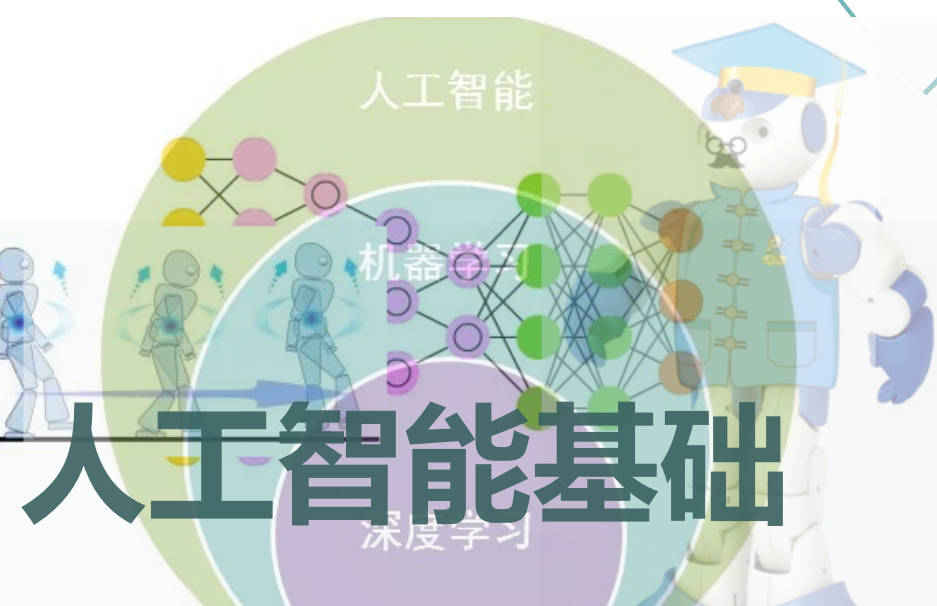


Version 3.14

编译日期: 2020-01-7

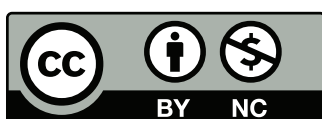
任何建议及错误信息请发送至邮箱  
ocnzhao@163.com



# 人工智能基础

赵国亮 整理

$$\rho := \frac{1 + \sqrt{-3}}{2}$$

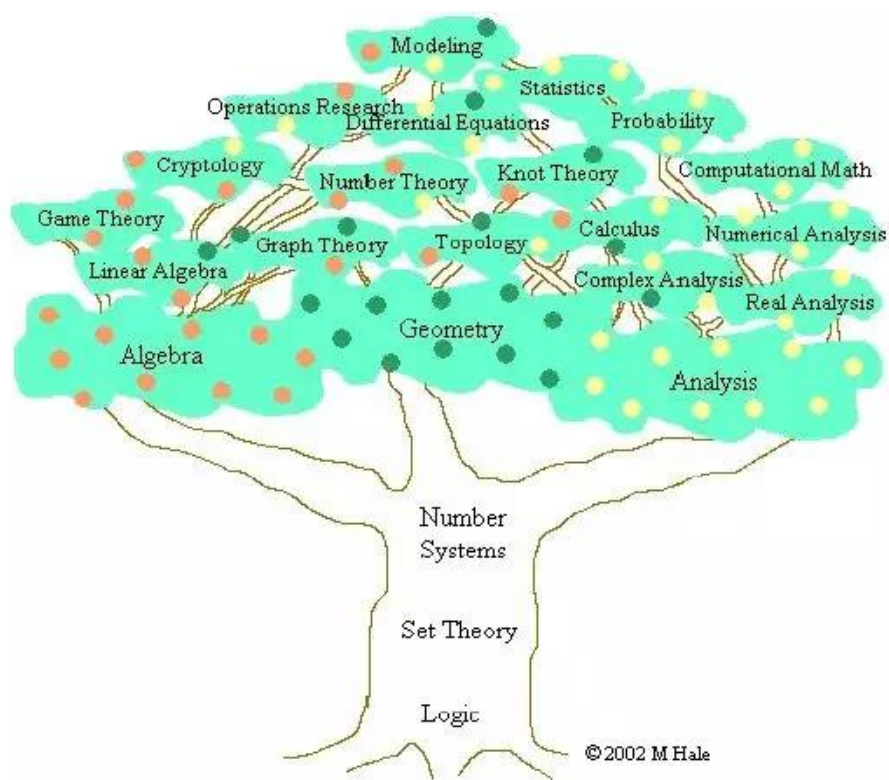


本作品采用知识共享署名-非商业性使用 4.0 国际许可协议进行许可。访问<http://creativecommons.org/licenses/by-nc/4.0/>查看该许可协议。



# 前言

材料来自于网上，如有侵权，请联系 ocnzhao@163.com 予以纠正,THX





# 目 录

前 言	i
概 览	vii
第一章 计算智能	1
1.1 概述	2
1.1.1 什么是计算智能	2
1.1.2 计算智能与人工智能的关系	2
1.1.3 计算智能的产生与发展	3
1.2 神经计算	4
1.2.1 神经计算基础	4
1.2.2 人工神经网络的互连结构	6
1.3 人工神经网络的典型模型	11
1.3.1 感知机 (Perceptron) 模型	11
1.3.2 反向传播 (BP) 模型	16
1.3.3 反馈网络 (Hopfield) 模型	17
1.4 其它神经学习算法	18
1.4.1 ELM	18
1.4.2 TROP-ELM	22
1.4.3 Allen's PRESS(prediction sum of squares)	22
1.4.4 区间二型 RVFL	24
1.5 进化计算	37
1.5.1 进化计算的产生与发展	38
1.5.2 进化计算的基本结构	39
1.6 遗传编码	41

1.7	PSO	54
1.8	QBFA	55
1.9	其它算法 *	55
1.9.1	鱼群算法	55
1.9.2	鸟群算法	55
1.9.3	象群算法	55
1.9.4	蚁群算法	55
1.9.5	狼群算法	55
1.9.6	蝙蝠算法	55
1.9.7	随机森林	55
1.9.8	ADP 聚类算法	55
1.10	模糊计算	66
1.10.1	模糊集及其运算	67
1.10.2	模糊关系及其运算	70
1.10.3	FNN	73
1.10.4	DFNN	73
1.11	神经模糊系统和模糊神经网络	73
1.12	作业	73
<b>第二章</b>	<b>不确定性推理</b>	<b>77</b>
2.1	不确定性推理的含义	78
2.2	不确定性推理的基本问题	78
2.3	不确定性推理的概率论基础	79
2.3.1	可信度的概念	82
2.3.2	CF 模型	82
2.3.3	可信度的定义与性质	83
2.4	证据理论	85
2.4.1	证据不确定性的表示	85
2.4.2	不确定性的更新	86
2.4.3	结论不确定性的合成	86
2.5	主观 Bayes 方法	88
2.6	证据理论	89
2.7	DS 理论的形式描述	89
2.7.1	证据理论的推理模型	95

2.8	模糊推理 . . . . .	104
2.8.1	模糊概念的匹配 . . . . .	106
2.8.2	模糊推理 . . . . .	107
2.9	高阶模糊推理 . . . . .	112
2.10	作业 . . . . .	112





# 概 览

推荐参考文献-来自





# 1

## 计算智能

### 计算智能

计算智能是信息科学、生命科学、认知科学等不同学科相互交叉的产物. 它主要借鉴仿生学的思想, 基于人们对生物体智能机理的认识, 采用数值计算的方法去模拟和实现人类的智能. 计算智能的主要研究领域包括: 神经计算、进化计算、模糊计算、免疫计算、DNA 计算和人工生命等.



## 1.1 概述

本章主要讨论神经计算、进化计算和模糊计算问题.

### 1.1.1 什么是计算智能

计算智能 (Computational Intelligence, CI) 目前还没有一个统一的定义, 使用较多的是美国科学家贝慈德克 (J. C. Bezdek) 从计算智能系统角度所给出的定义:

如果一个系统仅处理低层的数值数据, 含有模式识别部件, 没有使用人工智能意义上的知识, 且具有计算适应性、计算容错力、接近人的计算速度和近似于人的误差率这 4 个特性, 则它是计算智能的.

从学科范畴看, 计算智能是在神经网络 (Neural Networks, NN)、进化计算 (Evolutionary Computation, EC) 及模糊系统 (Fuzzy System, FS) 这 3 个领域发展相对成熟的基础上形成的一个统一的学科概念.

神经网络是一种对人类智能的结构模拟方法, 它是通过对大量人工神经元的广泛并行互联, 构造人工神经网络系统去模拟生物神经系统的智能机理的.

进化计算是一种对人类智能的演化模拟方法, 它是通过对生物遗传和演化过程的认识, 用进化算法去模拟人类智能的进化规律的.

模糊计算是一种对人类智能的逻辑模拟方法, 它是通过对人类处理模糊现象的认知能力的认识, 用模糊逻辑去模拟人类的智能行为的.

从贝慈德克对计算智能的定义和上述计算智能学科范畴的分析, 可以看出以下 2 点:

- 第一, 计算智能是借鉴仿生学的思想, 基于生物神经系统的结构、进化和认知对自然智能进行模拟的.
- 第二, 计算智能是一种以模型 (计算模型、数学模型) 为基础, 以分布、并行计算为特征的自然智能模拟方法.

### 1.1.2 计算智能与人工智能的关系

目前, 对计算智能与人工智能的关系有 2 种不同观点, 一种观点认为计算智能是人工智能的一个子集, 另一种观点认为计算智能和人工智能是不同的范畴.

第一种观点的代表人物是贝慈德克. 他把智能 (Intelligence, I) 和神经网络 (Neural Network, NN) 都分为计算的 (Computational, C)、人工的 (Artificial, A) 和生物的 (Biological, B) 3 个层次, 并以模式识别 (PR) 为例, 给出了下图所示的智慧的层次结构.

在图 1-1 中, 底层是计算智能 (CI), 它通过数值计算来实现, 其基础是 CNN; 中间层是人工智能 (AI), 它通过人造的符号系统实现, 其基础是 ANN; 顶层是生物智能 (BI), 它

通过生物神经系统来实现, 其基础是 BNN.

按照贝慈德克的观点, CNN 是指按生物激励模型构造的 NN, ANN 是指 CNN+ 知识, BNN 是指人脑, 即 ANN 包含了 CNN, BNN 又包含了 ANN. 对智能也一样, 贝慈德克认为 AI 包含了 CI, BI 又包含了 AI, 即计算智能是人工智能的一个子集.

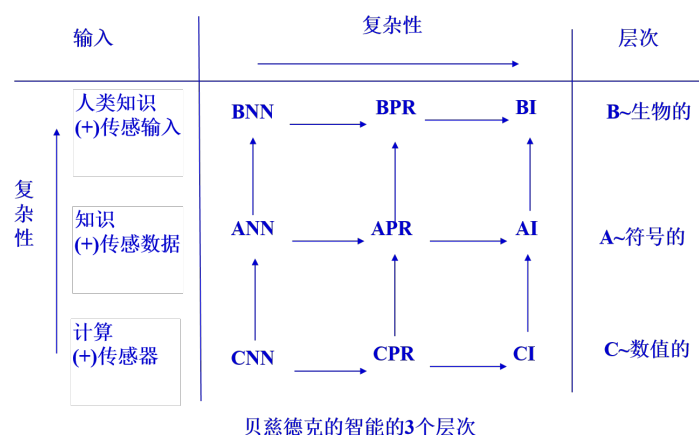


图 1-1 贝慈德克的智能的 3 个层次

第二种观点是大多数学者所持有的观点, 其代表人物是艾伯哈特 (R.C.Eberhart). 他们认为: 虽然人工智能与计算智能之间有重合, 但计算智能是一个全新的学科领域, 无论是生物智能还是机器智能, 计算智能都是其最核心的部分, 而人工智能则是外层.

事实上, CI 和传统的 AI 只是智能的两个不同层次, 各自都有自身的优势和局限性, 相互之间只应该互补, 而不能取代.

大量实践证明, 只有把 AI 和 CI 很好地结合起来, 才能更好地模拟人类智能, 才是智能科学技术发展的正确方向.

### 1.1.3 计算智能的产生与发展

1992 年, 贝慈德克在《Approximate Reasoning》学报上首次提出了“计算智能”的概念.

1994 年 6 月底到 7 月初, IEEE 在美国佛罗里达州的奥兰多市召开了首届国际计算智能大会 (简称 WCCI'94). 会议第一次将神经网络、进化计算和模糊系统这三个领域合并在一起, 形成了“计算智能”这个统一的学科范畴.

在此之后, WCCI 大会就成了 IEEE 的一个系列性学术会议, 每 4 年举办一次. 1998 年 5 月, 在美国阿拉斯加州的安克雷奇市又召开了第 2 届计算智能国际会议 WCCI'98. 2002 年 5 月, 1 在美国夏威夷州首府火奴鲁鲁市又召开了第 3 届计算智能国际会议

WCCI'02. 此外, IEEE 还出版了一些与计算智能有关的刊物. 目前, 计算智能的发展得到了国内外众多的学术组织和研究机构的高度重视, 并已成为智能科学技术一个重要的研究领域.

## 1.2 神经计算

神经计算或叫神经网络, 是计算智能的重要基础和核心, 也是计算智能乃至智能科学技术的一个重要研究领域.

神经网络是受生物神经元启发构建的计算系统. 神经网络由许多独立的单元组成, 每个单元接收来自上一层单元的输入, 并将输出发送到下个单元(「单元」不一定是单独的物理存在; 它们可以被认为是计算机程序的不同组成部分). 单元的输出通常通过取输入的加权和通过某种简单的非线性转型, 神经网络的关键特性是基于经验修改与单元之间的链接比较相关权重.

常见误解—Stuart Russell

「神经网络是一种新型计算机」。在实践中, 几乎所有的神经网络都运行在普通的计算机架构上. 一些公司正在设计专用机器, 它们有时会被称作是「神经计算机」, 可以有效地运行神经网络, 但目前为止, 这类机器无法提供足够的优势, 值得花费大量时间去开发.

「神经网络像大脑一样工作」。事实上, 生物神经元的工作方式比神经网络复杂得多, 自然界存在很多种不同的神经元, 神经元的连接可以随时间进行改变, 大脑中也存在其他的机制, 可以影响动物的行为.

本节的主要内容包括:

### 1.2.1 神经计算基础

生物神经系统是人工神经网络的基础. 人工神经网络是对人脑神经系统的简化、抽象和模拟, 具有人脑功能的许多基本特征.

为方便对神经网络的进一步讨论, 下面先介绍:

**1.2.1.0.1 生物神经元的结构** 它由细胞体 (Soma)、轴突 (Axon) 和树突 (Dendrite) 三个主要部分组成

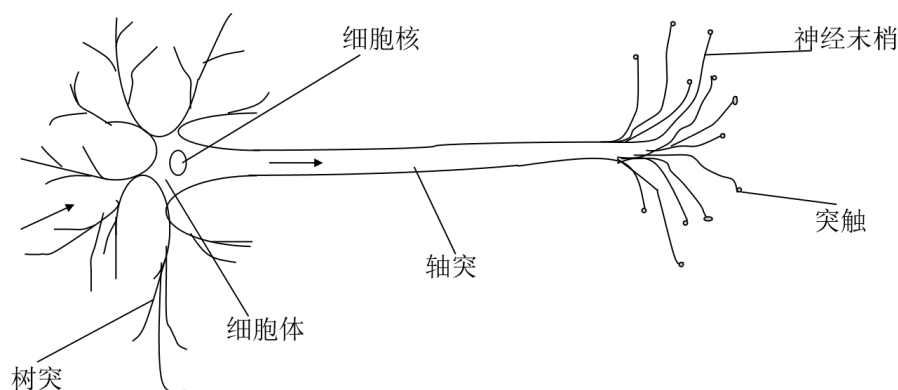


图 1-2

细胞体由细胞核、细胞质和细胞膜等组成, 其直径大约为 0.5-100 $\mu$ m 大小不等. 细胞体是神经元的主体, 用于处理由树突接受的其它神经元传来的信号, 其内部是细胞核, 外部是细胞膜, 细胞膜的外面是许多向外延伸出的纤维.

轴突是由细胞体向外延伸出的所有纤维中最长的一条分枝, 用来向外传递神经元产生的输出电信号.

- ▶ 每个神经元都有一条轴突, 其最大长度可达 1m 以上. 在轴突的末端形成了许多很细的分枝, 这些分枝叫神经末梢.
- ▶ 每一条神经末梢可以与其它神经元形成功能性接触, 该接触部位称为突触. 所谓功能性接触, 是指非永久性的接触, 这正是神经元之间传递信息的奥秘之处
- ▶ 树突是指由细胞体向外延伸的除轴突以外的其它所有分支. 树突的长度一般较短, 但数量很多, 它是神经元的输入端, 用于接受从其它神经元的突触传来的信号.

**1.2.1.0.2 生物神经元的功能** 根据神经生理学的研究, 生物神经元的 2 个主要功能是: 神经元的兴奋与抑制, 神经元内神经冲动的传导.

▶ ① 神经元的抑制与兴奋

**抑制状态**是指神经元在没有产生冲动时的工作状态. **兴奋状态**是指神经元产生冲动时的工作状态.

通常情况下, 神经元膜电位约为 -70 毫伏, 膜内为负, 膜外为正, 处于抑制状态. 当神经元受到外部刺激时, 其膜电位随之发生变化, 即膜内电位上升、膜外电位下降, 当膜内外的电位差大于阈值电位 (约 +40 毫伏) 时, 神经元产生冲动而进入兴奋状态.

**说明:** 神经元每次冲动的持续时间大约 1 毫秒左右, 在此期间, 即使刺激强度再增



加也不会引起冲动强度的增加. 神经元每次冲动结束后, 都会重新回到抑制状态. 如果神经元受到的刺激作用不能使细胞膜内外的电位差大于阈值电位, 则神经元不会产生冲动, 将仍处于抑制状态.

► ② 神经元内神经冲动的传导

神经冲动在神经元内的传导是一种电传导过程, 神经冲动沿神经纤维传导的速度却在 3.2—320km/s 之间, 且其传导速度与纤维的粗细、髓鞘的有无有一定关系. 一般来说, 有髓鞘的纤维的传导速度较快, 而无髓鞘的纤维的传导速度较慢.

### 1.2.1.0.3 人脑神经系统的联结机制 ① 人脑神经系统的联结规模

人类大脑中由 860—1012 亿个神经元所组成, 其中每个神经元大约有  $3 \times 10^4$  个突触. 小脑中的每个神经元大约有 105 个突触, 并且每个突触都可以与别的神经元的一个树突相连. 人脑神经系统就是由这些巨量的生物神经元经广泛并行互连所形成的一个高度并行性、非常复杂的神经网络系统.

#### ② 人脑神经系统的分布功能

人脑神经系的记忆和处理功能是有机的结合在一起的, 每个神经元既具有存储功能, 同时又具有处理能力. 从结构上看, 人脑神经系统又是一种分布式系统. 人们通过对脑损坏病人所做的神经生理学研究, 没有发现大脑中的哪一部分可以决定其余所有各部分的活动, 也没有发现在大脑中存在有用于驱动和管理整个智能处理过程的任何中央控制部分. 即人类大脑的各个部分是协同工作、相互影响的. 在大脑中, 不仅知识的存储是分散的, 而且其控制和决策也是分散的.

## 1.2.2 人工神经网络的互连结构

人工神经网络是由大量的人工神经元经广泛互联所形成的一种人工网络系统, 用以模拟人类神经系统的结构和功能.

### 1.2.2.0.1 人工神经元的结构

**1.2.2.0.2 生物神经元的结构** 人工神经元是对生物神经元的抽象与模拟, 图1-3是一个 MP 神经元模型, 它由细胞体 (Soma)、轴突 (Axon) 和树突 (Dendrite) 三个主要部分组成

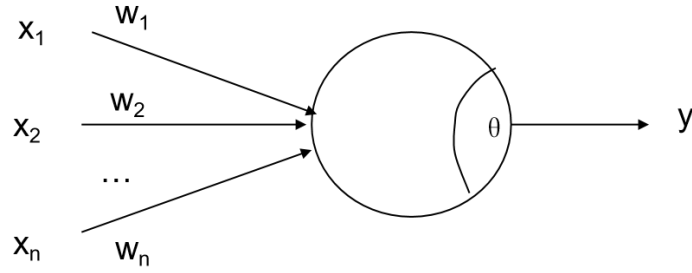


图 1-3 MP 神经元模型

**1.2.2.0.3 常用的人工神经元模型** 根据功能函数的不同, 可得到不同的神经元模型. 常用模型包括:

- ▶ **阈值型 (Threshold):** 这种模型的神经元没有内部状态, 作用函数  $f$  是一个阶跃函数, 他表示激活值  $\sigma$  和输出之间的关系.
- ▶ **分段线性强饱和型 (Linear Saturation):** 这种模型又称为**伪线性**, 其输入/输出之间在一定范围内满足线性关系, 一直延续到输出为最大值 1 为止. 但当达到最大值后, 输出就不再增.
- ▶ **S 型 (Sigmoid)**  
这是一种连续的神经元模型, 其输入输出特性常用指数、对数或双曲正切等 S 型函数表示. 它反映的是神经元的饱和特性.
- ▶ **子阈累积型 (Subthreshold Summation)**  
也是一个非线性函数, 当产生的激活值超过  $T$  值时, 该神经元被激活产生反响. 在线性范围内, 系统的反响是线性的.

人工神经网络的互连结构 (或称拓扑结构) 是指单个神经元之间的连接模式, 它是构造神经网络的基础, 也是神经网络诱发偏差的主要来源. 从互连结构的角度: 仅含输入层和输出层, 且只有输出层的神经元是可计算节点. 除拥有输入、输出层外, 还至少含有一个、或更多个隐含层的前向网络.

$$y_j = f \left( \sum_{i=1}^n w_{ij} x_i - \theta_j \right) \quad j = 1, 2, \dots, m \quad (1.1)$$

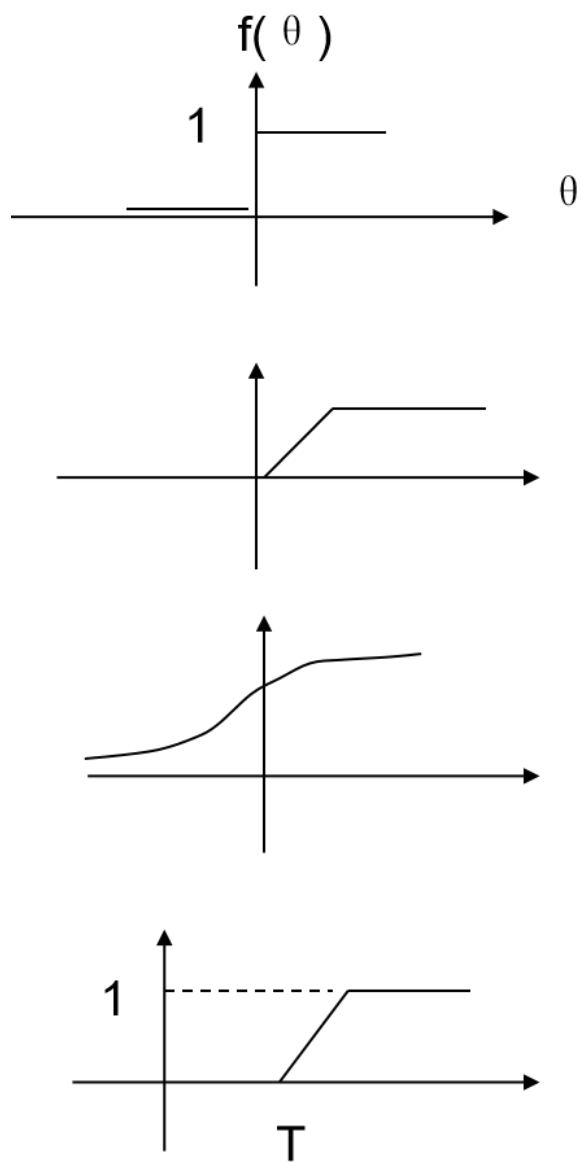


图 1-4 MP 神经元模型

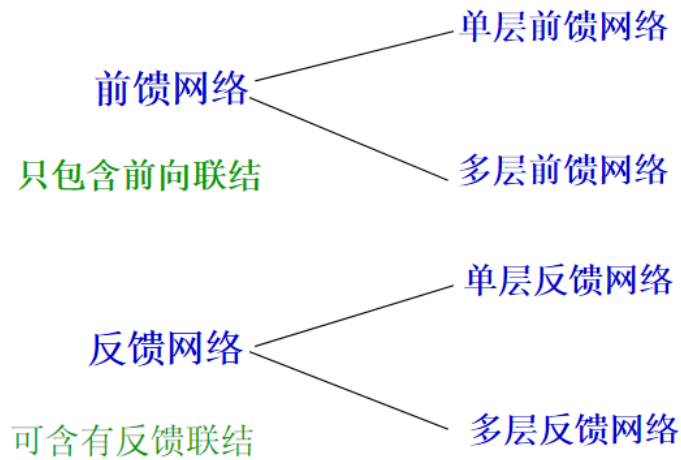


图 1-5 MP 神经元模型

**1.2.2.0.4 前馈网络** 指不拥有隐含层的反馈网络, 包括单层前馈网络和多层前馈网络.

单层前馈网络是指那种只拥有单层计算节点的前向网络. 它仅含有输入层和输出层, 且只有输出层的神经元是可计算节点, 如图1-6所示

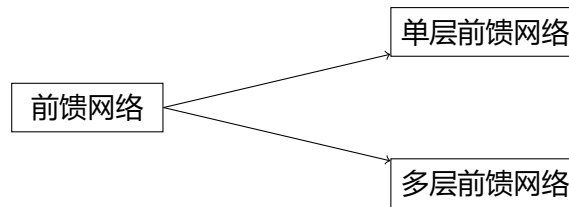


图 1-6 单层前馈网络结构

其中, 输入向量为  $X = (x_1, x_2, \dots, x_n)$ ; 输出向量为  $Y = (y_1, y_2, \dots, y_m)$ ; 输入层各个输入到相应神经元的连接权值分别是  $w_{ij}, i = 1, 2, \dots, n, j = 1, 2, \dots, m$ .

若假设各神经元的阈值分别是  $\theta_j, j = 1, 2, \dots, m$ , 则各神经元的输出  $y_j, j = 1, 2, \dots, m$  分别为:

$$y_j = f \left( \sum_{i=1}^n w_{ij} x_i - \theta_j \right) \quad j = 1, 2, \dots, m \quad (1.2)$$

其中, 由所有连接权值  $w_{ij}$  构成的连接权值矩阵  $W$  为:

$$W = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & & \vdots \\ w_{nl} & w_{n2} & \cdots & w_{nm} \end{pmatrix} \quad (1.3)$$

在实际应用中, 该矩阵是通过大量的训练示例学习而形成的.

**1.2.2.0.5 多层前馈网络** 多层前馈网络是指那种除拥有输入、输出层外, 还至少含有一个、或更多个隐含层的前馈网络.

隐含层是指由那些既不属于输入层又不属于输出层的神经元所构成的处理层, 也被称为中间层. 隐含层的作用是通过输入层信号的加权处理, 将其转移成更能被输出层接受的形式.

**1.2.2.0.6 多层前馈网络结构** 多层前馈网络的输入层的输出向量是第一隐含层的输入信号, 而第一隐含层的输出则是第二隐含层的输入信号, 以此类推, 直到输出层. 多层前馈网络的典型代表是 BP 网络.

**1.2.2.0.7 反馈网络** 反馈网络是指允许采用反馈联结方式所形成的神经网络, 图1-7. 所谓反馈联结方式是指一个神经元的输出可以被反馈至同层或前层的神经元.

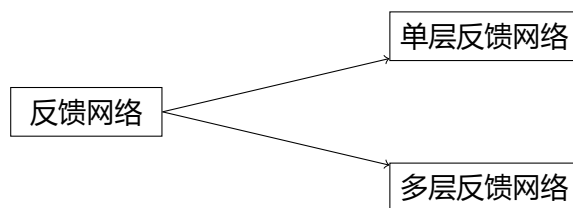


图 1-7 单层前馈网络结构

**1.2.2.0.8 反馈网络和前向网络不同** 前向网络属于非循环连接模式, 它的每个神经元的输入都没有包含该神经元先前的输出, 因此不具有“短期记忆”的性质.

反馈网络则不同, 它的每个神经元的输入都有可能包含有该神经元先前输出的反馈信息, 即一个神经元的输出是由该神经元当前的输入和先前的输出这两者来决定的, 这就有点类似于人类的短期记忆的性质.

反馈网络的典型例子是后面将要介绍的 Hopfield 网络.

## 1.3 人工神经网络的典型模型

至于基于神经网络的连接学习机制放到第??章学习部分讨论. 人工神经网络的模型是指对网络结构、联结权值和学习能力的总括. 常用的网络模型已有数十种. 例如:

- ▶ 传统的感知机模型,
- ▶ 具有误差反向传播功能的反向传播网络模型,
- ▶ 采用多变量插值的径向基函数网络模型,
- ▶ 建立在统计学习理论基础的支撑向量机网络模型,
- ▶ 采用反馈联接方式的反馈网络模型,
- ▶ 基于模拟退火算法的随机网络模型.

### 1.3.1 感知机 (Perceptron) 模型

感知器是美国学者罗森勃拉特 (Rosenblatt) 于 1957 年为研究大脑的存储、学习和认知过程而提出的一类具有自学习能力的神经网络模型, 其拓扑结构是一种分层前向网络. 它包括:

**1.3.1.0.1 单层感知器** 单层感知器是一种只具有单层可调节连接权值神经元的前向网络, 这些神经元构成了单层感知器的输出层, 是感知器的可计算节点.

在单层感知器中, 每个可计算节点都是一个线性阈值神经元. 当输入信息的加权和大于或等于阈值时, 输出为 1, 否则输出为 0 或 -1.

单层感知器的输出层的每个神经元都只有一个输出, 且该输出仅与本神经元的输入及联接权值有关, 而与其他神经元无关.

若假设各神经元的阈值分别是  $\theta_j, j = 1, 2, \dots, m$ , 则各神经元的输出  $y_j, j = 1, 2, \dots, m$  分别为

$$y_j = f \left( \sum_{i=1}^n w_{ij} x_i - \theta_j \right) \quad j = 1, 2, \dots, m \quad (1.4)$$

其中, 由所有连接权值  $w_{ji}$  构成的连接权值矩阵  $W$  为:

在实际应用中, 该矩阵是通过大量的训练示例学习而形成的

$$W = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & & \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{pmatrix} \quad (1.5)$$

输入向量为  $X = (x_1, x_2, \dots, x_n)$ ; 输出向量为  $Y = (y_1, y_2, \dots, y_m)$ ;

输入层各个输入到相应神经元的连接权值分别是  $w_{ij}, i = 1, 2, \dots, n, j = 1, 2, \dots, m$ .

使用感知器的主要目的是为了对外部输入进行分类. 罗森勃拉特已经证明, 如果外部输入是线性可分的 (指存在一个超平面可以将它们分开), 则单层感知器一定能够把它划分为两类. 其判别超平面由如下判别式确定:

$$\sum_{i=1}^n w_{ij} x_i - \theta_j = 0 \quad j = 1, 2, \dots, m \quad (1.6)$$

作为例子, 下面讨论用单个感知器实现逻辑运算的问题. 事实上, 单层感知器可以很好地实现“与”、“或”、“非”运算, 但却不能解决“异或”问题.

**例 1.1** “与”运算 ( $x_1 \wedge x_2$ )

输入		输出	超平面	阈值条件
$x_1$	$x_2$	$x_1 \wedge x_2$	$w_1 * x_1 + w_2 * x_2 - \theta = 0$	
0	0	0	$w_1 * 0 + w_2 * 0 - \theta < 0$	$\theta > 0$
0	1	0	$w_1 * 0 + w_2 * 1 - \theta < 0$	$\theta > w_2$
1	0	0	$w_1 * 1 + w_2 * 0 - \theta < 0$	$\theta > w_1$
1	1	1	$w_1 * 1 + w_2 * 1 - \theta \geq 0$	$\theta \leq w_1 + w_2$

图 1-8

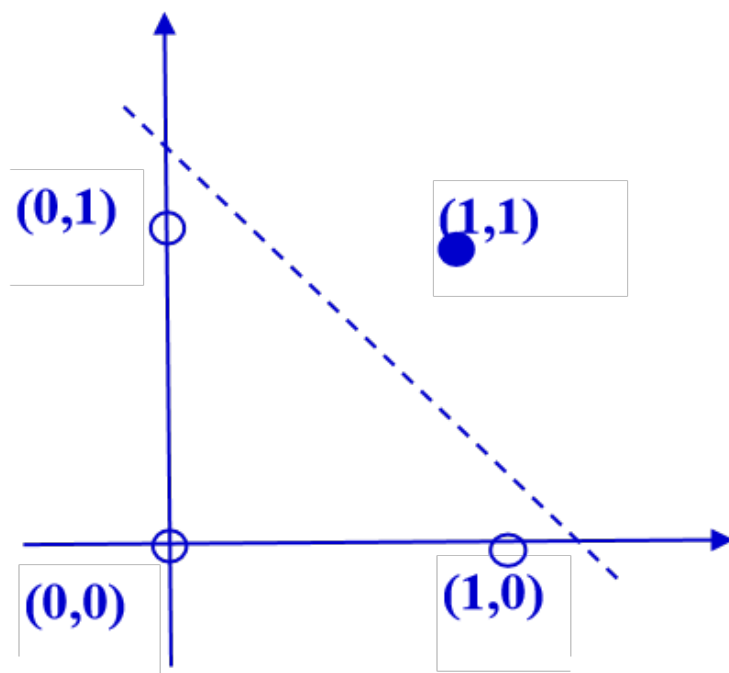


图 1-9 与运算问题

可以证明此表有解, 例如取  $w_1 = 1$ ,  $w_2 = 1$ ,  $\theta = 1.5$ , 其分类结果如图1-9所示. 其中, 输出为 1 的用实心圆, 输出为 0 的用空心圆. 后面约定相同.

#### 例 1.2 “非”运算 ( $\neg x_1$ )

此表也有解, 例如取  $w_1 = -1$ ,  $\theta = -0.5$ , 其分类结果如右图所示.

表 1-1 初始种群情况表

输入	输出	超平面	阈值条件
$x_1$	$\neg x_1$	$w_1 \cdot x_1 - \theta = 0$	$\theta = 0$
0	1	$w_1 \cdot 0 - \theta \geq 0$	$\theta \geq 0$
1	0	$w_1 \cdot 1 - \theta < 0$	$\theta < -w_1$

#### 例 1.3 “异或”运算 ( $x_1 \text{ XOR } x_2$ )



表 1-2 初始种群情况表

输入	输出	超平面	阈值条件
$x_1$	$x_2$	$x_1 \text{ XOR } x_2$	$w_1 * x_1 + w_2 * x_2 - \theta = 0$
0	0	0	$w_1 * 0 + w_2 * 0 - \theta < 0$
0	1	1	$w_1 * 0 + w_2 * 1 - \theta \geq 0$
1	0	1	$w_1 * 1 + w_2 * 0 - \theta \geq 0$
1	1	0	$w_1 * 1 + w_2 * 1 - \theta < 0$

此表无解, 即无法找到满足条件的  $w_1, w_2$  和  $\theta$ , 如图1-10所示. 因为异或问题是一个非线性可分问题, 需要用多层感知器来解决.

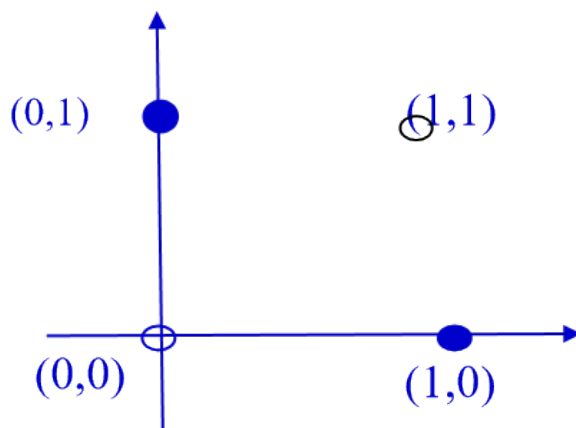


图 1-10

**1.3.1.0.2 多层感知器** 多层感知器是通过在单层感知器的输入、输出层之间加入一层或多层处理单元所构成的. 其拓扑结构与图1-11所示的多层前向网络相似, 差别也在于其计算节点的连接权值是可变的.

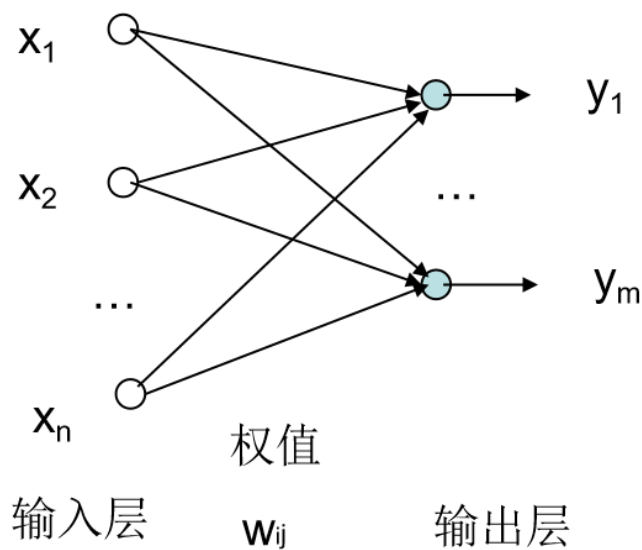


图 1-11

多层感知器的输入与输出之间是一种高度非线性的映射关系, 如图1-11所示的多层前向网络, 若采用多层感知器模型, 则该网络就是一个从  $n$  维欧氏空间到  $m$  维欧氏空间的非线性映射. 因此, 多层感知器可以实现非线性可分问题的分类.

**例 1.4** 对“异或”运算, 用图1-12所示的多层感知器即可解决.

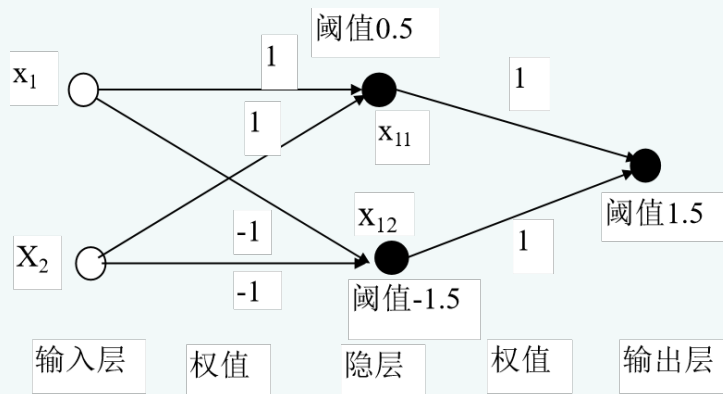


图 1-12

在图1-12中, 隐层神经元  $x_{11}$  所确定的直线方程为

$$1 * x_1 + 1 * x_2 - 0.5 = 0 \quad (1.7)$$

它可以识别一个半平面. 隐层神经元  $x_{12}$  所确定的直线方程为

$$1 * x_1 + 1 * x_2 - 1.5 = 0 \quad (1.8)$$

它也可以识别一个半平面.

输出层神经元所确定的直线方程为

$$1 * x_{11} + 1 * x_{12} - 1.5 = 0 \quad (1.9)$$

它相当于对隐层神经元  $x_{11}$  和  $x_{12}$  的输出作“逻辑与”运算, 因此可识别由隐层已识别的两个半平面的交集所构成的一个凸多边形, 如图1-13所示.

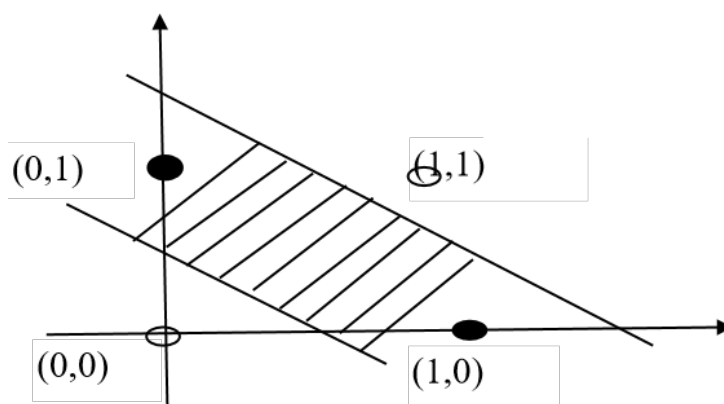


图 1-13

### 1.3.2 误差反向传播 (BP) 模型

误差反向传播 (Error Back Propagation) 网络通常简称为 BP(Back Propagation) 网络, 是由美国加州大学的鲁梅尔哈特和麦克莱兰在研究并行分布式信息处理方法, 探索人类认知微结构的过程中, 于 1985 年提出的一种网络模型. BP 网络的网络拓扑结构是多层前向网络, 如图1-14所示. 在 BP 网络中, 同层节点之间不存在相互连接, 层与层之间多采用全互连方式, 且各层的连接权值可调. BP 网络实现了明斯基的多层网络的设想, 是当今神经网络模型中使用最广泛的一种.

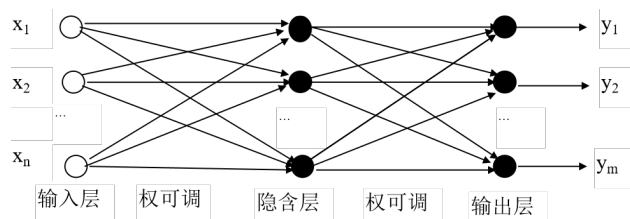


图 1-14 多层 BP 网络的结构

对 BP 网络需说明以下两点:

- 第一, BP 网络的每个处理单元均为非线性输入/输出关系, 其作用函数通常采用的是可微的 Sigmoid 函数, 如:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (1.10)$$

- 第二, BP 网络的学习过程是由工作信号的正向传播和误差信号的反向传播组成的. 所谓正向传播, 是指输入模式经隐层到输出层, 最后形成输出模式; 所谓误差反向传播, 是指从输出层开始逐层将误差传到输入层, 并修改各层联接权值, 使误差信号为最小的过程.

### 1.3.3 反馈网络 (Hopfield) 模型

Hopfield 网络是由美国加州工学院物理学家霍普菲尔特 1982 年提出来的一种单层全互连的对称反馈网络模型. 它可分为离散 Hopfield 网络和连续 Hopfield 网络, 限于篇幅, 本书重点讨论离散 Hopfield 网络.

**1.3.3.0.1 离散 Hopfield 网络的结构** 离散 Hopfield 网络是在非线性动力学的基础上由若干基本神经元构成的一种单层全互连网络, 其任意神经元之间均有连接, 并且是一种对称连接结构. 一个典型的离散 Hopfield 网络结构如图 1-15 所示. 离散 Hopfield 网络模型是一个离散时间系统, 每个神经元只有 0 和 1 (或 -1 和 1) 两种状态, 任意神经元  $i$  和  $j$  之间的连接权值为  $w_{ij}$ . 由于神经元之间为对称连接, 且神经元自身无连接, 因此有

$$w_{ij} = \begin{cases} w_{ji} & i \neq j \\ 0 & i = j \end{cases} \quad (1.11)$$

由该连接权值所构成的连接矩阵是一个对角为零的对称矩阵.

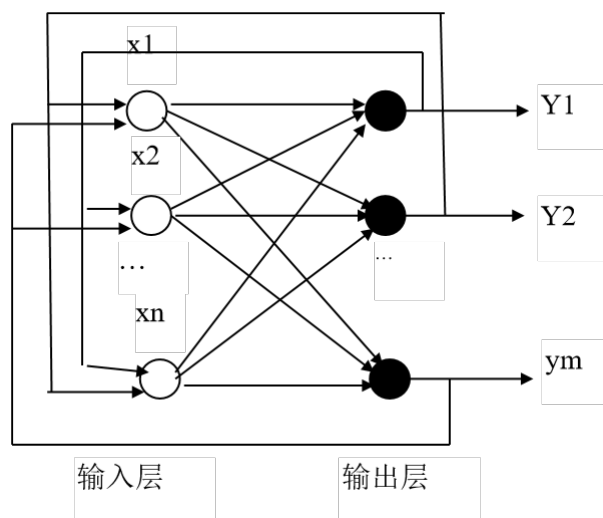


图 1-15

在 Hopfield 网络中, 虽然神经元自身无连接, 但由于每个神经元都与其他神经元相连, 即每个神经元的输出都将通过突触连接权值传递给别的神经元, 同时每个神经元又都接受其他神经元传来的信息, 这样对每个神经元来说, 其输出经过其他神经元后又有可能反馈给自己, 因此 Hopfield 网络是一种反馈神经网络。

## 1.4 其它神经学习算法

### 1.4.1 ELM

Fig. 1-16 shows the three steps of OP-ELM. The optimally pruned extreme learning machine (OP-ELM) is proposed in (EIML Group. The op-elm toolbox, Available online at <http://www.cis.hut.fi/projects/eiml/research/download> “corrupt” some of the neurons. As described at more length in [Miche2008OP, Miche2008A, Miche2010OP], it can be illustrated on a toy example as in Fig. 1: the plots give the ELM fit in light blue dots over the training points in black crosses. On the leftmost part of the figure, the fit by the ELM is good, but when a pure random noise variable is added, on the rightmost figure (the added noise variable is not pictured on the figure), the fit becomes loose and spread.

Indeed, the ELM is not designed to cope with such variables irrelevant to the problem at hand. In this spirit, the **OP-ELM** proposes a three-steps methodology, shortly described here, to address this issue. The idea is to build a wrapper around the original ELM, with a

neuron pruning strategy. For this matter, as can be seen in Fig. 2, the construction of the SLFN by the ELM is retained, and two steps are added afterwards. First comes a ranking of the neurons by a least angle regression (LARS [EfronHastie2004-32498]; in practice the **MRSR** [SimiläTikka2005-32497] implementation of LARS is used for it also applies to multi-output cases), which sorts them by their usefulness regarding the output. And then a leave-one-out criterion is used to determine how many of the-sorted-neurons should be kept for the final OP-ELM model structure.

The **LARS algorithm** is not detailed here since it is described and discussed at length (or more precisely the idea it implements, Lasso) in Section 3, but it has the property of providing an exact ranking of the hidden-layer neurons in the case of the OP-ELM, since the relation between the neurons and the output is linear (by design of the OP-ELM).

The **leave-one-out** (LOO) method is usually a costly approach to optimize a parameter since it requires to train the model on the whole data set but one sample, and evaluate on this sample, repeatedly for all the samples of the data set. In the OP-ELM structure though, the situation is linear (between the hidden layer and the output one), and the LOO error has a closed matrix form, given by **Allen's prediction sum of squares** (PRESS) [MADavid1972] (details of the computation of the PRESS LOO error are given in Section 4). This closed form allows for fast computation of the mean square error and hence of the output weights  $\mathbf{b}$ , making the OP-ELM still computationally efficient and more robust than the original ELM to irrelevant/correlated variables. Hence, the OP-ELM can be seen as a "regularized" ELM, by the use of a **LARS** approach, which is a  $L_1$  penalty on a regression problem, here.

Meanwhile, the decision over the final number of neurons to retain (by a **LOO criterion**) has shown potential instabilities (numerically), due to the nature of the matrix operations performed in the PRESS formula (see Section 4 for these calculations). The proposed solution in this paper is to use regularization in the calculations of the PRESS formula. In the following are reviewed the most well-known algorithms used to perform regularization, using a  $L_1$  and  $L_2$  (and jointly  $L_1$  and  $L_2$ ) penalty on the regression problem. The proposed approach in Section 4 combines both  $L_1$  and  $L_2$  penalties in the OP-ELM, to regularize the network.

#### 1.4.1.1 Tikhonov regularization

Tikhonov regularization, named for Andrey Tikhonov, is the most commonly used method of regularization of ill-posed problems. In statistics, the method is known as **ridge regression**, and with multiple independent discoveries, it is also variously known as the **Tikhonov-Miller**



图 1-16 Illustration of the three OP-ELM steps: the SLFN is first built using the ELM approach (random initialization of internal weights and biases); then a LARS algorithm is used to rank the neurons of the hidden layer; finally the selection of the optimal number of neurons for the OP-ELM model is performed using a Leave-One-Out criterion.

**method**, the **Phillips-Twomey method**, the **Constrained linear inversion method**, and the method of **linear regularization**. It is related to the **Levenberg-Marquardt algorithm** for **non-linear least-squares** problems.

Suppose that for a known matrix  $A$  and vector  $b$ , we wish to find a vector  $x$  such that

$$Ax = b. \quad (1.12)$$

The standard approach is ordinary least squares linear regression. However, if no  $x$  satisfies the equation or more than one  $x$  does—that is the solution is not unique—the problem is said not to be well posed. In such cases, ordinary least squares estimation leads to an overdetermined (over-fitted), or more often an underdetermined (under-fitted) system of equations. Most real-world phenomena have the effect of **low-pass filters** in the forward direction where  $A$  maps  $x$  to  $b$ . Therefore, in solving the inverse-problem, the inverse mapping operates as a high-pass filter that has the undesirable tendency of amplifying noise (eigenvalues / singular values are largest in the reverse mapping where they were smallest in the forward mapping). In addition, ordinary least squares implicitly nullifies every element of the reconstructed version of  $x$  that is in the null-space of  $A$ , rather than allowing for a model to be used as a prior for  $x$ . Ordinary least squares seeks to minimize the sum of squared residuals, which can be compactly written as

$$\|Ax - b\|^2 \quad (1.13)$$

where  $\|\cdot\|$  is the Euclidean norm. In order to give preference to a particular solution with desirable properties, a regularization term can be included in this minimization:

$$\|Ax - b\|^2 + \|\Gamma x\|^2 \quad (1.14)$$

for some suitably chosen **Tikhonov matrix**,  $\Gamma$ . In many cases, this matrix is chosen as a multiple of the identity matrix ( $\Gamma = \alpha I$ ), giving preference to solutions with smaller norms; this is known as  $L_2$  regularization [Ng2004-32636]. In other cases, **lowpass operators** (e.g., a difference operator or a **weighted Fourier operator**) may be used to enforce smoothness if the underlying vector is believed to be mostly continuous. This regularization improves the conditioning of the problem, thus enabling a direct numerical solution. An explicit solution, denoted by  $\hat{x}$ , is given by:

$$\hat{x} = (A^T A + \Gamma^T \Gamma)^{-1} A^T b \quad (1.15)$$

The effect of regularization may be varied via the scale of matrix  $\Gamma$ . For  $\Gamma = 0$  this reduces to the unregularized least squares solution provided that  $(A^T A)^{-1}$  exists.

$L_2$  regularization is used in many contexts aside from linear regression, such as classification with logistic regression or support vector machines [Fan2008LIBLINEAR], and matrix factorization [Guan2012Online].

**Tikhonov regularization** has been invented independently in many different contexts. It became widely known from its application to integral equations from the work of Andrey Tikhonov and David L. Phillips. Some authors use the term **Tikhonov-Phillips regularization**. The finite-dimensional case was expounded by Arthur E. Hoerl, who took a statistical approach, and by Manus Foster, who interpreted this method as a **Wiener-Kolmogorov (Kriging) filter**. Following Hoerl, it is known in the statistical literature as **ridge regression**.

#### Generalized Tikhonov regularization

For general multivariate normal distributions for  $x$  and the data error, one can apply a transformation of the variables to reduce to the case above. Equivalently, one can seek an  $x$  to minimize

$$\|Ax - b\|_P^2 + \|x - x_0\|_Q^2 \quad (1.16)$$

where we have used  $\|x\|_Q^2$  to stand for the weighted norm  $x^T Q x$  (compare with the Mahalanobis distance). In the Bayesian interpretation  $P$  is the inverse covariance matrix of  $b$ ,  $x_0$  is the expected value of  $x$ , and  $Q$  is the inverse covariance matrix of  $x$ . The Tikhonov matrix is then given as a factorization of the matrix  $Q = \Gamma^T \Gamma$  (e.g. the **Cholesky factorization**), and is considered a **whitening filter**.

This generalized problem has an optimal solution  $x^*$  which can be solved explicitly using the formula

$$x^* = (A^T P A + Q)^{-1} (A^T P b + Q x_0). \quad (1.17)$$



or equivalently

$$\mathbf{x}^* = \mathbf{x}_0 + (\mathbf{A}^T \mathbf{P} \mathbf{A} + \mathbf{Q})^{-1} (\mathbf{A}^T \mathbf{P} (\mathbf{b} - \mathbf{A} \mathbf{x}_0)). \quad (1.18)$$

### 1.4.2 TROP-ELM

**TROP-ELM:** A double-regularized ELM using LARS and Tikhonov regularization [MichevanHeeswijk2011-30641], 2011.

Fig. 1-17 shows the proposed regularized OP-ELM (TROP-ELM).

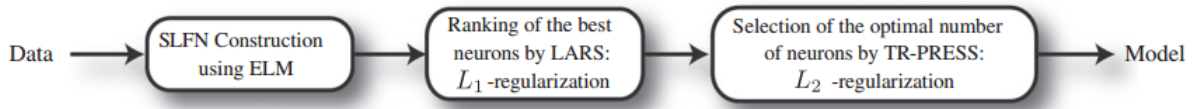


图 1-17 The proposed regularized OP-ELM (TROP-ELM) as a modification of Fig. 1-16.

### 1.4.3 Allen's PRESS(prediction sum of squares)

The original PRESS formula used in the OP-ELM was proposed by Allen in [MADavid1972]. The original PRESS formula can be expressed as

$$\text{MSE}^{\text{TR-PRESS}} = \frac{1}{n} \sum_{i=1}^n \left( \frac{y_i - \mathbf{x}_i (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{x}_i^T \mathbf{y}}{1 - \mathbf{x}_i (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{x}_i^T} \right)^2 \quad (1.19)$$

which means that each observation is “predicted” using the other  $n - 1$  observations and the residuals are finally squared and summed up. Algorithm ?? proposes to implement this formula in an efficient way, by matrix computations.

**Algorithm ??.** 1: Compute the utility matrix  $\mathbf{C} = (\mathbf{X}^T \mathbf{X})^{-1}$

2: And  $\mathbf{P} = \mathbf{X} \mathbf{C}$ ;

3: Compute the pseudo-inverse  $\mathbf{w} = \mathbf{C} \mathbf{X}^T \mathbf{y}$ ;

4: Compute the denominator of the PRESS  $D = \mathbf{1} - \text{diag}(\mathbf{P} \mathbf{X}^T)$ ;

5: And finally the PRESS error  $e = \frac{\mathbf{y} - \mathbf{X} \mathbf{w}}{D}$ ;

6: Reduced to a  $\text{MSE}^{\text{TR-PRESS}} = \frac{1}{n} \sum_{i=1}^n \epsilon_i^2$

The main drawback of this approach lies in the use of a pseudo-inverse in the calculation (in the **Moore-Penrose sense**), which can lead to numerical instabilities if the data set  $\mathbf{X}$  is not full rank. This is unfortunately very often the case, with real world data sets. The following approach proposes two improvements on the computation of the original **PRESS**: regularization and fast matrix calculations.

## 1.4.3.1 Tikhonov-regularized PRESS (TR-PRESS)

In [Golub1979Generalized], Golub et al. note that the singular value decomposition (SVD) approach to compute the PRESS statistic is preferable to the traditional pseudo-inverse mentioned above, for numerical reasons. In this very same paper is proposed a generalization of Allen's PRESS, as the **generalized cross-validation** (GCV) method, which is technically superior to the original **PRESS**, for it can handle cases where the data is extremely badly defined—for example if all  $X$  entries are 0 except the diagonal ones.

In practice, from our experiments, while the GCV is supposedly superior, it leads to identical solutions with an increased computational time, compared to the original **PRESS** and the Tikhonov regularized version of PRESS presented below. Algorithm ?? gives the computational steps used, in matrix form, to determine the  $\text{MSE}^{\text{TR-PRESS}}$  from

$$\text{MSE}^{\text{TR-PRESS}} = \frac{1}{n} \sum_{i=1}^n \left( \frac{y_i - \mathbf{x}_i(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{x}_i^T \mathbf{y}}{1 - \mathbf{x}_i(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{x}_i^T} \right)^2 \quad (1.20)$$

which is the regularized version of Eq. (18). The notation  $A \circ B$  denotes element-wise product between matrices  $A$  and  $B$  (Schur product). It is used in step 4 of Algorithm 2 for it is faster than standard matrix product.

Decompose  $X$  by SVD:  $X = USV^T$ ;

Compute the products (used later):  $A = XV$  and  $B = U^T y$ ;

repeat

Using the SVD of  $X$ , compute the  $C$  matrix by:

Compute the  $P$  matrix by:  $P = CB$ ;

Compute  $D$  by:  $D = \mathbf{1} - \text{diag}(CU^T)$ ;

Evaluate  $e = \frac{y-P}{D}$  and the actual MSE by  $\text{MSE}^{\text{TR-PRESS}} = \frac{1}{n} \sum_{i=1}^n \epsilon_i^2$ .

**until** convergence on  $\lambda$  is achieved

Keep the best  $\text{MSE}^{\text{TR-PRESS}}$  and the  $\lambda$  value associated. **Algorithm ??.** **Tikhonov-regularized PRESS.** In practice, the REPEAT part of this algorithm (convergence for 1) is solved by a **Nelder-Mead approach** [Nelder1965A], a.k.a. downhill simplex.

Globally, the algorithm uses the SVD of  $X$  to avoid computational issues, and introduces the Tikhonov regularization parameter in the calculation of the pseudo-inverse by the SVD. This specific implementation happens to run very quickly, thanks to the pre-calculation of utility matrices ( $A$ ,  $B$  and  $C$ ) before the optimization of  $\lambda$ .

In practice, the optimization of  $\lambda$  in this algorithm is performed by a **Nelder - Mead minimization approach** [Nelder1965A], which happens to converge very quickly on this

problem (fminsearch function in Matlab). Through the use of this modified version of PRESS, the OP-ELM has an  $L_2$  penalty on the regression weights (regression between the hidden and output layer), for which the neurons have already been ranked using an  $L_1$  penalty. Fig. 1-17 is a modified version of Fig. 1-16 illustrating the **TROP-ELM** approach.

#### 1.4.4 区间二型 RVFL

Now we use the new learning paradigm to train the RVFL network. Given a set of additional privileged information  $\{\tilde{x}_i \in \mathbb{R}^d; i = 1, \dots, N\}$  in the training process, the training data becomes  $\{(x_i, \tilde{x}_i, y_i) | x_i \in \mathbb{R}^n, \tilde{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}^m; i = 1, \dots, N\}$ . In this new training set,  $x_i \in X$  is the original feature and in general the privileged feature  $\tilde{x}_i \in \tilde{X}$  belongs to the privileged feature space  $\tilde{X}$  that is different from the original feature space  $X$ .

The model structure of the RVFL is so simple, why does the RVFL work well for most tasks? Giryes et. al. [42] give a possible theoretical explanation for this open problem. To provide this explanation, first of all, Giryes et. al. reveal the essence of training in learning algorithms. Generally speaking, the angles of samples between different classes are larger than that of samples within the same class [43]. Therefore, the role of training in learning algorithms is that 'the angles between points from different classes are penalized more than the angles between the points in the same class' [42]. Moreover, in this big data era, due to the highly complicated model architecture, it is quick difficult to tune all parameters in leaning process, which needs extremely high computational cost. To tackle this problem, randomization is an ideal choice for some learning algorithms, leading to cheaper computational cost. A great random initialization allows the learning algorithm to be a universal one before training. Now we revisit the RVFL. The RVFL uses the hybrid strategy to train the entire network. In the RVFL, the random initial parameters between the input layer and the enhancement nodes handle well input samples having distinguishable angles, and the turned output weights further deal with the remaining data.

The RVFL network is a classical single layer feedforward neural network, and the architecture of the RVFL is shown in Figure 1-18. The RVFL initializes randomly all weights and biases between the input layer and enhancement nodes, and then these parameters are fixed and do not need to be tuned during the training stage. The output weights on red solid lines in Figure 1-18 can be calculated by the Moore-Penrose pseudoinverse [PaoPhillips1995-6471, IgelnikPao1995-6470] or the ridge regression [Bishop2012-6469]. Moreover, the direct link between the input layer and the output layer is an effective and simple regularization technique

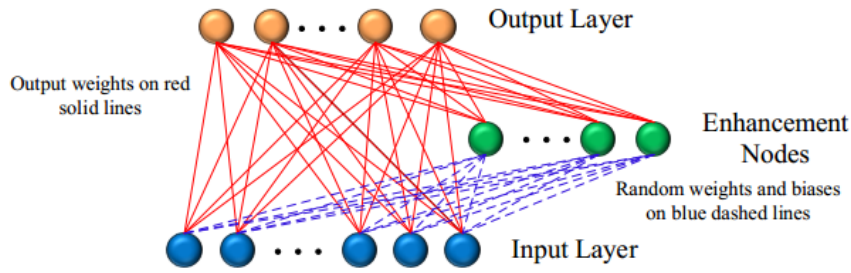


图 1-18 The architecture of the RVFL network.

preventing RVFL networks from overfitting.

The RVFL+ is an unified learning algorithm for all binary, multiclass classification and regression, and the output function in (19) can be straightforwardly applied for all three cases.

1. Binary Classification: The predicted label of the test sample is determined by

$$\hat{y} = \text{sign}(f_{test}(z)) \quad (1.21)$$

2. Multiclass Classification: We adopt the one-vs.-all (OvA) strategy to determine the predicted label in the multiclass classification. Let  $f_{test}^k(z)$  be the output function of the  $k$ -th output nodes. The predicted label of the test sample is determined by

$$\hat{y} = \arg \max_{k \in 1, \dots, m} f_{test}^k(z) \quad (1.22)$$

3. Regression: The predicted value is equal to the output function  $f_{test}(z)$  of the RVFL+

$$\hat{y} = f_{test}(z) \quad (1.23)$$

Following the formula of the SVM+ in [VAPNIK2009544], we can write the RVFL+ as

$$\begin{aligned} \min_{\mathbf{w}, \tilde{\mathbf{w}}} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + \frac{\gamma}{2} \|\tilde{\mathbf{w}}\|_2^2 + C \sum_{i=1}^N \zeta_i(\tilde{\mathbf{w}}, \tilde{h}(\tilde{\mathbf{x}}_i)) \\ \text{s.t.} \quad & \mathbf{h}(\tilde{\mathbf{x}}_i) \mathbf{w} = \mathbf{y}_i - \zeta_i(\tilde{\mathbf{w}}; \tilde{h}(\tilde{\mathbf{x}}_i)), \forall i \leq N. \end{aligned} \quad (1.24)$$

where  $\gamma$  is a regularization coefficient. Likewise to  $\mathbf{h}(\mathbf{x}_i)$ ,  $\tilde{\mathbf{h}}(\tilde{\mathbf{x}}_i)$  is also an enhanced layer output vector corresponding to the privileged feature  $\tilde{\mathbf{x}}_i$ , which can be calculated in the same fashion.  $\zeta_i(\tilde{\mathbf{w}}, \tilde{\mathbf{h}}(\tilde{\mathbf{x}}_i))$  is the correcting function (or slack function) in the privileged feature space, and  $\tilde{\mathbf{w}}$  is an output weight vector for the correcting function.

$$\zeta_i(\tilde{\mathbf{w}}, \tilde{\mathbf{h}}(\tilde{\mathbf{x}}_i)) = \tilde{\mathbf{h}}(\tilde{\mathbf{x}}_i)\tilde{\mathbf{w}} \quad (1.25)$$

Substituting (1.25) into (1.24), we have the primal form of the RVFL+ as follows.

$$\begin{aligned} \min_{\mathbf{w}, \tilde{\mathbf{w}}} & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{\gamma}{2}\|\tilde{\mathbf{w}}\|_2^2 + C \sum_{i=1}^N \tilde{\mathbf{h}}(\mathbf{x}_i)\tilde{\mathbf{w}} \\ \text{s.t. } & \mathbf{h}(\tilde{\mathbf{x}}_i)\mathbf{w} = \mathbf{y}_i - \tilde{\mathbf{h}}(\mathbf{x}_i)\tilde{\mathbf{w}}, \forall i \leq N. \end{aligned} \quad (1.26)$$

From (1.26), we note that the RVFL+ minimizes the objective function over both  $\mathbf{w}$  and  $\tilde{\mathbf{w}}$ . Therefore, Not only the original features but also the privileged information determine meanwhile the separating hyperplane of the RVFL+ during the training stage.

Moreover, in contrasts to the primal form of the SVM+ in [VAPNIK2009544], the correcting function of the RVFL+ is either positive or negative. In other words, the RVFL+ does not consider a group of constraints  $\zeta_i(\tilde{\mathbf{h}}, b, \phi(\mathbf{x}_i)) \geq 0 (i = 1, \dots, N)$ . As a result, the number of constraints in the RVFL+ is at least  $N$  less than that of the SVM+ for the binary classification, which results that the RVFL+ has much milder optimization constraints than the SVM+.

Furthermore, in order to address the optimization problem in (1.26), we construct the Lagrangian function  $\mathcal{L}(\mathbf{w}, \tilde{\mathbf{w}}, \boldsymbol{\lambda})$  as

$$\min_{\mathbf{w}, \tilde{\mathbf{w}}, \boldsymbol{\lambda}} \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{\gamma}{2}\|\tilde{\mathbf{w}}\|_2^2 + C \sum_{i=1}^N \tilde{\mathbf{h}}(\mathbf{x}_i)\tilde{\mathbf{w}} - \sum_{i=1}^N \lambda_i (\mathbf{h}(\tilde{\mathbf{x}}_i)\mathbf{w} - \mathbf{y}_i + \tilde{\mathbf{h}}(\mathbf{x}_i)\tilde{\mathbf{w}}), \quad (1.27)$$

where  $\boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_N]^T$  are Lagrange multipliers. To find solutions, we use the KKT condition to calculate the saddle points of the Lagrangian function  $\mathcal{L}(\mathbf{w}, \tilde{\mathbf{w}}, \boldsymbol{\lambda})$  with respect to  $\mathbf{w}$ ,  $\tilde{\mathbf{w}}$  and  $\boldsymbol{\lambda}$ .

$$\frac{\partial \mathcal{L}(\mathbf{w}, \tilde{\mathbf{w}}, \boldsymbol{\lambda})}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \mathbf{H}^T \boldsymbol{\lambda}, \quad (1.28)$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, \tilde{\mathbf{w}}, \boldsymbol{\lambda})}{\partial \tilde{\mathbf{w}}} = 0 \Rightarrow \tilde{\mathbf{w}} = \frac{1}{\gamma}(\tilde{\mathbf{H}}^T \boldsymbol{\lambda} - \tilde{\mathbf{H}}^T \mathbf{C} \mathbf{1}), \quad (1.29)$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, \tilde{\mathbf{w}}, \boldsymbol{\lambda})}{\partial \lambda_i} = 0 \Rightarrow \mathbf{h}(\mathbf{x}_i)\mathbf{w} - \mathbf{y}_i + \tilde{\mathbf{h}}(\mathbf{x}_i)\tilde{\mathbf{w}} = 0, 1 \leq i \leq N. \quad (1.30)$$

where  $\mathbf{1} \in \mathbb{R}^{N \times m}$  is an identify matrix.  $\tilde{\mathbf{H}}$  is also a concatenated output matrix from the enhancement nodes, which corresponds to the privileged features.

Substituting (1.28) and (1.29) into (1.30), we have

$$\mathbf{H}\mathbf{H}^T\boldsymbol{\lambda} + \frac{1}{\gamma}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^T(\boldsymbol{\lambda} - \mathbf{C}\mathbf{1}) = \mathbf{Y} \quad (1.31)$$

We can further reformulate (1.31) as

$$\left(\mathbf{H}\mathbf{H}^T + \frac{1}{\gamma}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^T\right)\boldsymbol{\lambda} = \mathbf{Y} - \frac{\mathbf{C}\mathbf{1}}{\gamma}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^T \quad (1.32)$$

Combining (1.28) and (1.32), we obtain the closed-form solution to the RVFL+ as follows.

$$\mathbf{w} = \mathbf{H}^T \left(\mathbf{H}\mathbf{H}^T + \frac{1}{\gamma}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^T\right)^{-1} \left(\mathbf{Y} - \frac{\mathbf{C}\mathbf{1}}{\gamma}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^T\right) \quad (1.33)$$

According to the ridge regression [Bishop2012-6469], we also impose an additional term  $\frac{\mathbf{I}}{C}$  in order to avoid singularity and guarantee the stability of the RVFL+. As a result, we can achieve the aftermost closed-form solution to the RVFL+ as

$$\mathbf{w} = \mathbf{H}^T \left(\mathbf{H}\mathbf{H}^T + \frac{1}{\gamma}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^T + \frac{\mathbf{I}}{C}\right)^{-1} \left(\mathbf{Y} - \frac{\mathbf{C}\mathbf{1}}{\gamma}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^T\right) \quad (1.34)$$

Consequently, the output function of the RVFL+ is defined as

$$f(x) = \mathbf{h}(x)\mathbf{w} = \mathbf{h}(x)\mathbf{H}^T \left(\mathbf{H}\mathbf{H}^T + \frac{1}{\gamma}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^T + \frac{\mathbf{I}}{C}\right)^{-1} \left(\mathbf{Y} - \frac{\mathbf{C}\mathbf{1}}{\gamma}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^T\right). \quad (1.35)$$

In addition, we can obtain straightforwardly the output function  $f_{test}(z) = \mathbf{h}(z)\mathbf{w}$  in the test stage, when using the test data  $z$  instead of the training data  $x$ .

#### 1.4.4.1 区间二型模糊集

本节将随机向量函数链接网络 (RVFL) 推广到了张量结构, 增强节点使用了区间二型模糊集来扩展非线性激活函数, 主要目的是用二型模糊集来捕捉输入输出关系中蕴含的高阶信息. 提出了两种随机向量函数链接网络, 一种是区间二型随机向量函数链接网络, 另一种是基于张量的二型随机向量函数链接网络. 区间二型随机向量函数链接网络和随机向量函数链接网络类似, 不同之处在于其输出增强节点的输出是不确定加权方法的解模糊结果. 张量的二型随机向量函数链接网络也使用了不确定加权方法, 同时还使用了下隶属函数值和上隶属函数值, 由此形成了一个 4 阶张量. 基于这个结构, 基于 Einstein 积的偶数阶张量的 Moore-Penrose 逆被用来求解张量方程. 此外, 权重向量由一个平衡因子综合得到, 它是 RVFL 网络结果和张量方程结果的加权平均. 最后, 分别在三个非线性测试函数、非线性系统识别问题和四个回归问题上验证了给出的算法.

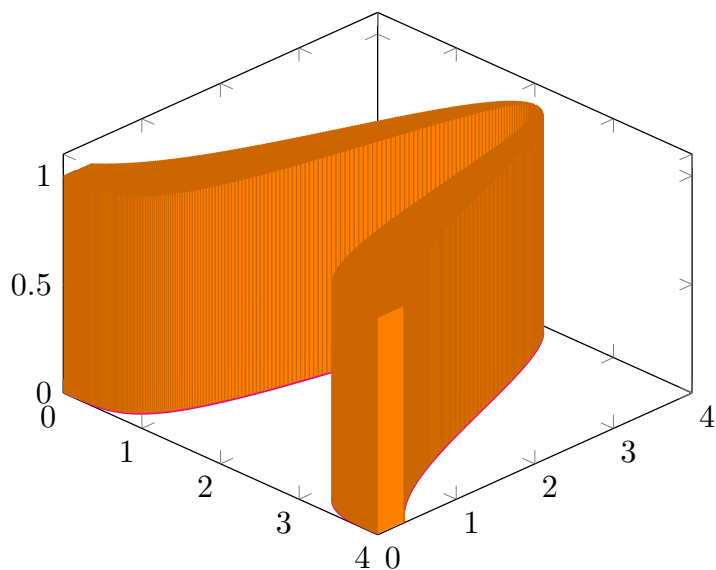


图 1-19 区间二型模糊集

高斯区间二型模糊集有两种定义形式, 不确定均值或者是不确定方差. 本节使用的区间二型模糊集上隶属和下隶属的均值相同. 图 1-19 给出了均值相同、方差不同的区间二型模糊集. 隶属函数按如下的方式定义

$$\bar{\mu}_i(x_i) = \exp\left(-\frac{(x_i - m_i)^2}{\bar{\sigma}_i^2}\right), \quad (1.36)$$

$$\underline{\mu}_i(x_i) = \exp\left(-\frac{(x_i - m_i)^2}{\underline{\sigma}_i^2}\right), \quad (1.37)$$

其中  $m_i$ ,  $\bar{\sigma}_i$  和  $\underline{\sigma}_i$  ( $1 \leq i \leq L$ ) 分别是二型模糊集的下隶属函数和上隶属函数的均值和方差. 下隶属函数和上隶属函数之间的隶属度是 1 的二型模糊集就是区间二型模糊集.

图 1-20 给出了基于张量积的区间二型随机向量函数链接网络 (TT2-RVFL), 其中  $\tilde{g}_i(\cdot)$  ( $i = 1, 2, \dots, L$ ) 是区间二型模糊激活函数. 不同于 RVFL, TT2-RVFL 使用区间二型模糊集, 扩展了 RVFL 的增强节点部分, 因此, 区间二型模糊集的输出可以看成是原来两个激活函数的输出结果. RVFL 网络中的乘法需要修正, 对于高维结构的 TT2-RVFL, 增强型节点使用了张量的 Einstein 积运算. TT2-RVFL 的输出模型如图 1-20:

$$\mathbf{Y} = \mathcal{A} *_N \mathbf{X} + \mathbf{H}\mathbf{\Omega}, \quad (1.38)$$

其中  $\mathcal{A} *_N \mathbf{X}$  为增强节点的张量乘积,  $\mathbf{H}\mathbf{\Omega}$  为线性部分的输出,  $\mathbf{X}$  为增强节点部分的权重矩阵; 类似于 RVFL,  $\mathbf{H}$  是输入数据矩阵, 单列输出的数据需要复制一次, 以便与张量和矩阵的运算结果相容,  $\mathbf{\Omega}$  是线性部分的权矩阵, 且  $\mathcal{A}$  是一个张量, 它是由叠加区

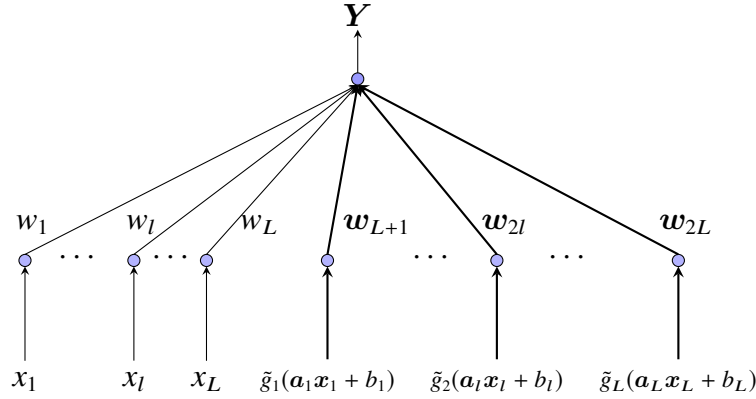


图 1-20 TT2-RVFL 的结构

间二型模糊集的映射结果得到的张量. 对于 TT2-RVFL,  $\mathbf{Y}$ ,  $\mathbf{X}$  和  $\mathbf{\Omega}$  需要特别考虑, 以便和后件形成的张量方程相容. 下一节将介绍由三个变量组成的张量结构.

#### 1.4.4.2 TT2-RVFL

TT2-RVFL RVFL 常使用 Radbas ( $y = \exp(-s^2)$ ), Sine ( $y = \sin(s)$ ), Sigmoid ( $y = \frac{1}{1+e^{-s}}$ ) 和 Tribas ( $y = \max(1 - |s|, 0)$ ) 这几类激活函数, 其中  $s$  和  $y$  分别表示输入和输出变量. 对于 TT2-RVFL, 增强节点使用了区间二型模糊集. RVFL 的 Radbas 激活函数被推广成区间二型模糊集, 扩展后的 RVFL 使用了区间二型模糊集, 扩展后的激活函数简记为 IT2Radbas, 它是 (1.36) 和 (1.37) 的变种, 记为  $\tilde{g}_i(\cdot)$ , 可以用下面的式子表示为

$$\bar{g}_i(x_i) = \exp(-k_1 s^2), \quad (1.39)$$

$$\underline{g}_i(x_i) = \exp(-k_2 s^2), \quad (1.40)$$

其中  $s = x_i - m_i$ ,  $k_1 = \frac{1}{\bar{\sigma}_i^2}$ ,  $k_2 = \frac{1}{\underline{\sigma}_i^2}$  ( $i = 1, 2, \dots, L$ ).

对于测试集  $\{D_t\}_{t=1}^N$ , 其中  $D_t = (\mathbf{x}_t, y_t)$ ,  $\mathbf{x}_t = (x_{t1}, x_{t2}, \dots, x_{tN}) \in \mathbb{R}^N$ ,  $y_t \in \mathbb{R}$  且



$\mathbf{y} = [y_1, \dots, y_N]^T$ . 下隶属函数矩阵  $\underline{\Phi} \in \mathbb{R}^{N \times 2 \times L \times 1}$  可用下式构建

$$\underline{\Phi}_{:, :, 1, 1} = \begin{bmatrix} \underline{g}_1(\mathbf{a}_{11}\mathbf{x}_1 + b_{11}) & \underline{g}_1(\mathbf{a}_{12}\mathbf{x}_1 + b_{12}) \\ \vdots & \vdots \\ \underline{g}_1(\mathbf{a}_{11}\mathbf{x}_N + b_{11}) & \underline{g}_1(\mathbf{a}_{12}\mathbf{x}_N + b_{12}) \end{bmatrix},$$

$$\vdots$$

$$\underline{\Phi}_{:, :, L, 1} = \begin{bmatrix} \underline{g}_L(\mathbf{a}_{L1}\mathbf{x}_1 + b_{L1}) & \underline{g}_L(\mathbf{a}_{L2}\mathbf{x}_1 + b_{L2}) \\ \vdots & \vdots \\ \underline{g}_L(\mathbf{a}_{L1}\mathbf{x}_N + b_{L1}) & \underline{g}_L(\mathbf{a}_{L2}\mathbf{x}_N + b_{L2}) \end{bmatrix},$$

其中  $b_{il}$  和  $\mathbf{a}_{il} = [w_{i1}, w_{i2}, \dots, w_{iK}]$  ( $i = 1, 2, \dots, L; l = 1, 2$ ) 是随机生成的偏置和输入权重. 下隶属函数矩阵使用下隶属函数来逼近输入  $\mathbf{x}_t$  和期望输出  $y_t$  的关系. 上隶属函数矩阵  $\bar{\Phi} \in \mathbb{R}^{N \times 2 \times L \times 1}$  可以用类似的方式建立, 具有如下形式

$$\bar{\Phi}_{:, :, 1, 2} = \begin{bmatrix} \bar{g}_1(\mathbf{a}_{11}\mathbf{x}_1 + b_{11}) & \bar{g}_1(\mathbf{a}_{12}\mathbf{x}_1 + b_{12}) \\ \vdots & \vdots \\ \bar{g}_1(\mathbf{a}_{11}\mathbf{x}_N + b_{11}) & \bar{g}_1(\mathbf{a}_{12}\mathbf{x}_N + b_{12}) \end{bmatrix},$$

$$\vdots$$

$$\bar{\Phi}_{:, :, L, 2} = \begin{bmatrix} \bar{g}_L(\mathbf{a}_{L1}\mathbf{x}_1 + b_{L1}) & \bar{g}_L(\mathbf{a}_{L2}\mathbf{x}_1 + b_{L2}) \\ \vdots & \vdots \\ \bar{g}_L(\mathbf{a}_{L1}\mathbf{x}_N + b_{L1}) & \bar{g}_L(\mathbf{a}_{L2}\mathbf{x}_N + b_{L2}) \end{bmatrix},$$

其中  $\mathbf{a}_{ij}$  ( $i = 1, 2, \dots, L; j = 1, 2$ ) 是构建张量所用的随机生成的权重向量.


 **注 1.5.** 不确定加权方法 [RunklerCoupland2018-6356] 被用于区间二型模糊激活函数的解模糊化计算, 它代表了隶属函数隶属度对于整体解模糊化结果的影响程度, 映射结果由下式计算得到

$$u_{UW}(x) = \frac{1}{2}(\underline{u}(x) + \bar{u}(x)) \cdot (1 + \underline{u}(x) - \bar{u}(x))^\gamma, \quad (1.41)$$

其中  $\gamma > 0$  用于调整由  $(1 + \underline{u}(x) - \bar{u}(x))^\gamma$  给出解模糊化结果. 不确定加权方法推广了简单的平均下隶属和上隶属度方法, 清晰输出结果由下式给出

$$(1 + \underline{u}(x) - \bar{u}(x))^\gamma = \begin{cases} 0, & \underline{u}(x) = 0, \bar{u}(x) = 1, \\ 1, & \underline{u}(x) = \bar{u}(x). \end{cases} \quad (1.42)$$

方程 (1.42) 显示出: 当  $\gamma = 1$  时, 解模糊结构呈线性, 对于所有的  $\gamma < 1$  线性增加, 且对于所有  $\gamma > 1$ , 线性性将会减少.

 **注 1.6.** 当 (1.42) 被用于解模糊化, 只用区间二型模糊集拓展随机向量函数链接网络, 也即, 只是扩展了原网络的增强节点为区间二型模糊集, 这种类型的网络记为 *IT2-RVFL*.

最后得到的 4 阶张量  $\Phi \in \mathbb{R}^{N \times 2 \times L \times 2}$  是由前面提到的张量  $\underline{\Phi}_{:, :, :, 1}$  和  $\bar{\Phi}_{:, :, :, 2}$  构建出来的. 接下来, 张量  $\Phi$  用  $\mathcal{A}$  来记, 这是由于经典的张量方程中经常使用  $\mathcal{A}$  表示张量方程的系数.


基于张量的乘积运算, 得到的增强节点的权重矩阵维数是  $L \times 2$ , 将 *TT2-RVFL* 的增强节点部分的权重矩阵记为


$$\mathcal{X} = \begin{bmatrix} \mathbf{w}_{L+11} & \mathbf{w}_{L+12} \\ \mathbf{w}_{L+21} & \mathbf{w}_{L+22} \\ \vdots & \vdots \\ \mathbf{w}_{2L1} & \mathbf{w}_{2L2} \end{bmatrix}. \quad (1.43)$$

*TT2-RVFL* 的输出可用下式表示

$$\mathbf{Y} = \alpha \mathcal{A} *_N \mathcal{X} + (1 - \alpha) \mathbf{H} \mathbf{\Omega}, 0 \leq \alpha \leq 1, \quad (1.44)$$

其中  $\alpha$  是 *TT2-RVFL* 的平衡因子,  $\mathcal{A}$  是区间二型模糊激活函数的映射结果,  $\mathcal{X}$  是权重矩阵,  $\mathbf{Y} = [\mathbf{y} \mathbf{y}]$ , 从输入到增强节点的权重向量  $\mathbf{a}_i$  ( $i = 1, 2, \dots, L$ ) 是随机生成的, 使得  $\underline{\Phi}, \bar{\Phi}$  不饱和;  $\mathbf{\Omega} = [\boldsymbol{\omega} \boldsymbol{\omega}]$  是待求的输入权重矩阵,  $\mathbf{H}$  是由输入样本构建的输入矩阵.


 **注 1.7.** 当下三角隶属函数和上三角隶属函数一样时, *TT2-RVFL* 退化为 *RVFL*, 也即, 激活函数是一型模糊集. 当不适用直接连接方式, *TT2-RVFL* 将会退化为张量型极限学习机 [HuangZhao2018NCAA-5838]. 当同时使用 *RVFL* 和张量结构时, *TT2-RVFL* 是这两种结构的混合结果.

 **注 1.8.** 和 *RVFL* 相比, 基于张量的二型 *RVFL* 模型 (1.44) 做了三方面的扩展: 1) 增强节点使用区间二型模糊激活函数来得到非线性映射结果, 增强节点部分由张量和张量乘法表示, 且使用了张量方程的求解算法. 2) 为了和增强节点的运算相容, *TT2-RVFL* 的线性部分需要复制权重向量一次, 权重矩阵  $\mathbf{\Omega}$  等于  $[\boldsymbol{\omega} \boldsymbol{\omega}]$ , 其中  $\boldsymbol{\omega}$  为权重列向量. 3) 对于输出  $\mathbf{Y}$ , 和以前的输出向量  $\mathbf{y}$  不同, *TT2-RVFL* 需要重复该向量一次, 以便和张量运算相容.

#### 1.4.4.3 *TT2-RVFL* 中的张量逆

张量方程  $\mathcal{A} *_N \mathcal{X} = \mathcal{Y}$  的张量运算被用于求解 *TT2-RVFL* 网络的权重向量, 该运算非常适合计算 *TT2-RVFL* 增强节点部分的权重向量. 求解方程使用的偶数阶张量的

Moore-Penrose (M-P) 逆在算法 ?? 中给出, 在得到 M-P 逆的基础上, 可以很容易算出权重矩阵.

 **注 1.9.** 方程  $\mathcal{A} *_N \mathcal{X} = \mathcal{Y}$  的解也是如下张量方程的解

$$\mathcal{A}^T *_N \mathcal{A} *_N \mathcal{X} = \mathcal{A}^T *_N \mathcal{Y}, \quad (1.45)$$

其中张量  $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_N}$ ,  $\mathcal{X} \in \mathbb{R}^{J_1 \times \dots \times J_N \times K_1 \times \dots \times K_M}$  且  $\mathcal{Y} \in \mathbb{R}^{I_1 \times \dots \times I_N \times K_1 \times \dots \times K_M}$ , 不论方程是否相容, 都可以应用算法 ?? 来求解方程  $\mathcal{A} *_N \mathcal{X} = \mathcal{Y}$  [HuangZhao2018NCAA-5838].

#### 1.4.4.4 求解 TT2-RVFL 的张量逆算法

Tensor inverse algorithms for solving the TT2-RVFL 为了得到 TT2-RVFL 的解, 模型 (1.44) 的最小范目标函数定义为

$$\min_{\Omega, \mathcal{X}} \mathcal{E}_1(\mathcal{X}) + \mathcal{E}_2(\Omega), \quad (1.46)$$

其中  $\mathcal{E}_1(\mathcal{X}) = \|\mathcal{Y} - \mathcal{A}\mathcal{X}\|_F^2 + \|\mathcal{X}\|_F^2$ ,  $\mathcal{E}_2(\Omega) = \|\mathcal{y} - \mathcal{H}\Omega\|_2^2 + \|\Omega\|_2^2$ . 模型 (1.44) 的极小化问题可以划分成两个子问题来求解, 由下面的定理给出:

#### 定理 1.10 极小化模型的解


对于极小化模型 (1.44), 令  $\Omega^*$  是  $\mathcal{y} = \mathcal{H}\Omega$  的解集,  $\mathcal{X}^*$  是  $\mathcal{Y} = \mathcal{A} *_N \mathcal{X}$  的解. 对于模型 (1.46) 的两个子问题  $\mathcal{E}_1(\mathcal{X})$  和  $\mathcal{E}_2(\Omega)$ ,  $\mathcal{Y}$  退化为一个矩阵, 即  $\mathcal{Y} = [\mathcal{y} \mathcal{y}]$ , 且  $\mathcal{y}$  是训练结果列向量. 式 (1.44) 的充分条件就是寻找如下子问题的解

$$\min_{\mathcal{X}} \mathcal{E}_1(\mathcal{X}) \text{ 和 } \min_{\Omega} \mathcal{E}_2(\Omega), \quad (1.47)$$


则方程的解  $\begin{bmatrix} \mathcal{X} & \Omega \end{bmatrix}^T$  满足下式

$$\mathcal{Y} = \mathcal{A} *_N \mathcal{X} + \mathcal{H}\Omega = \begin{bmatrix} \mathcal{A} & \mathcal{H} \end{bmatrix} * \begin{bmatrix} \mathcal{X} \\ \Omega \end{bmatrix}, \quad (1.48)$$

$$\begin{bmatrix} \mathcal{X} \\ \Omega \end{bmatrix} = \begin{bmatrix} \alpha \mathcal{X}^* \\ (1 - \alpha) \Omega^* \end{bmatrix}, \quad (1.49)$$

其中  $\alpha$  是 TT2-RVFL ( $0 \leq \alpha \leq 1$ ) 的平衡因子,  $*_N$  是张量  $\mathcal{A}$  和  $\mathcal{X}$  的 Einstein 乘积. 

**证明.**  $\mathcal{H}\Omega = \mathcal{Y}$  的解是  $\Omega^*$ ,  $\mathcal{H}\Omega^* = \mathcal{Y}$  成立.  $\mathcal{X}^*$  是  $\mathcal{A} *_N \mathcal{X} = \mathcal{Y}$  的解, 则  $\mathcal{A} *_N \mathcal{X}^* = \mathcal{Y}$ . 给定平衡因子  $\alpha$ ,  $\mathcal{A} *_N \alpha \mathcal{X}^* + \mathcal{H}(1 - \alpha) \Omega^* = \alpha \mathcal{Y} + (1 - \alpha) \mathcal{Y} = \mathcal{Y}$  成立. 对于  $\mathcal{Y} \equiv \mathcal{Y}$  的情况, 张量  $\mathcal{Y}$  退化为矩阵  $\mathbf{Y}$ , 且  $\mathbf{Y} \in \mathbb{R}^{L \times 2}$ .  $\square$

 **注 1.11.** 对于 *TT2-RVFL*, 权重向量是 *RVFL* 的输出结果和张量方程解结果的综合, *RVFL* 的权重向量记为  $\omega = (H^T H + \frac{I}{C})^{-1} H^T Y$ , 其中  $I$  是单位矩阵, 且  $C$  是平衡参数.

#### 1.4.4.5 仿真实验

提出的算法使用了 Frobenius 范数来度量训练误差和测试误差, 所有结果都由 1000 次运行结果求得. 为了说明 IT2-RVFL 和 TT2-RVFL 的学习能力, 隐藏节点数选为  $L = [30, 35, 40]$ . IT2-RVFL 和 TT2-RVFL 使用了均值一样, 方差不确定这一形式, 下隶属函数从区间  $[0.5, 0.9]$  随机选取, 上隶属函数通过给下隶属函数值加 0.1 得到; 则  $k_1, k_2$  可由关系式  $k_1 = \frac{1}{\sigma_i^2}, k_2 = \frac{1}{\sigma_i^2} (i = 1, 2, \dots, L)$  求得.

#### 1.4.4.6 非线性函数

**例 4.1:** 单输入 Sinc 函数的逼近问题如下

$$y = \begin{cases} 1, & x = 0 \\ \frac{\sin x}{x}, & x \neq 0 \end{cases} \quad x \in [-10, 10]. \quad (1.50)$$

对于 IT2RVFL 和 TT2-RVFL, 平衡参数设置为  $C = 2^{10}$ ,  $\gamma = 0.95$ . 表 1-3 给出了例 4.1 在不同  $L$  下的训练误差和测试误差. 对于 IT2-RVFL, 当  $L = 30, 35$ , 它的训练误差要小于 TT2-RVFL 的训练误差. 然而, TT2-RVFL 的测试误差要小于 IT2-RVFL 的测试误差, 这意味着 TT2-RVFL 中的平衡因子对网络的学习能力有一定的影响.

表 1-3 例 4.1 中不同  $L$  下的训练和测试误差

$L$	算法	训练误差		预测误差	
		均值	方差	均值	方差
30	IT2-RVFL	2.53e-02	4.51e-05	4.82e-02	7.22e-04
	TT2-RVFL	<b>2.36e-04</b>	<b>1.71e-05</b>	<b>2.32e-04</b>	<b>1.70e-05</b>
35	IT2-RVFL	2.34e-02	9.15e-05	4.66e-02	3.17e-04
	TT2-RVFL	<b>1.57e-02</b>	<b>7.44e-05</b>	<b>1.54e-02</b>	<b>7.61e-05</b>
40	IT2-RVFL	<b>2.18e-02</b>	<b>1.38e-04</b>	4.57e-02	7.92e-04
	TT2-RVFL	3.86e-02	4.25e-04	<b>3.76e-02</b>	<b>4.25e-04</b>

**例 4.2:** 含噪的非线性函数具有如下形式

$$y = \frac{0.9x}{1.2x^3 + x + 0.3} + \eta, x \in [0, 1], \quad (1.51)$$

其中  $\eta$  是均匀分布的白噪声,  $\eta$  的幅值设置为 0.2.

例 4.2 用来测试给出算法的野值拒绝能力. 对于 IT2RVFL 和 TT2-RVFL, 平衡参数设置为  $C = 2^{10}$ ,  $\gamma = 1$ . 表 1-4 给出了例 4.2 在不同  $L$  值下的训练误差和测试误差. 结果表明, 当  $L = 30, 35, 40$  时, TT2-RVFL 性能要好于 IT2-RVFL, 其平均训练误差和测试误差要小于 IT2-RVFL 的结果.

表 1-4 例 4.2 中不同  $L$  下的训练误差和预测误差

$L$	算法	训练误差		预测误差	
		均值	方差	均值	方差
30	IT2-RVFL	2.85e-01	<b>7.30e-06</b>	4.13e-01	5.58e-03
	TT2-RVFL	<b>2.07e-01</b>	8.43e-04	<b>3.07e-01</b>	<b>2.19e-03</b>
35	IT2-RVFL	2.85e-01	<b>2.94e-06</b>	4.15e-01	2.45e-03
	TT2-RVFL	<b>2.07e-01</b>	1.75e-03	<b>3.06e-01</b>	<b>1.96e-03</b>
40	IT2-RVFL	2.85e-01	<b>5.14e-06</b>	4.11e-01	5.10e-03
	TT2-RVFL	<b>2.06e-01</b>	4.54e-04	<b>3.04e-01</b>	<b>1.13e-03</b>

例 4.3: 双输入单输出 Sinc 函数的逼近问题.

$$z = \left| \frac{\sin x \cdot \sin y}{xy} \right|, (x, y) \in [-\pi, \pi]^2. \quad (1.52)$$

采用等距采样方式,  $41 \times 41$  训练数据对用于训练,  $30 \times 30$  等距采样结果用来检验模型的泛化能力. 对于 IT2-RVFL 和 TT2-RVFL, 平衡因子设置为  $C = 2^{10}$ ,  $\gamma = 1$ . 表 1-5 给出了例 4.3 在不同  $L$  下的训练误差和测试误差. TT2-RVFL 的结果要好于 IT2-RVFL 的结果, 训练误差和测试误差都要小于 IT2-RVFL 的结果.

#### 1.4.4.7 非线性系统参数识别

对于如下的非线性系统 [Rong2009-4804697]

$$y_p(k) = \frac{y_p(k-1)y_p(k-2)(y_p(k-1) + 2.5)}{1 + (y_p(k-1))^2 + (y_p(k-2))^2} + u(k-1). \quad (1.53)$$

系统的平衡态为 (0,0), 输入  $u(k) \in \{-2, 2\}$ , 操作区间规定为  $[-2, 2]$ . 均匀分布的随机变量从  $\{-2, 2\}$  中产生, 测试使用的输入  $u(k) = \sin(2\pi k/25)$ . 通过选取  $[y_p(k-1), y_p(k-2), u(k-1)]$  作为输入变量,  $y_p(k)$  作为输出变量. 系统可以写为

$$\hat{y}_p(k) = \hat{f}(y_p(k-1), y_p(k-2), u(k-1)). \quad (1.54)$$

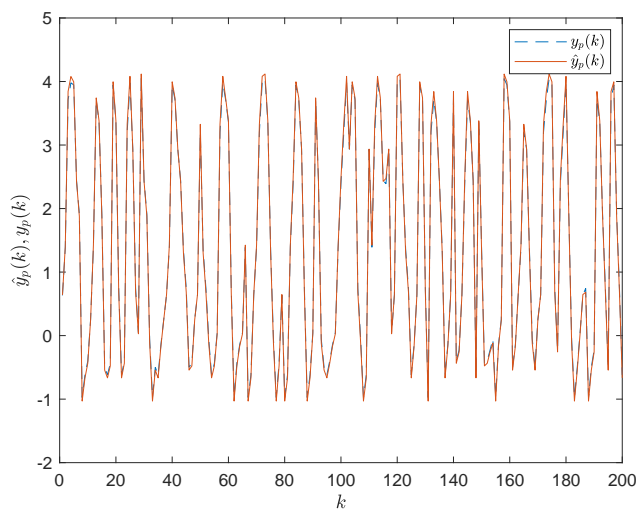


图 1-21 TT2-RVFL 对非线性系统识别问题的预测结果

表 1-5 例 4.3 中不同  $L$  下的训练误差和预测误差

$L$	算法	训练误差		预测误差	
		均值	方差	均值	方差
30	IT2-RVFL	4.65e-02	<b>3.09e-06</b>	7.06e-02	<b>1.43e-09</b>
	TT2-RVFL	<b>4.36e-02</b>	1.30e-05	<b>6.77e-02</b>	6.05e-06
35	IT2-RVFL	4.63e-02	<b>8.69e-06</b>	7.06e-02	<b>2.40e-08</b>
	TT2-RVFL	<b>4.07e-02</b>	3.38e-05	<b>6.73e-02</b>	9.21e-06
40	IT2-RVFL	<b>4.63e-02</b>	<b>2.95e-06</b>	7.06e-02	<b>1.56e-09</b>
	TT2-RVFL	<b>3.98e-02</b>	4.72e-05	<b>6.75e-02</b>	1.09e-05

随机选用了 800 组数据, 600 组用于训练, 200 组用于测试. 对于 IT2RVFL 和 TT2-RVFL, 平衡参数设置为  $C = 2^{10}$ . 对于仿真, TT2-RVFL 的预测结果见图 1-21. 训练误差均值的 Frobenius 范数是  $1.87\text{e-}01$ , 测试误差均值的 Frobenius 范数是  $3.94\text{e-}01$ , 结果显示, TT2-RVFL 能够得到满意的精度.

#### 1.4.4.8 回归问题

Regression problems

表 1-6 数据集 Auto-Mpg, Bank, Diabetes 和 Triazines 上的训练和测试误差

数据集	算法	训练误差		预测误差	
		均值	方差	均值	方差
Auto-Mpg	ELM	4.3511e-01	2.4125e+0	4.3423e-01	2.5502e+0
	RELM	1.3514e+0	1.6125e+0	1.3132e+0	1.6101e+0
	RVFL	9.9152e-01	6.3012e-03	9.2356e-01	5.1511e-03
	IT2-RVFL	<b>2.2161e-02</b>	<b>1.5915e-04</b>	4.5358e-02	<b>1.6692e-04</b>
	TT2-RVFL	4.4055e-02	8.1940e-04	<b>4.3847e-02</b>	8.1024e-04
Bank	ELM	3.2400e-02	5.8710e-02	3.2103e-02	5.8425e-02
	RELM	3.3214e-02	5.8321e-02	<b>3.0936e-02</b>	5.8631e-02
	RVFL	1.1784e-01	6.3033e-03	5.0163e-02	1.3696e-04
	IT2-RVFL	<b>2.9035e-02</b>	<b>3.4437e-05</b>	7.0701e-02	9.9726e-05
	TT2-RVFL	4.3582e-02	1.1435e-04	5.1382e-02	<b>8.3401e-05</b>
Diabetes	ELM	1.5199e-01	1.5535e-01	1.5724e-01	1.3706e-01
	RELM	1.5046e-01	1.3448e-01	1.5741e-01	1.4002e-01
	RVFL	3.3931e-01	6.2734e-02	<b>8.6827e-02</b>	1.0573e-02
	IT2-RVFL	1.4838e-01	2.6181e-05	1.5938e-01	4.3915e-04
	TT2-RVFL	<b>1.4548e-01</b>	<b>2.1532e-06</b>	1.4522e-01	<b>8.8627e-05</b>
Triazines	ELM	9.8322e-02	2.4095e-01	1.0947e-01	1.6613e-01
	RELM	9.9402e-02	2.1026e-01	1.0739e-01	1.6718e-01
	RVFL	3.0201e-01	2.8114e-01	<b>2.2935e-02</b>	1.3741e-02
	IT2-RVFL	<b>8.2873e-02</b>	<b>6.5574e-05</b>	4.0105e-01	3.9270e-03
	TT2-RVFL	1.0360e-01	3.1127e-04	2.6182e-01	<b>1.0300e-03</b>

测试使用了四个回归数据集 (Auto-Mpg, Bank, Diabetes 和 Triazines). Auto-MPG 数据集有 392 个样本, 每一个样本有 8 个属性. Bank 数据集有 8192 个样本, 每一个样本有 8 个属性. Diabetes 数据集有 768 个样本, 每一个样本有 4 个属性. Triazines 数据集有 186 个样本, 每一个样本有 5 个属性. 对于 Auto-MPG 和 Bank, 使用了 mapminmax 方法来计算映射到区间  $[-1, 1]$  的数据. 而对于 Diabetes 和 Triazines, 则直接使用原来的数据样本. 对于 IT2-RVFL 和 TT2-RVFL, 平衡参数设置为  $C = 2^{10}$ ,  $\gamma = 1$ . RVFL 的代码可以从 Matlab 的 File Exchange 下载, ELM 和 RELM 的代码可以从[主页下载](#). 表 1-6 给出了 ELM、RELM、RVFL、IT2-RVFL 和 TT2-RVFL 算法的训练误差和测试误差. 对于



Auto-MPG, 扩展后的 RVFL 的训练误差和测试误差都得到了很好的结果. IT2-RVFL 在 Bank 上的训练误差最小, RELM 在 Bank 上的测试误差最小, 这说明正则化项在 ELM 的训练过程中是发挥作用的. TT2-RVFL 在 Diabetes 上的训练误差最小, 它的测试误差要大于 RVFL, 均值的误差大约是 0.0584. IT2-RVFL 在 Triazines 的训练误差最小, 而 RVFL 的测试误差则总体较小. 对于四个数据集的回归结果, 可以得出 IT2-RVFL 和 TT2-RVFL 是可以作为回归问题的求解器使用的. 如表 1-4 的结果所示, TT2-RVFL 更适合于数据含噪的情形.

## 1.5 进化计算

进化计算 (Evolutionary Computation, EC) 是在达尔文 (Darwin) 的进化论和孟德尔 (Mendel) 的遗传变异理论的基础上产生的一种在基因和种群层次上模拟自然界生物进化过程与机制的问题求解技术. 它主要包括

- ▶ 遗传算法 (Genetic Algorithm, GA)
- ▶ 进化策略 (Evolutionary Strategy, ES)
- ▶ 进化规划 (Evolutionary Programming, EP)
- ▶ 遗传规划 (Genetic Programming, GP) 四大分支.

其中, 第一个分支是进化计算中最初形成的一种具有普遍影响的模拟进化优化算法. 因此我们主要讨论遗传算法.

进化计算是一种模拟自然界生物进化过程与机制进行问题求解的自组织、自适应的随机搜索技术. 它以达尔文进化论的“物竞天择、适者生存”作为算法的进化规则, 并结合孟德尔的遗传变异理论, 将生物进化过程中的

- ▶ 繁殖 (Reproduction)
- ▶ 变异 (Mutation)
- ▶ 竞争 (Competition)
- ▶ 选择 (Selection)

引入到了算法中.

### (2) 进化计算的生物学基础

自然界生物进化过程是进化计算的生物学基础, 它主要包括遗传 (Heredity)、变异 (Mutation) 和进化 (Evolution) 理论.

#### ① 遗传理论

遗传是指父代 (或亲代) 利用遗传基因将自身的基因信息传递给下一代 (或子代), 使子代能够继承其父代的特征或性状的这种生命现象. 正是由于遗传的作用, 自然界才能



有稳定的物种。

在自然界,构成生物基本结构与功能的单位是细胞 (Cell)。细胞中含有一种包含着所有遗传信息的复杂而又微小的丝状化合物,人们称其为染色体 (Chromosome)。在染色体中,遗传信息由基因 (Gene) 所组成,基因决定着生物的性状,是遗传的基本单位。染色体的形状是一种双螺旋结构,构成染色体的主要物质叫做脱氧核糖核酸 (DNA),每个基因都在 DNA 长链中占有一定的位置。一个细胞中的所有染色体所携带的遗传信息的全体称为一个基因组 (Genome)。细胞在分裂过程中,其遗传物质 DNA 通过复制转移到新生细胞中,从而实现了生物的遗传功能。

### ② 变异理论

变异是指子代和父代之间,以及子代的各个不同个体之间产生差异的现象。变异是生物进化过程中发生的一种随机现象,是一种不可逆过程,在生物多样性方面具有不可替代的作用。引起变异的主要原因有以下两种:杂交,是指有性生殖生物在繁殖下一代时两个同源染色体之间的交配重组,即两个染色体在某一相同处的 DNA 被切断后再进行交配重组,形成两个新的染色体。复制差错,是指在细胞复制过程中因 DNA 上某些基因结构的随机改变而产生出新的染色体。

### ③ 进化论

进化是指在生物延续生存过程中,逐渐适应其生存环境,使得其品质不断得到改良的这种生命现象。遗传和变异是生物进化的两种基本现象,优胜劣汰、适者生存是生物进化的基本规律。

达尔文的自然选择学说认为:在生物进化中,一种基因有可能发生变异而产生出另一种新的生物基因。这种新基因将依据其与生存环境的适应性而决定其增殖能力。一般情况下,适应性强的基因会不断增多,而适应性差的基因则会逐渐减少。通过这种自然选择,物种将逐渐向适应于生存环境的方向进化,甚至会演变成为另一个新的物种,而那些不适应于环境的物种将会逐渐被淘汰。

## 1.5.1 进化计算的产生与发展

进化计算自 20 世纪 50 年代以来,其发展过程大致可分为三个阶段。

### ① 萌芽阶段

这一阶段是从 20 世纪 50 年代后期到 70 年代中期。20 世纪 50 年代后期,一些生物学家在研究如何用计算机模拟生物遗传系统中,产生了遗传算法的基本思想,并于 1962 年由美国密执安 (Michigan) 大学霍兰德 (Holland) 提出。1965 年德国数学家雷切伯格 (Rechenberg) 等人提出了一种只有单个个体参与进化,并且仅有变异这一种进化操作的进化策略。同年,美国学者弗格尔 (Fogel) 提出了一种具有多个个体和仅有变异一种进化

操作的进化规划. 1969 年美国密执安 (Michigan) 大学的霍兰德 (Holland) 提出了系统本身和外部环境相互协调的遗传算法. 至此, 进化计算的三大分支基本形成.

### ② 成长阶段

这一阶段是从 20 世纪 70 年代中期到 80 年代后期. 1975 年, 霍兰德出版专著《自然和人工系统的适应性 (Adaptation in Natural and Artificial System)》, 全面介绍了遗传算法. 同年, 德国学者施韦费尔 (Schwefel) 在其博士论文中提出了一种由多个个体组成的群体参与进化的, 并且包括了变异和重组这两种进化操作的进化策略. 1989 年, 霍兰德的学生戈尔德伯格 (Goldberg) 博士出版专著《遗传算法——搜索、优化及机器学习 (Genetic Algorithm—in Search Optimization and Machine Learning)》, 使遗传算法得到了普及与推广.

### ③ 发展阶段

这一阶段是从 20 世纪 90 年代至今. 1989 年, 美国斯坦福 (Stanford) 大学的科扎 (Koza) 提出了遗传规划的新概念, 并于 1992 年出版了专著《遗传规划——应用自然选择法则的计算机程序设计 (Genetic Programming: on the Programming of Computer by Means of Natural Selection)》. 该书全面介绍了遗传规划的基本原理及应用实例, 标志着遗传规划作为计算智能的一个分支已基本形成.

进入 20 世纪 90 年代以来, 进化计算得到了众多研究机构 and 学者的高度重视, 新的研究成果不断出现、应用领域不断扩大. 目前, 进化计算已成为人工智能领域的又一个新的研究热点.

## 1.5.2 进化计算的基本结构

进化计算尽管有多个重要分支, 并且不同分支的编码方案、选择策略和进化操作也有可能不同, 但它们却有着共同的进化框架. 若假设  $P$  为种群 (Population, 或称为群体),  $t$  为进化代数,  $P(t)$  为第  $t$  代种群, 则进化计算的基本结构可粗略描述如下:

{ 确定编码形式并生成搜索空间;

    初始化各个进化参数, 并设置进化代数  $t=0$ ;

    初始化种群  $P(0)$ ;

    对初始种群进行评价 (即适应度计算);

    while(不满足终止条件)do

        {

$t=t+1$ ;

            利用选择操作从  $P(t-1)$  代中选出  $P(t)$  代群体;

            对  $P(t)$  代种群执行进化操作;

```

        对执行完进化操作后的种群进行评价 (即适应度计算);
    }
}

```

可以看出,上述基本结构包含了生物进化中所必需的选择操作、进化操作和适应度评价等过程。

遗传算法的基本思想是从初始种群出发,采用优胜劣汰、适者生存的自然法则选择个体,并通过杂交、变异来产生新一代种群,如此逐代进化,直到满足目标为止。遗传算法所涉及到的基本概念主要有以下几个:

- ▶ 种群 (Population): 种群是指用遗传算法求解问题时,初始给定的多个解的集合。遗传算法的求解过程是从这个子集开始的。
- ▶ 个体 (Individual): 个体是指种群中的单个元素,它通常由一个用于描述其基本遗传结构的数据结构来表示。例如,可以用 0、1 组成的长度为 1 的串来表示个体。
- ▶ 染色体 (Chromos): 染色体是指对个体进行编码后所得到的编码串。染色体中的每 1 位称为基因,染色体上由若干个基因构成的一个有效信息段称为基因组。
- ▶ 适应度 (Fitness) 函数: 适应度函数是一种用来对种群中各个个体的环境适应性进行度量的函数。其函数值是遗传算法实现优胜劣汰的主要依据。
- ▶ 遗传操作 (Genetic Operator): 遗传操作是指作用于种群而产生新的种群的操作。标准的遗传操作包括以下 3 种基本形式:
  - ▶ 选择 (Selection)
  - ▶ 交叉 (Crossover)
  - ▶ 变异 (Mutation)

遗传算法主要由染色体编码、初始种群设定、适应度函数设定、遗传操作设计等几大部分所组成,其算法主要内容和基本步骤可描述如下:

(1) 选择编码策略,将问题搜索空间中每个可能的点用相应的编码策略表示出来,即形成染色体;

(2) 定义遗传策略,包括种群规模  $N$ ,交叉、变异方法,以及选择概率  $P_r$ 、交叉概率  $P_c$ 、变异概率  $P_m$  等遗传参数;

(3) 令  $t = 0$ ,随机选择  $N$  个染色体初始化种群  $P(0)$ ;

(4) 定义适应度函数  $f(f > 0)$ ;

(5) 计算  $P(t)$  中每个染色体的适应值;

(6)  $t = t + 1$ ;

(7) 运用选择算子,从  $P(t - 1)$  中得到  $P(t)$ ;

(8) 对  $P(t)$  中的每个染色体,按概率  $P_c$  参与交叉;

- (9) 对染色体中的基因, 以概率  $P_m$  参与变异运算;  
 (10) 判断群体性能是否满足预先设定的终止标准, 若不满足则返回 (5).  
 其算法流程如图1-22所示.

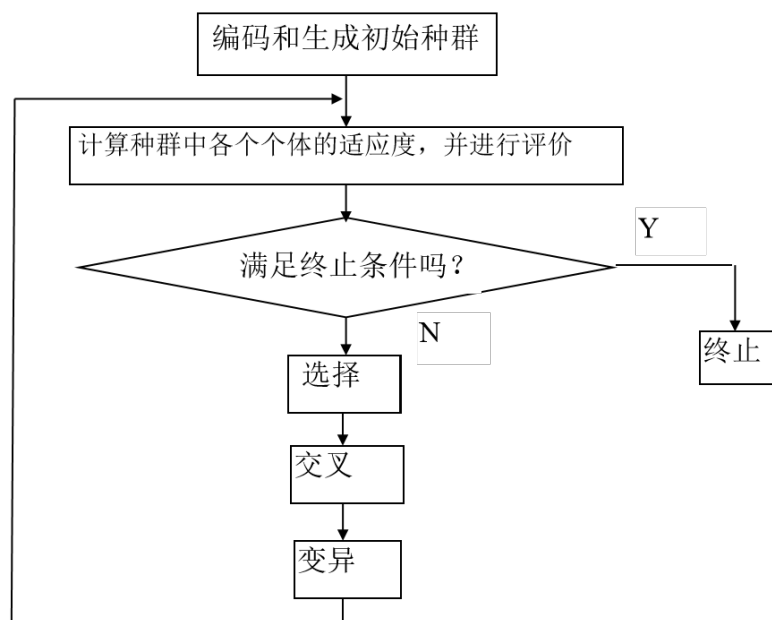


图5-18 基本遗传算法的算法流程图

图 1-22

## 1.6 遗传编码

常用的遗传编码算法有霍兰德二进制码、格雷码 (Gray Code)、实数编码和字符编码等.

### (1) 二进制编码 (Binary encoding)

二进制编码是将原问题的结构变换为染色体的位串结构. 在二进制编码中, 首先要确定二进制字符串的长度  $l$ , 该长度与变量的定义域和所求问题的计算精度有关.

**例 1.12** 假设变量  $x \in [5, 10]$ , 要求的计算精度为  $10^{-5}$ , 则需要将  $[5, 10]$  至少分为 600000 个等长小区间, 每个小区间用一个二进制串表示. 于是, 串长至少等于 20, 原因是:

$$524288 = 2^{19} < 600000 < 2^{20} = 1048576$$

这样, 对应于区间  $[5, 10]$  内满足精度要求的每个值  $x$ , 都可用一个 20 位编码的二进制串  $\langle b_{19}, b_{18}, \dots, b_0 \rangle$  来表示.

二进制编码存在的主要缺点是汉明 (Hamming) 悬崖.

**例 1.13** 7 和 8 的二进制数分别为 0111 和 1000, 当算法从 7 改进到 8 时, 就必须改变所有的位.

## (2) 格雷编码 (Gray encoding)

格雷编码是对二进制编码进行变换后所得到的一种编码方法. 这种编码方法要求两个连续整数的编码之间只能有一个码位不同, 其余码位都是完全相同的. 它有效地解决了汉明悬崖问题, 其基本原理如下:

设有二进制串  $b_1, b_2, \dots, b_n$ , 对应的格雷串为  $a_1, a_2, \dots, a_n$ , 则从二进制编码到格雷编码的变换为:

$$a_i = \begin{cases} b_1, & i = 1 \\ b_{i-1} \oplus b_i & i > 1 \end{cases} \quad (1.55)$$

其中,  $\oplus$  表示模 2 加法. 而从一个格雷串到二进制串的变换为:

$$b_i = \sum_{j=1}^i a_j (\text{mod } 2) \quad (1.56)$$

**例 1.14** 十进制数 7 和 8 的二进制编码分别为 0111 和 1000, 而其格雷编码分别为 0100 和 1100.

## (3) 实数编码 (Real encoding)

实数编码是将每个个体的染色体都用某一范围的一个实数 (浮点数) 来表示, 其编码长度等于该问题变量的个数.

这种编码方法是将问题的解空间映射到实数空间上, 然后在实数空间上进行遗传操作. 由于实数编码使用的是变量的真实值, 因此这种编码方法也叫做真值编码方法.

实数编码适应于那种多维、高精度要求的连续函数优化问题.

适应度函数是一个用于对个体的适应性进行度量的函数. 通常, 一个个体的适应度值越大, 它被遗传到下一代种群中的概率也就越大.

### (1) 常用的适应度函数

在遗传算法中, 有许多计算适应度的方法, 其中最常用的适应度函数有以下两种:

#### ① 原始适应度函数

它是直接将待求解问题的目标函数  $f(x)$  定义为遗传算法的适应度函数. 例如, 在求解极值问题

$$\max_{x \in [a, b]} f(x) \quad (1.57)$$

时,  $f(x)$  即为  $x$  的原始适应度函数.

采用原始适应度函数的优点是能够直接反映出待求解问题的最初求解目标, 其缺点是有可能出现适应度值为负的情况.

### ② 标准适应度函数

在遗传算法中, 一般要求适应度函数非负, 并其适应度值越大越好. 这就往往需要对原始适应函数进行某种变换, 将其转换为标准的度量方式, 以满足进化操作的要求, 这样所得到的适应度函数被称为标准适应度函数  $f_{\text{Normal}}(x)$ .

**1.6.0.0.1 极小化问题** 对极小化问题, 其标准适应度函数可定义为

$$f(x) = \begin{cases} f_{\max}(x) - f(x) & f(x) < f_{\max}(x) \\ 0 & f(x) \geq f_{\max}(x) \end{cases} \quad (1.58)$$

其中,  $f_{\max}(x)$  是原始适应函数  $f(x)$  的一个上界. 如果  $f_{\max}(x)$  未知, 则可用当前代或到目前为止各演化代中的  $f(x)$  的最大值来代替. 可见,  $f_{\max}(x)$  是会随着进化代数的增加而不断变化的.

**1.6.0.0.2 极大化问题** 对极大化问题, 其标准适应度函数可定义为

$$f(x) = \begin{cases} f(x) - f_{\min}(x) & f(x) > f_{\min}(x) \\ 0 & f(x) \leq f_{\min}(x) \end{cases} \quad (1.59)$$

其中,  $f_{\min}(x)$  是原始适应函数  $f(x)$  的一个下界. 如果  $f_{\min}(x)$  未知, 则可用当前代或到目前为止各演化代中的  $f(x)$  的最小值来代替.

### (2) 适应度函数的加速变换

在某些情况下, 适应度函数在极值附近的变化可能会非常小, 以至于不同个体的适应值非常接近, 使得难以区分出哪个染色体更占优势. 对此, 最好能定义新的适应度函数, 使该适应度函数既与问题的目标函数具有相同的变化趋势, 又有更快的变化速度.

适应度函数的加速变换有两种基本方法

#### ① 线性加速

适应度函数的定义如下:

$$f'(x) = \alpha f(x) + \beta \quad (1.60)$$

其中,  $f(x)$  是加速转换前的适应度函数;  $f'(x)$  是加速转换后的适应度函数;  $\alpha$  和  $\beta$  是转换系数.

② 非线性加速

► 幂函数变换方法

$$f'(x) = f(x)k \quad (1.61)$$

► 指数变换方法

$$f'(x) = \exp^{-\beta f(x)} \quad (1.62)$$

遗传算法中的基本遗传操作包括选择、交叉和变异 3 种, 而每种操作又包括多种不同的方法, 下面分别对它们进行介绍.

► 选择操作

选择 (Selection) 操作是指根据选择概率按某种策略从当前种群中挑选出一定数目的个体, 使它们能够有更多的机会被遗传到下一代中.

常用的选择策略可分为比例选择、排序选择和竞技选择三种类型.

► 比例选择

比例选择方法 (Proportional Model) 的基本思想是: 各个个体被选中的概率与其适应度大小成正比.

常用的比例选择策略包括: (1) 轮盘赌选择. (2) 繁殖池选择

► 轮盘赌选择

轮盘赌选择法又被称为转盘赌选择法或轮盘选择法. 在这种方法中, 个体被选中的概率取决于该个体的相对适应度. 而相对适应度的定义为:

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^N f(x_j)} \quad (1.63)$$

其中,  $P(x_i)$  是个体  $x_i$  的相对适应度, 即个体  $x_i$  被选中的概率;  $f(x_i)$  是个体  $x_i$  的原始适应度; 是种群的累加适应度.

轮盘赌选择算法的基本思想是: 根据每个个体的选择概率  $P(x_i)$  将一个圆盘分成  $N$  个扇区, 其中第  $i$  个扇区的中心角为:

$$2\pi \frac{f(x_i)}{\sum_{j=1}^N f(x_j)} = 2\pi p(x_i) \quad (1.64)$$

并再设立一个固定指针. 当进行选择时, 可以假想转动圆盘, 若圆盘静止时指针指向第  $i$  个扇区, 则选择个体  $i$ . 其物理意义如图 5-19 所示.

从统计角度看, 个体的适应度值越大, 其对应的扇区的面积越大, 被选中的可能性也越大. 这种方法有点类似于发放奖品使用的轮盘, 并带有某种赌博的意思, 因此亦被称为轮盘赌选择.

## (2) 交叉操作

交叉 (Crossover) 操作是指按照某种方式对选择的父代个体的染色体的部分基因进行交配重组, 从而形成新的个体. 交配重组是自然界中生物遗传进化的一个主要环节, 也是遗传算法中产生新的个体的最主要方法. 根据个体编码方法的不同, 遗传算法中的交叉操作可分为二进制交叉和实值交叉两种类型.

### ① 二进制交叉

二进制交叉 (Binary Valued Crossover) 是指二进制编码情况下所采用的交叉操作, 它主要包括单点交叉、两点交叉、多点交叉和均匀交叉等方法.

### ② 单点交叉

单点交叉也称简单交叉, 它是先在两个父代个体的编码串中随机设定一个交叉点, 然后对这两个父代个体交叉点前面或后面部分的基因进行交换, 并生成子代中的两个新的个体. 假设两个父代的个体串分别是:

$$X = x_1 x_2 \cdots x_k x_{k+1} \cdots x_n \quad (1.65)$$

$$Y = y_1 y_2 \cdots y_k y_{k+1} \cdots y_n \quad (1.66)$$

随机选择第  $k$  位为交叉点, 若采用对交叉点后面的基因进行交换的方法, 但点交叉是将  $X$  中的  $x_{k+1}$  到  $x_n$  部分与  $Y$  中的  $y_{k+1}$  到  $y_n$  部分进行交叉, 交叉后生成的两个新的个体是:

$$X' = x_1 x_2 \cdots x_k y_{k+1} \cdots y_n \quad (1.67)$$

$$Y' = y_1 y_2 \cdots y_k x_{k+1} \cdots x_n \quad (1.68)$$

**例 1.15** 设有两个父代的个体串  $A = 001101$  和  $B = 110010$ , 若随机交叉点为 4, 则交叉后生成的两个新的个体是:

$$A' = 001110 \quad (1.69)$$

$$B' = 110001 \quad (1.70)$$



### ③ 两点交叉

两点交叉是指先在两个父代个体的编码串中随机设定两个交叉点, 然后再按这两个交叉点进行部分基因交换, 生成子代中的两个新的个体.

假设两个父代的个体串分别是:

$$X = x_1 x_2 \cdots x_i \cdots x_j \cdots x_n \quad (1.71)$$

$$Y = y_1 y_2 \cdots y_i \cdots y_j \cdots y_n \quad (1.72)$$

随机设定第  $i, j$  位为两个交叉点 (其中  $i < j < n$ ), 两点交叉是将  $X$  中的  $x_{i+1}$  到  $x_j$  部分与  $Y$  中的  $y_{i+1}$  到  $y_j$  部分进行交换, 交叉后生成的两个新的个体是:

$$X' = x_1 x_2 \cdots x_i y_{i+1} \cdots y_j x_{j+1} \cdots x_n \quad (1.73)$$

$$Y' = y_1 y_2 \cdots y_i x_{i+1} \cdots x_j y_{j+1} \cdots y_n \quad (1.74)$$

**例 1.16** 设有两个父代的个体串  $A = 001101$  和  $B = 110010$ , 若随机交叉点为 3 和 5, 则交叉后的两个新的个体是:

$$A' = 001011 \quad (1.75)$$

$$B' = 110100 \quad (1.76)$$

### ④ 多点交叉

多点交叉是指先随机生成多个交叉点, 然后再按这些交叉点分段地进行部分基因交换, 生成子代中的两个新的个体.

假设交叉点个数为  $m$ , 则可将个体串划分为  $m + 1$  个分段, 其划分方法是:

当  $m$  为偶数时, 对全部交叉点依次进行两两配对, 构成  $m/2$  个交叉段.

当  $m$  为奇数时, 对前  $(m - 1)$  个交叉点依次进行两两配对, 构成  $(m - 1)/2$  个交叉段, 而第  $m$  个交叉点则按单点交叉方法构成一个交叉段.

下面以  $m = 3$  为例进行讨论.

假设两个父代的个体串分别是

$$X = x_1 x_2 \cdots x_i \cdots x_j \cdots x_k \cdots x_n$$

和

$$Y = y_1 y_2 \cdots y_i \cdots y_j \cdots y_k \cdots y_n,$$

随机设定第  $i, j, k$  位为三个交叉点 (其中  $i < j < k < n$ ), 则将构成两个交叉段. 交叉后生成的两个新的个体是:

$$X' = x_1 x_2 \cdots x_i y_{i+1} \cdots y_j x_{j+1} \cdots x_k y_{k+1} \cdots y_n \quad (1.77)$$

$$Y' = y_1 y_2 \cdots y_i x_{i+1} \cdots x_j y_{j+1} \cdots y_k x_{k+1} \cdots x_n \quad (1.78)$$

**例 1.17** 设有两个父代的个体串  $A = 001101$  和  $B = 110010$ , 若随机交叉点为 1、3 和 5, 则交叉后的两个新的个体是:

$$A' = 010100 \quad (1.79)$$

$$B' = 101011 \quad (1.80)$$

#### ⑤ 均匀交叉

均匀交叉 (Uniform Crossover) 是先随机生成一个与父串具有相同长度, 并被称为交叉模版 (或交叉掩码) 的二进制串, 然后再利用该模版对两个父串进行交叉, 即将模版中 1 对应的位进行交换, 而 0 对应的位不交换, 依此生成子代中的两个新的个体. 事实上, 这种方法对父串中的每一位都是以相同的概率随机进行交叉的.

**例 1.18** 设有两个父代的个体串  $A = 001101$  和  $B = 110010$ , 若随机生成的模版  $T = 010011$ , 则交叉后的两个新的个体是  $A' = 011010$  和  $B' = 100101$ . 即

A: 0 0 1 1 0 1

B: 1 1 0 0 1 0

T: 0 1 0 0 1 1

A': 0 1 1 1 1 0

B': 1 0 0 0 0 1

#### ⑥ 实值交叉

实值交叉是在实数编码情况下所采用的交叉操作, 主要包括离散交叉和算术交叉, 下面主要讨论离散交叉 (部分离散交叉和整体离散交叉).

部分离散交叉是先在两个父代个体的编码向量中随机选择一部分分量, 然后对这部分分量进行交换, 生成子代中的两个新的个体.

整体交叉则是对两个父代个体的编码向量中的所有分量, 都以  $1/2$  的概率进行交换, 从而生成子代中的两个新的个体.

以部分离散交叉为例,假设两个父代个体的 $n$ 维实向量分别是 $X = x_1 x_2 \cdots x_i \cdots x_k \cdots x_n$ 和 $Y = y_1 y_2 \cdots y_i \cdots y_k \cdots y_n$ ,若随机选择对第 $k$ 个分量以后的所有分量进行交换,则生成的两个新的个体向量是:

$$X' = x_1 x_2 \cdots x_k y_{k+1} \cdots y_n \quad (1.81)$$

$$Y' = y_1 y_2 \cdots y_k x_{k+1} \cdots x_n \quad (1.82)$$

**例 1.19** 设有两个父代个体向量  $A = 20 \ 16 \ 19 \ 32 \ 18 \ 26$  和  $B = 36 \ 25 \ 38 \ 12 \ 21 \ 30$ , 若随机选择对第 3 个分量以后的所有分量进行交叉, 则交叉后两个新的个体向量是:

$$A' = 20 \ 16 \ 19 \ 12 \ 21 \ 30 \quad (1.83)$$

$$B' = 36 \ 25 \ 38 \ 32 \ 18 \ 26 \quad (1.84)$$

### (3) 变异操作

变异 (Mutation) 是指对选中个体的染色体中的某些基因进行变动, 以形成新的个体. 变异也是生物遗传和自然进化中的一种基本现象, 它可增强种群的多样性. 遗传算法中的变异操作增加了算法的局部随机搜索能力, 从而可以维持种群的多样性. 根据个体编码方式的不同, 变异操作可分为二进制变异和实值变异两种类型.

#### ① 二进制变异

当个体的染色体采用二进制编码表示时, 其变异操作应采用二进制变异方法. 该变异方法是先随机地产生一个变异位, 然后将该变异位置上的基因值由“0”变为“1”, 或由“1”变为“0”, 产生一个新的个体.

**例 1.20** 设变异前的个体为  $A = 001101$ , 若随机产生的变异位置是 2, 则该个体的第 2 位由“0”变为“1”. 变异后的新的个体是  $A' = 011101$ .

#### ② 实值变异

当个体的染色体采用实数编码表示时, 其变异操作应采用实值变异方法. 该方法是用另外一个在规定范围内的随机实数去替换原变异位置上的基因值, 产生一个新的个体. 最常用的实值变异操作有:

##### 基于位置的变异方法

该方法是先随机地产生两个变异位置, 然后将第二个变异位置上的基因移动到第一个变异位置的前面.

**例 1.21** 设选中的个体向量  $C = 20\ 16\ 19\ 12\ 21\ 30$ , 若随机产生的两个变异位置分别是 2 和 4, 则变异后的新的个体向量是:

$$C' = 20\ 12\ 16\ 19\ 21\ 30 \quad (1.85)$$

基于次序的变异

该方法是先随机地产生两个变异位置, 然后交换这两个变异位置上的基因.

**例 1.22** 设选中的个体向量  $D = 20\ 12\ 16\ 19\ 21\ 30$ , 若随机产生的两个变异位置分别是 2 和 4, 则变异后的新的个体向量是:

$$D' = 20\ 19\ 16\ 12\ 21\ 30 \quad (1.86)$$

**例 1.23** 用遗传算法求函数  $f(x) = x_2$  的最大值, 其中  $x \in [0, 31], x \in \mathbb{Z}$  间的整数.

解: 这个问题本身比较简单, 其最大值很显然是在  $x = 31$  处. 但作为一个例子, 它有着较好的示范性和可理解性.

按照遗传算法, 其求解过程如下:

(1) 编码

由于  $x$  的定义域是区间  $[0, 31]$  上的整数, 由 5 位二进制数即可全部表示. 因此, 可采用二进制编码方法, 其编码串的长度为 5.

例如, 用二进制串 00000 来表示  $x = 0$ , 11111 来表示  $x = 31$  等. 其中的 0 和 1 为基因值.

(2) 生成初始种群

若假设给定的种群规模  $N = 4$ , 则可用 4 个随机生成的长度为 5 的二进制串作为初始种群. 再假设随机生成的初始种群 (即第 0 代种群) 为:

$$\begin{aligned} s_{01} &= 01101 & s_{02} &= 11001 \\ s_{03} &= 01000 & s_{04} &= 10010 \end{aligned}$$

(3) 计算适应度

要计算个体的适应度, 首先应该定义适应度函数. 由于本例是求  $f(x)$  的最大值, 因此可直接用  $f(x)$  来作为适应度函数. 即:

$$f(s) = f(x) \quad (1.87)$$

其中的二进制串  $s$  对应着变量  $x$  的值. 根据此函数, 初始种群中各个个体的适应值及其所占比例如表1-7所示.

表 1-7 初始种群情况表

编号	个体串 (染色体)	$x$	适应值	百分比%	累计百分比%	选中次数
$S_{01}$	01101	13	169	14.44	14.44	1
$S_{02}$	11001	25	625	52.88	67.18	2
$S_{03}$	01000	8	64	5.41	72.59	0
$S_{04}$	10010	18	324	27.41	100	1

可以看出, 在 4 个个体中  $S_{03}$  的适应值最大, 是当前最佳个体.

(4) 选择操作

假设采用轮盘赌方式选择个体, 且依次生成的 4 个随机数 (相当于轮盘上指针所指的数) 为 0.85、0.32、0.12 和 0.46, 经选择后得到的新的种群为:

表 1-8 初始种群情况表

S01=10010
S02=11001
S03=01101
S04=11001

其中, 染色体 11001 在种群中出现了 2 次, 而原染色体 01000 则因适应值太小而被淘汰

(5) 交叉

假设交叉概率  $\pi$  为 50%, 则种群中只有 1/2 的染色体参与交叉. 若规定种群中的染色体按顺序两两配对交叉, 且有  $S_{01}$  与  $S_{02}$  交叉,  $S_{03}$  与  $S_{04}$  不交叉, 则交叉情况如表1-9所示.

表 1-9 初始种群的交叉情况表

编号	个体串 (染色体)	交叉对象	交叉位	子代	适应值
$S_{01}$	10010	$S_{02}$	3	10001	289
$S_{02}$	11001	$S_{01}$	3	11010	676
$S_{03}$	01101	$S_{04}$	N	01101	169
$S_{04}$	11001	$S_{03}$	N	11001	625

可见, 经交叉后得到的新的种群为:

表 1-10 种群情况表

$S_{01}=10001$
$S_{02}=11010$
$S_{03}=01101$
$S_{04}=11001$

#### (6) 变异

变异概率  $P_m$  一般都很小, 假设本次循环中没有发生变异, 则变异前的种群即为进化后所得到的第 1 代种群. 即:

表 1-11 种群情况表

$S_{11}=10001$
$S_{12}=11010$
$S_{13}=01101$
$S_{14}=11001$

然后, 对第 1 代种群重复上述 (4)-(6) 的操作

对第 1 代种群, 同样重复上述 (4)-(6) 的操作. 其选择情况如表 1-12 所示.

表 1-12 第 1 代种群的选择情况表

编号	个体串 (染色体)	$x$	适应值	百分比%	累计百分比%	选中次数
$S_{11}$	10001	27	289	16.43	16.437	1
$S_{12}$	11010	26	676	38.43	54.86	2
$S_{13}$	01101	13	169	9.61	64.47	0
$S_{14}$	11001	25	625	35.53	100	1

其中若假设按轮盘赌选择时依次生成的 4 个随机数为 0.14、0.51、0.24 和 0.82, 经选择后得到的新的种群为:

表 1-13 种群情况表

$S_{11}=10001$
$S_{12}=11010$
$S_{13}=11010$
$S_{14}=11001$

可以看出, 染色体 11010 被选择了 2 次, 而原染色体 01101 则因适应值太小而被淘汰.

对第 1 代种群, 其交叉情况如表1-14所示.

表 1-14 第 1 代种群的交叉情况表

编号	个体串 (染色体)	交叉对象	交叉位	子代	适应值
$S_{11}$	10001	$S_{12}$	3	10010	324
$S_{12}$	11010	$S_{11}$	3	11001	625
$S_{13}$	11010	$S_{14}$	2	11001	625
$S_{14}$	11001	$S_{13}$	2	11010	675

可见, 经杂交后得到的新的种群为:

表 1-15 种群情况表

$S_{11}=10010$
$S_{12}=11001$
$S_{13}=11001$
$S_{14}=11010$

可以看出, 第 3 位基因均为 0, 已经不可能通过交配达到最优解. 这种过早陷入局部最优解的现象称为早熟. 为解决这一问题, 需要采用变异操作.

对第 1 代种群, 其变异情况如表 1-16 所示.

表 1-16 第 1 代种群的变异情况表

编号	个体串 (染色体)	是否变异	变异位	子代	适应值
$S_{11}$	10010	N		10010	324
$S_{12}$	11001	N		11001	625
$S_{13}$	11001	N		11001	625
$S_{14}$	11010	Y		11110	900

它是通过对  $S_{14}$  的第 3 位的变异来实现的. 变异后所得到的第 2 代种群为:

表 1-17 种群情况表

$S_{21}=10010$
$S_{22}=11001$
$S_{23}=11001$
$S_{24}=11110$

接着, 再对第 2 代种群同样重复上述 (4)-(6) 的操作:

对第 2 代种群, 同样重复上述 (4)-(6) 的操作. 其选择情况如表 1-19 所示.



表 1-18 第 2 代种群的选择情况表

编号	个体串 (染色体)	$x$	适应值	百分比%	累计百分比%	选中次数
$S_{21}$	10010	18	324	23.92	23.92	1
$S_{22}$	11001	25	625	22.12	46.04	1
$S_{23}$	11001	25	625	22.12	68.16	1
$S_{24}$	11110	30	900	31.84	100	1

对第 2 代种群, 其交叉情况如表1-19所示.

表 1-19 第 2 代种群的选择情况表

编号	个体串 (染色体)	交叉对象	交叉位	子代	适应值
$S_{21}$	11001		$S_{22}$	3	11010 676
$S_{22}$	10010		$S_{21}$	3	10001 289
$S_{23}$	11001		$S_{24}$	4	11000 576
$S_{24}$	11110		$S_{23}$	4	11111 961

这时, 函数的最大值已经出现, 其对应的染色体为 11111, 经解码后可知问题的最优解是在点  $x = 31$  处. 求解过程结束. 其中若假设按轮盘赌选择时依次生成的 4 个随机数为 0.42、0.15、0.59 和 0.91, 经选择后得到的新的种群为:

表 1-20 种群情况表

$S_{21}=11001$
$S_{22}=10010$
$S_{23}=11001$
$S_{24}=11110$

1.7  PSO

## 1.8 QBFA

### 1.9 其它算法 \*

#### 1.9.1 鱼群算法

#### 1.9.2 鸟群算法

#### 1.9.3 象群算法

#### 1.9.4 蚁群算法

#### 1.9.5 狼群算法

#### 1.9.6 蝙蝠算法

#### 1.9.7 随机森林

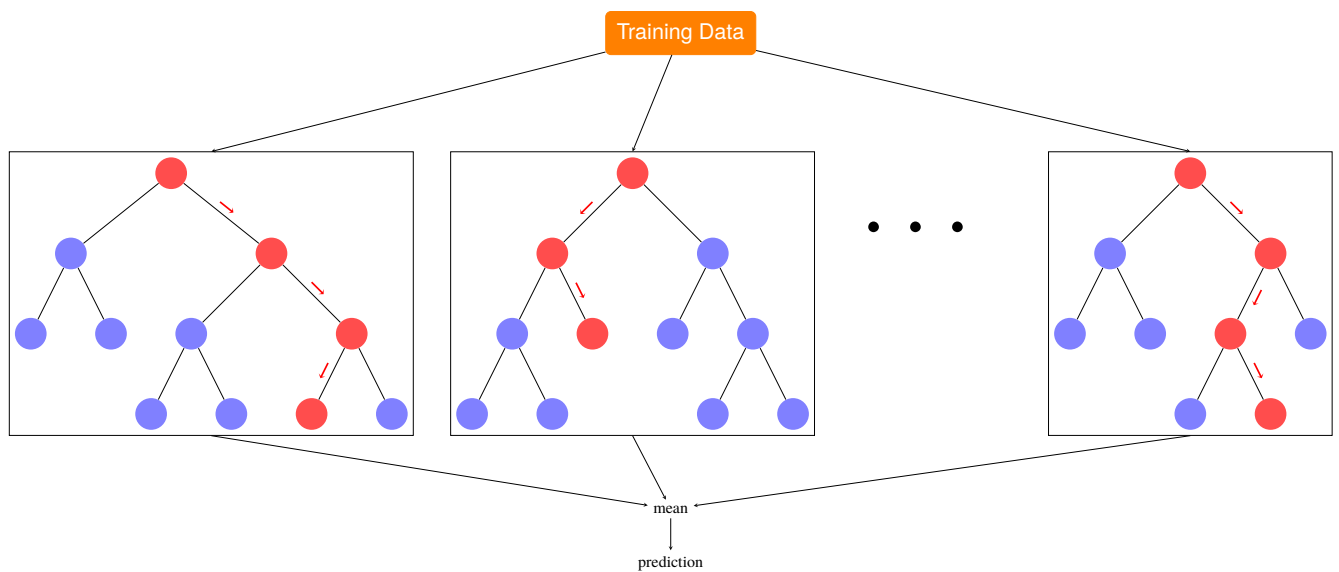


图 1-23 语义网络的基本网元

#### 1.9.8 ADP 聚类算法

A method for autonomous data partitioning, 2018.8 [GuAngelov2018-5667]

<https://ww2.mathworks.cn/matlabcentral/fileexchange/67463-autonomous-clustering>  
s\_tid=prof\_contriblnk

The package contains:

1. The recently introduced autonomous data partitioning algorithm;
2. A demo for offline data partitioning;
3. A demo for online data partitioning;
4. A demo for creating a hybrid between the offline and online versions

Reference: X. Gu, P. Angelov and J. Principe, “A Method for Autonomous Data Partitioning,” *Information Sciences*, 2018, DOI:10.1016/j.ins.2018.05.030.

For any queries about the codes, please contact Prof. Plamen P. Angelov (p.angelov@lancaster.ac.uk) and Mr. Xiaowei Gu (x.gu3@lancaster.ac.uk)

The proposed ADP algorithm has an offline version and an evolving version. Its offline version is based on the ranks of the data samples in terms of their densities and mutual distances instead of the commonly used means and variances. Ranks are very important but other approaches avoid them because they are nonlinear and discrete operators. Therefore, the offline version is more stable and effective in partitioning static datasets. The evolving version is also generic and fully data-driven. It can continue the offline partitioning process with the newly arrived data, but can also start without initial conditions.

Lifting these assumptions is possible with information theoretic [JenssenErdogmus2007-5673] and spectral clustering [vonLuxburg2007-5674], but at the expense of much higher computational cost.

The proposed ADP approach is proposed within the recently introduced Empirical Data Analysis (EDA) framework [Angelov2014Outside, Angelov2016-7844219, Angelov2017Empirical]. EDA resembles statistical learning in its nature but is free from the range of assumptions made in traditional probability theory and statistical learning approaches [Angelov2014Outside, Angelov2016-7844219, Angelov2017Empirical]. EDA measures play an instrumental role in the proposed ADP approach for extracting the ensemble properties from the observed data (see Fig. 1(b)), and frees the ADP algorithm from the requirement to make prior assumptions on the data generation model and user- and problem-specific parameters (see Fig. 1(a)). Most importantly, they guarantee objectiveness of the partitioning results.

Firstly, we define the data set/stream in the Euclidean data space  $\mathbb{R}^N$  as  $\{\mathbf{x}\}_K = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_K\}$ ,  $\mathbf{x}_k = [x_{k,1}, x_{k,2}, \dots, x_{k,N}]^T \in \mathbb{R}^N$ , where  $k = 1, 2, \dots, K$  is an order index. We, further, consider that some data samples within the data set/stream can have the same value, i.e.  $\exists \mathbf{x}_k = \mathbf{x}_j, k \neq j$ . The set of sorted unique data samples is denoted as  $\{\mathbf{u}\}_{L_K} = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{L_K}\}$ ,  $\mathbf{u}_j = [u_{j,1}, u_{j,2}, \dots, u_{j,N}]^T (j = 1, 2, \dots, L_K)$ ,  $\{\mathbf{u}\}_{L_K} \subseteq \{\mathbf{x}\}_K, L_K \leq K$  with

the corresponding occurrence frequencies  $\{f\}_{L_K} = \{f_1, f_2, \dots, f_{L_K}\}$ ,  $\sum_{k=1}^{L_K} f_k = 1$ .

In this paper, we use the natural metric of the space of samples (i.e. Euclidean distance) for clarity, however, various types of distances within the data space,  $\mathbb{R}^N$  can be considered as well. In the remainder of this paper, all the derivations are conducted at the  $K$ th time instance except when specifically mentioned.

In this section, we will summarize the nonparametric EDA measures that we use in ADP:

I) local density,  $D$  [Angelov2016-7844219, Angelov2017Empirical];

II) global density,  $D^G$  [Angelov2016-7844219, Angelov2017Empirical];

and their recursive implementations to make the paper self-contained.

### 3.1. Local density

Local density [Angelov2016-7844219] is a measure within EDA framework identifying the main local mode of the data distribution and is derived empirically from all the observed data samples in a parameter-free way. The local density,  $D$  of the data sample  $\mathbf{x}_k$  is expressed as follows ( $k = 1, 2, \dots, K; L_K > 1$ ) [Angelov2016-7844219, Angelov2017Empirical]:

$$D_K(\mathbf{x}_k) = \frac{\sum_{j=1}^K \sum_{l=1}^K d^2(\mathbf{x}_j, \mathbf{x}_l)}{2K \sum_{l=1}^K d^2(\mathbf{x}_k, \mathbf{x}_l)} \quad (1.88)$$

where  $d(\mathbf{x}_k, \mathbf{x}_l)$  is the distance between data samples  $\mathbf{x}_k$  and  $\mathbf{x}_l$ ; the coefficient 2 is used in the denominator for normalization (because each distance is counted twice in the numerator).

For the case of Euclidean distance, the calculation of  $\sum_{l=1}^K d^2(\mathbf{x}_k, \mathbf{x}_l) = \sum_{l=1}^K \|\mathbf{x}_k - \mathbf{x}_l\|^2$  and  $\sum_{j=1}^K \sum_{l=1}^K d^2(\mathbf{x}_j, \mathbf{x}_l) = \sum_{j=1}^K \sum_{l=1}^K \|\mathbf{x}_j - \mathbf{x}_l\|^2$  can be simplified by using the mean of  $\{\mathbf{x}\}_K$ ,  $\boldsymbol{\mu}_K$  and the average scalar product,  $X_K$  as [Angelov2016-7844219, Angelov2017Empirical]:

$$\sum_{l=1}^K \|\mathbf{x}_k - \mathbf{x}_l\|^2 = K(\|\mathbf{x}_k - \boldsymbol{\mu}_K\|^2 + X_K - \|\boldsymbol{\mu}_K\|^2) \quad (1.89)$$

$$\sum_{j=1}^K \sum_{l=1}^K \|\mathbf{x}_j - \mathbf{x}_l\|^2 = 2K^2(X_K - \|\boldsymbol{\mu}_K\|^2) \quad (1.90)$$

$\boldsymbol{\mu}_K$  and  $X_K$  can be updated recursively as [Angelov2012Autonomous]:

$$\boldsymbol{\mu}_k = \frac{k-1}{k} \boldsymbol{\mu}_{k-1} + \frac{1}{k} \mathbf{x}_k, \boldsymbol{\mu}_1 = \mathbf{x}_1, k = 1, 2, \dots, K \quad (1.91)$$

$$X_k = \frac{k-1}{k} X_{k-1} + \frac{1}{k} \|\mathbf{x}_k\|^2, X_1 = \|\mathbf{x}_1\|^2, k = 1, 2, \dots, K \quad (1.92)$$

The recursive calculation of  $\boldsymbol{\mu}_K$  and  $X_K$  allows for “one pass” calculation, thus, ensures computation-efficiency because only the key aggregated/summarized information has to be kept in memory.

Combining (1.88)-(1.92), we can re-formulate the local density,  $D$  in a recursive form:

$$D_K(\mathbf{x}_k) = \frac{1}{1 + \frac{\|\mathbf{x}_k - \boldsymbol{\mu}_K\|^2}{X_K - \|\boldsymbol{\mu}_K\|^2}} \quad (1.93)$$

From (1.93) we can see that when the Euclidean distance is used, the local density,  $D$  behaves as a **Cauchy function**, while there is **no prior assumption about the type of the distribution involved**. Note that  $0 < D_K(\mathbf{x}_k) \leq 1$  and the closer  $\mathbf{x}_k$  is to the main local mode, the higher the value of  $D_K(\mathbf{x}_k)$  is.

### 3.2. Global density

The global density,  $D^G$  estimated at the unique data sample  $\mathbf{u}_k$  is a weighted sum of its local density by the corresponding occurrence,  $f_k$  ( $k = 1, 2, \dots, L_K; L_K > 1$ ), expressed as follows [Angelov2016-7844219, Angelov2017Empirical]:

$$D_K^G(\mathbf{u}_k) = \frac{f_k}{\sum_{j=1}^{L_K} f_j} D_K(\mathbf{u}_k) = \frac{f_k}{1 + \frac{\|\mathbf{u}_k - \boldsymbol{\mu}_K\|^2}{X_K - \|\boldsymbol{\mu}_K\|^2}} \quad (1.94)$$

It is clear that it behaves locally as a Cauchy function, but has multiple local modes/peaks which depend on  $f_k$  (if  $L_K = K$ , global density reduces to the local density but with a smaller amplitude). The largest of the peaks is called the global peak. The global density can directly disclose the data pattern without any further pre-processing. This property can be very useful in real cases where the units of measurement are fixed and data samples measured at different indices are likely to share the same values. The reason we do not stop with  $D^G$  but move further with partitioning is that although it is fully automatic and objective,  $D^G$  often reveals too many local peaks.

## 4. The autonomous data partitioning algorithm

In this section, we will describe the proposed ADP algorithm for further refining the local modes of the data set/stream. The local modes are defined as the local maxima of the global data density and are constructed directly from samples. These local modes play a key role in partitioning the data space into shape-free data clouds [Angelov2012Autonomous, Angelov2014Outside] by aggregating data samples around them and forming naturally a Voronoi tessellation [OkabeBoots2000-5677]. We will present two independent algorithms for the two versions, i) offline and ii) evolving.

### 4.1. Offline ADP algorithm

The offline ADP algorithm works with the global density,  $D^G$  of the observed data samples. It is based on the ranks of the data samples in terms of their global densities and mutual distributions. Its main procedure is described as follows:

### Stage 1: Rank order the samples with regards to the distance to the global mode

We start by organizing the unique data samples  $\{\mathbf{u}\}_K$  into an indexing list, denoted by  $\{\mathbf{u}^*\}_K$  based on their mutual distances and values of the global density,  $D^G$ . Firstly, the global densities of the unique data samples,  $D_K^G(\mathbf{u}_i)(i = 1, 2, \dots, L_K)$  are calculated using (1.94). The unique data sample with the highest global density is then selected as the first element of the indexing list  $\{\mathbf{u}^*\}_{L_K}$ :

$$\mathbf{u}^{*1} = \arg \max_{j=1,2,\dots,L_K} (D_K^G(\mathbf{u}_j)) \quad (1.95)$$

We set  $\mathbf{u}^{*1}$  as the first reference point:  $\mathbf{u}^{*r} \leftarrow \mathbf{u}^{*1}$  and remove  $\mathbf{u}^{*1}$  from  $\{\mathbf{u}\}_{L_K}$ . Then, by selecting out the unique data sample which is nearest to  $\mathbf{u}^{*r}$ , the second element of  $\{\mathbf{u}^*\}_{L_K}$  denoted by  $\mathbf{u}^{*2}$  is identified and it is set as the new reference point  $\mathbf{u}^{*r} \leftarrow \mathbf{u}^{*2}$  and, is also removed from  $\{\mathbf{u}\}_{L_K}$ . The process is repeated until  $\{\mathbf{u}\}_{L_K}$  becomes empty, and the rank ordered list  $\{\mathbf{u}^*\}_{L_K}$  is finally derived. Based on this list, we can rank the global density of the unique data samples as:

$$\{D_K^G(\mathbf{u}^*)\} = \{D_K^G(\mathbf{u}^{*1}), D_K^G(\mathbf{u}^{*2}), \dots, D_K^G(\mathbf{u}^{*L_K})\}. \quad (1.96)$$

An example using the wine dataset [AeberhardWinedata1992] is shown in Fig. 2. Fig. 2(a) shows the global density of the wine dataset, Fig. 1–24 shows the ranked global density.

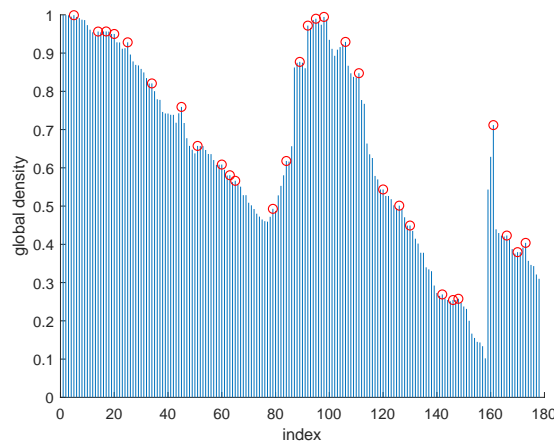


图 1–24 The ranked  $D^G$  and the identified local maxima.

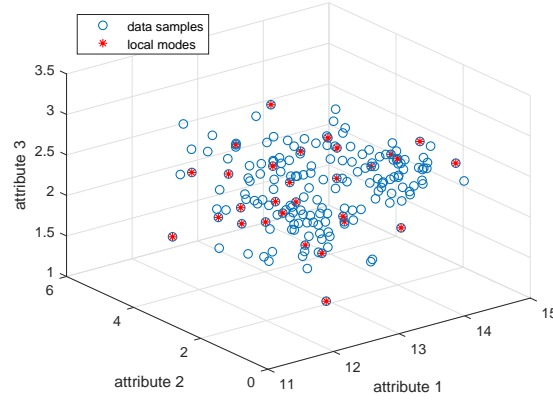


图 1-25 The location of local maxima in the data space.

### Stage 2: Detecting local maxima (local modes)

Based on the list  $\{\mathbf{u}^*\}_{L_K}$  and the ranked global density  $\{D_K^G(\mathbf{u}^*)\}$ , we can identify all the data samples with the local maxima of  $D^G$  directly as follows:

$$\begin{aligned} \text{IF } \text{sgn} D_K^G(\mathbf{u}^{*j}) - D_K^G(\mathbf{u}^{*j+1}) = 1 \text{ AND } \text{sgn} D_K^G(\mathbf{u}^{*j-1}) - D_K^G(\mathbf{u}^{*j}) = -1 \\ \text{THEN } \mathbf{u}^{*j} \text{ is a local maximum of } D^G \end{aligned} \quad (1.97)$$

where

$$\text{sgn}(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}$$

is the sign function; we denote the collection of data samples with the local maxima of  $D^G$  as  $\{\mathbf{u}^{**}\}_{L_K^*} = \{\mathbf{u}^{*j} | j = 1, 2, \dots, L_K^* (L_K^* < L_K)\}$ . The local maxima (peaks) identified from the wine dataset [AeberhardWinedata1992] are marked by red circles in Fig. 2(b). The locations of the local maxima in the data space are depicted in Fig. 1-25.

### Stage 3: Forming data clouds

The local peaks identified from the indexing list, namely,  $\{\mathbf{u}^{**}\}_{L_K^*}$ , are then used to attract the data samples  $\{\mathbf{x}\}_K$  that are closer to them using a min operator:

$$\text{winning cloud} = \arg \min_{j=1,2,\dots,L_K^*} (\|\mathbf{x}_i - \mathbf{u}^{*j}\|) \quad i = 1, 2, \dots, K; L_K^* > 1 \quad (1.98)$$

By assigning all the data samples to the nearest local maxima, a number of Voronoi tessellations [OkabeBoots2000-5677] are naturally formed and data clouds are built around

the local maxima [Angelov2012Autonomous, Angelov2014Outside]. More importantly, the process is free from any threshold.

After the data clouds are formed, the actual centre (mean)  $\mu_j$  and the standard deviation  $\sigma_j (j = 1, 2, \dots, L_K^*)$  per data cloud and the support,  $S^j$ , namely, the number of data samples within the data cloud can be calculated easily. Note that, all this procedure is post factum, i.e. it is determined bottom up from the data without a prior assumption, except the selection of the metric.

#### Stage 4: Filtering local modes

The data clouds formed in the previous stage may contain some less representative ones; therefore, in this stage, we filter the initial Voronoi tessellations and combine them into larger, more meaningful data clouds. The global densities at the data clouds centres  $\{\mu\}_{L_K^*}$  are firstly calculated as follows ( $j = 1, 2, \dots, L_K^*$ ):

$$D_K^G(\mu^j) = \frac{S^j}{1 + \frac{\|\mu^j - \mu_K\|^2}{X_K - \|\mu_K\|^2}} \quad (1.99)$$

where  $S^j$  is the support of the data cloud.

In order to identify the centres with the local maxima of the global density, we introduce the following three objectively derived quantifiers of the data pattern:

$$\eta_K^c = 2 \sum_{p=1}^{L_K^*-1} \sum_{q=p+1}^{L_K^*} \|\mu^p - \mu^q\| / L_K^* (L_K^* - 1) \quad (1.100)$$

$$\gamma_K^c = \frac{\sum_{x \neq y, \|x-y\| \leq \eta_K^c, x, y \in \{\mu\}_{L_K^*}} \|x - y\|}{M_\eta} \quad (1.101)$$

$$\lambda_K^c = \frac{\sum_{x \neq y, \|x-y\| \leq \gamma_K^c, x, y \in \{\mu\}_{L_K^*}} \|x - y\|}{M_\gamma} \quad (1.102)$$

$\eta_K^c$  is the average Euclidean distance between any pair of existing local modes.  $\gamma_K^c$  is the average Euclidean distance between any pair of existing local modes within a distance less than  $\eta_K^c$ , and  $M_\eta$  in (1.100) is the number of such local mode pairs.  $\lambda_K^c$  is the average Euclidean distance between any pair of existing local modes within a distance less than  $\gamma_K^c$ ,  $M_\gamma$  in (1.102) is the number of such local mode pairs.

Note that  $\eta_K^c$ ,  $\gamma_K^c$  and  $\lambda_K^c$  are not problem- specific and are parameter-free. The relationship between  $\eta_K^c$ ,  $\gamma_K^c$  and  $\lambda_K^c$  is depicted in Fig. 3 using the wine dataset [AeberhardWinedata1992] again, where one can see that  $\lambda_K^c$  is a smaller value compared with  $\eta_K^c$  and  $\gamma_K^c$  which are obtained



as the average distance of two very close data cloud centres. The quantifier  $\lambda_K^c$  can be viewed as the estimation of the distances between the strongly connected data clouds condensing the local information from the whole dataset. Moreover, instead of relying on a fixed threshold, which may frequently fail,  $\eta_K^c, \gamma_K^c$  and  $\lambda_K^c$  are derived from the dataset objectively and conforming with the concept of a mode. Experience has shown that they provide meaningful tessellations, regardless of the distribution of the data.

Each centre  $\mu^j (j = 1, 2, \dots, L_K^*)$  is compared with the centres of neighbouring data clouds in terms of the global density:

$$\text{IF } D_K^G(\mathbf{u}^j) = \max \left\{ \{D_K^G(\mu)\}_n^j, D_K^G(\mu^j) \right\} \text{ THEN } \mathbf{u}^j \text{ is a local maximum} \quad (1.103)$$

where  $\{D_K^G(\mathbf{u}^{*j})\}_n^j$  is the collection of global densities of the neighbouring centres, which satisfy the following condition:

$$\text{IF } \left( \|\mu^i - \mu^j\| \leq \frac{\lambda_K^c}{2} \right) \text{ THEN } \mathbf{u}^i \text{ is neighbouring } \mathbf{u}^j \quad (1.104)$$

The criterion of neighbouring range is defined in this way because two centres with the distance smaller than  $\gamma_K^c$  can be considered to be potentially relevant in the sense of spatial distance;  $\lambda_K^c$  is the average distance between the centres of any two potentially relevant data clouds. Therefore, when the condition (16) is satisfied, both  $\mu^i$  and  $\mu^j$  are highly influencing each other and, the data samples within the two corresponding data clouds are strongly connected. Therefore, the two data clouds are considered as neighbours. This criterion also guarantees that only small-size (less important) data clouds that significantly overlap with large-size (more important) ones will be removed during the filtering operation.

After the filtering operation (stage 4), the data cloud centres with local maximum global densities denoted by  $\{\mu^*\}_{L_K^{**}} = \mu^{*j} | j = 1, 2, \dots, L_K^{**}, L_K^{**} \leq L_K^*$  are obtained. Then,  $\{\mu^*\}_{L_K^{**}}$  are used as local modes for forming data clouds in stage 3 and are filtered in stage 4.

Stages 3 and 4 are repeated until all the distances between the existing local modes exceed  $\frac{\lambda_K^c}{2}$ . Finally, we obtain the remaining centres with the local maxima of  $D^G$ , denoted by  $\{\mu^o\}$ , and use them as the local modes to form data clouds using (1.97).

After the data clouds are formed, the corresponding centres, standard deviations, supports, members and other parameters of the formed data clouds can be extracted post factum. The final partitioning result of the wine dataset [AeberhardWinedata1992] is depicted in Fig. 4, where the dots in different colours stand for data samples from different data clouds, the black asterisks stand for the centres of the data clouds.

#### 4.2. Evolving ADP algorithm

The proposed evolving ADP algorithm works with the local density,  $D$  of the streaming data. This algorithm is able to start “from scratch”, i.e. with a single sample. In addition, a hybrid between the evolving and the offline versions is also possible.

The main procedure of the evolving algorithm is as follows.

##### Stage 1: Initialization

We select the first data sample within the data stream as the first local mode. The proposed algorithm then starts to self-evolve its structure and update the parameters based on the arriving data samples.

##### Stage 2: System structure and meta-parameters update

For each newly arrived data sample at the current time instance  $k \leftarrow k + 1$ , denoted as  $\mathbf{x}_k$ , the global meta-parameters  $\boldsymbol{\mu}_k$  and  $X_k$  are updated with  $\mathbf{x}_k$  firstly using (1.91) and (1.92). The local density at  $\mathbf{x}_k$  and the centres of all the existing data clouds,  $D_k(\mathbf{x}_k)$  and  $D_k(\boldsymbol{\mu}_k^i)(i = 1, 2, \dots, C_k)$  are calculated using (1.93); here, we use  $C_k$  as the number of existing local modes at the  $k$ th time instance.

Then, the following condition [Angelov2012Autonomous] is checked to decide whether  $\mathbf{x}_k$  will form a new data cloud:

$$\begin{aligned} &\text{IF } D_k(\mathbf{x}_k) > \max_{i=1}^{C_k}(D_k(\boldsymbol{\mu}_k^i)) \text{ OR } D_k(\mathbf{x}_k) < \min_{i=1}^{C_k}(D_k(\boldsymbol{\mu}_k^i)) \\ &\text{THEN } (\mathbf{x}_k \text{ becomes a new local point}) \end{aligned} \quad (1.105)$$

If the condition is met, a new data cloud is added with  $\mathbf{x}_k$  as its local mode ( $C_k \leftarrow C_k + 1, \boldsymbol{\mu}_k^{C_k} \leftarrow \mathbf{x}_k$  and  $S_k^{C_k} \leftarrow 1$ ). Otherwise, the existing local mode closest to  $\mathbf{x}_k$  is found, denoted as  $\boldsymbol{\mu}_k^n$ . Then, the following condition is checked before  $\mathbf{x}_k$  is assigned to the data cloud formed around  $\boldsymbol{\mu}_k^n$ :

$$\text{IF } \|\mathbf{x}_k - \boldsymbol{\mu}_k^n\| \leq \frac{\eta_k^c}{2}, \text{ THEN } (\mathbf{x}_k \text{ is assigned to } \boldsymbol{\mu}_k^n) \quad (1.106)$$

However, it is not computationally efficient to calculate  $\eta_k^c$  at each time when a new data sample arrives. Since the average distance between all the data samples  $\eta_k^d$  is approximately equal to  $\eta_k^c$ ,  $\eta_k^c \approx \eta_k^d$ ,  $\eta_k^c$  can be replaced as:

$$\eta_k^c \approx \eta_k^d = \sqrt{\frac{\sum_{i=1}^k \sum_{l=1}^k \|\mathbf{x}_i - \mathbf{x}_l\|^2}{k^2}} = \sqrt{2(X_k - \|\boldsymbol{\mu}_k\|^2)} \quad (1.107)$$

If the condition (1.106) is satisfied, then  $x_k$  is associated with the nearest existing local mode  $\mu_k^n$  and the meta-parameters of  $\mu_k^n$  are updated as follows:

$$S_k^n \leftarrow S_k^n + 1 \quad (1.108)$$

$$\mu_k^n \leftarrow \frac{S_k^n - 1}{S_k^n} \mu_k^n + \frac{1}{S_k^n} x_k \quad (1.109)$$

If the condition (1.106) is not satisfied, then  $x_k$  starts a new data cloud: ( $C_k \leftarrow C_k + 1, \mu_k^{C_k} \leftarrow x_k$  and  $S_k^{C_k} \leftarrow 1$ ). The local modes and supports of other data clouds that do not get the new data sample stay the same for the next processing cycle. After the update of the system structure and the meta-parameters, the algorithm is ready for the next data sample.

### Stage 3: Forming data clouds

When there are no more data samples, the identified local modes (renamed as  $\{\mu^o\}$ ) are used to build data clouds using (1.98). The parameters of these data clouds can be extracted post factum. The main procedure of the proposed ADP algorithm (evolving version) is presented in the following pseudo code.

In this subsection, a number of benchmark datasets are used in the performance evaluation as tabulated in Table 1. During the experiments, we assume that we do not have any prior knowledge about the benchmark datasets. The following well-known algorithms are used for comparison:

- I MS: Mean-shift clustering algorithm 2002 [DPComaniciu2002-5678];
- II SUB: Subtractive clustering algorithm 1994 [Chiu1994-5679]; subclust–Find cluster centers using subtractive clustering-matlab 2006a up
- III DBS: DBScan clustering algorithm 1996 [EsterKriegel1996-5680];  
<http://yarpiz.com/255/ypml110-dbscan-clustering>
- IV SOM: Self-organizing map algorithm 1997 [Kohonen1997-5691];
- V ELM: Evolving local means clustering algorithm 2012 [RPDutta2012-5681];
- VI DP: Density peaks clustering algorithm 2014 Science [RodriguezLaio2014-5691];
- VII NMM: Nonparametric mixture model based clustering algorithm 2006[Blei2006-5683];
- VIII NMI: Nonparametric mode identification based clustering algorithm 2007 [LiRay2007-5690];
- IX CEDS: Clustering of evolving data streams algorithm 2017 [HydeAngelov2017-5682].

Fully online clustering of evolving data streams into arbitrarily shaped clusters, 2017 [HydeAngelov2017-5682]

In recent times there has been an increase in data availability in continuous data streams and clustering of this data has many advantages in data analysis. It is often the case that these

data streams are not stationary, but evolve over time, and also that the clusters are not regular shapes but form arbitrary shapes in the data space. Previous techniques for clustering such data streams are either hybrid online / offline methods, windowed offline methods, or find only hyper-elliptical clusters. In this paper we present a fully online technique for clustering evolving data streams into arbitrary shaped clusters. It is a two stage technique that is accurate, robust to noise, computationally and memory efficient, with a low time penalty as the number of data dimensions increases. The first stage of the technique produces micro-clusters and the second stage combines these micro-clusters into macro-clusters. Dimensional stability and high speed is achieved through keeping the calculations both simple and minimal using hyper-spherical micro-clusters. By maintaining a graph structure, where the micro-clusters are the nodes and the edges are its pairs with intersecting micro-clusters, we minimise the calculations required for macro-cluster maintenance. The micro-clusters themselves are described in such a way that there is no calculation required for the core and shell regions and no separate definition of outer micro-clusters necessary. We demonstrate the ability of the proposed technique to join and separate macro-clusters as they evolve in a fully online manner. There are no other fully online techniques that the authors are aware of and so we compare the technique with popular online / offline hybrid alternatives for accuracy, purity and speed. The technique is then applied to real atmospheric science data streams and used to discover short term, long term and seasonal drift and their effects on anomaly detection. As well as having favourable computational characteristics, the technique can add analytic value over hyper-elliptical methods by characterising the cluster hyper-shape using Euclidean or fractal shape factors. Because the technique records macro-clusters as graphs, further analytic value accrues from characterising the order, degree, and completeness of the cluster-graphs as they evolve over time.

Rodriguez, Alex and Laio, Alessandro, [**RodriguezLaio2014-5691**], Clustering by fast search and find of density peaks, Science 2014

Cluster analysis is used in many disciplines to group objects according to a defined measure of distance. Numerous algorithms exist, some based on the analysis of the local density of data points, and others on predefined probability distributions. Rodriguez and Laio devised a method in which the cluster centers are recognized as local density maxima that are far away from any points of higher density. The algorithm depends only on the relative densities rather than their absolute values. The authors tested the method on a series of data sets, and its performance compared favorably to that of established techniques. Cluster analysis is aimed at classifying elements into categories on the basis of their similarity. Its applications range from astronomy

to bioinformatics, bibliometrics, and pattern recognition. We propose an approach based on the idea that cluster centers are characterized by a higher density than their neighbors and by a relatively large distance from points with higher densities. This idea forms the basis of a clustering procedure in which the number of clusters arises intuitively, outliers are automatically spotted and excluded from the analysis, and clusters are recognized regardless of their shape and of the dimensionality of the space in which they are embedded. We demonstrate the power of the algorithm on several test cases.

a The time stamps in the original dataset have been removed. b Number of attributes. c Number of samples. d Number of classes.

## 1.10 模糊计算

美国加州大学扎德 (Zadeh) 教授于 1965 年提出的模糊集合与模糊逻辑理论是模糊计算的数学基础. 它主要用来处理现实世界中因模糊而引起的不确定性. 目前, 模糊理论已经在推理、控制、决策等方面得到了非常广泛的应用. 本节主要讨论模糊理论的基础, 对于模糊推理将放到下一章讨论.

20 世纪 70 年代中期, 中国开始引进模糊数学. 80 年代在理论研究和应用研究方面都非常活跃并取得了大量研究成果, 主要代表人物有著名数学家汪培庄、张文修、蒲保明、刘应明、王国俊等教授. 1988 年, 汪培庄教授及其指导的几名博士生研制成功一台模糊推理机——分立元件样机, 它的推理速度为 1500 万次/秒; 这表明中国在突破模糊信息处理难关方面迈出了重要的一步, 确实是一项了不起的研究成果. 2005 年, 中国科学院院士刘应明教授获国际模糊系统协会 (IFSA) 授予 “Fuzzy Fellow” 称号, 他是作为首位非发达国家学者获此殊荣的。



图 1-26

中国著名学者周海中教授曾指出：“模糊数学的诞生，是科学技术发展的必然结果，更是现代数学发展的必然产物。但就现状而言，模糊数学的理论尚未成熟、体系还未形成，对它也还存在不同看法和意见；这些都有待日后完善和实践检验。”

#### 定义 1.24 模糊集


设  $U$  是给定论域，是把任意  $u \in U$  映射为  $[0, 1]$  上某个实值的函数，即

$$U \rightarrow [0, 1] \quad (1.110)$$

$$u \rightarrow \mu_F(u) \quad (1.111)$$

则称  $\mu_F(x)$  为定义在  $U$  上的一个隶属函数，由 (对所有) 所构成的集合  $F$  称为  $U$  上的一个模糊集，称为  $u$  对  $F$  的隶属度。



 **注 1.25.** ① 模糊集  $F$  完全是由隶属函数  $\mu_F$  来刻画的，把  $U$  中的每一个元素  $u$  都映射为  $[0, 1]$  上的一个值  $\mu_F(x)$ 。

②  $\mu_F(u)$  的值表示  $u$  隶属于  $F$  的程度，其值越大，表示  $u$  隶属于  $F$  的程度越高。当  $\mu_F(u)$  仅取 0 和 1 时，模糊集  $F$  便退化为一个普通集合。

#### 1.10.1 模糊集及其运算

**例 1.26** 设论域  $U = \{20, 30, 40, 50, 60\}$  给出的是年龄，请确定一个刻画模糊概念“年轻”的模糊集  $F$ 。

解：由于模糊集是用其隶属函数来刻画的，因此需要先求出描述模糊概念“青年”的

隶属函数. 假设对论域  $U$  中的元素, 其隶属函数值分别为:

$$\begin{aligned}\mu_F(20) &= 1, \mu_F(30) = 0.8, \mu_F(40) = 0.4 \\ \mu_F(50) &= 0.1, \mu_F(60) = 0\end{aligned}\quad (1.112)$$

则可得到刻画模糊概念“年轻”的模糊集

$$F = \{1, 0.8, 0.4, 0.1, 0\} \quad (1.113)$$

说明其含义.

(1) 离散且为有限论域的表示方法

设论域  $U = \{u_1, u_2, \dots, u_n\}$  为离散论域, 则其模糊集可表示为:

$$F = \{\mu_F(u_1), \mu_F(u_2), \dots, \mu_F(u_n)\} \quad (1.114)$$

为了能够表示出论域中的元素与其隶属度之间的对应关系, 扎德引入了一种模糊集的表示方式: 先为论域中的每个元素都标上其隶属度, 然后再用“+”号把它们连接起来, 即

$$F = \mu_F(u_1)/u_1 + \mu_F(u_2)/u_2 + \dots + \mu_F(u_n)/u_n \quad (1.115)$$

也可写成

$$F = \sum_{i=1}^n \mu_F(u_i)/u_i \quad (1.116)$$

其中,  $\mu_F(u_i)$  为  $u_i$  对  $F$  的隶属度; “ $\mu_F(u_i)/u_i$ ” 不是相除关系, 只是一个记号; “+” 也不是算术意义上的加, 只是一个连接符号.

$$\begin{aligned}F &= \{\mu_F(u_1)/u_1, \mu_F(u_2)/u_2, \dots, \mu_F(u_n)/u_n\} \\ F &= \{(\mu_F(u_1), u_1), (\mu_F(u_2), u_2), \dots, (\mu_F(u_n), u_n)\}\end{aligned}\quad (1.117)$$

在这种表示方法中, 当某个  $u_i$  对  $F$  的隶属度  $=0$  时, 可省略不写. 例如, 前面例 5.15 的模糊集  $F$  可表示为:

$$F = 1/20 + 0.8/30 + 0.6/40 + 0.2/50 \quad (1.118)$$

有时, 模糊集也可写成如下两种形式:

$$\begin{aligned}F &= \{\mu_F(u_1)/u_1, \mu_F(u_2)/u_2, \dots, \mu_F(u_n)/u_n\} \\ F &= \{(\mu_F(u_1), u_1), (\mu_F(u_2), u_2), \dots, (\mu_F(u_n), u_n)\}\end{aligned}\quad (1.119)$$

其中, 前一种称为单点形式, 后一种称为序偶形式.

### (2) 连续论域的表达方法

如果论域是连续的, 则其模糊集可用一个实函数来表示. 例如, 扎德以年龄为论域, 取  $U = [0, 100]$ , 给出了“年轻”与“年老”这两的模糊概念的隶属函数

$$\begin{aligned}\mu_{\text{老}} = \frac{(l)}{*_{\xi}}(u) &= \begin{cases} 0, & 0 \leq u \leq 50 \\ \left[1 + \left(\frac{5}{u-50}\right)^2\right]^{-1}, & 50 < u \leq 100 \end{cases} \\ \mu_{\text{年轻}}(u) &= \begin{cases} 1, & 0 \leq u \leq 25 \\ \left[1 + \left(\frac{u-25}{5}\right)^2\right]^{-1}, & 25 < u \leq 100 \end{cases}\end{aligned}\quad (1.120)$$

### (3) 一般表示方法

不管论域  $U$  是有限的还是无限的, 是连续的还是离散的, 扎德又给出了一种类似于积分的一般表示形式:

$$F = \int_{u \in U} \mu_F(u)/u \quad (1.121)$$

这里的记号不是数学中的积分符号, 也不是求和, 只是表示论域中各元素与其隶属度对应关系的总括.

#### 定义 1.27 模糊集相等

设  $F, G$  分别是  $U$  上的两个模糊集, 对任意  $u \in U$ , 都有  $\mu_F(u) = \mu_G(u)$  成立, 则称  $F$  等于  $G$ , 记为  $F = G$ .



#### 定义 1.28 模糊集包含

设  $F, G$  分别是  $U$  上的两个模糊集, 对任意  $u \in U$ , 都有  $\mu_F(u) \leq \mu_G(u)$  成立, 则称  $F$  等于  $G$ , 记为  $F \subseteq G$ .



#### 定义 1.29 模糊集并集交集

设  $F, G$  分别是  $U$  上的两个模糊集, 则  $F \cup G$ 、 $F \cap G$  分别称为  $F$  与  $G$  的并集、交集, 它们的隶属函数分别为:

$$\begin{aligned}F \cup G : \mu_{F \cup G}(u) &= \max_{u \in U} \{\mu_F(u), \mu_G(u)\} \\ F \cap G : \mu_{F \cap G}(u) &= \min_{u \in U} \{\mu_F(u), \mu_G(u)\}\end{aligned}\quad (1.122)$$





**定义 1.30 模糊集补集**

设  $F$  为  $U$  上的模糊集, 称  $\neg F$  为  $F$  的补集, 其隶属函数为:

$$\neg F : \mu_{\neg F}(u) = 1 - \mu_F(u) \quad (1.123)$$

**定义 1.31 模糊集补集**

设  $U = \{1, 2, 3\}$ ,  $F$  和  $G$  分别是  $U$  上的两个模糊集, 即

$$F = \{\text{小}\} = 1/1 + 0.6/2 + 0.1/3 \quad (1.124)$$

$$G = \{\text{大}\} = 0.1/1 + 0.6/2 + 1/3 \quad (1.125)$$

则

$$F \cup G = (1 \vee 0.1)/1 + (0.6 \vee 0.6)/2 + (0.1 \vee 1)/3 = 1/1 + 0.6/1 + 1/1,$$

$$F \cap G = (1 \wedge 0.1)/1 + (0.6 \wedge 0.6)/2 + (0.1 \wedge 1)/3 = 0.1/1 + 0.6/1 + 0.1/1,$$

$$\neg F = (1 - 1)/1 + (1 - 0.6)/2 + (1 - 0.1)/3 = 0.4/2 + 0.9/3.$$

从这个例子可以看出, 两个模糊集之间的运算实际上就是逐点对隶属函数作相应的运算.

**1.10.2 模糊关系及其运算**

**1.10.2.0.1 模糊关系的定义** 设  $V$  与  $W$  是两个普通集合,  $V$  与  $W$  的笛卡尔乘积为

$$V \times W = \{(v, w) | v \in V, w \in W\} \quad (1.126)$$

所谓从  $V$  到  $W$  的关系  $R$ , 是指  $V \times W$  上的一个子集, 即

$$R \subseteq V \times W \quad (1.127)$$

记为

$$V \xrightarrow{R} W \quad (1.128)$$

对于  $V \times W$  中的元素  $(v, w)$ , 若  $(v, w) \in R$ , 则称  $v$  与  $w$  有关系  $R$ ; 若  $(v, w) \notin R$ , 则称  $v$  与  $w$  没有关系.

**例 1.32** 设  $V = \{1\text{班}, 2\text{班}, 3\text{班}\}$ ,  $W = \{\text{男队}, \text{女队}\}$  则  $V \times W$  中有 6 个元素, 即


$$V \times W = \{(1\text{班}, \text{男队}), (2\text{班}, \text{男队}), (3\text{班}, \text{男队}), (1\text{班}, \text{女队}), (2\text{班}, \text{女队}), (3\text{班}, \text{女队})\} \quad (1.129)$$

其中, 每个元素是一代表队. 假设要进行一种双方对垒的循环赛, 则每一个赛局都是  $V \times W$  中的一个子集, 它构成了  $V \times W$  上的一个关系.

### 定义 1.33 模糊集

设  $F_i$  是  $U_i (i = 1, 2, \dots, n)$  上的模糊集, 则称

$$F_1 \times F_2 \times \dots \times F_n = \int_{u_1 \times u_2 \times \dots \times u_n} (\mu_{F_1}(u_1) \wedge \mu_{F_2}(u_2) \wedge \dots \wedge \mu_{F_n}(u_n)) / (u_1, u_2, \dots, u_n). \quad (1.130)$$

为  $F_1, F_2, \dots, F_n$  的笛卡尔乘积, 它是  $U_1 \times U_2 \times \dots \times U_n$  上的一个模糊集. 

### 定义 1.34 模糊关系

在  $\prod_{i=1}^n U_i$  上的一个  $n$  元模糊关系  $R$  是指以  $\prod_{i=1}^n U_i$  上为论域的一个模糊集, 记为

$$R = \int_{U_1 \times U_2 \times \dots \times U_n} \mu_R(u_1, u_2, \dots, u_n) / (u_1, u_2, \dots, u_n) \quad (1.131) \quad \text{♣}$$

**例 1.35** 设有一组学生  $U = u_1, u_2 = \{\text{秦学}, \text{郝玩}\}$ , 一些在计算机上的活动  $V = v_1, v_2, v_3 = \{\text{编程}, \text{上网}, \text{玩游戏}\}$  并设每个学生对各种活动的爱好程度分别为  $\mu_F(u_i, v_j) \ i = 1, 2; j = 1, 2, 3$ , 即

$$\mu_R(u_i, v_j) = \mu_R \begin{cases} (u_1, v_1), & (u_1, v_2), & (u_1, v_3) \\ (u_2, v_1), & (u_2, v_2), & (u_2, v_3) \end{cases} \quad (1.132)$$

则  $U \times V$  上的模糊关系  $R$  为


$$R = \begin{bmatrix} 0.9 & 0.6 & 0 \\ 0.2 & 0.3 & 0.8 \end{bmatrix} \quad (1.133)$$

#### 1.10.2.0.2 模糊关系的合成

**定义 1.36 模糊关系**

设  $R_1$  与  $R_2$  分别是  $U \times V$  与  $U \times W$  上的两个模糊关系, 则  $R_1$  与  $R_2$  的合成是从  $U$  到  $W$  的一个模糊关系, 记为  $R_1 \circ R_2$ . 其隶属函数为

$$\mu_{R_1 R_2}(u, w) = \vee \{ \mu_{R_1}(u, v) \wedge \mu_{R_2}(v, w) \} \quad (1.134)$$

其中,  $\wedge$  和  $\vee$  分别表示取最小和取最大. 

**例 1.37** 设有以下两个模糊关系

$$R_1 = \begin{bmatrix} 0.4 & 0.5 & 0.6 \\ 0.8 & 0.3 & 0.7 \end{bmatrix},$$


$$R_2 = \begin{bmatrix} 0.7 & 0.9 \\ 0.2 & 0.8 \\ 0.5 & 0.3 \end{bmatrix}$$

则  $R_1$  与  $R_2$  的合成是

$$R = R_1 \circ R_2 = \begin{bmatrix} 0.5 & 0.5 \\ 0.7 & 0.8 \end{bmatrix} \quad (1.135)$$

其方法是把  $R_1$  的第  $i$  行元素分别与  $R_2$  的第  $j$  列的对应元素相比较, 两个数中取最小者, 然后再在所得的一组最小数中取最大的一个, 并以此数作为  $R_1 \circ R_2$  的元素  $R(i, j)$ .

**1.10.2.0.3 模糊变换****定义 1.38 模糊变换**

设  $F = \{ \mu_F(u_1), \mu_F(u_2), \dots, \mu_F(u_n) \}$ , 是论域  $U$  上的模糊集,  $R$  是  $U \times V$  上的模糊关系, 则  $F \circ R = G$  称为模糊变换. 

**例 1.39** 设  $F = (1, 0.6, 0.2)$

$$R = \begin{bmatrix} 1 & 0.5 & 0 & 0 \\ 0.5 & 1 & 0.5 & 0 \\ 0 & 0.5 & 1 & 0.5 \end{bmatrix} \quad (1.136)$$

则

$$\begin{aligned} G = F \circ R &= \{1 \wedge 1 \vee 0.6 \wedge 0.5 \vee 0.2 \wedge 0, 1 \wedge 0.5 \vee 0.6 \wedge 1 \vee 0.2 \wedge 0.5, \\ &\quad 1 \wedge 0 \vee 0.6 \wedge 0.5 \vee 0.2 \wedge 1, 1 \wedge 0 \vee 0.6 \wedge 0 \vee 0.2 \wedge 0.5\} \\ &= \{1, 0.6, 0.5, 0.2\} \end{aligned}$$

### 1.10.3 FNN

### 1.10.4 DFNN

## 1.11 神经模糊系统和模糊神经网络

## 1.12 作业

### 探索

用遗传算法求  $f(x) = x \sin(10x) + 1.0$  的最大值, 其中  $x \in [-1, 2]$ .

### 探索

设有论域  $U = \{u_1, u_2, u_3, u_4, u_5\}$ , 并设  $F, G$  是  $U$  上的两个模糊集, 且有

$$F = 0.9/u_1 + 0.7/u_2 + 0.5/u_3 + 0.3/u_4$$

$$G = 0.6/u_3 + 0.8/u_4 + 1/u_5$$

请分别计算  $F \cap G, F \cup G, \neg F$ .

### 探索

设有如下两个模糊关系:

$$R_1 = \begin{bmatrix} 0.3 & 0.7 & 0.2 \\ 1 & 0 & 0.4 \\ 0 & 0.5 & 1 \end{bmatrix}$$
$$R_2 = \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \\ 0.9 & 0.1 \end{bmatrix}$$

请写出  $R_1$  与  $R_2$  的合成  $R_1 \circ R_2$ .

### 探索

基于剪枝技术的一字棋博弈系统

#### 1. 实验目的

理解和掌握博弈树的启发式搜索过程, 能够用某种程序语言建立一个简单的博弈系统.

#### 2. 实验环境

在微型计算机上, 任选一种编程语言.

#### 3. 实验要求

- (1) 规定棋盘大小为 5 行 5 列, 要求自行设计估价函数, 按极大极小搜索方法, 并采用 -剪枝技术.
- (2) 采用人机对弈方式, 一方走完一步后, 等待对方走步, 对弈过程每一时刻的棋局都在屏幕上显示出来.
- (3) 提交完整的软件系统和相关文档, 包括源程序和可执行程序.

表 1-21 Training and testing error on Auto-Mpg, Bank, Diabetes and Triazines.

		Web Link
Codes	ADP Algorithm	<a href="http://empiricaldataanalytics.org/downloads.html">http://empiricaldataanalytics.org/downloads.html</a>
	Pre-trained VGG-VD-16 convolutional neural network [46]	<a href="http://www.vlfeat.org/matconvnet/pretrained/">http://www.vlfeat.org/matconvnet/pretrained/</a>
Datasets	PIMA [47]	<a href="https://archive.ics.uci.edu/ml/datasets/pima+indian">https://archive.ics.uci.edu/ml/datasets/pima+indian</a>
	Banknote Authentication [37]	<a href="https://archive.ics.uci.edu/ml/datasets/banknote+au">https://archive.ics.uci.edu/ml/datasets/banknote+au</a>
	S1 [23]	<a href="http://cs.joensuu.fi/sipu/datasets/">http://cs.joensuu.fi/sipu/datasets/</a>
	S2 [23]	<a href="http://cs.joensuu.fi/sipu/datasets/">http://cs.joensuu.fi/sipu/datasets/</a>
	Cardiotocography [9]	<a href="https://archive.ics.uci.edu/ml/datasets/cardiotocog">https://archive.ics.uci.edu/ml/datasets/cardiotocog</a>
	Pen-Based Handwritten Digits Recognition [2]	<a href="https://archive.ics.uci.edu/ml/datasets/Pen-Based+R">https://archive.ics.uci.edu/ml/datasets/Pen-Based+R</a>
	Steel Plates Faults [14]	<a href="http://archive.ics.uci.edu/ml/datasets/steel+plates">http://archive.ics.uci.edu/ml/datasets/steel+plates</a>
	Multiple Features [30]	<a href="https://archive.ics.uci.edu/ml/datasets/Multiple+Fe">https://archive.ics.uci.edu/ml/datasets/Multiple+Fe</a>
	Occupancy Detection [16]	<a href="https://archive.ics.uci.edu/ml/datasets/Occupancy+D">https://archive.ics.uci.edu/ml/datasets/Occupancy+D</a>
	MAGIC Gamma Telescope [13]	<a href="https://archive.ics.uci.edu/ml/datasets/magic+gamma">https://archive.ics.uci.edu/ml/datasets/magic+gamma</a>
	Letter Recognition [24]	<a href="https://archive.ics.uci.edu/ml/datasets/letter+reco">https://archive.ics.uci.edu/ml/datasets/letter+reco</a>
	Singapore [26]	<a href="http://icn.bjtu.edu.cn/Visint/resources/Scenesig.asp">http://icn.bjtu.edu.cn/Visint/resources/Scenesig.asp</a>
	Caltech 101 [22]	<a href="http://www.vision.caltech.edu/Image_Datasets/Caltech">http://www.vision.caltech.edu/Image_Datasets/Caltech</a>
	MNIST [35]	<a href="http://yann.lecun.com/exdb/mnist/">http://yann.lecun.com/exdb/mnist/</a>

表 1-22 Details of the benchmark datasets for evaluation.

Abbreviation	Dataset	$N_{Ab}$	$N_{Sc}$	$N_{Cd}$
PI	PIMA [SmithEverhart1988-5692]	8	768	2
BA	Banknote Authentication [LohwegHoffmann2013-5693]	4	1372	2
S1	S1 [FrantiVirmajoki2006-5694]	2	5000	15
S2	S2 [FrantiVirmajoki2006-5694]	2	5000	15
CA	Cardiotocography [AyreseCamposBernardes2000-5695]	22	2126	3
PB	Pen-Based Handwritten Digits Recognition [AlimogluAlpaydin1996-5696]	16	10,992	10
ST	Steel Plates Faults [Buscema1998-5698]	27	1941	7
MU	Multiple Features [ARJain2000-5703]	649	2000	10
OD	Occupancy Detection [CandanedoFeldheim2016-5699]a	5	20,560	2
MA	MAGIC Gamma Telescope [BockChilingarian2004-5701]	10	19,020	2
LE	Letter Recognition [FreySlate1991-5702]	16	20,000	26

# 2

## 不确定性推理

### 不精确性推理方法

现实世界中的大多数问题是不精确、非完备的. 对于这些问题, 若采用前面所讨论的精确性推理方法显然是无法解决的. 为此, 人工智能需要研究不精确性的推理方法, 以满足客观问题的需求.





## 2.1 不确定性推理的含义

### 2.1.0.0.1 什么是不确定性推理

- ▶ 不确定性推理泛指除精确推理以外的其它各种推理问题. 包括不完备、不精确知识的推理, 模糊知识的推理, 非单调性推理等.
- ▶ 不确定性推理过程实际上是一种从不确定的初始证据出发, 通过运用不确定性知识, 最终推出具有一定不确定性但却又是合理或基本合理的结论的思维过程.

### 2.1.0.0.2 为什么要采用不确定性推理

- ▶ 所需知识不完备
- ▶ 不精确所需知识描述模糊
- ▶ 多种原因导致同一结论
- ▶ 问题的背景知识不足
- ▶ 解题方案不唯一

## 2.2 不确定性推理的基本问题

### 2.2.0.0.1 不确定性的表示 (1) 知识的不确定性的表示

- ▶ 考虑因素: 问题的描述能力  
推理中不确定性的计算
- ▶ 含义: 知识的确定性程度, 或动态强度
- ▶ 表示: 用概率,  $[0,1]$ , 0 接近于假, 1 接近于真. 用可信度,  $[-1,1]$ , 大于 0 接近于真, 小于 0 接近于假

(2) 证据的非精确性表示 证据来源: 初始证据, 中间结论表示: 用概率或可信度

- ▶ 含义  
不确定的前提条件与不确定的事实匹配
- ▶ 问题  
前提是不确定的, 事实也是不确定的
- ▶ 方法  
设计一个计算相似程度的算法, 给出相似的限度
- ▶ 标志  
相似度落在规定限度内为匹配, 否则为不匹配
- ▶ 含义

知识的前提条件是多个证据的组合

#### ► 方法

最大最小方法, 如合取取最小、析取取最大

概率方法, 按概率

#### 4. 非精确性的更新

主要问题

① 如何用证据的不确定性去更新结论的不确定性

② 如何在推理中把初始证据的不确定性传递给最终结论

解决方法

对① 不同推理方法的解决方法不同

对② 不同推理方法的解决方法基本相同, 即把当前结论及其不确定性作为新的结论放入综合数据库, 依次传递, 直到得出最终结论

#### 5. 非精确性结论的合成

含义: 多个不同知识推出同一结论, 且不确定性程度不同

方法: 视不同推理方法而定

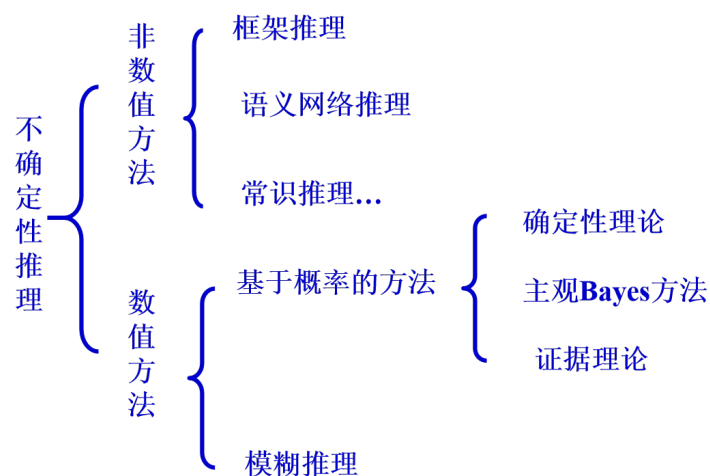


图 2-1

## 2.3 不确定性推理的概率论基础

在概率论中, 把试验中每一个可能出现的结果称为试验的一个样本点, 由全体样本点构成的集合称为样本空间.

表示: 通常用  $D$  表示样本空间,  $d$  表示样本点.

**例 2.1** 在掷币试验中, 若用  $d_1$  表示硬币的正面向上, 用  $d_2$  表示硬币的反面向上, 则该试验的样本空间为:  $D = \{d_1, d_2\}$

概念: 由样本点构成的集合称为随机事件

**例 2.2** 在掷币试验中, 若用  $A$  表示硬币正面向上这一事件, 则有  $A = \{d_1\}$


概率运算

- ▶ 并事件: 事件  $A$  与事件  $B$  至少有一个发生, 记为  $A \cup B$
- ▶ 交事件: 事件  $A$  与事件  $B$  同时发生, 记为  $A \cap B$
- ▶ 互逆事件
- ▶ 事件  $A$  与  $B$  之间满足 “ $A \cap B = \emptyset, A \cup B = D$ ”

### 定义 2.3 频率的概念

统计概率是通过某一事件出现的频率定义的. 频率:

$$f_n(A) = m/n \quad (2.1)$$

式中,  $A$  所讨论的事件,  $n$  是试验的总次数,  $m$  是实验中  $A$  发生的次数. 

### 定义 2.4 概率的统计定义

在同一组条件下所进行大量重复试验时, 如果事件  $A$  出现的频率总是在区间  $[0, 1]$  上的一个确定常数  $p$  附近摆动, 并且稳定于  $p$ , 则称  $p$  为事件  $A$  的统计概率. 即

$$P(A) = p \quad (2.2) $$

**例 2.5** 在掷币试验中, 当掷币次数足够多时有

$$f_n(\text{正面向上}) = 0.5 \quad (2.3)$$

则称正面向上的概率为 0.5, 即

$$P(\text{正面向上}) = 0.5 \quad (2.4)$$

**定义 2.6 统计概率的性质**

- (1) 对任一事件  $A$ , 有  $0 \leq P(A) \leq 1$ .
- (2) 必然事件  $D$  的概率  $P(D) = 1$ , 不可能事件的概率  $P(\emptyset) = 0$ .
- (3) 对任一事件  $A$ , 有  $P(\neg A) = 1 - P(A)$
- (4) 设事件  $A_1, A_2, \dots, A_k (k \leq n)$  是两两互不相容的事件, 即有  $A_i \cap A_j = \emptyset (i \neq j)$ , 则
- (5) 设  $A, B$  是两个事件, 则  $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

**定义 2.7 条件概率**

设  $A$  与  $B$  是两个随机事件,  $P(B) > 0$ , 则称:

$$P(A|B) = P(A \cap B) / P(B) \quad (2.5)$$

为在事件  $B$  发生的条件下事件  $A$  的条件概率.



**例 2.8** 设样本空间  $D$  是扑克牌中的 54 张牌, 即  $D = \{\text{红桃 A, 方块 A, 黑桃 A, 梅花 A, 红桃 2, 方块 2, } \dots, \text{小王, 大王}\}$ , 且有以下两个事件  $A = \{\text{取花脸牌}\}$ ,  $B = \{\text{取红桃牌}\}$ , 求在事件  $B$  发生的条件下事件  $A$  发生的概率  $P(A|B)$ .

**解:** 由于事件  $B$  已经发生, 因此以下事件 {取到红桃 A; 取到红桃 2; 取到红桃 3;  $\dots$ ; 取到红桃  $K$ } 中必有一个出现.

而对事件  $A$ , 在事件  $B$  发生的前提下, 只有以下事件 {取到红桃  $J$ ; 取到红桃  $Q$ ; 取到红桃  $K$  中的一个} 发生时事件  $A$  才能发生.

因此, 在事件  $B$  发生的条件下事件  $A$  发生的概率是  $3/13$ .

**定理 2.9 全概率公式**

设事件  $A_1, A_2, \dots, A_n$  满足:

- (1) 任意两个事件都互不相容, 即当  $i \neq j$  时, 有  $A_i \cap A_j = \emptyset (i = 1, 2, \dots, n_j = 1, 2, \dots, n)$ ;
- (2)  $P(A_i) > 0 (i = 1, 2, \dots, n)$ ;
- (3)  $D = \bigcup_{i=1}^n A_i$ , 则对任何事件  $B$  由下式成立:

$$P(B) = \sum_{i=1}^n P(A_i) \times P(B|A_i) \quad (2.6)$$


该公式称为全概率公式, 它提供了一种计算  $P(B)$  的方法.



### 定理 2.10 Bayes 定理

设事件  $A_1, A_2, \dots, A_n$  满足定理 6.1 规定的条件, 则对任何事件  $B$  有下式成立:

$$P(A_i|B) = \frac{P(A_i) \times P(B|A_i)}{\sum_{j=1}^n P(A_j) \times P(B|A_j)} \quad i = 1, 2, \dots, n \quad (2.7)$$

该定理称为 Bayes 定理, 上式称为 Bayes 公式. 其中  $P(A_i)$  是事件  $A_i$  的先验概率,  $P(B|A_i)$  是在事件  $A_i$  发生条件下事件  $B$  的条件概率;  $P(A_i|B)$  是在事件  $B$  发生条件下事件  $A_i$  的条件概率. 

如果把全概率公式代入 Bayes 公式, 则有:

$$P(A_i|B) = \frac{P(A) \times P(B|A_i)}{P(B)}, i = 1, 2, \dots, n \quad (2.8)$$

即

$$P(A_i|B) \times P(B) = P(B|A_i) \times P(A_i) \quad i = 1, 2, \dots, n \quad (2.9)$$

这是 Bayes 公式的另一种形式.

Bayes 定理给出了用逆概率  $P(B|A_i)$  求原概率  $P(A_i|B)$  的方法.

#### 2.3.1 可信度的概念

可信度是指人们根据以往经验对某个事物或现象为真的程度的一个判断, 或者说是人们对某个事物或现象为真的相信程度.

例如, 沈强昨天没来上课, 理由是头疼. 就此理由, 只有以下两种可能: 一是真的头疼了, 理由为真; 二是没有头疼, 理由为假. 但就听话人而言, 因不能确切知道, 就只能某种程度上相信, 即可信度.

可信度具有一定的主观性, 较难把握. 但对某一特定领域, 让该领域专家给出可信度还是可行的.

#### 2.3.2 CF 模型


##### 1. 知识不确定性的表示

表示形式:

在 C-F 模型中, 知识是用产生式规则表示的, 其一般形式为:

$$\text{IF } E \text{ THEN } H \text{ (CF(H, E))},$$

其中,  $E$  是知识的前提条件;  $H$  是知识的结论;  $CF(H, E)$  是知识的可信度.

 **注 2.11.** (1)  $E$  可以是单一条件, 也可以是复合条件. 例如:

$$E = (E_1 \text{ OR } E_2) \text{ AND } E_3 \text{ AND } E_4$$

(2)  $H$  可以是单一结论, 也可以是多个结论

(3)  $CF$  是知识的静态强度,  $CF(H, E)$  的取值为  $[-1, 1]$ , 表示当  $E$  为真时, 证据对  $H$  的支持程度, 其值越大, 支持程度越大.

**例 2.12** IF 发烧 AND 流鼻涕 THEN 感冒 (0.8)

表示当某人确实有“发烧”及“流鼻涕”症状时, 则有 80% 的把握是患了感冒.

### 2.3.3 可信度的定义与性质

**2.3.3.0.1 可信度的定义** 在  $CF$  模型中, 把  $CF(H, E)$  定义为

$$CF(H, E) = MB(H, E) - MD(H, E)$$

式中  $MB$  称为信任增长度,  $MB(H, E)$  定义为

$$MB(H, E) = \begin{cases} 1, & \text{若 } P(H) = 1 \\ \frac{\max\{P(H|E), P(H)\} - P(H)}{1 - P(H)}, & \text{否则} \end{cases} \quad (2.10)$$

$MD$  称为不信任增长度,  $MD(H, E)$  定义为

$$MD(H, E) = \begin{cases} 1, & \text{若 } P(H) = 0 \\ \frac{\min\{P(H|E), P(H)\} - P(H)}{-P(H)}, & \text{否则} \end{cases} \quad (2.11)$$

#### MB 和 MD 的关系

- ① 当  $MB(H, E) > 0$  时, 有  $P(H|E) > P(H)$ , 即  $E$  的出现增加了  $H$  的概率;
- ② 当  $MD(H, E) > 0$  时, 有  $P(H|E) < P(H)$ , 即  $E$  的出现降低了  $H$  的概率.

根据前面对  $CF(H, E)$  可信度、 $MB(H, E)$  信任增长度、 $MD(H, E)$  不信增长度的定义, 可得到  $CF(H, E)$  的计算公式:

$$CF(H, E) = \begin{cases} MB(H, E) - 0 = \frac{P(H|E) - P(H)}{1 - P(H)} & \text{若 } P(H|E) > P(H) \\ 0 & \text{若 } P(H|E) = P(H) \\ 0 - MD(H, E) = -\frac{P(H) - P(H|E)}{P(H)} & \text{若 } P(H|E) < P(H) \end{cases} \quad (2.12)$$

分别解释  $CF(H, E) > 0, CF(H, E) = 0, CF(H, E) < 0$ .

### 可信度的性质

#### ► 互斥性

对同一证据, 它不可能既增加对  $H$  的信任程度, 又同时增加对  $H$  的不信任程度, 这说明  $MB$  与  $MD$  是互斥的. 即有如下互斥性:

- 当  $MB(H, E) > 0, MD(H, E) = 0$
- 当  $MD(H, E) > 0, MB(H, E) = 0$

#### ► 值域

#### ► 典型值

当  $CF(H, E) = 1$  时, 有  $P(H|E) = 1$ , 它说明由于  $E$  所对应证据的出现使  $H$  为真. 此时,  $MB(H, E) = 1, MD(H, E) = 0$ .

当  $CF(H, E) = -1$  时, 有  $P(H|E) = 0$ , 说明由于  $E$  所对应证据的出现使  $H$  为假. 此时,  $MB(H, E) = 0, MD(H, E) = 1$ .

当  $CF(H, E) = 0$  时, 有  $MB(H, E) = 0, MD(H, E) = 0$ . 前者说明  $E$  所对应证据的出现不证实  $H$ ; 后者说明  $E$  所对应证据的出现不否认  $H$ .

#### ► 典型值对 $H$ 的信任增长长度等于对非 $H$ 的不信任增长长度

根据  $MB$ 、 $MD$  的定义及概率的性质有:

$$MD(\neg H, E) = \frac{P(\neg H|E) - P(\neg H)}{-P(\neg H)} = \frac{(1 - P(H|E)) - (1 - P(H))}{-(1 - P(H))} \quad (2.13)$$

$$= \frac{-P(H|E) + P(H)}{-(1 - P(H))} = \frac{-(P(H|E) - P(H))}{-(1 - P(H))} \quad (2.14)$$

$$= \frac{P(H|E) - P(H)}{1 - P(H)} = MB(H, E) \text{信任增长长度} \quad (2.15)$$

再根据  $CF$  的定义和  $MB$ 、 $MD$  的互斥性有

$$\begin{aligned} CF(H, E) + CF(\neg H, E) &= (MB(H, E) - MD(H, E)) + (MB(\neg H, E) - MD(\neg H, E)) \\ &= (MB(H, E) - 0) + (0 - MD(\neg H, E)) \text{(由互斥性)} \\ &= MB(H, E) - MD(\neg H, E) = 0 \end{aligned}$$

它说明

- (1) 对  $H$  的信任增长长度等于对非  $H$  的不信任增长长度
- (2) 对  $H$  的可信度与非  $H$  的可信度之和等于 0
- (3) 可信度不是概率, 不满足

$$P(H) + P(\neg H) = 1, 0 \leq P(H), P(\neg H) \leq 1.$$

(5) 对同一前提  $E$ , 若支持若干个不同的结论  $H_i (i = 1, 2, \dots, n)$ , 则因此, 如果发现专家给出的知识有如下情况

$$CF(H_1, E) = 0.7, CF(H_2, E) = 0.4$$

则因  $0.7 + 0.4 = 1.1 > 1$  为非法, 应进行调整或规范化.

## 2.4 证据理论

### 2.4.1 证据不确定性的表示

**2.4.1.0.1 不确定性的表示** 证据的不确定性也是用可信度来表示的, 其取值范围也为  $[-1, 1]$

- ▶ 若  $E$  为初始证据, 其值由用户给出.
- ▶ 若  $E$  为中间结论, 其值可通过计算得到.

**2.4.1.0.2 不确定性的含义** 对  $E$ , 其可信度  $CF(E)$  的含义如下:

- ▶  $CF(E) = 1$ , 证据  $E$  肯定它为真
- ▶  $CF(E) = -1$ , 证据  $E$  肯定它为假
- ▶  $CF(E) = 0$ , 对证据  $E$  一无所知
- ▶  $0 < CF(E) < 1$ , 证据  $E$  以  $CF(E)$  程度为真
- ▶  $-1 < CF(E) < 0$ , 证据  $E$  以  $CF(E)$  程度为假

### 2.4.1.0.3 否定证据不确定性的计算

$$CF(\neg E) = -CF(E) \quad (2.16)$$

**2.4.1.0.4 组合证据不确定性的计算** 对证据的组合形式可分为“合取”与“析取”两种基本情况.

- ▶ 合取

当组合证据是多个单一证据的组合时, 即  $E = E_1 \text{ AND } E_2 \text{ AND } \dots \text{ AND } E_n$  时, 若已知  $CF(E_1), CF(E_2), \dots, CF(E_n)$ , 则

$$CF(E) = \min\{CF(E_1), CF(E_2), \dots, CF(E_n)\} \quad (2.17)$$



► 析取

当组合证据是多个单一证据的析取时, 即  $E = E_1 \text{ OR } E_2 \text{ OR } \cdots \text{ OR } E_n$  时, 若已知  $CF(E_1), CF(E_2), \cdots, CF(E_n)$ , 则

$$CF(E) = \max\{CF(E_1), CF(E_2), \cdots, CF(E_n)\} \quad (2.18)$$

### 2.4.2 不确定性的更新

CF 模型中的不确定性推理实际上是从不确定的初始证据出发, 不断运用相关的不确定性知识, 逐步推出最终结论和该结论可信度的过程. 而每一次运用不确定性知识, 都需要由证据的不确定性和知识的不确定性去计算结论的不确定性. 不确定性的更新公式

$$CF(H) = CF(H, E) \times \max\{0, CF(E)\} \quad (2.19)$$

若  $CF(E) < 0$ , 则

$$CF(H) = 0 \quad (2.20)$$

即该模型没考虑  $E$  为假对  $H$  的影响. 若  $CF(E) = 1$ , 则

$$CF(H) = CF(H, E) \quad (2.21)$$

即规则强度  $CF(H, E)$  实际上是在  $E$  为真时,  $H$  的可信度.

### 2.4.3 结论不确定性的合成

当有多条知识支持同一个结论, 且这些知识的前提相互独立, 结论的可信度又不相同时, 可利用不确定性的合成算法求出结论的综合可信度. 设有知识:

$$\text{IF } E_1 \text{ THEN } H(CF(H, E_1)) \quad (2.22)$$

$$\text{IF } E_2 \text{ THEN } H(CF(H, E_2)) \quad (2.23)$$

则结论  $H$  的综合可信度可分以下两步计算:

(1) 分别对每条知识求出其  $CF(H)$ . 即

$$CF_1(H) = CF(H, E_1) \times \max\{0, CF(E_1)\} \quad (2.24)$$

$$CF_2(H) = CF(H, E_2) \times \max\{0, CF(E_2)\} \quad (2.25)$$

(2) 用如下公式求  $E_1$  与  $E_2$  对  $H$  的综合可信度

$$CF(H) = \begin{cases} CF_1(H) + CF_2(H) - CF_1(H) \times CF_2(H) & \text{若 } CF_1(H) \geq 0 \\ & \text{且 } CF_2(H) \geq 0 \\ CF_1(H) + CF_2(H) + CF_1(H) \times CF_2(H) & \text{若 } CF_1(H) < 0 \\ CF_1(H) + CF_2(H) & \text{若 } CF(H) < 0 \\ \frac{CF_1(H) + CF_2(H)}{1 - \min\{CF_1(H), |CF_2(H)|\}} & \text{且 } CF_2(H) \vec{F} \end{cases} \quad (2.26)$$

**例 2.13** 设有如下一组知识:

$r_1$ : IF  $E_1$  THEN  $H(0.9)$

$r_2$ : IF  $E_2$  THEN  $H(0.6)$

$r_3$ : IF  $E_3$  THEN  $H(-0.5)$

$r_4$ : IF  $E_4$  AND ( $E_5$  OR  $E_6$ ) THEN  $E_1(0.8)$

已知:  $CF(E_2) = 0.8, CF(E_3) = 0.6, CF(E_4) = 0.5, CF(E_5) = 0.6, CF(E_6) = 0.8$ , 求:  $CF(H) = ?$

**解:** 由  $r_4$  得到:

$$\begin{aligned} CF(E_1) &= 0.8 \times \max\{0, CF(E_4 \text{ AND } (E_5 \text{ OR } E_6))\} \\ &= 0.8 \times \max\{0, \min\{CF(E_4), CF(E_5 \text{ OR } E_6)\}\} \\ &= 0.8 \times \max\{0, \min\{CF(E_4), \max\{CF(E_5), CF(E_6)\}\}\} \\ &= 0.8 \times \max\{0, \min\{CF(E_4), \max\{0.6, 0.8\}\}\} \\ &= 0.8 \times \max\{0, \min\{0.5, 0.8\}\} \\ &= 0.8 \times \max\{0, 0.5\} = 0.4. \end{aligned}$$

由  $r_1$  得到:

$$CF_1(H) = CF(H, E_1) \times \max\{0, CF(E_1)\} = 0.9 \times \max\{0, 0.4\} = 0.36 \quad (2.27)$$

由  $r_2$  得到:

$$CF_2(H) = CF(H, E_2) \times \max\{0, CF(E_2)\} = 0.6 \times \max\{0, 0.8\} = 0.48 \quad (2.28)$$

由  $r_3$  得到:

$$CF_3(H) = CF(H, E_3) \times \max\{0, CF(E_3)\} = -0.5 \times \max\{0, 0.6\} = -0.3 \quad (2.29)$$

根据结论不精确性的合成算法,  $CF_1(H)$  和  $CF_2(H)$  同号, 有:

$$\begin{aligned} CF_{1,2}(H) &= CF_1(H) + CF_2(H) - CF_1(H) \times CF_2(H) \\ &= 0.36 + 0.48 - 0.36 \times 0.48 \\ &= 0.84 - 0.17 = 0.67. \end{aligned} \quad (2.30)$$

$CF_{1,2}(H)$  和  $CF_3(H)$  异号, 有:

$$\begin{aligned} CF_{1,2,3}(H) &= \frac{CF_{1,2}(H) + CF_3(H)}{1 - \min \{ |CF_{1,2}(H)|, |CF_3(H)| \}} \\ &= \frac{0.67 - 0.3}{1 - \min \{ 0.67, 0.3 \}} = \frac{0.37}{0.7} \\ &= 0.53. \end{aligned} \quad (2.31)$$

## 2.5 主观 Bayes 方法

在主观 Bayes 方法中, 知识是用产生式表示的, 其形式为:

$$\text{IF } E \text{ THEN (LS, LN) } H \quad (2.32)$$

其中, (LS, LN) 用来表示该知识的知识强度, LS(充分性度量) 和 LN(必要性度量) 的表示形式分别为:

$$\begin{aligned} LS &= \frac{P(E|H)}{P(E|\neg H)} \\ LN &= \frac{P(\neg E|H)}{P(\neg E|\neg H)} = \frac{1 - P(E|H)}{1 - P(E|\neg H)} \end{aligned} \quad (2.33)$$

下面进一步讨论 LS 和 LN 的含义. 由本章第2.2节的 Bayes 公式可知:

$$\begin{aligned} P(H|E) &= \frac{P(E|H) \times P(H)}{P(E)} \\ P(\neg H|E) &= \frac{P(E|\neg H) \times P(\neg H)}{P(E)} \end{aligned} \quad (2.34)$$

两式相除得:

$$\frac{P(H|E)}{P(\neg H|E)} = \frac{P(E|H)}{P(E|\neg H)} \times \frac{P(H)}{P(\neg H)} \quad (2.35)$$

为讨论方便, 下面引入概念

**定义 2.14 几率函数**

$$O(X) = \frac{P(X)}{1 - P(X)} P(X) = \frac{P(X)}{P(\neg X)} \quad (2.36)$$

可见,  $X$  的几率等于  $X$  出现的概率与  $X$  不出现的概率之比,  $P(X)$  与  $O(X)$  的变化一致, 且有:

$$P(X) = 0, O(X) = 0; \quad (2.37)$$

$$P(X) = 1, O(X) = +\infty. \quad (2.38)$$

即把取值为  $[0, 1]$  的  $P(X)$  放大为取值为  $[0, +\infty]$  的  $O(X)$

把(2.34)式代入(2.37)式有:

$$O(H|E) = \frac{P(E|H)}{P(E|\neg H)} \times O(H) \quad (2.39)$$

再把 LS 代入此式, 可得:

$$Q(H|E) = LS \times QH \quad (2.40)$$

同理可得到关于 LN 的公式:

$$\frac{P(H|\neg E)}{P(\neg H|\neg E)} = \frac{P(E|H)}{P(\neg E|\neg H)} \times \frac{P(H)}{P(\neg H)} \quad (2.41)$$

$$Q(H|\neg E) = LN \times QH \quad (2.42)$$

式(2.41)和(2.42)就是修改的 Bayes 公式.

## 2.6 证据理论

证据理论是由德普斯特 (A. P. Dempster) 首先提出, 并有沙佛 (G. Shafer) 进一步发展起来的用于处理不确定性的一种理论, 也称 DS (Dempster-Shafer) 理论. 它将概率论中的单点赋值扩展为集合赋值, 可以处理由“不知道”所引起的不确定性, 比主观 Bayes 方法有着更大的灵活性.

## 2.7 DS 理论的形式描述

在 DS 理论中, 可以分别用信任函数、似然函数及类概率函数来描述知识的精确信任度、不可驳斥信任度及估计信任度.

DS 理论处理的是集合上的不确定性问题, 为此需要先建立命题与集合之间的一一对应关系, 以把命题的不确定性转化为集合的不确定性.

设  $\Omega$  为样本空间, 且  $\Omega$  中的每个元素都相互独立, 则由  $\Omega$  的所有子集构成的幂集记为  $2^\Omega$ .

当  $\Omega$  中的元素个数为  $N$  时, 则其幂集  $2^\Omega$  的元素个数为  $2^N$ , 且其中的每一个元素都对应于一个关于  $x$  取值情况的命题.

**例 2.15** 设  $\Omega = \{\text{红}, \text{黄}, \text{白}\}$ , 求  $\Omega$  的幂集  $2^\Omega$ .

**解:**  $\Omega$  的幂集可包括如下子集:

$$\begin{aligned} A_0 &= \emptyset, A_1 = \{\text{红}\}, A_2 = \{\text{黄}\}, A_3 = \{\text{白}\}, \\ A_4 &= \{\text{红}, \text{黄}\}, A_5 = \{\text{红}, \text{白}\}, A_6 = \{\text{黄}, \text{白}\}, A_7 = \{\text{红}, \text{黄}, \text{白}\} \end{aligned}$$

其中,  $\emptyset$  表示空集, 空集也可表示为  $\{\}$ . 上述子集的个数正好是  $2^3 = 8$ .

**例 2.16** 设函数  $m : 2^\Omega \rightarrow [0, 1]$ , 且满足

$$\begin{aligned} m(\Phi) &= 0 \\ \sum_{A \subseteq \Omega} m(A) &= 1 \end{aligned} \tag{2.43}$$

则称  $m$  是  $2^\Omega$  上的概率分配函数,  $m(A)$  称为  $A$  的基本概率数.

对例 6.4 若定义  $2^\Omega$  上的一个基本函数  $m$ :

$$m(\{\text{红}\}, \{\text{黄}\}, \{\text{白}\}, \{\text{红}, \text{黄}\}, \{\text{红}, \text{白}\}, \{\text{黄}, \text{白}\}, \{\text{红}, \text{黄}, \text{白}\}) = (0, 0.3, 0, 0.1, 0.2, 0.2, 0, 0.2) \tag{2.44}$$

其中,  $(0, 0.3, 0, 0.1, 0.2, 0.2, 0, 0.2)$  分别是幂集  $2^\Omega$  中各个子集的基本概率数. 显然  $m$  满足概率分配函数的定义.

对概率分配函数的说明

(1) 概率分配函数的作用是把  $\Omega$  的任一子集映射为  $[0, 1]$  上的一个数  $m(A)$

- ▶ 当  $A \subset \Omega$ , 且  $A$  由单个元素组成时, 则  $m(A)$  表示对  $A$  的精确信任度;
- ▶ 当  $A \subset \Omega$ ,  $A \neq \Omega$ , 且  $A$  由多个元素组成时,  $m(A)$  也表示对  $A$  的精确信任度, 但却不知道这部分信任度该分给  $A$  中哪些元素;
- ▶ 当  $A \subseteq \Omega$  时, 则  $m(A)$  也表示不知道该如何分配的部分.

**例 2.17** 对上例所给出的有限集  $\Omega$  及基本函数  $m$ , 当

- ▶  $A = \{\text{红}\}$  时, 有  $m(A) = 0.3$ , 它表示对命题“ $x$  是红色”的精确信任度为 0.3.
- ▶  $B = \{\text{红}, \text{黄}\}$  时, 有  $m(B) = 0.2$ , 它表示对命题“ $x$  或者是红色, 或者是黄色”的精确信任度为 0.2, 却不知道该怎么把这 0.2 分给  $\{\text{红}\}$  还是分给  $\{\text{黄}\}$ .
- ▶  $C = \Omega = \{\text{红}, \text{黄}, \text{白}\}$  时, 有  $m(\Omega) = 0.2$ , 表示不知道该对这 0.2 如何分配, 但知道它不属于红, 就一定属于  $\{\text{黄}\}$  或  $\{\text{白}\}$ .

(2) 概率分配函数不是概率

**例 2.18** 在例 6.5 中,  $m$  符合概率分配函数的定义, 但

$$m(\{\text{红}\}) + m(\{\text{黄}\}) + m(\{\text{白}\}) = 0.3 + 0 + 0.1 = 0.4 < 1. \quad (2.45)$$

因此  $m$  不是概率, 因为概率  $P$  要求:  $P(\{\text{红}\}) + P(\text{黄}) + P(\text{白}) = 1$ .

### 定义 2.19 信任函数

信任函数  $Bel : 2^\Omega \rightarrow [0, 1]$  为

$$Bel(A) = \sum_{B \subseteq A} m(B) \quad (2.46)$$

其中,  $2^\Omega$  是  $\Omega$  的幂集.  $Bel$  又称为下限函数,  $Bel(A)$  表示对  $A$  的总的信任度. ♣

**例 2.20** 对例 6.5 有

$$Bel(\{\text{红}\}) = 0.3 \quad (2.47)$$

$$Bel(\{\text{红}, \text{白}\}) = m(\{\text{红}\}) + m(\{\text{白}\}) + m(\{\text{红}, \text{白}\}) = 0.3 + 0.1 + 0.2 = 0.6. \quad (2.48)$$

根据定义还可以得到:

$$\begin{aligned} Bel(\Phi) &= m(\Phi) = 0 \\ Bel(\Omega) &= \sum_{B \subseteq \Omega} m(B) = 1 \end{aligned} \quad (2.49)$$

**例 2.21** 对例 6.5 有

$$Bel(\emptyset) = m(\emptyset) = 0 \quad (2.50)$$

$$\begin{aligned} Bel(\{\text{红, 黄, 白}\}) &= m(\emptyset) + m(\{\text{红}\}) + m(\{\text{黄}\}) + m(\{\text{白}\}) + m(\{\text{红, 黄}\}) \\ &\quad + m(\{\text{红, 白}\}) + m(\{\text{黄, 白}\}) + m(\{\text{红, 黄, 白}\}) \\ &= 0 + 0.3 + 0 + 0.1 + 0.2 + 0.2 + 0 + 0.2 = 1 \end{aligned} \quad (2.51)$$

### 定义 2.22 似然函数

似然函数  $P_l : 2^\Omega \rightarrow [0, 1]$  为

$$P_l(A) = 1 - Bel(\neg A), A \subseteq \Omega \quad (2.52)$$

其中,  $\neg A = \Omega - A$ .

似然函数又称为不可驳斥函数或上限函数. 由于  $Bel(\neg A)$  表示对  $\neg A$  的信任度, 即  $A$  为假的信任度, 因此,  $P_l(A)$  表示对  $A$  为非假的信任度.

**例 2.23** 以例 6.5 为例:

$$\begin{aligned} P_l(\{\text{红}\}) &= 1 - Bel(\neg\{\text{红}\}) \\ &= 1 - Bel(\{\text{黄, 白}\}) \\ &= 1 - (m(\{\text{黄}\}) + m(\{\text{白}\}) + m(\{\text{黄, 白}\})) = 1 - (0 + 0.1 + 0) = 0.9. \end{aligned} \quad (2.53)$$

这里的 0.9 是“红”为非假的信任度. 由于“红”为真的精确信任度为 0.3, 而剩下的  $0.9 - 0.3 = 0.6$ , 则是知道非假, 但却不能肯定为真的那部分.

**例 2.24** 再如:

$$P_l(\{\text{黄, 白}\}) = 1 - Bel(\neg\{\text{黄, 白}\}) = 1 - Bel(\{\text{红}\}) = 1 - 0.3 = 0.7 \quad (2.54)$$

这里的 0.7 的含义与上面分析类似.

似然函数的另外一种计算办法:

$$\sum_{\{\text{红}\} \cap B = \emptyset} m(B) = 0.3 + 0.2 + 0.2 + 0.2 = 0.9 \quad (2.55)$$

由于

$$\sum_{\{\text{黄}, \text{白}\} \cap B = \emptyset} m(B) = 0 + 0.1 + 0 + 0.2 + 0.2 + 0.2 = 0.7 \quad (2.56)$$

可见,  $P_l(\{\text{红}\}), P_l(\{\text{黄}, \text{白}\})$  亦可分别用下式计算:

$$P_l(\{\text{红}\}) = \sum_{\{\text{红}\} \cap B = \emptyset} m(B), \quad P_l(\{\text{黄}, \text{白}\}) = \sum_{\{\text{黄}, \text{白}\} \cap B = \emptyset} m(B). \quad (2.57)$$

如果把它推广到一般可得公式:

$$P_l(A) = \sum_{A \cap B \neq \emptyset} m(B) \quad (2.58)$$

其证明见教材.

信任函数和似然函数之间存在关系:

$$P_l(A) \geq Bel(A) \quad (2.59)$$

证明.

$$\begin{aligned} Bel(A) + Be(\mathcal{T}A) &= \sum_{B \in A} m(B) + \sum_{C \in \mathcal{T}A} m(C) \leq \sum_{E \subseteq \Omega} m(E) = 1 \\ P_l(A) - Bel(A) &= 1 - Bel(\neg A) - Bel(A) \\ &= 1 - (Bel(\neg A) + Bel(A)) \geq 0 \\ P_l(A) &\geq Bel(A) \end{aligned} \quad (2.60)$$

由于  $Bel(A)$  和  $P_l(A)$  分别表示  $A$  为真的信任度和  $A$  为非假的信任度, 因此, 可分别称  $Bel(A)$  和  $P_l(A)$  为对  $A$  信任程度的下限和上限, 记为:  $A[Bel(A), P_l(A)]$   $\square$

**例 2.25** 在前面的例子中

$$Bel(\{\}) = 0.3, P_l(\{\}) = 0.9 \quad (2.61)$$

即:  $\{\text{红}\} = [0.3, 0.9]$ . 它表示对  $\{\text{红}\}$  的精确信任度为 0.3, 不可驳斥部分为 0.9, 肯定不是  $\{\text{红}\}$  的为 0.1.

同理可以求得

- 黄  $[0, 0.4]$  白  $[0.1, 0.5]$
- 红, 黄  $[0.5, 0.9]$  红, 白  $[0.6, 1]$



► 黄, 白 [0.1, 0.7] 红, 黄, 白 [1, 1]

► [0, 0]

一些典型值的含义:  $A[0, 1]$ : 说明对  $A$  一无所知. 其中,  $Bel(A) = 0$ , 说明对  $A$  无信任; 再由  $P_l(A) = 1 - Bel(\neg A) = 1$ , 可知  $Bel(\neg A) = 0$ , 说明对  $\neg A$  也没有信任.

►  $A[0, 0]$ : 说明  $A$  为假. 即  $Bel(A) = 0, Bel(\neg A) = 1$ .

►  $A[1, 1]$ : 说明  $A$  为真. 即  $Bel(A) = 1, Bel(\neg A) = 0$ .

►  $A[0.6, 1]$ : 说明对  $A$  部分信任. 即  $Bel(A) = 0.6, Bel(\neg A) = 0$ .

►  $A[0, 0.4]$ : 说明对  $\neg A$  部分信任. 即  $Bel(A) = 0, Bel(\neg A) = 0.6$ .

►  $A[0.3, 0.9]$ : 说明对  $A$  和  $\neg A$  都有部分信任. 其中,  $Bel(A)=0.3$ , 说明对  $A$  为真有 0.3 的信任度;  $Bel(\neg A) = 1 - 0.9 = 0.1$ , 说明对  $A$  为假有 0.1 的信任度. 因此,  $A[0.3, 0.9]$  表示对  $A$  为真的信任度比  $A$  为假的信任度稍高一些.

当证据来源不同时, 可能会得到不同的概率分配函数. 例如, 对

$$\Omega = \{\text{红}, \text{黄}\} \quad (2.62)$$

假设从不同知识源得到的两个概率分配函数分别为:

$$m_1(\{\text{红}\}, \{\text{黄}\}, \{\text{红}, \text{黄}\}) = (0, 0.4, 0.5, 0.1) \quad (2.63)$$

$$m_2(\{\text{红}\}, \{\text{黄}\}, \{\text{红}, \text{黄}\}) = (0, 0.6, 0.2, 0.2) \quad (2.64)$$

可采用德普斯特提出的求正交和的方法来组合这些函数

### 定义 2.26 概率分配函数的正交和

设  $m_1$  和  $m_2$  是两个不同的概率分配函数, 则其正交和  $m = m_1 \oplus m_2$  满足

$$\begin{aligned} m(\Phi) &= 0 \\ m(A) &= K^{-1} \times \sum_{x \cap y = A} m_1(x) \times m_2(y) \end{aligned} \quad (2.65)$$

其中:

$$K = 1 - \sum_{x \cap y = \Phi} m_1(x) \times m_2(y) = \sum_{x \cap y \neq \Phi} m_1(x) \times m_2(y) \quad (2.66)$$

如果  $K \neq 0$ , 则正交和也是一个概率分配函数; 如果  $K = 0$ , 则不存在正交和  $m$ , 称  $m_1$  与  $m_2$  矛盾.



**例 2.27** 设  $\Omega = \{a, b\}$ , 且从不同知识源得到的概率分配函数分别为

$$m_1(\{a\}, \{b\}, \{a, b\}) = (0, 0.3, 0.5, 0.2) \quad (2.67)$$

$$m_2(\{a\}, \{b\}, \{a, b\}) = (0, 0.6, 0.3, 0.1) \quad (2.68)$$

求正交和  $m = m_1 \oplus m_2$ .

**解:** 先求  $K$

$$\begin{aligned} K &= 1 - \sum_{x \cap y = \Phi} m_1(x) \times m_2(y) \\ &= 1 - (m_1(\{a\}) \times m_2(\{b\}) + m_1(\{b\}) \times m_2(\{a\})) \\ &= 1 - (0.3 \times 0.3 + 0.5 \times 0.6) = 0.61 \end{aligned} \quad (2.69)$$

再求  $m(\{a\}, \{b\}, \{a, b\})$ , 由于

$$\begin{aligned} m(\{a\}) &= \frac{1}{0.61} \times \sum_{x \cap y = \{a\}} m_1(x) \times m_2(y) \\ &= \frac{1}{0.61} \times (m_1(\{a\}) \times m_2(\{a\}) + m_1(\{a\}) \times m_2(\{a, b\}) + m_1(\{a, b\}) \times m_2(\{a\})) \\ &= \frac{1}{0.61} \times (0.3 \times 0.6 + 0.3 \times 0.1 + 0.2 \times 0.6) = 0.54 \end{aligned} \quad (2.70)$$

同理可求得

$$m(\{b\}) = 0.43 \quad (2.71)$$

$$m(\{a, b\}) = 0.03 \quad (2.72)$$

故有

$$m(\{a\}, \{b\}, \{a, b\}) = \{0, 0.54, 0.43, 0.03\} \quad (2.73)$$

对于多个概率分配函数的组合, 方法类似.

### 2.7.1 证据理论的推理模型

$Bel(A)$  和  $P_l(A)$  分别表示命题  $A$  的信任度的下限和上限, 同时也可用来表示知识强度的下限和上限.

从信任函数和似然函数的定义看, 它们都是建立在概率分配函数之上的, 可见不同的概率分配函数将得到不同的推理模型.

下面就给出一个特殊的概率分配函数, 并在其上建立推理模型.

设  $\Omega = s_1, s_2, \dots, s_n$ ,  $m$  为定义在  $2^\Omega$  上的概率分配函数, 且  $m$  满足

$$\begin{aligned} (1) & m(\{s_i\}) \geq 0 \\ (2) & \sum_{i=1}^n m(\{s_i\}) \leq 1 \quad A \subset \Omega, |A| > 1 \text{ 或者 } |A| = 0 \Rightarrow m(A) = 0 \\ (3) & m(\Omega) = 1 - \sum_{i=1}^n m(\{s_i\}) \end{aligned} \quad (2.74)$$

其中,  $|A|$  表示命题  $A$  所对应的集合中的元素个数.

该概率分配函数的特殊性:

- ① 只有当子集中的元素个数为 1 时, 其概率分配数才有可能大于 0;
- ② 当子集中有多个或 0 个元素, 且不等于全集时, 其概率分配数均为 0;
- ③ 全集  $\Omega$  的概率分配数按 (3) 计算.

**例 2.28** 设  $\Omega = \{\text{红}, \text{黄}, \text{白}\}$ , 有如下概率分配函数

$$m(\{\text{红}\}, \{\text{黄}\}, \{\text{白}\}, \{\text{红}, \text{黄}, \text{白}\}) = (0, 0.6, 0.2, 0.1, 0.1) \quad (2.75)$$

其中:  $m(\{\text{红}\}, \{\text{黄}\}, \{\text{白}\}, \{\text{红}, \text{黄}, \text{白}\}) = 0$ , 可见,  $m$  符合上述概率分配函数的定义.

**例 2.29** 对任何命题  $A \subset \Omega$ , 其信任函数为

$$\begin{aligned} \text{Bel}(A) &= \sum_{s_i \in A} m(\{s_i\}) \\ \text{Bel}(\Omega) &= \sum_{B \subseteq \Omega} m(B) = \sum_{i=1}^n m(\{s_i\}) + m(\Omega) = 1 \end{aligned} \quad (2.76)$$

### 定义 2.30 其似然函数

对任何命题  $A \subset \Omega$ , 其似然函数为

$$\begin{aligned} P_l(A) &= 1 - \text{Bel}(\neg A) = 1 - \sum_{s_i \in \neg A} m(\{s_i\}) = 1 - \left[ \sum_{i=1}^n m(\{s_i\}) - \sum_{s_i \in A} m(\{s_i\}) \right] \\ &= 1 - [1 - m(\Omega) - \text{Bel}(A)] \\ &= m(\Omega) + \text{Bel}(A) \\ P_l(\Omega) &= 1 - \text{Bel}(\neg \Omega) = 1 - \text{Bel}(\Phi) = 1 \end{aligned} \quad (2.77)$$

可以看出, 对任意命题  $A \subset \Omega$  和  $B \subset \Omega$  均有:

$$P_l(A) - \text{Bel}(A) = P_l(B) - \text{Bel}(B) = m(\Omega) \quad (2.78)$$

它表示对  $A$ (或  $B$ ) 不知道的程度.

**例 2.31** 设  $\Omega = \{\text{红}, \text{黄}, \text{白}\}$ , 概率分配函数

$$m(\{\text{红}\}, \{\text{黄}\}, \{\text{红}, \text{黄}\}, \{\text{红}, \text{黄}, \text{白}\}) = (0, 0.6, 0.2, 0.1, 0.1) \quad (2.79)$$

$A = \{\text{红}, \text{黄}\}$ , 求  $m(\Omega)$ 、 $Bel(A)$  和  $P_l(A)$  的值.

解:

$$m(\Omega) = 1 - [m(\{\text{红}\}) + m(\{\text{黄}\}) + m(\{\text{白}\})] = 1 - (0.6 + 0.2 + 0.1) = 0.1 \quad (2.80)$$

$$Bel(\{\text{红}, \text{黄}\}) = m(\{\text{红}\}) + m(\{\text{黄}\}) = 0.6 + 0.2 = 0.8 \quad (2.81)$$

$$P_l(\{\text{红}, \text{黄}\}) = m(\Omega) + Bel(\{\text{红}, \text{黄}\}) = 0.1 + 0.8 = 0.9 \quad (2.82)$$

或

$$P_l(\{\text{红}, \text{黄}\}) = 1 - Bel(\neg\{\text{红}, \text{黄}\}) = 1 - Bel(\{\text{白}\}) = 1 - 0.1 = 0.9 \quad (2.83)$$

### 定义 2.32 基本概率分配函数的正交和

设  $m_1$  和  $m_2$  是  $2^\Omega$  上的基本概率分配函数, 它们的正交和定义为

$$m(\{s_i\}) = K^{-1} \times [m_1(s_i) \times m_2(s_i) + m_1(s_i) \times m_2(\Omega) + m_1(\Omega) \times m_2(s_i)] \quad (2.84)$$

**例 2.33** 设  $\Omega = \{\text{红}, \text{黄}, \text{白}\}$ , 概率分配函数

$$m(\{\text{红}\}, \{\text{黄}\}, \{\text{白}\}, \{\text{红}, \text{黄}, \text{白}\}) = (0, 0.6, 0.2, 0.1, 0.1) \quad (2.85)$$

$A = \{\text{红}, \text{黄}\}$ , 求  $m(\Omega)$ 、 $Bel(A)$  和  $P_l(A)$  的值.

解:

$$m(\Omega) = 1 - [m(\{\text{红}\}) + m(\{\text{黄}\}) + m(\{\text{白}\})] = 1 - (0.6 + 0.2 + 0.1) = 0.1 \quad (2.86)$$

$$Bel(\{\text{红}, \text{黄}\}) = m(\{\text{红}\}) + m(\{\text{黄}\}) = 0.6 + 0.2 = 0.8 \quad (2.87)$$

$$P_l(\{\text{红}, \text{黄}\}) = m(\Omega) + Bel(\{\text{红}, \text{黄}\}) = 0.1 + 0.8 = 0.9 \quad (2.88)$$

或

$$P_l(\{\text{红}, \text{黄}\}) = 1 - Bel(\neg\{\text{红}, \text{黄}\}) = 1 - Bel(\{\text{白}\}) = 1 - 0.1 = 0.9. \quad (2.89)$$

**定义 2.34 概率分配函数的正交和**

设  $m_1$  和  $m_2$  是  $2^\Omega$  上的基本概率分配函数, 它们的正交和定义为

$$m(\{s_i\}) = K^{-1} \times [m_1(s_i) \times m_2(s_i) + m_1(s_i) \times m_2(\Omega) + m_1(\Omega) \times m_2(s_i)] \quad (2.90)$$

其中,

$$K = m_1(\Omega) \times m_2(\Omega) + \sum_{i=1}^n [m_1(s_i) \times m_2(s_i) + m_1(s_i) \times m_2(\Omega) + m_1(\Omega) \times m_2(s_i)] \quad (2.91)$$

**定义 2.35 类概率函数**

设  $\Omega$  为有限域, 对任何命题  $A \subset \Omega$ , 命题  $A$  的类概率函数为

$$f(A) = \text{Bel}(A) + \frac{|A|}{|\Omega|} \times [P_l(A) - \text{Bel}(A)] \quad (2.92)$$

其中,  $|A|$  和  $|\Omega|$  分别是  $A$  及  $\Omega$  中元素的个数.

类概率函数  $f(A)$  的性质

$$(1) \sum_{i=1}^n f(\{S_i\}) = 1$$

$$\begin{aligned} \because f(\{s_i\}) &= \text{Bel}(\{s_i\}) + \frac{|\{s_i\}|}{|\Omega|} \times [P_l(\{s_i\}) - \text{Bel}(\{s_i\})] \\ &= m(\{s_i\}) + \frac{1}{n} \times m(\Omega) \quad i = 1, 2, \dots, n \\ \therefore \sum_{i=1}^n f(\{s_i\}) &= \sum_{i=1}^n \left[ m(s_i) + \frac{1}{n} \times m(\Omega) \right] \\ &= \sum_{i=1}^n m(\{s_i\}) + m(\Omega) = 1 \end{aligned} \quad (2.93)$$

$$(2) \text{对任何 } A \subset \Omega, \text{ 有 } \text{Bel}(A) \leq f(A) \leq P_l(A)$$

$$\begin{aligned} \because P_l(A) - P_l(B) &= m(\Omega), \quad \frac{|A|}{|\Omega|} \geq 0 \\ \therefore \text{Bel}(A) &\leq f(A) \\ \because \frac{|A|}{|\Omega|} &\leq 1, \text{ BP } f(A) \leq \text{Bel}(A) + \text{Pl}(A) - \text{Bel}(A) \\ \therefore f(A) &\leq P_l(A) \end{aligned} \quad (2.94)$$

$$(3) \text{对任何 } A \subseteq \Omega, \text{ 有 } f(\neg A) = 1 - f(A).$$

$$\therefore f(\neg A) = Bel(\neg A) + \frac{|\mathcal{A}|}{|P_l(\neg A) - Bel(\neg A)|} \quad (2.95)$$

$$\begin{aligned} Bel(\neg A) &= \sum_{\substack{s_i \in \neg A \\ s_i \in A}} m(\{s_i\}) \\ &= 1 - \sum_{\substack{s_i \in A \\ s_i \in d}} m(\{s_i\}) - m(\Omega) = 1 - Bel(A) - m(\Omega) \end{aligned} \quad (2.96)$$

$$|\neg A| = |\Omega| - |A| \quad (2.97)$$

$$|\neg A| = |\Omega| - |A| \quad (2.98)$$

$$P_l(\neg A) - Bel(\neg A) = m(\Omega) \quad (2.99)$$

$$\begin{aligned} \therefore f(\neg A) &= 1 - Bel(A) - m(\Omega) + \frac{|\Omega| - |A|}{|\Omega|} \times m(\Omega) \\ &= 1 - Bel(A) - m(\Omega) + m(\Omega) - \frac{|A|}{|\Omega|} \times m(\Omega) \\ &= 1 - \left[ Bel(A) + \frac{|A|}{|\Omega|} \times m(\Omega) \right] = 1 - f(A) \end{aligned} \quad (2.100)$$

推论

$$(1) f(\emptyset) = 0;$$

$$(2) f(\Omega) = 1;$$

$$(3) \text{ 对任何 } A \subseteq \Omega, \text{ 有 } 0 \leq f(A) \leq 1.$$

**例 2.36** 设  $\Omega = \{\text{红}, \text{黄}, \text{白}\}$ , 概率分配函数

$$m(\{\text{红}\}, \{\text{黄}\}, \{\text{白}\}, \{\text{红}, \text{黄}, \text{白}\}) = (0, 0.6, 0.2, 0.1, 0.1) \quad (2.101)$$

若  $A = \{\text{红}, \text{黄}\}$ , 求  $f(A)$  的值.

解:

$$\begin{aligned} f(A) &= Bel(A) + \frac{|A|}{|\Omega|} \times [P_l(A) - Bel(A)] \\ &= m(\{\underline{I}\}) + m(\{\#\}) + \frac{2}{3} \times m(\{\underline{L}, \#, \#\}) \\ &= 0.6 + 0.2 + \frac{2}{3} \times 0.1 = 0.87 \end{aligned} \quad (2.102)$$

表示形式:

$$\text{IF } E \text{ THEN } H = h_1, h_2, \dots, h_n \text{ CF} = \{c_1, c_2, \dots, c_n\} \quad (2.103)$$

其中:  $E$  为前提条件, 它既可以是简单条件, 也可以是用合取或析取词连接起来的复合条件;

$H$  是结论, 它用样本空间中的子集表示,  $h_1, h_2, \dots, h_n$  是该子集中的元素;

$CF$  是可信度因子, 用集合形式表示. 该集合中的元素  $c_1, c_2, \dots, c_n$  用来指出  $h_1, h_2, \dots, h_n$  的可信度,  $c_i$  与  $h_i$  一一对应.

并且  $c_i$  应满足如下条件:

$$\begin{aligned} c_i &\geq 0 & i = 1, 2, \dots, n \\ \sum_{i=1}^n c_i &\leq 1 \end{aligned} \quad (2.104)$$

**例 2.37** 设  $A$  是规则条件部分的命题,  $E'$  是外部输入的证据和已证实的命题, 在证据  $E'$  的条件下, 命题  $A$  与证据  $E'$  的匹配程度为

$$MD(A/E') = \begin{cases} 1 \\ 0 \end{cases} \quad (2.105)$$

### 定义 2.38 条件部分命题 $A$ 的确定性

条件部分命题  $A$  的确定性为

$$CER(A) = MD(A/E') \times f(A) \quad (2.106)$$

其中  $f(A)$  为类概率函数.

由于  $f(A) \in [0, 1]$ , 因此  $CER(A) \in [0, 1]$

当组合证据是多个证据的合取时:

$$E = E_1 \text{ AND } E_2 \text{ AND } \dots \text{ AND } E_n \quad (2.107)$$

则

$$CER(E) = \min\{CER(E_1), CER(E_2), \dots, CER(E_n)\} \quad (2.108)$$

当组合证据是多个证据的析取时:

$$E = E_1 \text{ OR } E_2 \text{ OR } \dots \text{ OR } E_n \quad (2.109)$$

则

$$CER(E) = \max\{CER(E_1), CER(E_2), \dots, CER(E_n)\} \quad (2.110)$$

设有知识 IF  $E$  THEN  $H = \{h_1, h_2, \dots, h_n\}$   $CF = \{c_1, c_2, \dots, c_n\}$ , 则求结论  $H$  的确定性  $CER(H)$  的方法如下:

(1) 求  $H$  的概率分配函数

$$\begin{aligned} m(\{h_1\}, \{h_2\}, \dots, \{h_n\}) &= (CER(E) \times c_1, CER(E) \times c_2, \dots, CER(E) \times c_n) \\ m(\Omega) &= 1 - \sum_{i=1}^n CER(E) \times c_i \end{aligned} \quad (2.111)$$

如果有两条或多条知识支持同一结论  $H$ , 例:

$$IF E THEN H = h_1, h_2, \dots, h_n \quad CF = c_{11}, c_{12}, \dots, c_{1n} \quad (2.112)$$

$$IF E THEN H = h_1, h_2, \dots, h_n \quad CF = c_{21}, c_{22}, \dots, c_{2n} \quad (2.113)$$

则按正交和求  $CER(H)$ , 即先求出:

$$m_1 = m(h_1, h_2, \dots, h_n) \quad (2.114)$$

$$m_2 = m(h_1, h_2, \dots, h_n) \quad (2.115)$$

然后再用公式  $m = m_1 \oplus m_2$  求  $m_1$  和  $m_2$  的正交和, 最后求得  $H$  的  $m$ .

(2) 求  $Bel(H)$ 、 $P_l(H)$  及  $f(H)$

$$\begin{aligned} Bel(H) &= \sum_{i=1}^n m(\{h_i\}) \\ P_l(H) &= 1 - Bel(\neg H) \\ f(H) &= Be(H) + \frac{|H|}{|\Omega|} \times [P_l(H) - Be(H)] = Be(H) + \frac{|H|}{|\Omega|} \times m(\Omega) \end{aligned} \quad (2.116)$$

(3) 求  $H$  的确定性  $CER(H)$

按公式  $CER(H) = MD(H/E') \times f(H)$  计算结论  $H$  确定性.

**例 2.39** 设有如下规则:

- $r_1$ : IF  $E_1$  AND  $E_2$  THEN  $A=a_1, a_2$   $CF=\{0.3, 0.5\}$
- $r_2$ : IF  $E_3$  AND ( $E_4$  OR  $E_5$ ) THEN  $B=b_1$   $CF=\{0.7\}$
- $r_3$ : IF  $A$  THEN  $H=\{h_1, h_2, h_3\}$   $CF=\{0.1, 0.5, 0.3\}$
- $r_4$ : IF  $B$  THEN  $H=\{h_1, h_2, h_3\}$   $CF=\{0.4, 0.2, 0.1\}$

已知用户对初始证据给出的确定性为:

$$CER(E_1)=0.8 \quad CER(E_2)=0.6 \quad CER(E_3)=0.9 \quad CER(E_4)=0.5 \quad CER(E_5)=0.7$$

并假定中的元素个数  $\Omega = 10$ , 求:  $CER(H) = ?$

**解:** 由给定知识形成的推理网络为:



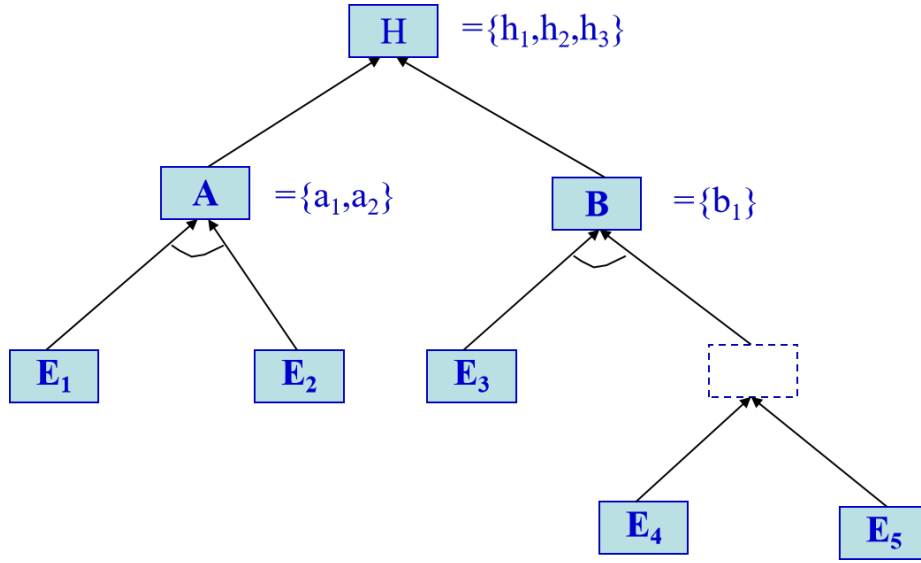


图 2-2

(1) 求 CER(A)

$$\text{CER}(E_1 \text{ AND } E_2) = \min\{\text{CER}(E_1), \text{CER}(E_2)\} = \min\{0.8, 0.6\} = 0.6$$

$$m(a_1, a_2) = \{0.6 \times 0.3, 0.6 \times 0.5\} = \{0.18, 0.3\}$$

$$\text{Bel}(A) = m(a_1) + m(a_2) = 0.18 + 0.3 = 0.48$$

$$P_l(A) = 1 - \text{Bel}(\neg A) = 1 - 0 = 1$$

$$f(A) = \text{Bel}(A) + |A|/|\Omega| * [P_l(A) - \text{Bel}(A)] = 0.48 + 2/10 * [1 - 0.48] = 0.584$$

$$\text{CER}(A) = MD(A/E') \times f(A) = 0.584$$

(2) 求 CER(B)

$$\text{CER}(E_3 \text{ AND } (E_4 \text{ OR } E_5)) = \min\{\text{CER}(E_3), \max\{\text{CER}(E_4), \text{CER}(E_5)\}\}$$

$$= \min\{0.9, \max\{0.5, 0.7\}\} = \min\{0.9, 0.7\} = 0.7$$

$$m(b_1) = 0.7 \times 0.7 = 0.49$$

$$\text{Bel}(B) = m(b_1) = 0.49$$

$$P_l(B) = 1 - \text{Bel}(\neg B) = 1 - 0 = 1$$

$$F(B) = \text{Bel}(B) + |B|/|\Omega| * [P_l(B) - \text{Bel}(B)]$$

$$= 0.49 + 1/10 * [1 - 0.49] = 0.541$$

$$\text{CER}(B) = MD(B/E') \times f(B) = 0.541$$

(3) 求  $\text{CER}(H)$ , 由  $r_3$  可得

$$\begin{aligned} m_1(h_1, h_2, h_3) &= \{\text{CER}(A) \times 0.1, \text{CER}(A) \times 0.5, \text{CER}(A) \times 0.3\} \\ &= \{0.584 \times 0.1, 0.584 \times 0.5, 0.584 \times 0.3\} = \{0.058, 0.292, 0.175\} \end{aligned}$$

$$m_1(\Omega) = 1 - [m_1(h_1) + m_1(h_2) + m_1(h_3)] = 1 - (0.058 + 0.292 + 0.175) = 0.475.$$

再由  $r_4$  可得

$$\begin{aligned} m_2(h_1, h_2, h_3) &= \text{CER}(B) \times 0.4, \text{CER}(B) \times 0.2, \text{CER}(B) \times 0.1 \\ &= \{0.541 \times 0.4, 0.541 \times 0.2, 0.541 \times 0.1\} = \{0.216, 0.108, 0.054\} \\ m_2(\Omega) &= 1 - [m_2(h_1) + m_2(h_2) + m_2(h_3)] \\ &= 1 - [0.216 + 0.108 + 0.054] = 0.622. \end{aligned}$$

求正交和  $m = m_1 \oplus m_2$

$$\begin{aligned} K &= m_1(\Omega) \times m_2(\Omega) + m_1(h_1) \times m_2(h_1) + m_1(h_1) \times m_2(\Omega) + m_1(\Omega) \times m_2(h_1) \\ &\quad + m_1(h_2) \times m_2(h_2) + m_1(h_2) \times m_2(\Omega) + m_1(\Omega) \times m_2(h_2) \\ &\quad + m_1(h_3) \times m_2(h_3) + m_1(h_3) \times m_2(\Omega) + m_1(\Omega) \times m_2(h_3) \\ &= 0.475 \times 0.622 + 0.058 \times 0.216 + 0.058 \times 0.622 + 0.475 \times 0.216 \\ &\quad + 0.292 \times 0.108 + 0.292 \times 0.622 + 0.475 \times 0.108 \\ &\quad + 0.175 \times 0.054 + 0.175 \times 0.622 + 0.475 \times 0.054 = 0.855. \end{aligned}$$

$$\begin{aligned} m(h_1) &= \frac{1}{K} \times [m_1(\{h_1\}) \times m_2(\{h_1\}) + m_1(\{h_1\}) \times m_2(\Omega) + m_1(\Omega) \times m_2(\{h_1\})] \\ &= \frac{1}{0.855} \times [0.058 \times 0.216 + 0.058 \times 0.622 + 0.475 \times 0.216] \end{aligned}$$

同理可得:

$$\begin{aligned} m(h_2) &= \frac{1}{K} \times [m_1(\{h_2\}) \times m_2(\{h_2\}) + m_1(\{h_2\}) \times m_2(\Omega) + m_1(\Omega) \times m_2(\{h_2\})] \\ &= \frac{1}{0.855} \times [0.292 \times 0.108 + 0.292 \times 0.622 + 0.475 \times 0.108] \\ &= 0.309 \\ m(h_3) &= \frac{1}{K} \times [m_1(\{h_3\}) \times m_2(\{h_3\}) + m_1(\{h_3\}) \times m_2(\Omega) + m_1(\Omega) \times m_2(\{h_3\})] \\ &= \frac{1}{0.855} \times [0.175 \times 0.054 + 0.175 \times 0.622 + 0.475 \times 0.054] \\ &= 0.168 \end{aligned}$$

$$m(\Omega) = 1 - [m(h_1) + m(h_2) + m(h_3)] = 1 - (0.178 + 0.309 + 0.168) = 0.345.$$

再根据  $m$  可得

$$Bel(H) = m(h_1) + m(h_2) + m(h_3) = 0.178 + 0.309 + 0.168 = 0.655$$

$$P_l(H) = m(\Omega) + Bel(H) = 0.345 + 0.655 = 1$$

$$\begin{aligned} f(H) &= Bel(H) + \frac{|H|}{|\Omega|} \times [P_l(H) - Bel(H)] = 0.655 + \frac{3}{10} \times [1 - 0.655] \\ &= 0.759 \end{aligned}$$

$$CER(H) = MD(H/E') \times f(H) = 0.759.$$

优点: 能处理由“不知道”所引起的非精确性; 并且由于辨别框 (样本空间) 的子集可以是多个元素的集合, 这样更有利于领域专家在不同层次上进行知识表示.

缺点: 要求辨别框中的元素满足互斥条件, 这在实际系统中不易实现; 并且, 需要给出的概率分配数太多, 计算比较复杂.

## 2.8 模糊推理

用自然语言中的词或句子表示的变量

**例 2.40** 变量“年龄”在普通集合中为数字变量  $u = [0, 150]$ , 而在模糊集和中可使用语言变量, 该语言变量的取值可以是年轻、很年轻、不很年轻、老、很老、不很老等. 这些值可看作是论域  $U = [0, 150]$  上模糊集的集合名.

**2.8.0.0.1 模糊谓词** 设  $x \in U$ ,  $F$  为模糊谓词, 即  $U$  中的一个模糊关系, 则模糊命题可表示为

$$x \text{ is } F \quad (2.117)$$

其中的模糊谓词  $F$  可以是 大、小、年轻、年老、冷、暖、长、短等.

**2.8.0.0.2 模糊量词** 模糊逻辑中使用的模糊量词, 如极少、很少、几个、少数、多数、大多数、几乎所有等. 这些模糊量词可以很方便地描述类似于下面的命题:

► 大多数成绩好的学生学习都很刻苦.

► 很少有成绩好的学生特别贪玩.

模糊概率、模糊可能性和模糊真值

设  $\lambda$  为模糊概率,  $\mu$  为模糊可能性,  $\tau$  为模糊真值, 则对命题还可以附加概率限定、可能性限定和真值限定:

$$(x \text{ is } F) \text{ is } \lambda(x \text{ is } F) \text{ is } \prod (x \text{ is } F) \text{ is } \tau \quad (2.118)$$

其中,  $\lambda$  可以是“或许”、“必须”等; 可以是“非常可能”、“很不可能”等; 可以是“非常真”、“有些假”等.

**例 2.41** “常青很可能是年轻的”可表示为

$$(\text{Age}(\text{Chang qing}) \text{ is young}) \text{ is likely}$$

**2.8.0.0.3 模糊修饰语** 设  $m$  是模糊修饰语,  $x$  是变量,  $F$  谓模糊谓词, 则模糊命题可表示为  $x \text{ is } m_F$ , 模糊修饰语也称为程度词, 常用的程度词有“很”、“非常”、“有些”、“绝对”等.

模糊修饰语的表达主要通过以下四种运算实现:

① 求补表示否定, 如“不”、“非”等, 其隶属函数的表示为

$$\mu_{\neg F}(u) = 1 - \mu_F(u) \quad u \in [0, 1]$$

② 集中表示“很”、“非常”等, 其效果是减少隶属函数的值:

$$\mu_{\text{非常}F}(u) = \mu_F^2(u) \quad u \in [0, 1]$$

③ 扩张表示“有些”、“稍微”等, 其效果是增加隶属函数的值:

$$\mu_{\text{有些}F}(u) = \mu_F^{\frac{1}{2}}(u) \quad u \in [0, 1]$$

④ 加强对比表示“明确”、“确定”等, 其效果是增加 0.5 以上隶属函数的值, 减少 0.5 以下隶属函数的值:

$$\mu_{\text{确实}F}(u) = \begin{cases} 2\mu_F^2(u), & 0 \leq \mu_F(u) \leq 0.5 \\ 1 - 2(1 - \mu_F(u))^2, & 0.5 < \mu_F(u) \leq 1 \end{cases}$$

在以上 4 种运算中, 集中与扩张用的较多. 例如, 语言变量“真实性”取值“真”和“假”的隶属函数定义为:

$$\begin{aligned}\mu_{\text{真}}(u) &= u & u \in [0, 1] \\ \mu_{\text{假}}(u) &= 1 - u & u \in [0, 1]\end{aligned}$$

则“非常真”、“有些真”、“非常假”、“有些假”可定义为

$$\mu_{\text{非常真}}(u) = u^2, u \in [0, 1]; \quad \mu_{\text{非常假}}(u) = \mathbb{R} \quad (u) = (1 - u)^2, u \in [0, 1] \quad (2.119)$$

$$\mu_{\text{有些真}}(u) = u^{\frac{1}{2}}, u \in [0, 1]; \quad \mu_{\text{有些假}}(u) = \pm \lim_{\mathbb{R}} \quad (u) = (1 - u)^{\frac{1}{2}}, u \in [0, 1] \quad (2.120)$$

在扎德的推理模型中, 产生式规则的表示形式是

$$\text{IF } x \text{ is } F \text{ THEN } y \text{ is } G$$

其中:  $x$  和  $y$  是变量, 表示对象;  $F$  和  $G$  分别是论域  $U$  和  $V$  上的模糊集, 表示概念.

### 2.8.1 模糊概念的匹配

语义距离用于刻划两个模糊概念之间的差异. 这里主要讨论海明距离.

**离散论域:** 设  $U = u_1, u_2, \dots, u_n$  是一个离散有限论域,  $F$  和  $G$  分别是论域  $U$  上的两个模糊概念的模糊集, 则  $F$  和  $G$  的海明距离定义为

$$d(F, G) = \frac{1}{n} \times \sum_{i=1}^n |\mu_F(u_i) - \mu_G(u_i)| \quad (2.121)$$

**连续论域:** 如果论域  $U$  是实数域上的某个闭区间  $[a, b]$ , 则海明距离为

$$d(F, G) = \frac{1}{b-a} \int_a^b |\mu_F(u) - \mu_G(u)| d(u) \quad (2.122)$$

**例 2.42** 设论域  $U = \{-10, 0, 10, 20, 30\}$  表示温度, 模糊集

$$F = 0.8/-10 + 0.5/0 + 0.1/10 \quad (2.123)$$

$$G = 0.9/-10 + 0.6/0 + 0.2/10 \quad (2.124)$$

分别表示“冷”和“比较冷”, 则

$$d(F, G) = 0.2 \times (|0.8 - 0.9| + |0.5 - 0.6| + |0.1 - 0.2|) = 0.2 \times 0.3 = 0.06 \quad (2.125)$$

即  $F$  和  $G$  的海明距离为 0.06.

设  $F$  和  $G$  分别是论域  $U = u_1, u_2, \dots, u_n$  上的两个模糊概念的模糊集, 则它们的贴近距离定义为

$$(F, G) = (1/2)(F \cdot G + (1 - F \odot G)) \quad (2.126)$$

其中:

$$\begin{aligned} F \cdot G &= \bigvee (\mu_F(u_i) \wedge \mu_G(u_i)) \\ F \odot G &= \bigwedge_U (\mu_F(u_i) \vee \mu_G(u_i)) \end{aligned} \quad (2.127)$$

称  $F \cdot G$  为内积,  $F \odot G$  为外积.

**例 2.43** 设论域  $U$  及其上的模糊集  $F$  和  $G$  如上例所示, 则

$$\begin{aligned} F \cdot G &= 0.8 \wedge 0.9 \vee 0.5 \wedge 0.6 \vee 0.1 \wedge 0.2 \vee 0 \wedge 0 \vee 0 \wedge 0 = 0.8 \vee 0.5 \vee 0.1 \vee 0 \vee 0 = 0.8 \\ F \odot G &= (0.8 \vee 0.9) \wedge (0.5 \vee 0.6) \wedge (0.1 \vee 0.2) \wedge (0 \vee 0) \wedge (0 \vee 0) \\ &= 0.9 \wedge 0.6 \wedge 0.2 \wedge 0 \wedge 0 = 0 \\ (F, G) &= 0.5 \times (0.8 + (1 - 0)) = 0.5 \times 1.8 = 0.9 \end{aligned}$$

即  $F$  和  $G$  的贴近距离为 0.9.

## 2.8.2 模糊推理

模糊推理实际上是按照给定的推理模式, 通过模糊集合与模糊关系的合成来实现的. 主要讨论:

### 2.8.2.0.1 模糊关系的构造

**模糊关系  $R_m$**   $R_m$  是由扎德提出的一种构造模糊关系的方法. 设  $F$  和  $G$  分别是论域  $U$  和  $V$  上的两个模糊集, 则  $R_m$  定义为

$$R_m = \int_{U \times V} (\mu_F(u) \wedge \mu_G(v)) \vee (1 - \mu_F(u)) / (u, v) \quad (2.128)$$

其中,  $\times$  号表示模糊集的笛卡尔乘积.

**例 2.44** 设  $U = V = \{1, 2, 3\}$ ,  $F$  和  $G$  分别是  $U$  和  $V$  上的两个模糊集, 且  $F = 1/1 + 0.6/2 + 0.1/3$ ,  $G = 0.1/1 + 0.6/2 + 1/3$ , 求  $U \times V$  上的  $R_m$

$$R_m = \int_{U \times V} (\mu_F(u) \wedge \mu_G(v)) \vee (1 - \mu_F(u)) / (u, v)$$

解:

$$R_m = \begin{bmatrix} 0.1 & 0.6 & 1 \\ 0.4 & 0.6 & 0.6 \\ 0.9 & 0.9 & 0.9 \end{bmatrix}$$

如:  $R_m(2,3) = (0.6 \wedge 1) \vee (1 - 0.6) = 0.6 \vee 0.4 = 0.6$ .

**模糊关系  $R_c$**   $R_c$  是由麦姆德尼 (Mamdani) 提出的一种构造模糊关系的方法. 设  $F$  和  $G$  分别是论域  $U$  和  $V$  上的两个模糊集, 则  $R_c$  义为

$$R_c = \int_{U \times V} (\mu_F(u) \wedge \mu_G(v))_Q / (u, v)$$

**例 2.45** 对例 6.12 所给出的模糊集

$$F = 1/1 + 0.6/2 + 0.1/3, G = 0.1/1 + 0.6/2 + 1/3$$

其  $R_c$  为

$$R_c = \begin{bmatrix} 0.1 & 0.6 & 1 \\ 0.1 & 0.6 & 0.6 \\ 0.1 & 0.1 & 0.1 \end{bmatrix},$$

如  $R_c(3,2)$ :  $R_c(3,2) = \mu_F(u_3) \wedge \mu_G(v_2) = 0.1 \wedge 0.6 = 0.1$ .

**模糊关系  $R_g$**   $R_g$  是米祖莫托 (Mizumoto) 提出的一种构造模糊关系的方法. 设  $F$  和  $G$  分别是论域  $U$  和  $V$  上的两个模糊集, 则  $R_g$  定义为

$$R_g = \int_{U \times V} (\mu_F(u) \rightarrow \mu_G(v)) / (u, v)$$

其中

$$\mu_F(u) \rightarrow \mu_G(v) = \begin{cases} 1, & \mu_F(u) \leq \mu_G(v) \\ \mu_G(v), & \mu_F(u) > \mu_G(v) \end{cases}$$

**例 2.46** 对例 6.12 所给出的模糊集

$$F = 1/1 + 0.6/2 + 0.1/3, G = 0.1/1 + 0.6/2 + 1/3$$

其  $R_g$  为

$$R_g = \begin{bmatrix} 0.1 & 0.6 & 1 \\ 0.1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

**模糊假言推理** 设  $F$  和  $G$  分别是  $U$  和  $V$  上的两个模糊集, 且有知识

知识: IF  $x$  is  $F$  THEN  $y$  is  $G$

若有  $U$  上的一个模糊集  $F'$ , 且  $F$  可以和  $F'$  匹配, 则可以推出  $y$  is  $G'$ , 且  $G'$  是  $V$  上的一个模糊集. 这种推理模式称为模糊假言推理, 其表示形式为:

知识: IF  $x$  is  $F$  THEN  $y$  is  $G$

证据: IF  $x$  is  $F'$

-----

结论: THEN  $y$  is  $G'$

在这种推理模式下, 模糊知识

IF  $x$  is  $F$  THEN  $y$  is  $G$

表示在  $F$  与  $G$  之间存在着确定的因果关系, 设此因果关系为  $R$ . 则有

$$G' = F' \circ R$$

其中的模糊关系  $R$ , 可以是  $R_m$ 、 $R_c$  或  $R_g$  中的任何一种.

### 2.8.2.0.2 模糊推理的基本方法

**例 2.47** 对例 4.19 所给出的  $F$ 、 $G$ , 以及所求出的  $R_m$ , 设有已知事实:  $\{x \text{ is 较小}\}$ , 并设“较小”的模糊集为: 较小  $= 1/1 + 0.7/2 + 0.2/3$ , 求在此已知事实下的模糊结论.



**解:** 本例的模糊关系  $R_m$  已在例 6.12 中求出, 设已知模糊事实“较小”为  $F'$ ,  $F'$  与  $R_m$  的合成即为所求结论  $G'$ .

$$G' = F' \circ R_m = \{1, 0.7, 0.2\} \circ \begin{bmatrix} 0.1 & 0.6 & 1 \\ 0.4 & 0.6 & 0.6 \\ 0.9 & 0.9 & 0.9 \end{bmatrix} = 0.4, 0.6, 1$$

即所求出的模糊结论  $G'$  为  $G' = 0.4/1 + 0.6/2 + 1/3$ .

**模糊拒取式推理** 设  $F$  和  $G$  分别是  $U$  和  $V$  上的两个模糊集, 且有知识

$$\text{IF } x \text{ is } F \text{ THEN } y \text{ is } G$$

若有  $V$  上的一个模糊集  $G'$ , 且  $G$  可以和  $G'$  匹配, 则可以推出  $x \text{ is } F'$ , 且  $F'$  是  $U$  上的一个模糊集. 这种推理模式称为**模糊拒取式推理**, 其表示形式为:

知识:     IF  $x \text{ is } F$  THEN  $y \text{ is } G$

证据:                     THEN  $z \text{ is } H$

-----

结论:     IF  $x \text{ is } F'$

在这种推理模式下, 模糊知识

$$\text{IF } x \text{ is } F \text{ THEN } y \text{ is } G$$

也表示在  $F$  与  $G$  之间存在着确定的因果关系, 设此因果关系为  $R$ , 则有

$$F' = R \circ G'$$

其中的模糊关系  $R$ , 可以是  $R_m$ 、 $R_c$  或  $R_g$  中的任何一种.

**例 2.48** 设  $F$  和  $G$  如例 4.19 所示, 已知事实为  $\{y \text{ is 较大}\}$  且“较大”的模糊集为: 较大  $= 0.2/1 + 0.7/2 + 1/3$ , 若已知事实与  $G$  匹配, 以模糊关系  $R_c$  为例, 在此已知事实下推出  $F'$ .

**解:** 本例的模糊关系  $R_c$  已在前面求出, 设模糊概念“较大”为  $G'$ , 则  $R_c$  与  $G'$  的合成即为所求的  $F'$ .

$$F' = R_c \circ G' = \begin{bmatrix} 0.1 & 0.6 & 1 \\ 0.1 & 0.6 & 0.6 \\ 0.1 & 0.1 & 0.1 \end{bmatrix} \circ \begin{bmatrix} 0.2 \\ 0.7 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.6 \\ 0.1 \end{bmatrix}$$

即所求出的  $F'$  为  $G' = 1/1 + 0.6/2 + 0.1/3$ .

**模糊假言三段论推理** 设  $F, G, H$  分别是  $U, V, W$  上的 3 个模糊集, 且由知识

IF  $x$  is  $F$  THEN  $y$  is  $G$

IF  $y$  is  $G$  THEN  $z$  is  $H$

则可推出:

IF  $x$  is  $F$  THEN  $z$  is  $H$

这种推理模式称为**模糊假言三段论推理**. 它可表示为:

IF  $x$  is  $F$  THEN  $y$  is  $G$

IF  $y$  is  $G$  THEN  $z$  is  $H$

-----

IF  $x$  is  $F$  THEN  $z$  is  $H$

在模糊假言三段论推理模式下, 模糊知识

$r_1$  : IF  $x$  is  $F$  THEN  $y$  is  $G$

表示在  $F$  与  $G$  之间存在着确定的因果关系, 设此因果关系为  $R_1$ .

模糊知识

$r_2$  : IF  $y$  is  $G$  THEN  $z$  is  $H$

表示在  $G$  与  $H$  之间存在着确定的因果关系, 设此因果关系为  $R_2$ .

若模糊假言三段论成立, 则模糊结论

$r_3$  : IF  $x$  is  $F$  THEN  $z$  is  $H$

的模糊关系  $R_3$  可由  $R_1$  与  $R_2$  的合成得到. 即

$$R_3 = R_1 \circ R_2$$

这里的关系  $R_1$ 、 $R_2$  和  $R_3$  都可以是前面所讨论过的  $R_m$ 、 $R_c$ 、 $R_g$  中的任何一种.

**例 2.49** 设  $U = W = V = 1, 2, 3$ ,  $E = 1/1 + 0.6/2 + 0.2/3$ ,  $F = 0.8/1 + 0.5 + 0.1/3$ ,  $G = 0.2/1 + 0.6 + 1/3$ . 按  $R_g$  求  $E \times F \times G$  上的关系  $R$ .

**解:** 先求  $E \times F$  上的关系  $R_1$

$$R_1 = \begin{bmatrix} 0.8 & 0.5 & 0.1 \\ 1 & 0.5 & 0.1 \\ 1 & 1 & 0.1 \end{bmatrix}$$

再求  $E \times G$  上的关系  $R_2$

$$R_2 = \begin{bmatrix} 0.2 & 0.6 & 1 \\ 0.2 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

最后求  $E \times F \times G$  上的关系  $R$

$$R = R_1 \circ R_2 = \begin{bmatrix} 0.2 & 0.6 & 0.8 \\ 0.2 & 0.6 & 1 \\ 0.2 & 1 & 1 \end{bmatrix}$$

## 2.9 高阶模糊推理

### 2.10 作业

#### 思考

设有如下一组推理规则:

r1: IF E1 THEN E2 (0.6)

r2: IF E2 AND E3 THEN E4 (0.7)

r3: IF E4 THEN H (0.8)

r4: IF E5 THEN H (0.9)

且已知  $CF(E1) = 0.5$ ,  $CF(E2) = 0.6$ ,  $CF(E3) = 0.7$ . 求  $CF(H) = ?$

**思考**

设  $U = V = \{1, 2, 3, 4, 5\}$  且有如下推理规则:

IF  $x$  is 少 THEN  $y$  is 多

其中, “少”与“多”分别是  $U$  与  $V$  上的模糊集, 设

$$\text{少} = 0.9/1 + 0.7/2 + 0.4/3 \quad (2.129)$$

$$\text{多} = 0.3/3 + 0.7/4 + 0.9/5 \quad (2.130)$$

已知事实为

$$x \text{ is 较少} \quad (2.131)$$

“较少”的模糊集为

$$\text{较少} = 0.8/1 + 0.5/2 + 0.2/3 \quad (2.132)$$

请用模糊关系  $R_m$  求出模糊结论.

