrow{s}')$ is the subsequent state after action A has been taken.

Now, we have a problem. If the Q function corresponds to the optimal strategy, how should the agent obtain the Q function? Let us first make the following analysis. At first, we build a relationship between Q and V^* and the following relationship exists:

$$V^*(s) = \max_{a'} Q(s, a'). \quad (7.3)$$

Through a simple mathematical derivation, we have

$$Q((\overleftarrow{s}, \overrightarrow{s}), a) \equiv (\overleftarrow{R}, \overrightarrow{R})((\overleftarrow{s}, \overrightarrow{s}), a) + \gamma \max_{a'} Q((\overleftarrow{s}', \overrightarrow{s}'), a'). \quad (7.4)$$

Now we know the Q function is determined by the subsequent Q function and the immediate return functions.

7.3.4 Q-learning algorithm based on DFL

We now present Algorithm 7.7 (the Q-learning algorithm based on DFL) and give an example.

Example 7.3 We have six lattices, each representing a state that the agent can access (See Fig. 7.3). To express this conveniently, we set the serial numbers of the lattices to run from 1 to 6 from left to right, top to bottom. The central lattice is number 3 and is the target state.

Algorithm 7.7 The Q-learning algorithm based on DFL

Input: state of the agent is $(\overleftarrow{s}, \overrightarrow{s})$, action of the agent is A, value of the Q function $Q((\overleftarrow{s}, \overrightarrow{s}, a))$ is $(\overleftarrow{0}, \overrightarrow{0})$.

Repeated: Monitoring the current state

Select an action from the current action list, and conduct it.

When the agent receives the action, its immediate rewarding value is $(\overleftarrow{r}, \overrightarrow{r})$.

Check the new state $(\overleftarrow{s}', \overrightarrow{s}')$.

Update the list item $Q((\overleftarrow{s}, \overrightarrow{s}), a)$ as follows:

$Q((\overleftarrow{s}, \overrightarrow{s}), a) \leftarrow (\overleftarrow{R}, \overrightarrow{R})((\overleftarrow{s}, \overrightarrow{s}), a) + \gamma \max Q((\overleftarrow{s}', \overrightarrow{s}'), a')$

Modify the current state $(\overleftarrow{s}, \overrightarrow{s} \leftarrow (\overleftarrow{s}', \overrightarrow{s}'))^a$.

Because the state of the agent is a dynamic fuzzy variable, we assume the agent will obtain an enhanced reward value with reduced ability when its state changes from number 6 to number 3. However, the agent will obtain a subdued reward value with enhanced ability when its state changes from number 5 to number 2.

Therefore, we assume that the immediate reward value of every state in the learning process of the agent will change. Then, under the Q-learning algorithm based on DFL, we obtain the corresponding value of the state $Q((\overleftarrow{s}, \overrightarrow{s}), a)$ as in Fig. 7.3. If the initial state of the agent is number 4, the optimal path to reach the optimal state is $4 \rightarrow 5 \rightarrow 2 \rightarrow 3$.

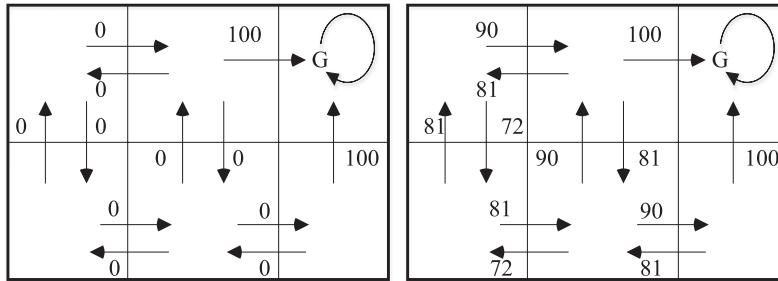


Fig. 7.3: The example of Algorithm 7.7.

Example 7.4 We use the Q-learning algorithm based on DFL to solve Example 7.3. Set the maximal reward value of the agent when starting from number n to be $Q(n, *)$. Then, we have $Q(9, 9 \rightarrow 12) = (\overleftarrow{4}, \overrightarrow{4})$. So, $Q(9, *) = Q(9, 9 \rightarrow 12) = (\overleftarrow{4}, \overrightarrow{4})$ and $Q(10, 10 \rightarrow 12) = (\overleftarrow{2}, \overrightarrow{2})$.

Thus, it convert to $Q(10, *) = Q(10, 10 \rightarrow 12) = (\overleftarrow{2}, \overrightarrow{2})$ and $Q(11, 11 \rightarrow 12) = (\overleftarrow{5}, \overrightarrow{5})$.

Then, we have

$$Q(11, *) = Q(11, 11 \rightarrow 12) = (\overleftarrow{5}, \overrightarrow{5}), \quad (7.5)$$

$$Q(6, 6 \rightarrow 9) = (\overleftarrow{6}, \overrightarrow{6}) + Q(9, *) = (\overleftarrow{6}, \overrightarrow{6}) + (\overleftarrow{4}, \overrightarrow{4}) = (\overleftarrow{10}, \overrightarrow{10}), \quad (7.6)$$

$$Q(6, 6 \rightarrow 10) = (\overleftarrow{5}, \overrightarrow{5}) + Q(10, *) = (\overleftarrow{5}, \overrightarrow{5}) + (\overleftarrow{2}, \overrightarrow{2}) = (\overleftarrow{7}, \overrightarrow{7}). \quad (7.7)$$

Therefore, we know

$$Q(6, *) = \max\{Q(6, 6 \rightarrow 9), Q(6, 6 \rightarrow 10)\} = (\overleftarrow{10}, \overrightarrow{10}) \quad (7.8)$$

$$Q(7, 7 \rightarrow 9) = (\overleftarrow{4}, \overrightarrow{4}) + Q(9, *) = (\overleftarrow{4}, \overrightarrow{4}) + (\overleftarrow{4}, \overrightarrow{4}) = (\overleftarrow{8}, \overrightarrow{8}), \quad (7.9)$$

$$Q(7, 7 \rightarrow 10) = (\overleftarrow{3}, \overrightarrow{3}) + Q(10, *) = (\overleftarrow{3}, \overrightarrow{3}) + (\overleftarrow{2}, \overrightarrow{2}) = (\overleftarrow{5}, \overrightarrow{5}). \quad (7.10)$$

Hence, in the next step, we have

$$Q(7, *) = \max\{Q(7, 7 \rightarrow 9), Q(7, 7 \rightarrow 10)\} = (\overleftarrow{8}, \overrightarrow{8}), \quad (7.11)$$

$$Q(8, 8 \rightarrow 10) = (\overleftarrow{5}, \overrightarrow{5}) + Q(10, *) = (\overleftarrow{5}, \overrightarrow{5}) + (\overleftarrow{2}, \overrightarrow{2}) = (\overleftarrow{7}, \overrightarrow{7}), \quad (7.12)$$

$$Q(8, 8 \rightarrow 11) = (\overleftarrow{6}, \overrightarrow{6}) + Q(11, *) = (\overleftarrow{6}, \overrightarrow{6}) + (\overleftarrow{5}, \overrightarrow{5}) = (\overleftarrow{11}, \overrightarrow{11}). \quad (7.13)$$

Finally, $Q(8, *) = \max\{Q(8, 8 \rightarrow 10), Q(8, 8 \rightarrow 11)\} = (\overleftarrow{11}, \overrightarrow{11})$.

The same can be processed as $Q(2, *) = (\overleftarrow{14}, \overrightarrow{14})$, $Q(3, *) = (\overleftarrow{15}, \overrightarrow{15})$, $Q(4, *) = (\overleftarrow{22}, \overrightarrow{22})$, $Q(5, *) = (\overleftarrow{19}, \overrightarrow{19})$.

Thus, we get

$$Q(1, 1 \rightarrow 2) = (\overleftarrow{9}, \overrightarrow{9}) + Q(2, *) = (\overleftarrow{9}, \overrightarrow{9}) + (\overleftarrow{14}, \overrightarrow{14}) = (\overleftarrow{23}, \overrightarrow{23}), \quad (7.14)$$

$$Q(1, 1 \rightarrow 3) = (\overleftarrow{7}, \overrightarrow{7}) + Q(3, *) = (\overleftarrow{7}, \overrightarrow{7}) + (\overleftarrow{15}, \overrightarrow{15}) = (\overleftarrow{22}, \overrightarrow{22}), \quad (7.15)$$

$$Q(1, 1 \rightarrow 4) = (\overleftarrow{3}, \overrightarrow{3}) + Q(4, *) = (\overleftarrow{3}, \overrightarrow{3}) + (\overleftarrow{22}, \overrightarrow{22}) = (\overleftarrow{25}, \overrightarrow{25}), \quad (7.16)$$

$$Q(1, 1 \rightarrow 5) = (\overleftarrow{2}, \overrightarrow{2}) + Q(5, *) = (\overleftarrow{2}, \overrightarrow{2}) + (\overleftarrow{19}, \overrightarrow{19}) = (\overleftarrow{21}, \overrightarrow{21}). \quad (7.17)$$

Here, $Q(1, *) = (\overleftarrow{25}, \overrightarrow{25})$, so the optimal strategy is $\pi = \{1 \rightarrow 4, 4 \rightarrow 8, 8 \rightarrow 11, 11 \rightarrow 12\}$, and the maximal accumulated reward value is $(\overleftarrow{25}, \overrightarrow{25})$, or $Q(1, *) = (\overleftarrow{25}, \overrightarrow{25})$.

7.4 Multi-agent learning algorithm based on DFL

In the previous section, we developed a single-agent learning algorithm based on DFL. All the single-agent learning algorithms build a foundation for multi-agent learning. In the multi-agent learning model, each agent is learning according to a particular single-agent learning algorithm based on its own position in the model.

7.4.1 Multi-agent learning model based on DFL

Definition 7.5 In the multi-agent learning model, each circular symbol is an agent (see Fig. 7.4). Some agents form an agent group. Each agent in the same group cooperates with the other in the group. Different groups compete with each other.

We now introduce a cooperative multi-agent learning algorithm and a competitive multi-agent learning algorithm.

7.4.2 Cooperative multi-agent learning algorithm based on DFL

In the real world, when a team is tasked with achieving a goal, there are various ways of proceeding. For example, each person in the team can provide an idea, and the team as a whole makes a comprehensive assessment of all the ideas. Alternatively, the

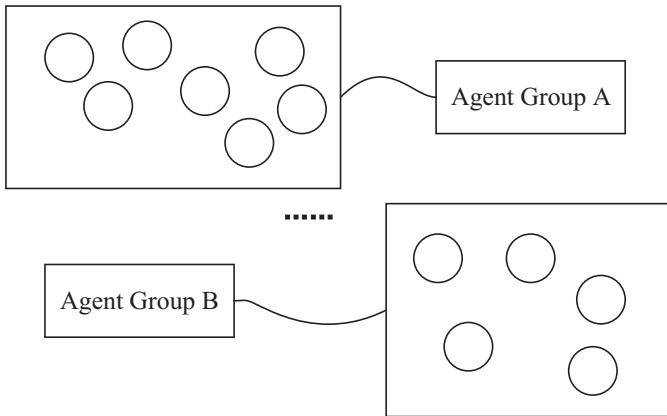


Fig. 7.4: The multi-agent learning model.

mission could be divided into several parts, each of which is assigned to a different team member.

Based on the existing examples in the real world, we conduct cooperative multi-agent learning in two steps, common learning and comprehensive learning. The definition of common learning is that we set each agent to be an independent entity in the multi-agent system. Each agent completes its own task according to a particular single learning algorithm. An agent has properties such as independence and integrity. Independence means that each agent is an independent entity that is unaffected by other agents. Integrity means that each agent has a complete capacity for learning without any help from other agents. Comprehensive learning means that there is a comprehensive agent that manages every agent and obtains the optimal learning result in the shortest time. The comprehensive agent has properties such as comprehensiveness and sharing. Comprehensiveness means that the comprehensive agent can manage every agent in the learning system and assign different tasks to each agent. At the end of the mission, the comprehensive agent will choose the optimal learning sequence based on the reward value. Sharing means that each agent in the learning system has no need to complete all the learning tasks. This method will reduce the complexity of the agent.

At first, we define the common cooperative multi-agent learning model as

$$< (agent1, agent2, \dots, agentn), ((\overleftarrow{S}_0, \overrightarrow{S}_0), (\overleftarrow{S}_g, \overrightarrow{S}_g)) > . \quad (7.18)$$

Here, we have a total of n agents, each with the same initial state and the same target state. In the multi-agent model, we concurrently conduct single-agent learning and orchestrate the multiple agents to form a system. Over a period of time, the state of

the agents changes several times, from the initial state $(\overleftarrow{S}_0, \overrightarrow{S}_0)$ to the present state $(\overleftarrow{S}_g, \overrightarrow{S}_g)$ under the above learning method.

Secondly, the common cooperative multi-agent learning algorithm is as follows. In Algorithm 7.8, we assume that the agent has to conduct m_i tests when state $(\overleftarrow{S}_i, \overrightarrow{S}_i)$ ($0 \leq i \leq g - 1$) is converted into state $(\overleftarrow{S}_{i+1}, \overrightarrow{S}_{i+1})$. Then, the agent can decide which action best assist the change in state.

In Algorithm 7.8, we first finish the parallel learning mission for every agent and then group multiple agents to form the system. In this way, the time complexity of the grouping is lower than for parallel learning. Therefore, the time complexity of this method is $T_1 = t_1 + t_2 + \dots + t_n + t_{else}$, where t_{else} is the extra time cost of operations such as grouping.

Algorithm 7.8 The common cooperative multi-agent learning algorithm

Input 1: the number of the agent is n , the state of each agent is $(\overleftarrow{S}, \overrightarrow{S})$ (1:n), the action set of each agent is $A(1:n)$, the rewarding value of each agent is $(\overleftarrow{R}, \overrightarrow{R})$ (1:n), $A'(1..m)$ represent the executable action list of each agent in time t .

```

for each agent i:  $1 \leq i \leq n$ 
  Generate the action list  $A'(1..m)$  which can be performed
   $(\overleftarrow{R}_i, \overrightarrow{R}_i) = (\overleftarrow{0}, \overrightarrow{0}), A_i = \varphi;$ 
  for  $j = 1$  to  $m$  do
    perform action  $A'_j$ ;
    get the reward  $(\overleftarrow{r}_j, \overrightarrow{r}_j)$ ;
    if  $(\overleftarrow{r}_j, \overrightarrow{r}_j) > (\overleftarrow{R}_i, \overrightarrow{R}_i)$ 
      then  $(\overleftarrow{R}_i, \overrightarrow{R}_i) = (\overleftarrow{r}_j, \overrightarrow{r}_j), A_i = A'_j$ 
    endif
  endfor
endfor

```

The above learning model does not take full advantage of the collaborative capacity of multiple agents to improve the learning speed. Now, we build a novel multi-agent collaborative learning model. In this new model, we assume there are n agents who learn together and execute some missions in parallel. To coordinate the learning tasks of n agents, the system has a special comprehensive agent, or agent manager. In Fig. 7.5, the double-sided arrow represents the communicating mechanism between agent i and the agent manager.

Agents $1 - n$ have the same structure. Each agent consists of three parts, a learner (L), motion converter (C), and state perceptron, and is connected with the agent manager. Normal agents can communicate with the agent manager, but there is no direct connection between them.

As well as these three basic modules, each agent has a status buffer that saves some intermediate results.

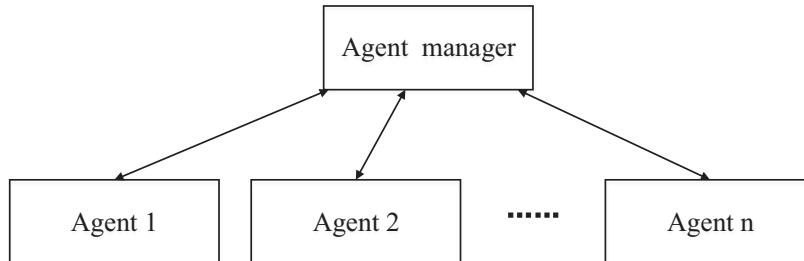


Fig. 7.5: The comprehensive cooperative multi-agent learning model.

All agents, including the agent manager, have an identifier defined by the numbers $1 - n$. Moreover, the strategy database and state of all the agents are the same.

The state perceptron of the agent manager estimates the action taken by each normal agent based on the current state information. Then, the agent manager combines these actions and forms an action list that is sent to every agent.

We set the initial reward value of agent i ($1 \leq i \leq n$) to be $(\bar{r}_i, \bar{r}_i) = (\overleftarrow{0}, \overrightarrow{0})$ and the other executable action a_i to be null. When agent i receives the agent list from the agent manager, it executes action j in sequence. Action j meets the requirement that $j \bmod n = i$. During the execution, agent i saves the execution records. If $(\bar{r}'_i, \bar{r}'_i) > (\bar{r}_i, \bar{r}_i)$, we set $(\bar{r}'_i, \bar{r}'_i) = (\bar{r}_i, \bar{r}_i)$, $a = a'_i$. When the agent has executed the action, it obtains the reward value (\bar{r}'_i, \bar{r}'_i) and sends this value to the agent manager.

When the agent manager has received all reward value information from the normal agents, it chooses the maximal value and notifies every agent that this value has been obtained. Without loss of generality, we set the number of agents to be ($1 \leq u \leq n$). Agent u transmits the action corresponding to the maximal reward value to the agent manager. The agent manager notifies every agent using the broadcast mode. Each agent updates its own strategy database. By this method, each agent has learned the strategy corresponding to the maximal reward value. Fig. 7.6 shows the algorithm flowchart.

We can see from this figure that there is a processing module, called the module of agent i . The main work of this module is the corresponding management tasks when agent i receives the action list. The processing flowchart of this module is shown in Fig. 7.7.

In Algorithm 7.9, we can see that there are n agents and m executable actions. Each agent has to execute k actions. If the time required to execute action a_j ($1 \leq j \leq m$) is t_j and $t_{\max} = \max(t_1, t_2, \dots, t_m)$, the execution time for all actions of agent i will be $t_i \leq k^* t_{\max}$. In addition, the agent manager will provide the action list and send it to every normal agent in broadcast mode. When the agent manager receives the reward values from the normal agents, it sends a notification that the time of the

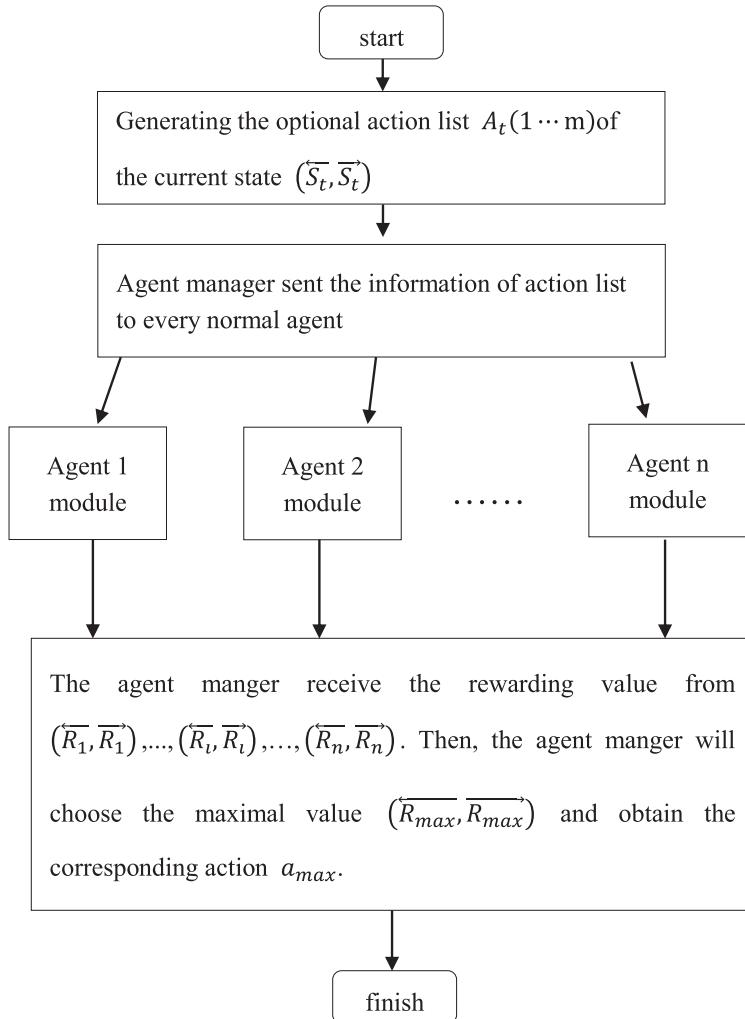


Fig. 7.6: Algorithm flowchart of the comprehensive cooperative multi-agent learning algorithm.

action corresponding to a strategy is t_{else} . Each agent takes appropriate action when it receives the information of the maximal reward value. It is easy to see that $t_{re} \leq t_{max}$, so the time complexity of the algorithm is $T_2 \leq k^* t_{max} + t_{else} + t_{max} \leq (k+1)^* t_{max} + t_{else}$.

Example 7.5 We now analyse the comprehensive cooperative multi-agent learning algorithm. We take a classic machine learning problem, Sudoku, as an example.

In this example, we assume that there is an agent manager and four normal agents, numbered 1–4. The initial state is S_0 and the target state is S_g in Fig. 7.8.

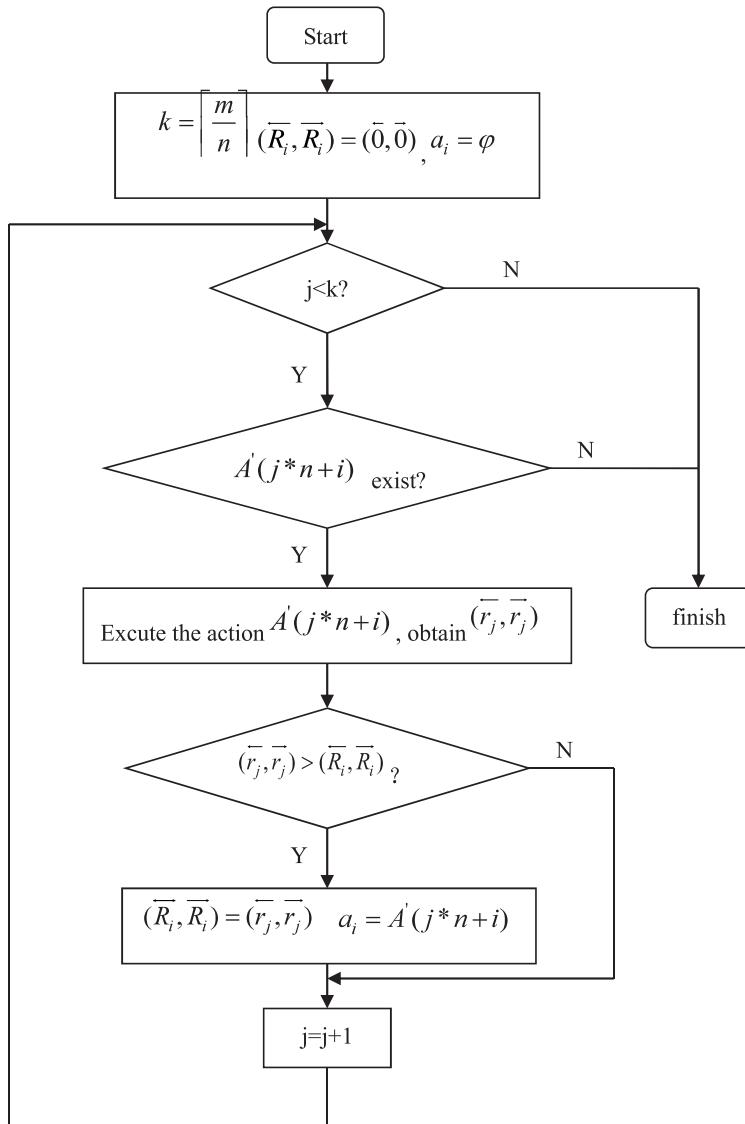


Fig. 7.7: Flowchart of the normal agent i .

In this learning system, the computational method of the reward value is defined as follows: assume that a represents the different bits between the initial state and the target state and b represents the current depth. If the current state is not the target state, the reward value will be $r = (10 - a - b)/10$. If the different bits of the current state become the target state by proper angle rotation, the reward value will be \vec{r} ; otherwise, it will be \vec{r} . The action sequence of this problem consists of four directions,

Algorithm 7.9 The comprehensive cooperative multi-agent learning algorithm

Input: the number of the agent is n , the action set of each agent is $A(1, n)$, the rewarding value of each agent is (\vec{R}, \vec{R}) , $A'(1..m)$ represent the executable action list of each agent in time t . agent Manager generate the action list $A'(1..m)$ to each agent in the multi-agent system.

```

for each agent  $i: 1 \leq i \leq n$ 
   $r_i = 0, a_i = \varphi;$ 
  for  $j = 0: k - 1$  do
    if  $(\vec{r}_j, \vec{r}_j) > (\vec{R}_i, \vec{R}_i)$ , get the reward  $(\vec{r}_j, \vec{r}_j)$ 
      then  $\{\vec{R}_i, \vec{R}_i = (\vec{r}_j, \vec{r}_j), A_i = A'_{j+n+i}\}$ 
    endif
  endfor
  send the reward  $(\vec{R}_i, \vec{R}_i)$  to the agent Manager;
  the agent Manager compare  $(\vec{R}, \vec{R})$  and choose the largest reward;
  the agent Manager tells the agent (let it be agent  $u, 1 \leq u \leq n$ ) who
  gets the largest reward;
  the agent  $u$  sends its action  $A_u$  to the agent Manager;
  the agent Manager sends  $A_u$  to the other agents in the system;
  for each agent  $1 \leq i \leq n$  and  $i \neq u$  pardo
     $A_i = A_u;$ 
  endfor
endfor

```

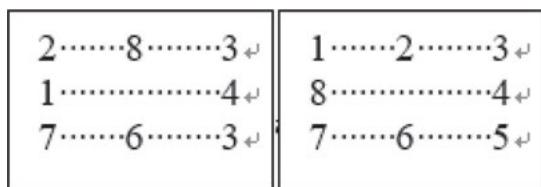


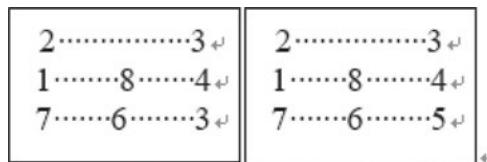
Fig. 7.8: The initial state and the target state.

up, down, left, and right. Based on the above notation and preparation, we will solve the following problem.

- (1) The agent manager obtains the action sequence (up, down, left, right) based on the initial state. This action sequence is sent to the four normal agents.
- (2) Agents 1–4 conduct the four actions, respectively, and obtain four different states. The reward values are presented in Tab. 7.8.
- (3) The agent manager receives the four reward values and chooses the maximal value R_1^1 . Then, the agent manager tells agent 1 to send its own action to the manager and to the other agents. Now, all the agents have the same action $A_1 = \text{up}$ and state S_1 in Fig. 7.9. Thus, the system has a new state. The agent manager then obtains the new action set (down, left, right) based on the current state S_1 . Similar to the above method, the system will continue to search in Tabs. 7.9–7.11

Tab. 7.8: Reward values of agents in state S_1 .

	Agent 1	Agent 2	Agent 3	Agent 4																																			
S_1	<table border="1"> <tr><td>2</td><td>3</td></tr> <tr><td>1</td><td>8</td><td>4</td></tr> <tr><td>7</td><td>6</td><td>5</td></tr> </table>	2	3	1	8	4	7	6	5	<table border="1"> <tr><td>2</td><td>8</td><td>3</td></tr> <tr><td>1</td><td>6</td><td>4</td></tr> <tr><td>7</td><td></td><td>5</td></tr> </table>	2	8	3	1	6	4	7		5	<table border="1"> <tr><td>2</td><td>8</td><td>3</td></tr> <tr><td>1</td><td></td><td>4</td></tr> <tr><td>7</td><td>6</td><td>5</td></tr> </table>	2	8	3	1		4	7	6	5	<table border="1"> <tr><td>2</td><td>8</td><td>3</td></tr> <tr><td>1</td><td></td><td>4</td></tr> <tr><td>7</td><td>6</td><td>5</td></tr> </table>	2	8	3	1		4	7	6	5
2	3																																						
1	8	4																																					
7	6	5																																					
2	8	3																																					
1	6	4																																					
7		5																																					
2	8	3																																					
1		4																																					
7	6	5																																					
2	8	3																																					
1		4																																					
7	6	5																																					
R_1	$\overrightarrow{0.5}$	$\overleftarrow{0.4}$	$\overleftarrow{0.5}$	$\overleftarrow{0.4}$																																			

**Fig. 7.9:** The state S_1 and the state S_2 .**Tab. 7.9:** Reward values of agents in state S_2 .

	Agent 1	Agent 2	Agent 3	Agent 4																									
S_2	<table border="1"> <tr><td>2</td><td>8</td><td>3</td></tr> <tr><td>1</td><td></td><td>4</td></tr> <tr><td>7</td><td>6</td><td>5</td></tr> </table>	2	8	3	1		4	7	6	5	<table border="1"> <tr><td>2</td><td>3</td></tr> <tr><td>1</td><td>8</td><td>4</td></tr> <tr><td>7</td><td>6</td><td>5</td></tr> </table>	2	3	1	8	4	7	6	5	<table border="1"> <tr><td>2</td><td>3</td></tr> <tr><td>1</td><td>8</td><td>4</td></tr> <tr><td>7</td><td>6</td><td>5</td></tr> </table>	2	3	1	8	4	7	6	5	Null
2	8	3																											
1		4																											
7	6	5																											
2	3																												
1	8	4																											
7	6	5																											
2	3																												
1	8	4																											
7	6	5																											
R_2	$\overrightarrow{0.4}$	$\overrightarrow{0.5}$	$\overrightarrow{0.5}$	0																									

Tab. 7.10: Reward values of agents in state S_3 .

	Agent 1	Agent 2	Agent 3	Agent 4																
S_3	<table border="1"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>8</td><td>4</td></tr> <tr><td>7</td><td>6</td><td>5</td></tr> </table>	1	2	3	8	4	7	6	5	<table border="1"> <tr><td>2</td><td>3</td></tr> <tr><td>1</td><td>8</td><td>4</td></tr> <tr><td>7</td><td>6</td><td>5</td></tr> </table>	2	3	1	8	4	7	6	5	Null	Null
1	2	3																		
8	4																			
7	6	5																		
2	3																			
1	8	4																		
7	6	5																		
R_3	$\overrightarrow{0.5}$	$\overrightarrow{0.3}$	0	0																

and obtains $A_2 = \text{left}$, State S_2 in Fig. 7.9. The agent manager will obtain the new action sequence (down, right).

Finally, we obtain $A_4 = \text{right}$ and state S_4 . Because $S_4 = S_g$, the system has reached the target state. The system terminates the learning process.

In the above example, there are two options in the process of converting state S_0 into S_1 without considering the dynamic fuzzy property. If the dynamic fuzzy property was considered, the search speed would be accelerated.

Tab. 7.11: Reward values of agents in state S_3 .

	Agent 1	Agent 2	Agent 3	Agent 4																								
S_4	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>2</td><td>3</td></tr> <tr><td>1</td><td>8</td><td>4</td></tr> <tr><td>7</td><td>6</td><td>5</td></tr> </table>	2	3	1	8	4	7	6	5	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>7</td><td>8</td><td>4</td></tr> <tr><td>6</td><td>5</td></tr> </table>	1	2	3	7	8	4	6	5	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>8</td><td>4</td></tr> <tr><td>7</td><td>6</td><td>5</td></tr> </table>	1	2	3	8	4	7	6	5	Null
2	3																											
1	8	4																										
7	6	5																										
1	2	3																										
7	8	4																										
6	5																											
1	2	3																										
8	4																											
7	6	5																										
R_4	$\overrightarrow{0.3}$	$\overrightarrow{0.3}$	1	0																								

Example 7.6 This experiment will simulate a multi-agent system. During the learning process of the system, there are some optional actions numbered 1–4. These actions are selectable when the system is being converted to a new state. The execution time of each action is from 0 to 9 s. Now, we randomly generate 30 initial states. Each initial state can attain the target state. We employ a heuristic method and comprehensive multi-agent learning method. The results are presented in Tabs. 7.12–7.14 and Fig. 7.10.

It can be seen from Fig. 7.10 that the time complexity of the comprehensive multi-agent learning method is far less than the heuristic method for larger numbers of experiments. Moreover, if the initial state repeatedly emerges or is an intermediate state, the time complexity of the learning will be zero, which means no work is required.

In the above algorithm, each agent is based on the real-time single-agent learning algorithm in a multi-agent system. However, Example 7.5 indicates that we can find a

Tab. 7.12: Execution time of action in state 1–10.

State	1	2	3	4	5	6	7	8	9	10
Time of heuristic method(s)	0	5	3	6	2	3	3	6	3	3
Time of comprehensive multi-agent(s)	1	6	6	0	0	3	0	0	0	3

Tab. 7.13: Execution time of action in state 11–20.

State	11	12	13	14	15	16	17	18	19	20
Time of heuristic method(s)	3	6	3	2	4	0	3	9	3	1
Time of comprehensive multi-agent(s)	0	0	0	0	0	0	0	9	0	0

Tab. 7.14: Execution time of action in state 21–30.

State	21	22	23	24	25	26	27	28	29	30
Time of heuristic method(s)	3	3	3	1	4	2	9	0	1	0
Time of comprehensive multi-agent(s)	0	0	0	1	0	0	0	0	0	0

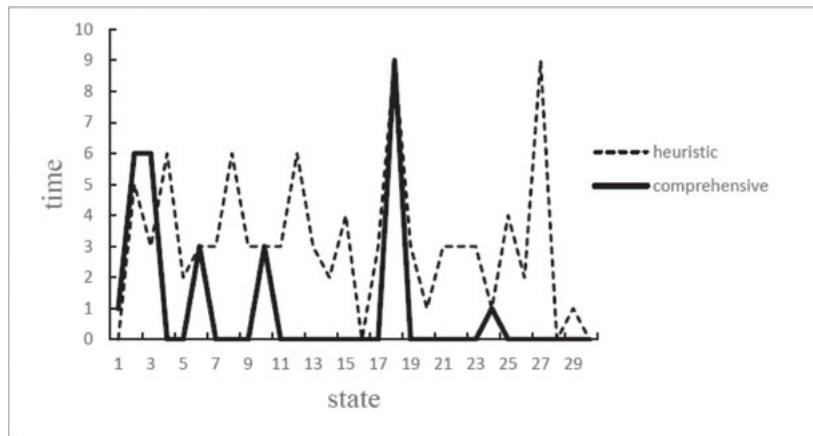


Fig. 7.10: The execution time of action from state 1 to 30.

strategy to solve the problem using the single-agent learning algorithm, although this strategy is not optimal. Therefore, we need to employ the optimal strategy to construct the multi-agent system. Here, we employ Q-learning based on DFL to construct the cooperative multi-agent learning system.

The Q-learning method has to consider that future actions will affect the current Q-value. Therefore, we employ the topological structure of the previous comprehensive multi-agent learning system. However, the internal structure of the agents is more complex. Each agent not only has the necessary intention database and the capacity database of the mental model, but also needs a stack and a queue. The stack is used to store the Q information of the agent to be processed. The queue is used to update the Q information for which each agent is responsible. The non-real-time cooperative multi-agent learning model is shown in Fig. 7.11.

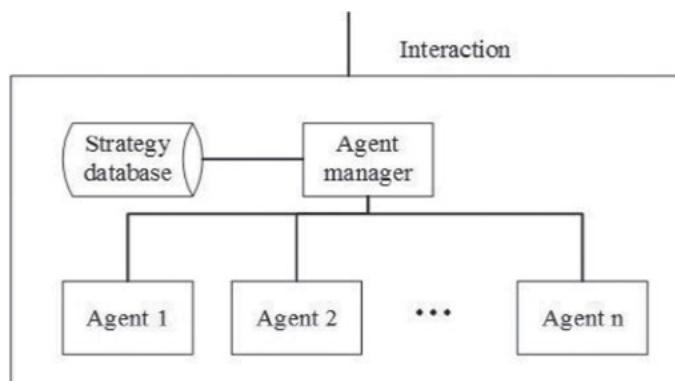


Fig. 7.11: The non-real-time cooperative multi-agent learning model.

The system has a comprehensive agent and several normal agents. These agents are arranged in a star-topology, which means that each agent can communicate with the other agents. Each agent has a database associated with its own mental state, for example, the belief database and the action database. To store some learning strategies, the system has a public strategy database. Employing the public strategy database, the system can directly call the corresponding strategy when the system encounters a similar state, thus avoiding any additional learning process.

Besides handling the overall planning of the system, the comprehensive agent acts as an interactive interface. Interacting with other agents through the comprehensive agent, the system then decides whether to conduct a learning process or provide the corresponding learning strategy to another agent system that needs to learn. The normal agents mainly depend on the overall planning of the comprehensive agent and learn the new strategy. At the same time, the normal agents will insert the latest strategy in their own databases. This updating method will help the system to perform well in the next learning process.

In general, the comprehensive agent gives corresponding instructions when it has received a specific message. Therefore, besides the basic function module of the mental state of the agent, it needs a message interrupting module. The interrupting module has a relatively simple function. The main algorithm flowchart is shown in Fig. 7.12.

However, the normal agent makes the corresponding action when it receives the instruction from the comprehensive agent. Without receiving the message, the normal agent also needs to conduct some learning process. The internal structure of the normal agent is relatively simple, which is shown in Fig. 7.13.

The learning module mainly conducts the learning task in the stack, which stores the learning task. The algorithm is show in Algorithm. 4.4 and the algorithm flow chart is show in Fig. 7.14.

The message interrupting module mainly conducts the message received by the agent. During the processing of the learning task, other agents would like to send some task instructions and learning results to the current agent. The message interrupting module needs to make corresponding response when it receives some related message. The algorithm flow chart is shown in Fig. 7.15.

Algorithm 7.10 Algorithm of the message interrupting module of the agent manager

```

procedure message_manage_agent(message)
if (the Q of the message doesn't exist in the queue)
    put it and the agent id who sends it into the queue and send it to one of the agents
else
    add the agent id into the queue
endif

```

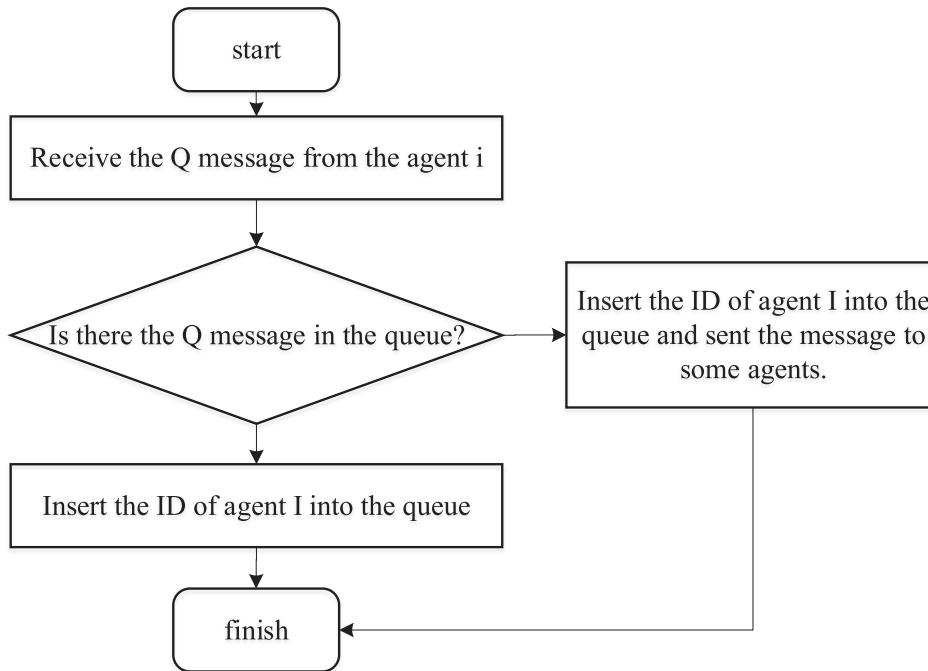


Fig. 7.12: Flowchart of message interrupting module.

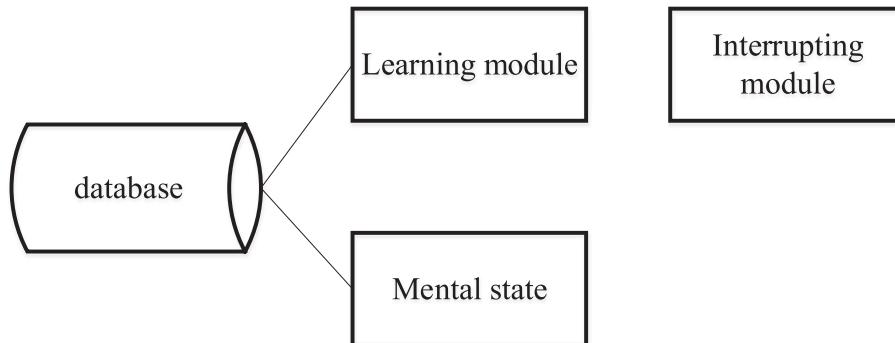


Fig. 7.13: The normal agent.

It can be seen from the above algorithm that Q-learning is a kind of recursive approach. During the computing of $Q((\overleftarrow{s}, \overrightarrow{s}), a)$, we need to compute the Q-value of the follow-up action a . Therefore, if we need to compute the Q-value m times to obtain the final result and the time complexity of each computation is t , the total time complexity

Algorithm 7.11 The algorithm of the learning module of the normal agent

```

procedure agent_learning (n( $\vec{S}$ ,  $\vec{S}$ ), A, ( $\vec{R}$ ,  $\vec{R}$ ))
for each agent  $i$ :  $1 \leq i \leq n$ 
    if (the stack is empty)
        break;
    else
        exceed(stack);
        put the Q into the queue;
        if (the Q which should be calculated can't be calculated)
            send the Qs to the manage agent;
        else
            calculate the Q and send it to the agent who sends the Q;
    endif
endif
endfor

```

Algorithm 7.12 The algorithm of the message interrupting module of the normal agent

```

procedure message_normal_agent(message)
if (the message is the Q which should be calculated)
    put the information of the Q into the stack;
    endif
if (the message is the result of the Q which has been calculated)
    update the queue;
    if (the updated result of the Q is the needed result)
        send it to the agent who sends the Q;
endif
endif

```

of the algorithm is $T = t^* \lceil m/n \rceil$. The data structure of the queue node in the agent manager is defined as follows:

Q(state, *) agent ID

Here, Q(state, *) is the ID of the queue node. The agent ID is needed to define, for example, agent 1, agent 2, or agent 3. Hence, the agent ID of the field is 1, 2, or 3. The data structure of the queue node in the normal agent is defined as follows:

Q(state, *) Q value Action agent ID counter Is it latest?

Here, Q(state, *) is the ID of the queue node. The Q-value is used to store the updated Q-value. Action is used to store the action that obtains the Q-value. The counter is used to count whether all actions have been calculated.

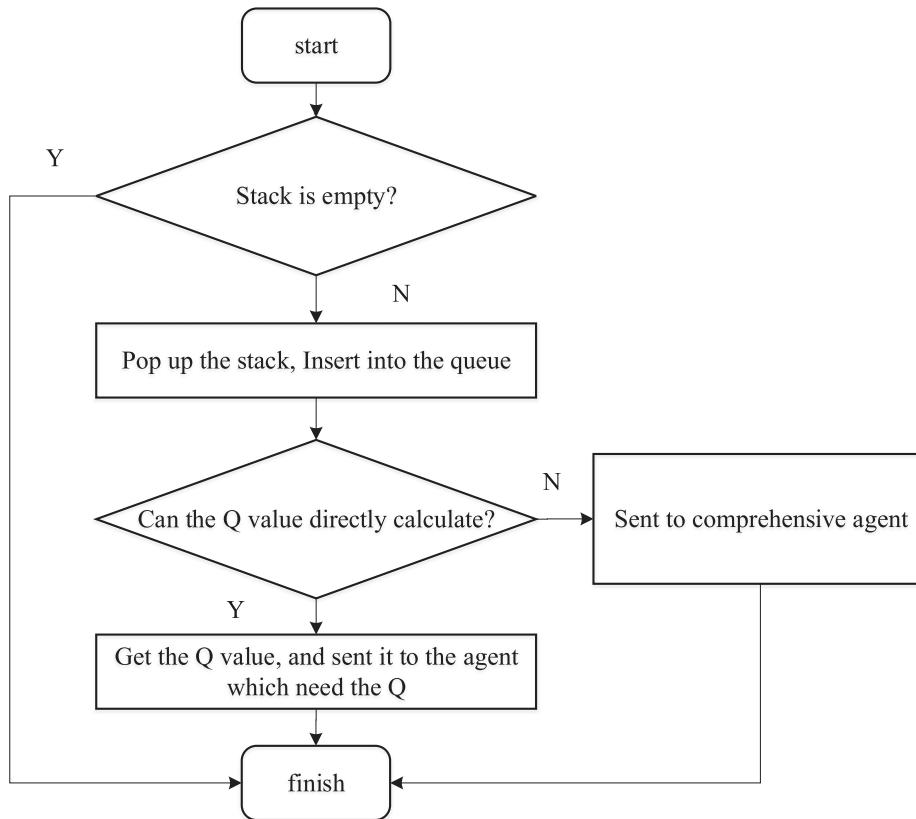


Fig. 7.14: Flowchart of the learning module of the normal agent.

“Is it latest?” identifies whether the Q-value has been fully updated. If the Q-value has been updated, this value should be T. Otherwise, the value should be F. The data structure of the stack node in the normal agent is defined as follows. Here, $Q(state, *)$ is the ID of the stack node.

$Q(state, *)$ agent ID

Example 7.7 The initial setting of this example is the same as in Example 7.2. We explain Algorithms 4.3–4.5 in detail by solving the problem. First, we assume that the multi-agent system has four agents, numbered 1–4.

- (1) Agents 1–4 initialize their stacks. The agent manager sends $Q(1, *)$ to one agent.
- (2) Select one agent at random to receive $Q(1, *)$. In general, we set agent 1 to receive this information and push the value onto the stack of agent 1.

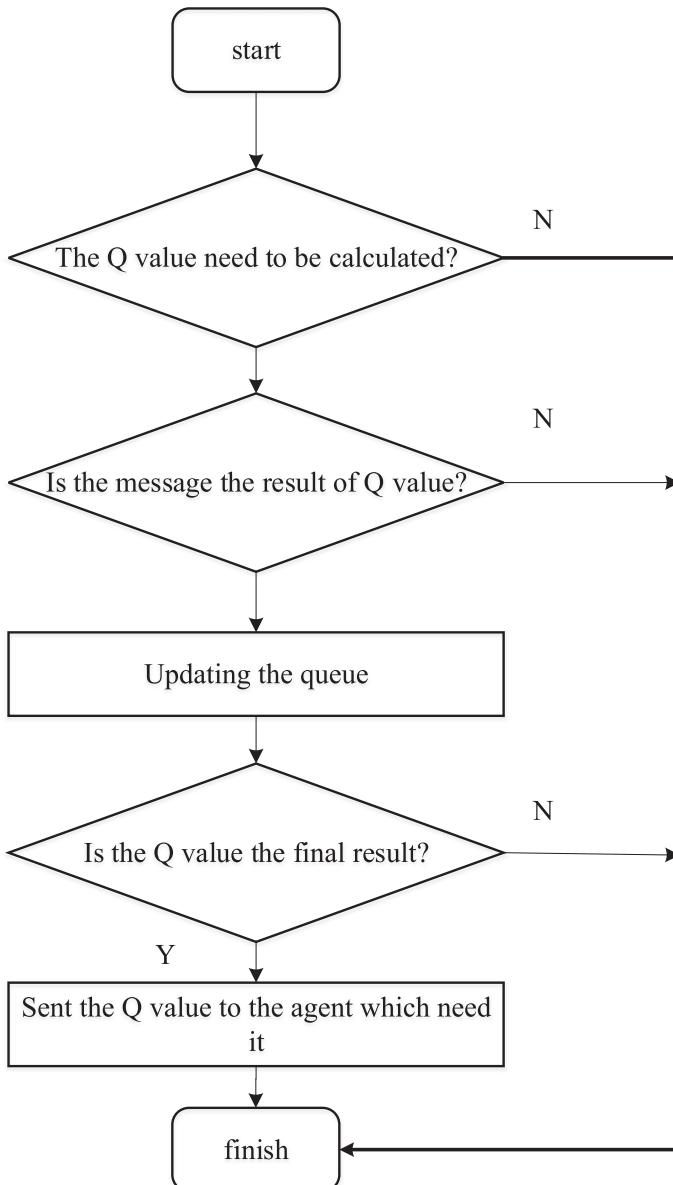


Fig. 7.15: Flowchart of the message interrupting module of the normal agent.

- (3) All agents pop the pending nodes off their stacks. Agent 1 conducts the calculations for $Q(1, *)$ and places the result into the corresponding queue. Agent 1 then sends $Q(2, *), Q(3, *), Q(4, *),$ and $Q(5, *)$ to the agent manager. The agent manager will judge whether these Q-values should be sent to the specific agents.

If these Q-values are not sent to the specific agents, the agent manager will resend the information. Without loss of generality, we assume that agent i needs to conduct $Q(i + 1, *)$, $1 \leq i \leq 4$. When the other agents have no node calculations to conduct, the stacks of agents 1–4 contain $Q(2, *)$, $Q(3, *)$, $Q(4, *)$ and $Q(5, *)$, respectively.

- (4) When all agents pop the pending nodes off their stacks, agent i places $Q(i + 1, *)$ into the corresponding queue. Agent i sends the corresponding Q-value to the agent manager. The agent manager coordinates all the Q-values and sends them to each agent. At this point, the queue and stack of each agent are as presented in Tabs. 7.15 and 7.16.
- (5) All agents pop the pending nodes off their stacks and put them into the corresponding queues. The normal agents send the Q-values to the agent manager. The agent manager coordinates all the Q-values and sends them to each agent. The queue and stack of each agent are as given in Tabs. 7.17 and 7.18.
- (6) All agents pop the pending nodes off their stacks and put them into the corresponding queues. The normal agents send the Q-values to the agent manager. The agent manager coordinates all the Q-values and sends them to each agent. The queue and stack of each agent are as given in Tabs. 7.19 and 7.20.
- (7) All the agents pop the pending nodes off their stacks and put them into the corresponding queues. At this moment, the system will instantly compute the results and send them to agents 1–3.
- (8) When we have $Q(12, *)$, agents 1–3 calculate $Q(9, *)$, $Q(10, *)$, and $Q(11, *)$, respectively. These results are then sent to the agents that need them.
- (9) Each agent calculates $Q(6, *)$, $Q(7, *)$, and $Q(8, *)$ based on the current received Q-value and sends the results to the agents that need them.
- (10) Each agent calculates $Q(2, *)$, $Q(3, *)$, $Q(4, *)$, and $Q(5, *)$ based on the current received Q-value and sends the results to the agents that need them.
- (11) Agent 1 calculates $Q(1, *)$ and obtains the result.
- (12) Finished.

Tab. 7.15: Queue and stack of each agent.

Agent ID	Stack (bottom ↔ top)	Queue (head ↔ tail)
Agent 1	$Q(6, *)$	$Q(1, *), Q(2, *)$
Agent 2	$Q(7, *)$	$Q(3, *)$
Agent 3	$Q(8, *)$	$Q(4, *)$
Agent 4		$Q(5, *)$

Tab. 7.16: Queue of the agent manager.

$(Q(1, *), (1))$, $(Q(2, *), (1))$, $(Q(3, *), (1))$, $(Q(4, *), (1))$, $(Q(5, *), (1))$ $(Q(6, *), (1, 2))$, $(Q(7, *), (1, 2, 4))$, $(Q(8, *), (1, 3, 4))$

Tab. 7.17: Queue and stack of each agent.

Agent ID	Stack (bottom ↔ top)	Queue (head ↔ tail)
Agent 1	$Q(9, *)$	$Q(1, *), Q(2, *), Q(6, *)$
Agent 2	$Q(10, *)$	$Q(3, *), Q(7, *)$
Agent 3	$Q(11, *)$	$Q(4, *), Q(8, *)$
Agent 4		$Q(5, *)$

Tab. 7.18: Queue of the agent manager.

$(Q(1, *), (1)), (Q(2, *), (1)), (Q(3, *), (1)), (Q(4, *), (1)), (Q(5, *), (1))$
 $(Q(6, *), (1, 2)), (Q(7, *), (1, 2, 4)), (Q(8, *), (1, 3, 4))$
 $(Q(9, *), (1, 2)), (Q(10, *), (1, 2, 3)), (Q(11, *), (3))$

Tab. 7.19: Queue and stack of each agent.

Agent ID	Stack (bottom ↔ top)	Queue (head ↔ tail)
Agent 1	$Q(12, *)$	$Q(1, *), Q(2, *), Q(6, *), Q(9, *)$
Agent 2		$Q(3, *), Q(7, *), Q(10, *)$
Agent 3		$Q(4, *), Q(8, *), Q(11, *)$
Agent 4		$Q(5, *)$

Tab. 7.20: Queue of the agent manager.

$(Q(1, *), (1)), (Q(2, *), (1)), (Q(3, *), (1)), (Q(4, *), (1)), (Q(5, *), (1))$
 $(Q(6, *), (1, 2)), (Q(7, *), (1, 2, 4)), (Q(8, *), (1, 3, 4))$
 $(Q(9, *), (1, 2)), (Q(10, *), (1, 2, 3)), (Q(11, *), (3))$
 $(Q(12, *), (1, 2, 3))$

7.4.3 Competitive multi-agent learning algorithm based on DFL

The relationship between the agents not only has a cooperative attribute but also has a competitive attribute. Therefore, when discussing whether the learning strategy of an agent system is optimal, we need to consider the accumulated reward value of the competitive multi-agents. The learning algorithm is introduced below.

Definition 7.6 In the multi-agent system, there is a kind of competitive relationship between P and Q. It is assumed that, at some time, the multi-agent system is left in state $(\overline{S}_{at}, \vec{S}_{at})$, and the system can takeaction a_t . Moreover, system Q can take

action b_t . The accumulated reward value of the multi-agent system is calculated as follows:

$$V(\overleftarrow{S_{at}}, \overrightarrow{S_{at}}) = \max_{a_t \in A} \min_{b_t \in B} Q((\overleftarrow{S_{at}}, \overrightarrow{S_{at}}), a_t, b_t). \quad (7.19)$$

Here, A represents the action set of system P at this moment and B represents the action set of system Q.

7.5 Summary

Agent learning is a hot topic in many academic fields. The learning model of each agent follows a kind of dynamic fuzzy process. However, previous approaches do not solve dynamic fuzzy problems well. Therefore, this chapter has presented a kind of multi-agent learning model. The proposed model includes the mental model, immediate return single-agent learning algorithm, non-real-time cooperative multi-agent learning model, cooperative multi-agent learning algorithm, and competitive multi-agent learning algorithm. The proposed model employs DFL theory to solve problems using agent learning.

The innovative points of this chapter include the following:

- (1) We proposed a structure for the agent mental model based on DFL. The model structure is the foundation for researching agent learning.
- (2) We proposed an immediate return single-agent learning algorithm and non-immediate return single-agent algorithm based on DFL. These methods are the foundation of further research on agent learning.
- (3) Finally, we proposed a multi-agent learning model based on DFL. From the multi-agent system, we developed cooperative and competitive multi-agent learning algorithms based on different relationships between the agents.

References

- [1] Tan M. Multi-agent reinforcement learning: independent vs. cooperative agents. In Proceedings of Tenth International Conference on Machine Learning. San Francisco, Morgan Kaufmann, 1993, 487–494.
- [2] Pan G. A framework for distributed reinforcement learning. In Proceedings of Adaption and learning in multi-agent system. Lecture Notes in Computer Sciences, NY, Springer, 1996, 97–112.
- [3] Littman ML. Markov games as a framework for multi-agent reinforcement learning. In Proceedings of Eleventh International Conference on Machine Learning. New Brunswick, NJ, Morgan Kaufmann, 1994, 157–163.
- [4] Roy N, Pineau J, Thrun S. Spoken dialogue management using probabilistic reasoning. In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics, 2000.

- [5] Peter S, Manuela V. Team-partitioned, opaque-transition reinforcement learning. In Proceedings of the 3rd International Conference on Autonomous Agents. Seattle, ACM Press, 1999, 206–212.
- [6] <http://www.intel.com/research/mrl/pnl>.
- [7] Li FZ. Description of differential learning conception. *Asian Journal of Information Technology*, 2005, 4(3): 16–21.
- [8] Tong HF. Design of auto-reasoning platform of multi-agent based on DFL. Master's Thesis, Soochow University, Suzhou, China, 2005.
- [9] Xie LP. Research on Multi-agent learning model based on DFL. Master's Thesis, Soochow University, Suzhou, China, 2006.
- [10] Xie LP, Li FZ. The multi-agent learning model based on dynamic fuzzy logic. *Journal of Communication and Computer*, 2006, 3(3): 87–89.
- [11] Bradshaw M. An introduction to software agents. Menlo Park, CA, MIT Press, 1997.
- [12] Singh MP. Multi-agent systems: a theoretical framework for intention, know-how, and communication. Berlin, Germany: Springer-Verlag, 1994.
- [13] Wooldridge M, Jennings NR. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 1995, 10(2): 115–152.
- [14] Hu SL, Shi CY. Agent-BDI logic. *Journal of Software*, 2000, 11(10): 1353–1360.

8 Appendix

8.1 Dynamic fuzzy sets

8.1.1 Definition of dynamic fuzzy sets

Definition 8.1 A map defined in the domain of discourse U . $(\overleftarrow{A}, \overrightarrow{A}) : (\overleftarrow{U}, \overrightarrow{U}) \rightarrow [0, 1] \times [\leftarrow, \rightarrow]$, $(\overleftarrow{u}, \overrightarrow{u}) \mapsto (\overleftarrow{A}(\overleftarrow{u}), \overrightarrow{A}(\overrightarrow{u}))$, marked as $(\overleftarrow{A}, \overrightarrow{A}) = \overleftarrow{A}$ or \overrightarrow{A} . Then we call $(\overleftarrow{A}, \overrightarrow{A})$ as the dynamic fuzzy sets in $(\overleftarrow{U}, \overrightarrow{U})$, shorten for DFS. We call $(\overleftarrow{A}(\overleftarrow{u}), \overrightarrow{A}(\overrightarrow{u}))$ as the membership function of membership degree $(\overleftarrow{A}, \overrightarrow{A})$.

*Explanation: For any $a \in [0, 1]$, we can make it dynamic fuzzilization as $a \stackrel{DF}{=} (\overleftarrow{a}, \overrightarrow{a})$, $a \stackrel{DF}{=} \overleftarrow{a}$ or \overrightarrow{a} , $\max(\overleftarrow{a}, \overrightarrow{a}) \stackrel{\Delta}{=} \overrightarrow{a}$, $\min(\overleftarrow{a}, \overrightarrow{a}) \stackrel{\Delta}{=} \overleftarrow{a}$. So that we can visually present variation trends of number a .

In the domain of discourse U , there are many dynamic fuzzy sets. We mark the entire dynamic fuzzy sets as $DF(U)$, namely:

$$\begin{aligned} DF(U) &= \{(\overleftarrow{A}, \overrightarrow{A}) | (\overleftarrow{A}, \overrightarrow{A}), (\overleftarrow{u}, \overrightarrow{u}) \mapsto [0, 1] \times [\leftarrow, \rightarrow]\} \\ &= \{(A \times (\leftarrow, \rightarrow)) | (A \times (\leftarrow, \rightarrow)), (u \times (\leftarrow, \rightarrow)) \mapsto [0, 1] \times [\leftarrow, \rightarrow]\}. \end{aligned}$$

If domain of discourse U is time varied domain, then we define it as U_T (there T means time). Correspondingly, we note $DF(U_T)$ as

$$\begin{aligned} DF(U_T) &= \{(\overleftarrow{A}_t, \overrightarrow{A}_t) | (\overleftarrow{A}_t, \overrightarrow{A}_t), (\overleftarrow{u}, \overrightarrow{u}) \mapsto [0, 1] \times [\leftarrow, \rightarrow]\} \\ &= \{(A_t \times (\leftarrow, \rightarrow)) | (A_t \times (\leftarrow, \rightarrow)), (u \times (\leftarrow, \rightarrow)) \mapsto [0, 1] \times [\leftarrow, \rightarrow]\} \end{aligned}$$

(there $t \in T$).

8.1.2 Operation of dynamic fuzzy sets

Operations between two DF subsets can be understood as the corresponding operation of membership function, we have the following definition: “ \forall ” means “any”, “ \exists ” means “exist”.

Definition 8.2 Assume $(\overleftarrow{A}, \overrightarrow{A})$ and $(\overleftarrow{B}, \overrightarrow{B}) \in DF(U)$, if $\forall(\overleftarrow{u}, \overrightarrow{u}) \in U$ and $(\overleftarrow{B}, \overrightarrow{B})(\overleftarrow{u}, \overrightarrow{u}) \subseteq (\overleftarrow{A}, \overrightarrow{A})(\overleftarrow{u}, \overrightarrow{u})$, then $(\overleftarrow{A}, \overrightarrow{A})$ contains $(\overleftarrow{B}, \overrightarrow{B})$, we note that $(\overleftarrow{B}, \overrightarrow{B}) \subseteq (\overleftarrow{A}, \overrightarrow{A})$;
If $(\overleftarrow{A}, \overrightarrow{A}) \subseteq (\overleftarrow{B}, \overrightarrow{B})$, and $(\overleftarrow{B}, \overrightarrow{B}) \subseteq (\overleftarrow{A}, \overrightarrow{A})$, then $(\overleftarrow{A}, \overrightarrow{A}) = (\overleftarrow{B}, \overrightarrow{B})$.

Obviously, contain relationship “ \subseteq ” is a kind of relationship on DF power set U , with the follow property:

(1) Reflexivity

$$\forall (\overleftarrow{A}, \overrightarrow{A}) \in DF(U), (\overleftarrow{A}, \overrightarrow{A}) \subseteq (\overleftarrow{A}, \overrightarrow{A});$$

(2) Anti-symmetry

$$\text{If } (\overleftarrow{A}, \overrightarrow{A}) \subseteq (\overleftarrow{B}, \overrightarrow{B}), (\overleftarrow{B}, \overrightarrow{B}) \subseteq (\overleftarrow{A}, \overrightarrow{A}), \Rightarrow (\overleftarrow{A}, \overrightarrow{A}) = (\overleftarrow{B}, \overrightarrow{B});$$

(3) Transitivity

$$\text{If } (\overleftarrow{A}, \overrightarrow{A}) \subseteq (\overleftarrow{B}, \overrightarrow{B}), (\overleftarrow{B}, \overrightarrow{B}) \subseteq (\overleftarrow{C}, \overrightarrow{C}), \Rightarrow (\overleftarrow{A}, \overrightarrow{A}) \subseteq (\overleftarrow{C}, \overrightarrow{C}).$$

Definition 8.3 Assume $(\overleftarrow{A}, \overrightarrow{A}), (\overleftarrow{B}, \overrightarrow{B}) \in DF(U)$, we refer the operation $(\overleftarrow{A}, \overrightarrow{A}) \cup (\overleftarrow{B}, \overrightarrow{B}), (\overleftarrow{A}, \overrightarrow{A}) \cap (\overleftarrow{B}, \overrightarrow{B})$ as the union and intersection of $(\overleftarrow{A}, \overrightarrow{A}), (\overleftarrow{B}, \overrightarrow{B})$. $(\overleftarrow{A}, \overrightarrow{A})^c$ is the complement of $(\overleftarrow{A}, \overrightarrow{A})$. Their membership function is

$$\begin{aligned} ((\overleftarrow{A}, \overrightarrow{A}) \cup (\overleftarrow{B}, \overrightarrow{B}))(u) &= (\overleftarrow{A}, \overrightarrow{A})(u) \vee (\overleftarrow{B}, \overrightarrow{B})(u) \\ &\triangleq \max((\overleftarrow{A}, \overrightarrow{A})(u), (\overleftarrow{B}, \overrightarrow{B})(u)) \\ ((\overleftarrow{A}, \overrightarrow{A}) \cap (\overleftarrow{B}, \overrightarrow{B}))(u) &= (\overleftarrow{A}, \overrightarrow{A})(u) \wedge (\overleftarrow{B}, \overrightarrow{B})(u) \\ &\triangleq \min((\overleftarrow{A}, \overrightarrow{A})(u), (\overleftarrow{B}, \overrightarrow{B})(u)) \\ (\overleftarrow{A}, \overrightarrow{A})^c(u) &= 1 - (\overleftarrow{A}, \overrightarrow{A})(u) \\ &\triangleq (\overleftarrow{1} - \overleftarrow{A}(\overleftarrow{u}), \overrightarrow{1} - \overrightarrow{A}(\overrightarrow{u})) \end{aligned}$$

[where $u \stackrel{DF}{=} (\overleftarrow{u}, \overrightarrow{u})$].

It divides by whether domain U is finite or infinite.

(1) Domain $U = \{(\overleftarrow{u}_1, \overrightarrow{u}_1), (\overleftarrow{u}_2, \overrightarrow{u}_2), \dots, (\overleftarrow{u}_n, \overrightarrow{u}_n)\}$ is finite set and DF set.

$$\begin{aligned} (\overleftarrow{A}, \overrightarrow{A}) &= \left(\sum \frac{\overleftarrow{A}(\overleftarrow{u}_i)}{\overleftarrow{u}_i}, \sum \frac{\overrightarrow{A}(\overrightarrow{u}_i)}{\overrightarrow{u}_i} \right) \\ (\overleftarrow{B}, \overrightarrow{B}) &= \left(\sum \frac{\overleftarrow{B}(\overleftarrow{u}_i)}{\overleftarrow{u}_i}, \sum \frac{\overrightarrow{B}(\overrightarrow{u}_i)}{\overrightarrow{u}_i} \right) \end{aligned}$$

Then,

$$\begin{aligned} (\overleftarrow{A}, \overrightarrow{A}) \cup (\overleftarrow{B}, \overrightarrow{B}) &= \left(\sum \frac{\overleftarrow{A}(\overleftarrow{u}_i) \vee \overleftarrow{B}(\overleftarrow{u}_i)}{\overleftarrow{u}_i}, \sum \frac{\overrightarrow{A}(\overrightarrow{u}_i) \vee \overrightarrow{B}(\overrightarrow{u}_i)}{\overrightarrow{u}_i} \right) \\ (\overleftarrow{A}, \overrightarrow{A}) \cap (\overleftarrow{B}, \overrightarrow{B}) &= \left(\sum \frac{\overleftarrow{A}(\overleftarrow{u}_i) \wedge \overleftarrow{B}(\overleftarrow{u}_i)}{\overleftarrow{u}_i}, \sum \frac{\overrightarrow{A}(\overrightarrow{u}_i) \wedge \overrightarrow{B}(\overrightarrow{u}_i)}{\overrightarrow{u}_i} \right) \\ (\overleftarrow{A}, \overrightarrow{A})^c &= \left(\sum \frac{\overleftarrow{1} - \overleftarrow{A}(\overleftarrow{u}_i)}{\overleftarrow{u}_i}, \sum \frac{\overrightarrow{1} - \overrightarrow{A}(\overrightarrow{u}_i)}{\overrightarrow{u}_i} \right). \end{aligned}$$

(2) Domain U is infinite set and DF set

$$\begin{aligned} (\overleftarrow{A}, \overrightarrow{A}) &= \left(\int_{\overleftarrow{u} \in U} \frac{\overleftarrow{A}(\overleftarrow{u})}{\overleftarrow{u}}, \int_{\overrightarrow{u} \in U} \frac{\overrightarrow{A}(\overrightarrow{u})}{\overrightarrow{u}} \right) \\ (\overleftarrow{B}, \overrightarrow{B}) &= \left(\int_{\overleftarrow{u} \in U} \frac{\overleftarrow{B}(\overleftarrow{u})}{\overleftarrow{u}}, \int_{\overrightarrow{u} \in U} \frac{\overrightarrow{B}(\overrightarrow{u})}{\overrightarrow{u}} \right). \end{aligned}$$

Then,

$$\begin{aligned} (\overleftarrow{A}, \overrightarrow{A}) \cup (\overleftarrow{B}, \overrightarrow{B}) &= \left(\int \frac{\overleftarrow{A}(\overleftarrow{u}) \vee \overleftarrow{B}(\overleftarrow{u})}{\overleftarrow{u}}, \int \frac{\overrightarrow{A}(\overrightarrow{u}) \vee \overrightarrow{B}(\overrightarrow{u})}{\overrightarrow{u}} \right) \\ (\overleftarrow{A}, \overrightarrow{A}) \cap (\overleftarrow{B}, \overrightarrow{B}) &= \left(\int \frac{\overleftarrow{A}(\overleftarrow{u}) \wedge \overleftarrow{B}(\overleftarrow{u})}{\overleftarrow{u}}, \int \frac{\overrightarrow{A}(\overrightarrow{u}) \wedge \overrightarrow{B}(\overrightarrow{u})}{\overrightarrow{u}} \right) \\ (\overleftarrow{A}, \overrightarrow{A})^c &= \left(\int \frac{\overleftarrow{1} - \overleftarrow{A}(\overleftarrow{u})}{\overleftarrow{u}}, \int \frac{\overrightarrow{1} - \overrightarrow{A}(\overrightarrow{u})}{\overrightarrow{u}} \right). \end{aligned}$$

Theorem 8.1 $(DF(U), \cup, \cap, c)$ have the follow properties:

(1) **Idempotent law**

$$\begin{aligned} (\overleftarrow{A}, \overrightarrow{A}) \cup (\overleftarrow{A}, \overrightarrow{A}) &= (\overleftarrow{A}, \overrightarrow{A}) \\ (\overleftarrow{A}, \overrightarrow{A}) \cap (\overleftarrow{A}, \overrightarrow{A}) &= (\overleftarrow{A}, \overrightarrow{A}) \end{aligned}$$

(2) **Commutative law**

$$\begin{aligned} (\overleftarrow{A}, \overrightarrow{A}) \cup (\overleftarrow{B}, \overrightarrow{B}) &= (\overleftarrow{B}, \overrightarrow{B}) \cup (\overleftarrow{A}, \overrightarrow{A}) \\ (\overleftarrow{A}, \overrightarrow{A}) \cap (\overleftarrow{B}, \overrightarrow{B}) &= (\overleftarrow{B}, \overrightarrow{B}) \cap (\overleftarrow{A}, \overrightarrow{A}) \end{aligned}$$

(3) **Associative law**

$$\begin{aligned} ((\overleftarrow{A}, \overrightarrow{A}) \cup (\overleftarrow{B}, \overrightarrow{B})) \cup (\overleftarrow{C}, \overrightarrow{C}) &= (\overleftarrow{A}, \overrightarrow{A}) \cup ((\overleftarrow{B}, \overrightarrow{B}) \cup (\overleftarrow{C}, \overrightarrow{C})) \\ ((\overleftarrow{A}, \overrightarrow{A}) \cap (\overleftarrow{B}, \overrightarrow{B})) \cap (\overleftarrow{C}, \overrightarrow{C}) &= (\overleftarrow{A}, \overrightarrow{A}) \cap ((\overleftarrow{B}, \overrightarrow{B}) \cap (\overleftarrow{C}, \overrightarrow{C})) \end{aligned}$$

(4) **Absorption law**

$$\begin{aligned} ((\overleftarrow{A}, \overrightarrow{A}) \cup (\overleftarrow{B}, \overrightarrow{B})) \cap (\overleftarrow{A}, \overrightarrow{A}) &= (\overleftarrow{A}, \overrightarrow{A}) \\ ((\overleftarrow{A}, \overrightarrow{A}) \cap (\overleftarrow{B}, \overrightarrow{B})) \cup (\overleftarrow{A}, \overrightarrow{A}) &= (\overleftarrow{A}, \overrightarrow{A}) \end{aligned}$$

(5) **Distribution law**

$$\begin{aligned} ((\overleftarrow{A}, \overrightarrow{A}) \cup (\overleftarrow{B}, \overrightarrow{B})) \cap (\overleftarrow{C}, \overrightarrow{C}) &= ((\overleftarrow{A}, \overrightarrow{A}) \cap (\overleftarrow{C}, \overrightarrow{C})) \cup ((\overleftarrow{B}, \overrightarrow{B}) \cap (\overleftarrow{C}, \overrightarrow{C})) \\ ((\overleftarrow{A}, \overrightarrow{A}) \cap (\overleftarrow{B}, \overrightarrow{B})) \cup (\overleftarrow{C}, \overrightarrow{C}) &= ((\overleftarrow{A}, \overrightarrow{A}) \cup (\overleftarrow{C}, \overrightarrow{C})) \cap ((\overleftarrow{B}, \overrightarrow{B}) \cup (\overleftarrow{C}, \overrightarrow{C})) \end{aligned}$$

(6) **Zero-one law**

$$\begin{aligned} (\overleftarrow{A}, \overrightarrow{A}) \cup (\overleftarrow{\Phi}, \overrightarrow{\Phi}) &= (\overleftarrow{A}, \overrightarrow{A}) \\ (\overleftarrow{A}, \overrightarrow{A}) \cap (\overleftarrow{\Phi}, \overrightarrow{\Phi}) &= (\overleftarrow{\Phi}, \overrightarrow{\Phi}) \\ (\overleftarrow{A}, \overrightarrow{A}) \cup (\overleftarrow{U}, \overrightarrow{U}) &= (\overleftarrow{U}, \overrightarrow{U}) \\ (\overleftarrow{A}, \overrightarrow{A}) \cap (\overleftarrow{U}, \overrightarrow{U}) &= (\overleftarrow{A}, \overrightarrow{A}) \end{aligned}$$

(7) **Pull back law**

$$((\overleftarrow{A}, \overrightarrow{A})^c)^c = (\overleftarrow{A}, \overrightarrow{A})$$

(8) Dualization law

$$((\overleftarrow{A}, \overrightarrow{A}) \cup (\overleftarrow{B}, \overrightarrow{B}))^c = (\overleftarrow{A}, \overrightarrow{A})^c \cap (\overleftarrow{B}, \overrightarrow{B})^c$$

$$((\overleftarrow{A}, \overrightarrow{A}) \cap (\overleftarrow{B}, \overrightarrow{B}))^c = (\overleftarrow{A}, \overrightarrow{A})^c \cup (\overleftarrow{B}, \overrightarrow{B})^c$$

DF set does not satisfy the complementary law. The fundamental reason is that there is no clear boundary of DF set, that is, the boundary of the dynamic fuzzy.

8.1.3 Cut set of dynamic fuzzy sets

In the transformation of DF set and common set, an important concept is the horizontal section set $(\overleftarrow{\lambda}, \overrightarrow{\lambda})$.

Definition 8.4 Assume $(\overleftarrow{A}, \overrightarrow{A}) \in DF(U)$, $(\overleftarrow{\lambda}, \overrightarrow{\lambda}) \in [0, 1] \times [\leftarrow, \rightarrow]$, we note the following:

- (1) $\overleftarrow{A}_{\overleftarrow{\lambda}} = \{\overleftarrow{u} \mid \overleftarrow{u} \in U, \overleftarrow{A}(\overleftarrow{u}) \geq \overleftarrow{\lambda}\}$ or $\overrightarrow{A}_{\overrightarrow{\lambda}} = \{\overrightarrow{u} \mid \overrightarrow{u} \in U, \overrightarrow{A}(\overrightarrow{u}) \geq \overrightarrow{\lambda}\}$, ($0 \leq (\overleftarrow{\lambda}, \overrightarrow{\lambda}) < 1$), we note that $(\overleftarrow{A}_{\overleftarrow{\lambda}}, \overrightarrow{A}_{\overrightarrow{\lambda}})$ is the horizontal section set $(\overleftarrow{\lambda}, \overrightarrow{\lambda})$ of $(\overleftarrow{A}, \overrightarrow{A})$.
- (2) $\overleftarrow{A}_{\overleftarrow{\lambda}} = \{\overleftarrow{u} \mid \overleftarrow{u} \in U, \overleftarrow{A}(\overleftarrow{u}) > \overleftarrow{\lambda}\}$ or $\overrightarrow{A}_{\overrightarrow{\lambda}} = \{\overrightarrow{u} \mid \overrightarrow{u} \in U, \overrightarrow{A}(\overrightarrow{u}) > \overrightarrow{\lambda}\}$, ($0 \leq (\overleftarrow{\lambda}, \overrightarrow{\lambda}) < 1$), we note that $(\overleftarrow{A}_{\overleftarrow{\lambda}}, \overrightarrow{A}_{\overrightarrow{\lambda}})$ is the weak section set $(\overleftarrow{\lambda}, \overrightarrow{\lambda})$ of $(\overleftarrow{A}, \overrightarrow{A})$.

According to definition, for $\forall (\overleftarrow{u}, \overrightarrow{u}) \in U$, if $(\overleftarrow{A}(\overleftarrow{u}), \overrightarrow{A}(\overrightarrow{u})) \geq (\overleftarrow{\lambda}, \overrightarrow{\lambda})$, then $(\overleftarrow{u}, \overrightarrow{u}) \in (\overleftarrow{A}_{\overleftarrow{\lambda}}, \overrightarrow{A}_{\overrightarrow{\lambda}})$. That means, in the horizontal $(\overleftarrow{\lambda}, \overrightarrow{\lambda})$, $(\overleftarrow{u}, \overrightarrow{u})$ belongs to the DF set $(\overleftarrow{A}, \overrightarrow{A})$. If $(\overleftarrow{A}(\overleftarrow{u}), \overrightarrow{A}(\overrightarrow{u})) < (\overleftarrow{\lambda}, \overrightarrow{\lambda})$, then $(\overleftarrow{u}, \overrightarrow{u}) \notin (\overleftarrow{A}_{\overleftarrow{\lambda}}, \overrightarrow{A}_{\overrightarrow{\lambda}})$. That means, in the horizontal $(\overleftarrow{\lambda}, \overrightarrow{\lambda})$, $(\overleftarrow{u}, \overrightarrow{u})$ does not belong to the DF set $(\overleftarrow{A}, \overrightarrow{A})$. Therefore, a DF set can be regarded as an unclear dynamic set with only moving boundary.

Property 8.1 Assume $(\overleftarrow{A}, \overrightarrow{A}), (\overleftarrow{B}, \overrightarrow{B}) \in DF(U)$,
then

$$((\overleftarrow{A}, \overrightarrow{A}) \cup (\overleftarrow{B}, \overrightarrow{B}))_{(\overleftarrow{\lambda}, \overrightarrow{\lambda})} = (\overleftarrow{A}, \overrightarrow{A})_{(\overleftarrow{\lambda}, \overrightarrow{\lambda})} \cup (\overleftarrow{B}, \overrightarrow{B})_{(\overleftarrow{\lambda}, \overrightarrow{\lambda})}$$

$$((\overleftarrow{A}, \overrightarrow{A}) \cap (\overleftarrow{B}, \overrightarrow{B}))_{(\overleftarrow{\lambda}, \overrightarrow{\lambda})} = (\overleftarrow{A}, \overrightarrow{A})_{(\overleftarrow{\lambda}, \overrightarrow{\lambda})} \cap (\overleftarrow{B}, \overrightarrow{B})_{(\overleftarrow{\lambda}, \overrightarrow{\lambda})}.$$

Obviously, for the finite number of DF set in $DF(U)$, these conclusions still hold.

Namely:

$$(\bigcup_{i=1}^n (\overleftarrow{A}_i, \overrightarrow{A}_i))_{(\overleftarrow{\lambda}, \overrightarrow{\lambda})} = \bigcup_{i=1}^n (\overleftarrow{A}_{i\overleftarrow{\lambda}}, \overrightarrow{A}_{i\overrightarrow{\lambda}})$$

$$(\bigcap_{i=1}^n (\overleftarrow{A}_i, \overrightarrow{A}_i))_{(\overleftarrow{\lambda}, \overrightarrow{\lambda})} = \bigcap_{i=1}^n (\overleftarrow{A}_{i\overleftarrow{\lambda}}, \overrightarrow{A}_{i\overrightarrow{\lambda}}).$$

Property 8.2 If $(\overleftarrow{A}_t, \overrightarrow{A}_t)_{(\overleftarrow{t}, \overrightarrow{t}) \in T} \subseteq DF(U)$, then

$$\begin{aligned} \bigcup_{(\overleftarrow{t}, \overrightarrow{t}) \in T} (\overleftarrow{A}_t, \overrightarrow{A}_t)(\overleftarrow{x}, \overrightarrow{x}) &= \bigcup_{(\overleftarrow{t}, \overrightarrow{t}) \in T} (\overleftarrow{A}_t, \overrightarrow{A}_t)(\overleftarrow{x}, \overrightarrow{x}) \\ \bigcap_{(\overleftarrow{t}, \overrightarrow{t}) \in T} (\overleftarrow{A}_t, \overrightarrow{A}_t)(\overleftarrow{x}, \overrightarrow{x}) &= \bigcap_{(\overleftarrow{t}, \overrightarrow{t}) \in T} (\overleftarrow{A}_t, \overrightarrow{A}_t)(\overleftarrow{x}, \overrightarrow{x}). \end{aligned}$$

Property 8.3 Assume $(\overleftarrow{\lambda}_1, \overrightarrow{\lambda}_1), (\overleftarrow{\lambda}_2, \overrightarrow{\lambda}_2) \in [0, 1] \times [\leftarrow, \rightarrow]$; $(\overleftarrow{A}, \overrightarrow{A}) \in DF(U)$. If $(\overleftarrow{\lambda}_2, \overrightarrow{\lambda}_2) \leq (\overleftarrow{\lambda}_1, \overrightarrow{\lambda}_1)$, then

$$(\overleftarrow{A}, \overrightarrow{A})_{(\overleftarrow{\lambda}_2, \overrightarrow{\lambda}_2)} \supseteq (\overleftarrow{A}, \overrightarrow{A})_{(\overleftarrow{\lambda}_1, \overrightarrow{\lambda}_1)}.$$

Property 8.4 Assume $\forall t \in T, (\overleftarrow{\lambda}_t, \overrightarrow{\lambda}_t) \in [0, 1] \times [\leftarrow, \rightarrow]$, then

$$(\overleftarrow{A}, \overrightarrow{A})_{\bigcup_{t \in T} (\overleftarrow{\lambda}_t, \overrightarrow{\lambda}_t)} = \bigcup_{t \in T} (\overleftarrow{A}_{\overleftarrow{\lambda}_t}, \overrightarrow{A}_{\overrightarrow{\lambda}_t}).$$

8.1.4 Dynamic fuzzy sets decomposition theorem

According to the concept of horizontal section set the on the last section, we can get the following theorem:

Theorem 8.2 The horizontal section set $(\overline{\alpha}, \overline{\alpha})$ and the weak horizontal section set $(\overline{\alpha}, \overline{\alpha})$ have the following properties:

- (1) $((\overleftarrow{A}, \overrightarrow{A}) \cup (\overleftarrow{B}, \overrightarrow{B}))_{(\overline{\alpha}, \overline{\alpha})} = ((\overleftarrow{A}, \overrightarrow{A})_{(\overline{\alpha}, \overline{\alpha})}) \cup ((\overleftarrow{B}, \overrightarrow{B})_{(\overline{\alpha}, \overline{\alpha})})$
 $((\overleftarrow{A}, \overrightarrow{A}) \cap (\overleftarrow{B}, \overrightarrow{B}))_{(\overline{\alpha}, \overline{\alpha})} = ((\overleftarrow{A}, \overrightarrow{A})_{(\overline{\alpha}, \overline{\alpha})}) \cap ((\overleftarrow{B}, \overrightarrow{B})_{(\overline{\alpha}, \overline{\alpha})})$
- (2) $((\overleftarrow{A}, \overrightarrow{A}) \cup (\overleftarrow{B}, \overrightarrow{B}))_{(\overline{\alpha}, \overline{\alpha})} = ((\overleftarrow{A}, \overrightarrow{A})_{(\overline{\alpha}, \overline{\alpha})}) \cup ((\overleftarrow{B}, \overrightarrow{B})_{(\overline{\alpha}, \overline{\alpha})})$
 $((\overleftarrow{A}, \overrightarrow{A}) \cap (\overleftarrow{B}, \overrightarrow{B}))_{(\overline{\alpha}, \overline{\alpha})} = ((\overleftarrow{A}, \overrightarrow{A})_{(\overline{\alpha}, \overline{\alpha})}) \cap ((\overleftarrow{B}, \overrightarrow{B})_{(\overline{\alpha}, \overline{\alpha})})$

Theorem 8.3 If $\{(\overleftarrow{A}, \overrightarrow{A})_{(\overleftarrow{t}, \overrightarrow{t})}; (\overleftarrow{t}, \overrightarrow{t}) \in T\} \subset DF(\overleftarrow{X}, \overrightarrow{X})$, then we have the following properties:

- (1) $(\bigcup_{(\overleftarrow{t}, \overrightarrow{t}) \in T} ((\overleftarrow{A}, \overrightarrow{A})_{(\overleftarrow{t}, \overrightarrow{t})}))_{(\overline{\alpha}, \overline{\alpha})} \supset \bigcup_{(\overleftarrow{t}, \overrightarrow{t}) \in T} ((\overleftarrow{A}, \overrightarrow{A})_{(\overleftarrow{t}, \overrightarrow{t})})_{(\overline{\alpha}, \overline{\alpha})}$
- (2) $(\bigcap_{(\overleftarrow{t}, \overrightarrow{t}) \in T} ((\overleftarrow{A}_{(\overleftarrow{t}, \overrightarrow{t})}, \overrightarrow{A}_{(\overleftarrow{t}, \overrightarrow{t})}))_{(\overline{\alpha}, \overline{\alpha})} = \bigcap_{(\overleftarrow{t}, \overrightarrow{t}) \in T} ((\overleftarrow{A}_{(\overleftarrow{t}, \overrightarrow{t})}, \overrightarrow{A}_{(\overleftarrow{t}, \overrightarrow{t})})_{(\overline{\alpha}, \overline{\alpha})})$
- (3) $(\bigcup_{(\overleftarrow{t}, \overrightarrow{t}) \in T} ((\overleftarrow{A}_{\overleftarrow{t}}, \overrightarrow{A}_{\overrightarrow{t}})))_{(\overline{\alpha}, \overline{\alpha})} = \bigcup_{(\overleftarrow{t}, \overrightarrow{t}) \in T} ((\overleftarrow{A}_{\overleftarrow{t}}, \overrightarrow{A}_{\overrightarrow{t}})_{(\overline{\alpha}, \overline{\alpha})})$
- (4) $(\bigcap_{(\overleftarrow{t}, \overrightarrow{t}) \in T} (\overleftarrow{A}_{\overleftarrow{t}}, \overrightarrow{A}_{\overrightarrow{t}}))_{(\overline{\alpha}, \overline{\alpha})} \subset \bigcap_{(\overleftarrow{t}, \overrightarrow{t}) \in T} ((\overleftarrow{A}_{\overleftarrow{t}}, \overrightarrow{A}_{\overrightarrow{t}})_{(\overline{\alpha}, \overline{\alpha})})$

Proof: If $(\overleftarrow{x}, \overrightarrow{x}) \in \bigcup_{(\overleftarrow{t}, \overrightarrow{t}) \in T} ((\overleftarrow{A}_{\overleftarrow{t}}, \overrightarrow{A}_{\overrightarrow{t}}))_{(\overleftarrow{\alpha}, \overrightarrow{\alpha})}$, then it exists $(\overleftarrow{t}_0, \overrightarrow{t}_0) \in T$. Let satisfied $(\overleftarrow{x}, \overrightarrow{x}) \in ((\overleftarrow{A}_{\overleftarrow{t}_0}, \overrightarrow{A}_{\overrightarrow{t}_0}))_{(\overleftarrow{\alpha}, \overrightarrow{\alpha})}$, Therefore, $(\overleftarrow{A}_{\overleftarrow{t}_0}, \overrightarrow{A}_{\overrightarrow{t}_0})(\overleftarrow{x}, \overrightarrow{x}) > (\overleftarrow{\alpha}, \overrightarrow{\alpha})$, namely, $\text{Sup}_{(\overleftarrow{t}, \overrightarrow{t}) \in T} (\overleftarrow{A}_{\overleftarrow{t}}, \overrightarrow{A}_{\overrightarrow{t}})(\overleftarrow{x}, \overrightarrow{x}) > (\overleftarrow{\alpha}, \overrightarrow{\alpha})$; thus, $(\overleftarrow{x}, \overrightarrow{x}) \in \bigcup_{(\overleftarrow{t}, \overrightarrow{t}) \in T} ((\overleftarrow{A}_{\overleftarrow{t}}, \overrightarrow{A}_{\overrightarrow{t}}))_{(\overleftarrow{\alpha}, \overrightarrow{\alpha})}$. So we prove property

(1). The rest of the situation is similar to the certificate.
[Prove up].

Theorem 8.4 Assume $(\overleftarrow{A}, \overrightarrow{A}) \in DF(\overleftarrow{X}, \overrightarrow{X})$, $\{(\overleftarrow{\alpha}_{\overleftarrow{t}}, \overrightarrow{\alpha}_{\overrightarrow{t}}); (\overleftarrow{t}, \overrightarrow{t}) \in T\} \subset [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]$. Then we have

$$(1) (\overleftarrow{A}_{\overleftarrow{t}}, \overrightarrow{A}_{\overrightarrow{t}})_{(\overleftarrow{\alpha}, \overrightarrow{\alpha})} = \bigcap_{(\overleftarrow{t}, \overrightarrow{t}) \in T} (\overleftarrow{A}_{\overleftarrow{\alpha}_{\overleftarrow{t}}}, \overrightarrow{A}_{\overrightarrow{\alpha}_{\overrightarrow{t}}}), (\overleftarrow{A}_{\overleftarrow{t}}, \overrightarrow{A}_{\overrightarrow{t}}) \supset \bigcup_{(\overleftarrow{t}, \overrightarrow{t}) \in T} (\overleftarrow{A}_{\overleftarrow{\alpha}_{\overleftarrow{t}}}, \overrightarrow{A}_{\overrightarrow{\alpha}_{\overrightarrow{t}}})$$

$$(2) (\overleftarrow{A}_{\overleftarrow{\alpha}}, \overrightarrow{A}_{\overrightarrow{\alpha}}) \subset \bigcap_{(\overleftarrow{t}, \overrightarrow{t}) \in T} (\overleftarrow{A}_{\overleftarrow{\alpha}_{\overleftarrow{t}}}, \overrightarrow{A}_{\overrightarrow{\alpha}_{\overrightarrow{t}}}), (\overleftarrow{A}_{\overleftarrow{t}}, \overrightarrow{A}_{\overrightarrow{t}}) = \bigcup_{(\overleftarrow{t}, \overrightarrow{t}) \in T} (\overleftarrow{A}_{\overleftarrow{\alpha}_{\overleftarrow{t}}}, \overrightarrow{A}_{\overrightarrow{\alpha}_{\overrightarrow{t}}}),$$

where

$$(\overleftarrow{\alpha}, \overrightarrow{\alpha}) = \bigvee_{(\overleftarrow{t}, \overrightarrow{t}) \in T} (\overleftarrow{\alpha}_{\overleftarrow{t}}, \overrightarrow{\alpha}_{\overrightarrow{t}}), (\overleftarrow{\beta}, \overrightarrow{\beta}) = \bigwedge_{(\overleftarrow{t}, \overrightarrow{t}) \in T} (\overleftarrow{\alpha}_{\overleftarrow{t}}, \overrightarrow{\alpha}_{\overrightarrow{t}}).$$

Proof: According to

$$\begin{aligned} (\overleftarrow{A}_{\overleftarrow{\alpha}}, \overrightarrow{A}_{\overrightarrow{\alpha}}) &= \{(\overleftarrow{x}, \overrightarrow{x}); (\overleftarrow{A}, \overrightarrow{A})(\overleftarrow{x}, \overrightarrow{x}) \geq \bigvee_{(\overleftarrow{t}, \overrightarrow{t}) \in T} (\overleftarrow{\alpha}_{\overleftarrow{t}}, \overrightarrow{\alpha}_{\overrightarrow{t}})\} \\ &= \bigcap_{(\overleftarrow{t}, \overrightarrow{t}) \in T} \{(\overleftarrow{x}, \overrightarrow{x}); (\overleftarrow{A}, \overrightarrow{A})(\overleftarrow{x}, \overrightarrow{x}) \geq (\overleftarrow{\alpha}_{\overleftarrow{t}}, \overrightarrow{\alpha}_{\overrightarrow{t}})\} \\ &= \bigcap_{(\overleftarrow{t}, \overrightarrow{t}) \in T} (\overleftarrow{A}_{\overleftarrow{\alpha}_{\overleftarrow{t}}}, \overrightarrow{A}_{\overrightarrow{\alpha}_{\overrightarrow{t}}}). \end{aligned}$$

Then we know

$$\begin{aligned} (\overleftarrow{A}_{\overleftarrow{\beta}}, \overrightarrow{A}_{\overrightarrow{\beta}}) &= \{(\overleftarrow{x}, \overrightarrow{x}); (\overleftarrow{A}, \overrightarrow{A})(\overleftarrow{x}, \overrightarrow{x}) \geq \bigwedge_{(\overleftarrow{t}, \overrightarrow{t}) \in T} (\overleftarrow{\alpha}_{\overleftarrow{t}}, \overrightarrow{\alpha}_{\overrightarrow{t}})\} \\ &\supset \bigcup_{(\overleftarrow{t}, \overrightarrow{t}) \in T} \{(\overleftarrow{x}, \overrightarrow{x}); (\overleftarrow{A}, \overrightarrow{A})(\overleftarrow{x}, \overrightarrow{x}) \geq (\overleftarrow{\alpha}_{\overleftarrow{t}}, \overrightarrow{\alpha}_{\overrightarrow{t}})\} \\ &\supset \bigcup_{(\overleftarrow{t}, \overrightarrow{t}) \in T} (\overleftarrow{A}_{\overleftarrow{\alpha}_{\overleftarrow{t}}}, \overrightarrow{A}_{\overrightarrow{\alpha}_{\overrightarrow{t}}}) \end{aligned}$$

[Prove up].

(2) It is Similar to permit, so we omitted.

Theorem 8.5 For any $(\overleftarrow{A}, \overrightarrow{A}) \in DF(\overleftarrow{X}, \overrightarrow{X})$, we have

$$(1) (\overleftarrow{A}_{\overrightarrow{\alpha}}, \overrightarrow{A}_{\overrightarrow{\alpha}}) = \bigcap_{(\overrightarrow{\lambda}, \overrightarrow{\lambda}) < (\overrightarrow{\alpha}, \overrightarrow{\alpha})} (\overleftarrow{A}_{\overrightarrow{\lambda}}, \overrightarrow{A}_{\overrightarrow{\lambda}})$$

$$(2) (\overleftarrow{A}_{\overrightarrow{\alpha}}, \overrightarrow{A}_{\overrightarrow{\alpha}}) = \bigcup_{(\overrightarrow{\lambda}, \overrightarrow{\lambda}) > (\overrightarrow{\alpha}, \overrightarrow{\alpha})} (\overleftarrow{A}_{\overrightarrow{\lambda}}, \overrightarrow{A}_{\overrightarrow{\lambda}})$$

Definition 8.5 Assume $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \in [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]$, $(\overleftarrow{A}, \overrightarrow{A}) \in DF(\overleftarrow{X}, \overrightarrow{X})$, then the number product of $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$ with $(\overleftarrow{A}, \overrightarrow{A})$ is $((\overleftarrow{\alpha}, \overrightarrow{\alpha})(\overleftarrow{A}, \overrightarrow{A}))(\overleftarrow{X}, \overrightarrow{X}) = (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge ((\overleftarrow{A}, \overrightarrow{A})(\overleftarrow{X}, \overrightarrow{X}))$.

Theorem 8.6 (DF set decomposition theorem) For any $(\overleftarrow{A}, \overrightarrow{A}) \in DF(\overleftarrow{X}, \overrightarrow{X})$, we have

$$\begin{aligned} (\overleftarrow{A}, \overrightarrow{A}) &= \bigcup_{(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \in [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]} (\overleftarrow{\alpha}, \overrightarrow{\alpha}) (\overleftarrow{A}, \overrightarrow{A})_{(\overleftarrow{\alpha}, \overrightarrow{\alpha})} \\ (\overleftarrow{A}, \overrightarrow{A}) &= \bigcup_{(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \in [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]} (\overleftarrow{\alpha}, \overrightarrow{\alpha}) (\overleftarrow{A}, \overrightarrow{A})_{(\overleftarrow{\alpha}, \overrightarrow{\alpha})}. \end{aligned}$$

Proof: Since

$$(\overleftarrow{A}, \overrightarrow{A})_{(\overleftarrow{\alpha}, \overrightarrow{\alpha})} (\overleftarrow{X}, \overrightarrow{X}) = \begin{cases} (\overleftarrow{1}, \overrightarrow{1}), (\overleftarrow{X}, \overrightarrow{X}) \in (\overleftarrow{A}_{\overrightarrow{\alpha}}, \overrightarrow{A}_{\overrightarrow{\alpha}}), \\ (\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{X}, \overrightarrow{X}) \notin (\overleftarrow{A}_{\overrightarrow{\alpha}}, \overrightarrow{A}_{\overrightarrow{\alpha}}) \end{cases},$$

then we have

$$\begin{aligned} & \left(\bigcup_{(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \in [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]} (\overleftarrow{\alpha}, \overrightarrow{\alpha}) (\overleftarrow{A}, \overrightarrow{A})_{(\overleftarrow{\alpha}, \overrightarrow{\alpha})} \right) (\overleftarrow{X}, \overrightarrow{X}) \\ &= \text{Sup}(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \bullet ((\overleftarrow{A}_{\overrightarrow{\alpha}}, \overrightarrow{A}_{\overrightarrow{\alpha}})(\overleftarrow{X}, \overrightarrow{X})) \\ &= \text{Sup}_{(\overleftarrow{X}, \overrightarrow{X}) \in (\overleftarrow{A}, \overrightarrow{A})_{(\overleftarrow{\alpha}, \overrightarrow{\alpha})}} (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \\ &= \text{Sup}_{(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \leq (\overleftarrow{A}_{\overrightarrow{X}}, \overrightarrow{A}_{\overrightarrow{X}})} (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \\ &= (\overleftarrow{A}, \overrightarrow{A})(\overleftarrow{X}, \overrightarrow{X}). \end{aligned}$$

Other forms are similar to prove.

[Prove up].

Theorem 8.7 Assume $(\overleftarrow{A}, \overrightarrow{A})$ and $(\overleftarrow{B}, \overrightarrow{B}) \in DF(U)$, then the necessary and sufficient condition for $(\overleftarrow{A}, \overrightarrow{A}) \subset (\overleftarrow{B}, \overrightarrow{B})$ is $(\overleftarrow{A}_{\overline{\alpha}}, \overrightarrow{A}_{\overline{\alpha}}) \subset (\overleftarrow{B}_{\overline{\alpha}}, \overrightarrow{B}_{\overline{\alpha}})$ ($(\overline{\alpha}, \overline{\alpha}) \in [0, 1] \times [\leftarrow, \rightarrow]$) or $(\overleftarrow{A}_{\overline{\alpha}}, \overrightarrow{A}_{\overline{\alpha}}) \subset (\overleftarrow{B}_{\overline{\alpha}}, \overrightarrow{B}_{\overline{\alpha}})$ ($(\overline{\alpha}, \overline{\alpha}) \in [0, 1] \times [\leftarrow, \rightarrow]$) or $(\overleftarrow{A}_{\overline{\alpha}}, \overrightarrow{A}_{\overline{\alpha}}) \subset (\overleftarrow{B}_{\overline{\alpha}}, \overrightarrow{B}_{\overline{\alpha}})$ ($(\overline{\alpha}, \overline{\alpha}) \in R_0$, where R_0 is the rational point set in $[0, 1]$).

Proof: Obvious necessity.

If $(\overleftarrow{A}_{\overline{\alpha}}, \overrightarrow{A}_{\overline{\alpha}}) \subset (\overleftarrow{B}_{\overline{\alpha}}, \overrightarrow{B}_{\overline{\alpha}})$ ($(\overline{\alpha}, \overline{\alpha}) \in [0, 1] \times [\leftarrow, \rightarrow]$), according to Theorem 8.6, we have

$$\begin{aligned} (\overleftarrow{A}, \overrightarrow{A})(\overleftarrow{x}, \overrightarrow{x}) &= \bigvee_{(\overline{\alpha}, \overline{\alpha}) \in [0, 1] \times [\leftarrow, \rightarrow]} (\overline{\alpha}, \overline{\alpha}) \bullet ((\overleftarrow{A}, \overrightarrow{A})_{(\overline{\alpha}, \overline{\alpha})}(\overleftarrow{x}, \overrightarrow{x})) \\ &\leq \bigvee_{(\overline{\alpha}, \overline{\alpha}) \in [0, 1] \times [\leftarrow, \rightarrow]} (\overline{\alpha}, \overline{\alpha}) \bullet ((\overleftarrow{B}, \overrightarrow{B})_{(\overline{\alpha}, \overline{\alpha})}(\overleftarrow{x}, \overrightarrow{x})) \\ &\leq (\overleftarrow{B}, \overrightarrow{B})(\overleftarrow{x}, \overrightarrow{x}), \end{aligned}$$

Namely, $(\overleftarrow{A}, \overrightarrow{A}) \subset (\overleftarrow{B}, \overrightarrow{B})$.

Other forms are similar to prove.

[Prove up].

Theorem 8.8 Assume $(\overleftarrow{A}, \overrightarrow{A}) \in DF(\overleftarrow{X}, \overrightarrow{X})$, if it exists $(\overleftarrow{A}^*, \overrightarrow{A}^*)_{(\overline{\alpha}, \overline{\alpha})}$, ($(\overline{\alpha}, \overline{\alpha}) \in [0, 1] \times [\leftarrow, \rightarrow]$). Let satisfy

$$(\overleftarrow{A}, \overrightarrow{A})_{(\overline{\alpha}, \overline{\alpha})} \subset (\overleftarrow{A}^*, \overrightarrow{A}^*)_{(\overline{\alpha}, \overline{\alpha})} \subset (\overleftarrow{A}, \overrightarrow{A})_{(\overline{\alpha}, \overline{\alpha})}, ((\overline{\alpha}, \overline{\alpha}) \in [0, 1] \times [\leftarrow, \rightarrow])$$

then

$$(\overleftarrow{A}, \overrightarrow{A}) = \bigcup_{(\overline{\alpha}, \overline{\alpha}) \in [0, 1] \times [\leftarrow, \rightarrow]} (\overline{\alpha}, \overline{\alpha})(\overleftarrow{A}_{\overline{\alpha}}^*, \overrightarrow{A}_{\overline{\alpha}}^*).$$

Proof: (omitted). [Prove up].

8.2 Dynamic fuzzy relations

8.2.1 The conception dynamic fuzzy relations

In the real word, there are a large number of relations that are dynamic and fuzzy. In some way, it reflects whether the relationship is close or apart and whether the relations are dynamic. “Wang Fang grows more and more like her mother” and “The

relationship between the students and teachers is more and more harmonious" – these are dynamic fuzzy relations.

Definition 8.6 Assume $(\overleftarrow{X}, \overrightarrow{X})$ is a common DFD set and $(\overleftarrow{L}, \overrightarrow{L})$ is dynamic fuzzy lattice; we call the mapping $(\overleftarrow{A}, \overrightarrow{A}) : (\overleftarrow{X}, \overrightarrow{X}) \rightarrow (\overleftarrow{L}, \overrightarrow{L})$ as $(\overleftarrow{L}, \overrightarrow{L})$ type dynamic fuzzy set, and we note that $DF(\overleftarrow{L}, \overrightarrow{L})(\overleftarrow{X}, \overrightarrow{X}) = \{(\overleftarrow{A}, \overrightarrow{A}) : (\overleftarrow{X}, \overrightarrow{X}) \rightarrow (\overleftarrow{L}, \overrightarrow{L})\}$.

Assume $(\overleftarrow{A}, \overrightarrow{A}), (\overleftarrow{B}, \overrightarrow{B}) \in DF(\overleftarrow{L}, \overrightarrow{L})(\overleftarrow{X}, \overrightarrow{X})$, if $(\overleftarrow{A}, \overrightarrow{A})(\overleftarrow{x}, \overrightarrow{x}) \leq (\overleftarrow{B}, \overrightarrow{B})(\overleftarrow{x}, \overrightarrow{x})$, $(\forall (\overleftarrow{x}, \overrightarrow{x}) \in (\overleftarrow{X}, \overrightarrow{X}))$. Namely, $(\overleftarrow{A}, \overrightarrow{A})$ contained in $(\overleftarrow{B}, \overrightarrow{B})$, we note that $(\overleftarrow{A}, \overrightarrow{A}) \subset (\overleftarrow{B}, \overrightarrow{B})$, then $(DF(\overleftarrow{L}, \overrightarrow{L})(\overleftarrow{X}, \overrightarrow{X}), c)$ is dynamic fuzzy poset.

Definition 8.7 If $(\overleftarrow{R}, \overrightarrow{R}) \in DF((\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y}))$, we note that $(\overleftarrow{R}, \overrightarrow{R})$ is the L-type dynamic fuzzy relations among $(\overleftarrow{X}, \overrightarrow{X})$ to $(\overleftarrow{Y}, \overrightarrow{Y})$; if $(\overleftarrow{L}, \overrightarrow{L}) = [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]$, $(\overleftarrow{R}, \overrightarrow{R}) \in DF((\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y}))$, then we note that $(\overleftarrow{R}, \overrightarrow{R})$ is the dynamic fuzzy relations among $(\overleftarrow{X}, \overrightarrow{X})$ to $(\overleftarrow{Y}, \overrightarrow{Y})$.

Definition 8.8 As mentioned, binary dynamic fuzzy relations can be promoted to n set dynamic fuzzy relations.

Definition 8.9 Assume $(\overleftarrow{X}_1, \overrightarrow{X}_1), (\overleftarrow{X}_2, \overrightarrow{X}_2), \dots, (\overleftarrow{X}_n, \overrightarrow{X}_n)$ is n nonempty set, then

$$(\overleftarrow{R}, \overrightarrow{R}) : (\overleftarrow{X}_1, \overrightarrow{X}_1) \times (\overleftarrow{X}_2, \overrightarrow{X}_2) \times \dots \times (\overleftarrow{X}_n, \overrightarrow{X}_n) \rightarrow [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})].$$

We note that $(\overleftarrow{X}, \overrightarrow{X}) = (\overleftarrow{X}_1, \overrightarrow{X}_1) \times (\overleftarrow{X}_2, \overrightarrow{X}_2) \times \dots \times (\overleftarrow{X}_n, \overrightarrow{X}_n)$ as the n set dynamic fuzzy relations.

8.2.2 Property of dynamic fuzzy relations

Proposition 8.1 Assume $(\overleftarrow{R}, \overrightarrow{R}), (\overleftarrow{S}, \overrightarrow{S}), (\overleftarrow{P}, \overrightarrow{P}), (\overleftarrow{R}, \overrightarrow{R})_{(\overleftarrow{t}, \overrightarrow{t})} \in DF((\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y}))$,

$(\overleftarrow{t}, \overrightarrow{t}) \in (\overleftarrow{T}, \overrightarrow{T})$, then we have the following properties:

- (1) $(\overleftarrow{R}, \overrightarrow{R}) \vee (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R}), (\overleftarrow{R}, \overrightarrow{R}) \wedge (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R})$
- (2) $(\overleftarrow{R}, \overrightarrow{R}) \vee (\overleftarrow{S}, \overrightarrow{S}) = (\overleftarrow{S}, \overrightarrow{S}) \vee (\overleftarrow{R}, \overrightarrow{R}), (\overleftarrow{R}, \overrightarrow{R}) \wedge (\overleftarrow{S}, \overrightarrow{S}) = (\overleftarrow{S}, \overrightarrow{S}) \wedge (\overleftarrow{R}, \overrightarrow{R})$
- (3) $((\overleftarrow{R}, \overrightarrow{R}) \vee (\overleftarrow{S}, \overrightarrow{S})) \vee (\overleftarrow{P}, \overrightarrow{P}) = (\overleftarrow{R}, \overrightarrow{R}) \vee ((\overleftarrow{S}, \overrightarrow{S}) \vee (\overleftarrow{P}, \overrightarrow{P}))$
 $((\overleftarrow{R}, \overrightarrow{R}) \wedge (\overleftarrow{S}, \overrightarrow{S})) \wedge (\overleftarrow{P}, \overrightarrow{P}) = (\overleftarrow{R}, \overrightarrow{R}) \wedge ((\overleftarrow{S}, \overrightarrow{S}) \wedge (\overleftarrow{P}, \overrightarrow{P}))$
- (4) $((\overleftarrow{R}, \overrightarrow{R}) \vee (\overleftarrow{S}, \overrightarrow{S})) \wedge (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R})$
 $((\overleftarrow{R}, \overrightarrow{R}) \wedge (\overleftarrow{S}, \overrightarrow{S})) \vee (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R})$
- (5) $(\overleftarrow{R}, \overrightarrow{R}) \wedge (\bigvee_{(\overleftarrow{t}, \overrightarrow{t}) \in (\overleftarrow{T}, \overrightarrow{T})} (\overleftarrow{R}, \overrightarrow{R})_{(\overleftarrow{t}, \overrightarrow{t})}) = \bigvee_{(\overleftarrow{t}, \overrightarrow{t}) \in (\overleftarrow{T}, \overrightarrow{T})} ((\overleftarrow{R}, \overrightarrow{R}) \wedge (\overleftarrow{R}, \overrightarrow{R})_{(\overleftarrow{t}, \overrightarrow{t})})$
 $(\overleftarrow{R}, \overrightarrow{R}) \vee (\bigwedge_{(\overleftarrow{t}, \overrightarrow{t}) \in (\overleftarrow{T}, \overrightarrow{T})} (\overleftarrow{R}, \overrightarrow{R})_{(\overleftarrow{t}, \overrightarrow{t})}) = \bigwedge_{(\overleftarrow{t}, \overrightarrow{t}) \in (\overleftarrow{T}, \overrightarrow{T})} ((\overleftarrow{R}, \overrightarrow{R}) \vee (\overleftarrow{R}, \overrightarrow{R})_{(\overleftarrow{t}, \overrightarrow{t})})$

$$(6) \quad (\overleftarrow{R}, \overrightarrow{R}) \vee (\overleftarrow{I}, \overrightarrow{I}) = (\overleftarrow{I}, \overrightarrow{I}), (\overleftarrow{R}, \overrightarrow{R}) \wedge (\overleftarrow{I}, \overrightarrow{I}) = (\overleftarrow{R}, \overrightarrow{R}) \\ (\overleftarrow{R}, \overrightarrow{R}) \vee (\overrightarrow{O}, \overrightarrow{O}) = (\overrightarrow{R}, \overrightarrow{R}), (\overleftarrow{R}, \overrightarrow{R}) \wedge (\overrightarrow{O}, \overrightarrow{O}) = (\overrightarrow{O}, \overrightarrow{O}),$$

where $(\overleftarrow{I}, \overrightarrow{I}), (\overrightarrow{O}, \overrightarrow{O}) \in DF((\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y})), (\overleftarrow{I}, \overrightarrow{I})((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) = (\overleftarrow{1}, \overrightarrow{1}),$
 $(\overrightarrow{O}, \overrightarrow{O})((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) = (\overrightarrow{0}, \overrightarrow{0}) \quad (\forall ((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) \in ((\overleftarrow{X}, \overrightarrow{X}), (\overleftarrow{Y}, \overrightarrow{Y}))).$ We note respectively that dynamic fuzzy universal relation and dynamic fuzzy zero relation among $(\overleftarrow{X}, \overrightarrow{X})$ to $(\overleftarrow{Y}, \overrightarrow{Y})$.

Definition 8.10 If $(\overleftarrow{R}_1, \overrightarrow{R}_1), (\overleftarrow{R}_2, \overrightarrow{R}_2) \in DF_{(\overleftarrow{L}, \overrightarrow{L})}((\overleftarrow{X}, \overrightarrow{X}), (\overleftarrow{Y}, \overrightarrow{Y})), \forall ((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) \in ((\overleftarrow{X}, \overrightarrow{X}), (\overleftarrow{Y}, \overrightarrow{Y}))$, we define that:

- 1) $(\overleftarrow{R}_1, \overrightarrow{R}_1) \subset (\overleftarrow{R}_2, \overrightarrow{R}_2) \Leftrightarrow (\overleftarrow{R}_1, \overrightarrow{R}_1)((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) \leq (\overleftarrow{R}_2, \overrightarrow{R}_2)((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y}))$
- 2) $((\overleftarrow{R}_1, \overrightarrow{R}_1) \cup (\overleftarrow{R}_2, \overrightarrow{R}_2))((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) = (\overleftarrow{R}_1, \overrightarrow{R}_1)((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) \vee (\overleftarrow{R}_2, \overrightarrow{R}_2)((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y}))$
 $((\overleftarrow{R}_1, \overrightarrow{R}_1) \cap (\overleftarrow{R}_2, \overrightarrow{R}_2))((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) = (\overleftarrow{R}_1, \overrightarrow{R}_1)((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) \wedge (\overleftarrow{R}_2, \overrightarrow{R}_2)((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y}))$
- 3) $(\overleftarrow{R}, \overrightarrow{R})^{-1}((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{y}, \overrightarrow{y})) = (\overleftarrow{R}, \overrightarrow{R})((\overleftarrow{y}, \overrightarrow{y}), (\overleftarrow{x}, \overrightarrow{x}))$

Theorem 8.9 $(\overleftarrow{R}, \overrightarrow{R})^{-1}$ have the following properties:

- 1) $(\overleftarrow{R}_1, \overrightarrow{R}_1) \subset (\overleftarrow{R}_2, \overrightarrow{R}_2) \Rightarrow (\overleftarrow{R}_1, \overrightarrow{R}_1)^{-1} \subset (\overleftarrow{R}_2, \overrightarrow{R}_2)^{-1}$
- 2) $((\overleftarrow{R}, \overrightarrow{R})^{-1})^{-1} = (\overleftarrow{R}, \overrightarrow{R})$
- 3) $((\overleftarrow{R}_1, \overrightarrow{R}_1) \cup (\overleftarrow{R}_2, \overrightarrow{R}_2))^{-1} = (\overleftarrow{R}_1, \overrightarrow{R}_1)^{-1} \cup (\overleftarrow{R}_2, \overrightarrow{R}_2)^{-1}$
- 4) $((\overleftarrow{R}_1, \overrightarrow{R}_1) \cap (\overleftarrow{R}_2, \overrightarrow{R}_2))^{-1} = (\overleftarrow{R}_1, \overrightarrow{R}_1)^{-1} \cap (\overleftarrow{R}_2, \overrightarrow{R}_2)^{-1}$

8.2.3 Dynamic fuzzy matrix

Definition 8.11 We name the matrix like

$$(\overleftarrow{E}, \overrightarrow{E}) = \begin{bmatrix} (\overleftarrow{e}_{11}, \overrightarrow{e}_{11}) & (\overleftarrow{e}_{12}, \overrightarrow{e}_{12}) & \dots & (\overleftarrow{e}_{1n}, \overrightarrow{e}_{1n}) \\ (\overleftarrow{e}_{21}, \overrightarrow{e}_{21}) & (\overleftarrow{e}_{22}, \overrightarrow{e}_{22}) & \dots & (\overleftarrow{e}_{2n}, \overrightarrow{e}_{2n}) \\ \dots & \dots & \dots & \dots \\ (\overleftarrow{e}_{m1}, \overrightarrow{e}_{m1}) & (\overleftarrow{e}_{m2}, \overrightarrow{e}_{m2}) & \dots & (\overleftarrow{e}_{mn}, \overrightarrow{e}_{mn}) \end{bmatrix} \text{ as DF matrix with } m \times n$$

order, where $(\overleftarrow{e}_{ij}, \overrightarrow{e}_{ij}) \in [(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})]$, ($i = 1, 2, \dots, m$; $j = 1, 2, \dots, n$), we name $(\overleftarrow{E}, \overrightarrow{E}) = (\overleftarrow{e}_{ij}, \overrightarrow{e}_{ij})_{m \times n}$, $(\overleftarrow{e}_{ij}, \overrightarrow{e}_{ij})$ as the elements in row i and column j of DF matrix $(\overleftarrow{E}, \overrightarrow{E})$.

Assume $(\overleftarrow{X}, \overrightarrow{X}) = \{(\overleftarrow{x}_1, \overrightarrow{x}_1), (\overleftarrow{x}_2, \overrightarrow{x}_2), \dots, (\overleftarrow{x}_n, \overrightarrow{x}_n)\}$, $(\overleftarrow{Y}, \overrightarrow{Y}) = \{(\overleftarrow{y}_1, \overrightarrow{y}_1), (\overleftarrow{y}_2, \overrightarrow{y}_2), \dots, (\overleftarrow{y}_m, \overrightarrow{y}_m)\}$, $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ is DF lattice, $(\overleftarrow{0}, \overrightarrow{0}), (\overleftarrow{1}, \overrightarrow{1})$ are the maximum and minimum element in $(\overleftarrow{L}, \overrightarrow{L})$.

Definition 8.12 If $(\overleftarrow{R}, \overrightarrow{R}) : (\overleftarrow{X}, \overrightarrow{X}) \times (\overleftarrow{Y}, \overrightarrow{Y}) \rightarrow (\overleftarrow{L}, \overrightarrow{L})$, we name that $(\overleftarrow{R}, \overrightarrow{R})$ is $(\overleftarrow{L}, \overrightarrow{L})$ -type DF matrix; we note that

$$(\overleftarrow{R}, \overrightarrow{R}) = \begin{bmatrix} (\overleftarrow{r}_{11}, \overrightarrow{r}_{11}) & (\overleftarrow{r}_{12}, \overrightarrow{r}_{12}) & \dots & (\overleftarrow{r}_{1m}, \overrightarrow{r}_{1m}) \\ (\overleftarrow{r}_{21}, \overrightarrow{r}_{21}) & (\overleftarrow{r}_{22}, \overrightarrow{r}_{22}) & \dots & (\overleftarrow{r}_{2m}, \overrightarrow{r}_{2m}) \\ \dots & \dots & \dots & \dots \\ (\overleftarrow{r}_{n1}, \overrightarrow{r}_{n1}) & (\overleftarrow{r}_{n2}, \overrightarrow{r}_{n2}) & \dots & (\overleftarrow{r}_{nm}, \overrightarrow{r}_{nm}) \end{bmatrix},$$

where, $(\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}) = (\overleftarrow{R}, \overrightarrow{R})((\overleftarrow{x}_i, \overrightarrow{x}_i), (\overleftarrow{y}_j, \overrightarrow{y}_j))$ ($i \leq n, j \leq m$).

Obviously, $(\overleftarrow{L}, \overrightarrow{L})$ -type DF matrix is $(\overleftarrow{L}, \overrightarrow{L})$ -type DF relations, and DF matrix is DF relations.

Definition 8.13 Assume $(\overleftarrow{R}, \overrightarrow{R})$ is type DF matrix, and $(\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})_{n \times m}$ ($i \leq n, j \leq m$), if it exists $(\overleftarrow{L}, \overrightarrow{L})$ -type DF matrix $(\overleftarrow{Q}, \overrightarrow{Q}) = (\overleftarrow{q}_{ji}, \overrightarrow{q}_{ji})_{m \times n}$ ($j \leq m, i \leq n$), let satisfy $(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R})$, then $(\overleftarrow{R}, \overrightarrow{R})$ is regular, $(\overleftarrow{Q}, \overrightarrow{Q})$ is the generalized $(\overleftarrow{L}, \overrightarrow{L})$ -type DF matrix of $(\overleftarrow{R}, \overrightarrow{R})$.

If we further have $(\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) = (\overleftarrow{Q}, \overrightarrow{Q})$, then $(\overleftarrow{Q}, \overrightarrow{Q})$ is the T inverse matrix of $(\overleftarrow{R}, \overrightarrow{R})$.

Theorem 8.10 Assume $(\overleftarrow{Q}, \overrightarrow{Q})$ is the generalized $(\overleftarrow{L}, \overrightarrow{L})$ -type DF inverse matrix of $(\overleftarrow{R}, \overrightarrow{R})$, then $(\overleftarrow{S}, \overrightarrow{S}) = (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q})$ is T inverse matrix of $(\overleftarrow{R}, \overrightarrow{R})$, where L is a distributive lattice.

Proof: According to Definition 8.13, we can prove $(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{S}, \overrightarrow{S}) \circ (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R})$. Corollaries, $(\overleftarrow{S}, \overrightarrow{S}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{S}, \overrightarrow{S}) = (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) = (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) = (\overleftarrow{S}, \overrightarrow{S})$

[Prove up].

Theorem 8.11 Assume that $(\overleftarrow{R}, \overrightarrow{R})$ is DF matrix, then the necessary and sufficient condition of $(\overleftarrow{R}, \overrightarrow{R})$ is regular is $(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{X}, \overrightarrow{X}) \circ (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R})$, where, $(\overleftarrow{X}, \overrightarrow{X})$ is the maximum generalized DF inverse matrix of $(\overleftarrow{R}, \overrightarrow{R})$. Namely, if $(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R})$, then we must have $(\overleftarrow{Q}, \overrightarrow{Q}) \subset (\overleftarrow{X}, \overrightarrow{X})$.

Proof: Sufficiency is obvious, and the necessity is presented as follows. Assume $(\overleftarrow{R}, \overrightarrow{R})$ is regular, then it exists $(\overleftarrow{Q}, \overrightarrow{Q})$; let satisfy $(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R})$.

Therefore,

$$\bigvee_{k=1}^n \bigwedge_{j=1}^m ((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{q}_{ik}, \overrightarrow{q}_{ik})) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj}) = (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}) \quad (i \leq n, j \leq m).$$

Namely,

$$\bigvee_{k=1}^n \left(\bigvee_{e=1}^m ((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{q}_{ek}, \overrightarrow{q}_{ek})) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj}) \right) = r_{ij} (i \leq n, j \leq m).$$

Thus, for $\forall k \leq n, e \leq m$, we have $((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{q}_{ek}, \overrightarrow{q}_{ek}) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj})) \leq r_{ij}$ ($i \leq n, j \leq m$). We hence have

$$\begin{aligned} ((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj})) &\leq (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}) \Rightarrow 0 \leq (\overleftarrow{q}_{ek}, \overrightarrow{q}_{ek}) \leq 1; ((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj})) > \\ &(\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}) \Rightarrow 0 \leq (\overleftarrow{q}_{ek}, \overrightarrow{q}_{ek}) \leq (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij}). \end{aligned}$$

Then, $0 \leq (\overleftarrow{q}_{ek}, \overrightarrow{q}_{ek}) \leq \wedge \{(\overleftarrow{r}_{st}, \overrightarrow{r}_{st}); (\overleftarrow{r}_{st}, \overrightarrow{r}_{st}) < ((\overleftarrow{r}_{st}, \overrightarrow{r}_{st}) \wedge (\overleftarrow{r}_{nt}, \overrightarrow{r}_{nt}))\} = (\overleftarrow{x}_{ek}, \overrightarrow{x}_{ek})$. Namely, $(\overleftarrow{Q}, \overrightarrow{Q}) \subset (\overleftarrow{X}, \overrightarrow{X})$ and $(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) \subset (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{X}, \overrightarrow{X}) \circ (\overleftarrow{R}, \overrightarrow{R})$.

Assume $(\overleftarrow{U}, \overrightarrow{U}) = (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{X}, \overrightarrow{X}) \circ (\overleftarrow{R}, \overrightarrow{R})$, then

$$u_{ij} = \bigvee_{k=1}^n \bigvee_{e=1}^m ((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{x}_{ek}, \overrightarrow{x}_{ek}) \wedge (\overleftarrow{r}_{ke}, \overrightarrow{r}_{ke})) (i \leq n, j \leq m).$$

We can prove $u_{ij} \leq r_{ij}$ ($i \leq n, j \leq m$).

In fact, when $((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{r}_{ej}, \overrightarrow{r}_{ej})) \leq (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})$, we have $((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{x}_{ik}, \overrightarrow{x}_{ik}) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj})) \leq ((\overleftarrow{r}_{ik}, \overrightarrow{r}_{ik}) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj})) \leq (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})$. If $((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj})) > (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})$, then $((\overleftarrow{r}_{ie}, \overrightarrow{r}_{ie}) \wedge (\overleftarrow{x}_{ek}, \overrightarrow{x}_{ek}) \wedge (\overleftarrow{r}_{kj}, \overrightarrow{r}_{kj})) \leq (\overleftarrow{x}_{ek}, \overrightarrow{x}_{ek}) \leq (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})$. Therefore, $(\overleftarrow{u}_{ij}, \overrightarrow{u}_{ij}) \leq (\overleftarrow{r}_{ij}, \overrightarrow{r}_{ij})$ ($i \leq n, j \leq m$), namely, $(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{X}, \overrightarrow{X}) \circ (\overleftarrow{R}, \overrightarrow{R}) \subset (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R})$; hence, $(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{X}, \overrightarrow{X}) \circ (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R})$.

If $(\overleftarrow{Q}', \overrightarrow{Q}')$ satisfies $(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}', \overrightarrow{Q}') \circ (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R})$ and $(\overleftarrow{R}, \overrightarrow{R}) \subset (\overleftarrow{X}, \overrightarrow{X})$, then $(\overleftarrow{X}, \overrightarrow{X})$ is the maximum generalized DF inverse matrix of $(\overleftarrow{R}, \overrightarrow{R}) \circ (\overleftarrow{Q}, \overrightarrow{Q}) \circ (\overleftarrow{R}, \overrightarrow{R}) = (\overleftarrow{R}, \overrightarrow{R})$.

[Prove up].

8.3 Dynamic fuzzy logic

8.3.1 Dynamic fuzzy Boolean variable

(1) Dynamic fuzzy number Boolean variable

DF number Boolean variable is the Boolean variable in $[0, 1] \times [\leftarrow, \rightarrow]$. While “0” means false, we note that $\overleftarrow{0}$ or $\overrightarrow{0}$ means DF false. While “1” means true, we note that $\overleftarrow{1}$ or $\overrightarrow{1}$ means DF true. We use a number which is greater than the number “0” and less than “1” to indicate a Boolean variable between DF true and false. For example $(\overleftarrow{0.7}, \overrightarrow{0.7})$, real number 0.7 has two trends: it can tend to “0” and but also to “1.”

(2) Dynamic fuzzy interval Boolean variable

Definition 8.14 Assume that any dynamic fuzzy number $(\overleftarrow{a}, \overrightarrow{a})$ and $(\overleftarrow{b}, \overrightarrow{b})$ satisfies $(\overleftarrow{0}, \overrightarrow{0}) \leq (\overleftarrow{a}, \overrightarrow{a}) \leq (\overleftarrow{b}, \overrightarrow{b}) \leq (\overleftarrow{1}, \overrightarrow{1})$, and any dynamic fuzzy number $(\overleftarrow{p}, \overrightarrow{p})$ satisfies $(\overleftarrow{0}, \overrightarrow{0}) \leq (\overleftarrow{p}, \overrightarrow{p}) \leq (\overleftarrow{1}, \overrightarrow{1})$, then we name that $[(\overleftarrow{a}, \overrightarrow{a}), (\overleftarrow{b}, \overrightarrow{b})]$ as interval Boolean variable, $[(\overleftarrow{a}, \overrightarrow{a}), (\overleftarrow{b}, \overrightarrow{b})]/(\overleftarrow{p}, \overrightarrow{p})$ as DF interval Boolean variable.

Interval Boolean variable $[(\overleftarrow{a}, \overrightarrow{a}), (\overleftarrow{b}, \overrightarrow{b})]$ means a value between $(\overleftarrow{a}, \overrightarrow{a})$ and $(\overleftarrow{b}, \overrightarrow{b})$. DF interval Boolean variable $[(\overleftarrow{a}, \overrightarrow{a}), (\overleftarrow{b}, \overrightarrow{b})]/(\overleftarrow{p}, \overrightarrow{p})$ means we use $(\overleftarrow{p}, \overrightarrow{p})$ to describe the possibility of a real number locates in $[(\overleftarrow{a}, \overrightarrow{a}), (\overleftarrow{b}, \overrightarrow{b})]$.

(3) Dynamic fuzzy linguistic Boolean variable

Using natural language to describe the true degree is one kind of the most intuitive way. This so-called “true and false” is actually a group of natural language vocabulary expressed, such as using the phrase $TV = \{\text{the most true, quite true, very true, almost true, ...}\}$ to describe “degree of true and false”. Although these words in different natural languages, which are sometimes difficult to distinguish the boundaries, appear different representations, their meaning can be correctly understood and will not cause misunderstanding.

In fact, according to the specific requirements of the problem, we can also assume the external variable of language set, that is, DF linguistic Boolean variable, as long as their semantics can be correctly understood, and will not cause misunderstanding.

8.3.2 DF proposition logic formation

In the real word, there are often the following sentences.

- (1) It becomes fine after the rain.
- (2) She is almost as beautiful as her mother.

In these sentences, “almost as beautiful as” and “becomes fine” have the character of dynamic fuzzy.

Definition 8.15 In a declarative sentence with the character of dynamic fuzzy, we note that as DFL proposition. This is expressed with capital letters A, B, C, For a DF proposition, there is no absolute true or false. We can only ask how about the dynamic fuzzy true or false degree.

Definition 8.16 We use a DF number $(\overleftarrow{a}, \overrightarrow{a}) \in [0, 1] \times [\leftarrow, \rightarrow]$ to represent and measure the truth or falsehood of a DF proposition. We call that degree of truth and falsehood. We usually use lower case letters $(\overleftarrow{a}, \overrightarrow{a}), (\overleftarrow{b}, \overrightarrow{b}), (\overleftarrow{c}, \overrightarrow{c}), \dots$ to express this.

Definition 8.17 A variable of DF proposition is the variable of closed interval $[0, 1] \times [\leftarrow, \rightarrow]$. We call that as DF proposition variable. We usually use lower case letters to express this.

For DF variable $(\bar{x}, \vec{x}), (\bar{y}, \vec{y}) \in [0, 1] \times [\leftarrow, \rightarrow]$, the rules are as follows: (notes: $(\bar{x}, \vec{x}) = \bar{x}$ or \vec{x} , $\max(\bar{x}, \vec{x}) \underline{\Delta} \bar{x}$, $\min(\bar{x}, \vec{x}) \underline{\Delta} \vec{x}$)

(1) Negation “ \neg ”.

For example: The negation of (\bar{x}, \vec{x}) is $\overline{(\bar{x}, \vec{x})}$, and $\overline{(\bar{x}, \vec{x})} \underline{\Delta} ((\bar{1} - \bar{x}), (\bar{1} - \vec{x}))$;

(2) Disjunction “ \vee ”.

For example: The disjunction of (\bar{x}, \vec{x}) and (\bar{y}, \vec{y}) is $(\bar{x}, \vec{x}) \vee (\bar{y}, \vec{y}) \underline{\Delta} \max((\bar{x}, \vec{x}), (\bar{y}, \vec{y}))$;

(3) Conjunction “ \wedge ”.

For example: The conjunction of (\bar{x}, \vec{x}) and (\bar{y}, \vec{y}) is $(\bar{x}, \vec{x}) \wedge (\bar{y}, \vec{y}) \underline{\Delta} \min((\bar{x}, \vec{x}), (\bar{y}, \vec{y}))$;

(4) Condition “ \rightarrow ”.

For example: $(\bar{x}, \vec{x}) \rightarrow (\bar{y}, \vec{y}) \Leftrightarrow \overline{(\bar{x}, \vec{x})} \vee (\bar{y}, \vec{y}) \underline{\Delta} \max((\bar{x}, \vec{x}), (\bar{y}, \vec{y}))$;

(5) Bi-condition “ \leftrightarrow ”.

For example: $(\bar{x}, \vec{x}) \leftrightarrow (\bar{y}, \vec{y})$

$\Leftrightarrow \underline{\Delta} ((\bar{x}, \vec{x}) \rightarrow (\bar{y}, \vec{y})) \wedge ((\bar{y}, \vec{y}) \rightarrow (\bar{x}, \vec{x}))$

$\Leftrightarrow \underline{\Delta} ((\bar{x}, \vec{x}) \vee (\bar{y}, \vec{y})) \wedge ((\bar{y}, \vec{y}) \vee (\bar{x}, \vec{x}))$

$\Leftrightarrow \underline{\Delta} \min(\max((\bar{x}, \vec{x}), (\bar{y}, \vec{y})), \max((\bar{y}, \vec{y}), (\bar{x}, \vec{x})))$.

For any DF proposition $(\bar{x}, \vec{x})P, (\bar{y}, \vec{y})Q, \overline{(\bar{x}, \vec{x})P}, (\bar{x}, \vec{x})P \vee (\bar{y}, \vec{y})Q, (\bar{x}, \vec{x})P \rightarrow (\bar{y}, \vec{y})Q$ is a DF proposition.

Definition 8.18 We can define dynamic fuzzy calculus logic formation as

(1) Single DF proposition is a well-formed formula.

(2) If $(\bar{x}, \vec{x})P$ is a well-formed formula, then $\overline{(\bar{x}, \vec{x})P}$ also is well-formed formula.

(3) If $(\bar{x}, \vec{x})P$ and $(\bar{y}, \vec{y})Q$ is a well-formed formula, then

$(\bar{x}, \vec{x})P \vee (\bar{y}, \vec{y})Q, (\bar{x}, \vec{x})P \wedge (\bar{y}, \vec{y})Q$,

$(\bar{x}, \vec{x})P \rightarrow (\bar{y}, \vec{y})Q, (\bar{x}, \vec{x})P \leftrightarrow (\bar{y}, \vec{y})Q$

are also well-formed formula.

(4) A symbol string is a well-formed formula if and only if the symbol is constituted of variable coupling words, brackets and proposition which is obtained by finitely applying (1)(2) and (3).

The main law of DFL is as follows:

(1) Idempotent law

$(\bar{x}, \vec{x})A \vee (\bar{x}, \vec{x})A = (\bar{x}, \vec{x})A$

$(\bar{x}, \vec{x})A \wedge (\bar{x}, \vec{x})A = (\bar{x}, \vec{x})A$

(2) Law of commutation

$(\bar{x}, \vec{x})A \vee (\bar{y}, \vec{y})B = (\bar{y}, \vec{y})B \vee (\bar{x}, \vec{x})A$

$(\bar{x}, \vec{x})A \wedge (\bar{y}, \vec{y})B = (\bar{y}, \vec{y})B \wedge (\bar{x}, \vec{x})A$

(3) Law of association

$$(\overleftarrow{x}, \overrightarrow{x})A \vee ((\overleftarrow{y}, \overrightarrow{y})B \vee (\overleftarrow{c}, \overrightarrow{c})C) = ((\overleftarrow{x}, \overrightarrow{x})A \vee (\overleftarrow{y}, \overrightarrow{y})B) \vee (\overleftarrow{c}, \overrightarrow{c})C$$

$$(\overleftarrow{x}, \overrightarrow{x})A \wedge ((\overleftarrow{y}, \overrightarrow{y})B \wedge (\overleftarrow{c}, \overrightarrow{c})C) = ((\overleftarrow{x}, \overrightarrow{x})A \wedge (\overleftarrow{y}, \overrightarrow{y})B) \wedge (\overleftarrow{c}, \overrightarrow{c})C$$

(4) Absorption law

$$(\overleftarrow{x}, \overrightarrow{x})A \vee ((\overleftarrow{y}, \overrightarrow{y})B \wedge (\overleftarrow{x}, \overrightarrow{x})A) = (\overleftarrow{x}, \overrightarrow{x})A$$

$$(\overleftarrow{x}, \overrightarrow{x})A \wedge ((\overleftarrow{y}, \overrightarrow{y})B \vee (\overleftarrow{x}, \overrightarrow{x})A) = (\overleftarrow{x}, \overrightarrow{x})A$$

(5) De Morgan Rule

$$\overline{(\overleftarrow{x}, \overrightarrow{x})A \wedge (\overleftarrow{y}, \overrightarrow{y})B} = \overline{(\overleftarrow{x}, \overrightarrow{x})A} \vee \overline{(\overleftarrow{y}, \overrightarrow{y})B}$$

$$\overline{(\overleftarrow{x}, \overrightarrow{x})A \vee (\overleftarrow{y}, \overrightarrow{y})B} = \overline{(\overleftarrow{x}, \overrightarrow{x})A} \wedge \overline{(\overleftarrow{y}, \overrightarrow{y})B}$$

(6) Operational Rule of Constant

$$A \vee (\overleftarrow{x}, \overrightarrow{x})A = A$$

$$A \wedge (\overleftarrow{x}, \overrightarrow{x})A = (\overleftarrow{x}, \overrightarrow{x})A$$

Definition 8.19 Assume P is a DFL formula; if for all the variables in P there are $T(p) \geq (\overleftarrow{a}, \overrightarrow{a}), (\overleftarrow{a}, \overrightarrow{a}) \in [0, 1] \times [\leftarrow, \rightarrow]$, then we call that the DFL formula P is constant true formula of $(\overleftarrow{a}, \overrightarrow{a})$.

Especially, when $T(P) \geq (\frac{1}{2}, \frac{1}{2})$, we call the above formula P as a DF constant true formula. When $T(P) < (\frac{1}{2}, \frac{1}{2})$, we call the formula P as a DF constant false formula.

We call variable $(\overleftarrow{x}, \overrightarrow{x})$ and $\overline{(\overleftarrow{x}, \overrightarrow{x})}$ as word (note as L), and the disjunction of word $L_1 \vee L_2 \vee L_3 \vee \dots \vee L_r$ as clause; we note C , the conjunction of word $L_1 \wedge L_2 \wedge L_3 \wedge \dots \wedge L_r$ as word group, and we note that ϕ .

Theorem 8.12 The necessary and sufficient condition of clause C is DF true is clause C contains variable pair $((\overleftarrow{x}, \overrightarrow{x}), (\overleftarrow{x}, \overrightarrow{x}))$. The necessary and sufficient condition of word group ϕ is DF false is word group ϕ contains variable pair $((\overleftarrow{x}, \overrightarrow{x}), \overline{(\overleftarrow{x}, \overrightarrow{x})})$.

Definition 8.20 If DF formula f represents disjunction of word group $\phi_i, i = 1, 2, 3, \dots, p$, namely:

$$f = \phi_1 \vee \phi_2 \vee \dots \vee \phi_p,$$

then we call the formula as the disjunction normal form of f .

If the DF formula f represents conjunction of clause, namely,

$$f = C_1 \wedge C_2 \wedge \dots \wedge C_p,$$

then we call the formula as the conjunction normal form of f .

Theorem 8.13 The necessary and sufficient condition of disjunction normal form $P = \phi_1 \vee \phi_2 \vee \dots \vee \phi_p$ is false is all word group ϕ_i are DF false.

The necessary and sufficient condition of conjunction normal form $Q = C_1 \wedge C_2 \wedge \dots \wedge C_p$ is DF true is all clause C_i are DF true.

8.4 Dynamic fuzzy lattice and its property

Definition 8.21 We call $((\overleftarrow{K}, \overrightarrow{K}), \leq)$ as sub-ordered set if the relations “ \leq ” on $(\overleftarrow{K}, \overrightarrow{K})$ meet the following conditions:

(1) Reflexivity: $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \leq (\overleftarrow{\alpha}, \overrightarrow{\alpha})$

(2) Transitive: $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \leq (\overleftarrow{\beta}, \overrightarrow{\beta}), (\overleftarrow{\beta}, \overrightarrow{\beta}) \leq (\overleftarrow{\gamma}, \overrightarrow{\gamma}) \Rightarrow (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \leq (\overleftarrow{\gamma}, \overrightarrow{\gamma})$.

We call $((\overleftarrow{P}, \overrightarrow{P}), \leq)$ as partial order set. If it is sub-ordered set and meets (1) and (2):

(3) Anti symmetry: $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \leq (\overleftarrow{\beta}, \overrightarrow{\beta}), (\overleftarrow{\beta}, \overrightarrow{\beta}) \leq (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \Rightarrow (\overleftarrow{\alpha}, \overrightarrow{\alpha}) = (\overleftarrow{\beta}, \overrightarrow{\beta})$.

In a partial order set $(\overleftarrow{P}, \overrightarrow{P})$, there not necessarily is $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \leq (\overleftarrow{\beta}, \overrightarrow{\beta})$ or $(\overleftarrow{\beta}, \overrightarrow{\beta}) \leq (\overleftarrow{\alpha}, \overrightarrow{\alpha})$. If for any $(\overleftarrow{\alpha}, \overrightarrow{\alpha}), (\overleftarrow{\beta}, \overrightarrow{\beta}) \in (\overleftarrow{P}, \overrightarrow{P})$, there must be $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \leq (\overleftarrow{\beta}, \overrightarrow{\beta})$ or $(\overleftarrow{\beta}, \overrightarrow{\beta}) \leq (\overleftarrow{\alpha}, \overrightarrow{\alpha})$, we call that $(\overleftarrow{P}, \overrightarrow{P})$ is linear order set.

For a partial order set $((\overleftarrow{P}, \overrightarrow{P}), \leq)$, if $(\overleftarrow{H}, \overrightarrow{H}) \subset (\overleftarrow{P}, \overrightarrow{P})$, $\exists (\overleftarrow{u}, \overrightarrow{u}) \in (\overleftarrow{P}, \overrightarrow{P})$, $\forall (\overleftarrow{h}, \overrightarrow{h}) \in (\overleftarrow{H}, \overrightarrow{H})$, let satisfy $(\overleftarrow{h}, \overrightarrow{h}) \leq (\overleftarrow{u}, \overrightarrow{u})$, then we call $(\overleftarrow{u}, \overrightarrow{u})$ as the upper bound of $(\overleftarrow{H}, \overrightarrow{H})$; if the upper bound set of $(\overleftarrow{H}, \overrightarrow{H})$ has a minimum element, then we call that as the minimum upper bound of $(\overleftarrow{H}, \overrightarrow{H})$. We note that as

$$\sup(\overleftarrow{H}, \overrightarrow{H}) \text{ or } \bigvee_{(\overleftarrow{h}, \overrightarrow{h}) \in (\overleftarrow{H}, \overrightarrow{H})} (\overleftarrow{h}, \overrightarrow{h}).$$

If $\exists (\overleftarrow{v}, \overrightarrow{v}) \in (\overleftarrow{P}, \overrightarrow{P})$, $\forall (\overleftarrow{h}, \overrightarrow{h}) \in (\overleftarrow{H}, \overrightarrow{H})$, $(\overleftarrow{v}, \overrightarrow{v}) \leq (\overleftarrow{h}, \overrightarrow{h})$, we call $(\overleftarrow{v}, \overrightarrow{v})$ as the lower bound of $(\overleftarrow{H}, \overrightarrow{H})$; if the lower bound set of $(\overleftarrow{H}, \overrightarrow{H})$ has a maximum element, we call that as the maximum lower bound of $(\overleftarrow{H}, \overrightarrow{H})$. We note that as

$$\inf(\overleftarrow{H}, \overrightarrow{H}) \text{ or } \bigwedge_{(\overleftarrow{h}, \overrightarrow{h}) \in (\overleftarrow{H}, \overrightarrow{H})} (\overleftarrow{h}, \overrightarrow{h}).$$

Especially, we note the minimum upper bound of two elements $(\overleftarrow{a}, \overrightarrow{a})$ and $(\overleftarrow{b}, \overrightarrow{b})$ as $(\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{b}, \overrightarrow{b})$, and the maximum lower bound as $(\overleftarrow{a}, \overrightarrow{a}) \wedge (\overleftarrow{b}, \overrightarrow{b})$.

Thus, we have:

Theorem 8.14 Assume $((\overleftarrow{P}, \overrightarrow{P}), \leq)$ is partial order set; the minimum upper bound and the maximum lower bound have the following properties:

(1) $(\overleftarrow{a}, \overrightarrow{a}) \wedge (\overleftarrow{b}, \overrightarrow{b}) \leq (\overleftarrow{a}, \overrightarrow{a})$

$(\overleftarrow{a}, \overrightarrow{a}) \wedge (\overleftarrow{b}, \overrightarrow{b}) \leq (\overleftarrow{b}, \overrightarrow{b})$

$(\overleftarrow{a}, \overrightarrow{a}) \leq (\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{b}, \overrightarrow{b})$

$(\overleftarrow{b}, \overrightarrow{b}) \leq (\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{b}, \overrightarrow{b})$

(2) $(\overleftarrow{a}, \overrightarrow{a}) \wedge (\overleftarrow{b}, \overrightarrow{b}) = (\overleftarrow{b}, \overrightarrow{b}) \wedge (\overleftarrow{a}, \overrightarrow{a})$

$(\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{b}, \overrightarrow{b}) = (\overleftarrow{b}, \overrightarrow{b}) \vee (\overleftarrow{a}, \overrightarrow{a})$

- (3) $(\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{a}, \overrightarrow{a}) = (\overleftarrow{a}, \overrightarrow{a}), (\overleftarrow{a}, \overrightarrow{a}) \wedge (\overleftarrow{a}, \overrightarrow{a}) = (\overleftarrow{a}, \overrightarrow{a})$
- (4) $(\overleftarrow{a}, \overrightarrow{a}) \leq (\overleftarrow{b}, \overrightarrow{b}) \Leftrightarrow (\overleftarrow{a}, \overrightarrow{a}) \wedge (\overleftarrow{b}, \overrightarrow{b}) = (\overleftarrow{a}, \overrightarrow{a})$
 $(\overleftarrow{a}, \overrightarrow{a}) \leq (\overrightarrow{b}, \overrightarrow{b}) \Leftrightarrow (\overleftarrow{a}, \overrightarrow{a}) \vee (\overrightarrow{b}, \overrightarrow{b}) = (\overrightarrow{b}, \overrightarrow{b})$
- (5) $(\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{b}, \overrightarrow{b}) \wedge (\overrightarrow{c}, \overrightarrow{c}) = ((\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{b}, \overrightarrow{b})) \wedge (\overrightarrow{c}, \overrightarrow{c})$
 $(\overleftarrow{a}, \overrightarrow{a}) \vee ((\overrightarrow{b}, \overrightarrow{b}) \vee (\overrightarrow{c}, \overrightarrow{c})) = ((\overleftarrow{a}, \overrightarrow{a}) \vee (\overrightarrow{b}, \overrightarrow{b})) \vee (\overrightarrow{c}, \overrightarrow{c})$
- (6) $(\overleftarrow{a}, \overrightarrow{a}) \wedge ((\overleftarrow{a}, \overrightarrow{a}) \vee (\overrightarrow{b}, \overrightarrow{b})) = (\overleftarrow{a}, \overrightarrow{a})$
 $(\overleftarrow{a}, \overrightarrow{a}) \vee ((\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{b}, \overrightarrow{b})) = (\overleftarrow{a}, \overrightarrow{a})$

Proof: (1), (2), and (3) are obvious.

Now we prove (4): if $(\overleftarrow{a}, \overrightarrow{a}) \leq (\overrightarrow{b}, \overrightarrow{b})$, then $(\overleftarrow{a}, \overrightarrow{a})$ is the lower bound of $(\overleftarrow{a}, \overrightarrow{a})$ and $(\overrightarrow{b}, \overrightarrow{b})$. Thus, $(\overleftarrow{a}, \overrightarrow{a}) \leq (\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{b}, \overrightarrow{b})$ according to (1): $(\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{b}, \overrightarrow{b}) \leq (\overleftarrow{a}, \overrightarrow{a})$.

Conversely, if $(\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{b}, \overrightarrow{b}) = (\overleftarrow{a}, \overrightarrow{a})$ according to (1),

$$(\overleftarrow{a}, \overrightarrow{a}) = (\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{b}, \overrightarrow{b}) \leq (\overrightarrow{b}, \overrightarrow{b})$$

$$\text{Then, } (\overleftarrow{a}, \overrightarrow{a}) \leq (\overrightarrow{b}, \overrightarrow{b}) \Leftrightarrow (\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{b}, \overrightarrow{b}) = (\overrightarrow{b}, \overrightarrow{b}).$$

$$\text{Corollaries: } (\overleftarrow{a}, \overrightarrow{a}) \leq (\overrightarrow{b}, \overrightarrow{b}) \Leftrightarrow (\overleftarrow{a}, \overrightarrow{a}) \vee (\overrightarrow{b}, \overrightarrow{b}) = (\overrightarrow{b}, \overrightarrow{b})$$

Thus, we prove (4).

Having $(\overrightarrow{b}, \overrightarrow{b}) \wedge (\overrightarrow{c}, \overrightarrow{c}) \leq (\overrightarrow{b}, \overrightarrow{b}), (\overrightarrow{b}, \overrightarrow{b}) \wedge (\overrightarrow{c}, \overrightarrow{c}) \leq (\overrightarrow{c}, \overrightarrow{c})$, then

$$(\overleftarrow{a}, \overrightarrow{a}) \wedge ((\overrightarrow{b}, \overrightarrow{b}) \vee (\overrightarrow{c}, \overrightarrow{c})) \leq (\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{b}, \overrightarrow{b}),$$

$$(\overleftarrow{a}, \overrightarrow{a}) \wedge ((\overrightarrow{b}, \overrightarrow{b}) \vee (\overrightarrow{c}, \overrightarrow{c})) \leq (\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{c}, \overrightarrow{c}) \leq (\overrightarrow{c}, \overrightarrow{c}).$$

Thus, $(\overleftarrow{a}, \overrightarrow{a}) \wedge ((\overrightarrow{b}, \overrightarrow{b}) \wedge (\overrightarrow{c}, \overrightarrow{c})) \leq ((\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{b}, \overrightarrow{b})) \wedge (\overrightarrow{c}, \overrightarrow{c})$.

$$\text{Corollaries: } ((\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{b}, \overrightarrow{b})) \wedge (\overrightarrow{c}, \overrightarrow{c}) \leq (\overleftarrow{a}, \overrightarrow{a}) \wedge ((\overrightarrow{b}, \overrightarrow{b}) \vee (\overrightarrow{c}, \overrightarrow{c}))$$

The prior of (5) is proved. The after is similar.

Having $(\overleftarrow{a}, \overrightarrow{a}) \leq (\overleftarrow{a}, \overrightarrow{a}), (\overleftarrow{a}, \overrightarrow{a}) \leq (\overleftarrow{a}, \overrightarrow{a}) \vee (\overrightarrow{b}, \overrightarrow{b})$, then

$$(\overleftarrow{a}, \overrightarrow{a}) \leq (\overleftarrow{a}, \overrightarrow{a}) \wedge ((\overleftarrow{a}, \overrightarrow{a}) \vee (\overrightarrow{b}, \overrightarrow{b})) \leq (\overleftarrow{a}, \overrightarrow{a}); \text{ thus,}$$

$$(\overleftarrow{a}, \overrightarrow{a}) \wedge ((\overleftarrow{a}, \overrightarrow{a}) \vee (\overrightarrow{b}, \overrightarrow{b})) = (\overleftarrow{a}, \overrightarrow{a}).$$

$$\text{Corollaries: } (\overleftarrow{a}, \overrightarrow{a}) \vee ((\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{b}, \overrightarrow{b})) = (\overleftarrow{a}, \overrightarrow{a})$$

Definition 8.22 We call partial order set $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ semilattice; when $\forall (\overleftarrow{\alpha}, \overrightarrow{\alpha}), (\overleftarrow{\beta}, \overrightarrow{\beta}) \in (\overleftarrow{L}, \overrightarrow{L})$, we have $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee (\overleftarrow{\beta}, \overrightarrow{\beta}) \in (\overleftarrow{L}, \overrightarrow{L})$ or $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge (\overleftarrow{\beta}, \overrightarrow{\beta}) \in (\overleftarrow{L}, \overrightarrow{L})$.

When $\forall (\overleftarrow{\alpha}, \overrightarrow{\alpha}), (\overleftarrow{\beta}, \overrightarrow{\beta}) \in (\overleftarrow{L}, \overrightarrow{L})$, we have $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee (\overleftarrow{\beta}, \overrightarrow{\beta}) \in (\overleftarrow{L}, \overrightarrow{L}), (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge (\overleftarrow{\beta}, \overrightarrow{\beta}) \in (\overleftarrow{L}, \overrightarrow{L})$, then we call the partial order set $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ as lattice.

Theorem 8.15 The necessary and sufficient conditions of set $(\overleftarrow{L}, \overrightarrow{L})$ are a lattice:

- (1) There are two operations \vee, \wedge , for any $(\overleftarrow{a}, \overrightarrow{a}), (\overrightarrow{b}, \overrightarrow{b}) \in (\overleftarrow{L}, \overrightarrow{L})$, satisfying $(\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{b}, \overrightarrow{b}) \in (\overleftarrow{L}, \overrightarrow{L}), (\overleftarrow{a}, \overrightarrow{a}) \vee (\overrightarrow{b}, \overrightarrow{b}) \in (\overleftarrow{L}, \overrightarrow{L})$.
- (2) $(\overleftarrow{a}, \overrightarrow{a}) \vee (\overrightarrow{b}, \overrightarrow{b}) = (\overrightarrow{b}, \overrightarrow{b}) \vee (\overleftarrow{a}, \overrightarrow{a}), (\overrightarrow{b}, \overrightarrow{b}) \wedge (\overleftarrow{a}, \overrightarrow{a}) = (\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{b}, \overrightarrow{b})$ (Commutative law)
- (3) $((\overleftarrow{a}, \overrightarrow{a}) \vee (\overrightarrow{b}, \overrightarrow{b})) \vee (\overrightarrow{c}, \overrightarrow{c}) = (\overleftarrow{a}, \overrightarrow{a}) \vee ((\overrightarrow{b}, \overrightarrow{b}) \vee (\overrightarrow{c}, \overrightarrow{c}))$
 $((\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{b}, \overrightarrow{b}) \wedge (\overrightarrow{c}, \overrightarrow{c})) = ((\overleftarrow{a}, \overrightarrow{a}) \wedge (\overrightarrow{b}, \overrightarrow{b})) \wedge (\overrightarrow{c}, \overrightarrow{c})$ (Associative law)

$$(4) (\overleftarrow{a}, \overrightarrow{a}) \wedge ((\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{b}, \overrightarrow{b})) = (\overleftarrow{a}, \overrightarrow{a})$$

$$(\overleftarrow{a}, \overrightarrow{a}) \vee ((\overleftarrow{a}, \overrightarrow{a}) \wedge (\overleftarrow{b}, \overrightarrow{b})) = (\overleftarrow{a}, \overrightarrow{a}) \text{ (Absorption law)}$$

Proof: (sufficient)

(1): By known, we have $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ is lattice, according the definition of lattice. We prove (1).

(2): Assume $(\overleftarrow{a}, \overrightarrow{a}) \leq (\overleftarrow{b}, \overrightarrow{b})$, then $(\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{b}, \overrightarrow{b}) = (\overleftarrow{b}, \overrightarrow{b})$, $(\overleftarrow{b}, \overrightarrow{b}) \vee (\overleftarrow{a}, \overrightarrow{a}) = (\overleftarrow{b}, \overrightarrow{b})$
 $(\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{b}, \overrightarrow{b}) = (\overleftarrow{b}, \overrightarrow{b}) \vee (\overleftarrow{a}, \overrightarrow{a})$; in the same way, $(\overleftarrow{b}, \overrightarrow{b}) \wedge (\overleftarrow{a}, \overrightarrow{a}) = (\overleftarrow{a}, \overrightarrow{a}) \wedge (\overleftarrow{b}, \overrightarrow{b})$.

When $(\overleftarrow{b}, \overrightarrow{b}) \leq (\overleftarrow{a}, \overrightarrow{a})$, in the same way, we can prove (2).

(3) (4): Corollaries, according to the method in (2) (necessary):

(1) For any $(\overleftarrow{a}, \overrightarrow{a}), (\overleftarrow{b}, \overrightarrow{b}) \in (\overleftarrow{L}, \overrightarrow{L})$, there must be $(\overleftarrow{a}, \overrightarrow{a}) \leq (\overleftarrow{a}, \overrightarrow{a})$, namely, set $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ is reflex.

(2) Assume $(\overleftarrow{a}, \overrightarrow{a}) \leq (\overleftarrow{b}, \overrightarrow{b})$ and $(\overleftarrow{b}, \overrightarrow{b}) \leq (\overleftarrow{a}, \overrightarrow{a})$. Then,

$$(\overleftarrow{a}, \overrightarrow{a}) \leq (\overleftarrow{b}, \overrightarrow{b}) \Rightarrow (\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{b}, \overrightarrow{b}) = (\overleftarrow{b}, \overrightarrow{b})$$

$$(\overleftarrow{b}, \overrightarrow{b}) \leq (\overleftarrow{a}, \overrightarrow{a}) \Rightarrow (\overleftarrow{b}, \overrightarrow{b}) \vee (\overleftarrow{a}, \overrightarrow{a}) = (\overleftarrow{a}, \overrightarrow{a})$$

$$\text{According to (2), } (\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{b}, \overrightarrow{b}) = (\overleftarrow{b}, \overrightarrow{b}) \vee (\overleftarrow{a}, \overrightarrow{a}) \Rightarrow (\overleftarrow{a}, \overrightarrow{a}) = (\overleftarrow{b}, \overrightarrow{b})$$

Then, set $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ is antisymmetry.

(3) Assume $(\overleftarrow{a}, \overrightarrow{a}) \leq (\overleftarrow{b}, \overrightarrow{b}), (\overleftarrow{b}, \overrightarrow{b}) \leq (\overleftarrow{c}, \overrightarrow{c})$. Then,

$$((\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{b}, \overrightarrow{b})) \vee (\overleftarrow{c}, \overrightarrow{c}) = (\overleftarrow{b}, \overrightarrow{b}) \vee (\overleftarrow{c}, \overrightarrow{c}) = (\overleftarrow{c}, \overrightarrow{c})$$

$$(\overleftarrow{a}, \overrightarrow{a}) \vee ((\overleftarrow{b}, \overrightarrow{b}) \vee (\overleftarrow{c}, \overrightarrow{c})) = (\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{c}, \overrightarrow{c})$$

$$\text{According to (3) } ((\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{b}, \overrightarrow{b})) \vee (\overleftarrow{c}, \overrightarrow{c}) = (\overleftarrow{a}, \overrightarrow{a}) \vee ((\overleftarrow{b}, \overrightarrow{b}) \vee (\overleftarrow{c}, \overrightarrow{c}))$$

$$\Rightarrow (\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{c}, \overrightarrow{c}) = (\overleftarrow{c}, \overrightarrow{c}) \Rightarrow (\overleftarrow{a}, \overrightarrow{a}) \leq (\overleftarrow{c}, \overrightarrow{c})$$

Namely $(\overleftarrow{a}, \overrightarrow{a}) \leq (\overleftarrow{b}, \overrightarrow{b}), (\overleftarrow{b}, \overrightarrow{b}) \leq (\overleftarrow{c}, \overrightarrow{c}) \Rightarrow (\overleftarrow{a}, \overrightarrow{a}) \leq (\overleftarrow{c}, \overrightarrow{c})$,

Then, set $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ is transitive.

(4) According to (1), for any $(\overleftarrow{a}, \overrightarrow{a}), (\overleftarrow{b}, \overrightarrow{b}) \in (\overleftarrow{L}, \overrightarrow{L})$, there are $(\overleftarrow{a}, \overrightarrow{a}) \wedge (\overleftarrow{b}, \overrightarrow{b}) \in (\overleftarrow{L}, \overrightarrow{L}), (\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{b}, \overrightarrow{b}) \in (\overleftarrow{L}, \overrightarrow{L})$. Namely, the maximum lower bound and the minimum upper bound of any two elements are in the set. By (1)–(4), we can know that set $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ is lattice.

According to the necessary and sufficient condition, the theorem holds.

(Prove up).

Theorem 8.16 Assume $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ is lattice, then we have the following properties:

$$(1) (\overleftarrow{b}, \overrightarrow{b}) \leq (\overleftarrow{c}, \overrightarrow{c}) \Rightarrow (\overleftarrow{a}, \overrightarrow{a}) \wedge (\overleftarrow{b}, \overrightarrow{b}) \leq (\overleftarrow{a}, \overrightarrow{a}) \wedge (\overleftarrow{c}, \overrightarrow{c})$$

$$(\overleftarrow{b}, \overrightarrow{b}) \leq (\overleftarrow{c}, \overrightarrow{c}) \Rightarrow (\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{b}, \overrightarrow{b}) \leq (\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{c}, \overrightarrow{c})$$

$$(2) (\overleftarrow{a}, \overrightarrow{a}) \vee ((\overleftarrow{b}, \overrightarrow{b}) \wedge (\overleftarrow{c}, \overrightarrow{c})) \leq ((\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{b}, \overrightarrow{b})) \wedge ((\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{c}, \overrightarrow{c}))$$

$$(\overleftarrow{a}, \overrightarrow{a}) \wedge ((\overleftarrow{b}, \overrightarrow{b}) \vee (\overleftarrow{c}, \overrightarrow{c})) \geq ((\overleftarrow{a}, \overrightarrow{a}) \wedge (\overleftarrow{b}, \overrightarrow{b})) \vee ((\overleftarrow{a}, \overrightarrow{a}) \wedge (\overleftarrow{c}, \overrightarrow{c}))$$

$$(3) (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \leq (\overleftarrow{c}, \overrightarrow{c}) \Leftrightarrow (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee ((\overleftarrow{b}, \overrightarrow{b}) \wedge (\overleftarrow{c}, \overrightarrow{c})) \leq ((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee ((\overleftarrow{b}, \overrightarrow{b})) \wedge (\overleftarrow{c}, \overrightarrow{c})) \\ \Leftrightarrow (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge ((\overleftarrow{b}, \overrightarrow{b}) \vee (\overleftarrow{c}, \overrightarrow{c})) \geq ((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge (\overleftarrow{b}, \overrightarrow{b})) \vee (\overleftarrow{c}, \overrightarrow{c})$$

Proof: (1) By Theorem 8.14: $(\overleftarrow{b}, \overrightarrow{b}) \leq (\overleftarrow{c}, \overrightarrow{c}) \Leftrightarrow (\overleftarrow{b}, \overrightarrow{b}) \wedge (\overleftarrow{c}, \overrightarrow{c}) = (\overleftarrow{b}, \overrightarrow{b})$, then $((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge ((\overleftarrow{b}, \overrightarrow{b})) \wedge ((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge (\overleftarrow{c}, \overrightarrow{c}))) = ((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge (\overleftarrow{\alpha}, \overrightarrow{\alpha})) \wedge ((\overleftarrow{b}, \overrightarrow{b}) \wedge (\overleftarrow{c}, \overrightarrow{c})) = (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge (\overleftarrow{b}, \overrightarrow{b})$, thus $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge (\overleftarrow{b}, \overrightarrow{b}) \leq (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge (\overleftarrow{c}, \overrightarrow{c})$.

Corollaries: $((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee (\overleftarrow{b}, \overrightarrow{b})) \leq ((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee (\overleftarrow{c}, \overrightarrow{c}))$

- (2) Hence, $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \leq ((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee (\overleftarrow{b}, \overrightarrow{b})) \wedge ((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee (\overleftarrow{c}, \overrightarrow{c}))$ and $((\overleftarrow{b}, \overrightarrow{b}) \wedge (\overleftarrow{c}, \overrightarrow{c})) \leq ((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee (\overleftarrow{b}, \overrightarrow{b})) \wedge ((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee (\overleftarrow{c}, \overrightarrow{c}))$; then the first part of (2) has been proved. In the same way, we can prove the second part.
- (3) According to $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \leq (\overleftarrow{c}, \overrightarrow{c}) \Leftrightarrow (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee (\overleftarrow{c}, \overrightarrow{c}) = (\overleftarrow{c}, \overrightarrow{c}) \Leftrightarrow (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge (\overleftarrow{c}, \overrightarrow{c}) = (\overleftarrow{\alpha}, \overrightarrow{\alpha})$, namely (3).

[Prove up].

Definition 8.23 Lattice $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ is dense; when $\forall (\overleftarrow{\alpha}, \overrightarrow{\alpha}), (\overleftarrow{\beta}, \overrightarrow{\beta}) \in (\overleftarrow{L}, \overrightarrow{L})$ and $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) < (\overleftarrow{\beta}, \overrightarrow{\beta})$, there exists $(\overleftarrow{\gamma}, \overrightarrow{\gamma}) \in (\overleftarrow{L}, \overrightarrow{L})$, let satisfy $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) < (\overleftarrow{\gamma}, \overrightarrow{\gamma}) < (\overleftarrow{\beta}, \overrightarrow{\beta})$.

Obviously, a dense complete lattice has the following property: $\vee\{(\overleftarrow{\alpha}, \overrightarrow{\alpha}); (\overleftarrow{\alpha}, \overrightarrow{\alpha}) < (\overleftarrow{\beta}, \overrightarrow{\beta})\} = (\overleftarrow{\beta}, \overrightarrow{\beta})$; $\wedge\{(\overleftarrow{\alpha}, \overrightarrow{\alpha}); (\overleftarrow{\alpha}, \overrightarrow{\alpha}) > (\overleftarrow{\beta}, \overrightarrow{\beta})\} = (\overleftarrow{\beta}, \overrightarrow{\beta})$.

Definition 8.24 Assume $((\overleftarrow{K}, \overrightarrow{K}), \leq)$ is a sub-ordered set; mapping $N: (\overleftarrow{K}, \overrightarrow{K}) \rightarrow (\overleftarrow{K}, \overrightarrow{K})$ is order preserving mapping. When $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \leq (\overleftarrow{b}, \overrightarrow{b})$ ($\forall (\overleftarrow{\alpha}, \overrightarrow{\alpha}), (\overleftarrow{b}, \overrightarrow{b}) \in (\overleftarrow{K}, \overrightarrow{K})$), there is $N(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \leq N(\overleftarrow{b}, \overrightarrow{b})$. When $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \geq (\overleftarrow{b}, \overrightarrow{b})$ ($\forall (\overleftarrow{\alpha}, \overrightarrow{\alpha}), (\overleftarrow{b}, \overrightarrow{b}) \in (\overleftarrow{K}, \overrightarrow{K})$), there is $N(\overleftarrow{b}, \overrightarrow{b}) \leq N(\overleftarrow{\alpha}, \overrightarrow{\alpha})$, and we call N as reverse mapping. If N reverse mapping N satisfies the law of unity, namely, $N(N(\overleftarrow{\alpha}, \overrightarrow{\alpha})) = (\overleftarrow{\alpha}, \overrightarrow{\alpha})$, then we call N as pseudo-complement.

Theorem 8.17 Assume N is the pseudo-complement on lattice $((\overleftarrow{L}, \overrightarrow{L}), \leq)$, then

$$N((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee (\overleftarrow{b}, \overrightarrow{b})) = N(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge N(\overleftarrow{b}, \overrightarrow{b}); N((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge (\overleftarrow{b}, \overrightarrow{b})) = N(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee N(\overleftarrow{b}, \overrightarrow{b}).$$

Proof: By $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \leq ((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee (\overleftarrow{b}, \overrightarrow{b})), (\overleftarrow{b}, \overrightarrow{b}) \leq ((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee (\overleftarrow{b}, \overrightarrow{b}))$, $N((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee (\overleftarrow{b}, \overrightarrow{b})) \leq (N(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge N(\overleftarrow{b}, \overrightarrow{b}))$.

Corollaries: $N((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge (\overleftarrow{b}, \overrightarrow{b})) \geq (N(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee N(\overleftarrow{b}, \overrightarrow{b}))$, then $N(N(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge N(\overleftarrow{b}, \overrightarrow{b})) \geq ((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee (\overleftarrow{b}, \overrightarrow{b}))$; thus, $(N(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge N(\overleftarrow{b}, \overrightarrow{b})) \leq N((\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee (\overleftarrow{b}, \overrightarrow{b}))$. Namely, the first part of theorem has been proved. A similar process is used to prove the other part.

[Prove up].

Definition 8.25 Assume $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ is lattice; if $\forall (\overleftarrow{H}, \overrightarrow{H}) \subset (\overleftarrow{L}, \overrightarrow{L}), (\overleftarrow{H}, \overrightarrow{H})$ has the maximum lower bound and the minimum upper bound, we call $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ as complete lattice. If $\forall (\overleftarrow{H}, \overrightarrow{H}) \subset (\overleftarrow{L}, \overrightarrow{L})$, there is

$$(\overleftarrow{a}, \overrightarrow{a}) \vee (\bigwedge_{(\overleftarrow{h}, \overrightarrow{h}) \in (\overleftarrow{H}, \overrightarrow{H})} (\overleftarrow{h}, \overrightarrow{h})) = \bigwedge_{(\overleftarrow{h}, \overrightarrow{h}) \in (\overleftarrow{H}, \overrightarrow{H})} ((\overleftarrow{a}, \overrightarrow{a}) \vee (\overleftarrow{h}, \overrightarrow{h})) \quad (8.1)$$

$$(\overleftarrow{a}, \overrightarrow{a}) \wedge (\bigvee_{(\overleftarrow{h}, \overrightarrow{h}) \in (\overleftarrow{H}, \overrightarrow{H})} (\overleftarrow{h}, \overrightarrow{h})) = \bigvee_{(\overleftarrow{h}, \overrightarrow{h}) \in (\overleftarrow{H}, \overrightarrow{H})} ((\overleftarrow{a}, \overrightarrow{a}) \wedge (\overleftarrow{h}, \overrightarrow{h})) \quad (8.2)$$

And we call $(\overleftarrow{L}, \overrightarrow{L})$ as complete distributive lattice.

Definition 8.26 Assume $(\overleftarrow{L}, \overrightarrow{L})$ is complete lattice, $(\overleftarrow{a}, \overrightarrow{a}) \in (\overleftarrow{L}, \overrightarrow{L}), (\overleftarrow{A}, \overrightarrow{A}) \subset (\overleftarrow{L}, \overrightarrow{L})$; we call $(\overleftarrow{A}, \overrightarrow{A})$ as a minimal family of $(\overleftarrow{a}, \overrightarrow{a})$. When $(\overleftarrow{A}, \overrightarrow{A})$ meets: (1) $\sup(\overleftarrow{A}, \overrightarrow{A}) = (\overleftarrow{a}, \overrightarrow{a})$, (2) if $(\overleftarrow{B}, \overrightarrow{B}) \subset (\overleftarrow{L}, \overrightarrow{L})$, $\sup(\overleftarrow{B}, \overrightarrow{B}) = (\overleftarrow{a}, \overrightarrow{a})$, then $\forall (\overleftarrow{x}, \overrightarrow{x}) \in (\overleftarrow{A}, \overrightarrow{A}), \exists (\overleftarrow{y}, \overrightarrow{y}) \in (\overleftarrow{B}, \overrightarrow{B}), (\overleftarrow{y}, \overrightarrow{y}) \geq (\overleftarrow{x}, \overrightarrow{x})$, we call $(\overleftarrow{A}, \overrightarrow{A})$ as a very big family of $(\overleftarrow{a}, \overrightarrow{a})$. When $(\overleftarrow{A}, \overrightarrow{A})$ meets: (1) $\inf(\overleftarrow{A}, \overrightarrow{A}) = (\overleftarrow{a}, \overrightarrow{a})$, (2) if $(\overleftarrow{B}, \overrightarrow{B}) \subset (\overleftarrow{L}, \overrightarrow{L})$, $\inf(\overleftarrow{B}, \overrightarrow{B}) = (\overleftarrow{a}, \overrightarrow{a})$, then $\forall (\overleftarrow{x}, \overrightarrow{x}) \in (\overleftarrow{A}, \overrightarrow{A}), \exists (\overleftarrow{y}, \overrightarrow{y}) \in (\overleftarrow{B}, \overrightarrow{B}), (\overleftarrow{y}, \overrightarrow{y}) \leq (\overleftarrow{x}, \overrightarrow{x})$.

Definition 8.27 Assume lattice $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ has $(\overleftarrow{O}, \overrightarrow{O}), (\overleftarrow{I}, \overrightarrow{I})$ [where $(\overleftarrow{O}, \overrightarrow{O})$ is zero element and $(\overleftarrow{I}, \overrightarrow{I})$ is identity]; if $(\overleftarrow{\alpha}, \overrightarrow{\alpha}) \vee (\overleftarrow{\beta}, \overrightarrow{\beta}) = (\overleftarrow{I}, \overrightarrow{I}), (\overleftarrow{\alpha}, \overrightarrow{\alpha}) \wedge (\overleftarrow{\beta}, \overrightarrow{\beta}) = (\overleftarrow{O}, \overrightarrow{O})$, then we note $(\overleftarrow{\beta}, \overrightarrow{\beta})$ as a complement of $(\overleftarrow{\alpha}, \overrightarrow{\alpha})$.

Definition 8.28 Lattice $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ is a complemented lattice when $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ has $(\overleftarrow{O}, \overrightarrow{O}), (\overleftarrow{I}, \overrightarrow{I})$ and the other element has a complement.

Definition 8.29 Lattice $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ is a complemented distributive lattice when lattice $((\overleftarrow{L}, \overrightarrow{L}), \leq)$ is a complemented lattice and meets ((8.1)) and ((8.2)). We call complemented distributive lattice as Boolean lattice.

Index

α -certainty type 221
 α -explanatory type 220

Action Choice 274
Agent mental model 267
And/or tree 237
Approximation function 46
Asymptotically stable 12
Attribute-value learning 214
Auto-evaluating 55
Auto-knowledge 54
Auto-monitoring 55
Autonomic Learning 43
Autonomic Learning Space 56
Autonomic Learning Subspace 57

Batch algorithms 129
Bayesian Logic Programs 41
Belie 269
Belief Revision 274

Capacity 267
Classification 163
Cluster Assumption 169
Concept Lattice 129
Concept Learning 110
Connected graph 251
Cooperative multi-agent learning
algorithm 282
Cost minimization 164
Critical Value Pruning 101
Curse of dimensionality 4

DF Concept Error Boundary 120
DF concept lattice 129
DF concept lattice critical layer construction
algorithm 133
DF Concept Lattice Reduction 135
DF Concept Learning Space Model 115
DF Concept Sample Complexity 120
DF concept tree 142
DF Error Bound Model 124
DF Galois connection 117
DF hypothesis space 118
DF match 236
DF production 236

Direct graph 249
Disconnected graph 251
Discretization 88
Distinguishability 192
Distribution grid 227
Distribution number 183
Division 84
Dundant separation point 146
Dynamic fuzzy association rules 137
Dynamic fuzzy background 116
Dynamic fuzzy binary decision tree 82
Dynamic fuzzy character 1
Dynamic Fuzzy Concepts 115
Dynamic Fuzzy Data 183
Dynamic fuzzy decision tree 112
Dynamic Fuzzy Decision Tree Learning 69
Dynamic Fuzzy Distribution 183
Dynamic fuzzy graph hierarchical relation-
ship learning 209
Dynamic fuzzy graph hierarchical
relationship learning 249
Dynamic fuzzy hierarchical
relational-learning 209
Dynamic fuzzy hierarchical
relationships 209
Dynamic fuzzy information system 116
Dynamic Fuzzy K-Nearest Neighbour
Algorithm 193
Dynamic fuzzy lattice 72
Dynamic fuzzy learning controller 34
Dynamic fuzzy logic relation learning
algorithm 217
Dynamic Fuzzy Machine Learning 1
Dynamic Fuzzy Machine Learning Model 2
Dynamic Fuzzy Machine Learning Space 2
Dynamic Fuzzy Machine Learning System 2
Dynamic fuzzy matrix 226
Dynamic fuzzy matrix hierarchical relational
learning algorithm 226
Dynamic fuzzy mean 138
Dynamic Fuzzy Multi-Task Learning 180
Dynamic fuzzy partitioning grids 85
Dynamic fuzzy random probability 183
Dynamic fuzzy relational learning 43
Dynamic fuzzy relations 39
Dynamic fuzzy rules 10

- Dynamic fuzzy semi-supervised adaptive learning algorithm 196
- Dynamic fuzzy semi-supervised learning model 180
- Dynamic fuzzy semi-supervised multi-task adaptive learning algorithm 202
- Dynamic Fuzzy Semi-Supervised Multi-Task Learning 181
- Dynamic fuzzy semi-supervised multi-task learning model 182
- Dynamic fuzzy semi-supervised multi-task matching algorithm 182
- Dynamic Fuzzy Task 180
- Dynamic fuzzy tree 235
- Dynamic fuzzy tree hierarchical relation learning 235
- Dynamic fuzzy variance 138
- Error-Based Pruning 101
- Estimated error rate 101
- Evaluation function 279
- Expectation output 10
- Extraction of DF concept rules 137
- F-direct graph 251
- Failure-Adjusted Maximization 42
- Flexible rule 219
- Generative models 170
- Gradual rule 218
- Graph 249
- Graph Based Relational Learning 209
- Graph-based approaches 168
- Hierarchical Relational Learning 209
- ID3 algorithm 72
- Immediate return 277
- Improved locally linear embedding 8
- Increment algorithms 129
- Inductive logic programming 40
- Inductive semi-supervised learning 171
- Inferential 267
- Information entropy 88
- Intention Revision 274
- Isomorphic 251
- Iterative operator 26
- Knowledge base 56
- Knowledge transfer 188
- Laplacian Error Estimation 102
- LDA 126
- Learnability 268
- Learning Space 56
- Learning Subspace 56
- Least-squares error 17
- Linear ordered set 85
- Locally linear embedding 5
- Logic Hidden Markov Model 42
- Logic relation database 217
- Low-density division 170
- L-type DF matrix 227
- Mahalanobis distance metric 192
- Manifold Assumption 169
- Markov Logic Network 42
- Maximum Likelihood Estimation 21
- Metric 192
- Minimal Cost Complexity Pruning 101
- Minimum Error Pruning 101
- Multi-agent learning algorithm based on DFL 282
- Multi-Agent Learning Model 267
- Multi-task learning 162
- Non-direct graph 250
- Nonnegativity 192
- Order model of DF concept learning 117
- Overfit 99
- Parameter estimation 163
- Partial order relation 76
- Pessimistic Error Pruning 101
- Post-pruning 101
- Pre-pruning 100
- Probabilistic Logic Programs 41
- Probabilistic Relational Model 41
- Probably approximately correct 121
- Programming In Statistical Modeling 42
- Q-learning algorithm based on DFL 280
- Q-learning function based on DFL 279
- Reactivity 267
- Reduced Error Pruning 101
- Regression 167

- Regression coefficients 23
- Reinforcement learning 1
- Relational Markov Model 42
- Relational Markov Network 41
- Relationship Learning 209
- Representation theorem for DF variant
 - space 119
- Robust factor 10
- Rough Set 1
- Sample error rate 121
- Self-control 267
- Semi-supervised learning 161
- Semi-Supervised Multi-Task Learning 161
- Semi-supervised multi-task learning
 - model 167
- Semi-Supervised Smoothness
 - Assumption 169
- Separation point level 93
- Simple graph 251
- Single-agent learning algorithm based on
 - DFL 277
- Soft Decision Tree 69
- Stability 12
- Statistical Relational Learning 41
- Structure Activity Relationship 258
- Sub-graph 251
- Sub-graph isomorphic 251
- Sufficient and necessary condition 228
- Symmetry 192
- Topological order 253
- Transductive learning 172
- Transfer learning 176
- Triangle inequality 192
- True error rate 121
- Truncation operation 116
- Type L relationship 226
- Type-one certainty rule 221
- Type-two certainty rule 222
- Uncertain dynamic fuzzy autonomous
 - system 32
- Unlabelled data 167
- VC dimension 123
- Weight coefficient 12

