

Li Fanzhang, Zhang Li, Zhang Zhao

Lie Group Machine Learning

DE GRUYTER

Authors

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

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

Dr. Zhao Zhang
Soochow University
School of Computer Science and Technology
No. 1 Shizi Road
215006 Suzhou, China
cszzhang@suda.edu.cn

ISBN 978-3-11-050068-4
e-ISBN (PDF) 978-3-11-049950-6
e-ISBN (EPUB) 978-3-11-049807-3

Library of Congress Control Number: 2018951019

© 2019 Walter de Gruyter GmbH

www.degruyter.com

Preface

Machine learning, as a branch of artificial intelligence, has been playing an increasingly important role in scientific research in recent years. The main reason for this is that the development of this discipline requires intelligent methods to enrich the content of related disciplines. In the intelligent method, machine learning is the most basic and key core technology. Machine learning algorithms have been widely used in bioinformatics, physics, chemistry, material analysis, large data analysis and so forth. However, we should clearly realize that although machine learning plays an important role in modern scientific research, there is no final consensus on the essence of machine learning. Since 1943, with the proposing of the M-P model and function by American psychologist McCulloch and mathematician Pitts, rule learning based on symbolic calculus and decision tree learning has been produced. Subsequently, scientists have introduced induction, interpretation, analogy, etc. to the concept of cognitive psychology. The 2010 Turing Award winner, Harvard University professor Leslie Valiant, proposed the calculation learning theory, and the 2011 Turing Award winner Judea Pearl proposed the probabilistic causal reasoning algorithm. These different learning methods have their own advantages and disadvantages.

From a historical perspective, machine learning research should use cognitive science as its foundation, mathematics for its methods, theoretical calculation as its standard, analysing rules of data as its goal, and computer technology as its way to construct machine learning theory, technology, methods and applications. In this way, the machine learning methods can be tested, so that it is possible to form a platform and industry with machine learning technology as the main body, and truly realize the purpose of machine learning in leading future technology. Therefore, it is probably one of the most appropriate approaches, to study the theory and algorithm of machine learning by the complementary mechanism of model and analysis. The reasons for this are as follows:

First, machine learning is the process by which machines process input data with algorithms. This allows the machine to correspond with the human. As Neisser pointed out in cognitive psychology, "cognition refers to the whole process of information perception, input change, simplification, processing, storage, recovery and use." This view emphasizes the flow of information in the human brain, beginning with the perception of information until the end.

Second, machine learning is the machine's processing of symbols. This interpretation allows the machine and the human brain to correspond to the operation of symbols. A symbol represents a representation different from itself, such as language, notation, mathematical symbols, etc.

Third, machine learning is the process of problem solving. This interpretation can be understood as a process in which the human brain uses existing knowledge and experience to solve problems.

Fourth, machine learning is the process in which machines optimize data. The purpose of human learning is to recognize the laws of nature and social laws. In the process of learning, optimization is a very important task. The goal of machine learning is to analyse the laws of data. Therefore, optimization is a very important task in machine learning.

The common machine learning features of the four different explanations for machine learning are: First, the machine learning system is a symbolic system. Second, the machine learning system is a multi-layer and multi method system that can fuse data from different sources. Third, the capacity of machine learning systems to handle data capacity is limited. Fourth, the machine learning system has the ability to process both qualitative and quantitative data. Fifth, the machine learning system has the ability to combine learning, reasoning and analysis.

It is in this way that this book introduces Lie group machine learning. This book is committed to answering how to use model continuity theory to solve realistic discrete data, how to model more data with minimal data constructs, how to solve unstructured data by structured methods, how to solve nonlinear data by linear methods, how to solve the two-way mechanism of perception and cognition, the correspondence between individuals and environment, and the dynamic permission relationship by qualitative and quantitative methods.

It is common understanding that Lie group machine learning is the theoretical basis of brain intelligence, brain learning, higher machine learning, and higher artificial intelligence. There are 17 chapters in this book introducing the basic content of Lie group machine learning, Lie group machine learning model, Lie Group Subspace Orbit Generation Learning, Symplectic Group Learning, Quantum Group Learning, Lie Group Fibre Bundle Learning, Lie group covers learning, Lie group deep structure learning, Lie group semi-supervised learning, Lie group kernel learning, Tensor learning, Frame bundle connection learning, Spectral estimation learning, Finsler geometric learning, Homology boundary learning, Category representation learning, and Neuromorphic synergy learning have been studied in each chapter.

This book 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. In addition, this text can be used as the basis for teaching the principles of machine learning.

The overall design, revision and finalization of the book were completed by three professors, Li Fanzhang, Zhang Li and Zhang Zhao. Participating authors include Wang Lei, Huang Shuning, Luo Xiaohui, Wu Xinjian, Li Meixuan, Cui Jingmei, Yin Hongwei, Zou Peng, Xu Xiaoxiang, Yang Mengduo, and others. We are also grateful for the wisdom and work of all the teachers and students involved. 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 the references. Thanks to the National Natural Science Founda-

tion (61033013, 60775045, 61672364, 61672365), Soochow scholar program (14317360, 58320007), and the key subjects of Jiangsu Province “Technology of Computer Science” and “Software Engineering” for their support of this book.

Finally, I hope the book can bring happiness and inspiration to readers! As this work is a first edition, combined with the author's limited experience and knowledge, please contact us with any queries about the content of the book. Email: lfzh@suda.edu.cn, Tel: 13962116494.

Soochow University, July 28, 2017

Li Fanzhang

Contents

Preface — V

1	Lie group machine learning model — 1
1.1	Introduction — 1
1.2	Concepts of Lie group machine learning — 1
1.3	Algebraic model of Lie group machine learning — 10
1.3.1	Lie algebras — 10
1.3.2	One-parameter subgroup — 11
1.3.3	Algebraic model — 12
1.4	Geometric model of Lie group machine learning — 13
1.5	Axiom hypothesis of Lie group machine learning — 15
1.6	Geometric learning algorithm for dynkin graphs in Lie group machine learning — 26
1.6.1	Overview of dynkin graphs in Lie group machine learning — 27
1.6.2	Classification algorithm of dynkin diagrams in Lie group machine learning — 29
1.7	Linear classifier design of Lie group machine learning — 33
1.7.1	Linear classifier design of Lie group machine learning — 33
1.7.2	Lie group machine learning SO(3) classifier — 34
1.7.3	Classification of text based on Lie group machine learning — 35
1.8	Chapter summary — 37
2	Lie group subspace orbit generation learning — 39
2.1	Basic concepts of partial order and lattice in LML — 39
2.1.1	Basic Concepts — 39
2.1.2	Partially ordered set in LML — 43
2.1.3	Möbius function on local finite partial order in LML — 46
2.1.4	Gaussian coefficients and gaussian polynomials in LML — 49
2.1.5	Lattices in LML — 54
2.2	LML subspace orbit generating lattice learning algorithm — 56
2.2.1	LML subspace orbit generation lattice — 56
2.2.2	Orbit generation learning algorithm for LML subspace — 59
2.3	LML learning subspace orbit generation lattice learning algorithm under the action of the general linear group $GL_n(\mathbb{F}_n)$ — 71
2.3.1	Problem description — 71
2.3.2	LML learning subspace orbit generation lattice under the action of the general linear group $GL_n(\mathbb{F}_n)$ — 72
2.3.3	Learning algorithms and examples — 76
2.4	Summary — 79

3	Symplectic group learning — 81
3.1	Problem presentation — 81
3.2	Design of the symplectic group classifier in Lie group machine learning — 88
3.2.1	Symplectic group classifier description — 88
3.2.2	Design method of symplectic group classifier — 94
3.2.3	Design of symplectic group classifier in face recognition — 100
3.2.4	Symplectic group classifier design for classification process of data set — 103
3.3	Symplectic group classifier algorithm in Lie group machine learning — 105
3.3.1	Symplectic group classifier algorithm — 105
3.3.2	Verification of symplectic group classifier algorithm in face recognition — 107
3.3.3	Verification of symplectic group classifier algorithm in data set classification — 109
3.4	Application example — 111
3.4.1	Processing of images under symplectic matrix — 112
3.4.2	Instance validation — 116
3.5	Summary — 119
4	Quantum group learning — 123
4.1	Problem presentation — 123
4.2	Construction method of quantum group classifier in Lie group machine learning — 124
4.2.1	Problem description — 124
4.2.2	Construction of quantum group classifier in Lie group machine learning — 125
4.2.3	DNA Sequence classification based on quantum group classifier in Lie group machine learning — 128
4.3	Application of quantum group learning algorithm in molecular docking — 132
4.3.1	Introduction to molecular docking algorithm — 132
4.3.2	Molecular docking design model based on quantum group — 135
4.3.3	Molecular matching algorithm based on quantum group generators — 138
4.3.4	Molecular Docking Simulation based on Quantum Group — 139
4.3.5	Experimental results and analysis — 140
4.4	Summary — 144
5	Lie group fibre bundle learning — 147
5.1	Problem presentation — 147

5.2	Fibre bundle model — 148
5.2.1	Expression of fibre bundles in manifold learning — 148
5.2.2	Tangent bundle model for manifold learning — 150
5.2.3	Main fibre bundle model — 152
5.3	Fibre bundle learning algorithm — 153
5.3.1	Vector reduction algorithm based on local principal direction of tangent — 154
5.3.2	Main link curve construction algorithm based on tangent contact — 156
5.4	Summary — 162

6 Lie group covering learning — 165

6.1	Theory of Lie group machine learning covering algorithm — 165
6.1.1	Linear representation of a group — 165
6.1.2	Basic properties of the Lie group — 167
6.2	Simply connected covering algorithm of the Lie group — 173
6.2.1	Research status of algorithm based on covering idea — 173
6.2.2	Simply connected covering of Lie group machine learning — 175
6.2.3	Algorithm design — 178
6.2.4	Example application analysis — 179
6.3	Multiply connected covering algorithm of Lie group machine learning — 181
6.3.1	LML multiply connected covering model — 182
6.3.2	Multiply connected covering algorithm design — 184
6.3.3	Applications — 187
6.4	Application of the covering algorithm in molecular docking — 189
6.4.1	Introduction to the molecular docking algorithm — 189
6.4.2	Mathematical model and evaluation function of molecular docking — 191
6.4.3	Covering strategy and implementation of molecular docking — 193
6.4.4	Experimental results and analysis — 199
6.5	Summary — 204

7 Lie group deep structure learning — 207

7.1	Introduction — 207
7.2	Lie group deep structure learning — 209
7.2.1	Deep structure learning — 209
7.2.2	Construct deep structure model — 210
7.2.3	Deep structure learning algorithm — 212
7.2.4	Lie group deep structure learning algorithm — 214
7.2.5	Experiment analysis — 218
7.3	Lie group layered learning algorithm — 220
7.3.1	Singular value feature extraction — 221
7.3.2	Layered learning algorithm — 223
7.3.3	Experiment and analysis — 225

7.4	Lie group deep structure heuristic learning — 227
7.4.1	Heuristic learning algorithm — 227
7.4.2	A* algorithm — 228
7.4.3	Lie group deep structure heuristic learning algorithm — 229
7.4.4	Experiment and analysis — 230
7.5	Summary — 231
8	Lie group semi-supervised learning — 235
8.1	Introduction — 235
8.1.1	Research status of semi-supervised learning — 235
8.1.2	Questions raised — 243
8.2	Semi-supervised learning model based on the Lie group — 244
8.2.1	Representation of the Lie group in semi-supervised study — 244
8.2.2	Semi-supervised learning model based on Lie group algebra structure — 245
8.2.3	Semi-supervised learning model based on the geological structure of the Lie group — 247
8.3	Semi-supervised learning algorithm based on a linear Lie group — 249
8.3.1	The general linear group — 250
8.3.2	Semi-supervised learning algorithm based on the linear Lie group — 253
8.3.3	Experiment — 256
8.4	Semi-supervised learning algorithm based on the parameter Lie group — 259
8.4.1	Sample data representation — 259
8.4.2	Semi-supervised learning algorithm based on the parameter Lie group — 260
8.4.3	Experiment — 264
8.5	Summary — 269
9	Lie group kernel learning — 275
9.1	Matrix group learning algorithm — 275
9.1.1	Related basic concepts — 275
9.1.2	Matrix group — 275
9.1.3	The learning algorithm of the matrix group — 280
9.1.4	Cases analysis — 281
9.2	Gaussian distribution on the Lie group — 284
9.2.1	Gaussian distribution of R^+ — 285
9.2.2	Gaussian distribution of $SO(2)$ — 285
9.2.3	Gaussian distribution of $SO(3)$ — 287
9.3	Calculation of the Lie group inner mean value — 288
9.4	Lie-mean learning algorithm — 291

9.4.1	FLDA algorithm — 292
9.4.2	Fisher mapping in Lie group space — 292
9.4.3	Lie–Fisher discriminant analysis — 294
9.5	Nuclear learning algorithm of the Lie group — 298
9.5.1	Principle of the SVM algorithm — 299
9.5.2	The principle of KFDA — 300
9.5.3	Kernel — 302
9.5.4	Kernel of the Lie group — 303
9.5.5	KLieDA algorithm based on the Lie group kernel function — 305
9.6	Case application — 306
9.6.1	Experimental analysis of the Lie–Fisher algorithm — 306
9.6.2	Artificial data set — 307
9.6.3	Handwriting recognition — 309
9.6.4	Covariance Lie group characteristic of the Lie–Fisher handwriting classification — 311
9.7	Summary — 314
10	Tensor learning — 319
10.1	Data reduction based on tensor methods — 319
10.1.1	GLRAM — 319
10.1.2	HOOI — 320
10.1.3	2DPCA — 320
10.1.4	CubeSVD — 321
10.1.5	TSA — 322
10.1.6	Related problem — 322
10.2	Data reduction model based on tensor fields — 323
10.2.1	Tensor field on a manifold — 323
10.2.2	Reduction model based on the tensor field — 325
10.2.3	Design of data reduction algorithm based on the tensor field — 327
10.2.4	Experiment — 330
10.3	The learning model and algorithm based on the tensor field — 332
10.3.1	Learning model based on the tensor field — 332
10.3.2	Tensor bundle learning algorithm — 335
10.3.3	Classification model based on the tensor field — 337
10.3.4	Classification algorithm based on the tensor field — 342
10.4	Summary — 344
11	Frame bundle connection learning — 347
11.1	Longitudinal space learning model based on frame bundle — 347
11.2	Longitudinal space connection learning model based on frame bundle — 350

11.3	Horizontal space connection learning model based on frame bundle — 352
11.4	Related applications — 353
11.5	Summary — 355
12	Spectral estimation learning — 357
12.1	Concept and definition of spectral estimation — 357
12.1.1	Research background of the spectral estimation method — 357
12.1.2	Concept and definition of spectral estimation — 357
12.1.3	Research progress in learning methods of spectral estimation — 358
12.2	Relevant theoretical basis — 359
12.2.1	How to construct a similarity matrix — 360
12.2.2	How to choose the appropriate laplacian matrix — 362
12.2.3	Selecting the appropriate feature vector — 362
12.2.4	Determining the number of clusters — 364
12.3	Synchronous spectrum estimation learning algorithm — 365
12.3.1	Graph optimisation criterion for locally preserving mappings — 365
12.3.2	Asynchronous spectrum estimation learning model — 366
12.3.3	Synchronous spectrum estimation learning algorithm — 368
12.3.4	Case verification — 369
12.4	The comparison principle of image feature manifolds — 371
12.4.1	Topological spherical theorem — 372
12.4.2	Polarisation theorem of image feature manifolds — 374
12.4.3	Manifold dimensionality reduction algorithm — 375
12.5	Spectral estimation learning algorithm for topological invariance of image feature manifolds — 376
12.5.1	Spectral estimation learning algorithm for the topological invariance of image feature manifolds — 377
12.5.2	Algorithm analysis — 377
12.5.3	Example analysis — 378
12.6	Clustering algorithm based on the topological invariance spectral estimation of image feature manifolds — 380
12.7	Summary — 380
13	Finsler geometric learning — 383
13.1	Basic concept — 383
13.1.1	Riemann manifold — 383
13.1.2	Finsler geometry — 384
13.2	KNN algorithm based on the Finsler metric — 384
13.2.1	K nearest neighbour algorithm — 384
13.2.2	KNN algorithm based on the Finsler metric — 386
13.2.3	Experimental results and analysis — 388

13.3	Geometric learning algorithm based on the Finsler metric — 390
13.3.1	Supervised manifold learning algorithm — 391
13.3.2	Finsler geometric learning algorithm — 393
13.4	Summary — 398
14	Homology boundary learning — 401
14.1	Boundary learning algorithm — 401
14.1.1	Tangent vector quantisation (TVQ) algorithm — 401
14.1.2	Regularised large marginal classifier (RLMC) — 402
14.1.3	Boundary detection algorithm based on the boundary Markov random field and Boltzmann machine — 402
14.1.4	Fuzzy edge detection algorithm based on the qualification function — 403
14.2	Edge division method based on homological algebra — 404
14.2.1	Basic concepts of homological algebra — 404
14.2.2	Mapping of homotopy and the space of homotopy — 405
14.2.3	Cohomology edge algorithm — 406
14.2.4	Cell homology edge algorithm — 408
14.3	Design and analysis of homology edge learning algorithm — 412
14.4	Summary — 413
15	Category representation learning — 415
15.1	Introduction — 415
15.1.1	Research background — 415
15.1.2	The relation between category theory and computer science — 418
15.1.3	Basic concepts of category theory — 420
15.1.4	Proposed problem — 424
15.2	Category representation of learning expressions — 425
15.2.1	Category representation of machine learning systems — 425
15.2.2	Category representation of learning expressions — 429
15.2.3	Category Representation of the Learning Expression Functor — 431
15.2.4	Natural transformation — 432
15.3	Mapping mechanism for learning expressions — 433
15.3.1	Abstract concept of an expression — 433
15.3.2	Mapping mechanism between expressions — 436
15.4	Classifier design for learning expression mapping mechanism — 441
15.4.1	Classifier algorithm — 442
15.4.2	Classifier based on the learning expression mapping mechanism — 445
15.4.3	Example analysis and results — 448
15.5	Example analysis — 451
15.5.1	Instance analysis of learning expression mapping — 451

15.5.2	Case analysis of image recognition — 459
15.6	Summary — 462
16	Neuromorphic synergy learning — 465
16.1	Introduction — 465
16.2	Core scientific problems — 469
16.3	Lie group cognitive theory framework — 471
16.4	Neuromorphic synergy learning theoretical framework — 475
16.4.1	Symbol grounding learning — 477
16.4.2	Bidirectional synergy learning — 478
16.4.3	Affordance learning — 479
16.4.4	Multi-scale synergy learning — 480
16.5	Design of a neuromorphic synergy learning verification platform — 482
16.6	Summary — 483
17	Appendix — 485
17.1	Topological group — 485
17.2	Concept of differential geometry — 488
17.3	Manifold learning algorithm — 490
17.3.1	Local linear embedding (LLE) — 490
17.3.2	Isometric mapping (Isomap) — 491
17.3.3	Horizontal linear embedding (HLLE) — 492
17.3.4	Laplacian eigenmap — 493
17.3.5	Local tangency space arrangement (LTSA) — 494
17.4	Basic concept and nature of symplectic group — 494
17.5	Basic concepts of quantum groups — 499
17.5.1	Image description of the quantum group — 499
17.5.2	Definition and decision algorithm of quantum group — 500
17.5.3	Quantisation — 503
17.5.4	Representation of quantum groups — 505
17.6	Fibre bundle — 506

Index — 515

1 Lie group machine learning model

This chapter is divided into eight sections. The first section is the introduction. The second section presents the concepts of Lie group machine learning. The third and fourth sections describe the algebraic and geometric models of Lie group machine learning, respectively. The fifth section discusses the axiom hypothesis of Lie group machine learning. The sixth and seventh sections deal with the geometric learning algorithm and linear classifier design of Dynkin graphs, respectively. Finally, the eighth section summarises the chapter.

1.1 Introduction

In this chapter, the theory of Lie group is introduced into the field of machine learning, and the related concepts and model of Lie group machine learning are presented. Understanding this chapter requires relevant knowledge of topological and differential geometries, which is included in Appendix 17.1 and 17.2.

1.2 Concepts of Lie group machine learning

Before establishing the Lie group machine learning model, we first introduce some basic concepts of Lie group [1].

Definition 1.1. Semigroup: A non-empty set S satisfies the operation of the combined law. In general, this operation of semigroups is called multiplication and denoted by ‘ \cdot ’. Therefore, semigroups are denoted by (S, \cdot) .

The generalised combined law of semigroup S means that the operation results of the arbitrary sequence a_1, a_2, \dots, a_n in S and any reasonable brackets are the same. In other words, the result of the operation depends only on order of the elements instead of the brackets.

According to the definition of semigroups, the operation does not necessarily satisfy the law of commutation for the elements a and b (i.e., $ab \neq ba$). Semigroups satisfying the commutative law are called exchange semigroups. The product of n elements a_1, a_2, \dots, a_n in an exchange semigroup is independent of the factor order.

Definition 1.2. Unitary group: Let (M, \cdot) be a semigroup. If $e \in M$ satisfies $ea = ae = a$ for any $a \in M$, then M is a unitary group.

Definition 1.3. Consider a non-empty set G that satisfies the operation of the law of association(generally termed multiplication). If an element e satisfies $ea = a$ for all $a \in G$, and if for any $a \in G$, $a' \in G$ satisfies $a'a = e$, then G is called a group.

Algorithm 1.1: Judge a group**Initialisation:** G is a non-empty set//**Input:** G //**Output:** {0, 1}//

```
procedure GROUP( $G$ )
for  $\forall a, b, c \in G$  do
if not  $(a(bc)) == ((ab)c)$ 
    then return 0;
endif
repeat
endfor
for  $\forall a \in G$  do
if not  $(\exists e \text{ and } ea = a)$ 
    then return 0;
endif
if not  $(\exists a' \text{ and } aa' = e)$ 
    then return 0;
endif
repeat
endfor
return 1;
end GROUP
```

Definition 1.4. Let M be a non-empty set. If its subset τ satisfies

- (1) $M, \emptyset \in \tau$;
- (2) if $U_1, U_2 \in \tau$, then $U_1 \cap U_2 \in \tau$;
- (3) if $U_\alpha \in \tau$, $\alpha \in \Gamma$ (index set), $\bigcup_{\alpha \in \Gamma} U_\alpha \in \tau$, then τ is termed as a topology on M ,

(M, τ) is a topological space, the element U in τ is termed the open set of (M, τ) , and the complete set $U^c = M - U$ of the open set U is termed the closed set of (M, τ) .

Suppose that $A \subset M$, $x \in A$. If for any open neighbourhood U of x (containing the open set of x), there exists $a \in A \cup (U - \{x\})$, then x is termed the accumulation point(limit point) of A . Further, A' , called guide, are all the accumulation points of A . The points in $A - A'$ are termed isolated points of A . If $\bar{A} = A \cup A'$ is the closure of A , then obviously, A is the closed set $\Leftrightarrow A' \subset A \Leftrightarrow \bar{A} = A$.

Definition 1.5. For a set $O_\varepsilon(x)$ of all points, $x \in X$ satisfies $p(x, Y) < \varepsilon$ termed the sphere neighbourhood of the set Y whose radius is ε . If it is a certain sphere neighbourhood contained in Y , i.e., $O_\varepsilon(x) \subset Y$ for a point x of a set Y , x is an inner point of Y . All points in space form a subset of inner points becoming an open set.

Definition 1.6. Suppose that (X, τ) is a topological space, $x \in X$. If U is a subset of X and satisfies the condition that an open set $V \in \tau$ exists, such that $x \in V \subset U$, then U is a neighbourhood of point x . All neighbourhoods of point x constitute a subset cluster of point x , called a neighbourhood system. Obviously, if U contains an open set of point x , U must be a neighbourhood of x ; then, we call U an open neighbourhood of x .

Definition 1.7. Suppose that (M, τ) is a topological space. If for any $x, y \in M$, there exist an open neighbourhood U of x and an open neighbourhood V of y satisfying $U \cap V \neq \emptyset$, then (M, τ) is a T_2 (Hausdorff) space. If the topological space (M, τ) has a countable topological base and Γ is a finite or ordered set (one-to-one correspondence with natural numbers N), (M, τ) is an A_2 space.

Algorithm 1.2: Judge topological space of set

Initialisation: X is a set, τ is a subset cluster of X //

Input: X, τ //

Output: {0, 1}//

```
procedure TOPO ( $X, \tau$ )
if not ( $\emptyset \in \tau$  and  $X \in \tau$ )
    then return 0;
endif
for  $\forall x_1, x_2, \dots, x_m \in \tau$  do
    if not ( $x_1 \cup x_2 \cup \dots \cup x_m \in \tau$ )
        then return 0;
    endif
repeat
endfor
for  $\forall x, y \in \tau$  do
    if not ( $x \cap y \in \tau$ )
then return 0;
    endif
repeat
endfor
return 1; // $\tau$  is a topology of set  $X$ //
end TOPO
```

Algorithm 1.3: Judge inner point

Initialisation: X is a set, x is an element of set X , ε is the radius //

Input: X, x, ε //

Output: {0, 1}//

```
procedure INNERPOINT ( $X, x, \varepsilon$ )
maxdistance=0;
 $x_p = kmean(x, 1)$ ; //using k-means Algorithm to obtain the centre of set  $X$ //
/*seeking maximal margin of set  $X$ */
for  $\forall x \in X$  do
    distance =  $\|x - x_p\|_2$  // $\|x - x_p\|_2$  represents the Euclidean distance between two points//
if maxdistance<distance
    then maxdistance=distance
endif
endfor
/*Judge if  $x$  contained in  $X$ */
if  $\|x - x_p\|_2 \leq maxdistance$ 
    then return 1;
```

```

else if  $\|x - x_p\|_2$ -maxdistance< $\varepsilon$ 
    then return 1;
endif
endif
return 0;
end INNERPOINT

```

Algorithm 1.4: Judge open set

Initialisation: (X, ε) is a topological space, $A \subset M$ //

Input: A //

Output: {0, 1}//

```

procedure OPENSET( $A$ )
if TOPO( $X, \tau$ ) //  $(X, \varepsilon)$  is a topological space// 
    for  $\forall x \in A$  do
        if INNERPOINT( $A, x, \varepsilon$ )  $\leq 0$  //point  $x$  is not an inner point, then return 0 //
            return 0;
        endif
    repeat
    endfor
return 1;
endif
end OPENSET

```

Algorithm 1.5: Judge open neighbourhood

Initialisation: (X, τ) is a topological space, $x \in X$ //

Input: A //

Output: {0, 1}//

```

procedure OPENNEIGHBOR( $x, U$ )
if TOPO( $X, \tau$ ) and  $U \subset X$ 
    if not(OPENSET( $U$ ) and  $x \in U$ )
        then return 0;
        else return 1;
    endif
endif
end OPENNEIGHBOR

```

Algorithm 1.6: Judge T_2 (Hausdorff) space

Initialisation: (X, τ) is a topological space, U, V are the open neighbourhoods of x, y , respectively //

Input: X //

Output: {0, 1}//

```

procedure HAUSDORFF( $X$ )
if TOPO( $X, \tau$ )
    for  $\forall x, y \in X$  do
        if not ( $x \neq y$  and  $U \cap V = \emptyset$ )
            then return 0;

```

```

endif
repeat
endfor
return 1;
endif
end HAUSDORFF

```

Definition 1.8. Continuous mapping: Suppose that (M_1, τ_1) and (M_2, τ_2) are topological spaces, $f: M_1 \rightarrow M_2$ is a mapping, $x_0 \in M_1$, and $y_0 = f(x_0) \in M_2$. If for any open neighbourhood V of $y_0 = f(x_0)$, there is an open neighbourhood U of x_0 such that $f(U) = \{f(x) | x \in U\} \subset V$, then f is continuous in point x_0 . If f is continuous in every point of M_1 , then f is a continuous mapping.

Note that according to the definition of the continuous property of topology, f is continuous \Leftrightarrow for any open set V of (M_2, τ_2) , $f^{-1}(V) = \{x \in M_1 | f(x) \in V\}$ is an open set of (M_1, τ_1) \Leftrightarrow for any closed set F of (M_2, τ_2) , and $f^{-1}(F) = \{x \in M_1 | f(x) \in F\}$ is a closed set of (M_1, τ_1) .

Definition 1.9. If $f: M_1 \rightarrow M_2$ is a one-to-one correspondence, and f and f^{-1} are continuous mappings, then f is a homeomorphism (topology) mapping.

Definition 1.10. Suppose that M is a T_2 (Hausdorff) space. If for any $p \in M$, there exists an open neighbourhood U of p in M and homeomorphism $\varphi: u \rightarrow \varphi(u)$, where $\varphi(u) \subset \mathbf{R}^n$ is an open set (partial Euclidean), then M is a n -dimensional topological manifold or C^o manifold.

(U, φ) is a partial coordinate system (coordinate card, picture), U is a partial coordinate neighbourhood, φ is a partial coordinate mapping, and $x^i(p) = (\varphi(p))^i$ ($1 \leq i \leq n$), is a partial coordinate of $p \in U$, simplified as $\{x^i\}$, which is sometimes called a partial coordinate system. A topological manifold is an atlas consisting of pictures from all D of the partial coordinate system. If $p \in U$, then (U, φ) is a partial coordinate system of p .

Algorithm 1.7: Judge manifold

```

Input:  $M$  //
Output: {0, 1} //
procedure MANIFOLD( $M$ )
if HAUSDORFF( $M$ )  $\leq 0$ 
    then return 0;
endif
for  $\forall x \in M$  do
    if  $\exists U$  and OPENNEIGHBOR( $x, U$ )
        if OPENSET( $\phi(U)$ ) // Homeomorphism  $\phi(U)$  of  $U$  is an open set //
            then return 1;
        endif
    endif

```

```

repeat
endfor
end MANIFOLD

```

Definition 1.11. Suppose that $\varphi: U \rightarrow \varphi(U) \subset \mathbf{R}^n$ is a homeomorphism, where $\varphi(U)$ is an open set of \mathbf{R}^n ; then, (U, φ) is a coordinate card of manifold M .

Definition 1.12. Suppose that $(U_1, \varphi_1), (U_2, \varphi_2)$ are two coordinate cards of manifold M with n dimensions. If $U_1 \cap U_2 \neq \emptyset$ and $\varphi_2 \circ \varphi_1^{-1}: \varphi_1(U_1 \cap U_2) \rightarrow \varphi_2(U_1 \cap U_2)$ and its inverse mapping are differentiable r times, then $(U_1, \varphi_1), (U_2, \varphi_2)$ are related to C^r .

Definition 1.13. Suppose that A is a subset of X and $U = \{U_\alpha\}_{\alpha \in A}$ is a subset cluster of X . If $A \subset \cup_\alpha U_\alpha$, then U is a cover of A . If any U_α is an open set of topological space X , then U is an open cover of A .

Definition 1.14. Suppose that M is a manifold with n dimensions. Further, suppose that $\Lambda = \{(U_\alpha, \varphi_\alpha): \alpha \in I\}$ is a subset of a coordinate card in M satisfying the following conditions:

- (1) $\{U_\alpha: \alpha \in I\}$ consists of an open cover of M ;
- (2) Any two coordinate cards belonging to Λ are related to C^r ;
- (3) Λ is maximum.

Then, Λ is a C^r differential structure in M .

Definition 1.15. Suppose that M is a manifold with n dimensions. If we specify a C^r differential structure Λ in M , then (M, Λ) is a C^r differential manifold with n dimensions. Figure 1.1 shows a differential manifold.

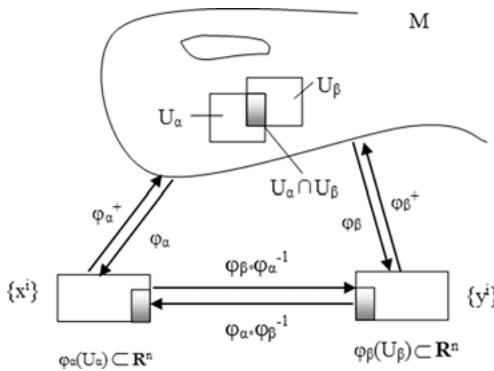


Fig. 1.1: Differential Manifold.

Algorithm 1.8: Judge open cover**Initialisation:** A is a subset of X , $U = \{U_\alpha\}_{\alpha \in A}$ is a subset cluster of X //**Input:** A, U //**Output:** {0, 1}//

```
procedure OPENCOVER( $A, U$ )
if not ( $A \subset \bigcup_{\alpha} U_\alpha$ )
    then return 0;
endif
for  $\forall U_\alpha$  do
if OPENSET( $U_\alpha$ )  $\leq 0$  //  $U_\alpha$  is not an open set of topological space  $X$ , then return 0//
    then return 0;
endif
repeat
endfor
return 1;
end OPENCOVER
```

Algorithm 1.9: Judge differential manifold**Initialisation:** M is a manifold with n dimensions, $V = \{(U_\alpha, \varphi_\alpha) : \alpha \in I\}$ is a subset of a coordinate card in M //**Input:** M //**Output:** {0, 1}//

```
procedure DIFFMANI( $M$ )
if DIFFMANI( $M$ )  $\leq 0$ 
    then return 0;
endif
for  $\forall \alpha \in I$  do
if OPENCVER ( $M, U_\alpha$ )  $\leq 0$  //  $U_\alpha$  is not an open cover of  $M$ , then return 0//
    then return 0;
endif
repeat
endfor
for  $\forall V_a, V_b \in V$  do
if ASSOCIATE ( $V_a, V_b$ )  $\leq 0$  //  $V_a, V_b$  are not related to  $C^r$ , then return 0//
    then return 0;
endif
if MAXIMUM( $V$ )  $\leq 0$  //  $V$  is not maximum, then return 0//
    then return 0;
endif
repeat
endfor
return 1;
end DIFFMANI
```

Now, we can give the specific definition of a Lie group.

Definition 1.16. Suppose that G is a non-empty set satisfying the following conditions:

- G is a group;
- G is a differential manifold;
- The operations of the group are differentiable, i.e., the mapping $(g_1, g_2) \mapsto g_1g_2^{-1}$ from $G \times G$ to G is a differentiable mapping.

Then, G is a Lie group.

According to the definition of a Lie group, we know that a Lie group is not only a group but also a differential manifold. In general, a manifold is the extension of the concept of a point, line, surface, or various high-dimensional continuous spaces. All observed data in the field of machine learning can be connected with a point, line, surface or other structures. Thus, a Lie group can be used to represent data in the field of machine learning. Through the geometric and algebraic representation of a Lie group for data, we can provide a qualitative description from the geometric point of view and a quantitative description from the algebraic point of view. Through this learning approach for many-to-many expressions, it is possible to find a natural way to solve many problems.

Algorithm 1.10: Judge Lie group

Initialisation: G is a non-empty set //

Input: G //

Output: {0, 1} //

```
procedure LIEGROUP( $G$ )
if GROUP( $G$ ) ≤ 0 //  $G$  is not a group, then return 0 //
    then return 0;
endif
if DIFFMANI( $G$ ) ≤ 0 //  $G$  is not a differential manifold, then return 0 //
    then return 0;
endif
if DIFFERENTIAL( $G$ ) //operation of  $G$  is differentiable //
    then return 1;
    else return 0;
endif
end LIEGROUP
```

Based on the above preparation as well as the extensive experience of physicists and chemists who have used Lie groups, the extension of Lie groups to machine learning can be regarded as the discovery of the laws of data implication from the perspective of machine learning applied to data analysis. Experts in physics, chemistry, and other areas use the Lie group as a tool to solve problems, which is also implicit in the field of machine learning, where experts can adopt such a vision to find deep data solutions. For further details, readers may refer to the relevant physical and chemical literature. We now give the definition of Lie group machine learning.

Definition 1.17. Lie group machine learning: In general, G is used to represent the input space and M is used to represent the output field. Let $G \subseteq R^D$, $M \subseteq R^d$, $D > d$. Then, the definition of Lie group G for the mapping φ on M is as follows:

$$\begin{aligned}\varphi: G \times M &\rightarrow M \\ g, x &\rightarrow \varphi(g, x).\end{aligned}$$

It should hold that $\varphi(g_1, \varphi(g_2, x)) = \varphi(g_1g_2, x)$, $\varphi(e, x) = x$. Then, for any $g \in G$, M is a diffeomorphism exchange

$$\begin{aligned}\varphi_g: M &\rightarrow M \\ x &\rightarrow \varphi(g, x).\end{aligned}$$

Mapping set $\{\varphi_g, g \in G\}$ satisfies $\varphi_{g_1} \cdot \varphi_{g_2} = \varphi_{g_1g_2}$, $\varphi_e = id_M$. The diagram of Lie group machine learning is shown in Figure 1.2.

Similarly, we can define the right effect of group G on M

$$\begin{aligned}\psi: M \times G &\rightarrow M \\ x, g &\rightarrow \psi(x, g)\end{aligned}$$

satisfying $\psi(\psi(x, g_1), g_2) = \psi(x, g_1g_2)$, $\psi(x, e) = x$. The right effect of Lie group machine learning is shown in Figure 1.3.

It is easy to see the diffeomorphism exchange of the right effect from G to M

$$\begin{aligned}\psi_g: M &\rightarrow M \\ x &\rightarrow \psi(x, g).\end{aligned}$$

Because a Lie Group is an analytic manifold with a group structure, and group operations are analytic, the Lie group machine learning norm can be used to analyse the dimension, compactness, connectivity, nilpotency, subgroup, cosets, quotient groups, quantum groups, and so on. These basic structures serve as the foundation for the design of the Lie group machine learning algorithm [2–4, 8].

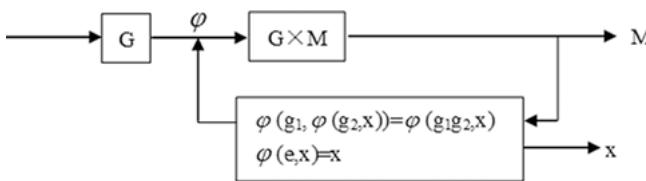


Fig. 1.2: Lie Group Machine Learning left effect model.

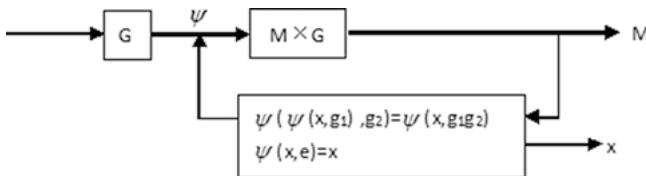


Fig. 1.3: Lie Group Machine Learning right effect model.

1.3 Algebraic model of Lie group machine learning [2, 3, 10]

According to the concept of Lie group machine learning, four different mathematical structures, namely Lie algebras, one-parameter subgroup, left invariant vector field, and left invariant manifold, are defined as a natural correspondence, i.e., Lie algebras \cong {left invariant vector field} \cong {left invariant flow} \cong {one-parameter subgroup}. These mathematical structures serve as the foundation for the study of the Lie algebra root calculation model, decomposition model, and classification model of Lie group machine learning.

1.3.1 Lie algebras

For convenience, suppose that in the Lie group machine learning system, there is an input space G . Suppose that the linear space g on G has a binary operation $(x, y) \rightarrow [x, y]$ (usually called transposition operation or product), and satisfies the following conditions:

(1) The binary operation is bilinear, i.e.,

$$[\lambda x + \mu y, z] = \lambda [x, z] + \mu [y, z], \forall \lambda, \mu \in G, \forall x, y, z \in g;$$

(2) $[x, x] = 0, \forall x \in g$;

(3) A Jacobi identity is established, i.e.,

$$[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0 \forall x, y, z \in g. \text{ Then, } g \text{ is a Lie algebra on } G.$$

From (1), (2), we obtain $[x, y] = -[y, x], \forall x, y \in g$.

Algorithm 1.11: Judge Lie algebras

Initialisation: g is a linear space on G , $[x, y]$ is a blanket product operation on it //

Input: g //

Output: {0, 1} //

```
procedure LIEALGEBRA(g)
for  $\forall x, y, z \in g$  do
    if  $[\lambda x + \mu y, z] == \lambda [x, z] + \mu [y, z]$ 
    then return 0;
    endif
    if not ( $[x, x] == 0$ )
    then return 0;
    endif
    if  $[x, [y, z]] + [y, [z, x]] + [z, [x, y]] == 0$ 
    then return 0;
    endif
repeat
endfor
return 1; //If the above three conditions are satisfied, then  $g$  is a Lie algebra of  $G$  //
end LIEALGEBR
```

If $D^1(G)$ represents a differentiable vector field set and $D_1(G)$ represents a differential form set, and if $\forall g \in G$, $dL_g(X) = X$, then vector field $X \in D^1(G)$ is termed a left invariant vector field.

Suppose that $X, Y \in D^1(G)$ are left invariant vector fields. We define the transposition operation

$$[X, Y] = X \circ Y - Y \circ X .$$

Then, $[X, Y]$ is also left invariant (\circ represents the procedure of mapping).

Lie algebras can not only be regarded as the collection of left invariant vector field isomorphisms but also be understood as the tangent space $T_e(G)$ of unit element e , i.e.,

- (1) For input data G comprising a Lie group, all the left invariant vector fields of Lie group G constitute Lie algebras for all definitions of the transposition operation, called Lie algebras of Lie group G .
- (2) Lie algebras g of Lie group G are mappings with tangent space $T_e(G)$ of G at e

$$X \rightarrow X|_e = Xe, X \in g .$$

The following is a linear space of an isomorphism, i.e., $\dim g = \dim G$. As a result, we can define the transposition operation in $T_e(G)$ as $[X_e, Y_e] = [X, Y]_e$, $\forall X_e, Y_e \in T_e(G)$, where X, Y are left invariant vector fields. The values at e are X_e and Y_e , and $T_e(G)$ is isomorphic with the Lie algebras of g .

1.3.2 One-parameter subgroup

In Lie group machine learning, the one-parameter subgroup of G is the micro-homomorphism θ of the real addition group R to G : $R \rightarrow G$. If t represents the parameters of the elements in R , then $\{\theta(t); t \in R\}$ is the condition that G satisfies

$$\theta(t_1) \cdot \theta(t_2) = \theta(t_1 + t_2)$$

of a different parameter curve.

$\dot{\theta}(t)$ denotes the tangent vector of the curve at $\dot{\theta}(t)$. If g is a Lie algebra of Lie group G , then g has one-to-one correspondence with the one-parameter subgroup of G : $X \rightarrow \dot{\theta}(t)$, such that $\theta(0) = e$, $\dot{\theta}(t) = X_{\theta(t)}$. One of the most important mappings is the $g \rightarrow G$: index mapping. If any element X in the Lie algebra of Lie group G , $\dot{\theta}(t)$ corresponds to a one-parameter subgroup, then the mapping $\exp: X \rightarrow \theta(1)$ is an index mapping.

Through the definition of Lie algebras, we know that the discussion of one-parameter subgroup can achieve linearization of the Lie group structure, which is a vector space g with a blanket product, i.e., Lie algebras of Lie group G . The relationship between Lie group and Lie algebras provides a feasible solution for the conversion of nonlinear data into linear data.

1.3.3 Algebraic model

In Lie group machine learning, there is an input data set G . We suppose that $\theta: \mathbb{R} \rightarrow G$ is a given one-parameter subgroup. Then, we can use a right transformation in G to obtain the transform mapping $R \times G \rightarrow G: (t, g) \rightarrow g \cdot \theta(t)$, which is a differentiable transformation group (single parameter) on manifold G , usually called flow transformation. Moreover, the combined law of a group is that any right transformation $r_a: G \rightarrow G$, $r_a(g) = ga$, $g \in G$ ($a \in G$), and any left transformation $l_a: G \rightarrow G$, $l_a(g) = ag$, $g \in G$ ($a \in G$), are exchangeable such that the roles of the above differentiable transformation group and any left transformation are exchangeable, i.e., $a(g \cdot \theta(t)) = (a \cdot g) \theta(t)$. Therefore, it is also called a left invariant single parameter differentiable transformation group or left invariant flow. Thus, we can obtain (I): {one-parameter subgroup} $\xleftrightarrow{(I)} \{\text{left invariant flow}\}$.

If X is a left invariant vector field, then X is uniquely determined by its value at unit point X_e . By contrast, for any given vector X_0 in g , there must be a left invariant vector field X such that $X_e = X_0$. Thus, we can get (II): $g \xleftrightarrow{(II)} \{\text{left invariant vector field}\}$.

Since {one-parameter subgroup} $\rightarrow g$ and {left invariant flow} $\rightarrow \{\text{left invariant vector field}\}$ are vectors for determining speed, we mark them by differentials.

We use a left invariant vector field X to determine a left invariant flow $\theta: \mathbb{R} \times G \rightarrow G$, such that its speed vector field is X . Then, $\{\text{left invariant vector field}\} \xrightarrow{\text{integral}} \{\text{left invariant flow}\}$.

Finally, we use the combination of $g \xrightarrow{(II)} \{\text{Left invariant Vector Field}\} \xrightarrow{\text{integral}} \{\text{Left invariant Flow}\} \xleftrightarrow{(I)} \{\text{One-Parameter Subgroup}\}$ to achieve $g \rightarrow \{\text{One-Parameter Subgroup}\}$. It is obviously the inverse mapping of $\{\text{one-parameter subgroup}\} \xrightarrow{\text{differential}} g$.

As a result, we can put these four things together, i.e., $g \cong \{\text{left invariant vector field}\} \cong \{\text{left invariant flow}\} \cong \{\text{one-parameter subgroup}\}$. Then, we obtain the algebraic model of Lie group machine learning, as shown in Figure 1.4. Lie algebra g has four ‘identify’ aspects.

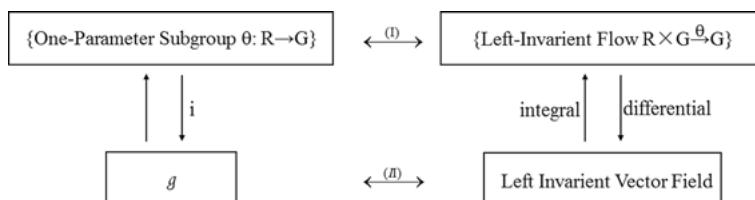


Fig. 1.4: Lie Group Machine Learning Algebras Model.

1.4 Geometric model of Lie group machine learning [2, 3, 8, 10, 12]

In this section, we present the geometric model of Lie group machine learning by using some geometric properties of the Lie group, such as translational and geodesic properties. They will enable us to learn the system representation and measurement.

In general, given an observation set in a learning system, the target results are learned according to the knowledge in the knowledge base. We assume that the observed set of observations is compactly connected so that we can take an invariant inner product $\langle \cdot, \cdot \rangle$ under the action of the automorphism $Ad(G)$ at the unit point in the observation set. We take a set of standard orthogonal bases and then use the left translation to expand them into the left invariant vector field of the observed set, so that it can uniquely set the inner product at every point of the observation set. The inner product is exactly the standard orthogonal basis of the tangent space of the point, thus forming a Riemannian space.

From the left invariance of the orthogonal vector field group, all left translations are isometric transformations of the Riemannian space. From the invariance of the inner product, all right translations are isometric transformations. Let $T_e(G)$ be a tangent space of a single point, g be a Lie algebra g of G , $T_a(G)$ be the tangent space of any sample point a , dl_a , dr_a be left transformations, l_a be a right transformation, and r_a be the induced linear mapping between the tangent spaces. Thus, we get the following geometric model structure, as shown in Figure 1.5.

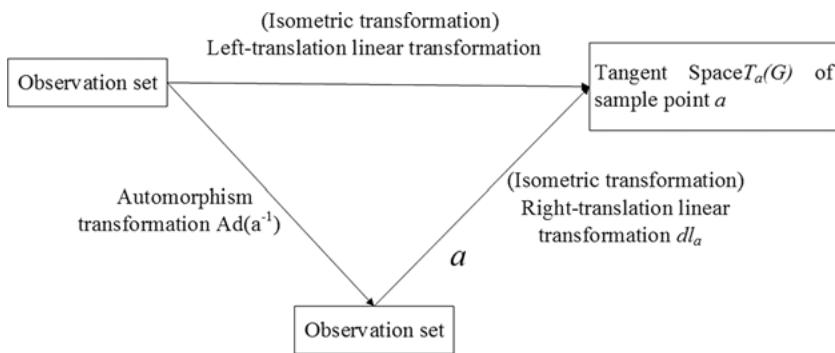


Fig. 1.5: Lie Group Machine Learning geometric model.

The basic steps of the isometric mapping algorithm of Lie group machine learning are given below.

Algorithm 1.12: Isometric transformation algorithm of Lie group machine learning

Initialisation: (X, τ) is a topological space, $x \in X$ //

Input: Sample set $X = \{x_1, x_2, \dots, x_n \in R^D\}$ //

Output: $Y = \{y_1, y_2, \dots, y_n \in R^d\}, d < D //$

- (1) Input sample set $X = \{x_1, x_2, \dots, x_n \in R^D\}$, X satisfies the Lie group structure.
- (2) Find an invariant inner product under the action of automorphism $Ad(G)$ on Lie algebra g of X :

$$\langle Ad(g)x_1, Ad(g)x_2 \rangle = \langle x_1, x_2 \rangle, x_1, x_2 \in g, g \in G.$$

- (3) Give a set of standard orthogonal bases $\{X_i; 1 \leq i \leq n\}$, $\langle X_i, X_j \rangle = \delta_{ij}$. Use the left transformation to transform $\{X_i; 1 \leq i \leq n\}$ into the left invariant vector field on X , denoted by X_i . This is to establish the Riemannian space structure.
 - (4) On this Riemannian space, make the left (or right) translational linear transformation on X , which constitutes the isometric transformation ϕ of the sample set: $X \rightarrow Y$, $Y = \{y_1, y_2, \dots, y_n \in R^d\}$, $d < D$.
-

A geodesic is one of the most basic and most important concepts of Riemann manifolds. It is the local shortest path in a Riemann manifold, which is very important for studying the manifold in high-dimensional space as a means of measurement. In a differential manifold, it is very difficult to calculate the distance of a geodesic directly because the manifold is locally Euclidean. Hence, the smaller part of the manifold is often replaced by a tangent plane so that the calculation of the geodesic can be transformed into Euclidean calculation of distance in a two-dimension plane.

In Lie group machine learning, suppose that the sample space G is embedded in the differential manifold M . For any sample a , the learning process determines a mapping $x_a: G \supset U_a \rightarrow V_a \subset R^D$, where (U_a, x_a) is called point a ($a \in U_a$) in the neighbourhood coordinate system of M , U_a is the coordinate neighbourhood of point a , and V_a is an open set of R^D .

Definition 1.18. Let (M, X) be a differential manifold, where X is a local coordinate system on U and $r_j: R^D \rightarrow R$ is the j -th coordinate function on R^D , i.e., $r_j(a_1, a_2, \dots, a_n) = a_j$. Then, the j -th coordinate function of x is $x_i: U \rightarrow R$, i.e., $x_j = r_j x$.

Definition 1.19. Let (M, X) be a differential manifold, $a \in M$, and $C^\infty(M, a, R)$ be a collection of real-valued functions in the neighbourhood of a . Then, the tangent vector at point a is a mapping, i.e., $v: C^\infty(M, a, R) \rightarrow R$. If x is a local coordinate system in the neighbourhood of x , then there exists a real-valued n tuple (a_1, a_2, \dots, a_n) . For every $f \in C^\infty(M, a, R)$, this n tuple satisfies

$$v(f) = \sum_{i=1}^n a_i \frac{\partial}{\partial r_i}(fx^{-1})|_{x(a)} . \quad (1.1)$$

The set of all tangent vectors M at point a is represented by $T_a(M)$, and it implies that M is a tangent space at point a . If $x: U \rightarrow R^D$ is a coordinate system at point a and $(\partial/\partial x_i)$ is the direction vector of x_i at point a , then $\{(\partial/\partial x_1), (\partial/\partial x_2), \dots, (\partial/\partial x_n)\}$ is the coordinate base of tangent space $T_a(M)$. If the inner product $g_{i,j}(a) = \langle (\partial/\partial x_i), (\partial/\partial x_j) \rangle$ is a differentiable function on U , then the distance between points a and b

in the tangent space is

$$D(a, b) = \sqrt{\sum_{i,j=1}^n g_{i,j}(a)(a_i - b_i)(a_j - b_j)} . \quad (1.2)$$

Therefore, in Lie group machine learning, the basic steps of the learning algorithm based on geodesic distance can be summarised as follows.

Algorithm 1.13: Geodesic distance algorithm of Lie group machine learning

Initialisation: (X, τ) is a topological space, $x \in X$ //

Input: Sample point a //

Output: The distance between all points in the coordinate neighbourhood of a and a //

- (1) Generate the neighbourhood U_a of the sample point a .
 - (2) Analyse the neighbourhood information of a in the sample set of points to take an automorphism under the invariant inner product $g_{i,j}(a)$, and calculate the value of $g_{i,j}(a)$.
 - (3) Substitute the resulting value $g_{i,j}(a)$ into Equation (1.2) to calculate the distance between each point in U_a and a .
 - (4) Output all values of distance (the smaller the distance, the closer is the result to be learned)
-

1.5 Axiom hypothesis of Lie group machine learning [2, 3, 5]

This section focuses on the axiom hypothesis of Lie group machine learning.

Hypothesis 1.1. Lie group machine learning generalisation ability hypothesis axiom including 1), 2), and 3):

- (1) Let G be a Lie group, and for any Lie sub-algebra h of Lie algebra g of G , there exists a unique connected Lie subgroup H of G , such that the following graphs are interchangeable.

$$\begin{array}{ccc} \eta \subset g & & \\ \exp \downarrow & \downarrow \exp & \\ H \subset G & & \end{array}$$

- (2) Let V be an n -dimensional linear space, and let G be a Lie subgroup of $\text{GL}(V)$. If

$$(g(x), g(y)) = (x, y) \forall x, y \in V, g \in G,$$

then the inner product (x, y) in V is called unchanged under G .

- (3) Set the left and right invariant Riemannian structures with a total volume of 1 in the compactive connected Lie group G , where f is any central function on it. Then, we have the following Weyl integral formula

$$\int_G f(g) dg = \frac{1}{|W|} \int_T f(t) |Q(t)|^2 dt \quad (1.3)$$

where $|W|$ represents the number of elements in W . When $t = \exp H$ ($H \in \eta$), $Q(t)$ can be denoted as

$$Q(\exp H) = \sum_{\sigma \in W} \text{sign}(\sigma) e^{2xi(\sigma(\delta), H)} \quad (1.4)$$

where δ is half of the sum of the roots of C^0 , i.e.,

$$\delta = \frac{1}{2} \sum_{\alpha \in \Delta^+(G)} \alpha.$$

Proof. First, we prove axiom 1). Let X_1, X_2, \dots, X_m be any set of bases from vector subspace h . We regard them as the left invariant vector field on G . Then, every tangent space of point g on G can be expanded into a subspace, i.e., $D(g) = \text{span}\{X_1(g), \dots, X_m(g)\}$, where $X_i(g)$ denotes the value of vector field X_i at point g . In the tangent space of each point, taking an m -dimensional space is called an m -dimensional distribution on G , which can be regarded as a high-dimensional promotion of the vector field. Therefore, all X_i ($1 \leq i \leq m$) are left invariant, i.e., $dl_\alpha(X_i(g)) = X_i(ag)$. The distribution D expanded by them is left invariant, i.e., $dl_\alpha(D(g)) = D(ag)$. Moreover, $\eta = \text{span}\{X_1(g), \dots, X_m(g)\}$ is a Lie sub-algebra; hence, $[X_i, X_j]$ ($1 \leq i, j \leq m$) can always use the linear combination of X_1, X_2, \dots, X_m for representation. This shows that the above distributions D satisfies the integrable conditions in the Frobenius theorem. Hence, there is a unique maximum integral submanifold corresponding to D , which is represented by unit element e , denoted by H .

Now, we prove that H is a connected subgroup. Let h be any element in H . For the left invariant of D , obviously, $l_h^{-1} \cdot H = h^{-1} \cdot H$ is a maximum integral submanifold of D . By definition, $e \in h^{-1} \cdot H$, i.e., $h^{-1} \cdot H$ passes point e . Owing to the uniqueness of the maximum integral submanifold, $h^{-1} \cdot H = H$. If h is arbitrary, then

$$H^{-1} \cdot H = \bigcup_{h \in H} h^{-1} \cdot H = h.$$

This proves that H is a connected subgroup. 2) omitted.

3) Proof: We prove it in the following steps. (1) From the analysis of the orbital geometry of \tilde{Ad} (accompanying transformation) on G , we know that T is a public geodesic submanifold and is orthogonal to each conjugate class. Moreover, $C_0 \cong T/W \cong G/\tilde{Ad}$ is a good basic domain on which we take any two points t_1 and t_2 such that their distance in \overline{C}_0 is the shortest distance of the conjugate class in G , referred to as the orbital distance.

(2) For any given inner point $t \in C_0$, its orbit type is G/T . For any given boundary point $t \in \partial \overline{C}_0 = \overline{C}_0 \setminus C_0$, its number of orbits is always less than $\dim G/T$. Therefore, it is not difficult to see the collection of non- G/T orbit types, i.e.,

$$G(\partial \overline{C}_0) = \bigcup_{x \in \partial \overline{C}_0} G(x).$$

In other words, when we perform integration on G , the above subset can be omitted, and we only need to calculate on $G(C_0)$.

(3) Let t be a moving point on C_0 , $m = \dim G/T = \dim G - \dim T$. Then, $G(t)$ is a G/T -type orbit and its m -dimensional volume is a function whose domain is C_0 . We call it the (main) orbital volume function and denote it by $v(t)$. Since each orbit $G(t)$ is the same G/T -type homogeneous Riemannian space, its volume ratio is equal to the ratio of its volume elements. We can take a benchmark of G/T homogeneous Riemannian space. For each G/T -type orbit $G(t)$, the coset $g \cdot T$ mapped to $g(t)$ is a homogeneous Riemannian space between the G -isometric mapping E

$$E: G/T \rightarrow G(t), g \cdot t \mapsto g(t) = gtg^{-1}.$$

Then, the ratio of the volume element of the above is the Jacobi determinant of the above mapping E_t , i.e., the object of the volume function $v(t)$ can be calculated by

$$v(t) = c \cdot \det(dE_t|_0), \quad (1.5)$$

where c is an undetermined proportional constant and $dE_t|_0$ is the linear mapping of the base point on the tangent space.

$$dE_t|_0: \oplus_{a \in \Delta^+} R_{(\pm a)}^2 \rightarrow \mathbb{J}_0(G(t)). \quad (1.6)$$

(4) Since E_t is G -isomorphic, $dE_t|_0$ is a T -norm (because the base point is a fixed point under the effect of T). Thus, $dE_t|_0$ can be decomposed into linear straight sums of the following two-dimensional T -invariant subspace

$$dE_t|_0, a: R_{(\pm a)}^2 \rightarrow \mathbb{J}_0(G_0(t)). \quad (1.7)$$

Hence, we can use the following formula to calculate $\det(dE_t|_0)$, i.e.,

$$\det(dE_t|_0) = \prod_{a \in \Delta^+} \det(dE_t|_0, a). \quad (1.8)$$

However, $\det(dE_t|_0)$ is proportional to the S^2 -type(main) orbit volume function in $G_a/T_a = \tilde{G}_a \cong S^3$. In other words, we can use the diagram

$$\begin{array}{ccc} Ga/T & \xrightarrow{E_{i,a}} & Ga(t) \subset Ga/Ta = \tilde{G} \\ \cap & & \cap \\ G/T & \xrightarrow{E_t} & G(t) = G \end{array} \quad (1.9)$$

to put the calculation of $\det(dE_t|_{0,a})$ into the S^2 -type volume function in \tilde{G}_a (\tilde{G}_a is an isomorphism with S^3 or $SO(3)$; hence, \tilde{G}_a has the same volume elements as S^3 and we thus see \tilde{G}_a as S^3 in the above problem). Then, we get

$$\det(dE_t|_{0,a}) = c' \cdot \sin^2 \pi(a, H) \quad (1.10)$$

where H satisfies $t = \exp H$ (note that the positive root of S^3 is 2Θ). By integrating formulas (1), (4), and (5), we get

$$v(\exp H) = c'' \cdot \sin^2 \pi(a, H) = c \cdot |\tilde{Q}(\exp H)|^2 \quad (1.11)$$

where $\tilde{Q}(\exp H) = \prod_{a \in \Delta^+} (e^{\pi i(a, H)} - e^{-\pi i(a, H)})$.

(5) Finally, we will prove that $\tilde{Q}(\exp H)$ in the above formula is equal to $Q(\exp H)$ in (*'). It is proved as follows: Let $\{a_1, \dots, a_k\}$ be the simple root system for the Weyl room C_0 . Then, according to the definition of reflection symmetry, we get $r\alpha_i(\alpha_i) = -a_i$, which is equivalent to the two half-space positive and negative sides divided by $\langle a_i \rangle \perp$, and it naturally corresponds to the other root a . When divided by $\langle a_i \rangle \perp$, the two selected half-space positive and negative sides have no effect. Hence, the next $\Delta^+ \setminus \{a_i\}$ still becomes $\Delta^+ \setminus \{a_i\}$, and we have

$$r\alpha_i(\Delta^+) = (\Delta^+ \setminus \{a_i\}) \cup \{-a_i\} \quad (1 \leq i \leq k). \quad (1.12)$$

Let $\delta = 1/2 \sum_{a \in \Delta^+} a$; according to (1.12), $r_{\alpha_i}(\delta) = \delta - a_i$, and $r_a(\xi) = \xi - 2(\xi, a)/(a, a)a$, we get

$$r_a(\delta) = \delta - \frac{2(\delta, a_i)}{(a_i, a_i)} a_i, \quad 1 \leq i \leq k.$$

Hence,

$$\frac{2(\delta, a_i)}{(a_i, a_i)} = 1 \quad (1.13)$$

Moreover, according to the expression of $\tilde{Q}(\exp H)$ and formula (1.12), we can infer that

$$\tilde{Q}(\exp r_a(H)) = (-1) \cdot \tilde{Q}(\exp H). \quad (1.14)$$

As a result, for the effect of W , $\tilde{Q}(\exp H)$ is an odd function, i.e.,

$$\tilde{Q}(\exp \sigma(H)) = \text{sign}(\sigma) \cdot \tilde{Q}(\exp H), \quad \sigma \in W \quad (1.15)$$

where $\text{sign}(\sigma)$ is the sign of σ for the determinant of the orthogonal transform on h .

There is a dominant item in the expansion $\tilde{Q}(\exp H)$. From (1.15), we can see that all the items $\text{sign}(\sigma) \cdot e^{2\pi i(\sigma(\delta), H)}$ should be included in the expansion of $\tilde{Q}(\exp H)$. Moreover, it is not difficult to prove from (1.13) that it cannot contain other items in the expansion, so it can only be

$$\tilde{Q}(\exp H) = Q(\exp H).$$

(6) Let f be a central function defined on G . When seeking G the integral $\int_G f(g) dg$, we first take the product along the G -orbit. Since f is equal on each G -orbit $G(t)$, we have

$$\begin{aligned} \int_G f(g) dg &= \int_{C_0} f(t)v(t) dt = \frac{1}{|W|} \int_T f(t)v(t) dt \\ &= \frac{1}{|W|} \int_T f(t) |Q(t)|^2 dt, \end{aligned} \quad (1.16)$$

where C_0 is an undetermined constant, and $|W|$ is the order of W (T is separated into $|W|$ pieces of Weyl rooms). Then, substitute $f(g) \equiv 1$ into (1.16); it is easy to determine that the above-mentioned constant must be 1. \square

Theorem 1.1 (Cartan–Weyl). A complex number of G indicates that the highest weight of multiplicity of \wedge_ϕ must be 1, and its characteristic function X_ϕ can be expressed with the highest formula as follows:

$$X_\phi(\exp H) = \frac{\sum_{\sigma \in W} \text{sign}(\sigma) e^{2\pi i(\sigma(\wedge\varphi + \delta), H)}}{\sum_{\sigma \in W} \text{sign}(\sigma) e^{2\pi i(\sigma(\delta), H)}}, \quad H \in h.$$

Thus, the necessary and sufficient condition for the equivalence of ϕ_1 and ϕ_2 is that the two complex irreducible terms of G are their highest weights, i.e., \wedge_{ϕ_1} is the same as \wedge_{ϕ_2} .

Proof. From the above analysis, we know that to prove

$$\chi_\phi(\exp H) \cdot Q(\exp H) = m(\wedge_\phi) \cdot \sum_{\sigma \in W} \text{sign}(\sigma) \cdot e^{2\pi i(\sigma(\delta), H)}$$

+ there may be ‘lower’ items, $m(\wedge_\phi) = 1$, and there cannot be any ‘lower’ item. The proof is as follows: $\text{In}(\int_G f(g) dg = 1/|W| \int_T f(t) |Q(t)|^2 dt)$, let $f(g) = |x\varphi(g)|^2$, $t = \exp H$, $H \in h$. From theorem,

$$\begin{aligned} 1 &= \int_G |\chi_\phi(g)|^2 dg = \frac{1}{|W|} \int_T \chi_\phi(t) |Q(t)|^2 dt \\ &= \frac{1}{|W|} \int_T \left| m(\wedge\varphi) \sum_{\sigma \in W} \text{sign}(\sigma) e^{2\pi i(\sigma(\wedge\varphi + \delta), H)} + \dots \right|^2 dt. \end{aligned}$$

From the point of view of $L_2(T)$, the expansion of the product of the right-hand side of the above equation is the expansion of $X_\phi(t) \cdot Q(t)$ for the standard orthogonal base $\{e^{2\pi i(\sigma(\wedge\varphi + \delta), H)}\}$. Hence, the above formula can be written as

$$\begin{aligned} l &= |\chi_\phi(g)|^2_{L_2(G)} = \frac{1}{|W|} |\chi_\phi(t) Q(t)|^2_{L_2(T)} \\ &= \frac{1}{|W|} \{[m(\wedge_\phi)]^2 \cdot |W| + \text{may be a lower order}^2\}. \end{aligned}$$

However, the above equation is only possible if $m(\wedge_\phi) = 1$ and there is no longer a lower order, which proves that $m(\wedge_\phi) = 1$ and the formula in the theorem holds.

Note: It is easy to see that the above-mentioned highest weight \wedge_ϕ must meet the following conditions

$$\frac{2(\wedge\varphi, ai)}{(ai, ai)} = q_i, \quad q_i \text{ are non-negative integers}, \quad a \in \prod$$

If the connected Lie group G is simply connected, it can be proved that for any given non-negative term $\{q_i; i = 1, \dots, k\}$, by the formula

$$\frac{2(\wedge, ai)}{(ai, ai)} = q_i \quad (i = 1, \dots, k)$$

only the definite \wedge must be the highest weight of a complex irreducible representation of G . Thus, there exists a complex irreducible $\phi: G \rightarrow \text{GL}(V)$ such that $\wedge = \wedge_\phi$. \square

Inference 1.1. The dimensionless $\dim \phi$ represented by the above irreducible can be calculated directly from the highest weight \wedge_ϕ using the following formula

$$d \dim \phi = \frac{\prod_{\alpha \in \Delta^+} (\wedge \varphi + \delta, \alpha)}{\prod_{\alpha \in \Delta^+} (\delta, \alpha)}.$$

Assuming 1.2 (Lie group machine learning duality hypothesis). In the learning system, the result of learning is based on knowledge in the knowledge base to be learned. The results of the study should be in the same knowledge base as the original knowledge. Toward this end, the following concepts are used to establish the dual space of the learning system.

Definition 1.20. Let P be a point on the Riemannian manifold M^n . If there is a pair of coins on M^n , $\sigma_p: M^n \rightarrow M^n$, $\sigma_p^2 = id$, making P an isolated fixed point, then M^n is symmetrical to P , σ_p is called the centre symmetry for point P , and X and $\sigma_p(X)$ ($X \in M^n$) are called the symmetry points of point P . If M^n is symmetrical about any point P , then M^n is called a symmetric space. In other words, the symmetry space is a symmetrical Riemannian manifold for any point on it.

- (1) In a Riemannian manifold, the simplest and most basic concept is pushing a geodesic that in the knowledge base is the most common knowledge corresponding to the collection.
- (2) Let P_1, P_2, P_3 be any three points in a connected Riemannian manifold M^n . Then, they satisfy the ‘triangular inequality’

$$d(P_1, P_2) + d(P_2, P_3) \geq d(P_1, P_3).$$

This relationship is used for measurement between knowledge bars.

- (3) The necessary condition for the point column $\{P_n; n \in N\}$ to converge to a certain point a (i.e., its limit) in the knowledge metric space X is that it satisfies the usual Cauchy condition, i.e., $\forall \varepsilon > 0$. There must be a sufficiently large N such that $m, n \geq N$, $d(P_m, P_n) < \varepsilon$. If the above Cauchy condition is also a sufficient condition for the existence of the limit point of the midpoint of X , then X is a complete measure space, and if a Riemannian manifold M^n is complete for the measure structure on it, it is called a complete Riemannian manifold
- (4) Parallelism and covariant differentiation.

Theorem 1.2. In any given Riemannian manifold M^n , there is a covariant differentiation that uniquely satisfies the following laws of operation

$$\begin{cases} \nabla_{f_1 X_1 + f_2 X_2} Y = f_1 \nabla_{X_1} Y + f_2 \nabla_{X_2} Y \\ \nabla_X (f_1 X_1 + f_2 X_2) = (Xf_1) Y_1 + f_1 \nabla_X Y_1 + (Xf_2) Y_2 + f_2 \nabla_X Y_2 \\ \nabla_X Y - \nabla_Y X = [X, Y] \\ \nabla_Z \langle X, Y \rangle = \langle \nabla_Z X, Y \rangle + \langle X, \nabla_Z Y \rangle \end{cases}$$

where $\langle X, Y \rangle = g(X, Y) = \sum_{i,j=1}^n g_{ij} \xi_i \eta_j$.

Proof. From

$$\begin{cases} \nabla_{f_1 X_1 + f_2 X_2} Y = f_1 \nabla_{X_1} Y + f_2 \nabla_{X_2} Y \\ \nabla_X (f_1 X_1 + f_2 X_2) = (Xf_1) Y_1 + f_1 \nabla_X Y_1 + (Xf_2) Y_2 + f_2 \nabla_X Y_2 \\ \nabla_X Y - \nabla_Y X = [X, Y] \\ \nabla_Z \langle X, Y \rangle = \langle \nabla_Z X, Y \rangle + \langle X, \nabla_Z Y \rangle \end{cases}$$

obtain

$$\begin{aligned} Z \langle X, Y \rangle &= \langle \nabla_Z X, Y \rangle + \langle X, \nabla_Z Y \rangle \\ &= \langle \nabla_X Z, Y \rangle + \langle X, \nabla_Z Y \rangle + \langle [Z, X], Y \rangle . \end{aligned}$$

Let X, Y, Z in above formula be exchanged, i.e., the other two types

$$\begin{aligned} X \langle Y, Z \rangle &= \langle \nabla_Y X, Z \rangle + \langle Y, \nabla_X Z \rangle + \langle [X, Y], Z \rangle \\ Y \langle Z, X \rangle &= \langle \nabla_Z Y, X \rangle + \langle Z, \nabla_Y X \rangle + \langle [Y, Z], X \rangle , \end{aligned}$$

to obtain

$$\begin{aligned} Z \langle X, \nabla_Z Y \rangle &= Z \langle X, Y \rangle + \langle Z, [X, Y] \rangle + Y \langle X, Z \rangle \\ &\quad + \langle Y, [X, Z] \rangle - X \langle Y, Z \rangle - \langle X, [Y, Z] \rangle . \end{aligned}$$

Thus, the left part of the above formula uniquely determines the value of $\nabla_Z Y$ at each point in the tangent space. This proves the existence and uniqueness of covariant differentiation.

Hence, the hypothetical axioms of dual spaces in Lie group machine learning include the following:

- (1) Let M be a pair of dual spaces and $G(M)$ be the conserved transformation group of M . Then, \wedge_ϕ is directly calculated:
 - (a) is a complete Riemannian manifold;
 - (b) M is a homogeneous space, i.e., the effect of $G(M)$ on M is deliverable;
 - (c) If K is a fixed grouping of subgroups to make a certain set of base point $P_0 \in M$, then K is a compact subgroup of $G(M)$, and is a Lie group;
 - (d) For any geodesic $r: R \rightarrow M$ in M , there must be a unique one-parameter subgroup $\Phi r: R \rightarrow G(M)$ in $G(M)$ that satisfies the following conditions

$$\begin{aligned} \Phi r(t_1) \cdot r(t_2) &= r(t_1 + t_2) \\ d\Phi r(t_1)|_{TM_{r(t_2)}} &= //r(t_2, t_1 + t_2) . \quad \square \end{aligned}$$

Proof.

- (1) To prove the completeness of M , we need to show that any geodesic in M can always extend indefinitely. Let $r: [a, b] \rightarrow M$ be any given geodesic in the symmetric spaces M , and let $\varepsilon < 1/2(b - a)$.

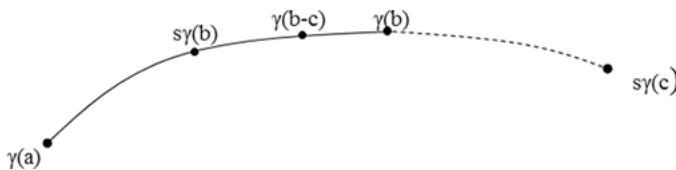


Fig. 1.6: The extension of r .

Let s and M be the central symmetry of the point of $r(b - \varepsilon)$. Then, r and $s \cdot r$ have convergence at a length of 2ε . This also means that r can extend to at least a length of $b - a - 2\varepsilon$ (as shown in Figure 1.6). This also means that any geodesic can always extend indefinitely. Therefore, according to the Hopf–Rinow theorem, M is complete.

- (2) Let p, q be any two points on M . From the completeness of M , there is the shortest geodesic section $r: [0, d(p, q)] \rightarrow M$ that links p, q and makes $r(0) = p, r(d(p, q)) = q$. Let s be the centre symmetry of M for the point $r(1/2(p, q))$ on M . Obviously, we have

$$s(p) = q, s(q) = p .$$

Hence, the effect of $G(M)$ on M can be delivered. This means that M is homogeneous.

- (3) Let K be a fixed grouping of subgroups to make a certain set of base points $P_0 \in M$, where $O(n)$ is the conservative transformation group of TM_{P_0} . It is not difficult to see that each element in K is uniquely determined by its induction of conservative transformation on TM_{P_0} . Thus,

$$\begin{array}{ccc} K & \rightarrow & O(n) \\ \cup & & \cup \\ k & \rightarrow & dk|_{P_0} \end{array}$$

is an isomorphism, and the ‘image’ of K is a closed subgroup of $O(n)$. Thus, K is a compact subgroup of $G(M)$ and a Lie group.

- (4) Let $\gamma: R \rightarrow M$ be a geodesic (its parameter is the arc length) in M . We temporarily use s_t to denote the central symmetry of M for point $\gamma(t)$. Let $X(t), t \in [t_1, t_2]$ be the definition of a parallel vector field on $\gamma([2t_1 - t_2, t_1])$. Then,

$$s_{t_1} X(t) = X'(2t_1 - t)$$

is the definition of a parallel vector field on $\gamma([2t_1 - t_2, t_1])$, and

$$X'(t_1) = s_{t_1} X(t_1) = -X(t_1) ,$$

as shown in Figure 1.7.

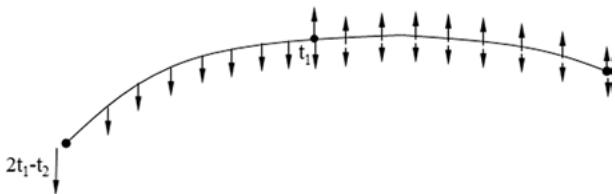


Fig. 1.7: Parallel vector field.

Thus,

$$\begin{aligned} s_{t_1} &= X(t_2) - X'(2t_1 - t_2) \\ &= //Y(t_1, 2t_1 - t_2)(-X(t_1)) \\ &= //Y(t_1, 2t_1 - t_2) \cdot //Y(t_2, t_1)(-X(t_2)) \\ &= -//Y(t_2, 2t_1 - t_2)X(t_2). \end{aligned}$$

In other words, under the centrosymmetric s_{t_1} , the two tangent vectors $X(t_2)$ and $X'(2t_1 - t_2)$ correspond to each other, and their relationship is reversed after the parallel movement along $\gamma([t_2, 2t_1 - t_2])$. Thus, two such symmetrical combinations of centres

$$\phi_\gamma(t_1) = s_{t_1/2} \cdot s_0 \quad t_1 \in R$$

is a conformity change that satisfies the following two conditions

$$\begin{cases} \phi_\gamma(t_1) \cdot \gamma(t_2) = \gamma(t_1 + t_2) \\ d\phi_\gamma(t_1)|_{TM_{\gamma(t_2)}} = //Y(t_2, t_1 + t_2) \end{cases} \quad t_1, t_2 \in R.$$

From the above, it is not difficult to see that $\phi_\gamma(t_1) \cdot \phi_\gamma(t'_1)$ and $\phi_\gamma(t_1 + t'_1)$ both are conformity changes satisfying the following conditions

$$\begin{cases} \phi_\gamma(t_1 + t'_1) \gamma(t_2) = \gamma(t_1 + t'_1 + t_2), \\ d(\phi_\gamma(t_1) \cdot \phi_\gamma(t'_1))|_{TM_{\gamma(t_2)}} = //Y(t_2, t_1 + t'_1 + t_2), \\ \phi_\gamma(t_1 + t'_1) \gamma(t_2) = \gamma(t_1 + t'_1 + t_2) \\ d(\phi_\gamma(t_1 + t'_1))|_{TM_{\gamma(t_2)}} = //Y(t_2, t_1 + t'_1 + t_2). \end{cases}$$

Thus,

$$\phi_\gamma(t_1) \phi_\gamma(t'_1) = \phi_\gamma(t_1 + t'_1).$$

In other words, $\phi_\gamma: R \rightarrow (M)$ is a one-parameter subgroup found in 4).

2° Let M be a symmetric space, P_0 be an arbitrarily determined basis point, S_0 be the centre symmetry of M for the base point P_0 , $G(M)$ be the conserved transformation group of M , and K be the preservative group to fix P_0 . Then, 1) $M \cong G(M)/K$, where K is a compact Lie group; 2) $\sigma: G(M) \rightarrow G(M)$ is defined as $\sigma(g) =$

$s_0 g s_0, \forall g \in G(M)$; then, σ is a dual automorphism of $G(M)$; 3) Let $F(\sigma)$ be a fixed point subset of σ , i.e., $F(\sigma) = \{g \in G(M) | \sigma(g) = g\}$, and let $F^0(\sigma)$ be the unit connected area of $F(\sigma)$; then, $F(\sigma)$ and $F^0(\sigma)$ are a closed subgroup of $G(M)$, and $F^0(\sigma) \subseteq K \subseteq F(\sigma)$; 4) Let g and k be the Lie algebras of $G(M)$ and K , respectively; then, $g = k \oplus p$, $k = \ker(d\sigma - id)$, $P = \ker(d\sigma + id)$; 5) For any given $X \in P$, $r(t) = \exp X(p_0)$ is a geodesic passing through p_0 in M , and $\exp tX = \Phi_r(t)(\Phi_r(t))$ is a one-parameter subgroup of $G(M)$. \square

Proof.

- (1) $G(M)$ is a transferable transform Lie group affecting M , where K is a closed subgroup of $O(n)$ on the of induction of TM_{p_0} . Hence, K is a compact Lie group, and we have $M \cdot G(M)/K$. Moreover, $\forall g \in G(M), \forall g \in G(M), \sigma(g) = s_0 g s_0$; hence, it is clear that $G(M)$ is a pair of automorphisms. Therefore, its fixed-point subset $F(\sigma)$ is a closed subgroup, and its unit connected area $F^0(\sigma)$ is also a closed subgroup.
- (2) Let $i: K \rightarrow O(n)$ is $i(k) = dk|_{P_0}, k \in K$. It is a natural isomorphism mapping, and

$$i(s_0) = -id \in O(n).$$

Thus, for any given $k \in K$, there always is

$$i(s_0 k s_0) = (-id) t(k) (-id) = t(k)$$

i.e., $s_0 k s_0 = k, K \subseteq F(\sigma)$. Moreover, let $\exp tX$ be a one-parameter subgroup satisfying $s_0 (\exp tX) s_0 = \exp tX$ in $G(M)$, i.e., $(\exp tX) \cdot s_0 \cdot \exp(-tX) = s_0, \forall t \in R$; then, $s_0 (\exp tX \cdot p_0) = (\exp tX) \cdot s_0 \exp(-tX) \exp(-tX) \cdot p_0 = \exp tX \cdot p_0, \forall t \in R$. In other words, $\{\exp tX \cdot p_0; \forall t \in R\}$ is a fixed point subset including p_0 of s_0 , while point p_0 is an isolated fixed point of s_0 . Thus,

$$\exp tX \cdot p_0 = p_0, \forall t \in R,$$

i.e., $\exp tX \in K$. This proves that $F^0(\sigma) \subseteq K$.

- (3) According to the assumption and 1) above, $d\sigma: g \rightarrow g$ is a pair of automorphisms of g . Thus,

$$g = k \oplus p, k = \ker(d\sigma - id), p = \ker(d\sigma + id)$$

where 2) proves that k is a Lie algebra of K (i.e., $F^0(\sigma)$). Moreover, for the characteristic of $X \in p$, $\exp tX$ is $s_0 (\exp tX) s_0 = \exp(-tX)$.

Let X_0 be the tangent vector of curve $\{\exp tX(p_0); t \in R\}$ at point p_0 on M , and γ be the geodesic at the starting velocity vector of X_0 . Then, $\phi_\gamma(t) = s_t \cdot s_0$ obviously is a one-parameter subgroup of $G(M)$ because

$$\begin{aligned} s_0 \phi_\gamma(t) &= s_0 (s_{\frac{t}{2}} \cdot s_0) = s_0 \cdot s_{\frac{t}{2}} \\ &= \varphi_\gamma(t)^{-1} = \varphi_\gamma(-t), \end{aligned}$$

thus, $\exp tX = \varphi_\gamma(t)$, $\gamma(t) = \exp tX(p_0)$.

This axiom implies that the close relationship among the dual space, the Lie group, and the Lie algebras, can be regarded as a particular Lie group and Lie algebras. \square

Hypothesis 1.2. Division of independence hypothesis of Lie group machine learning.

- (1) Let g be a real Lie algebra and σ be its dual automorphism. Then, g has a decomposition of σ

$$\begin{cases} g = k \oplus p \\ k = \ker(\sigma - id) \\ p = \ker(\sigma + id) \end{cases}$$

If $ad_p: k \rightarrow gl(p)$, $ad, (X) \cdot Y = [X, Y], X \in K, Y \in P$ is a one-to-one correspondence, and P has an ad , K is a constant inner product Q , i.e., for any given $X \in K, Y, Z \in P$,

$$Q([X, Y], Z) + Q(Y, [X, Z]) \equiv 0$$

is established, then (g, σ, Q) is an orthogonal duality Lie algebra.

- (2) Any orthogonal duality Lie algebra (g, σ, Q) can be uniquely broken down into the following formula

$$(g, \sigma, Q) = (g_0, \sigma_0, Q_0) \oplus (g_+, \sigma_+, Q_+) + (g_-, \sigma_-, Q_-)$$

where $g_0 = k_0 \oplus p_0, g_+ = k_+ \oplus p_+, g_- = k_- \oplus p_-$,

$$\sigma_0 = \sigma|_{g_0}, \sigma_+ = \sigma|_{g_+}, \sigma_- = \sigma|_{g_-},$$

$$Q_0 = Q|_{p_0}, Q_+ = Q|_{p_+}, Q_- = Q|_{p_-},$$

and $B|P_+, B|P_-$ represent positive and negative, respectively.

Hypothesis 1.3. Consistency hypothesis of Lie group machine learning,

- (1) Let Q be the homomorphism of Lie group G_1 to Lie group G_2 . Then, $Q^{-1}(I_2)$ is the closed normal subgroup of G_1 . Let π be the natural homomorphism of G_1 to $G_1(Q^{-1}(I_2))$. Then, the homomorphism ϕ of $G_1/Q^{-1}(I_2)$ to Lie group G_2 makes $Q = \phi\pi$.
- (2) Let G be the n -dimensional connected exchange group (or complex Lie group). Then,

$$G \cong R^k \otimes T^{n-k}$$

where $T^{n-k} = \underbrace{R/Z \otimes R/Z \otimes \cdots \otimes R/Z}_{n-k}$ is the $n - k$ -dimensional torus.

Definition 1.21. If there is an n^2 polynomial set of S , the necessary and sufficient condition for $g = (g_{ij}) \in G$ is

$$P(g) = P(g_{ij}) = 0 (\forall p \in S)$$

Then, the subgroup G of the general linear group $(GL(n, \mathbb{R}), GL(n, \mathbb{C}))$ is an algebraic group.

Tab. 1.1: Classical Lie Group and its Lie Algebras.

name	symbol	Lie Algebras	dimension
Euclidean space	\mathbb{R}^n	\mathbb{R}^n	n
real face	$T^n = \mathbb{R}^n / \mathbb{Z}^n$	\mathbb{R}^n	n
complex torus	$T\mathbb{C}^n = \mathbb{C}^n / \mathbb{Z}^n + \sqrt{-1}\mathbb{Z}$	\mathbb{C}^n	n
general linear group	$GL(n, F)$	$gl(n, F)$	n^2
special linear group	$SL(n, F)$	$sl(n, F)$	$n^2 - 1$
orthogonal group	$O(n, F)$	$o(n, F)$	$1/2n(n-1)$
special orthogonal group	$SO(n, F)$	$so(n, F)$	$1/2n(n-1)$
Symplectic group	$SP(n, F)$	$sp(n, F)$	$2n^2 + n$
(p,q) type Lorenz group	$O(p, q, F)$	$o(p, q, F)$	$(p+q)(p+q-1)/2$
Unitary group	$U(n)$	$u(n)$	n^2
Special Unitary group	$SU(n)$	$su(n)$	$n^2 - 1$
(p,q) type Unitary group	$U(p, q)$	$u(p, q)$	$(p+q)^2$
Special (p,q) type Unitary group	$SU(p, q)$	$su(p, q)$	$(p+q)^2 - 1$
(p,q) type Symplectic group	$SP(p, q)$	$sp(p, q)$	$2(p+q)^2 + p + q$
Special orthogonal stars group	$SO^*(2n)$	$so^*(2n)$	$2n^2 - n$
Unitary Symplectic type	$U^*(2n)$	$u^*(2n)$	$4n^2$
special two stars group	$SU^*(2n)$	$su^*(2n)$	$4n^2 - 1$

If there is a $2n^2$ polynomial set of S , the necessary and sufficient condition for $g = a_{ij} + b_{ij}\sqrt{-1} \in G$ ($a_{ij}, b_{ij} \in \mathbb{R}$) is $P(g) = P(a_{ij}, b_{ij}) = 0, \forall p \in S$. Then, the subgroup G of $GL(n, \mathbb{C})$ is a pseudo-algebraic group.

Theorem 1.3. *The algebraic and pseudo-algebraic groups of general linear groups are both closed subgroups and therefore are Lie subgroups.*

Let $F = \mathbb{R}$ or \mathbb{C} and regard the Lie algebras of $GL(U, F)$ as $gl(n, F)$. Then, $x \in gl(n, F), \exp tx = e^{tx}$

If H is a closed subgroup of G and h is a Lie algebra of H , then

$$h = \{x \in gl(n, F) | e^{\pi x} \in H, \forall x \in F\}$$

Now, we list the commonly used Lie group and Lie algebras in Table 1.1.

1.6 Geometric learning algorithm for dynkin graphs in Lie group machine learning [6–8]

In this section, we use geometric methods to describe the Lie group machine learning system. In this case, we start our discussion with the complex semisimple Lie algebras of study space G , because complex semisimple Lie algebras constitute the most basic and most important part of Lie algebra theory [1]. Complex semisimple Lie algebras consist of complex simple Lie algebras, which are determined by their roots. The root structure is represented by roots, which are determined by Dynkin graphs. Therefore,

we propose a Dynkin graph geometry learning algorithm based on Lie group machine learning.

1.6.1 Overview of dynkin graphs in Lie group machine learning

In Lie group machine learning, suppose that g is a complex semi simple Lie algebra of study space G and η is its Cantan sub-algebra. Furthermore, g decomposes the root subspace of η into

$$g = \eta + \sum_{\alpha \in \Delta} g_\alpha$$

where Δ is the root system of η and $|\Delta|$ is the number of elements. Further, η^* is the dual space of η ; hence, $\Delta \subset \eta^*$. Then, let (x, y) represent the Killin g type of g and $\Pi = \{\alpha_1, \alpha_2, \dots, \alpha_l\}$ be the simple root system of complex semi simple Lie algebra g . Then, Π can be decomposed into the sum of two orthogonal subsets, and each subset cannot be decomposed

$$\Pi = \prod_1 \cup \prod_2 \cup \dots \cup \prod_k; \quad \left(\prod_i, \prod_j \right) = 0 \quad (i \neq j). \quad (1.17)$$

Naturally, this division of Π must derive an equivalence relationship in Π ; each Π_i is an equivalence class. This equivalence class relationship can be established in the following way.

Let $\alpha, \beta \in \Pi$, if there exists a sequence in Π

$$\alpha = \alpha_{i1}, \alpha_{i2}, \dots, \alpha_{im} = \beta$$

such that $\alpha_{ij}, \alpha_{ij+1} \neq 0$ ($j = 1, 2, \dots, m - 1$), then α and β have a relationship denoted by $\alpha \sim \beta$.

Obviously, the relationship \sim has the following properties:

- (1) $\alpha \sim \alpha$;
- (2) if $\alpha \sim \beta$, then $\beta \sim \alpha$;
- (3) if $\alpha \sim \beta, \beta \sim \gamma$, then $\alpha \sim \gamma$;

Thus, \sim is an equivalence relationship. The same equivalence class cannot be decomposed into two mutually orthogonal subsets. Different equivalence classes are orthogonal to each other, i.e., the division of the equivalence relation corresponds to Π (1.17).

Definition 1.22. If the simple root system Π cannot be decomposed into the sum of two non-empty mutually orthogonal subsets, then Π is an irreducible simple root system; otherwise, Π is reducible.

In Lie group machine learning, the reducibility of the simple root system is used to determine whether the learning problem is solvable, i.e., if the simple root system Π is irreducible, then the learning problem is solvable. By contrast, if Π is reducible,

then the learning problem is not solvable. Thus, the solvability discriminant algorithm of Lie group machine learning can be described as follows:

Algorithm 1.14: Solvability discriminant of Lie group machine learning

Initialisation: Input data set simple root system of Lie algebras is $\Pi = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$ //

Input Π //

Output: {0, 1} //

```

procedure SOLVABLE1( $\Pi$ )
 $DECOMPOSE(\Pi)$  //decompose  $\Pi$  //
if not ( $\Pi = \Pi_1 \cup \Pi_2 \cup \dots \cup \Pi_k$  and  $((\Pi_i, \Pi_j) = 0 \text{ and } i \neq j)$ )
    then return 0;
endif
for i=1 to k do
    if  $DECOMPOSE(\Pi_i)$  // $\Pi_i$  is decomposable//
        then return 0; // the learning problem is not solvable//
        endif
repeat
endfor
return 1; // the learning problem is solvable //
end SOLVABLE1

```

Definition 1.23. Let $\Pi = \{\alpha_1, \alpha_2, \dots, \alpha_l\}$ be a simple root system of complex semisimple Lie algebras, use l point denoted $\alpha_1, \alpha_2, \dots, \alpha_l$, and the i -th and j -th use the number of

$$\frac{2(\alpha_i, \alpha_j)}{(\alpha_i, \alpha_i)} \cdot \frac{2(\alpha_i, \alpha_j)}{(\alpha_j, \alpha_j)} = r_{ij}$$

lines to be connected. If $(\alpha_i, \alpha_i) < (\alpha_j, \alpha_j)$, then draw an arrow pointing from point j to point i . The resulting graph is called the Dynkin diagram of Π (or g).

Obviously, if Π is decomposable (1.17), then the Dynkin diagram of Π can be composed into k interconnected branches.

For convenience of study, we can abstract the property of the simple root system as the property of the space vector group.

Definition 1.24. Let E be an n -dimensional study space, $\Pi = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$ be a positive vector group of E , and when $i \neq j$, $2(\alpha_i, \alpha_j)/(\alpha_i, \alpha_i) \in Z_-$. We call $\Pi = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$ as a π -system.

The following four conclusions are obvious.

- (1) Π is a linear unrelated group, so $m \leq n$. Π is a π -system of E 's subspace $L(\Pi)$; when $m = 1$, Π is the base of E .
- (2) Any subset Π_1 of Π is a π -system.
- (3) If $\alpha, \beta \in \Pi$, $\alpha \neq \beta$, then

$$0 \leq \frac{2(\alpha, \beta)}{(\alpha, \alpha)} \cdot \frac{2(\alpha, \beta)}{(\beta, \beta)} \leq 3 .$$

- (4) The simple root system Π of semisimple Lie algebras is a π -system.

Definition 1.25. If the π -system \prod cannot be decomposed into a sum of two orthogonal non-empty subsets, then \prod is an irreducible π -system.

Obviously, an irreducible simple root system is an irreducible π -system.

In the same way that the Dynkin graphs of the simple root system are constructed under Definition 1.23, we can also construct a graph, called the Dynkin diagram of \prod , for each π -system \prod .

Obviously, a π -system \prod is irreducible if and only if its Dynkin diagram is connected. Hence, we can also judge the solvability of learning problems through the Dynkin diagram of the simple root system. The algorithm is described as follows:

Algorithm 1.15: Solvability discriminant of Lie group machine learning

Initialisation: Let the input space vector group be denoted by $\prod = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$, D is its Dynkin diagram, it contains n nodes $\alpha_1, \alpha_2, \dots, \alpha_n$ //

Input: D //

Output: {0, 1} //

```
procedure SOLVABLE2(D)
for i=1 to n-1; j=2 to n do
    if there is no path from  $\alpha_i$  to  $\alpha_j$ 
        then return 0; // the learning problem is not solvable //
    else
        i++;
        j++;
    endif
    repeat
    endfor
    return 1; // the learning problem is solvable //
end SOLVABLE2
```

If \prod_1 is a subset of \prod , then the Dynkin diagram of \prod_1 is the Dynkin diagram with \prod elements of the points and their interconnected lines are retained, and the remaining points and lines are removed from the map. The Dynkin diagram of \prod_1 is also the subset Dynkin diagram of \prod .

1.6.2 Classification algorithm of dynkin diagrams in Lie group machine learning

In this section, we will give some basic algorithms such as the isomorphism algorithm and classification algorithm for Lie group machine learning, based on some basic theorem of Dynkin diagrams in complex semisimple Lie algebras.

Theorem 1.4 (conjugate theorem of Cantan sub-Algebras). *Let η_1, η_2 be two Cantan sub-algebras of complex semisimple Lie algebras. Then, η_1 and η_2 are conjugate on $\mathcal{E}(g)$, and there exists $\theta \in \mathcal{E}(g)$ such that $\theta(\eta_1) = \eta_2$.*

Proof. Let b_1, b_2 respectively be Borel sub-algebras of g including η_1, η_2 ; then, η_1, η_2 are Cantan sub-Algebras of b_1, b_2 , respectively. For $\tau \in \varepsilon(g)$ such that $\tau(b_1) = b_2$, $\tau(b_1)$ is the Cantan sub-algebra of g and b_1 . For $\sigma \in \varepsilon(b_2) \subseteq \varepsilon(g)$ such that $\sigma(\tau(\eta_1)) = \eta_2$, $\theta = \sigma\tau \in \varepsilon(g)$ such that $\theta(\eta_1) = \eta_2$. \square

Theorem 1.5. A complex semisimple Lie algebra corresponds to only one Dynkin diagram.

Proof. Let η_1, η_2 be two Cantan sub-algebras of g , Δ_1, Δ_2 be root systems of η_1, η_2 , respectively, for g , and Π_1, Π_2 be simple root systems of Δ_1, Δ_2 , respectively. From Theorem 1.4, there exists $\theta_1 \in \varepsilon(g)$ such that $\theta_1(\eta_1) = \eta_2$. Since $(\theta_1(x), \theta_1(y)) = (x, y), \forall x, y \in g; \theta_1([x, y]) = [\theta_1(x), \theta_1(y)], \forall x, y \in g$. Hence, $\theta_1(\Delta_1) = \Delta_2, \theta_1(\Pi_1)$ is a simple root system of η_{2R} of a certain order. Let W be a Weyl group determined by η_2 . It is known that $w \in W$ such that $w(\theta_1(\Pi_1)) = \Pi_2$, and $\theta_2 \in \varepsilon(g)$ such that $\theta_2|_{\eta_{2R}} = w$; hence, $\theta = \theta_2\theta_1 \in \varepsilon(g)$ such that $\theta(\eta_1) = \eta_2, \theta(\Pi_1) = \Pi_2$. Thus, the Dynkin diagram determined by Π_1, Π_2 is the same. \square

Theorem 1.6. Let g, g' be two complex semisimple Lie algebras. Further, $(X, Y); (X', Y')$; η, η' ; and η_R, η'_R , respectively, are Killin g-type of g, g' , Cantan sub-algebras, a root system, and the Euclidean space generated by the root system. In addition, φ is a linear isomorphism of η_R to η'_R , and $\varphi(\Delta) = \Delta'$; then, φ is an isomorphism of the Euclidean space. Thus, $(\varphi(x), \varphi(y))' = (x, y), \forall x, y \in \eta_R$, and φ can be developed for g to g' of Lie algebra isomorphism.

Theorem 1.7. Two complex semisimple Lie algebras have isomorphism if and only if their Dynkin diagrams are the same.

Proof. Let the Dynkin diagrams of the complex semisimple Lie algebras g, g' be the same, i.e., g, g' have Cantan sub-algebras η, η' ; root systems Δ, Δ' ; and simple root systems $\Pi = \{\alpha_1, \alpha_2, \dots, \alpha_n\}, \Pi' = \{\alpha'_1, \alpha'_2, \dots, \alpha'_n\}$, respectively. Further, there is a mapping of Π to Π' satisfying

$$\varphi(\alpha_i) = \alpha'_i, 1 \leq i \leq n, \quad (1.18)$$

$$\frac{2(\alpha_i, \alpha_j)}{(\alpha_i, \alpha_i)} = \frac{2(\alpha'_i, \alpha'_j)}{(\alpha'_i, \alpha'_i)}, 1 \leq i, j \leq n. \quad (1.19)$$

Since Π, Π' are the bases of η_R, η'_R , respectively, Φ can be exploited for the linear isomorphism of η_R to η'_R .

Let W, W' be the Weyl groups of g, g' , respectively; hence, from (1.18), we know that $r_{\alpha_i}, r'_{\alpha_i}$ have the same matrix under the base of Π, Π' , i.e.,

$$M_{\{\alpha_1, \alpha_2, \dots, \alpha_n\}} r_{\alpha_i} = M_{\{\alpha'_1, \alpha'_2, \dots, \alpha'_n\}} r'_{\alpha_i} \quad (1.20)$$

while $\{r_{\alpha_i} | 1 \leq i \leq n\}, \{r'_{\alpha_i} | 1 \leq i \leq n\}$ are generated groups of W, W' , respectively; hence, there is an isomorphism mapping Φ^* of W to W'

$$\varphi_*(w) = \varphi w \varphi^{-1} = w', \quad \forall w \in W \quad (1.21)$$

Suppose that $\alpha \in \Delta$, and there exists $w \in W$, $\alpha_i \in \Pi$, such that $w(\alpha) = \alpha_i$; hence,

$$\varphi(\alpha) = \varphi(w^{-1}(\alpha_i)) = \varphi w^{-1} \varphi^{-1}(\varphi(\alpha_i)) = (w')^{-1} \alpha'_i \in \Delta' ,$$

and $\varphi(\Delta) \subseteq \Delta'$. Similarly, $\varphi^{-1}(\Delta') \subseteq \Delta$, so $\varphi(\Delta) \subseteq \Delta'$.

By Theorem 1.6, we know that Φ can be developed for g to g' of Lie algebra isomorphism, i.e., isomorphism of g and g' .

By contrast, let Φ be an isomorphism of complex semisimple Lie algebras from g to g' ; then,

$$(\Phi(x), \Phi(y))' = (x, y), \forall (x, y) \in g . \quad (1.22)$$

Let η be a Cantan sub-algebra of g , where Δ, Π are the root system and simple root system, respectively; then, $g = \eta + \sum_{\alpha \in \Delta} g_\alpha$. Here, $\eta' = \Phi(\eta)$ is a Cantan sub-algebra of g' , and $g' = \eta' + \sum_{\alpha' \in \Delta'} g'_\alpha$.

For $h \in \eta, ea \in ga$, we have

$$\begin{aligned} \varphi([h, e_\alpha]) &= (\alpha, h)\varphi(e_\alpha) , \\ [\varphi(h)\varphi(e_\alpha)] &= \varphi([h, e_\alpha]) , \end{aligned}$$

Hence, $\Phi(ga) = g\Phi(a)$, $\varphi(\Delta) = \Delta'$. Further, $\varphi(\Pi) = \Pi'$ is a simple root system of $\Phi(\eta_R)$ of a certain order, and from (1.21), we know that the Dynkin diagrams of Π and Π' are the same.

According to the above theorem, we can get the study space isomorphism discriminant algorithm in Lie group machine learning: \square

Algorithm 1.16: Study space isomorphism discriminant algorithm in Lie group machine learning

Initialisation: D_1, D_2 are two Dynkin diagrams corresponding to data sets in the study space //

Input: D_1, D_2 //

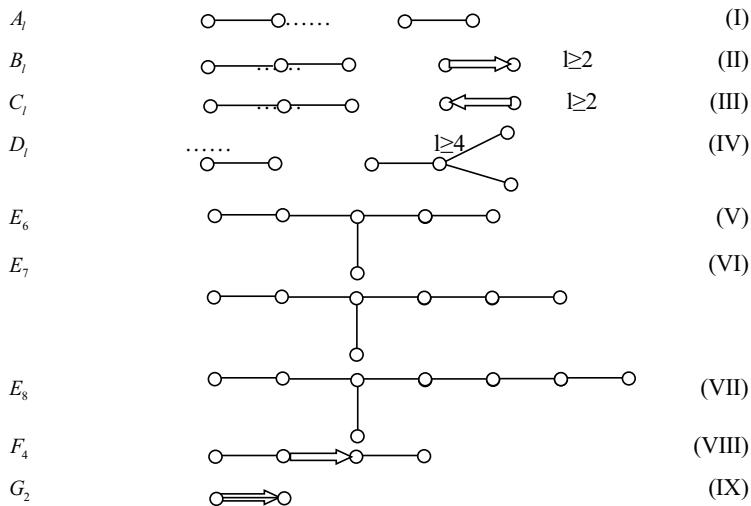
Output: $\{0, 1\}$ //

```

procedure ISOMORPHIC ( $D_1, D_2$ )
if EQUAL( $D_1, D_2$ )
    then return 1; // Dynkin diagrams are the same, then the two study spaces have isomorphism//
    else return 0;
endif
end ISOMORPHIC

```

Through the definition and property of the Dynkin diagram, we can see that in the sense of isomorphism, the input data vector space has its corresponding Dynkin diagram, and only $A_l, B_l, C_l, D_l, E_6, E_7, E_8, F_4$, and G_2 (a total of 9 types) are possible [1, 6]:



Thus, suppose that there are n input data in the study space. These data are denoted by the vector group $\prod = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$. Using the Dynkin diagram for classification, the classification algorithm can be described as follows:

Algorithm 1.17: Classification algorithm of Dynkin diagram in Lie group machine learning

Initialisation: Let the input space vector groups be denoted by $\prod = \{\alpha_1, \alpha_2, \dots, \alpha_n\} //$

Input: $\prod //$

Output: $\{A_l, B_l, C_l, D_l, E_6, E_7, E_8, F_4, G_2\} //$

procedure CLASSIFICATION(\prod)

if SOLVABLE1(\prod)=1 //the learning problem is solvable//

then use n points to denote $\alpha_1, \alpha_2, \dots, \alpha_n$

for $i=1$ to $n-1$; $j=2$ to n do

Draw line $r_{ij} = 2(\alpha_i, \alpha_j)/(\alpha_i, \alpha_i) \cdot 2(\alpha_i, \alpha_j)/(\alpha_j, \alpha_j)$ connect i -th point and j -th point;

if $(\alpha_i, \alpha_i) < (\alpha_j, \alpha_j)$

then draw an arrow pointing from the j -th point to the i -th point;

endif

repeat

endfor

endif

if there is a loop

then return;

endif

case //Classification//

Figure(I): A_l ;

Figure(II): B_l ;

Figure(III): C_l ;

Figure(IV): D_l ;

Figure(V): E_6 ;

Figure(VI): E_7 ;

Figure(VII): E_8 ;

```
Figure(VIII): F4;  
Figure(IX): G2;  
else: return  
endcase  
end CLASSIFICATION
```

This algorithm shows that if the learning problem is solvable, then we can draw the Dynkin diagram corresponding to the learning space by using $A_l, B_l, C_l, D_l, E_6, E_7, E_8, F_4, G_2$, and so on. According to the Dynkin diagram formed by the data, we can find the corresponding Lie group, such as A_l can correspond to the special unit group $SU(l + 1)$ and B_l can correspond to the special orthogonal group $SO(2l + 1)$. According to the corresponding Lie group structure, we can find the corresponding solution. Thus, we can use the Lie group as the main basis for machine learning theory.

1.7 Linear classifier design of Lie group machine learning

1.7.1 Linear classifier design of Lie group machine learning

Machine learning is a practical subject. With the advent of the information age, machine learning algorithms are encountering various types of data, such as aerospace data, biological data, and crystal data. According to the definition of Lie group machine learning $G \times M \rightarrow M$, these specific observational data G can be constructed as a corresponding typical group for processing, and these typical groups can be embedded into the higher-order $U(n)$ group as a subgroup for study. In general, the linear classifier can be structured as follows:

- Step 1: Map the sample set to this non-empty set of G ;
- Step 2: Construct the corresponding Lie group structure according to G ;
- Step 3: Apply the resulting Lie group to the established Lie group machine learning model;
- Step 4: Form the corresponding classifier;
- Step 5: Example Test;
- Step 6: Apply.

The second step in the process of structuring the classifier is very important. It can be described as follows. First, describe all the aspects that meet certain conditions as the observation set, each of which is an element of this collection. Second, define a linear combination for a given set, and if any linear combination of any set of elements in the set is still an element of the set, then a linear space is formed. Then, a set of groups in the linear space is selected again. Finally, for any two elements A_1 and A_2 , define the operations $[A_1, A_2]$ in the linear set, the result of which is still an element of the

set, and for the elements A_1 , A_2 , and A_3 and the constants f_1 , f_2 , and f_3 , if we have

$$[A_1, f_2 A_2 + f_3 A_3] = f_2 [A_1, A_2] + f_3 [A_1, A_3] \quad (1.23)$$

$$[f_1 A_1 + f_2 A_2, A_3] = f_1 [A_1, A_3] + f_2 [A_2, A_3] \quad (1.24)$$

$$[A_1, A_2] = -[A_2, A_1] \quad (1.25)$$

$$[A_1, [A_2, A_3]] + [A_3, [A_1, A_3]] + [A_2, [A_3, A_1]] = 0 \quad (1.26)$$

then this linear set called a Lie algebra. With the Lie algebra, we can get the corresponding Lie group.

1.7.2 Lie group machine learning SO(3) classifier

For a 0 or 1 classification problem, suppose that the sample set G is $(x_1, y_1), (x_2, y_2), \dots, (x_l, y_l) \in \mathbb{R}^N \times Y$, where $Y = \{-1, +1\}$, to seek classifier $f(X)$, such that $y \cdot f(x) > 0$.

Then, the $\text{SO}(3)$ classifier of Lie group machine learning will be constructed on the basis of the typical group $\text{SO}(n)$.

Definition 1.26. The set of all orthogonal transforms in n -dimensional vector space is called an n -dimensional orthogonal group $O(n)$.

In this case, the problem is transformed into the three-dimensional vector space, i.e., the three-dimensional orthogonal group $O(3)$ is discussed. This group can also be defined as a group of all 3×3 orthogonal matrices, since these two groups have isomorphism [12].

Definition 1.27. If R is an orthogonal matrix, it satisfies the equation $RR^T = R^T R = I$, where I is the identity matrix and R^T is the transpose matrix of R . Take the determinant of both sides of the equation, and note that $\det R = \det R^T$, yielding $(\det R)^2 = 1$ or $\det R = \pm 1$, such that the matrix of $O(3)$ groups is divided into two groups. A set of matrix determinants is $+1$ and the other set of determinants is -1 . Then, all orthogonal matrices with a determinant of $+1$ form a group called a three-dimensional special orthogonal group, and it is represented by $\text{SO}(3)$, i.e., $\text{SO}(3) = \{O \in O(3) | |O| = 1\}$. The $\text{SO}(3)$ classifier algorithm is described as follows:

Algorithm 1.18: Classification algorithm of $\text{SO}(3)$ classifier

Initialisation: G is a group, its group elements are $\{x_1, x_2, \dots, x_n\}$ //

Input: G //

Output: $\{1, -1\}$ //

procedure SOClassifier (G)

if G is a 3×3 orthogonal matrix group

//That is, this input data set is a three-dimensional orthogonal group $O(3)$ //

then

for $i = 1$ to n do

//calculate the matrix determinant value of each group element //

if $\det x_i == -1$

```

    then return -1;
endif
repeat
endfor
if i = n + 1
    then return 1; //G is an SO(3) group//
endif
endif
end SOClassifier

```

In a Lie group, this typical group $SO(n)$ is closely related to many high-dimensional data in the real world. Many of the data can be expressed in this group. Readers may refer to the relevant physical and chemical literature.

1.7.3 Classification of text based on Lie group machine learning

In the traditional classification algorithm, it is often necessary to measure the similarity between two documents; commonly used metrics are the Euclidean distance, vector angle cosine, and so on. In Lie group machine learning, we consider the vector space of the document to be embedded in a differential manifold, each of which corresponds to a point on the manifold, and we then use the geodesic distance instead of the Euclidean distance to measure the similarity between the two documents. We refer to this new text classification method as Lie group classification (LGC).

The LGC algorithm is described as follows:

Algorithm 1.19: LGC

- 1 The training document vector is described according to the set of feature words. Suppose that the feature set of document i is $t_i = (t_{i1}, t_{i2}, \dots, t_{in})$ and the corresponding weight of document i is $w_i = (w_{i1}, w_{i2}, \dots, w_{in})$. Then, the vector representation of document i is $d_i = (t_{i1}, w_{i1}; t_{i2}, w_{i2}; \dots; t_{in}, w_{in})$. Thus, all document vectors form a vector space, which is embedded in a differential manifold. Each document is a point on the manifold.
 - 2 For a new document, determine the vector representation of the document according to the feature set.
 - 3 According to Algorithm 1.13, generate the coordinate neighbourhood U_a of a new document a , i.e., the document set S , establishing its tangent space local coordinate system, computing the geometric distance of every document s_k and new document a in set S
- $$D(a, s_k) = \sqrt{\sum_{i,j=1}^n g_{i,j}(a)(a_i - s_{ki})(a_j - s_{kj})}.$$
- 4 Compute the weight of each point in S for the classes of document c_x $W(d, c_x) = \sum_{s_k \in S} D(a, s_k) \times f(s_k, c_x)$,
where $f(s_k, c_x)$ is the class function. Thus, if s_k belongs to class c_x , its function value is 1; otherwise, it is 0.
 - 5 Comparing the weights of the classes, the document a belongs to the largest weight of the class.
-

We used the Chinese language management open platform (<http://www.nlp.org.cn>) at Fudan University. Dr. Li Ronglu has developed a text categorisation corpus for a text classification performance test. The corpus has a total of 2816 documents, divided into transportation, sports, military, and 10 other categories. We selected 1882 documents as training documents and 934 documents as test documents.

We used the LGC algorithm to classify these documents. The classification of the various documents is shown in Table 1.2 and Table 1.3.

We used three methods, namely the LGC algorithm, KNN (K nearest neighbour), which uses the Euclidean distance as a metric, and SVM (support vector machine), to classify these documents. From Table 1.4, we can see that the recall and precision of text categorisation with LGC are higher than those with KNN and SVM. Because the text classification algorithm in Lie group machine learning regards the sample vector space as a differential manifold, the document vector space is considered to be curved, and it is more capable of dealing with heterogeneous and complex data properties than the Euclidean space description document set. It not only fully reflects the benefits of differential manifold geometric characteristics but also the advantages of

Tab. 1.2: Classification of various documents.

	traffic	sports	military	medicine	political	education	environment	economic	art	computer
Number of test document	71	149	83	68	167	73	67	108	82	66
Number of current document	69	166	62	60	199	71	60	144	79	54
Number of correct document	67	147	59	57	159	66	55	98	75	54

Tab. 1.3: Check rate and precision rate of various documents.

	traffic	sport	military	medicine	political	education	environment	economic	art	computer
Check rate (%)	94.366	98.658	71.084	83.824	95.21	90.411	82.09	90.741	91.463	81.818
Precision rate (%)	97.101	88.554	95.161	95	79.899	92.958	91.667	85.965	94.937	100

Tab. 1.4: Classification performance of two kinds of text.

	Macro average (%)			Micro average (%)		
	Check rate	Precision	F ₁ value	Check rate	Precision rate	F ₁ value
LGC	87.966	92.124	89.997	89.615	89.615	89.615
KNN	85.517	91.885	88.642	88.009	88.009	88.009
SVM	85.333	89.308	87.275	87.045	87.045	87.045

the Lie group geometric nature. The experimental results show that the LGC algorithm improves the performance of article classification.

1.8 Chapter summary

The Lie group is a natural combination of algebra and geometry, which has many applications in the field of computer science. This chapter introduced the Lie group into the field of machine learning, discussed the inherent connection between the Lie group and machine learning, and presented the Lie group machine learning model as well as related algorithms. It mainly dealt with (1) the basic concepts of Lie group machine learning and the algebraic and geometric models of Lie group machine learning; (2) The Dynkin diagram of the geometric learning algorithm in Lie group machine learning; (3) The classifier construction method of Lie group machine learning and the structure of the SO(3) classifier in Lie group machine learning.

Bibliography

- [1] Seung HS, Lee DD. The manifold ways perception [J]. SCIENCE, 2000, 290, 2268–2269.
- [2] Xu H. Lie Group Machine Learning Model and application research (Master thesis). Soochow University, 2007.
- [3] Li FZ, Xu H. The theory framework of Lie group machine learning (LML). Computer technology and application, 2007, 13(4), 251–255.
- [4] Li FZ, Kang Y. The theoretical framework based on Lie Group Machine Learning. Journal of Yunnan Nationalities University, 2004, 13(4), 251–255.
- [5] Xu H, Li FZ. Lie group machine learning's axioms hypothesizes. In proceedings of 2006 IEEE International Conference on Granular Computing, 401–402.
- [6] Xu H, Li FZ. Geometry Algorithms of Dynkin Diagrams in Lie Group Machine Learning. Journal of Nanchang Institute of Technology, 2006, 25(2), 75–78.
- [7] Xu H, Li FZ. Algorithms of Dynkin Diagrams in Lie Group Machine Learning. Journal of Communication and Computer, 2006, 4(3), 13–17.
- [8] Li FZ. Research on the Lie group machine learning. Technical Report, School of Computer Science and Technology, Suzhou University, P.R. China, 2004.
- [9] Xu H, Li FZ. The Design of SU(n) Classifier of Lie Group Machine Learning (LML). Journal of Computational Information Systems, 2005, 1(4), 835–841.
- [10] Li FZ, Xu H. Study on Lie group machine learning. Journal of Computational Information Systems, 2005, (194), 843–849.
- [11] Li FZ, Xu H. SO(3) Classifier of Lie Group Machine Learning (LML). Journal of Communication and Computer, 2006, 4(3), 76–80.
- [12] Li FZ, He SP, Lin TY. Lie group machine learning. The second annual Young Scientists in Jiangsu Province, 2008, Nanjing.

2 Lie group subspace orbit generation learning

This chapter is divided into four sections. The first section introduces the basic concepts of partial order and lattice in Lie group machine learning (LML). The second section describes the LML subspace orbit generation lattice learning algorithm. The third section discusses the subspace orbit generation lattice learning algorithm under the action of a general linear group $GL_n(\mathbb{F}_n)$ in LML. Finally, the fourth section summarises the chapter.

2.1 Basic concepts of partial order and lattice in LML

The basic concepts of Lie group machine learning are given in [1, 2]. Based on these studies, we present the related theories of Lie group machine learning space, learning subspace, orbit, target orbit, partial order, lattice, etc.

2.1.1 Basic Concepts

Definition 2.1. Lie group machine learning space: The set of all the paths (or orbits) used to describe the learning process, which consists of all the learning elements, is called the learning space. It consists of five parts, i.e., {learning examples, learning algorithms, input data, output data, representation theory}, expressed as $S = \{EX, AL, X, Y, TE\}$.

Definition 2.2. Let M be the state set, which represents the various states (including the initial and target states) $\{m_0, m_1, \dots\}$ in the learning process, where m_i ($i = 0, 1, \dots$) are the states of different moments in the learning process.

Definition 2.3. Let T_i be an operation operator, which represents an operation of a state in the learning process. Each operation of the operator causes the learning problem to transition from one state to another. The sequence T_0, T_1, \dots, T_{g-1} of operation operators used from the initial state m_0 to the target state m_g is a solution to the learning problem.

Further, G_n is the set of state transition functions; $G_n = \{T_0, T_1, \dots\}$ represents each step (operation operator) set in the learning process, or it can be understood as a representation of the degree of completion of the learning problem at different stages.

Definition 2.4. Learning subspace: A spatial description of a particular learning problem, i.e., a subset of the learning space, includes all possible learning paths for learning this problem. It is represented by a quaternion $\{M, G_n, P, C\}$, where P is the sample data set, i.e., the object to be learned, C is the objective function, M is the state set, and G_n is the set of state transition functions.

Definition 2.5. Let E be the cost of learning the problem from one state to another. When the total cost of the entire learning process $\sum_{i=0}^{g-1} E(T_i)$ is minimum, the resulting orbit (path) is the optimal orbit.

Definition 2.6. Orbit: An object that is already learnt constitutes a path, namely the learning path; it is a type of geometric description of the learning process. There must be a learning path in the learning subspace that encompasses a series of learning operations (operators) in the learning process. It is represented by a triplet $\{L(M), F, E\}$, where $L(M)$ represents the state set through which the learning path passes, and $L(M) \subset M$; F represents the set of state transition functions used in the learning path, and $F \subset G_n$; and E is the orbital learning cost.

All orbits form the entire learning subspace; a state subset and the operator used constitute an orbit, and an orbit constitutes a partial chain.

Definition 2.7. Target orbit: The optimal (approximate optimal) learning path that can correctly classify the sample data, i.e., the learning function (set) expected to be selected by the learning algorithm from the learning subspace, is the goal of learning the problem.

The purpose of learning is to find the optimal path from the initial state to the target state. If each state is treated as a node, the learning subspace can be represented by a directed graph. This graph is not necessarily fully connected, i.e., some states may not be able to reach another state, but in the case of a connected state, each arc represents an operator, and steering of the state can be achieved (i.e., transfer of the state can be achieved by the connectivity). When a specific problem is solved, a specific cost can be set for the operator according to different problems. If there is a path leading to the target state (node) from the initial state (node), then the learning problem represented by the target state is solvable. In the graph structure, the tree is a special directed graph with only one path between any two nodes. The tree has only one starting node (root node), and the path of the tree does not have loops. Thus, the tree avoids the case in which the target state cannot be reached in the directed graph (as long as there is a target state). Each branch of the tree forms a partial set, forming an orbit, and the terminal (leaf) node is the largest (smallest) node. Of course, some states can be generated by other states at the same time, i.e., the learning subspace is the structure of the graph (or net), and the tree is a special description of it. According to the connection between nodes, the corresponding matrix can be used to represent the cost between the nodes by the value of the matrix element. We express each group as a matrix in a Lie group and realise the process of solving the problem by the nature and operation of different groups, which can be seen in the Lie group learning problems and studied through the structure of the map. The learning problem of Lie groups can be studied from the graph structure.

Theorem 2.1. *Learning space \supset learning subspace \supset orbit \supset target orbit.*

Figure 2.1 shows the relationship between learning space, learning subspace, orbit, and target orbit.

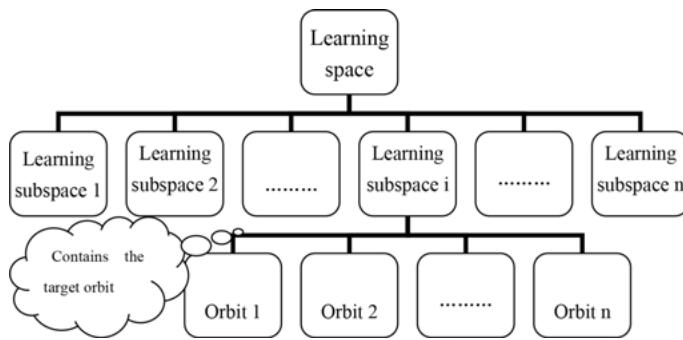


Fig. 2.1: learning space, learning subspace, orbit, target orbit diagram.

Theorem 2.2. *Real-world problems can be described by learning subspaces, and a learning subspace contains all possible trajectories. If the problem is solvable, it is necessary to find a learning path from it as well as the optimal (approximate optimal) learning path [18].*

Proof. The learning problem is mapped to the corresponding learning subspace. From the initial state of the problem, in the course of the study, for each step of the orbit division, describe all possible descent states of the current state, and according to the specific criteria for determining the best state, perform further division until the target orbit is found, which ensures that we can find the desired orbit. It should be noted that in the specific learning algorithm, considering the computation time and storage space, all the states are generally not extended in large-scale learning problems, but regardless of whether it is a global or local optimisation problem, the optimal (near optimal) orbit must be found according to the information of the learning problem.

In dealing with learning problems, we must find not only the target but also the optimal path (minimum cost). An orbit generation algorithm is said to be admissible when the target orbit is found as long as it exists.

The learning process can be described as a different state, and state conversion can be accomplished by an operation (operator). Thus, there must be a path from the initial state to the target state; there are multiple paths, and at least one path is the shortest (optimal) relative to the other paths.

In the learning algorithm, to generate the order of nodes of a learning subspace describing the structure of a tree or graph, we can consider the following two cases: depth first and breadth first. The breadth-first algorithm generates the learning subspace layer by layer, i.e., transfer to the next layer occurs only when the state of a layer is fully generated. Because the breadth priority must be extended to all the states of the layer before entering the next layer, the first generation state is first extended, so that the breadth-first algorithm can ensure that the shortest path from the initial state to the target state is found. Thus, the length of the path must be greater than or equal to that of the previous path for the second time to reach a certain state, because a good

path does not enter a state many times; it simply gives it up the second time it arrives at a certain state. By contrast, in the case of the depth-first algorithm, when a state is generated, all of its child states and descendent nodes of the sub-state must be generated before the brother state. In the generation of learning sub-spaces, the depth-first algorithm goes as deep as possible. It can not only generate the states of the descendants of the state but also consider its brother state. The depth priority ensures that the path that encounters a state for the first time is the shortest path to this state. For any state, it is possible to find another path to it. If the length (or cost) of the path is critical to learning, the shortest path should be maintained when the same state is encountered multiple times. For global generation, the breadth-first algorithm must be able to find the best path. The depth-first algorithm generates solutions locally; although its learning efficiency is higher, it does not necessarily find the best path.

Because all states of each layer are expanded in breadth priority, there is a need for sufficient space to store these state nodes. When using depth precedence, nodes are generated down a single path, and only when there is no descent state, is another path considered. This approach is not conducive to reaching the target state from the initial state via the optimal path, and for many problems, the resulting depth may be infinitely deep or easily follow an unprofitable path. Thus, the breadth-first and depth-first approaches have a certain blindness property in the process of orbit generation. Therefore, to identify a learning orbit effectively, some special information related to the specific field of learning is required, namely heuristic information [3], which guides this study in the most promising direction. □

Definition 2.8. Let $f(x)$ be the valuation function, which represents the cost estimate of the optimal orbit of the state x from the initial state m_0 to the target state m_g . The general form is $f(x) = g(x) + h(x)$, where $g(x)$ is the actual cost paid from the initial state m_0 to the state x , and $h(x)$ is the estimated cost of the optimal orbit of the state x to the target state m_g , which is also called the heuristic function.

We use the valuation function to determine the extended state that is most likely to reach the target state in each state extension. It may extend outward along a certain edge area that is considered the most promising. Using heuristic information, we can find an optimal path from the initial state to the target state. Because there is no need to blindly expand all possible states, it is only possible to reach the target state along the most promising state. Thus, a relatively small range is generated, and it requires a relatively small storage space.

According to the relationship between the particularity and the generality of learning, the learning subspace is expressed as a form of partial order. The maximal element of the partially ordered set is the most general point in the learning subspace, i.e., the point with zero description. Such a point does not have any restrictions on learning examples; hence, it can describe anything. The minimal element of the partial order is the most specific point in the learning subspace, i.e., it corresponds with the learning sample (the sample itself can be regarded as a specific concept).

Therefore, the learning sample set P is considered to be standardised into a partial order and mapped to a corresponding learning subspace as required, so that it can correspond to the reasonable operation operator set F ($F \subset G_n$). Then, we use partially ordered set theory to outline this type of learning problem.

2.1.2 Partially ordered set in LML

The sample data in the study of the Lie group is regulated by some basic theories that satisfy the following conditions. Accordingly, we will discuss the related problems [4–8]. To better explain and understand the following definitions, we first elaborate a simple example.

Suppose a dog is classified by whether or not it is ‘dangerous’, and the character of the dog is described in terms of its colour, size, and hair type. The colour values are brown and black; the sizes are large, medium, and small; and the hair types are smooth and curly. There are 12 dogs in the sample set, as shown in Table 2.1; 0 and 1 are used to indicate that the dog is ‘dangerous’ and ‘non-dangerous’, respectively.

The sample set $P = \{1, 2, \dots, 12\}$ is based on the size of the dog for division for the first time (here, temporarily to determine the rules for discussion). Then, we get $m_1 = \{1, 2, 6, 9\}$, $m_2 = \{3, 5, 10, 11\}$, and $m_3 = \{4, 7, 8, 12\}$. Here, m_1 is defined as 1 and no further division is made. For m_2 and m_3 , colour is used as the basis for division. We get $m_{21} = \{10, 11\}$, $m_{22} = \{3, 5\}$, $m_{31} = \{4, 7\}$, and $m_{32} = \{8, 12\}$. The categories of m_{21} and m_{31} are determined to be 0, and the division of m_{22} and m_{32} is based on hair type. Thus, we have $m_{221} = \{3\}$, $m_{222} = \{5\}$, $m_{321} = \{7\}$, and $m_{322} = \{8\}$, where m_{221} and m_{322} belong to category 0, and m_{222} and m_{321} belong to category 1. Thus, the category of each sample has been identified, i.e., the classification is complete.

Tab. 2.1: Dog’s sample set.

Serial number	Color	Body form	Wool type	Category
1	Black	Large	Curly	1
2	Brown	Large	Smooth	1
3	Brown	Medium	Curly	0
4	Black	Small	Curly	0
5	Brown	Medium	Smooth	1
6	Black	Large	Smooth	1
7	Brown	Small	Curly	1
8	Brown	Small	Smooth	0
9	Brown	Large	Curly	1
10	Black	Medium	Curly	0
11	Black	Medium	Smooth	0
12	Black	Small	Smooth	0

Definition 2.9. Let the LML sample data set P be a nonempty set and \geq be a binary relationship defined on P . If the following three axioms PO1–PO3 are established, P is called a partially ordered set and \geq is called a partial order on P (order for short).

PO1: For any $x \in P$, there is $x \geq x$;

PO2: For any $x, y \in P$, if $x \geq y$ and $y \geq x$, then $x = y$;

PO3: For any $x, y, z \in P$, if $x \geq y$ and $y \geq z$, then $x \geq z$.

In addition to the above three axioms, if the following axiom PO4 is also established, then P is called a serially ordered set or chain and \geq is called the total order on P .

PO4: For any $x, y \in P$, at least one of $x \geq y$ and $y \geq x$ holds.

Let T be a subset of the partially ordered set P , $m \in T$, and if there is no $x \in T$, let $m < x$ ($m > x$), where m is called the maximal element (or minimal element) of T . If for all $x \in T$, $m \geq x$ ($m \leq x$), then m is called the largest element (or smallest element) of T . If $u \in P$, for all $x \in T$, i.e., $u \geq x$ (or $u \leq x$), u is called an upper bound (or lower bound) of T . Note that the upper bound (lower bound) of T does not necessarily belong to T . If u is the upper bound of T , and for any upper bound of T $v \geq u$, then u is called the supremum of T , and the infimum of T can be defined. According to PO2, if T has an upper bound (or lower bound), it must be unique and is denoted as $\sup T$ (or $\inf T$). Similarly, the supremum (or infimum) of T does not necessarily belong to T .

If $x, y \in P$ and $x \leq y$, where $[x, y] = \{z \in P \mid x \leq z \leq y\}$, then $[x, y]$ is called an interval with x and y as endpoints. $x < y$.

If $x, y \in P$, $x < y$, and there is no $z \in P$ such that $x < z < y$, we say that y is the coverage of x , denoted as $x < \cdot y$.

The sample set P in the example is a partially ordered set, and m_3 is a sub-partial set of it. Further, m_{321} is a minimal element of m_3 and a lower bound, $[m_{321}, m_3]$ is called an interval, and m_{32} is the coverage of m_{321} .

For example, in face recognition, the eyes, nose, mouth, and other local features are used to identify the face. These features are studied from different angles. When viewed from top to bottom, i.e., considering the positional relation ‘higher than’ to describe the partial order \geq , we get eyes \geq nose \geq mouth. By contrast, in the case of bottom up observation, we get mouth \geq nose \geq eyes. Of course, these local features are generally rectangular. From the perspective of independent individual observation, they are separate from each other, i.e., there is no intersection. However, from the neighbourhood point of view, they are continuous, i.e., their union forms the whole face.

As another example, we consider statements $a_1, a_2, \dots, a_i, \dots, a_n$ that constitute a program. The statements have a certain order of execution, and there is a relationship between most of the statements, i.e., only if the statement a_i is executed, the statement a_{i+1} can be executed and the corresponding result is obtained, and the order of execution between such statements can be defined as a partial order. During the execution of the program, the order of some statements can be changed or other statements can be used in the implementation, but in a local (sub) block (such as $a_i \dots a_j$), they are

often in a partial order relationship, i.e., ‘before’. Further, $[a_i, a_j]$ is called an interval and a_{i+1} is the coverage of a_i .

Definition 2.10. Let the LML sample data set P be a partial set and P' be a nonempty subset of P . Obviously, P' for the partial order of P is also a partial order, called a sub-partially ordered set of P .

A chain (i.e., a totally ordered set) that contains a limited number of elements is known as a finite chain; otherwise, it is known as an infinite chain.

If the sub-partially ordered set S of a partially ordered set P is a chain, S is called a chain of P .

Let P be a partially ordered set, $x, y \in P$, and $x < y$. If there exists $x = x_0, x_1, \dots, x_n = y$, make

$$x = x_0 < x_1 < x_2 < \dots \leq x_n = y \quad (2.1)$$

where x is called the starting point and y is the ending point of the chain, referred to as x, y chain, and n is called its length.

If $x_i < x_{i+1}$, chain (2.1) is called the maximal chain of x, y . If

$$x = x'_0 < x'_1 < x'_2 < \dots < x'_m = y \quad (2.2)$$

is also a chain with x as the starting point and y as the ending point, and each $x_i (1 \leq i \leq n)$ appears in (2.2), then (2.2) is called the refinement of (2.1). Assuming that x is the starting point, the chain with y as the ending point can be refined into a maximal chain and the maximum value of the maximal chain with the starting point of x and ending point of y exists, and it is denoted by $d(x, y)$. Obviously, $d(x, x) = 0$. If a chain with x as the starting point and y as the ending point cannot be refined into a maximal chain, or the length of a maximal chain with x as the starting point and y as the ending point is without a maximum, define $d(x, y) = \infty$. If the chain with x as the starting point and y as the ending point can be refined into a maximal chain, and the maximum chain with x as the starting point and y as the ending point length is equal, make $l(x, y) = d(x, y)$, and call it ‘long’ from x to y .

In the above-mentioned case of determining whether a dog is dangerous, $m_{321} < m_3 < P$ is the chain with the starting point as m_{321} and the ending point as P , and it can be refined into $m_{321} < m_{32} < m_3 < P$.

In the learning problem, although the sample data are sometimes very large, the research object is often limited. Let $x (x_0)$ be the initial state of the sample, $x_1 \dots x_{n-1}$ be the states generated during the learning process, $Y(x_n)$ be the target state, and n be the generated orbital length. If the length of the orbit is used as a measure, (2.2) is superior to (2.1) in the form of orbital (2.1), thus take the price generated in each state as the standard, to compare the cost of both. $d(x, y)$ is the maximum possible length of the learning orbit.

Definition 2.11. The partial order P is said to satisfy the Jordan–Dedekind condition, referred to as the *JD* condition, if for any $a, b \in P$, and $a < b$, all great chains with a as the starting point and b as the ending point have the same limited length.

The JD condition is satisfied when the states from state a to state b are different (or not all the same) but the number of states on the orbit is the same, i.e., the length of the orbit is unchanged.

Definition 2.12. Let P and P' be partially ordered sets, where the partial order in P is denoted as \leq and the partial order in P' is denoted by \leq' . Assume that $f: P \rightarrow P'$ is a bijection. For any $x, y \in P$, $x \leq y$, if and only if $f(x) \leq' f(y)$, then f is called an isomorphic mapping of P to P' , and P and P' are called isomorphisms, denoted as $P \cong P'$.

In the real world, to better describe the problems that we deal with, we can map the sample data set P to a new learning subspace through a certain transformation and then make full use of the invariance of the transformation, such as translation invariance and relationship invariance. On the one hand, we can regard P' as a new set of samples generated by P , i.e., the orbital division of sample set P does not change its essential characteristics to ensure the correctness of orbital division. On the other hand, for the two isomorphic sample sets P and P' , the same orbital partitioning method can be used to avoid repetitive learning.

For example, in face recognition based on various local features (eyes, nose, mouth, etc.), take a number of sample points and their neighbourhood points. The original neighbourhood relation remains the same after the mapping of the sample points, i.e., the neighbouring samples remain adjacent to the mapped space in the original space. This fully reflects the relationship between actual problems and digital laws. In fact, regardless of the complexity of the formal method, the goal is to reveal the laws of nature and produce, i.e., the theory comes from nature and serves the natural. Machine learning as a method of data analysis is no exception.

2.1.3 Möbius function on local finite partial order in LML [23]

Consider human learning as an example. In the whole process of learning, there must be a sequence of relations between the subject and the knowledge that we have learned. Only by learning the basics of knowledge, can we learn and understand the follow-up courses. Similarly, in the discipline, the order of learning knowledge is not arbitrary, e.g., learning addition and subtraction followed by multiplication and division in the study of mathematics. Each part of knowledge combination is constructed into a complete knowledge structure system. The process is summarised as local formation of the whole and overall dependence on local information. This is also consistent with the overall principles of cognitive science.

When the sample set is mapped from the original high-dimensional space to the low-dimensional space, the local partial order relation between the samples is unchanged, so that the original structure and the intrinsic structure relation remain after the sample is mapped. The samples satisfying the partial order in the original space

still satisfy the partial order relation in the learning subspace. For example, given half of a lion's image, most people still recognise this mapping, although the mapping is missing nearly half of the dimensions, because the human brain has the ability to think of the overall image from the fitting of local information. If the whole image is severely distorted, although this mapping does not seem to have lost any one-dimensional information, almost no one can identify it, because the meaningful local information has disappeared [9].

The local finite partial set in LML is defined as follows:

Definition 2.13. Let the sample data set P in the LML be a partially ordered set. If for any $x, y \in P$ and $x < y$, the interval $[x, y]$ is a finite set, then P is called the local finite partially ordered set. If P is a finite set, P is called a finite partially ordered set.

Definition 2.14. Let the sample data set P in LML be a local finite partially ordered set, R be a commutative ring with the unit element, and $\mu(x, y)$ be a binary function defined on P and valued in R , assuming that $\mu(x, y)$ satisfies the following three conditions:

- (1) For any $x \in P$, there is always $\mu(x, x) = 1$;
- (2) If $x \not\leq y$, then $\mu(x, y) = 0$;
- (3) If $x < y$, then $\sum_{x \leq z \leq y} \mu(x, z) = 0$.

Let $\mu(x, y)$ be a Möbius function defined on P and valued in R .

The Möbius function can be used to determine the parameters of the relationship between the states in the learning process. If the two states are equivalent, the function value is 1; otherwise, it is 0.

Lemma 2.1. Let the sample data set P in LML be a local finite partially ordered set. $x, y \in P$, if $x < y$, then $d(x, y) < \infty$.

Proposition 2.1. The local finite partially ordered set of the sample data set P in LML must have a Möbius function and is unique.

Proof. Let the sample data set P in LML be a locally finite partially ordered set. First, we prove that there must be a Möbius function on P . We define

$$\begin{aligned}\mu(x, x) &= 1 \text{ for any } x \in P, \\ \mu(x, y) &= 0 \text{ if } x, y \in P \text{ and } x \not\leq y.\end{aligned}$$

Let $x, y \in P$ and $x \leq y$ for the induction of $d(x, y)$ to define $\mu(x, y)$. When $d(x, y) = 0$, $x = y$, and $\mu(x, y) = 1$ as defined above, set $d(x, y) > 0$. For $z \in P$ and $x \leq z \leq y$, there is obviously $d(x, z) < d(x, y)$; hence, $d(x, z)$ is defined. Thus,

$$\mu(x, y) = - \sum_{x \leq z \leq y} \mu(x, z).$$

According to the definition of $\mu(x, y)$, it is suitable for (1), (2), and (3). Thus $\mu(x, y)$ is the Möbius function on P .

Let $\mu'(x, y)$ also be a Möbius function on P . From (1), for any $x \in P$, we have $\mu(x, x) = 1$. Therefore,

$$\mu'(x, x) = \mu(x, x) = 1 \text{ for any } x \in P.$$

Let $x, y \in P$. If $x > y$, according to (2), we have $\mu'(x, y) = 0$. Hence,

$$\mu'(x, y) = \mu(x, y) = 0, \text{ if } x > y.$$

We set $x \leq y$. Mathematical induction is applied to $d(x, y)$ to prove that $\mu'(x, y) = \mu(x, y)$. When $d(x, y) = 0$, $x = y$, it is certified that $\mu'(x, x) = 1 = \mu(x, x)$. We set $d(x, y) > 0$. For $z \in P$ and $x \leq z \leq y$, obviously $d(x, z) < d(x, y)$. According to the induction hypothesis, $\mu'(x, z) = \mu(x, z)$. Then,

$$\begin{aligned} \mu'(x, y) &= - \sum_{x \leq z \leq y} \mu'(x, z) \quad (\text{as } \mu'(x, y) \text{ is suitable for condition (3)}) \\ &= - \sum_{x \leq z \leq y} \mu(x, y) \quad (\text{based on the induction hypothesis}) \\ &= \mu(x, y). \quad (\text{as } \mu(x, y) \text{ is suitable for condition (3)}) \quad \square \end{aligned}$$

Proposition 2.2. Let the sample data set P in LML be a locally finite partially ordered set with the smallest element O and R be the commutative ring with the unit element. Let $\mu(x, y)$ be a Möbius function defined on P and valued in R . Further, $f(x)$ is a function defined on P and valued in R . For any $x \in P$, make

$$g(x) = \sum_{y \leq x} f(y). \quad (2.3)$$

Then,

$$f(x) = \sum_{y \leq x} g(y) \mu(y, x). \quad (2.4)$$

Instead, let $g(x)$ be a function defined on P and valued in R . For any $x \in P$, by (2.4), to define $f(x)$, (2.3) is set up.

Proof. Since P is a locally finite partially ordered set with the smallest element O , the interval $[0, x]$ is a finite set and the sum in (2.3) is a finite sum. Therefore, $g(x)$ defined by (2.3) is reasonable. Similarly, the sum in (2.4) is also finite and

$$\begin{aligned} \sum_{y \leq x} g(y) \mu(y, x) &= \sum_{y \leq x} \left(\sum_{z \leq y} f(z) \right) \mu(y, x) \quad (\text{substituting (2.3)}) \\ &= \sum_{z \leq x} f(z) \sum_{z \leq y \leq x} \mu(y, x) \quad (\text{exchange summation order}) \\ &= \sum_{z \leq x} f(z) \delta_{z,x} \quad (\text{according to condition (3)}) \\ &= f(x) \end{aligned}$$

where $\delta_{z,x}$ is the delta function.

By contrast,

$$\begin{aligned}
 \sum_{y \leq x} f(y) &= \sum_{y \leq x} \left(\sum_{z \leq y} g(z) \mu(z, y) \right) \quad (\text{substituting (2.4)}) \\
 &= \sum_{z \leq x} g(z) \sum_{z \leq y \leq x} \mu(z, y) \quad (\text{exchange summation order}) \\
 &= \sum_{z \leq x} g(z) \delta_{z,x} \quad (\text{according to condition (3)}) \\
 &= g(x) .
 \end{aligned}$$

Where (2.4) and (2.3) are called the Möbius inversion formulas of (2.3) and (2.4), respectively.

Further, $\mu(x, y)$ is used as the control weight generated by the orbit to guide the generation of the new sample, and $f(x)$ and $g(x)$ are the transform functions of samples x and y , respectively. \square

2.1.4 Gaussian coefficients and gaussian polynomials in LML

The complexity of the objective function of learning depends on its representation. In the ideal case, a representation that matches a particular learning problem should be chosen. Thus, a common preprocessing strategy in machine learning involves changing the expression of sample data, such as

$$x = (x_1, \dots, x_n) \mapsto \phi(x) = (\phi_1(x), \dots, \phi_N(x)) .$$

This step is equivalent to mapping the sample data set to a new space, $F = \{\phi(x) | x \in X\}$.

Example: the objective function

$$f(m_1, m_2, r) = C \frac{m_1 m_2}{r^2} .$$

Given the law of gravitation, consider two objects with mass m_1, m_2 and distance r to represent gravity. This law is expressed by observable quantities, i.e., mass and distance, but cannot be expressed by linear functions. However, if the coordinate is changed simply as

$$(m_1, m_2, r) \mapsto (x, y, z) = (\ln m_1, \ln m_2, \ln r) ,$$

the expression is given by

$$g(x, y, z) = \ln f(m_1, m_2, r) = \ln C + \ln m_1 + \ln m_2 - 2 \ln r$$

and it becomes a linear relation.

This process proves that simple mapping of the sample data in the input space into another space can simplify the task. The sample data in the input space cannot

be separated by a linear function, but it is possible in the new mapping space. This approach usually involves finding the minimal feature set that contains the necessary information about the original property, which is called dimensionality reduction:

$$x = (x_1, \dots, x_n) \mapsto \phi(x) = (\phi_1(x), \dots, \phi_d(x)) \text{ } d < n .$$

It is useful for both computation and generalisation [10].

We map the sample data in LML to \mathbb{F}_q . Let \mathbb{F}_q be a finite field of q elements, q be a prime power, and n be a nonnegative integer. Make

$$\mathbb{F}_q^n = \{(x_1, x_2, \dots, x_n) | x_i \in \mathbb{F}_q, \quad i = 1, 2, \dots, n\} .$$

The element (x_1, x_2, \dots, x_n) in \mathbb{F}_q is called the n -dimensional line vector on \mathbb{F}_q . Addition and scalar multiplication of the n -dimensional row vectors are defined as follows:

$$(x_1, x_2, \dots, x_n) + (y_1, y_2, \dots, y_n) = (x_1 + y_1, x_2 + y_2, \dots, x_n + y_n) \\ x(x_1, x_2, \dots, x_n) = (xx_1, xx_2, \dots, xx_n), x \in \mathbb{F}_q$$

\mathbb{F}_q^n is the n -dimensional vector space on \mathbb{F}_q , called the n -dimensional row vector space.

Proposition 2.3. *Let \mathbb{F}_q be a q finite field, where q is a prime power, both n and m are nonnegative integers, and \mathbb{F}_q^n is the n -dimensional vector row space on \mathbb{F}_q . Then, the number of m -dimensional subspaces in \mathbb{F}_q^n is exactly the same*

$$\frac{(q^n - 1)(q^{n-1} - 1) \dots (q^{n-m+1} - 1)}{(q^m - 1)(q^{m-1} - 1) \dots (q - 1)} . \quad (2.5)$$

Proof. When $m = 0$, \mathbb{F}_q^n has a unique zero-dimensional subspace, which consists of the zero vector $0 = (0, 0, \dots, 0)$. At this time, (2.5) is equal to 1. Hence, the proposition is established at $m = 0$.

Now, hypothesise that $m > 0$. When $n < m$, \mathbb{F}_q^n has no m -dimensional subspace, i.e., the number of m -dimensional subspaces of \mathbb{F}_q^n is equal to zero. However,

$$(1 - q^n)(1 - q^{n-1}) \dots (1 - q^{n-(m-1)}) \\ = (1 - q^n)(1 - q^{n-1}) \dots (1 - q)(1 - q^0)(1 - q^{-1}) \dots (1 - q^{n-(m-1)}) = 0 .$$

Thus, Proposition 2.3 is also established at $n < m$.

Finally, we investigate the situation of $n \geq m > 0$. Let V be any m -dimensional subspace of \mathbb{F}_q^n and v_1, v_2, \dots, v_m be the base of V . Here, v_1 can be any nonzero vector in \mathbb{F}_q^n , and \mathbb{F}_q^n has a total of q^n vectors; thus, v_1 has $q^n - 1$ possible choices. When v_1 is selected, v_2 can be any vector that is independent of v_1 in \mathbb{F}_q^n , and there is a total of q vectors that are linearly related to v_1 ; hence, there are $q^n - q$ possible choices for. When v_1, v_2, \dots, v_{m-1} is selected, v_m can be any vector independent of v_1, v_2, \dots, v_{m-1} . By the selection of $v_i (i = 1, 2, \dots, m - 1)$, we can see that v_1, v_2, \dots, v_{m-1} is linearly independent. Then, the vector associated with v_1, v_2, \dots, v_{m-1} has a total of

$(q^n - 1)(q^n - q) \cdots (q^n - q^{m-1})$ possible choices. However, a definite m -dimensional subspace can have different bases.

According to the above reasoning, we can see that an m -dimensional subspace has $(q^n - 1)(q^n - q) \cdots (q^n - q^{m-1})$ different bases.

Thus, the number of m -dimensional subspaces of \mathbb{F}_q^n is equal to

$$\frac{(q^n - 1)(q^n - q) \cdots (q^n - q^{m-1})}{(q^m - 1)(q^m - q) \cdots (q^m - q^{m-1})} = \frac{(q^n - 1)(q^{n-1} - 1) \cdots (q^{n-(m-1)} - 1)}{(q^m - 1)(q^{m-1} - 1) \cdots (q - 1)}.$$

It is proven that the number of m -dimensional subspaces is equivalent to the number of m orbits that can be generated (i.e., the number of operators used). In the course of learning, to distinguish each subspace, it is marked with specific real values. \square

Definition 2.15. Let m and n be nonnegative integers. Import tokens

$$\begin{bmatrix} n \\ m \end{bmatrix}_q = \frac{(q^n - 1)(q^{n-1} - 1) \cdots (q^{n-(m-1)} - 1)}{(q^m - 1)(q^{m-1} - 1) \cdots (q - 1)}, \text{ if } m > 0,$$

and

$$\begin{bmatrix} n \\ 0 \end{bmatrix}_q = 1$$

and call them the Gaussian coefficients.

Corollary 2.1. For a nonnegative integer k , there is an element a_k in R that corresponds to it, which defines a function a on a nonnegative integer set $R_0 = R \cup \{0\}$, i.e.,

$$a: \mathbb{N}_0 \rightarrow \mathbf{R}, k \mapsto a_k$$

For nonnegative integers n , make

$$b_n = \sum_{k=0}^n \begin{bmatrix} n \\ k \end{bmatrix}_q a_k. \quad (2.6)$$

Then,

$$a_n = \sum_{k=0}^n (-1)^{n-k} q^{\binom{n-k}{2}} \begin{bmatrix} n \\ k \end{bmatrix}_q b_k \quad (2.7)$$

Conversely, suppose that b is a function defined on \mathbb{N}_0 and valued in R :

$$b: \mathbb{N}_0 \rightarrow \mathbf{R}, k \mapsto b_k$$

For any nonnegative integer n , by (2.7), to define a_n , (2.6) holds.

Proof. Let V be the infinite-dimensional vector space on \mathbb{F}_q . For any n -dimensional subspace W of V , the elements a_n in R are defined to correspond to them. This defines a function defined on $L_f(V)$ and valued in R .

$$a: L_f(V) \rightarrow R, W \mapsto a(W) = a_{\dim W}$$

For any $W \in L_f(V)$, set $\dim W = n$. Let U be the subspace of W , and specify

$$b(W) = \sum_{U \leq W} a(U) \quad (2.8)$$

Obviously, if $\dim W = \dim W'$, then $b(W) = b(W')$. Make $b_{\dim W} = b(W)$. Since the number of k -dimensional subspaces in W is $\begin{bmatrix} n \\ k \end{bmatrix}_q$, then (2.8) is equivalent to (2.6).

From (2.8) and according to Proposition 2.2,

$$a(W) = \sum_{U \leq W} b(U) \mu(U, W) \quad (2.9)$$

By definition, $a(W) = a_{\dim W}$, $b(U) = b_{\dim W}$. When $\dim W = n$, $\dim U = k$, we have $\mu(U, W) = (-1)^{n-k} q^{\binom{n-k}{2}}$. Therefore, (2.7) can be derived from (2.9).

By contrast, according to

$$b: L_f(V) \rightarrow R, W \mapsto b(W) = b_{\dim W},$$

define function b defined on $L_f(V)$ and valued in R . For any $W \in L_f(V)$, we define $a(W)$ by (2.9), where U is the subspace of W . When $\dim W = \dim W'$, we have $a(W) = a(W')$. Let $\dim W = n$, $\dim U = k$, $0 \leq k \leq n$. Then, (2.9) is equivalent to (2.7), and according to Proposition 2.2, (2.8) can be derived from (2.9) and (2.6) can be derived from (2.8). \square

Proposition 2.4. *Let x be an undetermined element and n be a nonnegative integer. Then,*

$$\prod_{i=0}^{n-1} (x - q^i) = \sum_{m=0}^n (-1)^{n-m} q^{\binom{n-m}{2}} \begin{bmatrix} n \\ m \end{bmatrix}_q x^m \quad (2.10)$$

Definition 2.16. Let q be a prime power, n be a nonnegative integer, and x be an indeterminate form. The polynomials

$$g_n(x) = (x - 1)(x - q) \dots (x - q^{n-1}), \text{ if } n \geq 1,$$

and

$$g_0(x) = 1$$

are called Gaussian polynomials. Then,

$$g_n(x) = \sum_{m=0}^n (-1)^{n-m} q^{\binom{n-m}{2}} \begin{bmatrix} n \\ m \end{bmatrix}_q x^m$$

is based on the Gaussian inversion formula.

Proposition 2.5. Let b be a prime power, n be a nonnegative integer, and x be an undetermined element. Then,

$$x^n = \sum_{k=0}^n \begin{bmatrix} n \\ k \end{bmatrix}_q g_k(x) .$$

Definition 2.17. Let P be a partially ordered set containing the smallest element 0, and for $a \in P$, if $l(0, a)$ exists, i.e., 0, the a chain can be refined into a maximal chain with all 0. The maximum chain a has the same finite length, and it is called the rank of a , denoted by $r(a)$. If any of $x \in P$ has the rank $r(x)$, P is said to have the rank function $r: P \rightarrow N_0$, where N_0 is a collection of all nonnegative integers. We sometimes say that P is a rank function of r .

It is easy to understand that the set P containing several samples can be represented in terms of duality according to the dual hypothesis whereby each instance to be learned can be expressed in terms of a particular value. Here, the rank function can also be seen as the maximum possible length of the orbit limit, orbit length $\leq r(x)$.

Proposition 2.6. Let P be a partially ordered set containing the smallest element 0 and assume that for any $a, b \in P$ and $a < b$, a, b chains can be refined into a maximal chain. If P satisfies the JD condition, then there exists the rank function $r: P \rightarrow N_0$ on P . Then,

- (1) $r(0) = 0$,
- (2) If $a < \cdot b$, then $r(b) = r(a) + 1$.

Conversely, if there exists a function r defined on P and valued in N_0 , and satisfies (1), (2), then P satisfies the JD condition and r is the rank function on P .

Proof. Suppose that P satisfies the JD condition. For any $a \in P$, because the 0, a chain can be refined into a great chain, and their lengths are equal; hence, $l(0, a)$ must exist. According to the definition of 2.16, $r(a) = l(0, a)$, and $r: P \rightarrow N_0$ is the rank function on P . Obviously, (1) is established. Set up $a < \cdot b$. Assume that $r(a) = n$ and

$$0 < \cdot a_1 < \cdot a_2 < \cdots < \cdot a_n = a$$

is the 0, a great chain. Then,

$$0 < \cdot a_1 < \cdot a_2 < \cdots < \cdot a_n < \cdot b$$

is a 0, b great chain, $r(b) = n + 1 = r(a) + 1$, i.e., (2) holds.

Now, let $r: P \rightarrow N_0$ be a function that satisfies (1), (2). Let $a, b \in P$ and $a < b$. Assume that

$$a = a_0 < \cdot a_1 < \cdot a_2 < \cdots < \cdot a_n = b ,$$

$$a = a'_0 < \cdot a'_1 < \cdot a'_2 < \cdots < \cdot a'_m < \cdot b$$

are two a, b great chains. According to (2),

$$r(b) = r(a_n) = r(a_{n-1}) + 1 = \cdots = r(a_0) + n = r(a) + n ,$$

$$r(b) = r(a'_m) = r(a'_{m-1}) + 1 = \cdots = r(a'_0) + m = r(a) + m .$$

Therefore, $n = m$. This proves that P satisfies the JD condition. The rank function of P is denoted by r' . For any $a \in P$, let

$$0 < \cdot a_1 < \cdot a_2 < \cdots < \cdot a_n = a$$

be a 0, a great chain. According to definition 2.16, $r'(a) = n$. Because r satisfies (1) and (2),

$$r(a) = r(a_n) = r(a_{n-1}) + 1 = \cdots = r(0) + n = n$$

Then, $r = r'$, i.e., r is the rank function of P .

The sample data set P is expressed in the form of a matrix. The calculation of the rank function, i.e., the calculation of the number of samples useful for learning or a given set of original samples after a certain mapping transformation, i.e., the number of members of the new sample set P' , can be obtained. \square

Definition 2.18. Let P be a finite partially ordered set with the smallest element 0 and largest element 1, and let P have the rank function r . Then,

$$X(P, x) = \sum_{a \in P} \mu(0, a) x^{r(1)-r(a)} \quad (2.11)$$

is called the characteristic polynomial on P .

Note: Here, the characteristic polynomial can be understood as a polynomial representation of the sample in the learning subspace.

Proposition 2.7. Let V be the n -dimensional vector space on domain \mathbb{F}_q . Let $L(V)$ be the set of all the subspaces in V . For $U, W \in L(V)$, if $U \supset W$, specify $U \geq W$. Then, for the prescribed partial order relationship \geq , we have

$$X(L(V), x) = \prod_{i=0}^{n-1} (x - q^i) = g_n(x).$$

2.1.5 Lattices in LML

As described in Section 1.1, the learning subspace (specific learning problem) can be described as a directed graph or tree containing several nodes. It is a lattice structure that has a path from the initial state to the target state (the branch of the tree that starts at the root node to the leaf node). The node relationship on the path cannot be arbitrarily exchanged. It is embodied in the specific problem and can be understood as the sample to be learned (classification) in the node where the specified operator is the best, i.e., conducive to obtaining the optimal orbit. To effectively generate the target orbit, the idea of introducing the grid is as follows.

Definition 2.19. The sample data in the LML is ordered as a lattice if any two elements in L have upper and lower bounds. The upper and lower bounds of the elements a and

b in L are denoted by $a \vee b$ and $a \wedge b$, respectively.

$$a \vee b = \sup \{a, b\}, a \wedge b = \inf \{a, b\}.$$

$a \vee b$ is read as a and b , whereas $a \wedge b$ is read as a cross b .

When the lattice L contains a finite number of elements, it is called a finite lattice.

It is easy to see that \vee and \wedge are two algebraic operations of L , and for any of the elements a, b, c in L , the following properties are true:

L1 $a \vee b = b \vee a, a \wedge b = b \wedge a$ (Exchange law);

L2 $(a \vee b) \vee c = a \vee (b \vee c), (a \wedge b) \wedge c = a \wedge (b \wedge c)$ (Combined law);

L3 $a \vee a = a, a \wedge a = a$ (Idempotent law);

L4 $(a \vee b) \wedge a = a, (a \wedge b) \vee a = a$ (Absorption law).

Definition 2.20. Let L be a grid and S be a nonempty set of L . If S is closed for \wedge and \vee of L , then S is called a sublattice of L . Using the example given in 1.2, $\{m_{321}, m_{32}, m_3\}$ is a lattice, and the supremum and infimum are m_{321} and m_{32} , respectively. $\{m_{321}, m_{32}\}$ is a sublattice of $\{m_{321}, m_{32}, m_3\}$.

Definition 2.21. Let L and L' be two cells with operations $\vee, \wedge, \vee', \wedge'$, and let ϕ be a bijection of L to L' . If

$$\phi(a \vee b) = \phi(a) \vee' \phi(b), \phi(a \wedge b) = \phi(a) \wedge' \phi(b),$$

ϕ is called the L -to- L' lattice isomorphic mapping. L and L' are isomorphic, denoted by $L \simeq L'$.

Throughout the learning process, because the use of the operator (division rules) differs, paths with the same starting state will appear, which means that from a certain state, although the path is different, the final state is the same. This is just as the saying goes, ‘all roads lead to Rome’. If you can find a function ϕ by a path transformation to get another path, then the two paths are isomorphic. As for which path to choose, it depends on the cost of generating the path.

Definition 2.22. Grid L is called a semi-grid. If for all $a, b \in L$,

$$a \wedge b < \cdot a \Rightarrow b < \cdot a \vee b, \quad (2.12)$$

the lattice L is called the lower semi-modular lattice. If for all $a, b \in L$,

$$b < \cdot a \vee b \Rightarrow a \wedge b < \cdot a, \quad (2.13)$$

the lattice L is called the higher semi-modular lattice.

Definition 2.23. In the lattice L containing the minimal element 0, the element covering 0 is called the atom of L .

Definition 2.24. The lattice L containing the minimal element 0 is called the atomic lattice, and for each $a \in L \setminus \{0\}$, a is the upper bound of some atoms in L i.e., $a = \sup \{p \in L : 0 < \cdot p \leq a\}$.

Definition 2.25. The lattice L is called a geometric lattice if L is a half-mode atomic lattice without an infinite chain.

Proposition 2.8. In the geometric lattice L , for any $x \in L \setminus \{0\}$, x is the upper bound of finite atoms.

Proof. Suppose that there exists $x \in L \setminus \{0\}$, and x is not the upper bound of the finite atom. Since L is an atomic lattice, x must be the supremum of the set A of infinite atoms. Then, there is always $p_i \in A$, $i = 2, 3, \dots$. Make

$$p_i \notin p_1 \vee p_2 \vee \dots \vee p_{i-1}$$

by considering that L is a semi-modular lattice with minimal element 0. We have

$$0 < \cdot p_1 < \cdot p_1 \vee p_2 < \dots < \cdot p_1 \vee p_2 \dots \vee p_{i-1} < \cdot p_1 \vee p_2 \dots \vee p_{i-1} \vee p_i < \dots$$

which conflicts with L not containing infinite chains.

Considering face recognition as example, as in Section 1.2, we characterise the local features of the eye, nose, and mouth as two partial order structures from different angles of view. With a rectangular area to represent these local features, they cannot be exchanged for the relationship between the formation of a lattice structure. Once the key for the graphics is determined, their non-exchangeable precedence relationship forms a lattice structure. Once the key points of the graph are determined, to find additional features from a local feature, we need to consider the positional relationship between the two. If the neighbourhood of the mouth and nose is connected, we can determine the location of the other part by calculating their intersection and connection. \square

2.2 LML subspace orbit generating lattice learning algorithm

Conclusion: According to the basic concept of a given partially ordered set, i.e., lattice, the LML subspace orbit generation lattice theory is formed. On this basis, this section presents the breadth-first, depth-first, and heuristic learning algorithms for orbit generation and some validation examples.

2.2.1 LML subspace orbit generation lattice

The formation of the orbit can be considered in two directions: (1) from the given set of samples gradually ‘to’ the objective function to generate the orbit; (2) from the objective function back to the sample set can also generate the desired orbit. It is proved that the partial order of the subspace can be found by the relation of inclusion and anti-inclusion.

Let the sample data be mapped to the new data set \mathbb{F}_q , i.e., a finite field of q elements, where q is a prime power. Let \mathbb{F}_q^n be the n -dimensional row vector space on \mathbb{F}_q . In the set $L(\mathbb{F}_q^n)$ of all subspaces of \mathbb{F}_q^n , the partial order \geq of the subspace is found by the inclusion relation, referred to as finding the partial order \geq of $L(\mathbb{F}_q^n)$ by the inclusion relation. Then, $L(\mathbb{F}_q^n)$ makes a partially ordered set for the partial order \geq sought, and it is denoted by $L_O(\mathbb{F}_q^n)$. In the partially ordered set, there are subspaces X and Y for \mathbb{F}_q^n :

$$X \vee Y = \cap \{Z \in L(\mathbb{F}_q^n) | X \cup Y \subset Z\}, X \wedge Y = X \cap Y.$$

Hence, $L_O(\mathbb{F}_q^n)$ is a lattice, O is its minimal element, the 1-dimensional subspace is its atom, and the dimension of the subspace is its rank function.

In $L(\mathbb{F}_q^n)$, if the inverse order is used to find the partial order \geq of the subspace, $L(\mathbb{F}_q^n)$ is still the partially ordered set, denoted as $L_R(\mathbb{F}_q^n)$. Since $\sup \{X, Y\}$ and $\inf \{X, Y\}$ in $L_R(\mathbb{F}_q^n)$ are exactly the same as $\inf \{X, Y\}$ and $\sup \{X, Y\}$ in the partially ordered set $L_O(\mathbb{F}_q^n)$, hence, in $L_R(\mathbb{F}_q^n)$, we have

$$X \vee Y = X \cap Y, X \wedge Y = \cap \{Z \in L_R(\mathbb{F}_q^n) | X \cup Y \subset Z\}$$

Thus, $L_R(\mathbb{F}_q^n)$ is also a grid.

So there is:

Proposition 2.9. Mapping

$$\phi: L_O(\mathbb{F}_q^n) \rightarrow L_R(\mathbb{F}_q^n), P \mapsto P^\perp$$

is an isomorphic mapping. The lattices $L_O(\mathbb{F}_q^n)$ and $L_R(\mathbb{F}_q^n)$ are lattice isomorphic.

The orbit from the sample data set ‘towards’ the target result is formed in isomorphism with the target result, i.e., it reverses back to the trajectory of the sample set.

From the sample data set ‘to’ the target results of the orbit and the target results back to the sample set of the orbital isomorphism.

Proposition 2.10. The characteristic polynomial of lattice $L_R(\mathbb{F}_q^n)$ is

$$X(L_R(\mathbb{F}_q^n), x) = \prod_{i=0}^{n-1} (x - q^i) = g_n(x)$$

where $g_n(x)$ is a Gaussian polynomial.

Let A be a set of several m -dimensional sub-spaces ($1 \leq m \leq n - 1$) in \mathbb{F}_q^n , and let $L(A)$ be a set of crosses of subspaces of A . Conventionally, \mathbb{F}_q^n is the intersection of 0 m -dimensional subspaces, i.e., $\mathbb{F}_q^n \in L(A)$. Let $L(A)$ be a collection generated by A . If we find the partial order \geq of $L(A)$ by the inclusion relation, i.e., for

$$X, Y \in L(A), X \geq Y \Leftrightarrow X \supset Y$$

then $L(A)$ is a partially ordered set, denoted by $L_R(A)$, and $\cap_{X \in A} X$ and \mathbb{F}_q^n are its smallest and the largest elements, respectively. If we find the partial order \geq of $L(A)$ by the anti-inclusion relation, i.e., for

$$X, Y \in L(A), X \geq Y \Leftrightarrow Y \supset X$$

then $L(A)$ is also a partially ordered set, denoted by $L_R(A)$, \mathbb{F}_q^n is its smallest element, and $\cap_{X \in A} X$ is its largest element.

Theorem 2.3. $L_0(A) (L_R(A))$ is a finite lattice, and $\cap_{X \in A} X$ and \mathbb{F}_q^n are the smallest (largest) and largest (smallest) elements of $L_0(A) (L_R(A))$, respectively.

Proof. We consider two cases of finding the partial orders of $L(A)$ in terms of inclusion and anti-inclusion relations. We prove that the method is the same, and we only prove the case of anti-inclusion. Obviously, $|L(A)| < \infty$, and $\cap_{X \in A} X$ and \mathbb{F}_q^n are the largest and smallest elements in $L_R(A)$, respectively. Now, we prove that it is a lattice. Since $L_R(A)$ is a partially ordered set, it is only necessary to prove that in the partially ordered set, $L(A)$ is closed for intersection and union. Take $A_1, A_2 \in L_R(A)$ arbitrarily. Both A_1 and A_2 are the intersection of the m -dimensional subspaces in A , which are known as $A_1 \vee A_2 = A_1 \cap A_2 \in L_R(A)$ by $A_1 \cup A_2 \subset \mathbb{F}_q^n$, and $\mathbb{F}_q^n \in L_R(A)$. Further, the intersection of subspaces containing $A_1 \cup A_2$ in $L_R(A)$ also contains $A_1 \cup A_2$. Hence, there is only the smallest subspace in $L(A)$ that contains $A_1 \cup A_2$. Therefore,

$$A_1 \wedge A_2 = \cap\{Z \in L_R(A) | A_1 \cup A_2 \subset Z\} \in L_R(A)$$

□

Definition 2.26. Finding the partial order $L(A)$ in terms of the containment relation or anti-inclusion relation, the resulting lattice $L_0(A)$ or $L_R(A)$ is called by the inclusion or anti-inclusion of the lattice generated by A , referred to as a lattice generated by A .

Let G_n be one of the n -level typical groups on the sample data set in LML, i.e., $G_n = GL_n(\mathbb{F}_q)$, $Sp_{2v}(\mathbb{F}_q)$ (where $n = 2v$ is even), $U_n(\mathbb{F}_q)$ (where q is the square), $O_{2v+\delta}(\mathbb{F}_q)$ (where $n = 2v + \delta$, $\delta = 0, 1$ or 2), $P_{S_{2v+\delta}}(\mathbb{F}_q)$ (where $n = 2v + \delta$, $\delta = 1, 2$, and q is even) [23]. The role of G_n in \mathbb{F}_q^n is as follows:

$$\mathbb{F}_q^n \times G_n \rightarrow \mathbb{F}_q^n, ((x_1, x_2, \dots, x_n), T) \mapsto (x_1, x_2, \dots, x_n) T$$

This function derives the role of G_n in the subspace set of \mathbb{F}_q^n , i.e., if P is an m -dimensional subspace of \mathbb{F}_q^n , then P can be transformed into the m -dimensional subspace PT by $T \in G_n$. Hence, the subspace set of \mathbb{F}_q^n is divided into some orbits by the action of G_n . Obviously, $\{\{0\}\}$ and $\{\mathbb{F}_q^n\}$ are two orbits, which is an ordinary situation. Let M be a non-trivial orbit under the action of G_n . Then, M is a set of finite m -dimensional ($0 < m < n$) subspaces in \mathbb{F}_q^n .

Definition 2.27. Let M be a subspace orbit under the action of a given typical group G_n , and the set generated by M is denoted by $L(M)$, i.e., $L(M)$ is a set of M subspaces consisting of intersection, and \mathbb{F}_q^n is the intersection of zero subspace sets in M . If

the subspace is contained (anti-inclusive) to find its partial order, the resulting grid is denoted as $L_O(M)$ ($L_R(M)$). The grid $L_O(M)$ or $L_R(M)$ is called the grid generated by the orbit M .

Theorem 2.4. $L_O(M)$ ($L_R(M)$) is a finite lattice, and $\cap_{X \in M} X$ and \mathbb{F}_q^n are the smallest (largest) and largest (smallest) elements of $L_O(M)$ ($L_R(M)$), respectively.

In LML, the maximum or minimum element of the objective function is determined using the orbit lattices formed by $L(M)$.

Theorem 2.5. Lattice $L_R(M)$ is a finite atomic lattice, and M is its atomic set.

2.2.2 Orbit generation learning algorithm for LML subspace [18–22]

2.2.2.1 LML subspace orbit generation breadth-first and depth-first learning algorithm

For orbit generation, we can proceed from a given weight (known first-level weight) using each existing weight for a simple calculation to create a new weight, i.e., the list of all the weights in the second-level orbit. An ordinary weight corresponds to a list of elements in a sample of length 2. The process is repeated and the weight list of any level can be obtained using the weights and simple calculations of the next lower layer. Thus, in the case where the given first layer weights are ordered, the entire orbit, or a diagram of the entire learning subspace, can be generated in this way. However, there are two issues to consider: the storage space required for computing orbits and the number of repeated computations required to generate the next level of the list using this method. Because the simple lookup for the repeatedly created list of newly created orbit elements is slow and inefficient, a method is proposed for deciding whether to add an orbit element to a list of weights at the next higher level. The higher level of the orbit requires only local data, i.e., a simple calculation of the new weights and creation of the weights. This solves the problem of repeated computation and provides a new method for generating orbits. Instead of generating a hierarchy at a time, find the ownership value of that layer. We can consider the orbit as a tree, and according to the ‘depth of priority’ for calculation, deeper levels of the weight of the orbit can be reached as far as possible. A weight is stored on the basis of certain specifications before tracing back to the uncomputed ‘branch’. The following method (depth-first orbit generation) is easy to handle with a stack, which has a priori range of members, i.e., the number of samples in the sample set, which helps to reduce the storage space. Because it is difficult to store a very large orbit in a computer at once, this new approach can be used to generate orbital nodes because of its small space requirements.

Suppose that L_k represents the k -level orbit, i.e., the weight set of the k -th layer. The sample set is the disjoint union of L_0, \dots, L_N , where N is the maximum level that the orbit may reach. Note that N is defined by the number of samples, i.e., the

maximum length of the sample set. The level L_0 consists only of μ , $L_0 = \mu$. To generate level L_1 , use a simple calculation to calculate μ . If $\xi = (x_1, \dots, x_l) \in P$, then

$$\text{level}(\sigma_i(\xi)) = \begin{cases} \text{level}(\xi) + 1 & \text{if } x_i > 0 \\ \text{level}(\xi) & \text{if } x_i = 0 \\ \text{level}(\xi) - 1 & \text{if } x_i < 0. \end{cases} \quad (2.14)$$

The whole process is calculated from the level L_k to the generation level L_{k+1} by an induction method. From (2.14), we know that

$$L_{k+1} = \{\sigma_i(v) \mid i = 1, \dots, l, v = (n_1, \dots, n_l) \in L_k, n_i > 0\} \quad (2.15)$$

where

$$\sigma_i(\mu) = \mu - \langle \mu, \alpha_i \rangle \alpha_i = (m_1 - m_i c_{i1}, \dots, m_l - m_i c_{il}) \quad (2.16)$$

and (c_{i1}, \dots, c_{il}) is a vector representation of α_i . From one aspect, the calculation formula indicates that it is easy to determine whether a weight in the hierarchical L_k is easily computed to obtain the weights in the hierarchical L_{k+1} . However, the formula hides the real problem of computing orbits, i.e., how to deal with repeated computations at the computational hierarchy of L_{k+1} . Because such a situation arises, when $v_1, v_2 \in L_k$ has $\sigma_i(v_1) = \sigma_j(v_2)$, and the hierarchy L_k can be very large, it is impractical to conduct simple searches for the levels created by the weights computed from L_k to L_{k+1} . Of course, using fast tables can speed up the search process, but it is still invalid for collections containing more sample data. Based on the advantage of weights as natural sequences, it is easy to avoid repeated computations. Now, let us prove that deciding whether to store $\sigma_i(v)$ depends solely on its coordinates and index i .

Theorem 2.6. [11–14] Let L_k be the k -th layer of the orbit with the dominating weight μ . Thus, for each $\xi = (x_1, \dots, x_l) \in L_{k+1}$, there is a unique $v \in L_k$ and a unique simple calculation σ_i , i.e., $\sigma_i(v) = \xi$, and for $j > i$, $x_j \geq 0$. In particular, the i -th coordinates of v are nonnegative if and only if the coordinates of $\sigma_i(v)$ after the i -th coordinates are nonnegative. The next level L_{k+1} can be obtained from the weight v in L_k by adding $\sigma_i(v)$ to $\sigma_i(v)$ without repeating the creation.

$$L_{k+1} = \{\sigma_i(v) = (x_1, \dots, x_l) \mid i = 1, \dots, l, v = (n_1, \dots, n_l) \in L_k, n_i > 0, x_j \geq 0, j > i\} \quad (2.17)$$

Proof. Let $\xi = (x_1, \dots, x_l)$ be any element of L_{k+1} and i be the index of the last negative coordinate in ξ , i.e., $x_i < 0$, for $j > i$, $x_j \geq 0$. Assume that $v =: \sigma_i(\xi) = (n_1, \dots, n_l)$. Therefore, $\sigma_i(v) = \xi$. By (2.15), $n_i = x_i - x_i c_{ii} = -x_i > 0$, and by (2.14), $v \in L_k$. This proves (2.17). Now, suppose that for some $\omega \in L_k$ rather than v , $\sigma_m(\omega) = \xi$. By (2.14), the m -th coordinate of ω must be positive; thus, $x_m < 0$. Hence, after the choice of i (because $\omega \neq v$, $i \neq m$), $m < i$. Because $x_i < 0$, it is proved that when $\sigma_i(v)$ with $v \in L_k$ and for $j > i$, $x_j \geq 0$, the expression of ξ is unique.

Note: In the specific learning algorithm, the two issues can be directly expressed by positive and negative coordinates, and there are many types of problems with integer coordinates as well as regression problems with real coordinates. Of course, regardless of the type of complex problem, it can be classified into one of the two categories to be dealt with.

The above theorem provides an effective way to calculate another orbit from one orbit, and it should be noted that the new generated weight $\sigma_i(v)$ can be determined only by referring to the coordinates and index i of $\sigma_i(v)$. In other words, there is no need to remember (store) too much weight, which is quite meaningful for reducing the number of weights at the highest level.

The algorithm is described as follows:

Branch(v):

- (1) Set $i_0 = 0$.
- (2) Suppose that i is the first exponent, with $i > i_0$ and $n_i > 0$. If there is no such index i , the algorithm is terminated.
- (3) Calculate the weight $\xi = \sigma_i(v) = (x_1, \dots, x_l)$.
- (4) If $x_{i+1}, \dots, x_l \geq 0$, then output ξ and execute *Branch*(ξ).
- (5) Set $i_0 = i$ and go to step 2.
- (6) *Branch*(μ) will give all the weight points in the tree. The only operation in *Branch* is to compare the integer and the calculation in the third step. The shortest path in the tree (from ξ to root node μ) is found to indicate that the hierarchy of $\xi = (x_1, \dots, x_l)$ can be quickly calculated.

Reflection(ξ):

- (1) Let i be the index $x_i < 0$ of the first negative coordinate. If there is no such index, the algorithm is terminated.
- (2) Calculate the weight $v = \sigma_i(\xi)$.
- (3) Output index i .
- (4) Set $\xi = v$ and go to the first step of the algorithm.

Description: The number given by *Reflection*(ξ) is clearly the level of ξ , since the level is reduced by one in the second step. The algorithm can control the maximum number of computational levels, since an orbit needs only to be calculated to the $N/2$ layer, where N is the number of layers in the orbit. The rest of the orbit can be calculated using the equation $\omega_0(v) = (-n_{s(1)}, \dots, -n_{s(l)})$, assuming that $v = (n_1, \dots, n_l)$ has level k . □

Algorithm 2.1: Orbit generation breadth-first algorithm

Orbit–Breadth–First

Data structure: levelcount: An array of integers counted for each layer (initialised to 0)

current-level, next-level: Link list of weight nodes

Each weight node contains: $\omega = (\omega_1, \dots, \omega_l)$: Weight vector

next: A pointer to the next weight point

Input: $m = (m_1, \dots, m_l)$: Dominance vector (all $m_i > 0$)

D: Dynkin matrix

l: rank of G

Output: Weight vector list $\omega = (\omega_1, \dots, \omega_l)$ and three related integers

The first integer indicates the position of ω in the list

The second and third integers indicate the weight vector and the calculation of ω

Process: output m; // initial state, i.e., the original sample set that has not been studied

Insert m into the header of the current-level list;

Make $level = 0$, $levelcount[level] = 1$;

Make $from = 1$; //Position of the patriarchal values

Make $to = 1$; //Position of the new weight value is also the total number of weights that are calculated

while ($levelcount[level] > 0$ and $level < max level$)

{ $level++$;

Set $from = to$; //Count down the weights in the current-level list

for (Each weight ω in the current-level list)

{ for ($i = 1$ to l)

if ($\omega_i > 0$)

{ Calculate σ_i to ω via v;

if ($v_j \geq 0$ to $i < j \leq l$)

{ //Store v? }

Insert v into the header of the next-level list;

$levelcount[level]++$;

Output v, to, from, i;

}

}

$from--$; // Follow the ‘first in, first out’ principle in the current list

}

Release the previous list current-level;

Make $current - level = next - level$;

Make $next - level =$ empty table;

Description: The *current-level* is used to store the weights that have just been calculated (initially, the *current-level* list only includes the control weights μ). The *next-level* stores the new weights generated by the *current-level*. When the weights in the *current-level* are all calculated, the *next-level* list is assigned to it and the original list is released.

Algorithm 2.2: Orbit generation depth-first algorithm

Orbit-depth-first

Data structure: *levelcount*: An integer array that counts for each layer

weight-structure stack

Each *w weight-structure* contains:

$\omega = (\omega_1, \dots, \omega_l)$: Weight vector

level: Rank of ω

from: An integer that labels the parent value

i: The next index to be tested

Input: $m = (m_1, \dots, m_l)$: Control weight

D: Dynkin matrix

l: rank of G

maxlevel: Calculate the maximum level of the track

Output: Weight vector list $\omega = (\omega_1, \dots, \omega_l)$ and four related integers

The first integer indicates the position of ω in the list

The second integer is the hierarchy of ω

The third and fourth integers indicate the weight vector and the calculation of ω

Process: Make *level* = 0, *levelcount*[*level*] = 1, *i* = 1;

Make *from* = 1; // Location of paternity values

Make *to* = 1; //Position of the new weight, i.e., the total number of weights computed

Output *m*; // Initial state, i.e., original collection of samples that have not been learned

Put *m*, *level*, *from*, *i* into the stack;

while (The stack is nonempty)

{ Out ω , *level*, *from*, *i* of stack;

 while (*i* ≤ *l*)

 if ($\omega_i > 0$)

 { if ($v_j \geq 0$ to $i < j \leq l$)

 { //Store v ?

 If ($i = 1 \leq j$)

 Out ω , *level*, *from*, *i*+1 of stack; // Then return to ω

level++;

levelcount[*level*]++;

 Output v , *level*, *from*, *i*, *to*;

 if (*level* ≥ *max level*) jump out while loop;

 Copy v to ω ; // Continue with the track of $\omega = v$

 Let *i* be equal to the first element in D_i ;

 }

 Otherwise *i*++;

 }

 Otherwise *i*++;

}

Note: The weights mentioned above are treated as sample data or related parameter weights generated in the learning algorithm.

Example analysis

Eight data sets were downloaded from the UCI's dedicated data set (as shown in Table 2.2), which include the number of samples. The number of samples and the number of attribute values of these data sets are different. In the process of testing, the examples of incomplete attribute values are removed, and continuous values are discretised. Here, it is divided into three discrete states.

In this example, eight data sets are tested separately. The initial sample set is taken as the input control weight, and the corresponding Dynkin matrix is established. The relationship between ω_i and 0 is used to determine the generation of the next level node, where $\omega_i > 0$ indicates that the sample set has not yet completed classification, i.e., there are multiple categories. The above-mentioned orbit generation breadth-first and depth-first learning algorithms are performed according to the control weights and the corresponding exponential values in the Dynkin matrix. We have pointed out in the analysis of the algorithm that the depth-first algorithm can reduce the storage costs and improve the time efficiency to a certain extent. The OBF (orbital breath first) and ODF (orbital depth first) algorithms are tested and compared, and the test results are summarised in Table 2.3.

Tab. 2.2: Test data set.

Name	Attribution	Sort number	Sample number
Soybean-Large	35	19	562
Pima	8	2	768
Balance-Scale	4	3	625
Baloon	4	2	20
Hayes	4	3	132
Housing	13	5	506
Ionosphere	34	2	351
Lenses	4	4	24

Tab. 2.3: Comparison of test results.

Name	OBF	ODF	OBF Time (second)	ODF Time (second)
Soybean-Large	91.10	90.93	132.23	43.54
Pima	77.08	75.00	9.00	3.68
Balance-Scale	73.76	71.08	4.72	1.05
Baloon	80.26	75.00	1.12	0.23
Hayes	78.79	78.79	1.08	0.18
Housing	63.38	66.45	14.00	4.32
Ionosphere	91.73	88.75	11.98	3.21
Lenses	89.16	86.83	1.61	0.02
Mean data	80.66	79.10		

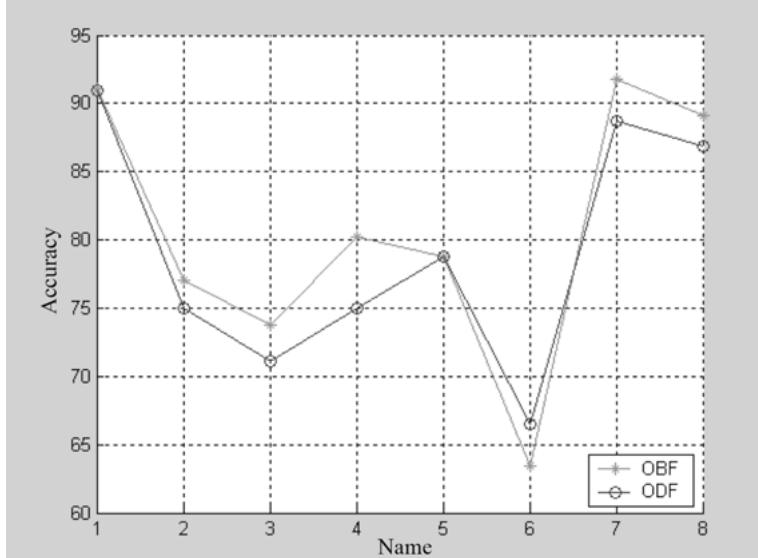


Fig. 2.2: OBF and ODF algorithm classification accuracy comparison.

OBF represents the breadth-first algorithm and ODF represents the depth-first algorithm. OBF Time and ODF Time are the execution times of the OBF and ODF algorithms, respectively. It is easy to see that the overall classification accuracy of OBF is higher than that of ODF. Due to the size and number of attributes of each test data set, the accuracy differs, and ODF has a more obvious advantage in terms of the time cost, as shown in Figure 2.2.

2.2.2.2 LML subspace orbit generation learning algorithm with heuristic information

Algorithm 2.3: LML subspace orbit generation learning algorithm with heuristic information
GENERATED_ORBITS (P , $c(x)$, G_n)

// P is the sample data set, $c(x)$ is the objective function, and G_n is the basis of the track division

- 1) The sample data set P is mapped to A, i.e., $\Phi: P \rightarrow \mathbb{F}_q^n$
// Corresponds to the specific problem using different A
- 2) Take the n -dimensional row vector on \mathbb{F}_q to form a set \mathbb{F}_q^n ,

$$\mathbb{F}_q^n = \{(x_1, x_2, \dots, x_i, \dots, x_n) | x_i \in \mathbb{F}_q, i = 1, 2, 3, \dots, n\}$$

// Use vector $(x_1, x_2, \dots, x_i, \dots, x_n)$ to represent any sample

- 3) Call algorithm 4, set T , get G_n // Specific problem in the setting of T is different, so the use of G_n is different
- 4) **for** ($i = 1; i \leq k; i + +$)

- (1) Take $T_i \in G_n$, and treat \mathbb{F}_q^n as follows:

$$((x_1, x_2, \dots, x_i, \dots, x_n), T_i) \mapsto (x_1, x_2, \dots, x_i, \dots, x_n) T_i$$

- The subspace orbit M of \mathbb{F}_q^n is obtained.
- (2) Intersection generation sets $L(M)$ of the subspaces of M according to the inclusion (inverse inclusion) relation. The ordered spanning lattice $L_O(M)$ (or $L_R(M)$) is given by the partial order on $L(M)$.
- (3) Determine the largest (or smallest) $\cap_{X \in M} X$ on $L_O(M)$ (or $L_R(M)$)
 → If successful in finding X , then get the value of $c(x)$, $F \leftarrow (T_i, c(x))$;
 → Else discard X ;
End if;
 (4) Delete the sample in the sample set \mathbb{F}_q^n , which gets the target learning result and gets the new sample set \mathbb{F}_q^n ;
End for;
 (5) For each target function $c(x)$, output set F ;
 // In dealing with the classification problems, the objective function $c(x)$ is evaluated in R for two types of problems $C = \{-1, 1\}$: for the multi-class problem $C = \{1, 2, \dots, k\}$ and for the regression problem $C \subseteq R$ [15].
-

Note: for Algorithm 2.3:

- (1) Arrange the various states and tracks of the learning problem;
 (2) Choice T ($T \in G_n$ (and rules of setting T reasonably);
 (3) Distinguish the correctness of the set of learning samples;

The correctness rate of the learning sample is $L(g)$: $L(f) = P\{f(x) = y\}$, x is the example to be studied, y is the real result of x , and f is the learning method. Usually, we want to find the learning method $f(x)$ that makes $y \cdot f(x) > 0$.

- (4) Determine the target track:

According to each time to find the largest yuan (minimum element), to obtain the minimum partial chain, i.e., the target track.

- (5) Condition of termination.

All the samples to be studied are placed in the result set (i.e., the sample set is empty) and the corresponding target track.

In the process of generating the orbit, the key step is how to determine the nodes to be extended. Different methods of determination lead to the formation of different trajectory generation strategies. If we can make full use of the inspiration information related to the learning problem when determining the track node, by estimating the importance of nodes, we can select the nodes of higher importance to expand when the orbit is divided, to obtain the optimal orbit. Algorithm 4 is used to solve the problem of setting of the divide rule T and the arrangement of the tracks.

Algorithm 2.4: Setting of the divide rule T and the arrangement of the tracks:RULE ($P, c(x), G_n$)// P is the sample set, $c(x)$ is the objective function, and G_n is the final set of partitioning rules(1) Set the property set based on the sample set P : $Attribute = \{A_1, A_2, \dots, A_i, \dots, A_n\}$ // n is the number of attributes

(2) if property values are discrete values, then

Set the property-value set for each property: $Value(A_i) = \{k_1, k_2, \dots, k_i, \dots, k_m\}$;// m is the number of attribute values

Else discrete continuous attributes, and then set the property-value set;

(3) $h = 0$; for ($j = 1; j \leq n, j + +$)for ($i = 1; i \leq m, i + +$)For the statistical categories C that k_i correspond to, calculate the number of samples in which P_{C_i} belongs to the class C and the attribute is valued as k_i , and calculate the total number of samples with the attribute value k_i .if $P_{C_i} = 1$, then { attribute $i = k_i$ }; $G_n \leftarrow T_h$; $h + +$;Remove samples of known categories from the sample set P ;

End for

End for

(4) In the remaining sample set P' statistics the P_{C_i} value of k_i and $k_{i'}$, as in steps 3 above, and get T_h ;// k_i and $k_{i'}$ come from different attributes(5) Increase the attribute value of the statistics until the sample set P is empty and return the partition rule set G_n .

In the classification problem, to reasonably choose T_i in the rule set G_n generated by Algorithm 2.4, define the heuristic information as the number (the largest number) of samples that can be distinguished from the category. For a given sample, the category is known. Starting from the starting state (not classified), according to the inspiration information for finding the target, if the sample is properly classified, the target track is found. Complete the algorithm. Otherwise, discard the path and withdraw one step. Repeat until all samples are classified correctly.

Algorithm 2.5: Planning Selection Algorithm:SELECT-RULE (G_n)(1) Select two rules from rule set G_n . Let each rule be represented by a set of characteristic attributes that may be different, $T_i = \{k_1, k_2, \dots, k_n\}$ and $T_j = \{k'_1, k'_2, \dots, k'_m\}$, and mark the selected rules, indicating that they have been compared.(2) Calculate the similarity function $simi(T_i, T_j) = 1 - dist(T_i, T_j)/max_dist$. Here, the importance and contribution of the judgment of similarity according to the different attributes are different. Assign different weights to different feature attributes according to the different characteristics of the characteristics of similarity to determine the importance and contribution, regular similarity functions with weighted feature attributes can be defined.

$$simi(T_i, T_j) = \frac{|T_i \cap T_j|}{|T_i|} = \frac{\sum_{k_i \in T_i \cap T_j} w_i}{\sum_{k_i \in T_i} w_i}$$

// According to the value of the similarity function, we can see the similarity of the rules, where $dist(T_i, T_j)$ is the distance between rule T_i and T_j , $\max - dist$ is the maximum distance in the rule set, and w_i is the characteristic attribute weight of T_i . According to the experience and the actual situation given, it is only to determine the importance of the various characteristic attributes.

- (3) Otherwise, re-select the comparison rule until the rules in the rule set are compared and end the algorithm.
-

For supervised learning, the learner enters a training set (or input) with a marker (or output value). The sample is usually given in the form of an attribute vector, so the input space is a subset of R^n . Once an input vector is given, a certain number of hypothetical function sets can be selected for the problem, where the linear function is the easiest to understand and the application is also the simplest. It can be expressed in dual form. However, complex real-world applications need to be more expressive than the linear function of the hypothesis space; they require a more expressive space than linear functions. In other words, the concept of goals cannot usually be generated by simple linear functions of given attributes. Instead, we should look for more abstract features of the data to study. Mapping the data to high-dimensional space increases the computational power of the linear learner, and the expression of the dual space makes the implicit operation of this step possible. The advantage of a dual-form representation learner is that the number of adjustable parameters does not depend on the number of input attributes in the representation. In specific learning practice, there are some specific learning problems. First, the learning algorithm may be inefficient, e.g., a local minimum situation occurs. Second, the hypothesis of output scale is often too large to be unrealistic. Third, if the number of learning examples is limited, excessive hypothetical function classes will lead to overfitting and poor generalisation. Fourth, learning algorithms are often controlled by a large number of parameters; their choice is often through the heuristic parameter adjustment process, making the learning system difficult to use and unreliable [10].

Examples and analysis [16]

Example 1: Taking population classification as an example, the training example is shown in Table 2.4.

- (1) Corresponding to the algorithm, the first step is to identify the most representative attributes of the population: hair colour, eye colour, height, and weight. Height and weight are continuous attributes; we discretise them by regional values and divide the height into three groups: $> 180.00 \text{ cm}$, $< 170.00 \text{ cm}$, and $170.00 \text{ cm} \leq \text{height} \leq 180.00 \text{ cm}$. Weight is divided into $> 100.00 \text{ kg}$, $< 60.00 \text{ kg}$, and $60.00 \text{ kg} \leq \text{weight} \leq 100.00 \text{ kg}$. Of course, in many problems, Φ can be used to discretise attributes, thus simplifying the task.
- (2) The attribute of each instance is written in the form (x_1, x_2, x_3, x_4) (there are four attributes) of the row vector, and the partial order structure of the sample set is defined by calculating the information gain of each attribute.

Tab. 2.4: Experimental Data Table.

ID	Crowd classification	Hair color	Eye color	Height(cm)	Weight(kg)
1	Caucasian	Gold	Blue	189.00	160.00
2	Caucasian	Gold	Blue	215.00	235.00
3	Caucasian	Red	Black	170.00	140.00
4	Caucasian	Gold	Black	175.00	190.00
5	Caucasian	Red	Brown	185.00	200.00
6	Caucasian	Red	Brown	155.00	100.00
7	Caucasian	Black	Black	210.00	120.00
8	Caucasian	Black	Brown	220.00	123.00
9	Caucasian	Black	Brown	215.00	118.00
10	Black people	Black	Brown	170.00	70.00
11	Black people	Black	Brown	178.00	73.00
12	Black people	Black	Brown	180.00	82.50
13	Black people	Black	Brown	175.00	95.00
14	Black people	Black	Brown	185.00	100.00
15	Black people	Gray	Brown	185.00	200.00
16	Black people	Gray	Brown	155.00	100.00
17	Black people	Gray	Brown	166.00	100.00
18	Black people	Gray	Brown	160.00	95.00
19	Black people	Gray	Brown	167.00	107.00
20	Black people	Gray	Brown	166.00	100.00
21	Asian	Black	Brown	155.00	50.00
22	Asian	Black	Brown	165.00	60.00
23	Asian	Black	Brown	168.00	60.00
24	Asian	Black	Brown	153.00	45.00
25	Asian	Black	Brown	132.00	42.50
26	Asian	Black	Brown	166.00	50.00
27	Asian	Black	Black	160.00	47.50
28	Asian	Black	Black	167.00	53.50
29	Asian	Black	Black	160.00	47.50
30	Asian	Black	Black	167.00	53.50
31	Asian	Black	Black	160.00	47.50
32	Asian	Black	Black	167.00	53.50

- (3) Set T , here by observation, statistics, and calculations, to get a reasonable G_n , Hair = ‘golden’ or ‘red’ => ‘Caucasian’, Hair = ‘grey’ => ‘Black people’, eyes = ‘black’ => ‘not Black people’ and so on. We give T as follows:

$$T_1 = \{\text{gray}, ?, ?, ?, ?\}$$

$$T_2 = \{\text{gold}, ?, ?, ?, ?\}$$

$$T_3 = \{\text{red}, ?, ?, ?, ?\}$$

$$T_4 = \{?, ?, < 170.00, ?\}$$

$$T_5 = \{?, ?, ?, > 100.00\}$$

- (4) Take $T_i \in G_n$ ($i = 1, 2, 3, 4$) in turn. Correspondingly, the ID number of 15 to 20 for $c(x) = Black\ people$, 1,2,4 for $c(x) = Caucasian$, 3,5,6 for $c(x) = Caucasian$, 21 ~ 32 for $c(x) = Asian$, 7 ~ 9 for $c(x) = Caucasian$, 10 ~ 14 for $c(x) = Black\ people$. Write down the corresponding F for each value (there are three values for $c(x)$).

- (5) output $c(x) = Caucasian$, $F = \{(T_2), (T_3), (T_1, T_2, T_3, T_4, T_5)\}$, $c(x) = Caucasian$, $c(x) = Black\ people$, $F = \{(T_1), (T_1, T_2, T_3, T_4, T_5)\}$, $c(x) = Black\ people\}$, $c(x) = Asian$, $F = \{(T_1, T_2, T_3, T_4)\}$, $c(x) = Asian\}$

Comparing the classification accuracy (error) obtained by the algorithm described in this paper (abbreviated as GO) with the results obtained by the N-Bayes and C4.5 algorithms (listed in Table 2.5), shows that the algorithm is feasible and to a certain extent, superior to N-Bayes and C4.5.

Tab. 2.5: Accuracy comparison table.

Algorithm	C4.5	N-Bayes	GO
Accuracy (%)	91.53	86.82	97.69

The target track in this example is shown in Figure 2.3:

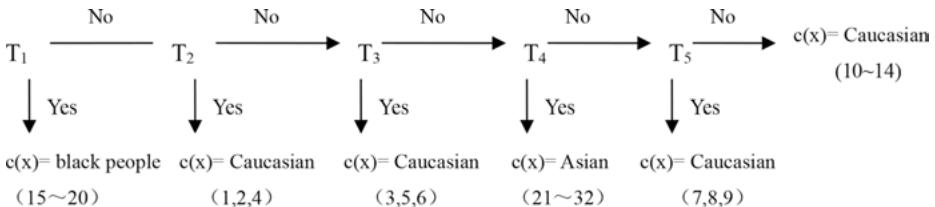


Fig. 2.3: Target orbit diagram.

Example 2: Eight data sets are collected from the UCI database (shown in Table 2.2), which does not have property defaults. These algorithms are used to classify the samples in the data set; 2/3 of the data are used for training, and the remaining 1/3 are used for testing. The classification accuracy rate of the algorithm presented in this paper (abbreviated as GO) is compared with the results obtained using the N-Bayes, C4.5, and K-nearest neighbour (K-NN) algorithms (as shown in Table 2.6), and the results are shown in Figure 2.4. It is easy to see that the algorithm presented here is superior to the other algorithms in terms of the classification accuracy. The feasibility and effectiveness of the algorithm are also explained.

Tab. 2.6: classification accuracy comparison.

Domain	N-Bayes	C4.5	K-NN	GO
Pima	75.00	96.13	97.53	98.30
Balance-Scale	58.08	85.30	84.76	89.48
Soybean-Large	90.03	62.45	77.41	88.21
Housing	60.67	57.37	77.90	84.50
Ionosphere	87.75	80.82	82.35	92.15
Hayes	78.03	87.42	75.84	95.20
Lenses	70.83	59.63	92.00	96.00
Baloon	75.00	80.01	79.86	81.65
Mean data	74.42	76.14	83.46	90.69

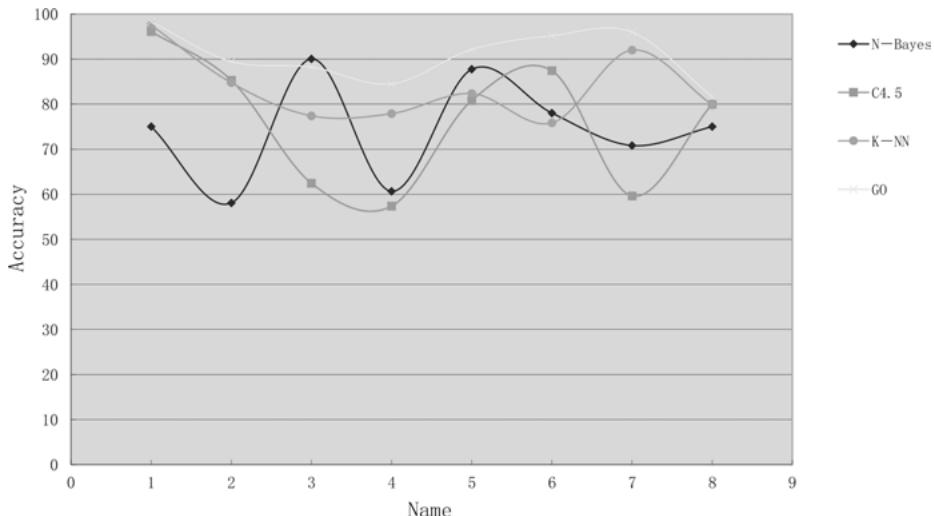


Fig. 2.4: classification accuracy comparison.

2.3 LML learning subspace orbit generation lattice learning algorithm under the action of the general linear group $\text{GL}_n(\mathbb{F}_q)$

2.3.1 Problem description

- (1) What is the effect of the inclusion relation between the lattice $L_R(M)$ generated by each orbit M for a given G_n in the learning process;
- (2) What is the character of the subspace in a given lattice $L_R(M)$, i.e., what role does the condition of a subspace of \mathbb{F}_q^n in $L_R(M)$ play in the learning process;

- (3) What is the significance of the characteristic polynomial of each lattice $L_R(M)$ in the course of learning;
- (4) If the partial relation of the set $L(M)$ is searched according to the containment relation, when $L_O(M)$ and $L_R(M)$ form geometric lattices in the learning process

The role of the above four issues in learning activities can be understood as follows:

- (1) the inclusion relation between the orbit generation lattices can be used to determine the use of the partition rule to divide the sample data set, thus generating the order of the orbital nodes;
- (2) the sample data set is divided into the transformation conditions required for the orbit;
- (3) the expression of the relationship between the data in the orbital node;
- (4) determine the geometric properties of the orbital nodes generated from the set of samples (or from the objective function), which differ according to the nature of dividing the rule sets. The conditions that the resulting orbit space meet are different. Here, we discuss these four issues with $G_n = \text{GL}_n(\mathbb{F}_q)$.

2.3.2 LML learning subspace orbit generation lattice under the action of the general linear group $\text{GL}_n\mathbb{F}_q$

Let the orbit made up of all m ($0 < m < n$)-dimensional subspaces under the action of $\text{GL}_n\mathbb{F}_q$ in \mathbb{F}_q^n be denoted as $M(m, n)$. The set generated by $M(m, n)$ is denoted as $L(m, n)$, i.e., $L(m, n)$ is a set of intersection of the subspaces of $M(m, n)$, and \mathbb{F}_q^n is the intersection of the zero subspace sets in $M(m, n)$. The partial ordering of $L(m, n)$ is specified according to the inclusion (anti-inclusion) relation of subspaces, and it is denoted by $L_O(m, n)$ ($L_R(m, n)$). From Theorem 2.2, we know that $L_O(m, n)$ ($L_R(m, n)$) is a finite lattice, and $\cap_{X \in M(m, n)} X$ and \mathbb{F}_q^n are the largest (smallest) and smallest (largest) elements of $L_O(m, n)$ ($L_R(m, n)$), respectively. By Theorem 2.3, $L_R(m, n)$ is the atomic lattice, and $M(m, n)$ is its atomic set.

Definition 2.28. Lattices $L_O(m, n)$ and $L_R(m, n)$ are called **the lattices** generated by subspace orbit $M(m, n)$ under the action of general linear group $\text{GL}_n(\mathbb{F}_q)$.

Theorem 2.7. Set $n > m \geq 0$. Then, the necessary and sufficient condition for

$$L_R(m, n) \supset L_R(m_1, n) \quad (2.18)$$

is $m \geq m_1 \geq 0$.

Proof. The adequacy of $n = 1$; (2.18) is established naturally. Set $n \geq 2$ below. Let us prove that

$$L_R(m, n) \supset L_R(m - 1, n) . \quad (2.19)$$

First, to prove it, just prove that

$$M(m-1, n) \subset L_R(m, n). \quad (2.20)$$

Let $P \in M(m-1, n)$, where P is an $m-1$ -dimensional subspace. Taking $v_1, v_2, \dots, v_{m-1}, v_m, \dots, v_n$ as a base of \mathbb{F}_q^n , where v_1, v_2, \dots, v_{m-1} is a base of P , because $m < n$, there are two different m -dimensional subspaces in \mathbb{F}_q^n

$$\binom{P}{v_m} \text{ and } \binom{P}{v_{m+1}}$$

makes

$$P = \binom{P}{v_m} \cap \binom{P}{v_{m+1}} \in L_R(m, n).$$

Therefore, (2.20) is established.

The following proves (2.18). When $m = m_1$, (2.18) is established naturally. Under $m > m_1$, by (2.19),

$$L_R(m, n) \supset L_R(m-1, n) \supset \cdots \supset L_R(m_1, n).$$

Thus, (2.18) is established.

Necessity: From $M(m_1, n) \subset L_R(m_1, n) \subset L_R(m, n)$, we can see that $L_R(m, n)$ contains an m_1 -dimensional subspace Q , and Q is the intersection of the m -dimensional subspace in $M(m, n)$. Therefore, there exists an m -dimensional subspace P in $L_R(m, n)$, which makes $P \supset Q$. Therefore, $m \geq m_1 \geq 0$. \square

Inference 2.1. [8] Set $n > m \geq 0$. Then,

$$\{0\} \in L_R(m, n).$$

Thus, the maximal element of $L_R(m, n)$ is $\cap_{X \in M(m, n)} X = \{0\}$.

Now, give the condition of a subspace in \mathbb{F}_q^n as the elements in $L_R(m, n)$.

Theorem 2.8. Given $n > m \geq 0$, $L_R(m, n)$ consists of \mathbb{F}_q^n and all subspaces of the dimension $\leq m$ in \mathbb{F}_q^n .

Proof. According to the definition of lattice $L_R(m, n)$, $\mathbb{F}_q^n \in L_R(m, n)$. As each subspace in $L_R(m, n)$ is the intersection of finite m -dimensional subspaces, each subspace dimension in $L_R(m, n)$ that is not equal to \mathbb{F}_q^n is $\leq m$. Conversely, suppose that P is a k -dimensional subspace of \mathbb{F}_q^n , $0 \leq k \leq m$. Then, $P \in M(k, n) \subset L_R(k, n)$. From Theorem 2.7, $L_R(k, n) \subset L_R(m, n)$. Therefore, $P \in L_R(m, n)$. \square

Inference 2.2. Set $L_R(\mathbb{F}_q^n)$. Then,

$$L_R(n-1, n) = L(n, \mathbb{F}_q).$$

Theorem 2.9. Let $1 \leq m \leq n - 1$. For any $X \in L_O(m, n)$, define

$$r(X) = \begin{cases} \dim X, & \text{if } X \neq \mathbb{F}_q^n \\ m + 1, & \text{if } X = \mathbb{F}_q^n \end{cases} \quad (2.21)$$

Then, $r: L_O(m, n) \rightarrow \mathbb{N}$ is the rank function of lattice $L_O(m, n)$.

Theorem 2.10. Let $1 \leq m \leq n - 1$. For any $X \in L_R(m, n)$, define

$$r'(X) = \begin{cases} m + 1 - \dim X, & \text{if } X \neq \mathbb{F}_q^n \\ 0, & \text{if } X = \mathbb{F}_q^n \end{cases} \quad (2.22)$$

Then, $r': L_R(m, n) \rightarrow \mathbb{N}$ is the rank function of lattice $L_R(m, n)$.

Now, the characteristic polynomial of $L_R(m, n)$ is given.

Theorem 2.11. Let $n > m \geq 0$. Then,

$$X(L_R(m, n), t) = \sum_{k=m+1}^n N(k, n) g_k(t) .$$

where

$$N(k, n) = |\mathcal{M}(k, n)| = \binom{n}{k}_q = \frac{\prod_{i=n-k+1}^n (q^i - 1)}{\prod_{i=1}^n (q^i - 1)} \quad (2.23)$$

is the number of k -dimensional subspaces in \mathbb{F}_q^n , and $g_n(x)$ is the Gaussian polynomial.

Proof. Mark $V = \mathbb{F}_q^n$, $L = L_R(m, n)$, and $L_O = L_R(\mathbb{F}_q^n)$. For $P \in L$, specify

$$L^P = \{Q \in L \mid Q \subset P\} = \{Q \in L \mid Q \geq P\}$$

and

$$L_O^P = \{Q \in L_O \mid Q \subset P\} = \{Q \in L_O \mid Q \geq P\} .$$

Obviously, $L^V = L$. For $P \in L$, $P \neq V$, according to Theorem 2.8, there is $L^P = L_O^P$. Take any $P \in L_O$, where $L(X_0^P, t) = g_{\dim P}(t)$. Further, $r': L_R(m, n) \rightarrow \mathbb{N}$ defined by (2.22) is the rank function of $L = L_R(m, n)$, i.e., there is a rank function r' on L^V , and L^V is the lattice with the maximum $\{0\}$ and the minimum \mathbb{F}_q^n . Hence,

$$X(L^V, t) = \sum_{P \in L^V} \mu(V, P) t^{r'(0) - r'(P)} .$$

Think of t as a given number. Here, $f(x)$ on the left-hand side of the formula $f(x) = \sum_{x \leq y} g(y)\mu(x, y)$ is taken as $f(V) = X(L^V, t)$, and the right-hand side $g(y)$ is taken as $g(P) = t^{r'(0) - r'(P)} = t^{m+1} - r'(P)$. Then,

$$t^{m+1} = g(V) = \sum_{P \in L^V} X(L^P, t) = \sum_{P \in L} X(L^P, t) .$$

Again, we have

$$t^{m+1} = \sum_{P \in L_0} X(L_O^P, t).$$

According to Theorem 2.8, we can see that $\{P \in L_0 \mid \dim P \leq m\} = \{P \in L \setminus \{V\}\}$. Hence,

$$\begin{aligned} X(L, t) &= X(L^V, t) = t^{m+1} - \sum_{P \in L \setminus \{V\}} X(L^P, t) \\ &= \sum_{P \in L_0} X(L_O^P, t) + \sum_{P \in L \setminus \{V\}} X(L^P, t) \\ &= \sum_{\substack{P \in L_0 \\ \dim P > m}} X(L_O^P, t) - \sum_{\substack{P \in L_0 \\ \dim P \leq m}} X(L_O^P, t) - \sum_{P \in L \setminus \{V\}} X(L^P, t) \\ &= \sum_{\substack{P \in L_0 \\ \dim P > m}} X(L_O^P, t) + \sum_{P \in L \setminus \{V\}} X(L^P, t) - \sum_{P \in L \setminus \{V\}} X(L^P, t) \\ &= \sum_{\substack{P \in L_0 \\ \dim P > m}} X(L_O^P, t) = \sum_{k=m+1}^n N(k, n) g_k(t) \end{aligned}$$

where the representation of A is given by (2.24).

Now, we discuss the geometry of $L_0(m, n)$ and $L_R(m, n)$. □

Theorem 2.12. Given $1 \leq m \leq n - 1$, $L_0(m, n)$ is a finite number of lattices.

Theorem 2.13. [8] Set $1 \leq m \leq n - 1$, then

- (a) $L_R(1, n)$ and $L_R(n - 1, n)$ are finite geometric lattices;
- (b) For $2 \leq m \leq n - 2$, $L_R(m, n)$ is a finite atomic lattice, but not a geometric lattice.

Proof. By Theorem 2.8, \mathbb{F}_q^n is the only minimal element, and the atoms in m -dimensional subspaces $L_R(m, n)$ ($1 \leq m \leq n - 1$). Hence, $L_R(m, n)$ is a finite atomic lattice.

For $X \in L_R(m, n)$, define $r'(x)$ by (2.22). By Theorem 2.9, r' is the rank function of lattice $L_R(m, n)$.

When $m = n - 1$, $L_R(n - 1, n)$ is a finite number of lattices.

(b) Let v_1, v_2, \dots, v_n be a base of \mathbb{F}_q^n . Because $2 \leq m \leq n - 2$, we have the desirable $U = \langle v_1, v_2, \dots, v_m \rangle$, $W = \langle v_3, \dots, v_{m+2} \rangle \in M(m, n) \subset L_R(m, n)$. It is easy to see that $U \wedge W = \mathbb{F}_q^n$ and $r'(U \wedge W) = 0$. If $m = 2$, then $U \vee W = \{0\}$. If $m \geq 2$, then $U \vee W = \langle v_3, \dots, v_m \rangle$. In both cases, there are $\dim(U \vee W) = m - 2$ and $r'(U \vee W) = m + 1 - (m - 2) = 3$. However, $r'(U) = r'(W) = m + 1 - m = 1$. Hence,

$$r'(U \wedge W) + r'(U \vee W) \leq r'(U) + r'(W) \tag{2.24}$$

is invalid. Thus, for U and W , G_2 does not hold. Hence, for $2 \leq m \leq n - 2$, $L_R(m, n)$ is not a geometric lattice. □

2.3.3 Learning algorithms and examples

The group elements formed by n -order regular matrices can be characterised by r -parameter α_j under certain conditions. Then, $A = A(\alpha_1, \dots, \alpha_r)$, a continuous change of r parameters, may produce the entire manifold. For $GL_n(\mathbb{F}_q)$, the n^2 elements of matrix A calibrate the points in the n^2 -dimensional Euclidean space. This change in the n^2 parameters describes the change relation from one point in Euclidean space to another. Thus, $GL_n(\mathbb{F}_q)$'s subgroups of various r parameters can be represented by some r -dimensional subspaces in the n^2 -dimensional Euclidean space. The change of data state in the learning system is proved by the transformation of the parameter.

More specifically, starting from any element a , a continuous change through the r parameter produces an arc that links any pair of spaces in a group, forming the structure of the graph.

Here is an algebraic view of the Lie group's topology, where the set $U^q(n, r)$ consists of integer $n + mq^r$ ($m = 0, \pm 1, \pm 2, \dots, q$) in any neighbourhood of any integer $n \in M$ (where q is a fixed prime and r is an integer). Selecting a q defines a specific topology for M . For a given neighbourhood of n , the integer is fixed, m is dynamic, and different r values give the different neighbourhoods of n [17].

Through connectivity and topology, consider specific learning issues as follows. Each sample x_i is represented $x_i = (k_1, k_2, \dots, k_m)$ by specific attribute values, and assume that the problem has several categories, denoted by $C = (c_1, c_2, \dots, c_n)$. According to the attribute value, each sample in the sample set is represented in the learning subspace, and the samples of the same class have similar or equal attribute values. They should be connected to each other, and some attribute values of different classes may be similar or equal, but different attribute values are bound to exist. Of course, we can first determine the key attribute value k_i (centre point) of each category, and by comparing the distance from k_i , we can effectively determine the category to which the sample belongs. A sample in the same class must exist in the neighbourhood of another sample, and different class sets are disconnected from each other. The learning subspace orbit generation lattice algorithm under the action of the general linear group is given below.

2.3.3.1 Learning subspace orbit generation lattice algorithm under the action of the general linear group

Algorithm 2.6: ILLE algorithm

General Linear Group Generated_ORBITS ($P, c(x), GL_n(\mathbb{F}_q)$)

// P is the sample data set, $c(x)$ is the objective function, and $GL_n(\mathbb{F}_q)$ is the basis of the track division

- 1) The sample data set P is mapped to \mathbb{F}_q , i.e., $\Phi: P \rightarrow \mathbb{F}_q$
// Corresponding to the specific problem using different Φ
- 2) Take the n -dimensional row vector on \mathbb{F}_q to form a set \mathbb{F}_q^n ,

$$\mathbb{F}_q^n = \{(x_1, x_2, \dots, x_i, \dots, x_n) | x_i \in \mathbb{F}_q, \quad i = 1, 2, 3, \dots, n\}$$

// Use vector $(x_1, x_2, \dots, x_i, \dots, x_n)$ to represent any sample

- 3) Call algorithm 4, set T , get $\text{GL}_n(\mathbb{F}_q)$
- 4) **for** ($i = 1; i \leq m, i++$)
- (1) Take $T_i \in \text{GL}_n(\mathbb{F}_q)$ and treat \mathbb{F}_q^n as follows:

$$((x_1, x_2, \dots, x_i, \dots, x_n), T_i) \mapsto (x_1, x_2, \dots, x_i, \dots, x_n) T_i$$

The subspace orbits $M(m, n)$ of \mathbb{F}_q^n are obtained; //m satisfaction relation

$$0 < m < n$$

- (2) The set $L(m, n)$ is generated from the intersection of the subspace of $M(m, n)$. Specify the partial order on $L(m, n)$ according to the inclusion (inverse inclusion) relationship. The orbit generating lattice $L_O(m, n)$ (or $L_R(m, n)$) is obtained.
 - (3) Determine the greatest element (or smallest element) $\cap_{X \in M(m, n)} X$ $L_O(m, n)$ (or $L_R(m, n)$)
 - **if** X is found successfully, then get the value of $c(x)$, $F \leftarrow (T_i, c(x))$;
 - **Else** discard X ;
 - End if**;
 - (4) Delete the sample in the sample set \mathbb{F}_q^n that has obtained the target learning results, and get the new sample set \mathbb{F}_q^n ;
 - End for**;
 - (5) **For** each objective function $c(x)$, output set F .
-

2.3.3.2 Case study [15]

Take a set of data that is the result of the analysis of the chemical composition of a wine produced in the same area of Italy, produced from three different plants. Some of the data is listed in Table 2.7. The data in Table 2.7 has 15 columns, where ID is the data number, and the class for its category. Each data has 13 attributes, namely

- 1) Alcohol
- 2) Malic acid
- 3) Ash
- 4) Alkalinity of ash
- 5) Magnesium
- 6) Total phenols
- 7) Flavanoids
- 8) Nonflavanoid phenols
- 9) Proanthocyanins
- 10) Colour intensity
- 11) Hue
- 12) OD280/OD315 of diluted wines
- 13) Proline

These are represented by Attr1, Attr2... Attr13 in the table. There is no attribute deletion for the data given.

There are 178 experimental data, divided into three categories:

class 1 59 class 2 71 class 3 48

Tab. 2.7: part of the experimental data sheet.

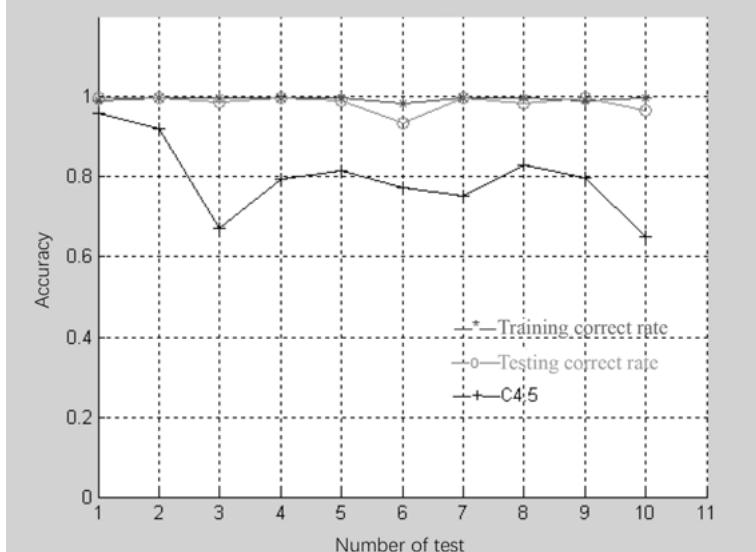
ID	Class	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7
1	1	14.23	1.71	2.43	15.6	127	2.8	3.06
2	1	13.2	1.78	2.14	11.2	100	2.65	2.76
3	1	13.16	2.36	2.67	18.6	101	2.8	3.24
4	1	13.24	2.59	2.87	21	118	2.8	2.69
...								
120	2	13.49	1.66	2.24	24	87	1.88	1.84
121	2	12.99	1.67	2.6	30	139	3.3	2.89
122	2	11.96	1.09	2.3	21	101	3.38	2.14
123	2	11.66	1.88	1.92	16	97	1.61	1.57
...								
ID	Class	Attr8	Attr9	Attr10	Attr11	Attr12	Attr13	
1	1	0.28	2.29	5.64	1.04	3.92	1065	
2	1	0.26	1.28	4.38	1.05	3.4	1050	
3	1	0.3	2.81	5.68	1.03	3.17	1185	
4	1	0.39	1.82	4.32	1.04	2.93	735	
...								
120	2	0.27	1.03	3.74	0.98	2.78	472	
121	2	0.21	1.96	3.35	1.31	3.5	985	
122	2	0.13	1.65	3.21	0.99	3.13	886	
123	2	0.34	1.15	3.8	1.23	2.14	428	
...								

In this example, each sample data is denoted by $x = \{x_1, x_2, \dots, x_{12}, x_{13}\}$, where x_i represents a single attribute value for sample x , representing sample sets with P , $P = \{x_1, x_2, \dots, x_i, \dots\}$, i.e., the sample set can be represented as a matrix structure, and each sample is a row vector space. Since the selected sample data are continuous data, consider the distance between the attribute values (which can be regarded as similarity) to specify the partial order relation on the sample set. Select a specific sample in the collection as a reference point, and calculate the similarity of the test sample with it to get the attribution of each sample, i.e., the final classification. Each

Tab. 2.8: classification accuracy rate comparison.

The number of experiments	1	2	3	4	5	Average correct rate
Training correct rate	0.9916	1	1	1	1	0.9926
Testing correct rate	1	1	0.9863	1	0.9926	0.98637
C4.5	0.9611	0.9214	0.6732	0.7958	0.8150	
The number of experiments	6	7	8	9	10	
Training correct rate	0.9845	1	1	0.9916	1	0.99677
Testing correct rate	0.9352	1	0.9835	1	0.9661	0.98637
C4.5	0.7746	0.7521	0.8311	0.7984	0.6522	0.79749

Classification result chart

**Fig. 2.5:** classification accuracy graph.

time, from the three types of data, two-thirds of the data are randomly selected for training, and the remaining one-third of the data are used for testing. The correctness of the final classification results is compared with the correctness of the results obtained using the decision tree C4.5 algorithm. Table 2.8 shows the correctness of the classification, and the classification accuracy is shown in Figure 2.5.

2.4 Summary

We further expanded the content of Lie group machine learning, given (1) the basic concept of the relevant partial set and lattice in LML, (2) the LML subspace orbit generation lattice learning algorithm, and (3) the LML learning subspace orbit generation lattice learning algorithm under the action of general linear group $GL_n(\mathbb{F}_n)$. According to the independent division hypothesis axiom, consistency hypothesis axioms and the basic facts of each element in the observation space set can be described by a set of independent real parameters, combined with the characteristics of the geometric model of Lie group machine learning. On the basis of the Lie group machine learning space, the concept of Lie group machine learning subspace orbit generating lattice was established, and the generating lattice theory and related learning algorithm lattice theory, which are derived from multiple identical dimensions or orbits and rank subspaces, were given under the action of various typical groups in the observation set, such as the orbit generating lattice learning algorithm of general linear group, orbit generating

lattice breadth-first and depth-first learning algorithms, heuristic learning algorithm, and other typical groups of learning algorithms. The algorithm proposed in this chapter was also verified by a corresponding example.

Bibliography

- [1] Li FZ, Kang Y. The study of machine learning theory frame based on Lie group. *Journal of Yunnan University for Nationalities*, 2004, 13(4), 251–255.
- [2] Xu H, Li FZ. Lie group machine learning's axiom hypothesizes// IEEE International Conference on Granular Computing. IEEE, 2006, 401–404.
- [3] Shi J, Cao H. Research on Heuristic Search. *Microcomputer & Its Applications*, 2004, 23(10), 10–12.
- [4] Wan Z. *Geometry of Classical Groups over Finite Fields*, Student literature. Lund, Sweden/Chartwell_Bratt, Browley, United Kingdom, 1993.
- [5] Artin E. *Geometric Algebra*. Interscience Publishers, 1957.
- [6] Huo Y, Wan ZX. On the Geometricity of Lattices Generated by Orbit of Subspaces Under Finite Classical Groups. *Journal of Algebra*, 2001, 243(1), 339–359.
- [7] Witt E. Theorie der quadratischen Formen in beliebigen Körpern. *Journal Für Die Reine Und Angewandte Mathematik*, 1937, 1937(176), 31–44.
- [8] Huo Y, Liu Y, Wan ZX. Lattices generated by transitive sets of subspaces under finite classical groups II the orthogonal case of odd characteristic. *Communications in Algebra*, 1992, 20(4), 1123–1144.
- [9] Wan HP, He HC, Zhou YQ. Local nuclear methods and applications. *Journal of Shandong University*, 2006, 41(3), 60–62.
- [10] Cristianini N, Shawe-Taylor J. *An introduction to support Vector Machines: and other kernel-based learning methods*. Printed in the United Kingdom at the University Press, 2000.
- [11] Borel A. *Linear Algebraic Groups*. Benjamin, New York, 1971.
- [12] Bott R. Homogeneous Vector Bundles. *Annals of Mathematics*, 1957, 66(2), 203–248.
- [13] Jacobson N. *Lie Algebras*. Wiley, New York, 1962.
- [14] Snow DM. Weyl group orbits. *ACM Transactions on Mathematical Software*, 1990, 16(1), 94–108.
- [15] UCI Learning Repository, <http://www.ics.uci.edu/mlearn/databases.html>.
- [16] Chen SF, Chen ZQ. *Artificial Intelligence and Knowledge Engineering*. Nanjing: Nanjing University Press, 1997.
- [17] Ma ZQ. Group theory in Physics. Beijing: *Science China Press*, 1998.
- [18] Chen F. Lie Group Machine Learning (LML) Subspace Orbit Generation Algorithm and Its Application. Thesis. Soochow University, 2007.
- [19] Chen F, Li FZ. Lie Group Machine Learning (LML) Subspace Orbit Generation Theory and Algorithm. *Journal of Soochow University*, 2007, 23(1), 61–66.
- [20] Chen F, Li FZ. A Study on Learning Subspace Orbit Generating Lattice Algorithm under the Action of $GL_n(F_q)$ Group in Lie Group Machine Learning (LML). Nanjing. First Jiangsu Province Young Scientists Annual Meeting Proceedings collected papers, 2006, 36–46.
- [21] Chen F, Li FZ. Lie Group Machine Learning (LML) Subspace Orbit Generation Lattice Algorithm. *Computer Engineering and Applications*, 2007, 43(15), 184–187.
- [22] Li FZ, He SP, Lin TY. Lie group machine learning, Nanjing. The second annual Young Scientists in Jiangsu Province, 2008.
- [23] Wan ZX, Huo ZX. A Lattice Generated by a Limited Typical Subgroup Space Orbit, Beijing: *Science China Press*, 2004.

3 Symplectic group learning

This chapter is divided into five sections. Section 3.1 presents a question. Section 3.2 introduces the design of the symplectic group classifier in Lie group machine learning. Section 3.3 describes the algorithm of the symplectic group classifier in Lie group machine learning. Section 3.4 presents application examples. Finally, Section 3.5 summarises the chapter.

3.1 Problem presentation

We know that when a problem is described by the group theory, the basic characteristics of the problem prototype should be kept as discrete as possible. To achieve this effect, discretisation should be carried out as far as possible in the same framework of the problem prototype. For example, the finite element method is carried out under the same framework of the Sobolev function space composed of the discrete solution of the prototype, so that the basic features, such as symmetry and positive qualitative, are preserved. Thus, validity and reliability are guaranteed fundamentally. To further study Lie group machine learning, among the various types of Lie groups, we choose the symplectic group, which is a typical Lie group, to construct the symplectic group classifier. The following is a brief introduction of the symplectic group and symplectic geometric research results.

- (1) Implementation process that Feng Kang et al. applied to a Hamiltonian system using the symplectic geometry algorithm [1].

A Hamiltonian system is an important type of dynamical system. All real and dissipative negligible physical processes can be expressed as Hamiltonian systems, which are widely used in structural biology, celestial mechanics, and partial differential equations. Some of them have been included in the American research focus program ‘Grand Challenges’. Feng Kang and his research group carried out an in-depth and systematic study of this problem. They developed a new branch-symplectic geometry algorithm of computational methods and achieved important results in this field. These results have been widely applied to dynamic astronomy research and analytical mechanics theory. Feng Kang proposed the application of symplectic geometry to numerical analysis at the Beijing Symposium on Differential Geometry and Differential Equations–Computation of Partial Differential Equations in 1984, mainly on the basis of the basic theorem in analytical mechanics: the solution of the system is a single parameter measure preserving transformation, i.e., symplectic transformation, which creates a new method of Hamilton’s mechanical calculation and overcomes the problem of the discretised equation maintaining the symplectic structure of the original system. Its discrete phase flow can be regarded as a series of discrete symplectic transformations to

maintain the system phase area and phase volume. The basis of the Hamiltonian method is the symplectic difference scheme. It is valuable to establish the equivariant theory of bifurcation and chaos of the generalised Hamiltonian system by the group theory. In particular, the Hamiltonian system with certain symmetry can be transformed into a lower-dimensional system on the appropriate Poisson manifold. Feng Kang's research achievements in the field of symplectic geometry can be summarised as follows where Kang:

- (a) Proposed a complete theoretical framework of symplectic geometry algorithm for Hamiltonian systems.
- (b) Popularised the generation function of the analytical mechanics and the Hamilton–Jacobi equation theory, and constructed a symplectic scheme with multiple arbitrary-order precisions.
- (c) Developed a complete theory of the power series of the form vector field and the phase flow, now called the posterior error analysis of the ordinary differential equation.
- (d) Generalised the symplectic algorithm of Hamiltonian systems to dynamical systems with the Lie algebra structure, realised the geometricisation of the dynamical system algorithm, and constructed the contact algorithm for the contact system as well as the volume preserving algorithm for the passive system.

This algorithm has a unique advantage in terms of stability and long-term tracking capability. Compared with the classical R-K method, it has good stability.

- (2) William T Shaw explained the difference between Penrose's double-sided twisting theory and Ashtekar's BMS group theory, and compared the two methods [2]. Ashtekar and Streubel proposed a phase space of the radiation model in a gravitational field \hbar^+ . The authors concluded that the BMS group in this phase space preserves the theory of the symplectic structure. In the unit space, the angular momentum in the theory of the electromagnetic field corresponds to the integral of the density of the Hamiltonian system. Specifically, it is the transformation of the generator K . The following equation is used in the symplectic method to obtain the integral of the two spherical surfaces,

$$H_{Q_K}[S_2] - H_{Q_K}[S_1] = \int\limits_{S_1}^{S_2} H_{F_K} d^3 S$$

where S_1 and S_2 are two spheres in gravity field \hbar^+ . To observe the similarities and differences with regard to the Penrose method, the author presented the proof of this method, from which we can see that the symplectic structure is superior in solving the momentum problem.

- (3) Gao et al. applied the symplectic geometry algorithm to seismic ray calculation [3]. The results showed that the obtained ray has good smoothness and a strong ability to penetrate the shadow area. From the definition of the symplectic group,

it can be shown that its Lie algebra is composed of an infinitesimal symplectic transformation B satisfying $B'J + JB = 0$. Since there is no non-degenerate anti-symmetric matrix in the odd dimension, the symplectic space is even-numbered-dimensional, as is the phase space. The symplectic geometry algorithm is a computational method for Hamiltonian systems. It makes the discretisation equation maintain the symplectic structure of the system. The author used the symplectic geometry algorithm and the fourth-order Runge–Kutta algorithm, which is a dissipative algorithm, to calculate and compare the ray path and travel time. At the same time, the author used the symplectic algorithm to calculate the ray path and amplitude of the geological structure of the first section. In this paper, the symplectic difference scheme of the seismic ray equation is given. From the general form of the ray tracing equation, a set of regular equations is obtained by simplification, and then a simple second-order Euler-type symplectic difference scheme is adopted,

$$\begin{cases} q^l = q + \frac{\tau}{2}\varphi_p(p) \\ \hat{p} = p - \tau\phi_q(q') \\ \hat{q} = q^l + \frac{\tau}{2}\varphi_p(\hat{p}) \end{cases}$$

where q^l represents the intermediate variable and p represents the generalised momentum. After the demonstration analysis, the results of the first-order cross-section design using the symplectic algorithm are found to be in agreement with the expected values.

- (4) Wu et al. applied the symplectic algorithm to the dynamics of the multi-body system [4], which has been successfully used to solve the multi-body system dynamics equation. In general, the symplectic algorithm of the dynamical equation of the non-constrained regular form can be directly exploited by Feng Kang's symplectic algorithm, but the solution of the multi-body system can be obtained using the synthetic symplectic algorithm. Because the main part of the multi-power system is separated into two parts, it is natural to think of the usual R-K (Runge–Kutta) method for integration. This algorithm has a good truncation error. The solution of such a multi-power equation can be a simple Newton iteration method, but the Newton iteration method has some drawbacks in solving stiff Hamiltonian systems. Therefore, the author proposed a synthetic symplectic algorithm based on the backward error analysis theory of numerical analysis on the manifold. This is based on the numerical solution of ordinary differential equations, which can be regarded as the exact solution of the perturbed differential equation, and then extended to the manifold M .

Suppose that X is the C^∞ vector field ($M \xrightarrow{X} TM$)

$$\dot{x} = X(x) \quad (3.1)$$

i.e., C^∞ , manifold $M \in R^m$, where $x(t) = \exp(tX)$ is the phase flow of (3.1). Further, $\psi_h : M \rightarrow M$ is the numerical approximation of the exact time h flow of (3.1), and

there is a perturbation vector field $\hat{X}(h): M \rightarrow M$

$$\psi_h = \exp(h\hat{X}(h)) \quad X(h) = \hat{X}(h) + O(h^0).$$

If X is a Hamiltonian structure, ψ_h is a symplectic form and $\hat{X}(h)$ is also a Hamiltonian structure. This algorithm can be used to construct the high-order symplectic algorithm. It is based on the numerical solution of ordinary differential equations. Further, it can be regarded as the exact solution of the corresponding perturbed differential equation and then extended to the manifold problem.

- (5) Wang et al. established the regular equation of the dynamics of a robot system and combined the equation with the symplectic algorithm to eliminate the accumulation of the error in the dynamic calculation and ensure the accuracy of the calculation [5]. The authors proposed that the kinetic equations of the robot system have many methods, but the methods are equivalent. Their results can be expressed as a set of differential-algebraic equations. In terms of the solution of the differential-algebraic equations, there are three main problems:
 - (a) The constrained rank reduction problem;
 - (b) The default problem of the integral process;
 - (c) The stability of the numerical calculation.

The default problem of the numerical integration process is the key to the accuracy of the numerical calculation. To ensure the accuracy of the calculation, Baurngarte proposed a correction based on the amount of breaching, and this correction needs the introduction of the correction coefficient. For this reason, the author combined the regular equation and symplectic group scheme, which is used to eliminate the error accumulation of the robot dynamics calculation, to ensure the accuracy of the calculation.

According to the second-order dynamical equations of the system and the regular equations of the system, the regular equations of the system are unified as

$$\dot{Z} = f(Z) = J^{-1} \frac{\partial H}{\partial Z}.$$

Then, the system matrix of the equation is given. The authors method divided the differential manifold into a symplectic manifold, which reflects the essence of the discrete system dynamics and improves the computational precision.

- (6) Han et al. used the symplectic algorithm for molecular dynamics calculations in nanometre processing [6]. Assume that the expression of the Hamiltonian system is

$$H = \sum_i \frac{p_i^2}{2m_i} + U$$

where p_i is the momentum of the i -th particle and U describes the potential energy function of the interaction between particles. This formula can generally be solved

numerically as follows:

do $i = 1$ to m

$$\begin{aligned} p_i &= p_{i-1} - d_i \tau \frac{\partial U(q_{i-1})}{\partial q_{i-1}} \\ q_i &= q_{i-1} - c_i \tau \frac{\partial U(p_{i-1})}{\partial p_{i-1}} \end{aligned} \quad (3.2)$$

$i = i + 1$

end do

In (3.2), q_i is the position coordinate of the i -th particle. In the symplectic integral, the coefficient (c_i, d_i) should be determined by the condition that the error of the newly calculated coordinate satisfies $O(\tau_{L+1})$, and the difference equation at this time has L -order accuracy. When the order L is lower than 4, the factor in (3.2) is equal to 0, which simplifies the calculation. When $L > 4$, we can repeatedly apply (3.2) to get the trajectories of particles in space. Note that (3.2) is self-starting, so it is not necessary to store the intermediate results in general. Thus, this method requires very low storage capacity. It has been proven that the symplectic algorithm is a high-precision convergence algorithm. In the molecular dynamics simulation, the equations of motion represent a series of initial difference equations. In principle, these equations can be obtained by standard numerical integral solutions. However, many of the commonly used numerical integral algorithms are non-symplectic algorithms. The disadvantage of this algorithm is that the energy of the system will be dissipated in the simulation for a long time; hence, the simulation results cannot be analysed by the principle of statistical mechanics. The symplectic numerical integration algorithm used in molecular dynamics can be divided into two types: prediction algorithm and prediction-correction algorithm. The author proved the effectiveness of the symplectic algorithm.

- (7) Zhang Kai applied a type of multi-symplectic algorithm to the PML (perfectly matched layer) method in an electromagnetic field [7]. The symplectic method is used to reveal the equivalence of two Hamiltonian forms of a Hamiltonian microsystem. For the nonlinear wave equation, the relationship between the algorithm and the general difference scheme is established using this multi-symplectic algorithm. Then, according to the different forms of the nonlinear wave equation and the effect of different methods in the symplectic space and time direction, an effective symplectic algorithm is obtained. In addition, the author considered the stability of the hyperbolic equation numerical format, gave a criterion for evaluating the form of the symplectic form in many aspects, and discussed the Euler box format of a variety of symplectic algorithms as well as some results of the stability.
- (8) Xu et al. gave the numerical solution of a differential strategy based on a Hamiltonian system and the symplectic algorithm [8]. It is useful to employ a differential

solution to solve the two-point problem. The differential game problem can be attributed to the Hamiltonian system. An important geometric feature of the Hamiltonian system is that its phase flow maintains the symplectic structure; hence, the symplectic geometry algorithm has a good effect on the linear optimal problem. The symplectic geometry algorithm can maintain the symmetry; hence, the algorithm can maintain the Hamiltonian symplectic structure and all the features, such that the energy of the system does not change for a long time, and it yields good results in solving the differential game calculation problem.

- (9) Sun et al. presented an application of the symplectic semi-analytical method based on a Hamiltonian system in heterogeneous electromagnetic waveguides [9]. The Hamiltonian system in mechanics uses a dual variable to describe the problem. In the electromagnetic field, there is an electric field and a magnetic field forming a dual variable. The Hamiltonian system is applied to the electromagnetic waveguide problem. According to the Hamiltonian system theory of electromagnetic waveguides, the symplectic geometry is applied to the system. The symplectic geometry can be used for any heterogeneous material, and the symplectic semi-analytic method is given. According to the Hamiltonian variational principle, the Hamiltonian system is fundamentally a mixed hybrid system, and the problem is described by the dual variable, which coincides with the characteristics of the electromagnetic field. From the experimental results, we can see that the symplectic algorithm can yield very good results.
- (10) Zhang Peiguo presented the DNA structure dynamics computer simulation algorithm and the graphics post-processing method [10]. In this paper, the ordinary differential/algebraic equations describing the formation of super-long elastic rods are established by combining the elastic thin rod model and the related research results. The equations are expressed as the Hamilton equations, and the numerical solution of the surface equation is given by the symplectic algorithm. The paper presents a computational example of the graphics processing of the super-long elastic rods. For the study of DNA, the elastic rod with the original twist rate is used as the macroscopic mechanical model of DNA. In the process of solving the problem, not only the main parameters, such as curvature, disturbance rate, and frame, are used, but also the corresponding graphical representation model and algorithm are designed to give the computer graphics representation of DNA. In this paper, the mathematical model of the DNA structure is transformed using the symplectic structure preserving the transformation method. Finally, the relevant algorithm is used to simulate the DNA model very accurately. This method provides support for models and algorithms for the computer graphics of the structure and dynamics of ultra-thin elastic rods.

In summary, symplectic geometry and symplectic groups have been successfully applied to many fields, but these applications are mostly involved in the field of dynamics. In recent years, the application of partial differential equations and contour

curve evolution to image processing and computer vision has emerged as an important method. As the symplectic geometry has obvious advantages in dealing with partial differential problems, we combine the image processing and study the problem of learning from the perspective of machine learning. Since the symplectic group is also a differential manifold, in the field of image processing, we can study the evolution of the curve of the geometric surface by the differential method, which transforms the curve evolution problem into a purely partial differential equation. In describing the geometric characteristics of the curve, the unit vector and curvature are two important parameters. The unit vector describes the direction of the curve, while the curvature describes the bending degree of the curve. Curve evolution theory uses the geometric parameters of the unit vector and curvature of a curve to study the curve changes with time. In the case of the parameter curve, the curve relies on the derivative of any parameterised curve. Assume that the curve is $C(p, t) = [x(p, t), y(p, t)]$, where p is the amount of any parameter change and t is the time. If the inward unit normal vector of the curve is n , and the curvature is k , the evolution of the curve along its unit vector direction can be expressed by the following equation

$$\frac{\partial c}{\partial t} = V(k)n .$$

If the geometric curve of the model is applied to image segmentation, we need to introduce the concept of geodesic distance. At the same time, according to the function of the image data, the information is segmented from the image. The description of the problem usually requires that the parameters of the model must be elements of the Euclidean vector space. Here, we try to study the problem in symplectic space, which has different structures compared to the Euclidean space. Since the orthogonal group is obtained by the Euclidean scalar product, and the symplectic group is obtained by the anti-symmetric scalar product, this chapter applies Lie group to the symplectic manifold. The symplectic structure on the manifold is a non-degenerate closed differential 2 form, which is the same in the symplectic manifold and in the Riemannian manifold. A vector field on a manifold defines a phase flow, i.e., a single parameter differential homologous group. The vector field on the manifold can also form a Lie algebra. By the relevant theorem, any part of the symplectic manifold can be transformed into a standard symplectic structure. Since the symplectic structure of symplectic space is similar to that of the Euclidean structure in some respects, the symplectic structure of the symplectic manifold is similar to the Riemann structure. The Riemann structure gives the Euclidean structure on the tangent space and makes the tangent space and the cotangent space isomorphic. The symplectic structure is the same. Assuming that (M, ω) is a symplectic manifold, any tangent vector $\eta \in T_x M$ gives a linear form on $T_x M$. The symplectic algebraic structure is the symplectic geometry after integration. Further, the symplectic geometry is used to construct the surface segmentation model on the symplectic space to solve the problem of geometric segmentation in image processing. Then, the corresponding learning classifier is constructed according

to the symplectic group structure to solve key problems in reasoning, learning, and decision-making.

From the concept and related properties of the symplectic group, we know that the symplectic group is a $2n$ -dimensional structure, which is composed of the product of the matrix. In the $2n$ -dimensional space, a 2-dimensional opposition matrix is introduced, and the unit matrix I in the real orthogonal matrix definition is replaced by the antisymmetric matrix J , so that the real pseudo-matrix can be obtained. The symplectic group is the group of the product of all the $2J \times 2J$ real pseudo-matrices. We can use the structure of a $2n$ dimension of a symplectic group to carry out the process of learning in the framework of a group, which satisfies the duality hypothesis in Lie group machine learning. Since the real pseudo-matrices of the symplectic group are orthogonal, it is assumed that the v_1 and v_2 matrices are orthogonal. If v_1 and v_2 are mapped to the object of the learning process, the process corresponds to the learning process of training samples and training objectives. The problems in the study of Lie group machine learning are presented in a symplectic group structure to study the classifier problem in Lie group machine learning. The following section introduces the symplectic group of classifier design. Please refer to Appendix 17.4 for the basic concepts of symplectic groups.

3.2 Design of the symplectic group classifier in Lie group machine learning

In this section, we present the description of the symplectic group classifier. The dimensionality reduction process of a high-dimensional image in the symplectic group framework is given for a concrete example of image classification. Then, the problem description in the sympathetic group framework is given. The sample of the classification is expressed as a symplectic matrix. Then, the method of sample representation in the symplectic matrix and the simplification method of the symplectic matrix are given. Finally, the design method and process of the symplectic group classifier are given [11, 12].

3.2.1 Symplectic group classifier description

References [13–20] have described the relevant knowledge of the classifier and the general design method. This section uses the symplectic group to construct the classifier, i.e., the symplectic group classifier. The principle can be described as follows: a $2n$ -dimensional eigenvector is obtained by feature extraction of a $2n$ -dimensional sample. The process of solving the weight vector is the training process of the classifier. The training process for obtaining the weight vector using the known class learning sample is shown in Figure 3.1:

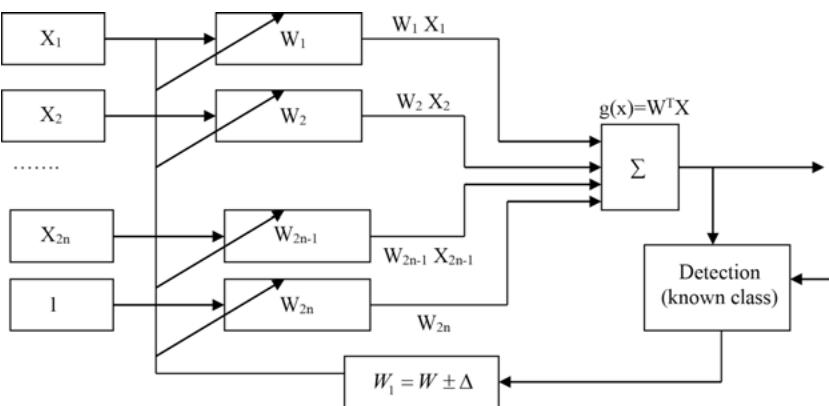


Fig. 3.1: Training illustrations for obtaining weight vectors by using known samples.

Known $x_1 \in \omega_1$, by detecting the adjustment weight vector, and finally let $x_1 \in \omega_1$. Known $x_2 \in \omega_2$, by detecting the adjustment weight vector, and finally let $x_2 \in \omega_2$. In this way, we can determine the weight vector by a limited number of samples.

The training process is to solve the weight vector w of a known class of sample sets, which is a process of solving a linear simultaneous inequality equation. When solving:

- (1) Only a linear separable problem $g(x) = W^T X$ can be solved;
- (2) The solution of simultaneous equations is non-singular under different conditions, there are different solutions; hence, the problem of finding the optimal solution is obtained.
- (3) The process of solving W is the process of training. The common point of training is to first give the criterion function first and then find the optimisation algorithm that makes the criterion function tend to be extreme. Different algorithms have different criterion functions. The algorithm can be divided into iterative methods and non-iterative methods. In this chapter, we use the knowledge of the symplectic group to give the design process of the symplectic group classifier in classification and image recognition.

We propose a dimensionality reduction method for the dimensionality reduction of high-dimensional images in the group of symplectic groups, which serves as the foundation for the study of the classifier design process. 3D image stream analysis and shape analysis are important research areas in the fields of computer vision and image processing. Squire et al. [21] has introduced the technique of applying the Lie group method to 3D visual motion and 3D image dimension reduction in 2-dimensional space. In the actual process, the dimension of the image to be processed is usually high, so it must be reduced. The n -dimensional image is reduced to an $n-1$ -dimensional image, the $n-1$ -dimensional image is reduced to an $n-2$ -dimensional image,

and so on. Then, the image can be reduced to two dimensions and the image processing becomes much simpler. When sampling n -dimensional images, the coordinates of the sample points (x_1, x_2, \dots, x_n) are rational numbers (satisfying the sampling theorem). In other words, we map the points (x_1, x_2, \dots, x_n) of the n -dimensional space to the $n-1$ -dimensional space. The corresponding points are constructed

$$\left(\frac{i_1}{j_1}, \frac{i_2}{j_2}, \dots, \frac{i_n}{j_n} \right)$$

These points are mapped to

$$\left(\frac{i_1}{j_1} + \lambda \cdot \frac{i_n}{j_n} \cdot \cos \alpha, \frac{i_2}{j_2} + \lambda \cdot \frac{i_n}{j_n} \cdot \sin \alpha, \dots, \frac{i_{n-1}}{j_{n-1}} \right)$$

where $\lambda > 0$ is the scaling factor and α satisfies the condition that $\tan \alpha$ is an irrational number, which ensures that the map is a single mapping (different points in the n -dimensional space are also mapped to different points in the $n-1$ -dimensional space). In this way, an n -dimensional image can be reconstructed in the $n-1$ -dimensional space. The selection of the above α ensures that the mapping is a single mapping. The reason is as follows: if two different points

$$\left(\frac{i_1}{j_1}, \frac{i_2}{j_2}, \dots, \frac{i_n}{j_n} \right) \text{ and } \left(\frac{p_1}{q_1}, \frac{p_2}{q_2}, \dots, \frac{p_n}{q_n} \right)$$

are mapped to the same point in the $n-1$ -dimensional space, according to the definition of the mapping, for $2 < k < n$, we have

$$\frac{i_k}{j_k} = \frac{p_n}{q_n}$$

and

$$\begin{aligned} \frac{i_1}{j_1} + \lambda \cdot \frac{i_n}{j_n} \cdot \cos \alpha &= \frac{p_1}{q_1} + \lambda \cdot \frac{p_n}{q_n} \cdot \cos \alpha \\ \frac{i_2}{j_2} + \lambda \cdot \frac{i_n}{j_n} \cdot \sin \alpha &= \frac{p_2}{q_2} + \lambda \cdot \frac{p_n}{q_n} \cdot \sin \alpha . \end{aligned}$$

Then, explain the case for $k = 1, 2$; if $i_1/j_1 \neq p_1/q_1$, then we have

$$\tan \alpha = \frac{\frac{i_2}{j_2} - \frac{p_2}{q_2}}{\frac{i_1}{j_1} - \frac{p_1}{q_1}} .$$

This is contradictory to $\tan \alpha$, so $i_1/j_1 = p_1/q_1$, which also explains $i_2/j_2 = p_2/q_2, \dots$ so that points $(i_1/j_1, i_2/j_2, \dots, i_n/j_n)$ and $(p_1/q_1, p_2/q_2, \dots, p_n/q_n)$ are certain.

If the definition $f(i_1/j_1, i_2/j_2, \dots, i_n/j_n)$ is the grey level of point $(i_1/j_1, i_2/j_2, \dots, i_n/j_n)$, the grey level of point $(i_1/j_1 + \lambda \cdot i_n/j_n \cdot \cos \alpha, i_2/j_2 + \lambda \cdot i_n/j_n \cdot \sin \alpha, \dots, i_{n-1}/j_{n-1})$ is also $f(i_1/j_1, i_2/j_2, \dots, i_n/j_n)$, i.e., under such a mapping, $f(x_1, x_2, \dots, x_n)$ at n dimension and $F(x_1, x_2, \dots, x_{n-1})$ at $n-1$ dimension satisfy

$$f(x_1, x_2, \dots, x_n) = F(x_1 + \lambda x_n \cos \alpha, x_2 + \lambda x_n \sin \alpha, x_3, \dots, x_{n-1}) .$$

Considering the convolution of $f(x_1, x_2, \dots, x_n)$ and $g(x_1, x_2, \dots, x_n)$,

$$[f(x_1, x_2, \dots, x_n) * g(x_1, x_2, \dots, x_n)](u_1, u_2, \dots, u_n) \\ = \int \cdots \int f(x_1, x_2, \dots, x_n) \times g(u_1 - x_1, u_2 - x_2, \dots, u_n - x_n) dx_1 dx_2 \dots dx_n .$$

According to the above mapping, we can get

$$f(x_1, x_2, \dots, x_n) = F(x_1 + \lambda x_n \cos \alpha, x_2 + \lambda x_n \sin \alpha, x_3 \dots, x_{n-1}) g(x_1, x_2, \dots, x_n) \\ = G(x_1 + \lambda x_n \cos \alpha, x_2 + \lambda x_n \sin \alpha, x_3 \dots, x_{n-1}) .$$

Therefore,

$$\int \cdots \int f(x_1, x_2, \dots, x_n) \times g(u_1 - x_1, u_2 - x_2, \dots, u_n - x_n) dx_1 dx_2 \dots dx_n \\ = \int \cdots \int F(x_1 + \lambda x_n \cos \alpha, x_2 + \lambda x_n \sin \alpha, x_3 \dots, x_{n-1}) \\ \cdot G(u_1 - x_1 + \lambda(u_n - x_n) \cos \alpha, u_2 - x_2 \\ + \lambda(u_n - x_n) \sin \alpha, u_3 - x_3 \dots, u_{n-1} - x_{n-1}) dx_1 dx_2 \dots dx_n .$$

The corresponding coordinates are transformed as follows,

$$\begin{cases} S_1 = x_1 + \lambda x_n \cos \alpha ; \\ S_2 = x_2 + \lambda x_n \sin \alpha ; \\ S_3 = x_3 ; \\ \dots \\ S_n = x_n \end{cases}$$

then,

$$\int \cdots \int F(x_1 + \lambda x_n \cos \alpha, x_2 + \lambda x_n \sin \alpha, x_3 \dots, x_{n-1}) \\ \cdot G(u_1 - x_1 + \lambda(u_n - x_n) \cos \alpha, u_2 - x_2 + \lambda(u_n - x_n) \sin \alpha , \\ u_3 - x_3, \dots, u_{n-1} - x_{n-1}) dx_1 dx_2 \dots dx_n \\ = \int \cdots \int F(s_1, s_2, \dots, s_n) \times G(u_1 + \lambda u_n \cos \alpha - s_1 , \\ u_2 + \lambda u_n \cos \alpha - s_2 u_3 - s_3, \dots, u_{n-1} - s_{n-1}) \\ \cdot \left| \frac{D(x_1, x_2, \dots, x_n)}{D(s_1, s_2, \dots, s_n)} \right| ds_1 ds_2 \dots ds_n .$$

Hence, the following equation can be obtained,

$$\int \cdots \int f(x_1, x_2, \dots, x_n) g(u_1 - x_1, u_2 - x_2, \dots, u_n - x_n) dx_1 dx_2 \dots dx_n \\ = \int \cdots \int F(s_1, s_2, \dots, s_n) G(u_1 + \lambda u_n \cos \alpha - s_1 ,$$

$$\begin{aligned}
& u_2 + \lambda u_n \cos \alpha - s_2 u_3 - s_3, \dots, u_{n-1} - s_{n-1}) ds_1 ds_2 \dots ds_n \\
& = K \int \dots \int F(s_1, s_2, \dots, s_{n-1}) G(u_1 + \lambda u_n \cos \alpha - s_1, \\
& \quad u_2 + \lambda u_n \cos \alpha - s_2 u_3 - s_3, \dots, u_{n-1} - s_{n-1}) ds_1 ds_2 \dots ds_{n-1}.
\end{aligned}$$

where $K = (x_n^{\max} - x_n^{\min})$, and x_n^{\max} and x_n^{\min} are the upper and lower bounds of x_n , respectively. Further, x_n satisfies the condition that for any $x_n \notin [x_n^{\min}, x_n^{\max}]$, $f(x_1, x_2, \dots, x_n)g(u_1 - x_1, u_2 - x_2, \dots, u_n - x_n) = 0$. According to this, we can obtain the following:

Theorem 3.1.

$$\begin{aligned}
& f(x_1, x_2, \dots, x_n) * g(x_1, x_2, \dots, x_n)(u_1, u_2, \dots, u_n) \\
& = K \cdot [F(x_1, x_2, \dots, x_{n-1}) * G(x_1, x_2, \dots, x_{n-1})] (u_1 + \lambda u_n \cos \alpha, \\
& \quad u_2 + \lambda u_n \sin \alpha, u_3, \dots, u_n)
\end{aligned}$$

where K is a variable determined by f and g ,

$$\begin{aligned}
F(x_1, x_2, \dots, x_{n-1}) &= f(x_1 - \lambda X_n(x_1, x_2) \cos \alpha, \\
& \quad x_2 - \lambda X_n(x_1, x_2) \sin \alpha, x_3, \dots, x_{n-1}, X_n(x_1, x_2)) \\
G(x_1, x_2, \dots, x_{n-1}) &= g(x_1 - \lambda X_n(x_1, x_2) \cos \alpha, \\
& \quad x_2 - \lambda X_n(x_1, x_2) \sin \alpha, x_3, \dots, x_{n-1}, X_n(x_1, x_2)) \\
X_n(x_1, x_2) &= \frac{\sqrt{(x_1 - \frac{i}{j})^2 + (x_2 - \frac{m}{n})^2}}{\lambda}
\end{aligned}$$

where i, j, m , and n are integers that satisfy the equation

$$\frac{x_2 - \frac{m}{n}}{x_1 - \frac{i}{j}} = \tan \alpha \quad (\tan \alpha \text{ is an irrational number}).$$

This theorem provides us with a way to reduce the n -dimensional image to two dimensions. According to this theorem, the high-dimensional image is reduced to a low-dimensional image, and the low-dimensional image is then processed by the corresponding method.

According to the above analysis, we classify image recognition into the symplectic group frame, study the symplectic group classifier method of image recognition, and establish the independent model in feature extraction of the image. For a 2-D space image, the matrix representation of the corresponding image can be given according to Appendix 17.4. Assuming that the training set formed by the feature extraction of the different images is $F = \{F^1, F^2, \dots, F^M\}$, $F^1 \in R^N$, the matrix $T = [F^1, F^2, \dots, F^M]^T$ may be formed, and a special form of the symplectic matrix can be obtained by transformation. If the form of the image matrix is different, different images can be distinguished according to different symplectic matrices to achieve the purpose of classification. By n statistically independent source images $S = (S^1, S^2, \dots, S^n)^T$, $T = AS$ is obtained by

linear mixing, where S is the matrix of the source image set, each row vector represents a source image, and A is a mixed matrix. The goal is to find the recovery matrix W and let $WT = WAS = U$, so that we obtain the output matrix that enables the independent source image to be reproduced: $U = WT = WAS$. After U is obtained, the feature space is constructed using its row vectors as feature vectors, and each image is represented by a linear combination of the independent images. Let F^P be the image to be recognised. We have $F^P = \alpha_1 U_1 + \alpha_2 U_2 + \cdots + \alpha_n U_n$, where $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$ is the combination coefficient of the image in the feature space. We use the formula $\alpha = (F^P)^T \times (U)^+$, where $(U)^+$ denotes the pseudo-inverse of U , $(U)^+ = U(UU^T)^{-1}$. Further, with the coordinates of the new feature space, we can classify the image using the classifier.

A 2-D source image A can be constructed as a corresponding symplectic matrix by nature (see Appendix 17.4). Let B be the image to be detected, and use the same method of processing the source image. We extract the image B and then construct the symplectic matrix of the corresponding detection image. Since the representation matrix of the images A and B are different under different octyl groups, from the transformation properties of the conformal symmetric standard matrix, we can obtain the matrices A_1 and B_1 composed of A and B as the symplectic matrix in the form of $\begin{pmatrix} A_1 & O \\ O & B_1 \end{pmatrix}$, and simplify the matrix $\begin{pmatrix} A_1 & O \\ O & B_1 \end{pmatrix}$ by the simplification method of the standard contract matrix. Further, the number of special forms of the existing symplectic matrices is described by Feng et al. [1]. If the symplectic matrix is in a special form, it can be used as the training example of the learning process. The other image should be processed by the same method to achieve the purpose of classification. Consider the special form of symplectic group,

$$M = \begin{pmatrix} A_1 & O \\ O & B_1 \end{pmatrix} \in \mathrm{Sp}(2n).$$

We can get

$$M = \begin{pmatrix} A_1 & O \\ O & A_1^{-1'} \end{pmatrix}, A_1 \in \mathrm{GL}(n, R)$$

where A_1 is the matrix of the source image, which is also calculated

$$|M| = |A_1| |A_1'|^{-1} = 1.$$

Here, we can see that

$$A_1 \in \mathrm{GL}(n, R) \Rightarrow A_1 = TP, T' T = I_n, P' = P > 0 \quad (\text{Positive definite matrix}).$$

Through calculation, we can see that

$$\begin{bmatrix} p \\ q \end{bmatrix} \in R^{2n} \rightarrow M \begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} A_1 p \\ (A_1^{-1})' q \end{bmatrix} \in R^{2n}.$$

The composition matrix is transform processed to get

$$\begin{bmatrix} A_1 & O \\ O & A_1^{-1'} \end{bmatrix} = \begin{bmatrix} T & O \\ O & T \end{bmatrix} \begin{bmatrix} P & O \\ O & P^{-1} \end{bmatrix}$$

where

$$\begin{aligned} \begin{bmatrix} T & O \\ O & T \end{bmatrix} &\in O(n) \subset U(n)O(n) = \left\{ \begin{bmatrix} T & O \\ O & T \end{bmatrix} \mid T'T = I_n \right\} \\ &= U(n) \cap \text{GL}(n, R) \left\{ \begin{bmatrix} P & O \\ O & P^{-1} \end{bmatrix} \mid P' = P > 0 \right\}. \end{aligned}$$

We can see that a symplectic matrix can be decomposed into the product of two symplectic matrices, and the resulting symplectic matrix can be used as the matrix of the detection results. If the matrix formed by the other image to be examined and the symplectic matrix obtained by conformal transformation of source image A are the same as the form of the symplectic matrix, then the image to be examined and the source image are consistent, thus achieving the purpose of classification.

3.2.2 Design method of symplectic group classifier

In the design of the symplectic classifier, first, the classification samples should be expressed as symplectic groups, and then, the samples are constructed into corresponding symplectic matrices by using the tensor representation of symplectic groups. The symplectic matrix is simplified, and the corresponding training features are obtained, which are used as training examples for the learning process. Finally, the detection samples are processed in the same way, and the sample features are obtained by using the symplectic group classification algorithm. If the two types of features match, the result is output. The design method of the symplectic group classifier can be divided into two steps: the symplectic matrix representation of the sample and the simplification method of the matrix under the symplectic space. The sample is expressed in the tensor form of the symplectic group; then, the symplectic matrix representation of the sample is given. The next point is to simplify the symplectic matrix.

3.2.2.1 Simplification method of symplectic matrix

Symplectic geometry is the geometry of phase space. The symplectic space (i.e., the phase space) has a specific symplectic structure, depending on a bilinear antisymmetric non-degenerate inner product, i.e., the inner product $[x, y] = \langle x, Jy \rangle$, where

$$J = J_{2n} = \begin{bmatrix} 0 & +I_n \\ -I_n & 0 \end{bmatrix}.$$

When $n = 1$,

$$[x, y] = \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix}.$$

Thus, in contrast to the Euclidean space, the measure in the symplectic space is the measure of the area rather than the length, and there is an orthogonal concept in the symplectic space, which is the essential difference between the symplectic space and the Euclidean space. Since there is no non-degenerate antisymmetric array in the classical sense in the odd dimension, the symplectic space here is even-dimensional. The symplectic transformation in symplectic geometry is essentially a nonlinear transformation (also called a regular transformation), which is suitable for nonlinear system processing. Its eigenvalues can be obtained by the symplectic QR method. Symplectic QR is different from the standard QR decomposition. Its floating point operation is the original quarter, and its original $2n$ -dimensional space can be transformed into an n -dimensional space.

Theorem 3.2. Any matrix can be constructed as a Hamiltonian matrix. For matrix A ,

$$\begin{vmatrix} A & 0 \\ 0 & -A^H \end{vmatrix} \text{ is the Hamiltonian matrix [22, 23].}$$

Theorem 3.3. Hamiltonian matrices are invariant under symplectic similarity transformations [22, 23].

Theorem 3.4. Let $H \in C^{2n \times 2n}$ be a Hamiltonian matrix. Then, e^H is a symplectic matrix [2, 22, 23].

Theorem 3.5. Let $M = \begin{bmatrix} A_1 & B_1 \\ A_2 & B_2 \end{bmatrix} \in \mathrm{Sp}(2n)$. Then, M can be decomposed into $M = M_{2'} M_1 M_2$, where $M_{2'} \in \mathrm{Sp}(2n)$, $M_1 \in \mathrm{Sp}_{2n}(2n)$, $M_2 \in \mathrm{Sp}_{2n}(2n)$.

Proof.

$$\begin{aligned} \begin{bmatrix} A_1 & B_1 \\ A_2 & B_2 \end{bmatrix} \begin{bmatrix} I & -A_1^{-1}B_1 \\ O & I \end{bmatrix} &= \begin{bmatrix} A_1 & O \\ A_2 & B_2 - A_2 A_1^{-1} B_1 \end{bmatrix}, \\ \begin{bmatrix} I & O \\ -A_2 A_1^{-1} & I \end{bmatrix} \begin{bmatrix} A_1 & O \\ A_2 & B_2 - A_2 A_1^{-1} B_1 \end{bmatrix} &= \begin{bmatrix} A_1 & O \\ O & B_2 - A_2 A_1^{-1} B_1 \end{bmatrix}. \end{aligned}$$

Because $M = [A, B] \in \mathrm{Sp}(2n)$, $A'JB = I$ and A' are symmetrical, i.e.,

$$A'_1 B_2 - A'_2 B_1 = I, A_2 A_1^{-1} \in \mathrm{Sp}(2n).$$

Hence,

$$A_1^{-1'} = B_2 - A_1^{-1} A'_2 B_1 = B_2 - (A_2 A_1^{-1})' B_1 = B_2 - A_2 A_1^{-1} B_1.$$

Let

$$\begin{aligned} M_1 &= \begin{bmatrix} A_1 & O \\ O & B_2 - A_2 A_1^{-1} B_1 \end{bmatrix} = \begin{bmatrix} A_1 & O \\ O & A_1^{-1'} \end{bmatrix} \in \mathrm{Sp}_{2n}(2n) \\ M_{2'} &= \begin{bmatrix} I & O \\ -A_2 A_1^{-1} & I \end{bmatrix} \in \mathrm{Sp}(2n); \quad M_2 = \begin{bmatrix} I & -A_1^{-1} B_1 \\ O & I \end{bmatrix}^{-1} \in \mathrm{Sp}(2n); \end{aligned}$$

hence, we can get $M = M_{2'} M_1 M_2$.

□

Theorem 3.6. Any symplectic matrix can be decomposed into the product of three special symplectic matrices [22, 23]:

$$M = M_0 M_1 M_2 , \quad M_0, M_1, M_2 \in \mathrm{Sp}(2n) .$$

Proof. Let $M = [A, B] \in \mathrm{Sp}(2n)$. Then, $A \in U(n)$. According to the relevant theorem, the existence of $M_0 \in \mathrm{Sp}(2n) = U(n)$ makes

$$A = M_0 \begin{bmatrix} I \\ O \end{bmatrix} Q = M_0 \begin{bmatrix} Q \\ O \end{bmatrix} = M_0 \begin{bmatrix} Q & O \\ O & Q'^{-1} \end{bmatrix} \begin{bmatrix} I \\ O \end{bmatrix} .$$

Let

$$B_1 = \begin{bmatrix} Q & O \\ O & Q'^{-1} \end{bmatrix} M_0^{-1} B ,$$

i.e.,

$$B = M_0 \begin{bmatrix} Q & O \\ O & Q'^{-1} \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} .$$

Hence, we have

$$\begin{aligned} M = [A, B] &= \left[M_0 \begin{bmatrix} Q & O \\ O & Q'^{-1} \end{bmatrix} \begin{bmatrix} I \\ O \end{bmatrix}, M_0 \begin{bmatrix} Q & O \\ O & Q'^{-1} \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \right] \\ &= M_0 \begin{bmatrix} Q & O \\ O & Q'^{-1} \end{bmatrix} \begin{bmatrix} I & C_1 \\ O & C_2 \end{bmatrix} \in \mathrm{Sp}(2n) \\ &= M_0 M_1 M_2 \end{aligned}$$

where

$$M_1 = \begin{bmatrix} Q & O \\ O & Q'^{-1} \end{bmatrix} \in \mathrm{Sp}(2n), M_2 = \begin{bmatrix} I & C_1 \\ O & C_2 \end{bmatrix} \in \mathrm{Sp}(2n) . \quad \square$$

Theorem 3.7. Let $S \in \mathcal{C}^{2n \times 2n}$ be a symplectic matrix. Then, decompose $S = QR$, where Q is the symplectic unitary matrix and R is the upper triangular matrix [22, 23].

Theorem 3.8. The product of two symplectic matrices is a symplectic matrix [22, 23].

For the Hamiltonian matrix M , we can use symplectic similarity transformation to identify the eigenvalues of the system. Moreover, the original $2n$ -dimensional space can be transformed into an n -dimensional space. The specific steps are as follows:

- (1) Let $N = M^2 = \begin{bmatrix} A^T & G \\ F & -A \end{bmatrix}$
- (2) Find a symplectic matrix Q and let $Q^T N Q = \begin{bmatrix} B & R \\ 0 & B^T \end{bmatrix}$, where B is the upper Hessenberg matrix ($b_{ij} = 0$, $i > j + 1$), and Q is the Householder matrix H .
- (3) Calculate $\lambda(B) = \{\mu_1, \dots, \mu_n\}$ by the symplectic QR decomposition method.
- (4) $\lambda_1 = \sqrt{\mu_1}, \lambda_{n+i} = -\lambda_i$.

Theorem 3.9. Let the Householder matrix H be

$$H = H(k, \mu) = \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix}$$

where

$$P = I_n - \frac{2UU^*}{U^*U}, \quad U = (0, \dots, 0; u_k, \dots, u_n)^T \neq 0.$$

Then, H is a symplectic unitary matrix and U^* is the conjugate transpose of U .

Proof. To prove that H is a symplectic matrix, we only need to prove that $H^*JH = J$ is true.

$$H^*JH = \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix}^* J \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix} = \begin{bmatrix} 0 & P^*P \\ -P^*P & 0 \end{bmatrix}. \quad (3.3)$$

Because $P = I_n - 2UU^*/U^*U$, $P^* = P$ and

$$\begin{aligned} P^*P &= P^2 = \left(I_n - \frac{2UU^*}{U^*U}\right)\left(I_n - \frac{2UU^*}{U^*U}\right) \\ &= I_n - \frac{4UU^*}{U^*U} + \frac{4U(U^*U)U^*}{(U^*U)(U^*U)} = I_n \end{aligned} \quad (3.4)$$

where $U = (0, \dots, 0; u_k, \dots, u_n)^T \neq 0$. Substituting (3.4) into (3.3) gives

$$H^*JH = J.$$

Therefore, H is a symplectic matrix.

$$H^*H = \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix}^* \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix} = \begin{bmatrix} P^*P & 0 \\ 0 & P^*P \end{bmatrix}.$$

Hence, H is not only a unitary matrix but also a symplectic unitary matrix.

Let X be an $m \times n$ -dimensional real matrix and $A = XX'$ be a real symmetric matrix. M can be constructed as a Hamiltonian matrix:

$$M = \begin{bmatrix} A & 0 \\ 0 & -A' \end{bmatrix}.$$

Then, there is a Householder matrix H , which makes HMH' an upper Hessenberg matrix

$$HMH' = \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & -A' \end{bmatrix} \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix} = \begin{bmatrix} PAP' & 0 \\ 0 & -PA'P' \end{bmatrix} = \begin{bmatrix} B & 0 \\ 0 & -B' \end{bmatrix}.$$

Therefore, the eigenvalue $\lambda(A)$ of A is the eigenvalue $\lambda(B)$ of B , and the eigenvalue of X is $\sqrt{\lambda A}$, such that the original $2n$ -dimensional space is transformed into an n -dimensional space. In fact, for any real matrix X , the orthogonal matrix P can be found directly by the Householder method such that PXP' is the upper Hessenberg matrix, and the final principal component spectrum of X is composed of the eigenvalues in

descending order, i.e., $\sigma_1 = \lambda_{\max}, \dots, \sigma_n = \lambda_{\min}$ (n is the dimension of X). Let vector $\omega = (\omega_1, \omega_2, \dots, \omega_n)^T$ satisfy $\|\omega\|_2 = 1$. Then, matrix

$$H(\omega) = 1 - 2\omega\omega^T = \begin{bmatrix} 1 - 2\omega_1^2 & -2\omega_1\omega_2 & \dots & -2\omega_1\omega_n \\ -2\omega_2\omega_1 & 1 - 2\omega_2^2 & \dots & -2\omega_2\omega_n \\ \vdots & \vdots & \vdots & \vdots \\ -2\omega_n\omega_1 & -2\omega_n\omega_2 & \dots & 1 - 2\omega_n^2 \end{bmatrix}.$$

The matrix is an elementary reflection matrix. It is easy to prove that the elementary reflection matrix H is a symmetric matrix ($H^T = H$), an orthogonal matrix ($H^T H = 1$), and an involution matrix ($H^2 = 1$). \square

Theorem 3.10. *Let x and y be two unequal n -dimensional vectors, and $\|x\|_2 = \|y\|_2$. Then, the primary reflection matrix $H = 1 - \omega\omega^T$ can make $Hx = y$, where $\omega = x - y/\|x - y\|_2$ [22].*

From Theorem 3.10, it is not difficult to see that for a nonzero n -dimensional vector $x = (x_1, \dots, x_n)^T$, where $\alpha = \|x\|_2$, the elementary reflection array $H = 1 - \omega\omega^T$ can make $Hx = \alpha e_1$, where $e_1 = (1, 0, \dots, 0)^T$, $\rho = \|x - \alpha e_1\|_2$.

The following describes how the real matrix A seeks the orthogonal matrix H so that HMH^T is the upper triangular matrix. Let

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & A_{12}^{(1)} \\ \lambda_{21}^{(1)} & A_{22}^{(1)} \end{bmatrix}.$$

In the first step, we may wish to set $\lambda_{21}^{(1)} \neq 0$. Otherwise, this step will not be reduced. Set the first column vector of A as

$$V^{(1)} = (a_{11}^{(1)}, a_{21}^{(1)}, \dots, a_{n1}^{(1)})^T = (a_{11}, a_{21}, \dots, a_{n1}).$$

Select the elementary reflection matrix $H^{(1)}$ as

$$H^{(1)} = I - 2\omega^{(1)}(\omega^{(1)})^T$$

where

$$\begin{cases} \lambda_1 = \|V^{(1)}\|_2 \\ E^{(1)} = (1, 0, \dots, 0)^T \\ \rho_1 = \|V^{(1)} - \lambda_1 E^{(1)}\| \\ \omega^{(1)} = \frac{(V^{(1)} - \lambda_1 E^{(1)})}{\rho_1}. \end{cases}$$

Then, with A after $H^{(1)}$ transformation, we get

$$H^{(1)}A = \begin{bmatrix} \sigma_1 & a_{12}^{(2)} & \dots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{bmatrix} = A^{(2)}.$$

In the second step, the same method is used to transform the second column vector of $A^{(2)}$ and set the vector

$$V^{(2)} = \left(0, a_{22}^{(2)}, \dots, a_{n2}^{(2)} \right)^T.$$

Construct the matrix $H^{(2)}$ as

$$H^{(2)} = I - 2\omega^{(2)}(\omega^{(2)})^T$$

where

$$\begin{cases} \lambda_2 = \|V^{(2)}\|_2 \\ E^{(2)} = (0, 1, \dots, 0)^T \\ \rho_{(2)} = \|V^{(2)} - \lambda_2 E^{(2)}\| \\ \omega^{(2)} = (V^{(2)} - \lambda_2 E^{(2)})/\rho_{(2)}. \end{cases}$$

For $H^{(2)}$, the second column vector can be changed to 0 vectors except 1 and 2 lines, i.e.,

$$H^{(2)} A^{(2)} = A^{(2)}$$

and so on. Repeat this until $A^{(n)}$ becomes the upper triangular matrix such that

$$H = H^{(n)} H^{(n-1)} \dots H^{(1)}.$$

Then, A becomes an $n \times n$ -dimensional upper triangular matrix

$$\begin{bmatrix} \sigma_1 & * & \dots & * \\ 0 & \sigma_2 & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n \end{bmatrix}.$$

The symplectic algorithm of the simplified matrix can be described as follows:

- (1) In the original sequence, the mean is removed and normalised to unit variance.
- (2) According to the minimum embedding dimension d_1 , the orbital matrix X and the real symmetric matrix A are constructed. The upper triangular matrix is calculated.
- (3) The embedding dimension is increased to d_2 , a new matrix is constructed, and the d_2 -dimensional symplectic geometry is mapped into d_1 normalised symplectic principal components. Finally, the average values of the symplectic geometry of K with different embedding dimensions are obtained.
- (4) Repeat steps 2 and 3 for the surrogate data for the original sequence.
- (5) Identify the determinacy of the sequence based on the test statistic.

The alternative data adopt the Fourier transform method of iterative amplitude adjustment. The symplectic algorithm is used to construct the symplectic matrix, and the symplectic matrix is reduced to the upper triangular matrix, so that the original sequence can be transformed into an upper triangular matrix.

According to the above steps, we can establish the design process of the symplectic group classifier. Further, we give the corresponding symplectic group classifier design step for the face recognition process and the data set classification process.

3.2.3 Design of symplectic group classifier in face recognition

Normally, the collected face image is represented by a matrix [24–27]. For greyscale images, the element values of the matrix are the grey values of the image pixels. At present, face feature extraction and recognition methods usually take the grey face image sample matrix as rows or columns as vectors, so that a higher-dimensional vector sample can be obtained. The dimension of the sample is much greater than the number of samples of available face images. In the first step of this section, we use the symplectic group matrix to construct the singular value of the corresponding image in the symplectic matrix space, and then we construct the feature subspace of the image in the symplectic matrix space. Singular value decomposition is used to decompose the image into a linear combination of the base image with rank 1. The second step is to use the symplectic classifier algorithm to calculate the projection length of the identified samples in each feature subspace, and then classify the sample into a class of characteristic subspaces corresponding to the maximum projection length. In the third step, the nearest feature line method is extended to the matrix space, the feature lines are constructed directly in the symplectic matrix space by using two different face image matrices of each category, and the ability of the two samples to characterise the respective categories is expanded. The nearest neighbour characteristic line method is extended to the symplectic matrix space. When the symplectic matrix is transformed, the transformed symplectic matrix and several special symplectic matrices can be compared by referring to the transformation properties of the conformal symmetric standard matrix. If the symplectic matrix conforms to a particular form, the matrix is used as a training example in the learning process, which can be decomposed into the product of two matrices (here, we can get the resulting matrix as the matrix of the test results). If the matrix of the other image to be examined and the matrix obtained by the transcoding of the source image A have the same form as the symplectic matrix, the image to be examined and the source image are consistent, i.e., the purpose of recognition is achieved. This method can directly deal with the original face image matrix, and it is not necessary to straighten it into a high-dimension vector. Therefore, the computational complexity of high-dimensional vector extraction is avoided, and the problem of face recognition is solved using the symplectic group classifier.

First, we define the inner product of two complex symplectic matrices, and let the matrix $A, B \in C^{m \times n}, C^{m \times n}$ be a recursive space of $m \times n$ dimensionality. Let $X \in C^{m \times n}$ and the rank of X be r . Then, there are m orders of the symplectic identity matrix U and n orders of the symplectic identity matrix V , giving

$$U^H X V = \begin{bmatrix} \Lambda & 0 \\ 0 & 0 \end{bmatrix}$$

where $\Lambda = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r)$, $\sigma_i (i = 1, 2, \dots, r)$ is the nonzero singular value of the matrix X arranged in descending order. The feature vector of XX^H is obtained by

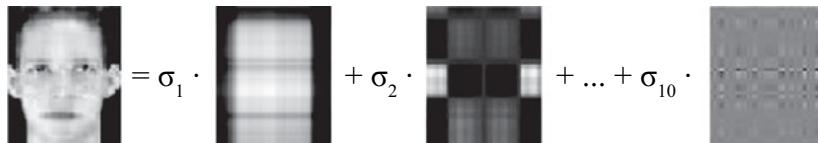
the column vector transformed by symplectic identity matrix U , and the feature vector of $X^H X$ is obtained by the column vector transformed by symplectic identity matrix V . Singular value decomposition is also applicable to the case where the matrix elements are complex numbers. The symplectic matrix elements of the image are real numbers, and the elements of the transformation matrix U and V are also real numbers [28]. By the singular value decomposition theorem, the image matrix X can be decomposed into the sum of r matrices of rank 1.

$$X = U \begin{bmatrix} \Lambda & 0 \\ 0 & 0 \end{bmatrix} V^H = \sum_{i=1}^r \sigma_i u_i v_i^H = \sum_{i=1}^r \sigma_i A_i$$

where $A_i = u_i v_i^H$ ($i = 1, 2, \dots, r$), whose rank is 1, and u_i and v_i are the column vectors of the corresponding singular value σ_i for the symplectic unitary matrices U and V respectively. Further, u_i and v_i are orthogonal to each other, and the inner product of the symplectic matrix can be obtained from the inner product of the matrix. It is easy to see that the base images obtained by the singular value decomposition are orthogonal to each other under the inner product of the defined symplectic matrix, which constitutes a standard orthogonal coordinate system in the $R^{m \times n}$ -dimensional symplectic matrix space. The original image is reconstructed using the first k largest singular values in the above equation and its corresponding base image, and the reconstructed image is given by

$$\hat{X} = \sum_{i=1}^k \sigma_i A_i .$$

The singular value σ_i of X is obtained using the standard orthogonality of the base image, i.e., the projection value of X on the base image A_j obtained by its decomposition, as the coordinate value of X in the symplectic orthogonal coordinate system formed by A_i . The original face image reconstructed by the base image can be obtained by the singular value.



According to the image X , we can get the base image A_i of its category. The image to be identified, i.e., image X_1 , is regarded as a linear combination of the base image A_i , and the combination expression is

$$X_1 = \sum_{i=1}^r a_i A_i .$$

The combination coefficient a_i in the formula is a projection value of X_1 on the base image A_i . It can be calculated using the inner product of the symplectic matrix defined

above.

$$\langle X_1, A_j \rangle = \left\langle \sum_{i=1}^r a_i A_i, A_j \right\rangle = a_j \quad (j = 1, 2, \dots, r).$$

Similarly, the projection a_i of X_1 on A_i can be used to calculate the projection length of X_1 in the symplectic feature subspaces composed of A_i

$$l_1 = \sqrt{\sum_{i=1}^r a_i^2}.$$

This projection length measures the proximity of the symplectic feature subspace composed of images X_1 and A_i , which are to be recognised. The longer the projection length, the shorter is the distance between X_1 and the symplectic feature subspace composed of A_i . Therefore, if the projection length is greater than the projection length of X_1 in the other class symplectic feature subspace, X_1 can be attributed to the category corresponding to the symplectic feature subspace of A_i .

If there are multiple training sample images in each class, the singular value of each training sample image can be decomposed separately, and the symplectic feature subspaces of their category are constructed. The projection length of the sample to be identified into each symplectic feature subspace is calculated. In the process of identification, the symplectic group classifier method is used to classify the sample to be identified into the category of the symplectic feature subspace with the largest projection length.

Next, we present the recognition process of the symplectic group classifier in the process of image classification:

- (1) In the training phase, perform singular value decomposition on the i -th training sample X_{ji} of the j -th class. Take the base image $A_{ji}(m) = u_{ji}^m(v_{ji}^m)^H$ ($m = 1, 2, \dots, k$) corresponding to the k largest singular values. A set of base images obtained from the training samples X_{ji} constitutes a discriminant function $g_{ji}(X)$ of the j -th class.

$$g_{ji}(X) = \sqrt{\sum_{m=1}^k \alpha_{ji}^2(m)} = \sqrt{\sum_{m=1}^k |\langle A_{ji}(m), X \rangle|^2} \quad j = 1, 2, \dots, C, \quad i = 1, 2, \dots, N$$

where $\alpha_{ji}(m) = \langle A_{ji}(m), X \rangle$ is the projection value of the m th base image $A_{ji}(m)$ of the i -th training sample X_{ji} to be recognised by the sample image X for the j th class.

- (2) The classifier recognition process is to input the image X to be recognised into each discriminant function $g_{ji}(X)$ and to find the output of the discriminant function.

Put X in the corresponding category of $g_{ji}(X)$ of the maximum output. If each class has multiple training samples, first calculate the maximum length for each class

$$h_j = \max_i(g_{ji}(X)) \quad (j = 1, 2, \dots, C, \quad i = 1, 2, \dots, N).$$

This is so that X can be classified into the c th category.

Design steps of symplectic group classifier:

Step 1: The extracted feature points on the source image form the sequence $S(a_1, \dots, a_n)$.

Step 2: Obtain the matrix S with the characteristic sequence as the row vector.

Step 3: The extracted feature points on the image to be checked form the sequence $T(a_1, \dots, a_n)$.

Step 4: Obtain the matrix T with the characteristic sequence as the row vector.

Step 5: Perform singular value decomposition on T and S , respectively, to find their singular values.

Step 6: Combine the eigenvalues of T and S to form matrices U , such as

$$\begin{pmatrix} A_1 & O \\ O & B_1 \end{pmatrix}.$$

Step 7: According to the classification recognition process, the various categories of the symplectic matrix are obtained.

Step 8: Obtain the contract standard of the conformal symmetric matrix of U .

Step 9: Get the final matrix form W .

Step 10: Compare the resulting W form with the special form of the symplectic group, and form the corresponding symplectic group classifier.

Step 11: Use the resulting symplectic group classifier to classify other images.

We can transform the matrix of the source image and the image to be examined and compare it with several forms of the above symplectic groups to form a symplectic group classifier in Lie group machine learning.

3.2.4 Symplectic group classifier design for classification process of data set

For the classification of data sets, data preprocessing is first applied to the classified data sets. The quality of raw data is the direct factor that affects the classification results. Therefore, understanding the inherent characteristics of the data plays an important role in the design of classification algorithms. The data characteristics mainly include the following aspects:

- (1) Data diversity: the data sets to be classified are generally derived from multiple data sources. In addition, data describing the same feature may have different names in different data sets, so it is necessary to understand the diversity of the data.
- (2) Data redundancy: the data set to be classified has a large amount of redundant data, e.g., the data in the data set has repetition or similarity, and the attributes of the data objects are the same, which can be eliminated by data cleansing or reduction.
- (3) High dimensionality of data: the data sets of general classification are classified in a high-dimensional space, so it is necessary to first understand the dimension

of the data set. In the design of the symplectic group classifier, owing to the particularity of the symplectic group, the data set to be classified must be even-dimensional data. Only in this way can we use the symplectic groupifier algorithm to classify it.

For the data set classification process, the data set is first pre-processed. The pre-processing takes the feature extraction of the data in the data set and extracts the feature sets that can be used for classification, which can improve the efficiency of the data classification process. Then, the data in the data set is integrated to form a unified data set, including data pattern matching and integration, data conflict or inconsistency processing, redundancy elimination, and other steps, which serve as the foundation for data classification. The general data classification process consists of two steps: first, in a given data set, find a common set of objects to establish a classification model, and then use the obtained model for classification; the data items are mapped to one of the classes. To establish the classification model, a part of the data set X is generally used as the training sample set X_e and the remaining part of it is used as the test sample set X_t . All the samples have a known class identifier. The classification algorithm forms a feature description by analysing X_e and establishes a classification model. Further, X_t is used to test the model classification accuracy; if the obtained accuracy is acceptable, the corresponding classification model can be used to classify future data [29].

According to the above process, this section describes the design process of the symplectic group classifier in data set classification. In the feature extraction stage, the data set to be classified is represented in the form of a symplectic group $\text{Sp}(2n)$. Special attention is paid to the fact that the data in the data set in the form of the symplectic group must be even-numbered; otherwise, it cannot be expressed as a symplectic group. The $2n$ -dimensional data is represented as a sequence of $(x_1, x_2, \dots, x_{2n})$, and the sequence is then divided into two n -dimensional vectors, $u = [x_1, \dots, x_n]^T$, $v = [y_1, \dots, y_n]^T$. Finally, two vectors are constructed into the symplectic inner product:

$$\omega(u, v) = u^T J v^t$$

where

$$J = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}.$$

Here, the $2n$ -dimensional data set is represented in a symplectic space. In this symplectic space, for the symplectic matrices consisting of two vectors in the data set, we use the matrix simplification method described in Section 3.2.1 and the method of symplectic transformation to reduce the symplectic matrix and obtain the singular eigenvalues of the matrices. The resulting singular values are arranged in the singular vector $[\lambda_1, \lambda_2, \dots, \lambda, 0, \dots, 0]^T$ in descending order. This vector is used as the singu-

lar eigenvector of the original data set. In the process of the symplectic group classifier design, the vector of the data set forms the vector space in the symplectic space, and the vector space is used to construct the symplectic group. The following special types of symplectic groups can be used as different types of classification categories.

Let $M = [A, B] = \begin{pmatrix} A_1 & B_1 \\ A_2 & B_2 \end{pmatrix} \in \mathrm{Sp}(2n)$. We have

$$M \in \mathrm{Sp}(2n) \cap O_{2n} = U(n)$$

$$\Rightarrow M = [A, J^{-1}A], A'JA = O_n, A'A = I_n \quad (\text{As M(1)})$$

$$M = \begin{pmatrix} A_1 & O \\ O & B_1 \end{pmatrix} \in \mathrm{Sp}(2n)$$

$$\Rightarrow M = \begin{pmatrix} A_1 & O \\ O & A_1^{-1'} \end{pmatrix}, A_1 \in \mathrm{GL}(n, R) \quad (\text{As M(2)})$$

$$M = \begin{pmatrix} I & O \\ O & B_2 \end{pmatrix} \in \mathrm{Sp}(2n)$$

$$\Rightarrow M = \begin{pmatrix} I & O \\ O & I \end{pmatrix}, S' = S \quad (\text{As M(3)})$$

$$M = \begin{pmatrix} A_1 & O \\ A_2 & I \end{pmatrix} \in \mathrm{Sp}(2n)$$

$$\Rightarrow M = \begin{pmatrix} I & O \\ O & I \end{pmatrix}, S' = S \quad (\text{As M(4)})$$

$$M = J_a = \begin{pmatrix} I\hat{\alpha} & I\hat{\alpha} \\ -I\hat{\alpha} & I\hat{\alpha} \end{pmatrix}, \alpha \subset v \in \mathrm{Sp}(2n) \quad (\text{As M(5)}) .$$

3.3 Symplectic group classifier algorithm in Lie group machine learning

This section is based on the design process of the symplectic group classifier. It presents the symplectic group classifier algorithm [30, 31] and validates the algorithm with an example. The results show that the symplectic group classifier algorithm has obvious advantages in the case of large sample size.

3.3.1 Symplectic group classifier algorithm

According to the design process of the symplectic groupifier and the data set classification process for face recognition given in the previous section, the symplectic groupifier algorithm is given in this section, and it is validated with an example.

3.3.1.1 Symplectic group classifier algorithm in face recognition

We construct the sample matrix of the symplectic group learning problem according to the antisymmetric property of the symplectic matrix and the singular value evaluation of the symplectic matrix. Through different transformations of the symplectic matrices, the corresponding learning methods can be obtained. As the symplectic group has both algebraic and geometric properties, this makes the symplectic group maintain even-dimensionality in the learning process. The symplectic group learning algorithm is described as follows:

Algorithm 3.1: Symplectic group classifier algorithm in face recognition

Input: Sample set D , $\text{Sp}(2n)$ is a symplectic group classification algorithm.

Output: Some special type of symplectic matrices $\{\mathbf{M} (1), \mathbf{M} (2), \mathbf{M} (3), \mathbf{M} (4), \mathbf{M} (5)\}$.

Step 1 The sample data set D is mapped to $\text{Sp}(2n)$, and the n -dimensional row vectors on $\text{Sp}(2n)$ are composed of sets $S_q^n, S_q^n = \{(x_1, x_2, \dots, x_i, \dots, x_n) | x_i \in S_q, i = 1, 2, 3, \dots, n\}$.
// $n = 2v$, v is a positive integer, and vector $(x_1, x_2, \dots, x_i, \dots, x_n)$ represents any one sample.

Step 2 Select $Q_i \in \text{Sp}(2n)$. For Q_i , it is transformed into the corresponding symplectic matrix, and the singular value of the symplectic matrix is obtained.

Step 3 Q_i is applied to the sample data set $((x_1, x_2, \dots, x_i, \dots, x_n), Q_i) \mapsto (x_1, x_2, \dots, x_i, \dots, x_n) Q_i$ as a training column in the learning process.

Step 4 In the training phase, singular value decomposition is performed on the training sample X_{ji} , and the sample $A_{ji}(m) = u_{ji}^m (v_{ji}^m)^H$ ($m = 1, 2, \dots, k$) corresponding to the first k largest singular values is taken. The sample obtained by training sample X_{ji} constitutes a discriminant function of class j $g_{ji}(X)$, $g_{ji}(X) = \sqrt{\sum_{m=1}^k \alpha_{ji}^2(m)} = \sqrt{\sum_{m=1}^k |\langle A_{ji}(m), X \rangle|^2} j = 1, 2, \dots, C, i = 1, 2, \dots, N$, where $\alpha_{ji}(m) = \langle A_{ji}(m), X \rangle$ is the projection value of the m th base image $A_{ji}(m)$ of the i th training sample X_{ji} to be recognised by the sample image X for the j th class.

Step 5 In the sample set D , the sample of the target learning result is deleted, and a new sample set D' is obtained.

Step 6 The discriminant sample X is input to each discriminant function $g_{ji}(X)$, and the output of the discriminant function is given. Put X in the corresponding category of $g_{ji}(X)$ of the maximum output. If there are multiple training samples in each class, the maximum length $h_j = \max_i(g_{ji}(X)) (j = 1, 2, \dots, C, i = 1, 2, \dots, N)$ is calculated for each class so that we can classify X into the c th category.

Step 7 End

Since the symplectic group is even-dimensional, it can be reduced to the symplectic space only if the dimension of the vector space is even. Then, considering the properties of the symplectic group, we choose even numbers of samples, and positive and negative samples are equal or relatively close to maximise the efficiency of learning.

3.3.1.2 Symplectic group classifier algorithm in data set classification

According to the classifier design process in the process of data set classification, the sample to be classified is first expressed as a $2n$ -dimensional vector, and the $2n$ -

dimensional vector is then decomposed into the inner product of two vectors. According to the construction method of symplectic matrices, the matrix of symplectic space is constructed, and the matrices are mapped in a differential manifold, and then classified according to several special symplectic matrices.

Algorithm 3.2: Symplectic group classifier algorithm in data set classification

Input: Data set D .

Output: A special symplectic matrix.

Step 1 Determine whether the data set is even-dimensional. If it is, continue; otherwise, exit.

Step 2 Take the sample D_i in the data set for feature extraction, and obtain the $2n$ -dimensional feature vector. The former n -dimensional vector is used as the training example, and the other n -dimensional vectors are taken as the objective function.

Step 3 A symplectic group matrix is constructed using the eigenvectors of the sample to be classified, and singular value decomposition of the matrix is performed to obtain the eigenvalues of the symplectic matrix.

Step 4 The eigenvalues obtained from the training samples and the eigenvalues of the samples to be identified constitute the symplectic matrix.

Step 5 Compare the symplectic matrix of Step 4 with several special symplectic matrices to achieve classification.

3.3.2 Verification of symplectic group classifier algorithm in face recognition

To verify the correct rate and performance of the symplectic classifier classification, we performed a test on the ORL general face image test library. The experimental method involved the image of a human face as the source image. The source image was pre-processed according to the previous method. Then, different photos of the same person's face were found to form a training example, and this classifier was used to process other images. The ORL face database has been established by the University of Cambridge, and it includes a total of 400 (i.e., $40 * 10$) images, which is a positive image of facial basic naked, except for light, perspective and expression changes slightly. The ORL face database is a simple database. The purpose of this test is to verify the performance of the proposed algorithm classification. Take the face image of a person in the face database, and then classify the image to be tested with the formed symplectic group classifier for the other images. To balance the effects of different training samples on the identification, evaluate the recognition performance of the algorithm. According to the effect given in the above-mentioned face database, divide the face into seven types of gestures, namely the front, the prone, the pitching, the inside of the left plane, the lateral of the left plane, the inside of the right plane and the outside of the right plane. As shown in Figure 3.2, the image of the face is pre-treated according to the analysis, and the feature space of the image is obtained. The proposed method is compared with the existing SVM algorithm (see Figure 3.3 and Figure 3.4).

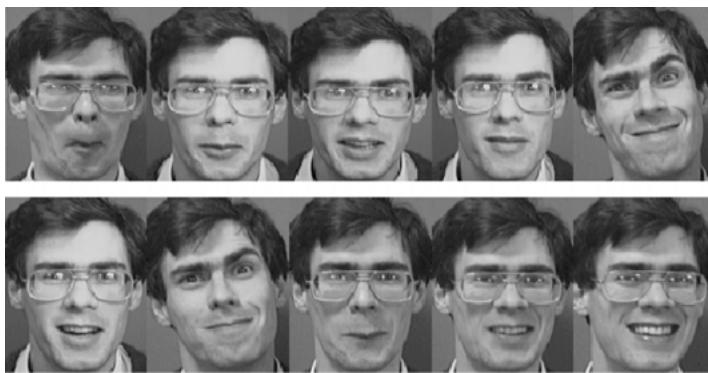


Fig. 3.2: Different facial expressions for the same person.

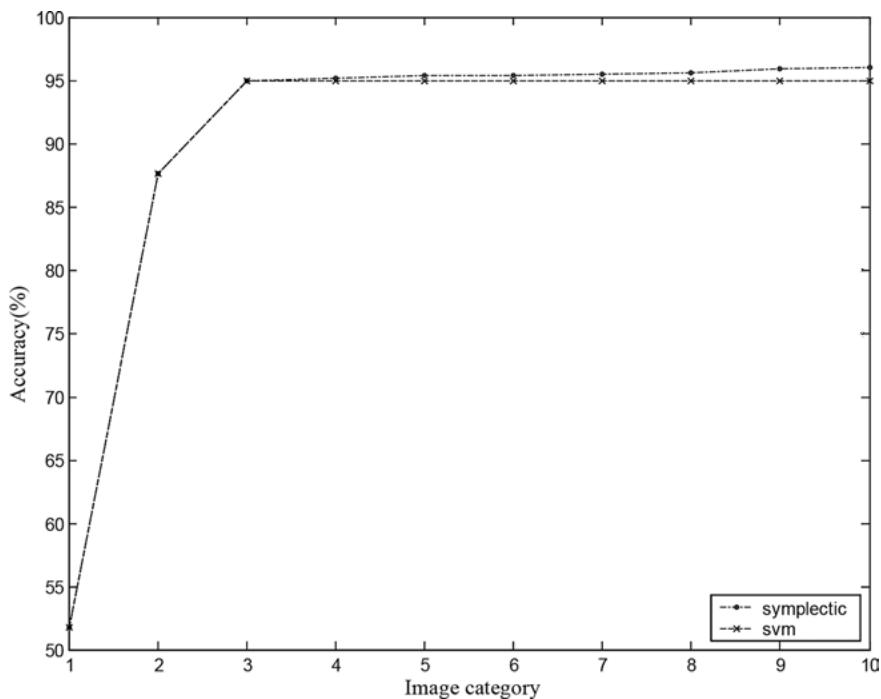


Fig. 3.3: Effect comparison between the symplectic group classifier and the SVM classifier on the frontal face image.

Experiments show that when the face image is extracted by the feature image, the symplectic group classifier algorithm has certain advantages in terms of the classification accuracy when more face images are available. This shows that the symplectic group classifier algorithm has a better classification effect when dealing with a classification problem involving a large number of samples (see Table 3.1).

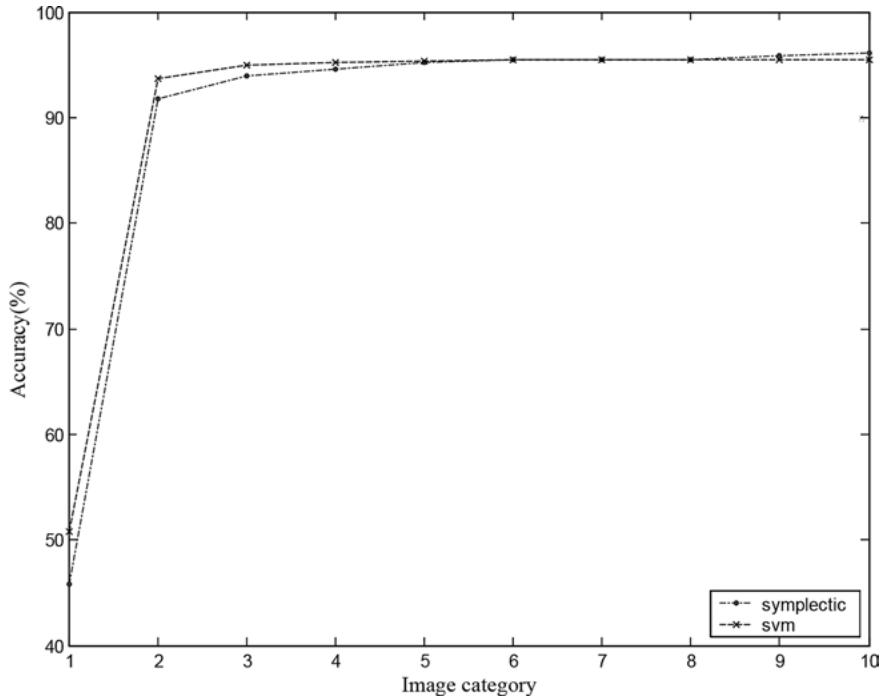


Fig. 3.4: Effect comparison between the symplectic group classifier and the SVM classifier on the side face image.

Tab. 3.1: Comparison between SVM classification average recognition rate and symplectic group classification average recognition rate.

Attitude Algorithm	Frontal image [%]	Top view image [%]	Upward view image [%]	Left plane inner image [%]	Right plane inner image [%]
SVM classification algorithm	95.4	93.8	90.2	93.4	93.4
Symplectic group classification algorithm	95.2	92.6	93.1	94.6	94.6

3.3.3 Verification of symplectic group classifier algorithm in data set classification

To verify the classification performance of the symplectic classification algorithm in the data set classification process, this section selects the *adult* data set in the *UCI* learning library for testing. The data in the *adult* data set is the data set constructed for the personal information of some people. In the *adult* data set, the age, job category, work number, educational level, length of service, marital status, position, social relationship, race, sex, annual income, expenditure, week workload, nationality, and

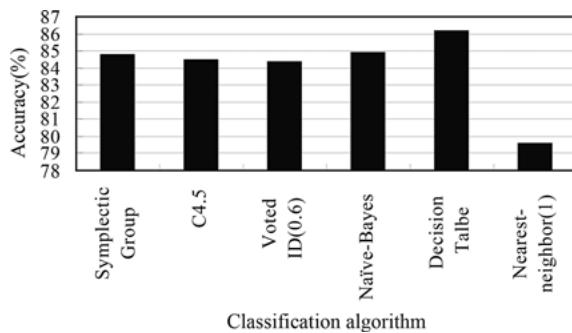


Fig. 3.5: Comparison of recognition rates of symplectic group classification algorithms and other algorithms.

total annual income are listed. For these attributes of the *adult* data set, some of these properties are classified. Before the classification began, we put the age, education level, and annual income status as the properties of the classification sample. First, the corresponding attributes of age and education level constitute the corresponding vector, and then the vector of the two attributes constitute a matrix. The two columns in the matrix correspond to the two attributes of age and education degree. The matrix is reduced, and the reduced symplectic matrix is evaluated according to the symplectic group classification algorithm. In this process, we must ensure that the number of samples is even; otherwise, we cannot use the classification algorithm. In this data set, there are two cases of annual income for one person, namely ‘ $\leq 50k$ ’ and ‘ $> 50k$ ’. The symplectic group classifier algorithm is used to study the class of the two income categories. In this data set, we validate 2000 data; the correct rate of classification is shown in Figure 3.5.

By analysing the classification effect, we can see that the symplectic group classification algorithm has some advantages compared with some of these algorithms, which is only based on some preliminary results. In fact, these algorithms can be further improved. Readers may investigate further. In addition, we selected 10 data from the *UCI* data set for data classification experiments. The experimental data are shown in Table 3.2.

For the above 10 types of data, each class is pretreated. Then, for these processed data, using the symplectic group classification algorithm, we perform data set classification to classify the 10 types of data. The correct rate of classification is shown in Figure 3.6.

In Figure 3.6, the number of samples in the data set of category 2 is the largest, and the classification rate of this type of data is the highest. When the number of sample instances is large and the classification attributes are more, the classification rate of the symplectic group classification algorithm is higher. When the number of instances is small, the classification accuracy is low. For a large sample of the data set, the use of the symplectic group algorithm for classification can have a better classification effect.

Tab. 3.2: Attributes of category data.

Name	Number of instances	Number of attributes	Number of categories
Balance-scale	625	5	3
bridges	108	13	2
car	1728	6	4
cmc	1473	10	3
hepatitis	155	20	2
iris	150	4	3
kinship	104	12	2
Lung-cancer	32	2	2
spect	267	23	2
wine	178	13	3

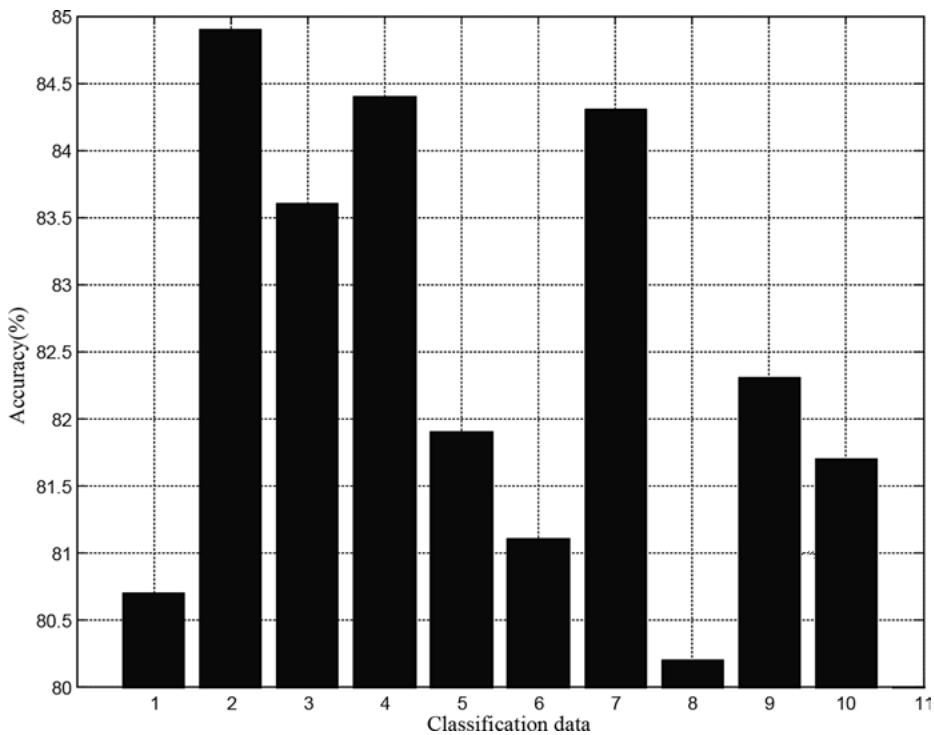


Fig. 3.6: Data set classification effect diagram.

3.4 Application example

Face recognition is an important topic in the field of pattern recognition and artificial intelligence [32–35]. It has more direct and friendlier features than fingerprint and other human biological characteristics. However, for the computer, face recog-

nition is still a difficult task. This is mainly because the face image is affected by external factors, such as object occlusion, expression transformation, and other factors affecting the identification process. Studies on face recognition are underway, and there are many methods for face recognition at present. These methods can be divided into four categories: feature-based, template-based, 3D-based and random sequence methods. Usually, in the face recognition process, image acquisition and face detection need to use considerable image processing technology. In this section, we use the constructed symplectic group classification algorithm to recognise the image. The feature sequence of the image is composed of the symplectic matrix. Algebraic transformation and geometric transformation of the matrix are carried out, and then, the singular eigenvalues of the symplectic matrix are obtained. The singular values are decomposed according to the method described in Section 3.3.1.1, and the singular value vector of the symplectic matrix is obtained. For a face image, the symplectic group eigenvalues formed by it are used as the eigenvectors of the face. The feature database of each face image training sample can be divided into several categories according to the eigenvalues. The symplectic classification algorithm can then be used to identify images in the image library that are consistent with the known face images. In the following, the feature extraction and recognition process of the face recognition are given. For image acquisition and face detection, due to the complexity involved in image processing, we do not elaborate further here. The face images used in this example are derived from the images in the existing image library. The image recognition is carried out by pre-processing the image and extracting the features, and the recognition accuracy of the symplectic group classification algorithm is given.

3.4.1 Processing of images under symplectic matrix

For a face image, the image is represented by the symplectic matrix according to the method described in Section 3.3.1.1. Then, singular eigenvalue decomposition of the matrix is carried out. The other images are processed accordingly. If the image is consistent with the source image after singular value decomposition, it matches the source image and finally produces the output. The recognition process of the symplectic group classification algorithm is shown in Figure 3.7.

Assuming that the image A_k is transformed into an mn -dimensional image vector $z_k(k = 1, 2, \dots, n)$, for n training images $A_k \in R^{m \times n}(k = 1, 2, \dots, n)$, the vectors of all the training images are written as Z . The following matrix is constructed using the singular value decomposition theorem

$$D = Z^T Z \in R^{N \times N}.$$

Here, we call matrix D the generating matrix and use the calculation method of the symplectic matrix to obtain the eigenvalue λ of the symplectic matrix and the corresponding orthogonal normal vector v . Then, we obtain an orthogonal normal vector

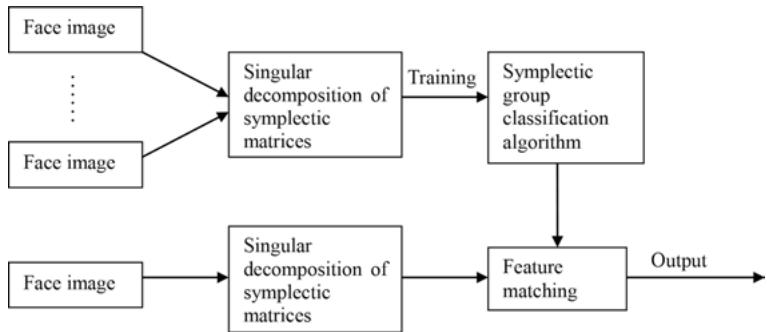


Fig. 3.7: recognition process of symplectic algorithm.

according to the singular value decomposition process as follows:

$$u_i = \frac{1}{\sqrt{\lambda_i}} Z v_i .$$

This vector is called the eigenvector of the image in the image recognition process, and each face image is projected into a symplectic space. Each image corresponds to a point in the symplectic space, so that n training images are decomposed by singular values. All the singular values are arranged in descending order into the vector $[\lambda_1, \lambda_2, \dots, \lambda, 0, \dots, 0]^T$, which is the singularity of the image. Because the singular value decomposition is unique for any one of the images A , the singular value feature of the image is unique when all the singular values are arranged in descending order. Thus, the singular value feature can be used as a numerical characteristic to describe the grey scale matrix. For the description of the characteristics of the image, if there is a small change in the image grey scale, this change is not significant, and it indicates stability. According to the perturbation theory, the singular value has good stability. The following theorem illustrates this point.

Theorem 3.11. If $A \in R^{p \times q}$, $B \in R^{p \times q}$ (without loss of generality, set $p \geq q$), the singular value of $\lambda_1^a \geq \lambda_2^a \geq \dots \geq \lambda_q^a$, $\lambda_1^b \geq \lambda_2^b \geq \dots \geq \lambda_q^b$, respectively. For any unitary invariant norm of $R^{p \times q}$, all of the following inequalities are established,

$$\|\text{diag}(\lambda_1^a - \lambda_1^b, \dots, \lambda_q^a - \lambda_q^b)\| \leq \|B - A\| .$$

If the norm in the theorem is the Frobenius norm, then the above equation can be changed to

$$\sqrt{\sum_{i=1}^q (\lambda_i^a - \lambda_i^b)^2} \leq \|A - B\| .$$

Small changes in the grey scale of the image do not cause a large change in the singular value, which is insensitive to image noise. The displacement of the image is equivalent to the permutation of the image matrix, i.e., the elementary transformation of the two rows

of the image matrix is equivalent to matrix left multiplication $I_{ij} = I - (e_i - e_j)(e_i - e_j)^T$, where e_i, e_j are the i -th and j -th columns of the unit matrix I , respectively. The transformed matrix is $I_{ij}A$. Apparently, $I_{ij}^T = I_{ij} = I_{ij}^{-1}$. Then, the characteristic equation of $(I_{ij}A)(I_{ij}A)^T$ is

$$|(I_{ij}A)(I_{ij}A)^T - \lambda I| = 0.$$

The equation can be transformed into

$$|I_{ij}AA^T I_{ij} - \lambda I| = |I_{ij}| \times |AA^T - \lambda^2 I_{ij}^{-1} I_{ij}^{-1}| \times |I_{ij}| = |AA^T - \lambda^2 I| = 0.$$

Therefore, the original image A has the same singular value characteristic as the image $I_{ij}A$ after its replacement. Similarly, the column substitution has the same result, i.e., the singular value feature has displacement invariance. The singular value changes in proportion to the brightness of the corresponding image. When the brightness of the whole image changes proportionally, the singular value of the image changes proportionally. This change in proportion does not change the identification information it contains, so normalising the singular value feature can eliminate the influence of the scale factor. If the image grey scale changes in proportion, i.e., if it is multiplied by a nonzero real a on A , the image matrix aA is obtained. Since $\text{rank}(A) = \text{rank}(aA)$, we can set $\text{rank}(A) = r$. The singular values of A and aA are

$$\lambda_1 \geq \lambda_2 \geq \dots \geq_r \lambda'_1 \geq \lambda'_2 \geq \dots \geq \lambda'_r.$$

Then, the characteristic equation of $(aA)(aA)^T$ is

$$|(aA)(aA)^T - \tau^2 I| = 0$$

i.e., $|AA^T - \tau^2 I/a^2| = 0$. We can then get

$$(\lambda_1, \lambda_2, \dots, \lambda_r, 0, \dots, 0)^T = |a| (\lambda'_1, \lambda'_2, \dots, \lambda'_r, 0, \dots, 0)^T.$$

The grey scale of the image varies in proportion, which is equivalent to multiplying a singular value by a scale factor.

According to the symplectic matrix theory, any rotation transformation can be decomposed into the product of two orthogonal matrices, such as in the two-dimensional case

$$\begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} = \left(I - 2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \times \left(I - 2 \begin{bmatrix} \sin \theta/2 \\ -\cos \theta/2 \end{bmatrix} \begin{bmatrix} \sin \theta/2 & \cos \theta/2 \end{bmatrix} \right).$$

Therefore, the rotation transformation of the source image A is equivalent to the left multiplication orthogonal matrix P , and the rotated transformed image can be denoted as PA , since

$$(PA)(PA)^T = P(AA^T)P^T = P(AA^T)P^{-1}.$$

Orthogonal transformation of the image matrix A may result in a similar transformation to the matrix AA^T or A^TA , while the similarity matrix $(AA^T), P(AA^T)P^{-1}$ has the same

singular value, indicating that the original image and the rotated transformed image have the same singular value. By the singular value decomposition theorem, we have

$$AA^T u = \lambda^2 u, A^T Av = \lambda^2 v.$$

i.e., A and A^T have the same singular value, i.e., the same singular value feature. It is precisely because the singular value feature of the image has the above attributes. The singularity feature is considered to be an ideal algebraic feature. This method can be used in pattern recognition.

For two different frontal face images, we separately carried out singular value decomposition: $A_1 = U_1 \Lambda_1 V_1^T$, $A_2 = U_2 \Lambda_2 V_2^T$. First, to ensure that the orthogonal matrix U_1, V_1, U_2, V_2 is unchanged, transform Λ_1, Λ_2 to reconstruct the images separately. After reconstruction, it is easy to see that, although A_2 and $U_1 \Lambda_2 V_1^T$ have the same singular value characteristics, both of which characterise two different images. Based on this method to reconstruct the image, it is easy to see that the reconstructed image still represents the same person compared with the original image. The difference is mainly due to the different angles of image acquisition. In the case of face image recognition, it is not sufficient to adopt such a method, and more useful information is included in the two orthogonal matrices U and V decomposed by singular values. Taking into account the orthogonality of U, V , we can get the equation

$$\Lambda = U^T AV.$$

Therefore, recognition based on the singular value feature compares and identifies the representation of the face image in different spaces. As the projection transformation is different, i.e., the choice of benchmarks is different, only the use of singular value characteristics for identification is unreasonable. There are two ways to solve these two problems:

- (1) Extract the identification information contained in the two orthogonal matrices.
- (2) The image is projected onto two orthogonal matrices of each class for feature extraction. The mean values of the various classes are calculated and the singular values are decomposed to obtain the two orthogonal matrices U and V of the mean values. Then, the samples are projected onto U and V and then the feature is extracted. Let the two orthogonal matrices of the i -th class mean be

$$U^i = [u_1^i, \dots, u_p^i], V^i = [v_1^i, \dots, v_q^i] (p \geq q).$$

The feature obtained by projecting the sample A_x into the space may be represented as $[a_1^i, \dots, a_q^i]^T$, where

$$a_j = (v_j^i)^T A_x^T u_j^i.$$

Then, we use the feature extraction method. First, a standard image is given to perform singular value decomposition, and the standard symplectic space and the corresponding feature vectors are obtained. An image is represented as a form of a symplectic matrix. If A is an image of the matrix, $B = AA^T$, we call matrix B the feature matrix of image matrix

A. Feature extraction is performed using the concentration of the eigenvalue distribution of the feature matrix. For each image A_j^i , we project it into a $2n$ -dimensional vector, i.e.,

$$X_j^i = A_j^i v .$$

We call v and X_j^i as projection vectors and projection feature vectors, respectively. After the two images are transformed by the symplectic singular matrix, two new symplectic matrices are obtained. The original images are reconstructed by the two symplectic orthogonal matrices, and the original images are projected onto the two symplectic orthogonal matrices. In this projection space, the symplectic classifier is used for image classification. For different face images, some images are selected as training examples. In this process, the image is extracted by features and then expressed as a symplectic matrix. Using the simplification method of the symplectic matrix, we can simplify the matrix of the image, and the feature subspace of the image can be obtained. Then, the image is used as a training example, and other images in the image library are used as examples to be recognised.

For the image in the image library to be recognised, the image is composed with the same feature combination, the image is represented by the symplectic matrix, and the matrix is then singularly decomposed to obtain the corresponding symplectic representation method. Then, we use the symplectic group classifier algorithm in the face recognition process to deal with the image.

3.4.2 Instance validation

To verify the performance of the symplectic group classifier algorithm, we used ORL and YALE, two types of face image libraries, to test the algorithm performance. The ORL library is described in Section 3.3.2. The YALE database consists of 15 people (11 images per person) with greater lighting and facial changes. The hardware environment of the test system is *Pentium (R) 4 CPU (2.80 GHz) with 512 MB memory*. The software platform is *MATLAB 6.5*. For an image in the image library, we first calculate its symplectic matrix and use the symplectic algorithm for training. Then, in the database, to find another expression of the same person, according to the correct rate of finding the image, the algorithm recognition efficiency is obtained. As shown in Figure 3.8, the performances of several algorithms are compared.

Figure 3.9 shows the recognition rate of the two classification algorithms in the ORL library. For each class of images in the ORL library, we use the samples 2, 3, .., 9 as the training samples, and the others as the test samples. Compared with the recognition algorithm, the recognition rate of the image is higher when the training group is larger.

Figure 3.10 shows the recognition rate of the two classification algorithms in the YALE library. For each type of image in the YALE library, 2, 3, .., 10 images are selected as the classification samples and the others as the test samples. Compared with PCA



Symplectic group recognition



Fig. 3.8: Face recognition effect diagram of symplectic group classification algorithm.

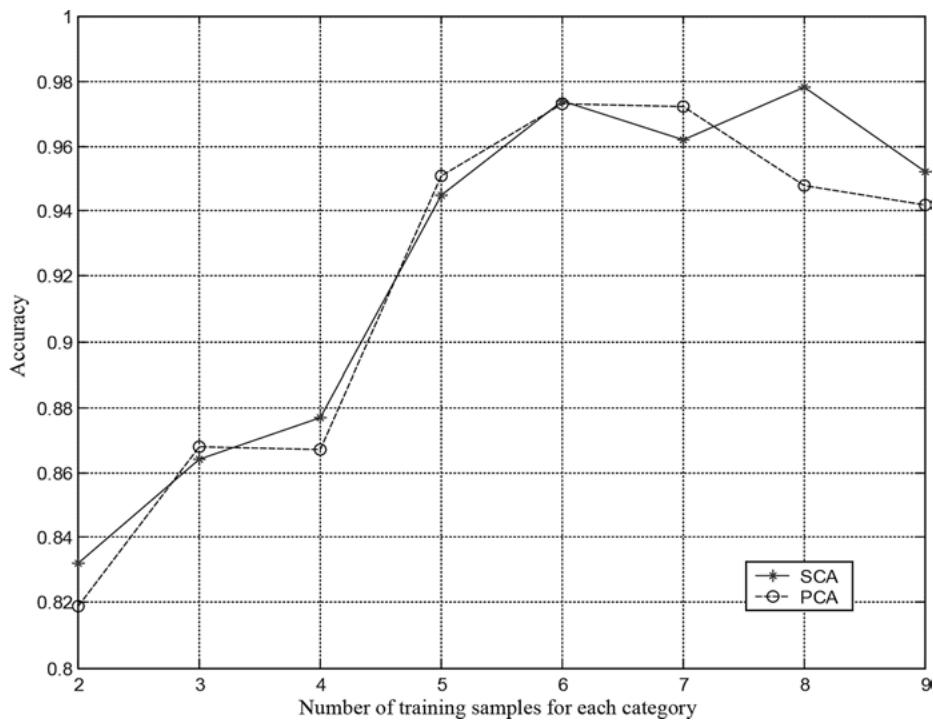


Fig. 3.9: Face recognition rate in the ORL library.

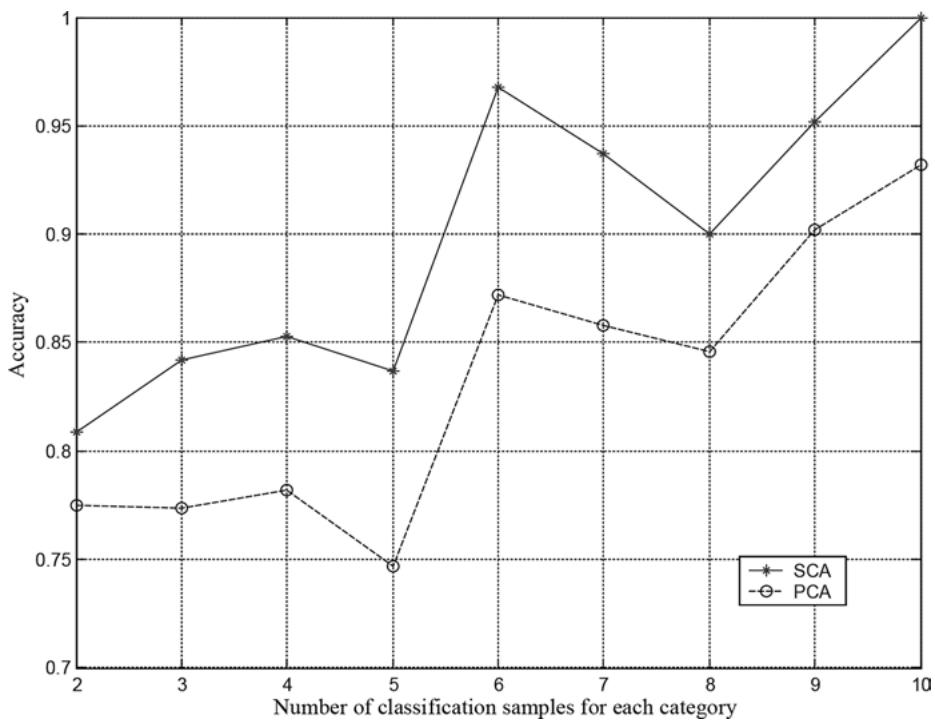


Fig. 3.10: Recognition rate in the YALE library.

recognition algorithm, the recognition efficiency of symplectic group classification algorithm is high.

In the process of recognising the image, we use the symplectic group classification algorithm to use multiple images as the training samples. Here, we give the recognition rate of the symplectic group classification algorithm using only one image as the training sample and other images as the test sample. Through the recognition process, we find that, even with only one image as a training example, the symplectic group classifier algorithm has a higher recognition accuracy. This is because, for an image, the image processed by the symplectic classification algorithm obtains the feature vector of the image, and the feature vector is stable for different images of the same face. Tables 3.3 and 3.4 give the performance comparison of several classification algorithms on the ORL and YALE libraries in the case of only one training sample [36, 37].

Tab. 3.3: Performance comparison of several algorithms on ORL Library.

Classification algorithm	Number of training samples	Average recognition accuracy (%)
Symplectic group	1	92.6
Spectroface	1	90.8
Linear subspace	1	84.2

Tab. 3.4: Performance comparison of several algorithms on YALE Library.

Classification algorithm	Number of training samples	Average recognition accuracy (%)
Symplectic group	1	92.6
Spectroface	1	95.7
Linear subspace	1	84.4

The results show that the accuracy rate of the average recognition is at a high level even when there is only one training classification sample, and the correctness rate of the symplectic group is relatively high compared with the linear subspace method.

3.5 Summary

As a new method of machine learning, Lie group machine learning has attracted considerable attention from researchers. At present, Lie group machine learning theory has been applied to many fields, e.g., there have been some successful examples in the field of classification. In this chapter, we discussed the design method of the symplectic group classifier in Lie group machine learning, studied the symplectic group classifier algorithm in the framework of Lie group machine learning, and used the classifier for image recognition and data set classification. Thus, this method was shown to be advantageous.

Bibliography

- [1] Feng K, Qin MZ. Symplectic Algebraic Algorithm of Hamiltonian System. Zhejiang: Zhejiang University Press, 2006.
- [2] Shaw WT. Symplectic Geometry of null Infinity and Two-surface Twisters. Mathematical Institute, Oxford UK.
- [3] Gao L, Li YM, Chen XR, Yang KQ. An Attempt to Seismic Ray Tracing with Symplectic Algorithm. Chinese Journal of Geophysics, 2000, 5(3), 402–410.
- [4] Wu Y, Hu JY, Yin XG. Symplectic Integrators of the Equations of Multibody System Dynamics on Manifolds. Advantages in Mechanics, 2002, 2(32), 189–195.
- [5] Wang SX, Liu YW, Wu HT. Study on Computational Accuracy of Robot Dynamics. Chinese Journal of Mechanical Engineering, 1998, 34(1), 14–20.
- [6] Han XS, Wang SX, Yu SY. Investigation of Nanomanufacturing Process by Molecular Dynamics Simulation Based on Symplectic Algorithm. Chinese Journal of Mechanical Engineering, 2005, 41(4), 17–22.
- [7] Zhang K. The study on PML-FDTD and Boundary Connecting Conditions of Total-scattering Fields in 3-D Cylindrical Coordinates. Journal of Chongqing University of Posts and Telecommunications, 2003, 3, 21–26.
- [8] Xu ZX, Zhou DY, Deng ZC. Numerical Method Based on Hamilton System and Symplectic Algorithm to Differential Games. Applied Mathematics and Mechanics, 2006, 27(3), 305–310.

- [9] Sun Y, Xie J. Hamiltonian Symplectic Semi-analytical method and its application in anisotropic electromagnetic waveguide. *Chinese Journal of Computational Mechanics*, 2005, 22(06), 690–693.
- [10] Zhang PG. A Post-processing Algorithm for the Dynamic Analysis of DNA Elastic Slender Structure. *Journal of Qingdao University*, 2006, 21(3), 71–74.
- [11] Ritter G, Gallegos MT. A Bayesian approach to object identification in pattern recognition. *Proceedings of the International Conference on Pattern Recognition*, 2000, IEEE: 1082–1086.
- [12] Lovell BC, Bradley AP. The Multiscale Classifier. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 1996, 2(18), 124–137.
- [13] Dawi AHM, Chong TS, Fong SF. The Effects of Different Three-Dimensional Animated Visual Displays in Computer-Based Multimedia on Learners with Different Spatial Abilities. *Proceedings of the Fifth IEEE International Conference on Advanced Learning Technologies*. IEEE, 2005, ICALT'05, 798–801.
- [14] Tuzel O, Subbarao R, Meer P. Simultaneous Multiple 3D Motion Estimation via Mode Finding on Lie Groups. *Proceedings of the Tenth IEEE International Conference on Computer Vision. Pattern Anal. Machine Intelligence*, 2005, 1, 18–25.
- [15] Fletcher PT, Lu CL, Joshi S. Statistics of Shape via Principal Geodesic Analysis on Lie Groups. *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Transaction on Medical Imaging*, 2003, 1, 95–99.
- [16] Miasnikov AD, Haralick RM. Regression analysis and automorphic orbits in free groups of Rank 2. *Proceedings of the 17th International Conference on Pattern Recognition*. IEEE, 2004, ICPR'04, 180–183.
- [17] Bonmassar G, Schwartz EL. Lie Group, Space-Variant Fourier Analysis and the Exponential Chirp Transform. *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition*. IEEE, 1996, CVPR'96, 492–496.
- [18] Yang LY, Qin Z, Zhang XP. Classifier Simulation Algorithm and Its Applications. *Journal of Xi'an Jiaotong University*, 2005, 39(12), 1311–1314.
- [19] Mitchell TM. *Machine Learning*. Beijing: Machinery Industry Press, 2003.
- [20] He YH, Zhao L, Zou CR. Kernel Nearest Features Classifier for Face Recognition. *Journal of Applied Sciences*, 2006, 24(3), 227–231.
- [21] Squire DM, Caelli TM. Invariance Signatures. Characterizing Contours by Their Departures from Invariance. *Computer Vision and Image Understanding*, 2000, 77(3), 284–316.
- [22] Zhong WX, Ouyang HJ. Hamiltonian System and Simplectic Geometry in Mechanics of Composite Materials – Plane Stress Problem. *Applied Mathematics and Mechanics*, 1992, 13(12), 1031–1035.
- [23] Ma ZJ. *Group Theory in Physics*. Science Press, 2006.
- [24] Lai JH, Yan XH, Deng DG. Finishing the Classifier Design of Spectroface Human Face Recognition. *Journal of Image and Graphics*, 2002, 7(5), 466–471.
- [25] Yu HB. Detection and Tracking of Human Body Based on Head Feature Extraction and Its Application. Hangzhou: Zhejiang University, 2007.
- [26] Wang P. Research on Classification Algorithm on Data Stream. Shanghai: Fudan University, 2007.
- [27] Xie YH. Research on Some Problems of Face Feature Extraction and Classifier Design. Nanjing: Nanjing University of Science and Technology, 2006.
- [28] Li L. Using Symplectic Geometry to Construct Authentication Codes With Arbitration. *Journal of Jilin Architectural and Civil Engineering Institute*, 2006, 23(4), 63–67.
- [29] Zhao LiH, Sun XH. Skin Color-Based Face Detection. *Journal of Engineering Graphics*, 2005, 3, 84–88.

- [30] Fu HX. Research on Symplectic Group Classifier in Lie Group Machine Learning. Suzhou: Soochow University, 2008.
- [31] Fu HX, Li FZ. Research of the symplectic group classifier based on Lie group machine learning. In proceedings of 2007 IEEE the 4th International conference on Fuzzy System and Knowledge Discovery (FSKD'07), 2007, 649–655.
- [32] Sun MM. Research on Manifold Learning Theory and Algorithm. Nanjing: Nanjing University of Science and Technology, 2007.
- [33] Huang QH. Research on Manifold Learning Method Theory and Image Application. Chengdu: University of Electronic Science and Technology, 2007.
- [34] Wang J. Research on the Theory and Method of Manifold Learning. Hangzhou: Zhejiang University, 2006.
- [35] Wen QD. Training Classifier Algorithm Based on EP. Zhengzhou: Zhengzhou University, 2007.
- [36] Lai JH, Yuen PC, et al. Face Recognition using holistic Fourier invariant features. Pattern Recognition, 2001 34(1), 5–109.
- [37] Liu XW, Wang DL. Appearance-based recognition using perceptual components. Proceedings of 2001. IJCNN, 2001, 3, 1943–1948.

4 Quantum group learning

This chapter is divided into four sections. Section 4.1 presents a question. Section 4.2 introduces the construction method of the quantum group classifier in Lie group machine learning. Section 4.3 describes an application of the quantum group learning algorithm in molecular docking. Finally, Section 4.4 summarises the chapter.

4.1 Problem presentation

The quantum group was proposed by the 1990 Fields Award winner Drinfeld at the 1986 ICM at Berkeley [1]. The first example involved two types of special Hopf algebras; one is the deformation of the generalised enveloping Hopf algebra of semi-simple Lie algebras L , and the other is the deformation of the coordinate rings of affine algebraic groups G . There is no well-defined axiomatic definition for quantum groups. Mathematician S. P. Smith stated, ‘a non-commutative and non-co-commutative Hopf algebra is called a quantum group’ [2]. Drinfeld defined quantum groups as the spectra of noncommutative Hopf algebras. The quantum space category is defined as the duality of noncommutative algebraic categories, and the quantum space corresponding to the algebra A is denoted as $\text{spec}A$ (spectrum of A). According to this definition, the two concepts of Hopf algebra and quantum group are actually equivalent, but the latter is more abundant in geometry.

Quantum group is the generalised concept of classical symmetry of the Lie group and Lie algebra. It is a special noncommutative and non-exchange Hopf algebra. In particular, the Hopf algebra is quantised from the enveloping algebra of Lie algebra and the function algebra of the group. In recent years, the quantum group and its related q-deformation theory have been applied to various branches of physics, such as integrable models in statistical physics, conformal field theory, canonical theory, string theory, vibrator systems, quantum optics, atomic and molecular physics, and nuclear physics. Nikolaos Nasios et al. applied these concepts to nuclear classification and **achieved** satisfactory results [2]. It can be seen that quantum group theory has been widely used in many fields, which represent excellent opportunities for machine learning and provide useful examples for data analysis of machine learning. Therefore, we learn from the perspective of machine learning and introduce the basic theory of quantum cluster into the machine learning domain. Our achievements are summarised as follows.

4.2 Construction method of quantum group classifier in Lie group machine learning

With the basic theory of Lie group, references [3–13] give the basic concepts and related algorithms of Lie group machine learning. For non-commutative and non-symmetry problems of machine learning systems, based on the basic theory of quantum groups, this section gives the basic construction method and related algorithm of the quantum group classifier in Lie group machine learning [14–16].

4.2.1 Problem description

Bioinformatics is a new discipline involving the interdisciplinary aspects of biology, computer science, and applied mathematics. It tries to use information science technology to study the phenomena and laws with respect to DNA and genes, gene expression, proteins, gene circuits, cells, physiological manifestations, and so on. Biological experimental data acquisition, processing, storage, retrieval, and analysis are performed to achieve the purpose of revealing the biological significance of the data. DNA sequence analysis is an important part of modern biological sequence analysis in biological gene information systems, and it is a hot topic in bioinformatics research. Through sequence analysis, we can predict the structure and function of new sequences, and we can analyse the similarity and homology between sequences. For molecular biology, when we focus on gene sequencing, we do not need to ‘learn’. However, when we focus on the problem of gene function, we have to address the ‘learning’ problem.

In general, the basic task of machine learning for bioinformatics research is to find meaningful knowledge from existing biology databases and to construct meaningful and understandable models (see Table 4.1 for specific tasks).

In 1994, Adleman first published a pioneering article on DNA molecular algorithms in Science [17]. Currently, DNA sequence analysis is an active research area. Existing machine learning methods include neural network methods, decision tree methods, case-based learning methods, symbolic knowledge optimisation methods, logic-based induction learning methods [18], statistical learning theory [19], and sup-

Tab. 4.1: Tasks of machine learning in Bioinformatics.

Classification	Functional description
Classification	Predict project category
Clustering	Discover project team
Characterization	Describe the characteristics of the group
Deviation Detection	Find changes
Link Analysis	Look for relationships and links
Visualization	Use visual method to express data to facilitate the discovery of knowledge

port vector machines [20–22]. The hidden Markov model (HMM) is a probabilistic model that has been widely used in the field of gene recognition.

A gene contains an exon, intron, and gene spacer. In the DNA transcription process, the exon as a coding sequence is retained, while the other noncoding region, i.e., the intron and gene spacer, are cut off. Exons are spliced together to form a messenger ribonucleic acid (mRNA) and then translated into a polypeptide chain, constituting a variety of structural proteins and catalysing a variety of biochemical reactions of enzymes and hormones. In DNA sequence analysis, DNA-based gene detection is the key to gene discovery. The DNA sequence is divided into coding and non-coding regions. The coding region of DNA is often the region where the gene is located. Therefore, segmentation of DNA sequences is important for the study of genes.

The advantages and disadvantages of the support vector machine (SVM) algorithm are briefly described as follows.

A support vector machine is a type of machine learning method based on statistical theory. Through the learning algorithm, SVM can automatically find support vectors that have a better ability to distinguish between categories. The resulting classifier can maximise the interval between classes and hence shows better generalisation performance and higher classification accuracy. The main idea of SVM is to solve the classification problem of two types of categories, and to find a hyperplane in high-dimensional space as two types of segmentation to ensure the minimum classification error rate. An important advantage of SVM is that it can handle linear indivisibility. Using the idea of SVM classification, we first extract the features from the original space and map the samples of the original space into a vector in the high-dimensional feature space to solve the problem of linear indivisibility in the original space. SVM can be used for regression, classification, and density estimation. An artificial neural network and the ‘black box’ algorithm can be used for similar purposes. Although the SVM has good versatility, it is time-consuming and poorly expressed in the testing phase.

In view of these advantages and disadvantages, this chapter presents a DNA population classification method based on quantum population and applies it to the entire sequence of the plant *Arabidopsis thaliana*.

4.2.2 Construction of quantum group classifier in Lie group machine learning

The quantum group classifier can be constructed as long as the Jacobi conditional analysis of the quantum group is used to observe the properties of the quantum group and quantum algebra of the data. The quantum group is regarded as a quantum hyperplane linear system. Through the non-exchange space on the dimensionality, linearization, and other forms, the observation data is processed. The nonlinear structure of the observed data is reduced to a linear structure, and it is then classified according to the linear transformation learning algorithm for the quantum group.

4.2.2.1 Symmetric linear transformation learning algorithm for quantum group

On the two-dimensional quantum hyperplane, it is assumed that the learning problem in the learning space is represented by ‘coordinates’ (x, y) , where x represents an instance in the instance space in the learning system and y represents an example in the sample space, satisfying the exchange relation

$$xy = qyx \quad (q \in C) \quad (4.1)$$

where q is the adjustment factor and C is the set of adjustment factors. In addition, there is a related covariant differential calculation of (x, y) , i.e., the ‘form’ (ξ, η) , which satisfies

$$\xi\eta = -q^{-1}\eta\xi, \quad \xi^2 = 0, \quad \eta^2 = 0. \quad (4.2)$$

When $q = 1$, from (4.1), it can be seen that x and y are symmetric, and from (4.2), it can be seen that ξ and η are symmetric. Further, (ξ, η) can be interpreted as the differential of (x, y) , i.e., to reduce the number of examples in the learning space processing. Let

$$\varphi = \begin{pmatrix} x \\ y \end{pmatrix}, \quad \psi = d\varphi = \begin{pmatrix} \xi \\ \eta \end{pmatrix}. \quad (4.3)$$

The ‘coordinates’ (x, y) on the quantum hyperplane are linearly transformed with the ‘form’ (ξ, η) at the same time,

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}, \quad \begin{pmatrix} \xi' \\ \eta' \end{pmatrix} = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \begin{pmatrix} \xi \\ \eta \end{pmatrix} \quad (4.4)$$

where $(\alpha, \beta, \gamma, \delta)$ and (x, y, ξ, η) are interchangeable, and if the requirements of (x', y') and (ξ', η') are satisfied with the same relationship between (x, y) and (ξ, η) , then $(\alpha, \beta, \gamma, \delta)$ must have the following relationship

$$\begin{aligned} \alpha\beta &= q\beta\alpha, & \alpha\gamma &= q\gamma\alpha, & \beta\gamma &= q\gamma\beta, & \gamma\delta &= q\delta\gamma, & \beta\delta &= q\delta\beta \\ \alpha\delta - \delta\alpha &= (q - q^{-1})\beta\gamma. \end{aligned} \quad (4.5)$$

Let algebra

$$A = C \langle \alpha, \beta, \gamma, \delta \rangle / \sim. \quad (4.6)$$

It is the associative algebra generated by $(\alpha, \beta, \gamma, \delta)$, and it satisfies (4.5), i.e.,

$$T = \left(T_j^i \right) = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \in \mathrm{GL}_q(2). \quad (4.7)$$

The above transformation will form a quantum super-symmetric linear transformation and yield the quantum group $\mathrm{GL}_q(2)$, whose centre is

$$c = \det_q T = \alpha\delta - q\beta\gamma = \delta\alpha - q^{-1}\beta\gamma. \quad (4.8)$$

If $\det_q T = 1$ is required, it is called quantum group $\mathrm{SL}_q(2)$.

In the formation of quantum groups $GL_q(2)$ and $SL_q(2)$, a quantum group symmetric linear transformation learning algorithm is also produced at the same time. The algorithm is described as follows:

Algorithm 4.1: Symmetric linear transformation learning algorithm on quantum group:

QgroupLinear (X , Y)

Input: Instance set $X = \{x_1, x_2, \dots, x_n\}$, sample set $Y = \{y_1, y_2, \dots, y_n\}$

Output: Symmetric linear transformation of quantum group

Step1 After X and Y are reduced through dimension processing, obtain ξ and η , and they must satisfy (4.2);

Step2 For X , Y and ξ , η for a symmetrical linear transformation Z , the transformation of X , Y and ξ , η should satisfy the relationship between (4.1) and (4.2);

Step3 Verify that the symmetric linear transformation Z satisfies (4.5); if not, adjust the coefficient q until a qualifying change is found. Then, Z is the required symmetric linear transformation.

Step4 Return to linear transformation Z .

4.2.2.2 Quantum group classifier in Lie group machine learning

For each of these two classification problems, assume that the sample set G is $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n) \in R^N \times Y$, where $Y = \{-1, +1\}$. Our goal is to find the classifier $f(x)$, making $y \cdot f(x) > 0$.

In this way, by using the transformation of the symmetric linear transformation learning algorithm 4.1 of the quantum group, we can transform the data of the sample set into the classical quantum group that we need, and the classification can be directly determined by the value of its determinant. The classification algorithm of the quantum group classifier in machine learning is described as follows.

Algorithm 4.2:

Classification algorithm of quantum group classifier in machine learning:

QGClassifier (G)

Initialisation: G is a quantum group whose group elements are $\{x_1, x_2, \dots, x_n\}$

Input: G

Output: $\{1, -1\}$

```
{  
for i = 1 to n do //The value of its matrix determinant  $\det_q T$  is calculated for each group element//  
if  $\det_q x_i == 1$   
then return 1; //  $G$  is  $SL_q(n)$  group//  
else return -1; //  $G$  is  $GL_q(n)$  group//  
end if  
repeat  
end for  
}
```

The construction of a specific quantum group learning classifier is described below:

Step 1 Map the sample set to this non-empty set of G ;

Step 2 Construct the corresponding quantum group structure according to G and use the algorithm (see Appendix 17.5) to test whether it is a quantum group;

- Step 3** The linear transformation of the symmetric linear transformation learning algorithm 4.1 of the quantum group is applied to the established quantum group structure to obtain the classical quantum group that we need;
- Step 4** Use the classification algorithm of quantum group classifier 4.2 to classify and train the appropriate classifier;
- Step 5** Instance test.
-

4.2.3 DNA Sequence classification based on quantum group classifier in Lie group machine learning

Before introducing the specific DNA sequence classification experiment, we give a simple explanation of the experimental data.

4.2.3.1 Data preparation

We selected the full sequence of *Arabidopsis thaliana* in eukaryotes as the experimental data. The data was obtained from the National Center for Biotechnology Information (NCBI); its website is <http://www.ncbi.nlm.nih.gov>. The entire sequence consists of five chromosomes, each of which is in the FASTA file format. Owing to space constraints, the description of the FASTA file format is omitted; only the index ID length and file size of each chromosome are described (see Table 4.2).

In the process of information extraction, we use the following structure to store the information read from each chromosome sequence.

```
typedef struct
{
    char * seqname; // Gene sequence name //
    char * wholeseq; // Complete gene sequence //
    char * info; // Gene additional information //
    long index; // Position of the gene sequence in the gene pool //
    unsigned len; // Length of the gene sequence //
    int * count; // Number of observations in the gene sequence //
}SeqInfo;
```

In extracting the data, our main purpose is to extract the coding and noncoding regions of each chromosome sequence separately. The method used is described below.

- (1) For each chromosome, read and save the gene after the start and end of the base number and CDS after the base of the base number.
- (2) In the final sequence to find the location of the logo in the CDS, these segments are connected as the *i*-th SeqInfo structure into the coding area file coding.txt.
- (3) Connect the remaining gene segments as the *i*-th SeqInfo structure into the non-coding region file noncoding.txt.
- (4) Repeat the above steps until the last chromosome.

Tab. 4.2: Description of chromosome file.

Chromosome	Index ID	Length	Size
1	NC_003070	30,080,809bp	43M
2	NC_003071	19,643,621bp	27.7M
3	NC_003074	23,465,812bp	33.5M
4	NC_003075	17,549,528bp	24.9M
5	NC_003076	26,689,408bp	38M

Tab. 4.3: Number of data of each chromosome coding region and non-coding region.

Chromosome	Encoding area data size	Non-coded area data file
1	6761	6572
2	4216	4134
3	5252	5027
4	3958	3482
5	5987	5160

For the five chromosomes, we get five files that store the coding area and five files that store the non-coding regions. The number of data for each document is listed in Table 4.3.

4.2.3.2 DNA sequence classification

Because the coding and non-coding areas have different functions, there will be some sequence in the coding area that appears frequently. However, in the non-coding region, it appears rarely. These words can be called the coding region of the feature sequence. By contrast, it can be seen as a non-coding region of the characteristic sequence. Because each DNA sequence is composed of A, T, G, C, i.e., four characters immediately composed of strings (such as *aggcacggaaaacggaaatt*), there are many ways to digitise DNA sequences and different methods have a significant impact on the classification.

In general, before DNA sequence processing, a mathematical expression of the DNA sequence is first found and then other tools are used for its analysis and research. There are many mathematical expressions of DNA sequences that are used globally, such as the random walk DNA model, CGR method based on chaos theory [23], and Z-curve representation using the geometric method [24]. In this chapter, we use the sequence-based entropy method. Because the sequence set has five chromosomes, we can extract the feature sequence in each chromosome and then apply the designed quantum group classification to the coding and non-coding regions in each gene for sorting.

First, a string of DNA sequences is treated as a stream of information, considering the amount of information (i.e., entropy) contained in its unit sequence. Intuitively, we believe that the greater the repetition, the smaller is the amount of information.

Set the sequence $L = (a_1, a_2, \dots, a_n)$. The amount of information in the first m characters is $f_m(l)$, i.e.,

$$g_m(l) = f_m(l) - f_{m-1}(l) \quad (4.9)$$

where $g_m(l)$ is the amount of information added after adding the m -th letter. Then, from (4.9), $f_n(l) = \sum_{i=1}^n g_i(l)$ can be obtained. Note that $f_n(l)$ is the amount of information contained in the entire sequence. Further, $f_n(l)/|l|$ is the amount of information per unit length. Now, the problem is reduced to finding a suitable $g_m(l)$.

In this regard, we assume that $g_m(l)$ should have the following properties:

Nature 4.1. $g_m(l) > 0$; it is more or less with a certain amount of information when adding a character;

Nature 4.2. The more the m -th character (or a shorter sequence ending with it) is repeated with the previous sequence (information flow), the smaller is the value of $g_m(l)$;

Nature 4.3. The more the m -th character (or a shorter sequence ending with it) is repeated with the character that is closer to it, the smaller is the value of $g_m(l)$ and vice versa;

Nature 4.4. $f_0(l) = 0$.

Therefore, we construct the following functions

$$g_m(l) = \frac{b}{b + t_1\sigma_1 + t_2\sigma_2 + \dots + t_p\sigma_p} \quad (4.10)$$

where b is a small positive number set to prevent the denominator from being zero, $\sigma_i = \sum_{t=1}^n a^t \delta_{it}$, and

$$\delta_{it} = \begin{cases} 1, & \text{the } i \text{ string ending at the end of the } m-t \text{ character is exactly} \\ & \text{the same as the } i \text{ string ending with the } t \text{ character} \\ 0, & \text{otherwise} \end{cases}$$

where a is a number less than 1 and p is a fixed positive number. After repeating the machine search, we take $p = 6$, i.e., only check length 1 to 6 of the string, because a string that is too repetitive is very rare. In addition, take $a = 0.392$, $b = 0.1$, and $c = 3$.

Using MATLAB, we extract the signature sequence consisting of four bases, i.e., A, T, G, and C. The number of characteristic sequences of each chromosome is 100, 76, 86, 76, and 93, respectively.

On the quantum hyperplane, suppose that there are n feature sequences $\{x_i\}$, which are collections of feature sequences of different lengths. For a DNA sequence of length L , we can obtain the mathematical representation of this DNA sequence by

the following calculations

$$V_{ij} = \lambda \cdot \frac{T_i}{L - l_i + 1} \cdot \frac{l_i}{l_j} \quad (i, j = 1, 2, \dots, n) \quad (4.11)$$

where V_{ij} is the element of row i and row j of group V . In addition, the elements of V in the row are arranged by the priority principle in accordance with the length in ascending order, and the elements having the same length are arranged in the order of the characters. Further, T_i is the total number of times that the i -th feature sequence appears in this DNA sequence, and l_i and l_j are the lengths of the i -th and j -th feature sequences, respectively. Note that λ is a magnification factor that is used to avoid calculation of the overflow for extremely small component values.

After constructing the group V , call the algorithm (see Appendix 17.5) and judge whether V satisfies the definition of a quantum group. Then, apply the symmetric linear transformation learning algorithm 4.1 of the quantum group to quantum group V and obtain the structure of the classical quantum group that is required. Finally, we can use the quantum group classifier classification algorithm 4.2 for classification.

In the experiment, we divide the data into five segments on average for each chromosome and take two segments as the training set at random. The judgment function is obtained. The remaining three segments are used as the test set. We compare the accuracy of the classification of the five chromosomes using the quantum group classifier with the classification accuracy of the SVM algorithm, as shown in Table 4.4.

Tab. 4.4: Comparison of quantum group classifier and SVM classification.

Chromosome number	1	2	3	4	5
SVM classification accuracy (100%)	97.34	92.29	94.76	92.21	97.31
Quantum group classifier classification accuracy (100%)	97.58	93.44	95.25	92.57	97.93

4.2.3.3 Experiment analysis

In the process of feature extraction, the amount of information contained in each DNA is taken into account. We only consider the characteristics of a certain aspect. Hence, there is always something unsatisfactory. We should take the cyclical frequency at which a single letter appears and other factors into account. Simultaneously, considering the local properties of the elements in the sequence and the global properties of the sequences, the characteristics of the sequences can be reflected and the classification is more scientific.

From the experimental results, on the second and fourth chromosomes, the classification accuracies of this method and the SVM method are found to be low. This is because the second and fourth chromosomes on the extraction of the feature sequence are relatively small, but the accuracy of our method is significantly higher than that

of the SVM method. On the first, third, and fifth chromosomes, the accuracies of both methods are similar. Experiments show that the quantum group classifier proposed in this paper can achieve good classification even when the feature sequence is relatively small, and the classification accuracy is relatively high.

4.3 Application of quantum group learning algorithm in molecular docking

Drug molecular docking design is not only an effective way for finding new drugs but also an ideal way for large-scale database screening. As the drug molecule docking involved in the search space is very large, the computing environment and docking time of each docking process have stringent requirements. Based on the basic theory of quantum group and the molecular docking problem in drug molecule design, this paper presents a new method of drug molecular docking design based on quantum group generators, which further enriches the theory of cluster learning. First, a molecular docking model based on quantum group is proposed, and a molecular matching algorithm based on the quantum group generator is designed. The algorithm is used to dock the molecules in the small molecule database. The test results are compared and analysed in terms of the time and precision using AutoDock 4.0 and molecular dynamics algorithms [15, 25].

4.3.1 Introduction to molecular docking algorithm

The method of computer-aided drug design can be divided into drug-based design based on small molecules, drug design based on receptor structure, and the computer combination chemical method. With the development of X-ray diffraction and nuclear magnetic resonance technology, three-dimensional structures of biological macromolecules are being increasingly determined. Thus, drug design based on receptor structure has more practical significance.

Molecular docking is a very important method of drug discovery and design, and it is also a hot research topic worldwide. It uses computer pattern recognition and optimisation techniques to search for a geometrically and chemically matched molecule that can target a specific drug action in a three-dimensional structure database to achieve computer-aided drug screening (virtual screening). Virtual screening saves most of the compound synthesis or purchase as well as the actual time and cost of bio-testing, thus accelerating the process of new drug development and reducing costs significantly. The advantages and disadvantages of molecular docking are keys to the success of virtual screening. Based on these core problems, this chapter uses machine learning and tries to provide a new solution.

Molecular docking refers to the ligand molecules placed in the active site of the receptor to find its reasonable orientation and conformation. In the process of molecular docking, we focus on the following issues: how to match the ligand and the receptor, how to determine the best match, and how to achieve the best match. According to the structure of the ligand molecules for docking, molecular docking can be divided into whole molecular docking and fragment-based molecular docking. The former refers to the use of a specific search algorithm to consider the ability of ligand molecules in the receptor binding site, and finds the optimal combination of a given score function, which is representative of the software DOCK 3.5 and AutoDock. The latter is a collection of ligands as a collection of structural fragments, one or several basic fragments of which are first collected in the receptor binding pocket and then in the active site to build the rest of the molecule. Ultimately, the theoretically best combination is obtained. Some representative software programs are DOCK 4, DOCK 5, FlexiDock, and FlexX. The fragment-based molecular docking method is usually faster than the overall molecular docking method. However, because the fragment-based molecular docking method only considers the local interaction between the receptor binding site and the constructed fragment, it is difficult to evaluate whether the optimised structure is the global optimal conformational representation of the ligand molecule. In addition to the fragment-based molecular docking method, the docking results are susceptible to the effects of fragments and their locations.

The molecular docking method can be divided into three categories according to the degree of simplification: rigid butt joint, semi-flexible docking, and flexible docking. Rigid butt means that the conformation of the research system does not change during the docking process. Semi-flexible docking refers to the docking process in which the study system, especially the conformation of the ligand, allows a certain range of changes. Flexible docking refers to the docking process in which the study system is free to change. Table 4.5 lists representative molecular docking methods.

Typical docking algorithms are simulated annealing (SA), genetic algorithm (GA), tabu algorithm (TA), Monte Carlo (MC), and their variants. Some other random algorithms have recently emerged, such as random clustering search algorithms, stochastic approximation with smoothing (SAS) algorithms [26], and potential smoothing search (PSS) algorithms [27]. In addition, the molecular dynamics (MD) method [28] is often used for conformational search.

AutoDock [29, 30] was developed by the Scripps Institute Olson Group, and the latest version is 4.0. In AutoDock, the Lamarckian genetic algorithm (LGA) by Morris et al. [29, 30] introduces local search operations on the basis of general genetic algorithm selection, hybridisation, and mutation operations. Each chromosome in the algorithm consists of real gene strings, which include three Cartesian coordinates that identify the position of the ligand, a quaternary group used to describe the orientation of the ligand, and a variable of the rotatable bond of the ligand. The gene sequence that encodes the angle of the rotated key is defined by the torsion tree generated by AutoTors. The disadvantage of AutoDock is that it can only dock a single molecule file

Tab. 4.5: Representative molecular docking methods.

Method	Optimization method	Evaluation function	Speed
DOCK	Fragment growth	Molecular force field, surface matching/chemical environment matching score	Fast
eHiTS	System search	Semiempirical free energy evaluation function	Fast
EUDOC	System search	Molecular force field	Fast
FlexX	Fragment growth	Semiempirical free energy evaluation function	Fast
ICM-Dock	Random global optimization	Semiempirical free energy evaluation function	Fast
LigandFit	Monte Carlo simulation	Semiempirical free energy evaluation function	Fast
FlexiDock	Genetic algorithm	Molecular force field	Slow
Fred	System search	Geometric matching score, semiempirical free energy evaluation function	Fast
Slide	System search	Semiempirical free energy evaluation function	Fast
Affinity	Monte Carlo/molecular mechanics/molecular dynamics	Molecular force field	Slow
AutoDock	Genetic algorithm	Semiempirical free energy evaluation function	General
Glide	System search	Semiempirical free energy evaluation function	General
Gold	Genetic algorithm	Semiempirical free energy evaluation function	Fast
QXP	Monte Carlo simulation	Molecular force field	General

(a file contains only one molecule); it cannot filter multi-molecular files. Hence, for files containing multiple molecules (such as `multi_mol2` and `SDF`), we need to break them down into multiple single-molecule files.

Molecular docking can also be modelled by molecular dynamics (MD), but because this approach tracks a path from the initial orientation to the low-energy conformation, the simulation process is time-consuming. In addition, it must perform different initial orientation simulations to obtain a statistically significant result. In the MD method, the force field is used to calculate the force of each atom of the simulated system. The velocity and acceleration are calculated according to the Newtonian mechanics formula, and the atoms then move in a small range at a given time step. A good example of this method was presented by Di Nola et al. [31], who exploited the temperature to make the system jump out of the local optimal solution. Given and Gilson [32] proposed a four-stage docking protocol based on MD. First, MD is used to generate a conformational set of ligands by heating and cooling processes, where the purpose of heating is to cause structural disturbances and the purpose of cooling to optimise the structure. The ligand is then randomly placed at the receptor binding site, repeated several times, and the third stage is minimised. In the fourth stage, the most stable conformation is further considered by using the alternating equation of heating and cooling with the MD equation. The purpose of the last stage is to explore the conformation space more closely in the vicinity of these stable conformations.

The current docking algorithm typically uses a fast and simple scoring function to search for conformations and optimal structures. However, recent studies have shown that these algorithms have some obvious shortcomings and require the introduction of more efficient tools for the study of molecular docking algorithms. The generalised concept of quantum group as a classical Lie group and Lie algebra has the properties of group and manifold, which are closely related to non-exchange geometry and quantum symmetry, and involve deep intrinsic theory. In 1991, Tien-Ren Tsao et al. presented a new method for neural computation of three-dimensional rigid body motion with noisy two-dimensional image flow [33]. Therefore, in this paper, we propose a molecular docking drug design method based on quantum group, and we compare this algorithm with AutoDock 4.0 and the MD-based molecular docking algorithm for the problem of molecular docking in drug design.

4.3.2 Molecular docking design model based on quantum group [14–16]

4.3.2.1 Mathematical model of molecular docking

The molecular docking problem is to find the most stable binding state between the interacting molecules. By placing the small molecular compound of known three-dimensional structure into the binding site of the target molecule, we can optimise the position, orientation, and conformation of the small molecule compounds, find the best binding conformation of the small molecules and target biomolecules, and calculate or predict the interaction energy between the ligand molecules and the biological macromolecules. In mathematical programming, the minimisation of the binding free energy of an interaction system can usually be described as an optimisation problem as follows

$$\begin{cases} \min & f(x) \\ \text{s.t.} & g_k(x) \leq 0, \quad k = 1, 2, \dots, q \end{cases} \quad (4.12)$$

where $x = \{T_x, T_y, T_z, R_x, R_y, R_z, T_{b1}, T_{b2}, \dots, T_{bn}\}$, T is a design variable of q ($q = n + 6$) dimensions, $T_{b1}, T_{b2}, \dots, T_{bn}$ is a rotatable bond of a ligand molecule that describes the conformational information of the ligand molecule, and n is the number of rotatable bonds. Further, T_x, T_y, T_z, R_x, R_y , and R_z denote the geometric centre of the ligand molecule and the degree of rotation, corresponding to the orientation of the ligand molecule, and $g(x)$ is the size constraint of the design variable

$$\begin{cases} \underline{X} \leq T_x \leq \bar{X} \\ \underline{Y} \leq T_y \leq \bar{Y} \\ \underline{Z} \leq T_z \leq \bar{Z} \\ -\pi \leq \text{angel} \leq \pi, \quad \text{angel} = R_x, R_y, R_z, T_{b1}, \dots, T_{bn} \end{cases} \quad (4.13)$$

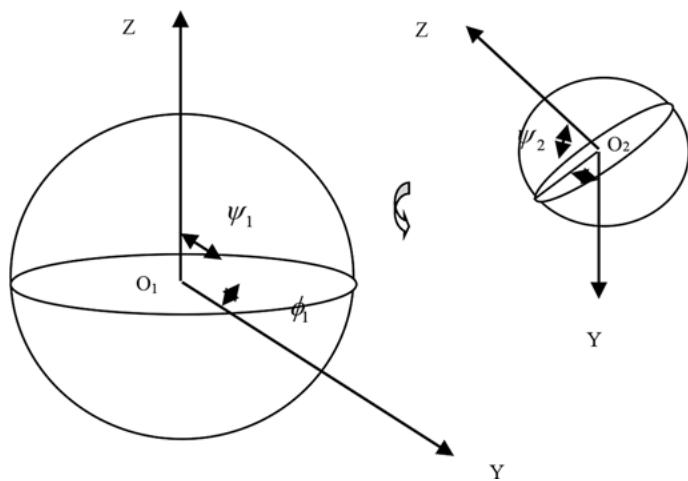


Fig. 4.1: Spherical sketch of molecular docking.

In this model, the optimised conformation of the ligand consists of three translational components (T_x, T_y, T_z), three rotational components (R_x, R_y, R_z), and rotation key ($T_{b1}, T_{b2}, \dots, T_{bn}$) of the ligand. The range of the translation variable is limited to the outer cuboid in the receptor active site, and the change range of the rotation variable is $[-\pi, \pi]$. Furthermore, $f(x)$ is the objective function, i.e., scoring function.

Figure 4.1 shows a spherical representation of molecular docking, where ψ_1 and ϕ_1 determine the orientation of the geometric centre O_2 of the ligand molecule relative to the acceptor molecule; ψ_2 and ϕ_2 determine the orientation of the geometric centre O_1 of the ligand molecule relative to the acceptor molecule; and ϕ is the corner of the junction of the two molecules geometrically.

4.3.2.2 Quantum group model for molecular docking

We know that any group is generated by its generators. With a set of generators, it is very convenient to represent a large group. For example, the quantum algebra of quantum group $GL_q(3)$ is the linear space of the infinitesimal generator of a single-parameter single quantum group of quantum groups. The three-dimensional Euclidean group is composed of six single parameter subgroups, so the dimension of the quantum algebra is 6. The infinitesimal generator of quantum group $GL_q(3)$ is derived from a single-parameter quantum subgroup at $t = 0$.

If we use t as a parameter, three rotation parameters and three translation parameters are taken as the coordinates of point $G(X, Y, Z)$, and we can get six infinitesimal generating subgroups of a single-parameter rotation subgroup and one parameter shift subgroup. Thus, any infinitesimal transform of quantum group $GL_q(3)$ is a linear combination of six infinitesimal generators. The single-parameter rotation around the

X , Y , and Z axes has the following matrix form on R^3

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi & -\sin \varphi \\ 0 & \sin \varphi & \cos \varphi \end{bmatrix}, \quad \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}, \quad \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where φ , θ , and ψ represent the rotation angles of the X , Y , and Z axes, respectively, corresponding to φ , θ , and ψ in Fig. 4.1, and the corresponding infinitesimal generation is

$$e_1 = \begin{bmatrix} q & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \quad e_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & q & 0 \\ -1 & 0 & 0 \end{bmatrix}, \quad e_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & q \end{bmatrix}.$$

The translation subgroup has no representation of the matrix on R^3 , but is defined in the vector space of the differential function on R^3 , expressed as a displacement operator

$$T_x(\xi) : f(X, Y, Z) \rightarrow f(X + \xi, Y, Z)$$

$$T_y(\eta) : f(X, Y, Z) \rightarrow f(X, Y + \eta, Z)$$

$$T_z(\zeta) : f(X, Y, Z) \rightarrow f(X, Y, Z + \zeta)$$

where ξ , η , and ζ represent the translational directions on the X , Y , and Z axes, respectively, and the related infinitesimal generators are denoted as partial differential operators

$$e_4 = \frac{\partial}{\partial X}, \quad e_5 = \frac{\partial}{\partial Y}, \quad e_6 = \frac{\partial}{\partial Z}.$$

With these six infinitesimal generators, the infinitesimal transforms e of any three-dimensional quantum group can be expressed as

$$e = \sum_{i=1}^6 v_i e_i \tag{4.14}$$

where v_i ($i = 1, \dots, 6$) is a coefficient of e_i .

As the acceptor molecule is assumed to be rigid, only changes in ligand conformations are considered in the docking process. The above treatment allows the calculation of molecular docking to be related only to the Descartes coordinates of the atoms in the ligand and thus can be obtained by solving the optimisation problem (4.12)–(4.13). The range of the translation variable is limited to the cavity of the receiver. To avoid the cumbersome work of solving the exact boundary of the receptor cavity, the external box of the space occupied by the sphere in the surface of the receiver is defined as the variable space. The range of rotation and rotation variables is $[-\pi, \pi]$. For each conformation produced by the optimisation process, as long as one of the atoms falls within the range of the rectangular sphere, it is regarded as an invalid con-

formation, at which time the score of the conformation is not calculated and a new conformation is regenerated until the new conformation is an effective conformation.

A global search is performed on the entire conformation of the ligand, i.e., the position space of the ligand is searched by the translation component in the model. The orientation space of the ligand is searched by rotating components. The flexible conformational space of the ligand is searched by the flex key component. This eliminates a series of preparations, such as segmentation of the ligand, definition of anchor points, division of layers, calculation of anchor distance, and search for matching. After receiving the input file for data initialisation, the conformation search is carried out directly. As a result, the docking computation time is significantly reduced.

4.3.3 Molecular matching algorithm based on quantum group generators

In the molecular docking, if the ligand is represented by m spheres, the number of negative images in the active pocket of the receptor is n , and the number of possible matches between them is $n!/(n - m)!$. To simplify the calculation, we use the following algorithm for matching:

- (1) First, match the ball i of the ligand sphere with a negative k of the active site.
- (2) The distance d_{ij} between the ligand sphere i and the other ligand ball j is then calculated, and the distance d_{kl} between the negative image k and the other negative image l is calculated. If the following conditions are met, then the ligand sphere j and the negative image k can be effectively matched, where the value of ε is generally between 1.0–2.0 Å.

$$|d_{ij} - d_{kl}| < \varepsilon . \quad (4.15)$$

- (3) After the second match point is determined, the third match point can be determined. The third collocation must satisfy two constraints as shown in (4.16). Likewise, the fourth point must satisfy three constraints as shown in (4.16). The number of matching points for each valid match must not be less than 4.
- (4) According to the above operation, the receptor and ligand matching point set is obtained, and further adjustment is performed. There are three purposes for the adjustment: the internal distance is relatively rough and needs to be further adjusted; some non-matching atoms may be in an unreasonable position and need to be further adjusted; perform coordinate transformation to determine the ligand relative to the receptor position. In the adjustment, the pairing distance is first converted into Cartesian coordinates. The ligand set is superimposed on the corresponding negative image of the receptor by translation and rotation, and this process is carried out using the least-squares method. The magnitude of the error from the least-squares method corresponds to the overlap between the ligand sphere and the negative image of the receptor.

- (5) If the error in the results given by the least squares of the ligand and the negative of the receptor is within a certain range, the alignment of the ligand is adjusted so that the match between the ligand and the receptor is more reasonable. Here, we define a simple function to evaluate the match between the ligand and the receptor

$$E = \sum (r_i + r_j - d_{ik}) \quad (4.16)$$

where r_i and r_k are the Van der Waals radii of ligand atoms and acceptor atoms, respectively, and d_{ik} is the distance between atoms. The smaller the value of E , the better the match. When optimised, let the ligand atoms move to the nearest negative image, optimise E , and find the optimum position. The purpose of this step is not to adjust the ligand molecules to a comprehensive optimisation but to remove some obviously unreasonable institutions. The spatial orientation of all the ligands obtained by this step will be further evaluated using the scoring function.

Scoring functions usually have the following categories: the score function based on the force field uses non-bond interaction as a conformation score, and sometimes adds to the solvent effect considerations. The empirical scoring function calculates a set of training sets of protein-ligand complexes based on known binding constants, and uses the multivariate linear regression method to fit its intrinsic structural functional coefficients. The knowledge-based scoring function uses statistical data from the protein-ligand interaction in the X-ray diffraction structure database as the scoring criteria. Here, we use the score function based on AMBER force field

$$f(x) = E = \sum \sum \left(\frac{A_{ij}}{r_{ij}^a} - \frac{B_{ij}}{r_{ij}^b} + 332.0 \frac{q_i q_j}{D r_{ij}} \right) \quad (4.17)$$

where r_{ij} is the distance between atom i in the ligand molecule and atom j in the acceptor molecule; A_{ij} , B_{ij} is the van der Waals parameter; a , b are the van der Waals force rejection and attraction index, respectively, using the Lennard-Jones (12, 6) potential function; q_i , q_j are the charge of atom i and atom j , respectively; D is the dielectric constant; and 332 is a factor that translates electrostatic potential energy into a standard unit (kcal/mol).

The molecular docking method can be described as a combination of two key aspects: one is the energy function between the ligand and the receptor, and the other is the matching algorithm. The above matching algorithm is referred to as QGDock (quantum group docking).

4.3.4 Molecular Docking Simulation based on Quantum Group

The basic flow of the molecular docking simulation based on quantum group as described in this section is as follows:

- (1) Collect information on the structure of small molecule compounds published in the literature and form a two-dimensional (2D) small molecule database. The atom types and chemical bonds of each small molecule are assigned, and the 2D structure is transformed into a three-dimensional (3D) structure and optimised to form a 3D small molecular database;
- (2) Read the file of protein and small molecules, mainly the mol2 file format; the file contains the type of atom, coordinates, and type of key;
- (3) Determine the atomic type of protein and small molecule ligands;
- (4) The protonation and atomic charge attribution of the bio-macro-molecule (protein) are carried out, the structure is optimised to determine the small molecule binding site, and the calculation grid is constructed.
- (5) Randomly initialise the position and angle of proteins and small molecular ligands;
- (6) Run the QGDock algorithm with the 3D small molecule database of each compound docked to the biological macromolecule active sites. Use the quantum group 111 translational operator and the rotation operator to optimise the coordinates and angle of the small molecule ligand, and score, i.e., calculate the binding strength of small molecules to biological macromolecules (binding free energy);
- (7) Analyse the results, calculate the root mean square deviation (RMSD), and output the results q .

4.3.5 Experimental results and analysis

The molecular docking algorithm was simulated on a data set consisting of 31 protein-ligand complexes. The data set was obtained from the LPDB database (<http://lpdb.scripps.edu/>) used in Reference [34]. The standard for data selection is that each protein has at least two ligands (except for the intestinal fatty acid binding protein) in the PDB, and the rotation bonds of each ligand varies from 1 to 30, as shown in Table 4.6.

Tab. 4.6: Experimental data sets (numbers next to each ligand represent the number of rotation keys).

Protein	Protein – ligand complex PDB index number
Insulin	3ptb(3),1tnj(2),1tnj(3),1tnk(4),1tni(5),1tpp(7),1pph(11)
Cytochrome P450 _{cam}	1phff(1),1phgf(5),2cpp(3)
Sialidase	1nsc(12),1nsd(11),1nnb(11)
Carboxypeptidase	1cbx(5),3cpa(8),6cpa(16)
L-gelatin aldehydes	1abe(4),1abf(5),5abp(6)
E-thrombin	1etr(15),1ets(13),1ett(11)
Thermophilic protease	3tmn(10),5tlm(14),6tmn(20)
Mycelium pigment	1apt(30),1apu(29)
Intestinal fatty acid binding protein	2ifb(15)
Carbonic anhydrase II	1cil(6),1okl(5),1cnx(13)

These protein files are stored in the PDB file format. In the experiment, the protein molecules were pre-treated by AutoDock 4.0 software, and all the water molecules and metal ions in the crystal structure were removed. They are divided into two parts, namely the receptor and the ligand, which are stored as two mol2 files.

In each docking simulation, each raster map consists of a three-dimensional lattice surrounded by uniformly distributed dots and is located in the middle of the active protein receptor. Each point in the raster map stores the potential energy of a ‘detected’ atom based on its interaction with the macromolecule, e.g., a carbon grid whose internal point corresponds to the potential energy between the location of the carbon atom and all the carbon atoms on the protein receptor. Since the grid spacing is between

Tab. 4.7: comparison of the root mean square deviations of different docking algorithms.

ID	Protein – ligand complex PDB index number	Number of rotary keys	Autodock	MD	QGDock
1	3ptb	3	0.80	0.56	0.54
2	1tng	2	0.62	0.69	0.67
3	1ynj	3	1.21	1.50	1.42
4	1tnk	4	1.69	1.14	1.16
5	1tni	5	2.61	2.22	2.21
6	1tpp	7	1.80	2.53	1.71
7	1pph	11	5.14	0.53	1.44
8	1phf	1	2.09	1.70	1.20
9	1phg	5	3.52	1.08	0.87
10	2cpp	3	3.40	3.27	2.53
11	1nsc	12	1.40	1.40	1.32
12	1nsd	11	1.20	1.85	1.04
13	1nnb	11	0.92	3.97	0.92
14	1cbx	5	1.33	0.92	0.62
15	3cpa	8	2.22	2.22	1.51
16	6cpa	16	8.30	4.00	4.96
17	1abe	4	0.16	0.56	0.36
18	1abf	5	0.48	0.70	0.51
19	5abp	6	0.48	0.51	0.48
20	1etr	15	4.61	1.09	1.02
21	1ets	13	5.06	1.97	2.11
22	1ett	11	8.12	0.82	1.30
23	3tmn	10	4.51	3.65	1.76
24	5tlm	14	5.34	1.21	1.21
25	6tmn	20	8.72	2.21	2.18
26	1apt	30	1.89	4.97	1.98
27	1apu	29	9.10	1.33	1.32
28	2ifb	15	3.09	5.19	0.97
29	1cil	6	5.81	1.86	1.85
30	1okl	5	8.54	2.84	3.03
31	1cnx	13	10.9	6.20	5.19

0.25 Å and 1 Å, there is no significant difference in docking accuracy. In the experiment, we take the grid spacing of 1 Å. The position of each point in these grids corresponds to a linear combination of infinitesimal generators in quantum group $GL_q(3)$, and the potential energy of interaction between atoms is the coefficient of (4.1). The move operation corresponds to its translation operator, and the rotation operation corresponds to its rotation operator.

The QGDock algorithm was used to calculate the 31 protein-ligand complexes, and each complex was calculated 10 times. The best result was obtained, i.e., the lowest energy calculation result, and the RMSD value was obtained. Table 4.7 and Figure 4.2 show the root mean square deviation (RMSD) comparison of three different docking algorithms, namely AutoDock4.0, the molecular dynamics (MD) method, and quantum confinement based on molecular docking (QGDock). If the docking small molecule conformation and protein structure of the reference molecular conformation are relatively close, then the root mean square deviation is relatively small, and the docking is more successful. Usually, the RMSD value is less than 2.0 Å, indicating docking to a more accurate position.

We define the algorithm's docking accuracy (DA) as follows

$$DA = f_{rmsd} \leq 2 + 0.5(f_{rmsd} \leq 3 - f_{rmsd} \leq 2) \quad (4.18)$$

where $f_{rmsd} \leq a$ is the fraction in the crystal structure when the root mean square deviation of all non-hydrogen ligand atoms between the lowest position of the small molecule potential energy and the ligand position is $\leq a\text{Å}$.

Tables 4.8 and 4.9 compare our docking algorithm with AutoDock 4.0 and MD methods in terms of the docking time and docking accuracy. From Figures 4.2 and 4.3, we can see that the quantum docking algorithm based on quantum group is more accurate than the other two algorithms. The docking time is shorter than that of AutoDock but longer than that of the MD method.

Tab. 4.8: Comparison of different docking simulations for different proteins.

Protein ID	Protein	AutoDock	MD	QGDock
1	Insulin	391	182	281
2	Cytochrome P450 _{cam}	297	87	184
3	Sialidase	645	121	311
4	Carboxypeptidase	614	125	215
5	L-gelatin aldehydes	368	98	176
6	E-thrombin	1185	303	503
7	Thermophilic protease	798	294	348
8	Mycelium pigment	1143	290	276
9	Intestinal fatty acid binding protein	572	489	345
10	Carbonic anhydrase II	520	304	437

Tab. 4.9: Comparison of precision (DA) for different docking algorithms.

	Autodock	MD	QGDock
Docking accuracy (100%)	0.68	0.73	0.82

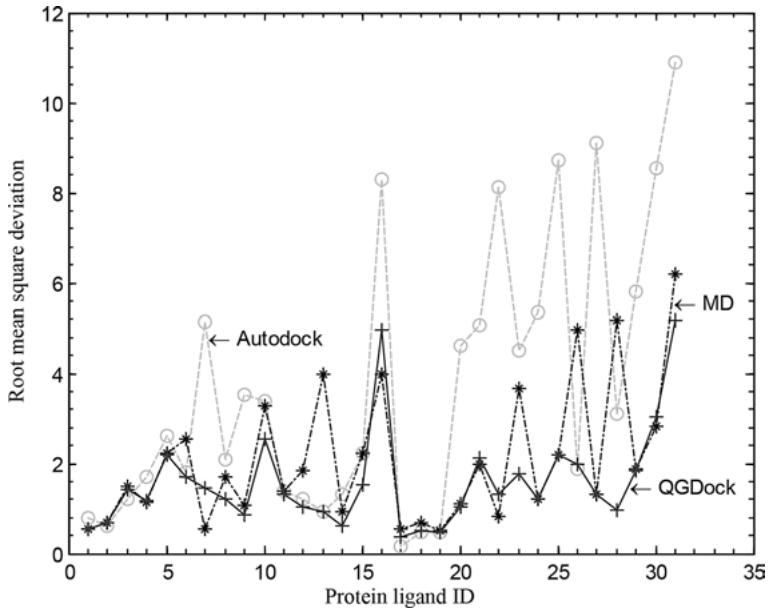


Fig. 4.2: Comparison of the root mean square deviation of different docking algorithms.

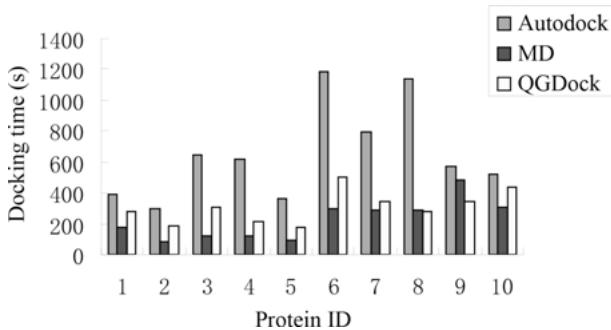


Fig. 4.3: Comparison of docking time for different protein docking proteins.

As we use the infinitesimal sub-generation method based on quantum group to control the search space, we further enhance the flexibility and validity of the search. Experiments show that our method can improve the precision of molecular docking while also improving its speed.

4.4 Summary

The quantum group was introduced into the field of machine learning. The construction method of the quantum group classifier and an application of the quantum group learning algorithm in molecular docking were presented.

Bibliography

- [1] Drinfel'd VG. Quantum groups. Proceedings ICM, 1986, 798–820.
- [2] Nasios N, Adrian NB. Kernel-based classification using quantum mechanics, Pattern Recognition, 2007, 40(3), 875–889.
- [3] Kondor IR. Group theoretical methods in machine learning. Columbia University, 2008.
- [4] Xu H, Li FZ. Lie Group Machine Learning's Axioms Hypothesize. Proceedings of 2006 IEEE International Conference on Granular Computing, 401–404.
- [5] Xu H, Li FZ. The Design of SU(n) Classifier of Lie Group Machine learning (LML). Journal of Computational Information Systems, 2005, 1(4), 835–841.
- [6] Xu H, Li FZ. Geometry Algorithms of Dynkin Diagrams in Lie Group Machine Learning. Journal of Nanchang Institute of Technology, 2006, 25(2), 75–78.
- [7] Xu H, Li FZ. Study on Lie Group Machine Learning. Journal of Computational Information Systems, 2005, 1(4), 843–849.
- [8] Xu H, Li FZ. Algorithms of Dynkin Diagrams in Lie Group Machine Learning. Journal of Communication and Computer, 2006, 4(3), 13–17.
- [9] Xu H, Li FZ. SO(3) Classifier of Lie Group Machine Learning. Journal of Communication and Computer, 2006, 4(3), 76–80.
- [10] Chen F, Li FZ. Orbit Generated Theory of Learning Subspace and its Algorithm in Lie–Group Machine Learning (LML). Journal of Suzhou University, 2007, 23(1), 61–66.
- [11] Chen F, Li FZ. Learning Subspace Orbit Generation Lattice Algorithm Research under the Action of $GL_n(F_q)$ Group in Lie Group Machine Learning (LML), 2006, 36–46.
- [12] Xu H. Research and Application on Lie Group Machine Learning Models. Suzhou: Soochow University, 2007.
- [13] Chen F. Research and Application on Orbit Generated Algorithm of Learning Subspace in Lie Group Machine Learning(LML). Suzhou: Soochow University, 2007.
- [14] He SP. Research on Quantum Group Classifier in Lie Group Machine Learning. Suzhou: Soochow University, 2008.
- [15] He SP, Li FZ. A Molecular Docking Drug Design Algorithm Based on Quantum Group. Journal of Nanjing University, 2008, 44(5), 512–519.
- [16] He SP, Li FZ. A Symmetrical Linear Transform Learning Algorithm on Quantum Group. Proceedings of 2007 IEEE International Conference on Granular Computing, 2007, 724–727.
- [17] Adleman LM. Molecular computation of solutions to combination problems. Science, 1994, 266(11), 1021–1023.
- [18] Baldi P, Brunak S. Bioinformatics: the machine learning approach. MIT Press, 1998.
- [19] Durbin R, Eddy S, Krogh A, et al. Biological sequence analysis, Probabilistic models of proteins and nucleic acids, London. Cambridge University Press, 1998.
- [20] Simek K, Fujarewicz K, Swierniak A, et al. Using SVD and SVM methods for methods for selection, classification, clustering and modeling of DNA microarray data. Engineering Application of Artificial Intelligence, 2004, 17(4), 417–427.

- [21] Yu XJ, Cao JP, Cai JD, et al. Predicting rRNA-, RNA-, and DNA-binding proteins from primary structure with support vector machines. *Journal of Theoretical Biology*, 2006, 240(2), 175–184.
- [22] Cai YD, Lin SL. Support vector machines for predicting rRNA-, RNA-, and DNA-binding proteins from amino acid sequence. *Biochimica et Biophysica Acta*, 2003, 1648(1–2), 127–133.
- [23] Almeida JS, Carrico JA, Maretzek A, et al. Analysis of genomic sequence by chaos game representation. *Bioinformatics*, 2001, 17(5), 429–437.
- [24] Zhang CT. A symmetrical theory of DNA sequences and its applications [J]. *Journal of Theoretical Biology*, 1997, 187(3), 297–306.
- [25] Xu XJ, Hou TJ, Qiao XB, et al. Computer Aided Design of Drug Molecular. Beijing: Chemical Industry Press, 2004.
- [26] Diller DJ, Berlinde CLM. A critical evaluation of several global optimization algorithms for the purpose of molecular docking. *Journal of Computational Chemistry*, 1999, 20(16), 1740–1751.
- [27] Hart RK, Pappu RV, Ponder JW. Exploring the similarities between potential smoothing and simulated annealing. *Journal of Computational Chemistry*, 2000, 21(7), 531–552.
- [28] Taufer M, Crowley M, Price D, et al. Study of a highly accurate and fast protein-ligand docking algorithm based on molecular dynamics. *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04)-Workshop 9*, 2004, Santa Fe, New Mexico.
- [29] Morris GM, Goodsell DS, Halliday RS, et al. Automated docking using a Lamarckian genetic algorithm and empirical binding free energy function. *Journal of Computational Chemistry*, 1998 19(14), 1639–1662.
- [30] Morris GM, Goodsell DS, Huey R, et al. Distributed Automaten docking of flexible ligands to proteins: Parallel applications of AutoDock 2.4. *Journal of Computer-Aided Molecular Design*, 1996, (10), 293–304.
- [31] Mangoni R, Roccatano D, Di NA. Docking of flexible ligands to flexible receptors in solution by molecular dynamics simulation. *Proteins: Structure, Function, and Genetics*, 1999, 35(2), 153–162.
- [32] Given JA, Gilson JA. A hierarchical method for generating low-energy conformers of a protein-ligand complex. *Proteins: Structure, Function, and Genetics*, 1998, (33), 475–495.
- [33] Tsao TR, Shyu HJ, Libert JM, et al. A Lie group approach to a neural system for three-dimensional interpretation of visual motion. *IEEE transactions on neural networks*, 1991, 2(1), 149–155.
- [34] Sottriffer GM, Flader W, Winger RH, et al. Automated docking of ligands to antibodies: methods and applications. *Methods: A Companion to Methods in Enzymology*, 2000, 20(3), 280–291.

5 Lie group fibre bundle learning

This chapter is divided into four sections. Section 5.1 presents a question. Section 5.2 describes the fibre bundle model. Section 5.3 introduces the learning algorithm of fibre bundles. Finally, Section 5.4 summarises the chapter.

5.1 Problem presentation

Fibre bundle theory is an important part of the study of geometry in the 20th century. In a less strict sense, large-scale differential geometry (including fibre clusters) studies the relationship between the global and local properties of the differential geometry object. The method of connecting and dealing with different geometric spaces and different geometric amounts of fibre bundle theory serves as a feasible mathematical method for studying the global and local relations of data sets. The fibre bundle is a generalised product of the manifold. In short, let E, M be two smooth manifolds. The mapping of E to M is smooth, the affine space at each point on M is composed of an n -dimensional vector set, and (E, M) is the vector bundle on manifold M . Intuitively, the vector bundle E is the result of the product manifold and fibre bonding. Adhesion requires that the linear relationship on the fibre remain unchanged.

Fibre bundle as a mathematical tool has been successfully applied to computer vision, pattern recognition, neural networks, and cybernetics. In computer vision, Nir Sochen et al. used fibre bundle as the main tool and established a framework for nonlinear diffusion. Beltrami [1] successfully used fibre bundles in the merging of concurrent diffusion stereo images [2]. In neural networks, D. W. Pearson developed the fibre bundle of feedforward neural network representation [3]. In pattern recognition, Jin-hui Chao and Jongdae Kim established the fibre bundle model of surface shape and realised the rapid generation of surface shape [4]. In cybernetics, in 1976 and 1983, R. W. Brockett [5, 6] analysed specific questions and pointed out that the general theory of nonlinear control systems should be based on a fibre bundle. In 1996, on the basis of previous studies, M. U. Xiaowu developed a method for establishing the correct geometric frame of a nonlinear control system by means of fibre bundles, and proposed the corresponding theory [7].

A manifold is a local coordinate topological space, mapping from an open set (neighbourhood) of topological spaces to the open subset of Euclidean spaces homeomorphically, accounting for each local coordinate. Its essence is piecewise linear processing. Many methods of manifold learning are based on the properties of the manifold. We have found that some of them use tangent spaces, such as the LTSA local studies of manifolds. This idea has significant implications for the study of this chapter. The fibre bundle learning model and algorithm established in this chapter first considers the tangent bundle (see the relevant section on the relationship between fi-

bre bundle and Lie group machine learning). The local properties of the manifold are approximated by the tangent space of each point on the manifold, and each tangent space composed of the tangent bundle, allowing for the use of fibre bundle as a tool for further analytical processing of manifold structure and its tangent bundle. Refer to Appendix 17.2, 17.3 for the differential geometry knowledge and manifold learning algorithms covered in this chapter.

5.2 Fibre bundle model

5.2.1 Expression of fibre bundles in manifold learning [8, 9]

Fibre bundle is a generalisation of the product on manifolds. The vector $E = \bigcup_{x \in M} F_x$ is the union of the fibres at all points on the bottom manifold M and the vector space family using manifold M as a parameter. The local area can be expressed as the topological product $U \times F$ of the parameter space U and the vector space F , but as a whole, it cannot be represented as an ordinary topological product.

Here is a real-world example. We found that the cactus plant has striking similarities with the fibre bundle structure, as shown in Figure 5.1. Intuitively, the bottom of the fibre bundle is analogous to the cactus bulbs, and the fibre is analogous to the cactus needles. The cactus can be described in the fibre bundle language as follows:

- (1) The manifold made of bulbs is base space M , and all needles constitute the bundle space E
- (2) The projection $\pi: E \rightarrow M$ is defined as the intersection point from any spot on the needle of a cactus to the intersection of the conifer and the bulb.
- (3) Fibre type F is defined as a one-dimensional space R .

In a number of specific issues, the actual data describing the problem is easy to obtain, but the intrinsic information or knowledge hidden in the data is not easy to obtain

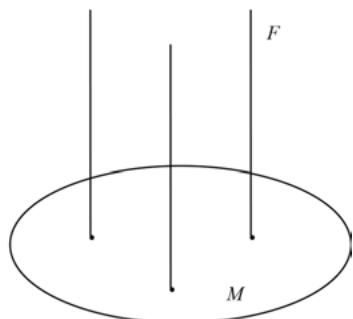
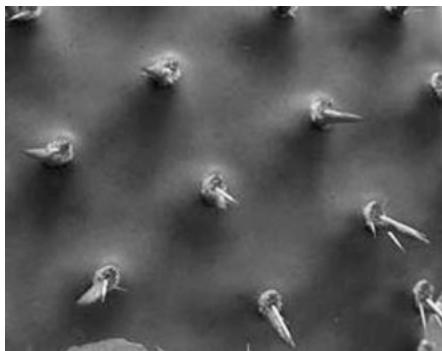


Fig. 5.1: Cactus and fiber bundle schematic.

directly. Deep analysis or deep classification of the data is required to extract useful information. These applications provide opportunities for machine learning. The data objects that are actually processed can be grouped or categorised according to the similarity between objects. After such processing, the data object produces the structure of the class or cluster. This cluster structure is consistent with the fibre bundle structure. Different types of high-dimensional data in clustering analysis may correspond to different subspaces. The dimensions of each subspace may also be different. It is often difficult to get a clustering structure in full space. In fibre bundle language, every subspace generated by the higher-dimensional clustering is a fibre, and the set of all subspaces is a bundle space. In the language of fibre bundles, the bundle space is also known as the entire space. The following discussion is based on this concept.

Definition 5.1. In the manifold study, the fibre bundle is expressed as the six-membered group $\langle X, E, M, F, \pi, G \rangle$, where

X : Input data set $X = \{x_1, x_2, \dots, x_n\}$;

E : The fibre bundle is a union of all the fibres on the bottom manifold.

M : The bottom space is obtained after a certain transformation of the original data set;

F : The typical fibre, which specifies the topological properties of the fibre space, and all the fibres of the fibre bundle satisfy the properties of the typical fibre. It is the true projection of the original high-dimensional data. For example, F is desirable for R^k . Then, all fibres are k -dimensional real spaces.

π : The projection transformation between fibre bundle E and bottom manifold M

$$\pi: E \rightarrow M$$

The fibre F_x is obtained from the inverse transformation on the bottom manifold M at each point x

$$\pi^{-1}: x \rightarrow F_x$$

Fibre F_x is a k -dimensional space, called fibre through x points, then $E = \bigcup_{x \in M} F_x$.

G : is called the structure group, which is a matrix group generated by local coordinate transformation in the bottom space. It gives the structure of the fibre bundle

Definition 5.2. On fibre bundle $\langle X, E, M, F, \pi, G \rangle$, $\forall x'_i \in X' \subseteq M, \exists U_\alpha, \ni x'_i \in U_\alpha$; then, $x_i \in \pi^{-1}(U_\alpha)$.

Algorithm 5.1: Fibre cluster generation algorithm (FbundleProduce)

Input: Collection of data sets $X = \{X_i\}$

Output: Fibre bundle $\langle X_i, E_i, M_i, F_i, \pi_i, G_i \rangle$

Process pseudo-code:

For each data set X_i **in** X .

(1) $M_i \leftarrow X_i$

(2) $F_i = \mathbb{R}^D$, $D = X_i$'s dimension, $G_i = GL(D)$

(3) K-means clustering is used to partition X_i so that each sample belongs to a clustering

$$C_{ij}, \quad \text{where } j \text{ is the number of clusters, } E = \{C_{ij}\}$$

(4) The central sample for $\pi_i: x_{ijk} \rightarrow C_{ij}$ is x_{ijo}

5.2.2 Tangent bundle model for manifold learning

5.2.2.1 Tangent bundle and vector fields

From the relevant theory of Appendix 17.6, we can see that manifold M is a topological space of local image R^n , and it is differentiable. Hence, analysis tools can be used to study manifolds. One of the most important methods of analysis is to analyse the infinitesimal of an object, which can make the problem linear and easy to study.

The properties of a function near a point can be expressed by its derivatives. The local properties of the Lie group near a constant element can be expressed by Lie algebra. The property of a curve near a point can be represented by the tangent at the point.

Similarly, for a data manifold M , the properties near each point can be approximated by linear spaces (tangent spaces). The tangent space is composed of tangent vectors, and the tangent vector of p in a certain direction in manifold M is independent of the selection of the coordinate system, i.e., it is not related to the selection of the neighbourhood coordinate card of this point. The main example of a vector bundle is a tangent bundle of a manifold.

Definition 5.3. The union of all points of the tangent space on the manifold M

$$T(M) = \bigcup_p T_p(M)$$

is called the tangent bundle of the manifold M . The tangent bundle $T(M)$ is a $2n$ -dimensional manifold, and the local is a direct product manifold, but the same is not necessarily true for the whole. The tangent vectors at the same point can be added, while the tangent vectors at different points are independent.

Each point has an n -dimensional tangent space through the manifold M , which can be represented by the straight line of the point. The tangent spaces at different points are independent of each other and are represented by parallel lines. The tangent space at each point p is $T_p(M) \simeq R^n$.

In Appendix 17.6, the general definition of the cross section of the fibre bundle theory has been introduced. On this basis, the important concept of vector field $X(x)$ is introduced.

Definition 5.4. Tangent vector field (x) is also called a cross section of the tangent bundle $T(M)$. A tangent vector $X_p \in T_p(M)$ is chosen at each point p on the mani-

fold M . After the local coordinate system is selected, the tangent vector field (x) can be represented as

$$\mathcal{F}(x) = \sum_{j=1}^n \xi^j(x) \partial_j$$

If the function $\xi^j(x)$ is continuous and differentiable, then the vector field (x) is continuous and differentiable.

5.2.2.2 Tangent bundle model for manifold learning

Appendix 17.3 gives a detailed introduction to manifold learning, and a brief description of manifold learning tasks is given below. Consider a high-dimensional sample data set $X = \{x_1, x_2, \dots, x_N\}$, where $x_i \in R^D$, uniformly sampled in a smooth d -dimensional manifold M , where M is the d -dimensional embedded manifold in D -dimensional Euclidean space, and the embedding mapping is $\tau: M \rightarrow R^d$, where $M \subset R^D$, $d \ll D$.

Manifold learning uses the manifold of Euclidean space as the mathematical basis because the local manifolds are Euclidean topological spaces. There is a neighbourhood for every point in space, i.e., a manifold is a local coordinate topological space. Hence, we can deal with the piecewise linear case.

The tangent cluster model for manifold learning is given below.

Definition 5.5. Consider a high-dimensional data set $X = \{x_1, x_2, \dots, x_N\}$, where $x_i \in R^D$, uniformly sampled in smooth d -dimensional manifolds M , where M is the d -dimensional embedded manifold of D -dimensional Euclidean space, and the embedding mapping is $\tau: M \rightarrow R^d$, where $M \subset R^D$, $d \ll D$. The union of all points on the tangent space of X

$$T(X) = \bigcup_p T_p(X)$$

is called the tangent bundle of manifolds X .

Definition 5.6. A tangent bundle model of data manifolds X contains the following elements:

- (1) Tangent bundle $T(X) = \bigcup_p T_p(X)$
- (2) Vector field (x) is called a cross section of the tangent bundle $T(X)$. Select a tangent vector $X_p \in T_p(X)$ at every point p on the manifold X . After the local coordinate system is selected, the tangent vector field (x) can be represented as

$$\mathcal{F}(x) = \sum_{j=1}^n \xi^j(x) \partial_j$$

If the function $\xi^j(x)$ is continuous and differentiable, then the vector field (x) is continuous and differentiable.

(3) Contact D , satisfying $D: (X) \times (X) \rightarrow (X)$:

- a) $D_{Y+fZ}X = D_YX + fD_ZX$
- b) $D_Y(X + \lambda Z) = D_YX + \lambda D_YZ$
- c) $D_Y(fX) = Y(f)X + fD_YX$

where $D_YX = D(X, Y), X, Y, Z \in (X)$, $\lambda \in \mathbb{R}$, $f \in C^\infty(M)$. The fibre bundle $\langle X, E, M, F, \pi, G \rangle$ can be represented as $\langle T(X), X, F, \pi, G \rangle$.

5.2.3 Main fibre bundle model

Definition 5.7. The main bundle $P = P(M, G)$ is such a fibre bundle, the fibre is Lie group manifold G , the structure of cluster P still belongs to G , and perform left transformation on the fibre G .

Since the left and right effects of the Lie group G can be defined, we can define the right effect of group G on P . Its invariant structure group is

$$\begin{aligned} P \times G &\rightarrow P \\ u, a \rightarrow u_a &= R_a u \in P, \quad u \in P, \quad a \in G. \end{aligned}$$

Since the fibre itself is a group manifold, the effect of group G on the main bundle P is free (i.e., except the constant element, there are no fixed points for all functions)

so that each orbit is a homologous group G manifold, i.e., the whole orbit of the action of the group G is considered as a point of orbital space. Obtain the orbital space, i.e., the bottom space $M = P/G$. Then, there is the regular projection of the main bundle P to the bottom manifold M

$$\pi: P \rightarrow M$$

The regular projection preserves the fibre, i.e., the properties

$$\pi(R_a u) = \pi(u) \quad (u \in P, a \in G)$$

The primary bundle P can be denoted as $P(M, G)$, or can be represented as

$$G \mapsto P \xrightarrow{\pi} M$$

The primary bundle P is locally mediocre. There is a homeomorphic mapping at each point on the fibre of the bottom manifold neighbourhood U_a

$$\begin{aligned} \psi_a: \pi(U_a) &\rightarrow U_a \times G \\ u \rightarrow (\pi(u), \psi_a(u)) &= (x, g_a(x)), \quad g_a \in G \end{aligned}$$

The overall structure of the cluster depends on the smoothness of the fibres bonded in the overlapping zone, i.e., when $x \in U_\alpha \cap U_\beta$, on point u of fibre $\pi^{-1}(x)$, we have

$$\psi_\alpha(u) = g_\alpha, \quad \psi_\beta(u) = g_\beta, \quad g_\alpha = \psi_\alpha \psi_\beta^{-1} g_\beta = \lambda_{\alpha\beta}(x) g_\beta$$

The transfer function $\lambda_{\alpha\beta}(x) \in G$ satisfies the compatibility condition. The set of transform functions $\{\lambda_{\alpha\beta}(x)\}$ forms the structural group of the cluster. The structural group of the main bundle P belongs to the standard fibre Lie group G of the cluster. The main bundle P is also called the G -main bundle.

Definition 5.8. The G -main bundle on the m -dimensional smooth manifold M , $\dim G = r$. Let H be an m -dimensional smooth distribution on the cluster space E , or equivalently, H is a smooth m -dimensional tangent subspace field on E . If the following conditions are established:

- (1) At each point $b \in E$, tangent space $T_b E$ has direct factorisation

$$T_b E = H_b \oplus V_b$$

Moreover, the restriction $(\pi^*)|_{H_b}: H_b \rightarrow T_{\pi(b)}M$ of the tangent mapping π^* on H_b is a linear isomorphism;

- (2) The distribution H is invariant under the right action of G , i.e., for any $b \in E$, $g \in G$, we have

$$(R_g)_*(H_b) = H_{b \cdot g}$$

then H is called a connection on the G -main bundle $\pi: E \rightarrow M$.

The concepts of fibre bundle representation and main cluster in manifold learning have been given. The main fibre bundle model of Lie group machine learning is defined below.

Definition 5.9. The main fibre bundle model of Lie group machine learning is expressed as

E : Fibre bundle, is the union of all fibre on the base manifold.

M : Base space.

π : Projection transformation between fibre bundle E and base manifold M .

G : Lie group G , represents both the standard fibre of the main bundle and the structural group.

5.3 Fibre bundle learning algorithm

In the previous section, we gave a model of fibre bundle learning, in which the cut-plexiform model unifies the tangent and tangential vectors and extends the concept of faceted contact using the fibre bundle language. On this basis, we give two algorithms under the fibre bundle learning model.

5.3.1 Vector reduction algorithm based on local principal direction of tangent

5.3.1.1 Algorithm description

As has been proven in the literature [10], the local tangent space can be expressed by the eigenvector of the local sample covariance matrix. It can use the first d eigenvectors of the principal component analysis as the basis, which approximately represent the tangent space, and a cluster consists of the union of the tangential space.

The algorithm first constructs tangent sets, learning from PLTSA. K-means clustering is used for the data set to divide the manifold into t sufficiently small local blocks, and principal component analysis is performed on the block. Taking the first principal component direction to form the vector field Z , it is a section of the tangent cut. Since the main eigenvector of the covariance matrix of each neighbourhood is the global coordinate, we reduce the dimension of Z , and the vector field Z is itself a manifold. By observing the rule of manifold Z , we find the internal structure of the high-dimensional data set. In the dimensionality reduction, we use the classical manifold learning algorithm and compare their effects in the experiment.

Algorithm 5.2: Vector field dimensionality reduction algorithm based on tangent local principal direction (TBFR, Tangent Bundle Field Reduction)

Input: sample set X , dimension d of the low-dimensional embedded space;

Step 1 Find the t centres of X with the k -mean; divide X into t blocks, each expressed as $X_i = [x_{i1}, \dots, x_{ik}]$;

Step 2 Local Principal Component Analysis. Calculate the covariance matrix $(X_i - \bar{x}_i e^T)^T (X_i - \bar{x}_i e^T)$ to obtain the first principal eigenvector θ_i . The first principal component is $z_i = (\theta_i^T \cdot X)^T$;

Step 3 Find the coordinates $z_i = z_i - c_i$ of the first principal component vector z_1 on each block, where c_i is the centre of each cluster, and the matrix $Z = [z_1, z_2, \dots, z_t]$

Step 4 Call manifold learning algorithm dimension reduction vector field Z .

The algorithm uses the k-means algorithm to solve the eigenvalues of the covariance matrix, whose size is $l \times t$ (l is the number of samples on each block).

In the neighbourhood selection, we use the k-means algorithm to divide the data set. We do not choose the k neighbourhood algorithm to construct the neighbourhood for each data, because if we would do so, a large amount of computation would be required for calculating the local principal component, and the computational complexity will increase significantly. Moreover, and if the number of samples is large, the calculation is extensive.

5.3.1.2 Experiment and analysis

We will experiment with the Swiss-roll data set to illustrate the effectiveness of the algorithm. The Swiss-roll data set is obtained by homogeneous sampling from a noiseless Swiss-roll surface in three-dimensional Euclidean space [10]. In the experiment,

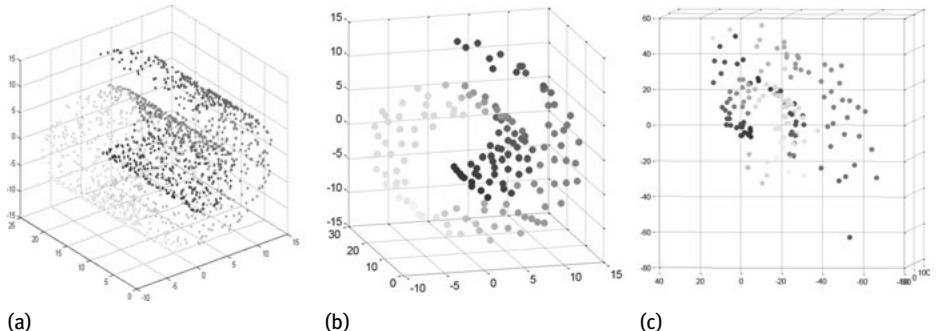


Fig. 5.2: Swiss-roll sampling surface and TBFR running process screenshots.

we take $N = 1600$, $d = 2$, and $t = 160$. The Swiss-roll data set is divided into 160 blocks, which has nothing to do with the sampling of the data; there is a certain randomness.

Figure 5.2 mainly shows the process of running TBFR on the Swiss-roll data set. The left graph is the Swiss-roll surface. The middle graph is the representation of the clustering centres in the three-dimensional space after clustering. The right figure is the position of the points in the matrix Z in the space after running Step 3. Noted that the left two figures use the same colour vector annotation data. The colours of the points in the right picture are determined by the average value of the colours of all the points in the block.

By directly using Isomap, LLE, and LTSA to reduce the dimension of the data set, we get three graphs in the first row of Figure 5.3; then, after the use of the vector field reduction algorithm with the above three algorithms to reduce the dimension, the result is the second row of Figure 5.3. Through the experiment, we find that when the manifolds Z in the vector space are composed in the main direction of each block, they can still reflect the geometric structure of the original manifold. After calling the reduction dimension algorithm, we find that although the two-dimensional projection of the manifold Z shows different effects of using the dimensionality reduction algorithm directly for the original manifold, the projection is still spread on the plane in an orderly manner.

In the experiment, we find two parameters that affect the experiment. The first is the number of clusters t . Owing to the use of the k-means algorithm, the clustering algorithm needs to pre-enter the number of clusters. In the experiment, we found that the number of clusters t affects the final results of the experiment. The second is the neighbourhood parameters of the dimension algorithm (Step 4), which is determined by the characteristics of the dimensionality reduction algorithm itself. In addition, because the k-means clustering results have a certain degree of randomness, we found that although the results of each experiment have common characteristics and laws, the results themselves are not the same.

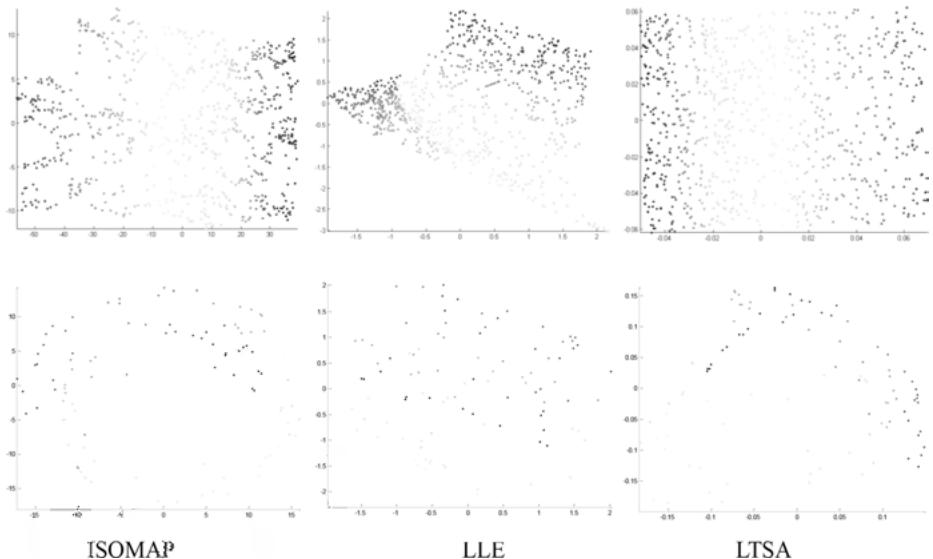


Fig. 5.3: TBFR experimental results on Swiss-roll.

5.3.2 Main link curve construction algorithm based on tangent contact

5.3.2.1 Introduction

Searching for low-dimensional manifolds embedded in high-dimensional space is a basic problem in machine learning, pattern recognition, and other fields. In 1904, Spearman proposed the linear principal component analysis method, which remains an important tool for data analysis. Principal component analysis limits the objective function to the maximum variance of the data set to a vector (straight) projection. The main curve can be regarded as a one-dimensional main form. It is the first principal component of the non-linear promotion, and its basic idea is to find a smooth curve through the middle of the data distribution. Hastie and Sutetze [11] proposed the main curve concept (HS main curve) in their ground breaking thesis on the main curve, hoping to find the ‘skeleton’ that truly reflects the ‘cloud’ (data distribution) and is statistically defined to satisfy the self-consistent characteristics. However, the goal of the main curve is not only to search for data sets of the backbone but also to obtain its skeleton. The spine of the data set can be solved by a linear method, but the skeleton requires a non-linear method. Tharpey and Flury [12] extended the definition of self-consistent random variables from the statistical point of view, set self-consistency as a theoretical basis for constructing statistical methods, such as principal components and main curves, and analysed the relationship between self-consistency and maths. The study of the main curve theory is based on Hastie’s research on the interdisciplinary distribution of data to enrich the results [13–15]. Duchamp et al. [16, 17] and Martínez-Morales [18] examined the overall differential geometric properties of the plane principal curve, noting that the principal curve on the plane is the saddle point

of the expected square distance of the data when the curvature is not zero, and there is no local minimum, which will not only lead to model fitting failure but also not allow the use of the cross-validation method to select the model complexity. It seems that the main curve through the distribution of data only has a qualitative meaning and lacks sufficient theoretical support. Thus, it is difficult to promote. In addition, the existence proof of the current HS main curve and other main curve definitions is only applicable to some special distribution. Kegl [19] defined a length-constrained main curve concept. Once the length of the main curve is fixed, the minimum square point-to-curve distance can be calculated. It is proven that under the premise that the data distribution has a finite second-order moment, the main curve has existence and uniqueness, but how to choose a pre-fixed main curve length is also a problem.

In the literature [20], according to the relationship between the local tangent space and the principal component, gives a new principal curve construction algorithm under the limit sense, proving that the main curve satisfies the autocorrelation characteristic and is unique to any given existing open cover. The following is the main curve construction algorithm [20].

Algorithm 5.3: Algorithm for building the main curve based on the tangent space

(In the next section, it is convenient to refer to this algorithm as the tangent space algorithm.)

- Step 1** Cover the sample data using k spherical neighbourhoods $\{Q_k(\lambda_1, \delta_1), Q_k(\lambda_2, \delta_2), \dots, Q_k(\lambda_k, \delta_k)\}$;
- Step 2** Calculate $E_{ki}(x)$, $\text{Cov}_{ki}(x)$ and its eigenvector $\xi_{kij} = (\xi_{i1}, \xi_{i2}, \dots, \xi_{ij})$ of x_{i1}, x_{i2}, \dots for each neighbourhood $Q_k(\lambda_i, \delta_i)$, $i = 1, 2, \dots$;
- Step 3** $\xi_{kl} = \min \{\xi_{(k-1)l}^T, \xi_{im}\}$ ($m = 1, 2, \dots, j$), where $\xi_{(k-1)i}$ is the principal eigenvector of the covariance matrix of data points in $Q_{(k-1)}(\lambda_l, \delta_l)$, and $Q_k(\lambda_l, \delta_l)$ is included in $Q_k(\lambda_i, \delta_i)$. We calculate the projection of the data point x_{ki} on ξ_{ki} in $Q_k(\lambda_i, \delta_i)$, $i = 1, 2, \dots, k$, and local reconstruction vector x_{ik} ;
- Step 4** Connect x_{ik} , and apply the spline function or local smoothing method to approximate the smooth main curve;
- Step 5** Calculate the global reconstruction error GMSE_k ; if $\text{GMSE}_k - \text{GMSE}_{k-1}$ is less than a threshold, then stop iteration. Otherwise, let $k = k + 1$, and go to Step 1

The algorithm divides the data set into open coverage of the spherical neighbourhood, constructs the tangent space in the neighbourhood to fit the main curve, and constructs the main curve in the limit sense by constantly subdividing the data set. In this section, we develop a new idea and method for constructing the main curve by constructing the tangent data set and its contact.

5.3.2.2 HS main curve definition and self-consistent characteristics

Problem Description: Consider the D -dimensional random vector $X = (X_1, X_2, \dots, X_D)$ with continuous probability density function $p_X(x)$, x_1, x_2, \dots, x_n for n random samples from X ; the problem is how to find a one-dimensional smooth curve $f(\lambda)$ in the middle of the distribution.

Hastie described the main curve on each point is projected to the point of the data points of the expectations; it is defined as a self-consistent curve. The self-consistent statistical definition is as follows:

Definition 5.10. (self-consistent) X and Y are random vectors. If each point on the support of Y is the condition of X projected to that point, i.e., $E(X|Y) = Y$, then X is a self-random variable.

Definition 5.11. For a given data distribution or data set, the self-consistent smooth curve is called the main curve.

5.3.2.3 Main curve construction algorithm based on tangent congestion

If the random vector X has a continuous probability density function $p_X(x)$ and a finite second order moment, $f(\lambda)$ is a curve with λ as a parameter on \mathbb{R}^D , where λ is a compact subset on R . Ball neighbourhood clusters $\{Q(\lambda_1, \delta_1), Q(\lambda_2, \delta_2), \dots, Q(\lambda_k, \delta_k)\}$ form the open cover of λ , where $\lambda_i \in \lambda$, $i = 1, 2, \dots, k$, ξ_i is the local tangent vector of $f(\lambda)$ on $Q(\lambda_i, \delta_i)$, written as $P\xi_i(x)$, and $x \in Q(\lambda_i, \delta_i)$ is the main projection vector of ξ_i . If

$$\lim_{\delta \rightarrow 0} \sum_{i=1}^k P\xi_i(x) \delta_i$$

exists, then $f(\lambda) = \lim_{\delta \rightarrow 0} \sum_{i=1}^k P\xi_i(x) \delta_i$ is the main curve and satisfies the autocorrelation characteristic, and $\delta = \max_{i=1, 2, \dots, k} \{\delta_i\}$.

Let D be a contact on vectors $\pi: E \rightarrow M$, $p \in M$. Then, for any $\xi \in \Gamma(E)$, $v \in T_p(M)$, $D_v \xi \in \pi^{-1}(p)$ is defined. $D_v \xi$ is called the covariance derivative of the smooth cross section ξ at the point p along the tangential dimension X . In addition, for the smooth curve $\gamma: [0, b] \rightarrow M$ in M , if $\xi \in \Gamma(E)$ satisfies

$$D_{\gamma'(t)} \xi = 0, \quad \forall t \in [0, b]$$

Then, we say that the smooth section ξ along the curve γ is parallel.

This algorithm from the literature [20] is based on the idea that through the open neighbourhood coverage, the limit is gradually approaching and seeking the main curve. In addition, the algorithm is used to calculate the contact on the open neighbour, using the minimum of the contact to get the result.

Algorithm 5.4: Main curve construction algorithm based on the tangent contact

(In the next section, it is convenient to refer to this algorithm as the tangent contact algorithm.)

Initially, the input is the entire data set $Q_0(\lambda_1, \delta_1)$, $k = 1$

Step 1 If the number of samples in the input data set is less than a threshold C , the program ends;

Step 2 Use two spherical neighbourhoods $\{Q_k(\lambda_1, \delta_1), Q_k(\lambda_2, \delta_2)\}$ to cover the sample data; k is for the k -th recursion;

Step 3 Analyse the principal component of the data points $x_{i1}, x_{i2}, \dots, x_{in}$ of each neighbour area $Q_k(\lambda_i, \delta_i)$, $i = 1, 2$, and get the main feature vector $\xi_{kij} = (\xi_{i1}, \xi_{i2}, \dots, \xi_{ij})$;

Step 4 $\xi_{kl} = \min \{\xi_{(k-1)l}^T, \xi_{im}\}$ ($m = 1, 2, \dots, j$), where $\xi_{(k-1)i}$ is the principal eigenvector of the covariance matrix of data points in $Q_{(k-1)}(\lambda_l, \delta_l)$, and $Q_k(\lambda_l, \delta_l)$ is included in $Q_k(\lambda_i, \delta_i)$. We calculate the projection of the data point x_{ki} on ξ_{ki} in $Q_k(\lambda_i, \delta_i)$ ($i = 1, 2, \dots, k$), and the local reconstruction vector $\hat{x}_{ik}, \{\hat{x}_{ik}\}$ ($i = 1, 2$) is a manifold local section;

Step 5 Calculate the contact D of $\{\hat{x}_{ik}\}$;

If D is less than a certain threshold β ,

Connect \hat{x}_{ik} , and apply a spline function or a local smoothing method to the nearly smooth main curve.

The program ends here;

Else,

For the first spherical neighbourhood, call the algorithm to enter the $k + 1$ layer recursion;

For the second spherical neighbourhood, call the algorithm to enter the $k + 1$ layer recursion;

5.3.2.4 Algorithm analysis and discussion

The tangent contact algorithm has similarity with the vector reduction algorithm based on the local principal direction of Algorithm 5.2, and the eigenvectors of the local sample covariance matrix are approximated to form a local tangent space. PCA is performed in each local region. We can get the approximate information of the tangent space, and the local topological structure is preserved in the tangent space.

The algorithm and the tangent space algorithm both use the cosine neighbourhood to cover the continuous data to approximate the main curve, and the method of calculating the eigenvector and the local reconstruction vector in the neighbourhood of the sphere are consistent.

The main differences are as follows:

- (1) The approximation is different. The tangent space algorithm splits the data set by increasing the number of open coverage segments in detail. When k is sufficiently large, the localised space is approximated by the limit condition. The tangent cut contact algorithm is always divided into two neighbourhoods, and the calculated neighbourhood recursively closes the tangent space.
- (2) The convergence conditions are different. The tangent space algorithm determines whether to stop the iteration by checking the global error. The tangent cut contact algorithm calculates the correspondence of the reconstructed vectors on the current two sphere neighbourhoods. When the contact is less than the threshold β , the iteration is stopped.
- (3) The program runs differently. The tangent space algorithm loops through Step 1 to Step 5 each time to increase the coverage of a neighbourhood, through the continuous increase in the neighbourhood coverage approximation; the tangent cut contact algorithm is recursively divided into small neighbourhood approximation.
- (4) Parameter control is different. The tangent space algorithm uses the parameter k to set the number of cycles to control the approximation granularity. The tangent cut contact algorithm controls the size of the data by setting the input data size.

Since Algorithm 5.3 runs recursively, if the data set is large, it needs deep recursion when approaching the small neighbourhood, which consumes considerable memory space. To some extent, this feature is unacceptable; hence, in the specific algorithm operation, we rewrite Algorithm 5.3 into a non-recursive form of operation.

The self-consistent characteristic is the most important characteristic of the main curve. The following proves that the main curve constructed by the chopping algorithm is consistent with the autocorrelation property.

Proof. According to the definition of self-consistency, we only need to prove that each point on the Y support is the conditional expectation of Y that is projected to that point, i.e., $E(X|Y) = Y$. Consider any given neighbourhood $Q(\lambda_i, \delta_i)$ and let P be the orthogonal projection matrix of \mathbb{R}^D to d -dimensional subspace M . Then, $Y_i = (I - P)E(X_i) + PX$. It is only necessary to prove that the base of M is the d -dimensional eigenvector of $\text{Cov}(X_i)$. Since the main feature vector ξ and the first principal component of $\text{Cov}(X_i)$ are chosen as the base of M in Step 3 of the algorithm, it is possible to obtain $Y_i = (I - P)E(X_i) + PX$ as X_i is self-consistent, $P = \xi$.

Since the spherical neighbourhood recursively covers the data set, any one point $f(\lambda^*)$ belongs to $f(\lambda)$, and there is always a point $\lambda' \in \lambda$, $f(\lambda^*) \in Y_i$ (where Y_i belongs to $Q_k(\lambda_i, \delta_i)$), satisfying $E(X|\lambda_f(X) = \lambda^*) = f(\lambda^*)$, where $f(\lambda)$ satisfies the autocorrelation property. Thus, the proof is complete. \square

5.3.2.5 Simulation results

1. Algorithm results when the data set is relatively dense or normative

Experiment 1: The sample data is sampled in a continuous function with no noise. Consider the curve $y = (1/2)\sin((1/2)x)$, $x \in [0, 20]$, and uniformly sample 100 points. In the parameter setting, C is set to 10 in the algorithm, and β is equal to 1. The experimental results are shown in Figure 5.4, where (1) shows the data before the algorithm is used, and (2) shows the main curve learned by the tangent cut contact algorithm. We can see that for the data having the set of regular data points, this algorithm has good learning results.

Experiment 2: The sample data is sampled in a continuous function, but the sampling rate is low and independently distributed Gaussian white noise is added to produce a relatively sparse but distributed data set, as shown in the left column of Figure 5.5. The algorithm parameter setting is $C = 12$ and $\beta = 1.5$. The results are shown in Figure 5.5 in the right column; for the sparse data sets, after setting the appropriate adjustment parameters, more satisfactory results are obtained.

2. The algorithm results when the data set is relatively sparse and irregular

Experiment 3: The sample data is sampled in a continuous function, but the sampling rate is very low and more independently distributed Gaussian white noise is added to

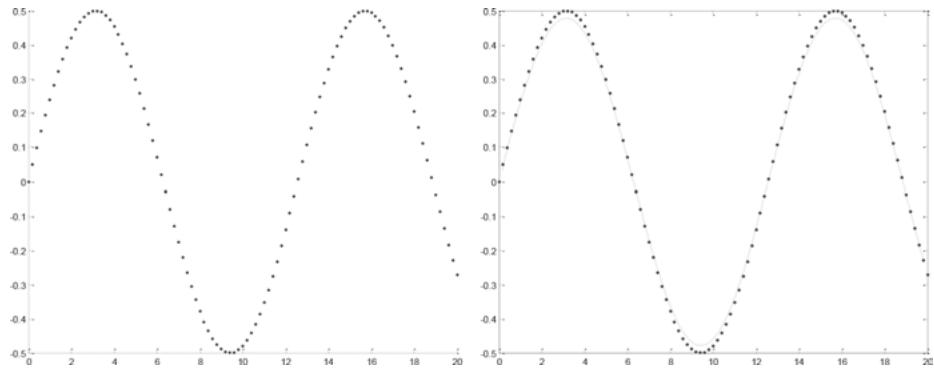


Fig. 5.4: Experiment 1: Conducting on sparse regular sampled data of a sine function.

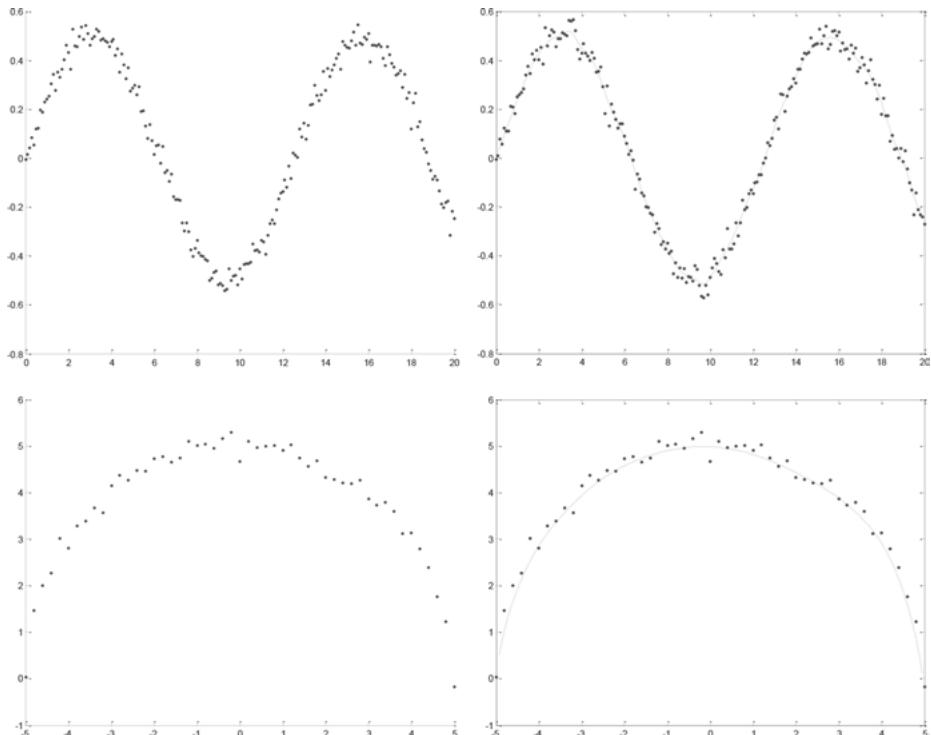


Fig. 5.5: Experiment 2: Data added with Gaussian noise after continuous function sampling.

produce a relatively sparse and irregularly distributed data set. In this case, we repeatedly adjusted the parameter settings to achieve the best results. Figure 5.6 shows the distribution of the generated data set and the results of the algorithm after iteration. The parameters in the graph are set to $C = 15$ and $\beta = 2.5$.

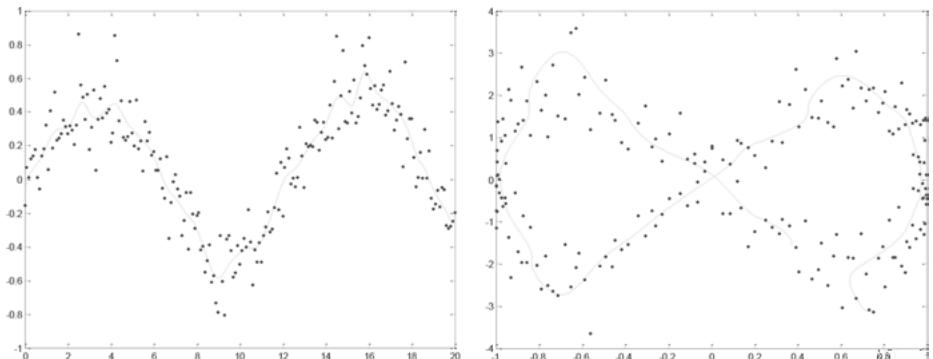


Fig. 5.6: Experiment 3: The Gaussian noise is added after the continuous function sampling. The Gaussian noise is more in this experiment, and the data is sparse and irregular.

3. The effect of parameter setting on the generation of the main curve

There are two key parameters in the algorithm: the input threshold C and the contact threshold β . Both parameters are related to convergence. The threshold value is related to the main curve generation accuracy. In general, we want to stop the program by the contact threshold. The input threshold C is the accuracy of the neighbourhood approximation. If the contact threshold is not always accurate, the program of the input threshold judgment ends. In the experiment, the contact threshold is related to the final form of the main curve, and the input threshold is strongly related with the data set. For sparse data sets, C needs to be appropriately larger, but if it is too large, the program will end prematurely; thus, to some extent, the value of C is empirical. It should also be noted that in the experiment, we found that the contact threshold and the input threshold are interrelated. When the threshold value is too small, the judgment is often based on the input threshold, which means that when the local precision is reached, the neighbourhood stops at the limit.

5.4 Summary

In this chapter, the fibre bundle theory was introduced into machine learning, and the fibre bundle learning model was established. The fibre bundle representation and basic concepts in machine learning were given, and the model of the bundle and the main fibre bundle were presented. Moreover, the vector reduction algorithm based on the local principal direction of the cut and the main curve building algorithm based on the piles were described in detail.

Bibliography

- [1] Nir SC, Ron K, Ravikanth M. A General Framework for Low Level Vision. *IEEE Transaction on Image Processing*, 1998, 7(3), 310–318.
- [2] Chen S, Sochen NA, Kimmel R. Stereographic Combing a Porcupine or Studies on Direction Diffusion in Image Processing. *Siam Journal of Applied Mathematics*, 2004, 64(5), 1477–1508.
- [3] Pearson DW. Approximating vertical vector fields for feedforward neural networks. *Applied Mathematics Letters*, 1996, 9(2), 61–64.
- [4] Chao J, Kim J. A Fibre Bundle Model of Surfaces and its Generalization. *International Conference on Pattern Recognition*. IEEE Computer Society, 2004, 560–563.
- [5] Brockett RW. Control Theory and Analytical Mechanics, Geometrical Control Theory. Ed, by C, Martin and R, Hermann, Math. Sci. Press, 1977.
- [6] Brockett RW. Nonlinear control theory and differential geometry, Warszawa. Proceedings of the International Congress of Mathematicians, August 16–24, 1983.
- [7] Mu XW. The Geometrical Framework of General Nonlinear Control System on Fiber Plexus and its Minimum Realization Theory. *Applied Mathematics and Mechanics*, 1996, 17(10), 939–950.
- [8] Zhang J. Research of the fiber bundle learning algorithm based on popular learning. Soochow: Thesis. Soochow University, 2008.
- [9] Zhang J, Li FZ: Study on the fiber cluster learning model based on manifold learning. *Journal of Nanjing University*, 2008, 44(5), 477–485.
- [10] Min WL, Lu L, He XF. Locality pursuit embedding. *Pattern Recognition*, 2004, 37(4), 781–788.
- [11] Hastie T, Stuetzle W. Principia Curves. *Journal of the American Statistical Association*, 1989, 84; 502–615.
- [12] Tarpey T, Flury B. Self–Consistency: A Fundamental Concept in Statistics. *Statistical Science*, 1996, 11(3), 229–243.
- [13] Tibshirani R. Principal Curves Revisited. *Statistics and Computation*, 1992, 2, 183–190.
- [14] Delicado P. Another Look at Principal Curves and Surfaces. *Journal of Multivariate Analysis*, 2001, 77, 84–116.
- [15] Chang KY, Ghosh J. A unified Model for Probabilistic Principal Surfaces. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 2001, 23(1), 22–41.
- [16] Duchamp T, Stuetzle W. Geometric Properties of Principal Curves in the Plane. *Robust Statistics, Data Analysis and Computer Intensive Methods*. New York: Springer Press, 1996, 109, 135–152.
- [17] Duchamp T, Stuetzle W. Extremal Properties of Principal Curves in the Plane. *Annals of Statistics*, 1996, 24(4), 1511–1520.
- [18] Martínez-Morales JL. Extremal Properties of Principal Embeddings. *Journal De Mathématiques Pures et Appliquées*, 1999, 78(9), 913–923.
- [19] Krzyzak A, Zeger K. Learning and Design of Principal Curves. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2000, 22(3), 281–297.
- [20] Zhao LW, Luo SW, Liao LZ, Tian M. Research on Main Curve Construction Algorithm. *Journal of Beijing Jiaotong University*, 2005, 30(2), 80–83.

6 Lie group covering learning

This chapter deals with the related theory of the Lie group machine learning covering algorithm, the simply connected covering algorithm of Lie group machine learning, the multiply connected covering algorithm of Lie group machine learning, and an application of the covering algorithm in molecular docking.

6.1 Theory of Lie group machine learning covering algorithm

This section deals with the theory of linear representation of a group, the basic properties of the Lie group, and the covering group theory of the Lie group. This knowledge serves as the theoretical basis for studying the Lie group machine learning covering algorithm.

6.1.1 Linear representation of a group

From the group theory point of view, two isomorphic groups have the same properties. Because the matrix group is easier to study, if we can find a matrix group and a given group isomorphism, then we can study the nature of the matrix group and fully grasp the nature of a given group. If the matrix group is only homogeneous with a given group, it not only reflects the partial properties of the given group but also plays a role in studying the properties of the given group. A matrix group with a given group isomorphism or homomorphism is called a linear representation of a given group.

Definition 6.1. If a group $D(G)$ consists of a set of $m \times m$ matrices having a determinant that is not zero and G is a given isomorphism or homomorphism, then $D(G)$ is called an m -dimensional linear representation of G . In $D(G)$, the matrix $D(R)$ corresponding to the element R in G is called the representation matrix of the element R in the expression $D(G)$, and the matrix trace $\chi(R) = \text{Tr } D(R)$ of $D(R)$ is called the character R in the representation $D(G)$.

The determinant of the matrix is not zero in order to exclude the mediator of the representation matrix and the zero matrix. Under this rule, the representation matrix of the constant element is the unit matrix, $D(E) = 1$, and the representation matrix of the reciprocal elements is an inverse matrix, $D(R^{-1}) = D(R)^{-1}$. If $D(G)$ is isomorphic to group G , then $D(G)$ is called the true representation of group G , and if it is homomorphic, it is not the true representation of group G . When all the elements in the group correspond to 1, $D(R) = 1$; this representation is called the identity representation, also known as mediocre representation.

Any group has a constant expression. The matrix group is a representation of itself. The representation matrix is the representation of the unitary matrix. Further,

the representation matrix is the representation of the real orthogonal matrix, called the real orthogonal representation.

Before we can find all the linear representations of a group, we should simplify the problem.

Definition 6.2. If the two representations $D(G)$ of all the elements R of the group G and the representation matrix in the group G have the same transformation relation $\bar{D}(R) = X^{-1}D(R)X$, then the two representations are called equivalent, i.e., $D(G) \sim \bar{D}(G)$.

The two equivalents represent equal dimensions, and the similarity transformation matrix X also has the same dimension. The two equivalent representations have equivalent group elements. Equivalence means that there is no substantial difference in the search for all the problems expressed in the group G to find all the unequal representations of the group G .

How do we judge whether the two are equivalent? The identity of any element in two equivalents is equal: $\chi(R) = \text{Tr } D(R) = \text{Tr } \bar{D}(R) = \chi(\bar{R})$.

For a finite group, the eigenvalues of each element in the two representations are equal and are two necessary equivalents. As the signature is a function of the class, in order to test the equivalence of the two representations, we only need to select an element in each class to check whether the signatures in the two representations are equal.

Definition 6.3. If the group G indicates that each representation matrix $D(R)$ of $D(G)$ can be transformed into the same form of the step matrix by a similar transformation X

$$X^{-1}D(R)X = \begin{pmatrix} D^{(1)}(R) & M(R) \\ 0 & D^{(2)}(R) \end{pmatrix}$$

then this representation is called an irreducible representation; otherwise, it is called reducible. The set of two sub-matrices $D^{(1)}(R)$ and $D^{(2)}(R)$ in the above equation constitutes a linear representation of group G . The signature of the element in the reducible representation is equal to the sum of the signatures in the child representation:

$$\chi(R) = \chi^{(1)}(R) + \chi^{(2)}(R).$$

The nature of $D(G)$ is expressed by the properties of two sub-representations. On the other hand, the number of irreducibles represents straight and together; it constitutes a fully reducible expression. Such an exponential representation does not give any new properties; its representation space is a number of irreducible representations of the sum of spaces. The vectors in the space can be uniquely decomposed into the sum of the vectors belonging to each subspace and are transformed according to each irreducible representation. Thus, the problem of finding all unequal representations of the group is further reduced to the problem of finding all unequal irreducible representations of the group.

Hence, we can say that the basic task of group theory is how to determine the equivalence and irreducibility of representation, to find all unequal irreducible representations of a given group, and how to denote the irreducible expression as a distinct sum.

Schur's theorem is the most basic theorem in group representation theory, which applies to all groups and explains the basic characteristics of the unequal irreducible representation of groups.

Theorem 6.1. Let $D^{(1)}(R)$ and $D^{(2)}(R)$ be the two unequal irreducible representations of group G , with dimensions m_1 and m_2 , respectively, and let X be an $m_1 \times m_2$ matrix. If every element R satisfies:

$$D^{(1)}(R)X = XD^{(2)}(R),$$

$$\sum_{\rho} D_{vp}^{(1)}(R)X_{\rho\mu} = \sum_{\rho} X_{vp}D_{\rho\mu}^{(2)}(R)$$

then $X = 0$.

Inference 6.1. The matrix of all the representable matrices $D(R)$ of the irreducible representation $D(G)$ must be a constant matrix, i.e., if $D(R)X = XD(R)$, then $X = \lambda 1$, where λ is a constant.

The main task of group theory is to find all the inequality irreducible representations of various typical groups and study the reduction method of reducible representation. First, find the group of all non-equivalent irreducible characteristics of the standard, i.e., the characteristics of the standard table, called the signature table. Then, we determine the irreducible representation matrix. Since the group elements can be expressed as the product of the generator, it is possible to determine the matrix of the generator.

6.1.2 Basic properties of the Lie group

The Lie group is a continuous group, and each element R can be described by a set of independent real parameters (r_1, r_2, \dots) that change continuously within a certain area. A one-to-one correspondence is required between the group element and the parameter value in the region of change of the parameter. The number of independent real parameters is called the order of successive groups. The change area of parameters is called the group space.

Set element $R \in G$. The parameter is (r_1, r_2, \dots, r_g) , abbreviated as $R(r_1, r_2, \dots, r_g) = R(r)$. For the product of group elements, $R(r)S(s) = T(t)$, g parameters t_j are functions of $2g$ variables r_i and s_k : $t_j = f_j(r_1, \dots, r_g; s_1, \dots, s_g) = f_j(r; s)$.

The g functions $f_j(r; s)$ are called combinatorial functions of continuous groups, which fully describe the product rules of group elements. The combination functions

of the Lie group are analytic functions. As the function is continuously differentiable, calculus can be used to study the Lie group. Thus, the Lie group is the most extensively studied group.

6.1.2.1 Local properties of the Lie group

In the group space adjacent to the corresponding elements, called the adjacent elements, because the parameters of the constant element are often selected as zero (constant element adjacent elements), the parameters are infinitesimal, known as infinitesimal elements. Infinitesimal is a limit process. The nature of infinitesimal elements of the Lie group determines the local properties of the Lie group.

Consider an element $g(\alpha)$ of the Lie group, where α represents a set of m consecutive group parameters, and $\alpha = 0$ represents a unit element. If α is small, $g(\alpha)$ is very close to the unit, and it can be used in the vicinity of units for Taylor expansion:

$$g(\alpha) = g(0) + \alpha^i \frac{\partial g}{\partial \alpha^i} \Big|_{\alpha^i=0} + \dots$$

Here, we use the tensor symbol to omit the sum of i , and we introduce the symbol $X_i = \partial g / \partial \alpha^i|_{\alpha^i=0}$, which is called an infinite scent operator or generator of a group. It is the derivative of the group element in the vicinity of the unit, and its number is also m . Note that the generator X_i is independent of the group element, which reflects the nature of the group, i.e., the elements of the Lie group are expressed by generating generators and group parameters.

Thus, the difference between different populations is expressed as the nature and number of generators, and the number of generators, i.e., the number of group parameters, is called the dimension of the group. The exchange algebra between generators portrays the nature of the Lie group.

The generators of linear representations satisfy the common reciprocity relation:

$$I_j I_k - I_k I_j = i \sum_{\ell} C_{jk}^{\ell} I_{\ell}$$

$$C_{jk}^{\ell} = \left\{ \frac{\partial S_{\ell k}(r)}{r_j} - \frac{\partial S_{\ell j}(r)}{r_k} \right\} |_{r=0} .$$

The algebra is closely related to the concept of the Lie group, and many of the properties of the Lie group can be discussed directly through Lie algebra. According to this concept, the reciprocity relation of the Lie group generators constitutes a reflection of the nature of Lie algebra.

Consider an input space G . If the linear space g on G has a binary operation $(x, y) \rightarrow [x, y]$ (often referred to as transposition or inclusion), and it satisfies the following conditions:

(1) This binary operation is bilinear, i.e.,

$$[\lambda x + \mu y, z] = \lambda[x, z] + \mu[y, z], \quad \forall \lambda, \mu \in G, \quad \forall x, y, z \in g;$$

- (2) $[x, x] = 0, \forall x \in g$;
 (3) The Jacobi identity is established, i.e.,

$$[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0, \quad \forall x, y, z \in g$$

then we say that g is the Lie algebra on G .

In fact, when the nature of the Lie group is attributed to easy generation, such as an operation loop, i.e., the total effect after several operations, and returning to the unit element. This mainly reflects the local nature of the continuum. Therefore, the local identity of the Lie group is determined. If there are two or more groups, although the overall nature is different, the local nature and structure are the same; they correspond to the same Lie algebra, such as SU(2) group and SO(3) group. This relationship between the Lie group and its covering group is what we will discuss.

6.1.2.2 Overall properties of the Lie group

The overall nature of the Lie group study is to study the topological properties of the Lie group space. First, we discuss the connectivity of the group space. If any two elements in the group, which are in the group space as corresponding points, can be contained in a completely inclusive connection within the group space, then the group space is called connected. Such a continuous group is called a simply continuous group. The SO(3) group is a simple Lie group. Conversely, if the group space is divided into several pieces that are not linked, then this continuous group is called a mixed continuous group. For simple Lie groups, the elements of the groups can be expressed as infinitely many infinitesimal elements. In addition to the infinitesimal elements, it is necessary to give a special element (including the constant) in each successive slice of the group space, and their product can represent any element. For the linear representation, we can see that the representation matrix $D(R)$ of any element R of the simple Lie group can be calculated by the generator, but for the mixed Lie group, we must also know the representation matrix $D(\sigma)$ of the spatial inversion element.

In the group space of the mixed Lie group, the set of the corresponding elements of the continuous piece containing the constant element constitutes the invariant subgroup of the mixed Lie group, and the other successive elements correspond to the cosine of the invariant subgroup. Therefore, the nature of the mixed Lie group is determined by the nature of a simple Lie group (invariant subgroup) and a representative element in each successive piece (coset).

Next, we discuss the connectivity of the Lie group. In the group space of a simple Lie group, the point of the element R can be connected to many lines by the corresponding point of the constant element. Some connections can change continuously in the group space, whereas others cannot. In this way, these connections are divided into groups, and the same group of connections can be continuously changed in the group space, whereas the connection of different groups cannot. The number of

groups of these connections is called the connectivity of the group space. The nature of the Lie group is closely related to the connectivity of the group space.

6.1.2.3 Covering group of the Lie group

Definition 6.4. Let m, n be Hausdorff spaces. A continuous mapping $J: m \rightarrow n$ is called a covering mapping from m to n . If $\forall p \in n$, we take $p' \in f^{-1}(p)$; then, there exists a neighbourhood $V_{p'}$ of p' in m such that $f(V_{p'})$ is the neighbourhood of p in n , and $V_{p'}$ and $f(V_{p'})$ are homologous at f . Further, m is called the covering space of n and (m, f) is the covering of n .

Definition 6.5. Connect the homomorphism of the Lie group G_1 to G_2 , and if it is the coverage of G_2 , G_1 is called the coverage group of G_2 . For a unit element of G_2 , $f^{-1}(e) = \Gamma$ is called the Poincare group of the coverage group (G_1, f) with respect to G_2 , and the number of elements of the Poincare group is called the leaf coverage.

The Poincare group has the following properties:

Lemma 6.1. (G_1, f) is the coverage group of the connected Lie group G_2 . (G_1, f) for the Poincare group $\Gamma = f^{-1}(e)$ of G_2 is a discrete normal subgroup, so it is in the centre of G_1 . If we take any $g_2 \in G_2$, $g_1 \in f^{-1} \subset G_1$, then $f^{-1}(g_2) = g_1\Gamma = g_1f^{-1}(e) = \{g_1h \in G_1 | \forall h \in \Gamma\}$. The complete primitive of the midpoint of G_2 constitutes the left parcel of the topological group G_1 with respect to the normal subgroup Γ , and the topological group G_1/Γ is isomorphic to the topological group G_2 .

To maintain the coverage nature, we introduce the following:

Definition 6.6. The two covering groups (G_1, f_1) and (G_2, f_2) of Lie group G are called equivalents, and if there is an isomorphism on G_1 to G_2 , then $f_1 = f_2 \circ \varphi$, i.e., an exchange diagram:

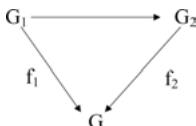


Fig. 6.1: Exchange diagram of G_1 and G_2 .

It is known from the definition that if the two groups (G_1, f_1) and (G_2, f_2) of Lie group G are equivalent under φ , then the unit element e of G is obviously $\varphi(f_1^{-1}(e)) = f_2^{-1}(e)$, which leads to the isomorphism of the Poincare group; hence, the coverage state of the two groups is exactly the same.

Definition 6.7. The Hausdorff topological space m is called single-connected, and if its closed path (i.e., the closed curve between any two points) is a homotopy to zero, then the first-order homotopy group of m is zero, i.e., any closed road can be continuously shrunk to a point.

Note that single communication and connectivity are irrelevant concepts, and single connectivity cannot be connected. Now, considering the connected manifold, because the manifold must be connected, the single-connected manifold is equivalent to any of two points p, q , and any two consecutive curves of p and q can be continuously shifted to each other.

Theorem 6.2. *If N is a connected Lie group, then there exists a single-connected Lie group M that is the coverage group of N .*

Proof. From the theoretical knowledge of the topological coverage space, where N is a connected manifold, a single-connected manifold M is the coverage space of N ; $P: M \rightarrow N$ is the covering mapping.

For a road $x: [0, 1] \rightarrow N$ with e_N as the starting point, $x(0) = e_N$, the road class $[x]$ to which x belongs is a point in M and covers the mapping $P[x] = x(1)$.

For two roads x, y in N with e_N as the starting point, we define z as $z: [0, 1] \rightarrow N$, $z(t) = x(t)y(t)$. We define multiplication of points in M as $[x][y] = [z]$ or $[x][y] = [xy]$. This definition is significant because when $[x^*] = [x]$ and $[y^*] = [y]$, H_1 is the fixed-point homotopy of the road x and x^* , and H_2 is fixed-point homotopy of the road y and y^* . We define $H(s, t) = H_1(s, t)H_2(s, t)$, where H is fixed-point homotopy of the road xy and x^*y^* , i.e., $[xy] = [x^*y^*]$. Further, $[e_N]$ (road class of constant road e_N in N) = e_M is the unit of M . In M , the inverse of point $[x]$ $[x]^{-1} = [x^{-1}]$, where $x^{-1}(t) = (x(t))^{-1}$.

After defining the multiplication, M is obviously a group. The covering mapping $P([x][y]) = P[xy] = x(1)y(1) = P[x]P[y]$ is a group of homomorphisms.

Since M is present in the open nuclei V^* of the nucleus $P(V^*)$ in N , M can be given a differential, and M becomes a Lie group. Further, M and N are local differential isomorphism. Thus, $\Lambda(M)$ is $\Lambda(N)$. \square

Theorem 6.3. *Let G be a connected Lie group. Then, there exists a unique connectivity in the isomorphic sense and a single-connected Lie group \tilde{G} , and the covering map f on \tilde{G} to G such that (\tilde{G}, f) is the covering group of G , which is called the universal covering group of (\tilde{G}, f) .*

Proof. Let (\tilde{G}, f) be the general covering manifold of manifold G , $\Gamma = f^{-1}(e)$. Let $e^* \in \Gamma$ and $a^*, b^* \in \tilde{G}$. Then, there are roads r_1^*, r_2^* connecting e^* and a^* , and e^* and b^* , respectively, in \tilde{G} . Thus, $r_1 = f(r_1^*)$, $r_2 = f(r_2^*)$ are the roads from e to $a = f(a^*)$ and from e to $b = f(b^*)$, respectively. Therefore, $r = r_1(ar_2)$ is the path from e to ab in G .

Raise r to the road whose starting point is e^* in B , and set its end as D . Define multiplication in A as follows: $a^*b^* = c^*$. It should be noted that, c^* does not depend on the choice of r_1^* and r_2^* . Let w_1^*, w_2^* be other roads connecting e^* and a^* , and e^* and b^* , respectively, in \tilde{G} . Then, w_1, w_2 are roads connecting e and a , and e and b , respectively, in G . Further, $w = w_1(\alpha w_2)$ is the path from e to ab in G . Let w^* be the raise of w , so that r_i^* can be continuously changed to w_i^* because \tilde{G} is single-connected. Hence, w_i can be continuously changed to r_i , and w can be continuously changed to r . Since

the starting and ending points of w and r are the same, the end of w^* and r^* is the same. Therefore, the \tilde{G} multiplication of the above definition is reasonable.

The combined law of multiplication in \tilde{G} can be obtained by the combination of road multiplication.

From the definition, we also know that A is a unit element. Further, $a^{-1}r_1^{-1}$ is the road connecting e and a^{-1} in G . It is raised to the road connecting e^* and $(a^*)^{-1}$ in \tilde{G} . Then, we know that $a^*(a^*)^{-1} = e^*$, i.e., $(a^*)^{-1}$ is the inverse element of a^* .

In summary, \tilde{G} is a group. We know that f is a group homomorphism from \tilde{G} to G because $f(a^*b^*) = ab = f(a^*)f(b^*)$.

Since f is the covering mapping, the inverse operation and multiplication of \tilde{G} are continuous, and the group structure of \tilde{G} is a local group of \tilde{G} , we can prove that \tilde{G} is a topological group. In addition, \tilde{G} is not only a connected topological group but also a local Lie group; hence, there is a unique analytic structure such that \tilde{G} is a Lie group.

Thus, the proof is complete. \square

If (G_1, f_1) is also the general coverage group of G , then the homeomorphic mapping f' of \tilde{G} to G_1 makes $f = f_1f'$. Thus, $f_1f'(e^*) = f(e^*) = e$. Obviously, $L_{a_1^{-1}}f'$ still is a homeomorphism, and $L_{a_1^{-1}}f'(e^*) = e_1$. $L_{a_1^{-1}}f'$ is still referred to as f' . Then,

$$\begin{aligned} f_1f'(a^*b^*) &= f(a^*b^*) = f(a^*)f(b^*) \\ &= f_1(f'(a^*))f_1(f'(b^*)) = f_1(f'(a^*)f'(b^*)) \end{aligned}$$

Because f_1 is local isomorphism, hence, in a unit neighbourhood

$$f'(a^*b^*) = f'(a^*)f'(b^*) .$$

Further, \tilde{G} and G_1 are connected, and f' is a one-to-one correspondence; hence f' is the Lie group of the isomorphism.

Inference 6.2. Let (\tilde{G}, f) be the general covering group of connected Lie group G . Then, df is the isomorphic mapping of $\text{Lie}\tilde{G}$ to $\text{Lie}G$.

This conclusion can be obtained from the fact that f is the homomorphism of \tilde{G} to G and is locally homogeneous.

Theorem 6.4. The connected group G must be connected by a single covering group (\tilde{G}, f) , called the general coverage group of G . Further, f is an isomorphism if and only if G itself is a single-connected Lie group.

Here, we can give two examples of covering groups:

$$f: R^1 \rightarrow S^1, \quad f(t) = e^{a\pi it}, \quad t \in R^1; \quad \text{thus, } R^1 \text{ is the covering group of } S^1$$

Ad: $S^3 \rightarrow \text{SO}(3)$. For $\text{SO}(3)$, the appropriate unit neighbourhood V , $\text{Ad}^{-1}(V)$ has two connected branches, i.e., it is a ‘two-layer’ coverage, while the upper case is an ‘infinite multi-layer’ coverage.

We can see from the above definition and the nature of the connected group, that since Lie algebra is the local structural invariant of the Lie group, $df: g_1 \rightarrow g_2$ is the isomorphism of the Lie algebra when $f: G_1 \rightarrow G_2$ is the coherent state of the Lie group. On the other hand, when the Lie algebras g_1 and g_2 of G_1 and G_2 are isomorphic, then there must be a pair of homomorphisms such as Lie group \tilde{G} and $f_i: \tilde{G} \rightarrow G_i$ ($i = 1, 2$).

It can be seen that the Lie group theory of covering group plays an important role in Lie group research.

6.2 Simply connected covering algorithm of the Lie group

First, the research status of the algorithm based on the covering idea is introduced. Then, the single-connected overlay algorithm of Lie group machine learning is given, including the identification of coverage group, universal covering group solving algorithm, and single-connected covering algorithm. Finally, an example is introduced.

6.2.1 Research status of algorithm based on covering idea

The technique of covering is an important tool for studying algebraic geometry, such as coverage in research-based transformations, model maps, and singularities and so on. The idea of covering has been applied to many areas, such as data mining and machine learning.

- 1) Professor Zhang et al. [28] gave the geometric significance of M-P neurons and transformed the optimal design problem of neural network into some optimal coverage problem by spherical projection transformation. They corresponded neurons to the spherical neighbourhood of geometric samples. With this geometric intuitionistic approach, they designed the multi-layer forward neural network as a ‘classifier’ and used this design as a ‘spherical neighbourhood’, with the input x^i ($i = 1, 2, 3, \dots, d$) divided by the category to which they belong. One of the simplest design methods is to cover the x^i ($i = 1, 2, 3, \dots, d$) of a class with a set of spherical neighbourhoods without covering the x^i ($i = 1, 2, 3, \dots, d$) that do not belong to that class. Thus, different types of inputs are covered by spherical neighbourhoods of different groups. Then, the output of neurons corresponding to the spherical neighbourhood of the same group is put together using an ‘or gate’. Thus, a classifier is designed. The spherical neighbourhood of a covering corresponds to a hidden layer. The minimum coverage corresponds to the minimum hidden neural network structure. They also showed that any method of sub-optimal coverage and the spherical neighbourhood method can give a learning algorithm of neural networks, which is called neighbourhood covering algorithm.

- 2) The theoretical basis of the bionic pattern recognition (topological pattern recognition) proposed by the academician Wang Shoujue [30] is that it confirms the continuity of similar samples in the feature space (which cannot be split into two parts that are not adjacent to each other). After introducing the continuity rule of the same kind type samples in the bionic pattern recognition, the ‘cognition’ of a class of things is essentially the analysis and ‘understanding’ of ‘shape’ for the infinite set of all the things that are formed in the feature space. Bionic pattern recognition is based on the best coverage of the distribution of a class of samples in the feature space.
- 3) Zhu Yonghong [33] proposed a covering clustering algorithm that uses the concept of coverage to aggregate more concentrated samples together to find classes that are implicit in the sample set. For the sparse samples, he used the shortest distance method to get the clustering effect.
- 4) He Qing and Shi Zhongzhi [31] proposed a basic algorithm based on the hypersurface classification and discrimination method, i.e., through the regional consolidation calculation, to obtain a number of ultra-planar compositions of the closed surface. Let a classification of hypersurfaces divide the space, i.e., a closed area is formed around the sample point. The region is surrounded by multiple classified hyperplanes that make the region cover a class of as many sample points as possible without covering heterogeneous sample points.
- 5) Wu Tao and Zhang Ling et al. [29] proposed a new kernel covering algorithm by combining the kernel function method in SVM with the covering algorithm of constructive learning. By overlaying the coverage domain by covering the field of coverage in the sample space, the coverage area belonging to the same type is merged into a sphere of the feature space, and the classification error is minimised. Thus, in the feature space, we find the optimal envelope of each category, so that similar areas of similar integration are together. The kernel overlay algorithm combines the advantages of both the coverage algorithm and the SVM, and also has the advantages of fast operation speed, high precision, and robustness.
- 6) Zhang Junping [37] proposed a manifold learning method, whereby a nested manifold structure can be described by a limited number of data points. The set of finite data points can cover the support fields of the nested manifold so that the structural information of the nested manifolds is not lost while keeping the required code tables to be as few as possible. To describe the manifold structure from the finite sample set, the paper refers to the idea of neighbourhood coverage. Define (χ, ρ) as the pseudo-measure space, let $B_r(x)$ be the closed ball of radius r , let S be the subset of χ , and let ε be a positive constant. Then, the number of neighbourhoods describing the manifold structure can be defined as the minimum number of point sets $X \subset \chi$ such that $S \subseteq \bigcup_{x_i \in X} B_\varepsilon(x_i)$. This ensures that the maximum difference between the elements in any data support set and the nearest element in X

is less than or equal to ε after using neighbourhood coverage. The local neighbourhood coverage set is used so that the overall structure of the information will not be lost. Then, the corresponding two-dimensional mapping of the local neighbourhood covering set is obtained. Finally, the whole data set is mapped to two-dimensional space and reconstructed by combining the idea of kernel and coordinate map.

It can be seen that in the field of machine learning, the covering idea has been widely used. This book is mainly based on the perspective of the group. We can use the Lie group special algebraic geometry to express the sample data effectively. The feature space of the sample is embedded in a differential manifold to form a Lie group.

Through the analysis of the nature of the Lie group, the nature of the Lie group is closely related to the connectivity of its group space. When the group space of the Lie group is multiply connected, the real representation of the group is multivalued. In the process of learning, it is often difficult to analyse the problem of multi-valued representation directly. Therefore, we can solve the problem of the Lie group and use the covariance relationship between the Lie group and the cover group to solve this problem. The main idea of covering is to cover the same type of problems, so that different types of problems are separated. The advantage of this approach is obvious: through the same type of sample learning, it can acquire a better learning ability. When we can fit many types of problems, we can use the Lie group special algebraic geometry to maintain the invariance of the problem space to the greatest extent.

First, we consider the covering algorithm of Lie group machine learning in the case of single communication.

6.2.2 Simply connected covering of Lie group machine learning

Theorem 6.5. *The Hausdorff topological space m is called single-connected if m cannot be expressed as a union of two non-intersecting nonempty open sets. In other words, if any of its closed paths (i.e., closed continuous curves) are a homotopy to zero, i.e., the first-order homotopic group of m is zero, then any closed road can be continuously shrunk to a point.*

Note that single communication and connectivity are irrelevant concepts, and single communication cannot be connected. Now, consider connected manifolds. Since connected manifolds are arc connected, the single connection is equivalent to any two continuous curves of any two points P, Q , over P and Q in M , which can be moved continuously.

Theorem 6.6. *Let G and H be Lie groups with g and h as Lie algebras, where G is single-connected. Let $\psi: g \rightarrow h$ be a homomorphism. Then, there is a unique homomorphism $\varphi: G \rightarrow H$ that makes $d\varphi = \psi$.*

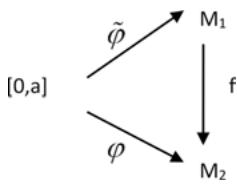


Fig. 6.2: Exchange diagram of M_1 and M_2 .

Proof. The uniqueness is easy to prove. The following proof is about existence.

Let $\{w_i\}$ be the base of the left invariant form 1, so that ψ^* is the transpose of ψ . The form 1 on $G \times H$ is $\{\delta\pi_1(\psi^*(w_i)) - \delta\pi_2(w_i)\}$, where π_1 and π_2 are the standard projections on $G \times H$ to G and H , respectively. This form is left unchanged, and the generated ideal ℓ is a differential ideal. Thus, the maximal connected integral manifold I of ℓ through $(e, e) \in G \times H$ is a plum cluster of $G \times H$, and its dimension is equal to the dimension of G . In other words, it is a covering homomorphism.

It is assumed that G is simply connected, so $(\pi_1|_I): I \rightarrow G$ is an isomorphism. Define $\varphi: G \rightarrow H$ as $\varphi = \pi_2 \circ (\pi_1|_I)^{-1}$. Then, φ is a Lie group homomorphism, and $d\varphi(w_i) = \psi^*(w_i)$. Thus, $d\varphi = \psi$ and the theorem is proved. \square

Inference 6.3. If a single-connected Lie group G and H have isomorphic Lie algebras, then G and H are isomorphic.

Ado theorem holds that for each Lie algebra, for a certain n , there is a faithful representation in $GL(n, R)$. According to this theorem, if g is a Lie algebra, then there is a Lie group, especially a single-connected Lie group, with g for the algebra. In view of this result, we have the following:

Theorem 6.7. There is a one-to-one correspondence between the isomorphism of the algebra and the isomorphism of the single-connected Lie group.

Proof. If there are two curves $\varphi_i: [0, a] \rightarrow M$ ($i = 1, 2$), $\varphi_1(0) = \varphi_2(0)$, and $\varphi_1(a) = \varphi_2(a)$, having common start points and end points in M , then M is simply connected. Let $\tilde{f}: \tilde{M} \rightarrow M$ be a differentiable covering. The necessary and sufficient condition for being a universal coverage is that \tilde{M} is simply connected. Furthermore, the necessary and sufficient condition for a connected manifold M to be simply connected is that there is no nontrivial cover in it. In other words, if $f_1: M_1 \rightarrow M$ is an override mapping in M , then f must be a homeomorphism.

Let $f: M_1 \rightarrow M_2$ be a differentiable covering, $\varphi: [0, a] \rightarrow M_2$ be a continuous parameter curve in M_2 , and p be an arbitrarily selected point in $f^{-1}(\varphi(0))$. Then, there is a unique continuous mapping $\tilde{\varphi}: [0, a] \rightarrow M_1$, which makes $\tilde{\varphi}(0) = p$, $f \circ \tilde{\varphi} = \varphi$, as shown in the following illustration, where $\tilde{\varphi}$ is called a coverage curve for φ . \square

Definition 6.8. Let $p: M \rightarrow N$ be a covering map and $f: I \rightarrow N$ be a road on N . Then, f can be uniquely raised to road $g: I \rightarrow M$ in M , which makes $p(g) = f$.

Algorithm 6.1: Judge the covering group

```
// Judge whether Lie group M is the covering group of the connected group
// Input: Lie group M, N
// Output: {0, 1}
Procedure IsCovergroup(M,N)
If Isconnect(M) then
    For do
        if f and // At any point in M, a neighbourhood through f mapping belongs to N //
        then return 1;
    else
        return 0;
    endif
    Endfor
else return 0;
endif
Endprocedure
```

From the previous description, there is a one-to-one relationship between a Lie group and its Lie algebra. Since the connected groups of connected Lie group and single-connected covering groups are unique in the equivalent sense, it is only necessary to find the general coverage group of the Lie group, which can be used to solve a group of problems. Each point on the group manifold has a one-to-one correspondence with the group element, and the connection of any two points in the group manifold can be transformed into the equivalent curve on the group by the continuous deformation of the parameter space. As long as the cover of all the unequal irreducible expressions is found, the corresponding Lie group will be found.

Theorem 6.8. *The connected group G must be connected, and the simply connected covering group (\tilde{G}, f) is called the universal covering group of G . Further, f is an isomorphism if and only if G itself is a single-connected Lie group.*

Next, we present an algorithm for the universal covering group of the Lie group and simplify the Lie group problem to the problem of covering it.

Algorithm 6.2: Universal covering group algorithm for connected Lie group SCCG

```
//Input: Connected Lie group G
//Output: Universal covering group of G and covering mapping
Step 1: Take the unit frame  $(U, \varphi)$ . Create point set  $\bigcup_{m=1}^{\infty} (\overline{U \times U \times \dots \times U})$  starting from  $U$ .
Step 2: Take the elements  $(u_1, u_2, \dots, u_m)$  and  $(v_1, v_2, \dots, v_p)$  of this point set. According to
 $u, v, uv \in U$ , we have the role  $(u, v) = uv$ . Further,  $(u_1, u_2, \dots, u_m)$  changes with a limited
step; check whether it can become  $(v_1, v_2, \dots, v_p)$ ;
Step 3: The equivalence class of the element that satisfies the equivalent condition is denoted by
 $(u_1, u_2, \dots, u_m)^*$ . Continue to select other elements;
Step 4: Make all the equivalence classes that fit the equivalence relation into a set of  $\tilde{G}$ .
Step 5: If the covering mapping  $f: (u_1, u_2, \dots, u_m)^* \rightarrow u_1 u_2 \dots u_m$  is established between the
equivalence class set and  $(u_1, u_2, \dots, u_m)$ , then  $(\tilde{G}, f)$  is the coverage group of  $G$ .
```

The construction of the covering group shows that df gives the isomorphism between the Lie algebra of \tilde{G} and the Lie algebra of G . The reason is that for the unit elements \tilde{e} and e of \tilde{G} and G , there exists the neighbourhood coordinates U of the three-dimensional coordinates of the neighbourhood \tilde{U} and G , so that f is the homeomorphism of \tilde{U} to U , and df gives the algebraic isomorphism of the Lie algebra.

After finding the coverage group of the connected Lie group, for the unequal irreducible representation of the covering group, we must first find all their unequal irreducible signatures and list the signatures as the characteristic table. Then, we find the irreducible representation matrix. Since the group elements can be expressed as the product of the generator, it is sufficient to find the representation matrix of the generator. In fact, the common group of irreducible representation of the standard table, i.e., the matrix of the standard form and irreducible representation of the direct representation of decomposition, can be found in the book, and they only need to be applied to the learning problem.

6.2.3 Algorithm design

The purpose of learning is to find the optimal path from the initial state to the target state. If each state is called a node, it can form a group space. The group space is single-connected, i.e., each state can reach another state. In each connected state, each arc represents an operator, and one state can be directed to another state. Of course, according to different problems, a specific price can be set for the operation operator. If the starting point is from the initial node, and there is a path leading to the target node, then we can say that the learning problem is solvable. A path from the initial state to the target state forms an overlay.

To better describe real-world problems, we can map the sample data set D to a new Lie group structure. However, the relationship between the various elements will not change corresponding to the original group of elements. On the one hand, we can regard D' as a new set of samples produced by D , i.e., the coverage of the sample set D does not change its essential characteristics, thus ensuring the correctness of the coverage mapping. On the other hand, for the two isomorphic sample sets D and D' , the same road division method can be used, thus avoiding repetitive learning.

For example, for the individual points in the face (eyes, nose, mouth, etc.), take a number of sample points and their neighbourhood points. The continuous coverage of the mapping between the sample points maintains the original neighbourhood relationship, i.e., in the original space, the adjacent sample points remain adjacent in the mapped group.

Since the two Lie groups have local isomorphism, the topological properties of the original Lie group are maintained (both have the same Lie algebra). We know that the Lie algebra represents the reciprocal relation of the generators; hence, the groups also have the same generators. At the same time, the Lie algebra can be understood as a set

of left isometric vector isomorphisms. Hence, solving the covering map is equivalent to dividing the different classes by using the covering group in the sample set.

According to the above description, we can get a simply connected covering algorithm for Lie group machine learning (LSCA).

Set the sample set D to $D = \{r^t = (x^t, y^t), t = 1, 2, \dots, s\}$, where x^t represents the set of eigenvalues of the sample and y^t represents the weight of the sample eigenvalue. All the samples form a feature space, which is nested in a differential manifold structure that constitutes a simple Lie group. Each sample point is an element in the group j .

- (1) Let the input sample set $X = \{x^1, x^2, \dots, x^n \in R^D\}$, where X satisfies the Lie group structure.
- (2) The SCCG algorithm is used to obtain the coverage and coverage mapping.
- (3) Find the generator and structure constants of the covering group.
- (4) For a new sample, it is mapped to the coverage group by covering the mapping to determine if there is a path between the point and the unit of the coverage group and whether the path has homotopy to zero.
- (5) If the condition A is satisfied, it indicates that the new sample is in accordance with the output; the new sample is added to the cluster group of the covering group, and the operator of the coverage map is adjusted.
- (6) If this is not the case, the sample does not belong to the target result to be studied.
- (7) Instance test.

6.2.4 Example application analysis

Quantitative structure-activity relationship research is a very important topic in modern drug design methods, and it is the most widely used drug design method. The purpose of structure-activity relationship analysis is to study and reveal the quantitative change between compound activity and its molecular structure or physicochemical characteristics. To verify the effectiveness of the algorithm, the experimental example used in this section is dihydrofolate reductase (DHFR). Dihydrofolate reductase is present in many biological organisms. The inhibitor can selectively bind to DHFR and inhibit its biochemical enzyme activity so that dihydrofolate cannot be successfully converted into tetrahydrofolate, thus inhibiting folic acid metabolism, interfering with DNA and protein synthesis, and ultimately leading to cell death. Therefore, DHFR inhibitors have anti-cancer, anti-bacterial, and anti-malarial effects.

6.2.4.1 Preparation for experimenting

The biological activity data (IC_{50}) of the 761 DHFR inhibitors studied in this paper are derived from the literature [34, 35], i.e., from three different biological organisms, and the selected data are the activity data under PC conditions. Using $pIC_{50} = 6.00$ as an

active classification standard, the 761 inhibitors were divided into two classes, namely active ($\text{pIC50} \geq 6.00$) and inactive ($\text{pIC50} < 6.00$), with 296 active molecules, labelled as (+), and 465 inactive molecules, labelled as (-).

For each molecule, 158 molecular descriptors [36] were used to characterise their structure and physicochemical properties. These descriptors include the composition descriptor and the topological exponent descriptor.

6.2.4.2 Experimental process

In the experiment, we first use the Kennard–Stone method to use three quarters of the sample as the training set and the remaining as the test set. Thus, the training set has 570 molecules (active 222, inactive 348), and the test set has 191 molecules (active 74, inactive 117).

In the following, we apply the Lie group machine learning method to the establishment of the forecasting model. First, the training set constructs a Lie group structure, and each sample molecule constitutes an element of the Lie group. The coverage and coverage mapping can be found using the single-connected overlay algorithm. By covering the mapping, we can establish a one-to-one relationship with the elements of the Lie group and the elements of the covering group. This transforms the Lie group problem into the problem of covering the group, i.e., the nonlinear problem is transformed into a linear problem.

Through the study of the sample instance, adjust the control factor of the coverage mapping, and map the same sample to the coverage group. For a new sample, it is determined whether the sample belongs to the class by mapping it to the covering group by covering the mapping and determining whether there is a path to the generator of the covering group.

To evaluate the model's ability to fit the training set and the ability to predict the training set, three indicators were used:

$$\text{sensitivity} \quad S_n = \frac{\text{TP}}{\text{TP} + \text{FN}} ;$$

$$\text{specificity} \quad S_p = \frac{\text{TN}}{\text{TN} + \text{FP}} ;$$

$$\text{average accuracy} \quad A_c = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{TN} + \text{FP}} ;$$

where TP, TN, FP, and FN represent the correct number of positive samples, the correct number of negative samples, the number of positive samples, and the number of negative samples, respectively. Further, S_n is used to evaluate the correctness of the positive samples, S_p is used to evaluate the correctness of the negative samples, and A_c is used to evaluate the prediction accuracy of the overall sample.

6.2.4.3 Experimental results

We used the Liechang machine learning method to train active and inactive molecules, respectively, and established the optimisation model of active classification prediction. The prediction results of the test set are as follows:

A total of 191 molecules were tested (74 active and 117 inactive). After the model was predicted, the predicted results were 72 (+) molecules and 119 (-) molecules; the real activity was 65 and the real inactivity was 110.

$$Sn = TP / (TP + FN) = 65 / (65 + 7) = 90.28\%$$

$$Sp = TN / (TN + FP) = 110 / (110 + 9) = 92.44\%$$

$$AC = (TP + TN) / (TP + FN + TN + FP) = 175 / 191 = 91.60\% .$$

From the prediction results, the active probability of the active molecules predicted by the training method of the Lie group was 90.28%, and the prediction accuracy of the total samples was 91.60%. This shows that the method can not only ensure that the credibility of the forecast is relatively high but also guarantee that a certain percentage of true and effective candidate molecules can be predicted, so as to ensure that the test has a certain number of molecules to choose from.

At the same time, we compare the results of this algorithm with those of established optimisation models and predictions [36], as shown in Figure 6.3.

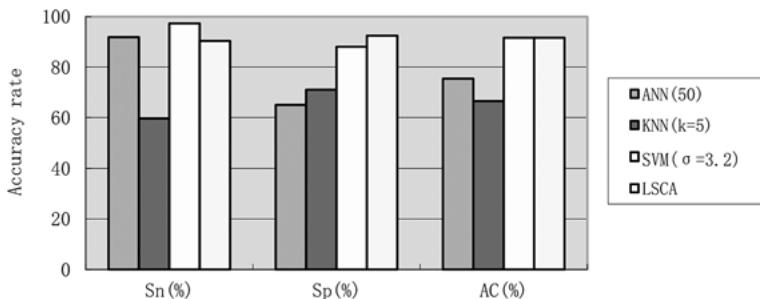


Fig. 6.3: Comparison of prediction results of various algorithms.

It can be seen that compared with other algorithms, the predictive model established by the method we use has better predictive accuracy for both positive and negative samples. Although the correctness of this method is not as good as that of SVM, the accuracy of the negative sample prediction is higher. Relatively speaking, the method achieves satisfactory results.

6.3 Multiply connected covering algorithm of Lie group machine learning

In many cases, the learning problem is not a one-to-one correspondence. It often has multiple input spaces and output fields, as well as multiple inputs and outputs. In the learning process, it is often difficult to analyse the multi-valued representation prob-

lem directly. Therefore, we solve the problem by transforming the Lie group problem into its coverage group and using the isomorphic relation between the Lie group and the covering group. Many of the overall Lie group properties can be derived from the infinitesimal structure of the group, i.e., given the number of pieces of the group, the overall nature can be completely determined. Therefore, this chapter studies the multiply connected covering problem of Lie group machine learning.

6.3.1 LML multiply connected covering model

In the group space of the Lie group, the correspondence between an element and a constant element is connected by many connections. Some of these connections can change continuously in the group space, whereas others cannot. Thus, these connections are divided into groups. The same group of connections can continuously change in the group space, whereas different groups of connections cannot. The number of groups of these connections is the connectivity of the group space. This chapter considers the nature of the Lie group and its covering groups as well as a practical application in the case of multiply connected cases.

It has been proven mathematically that when the group space of the connected Lie group G is multiply connected, it must be homomorphic to another group space of a simple connected Lie group, and the homomorphism correspondence is $1 : n$. The Lie group is called the covering group of Lie group G . The true representation of the coverage group is the n value of group G . The existence of the covering group is the mathematical basis of the multi-valued representation of the Lie group.

The connections within the group space of the multiply connected Lie group can be divided into groups according to whether they can be continuously changed, and the connections within the same group can change relative to each other. Thus, the group space is divided into a number of non-linked pieces, where each piece represents an independent part that can be found in the covering group. The covering group is homogeneous with the Lie group, so the combinatorial function of the covering group is a true reflection of the properties of the multi-connected Lie group. According to the feature of the covering group, the complex multi-connectivity problem can be reduced to the grouping problem.

According to the nature of the Lie group, the road from any point in the group space to the invariant can be shrunk, i.e., the road is equivalent. An overlay is the equivalent set of all roads in the group space. If the road f and g of \tilde{G} are in the road class $[a]$, there are fixed-point shifts $H : [0, 1] \times [0, 1] \rightarrow \tilde{G}$, $H(t, 0) = f(t)$, $H(t, 1) = g(t)$. Further, $s_0 \in [0, 1]$ is fixed such that $0 = t_0 < t_1 < \dots < t_n = 1$ is a detailed division of the road $H(t, s_0)$. By the continuity of H , it is possible to select a fixed positive δ that is sufficiently close to zero. When $s \in [s_0 - \delta, s_0 + \delta]$, the above $\{t_0, t_1, \dots, t_n\}$ is a detailed division of the road $H(t, s)$; then, $H(t_{k-1}, s)^{-1}H(t_k, s) \in U$. In addition,

the above A can be reduced to fix

$$\begin{aligned} H(t_{k-1}, s)^{-1}H(t_k, s_0) &\in U, \\ H(t_k, s)^{-1}H(t_k, s_0) &\in U, \\ H(t_{k-1}, s_0)^{-1}H(t_k, s) &\in U (1 \leq k \leq n). \end{aligned}$$

Because $U = U^{-1}$, $H(t_k, s_0)^{-1}H(t_k, s) \in U$.

For $1 \leq k \leq n$, we have

$$\begin{aligned} F_1[H(t_0, s_0)^{-1}H(t_1, s_0)] \dots F_1[H(t_{k-1}, s_0)^{-1}H(t_k, s_0)] \\ = F_1[H(t_0, s_0)^{-1}H(t_1, s)] \dots F_1[H(t_{k-1}, s)^{-1}H(t_k, s_0)]F_1[H(t_k, s)^{-1}H(t_k, s_0)]. \end{aligned}$$

Then, let $\alpha_{s_0}(t) = H(t, s_0)$, $\alpha_s(t) = H(t, s)$. We have $F(\alpha_{s_0}) = F(\alpha_s)$. Therefore, it can be easily seen that $F(f) = F(g)$. In other words, $F(\alpha)$ depends only on the road class $[\alpha]$.

Thus, the mapping $F_2: G \rightarrow G_2$ is defined, where G_2 is another slice that is not connected in the group of Lie group G , and G' is its covering group. Then, $F_2(x) = \sigma F(\alpha)$, σ is its control factor, where α represents any way of connecting the units E and F within G' . Since F_2 is a covering homomorphism, $\forall x, y \in G'$, for road β connecting $e_{G'}$ to point y and the road α connecting point $e_{G'}$ to point x , $x\beta$ is a path of point x to point xy . The product road A is a road connecting B to xy because

$$\begin{aligned} F_2(xy) &= F_2(\alpha * (x\beta)) = F_2((\alpha(t_0))^{-1}\alpha(t_1)) \dots F_2((\alpha(t_{n-1}))^{-1}\alpha(t_n)) \\ &\quad F_2((x\beta(t_0))^{-1}(x\beta(t_1))) \dots F_2((x\beta(t_{n-1}))^{-1}(x\beta(t_n))) \\ &= F_2(\alpha)F_2(\beta) \\ &= F_2(x)F_2(y). \end{aligned}$$

Let $\forall x \in V$ and take a road α of V to connect the unit and x in G' ; make $0 = t_0 < t_1 = 1$ a fine division of $[0,1]$. Then, we have

$$F(\alpha) = \sigma_2 F_2(x) \times \sigma_1 F_1(x) = \sigma_1 \sigma_2 F_2[(\alpha(t_0))^{-1}\alpha(t_1)]F_1[(\alpha(t_0))^{-1}\alpha(t_1)]$$

where σ_1, σ_2 are the control factors covering groups \tilde{G}, G' , respectively. Hence, the nature of the multi-connected Lie group can be expressed by the combination of the non-connected pieces. For practical problems, we only need to find the control factor, and we can complete the representation problem.

Since the infinitesimal element only describes the local properties of the Lie group, as the group space is a multi-connected Lie group G , it has a common small element with its coverage group. According to Lie's second theorem, a set of matrices satisfying certain conditions is used as the generator of the simple Lie group. The linear representation of the solution may be the true representation of the covering group and the multi-valued representation of group G .

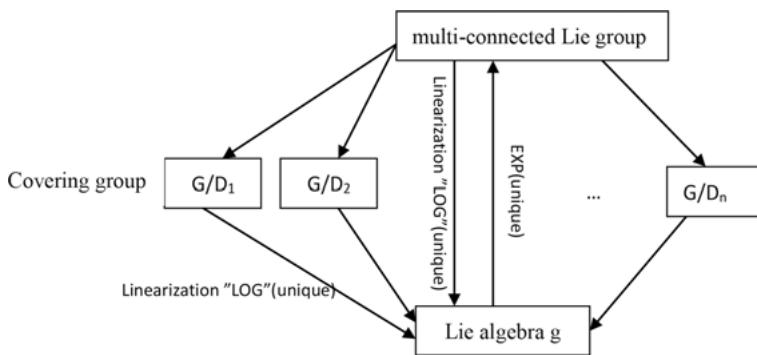


Fig. 6.4: The Relationship between a multiply connected Lie group and its covering group.

Figure 6.4 shows the relationship between the multi-connected Lie group and its covering group, and the combination of each covering group can fully express the properties of the multi-connected Lie group. Hence, when the pieces of the Lie group are given, the overall nature can be completely determined.

The Lie algebra of the multi-connected Lie group G is g , G_1, G_2, \dots, G_n is the covering group of G , and the Lie algebra is g_1, g_2, \dots, g_n . Then, the Lie algebra of the covering group and the multi-connected Lie group can be decomposed as in Figure 6.5:

$$K \times G_1 \times \dots \times G_m \quad \xleftarrow{\pi} \quad G \\ \downarrow \exp \qquad \qquad \qquad \downarrow \exp \\ Z(k) \oplus g_1 \oplus \dots \oplus g_m \quad \xleftarrow{id} \quad g$$

Fig. 6.5: The Lie algebra exploded map of the covering group and the multi-connected Lie group.

Where K is the fitting control between the covering group and the multi-connected Lie group. Therefore, the properties of the multi-connected Lie group can be analysed by decomposing each covering group, and the algebra is also connected with homomorphism.

6.3.2 Multiply connected covering algorithm design

Corresponding to the learning problem, each sample is characterised by a specific attribute value, assuming that there are several categories of problems, denoted as A . Each sample in the sample set is represented in the learning space according to the attribute value, and each sample in the sample set is represented in the learning space according to the attribute value. It is embedded in a differential manifold, which constitutes a Lie group. Each sample is an element in the group. The element group R can be described by a set of independent real parameters ($r_1, r_2 \dots$) that change continuously within a certain region. Samples of the same category have similar or equal

attribute values, and the connection between any two points should be connected, i.e., a change in parameters can represent a class of problems. Some of the different types of property values may also be similar or equal, but there must be different attribute values. Of course, we can first determine the key attribute values of each category. The unit of the sample formed between the mapping relationship is called the road. The homotopy of the road can form a coverage group, and the coverage group contains the same sample. Different sets of categories are not interconnected, so the non-connected parts of the group space respectively represent different categories.

The following is a group space segmentation algorithm for multi-connected Lie group:

Step 1: Take the unit frame (U, φ) . We can create a point set starting from U

$$\bigcup_{m=1}^{\infty} \overbrace{(U \times U \times \cdots \times U)}^m.$$

Step 2: Take the elements (u_1, u_2, \dots, u_m) and (v_1, v_2, \dots, v_p) of this point set. According to the role that if $u, v, uv \in U$, then $(u, v) = uv$, we can perform a finite step change for (u_1, u_2, \dots, u_m) , and check whether it can be changed to (v_1, v_2, \dots, v_p) .

Step 3: The equivalence class of the element satisfying the equivalent condition is denoted as $(u_1, u_2, \dots, u_m)^* \subseteq G_i$. The element that does not conform to the equivalent condition is the element of G_{i+1} ; then, another element is selected.

Step 4: Continue until all the elements in the point set have been traversed and all equivalence classes that conform to the equivalence relation form G_i .

Step 5: From an element of A , find the equivalent element, into the collection, until all the elements are divided into the corresponding group space.

The group space of the multiply connected Lie group is divided into several blocks that are not relevant.

By dividing the group space of the multiply connected Lie group, the group space of the multi-connected Lie group can be divided into several different slices, and each slice contains the equivalent elements, which are equivalent to the road. By using the coverage group of the Lie group, the elements of different classes are distinguished, and the overall properties of the multiply connected Lie group are obtained by analysing the local properties.

For two given groups G' and G'' , use $a'_k \in G'$ and $a''_l \in G''$ to represent their respective elements. Define the element $a_{kl} = (a'_k, a''_l)$, and define the multiplication $(a'_1, a''_1)(a'_2, a''_2) = (a'_1 a'_2, a''_1 a''_2)$. Then, the element (a'_k, a''_l) constitutes a new group G , and the product of the ranks n' and n'' of G' and G'' is $n = n' n''$. Group G is called the direct product of groups G' and G'' , denoted by $G = G' \times G''$.

It is important to note that only the two groups are acting in different spaces and are independent of each other, and when they are considered for their effects, they

can be the product of their straight product. Fortunately, the multi-connected coverage groups considered in this paper are simply non-connected segments in the group space, so they can be expressed in the form of a straight product when seeking the properties between the two groups.

If group G is given by the direct product of group G' and G'' , if D' and D'' are respectively a representation of G' and G'' , then $D = D' \times D''$ is a representation of G . If D' and D'' are the irreducible representations of G' and G'' , respectively, then the resulting D is an irreducible representation of G .

Here, we give the Lie group machine learning multi-connected coverage algorithm, described as follows:

- (1) Let the input sample set be $X = \{x_1, x_2, \dots, x_n \in R^D\}$, where X satisfies the Lie group structure
- (2) According to the group space structure of the Lie group, the group space of Lie group is divided into several slices by using the group space segmentation algorithm.
- (3) By using the covering group algorithm (SCCG) for solving the Lie group, the covering group C_i and the covering map f_i are obtained for each group of the group space.
- (4) First, the coverage group with the largest dimension is analysed to find out its generating element representation. The road set between the elements in the group and the generator is expressed as the study of the slice data, and the control factor is obtained.
- (5) For the next coverage group to learn, it will be connected to the previous coverage group, i.e., the two groups of overcrowding.
- (6) Find the corresponding control factor σ_{i+1} , so that the two groups can be expressed as $C' = \sigma_i C_i \times \sigma_{i+1} C_{i+1}$. Thus, the nature of the whole Lie group can be represented by multiple covering groups.

Through the analysis of the Lie group properties, we can get the structural properties of the whole Lie group. The advantage of the overlay algorithm is that the Lie group structure is decomposed and mapped to each covering group, which effectively improves the efficiency of the algorithm and enhances the application range of the Lie group. For different Lie group problems, through the learning process, to find the control factor of each coverage group, we will represent the problem as a group representation. For some type of learning problem, the relationship between the input space and the output space can be expressed as

$$y_t = \sigma f_1(x_t) \times f_2(x_t) \times \cdots \times f_n(x_t).$$

According to the above process, the coverage group control factor is found in the learning process through the local isomorphism relation between the Lie group and the covering group, so that the problem of Lie group representation can be expressed by its covering group.

6.3.3 Applications

In this section, we validate our multiply connected covering model through an important Lie group. The SO(4) group is an orthogonal transformation group of four-dimensional real space. It is a compact Lie group, and the Lorentz group is very close, but the parameters of the real terms are different. The SO(4) group is a typical multi-connected Lie group. By studying the relationship between the SO(4) group and its coverage group, we can verify our idea of multiply connected algorithm design.

We know that the SO(4) group is a double-connected Lie group. First, the SO(4) group is expressed by the SO(4) group itself, and the relationship between the SO(4) group and the two SU(2). Further, the group elements of the SO(4) group are grouped as two straight sets of SU(2), specifically, to determine the 1:2 homomorphic relationship. According to this correspondence, we can use the corresponding parameters of the SU(2) group to find the unequal irreducible analytic form of the SO(4) group.

The SO(4) group itself has six generators, making appropriate combinations of these generators, such as

$$T_1^{(\pm)} = \frac{1}{2}(T_{23} \pm T_{14}) = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & \mp 1 \\ 0 & 0 & -i & 0 \\ 0 & i & 0 & 0 \\ \pm i & 0 & 0 & 0 \end{pmatrix}$$

and its 1,2,3 cycle. The above equation can be expressed as a direct form of a two-dimensional matrix.

$$\begin{aligned} T_1^{(+)} &= \frac{1}{2}\sigma_2 \times \sigma_1, & T_2^{(+)} &= \frac{-1}{2}\sigma_2 \times \sigma_3, & T_3^{(+)} &= \frac{1}{2}l_2 \times \sigma_2, \\ T_1^{(-)} &= \frac{-1}{2}\sigma_1 \times \sigma_2, & T_2^{(-)} &= \frac{-1}{2}\sigma_2 \times l_2, & T_3^{(-)} &= \frac{1}{2}\sigma_3 \times \sigma_2. \end{aligned}$$

They are divided into two sets of generators to satisfy the SU(2) group of generative pairs of easy relations.

$$[T_a^{(\pm)}, T_b^{(\pm)}] = i \sum_{c=1}^3 \epsilon_{abc} T_c^{(\pm)}, [T_a^{(+)}, T_b^{(-)}] = 0.$$

After the above formula is changed to N , the decomposition can be seen more clearly:

$$\begin{aligned} N^{-1} T_a^{(+)} N &= (\sigma_a/2) \times l_2, & N^{-1} T_a^{(-)} N &= l_2 \times (\sigma_a/2), \\ N &= \frac{1}{\sqrt{2}} \begin{pmatrix} -1 & 0 & 0 & 1 \\ -i & 0 & 0 & -i \\ 0 & 1 & 1 & 0 \\ 0 & i & -i & 0 \end{pmatrix}. \end{aligned}$$

Thus, any element of the SO(4) group may be expressed as

$$\begin{aligned}
R &= \exp \left(-i \sum_{a<b}^4 w_{ab} T_{ab} \right) \\
&= \exp \left\{ -i \sum_{a=1}^3 \left(w_a^{(+)} T_a^{(+)} + w_a^{(-)} T_a^{(-)} \right) \right\} \\
&= \exp \left\{ -iw^{(+)} \hat{n}^{(+)} \cdot \vec{T}^{(+)} \right\} \exp \left\{ -iw^{(-)} \hat{n}^{(-)} \cdot \vec{T}^{(-)} \right\} \\
&= N \left\{ u(\hat{n}^{(+)}, w^{(+)}) \times u(\hat{n}^{(-)}, w^{(-)}) \right\} N^{-1}
\end{aligned} \tag{6.1}$$

where

$$w_1^{(\pm)} = w_{23} \pm w_{14} = w^{(\pm)} n_1^{(\pm)},$$

$$w_2^{(\pm)} = w_{31} \pm w_{24} = w^{(\pm)} n_2^{(\pm)}, \quad w^{(\pm)} = \left\{ \sum_{a=1}^3 (w_a^{(\pm)})^2 \right\}^{1/2}$$

$$w_3^{(\pm)} = w_{12} \pm w_{34} = w^{(\pm)} n_3^{(\pm)}.$$

It can be seen that the matrix is clearly expressed as the product of two two-dimensional unitary matrices u . The two A matrices simultaneously change the sign, and the B matrix remains unchanged. Thus, (6.1) gives a two-to-two correspondence between the SO(4) group element and two SU(2) group direct-passing elements, and this correspondence relationship remains constant for the group element product. Thus, $\text{SO}(4) \sim \text{SU}(2) \times \text{SU}(2)'$. The parameters of the two SU(2) groups can be chosen as the group parameters of the SO(4) group to make the parameters of the group space and the group elements have a one-to-one relationship. In determining the group space of SO(4), the group space of a group can be reduced by half, i.e., the change area of the SO(4) group parameter is as follows:

$$\begin{aligned}
0 \leq w^{(+)} &\leq 2\pi, \quad 0 \leq w^{(-)} \leq \pi, \\
0 \leq \theta^{(+)} &\leq \pi, \quad -\pi \leq \varphi^{(\pm)} \leq \pi
\end{aligned}$$

where $\theta^{(\pm)}$ and $\varphi^{(\pm)}$ are the polar and azimuth angles in the $\hat{n}^{(\pm)}$ direction. Since the group space of group A is reduced in general, it is similar to the group space of the SO(3) group. The points at both ends of the diameter of the group space correspond to the same group of elements, which determines that the group space of the SO(4) group is double-connected.

Thus, the covering group of the SO(4) group can be obtained as a group $\text{SU}(2) \times \text{SU}(2)'$, i.e., the group space of the double-connected SO(4) group is divided into two pieces, and each covering group is an SU(2) group. The direct product of two SU(2) groups is the covering group of the whole SO(4) group. After selecting this set of parameters, the density function of the integral on the SO(4) group is

$$\begin{aligned}
dR &= \frac{1}{8\pi^4} \sin^2(w^{(+)}/2) \sin^2(w^{(-)}/2) \sin \theta^{(+)} \sin \theta^{(-)} \\
&\quad \times dw^{(+)} dw^{(-)} d\theta^{(+)} d\theta^{(-)} d\varphi^{(+)} d\varphi^{(-)}.
\end{aligned}$$

The unequal irreducible representation of the SO(4) group can be expressed as two direct products of the SU(2) group unequal irreducible representation, denoted as D^{jk}

$$D^{jk}(\text{SO}(4)) = D^j(\text{SU}(2)) \times D^k(\text{SU}(2)'),$$

$$D^{jk}(\hat{n}^{(+)}, w^{(+)}; \hat{n}^{(-)}, w^{(-)}) = D^j(\hat{n}^{(+)}, w^{(+)}) \times D^k(\hat{n}^{(-)}, w^{(-)}),$$

D^{jk} is $(2j+1)(2k+1)$ -dimensional. Its row (column) indicator is co-marked with two letters A, and its generator B can be represented by the SU(2) group corresponding to the generation element C.

$$I_a^{jk(+)} = I_a^j \times 1_{2k+1}, \quad I_a^{jk(-)} = 1_{2j+1} \times I_a^k,$$

$$I_{ab}^{jk} = \sum_{c=1}^3 \varepsilon_{abc} (I_c^{jk(+)} + I_c^{jk(-)}) = \sum_{c=1}^3 \varepsilon_{abc} (I_c^j \times 1_{2k+1} + 1_{2j+1} \times I_c^k)$$

$$I_{a4}^{jk} = I_a^{jk(+)} - I_a^{jk(-)} = I_a^j \times 1_{2k+1} - 1_{2j+1} \times I_a^k.$$

The nature and representation of the multiply connected Lie group can be obtained by its coverage group. In the real learning problem, the problem of multi-connected Lie group is often difficult to solve directly. According to the multi-connected coverage algorithm in this paper, we can find the coverage group, so that the problem of the connected group is transformed into the representation of the coverage group, and the problem is thus solved.

The main idea of this section is to verify the relationship between the multiply connected Lie group and its covering group. By analysing the SO(4) group of a typical multiply connected Lie group, it can be decomposed into the direct product of two SU(2) groups. Since the SO(4) group is a four-dimensional symmetric transformation group, $\det(A) = \pm 1$, for the coverage group of the SO(4) group, its coverage group control factor takes the unit element. In practical problems, we often need to learn the data set to determine its control factor.

6.4 Application of the covering algorithm in molecular docking

6.4.1 Introduction to the molecular docking algorithm

Molecular docking is the process of recognition through geometric matching and energy matching between two or more molecules. Molecular docking is very important in drug design. In the process involving drug molecules to produce pharmacological reactions, the drug molecules and target enzymes combine with each other. First, we need two molecules close to the appropriate orientation, so that their necessary parts can mutually fit and then interact with each other. Through appropriate conformational adjustment, the process results in a stable complex conformation. The relative position and orientation of the two molecules in the complex are determined by molecular docking, and the optimal binding pattern between the molecules is found.

The study of rapid molecular docking is an important topic for rational drug screening. The molecular docking algorithm, in fact, can be used as a predictor of ligand-receptor complexes, or a spatial search problem of system-bound energy extremes. Even if the receptor molecule is rigid, molecular docking involves a search space that is still very large, and the number of solutions is still considerable. Gehlhaar et al. used a flexible small molecule as an example to roughly estimate that the search space contained at least 10^{30} solutions.

The development of a fast and effective search algorithm has been the focus of many studies; even molecular docking has been considered for drug screening. In recent years, research on the molecular docking search algorithm has made significant progress. Many random search algorithms have been used to solve such problems, including simulated annealing (such as AutoDock2, DockVision, MCDOCK), genetic algorithm (GOLD, AutoDock3, DockVision), and tabu search (ProLeads). These algorithms are based on random search techniques and scoring functions, but they do not systematically detect the entire search space.

System-based randomised test methods also exist, including incremental construction and multiple conformer rigid body docking. Although these methods are systematic, they still cannot perform a complete search of conformational and orientation spaces. The incremental construction method uses a series of discrete rotation conformations to obtain a rough sample of the constellation. The multiple isomer rigid body docking system uses a series of low-energy isomers of rigid bodies.

Figure 6.6 shows the commonly used molecular docking algorithm classification [39].

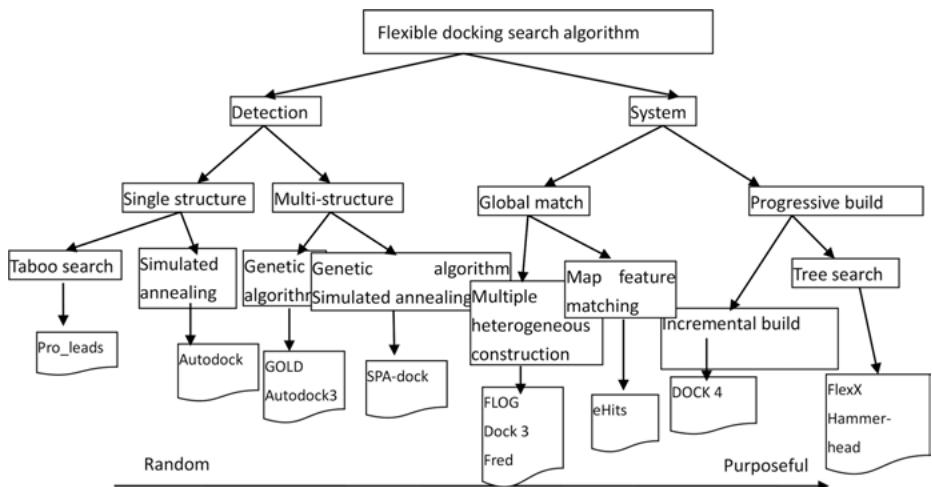


Fig. 6.6: Flexible docking algorithm classification.

A greater problem based on a random search is that it is difficult to say that the search must find the best conformation of the energy during a limited search time, so it is difficult to define the criteria for the end of the calculation. In a random search, if the new conformation cannot be obtained after several cycles, it is generally believed that the calculation has converged, but in fact, it may not have reached the best energy conformation, and there will be a missed search situation. Molecular docking based on progressive construction is usually faster than other molecular docking methods. However, because the molecular docking method based on progressive construction only considers the local interaction between the receptor binding site and the constructed fragment, it is difficult to evaluate whether the optimised structure is a ligand molecule of the global optimal constellation. The Lie group and its covering group exist between the local isomorphic relationship, corresponding to the fragment and the overall structure. We naturally consider the Lie group applied to the molecular docking algorithm. This section is based on the gradual construction of the search strategy, combined with the idea of the Lie group to design a new molecular docking algorithm.

6.4.2 Mathematical model and evaluation function of molecular docking

The molecular docking problem is to find the most stable binding state between the interacting molecules by setting the small molecular compound of the known three-dimensional structure into the binding site of the target molecule; optimising the position, orientation, and conformation of the small molecule compound; targeting the biomolecule interaction of the best combination of conformation; and calculating or predicting the interaction energy of ligand molecules and biological macromolecules. In the mathematical planning sense, minimising the interaction of the system of free energy can usually be described as the following optimal design problem [14, 40]:

$$\begin{cases} \min & f(x) \\ \text{s.t.} & gk(x) \leq 0, \quad k = 1, 2, \dots, q \end{cases} \quad (6.2)$$

where $x = \{T_x, T_y, T_z, R_x, R_y, R_z, T_{b1}, T_{b2}, \dots, T_{bn}\}^T$ is the design variable, T is the design variable of the q ($q = n + 6$) dimension, and $T_{b1}, T_{b2}, \dots, T_{bn}$ is the rotation angle of the rotatable key of the ligand molecule. It describes the conformational information of the ligand molecule. Further, n is the number of rotatable keys, T_x, T_y, T_z, R_x, R_y , and R_z represent the geometric centre of the ligand molecule and the degree of rotation, corresponding to the orientation of the ligand molecule (Pose), and $g(x)$ is the size of the design variable constraints:

$$\begin{cases} \underline{X} \leq T_x \leq \bar{X} \\ \underline{Y} \leq T_y \leq \bar{Y} \\ \underline{Z} \leq T_z \leq \bar{Z} \\ -\pi \leq \text{angel} \leq \pi, \quad \text{angel} = R_x, R_y, R_z, T_{b1}, \dots, T_{bn} \end{cases} \quad (6.3)$$

In this model, the optimal conformation of the ligand is represented by three translation components (T_x , T_y , T_z), three rotational components (R_x , R_y , R_z), and the rotation of the ligand (T_{b1} , T_{b2} , ..., T_{bn}), where the translation variable is used to search for the position space of the ligand, and the rotation variable is used to search for the orientation space of the ligand.

Further, $f(x)$ is the objective function, i.e., scoring function (scoring function). Score functions are usually found in the following categories. The score function is based on the force field using the non-bond interaction as a conformation score, and is sometimes added to the solvent effect considerations. The empirical score function based on the known binding constant to calculate the protein-ligand complex A set of training sets is used to fit the intrinsic functional coefficient of the multivariate linear regression method. The scoring function based on knowledge is based on the statistical data of the protein-ligand interaction in the X-ray diffraction structure database as the scoring standard. Here, we use a semi-empirical free-energy scoring function similar to that proposed by Böhm [41]:

$$\begin{aligned}\Delta G = & \Delta G_0 + \Delta G_{\text{rot}} \times N_{\text{rot}} + \Delta G_{hb} \sum f(\Delta R, \Delta \alpha) + \\ & \Delta G_{io} \sum f(\Delta R, \Delta \alpha) + \Delta G_{aro} \sum f(\Delta R, \Delta \alpha) + \\ & \Delta G_{lipo} \sum f^*(\Delta R)\end{aligned}\quad (6.4)$$

where N_{rot} is the number of rotatable keys that are frozen during the binding process; ΔG_{rot} , ΔG_{hb} , ΔG_{io} , and ΔG_0 are semi-empirical parameters obtained by fitting; $f(\Delta R, \Delta \alpha)$ is the penalty function deviating from the optimal conformation; and $\Delta G_{aro} = -0.70 \text{ kcal/mol}$. The last term indicates the contact score of the lipophilic atom, and ΔG_{lipo} is the semi-empirical parameter of the lipophilic exposure score, where $f^*(\Delta R)$ has different values at different ΔR .

$$f^*(\Delta R) = \begin{cases} 0 & \Delta R > 0.6\text{\AA} \\ 1 - (\Delta R - 0.2)/0.4 & 0.2A < \Delta R < 0.6A \\ 1 & -0.2A < \Delta R < 0.2A \\ 1 - (-\Delta R - 0.2)/0.4 & -0.6A < \Delta R < -0.2A \\ (\Delta R + 0.6)/0.2 & \Delta R = -0.6A . \end{cases} \quad (6.5)$$

These parameters and the form of $f^*(\Delta R)$ function are directly used by the reported results of Böhm. The function is always used as a corresponding evaluation function in most molecular docking programs. We use this function to help with other docking procedures for comparison of the conformational search results.

6.4.3 Covering strategy and implementation of molecular docking

In the case of flexible molecular docking, the ligand molecule undergoes a translational and rotational operation at the docking site of the receptor until an optimal conformation is found. The translational rotation of the ligand constitutes the Lie group, in which the rotatable bond is used to probe the ligand. In the case of flexible conformational spaces, the efficiency of a global search is acceptable for relatively small ligand molecules. However, if the rotatable bonds are relatively large, the orientation and conformational spaces of the ligand are searched globally. Thus, a considerable amount of calculation is required, which is time-consuming, and it is often difficult to get the desired results.

At the active site of the receptor, the ligand molecules are constantly aligned. The conformational search space is very large owing to the presence of the rotatable bond. Hence, the conformational space must be reduced.

This can be achieved by the following operations when considering the conformational changes of the flexible molecules. First, we need to determine rigid segments and search constants. For a flexible molecule, its rotatable dihedral angle can be changed. Hence, in the docking process, we consider a flexible molecule that can be regarded as a combination of multiple rigid fragments. Each rigid segment consists of a set of atoms connected by non-rotatable bonds, while the rigid segments are connected by rotatable bonds.

The first step in determining the rigid segment is the judgment of the ring, and the keys in all the rings can be regarded as rigid. The second step is to determine the rotation of the key. We can identify the type of key on both sides of the key type. In general, double bonds are regarded as rigid keys. Each rigid segment contains only non-rotatable keys. Figure 6.7 shows a basic schematic diagram for dividing a molecule according to a rotatable key.

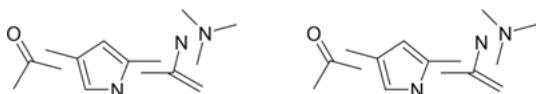


Fig. 6.7: Schematic diagram of flexible molecular division.

Then, the nature of the flexible molecule can be decomposed as in Figure 6.8:

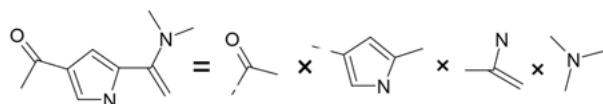


Fig. 6.8: Exploded map of Flexible Molecules.

Because the translational rotation of the ligand molecule constitutes a Lie group, and the Lie group is multi-connected, the rotatable bond of the ligand divides the group space of the Lie group into several different slices, which can be used by the multiply connected covering algorithm for analysis. The resulting conformation space is divided into different fragments by the group space segmentation algorithm. The molecular rotation of the key and the ring is defined as a split point, starting from any one element. Then, we can find its equivalent road class and merge them into a fragment until all the elements are traversed. The operation is thus complete.

According to the multi-connected covering algorithm, the properties of the molecule can be expressed by the following equation: $C = C_1 \times C_2 \times C_3 \times C_4$. In other words, as long as C_i is analysed, all irreducible unequal representations of C_i are found, and then, according to the local isomorphism of the covering group and the Lie group, it can be decomposed on the entire Lie group.

The next step is a conformational search of a flexible molecule. Based on the step-by-step method, after constructing the structural fragments, it is necessary to generate the conformation pattern of all the fragments, and the conformational database is used to store the typical conformation of these fragments. After the conformation of the structural fragments is obtained, the structural fragments are linked to each other to produce intact molecules. In the process of connection, because each structure fragment may exist with multiple conformations, the connection process uses a tree search method. However, the drawback of this approach is that the conformation between the structural fragments may not be a good substitute for the conformational pattern in the molecule, and the connection conformation between the structural fragments does not fully reflect the true conformation.

Therefore, we consider the local isomorphism between the Lie group and its covering group. The problem of Lie group representation can be decomposed into each covering group to represent and maintain the whole characteristic. Conformation searches for structural fragments can also be generated by the generation of the Lie group, which can cover the entire search space to the maximum extent by translation and rotation operations.

We know that the Lie group characteristic can be attributed to its generative relationship. Using generators, we can conveniently represent a group. For example, the Lie algebra of the three-dimensional rotating group is the linear space of its infinitesimal generating element, and the dimension of the Lie algebra is six. In the molecular docking model, the covering group obtained by the gradual construction is a three-dimensional Euclidean group $SE(3) = T(3) \times SO(3)$. The generator of the three-dimensional Euclidean group is derived from its group parameter $t = 0$.

If t is used as the parameter, the three rotation parameters and the three translation parameters are taken as the coordinates of point $G(X, Y, Z)$, and the six generators of $SE(3)$ can be obtained. Therefore, the group elements of $SE(3)$ are linear combinations of the six generators. Rotation around axes X , Y , and Z has the following

matrix form on R^3 :

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix} \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where α , β , and γ represent the rotation angles of the X , Y , and Z axes, respectively, and the corresponding generator is

$$e_1 = \begin{bmatrix} q & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, e_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & q & 0 \\ -1 & 0 & 0 \end{bmatrix}, e_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & q \end{bmatrix}.$$

The translation subgroup has no representation of the matrix on R^3 , but it is defined as a displacement operator in the vector space of the differential function on R^3 :

$$T_x(\phi): f(X, Y, Z) \rightarrow f(X + \phi, Y, Z),$$

$$T_y(\varphi): f(X, Y, Z) \rightarrow f(X, Y + \varphi, Z),$$

$$T_z(\theta): f(X, Y, Z) \rightarrow f(X, Y, Z + \theta),$$

where ϕ , φ , and θ represent the translational directions on the X , Y , and Z axes, respectively, and the infinitesimal generator is expressed as a partial differential operator:

$$e_4 = \frac{\partial}{\partial X}, \quad e_5 = \frac{\partial}{\partial Y}, \quad e_6 = \frac{\partial}{\partial Z}.$$

With these six infinitesimal generators, the infinitesimal transform of any three-dimensional Euclidean group can be expressed as

$$e = \sum_{i=1}^6 v_i e_i,$$

where v_i ($i = 1, \dots, 6$) is the coefficient of e_i .

In this paper, the conformational search method based on the covering group is used. As mentioned above, a molecule can be divided into multiple rigid segments connected by a rotatable key. The transformation of each segment in space actually constitutes a covering group, and the characteristic of each covering group reflects the characteristic of the whole molecule. We select one of the highest coverings as the core fragment.

When docking a ligand into an active pocket, it is first necessary to position, evaluate, and optimise to determine the position of the core fragment in the active pocket. After the core fragment position is determined, it is possible to connect the other segments on the core segment and search for conformation.

The specific steps are as follows (see Figure 6.9):

- (1) First, docking of the core fragment is carried out, and all the operations of the fragment at the active site constitute the covering group. The position space of the ligand is searched by the component of the parallel movement in the covering group model, and the orientation space of the ligand is searched by the rotation component. Get the possible docking position of the core fragment in the active pocket N₀.
- (2) For the resulting N₀ conformations, keep a portion of the best docking position according to the matching score. Then, reduce this part of the retention of the docking position. According to the position of the diversity of atoms, retain the representative of N_c conformations. The location diversity is evaluated using weighted root mean square deviation (wRMSD).

$$\text{wRMSD} = \frac{\sum_{i=1}^{N_{\text{atoms}}} L_i [(x_i - x_{ri})^2 + (y_i - y_{ri})^2 + (z_i - z_{ri})^2]}{\sum_{i=1}^{N_{\text{atoms}}} L_i}$$

where L_i is the control factor for the covering group. Note that wRMSD can be more sensitive to the overall characteristic after weighting L_i .

Corresponding to those values of wRMSD less than a certain threshold of the spatial location of the merger, retain only those with a large difference between the spatial locations. Finally, check the interaction between these core structures and receptors, and make simple modifications to the structure.

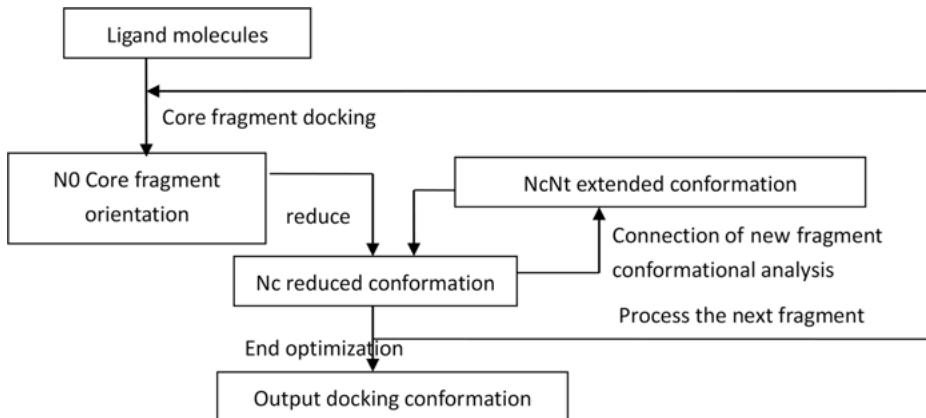


Fig. 6.9: Basic steps for core fragment search.

- (3) Connect the new fragment and use the multiply connected covering algorithm to directly multiply the two groups. Obtain the control factor of the covering group according to the directly multiplied weight space. At the same time, the conformational analysis is carried out to obtain N_cN_t conformations. After a straight mul-

tiplication, a new Lie group is obtained, and its conformation preserves the Nc optimal conformations according to the position diversity.

- (4) Repeat the third step and analyse the next covering group until the end of the fragment forming complete ligand molecules.
- (5) If there are multiple core clips, the operation returns to the first step and finds the docking position of all the core segments in the active pocket.

When the fragment is connected, the k best combinations of ligands and receptors can be obtained, from which the user can select the desired result. By combining the asymptotic search and the global search, we can maximise the search space by using the isomorphic relation between the Lie group and its covering group, so as to obtain the best conformation of the flexible molecule.

The basic flow of molecular docking simulation based on the Lie group machine learning covering algorithm described in this section is as follows.

Because the molecular structure itself is very complex, in our docking process, the pre-processing steps mainly use the molecular docking tool AutoDock 4.0 developed by Scripps Institute's Dr. Olson's research group. It includes the preparation of the ligand, the determination of the active point, and so on. In the next step, we will give some of the processing flow and parameters. In the molecular docking stage, MATLAB is used to calculate the docking accuracy between the ligand and the receptor, and the Böhm evaluation function is used to evaluate the results.

- (1) Preparation for the receptor. Because the PDB protein used in this experiment comes with a ligand, the first step is to remove the ligand. While removing the water, adding hydrogenation, adding charge, and other operations are performed to get the desired receptor for docking.

To remove water and ligands,

read

Pdb1wab.ent // Note that the Department of ID 1wab protein is an example to illustrate the process.

Do not CENTER the protein, you can pull it to the centre.

ATOM ... SET, Select WATER. Remove all water.

ATOM... SUBSTRUCTURES, select the box for amino acids to appear; pull down in the end, i.e., ACT300, which is the ligand, select DELETE.

Add hydrogenation:

ADD HYDROGEN ... ALL ... ESSENTIAL ONLY

Add charge:

Kollman_all or KILLMAN_UNI is all right.

Optimise the location of hydrogen (this step cannot be done; of course, it is best to do something to get theoretically better docking results):

Minimise

Method: Powell

Initial Optimisation: None

Termination: Gradient 0.01 kcal / mol

Max Iterations: 1000

Energy Setup Modify ...

Force Field: Tripos

Charges: Use Current

Check the Aggregates Box and Click on the ellipsis ... (A dialog appears)

Click on New (An Atom Expression Box appears)

Click on Atom Types and select H on the right. Click OK.

Click in Invert in the Atom Expression Box, then Click OK.

Call the aggregate NON_HYD. Click OK

Comment String ‘Non-hydrogen atoms’, Click OK

Click OK to start optimisation.

SAVE AS

1wab.mol2

- (2) Preparation for the ligand: the ligand is isolated from the PDB protein.

Open SYBYL.

Read

Pdb1wab.ent

Do not CENTER the protein, you can pull it to the centre. You can also point SYBYL (lower left corner of the icon), and RESET EXTENDS. You cannot find it manually.

ATOM ... SET, Select WATER. Remove all water.

ATOM ... SUBSTRUCTURES, select the box for amino acids to appear; pull down in the end, i.e., ACT300, which is a ligand,

OK, after the dialog box has an INVERSE. Select it; it will be in addition to the ligand other than the amino acid selected to remove all the amino acids. The ligand is actually CH₃COOH.

Hydrogenation:

Add hydrogen.

Geister Huckel Charge:

Save as ... lig.mol2

- (3) Determine the active site of the receptor molecule and use the blind docking program of the AutoDock procedures to find the pocket position on the receptor surface.

For the search of protein active sites, the accuracy of the protein mainly depends on the surface properties of the protein: if the surface of the protein is relatively smooth, the accuracy is relatively high. If the surface is more uneven, the results will not be very accurate, so it is necessary to use empirical knowledge to carry on subjective judgment. For the purpose of this article, we directly use blind docking to get the active site pocket position for the next step.

- (4) According to the Lie group space segmentation algorithm, the conformation space of ligand molecules is divided into different fragments; each segment is a covering group.

The ligand molecule is in the form of mol2, which mainly preserves the elemental information and topological properties of the molecule. All the operations of the ligands in the topological space constitute a Lie group, and the conformation space resulting from the group space segmentation algorithm is divided into different fragments. The molecular rotation of the key and the ring is defined as a split point. Starting from one any element, find its equivalent road class and merge it into a fragment. The operation will be completed after all the elements are traversed.

- (5) Select a fragment with the highest dimension as the core fragment. The position space of the ligand is searched by the component of the parallel movement in the covering group model, and the orientation space of the ligand is searched by the rotation component. Dock it to the active site of the receptor, and get the possible docking position of the core fragment in the active pocket N_0 .

In the covering group model, any change can be expressed as an infinitesimal generator of the translation component and the rotation component, i.e., $e = \sum_{i=1}^6 v_i e_i$. Hence, in the entire search space, two parts of the search are needed. Constantly adjust the translation component and rotation components. Use the evaluation function to score, and take the top N_0 scores as the highest position, where the parameter of the translation component $\sigma = 0.5 \text{ \AA}$ and the parameter $\varphi = 10^\circ$ of the rotation component.

- (6) Using the multiply connected covering algorithm in the previous chapter to grow the fragment, adjust the control factor of the coverage group, and connect the multiple segments to find the best conformation until the end of the fragment to form a complete ligand molecule.

In the process of fragment growth, for the operation of connecting two fragments, i.e., the two coverage groups for direct multiply, in which the control factor of the covering group needs to be adjusted, the adjustment step is carried out according to the weight vector of the weight space represented by the group, $\Lambda = \sigma_1 \Lambda^{(1)} + \sigma_2 \Lambda^{(2)}$, where σ_1, σ_2 are set to 1. Then, set the rotation component parameter $\varphi = 10^\circ$, using the evaluation function to score the results, while constantly adjusting the values of σ_1, σ_2 .

- (7) Perform multiple rounds of docking using the evaluation function to analyse the results to find the best conformation. Here, consider the time complexity ten times.

6.4.4 Experimental results and analysis

The algorithm is modelled on a data set consisting of 19 protein-ligand complexes derived from the PDB database (www.rcsb.org/) in the literature [42], as shown in Table 6.1.

Tab. 6.1: Experimental data sets.

No.	Kinds	Ligand	Protein
I		5tim	triose-phosphate isomerase
II	Complex with small ligand	1ldm	lactate dehydrogenase and oxamate
III		2phh	p-hydroxybenzoate hydroxylase and p-hydroxybenzoate
IV		3ptb	Benzamidine
V		1ulb	purine-nucleoside phosphorylase with guanine
VI	Short peptides	3tpi	trypsinogen with the dipeptide Ile–Val
VII		4tsl	tyrosyl-transfer-RNA synthetase
VIII	Dihydrofolate reductase	4dfr	Dihydrofolate reductase
IX	Streptavidin	1stp	streptavidin-biotin
X	Thrombin	1dwd	NAPAP
XI		1dwC	argatroban
XII	Ribonuclease	6rsa	uridine vanadate
XIII		1rnt	nucleotide guanosine 2'-monophosphate
XIV	Thermolysin	1tmn	N-(1-carboxylato-3-phenylprop-1-yl)-L-leucyl-L-tryptophan
XV		4tln	L-leucyl-hydroxylamine
XVI	Carboxypeptidase A	3cpa	L-leucyl-hydroxylamine
XVII		2ctc	L-phenyllactate
XVIII	HIV protease	4phv	Inhibitors of HIV-1 protease
XIX	RAS protein	121p	inhibitor guanosine 5'-B,Gmethylene-triphosphate

Each type of ligand has a different nature, and *I*–*V* ligands do not need to be progressively constructed. *V*–*VI* contain short-chain amino acid ligands and have only one fragment; *VIII* contains two rings and 10 rotatable keys; *IX* structure is more special, i.e., there are two five-membered rings; *X*–*XI* contain two hydrophobic rings; *XII*–*XIII* contain two large fragments; *XIV*–*XV* contain Zinc in the active site; *XVI*–*XVII* can be divided into five fragments; *XVIII* is our relatively large test data, which contains 17 rotatable keys; and *XIX* contains Mg²⁺ in the active site.

At the time of each docking simulation, the numerator is divided into different segments according to the rotatable key, and the segment with the highest priority is found. The position space of the ligand is searched by the component of the parallel movement in the model, and the orientation space of the ligand is searched by the rotation component to obtain the possible docking position. The fragments are then grown, and multiple fragments are covered to give the best conformation of the whole molecule.

The crystal structures of these 19 protein-ligand complexes have been detected. Table 6.2 lists the accuracy of each ligand (based on the minimum root mean square deviation of the crystal conformation, in units of Å) and the complexity (the number of theoretical conformations).

The 31 protein-ligand complexes were calculated using the LCDock algorithm. Each compound was calculated ten times, and the best result was selected, i.e., the

Tab. 6.2: Accuracy and complexity of a 19 protein-ligand.

No.	PDB	Acc. \AA	Com.
I	5tim	0.04	1
II	1ldm	0.22	2
III	2phh	0.05	3
IV	3ptb	0.09	3
V	1ulb	0.18	1
VI	3tpi	0.22	$7.0 \cdot 10^5$
VII	4tsl	0.09	$1.2 \cdot 10^2$
VIII	4dfr	0.40	$3.9 \cdot 10^7$
IX	1stp	0.29	$6.7 \cdot 10^5$
X	1dwd	0.34	$2.5 \cdot 10^8$
XI	1dwc	0.47	$7.2 \cdot 10^9$
XII	6rsa	0.06	$1.2 \cdot 10^2$
XIII	1rnt	0.53	$8.1 \cdot 10^3$
XIV	1tmn	0.44	$7.8 \cdot 10^{10}$
XV	4tln	0.54	$2.4 \cdot 10^3$
XVI	3cpa	0.17	$1.7 \cdot 10^5$
XVII	2ctc	0.23	$4.5 \cdot 10^2$
XVIII	4phv	0.69	$1.1 \cdot 10^{13}$
XIX	121p	0.54	$2.6 \cdot 10^6$

lowest energy calculation result, and the RMSD value was obtained. If the docking small molecule conformation and protein structure of the reference molecular conformation are relatively close, the value of the root mean square deviation is relatively small, i.e., the docking is more successful. Normally, the RMSD value is less than the acc. value in the table above, which indicates that the docking is in a more accurate position.

The following table lists the experimental results of docking via the LCDock algorithm. It lists the free energy scores and root mean square deviations at the lowest energy, as well as the scores and rankings of the best conformation. At the same time, we compared the algorithm with two other molecular docking procedures, namely FlexX, which uses a gradual construction, and Gold, which uses a genetic algorithm to search for the conformation space; they use the Böhm semi-empirical free energy score equation as the evaluation function, which is convenient for comparison. The Böhm parameters are set as follows: $\Delta G_{\text{tot}} = 0.33$, $\Delta G_{hb} = -1.1$, $\Delta G_{i0} = -2.0$, $\Delta G_0 = 1.3$.

The second column in the above table lists the conductor numbers calculated by the algorithm satisfying the conditions. Columns 3 and 4 show the evaluation function scores and the RMSD values at the lowest energy level. Columns 5, 6, and 7 respectively represent the scores, RMSD values, and rankings in the conformation when the optimal conformation is obtained.

It can be seen that the red and bold labelled ligands in Table 6.3 yielded the best results in the comparison of the three docking methods, with a total of 13, accounting for 68.4% of the total. Compared to the FlexX and Gold docking programs, LCDock shows

Tab. 6.3: Comparison of LCDock and FlexX, Gold docking results.

PDB	No. of Sol.	Low-energy		Best prediction			FlexX ΔG	Gold ΔG	Cryst. ΔG
		ΔG	rmsd	ΔG	rmsd	rank			
5tim	6	-12.3	1.59	-9.6	0.83	2	-10.4	-12.9	-7.8
1ldm	25	-29.3	0.60	-29.3	0.60	1	-32.8	-31.2	-29.1
2phh	170	-30.1	0.53	-30.1	0.53	1	-32.1	-25.8	-30.3
3ptb	28	-29.3	0.48	-29.3	0.48	1	-28.3	-27.1	-30.2
1ulb	131	-18.6	0.65	-18.6	0.65	1	-17.9	-28.5	-23.2
3tpi	215	-23.2	0.58	-23.2	0.58	1	-24.0	-25.1	-28.1
4tsl	143	-39.1	0.71	-39.1	0.71	1	-11.8	-31.0	-21.9
4dfr	80	-62.9	1.34	-55.7	0.85	2	-62.2	-54.8	-53.7
1stp	165	-30.5	0.81	-30.5	0.81	1	-31.2	-74.5	-33.0
1dwd	301	-45.6	2.03	-38.0	0.63	20	-38.6	-49.1	-37.9
1dwc	181	-32.0	2.66	-27.1	1.20	11	-29.1	-43.0	-22.7
6rsa	115	-36.9	0.85	-36.9	0.85	1	-35.9	-28.2	-37.8
1rnt	123	-41.0	1.48	-30.1	0.96	25	-31.4	-34.2	-30.2
1tmn	237	-38.2	0.87	-38.2	0.87	1	-39.1	-41.3	-35.5
4tln	120	-13.8	4.50	-7.3	0.91	15	-8.6	-19.2	-9.4
3cpa	105	-35.3	2.48	-32.9	0.96	2	-32.9	-21.0	-24.1
2ctc	160	-28.6	-0.62	-28.6	0.62	1	-30.3	-21.3	-24.8
4phv	13	-36.7	0.94	-36.7	0.94	1	-38.2	-50.0	-42.8
12lp	41	-53.0	2.00	-41.5	1.14	37	-30.5	-24.9	-43.4

better experimental results. By comparing the crystal structure conformational function, it can be seen that the results of the LCDock program are closer and the docked RMSD values remain between 0.5 Å and 1.2 Å, see Figure 6.10.

While achieving the effect of molecular docking, we must also compare the time of molecular docking. Here, we compare the molecular docking time of the three programs, see Table 6.4.

Because the LCDOCK and FlexX programs are based on progressive construction, their run time is divided into two steps; BP represents the run time of the core fragment algorithm, and CC represents the gradual growth time. The comparison of the docking time for the different docking algorithms is shown in Figure 6.11.

Through the comparison of the running time, we can see that LCDock with the covering algorithm has obvious superiority; it achieves a relatively short time in most of the docking experiments, and the effect of docking is also better.

Because the progressive constructive strategy is based on the adopted covering, we make full use of the local isomorphism relation between the Lie group and its covering group, and we use the Lie group to generate a meta method to control the search space of the passage, which further enhances the search flexibility and effectiveness. Experiments show that our method improves the speed as well as the accuracy of molecular docking significantly.

Comparison of LCDock, FlexX and Gold

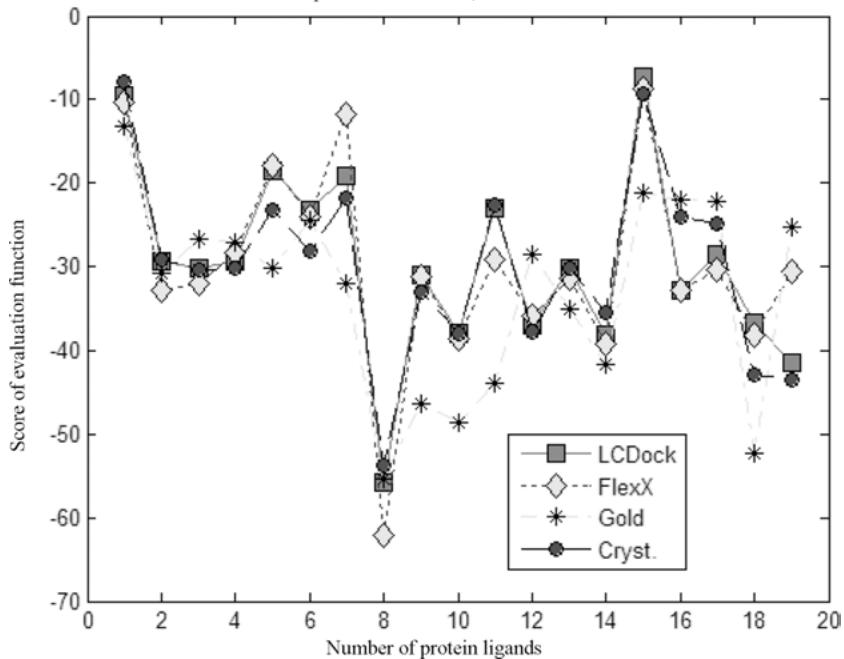


Fig. 6.10: Comparison of various docking methods' effects.

Tab. 6.4: Comparison of the time for molecular docking procedures.

PDB Code	Run time (LCDock)		Run time (Flexx)		Run time (Gold)
	BP	CC	BP	CC	
5tim	3.02	0.00	3.12	0.00	3.89
1ldm	11.42	0.00	11.38	0.00	10.56
2phh	35.20	0.00	36.37	0.00	41.20
3ptb	17.54	0.00	18.86	0.00	19.50
1ulb	32.56	0.00	46.56	0.00	55.20
3tpi	5.50	25.00	6.65	27.36	31.20
4tsl	6.15	5.32	6.30	7.11	18.70
4dfr	25.45	55.23	38.00	55.80	1:10.12
1stp	9.87	13.02	10.73	14.06	45.20
1dwd	20.64	23.01	24.02	1:21.94	1:12.32
1dwc	17.98	34.15	27.91	1:16.18	1:32.12
6rsa	23.16	12.08	34.64	11.38	55.25
1rnt	36.25	21.20	39.54	33.93	1:26.32
1tmn	15.20	32.25	21.47	1:25.86	1:45.65
4tln	0.58	1.21	1.45	1.66	12.38
3cpa	51.30	23.00	54.70	36.00	1:25.15
2ctc	35.20	5.89	1:01.98	5.50	1:05.23
4phv	0.35	20.23	0.65	1:30.45	1:12.81
12lp	15.25	45.30	31.91	49.49	1:35.23

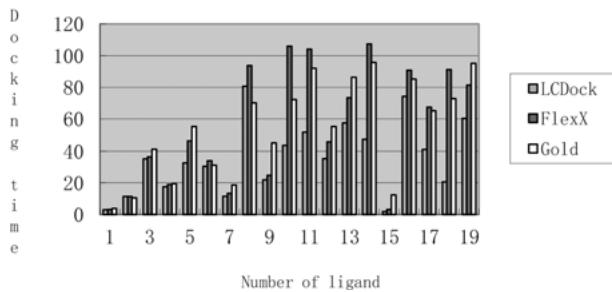


Fig. 6.11: Comparison of molecular docking time.

6.5 Summary

The idea of covering was introduced into the field of machine learning, and the simply connected covering algorithm and multiply connected covering algorithm for Lie group machine learning were presented. At the same time, the correlation learning algorithm was applied to drug molecule design to verify the effectiveness and superiority of the algorithm. The main contents of this chapter are as follows: 1. We studied the mathematical connotation of the mathematical properties of the Lie group, such as the whole nature, local properties, generating elements, group linear representation, and covering group. 2. The simply connected Lie group algorithm was applied to the model of drug structure-activity relationship. By changing the Lie group problem to solve its covering group problem, the efficiency of learning was improved effectively. 3. The multiply connected covering algorithm of Lie group machine learning was given for the complex learning model to solve the learning problems under multi-connected conditions. 4. Lie group machine learning was introduced into the field of drug molecular design and specifically applied to the structure-based relationship model and molecular docking model, yielding satisfactory results.

Bibliography

- [1] Li FZ, Xu H. The theory framework of Lie group machine learning (LML). Computer Technology and Application, 2007, 3, 62–80.
- [2] Xu H, Li FZ. Lie Group Machine Learning's Axioms Hypotheses. Proceedings of 2006 IEEE International Conference on Granular Computing, 2006, 401–404.
- [3] Xu H, Li FZ. Study on Lie Group Machine Learning. Journal of Computational Information Systems, 2005, 1(4), 843–849.
- [4] Xu H, Li FZ. Geometry Algorithms of Dynkin Diagrams in Lie Group Machine Learning. Journal of Nanchang Institute of Technology, 2006, 25(2), 75–78.

- [5] Xu H, Li FZ. Algorithms of Dynkin Diagrams in Lie Group Machine Learning. *Journal of Communication and Computer*, 2006, 4(3), 13–17.
- [6] Xu H. Research on Machine Learning Model and Its Research. Suzhou: Soochow University, Thesis, 2007.
- [7] Xu H, Li FZ. The Design of SU(n) Classifier of Lie Group Machine learning(LML). *Journal of Computational Information Systems*, 2005, 1(4), 835–841.
- [8] Xu H, Li FZ. SO(3) Classifier of Lie Group Machine Learning. *Journal of Communication and Computer*, 2006, 4(3), 76–80.
- [9] Chen F, Li FZ. Research on Learning Algorithm of Subspace Orbit Generation in Lie group Machine Learning. *Computer Engineering and Applications*, 2007, 43(15), 184–187.
- [10] Chen F, Li FZ. Study on Learning Subspace orbit Generation Grid under the Action of GLn (Fq) Group in Lie group Machine Learning. *The Proceedings of the First Annual Conference of Young Scientists in Jiangsu Province*, 2006, 36–46.
- [11] Chen F. Research on Subspace Generation Algorithm of Lie group Machine Learning Subspace and Its Application. Suzhou: Soochow University, Thesis, 2007.
- [12] He SP. Research on Quantum Group Classifier in Lie Group Machine Learning. Suzhou: Soochow University, Thesis, 2008.
- [13] Fu HX. Study on the Symplectic Classifier in Machine Learning. Suzhou: Soochow University, Thesis, 2008.
- [14] Zhang J. Research on Learning Algorithm of Fiber Cong Based on Manifold Learning. Suzhou: Soochow University, Thesis, 2008.
- [15] Rao RPN, Ruderman DL. Learning Lie Groups for Invariant Visual Perception. *Advances in Neural Info Processing System*, 1999, 11, 810–816.
- [16] Xu XJ, Hou TJ. Computer aided drug design. Chemical Industry Press, 2004.
- [17] Burbidge R, Trotter M. Drug design by machine learning: support vector machines for pharmaceutical data analysis. *Computers and Chemistry*, 2001, 26, 5–14.
- [18] Yuan FX, Lu WC. Using support vector regression for QSAR study on the aldose reductase inhibitory activity of some flavones. *Computers and Applied Chemistry*, 2006, 6(24), 720–724.
- [19] Song XF, Han P. Optimal support vector machine and its application on modeling of HIV-1 protease inhibitors' QSAR. *Computers and Application Chemistry*, 2007, 11(24), 1475–1478.
- [20] Li JL, Wang ZZ. Predict siRNA activity based on back-propagation neural network. *ACTA BIOPHYSICA SINICA* [J], 2006, 12(6), 429–434.
- [21] Liu ZG, Zhao XH. Application of Genetic Algorithms on 2D-QSAR Analysis of Schiff Bases of Hydroxy semi-carbaside Ribonucleotide Reductase Inhibitors. *Journal of Human City University (Natural Science)*, 2007, 4(16), 52–56.
- [22] Wang JM, Hou TJ, Xu XJ. Conformational analysis of peptides using Monte Carlo simulations combined with the genetic algorithm. *Chemometer Intell. Lab*, 1999, 45, 347–351.
- [23] Ma ZQ. Group Theory in Physics. Beijing: Science Press, 1998.
- [24] Warner FW. Foundation of Differentiable Manifolds and Lie Groups. Springer-Verlag, 1983.
- [25] Huang XG. Basis of Lie group (Second edition). Fudan Press, 2007.
- [26] Yan ZD, Xu YC. Lie group and Lie algebra. Higher Education Press, 1985.
- [27] Michell TM. Machine Learning. China Machine Press, 2004.
- [28] Zhang L, Zhang B. A geometrical representation of McCulloch-Pitts neural model and its applications. *IEEE Transaction on Neural Networks*, 1999, 10(4), 925–929.
- [29] Wu T, Zhang L, Zhang YP. Kernel Overlay Algorithm in Machine Learning. *Chinese Journal of Computers*, 2005, 8(8), 1295–1301.
- [30] Wang SJ. Bionic Pattern Recognition (Topology pattern recognition)–Theory and Application of a New Pattern Recognition Model. *Chinese Journal of Electronics*, 2002, 30 (10), 1417–1420.

- [31] He Q, Shi ZZ. Research on classification algorithm based on coverage [J]. Chinese Journal of Computers, 2007, 8(30), 1235–1242.
- [32] He Q, Ren L, Shi ZZ. Direct Classification of Massive Data Based on Hypersurfaces. Chinese Journal of Computers, 2003, 26(2), 206–211.
- [33] Zhu YH. Application of Covering Clustering Algorithm. Computer Technology and Development, 2007, 1, 123–125.
- [34] Sutherland JJ, O'Brien LA, Weaver DF. A comparison of methods for modeling quantitative structure-activity relationships. J. Chem. Inf. Comp. Sci. [J], 2004, 47, 5541–5554.
- [35] Kuntz ID. Structure-based strategies for drug design and discovery [J]. Science, 1992, 257, 1078–1081.
- [36] Chen XM, Rao HB, et al. Prediction of Dihydrofolate Reductase Inhibitors Activity Using Machine Learning Methods. Chemical journal of Chinese universities, 2007, 28, 2171–2178.
- [37] Zhou ZH, Cao CG. Neural network and its application. Tsinghua University Press, 2004.
- [38] Jeffrey J, Donald S, Weaver BF. Three-dimensional quantitative structure- activity and structure-selectivity relationships of dihydrofolate reductase inhibitors. Journal of computer-aided molecular design, 2004, 18, 309–331.
- [39] Zsoldos Z, Reid D, et al. eHiTS: A new fast, exhaustive flexible ligand docking system. Journal of Molecular Graphics and Modeling, 2007, 26, 198–212.
- [40] Li CL. Research on Algorithm and Software of Molecular Docking Optimization Design in Drug Design. PhD thesis, Dalian University of Technology, 2004.
- [41] Böhm HJ. The computer program LUDI: a new method for the de novo design of enzyme inhibitors. J. Computer Aided Mol. Des., 1992, 6, 71–78.
- [42] Rarey M, Kramer B. A Fast Flexible Docking Method using an Incremental Construction Algorithm. J. Mol. Biol., 1996, 261, 470–489.
- [43] Kramer B, Rarey M, Lengauer T. Evaluation of the FlexX incremental construction algorithm for protein-ligand docking. Proteins, 1999, 2(37), 228–241.
- [44] Broughton HB. A method for including protein flexibility in protein-ligand docking: Improving tools for database mining and virtual screening [J]. Journal of Molecular Graphics and Modeling, 2000, 18, 247–257.
- [45] Jones G, Willett P, et al. Development and Validation of a Genetic Algorithm for Flexible Docking. J.Mol.Bio, 1997, 267, 727–748.
- [46] Miguel AS. Stabilization and attitude of a triaxial rigid body by Lie group methods [J]. Journal of Computational Physics, 2003, 191, 85–96.
- [47] Ding XL. Lie and Lie algebra analysis methods for spatial elastic deformation members [J]. Chinese Journal of Mechanical Engineering, 2005, (41), 16–23.
- [48] Wang LX, Zheng H, Lao XZ. Advances in Flexible Method of Protein Molecule [J]. Chinese Journal of Pharmaceutical Biotechnology, 2008, 4(15), 302–306.
- [49] Oshiro CM, Kuntz ID, Dixcon JS. Flexible ligand docking using a genetic algorithm [J]. Journal of Computer-Aided Molecular Design, 1995, 9, 113–130.
- [50] Makino S, Kayahara T, et al. Discovery of a novel serine protease inhibitor utilizing a structure-based and experimental selection of fragments technique. J Comp. Aided Mol. Des., 2001, 6, 553–559.
- [51] Guan WW, Li FZ. Drug molecular design using Lie Group Machine Learning (LML). 2008 International Conference on Advanced Intelligence (ICAI 08'), 2008, 411–414.
- [52] Guan WW. Research on the Covering Algorithm of Machine Learning and Its Application. Suzhou: Soochow University, Thesis, 2009.
- [53] Autodock. <http://www.scripps.edu/pub/olson-web/doc/autodock/>.
- [54] Gold. <http://www.ccdc.cam.ac.hk/prods/gold/>.
- [55] FlexX. <http://www.biosolveit.ed/FlexX/>.

7 Lie group deep structure learning

7.1 Introduction

Lie group machine learning and deep structure learning are new machine learning methods that have attracted considerable attention in recent years. This section introduces and analyses the research progress of deep structure learning.

Deep structure learning [14] refers to a learning model that contains multiple levels of non-linear operations, such as neural networks that contain multiple hidden layers, which consist of many non-linear layers that are non-linear functions and transformable functions. There is a certain connection between layers, which offers some significant advantages:

- (1) Deep structure learning can automatically extract an increasing number of abstract features. The low-level part extracts the specific features of the lower layer. After receiving these low-level features, the high-level part extracts the advanced abstract features from them and finally uses them for classification or understanding.
- (2) The deep structure has the characteristics of multi-layer nonlinearity, which can represent the highly nonlinear and complex functions needed by artificial intelligence tasks, such as computer vision. These functions finally map the input of the image pixel into the output of the category. A shallow structure does not necessarily have this ability.
- (3) In the same complex function, the deep structure is more efficient than the shallow structure, and it requires fewer calculation units and samples per layer.
- (4) Deep structure learning can be unsupervised or semi-supervised learning for discovering the characteristics of each layer. Thus, avoiding learning supervision may lead to fewer difficulties.

However, deep structure learning has not been investigated extensively. Hinton and others proposed a greedy unsupervised learning process [20]. This method uses the RBM learning rules to initialise the deep structure of the data. The generated model comprises numerous hidden variables, and it has some limitations. Specifically, it can only optimise local small-range data having a certain role. The proposed Bohr Man machine, which is a self-compiled, self-linking system, and other algorithms, can be applied to the shallow structure, but they are not effective when applied to the deep structure. Bengio et al. [14] used logic as an example to demonstrate that in the shallow structure, the number of logical gates required for the underlying structure when representing the same logical function increases exponentially. In addition, they also applied SVM (Support Vector Machine) directly to test and training samples between the core function values of a linear combination to achieve the classification decision. This approach is applicable only to the shallow structure. In the shallow structure, it is nec-

essary to represent a highly nonlinear function with a sharp change in shape, which needs to be exponentially related to the dimension of the sample distribution manifold. Therefore, the dimensionality problem still exists. Hugo, Bengio et al. used an unsupervised learning algorithm [21] to initialise the data in a deep network and demonstrated that using this algorithm to initialise the data can improve the induction. However, this algorithm is not applicable to large-scale data. Jason et al. proposed a simple method of deep structure learning [22]. This method is improved in the existing semi-supervisory law, combined with the embedded law and semi-supervised learning. It plays a key role in the analysis of data complexity, but when applied to semantics, its effect is not very obvious. In 2006, Osindero et al. proposed a deep Bayesian network [31] (DBN), which is a multi-layer network. The layers of the network are based on iterative training in RBMs, where the RBM is a two-tier network, and random pixels are associated with hidden random pixel detectors, which are highly effective in training samples. It raises the data by comparing the divergent processes, thus increasing the number of training data and reducing the performance of the model generation samples. The author used this method to train the analytical model of the deep adjacent elements in advance, and it can also be used to train the self-compiled model at the same time. S. T. Roweis et al. proposed a local linear embedding algorithm, i.e., the reduced dimensionality method (LLE) of manifold learning [50]. The main idea of this algorithm is that, first, it is assumed that the embedded low-dimensional manifolds are linearly distributed in the data set of the test samples, i.e., each sample point can be represented by several relatively close sample points in its neighbourhood. Then, the weight of each local neighbourhood is changed, and the low-dimensional representation of the high-order sample is obtained by the minimum reconstruction error. This algorithm is a geometrical structure of the manifold learning algorithm, with rotation, translation, and scaling. In 2000, on the basis of the multidimensional scale transform (MDS), Tenenbaum et al. proposed a new nonlinear dimensionality reduction method (ISOMAP) [51]. Here, multidimensional scaling means that in low-dimensional space, the distance between any two points after dimension reduction should be the same as their distance in high-dimensional space. To maintain the intrinsic geometric properties of the data points, the isometric distance uses the geodesic distance to replace the original continental distance. The algorithm is a method to maintain the global geometry of manifolds. It can effectively describe the nonlinear distribution structure of high-dimensional data, but the method is more suitable for the internal smooth flow pattern. Recently, Ranzato et al. introduced another method of deep structure learning, i.e., a coding/decoding structure called SESM [23]. They chose a compiler and a decoder with a common weight. The first encoder output is very close to the encoding Z. In the second case, the encoder output is used as the input encoding. The process for the third and fourth cases is the same as that for the previous two. This method of high-dimensional learning with a small amount of data is very fast, but it cannot effectively analyse the complexity of the data. In the output layer, the deep structures of the hidden layer and the auxiliary layer are embedded in the supervised learning

algorithm based on a linear semi-supervised Lie Group and Lie Group parameters according to the semi-supervised learning algorithm [24]. The deeper the structure, the lower the error rate and the stronger the effect; however, the time complexity is higher.

It is difficult to analyse the complex data of the image structure invariance as well as material properties, physical properties, and chemical properties of molecular structures by using existing computational techniques to complete the deep relationship of these data structures. Therefore, we introduce a deep structure to the learning algorithm to solve such problems.

7.2 Lie group deep structure learning

In this chapter, we propose a Lie group deep structure learning algorithm for data complexity and image structure invariance.

7.2.1 Deep structure learning

Definition 7.1. Deep structure refers to a learning model that contains multiple levels of non-linear operations, such as neural networks that contain multiple hidden layers. It consists of many non-linear layers. These layers are defined by some non-linear and transformable functions, and there is a connection between the layers.

The deep structure has several significant advantages:

- (1) It automatically extract layers of abstract features.
- (2) It represents a highly nonlinear complex function required for complex artificial intelligence tasks.
- (3) It effectively analyses the complexity of the data.

Definition 7.2. Deep structure learning is based on the basic content of the deep structure from the perspective of machine learning. The formation of a solution to the deep structure of the learning paradigm can automatically extract layers of abstract features. The low-level part extracts the specific features of the lower layer. After receiving these low-level features, the high-level part extracts the advanced abstract features from them, and finally uses them for classification or understanding. The characteristics of each layer are automatically discovered by the learning algorithm without manual intervention.

Consider computer vision as an example. The image content can be viewed from different levels of pixels, edges, corners, parts, objects, scenes, etc. Deep structure learning from low to high layers can automatically discover these increasingly abstract features and finally transform a complex problem into a simple problem through the layered process.

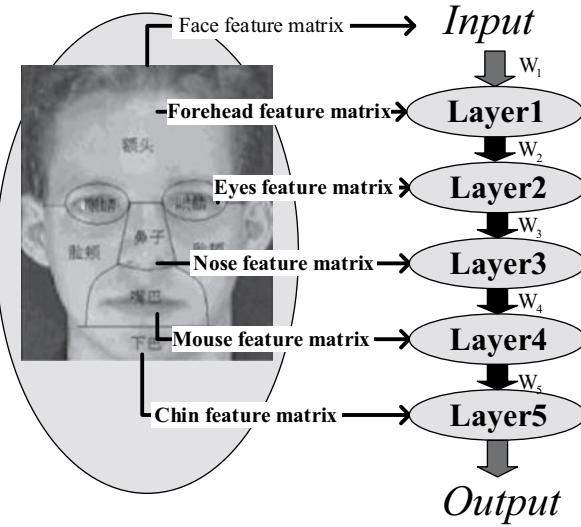


Fig. 7.1: The construction of deep structure.

From the complex circuit theory, we can see that the deep structure is more efficient than the shallow structure [14]. In the complex circuit, there are many different computing elements and parameters. These elements and parameters need to use functions for representation. However, they cannot be expressed in all the questions. Therefore, we consider the use of a deep structure. Then, a complex problem is broken down into n simple questions. These n simple questions correspond to $n!$ permutations and combinations, i.e., there are $n!$ rules to define the deep structure. In this paper, we choose one of the rules so that each layer corresponds to a complex problem decomposition of a simple problem; there is no combination. Then, we provide a valid index for each layer. Finally, complex problems can be reduced to simple problems through hierarchical processes.

For example, in face recognition, the top-down rules are chosen to construct the human face into a deep structure, followed by analysis. Figure 7.1 shows the face with a number of local areas. One of the local feature matrices is extracted as a layer of a deep structure. Then, a weight W between the two layers is provided, and W_i represents the relationship between the i -1-th layer and the i -th layer.

7.2.2 Construct deep structure model

This section constructs the deep structure model on the basis of the previous section.

Assume that the input value is x and $\hat{h}_j^i(x)$ represents the value of the j -th floor of the i layer. When $i = 0$, it represents the input layer. When $i = l + 1$, it represents the

input layer. Further, $|h^i(x)|$ represents the size of layer i , and b_j^i represents the inner offset of the i -th layer's j -th unit, which is the default activation code. W_{jk}^i represents the weight set between the $i-1$ -th layer $h_k^{i-1}(x)$ and the i -th layer $h_j^i(x)$. $\hat{h}_j^i(x)$ is defined as follows: $\hat{h}_j^i(x) = \text{sigm}(a_j^i)$, where

$$a_j^i(x) = b_j^i + \sum_k W_{jk}^i \hat{h}_k^{i-1}(x), \forall i \in \{1, \dots, l\}, \hat{h}^0(x) = x. \quad (7.1)$$

Here, $\text{sigm}(\bullet)$ is an S-shaped function: $\hat{h}_j^i(x) = \text{sigm}(a) = 1/(1 + e^{-a})$. The output layer is expressed as follows: $o(x) = \hat{h}^{l+1}(x) = f(a^{l+1}(x))$, where $a^{l+1}(x) = b^{l+1} + W^{l+1} \hat{h}^l(x)$.

A is a startup function for classification; it is defined as follows:

$$f_j(a) = \text{soft max}_j(a) = \frac{e^{a_j}}{\sum_{K=1}^K e^{a_k}} \quad (7.2)$$

where k represents k classes.

For the deep structure of the data, it is expressed as a manifold structure, so that each submanifold corresponds to a simple structure. Thus, a learning problem is decomposed into a number of simple questions. Each simple problem corresponds to a layer in the deep structure, and each layer provides a valid index on the submanifold (Fig. 7.2).

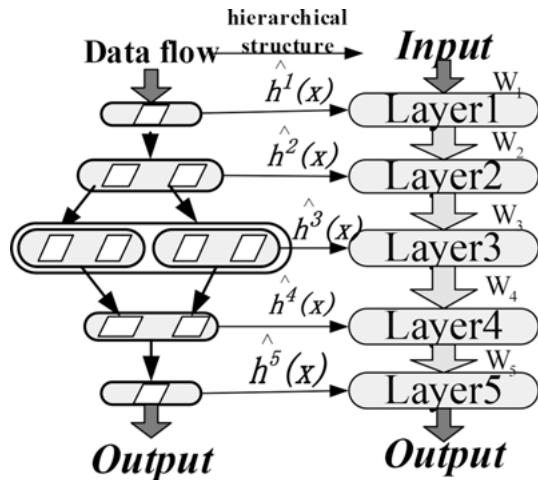


Fig. 7.2: Deep Structure.

7.2.3 Deep structure learning algorithm

The Siamese network is a typical method of non-linear embedding. Researchers in the field of neural networks argue that this network with the same function has the same weight and feeds back to the ‘distance measurement’ layer, which is used to calculate the similarity of two samples and to give the tag data.

The Siamese network also presents several loss functions, where we describe an edge loss:

$$L(f_i, f_j, W_{ij}) = \begin{cases} \|f_i - f_j\|^2 & W_{ij} = 1 \\ \max(0, m - \|f_i - f_j\|^2) & W_{ij} = 0 \end{cases}. \quad (7.3)$$

This avoids the appearance of similar matrices, and the distance difference between similar samples is at least m .

In daily life, we can often see some spherical S^2 and tread T^2 as well as other two-dimensional manifolds. From the mathematical point of view, S^2 and T^2 as submanifolds in R^3 are embedded in the three-dimensional space. In fact, each n -dimensional manifold as a topological space contains many different types of submanifolds, and this inclusion is called embedding. If the allowed graph contains a self-intersection, the inclusion is called an immersion. The definitions of immersion and embedding are given below.

Definition 7.3. Let M and N be C^r manifolds and $f: M \rightarrow N$ be a C^r mapping. Then, the definitions of immersion and embedding are as follows:

- (1) If for each point $p \in M$, the tangent mapping $Df: T_p M \rightarrow T_{f(p)} N$ is a single, i.e., its rank $Df = \dim$, then f is called a C^r immersion.
- (2) If $f: M \rightarrow N$ is a single, then (M, f) is called a C^r submerged submanifold of (N, f) .
- (3) If $f: M \rightarrow N$ is a single immersion, and $f: M \rightarrow f(M) \subset N$ is the dominant topology for N , i.e., the C^r -homomorphism, then f is called a C^r embedding and (M, f) is called the embedded manifold of C^r .

7.2.3.1 Deep bayesian network

The deep Bayesian network [31] (DBN) is a multi-layer network based on iterative training in RBMs. RBM is a two-layer network, and random pixel v is associated with a hidden random pixel detector h . Their relationship is as follows:

$$p(x) = \sum_{h \in H} P(x, h) = \sum_{h \in H} \frac{e^{-E(x, h)}}{Z}$$

$$E(x, h) = - \sum_{i \in \text{pixel}} w_i^P v_i - \sum_{j \in \text{features}} w_j^F h_j - \sum_{i,j} v_i h_j w_{ij}.$$

This idea has a great effect in training the sample. It raises the data by comparing the divergent processes so that the number of training data can be increased and the performance of the generated model can be reduced. The author used this method to

develop the analysis model of the deep adjacent elements in advance, and it can also be used to train the self-compiled model at the same time.

7.2.3.2 Encoding/decoding algorithm

Ranzato et al. introduced another method of deep learning, which is a coded/decoded structure [33], called SESM. They chose a compiler $f_{enc}(x) = w^T x + b_{enc}$ and a decoder $f_{dec}(z) = aS(z) + b_{dec}$ with a common weight.

The optimal loss function used in [33] is

$$a_e \|z - f_{enc}(x)\|_2^2 + \|x - f_{dec}(z)\|_2^2 + a_s h(z) + a_r \|w\|_1 .$$

In the first case, the output of the encoder is very close to the code z . In the second case, the encoder output is used as the input code. The process for the third and fourth cases is the same as that for the previous two. The weights of the encoders and decoders are w , and a_e , a_s , and a_r are super-elements. Because the algorithm is required to be optimal, in the training data, an online reconciliation program is needed. In fact, the embedding method can also be seen as a compiler A, embedded in low-dimensional space data. However, in the training process, it does not need to be decoded. Further, if the amount of high-dimensional data is small, the speed will be very high.

7.2.3.3 Locally linear embedding (LLE)

S. T. Roweis et al. proposed a local linear embedding algorithm, i.e., the reduced dimension method of manifold learning [50]. The main idea of this algorithm is as follows. First, it is assumed that the embedded low-dimensional manifolds are linearly distributed in the observed sample data, i.e., each sample point can be represented by a linear reconstruction of several relatively close sample points in its neighbourhood. Then, in the process of dimension reduction, the weights of each local neighbourhood are invariant, and the low-dimensional representation of the high-order sample is obtained by the minimum reconstruction error. The specific steps of the LLE algorithm are as follows:

- (1) Suppose that in the D -dimensional space, there are N data points on the same manifold, X_i is the i -th data point, $i = 1, 2, 3, \dots, N$. Here, it is assumed that the data points are sufficient because the local part of the manifold is linear and each sample point can be represented linearly by its neighbouring points. The nearest neighbour points of each data point need to be calculated.
- (2) Find the weight W_{ij} of each sample data point. We use the following formula to minimise the error between each data point and its nearest neighbour linear reconstruction

$$\epsilon(W) = \sum_i \left| X_i - \sum_j W_{ij} X_j \right|^2$$

Here, if X_j is not the neighbour of X_i , then the weight $W_{ij} = 0$ and $\sum_j W_{ij} = 1$. The cumulative value of each row of the weight matrix is 1.

- (3) To ensure that the weight remains unchanged, in the low-dimensional space, we can minimise the above formula. The original high-dimensional data points are reconstructed to obtain the low-dimensional representation:

$$\phi(Y) = \sum_i \left| Y_i - \sum_j W_{ij} Y_j \right|^2.$$

The local linear embedding (LLE) algorithm is a manifold learning method that preserves local geometry. It has some properties of rotation, translation, and scaling, and can learn the local linear low-dimensional manifolds of arbitrary dimension. This algorithm is simple, the computational complexity and time complexity are relatively low, and it can be attributed to the eigenvalue problem, which is simple and easy.

7.2.4 Lie group deep structure learning algorithm

In this section, the deep structure learning and Lie group learning algorithms are combined to give the Lie group deep structure learning algorithm, i.e., Lie group deep structure learning = Lie group machine learning + deep structure learning.

7.2.4.1 Deep structure semi-supervised learning algorithm

The deep structure semi-supervised learning algorithm refers to embedding the semi-supervised learning algorithm in the deep structure. Before embedding, we first set up a simple embedded model [22] and divide it into the following steps:

- 1 Choose an unsupervised learning method;
- 2 Select a model of a deep structure;
- 3 Unsupervised learning is inserted into every level of the architecture as an auxiliary task;
- 4 The same structure is used to train supervised and unsupervised tasks.

Deep learning involves learning a model with several layers of nonlinear mapping. Here, we will consider a multilayer network with M -layer hidden units and give a C -dimensional output vector:

$$f_i(x) = \sum_{j=1}^d w_j^{o,i} h_j^M(x) + b^{o,i}, i = 1, \dots, C \quad (7.4)$$

where w^o is the weight of the output layer. The k th layer is defined as

$$h_i^k(x) = S \left(\sum_j w_j^{k,i} h_j^{k-1}(x) + b^{k,i} \right), k > 1 \quad (7.5)$$

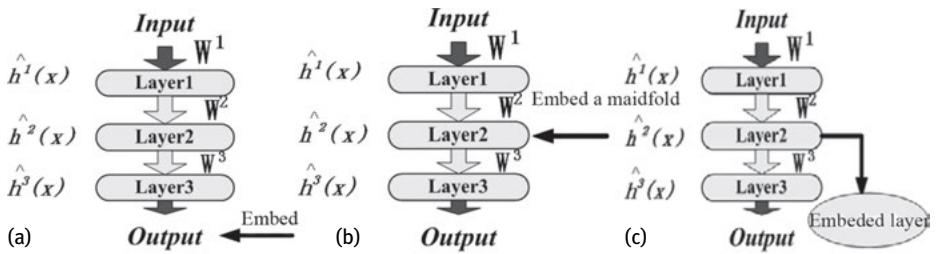


Fig. 7.3: Three kinds of embedding situation; (a) Embedded in the output layer, (b) Embedded in the hidden layer, (c) Embedded in the auxiliary layer.

$$h_i^1(x) = S \left(\sum_j w_j^{1,i} x_j + b^{1,i} \right) \quad (7.6)$$

where S is a deformation function similar to a hyperbolic tangent function. The most general way of semi-supervising deep learning is to embed a semi-supervisory rule in the deep structure (see Figure 7.3):

- (a) In the output section of the entire network, add a semi-supervised loss (regularisation) to supervise the loss (7.3):

$$\sum_{i=1}^L e(f(x_i), y_i) + \lambda \sum_{i,j=1}^{L+U} L(f(x_i), f(x_j), W_{ij}) . \quad (7.7)$$

This is similar to the simple technique described above, such as in (7.2).

- (b) Directly regularise the middle k th layer (7.5),

$$\sum_{i=1}^L e(f(x_i), y_i) + \lambda \sum_{i,j=1}^{L+U} L(f^k(x_i), f^k(x_j), W_{ij}) \quad (7.8)$$

$f^k(x) = (h_1^k(x), \dots, h_{N_k}^k(x))$ is the network output part of the k th layer.

- (c) An auxiliary network is generated by sharing the front k layers of the initial network, but there is a new weight set:

$$g_i(x) = \sum_j w_j^{AUX,i} h_j^k(x) + b^{AUX,i} . \quad (7.9)$$

We embed unlabelled data when we train this network, and we train the initial network on the tag data.

In the experiment, we embed the loss function (7.3), and a key loss is

$$e(f(x), y) = \sum_{c=1}^C H(y(c)f_c(x)) .$$

Jason et al. [22] embedded the semi-supervised and supervised algorithms in the output layer of the deep structure, one of the hidden layers, the auxiliary layer, and all other layers. Thus, they concluded that their approach to simplify complex tasks can bring greater benefits.

7.2.4.2 Lie group deep structure semi-supervised learning algorithm

Based on the idea of embedding the semi-supervised learning algorithm in the deep structure, this paper embeds the semi-supervised learning algorithm on the basis of the linear Lie group or the parameters on the basis of the parameters of the Lie group. Based on the unsupervised algorithm, in the output layer, hidden layer, and auxiliary layer, the semi-supervised learning algorithm is implemented.

The linear Lie group semi-supervised learning algorithm is described as follows. Sample $X = \{x_1, x_2, \dots, x_l, x_{l+1}, \dots, x_{l+u}\}$, where x_1, x_2, \dots, x_l is the marked data, i.e., the set $L = \{(x_1, y_1), (x_2, y_2), \dots, (x_l, y_l)\}$, x_{l+1}, \dots, x_{l+u} is the unmarked data ($l \ll u$). The output of the data x is y . The input sample is represented by a matrix, and the isomorphic linear Lie group of the corresponding learning system is found.

Take the marked data as the base point through the infinitesimal right to find its operator generation element:

$$X_\mu(x) = \varphi_\mu^v(x) \frac{\partial}{\partial x^\mu} = \left. \frac{\partial \phi^v(x, y)}{\partial y^\mu} \right|_{y=e} \frac{\partial}{\partial x^\nu} \quad (7.10)$$

Here, $\varphi_\mu^v = \frac{\partial \phi^v}{\partial y^\mu}|_{y=e}$ ($\mu, v = 1, 2, \dots, l$) is the transformation function of the Lie group, and ϕ^v is the analytic function, also known as the Lie group synthetic function [26].

The semi-supervised learning algorithm based on the parameter Lie group is described as follows: Sample $X = \{x_1, x_2, \dots, x_l, x_{l+1}, \dots, x_{l+u}\}$, where x_1, x_2, \dots, x_l is the marked data, i.e., the set $L = \{(x_1, y_1), (x_2, y_2), \dots, (x_l, y_l)\}$, x_{l+1}, \dots, x_{l+u} is the unmarked data ($l \ll u$). The output of the data x is y . The input sample is represented by the parameters, and the isomorphic parameters of the corresponding learning system are found. According to the algebraic structure of the parameter Lie group, the grouping method is used to mark the generator. Based on the marker data, the generator is defined by the definition of the parameters of the Lie group:

$$X_\mu = \lim_{a^\mu \rightarrow 0} \frac{(e + i \frac{a^\mu}{N} X_\mu)^N - e}{ia^\mu} \quad (7.11)$$

where a^μ ($\mu = 1, 2, \dots, r$) represents r consecutive parameters and N is a positive integer. Further, A is the generator of the calculation. The other unlabelled data are determined according to the group operation rule, and if the unlabelled data can be generated by the operator, the class is marked. The domain U_i ($1 < i < l$) information of each marked sample point X_i is analysed, and an inner product $g_{i,j}(a)$ whose volume is invariant is taken at the unit point in the sample set. We calculate the value of $g_{i,j}(a)$, $g_{i,j}(a) = \langle (\partial/\partial x_i), (\partial/\partial x_j) \rangle$, and then put the value of $g_{i,j}(a)$ into the following formula:

$$d(x_i, x_j) = \sqrt{\sum_{i,j=1}^n g_{i,j}(a)(x_{ii} - x_{ji})(x_{ij} - x_{jj})} \quad (7.12)$$

Calculate the distance between x_i and x_j in U_i ($1 < i < l$).

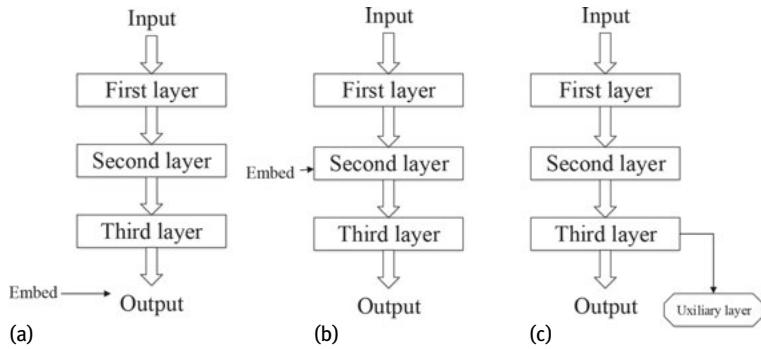


Fig. 7.4: Embedded Lie group semi-supervised algorithm; (a) Embedded in the output layer, (b) Embedded in the hidden layer, (c) Embedded in the auxiliary layer.

The structure of the semi-supervised learning algorithm for the deep structure of the Lie group is shown in Figure 7.4.

The main steps of this algorithm are as follows:

- (1) Use the unsupervised algorithm process to initialise the entire deep structure and then fine-tune the entire network;
- (2) On the basis of embedding the semi-supervised algorithm in each layer, the Lie group semi-supervisory algorithm SSLA-GL (n) or (SSLA-PLG) is embedded in the output layer, hidden layer, auxiliary layer, and all layers, respectively. The hidden layer is the hidden submanifold k , which creates a new helper layer by sharing the previous hidden layer.

In the deep structure embedded in the linear Lie group based on the semi-supervised algorithm, the basic steps are as follows:

- a) The semi-supervised algorithm SSLA-GL (n) based on the linear Lie group is embedded in the output layer and the semi-supervised loss is added to the supervisor

$$\sum_{i=1}^L e \left(\frac{\partial \phi^i(x, y)}{\partial y} \Big|_{y=e} \frac{\partial}{\partial x^i}, y^i \right) + \lambda \sum_{i=1}^{L+U} L(f(x^i), f(y^i), W_i) \quad (7.13)$$

where L denotes the number of tag data y_i and U denotes the unlabelled data.

- b) A semi-supervised algorithm based on the linear Lie group is embedded in the k -th sublayer

$$\sum_{i=1}^L e \left(\frac{\partial \phi^i(x, y)}{\partial y} \Big|_{y=e} \frac{\partial}{\partial x^i}, y^i \right) + \lambda \sum_{i=1}^{L+U} L(f^k(x^i), f^k(y^i), W_i) \quad (7.14)$$

where $f^k(x) = (X_1(x), \dots, X_N(x))$ represents the output portion of the k -th layer.

- c) The previous hidden layer is shared to form an auxiliary layer, and a new auxiliary right is generated. This auxiliary layer is embedded in the linear Lie group of the

semi-supervised algorithm.

$$\hat{g}_i(x) = \sum_j w_j^{\text{aux},i} \left. \frac{\partial \phi^j(x, y)}{\partial y} \right|_{y=e} \frac{\partial}{\partial x^j} + b^{\text{aux},i}. \quad (7.15)$$

In the deep structure embedded in the parameters based on the Lie group semi-supervised algorithm, the basic steps are as follows:

- a) In the output layer, the semi-supervised algorithm SSLA-PLG based on the parameter Lie group is embedded, and the semi-supervised loss is added to the supervisor.

$$\sum_{i=1}^L e \left(\lim_{a^i \rightarrow 0} \frac{(e + i \frac{a^i}{N} X_i)^N - e}{ia^i}, y^i \right) + \sum_{i=1}^{L+U} L(f(x^i), f(y^i)). \quad (7.16)$$

- b) The semi-supervised algorithm SSLA-PLG based on the parameter Lie group is embedded in the k -th hidden sublayer.

$$\sum_{i=1}^L e \left(\lim_{a^i \rightarrow 0} \frac{(e + i \frac{a^i}{N} X_i)^N - e}{ia^i}, y^i \right) + \lambda \sum_{i=1}^{L+U} L(f^k(x^i), f^k(y^i), W_i). \quad (7.17)$$

- c) The previous hidden layer is shared to form an auxiliary layer, and a new auxiliary right is generated. This auxiliary layer is embedded in the linear Lie group of the semi-supervised algorithm.

$$\hat{g}_i(x) = \sum_j w_j^{\text{aux},i} \lim_{a^i \rightarrow 0} \frac{(e + i \frac{a^i}{N} X_i)^N - e}{ia^i} + b^{\text{aux},i}. \quad (7.18)$$

In the experiment, this section uses the edge loss function in the Siamese network:

$$L(f_i, f_j, W_{ij}) = \begin{cases} \|f_i - f_j\|, & \text{if } W_{ij} = 1 \\ \max(0, m - \|f_i - f_j\|^2), & \text{if } W_{ij} = 0 \end{cases}, \quad m \text{ represents the maximum distance between similar examples.}$$

A key loss for the above formula is

$$e(f(x), y) = \sum_{c=1}^C \max(0, 1 - y(c)f_c(x)).$$

7.2.5 Experiment analysis

Based on the algorithm of deep structure of the Lie group, we carried out an experiment on the ORL face database. The experiment is divided into two parts:

- (1) The semi-supervised learning algorithm based on the linear Lie group and the semi-supervised learning algorithm based on the parameter Lie group are embedded in the output layer, the auxiliary layer, the middle layer, and all layers.
- (2) Respectively, perform the experiment in the layers 3, 4, 5, and 6 in the deep structure.

7.2.5.1 Parameter selection

The ORL face database contains 400 images of 40 people, each containing 10 images. The sample is normalised and the resolution of the image is 64×64 , forming a vector of 4096 dimensions. In this section, let $X = \{x_1, x_2, \dots, x_N\}$ be a set of training consisting of N samples; $x_i \in R^n$ represents the i -th image and $Y = \{y_i, i = 1, 2, \dots, Ny_i \in R^d\} (d < m)$ denotes the corresponding projection result. Take the weight W between each layer, which can be arbitrarily selected in the range $[-1, 1]$. The ORL face database (10 per person) is divided into five groups; one group is chosen as a training sample, and the remaining groups are used as a test sample.

7.2.5.2 Experimental results

The semi-supervised learning based algorithm on the linear Lie group and the semi-supervised learning algorithm based on the parametric Lie group are embedded in the output layer, middle layer, auxiliary layer, and all layers of the three-layer deep structure. The corresponding experimental results are shown in Table 7.1 and Table 7.2, respectively.

From Table 7.1 and Table 7.2, it can be seen that the Lie group semi-supervised learning algorithm works better in all the layers. Therefore, the following set of experiments was conducted in layers 3, 4, 5, and 6 of the deep structure of all layers embedded in the Lie group semi-supervised learning algorithm. The first group of samples was the training sample, and groups 2, 3, and 4 constituted the test sample. The specific experimental results are shown in Fig. 7.5.

Here, Embed_ALL_PLG means that the semi-supervised learning algorithm based on the parameter Lie group is embedded in all the layers of the deep structure; Embed_ALL_GL implies embedding a semi-supervised learning algorithm based on the

Tab. 7.1: A semi-supervised learning algorithm based on linear Lie group is embedded on the 3-layer deep structure.

	Group 1 training Group 2,3 testing	Group 2 training Group 3,4 testing	Group 3 training Group 2,4 testing	Group 4 training Group 1,3 testing	Average error rate [%]
Embedded in the output layer	20/160	23/160	24/160	20/160	13.59
Embedded in the middle layer	19/160	21/160	25/160	21/160	13.44
Embedded in the auxiliary layer	17/160	20/160	23/160	25/160	13.28
Embedded in all layers	18/160	22/160	19/160	18/160	12.03

Tab. 7.2: The semi-supervised learning algorithm based on the parameter Lie group is embedded on the 3-layer deep structure.

	Group 1 training Group 2,3 testing	Group 2 training Group 3,4 testing	Group 3 training Group 2,4 testing	Group 4 training Group 1,3 testing	Average error rate [%]
Embedded in the output layer	21/160	20/160	15/160	22/160	12.2
Embedded in the middle layer	24/160	19/160	17/160	23/160	12.97
Embedded in the auxiliary layer	23/160	20/160	24/160	26/160	14.53
Embedded in all layers	18/160	21/160	17/160	18/160	11.56

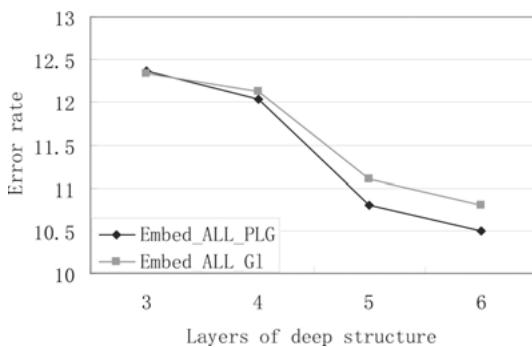


Fig. 7.5: Experimental results.

linear Lie group in all layers of the deep structure. From the experimental results, we can see that the results improve as the number of layers increases.

7.3 Lie group layered learning algorithm

The face space is the low manifold embedded in the high-dimensional observation space. To better detect superficial structures, such as protrusions and recesses in the face space, we combine the manifolds to study the Lie group deep structure learning in this section:

- (1) We use the singular value feature extraction method to extract the main features of the face;
- (2) The left and right matrices of SVD are decomposed into the Lie group;
- (3) The Lie group deep structure layered learning algorithm is implemented.

7.3.1 Singular value feature extraction

Face recognition technology is one of the main research topics in the field of computer pattern recognition, and it has a wide range of prospects. In identifying problems, extracting valid features and designing classifiers is the key to solving the problem. The face feature is the best way to describe the face pattern, and the dimensions of the feature can describe the face pattern from a different perspective. In the ideal case, the dimensions of the face feature are complementary. If not directly extracted by the feature of the face classification, reflecting the representative characteristics of the face is bound to weaken the generalisation of the classifier. When the dimensions of the sample space are high, the classification requires a considerable amount of computation, and the sample contains a large amount of information that is not of much use for the classification, with side effects. Therefore, it is necessary to select some of the most discriminant features from the sample by transforming, selecting, and learning before training the data. Feature extraction essentially seeks the most efficient features in numerous feature data while performing appropriate dimensionality compression [34]. Hong Z Q et al. proposed a face recognition method based on singular value vector features [47] and proved that the singular value vector has the properties of stability, translation, rotation, and mirror invariance. This algorithm is divided into the following three steps:

- (1) Take the whole of each image as a matrix;
- (2) Singular value decomposition of the matrix is carried out, and its singular value vector feature is extracted;
- (3) These features are projected onto the plane for classification and recognition.

Tian et al. [53] found that the singular value of the face contains only a small amount of effective information in the image, and more valid information is included in the two orthogonal matrices of the face image matrix after the singular value decomposition [53], which will lead to a lower face recognition rate. This chapter will deal with this problem; the left and right orthogonal matrices form a Lie group, i.e., the deep structure of learning.

In the process of training the data set, the data set is preprocessed. The preprocessing is performed to extract the data, and the feature sets that can be trained are extracted. Thus, the data training process is efficient; it reduces the time complexity and spatial complexity.

7.3.1.1 Two theorems of singular value decomposition

Theorem 7.1. Singular Value Decomposition Theorem of Matrix

Suppose that $A \in R^{m \times n}$ and $\text{rank}(A) = r \leq \min(m, n)$. Then, there are always orthogonal matrices $U \in R^{m \times m}$ and $V \in R^{n \times n}$ that make

$$A = U \sum_{i=1}^r \sigma_i u_i v_i^T.$$

Here, the symbol 'T' represents the transpose of the matrix

$$U = (u_1, u_2, \dots, u_m) \in R^{m \times m}$$

$$v = (v_1, v_2, \dots, v_n) \in R^{n \times n}$$

$$\sum = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \in R^{m \times n}$$

where $\Sigma_1 = \text{diag}(\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_r)$ and $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \dots \geq \sigma_r > \sigma_{r+1} = \sigma_{r+2} = \dots = \sigma_p = 0$, $p = \min(m, n)$. Here, B is a real number, i.e., the singular value of A . Further, $u_i, v_i (i = 1, 2, \dots, r)$ are the left and right singular vectors belonging to C of matrix A , respectively. All the singular values are arranged in ascending order. The vector $(\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_r)$ is the singular value vector of the matrix [48].

On the one hand, this theorem characterises the decomposition structure of the matrix more clearly. On the other hand, this theorem enables us to deal with the rank of a matrix reasonably. From the formula $A = \sum_{i=1}^r \sigma_i u_i v_i^T = U_r \sum_r V_r^T$, the matrix is actually obtained through the orthogonal changes that can express the rank and the number of ranks [49]. In the application of face image recognition, the rank of the matrix is a complicated issue because the acquisition of the basic data of the face image is affected by the expression, light, and attitude. However, we can obtain the effective rank of the image matrix by singular value decomposition of the matrix and judgment of the singular value. The following theorem illustrates the relationship between the matrix of some singular values and the original matrix.

Theorem 7.2. If the matrix $A \in R^{m \times n}$, set $m \leq n$ (m, n, r, s are positive integers). The matrix singular value decomposition theorem [53] is

$$A = UWV^T,$$

$W = \text{diag}(\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_r, \dots, \sigma_n)$. Let $W_s = \text{diag}(\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_s)$;

$$A_s = U \begin{bmatrix} W_s & 0 \\ 0 & 0 \end{bmatrix} V^T = \sum_{i=1}^s \sigma_i u_i v_i^T.$$

From the above formula, it can be concluded that $\text{rank}(A_s) = \text{rank}(W_s) = s$. Then,

$$\|A - A_s\|_F = \min \{ \|A - B\|_F \mid B \in R_s^{m \times n} \}$$

where $R_s^{m \times n}$ is the linear space of the $m \times n$ -order matrix with rank s .

Further, A_s is an approximation of matrix A in space $R_s^{m \times n}$. In this way, we can choose the best similarity matrix of A . Tian Y et al. [53] demonstrated that a large amount of decomposition information of the image is reflected in the left and right quadrature matrices after the image singular value decomposition.

Andrew Baker proved that any *qlinear* matrix is a Lie group [66]. Since any orthogonal matrix is a linear transformation of n -dimensional E-space $V_n(R)$, S and V decomposition of the U and V matrices gives a linear matrix; hence, U and V is a Lie Group.

7.3.1.2 Feature extraction

In the face recognition task, the training sample set is $\{X_i^j, i = 1, \dots, N, j = 1, \dots, K\} \in R^{m \times n}$, where i is the category, j is the number of face pictures for each person, $m \leq n$, and each sample is represented by a matrix, i.e., X_i^j is a matrix.

Step 1: Define the average face of all training samples, $M_i = 1/N \sum_{j=1}^n X_i^j$;

Step 2: The feature points of the average face are composed of the sequence $M(a_1, \dots, a_n)$, and the matrix M with the characteristic sequence as the row vector is obtained;

Step 3: The feature points extracted from the sample to be tested are composed of the sequence $T(a_1, \dots, a_n)$, and the corresponding matrix T with the feature as the row vector is obtained;

Step 4: The singular value decomposition of matrix M and matrix T is obtained, and the left and right singular value vector matrices U, V are obtained.

Step 5: Let $A_i = U_i / \sqrt[m]{|U_i|} \times V_i / \sqrt[n]{|V_i|}$, where m, n are the ranks of U_i, V_i , respectively, and A_i is a Lie group. The following gives the corresponding proof;

Step 6: Find the generator $A_\mu = \lim_{a^\mu \rightarrow 0} (e + ia^\mu/NA_\mu)^N - e/ia^\mu$ of Lie group A_i , where $a^\mu (\mu = 1, 2, \dots, r)$ represents r consecutive arguments and N is a positive integer.

The following proves that A_i is a Lie group.

Proof. Since U_i and V_i are $m \times m, n \times n$ matrices obtained by singular value decomposition, and the vectors inside them are orthogonal, U_i, V_i is an orthogonal matrix. Because the orthogonal matrix is a Lie group, $U_i / \sqrt[m]{|U_i|}$ and $V_i / \sqrt[n]{|V_i|}$ are orthogonal groups, and their Cartesian plot also constitutes a Lie group, i.e., $A_i = U_i / \sqrt[m]{|U_i|} \times V_i / \sqrt[n]{|V_i|}$ is a Lie group structure. \square

7.3.2 Layered learning algorithm

A complex AI task is usually composed of many sub-tasks, and there is a certain relationship between the sub-tasks. Hence, we can compile these sub-tasks into each layer of the deep structure and then study a deep structure algorithm to solve this problem. In other words, a complex problem is broken down into a number of simple questions.

Each simple problem corresponds to a layer, which constitutes a deep structure, and in the problem, there is a certain link between the deep structure of each layer, which are not mutually independent, but with numerous non-linear layers. On this basis, we can give a definition of learning by layer.

Definition 7.4. Layer-by-layer learning refers to the deep structure of top-down learning. First, learn the first layer. The first layer of the output value is the second layer of the input value, and so on. The i -th level output is the input sample of $i + 1$.

7.3.2.1 Lie group layered learning

In general, the collected images are represented by the Lie group structure. For greyscale images, the element values in the Lie group are the grey values of the image pixels. The current image feature extraction and recognition methods usually first obtain greyscale face image samples by matrix rows or columns straight for the vector. In this way, we can get a higher-dimension vector sample. The dimension of the sample is far greater than the sample dimension of the available image. In this section, we use the Lie group matrix to construct the singular value of the corresponding image and find the characteristic grey value of the image.

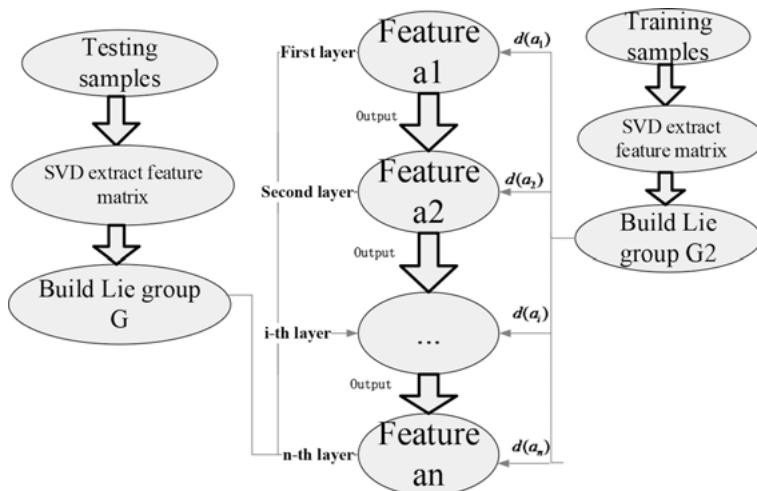


Fig. 7.6: Structural diagram of Lie Group Layered Learning.

Fig. 7.6 shows the structural diagram of the Lie group learning algorithm. The algorithm consists of the following parts:

- (1) The main features of the source image are extracted by the singular value decomposition method, and the deep structure based on the image eigenvalue is constructed according to the method mentioned in Chapter 3;

- (2) According to the method described in Section 7.3.1.2, the training samples and the test samples are decomposed by singular value decomposition to obtain the left and right orthogonal matrices;
- (3) Each layer of this deep structure corresponds to the Lie group generator A_u of the training sample and the test sample;
- (4) Learning from the top down, the first layer is learned for measured samples and training samples of the geodesic distance d , in order to retain the geodesic distance $d \leq \min$ for distance test samples;
- (5) According to the $n-1$ layer, to retain the test sample image, find the n th layer of training samples and test samples between the geodesic distance d_n . The result of the last layer is the result of the required test, and the output of layer i is the input sample of layer $i + 1$.

The detailed description of the deep layer structure learning algorithm of the Lie group is as follows:

Input: Training sample composition of Lie group G

Output: Selected test samples Y

- (1) Construct the Lie group deep structure
- (2) Find the distance between the first layer of the training sample and the test sample

$$d_1 = \|\log(X_1^{-1}X_1^*)\|_F$$

Here, the F-paradigm of matrix A_F is $\|A_{ij}\|_F = \sqrt{\sum_{i,j=1}^n |a_{ij}|^2}$, where n is the dimension of the matrix;

- (3) Learning from top to bottom, learn the first layer, seek samples and training samples of the geodesic, and let those with geodesic distance $d_1 \leq \min$ be test samples. The output data of the first layer is the input sample of the second layer.
- (4) By iteration, according to the $n-1$ layer to retain the test sample image, find the n -level training samples and test samples between the geodesic d_n . The result of the last layer is the desired result, and the output of the i -th layer is the input of the $i + 1$ th layer.
- (5) Output the test data corresponding to the last layer of data.

7.3.3 Experiment and analysis

Experiments were performed using the ORL International Standard Face Mask. The ORL database includes a total of 40 individuals with 400 images, i.e., each person has 10 images, with gestures, facial details, and facial changes. When the sample is selected, the resolution of the image is 64×64 , forming a 4096-dimension vector. In this experiment, 10 images per person were divided into 4 groups; each group had

one, two, three, and four images respectively, and two groups were selected as training samples, while the remaining two groups were used as test samples.

7.3.3.1 Experiment

The experiment is divided into two parts:

- (1) The algorithm is applied to the three-layer deep structure, and 3, 4, 5, 6 images are taken as the training samples for each of the 10 images. It can be compared with the singular value decomposition of the face recognition method;
- (2) Experiments were performed on the deep structure of layers 3, 4, 5, and 6, and the number of training samples was 3.

The experimental results are shown in Table 7.3

Tab. 7.3: Comparison of Two Algorithms.

Training samples	Group 1,2 training	Group 1,3 training	Group 1,4 training	Group 2,4 training	Group 3,4 training	Average recognition rate
	Group 3,4 testing	Group 2,4 testing	Group 2,3 testing	Group 1,3 testing	Group 1,2 testing	
SVD	60/280	59/240	62/200	42/160	34/120	26.32%
Step_DP	230/280	203/240	175/200	141/160	107/120	86.3%

The Lie group deep structure layered learning algorithm is tested respectively in 3, 4, 5, and 6 layers of the deep structure. The training sample comprises the first two groups; the test sample comprises the remaining two groups. The specific experimental results are shown in Table 7.4.

Tab. 7.4: Experimental results.

Number of layers	3	4	5	6	Average recognition rate
Step_DP	230/280	237/280	241/280	245/280	85.09%

7.3.3.2 Experimental analysis

From the experimental results, we find that the accuracy of face recognition in the Lieong learning algorithm is better than that of singular value decomposition. As can be seen from Table 7.5, the deeper the depth, the higher is the accuracy of recognition. The reason can be attributed to the fact that when the singularity feature is used for identification, the extracted singularity eigenvector contains only a small amount of effective information of the face image. Singular value decomposition of the more

effective information occurs in the left and right singular value vector matrix U, V . Therefore, the Lie group layered learning algorithm, through the U, V structure of Lie group, learns by layer to improve the accuracy of identification.

7.4 Lie group deep structure heuristic learning

The heuristic approach uses a method that facilitates the problem itself to guide the learning process. In the field of artificial intelligence, diagnosis, machine learning, automatic reasoning, and others, the heuristic search algorithm has been widely used. In this paper, based on the third section, and combined with the idea of the heuristic search algorithm, Lie group deep structure heuristic learning is proposed.

7.4.1 Heuristic learning algorithm

In artificial intelligence, a large number of problems in the solution technology depend on the search. The heuristic approach uses the problem itself to guide the information and the search process. In this learning process, the selection of the heuristic function is crucial, and the efficiency and success of the whole algorithm are determined. Heuristic search is usually used for two different types of problems:

- (1) Forward inference
- (2) Backward inference

Forward inference is generally used for a state space search. In forward inference, inference is performed from the predefined initial state to the target state in the opposite direction. Backward inference is generally used for problem reduction. In backward inference, inference is performed from the given target state to the initial state.

The function used to evaluate the importance of a node is called an evaluation function. The evaluation function $f(x)$ is defined as the estimated value of the minimum path cost of all paths from the initial node S_0 subjectively to the target node S_g via the node x .

The general form is

$$f(x) = g(x) + h(x)$$

where $g(x)$ represents the actual cost from the initial node S_0 to the node x , and $h(x)$ represents the cost of the optimal path from x to the target node S_g . It embodies the heuristic information of the problem, which is determined by the characteristics of the problem, and $h(x)$ is called a heuristic function [38]. Thus, the heuristic approach translates the description of the problem state into a description of the degree of problem solving, which is represented by the value of the evaluation function.

7.4.2 A* algorithm

The evaluation function f makes it possible to estimate the sum of the minimum cost paths from node S to node n on any node with function value $f(n)$, i.e., $f(n)$ is an estimate of the cost of restraining a minimum cost path through node n . Hence, the node with the smallest value on the table (OPEN) of the extended node is the estimated node with the least stringent constraint, and it is appropriate to extend the node to the next step.

Before introducing the A algorithm, we first describe a few commonly used symbols. $k(n_i, n_j)$ represents the actual cost of the minimum cost between the i -th node and the j -th node, and if there is no path between the two nodes, the function k does not make any sense. $h^*(n)$ represents the smallest of the actual cost of all $k(n, t_i)$ on the entire set $\{t_i\}$ of target nodes. Hence, $h^*(n)$ represents the cost of n to the minimum cost path of the target node. In this way, any path on $h^*(n)$ is the best path for n to the target node.

The actual cost of an optimal path from the starting node S to any node n is $k(S, n)$. To simplify the symbol and introduce a new function g^* , for the starting node S to any node n , the function g^* can be defined as

$$g^*(n) = k(S, n). \quad (7.19)$$

The function f^* is defined so that the function value $f^*(n)$ of any node n represents the sum of the actual algebra from the starting node S to an optimal path of any node and the optimal path $h^*(n)$ from the node n to the target node.

$$f^*(n) = g^*(n) + h^*(n). \quad (7.20)$$

Here, g is the estimate of g^* , and h is the estimate of h^* . $g(n)$ represents the minimum cost path that adds up from all the arcs of the starting node S to any node n , and it can be seen that $g(n) \geq g^*(n)$. $h(n)$ can be regarded as an estimated function of $h^*(n)$, relying on the heuristic information about the problem domain, so that $h(n)$ is the heuristic function.

In the A algorithm, if $h(x) \leq h^*(x)$ exists for all x , then $h(x)$ is called the lower bound of $h^*(x)$. If the lower bound $h(x)$ of $h^*(x)$ is the A algorithm of the heuristic function, the algorithm is called the A^* algorithm. When $h = 0$, the A^* algorithm [38] becomes an ordered search algorithm, which is mainly based on the definition of the evaluation function. For the general ordered search algorithm, we always select the value of the smallest node f as a node extension. Thus, f is based on the demand to find a minimum cost path to estimate the point of view; each node n has two evaluation functions on average:

- (1) The estimated value from the starting node S to the node n ;
- (2) The estimated value from node n to target node M .

In addition, there are other heuristic learning algorithms. Ramalingam et al. proposed an incremental algorithm for solving dynamic SWSF fixed point problems, DynamicSW SF-FP [54]. Koenig et al. proposed the *LPA** algorithm, which incorporates the A* algorithm on the basis of the DynamicSW SF-FP algorithm [55, 56]. The *D* Lite* algorithm [57] is a combination of the *D** and *LPA** algorithms in robot navigation problems. The *RTAA** algorithm [58] can adjust the information in real time to improve the search efficiency.

7.4.3 Lie group deep structure heuristic learning algorithm

Lie group theory is one of the representatives of the spatial transformation criterion. The three central elements of the Lie group [26] are Lie distance, specific mean value, and geodesic. The Lie distance is used to test two similar transformations. The specific average represents an ‘average’ of the transform set, which can reduce the distance between all transformations. The geodesic is a one-dimensional transformed subspace that is the shortest path between two transformations. A standard geodesic can be used to compute the maximum set of variables on this path to form a standard covariance matrix. Fletcher et al. [41] used the Lie group theory to test the shape of the plume, and the experimental data were human organs, such as human kidneys, which are three-dimensional structures, denoted as M-rep. In the Lie group manifold, the minimum distance curve (geodesic) represents the variability of the organ shape itself. Klassen et al. [42] used geodesic paths to spread the entire shape space and demonstrated that in one category, this method could produce a series of image prototypes. In the transformation space, according to their shape, we perform clustering using the Lie distance. Tuzel et al. [43] used the linear feature of Lie group to perform an averaging-transfer algorithm for clustering transformations on the tangent plane of the Lie group. Tuzel et al. [44] also used the Lie group to establish covariance descriptors to detect people’s scenarios in videos. Huber et al. [45] have given examples of three-dimensional vehicles.

The Lie group framework plays a key role in recognising the category of category objects, such as faces. This category is defined by the transformation of the face shape, and the similarity between the faces can be measured by the distance. Ferryman J et al. [46] demonstrated that the Lie distance can effectively identify similar images.

A new selection criterion is designed for the heuristic learning algorithm of the deep structure proposed in this section, namely the difference measure:

- (1) According to the layer-by-layer learning algorithm proposed in Section 7.3, we choose two different levels. Assume that the number of training samples in the i -th layer is N_i , and the number of misclassifications is N_{0i} . Then, the difference measure is

$$\begin{cases} I_{i+1} = d_{i+1} + W_i \times \frac{N_{0i}}{N_i}, & i \neq 0 \\ I_1 = d_1, & i = 0. \end{cases} \quad (7.21)$$

- (2) The deep structure heuristic learning algorithm is described as follows:

Input: Training sample composition of Lie group G .

Output: Selected test samples Y .

- (a) Construct the Lie group deep structure by the heuristic method.
- (b) Find the distance between the first layer of the training sample and the test sample

$$d_1 = \|\log(X_1^{-1} X_1^*)\|_F$$

where n is the dimension of the matrix;

- (c) Learning from the top down to learn the first layer, find the degree of difference I_1 and keep the difference $I_1 \leq \min$ test sample as the output of the first layer. The first layer of the output sample is the second layer of the input sample. In this way, the difference degree A of the i -th layer is deduced, the sample with a smaller difference is retained, and the output sample of the i -th layer is taken as the input sample of the $i+1$ -th layer.
- (d) Output the test data corresponding to the last layer of data.

7.4.4 Experiment and analysis

7.4.4.1 Experiment

This experiment was conducted on the international standard ORL database. The ORL database consists of a total of 40 individuals and 400 images, i.e., each person has 10 64×64 different images. These 10 images are with differences in light, expressions (do not laugh or laugh, eyes closed or eyes open), gestures, and so on. Each of the 10 images was divided into 4 groups, each having 1, 2, 3, and 4 images.

The experiment in this section is divided into two parts:

- (1) In the three layers of the deep structure of the experiment, each of the 10 images were extracted with 3, 4, 5, 6, and 7 images as a training sample;
- (2) The experiment was conducted on the deep structure of layers 3, 4, 5, and 6, respectively, and four images were taken as training samples.

The experimental results are shown in Table 7.5.

Tab. 7.5: Experimental results of different training samples.

Training samples	Group 1,2 training	Group 1,3 training	Group 1,4 training	Group 2,4 training	Group 3,4 training	Average recognition rate
Heur_DP	232/280	204/240	171/200	139/160	107/120	85.87%
Group 3,4 testing	Group 2,4 testing	Group 2,3 testing	Group 1,3 testing	Group 1,2 testing		

Tab. 7.6: Experimental results.

Number of layers	3	4	5	6	Average recognition rate
Heur_DP	204/240	209/240	213/240	217/240	87.81%

In layers 3, 4, 5, and 6 in the deep structure of the experiment, the first and third groups are training samples, and the remaining two groups are test samples. The experimental results are summarised in Table 7.6

7.4.4.2 Analysis of results

Table 7.5 shows the three layers of the deep structure of the experiment. The training samples were 3, 4, 5, 6, and 7 images. The results can be seen from the experiment. The higher the accuracy rate increases with the number of samples. Table 7.6 shows 3, 4, 5, and 6 layers in the deep structure of the experiment. The number of training samples is three. From the experimental results, it can be seen that the greater the number of layers of the structure, the higher is the accuracy rate. The rate of accuracy is high mainly because the amount of data in the test sample is very large, resulting in effective classification. Stratification can be layered through screening of the layers to reduce the amount of data to be tested, thereby improving the accuracy of classification.

7.5 Summary

The deep structure is mainly studied for the analysis of complex problems in real life, such as the analysis of invariance of image structures as well as the analysis of material, physical, and chemical properties of molecular structures. The Lie group is a natural combination of geometrical and algebraic structures, and it is a useful tool for studying complex data and inter-data transformations. Therefore, this chapter introduced some theories and commonly used algorithms of the Lie group into the deep structure, analysed and expressed the data using the good geometric and algebraic structures of the Lie group, and presented some algorithms for the deep structure learning of the Lie group. This chapter mainly includes the following aspects:

- (1) Analyses the research status of deep structure learning;
- (2) Gives the construction of the deep structure of the process to break down a complex problem into a number of simple questions and to establish an effective link between each simple question. Thus, for the complex data, with deep structure analysis, each layer corresponds to a complex problem decomposition of a simple problem, and each layer has a valid index. Hence, the complexity of the analysis of data is relatively low;

- (3) In this paper, we presented the algorithm of semi-supervised learning in the Lie group, Lie group learning algorithm, and Lie group heuristic learning algorithm. Our method was applied to the ORL standard face database and YALEB face database for identification, and the accuracy of recognition was given.

Bibliography

- [1] Arnold DG, Sturte K, Velten V, Nadhakumar N. Dominant–Subspace Invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000, 22(7), 649–662.
- [2] Rao RPN, Ruderman DL. Learning Lie Groups for Invariant Visual Perception. *Advances in Neural Info Processing Systems*, 1999(11), 810–816.
- [3] Nishimari Y, Akaho S. Learning Algorithms Utilizing Quasi-geodesic Flows on The Stiefel Manifold. *Neural Computing*, 2005, 67, 106–135.
- [4] Tsao TR, Shyn HJ, Libert JM, Chen VC. A Lie group approach to a neural system for three dimensional interpretation of visual motion. *IEEE Transaction On Neural Networks*, 1991, 2(1), 149–155.
- [5] Squire DM, Signatures I. Characterizing Contours by Their Departures from Invariance. *Computer Vision and Image Understanding*, 2000, 77, 284–316.
- [6] Fiori S. Unsupervised Neural Learning on Lie Group. *International Journal of Neural System*, 2002, 12(13), 1–28.
- [7] Mansouri AR, Mukherjee DP, Action ST. Constraining Active Contour Evolution via Lie Groups of Transformation. *IEEE Transactions on Image Processing*, 2004, 13(6), 853–863.
- [8] Govindu VM. Lie–Algebraic Averaging For Globally Consistent Motion Estimation. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004, 1, 1684–1691.
- [9] Kantor GA, Krishnaprasad PS. An Application of Lie Groups in Distributed Control Networks. Preprint submitted to Elsevier Preprint, 2000.
- [10] Lebanon G. Metric Learning for Text Documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006, 28(4), 1–12.
- [11] Kondor R. Group theoretical methods in machine learning. Columbia University, 2008, 5.
- [12] Li FZ, Qian XP, Xie L, He SP. Machine learning theory and application. Press of University of Science and Technology of China, 2009.
- [13] Meng DJ, Bai CM. Lie Group First edition. Beijing: Science Press, 2003.
- [14] Bengio Y. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2009.
- [15] Hinton GE. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 2002, 14, 1771–1800.
- [16] Hinton GE. Connectionist learning procedures. *Artificial Intelligence*, 1989, 40, 185–234.
- [17] Hinton GE. To recognize shapes, first learn to generate images. Technical Report UTML TR 2006–2003, University of Toronto, 2006.
- [18] Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. *Science*, 2006, 313(5786), 504–507.
- [19] Hinton GE, Dayan P, Frey BJ, Neal RM. The wake-sleep algorithm for unsupervised neural networks. *Science*, 1995, 268, 1558–1161.
- [20] Hinton GE, Osindero S, The YW. A fast learning algorithm for deep belief nets. *Neural Computation*, 2006, 18, 1527–1554.

- [21] Larochelle H, Bengio Y, Jerome and Lamblin P. Exploring Strategies for Training Deep Neural Networks. *Journal of Machine Learning Research*, 2009, 1–40.
- [22] Weston J, Ratle F, Collobert R. Deep Learning via Semi-Supervised Embedding, 2008, 1168–1175.
- [23] Ranzato MA, Boureau FJ, Lecun Y. Unsupervised learning of invariant feature hierarchies with applications to object recognition. Conference on computer vision and pattern recognition, 2007, 1–8.
- [24] He WH, Li FZ. Research on Lie Group Deep Structure Learning Algorithm. *Journal of Frontiers of Computer Science*, 2010, 7(3), 646–653.
- [25] Chen WH. Introduction of Differential manifold Second edition. Beijing: Science Press, 2001.
- [26] Shao D, Shao L, Guo Z. Lie Group First edition. Beijing: Science Press, 2008.
- [27] Network Learning Resources of Lie Group <http://www.maths.dept.shef.ac.uk/magic/corse.php?id=43>.
- [28] Tian C. Lie Group and Its Application in Differential Equations First edition. Beijing: Science Press, 2001.
- [29] Zhang DM, Zhong ZC. Application of group theory First edition. Wu Han: Huazhong University of Science and Technology Press, 2001.
- [30] Gao CS. Group Theory and Its Application in Particle Physics First edition. Beijing: Higher Education Press, 1992.
- [31] Hinton GE, Osindero S, The YW. A fast learning algorithm for deep belief nets. *Neural Computation*, 2006, 18, 1527–1554.
- [32] Bengio Y, Lamblin P, Popovici D, Larochelle H. Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, NIPS19, 2007.
- [33] Ranzato M, Huang F, Boureau Y, LeCun Y. Unsupervised learning of invariant feature hierarchies with applications to object recognition. Proc. Computer Vision and pattern Recognition Conference (CVPR'07). IEEE Press, 2007.
- [34] Bian ZQ, Zhang XG. Pattern Recognition (Second edition). Beijing: Tsinghua University Press, 2000, 32–85.
- [35] Zhan XZ. Matrix Theory. Higher Education Press, 2010.
- [36] Pradhan SS, Ward W, Hacioglu K, Martin J, Jurafsky D. Shallow semantic parsing using support vector machines.in North American, Proceedings of HLT/NAACL, 2004.
- [37] Ronan C, Jason W. Fast semantic extraction using a novel neural network architecture. Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, 2007, 560–567.
- [38] Cai ZX, Xu GY. Artificial Intelligence and Its Application. Beijing: Tsinghua University Press, 2003.
- [39] Zhang P, Cui WL. Heuristic Algorithm for Optimizing Neural Network Structure Based on Rough Set. *Control engineering*, 2009, 16(1).
- [40] Zhang WX, Qiu GF. Uncertain decision based on rough set. Beijing: Tsinghua University Press, 2005, 60–120.
- [41] Fletcher T, Lu C, Joshi S. Statistics of Shape via Principal Geodesic Analysis on Lie Groups [C]. CVPR 2003.
- [42] Klassen E, Srivastava A, Mio W, Joshi S. Analysis of Planar Shapes Using Geodesic Paths on Shape Spaces. PAMI, 2004.
- [43] Tuzel O, SubbaRao R, Meer P. Nonlinear Mean Shift for clustering over Analytic Manifolds. CVPR, 2006.
- [44] Tuzel O, Porikli F, Meer P. Human Detection via Classification on Riemannian Manifolds. CVPR, 2007.

- [45] Huber D, Kupuria A, Donamukkala R, Hebert M. Parts-based 3-D object classification. *CVPR*, 2004.
- [46] Ferryman J, Worrall A, Manbank S. Learning enhanced 3-D Models for Vehicle. *BMVC*, 1998.
- [47] Hong ZQ. Algebraic feature extraction of images for recognition. *Pattern Recognition*, 1991, 24(3), 211–219.
- [48] Ran RS, Huang TZ. Color Face Recognition Based on Singular Value Feature Extraction. *Application Research of Computers*, 2007, 24 (7), 298–302.
- [49] Sun JJ, Zhang HF, Sun C. A New Method of Face Recognition Based on Singular Value Decomposition. *Science Technology and Engineering*, 2010, 10(25) 6204–6208.
- [50] Roweis S, Saul LK. Nonlinear dimensional reduction by locally linear embedding. *Science*, December 2000, Vol. 290.
- [51] Tenenbaum JB, Sliva VD, Langford JC. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, December 2000, Vol. 290.
- [52] Belkin M, Niyogi P. Laplacian Eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems*, 2001, Vol. 15.
- [53] Tian Y, Tan TN, Wang YH. Do singular values contain adequate information for recognition. *Pattern Recognition*, 2003, 3(6), 649–655.
- [54] Ramalingam G, Reps TW. An Incremental Algorithm for a Generalization of the Shortest-path Problem [J]. *Journal of Algorithms* 1669, 21(2), 267–305.
- [55] Koenig S, Likhachev M, Furcy D. Lifelong Planning A*. *Artificial Intelligence*, 2004, 155(1/2):93–146.
- [56] Koenig S, Likhachev M. D* Lite. *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Application of Artificial Intelligence* Menlo Park AAAI Press, 2002, 476–483.
- [57] Koenig S, Likhachev M. Real-time Adaptive A*. 5th International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2006). New York ACM, 2006, 281–288.
- [58] Mills-Tettey GA, Stentz A, Dias MB. DD* Lite Efficient Incremental Search with State Dominance. Tech Rep, Robotics Institute. Pennsylvanian: Carnegie Mellon University, 2007, 1032–1038.
- [59] Xu H, Li FZ. Study on Lie Group Machine Learning, *Journal of Computational Information Systems*, 2005, 1(4), 843–849.
- [60] Li FZ, Kang Y. The Theoretical Framework of Machine Learning Based on Lie Group [J]. *Journal of Yunnan University of Nationalities*, 2004, 13(4), 251–255.
- [61] Chen F, Li FZ. Research on Learning Subspace orbit Generation Grid Algorithm of Machine Learning (LML), *Computer Engineering and Applications*, 2007, 43(15), 184–187.
- [62] He SP. Research on Quantum Group Classifier in Lie Group Machine Learning. Suzhou: Soochow University, Thesis, 2008.
- [63] Fu HX. Study on Symplectic Classifier in Machine Learning. Suzhou: Soochow University, Thesis, 2008.
- [64] Zhang J. Research on Learning Algorithm of Fiber Cong Based on Manifold Learning [D]. Suzhou: Soochow University, Thesis, 2008.
- [65] Zhang J, Li FZ. Research on fiber bundle learning model based on popular learning [J]. *Journal of Nanjing University*, 2008, 44(5), 477–485.
- [66] Baker A. Matrix Groups An Introduction to Lie Group Theory [M]. Tsinghua University Press, 2009.

8 Lie group semi-supervised learning

8.1 Introduction

Semi-supervised learning and Lie group machine learning are new machine learning methods that have attracted considerable attention in recent years. The research progress in semi-supervised learning is analysed in this section. With regard to semi-supervised learning, we mainly introduce five currently popular algorithms: self-training algorithm, EM algorithm with the hybrid model, co-training (collaborative training) algorithm, transductive support vector machine algorithm (TSVM), and graph-based algorithm.

8.1.1 Research status of semi-supervised learning

The use of a large number of unlabelled examples not only leads to the construction of a better classifier but also reduces the manpower and material resources required, thereby improving the performance of semi-supervised machine learning, which has attracted considerable research interest in theory and practice, as one of the most important topics in current machine learning research [1, 2].

Semi-supervised learning methods include the EM algorithm with the hybrid model, self-training algorithm, co-training algorithm, transductive support vector machine (TSVM) algorithm (data-dense regions are avoided as the decision boundary), and the graph-based method [2]. Among these algorithms, a suitable method is chosen as follows. If the data can form a good clustering, we choose the EM algorithm with the hybrid model. If the features can be decomposed into two subsets, then we select the co-training algorithm. If there are two points with similar characteristics in the same class, we use the graph-based method. In the case of SVM, the semi-supervised TSVM is a natural extension. If a complex supervised classifier already exists, we use the self-training algorithm [3–8].

8.1.1.1 Self-training algorithm

Self-training is one of the earliest research methods in semi-supervised learning, and it is also one of the simplest algorithms. The flowchart of the algorithm is shown in Algorithm 8.1.

Self-training assumes that high-confidence predictions are correct. A small amount of labelled data is first used to train a classifier in the self-training algorithm. Then, the classifier is used to classify the unlabelled data. In general, the unlabelled samples with the highest confidence are selected and put together with the prediction marks into the training set. After completion of the above steps, the classifier is retrained and the process is iterated.

Algorithm 8.1: Self-training algorithm processing

Input: feature set V ; training has been marked sample set L ; unmarked sample set U ; Self-training cycle number T , sample confidence threshold T_w

For $t = 1, 2, \dots, T$

- (1) Using the annotation sample set L to train the classifier C on the feature set V ;
- (2) Use the classifier C to predict all unlabeled samples $u \in U$, calculate their predictive confidence, determine whether T_w is greater than T_w , if less than T_w , discard the forecast;
- (3) Add a higher confidence sample to the trained marker set L .

End for

Output: classification result and final classifier C .

Note that the classifier uses its own predictions to learn; hence, this process is called self-teaching or bootstrapping. To avoid a classifier in order to eliminate its errors, some algorithms try to use the method in which unmarked points are not learnt if the confidence is lower than a threshold value. In general, EM algorithms with the generation of mixed models can be seen as a special case of soft self-training. Self-training has been used for real-world natural language processing tasks. Yarowsky (1995) used self-training to disambiguate word meanings, e.g., the determination of whether the word ‘plant’ refers to an ‘organism’ or a ‘factory’ in a given context [9].

The advantage of self-training lies in its simplicity. It is a wrapping algorithm that takes advantage of existing classifiers; however, early errors are enhanced, resulting in poor algorithm performance.

8.1.1.2 Generative model algorithm

Generative models are commonly used for semi-supervised learning. It is assumed that the model satisfies the probabilistic condition $p(x, y) = p(y)p(x|y)$, where $p(x|y)$ is a recognisable mixed distribution, such as a Gaussian mixture model. This algorithm uses the generative model as a classifier, treating the probability that a large number of unmarked data belongs to each category as a set of missing parameters. Then, maximum likelihood estimation (MLE), maximum a posteriori estimation (MAP), or the Bayes algorithm is used to estimate the makers and model parameters, so that the mixed components can be determined. In ideal cases, the mixed distribution can be fully determined using only a marked example of a component. This algorithm can be seen to cluster around a small number of marked data, which is similar to a previous method that directly uses clustering assumptions.

For semi-supervised classification, the generative model can describe the identified categories that the data points belong to, rather than identifying the category by estimating the joint probability distribution of the data points. Commonly generated models include Gaussian distribution, Gaussian mixture model, EM algorithm (commonly used in image classification), polynomial distribution, naive Bayes model (e.g., EM algorithm and commonly used in text classification), and hidden Markov models

(e.g., Baum–Welch algorithm, commonly used in speech recognition). The generative model algorithm often focuses on the following questions [2]:

(1) Identifiable.

The hybrid model should ideally be identifiable. In general, $\{p_\theta\}$ is used to represent a distribution family indexed by a parameter θ . Let

$$\theta_1 \neq \theta_2 \Rightarrow p_{\theta_1} \neq p_{\theta_2}$$

represent the replacement of the mixed component. If the model family is identifiable, θ can be learnt from the replacement of the component model in field U .

(2) Model correct rate.

If the mixed model is assumed to be correct, unlabelled data can guarantee an increase in the correct rate; however, if the model is wrong, the unlabelled model is likely to reduce the correct rate. Cozman et al. (2003) observed this result [10]. Thus, it is important to construct a mixed model that corresponds to the practical situation. For example, Nigam (2001) pointed out that a title may contain several subheadings in the context of the text, and a model that combines multiple sub-headings is better than a model that uses a single one. Moreover, a method that reduces the weight of unlabelled samples is also important [11].

(3) EM local maximisation.

Even if the hybrid model is assumed to be correct, the actual mixed component is determined by the EM algorithm, but the EM algorithm easily leads to a local maximum. Nigam (2001) pointed out that if the local maximum deviates from the global maximum, unmarked samples will undermine the study. The relevant corrective action is to take the initiative to learn a precise choice for the starting point [11].

(4) Clustering and marking.

Instead of using a probability generation hybrid model, some methods use multiple clustering algorithms to aggregate the entire data set and then mark each group with labelled samples. Dara et al. (2002) explored this problem [12]. Although their performance is good, it is difficult to use these methods to analyse the nature of the algorithm, especially when the clustering algorithm matches the real data distribution.

Therefore, the advantage of the generative model is that it facilitates a clear and in-depth study of the probability of the framework. When the model is correct, the algorithm performance will be very good; however, the correctness of the model is difficult to prove and how to determine the actual match model remains an open problem. If the model does not match, unmarked data will degrade the performance of the algorithm.

8.1.1.3 Co-training algorithm and multi-view learning

(1) Co-training algorithm

The original co-training algorithm was proposed by A. Blum and T. Mitchell [13]. They assumed that a data set has two sufficiently redundant views that satisfy the following assumptions: (1) the feature can be divided into two sets; (2) each sub-feature set is sufficient to train into a good classification; and (3) the two sets are conditionally independent for a given class. Using the marked data on two independent subsets, two different classifiers can be initially trained. The two classifiers classify the unmarked data, each of which identifies each part of the data from the other with a high confidence level and then retrains the model to update with further iterations until there is no unlabelled data. The algorithm implicitly utilises clustering assumptions or manifold assumptions.

Algorithm 8.2: Standard collaborative training algorithm

Input: Given two fully redundant feature sets V_1 and V_2 ; Marked training set L ; Unmarked training set U

While unlabeled set U is not empty

 Classifier C_1 updates classifier C_2 :

- (a) Training the classifier C_1 on the feature set V_1 of the training set L ; U
- (b) classify the samples that are not marked in the sample set and calculate their confidence;
- (c) Add the n highest confidence samples to the training sample set L from the unlabeled sample set U . Classifier C_2 update classifier C_1 :
- (a) Training the classifier C_2 on the feature set V_2 of the training set L ;
- (b) classify the samples that are not marked in the sample set U and calculate their confidence;
- (c) Add the n highest confidence samples to the training sample set from the unlabeled sample set U .

End while

Output: classifier C_1 and C_2 , and final classification results

In the co-training algorithm, unmarked data helps to reduce the size of the form space. In other words, the two classifiers (or assumptions) must be consistent for a large number of unlabelled data and marked data. It is necessary to assume that the two sub-features are sufficient to take advantage of the markings of each learner, and that the sub-features must be conditionally independent so that the high-confidence data points of the classifier are independent and identical samples with respect to the other classifiers [14].

Nigam conducted extensive experiments for comparing the co-training algorithm and EM-generated mixed models. The results showed that the co-training algorithm performs better if the conditional independent assumptions are indeed satisfied. In addition, instead of labelling a small portion of the high-confidence point, the whole of U is marked as far as possible, and the example is called Co-EM. Finally, if there is no essential feature decomposition, the feature set is decomposed into two subsets randomly to establish the artificial decomposition, indicating that the co-training algorithm with artificial feature decomposition

is still useful, albeit not as useful as it was previously [11]. Jones (2005) used co-training, Co-EM, and other related algorithms for text information extraction. Balcan and Blum (2006) showed that the co-training algorithm is very efficient in the case where the classifier has only one marked point. Zhou et al. (2007) presented a co-training algorithm that utilises a typical correlation analysis and only requires a marked point [2].

The co-training algorithm requires strong assumptions about feature decomposition, and many researchers have considered whether these conditions can be reduced. Goldman and Zhou (2000) used two different classifiers trained from the same attribute set and a high-confidence data point of a learner. During the co-training process, each classifier estimates the marking confidence by statistical techniques, and the example with the highest marking confidence is submitted to another classifier as a marked training case, so that the other party can update the process, i.e., the process is repeated until a certain stop condition is reached [4]. Since then, Zhou (2004) extended the algorithm and proposed a democratic single-view multi-learner co-training algorithm to enable the use of multiple types of classifiers trained from labelled data, and thus predict the unlabelled samples. If the majority of the learners determine the classification of unmarked points, the category is marked as a marker. The sample point and its mark are re-added to the training set, and all the learners are restarted on the updated training set. The final mark is generated by all learners voting according to the weight [15]. Similarly, Zhou and Li (2005) proposed the tri-training algorithm using three learners; if two of them agree on the category of unmarked points, the category can be used for the third learner's learning. Thus, the need for a large number of measurements to measure the confidence of any learner is avoided, which saves time and improves availability available. For example, it can be used for data sets without different views or classifiers [16].

(2) Multi-view Learning.

More generally, we can use the protocol between learners to define learning paradigms, i.e., multi-view learning. The special assumptions of the co-training algorithm are generally no longer needed in the multi-view learning model. Conversely, if multiple hypotheses (with different inductive biases, e.g., decision trees and SVMs) are trained from the same labelled data set, it needs to be approximately estimated for any given unlabelled sample. DeSa (1993) studied multi-view learning. Sindhwani (2005) and Brefeld (2006) used it for semi-supervised regression, and Farquhar (2006) established a theoretical analysis among inter-learning agreement [2].

8.1.1.4 TSVM algorithm

Semi-supervised SVMs (S3VMs) are equivalent to transductive SVMs (TSVMs), which assume that different classes of unlabelled data can be separated by a large margin. Hence, the algorithm is implemented primarily by maximising the 'unmarked data

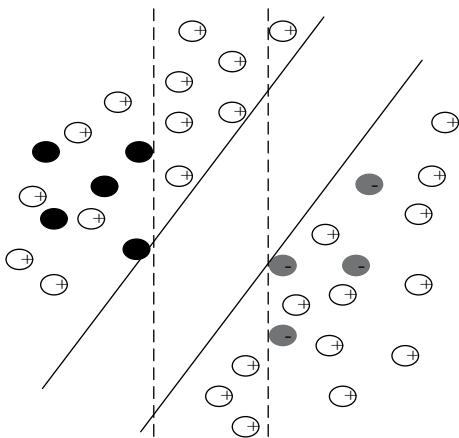


Fig. 8.1: TSVM.

margin'. S3VM is an extension of the standard support vector machine with unlabelled data [17]. Only labelled data can be used in the standard SVM to find the linear margin of the largest margin in the reconstructed kernel Hilbert space. In S3VM, unlabelled data can also be used to find unmarked data tags, by adding constraints to make unlabelled data outside the margin, so that the linear boundary has the largest margin in the marked and unmarked data. Decision boundaries have minimal generalisation errors on unmarked data [18].

As in Fig. 8.1, unlabelled data help to determine the sparse areas of the boundary. If there are only marked data, the maximum margin boundary is represented by the dotted line, and if there are unmarked points (white dots), the solid line indicates the maximum bound.

Algorithm 8.3: Semi-supervised SVMs algorithm flow

Input: kernel K ; weight λ_1, λ_2 ; labeled sample set $L = \{x_i, y_i\}$; unlabeled sample set $U = \{x_u\}$

Output: classification of test sample x with the symbol of $f(x)$

TSVM can be viewed as adding an extra regularization term on unlabeled sample.

Set $f(x) = h(x) + b$ where $h(x) \in H_K$. Solve the optimization problem:

$$\begin{aligned} \min_f & \sum_{i=1}^l (1 - y_i f(x_i))_+ + \lambda_1 \|h\|_{H_K}^2 + \lambda_2 \sum_{i=l+1}^n (1 - |f(x_i)|)_+ \\ \text{s.t. } & \frac{1}{n-1} \sum_{i=l+1}^n f(x_i) = \frac{1}{l} \sum_{i=1}^l y_i . \end{aligned}$$

The last item of $(z_+) = \max(z, 0)$ is generated by the unmarked point x with assignment flag $f(x)$, and the margin on the unmarked point is marked as $|f(x)|$. The loss function $(1 - |f(x_i)|)_+$ has a non-convex hat shape, as shown in Fig 8.2, which is dif-

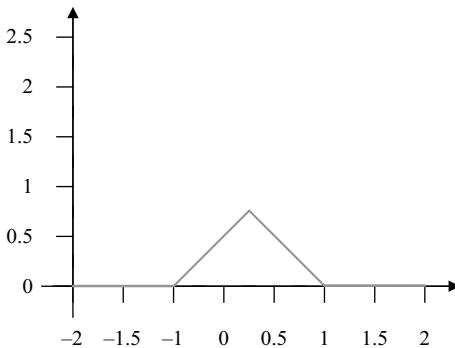


Fig. 8.2: TSVM loss function $(1 - |f(x_i)|)_+$.

ferent from the convex function of the standard SVM, which is the root cause of the difficulty in solving the optimisation problem.

The methods for solving this problem include SVM^{light}, vSVM, continuous S3VM, deterministic annealing method, bump program processing (CCCP), branch and bound method, and other methods. Chapelle and Zien (2005) proposed the use of a Gaussian function to estimate the cap-shaped loss and performed a gradient search in the original space. Sindhwani et al. (2006) used deterministic annealing methods to start with a ‘simple’ problem and gradually deform to TSVM goals. Similarly, Chapelle et al. (2006) used a continuous method that also begins with minimising a simple convex objective function that gradually deforms to the TSVM target (using Gaussian instead of cap-shaped loss), where the previous iteration is used to initialise the next iteration. Colcerbert et al. (2006) directly optimised the hard SVM by using an approximation optimisation process called the concavity and convexity program (CCCP), whose key is to check whether the net loss is the sum of the convex and concave functions. The upper limit is replaced by a linear upper limit that minimises the convex function to produce the upper limit of the loss function, which is repeated until the local minimum is reached. Using CCCP can significantly accelerate TSVM training. Sindhwani and Keerthi (2006) proposed a fast algorithm for linear S3VMs that is suitable for large-scale text applications [1]. Using the branch and bound search technique, Chapelle et al. (2006) found a global optimal method for smaller data sets and concluded that a higher accuracy rate can be achieved. Although branching and delimiting may not be useful for large data sets, the results still reflect the potential of TSVM with better estimation methods [2].

Thus, the advantage of semi-supervised SVMs is that there is a clear mathematical framework, where SVM algorithms and semi-supervised SVM algorithms can be used. However, in the process of the algorithm, the optimisation is very difficult and can easily fall into local optima. The model hypothesis condition is lower than the generation model, but the effect may be worse.

8.1.1.5 Graph-based algorithm

The graph is given on the marked and unlabelled points, which assumes that the example of the edge-weighted connection tends to the same mark. Hence, the semi-supervised learning algorithm based on the graph regularisation framework directly or indirectly utilises the manifold hypothesis. The concrete steps are as follows. First, establish a graph based on the training case and some similarity measure, where the nodes represent the labelled and unlabelled samples in the data set, and the edges (which may be weighted) reflect the similarity between the samples. Then, define the desired optimisation function, and the smoothness of the decision function on the graph is used as the regularisation term to obtain the optimal model parameters. These methods usually assume that the markings on the graph are smooth. The graphical method is essentially a parametric, discriminant, transitive method.

- (1) Normalisation of the graph.

Many graph-based methods can be considered as estimating functions f on graphs. Two conditions must be met by f at the same time: 1) it should be close to the label y_L of the given marked node; 2) the whole graph should be smooth. This indicates that the first term in a normalised frame is a loss function and the second term is a regularisation term.

Several graph-based methods are basically similar, which are different from the chosen loss functions and regularisation terms. Blum and Chawla (2005) regarded the semi-supervised learning problem as the minimum cut-off process on the graph, and they divided the graphs and obtained different regions corresponding to different categories [19]. Szummer and Jaakkola (2001) use the random walk theory on the graph to establish a semi-supervised learning random field model, which is somewhat similar to the diffusion kernel [20]. Zhu et al. (2003) pointed out that the Gaussian random field and harmonic function method is a continuous relaxation method for discrete Markov random fields. The objective function is a quadratic loss function with infinite weight and the normalisation of the Laplacian is based on the graph item [21]. Zhou D et al. (2003) pointed out that the objective function of the local and global coherence methods is the quadratic loss function plus the regularisation term, where the regularisation term is the normalised Laplacian [22]. Belkin et al. (2004) showed that the two points in regularities of the input space should have the same markings. On the basis of the Tikhonov regularisation algorithm, using the geometrical properties of the edge distribution, we introduce the regularisation that represents the inherent geometry of the sample item. However, after the incorporation of new regularisation terms, it is necessary to select two regularisation parameters, which not only increases the difficulty in solving but also requires the theoretical basis to be further improved [23]. In addition, researchers have explored the nucleus of the Laplacian spectrum, spectral transformation, local learning regularisation, and other aspects [2].

(2) Construction of the graph.

It is clear that graphs are the core of the algorithm in semi-supervised learning based on graphs. In practical applications, the graph is the direct form of data representation. Hence, the construction of the graph affects the performance of the algorithm to a certain extent. Zhu (2005) argued that building a good graph (which can adequately reflect the geometric structure of the data) is more important than choosing a method, and discussed the construction of the graph. However, the method for constructing the graph has not been studied extensively. On the basis of the minimum spanning tree (MST), Carreira-Perpinan and Zemel (2005) constructed a robust graph using the tree integration method. At present, the following methods are used for studying the structure: domain knowledge, nearest neighbour graph, and local fitting [2].

The advantage of the graph-based approach is that it has a clear mathematical framework. If the constructed graph fits the problem, the performance of the algorithm is good and it can be extended to directed graphs. However, if the graph does not match, the performance of the algorithm is poor, and the algorithm is very sensitive to the structure of the graph and the weight of the edge.

In summary, despite the various achievements of current semi-supervised learning methods, semi-supervised learning formed by the model ‘data = marked data + not marked data’ lacks a general system theory. In view of this practical situation, we conduct a study from the perspective of symmetry.

8.1.2 Questions raised

Symmetry is the most common and most important characteristic in nature. Modern science shows that all the important laws of nature are related to a certain symmetry, and all the interactions in nature have some type of special symmetry, i.e., ‘normative symmetry’ [24].

In mathematics, the relationship between two cases through established rules is called a transformation from one case to another. In physics, the concept of symmetry can be summarised as follows: if a phenomenon or system in a transformation (such as translation, rotation, space reflection, time inversion, stretching, and replacement) does not change, the phenomenon or system has symmetry corresponding to the transformation [25]. In general, symmetry transformation is a state transformation that maintains the state of the system i.e., the system is unchanged. In symmetry transformation, the statistical interpretation of the system is the same, and the laws of the system are unchanged [24]. Since each symmetry is associated with a particular transformation, the diversity of the symmetry is reflected in the various associated transformations. From the study of mathematical theory, we can see that the mathe-

mathematical branch of the general relation between the various transformations is group theory. Thus, the group is the basis for studying the symmetry problem. In semi-supervised learning, the learning system is a pattern of ‘data = notifiable data + unlabelled data’, and the identification and uncertainty of the markings can be linked by a certain rule, i.e., the unlabelled data can obtain the mark value through some transformation and turn into labelled data, and the transformation will not affect the essence of the interpretation or the inherent laws of the semi-supervised learning system, regardless of whether the data that marks itself forms a symmetrical transformation relationship naturally. In addition, the set of application objects, together with the operations defined on them, becomes a group; hence, the data naturally contains a group structure from the viewpoint of data analysis. Among the various groups, the most widely used by physicists is the Lie group. All symmetry transformations are composed of Lie groups, and all subspaces, linear transformations, and nonsingular matrices are also composed of Lie groups that are based on normal matrix multiplication. The mapping, transformation, measurement, division, etc. have great significance to the research of Lie group machine learning.

In this regard, Professor Li Fanzhang and his Lie group machine learning research group have been conducting studies since 2004. They first proposed the concept of Lie group machine learning and established the model of Lie group machine learning and related learning algorithms. Therefore, this chapter introduces Lie group into the semi-supervised learning method and gives the relevant contents of the Lie group semi-supervised learning algorithm.

8.2 Semi-supervised learning model based on the Lie group

According to the theory of Lie group and the research goal of semi-supervised learning, the method of Lie group representation in semi-supervised learning is given, and the semi-supervised learning model based on the Lie group algebra structure and geometric structure is written separately, which provides a theoretical framework for the study of the algorithm.

8.2.1 Representation of the Lie group in semi-supervised study

For any learning system, we can use Lie group G to represent it. Because Lie group itself is a non-empty set and a system, its learning process can be seen as a manifold (high dimension) mapped to an observable space (low dimensional) through a micro-mapping. The system differential can reduce the database dimension in the system or decompose the learning rules, so as to achieve the purpose of changing the system performance and achieving the learning objectives. Then, in the process of manifold differentiation of the learning system, the Lie group differential can be transformed into

the Lie algebraic differential by a Lie algebra. Then, the Lie algebra differential (equivalent to solving in linear space) is calculated, the numerical solution of the Lie algebra differential is pulled back to the Lie group, and the numerical solution of the Lie group learning system is obtained, which ensures that the numerical solution of the system is on the manifold. Massive amounts of data are always divided into groups that satisfy their group structure and groups that do not satisfy their group structure (intermediate data can be transformed by cohomology). The purpose of learning is to find effective data that fits specific learning objectives from the massive data of this learning system for analysis and use. However, the labelled data that is currently known to meet its group structure and meet the specific learning objectives may not be sufficient, and the vast majority of data is unlabelled data that is not aware of the learning objectives. Hence, Lie group can be used in a semi-supervised learning model to mine data [46].

Lie group semi-supervised learning is expressed as follows:

Through the analysis of the learning system, Lie group G that meets the specific structure is found. Suppose that there is a large sample set, where the sample number $n = l + k$ is massive and the sample dimension D is high.

$$G = \{x_1, x_2, \dots, x_l, x_{l+1}, x_{l+2}, \dots, x_{l+k} \in R^D\}$$

where x_1, x_2, \dots, x_l is a set of marked data that satisfies the structure of the learning system, i.e., there is a set $L = \{(x_1, y_1), (x_2, y_2), \dots, (x_l, y_l)\} \subset G$, $U = \{x_{l+1}, x_{l+2}, \dots, x_{l+k}\}$ which is a large set of unlabelled data sets ($l \ll k$) that do not know whether the group structure is satisfied. The purpose of the study is to mine as much data as possible, which meets the learning objectives, through the analysis and use of the structure and the Lie group (Lie algebra) to make sense of the structure and provide sufficient information for dimension reduction and structure learning.

8.2.2 Semi-supervised learning model based on Lie group algebra structure

Semi-supervised learning is considered from the algebraic group structure of the Lie group generating element, single parameter subgroup, and Lie algebra.

Definition 8.1. All things that meet certain conditions constitute a collection, and each of them is called an element of the collection.

Definition 8.2. If a linear combination is defined for elements in a collection and any linear combination of any set of elements in the set is still an element of the set, the set is called a linear set or a linear space.

Definition 8.3. If a set of linearly independent elements in a linear set can linearly be combined into any element in the set, it can be called a set of base of the linear set, where the number of linearly independent elements is called the dimension number of the linear set.

Definition 8.4. Define operations $[g_1, g_2]$ for any two elements g_1 and g_2 in a linear set (usually, a matrix can be used to represent it, where the element is marked). The result of the operation is still an element of the set. Assume that f_1, f_2 , and f_3 are constant and g_1, g_2 , and g_3 satisfy the following three relationships:

- (1) $[g_1, f_2g_2 + f_3g_3] = f_2[g_1, g_2] + f_3[g_1, g_3]$, or $[f_1g_1 + f_2g_2, g_3] = f_1[g_1, g_3] + f_2[g_2, g_3]$;
- (2) $[g_1, g_2] = -[g_2, g_1]$;
- (3) $[g_1, [g_2, g_3]] + [g_3, [g_1, g_2]] + [g_2, [g_3, g_1]] = 0$;

Then, this linear set constitutes a Lie algebra g .

Therefore, according to the definition of 8.1–8.4, a typical Lie algebra can be constructed in accordance with the data samples. The corresponding typical Lie group can be obtained from the Lie algebra. After obtaining the specific Lie group, combined with the tagged data L satisfying the group structure, the infinitesimal generators can be obtained according to the definition of the generator. Other unlabelled data in U can be determined according to the defined Lie group operation rule. If the unmarked data can be represented by the generator, then mark it by the category.

Furthermore, consider the isomorphism relationship between Lie algebra g and a single parameter subgroup from the viewpoint of differentiation. Assuming that a marked element and a unit element are known in Lie group G , a single parameter subgroup satisfying the defined condition can be found, and other unlabelled elements on the single parameter subgroup can be marked.

The semi-supervised learning model based on the Lie algebra structure is shown in Figure 8.3:

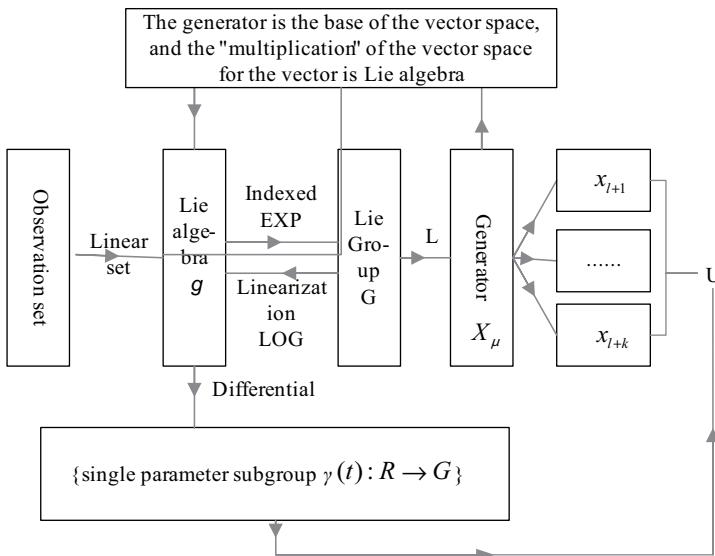


Fig. 8.3: Semi – supervised Learning Model Based on Lie Group Algebra Structure.

8.2.3 Semi-supervised learning model based on the geological structure of the Lie group

In this paper, we use the geometric properties of Lie group, such as translational and geodesic properties, which give the semi-supervised learning model based on Lie group geometry.

(1) Translality of Lie group.

Geometry is a homogeneous space in which objects can be moved freely and remain shaped, such that they can be controlled by an associated symmetric group. The Euclidean group gives Euclidean geometry, while the hyperbolic geometry originates from another Lie group. Thus, each homogeneous geometry corresponds to a different Lie group. However, with the subsequent development of Riemann's geometric work, geometries that are not homogeneous have attracted considerable attention, wherein the curvature changes with the position and the space no longer has overall symmetry. However, the Lie group still plays an important role, because there are Euclidean coordinates in the cut space. Thus, the Lie group can appear on an infinitely small level, but because of the need to distinguish among different points at different locations, a special method that can handle different Lie groups is needed to move objects.

In general, an observation set is given in a learning system, and the target results are learned according to the knowledge in the knowledge base. In this chapter, the observation set is assumed to be compactly connected, so that we can take an internal product $\langle \cdot, \cdot \rangle$ under the action of automorphism $Ad(G)$ in a single point of the observation set. For the inner product, a group of standard orthogonal bases is selected, and the left translation is used to expand them into the left invariant vector field of the observation set, so that the inner product can be uniquely scored at each point of the cut space on the observation set, which makes the inner product the standard orthogonal basis of the tangent space of the left invariant vector point, thus forming a Riemannian space.

From the left invariance of the orthogonal vector field group, it can be seen that all left translations are isometric transformations of the Riemannian space. From the invariance of the inner product, all the right translations are isometric transformations. Let $T_e(G)$ be the cut space of the unit point, i.e., the Lie algebra g of the cut space G . $T_e(G)$ is the cut space of G on any sample a , and dl_a, dr_a are the linear mappings induced between the left translation l_a and the right translation r_a , so that the geometric model can be studied from these aspects.

(2) Geodesic properties of Lie group.

A geodesic is one of the most basic and most important concepts of Riemannian manifolds, and it is very important for studying manifolds in high-dimensional space as a metric [45].

Definition 8.5. If ∇ is the affine contact of differential manifold M , $\theta: t \rightarrow \theta(t)$ is a curve that can be different, and $\dot{\theta}(t) = d\theta(d/dt)$ is a vector in $T_{\theta(t)}(M)$, which is called the tangent vector of the curve in $\theta(t)$. If

$$(\nabla_{\dot{\theta}(t)}(\dot{\theta}(t)))_{\theta(t)} = 0 \quad (8.1)$$

then $\theta(t)$ is a geodesic.

The geodesic is the natural extension of the straight line segment in the Euclidean space. On any point on the surface, there is only one geodesic in the given direction. The geodesic is the partial shortest path in the differential manifold. For the sample data distributed on the manifold, the geodesic can show the local neighbourhood properties.

For example, in Figure 8.4(a), according to the definition of Euclidean distance, two points can be considered in the same neighbourhood, but if the data distribution of the information is considered, from the geodesic distance, the two points cannot be adjacent points, as shown in Figure 8.4(b). Thus, the nature of the data manifold structure can be clearly considered from the geodesic departure.

In the differential manifold, it is very difficult to directly calculate the distance of the geodesic, since the manifold is locally Euclidean. Hence, the smaller part of the manifold is often replaced by the tangent plane, so that the calculation of the geodesic can be transformed into the calculation of the Euclidean distance in the two-dimensional plane.

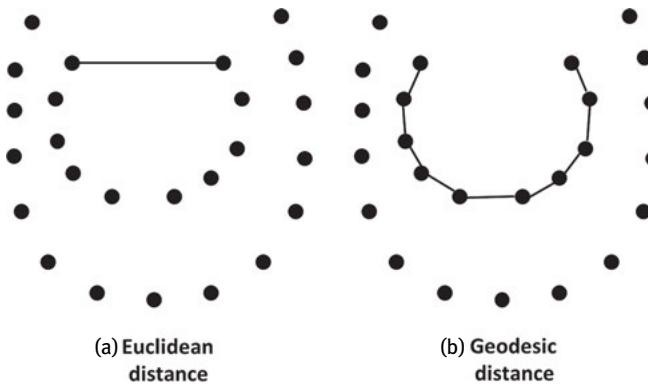


Fig. 8.4: Comparison of Euclidean Distance and Geodesic Distance of Manifold Data.

Definition 8.6. Let (M, x) be a differential manifold, where X is a local coordinate system on U , and $r_j: R^D \rightarrow R$ is the j -th coordinate function on R^D , i.e., $r_j(a_1, a_2, \dots, a_n) = a_j$, so that the j th coordinate function of x is $x_j: U \rightarrow R$, i.e., $x_j = r_j x$.

Definition 8.7. Let (M, x) be a differential manifold, $a \in M$, and let $C^\infty(M, a, R)$ be the set of real-valued functions in the area of a . Then, the tangent vector at the point a is a mapping, i.e., $v: C^\infty(M, a, R) \rightarrow R$. If x is a local coordinate system in the area a , then there exists a real-valued n tuple (a_1, a_2, \dots, a_n) , for each $f \in C^\infty(M, a, R)$. The n tuple satisfies

$$v(f) = \sum_{i=1}^n a_i \frac{\partial}{\partial r_i} (fx^{-1}) \Big|_{x(a)} . \quad (8.2)$$

All the tangent vectors of M at a are represented by $T_a(M)$, which is called the tangent space of M at a . If $x: U \rightarrow R^D$ is a coordinate system $(\partial/\partial x_i)$ at a and is the direction vector about x_i , $\{(\partial/\partial x_1), (\partial/\partial x_2), \dots, (\partial/\partial x_n)\}$ is the base coordinate of the tangent space $T_a(M)$. If the inner production $g_{i,j}(a) = \langle (\partial/\partial x_i), (\partial/\partial x_j) \rangle$ is a differential function on U , then the distance between a and b in the tangent space is

$$d(a, b) = \sqrt{\sum_{i,j=1}^n g_{i,j}(a)(a_i - b_i)(a_j - b_j)} . \quad (8.3)$$

We review semi-supervised learning from the geological structure of the Lie group by analysing manifolds, which mainly uses the Lie group left shift and right shift as well as other transformations and geodesic property. Suppose that the Lie group structure of the learning system is known. We generate its neighbourhood (with the k nearest neighbour or ϵ sphere neighbourhood) to satisfy each point $x_i (1 \leq i \leq l)$ in the marked data L of the group structure as a base point, and find the geodesic distance of x_i in the neighbourhood to find the shortest distance of the sample and mark it. Alternatively, by the left and right linear transformation (conformal transformation), the high-dimensional Lie group of manifolds is mapped to the corresponding tangent space, and in the linear structure of the tangent space, the unmarked points are easily described.

The semi-supervised learning model based on the geological structure of Lie group is shown in Figure 8.5.

8.3 Semi-supervised learning algorithm based on a linear Lie group

According to the theory of linear Lie group and the model framework described in Section 8.2, we choose $GL(n)$ ($n \times n$ A collection of non-degenerate real coefficient matrices), which is the most common linear Lie group and the most widely used example of the algorithm. The group element of $GL(n)$ is represented by a matrix, so it is also a matrix group. The semi-supervised learning algorithm based on linear Lie group is given and applied to the activity prediction of a class of steroid drugs with the operator generators of the linear Lie group and the translator on the manifold.

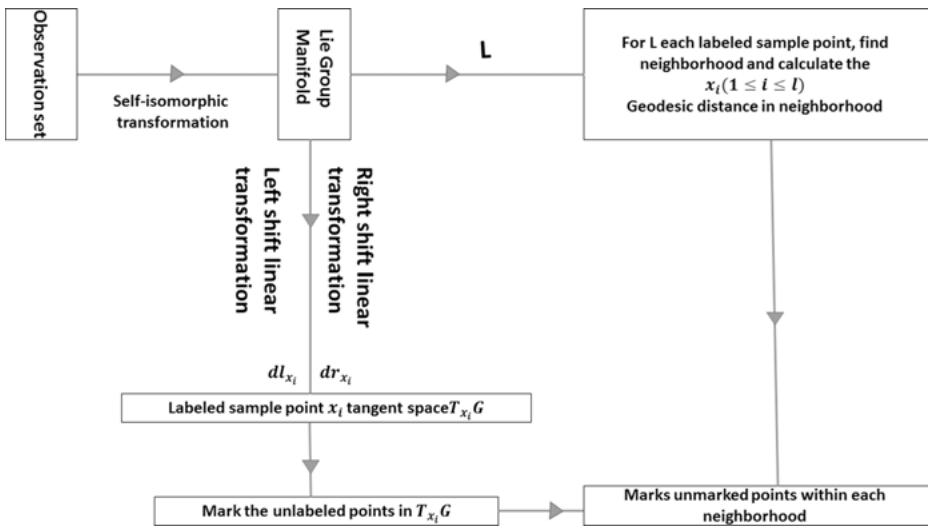


Fig. 8.5: Semi – supervised Learning Model Based on Geological Structure of Lie Group.

8.3.1 The general linear group

Definition 8.8. Let $A \in GL(n)$. Then,

$$A = \begin{pmatrix} g_1^1 & \dots & g_n^1 \\ \dots & \dots & \dots \\ g_n^1 & \dots & g_n^n \end{pmatrix} \text{ and } |A| \neq 0$$

Hence, $GL(n)$, which represents the multiplication of matrices, constitutes a group, and the unit matrix is the unit element of the group. Otherwise, $GL(n)$ is an open subset of R^{n^2} ; hence, $GL(n)$ has smooth structure as an open sub-manifold of R^{n^2} . Assume that $A = (a_i^j)$, $B = (b_i^j) \in GL(n)$, where a_i^j is the j th row and i th column element of A , and the j th row and i th column element of B is

$$(A \cdot B)_i^j = \sum_{k=1}^n a_k^j b_i^k. \quad (8.4)$$

In addition,

$$(A^{-1})_i^j = \frac{\bar{A}_i^j}{\det A}, \quad (8.5)$$

where \bar{A}_i^j represents the algebra of the element a_i^j of the determinant $\det A$. Thus, the inverse matrix A^{-1} is the rational formula of the element of the matrix A . Thus, the multiplication and the inverse operation of the group $GL(n)$ are both smooth. Hence, $GL(n)$ is an n^2 -dimensional Lie group [42].

I is used to represent the $n \times n$ unit matrix, which is the unit element of $GL(n)$. The set of $n \times n$ real matrices is recorded as $M(n, n)$; obviously, $M(n, n) \cong R^{n^2}$. The base

E_i^j ($i, j \leq n$) in $M(n, n)$ is obtained, where E_i^j ($i, j \leq n$) represents the $n \times n$ matrix whose i th row and j th column element is 1, and other elements are 0. Hence, any $n \times n$ matrix can be represented as

$$X = \sum_{i,j=1}^n X_j^i E_i^j \quad (8.6)$$

where X_j^i represents the element in the i th row and j th column of matrix X . Further, according to (4.6), the coordinate of the point $X \in M(n, n) \cong R^{n^2}$ under the standard frame $\{O; E_i^j\}$ is X_j^i , i.e., the coordinate system in $M(n, n)$ is given. The natural standard frame produced under the coordinate system is $\{X, \partial/\partial X_j^i|_X\}$, and

$$\left. \frac{\partial}{\partial X_j^i} \right|_X = E_i^j. \quad (8.7)$$

Because $GL(n)$ is the open subset of $M(n, n)$, the above coordinate system and natural standard frame of $M(n, n) \cong R^{n^2}$ can also fit $GL(n)$. Let the multiplication in $GL(n)$ be denoted by φ . Then, the expression of φ in the above coordinate system is

$$(\varphi(A, B))_j^i = (A \cdot B)_j^i = \sum_{k=1}^n A_k^i B_j^k. \quad (8.8)$$

We now require the left invariant vector field produced by the leftward movement of the tangent vector $E_i^j = \partial/\partial X_j^i|_I$ at the unit element I on the Lie group $GL(n)$, denoted by \bar{E}_i^j . Then,

$$\begin{aligned} \bar{E}_i^j|_A &= (L_A)_{*I} \left(\left. \frac{\partial}{\partial X_j^i} \right|_I \right) \\ &= \sum_{k,l=1}^n \left. \frac{\partial(\varphi(A, X))_l^k}{\partial X_j^i} \right|_{X=I} \cdot \left. \frac{\partial}{\partial X_l^k} \right|_A \\ &= \sum_{k,h,l=1}^n A_h^k \delta_i^h \delta_l^j \left. \frac{\partial}{\partial X_l^k} \right|_A \\ &= \sum_{k=1}^n A_i^k \left. \frac{\partial}{\partial X_j^k} \right|_A = \left(\sum_{k=1}^n X_i^k \frac{\partial}{\partial X_j^k} \right) \Big|_{X=A}. \end{aligned} \quad (8.9)$$

We directly calculate

$$\begin{aligned} [\bar{E}_i^j, \bar{E}_k^l] &= \left[\sum_{p=1}^n X_i^p \frac{\partial}{\partial X_j^p}, \sum_{q=1}^n X_k^q \frac{\partial}{\partial X_l^q} \right] \\ &= \sum_{p=1}^n \left(\delta_k^j X_i^p \frac{\partial}{\partial X_l^p} - \delta_i^l X_k^p \frac{\partial}{\partial X_j^p} \right) \\ &= \delta_k^j \bar{E}_i^l - \delta_i^l \bar{E}_k^j. \end{aligned} \quad (8.10)$$

If

$$\left[\overline{E}_i^j, \overline{E}_k^l \right] = \sum_{(p,q)} C_{(i,j)(k,l)}^{(p,q)} \overline{E}_p^q \quad (8.11)$$

the structural constants of Lie group $\text{GL}(n)$ are

$$C_{(i,j)(k,l)}^{(p,q)} = \delta_k^j \delta_i^p \delta_q^l - \delta_i^l \delta_k^p \delta_q^j. \quad (8.12)$$

Note that (8.10) is limited to the unit element I , and the multiplication table of the brackets in the tangent space $T_I(\text{GL}(n))$ is obtained:

$$\left[E_i^j, E_k^l \right] = \delta_k^j E_i^l - \delta_i^l E_k^j$$

Assume that $A, B \in M(n, n) \in T_I(\text{GL}(n))$, and set

$$A = \sum_{i,j=1}^n A_j^i E_i^j, \quad B = \sum_{i,j=1}^n B_j^i E_i^j.$$

Then,

$$\begin{aligned} [A, B] &= \sum_{i,j,k,l=1}^n A_j^i B_l^k \left[E_i^j, E_k^l \right] \\ &= \sum_{i,j,k,l=1}^n A_j^i B_l^k \left(\delta_k^j E_i^l - \delta_i^l E_k^j \right) \\ &= \sum_{i,j,k=1}^n \left(A_k^i B_j^k - B_k^i A_j^k \right) E_i^j \\ &= A \cdot B - B \cdot A \end{aligned} \quad (8.13)$$

The ‘ \cdot ’ in (8.13) represents matrix multiplication. The set $M(n, n)$ of $n \times n$ real matrices is the tangent space at the unit element of Lie group $\text{GL}(n)$. When it is the Lie algebra of Lie group $\text{GL}(n)$, the parentheses $[A, B]$ of any two elements $A, B \in M(n, n)$ represent its exchange $A \cdot B - B \cdot A$. The Lie algebra of Lie group $\text{GL}(n)$ is usually denoted by $gl(n)$.

The general linear group $\text{GL}(n)$ can be viewed as the automorphism group of n -dimensional vector space. Assume that V is n -dimensional vector space and $\text{GL}(V)$ represents the automorphism set of vector space V . The automorphism of V is the non-degenerate linear transformation from V to itself. A base $\{\delta_i\}$ in V is selected, and the automorphism $\alpha \in \text{GL}(V)$ accords the matrix $A = (a_i^j)$ under the base $\{\delta_i\}$, i.e.,

$$\alpha(\delta_i) = a_i^j \delta_j$$

Because α is non-degenerate, $\det A \neq 0$, and the matrix of the inverse mapping α^{-1} of α under the base $\{\delta_i\}$ is A^{-1} .

If β is another automorphism of V , $\alpha \circ \beta$ is also the automorphism of V . Thus, the combination of $\text{GL}(V)$ for mapping constitutes a group, and the unit element of the group is the constant mapping from the vector space to itself.

Assume that the matrix β under the base $\{\delta_i\}$ is $B = (b_i^j)$. Then,

$$\alpha \circ \beta(\delta_i) = \alpha \left(\sum_j b_i^j \delta_j \right) = \sum_j b_i^j \alpha(\delta_j) = \sum_{j,k} b_i^j a_j^k \delta_k .$$

Thus, the matrix $\alpha \circ \beta$ under the base $\{\delta_i\}$ is

$$\begin{bmatrix} a_1^1 b_1^j & \vdots & a_1^n b_n^j \\ \dots & \dots & \dots \\ a_j^n b_1^j & \vdots & a_j^n b_n^j \end{bmatrix} = \begin{bmatrix} a_1^1 & \dots & a_1^n \\ \vdots & \vdots & \vdots \\ a_1^n & \dots & a_1^n \end{bmatrix} \cdot \begin{bmatrix} b_1^1 & \dots & b_n^1 \\ \vdots & \vdots & \vdots \\ b_1^n & \dots & b_n^n \end{bmatrix} = A \cdot B .$$

Hence, if a base $\{\delta_i\}$ in V is selected, let $\alpha \in \mathrm{GL}(V)$ accord its matrix $A \in \mathrm{GL}(n)$ under the base $\{\delta_i\}$. Then, the mapping is one-to-one and is the automorphism from group $\mathrm{GL}(V)$ to group $\mathrm{GL}(n)$. Thus, $\mathrm{GL}(V)$ is an n^2 -dimensional Lie group. Its smooth structure is induced from $\mathrm{GL}(n)$.

8.3.2 Semi-supervised learning algorithm based on the linear Lie group

8.3.2.1 Operator generators of the linear Lie group

According to the above definition of $\mathrm{GL}(n)$, using the right movement as an example, from the local point of view, the definition of the whole infinite small right shift of Lie group is g , and g is closed on the commutator and thus constitutes the Lie algebra, such that G is generated locally with the move around the left and right of the current element g , which can be moved by the current element g . The operation is generated locally. First, we introduce the following agreement: f is the function defined on G , $g \in G$, and in G , we use

$$f^g(x) = f(gx) . \quad (8.14)$$

Defining the function f^g : from $(f^g)^h(x) = f^g(hx) = f(ghx) = f^{gh}(x)$, infer that there is a relationship

$$(f^g)^h = f^{gh} . \quad (8.15)$$

Definition 8.9. Take a cut vector Z at the unit element e of Lie group G , and use

$$X_x f = Z f^x (x \in G) . \quad (8.16)$$

to define the infinitesimal transformation X , which is called the whole infinitesimal right movement of G , denoted by g . Now, determine the explicit representation of X defined in (8.14). The multiplication of the nearest neighbour of unit element e of G in a local coordinate system is

$$(xy)^\mu = \phi^\mu(x, y), u = 1, 2, \dots, r .$$

which requires the synthesis function ϕ^μ to be parsed. When x is close to e , then $f^x(y) = f(xy)$ is meaningful to the other point y of the e neighbourhood, and

$$X_x f = Z f^x = \lambda^\mu \frac{\partial}{\partial y^\mu} . \quad (8.17)$$

In (8.17), $Z = \lambda^\mu \partial/\partial y^\mu$. If $xy = z$, (8.12) becomes

$$X_x f = \lambda^\mu \frac{\partial f}{\partial z^\nu} \left. \frac{\partial \phi^\nu}{\partial y^\mu} \right|_{y=e} . \quad (8.18)$$

To visually see that this is an infinitesimal rightward move, let $y^\mu = e^\mu + \lambda^\mu \delta_t$, and omit the item δ_t , which is higher than one. The mapping $x \mapsto xy$ can be regarded as an infinitesimal rightward move. The corresponding change of f is

$$\delta f = f(z) - f(x) = \left. \frac{\partial f}{\partial z^\nu} \frac{\partial \phi^\nu(x, y)}{\partial y^\mu} \right|_{y=e} \lambda^\mu \delta_t . \quad (8.19)$$

Comparing (8.18) with (8.19), we can see that $X_x f$ represents the following: when you move right along the given direction λ^μ , it corresponds to a change in the value of the unit of parameter t . To write (4.18) in a more convenient form, set

$$\psi_\mu^\nu = \left. \frac{\partial \phi^\nu}{\partial y^\mu} \right|_{y=e} (\mu, \nu = 1, \dots, r) . \quad (8.20)$$

The function ψ_μ^ν defined as such is the transform function of the Lie group. Using (8.20), we can write

$$X_x f = \lambda^\mu \psi_\mu^\nu(x) \frac{\partial f}{\partial x^\nu}$$

such that

$$X_x = \lambda^\mu \psi_\mu^\nu(x) \frac{\partial}{\partial x^\nu} (\mu, \nu = 1, \dots, r) . \quad (8.21)$$

Here, (8.21) is the infinitesimal rightward movement defined by $Z = \lambda^\mu \partial/\partial y^\mu$, and it can be seen that the infinitesimal rightward motion is the inverse vector field. From the infinitesimal right move (8.21), we know that

$$X_\mu(x) = \psi_\mu^\nu(x) \frac{\partial}{\partial x^\nu} = \left. \frac{\partial \phi^\nu(x, y)}{\partial y^\mu} \right|_{y=e} \frac{\partial}{\partial x^\nu} \quad (8.22)$$

is the generator of the Lie group. This generator is given in the form of an operator, called the Lie generation operator [43].

Similarly, we can define

$$X_\mu(y) = \psi_\mu^\nu(y) \frac{\partial}{\partial y^\nu} = \left. \frac{\partial \phi^\nu(x, y)}{\partial x^\mu} \right|_{x=e} \frac{\partial}{\partial y^\nu} \quad (8.23)$$

as the operator generating element corresponding to the infinitesimal leftward motion of the Lie group. When the synthetic function of the Lie group is given, it is easy to find the operator generating element of the Lie group.

From the viewpoint of the Lie group, (8.22) and (8.23) can be regarded as the generators of the Lie group, but from the viewpoint of Lie algebra, (8.22) and (8.23) are the basis vectors of Lie algebra.

8.3.2.2 Algorithm description

Using the above theoretical analysis, the semi-supervised learning algorithm based on linear Lie group is described in Algorithm 8.4:

Algorithm 8.4: SSLA-GL(n) algorithm flow

//algorithm function: Semi-supervised learning with linear Lie group method.//

Input: Sample set $X = (x_1, x_2, \dots, x_l, x_{l+1}, \dots, x_{l+k} \in R^D)$, where x_1, x_2, \dots, x_l are labelled data, and there exists a set $L = \{(x_1, y_1), (x_2, y_2), \dots, (x_l, y_l)\}$, $x_{l+1}, x_{l+2}, \dots, x_{l+k}$ are unlabelled data ($l < k$).

Output: The label y of data x

Steps:

Step 1: According to the input sample set $X = (x_1, x_2, \dots, x_l, x_{l+1}, \dots, x_{l+k} \in R^D)$, represented by a matrix, find the automorphism of the $GL(n)$ linear Lie group of the corresponding learning system using labelled data fully, representing the object that is to be studied as a linear Lie group, requiring the learning object to satisfy the linear Lie group structure.

Step 2: According to the algebra structure of Lie group, the group method is used for labelling.
Based on the marked data as the base point, find its operator generator (or the base vector of the corresponding algebra) through its infinitesimal right shift

$$X_\mu(x) = \psi_\mu^v(x) \frac{\partial}{\partial x^v} = \left. \frac{\partial \phi^v(x, y)}{\partial y^u} \right|_{y=e} \frac{\partial}{\partial x^v};$$

Determine other unlabelled data according to the group operation rules. If the unmarked data can be used to generate the meta-representation of the operator, it is marked by the category.

Step 3: According to the geometric structure of the linear Lie group, the analytic manifold is used for marking.

A constant inner product under the action of the automorphism is taken in the Lie algebra g of X :

$$\langle Ad(g)x_1, Ad(g)x_2 \rangle = \langle x_1, x_2 \rangle, x_1, x_2 \in g, g \in G$$

Take a set of standard orthogonal bases $\{X_i; 1 \leq i \leq l+k\}$, $\langle X_i, X_j \rangle = \delta_{ij}$. The left shift expands $\{X_i; 1 \leq i \leq l+k\}$ into the left constant vector field of G , which is recorded as X_i . Thus, the Riemann space structure is established;

In the Riemann space structure, the unmarked data is marked in each $T_{x_i}G$ space, with the marked data as the base point and the left (or right) translation linear transformation dl_{x_i} (or dr_{x_i}) of X . The tangent space $T_{x_i}G$ is mapped to the marked sample point x_i .

Step 4: Check whether all the data is marked. If not, then return to Step 2 and repeat; otherwise, go to the next step of Lie group machine learning, which is to classify the data. Complete data reduction and structural learning.

8.3.2.3 Algorithm analysis

The semi-supervised learning model based on Lie group algebra structure and geometric structure finds a special structure of the linear Lie group, targets the special structure of the linear Lie group, and finds its corresponding infinitesimal left and right operator generation elements, which is equivalent to finding the base vector of the Lie algebra corresponding to the Lie group. Based on a small number of marked data, according to the closure of the group algorithm and the law of the joint, it can be determined whether the other elements of the learning system can be represented

by the operator generator of the linear Lie group. If the unlabelled data can be represented by the generator of the group, the unlabelled data is also consistent with the Lie group and has the same markings as the tagged data of the Lie group. In addition, the identification of unlabelled data marks can be based on the geometric structure of the Lie group, i.e., the Lie group is an analytical manifold. After the data is mapped to a linear cut space, it is marked. This is an iterative process. When the data meets the learning needs, the process of marking can be interrupted.

The algorithm mainly considers the Lie group relationship $[X, Y]$ from the learning object X and the learning object Y , and makes full use of the algebraic group structure and the geometric manifold structure of Lie group to consider the unlabelled example from the perspective of the operator generating element and the translational dimension of the manifold, whose accuracy is higher than a single algorithm with algebraic or geometric methods. According to the needs of the problem, Step 2 and Step 3 can be carried out separately.

8.3.3 Experiment

Semi-supervised learning is often used for the processing of medical data. The structure-activity relationship model of the established drug can guide the design of new drugs, and the computer's ability to predict the activity of new drugs can save experimental costs and improve drug design efficiency significantly [50]. However, the cost of the assay is very high, the measured activity in the drug molecule is small, and the number of unmeasured [51, 52] cases is large. It would be useful to employ the unmeasured data to improve the accuracy of the model. Hence, the semi-supervised learning method, which can use non-tagged data to promote learning performance characteristics, is fully considered for drug activity prediction. Therefore, this paper is based on the semi-supervised learning algorithm of linear Lie group applied to drug activity prediction, which is expected to yield a new method of drug design.

8.3.3.1 Drug activity data set

Steroids have strong anti-inflammatory effects on inflammation caused by physical, chemical, biological, and immunological causes. In this chapter, the activity of corticosteroids in steroid compounds is studied and implemented according to the literature [53] by using the above SSLA-GL(n) algorithm to illustrate the effectiveness of the algorithm (note: in the literature [53], the SIMCA similarity pattern recognition method is used).

In the literature [53], a total of 77 steroid drugs with the structure shown in Figure 8.4 were selected. R1, R2, R3, R4, R5, and R6 are replaced with different substituents and the presence or absence of a double bond between the 1,2-position is taken into account, based on hydrocortisone 17-butyratate, a class of high activity; otherwise, the

second type of low activity is considered. The size of the training set is 77, of which 51 are the first high-activity compounds and 26 are the second low-activity compounds. To simulate the implementation of this algorithm, the data are segmented before the experiment; the 77 data sets are divided into marked and unmarked according to different proportions. In this paper, the data used in the literature is known, but the number of marked data is very small. The whole drug data set was randomly divided 10 times. The final result of the experiment was obtained by averaging the results of the 10 experiments. Then, the activity of the 8 parental compounds to be synthesised was predicted and the results were compared. For comparison of the consistency, the molecular descriptors and specific data for the molecular structure of this paper are shown in the literature.

8.3.3.2 Forecast result

The experimental environment was MATLAB 7.0 running on a computer with the following configuration: Pentium (R) D CPU 2.80 GHz, 1.0 GB memory. The number of marked and unlabelled data in the training data set was set to 2: 8, 3: 7, 4: 6, and 9: 1, represented by R . In the experiment, the sample data was first expressed as the simplest Lie group, i.e., the general linear group. The category of the sample data is marked according to the SSLA-algorithm. The distribution of pharmacokinetic activity is shown in Figure 8.6, where the x -axis represents the standard deviation of the activity of the drug to class 1, i.e., the high activity. The z -axis represents the standard deviation of the drug activity to class 2, i.e., low activity. The y -axis represents the predicted F value of drug activity [53].

In Figure 8.6, red + represents a highly active Class 1 compound, blue * represents a low-activity class 2 compound, green ◊ represents an unlabelled compound, and the eight black special symbols are the eight compounds to be predicted. The experimental data are analysed and the predicted results are shown in Table 8.1.

As can be seen from Table 8.1, the semi-supervised learning algorithm based on the linear Lie group has more than half of the accurate predictions of the activity, which is very useful, as the sample has been marked in the drug activity prediction and the number of unlabelled samples is useful. When the amount of data increases, the algorithm of SSLA-GL(n) reaches and exceeds the prediction accuracy of the literature [53], which indicates the effectiveness of the algorithm.

At the same time, to verify the correctness of the algorithm, SSLA-GL(n) is compared with common semi-supervised learning algorithms, such as self-training, TSVM, and co-training. In the experiment, self-training uses the C4.5 algorithm training classifier. The TSVM source code uses the LIBSVM package (LIBSVM package download address: <http://www.csie.ntu.edu.tw/~cjlin/>), where the radial basis function (RBF) is chosen as the kernel function. Co-training uses two K -nearest homogeneous supervised classifiers with different distance measures for semi-supervised learning.

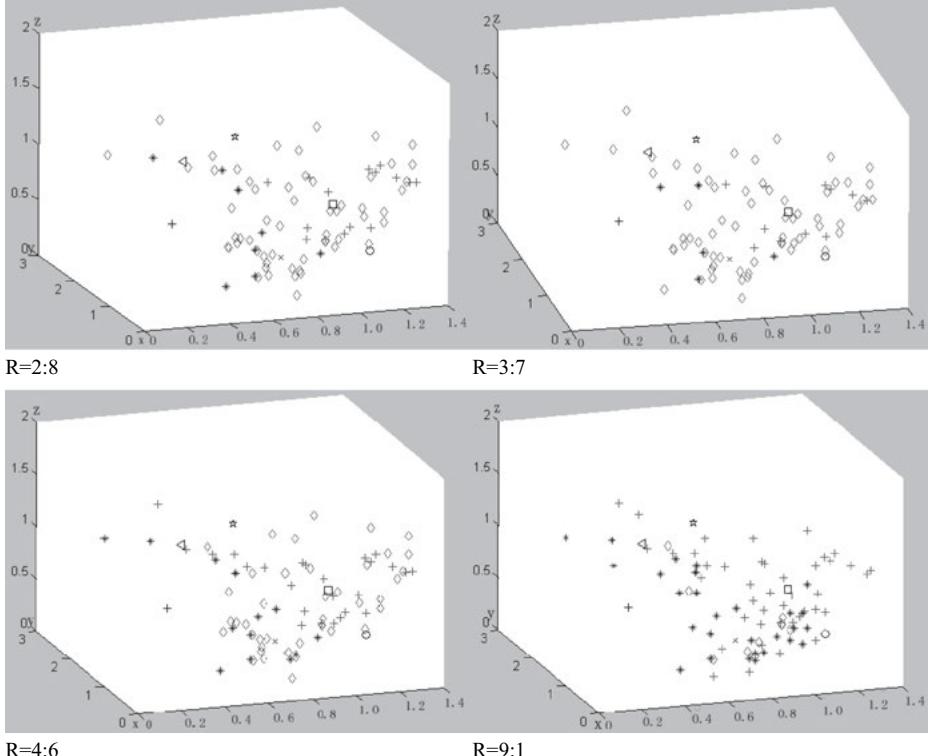


Fig. 8.6: Compound activity profile.

Tab. 8.1: The activity predicted results of 8 Unknown compounds (High activity is expressed as H, low activity is expressed by L, [H, L] indicates drug activity between high activity and low activity).

method activity No	SSLA-GL(n)				SIMCA
	2:8	3:7	4:6	9:1	
1	H	H	H	H	H
2	[H,L]	[H,L]	[H,L]	H	[H,L]
3	H	H	H	H	H
4	L	L	L	L	H
5	H	H	H	H	L
6	L	L	L	L	H
7	[H,L]	H	H	H	H
8	L	L	L	L	H
accuracy	62.5%	75%	81.5%	87.5%	85%

Experimental results on steroid drug activity data set

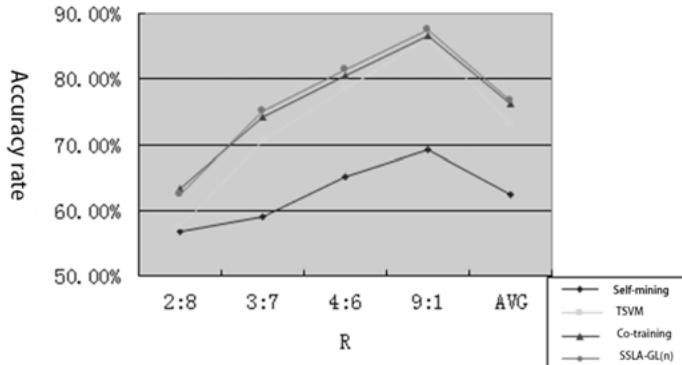


Fig. 8.7: Comparison of four algorithms on steroid drug activity data.

It can be seen from Figure 8.7 that the prediction results of the SSLA-GL(n) algorithm are nearly 10 percentage points higher than those of self-training and nearly 3 percentage points higher than those of TSVM. The results are similar to those of co-training, which indicates the accuracy of the SSLA-GL(n) algorithm.

8.4 Semi-supervised learning algorithm based on the parameter Lie group

According to the relevant theory in Section 2.3 and the model framework described in Chapter 3, the sample data is mapped to the parameter Lie group according to the representation of parameters. The semi-supervised learning algorithm based on the parameter Lie group is given from the matrix generator of the parameter Lie group and its geodesic distance on the manifold, and applied to the dimensionality reduction and classification of the five medical data sets on the UCI.

8.4.1 Sample data representation

Through the analysis of Lie group, we know that each element in the learning system G can be described by n properties. Each attribute can be described by r ($1 \leq r \leq n$) variables. The original data matrix is obtained from the sample of the actual problem. The learning object is expressed as a matrix group. Through the related theory of the parameter Lie group, we can see that the learning system G and a parameter Lie group have homology (homomorphism).

Assume that n samples (individual) are selected; r variables are selected for each sample, and the observed value $x_{i,j}$ of the j th variable of the i th sample is measured, with y_i representing the class marking in which x_i is. Then, the observed value and the corresponding tag value of n samples of r variables can be expressed as follows:

$$G = \begin{pmatrix} x_{1,1} & \dots & x_{1,r}, y_1 \\ \dots & \dots & \dots \\ x_{l,1} & \dots & x_{l,r}, y_l \\ x_{l+1,1} & \dots & x_{l+1,r}, y_{l+1} \\ \dots & \dots & \dots \\ x_{n,1} & \dots & x_{n,r}, y_{n,r} \end{pmatrix} = \begin{pmatrix} X_1 \\ X_2 \\ \dots \\ X_n \end{pmatrix} = (X_1, X_2, \dots, X_n)' . \quad (8.24)$$

Equation (8.24) constitutes a sample data matrix in which y_1 to y_l is the marker whose observed value is determined corresponding to x_1 to x_l . The marker from x_{l+1} to x_n is not determined and the corresponding y_{l+1} to y_n value is unknown, i.e., $l \ll n$. The unlabelled data accounts for a large proportion of the entire sample data, where the observed values of the r variables of the i th sample are vectors:

$$X_i = (x_{i,1}, x_{i,2}, x_{i,3}, \dots, x_{i,r}, y_i) \quad (8.25)$$

$x_{i,j}$ ($i = 1, \dots, n; j = 1, \dots, r$) is the observed data of the j th index of the i th sample.

8.4.2 Semi-supervised learning algorithm based on the parameter Lie group

8.4.2.1 Matrix generators of the parameter Lie group

Assume that G is a Lie group with r continuous parameters $\alpha^\mu (\mu = 1, \dots, r)$ and g is its element, so that r parameters can be selected, which makes its unit element correspond to the origin of the parameter space; i.e., $e \equiv g(0, \dots, 0)$. This is because of the analytic nature of the Lie group:

$$g(0, \dots, \alpha^r, \dots, 0) \approx g(0, \dots, 0) + i\alpha^\mu X_\mu(0, \dots, 0) \text{ (do not sum } \mu \text{)} \quad (8.26)$$

which gives an infinitely small element. The first order approximation of $\alpha^\mu (\mu = 1, \dots, r)$ can only be calculated, where i is the unit imaginary number. From (8.26), X_μ can be obtained as follows:

$$X_\mu = \lim_{\alpha^\mu \rightarrow 0} \frac{g(0, \dots, \alpha^\mu, \dots, 0) - g(0, \dots, 0)}{i\alpha^\mu} . \quad (8.27)$$

The entire nature of the parameter Lie group can be obtained from $X_\mu (1 \leq \mu \leq r)$ defined by the unit element annexed with the quantity r , and X_μ is the generated meta of the parameter Lie group. If there are r continuous parameters in the Lie group, there are r generated meta [43].

Using the multiplication rules of the group from a generated meta continuously, the group elements with finite distance from the unit element can be obtained. To get

the element $g(0, \dots, \alpha^\mu, \dots, 0)$, let $\alpha^\mu = N\alpha^\mu$, where N is a large positive integer and α^μ is a small one. Hence,

$$\begin{aligned} & g(0, \dots, \alpha^\mu, \dots, 0) \\ &= (g(0, \dots, \alpha^\mu, \dots, 0))^N \\ &= (e + i\alpha^\mu X_\mu)^N \quad (\text{do not sum } \mu). \\ &= \left(e + i\frac{\alpha^\mu}{N} X_\mu \right)^N \end{aligned} \tag{8.28}$$

Let $\alpha^\mu \rightarrow 0$, $N \rightarrow 0$, and use the algebraic identity

$$\lim_{N \rightarrow \infty} \left(1 + \frac{g}{N} \right)^N = \exp g. \tag{8.29}$$

Obtain

$$g(0, \dots, \alpha^\mu, \dots, 0) = \exp(i\alpha^\mu X_\mu) \quad (\text{do not sum } \mu) \tag{8.30}$$

which is an accurate result.

For the general element of parameter Lie group G , use the above results. Then, we can get

$$g(\alpha^1, \dots, \alpha^r) = \exp(i\alpha^\mu X_\mu). \tag{8.31}$$

The infinitely small element that is near the unit element of G , i.e., the infinitely small transformation, will constitute a transform group. Hence, let the infinitely small element be

$$g_\lambda = g(0, \dots, \alpha^\lambda, \dots, 0) = e + i\alpha^\lambda X_\lambda \quad (\text{do not sum } \lambda) \tag{8.32}$$

$$g_\nu = g(0, \dots, \alpha^\nu, \dots, 0) = e + i\alpha^\nu X_\nu \quad (\text{do not sum } \nu). \tag{8.33}$$

After calculating the first-order approximate of $\alpha^\mu (\mu = 1, \dots, r)$, we have

$$g_\lambda g_\nu = g_\nu g_\lambda \approx e + i(\alpha^\lambda X_\lambda + \alpha^\nu X_\nu) \quad (\text{do not sum } \lambda, \nu). \tag{8.34}$$

Here, (8.34) still gives the infinitely small element of G .

Theorem 8.1. *In the parameter space (r dimension) of Lie group G , the group element that is in the unit element neighbourhood can be generated by the following infinitely small generated meta (where $a_\rho (\rho = 1, \dots, r)$ is a group element):*

$$\begin{aligned} X^{(\beta)} &= V_{\beta\rho}(a) \frac{\partial}{\partial a_\rho} \equiv \left(\frac{\partial \phi_\rho(a, b)}{\partial b_\beta} \right)_{b=0} \frac{\partial}{\partial a_\rho} \\ &\quad (a = (a_1, a_2, \dots, a_r)) \end{aligned} \tag{8.35}$$

where $\phi_i(a, \delta a) \equiv a_i + da_i$ is the transformation function of group parameter space (as shown in Fig 8.8), and the parameter space is an r -dimensional linear space. Group element $A(a)$ is just like $F(x)$, which is a function of n -dimensional linear space.

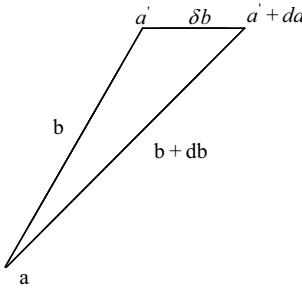


Fig. 8.8: The change of parameter vector in parameter space.

Proof. Note that $\phi(a, 0) = a$, $\phi(a, \delta a) = a + da$. As shown in Fig. 8.8, there are two paths from $a \rightarrow a' + da'$. One is from a to $a' + da'$ directly, i.e.,

$$\phi(a, b + db) = a' + da' . \quad (8.36)$$

The other one passes a' and then arrives at $a' + da'$. Let $a' = \phi(a, b)$. Hence, $a' + da' = \phi(a' + \delta b)$.

Apply Taylor unfolds to (8.36), and write it as a component formula:

$$\begin{aligned} \phi_\rho(a', \delta b) &= \phi_\rho(a', 0) + \sum_{\sigma=1}^r \left[\frac{\partial \phi_\rho(a, b)}{\partial b_\sigma} \right]_{b=0} \delta b_\sigma \\ &= \phi_\rho(a', 0) + \sum_{\sigma=1}^r V_{\rho\sigma}(a) \delta b_\sigma \end{aligned}$$

i.e.,

$$da_\rho = \sum_{\sigma} V_{\rho\sigma}(a) \delta b_\sigma (\rho = 1, 2, \dots, r)$$

$\forall A(a) \in G$ can be viewed as the function of r variables in parameter space. Further, $a \rightarrow a + da$ can induce the transformation $A(a) \rightarrow A(a) + dA(a)$ in $A(a)$.

In addition,

$$\begin{aligned} dA(a) &= \sum_{\rho=1} \frac{\partial A(a)}{\partial a_\rho} da_\rho \\ &= \sum_{\rho=1}^r \sum_{\sigma=1}^r V_{\rho\sigma}(a) \delta a_\rho \frac{\partial A(a)}{\partial a_\rho} \\ &= \sum_{\rho, \sigma} \delta a^\rho V_{\rho\sigma}(a) \frac{\partial A(a)}{\partial a_\sigma} . \end{aligned}$$

Thus, the structure of the generated meta of the Lie group and infinitely small calculator are naturally the same. \square

8.4.2.2 Algorithm description

Using the above theoretical analysis, we describe the semi-supervised learning algorithm based on the parameter Lie group (SSLA-PLG), as shown in Algorithm 8.5:

Algorithm 8.5: SSLA-PLG Algorithm flow

//Algorithm function: use Lie group method for semi-supervised learning //

Input: Sample set $X = (x_1, x_2, \dots, x_l, x_{l+1}, \dots, x_{l+k} \in R^D)$, where x_1, x_2, \dots, x_l is the labelled data, i.e., there exists the set $L = \{(x_1, y_1), (x_2, y_2), \dots, (x_l, y_l)\}$, where $x_{l+1}, x_{l+2}, \dots, x_{l+k}$ are unlabelled data ($l < k$).

Output: Label y of data x

Process:

Step 1: According to the input sample set $X = (x_1, x_2, \dots, x_l, x_{l+1}, \dots, x_{l+k} \in R^D)$, represent it with a parameter separately. Determine the corresponding learning system isomorphism (state) parameter Lie group and make full use of the marked data, representing the object to be learned with the parameters of Lie group. The object is required to meet the structure of the parameter Lie group.

Step 2: According to the algebraic structure of the parameter Lie group, the group method is used for marking.

Based on the marked data as the base point, the definition of the element generated by the parameter Lie group is

$$X_\mu = \lim_{\alpha^\mu \rightarrow 0} \frac{g(0, \dots, \alpha^\mu, \dots, 0) - g(0, \dots, 0)}{i\alpha^\mu}$$

which calculates the generated meta.

Determine other unlabelled data according to the group operation rule, and if the unlabelled data can be represented by the matrix generator, the category is marked.

Step 3: According to the geometric structure of the parameter Lie group, the analytic manifold is used for marking.

Generate the field $U_i (1 < i < l)$ of all labelled sample points x_i ;

Analyse the neighbourhood information $U_i (1 < i < l)$ of each labelled sample point x_i . Select each constant inner product $g_{i,j}(x_i)$ under the function of self-isomorphism on the unit point in the sample set. Calculate the value of $g_{i,j}(x_i)$;

Put the value of $g_{i,j}(x_i)$ into the following formula:

$$d(x_i, x_j) = \sqrt{\sum_{i,j=1}^n g_{i,j}(x_i)(x_{ii} - x_{ji})(x_{ij} - x_{jj})}$$

Calculate the distance between x_i and x_j in $U_i (1 < i < l)$.

Determine the smallest value d of each point x_i corresponding to x_j , and mark x_j as the marker that is the same as x_i .

Step 4: Check whether the data is marked; if not, return to Step2 and repeat. Otherwise, go to the next step, i.e., data classification. Complete the data reduction and structural learning steps.

8.4.2.3 Algorithm analysis

The semi-supervised learning model based on the algebraic and geometric structure of Lie group, i.e., the special expression of the parameter Lie group, is studied. Through the matrix generator of the parameter Lie group, based on the marked data and ac-

cording to the closure and union law of the group algorithm, we determine whether the other elements of the learning system can be used to represent the generator. If the unlabelled data can be represented by the matrix generating element of the parameter Lie group, then the unlabelled data is also in accordance with its learning objectives and can be marked with the same markings as the marked data. In addition, according to the geometric structure of the Lie group, based on the marked data, we find its local neighbourhood, calculate the geodesic distance in the neighbourhood, and label the unmarked data with the shortest geometric distance. This is an iterative process. When the data meets the learning needs, the process of marking can be interrupted.

The algorithm mainly considers the Lie group relations $[X, Y]$ from the learning object X and the learning goal Y , and makes full use of the algebraic group structure and the geometric manifold structure of the Lie group. From the matrix generation element of the parameter Lie group and the geodesic distance of the Lie manifold, the unlabelled example is marked, which is more accurate than a single algebraic or geometric algorithm. According to the needs of the problem, Step 2 and Step 3 can be carried out separately.

8.4.3 Experiment

With the rapid development of computing power and storage technology, large-scale data acquisition is easy but raises a new problem. In fields such as bioinformatics, face recognition, image retrieval, and medical data processing, the data obtained are often high-dimensional and high-order data. Without effective feature extraction, the so-called ‘dimension of the disaster’ problem can easily occur. On the other hand, the inherent laws of the data cannot be directly understood and found [54]. Therefore, the validity of dimension reduction is the key to the success of many machine learning and data mining algorithms.

8.4.3.1 Semi-supervised dimension reduction

The two most widely used methods of dimensionality reduction are principal component analysis (PCA), i.e., an unsupervised algorithm, and linear discriminant analysis (LDA), i.e., a supervised algorithm [54, 55]. For an unsupervised algorithm, because there is no known class of sample set, or the number of sample data cannot be found, the amount of information that can be used is very small, resulting in a great deal of uncertainty in the algorithm. For supervised algorithms, the lack of a large number of marked data results in weak training of the model generalisation ability. In view of the actual case of a large number of unlabelled data and relatively poorly marked data [2], we can use the semi-supervised dimensionality reduction method for processing.

First, introduce semi-supervised dimensionality reduction (abbreviated as SSDR). Given: a set of sample data sets $G = [g_1, g_2, \dots, g_n]$, where each data is represented

by a D -dimensional matrix according to its attribute parameters; a set of positive tags and negative tags: $L = L_+ \cup L_-$. The purpose of dimension reduction is to find a projection relation vector $R(r_1, r_2, \dots, r_d)$ ($d \ll D$), so that the low-dimensional representation of the data $Y = R^T G$ can preserve the original structure of the data and the tag information [56]. Hence, the proposed SSLA-PLG algorithm for data dimensionality reduction can be considered.

8.4.3.2 UCI data set

In this experiment, five medical data sets from the UCI machine learning database were selected, dimension-reduced, and classified by SSLA-PLG. Table 8.2 lists some basic information about the data set. The download address of the data set is <http://archive.ics.uci.edu/ml/index.html>.

In the experiment, first, the geodesic distance $d(x_i, x_j)$ between sample x_i and x_j is calculated. For the semi-supervised dimensionality reduction, combined with the partial mark information and the neighbour information of the sample, the parameters α, β are introduced to adjust the distance matrix:

$$D(x_i, x_j) = \begin{cases} \sqrt{1 - \exp(-d^2(x_i, x_j)/\beta)} - \alpha & \text{if } \exists y_j = y_i \\ \sqrt{1 - \exp(-d^2(x_i, x_j)/\beta)} & \text{if } \exists y_j, \text{ but } y_j \neq y_i \\ \sqrt{\exp(-d^2(x_i, x_j)/\beta)} & \text{if } y_j \text{ unknown .} \end{cases}$$

Thus, the distance between the classes becomes narrow, and the distance between the expansions of the classes increases, which supports the data after the dimensionality reduction to maintain the same distribution. The neighbour relationships of the samples can be maintained by $D(x_i, x_j)$. In this experiment, the parameter α of the five data sets takes a value of 0.1, while the parameter β changes with different marked samples. Based on the ten-fold cross-validation method, the first group was used as the test sample, and the remaining nine groups were used as training examples from

Tab. 8.2: Data description.

Data set name	Total number of samples	Feature dimension	Number and number of categories	Category meaning
Breast Cancer	569	30	2 class 357: 212	Benign or malignant tumor
Parkinsons	197	23	2 class 149: 48	Whether suffering from Parkinson's disease
Hepatitis	155	19	2 class 71: 84	Surviving or dead from hepatitis
Diabetes	768	8	2 class 500: 268	Whether suffering from diabetes
Thyroid	125	5	3 class 60: 35: 30	Type of thyroid disease

the labelled samples, based on the idea of semi-supervised learning after dimension reduction. Then, a cycle operation was performed. The correct rate of each classification and the final correct rate, i.e., the average of 10 values, were determined.

8.4.3.3 Experimental results

The experimental environment was MATLAB 7.0 running on a computer with the following configuration: Pentium (R) D CPU 2.80 GHz, 1.0 GB memory. For different data sets, the dimension of the dimension reduction space, the size of the neighbourhood (K) in the algorithm, and the ratio of the marked samples (R) of the data are determined. Part of the experimental data is shown in Figures 8.9 to 8.14:

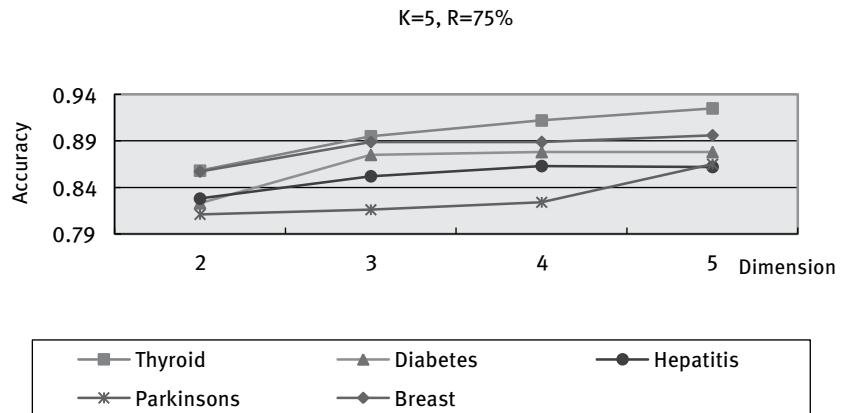


Fig. 8.9: The Change of Classification Recognition Rate with Dimensionality of Dimension reduction space when $K=5, R=75\%$.

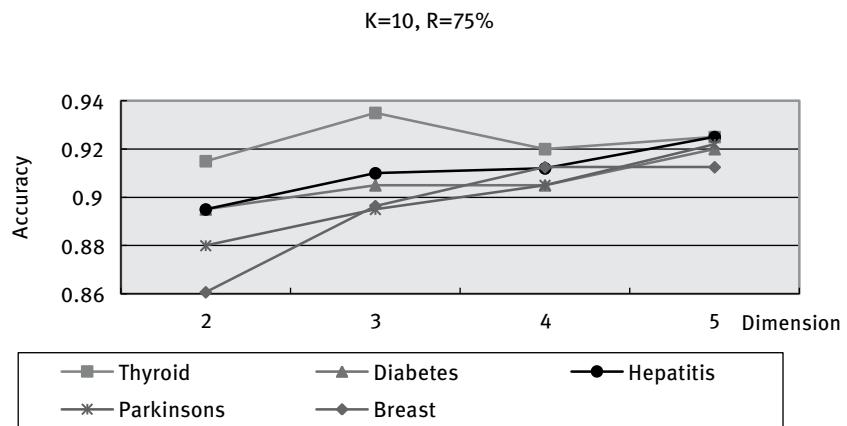


Fig. 8.10: The Change of Classification Recognition Rate with Dimensionality of Dimension reduction space when $K=10, R=75\%$.

From Figures 8.9 to 8.14, we can see the following:

- (1) The experimental results show that the correctness of the data set is consistent with the feature dimension and distribution structure information of the data set itself. The dimension of the dimension space after dimension reduction, the size of the neighbourhood (K) when the algorithm is used, and the ratio of the data that has been marked (R) are determined. However, the choice of K and its dimension will also affect the running time of the algorithm. In practice, consider the results and select the appropriate parameters.
- (2) The experimental results show that when $K = 10$ and $R = 75\%$, the accuracy of the five data sets is greater than 86%, which indicates the effectiveness of the algorithm. Considering the overall recognition rate of the samples is 75% or more, it has a good effect on the actual problem.

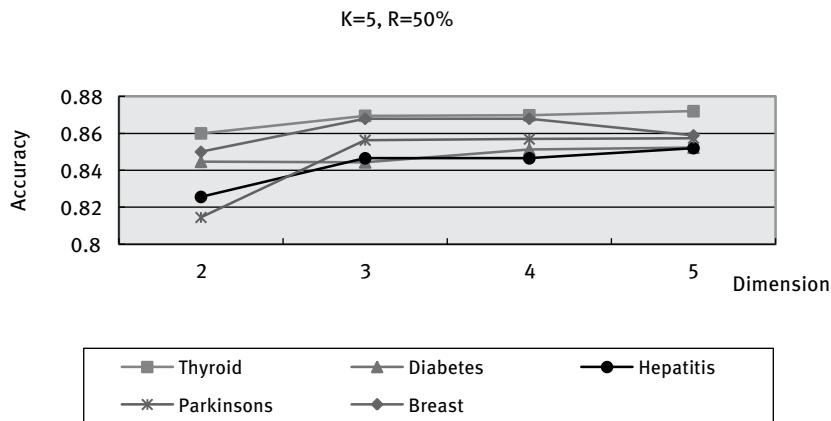


Fig. 8.11: The Change of Classification Recognition Rate with Dimensionality of Dimension reduction space when $K=5$, $R=50\%$.

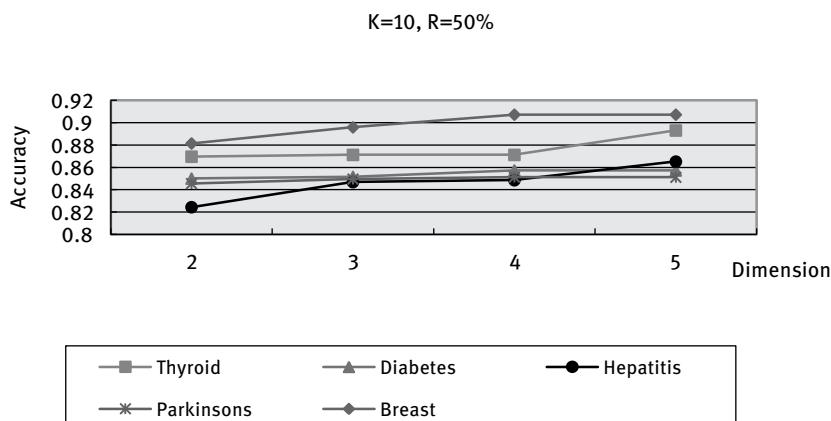


Fig. 8.12: The Change of Classification Recognition Rate with Dimensionality of Dimension reduction space when $K=10$, $R=50\%$.

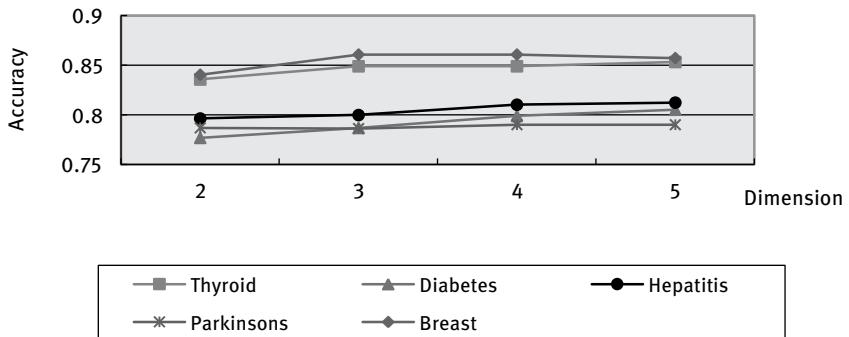


Fig. 8.13: The Change of Classification Recognition Rate with Dimensionality of Dimension reduction space when K=5, R=25%.

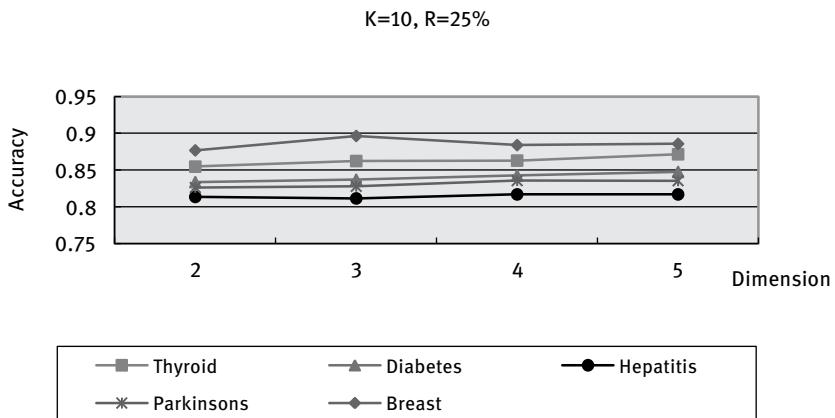


Fig. 8.14: The Change of Classification Recognition Rate with Dimensionality of Dimension reduction space when K=10, R=25%.

At the same time, the TSVM algorithm is used in the experiment to introduce the unmarked data, using the LIBSVM package, and the RBF kernel function is chosen. When R is 25%, 50%, and 75% respectively, compared with Algorithm 8.5 (parameter K = 10, down to three-dimensional space) in terms of the classification accuracy rate, the results are shown in Figure 8.15.

It can be seen from Figure 8.15 that when R = 75%, the classification accuracy of the SSLA-PLG algorithm is similar to that of the TSVM algorithm, but when the ratio of labelled and unlabelled samples becomes smaller, the classification accuracy of the SSLA-PLG algorithm is generally higher than that of TSVM, indicating the effectiveness of the SSLA-PLG algorithm. The reasons are summarised as follows:

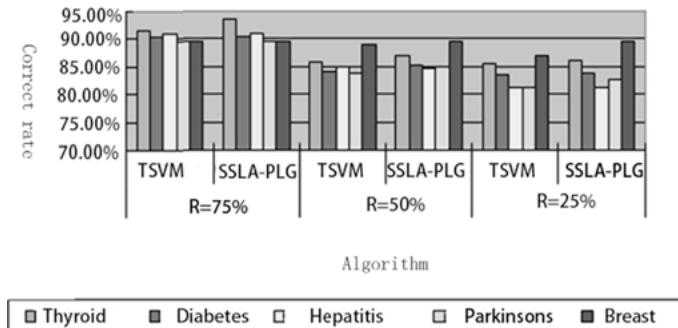


Fig. 8.15: The Comparison result between algorithm SSLA-PLG and TSVM.

- (1) SSLA-PLG is a semi-supervised learning method that preserves local (manifold) and global (group) relationships. A major flaw based on the local algorithm is that it is affected by the noise. In the experiment, the neighbourhood of the data is considered as the basis of the marked data, which is equivalent to the local data manifold. The redundancy information is reduced, the data set is limited to a group model, and the overall information of the data set after dimension reduction will not be reduced.
- (2) In the semi-supervised dimensionality reduction, the labelled sample and the unlabelled sample are trained together, requiring a sample adjacent to the high-dimensional sample space (using a geodetic distance that reflects the nature of the data set). The distance is still close, and artificial parameters and the distance matrix are used for adjustment. However, K is determined by experience. These parameters require further investigation.

8.5 Summary

Semi-supervised learning mainly combines a small number of marked training data, analyses the structure and distribution of data from a large number of unlabelled data, and makes full use of the information to improve the performance of the classification algorithm. The Lie group is a natural combination of algebraic and geometric structures. It is a useful tool for studying the relationship between marked and unmarked data in semi-supervised learning. Therefore, this chapter introduced Lie group theory into semi-supervised learning. Good representation of the algebraic and geometric structure and analysis of the data serve as the basis for the Lie group semi-supervised learning algorithm. The main contributions are as follows:

- (1) This chapter analysed the research status of semi-supervised learning and summarised five common semi-supervised learning algorithms.
- (2) From the algebraic structure of the Lie group, group generators, single parameter subgroups, and Lie algebras, the semi-supervised learning model based on the Lie group algebra structure was obtained. In addition, the semi-supervised learning model based on Lie group geometry was given from the translational and geodetic properties of the Lie group.
- (3) For a class of special linear Lie groups- $GL(n)$, a semi-supervised learning algorithm (SSLA- $GL(n)$) based on a linear Lie group was given and applied to a class of steroid drug activity prediction. The experimental results showed that the algorithm is effective and accurate.
- (4) A semi-supervised learning algorithm (SSLA-PLG) based on the parameter Lie group was proposed for the special expression of the parameter Lie group, and it was used to reduce the dimension and classification of five UCI medical data sets. The experimental results showed that the algorithm is feasible and has a high accuracy rate.

Bibliography

- [1] Zhou ZH. Preface of the special issue on semi-supervised learning [J]. Journal of Software, 2008, 19(11), 2789–2790.
- [2] Zhu X. Semi-supervised learning literature survey [R]. Technical Report 1530, Department of Computer Sciences, University of Wisconsin at Madison, Madison, WI, Jul. 2008. <http://pages.cs.wisc.edu/~jerryzhu/research/ssl/semireview.html>.
- [3] Zhou ZH. Machine learning and Application 2007 [M]. Beijing: Tsinghua university press, 2007, 259–275.
- [4] Goldman S, Yan Z. Enhancing supervised learning with unlabeled data [C]. In: Proceedings of the 17th International Conference on Machine Learning(ICML'00), Morgan Kaufmann, San Francisco, CA, 2000, 327–334.
- [5] Joachims T. Transductive inference for text classification using support vector machines [C]. In: Proceedings of the 16th International Conference on Machine Learning (ICML'99), Bled, Slovenia, 1999, 200–209.
- [6] Lawrence ND, Jordan MI. Semi-supervised learning via Gaussian processes [C]. Advances in Neural Information Processing Systems (NIPS), Cambridge, MA: MIT Press, 2005, 17, 753–760.
- [7] Grandvalet Y, Bengio Y. Semi-supervised learning by entropy minimization [C]. Advances in Neural Information Processing Systems (NIPS), Cambridge, MA: MIT Press, 2005, 17, 529–536.
- [8] Zhu X, Ghahramani Z, Lafferty J. Semi-supervised learning using Gaussian fields and harmonic functions [C]. In: Proceedings of the 20th International Conference on Machine Learning (ICML'03), Washington, DC, 2003, 912–919.
- [9] Yarowsky D. Unsupervised word sense disambiguation rivalling supervised methods [C]. In: Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL'95), Cambridge, MA: MIT Press, 1995, 189–196.
- [10] Cozman F, Cohen I, Cirelo M. Semi-supervised learning of mixture models [C]. In: Proceedings of 20th International Conference on Machine Learning (ICML'03). Washington, DC, 2003, 1, 99–106.

- [11] Nigam K. Using unlabeled data to improve text classification [D]. Carnegie Mellon University. Doctoral Dissertation, 2001.
- [12] Dara R, Kremer S, Stacey D. Clustering unlabeled data with SOMs improves classification of labeled real-world data [C]. In: Proceedings of the World Congress on Computational Intelligence (WCCI'02), Honolulu, HI, 2002, 3, 2237–2242.
- [13] Blum A, Mitchell T. Combining labeled and unlabeled data with co-training [C]. In: Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT'98), Wisconsin, MI, 1998, 92–100.
- [14] Yi X. Research on some problems of Semi-supervised learning [D]. Master's degree thesis, Beijing: Tsinghua University, 2004.
- [15] Zhou Y, Goldman S. Democratic co-learning [C]. In: Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04), Boca Raton, FL, 2004, 594–602.
- [16] Zhou ZH, Li M. Tri-training: Exploiting unlabeled data using three classifiers [J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(11), 1529–1541.
- [17] Xu ZL, Jin R, et al. Efficient Convex Relaxation for Transductive Support Vector Machines. NIPS 2007. <http://nips.cc/Conferences/2007/Program/event.php?ID=901>.
- [18] Jia P, Yin J, Huang X, et al. Incremental Laplacian eigenmaps by preserving adjacent information between data points [J]. Pattern Recognition Lett., 2009, 30(16), 1457–1463.
- [19] Blum A, Chawla S. Learning from labeled and unlabeled data using graph mincuts [C]. In: Proceedings of International Conference of Machine Learning (ICML'05), San Francisco, CA, 2005, 19–26.
- [20] Szummer M, Jaakkola T. Partially labeled classification with Markov random walks [C]. Advances in Neural Information Processing Systems (NIPS), Cambridge, MA: MIT Press, 2002, 14, 945–952.
- [21] Zhu XJ, Ghahramani Z. Semi-supervised learning using Gaussian fields and harmonic functions [C]. In: Proceedings of International Conference of Machine Learning (ICML'03), Washington DC, 2003, 912–919.
- [22] Zhou D, Bousquet O, Lal T, et al. Learning with local and global consistency [C]. Advances in Neural Information Processing Systems (NIPS), Cambridge, MA: MIT Press, 2004, 16, 321–328.
- [23] Belkin M, Niyogi P. Semi-supervised learning on Riemannian manifolds [J]. Machine Learning, 2004, 56(1–3): 209–239.
- [24] Zhang DM, Zhong ZC. Application of group theory guidance [M] first edition. Wuhan: Huazhong University of Science and Technology Press, 2001.
- [25] Gao CS. Group theory and its application in particle physics [M] first edition. Beijing: Higher Education Press, 1992.
- [26] Meng DJ, Bai CM. Lie Group [M] First edition. Beijing: Science Press, 2003.
- [27] Li FZ, Xu H. The theory framework of Lie group Machine Learning (LML) [J]. Computer Technology and Application, 2007, 1(1), 62–80.
- [28] Li FZ, Xu H. Study on Lie group machine learning [J]. Journal of Computational Information Systems, 2005, 1(4), 843–849.
- [29] Li FZ, Xu H. Lie Group Machine Learning's Axiom Hypothesizes [C]. In: Proceedings of 2006 IEEE International Conference on Granular Computing (IGrC'06), 2006, 401–404.
- [30] Li FZ, Xu H. The Design of SU (n) Classifier of Lie Group Machine Learning (LML) [J]. Journal of Computational Information Systems, 2005, 1(4), 835–841.
- [31] Li FZ, Xu H. Geometry Algorithms of Dynkin Diagrams in Lie Group Machine Learning [J]. Journal of Nanchang Institute of Technology, 2006, 25(2), 75–78.
- [32] Li FZ, Xu H. SO(3) Classifier of Lie Group Machine Learning (LML) [J]. Journal of Communication and Computer, 2006, 4(3), 76–80.

- [33] Xu H, Li FZ. Algorithms of Dynkin Diagrams in Lie Group Machine Learning [J]. *Journal of Communication and Computer*, 2006, 4(3), 13–17.
- [34] Xu H. Lie Group machine learning machine application [D]. Master's degree thesis, Suzhou: Soochow University, 2007.
- [35] Chen F. Research on the algorithm of Lie Group machine learning subspace orbit generation [D]. Master's degree thesis, Suzhou: Soochow University, 2007.
- [36] Fu HX. A symplectic group classifier in Lie Group machine learning [D]. Master's degree thesis, Suzhou: Soochow University, 2008.
- [37] He SP. The study of quantum group classifier in Lie Group machine learning [D]. Master's degree thesis, Suzhou: Soochow University, 2008.
- [38] Xu HX. A semi supervised learning algorithm based on Lie Group and its application [D]. Master's degree thesis, Suzhou: Soochow University, 2009.
- [39] Zhang J. Research on fiber learning algorithm based on Manifold Learning [D]. Master's degree thesis, Suzhou: Soochow University, 2008.
- [40] Xu HX, Li FZ. Semi-Supervised Learning algorithm based on Lie Group [C]. In: *Proceedings of 2009 Global Congress on Intelligent Systems (GCIS'09)*. 2009, 573–577.
- [41] Min X, Li FZ. A homology coboundary learning algorithm for data reduction [C]. In: *Proceedings of 2008 International Conference on Advanced Intelligence (ICAI'08)*, 2008, 97–102.
- [42] Guan W, Li FZ. Drug molecular Design using Lie group Machine Learning (LML) [C]. In: *Proceedings of 2008 International Conference on Advanced Intelligence (ICAI'08)*, 2008, 411–414.
- [43] Chen WH. Differential Manifolds [M] Second edition. Beijing: Higher Education Press, 2001.
- [44] Shao D, Shao L, Guo Z. Lie Group [M]. First edition. Beijing: Science Press, 2008.
- [45] Lie Group network learning resources <http://www.maths.dept.shef.ac.uk/magic/course.php?id=43>.
- [46] Tian C. Lie Group and its application in differential equation [M] First edition. Beijing: Science Press, 2001.
- [47] Zhou ZH. Machine learning and data mining [J]. *Communication of Chinese Computer Society*, 2007, 3(12), 35–44.
- [48] Hall BC. Lie groups, Lie Algebras, and Representations: An Elementary Introduction [M], Springer, 2007.
- [49] Guo ZR. Drug molecular design [M] First edition. Beijing: Science Press, 2005.
- [50] Xu XJ, Hou TJ, Jiao XB. Computer aided drug molecular design [M] First edition. Beijing: Chemical Industry Press, 2004.
- [51] Liang YZ, Xu J, Yao JH. Course of chemical informatics [M] First edition. Beijing: Chemical Industry Press, 2005.
- [52] Li D, Li GZ, Lu WC. Co-Training method for the prediction of drug activity [J]. *Computer Science*, 2006, 33(12), 159–161.
- [53] Hu YH, Lu ZH. Cluster analysis of structure-activity relationship in drug design – discriminant analysis [J]. *Computer application research*, 2007, 11(24), 191–195.
- [54] Yao YY, Xu L. Study on the structural activity correlation of steroid compounds by SIMCA method [J]. *Computer and Applied Chemistry*, 1993, 10(2), 81–90.
- [55] Wei J, Peng H. A semi-supervised dimension reduction method based on local and global preservation [J]. *Journal of Software*, 2008, 19(11), 2833–2842.
- [56] Martinez AM, Kak AC. PCA Versus LDA [J]. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2001, 23(2), 228–233.
- [57] Yang X, Fu XH, et al. Semi-supervised nonlinear dimensionality reduction [C]. In: *Proceedings of the 20th International Conference on Machine Learning (ICML'06)*, Pittsburgh, USA, 2006, 1065–1072.

- [58] Fedichev P, Vinnik A. Biological Spectra Analysis: Linking Biological Activity Profiles to Molecular Toxicity [R]. Technical Report. <http://www.q-pharm.com/qtox.pdf>.
- [59] James RB, Gamberger D, et al. Evaluation of Artificial Intelligence Based Models for Chemical Biodegradability Prediction [J]. *Journal of molecules* 2004, 9(12), 989–1003.
- [60] Niu B, Jin YH, et al. Predicting toxic action mechanisms of phenols using AdaBoost Learner [J]. *Chemometrics and Intelligent Laboratory Systems*, 2009, 96, 43–48.
- [61] Niazi A, Jameh-Bozorghi S, et al. Prediction of toxicity of nitrobenzenes using ab initio and least squares support vector machines [J]. *Journal of Hazardous Materials*, 2008, 151, 603–609.
- [62] Chen XM. Prediction of dihydrofolate reductase inhibitors activity using machine learning methods [D]. Master's degree thesis, Chengdu: Sichuan University, 2007.
- [63] Wang ZG. The establishment of a machine learning method for the activity prediction model of selective cyclooxygenase – 2 inhibitors [D]. Master's degree thesis, Chengdu: Sichuan University, 2007, machine learning methods.

9 Lie group kernel learning

9.1 Matrix group learning algorithm

In this section, we will discuss the related theory of matrix group and analyse related techniques based on matrix group learning algorithms.

9.1.1 Related basic concepts

Let k represent a (swap) field. In the vast majority of cases, we are concerned with this field as $k = \mathbb{R}$ (real) or $k = \mathbb{C}$ (complex). This chapter deals only with real fields, i.e., $k = \mathbb{R}$. In addition, the framework of general machine learning problems is often accompanied by a suitable norm. Hence, this section will also simply discuss the norm of a matrix.

9.1.2 Matrix group

9.1.2.1 Definition of the matrix group

Let $M_{mn}(K)$ be an $m \times n$ matrix, where each element belongs to K . Let A_{ij} or a_{ij} denote the element that is in the i -th row and j -th column of matrix A . Then,

$$A = (a_{ij}) = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}.$$

We can also use a special representation: $M_n(K) = M_{nn}(K)$, $K^n = M_{n1}(K)$. $M_{mn}(K)$ is a K -vector space with matrix addition and scalar multiplication. The zero vector is denoted as O_{mn} , which is an $m \times n$ matrix composed of zero elements. It can also be simply expressed in terms of O , the concrete dimensions of which depends on the context in which it appears.

$$(E^{rs})_{ij} = \delta_{ir}\delta_{js} = \begin{cases} 1 & (\text{if } i = r \text{ and } j = s) \\ 0 & (\text{otherwise}) \end{cases}$$

Matrix E^{rs} forms a set of bases of $M_{mn}(K)$. Hence, the dimension of the K -vector space is

$$\dim_K M_{mn}(K) = mn.$$

When $n = 1$, the standard base vector of $K^n = M_{n1}(K)$ is represented as

$$e_r = E^{r1} (r = 1, 2, \dots, m).$$

Once the problem is limited to the K -vector space with dimension n^2 , then $M_n(K)$ is an algebraic loop with matrix addition and square multiplication operations, having zero element $O_n = O_{nn}$. Unless $n = 1$, $M_n(K)$ is not exchangeable by taking an $n \times n$ unit matrix O_n as a unitary element. $M_n(K)$ is also an important example of finite-dimensional K -algebras.

In matrix theory, the matrix determinant is a mapping $\det: M_n(K) \rightarrow K$, which has the following properties:

- (1) For any $A, B \in M_n(K)$, $\det(AB) = \det(A)\det(B)$;
- (2) $\det(I_n) = 1$;
- (3) If and only if $A \in M_n(K)$, $\det A \neq 0$, A is reversible.

Next, we present two very important matrix groups, namely the general linear group $GL_n(K)$ and special linear group $SL_n(K)$. Their definitions are as following:

$$GL_n(K) = \{A \in M_n(K) | \det A \neq 0\}$$

$$SL_n(K) = \{A \in M_n(K) | \det A = 1\} \subseteq GL_n(K)$$

$GL_n(K)$ and $SL_n(K)$, form a group under matrix multiplication. More recently, we have $SL_n(K) \leq GL_n(K)$, i.e., $SL_n(K)$ is a subgroup of $GL_n(K)$.

The matrix groups used in this chapter are subgroups of $GL_n(K)$ and $GL_n(\mathbb{R})$.

9.1.2.2 Matric space of matrix group

In this section, we will define a norm $\|\cdot\|$ on $M_n(K)$. It is worth noting that we choose a specific norm mainly for the convenience of multiplication. In fact, the norm of any other vector space gives an equivalent metric topology on $M_n(K)$. $M_n(K)$ has numerous valuable norms. For further details, readers may refer to [45].

The most basic element of a measure is the length of the vector in space, and the length of a vector $x = (x_1, x_2, \dots, x_n)^T \in K$ is defined as:

$$|x| = \sqrt{|x_1|^2 + |x_2|^2 + \dots + |x_n|^2}. \quad (9.1)$$

This is an optional norm in the multivariate norm of vector space K^n . For any $A \in M_n(K)$, consider the following set:

$$S_A = \left\{ \frac{|Ax|}{|x|} \mid 0 \neq x \in K^n \right\}.$$

It has a subset

$$S_A^1 \equiv \{ |Ax| \mid x \in K^n, |x| = 1 \} \subseteq S_A.$$

If $x \neq 0$, then

$$\frac{|Ax|}{|x|} = |Ax'|$$

where $x' = x/|x|$ and $|x'| = 1$. The subset $\{x \in K^n \mid |x| = 1\} \subseteq K^n$ is closed and bounded and therefore compact. Then, the real-valued function

$$\{x \in K^n \mid |x| = 1\} \rightarrow \mathbb{R}, x \mapsto |Ax|$$

is bounded, and its upper bound norm is

$$\sup S_A = \sup S_A^1 = \max S_A^1 = \max S_A .$$

This means that the following real numbers are defined:

$$\|A\|_{\mathbb{R}} = \max S_A = \max S_A^1 .$$

The norm $\|\cdot\| : M_n(K) \rightarrow \mathbb{R}$ here is called the operation on $M_n(K)$ or the upper bound norm. More general terms of the norm on the K -algebra can be found in [44] under Definition 4.31.

There are many types of normals in the matrix space, but none of the $M_n(K) \rightarrow \mathbb{R}$ maps can be used as a norm, because the mapping $\|\cdot\| : M_n(K) \rightarrow \mathbb{R}$ must satisfy the following properties:

- (1) If $t \in K$, $A \in M_n(K)$, then $\|tA\| = |t| \|A\|$;
- (2) If $A, B \in M_n(K)$, then $\|AB\| \leq \|A\| \|B\|$;
- (3) If $A, B \in M_n(K)$, then $\|A + B\| \leq \|A\| + \|B\|$;
- (4) If $A \in M_n(K)$, then $\|A\| = 0$ if and only if $A = 0$;
- (5) $\|I_n\| = 1$.

The most widely used norm in the matrix groups is the Hilbert-Schmidt norm, which is also called the Frobenius norm in the Euclidean space:

$$\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2} . \quad (9.2)$$

Obviously, the Frobenius norm satisfies the five above-mentioned properties. As long as there is no confusion, we use $\|A\|$ instead of $\|A\|_F$. Other aspects of the norm theory can be found in [44, 45].

9.1.2.3 Cases of the matrix group

A group $G \leq \mathrm{GL}_n(K)$ is also an K -matrix group. For convenience, G is usually called a subgroup of $\mathrm{GL}_n(K)$. Before we introduce some common examples of matrix groups, we review some of their properties.

- (1) If $G \leq \mathrm{GL}_n(K)$ is a matrix subgroup, then the closed subgroup $H \leq G$ is a matrix subgroup of $\mathrm{GL}_n(K)$.
- (2) If G is a matrix subgroup and $H \leq K$, $K \leq G$, then H is a matrix subgroup of G .

The proof of the above-mentioned properties is omitted, but can be found here [44].

Given a matrix subgroup $G \leq \mathrm{GL}_n(K)$, define the determinant on $\mathrm{GL}_n(K)$ as the following mapping:

$$\det_G : G \rightarrow K^*, \det_G A = \det A .$$

Using \det instead of \det_G the later will not produce ambiguity; \det_G is a continuous group homomorphism. When $K = \mathbb{R}$, we have

$$\mathbb{R}^+ = \{t \in \mathbb{R} | t > 0\}, \mathbb{R}^- = \{t \in \mathbb{R} | t < 0\}, \mathbb{R}^* = \mathbb{R}^+ \cup \mathbb{R}^-.$$

Note that \mathbb{R}^+ is a subgroup of $\mathrm{GL}_1(\mathbb{R}) = \mathbb{R}^*$ and an open set. Moreover, \mathbb{R}^- is an open set. Thus \mathbb{R}^+ and \mathbb{R}^- are closed subsets, i.e., closed open sets. For $G \leq \mathrm{GL}_n(K)$, we have

$$\det_G^{-1} \mathbb{R}^+ = G \cap \det^{-1} \mathbb{R}^+, G = \det_G^{-1} \mathbb{R}^+ \cup \det_G^{-1} \mathbb{R}^-.$$

Thus, G is the non-intersection union of the following two closed open subsets:

$$G^+ = \det_G^{-1} \mathbb{R}^+, G^- = \det_G^{-1} \mathbb{R}^-.$$

Because of $I_n \in G^+ = \det_G^{-1} \mathbb{R}^+$, G^+ will never be empty. In fact, G^+ is a closed subset of G . Thus, G^+ is a matrix subgroup of $\mathrm{GL}_n(\mathbb{R})$. When $G^- \neq \emptyset$, G is not connected because it is the union of two disjoint sets. When $G^- = \emptyset$ and $G = G^+$, G may be connected and may be disconnected.

The matrices and mathematical fields involved in this chapter are real fields, i.e., $K = \mathbb{R}$. This has already been described at the beginning of this section. Hence, the domain symbol K will no longer appear in the following symbol, and the default field is the real field \mathbb{R} . After this agreement, the symbol can be simplified and is thus easy to understand. Remove the domain symbol K ; then, the dimension symbol n is moved to the original position of K . For example, replace the original symbol $\mathrm{GL}_n(\mathbb{R})$ with $\mathrm{GL}(n)$, replace the original symbol $M_{mn}(\mathbb{R})$ with $M(m, n)$, and replace the original symbol $M_n(\mathbb{R})$ with $M(n)$.

1. Orthogonal group $O(n)$, special orthogonal group $\mathrm{SO}(n)$

An $n \times n$ matrix A satisfies an orthogonal matrix if each of its column vectors is standard orthogonal, i.e.,

$$\sum_{l=1}^n A_{lj} A_{lk} = \delta_{jk} \quad (1 \leq j, k \leq n)$$

where δ_{jk} is called the Kronecker symbol. When $j = k$, $\delta_{jk} = 1$; when $j \neq k$, $\delta_{jk} = 0$. Another description is as follows. When A can maintain its inner product as an orthogonal matrix, i.e., for all vectors x and y in \mathbb{R}^n , we have $\langle x, y \rangle = \langle Ax, Ay \rangle$, where the angular brackets represent the inner product of the vector, i.e., $\langle x, y \rangle = \sum_k x_k y_k$. A more straightforward definition can be that: $A^T A = I$, i.e., $A^T = A^{-1}$.

Since $\det A^T = \det A$, when A is an orthogonal matrix, we can get $\det(A^T A^{-1}) = \det(A^2) = \det I = 1$. Thus, we have $\det A = \pm 1$ for all orthogonal matrices. This formula shows that the orthogonal matrix must be an invertible matrix. Therefore, for the orthogonal matrix A , we have

$$\langle A^{-1}x, A^{-1}y \rangle = \langle A(A^{-1}x), (A^{-1}y) \rangle = \langle x, y \rangle.$$

Furthermore, if the product of the two orthogonal matrices is also orthogonal, then all $n \times n$ orthogonal matrix sets will form a group, i.e., the orthogonal group $O(n)$, which is a subgroup of $GL(n)$.

The set of all $n \times n$ orthogonal matrices with determinant values of 1 constitutes a special orthogonal group $SO(n)$, which is a subgroup of $O(n)$ and therefore is a subgroup of $GL(n)$. From the aspect of geometry, the elements of $O(n)$ are obtained either by rotation or rotation and reflection; however, the elements of $SO(n)$ are only transformed by rotation.

2. Unitary group $U(n)$, special unitary group $SU(n)$

An $n \times n$ complex matrix A is a unitary matrix when each of its column vectors is standard orthogonal, i.e.,

$$\sum_{l=1}^n \overline{A_{lj}} A_{lk} = \delta_{jk} \quad (1 \leq j, k \leq n) .$$

It can also be said that when A can maintain the inner product, it is a unitary matrix, i.e., for all vectors x and y in \mathbb{C}^n , we have $\langle x, y \rangle = \langle Ax, Ay \rangle$, where the angular brackets represent the inner product of the vector, i.e., $\langle x, y \rangle = \sum_k \overline{x_k} y_k$. Another definition may be more straightforward: $A^* A = I$, i.e., $A^* = A^{-1}$, where A^* is the adjoint matrix of A : $(A^*)_{jk} = \overline{A_{kj}}$.

Since $\det A^* = \overline{\det A}$, when A is a unitary matrix, we have $\det(A^* A) = |\det A|^2 = \det I = 1$. Thus, for all orthogonal matrices, we have $|\det A| = 1$. This shows that any unitary matrix is reversible. Like the orthogonal group, all $n \times n$ unitary matrices constitute the unitary group $U(n)$, which is a subgroup of $GL(n; \mathbb{C})$.

The set of all $n \times n$ unitary matrices with determinant values of 1 constitutes a special unit group $SU(n)$, and a unitary matrix has a determinant value $e^{i\theta}$ for any θ ; hence, $SU(n)$ is a smaller subset than $U(n)$.

3. Symmetric group $Sym(n)$ and symmetric orthogonal group $Sym^+(n)$

An $n \times n$ matrix A is a symmetric matrix if it is an element in the following set

$$Sym(n) = \{A \in M(n) \mid A = A^T\}$$

and an $n \times n$ matrix A is a positive definite symmetric matrix if and only if it is both a symmetric matrix and a positive definite matrix:

$$Sym^+(n) = \{A \in Sym(n) \mid A > 0\} .$$

According to the common matrix multiplication rule, $Sym(n)$ can constitute a vector space. However, $Sym^+(n)$ can only form a manifold space; it cannot form a vector

space. Nevertheless, it does not have the group structure in conventional matrix multiplication. In [25], it has been shown that as long as we build a new operation on the above set,

$$A_1 \odot A_2 = \exp(\log_2 A_1 + \log_2 A_2) . \quad (9.3)$$

Then we can form a group (Sym^+, \odot) , which is a Lie group, where \exp and \log_2 are the exponential and logarithmic operations of the matrix. In Lie group theory, we call these the exponential mapping and logarithm mapping, respectively, which will be described later.

In the above, we introduce several common matrix groups, where $O(n)$, $\text{SO}(n)$, $U(n)$, and $\text{SU}(n)$ are all Lie groups. $\text{Sym}^+(n)$ can also be formed as a Lie group after adding a new matrix operation. The detailed proof can be found in [25] and [43].

9.1.3 The learning algorithm of the matrix group

The previous section introduced the basic concepts of matrix groups and the knowledge of matrix group spatial measures. On this basis, this section constructs a KNN algorithm and K-means an algorithm based on a matrix vector space.

$M_{mn}(K)$ is a K vector space with matrix addition and scalar multiplication. A metric operator is added to the vector space: $\rho(A, B)$, where $A, B \in M_{mn}(K)$, which satisfies the five properties of the norm mapping. The specific expression of the norm depends on the specific situation and the specific matrix group. In Section 9.1.3, we present a case analysis.

Once the space formed by the matrix group has measurement, i.e., a form of distance, then for a specific problem, as long as the measurement is reasonable, we can use this measure to solve many problems. We extend the most basic classification and clustering algorithm in machine learning problems to the matrix group space, and the distance between the two matrices is represented by the correlation measure.

Algorithm 9.1: KNN based on matrix group

Input: Known c categories of n matrix samples $\{x_1, x_2, \dots, x_n\}$, m unknown class samples $\{y_1, y_2, \dots, y_m\}$, number of neighbours k .

Output: The specific category of each sample in the unknown sample set $\{y_1, y_2, \dots, y_m\}$.

- (1) Take a sample y_i from a sample of m unknown categories by a certain method.
- (2) Calculate the distance $\rho(y_i, x_j)$ between y_i and $\{x_1, x_2, \dots, x_n\}$ in all samples.
- (3) According to step (2), to get the ascending order, take the first k smallest metrics corresponding to $\{x_1, x_2, \dots, x_k\}$.
- (4) Take the category with the largest number of samples from $\{x_1, x_2, \dots, x_k\}$, assigning the value to the sample y_i .
- (5) Repeat the above steps until the sample in each unknown category sample set $\{y_1, y_2, \dots, y_m\}$ is processed.

Algorithm 9.2: K-means based on matrix group**Input:** Matrix sample set $\{x_1, x_2, \dots, x_n\}$, category number k .**Output:** Category assignment for each sample in the sample set $\{x_1, x_2, \dots, x_n\}$.

- (1) In the matrix space corresponding to the sample set, k sample centres $\{z_j^1, z_j^2, \dots, z_j^n\}$ are randomly generated, where the subscript of z represents the j -th iteration, and the superscript value are $1, \dots, n$, which represent the number of centres.
- (2) Take the distance $\rho(x_i, z_j)$ from a sample x_i of the sample set $\{x_1, x_2, \dots, x_n\}$ to each sample centre $\{z_j^1, z_j^2, \dots, z_j^n\}$.
- (3) According to k distance obtained by step (2), select the centre with the shortest distance as the category of x_i .
- (4) Repeat the above two steps until each sample in $\{x_1, x_2, \dots, x_n\}$ is processed.
- (5) According to the above steps, the category assignment of the sample set $\{x_1, x_2, \dots, x_n\}$ is obtained; then, recalculate the centre $\{z_{j+1}^1, z_{j+1}^2, \dots, z_{j+1}^k\}$ of each category.
- (6) If all the centres corresponding to $\{z_{j+1}^1, z_{j+1}^2, \dots, z_{j+1}^k\}$ and $\{z_j^1, z_j^2, \dots, z_j^k\}$ are equal, the algorithm terminates; otherwise, repeat steps (1) to (5).

Both Algorithm 9.1 and Algorithm 9.2 use the theory of matrix group, which is convenient for extending the algorithm of the traditional vector space to the matrix group space. Using this idea, we can also construct additional algorithms of the matrix sample set. Moreover, from the aspects of time complexity and spatial complexity, there is no additional overhead.

9.1.4 Cases analysis

This section demonstrates simple applications of the two algorithms described in Section 9.1.2 through a few examples. In the first experiment, the KNN algorithm is applied to the classification of matrix samples. In the second experiment, the K-means algorithm is applied to the clustering problem of matrix samples.

9.1.4.1 KNn matrix sample classification

First, three types of matrix samples are generated manually, and each sample is a random sample under a two-dimensional Gaussian distribution. However, the mean of each Gaussian sample is different. Hence, the sample is a subset of the $M(1, 2)$ matrix group. The mean values of the three samples are taken in two dimensions, $(0, 0)$, $(2, 2.5)$, and $(4, 4)$, respectively, and the variances are all $q = 1; 200$. Samples of each type are considered, of which 100 samples are used for training and the remaining 100 are used for testing. There are 600 matrix samples in total. The distribution of 300 training samples is shown in Figure 9.1.

Next, we perform Algorithm 9.1 for 300 test samples, where the parameter is $k = 5$ with the norm $\rho(A, B) = \|A - B\|_F$, $A, B \in M(1, 2)$. The results of the test samples are shown in Figure 9.2. To observe the category of the test samples more clearly, the train-

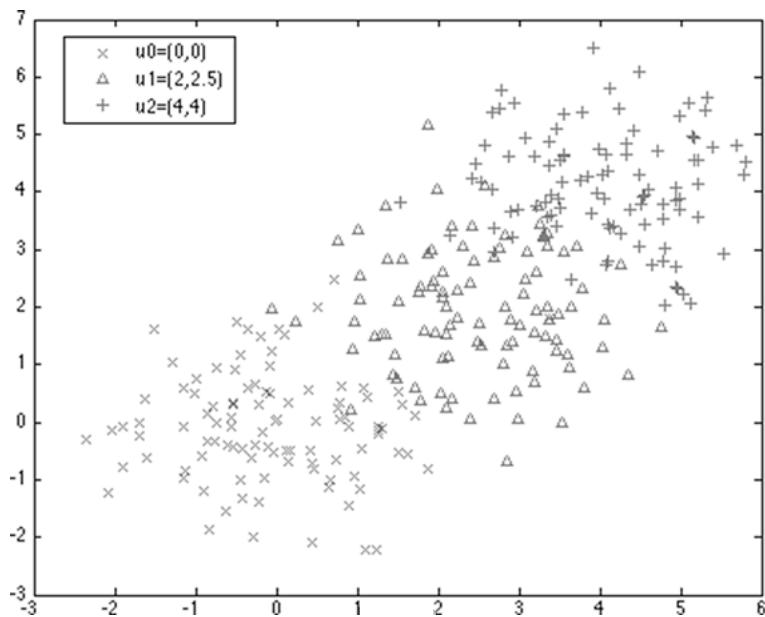


Fig. 9.1: Three sample sets of two-dimensional Gaussian distribution samples with mean values of $(0,0)$, $(2,2.5)$ and $(4,4)$ are denoted by three different symbols and colors, and the number of samples in each category is 100.

ing samples are not displayed; only the test samples are displayed, and the sample for the classification error is marked with a small black circle.

The samples in this experiment belong to a subset of the $M(1, 2)$ matrix group, where the space is R^2 , which is a typical vector space. We can also use the same method for matrix groups of arbitrary dimensions. The space formed by these matrix groups is not necessarily a vector space; it may be a nonlinear manifold space. However, the linear vector space is theoretically one of special circumstances of the streamlines space. Therefore, this chapter focuses on the more adaptable nonlinear manifold space.

9.1.4.2 K-means matrix sample clustering

For matrix sample data visualisation, the K-means matrix clustering experiment is conducted, and the matrix subgroups of the two-dimensional A group are still used. The algorithm is used to cluster the A matrix.

The sample set X in Algorithm 9.2 is instantiated as an A matrix, and the metric function of the matrix group space is still B . The initial mean is also randomly sampled in the A space. The formula for the linear mean space of the linear space is given for

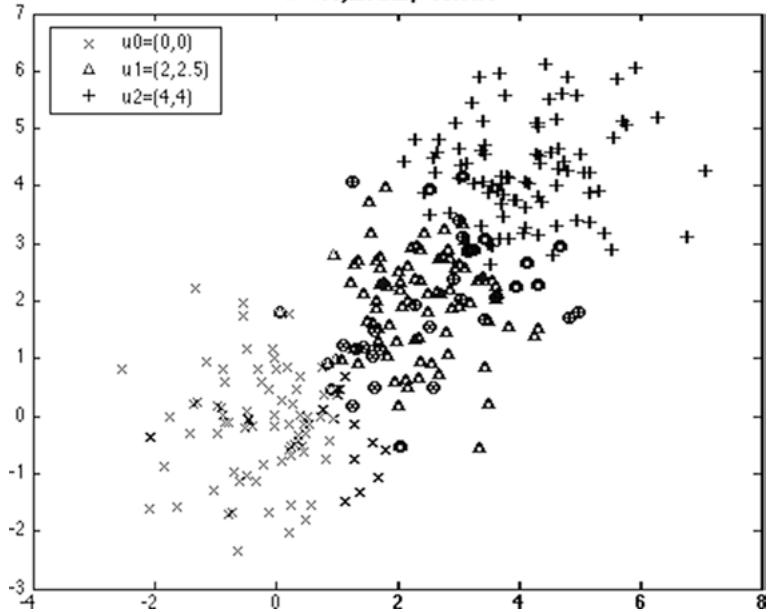


Fig. 9.2: The result of the 5NN classification of the two-dimensional Gaussian distribution sample set generated by the three types of samples in the same parameter configuration.

the new set of mean values for a class:

$$\mu = \frac{1}{n_c} \sum_{i=1}^{n_c} x_i \quad (9.4)$$

where X and N represent the number of samples of category C . Since the space of group A is linear, it is appropriate to use (9.4).

The experiment first generates 500 different Gaussian distributed A samples with different meanings, and the distribution shape is in the form of the letter ‘W’. Then, the 500 samples are clustered separately. From the experimental results shown in Figure 9.3, the result of clustering is consistent with the distribution of the sample in space, i.e., the same species are kept together.

When the actual sample distribution space is not a vector space, but a manifold space, only the new metric function P is designed for the sample of the manifold space, and the algorithm can be applied to the concrete problem. In addition, the feature vector in most learning algorithms is expressed as A , which is a characteristic of B , which shows that the matrix learning algorithm includes the vector learning algorithm in theory. The matrix has a row and column bi-directional structure; the one-way structure of the vector can represent more information. Wang et al. conducted some studies on the matrix sample set [29, 30]. The FDA algorithm extends to the matrix sample.

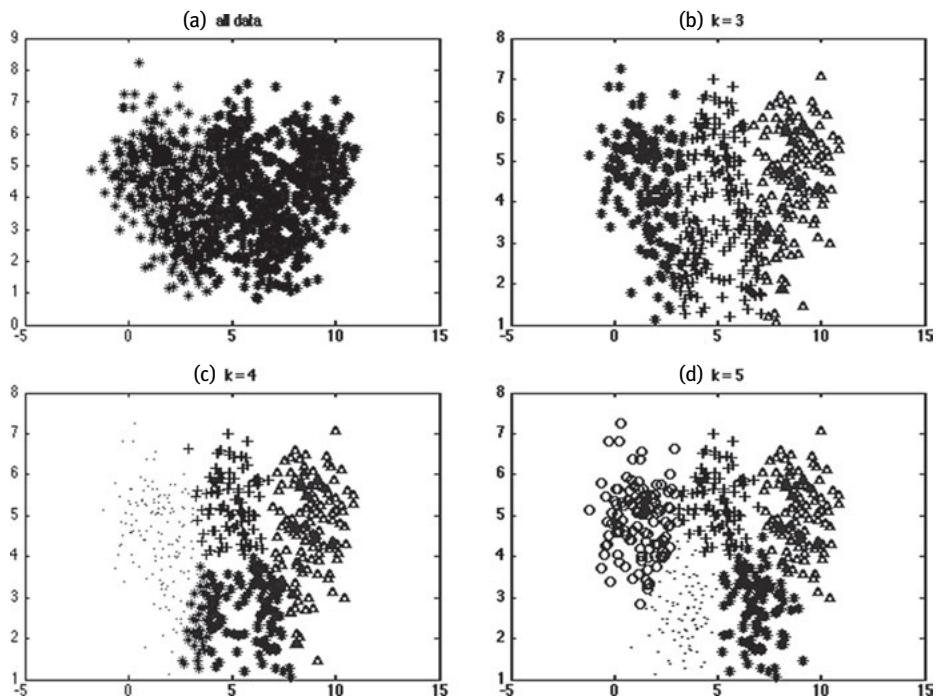


Fig. 9.3: (a) is the contribution of 500 ‘W’ type original samples; (b)–(d) are the results of clustering on the original samples of (a) with setting $k = 1, 2, 3$.

9.2 Gaussian distribution on the Lie group

In this section, we analyse the Gaussian distribution of several commonly used Lie groups in the manifold structure. The Gaussian distribution of the complex Lie group can be found in [73].

A Lie group with a unit value as the mean whose Gaussian distribution is a solution to the thermal equation is defined on the local coordinates:

$$\begin{aligned} \frac{\partial f}{\partial t} &= \Delta f = \text{div}(\text{grad}(f)) \\ &= g^{ij} \left(\frac{\partial^2 f}{\partial x^i \partial x^j} - \Gamma_{ij}^k \frac{\partial f}{\partial x^k} \right) \end{aligned}$$

where g^{ij} is the inverse of the Riemann metric and Γ_{ij}^k is the Christoffel sign [51]. Then, the density function of the Gaussian distribution of R^n is

$$p(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp \left(-\frac{(x - \mu)^T \Sigma^{-1} (x - \mu)}{2} \right). \quad (9.5)$$

The above equation is the solution of the thermal equation in R^n .

9.2.1 Gaussian distribution of R^+

For the Lie group R^+ composed by a positive real number in multiplication, the local coordinates are given by the logarithm, and the solution of the thermal equation is a logarithmic density function:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma x} \exp\left(-\frac{(\log(x) - \log(\mu))^2}{2\sigma^2}\right). \quad (9.6)$$

Given the independent and identical logarithmic distribution of the sample $x_1, \dots, x_n \in R^+$, the maximum likelihood estimates of their mean and variance are

$$\hat{\mu} = \left(\prod_{i=1}^n x_i \right)^{\frac{1}{n}}, \quad \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (\log(x_i) - \log(\hat{\mu}))^2$$

where $\hat{\mu}$ is the geometric mean, i.e., the point where the sum of squares of the distance of the geodesic in R is the smallest, i.e., the smallest point of $\sum_{i=1}^n \log(\mu^{-1}x_i)^2$.

9.2.2 Gaussian distribution of $SO(2)$

The parameter of $SO(2)$ is the rotation angle $\theta \in [0, 2\pi]$. Since $SO(2)$ is isomorphic to S^1 through the mapping $\theta \mapsto e^{i\theta}$, the mean μ and variance σ of the Gaussian distribution on $SO(2)$ are given by

$$p(\theta) = \frac{1}{\sqrt{2\pi}\sigma} \sum_{k=-\infty}^{+\infty} \exp\left(-\frac{(\theta - \mu - 2k\pi)^2}{2\sigma^2}\right). \quad (9.7)$$

The above equation is the solution to the thermal equation in the $[-\pi, \pi]$ cycle, also known as the ‘wrapped Gaussian’, as shown in Figure 9.4.

Now, we derive the maximum likelihood estimates of the mean and variance. Given the independent set of samples $\theta_i \in [0, 2\pi], i = 1, \dots, n$ distributed in the same form as in (9.7), we assume that $\sigma = 1$. Then, the maximum likelihood estimate for the mean is:

$$\hat{\mu} = \arg \max_{\mu \in [0, 2\pi]} \prod_{i=1}^n \frac{1}{\sqrt{2\pi}} \sum_{k=-\infty}^{+\infty} \exp\left(-\frac{(\theta_i - \mu - 2k\pi)^2}{2}\right) \quad (9.8)$$

Note that the two exponential functions are even functions, and their derivative functions are singular; for fixed k , we have

$$\frac{\partial}{\partial \mu} \exp\left(-\frac{(\theta_i - \mu - 2k\pi)^2}{2}\right) = -\frac{\partial}{\partial \mu} \exp\left(-\frac{(\theta_i - \mu + 2k\pi)^2}{2}\right)$$

Hence, when the form of (9.8) is simply reduced to $k = 0$, the maximisation problem becomes

$$\hat{\mu} = \arg \max_{\mu \in [0, 2\pi]} \prod_{i=1}^n \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\theta_i - \mu)^2}{2}\right). \quad (9.9)$$

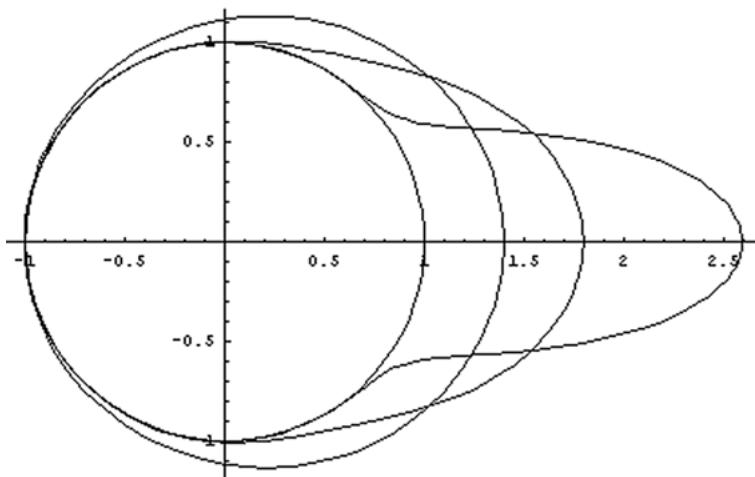
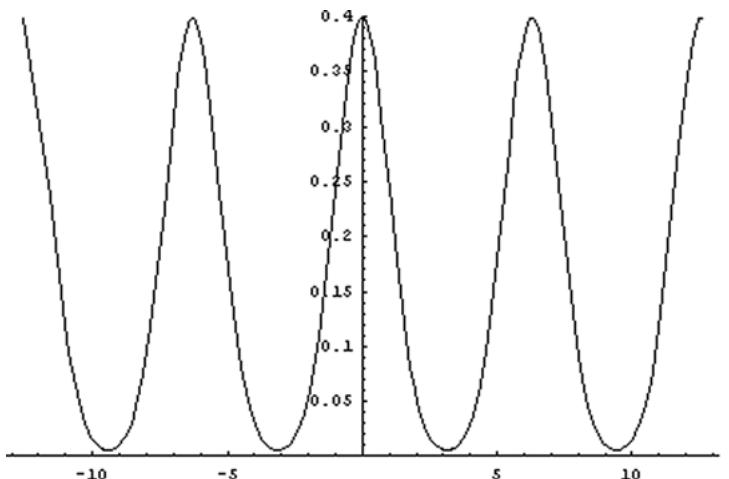


Fig. 9.4: Solution of the thermal equation and the wrapped Gaussian.

The above equation is the maximum likelihood equation of the mean of $\text{SO}(2)$ under a Gaussian distribution. Hence, we have $\hat{\mu} = 1/n \sum_{i=1}^n \theta_i$. Because of the diversity of θ_i representations, such as θ_i can belong to the interval $[0, 2\pi]$ or belong to $[-\pi, \pi]$, which makes the formula about $\hat{\mu}$ ambiguous, we need to limit θ_i to one of the intervals in a specific problem.

For the derivation of the maximum likelihood estimate of variance, consider the following log likelihood function:

$$l(\sigma; \hat{\mu}, \theta_1, \dots, \theta_n) = \sum_{i=1}^n \log \left(\frac{1}{\sqrt{2\pi}\sigma} \right) + \log \left(\sum_{k=-\infty}^{+\infty} \exp \left(-\frac{(\theta_i - \hat{\mu} - 2k\pi)^2}{2\sigma^2} \right) \right)$$

The following is obtained by the differential σ :

$$\frac{\partial l}{\partial \sigma} = -\frac{n}{\sigma} + \frac{1}{\sigma^3} \sum_{i=1}^n (\theta_i - \hat{\mu})^2 + \frac{1}{\sigma^3} \sum_{i=1}^n \left(\frac{\sum_{k=-\infty}^{+\infty} (2k\pi)^2 \exp\left(-\frac{(\theta_i - \hat{\mu} - 2k\pi)^2}{2\sigma^2}\right)}{\sum_{k=-\infty}^{+\infty} \exp\left(-\frac{(\theta_i - \hat{\mu} - 2k\pi)^2}{2\sigma^2}\right)} \right).$$

The third term on the right-hand side of the above equation converges, but it does not produce a closed form solution. According to [52], we can see the wrapped Gaussian density function, i.e., this is consistent with the angle constraints of the image processing problem when $\sigma^2 \leq 2\pi$ approximates the sum term of $k = 0$ in (9.7). Hence, we keep only the term $k = 0$. We then have

$$\frac{\partial l}{\partial \sigma} = -\frac{n}{\sigma} + \frac{1}{\sigma^3} \sum_{i=1}^n (\theta_i - \hat{\mu})^2.$$

Let the above equation be equal to 0; solving x , we can get the maximum likelihood estimate of the approximate variance:

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (\theta_i - \hat{\mu})^2 \quad (9.10)$$

9.2.3 Gaussian distribution of SO(3)

Similar to SO(2), we use a logarithmic mapping on S3 to define a wrapped Gaussian distribution with mean μ . It should be noted that the density function is not necessarily a solution to the thermal equation on SO(3). Let $\mu(x) = \Phi(\log(\mu^{-1}x)) \in R^3$, where $\Phi: SO(3) \rightarrow R^3$, $\Phi(A_v) = v$ form a standard isomorphism. Let $\bar{\mu}(x) = \mu(x)/\|\mu(x)\|$; according to (9.13), the wrapped Gaussian density function of SO(3) is

$$p(x) = \frac{1}{\sqrt{(2\pi)^3 |\Sigma|}} \sum_{k=-\infty}^{+\infty} \exp\left(-\frac{1}{2} (\mu(x) - 2k\pi\bar{\mu}(x))^T \sum^{-1} (\mu(x) - 2k\pi\bar{\mu}(x))\right) \quad (9.11)$$

The covariance structure is defined as the square form on Lie algebra SO(3), represented as a 3×3 matrix Σ .

Given the independent set of samples $x_1, \dots, x_n \in SO(3)$ distributed in the same form as in (9.11), the maximum likelihood estimates for the mean and variance are derived below. For the mean, without loss of generality, it can be assumed that the covariance is a unit matrix. The joint density is given by the density

$$p(\mu; x_1, \dots, x_n) = \prod_{i=1}^n \frac{1}{\sqrt{(2\pi)^3}} \sum_{k=-\infty}^{+\infty} \exp\left(-\frac{1}{2} \|\mu(x_i) - 2k\pi\bar{\mu}(x_i)\|^2\right). \quad (9.12)$$

Note that the geodesic of SO(3) is isomorphic to SO(2), and the density function (9.11) constrains a geodesic to the wrapped Gaussian on SO(2). It is now possible to use the

same assumptions as in the previous section to denote the derivative function of the elimination on the sum formula corresponding term. Maximising $p(\mu; x_1, \dots, x_n)$ is equivalent to finding $\hat{\mu}$ in the following formula:

$$\hat{\mu} = \arg \max_{\mu \in SO(3)} \prod_{i=1}^n \frac{1}{\sqrt{(2\pi)^3}} \exp \left(-\frac{1}{2} \|\mu(x_i)\|^2 \right). \quad (9.13)$$

After taking the logarithm of the above formula, we can get the mean of the maximum likelihood estimate:

$$\hat{\mu} = \arg \max_{\mu \in SO(3)} \sum_{i=1}^n \|\Phi(\log(\mu^{-1}x_i))\|^2 \quad (9.14)$$

where $\|\Phi(\log(\mu^{-1}x_i))\|$ is the Riemannian distance from μ to x . Thus, the maximum likelihood estimate of the mean is also the point at which the square of the geodesic distance is the smallest of all the sample points, i.e., the inner mean value of $SO(3)$.

Similar to $SO(2)$, we assume that the variance is sufficiently small, i.e., $\lambda \leq 2\pi$, where λ is the eigenvalue of Σ . Using the same assumption as in the previous section, the maximum likelihood estimate of the approximate variance is obtained:

$$\widehat{\Sigma} = \frac{1}{n} \sum_{i=1}^n \Phi(\log(\hat{\mu}^{-1}x_i)) \Phi(\log(\hat{\mu}^{-1}x_i))^T. \quad (9.15)$$

9.3 Calculation of the Lie group inner mean value

Figure 9.5(a) shows the inner mean value μ on manifold S^1 , where the distance from each data point to μ is the geodesic distance (curve length) on S^1 and the resulting mean value is also on S^1 . Figure 9.5(b) shows the average of the Euclidean distance calculated directly. The calculated μ is called the extrinsic mean, which is generally not on the manifold, as in the case where μ is in R^2 and not on S^1 .

In the space of R^d , the average of the n data point sets $\{x_i\}_1^n \in R^d$ can be calculated by (9.4), where the sample x_i can be a vector or a matrix, and the μ obtained can keep

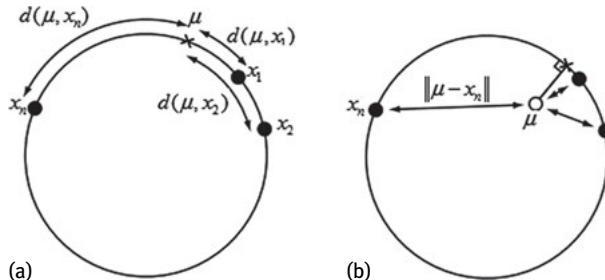


Fig. 9.5: Distribution of the Lie group inner value.

the sum of the Euclidean distances from each point in $\{x_i\}_1^n$, i.e., μ can be expressed as

$$\mu = \arg \min_{x \in R^d} \sum_{i=1}^n \|x - x_i\|^2 . \quad (9.16)$$

Since a d -dimensional manifold M^d is usually not a vector space, the actual distance between the two points on the manifold cannot be represented by the Euclidean distance. Hence, (9.16) cannot be directly applied to the manifold. The result is shown in Fig. 9.5. However, the d -dimensional manifold M^d is locally homologous to R^d , and it is possible to find a homeomorphism mapping $\Phi: M^d \rightarrow R^d$ in the local domain of M^d . Embedding the local part of M^d into R^d , such that $\mu_\Phi: \mu_\Phi = \arg \min_{x \in M^d} \sum_{i=1}^n \|\Phi(x) - \Phi(x_i)\|^2$ can be calculated using (9.16) in R^d . At the same time, we need to define a mapping $\pi: R^d \rightarrow M^d: \mu = \pi(\mu_\Phi)$ so that μ_Φ can be mapped back to M^d . By omitting μ_Φ , we can get

$$\mu = \pi \left(\arg \min_{x \in M^d} \sum_{i=1}^n \|\Phi(x) - \Phi(x_i)\|^2 \right) . \quad (9.17)$$

In practice, it is difficult to find the appropriate Φ and π for M^d , and M^d is not necessarily embedded in R^d by Φ . A more reasonable approach is to use the Riemann distance on M^d instead of the Φ -function in (9.17) to calculate the distance between any $x_i, x_j \in M^d$ directly on M^d , rather than in Euclidean space.

$$\mu = \bar{x} = \arg \min_{x \in M^d} \sum_{i=1}^n d(x, x_i)^2 . \quad (9.18)$$

In the above formula, $d(\cdot, \cdot)$ represents the geodesic distance between the two points in the parentheses. The shortest distance between any two elements X_1 and X_2 is defined as $d(X_1, X_2) = \|\log(X_1^{-1}X_2)\|$. By substituting it into (9.18), we can get

$$\mu = \arg \min_{x \in G} \sum_{i=1}^n \|\log(x^{-1}x_i)\|^2 . \quad (9.19)$$

Using the first-order BCH formula to approximate the right-hand side $\|\cdot\|$

$$\log(x^{-1}x_i) = -\log(x) + \log(x_i) . \quad (9.20)$$

To obtain a more accurate result, we can find the former n -order approximation, but according to the BCH formula, the calculation of the latter approximation becomes very large, and the contribution to the overall approximation is small. Hence, the first-order approximation is used to replace $\log(x^{-1}x_i)$, and the internal mean formula can be simplified as:

$$\begin{aligned} \hat{\mu} &= \arg \min_{x \in G} \sum_{i=1}^n \|-\log(x) + \log(x_i)\|^2 \\ &= \arg \min_{x \in G} \sum_{i=1}^n \|-m + m_i\|^2 . \end{aligned} \quad (9.21)$$

This is the last approximation formula. Now, the problem is to find $x \in G$ that satisfies the above condition. In the literature [2], the process of solving $\hat{\mu}$ is given by the gradient descent method, as follows: design the optimisation function as $f(x) = 1/2n \sum_{i=1}^n d(x, x_i)^2$. The gradient of f is $\nabla f(x) = -1/n \sum_{i=1}^n \log(x^{-1}x_i)$. Given the estimation of the inner mean of the k iteration, the $k + 1$ -iteration formula is $\mu_{k+1} = \exp(\tau/n \sum_{i=1}^n \log(\mu_k^{-1}x_i))$, where τ is the step length. The specific algorithm for the Lie group's inner mean is given by the following algorithm.

Although the gradient descent method converges to the local minimum, which only finds the local minimum and not necessarily the global smallest $\hat{\mu}$, modifying the initial estimate μ_0 and step length τ can have an ideal effect. The choice of τ is related to the structure of the Lie group G , and in the literature [47], it has been proved that it is appropriate for the ball manifold that $\tau = 1$ when the manifold G is a vector space; the gradient descent is equivalent to finding the linear average. If manifold G is R^+ is equivalent to finding the geometric average, and Algorithm 9.3 can converge within a single iteration. For the general case of a Lie group, if Algorithm 9.3 cannot converge when $\tau = 1$, then it is appropriate to take τ as a smaller positive number. According to the characteristics of the gradient descent method, when the value of τ is too large, it may cross the extreme point, and when it is too small, the convergence rate will be too slow.

Algorithm 9.3: Internal mean algorithm for N-class Lie group sample

Input: $\{x_{ij}\}_{j=1}^{j=1...n_i} \in G$, x_{ij} represents the j -th sample of the i -th category distributed over Lie group G , n_i represents the number of training samples in the i -th category, and there are c categories.

Output: $\mu_i, i = 1, \dots, c$, i.e., the internal mean of each category.

Procedure:

Do $i = 1$

$k = 0$

$\mu = x_{i1}$

Do

$\Delta\mu = \tau/n_i \sum_{j=1}^{n_i} \log(\mu^{-1}x_{ij})$

$\mu = \exp(\Delta\mu)$

$k = k + 1$

While $\|\Delta\mu\| > \varepsilon$ and $k < \text{Maximum_Iterations}$

$\mu_i = \mu$

$i = i + 1$

While $i \leq c$

Since the inner mean is more representative of the commonality of a class of things, if an unknown sample is closer to the inner mean of one category than that of the other categories, we believe that this unknown sample is most likely to belong to that category. Hence, we can design a simple classifier of the Lie group based on the inner mean value, i.e., the Lie-mean algorithm, after obtaining the inner mean value of training samples calculated by the category. For each test sample, we only need to calculate the geodesic distance of the sample to various inner mean values corresponding to its

category. The category of the sample is determined as the category of the inner mean within the shortest distance:

$$i^* = \arg \min_{i=1,\dots,c} \|\log(\mu_i^{-1}x)\|. \quad (9.22)$$

Algorithm 9.3 is the calculating method of the Lie-mean based on iteration, and it is suitable for a Lie group with a complex geometric structure. It is inconvenient to solve the inner mean value directly. By taking the characteristics of other Lie group manifolds into account, we can compute a more accurate mean using mathematical methods. For example, in the literature [22], any two symmetric positive definite matrices $S_1, S_2 \in Sym^+(n)$ are given to maintain affine invariant measures:

$$d(S_1, S_2) = \left\| \log \left(S_1^{-\frac{1}{2}} S_2 S_1^{-\frac{1}{2}} \right) \right\|. \quad (9.23)$$

After forming the Lie group structure by adding the operation of (9.3), the distance between the two elements can be expressed by the following formula:

$$d(S_1, S_2) = \|\log(S_2) - \log(S_1)\|. \quad (9.24)$$

We can get the new mean calculation formula of the $Sym^+(n)$ group directly:

$$\mu = \exp \left(\frac{1}{n} \sum_{i=1}^n \log(x_i) \right). \quad (9.25)$$

The above formula is called the log-Euclidean mean. The difference between Algorithm 9.3 and the log-Euclidean mean is as follows: (1) The log-Euclidean mean is the global mean. From the geometric viewpoint, the log-Euclidean mean calculates the geometrical average, which is calculated when all the group elements are mapped to the Lie algebraic linear space, and Algorithm 9.3 calculates the local mean, which is the inner mean of the sample set calculated directly in the Lie group manifold space. (2) The calculation of the log-Euclidean mean is performed once, and the time complexity is $O(n)$. However, Algorithm 9.3 is an iterative algorithm. The number of iterations depends on the specific manifold structure and the given parameters. When the number of iterations is k , the time complexity of Algorithm 9.3 is $O(kn)$.

When (9.25) is used as the mean calculated method, the category discrimination of the Lie-mean algorithm can be obtained as follows:

$$i^* = \arg \min_{i=1,\dots,c} \|\log(\mu_i) - \log(x)\|. \quad (9.26)$$

9.4 Lie-mean learning algorithm

On the basis of the Lie group mean theory discussed in the previous section, this section further designs the learning algorithm related to the mean. In Section 9.4.1, we

present the analysis of the classical linear discriminant algorithm, i.e., the FLDA algorithm. In Section 9.4.2, we analyse the concept and significance of Fisher projection in Lie group space according to the idea of the FLDA algorithm. In Section 9.4.3, we derive two forms of the Lie–Fisher algorithm and analyse their advantages and disadvantages as well as temporal and spatial complexity.

9.4.1 FLDA algorithm

The Fisher linear discriminant analysis (FLDA) algorithm, also known as the LDA algorithm, FDA algorithm, or Fisher linear discriminant, is a classical discriminant analysis classification algorithm in linear space [41]. The standard Fisher projection projects a sample point onto a hyperplane or line that is lower than the sample dimension, so that the ratio between the intra-class divergence and inter-class divergence within each class is the largest. The criterion function of Fisher is

$$J(v) = \frac{v^T S_b v}{v^T S_w v} = \frac{|\tilde{S}_b|}{|\tilde{S}_w|} \quad (9.27)$$

where S_b and S_w are the sum of the intra-class divergence and the sum of the inter-class divergence between the various samples before projection, respectively. Furthermore, \tilde{S}_b and \tilde{S}_w are the sum of the intra-class divergence and the sum of the inter-class divergence between various samples after projection, respectively, defined as follows:

$$\tilde{S}_b = \sum_{i=1}^c n_i (\tilde{\mu}_i - \tilde{\mu}) (\tilde{\mu}_i - \tilde{\mu})^T, \quad \tilde{S}_w = \sum_{i=1}^c \sum_{j=1}^{n_i} (y_{ij} - \tilde{\mu}_i) (y_{ij} - \tilde{\mu}_i)^T$$

where c is the number of categories, n_i and $\tilde{\mu}_i$ represent the number of samples of the i -th classification and the mean after projection, respectively, $\tilde{\mu}$ is the mean of the total samples after projection, and y_{ij} is the result of sample x_{ij} after projection. The problem of Fisher projection is to find v to maximise the value of (9.27). In Euclidean space, the direction of the vector v represents the direction of the line that needs to be projected.

9.4.2 Fisher mapping in Lie group space

The generator of Lie algebra can be used to generate a geodesic on a Lie group manifold. For example, the element $v \in g$ in Lie algebra corresponding to Lie group G is generated by a single parameter exponent mapping: $R \mapsto G: \exp(tv), t \in R$. A geodesic H_v can be generated on Lie group manifolds G . H_v is a single parameter subgroup of Lie group G , and it is uniquely determined by the algebraic element v of Lie group G :

$$H_v \triangleq \{\exp(tv) \in G, t \in R\}$$

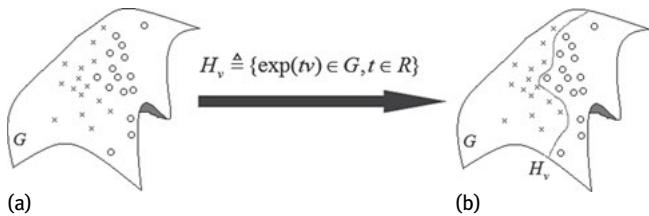


Fig. 9.6: The solution to finding the geodesic line in Lie group manifold space: (a) Two types of samples on the distribution of Lie group G , represented by forks and circles, respectively, are separated by their search for a geodesic H_v ; (b) the geodesic H_v , found by an algorithm, is a single parameter subgroup of G determined by v .

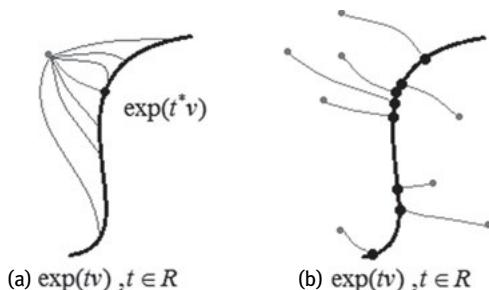


Fig. 9.7: Finding a projection point: (a) Connect a sample point on the Lie group to a point on the geodesic line $\exp(tv)$, and the point-to-point connection forms a geodesic line. The point, the shortest length of all such geodesic lines (in the diagram, it is the geodesic line that connects the dots on $\exp(tv)$) corresponding to a point on $\exp(tv)$, is the projection point $\exp(t^*v)$. (b) Put all the sample points projecting to the geodesic line in the same way.

Given a definite parameter value t , it also determines a point on the geodesic line H_v , and this point also belongs to G . Hence, the solution to finding a geodesic line in the Lie group manifold space is shown in Figure 9.6.

Then, we need to define the concept of the points on a Lie group projection to a geodesic. Set a sample $x \in G$ and a geodesic H_v on G . First, find

$$t^* = \arg \min d(x, \exp(tv)), t \in R.$$

Then, $\exp(t^*v)$ is the projection point of the sample point x to H_v . From a geometric point of view, this projection point is the closest geodesic distance point of all the points on H_v to x , as shown in Figure 9.7.

Then, let $y = \exp(t^*v)$ be the projection point of x on H_v . Then, a class of samples $\{x_{ij}\}_{j=1}^{n_i}$ is projected onto H to obtain a new point set: $\{y_{ij}\}_{j=1}^{n_i}$, the inner mean of each new sample, is calculated after projection:

$$\tilde{\mu}_i = \arg \min \sum_{j=1}^{n_i} d(y, y_{ij}) \quad (9.28)$$

and the inner mean of the total samples is

$$\tilde{\mu} = \arg \min_{\tilde{\mu}} \sum_{i=1}^c \sum_{j=1}^{n_i} d(y, y_{ij}) . \quad (9.29)$$

In the above formula, c represents the number of sample categories, n_i represents the number of samples in the i -th category, and y_{ij} represents the new sample after the j -th sample in the i -th category.

For multiple categories, the divergence between classes after projection is defined as

$$\begin{aligned} \tilde{S}_b &= \sum_{i=1}^c n_i \cdot d^2(\tilde{\mu}_i, \tilde{\mu}) \\ &= \sum_{i=1}^c n_i \cdot \|\log(\tilde{\mu}^{-1} \tilde{\mu}_i)\|^2 . \end{aligned} \quad (9.30)$$

The interclass divergence of a class of samples after projection is defined as $\tilde{S}_i^2 = \sum_{j=1}^{n_i} d^2(y_{ij}, \tilde{\mu}_i)$. Then, the sum of all \tilde{S}_i^2 is defined as the divergence of the total samples:

$$\begin{aligned} \tilde{S}_w &= \sum_{i=1}^c \tilde{S}_i^2 \\ &= \sum_{i=1}^c \sum_{j=1}^{n_i} d^2(y_{ij}, \tilde{\mu}_i) . \end{aligned} \quad (9.31)$$

Substituting (9.28) to (9.31) into (9.27) and approximating by the first-order BCH formula, the following form is obtained:

$$J(v) \approx \frac{\sum_{i=1}^c n_i \cdot \|\log(\tilde{\mu}_i) - \log(\tilde{\mu})\|^2}{\sum_{i=1}^c \sum_{j=1}^{n_i} \|\log(y_{ij}) - \log(\tilde{\mu}_i)\|^2} . \quad (9.32)$$

Correspondingly, the problem of the Lie–Fisher algorithm is reduced to finding a geodesic H_{v^*} satisfying the following condition:

$$v^* = \arg \max J(v) . \quad (9.33)$$

9.4.3 Lie–Fisher discriminant analysis

The process of Lie–Fisher discriminant analysis is to find a classification geodesic on the Lie group manifold. After all the samples are projected onto the geodesic line, the set of projection points of the sample has the maximum ratio between intra-class divergence and interclass divergence.

9.4.3.1 First form of the Lie–Fisher algorithm

In the previous section, the Fisher projection direction of the samples in the nonlinear space of the Lie group was deduced on the basis of the Fisher projection theory and the calculation formula was given. In practice, however, it is difficult to solve (9.32) directly because the function $J(v)$ and the parameter v do not have an explicit analytic expression. Therefore, we map the sample points on the Lie group to the corresponding Lie algebra space, and give the Lie–Fisher projection geodesic in the linear Lie algebra space:

First, we map the sample points on the Lie group to the corresponding Lie algebra space, thus forming a new sample set, and calculate the components

$$\mu = \frac{1}{n} \sum_{i=1}^c \sum_{j=1}^{n_i} \log(M_{ij}) \quad (9.34)$$

and the overall mean of new sample set and the mean of the various classes:

$$\mu_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \log(M_{ij}) . \quad (9.35)$$

According to the above two formulas, the standard Fisher method is used to solve the intra-class divergence

$$S_b = \sum_{i=1}^c n_i \left(\frac{1}{n_i} \sum_{j=1}^{n_i} \log(M_{ij}) - \frac{1}{n} \sum_{i=1}^c \sum_{j=1}^{n_i} \log(M_{ij}) \right) \left(\frac{1}{n_i} \sum_{j=1}^{n_i} \log(M_{ij}) - \frac{1}{n} \sum_{i=1}^c \sum_{j=1}^{n_i} \log(M_{ij}) \right)^T \quad (9.36)$$

and interclass divergence

$$S_w = \sum_{i=1}^c \sum_{j=1}^{n_i} \left(\log(M_{ij}) - \frac{1}{n_i} \sum_{j=1}^{n_i} \log(M_{ij}) \right) \left(\log(M_{ij}) - \frac{1}{n_i} \sum_{j=1}^{n_i} \log(M_{ij}) \right)^T . \quad (9.37)$$

Based on the above calculation formula, the projection direction v is finally obtained. In this way, the approximate formula for $\hat{J}(v)$ is obtained:

$$\begin{aligned} \hat{J}(v) = & \frac{v^T \sum_{i=1}^c n_i \left(\frac{1}{n_i} \sum_{j=1}^{n_i} \log(M_{ij}) - \frac{1}{n} \sum_{i=1}^c \sum_{j=1}^{n_i} \log(M_{ij}) \right)}{v^T \sum_{i=1}^c \sum_{j=1}^{n_i} \left(\log(M_{ij}) - \frac{1}{n_i} \sum_{j=1}^{n_i} \log(M_{ij}) \right)} \\ & \cdot \frac{\left(\frac{1}{n_i} \sum_{j=1}^{n_i} \log(M_{ij}) - \frac{1}{n} \sum_{i=1}^c \sum_{j=1}^{n_i} \log(M_{ij}) \right)^T v}{\left(\log(M_{ij}) - \frac{1}{n_i} \sum_{j=1}^{n_i} \log(M_{ij}) \right)^T v} . \end{aligned} \quad (9.38)$$

For the two types of classification problems, we can directly obtain (assuming that S_w is reversible)

$$\begin{aligned} v = & S_w^{-1} (\mu_1 - \mu_2) \\ = & \left(\sum_{i=1}^2 \sum_{j=1}^{n_i} \left(\log(M_{ij}) - \frac{1}{n_i} \sum_{j=1}^{n_i} \log(M_{ij}) \right) \left(\log(M_{ij}) - \frac{1}{n_i} \sum_{j=1}^{n_i} \log(M_{ij}) \right)^T \right)^{-1} \\ & \cdot \left(\frac{1}{n_1} \sum_{j=1}^{n_1} \log(M_{1j}) - \frac{1}{n_2} \sum_{j=1}^{n_2} \log(M_{2j}) \right) . \end{aligned} \quad (9.39)$$

In summary, the following algorithm can be obtained.

Algorithm 9.4: Finding Lie–Fisher projection geodesic (1)

Input: $\{M_{ij}\}_{\substack{j=1 \dots n_i \\ i=1 \dots c}} \in G$, M_{ij} represents the j -th sample of the i -th category, n_i represents the number of training samples in the i -th category, and there are c categories in total.

Output: the Lie algebra v corresponding to Lie–Fisher projection geodesic H_v .

Procedure:

- (1) For each Lie group sample M_{ij} , perform Lie algebra mapping: $x_{ij} = \log(M_{ij})$.
- (2) Calculate the mean of each new sample $\mu_i = 1/n_i \sum_{j=1}^{n_i} x_{ij}$, $i = 1, 2, \dots, c$, and overall mean $\mu = 1/n \sum_{i=1}^c \sum_{j=1}^{n_i} x_{ij}$, $i = 1, 2, \dots, c$.
- (3) Calculate S_b and S_w : $S_b = \sum_{i=1}^c n_i (\mu_i - \mu)(\mu_i - \mu)^T$, $S_w = \sum_{i=1}^c \sum_{j=1}^{n_i} (x_{ij} - \mu_i)(x_{ij} - \mu_i)^T$
- (4) Find $S_b v = \lambda S_w v$. Then, we can obtain v , which can generate the Lie–Fisher geodesic H_v according to $\exp(tv)$, $t \in R$.

According to the above algorithm, to obtain v , the category of test samples T can be determined by the following formula:

$$t^* = \arg \min_{i=1 \dots c} \|v^T \log(T) - \tilde{\mu}_i\| \quad (9.40)$$

where $\tilde{\mu}_i = 1/n_i \sum_{j=1}^{n_i} v^T \log(M_{ij})$ is the outer mean of the i th category of samples of the Lie group corresponding to the Lie algebra space after projection in the direction v .

9.4.3.2 Second form of the Lie–Fisher algorithm

Algorithm 9.4 maps all samples on the Lie group to the corresponding Lie algebra space and then solves the outer mean in the linear Lie algebra space to calculate v . However, when the discrete low-dimensional samples on the Lie group are logarithmically mapped, solving the mean value may cause the calculation error of the actual sample mean to increase.

Therefore, we can consider obtaining the inner mean μ of the total samples and the inner mean μ_i of various categories in the Lie group, and then mapping the result into the Lie algebra space and solving v . For example, we can use $\mu^{-1}\mu_i$ to remove the overall mean for each category, and remove the intra-class mean value for all i class samples according to $\mu_i^{-1}M_{ij}$. Then, we map the above calculation result into the Lie algebra space by logarithm mapping and solve v . Thus, we obtain $S_b = \sum_{i=1}^c n_i \log(\mu^{-1}\mu_i) \log(\mu^{-1}\mu_i)^T$, $S_w = \sum_{i=1}^c \sum_{j=1}^{n_i} \log(\mu_i^{-1}M_{ij}) \log(\mu_i^{-1}M_{ij})^T$. Thus, another approximation for $\hat{J}(v)$ can be obtained:

$$\hat{J}(v) = \frac{v^T \sum_{i=1}^c n_i \log(\mu^{-1}\mu_i) \log(\mu^{-1}\mu_i)^T v}{v^T \sum_{i=1}^c \sum_{j=1}^{n_i} \log(\mu_i^{-1}M_{ij}) \log(\mu_i^{-1}M_{ij})^T v}. \quad (9.41)$$

For the two types of classification problems, we can directly obtain (assuming that S_w is reversible):

$$v = \left(\sum_{i=1}^c \sum_{j=1}^{n_i} \log(\mu_i^{-1} M_{ij}) \log(\mu_i^{-1} M_{ij})^T \right)^{-1} \log(\mu_1^{-1} \mu_2) . \quad (9.42)$$

In summary, we obtain the new Lie–Fisher algorithm:

Algorithm 9.5: Find Lie–Fisher projection geodesic

Input: $\{M_{ij}\}_{i=1 \dots c}^{j=1 \dots n_i} \in G$, M_{ij} represents the j -th sample of the i -th category, n_i represents the number of training samples in the i -th category, and there are c categories in total.

Output: the Lie algebra v corresponding to Lie–Fisher projection geodesic H_v .

Procedure:

- (1) Using Algorithm 9.4, calculate the inner mean μ of all Lie group samples M_{ij} and the inner mean value μ_i , $i = 1, \dots, c$ of each category.
- (2) Calculate S_b and S_w :

$$S_b = \sum_{i=1}^c n_i \log(\mu^{-1} \mu_i) \log(\mu^{-1} \mu_i)^T, \quad S_w = \sum_{i=1}^c \sum_{j=1}^{n_i} \log(\mu_i^{-1} M_{ij}) \log(\mu_i^{-1} M_{ij})^T .$$

- (3) Find $S_b v = \lambda S_w v$. Then, we can obtain v , which can generate the Lie–Fisher geodesic H_v according to $\exp(tv)$, $t \in R$.
-

For testing the Lie group samples T , the category discriminant formula is as follows:

$$i^* = \arg \min_{i=1 \dots c} \left\| \log \left((\exp(v^T \log(T)))^{-1} \exp(v^T \log(\mu_i)) \right) \right\| . \quad (9.43)$$

Considering the computational problem, it is clear that (9.43) can also be approximated by the first-order BCH formula, so that the discriminant formula can be obtained:

$$i^* = \arg \min_{i=1 \dots c} \|v^T (\log(\mu_i - \log(T)))\| . \quad (9.44)$$

9.4.3.3 The difference between the two forms of the Lie–Fisher algorithm

From the derivation of the two Lie–Fisher forms, regardless of the distribution characteristics of the Lie group samples, Algorithm 9.4 first maps all of them by logarithm mapping, which maps the Lie group samples to the Lie algebra linear space, and then uses the linearity of the Lie algebra space to calculate the mean and the direction of the projection. The main difference between Algorithm 9.5 and Algorithm 9.4 is that the calculation of the mean is not done in the Lie algebra space, but in the nonlinear space of the Lie group.

We can see that Algorithm 9.4 has the following advantages by direct analysis.

- (1) It is computationally efficient. Map the sample to Lie algebra space only in the first step, which requires one logarithm mappings for all the samples, so that the overhead is slightly larger, but the subsequent steps are performed in the linear space and are more efficient.

- (2) It has a high discriminative accuracy for low-dimensional Lie group samples. Because it first maps the sample into Lie algebra space, when the dimension of the Lie group sample is higher, it is more difficult for the Euclidean distance of the Lie algebra space to represent the difference of the sample.
- (3) It has a high discriminative accuracy for samples distributed in flat Lie group manifolds. When the geometric structure of the Lie group manifold is more flat and closer to the hyperplane, the sample of the Lie algebra mapped by the Lie group sample will be more capable of maintaining equidistant and isometric features than the original sample.

Correspondingly, Algorithm 9.4 has the following disadvantages:

- (1) It may increase the mean computational error of high-dimensional samples;
- (2) It may degrade the sample accuracy with uneven distribution within the local neighbourhood of the Lie group manifold.

The shortcomings and advantages of Algorithm 9.4 are the advantages and disadvantages of Algorithm 9.5. If we use Algorithm 9.3 to calculate the mean value in Algorithm 9.5, it is possible that the local minimum value problem caused by the gradient descent method in Algorithm 9.3 will make the originally dominant Algorithm 9.5 in the Lie group manifold disadvantageous. For example, the classification accuracy of Algorithm 9.4 may be reduced for samples with uneven distribution within the local neighbourhood of the Lie group manifold. However, Algorithm 9.5 can deal with the problem well, but if the algorithm is trapped in the local minima of the uneven manifold owing to the calculation error of the inner mean value, then the advantages of Algorithm 9.5 are not reflected.

For the intra-mean calculation of Algorithm 9.5, the specific situation can be analysed in detail. For example, in addition to Algorithm 9.3, we can use the log-Euclidean mean of (9.25) instead of the inner mean, which is a better choice for the covariance feature of the image in terms of the calculation time $O(n)$. Moreover, the calculated mean value gives Algorithm 9.5 a higher classification accuracy.

9.5 Nuclear learning algorithm of the Lie group

This section is organised as follows. Section 9.5.1 introduces the principle of the SVM algorithm. Section 9.5.2 analyses the principle of the KFDA algorithm. Section 9.5.3 introduces the kernel function. Section 9.5.4 analyses the metric function of the matrix Lie group and derives the RBF, linear, etc., 5-matrix Lie group kernel function. Section 9.5.5 discusses the design kernel of Lie group, i.e., KLieDA, based on the kernel function of the Lie group derived in the previous section.

9.5.1 Principle of the SVM algorithm

A support vector machine (SVM) [55, 56] is a machine learning method based on statistical theory proposed by Vapnik. The basic classification idea can be summarised as follows: (i) SVM aims at analysing linearly separable situation cases. For the linear indivisible case, the low-dimensional input samples are mapped into the high-dimensional feature space by the nonlinear mapping function, which makes it linearly separable, so that the samples are classified by using the linear algorithm in this high-dimensional space. (ii) Based on the theory of structural risk minimisation, the optimal segmentation hyperplane is constructed according to the maximum classification interval in the feature space, so that the learner is globally optimised. The expectation risk of the whole sample space satisfies a certain upper bound.

The goal of SVM is to construct an objective function to distinguish two types of patterns or samples as much as possible according to the principle of structural risk minimisation. Next, we analyse the SVM in the linearly separable and linearly inseparable cases.

9.5.1.1 Analysis of SVM in linearly separable cases

The n -dimensional sample set in the linearly separable case means that there is always a hyperplane so that the training samples are completely separated, which can be described as $v \cdot x + b = 0$, where v is the n -dimensional vector and b is the offset. The optimal segmentation hyperplane maximises the distance between the hyperplane and the vector of a class of data closest to this hyperplane. This problem can be solved by solving the following quadratic optimisation problem:

$$\min \Phi(v) = \frac{1}{2} \|v\|^2 \quad (9.45)$$

satisfying the constraints $y_i(v \cdot x_i + b) \geq 1, i = 1, 2, \dots, n$. In the case of a large number of features, this quadratic programming problem can be transformed into its dual problem

$$\max V(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \quad (9.46)$$

satisfying constraints

$$\sum_{i=1}^n \alpha_i y_i = 0, \alpha_i \geq 0, \quad i = 1, 2, \dots, n \quad (9.47)$$

$$v^* = \sum_{i=1}^n \alpha_i y_i x_i \quad (9.48)$$

$$b^* = y_i - v \cdot x_i \quad (9.49)$$

where $\alpha = (\alpha_1, \dots, \alpha_n)$ is the Lagrange multiplier, v^* is the normal vector of the optimal hyperplane, and b^* is the offset of the optimal hyperplane. In (9.49), the solution

must satisfy

$$\alpha_i \{y_i(v \cdot x + b) - 1\} = 0, \quad i = 1, 2, \dots, n. \quad (9.50)$$

From (9.47), the samples of $\alpha_i = 0$ have no effect on the classification, and only the samples of $\alpha_i > 0$ can affect the classification. These samples support the optimal segmentation hyperplane, and these samples are called support vectors. At this point, we can get the final classification function as

$$f(x) = \sum_{i=1}^n y_i \alpha_i (x \cdot x_i) + b^*. \quad (9.51)$$

The category attribution of the unknown sample is determined according to the symbol of $f(x)$.

9.5.1.2 Analysis of SVM in linearly inseparable cases

When the sample set is linearly inseparable, the sample set can be mapped to the high-dimensional feature space F , and the inner product can be realised by using the function satisfying a certain condition of the original space in F . The nonlinear classification problem of the original space is transformed into the linear classification problem of the target space. According to the relevant theory of functional analysis, as long as a kernel function satisfies the Mercer condition [59], it corresponds to the inner product in a certain space. Therefore, simply selecting the appropriate inner product function can solve this linearly inseparable problem. The objective function at this time is

$$\max V(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K(x_i \cdot x_j) \quad (9.52)$$

and its corresponding classification function is

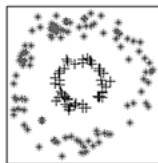
$$f(x) = \sum_{i=1}^n y_i \alpha_i K(x \cdot x_i) + b^*. \quad (9.53)$$

9.5.2 The principle of KFDA

The kernel method [54] is a very useful computational theory that was originally used in SVM [55, 56]. It is widely used in existing linear algorithms, such as the kernel-based KPCA algorithm [57], kernel-based KICA algorithm [58], and KFDA algorithm. Next, we focus on the KFDA algorithm [31].

Because the FDA algorithm can only deal with linearly separable data, the classification performance is greatly degraded for other non-linear distributions or unknown complex distributions. For example, for the classification of the six distribution samples in Figure 9.8, the linear FDA algorithm cannot handle separating the asterisk sample from the plus sample, because it can never find a straight line to these samples after projection to the line while maintaining a good distinction.

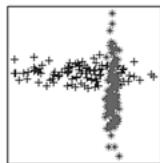
Two Circular Set



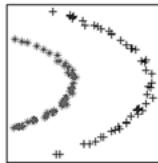
Banana Set



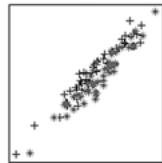
Highleyman Dataset



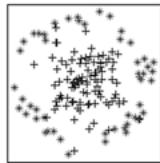
Lithuanian Classes



Difficult Dataset



Circular-Gauss Set

**Fig. 9.8:** 6 kinds of linear indivisible sample set.

From left to right at the top are the data sets of two concentric ring distributions, namely the Banana distribution and Highleyman distribution. From left to right at the bottom are the data sets of the Lithuanian distribution, difficult distribution, and the ring distribution embedding Gaussian distribution.

Thus, these seemingly non-linear samples cannot be directly classified by the linear algorithms in the same dimension of space. Assuming that these samples are linear in a higher-dimensional space, we continue to use the linear algorithm for classification in this high-dimensional space. This is the motivation for the KFDA algorithm.

Suppose that Φ is a non-linear mapping that maps the sample to a higher-dimensional feature space F . To find the direction of projection by linear classification in F , it is necessary to maximise the following equation:

$$J(v) = \frac{v^T S_B^\Phi v}{v^T S_W^\Phi v} \quad (9.54)$$

where $v \in F$, and S_B^Φ and S_W^Φ are corresponding class scatter matrices and intra class scatter matrices in F . For example, in the problems of two category classification, they are defined as follows:

$$S_B^\Phi = (m_1^\Phi - m_2^\Phi)(m_1^\Phi - m_2^\Phi)^T \quad (9.55)$$

$$S_W^\Phi = \sum_{i=1,2} \sum_{x \in \chi_i} (\Phi(x) - m_i^\Phi)(\Phi(x) - m_i^\Phi)^T \quad (9.56)$$

where $m_i^\Phi = 1/l_i \sum_{j=1}^{l_i} \Phi(x_j^i)$, l_i is the number of i -th category samples, x_j^i is the j -th sample of the i -th category, and χ_i is the set of i -th category samples.

9.5.3 Kernel

The dimension of the target feature space F described in Section 9.5.1 can be very high or even infinite, so it is impossible to solve directly. To overcome this difficulty, a similar approach to KPCA and SVM can be used, i.e., without explicitly mapping the data to high-dimensional space, instead of finding a function that directly obtains its dot product ($\Phi(x) \cdot \Phi(y)$) for the training sample. We can use the Mercer kernel to effectively calculate these dot products of points to solve the original problem. Kernel $k(x, y)$ is used to calculate the dot product of the eigenvector in space F , i.e., $k(x, y) = (\Phi(x) \cdot \Phi(y))$. Many of the choices for k have proven to be effective, e.g., the Gaussian RBF kernel $k(x, y) = \exp(-\|x - y\|^2/c)$ in SVM or KPCA, and the polynomial kernel $k(x, y) = (x \cdot y)^d$, where c and d are the corresponding positive numbers.

To calculate the Fisher discriminant projection in the feature space F , we first need a new formula corresponding to (9.55), which replaces the dot product of the input sample with the kernel function. From the theory of regenerative nuclei, any scheme $v \in F$ must be in the space of all training samples in feature space F . Therefore, F has the following form:

$$v = \sum_{i=1}^l \alpha_i \Phi(x_i). \quad (9.57)$$

Using the definition of (9.57) and m_i^Φ , we have

$$\begin{aligned} v^T m_i^\Phi &= \frac{1}{l_i} \sum_{j=1}^{l_i} \sum_{k=1}^{l_i} \alpha_j k(x_j, x_k^i) \\ &= \alpha^T M_i. \end{aligned} \quad (9.58)$$

In the above formula, $M_i = 1/l_i \sum_{k=1}^{l_i} k(x_j, x_k^i)$, and the dot product is replaced by a kernel function. According to (9.55) and (9.58), we can get

$$v^T S_B^\Phi v = \alpha^T M \alpha \quad (9.59)$$

In the above formula, $M = (M_1 - M_2)(M_1 - M_2)^T$. Similarly, using the definition of (9.57) and m_i^Φ , we have

$$v^T S_W^\Phi v = \alpha^T N \alpha \quad (9.60)$$

where $N = \sum_{j=1,2} K_j (I - 1_{l_j}) K_j^T$, K_j is an $l \times l_j$ matrix, and $(K_j)_{nm} = k(x_n, x_m^j)$. Further, $(K_j)_{nm}$ is called the kernel matrix of class j , where I is a unit matrix and 1_{l_j} is a matrix whose all elements are $1/l_j$.

Combining (9.59) and (9.60), the Fisher linear discriminant can be obtained in F

$$J(\alpha) = \frac{\alpha^T M \alpha}{\alpha^T N \alpha} \quad (9.61)$$

where $\alpha = N^{-1}(M_1 - M_2)$.

In this way, we can project the new sample x into v as follows:

$$(v \cdot \Phi(x)) = \sum_{i=1}^l \alpha_i k(x_i, x) . \quad (9.62)$$

Estimating the covariance matrix of l dimension from l samples may make the matrix N ambiguous. For this reason, a multiple of the unit matrix can be simply added, e.g., replacing N with N_μ :

$$N_\mu = N + \mu I . \quad (9.63)$$

The above problem can also be considered in one of the following different forms. (i) It makes the numericalisation of the problem more stable, and for a sufficiently large μ , N_μ will become positive definite. (ii) This method is similar to the method in [60], reducing the number of bases based on the sample eigenvalue estimates. (iii) It reinforces the normalisation of $\|\alpha\|^2$, which is a suitable solution with a small expansion coefficient. Although the real impact on the normalisation problem is unknown, there is indeed a link in the SVM [55]. In addition, there is a method for normalising N , i.e., using the penalty term $\|v^2\|$, which is similar to SVM (plus the full kernel matrix $K_{ij} = k(x_i, x_j)$).

9.5.4 Kernel of the Lie group

For the sample sets of the Lie group matrix, the matrix itself does not have the dot product. However, the result of matrix multiplication in the group operation is still a matrix. Therefore, the vector-based kernel function cannot be used for samples of the Lie group matrix, and we need to design new kernel functions.

9.5.4.1 RBF kernel of the Lie group

The Gaussian RBF kernel function is one of the most frequently used kernel functions. It is also called a radial basis function because it performs a good high-dimensional spatial linear mapping for the nonlinear data set of Figure 9.8. The vector-based RBF kernel function is of the form

$$k(x, y) = \exp\left(\frac{-\|x - y\|^2}{2\sigma^2}\right) . \quad (9.64)$$

In the above equation, the difference between the two vectors x and y is modulo, and there is no matrix subtraction for the general matrix Lie group. However, in the manifold space where the Lie group is located, the component that subtracts another element from one element can be achieved by the group operation and the inverse of the group element, e.g., $x^{-1}y, x, y \in G$. However, in addition to the exchange of the Lie group, the general group of the Lie group is not exchangeable, i.e., $\|x^{-1}y\| \neq \|y^{-1}x\|$. Moreover, for the matrix Lie group, we have $\|\log(x^{-1}y)\| = \|\log(y^{-1}x)\|$. From another

point of view, $\|x - y\|$ is the Euclidean distance between two points x and y in the linear space, and $\|\log(x^{-1}y)\|$ is the geodesic distance between two points in the Lie group manifold space. The geodesic distance $\|\log(x^{-1}y)\|$ from x to y is equal to the geodesic distance $\|\log(y^{-1}x)\|$ from y to x . In addition, the geodesic distance between any two Lie group elements is nonnegative, which satisfies the requirement that kernel matrix K is a symmetric positive definite matrix.

According to the above analysis, we can get the Lie group matrix RBF kernel function

$$k(x, y) = \exp\left(-\frac{\|\log(x^{-1}y)\|^2}{2\sigma^2}\right). \quad (9.65)$$

In the above equation, the nonlinear Lie group samples are mapped into infinite-dimensional space and the dot product is calculated. If the distribution of the samples in the original Lie group manifold space is in accordance with the radial characteristics, then using the kernel function will result in an ideal sample linearization result. For the fractional molecular part of (9.65), approximation of the RBM kernel function can be obtained by using the first-order BCH formula:

$$k(x, y) = \exp\left(-\frac{\|\log(y) - \log(x)\|^2}{2\sigma^2}\right). \quad (9.66)$$

9.5.4.2 Lie group polynomial kernel

The form of the polynomial kernel in the vector feature space is $k(x, y) = (x \cdot y + 1)^d$. The vector dot product is exchangeable, but the matrix multiplication is not exchangeable. Therefore, the vector-based polynomial kernel function cannot be used directly for matrix Lie group samples.

From matrix theory, we can define an inner product in the space of an $n \times n$ matrix.

$$\langle x, y \rangle = \text{tr}(y^H \cdot x) \quad (9.67)$$

where tr is a matrix trace operator and $y^H = \bar{y}^T$, which means that y^H is the conjugate transpose matrix of y :

$$y^H = \begin{pmatrix} \overline{a_{11}} & \dots & \overline{a_{1n}} \\ \vdots & \ddots & \vdots \\ \overline{a_{n1}} & \dots & \overline{a_{nn}} \end{pmatrix} = y^T. \quad (9.68)$$

The inner product of the matrix defined by (9.67) can be used to derive geometric quantities of many matrix spaces. For example, the angle of two matrix vectors in the matrix space can be defined as $\theta = \arccos(\text{tr}(y^H \cdot x) / (\|x\| \cdot \|y\|))$. At the same time, the inner product of the matrix conforms to all the properties of the inner vector product. Therefore, it also satisfies $\langle x, y \rangle = \text{tr}(y^H \cdot x) = \text{tr}(x^H \cdot y) = \langle y, x \rangle$. With this property, it is easy to derive the polynomial kernel function based on the matrix Lie group:

$$k(x, y) = (\text{tr}(y^H \cdot x) + 1)^d. \quad (9.69)$$

In the above equation, d is any positive real number, usually a positive integer.

9.5.4.3 Lie group linear kernel

The linear kernel function of the vector space is $k(x, y) = x \cdot y$. According to the matrix inner product defined by (9.67), the Lie group linear kernel function can be obtained directly:

$$k(x, y) = \text{tr}(y^H \cdot x) . \quad (9.70)$$

9.5.4.4 Lie group perceptron kernel

The most widely used perceptron functions are the sigmoid function and hyperbolic tangent function (\tanh). The form of the hyperbolic tangent function is $\tanh(b(x \cdot y) + c)$, $b > 0$, $c < 0$. Corresponding to (9.67), we get the Lie group perceptron kernel function:

$$k(x, y) = \tanh(b(\text{tr}(y^H \cdot x)) + c) . \quad (9.71)$$

Similarly, the sigmoid function is $1/(1 + e^{-x})$, and we can get the Lie group kernel function in sigmoid form:

$$k(x, y) = \frac{1}{1 + e^{c(\text{tr}(y^H \cdot x))}} . \quad (9.72)$$

In addition to (9.66), using the Lie group geodesic distance, (9.67) and subsequent kernel functions can be used not only for the matrix Lie group but also for general matrix samples.

9.5.5 KLieDA algorithm based on the Lie group kernel function

The KLieDA algorithm based on the Lie group kernel function can be designed using some Lie group kernel functions deduced in Section 9.5.4. The derivation of multi-class questions is similar to the Lie–Fisher algorithm. We present a two-class KLieDA algorithm, i.e., Algorithm 9.6.

Algorithm 9.6: Two-class Lie group KLieDA classification algorithm

Input: $\{x_{ij}\}_{i=1,2}^{j=1 \dots n_i} \in G$, x_{ij} represents the j -th sample of the i -th category, n_i represents the number of training samples in the i -th category.

Output: Output the category attribution for each unknown sample.

Procedure:

- (1) For each Lie group sample x_{ij} , find the means μ_1 and μ_2 of the two classes of Lie group samples. Use Algorithm 9.3 for the general Lie group, and use (9.25) to find the log-Euclidean mean of the covariance sample.
 - (2) Choose the specific Lie group kernel functions, and then obtain $M_i = 1/n_i \sum_{k=1}^{n_i} k(x_j, x_k^i)$ of two types Lie group samples. Then, set $M = (M_1 - M_2)(M_1 - M_2)^T$.
 - (3) Calculate $N = \sum_{i=1,2} K_i(I - 1_{n_i})K_i^T$, where K_i is an $n \times n_i$ matrix and $(K_i)_{nm} = k(x_n, x_m^i)$.
 - (4) Calculate α : $\alpha = N^{-1}(M_1 - M_2)$.
 - (5) Calculate the projection of the two classes of means μ_1 and μ_2 on the vector v in space F : $\tilde{\mu}_1 = (v \cdot \Phi(\mu_1)) = \sum_{i=1}^c \sum_{j=1}^{n_i} \alpha_i k(x_{ij}, \mu_1)$, $\tilde{\mu}_2 = (v \cdot \Phi(\mu_2)) = \sum_{i=1}^c \sum_{j=1}^{n_i} \alpha_i k(x_{ij}, \mu_2)$
 - (6) For unknown x , project it on the vector v in space F : $\tilde{x} = (v \cdot \Phi(x)) = \sum_{i=1}^c \sum_{j=1}^{n_i} \alpha_i k(x_{ij}, x)$. Then, the category attribution is obtained by $i^* = \arg \min_{i=1,2} |\tilde{x} - \tilde{\mu}_i|$.
-

Compared to the traditional KFDA algorithm, Algorithm 9.6 has two innovations in maintaining the same time and space complexity. (i) It uses the matrix Lie group kernel functions, which can handle matrix samples. (ii) During the execution of the algorithm, the mean of the original sample and the projection of the mean in F space are calculated. When the unknown sample is mapped to F -space, it is only necessary to judge the distance from each of the mean points, which requires less computation.

Both the KLieDA algorithm and the Lie–Fisher algorithm can classify nonlinear matrix Lie group samples. From the classification process, KLieDA is, through nonlinearly separable samples, mapped to high-dimensional space, so that they are linearly separable. During the classification period, the Lie group mean accelerates the classification discrimination of unknown samples. The whole classification process is not completed in the Lie group manifold space where the sample is located. The Lie–Fisher algorithm uses the Lie group mean and the projection of the sample to the geodesic to find a classification geodesic in the manifold space where the sample is located to complete the classification of the sample. The distribution characteristics of the original sample directly determine the classification performance of the geodesic. In terms of computational overhead, the KLieDA algorithm needs to calculate the kernel matrix, and the computational time complexity of the kernel matrix is $O(n^2)$, which is obviously larger than the time complexity $O(n)$ of the two divergence matrices calculated by the Lie–Fisher algorithm. Because of the computation of the kernel matrix, the spatial complexity of the KLieDA algorithm also reaches $O(n^2)$. However, the computational complexity of the Lie–Fisher algorithm is only $O(n)$. Therefore, for classification of large sample sets, the KLieDA algorithm is obviously slower than the Lie–Fisher algorithm. From the viewpoint of algorithm application, the Lie–Fisher algorithm classifies a fixed geodesic that has been learned, and the KLieDA algorithm can choose different Lie group kernel functions, according to the characteristics of different sample set distributions, so that the algorithm can achieve better classification results. Therefore, the KLieDA algorithm is more flexible than the Lie–Fisher algorithm in application.

9.6 Case application

In this paper, we introduce two Lie–Fisher algorithms that are applied to the classification experiment of an artificial data set and MNIST handwriting, respectively. At the same time, we design a method to extract Lie group characteristics from handwriting.

9.6.1 Experimental analysis of the Lie–Fisher algorithm

In this section, we will describe two experiments conducted to verify the performance of the Lie–Fisher algorithm. The first experiment is the classification of artificial geo-

metric data sets, and the second is the classification of MNIST handwriting. Because the mean value calculation and classification algorithm in this section are performed in the manifold space of the sample, we will make a comparison with the FLDA algorithm classification in the Euclidean space and Lie-mean classification in the manifold space. The KNN algorithm is used to classify artificial and real data sets, so that the experimental results can reflect the characteristics of the Lie–Fisher algorithm as much as possible. The second experiment first constructs the covariance Lie group characteristic of MNIST handwritten numerals and then uses Algorithm 9.4 and Algorithm 9.5 for classification.

9.6.2 Artificial data set

The classification of geometric figures is of great significance in basic graphics image processing and machine vision. The geometric figures in practical problems may exist in an incomplete form or with large amounts of noise. To address this problem, incomplete form and noise can be simulated by random sampling and Gaussian noise, respectively. Here, we design two basic geometric figures and use the Lie–Fisher algorithm to classify them, and compare them with other methods.

First, we construct two types of data sets by the method of generating a point cloud. The first is the unit square, and the second is the unit circle. Each unit square and each unit circle were randomly taken with n points to form a point cloud respectively, where the n points together form a single sample, rather than a point for a sample. Then, the corresponding variance Gaussian noise was added to these point clouds to form two new samples with noise. As shown in Figure 9.9, 50 circles and 50 squares, i.e., 100 training samples, were constructed using this method.

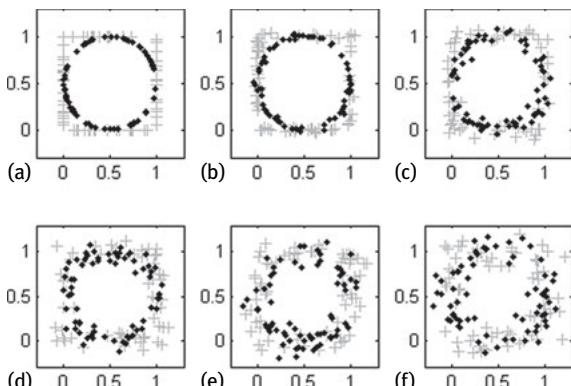


Fig. 9.9: The construction of point cloud.

(a) A point v_i in the original sample with Gaussian noise is represented by the Lie group M_i via a reference vector rv . (b) The method of (a) is used for each point in the sample to extract the Lie group characteristic feature T from a complete sample X .

The construction of the sample set is based on the method of [10], because the point cloud that makes up the sample is in R^2 , and any point in the connection origin and point cloud can form a two-dimensional vector v_i . Transform this vector into a specified reference vector rv . Without loss of generality, we can make $rv = (1, 1)$. For each vector, we use two parameters, namely vector length and vector direction, to express $v_i = (r_i n_i)^T$, where r_i represents the length $\|v_i\|$ of vector v_i , n_i is the direction of v_i , and $\|n_i\| = 1$. Then, the vector v_i is transformed into vector rv , which can be realised by scaling the length of M_i and rotating the direction $v_i = M_i \cdot rv$, where $M_i = \begin{pmatrix} e^{\alpha_i} & 0 \\ 0 & R_i \end{pmatrix}$, $\alpha_i = \|v_i\| / \|rv\|$ is the scaling factor of M_i , and R_i is a rotation matrix in R^2 space, i.e., SO(2). Since rv is fixed, v_i and M_i determine a one-to-one relationship, i.e., each point v_i can be represented by M_i . The sample construction process is shown in Figure 9.10. For a sample of k points, we can construct

$$X = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{pmatrix} = T \cdot \begin{pmatrix} rv \\ rv \\ \vdots \\ rv \end{pmatrix}, \quad T = \begin{pmatrix} M_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & M_k \end{pmatrix}.$$

Now, sample X is represented by T . Since each M_i is a Lie group of structure $R \times \text{SO}(2)$, T is the direct product form of $k M$; hence, T forms a $2k$ -dimensional Lie group. Find Lie algebra elements in Lie group T and obtain

$$\log(T) = \begin{pmatrix} \log(M_1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \log(M_k) \end{pmatrix}, \quad \text{where, } \log(M_i) = \alpha_i \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \theta_i \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

where α_i is given above and $\theta_i = \arccos(v_i \cdot rv / \|v_i\| \cdot \|rv\|)$ is the angle between the vector v_i and the reference vector rv .

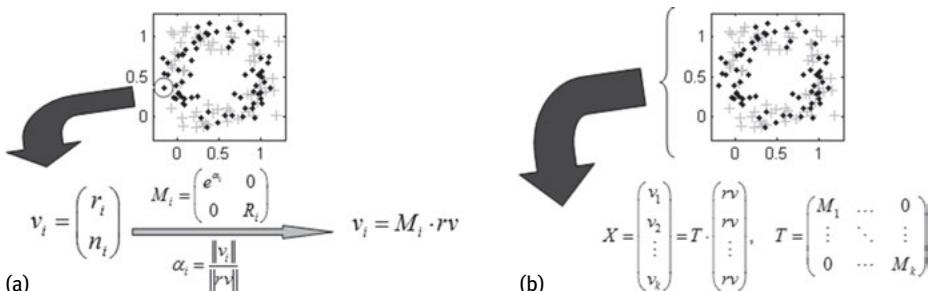


Fig. 9.10: Construction process of samples.

Tab. 9.1: Classification accuracy of each algorithm under different noise (%).

σ^2	Lie-Mean	5-NN	FLDA	Algorithm 9.4	Algorithm 9.5
0.00	100.00	76.67	96.67	100.00	100.00
0.02	94.00	76.00	94.00	100.00	100.00
0.04	91.67	85.33	95.00	100.00	100.00
0.06	85.00	78.67	91.67	95.00	95.00
0.08	84.00	76.67	88.33	93.33	93.33
0.10	82.67	84.67	93.33	91.34	91.33
0.20	80.00	80.00	83.33	90.00	90.00

For each sample in the sample set, the feature T is extracted by the above method, and the Lie–Fisher, FLDA, KNN, and Lie-mean algorithms are used for classification. The results are summarised in Table 9.1.

Table 9.1 shows the classification results of the two classes of samples under different variance Gaussian random noise. The average accuracy of the above data was obtained by five repetitions in the training sample with $n = 100$, $k = 70$, and a test sample of 30. Here, since each M is a 2-dimensional Lie group, T constitutes a Lie group with a dimension of 140. The Lie-mean algorithm is used to calculate the mean of the two types of samples in the Lie group manifold space and classify the test samples according to (9.22). The 5-NN algorithm is used to classify five samples that are nearest to the geodesic distance in the Lie group manifold space. The FLDA algorithm here is similar to the FLDA method of the vector sample. The difference is that the matrix T is used as a linear sample, and the mean value is calculated by $d(X_1, X_2) = \|\log(X_1^{-1}X_2)\|$, i.e., the outer mean of the Lie group sample is used as the mean of the Fisher operation, and the vector form of FLDA is then used to solve the matrix v . The category of the test sample is determined by the distance from the post-projection sample to the centre of each sample after projection: $i^* = \arg \min_{i=1,2,\dots,c} \|v^T T - \tilde{\mu}_i\|$.

As seen from the data in Table 9.1, for the Lie-mean algorithm, the recognition rate decreases as the noise increases. The performances of Algorithm 9.4 and Algorithm 9.5 are nearly the same and obviously better than those of the other algorithms. The algorithm is less sensitive to noise than several other algorithms. It is a relatively stable algorithm with better anti-noise ability. The 5-NN algorithm is less effective than other algorithms.

9.6.3 Handwriting recognition

Unlike previous methods of feature extraction and representation, this section uses the method of Lie group feature representation to extract the Lie group characteristics of shape information of the two-dimensional handwriting image and to classify the handwriting image.

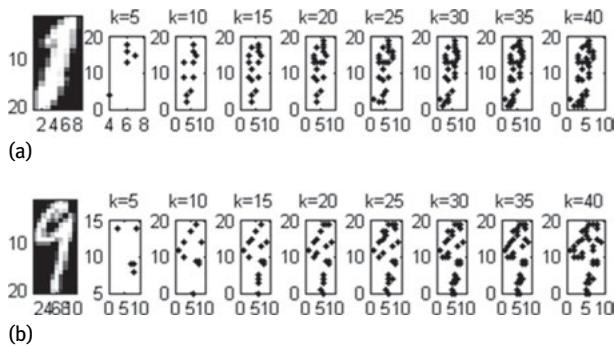


Fig. 9.11: Handwriting Sample Construction (a) Handwriting 1 (b) Handwriting 9.

This experiment uses the MNIST data set [80], each handwriting number of which is a 16×16 greyscale image. In the experiment, handwriting numbers 1 and 9 are classified. First, the original greyscale image is binarised with a threshold grey scale of 100, and then k points are randomly sampled so that k points are located on the pixels representing the handwriting number. As shown in Figure 9.11, the leftmost image is the result of the least effective external rectangle of the original handwriting number. The eight graphs on the right are the point cloud that randomly samples k points after binarisation, i.e., k from 5 to 40; the interval is 5.

First, from the handwriting data sets train 1 and train 9, randomly select 200 samples each, i.e., a total of 400 samples, as the training set. From test 1 and test 9, randomly selected 100 samples each, i.e., a total of 200 samples, as the test set. Finally, using the above method, randomly take k points to form a point cloud, with coordinates sorted from top to bottom and left to right, making the same type of digital image point cloud relatively stable. Construct $k \cdot (R \times SO(2))$ -dimensional Lie group element T for each ordered point cloud using the method described in Section 9.6.1. Using the Lie-mean, Lie group FLDA, Algorithm 9.4, and Algorithm 9.5 for classification, each sample is randomly selected, repeated five times, and the results are shown in Figure 9.12.

Figure 9.12(a) shows the result of not sorting points in the point cloud. It is clear that the Lie-Fisher algorithm is better than the other two algorithms. When $k < 20$, the performance of Algorithm 9.5 is slightly better than that of algorithm 9.4. The Lie-mean algorithm based on the Lie group inner mean is also superior to FLDA. Figure 9.12(b) shows the result of sorting the point cloud. As k increases from 5 to 25, Algorithm 9.4 and Algorithm 9.5 achieve the highest recognition rate of 95%; the performances of the two are very close. When k is less than 40, the performance of Algorithm 9.5 is better than that of Algorithm 9.4 from 0.1% to 1%. When k is more than 40, the classification effects of Algorithm 9.4 and Algorithm 9.5 are basically the same. Overall, the two forms of the Lie-Fisher algorithm are superior to the other two algorithms.

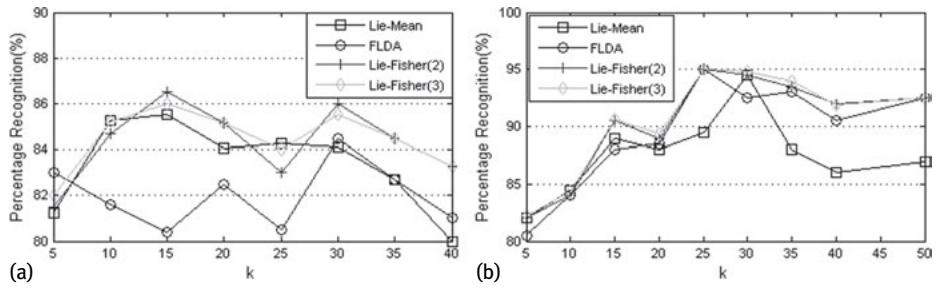


Fig. 9.12: Handwriting classification results of four algorithms with different k values.

9.6.4 Covariance Lie group characteristic of the Lie–Fisher handwriting classification

This section uses the MNIST handwriting data set and the Lie–Fisher algorithm for classification. The difference is that the feature of handwriting extraction is not the point cloud Lie group characteristic but the covariance Lie group characteristic of the image.

9.6.4.1 Image covariance characteristics

A conventional image is a set of grey values or three-dimensional colour components, which can be represented by I , and the attributes of each pixel can be denoted as $I(x, y)$. The method can also be used to represent other image formats, such as infrared images. Use F to represent the image feature extracted from image I ,

$$F(x, y) = \phi(I, x, y) . \quad (9.73)$$

The function ϕ in the above equation can represent any mapping about the image attributes, such as the grey value, the colour channel component value, or the gradient value. The range of $\phi(I, x, y)$ can be expressed as a d -dimensional real space. The local rectangle region R lies in the image I of interest, and $R \subseteq F$. The corresponding ϕ value of all the pixels in the region can be expressed as $\{z_i\}_{i=1}^n$, where z_i is a d -dimensional real vector, and n is the number of pixels in region R . The vector set $\{z_i\}_{i=1}^n$ of the image region is given by the covariance matrix

$$C_R = \frac{1}{n+1} \sum_{i=1}^n (z_i - \mu)(z_i - \mu)^T \quad (9.74)$$

where μ is the mean of the vector set $\{z_i\}_{i=1}^n$.

The regional covariance feature of the image has many excellent features. In particular, the integral image of the features can be used to calculate the covariance characteristics of any sub-region in the image. The classification performance is good, the calculation is fast, the feature dimension is low, and the dimension is not affected by the image size. A detailed description of the image covariance feature can be found in [20].

In this experiment, for the image in the MNIST database, set $R = F$, and $\phi(I, x, y)$ is given by

$$\phi(I, x, y) = \left(x, y, I(x, y), \left| \frac{\partial}{\partial x} I(x, y) \right|, \left| \frac{\partial}{\partial y} I(x, y) \right| \right)^T \quad (9.75)$$

$$\phi(I, x, y) = \left(x, y, I(x, y), \left| \frac{\partial}{\partial x} I(x, y) \right|, \left| \frac{\partial}{\partial y} I(x, y) \right|, \sqrt{\left| \frac{\partial}{\partial x} I(x, y) \right|^2 + \left| \frac{\partial}{\partial y} I(x, y) \right|^2} \right)^T \quad (9.76)$$

$$\phi(I, x, y) = \left(x, y, I(x, y), \left| \frac{\partial}{\partial x} I(x, y) \right|, \left| \frac{\partial}{\partial y} I(x, y) \right|, \left| \frac{\partial^2}{\partial x^2} I(x, y) \right|, \left| \frac{\partial^2}{\partial y^2} I(x, y) \right| \right)^T \quad (9.77)$$

$$\begin{aligned} \phi(I, x, y) = & \left(x, y, I(x, y), \left| \frac{\partial}{\partial x} I(x, y) \right|, \left| \frac{\partial}{\partial y} I(x, y) \right|, \left| \frac{\partial^2}{\partial x^2} I(x, y) \right|, \left| \frac{\partial^2}{\partial y^2} I(x, y) \right|, \right. \\ & \left. \tan^{-1} \left(\left| \frac{\partial}{\partial x} I(x, y) / \frac{\partial}{\partial y} I(x, y) \right| \right) \right)^T \end{aligned} \quad (9.78)$$

In (9.75)–(9.78), $I(x, y)$ is the luminance value at (x, y) , $\partial/\partial x I(x, y)$ is the first-order gradient in the transverse direction at (x, y) , $\partial^2/\partial x^2 I(x, y)$ is the second-order gradient, $\sqrt{|\partial/\partial x I(x, y)|^2 + |\partial/\partial y I(x, y)|^2}$ is the length of first-order gradient vector, and $\tan^{-1}(|\partial/\partial x I(x, y)/\partial/\partial y I(x, y)|)$ is the arc tangent of the ratio of the first-order gradient vector. After modelling, we can get that Z is a 5-dimensional, 6-dimensional, 7-dimensional, and 8-dimensional real vector for the four functions (9.75)–(9.78). For (9.78), the calculated C_R is an 8×8 real symmetric matrix. Since C_R is a symmetric matrix, $\sum_{i=1}^n i|_{n=8} = 36$ dimensions, which are lower than those for many image features. The covariance matrix is a real symmetric positive definite matrix. Many achievements have been made in the study of the spatial structure, metric, mean, and other properties of such matrices [14, 22, 25].

The covariance matrix of the image becomes a tensor space, and each matrix is a point within the space. The distance between any two points C_1, C_2 is given by [53]

$$\rho(C_1, C_2) = \sqrt{\sum_{i=1}^n \ln^2 \lambda_i(C_1, C_2)}, \quad (9.79)$$

and $\{\lambda_i(C_1, C_2)\}_{i=1}^n$ are generalised eigenvalues of C_1, C_2 , which can be calculated by

$$\lambda_i C_i x_i - C_2 x_i = 0, \quad i = 1, \dots, d \quad (9.80)$$

where $x_i \neq 0$.

The vector satisfying $x_i \neq 0$ is a generalised eigenvector. It can be proved that ρ satisfies the axioms of the second chapter about the metric in (9.79). This metric also satisfies the Lie group structure of positive definite matrices, which can be derived from the Lie algebra of positive definite matrices.

9.6.4.2 Covariance characteristics in handwriting recognition

Similar to Section 9.6.4, Handwriting 1 and Handwriting 9 are classified in this section. The covariance sample set C_R constructed from (9.75)–(9.78) is classified by Algorithm 9.4 and Algorithm 9.5, and the measure between the covariance samples in Algorithm 9.5 is given by (9.79). First, from the training set, randomly select 100 training samples of two types, and from the test set, randomly select 100 test samples of two types. The covariance matrix is constructed using (9.75)–(9.78) and then run 10 times for each case. In the experiment, the mean of Algorithm 9.5 is calculated by the log-Euclidean, i.e., (9.25), and Algorithm 9.4 does not change. The results obtained are shown in Figure 9.13.

From the experimental results in Figure 9.13, it can be clearly seen that Algorithm 9.4 has high classification accuracy in most cases by using (9.75), while Algorithm 9.5 and the matrix-based FLDA algorithm are essentially flat using (9.77). However, when using (9.76) and (9.78), the classification accuracy of the two Lie–Fisher algorithms is slightly lower than that of the FLDA algorithm, indicating that the sample distribution at this time is basically linearly separable. When mapping the mean of the sample and Lie group–Lie algebra, the accuracies of the two Lie–Fisher algorithms are reduced, resulting in a decrease in classification accuracy. In Figure 9.13(b), compared to (a), because the first-order gradient vector length operator is used, Algorithm 9.5 becomes very unstable with low classification accuracy, whereas the FLDA algorithm and Algorithm 9.4 are relatively stable with high accuracy. This

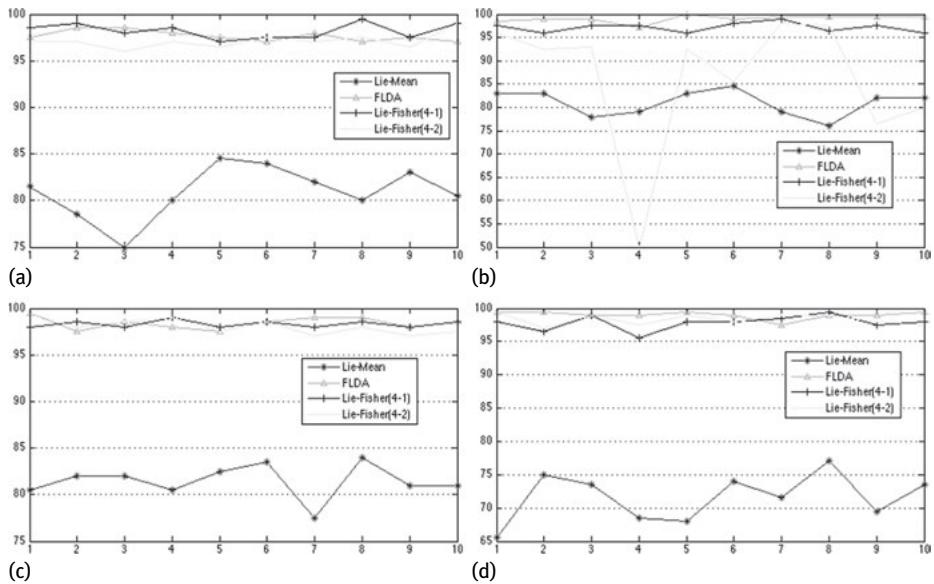


Fig. 9.13: Using the experimental results of (9.47)–(9.50), where the number of iterations is abscissa and the classification accuracy is ordinate.

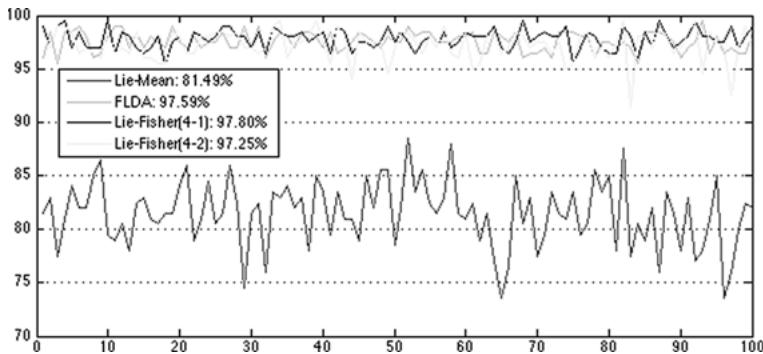


Fig. 9.14: Classification accuracy map using the (9.75) after 100 times to calculate. Abscissa represents the calculation times and vertical coordinates represents the classification accuracy.



Fig. 9.15: Shows incorrect test examples of algorithm 9.4.

is probably because the covariance manifold structure is generally close to linear but locally uneven after adding the operator; hence, Algorithm 9.5 uses log-Euclidean, but the standard Lie group distance $d(X_1, X_2) = \|\log(X_1^{-1}X_2)\|$ is used in the calculation of (9.44), rather than the affine invariant distance (9.23). Thus, the errors generated during the calculation are amplified in the case of when multiple mappings between Lie group and Lie algebra.

In addition, Figure 9.13(a) shows the effect of only using a 5-dimensional vector on z_i , with faster calculation. With the configuration of Intel i5 dual-core CPU (1.6 GHz), 4 GB memory, and MATLAB 2011b, for Algorithm 9.4, the total training and classification time of 200 training samples and 200 test samples is less than 1 second, and the classification accuracy is basically maintained at around 98%, as shown in Figure 9.14. Therefore, Algorithm 9.4 is suitable for applications where the speed requirements are relatively high, such as target tracking and rapid detection. Figure 9.15 shows incorrect test examples of Algorithm 9.4.

9.7 Summary

In this chapter, we started with the geometrical structure of the Lie group and Lie algebra, and tried to find a geodesic line on the Lie group manifold and project Lie group samples to the geodesic line, and maximised the ratio of divergence between classes and divergence within the classes after projection, so as to realise class discrimination

of the nonlinear Lie group space. Then, through the study of matrix theory and the kernel method, combined with Lie group geodesic theory, the kernel function based on the matrix Lie group was designed and realised. Thus, we extended Lie group machine learning to the classification and dimension reduction algorithm based on the kernel method. The Lie-group-based Lie–Fisher and KLieDA algorithms were shown to have high classification accuracy compared with the traditional method. The Lie–Fisher algorithm solves the problem of geodesic classification in Lie group manifold space. KLieDA solves the problem of high-dimensional spatial mapping and classification of nonlinear matrix Lie group samples.

Bibliography

- [1] Kondor R. Group Theoretical Methods in Machine Learning. 2008.
- [2] Fletcher T, Lu C, Joshi S. Statistics of Shape via Principal Geodesic Analysis on Lie Groups. IEEE Conference on Computer Vision and Pattern Recognition, 2003, 1, 95–101.
- [3] Fletcher PT, Lu C, Pizer SM, Joshi S. Principal geodesic analysis for the study of nonlinear statistics of shape. IEEE Transactions on Medical Imaging, 2004, 23(8), 995–1005.
- [4] Bayro CE, Ortegon AJ. Template Tracking with Lie Algebras. IEEE International Conference on Robotics and Automation, 2004, 5, 5183–5188.
- [5] Bayro CE, Ortegon AJ. Lie Algebra Template Tracking. International Conference on Pattern Recognition, 2004, 2, 56–59.
- [6] Fiori S. Unsupervised Neural Learning on Lie Group. International Journal of Neural Systems, 2002, 12(3–4): 219–246.
- [7] Tournier M, Wu X, Courty N, Arnaud E, Reveret L. Motion Compression using Principal Geodesics Analysis. ACM Conference on Computer and Communications Security, 2008, 28(2), 355–364.
- [8] Li G, Liu Y, Yin J, Shi ZL. Projective Tracking Based on Second-order Optimization on Lie Manifolds. Chinese Conference on Pattern Recognition, 2008, 1–6.
- [9] Fletcher PT, Joshi S, Lu C, Pizer S. Gaussian Distributions on Lie Groups and Their Application to Statistical Shape Analysis. International Conference on Information Processing and Medical Imaging, LNCS 2732, 2003, 450–462.
- [10] Yarlagadda P, Ozcanli O, Mundy J. Lie Group distance based generic 3-d vehicle Classification. International Conference on Pattern Recognition, 2008, 1–4.
- [11] Li G, Liu Y, Shi ZL, Yin J. Optimization on Lie Manifolds and Projective Tracking. CSSE, 2008, 1, 768–771.
- [12] Maher M. Means and Averaging in the Group of Rotations. SIAM Journal on Matrix Analysis and Applications, 2002, 24(1), 1–16.
- [13] Fiori S, Tanaka T. An Algorithm to Compute Averages on Matrix Lie Groups. IEEE Transactions on Signal Processing, 2009, 57(12), 4734–4743.
- [14] Fiori S, Tanaka T. Learning Averages over the Lie Group of Symmetric Positive–Definite Matrices. International Joint Conference on Neural Networks, 2009, 828–832.
- [15] Lin DH, Grimson E, Fisher J. Learning visual flows: A Lie algebraic approach. IEEE Conference on Computer Vision and Pattern Recognition, 2009, 747–754.
- [16] Porikli F, Tuzel O, Meer P. Covariance Tracking using Model Update Based on Lie Algebra. IEEE Conference on Computer Vision and Pattern Recognition, 2006, 728–735.

- [17] Chen DQ, Han JQ, Yu ZJ. Visual Tracking Using Region Covariance and Geometric Particle Filtering. International Congress on Image and Signal Processing (CISP), 2010, 1, 381–386.
- [18] Doretto G, Yao Y. Region Moments: Fast invariant descriptors for detecting small image structures. IEEE Conference on Computer Vision and Pattern Recognition, 2010, 3019–3026.
- [19] Tuzel O, Porikli F, Meer P. Learning on Lie Groups for Invariant Detection and Tracking. IEEE Conference on Computer Vision and Pattern Recognition, 2008, 1–8.
- [20] Tuzel O, Porikli F, Meer P. Region Covariance: A Fast Descriptor for Detection and Classification. European Conference on Computer Vision, 2006, 589–600.
- [21] Ando R, Ohki H, Fujita Y. A Comparison with Covariance with Covariance Features on Player Uniform Number Recognition. Korea–Japan Joint Workshop on Frontiers of Computer Vision, 2011, 1–4.
- [22] Pennec X, Fillard P, Ayache N. A Riemannian Framework for Tensor Computing. International Journal of Computer Vision, 2006, 66(1), 41–66.
- [23] Tuzel O, Porikli F, Meer P. Human Detection via Classification on Riemannian Manifold. IEEE Conference on Computer Vision and Pattern Recognition, 2007, 1–8.
- [24] Wang CM, Jascha SD, Ivana T, Bruno AO. Lie Group Transformation Models for Predictive Video Coding. Data Compression Conference, 2011, 83–92.
- [25] Arsigny V, Fillard P, Pennec X, Ayache N. Geometric means in a novel vector space structure on symmetric positive-definite matrices. SIAM Journal on Matrix Analysis and Applications, 2007, 29(1), 328–347.
- [26] Xu M, Rajesh PNR. Learning the Lie Groups of Visual Invariance. Neural Computation, 2007, 19(10), 2665–2693.
- [27] Plebe A. Learning Visual Invariance. The European Symposium on Artificial Neural Networks, 2006, 73–76.
- [28] Cadieu CF, Olshausen BA. Learning Transformational invariants from Natural Movies. Neural Information Processing Systems, 2008, 209–216.
- [29] Wang Z, Chen SC, Liu J, Zhang DQ. Pattern Representation in Feature Extraction and Classifier Design: Matrix Versus Vector. IEEE Transactions on Neural Networks, 2008, 19(5), 758–769.
- [30] Chen SC, Zhu YL, Zhang DQ, Yang JY. Feature extraction approaches based on matrix pattern: MatPCA and MatFLDA. Pattern Recognition Letters, 2005, 26(8), 1157–1167.
- [31] Mika S, Rätsch G, Weston J, Schölkopf B, Müller KR. Fisher discriminant analysis with kernels. Neural Networks for Signal Processing IX, 1999, 41–48.
- [32] Brun A, Westin CF, Herberthson M, Knutsson H. Intrinsic and Extrinsic Means on the Circle – A Maximum Likelihood Interpretation. International Conference on Acoustics, Speech, and Signal Processing, 2007, 3, 1053–1056.
- [33] Drummond T, Cipolla R. Real-Time Visual Tracking of Complex Structures. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002, 24(7), 932–946.
- [34] Dalal N, Triggs B. Histograms of Oriented Gradients for Human Detection. IEEE Conference on Computer Vision and Pattern Recognition, 2005, 1, 886–893.
- [35] Wang QJ, Zhang RB. LPP-HOG: A New Local Image Descriptor for Fast Human Detection. KAMW, 2008, 640–643.
- [36] Dalal N. Finding People in Images and Videos. 2006.
- [37] Stringer SM, Perry G, Rolls ET, Proske JH. Learning invariant object recognition in the visual system with continuous transformations. Biological Cybernetics, 2006, 94(2), 128–142.
- [38] Charles FC, Bruno AO. Learning Transformational Invariants from Time–Varying Natural Images. Neural Information Processing Systems, 2008.
- [39] Hu MK. Visual pattern recognition by moment invariants. IRE Transactions on Information Theory, 1962, 8, 179–187.

- [40] Li FZ, Qian PX. Machine learning theory and application. China University of Science and Technology, 2009.
- [41] Richard OD, Peter EH, David GS. Pattern Classification, Second edition. Wiley, 2000.
- [42] Mundy JL, Zisserman A. Geometric Invariance in Computer Vision. The MIT Press, 1992.
- [43] Hall BC. Lie Groups, Lie Algebras, and Representations: An Elementary Introduction. Springer, 2000.
- [44] Baker A. Matrix Groups An Introduction to Lie Group Theory. Springer, 2002.
- [45] Strang G. Linear Algebra and its Applications, Second edition. Academic Press, 1980.
- [46] Kaaniche MB, Bremond F. Tracking HoG Descriptors for Gesture Recognition. IEEE International Conference on Advanced Video and Signal Based Surveillance, 2009, 140–145.
- [47] Buss SR, Fillmore JP. Spherical averages and applications to spherical splines and interpolation. ACM Transactions on Graphics, 2001, 20, 95–126.
- [48] Lowe DG. Object Recognition from Local Scale-Invariant Features. International Conference on Computer Vision, 1999, 2, 1150–1157.
- [49] Paisitkriangkrai S, Shen CH, Zhang J. Fast Pedestrian Detection Using a Cascade of Boosted Covariance Features. IEEE Transactions on Circuits and Systems for Video Technology, 2008, 18(8), 1140–1151.
- [50] Ge YH, Yu JJ. A Scene Recognition Algorithm Based on Covariance Descriptor. International Conference on Computer and Information Sciences, 2008, 838–842.
- [51] Lee JM. Riemannian Manifolds: An Introduction to Curvature. Springer-Verlag, 1997.
- [52] Mardia KV. Directional Statistics. Wiley, 1999.
- [53] Förstner W, Moonen B. A metric for covariance matrices. Ratio, 1999, 66, 113–128.
- [54] Aizerman M, Braverman E, Rozonoer L. Theoretical foundations of the potential function method in pattern recognition learning. Automation and Remote Control, 1964, 25, 821–837.
- [55] Vapnik V. The nature of statistical learning theory. New York Springer Verlag, 1995.
- [56] Schölkopf B, Burges CJ, Smola AJ. Advances in Kernel Methods: Support Vector Learning. Neuropsychologia, 1999, 49(7), 185–208.
- [57] Schölkopf B, Smola A, Müller KR. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Neural Computation, 1998, 10(5), 1299–1319.
- [58] Cheng J, Liu Q, Lu HQ. Texture classification using kernel independent component analysis. International Conference on Pattern Recognition, 2004, 1, 620–623.
- [59] Saitoh S. Theory of Reproducing Kernels and its Applications. Longman Scientific & Technical, Harlow, England, 1988.
- [60] Friedman J. Regularized discriminant analysis. Journal of the American Statistical Association, 1989, 84(405), 165–175.
- [61] Xu H, Li FZ. Lie group machine Learning's axioms hypothesizes. Proceedings of the 2006 IEEE International Conference on Granular Computing, 2006, 401–404.
- [62] Li FZ. Research on the Lie group machine learning. School of Computer Science and Technology, Suzhou University, Technical Report, 2004.
- [63] Xu H. Lie group machine learning and application research. Soochow University, 2007.
- [64] Xu H, Li FZ. Study on Lie group machine learning. Journal of Computational Information Systems, 2005, 1(4), 843–849.
- [65] Li FZ, Kang Y. Based on Lie group machine learning theory. Journal of Yunnan Nationalities University, 2004, 13(4), 251–255.
- [66] Li FZ, Xu H. The theory framework of Lie group machine learning (LML). Computer Technology and Application, 2007, 1(3), 62–80.
- [67] Xu H, Li FZ. Geometry algorithms of Dynkin diagrams in Lie group machine learning. Journal of Nanchang Institute of Technology, 2006, 25(2), 75–78.

- [68] Xu H, Li FZ. Algorithms of Dynkin diagrams in Lie group machine learning. *Journal of Communication and Computer*, 2006, 4(3), 13–17.
- [69] Xu H, Li FZ. The design of SU(n) classifier of Lie group machine learning (LML) [J]. *Journal of Computational Information Systems*, 2005, 1(4), 835–841.
- [70] Li FZ, Xu H. SO(3) Classifier of Lie group machine learning. *Journal of Communication and Computer*, 2006, 4(3), 76–80.
- [71] He SP. Quantum group classifier research in Lie group machine learning. *Soochow University*, 2008.
- [72] Fu HX. Symplectic classifier research in Lie group machine learning. *Soochow University*, 2008.
- [73] Grenander U. *Probabilities on Algebraic Structures*. Wiley, 1963.
- [74] Ke Y, Sukthankar R. PCA-SIFT: a more distinctive representation for local image descriptors. *IEEE Conference on Computer Vision and Pattern Recognition*, 2004, 2, 506–513.
- [75] Abdel-Hakim AE, Farag AA. CSIFT: A SIFT Descriptor with Color Invariant Characteristics. *IEEE Conference on Computer Vision and Pattern Recognition*, 2006, 2, 1978–1983.
- [76] Bay H, Ess A, Tuytelaars T, Gool LV. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding*, 2008, 110(3), 346–359.
- [77] Morel JM, Yu G. ASIFT: A New Framework for Fully Affine Invariant Image Comparison. *SIAM Journal on Imaging Sciences*, 2009, 2(2), 438–469.
- [78] Mitchell T. *Machine Learning*. McGraw Hill, 1997.
- [79] Hastie T, Tibshirani R, Friedman J. *The Elements of Statistical Learning*. Springer Series in Statistics, 2001.
- [80] Savich AW, Moussa M. Resource Efficient Arithmetic Effects on RBM Neural Network Solution Quality Using MNIST. *International Conference on Reconfigurable Computing and FPGAs*, 2011, 35–40.
- [81] Gao C, Li FZ. Research on Lie group mean learning algorithm. *Pattern Recognition and Artificial Intelligence(Held)*, 2011. Simultaneously, this paper has also been rated by CRSSC-CWI-CGrC2012 as an excellent student thesis.
- [82] Shen C, Li FZ, Gao C, He SP. Lie group color image face tracking. *Computer Science and Exploration(Held)*, 2012.
- [83] Gao C, Li FZ. Lie group nuclear algorithm research. This paper has been submitted to the CRSSC-CWI-CGrC2012 joint meeting.
- [84] Gao C. Lie group mean learning algorithm and application research. *Soochow University*, 2012.

10 Tensor learning

10.1 Data reduction based on tensor methods

In practical applications, many types of data are not only high-dimensional but also involve multiple inputs. Because of this issue, a novel dimensionality reduction method based on multiple linear algebras (tensors) has attracted interest in this area besides traditional methods. Tensor is a term in multiple linear algebras. It is an extension of a vector, and a vector is a first-order tensor. Actually, a high-order tensor can describe multiple input data more naturally. Tensor methods provide a more effective way of dealing with multiple input data.

Existing tensor methods have been applied to pattern recognition, web data mining, textual analysis, signal processing, and so on. Tensor methods in the area of signal processing are still in their nascent stages. However, we already have some applications in data reduction and data mining. Next, we will introduce some popular algorithms based on the tensor method.

10.1.1 GLRAM

Generalised low rank approximation of matrices (GLRAM) [1, 16] aims at the following problem model. There is a matrix set $M_k, I \times J$, such as a set of images. By projecting with the same matrix, we will get a set of low-dimensional isomorphic matrix representations. Similar to the PCA algorithm, it vectorises images. GLRAM tries to keep the essential attributes of the image in matrix form, and the key concept of GLRAM is to find a series of kernel matrices $\widetilde{M}_i \in R^{r_1 \times r_2}, i = 1, \dots, N$ and two projection matrices $L \in R^{I \times r_1}, R \in R^{J \times r_2}$. The column vectors of these two matrices are orthogonal, and we can get the following objective:

$$\begin{array}{ll} \min_{\substack{L \in R^{I \times r_1}, L^T L = I \\ R \in R^{J \times r_2}, R^T R = I \\ \widetilde{M}_i \in R^{r_1 \times r_2}, i = 1, \dots, N}} & \sum_{i=1}^N \|M_i - L \widetilde{M}_i R^T\|_F^2. \end{array} \quad (10.1)$$

We solve this problem iteratively, and (10.1) is equivalent to

$$\phi = \sum_i \|L^T M_i R\|_F^2. \quad (10.2)$$

Through an iterative process, we get L and R . By fixing R we obtain L by minimising ϕ . In the next step, we fix L to obtain R . The algorithm can begin with an initial value of R , and an auxiliary matrix.

$$C_L \equiv \sum_{i=1}^N M_i R R^T M_i^T. \quad (10.3)$$

Here, we choose the eigenvectors corresponding to the greatest r_1 eigenvalues and form $L = [u_1, \dots, u_{r_1}]$. Then, we exchange L and R to obtain the new R . We continue this iterative process until we obtain stable L and R . The kernel matrix \tilde{M}_i can be obtained by solving $\tilde{M}_i = L^T M_i R$, and $M_i \approx L \tilde{M}_i R^T$ is the approximate value of M . Actually, GLRAM is an extension of 2D-PCA. If we view M as the tensor data, GLRAM can be regarded as the prototype of two-order tensor data dimensional reduction. To deal with higher-order tensor forms, we need the tensor decomposition technique.

10.1.2 HOOI

Higher-order orthogonal iteration (HOOI) [2, 17] is a type of data reduction algorithm that mainly aims at third-order tensor data. For example, there is an N -order tensor $A \in R^{I_1 \times I_2 \times \dots \times I_N}$ and K represents N numbers that satisfy $K_n \leq I_n$, ($n = 1 \dots N$). HOOI can obtain a low-dimensional representation of a high-order tensor, which means it can find $U^{(n)} \in R^{I_n \times K_n}$ and $S \in R^{K_1 \times K_2 \times \dots \times K_N}$ satisfying the following equation:

$$\min_{U^{(1)} U^{(2)} \dots U^{(N)}, S} \|A - S \times_1 U^{(1)} \times_2 U^{(2)} \dots \times_N U^{(N)}\|_F^2, \quad U^T U = I. \quad (10.4)$$

For simplicity, the HOOI algorithm only considers third-order tensors. The solving method is similar to GLARM, which obtains the third projection matrix by fixing the other two projection matrices in each iterative step.

Algorithm 10.1: HOOI algorithm

Input: $I \times J \times K$ tensor A , the target dimension K_1, K_2, K_3

Output: $L \in R^{I_1 \times K_1}, R \in R^{I_2 \times K_2}, R \in R^{I_3 \times K_3}, S$

- (1) Set the initial values of R and V . Orthogonalize the column vectors of R and V $C = A \times_2 R^T \times_3 V^T$.
- (2) Do:
- (3) $L = SVD(K_1, C_{(1)})D = A \times_1 L^T \times_3 V^T R = SVD(K_2, D_{(2)})$.
- (4) Until convergence. Solve $S = \varepsilon \times_3 V^T$.

Here, $U = SVD(K_i, C_{(i)})$ represents the singular value decomposition of the k -mode of C . Choose the eigenvectors corresponding to the K_i largest singular values, and constitute U . When the algorithm finally converges, we will obtain three projection matrices $V = SVD(K_3, \varepsilon_{(3)})$, L , R , and V . Because $r_1 < I$, $r_2 < J$, $r_3 < K$, we can reduce the number of dimensions.

10.1.3 2DPCA

Compared with the classical PCA algorithm, 2DPCA [19] is a matrix form of PCA. Set X represents normalised column vectors. The idea of 2DPCA is to project a matrix A ($m \times n$) to X by using a linear transformation $Y = AX$, and a projected eigenvector Y is obtained.

There are c pattern classes: $\omega_1, \omega_2, \dots, \omega_c$. Each class has n_i training sample matrices: A_1, A_2, \dots, A_M . The inner scatter matrix is

$$G = \frac{1}{M} \sum_{i=1}^M (A_i - \bar{A})(A_i - \bar{A})^T, \quad (10.5)$$

where \bar{A} represents the mean matrix of the training sample totality, and we define the criterion function

$$J(X) = X^T G X. \quad (10.6)$$

Maximise this criterion function. The unit vector X is called the optimal projection vector. The physical significance of the maximisation is to maximise the degree of dispersion of the eigenvectors. Actually, the optimal projection vector is the unit eigenvector corresponding to the largest eigenvalue of the scatter matrix [20].

10.1.4 CubeSVD

CubeSVD [14] is an application of high-order singular value decomposition (HOSVD) [21]. It is an algorithm that models high-order heterogeneous data by using tensors. For example, in Web data mining, we construct a tensor model to find the implication relation of users as well as to search terms and webpages, as shown in Figure 10.1.

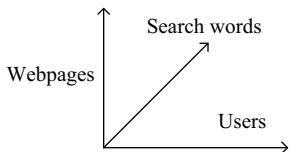


Fig. 10.1: The tensor model which includes users, search words and webpages.

CubeSVD decomposes the tensor that represents the relation of the users, search terms, and webpages. The high-order singular value decomposition of $A \in R^{I_1 \times I_2 \times I_3}$ is given by:

$$A = S \times_1 V_1 \times V_2 \times V_3 \quad (10.7)$$

Here $V_i = (v_1^{(i)}, v_2^{(i)}, \dots, v_{I_i}^{(i)})$, $i = 1, 2, 3$ and each i respectively corresponds to an orthogonal singular vector group. S represents the kernel tensor. In general, all elements of S are nonzero. If we just retain the $k_1 k_2 k_3$ dimensions of the three different directions of S , the tensor A could be approximated to a tensor:

$$A_{(k_1, k_2, k_3)} = S_{(k_1, k_2, k_3)} \times_1 V_{k_1}^{(1)} \times V_{k_2}^{(2)} \times V_{k_3}^{(3)}. \quad (10.8)$$

The reconstructive third-order tensor can represent much hidden information of a mass of raw data [22].

10.1.5 TSA

Tensor subspace analysis (TSA) [9] gives the linear dimensionality reduction method in the second-order tensor subspace. This algorithm regards each image as a $R^{n_1} \otimes R^{n_2}$ two-order tensor. Its problem model can be described as follows. M is a $R^{n_1} \otimes R^{n_2}$ sub-manifold, X is a data set of M . We can find two transfer matrices U and V , which are $n_1 \times l_1$ and $n_2 \times l_2$ respectively, to map the points in the data set into $Y_1, \dots, Y_m \in R^{l_1} \otimes R^{l_2}$ ($l_1 < n_1, l_2 < n_2$). The transformational relationship between Y_i and X_i is $Y_i = U^T X_i V$. TSA naturally describes the relationship between the row vectors and column vectors of an image. We detect the geometric structure of the tensor space by learning a low-dimensionality tensor space.

In the above-mentioned algorithms, GLRAM is a type of matrix algorithm that can be regarded as an approximation to a second-order tensor form. HOOI is a type of approximation algorithm of third-order tensors. 2DPCA is an extension of the PCA in matrix form, and it utilises a two-dimensional matrix to directly represent an image. TSA obtains a linear approximation to the Laplacian-Beltrami operator in a Riemannian manifold by using the tensor method. This method can preserve the local structure of the manifold. The tensor field data reduction model and high-order singular value decomposition described in this chapter both are developed on the basis of HOSVD. We will introduce HOSVD in the next chapter.

10.1.6 Related problem

A tensor field is a normal concept in physics and geometry, which is widely applied to differential geometry, manifold theory, algebraic geometry, the general theory of relativity, and stress analysis of materials [23, 24]. Examples include the contravariant tensor used in differential geometry and stress energy tensor used in physical and engineering analysis. Both of them are related to the general theory of relativity. The vector field can be regarded as a vector that represents a change from one point to another. The tensor field is a generalisation of the vector field. In engineering, many manifolds are usually tensors in a $V \otimes \dots \otimes V \otimes V^* \otimes \dots \otimes V^*$ space that is built by a tensor field in Euclidean 3D space.

Tensor fields are widely studied for diffusion tensor magnetic resonance imaging (DT-MRI, DTI) [19–26]. The DTI image data are different from traditional greyscale and colour medical images. The diffusion tensor field is already known. Every voxel data point in the basic 3D data space is a symmetric and positive definite two-order tensor. In addition, tensor field analytical methods are applied to image structure analysis on the basis of the local structure tensor [27] and material analysis in electromagnetism [28, 29]. Regardless of the application, we need to perform some regularisation processing or filtration processing to reduce the noise. Because the tensor is mathe-

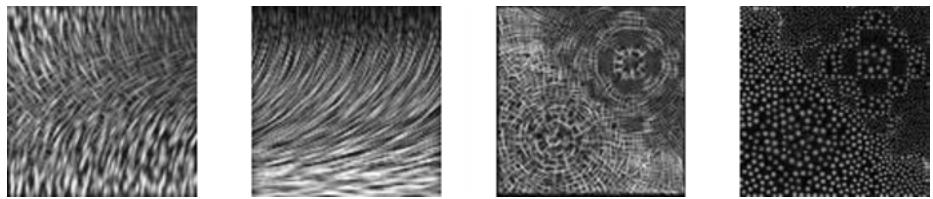


Fig. 10.2: The visual tensor field.

matically a linear function that is associated with multiple vector spaces, its complexity and high-order characteristic make the problem difficult to understand. For solving this problem, many researchers have studied the visualisation of tensor fields [30, 32] (Figure 10.2).

In summary, tensor fields have been widely applied to many research fields. However, in computer information processing, it is still in the preliminary exploration stage. There are many problems that need to be studied further. This chapter takes full advantage of the characteristics of tensor fields and combines them with the tensor decomposition technique and tensor field theory to construct the tensor learning model. We proposed the data reduction algorithm based on tensor fields to provide examples for machine learning applications.

10.2 Data reduction model based on tensor fields

Combined with the related theory of tensor fields, this section discusses the data reduction model and algorithm based on tensor fields. First, we introduce the concept of tensor field in manifolds. Then, we give the data reduction model in tensor fields and discuss the simplified form of data in tensor fields. Next, we describe the design processing of the data reduction algorithm based on tensor fields. Finally, we give an experimental example.

10.2.1 Tensor field on a manifold

Contemporary tensor fields are divided into two steps. First, there is the vector bundle, which is a vector space depending on the parameters, and the parameters constitute a manifold. Second, there is the introduction of the tensor bundle. The concept of the vector bundle is not related to the choice of any base. The usual procedure is to multiply two vector bundles and obtain the tensor bundle. The handling mechanism performs well, and its coordinates are irrelevant. Finally, the tensor field is similar to the vector field that is a cross section of the tensor bundle [23, 24].

For the data objects in data reduction analysis, we abstract them to elements in tensor space to utilise the handling method in tensor fields. The concept of tensor space is as follows:

Definition 10.1. Set V_1, V_2 and W as the vector space. If there is a vector space W and a bilinear map $\otimes: V_1 \times V_2 \rightarrow W$, we have $W = \langle \text{Im} \otimes \rangle = \langle \otimes(V_1, V_2) \rangle$. For any vector space Z and bilinear map $f: V_1 \times V_2 \rightarrow Z$, there is a linear map $g: W \rightarrow Z$ that makes $f = g \circ \otimes$. Then, we can call W the tensor space built by the tensor product of V_1, V_2 .

M is an n -dimensional differentiable manifold (C^∞ manifold). $F(M)$ represents the set of differentiable functions on M and can be regarded as an associative algebra in the mathematics domain.

$D^1(M)$ represents the set of all differentiable vector fields on M , which means that $X \in D^1(M)$ is a transform of $F(M)$. X satisfies the following two conditions:

$$\begin{aligned} X(\alpha f + \beta g) &= \alpha X(f) + \beta X(g), \quad \forall \alpha, \beta \in \mathbb{R}; f, g \in F(M) \\ X(f \cdot g) &= f \cdot X(g) + g \cdot X(f) \end{aligned} \tag{10.9}$$

Here, X is a derivative of $F(M)$. We can define addition in $D^1(M)$ and the product of numbers in R . Then, $D^1(M)$ becomes a linear space in R .

From, the perspective of geometry, if M_p represents the tangent space in point p , the value in point p will be a tangent vector. By setting a coordinate system on manifold M , the element in $D^1(M)$ can be represented by the analytic expression of the coordinate. It is certainly true that the element in $D^1(M)$ can be represented by the analytic expression of the coordinates.

Definition 10.2. $Y(t)$ is a tangent vector of point $y(t)$ on the manifold M , $Y(t) \in T(M)$. If $Y(t)$ relies on t , it will be called the vector field of point $y(t)$.

M is an m -dimensional C^k manifold, $p \in M$, and $T_p \in M$ is the tangent space of point p on m . The (r, s) tensor space in vector space $T_p(M)$ can be represented by $T_{s,p}^r(M)$, and we have:

$$T_s^r(M) = \bigcup_{p \in M} T_{s,p}^r(M) . \tag{10.10}$$

(U, ϕ, x^i) is a coordinate graph on M . $\{(dx^i)_p, 1 \leq i \leq m\}$ is a set of dual basis of $T_p(M)$. Then, $\Phi_p \in T_{s,p}^r(M)$ can be represented as:

$$\Phi_p = \Phi_{j_1 \dots j_s}^{i_1 \dots i_r} \left(\frac{\partial}{\partial x^{i_1}} \right)_p \otimes \dots \otimes \left(\frac{\partial}{\partial x^{i_r}} \right)_p \otimes (dx^{j_1})_p \otimes \dots \otimes (dx^{j_s})_p . \tag{10.11}$$

Here, $\tilde{U} = \{\Phi_p \in T_{s,p}^r(M) | p \in U\}$ and $\tilde{\phi}: \tilde{U} \rightarrow \phi(U) \times R^{m^{s+r}}$.

Hence, we have:

$$\Phi_p \mapsto (x^i(p) \Phi_{j_1 \dots j_s}^{i_1 \dots i_r}), \quad 1 \leq i, i_1, \dots, i_r, j_1, \dots, j_s \leq m . \tag{10.12}$$

For arbitrary open set $A \subset \phi(U)$, $B \subset R^{m^{s+r}}$, we define $\tilde{\phi}^{-1}(A \times B) \subset T_s^r(M)$. Naturally, we have a map:

$$\pi: T_s^r(M) \rightarrow M, \Phi_p \mapsto p . \tag{10.13}$$

Theorem 10.1. M is an m -dimensional C^k manifold. There is a topological structure in $T_s^r(M)$ that makes each coordinate graph (U, p) : $\tilde{U} = \pi^{-1}(U)$ be the open set in $T_s^r(M)$, and $\tilde{\phi}: \tilde{U} \rightarrow \phi(U) \times \mathbb{R}^{m^{s+r}}$ be a homeomorphic structure. The coordinate graphs $\{(\tilde{U}, \tilde{\phi})\}$ determine a C^{k-1} differential structure in $T_s^r(M)$ and makes $T_s^r(M)$ an $m+m^r$ dimensional differential manifold.

$(T_s^r(M), x, M)$ is a (r, s) tensor bundle on M , $T_s^r(M)$ is called a full space, M is called a bottom space, π is called a projection, and $T_s^r(M)$ is called a fibre or the (r, s) tensor bundle on M .

In particular, $\Lambda^r T_p^*(M)$ is the r -order antisymmetric covariant tensor space, and we have:

$$\Lambda^r T^*(M) = \bigcup_{p \in M} \Lambda^r T_p^*(M). \quad (10.14)$$

Here, we have a naturally C^{k-1} differential structure to build a $m+(m/s)$ dimensional C^{k-1} manifold. $\Lambda^r T^*(M)$ is the exterior derivative form bundle of degree r , or form bundle of degree r .

The concept of the generalised vector field is as follows:

Definition 10.3. There is an (r, s) tensor field Φ on C^k manifold M , and the tensor field is a map: $\Phi: M \rightarrow T_s^r(M)$. The tensor flow has $\pi \circ \Phi = I$, where I represents the identity mapping, $\Phi: p \mapsto \Phi_p \in T_{s,p}^r(M)$. A (r, s) tensor field can be called a cross section on the tensor bundle $T_s^r(M)$.

(U, ϕ, x^i) is a coordinate graph on M , and the (r, s) tensor field can be locally represented as

$$\Phi(x) = \Phi_{j_1 \dots j_s}^{i_1 \dots i_r}(x) \frac{\partial}{\partial x^{i_1}} \otimes \dots \otimes \frac{\partial}{\partial x^{i_r}} \otimes dx^{j_1} \otimes \dots \otimes dx^{j_s} \quad (10.15)$$

If $\Phi_{j_1 \dots j_s}^{i_1 \dots i_r}(x)$ is C^k in each coordinate graph, $1 \leq i, i_1, \dots, i_r, j_1, \dots, j_s \leq m$, we can call Φ the C^k tensor field. Obviously, the character of Φ is not related to the local coordinate system. The transformation formula of the component parts of Φ is

$$\Phi_{j_1 \dots j_s}^{i_1 \dots i_r}(x') = \Phi_{k_1 \dots k_s}^{h_1 \dots h_r}(x) \frac{\partial x'^{i_1}}{\partial x^{h_1}} \dots \frac{\partial x'^{i_r}}{\partial x^{h_r}} \frac{\partial x^{k_1}}{\partial x'^{j_1}} \dots \frac{\partial x^{k_s}}{\partial x'^{j_s}}. \quad (10.16)$$

Here the set of smooth (r, s) on M is $F_s^r(M)$.

10.2.2 Reduction model based on the tensor field

In many specific problems, the real data can be obtained easily. However, the hidden information of the data cannot be obtained easily, which means that we need to analyse the data or extract the effective information. The image sample includes multiple concept hierarchies, from texture to abstract semantics. The images are usually represented by points in high-dimensional space, which makes it difficult to extract useful

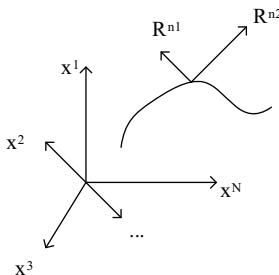


Fig. 10.3: The grey-scale map of tensor space.

information. Actually, the image should be regarded as a tensor product, which means that a greyscale image belongs to tensor space $R^{n_1} \otimes R^{n_2}$. The image space is usually embedded in a $R^{n_1} \otimes R^{n_2}$ sub-manifold [11] (Figure 10.3).

We do not change the tensor representation with the change in the coordinate system when we use a tensor to describe a data object. However, each tensor must consist of some classes. Relative to different coordinate systems, the component parts of the same tensor may be different while the object is the same. The tensor field data should have a definite relation in different coordinate systems. We can now analyse the data reduction model of the tensor field.

X is the input data set that represents the data objects needed to be processed. First, we abstract the tensor properties of the data and obtain the $\Phi(X)$ formula of the coordinate system, for which data set X exists. For obtaining the simplest form of data, we need to find the tensor field, $\Phi'(X)$, which is another coordinate system of tensor field $\Phi(X)$.

The data reduction model of tensor field $\langle X, \psi, \Phi, f, \Phi' \rangle$ is as follows:

X : input data set $X = \{X_1, \dots, X_N\}$;

Φ : Tensor field;

f : Affine transformation function of tensor field;

Φ' : Simple form of tensor field Φ ;

ψ : obtain the tensor field Φ from data set X , $\psi: X \rightarrow \Phi$;

X is the original data object and ψ represents the method that can abstract the tensor field model from the original data. If we treat an image as an element in tensor space $R^{n_1} \otimes R^{n_2}$, each data object X_i should correspond to a tensor element A_i in tensor field Φ .

Here, we introduce the concept of tensor field function.

Definition 10.4. There is an isomorphism tensor in a domain of the space

$$T(r) = T_{k \dots l}^{i \dots j} g_i \dots g_j g^k \dots g^l . \quad (10.17)$$

Then, we call $T(r)$ the tensor function of this domain.

In this section, we discuss the tensor field function, which is not just in the rectangular coordinate system. We extend this problem to any coordinate system (Figure 10.4).

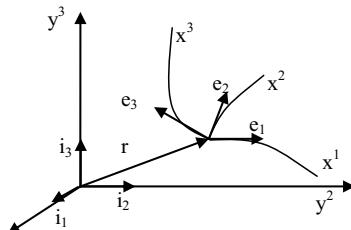


Fig. 10.4: The decompose of one point in tensor space based on a local base vector.

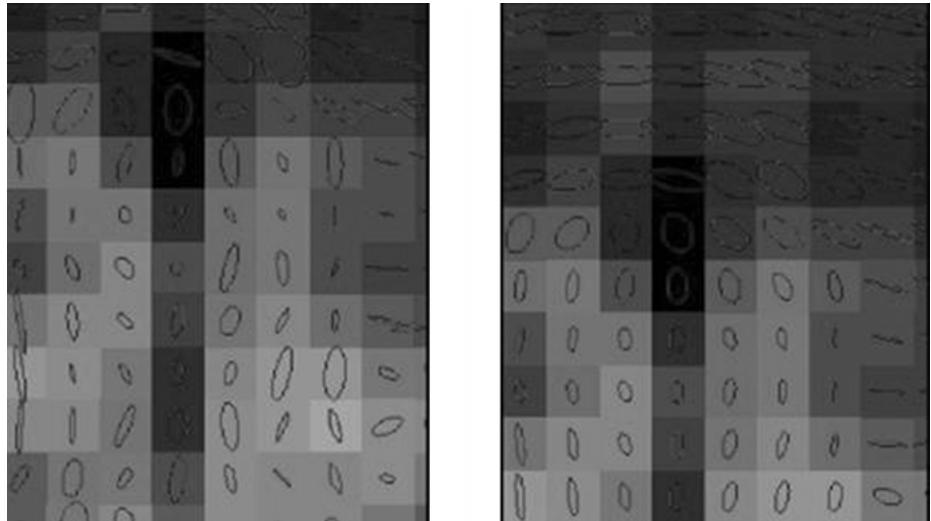


Fig. 10.5: The different representation of the tensor field on different coordinate systems.

Definition 10.5. In n -dimensional Euclidean space, a curvilinear coordinate system can be transformed into a new curvilinear coordinate system.

$$x^{i'} = f^{i'}(x^1, x^2, \dots, x^n). \quad (10.18)$$

The transformation formula of the affine frame is $e_{i'} = (\partial x^i / \partial x^{i'}) e_i$.

Thus, the representation of the component parts of the tensor field is different in different coordinate systems. In the $\langle X, \psi, \Phi, f, \Phi' \rangle$ model, the method to reduce the data is to find an appropriate affine transform f . Because of the tensor invariance, f is invertible (Figure 10.5).

10.2.3 Design of data reduction algorithm based on the tensor field

For obtaining a component form of the tensor field in low-dimensional space under the action of f , we solve the affine transformation f by using the high-order singular value decomposition (HOSVD). HOSVD is a commonly used tensor decomposition method, which can represent a tensor as the tensor product of matrices and a kernel tensor.

Definition 10.6. The tensor n -mode product $X \times_d U$ of tensor $X \in R^{N_1 \times \dots \times N_M}$ and a matrix is a tensor, and its form is $N_1 \times \dots \times N_{d-1} \times N' \times N_{d-1} \times \dots \times N_M$:

$$(X \times_d U)_{i_1 \dots i_{d-1} j_{d+1} \dots i_m} = \sum_{i_d} (X_{i_1 \dots i_m} U_{j_i d}) \quad (10.19)$$

Definition 10.7. The n pattern unfold of tensor $A \in R^{I_1 \times \dots \times I_N}$ is a matrix $A_{(n)} = R^{I_n \times (I_{n+1} \dots I_N I_1 \dots I_{n-1})}$. The element of this matrix is $a_{i_1 \dots i_N}$. The row-coordinate is i_n and the column-coordinate is

$$(i_{n+1} - 1)I_{n+2}I_{n+3} \dots I_N I_1 I_2 \dots I_{n-1} + (i_{n+2} - 1)I_{n+3}I_{n+4} \dots I_N I_1 I_2 \dots I_{n-1} + \\ (i_N - 1)I_1 I_2 \dots I_{n-1} + (i_1 - 1)I_2 I_3 \dots I_{n-1} + (i_2 - 1)I_3 I_4 \dots I_{n-1} + \dots + i_{n-1} .$$

Definition 10.8. The n -mode vector of tensor A is the column vector of A_n (Figure 10.6).

We can represent each tensor $A \in R^{I_1 \times \dots \times I_N}$ as an n -mode product by using HOSVD:

$$A = S \times_1 U^{(1)} \times_2 U^{(2)} \dots \times_N U^{(N)} \quad (10.20)$$

Here, we have the following rules:

- (1) $U^{(n)} = (u_1^n u_2^n \dots u_{I_n}^n)$ is called the n -mode $U^{(n)} \in R^{I_n \times I_n}$ matrix.
- (2) $S \in R^{I_1 \times \dots \times I_N}$ is a kernel tensor, and its sub-tensor has two characters. Complete orthogonal character: for any $\alpha, \beta, \alpha \neq \beta$, we have $\langle S_{i_n=\alpha}, S_{i_n=\beta} \rangle = 0$; Ordering, i.e., $\|S_{i_n=1}\| \geq \|S_{i_n=2}\| \geq \dots \geq \|S_{i_n=I_n}\| \geq 0$.

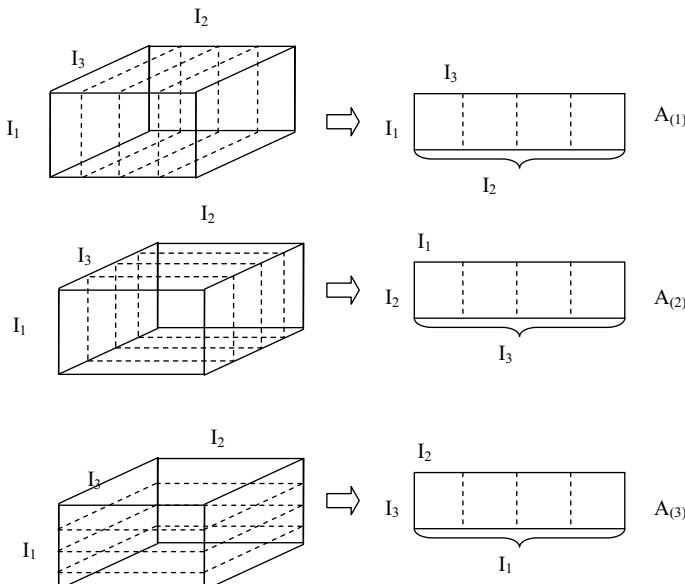


Fig. 10.6: The three order unfold of a tensor.

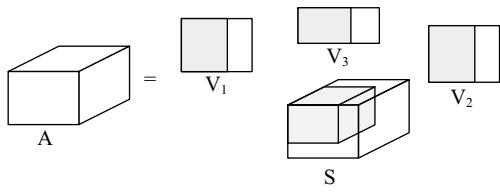


Fig. 10.7: The three order unfold of a tensor.

Intuitively, the ordering of a tensor [21, 21] means the energy of the kernel tensor. Based on this point, we can use A' to denote approximate tensor A (Figure 10.7):

$$A = S \times_1 U^{(1)} \times_2 U^{(2)} \cdots \times_N U^{(N)} \approx A' = S' \times_1 U'^{(1)} \times_2 U'^{(2)} \cdots \times_N U'^{(N)} \quad (10.21)$$

Here $A, S \in R^{I_1 \times \cdots \times I_N}$, $A', S' \in R^{K_1 \times \cdots \times K_N}$, $U^{(n)} \in R^{I_n \times K_n}$. Based on the decomposition method of HOSVD, we can find the implementation form of f .

We know that $A_i \in R^{I_1 \times \cdots \times I_N}$, $S_i \in R^{K_1 \times \cdots \times K_N}$ ($i = 1 \dots m$), which belongs to two tensor fields Φ and Φ' :

$$S_i = f(A_i) = A_i \times U^{(1)} \times U^{(2)} \cdots U^{(N)}. \quad (10.22)$$

Here, we have $U^{(n)} \in R^{I_n \times K_n}$ and $U^{(n)}(U^{(n)})^T = I_n$. $U^{(n)}$ is the transformation matrix of f . Because $K_n < I_n$, $n = 1 \dots N$, Φ' is a simple form of Φ , we can use $f(\Phi')$ to represent Φ , i.e.,

$$\Phi \approx f(\Phi'). \quad (10.23)$$

Specifically, we have to complete the optimisation problem as:

$$\min_{\substack{U^{(1)} \dots U^{(N)} \\ S_1 \dots S_N}} \sum_i \|A_i - S_i \times_1 (U^{(1)})^T \times_2 (U^{(2)})^T \cdots \times_N (U^{(N)})^T\|_F^2. \quad (10.24)$$

Then, we can deduce that the minimisation problem is equivalent to the following maximisation problem [4–9, 24].

$$\max_{U^{(1)} \dots U^{(N)}} \sum_i \|A_i \times_1 U^{(1)} \cdots \times_N U^{(N)}\|_F^2. \quad (10.25)$$

Then, we have:

$$Y_i^k = A_i \times_1 U^{(1)} \cdots \times_{k-1} U^{(k-1)} \times_{k+1} U^{(k+1)} \cdots \times_N U^{(N)}. \quad (10.26)$$

It can be transformed into:

$$\begin{aligned}
& \sum_i \|A_i \times_1 U^{(1)} \times_2 U^{(2)} \cdots \times_N U^{(N)}\|_F^2 \\
&= \sum_i \|Y_i^k \times_k U^{(k)}\|_F^2 \\
&= \sum_i \|\left(Y_i^{k(k)}\right)^T \times_k U^{(k)}\|_F^2 \\
&= \sum_i \text{tr} \left[\left(U^{(k)}\right)^T Y_i^{k(k)} \left(Y_i^{k(k)}\right)^T U^{(k)} \right] \\
&= \sum_i \text{tr} \left[\left(U^{(k)}\right)^T \left(\sum_i Y_i^{k(k)} \left(Y_i^{k(k)}\right)^T\right) U^{(k)} \right]. \tag{10.27}
\end{aligned}$$

The method of solving for $U^{(k)}$ is similar to the least-squares method of the HOOI algorithm. First, we assume that W_k are already known. To maximise (10.27), $U^{(k)} \in R^{I_k \times K_k}$ should include $K^{(k)}$ eigenvectors corresponding to the largest eigenvalues. The specific algorithm is shown in the following algorithm.

Algorithm 10.2: Data reduction algorithm based on tensor field

Input: Data $A_i \in R^{I_1 \times I_2 \times \cdots \times I_N}$, ($i = 1 \dots m$), $K = \{K_n\}$, ($n = 1 \dots N$).

Output: $U^{(n)} \in R^{I_n \times K_n}$, $S_i \in R^{K_1 \times \cdots \times K_N}$, ($n = 1, 2, \dots N$, $i = 1, 2, \dots m$).

Process:

- (1) set $U^{(n)}$ to a matrix; its column vectors are orthogonal to each other
- (2) do:
 - a) Solve Y_i^k ;
 - b) $W_k \leftarrow \sum_i Y_i^{k(k)} \left(Y_i^{k(k)}\right)^T$;
 - c) Do the singular value decomposition of W_k ;
 - d) Update $U^{(k)}$.
 End for
 Until convergence
- (3) calculate S_i : $S_i = A_i \times_1 U^{(1)} \times_2 U^{(2)} \cdots \times_N U^{(N)}$

10.2.4 Experiment

To verify the algorithm validity, we applied Algorithm 10.2 to human face reconstruction. In the data reduction model of the tensor field, X is the original image set. From the viewpoint of the tensor, a human face image is a second-order tensor, and it is the result of the tensor product of the column vector and row vector. For a grey image with size 112×92 , it represents an element in tensor space $R^{112} \otimes R^{92}$. The human face image is a low-dimensional manifold [7–10]. We should choose a different K and observe the experimental results (Figure 10.8).



Fig. 10.8: The rebuilt human face.

We calculate the kernel tensor $S_i \in R^{K_1 \times \dots \times K_N}$ and the affine transformation matrix $U^{(n)} \in R^{I_n \times K_n}$, and use storage space M to save them:

$$M_1 = I_1 \times K(1) \times \dots \times I_N \times K(N) + m \times K(1) \times \dots \times K(N). \quad (10.28)$$

The storage space of the original X is:

$$M_2 = m \times I_1 \times \dots \times I_N. \quad (10.29)$$

In the Algorithm 10.2, M_1 is smaller than M_2 , which can be proved by the experimental result. The following figure shows the rendering of the reconstruction of the human face image: Figure 10.9(a) shows the original figure, and (b)–(d) show the results under different parameters, i.e., $K = \{90, 80\}$, $K = \{80, 70\}$, $K = \{50, 40\}$.

Figure 10.9 shows the compressing quality of the algorithm, where k represents the reduction dimension and m is the number of images. From the experimental result, when the number of data objects is small, the experimental effect is not obvious. When the number of data objects is large, the required storage space is smaller than the original image. From the experimental result, the algorithm plays a key role. In the classical algorithm, the vector representation of images always occupies an important position, e.g., in *Eigen-faces*, *Fisher-faces*, *Laplacian-faces*.

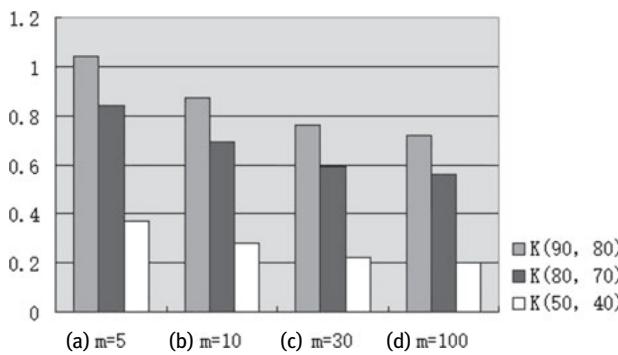


Fig. 10.9: The experiment results with different K and m values.

10.3 The learning model and algorithm based on the tensor field

10.3.1 Learning model based on the tensor field

We found that the objects of the actual processing can be grouped by the similarity between them. By adopting such a treatment, the objects have the structure of a cluster. This cluster is similar to a tensor bundle. In cluster analysis, there are some different types of high-dimensional data corresponding to different sub-spaces. The number of dimensions of each sub-space is different as well. To get a cluster structure in the full space is difficult. In the description of tensor bundle language, each sub-space produced by clustering is a tensor space. Each set of sub-spaces is a bundle space.

From the above, we can see that there is a corresponding relationship between the data and the tensor bundle. In Figure 10.10, we can see that each type of data object is a point x on the manifold M , and each point x has a tensor space $T_x(M)$. All the tensor spaces constitute a full space, which is defined as follows:

$$T(M) = \bigcup_{x \in M} T_x(M). \quad (10.30)$$

Here, we obtain the tensor bundle $(T(M), \pi, M)$, where $T(M)$ is the full space, M is the bottom space, and π is the bundle projection. Further, $T_x(M)$ is the fibre. Each element τ is a multi-linear map: $\tau: M \rightarrow T(M)$ in the tensor field on M . In addition, $\forall x \in M$, we have $\tau(x) \in T(x)$.

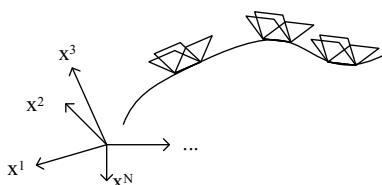


Fig. 10.10: The corresponding relationship between the data and tensor bundle.

The tensor field is a cross section of the tensor bundle. In a rectangular coordinate system, the affine frame of each point equals a constant in Euclidean space. However, in curvilinear coordinates, the affine frame of each point is a function that changes with the point, i.e., $e_i = \partial R / \partial x^i$, and e_i is the tangent space of point p in manifold M .

The tangent space is a special vector space on the manifold. There are some other vector spaces on M . In general, we call the vector space in point p the tensor product. For the data $X \in R^{I_1 \times \dots \times I_N}$, it is associated with the vector space $V_1 \dots V_N$. Then, we can obtain the tensor description of the data.

The tensor bundle learning method can generate the tensor bundle structure for the known data. For each data cluster, the tensor bundle is a point x on M . The tensor space $T_x(M)$ is a description of the data cluster. When we obtain the base tensor σ_i^x , each data object T has the tensor description of $T_x(M)$.

$$T = \sigma_i^x \otimes W . \quad (10.31)$$

Each data object can be represented by the corresponding base tensor. For a test sample D , we determine its properties for matching a suitable position in the tensor bundle. We describe it as $(T(M), \pi, M)$ by the tensor bundle. Here, we find an appropriate bundle projection π in $T(M)$.

$$\pi: T(M) \rightarrow M . \quad (10.32)$$

Here, we give the tensor bundle learning model $\langle X, M, T(M), \pi, \sigma \rangle$:

X : input data set: $X = \{X_1, \dots, X_N\}$;

$T(M)$: full space of tensor bundle;

M : bottom space;

σ : basement of tensor space;

π : bundle projection: $\pi: T(M) \rightarrow M$.

Here, $(T(M), \pi, M)$ is the tensor bundle. The tensor bundle learning model is processed to construct the tensor space of each point on M and select an appropriate bundle projection.

M is a matrix. It can be rewritten in the following form:

$$M = U \Sigma V^T = \sum_{i=1}^n \sigma_i u_i v_i^T . \quad (10.33)$$

With high-order singular value decomposition, tensor A can be decomposed into kernel S and projection matrices $U^{(n)}$, and we have $A = S \times_1 U^{(1)} \times_2 U^{(2)} \dots \times_N U^{(N)}$. Further, $S_{i_N=v}$ is a sub-tensor in the tensor, $A_v = S_{i_N=v} \times_1 U^{(1)} \times_2 U^{(2)} \dots \times_{N-1} U^{(N-1)}$.

Compared with matrix M , the tensor has a similar representation [9, 11, 12] (Figure 10.11):

$$A = \sum_{v=1}^{i_N} A_v \times_N w_v . \quad (10.34)$$

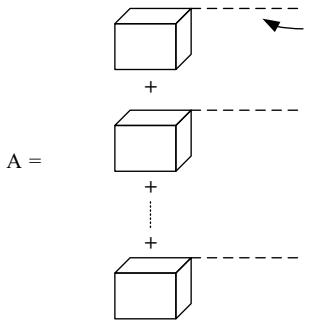


Fig. 10.11: The linear combination of every elements of one oint in the tensor space.

Here, A_v has orthogonality for each $v \neq \mu$.

$$\begin{aligned}
 & \langle A_v, A_\mu \rangle \\
 &= \text{tr}(A_v^T A_\mu) \\
 &= \text{tr}((S_{i_N=v} \times_1 U^{(1)} \times_2 U^{(2)} \cdots \times_{N-1} U^{(N-1)})^T (S_{i_N=\mu} \times_1 U^{(1)} \times_2 U^{(2)} \cdots \times_{N-1} U^{(N-1)})) \\
 &= \langle S_{i_N=v}, S_{i_N=\mu} \rangle \\
 &= 0.
 \end{aligned} \tag{10.35}$$

A series of tensors A_v can be explained as the basement of the tensor space. Thus, every element in the tensor space is a linear combination of the tensor basement. Then, we have

$$A_{i_N=v} = \sum_{v=1}^{i_N} \sigma_v A_v. \tag{10.36}$$

Here $w_v = (\sigma_{i_1}, \dots, \sigma_{i_N})$ is the linear coefficient.

For a data object D , we determine its categories to determine the tensor space that the full space is projected to (Figure 10.12).

In the process of training, each training sample can be represented by a linear description in the tensor space with coefficients $w_v = (\sigma_{i_1}, \dots, \sigma_{i_N})$. In the testing phase, each testing sample D has an approximate description in the tensor bundle with coefficients $w_v = (\sigma_{i_1}, \dots, \sigma_{i_N})$.

$$T \approx A_v w_v. \tag{10.37}$$

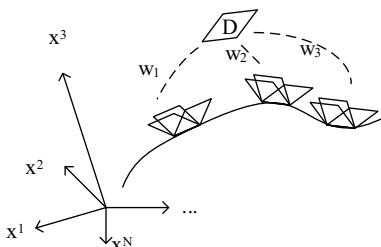


Fig. 10.12: The projection bundle of a tensor.

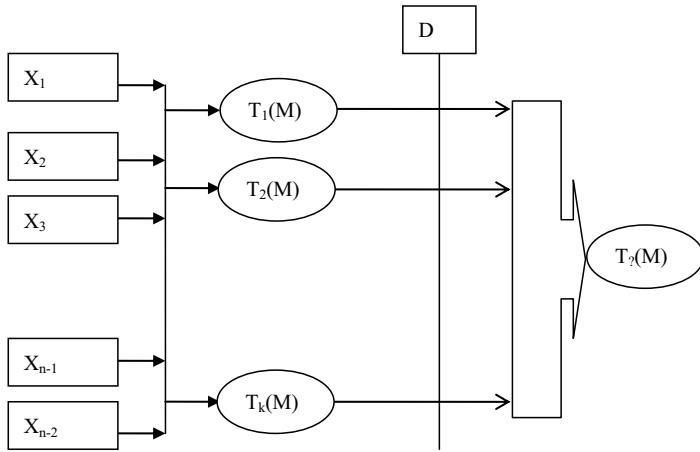


Fig. 10.13: The tensor bundle learning model.

Here, we measure the describing ability of a tensor space by using ε , and we have

$$\varepsilon = \|T - A_v w_v\| \quad (10.38)$$

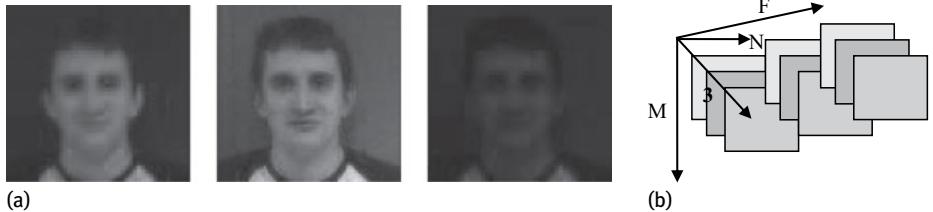
Finally, we summarise the tensor bundle learning model as follows: (Figure 10.13).

- Construct the tensor based on the training sample;
- Reconstruct the sub-tensor by tensor decomposition and build the tensor space;
- Project the testing sample into the tensor space of each object cluster;
- Determine the attributes of the testing sample with the projection results.

10.3.2 Tensor bundle learning algorithm

In recent years, research has indicated that colour provides important information for human face recognition [19–22, 36, 37]. If we use the colour information in the classical PCA algorithm, the recognition effect can be better than that with only the grey information. A common method to represent a colour image is to divide the RGB space into three parts. However, this method, in theory, lacks support and experimental evidence. Another way to represent a colour image is the linear combination of every colour channel [23], e.g., $D = x_1 R + x_2 G + x_3 B$. Here x_1 and x_3 are the combination factors. All these methods can manipulate the colour image vectorisation. In the tensor bundle learning algorithm, we can use the colour information as another view of the tensor, which would make it easier to understand the image.

First, we consider the construction method of the tensor model of a human face image. In RGB colour space, each image can be divided into three parts (Figure 10.14). Each part represents a different colour ingredient. For each person, we can construct a fourth-order tensor of their face as $T_p \in R^{I_1 \times I_2 \times I_3 \times I_4}$ (Figure 10.14), where $I_1 \times I_2 \times I_3 \times I_4$



(a)

(b)

Fig. 10.14: The tensor model of a human face.

I_4 are the numbers of dimensions of different orders. Then, we build the tensor data model of the human face image.

When the tensor model of the image data is obtained, we will analyse how to build a tensor space of a data cluster. Here, we followed high-order singular value decomposition to obtain the tensor space.

First, for each human tensor data, we perform singular value decomposition, and we have

$$T_p = S_p \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)} \times_4 U^{(4)}. \quad (10.39)$$

Then, we convert it to:

$$T_p = \sum_{f=1}^{i_4} B_f \times_N w_f. \quad (10.40)$$

Here, $B_f = S_f \times U^{(1)} \times U^{(2)} \times U^{(3)}$, where S_f is the f sub-tensor of S_p [21, 22] and $T_{i_4=f}$ is the f sub-tensor of T . Thus, we have

$$T_{i_4=f} = \sum_{v=1}^{i_4} \sigma_v B_v. \quad (10.41)$$

The relationship between σ_v and w_f is $w_f = (\sigma_1, \sigma_2, \dots, \sigma_{I_4})$.

From the above, obviously, each colour human face $T_{i_4=f}$ is a linear combination of B_f with the coefficient w_f . These B_f values can represent the tensor basements of the p -th human face. Furthermore it can confirm a tensor space of a colour human face image.

Finally, in the test phase, we need to confirm the tensor space, which includes the sample. To be specific, we need to confirm the coefficients $w_f = (\sigma_1, \sigma_2, \dots, \sigma_{F'})$ and make $\sigma_1 B_1 + \sigma_2 B_2 + \dots + \sigma_{F'} B_{F'}$ equal to it. We have the following objective function.

$$\min_{\sigma_i} \|E - \sum_{i=1}^{F'} \sigma_i B_i\|. \quad (10.42)$$

To solve this function is a least-squares problem [12]. We have:

$$\epsilon = \|E - \sum_{i=1}^{F'} \sigma_i B_i\|^2 = \left\langle E - \sum_{i=1}^{F'} \sigma_i B_i, E - \sum_{i=1}^{F'} \sigma_i B_i \right\rangle = \langle E, E \rangle - \sum_{i=1}^{F'} \langle E, B_i \rangle^2 = 1 - \sum_{i=1}^{F'} \langle E, B_i \rangle^2. \quad (10.43)$$

Here, ε is the smallest error. For each tensor base, we need to obtain a new. According to the above, we have the following tensor bundle learning algorithm, which includes two steps, namely training and testing.

Algorithm 10.3: Training step of tensor bundle learning algorithm**Input:** sample set X , reduction dimension vector $R = \{R_1, R_2, \dots, R_K\}$.**Output:** tensor base set $\{A_1, A_2, \dots, A_c\}$, number of clusters c .**Process:**

1. Transform X into tensor set T , and the number of orders is K .
 2. Manipulate every element of T by high-order singular value decomposition.
 3. Construct tensor basements against the last order K of tensor $T(c)$
 - a) Initialise S to be the sub tensor;
 - b) Get the first column vectors to construct U ;
 - c) Obtain the new A_j by the mode- j product of old A_j and U ;
 - d) Normalise A_j ;
 - e) Return to step a;
-

Algorithm 10.4: Testing step of tensor bundle learning algorithm**Input:** Test sample D .**Output:** Cluster that D belongs to.**Process:**

- (1) Transform D into tensor representation Z .
 - (2) Normalise Z .
 - (3) Obtain the error $\varepsilon(c)$ of the approximate representation of Z .
 - For each tensor space A_c of the tensor bundle
 - For each base tensor A_j of A_c
$$\varepsilon(c) \leftarrow \langle A_j, z \rangle^2$$
$$\varepsilon(c) \leftarrow 1 - \varepsilon(c)$$
 - End for
 - End for
 - (4) D is assigned to object cluster p .
-

10.3.3 Classification model based on the tensor field

The first step of the data classification problem is to preprocess the data. The quality of the original data influences the classifying quality. Therefore, it is important to understand the intrinsic properties of the data [38, 39]. The intrinsic properties include the following:

Diversity. In general, data sets come from multiple sources. On other hand, the same property of different data sets has different names. Therefore, it is important to understand the diversity of the data. Redundancy. Data sets have a mass of redundant data, e.g., a data set may have repeatability. High dimension. In general, the classification of data sets exists in high-dimensional space.

Tab. 10.1: The information of the data.

No.	User	Item	Tag	Weight
1	U1	I1	T1	1
2	U2	I2	T2	1
3	U3	I3	T3	1
4	U4	I4	T4	1

Tab. 10.2: The information of the constructed data.

No.	User	Item	Tag	Weight
1	U1	I1	T1	0.72
2	U2	I1	T1	1.17
3	U3	I2	T2	0.72
4	U4	I3	T3	1
5	U5	I2	T2	0.44

The classification process has two main steps. First, we build a classification model based on the same features of a group of objects in the data set. We then map the data term into a class by using the model. For building a classification model, we divide the data set X into two parts; one part is called the training sample set X_e , and the other part is called the testing sample set X_t . All the samples have the same cluster identification. The classification algorithm constructs the classification model by analysing X_e . We test the classification precision of the model by using X_t . If the precision is sufficient, we use the corresponding algorithm to classify the data set (Tables 10.1 and 10.2).

On the basis of the above process, we have the classification model of a tensor field. Its basic principles include the following:

The invariance principle of the tensor. A tensor can provide a uniform representation in different coordinate systems based on its tensor characteristics [23, 24].

Rich expression ability. A high-order tensor can represent richer information. Most importantly, we can find the hidden information of the data by tensor reconstruction.

The data discussed in this chapter belongs to the same type. The data type usually includes some characters. There are some close relationships between these characters, which are simply operated. We use the tensor method to analyse these relationships and reflect the internal connection between the data objects.

X is a data sample and represented by its neighbourhoods ξ_X in the high-dimensional space. Based on the above analysis, we can obtain the category of X by using ξ_X . Here, ξ_X describes the environmental information of X . Every point has a corresponding tensor in the high-dimensional linear space, which constructs a tensor field model (Figure 10.15).



Fig. 10.15: The tensor model.

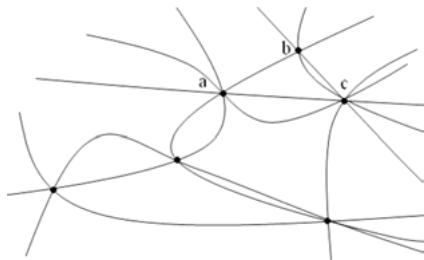


Fig. 10.16: The tensor model which is used to describe the interaction between different points.

Tensor reconstruction is useful for showing the inner rules of the data [14, 21]. If we decompose and reconstruct all the tensors in the tensor space, we can obtain the distribution rule of the high-dimensional space. There is an interaction between point X and ξ_X (Figure 10.16). However, this type of interaction is a local element, which is determined by the data distribution.

The block processing of data includes two important factors: dimensionality and locality.

Dimensionality. In the tensor field methods, the dimension of the data corresponds to the order of the tensor. Therefore, the dimension of the order of each tensor is determined by each dimension of the original data. In general, the dimension is larger. Then, the tensor has the high-dimensional characteristic. The block processing of data would reduce the dimension of the tensor and increase the calculation efficiency.

Locality. The relationship between close points is intimate. We can find this characteristic from the geometric structure characteristics. If we only processed all the

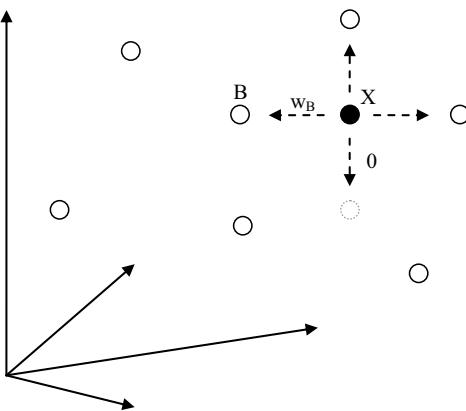


Fig. 10.17: The tensor model of point X .

points as a whole, it would break the local structure and not be useful to find the local characteristic.

Here, we explain this viewpoint by using a simple example. Figure 10.18(a) shows the distribution characteristic of known data points. The region A is null. We process these data points in the overall structure and local structure, respectively. Figure 10.18(b) and 10.18(c) show the results. Figure 10.18(b) and 10.18(c) can be made more explicit to show the internal regularity of the original data.

Each data point corresponds to a weight value w during tensor processing. The unknown data correspond to zero. Figure 10.17 shows tensor T_x in point X . The weight value w_B represents the extent of the effect of B on X . Combined with the previous analysis, we can see that all the data points would be given a new weight value. We can then estimate the cluster data point.

Considering the three points A , B , and C as well as ξ_A yields the tensor T_A . ξ_C determines the tensor T_C . All these tensors include the information about point B . The weight value w of B would be affected by tensor A and tensor B , respectively, and we obtain the new weight values w'_1 and w'_2 . The final weight value w' of B would be equal to the mean value of w'_1 and w'_2 , i.e., $w' = w'_1 + w'_2 / 2$.

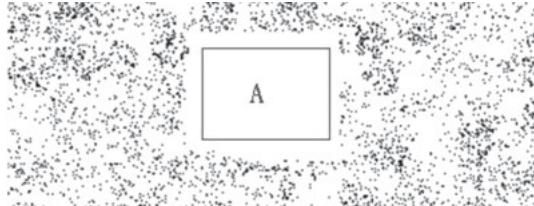
If one point is affected by its M neighbour points, the environmental information of this point could be represented by an N -order tensor $T^{I_1 \times I_2 \times \dots \times I_N}$. The final weight of point X will be as follows (Figure 10.19):

$$w_X = \frac{1}{M} \sum_{i=1}^M T_i . \quad (10.44)$$

If the orders of tensor A and tensor B are the same, the result of the tensor addition is C . It can be represented by $C_{ijk\dots} = A_{ijk\dots} + B_{ijk\dots}$.

The result of the reconstruction of the tensor field gives us the data distribution characteristics. We can estimate the related information of a point based on the data distribution characteristics. In order to achieve this goal, we have to construct a new tensor $\Gamma^{J_1 \times J_2 \times \dots \times J_N}$ by utilising the entire tensor field, which is also called the discrimi-

(a)



(b)



(c)

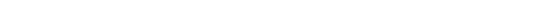


Fig. 10.18: The global and local processing step.

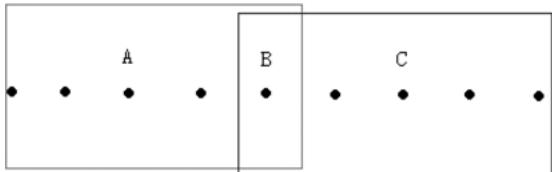


Fig. 10.19: The point B is infected by point A and point C at the same time.

nant tensor. From the perspective of the classification task, the discriminant tensor is a classifier. The information of the weight values of $\Gamma^{J_1 \times J_2 \times \dots \times J_N}$ could yield the classification attributes of the nodes to be considered.

If the cluster of the data object has the k order of Γ , the k -mode vector can describe the cluster characteristics of the data object. We can obtain $\Gamma_{(k)}$ by k -mode unfolding. Finally, we assess the cluster characteristics of the data object based on the k -mode vector.

10.3.4 Classification algorithm based on the tensor field

According to the above analysis, we introduce the tensor field in the classification algorithm. There is a data set $X = \{X_1, X_2, \dots, X_M\}$ and X_i is the sample. Each sample has N attributes, i.e., $X_i = \langle P_{i1}, P_{i2}, \dots, P_{iN} \rangle$, where P_j represents the j -th attribute of each sample. The range of values allowed for P_j is $[U_{Pj}, D_{Pj}]$ and includes C_j possible values.

First, we discuss how to obtain a tensor field model for a known data set. In simple terms, any data object can be described by each attribute in common. Each attribute is a view of the sample, and all the attributes construct a full expression of the sample. Figure 10.20 shows that sample A is described by four attributes.

Therefore, we can construct a tensor, each order of which corresponds to an attribute of samples. The dimension number of the j -th order is determined by the neighbour points ξ_{X_i} , and we have $I_j \leq \|U_{Pj} - D_{Pj}\|$. Therefore, we can construct a tensor $T_i^{I_1 \times I_2 \times \dots \times I_N}$ of each sample X_i based on its high-dimensional linear space.

$T_i^{I_1 \times I_2 \times \dots \times I_N}$ is the description of the local environmental information, which is also the description of the neighbouring region ξ_{X_i} . Therefore, the value of $T_i^{I_1 \times I_2 \times \dots \times I_N}$ is determined by ξ_{X_i} . If we have a sample $X_j \in \xi_{X_i}$, X_j would have a weight value w_{ij} . The component of $T_i^{I_1 \times I_2 \times \dots \times I_N}$ is:

$$T_i^{i_1 \times i_2 \times \dots \times i_N} = w_{ij}, X_j = (P_{j1}, P_{j2}, \dots, P_{jN}) \in \xi_{X_i}. \quad (10.45)$$

Here, each point of the linear space determines a tensor. Then, the data set X can determine a tensor field $\Phi(X)$.

We reconstruct the tensor field $\forall T_i \in \Phi(X)$ and decompose T_i by high-order singular value decomposition to

$$T_i = S_i \times_1 U^{(1)} \times_2 U^{(2)} \cdots \times_N U^{(N)}. \quad (10.46)$$

Here, $T_i, S_i \in R^{I_1 \times I_2 \times \dots \times I_N}$ and $U^{(i)}$ is a $I_n \times I_n$ square matrix. Setting $K = \{K_1, K_2, \dots, K_N\}$, it is the reduction dimension of $S(\xi_{X_i}) = \{I_1, I_2, \dots, I_N\}$.

$$\tilde{T}_i = \tilde{S}_i \times_1 \tilde{U}^{(1)} \times_2 \tilde{U}^{(2)} \cdots \times_N \tilde{U}^{(N)}, \quad (10.47)$$

which is the tensor after the reconstruction, and $\tilde{T}_i \in R^{I_1 \times I_2 \times \dots \times I_N}$, $\tilde{S}_i \in R^{K_1 \times K_2 \times \dots \times K_N}$, $\tilde{U}^{(n)} \in R^{I_n \times K_n}$.

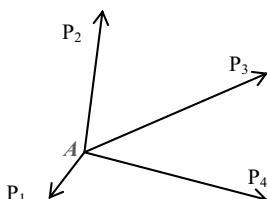


Fig. 10.20: The tensor model of the data set of A .

Tab. 10.3: The continuous attribute values of data set wine.

In the above algorithm, the attribute values of data are required to be discrete. For continuous attribute values, we need to discretise them. The continuous attribute values are discretised to set some divide point within the range. The discretisation reduces the system's demand for storage space and increases the release of storage space. Table 10.3 shows the Wine data set of UCI, and its condition attributes are all continuous values. From the previous section, we require I_1, I_2, \dots, I_N to be natural numbers for building a series of tensors $T_i^{I_1 \times I_2 \times \dots \times I_N}$. Therefore, we need to project every real attribute value of the data set Wine into the integer attribute values (Table 10.4).

The discretisation should boil down to partitioning of the space, which is achieved by the condition attributes. We divide the space with N dimensions into several regions. Each decision value of the objects in each region is the same. The process of choosing a breakpoint is also a process of merging attribute values. It can reduce the number of attribute values and reduce the problem complexity by merging the attribute values.

At present, the discretisation methods include two categories; one is the unsupervised discretisation method, e.g., the fixed-pitch interval method and equal frequency interval method, and the other is the supervised discretisation method, e.g., the discretisation method based on entropy and the discretisation method based on statistics [13]. The latter approach considers the dependency relationship between the condition attribute and the decision attribute during the processing step. Therefore, this article employs two algorithms to preprocess the data; one is based on the entropy,

Tab. 10.4: The discretization of the continuous attribute values.

called MDL, and the other is based on statistics, called CHI2. According to the above analysis, we present Classification Algorithm 10.5 and 10.6 based on tensor fields.

Algorithm 10.5: Training step of tensor field classification algorithm

Input: data set $X = \{X_1, X_2, \dots, X_M\}$, $X_i = \langle P_{i1}, P_{i2}, \dots, P_{iN} \rangle$, $i=1\dots M$

Output: Discriminant tensor Γ

Process:

- (1) Discretise the continuous attribute values of X by using MDL/CHI2 algorithm, and calculate the range of the attribute values of X , $U_i = (U_P, D_P)$, $i = 1 \dots N$.
 - (2) Determine the dimension $D = \{l_1, l_2, \dots, l_N\}$ of each order of the tensor based on U_i , and N is the dimension number of X .
 - (3) Construct tensor field $\Phi(X)$ of X . Initialise $\forall T_i \in \Phi(X)$, $i = 1 \dots M$.
 - (4) Process $\Phi(X)$ by using higher-order singular value decomposition (HOSVD), and reconstruct $\Phi(X)$.
 - (5) Perform $\Phi(X)$ equalisation and calculate discriminant tensor $\Gamma \in R^{U_1 \times U_2 \times \dots \times U_N}$.
-

Algorithm 10.6: Testing step of tensor field classification algorithm

Input: discriminant tensor Γ , testing sample set A , and $\|A\| = M$

Output: correct recognition rate

Process:

For $m = 1 \rightarrow M$

- (1) put the attribute value of sample A_m into the vector I (without include the cluster value)
 - (2) analyse the k-mode vector V of Γ (set the k -th attribute value to be the cluster attribute value)
For $i=1 \rightarrow U_k$
 - a) $V \leftarrow \Gamma_{I(1)\dots I(k-1)i(I(k+1)\dots I(N))}$;
 - b) $[P, I] = \max(V)$;
 - c) $J \leftarrow$ cluster attribute value of sample $A(i)$
 - d) If $I = J$
 - e) $C \leftarrow C + 1$, C is the number of correct recognitions;
- End if
- End for

End for

10.4 Summary

When using a quantity to describe the rapid growth of data regularity, we usually need to introduce a coordinate system to the quantitative data analysis. Then, the data essential rule, which is not related to the coordinate system, can make a connection with the original mathematical theory. The learning theory and method based on tensor field were generated in this context. Using the tensor method can simplify the derivation, express uniformity and uniformity, and describe some laws in the natural discipline with tensors, which have an invariable form under any coordinate system. Based

on the decomposition technique of tensors, the data can better describe the inner rules of data sets after reconstruction.

In this chapter, we introduced tensor field and tensor bundle theory into machine learning, and gave the basic concept of tensor cluster and tensor field on manifolds. In addition, we found the corresponding data reduction model of the corresponding tensor field for data reduction. A tensor cluster and a tensor field representation of the data in machine learning were presented.

Bibliography

- [1] Michell TM. Machine Learning [M]. Peking: Chinese machine press, 2004.
- [2] Jiawei H, Kamber M. Data Mining Concepts and Techniques Third Edition [M]. 2011.
- [3] Turk M, Pentland A. Eigenfaces for recognition [J]. Cognitive Neuroscience, 1991, 3(1), 71–86.
- [4] Belhumeur PN, Hespanha JP, Kriegman DJ. Eigenfaces vs Fisherfaces: recognition using class specific linear projection [J]. IEEE Transactions Pattern Analysis Machine Intelligence, 1997, 19(7), 711–720.
- [5] Lee DD, Seung HS. Learning the parts of objects with nonnegative matrix factorization. Nature, 1999, 401, 788–791.
- [6] Liu N, Zhang B, Yan J, et al. Text Representation: From Vector to Tensor [C]. International Conference on Data Mining, 2005, 725–728.
- [7] He X, Cai D, Niyogi P. Tensor subspace analysis [C]. NIPS, 2005.
- [8] Tao D, Li X, Wu X, et al. General tensor discriminant analysis and gabor features for gait recognition [C]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007, 29 (10), 1700–1715.
- [9] Vasilescu MAO, Terzopoulos D. Multilinear subspace analysis of image ensembles [C]. Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, 2003, 1–7.
- [10] Nie F, Xiang S, Song Y, et al. Extracting the Optimal Dimensionality for Local Tensor Discriminant Analysis [J]. Pattern Recognition, 2009, 42 (1), 105–114.
- [11] Savas B, Eldén L. Handwritten digit classification using higher order singular value decomposition [J]. Pattern Recognition 2007(40), 993–1003.
- [12] Sun JT, Zeng HJ, Liu H. CubeSVD:A Novel Approach to Personalized Web Search [C], In The 14th International World Wide Web Conference, 2005, 382–390.
- [13] Cai D, He X, Han J, Tensor Space Model for Document Analysis [C]. International Conference on Research and Development in Information Retrieval, 2006, 625–626.
- [14] Ye J. Generalized low rank approximations of matrices [C]. Proceedings of International Conference on Machine Learning [C], 2004, 887–894.
- [15] De Lathauwer L, De B, Vandewalle J. One the best rank-1 and rank- (R1, R2,...,RN) approximation of higher-order tensors [C]. SIAM J. Matrix Anal. Appl. 21, 2000, 1324–1342.
- [16] Sheehan B, Saad Y. Higher Order Orthogonal Iteration of Tensors (HOOI) and its Relation to PCA and GLRAM [C]. SIAM International Conference on Data Mining, 2007, 355–365.
- [17] Zhang JY, Frangi D, Jing-Yu Yang AF. Two-dimensional PCA: A new approach to appearance-based face representation and recognition [J]. IEEE Trans. Pattern Analysis and Machine Intelligence, 2004, PAMI-26(1), 131–137.
- [18] De Lathauwer L., De Moor B., Vandewalle J. A multilinear singular value decomposition [C]. SIAM J. Matrix Anal. Appl. 21(4), 2000, 1253–1278.

- [19] Basser P, Mattiello J, LeBihan D. MR diffusion tensor spectroscopy and imaging [J]. *Biophysica*, 1994, 66, 259–267.
- [20] Westin F, Maier S, Mamata H, et al. Processing and visualization for diffusion tensor MRI [J]. *Med. Image Anal.*, 2002, 6(2), 93–108.
- [21] Bigun J, Granlund GH. Optimal orientation detection of linear symmetry [C]. Proceedings of IEEE International Conference on Computer Vision, 1987, 433–438.
- [22] Boring E, Pang A. Interactive deformations from tensor fields [C]. Proc. IEEE Visual, 1998, 297–304.
- [23] Witalis EA. Incompressible steady flow with tensor conductivity leaving a transverse magnetic field [J]. *J. Nucl. Energy*, 1966, 137–143.
- [24] Thierry D, Lambertus H. Visualizing second order tensor fields with hyperstream lines. *IEEE Computer Graphics and Applications*, 1993, 13(4), 25–33.
- [25] Jeremic B, Scheuermann G, Frey J, et al. Tensor visualization in computational geomechanics [J]. *International Journal for Numerical and Analytical Methods in Geomechanics*, 2002, 26, 925–944.
- [26] Zheng X, Parlett B, Pang A. Topological lines in 3D tensor fields and discriminant hessian factorization [C]. *IEEE Transactions on Visualization and Computer Graphics*, 2005, 11(4), 395–407.
- [27] Hotz I, Feng L, Hagen H, et al. Physically Based Methods for Tensor Field Visualization [C]. Proceedings of the IEEE Visualization'04 Conference, 2004, 123–130.
- [28] Torres L, Reutter JY, Lorente L. The importance of the color information in face recognition [C]. *International Conference on Image Processing*, 1999, 3, 627–631.
- [29] Rajapakse M, Tan J, Rajapakse J. Color channel encoding with NMF for face recognition [C]. *International Conference on Image Processing*, 2004, 3, 2007–2010.
- [30] Kittler J, Sadeghi M. Physics-based Decorrelation of Image Data for Decision Level Fusion in Face Verification [C]. *MCS04*, 2004, 354–363.
- [31] Shih P, Liu C. Improving the Face Recognition Grand Challenge Baseline Performance Using Color Configurations Across Color Spaces [C]. *IEEE International Conference on Image Processing (ICIP 2006)*:1001–1004.
- [32] Xie C, Kumar BVK. Quaternion correlation filters for color face recognition [C]. Proceedings of the SPIE, 2005, 486–494.
- [33] Jones C. III, Abbott AL. Color face recognition by hyper complex gabor analysis [C]. *FGR 2006*, 2006, 126–131.
- [34] Yang J, Liu C. A discriminant color space method for face representation and verification on a large-scale database [C]. *ICPR*, 2008, 1–4.
- [35] Panagiotis S, Alexandros N, Yannis M. Tag Recommendations based on Tensor Dimensionality Reduction [C]. Proceedings of the 2008 ACM conference on Recommender systems, 2008, 43–50.
- [36] Wang C, Yin B, Bai X, et al. Color Face Recognition Based on 2DPCA [C]. *ICPR08*, 2008, 1–4.
- [37] Li X, Li FZ. A Model of Data Reduction Based on Tensor Field [C]. Accepted by Global Congress on Intelligent Systems 2009 (GCIS 2009).
- [38] Tao Sun J, Zeng HJ, Liu H. CubeSVD: A Novel Approach to Personalized Web Search [C], In: The 14th International World Wide Web Conference, 2005, 382–390.
- [39] De LL, De B, Vandewalle J. One the best rank-1 and rank- (R1,R2,...,RN) approximation of higher-order tensors [C]. *SIAM J. Matrix Anal. Appl.* 21, 2000, 1324–1342.

11 Frame bundle connection learning

This chapter introduces the connection learning algorithm based on frame bundle, which is the multi-manifold data processing model. By using this algorithm, we can unearth the inner geometry connection between different manifolds and provide an effective method for big data learning.

11.1 Longitudinal space learning model based on frame bundle

The frame bundle has a close relationship with the tangent bundle on a manifold. We give the following definition [1].

Definition 11.1. M is a differential manifold and its dimension equals m . A frame is a combinatorial structure $(l; v_1, \dots, v_m)$, and l is a point on manifold M , v_1, \dots, v_m denote m tangent bundles of point l on manifold M . A natural projection $\pi(l; v_1, \dots, v_m) = p$ is a smooth map from P to M , and (P, M, π) is a frame on M .

The manifold learning method based on the tangent bundle usually constructs a single frame for data processing. If a small amount of noise points exist, the method should be useful. If the noise changes into another type of manifold structure and abounds, the original algorithm might lose its efficacy. $\{e_i, i = 1, 2, \dots, k\}$ is a set that includes k clusters. e_i represents the i -th cluster, which has a single manifold data structure. Therefore, the set includes a multi-manifold structure. Some studies have investigated this type of manifold structure [2]. Although these methods improve the learning process to some extent, they have some shortcomings. To overcome these shortcomings, we have to identify the relationships between different manifolds, which requires the construction of a coordinate frame. By studying the structure of a manifold, we can solve the problem easily by using a connection operator. We construct a coordinate frame for each manifold and combine them to form the coordinate coverage. We then obtain the frame bundle. Through the above analysis, we can obtain the coordinate coverage of a multi-manifold structure (Figure 11.1).

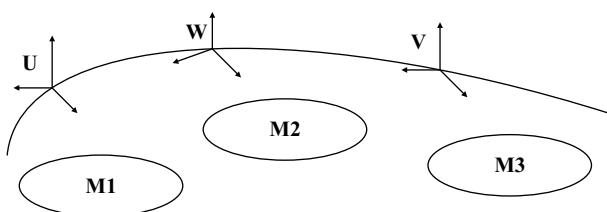


Fig. 11.1: Multi-manifold coordinate system.

Figure 11.1 shows a multi-manifold data set $\{M_1, M_2, M_3, \dots\}$. By using the frame projection, we can obtain a coordinate coverage $\{U, W, V, \dots\}$, and its elements consist of m linearly independent vectors. We then obtain a tangent space, which is the longitudinal space in the frame bundle.

We represent each coordinate domain on a manifold by using natural coordinates $(\partial/\partial u^1, \dots, \partial/\partial u^m)$. Therefore, any frame on the manifold can be represented as

$$X_i^k \left(\frac{\partial}{\partial u^k} \right)_p, \quad 1 \leq i \leq m. \quad (11.1)$$

Where (X_i^k) is a $m * m$ non-degenerate matrix.

If noise exists in a manifold data set, the data might lose its local geometric characteristic. Therefore, the algorithm will not converge, and its accuracy rate will decrease. In the real world, noise always exists. The algorithm's sensitivity to noise hinders its widespread use.

The theoretical basis of manifold learning includes the following:

- (1) When two sample points are sufficiently close, their neighbouring regions do not change after dimension reduction. This hypothesis is called local homeomorphism.
- (2) The sample points are required to cover the manifold. This hypothesis is called the density assumption.
- (3) In general, we assume that the data set is smooth and continuous. This hypothesis is called the continuity hypothesis. The noise might violate the three assumptions and yield inaccurate results.

We build a data set and show the noise problem experimentally. There are two data sets $N_1(\varphi_1, E)$, $N_2(\varphi_2, E)$, and we have $\varphi_1 = 1, 1/\sqrt{3}, \dots, 1/\sqrt{2 * n + 1}, \dots, 1/\sqrt{19}$, $\varphi_2 = -\varphi_1$. We employ the standard normal distribution to verify the effect of noise. The φ_1 of noise is equal to 0.1, 0.2, 0.5, and 1 respectively. The four cases were added to two data sets N_1 , N_2 . For each data set, we sampled 1000 points and employed the GTM algorithm [2]. The experimental results are presented below.

In Figure 11.2, four different types of noisy data are added to the 1000 sample points of N_1 . We can see that the GTM algorithm is highly robust with a relatively small error. However, as the deviation increases, the results fluctuate severely. Figure 11.3 shows the same phenomenon, which illustrates that the noise would affect manifold learning to some extent.

Next, we will analyse the influence of noise on the manifold learning algorithm in theory. In general, we mainly analyse the local linear mapping algorithm. We assume that there is a data set $X = \{x_1, x_2, \dots, x_N\}$ in the high-dimensional space R^d . For any point x_0 , $U(x_0)$ represents the neighbour domain of x_0 . If we have $x_1, x_2, \dots, x_k \in U(x_0)$, the weight values between point x_0 and its neighbour points can be represented

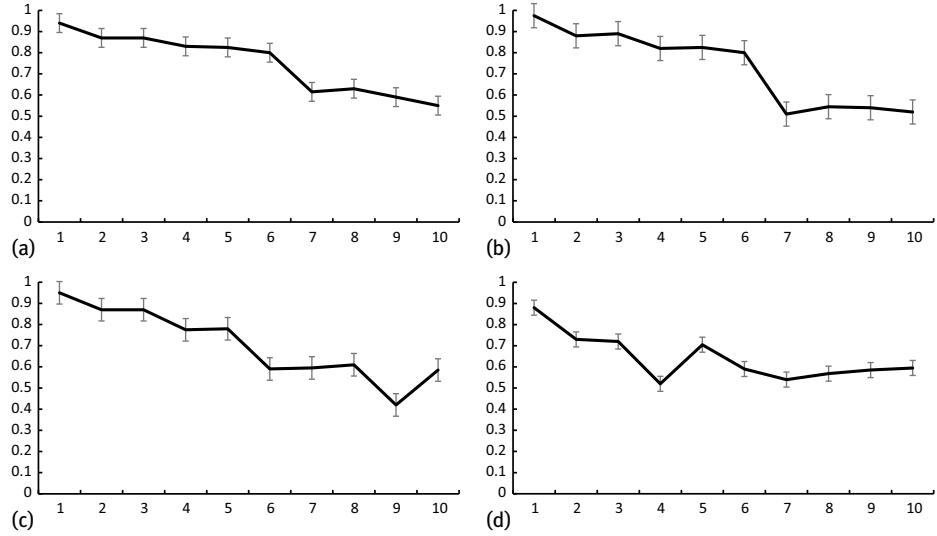


Fig. 11.2: The noise dataset where the error is 0.1 of the $N(1)$ dataset.

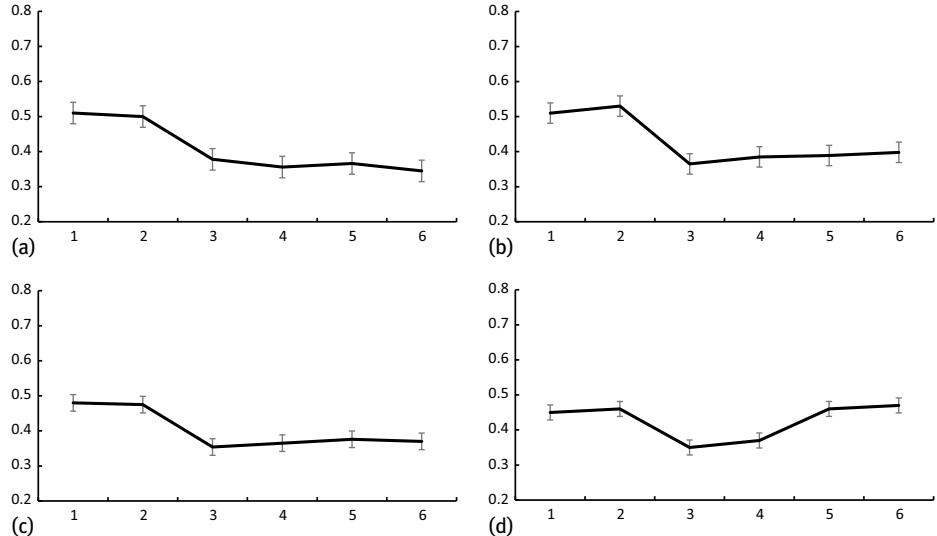


Fig. 11.3: The noise dataset where the error is 0.1 of the $N(2)$ dataset.

by w_i , and we have

$$x_0 = \sum_{i=1}^k w^i x_i, \quad \sum_{i=1}^k w^i = 1. \quad (11.2)$$

Where $x'_i = x_i + \varepsilon_i$, ($i = 0, 1, 2 \dots, k$) are the neighbour points disturbed by the noise, and the corresponding transformation is as follows:

$$x'_0 = \sum_{i=1}^k w^{i'} x'_i, \quad \sum_{i=1}^k w^{i'} = 1. \quad (11.3)$$

Here, we have $X^0 = (x_1, x_2, \dots, x_k)$, $X^{0'} = (x'_1, x'_2, \dots, x'_k)$, $w = (w^1, w^2, \dots, w^k)^T$, $w' = (w^{1'}, w^{2'}, \dots, w^{k'})^T$, and

$$x_0 = X^0 w, \quad x'_0 = X^{0'} w'. \quad (11.4)$$

In the above case, if each noise point is independent of the other with the same mean value, we have the following definition.

Definition 11.2. M is a smooth manifold, X is the sampling point set on the manifold, and ε represents the neighbour domain of x_0 . Furthermore, σ is the additional value of noise, and ε^i represents the i -th component of ε . $l = \text{rank}(X^0)$, λ_{\min} is the minimum nonzero eigenvalue of $\varepsilon^2 \varepsilon$, and w is the corresponding weight value. Suppose that $\delta w = w' - w$, then we have the following function:

$$E\|\delta w\|^2 \leq \frac{k(k+3)}{2\lambda_{\min} l} E\|w'\|^2. \quad (11.5)$$

Through the above theoretical analysis, we know that noise consists of two types; one includes the weak spots of the original sample points and the other one does not belong to the former type completely. For the first case, there are many improved methods. Many algorithms consider the first case because they cannot ignore this situation. If we add a regularisation item, we can make the local linear embedding algorithm more robust [3]. The local tangent space alignment algorithm has good anti-noise capacity [4]. The local smoothing algorithm can be regarded as the pre-processing method by using the locally weighted algorithm [5]. These improved methods all focus on the first case. However, this article adopts a new method to eliminate the effect of the second type of noise for maintaining the original advantage. This section regards such noise as a new type of manifold data structure. We separate the noise from the original sample data and handle it. Therefore, we introduce the connection learning algorithm based on frame bundle.

11.2 Longitudinal space connection learning model based on frame bundle

In general, there is no global frame field on manifold M with dimension m . Nonetheless, the affine connection exists. Therefore, there is a global frame field in the frame bundle P . We can handle multiple manifold structures from this perspective, which is easier than handling the manifold M .

Mesh indexing is a hot research topic at present, and the LTMADS algorithm is a well-known algorithm in this research field [5, 6]. Many studies have attempted to improve the performance of existing algorithms. By using the global frame of the frame bundle, we can improve the convergence of the algorithm as follows:

Algorithm 11.1:

Common algorithm

- (1) Initialise $k=0$. Set $x_0 \in \mathbb{R}^m$ as the initialisation point and set $\beta > 0$;
- (2) Set $N_k > 0, H_k > 0$;
- (3) Construct Φ_k and $h_k > H_k$ by the mean value zero;
- (4) Set $x_t \in \Omega_k = S_k \cup \{p_k\} \cup \Phi_k$ and S_k is the set of points chosen for the following:
 - a) $f(x_t) < f_k - N_k(h_k)^{1+\beta}$,
 - b) x_t is the optimum point in Ω_k and set $k = k + 1$;
- (5) If $f(x_t) < f_k - N_k(h_k)^{1+\beta}$, we should set $x_{k+1} = x_t$ and go to step 3. If not, we should continue to loop and choose x_{k+1} as the optimum point;
- (6) If the algorithm converges, it should be terminated. If not, go to step 2.

Algorithm 11.2:

Algorithm on manifold

- (1) Initialise $n = k = 0, \Delta_0^m, \Delta_0^p$. Set $p_0 \in M$ as the initialisation point and make $\beta, \delta > 0$;
- (2) Make D_k ;
- (3) Construct $p_t \in \Omega_k = \{S_k\} \cup \{p_k\} \cup \Phi_k$ and make
 - a) $f(p_t) < f_k - \delta(\Delta_k^m)^{1+\beta}$,
 - b) $p_t \in \Omega_k$ is the optimum point.
- (4) If $f(p_t) < f_k - \delta(\Delta_k^m)^{1+\beta}$, set $k = k + 1$ and $p_{k+1} = p_t$. Update Δ_k^m, Δ_k^p and go to step 2;
- (5) If not, set $z_n = p_k$ and $k = k + 1, n = n + 1, p_{k+1} = \arg \min\{f(p_k), f(p_t)\}$. Update Δ_k^m, Δ_k^p ;
- (6) If the algorithm converges, it should return p_k . If not, go to step 2.

This algorithm can replace the traditional method. By introducing the frame structure on the manifold, the convergence of the traditional method can be enhanced considerably.

This section presents the longitudinal space connection learning model based on frame bundle, which is an improved version of the traditional algorithm. This algorithm directly uses the local-tangent projection and adds the projection mapping of the global frame field. The basic algorithm process is as follows. In Algorithm 11.3, we have $x' = x'', \nabla = \exp_x(\Delta) = x^{1/2} \exp(x^{1/2} \Delta x^{-1/2})x^{1/2}$.

Algorithm 11.3:

Input: a tensor set X ;

Output: the internal distance of the tensor set t ;

- (1) Obtain the tensor subspace B_m by utilising the improved dimension reduction algorithm;
- (2) Obtain the m linearly independent tangent vectors $\{e_1, e_2, \dots, e_m\}$ of every point in the tensor subspace B_m ;

- (3) Obtain the corresponding tangent vector of every point in the tensor subspace by using the logarithmic function, which is the tangent space mapping;
 - (4) Transform every tangent vector into the same coordinate system by using the contact operator of the frame bundle.
 - (5) Obtain the distance between each pair of tangent vectors by using the tangent vectors in the same space as the metric function.
 - (6) Perform classification by combining the linear discriminant function.
-

11.3 Horizontal space connection learning model based on frame bundle

We have mentioned that the longitudinal space field satisfies the Pfaff equations, $\theta^i = 0$, $1 \leq i \leq m$, in the longitudinal space field V that is defined on the manifold M . In this section, we will discuss the variational form of the Pfaff equations, $\theta_j^i = 0$, $1 \leq i, j \leq m$, in the tangent subspace H of every point P [7].

The tangent space $T_x(p)$ can be decomposed into the direct sum form of every point $x \in p$.

$$T_x(p) = V(x) \oplus H(x). \quad (11.6)$$

Here, the horizontal space $H(x)$ and the tangent space $T_p(M)(p = \pi(x))$ on manifold M are isomorphic. The horizontal space H is invariable by moving $L_a(a \in \text{GL}(m; R))$, which means that for each point $x \in p$, we have

$$(L_a)_* H(x) = H(L_a(x)). \quad (11.7)$$

The horizontal space field H , which is determined by the affine connection ∇ , always has the two above-mentioned properties. Therefore, we utilise the two properties to propose the improved algorithm. In the last section, we introduced the multiple manifold structure data classification by using the global coordinate of the longitudinal space. We improved the algorithm performance by increasing the computation time. The dimension of the whole data set changed from m to m^2 . For changing this phenomenon, we employ the horizontal space field of the tangent space. Because the horizontal space field is the tangent subspace with m dimensions, we can reduce the time complexity by reducing the dimension of the original data set.

Algorithm 11.4:

Horizontal space connection learning model based on frame bundle

- (1) Represent the original data set by the tensor structure;
 - (2) Obtain a low-dimensional embedding data set by the dimension reduction algorithm;
 - (3) Make every point in the data set satisfy $\theta_j^i = 0$ and obtain the m lines of every point based on the longitudinal space model;
 - (4) Move all the tangent vectors into the same frame field by using the contact operator of the frame bundle;
 - (5) Calculate the distance between every pair of points.
-

By analysing the above algorithm model, we find that the time complexity decreases somewhat. The specific algorithm process is as follows:

Algorithm 11.5:

Input: a tensor set X ;

Output: the internal distance of the tensor set t ;

- (1) Obtain the tensor subspace B_m by utilising the improved dimension reduction algorithm;
 - (2) Obtain the m linearly independent tangent vectors $\{e_1, e_2, \dots, e_m\}$ of every point in the tensor subspace B_m ;
 - (3) Obtain the corresponding tangent vector of every point in the tensor subspace by using the logarithmic function, which is the tangent space mapping;
 - (4) Transform every tangent vector into the same coordinate system by using the contact operator of the frame bundle.
 - (5) Obtain the distance between each pair of tangent vectors by using the tangent vectors in the same subspace as the metric function.
 - (6) Perform classification by combining the linear discriminant function.
-

11.4 Related applications

The COIL-20 data set has 20 grey objects. The size of each object is 128*128 and we have 112 images. By post-processing, each image can be changed to 80*80. Some sample images are shown in Figure 11.4. Compared with the LPP, NPE, and MLLE algorithms, we randomly choose 36 images of each object as the training samples and regard the rest of the images as the testing samples. The LPP, NPE, and MLLE algorithms use the nearest neighbour classifier, and their accuracy and subspace dimensions are shown in Figure 11.5. From the figure, we can see that the accuracy is enhanced considerably.

In this paper, the basic data set is based on the original data set, and the classification comparison is made by changing the feature dimension of the original data set. Our analysis shows that in the beginning, our algorithm classification performance is weak, but with the increase in the joining of the data sets, the anti-noise ability of the four algorithms is enhanced gradually, and although the algorithm shows some volatility, the numerical method shows advantages as the amount of data increases. In this paper, the COIL-20 data sets were changed in turn from 5, 10, 15, 20, 25, 35, and 45. The results of these algorithms are shown in Figure 11.5

First, we use the artificial data set to test the proposed frame bundle on a horizontal space. Using the longitudinal space learning algorithm, we construct two tensor sets, and the initial data values are taken from a $(0, 1)$ Gaussian distribution; the dimensions are set as $10 * 8$. To construct the manifold data structure, we use the corresponding multiplication for each value of the Gaussian distribution $[(I_1 * I_2)/(i_1 * i_2)]^f$ and we have $f = 1/2, 1/4$. We set the dimension to be 10. To construct the manifold data structure for each value according to the Gaussian distribution, we adopt the corresponding multiplication among them, and in view of the noise, we use an average of



Fig. 11.4: The image examples in the COLT-20 dataset.

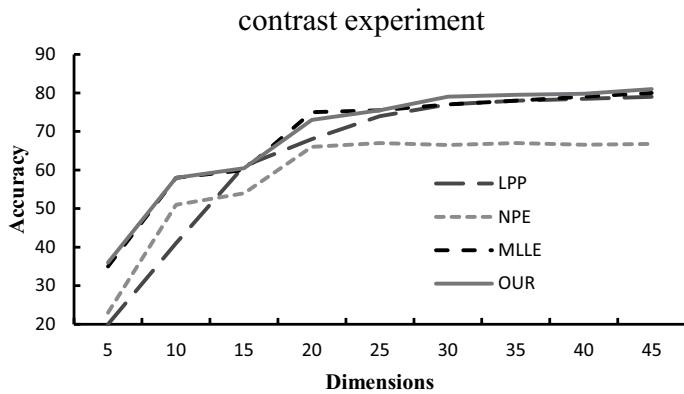


Fig. 11.5: The accuracy scores which use LPP, NPE, MLLE, and our method.

1 and a variance of 0.01 for the corresponding multiplication of the Gaussian distribution. For these two manifolds, 200 test sets are taken and 50 test sets are collected. We compare the existing literature with LPP and UDP. The specific experimental results are shown in Figure 11.6.

contrast experiment

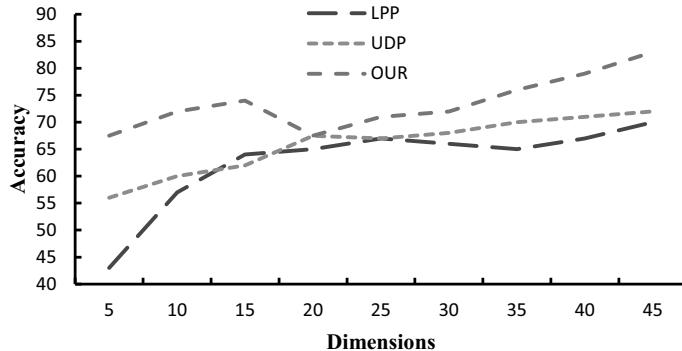


Fig. 11.6: The accuracy scores which use LPP, UDP and our method.

11.5 Summary

We solved the problem of the manifold structure for large-scale data, and we combined the frame bundle on the transverse space. The longitudinal space contact operator was given on the basis of the frame bundle longitudinal space contact learning algorithm. Using the longitudinal space, we further improved the algorithm and introduced the frame plexus of the transverse space contact learning operator. We also gave the detailed algorithm process.

Bibliography

- [1] Vasilescu M, Terzopoulos D. Multilinear analysis of image ensembles: Tensorfaces. Proc. 7th Eur. Conf. Comput. Vis, May 2002, 447–460.
- [2] Dreisigmeyer DW. Los Alamos National Laboratory direct search algorithms over Riemannian manifolds. IEEE, January 8, 2007, 12, 1101–1121s.
- [3] Zhang ZY, Zha HY. Principal manifold and nonlinear dimensionality reduction via tangent space alignment. SIAM Journal of Scientific Computing, 2005, 26(1), 313–338.
- [4] Cox TF, Cox MA. Multidimensional Scaling. Chapman & Hall. London, 2001.
- [5] Jenkins O. CandMataric MJ. A spatio-temporal extension to Isomap nonlinear dimension reduction, ICML: Twenty-first International Conference on Machine Learning, New York, NY, USA, ACM Press (2004).
- [6] DeSilva V., Tenenbaum JB. Global Versus Local Methods in Nonlinear Dimensionality Reduction, Advances in Neural Information Processing Systems [J](NIPS), MIT Press, Cambridge, MA, 705–712, 2003.
- [7] Jian Y, David Z. Two-Dimensional PCA: A New Approach to Appearance-Based Face Representation and Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004, 26(1).

12 Spectral estimation learning

12.1 Concept and definition of spectral estimation

Spectral analysis is a very important part of manifold learning. Herein, the related content of the spectral analysis method is briefly analysed.

12.1.1 Research background of the spectral estimation method

In recent years, the spectral analysis method has achieved good results in signal processing and image segmentation. Thus, spectral estimation learning methods have attracted considerable interest. As a classical mathematical analysis and algebraic method, the spectral analysis method already has a complete theoretical foundation. It has a very wide range of applications in low-dimensional representation and clustering of high-dimensional data. Spectral analysis also plays a crucial role in many unsupervised learning algorithms, such as image segmentation, data mining, data compression, and speech recognition. Therefore, spectral estimation learning methods have gradually emerged as important techniques in machine learning research and key technologies for solving problems. A number of spectral estimation clustering methods based on similarity matrix eigenvalue decomposition, have been proposed by some researchers. These methods are considered to be very effective, and spectral clustering and dimensionality reduction methods have been widely validated.

12.1.2 Concept and definition of spectral estimation

According to the given sample data set, the spectral method defines a relation matrix describing the similarity between pairs of data points, and calculates their eigenvalues and eigenvectors. Then, suitable feature vectors are selected to project the low-dimensional embedding of the data. The spectral estimation method is only concerned with the number of data points; it does not depend on the dimension of the manifold. Furthermore, it can avoid the singularity caused by high-dimensional feature vectors.

If the similarity matrix is defined on a given graph, e.g., the adjacency matrix and the Laplacian matrix on the graph, the method is called the spectral graph method [1]. By studying the spectrum of the matrix defined by the graph, the geometric information of the manifold contained in the graph is obtained by the spectral graph theory [2]. The connection between discrete and continuous spaces has been established by geometric, optimisation, and algebraic techniques.

The spectra of Laplace operators are closely related to the geometrical and topological properties of manifolds. Spectral estimation is defined on the basis of minimum and second minimum eigenvalues of the Laplace-Beltrami operator of the data similarity matrix [3]. The key of the spectral estimation method is to choose the proper kernel function as the criterion for choosing the similarity matrix. Because there is no complete theoretical basis for selection in the existing theory, it usually needs to be built on the basis of a suitable hypothesis. The Mercer kernel is used as the selection criterion, and the simplest one is the Gaussian kernel function. At present, there are few manifolds, and their spectra have been calculated. Most of the manifolds cannot be calculated efficiently, and the spectral estimation method is used to study the internal geometry of manifolds.

12.1.3 Research progress in learning methods of spectral estimation

At present, researchers have obtained rich results in the study of Laplace operator spectra on manifolds, and spectral geometry has emerged as an important branch of large-scale geometric analysis. Compared with the iterative method, the spectral method can find the inner structure of complex manifolds and cluster on any shape of the sample space.

Different criteria functions and similarity measures are chosen, and different spectral estimation learning methods are obtained. They can be divided into two categories: iterative spectral clustering (two-channel spectrum clustering) and multi-channel spectral clustering. Iterative spectral clustering algorithms include the PF algorithm [4], SM algorithm [5], SLH algorithm [6], KVV algorithm [7], and Mcut algorithm [8]. Multi-spectral clustering algorithms mainly include the NJW algorithm [9] and MS algorithm [10].

The PF algorithm [4] was proposed by Perona and Freeman in 1998. The first feature vector x_1 of similarity matrix W is used for clustering. The first eigenvector is the eigenvector corresponding to the largest eigenvalue λ_1 of the similarity matrix. As the earliest spectral learning algorithm, the PF algorithm shows that for the block diagonal similarity matrix, the nonzero elements in the eigenvector x_1 belong to the same class, and the corresponding points of the zero elements belong to the other class [11]. The SM algorithm [5] was proposed by Shi and Malik in 2000. The normalised similarity matrix is chosen to solve the first two characteristic vectors of $Lx = \lambda Dx$ by using the regularisation theorem. Clustering is then performed. The clustering result of the SM algorithm is better than that of the PF algorithm. The SLH algorithm [6] was proposed by Scott et al. The similarity matrix W and parameter k are used as input. The first k eigenvectors of the similar matrix W are solved, and a new matrix Q is obtained as the output. On the basis of this work, the KVV algorithm, Mcut algorithm, etc, have been proposed to reflect the similarity between the data points more accurately through continuous improvement of the similarity matrix W .

Experiments have shown that better clustering results can be obtained if more feature vectors are used for clustering and direct computation of multiple segmentation [11, 12]. Ng et al. proposed the NJW algorithm [9] in 2001, and chose the canonical Laplacian matrix $L = D^{-1/2} LD^{-1/2} = I - D^{-1/2} WD^{-1/2}$ as a similarity matrix, where D is the degree matrix. By finding the first k eigenvectors of the Laplacian matrix L and normalising it, the new matrix Y is obtained. Each row of Y is clustered as a point. The NJW algorithm overcomes the shortcomings of the SM algorithm when the overall degree of each cluster is different and the effect is not ideal [9]. In 2001, the MS algorithm [10] was proposed by Meila and Shi, who interpreted the similarity as the random walk probability in the Markov chain. The random walk probability matrix $P = D^{-1} W$ is used to replace the similarity matrix W , and clustering is carried out by solving the eigenvectors of the random walk probability matrix.

In recent years, researchers have proposed the self-tuning algorithm [13], B-matching spectral clustering algorithm [14], path-based spectral clustering algorithm [15], etc, which are superior to the above-mentioned algorithms, as they improve the Gaussian kernel function, optimise the similarity matrix, avoid the manual parameter error, automatically determine the number of clusters, and so on.

12.2 Relevant theoretical basis

As an unsupervised learning method, spectral estimation learning is derived from the theory of spectral graph partitioning. The low-dimensional representation and clustering results are obtained according to the criterion of partitioning [16]. The spectrum estimation learning method involves data with a number of sub-classes within the entire data set based on the description method. Each sub-class of data points has inherently high similarity; however, differences also exist between each sub category. The essence of the spectral estimation learning method is to solve the similarity matrix, degree matrix, and eigenvalue decomposition of the Laplacian matrix. Therefore, we can consider that the study of spectral graph theory is the study of the partition criterion approximation of graphs [17].

The existing spectral learning method is based on the definition of a criterion function. By searching for the optimal classification, the criterion function is optimised, and the optimal classification is achieved. In the spectral estimation algorithm, the data sample point is regarded as the vertex V in the graph, and a similarity matrix describing the paired data points is defined. The value of the edge E between vertices is W , and a graph $G = (V, E)$ corresponding to the data point is established. Then, we select the appropriate feature vectors for mapping and obtain the low-dimensional mapping of high-dimensional data. Both graphs and manifolds can be embedded into Euclidean space. To obtain the geometric topological property of manifolds, we only need to know the graphs corresponding to them. Their topological properties are represented by the weights of edges. Therefore, the rational definition of edges in

graphs can fully approximate the low-dimensional embedding of high-dimensional data manifolds.

12.2.1 How to construct a similarity matrix

From the above brief analysis of manifold learning problems, it is easy to see that the first step is to define the similarity between data points. A similarity measure between two data points is used to measure whether the data belong to the same subclass. The most commonly used similarity measures are the distance between the samples, the angle between the samples, and the volume. Similarity matrices (or affinity matrices) are usually represented by W . The Gaussian kernel function is used to define the similarity matrix as follows:

$$W_{ij} = \exp\left(-\frac{d(x_i, x_j)}{2\sigma^2}\right) \quad (12.1)$$

where $d(x_i, x_j)$ is calculated by the similarity measure, and the Euclidean distance is usually chosen as the similarity measure, i.e., $d(x_i, x_j) = \|x_i - x_j\|^2$, where σ is the input parameter.

Suppose that the Euclidean distance is chosen as a similarity measure. If two samples belong to the same class, then the distance between them is much smaller than that between sample points in different classes. Select a suitable threshold d_0 to determine whether the two sample points belong to the same category. For example, the Euclidean distance is chosen as a similarity measure. In the data clustering of three classes of samples, three different d_0 are given. It can be seen that the smaller the d_0 , the smaller are the classes, and the greater is the number of classes. The data are as follows:

$$\begin{aligned} R_1 &= \{(x, y) \in R^2 \mid 0 \leq x \leq 5, 3 \leq y \leq 5\}, \\ R_2 &= \{(x, y) \in R^2 \mid 0 \leq x \leq 2.5, 0 \leq y \leq 3\}, \\ R_3 &= \{(x, y) \in R^2 \mid 2.5 \leq x \leq 5, 0 \leq y \leq 3\}, \end{aligned}$$

100 samples are uniformly sampled in R_1, R_2, R_3 , respectively. Different thresholds affect the number of classes and the size of the class, as shown in Figure 12.1.

Definition 12.1. (Geodesic) [22] Suppose that $\xi = \{TM, M, \pi, GL(n, R), R^n, \varepsilon\}$ is the C^∞ tangent bundle for M . If $\nabla_{y'} y' = 0$, then y is a geodesic. If a geodesic is not the true limit of any geodesic, it is the largest geodesic.

Of course, other distance functions can also be chosen as similarity measures, such as Minkowski metrics and Mahalanobis metrics. More generally, when choosing the right similarity measure, we can choose the similarity function without distance measurement. It can be seen that the choice of the similarity measure is sufficient to show that the spectral estimation learning methods involve a wide range of fields and various solutions.

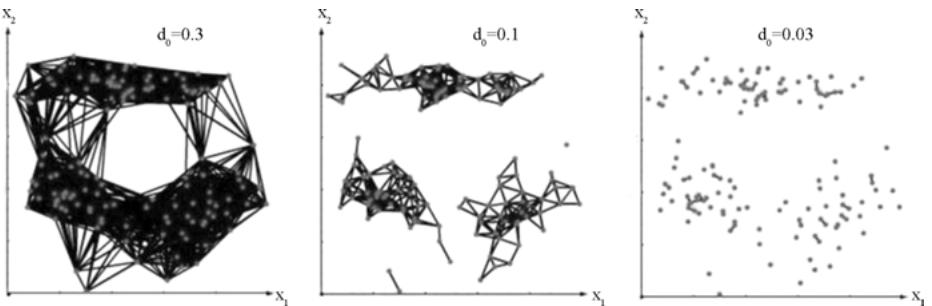


Fig. 12.1: The effect of the distance threshold d on the number of classes and the size of the class.

Next, we discuss the choice of the similarity matrix parameter σ . Since the input parameters are artificially set, there is a considerable amount of uncertainty and instability. The NJW algorithm [9] is used to select the best σ of the clustering results as parameters by repeatedly running the NJW algorithm. Despite the elimination of human input, it is still a considerable time cost. At present, many scholars are studying the selection of the parameters of σ , which are usually determined by experience. A uniform standard has not been formed, and the selection standards cannot be given quantitatively. They are limited in the scope of their respective fields of knowledge.

According to the flexibility of parameter selection, researchers have been trying to determine the parameter σ automatically according to the situation of the data set itself. In 2004, Zelnik-Manor et al. proposed a self-training (self-tuning) algorithm [18]. According to the geometric characteristics of each data point, the ST algorithm computes an adaptive parameter $\sigma_i = \|x_i - x_p\|$ for each data point x_i by neighbourhood information (x_p is the k th nearest neighbour of point x_i). Based on this idea, the similarity matrix is transformed into

$$W_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma_i \sigma_j}\right). \quad (12.2)$$

Similarly, Fisher et al. proposed a method for dynamically selecting the parameter σ in 2005 [19]. By solving $\sum_{j=1}^n \exp(-\|x_i - x_j\|^2 / 2\sigma_i^2) = \tau$, where τ is a constant, a parameter σ_i is determined for each data point x_i . Here, the similarity matrix is defined as

$$W_{ij} = \min \left\{ \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right), \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_j^2}\right) \right\}. \quad (12.3)$$

In summary, constructing a similar matrix depends on choosing not only the appropriate similarity measure but also the parameter σ that suits its own data set. This will become a hot and complex research topic in future spectral estimation learning methods.

12.2.2 How to choose the appropriate laplacian matrix

The Laplacian matrices used in the spectral estimation learning method are obtained by the deformation of the similarity matrix and the degree matrix. The commonly used Laplacian matrices have three forms:

$$L = D - W \quad (12.4)$$

$$L_{\text{sym}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \quad (12.5)$$

$$L_{\text{rw}} = D^{-1} L = I - D^{-1} W \quad (12.6)$$

where W is a similar matrix, D is a degree matrix (or diagonal matrix), $D_{ii} = \sum_{j=1}^n w_{ij}$, and I is the identity matrix.

At present, the specific environment of these three forms of Laplacian matrix has not been solved well. In [20], Brand et al. proposed a preliminary research scheme. If the diagonal element d_{ii} in degree matrix D is approximate, then the Laplace and dimensionality reduction results obtained by choosing any form of the matrix are basically the same. By contrast, the difference is considerable.

12.2.3 Selecting the appropriate feature vector

In the study of manifold dimensionality reduction and clustering with the spectral estimation method, choosing the eigenvectors to be mapped and selecting and calculating the eigenvectors are difficult problems in the study of spectral theory. In [21], it has been shown that not every feature vector contains the geometric features of the cluster space or the low-dimensional space. If a useless or irrelevant feature vector is chosen, severe dimensionality reduction or clustering failure might occur. For example, three sample data sets with overlapping portions are shown in Figure 12.2(a).

$k = 1, k = 2, k = 3$ feature vectors are selected for clustering, and the clustering results are shown in Figure 12.2(b)–(d). It can be seen that different feature vectors are selected for clustering, and their effects are different.

For researchers, it is only necessary to select characteristic vectors that contain manifold geometric characteristics. The original spectral estimation learning methods usually choose the eigenvectors of the first k eigenvalues of the Laplacian matrix to solve them. These methods are based on the assumptions of no noise interference and non-sparse data sets. However, in the presence of noise and data sets in real data, such a selection method is prone to error. In this section, an improved feature vector algorithm is proposed, which not only can select the appropriate number of feature vectors, but can also be applied to data clustering with noise.

The spectra of manifolds cannot be obtained by their eigenvalues, but they need to be obtained by other manifold methods. For example, consider the case when we want to reduce the dimensions of the face images of a person aged 40. The spectral

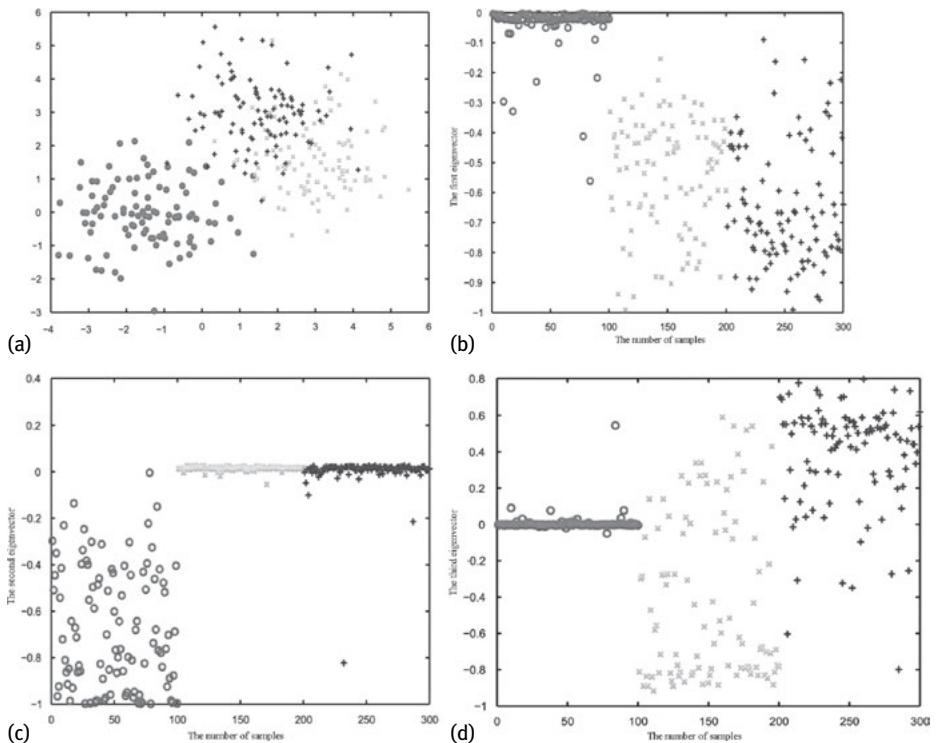


Fig. 12.2: (a) Sampling points in three overlapping intervals are uniformly sampled. (b) First feature vectors are selected for clustering. (c) Second feature vectors are selected for clustering. (d) Third feature vectors are selected for clustering.

estimation method is used to solve the feature vector, and the face images at the age of 30 can be solved indirectly. The idea of Laplacian eigenvalue estimation proposed in this paper is based on the theory of differential geometry and equal spectrum. First, the mathematical definitions and theorems of the related concepts are given.

To find a map that best maintains the local characteristics of the data, the objective function is defined as

$$\arg \min_{\|f\|_{L^2(M)}=1} \int_M \|\nabla f(x)\|^2$$

Suppose that $Lf \stackrel{\text{def}}{=} -\operatorname{div} \nabla(f)$ (L is a Laplacian matrix). The total eigenvalues of L are $\lambda_1, \lambda_2, \dots$, where $0 = \lambda_0 \leq \lambda_1 \leq \lambda_2 \leq \dots$, and f_i is the characteristic function corresponding to the eigenvalue λ_i . It is easy to see that the manifold is mapped to an independent point, requiring the embedded mapping f to be orthogonal to f_0 , and λ_1 is called the first eigenvalue.

Isometric manifold spectra must be equal and vice versa. The essence of the spectral problem is the inverse problem of spectral geometry. The geometric properties of

manifolds are determined by spectral values as well. After adding certain conditions, equal spectra can be derived to be isometric.

We can get some properties of equal spectrum manifolds from the asymptotic expansion theorem of Minakshisundaram-Pleijel in differential geometry. Some geometric and topological properties are given by equal spectra.

Theorem 12.1. [22] Suppose that $\text{spec}(M, g) = \text{spec}(\tilde{M}, g)$. Then, $\dim M = \dim \tilde{M}$, $\text{vol}M = \text{vol}\tilde{M}$.

Proof. Since $\text{spec}(M, g) = \text{spec}(\tilde{M}, g)$, then $\lambda_i = \tilde{\lambda}_i$, $i = 0, 1, 2, \dots$, i.e.,

$$(4\pi t)^{\frac{n}{2}} \sum_{i=0}^{+\infty} a_i t^i \sim \sum_{i=0}^{+\infty} e^{-\lambda_i t} = \sum_{i=0}^{+\infty} e^{-\tilde{\lambda}_i t} \sim (4\pi t)^{\frac{\tilde{n}}{2}} \sum_{i=0}^{+\infty} \tilde{a}_i t^i .$$

From this, we can see that

$$\dim M = n = \tilde{n} = \dim \tilde{M} ,$$

$$\text{vol}M = a_0 = \tilde{a}_0 = \text{vol}\tilde{M} .$$

In other words, if the spectra of the two manifolds are the same, then their internal geometric structure is the same. Therefore, we can obtain the eigenvalues of the 40-year-old face image by solving the eigenvalues of the face image of the same person at the age of 30, as shown in Figure 12.3. \square

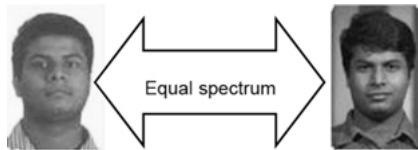


Fig. 12.3: Two photos of a person at different ages.

12.2.4 Determining the number of clusters

The number of clusters directly influences the clustering results. There are two ways to deal with the problem of the number of unknown clusters. One is to try different c values several times, compare the values of the criterion functions under different c , and select the c of the criterion function value as the number of clusters. The other is to automatically determine the number of clusters. Examples include Rcut-based spectral clustering algorithms [23], nonlinear dimensionality reduction (NLDR) algorithms [24], and distance-based heuristic methods [20].

Brand et al. proposed a new approach of angular similarity vector contraction between data points belonging to the same class and angular similarity vector stretching between data points that do not belong to the same class. Specifically, the vector

strictly increases with the decrease in dimensionality in the process of dimensionality reduction [20]. In [20], the author used the azimuth as the similarity measure and experimented on the Swiss roll data set. It was found that owing to the small distance between the data, especially at points at both ends of the data manifold, the similarity of non-local neighbours was obtained. Therefore, the azimuth angle similarity is used to avoid the distortion when the above-mentioned data are reduced in dimension. The heuristic learning method proposed in [20] is used in this paper. The clustering centre is used to represent the class, and it is updated according to the current value of the class centre and the new data set.

12.3 Synchronous spectrum estimation learning algorithm

In the last section, we mainly studied the similarity between sample points on image feature manifolds. Next, we perform a systematic analysis of another important problem, i.e., the criterion function of the spectral estimation learning method. The learning problem of the image feature manifold can be given as a mathematical model by defining the criterion function, i.e., the optimal division among the sub-classes of the sample points is found by solving the optimal solution of the criterion function. Thus, the similarity in each sub-class of the sample is the largest, and the similarity among the subclasses is the smallest [25].

12.3.1 Graph optimisation criterion for locally preserving mappings

Assume that a uniform sample set of samples is given by $X = [x_1, x_2, \dots, x_n]$, where $x_i \in R^D$, $i = 1, 2, \dots, n$. Here, we use the objective function defined in [26] as the basis for improvement:

$$\min_W \frac{\sum_{i,j=1}^n \|W^T x_i - W^T x_j\|^2 P_{ij}}{\sum_{i=1}^n D_{ii} \|W^T x_i\|^2} \quad (12.7)$$

where $W \in R^{D \times d}$ ($d < D$) is a similarity matrix, $D_{ii} = \sum_{j=1}^n P_{ij}$, and $P = (P_{ij})_{n \times n}$ is a weight matrix of adjacency graphs. Here, a graph optimisation problem is constructed. By solving the following objective function, the optimal partition and local mapping problem are solved:

$$\begin{aligned} \min_{W, S_{ij}} & \frac{\sum_{i,j=1}^n \|W^T x_i - W^T x_j\|^2 S_{ij}}{\sum_{i=1}^n \|W^T x_i\|^2} + \eta \sum_{i,j=1}^n S_{ij} \ln S_{ij} \\ \text{s.t. } & \sum_{j=1}^n S_{ij} = 1, S_{i,j} \geq 0 \quad (i, j = 1, \dots, n) \end{aligned} \quad (12.8)$$

where $S = (S_{ij})_{n \times n}$ is an unknown similarity weight matrix, $\sum_{i,j=1}^n S_{ij} \ln S_{ij}$ is a regularisation matrix, and η is a harmonic parameter. In the literature [27], it has been shown

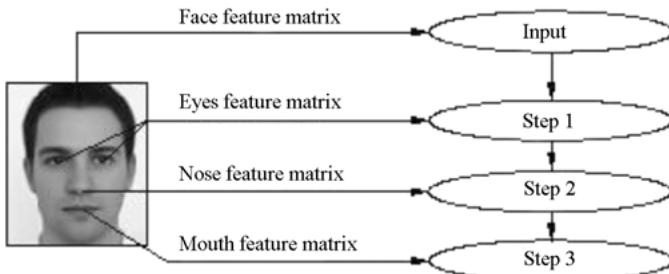


Fig. 12.4: Asynchronous Spectrum Estimation Learning Steps.

that for this optimisation problem, its iterative method can prove that the process is convergent. The purpose of manifold learning is to solve the optimisation problem, find the mapping matrix, and reconstruct the new coordinates of the sample.

There are two methods for solving the objective function, namely synchronous and asynchronous (Fig 12.4). By solving the criterion function of (12.8), the dimension reduction and clustering problem is approximated as a discrete optimisation problem: Find the optimal partition of the set of samples so that the criterion function takes the optimal solution. Because the samples are sampled in the image, the sample points are limited. Theoretically, the best solution can be found by using the lifting method, but for high-dimensional data, the time complexity of the lifting method is too high; hence, it is not suitable in the field of computer research. Therefore, to reduce the time complexity and space complexity of the algorithm, the idea of iteration and hierarchy is introduced into the study of the spectral estimation learning algorithm. For example, in face recognition, we can learn the partial structure of the human face separately, but also learn the whole structure of the human face at the same time.

12.3.2 Asynchronous spectrum estimation learning model

The idea of asynchrony originates from iterative thinking, and iterative optimisation is one of the most commonly used methods in optimisation theory for seeking optimal solutions. The basic idea is as follows. First, set a parameter partition (the parameter is artificially set), and then adjust the subclass that each sample belongs to, so that the adjusted criterion function is better than that before adjustment. These algorithms can only guarantee local optimisation rather than global optimisation, i.e., the local geometric structure of the manifold can be maintained, but the effect of maintaining the overall structure and global robustness of the manifold is not ideal.

This section improves the basis of the SPP algorithm [28], and the asynchronous spectrum estimation learning algorithm is obtained. The SPP algorithm is an unsupervised algorithm based on graph reconstruction. First, the sample points are selected from the sample space $X \setminus \{x_i\}$ to be as few as possible to reconstruct the sam-

ple point x_i . In different noise environments, different minimal l_1 norms can be constructed as follows [28]:

$$\begin{aligned} & \min_{S_i} \|S_i\|_1 \\ \text{s.t. } & \|x_i - XS_i\|^2 < \varepsilon \left(\sum_{j=1}^n S_{ij} = 1 \right) \end{aligned} \quad (12.9)$$

where $\|x_i - XS_i\|^2$ denotes the reconstruction error. The SPP algorithm constructs an adjacency graph $G = (X, (S_{ij})_{n \times n})$ to describe the relation between a sparse mapping and the sample points. The sparse graph can be well maintained by finding the mapping matrix W . Similar to the NPE algorithm and the LLE algorithm, the SPP algorithm solves the following objective function to get W :

$$\min_W \frac{\sum_{i=1}^n \|W^T(x_i - XS_i)\|^2}{\sum_{i=1}^n \|W^T x_i\|^2} \quad (12.10)$$

which is equivalent to solving the following problem of matrix traces:

$$\max_W \frac{\text{tr}(W^T X S_\beta X^T W)}{\text{tr}(W^T X X^T W)} \quad (12.11)$$

where $S_\beta = S + S^T - SS^T$ can be solved by generalised eigenvalues. Although the SPP algorithm can use the matrix trace as far as possible to balance the weight on each side of the graph, some useful information of the samples is lost.

The Laplacian eigenvalues (LPE) algorithm proposed in this section is based on the SPP algorithm. The approximation of the sample points by the local tangent space not only ensures the convergence of the local minimum error but also maintains the local geometric characteristics of the manifold. The asynchronous spectrum estimation manifold learning algorithm is modelled as follows:

$$\begin{aligned} \min_{W, S_i} J(W, S) &= \frac{\sum_{i=1}^n \|W^T(x_i - XS_i)\|^2}{\sum_{i=1}^n \|W^T x_i\|^2} + \sum_{i=1}^n \lambda_i \|S_i\|_1 \\ \text{s.t. } & W^T W = I \end{aligned} \quad (12.12)$$

where $W \in R^{D \times d}$ ($d < D$) is a mapping matrix, the sample point set is $X = \{x_1, x_2, \dots, x_n\}$, $x_i \in R^D$, $(S_{ij})_{n \times n}$ is a similarity matrix with diagonal 0, S_i is the i th column of the reconstructed coefficient matrix S of the sample point x_i , and λ_i is a harmonic parameter. λ_i is also the main eigenvalue of matrix $L = S^{-1}L = I - S^{-1}W$, and I is a $d \times d$ identity matrix. The purpose of this method is to avoid the non-unique solutions of the matrix trace by solving the regularisation problem.

Initialisation:

Initialise $W = I^2$, and model (12.12) is simplified as

$$\min_{S_i} \frac{\sum_{i=1}^n \|x_i - XS_i\|^2}{\sum_{i=1}^n \|x_i\|^2} + \sum_{i=1}^n \lambda_i \|S_i\|_1 .$$

It can be further simplified to the following form:

$$\min_{S_i} \sum_{i=1}^n \|x_i - XS_i\|^2 + \sum_{i=1}^n \bar{\lambda}_i \|S_i\|_1 \quad (12.13)$$

where $\bar{\lambda}_i$ is a constant factor of λ_i . As can be seen, the problem (12.13) has been transformed into a simple sparse mapping problem.

Iterative steps:

By solving the problem (12.13), the graph S is obtained, and the optimisation problem (12.12) is converted into

$$\begin{aligned} & \min_{W, S_i} \frac{\sum_{i=1}^n \|W^T(x_i - XS_i)\|^2}{\sum_{i=1}^n \|W^T x_i\|^2} \\ & \text{s.t. } W^T W = I \end{aligned}$$

which is equivalent to

$$\begin{aligned} & \min_W \frac{\text{tr}[W^T X(I - S - S^T + SS^T)X^T W]}{\text{tr}(W^T X X^T W)} \\ & \text{s.t. } , W^T W = I . \end{aligned} \quad (12.14)$$

This is a regularised matrix trace problem, and its global optimal solution can be obtained by an efficient iterative algorithm. For example, Y. Gao et al. proposed a generalised Fourier transform algorithm based on the Fisher discriminant [29], and H. Wang et al. [30] proposed the trace ratio method. This section adopts the efficient trace ratio method proposed by Y. Jia et al. [31], whose effectiveness has been demonstrated.

In combination with W obtained in the first step and $S = (S_{ij})_{n \times n}$ obtained from the optimisation problem, the objective function can be simplified to the following form:

$$\min_{S_i} \sum_{i=1}^n \|W^T(x_i - XS_i)\|^2 + \sum_{i=1}^n \lambda_i \|S_i\|_1 . \quad (12.15)$$

By solving (12.13) and (12.15), the above optimisation problem is completely transformed into a sparse reconstruction problem.

12.3.3 Synchronous spectrum estimation learning algorithm

It is undeniable that the asynchronous spectral estimation learning algorithm can learn the local geometric structure of the image feature manifold very well. In other

words, the eye structure of the face image can be learned first, followed by the mouth, nose, and other structures, step by step. However, to guarantee the global optimum of the manifold, it is more favourable for face manifold learning to learn the whole structure of the eyes, nose, mouth, and so on. Based on this global idea, a manifold learning method called Laplacian eigenvalue estimation (LPE) is proposed in this section.

Suppose that the sample set $X = [x_1, x_2, \dots, x_n]$ belongs to class r , and the LPE algorithm maps each data point to different strip plane curves. The algorithm is inspired by the LE algorithm, and its solution space is constrained by the similarity matrix and subspace. The result of each mapping is a column of the matrix $Y \in R^{m \times r}$, and the i th row of the matrix is the embedded coordinate of the i th point.

Algorithm 12.1: (Improved synchronous LPE algorithm)

Input: Data set $X = [x_1, x_2, \dots, x_n]$, regularisation parameter λ_i , input parameter r , threshold δ ;

Output: Mapping matrix Y ;

Initialisation: Let $W = I$, $k = 1$, and calculate S by (12.13);

(1) Let $J_0 = J(W, S)$, calculate the objective function (12.12);

(2) A similarity matrix W is obtained by solving (12.14);

(3) Each data point is mapped to r different lines, and the similarity matrix is modified $W = W_k$;

(4) Update W in (12.15), calculate the weight matrix S ;

(5) Let $J_k = J(W, S)$, calculate the objective function (12.12);

If $|J_k - J_{k-1}| < \delta$, then stop,

return W ;

Else let $k = k + 1$, go to (2);

(6) The result of the r mappings are used as y_1, y_2, \dots, y_r of matrix Y . The embedding coordinate of the point x_i is denoted by the i th row of the matrix $Y = [y_1, y_2, \dots, y_r]$.

12.3.4 Case verification

The Swiss roll data set is obtained from the sampling of the noiseless Swiss roll surface in the three-dimensional Euclidean space. In the experiment, we take the $N = 2000$, $d = 2$ Swiss roll data set, as shown in Figure 12.5. Different values of k , $\delta_s = 0.22$, and N are used, and the Laplacian eigenvalue estimation algorithm is adopted. The experimental results are shown in Figure 12.6. Compared with the traditional Laplacian eigenmap algorithm, when the k value is small, the result of our method is basically the same. As k increases, the algorithm proposed in this section becomes superior to the Laplacian eigenmap algorithm.

The Yale face database contains 165 frontal face images of 11 people. For each person, there are 15 greyscale images with different expressions, gestures, and light, and the image size is 320×112 . The image is pre-processed for alignment, and each image is cut to 40×40 pixels; the grey value of each pixel is 256. Each image represents a 1600-dimensional vector in the image space, as shown in Figure 12.7.

In the experiment, we choose Euclidean distance as the distance metric. For each person, $l (= 2, 3, 4, 5)$ images are randomly extracted as training samples, and the

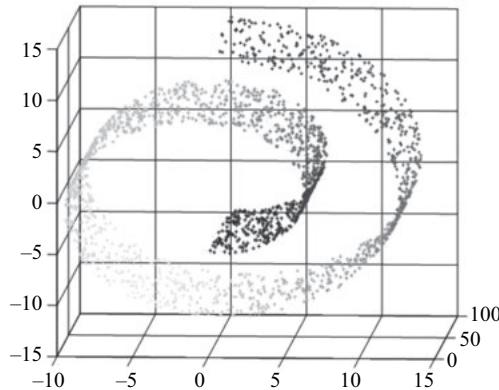


Fig. 12.5: 2000 random points sampled on the Swiss-roll data set.

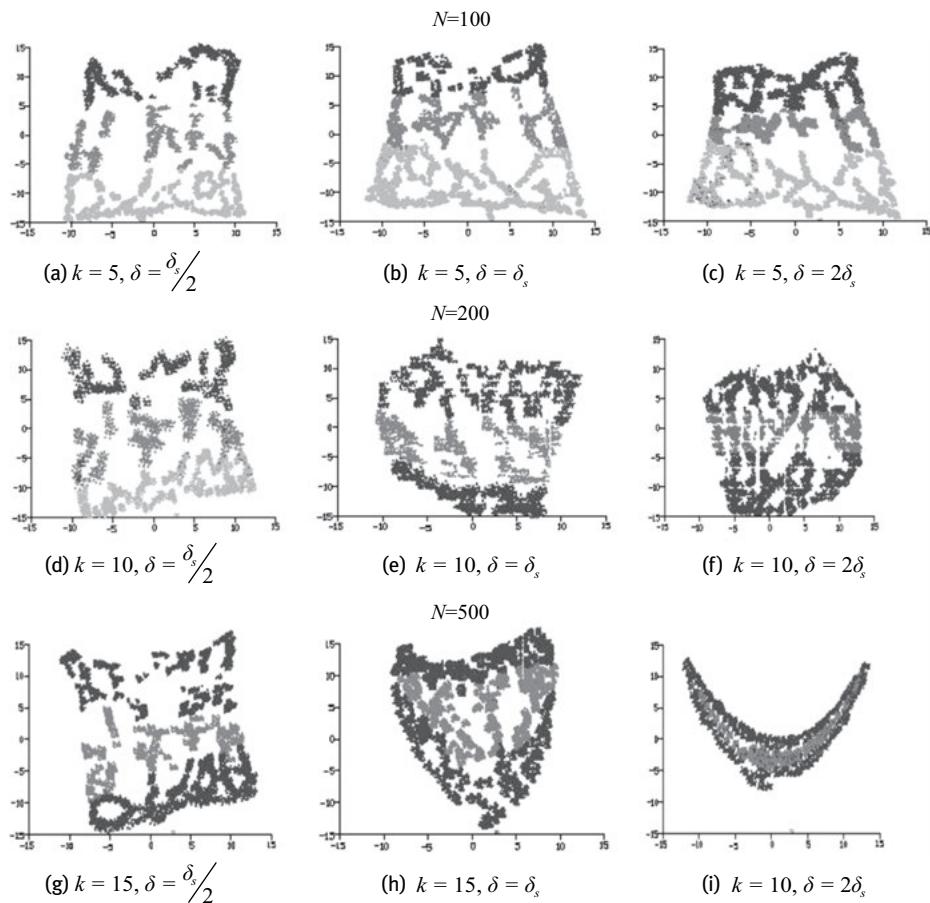


Fig. 12.6: Two dimensional representation of the Swiss-roll data set on different k , δ and N .

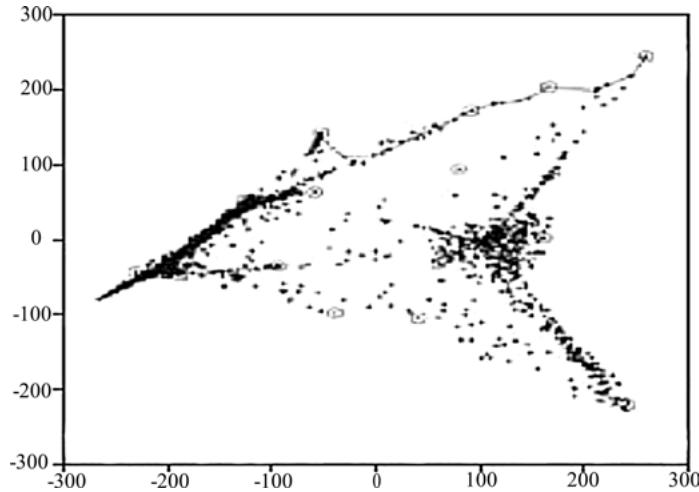


Fig. 12.7: Low dimensional embedding of face image space.

Tab. 12.1: The average recognition rate of various algorithms on the Yale database (%).

Face dataset	PCA	LLE	LP	LP on the tangent bundle
Yale	67.22	68.67	71.33	72.56

rest of the images are used as test samples. We obtained the average recognition rate of the PCA algorithm, LLE algorithm, Laplacian eigenmap algorithm, and Laplacian eigenvalue estimation algorithm on the tangent bundle, as shown in Table 12.1.

12.4 The comparison principle of image feature manifolds

Although the locality preserving learning method is superior to other methods in terms of preserving the locality of the geometric structure, local methods have significant disadvantages. (1) The existing local learning method is based on the inner dimension of the manifold to select the feature vectors. However, there is no mature method for determining the intrinsic dimensionality of the manifold (or the class number of the clustering data). (2) The locality preserving learning method usually outputs multiple feature vectors, but when reconstructing a low-dimensional manifold, a part of the useful eigenvectors is often needed, and a large part of the eigenvectors is redundant. Therefore, we can study the topological invariance of image feature manifolds by means of algebraic analysis and geometric analysis of manifolds, as well as by means of the topological structure. In this section, a topological invariant learning method based on manifold topological invariance is proposed to solve the global dimension of manifolds.

12.4.1 Topological spherical theorem

We assume the characteristic manifold of the image as follows:

- Assumption 1 (continuous mapping): For the sample points of the underlying manifold structure, adjacent points in the data space are adjacent to the manifold. If the underlying structure is unknown, this assumption limit is automatically satisfied because the manifold learning algorithm only learns the continuous mapping from the sample space to the underlying manifold.
- Assumption 2 (locally homeomorphic mapping): The neighbourhood of arbitrary sample points on the k -dimensional manifold and the local subspace in \mathbb{R}^k are homomorphic. For data sets with known underlying manifolds, if the sample points are too sparse or too noisy, then the inner structure of the manifold will not be able to learn effectively.
- Assumption 3 (topological space): The sample satisfies the Hausdorff space and second countable axiom, i.e., the sample is originally from Euclidean space. In fact, we do not need to know how Euclidean space constructs these samples; we only need to compute how the finite samples are distributed in Euclidean space.
- Assumption 4 (connectivity): Data manifolds are simply connected. If the manifold is multi-connected, then each path can be analysed separately, and manifold learning is no longer necessary to find the path [32].

We are used to analysing abstract things by concrete methods. For example, when we think about the size of an apple, we tend to consider an apple as an approximation of a sphere, see Figure 12.8.

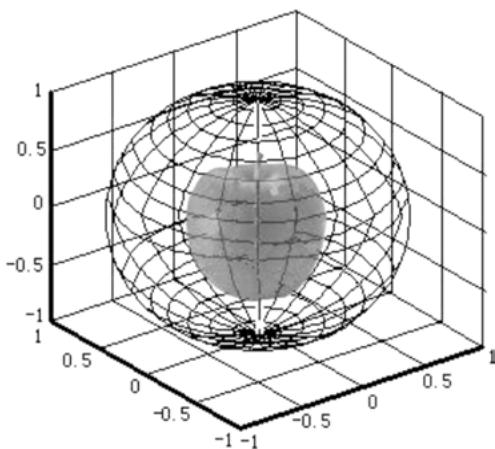


Fig. 12.8: Approximate calculation of the volume of an object.

P surface topology is a very popular two-dimensional topological manifold, e.g., the Swiss roll and S-curve. Nearly all manifold learning algorithms are based on P surfaces for experimental comparison. The S^2 spherical surface is a very important two-dimensional topological manifold.

Differential geometry indicates that it is possible to obtain topological invariance information of manifolds by adding proper geometric conditions and topological conditions to manifolds, such as curvature, diameter, compactness, connectivity, and homeomorphism [22]. Here, it is assumed that the diameter of a compact manifold is

$$d(M) = \max_{x,y \in M} \rho(x, y) = \rho(p, q)$$

where $p, q \in M$, and ρ is the distance function induced by g .

Theorem 12.2. (Bonnet–Myers theorem) [22]: Let $\gamma(t)a \leq t \leq b$ be geodesic. If there exists a conjugate point $\gamma(c)$ of $\gamma(a)$ along γ , $a < c < b$, then γ is not the shortest geodesic line connecting $\gamma(a)$ and $\gamma(b)$, i.e., the length of γ is greater than the distance between $\gamma(a)$ and $\gamma(b)$.

From the compactness of M and the Bonnet–Myers theorem, we can see that $d(M) < \pi/\sqrt{1/4} = 2\pi$. Then, from the Klingenberg theorem, we can see that $i(M) \geq \pi$. Hence, $i(M) \geq \pi > 1/2d(M)$.

For $\forall x \in M$, if $\rho(x, p) \geq i(M)$, then $\rho(x, q) < i(M)$. Let γ_1 be the shortest geodesic of p to x . Then, $L(\gamma_1) = \rho(x, p) \geq i(M)$.

Since p, q are the farthest apart ($\rho(p, q) = d(M)$ is the diameter), there is a shortest geodesic γ_2 that connects p to q , which makes the angle between γ'_1 and γ'_2 satisfy $\theta \leq \pi/2$. Then, take the shortest geodesic χ connecting x and q . As $R(P) > 1/4$ and from the compactness of M , we can get

$$R(P) \geq \frac{1}{r^2} > \frac{1}{4}$$

where $r < 2$ is a positive constant.

Consider a known sphere $S^2(r)$ with radius r in R^3 , whose Riemann sectional curvature is $1/r^2$. Geodesic triangle ABC is formed in $S^2(r)$ such that the lengths of AB and AC are $L(\gamma_1)$ and $L(\gamma_2) = d(M)$, respectively. The angle is $\angle\theta = \angle A$, as shown in Figure 12.9.

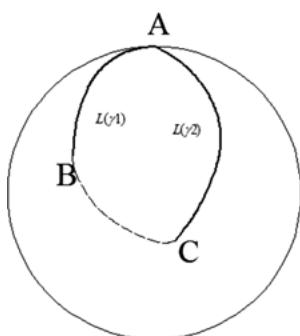


Fig. 12.9: Geodesic triangle.

Applying the Toponogov comparison theorem [22], we can obtain

$$d(x, q) = L(\chi) \leq BC < i(M) .$$

This indicates that M is the union of two open sets that are homologous to R^n . According to Brown's theorem, M is homologous to $S^n(1)$.

Theorem 12.3. (Topological spherical theorem) [22]: Let R be an n -dimensional compact single-connected Riemann manifold, and suppose that the Riemann sectional curvature $R(P)$ of any two-dimensional plane P of the tangent space satisfies

$$0 < \frac{1}{4} < R(P) \leq 1 .$$

Then, M is homologous to the n -dimensional unit sphere $S^n(1)$.

12.4.2 Polarisation theorem of image feature manifolds

As far as the feature vectors are selected to reduce the dimensionality, the detailed proof of their feasibility has been given in the study of spectral graph theory. When the eigenvectors corresponding to the smaller eigenvalues are ignored, the curvature (the correlation between the vectors and the correlation between the vectors) between the high similarities of the manifolds is hardly distorted, and the localities of the manifolds effectively maintain the geometric structure [20]. For example, consider the same person at different stages of the two face images, one of which has a scar. Then, in choosing the feature vector, we can ignore the scar and select only the eyes, nose, mouth, and other characteristics. The manifold polarisation theorem given here is the theoretical basis for choosing feature vectors.

Suppose that the nonnegative semi-definite symmetric matrix $A \in R^{D \times D}$ can be denoted by its eigenvectors $A = V\Lambda V^T$, where Λ is a diagonal matrix consisting of eigenvalues whose diagonal lines are in descending order. Define $X = \Lambda^{1/2} V^T$, and let $X_{(d)}$ be the matrix of the first d rows of X . Further, $A_{(d)} = X_{(d)}^T X_{(d)}$ is the approximate representation of A , whose rank is d .

Meanwhile, $Y_{(d)}$ is the conformal matrix of the column vectors of $X_{(d)}$, mapped to the d -dimensional hypersphere surface, and it is transformed into an identity matrix by unitary transformation. For $\forall x_i, x_j \in X_{(d)}$ (corresponding to $y_i, y_j \in Y_{(d)}$), the included angle is

$$\theta_{ij} = \angle(x_i, x_j) = \arccos \frac{x_i^T x_j}{\|x_i\| \cdot \|x_j\|} .$$

For $\forall x_i, x_j \in X_{(d)}$ (corresponding to $y_i, y_j \in Y_{(d)}$), the correlation is expressed as

$$\text{corr}(x_i, x_j) = y_i^T y_j = \cos \theta_{ij} .$$

Theorem 12.4. (Polarisation theorem) [20]: If the positive (nonnegative) symmetric matrix A can be expressed as a continuous reduced order form $A_{(D-1)}, A_{(D-2)}, \dots, A_{(d)}, \dots, A_{(2)}, A_{(1)}$, then $\sum_{i \neq j} (\cos \theta_{ij})^2$ (corresponding to $\|Y_{(d)}^T Y_{(d)}\|_F^2$) is also strictly decreasing (non-rising).

The inference obtained from Theorem 12.3 is the theoretical basis of the spectral estimation algorithm for generating the topological invariance of image feature manifolds.

Inference 12.1. (Manifold embedding principle) [20]: The similarity matrix A is replaced by the matrix $A_{(d)}$, whose rank is d . The resulting low-dimensional manifolds are similar to the inner structures of high-dimensional manifolds of data samples, and the curvature of the manifolds has almost no distortion.

The theorem and inference above are similar to those of PCA. In this algorithm, as the cosine of the angle between the vectors contains a strong energy in the similarity matrix, the data points are distributed over the hypersphere surface and precisely maintain the smaller vector angle [20]. This shows that the geometric features between adjacent points are completely preserved by vector angles.

12.4.3 Manifold dimensionality reduction algorithm

Theorem 12.1 shows that $Y_{(d)}$ is a low-dimensional nonlinear embedding of data sample manifolds over hypersphere surfaces. The arc length between two points is inversely proportional to their similarity [20]. First, the local dimension d of the low-dimensional manifold is chosen to ensure that the low-dimensional manifold can effectively maintain the geometric characteristics of the high-dimensional manifold. In the PCA algorithm, d is usually chosen by artificial estimation without prior knowledge or the influence of noise on data.

However, it is very difficult to estimate the embedding dimension of spherical low-dimensional manifolds. Therefore, the goal of this section is to embed the data samples into the vector space as accurately as possible. Suppose that $\forall p \in Y$ and $[e_1, e_2, \dots, e_{d-1}]$ is the orthogonal basis of the tangent space of point p . We set $I = E^T E$, where $E = [e_1, \dots, e_{d-1}, p]$. For $\forall e_i$, it can be regarded as a projection axis of space $R^a \times \Gamma^b$, where R^a is an a -dimensional vector space and Γ^b is a b -dimensional torus space satisfying $a + b = d - 1$. If the sample point is completely surrounding the hypersphere surface along the e_i direction, the projection axis in the e_i direction is the ring. The projection axis e_j is linear if the sample point only has an intermediate portion surrounding the hypersphere surface along the e_j direction.

Similar to the idea of the PCA algorithm, the main axis of the PCA algorithm satisfies the Gaussian distribution. The tangent space of the sample point and its projection axis E also satisfy the Gaussian distribution in the hypersphere surface [33]. When $\forall y \in CS^{d-1} \subset C^d$ is a unit vector and satisfies $y^* y = 1$, where y^* is the con-

jugate transpose matrix of y , the complex Bingham distribution can be regarded as a multivariate Gaussian distribution [33]:

$$p(y|\Sigma) = C(\Sigma)^{-1} \exp(y^* \Sigma y).$$

The Bingham distribution can be parameterised by the Hermitian matrix

$$\Sigma = E \operatorname{diag}([\kappa_1, \dots, \kappa_d]) E^*$$

where $\kappa_1 < \kappa_2 < \dots < \kappa_{d-1} < \kappa_d = 0$ is the concentration parameter value corresponding to the distribution density, and $\kappa_i \ll 0$ indicates that the data is distributed in a smaller range along the direction of e_i . The normalised constant $C(\Sigma)$ can be calculated by hypergeometric functions, and the calculation process is as follows [33]:

$$C(\Sigma) = F_1\left(\frac{1}{2}, \frac{d}{2}, \operatorname{diag}[\kappa_1, \dots, \kappa_d]\right) = 2\pi^{d-1} \sum_{j=1}^{d-1} \frac{e^{\kappa_j}}{\prod_{i \neq j} (\kappa_j - \kappa_i)}.$$

Jupp and Mardia et al. [34] demonstrated that the direction vector and concentration parameter $[\kappa_1, \dots, \kappa_d]$ in E are correlated by the eigenvalue decomposition form $YY^T = E \operatorname{diag}([\lambda_1, \dots, \lambda_d]) E^*$ of Y , where $0 < \lambda_1 < \lambda_2 < \dots < \lambda_d$ satisfies

$$\lambda_j = \frac{\partial \log C(\Sigma)}{\partial \kappa_j}.$$

For a larger sample space, there exists $Ncut$, where c is a constant. Although it is feasible to solve the concentration parameter by a numerical method, it is only necessary to obtain the eigenvalue decomposition in the manifold dimension reduction problem, and the exact concentration parameter is not needed. Its feature vector can be estimated using the tangent space maximum likelihood estimation of the sample points: for E , the sample points are nearly linearly distributed along the direction of e_i . When $\lambda_i \approx \lambda_d$, the sample points are evenly distributed along the axis of the torus.

Mardia et al. [35] excluded the possibility that the direction axis of e_i is a torus by means of normalised decomposition mapping. Therefore, the mapping matrix Y can be projected respectively along the direction of e_i and p , and the normalisation of the vector is then processed again. When the sample space is projected onto the hypersphere surface, the sample point $[0, \dots, 0, 1]$ is aligned with the Bingham axis.

12.5 Spectral estimation learning algorithm for topological invariance of image feature manifolds

The spectral estimation algorithm based on the topological invariance of image feature manifolds is based on the topological invariance theory of manifolds.

12.5.1 Spectral estimation learning algorithm for the topological invariance of image feature manifolds

Many existing manifold learning algorithms use the LLE feature vectors as features for pattern recognition, such as ISOMAP or other feature mapping methods. In this section, only the eigenvectors corresponding to the maximal two eigenvalues of the Laplacian operator in the manifold are used as features according to the spectral characteristics of the manifold, and pattern recognition and data dimension reduction can be achieved.

Algorithm 12.2: Spectral estimation learning algorithm for topological invariance of image feature manifolds

Input: Original data set $X = \{x_i\}$;

Output: Reconstructed data set $X' = \{x'_i\}$;

- (1) Determine whether the data set satisfies the four hypothetical conditions given in 12.4.1. If the local dimension of the image feature manifold is unknown, the local dimension d of the manifold can be computed by the maximum likelihood estimation algorithm.
 - (2) Run the LLE algorithm on the data sample set, and return the second feature vector $\{Y_n\}$, where n is the number of required eigenvectors.
 - (3) (Find the local extremum of each feature vector Y_i) For each data point, the corresponding neighbourhood $\{x_{ej}\}$ is obtained by the relation matrix E . If $\forall x_{ej} \in \{x_{ej}\}$ satisfies $Y_k(x_i) > Y_k(x_{ej})$ (or $Y_k(x_i) < Y_k(x_{ej})$), then x_i is a local maximum (or local minimum), where the number of local extrema represents the degree of analysis of the eigenvector.
 - (4) (Calculate local maximum and minimum curves) **Initialisation:** let $\{x_{ic}\} = \{x_i\}$. If $Y_k(x_i) \times Y_k(x_{ej}) > 0$, the adjacent point x_{ej} is added to the set $\{x_{ic}\}$ in turn. For all points in $\{x_{ic}\}$, repeat step (4) until there is no new point before adding the set. Each $\{x_{ic}\}$ is a curve corresponding to the nonlinear structure of the manifold.
-

12.5.2 Algorithm analysis

Local curves can preserve the nonlinear structure characteristics of manifolds. Because the local curve satisfies the continuous mapping in the bounded domain, there exists an extreme value in each neighbourhood. For a given vector x_i , its k nearest neighbours can be denoted as $x_{e1}, x_{e2}, \dots, x_{ek}$. Suppose that $x_{e1}, x_{e2}, \dots, x_{ek}$ satisfies

$$Y_k^{M^n}(x_{e1}) \leq Y_k^{M^n}(x_{e2}) \leq \dots \leq Y_k^{M^n}(x_{ek}).$$

If x_i is not a local extremum, the above expression can be rewritten as

$$Y_k^{M^n}(x_{e1}) \leq \dots \leq Y_k^{M^n}(x_{ej}) \leq Y_k^{M^n}(x_i) \leq Y_k^{M^n}(x_{ej+1}) \leq \dots \leq Y_k^{M^n}(x_{ek}),$$

then x_i is in its neighbourhood.

If x_i is a local extremum, suppose that x_i is a local maximum:

$$Y_k^{M^n}(x_{e1}) \leq Y_k^{M^n}(x_{e2}) \leq \dots \leq Y_k^{M^n}(x_{ek}) \leq Y_k^{M^n}(x_i)$$

then x_i is outside its neighbourhood.

If x_i is the centre of its neighbourhood, y_i should be the centre of its corresponding neighbourhood $\{y_{ej} = Y_k^{M^n}(x_{ej})\}$, so that the local reconstruction error $|y_i - \sum_{j=1}^k w_{ej} Y_k^{M^n}(x_{ej})|^2$ is minimised. Similarly, if x_i is far from the centre of its neighbourhood, then y_i should also take the local extrema away from the centre of the neighbourhood $\{y_{ej}\}$.

The curves of characteristic vectors of different scales maintain nonlinear manifold structures with different curvatures. In this paper, the local tangent space is selected to reconstruct the sample points, the local reconstruction error is minimised, and the reconstruction error is not increased in the global adjustment process. If the same scale criterion is chosen to adjust each local structure of the manifold, then the reconstruction point $x'_i = \sum_{j=1}^k w_{ej} x_{ej}$ will deviate from the sample point x_i and increase its reconstruction error. Compared with the nonlinear error, the reconstruction error is very small, because it is determined by different distribution intervals. However, our ultimate goal is to minimise the global error. Hence, we must consider the nonlinear error and the reconstruction error in equilibrium. The local curve has almost no error while it preserves the nonlinearity. Moreover, the tangent space also preserved the minimum error of the local reconstruction.

12.5.3 Example analysis

To verify that the spectral graph theory and spectral estimation algorithm can be well applied to manifold learning, we compare the algorithm in Section 12.2 with other related dimensionality reduction algorithms through experiments. The feasibility and effectiveness of the topological invariance theory algorithm are proved.

12.5.3.1 Dimension reduction of a three-dimensional cylindrical manifold

The sample points of the data manifold are sampled in a three-dimensional cylindrical manifold in the ten-dimensional data space. Through the topological invariance spectral estimation learning algorithm, the dimension of the sample points is reduced, and a two-dimensional representation is generated ($R^1 \times \Gamma^1$): one is the linear axis and the other is the axis of the torus. In the ten-dimensional Gaussian noise environment, 500 sample points along the three-dimensional cylindrical surface are generated randomly, as shown in Figure 12.10(a). The experimental results are shown in Figure 12.10(b), i.e., the two-dimensional representation of the data manifold.

12.5.3.2 Dimensionality reduction of the swiss roll data

The experiments sampled 500 random sample points in the Swiss roll data set, plus the Gaussian noise of the ten-dimensional environment, as shown in Figure 12.11(a). The sample points are distributed along the axis of the Bingham distribution, but they are not always around the equator centre. Thus, the super spherical surface reconstructs the plane coordinate system by mapping it in equal directions to the two-

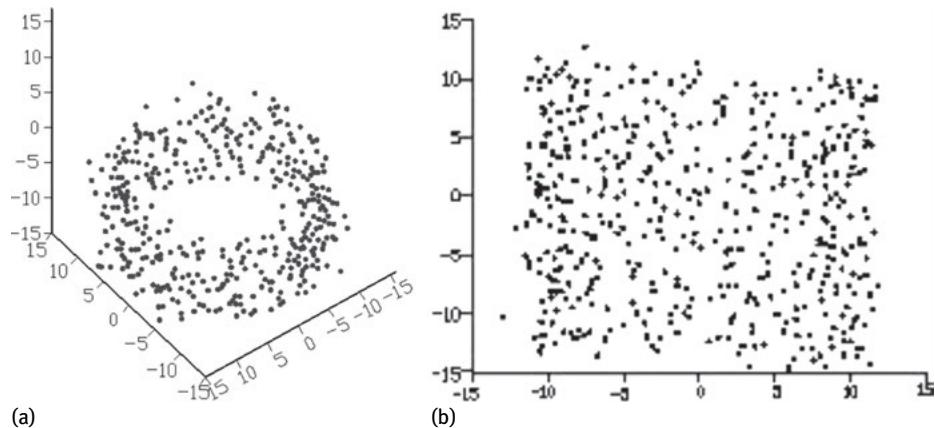


Fig. 12.10: 500 sample points along the three-dimensional cylindrical surface manifold and its two-dimensional representation.

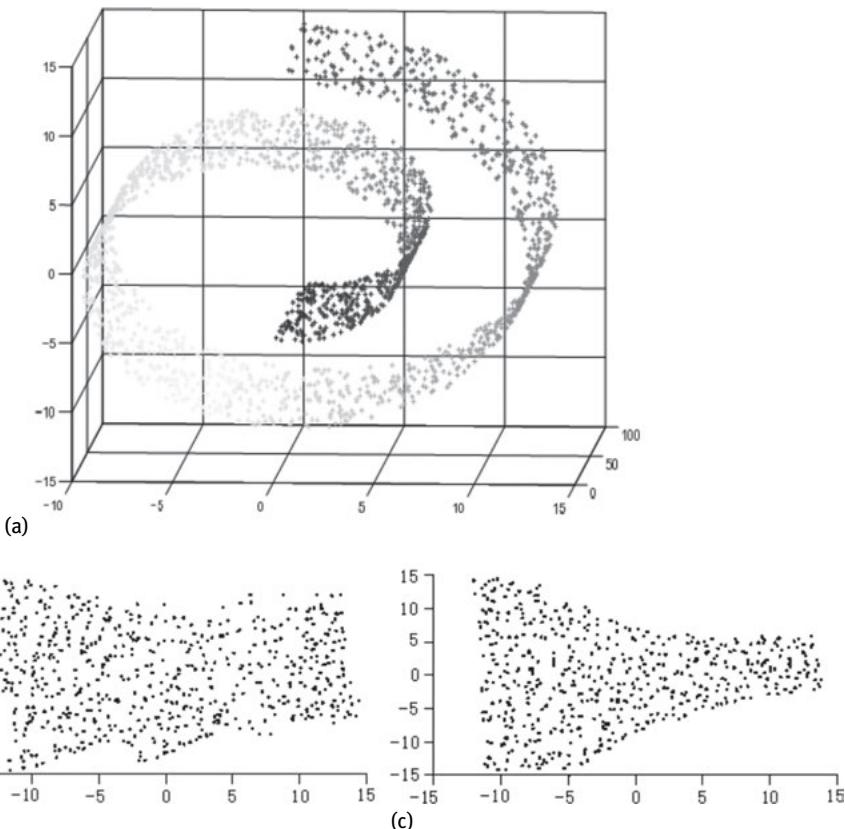


Fig. 12.11: (a) Random sampling of the 500 sample points of the Swiss-roll; (b) 2-Dimensional Dimension Reduction Results of TISE Algorithm; (c) 2-Dimensional Dimension Reduction Results of the LLE Algorithm.

dimensional space, as shown in Figure 12.11(b). Because the interval between the original data is very close, especially the sample points between the two ends of the manifold, there are some non-nearest neighbour sample points. Hence the reduced dimensionality results in a small distortion at both ends of the coordinates, as shown in Figure 12.11(c).

12.6 Clustering algorithm based on the topological invariance spectral estimation of image feature manifolds

If the dimension reduction problem is regarded as removing highly correlated (redundant) characteristics or merging these related (redundant) features, then the clustering technique can work here [20]. The clustering algorithm merges the rows of the matrix and uses a few clustering centres to represent all the data. Then, the topological invariance spectrum estimation clustering algorithm of image feature manifolds (TISC) is presented.

Algorithm 12.3: Topological invariance spectral estimation clustering of image feature manifolds (TISC)

Input: Sample set S , the corresponding graph of the sample set G , Characteristic vector number M , Cluster number K , Upper bound of standard cut set $Ncut$, parameter n ;

Output: Minimum distribution partition (A_1, A_2, \dots, A_K)

Step 1: Construct the similarity matrix of sample sets W , degree matrix D , and perform initialisation.
Given a distribution $\cup_{i=1, \dots, K} A_i = G$, satisfying $A_i \cap A_{j \neq i} = \emptyset$;

Step 2: Compute the first k eigenvalues $\lambda_1, \dots, \lambda_k$ of the matrix $L = I - D^{-1}W = U \wedge U^T$ and corresponding feature vectors x_1, \dots, x_k . The new matrix X' is obtained by regularisation of the feature vectors. Construct the matrix $X' = [x'_1, \dots, x'_k] \in R^{n \times k}$;

Step 3: Let $k = 1$, $ncut(A(k), W) = Ncut$, compute $\varepsilon(M) = ncut(A, W) - \sum_{i=1}^K \lambda_i / \lambda_{M+1} - \lambda_K$;

Step 4: If $k \leq n$ and $ncut(A(k+1), W) \leq ncut(A(k), W) + \varepsilon(M)\lambda_M$,

then $ncut(A(k), W) = Ncut(A(k), W)$, $k = k + 1$, return Step 4;

Else, return $ncut(A(k), W) = \infty$;

Step 5: Let $\min = ncut(A(k), W)$, minimum partition $\cup_{i=1, \dots, \min} A_i$ is obtained.

12.7 Summary

The algorithm of spectral estimation for topological invariance of image feature manifolds was proposed for high-dimensional data space, which is ubiquitous in real life. For visualisation in the research field of machine vision, it is often necessary to embed high-dimensional data into the low-dimensional manifold space, realise effective dimensionality reduction of feature manifolds, or perform unsupervised clustering of data sets. Because the data samples have high similarity characteristics within the sample, effectively maintaining the geometric characteristics of the internal structure

of the manifold, is the main problem that manifold learning needs to solve. Therefore, this chapter introduced the topology invariance theory and spectral theory in differential geometry into the manifold learning algorithm, and presented the spectral estimation learning algorithm for topological invariance of image feature manifolds.

Bibliography

- [1] Luo SW, Zhao LW. Manifold Learning Algorithms Based on Spectral Graph Theory. *Journal of Computer Research and Development*, 2006, 43(7), 1173–1179.
- [2] Chung FRK. *Spectral Graph Theory CBMS Series*. American Mathematical Society, 1997, 9(6), 55.
- [3] Fiedler M. Algebraic Connectivity of Graphs. *Czechoslovak Mathematical Journal*, 1973, 23(23), 298–305.
- [4] Perona P, Freeman WT. A Factorization Approach to Grouping. *European Conference on Computer Vision*. Springer Berlin Heidelberg, 1998, 655–670.
- [5] Shi J, Malik J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2000, 22(8), 888–905.
- [6] Scott GL, Longuet-Higgins HC. Feature grouping by relocalization of eigenvectors of the proximity matrix. *British Machine Vision Conference*, 1990, 103–108.
- [7] Kannan R, Vempala S, Veta A. On clusterings-good, bad and spectral. *Symposium on Foundations of Computer Science*. IEEE Computer Society, 2000, 367.
- [8] Ding CHQ, He X, Zha H, et al. A Min-max Cut Algorithm for Graph Partitioning and Data Clustering. *IEEE International Conference on Data Mining*. IEEE Computer Society, 2001, 107–114.
- [9] Ng AY, Jordan MI, Weiss Y. On spectral clustering: analysis and an algorithm. *International Conference on Neural Information Processing Systems: Natural and Synthetic*. MIT Press, 2001, 849–856.
- [10] Meila M, Shi J. Learning Segmentation by Random Walks. *Advances in Neural Information Processing Systems*, 2000, 873–879.
- [11] Cai XY, Dai GZ, Yang LB, et al. Survey on Spectral Clustering Algorithms. *Computer Science*, 2008, 35(7), 14–18.
- [12] Malik J, Belongie S, Leung T, et al. Contour and Texture Analysis for Image Segmentation. *International Journal of Computer Vision*, 2001, 43(1), 7–27.
- [13] Thierry D, Lambertus H. Visualizing Second-Order Tensor Fields with Hyperstreamlines. *IEEE Computer Graphics and Applications*, 1993, 13(4), 25–33.
- [14] Jebara T, Shchogolev V. B-Matching for spectral clustering. *European Conference on Machine Learning*. Springer-Verlag, 2006, 679–686.
- [15] Fischer B, Buhmann JM. Path-based clustering for grouping of smooth curves and texture segmentation. *Pattern Analysis & Machine Intelligence, IEEE Transactions on*, 2003, 25(4), 513–518.
- [16] Li XL. Research and Application on a data Reduction Method Based on Tensor Field. Master's Thesis, Soochow University, Suzhou, China, 2010.
- [17] Zhou D, Bousquet O, Lal TN, et al. Scholkopf: Learning with Local and Global Consistency. *Advances in Neural Information Processing Systems*, 2004, 16, 321–328.
- [18] Zelnik-Manor L, Perona P. Self-Tuning Spectral Clustering. *Advances in Neural Information Processing Systems*, 2004, 1601–1608.
- [19] Fischer I, Poland J. Amplifying the block matrix structure for spectral clustering. *Idsia*, 2005, 21–28.

- [20] Brand M, Huang K. A Unifying Theorem for Spectral Embedding and Clustering [J]. *Ai & Stats*, 2003.
- [21] Xiang T, Gong S. Spectral clustering with eigenvector selection. *Pattern Recognition*, 2008, 41(3), 1012–1029.
- [22] Xu SL, et al. Spectral Theory and Isospectrum problems, Curvature and Topological Invariants: Modern Differential Geometry. University of Science & Technology China press, 2009.
- [23] Chan PK, Schlag MDF, Zien JY. Spectral K-way ratio-cut partitioning and clustering, 1994, 13(9), 1088–1096.
- [24] Luxburg U. A tutorial on spectral clustering. Kluwer Academic Publishers, 2007.
- [25] Rathin Y, Tannenbaum A, Michailovich O. Segmenting Images on the Tensor Manifold. IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2007, 1–8.
- [26] He XF, Niyogi P. Locality preserving projections. *Advances in Neural Information Processing Systems*, 2004, 16(1), 186–197.
- [27] Zhang L, Qiao L, Chen S. Graph-optimized locality preserving projections. *Pattern Recognition*, 2010, 43(6), 1993–2002.
- [28] Qiao L, Chen S, Tan X. Sparsity preserving projections with applications to face recognition. *Pattern Recognition*, 2010, 43(1), 331–341.
- [29] Guo YF, Li SJ, Yang JY, et al. A generalized Foley-Sammon transform based on generalized Fisher discriminant criterion and its application to face recognition. *Pattern Recognition Letters*, 2003, 24(1–3): 147–158.
- [30] Wang H, Yan S, Xu D, et al. Trace Ratio vs. Ratio Trace for Dimensionality Reduction. IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, 2007, 1–8.
- [31] Jia Y, Nie F, Zhang C. Trace Ratio Problem Revisited. *IEEE Transactions on Neural Networks*, 2009, 20(4), 729–735.
- [32] Fan RKC. Spectral graph theory. Published for the conference board of the mathematical sciences by the American mathematical society, 1997.
- [33] Kent JT, Mardia KV, McDonnell P. The complex Bingham quartic distribution and shape analysis. *Journal of the Royal Statistical Society*, 2010, 68(5), 747–765.
- [34] Jupp PE, Mardia KV. Maximum Likelihood Estimators for the Matrix Von Mises-Fisher and Bingham Distributions. *Annals of Statistics*, 1979, 7(3), 599–606.
- [35] Mardia KV. Directional statistics and shape analysis. *Journal of Applied Statistics*, 1999, 26(8), 949–957.

13 Finsler geometric learning

Finsler geometry is a superset of Riemann geometry, and its measurement has a more general form, which is not subject to quadratic restriction. This chapter mainly introduces the related knowledge of Finsler geometry and the Finsler geometric learning algorithm.

13.1 Basic concept

13.1.1 Riemann manifold

Let M be an m -dimensional smooth manifold and G be the symmetric second-order covariant tensor field on M . If (U, u^i) is a local coordinate system of M , then the tensor field G on U can be expressed as

$$G = g_{ij} du^i \otimes du^j \quad (13.1)$$

where $g_{ij} = g_{ji}$ is a smooth function on U . G gives a double linear function on $T_p(M)$ at each point $p \in M$.

Suppose that

$$X = X^i \frac{\partial}{\partial u^i}, \quad Y = Y^i \frac{\partial}{\partial u^i}.$$

Let

$$G(X, Y) = g_{ij} X^i Y^j. \quad (13.2)$$

We say that tensor G is non-degenerate at point p . If there is a vector $X \in T_p(M)$ that makes $G(X, Y) = 0$ satisfy all $Y \in T_p(M)$, then it must hold that $\det(g_{ij}(p)) \neq 0$. In other words, G is non-degenerate at p if and only if the linear equations

$$g_{ij}(p) X^i = 0, \quad 1 \leq j \leq m$$

have only zero solutions, i.e., $\det(g_{ij}(p)) \neq 0$.

If any $X \in T_p(M)$ has

$$G(X, X) \geq 0 \quad (13.3)$$

and the equal sign is established only at $X = 0$, then the tensor G is positive definite at point p . It can be proven by linear algebra that G is positive definite, and the necessary and sufficient condition is that the matrix (g_{ij}) is positive definite. Therefore, the positive definite tensor G must be non-degenerate.

Definition 13.1. Suppose that there is a smooth and symmetric second-order covariant tensor field G on the m -dimensional smooth manifold M that is non-degenerate at each point. Then, M is called the generalised Riemann manifold, and G is called the fundamental tensor or metric tensor of M .

If G is positive definite, then M is called a Riemann manifold.

13.1.2 Finsler geometry

From the last section, we have seen that Riemann geometry is a metric geometry based on the positive definite two differential formula

$$G = ds^2 = g_{ij}(u)du^i \otimes du^j.$$

where u^i is a local coordinate and $g_{ij} = g_{ji}$ is a smooth function on the manifold.

In this section, we consider a more general case where the form of the metric is not the two-type restriction. Suppose that

$$ds = F(u^1, \dots, u^m; du^1, \dots, du^m) \quad (13.4)$$

where $F(x; y)$ is a non-negative smooth function with $2m$ independent variables, and it is called the Finsler function when $y = 0$. $F(x; y)$ is also required to be the first-order homogeneous function on the independent variable y , i.e.,

$$F(x^1, \dots, x^m; \lambda y^1, \dots, \lambda y^m) = |\lambda| F(x^1, \dots, x^m; y^1, \dots, y^m), \forall \lambda \in \mathbb{R}. \quad (13.5)$$

The starting point of Finsler geometry is the first concept of calculus, i.e., to calculate the arc length of a curve. Its various applications require the universality of (13.4).

Definition 13.2. Suppose that M is an m -dimensional manifold. If there is any curve

$$t \mapsto (u^1(t), \dots, u^m(t)), a \leq t \leq b$$

on it, its length s is given by the integral

$$s = \int_a^b F\left(u^1, \dots, u^m; \frac{du^1}{dt}, \dots, \frac{du^m}{dt}\right) dt \quad (13.6)$$

where F is a function of the property of (13.4), and M is called a Finsler manifold [1].

13.2 KNN algorithm based on the Finsler metric

In this section, we introduce the Finsler metric, which is more general than the Riemann metric. We replace the distance measure between sample points in the traditional K nearest neighbour (KNN) [3, 4] algorithm, and propose a KNN algorithm based on the Finsler metric [2]. Experimental results show that the classification performance of the KNN algorithm can be improved by introducing the Finsler metric.

13.2.1 K nearest neighbour algorithm

The KNN algorithm is one of the most mature algorithms in machine learning. It was originally proposed by Cover and Hart. It has the advantages of simple and intuitive

thinking, easy implementation, low classification error rate, and so on. The basic idea of the KNN algorithm is as follows. For a sample to be classified, the nearest K nearest neighbours are voted to determine their categories based on the classification attributes of the K nearest neighbours. The distance measurement method of the traditional KNN algorithm mainly uses the Euclidean distance between the sample to be measured and the training set, or the distance calculated by feature weighting.

First, the basic idea of the traditional KNN algorithm is introduced. The KNN algorithm assumes that all samples correspond to points in n -dimensional space R^n , and that the nearest neighbour of a sample is defined according to the standard Euclidean distance. The arbitrary sample \mathbf{x} represents the feature vector $\mathbf{x} = (x^1, x^2, \dots, x^m)$, where x^i represents the value of the i th feature of the sample \mathbf{x} . The distance between the two samples x_i and x_j is then defined as $d(x_i, x_j)$:

$$d(x_i, x_j) = \sqrt{\sum_{l=1}^m (x_i^l - x_j^l)^2}. \quad (13.7)$$

In nearest neighbour learning, the objective function can be a discrete problem (classification problem) or a continuous problem (regression problem). In this chapter, we discuss the case where the objective function is discrete, i.e., the classification problem can be described as $f: R^n \rightarrow V$, where $V = \{v_1, v_2, \dots, v_s\}$ corresponds to s categories. The return value $\tilde{f}(x_q)$ of the KNN algorithm is the estimate of $f(x_q)$, i.e., the most common f value in the K nearest training samples of sample x_q :

$$\tilde{f}(x_q) \leftarrow \arg \max_{v \in V} \sum_{i=1}^K \delta(v, f(x_i)) \delta(a, b) = \begin{cases} 1, & \text{if } a = b \\ 0, & \text{if } a \neq b. \end{cases} \quad (13.8)$$

As the KNN algorithm is an inert learning method, it has many shortcomings, such as low classification speed, influence of the weight of attributes and other factors on the accuracy, and strong dependence on the capacity of the sample library. Many researchers have attempted to improve the KNN algorithm. The main improvement methods are as follows:

- (1) Reducing computational complexity: Since each test sample requires its nearest K neighbours, each computation needs to traverse the entire training data set, and the computational distance needs to be sorted. To reduce the computational complexity of the algorithm, one class of methods reduced the size of the training set [5], whereas another uses fast algorithms to quickly search for the K nearest neighbours [6, 7].
- (2) Optimising the similarity measure method: The basic KNN algorithm uses the Euclidean metric, which is sensitive to noise characteristics. To overcome the defect of the feature in the KNN algorithm, different weights can be assigned to the feature in the distance formula for measuring similarity. This chapter adopts the method of optimising the similarity measurement, and uses the Finsler metric instead of the Euclidean distance to improve the KNN algorithm.

- (3) Optimising the decision strategy: The decision rules of the basic KNN algorithm have an obvious drawback. When the sample distribution density is not uniform, the sizes of the distances of the nearest K neighbours are considered instead of their distances, which may lead to miscalculations. Therefore, an improved strategy is to use the reciprocal of the distance as the weight of the sample contribution to the decision.

13.2.2 KNN algorithm based on the Finsler metric

The Finsler metric is a more general measure than the Riemann metric. Because of the generality of its definition, we can construct many Finsler metrics. Many scholars have studied Finsler geometry and given some Finsler metrics that have good properties or are easy to compute [8–15].

This section adopts a class of computable Finsler metrics, i.e., (α, β) metrics [13, 14]. They play a crucial role in Finsler geometry. Let $F = \alpha\phi(s)$, $s = \beta/\alpha$, where $\alpha = \sqrt{a_{ij}(x)y^iy^j}$ is the Riemann metric, $\beta = b_i(x)y^i$ is non-zero in the 1 form, y is the differential of x , and $\phi = \phi(s)$ is defined in the open interval $(-b_0, b_0)$ and satisfies

$$\phi(0) = 1, \phi(s) > 0, \phi(s) - s\phi'(s) + (b^2 - s^2)\phi''(s) > 0, |s| \leq b < b_0 \quad (13.9)$$

For any $x \in M$, F is the positive definite Finsler metric on manifold M . F is called the (α, β) metric. It is easy to verify that the (α, β) metric satisfies the definition of the Finsler metric as follows:

$$\begin{aligned} \text{Let: } \alpha_\lambda &= \alpha(x^1, \dots, x^m; \lambda y^1, \dots, \lambda y^m), \beta_\lambda = \beta(x^1, \dots, x^m; \lambda y^1, \dots, \lambda y^m) \\ \therefore \alpha_\lambda &= \alpha(x^1, \dots, x^m; \lambda y^1, \dots, \lambda y^m) = \sqrt{a_{ij}(x)\lambda y^i \lambda y^j} = |\lambda| \sqrt{a_{ij}(x)y^i y^j} \\ \beta_\lambda &= \beta(x^1, \dots, x^m; \lambda y^1, \dots, \lambda y^m) = b_i(x)\lambda y^i = \lambda(b_i(x)y^i) \\ \therefore F(x^1, \dots, x^m; \lambda y^1, \dots, \lambda y^m) &= \alpha_\lambda \phi(\beta_\lambda / \alpha_\lambda) = |\lambda| \sqrt{a_{ij}(x)y^i y^j} \phi\left(\frac{\lambda}{|\lambda|} \frac{\beta}{\alpha}\right). \end{aligned}$$

Hence, we just have to show that function ϕ is an even function and constantly greater than 0, i.e., $\phi(-s) = \phi(s)$ and $\phi(s) \geq 0$ can guarantee that the (α, β) metric is a Finsler metric.

This section uses the concise form of the Finsler metric. Let $F = \alpha + |\beta|$, where α is the Euclidean distance, $\alpha = \sqrt{y^i y^i}$, $\beta = t w^T y$, t is an arbitrary constant, and w is a projection vector. Note that w is the projection vector in the linear discriminant analysis (LDA) [16] algorithm. Suppose X as a data set whose classes are set as $C = \{C_1, C_2, \dots, C_s\}$. Suppose that $Z = w^T X$. Then, the intra-class distance S'_W and inter-class distance S'_B after projection can be expressed as

$$S'_W = \sum_{i=1}^s N_i (w^T m_i - w^T m) (w^T m_i - w^T m)^T. \quad (13.10)$$

$$S'_B = \sum_{i=1}^s \sum_{x^j \in C_i} (w^T x^j - w^T m_i) (w^T x^j - w^T m_i)^T \quad (13.11)$$

where $m = 1/s \sum_{i=1}^s m_i$, $N_i = \sum_{x^j \in C_i} 1$, $m_i = 1/N_i \sum_{x^j \in C_i} x^j$. The intra-class distance S_W and inter-class distance S_B after projection can be expressed as

$$S_W = \sum_{i=1}^s N_i (m_i - m) (m_i - m)^T \quad (13.12)$$

$$S_B = \sum_{i=1}^s \sum_{x^j \in C_i} (x^j - m_i) (x^j - m_i)^T. \quad (13.13)$$

Then, we have

$$S'_W = w^T S_W w \quad (13.14)$$

$$S'_B = w^T S_B w. \quad (13.15)$$

To make the intra-class distance smaller and the inter-class distance as large as possible, we can maximise the following function:

$$J(w) = \frac{w^T S_B w}{w^T S_W w}. \quad (13.16)$$

The projection vector w can be expressed as

$$w = \arg \max J(w). \quad (13.17)$$

Therefore, w can be solved by solving the following characteristic equation:

$$S_B w = \lambda S_W w. \quad (13.18)$$

The characteristic vector corresponding to the largest eigenvalue of the equation is the solution. Hence, the Finsler metric formula can be written as

$$F = \sqrt{y^i y^i} + |tw^T y|. \quad (13.19)$$

Hence, the distance between two points in the data set can be defined as

$$d_{\text{Finsler}}(x_i, x_j) = \sqrt{\sum_{l=1}^m (x_i^l - x_j^l)^2 + |tw^T (x_i - x_j)|}. \quad (13.20)$$

The basic flow of the Finsler metric KNN (FMKNN) algorithm is given below:

- (1) Set neighbour parameter K .
- (2) Find projection vector w according to (13.18).
- (3) Construct a neighbourhood graph according to the data set, where the nodes are all data samples. The distance between points can be obtained by (13.20).
- (4) For each point x_q in the test set, find the K nearest neighbours in the training set.

(5) Return

$$\tilde{f}(x_q) \leftarrow \arg \max_{c \in C} \sum_{i=1}^K \delta(c, f(x_i))$$

where $\tilde{f}(x_q)$ is the estimation of the class $f(x_q)$ of the sample x_q .

An improvement of the FMKNN algorithm is the weighted contribution to the K nearest neighbours, and the larger values are assigned to the nearest neighbours (the closer the sample distance, the higher the similarity, and the greater the contribution). It is usually weighted by the reciprocal of the distance from each nearest neighbour:

$$\tilde{f}(x_q) \leftarrow \arg \max_{c \in C} \sum_{i=1}^K w_i \delta(c, f(x_i)) , \quad w_i = \frac{1}{d_{\text{Finsler}}(x_i, x_q)} . \quad (13.21)$$

After replacing step (5) of the FMKNN algorithm with (13.21), we can reduce the influence of noise data on the performance of the algorithm, to improve the robustness of the algorithm.

13.2.3 Experimental results and analysis

In this experiment, we compare the classification performance of KNN, weighted KNN, FMKNN, and weighted FMKNN algorithms. We use MATLAB 7.1 to implement the algorithms. The platform is Windows 7, and the Wine [17] and Waveform [18] data sets from the UCI data set are employed. As the waveform data set is relatively large, 1000 samples in the waveform are randomly selected in the experiment. The experimental results are shown in Table 13.1, Table 13.2, Figure 13.1, and Figure 13.2.

From the result of the Wine data set, we can see that the classification performance of the FMKNN algorithm is better than that of the KNN and weighted KNN algorithms.

Tab. 13.1: Classification performance of each algorithm when K takes different values on the wine data set.

K	Classification Accuracy (%)			
	KNN	Weighted KNN	FMKNN	Weighted FMKNN
K=1	77.65	77.65	95.88	95.88
K=3	72.35	77.06	95.29	94.71
K=5	71.18	78.24	95.88	95.29
K=7	68.24	75.88	95.88	95.29
K=9	73.53	76.47	95.29	94.71
K=11	71.18	76.47	95.29	94.71
K=13	71.18	77.06	95.88	95.29
K=15	69.41	76.47	95.29	95.29

Tab. 13.2: Classification performance of each algorithm when K takes different values on the waveform data set.

K	Classification Accuracy (%)			
	KNN	Weighted KNN	FMKNN	Weighted FMKNN
K=1	86.6	86.6	89.9	89.9
K=3	89.1	89.1	91.2	91.4
K=5	89.5	89.5	91.8	91.8
K=7	90.1	90.1	91.3	91.4
K=9	91.1	91.1	91.5	91.7
K=11	91.5	91.5	92.0	91.8
K=13	90.8	90.8	92.2	91.6
K=15	91.3	91.2	92.2	91.8

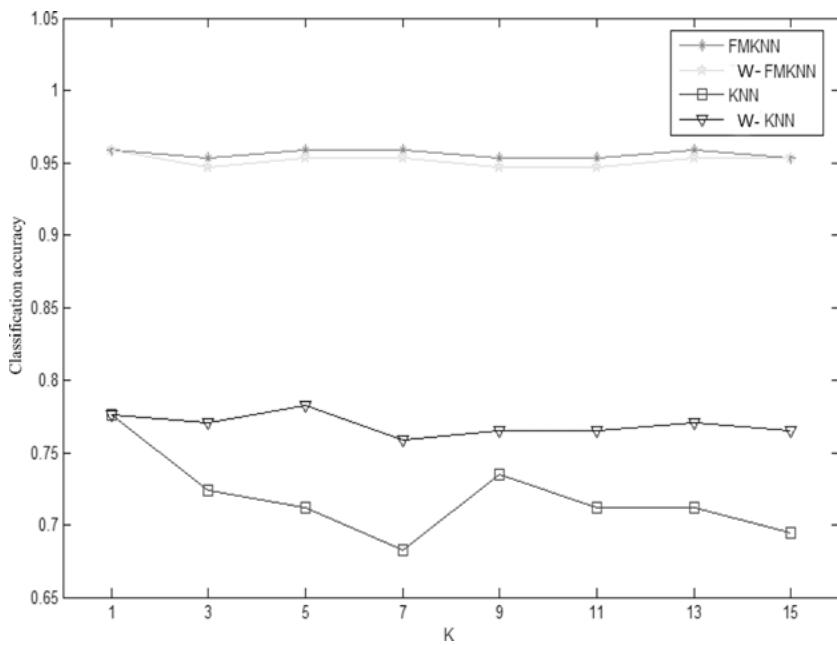


Fig. 13.1: The classification performance of the algorithms on the wine data set.

However, FMKNN does not improve the classification performance after weighting, owing to the parameter setting in the Finsler metric. In this experiment, the constant t in the FMKNN algorithm takes a large value of 1000. Therefore, in the weighted process, the weighted value will be small, and the weighted effect is not necessarily good. From the results on the Waveform data set, we can still see that the classification performance of the FMKNN algorithm is better than that of KNN and weighted KNN. However, there is no obvious relationship between FMKNN and weighted FMKNN, which is close to the experimental effect in the Wine data set.

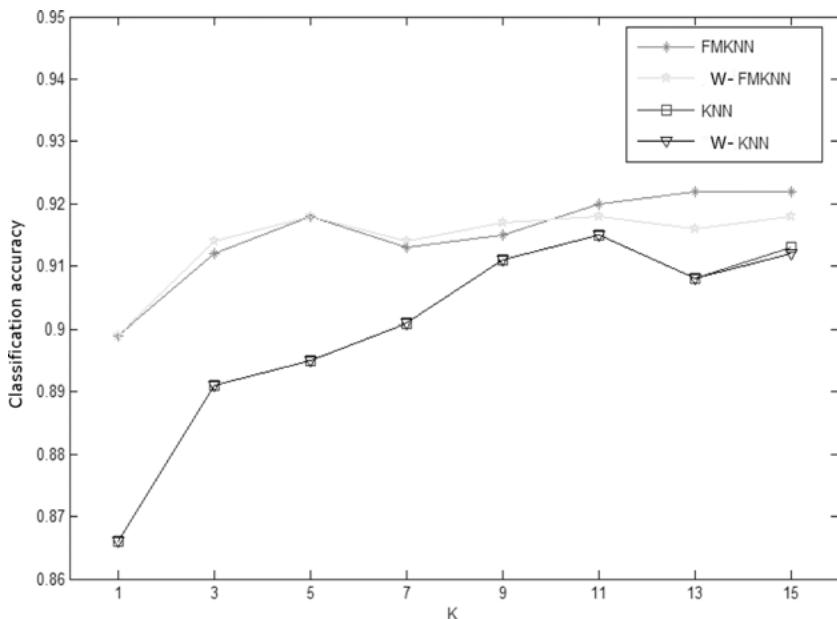


Fig. 13.2: The classification performance of the algorithms on the waveform data set.

The experimental results show that we can improve the classification performance of the algorithm after introducing the Finsler metric. FMKNN has some shortcomings. We use the projection vector of LDA when choosing the Finsler metric function. Hence, in the calculation of the Finsler metric, the FMKNN algorithm can achieve better results only on the data set with a better LDA effect. Therefore, how to select a more general Finsler metric function will be the focus of our work later.

13.3 Geometric learning algorithm based on the Finsler metric

In recent years, many researchers have improved the classical manifold learning algorithms. Supervised manifold learning algorithms have been proposed to reduce the difference within classes and to increase the difference between classes by using the label information of samples. In this section, we introduce the Finsler metric and give a geometric learning algorithm based on the Finsler metric by using the class label of samples. The algorithm is a supervised learning algorithm that can handle multi-manifold structured data. First, the supervised manifold learning algorithms are introduced briefly; then, the geometric learning algorithms based on the Finsler metric are given.

13.3.1 Supervised manifold learning algorithm

Many researchers have developed improved versions of the classical manifold learning algorithms, namely ISOMAP, LLE, and LE algorithms, and given the corresponding supervised learning algorithms, namely supervised isometric feature mapping (S-ISOMAP) [19], supervised locally linear embedding (S-LLE) [20], and supervised Laplacian eigenmap (S-LE), respectively [21]. This section gives a brief introduction to these supervised manifold learning algorithms.

13.3.1.1 S-ISOMAP

S-ISOMAP is a supervised learning algorithm for the ISOMAP algorithm. The main idea is to improve the formula of the distance between samples by using the class information of samples. The formula is as follows:

$$D(x_i, x_j) = \begin{cases} \sqrt{1 - e^{\frac{-d^2(x_i, x_j)}{\beta}}} & y_i = y_j \\ \sqrt{e^{\frac{d^2(x_i, x_j)}{\beta}} - \alpha} & y_i \neq y_j \end{cases} \quad (13.22)$$

where $d(x_i, x_j)$ represents the Euclidean distance between the samples x_i and x_j , y_i and y_j represent class labels for x_i and x_j , respectively, and α, β are undetermined parameters. α usually takes a constant value in the range $(0, 1)$, whereas β takes the mean value of the Euclidean distance between all the sample points in the data set. We can reduce the distance between the samples of the same class and increase the distance between samples of different classes by (13.22). This method can reduce noise interference. The dimension of the data is reduced according to the basic idea of ISOMAP. First, according to the KNN algorithm, the adjacency graph is constructed. Then, the Floyd shortest path algorithm is used to find the shortest path between arbitrary sample points and construct the adjacency matrix M by the shortest path. Finally, the MDS algorithm is adopted to reduce the dimension according to the M matrix.

13.3.1.2 S-LLE

S-LLE is a supervised learning algorithm for the LLE algorithm. The main idea is to modify the distance formula between the samples according to the class information of the sample, so as to change the nearest neighbour of the sample, thus affecting the final low-dimensional representation of the data set. The formula for the distance between samples is improved as follows:

$$\Delta' = \Delta + \alpha \max(\Delta) \Lambda$$
$$\Lambda_{ij} = \begin{cases} 0 & y_i = y_j \\ 1 & y_i \neq y_j \end{cases} \quad (13.23)$$

where Δ is the matrix of the Euclidean distance between the sample points, and α is a constant in the range $(0, 1)$. After the transformation of (13.23), the distance between the samples of the same class does not change, but the distance between different classes of data increases, so that more similar data can be obtained when seeking the nearest neighbours. After finding Δ' , the K nearest neighbours of LLE can be found by Δ' , and the low-dimensional representation of the data set is obtained by using the basic framework of the algorithm.

13.3.1.3 S-LE

S-LE is a supervised learning algorithm for the LE algorithm. The main idea is that the nearest neighbour sets of similar data and neighbour sets of different classes of data are first solved.

$$AS(x_i) = \frac{1}{N} \sum_{j=1}^N \exp\left(-\frac{\|x_i - x_j\|^2}{\beta}\right) \quad (13.24)$$

$$N_w(x_i) = \left\{ x_j \mid y_j = y_i, \exp\left(-\frac{\|x_i - x_j\|^2}{\beta}\right) > AS(x_i) \right\} \quad (13.25)$$

$$N_b(x_i) = \left\{ x_j \mid y_j \neq y_i, \exp\left(-\frac{\|x_i - x_j\|^2}{\beta}\right) > AS(x_i) \right\}. \quad (13.26)$$

Then, two adjacency matrices are obtained according to these two adjacent sets, and two optimisation functions are obtained as follows:

$$W_{w,ij} = \begin{cases} \exp\left(-\frac{\|x_i - x_j\|^2}{\beta}\right) & \text{if } x_j \in N_w(x_i) \text{ or } x_i \in N_w(x_j) \\ 0 & \text{otherwise} \end{cases} \quad (13.27)$$

$$W_{b,ij} = \begin{cases} 1 & \text{if } x_j \in N_b(x_i) \text{ or } x_i \in N_b(x_j) \\ 0 & \text{otherwise.} \end{cases} \quad (13.28)$$

Let x'_i be a low-dimensional representation of x_i . Then, for each sample point, there is an objective function

$$\min \frac{1}{2} \sum_{i,j} \|x'_i - x'_j\|^2 W_{w,ij} \quad (13.29)$$

$$\max \frac{1}{2} \sum_{i,j} \|x'_i - x'_j\|^2 W_{b,ij}. \quad (13.30)$$

Let $Z = [x'_1 \quad \dots \quad x'_N]^T$. Then, the objective functions (13.29), (13.30) can be rewritten as

$$\min \text{tr}(Z^T L_w Z) \quad (13.31)$$

$$\max \text{tr}(Z^T L_b Z) \quad (13.32)$$

where $L_w = D_w - W_w$, $L_b = D_b - W_b$. By using constraint $Z^T D_w Z = I$, the objective functions (13.31) and (13.32) can be combined into the objective function

$$\arg \max_Z \{y \operatorname{tr}(Z^T L_b Z) + (1-y) \operatorname{tr}(Z^T W_w Z)\} \quad \text{s.t.} \quad Z^T D_w Z = I \quad (13.33)$$

where $y \in [0, 1]$ is a constant. Let $B = yL_b + (1-y)W_w$. The objective function (13.33) can be written as

$$\arg \max_Z \operatorname{tr}(Z^T B Z) \quad \text{s.t.} \quad Z^T D_w Z = I. \quad (13.34)$$

The solving process of (13.34) is the following process of solving the characteristic equation:

$$Bz = \lambda D_w z. \quad (13.35)$$

The feature vector corresponding to the maximum d eigenvalue of the characteristic equation (13.35) is z_1, z_2, \dots, z_d , and $Z = [z_1, z_2, \dots, z_d]$ is the low-dimensional embedding of the data set.

13.3.2 Finsler geometric learning algorithm

Assuming that the data set is classified, the class information of the training data is known. First, the data set is divided according to the classification information of the training data. Suppose that the data set is S . A partition of the data set is S_1, S_2, \dots, S_n . Assuming that each partition S_i , $1 \leq i \leq n$ is a single manifold structure, the existing classical manifold learning algorithms can be used to reduce dimensionality. After we find the low-dimensional representation of the partition set, we need to give the method of partitioning set merging. The centre point of S_i is p_i , and the Finsler distance between any two centres is calculated to form the distance matrix M , $M_{ij} = d_{\text{Finsler}}(p_i, p_j)$. Then, the low-dimensional representation of the centre point is obtained by applying the MDS algorithm to the distance matrix. Finally, the low-dimensional representation of each partition set S_i and the low-dimensional representation of the centre points are rotated and translated, and the geometric structure relation between the partitioned data sets is maintained, so that the low-dimensional representation of the whole data set is obtained.

It is easy to partition the training data set because the training data set has its own class information. As long as the data of the same class is classified into the data of the same set, it can achieve the purpose of division. However, for the test data set, we do not have class information, and the division of the test data requires a certain operation. This paper adopts the data set partitioning method based on centre points to classify the test data, which first computes the training data centre of the data, and then divides the testing samples into a central point nearest the corresponding category.

For the test data point p , it belongs to the set C_p . Then, $C_p = S_i$, $i = \arg \min d(p, p_i)$. Here, d is the Euclidean distance.

A centre-based data set partition method is given as follows. See Algorithm 13.1.

Algorithm 13.1: Centre-based data set partition

Input: Training data set $X_{\text{tr}} = \{x_{\text{tr}1}, x_{\text{tr}2}, \dots, x_{\text{tr}N} \in R^D\}$, training data set label $Y_{\text{tr}} = \{y_{\text{tr}1}, y_{\text{tr}2}, \dots, y_{\text{tr}N} \in R\}$, testing data set $X_{\text{tst}} = \{x_{\text{tst}1}, x_{\text{tst}2}, \dots, x_{\text{tst}M} \in R^D\}$

Output: Partition of the entire data set S_1, S_2, \dots, S_n , n is the number of data classes
 (1) Find the total number of categories of the data set

$$T = \text{distinct}\{Y_{\text{tr}}\}$$

$$n = |T|$$

- (2) Initialise $S_i = \emptyset$, $1 \leq i \leq n$. Suppose that $T = \{T_1, T_2, \dots, T_n\}$.
 (3) Partition the training data set

$$S_i \leftarrow S_i \cup x_{\text{tr}j}, y_{\text{tr}j} = T_i, 1 \leq j \leq N$$

- (4) Find the data centre points of each category in the training data set

$$m_i = \frac{\sum_{S_{ij} \in S_i} S_{ij}}{|S_i|}$$

- (5) Partition the test data set

$$S_i \leftarrow S_i \cup x_{\text{tst}j}, i = \arg \min_{1 \leq i \leq n} d(p_i, x_{\text{tst}j}), 1 \leq j \leq M$$

- (6) Output partition set S_1, S_2, \dots, S_n

From Algorithm 13.1, we can see that the algorithm mainly focuses on the training data set and then divides the test data set into the nearest one.

Algorithm 13.1 can be regarded as a simple classification process that divides the whole data set and prepares for the data processing afterwards.

For already partitioned data sets, we assume that a single set S_i , $1 \leq i \leq n$ has a single manifold structure. From the above hypothesis, we can use the classical algorithms to reduce the dimension. In this paper, we adopt the LLE algorithm to reduce the dimension. For the entire data set, we will use the LLE algorithm to reduce the dimension of the set S_1, S_2, \dots, S_n partitioned by Algorithm 13.1. Then, the adjacency graph is constructed according to the data set, and the distance is measured by the Finsler metric.

Algorithm 13.2: Dimensionality reduction algorithm for data centers

Input: Sample dimension d , Neighborhood parameter K , training data set $X_{\text{tr}} = \{x_{\text{tr}1}, x_{\text{tr}2}, \dots, x_{\text{tr}N} \in R^D\}$, training data set label $Y_{\text{tr}} = \{y_{\text{tr}1}, y_{\text{tr}2}, \dots, y_{\text{tr}N} \in R\}$, testing data set $X_{\text{tst}} = \{x_{\text{tst}1}, x_{\text{tst}2}, \dots, x_{\text{tst}M} \in R^D\}$

Output: Low dimensional embedding of the central points

- (1) Construct a dataset nearest neighbor graph M

$$M_{ij} = \begin{cases} d(x_i, x_j) & x_i, x_j \in X_{\text{tr}}, y_i = y_j \\ d_{\text{Finsler}}(x_i, x_j) & \text{else} \end{cases}$$

$d_{\text{Finsler}}(x_i, x_j)$ is calculated by the formula (7.14) in Chapter 7.

- (2) According to the M computed in the first step, the K nearest neighbor of each data point is computed, and then the distances of the non K nearest neighbors are all infinite.
- (3) Find the point of dimensionality reduction for each set p_i

$$p_i = x_j$$

$$j = \arg \min_j d(x_j, m_i), x_j \in S_i$$

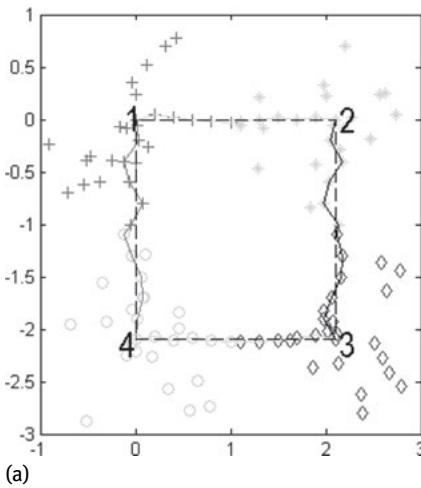
m_i is the geometric center point of the data set S_i calculated by algorithm 13.1.

- (4) Dijkstra shortest path algorithm is used to find the shortest path between the central points, and the adjacency graph M' is constructed, $M'_{ij} = s(p_i, p_j)$, $s(p, q)$ denotes the shortest path between point p and point q .
- (5) According to M' and d , the low dimensional embedding of each central point is obtained by the MDS algorithm, $P = \{p'_1, p'_2, \dots, p'_n\}$
-

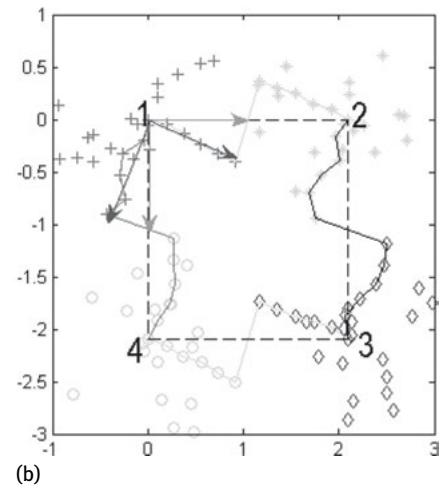
For the central point of each set, find the shortest path between them [22]. Finally, the central point is reduced by using the MDS algorithm, and the low-dimensional embedding of the centre point is obtained. Algorithm 13.2 for embedding the centre point is given below.

After obtaining the low-dimensional embedding of each partition set and the low-dimensional representation of the central point, each partition set must be merged together. The method of merging is given below. First, we can translate each set according to the low-dimensional representation of each central point, so that the distance between the central points of each set is the same. There is no guarantee that the existing connections between the data can be determined only after translation, because the results of the LLE algorithm can be represented by a number of rotations. In other words, if Y is the low-dimensional embedding representation of X through the LLE algorithm, then for any rotation transformation matrix, it satisfies the optimisation condition in the LLE algorithm. Therefore, we need to affine-transform the low-dimensional representation of the set to keep the initial structure information. As shown in Figure 13.3, $+$, $*$, \square , and \diamond represent the low-dimensional embedding of data for different partitioning sets, respectively. As we can see from Figure 13.3, if we simply translate the low-dimensional representations of different data sets, the structural information between different data sets may be lost. Hence, we have to rotate the partitioning of the low-dimensional embedding of the set so that it retains the original structural information.

In Figure 13.3, the solid line represents the shortest path between the central points of the collection. It is not difficult to see from the figure that if the point on the shortest path is linearly fitted, the line (dotted line) between the central points of the set can be approximately represented as a linear fitting of the shortest path. Hence, when rotating a partition set, we can determine the rotation range of the divided set according to the linear fitting of the shortest path and the line segment between the central points. For partitioning set S_i , assuming that S_i is adjacent to S_j , the shortest path between S'_i and S'_j is assumed to be $p_{ij0}p_{ij1}\dots p_{ijk}$, and it can be represented as



(a)



(b)

Fig. 13.3: (a) A low dimensional representation of the partition dataset without rotation, (b) a low dimensional representation of the partition dataset after rotation.

r_{ij} . Let $L(r_{ij})$ denote the number of edges in r_{ij} . Then, $r_{ij}(0) = p_{ij0}$, $r_{ij}(L(r_{ij})) = p_{ijk}$. Let T_{ij} be the shortest path between S'_i and S'_j , where the points belong to S'_i . Then, $T_{ij} = \{r_{ij}(0), \dots, r_{ij}(k) | r_{ij}(k) \in S'_i, r_{ij}(k+1) \in S'_j\}$. Linear fitting of all elements of T_{ij} is performed, the unit vector of the line is denoted as t_{ij} , and the unit vector of p'_i pointing to p'_j is e_{ij} . Suppose that the partition set adjacent to S_i is $S_{i1}, S_{i2}, \dots, S_{im}$. Then, $T_i = [t_{i(i1)} \ t_{i(i2)} \ \dots \ t_{i(ik)}]$, $E_i = [e_{i(i1)} \ e_{i(i2)} \ \dots \ e_{i(ik)}]$. If E_i can be obtained by a linear transformation of T_i , then

$$E_i = A_i \cdot T_i . \quad (13.36)$$

The solution of (13.36) can be obtained by solving the pseudo-inverse method [23]:

$$A_i = E_i (T_i^T T_i)^{-1} T_i^T . \quad (13.37)$$

To ensure that the data set is only rotated, we need to guarantee that $A_i \cdot A_i^T = I$. Therefore, we can obtain the transformation matrix A_i by solving (13.37):

$$E_i = A_i \cdot T_i \quad \text{s.t.} \quad A_i \cdot A_i^T = I . \quad (13.38)$$

Further, (13.38) can be solved by the inverse problem of the orthogonal matrix [24], and the solution is as follows:

Suppose that $T_i = P \begin{bmatrix} T & 0 \\ 0 & 0 \end{bmatrix} Q^T$, where P is a d -order orthogonal matrix, Q is a k -order orthogonal array, and T is an r -order non-singular matrix. Let $P = [P_1 \ P_2]$, $P_1 \in R_r^{d \times r}$, $Q = [Q_1 \ Q_2]$, $Q_1 \in R_r^{k \times r}$, and take the above formula in the form (13.38).

Then, we can get

$$A_i \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} T & 0 \\ 0 & 0 \end{bmatrix} = E_i Q$$

$$A_i P_1 T = E_i Q$$

$$A_i \begin{bmatrix} P_1 T & 0 \end{bmatrix} = E_i \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$$

$$A_i P_1 T = E_i Q_1$$

$$A_i P_1 = E_i Q_1 T^{-1}.$$

Let U, V be the orthogonal row full rank matrix that makes $\begin{bmatrix} E_i Q_1 T^{-1} & U \end{bmatrix}$ an orthogonal matrix. $\begin{bmatrix} P_1 & V \end{bmatrix}$ is an orthogonal matrix that satisfies $A_i \begin{bmatrix} P_1 & V \end{bmatrix} = \begin{bmatrix} E_i Q_1 T^{-1} & U \end{bmatrix}$, and $A_i = \begin{bmatrix} E_i Q_1 T^{-1} & U \end{bmatrix} \begin{bmatrix} P_1 & V \end{bmatrix}^T$.

Suppose that u_1, \dots, u_{n-r} is an orthonormal basis of $R(E_i Q_1 T^{-1})^\perp$, $U^* = [u_1 \dots u_{n-r}]$, v_1, \dots, v_{n-r} is a orthonormal basis of $R(P_1)^\perp$, $V^* = [v_1 \dots v_{n-r}]$.

Take A_i^* as the solution of (13.38). The singular value decomposition of $V^{*T} A_i^{*T} U^*$ is $V^{*T} A_i^{*T} U^* = JD_1 K$, and $A_i = E_i Q_1 T^{-1} P_1^T + U^* K^T J^T V^{*T}$ is the solution of (13.37).

Next, we give the dimensionality reduction algorithm of the whole data set, i.e., the geometric learning algorithm based on the Finsler metric, referred to as the Finsler geometric learning algorithm (FGLA).

From Algorithm 13.3, we can see that the Finsler geometric learning algorithm can effectively enlarge the distance between the data sets by dividing the data sets and finding the Finsler distance between the central points of the data sets. The local geometric structure of the data set can be well maintained through the linear transformation of the data set.

Algorithm 13.3: Finsler geometric learning algorithm

Input: Sample dimension d , Neighbourhood parameter K , training data set $X_{\text{tr}} = \{x_{\text{tr}1}, x_{\text{tr}2}, \dots, x_{\text{tr}N} \in R^d\}$, training data set label $Y_{\text{tr}} = \{y_{\text{tr}1}, y_{\text{tr}2}, \dots, y_{\text{tr}N} \in R\}$, testing data set $X_{\text{tst}} = \{x_{\text{tst}1}, x_{\text{tst}2}, \dots, x_{\text{tst}M} \in R^d\}$

Output: Low-dimensional representation of test data set and training data set X'

- (1) Algorithm 13.1 is used to divide the data set into a partition set S_1, S_2, \dots, S_n .
- (2) For each partition set S_i , the LLE algorithm is used to obtain its low-dimensional embedding S'_i .
- (3) Reduce the dimension of the central point of the partition set $\{S_1, S_2, \dots, S_n\}$ through Algorithm 13.2, and the low-dimensional embedding of the centre point is obtained. Save the shortest path between the central points in the fourth step of Algorithm 13.2.
- (4) According to the low-dimensional embedding of the central point and the path between the central points in the third step, E_i and T_i are obtained for each partition set S'_i , and then (13.38) is solved. The transformation matrix A_i is found. Then, S'_i can rotate around its centre to obtain the matrix. The index of the S'_i central point is k_i , and the number of elements is n_i . Then,

$$S'_i \leftarrow A_i \left(S'_i - \overbrace{\begin{bmatrix} S'_i(k_i) & \dots & S'_i(k_i) \end{bmatrix}}^{n_i} \right)$$

- (5) According to low-dimensional embedding $\{p'_1, p'_2, \dots, p'_n\}$ of the central point, translate the subset $S'_i \leftarrow S'_i + \begin{bmatrix} \overbrace{p'_i \dots p'_i}^{n_j} \end{bmatrix}$
- (6) Output the low-dimensional representation of the entire set $X' = \bigcup_{1 \leq i \leq n} S'_i$
-

The Finsler geometric learning algorithm is divided into four main parts:

- (1) Partition of the data set: the time complexity is $O(n(N + M))$.
- (2) Suppose that the data set is evenly distributed after division: $|S_1| \approx |S_2| \approx \dots \approx |S_n|$. The time complexity of reducing the dimension of the divided data set is

$$O\left(\frac{(N + M)^2}{n} \lg \frac{N + M}{n}\right).$$

- (3) The time complexity of solving the low-dimensional embedding of the central points is $O(n(N + M)^2)$.
- (4) When solving the rotation transformation matrix of each data set, we mainly use SVD and matrix multiplication. The time complexity is $O(\min\{nd^2, n^2d\} + d^3)$. Thus, the time complexity of the algorithm is

$$O\left(\max\left(\frac{(N + M)^2}{n} \lg(N + M), n(N + M)^2\right)\right).$$

13.4 Summary

Finsler geometry is a more general differential geometry than Riemann geometry. Its main feature is to adopt a more general metric than the Riemann metric. In this chapter, Finsler geometry was introduced into the manifold learning algorithm. First, the Finsler metric was combined with the KNN algorithm, and the KNN algorithm based on the Finsler metric was given. The validity of the Finsler metric was proven experimentally. Then, the Finsler metric was introduced into the data set with a manifold structure, and a supervised learning algorithm called the Finsler geometric learning algorithm was proposed.

Bibliography

- [1] Chen XS, Chen WH. Lectures on Differential Geometry. Peking University press, 2001.
- [2] Chen M, He SP, Li FZ. Research on Finsler Metric in KNN Algorithm. Journal of Frontiers of Computer Science & Technology, 2011, 5(11), 1021–1026.
- [3] Cover T, Hart P. Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 1967, 13(1), 21–27.
- [4] Michell TM. Machine Learning. Machinery Industry Press, 2004.

- [5] Hart PE. The condensed nearest neighbor rule. *IEEE Trans on Information Theory*, 1968, 14(3), 515–516.
- [6] Hwang WJ. Fast kNN classification algorithm based on partial distance search. *Electronics Letters*, 1998, 34(21), 2062–2063.
- [7] Pan JS, Qiao YL, Sun SH. A fast K nearest neighbors classification algorithm. *IEICE Trans. Fundamentals*, 2004, E87-A(4), 961–963.
- [8] Solange FR, Portugal R. Finsler. A computer algebra package for Finsler geometries. *Nonlinear Analysis: Theory, Methods & Applications*, 2001, 47(9), 6121–6134.
- [9] Chern SS. Finsler geometry is just Riemannian geometry without the quadratic restriction. *Notices of the American Mathematical Society*, 1996, 43(9), 959–963.
- [10] Nie Z. *The Laplace Operator and Its Properties in Finsler Space*. College Mathematics, 2004, 4(20), 52–58.
- [11] Nie Z. *On Distance Function and Geodesic Sphere of Finsler Manifolds*. College Mathematics, 2000, 6(16), 8–12.
- [12] Huang LB. Navigation problems of Finsler manifolds and their applications. Doctoral dissertation, Peking University, Beijing, China, 2008.
- [13] Li BL. On some important Finsler measures. Doctoral dissertation, Zhejiang University, Zhejiang, China, 2007.
- [14] Park HS, Lee IY. On projectively flat Finsler spaces with \$(\alpha, \beta)\$-metric. *Communications of the Korean Mathematical Society*, 1999, 14(2), 373–383.
- [15] Yang CH, Mo XH, Shen ZM. Construction of some projective flat Finsler measures. *Science in China(Series A)*, 2006, 36(2), 121–133.
- [16] Duda RO, Hart PE, Stork DG. *Pattern Classification* (2nd Edition). John Wiley & Sons, 2000.
- [17] <http://archive.ics.uci.edu/ml/datasets/Wine/>.
- [18] <http://archive.ics.uci.edu/ml/datasets/Waveform+Database+Generator+%28Version+2%29/>.
- [19] Geng X, Zhan DC, Zhou ZH. Supervised nonlinear dimensionality reduction for visualization and classification. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics: a publication of the IEEE Systems, Man, and Cybernetics Society*, 2005, 35(6), 1098.
- [20] Zhao L, Zhang Z. *Supervised locally linear embedding with probability-based distance for classification*. Pergamon Press, Inc. 2009.
- [21] Raducanu B, Dornaika F. A supervised non-linear dimensionality reduction approach for manifold learning. *Pattern Recognition*, 2012, 45(6), 2432–2444.
- [22] Sun HQ. *Graph Theory and Its Applications*. Science Press, 2004.
- [23] Hastie T, Tibshirani R, Friedman J. *The Elements of Statistical Learning*. Springer Series in Statistics, 2001.
- [24] Li HZ. Inverse problem of orthogonal matrix and its optimal approximation. *Journal of Shanghai Maritime University*, 1990, 4, 71–76.

14 Homology boundary learning

14.1 Boundary learning algorithm

Boundary division can be divided into data partition and image division. At present, the main boundary learning algorithms are the TVQ algorithm, [1], Canny algorithm, [2], and the RLMC algorithm, [3].

14.1.1 Tangent vector quantisation (TVQ) algorithm

For this algorithm, Fabio Aiolli and Alessandro Sperduti proposed a good scheme to improve the boundary problem, which is based on the boundary theory. The algorithm is based on the sample weighting strategy. The basic idea is to increase the new copy of the training set, and the sample is not classified under the sufficient boundary.

Fabio Aiolli and Alessandro Sperduti proposed this algorithm as an example of learning, i.e., a pre-scheduled instance. It includes the distance cut based on the 1-NN classifier. The 1-NN classifier implements quantisation of a cut distance prototype. The generalisation ability of the cut distance model created in this way is obviously improved. This generalisation ability comes from the standard cut model. Furthermore, the obtained model has advantages over other algorithms, such as the SVM algorithm, in OCR tasks.

Algorithm 14.1:

Input: T : Iteration number; Q : Number of patterns per class; θ : Marginal critical value

Output: W

(1) Initialisation: $W(1) \leftarrow N$, $\gamma_i \leftarrow 1/N$; for any y, q , In random mode,

$$M_y^{(q)} \leftarrow (c_y^{(q)}, T_y^{(q)}) .$$

for $t = 1, \dots, T$

$$\forall y, \forall q, \Delta M_y^{(q)} = 0 , \quad \Delta M_{y_i}^{(q_p)} \leftarrow \Delta M_{y_i}^{(q_p)} + r_i(t) \frac{\delta \mu_i}{\delta M_{y_i}^{(q_p)}} , \quad \Delta M_{y_i}^{(q_n)} \leftarrow \gamma_i(t) \frac{\delta \mu_i}{\delta M_{y_i}^{(q_n)}} .$$

(2) Regularisation r_i^s .

Compared with the SVM method, this method works directly on the nonlinear model instead of taking the kernel. When both methods are based on the concept of statistical learning theory, they converge uniformly from empirical risk to ideal risk.

This approach is very similar to the boosting algorithm. However, the boosting algorithm places greater emphasis on the independent hypothesis. Hence, multiplication strategies are better suited for switches.

14.1.2 Regularised large marginal classifier (RLMC)

By adding a rule item (in the class dispersion) to the edge critical value of the classical large margin classifier, the mapping feature vector can reach the maximum value when the class scatter reaches the minimum value [3].

Algorithm 14.2:

Input: The data set contains two categories of samples $\{x_i, y_i\}$, $x_i \in R^n$, $y_i \in \{+1, -1\}$ represents class markup

Output: W

- (1) According to the Fisher criterion, the existing LMC objective function is modified. By adding rule items, a new objective function is generated by group scattering.
- (2) By adding rule items, $\min J(w) = 1/2 \|w\|^2 = 1/2 w^T w$

$$\text{s.t. } \sum_{i=1}^N \alpha_i y_i = 0 \quad \alpha_i \geq 0, \quad i = 1, \dots, N$$

$$\min J(w) = 1/2 (\|w\|^2 + \eta w^T S_w w)$$

$$\text{s.t. } y_i (w^T x_i + b \geq 1) \quad i = 1, \dots, N$$

- (3) The upper equation maps the two classes of training samples to direction W^* to get a new model. By adding rule items, the upper equation can reach the maximum edge when the scatter of the class is minimal.

$$S_w = \sum_{i=1}^2 \sum_{j=1}^{N_i} (x_i - m_i)(x_j - m_i)^T, \quad m_i = \frac{1}{N_i} \sum_{j=1}^{N_i} x_j, \quad i = 1, 2$$

- (4) Calculate all classes of scatter matrices and average vectors within the class of training samples, especially when $\eta \geq 0$, where η is a weight that controls the edge of the balance and is discrete within the group.
-

The RLMC method can achieve good results in face recognition under time and illumination changes. Through the existing classifier, the training sample is mapped to the vector that is acquired.

14.1.3 Boundary detection algorithm based on the boundary Markov random field and Boltzmann machine

In general, edge detection is based on the discontinuity of the image grey scale, including surface fitting and boundary tracking. However, many edge detection algorithms are only concerned with grey information, and they are sensitive to noise. Tan [4] transformed the edge detection problem into a global optimisation problem, considering not only the grey change information but also the local edge structure. The algorithm initialises edge detection by the edge detection operator, and the model parameters are estimated by the Boltzmann machine. Finally, the edge relaxation Boltzmann machine search algorithm is used.

Algorithm 14.3:

Input: The data set contains two categories of samples $\{x_i, y_i\}$, $x_i \in R^n$, $y_i \in \{+1, -1\}$ represents class markup

Output: W

- (1) Initialises edge detection by the edge detection operator:

$e = \{e_{i,j} | i, j \in L\}$ represents the response intensity obtained by the edge detection operator based on the grey image Z . Based on the differential algorithm, edge detection depends only on the grey information.

$$S_{i,j} = \begin{cases} 1, & \text{if } p(z_{i,j} | x_{i,j} = 1) \geq p(z_{i,j} | x_{i,j} = 0) \\ 0, & \text{if } p(z_{i,j} | x_{i,j} = 1) < p(z_{i,j} | x_{i,j} = 0) \end{cases}.$$

- (2) Estimate the model parameters by the Boltzmann machine:

With the results of the initial edge detection as learning samples, according to the Boltzmann machine learning rule, estimate parameters α_k such that $P(s)$ is maximum. The problem of parameter estimation is transformed into a global optimisation problem.

- (3) Search algorithm of edge relaxation based on a Boltzmann machine:

Owing to the adoption of a 24-neighbourhood system, when the edge fault distance is greater than a 5*5 neighbourhood, the search algorithm is difficult to deal with; a larger neighbourhood system can be adopted, but it will increase the algorithm complexity.

The limitation of the algorithm is that the learning samples of the learning algorithm are initial edge detection results. Although they have some fault tolerance, it is difficult to carry out the algorithm when there is excessive noise. The learning algorithm can obtain the main local edge features, but it has some limitations. The image effect is better only with some regularity on the edges (i.e., when some local edge features are more obvious), and it is more suitable for texture image edge detection. The improved scheme uses standard learning samples for parameter estimation and is applicable to non-learning samples, but the learning samples and non-learning samples must be of the same type of image.

14.1.4 Fuzzy edge detection algorithm based on the qualification function

The procedure of the traditional image edge detection fuzzy algorithm, referred to as the Pal-King algorithm [5], is as follows: (1) The membership function is used to map the image $f(x, y)$ into fuzzy membership matrix $G(i, j)$. (2) Multiple nonlinear transformations T_r are performed on degree matrix $G(i, j)$ to enhance the edge information and weaken the non-edge information. (3) Perform reverse changes to the fuzzy membership matrix $G(i, j)$ to obtain enhanced images. (4) Extract the edge with ‘Min’ or ‘Max’.

However, this algorithm has many shortcomings. Therefore, improvements are required. We introduce the qualification function on the basis of the original image, com-

plete the transformation from image to membership matrix, and enhance the transformation of edge information. This greatly reduces the amount of computation and dynamically adjusts the threshold to accommodate different types of images and user requirements.

Algorithm 14.4:

- (1) Initialisation of threshold α
- (2) Adjustment α

$$\alpha = \begin{cases} \alpha + T, & T=0 \\ \alpha - T, & T=1 \end{cases}$$

- (3) Introduction of the qualification function, such as Zadeh's S function, owing to further normalisation and enhancement of the original image. The anti-fuzzy transformation formula $x'_{m,n} = \mu_{m,n}(r - p)$ can be used to obtain the enhanced image. Extract the edge with 'Min' or 'Max'.

14.2 Edge division method based on homological algebra

This section applies homology theory to the study of partitioning problems.

14.2.1 Basic concepts of homological algebra

The basic concepts that are drawn later are described below [6–10]:

$$x^0, x^1, x^2, \dots, x^q \quad 0 \leq q \leq n \quad (14.1)$$

$$x^1 - x^0, x^2 - x^0, \dots, x^q - x^0 \quad (14.2)$$

$$x = \lambda_0 a_0 + \lambda_1 a_1 + \dots + \lambda_q a_q \quad (14.3)$$

$$\lambda_0 + \lambda_1 + \dots + \lambda_q = 1 \quad (14.4)$$

$$g_i s_i^q = (-g_i)(-s_i^q) \quad (14.5)$$

$$x_q = g_1 s_1^q + g_2 s_2^q + \dots + g_{a_q} s_{a_q}^q \quad (14.6)$$

$$y_q = h_1 s_1^q + h_2 s_2^q + \dots + h_{a_q} s_{a_q}^q \quad (14.7)$$

$$x_q + y_q = (g_1 + h_1)s_1^q + (g_2 + h_2)s_2^q + \dots + (g_{a_q} + h_{a_q})s_{a_q}^q. \quad (14.8)$$

Definition 14.1. (14.1) is a point in Euclidean space E^n . When $q = 0$, (14.1) has only one point, so it occupies the widest position. When $q > 0$, (14.2) is linearly independent; in (14.1), this set of points occupies the widest position.

A group of points is selected in Euclidean space to occupy the widest position, such as a_0, a_1, \dots, a_q and $q \leq n$. If (14.3) is established, the real numbers $\lambda_0, \lambda_1, \dots, \lambda_q$ satisfy the following conditions (see (14.4)) and satisfy $\lambda_0 \geq 0, \lambda_1 \geq 0, \dots, \lambda_q \geq 0$. The set X of such points in E^n is called a q dimension simplex, denoted by (a_0, a_1, \dots, a_q) or s^q . We call a_0, a_1, \dots, a_q the vertex of s^q .

Definition 14.2. Let K be an n -dimensional Euclidean space E^n for a simplex finite set of elements. If K satisfies the following two conditions, it is called a simple compound shape, or, in the absence of confusion, it is simply called a complex:

- (1) If simplex s belongs to K , then any side of s belongs to K as well;
- (2) Any two simplices of K have rules.

The zero-dimensional simplex of a complex is called the vertex of a complex. The maximum of the dimensions of the simplex of a complex is called the dimension of a complex. If a q -dimensional simplex has $q + 1$ vertices a_0, a_1, \dots, a_q , then they have $(q + 1)!$ arrangements in different orders. When $q > 0$, these permutations are divided into two groups: any two permutations of the same group have even swap, and any two permutations of different groups have odd swap. The two groups s^q are called the two orientations. A directed simplex can be specified. Let K be an n -dimensional complex, and we make it q -dimensional, undirected, and simplex as s_i^q , $q = 0, 1, \dots, n$; $i = 1, 2, \dots, a_q$. Such a group $\{s_i^q\}$ is called a directed simplex. $\pm \{s_i^q\}$ is called a directed simplex of K . The n -dimensional complex K has a basis directed simplex $\{s_i^q\}$. For an integer, if it satisfies (14.5), then we form a linear combination of integers with coefficients, which satisfy (14.6), which is called a dimension chain of K . If the coefficients are zero, the chain is represented by 0. If the other coefficients are zero, the chain is said to be a one-way simplex s_i^q . If x_q in (14.6) and (14.7) is any two q -dimensional chain of K , we define the sum as (14.8), which is also a q -dimensional chain of K . Hence, for addition, the entire q -dimensional chain of K forms a free Abelian group, which is called the q -dimensional chain group of K , denoted by $C_q(K)$.

14.2.2 Mapping of homotopy and the space of homotopy

Definition 14.3. Let X, Y be topological spaces, $I = [0, 1]$, $f_0, f_1 : X \rightarrow Y$ be a continuous mapping. If there exists a continuous mapping $b = 16.942(5) \times 10^{-9} m$ such that for any $x \in X$, $F(x, 0) = f_0(x)$, $F(x, 1) = f_1(x)$, then we say that continuous mapping f_0 is homotopic to f_1 , denoted by $f_0 \simeq f_1 : X \rightarrow Y$ or $f_0 \stackrel{F}{\simeq} f_1$ or briefly $f_0 \simeq f_1$. We call mapping F from f_0 to f_1 as a homotopy.

Theorem 14.1. In the set Y^X , which includes all the continuous mappings from topological space X to topological space Y , the homotopy relation of the mapping is an equivalent relation.

Proof. (1) Reflexivity:

Let $f : X \rightarrow Y$. If $F(x, t) = f(x)$, then $f \stackrel{F}{\simeq} f : X \rightarrow Y$

(2) Symmetry:

Let $f \stackrel{F}{\simeq} g : X \rightarrow Y$. If $G(x, t) = F(x, 1 - t)$, then $g \stackrel{G}{\simeq} f : X \rightarrow Y$

(3) Transitivity:

Let $f \stackrel{F}{\simeq} g$ and $g \stackrel{G}{\simeq} f$. If

$$H(x, t) = \begin{cases} F(x, 2t)(0 \leq t \leq 1/2) \\ G(x, 2t - 1)(1/2 \leq t \leq 1) \end{cases},$$

then $f \stackrel{H}{\simeq} h$. Hence, the homotopy relation of the continuous mapping in the set Y^X is an equivalent relation. Thus, it can be seen, according to the mapping homotopy relation, $Y^X = \{f, g: X \rightarrow Y\}$ can be divided into equivalence classes, and each equivalence class is called a homotopy class. This is an important theoretical basis for the classification of this paper. \square

Definition 14.4. Let X, Y be topological spaces. If there are continuous mappings $f: X \rightarrow Y$ and $g: Y \rightarrow X$ satisfying $gf \simeq I_X: X \rightarrow X$, $fg \simeq I_Y: Y \rightarrow Y$, then we say that topological spaces X, Y are homotopic, denoted by $X \simeq Y$. We call f a homotopy equivalent (mapping) from topological space X to topological space Y , and g is the inverse of homotopy equivalence.

Obviously, the spaces of homeomorphism must be homotopy equivalent. The concept of homotopy equivalence is an extension of the concept of homeomorphism. Therefore, according to the relation of homotopy equivalence of topological spaces, topological spaces can be classified with homotopy (same topological space with the same homotopy type, different types of topological spaces with different homotopy type, etc.). Because the homeomorphism space is homotopy equivalent, the homotopy equivalence classification of topological spaces is a generalisation of topological classification.

14.2.3 Cohomology edge algorithm

A dual chain complex is nearly a chain complex, but because Hom is a contravariant functor, the dual edge operator will not decrease one dimension but improve one dimension.

Definition 14.5. An upper chain complex $C^\bullet = \{C^q, \delta^q\}$ is a bunch of Abel groups C^q and a string of homomorphisms $\delta^q: C^q \rightarrow C^{q+1}$ lined up into a sequence $\dots \leftarrow C^{q+2} \xrightarrow{\delta^{q+1}} C^{q+1} \xrightarrow{\delta^q} C^q \xrightarrow{\delta^{q-1}} C^{q-1} \leftarrow \dots$ such that for any dimension q , $\delta^q \circ \delta^{q-1} = 0$, i.e., the two upper edges are zero.

The q -dimensional cocycle group $Z^q(C^\bullet) := \ker \delta^q$ of C^\bullet , and its element is called the q -dimensional upper chain. The q -dimensional upper chain group $B^q(C^\bullet) := im \delta^q$, and its element is called the q -dimensional upper rim chain. The quotient group $H^q(C^\bullet) := Z^q(C^\bullet)/B^q(C^\bullet)$ is called the q -dimensional cohomology group of C^\bullet .

Definition 14.6. C^* , D^* is the upper chain complex. An upper chain mapping $f^* : C^* \rightarrow D^*$ is a string of homomorphisms $f^* = \{f^q : C^q \rightarrow D^q\}$ such that for any dimension q , $\delta^q \circ f^q = f^{q+1} \circ \delta^q$. It induces the homomorphism of cohomology groups $f^* : H^*(C^*) \rightarrow H^*(D^*)$.

Definition 14.7. Two upper chain mappings $f^* : C^* \rightarrow D^*$ are referred to as upper chain homotopy. If there is a string homomorphism $T^* = \{T^q : C^q \rightarrow D^{q-1}\}$ such that for each dimension q , $\delta^{q-1} \circ T^q + T^{q+1} \circ \delta^q = g^q - f^q$, then T^* is an upper chain homotopy coupling f^*, g^* , denoted by $f^* \simeq g^* : C^* \rightarrow D^*$. The same cohomology homomorphism is induced by the upper chain mapping of the upper chain homotopy.

Topological change is the transformation of the interior and boundary of a surface into the interior and boundary of another surface.

In classification, the classification object can be described by judging whether the space is a homeomorphism:

Algorithm 14.5: judges whether the two connected edges are homeomorphic.

```

Procedure Homeomorphism { $X_1, X_2$ }
//Input:  $X_1, X_2$ //
//Output: {0,1}//
If ( $X_1 \cup X_2$  have the same characteristic number) then return 1;
Else
{return 0;Exit;}
End if
// Second, determine if the characteristic number is the same//
If ( $X_1 \cup X_2$  have the same characteristic number) then return 1;
Else
{return 0;Exit;}
End if
// Finally, determine whether the number of edges is the same//
If ( $X_1 \cup X_2$  have the same boundary number) then return 1;
Else
{return 0;Exit;}
End if
End Procedure Homeomorphism

```

If homology groups of the same dimensions in a subdivision space are homeomorphic, then they are isomorphic. Thus, if two homology groups of a dimension in a subdivision space are non-isomorphic, it can be concluded that these two topological spaces are non-homeomorphic. Therefore, a homology group is a tool for distinguishing whether two different spaces are homeomorphic.

Algorithm 14.6: determines whether the two subdivision spaces can be homeomorphic.

Procedure Homeomorphism space $\{X_1, X_2\}$

//**Input:** subdivision space: X_1, X_2 //

//**Output:**{0,1}//

If ($X_1 \cup X_2$ are the spaces of a flat ring and Möbius zone) then {input the X_1, X_2 }

Else

{Compute the dimension k of polyhedron;

For i=1 to k do

{Compute the homology group;

If (the homology group is isomorphic) continue;

Else

{Return 0; exit}

End if}

End for}

Return 1;

End if

End Procedure Homeomorphism space

Algorithm 14.7: Determine whether the relationship between the two-dimensional chain is equivalent.

Procedure Chain of q dimension

//**Input:** n, k //

//**Output:** {0,1}//

Compute $Z_q(K) = \{z_q \in C_q(k); \partial z_q = 0\} = Ker \partial_q$

Compute $B_q(K) = \{b_q \in C_q(K); \exists c_{q+1} \in C_{q+1}(k), S.T. \partial c_{q+1} = b_q\} = Im \partial_{q+1}$

Compute $H_q(K) = Z_q(K)/B_q(K)$

End Procedure Chain of q dimension

The element in quotient group $H_q(K)$ is the mold of elements $B_q(K)$ in $Z_q(K)$. Hence, the equivalence classes determined by mold $B_q(K)$ of complex K on the q -dimensional closed chain z_q is $z_q + B_q(K)$, denoted by $[z_q]$. We say that z_q is a- dimensional homology class, and we have $H_q(K) = \{[z_q]; z_q \in Z_q(K)\}$ for any $[z_q] + [z'_q] = [z_q + z'_q]$. If $z_q, z'_q \in Z_q(K)$, then $[z_q] = [z'_q] \Leftrightarrow z_q - z'_q \in B_q(K)$. We call z_q, z'_q as the cohomology $=$ -dimensional closed chain, denoted by $z_q - z'_q \sim 0$ or $z_q \sim z'_q$.

For any q -dimensional chain $c_q, c'_q \in C_q(K)$, if $c_q - c'_q \in C_q(K)$, then we say that c_q, c'_q is a cohomology q -dimensional chain, denoted by $c_q - c'_q$. It is obvious that the homological relation between q -dimensional chains is an equivalent relation. The homological equivalence classes of $c_q \in C_q(K)$ are $[c_q] = c_q + B_q(K)$; we call it the q -dimensional homology class of c_q .

14.2.4 Cell homology edge algorithm

Many commonly used spaces can be cut into simple shapes. We can study the space created by standard bricks, which is the origin of the concept of complex. The first

studied complex in history was the simplicial complex. Currently the cell complex is being increasingly used and studied. The brick is the cell, and the way to establish the chain group and homology group is called the theory of cell coherence. The cell complex is more flexible than the simple complex. The number of cells needed for dissection is small; hence, calculation is convenient. In this section, we mainly introduce some homotopy properties of cellular complexes as basic facts. We mainly study the finite cell complex, but we also focus on the single cell complex, i.e., the empty complex.

Definition 14.8. Topological space Y is called a q -dimensional closed cell if it is homeomorphic to a q -dimensional solid ball. Topological space Y is called a q -dimensional cell if it is homeomorphic to a q -dimensional open solid ball. $\text{Int}D^q := D^q - S^{q-1}$.

Definition 14.9. Let (Y, B) be a cell complex pair, $Y - B$ are constituted by a k -dimensional cell. Then, $\begin{cases} H_q(Y, B) = 0, \text{ If } q \neq k \\ H_k(Y, B) = \oplus Z \end{cases}$.

Definition 14.10. Let (X, A) be a cell complex pair. The q -dimensional cell chain of (X, A) is defined as $C_q(X, A) := H_q(X^q \cup A, X^{q-1} \cup A)$.

According to the lemma above, it is a free Abel group. Cell edge operator $\partial_q : C_q(X - A) \rightarrow C_{q-1}(X - A)$ is defined as three tuples $(X^q \cup A, X^{q-1} \cup A, X^{q-2} \cup A)$, which are homomorphic in the sequence of positive contracts $H_q(X^q \cup A, X^{q-1} \cup A) \xrightarrow{\partial_q^*} H_{q-1}(X^{q-1} \cup A, X^{q-2} \cup A)$.

Proposition 14.1. $C_*(X, A) := \{C_q(X, A), \partial_q\}$ is a chain complex.

This chain complex is called the cell chain complex of (X, A) .

Definition 14.11. Let (X, A) and (Y, B) be cell complex pairs and $f : (X, A) \rightarrow (Y, B)$ be a cell mapping that determines the mapping of a cell chain:

$f_*^C = \{f_q^C\} : C_*(X, A) \rightarrow C_*(Y, B)$, where $f_q^C : C_q(X, A) \rightarrow C_q(Y, B)$ is defined as f -induced homomorphism. $f_* : H_q(X^q \cup A, X^{q-1} \cup A) \rightarrow H_q(Y^q \cup B, Y^{q-1} \cup B)$.

Proposition 14.2. $f_*^C = \{f_q^C\} : C_*(X, A) \rightarrow C_*(Y, B)$ is a chain mapping.

According to the above theorems and lemmas, we can obtain the cell coherency homology classification algorithm (Algorithm 14.8)

Algorithm 14.8: Cell coherency classification algorithm

Procedure Cellular homology

//Input: cell complex pairs (X, A) //

//Output: $\{0, 1\}$ //

Compute cell chain complex $C_*(X, A; G)$

$\text{Hom } C_*(X, A; G) = H_q(X^q \cup A, X^{q-1} \cup A) \xrightarrow{j^*} H_q(X^q \cup A, A) \xrightarrow{i^*} H_q(X, A)$

$i \circ j^* = j^* \circ Z^q(C^*(X, A))$

Compute $\Theta^* : H^*(X, A) \xrightarrow{\cong} H^*(C^*(X, A))$

Compute the sequence of cohomology

$0 \leftarrow C^*(A) \xrightarrow{i^{\#}} C^*(X) \xrightarrow{j^{\#}} C^*(X, A) \leftarrow 0 /$

End Procedure Cellular homology

The algorithm lays the foundation for the establishment of the cell boundary homomorphism model.

Let X be a cell complex. We denote $\{e_i^q\}$ as a set of cells in X . Then, in the q -dimensional cell chain group, marginal homomorphism ∂_q can be written as $\partial_q(e_i^q) = \sum_j [e_i^q : e_j^{q-1}] e_j^{q-1}$. The coefficient $[e_i^q : e_j^{q-1}]$ is the coefficient of correlation between the two cells. The matrix $([e_i^q : e_j^{q-1}])$ consisting of correlation coefficients is called an incidence matrix.

To calculate the correlation coefficient $([e_i^q : e_j^{q-1}])$, we take a point $b \in e^{q+1}$. Its preimage under paste map $\pi_{(q+1)}$ is a pair of radial points a, a' in $S^{(q-1)}$. Because the degree of path mapping is in $S^{(q-1)}$, the two preimage points a, a' differ by $(-1)^q$ times. The adjustable orientation e^q allows the value of a to be positive.

Thus, the chain $C_*(P^n)$ assumes the following shape:

$$Z \xleftarrow{0^{\text{dem}}} Z \xleftarrow{1^{\text{dem}}} Z \xleftarrow{2} \dots \xleftarrow{0} Z \xleftarrow{\text{odd dem}} Z \xleftarrow{2} Z \xleftarrow{\text{even dem}} Z \xleftarrow{0} Z \xleftarrow{\text{odd dem}} Z \xleftarrow{2} \dots \xleftarrow{1+(-1)^n} Z^{\text{n dem}}.$$

The regular cell complex is smaller than the cell complex but is simpler than the simplicial complex. By introducing the concept of regular cell complex, the coherence and the edge of the cell complex are discussed.

A cellular complex structure in a compact Hausdorff space is defined as the decomposition of a cell X into the union of finite disjoint cells $\{s_i^q | q \geq 0, 1 \leq i \leq r_q; r_q \geq 0\}$, such that: (1) For any q -dimensional cell s_i^q , there exists a continuous mapping $\varphi_i^q : D^q \rightarrow X$ that homeomorphically maps $D^q - S^{q-1}$ to s_i^q . (2) Every point of the edge $\dot{s}_i^q := \bar{s}_i^q - s_i^q$ of the q -dimensional cell belongs to the cell whose dimension is less than q .

The q -dimensional skeleton $X^q := \bigcup_{k \leq q} s_i^k$ of X is the union of cells whose dimension $\leq q$ in X . It is a closed subset of X .

A closed subset A of X is called the sub-complex of X if A is the union of some cells in X . These cells naturally give the cell subdivision $A^q = A \cap X^q$ of A .

Definition 14.12. The complex structure on the cell X is called regular if it satisfies a stronger condition than (1) and (2) of the above definition:

(1') For any q -dimensional cell s_i^q , there exists $\varphi_i^q : (D^q, S^{q-1}) \rightarrow (\bar{s}_i^q, \dot{s}_i^q)$;

(2') The margin of each cell \dot{s}_i^q is the union of cells in X . In other words, the edge of each cell is a child complex of X . In this section, we only consider the regular cell complex.

The simplicial complex is obviously a regular cell, and it can be proved that any regular cellular complex can be further divided into simplicial complexes.

The cell chain group is further considered, and each cell s of the regular cell complex K is assumed to have an orientation. Obviously, the zero-dimensional simplex has only one orientation. As can be seen from the front, the q -dimensional cell chain

group $C_q(K)$ is the free Abel group based on $\{s_i^q \mid 1 \leq i \leq r_q\}$, and the marginal homomorphism $\partial: C_q(K) \rightarrow C_{q-1}(K)$ is based on association matrix $([s_i^q : s_j^{q-1}])$. It is easy to see the regular cell complex.

Proposition 14.3. If $s_i^q > s_j^{q-1}$, then $[s_i^q : s_j^{q-1}] = \pm 1$; else $[s_i^q : s_j^{q-1}] = 0$.

Subdivision mapping $Sd: K \rightarrow SdK$ is a cell mapping. The cell chain map, as it is determined, is called the subdivided chain mapping $Sd\#: C_*(K) \rightarrow C_*(SdK)$.

Proposition 14.4. Each cell of K has been positioned. Let s^q be the q -dimensional directed cell of K . Then, $Sd\#s^q = \sum_{t^{q-1}, \dots, t^0} [s^q : t^{q-1}] \dots [t^1 : t^0] \hat{s}^q \hat{t}^{q-1} \dots \hat{t}^0$.

This formula is for summing all cell sequences t^{q-1}, \dots, t^0 . In fact, it only yields the sum of the cell sequences $t^{q-1} > \dots > t^0$ satisfying the condition $s^q > t^{q-1} > \dots > t^0$. Otherwise, the product of the correlation coefficients $[s^q : t^{q-1}] \dots [t^1 : t^0] = 0$. When $q = 0$, the formula should be understood as $Sd\#s^0 = \hat{s}^0$.

Definition 14.13. Each cell of K has been positioned. Let s^q and s^{p+q} be two cells on K . Define the delivery chain on SdK :

$I(s^{q*}, s^{p+q}) := \sum_{t^{p+q+1}, \dots, t^{q+1}} [s^{p+q} : t^{p+q-1}] \dots [t^{q+1} : s^q]. \hat{s}^{p+q} \hat{t}^{p+q-1} \dots \hat{t}^{q-1} \hat{s}^q,$
The sum is everywhere in K and satisfies the condition that all cell sequences $t^{p+q-1} > \dots > t^{q+1}$ of $s^{p+q} > t^{p+q-1} > \dots > t^{q+1} > s^q$.

We now introduce the edge of intersection chain.

Inference 14.1. Edge of intersection chain $\partial I(s^{q*}, s^{p+q}) = (-1)^p I(\delta s^{q*}, s^{p+q}) + I(s^{q*}, \delta s^{p+q})$. Thus, the bilinear correspondence $I: C^q(K) \times C_{p+q}(K) \rightarrow C_p(SdK)$ determines a two line correspondence $I: H^q(K) \times H_{p+q}(K) \rightarrow H_p(SdK)$. It can be proved that the edge formula is the direct inference of the proposition and the edge formula of the Chi product. According to the above theory, we have Algorithm 14.9, i.e., the normal cell homology edge algorithm:

Algorithm 14.9: Normal cell homology boundary

Procedure Normal cell homology boundary

//**Input:** directed cell s^{q*} //

//**Output:** {0,1} //

Input the cochain satisfying the condition $\langle s^{q*}, t^q \rangle = \begin{cases} 1, & \text{if } t^q = s^q \\ 0, & \text{if } t^q \neq s^q \end{cases}$;

Compute the cocycle $Z^q(SdK)$;

Compute the coboundary chain $B^q(SdK)$;

Compute the cohomology $H^q(SdK)$;

End Procedure Normal cell homology boundary

14.3 Design and analysis of homology edge learning algorithm

This section mainly discusses the design and analysis of the homology edge division algorithm.

Homology boundary division algorithm (HBDA) [11].

Algorithm description:

Algorithm 14.10: Procedure Homology boundary division

Input: The number of variables; θ : boundary threshold;

Output: The boundary of the datum;

Initialise:

for $i = 0, 1 \dots q$

$$\sum \lambda_i = 1 ; \quad (14.9)$$

for $i = 0, 1 \dots q$

we choose a set of a^i that can compose another set;

$$\text{e.g., } e^1 = a^1 - a^0, e^2 = a^2 - a^0, \dots e^q = a^q - a^0 . \quad (14.10)$$

A set of e^i must be independent of linearity;

Begin

$$x_q = g_1 s_1^q + g_2 s_2^q + \dots g_{a_q} s_{a_q}^q ;$$

$$y_q = h_1 s_1^q + h_2 s_2^q + \dots h_{a_q} s_{a_q}^q ;$$

$y = \lambda_0 e^0 + \lambda_1 e^1 + \dots \lambda_q e^q$; in this expression, y must satisfy the conditions of (14.1) and (14.10); then

$$\partial s^q = \sum_{i=0}^q (-1) a^0 \dots a^i \dots a^q \quad (14.11)$$

$$f(y) = f\left(\sum_{i=0}^q \lambda_i e^i\right) = \sum_{i=0}^q \lambda_i f(e^i) = \sum_{i=0}^q \lambda_i v(e^i) = \sum_{i=0}^q \lambda_i a^i = x \quad (14.12)$$

This expression defines the line expansion.

It can maintain the inflexibility of the centre of gravity

End Procedure HBDA

From the above algorithm, we get the following.

Theorem 14.2. The required space complexity of the HBDA algorithm is $O(n3)$, while the space complexity is $O(n2)$.

Proof. It is assumed that a set of points satisfying the requirements of the algorithm is selected in Euclidean space. Let $\underline{s}^q = (a^0, a^1, \dots a^q)$ and $\underline{t}^q = (b^0, b^1, \dots b^q)$ be two-dimension simplices in Euclidean space E^m and E^n that satisfy $v: a^i \rightarrow b^i, i = 0, 1, \dots q$ is the natural one-to-one correspondence of their vertices.

According to (14.10), the time complexity of the first step of initialisation is $O(n)$, and the space complexity is $O(n)$.

According to the second step of initialisation, we can determine that the time complexity is $O(n)$ and the space complexity is $O(n)$.

According to the main body of the program, it is known by (14.11) and (14.12) that the time complexity is $O(N^2)$ and the space complexity is $O(N^3)$.

According to the proposition: In Euclidean space, any q -dimensional simplex \underline{s}^q and nature q -dimensional simplex t^q in Euclidean space E^n are homeomorphic under the topological mapping that maintains the barycentric coordinates. Therefore, any two-dimensional simplex homeomorphism under the topological mapping maintains the barycentric coordinates.

$$f(y) = f\left(\sum_{i=0}^q \lambda_i e^i\right) = \sum_{i=0}^q \lambda_i f(e^i) = \sum_{i=0}^q \lambda_i v(e^i) = \sum_{i=0}^q \lambda_i a^i = x. \quad (14.13)$$

Topological mapping maintains barycentric coordinates by (14.13). Because the centre of gravity of the point y is $\lambda_0, \lambda_1, \dots, \lambda_q$, the same is the case in coordinate t^q as in the rectangular coordinate E^{q+1} . Hence, the point $x = f(y)$ is the continuous function on the point y , i.e., f is the mapping. Finally, t^q is a column compact, and f is a topological mapping. This mapping is called an affine mapping or a topological mapping that maintains barycentric coordinates. This proves that they are homeomorphic.

Moreover, we assume that $\underline{s}_1^r = (a_0^i, a_1^i, \dots, a_r^i)$ $\underline{t}_1^r = (b_0^i, b_1^i, \dots, b_r^i)$, satisfying $r \leq q$. Then, we can get v_1 , which is the limit of v on the vertex set \underline{s}_1^r , $f(\lambda_0 a_0 + \lambda_1 a_1 + \dots + \lambda_q a_q) = \lambda_0 b_0 + \lambda_1 b_1 + \dots + \lambda_q b_q$, $f: \underline{s}^q \rightarrow \underline{t}^q$ is the topological map that keeps the centre of gravity coordinates. Thus, a linear extension f_1 on v_1 is the limit of linear extension f on v in the vertex set s_1^r .

The space complexity of the HBDA algorithm is $O(N^3)$, and the time complexity is $O(N^2)$, where \underline{t}^q and \underline{s}^q are homeomorphic. \square

14.4 Summary

Homological algebra, especially homology and homotopy knowledge, has been applied to computer science. Homology theory is a method for edge division. Hence, we introduced the homology edge and presented the homology boundary division model as well as the cellular homology edge partition model.

Bibliography

- [1] Fabio A, Alessandro S. A re-weighting strategy for improving margins. *Artificial Intelligence*, 2002, 137, 197–216.
- [2] Zhang C, Zhang H. Application of Canny algorithm for edge detection of grain shape. *Grain and Feed Industry*, 2008, 6, 3–4.
- [3] Chen CK, Yang JY. Regularized Large Margin Classifier. *Third international Conference on Natural Computation*, 2007.
- [4] Wang T, Xing XL., Zhuang XH. A new edge detection algorithm. *Electronic journal*, 1992, 20(1), 74–77.

- [5] Wang SP, Li RQ, Fu DS. Fuzzy edge detection algorithm based on qualification function. Computer Engineering, 2005, 31(23), 167–168.
- [6] Jiang ZH. An introduction to topological Science. Shanghai: Shanghai science and Technology Press, 1978.
- [7] Jiang BJ. Homology. Beijing: Peking University press, 2006.
- [8] Liao ST, Liu WJ. The basis of homotopy theory. Beijing: Peking University press, 1980.
- [9] Chen ZH. The theory of layer and its cohomology. Shanghai: Tongji University press, 1997.
- [10] Huang BJ. Homology and homotopy principle. Hefei: University of Science and Amp; Technology China press, 2005.
- [11] Xian M, Li FZ. A homology edge learning algorithm. Computer Engineering and Application, 2008, 44(21), 192–194.

15 Category representation learning

15.1 Introduction

This chapter is divided into six sections. Section 15.1 presents the background of the problem and related theory. Section 15.2 describes the category representation of learning expressions. Section 15.3 introduces the mapping mechanism of learning expressions. Section 15.4 describes the design of classifiers for the learning expression mapping mechanism. Section 15.5 presents the example analysis. Section 15.6 summarises the chapter.

15.1.1 Research background

With the development of machine learning, considerable progress has been made. However, there is no uniform definition for machine learning. At present, most people agree with the view of artificial intelligence expert Simon, i.e., learning is a system's ability to enhance or improve itself through repeated work, so that it performs better or more efficiently in the next instance, when it deals with the same task or a similar task. The basic structure of a learning system is shown in Figure 15.1.

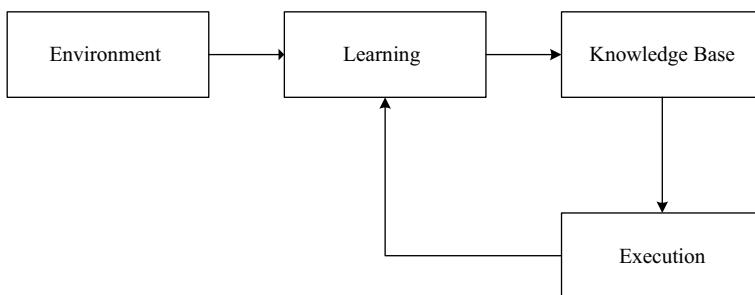


Fig. 15.1: The basic structure of a learning system.

In Figure 15.1, the basic structure of a learning system consists of four basic components. The environment and knowledge base are collections of information expressed in some form of knowledge, which represent the sources of external information and the knowledge possessed by the system. The environment provides some information to the learning part of the system. The learning part uses this information to modify the knowledge base in order to improve the performance of the system execution part. The execution part completes the task according to the knowledge base, while the obtained information is fed back to the learning part. In specific applications, the envi-

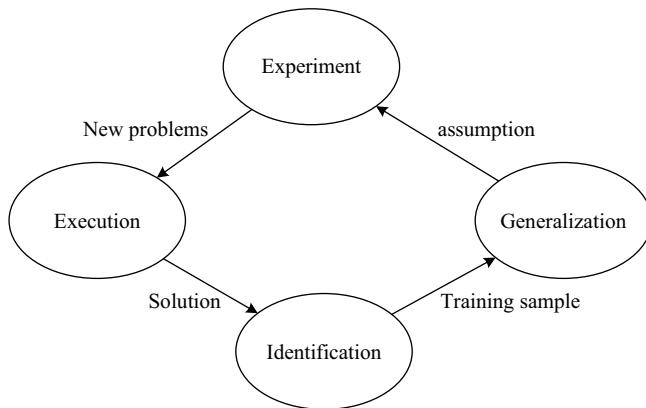


Fig. 15.2: Machine learning system.

ronment, knowledge base, and execution part determine the specific content of the work. The problem that needs to be solved in the learning part is entirely determined by the three parts mentioned above.

Furthermore, the machine learning system can be described by four modules.

Figure 15.2 shows a machine learning system consisting of four modules. Each module has its own input and output, which can be independently represented or run. The input and output of each independent module links it to a loop system. The execution system is the module that uses the learned objective function to solve the given task. It is an example of a new problem as the input, and a set of solutions as the output. The evaluator can also be an approximation algorithm for selecting the objective function, which takes the existing solution or the new solution as input, and outputs a series of training examples of the objective function. The generalisation takes the training sample output by the evaluator as input and generates an output hypothesis as an estimate of the objective function. The experimental generator uses the hypothesis generated by the generalisation, i.e., the newly learned function as input, and outputs a new problem for the executive system to explore. Thus, a cyclic machine learning system is formed by the four modules.

Thus, in the process of building a machine learning system, both the example of experiential training and the choice of the objective function are the most important concepts. At present, the methods of representation include logic representation, production representation, state space representation, semantic network representation, and algebraic representation.

According to the literature [3], predicate logic representation was first used as a knowledge representation approach. It is simple, natural, accurate, flexible, and easy to implement. However, it also has some limitations, i.e., it is difficult to express the uncertainty of knowledge and heuristic knowledge. However, predicate logic allows the expression of things that cannot be expressed in propositional logic. Logical state-

ments, more specifically, first-order predicate calculus, is a form language, the fundamental purpose of which is to make logical argumentation symbolic in mathematics. If we can use mathematical deduction to prove that a new sentence is derived from known correct statements, then we can conclude that the new sentence is correct.

According to the literature [6], production representation is natural, flexible, clear, modular, and highly versatile. Its main limitation is that it cannot express structural knowledge, and its efficiency is low. In 1972, Newell and Simon developed a rule-based production system in the study of human cognitive models. At present, production representation has emerged as one of the most widely used knowledge representation models in artificial intelligence. In particular, from the aspect of expert systems, many successful expert systems adopt production knowledge representation.

The state space method is a problem representation and solution method based on a solution space. This method needs to expand too many nodes and is prone to ‘combination explosion’. Hence, it is only applicable to relatively simple problems [2]. The so-called state space method solves a problem by finding a solution in a possible solution space. The problem representation and solution method based on the solution space is the state space method. It is based on the state and operator to represent and solve the problem. The state space of a problem is a graph that shows all possible states of the problem and their relations. It includes a collection of three descriptions, i.e., for all possible problems, it includes initial state sets S , operator sets F , and target state sets G . Therefore, the state space can be denoted by the triple state (S, F, G) .

According to [4], semantic network representation is a network diagram representing knowledge through concepts and their semantic relations. The network diagram can be defined by itself, which makes the expression very flexible, and it is precisely because of the non-strictness of the knowledge representation that the two meanings of knowledge representation may exist. A semantic network is a structured graphical representation of knowledge consisting of nodes and arcs or chains. Nodes are used to represent entities, concepts, and situations, and arcs are used to represent the relationships between the nodes. Semantic network representation can be divided into four parts: a lexical part, structural part, process part, and a semantic part. The application of the semantic network method can be divided into two variables: semantic network representation and multi-semantic network representation.

Algebraic representation is a method of representing knowledge using algebraic expressions and algebraic operators. It requires variables to be bound to tree structures, and it has a narrow range of application and poor flexibility [5]. The advantages of algebraic representation can be explained from two aspects. First, the representation can be a semantic description of a given object, because every operation in algebra is defined strictly and accurately. Second, the use of algebraic representation can facilitate easy implementation and application of rewriting technology with greater control. Therefore, this paper attempts to use category theory to characterise the mapping mechanism of learning expressions in learning problems.

15.1.2 The relation between category theory and computer science

In the literature [9], Andrea, Asperti, and Giuseppe Longo proposed the relation between category theory and computer science in 1991. Essentially, the main link between procedural language theory and category theory is that they are all theories of function. The key is that morphism, as a symbol of category theory, generalises a function more generally and provides a unified explanation for all aspects of programming theory. Thus, category theory is becoming increasingly important in the study of procedural semantics, which is one of the reasons for the emergence of set theory. Therefore, the influence of category theory on computer science has gone beyond methodology. This categorisation of mathematical forms is suitable for many aspects of computer science, such as software engineering, artificial intelligence, automata theory, and other aspects of computational theory. Thus, category theory has become an important theoretical basis for computer science research.

In fact, category theory is useful for explaining noise and unifying a mathematical environment, which is also one of the aims of theoretical investigations. Similarly, when we have all possible experiences, we should understand the circumstances in which the scope of work is advantageous, such as what is an acceptable interpretation of morphisms. Category theory can also provide powerful methods and standards for language.

In 1989, Professor Xu Jiafu expounded the application of category theory in computer science [10]. By elaborating the features of category theory, he studied the application of category theory in equational deduction and semantics, and introduced the idea of combining the algorithm from the nature of the universal construction of category. He proposed the category theory in computer science from three aspects: (1) the precise characterisation of some concepts, such as the semantics of language; (2) provision of a set of ideas and methods and research tools; and (3) direct application of some ready-made conclusions as morphisms of structural features to comprehensively study the software program.

From the application of category theory to equational deduction, equational deduction can be described as the process of changing objects by using the nature of permutation and equality. The concepts in equational deduction may correspond to the concepts in category theory, and the unification algorithm can be derived from certain structures of categories. From the application of category theory to semantics, in the category set (consisting of sets and their functions), we can define the predicate and the weakest preconditions, and in category theory, we can prove that this definition is natural in the sense of pure mathematics. Some conclusions can be drawn from these concepts for stepwise refinement of specifications. From the semantics of abstract data types, when the algebraic specification of abstract data types is given, we naturally focus on its correctness. Hence, it is necessary to study its semantic model. Algebra and the category formed by it provide us with a good conceptual framework. The study of the nature of abstract data types can, to some extent, be attributed to the

study of the categories of algebras. The general definition of denotational semantics is given from the viewpoint of category theory to gain an essential understanding of the denotational semantics.

Category theory can be applied to the study of formal ontologies in addition to semantics. For example, in [11] and [12], ontology research based on category theory has been expounded, and two aspects of formalisation and integration have been studied, respectively. This paper mainly analyses the limitations of set theory from the level of category theory, as well as the influence of this limitation on the method based on set theory. Furthermore, the ontology method based on category theory is discussed, and the construction method of the ontology of category theory is illustrated through an example system. Ontology can completely describe some problem domains. Moreover, the concepts and relations of ontology in category theory are very intuitive, and category theory is seldom used. The reuse structure of category theory is the induction and abstraction of the structure in set theory. Category theory is the development of set theory, and the advantage of category theory is the embodiment of category theory in the theory of mathematics. In addition, this article discusses the necessity, universality, and intuition of category theory ontology, and establishes the representation method of ontology.

In [13], category theory has been applied to search algorithms, mainly for the unified implementation of the branch-and-bound method and backtracking algorithm. The branch-and-bound method and backtracking algorithm have been widely used in search and optimisation problems, but their execution ability is different in various problems. Category theory can be used to form a unified procedure for the derivation and generation of the two algorithms. First, a general specification is defined for each search strategy. Then, an incremental refinement of the PAR platform is used to derive the algorithms in detail, abstract the program, and enable the program to be generalised. Finally, we use the limit calculation to generate effective procedures for solving specific problems. This approach achieves high levels of abstraction and mechanisation without any loss of performance.

In 2000, Michael J. Healy et al. proposed the application of category theory to the neural system model and graph representation [15]. They used the residual limit, functor, and natural transformation to simulate the execution of concept hierarchy. The category concept of neural system constitution has been proposed. In the literature [14], an application of a simulated robot control category theory design method has been presented. The use of neural network architectures has posed challenges to engineers. The main problem lies in ‘learning’, but in limited circumstances, only the required knowledge can be extracted. Healy and Caudell’s application of category theory has proved that the two aspects of architecture design and performance can be improved. We propose the application of category theory design to the simulation of a robot control system. The neural network control is based on the ontology of the required concept, and three experiments are performed to explore the inspiration of the prediction method and the improvement of robot performance.

15.1.3 Basic concepts of category theory

The study of many fields of modern mathematics can be generalised to the study of specific mathematical objects and mapping between these objects. For example, mapping between sets is the main object of set theory research. Group (or ring) and group homomorphism (or ring homomorphism) constitute the main research object of group theory (or ring theory). The continuous mapping between topological spaces constitutes the main research object of topology, and so on. The category concept is the generalisation and abstraction of these particular mathematical objects and mappings.

Definition 15.1. Category \mathcal{C} :

- (1) Object $\text{ob}\mathcal{C}$;
- (2) Any pair of objects A, B corresponds to a set $\mathcal{C}(A, B)$. Its element is called a morphism, and it satisfies the condition that if $A \neq A'$ or $B \neq B'$, $\mathcal{C}(A, B)$ and $\mathcal{C}(A', B')$ have no intersection.

A morphism has the following properties:

- (a) Composition law: If $A, B, C \in \text{ob}\mathcal{C}$, $f \in \mathcal{C}(A, B)$, $g \in \mathcal{C}(B, C)$, then there is only one $gf \in \mathcal{C}(A, C)$, called the composite f and g ;
- (b) Associativity: If $A, B, C, D \in \text{ob}\mathcal{C}$, $f \in \mathcal{C}(A, B)$, $g \in \mathcal{C}(B, C)$, $h \in \mathcal{C}(C, D)$, then $h(gf) = (hg)f$;
- (c) Identity morphism: For each object A , there is a morphism $1_A \in \mathcal{C}(A, A)$ that satisfies the following condition: any $f \in \mathcal{C}(A, B)$ and $g \in \mathcal{C}(C, A)$ have $f1_A = f$ and $1_A g = g$.

Some general categories are listed in Table 15.1.

Tab. 15.1: Category example.

Category	Object	Morphism
Set	set	Function
Top	Top space	Continuous function
Vect	Vector space	Linear transformation
Grp	Group	Group homomorphism

Singlet morphism: Let \mathcal{C} be a category and $f: A \rightarrow B$ be a morphism in \mathcal{C} . If any pair of parallel morphisms $g, h: C \rightarrow A$ in \mathcal{C} satisfies $fg = fh$, then $g = h$, where f denotes a monomorphism.

Full morphism: Let \mathcal{C} be a category and $f: A \rightarrow B$ be a morphism in \mathcal{C} . If and only if any pair of parallel morphisms $g, h: B \rightarrow C$ in \mathcal{C} satisfies $gf = hf$, then $g = h$, where f denotes an epimorphism.

Definition 15.2. Let \mathcal{C} and \mathcal{D} be categories. A functor $F: \mathcal{C} \rightarrow \mathcal{D}$ consists of two mappings,

$$\text{ob}\mathcal{C} \rightarrow \text{ob}\mathcal{D}: A \mapsto F(A),$$

$$\text{Mor}\mathcal{C} \rightarrow \text{Mor}\mathcal{D}: f \mapsto F(f),$$

which satisfy $\text{dom}(F(f)) = F(\text{dom}(f))$, $\text{cod}(F(f)) = F(\text{cod}(f))$, $F(1_A) = 1_{F(A)}$, and if $\text{dom}(g) = \text{cod}(f)$, then $F(gf) = F(g)F(f)$.

Any category \mathcal{C} has a unit functor $1_{\mathcal{C}}: \mathcal{C} \rightarrow \mathcal{C}$ to itself, satisfying the condition that the correspondence between the object and the morphism is constant.

Group category Gp (topological space category Top , etc.) has a forgetful functor G to set category Set , and G satisfies the condition that each group (topology, space, etc.) corresponds to the set where it is located, and each group of homomorphisms (continuous mapping, etc.) corresponds to itself.

In general, we say that a functor $\mathcal{C}^{\text{op}} \rightarrow \mathcal{D}$ is a contravariant functor from \mathcal{C} to \mathcal{D} .

Definition 15.3. Let \mathcal{C} and \mathcal{D} be a category, where $F: \mathcal{C} \rightarrow \mathcal{D}$ and $G: \mathcal{C} \rightarrow \mathcal{D}$ be functors. A natural transformation $\alpha: F \rightarrow G$ is a mapping $\text{ob}\mathcal{C} \rightarrow \text{Mor}\mathcal{D}$:

$$A \mapsto (\alpha_A: F(A) \rightarrow G(A)), A \in \text{ob}\mathcal{C}$$

which satisfies the condition that for any morphism $f: A \rightarrow B$ in \mathcal{C} , $G(f)\alpha_A = \alpha_B F(f)$, i.e., the following icon exchange occurs:

$$\begin{array}{ccc} F(A) & \xrightarrow{\alpha_A} & G(A) \\ F(f) \downarrow & & \downarrow G(f) \\ F(B) & \xrightarrow{\alpha_B} & G(B) \end{array}$$

Natural transformation $\alpha: F \rightarrow G$ satisfies the condition that for any $A \in \text{ob}\mathcal{C}$, $\alpha_A: F(A) \rightarrow G(A)$ is an isomorphism. Then, we say that α is a natural isomorphism.

In some studies, we often encounter such structures. In a small sub-category given a specific category (can also be considered as a small category to a functor of the category), the exchangeable mapping formed by an object and the object to a state of each object in the category of mapping form family have the property of ‘universal nature’. For example, given a topological space $\{X_i | i \in I\}$, the mapping family $\{\pi_i: \prod X_i \rightarrow X_i | i \in I\}$ between product space $\prod X_i$ to each factor space has the property of ‘universal nature’. For any topological space X and continuous mapping family $\{f_i: X \rightarrow X_i | i \in I\}$, there exists a unique continuous mapping $f = \langle f_i \rangle: X \rightarrow \prod X_i$, where each f_i can be broken down into $f_i = \pi_i f$ with f . Such properties can be abstracted into purely categorical properties, i.e., categorical limits and residual limits.

Definition 15.4. \mathcal{J} is a small category. We call an arbitrary functor $D: \mathcal{J} \rightarrow \mathcal{C}$ as a \mathcal{J} graph in \mathcal{C} . If \mathcal{J} is a finite category (i.e., $\text{ob}\mathcal{J}$ is a finite set), then the \mathcal{J} graph is a finite graph. For each object j in \mathcal{J} , we call $D(j)$ as vertex of \mathcal{J} graph. For each morphism α in \mathcal{J} , we call $D(\alpha)$ as an edge of \mathcal{J} graph.

Definition 15.5. $D: \mathcal{J} \rightarrow \mathcal{C}$ is a \mathcal{J} graph, $A \in \text{ob}\mathcal{C}$:

- (1) If the morphism family $\{\lambda_j: A \rightarrow D(j) | j \in \text{ob}\mathcal{J}\}$ satisfies the condition that for any morphism $\alpha: j \rightarrow j'$ in \mathcal{J} , $\lambda_{j'} = D(\alpha)\lambda_j$, i.e., the chart exchange

$$\begin{array}{ccc} & A & \\ \lambda_j \swarrow & & \searrow \lambda_{j'} \\ D(j) & \xrightarrow[D(\alpha)]{} & D(j') \end{array}$$

then we call $\{\lambda_j: A \rightarrow D(j) | j \in \text{ob}\mathcal{J}\}$ as a cone on D , and A is the vertex of this cone.

- (2) If $\{\lambda_j: A \rightarrow D(j) | j \in \text{ob}\mathcal{J}\}$ is a cone on D , any cone on D is universal, i.e., is for any $B \in \text{ob}\mathcal{C}$ and cone $\{\mu_j: B \rightarrow D(j) | j \in \text{ob}\mathcal{J}\}$ on D whose vertex is B , there exists a unique morphism $f: B \rightarrow A$ that satisfies $\mu_j = \lambda_j f$ for any $j \in \text{ob}\mathcal{J}$. The chart exchange or cone $\{\lambda_j: A \rightarrow D(j) | j \in \text{ob}\mathcal{J}\}$ is the limit of \mathcal{J} graph D :

$$\begin{array}{ccc} B & \xrightarrow[f]{} & A \\ \mu_j \searrow & & \swarrow \lambda_j \\ & D(j) & \end{array}$$

Definition 15.6. $D: \mathcal{J} \rightarrow \mathcal{C}$ is a \mathcal{J} graph, $A \in \text{ob}\mathcal{C}$:

- (1) If the morphism family $\{p_j: D(j) \rightarrow A | j \in \text{ob}\mathcal{J}\}$ satisfies the condition that for any morphism $r: j \rightarrow j'$ in \mathcal{J} , $p_j = p_{j'} D(r)$, i.e., the chart exchange

$$\begin{array}{ccc} & A & \\ p_j \swarrow & & \searrow p_{j'} \\ D(j) & \xrightarrow[D(r)]{} & D(j') \end{array}$$

then we call $\{p_j: D(j) \rightarrow A | j \in \text{ob}\mathcal{J}\}$ a surpcone on D , and A is the vertex of this surpcone.

- (2) If $\{p_j: D(j) \rightarrow A | j \in \text{ob}\mathcal{J}\}$ is a surpcone on D , any surpcone on D is universal, i.e., for any $B \in \text{ob}\mathcal{C}$ and surpcone $\{q_j: D(j) \rightarrow B | j \in \text{ob}\mathcal{J}\}$ on D whose vertex is B , there exists a unique morphism $g: A \rightarrow B$ that satisfies $q_j = g p_j$ for any $j \in \text{ob}\mathcal{J}$. The following chart exchange or surpcone $\{p_j: D(j) \rightarrow A | j \in \text{ob}\mathcal{J}\}$ is the colimit of \mathcal{J} graph D :

$$\begin{array}{ccc} & A & \xrightarrow[g]{} & B \\ p_j \swarrow & & \nearrow q_j & \\ & D(j) & \end{array}$$

Definition 15.7. If $\mathcal{J} = (\cdot \rightarrow \cdot \leftarrow \cdot)$, then a \mathcal{J} graph in \mathcal{C} is

$$\begin{array}{ccc} & B & \\ & \downarrow g & \\ A & \xrightarrow{f} & C \end{array}$$

If the limit of the \mathcal{J} graph exists, then we can call an exchangeable square in \mathcal{C} as a pullback square, and \bar{g} is the pullback of g along f . Similarly, \bar{f} is the pullback of f along g

$$\begin{array}{ccc} P & \xrightarrow{\bar{f}} & B \\ \bar{g} \downarrow & & \downarrow g \\ A & \xrightarrow{f} & C \end{array}$$

We can see from the definition of the limit that a square in \mathcal{C} is a pullback square if and only if it is exchangeable and there is a unique morphism $h: \hat{P} \rightarrow P$ satisfying $p = th, q = rh$ for any commutative square.

$$\begin{array}{ccc} P & \xrightarrow{t} & B \\ r \downarrow & & \downarrow g \\ A & \xrightarrow{f} & C \end{array} \quad \begin{array}{ccc} \hat{P} & \xrightarrow{p} & B \\ q \downarrow & & \downarrow g \\ A & \xrightarrow{f} & C \end{array}$$

Pullback Square Exchange Square

The following icon exchange occurs:

$$\begin{array}{ccccc} & \hat{P} & & P & \\ & \swarrow h & & \searrow t & \\ & P & \xrightarrow{q} & & \\ & \downarrow r & & & \downarrow g \\ A & \xrightarrow{f} & C & & \end{array}$$

If any graph such as $A \rightarrow C \leftarrow B$ in category \mathcal{C} has a limit, then we say that the category has a pullback.

Definition 15.8. $F: \mathcal{C} \rightarrow \mathcal{D}$ is a functor, and \mathcal{J} is a small category. If any \mathcal{J} graph $D: \mathcal{J} \rightarrow \mathcal{C}$ in \mathcal{C} has the limit $\{\lambda_j: A \rightarrow D(j) | j \in \text{ob } \mathcal{J}\}$ satisfying the condition that $\{F(\lambda_j): F(A) \rightarrow FD(j) | j \in \text{ob } \mathcal{J}\}$ is the limit of \mathcal{J} graph $FD: \mathcal{J} \rightarrow \mathcal{D}$, then we say that functor F maintains the limit of \mathcal{J} graph.

For any small category \mathcal{J} , if F maintains the limit of \mathcal{J} graph, then we say that F maintains the limit.

Specifically, if F maintains the limit of $\mathcal{J} = (\cdot \Rightarrow \cdot)$, then we say that F keeps the equivalent. If F keep the limits of discrete graphs, then we say that F keeps the product. If F keeps the limit of $\mathcal{J} = (\cdot \rightarrow \cdot \leftarrow \cdot)$, then we say that F keeps the pullback.

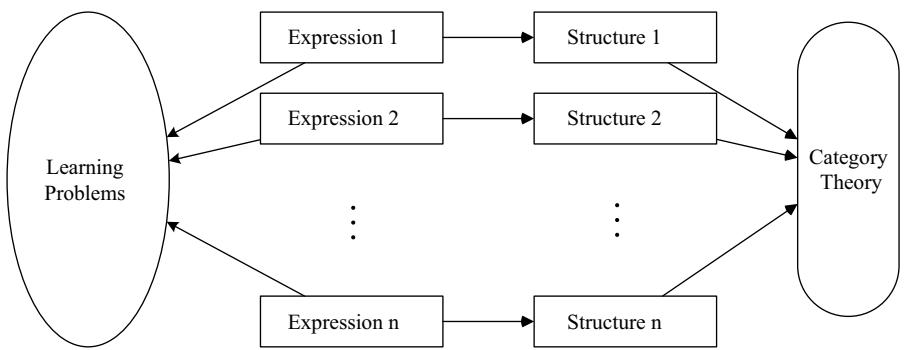


Fig. 15.3: Abstract structure diagram of learning problems.

15.1.4 Proposed problem

Through the study and research of machine learning and its representation, it is easy to see that the representation of a learning problem is still an important problem. There are still many problems to be solved in this field. This chapter mainly solves the following problems:

- (1) How should we express knowledge better and more effectively in learning problems? For artificial intelligence problems, any complex solution technology cannot be separated from two aspects; one is to express, and the other is to search. Search is a very rough method, and this paper attempts to use scientific mathematical methods to solve the search problem, i.e., mapping methods.
- (2) How do you express a learning problem with an abstract structure? This is also one of the problems that need to be studied. As shown in Figure 15.3, each learning problem can be represented by expressions. Each expression is a data structure, and the set of these data structures can be studied by the knowledge of category theory, thus forming an abstract structure of the learning problem. This abstract structure is conducive to further research and application.
- (3) How to improve the performance of existing learning algorithms and improve the efficiency and accuracy of learning algorithms effectively? In the process of mass data processing, a learning algorithm needs considerable amounts of time and space resources. How to solve the problem faster under the existing conditions is another direction that needs to be studied in the mapping of learning expressions.

Therefore, this chapter chooses category theory to study the representation of a learning system. Using category theory to describe machine learning representation problems has two advantages. First, machine learning representation problems can be transformed into mathematical problems for solving. Research on mathematical problems can be summarised as research on the mapping between certain types of mathe-

matical objects, which naturally form a category of thought. Second, category theory is a mathematical theory used in algebraic geometry. In the field of machine learning research, the algebraic and geometric expression patterns of machine learning have long been pursued.

15.2 Category representation of learning expressions

Expression is not only one of the ways in which human beings communicate but also an important part of human activities. Human interaction is difficult to imagine if there is no interaction between human beings. According to the relation between cognitive science and machine learning, expression plays an equally important role in the study of machine learning, and the research of expressions is an important problem for machines.

An expression is the same type of data (such as constants, variables, and functions) connected in a meaningful way with some symbolic rules.

At present, the mature methods of expression mainly include the following: regular expressions, S-expressions, inverse Polish expressions, and Polish expressions. A regular expression is usually called a pattern. A string is used to describe or match a series of syntactic rules. The S-expression is an agreement to express semi-structured data in human readable text. The inverse Polish expression is a type of book order of expression introduced by Polish mathematician Jan Łukasiewicz in 1920. In the inverse Polish representation, all operators are placed behind operands, also known as postfix notation. In the Polish notation, operators are placed in front of operands, also known as prefix notation.

In the field of machine learning, the mapping mechanism between expressions is studied to abstract various data structures. Using category theory to study and generalise the links between structures makes it easier to solve the problem of learning, which is one of the main concerns of this paper.

Category theory is a mathematical theory that abstractly deals with the connection between mathematical structures. A category is a mathematical structure for grasping the nature of a class of mathematical objects (such as groups in group theory). The category theory maintains the stability of the learning system and the accuracy of the learning efficiency by using the constant morphism property of the object structure between the mathematical objects.

15.2.1 Category representation of machine learning systems

According to the framework of machine learning systems and the basic concepts of category theory, a definition of a machine learning category is given first.

Definition 15.9. Machine learning category \mathcal{C} consists of the following two parts:

- (1) Object $\text{ob}\mathcal{C}$;
- (2) Any pair of objects A, B , corresponds to a set $\mathcal{C}(A, B)$. Its element is called a morphism, and it satisfies the condition that if $A \neq A'$ or $B \neq B'$, $\mathcal{C}(A, B)$ and $\mathcal{C}(A', B')$ have no intersection.

Explanation:

- (1) the objects here correspond to various input/output sets in the learning problem;
- (2) all objects form a group of objects;
- (3) a morphism is an abstraction of the process of preserving the structure between two objects, which corresponds to the process abstraction between the input set and the output set;

Theorem 15.1 is given below.

Theorem 15.1. *Machine learning systems form a category.*

Proof. According to the composition of the learning system described in the first chapter, each input object and output object have one-to-one relations, and the input/output set of the whole machine learning system constitutes a family object $\text{ob}\mathcal{C}$. In $\text{ob}\mathcal{C}$, for any pair of objects A, B , we can find a set $\mathcal{C}(A, B)$ such that they form a morphism. For example, the object A is defined as a set of inputs, and the output is defined as a collection of objects B . In any system, from the input to the output, there is a path $\mathcal{C}(A, B)$. We can use this path as a morphism from object A to object B .

At the same time, in a machine learning system, input objects A are processed by one or more modules, and they can become the output objects B . In this process, the morphism between each pair of objects is disjoint. In other words, if $A \neq A'$ or $B \neq B'$, $\mathcal{C}(A, B)$ and $\mathcal{C}(A', B')$ are disjoint. This satisfies the definition of a machine learning category. Thus, machine learning systems can form a category. \square

Similarly, the dual category of machine learning can be defined. Let \mathcal{C} be a machine learning category, with the object as the object \mathcal{C} , and the direction of the morphism \mathcal{C} is a morphism, forming a new machine learning category. We call it the dual category of \mathcal{C} , denoted by as $\mathcal{C}^{\text{op}}(f \in \mathcal{C}^{\text{op}}(A, B) \text{ if and only if } f \in \mathcal{C}(B, A))$.

Let P be a proposition about machine learning category \mathcal{A} , i.e., the condition and conclusion of a proposition P are composed of objects and a morphism in the machine learning category \mathcal{A} . If all the states in proposition P are inverted, we can obtain a new proposition P^* called the dual proposition of proposition P . It is easy to see from proposition P that the machine learning category \mathcal{A} is established if and only if the dual proposition P^* is concerned with the machine learning category \mathcal{A}^{op} . Then, $(\mathcal{A}^{\text{op}})^{\text{op}} = \mathcal{A}$ is available to any machine learning category \mathcal{A} , and thus, the dual principle is obtained.

Example 15.1. The learning problem given in the last chapter is described by the category idea: the solution set, training sample set, objective function set, and the new problem set are the object families $\text{ob}\mathcal{C}$ in the category \mathcal{C} of the learning problem. The function of the four modules corresponds to the morphism between sets: the solution set is object A , the training sample set is object B , the object function set is object C , and the new problem set is object D . The execution system, identifier, generalisation device, and experiment generator are morphisms f, g, h , and k , with $g \in \mathcal{C}(A, B)$, $h \in \mathcal{C}(B, C)$, $k \in \mathcal{C}(C, D)$, and $f \in \mathcal{C}(D, A)$.

According to Theorem 15.1, the machine learning category has the following rules of operation:

- (a) Composition law: If $A, B, C \in \text{ob}\mathcal{C}$, $f \in \mathcal{C}(A, B)$, $g \in \mathcal{C}(B, C)$, then there is only one $gf \in \mathcal{C}(A, C)$, called the composite f and g ;
- (b) Associativity: If $A, B, C, D \in \text{ob}\mathcal{C}$, $f \in \mathcal{C}(A, B)$, $g \in \mathcal{C}(B, C)$, $h \in \mathcal{C}(C, D)$, then $h(gf) = (hg)f$;
- (c) Identity morphism: For each object A , there is a morphism $1_A \in \mathcal{C}(A, A)$ such that any $f \in \mathcal{C}(A, B)$ and $g \in \mathcal{C}(C, A)$ have $f1_A = f$ and $1_Ag = g$.

Example 15.2. Decision tree learning is used as an example to prove the feasibility of machine learning.

Decision tree learning is a method of approximating a discrete-valued objective function, which can be expressed as a decision tree. The classification of instances is done mainly by arranging instances from root nodes to a leaf node. The classical decision tree ID3 algorithm is analysed as an example.

A complete ID3 learning algorithm is adopted for the learning category, data sets, objects, and sample sets, as well as the obvious morphism between objects. We do not go into further detail. Only the core algorithm is demonstrated below.

The whole sample set is set as the object X of learning category \mathcal{C} . The sample set is composed of instances, and the instances are represented by attribute-value pairs. The whole set of attributes corresponding to the sample set is the object Q of the learning category \mathcal{C} , and the resulting decision tree is the object T of the learning category \mathcal{C} .

f_1 : determine the sample set of leaf nodes belonging to the same class $\in \text{Mor}\mathcal{C}$, $f_1 : X \rightarrow M$; f_2 : determine whether the node attribute is empty $\in \text{Mor}\mathcal{C}$, $f_2 : Q \rightarrow M$.

Calculate the entropy of each attribute in the training sample $g_1 : \text{Entropy}(S) \equiv \sum_{i=1}^c -p_i \log_2 p_i \in \text{Mor}\mathcal{C}$, E : The entropy set corresponding to the whole property set $\in \text{ob}\mathcal{C}$, $g_1 : Q \rightarrow E$.

Using the entropy set object obtained from the above states, we can obtain the corresponding information gain object by morphism $g_2 : \text{Gain}(S, A) \equiv \text{Entropy}(S) - \sum_{v \in \text{Value}(A)} |S_v| / |S| \text{Entropy}(S_v) \in \text{Mor}\mathcal{C}$, i.e., $g_2 : E \rightarrow G$, $G \in \text{ob}\mathcal{C}$ is the information gain object of each attribute.

Finally, the maximum information gain value of the corresponding element in the information gain object is selected, and the information value (attribute name, attribute value) of the corresponding element is recorded. According to morphism

$h: \max \sum_{i,j} \text{Gain}(X_i, Q_{ij}) \in \text{Mor}\mathcal{C}$, we can see that $h: G \rightarrow M$. The object M is an attribute set with the highest information gain. Thus, the next node can be obtained, and the decision tree object T is then updated. The final decision tree is obtained by cyclic computation until the leaf node is generated.

Therefore, the category representation method of the decision tree learning algorithm is obtained, as shown in Figure 15.4.

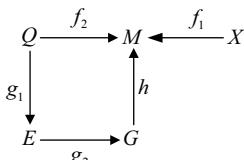


Fig. 15.4: Representation of decision tree algorithm based on category.

In summary, the ID3 algorithm has a family of objects and corresponding morphism, and satisfies three conditions. Thus, the ID3 algorithm is a machine learning category.

Bayes learning is used to verify the universality and practicability of applying category theory to machine learning.

Example 15.3. The Bayes learning algorithm can also be defined as a machine learning category.

Here, we take the classical algorithm of Bayesian learning, namely the EM algorithm, as a special case to prove that the Bayesian learning algorithm can be expressed as a machine learning category.

The EM algorithm is used in many problem frames and requires estimation of a family of parameters $\theta = \langle \mu_1, \mu_2, \dots, \mu_k \rangle$ describing the benchmark probability distribution. This can be regarded as an unknown parameter object U in the machine learning category \mathcal{C} . Given all the data that can be observed from this distribution, all data can be written as $k + 1$ tuples $\langle x_i, z_{i1}, z_{i2}, \dots, z_{ik} \rangle$, denoted by Y ; only x_i can be observed. $X = \langle x_1, \dots, x_m \rangle$ represents data that can be observed in the same instance. Then, $Y = X \cup Z$ represents all the data. The unobserved data object Z can be viewed as a random variable whose probability distribution depends on the unknown parameter object U and the known data object X .

Suppose that the current hypothesis h is the hypothesis set object H . The object H' represents the maximum likelihood hypothesis set, and the object Q is the posterior probability set. The following two steps correspond to the two morphisms in the machine learning category, which can describe the process of the EM algorithm:

- (1) Estimation step: use the current assumed object H and the observed data objects X to estimate the probability distribution on the object Y to calculate the object Q , and the calculation process can be represented by the morphism f .

$$f: HX \rightarrow Q$$

- (2) Maximisation step: replace the current hypothetical object H with the maximum likelihood assumption object H' that maximises the object Q . Use the morphism representation as

$$g: Q \rightarrow H'$$

Here, morphism f can be reduced specifically to $f: E[\ln P(Y|h'.)|h, X.]$, and morphism f' can also be represented by morphism $f': \int \log(L(\theta|X, y)) f(y|X, \theta'.) dy$. We can define that morphism g can be reduced to $g: \arg \max'_h Q$. Moreover, the object and the morphism of the machine learning domain satisfy the compound operation law, the combination law, and the unit morphism. Hence, the bias learning algorithm is also a machine learning category.

15.2.2 Category representation of learning expressions

First, we use categories to represent simple expressions, such as $y = x^2, x \in R$. The expression contains two sets: the defined domain and the range. Therefore, we can define two objects A, B in a category \mathcal{C} , where the domain is defined as A and the range is expressed as B , and the expression can be represented as a category:

$$f: A \mapsto B, \quad f \text{ is the corresponding morphism to } \mathcal{C}(A, B). \quad (15.1)$$

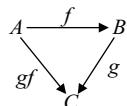
For expressions in learning problems, we can obtain them progressively from the standard description of the learning problem. In [1], T and the performance metrics for a class of tasks P have been defined. If a computer program on T uses P to measure its performance as the experience E , then the computer program is said to be learning from the experience E . We can simplify the learning task into an operable description of the discovery of an ideal objective function, which can be done in the form of a learning expression.

The object A is the initial state set of the training sample, the object B is the intermediate state, and the object M is the final output state. Then, the learning process can be expressed as

$$f: A \mapsto B, g: B \mapsto C. \quad (15.2)$$

f, g is the morphism of the corresponding objects. According to the complex operation law and the combination law of category,

$$gf: A \mapsto C \quad (15.3)$$



The following is a linear expression to complete the learning process, e.g., to illustrate the category representation of learning expressions.

$$f: W \times X \mapsto B \quad (15.4)$$

$$g: B \mapsto M \quad (15.5)$$

$$h: B \times M \mapsto W. \quad (15.6)$$

Explanation: By (15.1), (15.4), and (15.5), the initial weight object W and the initial state object X are mappings by morphism f to get the intermediate state object B , and the intermediate state object B gets the end state object M by morphism g . By (15.4), (15.5), and (15.6), the minimum error is used to recalculate the weight and update the weight object to complete a round of the learning process.

Linear expressions can be linear equations or linear inequalities, and the learning process using linear expressions is called linear expression learning. We can see from the literature [16] that maximisation or minimisation of a linear function by a linear equality or inequality is a type of linear programming technique. It is assumed that the object U is an unknown object, which can be obtained by minimising the state of the linear objective function f , which is an $m \times 1$ vector of known cost objects A , $f: A \rightarrow U$, namely $z_{\min} = \alpha^T \cdot u$, and $Bu \geq \beta$ holds, where object B consists of an $l \times m$ matrix and β is an $l \times 1$ vector. Hence, this linear programming can be expressed as a category \mathcal{A} of learning expressions. Figure 15.5 shows the transformations and relationships between the objects and the morphism. The surface of $z_{\min} = \alpha^T \cdot u$ is shown in grey. The surface of $Bu \geq \beta$ is shown in red. The unknown objects U are obtained by iterating the morphism f and the constraints.

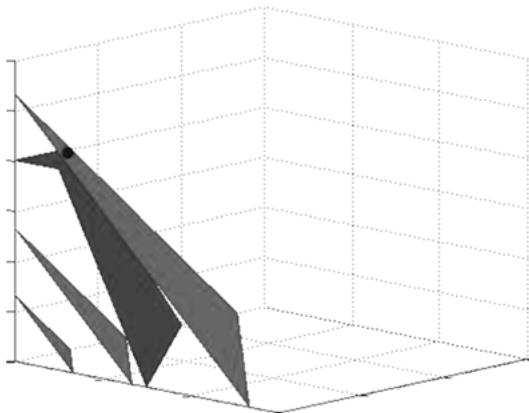


Fig. 15.5: Simple iteration of linear programming in the category of learning expressions.

15.2.3 Category Representation of the Learning Expression Functor

From the last section, we can see that the learning process can be expressed as an abstract algebraic structure of the category of learning expressions. Different learning processes belong to the same algebraic structure. Thus, there is a homomorphic relation between different learning processes, which forms the mapping mechanism between expressions, namely existence mapping $\varphi: \mathcal{C} \rightarrow \mathcal{D}$ satisfies $\varphi(u \cdot v) = \varphi(u) \cdot \varphi(v)$.

Definition 15.10. Suppose that \mathcal{C} and \mathcal{D} are a learning expression category. A learning expression functor $F: \mathcal{C} \rightarrow \mathcal{D}$ consists of two mappings:

$$\text{ob}\mathcal{C} \rightarrow \text{ob}\mathcal{D}: A \mapsto F(A) \quad (15.7)$$

$$\text{Mor}\mathcal{C} \rightarrow \text{Mor}\mathcal{D}: f \mapsto F(f). \quad (15.8)$$

Explanation:

- (1) represents the mapping between objects in two categories of learning expressions, mapping each object A that belongs to \mathcal{C} to another object B that belongs to \mathcal{D} , $B = F(A)$;
- (2) represents the mapping between morphisms in two learning expression categories, mapping each morphism f belonging to \mathcal{C} to another morphism g belongs to \mathcal{D} , which can be represented as $g = F(f)$, and satisfies $F(gf) = F(g)F(f)$.

\mathcal{C}, \mathcal{D} are learning expressions, and $F: \mathcal{C} \rightarrow \mathcal{D}$ is a learning expression functor. For any pair of objects A, B in \mathcal{C} and $f, g \in \mathcal{C}(A, B)$, we have

$$F(f) = F(g) \Rightarrow f = g$$

i.e., F locates in $\mathcal{C}(A, B)$. If $\mathcal{C}(A, B) \rightarrow \mathcal{D}(F(A), F(B))$ is a single shot, then learning expression functor F is a faithful learning expression functor.

If for any $g \in \mathcal{D}(F(A), F(B))$, there is $f \in \mathcal{C}(A, B)$ satisfying $F(f) = g$, i.e., F locates in $\mathcal{C}(A, B)$. Further, if $\mathcal{C}(A, B) \rightarrow \mathcal{D}(F(A), F(B))$ is a full shot, then learning expression functor F is a full learning expression functor.

Theorem 15.2. *There are functors between learning expression categories.*

Proof. Let \mathcal{C}, \mathcal{D} be learning expression categories. $W, X, B, M \in \text{ob}\mathcal{C}$, $W', X', B', M' \in \text{ob}\mathcal{D}$, $f, g, h \in \text{Mor}\mathcal{C}$, $f', g', h' \in \text{Mor}\mathcal{D}$. The objects in the category of learning expressions are determined by the set of training samples, and condition (15.7) is obvious:

$$(W, X, B, M) \mapsto F(W', X', B', M').$$

Assume that \cdot is the operator between morphisms, and mapping $F: X \rightarrow Y$ satisfies $F(u \cdot v) = F(u) \cdot F(v)$.

$$\begin{aligned} F(f \cdot f') &= F((W \times X \mapsto B) \cdot (W' \times X' \mapsto B')) \\ &= F((W \times X \rightarrow B) \cdot (F(W) \times F(X) \rightarrow F(B))) \\ &= F(W \times X \rightarrow B) \cdot F(F(W) \times F(X) \rightarrow F(B)) \\ &= F(f) \cdot F(f') \end{aligned}$$

Thus, learning expressions satisfy morphism mappings.

In summary, there are learning expression functors between two learning expression categories, i.e., there are mappings in category theory between learning expression categories. This provides a theoretical basis for further research on the mapping mechanism between learning expression categories. \square

15.2.4 Natural transformation

As can be seen from the previous section, the expression of the functor is studied to investigate the relationship between expressions of category learning. However, there exists relations between categories and between functors. Therefore, this section mainly introduces the study of relations between the expressions of functors, called natural transformation.

Definition 15.11. \mathcal{C} and \mathcal{D} are learning expression categories, and $F: \mathcal{C} \rightarrow \mathcal{D}$ and $G: \mathcal{C} \rightarrow \mathcal{D}$ are two functors corresponding to the learning expression categories. A natural transformation $\alpha: F \rightarrow G$ is a mapping $\text{ob}\mathcal{C} \rightarrow \text{Mor}\mathcal{D}$:

$$A \mapsto (\alpha_A: F(A) \rightarrow G(A)), A \in \text{ob}\mathcal{C},$$

such that for any morphism $f: A \rightarrow B$ in \mathcal{C} , we have $G(f)\alpha_A = \alpha_B F(f)$, i.e., the following icon exchange:

$$\begin{array}{ccc} F(A) & \xrightarrow{\alpha_A} & G(A) \\ F(f) \downarrow & & \downarrow G(f) \\ F(B) & \xrightarrow{\alpha_B} & G(B) \end{array}$$

Example 15.4. The classifier based on Bayesian decision theory obtains more forms and more methods through natural transformation, so that the classifier can be optimised.

By the definition of machine learning category, we can see that the classifier in the category is category \mathcal{C} . Data sets, feature vectors, prior probabilities, posterior prob-

abilities, and discriminant regions are all included in $\text{ob}\mathcal{C}$. Discriminant functions, eigenvectors, probability formulae, and so on are all included in $\text{Mor}\mathcal{C}$. Two classifiers constitute a special case of the multi-class case, which can be defined as category \mathcal{D} , so $\text{ob}\mathcal{C} \rightarrow \text{ob}\mathcal{D}$. Objects in \mathcal{C} can correspond to objects in \mathcal{D} , $\text{Mor}\mathcal{C} \rightarrow \text{Mor}\mathcal{D}$, and a morphism in \mathcal{C} can correspond to a morphism in \mathcal{D} .

$F: \mathcal{C} \rightarrow \mathcal{D}$, $A, B \in \text{ob}\mathcal{C}$, then $F(A), F(B) \in \text{ob}\mathcal{D}$. Assume that object A is a data set a_1 and object B is a feature vector. Similarly, $G: \mathcal{C} \rightarrow \mathcal{D}$, $A, B \in \text{ob}\mathcal{C}$, then $G(A), G(B) \in \text{ob}\mathcal{D}$. The object A is a data set a_2 , and object B is a feature vector.

By the definition of functor, for the functor F , $A \mapsto F(A)$ and functor G , $A \mapsto G(A)$, there is a certain connection between data set objects A_F and A_G . They all have the same law, i.e., they can produce many categories. Hence, there is a configuration shot that makes $A_F \rightarrow A_G$. By this reasoning, we obtain $F(A)$ and $G(A)$ for which there exists a morphism defined as $\alpha_A: F(A) \rightarrow G(A)$. Similarly, $\alpha_B: F(B) \rightarrow G(B)$.

The objects A, B in \mathcal{C} have a morphism, denoted by $f: A \rightarrow B$, whose process and function are mainly to solve the feature vectors of data sets. By the definition of the functor $f \mapsto F(f)$, we can obtain $F(f): F(A) \rightarrow F(B)$.

Similarly, $G(f): G(A) \rightarrow G(B)$.

In summary, the two classifiers based on Bayesian decision theory can use the natural transformation. It is proven that the natural transformation theory has some applications in the learning problem.

15.3 Mapping mechanism for learning expressions

In a learning system, the representation of any learning problem is expressed by a specific expression. The nature of the learning system is also determined by the effect of the expression. For a long time, this problem has been called the bottleneck problem of the system. According to the discussion in the previous section, the learning system can be represented by a category. How can the relation between the learning systems be expressed in the category of expressions? This section focuses on answering this question.

15.3.1 Abstract concept of an expression

The representation and application of the expression is the focus of this article. For convenience, we first need to find the commonalities of expressions. The original aim of category theory is to abstract many concrete mathematical objects and mappings with similar characteristics in different mathematical fields into unified concepts. This abstraction is achieved by morphism. For example, a set T is a single point set that can

be equivalently represented as any set. From the viewpoint of category theory, there happens to be a mapping to set T . Therefore, morphism plays a major role in category theory. In fact, the research and application of learning expressions is nothing more than the relation of various mappings, and the theory of morphism can better reflect the relation of each part.

As a hypothesis, we consider a morphism in this learning expression. If any pair of parallel states in a pair is a morphism, then there is a left reducible, which is called a single morphism (monomorphism) of the category of the learning expression. Assume that \mathcal{C} is a learning expression category, and $f: A \rightarrow B$ is a morphism in this learning expression. If any pair of parallel morphisms $g, h: C \rightarrow A$ in \mathcal{C} satisfies $fg = fh$, we have $g = h$ (now called f left reducible), where f is called a monomorphism of the learning expression category.

Similarly, it can be assumed that f is an epimorphism if and only if any pair of parallel morphism $g, h: B \rightarrow C$ satisfies $fg = fh$, and we have $g = h$ (now called f right reducible).

The mapping principle of expressions in a linear discriminant function can be studied by means of the monomorphism and epimorphism of learning expressions.

Example 15.5. In linear learning category \mathcal{C} , there exists an epimorphism of the learning expression.

Proof. In the linear learning category \mathcal{C} , the object X represents the data set W that needs to be discriminated as the weight vector object. The morphism $f: X \cdot W \rightarrow B$ is a mapping of the data set object and the weight vector object to the discriminant function value object, denoted by $f: B(x) = W \cdot X + w_0$, where w_0 represents a threshold weight or bias for a variable value. The morphism $g: B \rightarrow R$ represents the mapping of the arithmetic distance process of the point distance from the hyperplane of the data set object. The object R is an arithmetic distance object, which can distinguish the classification of the element points in X and the distance from the hyperplane. Morphism g can be represented by an expression $X = X_p + RW/\|W\|$ for its mapping process.

There is parallel morphism g and morphism $h: B \rightarrow R$ at the same time, and the mapping process can be expressed by $R = B(x)/\|W\|l$. For morphism f, g, h , $gf = hf$ can be obtained by the compound operation rate of category theory. The specific process is shown in Figure 15.6.

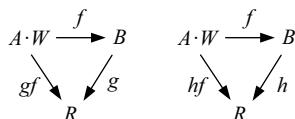


Fig. 15.6: Compound operation between morphism.

Since $g: X = X_p + RW/\|W\|$, i.e., $R = \|W\|/W(X - X_p)$ and $h: R = B(x)/\|W\|$, then

$$R = \frac{W \cdot X + w_0}{\|W\|} = \frac{W}{\|W\|}X + \frac{w_0}{\|W\|}.$$

Although the mapping process of morphism g, h is different, the result and meaning of the mapping are quite similar; then, $g = h$. Hence, it is possible to have the full morphism of the learning expression in the linear learning category \mathcal{C} . \square

Before studying the expression of the functor, we must first understand the following assumptions and theorems.

\mathcal{C} is a learning expression category, $A \in \text{ob}\mathcal{C}$; define a functor $\mathcal{C}(A, -): \mathcal{C} \rightarrow \text{Set}$ as follows:

$$B \mapsto \mathcal{C}(A, B), (f: B \rightarrow C) \mapsto (c(A, f): g \mapsto fg).$$

We call $\mathcal{C}(A, -)$ the hom-functor on A . We can define the duality functor $\mathcal{C}(A, -): \mathcal{C}^{\text{op}} \rightarrow \text{Set}$. Under antithesis, we can define the contravariant hom-functor on A .

Theorem 15.3. (*Yoneda lemma*) \mathcal{C} is a category of learning expressions $A \in \text{ob}\mathcal{C}$, $F: \mathcal{C} \rightarrow \text{Set}$ is a functor, and $\text{Nat}(\mathcal{C}(A, -), F)$ denotes all the natural transformations between functor $\mathcal{C}(A, -)$ and functor F . Then, there is a mapping from $\text{Nat}(\mathcal{C}(A, -), F)$ to $F(A)$.

Definition 15.12. \mathcal{C} is a category of learning expressions, $F: \mathcal{C} \rightarrow \text{Set}$ is a functor. If there is an object A in \mathcal{C} and a natural isomorphism $\phi: \mathcal{C}(A, -) \rightarrow F$, then we call ordered pair (A, ϕ) as a representation of F . If there is a representation of F , F is a representable functor.

In Definition 15.12, a representation of F can be considered as an ordered pair $(A, x)(A \in \text{ob}\mathcal{C}, x \in F(A))$ (such that the natural transformation corresponding to x is a natural isomorphism, i.e., for any object B and each $y \in F(B)$, there exists a unique morphism $f: A \rightarrow B$ satisfy $F(f)(x) = y$). Thus, we can obtain an equivalent description of F as follows:

For all the ordered pairs (A, x) as objects and when morphism $f: (A, x) \rightarrow (B, y)$ is the morphism $f: A \rightarrow B$ such that $F(f)(x) = y$, it is obvious that this can constitute a category of learning expressions, and that a representation of F is exactly an initial object in that category.

Example 15.6. Oblivion functor $U: \text{Gp} \rightarrow \text{Set}$ from Lie group machine learning category Gp to the set category Set is a representable functor.

Proof. According to the definition of the functor, assume that functor U exists. Then, each Lie group machine learning corresponds to the set, and each group homomorphism (continuous mapping, ring homomorphism, etc.) corresponds to this functor U .

As a mapping between the category of topological spaces Top and set category Set, there is a mapping between the objects. There is a corresponding mapping between morphisms. Thus, the oblivion functor exists.

We can prove that in this functor, there is a one-to-one mapping. $A \in \text{obGp}$, $\text{Nat}(\text{Gp}(A, -), U)$ is the all nature transformation between functor $\text{Gp}(A, -)$ and oblivion functor U . For each natural transformation $\alpha: \text{Gp}(A, -) \rightarrow U$, define $\Theta(\alpha) = \alpha_A(1_A) \in U(A)$. For $x \in U(A)$, define $\Phi(x): \text{Gp}(A, -) \rightarrow U$. Then, $\Phi(x)_B(f) = U(f)(x) \in U(B)$. Obviously, $\Phi(x)$ is a natural transformation. Hence, there is a mapping $\Phi: U(A) \rightarrow \text{Nat}(\text{Gp}(A, -), U)$. In summary, Θ and Φ are inverses of each other, such that there is a one-to-one mapping from $\text{Nat}(\text{Gp}(A, -), U)$ to $U(A)$, i.e., there are natural isomorphisms $\phi: \text{Gp}(A, -) \rightarrow U$.

According to the definition of representable functors, an ordered pair (A, ϕ) is a representation for U . Hence, it is a learning expression representable functor, namely oblivion functor $U: \text{Gp} \rightarrow \text{Set}$ from Lie group machine learning category Gp to the set category Set is a representable functor. \square

15.3.2 Mapping mechanism between expressions

The previous section described the basic concepts and abstract representations of learning expressions, which have provided a clear theoretical basis for studying the mapping mechanism between expressions. In this section, the mapping mechanism between learning expressions is verified by a specific example, and the feasibility of the mapping theory is proved. First, we select the specific problems in the learning problem and analyse their existence in the category theory. Then, the existing learning problems are improved by using the mapping mechanism of learning expressions, and better results are obtained. A practical application of abstract theory is demonstrated, and the correctness and feasibility of the theory are proven.

The decision tree category has been briefly introduced in the second section. This section gives the category proof of the decision tree learning algorithm.

A decision tree classifies instances by arranging instances from the root nodes (sort) to a leaf node. A leaf node is a classification of instances. Each node on the tree specifies a test of an attribute of an instance, and each succeeding branch of that node corresponds to a possible value of the property. The taxonomy example starts with the root node of the tree, tests the attributes specified by this node, and then moves down the corresponding branches according to the property value of the given instance. The process is then repeated on a subtree with a new node as the root.

The decision tree classification algorithm originated from the concept of learning systems. Then, through a series of developments, we have obtained the ID3 and C4.5 algorithms as well as a wide range of applications. Here, we mainly discuss the ID3 algorithm.

In 1986, Quinlan proposed the ID3 algorithm [17], which is also a representative of the decision tree algorithm. It is the earliest and most influential decision tree algorithm. Its advantages include simple description and fast classification. Its basic theory is relatively clear, its learning ability is strong, and the average depth of the decision tree is small. The ID3 algorithm is summarised in Table 15.2.

Tab. 15.2: Summary of the ID3 algorithm

Input: $E = D_1 \times D_2 \times \cdots \times D_n$

Output: Decision Tree

- (1) $E = D_1 \times D_2 \times \cdots \times D_n$ is a finite vector space, D_j is a finite discrete symbolic set, elements $e = < v_1, v_2, \dots, v_n >$ in E are called examples, $v_j \in D_j, j = 1, 2, \dots, n$. PE and NE are the two subsets of set E , called the positive set and counterexample set.
- (2) The sizes of PE and NE in vector space E are p and n . The following two assumptions need to be fulfilled: (1) the correct decision tree in vector space E is consistent with the probability of the classification of any instance and the positive counterexample; (2) the amount of information needed for an example that the decision tree can determine correctly is given by

$$\begin{aligned} \text{Entropy}(S) &= -p_+ \log_2 p_+ - p_- \log_2 p_- \\ &= -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n} \end{aligned}$$

- (3) Now, with attributes A as the decision tree root, A has m values $\{v_1, v_2, \dots, v_m\}$. It divides E into m subsets $\{E_1, E_2, \dots, E_m\}$. The desired information for property A to the root is

$$\text{Entropy}(S, A) = \sum_{i=1}^m \frac{p_i + n_i}{p+n} \cdot \text{Entropy}(S_i)$$

p_i and n_i are the positive and negative instance values of E_i , and $\text{Entropy}(S_i)$ is the desired information of subset E_i .

- (4) Calculate the information gain of root A . The formula is

$$\text{Gain}(S, A) = \text{Entropy}(S) - \text{Entropy}(S_i, A)$$

- (5) Select attribute A^* with the maximum $\text{Gain}(S, A)$, minimum $\text{Entropy}(S_i, A)$, as the root. For v subset E_i of E according to different values of A^* , the child nodes B_1, B_2, \dots, B_v of A^* are generated by recursively calling the above procedures 1–4.
-

After determining the process of decision tree learning and the method of constructing a decision tree, we continue to understand Bayesian learning. Bias reasoning provides a probabilistic means of reasoning but requires certain conditions, i.e., the quantity to be inspected follows a probability distribution. We can make inferences based on these probabilities and the observed data to make optimal decisions. Bias reasoning is also important for machine learning because it provides a metric for measuring the confidence of multiple hypotheses. Secondly, Bayesian reasoning provides the basis for the learning algorithm of direct operation probability and provides a theoretical framework for the analysis of other algorithms.

- The characteristics of bias learning methods can be summarised in the following:
- (1) Each training example observed can incrementally reduce or increase the estimated probability of a hypothesis. This provides a more reasonable way of learning than other algorithms. Other algorithms will completely remove the hypothesis when a hypothesis does not coincide with any of the examples.
 - (2) Prior knowledge can determine the final probability of the hypothesis with the observation data. In Bayesian learning, the form of prior knowledge can be as follows: (a) a priori probability of each candidate hypothesis; (b) each probability distribution that may be assumed on observable data.
 - (3) The bias method allows assumptions to be made to predict uncertainty.
 - (4) The new instance can be predicted by multiple assumptions and weighted by their probabilities.
 - (5) Even when the computational complexity of Bayesian methods is high, they can still be used as a criterion for optimal decision making.

The bias theory provides a direct way to calculate this possibility. More precisely, the Bayes rule provides a way to calculate hypothetical probabilities. It is based on the assumed prior probability, the probability of the different data observed under the given hypothesis, and the observed data itself.

The bias rule is the basis of the bias method of learning because it provides a method from prior probabilities $P(h)$, $P(D|h)$, and $P(D)$ to compute posterior probabilities $P(h|D)$. Table 15.3 lists the basic formulas for computing the probability.

Tab. 15.3: Formula of basic probability

- Product rule: The probability of intersection $P(A \wedge B)$ between two events A and B

$$P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A)$$

- Sum rule: The probability of union $P(A \vee B)$ between two events A and B .

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

- Bayes theorem: The posteriori probability $P(h|D)$ of h when D is given.

$$P(h|D) = P(D|h)P(h)/P(D)$$

- Theorem of total probability: If the events A_1, \dots, A_n are mutually exclusive and $\sum_{i=1}^n P(A_i) = 1$, then

$$P(B) = \sum_{i=1}^n P(B|A_i)P(A_i)$$

From the previous two cases, we study the theories of decision tree and bias. It can be found that the two theories have something in common. However, how to combine the two theories into one theory and make the learning algorithm more practical and

efficient remains the focus of our research. The mapping principle of learning expressions provides a strong theoretical basis for this process. Therefore, we can apply the mapping mechanism of learning expressions to this case.

Using the mapping mechanism between the categories of learning expressions, we can extend the classification range of decision tree learning. We allow sample instances to have unknown property characteristics. Using Bayesian theory, a priori and a posteriori probabilities, and the maximisation hypothesis, are employed to partition the instance space. However, training sample instances require all of the data.

First, the decision tree is constructed using the classical decision tree learning algorithm and by computing the information entropy and the information gain. Because of the independent identical distribution of the sample data attributes and their values, according to the decision tree constructed by the lack of data set of attributes, we can calculate prior, posterior probability, and estimate the related items. Then, we get the maximum value of the attribute extraction to finally perform data classification.

The classification algorithm can be divided into two steps. First, the decision tree is constructed, and then the decision tree is used to classify the data. The scope of the specific algorithm is expressed as follows:

(1) Constructing a decision tree.

Specific construction methods and detailed procedures are described in the second chapter.

(2) Using the decision tree to classify data.

First, we define objects, object attributes $Q = \theta \cap W$ in the test sample, in which θ is known as the unknown attribute set, and W attribute set. The probability and posterior probability of object P contains a priori object for maximum likelihood estimation F , the data object A with maximum likelihood estimation of the attribute value, and the classification results R of the image.

The posterior probability of the unknown attribute is obtained by the prior probability Q and the known probability P in the attribute object and the probability object, and the morphism is formed as $p = P(\theta_i|w) = P(w|\theta_i) \cdot P(\theta_i) / \sum_j P(w|\theta_{ij}) \cdot P(\theta_{ij})$, where j denotes the attribute values for each attribute. Then, objects F are obtained by morphism $f = f_{\theta|w}(\theta_i|w) = f_{\theta,w}(\theta_i|w)/f_w(w)$. By using the maximum value method, objects A are easily obtained from objects F , and objects Q can be updated by attribute objects A , so that the result objects can be updated using the decision tree to classify data. The specific mapping process is shown in Figure 15.7.

Next, we create the decision tree using the data in Table 15.4 as the training data.

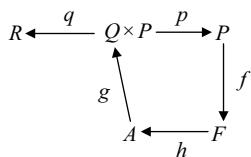


Fig. 15.7: Categorical data of category representation.

Tab. 15.4: Sample data set.

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

All attribute set objects Q can be defined as the set of data in Table 15.4 above, and the elements in object E can be obtained from the entropy morphism: $g_1 = \text{Entropy}(S) = \sum_{i=1}^c -p_i \log_2 p_i: Q \rightarrow E$

$$E \supset \left\{ \begin{array}{l} E(PT)=0.940, E(S_{\text{weak}}|\text{wind})=0.811, E(S_{\text{Strang}}|\text{wind})=1.00, E(\text{wind})=0.892, \\ E(S_{\text{sunny}}|\text{out})=0.971, E(S_{\text{overcast}}|\text{out})=0, E(S_{\text{rain}}|\text{out})=0.971, E(\text{out})=0.693, \\ E(S_{\text{cool}}|\text{temp})=0.811, E(S_{\text{mild}}|\text{temp})=0.918, E(S_{\text{hot}}|\text{temp})=1.00, E(\text{temp})=0.911, \\ E(S_{\text{high}}|\text{humidity})=0.985, E(S_{\text{normal}}|\text{humidity})=0.591, E(\text{humidity})=0.789 \end{array} \right\}$$

The following are obtained by information gain state projection: $g_2 = \text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Value}(A)} |S_v|/|S| \cdot \text{Entropy}(S_v): E \rightarrow G, G \supset \{G(S, \text{out}) = 0.246, G(S, \text{temp}) = 0.029, G(S, \text{humidity}) = 0.151, G(S, \text{wind}) = 0.048\}; h = \max \sum_{i,j} \text{Gain}(X_i, Q_{ij}): G \rightarrow M$. Then, we get the root node $M \supset \{\text{out(sunny, overcast, rain)}\}$.

Reselect the three attribute entries in the object Q , respectively, choose out(sunny) and out(rain) as a branch, repeat the process to get the other branches and leaf nodes, and get the final decision tree, as shown in Figure 15.8.

The sample data in Table 15.4 is randomly changed so that the value of the property item assumes a random missing state, and the newly generated data set is used as the test set. Now take the eighth data, e.g., the outlook attribute is missing; we can use it to examine the classification of data.

Test sample data property object $Q \supset [\emptyset, \text{Mild}, \text{High}, \text{Weak}, ?]$. The prior probabilities and other probability objects of the missing items obtained from Tables 15.4 and Figure 15.8 are as follows:

$$P \supset \left[\begin{array}{l} p(\text{Sunny}) = \frac{1}{3}, p(\text{Overcast}) = \frac{1}{3}, p(\text{Rain}) = \frac{1}{3}, \\ p(N|\text{Sunny}) = 0.60, p(N|\text{Overcast}) = 0, p(N|\text{Rain}) = 0.40, \\ p(Y|\text{Sunny}) = 0.40, p(Y|\text{Overcast}) = 1.0, p(Y|\text{Rain}) = 0.60 \end{array} \right]$$

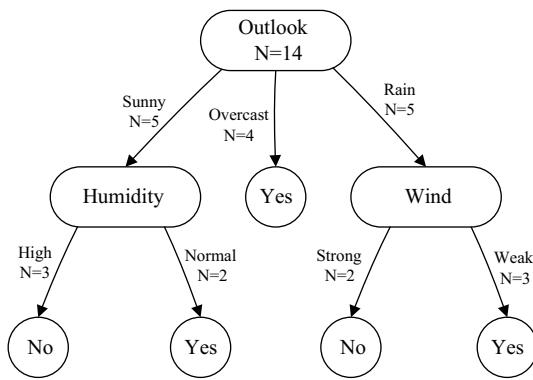


Fig. 15.8: Decision tree constructed from data in Table 15.4.

The posterior probability of the corresponding attribute value is computed by the morphism p , and the probability object P is updated:

$$P \supset \begin{bmatrix} p(\text{Sunny}|N) = 0.60, & p(\text{Sunny}|Y) = 0.20, & p(\text{Overcast}|N) = 0, \\ p(\text{Overcast}|Y) = 1.0, & p(\text{Rain}|N) = 0.40, & p(\text{Rain}|Y) = 0.30 \end{bmatrix}$$

The posterior probability in the selected object P is found in the object A by morphism f , and the object A is

$$A \supset [p(\text{Sunny}|N) = 0.60, \text{outlook}(\text{Sunny})]$$

The morphism k update test samples, the data property objects Q , and the original data is updated to $Q \supset [\text{Sunny}, \text{Mild}, \text{High}, \text{Weak}, ?]$. Finally, the decision tree T is used to classify the data.

The correct rate was 85.714%. Although the accuracy is not sufficiently ideal, it can still solve the classification when a small amount of data in the data set is missing.

15.4 Classifier design for learning expression mapping mechanism

This section presents the application and extension of the learning expression mapping mechanism in machine learning algorithms. By taking the classifier based on the learning expression mapping mechanism as an example, the application of the learning expression mapping mechanism is introduced in detail. Through the application analysis of the classifier, we not only verify the correctness of the theory of machine learning category but also prove the feasibility of the theory with practical experimental results.

15.4.1 Classifier algorithm

The design goal of the classifier is to automatically partition the data into known categories after learning. It is mainly used in search engines and various retrieval programs, and it is also widely used in the fields of data analysis and prediction. The classifier is based on a mathematical model. For different mathematical models, various classifiers can be generated, such as Bayes network classifier, decision tree classification, and support vector machine classification.

To study classifier algorithms, we need to understand classification and the data classification process. We can then study the classifier algorithm further and use the learning expression mapping mechanism for summary and analysis.

In [19], the data classification process is divided into two steps. The first step is to establish a classifier that describes the predefined data class or concept set. This process is called the ‘learning step’ or ‘training phase’. The classification algorithm constructs the classifier by analysing or learning from the training set. The training set consists of database tuples and their associated class labels. A learning model or classifier is provided in the form of a classification rule. The first step in the classification process can also be viewed as learning a mapping or function $y = f(X)$ that predicts the associated class label y of a given tuple X .

The second step is to use the model for classification. First, the prediction accuracy of the classifier is evaluated. If the training set is used to measure the accuracy of the classifier, the evaluation may be optimistic. Because the classifier tends to over fit the data, it is necessary to use the test set consisting of test tuples and associated class labels to test the accuracy of the classification rules. If the accuracy is acceptable, the rules are used for the new data tuple classification.

The system plays the role of a classifier based on a feature vector extractor, and an object to be measured is assigned a label. In general, perfect classification performance is impossible to obtain, and a more general task is to determine the probability of each possible category. The abstraction provided by the input data feature vector, makes it possible to establish the large-scale domain-independent classification theory.

In the classifier, the classification of the degree of difficulty depends primarily on two factors. The first is the difference between individual characteristics from the same categories of fluctuation. The second is the difference between the characteristics of samples belonging to different classes of values. In practice, this is often the case, and it is usually impossible to try to determine all of the eigenvalues from an input.

Through the analysis and research of the two steps of the data classification process, and based on the study of classifiers in [18], this paper briefly introduces the recognition process of classifiers in the process of image classification.

In the first stage of training, perform singular value decomposition of X_{ji} , i.e., the i -th training example sample of the j -th class. Get the k maximum singular value matrix $A_{ji}(m) = u_{ji}^m(v_{ji}^m)^H$ ($m = 1, 2, \dots, k$) corresponding to the image. A set of base

images obtained from training samples X_{ji} form the discriminant function $g_{ji}(X)$ of the j -th class:

$$g_{ji}(X) = \sqrt{\sum_{m=1}^k \alpha_{ji}^2(m)} = \sqrt{\sum_{m=1}^k |\langle A_{ji}(m), X \rangle|^2}, \quad j = 1, 2, \dots, C, \quad i = 1, 2, \dots, N$$

$\alpha_{ji}(m) = \langle A_{ji}(m), X \rangle$ is the m -th projection value from the sample image X to be recognised to X_{ji} .

Then, in the process of classifier recognition, the image X to be recognised is put into each discriminant function $g_{ji}(X)$, and the output of the discriminant function is obtained. Put X into the category corresponding to the maximum output of $g_{ji}(X)$. If each class has more than one training sample, the maximum length h is calculated for each class, i.e.,

$$h = \max_i(g_{ji}(X)), \quad j = 1, 2, \dots, C, \quad i = 1, 2, \dots, N$$

Thus, X can be subsumed into the c category.

The specific classification algorithm is as follows:

Input x_i , $i = 1, 2, \dots, m$, x_i is a data set of m dimensions, which can be divided into two categories, category 1 and category 2. If $y_i = 1$, it belongs to category 1; if $y_i = -1$, it belongs to category 2. If the data set is linearly separable, then we can find a classification function, i.e., the interval hyperplane: $D(x) = w^t x + b$, where w is a weight vector of m dimensions and b is a scalar:

$$y_i D(x_i) \geq 1, \quad i = 1, 2, \dots, m. \quad (15.9)$$

The distance between spaced hyperplanes is $D(x) = 0$, and the nearest training baseline from the hyperplane can be called the edge. When $D(x) = 0$, the hyperplane with the largest edge can be called the optimal interval hyperplane.

Now, consider the factors that determine the optimal spacing hyperplane. The Euclidean distance from the training baseline x to the classified hyperplane can be obtained by $D(x)/\|w\|$. Therefore, assuming that the edges are δ , all the training examples should satisfy the formula

$$\frac{y_k D(x_k)}{\|w\|} \geq \delta, \quad k = 1, \dots, m. \quad (15.10)$$

If w is a solution, then aw is also a solution, where a is a scalar. Therefore, we can use the following constraints:

$$\delta \|w\| = 1. \quad (15.11)$$

To find the optimal interval hyperplane, obtained from (15.10) and (15.11), we must find w of the minimum Euclidean criterion (15.9). Then, we can call the data that satisfies (15.9) a support vector. The optimal interval hyperplane can now be obtained by minimising

$$\frac{1}{2} \|w\|^2 \quad (15.12)$$

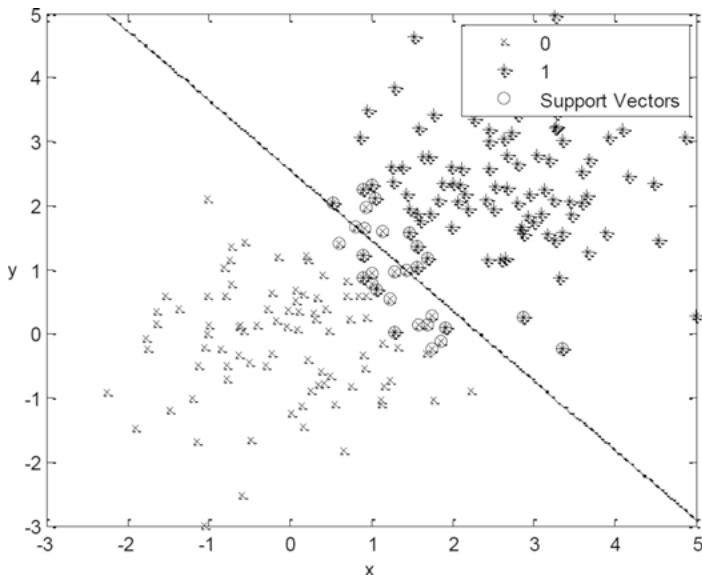


Fig. 15.9: Hyperplane.

where w and b satisfy the following constraints

$$y_i(w^t x_i + b) \geq 1, \quad i = 1, 2, \dots, m. \quad (15.13)$$

Figure 15.9 shows the idea of the optimal interval hyperplane for linear classification. There are four support vectors in the graph, which indicate that they are solid points.

This is a typical linear constrained convex programming problem, which uniquely determines the maximum margin hyperplane. The Lagrange function is as follows:

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^l \alpha_i (y_i(w \cdot x_i + b) - 1). \quad (15.14)$$

$\alpha_i \geq 0$ is the Lagrange multiplier corresponding to each sample. Minimise $L(w, b, \alpha)$ about w, b . According to the extremum condition $\nabla_b L(w, b, \alpha) = 0$ and $\nabla_w L(w, b, \alpha) = 0$:

$$\sum_{i=1}^l y_i \alpha_i = 0 \quad (15.15)$$

$$w = \sum_{i=1}^l \alpha_i y_i x_i. \quad (15.16)$$

Here, (15.15) and (15.16) are brought into the Lagrange function $L(w, b, \alpha)$, which can be obtained by the transformation:

$$L(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y_i y_j \alpha_i \alpha_j (x_i \cdot x_j). \quad (15.17)$$

The constraint is

$$\begin{cases} \sum_{i=1}^l y_i \alpha_i = 0 \\ \alpha_i \geq 0, \quad i = 1, \dots, l \end{cases} \quad (15.18)$$

Thus, we change the optimisation problem into a maximisation problem, and seeking maximisation of $L(\alpha)$ is the solution.

15.4.2 Classifier based on the learning expression mapping mechanism

According to the analysis and research in the machine learning domain and the learning expression mapping mechanism, it is easy to see that the classifier algorithm is also a category of machine learning.

The machine learning category description of the classifier is given. The input space is assumed to be the object X , $X \subset \mathbb{R}^n$. The sample set object $S \in X$, $S = \{(x_1, y_1), \dots, (x_t, y_t)\} \subset X \times \{-1, +1\}$, $x_i \in X \subset \mathbb{R}^n$, y_i is the target value of x_i , and y_i is the element in object Y . We can get q map $f_1, f_2, \dots, f_q \in f$ from the sample set object S . f is the map from X to Y , and $f: X \rightarrow Y$ is the reductive morphism. However, we need to find a morphism g that relates f_1, f_2, \dots, f_q with f , satisfying $f(x) = g(f_1(x), f_2(x), \dots, f_q(x))$. At this point, the morphism f is the final predicted morphism. Therefore, the classifier algorithm is defined as classifier category algorithm \mathcal{C} .

Through the description of classifier algorithm \mathcal{C} , we can see that the correlation and difference between morphism f_1, f_2, \dots, f_q and morphism f are the focus of this algorithm. Therefore, to improve the accuracy and universality of the algorithm, we should focus on the process of shooting to increase the diversity of the morphism f_1, f_2, \dots, f_q , which can reduce the error.

For classifier category \mathcal{C} , the general algorithm process is given below.

Algorithm 15.1: Classifier category algorithm

Input: input space object $X \subset \mathbb{R}^n$;

Learning sample set object $S = \{x_i, y_i\}_{i=1}^l, x_i \in X \subset \mathbb{R}^n$;

learning algorithm \mathcal{L} , Number of samples extracted by cycle q ;

Output: Decision morphism of classifier category f ;

- (1) Using pullback sampling, select l elements from the sample set object S to form a new sample data set object $S_i, S_i \subset S$;
 - (2) The training sample object S_i is trained by learning algorithm \mathcal{L} to get morphism f_i . The morphism is a predictive morphism, and the mapping is formed from object X to object Y ;
 - (3) If i is smaller than q , then go to (1), else continue;
 - (4) Some strategies are used to find the relation morphism g . Then, the decision morphism f of the classifier category is obtained.
-

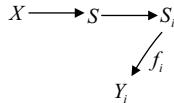


Fig. 15.10: Classifier category \mathcal{C} .

The classifier category algorithm is obtained by the above algorithm, as shown in Figure 15.10.

For the learning algorithm \mathcal{L} of Algorithm 15.1, we can use linear and nonlinear methods. When using a linear method, we can get the mapping l of sample object S to weighted object W by using the Lagrange morphism.

$l: S \rightarrow W$. This morphism can be obtained by maximising

$$L(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) .$$

For hyperplane morphism $k: X \cdot W \rightarrow D$, the morphism process can be represented as $w^t x + b = 0$, where D is a hyperplane object. At the same time, the maximum edge distance object M is obtained. Thus, morphism d can be mapped to objects Y , and the process can be accomplished by the inequality $y_k D(x_k)/\|w\| \geq \delta$. Figure 15.11 shows the intuitive emission process.

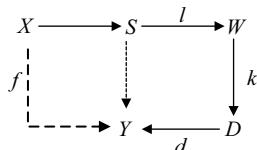


Fig. 15.11: Category representation of learning algorithms \mathcal{L} .

In general, three methods are used to find the relation morphism g . The detailed process can be found in the literature [20]. The first one uses majority voting to mark each morphism f_i and count it. If the number of votes of object f_i is assumed to be N , then $N_j = (i | f_j(x) \text{ belongs_to_class } j)$. Thus,

$$f(x) = g(f_1(x), f_2(x), \dots, f_q(x)) = \arg \max_j^q N_j .$$

Each N_j corresponds to a morphism f_i .

The second method is a weighted voting method, which calculates a weighting factor β_i for each morphism f_i . This weight is determined by the accuracy of the classifier's morphism, and the associated morphism g can be obtained:

$$f(x) = g(f_1(x), f_2(x), \dots, f_q(x)) = \sum_{i=1}^q \beta_i f_i(x) .$$

The third approach usually involves the Stacking method and uses a two-layer nested loop structure to re-learn the existing f_i as new samples to fit morphism h , so that the associated morphism g can be obtained:

$$f(x) = g(f_1(x), f_2(x), \dots, f_q(x)) = h(f_1(x), f_2(x), \dots, f_q(x)).$$

There are functors $F: \mathcal{C} \rightarrow \mathcal{L}$ between the learning algorithm category \mathcal{C} and classifier category \mathcal{L} , which exist between the corresponding objects. Sample object $S \in \text{ob}\mathcal{C}$ and $S \in \text{ob}\mathcal{L}$ are all sub-objects of the input space objects X , and there is also a relation between the corresponding morphisms. For example, $f_i \in \text{Mor}\mathcal{C}$ is obtained by transformation of morphism f in category \mathcal{L} . The classifiers of functors F are shown in Figure 15.12.

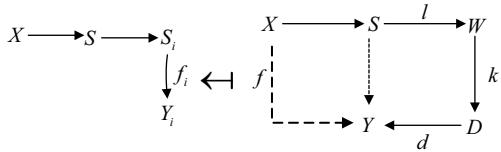


Fig. 15.12: Classifier functor F .

Through the above description of classifier functor F , it can be seen that the transformation of the morphism in the classifier category \mathcal{C} can also change the performance of the classifier algorithm. Similarly, the transformation of the morphism of the learning algorithm \mathcal{L} can also improve the classifier algorithm.

For example, an improved algorithm can be obtained by changing the input space objects in the classifier algorithm category. The input space object X is improved into feature set space object X ; at the same time, the sample set object S is transformed into feature subset object S , and a new algorithm is obtained according to the classification algorithm category. This algorithm is a feature integration algorithm proposed by Robert et al. in [21]. A large number of experimental results show that this algorithm is more effective than traditional classification algorithms in improving the generalisation performance, and it is suitable for decision trees and neural networks. We call this improved algorithm the attribute classifier algorithm.

For support vector machines, we need to consider two model parameters: kernel parameters σ and penalty parameters C . These two parameters are closely related to the generalisation capability of support vector machines (SVM) on a given learning set S . In [22], Valentini analysed the variation trend of the expected error, deviation, and variance with the model parameters of support vector machines through a large number of experiments. The obtained kernel parameters σ and penalty parameters C have an important influence on the expected error, deviation, and variance. In the value space of the model parameters (σ, C) , there is a low-bias region. When (σ, C)

is in a low-bias region, member classification appliances produce a deviation that is very low, the existence of such a low deviation region is a universal phenomenon.

Therefore, the choice of the parameters (σ, C) of the low-deviation region to complete the classifier algorithm category can also improve the classification effect, and it is called the parameter classifier algorithm.

To obtain better and more accurate classification algorithms, starting from the integration of the regression learning machine, Krogh and Vedelsby [7], deduced the integrated learning generalisation error formula. This generalisation error formula also has a very important guiding significance for the classifier category. For the learning machine q , the generalisation error of their ensemble E satisfies the following relations:

$$E = \bar{E} - \bar{A}.$$

\bar{E} is a weighted average of the generalisation error of the q learning machine, which reflects the inherent error of the learning machine. \bar{A} is a weighted average of the generalisation error of the q learning machine relative to the generalisation error of the ensemble, which indicates the difference between the learning machines (ambiguity).

Based on the two-dimensional random bagging ensemble learning algorithm described in [20], the transformation learning algorithm \mathcal{L} is implemented while changing the input space objects X . In the field of learning algorithms \mathcal{L} , the choice of appropriate parameters (σ, C) is the most important factor. However, through support vector machine model parameter research and discussion, it is found that in the low-bias region, the choice of parameters for the deviation is relatively low. Therefore, the model parameter pairs (σ, C) in the low-bias region are chosen randomly when the learning algorithm \mathcal{L} is trained to obtain the morphism f_i . The classifier combining the feature classifier algorithm and parameter classifier algorithm is called the two-dimensional classifier.

15.4.3 Example analysis and results

To verify the category theory and learning expression, mapping mechanism theory can be applied to the classifier algorithm. We perform verification and comparison through some numerical experiments and an improved algorithm is proposed, proving the validity of the theory and the expression of the improved algorithm feasibility. The feasibility of theoretical research is a topic for future study.

Before we perform the experiments, we first take the example data set (Figure 15.13), and then classify the results using the original classifier category algorithm. The result is shown in Figure 15.14. The data set is simple, but the classification effect can be seen intuitively.

Define X as Class 1, as Class 2, experimental results are obtained, and the error rates are for Class 1 errors=0.13, Class 2 errors=0.2, Total errors = 0.17 integrated.

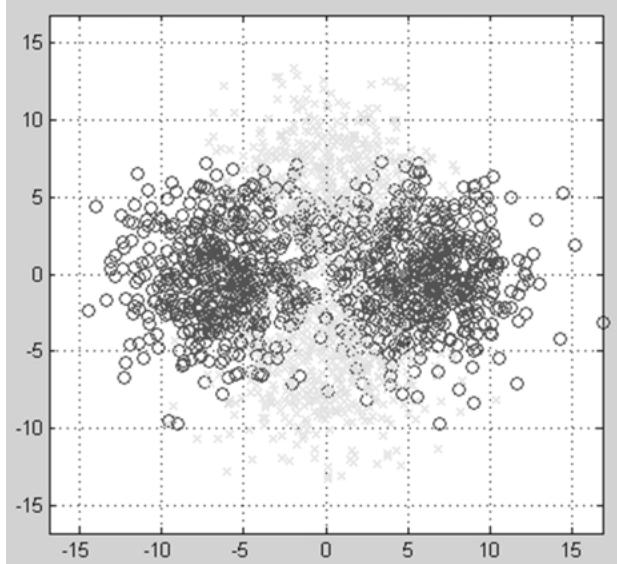


Fig. 15.13: Original dataset.

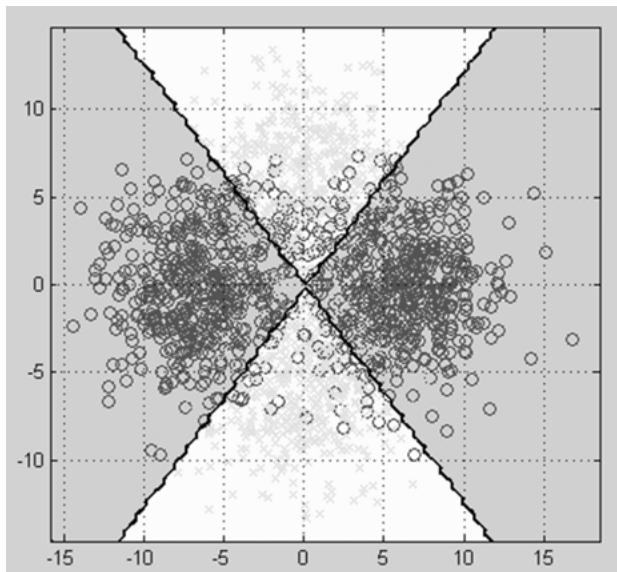


Fig. 15.14: Results after classification.

The UCI benchmark data set was chosen for the experimental data. This work chooses eight data sets for testing, including Glass, Wine, Soybean, Ionosphere, Segment, Audiology, Breast Cancer, and Balloons. Table 15.5 lists detailed information of these data sets.

Tab. 15.5: Information of UCI data set.

Serial number	Name	Sample number	Characteristic number	Category number
1	Glass	214	9	7
2	Wine	178	13	3
3	Soybean	683	35	19
4	Ionosphere	351	34	2
5	Segment	2310	19	7
6	Audiology	226	69	24
7	Breast-Cancer	286	9	2
8	Balloons	76	4	2

Tab. 15.6: Experimental results of each classifier algorithm on UCI data sets (100%).

Name	SVM	Original	Feature classification	Parameter classification	Two-dimensional classification
Glass	71.03 ± 0.29	71.96 ± 0.26	71.03 ± 0.27	72.06 ± 0.18	72.91 ± 0.24
Wine	87.01 ± 0.44	87.86 ± 0.21	88.20 ± 0.26	88.76 ± 0.17	94.94 ± 0.17
Soybean	94.58 ± 0.08	86.82 ± 0.10	87.12 ± 0.11	90.26 ± 0.25	89.82 ± 0.10
Ionosphere	93.73 ± 0.25	93.16 ± 0.25	88.32 ± 0.34	92.31 ± 0.26	94.12 ± 0.18
Segment	65.37 ± 0.31	96.97 ± 0.08	97.09 ± 0.23	96.04 ± 0.29	97.21 ± 0.11
Audiology	75.22 ± 0.14	76.55 ± 0.12	76.98 ± 0.16	73.01 ± 0.13	78.32 ± 0.12
Breast-Cancer	73.08 ± 0.52	67.83 ± 0.45	68.48 ± 0.46	68.53 ± 0.46	75.68 ± 0.43
Balloons	78.95 ± 0.46	76.32 ± 0.44	77.45 ± 0.43	77.63 ± 0.41	82.06 ± 0.31

In the experiment, the learning algorithm in all the category classifier algorithms uses the support vector machine (SVM) algorithm for training. The Gaussian kernel function, kernel parameter, and penalty parameter related to the transform were selected for 0.1 and 2; in addition, to get more stable and reliable experimental results, we choose the 10-fold cross-validation method. The data set is divided into 10 parts; nine of them are taken as the training suite, and one as the test suite (the mean of 10 times is the result of the estimation of the accuracy of the algorithm). We use the bootstrap sampling method to extract the sample from the data set. Table 15.6 shows the average test accuracy of each classifier algorithm in the eight data sets above.

From the experiment results in Table 15.6, we can see that the classifiers of category representation and support vector machines have different classification effects, and the trial range is different. Based on the mapping mechanism of learning expressions, the improved classifier category has an obvious effect on some data sets. Various improved classifier algorithms are applied to different data sets, mainly because of the different directions of improvement. In future research, to improve the classifier's classification effect, we can make changes to the classification algorithm. This section proposes a method that facilitates clearer and more explicit improvements for existing algorithms.

15.5 Example analysis

The purpose of this work is to investigate the mechanism of learning expression mapping by using the category theory and the cognitive model of cognitive science. To further verify the validity of the category learning expression theory proposed in this chapter, the following example is given.

15.5.1 Instance analysis of learning expression mapping

This section mainly uses the mapping mechanism of learning expressions to study the dimensionality reduction principle and process of high-dimensional data to illustrate the application prospects of the proposed learning expressions.

In recent years, image recognition and image analysis have attracted considerable attention in the field of pattern recognition and artificial intelligence. After nearly half a century of development, many achievements have been made in the field of image recognition and analysis, and accurate recognition has been realised in specific environments. Nevertheless, image processing technology is still facing many challenges, and the high dimensionality of images is one of the problems.

From the current results, the image dimensionality reduction algorithm of high-dimensional data can be divided into two categories. One includes linear methods, such as principal component analysis (PCA) and independent principal component analysis (ICA). The other includes a class of nonlinear methods, the most representative of which are isometric mapping (Isomap), locally linear embedding (LLE), Laplace mapping (LE), and local tangent space alignment (LTSA). Although many of these algorithms have been widely used, they all have limitations. The dimensionality reduction of high-dimensional images still cannot achieve the desired results. In this chapter, we use the mechanism of learning expression mapping and combine the similarity and difference of various dimensionality reduction algorithms to extract a high-dimensional data dimensionality reduction algorithm. Experiments show that this method is a robust and flexible method.

In [8], we proved that machine learning systems can be represented by categories, and that various learning algorithms can also be represented in terms of category theory. Thus, we can see that the dimensionality reduction algorithm of image data can also be represented by categories. Similarly, in the data reduction category, we can obtain the relation between the image data objects and the objects in the low-dimensional space according to the relation between the objects and the morphism. From this, we can derive the structural characteristics of clustering and spatial distribution of data sets. The effect of data pre-processing can also be obtained from the data reduction category. Converting high-dimensional data objects into relatively tractable low-dimensional data objects can reduce the complexity of morphism between various data objects.

Let the data reduction category be \mathcal{C} . Then, there is a family of data objects $\text{ob}\mathcal{C}$. The known objects $X \in \text{ob}\mathcal{C}$ are primitive objects of a high-dimensional data set. There is a family morphism such that any pair of objects A, B corresponds to a set $\mathcal{C}(A, B)$.

The main purpose of dimensionality reduction is to find the data manifold object category with appropriate low-dimensional coordinates, i.e., the original high-dimensional data set by various object morphisms is mapped into a low-dimensional space. The low-dimensional data object is a simple representation of the original high-dimensional data object.

Therefore, the dimensionality reduction problem of high-dimensional data can be represented by data dimensionality reduction category \mathcal{C} :

- (1) The original high-dimensional object $X: \Omega^D \subset X$, Ω^D is an object set for data samples, $\Omega^D = \{x_n\}_{n=1}^N \subset \mathbb{R}^D$, D is the dimension of the data.
- (2) Low-dimensional data objects after dimensionality reduction Y , i.e., the low-dimensional representation space of data: $\Omega^d \subset Y$, Ω^d is a low-dimensional data object that is mapped by a series of morphisms for high-dimensional data sample set objects Ω^D ; $d \leq D$ is the dimension of data after dimensionality reduction.
- (3) Data dimension reduction morphism $f: X \rightarrow Y, f \in \mathcal{C}(X, Y), X, Y \in \text{ob}\mathcal{C}$. Take sample data objects, e.g., $f(\Omega^D): \Omega^D \rightarrow \Omega^d$. We call Ω^d as the dimension reduction representation of Ω^D .

Similarly, we can obtain the corresponding data recovery process from the data reduction dimension category. The lower-dimensional data objects, which are reduced from the dimension, are transformed by the corresponding morphism to obtain the original high-dimensional data objects, represented as the following data recovery morphism:

$f^{-1}: Y \rightarrow X, f^{-1} \in \mathcal{C}(Y, X), X, Y \in \text{ob}\mathcal{C}$. The object represented by the sample data object after dimension reduction can be denoted as $f^{-1}(\Omega^d): \Omega^d \rightarrow \Omega^D$.

It needs to be explained that the data recovery morphism f^{-1} needs to satisfy certain conditions. When the condition is not satisfied, the data recovery morphism is meaningless. In general, in the linear data reduction dimension category \mathcal{C} , the data recovery morphism f^{-1} is meaningful, but in the nonlinear dimensionality reduction category, the data recovery morphism f^{-1} is meaningless.

In the data reduction category \mathcal{C} , searching for dimension reduction morphism f is the key to the formation of the whole category. However, the dimensionality reduction morphism f of this data can be of many forms. At present, from the viewpoint of mathematical thought, there are mainly linear and nonlinear representations of a morphism. Hence, we can start from two angles when we describe the data reduction category \mathcal{C} .

When the dimensionality reduction morphism of the data reduction category \mathcal{C} is represented by a linear representation, the dimensionality reduction algorithm based on the data reduction category can be described as follows:

Input: primitive high-dimensional data object X

Output: low-dimensional coordinate data object Y

- (1) Construct the associated objects M . Morphism g is the process of constructing associated objects $g: X \rightarrow M$. Associated objects M and morphism g are determined on the basis of high-dimensional data objects X and known relational objects Δ . Different relational objects Δ correspond to different association objects M . The resulting low-dimensional coordinate data objects Y have different representation meanings.
 - (2) Compute the eigenvalues Z and eigenvectors W of the associated objects $M, h: M \rightarrow Z, \varphi: Z \rightarrow W$. Obtain the compound operation law for the data reduction category $\varphi h: M \rightarrow W$.
 - (3) By analysing the characteristics and relations of known objects in the data-reduced category \mathcal{C} , the data objects Y of low-dimensional coordinates are obtained. A custom operator \cdot is needed to determine the intermediate transition object T , satisfying $\phi: T \rightarrow Y$.
-

We can represent the dimensionality reduction algorithm in the data-reduced category \mathcal{C} as a graph, as shown in Figure 15.15. From Figure 15.15, we can visually see the relation between the objects and their morphism in the dimensionality reduction category. The objects are obtained from known objects and a series of structured morphisms. This completes the flow of the algorithm from input to output.

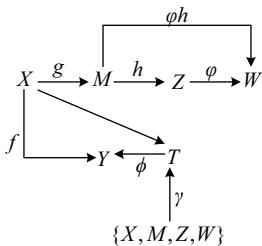


Fig. 15.15: A linear dimensionality reduction algorithm for dimensionality reduction of data.

When the dimensionality reduction morphism of the data reduction category \mathcal{C} is represented by a nonlinear representation, the dimensionality reduction algorithm based on the data reduction category can be described as follows:

Input: primitive high-dimensional data object X

Output: low-dimensional coordinate data object Y

- (1) Determine the neighbourhood points of the elements in the original high-level data object X . Construct neighbourhood objects G . Morphism l is the calculation of neighbourhood points of elements $l: X \rightarrow G$.
- (2) Construction of matrix object H . Neighbourhood objects G contain elements, attributes, and relationships (such as distance and weight) in primitive high-level data objects X . By using the known elements in neighbourhood objects G , a proper morphism mapping p is found to obtain the matrix object $H, p: G \rightarrow H$. The meaning of the matrix object H is related to the attributes of the elements in the selected neighbourhood object G .

- (3) Compute low-dimensional embedded data objects Y based on the resultant matrix object H . Use morphisms ψ to complete the mapping from H to Y , $\psi: H \rightarrow Y$. The form and method of morphism ψ depend on the meaning of matrix objects H , and different ψ need to construct different objects and morphisms.
-

Similarly, we can represent the dimensionality reduction algorithm in the data-reduced category \mathcal{C} as a graph, as shown in Figure 15.16. In Figure 15.16, there is a dotted arrow connecting the object to the object's state. It is shown that the morphism is obtained by the properties and operations of the complex operation law and the associative law.

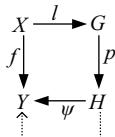


Fig. 15.16: A nonlinear dimensionality reduction algorithm for dimensionality reduction of data.

To verify the correctness and rationality of the dimensionality reduction and the dimensionality reduction process of the above data, we validate the existing algorithms. We choose the classical algorithm of linear dimensionality reduction, namely principal component analysis, and a nonlinear dimensionality reduction algorithm, namely the isometric mapping method.

Example 15.7. Define the category of the principal component analysis algorithm as the principal component analysis category \mathcal{C}_1 , known primitive high-dimensional data objects X , and vector representation of X , $\{x_1, x_2, \dots, x_m\} \subset X$. To make the raw data centralised, optimise the raw data in object X , mapping X to X' , defined by g_1 .

$$g_1(x) = x_i - \frac{1}{m} \sum_{i=1}^m x_i, \quad i = 1, \dots, m$$

$$g_1: X \rightarrow X' \tag{15.19}$$

Construct the associated matrix objects M . The associated matrix object here is the covariance matrix object, obtained by $g_2: g_2(x) = 1/m \sum x'_i (x'_i)^T$, $x'_i \in X'$:

$$g_2: X' \rightarrow M \tag{15.20}$$

Find the eigenvalues Z of the connected matrix objects M . Morphism h can be obtained from $\det(M - \lambda I) = 0$, $\lambda_i \in Z$:

$$h: M \rightarrow Z \tag{15.21}$$

where W is the feature vector object of the association matrix object M . Morphism φ can be obtained from $M_{x'_i} w_i = \lambda_i w_i$, $\lambda_i \in Z$, $w_i \in W$, $M_{x'_i}$ are the elements of the associated matrix object M :

$$\varphi: Z \rightarrow W \tag{15.22}$$

Define the operator \cdot , $\gamma: Z \cdot W \rightarrow T$, $\lambda_i \cdot w_i = \sqrt{\lambda_i}w_i$. The elements in the previous d eigenvalue object and their corresponding feature vectors are selected by the orthogonal feature of the feature vector object W . Low-dimensional data representation objects Y are obtained using morphism $\phi: \phi = [\sqrt{\lambda_1}w_1, \sqrt{\lambda_2}w_2, \dots, \sqrt{\lambda_d}w_d]^T$, $\sqrt{\lambda_i}w_i \in T$.

$$\phi: T \rightarrow Y. \quad (15.23)$$

Figure 15.17 shows the relation between objects and morphisms in the category of principal component analysis. The dimensionality reduction process of the original data is compared with the linear dimension reduction algorithm represented by dimensionality reduction.

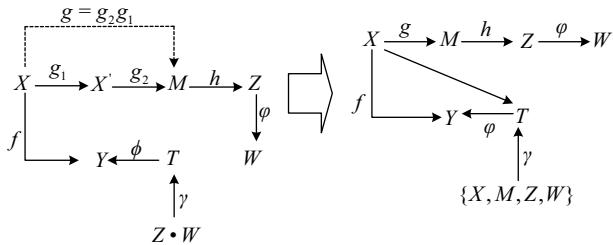


Fig. 15.17: Data reduction process of data based on principal component analysis.

Example 15.8. Define the category of the isometric mapping algorithm as the principal component analysis category \mathcal{C}_1 , known primitive high-dimensional data objects X , and the vector representation of X , $\{x_1, x_2, \dots, x_m\} \subset X$. First, the neighbourhood points of each element in the original high-level data object X are determined, and the neighbourhood objects G are constructed. Determine the distances between elements by means of the distance morphism $l_1 = d_E(i, j)$, $l_1: X \rightarrow D$. D is the distance object. The nearest k elements are taken as the neighbourhood, and the undirected graph objects G_0 are constructed by morphism $l_2: X \rightarrow G_0$. The nearest neighbour object G can be obtained by combining the distance object and undirected graph object. The weight of each connection line in the object G_0 is the corresponding element in the object D , which can be expressed as

$$l_3: D \cdot G_0 \rightarrow G. \quad (15.24)$$

Construct matrix object H . A matrix object H is formed by obtaining the geodesic distance between each element in the object G . We can get transition morphism p by using Dijkstra's algorithm, satisfying $p: G \rightarrow H$, $p = d_G(i, j) = \min_{1 \leq k \leq m} \{d_G(i, j), d_G(i, k) + d_G(k, j)\}$.

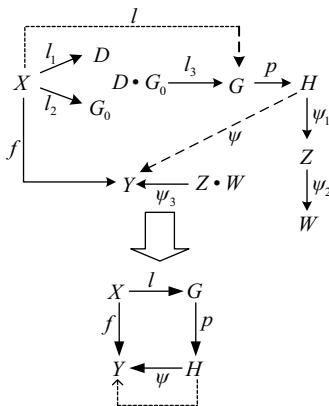


Fig. 15.18: Data reduction process based on the category of equidistance mapping.

Construct eigenvalue objects Z and feature vector objects W . Finally, low-dimensional embedded data objects Y are obtained according to objects Z and objects W , as described in Example 15.7:

$$\psi_1: H \rightarrow Z \quad (15.25)$$

$$\psi_2: Z \rightarrow W \quad (15.26)$$

$$\psi_3: Z \cdot W \rightarrow Y. \quad (15.27)$$

Figure 15.18 shows the relation between objects and morphisms in the category of isometric mapping. The dimensionality reduction of raw data is compared with the nonlinear dimensionality reduction algorithm represented by the dimensionality reduction category representation.

In this paper, we take the nonlinear dimensionality reduction algorithm in the data reduction category as the model and analyse the new data dimensionality reduction algorithm as an example to verify the feasibility and correctness of the dimensionality reduction of the data.

First, the morphism l is determined and neighbourhood objects G are obtained, and the specific morphism process is accomplished by using the neighbourhood selection algorithm based on the central point. Neighbourhood objects G include the union of m subset sets $\bigcup_{i=1}^m S_i$. Then, the matrix objects H are constructed according to neighbourhood objects G , and the covariance of the samples is mainly used. Finally, the low-dimensional embedded data objects Y are obtained by the morphism ψ of the eigenvalues and eigenvectors of each element in the matrix object H .

The Frey rawface database is used as the primitive high-dimensional data object, and the two-dimensional reduced-dimension data objects are obtained through the data reduction dimension category, as shown in Figure 15.19. Figure 15.20 shows a reduced dimension rendering of isometric mapping categories. We can see that the human face changes according to certain rules, which fully demonstrates the feasibility and correctness of the dimensionality reduction of the data.

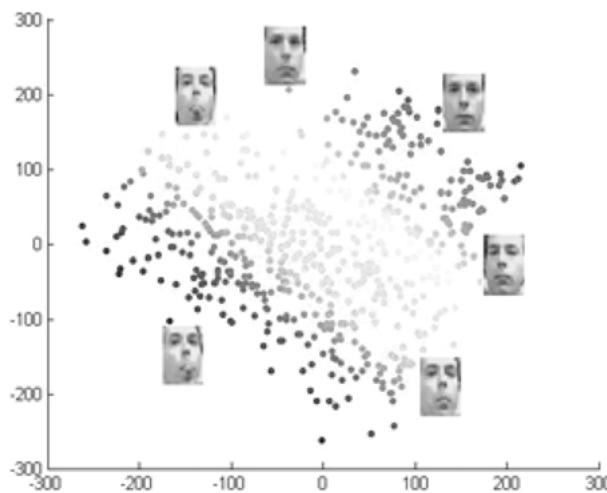


Fig. 15.19: Dimensionality reduction on Frey rawface data.

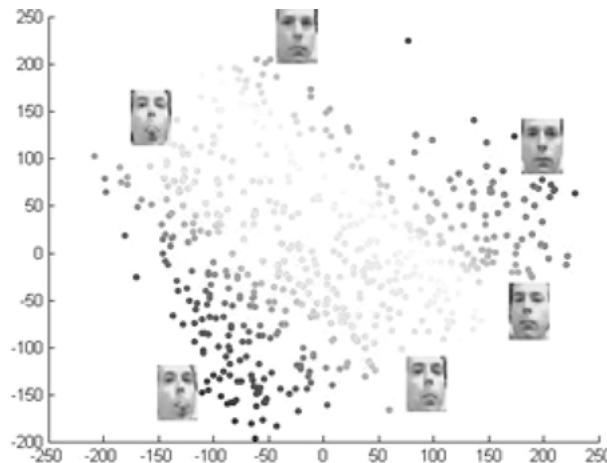


Fig. 15.20: Isometric mapping category dimensionality reduction on Frey rawface data.

The ORL standard face database is used to verify and compare the original high-level data objects. The face data of five people among 40 were chosen as the input data; for each of them, we selected the first five pictures, giving a total of 25 pictures as sample data. The other 25 were used as test data. The dimensionality reduction graphs in Figures 15.21 and 15.22 are obtained by reducing the dimension of the data reduction category and the isometric mapping category, respectively (different shapes, different colours, and different faces).

The dimensionality reduction of the Frey rawface data sets and ORL data sets is achieved by using the instance algorithm of data reduction category and isometric mapping category, respectively. The results of dimensionality reduction are identified

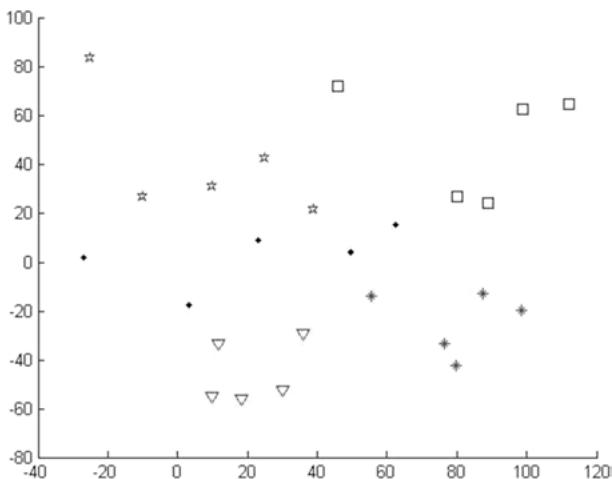


Fig. 15.21: Dimensionality reduction on ORL.

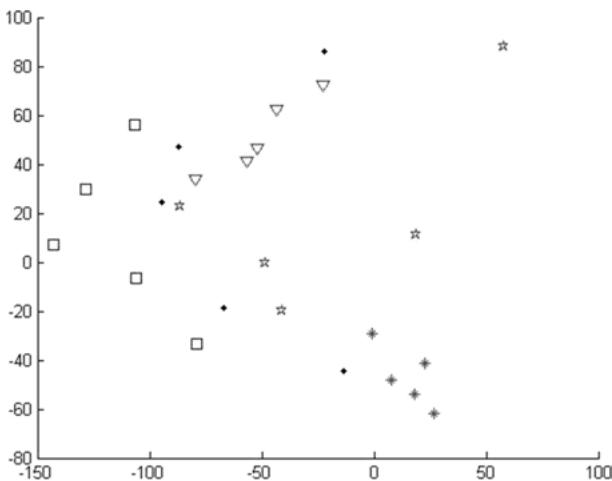


Fig. 15.22: Isometric mapping category dimensionality reduction on ORL.

by a semi-supervised learning algorithm, and the correct recognition rate is obtained, as shown in Figure 15.23.

In summary, this section gave the application of the mapping mechanism of learning expressions: the category representation of data reduction dimension. The description of the category of the data dimensionality reduction algorithm was given. It was further described in two aspects: linear data reduction dimension category and nonlinear data reduction dimension category. Experimental verification was presented to prove the validity and feasibility of the reduced dimension category and the possibility of further research.

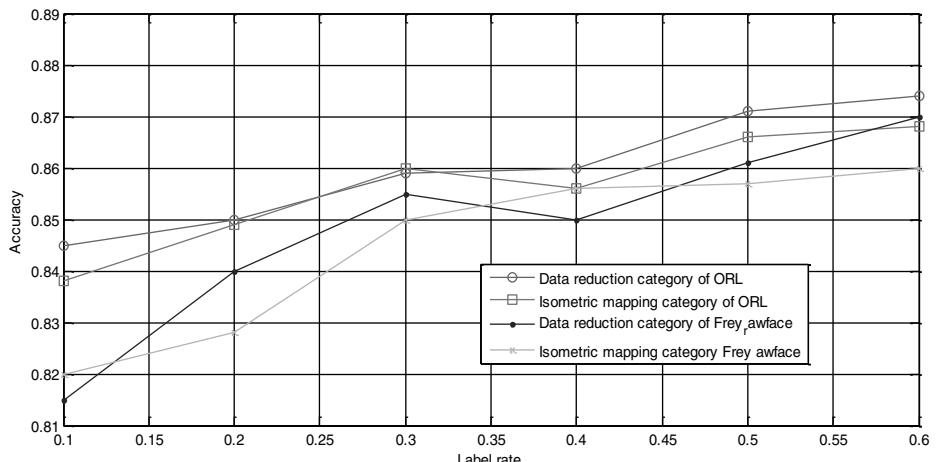


Fig. 15.23: The recognition rate obtained by using the semi supervised learning method in the case of data reduction and the category of equidistance mapping.

15.5.2 Case analysis of image recognition

In this section, face recognition and hand recognition are chosen as verification experiments to further demonstrate the feasibility and practicality of the proposed theory.

First, we develop a systematic understanding of the process of image recognition. Figure 15.24 shows the whole process of image recognition. We can see that in this chapter of image recognition, we need to reduce the high-dimensional image data first and then classify the processed data. On the one hand, it can simplify the storage of high-dimensional image data. On the other hand, it can simplify the classification process and alleviate the pressure for the classification system, so as to achieve higher efficiency.

For the dimensionality reduction part, we use the proposed dimensionality reduction category algorithm, and we use the linear data reduction category and the nonlinear data reduction category to preprocess the upper image data. For the classifier part, we mainly use the classification algorithm category proposed in the fourth chapter. In the experimental analysis section, we can get the improved classification algorithm, which has different effects on different data sets, some of which are obvious and some of which are mediocre. To ensure that the image recognition system gets the best results, three algorithms will be used here for verification. Finally, we will choose the classifier algorithm with the best results. By combining the previous chapters on dimensionality reduction and classifier category, we can get the category description of image recognition, as shown in Figure 15.25.

In this section, we use the ORL [23] face database. The ORL face database was created by the AT&T Laboratory of the University of Cambridge. It includes face images of

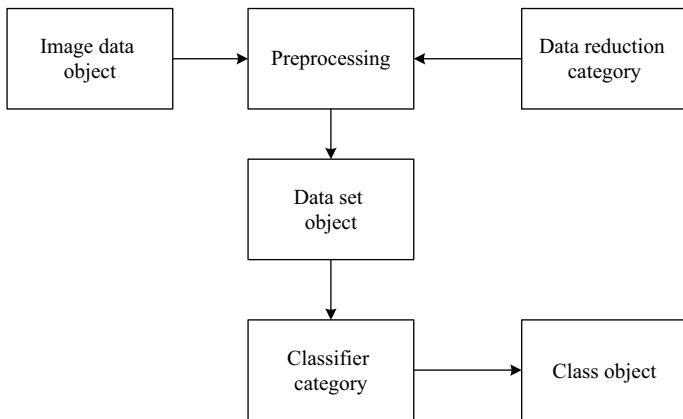


Fig. 15.24: Image recognition process.

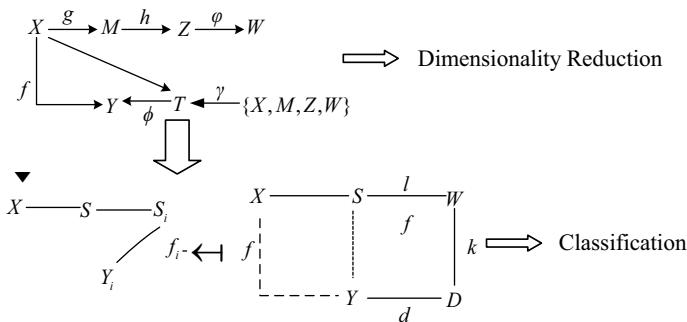


Fig. 15.25: Category representation of image recognition.

40 different people; each face set has 10 images captured at different times and under different light conditions, with different expressions and posture changes, ornaments, regimes, etc. Thus, the database has a total of 400 facial images. Figure 15.26 shows the 40 human faces in the ORL data.

According to the classifier category, we can see that the two-dimensional classifier algorithm is better. Therefore, we choose the method of data reduction dimension category to extract the features. For the parameter problem, a large number of experimental results are analysed, where the kernel parameter = 0.2 and penalty parameter = 2.

First, the dimensionality reduction category is described according to the dimensionality reduction in Figure 15.25. The 400 face images in the ORL database are reduced to two dimensions, and the resulting 2D space embedding map is shown in Figure 15.27.

The data after dimensionality reduction is used as the input object of the classifier category. For each person, eight pictures are taken as the training sample and the two



Fig. 15.26: 40 Face maps of the ORL face database.

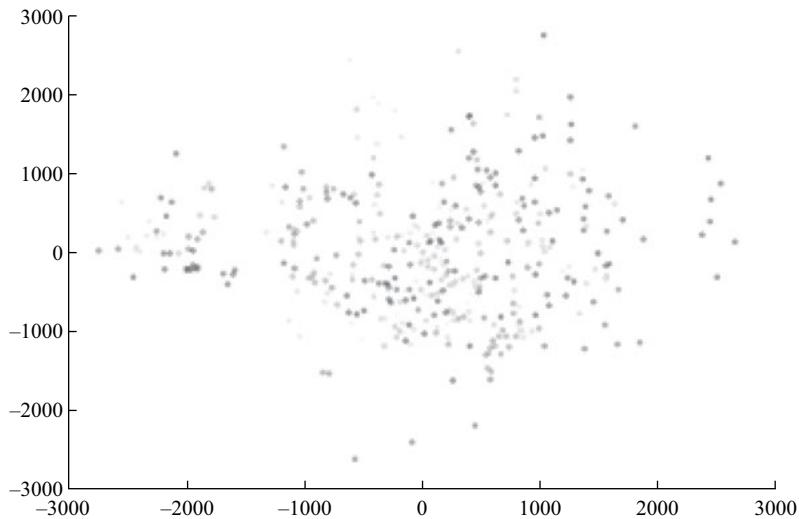


Fig. 15.27: Embedding ORL data sets into two-dimensional space by using the dimension reduction category.

remaining pictures are taken as the test object. The correctness of face recognition is 86.47% as indicated by 10-fold cross-validation. Although the recognition accuracy is not satisfactory, it is sufficient to prove the feasibility of image recognition.

15.6 Summary

Machine learning is an important research area in artificial intelligence applications. In recent years, machine learning has played an important role in many fields. In 2011, *Science* magazine published an article on the theme of ‘data processing’ (Dealing with Data), which clearly stated, ‘data is science, scientific data.’ To make good use of data, machine learning methods still have a long way to go. Therefore, exploring new learning theories and algorithms is an unavoidable problem in the field of machine learning. It is possible to select category theory to study the representation of learning systems.

In this chapter, we first introduced the representation of the learning expression category and gave the representation of both the machine learning system category and the learning operator. We derived the natural transformation in the category of the learning expression, and the system gave the basic definition of the category of the learning expression. We used the decision tree and Bayesian category as an example to prove that the machine learning theory system is feasible. In addition, the functor category theory can also be adopted to study the expression. Learning and mapping expressions between functors can also use the natural transformation theory to explain this. Thus, this work lays the theoretical foundation for future research.

Second, we introduced the mapping mechanism of learning expressions, and gave abstract concepts in the expression category as well as instances of the mapping mechanism between expressions. We listed the basic concepts of single morphism, epimorphism, objects, and functors as projective injective objects. From the machine learning category, we found that an expression mapping mechanism exists. We also used decision tree learning and Bayesian learning as examples to describe mapping learning.

Bibliography

- [1] Zeng HJ, Zhang YK. Machine Learning. Beijing: Machinery Industry Press, 2006.
- [2] Cai ZX. Artificial intelligence and its applications. Beijing: Tsinghua University press, 2004.
- [3] Nian ZG, Liang S, Ma FL, Li SP. Research and application of knowledge representation. Computer Application Research, 2007, 24(5), 234–235, 286.
- [4] Zhang HL. Research on image semantic networks. Jilin: Jilin University, 2007.
- [5] Li XD, Gu YQ. Algebraic representation of XML query and its query optimization. Computer Science, 2002, 29(6), 56–61.
- [6] Zhang P, Wang B, Qing XX. Integrated application of multiple knowledge representation methods in expert systems. Microcomputer application, 2004, 20(6), 4–6.
- [7] Vapnik V. The Nature of Statistical Learning Theory. Springer-Verlag, New York, 1995.
- [8] Zhou LL, Li FZ. Research on Mapping Mechanism of Learning Expression. In: Proceedings of 2010 Rough Set and Knowledge Technology (RSKT 10), 2010, LNAI 6401(298–303).
- [9] Asperti A, Longo G. Categories Types and Structures – An Introduction to Category Theory for the working computer scientist. M.I.T. PRESS, 1991.

- [10] Xu JF, Fei ZM. A number of applications of category theory in Computer Science. Institute of Computer Software, Nanjing University, 1989.
- [11] Zhang Y, Li SX. Formal ontology research based on category theory. Computer Science, 2006, 33(9), 1–3, 117.
- [12] Yang XT, He N. Ontology integration description based on category theory. Computer Engineering, 2009, 35(6), 76–78.
- [13] Zheng YJ, Xue JY. A Category Theoretic Approach to Search Algorithms: Towards a Unified Implementation for Branch-and-Bound and Backtracking. Proceedings of 2009 4th International Conference on Computer Science & Education. 845–850.
- [14] Dulany B, Michael J. An Application of Category-Theoretic Design Methods to the Control of a Simulated Robot [J]. Proceedings of International Joint Conference on Neural Networks, Orlando, Florida, USA, August 12–17, 2007.
- [15] Michael J. Category Theory Applied to Neural Modeling and Graphical Representations [C]. IEEE 2000, 35–40.
- [16] Li HD, Yao TX. Pattern Classification. Beijing: Machinery Industry Press, 2006.
- [17] Quinlan JR. Induction of decision trees. Machine Learning, 1986, 1(1), 81–106.
- [18] Fu HX. A symplectic group classifier in Li Qun machine learning. Suzhou: Soochow University, 2008.
- [19] Fan M, Meng XF. Data Mining: Concepts and Techniques. Beijing: Machinery Industry Press, 2007.
- [20] Wang L. Research on some problems of support vector machine learning algorithm. Chengdu: University of Electronic Science and Technology, 2007.
- [21] Robert B, Ricardo GO, Francis Q. Attribute Bagging: Improving Accuracy of Classifier Ensembles by Using Random Feature Subsets, Pattern Recognition, 2003, 36, 1291–1302.
- [22] Krogh A, Vedelsby J. Neural Network Ensembles, Cross Validation, and Action Learning. Advances in Neural Information Processing Systems, 1995, 7, 231–238.
- [23] <http://www.sc.toronto.edu/~roweis/data/olivettifaces.mat>.

16 Neuromorphic synergy learning

Neuromorphic synergy learning is an interdisciplinary frontier of cognitive science, artificial intelligence, brain-like intelligence, machine learning, robotics, pattern recognition, and image processing. Moreover, it is a key technology for big data, the internet of things, and cloud computing. Thus, this chapter mainly focuses on the three core scientific problems in neuromorphic synergy learning, i.e., symbol grounding relation, bidirectional mechanism, and affordance learning, and studies neuromorphic synergy learning from four levels, namely proposing a new theory, exploring a new method, initiating a new technique, and developing a new platform. This chapter is divided into the following sections: (1) Introduction, (2) Core scientific problems, (3) Lie group cognitive theory framework, (4) Neuromorphic synergy learning theoretical framework, (5) Design of neuromorphic synergy learning verification platform, and (6) Summary.

16.1 Introduction

In the post-genomic era, international artificial intelligence experts have been studying super-computers or super-chips with brain-like intelligence. By studying algorithm theories, they have been trying to research and establish a problem-analysing and knowledge-acquiring analogue system to explore the human brain mechanism. Because synergy is the main work mode of the human brain, neuromorphic synergy learning is one of the most challenging and significant scientific subjects in the field of brain-like intelligence. Moreover, it has been the research focus of recent studies in the fields of computer science, cognitive science, machine learning, and robotics. It has attracted considerable attention from the industrial community, especially in the age of big data, the internet of things, and cloud computing, along with the upgrade of Industry 4.0.

Related research on human-inspired planning has been determined as an important national strategic development direction in many countries (such as USA, Japan, European Union, and China). The USA was the first country to implement the brain project. The American brain project is called the brain activity map project or BAM. This project aims to explore the human brain working mechanism, plot the brain activity map, and finally develop therapy for brain diseases. To maintain a technical advantage, the USA took the lead in studying brain-like computing chips. By imitating the human brains working principle and replacing the von Neumann architecture with neurons and synapses, chips can process information data in an asynchronous, parallel, slow, and distributed manner, and they possess the ability for autonomous perception, recognition, and learning. The Obama administration's Brain Initiative is comparable with the Human Genome Project. America's National Institutes of Health brain

project committee co-chairman William Newsome has stated that BAM is more complex than the genome project and has a deeper scientific significance. The European Union human brain project or HBP is expected to greatly accelerate comprehensive understanding of the brain structure and function, and it will contribute to a deeper study of brain diseases and an optimised therapeutic schedule, thereby establishing a new, revolutionary form of information and communication technologies (ICT). This project was selected for the European Commission future and emerging technologies (FET). It is the most important human brain research project globally, and it is expected to receive 1 billion Euros in financial support from the European Commission over 10 years. The European Union HBP will be undertaken by the Helmholtz Association of German Research Centres, and it will carry out human brain simulation research by employing super-computers. This research plan is one of the most important international research programs, and it is also known as the flagship program. Over the next 10 years, the European Union is expected to provide 1 billion Euros to fund this project. The Helmholtz Association Julich Research Centre will develop super-computers combined with neuroscience to conduct detailed brain neuron studies. The objective of the research plan is to simulate all the human brain components in order to better understand their function as well as to facilitate diagnosis and therapy of diseases. Japan's brain mapping by integrated neuro-technologies for disease studies (Brain/MINDS), is expected to accelerate research on human brain diseases, such as Alzheimer's disease and schizophrenia, through the study of marmoset brains.

Brain science and brain-like intelligence have already emerged as research hot-spots globally. The Chinese government is devoting considerable efforts toward human brain study, and it has initiated the China Brain Project. Many colleges and research units have subsequently established corresponding research institutes. For example, in 2015, Peking University established the Brain Science and Brain-like Research Centre, while Tsinghua University established the Brain-like Computing Research Centre. In 2016, Soochow University and the State University of New York at Buffalo jointly established the Machine Learning and Brain-like Computing International Cooperation Laboratory.

The China Brain Project has launched research in three directions, namely brain acquaintance, brain protection, and brain simulation, and it aims to gradually formulate a research pattern that is closely related to fundamental research of the brain cognitive principle, serious brain diseases, and brain-like artificial intelligence. Thus, brain-like computing and artificial intelligence research are important components of the China Brain Project. Zheng Nanning, an academician from the Chinese Academy of Engineering, believes that research based on the human brain information processing mechanism and human intelligence will facilitate the development of a set of computing theories and technologies of brain-like intelligence, thereby guiding future information technology in an intellectualised direction. Brain-like computing is one of the great challenges for the scientific community in the present century, and

it has attracted considerable attention in the fields of current computer science and artificial intelligence. Tan Tieniu, an academician from the Chinese Academy of Sciences, has emphasised that brain-like intelligence has already emerged as a strategic behaviour in developed countries. Thus, developing next-generation intelligent systems based on the brain information processing mechanism has become an important trend in artificial intelligence research, and future machine intelligence will require a deep combination and integration between subjects such as brain science, neuroscience, cognitive science, and psychology. Academic journals, such as *Science* and *Trends in Neuroscience*, have reported on human brain projects and neural informatics, and they have claimed that the Human Brain Project, which includes an extensive context, is bigger than the Human Genome Project. As the core of brain-like intelligence, neuromorphic synergy learning algorithm research will play a decisive role in the field of brain computing.

For decades, brain science has stimulated machine intelligence and promoted the development of research on brain and cognitive theory, intelligent science, and information and computing techniques for brain-like intelligence. In recent years, the success of deep learning in vision, hearing, and speaking has further inspired studies on brain-like intelligence. Currently, brain-like intelligence is the representative of 21st century frontier technology, and plays an important role in human brain science plans launched by many countries around the world, especially in the fields of big data, the internet of things, and cloud computing. Thus, brain-like intelligence has become one of the most challenging technologies.

The history of neuromorphic synergy learning can be traced back to the last century, when artificial intelligence was born.

The concept of artificial intelligence was first proposed by John McCarthy et al. at the Dartmouth Conference in 1956. Artificial intelligence aims to make a machine implement human-like intelligent behaviours through computer programming. This is also the original intention of brain-like science. The relationship between the human brain and the computer information processing mechanism was explored preliminarily in 1958 [1]. The author discussed the difference between the mechanisms of a computer and the human brain, and emphasised that the brain neural system can be regarded as a hybrid computing system of digital and analogue computation.

Research in the field of brain-like science is still in its infancy. Hence, the academic community has not formed a widely accepted concept in this regard. In 2007, the term brain-like intelligence was first used at the International Symposium of Creating Brain-like Intelligence, which was organised by Sendhoff and Spoms et al. in Germany. The subsequently published conference proceedings pointed out that brain-like intelligence can achieve intelligence as demonstrated by the brain, preferably the brains of highly evolved creatures. Overall, the cognition of the human brain's information processing mechanism is relatively simple, and an intelligent system that is completely consistent with the human brain cannot be developed. Thus, the best way

to promote artificial intelligence research on the basis of the human brain's information processing mechanism is to borrow from the brain's working mechanism, instead of complete imitation.

In 1959, the Turing Award winner Newell started exploring an important scientific problem, i.e., how human thinking can be reproduced by computer systems. Cognitive psychology was the core of Newell's exploration, which led to the important achievement of cognitive architecture development. Thus far, in the fields of cognitive psychology and artificial intelligence, the cognitive architectures SOAR and ACT-R that have been widely applied to mental models are inspired or directly developed by Newell. Based on Newell's work, Laird proposed the SOAR system. This system is a production system and is based on the implementation of human cognitive function models [2, 3]. The SOAR system is widely used in cognitive robots and the military domain. The ACT-R cognitive architecture developed by Anderson is also inspired by Newell's work. Anderson introduced the brain information processing mechanism of cognitive neuroscience into ACT-R, and implemented the modelling of the brain's region circus, which can support the prediction of brain region circus activities under a certain task. As an application of intelligent systems, ACT-R is widely used in natural language processing, education, and the military domain.

In the literature [4–7], Marr established the foundation for storage, processing, and transfer of neural group computation. In particular, he made an important contribution toward learning, memory and vision-related circus computing. Hodgkin and Huxley created the first refined action potential model according to physiology experiment results and established the foundation for dimension calculation models of neuron ions. Tsodyks et al. built the synapse calculation model between neurons and laid the foundation for neural network information transfer.

Research on artificial neural networks can be traced back to the 1940s [8]. In 1957, the perceptron proposed by Frank Rosenblatt was the representative of a shallow artificial neural network. In 1974, the back-propagation algorithm proposed by Werbos et al. solved the learning difficulties of multi-layer perceptrons [9]. At that time, the promotion of computational abilities could not support the training of large-scale neural networks, which restricted the development of neural networks for a long time [10]. Before deep learning was developed, Rumelhart et al. proposed the error back-propagation algorithm [11] in 1986. In 1990, Le Cun et al. proposed the convolutional neural network [12]. Following the development of deep learning, in 2009, Hawkins et al. proposed the hierarchical temporal memory model [13, 14]. This model uses cortex six-layer organisation, the information transfer mechanism between neurons of different layers, and the cortical column information processing principle for reference. In 2012, at the University of Waterloo, Canada, Eliasmith's group proposed the Semantic Pointer Architecture Unified Network on the basis of their previous work. They put 2.5 million virtual neurons into 10 modularised brain regions and constructed the workflow brain region computing circus to implement multifunctional and multi-brain-region synergy neural networks. This is the signature work in the field of multi-

brain-region synergy computing. In 2015, DeepMind proposed a deep reinforcement learning model and carried out an independent study on a neural network system. This system learned to play 49 different computer games automatically by interaction with the environment, which is close to human performance.

In the research on artificial neural networks, most scholars have focused on the promotion of learning performance. Riesenhuber and Poggio et al. developed artificial neural networks in the brain-like direction, e.g., they imitated the human vision information processing path to construct the HMAX model [15, 16]. In addition, Bengio et al. proposed brain-like reinforcement learning according to the information processing mechanism of the basal ganglia and prefrontal lobe.

Under such circumstances, the united academic circles of Europe, USA, Japan, and other countries have introduced national brain plans. In 2013, the core research objective of the European Union Human Brain Project was to construct a human brain model by super-computers and to study how the human brain works by computer imitation methods, in order to provide inspiration for future computing systems and robotics, as well as to promote their transformation [17]. In 2014, the United States Intelligence Advanced Research Projects Activity (IARPA) organised a new machine intelligence project, namely Machine Intelligence from Cortical Networks (MICrOns), which inspired a new branch of machine intelligence through research on the cortex. In 2013, the Google Brain project led by Ng of Stanford University and Dean of Google used deep neural networks to achieve breakthroughs in image recognition on a large-scale parallel computing platform consisting of 16,000 CPU cores [18]. In the same year, Microsoft Research and Baidu used deep neural networks in the fields of speech and image processing, and rapidly improved the recognition effect in visual and auditory processing [19].

16.2 Core scientific problems

In [17], it has been pointed out that for decades, no natural or artificial system besides the human brain has shown the ability to adapt to new environments or challenges, automatically acquire new information and new skills, or make effective decisions and work stably in complex environments. No system can remain as robust as the human brain in the case of multiple lesions when dealing with the same complex tasks, and no artificial system can rival the brain's low energy consumption. Thus, from the viewpoint of information processing and intelligence, learning the brain's principles and proposing brain-like intelligence computing technology is an important direction for realizing artificial intelligence innovation. At the same time, in recent years, rapid progress in neuroscience and cognitive science has made it possible to observe different scales and different structures of the brain regions, neural clusters, and neural micro loops, and to obtain relevant data for some activity of brain tissue [26–28]. The process of human brain information processing is no longer based on guesswork.

The cognitive mechanism of the human brain is now more thoroughly known through interdisciplinary and experimental study. Therefore, studying neuromorphic synergy learning by using the brain's information processing mechanism, neural mechanism, and cognitive behaviour mechanism is an important method for developing brain intelligence.

The rise and promotion of big data, cloud computing, mobile terminals, and the internet of things have led to significant innovation and development prospects in the fields of information and smart processing. The core problems of these emerging technologies are data analysis, understanding, and efficient utilisation. Whether the data is perceptual data obtained through collection from and interaction with the natural environment, or synthesised data generated by users, most of the data is unstructured data, such as image, video, voice, and natural language. There is a significant gap between the ability of artificial intelligence to understand such data and human ability. At the same time, the magnitude of the data and the real-time requirements are major challenges for artificial intelligence. These issues impede the efficient use of data. Therefore, representation and understanding of multi-source and multi-scale massive, real-time, and unstructured data are effective ways to study neuromorphic synergy learning. The brain is a common type of intelligent system. It can extrapolate, understand, and deal with visual, hearing, language, and other data, and it can also undertake independent study, reasoning, decision-making, planning, and other types of problems. Moreover, the perception, intelligence, and thinking ability of the human brain can be formed and evolved naturally in continuous learning. However, the computer is based on the Turing machine model and the von Neumann architecture. The Turing model is limited by predefined rules and input-output, while the von Neumann architecture is also predefined; hence, they cannot evolve themselves on the basis of changes and demands in the outside world. At the same time, from the perspective of problem solving, nearly all artificial intelligence systems need to be formalised, turning problems into specific computational problems (such as search, automatic reasoning, and machine learning) and then processing them. Although efforts have been devoted toward developing a general problem solving system in the history of artificial intelligence, it still needs to be manually summarised as a series of formulae or Horn clauses. The human brain uses the same information processing system for automatic sensing, problem analysis and solving, decision-making control, etc. Therefore, constructing intelligent systems that can be automated and have autonomous evolution capability is another important and effective way to study neuromorphic synergy learning.

Thus, the core problems of neuromorphic synergy learning include the:

- (1) Symbol grounding relation problem.
- (2) Bidirectional mechanism problem.
- (3) Affordance learning problem.
- (4) Representation problem of neuromorphic multi-source data.

- (5) Fusion problem of multi-scale data in visual perception.
- (6) Understanding of massive, real-time, unstructured multi-source and multi-scale data problem.

In conclusion, this chapter mainly focused on the three core scientific problems of neuromorphic synergy learning, i.e., the symbol grounding relation, bidirectional mechanism, and affordance learning, and studied neuromorphic synergy learning from four levels, namely proposing new theories, exploring new methods, starting new technologies, and developing new platforms. Next, we present the framework of Lie group cognitive theory.

16.3 Lie group cognitive theory framework

The application of cognitive science in computer vision has attracted considerable attention from international scholars. Some famous domestic and foreign universities and institutes have established research centres dedicated to visual cognition, including the USA and China (Massachusetts Institute of Technology (MIT) Department of Brain and Cognitive Sciences, Brain Cognitive Sciences Artificial Intelligence Laboratory, as well as the Cognitive Research Center of Harvard University, Brown University Cognitive and Language Institute of Cognitive Sciences, University of California, Berkeley, Brain and Cognitive Sciences, Chinese Academy of Sciences Research Center, Cognitive Neuroscience Laboratory of Peking University, Tsinghua University Institute of Information Cognition and Intelligent Systems, etc. In addition, research on cognitive science and related information processing has been included in the national key scientific development strategies of some developed countries since the 1990s. Examples include the international Human Frontier Science Program proposed by multinational cooperation, the Human Brain Project proposed by the National Institutes of Health, the American Natural Science Foundation, Convergent Technology for Improving Human Performance proposed by the National Science Foundation, the U.S. Department of Commerce, and the Japanese Brain Science Age Plan.

Since the 1970s, with the development of cognitive science, important changes have taken place in computer vision research, leading to many visual cognitive computing theories guided by cognitive psychology. These theories of visual cognitive computing can be divided into the local priority theory mainly based on the Marr theory, the global priority theory based on topology theory, and the interaction activation theory proposed by McClelland.

The visual computing theory proposed by Marr in the 1970s and 1980s [20], divided the visual cognitive process into the formation stage of the three-stage representation. Marr called the first stage of the visual process the two-dimensional base element figure phase, which is the stage of early vision formation. Marr believed that

in this stage, the human vision system obtains the base map from the input image. According to the characteristic that the boundary of the object corresponds to the density in the image, Marr used the base element figure to describe the region of intensity change in the image, local geometric relation, and organisation structure. The base elements used in the figure include spots, endpoints, edge fragments, line segments, curve groups, and boundaries. The second stage is called the 2.5-dimension factor figure phase, which is the intermediate visual processing stage. In this stage, the base element in the first stage is organised into a 2.5-dimensional diagram in a different way through certain symbolic processing. This 2.5-dimensional image describes the orientation, depth, and discontinuity of the visible surface in the observer-centred coordinate system. The third stage of the visual process is also known as the late stage of vision. In this phase, the input image, 2-dimensional base element figure, and 2.5-dimensional factor figure will be integrated to form an object-centred 3-dimensional representation that can be used to perform the final pattern recognition task. On the basis of Marr's visual computing theory, Biederman proposed recognition by the component theory in 1987 [21]. The theory of component recognition assumes that human beings will break up objects with complex structures into simple components when they are in identification mode, i.e., objects are composed of some basic components. Biederman referred to these basic shapes as geometric ions and pointed out that the geometric ions include diamonds, columns, spheres, arcs, and wedges. Component identification theory is very similar to character analysis theory. The difference is that the geometric ions mostly have some three-dimensional geometry. The identification process can also be understood as the geometric ion matching process of mutual combination. However, the theory of component identification can only solve the problem of a relatively crude recognition category, and there are some limitations to the identification of different identities of similar things. Another popular visual cognitive computing theory guided by local priority theory is the feature integration theory proposed by Treisman et al [22]. Treisman divided visual processing into a feature registration stage and a feature integration stage. The feature registration stage mainly occurs in the early stage of visual processing. In this stage, the human visual system detects independent features in parallel, including colour, direction, size, and distance. By independent coding of different types of characteristics, the feature map describing each characteristic component is obtained. In the stage of feature integration, the human vision system integrates the independent features of parallel extraction to form a representation of objective things. This process occurs in the later stage of visual processing, and the integration of features occurs in areas of visual focus. All the characteristics that appear at a particular location are integrated to focus on the object being observed.

In the early 1980s, Chen Lin proposed the visual topological theory [23], which raised a powerful challenge to the visual computing theory of Marr. Visual topological theory suggests that in the early stages of human vision, the vision system is preferred to a wide range of topological properties of perception, and for simple parts or

local geometric properties of the component model, it is not sensitive. In 2005, various comments on the visual topology theory were made on the basis of visual cognition, and the concept of perceptual organisation was discussed from the perspective of invariance of the transformation. Chen Lin emphasised using topological invariance to describe the global properties of perceptual organisation; topological invariance includes graphics unchanged under topological transformations of connectivity and closeness, as well as topological properties such as holes. Chen Lin stressed that the perception of topological nature is superior to the perception of locality.

In 1981, McClelland et al. proposed the theory of the interaction activation model from the perspective of simulating human neuron activation and inhibition [24]. McClelland believed that the human visual system consists of many processing units, each of which is called a node. These nodes are connected to a large number of nodes by activation and suppression. The activation value of the node is affected by the input and adjacent nodes. Between nodes, a complex network that simulates the human nervous system can be formed through connections between the same layers and different levels. Under the guidance of this theory, the perceptron emerged as a representative of the first-generation neural network model, with back-propagation neural network as a representative of the second-generation neural network model. The perceptron is also known as a single-layer artificial neural network, which recognises objects by continuously learning the weights of the manual coding characteristics. The most serious defect of the perceptron is the inability to deal with the problem of linear inseparability. Owing to the limitations of the perceptron's learning ability, multilayer perceptron (MLP), which is composed of multiple node layers, was developed, followed by a full connection between adjacent nodes. The multi-layer perceptron is the generalisation of the single-layer perceptron. It overcomes the disadvantages of linear inseparability of the single-layer perceptron and can be used to obtain different forms of expression with different activation functions. The training of the multi-layer perceptron is carried out with a supervised back-propagation algorithm, and the learning network parameters can fit the complex classification function and have a wide range of application scenarios. In the 1980s, the back-propagation neural network was a very popular machine learning method. However, in the 1990s, many researchers abandoned the neural network model of the multi-layer perceptron owing to its large parameter space and difficulty in training. Moreover, Vapnik et al. invented a better perceptron model, called the support vector machine (SVM) [25].

In recent years, numerous physiological and psychological studies have provided a new way for us to understand how visual cognition can be used in visual perception. An increasing number of researchers have been focusing their attention on bio-inspired features, and a variety of feature models of simulated biological vision have been presented. For example, Riesenhuber and Poggio [15] proposed a quantised hierarchical feedforward model based on the what-path in the visual pathway. The invariant features are gradually extracted, and the rationality of this feature model is verified by experiments in simple artificial construction scenarios. Serre [16] continued

to extend the HMAX model and proposed the standard model feature (SMF). The non-deformed C2 characteristics of target displacement and scale are extracted through hierarchical units of cascade. In real-world target recognition, scene classification and other tasks have achieved good results.

The symbol grounding relation problem

is one of the core scientific issues in embodied cognition. The core assumption of classical artificial intelligence is the behaviour of the symbolic system, which is the same as the behaviour of a person with a brain. To prove that the computations were incorrect, Searle simply confirmed in 1980 that this was not possible. For example, computers handle Chinese characters only according to their shape, not their meaning. The meaning of symbols in a symbol system is not internal but external. In other words, cognition cannot be just a sign operation. The corresponding question is how to get meaning from the symbols related to the question, and the meaning of the question is related to consciousness and perception. For example, for robots to talk to people, they have to understand the correspondence between what they say and what their external environment is, which is a symbol grounding relation problem.

Definition 16.1. Symbol Grounding Relation of Cognition: The corresponding relation between the content of the cognition object and the external environment is called the symbol grounding relation of cognition.

The bidirectional mechanism problem

means that vision and cognition are not two separate systems. The visual system is an input system that extracts information from the environment. The cognitive system is a central processing system that processes information transmitted by the input system (such as speech, memory, and thinking). Cognition is also intellectual in nature, and it shares a common system with both the intuitive and the neurological levels. The visual state includes the unconscious neural representation of the external stimuli and the conscious experience of choice. Once the visual state is produced, some of the content is extracted and stored in long memory by selective attention. In subsequent recovery, this memory can play a symbolic role in representing the outside world and enter the symbolic operation process. A cognitive representation is formed when a large number of perceptual symbols are concentrated. The conformance relation of the recently formed perception and cognition directly leads to the generation of perceptual symbol theory. Cosy found that the one-way hypothesis is not only inconsistent with the perceptual cognition principle of a human being, but also one of the main bottlenecks that prevent the individual layer's collaborative computing technology from achieving the fundamental breakthrough. The 'two-way hypothesis' is proposed to replace the 'one way hypothesis' in order to overcome this bottleneck.

Definition 16.2. The Bidirectional Mechanism of Cognition: The mutual cognitive mechanism between perception and cognition is called the bidirectional mechanism of cognition.

The affordance relation problem

is the most central concept in the psychology of perception and ecology, and its specific definition remains disputed. It is widely believed that the affordance relation is an interaction between an individual and the environment, and the property of the environment makes it possible for the behaviour of the individual to be carried out. There are three main points: (1) affordance is a potential possibility; (2) affordance is an interaction between the individual and the environment; and (3) affordance extraction is restricted by intention. Affordance includes the individual, environment, individual attributes, and relationship between properties, and there is no complete theoretical system.

Definition 16.3. The Affordance Relation of Cognition: The interaction between the cognitive individual and the environment is called the affordance relation of cognition.

Definition 16.4. Lie Group Cognition: According to the bidirectional mechanism of perception and cognition, the relationship between the individual and the environment, and the affordance relation, Lie group cognition is the adaptive continuous cognitive theory that is formed by using the theory of Lie group, including the model library, arithmetic library, and evaluation system. Its features include (1) global and local unity, (2) qualitative and quantitative unity, (3) uniformity of invariant and variable features, (4) multi-parameter and single-parameter uniformity, (5) linear and nonlinear unity, and (6) unity of excitement and inhibition [29–34].

From the definition of Lie group cognition, it is not difficult to see that Lie group cognition includes topological cognition, which also includes Marr's theory and the interaction activation theory proposed by McClelland.

16.4 Neuromorphic synergy learning theoretical framework

Neuromorphic synergy learning is an important research content of embodied cognition. Since the 1980s, embodied cognition has emerged as an important concept in nearly all fields of cognitive science. Embodied cognition emphasises the dependence of the mind or cognition on the body and its sensory motor system. Unlike the traditional cognitive psychology symbol processing mode, it is assumed that the structure and status of the body, physical properties of the body and brain, and the body's special feelings influences cognitive function. Cognition is both involved and embedded.

The brain is embedded in the body and the body is embedded in the environment, which forms the body's cognitive system.

Definition 16.5. Neuromorphic Synergy Learning: This is achieved through the study of bidirectional perception and the cognitive mechanism, the corresponding relations between the individual and the environment, the affordance relation, and by using Lie group theory, topology, fibre bundle, category, and setup, including a model library, algorithm library, and an evaluation system of the machine learning method. It includes symbol ground learning, bidirectional synergy learning, and affordance learning [29–34].

The concept of **deep learning** originates from the research of artificial neural networks. A multilayer perceptron with multiple hidden layers is a deep learning structure. Deep learning represents the distributed characteristics of data by combining low-level features to form more abstract high-level representations of attribute categories or features. The concept of deep learning was proposed in 2006 by Hinton et al. Based on the deep belief network (DBN), an unsupervised greedy layer-wise training algorithm has been proposed. To solve optimisation problems of deep structure, the deep structure of a multi-layer auto encoder has been proposed. In addition, the convolutional neural network proposed by Le Cun et al. is the first real multi-layer structure learning algorithm that uses the space relative relation to reduce the number of parameters in order to improve the training performance. Deep learning is a method of representing learning based on data in machine learning. Observations (such as an image) can be represented in a variety of ways, such as vectors for each pixel's strength value, or they can be further abstracted into a range of edges, specific shapes, etc. It is easier to learn a task from an instance using some specific presentation methods (e.g., face recognition or facial expression recognition). The advantage of deep learning is that the manual acquisition feature is replaced with non-supervised or semi-supervised feature learning and hierarchical feature extraction. Deep learning is a new field in the study of machine learning. Its objective is to establish and simulate the human brain's analysis neural network for learning. It mimics the human brain to explain the mechanism of data, such as images, sound, and text. As with machine learning methods, deep machine learning methods include supervised learning and unsupervised learning. Different learning models are established under different learning frameworks. For example, a convolutional neural network (CNN) is a deep supervised machine learning model, while a deep belief network (DBN) is an unsupervised machine learning model.

It can be seen from above that deep learning can only show a bidirectional mechanism and symbol grounding relationship between perception and cognition; it is difficult to express the affordance relationship. Thus, neuromorphic synergy learning includes deep learning. In other words, deep learning is a neuromorphic synergy learning method.

16.4.1 Symbol grounding learning

The symbol grounding relation problem is a core scientific problem of neuromorphic synergy learning and visual cognition. In general, there are three types of approximate solutions for solving corresponding problems. The first method is to manually create a symbol system and establish a correspondence between the language and the real world, which is a correspondence between a predetermined set of behaviour spaces and an environment feature set. More specifically, the agent can at least get the significant characteristics of the sensory data set, abstract these characteristics, identify the classes and concepts, and use these representations to associate symbols. This type of method uses the structure of space language but usually does not involve study or perceptual feedback. Furthermore, the behaviour space is invariable. The second method involves learning the meaning of words in the robot's perceptual motor space, such as joint angles and images. These systems use the language vocabulary as a perceptual input, learning directly from complex features, thereby acquiring a limited set of commands that the robot can understand. The third method is to use learning to map from the language to environmental objects.

Definition 16.6. Symbol Grounding Learning: This is the study of the corresponding relations between the individual and the environment, by using Lie groups, topology, fibre bundle, category theory, and the setup, including the model library, algorithm library, and the evaluation system of the machine learning method. The main contents of symbol grounding learning are as follows:

(1) Assumption of symbol grounding learning

The symbol grounding relation problem is a core scientific problem of neuromorphic synergy learning. The difficulty of the symbol grounding relation problem is not only in associating symbols with objects but also in focusing on the semantics of symbols when the ontology interacts with the outside world. This means that symbolic interpretation must be intrinsic to the symbol system itself. For example, we can describe zero-semantic conventions as follows: (1) it is not allowed to have any form of innate ideas; no semantic source can be predicted to be stored in the ontology; (2) it is not allowed to have any form of phenomenon; and (3) ontologies should have their own capabilities and resources, such as computing, sensing, and using sensors so that they can be associated with symbols. Any method that violates this condition is in violation of the semantics and cannot solve the corresponding problem.

(2) Representation methods of symbol grounding learning

For example, for robots to be able to work together with people, they must understand the corresponding relationship between the content of learning and the external environment. Therefore, by using Lie group connectivity to study the symbol grounding

relation, the Lie group symbol grounding relation representation theory framework of neuromorphic synergy learning is established.

(3) Symbol grounding learning algorithms

For the symbol grounding relation features, by using the symplectic group learning algorithm, quantum group learning algorithm, and deep cognitive complementary model of ‘model + analysis’, the ‘model + analysis’ deep symbol grounding learning theory and algorithm framework of neuromorphic synergy learning are established.

16.4.2 Bidirectional synergy learning

Vision and cognition are not two separate systems. The visual system is an input system that extracts information from the environment; the cognitive system is a central processing system that processes information transmitted by the input system (such as speech, memory, and thinking). Cognition is also intellectual in nature, and it shares a common system with both the intuitive and the neurological levels. The visual state includes unconscious neural representation of the external stimuli and the conscious experience of choice. Once the visual state is produced, some of the content is extracted and stored in long memory by selective attention. In the subsequent recovery, this memory can play a symbolic role in representing the outside world and enter the symbolic operation process. A cognitive representation is formed when a large number of perceptual symbols are concentrated. The conformance relation of the recently formed perception and cognition directly leads to the generation of perceptual symbol theory. Cosy found that the one-way hypothesis is not only inconsistent with the perceptual cognition principle of a human being but also one of the main bottlenecks that prevent the individual layer’s collaborative computing technology from achieving the fundamental breakthrough. The ‘two-way hypothesis’ has replaced the ‘one-way hypothesis’ in order to overcome the bottleneck. Thus, the bi-directionality mechanism problem is a core scientific problem of neuromorphic synergy learning.

Definition 16.7. Bidirectional Synergy Learning is the study of the bi-directionality mechanism of perception and cognition and machine learning methods, including the model library, arithmetic library, and evaluation system, which are established by Lie group, topology, fibre bundle, category, and so on.

(1) Information acquisition theory and technology of bidirectional synergy learning

By focusing on the bidirectional synergy mechanism features of visual cognition and by using the binary relationship of the Lie group, the theory and technology of visual cognitive bidirectional synergy learning information acquisition are established, and the results are verified and implemented on the learning robot verification platform.

(2) Information feedback theory and technology of bidirectional synergy learning

By focusing on the bidirectional synergy information feedback mechanism, features of visual cognition, and by using the hypothesis axioms of solving the symbol grounding relation and the Lie group learning techniques of neuromorphic synergy learning, the theory and technology of visual cognitive bidirectional synergy learning information feedback is established, and the results are verified and implemented on the learning robot verification platform.

(3) Enhance learning theory and technology of bidirectional synergy learning

It is mainly aimed at the bidirectional synergy of visual cognition, using symbol grounding learning methods of neuromorphic synergy learning, to enhance the bidirectional learning function and establishing enhanced learning theory and technology of bidirectional synergy learning. The results are verified and implemented on the learning robot verification platform.

16.4.3 Affordance learning

The affordance relation is the most central concept in perceptual psychology and ecological psychology. It is generally believed that the affordance relation is an interaction between an individual and the environment, and the property of the environment makes it possible for individuals to carry out certain behaviours. There are three main points: (1) affordance is a potential possibility; (2) affordance is an interaction between the individual and the environment; and (3) affordance extraction is restricted by intention. Affordance includes the individual, environment, individual attributes, and relationship between properties. There is no complete theoretical system.

Definition 16.8. Affordance Learning: It is the study of the relationship between the individual and the environment and machine learning methods, including the model library, arithmetic library, and the evaluation system, which are established by Lie group, topology, fibre bundle, category, and so on. The main contents of affordance learning are as follows:

(1) Possible affordance learning methods

Based on the possible features of affordance learning, dynamic fuzzy logic, dynamic fuzzy relation, and dynamic fuzzy determination theory are used to study the possible affordance method of visual cognition.

(2) Interactive affordance learning technology

Based on the interactive features of affordance learning, the interactive method and topology theory of affine contact are used to study the interactive and interactive technology of visual cognition.

(3) Constrained affordance learning algorithm

To grasp the affordance relationship of affordance learning, by using the theory of Lie group relationship, the constraint affordance learning algorithm of visual cognition is studied, and the results are verified and implemented on the learning robot verification platform.

Reinforcement learning is the learning of the intelligent system from the environment to behaviour mapping to maximise the value of the reward signal (reinforcement signal) function. Reinforcement learning is different from supervised learning in connectionism; the main performance is in the teacher's signal. Reinforcement learning provided by the environment is a type of evaluation (usually a scalar signal) of the resulting action, instead of telling the reinforcement learning system how to produce the right action. Because the external environment provides little information, the RLS must learn from its own experience. In this way, the RLS obtains knowledge in the action-evaluation environment and improves the action plan to adapt to the environment.

The basic model and principle of reinforcement learning are as follows. Reinforcement learning is developed from the theory of animal learning, parameter perturbation, and adaptive control. The basic principle is as follows: If an agent's behaviour strategy leads to an environmentally positive reward (enhanced signal), then the trend of the agent to generate this behaviour strategy will be strengthened. The goal of the agent is to find the optimal strategy in each discrete state to maximise desired discounts and rewards. Reinforcement learning considers learning as a testing process. The agent selects an action for the environment, and the environment changes after the action. At the same time, a reinforcement signal (award or punishment) is given to the agent, and the agent selects the next action according to the strengthening signal and the current state of the environment. The principle of choice is to increase the probability of positive reinforcement. The action of choice not only affects the immediate reinforcement value but also affects the state of the next moment in the environment and the final reinforcement value.

In conclusion, it is not difficult to see that the relationship between affordance learning and reinforcement learning is that reinforcement learning is a type of affordance learning. Affordance learning includes reinforcement learning, and the problems of reinforcement learning can be solved by the methods of affordance learning, but not vice versa.

16.4.4 Multi-scale synergy learning

The visual cognitive multiscale technique of neuromorphic synergy learning is a new method of visual cognitive information processing. When one looks at an object with one's eyes, the distance between the object and oneself (distance is a scale parameter) constantly changes. The retina will perceive the changing image information,

and analyse and synthesise these different scales of visual information to obtain the essential characteristics of the observed object. This visual analysis method is called multi-scale vision technology. In summary, multi-scale technology is a visual analysis of the consistency of sampling at different scales.

The visual cognitive multiscale technique originated from the Gaussian filter in image processing and the Gaussian filtering model for the solution of the thermal diffusion equation. Thus, there will be visual cognitive information processing of multi-scale technology linked with partial differential equations, and various multi-scale techniques are gradually developed. The four most basic multi-scale technologies are linear multiscale technology, nonlinear multi-scale technology, multi-scale technology, and mathematical morphology multi-scale technology. These four basic multi-scale technologies constitute the bulk of the multi-scale concept. At present, many international multi-scale analytical methods belong to one of these four multi-scale technologies. For example, the Mumford–Shah functional or active contour model can be converted into nonlinear partial differential equations, which can be regarded as a nonlinear multi-scale technology. As a curve evolution algorithm, the algorithm can be considered as a type of multi-scale technique.

At present, numerous European and American scholars are committed to the multi-scale theory and its applications in this field. Representative examples are as follows. Morel, Guichard et al. carried out a large number of basic studies on the theory of partial differential equations with multiple scales. Sapiro et al. made remarkable progress in the method of geometrical partial differential equations. Chan et al. obtained additional research results in the method of partial differential equations for image processing. Domestic research in this field only began recently. Since the 1990s, researchers in the field of engineering and mathematics have reported on the active contour method, the Mumford–Shah variation method, the snake model, and PDE image processing.

The multi-scale feature is a common characteristic in the cognitive environment of neuromorphic synergy learning, e.g., multi-scale characteristics of information, knowledge, dynamic learning, and corresponding relationships.

Definition 16.9. Multi-scale Synergy Learning: This is the study of the multi-scale features that are ubiquitous in the cognitive environment of neuromorphic synergy learning and machine learning methods, including the model library, arithmetic library, and evaluation system, which are established by Lie group, topology, fibre bundle, category, and so on. The main contents of multi-scale synergy learning are as follows:

(1) Multi-scale information computing learning theory of neuromorphic synergy learning

For the multi-scale information features of neuromorphic synergy learning environment, using the multi-group theory, the multi-scale information computing learning

theory of neuromorphic synergy learning is established, and the results are verified and implemented on the learning robot verification platform.

(2) Multi-scale affordance computing learning technology of neuromorphic synergy learning

Based on the multi-scale feature of the affordance environment, the dynamic fuzzy determination theory is used to establish the multi-scale affordance computing technology of neuromorphic synergy learning, and the results are verified and implemented on the learning robot verification platform.

(3) Multi-scale symbol grounding computing learning method of neuromorphic synergy learning

For the multi-scale features of the symbol grounding relation, by using the deep cognitive complementary model of ‘model + analysis’, the multi-scale symbol grounding computing learning method of neuromorphic synergy learning is established, and the results are verified and implemented on the learning robot verification platform.

16.5 Design of a neuromorphic synergy learning verification platform

The contents are as follows:

(1) Evaluation system of the neuromorphic synergy learning verification platform

Combined with the theoretical analysis results, through user research, statistical analysis, model evaluation, and other means, the evaluation system of the neuromorphic synergy learning verification platform is established as the main basis for the performance and demonstration application of the test evaluation platform.

(2) Designing scheme of the neuromorphic synergy learning verification platform

The affordance perceptual organisation is built. The dialogue process, planning process, and learning process can be controlled by meta-level reasoning to achieve the synergy operation of these processes. The cooperative learning architecture of the robot is realised by two aspects: affordance perceptual organisation and meta-level synergy control.

(3) Implementation technique of the neuromorphic synergy learning verification platform

According to the platform design scheme, the relevant theories and technological achievements are transformed into efficient and practical algorithms. These algo-

rithms and other necessary supporting functions will be constructed to form a class-oriented cooperative learning perception analysis and control platform for learning robots, including all hardware, software, and modules.

16.6 Summary

This chapter mainly focused on the three core scientific problems in neuromorphic synergy learning, i.e., the symbol grounding relation, bidirectional mechanism, and affordance learning, and studied neuromorphic synergy learning from four levels, namely proposing a new theory, exploring a new method, initiating a new technique, and developing a new platform that includes the Lie group cognitive theory framework, neuromorphic synergy learning framework, symbol grounding learning, bidirectional synergy learning, and affordance learning.

Bibliography

- [1] Xu B, Liu C, Zeng Y. The present situation and development of brain intelligent research. *Bulletin of the Chinese Academy of Sciences*, 2016, 07, 793–802.
- [2] Laird JE, Newell A, Rosenbloom PS. SOAR: An architecture for general intelligence. *Artificial Intelligence*, 1987, 33(1), 1–64.
- [3] Newell A. *Unified Theories of Cognition*. Cambridge, USA: Harvard University Press, 1990.
- [4] Marr D, Cowan JD. A theory for cerebral neocortex. From the Retina to the Neocortex. Birkhäuser Boston, 1991, 161–234.
- [5] Marr D. Simple Memory: A Theory for Archicortex. *Philosophical Transactions of the Royal Society of London*, 1971, 262(841), 23–81.
- [6] Marr D. Approaches to Biological Information Processing. *Science*, 1975, 190(4217), 875–876.
- [7] Marr D, Poggio T. Cooperative Computation of Stereo Disparity. *Science*, 1976, 194(4262), 283–287.
- [8] McCulloch WS, Pitts WH. A logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 1942, 5, 115–133.
- [9] Werbos P. Beyond regression: new tools for prediction and analysis in the behavioral sciences. *Journal of Shandong Institute of Commerce & Technology*, 1974, 29(18), 65–78.
- [10] Hinton GE, Osindero S, Teh YW. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 1960, 18(7), 1527–1554.
- [11] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors.// *Neurocomputing: foundations of research*. MIT Press, 1986, 533–536.
- [12] Le Cun YL, Boser B, Denker JS, et al. Handwritten Digit Recognition with a Back-Propagation Network. *Advances in Neural Information Processing Systems*, 1997, 2, 396–404.
- [13] Hawkins J. *On Intelligence*. New York, USA: Times Books, 2004.
- [14] George D, Hawkins J. Towards a mathematical theory of cortical micro-circuits. *Plos Computational Biology*, 2009, 5(10), e1000532–e1000532.
- [15] Riesenhuber M, Poggio T. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 1999, 2(11), 1019–1025.
- [16] Serre T, Oliva A, Poggio T. A feedforward architecture accounts for rapid categorization. *Proceedings of the National Academy of Sciences*, 2007, 104(15), 6424–6429.

- [17] Markram H, Meier K, et al. The Human Brain Project: A Report to the European Commission. Technical Report, 2012.
- [18] Le QV. Building high-level features using large scale unsupervised learning, 2011, 8595–8598.
- [19] Kai Y, Lei J, Chen Y, et al. Deep Learning: Yesterday, Today, and Tomorrow. *Journal of Computer Research & Development*, 2013, 20(6), 1349.
- [20] Marr D. Vision: A Computational Investigation into the Human Representation and Processing of Visual Information, 1982, San Francisco, USA: Freeman.
- [21] Biederman I. Recognition by Components: A Theory of Human Image Understanding. *Psychological Review*, 1987, 94(2), 115–147.
- [22] Treisman A, Gelade G. A Feature Integration Theory of Attention. *Cognitive Psychology*, 1980, 12(1), 97–136.
- [23] Chen L. Topological structure in visual perception. *Science*, 1982, 218, 699–700.
- [24] McClelland JL, Rumelhart DE. An Interactive Activation Model of Context Effects in Letter Perception: Part I. An Account of Basic Findings. *Psychological Review*, 1981, 88(5), 375–407.
- [25] Cortes C, Vapnik V. Support-Vector Networks. *Machine Learning*, 1995, 20(3), 273–297.
- [26] Yi Z, Chenglin L, Tieniu T. Retrospect and outlook of brain-inspired intelligence research. *Chinese Journal of Computers*, 2016(1), 212–222.
- [27] Liwei W, Li X, Ping X, Hanchao Y, Minghui K, Yi S, Yongqing Z. Brain science and brain-like intelligence research in Chinese academy of sciences. *Bulletin of Chinese Academy of Sciences*, 2016, 07, 747–754.
- [28] Chapman AH, Soares MAR. The Computer and the Brain. The computer and the brain. Yale University Press, 1958, 161–163.
- [29] Li FZ, Li Z, Jiwen Y, Xupei Q, Bangjun W, Shuping H. Lie group machine learning. Hefei: China University of Science and Technology press, 2013.
- [30] Li FZ, Jia Z, Baojiang Z, Li Z, Jianxi F, Zhao Z. Study on learning theory and calculation method of cognitive image features. Second prize of Jiangsu Science and Technology award. The People's Government of Jiangsu Province, 2014.
- [31] Li FZ. In Recognition of His Pioneer work in Exploring Granular Computing on Lie group and Machine learning, Pioneer Award, IEEE CS Granular Computing Steering Committee, 2014.
- [32] Mengduo Y, Li FZ, Li Z. Advances in the study of Lie group machine learning in recent ten years. *Chinese Journal of Computers*, 2015, 38(7), 1337–1356.
- [33] Li FZ. Neuromorphic synergy learning. Academic report. Nanjing Normal University, 2016, 4.
- [34] Li FZ. Neuromorphic synergy learning theories and methods. Academic report. Soochow University, 2015, 9.

17 Appendix

17.1 Topological group

Definition 17.1. A topological group is a set G with the following properties:

- (1) G is a group;
- (2) G is a Hausdorff space;
- (3) the mapping $(x, y) \rightarrow xy^{-1}$ of $G \times G$ to G is connected;

Property (3) represents the group structure compatible with the topological structure of G . It can be written as follows:

- (4) $(x, y) \rightarrow xy^{-1}$, $x \rightarrow x^{-1}$ are the continuous mappings of $G \times G$ to G , G to G , respectively.

Definition 17.2. Let G be a topological group. For any $g \in G$, called the mapping of G to G ,

$$L_g: x \rightarrow gx, \quad \forall x \in G,$$

$$R_g: x \rightarrow xg, \quad \forall x \in G,$$

respectively, are the left transformation and right transformation of G .

Obviously, the following four properties are established.

$$(L_g)^{-1} = L_{g^{-1}}, \quad (R_g)^{-1} = R_{g^{-1}}.$$

L_g , R_g , and mapping $x \rightarrow x^{-1}$ are all homeomorphic mappings of G .

$adg = L_g R_{g^{-1}}$ is the self-homeomorphic mapping of G , and $\forall x \in G$, $adg(x) = gxg^{-1}$.

For any natural number n , the mapping $(x_1, x_2, \dots, x_n) \rightarrow x_1 x_2 \dots x_n$ is the continuous mapping of $\underbrace{G \times G \times \dots \times G}_n$.

Theorem 17.1. Let e be the unit of topological group G . Then, there is the basic neighbourhood group U of e , which fits the following conditions:

- (1) $V_1, V_2 \in U$, then $V_1 \cap V_2 \in U$;
- (2) $V \in U$, while $W \supset V$, then $W \in U$;
- (3) $\bigcap_{V \in U} V = \{e\}$;
- (4) $\forall V \in U, \exists V_1 \in U$, making $V_1 V_1^{-1} \subseteq V$;
- (5) $\forall V \in U, a \in G$, then $aVa^{-1} \in U$.

By contrast, given a group G and a set of U containing e satisfying the above five conditions, G can be uniquely defined as a topological group. Let U be the basic neighbourhood group of e .

Let G be a topological group, H be a subgroup, and o be the open family of G . Then, the subset family $O_H = \{O \cap H | O \in o\}$ of H satisfies the open set theorem. Thus, introduce the topology in H , becoming the induced topology of topological group G in H . It is easy to prove, with the topology on it, that H is a topological group.

Definition 17.3. If the group φ is isomorphic for the topological groups G_1 and G_2 with the continuous mapping of the topological space, then φ is a homomorphic mapping. If φ is a homogeneous group and a homeomorphic mapping, then φ is called an isomorphic mapping. The above two cases are referred to as homomorphism and isomorphism.

Theorem 17.2. Let φ be the homomorphism of the topological group G_1 to G_2 . The homomorphism of the topological group G_1 is continuous at the unit.

Proof. Let V_2 be the neighbourhood of the unit e_2 of G_2 . Since φ is continuous at e_1 and $\varphi(e_1) = e_2$, there is a neighbourhood V_1 of e making $\varphi(V_1) \subseteq (V_2)$. Therefore, for any $a \in G_1$, the neighbourhood of $\varphi(a)$ is $V_2\varphi(a)$. Thus, the neighbourhood V_1a of a makes $\varphi(V_1a) = \varphi(V_1)\varphi(a) \subseteq (V_2)\varphi(a)$, i.e., φ continuous at a . Hence, φ is continuous, i.e., φ is the homology of the topological group. \square

Theorem 17.3. Connected topological group G is generated by any unit neighbourhood V , and

$$G = \bigcap_{n=1}^{\infty} V^n .$$

Proof. Since V is a unit neighbourhood, there is a unit of an open neighbourhood $V_1 \subset V$. Hence, $V_2 = V_1 \cap V_1^{-1}$ also is an open neighbourhood and $V_2^{-2} = V_2$. Let G_1 be the subgroup of G generated by V_2 . Hence, there is

$$G_1 = \bigcup_{n=1}^{\infty} V_2^n \subseteq \bigcup_{n=1}^{\infty} V_1^n \subseteq \bigcup_{n=1}^{\infty} V^n \subseteq G$$

and V_2 is an open set. Thus, G_1 is an open set. G_1 is a closed set of G , and G is connected. Hence, $G = G_1$ and the theorem is established. \square

Definition 17.4. If the unit neighbourhood V , V' of topological group G , G' is a topological isomorphism as a local group, then G and G' are topologically isomorphic.

Obviously, an isomorphic topological group must be locally isomorphic. By contrast, otherwise it is not.

Theorem 17.4. Let G be a group, V be a subset of the unit e , V generate G , G be multiplied, and V be a connected local group. Then, the only way to define G as a Topological Group is that its induced topology on V is the original topology, its group structure is the original group structure, and G is connected.

Proof. Suppose that v is the basic neighbourhood group of V including e . Let

$$\omega = \{W \subseteq G | W \cap V \in v\} .$$

Obviously, ω meets the conditions (1)–(3) in Theorem 17.1.

Let $W \in \omega$; then $W \cap V \in \nu$. V is a local group; hence, $V_1 \in \nu$ makes $V_1 V_1^{-1} \subseteq \omega$ and $V_1 \in \omega$. Condition (4) in Theorem 17.1 is established. The following proves that condition (5) in Theorem 17.1 is also established. For this reason, we firstly prove the following proposition:

Let $W \in \omega$, $b \in V$. There is $U \in \nu$ making $bUb^{-1} \subseteq W$.

In fact, $(W \cap V)b \cap V$ is a neighbourhood of b connected by $eb = b$. Hence, $U' \in U$ makes $U'b \subseteq (W \cap V)b \cap V$, and $be = b$ is continuous. Therefore, $U \in \nu$ makes $bU \subseteq U'b$ and $bUb^{-1} \subseteq (U'b)b^{-1} \subseteq (W \cap V)bb^{-1} = W \cap V \subseteq W$.

Now, set $a \in G$, $W \in \omega$. For V , generate G , so that $a = a_1 a_2 \dots a_k$, $a_i \in V \cap V^{-1}$. $U_1 \in U$ makes $a_1 U a_1^{-1} \subseteq W$, and $U_2 \in U$ makes $a_2 U a_2^{-1} \subseteq U_1$. In general, there is $U_i \in \nu$ making $a_i U a_i^{-1} \subseteq U_{i-1}$ ($2 \leq i \leq k$). Hence,

$$a_k U a_k^{-1} = a_1 a_2 \dots a_k U_k a_k^{-1} \dots a_2^{-1} a_1^{-1} \subseteq W.$$

i.e., $U_k \subseteq a^{-1} Wa$. Hence, there is $a^{-1} Wa \in \omega$, i.e., W meets condition (5) in Theorem 17.1. Then, G can be defined as a topological group, W is a basic neighbourhood group of unit e , $W \supseteq U$, and $W \in \omega$, then $W \cap V \in \nu$. Thus, this topology in V induces the original topology since V is the unit neighbourhood,

By Theorem 17.2, we know that $G = \bigcup_{n=1}^{\infty} V^n$. Connected by V , V^n is the continuous mapping of the connected set $\underbrace{V \times V \times \dots \times V}_{n \text{ times}}$ total n . Hence, V^n is connected, and $e \in V^n$, so G is connected. \square

Definition 17.5. The topological space m is called a homogeneous space if there is a reversible effect on the continuous transformation group G , i.e., so that $g \in G$ makes $g(x) = y$. We mainly consider the case that m is a Hausdorff space.

Theorem 17.5. Let G be a continuous transformation group on the Hausdorff space. Then, the following result holds.

- (1) $\forall x \in M$, H_x is a closed subgroup of G .
- (2) $y \in G(x)$ if and only if H_x and H_y are conjugate, and if $y = g_0(x)$, then $H_y = g_0 H_x g_0^{-1}$.
- (3) m is the intersection of some interchangeable tracks, i.e.,

$$M = \bigcup_{\alpha} G(x_{\alpha}),$$

Here, when $G(x_{\alpha}) \neq G(x_{\beta})$, $G(x_{\alpha}) \cap G(x_{\beta}) = \emptyset$.

Proof. Since mapping $g \rightarrow g(x)$ is a continuous mapping of G to M , $\{x\}$ is a closed set of M . Hence, the original H_x is a closed set in G , which is a closed subgroup.

Let $g_0(x) = y$. Then, $g \in H_y$ if and only if $g(y) = y$, i.e., $g(g_0(x)) = g_0(x)$. Hence, $g_0^{-1} g g_0 \in H_x$, and (2) is established.

Define relationship \sim in m : $x \sim y$ if there is $g \in G$ making $y = g(x)$. Obviously, this is the equivalence relationship. The equivalence class of x is $G(x)$, so (3) is established. \square

17.2 Concept of differential geometry

Let M be an n -dimensional differential manifold (C^∞ manifold). Let $F(M)$ denote the set of differentiable functions on M . Naturally, we can see $F(M)$ as the combined algebra on the real field.

Let $D^1(M)$ denote the set of all differentiable vector fields of M , i.e., $\forall X \in D^1(M)$ is the exchange of $F(M)$, and satisfies the following two conditions:

$$X(\alpha f + \beta g) = \alpha X(f) + \beta X(g), \quad \forall \alpha, \beta \in \mathbf{R}; \quad f, g \in F(M)$$
$$X(fg) = f \cdot X(g) + g \cdot X(f),$$

where X is the derivative of $F(M)$.

In $D^1(M)$, we can define the addition and product of $D^1(M)$ and elements in the real field \mathbb{R} . $D^1(M)$ becomes a linear space on \mathbb{R} . If we define the blanket product in $D^1(M)$

$$[x, y] = XY - YX, \quad \forall x, y \in D^1(M),$$

then $D^1(M)$ is a Lie algebra on \mathbb{R} .

$D^1(M)$ can also be regarded as the left-hand side of $F(M)$, so we can consider the elements in its dual mode $D_1(M)$, where $D_1(M)$ is a differential form. Starting from $F(M)$, $D^1(M)$, and $D_1(M)$, we can construct a mixed tensor algebra on M , denoted by $D(M)$.

From the viewpoint of geometry, if M_p is the tangent space of M at a point, then the value X_p of the element X at point p in $D^1(M)$ is the tangent vector of M at point p , i.e., $X_p \in M_p$.

After taking the coordinate system in manifold M , the elements in $D^1(M)$ can be expressed by the analytic expression of the coordinates. Of course, the elements in $D^1(M)$ can also be expressed using the analytic expression of the coordinates.

Definition 17.6. An affine coupling ∇ of M is a mapping of $D^1(M) \times D_1(M)$ to $D^1(M)$, generally denoted by

$$\nabla(X, Y) = \nabla_X(Y),$$

meeting the following three conditions

- (1) ∇ is bilinear;
- (2) $\forall X, Y \in D^1(M), \nabla_{fX}(Y) = f\nabla_X(Y), f \in F(M);$
- (3) $\forall f \in F(M), X, Y \in D^1(M), \nabla_X(fY) = X(f) \cdot Y + f\nabla_X(Y).$

For a fixed $X \in D^1(M)$, ∇_X is a linear mapping of $D^1(M)$ to $D_1(M)$. However, ∇_X is not a modal homomorphism of the left $F(M)$ modes $D^1(M)$. We call $\nabla_X(Y)$ as the covariance differential of Y in the X direction.

Definition 17.7. Suppose that ∇ is an affine linkage of differential manifolds, called the mapping Tor of $D^1(M) \times D_1(M)$ to $D^1(M)$:

$$\text{Tor}(X, Y) = \nabla_X Y - \nabla_Y X - [X, Y]$$

For the torture operation of ∇ about M , the mapping T of $D_1(M) \times D^1(M) \times D^1(M)$ to $F(M)$ is

$$T(\omega, X, Y) = \omega(\text{Tor}(X, Y))$$

For the tortuous tensor of ∇ about M (this is a $(1, 2)$ -type tensor).

It is also known as the mapping R in $D^1(M) \times D^1(M)$ to $\{A|A: D^1(M) \rightarrow D^1(M)\}$:

$$R(X, Y) = \nabla_X \nabla_Y - \nabla_Y \nabla_X - \nabla_{[X, Y]}$$

This is the curvature operation of M about ∇ , and the mapping R of $D_1(M) \times D^1(M) \times D^1(M) \times D^1(M)$ to $F(M)$

$$R(\omega, X, Y, Z) = \omega(R(X, Y), Z)$$

is the curvature tensor of M about ∇ (it is a $(1, 3)$ -type tensor).

Definition 17.8. Let $Y(t)$ be a tangent vector of manifold M at point $y(t)$, i.e., $Y(t) \in T_{y(t)}(M)$. If $Y(t)$ is differentiable and relies on t , then $Y(t)$ is a vector field defined on $y(t)$.

Since $Y \in D^1(M)$ makes that the limit of Y on $y(t)$ to be $Y(t)$, then there is $Y|_{y(t)} = Y(t)$ ($t \in I$). In particular, for $\dot{y}(t)$, there is $X \in D^1(M)$ making $X_{y(t_1)} = y(t)$. Now, denote

$$\nabla_{y(t)} Y(t) = (\nabla_X Y)|_{y(t)} .$$

If the vector field $Y(t)$ defined on y satisfies

$$\nabla_{\dot{y}(t)} Y(t) = 0 , \quad \forall t \in I ,$$

then $Y(t)$ is parallel to y .

From the theory of differential equations, we can see that the vector field parallel to y constitutes an n -dimensional vector space denoted by $P(M, y)$. If $t_1 \in I$, then the mapping $E(t_1)$ of $P(M, y)$ to $M_{y(t_1)} = T_{y(t_1)}(M)$:

$$E(t_1)(X) = X_{y(t_1)} = X(t_1) ,$$

is the isomorphism of the linear space.

Definition 17.9. For $t_1, t_2 \in I$, define the mapping $\parallel_{y(t_1, t_2)}$ of $M_{y(t_1)}$ to $M_{y(t_2)}$:

$$\parallel_{y(t_1, t_2)} = E(t_2) \cdot E(t_1)^{-1}$$

We say that M makes parallel movement along y from $y(t_1)$ to $y(t_2)$.

In other words, $\forall Y_1 \in M_{y(t_1)}$, $Y(t)$ defined on y , and $Y(t_1) = Y$, $\nabla_{\dot{y}(t)} Y(t) = 0$, then $\parallel_{y(t_1, t_2)} Y_1 = Y(t_2)$.

Definition 17.10. If curve $y(t)$ in manifold M satisfies $\nabla_{\dot{y}(t)} \dot{y}(t) = 0$, then it is called a geodesic. It is a particularly important curve in the manifold.

17.3 Manifold learning algorithm

Manifold learning is a machine learning paradigm based on topological and differential geometry theory, which can be described in mathematical language as follows. Let $Y \subset R^d$ and $f: Y \rightarrow R^D$ be a smooth map, $D > d$. The goal of manifold learning is based on a given set of observed data $\{x_i\}$ on R^D to recover Y , and f in the Y hidden data $\{y_i\}$ is randomly generated, and then mapped to the observation space, so that $\{x_i = F(y_i)\}$. Manifold learning mainly includes the following classical algorithms.

17.3.1 Local linear embedding (LLE)

According to the LLE algorithm, in the local sense, the structure of the data is linear or the local point of the point in a super-plane. Hence, we can use its linear combination of adjacent points for representation. The three-dimensional data of B in Figure 17.1 is sampled from the two-dimensional manifold in A, and the colour band shows the neighbourhood of the reserved band after the LLE mapping. B and C are shown by black circles out of the single point of the neighbourhood part.

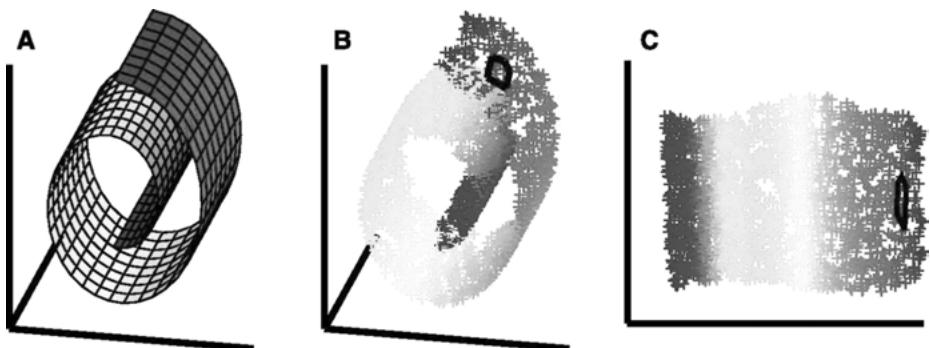


Fig. 17.1: The basic idea of LLE.

The LLE algorithm is described as follows:

Algorithm 17.1: LLE algorithm

Input: Sample input $X = \{x_1, x_2, \dots, x_n \in \mathbb{R}^D\}$, sample true dimension d , neighbourhood parameter k
Output: low-dimensional embedding $Y = \{y_1, y_2, \dots, y_n \in \mathbb{R}^d\}$

- (1) Calculate the nearest neighbour of each point (using K nearest neighbour or ε sphere neighbourhood).
- (2) Calculate the weights $w_{ij} = \sum_k G_{jk}^{i-1} / \sum_m G_{lm}^{i-1}$, where $G_{jk}^i = (x_i - \eta_j) \cdot (x_i - \eta_l)$, η_j, η_l is the x_i neighbouring point. Let w_{ij} minimise the error in x_i with its K nearest neighbours, i.e., by minimising $\|X_i - W_{ij}X_j\|$ to find w_{ij} .

- (3) Keep the weight w_{ij} unchanged and find x_i in the quadrant of the low-dimensional space y_i , making the low-dimensional reconstruction error minimum.

Minimise the embedded cost function as $\Phi(Y) = \sum_i |Y_i - \sum w_{ij} Y_j|^2$.

$M = (I - W)^T(I - W)$. The low-dimensional embedding is the smallest second to $d + 1$ -th eigenvectors of M .

The LLE algorithm can learn any dimension of the local linear low-dimensional manifold. Only a few parameters are to be determined, namely the coefficients d and k . The neighbour weights of each point in the LLE algorithm are kept constant under the translation, rotation, and telescopic transformation, and the overall optimal solution of the analysis is obtained without the iteration. The LLE algorithm is attributed to sparse matrix eigenvalue calculation; its computational complexity is relatively low and it is easy to implement. However, the LLE algorithm requires learned manifolds that are non-closed and locally localised. The samples are required to be densely sampled on the manifold. The parameters d, k have too many choices and are sensitive to noise in the sample.

17.3.2 Isometric mapping (Isomap)

Isomap is based on the multidimensional scale transform (MDS) and seeks to maintain the intrinsic geometric properties of data points, i.e., to maintain the geodesic distance between two points. It uses the geodesic distance on the manifold instead of the Euclidean distance in the classical MDS method. The core of the algorithm, i.e., the method of estimating the distance between two points is as follows:

- (1) The distance between very close points is replaced by the Euclidean distance;
- (2) The distance between distant points is approximated by the shortest path.

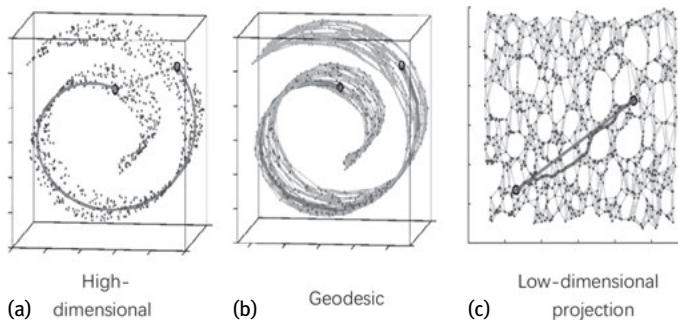


Fig. 17.2: Isomap's basic idea.

As shown in Figure 17.2, the samples in (a) are distributed on the Swiss roll, and the Euclidean distance (shown by the dashed line) between the two points does not represent the actual distance between the two points. The samples in (b) are distributed on the manifold. The curve is the distance between two points. The isomap dimension after the projection results of the two points and two paths (geodesic and short stitching) is shown in (c).

The Isomap algorithm is described below:

Algorithm 17.2: Isomap algorithm

Input: Sample input $X = \{x_1, x_2, \dots, x_n \in \mathbb{R}^D\}$, sample true dimension d , neighbourhood parameter k

Output: low-dimensional embedding $Y = \{y_1, y_2, \dots, y_n \in \mathbb{R}^d\}$

- (1) Calculate the nearest neighbour point of each point (using K nearest neighbour or ε sphere neighbourhood).
- (2) Define a weighted undirected graph on the sample set. If x_i and x_j are close neighbours, the weight of the edge is $d_X(i, j)$.
- (3) Calculate the shortest distance between two points in the graph. Let the distance matrix $D_G = \{d_G(i, j)\}$
- (4) Use MDS to find the low-dimensional embedded manifold. Let

$$S = (S_{ij}) = (D_{ij}^2), \quad H = (H_{ij}) = (\delta_{ij} - 1/N), \quad \tau(D) = -HSH/2,$$

The low-dimensional embedding is the eigenvector corresponding to the second smallest $d + 1$ eigenvalue.

Isomap is characterised by the fact that the algorithm is non-linear and suitable for learning the internal low-dimensional manifolds, but it is not suitable for learning manifolds with larger inner curvatures. The isomap algorithm has two parameters d and k . Like the LLE algorithm, the isomap algorithm also has some obvious defects. It needs to calculate the shortest distance between the two points on the map, and its implementation is relatively slow. It is unstable when used for visualisation, and it requires a larger neighbourhood to produce a short-line phenomenon. The low-dimensional projection results produce a large number of “holes”, or the graphs reconstructed by the shortest path algorithm are not connected, although the overall structure can be guaranteed to be stable after the point projection of the smaller neighbourhood part.

17.3.3 Horizontal linear embedding (HLLE)

The HLLE algorithm can recover the generated coordinates of the manifolds that are locally equidistant from the low-dimensional Euclidean spaces. The algorithm is described as follows:

Algorithm 17.3: HLLE algorithm

Input: Sample input $X = \{x_1, x_2, \dots, x_n \in \mathbb{R}^D\}$, sample true dimension d , neighbourhood parameter k

Output: low-dimensional embedding $Y = \{y_1, y_2, \dots, y_n \in \mathbb{R}^d\}$

- (1) Calculate the nearest neighbour point of each point (using K nearest neighbour or ϵ sphere neighbourhood).
 - (2) Obtain tangent coordinates. For the neighbourhood of each sample point, the largest singular vectors corresponding to the largest d singular values of the centralised matrix $X_i = [x_{i1}, \dots, x_{ik}]$ are calculated, and they are composed of the matrix V_i .
 - (3) Estimate the Hessian Matrix. Let $M_i = [1, V_i, (V_i(:, :s) * V_i(:, :l)) \text{ for } 1 \leq s \leq l \leq d]$, and the Gram–Schmidt orthogonalisation of the matrix M_i yields the column orthogonal matrix \tilde{M}_i . Then, Hessian Matrix $H_i = \tilde{M}_i(:, d+1: 1+d+d(d+1)/2)^T$ is the transpose of the last $d(d+1)/2$ column of \tilde{M}_i .
 - (4) Construct a quad ratic term. The symmetric matrix H is constructed using the sea matrices H_i , $i = 1, \dots, N$ of each neighbourhood, and its elements are $H_{ij} = \sum_{s=1}^N \sum_{l=1}^{d(d+1)/2} (H_i^s)_{il} i (H_j^s)_{lj}$,
 - (5) Calculate the zero space of H . $U = [u_2, \dots, u_{d+1}]$ is the zero vector of all the eigenvalues u_1, \dots, u_d of the smallest $d+1$ eigenvalues of H .
 - (6) Calculate the embedded result. Remember that matrix $R_{ij} = \sum_{l \in J_1} U_{l,i} U_{l,j}$ ($i, j = 1, \dots, d$), where J_1 represents the neighbourhood of a sample point. Then, $T = R^{-1}/U^T$ is the result of the embedding.
-

HLLE does not require the data set to have a convexity assumption, so it has a wider range of applications, but because it needs to calculate the second derivative, the algorithm is not very stable when there is noise.

17.3.4 Laplacian eigenmap

The basic idea of the Laplacian eigenmap is that the image from a very close point in the high-dimensional space is projected close to the low-dimensional space, and finally, the generalised eigenvalue problem of the Laplacian operator is solved. The algorithm is described as follows:

Algorithm 17.4: Laplacian Eigenmap algorithm

Input: Sample input $X = \{x_1, x_2, \dots, x_n \in \mathbb{R}^D\}$, sample true dimension d , neighbourhood parameter k

Output: low-dimensional embedding $Y = \{y_1, y_2, \dots, y_n \in \mathbb{R}^d\}$

- (1) Calculate the nearest neighbour point of each point (using K nearest neighbour or ϵ sphere neighbourhood). Construct a neighbouring graph. The vertices of the graph are the sample points, and the edges are connected by two points.
- (2) Give each side a weight. If the i -th point and the j -th point are not connected, the weight is 0; otherwise, $w_{ij} = 1$.
- (3) The generalised eigenvector of the graph Laplacian operator is calculated and the low-dimensional embedding is obtained.

Let D be the diagonal matrix $D_{ii} = \sum_j W_{ji}$, $L = D - W$, and L be the Laplacian operator on the neighbourhood graph, and solve the generalised eigenvalue problem $Lf = \lambda Df$.

The Laplacian eigenmap is characterised by the following. It is a local non-linear method, and the spectrum theory is very closely related. There are only two parameters k and d in the algorithm, and the global optimal solution is obtained by solving the eigenvalue problem of the sparse matrix. The algorithm makes very close points in the original space too close to the low-dimensional space, so it can be used for clustering.

17.3.5 Local tangency space arrangement (LTSA)

The basic idea of the LTSA algorithm is to use the tangent space of the sample point neighbourhood to represent the local geometric properties and then to arrange the local tangent space to construct the global coordinates of the manifold. The algorithm is described as follows:

Algorithm 17.5: LTSA algorithm

Input: Sample input $X = \{x_1, x_2, \dots, x_n \in \mathbb{R}^D\}$ sample true dimension d , neighbourhood parameter k
Output: low-dimensional embedding $Y = \{y_1, y_2, \dots, y_n \in \mathbb{R}^d\}$

- (1) Calculate the nearest neighbour point of each point (using K nearest neighbour or ε sphere neighbourhood).
- (2) Local linear projection. For the neighbourhood of each sample point, the right singular vector corresponding to the maximum d singular values of the centralised matrix $X_i - \bar{x}_i l_k^T$ is calculated and the d singular vectors are composed of the matrix V_i .
- (3) Local coordinate system arrangement. Construct the permutation matrix $\Phi = \sum_{i=1, \dots, N} S_i W_i W_i^T S_i^T$, and calculate the smallest $d+1$ eigenvalues corresponding to the eigenvector u_1, \dots, u_d . Then, $T = \{u_2, \dots, u_{d+1}\}^T$ is the calculated embedding result.

The LTSA algorithm can effectively study the overall embedding coordinates that represent the low-dimensional manifold structure of the data set. However, the order of the matrix for eigenvalue decomposition in the algorithm is equal to the number of samples, and the sample set cannot be processed when the sample set is large. It handles new sample points effectively.

17.4 Basic concept and nature of symplectic group

Definition 17.11. Let V and W be two vector spaces. Map $f: V \rightarrow W$, so that for any $X, Y \in V, a \in R$, we have

$$f(X + Y) = f(X) + f(Y), \quad f(aX) = af(X)$$

Then, we call f a linear mapping or homomorphism of vector spaces from V to W .

$$\text{Ker } f = \{X \in V | f(X) = 0\}$$

is called the core of linear mapping f .

$$\text{Im } f = \{f(X) | X \in V\}$$

is called the image of linear mapping f .

It is easy to prove that $\text{Ker } f$ and $\text{Im } f$ are subspaces of V and W , respectively. When $\text{Ker } f = 0$, f is said to be single shot. When $\text{Im } f = W$, f is said to be surjective. When the linear mapping is surjective and injective, f is called an isomorphic vector space. If there is an isomorphism f between the vector spaces V and W , it is also known as W and V isomorphism. Consider the mapping

$$a: V_1 \times V_2 \times \cdots \times V_r \rightarrow R \quad \forall X_1, \dots, X_r \in V, a(X_1, \dots, X_r) \in R$$

such that

$$a(X_1, \dots, X_i + Y_i, \dots, X_r) = a(X_1, \dots, X_i, \dots, X_r) + a(X_1, \dots, Y_i, \dots, X_r),$$

$$a(X_1, \dots, aX_i, \dots, X_r) = aa(X_1, \dots, X_i, \dots, X_r)$$

$$a(X_1, \dots, X_i, \dots, X_j, \dots, X_r) = -a(X_1, \dots, X_j, \dots, X_i, \dots, X_r)$$

$$(\forall i, j = 1, \dots, r, a \in R).$$

We call a the r -valued inverse linear function on the vector space V or the r -form on V .

Definition 17.12. Let u and v be vectors in $2n$ -dimensional space, where

$$u = [x_1, \dots, x_n, \xi_1, \dots, \xi_n]^T$$

$$v = [y_1, \dots, y_n, \eta_1, \dots, \eta_n]^T.$$

Then, the inner product is defined as

$$\omega(u, v) = \sum_{i=1}^n (x_i \eta_i - y_i \xi_i) = u^T J v^T \omega(Su, Sv) = \omega(u, v)$$

where $J = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}$, with the properties of $J^{-1} = J^T = -J$, I_n for the n -order unit matrix. The intrinsic properties of the inner product are as follows:

- (1) Bilinear: $\omega(\lambda_1 u_1 + \lambda_2 u_2, v) = \lambda_1 \omega(u_1, v) + \lambda_2 \omega(u_2, v)$;
- (2) Antisymmetry: $\varphi: \text{Fun}(Y) \rightarrow \text{Fun}(X)$
- (3) Non-degenerate: For each of $v \in R^{2n}$, if there are $\omega(u, v) = 0$, then $u = 0$;

Definition 17.13. The real vector space (R^{2n}, ω) of the symplectic product ω having the above condition is called a symplectic space.

Definition 17.14. If the linear transformation S in the symplectic space (R^{2n}, ω) satisfies $S: R^{2n} \rightarrow R^{2n}$, for all $u, v \in R^{2n}$ we have

$$\omega(Su, Sv) = \omega(u, v)$$

Then, the linear transformation S is a symplectic transformation or a regular transformation, and the symplectic transform preserves the two-dimensional product.

Theorem 17.6. A linear transformation of symplectic space S is a necessary and sufficient condition for symplectic transformation. It is

$$S^TJS = J.$$

Proof. In the symplectic space (R^{2n}, ω) , there is a matrix of a symplectic transform S on an octyl group $\{e_1, e_2, \dots, e_n, e_{n+1}, \dots, e_{2n}\}$. (R^{2n}, ω) is denoted by S . Then,

$$e_i \quad \text{and} \quad Y = \sum_{j=1}^{2n} y^j e_j.$$

From $\omega(SX, SY) = \omega(X, Y)$,

$$(x^1, \dots, x^{2n}) S^T JS \begin{pmatrix} y^1 \\ \vdots \\ y^{2n} \end{pmatrix} = (x^1, \dots, x^{2n}) J \begin{pmatrix} y^1 \\ \vdots \\ y^{2n} \end{pmatrix}$$

where $J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}$ is the matrix representation of the symplectic form ω under $\{e_1, e_2, \dots, e_n, e_{n+1}, \dots, e_{2n}\}$, and T is the transpose of the matrix. Hence,

$$S^TJS = J$$

is the full table condition of the symplectic transformation. \square

Algorithm 17.6: Judgment of symplectic transformation algorithm

Input: r and s.

Output: {1, -1}.

Procedure check_Symplectic_transform(r,s)

If $u = [x_1, \dots, x_n, \xi_1, \dots, \xi_n]^T \in r$ and $v = [y_1, \dots, y_n, \eta_1, \dots, \eta_n]^T \in r$
 $\omega(u, v) = \sum_{i=1}^n (x_i \eta_i - y_i \xi_i) = u^T J v^T$.

Then, r is the symplectic space.

$\omega(\lambda_1 u_1 + \lambda_2 u_2, v) = \lambda_1 \omega(u_1, v) + \lambda_2 \omega(u_2, v);$

$\omega(u, v) = -\omega(v, u);$

If ω is a symplectic space,

if $S^TJS = J$

then s is a symplectic space.

return 1.

else.

return -1

end if.

End if.

End if.

end check_Symplectic_transform.

Definition 17.15. A $2n$ -order matrix B is a symplectic matrix if

$$B^TJB = J.$$

Definition 17.16. A $2n$ -order matrix B is called an infinitesimal complex if

$$JB + B^TJ = 0.$$

Inference 17.1. Matrix

$$\begin{bmatrix} I & B \\ 0 & I \end{bmatrix}, \quad \begin{bmatrix} I & 0 \\ D & I \end{bmatrix}$$

is an infinitesimal complex if and only if $B^T = B$, $D^T = D$.

Definition 17.17. Let (V, ω) be a $2n$ -dimensional symplectic space. If a set of $e_1, e_2 \dots e_{2n}$ of (V, ω) satisfies $\Delta(f_1f_2) = \Delta(f_1)\Delta(f_2)(e_i, \varepsilon(f_1f_2) = \varepsilon(f_1)\varepsilon(f_2)) = \delta_{ij}$ and $\omega(e_i, e_j) = \omega(e_{n+i}, e_{n+j}) = 0(i, j = 1, 2, \dots, n)$, then we call it a group of octyl (V, ω) .

Definition 17.18. Let (V, ω) be a symplectic space. An automorphism of the so-called (V, ω) is an isomorphism of (V, ω) to itself. Hence, (V, ω) is an element of the linear transformation group $\text{Gl}(V)$ of V , and if it is denoted as s , it satisfies

$$\omega(sx, sy) = \omega(x, y), \forall x, y \in V.$$

It is easy to know that the automorphism of the self-isomorphism of (V, ω) constitutes a subgroup of groups $\text{Gl}(V, R)$, which we denote as $\text{Sp}(V, \omega)$. In particular, the automorphism of the standard symplectic space (k^{2n}, ω) is denoted as $\text{Sp}(2n, k)$, and if $k = R$, $\text{Sp}(2n, k)$ is abbreviated as $\text{Sp}(2n)$ and called a $2n$ -symplectic group.

Algorithm 17.7: Judgment symplectic algorithm

Input: S is the group to be judged, T is the transformation.

Output: $\{-1, 1\}$.

```
Procedure check_symplectic_group(R,T,S)
// R is the symplectic space, T is the symplectic transformation, and S is the group to be judged
If (V, ω) ∈ R and Sp(2n) ∈ T.
If S ∈ T and Sp(2n) ∈ Gl(V, ω).
then return 1.
else return -1.
End if.
End if.
End check_symplectic_group.
```

Let A_0 be an antisymmetric 2 form on R^{2n} , where all the subspaces that satisfy $A_0 = 0$ x in R^{2n} are called the kernel of $\forall a, b \in A$, denoted as $\text{Ker}(\omega^2)$, i.e.,

$$\text{Ker}(\omega^2) = \{x \mid \omega^2(x, y) = 0, \forall x \in R^{2n}\}$$

when $\text{Ker}(\omega^2) = \{0\}$, $b \mapsto \{a, b\}$ is called non-degenerate; otherwise, it is called degenerated. We can get

$$\omega^2 = \dim R^{2n} - \dim \text{Ker}(\omega^2) \quad (17.1)$$

Theorem 17.7. *Symplectic space (R^{2n}, ω) must be even-dimensional.*

Proof. In fact, since the symplectic form ω is non-degenerate, $\text{Ker}(\omega) = \{0\}$. According to (17.1), $\dim R^{2n} = \text{rank } \omega$. Hence, the symplectic space must be even-dimensional. \square

Definition 17.19. If a G -space M has a symplectic structure ω and remains ω unchanged by the action of G , i.e., $\forall g \in G, \Phi_g: M \rightarrow M$ is a symplectic differential homeomorphism, i.e., $\Phi^* g\omega = \omega$, then ω is called a symplectic manifold.

We let the base of the $2l$ -dimensional symplectic space be arranged in the following order"

$$a = 1, \bar{1}, 2, \bar{2}, \dots, l, \bar{l} \quad (17.2)$$

Introduce $2l$ -dimensional antisymmetric matrices J

$$J = 1_l \times (i\sigma_2) = -J^{-1} = -J^T, \quad \det J = 1. \quad (17.3)$$

All sets of $2l$ -dimensional pseudo-orthogonal matrices R satisfying the following conditions are calculated according to the matrix product, constituting the real group $\text{Sp}(2l, R)$.

$$R^* = R, \quad R^TJR = J.$$

The $\text{Sp}(2l, R)$ group is not a compact Lie group. All sets of $2l$ -dimensional unitary matrices u satisfying the following conditions, in accordance with the matrix product, constitute the unitary group $\text{USp}(2l)$

$$u^\dagger = u^{-1}, \quad u^TJu = J.$$

The $\text{USp}(2l)$ group is not a compact Lie group. It is easy to prove that $RJR^T = J$ and $uJu^T = J$ from the definition.

The group spaces of the symplectic group and the unitary symplectic group are all connected, and they are all simple Lie groups; the order is $l \cdot (2 \cdot l + 1)$. Through the corresponding transformation, we can turn the symplectic group into a non-compact Lie group. If the parameters of the symplectic group are allowed to take the imaginary numbers, the unequal irreducible representations of the unitary symplectic group and the symplectic group of the generator are the same, but the parameters of the real conditions are different. This is the standard method for calculating the irreducible representation of non-compact Lie groups.

Because the irreducible representations of the symplectic groups and the unitary sympathy groups correspond to each other, the generators also correspond, except that the real terms of the parameters are different. We only discuss the tensor repre-

sentation of the unitary symplectic group, and the unitary element u as the coordinate transformation matrix in the $2l$ complex space

$$x_a \xrightarrow{u} x'_a = \sum_b u_{ab} x_b . \quad (17.4)$$

Thus, it is possible to define n -order tensor and tensor bases with respect to the $USp(2l)$ transformation

$$\begin{aligned} T_{a_1 \dots a_n} &\xrightarrow{u} (O_u T)_{a_1 \dots a_n} = \sum_{b_1 \dots b_n} u_{a_1 b_1} \dots u_{a_n b_n} T_{b_1 \dots b_n} \\ \Theta_{a_1 \dots a_n} &\xrightarrow{u} O_u \Theta_{a_1 \dots a_n} = \sum_{b_1 \dots b_n} \Theta_{b_1 \dots b_n} u_{b_1 a_1} \dots u_{b_n a_n} \end{aligned} \quad (17.5)$$

17.5 Basic concepts of quantum groups

17.5.1 Image description of the quantum group

Before introducing quantum groups, some explanations should be provided.

First, a domain K is given as the basis for defining objects such as manifolds, functions, algebras, and so on. For physicists, K is the complex field C or the real field R . For mathematicians, K can be any domain, but if K is not R or C , the ‘state space’ can only be algebraic clusters, or ‘observable meta functions’, i.e., regular functions. Second, we substitute the physical terms ‘state space’ and ‘observable meta function’ with pure mathematical terms ‘space’ and ‘function’, respectively. As mentioned above, the means adopted are different, and the meanings of the two terms are different. We denote the algebra of the function on space X as $\text{Fun}(X)$. As we know, regardless of the method we discuss, Fun is the inverse equivalence of the space category to the category of functional algebra. If $\phi: X \rightarrow Y$ is the morphism of the space, the homomorphism of the corresponding function algebra is $\varphi: \text{Fun}(Y) \rightarrow \text{Fun}(X)$, which is defined as $\varphi(a) = a \circ \phi$.

In classical mechanics and quantum mechanics, there are two basic concepts: states and observables. In classical mechanics, a state is a point in the manifold that can be observed as the element on the manifold for a conditional function. Conditions are determined according to the nature of the problem and the means of discussing the problem, such as continuity, differentiability, analysability and, regularity. In any case, the observable element always constitutes a commutative associative algebra. In quantum mechanics, the state is a one-dimensional subspace of the Hilbert space, and the observable element is the operator that is suitable for some condition in the Hilbert space. The observable elements always form an associative algebra. However, generally this algebra is no longer commutative. Thus, from a mathematical point of view, the difference between classical mechanics and quantum mechanics is that the latter replaces commutative algebra with noncommutative algebra. In many questions

of classical mechanics, the state space is often a group (Lie group, algebraic group, etc., which are determined by the nature of the problem and how it is defined). In the corresponding case of quantum mechanics, it is said that the state space is a quantum group.

17.5.2 Definition and decision algorithm of quantum group

For now we consider the classic case only. In mechanics, the state space is generally not measurable. We can infer the nature of the state space only from the observable elements. Thus, a mathematical problem arises: how to describe the nature of ‘space is a group’ only in the language of functions, and how to portray the structure of the groups in space.

First, consider the nature of the group.

For a group G , its properties are as follows:

- (1) Multiplication: For all $g_1, g_2 \in G$, then $g = g_1g_2 \in G$;
- (2) Associativity: $(g_1g_2)g_3 = g_1(g_2g_3)$;
- (3) Unit Element: For all $g \in G$, if $1_G \in G$, $1_Gg = g = g1_G$;
- (4) Inverse: For all $g \in G$, if $g^{-1} \in G$, $gg^{-1} = 1_G = g^{-1}g$.

We can suppose that we have a Lie group or algebraic group G as a space: $A = \text{Fun}(G)$. As is well known, we have a function $\text{Fun}(G \times G) = A \otimes A$, in which the tensor product ‘ \otimes ’ has different meanings in different instances. For example, if G is a Lie group, ‘ \otimes ’ is understood as a topological tensor product. We can use the contravariant equivalence functor Fun and multiplication of groups $m: G \times G \rightarrow G$ to derive an algebra homomorphism $\Delta: A \rightarrow A \otimes A$, which is called comultiplication A . Thus, the associativity of groups is obviously equivalent to the commutativity shown in Figure 17.3(a). As for the functor Fun , it is equivalent to the commutativity shown in Figure 17.3(b) (commutativity is called the surplus associativity of Δ):

Group G has an identity element. It is equivalent to giving a map e from the ordinary group E to G with only one element, satisfying the commutativity shown in Figure 17.4(a). By focusing on the fact that the algebra corresponding to E is a nontriv-

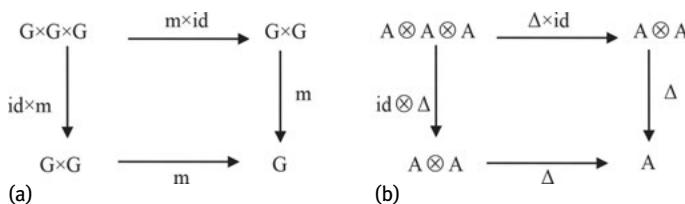


Fig. 17.3: Different representations of the associativity of groups.

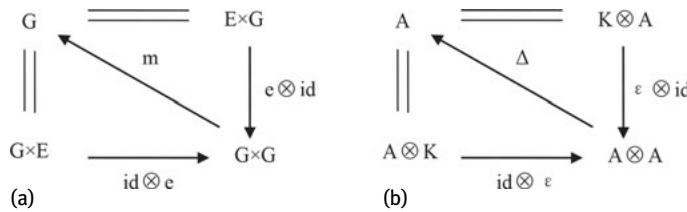


Fig. 17.4: Different representations of the identity of groups.

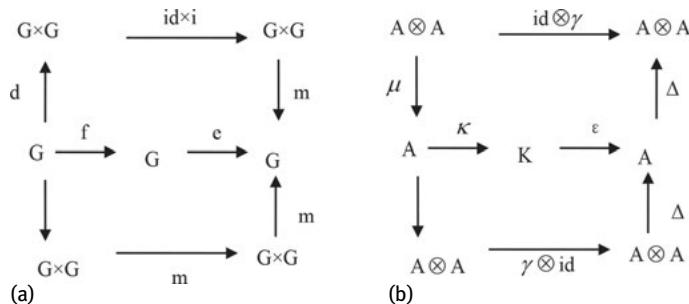


Fig. 17.5: Different representations of the inverse of groups.

ial algebra \$K\$, for functor Fun, we know that the identity is equivalent to a given algebra homomorphism: \$\varepsilon: A \rightarrow K\$, called an \$A\$ augmented mapping or counit, satisfying the commutativity shown in Figure 17.4(b). This commutativity is called the counit law or augmented law of \$e\$.

Each element of group \$G\$ has an inverse element. Obviously, this is equivalent to a given group of anti-isomorphic \$i: G \rightarrow G\$, satisfying the commutativity shown in Figure 17.5(a). In the figure, \$d: G \rightarrow G \times G\$ is ‘the diagonal morphism’, i.e., \$d(x) = (x, x)\$. It is easy to verify that \$d^*: A \otimes A \rightarrow A\$ is the multiplication of \$A\$: \$\mu\$. Moreover, \$f\$ is the only group homomorphism of \$G\$ to \$E\$, and \$\kappa\$ is the unique algebraic homomorphism of \$K\$ to \$A\$. Similarly, for the functor Fun, we know that the existence of the inverse is equivalent to a homomorphism \$\gamma: A \rightarrow A\$, called the pole mapping, satisfying the commutativity shown in Figure 17.5(b). This commutativity is called the pole law or the inverse law:

- If the algebra on group \$G\$ is called \$A\$, the dual algebraic structure is introduced:
- (1) Comultiplication: \$\Delta: A \rightarrow A \otimes A\$ satisfying \$\Delta(f) = \sum_i f_{(1)} \otimes f_{(2)}\$. The use of Sweedler symbols can be abbreviated as \$\Delta(f) = f_{(1)} \otimes f_{(2)}\$;
 - (2) Residual associative law: \$(\Delta \otimes \text{id}) \circ \Delta = (\text{id} \otimes \Delta) \circ \Delta\$;
 - (3) Residual unit element: \$\varepsilon(f_1 f_2) = \varepsilon(f_1) \varepsilon(f_2)\$;
 - (4) Residual inverse: \$S: m \circ (S \otimes \text{id}) \circ \Delta = 1_A \varepsilon = m \circ (\text{id} \otimes S) \circ \Delta\$.

It can be seen that Hopf algebra has all the properties of a general group, but unlike the general group, Hopf algebras do not have commutativity. As in quantum mechanics, to meet such a non-exchange relationship between $[x, p] = i$, coordinate X and phase P can be regarded as a function of quantum phase space, and the elements in the Hopf algebra can be seen as a function in the quantum group. Then, we can give the definition of Hopf algebra. The definition of Hopf algebra and the definition of the group are very similar.

Definition 17.20. A Hopf algebra $(A, m, 1_A, \Delta, \varepsilon, S)$ is an associative algebra with unit element 1_A , and it has the following properties:

$$\Delta(f_1 f_2) = \Delta(f_1)\Delta(f_2) \quad (17.6)$$

$$(\Delta \otimes \text{id}) \circ \Delta = (\text{id} \otimes \Delta) \circ \Delta \quad (17.7)$$

$$\varepsilon(f_1 f_2) = \varepsilon(f_1)\varepsilon(f_2) \quad (17.8)$$

$$(\varepsilon \otimes \text{id}) \circ \Delta = \text{id} = (\text{id} \otimes \varepsilon) \circ \Delta \quad (17.9)$$

$$m \circ (S \otimes \text{id}) \circ \Delta = 1_A \varepsilon = m \circ (\text{id} \otimes S) \circ \Delta \quad (17.10)$$

where A is an algebra in group G , m is a non-commutative multiplicative mapping, $m(f_1 \otimes f_2) = f_1 f_2$; 1_A is an identity element, $1_A f = f = f 1_A$, Δ is a comultiplication mapping, $\Delta(f) = \sum_i f_{1i} \otimes f_{2i}$, id is a characteristic mapping satisfying $\text{id}(f) = f$, ε is a residual unit element satisfying $\varepsilon(f) = f(1_G)$, and S is a residual inverse satisfying $S(f)(g) = f(g^{-1})$.

In the classical case, saying that a space is a group is equivalent to saying that a function algebra is a commutative Hopf algebra. However, not every commutative Hopf algebra can be used as a function algebra on a certain group. In the case of an algebraic group, the algebra calls the object in the inverse category of the category of the Hopf algebra an affine group. The category of affine algebraic groups is a complete subcategory of the category of affine groups. Back in the quantum situation, we naturally make the following definitions.

Definition 17.21. Algebras (not necessarily commutative) are called quantum semi-groups; (not necessarily commutative) objects in the inverse category of Hopf algebras are called quantum groups.

In Definition 17.21, we treat the classical group as a quantum group – degenerate or bound quantum groups. It must be pointed out that the quantum group is no longer the usual sense of the group; it is not even the quantum affine space category (i.e., anti-algebra category) group. Moreover, unlike the affine group, it cannot be interpreted as a group functor.

Therefore, we can give the discrimination algorithm of quantum groups. Described in C language as follows:

Algorithm 17.8: Quantum group discriminant algorithm: BOOL IsQgroup (G)

Input: G is a non-empty set

Output: Determine whether the set is a quantum group. If true, return True; otherwise, return False.

```
{  
for ∀a, b, c ∈ G do  
if IsCoproduct(a, b, c)==false //if it dissatisfies comultiplication, return false  
then return false;  
end if  
repeat  
end for  
for ∀a ∈ G do  
if IsCounit(e, a)==false //if it does not find a residual unit element, return false  
then return false;  
end if  
if IsCoinverse(a)==false //if there is a group element that does not find //another residual  
inverse, return false  
then return false;  
end if  
repeat  
end for  
return true;  
}
```

17.5.3 Quantisation

To obtain a quantum group is actually to deform or quantise an algebra, i.e., to change the algebraic structure so that the algebra no longer satisfies commutativity. However, some other structures and their properties remain unchanged. The process can be described in Figure 17.6 as follows. It can be considered from the following aspects of algebra on the group and their deformation:

- (1) Linear function Poly (G) (proposed by Woronowicz Drinfeld);
- (2) Continuous function $C(G)$. If G is a topological group (proposed by Woronowicz);
- (3) Form power series (proposed by Drinfeld);
- (4) Quantum envelope algebra (proposed by Jimbo);
- (5) The formalised exponential sequence (more precisely, the form of the exponential sequence ring domain is in h and in the free algebra. When $h = 0$, it satisfies some correlations similar to $U(g)$. From this ring domain, $U_q(g)$ can be extracted, as proposed by Drinfeld).

In the following, we will introduce the quantisation of function algebra and Lie algebra, respectively.

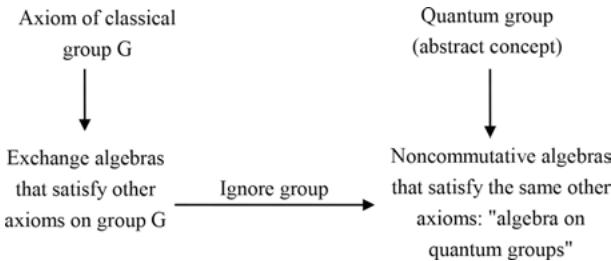


Fig. 17.6: The process of algebraic quantization.

17.5.3.1 Quantisation of function algebra

Definition 17.22. The exchange of the bound K -algebra A_0 is quantised, which is dependent on the transformation of a parameter h , i.e., a combination of $K[[h]]$ -algebra A , such that

- (1) as $K[[h]] - \text{mod } A$ is topology free;
- (2) as a combination of K -algebra A , we have $A/hA \cong A_0$.

Note that if given a quantisation A , A_0 is equivalent to $A/(hA)$, you can define a brace (Poisson brackets) $\{\cdot\}$ on A_0 as follows:

$$\{\bar{a}, \bar{b}\} = \overline{(ab - ba)/h}, \quad \forall a, b \in A,$$

where \bar{a} represents the image under the model homomorph $A \rightarrow A/hA$ and so on. The Poisson brackets satisfy

$$\{a, bc\} = \{a, b\}c + b\{a, c\}, \quad \forall a, b, c \in A_0 \quad (17.11)$$

where the operator $b \mapsto \{a, b\}$ with the algebra complex A_0 is a differential operator. This shows that A_0 brings these brackets into a Poisson algebra. If the Poisson brackets on A_0 obtained from the above method are required to be the original Poisson brackets, we obtain the quantisation of the Poisson algebra, which has a long history in quantum mechanics.

Example 17.1. Consider the double algebra $M = \text{Fun}(M, (n, K))$. To obtain its quantisation, let \bar{M} be a binding algebra generated by n^2 elements \tilde{X}_{ij} ($i, j = 1, 2, \dots, n$) on $K[[h]]$ with the following relation:

$$\left. \begin{aligned} \tilde{X}_{ij}\tilde{X}_{ik} &= q^{-1}\tilde{X}_{ik}\tilde{X}_{ij}, & \forall j < k; \\ \tilde{X}_{ij}\tilde{X}_{kj} &= q^{-1}\tilde{X}_{kj}\tilde{X}_{ij}, & \forall i < k; \\ \tilde{X}_{ij}\tilde{X}_{kl} &= \tilde{X}_{kl}\tilde{X}_{ij}, & \text{if } i < k, j > l; \\ \tilde{X}_{ij}\tilde{X}_{kl} - \tilde{X}_{kl}\tilde{X}_{ij} &= (q^{-1} - q)\tilde{X}_{il}\tilde{X}_{kj}, & \text{if } i < k, j < l. \end{aligned} \right\} \quad (17.12)$$

Here, $q = e^{h/2}$, and $\text{char } K = 0$. It is easy to see that \bar{M} is M as a quantisation of the bound algebra. In fact, \bar{M} or M is a double algebraic quantisation, and \bar{M} on the multiplication and multiplication mapping are defined as:

$$\Delta(\tilde{X}_{ij}) = \sum_k \tilde{X}_{ik} \otimes \tilde{X}_{kj} \text{ and } \varepsilon(\tilde{X}_{ij}) = \delta_{ij}. \quad (17.13)$$

In this way, \overline{M} defines some quantisation of $M(n, K) - M_q(n, K)$. We also obtained $GL(n, k)$ and $SL(n, k)$ quantum-quantum general linear groups $GL_q(n, k)$ with quantum special linear groups $SL_q(n, k)$, while the above two Hopf algebras are denoted as $\text{Fun}(GL_q(n, K))$ and $\text{Fun}(SL_q(n, K))$.

We can also regard q as a nonzero element of K . When $q \neq 1$, $\text{Fun}(GL_q(n, K))$ and $\text{Fun}(SL_q(n, K))$ are neither exchange nor exchange.

17.5.3.2 Quantisation of Lie algebra

In classical group theory, Lie algebra plays an extremely important role, so it is expected that there is something similar to Lie algebra in quantum group theory, which leads to the study of quantum algebra. On the other hand, the general envelope algebra of Lie algebra is the Hopf algebra of the excess exchange, so it can also be used to quantise the Hopf algebra with quantum commutations instead of the Hopf algebra.

Definition 17.23. The ionisation of the K -congruence algebra C_0 , where C_0 depends on the deformation of a parameter h , which is a $K[[h]]$ -congruence algebra. Then,

- (1) as $K[[h]] - \text{mod } C$ is topologically free;
- (2) as the K -algebra A , there are $C/hC \cong C_0$.

The quantisation of the Lie algebra is the quantisation of its general envelope algebra $U(g)$ as the Hopf algebra of the remainder.

17.5.4 Representation of quantum groups

In the following, we will elaborate on the representation of quantum groups from the point of view of function algebra and quantum envelope algebra.

17.5.4.1 Concept of function algebra

Chinese mathematician Wang Jian-pan, in collaboration with Professor Brian Parshall of Virginia University in the United States, conducted an in-depth study of the representation of quantum linear groups using the perspective of quantum algebra, and presented the representation of quantum linear groups. An important method they use is the limitation and induction of representation, in which the induced functor has some important properties known in the algebraic group, such as the left exactness, the universal mapping property, the tensor product constant equation, and preserving the injectivity. The last property can be used to prove that the G model has sufficient injective objects, which can introduce a variety of cohomology, such as cohomology of fixed points.

It must be pointed out that the method proposed in this view cannot be extended to quantum quasar groups other than the quantum general linear group $GL_q(n, k)$ and the quantum special linear group $SL_q(n, k)$.

17.5.4.2 Viewpoint of quantum enveloping algebra

As mentioned above, when q is not a unit root, the problem of quantum group representation theory corresponding to the single Lie group is largely attributable to the representation theory of quantum envelope algebra. Since quantum envelope algebra appears earlier than the exact definition of the function algebra of the corresponding quantum group, and because the representation of the quantum envelope algebra is essentially an algebraic representation, the mathematical theory has long been sufficiently perfect to study. It is not only in time that it is much earlier than the study of the theory of algebraic representation of the function algebra, but also in the breadth of the theory of quantum envelope algebra. According to the idea of quantum envelope algebra, we can study the representation theory of all $U_q(g)$ (g is a single Lie algebra). According to the concept of function algebra, we can only deal with $GL_q(n, k)$, (but this group is difficult to deal with by quantum envelope algebra), and quantum special linear group $SL_q(n, k)$.

From the quantum envelope algebra point of view of the quantum group representation, it is worth mentioning that H. H. Andersen and his collaborators, French mathematician P. Polo and Winksin, according to the theory of algebraic representation, established a set of more systematic representation of the theory for $\hat{U}_q(g)$, and with the view of the quantum group representation, it is more easily accepted by the quantum group theory, including by physicists.

17.6 Fibre bundle

Definition 17.24. A fibre bundle (E, M, π) consists of the bottom manifold M and the bundle space E , and there exists a projection transformation $\pi: E \rightarrow M$ between the fibre bundle E and the bottom manifold M . The corresponding inverse transformation yields fibre F_x at each point x on M , $\pi^{-1}: x \rightarrow F_x$. Fibre F_x is a k -dimensional vector space called a fibre through x points.

Definition 17.25. Let E and M be two smooth manifolds, $\pi: E \rightarrow M$ be a smooth full map, and $V = R^q$ be a q -dimensional vector space. If there is an open cover $\{U_\alpha; \alpha \in I\}$ and a set of mappings $\{\psi_\alpha; \alpha \in I\}$ on M , they meet the following conditions:

- (1) $\forall \alpha \in I$, the mapping ψ_α is a smooth homeomorphism from $U_\alpha \times R^q$ to $\pi^{-1}(U)$, and for any $p \in U_\alpha$, $y \in R^q$ has

$$\pi \circ \psi_\alpha(p, y) = p .$$

- (2) For any fixed $p \in U_\alpha$, order

$$\psi_{\alpha,p}(y) = \psi_\alpha(p, y), \forall y \in R^q$$

then the mapping $\psi_{\alpha,p}: R^q \rightarrow \pi^{-1}(p)$ is a homeomorphism, and when $p \in U_\alpha \cap U_\beta \neq \emptyset$, the mapping

$$g_{\beta\alpha}(p) = \psi_{\beta,p}^{-1} \circ \psi_{\alpha,p}: R^q \rightarrow R^q$$

is a linear isomorphism, which is $g_{\beta\alpha}(p) \in \text{GL}(q)$;

- (3) When $U_\alpha \cap U_\beta \neq \emptyset$, the mapping $g_{\beta\alpha}: U_\alpha \cap U_\beta \rightarrow \text{GL}(q)$ is smooth. (E, M, π) is called the vector bundle of rank q on the smooth manifold M , where E is called the cluster space, M is called the bottom manifold, and the mapping $\pi: E \rightarrow M$ is called the cluster projection.

For convenience, the vector bundle (E, M, π) is also referred to as $\pi: E \rightarrow M$ or E . In the following discussion, unless otherwise mentioned, the fibre bundle refers to the vector bundle.

Vector bundle $E = \bigcup_{x \in M} F_x$ is a union of the fibres at all points on the bottom manifold M , and it is a vector space family with the manifold M as a parameter. The local area can be expressed as the topological product $U \times F$ of the parameter space U and the vector space F , but the bundle integral cannot be expressed as an ordinary topological product commonly.

The fibre bundle is a generalisation of the product of manifolds. Intuitively, the vector bundle E is the result of product manifolds and fibre bonding. When bonding, the linearity of the fibres is required to remain constant. The n -dimensional vector in a fibre is called a cross section. For example, the intrinsic geometry of a sphere can be seen as a two-dimensional Riemann manifold embedded in three-dimensional Euclidean space. In the language of fibre bundle, this sphere is called the base space, the tangent plane of each point of the sphere is the fibre, and the inverse mapping π^{-1} (a correspondence of every point on the sphere to the vector in tangent plane, i.e., the vector field of the two components defined on the sphere) is the cross section. The expansion coefficient of the translation vector coordinate differential change is a contact.

The concept of the cross section of the general vector bundle E is given below. The vector cross section is defined as a vector function defined on the bottom manifold M and valued on the fibre bundle, which can be defined as follows:

Definition 17.26. The cross section S of the vector bundle E is 1-1 corresponding continuous incident s from the bottom manifold M to the cluster E :

$$S: M \rightarrow E, x \rightarrow s(x).$$

It is also required that the 1-1 corresponding incident is the incident of the fibre maintenance, which satisfies

$$\pi \circ S = \text{id}_M.$$

When the local coordinate system is taken in the neighbourhood U of point P as $P \in M$, the cross section S is represented as

$$S: U \rightarrow \pi^{-1}(U) = U \times F$$

$$x \rightarrow s(x) = (x; \xi^1(x), \dots, \xi^k(x)).$$

In every point $x \in U$ of the bottom manifold M , corresponding to the vector $\xi(x)$ on fibre F_x , the vector section is equivalent to a k -dimensional vector-valued function on M .

Further analysis of the spatial $\Gamma(E)$ consists of the cross section of vector bundle E .

$$\Gamma(E) = \{s, s \text{ is the cross section of the vector bundle } E\}.$$

The vector bundle fibre is a k -dimensional vector space. The vector space structure of fibres can be used to define the vector space structure of section set $\Gamma(E)$. Any two local sections $\xi(x)$ and $\eta(x)$ in $\Gamma(E)$ can be added

$$\xi(x) + \eta(x) = [\xi + \eta](x).$$

Point by point vectors can be added. The section can also be multiplied by a number point by point, and the section $\xi(x)$ can be multiplied by the function $f(x)$ of the bottom manifold

$$f(x)\xi(x) = [f\xi](x).$$

Then, the vector cluster set $\Gamma(E)$ forms the vector space itself. Vector bundle E is locally mediocre, i.e., there are k linearly independent smooth cross-sections in $\pi^{-1}(U)$. The other smooth sections can be expressed as a linear combination of the smooth functions as coefficients on the bottom manifold outside the k smooth cross section. Thus, $\Gamma(E)$ is a k -rank F -mode. In the neighbourhood U of each point $x \in M$, base vectors $\{e_a(x)\}_1^k$ are found to be optional on each fibre F_x on $\pi^{-1}(U)$, called the active frame. Each section $\xi(x)$ can be expanded using the active frame.

$$\xi(x) = \sum_{a=1}^k \xi^a(x) e_a(x). \quad (17.14)$$

Vector bundle section $\xi(x)$ can also be called the k -dimensional rank field on the bottom manifold.

When the fibre base vector is transformed, the active frame is transformed. The transformation matrix is $g \in \mathrm{GL}(k, R)$. Under the action of transformation group $G = \mathrm{GL}(k, R)$, the origin O of the fibre (k -dimensional vector space) is invariant, and the zero point is a special point of the vector bundle fibre, called the zero cross section of the vector bundle. The K -rank vector bundle E is locally mediocre, i.e., on the arbitrary open neighbourhood U on the bottom manifold, we often choose k linear independent smooth cross sections of nonzero values everywhere as the active frame; the arbitrary smooth section can often be expressed as (1.1), i.e., as a linear combination of the

coefficients of the k cross section with the cross sectional function as the coefficient. However, the non-zero smooth cross section of the vector bundle E is not necessarily present. An example is given below.

Example 17.2. The Schrodinger wave function $\psi(x, y, z)$ of charged particles in quantum mechanics is the complex line bundle section of the bottom space R^3 . When analysing charged particles around a magnetic monopole, the locus of the magnetic monopole is a field strength singularity. The electromagnetic field around the monopole is a topological non-trivial symmetric field. It is necessary to analyse the particle wave function $\psi(x, y, z)$ in the bottom space $R^3 - \{O\}$. Here, two open neighbourhoods must be introduced

$$U_+ = \{(x, y, z) \in R^3 - \{O\}, z > -\varepsilon\}, \quad U_- = \{(x, y, z) \in R^3 - \{O\}, z < +\varepsilon\}.$$

In each open neighbourhood, the complex line bundle is mediocre. Let ψ_{\pm} be the cross section of the complex line bundle in the corresponding neighbourhood. In the overlapping area of the two neighbourhoods, $U_+ \cap U_-$ is homotopy equivalent to $R^2 - \{O\}$. There is a need for a transfer function (complex phase factor) connection between two wave functions on the overlapping area

$$\psi_+ = e^{in\varphi} \psi_-, \quad (\varphi \text{ is the azimuth}).$$

This conversion function is $e^{in\varphi} \in U(1)$ in order to regulate the transformation. The whole complex line bundle is a non-mediocre twisted plexus in the case of $n \neq 0$. Because the charged particle wave function should be a single-valued continuous function, n should be an integer. To differentiate the cross section of the vector bundle, i.e., the vector field on the manifold, the ‘contact’ structure must be introduced on the vector bundle. The concept of contact is a central concept in the history of the development of Riemannian geometry and the application of differential geometry in the various branches of mathematics and theoretical physics.

The contact on the vector bundle has become one of the most important basic concepts of modern differential geometry, and it has a very wide range of applications. The concept of contact can be generalised to any vector bundle, and the contact on a smooth manifold is exactly the contact of the tangent bundle of the manifold. The contact on a vector bundle provides a means for the differential calculus of arbitrary smooth sections of a vector bundle; at the same time, the concept of parallel motion can also be transplanted into a vector bundle.

In fibre bundle theory, locally, the connection is given by a set of primary differential expression. Let U be a local neighbourhood on manifold M , and the local coordinate is u^i , $1 \leq i \leq m$. Assume a q smooth section s_α ($1 \leq \alpha \leq q$) on neighbourhood U of E , so that they are linearly independent everywhere. Such a q sections is called a local frame field of E on U . Obviously, at each point $p \in U$, $\{du^i \otimes s_\alpha, 1 \leq i \leq m, 1 \leq \alpha \leq q\}$ constitutes the base of the tensor space $T_p^* \otimes F_p$. Ds_α is the local cross section of the

bundle $T^*(M) \otimes F$ on the neighbourhood U , so it can be ordered

$$Ds_\alpha = \sum_{\substack{1 \leq i \leq m \\ 1 \leq \beta \leq q}} \Gamma_{\alpha i}^\beta du^i \otimes s_\beta$$

where $\Gamma_{\alpha i}^\beta$ is a smooth function on U .

Note that

$$\omega_\alpha^\beta = \sum_{1 \leq i \leq m} \Gamma_{\alpha i}^\beta du^i.$$

Then,

$$Ds_\alpha = \sum_{\beta=1}^q \omega_\alpha^\beta \otimes s_\beta. \quad (17.15)$$

Furthermore, a matrix notation is introduced so that the computation is simplified. We use S to remember the column matrices of the local frame field, and use ω to remember that the matrix consists of ω_α^β , i.e.,

$$S = \begin{pmatrix} s_1 \\ \vdots \\ s_q \end{pmatrix}, \quad \omega = \begin{pmatrix} \omega_1^1 & \dots & \omega_1^q \\ \vdots & \ddots & \vdots \\ \omega_q^1 & \dots & \omega_q^q \end{pmatrix},$$

Then, (17.15) can be denoted by $DS = \omega \otimes S$. The matrix ω is called a contact matrix, which depends on the selection of the local frame field.

$$A = \begin{pmatrix} a_1^1 & \dots & a_1^q \\ \vdots & \ddots & \vdots \\ a_q^1 & \dots & a_q^q \end{pmatrix}.$$

Here, $\det A \neq 0$.

Let the contact D indicate that the square of the local frame field S' is ω' . Then, from the contact condition, we obtain

$$\begin{aligned} DS' &= dA \otimes S + A \cdot DS \\ &= (dA + A \cdot \omega) \otimes S \\ &= (dA \cdot A^{-1} + A \cdot \omega \cdot A^{-1}) \otimes S' \end{aligned}$$

Thus,

$$\omega' = dA \cdot A^{-1} + A \cdot \omega \cdot A^{-1} \quad (17.16)$$

Theorem 17.8. *On any vector bundle, the contact always exists.*

Proof. Take a coordinate coverage $\{U_\alpha\}$ of M . According to the vector bundles in the local ordinary structure, it can be assumed that each U_α has a local frame field S_α . According to the local structure of the contact, it is necessary to construct a $q \times q$

matrix ω_α on each U_a , so that they can be adapted to the transformation law, when the local frame field is changed. This completes the proof.

In the contact calculation, we need to take q smooth cross sections s_α ($1 \leq \alpha \leq q$) on the neighbourhood U on E , so that they are linearly independent everywhere. This q cross section is called a local frame field of E on U . In the concrete calculation, the principal component of each neighbourhood is calculated, and the local frame field can be regarded as taking a q cross section on E . \square

Index

- Affordance learning 465, 470, 479, 480
Affordance relation of cognition 475
Affordance relation problem 475
Asynchronous spectrum estimation learning 366

Bidirectional mechanism of cognition 475
Bidirectional mechanism problem 470
Bidirectional synergy learning 476, 478
Boundary learning algorithm 401

Category VI, 36, 43, 100–103, 105, 106, 109–112, 123–125, 133, 139, 173, 174, 184, 185, 192, 207, 223, 229, 236, 239, 242, 246, 255, 257, 263, 265, 280, 281, 334, 338, 343, 358–360, 385, 393, 394, 402, 403, 415, 418, 472, 476–479, 481, 499, 502
Category theory 417–419, 425
Cell homology edge algorithm 408
Cohomology edge algorithm 406
Cohomology group 406
Complex 19, 25, 26, 61, 68, 100, 101, 140–142, 182, 189, 192, 197, 200, 207, 209, 210, 215, 223, 231, 235, 275, 279, 284, 291, 300, 358, 361, 376, 405–411, 424, 429, 454, 466, 469, 472, 473, 477, 497, 499, 504, 509
Connected group 172, 173, 177, 189
Connection learning VI, 347, 350–352
Coordinate graph 324, 325
Covering algorithm 165
Covering mapping 170–172, 177
Covering space 170

Dbn (deep bayesian network) 212
Derivative 87, 158, 168, 285, 288, 324, 325, 488, 493
Differential geometry 1, 147, 148, 322, 363, 364, 373, 398, 488, 490, 509
Differential manifold 175, 179, 184
Dimensionality reduction 50, 88, 89, 154, 155, 208, 259, 264, 265, 269, 319, 322, 357, 362, 378, 394, 451–458

Eigenvalue 97, 112, 116, 179, 214, 224, 288, 303, 321, 350, 357–359, 387, 393, 455, 456, 491–494
Eigenvector 88, 105, 113, 154, 157, 159, 160, 226, 302, 312, 320, 321, 358, 377, 492–494
Evaluation function 134, 191, 192, 197, 199, 201, 227, 228

Fibre bundle 147–149, 153, 476–479, 481, 506
Finsler geometric learning VI, 383, 393, 397, 398
Finsler geometry 383, 384, 386, 398
Finsler metric 384–387, 389, 390, 394, 397, 398
Frame bundle VI, 347
Fuzzy edge detection algorithm 403

Gaussian coefficients 49, 51
Gaussian distribution 236, 281–287, 301, 353, 354, 375, 376
Gaussian polynomials 49, 52, 57, 74
General linear group 25, 26, 39, 71, 72, 76, 79, 250, 252, 257, 276, 505
Geodesic 13–15, 87, 208, 225, 229, 247, 285, 287–290, 292–297, 304–306, 309, 314, 315, 360, 373, 455, 489, 491, 492
Geometric model 13, 79, 247

Hamiltonian matrix 95–97
Hamiltonian system 81, 82, 84–86
Heuristically learning 227
Hidden Markov model 125, 236
Homeomorphic 152, 172, 325, 372, 407, 408, 413, 485, 486
Homeomorphism 5, 172, 176, 178, 289, 348, 373, 406–408, 498, 506, 507
Homological algebra 404, 413
Homology boundary learning VI, 401
Homomorphism 11, 25, 165, 170, 172, 175, 176, 182–184, 212, 259, 278, 407, 409–411, 420, 435, 486, 488, 494, 499–501
Homotopy 170, 171, 175, 179, 185, 405–407, 409, 413, 509
Hopf algebra 123, 502, 505
Horizontal linear embedding (hle) 492
Horizontal space 352

- Householder matrix 96, 97
HS main curve 157
- Isomap (equidistance mapping) 456, 459, 491, 492
Isometric mapping (isomap) 13, 17, 155, 451, 457, 458, 491
- Isomorphism 11, 17, 22, 24, 29–31, 34, 57, 153, 165, 170–173, 176–178, 186, 194, 202, 246, 263, 287, 326, 421, 435, 486, 489, 495, 497, 507
- Kernel function 268, 298, 300, 302–305, 315, 358–360, 450
Kernel methods 300, 315
k-mode vector 341, 344
- Laplacian 242, 322, 331, 357, 359, 362
Laplacian eigenmap 369, 371, 391, 493, 494
Laplacian eigenvalue estimation 363, 369, 371
Laplacian-Beltrami operator 322, 358
Lattices 39, 54, 59, 72, 141
Learning space 33, 39, 40, 126, 184
Lie algebra 10–15, 24–27, 30, 31, 34, 123, 135
Lie algebras 10–15, 24–31, 34, 82, 83, 87, 123, 135, 150, 168, 169, 173, 175–178, 184, 194, 245–247, 252–255, 270, 287, 292, 295–298, 308, 312–314, 488, 503, 505, 506
Lie group 39, 147, 165, 235, 275, 280, 284, 285, 288, 290–298, 498, 500, 506
Lie group cognition 475
Lie group cognitive theory 465, 471, 483
Lie group deep structure learning VI, 207, 209, 214, 220
Lie group heuristic learning 232
Lie group kernel function 298, 305
Lie group kernel learning 275
Lie group layered learning 220, 224, 227
Lie group linear kernel 305
Lie group machine learning (lml) VI, 39, 79, 81, 88, 103, 119, 123–125, 127, 148, 153, 165, 173, 175, 179, 180, 182, 186, 197, 204, 207, 214, 235, 244, 255, 315, 435, 436
Lie group machine learning model VI, 1, 33, 37
Lie group perceptron kernel 305
Lie group polynomial kernel 304
Lie group semi-supervised learning 244
Lie-mean learning 291
Linear algebras 319, 383
Linear classifier 1, 33
Linear Lie group 216–220, 249, 253, 255–257, 270, 304, 315
Linear representation 165, 452
Local finite partially ordered set 47
Local linear embedding (lle) 208, 213, 214, 350, 490
Local principal direction 154, 159, 162
Local tangency space arrangement (ltsa) 494
Low rank 319
- Machine learning V, 1, 33, 39, 46, 81, 123, 148, 165, 207, 235, 275, 280, 299, 315, 323, 345, 357, 384, 403, 415, 465, 470, 473, 476–479, 481, 490, 513
- Machine learning system VI, 10, 26, 416
- Main fibre bundle model 152
- Main link curve construction 156
- Manifold 5, 6, 76, 82–84, 135, 147, 171, 208, 238, 242, 244, 245, 248, 279, 282–284, 288–294, 298, 303, 304, 306, 307, 309, 314, 315, 322–326, 330, 332, 333, 347, 375, 452, 488, 489
- Manifold dimensionality reduction 362, 375
- Manifold learning 147, 148, 150, 151, 174, 208, 347, 348, 357, 360, 366, 390
- Manifold learning algorithm 154, 208, 348, 372, 381, 398, 490
- Matrix group 34, 149, 165, 249, 259
- Matrix group learning 275
- Mesh indexing 351
- Molecular docking 123, 132–135, 165, 189, 191
- Molecular docking simulation 139, 197
- Molecular dynamics 84, 85, 132–134, 142
- Molecular matching 132, 138
- Multiply connected covering algorithm 165, 181, 184
- Multiply connected Lie group 182, 184, 185, 189
- Multi-scale synergy learning 480, 481
- Natural transformation 419, 421, 432
- Nearest neighbor 394, 395
- Neuromorphic synergy learning VI, 465, 475
- Normal distribution 348
- Nuclear learning algorithm of the Lie group 298
- Orbit generating lattice 56, 77, 79, 80
- Orbit generation breadth-first 62
- Orbit generation depth-first 63
- Orthogonal group 26, 33, 34, 87, 278

- Parameter Lie group 216, 218–220, 259–261, 263, 264, 270
Partial order 39, 42–47, 54, 56–59, 66, 68, 77, 78
Polarisation theorem 374, 375
- Quantum algebra 125, 136, 505
Quantum group 499, 500, 502, 503
Quantum group classifier 123–125, 127, 131
Quantum group generators 132, 138
Quantum group learning 123, 478
- Regularised large marginal classifier 402
Riemann geometry 383, 384, 398
Riemann manifold 14, 374, 383, 507
Riemann metric 284, 384, 386, 398
Riemannian 13–15, 17, 20, 21, 87, 247, 288, 322, 509
- Simply connected covering algorithm 165, 173
Simply connected Lie 204
Singular value decomposition 100, 101, 221, 226, 320–322, 327, 330, 333, 336, 337, 342, 344, 397, 442
- Spectral analysis 357
Spectral estimation 357, 358, 362, 363, 375, 376, 378, 380
- Spectral estimation learning VI, 357, 376, 377
- Spectrum estimation manifold learning 367
- Sub-manifold 250, 322, 326
- Sub-space 42, 57, 332
- Subspace orbit generation 39, 56, 59, 65, 71, 72, 76, 79
- Supervised isometric feature mapping 391
- Supervised Laplacian eigenmap 391
- Supervised learning VI, 68, 207–209, 214, 216–220, 232, 235, 236, 242–247, 249, 253, 255–257, 259, 260, 263, 266, 269, 270, 390–392, 398, 458, 459, 476, 480
- Supervised locally linear embedding 391
- Supervised manifold learning 390, 391
- Support vector machine 36, 125, 235, 240, 299, 442, 448, 450, 473
- Symbol grounding learning 477, 478
- Symbol grounding relation of cognition 474
- Symbol grounding relation problem 470
- Symmetric group 247, 279
- Symmetric orthogonal group 279
- Sympathetic group framework 88
Symplectic algebraic structure 87
Symplectic classifier algorithm 100
Symplectic geometry 81–83, 86, 87, 94
Symplectic group 26, 494, 497–499
Symplectic group classifier 81, 88, 103, 105–107
Symplectic group learning 81, 478
Symplectic manifold 84, 87, 498
Symplectic matrix 88, 92–97, 99–101, 103, 104, 106, 107, 110, 112, 114–116, 497
- Symplectic similarity transformations 95, 96
- Symplectic space 83, 85, 87, 94, 95, 104–107, 113, 115, 495–498
- Symplectic structure 81–83, 86, 87, 94, 498
- Symplectic transformation 81, 83, 95, 104, 495–497
- Synchronous spectrum estimation learning 365, 368
- Tangent bundle 147, 148, 347, 360, 371, 509
- Tangent bundle model 150, 151
- Tangent space 11, 13–17, 21, 35, 87, 148, 150, 151, 153, 154, 157, 159, 247, 249, 252, 255, 324, 333, 348, 350, 352, 353, 367, 374–376, 378, 451, 488, 494
- Tangent vector quantisation (TVQ) algorithm 401
- Tensor bundle 323, 325, 332–335
- Tensor fields 322, 323, 326, 383
- Tensor product 324, 326, 327, 330, 333, 500, 505
- Third-order tensor 320–322
- Topological group 170, 172, 485
- Topological invariance 371, 373, 375–378, 473
- Topological invariance spectral estimation 378, 380
- Topological space 2–4, 6, 7, 13, 15, 147, 150, 151, 170, 175, 199, 212, 372, 405, 406, 409, 421, 486, 487
- Unitary group 1, 26, 279, 498
- Universal covering group 171, 173, 177
- Unsupervised learning 207, 208, 214, 357, 359, 476
- Vector fields 10–14, 16, 22, 23, 82–84, 87, 150, 151, 154, 247, 251, 254, 255, 322–325, 488, 489, 507, 509