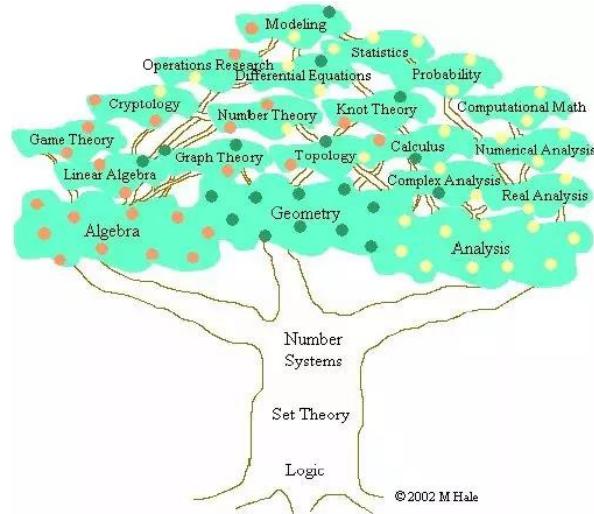




本作品采用知识共享署名-非商业性使用 4.0 国际许可协议进行许可。访问<http://creativecommons.org/licenses/by-nc/4.0/>查看该许可协议。

前 言

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



史忠植——2013年凭借“拓展知识工程核心理论、创新分布智能理论基础、构建智能科学理论体系”成果,荣获第三届吴文俊人工智能科学技术奖成就奖.

周志华——教育部“长江学者”特聘教授,国家杰出青年基金获得者;南京大学人工智能学院院长.有一个旧称叫国立东南大学.

[Science daily](#)

在线课程 课程相关的电子材料存放平台和链接地址。

[码云平台上的课程电子版和代码](#)

[Github 平台上的课程电子版和代码](#)

[1——人工智能基础-第一次基础简介](#)

[2——人工智能基础-第二次-发展和应用](#)

[3——人工智能——知识表示](#)





4——人工智能第三章——知识推理方法 泰迪云课堂

台大李宏毅老师的机器学习课程-2020

泰迪-深度学习原理及编程实现 _ 人邮版

目 录

前 言	i
概 览	xv
第一章 人工智能概述	1
1.1 AI 的定义及其研究目标	2
1.1.1 AI 的定义及其研究目标	2
1.1.2 何谓智能(自然智能)	2
1.1.3 认识智能的几种理论	9
1.1.4 智能的层次结构	10
1.1.5 智能包含的能力	10
1.1.6 何谓人工智能?	11
1.1.7 人工智能的近远期目标	13
1.1.8 人工智能对计算机科学的影响	13
1.1.9 计算机科学与技术、软件工程、物联网、大数据专业的区别	16
1.1.10 2020 年十大科技趋势预测——百度研究院	17
1.1.11 美国人工智能计划: 2020 年度报告	20
1.2 AI 的产生与发展	21
1.2.1 AI 的产生与发展-孕育期(1956 年以前)	22
1.2.2 AI 的产生与发展-形成期(1956—1970 年)	22
1.2.3 AI 的产生与发展-知识应用期(1971—80 年代末)	24
1.2.4 AI 的发展-知识深化期(1990-2000 年代末)	25
1.2.5 数据智能——发展期(2000-2009 年末)	26
1.2.6 数据智能——爆发期(2010-现在)	27
1.2.7 AI 的产生与发展-从学派分立到综合(20 世纪 80 年代到本世纪初)	31

1.2.8	AI 成功的标志: IBM 的“深蓝”和“小深”	32
1.2.9	编程语言	33
1.3	AI 研究的基本内容	43
1.3.1	AI 的学科位置	44
1.3.2	人工智能学科 (F06) 代码	45
1.3.3	与脑科学和认知科学的交叉研究-脑科学	45
1.3.4	智能模拟的方法和技术研究	47
1.4	AI 研究中的不同学派——不同学派	47
1.5	人工智能近期发展分析	49
1.5.1	人工智能发展方向	50
1.5.2	人工智能发展八大新趋势	52
1.5.3	达摩院 2020 十大科技趋势	56
1.6	AI 的相关应用领域	58
1.6.1	机器思维	59
1.6.2	机器感知	60
1.6.3	机器行为	62
1.6.4	计算智能	63
1.6.5	机器学习	65
1.6.6	分布智能	67
1.6.7	智能系统	68
1.6.8	人工心理与人工情感	69
1.6.9	人工生命	70
1.6.10	自动推理	70
1.6.11	图神经网络——Open Graph Benchmark	73
1.6.12	工业 4.0	73
1.7	人工智能的典型应用	77
1.7.1	博弈	77
1.7.2	深度学习求解高维偏微分方程	77
1.7.3	自动定理证明	80
1.7.4	国家安全领域	82
1.7.5	机器人	83
1.7.6	机械臂	85
1.7.7	闲鱼垃圾评论过滤系统	85

1.7.8	PCB 产业	87
1.7.9	百度飞桨: 开源开放的产业级深度学习平台	89
1.7.10	百度飞桨: 开源开放的产业级深度学习平台	90
1.7.11	计算机视觉的语义分割	91
1.7.12	威胁检测和预防	93
1.7.13	人工智能案例	96
1.8	人工智能的学习平台	106
1.8.1	Tensorflow	106
1.8.2	Kubeflow	113
1.8.3	自动机器学习 (AutoML) 框架	113
1.8.4	Angel——腾讯机器学习平台	116
1.8.5	Metaflow	117
1.8.6	华为 MindSpore 框架	118
1.8.7	CNTK	118
1.8.8	伯克利官方 AI 开源框架 Ray	122
1.8.9	DEAP	122
1.8.10	AI 的学习资源网站	123
1.9	我国智能科学技术教育体系	125
1.10	智能科学与技术研究生专业建设-建议名称	125
1.11	人工智能伦理	126
1.11.1	亚马逊智能助手 Alexa 劝人自杀	126
1.12	作业	127
第二章	知识表示方法	129
2.1	知识与知识表示	130
2.1.1	知识	130
2.1.2	知识表示的概念	132
2.2	一阶谓词逻辑表示法	133
2.2.1	一阶谓词逻辑表示的逻辑学基础——命题与真值	133
2.2.2	谓词逻辑表示的特征	141
2.2.3	知识表示——产生式表示法	143
2.2.4	规则的表示	144
2.2.5	产生式系统的基本结构——系统结构及说明	145
2.2.6	产生式系统的过 程——基本过程	148

2.2.7 产生式系统的特点	151
2.3 语义网络表示法	152
2.3.1 语义网络	152
2.3.2 表示一元关系	156
2.3.3 二元关系表示	157
2.3.4 表示多元关系	159
2.3.5 存在和全称量词的表示	162
2.3.6 语义网络的推理过程	163
2.3.7 继承	164
2.3.8 匹配	164
2.3.9 语义网络表示法的优缺点:	165
2.4 框架表示法	166
2.4.1 框架理论	166
2.4.2 框架结构和框架表示	166
2.4.3 框架表示法的优缺点	172
2.5 面向对象的知识表示 (知识 + 方法)	173
2.6 过程表示法	173
2.6.1 过程表示的主要优缺点	175
2.7 模糊认知图	175
2.7.1 模糊认知图的参数训练	176
2.8 小结	177
2.9 作业	177
第三章 知识推理方法	181
3.1 什么是推理	182
3.1.1 推理所用的事实	182
3.1.2 按推理的逻辑基础分类	182
3.1.3 推理的控制策略及分类	186
3.1.4 正向推理	186
3.1.5 混合推理的概念	190
3.2 逻辑学与人工智能	190
3.2.1 人工智能学科的诞生	191
3.2.2 逻辑学的发展	192
3.2.3 逻辑学在人工智能学科的研究方面的应用	192

3.2.4 人工智能——当代逻辑发展的动力	193
3.2.5 人工智能的产生与发展和逻辑学的发展	194
3.3 推理的逻辑基础	194
3.3.1 谓词公式	194
3.3.2 谓词公式的永真性和可满足性	196
3.3.3 谓词公式的范式	199
3.3.4 置换与合一	199
3.4 自然演绎推理	202
3.5 归结演绎推理	203
3.5.1 子句集及其化简	203
3.5.2 鲁滨逊归结原理	209
3.6 归结演绎推理的归结策略	221
3.6.1 广度优先策略	221
3.6.2 支持集策略	222
3.6.3 删除策略	223
3.6.4 用归结反演求取问题的答案	227
3.7 基于规则的演绎推理	229
3.7.1 规则正向演绎推理	229
3.7.2 规则逆向演绎推理	236
3.8 知识图谱 (knowledge graph) 的一阶归纳推理	237
3.8.1 基于符号逻辑的推理——本体推理	237
3.8.2 基于图结构和统计规则挖掘的推理	238
3.8.3 知识图谱推理 (路径排序算法)	238
3.8.4 基于路径排序学习方法有 PRA 和 Path ranking Algorithm	238
3.8.5 斯坦福的七步法知识图谱构建步骤	238
3.8.6 因果推理 (辛普森悖论、D-分离和 do 算子)	241
3.9 模糊逻辑 *	241
3.10 多值逻辑	244
3.11 Luckasiewicz 逻辑	244
3.12 三 I 算法的逻辑基础	244
3.13 强化学习	244
3.14 作业	246

第四章 产生式系统的搜索策略	249
4.1 搜索的基本概念	250
4.1.1 搜索的含义	250
4.2 搜索的类型	250
4.2.1 状态空间法	250
4.2.2 问题归约法	255
4.3 状态空间的盲目搜索	258
4.3.1 一般图搜索过程	258
4.3.2 广度优先和深度优先搜索	260
4.3.3 代价树搜索	263
4.4 搜索策略	264
4.4.1 状态空间的启发式搜索	264
4.4.2 A 算法	266
4.4.3 A* 算法	267
4.4.4 A* 算法的可采纳性	268
4.4.5 A* 算法应用举例	274
4.4.6 算法应用举例	275
4.5 与/或树的启发式搜索	279
4.6 博弈树的启发式搜索	283
4.6.1 极大极小过程	285
4.6.2 alpha-beta 剪枝	287
4.7 作业	288
第五章 计算智能	291
5.1 概述	292
5.1.1 什么是计算智能	292
5.1.2 计算智能与人工智能的关系	292
5.1.3 计算智能的产生与发展	293
5.2 浅层神经计算	294
5.2.1 神经计算基础	294
5.2.2 人工神经网络的互连结构	296
5.3 人工神经网络的典型模型	300
5.3.1 感知机 (Perceptron) 模型	300
5.3.2 反向传播 (BP) 模型	305

5.3.3 反馈网络 (Hopfield) 模型	306
5.3.4 神经网络的泛逼近性——可视化证明	307
5.4 其它神经学习算法	307
5.4.1 TROP-ELM 算法	310
5.4.2 Allen's PRESS(prediction sum of squares)	310
5.4.3 区间二型 RVFL	312
5.5 进化计算	324
5.5.1 进化计算的产生与发展	327
5.5.2 进化计算的基本结构	328
5.6 遗传编码	330
5.7 QBFA	346
5.8 模糊计算	346
5.8.1 模糊集及其运算	348
5.8.2 模糊关系及其运算	351
5.8.3 模糊关系的合成	353
5.9 作业	354
第六章 不确定性推理	359
6.1 不确定性推理的含义	360
6.2 不确定性推理的基本问题	360
6.3 不确定性推理的概率论基础	361
6.3.1 可信度的概念	364
6.3.2 CF 模型	364
6.3.3 可信度的定义与性质	365
6.4 证据理论	367
6.4.1 证据不确定性的表示	367
6.4.2 不确定性的更新	368
6.4.3 结论不确定性的合成	368
6.5 主观 Bayes 方法	370
6.6 证据理论	371
6.7 DS 理论的形式描述	371
6.7.1 证据理论的推理模型	377
6.8 模糊推理	386
6.8.1 模糊概念的匹配	388

6.8.2 模糊推理	389
6.9 高阶模糊推理	394
6.10 作业	395
第七章 机器学习	397
7.1 机器学习	398
7.1.1 机器学习概述	398
7.2 机器学习的主要策略	402
7.3 netZooPy 库	402
7.4 记忆学习	403
7.5 机械式学习的学习模型	404
7.5.1 归纳学习	404
7.5.2 示例学习的类型	404
7.6 示例学习的解释方法	405
7.6.1 ID3 算法	408
7.6.2 解释学习的空间描述	412
7.6.3 解释学习的模型	413
7.6.4 解释学习的基本原理	413
7.6.5 浅层神经学习的心理学基础	417
7.6.6 感知器学习	418
7.6.7 BP 网络学习	423
7.6.8 Hopfield 网络学习	423
7.7 深度神经网络——DNN	425
7.7.1 高速公路网络 (Highway network)	428
7.7.2 残差网络	429
7.8 卷积神经网络 (Convolutional Neural Networks, CNN)	430
7.8.1 卷积神经网络结构	435
7.8.2 CNN 的应用	455
7.8.3 卷积编译码网络学习语义图形实现水稻自动除草	456
7.8.4 图卷积神经网络	459
7.8.5 循环神经网络 (Recurrent Neural Networks, RNN)	464
7.8.6 深度神经网络轻量化	473
7.8.7 防止 CNN 的过拟合——diffGrad	476
7.8.8 无限宽度 DNN	476

7.9 强化学习及深度强化学习	477
7.10 AI 的未来-NeurIPS 2019	477
7.11 作业	478
第八章 自然语言理解	479
8.1 语言及其理解的基本概念	481
8.2 词法分析	482
8.3 句法分析	482
8.3.1 句法规则的表示方法	482
8.4 句义分析	487
8.4.1 语义文法	487
8.4.2 格文法	488
8.4.3 长程记忆模型——Compressive Transformer（压缩 Transformer） .	489
8.5 作业	490
第九章 分布智能	493
9.1 分布智能	494
9.1.1 分布智能	494
9.1.2 Agent 的结构	494
9.1.3 Agent 通信	495
9.1.4 多 Agent 合作	499
9.1.5 Agent 的协商	501
9.1.6 移动 Agent	502
9.2 作业	502
第十章 先进专家系统	503
10.1 专家系统	504
10.1.1 专家系统概述	504
10.1.2 专家系统的基本结构	504
10.1.3 基于规则和基于框架的专家系统	505
10.1.4 模糊专家系统和神经网络专家系统	506
10.1.5 基于 Web 的专家系统	506
10.1.6 分布式专家系统和协同式专家系统	507
10.2 专家系统的开发	507

10.3 作业	508
第十一章 计算机视觉 CV	509
11.1 深度学习 VS 传统计算机视觉	510
11.1.1 深度学习的优势	510
11.1.2 传统 CV 技术的优势	512
11.2 几何深度学习	514
11.3 作业	521

概 览

推荐参考文献-来自
电子教案地址：[人工智能基础——电子教案](#)

1

人工智能概述

人工智能

人工智能 (AI) 是研究、开发用于模拟、延伸和扩展人的智能的理论、方法、技术及应用系统的一门技术科学.

李飞飞

人工智能和机器学习仍然是一个进入门槛较高的领域，需要专业的知识和资源，很少有公司可以自己承担.

注：美国斯坦福大学计算机科学系教授，女，1976年出生于北京，2015年12月入选2015年“全球百大思想者”。2018年3月，获“影响世界华人大奖”。现为美国斯坦福大学红杉讲席教授，美国国家工程院院士，类人人工智能研究院（HAI）院长。

法国国民议会议员、数学家赛德里克·维拉尼 (Cédric Villani) 在其 2018 年提交的报告《有意义的人工智能》(For A Meaningful Artificial Intelligence) 中指出，“人工智能的雄心在于理解并复制人类认知”。



无名氏

传承是无法避免的，对机器也是可以这么要求的。

1.1 AI 的定义及其研究目标

1.1.1 AI 的定义及其研究目标

- ▶ 目前还没有形式化定义
- ▶ 一般解释：人工智能就是用人工的方法在机器（计算机）上实现的智能，或称机器智能。
- ▶ 无形式化定义的理由在于人工智能的严格定义依赖于对智能的定义：即要定义人工智能，首先应该定义智能，但智能本身也还无严格定义。

定义 1.1 人工智能-宋士吉

人工智能是研究、开发用于模拟、延伸和扩展人的智能的理论、方法、技术及应用系统。

因此，先对人类的自然智能进行讨论。研究思路就是用人工智能逼近自然智能。

1.1.2 何谓智能（自然智能）

- ▶ 自然智能：指人类和一些动物所具有的智力和行为能力。
- ▶ 人类的自然智能（简称智能）：指人类在认识客观世界中，由思维过程和脑力活动所表现出的综合能力。

- ▶ 人类大脑如何实现智能: 不是很清楚人类大脑的信息处理机制和运行机理. 简单来说, 一个人可能在识别图片的时候由于各种劳累和马虎, 在这个数据集的错误率高于机器. 但是只要你去和它谈任何一个图片它所理解的东西, 比如一个苹果, 你都会震惊于其信息之丰富, 不仅包含了真实苹果的各种感官, 还包含了关于苹果的各种文学影视, 从夏娃的苹果, 到白雪公主的苹果. 应该说, 人类理解的苹果更加接近概念网络里的一个节点, 和整个世界的所有其它概念相关联, 而非机器学习分类器里的 n 个互相分离的高斯分布 (机器需要辨识苹果、树、草地和人等). 大脑的化学反应的变化可以改变行为; Charles Robert Darwin 说: 人和高等动物, 显然只有程度上的区别, 种类上别无二致. 神经科学认为的一个假设是: 认知能力的提高跟大脑随着进化越变越大有关系.
- ▶ 实现智能的两大难题之一: 宇宙起源和人脑奥秘; 然而, 我们对人脑奥秘知之甚少.
- ▶ 目前已知关于人脑的知识

人脑结构: $10^{11} - 10^{12}$ 量级的神经元, 分布并行;

人脑功能: 记忆、思维、观察、分析等.

脑结构的三个不同层面: 区域层面、细胞结构层面和分子层面.

① 大脑皮层位于大脑的外部, 有点像一张覆盖在大脑其他部分之上皱巴巴的大抹布. 人类大脑和其他灵长类动物大脑在体积上的差异, 主要是皮层增大所致. 大脑皮层是高度关联的, 在大脑的所有连接中, 75% 位于皮层之内, 仅有 25% 是通往大脑其他部分和神经系统的输入输出连接.

② 新皮层是大脑皮层中较晚进化出来的区域, 它负责感官知觉、发出运动指令、进行空间推理和有意识思维, 也是智人语言产生的地方. 解剖学将新皮层分为四个脑叶: 额叶、顶叶、颞叶和枕叶. 对包括人类在内的灵长类动物而言, 新皮层大得异乎寻常. 刺猬的新皮层占大脑重量的 16%, 夜猴 (一种小型猴子) 占 46%, 黑猩猩占 76%. 人类新皮层所占的比例则更大.

当动物觉察到安全信息时, 杏仁核-内侧前额叶-海马环路的 theta 同步化活动降低, 而高频 gamma 同步化活动增强. 在恐惧记忆消退期间, 杏仁核中间神经元的 6-12 Hz 振荡及高频 gamma (70-120 Hz) 振荡强度增加, 与恐惧提取相关的低频 gamma (30-70 Hz) 活动强度则降低, 另外, 杏仁核的高频 gamma 与 mPFC 的 theta 振荡发生跨频段耦合, 并接受来自 mPFC 的抑制性 theta 输入. 随着消退的进行, mPFC 的高频 gamma 振荡增强, 以调节恐惧记忆的消退. 以人类为对象的研究结果显示, 恐惧记忆的习得与 dACC (图中橙色区域) 相关, 而 vmPFC 更多地参与到恐惧消退中. 在消退期间, vmPFC (图中红色区域) 内低频 gamma

(36~44 Hz) 活动减弱。与此同时，海马通过调控情景信息参与到恐惧记忆的消退中（图中绿色所示为人类研究结果，蓝色所示为动物研究结果）。



Michael S. Gazzaniga 认为新皮层枕叶包括初级视觉皮层，也叫纹状皮层。黑猩猩的纹状皮层占整个新皮层的 5%，人类的纹状皮层只占 2%，低于预期值。

生物神经网络最基本的结构特点是多层，无论是视觉、听觉，我们说基本的神经回路都有层级结构，而且经常是六层。这种纵深的层级，对应的编码原理正是从具体特征到抽象特征的层级编码结构。最有名的莫过于祖母细胞，这一思路直接催生了以 CNN 为代表的深度学习。从图 1-4 中的右侧子图可以看出，尼氏染色主要染的是细胞体，而高尔基染色主要染的是树突。

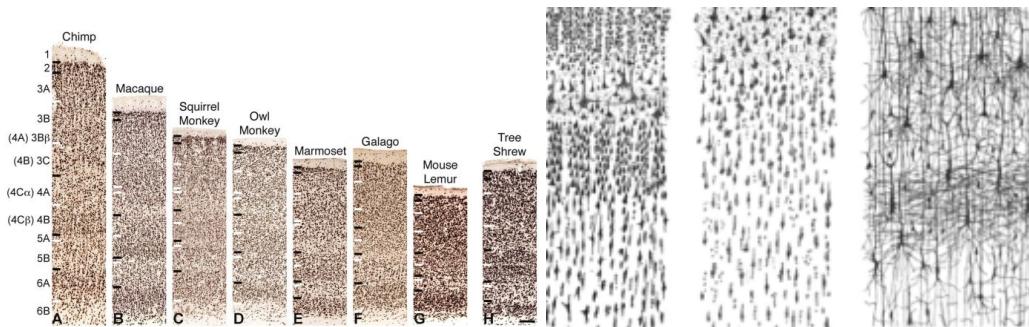


图 1-1 灵长类动物的初级视觉皮层分层结构，从左至右分别为黑猩猩，猕猴，松鼠猴，猫头鹰猴，狨猴，婴猴，狐猴，树鼩；人脑的视觉皮层（尼氏染色）；中：人脑的运动皮层（尼氏染色）；右：半个月大婴儿的皮层（高尔基染色）

在进入学龄期这个时期后 (7-12-18)，人的脑容量都能达到正常水平。即使人与人

之间在脑容量和褶皱结构有细微差异，这种差异也不影响智力水平。不过，这段时期脑内部各个功能区的灰质白质比例，却的确开始发生很大变化。总体来讲，这段时期人的脑灰质占比是不断缓慢下降的，而白质占比则不断增高。



图 1-2 不同时期脑部神经细胞密度

动物在做决定时如何利用环境 2020 年 3 月 20 日，南加州大学，复旦大学等全球 296 家单位协作在 Science 发表题为 “The genetic architecture of the human cerebral cortex”的研究论文，该研究对来自 51,665 个人的脑磁共振成像数据进行了全基因组关联荟萃分析，研究人员分析了整个皮质和 34 个已知功能脑区域的表面积和平均厚度。

该研究确定了 199 个重要的基因座，并发现了显著富集的基因座，可影响产前皮质发育过程中活跃的调节元件内的总表面积，支持放射状单位假说。影响区域表面积的基因座聚集在 Wnt 信号通路中的基因附近，从而影响祖细胞的扩增和脑区域身份。皮层结构的变化与认知功能，帕金森氏病，失眠，抑郁，神经质和注意缺陷多动障碍在遗传上相关。总而言之，这项工作确定了与皮质表面积和皮质厚度相关的全基因组重要位点，并使人们对人类大脑皮层的遗传结构及其模式有了更深入的了解。人的大脑皮层是大脑的外部灰质层，涉及更高的认知功能的多个方面。其独特的折叠模式具有凸（回旋）和凹（沟）区域的特征。计算脑图绘制方法使用跨单个皮层的一致折叠模式来标记脑区域。在胎儿发育过程中，兴奋性神经元（皮质中主要的神经元细胞类型）是由发育中的生发区的神经祖细胞产生的。皮质表面积（SA）的扩大是由这些神经祖细胞的增殖驱动的，而厚度（TH）由其神经原性分裂的数目决定。皮质 SA 和 TH 的整体和区域测量值的变化已可靠地与神经精神疾病和心理特征相关联。



图 1-3 识别遗传因素对人体皮质结构的影响

动物在做决定时如何利用环境 2020 年 3 月发表在的《神经元》(cell: Neuron-Context-Dependent Decision Making in a Premotor Circuit) 杂志上的一项研究发现, 这种利用环境的能力与老鼠大脑中一个被称为前外侧运动皮层 (ALM) 的区域, 以前人们认为它主要指导和计划运动. 这一发现为大脑非凡的决策能力提供了新的视角. 灵活的决策是理解周围环境的关键工具; 所处环境的使用允许决策者通过考虑上下文环境来对相同的信息做出不同的反应. “环境相关的决策是人类高级认知功能的基石,” 神经学家 Michael Shadlen 博士说. “就像我们今天的研究一样, 在老鼠大脑的运动区域观察这个过程, 让我们更接近于了解脑细胞和神经回路层面的认知功能.”

例 1.1.1

“如果有人在冷清的街道上不舒服地站在我身边, 我可能会试图逃跑, 但如果同样的事情发生在拥挤的地铁车厢里, 我就不会感到这样的危险,” 神经学家和第一作者郑(赫伯特)吴博士说. “我搬家或不搬家的决定取决于我所处的环境; 从而为我所做的选择提供一个理由.”



例 1.1.2

研究小组调查了大脑中几个专门处理和整合感觉信息的区域, *ALM* 关键区域是运动皮层的一部分。先前的实验表明, *ALM* 有一个相对简单的工作: 引导老鼠舌头和面部肌肉运动。基于这一认识, 研究人员设计了一项新的实验, 要求老鼠利用舌头和嗅觉系统做出灵活的决定, 而嗅觉系统是嗅觉的向导。在实验中, 一只老鼠第一次闻到一种气味。老鼠必须记住这种气味, 因为在短暂的停顿之后, 研究人员将另一种气味喷到老鼠的鼻孔上。如果两种气味相同, 老鼠必须舔左边的管子才能得到水。如果两种气味不同, 它必须向右舔一根管子。



先前在这类“延迟匹配样本”测试中所做的工作, 会让人认为老鼠会利用大脑中专门负责嗅觉的区域来决定舔食的方式。来自这些区域的大脑活动记录似乎证实了这一机制。“根据这些记录, 我们可以想象, 当老鼠闻到第二种气味时, 这些大脑区域就有了答案,” Shadlen 博士说。“剩下要做的就是把这个答案传递给大脑的运动系统, 从而产生对左右舔的适当反应。”如果是这样, 那么运动区域就不应该起作用, 直到提供了第二种气味, 而老鼠决定这两种气味是相同的还是不同的。吴博士想出了一个聪明的方法来验证这个预测。他关掉了老鼠的 *ALM*, 直到发出第二种气味, 然后及时打开 *ALM*, 让老鼠得到答案。按照标准的观点, 老鼠应该不会被这种操作所影响, 因为它们的嗅觉系统仍然完好无损。相反, 他们在这项任务中受到了损害。” Michael Shadlen, 医学博士, 研究论文的合著者, 哥伦比亚大学祖克曼研究所神经学家吴博士说: “我们的研究结果表明, 我们需要 *ALM* 来解决两种气味是否匹配的问题, 然后再决定舔到哪里, 这促使我们重新思考大脑是如何做出这些决定的。” *ALM* 与气味感知无关。

例 1.1.3

吴博士仔细观察了 *ALM* 中的脑细胞。他在离大脑表面很近的地方发现了一种新的神经元，这种神经元对第一种气味有反应。它将这些信息放在手边，直到接收到第二种气味。为了探索这一意想不到的结果，研究小组求助于理论神经学家 *Ashok Litwin-Kumar* 博士，以调查多种可能解释 *ALM* 作用的潜在机制。“传统观点认为，动物的嗅觉大脑区域应该自己处理气味，然后将信息反馈给 *ALM*，后者将指导舌头，” *Litwin-Kumar* 博士说。“但数据告诉我们一个不同的故事；第一种气味作为一种上下文线索，刺激 *ALM*，然后通过决定舔第二种气味的方式来表明这种关系。”



今天的发现，虽然集中在 *ALM* 上，但对于如何让科学家对整个大脑功能有更大的理解是很重要的。“最终，我们想要阐明解释简单行为的基本原理，但这也为深入了解人类的高级认知功能提供了思路，” *Shadlen* 博士说。要实现这个目标，关键的一步是用生物学和数学的语言把有关神经元、回路和行为的知识结合起来。这个合作项目凸显了这一战略的前景。”

分布强化学习的 AI 研究 DeepMind 与哈佛大学的研究人员受最近关于分布强化学习的 AI 研究启发，提出了一种基于多巴胺的强化学习的方法 [**DabneyKurth-Nelson2020-9599**]。和 AI 系统类似，大脑不是以“平均值”的方式预期未来可能的回报，而是以“概率分布”的方式来预期，从而证明大脑中存在“分布强化学习”。大脑进行强化学习，类似于顶级 AI 算法“大脑中的多巴胺是一种代表惊讶（surprise）的信号。” *Will Dabney* 说：“当情况好于预期时，就会释放出更多的多巴胺。”

生物神经系统包含种类丰富的抑制型神经元，它们往往在生物神经网络起到调控功能，如同控制信息流动的路由器，在合适的时候开启或关闭某个信号。当下的 AI 直接用 attention 的机制（把注意力集中放在重要的点上，而忽略其他不重要的因素。重要程度的判断取决于应用场景和主题的认识感知程度。根据应用场景的不同，Attention 分为空间注意力和时间注意力，前者用于图像处理，后者用于自然语言处理），或者 LSTM 里的输入门来调控是否让某个输入进入网络，其它一点类似路由器的作用，但是种类和形式的多样性远不及生物系统。皮层内的神经元都采取簇状结构，细胞之间不是独立的存在，而是聚集成团簇，犹如一个微型的柱状体。这些柱状体成为信息传输的基本单元。图1-4的右侧子图是初级感觉皮层

(左) 和初级运动皮层 (右) 的分层结构, 可以看到初级感觉皮层的第四层最厚, 而初级运动皮层的第四层特别薄.



图 1-4 兴奋型和抑制型神经元; 皮质小柱

意识可以理解为自我对自我本身的感知. 关于意识的起源, 已经成为一个重要的神经科学探索方向, 意识与多个脑区协同的集体放电相关 (The controversial correlates of consciousness -Science 2018). 但是, 关于意识的一个重大疑团是它对认知和智能到底有什么作用, 还是一个进化的副产物.

人类大脑海马区神经元放电峰值频率 500Hz, 以此推断人类大脑最大计算频率为 500hz, 而现在普通电脑的 cpu 运算也是 3.0ghz 以上, 而且还是 4 核芯, 每秒理论运算次数达到了惊人的 120 亿次, 运算速度已经远远超过人类大脑. 机器学习认知方面当下展示出来的智力水平最多也就是 3-5 岁的小孩子, 计算效率存在差异.

- 对智能的严格定义有待于人脑奥秘的揭示, 进一步认识.

1.1.3 认识智能的几种理论

- 思维理论

智能来源于思维活动, 智能的核心是思维, 人的一切知识都是思维的产物. 可望通过对思维规律和思维方法的研究, 来揭示智能的本质.

- 知识阈值理论

智能取决于知识的数量及其可运用程度. 一个系统所具有的可运用知识越多, 其智能就会越高.

► 进化理论

是美国 MIT 的 Brooks 在对人造机器虫研究的基础上提出来的。智能取决于感知和行为，取决于对外界复杂环境的适应，智能不需要知识、不需要表示、不需要推理，智能可由逐步进化来实现。

1.1.4 智能的层次结构

各种理论的观点不一致，从层次结构来说

- 高层智能：以大脑皮层（抑制中枢）为主，主要完成记忆、思维等活动。
- 中层智能：以丘脑（感觉中枢）为主，主要完成感知活动。
- 低层智能：以小脑、脊髓为主，主要完成动作反应活动。

💡 **注 1.2.** 不同观点在层次结构中的对应关系：高层智能：思维理论、知识阈值理论、进化理论。对应基础层。中层智能和低层智能分别对应技术层和应用层。

思考

大脑是深度学习吗？

💡 **注 1.3.** 大脑是一个复杂得难以想象的东西，它把许许多多结构不同的东西包括在内，而我们对大脑的了解还太少；大脑是不是深度学习，我们还给不出确定的答案。大脑总体来说不是深度学习，不过其中的某一些子模块可以用深度学习来描述，或者是部分符合深度学习的，比如视觉皮层就有深度层次化的特征表征，即便这些表征不都是学习得到的；视觉皮层也是深度学习的研究中重要的灵感来源。

1.1.5 智能包含的能力

- 感知能力：通过感知器官感知外界的能力。是人类获得外界信息的基本途径，其处理方式有以下两种：
 - 感知-动作方式：对简单、紧急信息。
 - 感知-思维-动作方式：对复杂信息。
- 记忆能力
 记忆：对感知到的外界信息和由思维产生的内部知识的存储过程。
- 思维能力 **思维：**对已存储信息或知识的本质属性、内部知识的认识过程。
 思维方式：
 - 抽象思维（逻辑思维）：根据逻辑规则对信息和知识进行处理的理性思维方式。

例 1.4 逻辑推理等.

- 形象思维(直感思维): 基于形象概念, 根据感性形象认识材料对客观现象进行处理的一种思维方式.

例 1.5 图像、景物识别等.

- 灵感思维(顿悟思维): 是一种显意识和潜意识相互作用的思维方式.

例 1.6 因灵感而顿时开窍.

► 学习和自适应能力

- 学习: 是一个具有特定目的的知识获取过程, 是人的一种本能. 不同人的学习方法、能力不同.
 - 自适应: 是一种通过自我调节适应外界环境的过程, 是人的一种本能. 不同人的适应能力不同.
- 行为能力: 是人们对感知到的外界信息作出动作反应的能力.
- 信息来源: 由感知直接获得的外界信息, 经过思维加工后的信息.
 - 实现过程: 通过脊髓来控制(来自四肢和躯干的各种感觉冲动都是通过脊髓的上行纤维束来传导, 即传导面部以外的痛觉、温度觉和粗触觉的脊髓丘脑束、传导本体感觉和精细触觉的薄束和楔束等, 以及脊髓小脑束的小脑本体感觉). 由语言、表情和体姿等来实现.

1.1.6 何谓人工智能?

UC Berkely 人工智能系统中心创始人兼计算机科学专业教授 Stuart Russell

人工智能是研究让计算机展现出智慧的方法. 计算机获得正确方向后就可以高效工作, 这里的正确的方向意味着最有可能实现目标的方向(术语就是最大化效果预期). 人工智能需要处理的任务包括学习、推理、规划、感知、语言识别和机器人控制等.

标志事件: AlphaGo 基于人工智能的增强技术和深度学习技术, 战胜了李世石. 该技术利用大量的数据来训练神经网络, 从而对新数据做出预测行为.

谷歌旗下的多款其他产品, 如地图、照片和邮箱等产品都开始利用这种技术增强用户体验; 谷歌翻译也将引入这种技术.

引入深度学习技术的机器翻译软件, 有望达到人工翻译的水准(AlphaGo 的改良版 AlphaGo Zero 使用了新一代的阿法元, 完全从零开始, 不需要任何历史棋谱的指引, 更不

需要参考人类任何的先验知识, 完全靠自己一个人强化学习和参悟, 棋艺增长远超阿法狗, 目前的战绩是百战百胜, 击溃阿法狗的记录是 100:0).

Google 翻译尝试在部分功能上使用深度学习技术,

例 1.7 利用摄像头即时取词翻译, 这相当于同声传译. 在国内, 微信、QQ 等都能取词翻译.

 **注 1.8.** 人工智能是一门专业性很强的计算机科学, 随着各大技术公司和学界的不懈努力, 这项技术未来会惠泽大众, 会给生活带来实实在在的好处.

对人工智能的常见误解

- ▶ 「人工智能是一个特定技术」. 例如在二十世纪八十年代到九十年代, 人们经常将人工智能与基于规则的专家系统混为一谈. 现在则是将人工智能经常会与多层卷积神经网络混淆. 这有点像把物理和蒸汽机的概念搞混了. 人工智能探究如何在机器中创造智能意识, 不是研究中产生的任何特定技术.
- ▶ 「人工智能是一个特定类别的技术方法」. 例如, 经常有人用符号化或逻辑化的方法将人工智能与「其他方法」相互比较, 如神经网络和遗传编程. 人工智能不是一种方法, 它是一个课题. 所有这些方法都是在对人工智能进行研究的产物.
- ▶ 「人工智能是一小群研究者的方向」. 这个误解与前几个错误有关. 一些作者使用「计算智能」来指代几个特定的研究者群体, 如研究神经网络, 模糊逻辑和遗传算法的研究者. 这是非常片面的, 这种分类方式让人工智能的研究陷入孤立境地, 研究成果不能得到广泛的注意.
- ▶ 「人工智能只是算法」. 严格说来不算是误解, 人工智能的确包含算法(也可粗略定义为程序), 它也包含计算机中其他的应用. 当然, 人工智能系统需要处理的任务相比传统算法任务(比如排序、算平方根)复杂得多.

综合各种不同观点, 仅从能力和学科两个方面讨论人工智能:

- ▶ 能力方面: 人工智能就是用人工的方法在机器(计算机)上实现的智能, 或称机器智能.
- ▶ 学科方面: 是一门研究如何构造智能机器或智能系统, 以模拟、延伸和扩展人类智能的学科.
- ▶ Turing 测试: 如图 1-5 所示. 能分辨出人和机器的概率小于 50%.

 **注 1.9.** Turing 测试存在的问题: 仅反映了结果的比较, 不涉及思维过程, 没指出是什么人.

—Turing 测试(图 AI:turingFig1)



图 1-5 图灵测试

1.1.7 人工智能的近远期目标

- ▶ 远期目标: 揭示人类智能的运行机理, 用智能机器去模拟、延伸和扩展人类的智能. 涉及到脑科学、认知科学、计算机科学、系统科学、控制论等多种学科的共同发展.
- ▶ 近期目标: 研究如何使现有的计算机更聪明, 即使它能够运用知识去处理问题, 能够模拟人类的智能行为.
- ▶ 相互关系: 远期目标为近期目标指明了方向. 近期目标则为远期目标奠定了理论和技术基础.

1.1.8 人工智能对计算机科学的影响

人工智能时代的来临, 给计算机体系结构的创新带来了新的黄金时代, 挑战的同时也要抓住机遇.

在过去的 50 年间, 人工智能的发展经历了三次起伏, 其中的三大关键因素便是**算法、大数据和算力**. 而在当下的第三次浪潮中, 软件和硬件的融合成为新的趋势, 其中 AI 芯片更是成为了此次浪潮中极为重要的因素. 传统意义上的软件公司如 Facebook、亚马逊,

以及中国的互联网企业都开始涌入这一领域。这一形势下，呈现出百花齐放的形势，让硬件研发存在更多的可能性。具体而言，可以探索的三个大方向：

一、(异构计算)。AI 时代没有哪一个单独的芯片能够做到一统天下，如 CPU、GPU、FPGA 这些不同的通用芯片，各有其优势，未来可能需在同一个平台上使用不同的芯片。研究异构计算方法，提高高性能分布式系统的训练效率。

二、(专用芯片)。从谷歌在 2017 年发布的第一款 ASIC TCP——TPU，到阿里前不久发布的平头哥含光 900 芯片，这些专用的芯片相比通用芯片的优势非常明显。都是为特定的场景定制，功耗低、成本低以及性能优。

三、(围绕存储做新架构探索)。AI 时代对存储的带宽容量的要求非常高，计算过程中将数据从存储搬到计算单元、再从计算单元搬回存储中所需的容量和性能损失远远超过做计算本身。因此，当下 AI 硬件创新面临的一个很重要的挑战，就是存储墙。有两个探索方向：

方向一是利用 3D 堆叠技术解决未来计算中的存储带宽问题。AMD 于 2015 年发布的 Fiji 核心便是这一思路的产物，其可以直接用 Fiji GPU 来加速 DNN，性能大幅提高。现在，几乎所有 AI 芯片公司都会沿用这一思路，考虑在芯片训练中使用这种 HBM(高带宽存储器)。

方向二是计算存储一体化，这是未来有可能改变传统的把计算和存储分开的冯·诺依曼架构的一个方向，并且这种方法不仅仅改变计算机体系结构，还能在材料、底层半导体技术上做更新。其中一个比较有趣的工作便是 ReRAM 技术，能够让计算和存储发生在同一个地方，而不需要做数据的搬移，节省的能耗和提高的性能非常多。

AI 时代，可以探索到计算机体系结构前沿主题，从底层看，可以尝试异构计算、3D 堆叠以及计算存储一体化等；往上看，则是在应用层做创新。

AI 计算系统体系提出了一致的愿景：未来的 AI 计算应该是分等级的分布式计算系统，即从云到边缘设备再到终端设备，让不同等级的数据在不同的地方进行计算和处理，从「AI in Cloud」变成「AI Everywhere」。而要实现这一愿景，还面临着一个难题：计算需求和功耗受限之间的矛盾。具体到 AI 芯片设计上，则主要有以下三个主要的挑战：

1) 来自于可编程能力。也就是说，随着算法的演进，AI 芯片能够通过编程得到持续改进。

2) 来自于 AI 芯片落地应用处理任务时，不仅需要处理神经网络，还需要处理大量常规的计算或经典的信号处理计算。

3) 能耗问题，尤其是对于边缘设备或者物联网设备而言，能耗问题非常突出。

为了实现这种超高能效、可编程又具有灵活性的计算，在过去六七年的时间里已经有了相当多的工作，主要沿着两个方向努力：一个方向是算法方向，对神经网络模型进行

剪枝、压缩、量化、低位宽化; 另一个方向则是在领域专用的体系结构上的探索, 包括数据粒度、编程和存储模型等.

针对体系架构 (尹首一教授等一系列工作), 其中就包括采用基础的可重构计算架构来做 AI 计算芯片, 主要从 MAC 单元、PE 及 PE 阵列架构三个层面上实现了硬件的可重构能力. 采用这种架构设计出来的芯片, 不仅能够支持灵活的、不同神经网络的编程, 还能极大地降低能耗. 下一阶段, 尝试实现可重构、可编程的体系结构和存储计算一体化 (In-Memory Computing) 的融合. 「这样才是一个将来能够真正把计算和能效继续推高, 把芯片的适用性和灵活性继续扩大的 AI 芯片解决方案. 」

现在的计算逐渐从云端走向边缘端, 然而边缘端的计算目前还存在很多问题: 一方面是移动设备「算不好」; 另一方面则是穿戴设备「算不了」. 而这些问题背后的原因主要还是边缘端的智能计算复杂度太高, 当前的芯片还无法满足这类边缘端计算的需求. 在过去很长一段时间, 国内学术界研究算法和研究硬件的人属于两个完全不同的领域, 各自「井水不犯河水」, 几乎很难一起做研究, 然而随着近几年来智能计算的发展, 尤其是深度学习模型对芯片架构提出了新的挑战和诉求, 计算和芯片二者在研究中结合得越来越紧密.

智能计算按应用领域分为云端和边缘端, 按任务可以分为训练和推理, 这四者可以组成四个象限: 云端训练、云端推理、边缘推理以及边缘训练. 而随着目前智能计算走过了从云端训练到云端推理、再到边缘推理的阶段, 下一步可以探索边缘训练, 特别是随着 5G 通信的到来, 将为这一方向的探索带来了更多的机会 (中国科学院自动化研究所南京人工智能芯片创新研究院常务副院长程健).

人工智能时代的到来使社会的生产要素发生了根本性变化. 生产力的三要素, 劳动者、劳动对象、劳动资料都在发生巨大变化, 这三要素都跟计算密切相关, 是人工智能计算是未来核心动力.

在农业时代、工业时代, 甚至信息时代, 劳动者没有发生太大的变化, 但是在智慧时代, 自然人和人工智能结合, 对劳动者的生产能力产生了极大的促进, 比如过去受自然环境的影响, 我们用人工去识别图像, 资料, 现在靠人工智能来做, 识别效率得到了成千上万倍的提升; 进入智慧时代, 数据成为重要的劳动对象, 以前的劳动对象, 是一种自然资源, 会在生产过程中, 消耗或转化, 而数据资源使用后仍然存在, 并且又生成了新的数据, 数据资源生生不息; 同时, 计算设备成为了新的劳动资料, 特别是人工智能时代, 劳动资料呈现指数级的需求.

“人工智能计算是未来核心动力, 代表着智慧计算的发展方向” (王恩东). 在人工智能计算中, 由于大场景、大计算需求越来越明显, 用通用芯片进行 AI 计算可能越来越不实用, 而更多的加速芯片会占据主流. 在这方面, 国内外很多的企业, 像谷歌、亚马逊、

Facebook, 中国的 BAT、浪潮, 都在 AI 领域不断的创新. 目前的 AI 计算服务, 一方面是以云的形式提供, 另一方面以物理服务器的形式提供.

目前, 浪潮在 AI 服务器领域, 中国市场占比超过 50%, 大部分 AI 企业使用浪潮服务器进行 AI 的深度学习和推理训练. 此外, 浪潮也在围绕 AI 算法优化、框架融合应用等方面持续创新.

人工智能推动了各个行业从信息化向智慧化升级, 提高了社会经济的效率, 并在多个行业引发了新一轮商业模式创新. 如在金融领域, 智能分析系统能够秒级完成人工一年 36 万小时的合同分析工作; 在制造领域, 一些大型的制造企业已经用智能机器人代替了 50% 以上的劳动力. 从宏观来看, 人工智能发展将成为中国经济增长的新引擎, 相关数据显示, 到 2035 年人工智能领域的经济总量在整体经济的占比将达到 20%.

1.1.9 计算机科学与技术、软件工程、物联网、大数据专业的区别

► 计算机科学与技术

计算机科学与技术专业是一个比较传统计算机类专业之一, 特点是注重基础知识结构的构建, 这个专业往往有较为全面的知识结构, 未来的就业面也相对更广一些, 目前很多 IT 行业的技术研发人员都是该专业毕业的. 不是该专业的人, 如果未来有明确的读研计划或者专业技能提升计划, 可以重点考虑一下计算机科学与技术专业, 未来在就业的方向上也会有更大的选择空间.

► 软件工程

软件工程专业比较注重学生动手实践能力的培养, 相对于计算机科学与技术专业来说, 软件工程专业的知识结构也有所调整, 会增加一部分关于软件项目管理方面的课程, 更偏向于软件方面的知识. 软件工程专业近些年来的就业情况非常不错, 这与当前软件行业的快速发展有较为直接的关系.

► 物联网

物联网专业的知识结构分为三大部分. 一、计算机硬件知识; 二、软件开发知识; 三、网络知识. 由于早期物联网领域的产业规模相对较小, 所以早期物联网专业的就业情况不太理想, 很多这个专业的人会选择软件开发领域的相关岗位. 但是, 随着 5G 的落地应用, 物联网未来的发展前景将非常广阔, 所以目前物联网专业也是热门专业之一.

► 大数据

大数据专业是新开设的专业之一, 大数据专业在知识结构上与其他传统计算机专业的差别还是比较明显的, 重点增加了统计学相关知识, 同时还会增加一些行业领域的专业知识, 比如经济学、社会学、医学等, 不同高校会根据自身的实际情况

况来设置具体的课程体系。

- ▶ 催生的新行业智能制造工程技术人员、工业互联网工程技术人员、虚拟现实工程技术人员、人工智能训练师、连锁经营管理师、供应链管理师、网约配送员、电气电子产品环保检测员、全媒体运营师、健康照护师、呼吸治疗师、出生缺陷防控咨询师、康复辅助技术咨询师、无人机装调检修工、铁路综合维修工和装配式建筑施工员等 16 个新职业 (2020 年 3 月, 人力资源和社会保障部与市场监管总局、国家统计局联合向社会发布).

1.1.10 2020 年十大科技趋势预测——百度研究院

- ▶ 趋势 1: AI 技术已发展到可大规模生产的工业化阶段, 2020 年将出现多家"AI 工厂", 运用在各行各业帮助产业升级;
- ▶ 趋势 2: 2020 年将会是 AI 芯片大规模落地的关键年;
- ▶ 趋势 3: 开源深度学习平台降低了技术开发门槛, 提高了 AI 应用的质量和效率, 深度学习技术将深入渗透到产业层面, 开始大规模应用;
- ▶ 趋势 4: 自动机器学习 AutoML 将大大降低机器学习的门槛;
- ▶ 趋势 5: 多模态深度语义理解技术进一步成熟并将得到更广泛应用;
- ▶ 趋势 6: 自然语言处理技术将与知识深度融合, 面向通用自然语言理解的计算平台得到广泛应用;
- ▶ 趋势 7: 物联网将在边界、维度和场景三个方向形成突破 (突破云计算中心的边界, 向万物蔓延. 物理世界的时间和空间维度将被拓展. 物联网与能源、电力、工业、物流、医疗、智慧城市等更多场景发生融合);
- ▶ 趋势 8: 智能交通将加速在园区、城市等多样化场景中的落地;
- ▶ 趋势 9: 区块链技术将以更加务实的姿态融入更多场景, 围绕区块链构建的数据确权、数据安全、数据使用、数据流通和交换等解决方案将在各行各业发挥巨大的作用;
- ▶ 趋势 10: 量子计算将迎来新一轮爆发, 为 AI 和云计算注入新活力.

2019 年 3 月召开的中央全面深化改革委员会第七次会议上, 审议通过了包括《关于促进人工智能和实体经济融合的指导意见》在内的八份文件, 是国家层面促进人工智能发展的又一份重要指导文件. 会议指出, 促进人工智能和实体经济深度融合, 要把握新一代人工智能发展的特点, 坚持以市场需求为导向, 以产业应用为目标, 深化改革创新, 优化制度环境, 激发企业创新活力和内生动力, 结合不同行业、不同区域特点, 探索创新成果应用转化的路径和方法, 构建数据驱动、人机协同、跨界融合、共创分享的智能经济形态.

“智能+”被写入政府工作报告，人工智能技术对社会的赋能被进一步提高。在工业经济由数量和规模扩张向质量和效益提升转变的关键期，“智能+”的理念给人工智能等数字技术提供了最广阔的落地空间和回报想象。通过智能化手段把传统工业生产的全链条要素打通，可以更好地推动制造业的数字化、网络化和智能化转型，更能反向助推技术自身的迭代和进步。

- 国产芯片和人工智能双突破

2019年8月，清华天机AI芯片登上《自然》封面，这也是中国芯片和人工智能领域第一次登上《自然》杂志。作为全球首款异构融合类脑芯片，它让自行车实现“无人驾驶”。据报道，天机芯片采用多核架构，有多个高度可重构的功能性核，可以同时支持机器学习算法和类脑电路。从长远来看，以通用人工智能为目标的“天机芯”，如果真能实现自己的理想，它将“无所不能”，可用于各行各业。

- 人脸识别纠纷走上法庭

2019年9月初，一款名为“ZAO”的APP火遍全网，只需在APP中上传一张照片，就能将自己的脸替换成大明星，效果几乎以假乱真。不过很快，这款APP就因为涉嫌侵犯用户隐私受到争议，用户上传脸部信息后，将面临侵权、盗刷等安全风险。另一边，11月，由于被强制要求刷脸入园，浙江理工大学特聘副教授郭兵把杭州野生动物世界告上法庭，这也成为国内人脸识别第一案。

- 新方法让机器人学习提速

2019年12月，《自然》评出2019年度十大杰出论文，其中包括苏黎世联邦理工学院用数据驱动的方法设计机器人软件来训练四足机器人，大大提高了机器人的运动能力和学习速度。新的训练方法利用强化学习，使机器人学习的速度提升了1000倍，动作灵活性和速度都大幅增强，而且踢不倒，甚至可以从跌倒中翻身站起。四足机器人在实验室中的小跑速度已经提升了25%，在被推倒或滑倒后，也能获得平衡，重新站稳。

- 人脸识别国标开跑

人脸识别技术也在争议中探索着符合用户期待又不损害技术发展的治理模式。在全国信标委生物特征识别分技术委员会换届大会上，人脸识别技术国家标准工作组正式成立，人脸识别国家标准制定工作全面启动。随着人脸识别技术的广泛应用，引发的问题也不可忽视，如技术精度等性能标准缺乏导致的仿冒身份、用户授权被盗用等使用安全问题，人脸信息收集、存储、处理等使用规范欠缺导致的信息泄露安全问题，数据滥用、隐私保护缺乏规范等伦理问题。公众也越发关注该技术在安全性上面临的挑战。

- 人工智能和宇宙

《生命3.0》让我们对人工智能和宇宙有了一个新的认识。作者是美国迈克斯·泰格马克，麻省理工学院物理系终身教授、未来生命研究所创始人。

例 1.10 你说一束光有目标吗？

看起来没有，向什么方向发射，光就奔向哪个方向啊，它自己能有啥目标？但是，你知道一个现象，光线射入水中会发生折射。光的路径不是一条直线，而是一个折线。为什么会这样呢？计算之后发现，因为这样速度最快。你可以想象一个救生员，救生员在海滩上奔跑的速度快，游泳的速度慢，所以，如果他要营救一个落水的游客，最佳路线不是直直地过去，而是在海滩上多跑一段，在水里少游一段，这样虽然看起来走了冤枉路，但是总体花的时间最短。

同样道理，光在空气中传播得快，在水中传播得慢，所以光会先在空气中多走一段，再射到水中，这样，它到达目的地的速度更快。你看，奇怪吧。光没有生命，但是它居然会有自己的行动原则，用最短时间从这一点到达那一点。所以，光的目标是什么？那就是在最短时间内到达目的地。



图 1–6 生命 3.0



图 1-7 营救溺水者的最佳路径和光的折射

例 1.11 那整个宇宙有没有这样的目标呢？

热力学第二定律说，就是追求“熵”的最大化。简单说就是整个宇宙越来越无序、越来越混乱。最后整个宇宙归于“热寂”。

注 1.12. 就是万事万物都扩散成一种无聊而又完美的均质状态。所有的星星都会熄灭，所有的秩序都会变成混乱，所有的地方温度都一样，再也没有什么值得流动的了，整个宇宙寂静了。所以，可以说宇宙的“目标”，就是达到热寂。

例 1.13 超级人工智能时代，人类来提供目标。

比如，我们人类说，地球太小了，我们要更大的生存空间，好，人工智能就会造一个以恒星为动力源的戴森球。比如，人类说，我想探索真理，我们想把圆周率尽可能地算下去，看看它到底有没有尽头，人工智能就干脆熄灭一个星系的能量，给人类算圆周率。人类说，我要星际旅行，人工智能就帮你完成这个目标，过程中耗掉无穷无尽的能量。这也是一个合作，人类提供意义和目标，人工智能提供能力和方法。这一合作，宇宙走向热寂的大目标就能越来越快地达成。

“发展中的问题，发展中解决”。

注 1.14. 人工智能的终极目标：机器 \equiv 人。

1.1.11 美国人工智能计划：2020 年度报告

2020 年 2 月 26 日，美国白宫科技政策办公室发布《“美国人工智能计划”：首个年度报告》（下称《报告》）。《报告》由美政府首席技术官克瑞西奥斯与副首席技术官帕

克联合签发, 主要从投资 AI 研发、共享 AI 资源、消除 AI 创新障碍等方面, 总结了美政府过去一年在实施“美国人工智能计划”方面取得的重大进展.

《报告》指出, 美政府拥有大量数据、模型及计算资源等重要 AI 资源. 美政府努力使各界获得高质量、可追溯的 AI 资源, 支持 AI 研发, 主要举措包括: 2019 年 12 月, 美政府发布《联邦数据战略 2020 年行动计划》, 指导各机构开发工具、完善流程, 高效使用数据. 美国国家科学基金会与 DARPA 合作开展“实时机器学习”项目, 探索可在连续数据流中学习的硬件与机器学习架构; 与联邦公路管理局合作研究隐私技术, 寻求远程访问高速公路数据库; 资助加利福尼亚大学及华盛顿大学开发并运营“云银行”(CloudBank), 为计算机科学人员提供公共云. 美国能源部为国家癌症研究所提供超级计算机, 开发治疗癌症的 AI 解决方案. 三、消除 AI 创新障碍《报告》认为, AI 仍然处于起步阶段. 美政府发布多项文件以指导制定相关法律、法规及标准, 消除 AI 创新障碍, 快速推进 AI 发展, 包括:

① 2019 年 4 月, 美国食品药品监督管理局提出针对医疗 AI 软件的监管框架, 确保这些软件安全有效.

② 2019 年 8 月, 美国国家标准技术研究院发布《联邦政府参与开发技术标准与相关工具的计划》, 强调美政府应持续参与 AI 标准开发活动, 推进可信任 AI 技术发展.

③ 2020 年 1 月, 美国运输部发布《确保美国在自动驾驶汽车技术中的领导地位: 自动驾驶汽车 4.0》文件, 详细阐述了美政府在自动驾驶汽车领域的若干原则. 2020 年 1 月, 美政府发布监管 AI 的 10 项原则, 包括公众信任、公众参与、科学诚信与信息质量等.

1.2 AI 的产生与发展

50 多年来, 人工智能走过了一条起伏和曲折的发展道路. 回顾历史, 可以按照不同时期的主要特征, 将其产生与发展过程分为 6 个阶段.

- ▶ 孕育期 (1956 年以前);
- ▶ 形成期 (1956—1970 年);
- ▶ 知识应用期 (1970—20 世纪 80 年代末);
- ▶ 从学派分离走向综合 (20 世纪 80 年代末到本世纪初);
- ▶ 智能科学技术学科的兴起 (21 世纪初以来);
- ▶ 深度学习为代表的人工智能技术 (2010 年以来).

1.2.1 AI 的产生与发展-孕育期 (1956 年以前)

AI 的产生与发展-孕育期 (远古——1946 年以前) 自远古以来, 人类就有用机器代替人们进行劳动的构想: 公元前 900 多年我国有歌舞机器人流传的记载; 公元前 850 年古希腊有制造机器人帮助人们劳动的神话传说; 三国时的木牛流马.

- ▶ 亚里斯多德 (Aristotle, 公元前 384——公元前 322): 古希腊伟大的哲学家和思想家, 创立了演绎法. 他提出的三段论至今仍然是演绎推理的最基本出发点.
- ▶ 莱布尼茨 (G. W. Leibnitz, 1646——1716): 德国数学家和哲学家, 把形式逻辑符号化, 奠定了数理逻辑的基础.
- ▶ 图灵 (A. M. Turing, 1912——1954): 英国数学家, 1936 年创立了自动机理论, 自动机理论亦称图灵机, 是一个理论计算机模型.
- ▶ 莫克利 (J. W. Mauchly, 1907——1980): 美国数学家、电子数字计算机的先驱, 与他的研究生埃克特 (J. P. Eckert) 合作, 1946 年研制成功了世界上第一台通用电子计算机 ENIAC.

AI 的产生与发展-孕育期 (1956 年以前)

- ▶ 麦克洛奇 (W. McCulloch) 和皮兹 (W. Pitts): 美国神经生理学家, 于 1943 年建成了第一个神经网络模型 (MP 模型).
- ▶ 维纳 (N. Wiener, 1874—1956): 美国著名数学家、控制论创始人. 1948 年创立了控制论, 使得控制论向人工智能的渗透, 形成了行为主义学派. 1954 年, 钱学森在 McGraw-Hill 出版了《工程控制论》.
- ▶ 图灵又于 1950 年, 发表题为《计算机能思维吗?》的著名论文, 明确提出了“机器能思维”的观点.

 **注 1.15.** 在人工智能诞生之前, 一些著名科学家就已经创立了数理逻辑、神经网络模型和控制论, 并发明了通用电子数字计算机. 为人工智能的诞生准备了必要的思想、理论和物质技术条件.

1.2.2 AI 的产生与发展-形成期 (1956—1970 年)

- AI 诞生于一次历史性的聚会.
- ▶ 时间: 1956 年夏季;
 - ▶ 地点: 达特茅斯 (Dartmouth) 大学;
 - ▶ 目的: 为使计算机变得更“聪明”, 或者说使计算机具有智能.
 - ▶ 发起人:

- 麦卡锡 (J. McCarthy), Dartmouth 的年轻数学家、计算机专家, 后为 MIT 教授.
 - 明斯基 (M. L. Minsky), 哈佛大学数学家、神经学家, 后为 MIT 教授.
 - 洛切斯特 (N. Lochester), IBM 公司信息中心负责人.
 - 香农 (C. E. Shannon), 贝尔实验室信息部数学研究员.
- 参加人:
- 莫尔 (T. More)、塞缪尔 (A. L. Samuel), IBM 公司.
 - 塞尔夫里奇 (O. Selfridge)、索罗蒙夫 (R. Solomonoff), MIT.
 - 纽厄尔 (A. Newell), 兰德 (RAND) 公司.
 - 西蒙 (H. A. Simon), 卡内基 (Carnegie) 工科大学.
- 会议结果:
- 由麦卡锡提议正式采用了“Artificial Intelligence”这一术语. AI 以“推理、知识”为重点.
- 心理学小组
- 1957 年, 纽厄尔、肖 (J. Shaw) 和西蒙等人的心理学小组研制了一个称为逻辑理论机 (Logic Theory Machine, 简称 LT) 的数学定理证明程序.
 - 1960 年研制了通用问题求解 (General Problem Solving) 程序. 该程序当时可以解决 11 种不同类型的问题, 如不定积分、三角函数、代数方程、猴子摘香蕉、河内梵塔、人—羊过河等.
- IBM 工程小组
- 1956 年, 塞缪尔在 IBM704 计算机上研制成功了具有自学习、自组织和自适应能力的西洋跳棋程序. 这个程序可以从棋谱中学习, 也可以在下棋过程中积累经验、提高棋艺. 通过不断学习, 该程序 1959 年击败了塞缪尔本人, 1962 年又击败了一个州的冠军.
- MIT 小组
- 1958 年, 麦卡锡建立了行动规划咨询系统.
 - 1960 年, 麦卡锡又研制了人工智能语言 LISP.
 - 1961 年, 明斯基发表了“走向人工智能的步骤”的论文, 推动了人工智能的发展.

其他方面

- ① 1965 年, 鲁宾逊 (J. A. Robinson) 提出了归结 (消解) 原理.
- ② 1965 年, 费根鲍姆 (E. A. Feigenbaum) 开始研究化学专家系统 DENDRAL.

1.2.3 AI 的产生与发展-知识应用期 (1971–80 年代末)

失败的预言 60 年代初, 西蒙预言: 10 年内计算机将成为世界冠军、将证明一个未发现的数学定理、将能谱写出具有优秀作曲家水平的乐曲、大多数心理学理论将在计算机上形成并被验证.

挫折和教训

- ▶ 在博弈方面, 塞缪尔的下棋程序在与世界冠军对弈时, 5 局败了 4 局.
- ▶ 在定理证明方面, 发现鲁宾逊归结法的能力有限. 当用归结原理证明两个连续函数之和还是连续函数时, 推了 10 万步也没证出结果.
- ▶ 在问题求解方面, 对于不良结构, 会产生组合爆炸问题.
- ▶ 在机器翻译方面, 发现并不那么简单, 甚至会闹出笑话. 例如, 把“心有余而力不足”的英语句子翻译成俄语, 再翻译回来时竟变成了“酒是好的, 肉变质了”.
- ▶ 在神经生理学方面, 研究发现人脑有 10^{11-12} 以上的神经元, 在现有技术条件下用机器从结构上模拟人脑是根本不可能的.
- ▶ 在其它方面, 人工智能也遇到了不少问题. 在英国, 剑桥大学的詹姆教授指责“人工智能研究不是骗局, 也是庸人自扰”. 从此, 形势急转直下, 在全世界范围内人工智能研究陷入困境、落入低谷.

以知识为中心的研究:

- ▶ 专家系统实现了人工智能从理论研究走向实际应用, 从一般思维规律探讨走向专门知识运用的重大突破, 是 AI 发展史上的一次重要转折.

2020 年 3 月, 日本日立钢机集团旗下的 Doctor-NET (东京港区) 将启动人工智能检测新型冠状病毒的实证试验. 该公司将与中国的 AI 开发企业北京推想科技合作, 向日本引进在武汉等地使用的新冠肺炎诊断系统.

- ▶ 1972 年, 费根鲍姆开始研究 MYCIN 专家系统, 并于 1976 年研制成功. 从应用角度看, 它能协助内科医生诊断细菌感染疾病, 并提供最佳处方. 从技术角度看, 他解决了知识表示、不精确推理、搜索策略、人机联系、知识获取及专家系统基本结构等一系列重大技术问题.
- ▶ 1976 年, 斯坦福大学的杜达 (R. D. Duda) 等人开始研制地质勘探专家系统 PROSPECTOR.

这一时期, 与专家系统同时发展的重要领域还有计算机视觉和机器人, 自然语言理解与机器翻译等. 但直到 80 年代末, 所有的 AI 程序都只是“玩具”.

新的问题: 专家系统本身所存在的应用领域狭窄、缺乏常识性知识、知识获取困难、推理方法单一、没有分布式功能、不能访问现存数据库等问题被逐渐暴露出来.

1.2.4 AI 的发展-知识深化期 (1990-2000 年代末)

以“特征”为重点

① 1993 年, 支持向量机出现。国内的王国俊教授提出了三 I 方法。由于模糊推理与模糊逻辑在模糊控制中有直接的应用, 模糊推理与模糊逻辑的研究受到模糊系统与人工智能学界的广泛关注。1997 年 3 月在美国召开的国际信息科学联合会议上, 王国俊作了“论模糊推理的逻辑基础”的报告。这一报告引起了很多与会代表的浓厚兴趣, 并受到前任国际模糊系统协会主席、纽约宾汉顿大学的 Klir 教授以及 Turksen 等著名教授在内的许多学者高度评价, 克雷顿大学模糊系统研究所所长 Mordeson 教授还在该校的研究报告中刊登王的全文。如今这一成果已全文发表于 *Information Sciences*。在此基础上, 王提出了模糊推理的三 I 方法, 于 1999 年发表于《中国科学》, 全面改进了传统的 CRI 方法。2004 年 7 月, 他在美国盐湖城召开的信息科学联合会议上当选为不确定性数学学会副理事长。

② 1992 年, 美国控制论专家 L. A. Zadeh 教授指出: 人工神经网络、模糊逻辑、遗传算法、蚁群算法与传统计算模式的区别, 在于它们与人脑对应, 具有在不确定及不精确环境中进行推理和学习的智能计算能力。除此之外, 粗糙集方法、机器学习方法以及各种随机搜索技术等也被纳入到人工智能的统一框架中, 人工智能技术已在众多应用领域取得了极大的成功。

③ 全蕴涵模糊推理机制与模糊规则的启发式生成方法。模糊控制系统的核心技术是模糊控制器的设计, 模糊控制器的主要组成部分是模糊推理机和模糊规则库。70 年代 L.A.Zadeh 教授提出的“复合蕴涵规则”方法虽得以广泛应用, 但模糊推理的基本形式 FMP 问题的合成推理方法求解使用的合成运算缺乏严格的逻辑依据, 模糊推理逻辑基础问题一直受到了科学家和工程师的极大关注, 同时也已取得了一系列有价值的研究成果。其中以我国前辈数学家王国俊教授提出的“全蕴涵模糊推理机制与推理方法”受到了最为广泛的关注。该推理方法在模糊推理的每一步运算过程都使用了蕴涵运算, 通过蕴涵算子给出模糊推理最优解的解析形式。基于不同的蕴涵方式, 有学者提出了反向模糊推理机制与模糊推理算法, 给出了模糊取式和模糊拒取式最优解的解析计算公式。

④ 模糊知识约简/模糊决策树生成模糊规则的启发式方法。1982 年 Z. Pawlak 教授提出的粗糙集理论在处理不完整、不一致和不精确数据的问题中取得成功, 目前已经发展成为一种较为成熟的定量分析数学工具, 在机器学习、数据挖掘、模式识别、故障诊断等领域有着广泛的应用。其主要思想是将模糊相似关系代替粗糙集的等价关系, 引入模糊上、下近似概念, 并提出模糊约简与模糊核的定义, 提出了一种启发式算法能够从初始模糊数据中学习模糊规则, 该算法从真正意义上实现了从模糊数据抽取模糊规则。项目进一步开展了模糊决策树归纳学习研究, 提出了一种产生模糊决策树的启发式算法, 将

粗集核学习与决策树归纳进行结合, 可生成高性能加权模糊规则.

⑤ 基于极限学习机或随机机会约束模型的智能学习方法. 极限学习机方法的前期工作仅适用于解决有监督的机器学习问题, 不能处理标签数据存在缺失的情形. 针对数据标签有缺失且带有分布流形特性的数据, 基于 Laplacian 矩阵的流形正则化思想学习数据的流形特性, 构造了两步骤学习的统一框架, 将有监督、无监督和半监督极限学习算法纳入其中. 进而项目研究了基于机会约束模型的有监督学习方法与半监督学习方法. 首先建立了基于随机机会约束的鲁棒支持向量非线性回归模型, 将随机约束优化问题转化为二次锥规划问题, 由内点法可实现高效求解, 解决了回归分析中输入数据和输出数据同时受到噪声干扰的问题.

1.2.5 数据智能——发展期 (2000-2009 年末)

2006 年, Geoffrey Hinton 提出深度学习, 使得神经网络得以重生. 关于人工智能的每一个进步都是由 30 年前的一篇阐述多层神经网络的训练方法的论文 (Learning representations by back-propagating errors, [RumelhartHinton1986-9415]) 演变而来, 它为人工智能在最近十年的发展奠定了基础, 但要保持这种进步, 就要面对人工智能严重的局限性.

多伦多市中心一栋高级大厦, 七层是新成立的人工智能研究所 Vector Institute 的所在地. 研究所的联合创始人乔丹·雅各布 (Jordan Jacobs). 该研究所于今年秋天正式成立, 致力于成为全球人工智能中心. 为了拜访杰弗里·辛顿 (Geoffrey Hinton) 来到多伦多. 他是“深度学习”之父, 正是这个技术让人工智能发展到今天这般炙手可热. 雅各布说: “我们 30 年后再往回看, 杰弗里就是人工智能 (我们认为深度学习就是人工智能) 的爱因斯坦.” 在人工智能领域最顶尖的研究人员当中, 辛顿的引用率最高, 超过了排在他后面三位研究人员的总和. 他的学生和博士后领导着苹果、Facebook 和 OpenAI 的人工智能实验室; 辛顿本人是谷歌大脑 (Google Brain) 人工智能团队的首席科学家.

事实上, 人工智能在最近十年里取得的几乎每一个成就, 包括语音识别、图像识别, 以及博弈, 在某种程度上都能追溯到辛顿的工作. Vector Institute 研究中心进一步升华了辛顿的研究. 在这里, 谷歌、Uber、Nvidia 等美国和加拿大的公司正努力将人工智能的技术商业化. 资金到位的速度比雅各布想象的更快; 他的两个联合创始人调研了多伦多的公司, 发现他们对人工智能专家的需求是加拿大每年培养的人数的 10 倍. 某种意义上, Vector 研究所是全球深度学习运动的原爆点: 无数公司靠这项技术牟利, 训练它、改进它、应用它. 到处都在建造数据中心, 创业公司挤满了摩天大楼, 整整新一代学生也纷纷投身这一领域.

1.2.6 数据智能——爆发期 (2010-现在)



图 1-8 神经网络的发展历史

◆ 2012 年, 深卷积神经网络错误率下降到了 16%; 在接下来的几年中, 错误率下降了几个百分点. 虽然 2012 年的突破是“前所未有的组合”, 但大幅量化的改进标志着全行业人工智能繁荣的开始. 到 2015 年, 研究人员报告说, 软件在狭窄的 ILSVRC 任务中超出了人类能力. 然而, 作为挑战组织者之一的 Olga Russakovsky 在 2015 年指出, 这些计划只需将图像识别为属于千分之一的图像; 人类可以识别更多的类别, 并且(不像程序)可以判断图像的上下文.

 **注 1.16. ILSVRC (ImageNet Large Scale Visual Recognition Challenge)** 是近年来机器视觉领域最受追捧也是最具权威的学术竞赛之一, 代表了图像领域的最高水平.

ImageNet 数据集是 ILSVRC 竞赛使用的是数据集, 由斯坦福大学李飞飞教授主导, 包含了超过 1400 万张全尺寸的有标记图片. ILSVRC 比赛会每年从 *ImageNet* 数据集中抽出部分样本, 以 2012 年为例, 比赛的训练集包含 1281167 张图片, 验证集包含 5E4 张图片, 测试集为 1E5 张图片.

◆ 2014 年, 超过 50 家机构参加了 ILSVRC. 2015 年, 百度科学家因使用不同帐户而被禁止使用一年, 大大超过每周两次提交的指定限制. 百度后来表示, 它解雇了涉及的团队领导, 并建立了一个科学咨询小组.

◆ 2016 年, AlphaGo 在一场复杂的围棋比赛中打败了世界冠军, 震惊了计算机科学家们. 它背后的 DeepMind 团队中的重要成员曾是特南鲍姆的博士后. 特南鲍姆参与了一家创业公司的工作, 这家公司致力于让自动驾驶汽车直观地了解一些基础物理学, 对其

他驾驶员的意图也能做出一定的直觉判断,从而更好地应对从未遇到过的情况,比如一辆卡车冲到前面或他人强行超车.

◆ 2016 年,以“学习”为重点,出现了[ImageNet 数据库](#). ImageNet 项目是一个用于视觉对象识别软件研究的大型可视化数据库. 超过 1400 万的图像 URL 被 ImageNet 手动注释,以指示图片中的对象;在至少一百万个图像中,还提供了边界框. ImageNet 包含 2 万多个类别;一个典型的类别,如“气球”或“草莓”,包含数百个图像. 第三方图像 URL 的注释数据库可以直接从 ImageNet 免费获得;但是,实际的图像不属于 ImageNet. 自 2010 年以来,ImageNet 项目每年举办一次软件比赛,即 ImageNet 大规模视觉识别挑战赛(ILSVRC),软件程序竞相正确分类检测物体和场景.ImageNet 挑战使用了一个“修剪”的 1000 个非重叠类的列表. 2012 年在解决 ImageNet 挑战方面取得了巨大的突破,被广泛认为是 2010 年的深度学习革命的开始. 辛顿在 1986 年取得了突破,他发现反向传播可以用来训练深度神经网络,即多于两层或三层的神经网络. 但自那以后又过了 26 年,不断增强的计算能力才使这一理论得以证实. 辛顿和他在多伦多的学生于 2012 年发表的一篇论文表明,用反向传播训练的深度神经网络在图像识别领域打败了当时最先进的系统——“深度学习”终于面世. 神经网络似乎能抓取图像、文字、某人说话的录音、医疗数据等事物,将它们放到数学家所说的高维矢量空间里,使这些事物之间的距离远近反映真实世界的一些重要特点. 辛顿相信,这就是大脑的运作方式.

◆ 2017 年,38 个竞争团队中有 29 个错误率低于 5%.

◆ 2017 年,ImageNet 宣布将在 2018 年推出一项新的,更加困难的挑战,其中涉及使用自然语言对 3D 对象进行分类. 由于创建 3D 数据比注释预先存在的 2D 图像更昂贵,数据集预计会更小. 这方面的进展应用范围从机器人导航到增强现实.

◆ 2017 年 11 月前后,谷歌的 AutoML 项目发展出新的神经网络拓扑结构,创建了 NASNet,这是一个针对 ImageNet 和 COCO 优化的系统. 据 Google 称, NASNet 的性能超过了以前发布的所有 ImageNet 性能.

◆ 2018 新一代人工智能院士高峰论坛,张正友博士加入腾讯并创建 Robotics X 机器人实验室,他对未来的判断是我们将迎来一个人机共生的时代. 当计算技术、感知技术得到充分应用后,人机协作、人机共生将有长足的发展. Robotics X 机器人有 6 个组成部分,本体、感知、执行器、动力系统、交互系统、决策. 机器人的未来趋势是自动化、智能化,要在不确定的环境中自主决策. SLAP 范式(SLAP 打耳光理论)的提出,解决机器人的自主决策问题;传感器和执行器的紧密结合,在学习和计划模块的帮助下提升能力、做出决策.

自主分为“反应式自主”和“有意识的自主”. 张正友博士提出了“SLAP 打耳光理论”实现自主:第一步,让感知(Sense)和行动(Act)相连,先得到反应式自主. 在上

层规划后, 做到长期自主去规划 (Plan), 就是有意识的自主.

第二步, 整个过程中, 机器人还需要不断学习 (Learn), 在学习中与外界交互, 与自身交互, 让机器能力越来越强. 这个 SLAP 范式, 在英文里也有打耳光的意思, 这就是它名字的由来.

当机器人进过 SLAP 范式进化后, 智能化程度变高, 就有丰富的应用场景, 从操作复杂工艺, 到陪伴和护理儿童与老年人, 到更复杂的人机合作.

机器人本体的研究有六大趋势: 仿生化, 灵巧操控, 触觉基础, 多机器人协同, 人机协同以及医疗辅助. 在技术达到人机协同的水平之前, 还有很多的技术研发需求, 技术突破点包括人工智能技术、机器人本体、自动控制、进化学习、情感理解、灵活操控、守护人类, 这对更先进、更智慧的机器人提出了要求. 最终目标是机器人要服务于人.

迁移学习和联邦学习 (香港科技大学教授杨强) 的提出是针对现代组织机构数据多且是小数据库或者是某一种任务大数据且另一任务小数据情形的智能化学习方案.

第一种方案是迁移学习, 希望像人类一样把以往的任务中学习到的技能运用在新任务中. 迁移学习还可以兼顾可靠性和隐私安全问题. 迁移学习把源领域的模型和任务迁移到新领域, 但迁移学习的本质是找出学习中存在的不变量. 迁移学习中, 定量分析表明模型的浅层比较容易迁移, 理论分析结果可以帮助我们更好地做迁移学习. 例子比如卫星图像识别, 贷款风控不同用户类别间的迁移, 推荐系统的策略迁移, 舆情分析中的迁移学习.

第二种方案是联邦学习, 它的目标是解决有许多不同的领域, 而每个领域都只有小数据要如何建立模型的问题. 这种方法也引起了很多金融企业的兴趣, 数据可以不离开本地的数据库, 正因为联邦学习的学习过程不需要大量的数据交换. 联邦学习有两种模式, 纵向联邦学习, 数据中有部分数据特征是同样的, A 方和 B 方都持有模型的一部分, 通过动态加密技术传递重要的参数; 第二种模式, 横向联邦学习, 在用户端更新模型并上传, 云端服务器根据一定的策略统一更新用户模型. 未来可以形成数据联盟, 让各方都受益.

国内的人工智能领域这几年取得蓬勃的发展. 然而短板也很明显, 如人工智能基础理论和原创算法差距较大、关键部件基础薄弱、高水平人才不足等, 关键的一点是, 尚未形成具有国际影响力的人工智能开源开放平台. 为此, 在科技部发起了新一代人工智能产业创新战略联盟 (ATSA) 组织、产学研用的通力协作下, 我国正式推出新一代人工智能开源开放平台——启智 (英文名称 OpenIntelligence, 简称 OpenI), 以促进人工智能领域的开源开放协同创新, 构建 OpenI 的技术链、创新链和生态链, 进而推动人工智能产业健康快速发展及其在社会经济各领域的广泛应用. 人工智能开源开放平台已在 2018 年 3 月 31 日取得开源许可证, 前期参与的单位包括北京大学、国防科技大学、北京航空航天

大学、华为、百度、阿里、腾讯、讯飞、商汤、微软、Intel、NVIDIA 等,在未来有望打造出一个学术机构、商业实体、自然人或任何其他法人等共建共享的开源软件开源硬件开源数据超级社区.

OpenI 启智平台基础设施及环境建设已经在万科云城的鹏城实验室 AI 中心大楼启动,建设内容包括 AI 超算、AI 研究中心、OpenI 启智深圳平合、OpenI 智源深圳社区和新一代人工智能产业创新联盟「启智空间」(含 AVS2 超高清影音中心、「启智未来 4k 超高清 VR 直播间、OpenI 门户网站、「源智造」流水线-AI 开发基础设施建设及系列推广活动、「智源」社区公号、「启智」官微抖音号等).

5G 是为非人类的使用而设计,其带来的更大的挑战包括万物互联、大数据应用以及新业务模式等,而「云端机器人」则是 5G 的「杀手级应用」,其需要的带宽是人类的 100 倍. 基于这种理念,黄晓庆博士将其创立的达闼科技定位为「服务云端机器人」,致力于通过公共基础设施创造有效的机器人网络,将云端机器人租给用户使用(华中科技大学电子信息与通信学院院长黄晓庆博士,《5G 时代的云端智能机器人发展》). 机器人通过云端连接所存在的安全隐患,为此,他认为应该利用新型网络架构和能力来将机器人与互联网进行隔离,一种方式是通过 5G 网络切片技术构建安全可靠的「云端机器人神经网络」,从而实现机器人的可控;另一种方式则是采用边缘云 (Edge Cloud) 的方式,将云端智能推理能力分布在 5gMEC 服务器上,形成一个类似于 CDN(内容分发网络) 的 IDN(Inference Distribution Network).

在「云端机器人」的产品化方面,达闼科技做的第一个项目是云端导盲机器人,不过这款产品还没来得及量产;另外在新零售领域和营销宣传方面,达闼科技则正在跟运营商进行云端机器人 +5G 的探索和合作,并尝试利用 5G 实现云端智能的更广泛的应用,包括超声数据分析、超声操作指导、拉曼光谱分析以及拉曼数据收集等. 同时,其还在推进「XR-Plan」计划,致力于建立服务机器人的标准模型,计划研发 4 款关节灵活、行动平衡的标准服务机器人(3 轮人形机器人、4 轮车形机器人、4 足车形机器人、2 足人形机器人),来引导未来机器人的开发.

360 公司依托运动引擎(例: 扫地机器人)、交互引擎(例: 儿童手表)、视觉引擎(例: 家庭安防生态、内容安全审核等)和决策引擎(金融风控、广告等). 这些也成为支撑 360 公司 IoT 业务和互联网业务的核心技术. 单就 2018 年上半年的表现而言,360 安全大脑成功在恶意程序、钓鱼攻击、骚扰电话、垃圾短信和网络诈骗等问题的解决上均取得不俗的成果(360 人工智能研究院院长颜水成博士,《视觉智能: 从攻坚到闭环》). 360 近期在视觉智能领域的最新研究成果——Global Reasoning Unit. Global Reasoning Unit 将 5 个 1×1 的卷积以模块的形式插入任意网络做学习,在浅层网络就能对远处的目标进行识别,使跨区域进行信息交换成为可能. 相较于通过增加 depth 进行优化的方式,Global

Reasoning Unit 能有效提升现有网络的性能,因此在消费级智能设备的应用上指日可待.

2019 年 12 月,新一代人工智能院士高峰论坛,世界计算机科学最高奖图灵奖获得者 John Hopcroft 院士,丁文华、王立军、王沙飞、刘韵洁、吴建平、张东晓、陈杰、赵沁平、高文和蒲慕明等十余位国内智能科技领域的院士专家,商汤、百度、腾讯、阿里巴巴、旷视、华为和寒武纪等产业内一线企业的领军人物,以及北京大学、清华大学、中科院等学术研究机构,通过自身所在行业领域的优势,分享智慧,共谋发展.

在高峰论坛上,院士们将重点关注人工智能在开源开放平台、高端芯片、类脑研究、智能应用的动态,关注人工智能技术领域的行业变革与技术创新;现场深度点评并集中讨论国家新一代人工智能开放创新平台单位的 AI 创新应用成果,以及鹏城实验室主导建设的 AI 开源开放基础平台. 强调让 AI 领先技术走出实验室,赋能产业. 作为论坛的举办地,深圳将在交通、医疗产业领域,先行先试,打造具有国际竞争力的人工智能产业集群,建设优质 AI 生态圈.

今天的神经网络由大平面层组成,但人类新皮层的真实神经元不仅是水平分布成层的,还有垂直排列的. 辛顿认为,他知道这些垂直结构的作用,比如在人类视觉系统中,这些结构确保我们在视角变化时保持识别物体的能力. 因此,他正在搭建一个叫做“胶囊”(capsules) 的人工视觉体系来验证这个理论. 目前,他还没有成功;这些胶囊并没有显著提升神经网络的表现. 但是,他研究反向传播时也遇到了同样的情况,而且持续了近 30 年.

1.2.7 AI 的产生与发展-从学派分立到综合 (20 世纪 80 年代到本世纪初)

人工智能研究形成了三大学派:随着人工神经网络的再度兴起和布鲁克(R.A.Brooks)的机器虫的出现,人工智能研究形成了符号主义、连接主义和行为主义三大学派.

► 符号主义学派

是指基于符号运算的人工智能学派,他们认为知识可以用符号来表示,认知可以通过符号运算来实现. 例如,专家系统等.

► 连接主义学派

是指神经网络学派,在神经网络方面,继鲁梅尔哈特研制出 BP 网络之后,1987 年,首届国际人工神经网络学术大会在美国的圣迭戈(San-Diego)举行,掀起了人工神经网络的第二次高潮. 之后,随着模糊逻辑和进化计算的逐步成熟,又形成了“计算智能”这个统一的学科范畴.

► 行为主义学派

是指进化主义学派,在行为模拟方面,麻省理工学院的布鲁克教授 1991 年研制成功了能在未知的动态环境中漫游的有 6 条腿的机器虫.

► 三大学派的综合集成

随着研究和应用的深入,人们又逐步认识到,三个学派各有所长,各有所短,应相互结合、取长补短,综合集成.

1.2.7.1 人工智能的进化

- 弱人工智能: 完成某个特定的任务,如图像及语音识别、智能翻译、自动驾驶、Watson(IBM 的 AI 系统; 2015 年成立独立的 WatsonHealth 部门,收购多家医疗数据公司)、AlphaGo.
- 强人工智能: 具备思考、推理和解决问题的能力、快速学习、在非监督学习的情况下处理前所未见的操作.
- 超人工智能: 在几乎所有领域都比最聪明的人类大脑都聪明很多,包括科学创新、通识和社交技能.

AI 的产生与发展-智能科学技术的兴起 (本世纪初以来) 目前,一个以人工智能为核心,以自然智能、人工智能、集成智能为一体的新的智能科学技术学科正在逐步兴起,并引起了人们的极大关注.

人工智能研究的主要特征包括以下几个方面:

- 由对人工智能的单一研究走向以自然智能、人工智能、集成智能为一体的协同研究;
- 由人工智能学科的独立研究走向与脑科学、认知科学等学科的交叉研究;
- 由多个不同学派的独立研究走向多学派的综合研究;
- 由对个体、集中智能的研究走向对群体、分布智能的研究.

1.2.8 AI 成功的标志: IBM 的“深蓝”和“小深”

“深蓝”

- 对弈情况:
 - 时间: 北京时间 1997 年 5 月 12 日凌晨 4 点 50 分;
 - 对手: IBM 的“深蓝”超级计算机;
 - 国际象棋世界冠军卡斯帕罗夫; 结局: 2 胜 1 负 3 平, 总比分 3.5 : 2.5, “深蓝”获胜.
- 技术指标
 - 32 个 CPU, 每个 CPU 有 12 个协处理器, 每个 CPU 有 256M 内存, 每个 CPU 的处理速度为 200 万步/秒.

- 对弈的实质机器智能与人类智能的较量.
- “小深”
- 对弈情况:
 - 时间: 北京时间 203 年 1 月 26 日至 2 月 7 日;
 - 对手: 比 “深蓝” 功能强大的 “小深” 超级计算机, 国际象棋世界冠军卡斯帕罗夫;
 - 结局: 1 胜 1 负 4 平, 平局.

 **注 1.17.** 计算机可以有智能; 计算机要完全战胜人类象棋大师也非易事, 深度学习基本解决了这个问题.

1.2.9 编程语言

在 21 世纪初, 在 21 世纪初, 开发人员更喜欢用 C 语言进行系统编程, 用 JAVA 开发企业应用程序, 用 SaaS 进行分析, 用 MATLAB 进行科学计算. 然而, 今天的开发人员使用 Rust 进行系统编程, Go 进行企业开发, 使用 Python/R 进行分析, 并使用 Julia 进行科学计算 (Julia Computing 的首席执行官 Viral Shah).

 **注 1.18.** *Julia* 是一种高级动态编程语言, 旨在满足高性能数值分析和计算科学的需求, 而不需要快速单独编译, 通用编程也很高效. *Julia* 设计的独特之处包括: 1) 具有参数多态性的类型系统; 2) 完全动态编程语言中的类型多个调度作为其核心编程范例; 3) 允许并发、并行和分布式计算; 4) 无需胶水代码即可直接调用 C 和 Fortran 库. *Julia* 有垃圾回收机制, 包括用于浮点计算、线性代数、随机数生成、快速傅立叶变换和正则表达式匹配的高效库.

Julia 作为一门新兴语言, 它的版本后向兼容行还有待提高, 原生环境里面一个包都不给配置, 需一个个地下载. 选择 *JuliaPro* 这个集成环境 (主要集成了 Atom+juno (*julia* 的第三方 IDE)、jupyter notebook(浏览器端的编辑器)). *JuliaPro* 初始化就配置了好许多常用的包, 不用下载和配置各种路径. 图 1-9 给出了 *JuliaPro v1.0.5* 界面下, 宏包命令 `Pkg.add("TensorFlow")` 的执行和依赖情况.



图 1–9 Julia V1.0.5

注 1.19. 过去动态语言大部分都是面向解释器设计, 这使得动态语言非常简单易学. 但实际上经常用程序不需要这么动态的特性, *Julia* 平衡各种动态性, 去掉了一些不那么重要的动态性, 是面向 *JIT* 等优化技术设计的动态语言. *Julia* 加入了一些限制, 消除了 *Python* 和 *MATLAB* 的局限. *Julia* 可以调用大部分的 *R* 语言包. 虽然 *Julia* 学习曲线平滑, 但是想用 *Julia* 写出性能好, 抽象干净的代码是需要一定时间的. 只有在科学计算这个领域, *Julia* 目前才是更加合适的解决方案, 因为它为科学计算而生, 但是在其它领域 *Julia* 就几乎没有比较优势了. *Python* 里使用 *Julia* 可以使用 *pyjulia*. *Julia* 可以调用 *MATLAB* 语言, 详情见包 *MATLAB.jl* 包.



图 1–10 Julia 使用 Matplotlib

```

File Edit View Juno Selection Find Packages Help
Project
pycalltest.jl Tensorflow... Tensorflows... MNNet... MNNet.jl GFLtest.jl FluxTest.jl repl.jl Workspace
Filter
Main
Base Base
Core Core
n _in 1000
n _out 10
n H 100
InteractiveUtils InteractiveUtils
n N 64
v ans nothing
atexit_hook_copy > Vector{Any} with 0 elements
v data > Base.Iterators.Take(Base.I...
l loss > loss
n lr 0.0000100
v opt > Descent
l predict > predict
v ps Params[[0.071606 0.81157...
v start_time 2020-01-11T13:27:16.005
v total_time 1111 milliseconds
w1 > 1000x10 Array{Float64,2}
w2 > 1000x10 Array{Float64,2}
x > 64x1000 Array{Float64,2}
y > 64x10 Array{Float64,2}

[1] include("juliaFlux.jl")
[2] include("relative(Module, String)") at <\Loading.jl>:1844
[3] include!(Module, String) at <\ying.jl>:29
[4] include!(Module, String) at <\ying.jl>:29
[5] in expression starting at C:\Users\oucnh\JuliaPro\JuliaPro_v1.0.5-2\packages\Flux\juliaFlux.jl:39
in expression starting at C:\Users\oucnh\JuliaPro\JuliaPro_v1.0.5-2\packages\Flux\juliaFlux.jl:37
└─ in expression starting at C:\Users\oucnh\JuliaPro\JuliaPro_v1.0.5-2\packages\Flux\juliaFlux.jl:39
Julia Pkg.build("ZippedFile")
building zipdir <(C:\Users\oucnh\JuliaPro\JuliaPro_v1.0.5-2\packages\juliaFlux\dep\build.log"
Julia Pkg.build("Core")
building depdir + C:\Users\oucnh\JuliaPro\JuliaPro_v1.0.5-2\packages\juliaFlux\dep\build.log"
Info: Precompiling file (858173a5-b771-4af7-ad37ff91939c)
warning: `param(x)` is deprecated, use `x` instead
└─ caller = top-level scope at none:0
  caller = top-level scope at none:0
warning: `param(x)` is deprecated, use `x` instead
└─ caller = top-level scope at none:0
  caller = top-level scope at none:0
[6] [Julia]

```

图 1-11 FLux 宏包示例

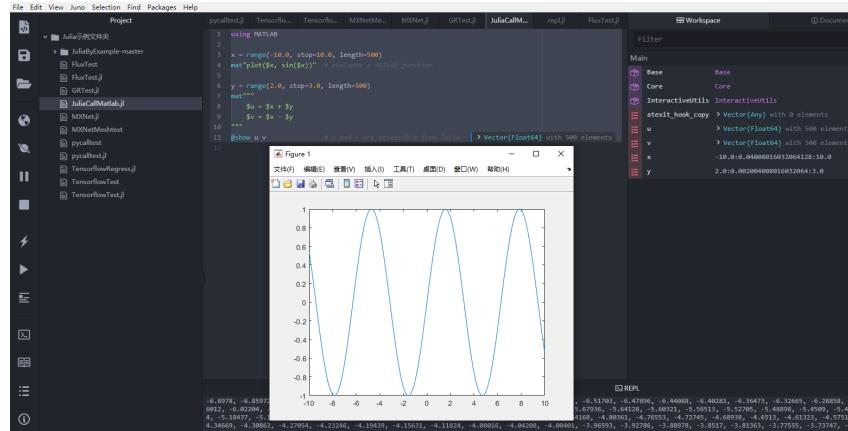


图 1-12 调用 MATLAB 宏包

自上世纪 90 年代以来, 编程语言 Python 已经取得了长足的进步。Python 含有优质的文档、丰富的 AI 库、机器学习库、自然语言和文本处理库。尤其是 Python 中的机器学习, 实现了人工智能领域中大量的需求。当 Guido Van Rossum (1956 年 1 月 31 日-) 开发 Python 时, 他几乎不知道 Python 会成为世界上最流行的语言之一。今天, Python 是人类历史上使用最广泛的编程语言之一, 应用于很多程序中。无论是企业级应用程序, 还是机器学习/人工智能模型、数据科学工作, Python 几乎在所有蓬勃发展的行业和领域中都受人青睐。全世界有超过 800 万的开发人员出于各种目的热衷于使用 Python。由于其动态特性和易扩展性, Python 已经成为开发人员的首选语言。这也是 Python 能够击败 Java 的原因, 而 Java 一度以来都是开发人员最喜欢的语言。由于语言的自然老化, Java 正在自

然老去。大多数新语言都是为解决现代面临的新挑战而设计的。之前开发的语言在解决当时的问题时效率极高，但要让它们跟上不断变化的行业和市场就变得极其困难。但是，Python 作为一种拥有如此庞大用户和开发者支持的开源语言，即使在今天仍然保持着它的巅峰状态。它丰富的库和内置的功能使其成为企业、开发人员和数据科学家的热门选择。尽管 Java 仍然被用于企业开发，但它在其他领域的相关性几乎为零。很难发现一个机器学习专家在 Java 上设计和训练模型。尽管如此，Java 是全球第二大最受开发人员欢迎的语言。

Python 已经成功地在大多数领域取代了 Java。在企业开发方面，Java 面临着来自谷歌的新编程语言 Go 的威胁。随着我们进入未来科技时代，对高性能计算的需求也在不断增长。这也是数据科学和人工智能的时代需求。尽管有人可能认为使用 extreme GPU 有助于提高速度和效率，但事实远非如此。它不能满足特定的数据处理需求。相反，前沿应用程序需要其他依赖项来优化性能，并帮助科学家和开发人员实现预期的目标。最终，这将引导企业和研究机构寻找更健壮的编程语言，为特定的任务及其交付速度而设计。

在这个人人都喜爱 Python 的时代，正面临着来自编程语言世界的新参与者——Julia 的威胁。这几年来我们能够感受到从 MATLAB 到 Python 的过渡。我们知道机器学习几乎在所有应用程序中使用，而且 Python 库使 ML 模型的实现更加容易，所以人们转向了 Python。在此之前，MATLAB 是这项任务的最佳选择，可以帮助人们进行分析和科学计算。但是很明显，人们会把目光转向更容易实现、容易理解、更快速、更高性能和可扩展的解决方案。因此，Python 完美地填补了 JAVA 和 MATLAB 的空白。

3.8 版的 Python 语言引入一个独特的算子。被称为海象算子 (Walrus Operator) 或命名表达式算子 (Named Expression operator)，记为「:=」。海象算子这个新算子 (:=) 能让我们为表达式中的一个变量赋值，看起来类似于海象的眼睛和犬齿。

例 1.20 海象算子可以在 if 语句之中直接执行声明和赋值操作。

- ▶ if country_size := len(countries) < 5: print("Length of countries is " + country_size)
- ▶ employees = [result for id in employee_ids if (result := fetch_employee(id))]
- ▶ while chunk := file.read(256): process(chunk)

海象算子是作为 PEP-572(Python 改进提议) 的一部分而引入的。海象算子的争议点：

- ▶ 句法变化问题：开发者们为 := 提议了多种替代方案，比如「表达式 -> NAME」、「NAME -> 表达式」、「表达式 NAME」等等。少数人建议使用现有的关键字，其他人则使用了新的算子。
- ▶ 后向兼容问题：这个特性无法向后兼容，也无法运行在之前的 Python 版本上。
- ▶ 算子名称问题：人们建议不要使用「海象算子」这样的代号，而是使用「赋值算



图 1-13 海象

子」、「命名表达式算子」、「成为算子」等术语，以免人们不明白。

Julia 和 Python 之间的一个关键区别是处理特定问题的方式。Julia 的构建是为了减轻高性能计算的挑战。尽管 Python 现在已经发展为一种快速的计算语言，但是我们必须承认它不是为这项工作而设计的。然而，Julia 是专门为高速处理和计算工作设计的。虽然它只有几个月的历史，却已经在研究人员和数据科学家中引起轰动。两个月前，Julia 发布了一个稳定的版本，称为 [Julia 1.3](#)，它已经得到了进一步的改进，可以有效地处理大量占用资源的数据科学项目。目前有超过 800 名 Julia 开发人员，他们正在为 GitHub 做贡献，帮助其成为首选语言。

Python 编辑器——Google Colaboratory

Colab 是一个免费的 Jupyter Notebook 环境（你可以想成是网页版多功能笔记本），它不需要进行任何设置就可以使用，并且完全在云端运行。

Tensorflow 云端示例：[Colab](#),

Anaconda 是一个基于 Python 的数据处理和科学计算平台，它已经内置了许多非常有用第三方库，装上 Anaconda，就相当于把 Python 和一些如 Numpy、Pandas、Scrip、Matplotlib 等常用的库自动安装好了，使得安装比常规 Python 安装要容易。如果是学习 Python 的小白，直接安装 Anaconda+Pycharm 就可以。

文件名列表

- ▶ [Anaconda3-2019.10-py3.7Windows-x86_64.exe](#)
- ▶ [torch-1.0.0-cp37-cp37m-win_amd64.whl](#), Pytorch 1.2 帮助文档
- ▶ [PyCharm Professional 2019.2.5](#), 提取码: m7d3, PyCharm Professional 2019.2.5 的启动界面如图1-17.



图 1-14 google-colab 云端 python 编辑器



图 1-15 Cuda90-torch-1.0.0-cp37-cp37m-win_amd64 安装



图 1-16 Anaconda 下测试 torch 包和 Cuda90 是否安装成功



图 1–17 Pycharm 2019.2.5 启动界面

创建项目并配置 Anaconda, 首先点击 create new project, location 为文件存储位置, project interpreter 为解释器, 也就是 Anaconda 中的 python.exe, 按图1–18中步骤操作. 在



图 1–18 Pycharm 中配置使用 Anaconda 中的 python

file 选项中选择 default setting, 选择 project interpreter 并且按步骤选中 Anaconda 中的 python.exe. 最后点击 create(图1–19), 创建完之后进入 pycharm 界面, 点击 close. 图1–20是在 ychrm 中运行 two_layer_net_tensor.py 示例.

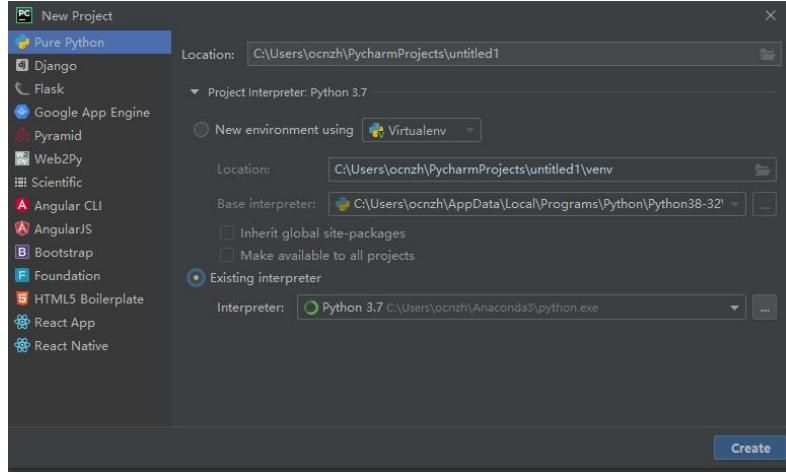


图 1-19 Pycharm 中配置使用 Anaconda 中的 python

```

14 # Randomly initialize weights
15 w1 = torch.randn(3, 10, device=device)
16 w2 = torch.randn(10, 2, device=device)
17
18 learning_rate = 1e-4
19 for t in range(1000):
20     # Forward pass: compute predicted y
21     h = x.mm(w1)
22     h_relu = h.clamp(min=0)
23     y_pred = h_relu.mm(w2)
24
25     # Compute and print loss; loss is a scalar, and is stored in a PyTorch Tensor
26     loss = ((y_pred - y).pow(2).sum())
27     print(t, loss.item())
28
29     # Backprop to compute gradients of w1 and w2 with respect to loss
30     grad_fn = h_relu.register_fn(lambda x: pred.backward(x))
31     grad_fn.retain_grad=True
32     grad_h_relu = grad_fn.grad_fn(w2.t())
33     grad_h = grad_h_relu.clone()
34     grad_h[(h < 0)] = 0
35     grad_w2 = x.t().mm(grad_h)
36
37     # Update weights using gradient descent
38     w1 -= learning_rate * grad_w1
39     w2 -= learning_rate * grad_w2
40
41 for t in range(500):
42     ...

```

图 1-20 Pycharm 中运行 two_layer_net_tensor.py

Python 四个重要应用：验证算法、快速开发、测试运维、数据分析

- ▶ 验证算法：就是对我们公司常见设计算法或者公式的验证，公式代码化。
- ▶ 快速开发：就是用更少的代码来开发网站，Python 在网站前后台有大量的成熟的框架，如 django, flask, bottle, tornado, flask 和 django 的使用较多，国内用 Python 开发的网站有：知乎、豆瓣、扇贝、腾讯、阿里巴巴；
- ▶ 测试运维：用 python 实现的测试工具及过程，包含服务器端、客户端、web、android、client 端的自动化测试，自动化性能测试的执行、监控和分析，常用 selenium

appium 等框架。做运维同学应该清楚，在 Linux 运维工作中日常操作涵盖了监控，部署，网络配置，日志分析，安全检测等等许许多多的方面，无所不包。python 可以写很多的脚本，把“操作”这个行为做到极致。Python 在服务器管理工具上非常丰富，配置管理（saltstack）批量执行（fabric, saltstack）监控（Zenoss, nagios 插件）虚拟化管理（python-libvirt）进程管理（supervisor）云计算（openstack）……还有大部分系统 C 库都有 python 绑定。

- ▶ 数据分析：Python 有三大神器：numpy, scipy, matplotlib，其中 numpy 很多底层使用 C 语言实现的，所以速度很快，用它参加各种数学建模大赛，完全可以替代 r 语言和 MATLAB.

Tensornetwork

利用别的软件下的 pip3 文件安装 tensornetwork，假设如下路径

C:\Users\ocn***\Anaconda3\pkgs\pip-19.2.3-py37_0\Scripts

D:\Program Files (x86)\Microsoft Visual Studio
 \Shared\Python37_64\Scripts

默认情况下，Windows PowerShell 不会从当前位置加载命令。如果信任此命令，切换到其中一个目录后，键入 “.\pip3 install tensornetwork”

查看 python 安装位置

```
python -version
import sys
sys.executable
```

所有 python 的路径：whereis python

当前使用的 python 路径：which python

升级 pip：python -m pip install –upgrade pip

代码：TensorNN.py [下载](#) 

[Theano 的安装说明](#) Theano 是一个让你去定义，优化，计算数学表达式，特别是多维数组（numpy.ndarray）的 Python 包。具有速度快的特点，支持 GPU。Theano 结合了计算机代数系统（computer algebra system, CAS）的特征和优化编译器的功能。Theano 可以使用 GPU 进行运算，用 GPU 运行比 CPU 快 100 倍左右，theano 是比较优秀的 python 模块。theano.function 可以被视为一个从纯符号图构建可调用对象的编译器接口，重要的特征之一是 theano.function 能够优化图，甚至能够将图的一部分或者全部编译成简单的机器指令。

CGT



```

    管理员: Windows PowerShell
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。
尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> pip3 install tensornetwork
Collecting tensornetwork
  Using cached https://files.pythonhosted.org/packages/6d/ed/ea8087d21b73650a3df360e5ae58d8b3ac8cb8787493caa32311355dc4ab/tensornetwork-0.2.0-py3-none-any.whl
Collecting opt-einsum<2.3.0
  Using cached https://files.pythonhosted.org/packages/b8/83/755bd5324777875e9dff19c2e59daec837d0378c09196634524a3d7269ac/opt_einsum-2.1.0.tar.gz
Requirement already satisfied: h5py>=2.9.0 in c:\users\ocnzh\anaconda3\lib\site-packages (from tensornetwork) (2.9.0)
Collecting graphviz<0.11.1
  Using cached https://files.pythonhosted.org/packages/f5/74/dbed754c0abd63768d3a7a7b472da35b08ac442cf87d73d5850a6f32391e/graphviz-0.13.2-py2.py3-none-any.whl
Requirement already satisfied: numpy>=1.16 in c:\users\ocnzh\anaconda3\lib\site-packages (from tensornetwork) (1.16.5)
Requirement already satisfied: six in c:\users\ocnzh\anaconda3\lib\site-packages (from h5py>=2.9.0->tensornetwork) (1.12.0)
Building wheels for collected packages: opt-einsum
  Building wheel for opt-einsum (setup.py) ... done
    Created wheel for opt-einsum: filename=opt_einsum-3.1.0-cp37-none-any.whl size=61701 sha256=86d0021aca797f2476bd6f1c56b3f9t4bc561b6c8beb615429bfed238f0e0a9b
    Stored in directory: C:\Users\ocnzh\AppData\Local\pip\Cache\wheels\2c\b1\94\43d03e130b929aae7ba3f8d15cbd7bc0d1cb5bb38a5c721833
Successfully built opt-einsum
Installing collected packages: opt-einsum, graphviz, tensornetwork
Successfully installed graphviz-0.13.2 opt-einsum-3.1.0 tensornetwork-0.2.0
PS C:\Windows\system32>

```

图 1-21 pip3 文件安装 tensornetwork



```

    管理员: Windows PowerShell
Windows PowerShell
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting theano
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/7d/c4/6341148ad458b6cd8361b774d7ee6895c38eab88f05331f22304c484ed5d/Theano-1.0.4.tar.gz (2.8 MB)
[██████████] 2.8 MB 656 kB/s
Requirement already satisfied: numpy>=1.9.1 in c:\users\ocnzh\anaconda3\lib\site-packages (from theano) (1.16.5)
Requirement already satisfied: scipy>=0.14 in c:\users\ocnzh\anaconda3\lib\site-packages (from theano) (1.4.1)
Requirement already satisfied: six>=1.9.0 in c:\users\ocnzh\anaconda3\lib\site-packages (from theano) (1.12.0)
Building wheels for collected packages: theano
  Building wheel for theano (setup.py) ... done
    Created wheel for theano: filename=Theano-1.0.4-py3-none-any.whl size=2667193 sha256=33856a1da59d29101fd6115ae9462421f177eaf6ba2d73fc593b40921156197
    Stored in directory: c:\users\ocnzh\appdata\local\pip\cache\wheels\96\f0\815383507c0653f0e58c21414cc4affd7c01216ff6214b04ea
Successfully built theano
Installing collected packages: theano
Successfully installed theano-1.0.4
PS C:\Windows\system32>

```

图 1-22 安装 theano

1.3 AI 研究的基本内容

现在,人类社会正在进入第四次技术革命,就是以互联网、大数据和人工智能技术驱动的技术革命,它会推动人类社会生活生产方式进入自动化时代,人工智能时代。而人类社会的教育,特别是学校教育,将会发生什么样的变化?具体来说有以下几个领域。

一、(教育的人文性). 也就是教育的价值转型,人文是教育的本质。脑科学证明,在人类接受幼儿和学校教育的阶段,离不开师生交往和生生交往。从这个意义上来看,教师不会被技术取代,需要转型的是教育的价值导向。我认为,技术变革使未来匮乏的不再是征服世界和改造世界的能力,而是对于人的心理、情感和品德的培养。

二、(教育的全域化特点). 在技术革命的影响下,学校教育、家庭教育和社会教育的边界正在解构。在全域教育时代,如何使学校教育线上线下融合、学校家庭沟通协同、校内和校外衔接、学校和社区共治,这是未来教育重大的挑战。

三、(教育的集智性). 教育将从过去封闭的、个体化的教学,转向开放的集智化教学时代。随着人工智能的发展,教育合作将基于人力配置进行重组,家庭与学校的教学协同性,教育资源配置的协同性都将产生重大变化。

四、(教育的自组织性). 未来教育将从过去的班级授课制转变为学习共同体,学校的功能定位要从过去服务于知识传承的教学型组织,转向服务自主探究的学习型组织。面向未来,应该推进学习型组织的转变,把传统教室进行改造,开展社团学习活动以及鼓励教师进行自组织教学活动。

五、(教育的个体性). 传统教育是同质化的,是按照统一的课程标准、教材、班级,以及统一的备课、上课、考试评价进行的。而未来强调人的主体性、个性化、差别化,我们要从知识的教学转向核心素养的教育。必须要改变传统的教学方式,也就是课程的供给方式要去同质化,要把统一的知识教学和针对个人的定制教育结合并存。

六、(教育的综合化). 未来课程组织的逻辑将从分科走向综合,传统的课程体系学科彼此独立。而今天的教育变革强调跨学科、主题性,要将课程去学科化,增加主题化、跨学科和生活化的教育,也就是将学科课程逻辑与实践课程逻辑相结合。此外,教育还要为学生提供试错的机会和发展潜能的平台,要注重学生身心合一引导其进行自主建构的学习,要将智能数据作为工具分析学生的学习状态、兴趣和优势,以及未来教育还要开展多元合作治理等等。(北京师范大学教授 | 中国教育学会副会长张志勇,现代教育报,2019-12-25)

1.3.1 AI 的学科位置

AI 是一门新兴的边缘学科,是自然科学与社会科学的交叉学科 AI 的交叉包括: 逻辑、思维、生理、心理、计算机、电子、语言、自动化、光、声等 AI 的核心是思维与智能,构成了自己独特的学科体系 AI 的基础学科包括: 数学(离散、模糊)、思维科学(认知心理、逻辑思维学、形象思维学)和计算机(硬件、软件)等.

人工智能是最宽泛的概念,机器学习则是实现人工智能的一种方式,也是目前较有效的方式. 深度学习是机器学习算法中最热的一个分支,在近些年取得了显著的进展,并代替了多数传统机器学习算法. 所以,三者的关系可用下图表示,人工智能 > 机器学习 > 深度学习.



图 1-23 人工智能、机器学习和深度学习三者的关系



图 1-24 学科树状示意图



AI 的研究方向分为 2 个层次，第 1 层分为计划与调度、专家系统、机器学习、推荐系统等。而机器学习还可以继续往下分，例如，有监督学习、无监督学习、深度学习等。在这 2 层分类中，发现了前面提到的机器学习和深度学习。



图 1-25 AI 研究的方向

1.3.2 人工智能学科（F06）代码

2020 年度人工智能学科（F06）代码进行了较大调整，大部分代码对应的内容都发生了变化，比如增加了新的申请方向“F0602 复杂性科学与人工智能理论”，原申请方向“机器感知与模式识别”分拆为“F0604 机器感知与机器视觉”和“F0605 模式识别与数据挖掘”等。

1.3.3 与脑科学和认知科学的交叉研究-脑科学

定义 1.21 脑科学

又称神经科学，其目的是要认识脑、保护脑和创造脑。美国神经科学学会的定义：神经科学是为了了解神经系统内分子水平、细胞水平及细胞间的变化过程，以及这些过程在中枢的功能、控制系统内的整合作用所进行的研究。



定义 1.22 脑的涵义

从狭义和广义两个方面来理解：从狭义方面，脑是指中枢神经系统，有时特指大脑；从广义方面，脑可泛指整个神经系统。人工智能是从广义角度来理解脑科学的，因此它涵盖了所有与认识脑和神经系统有关的研究。人脑是自然界中最复杂、最高级的智能系统：这种复杂性主要表现在人脑是由巨量神经元经其突触的广泛并行互连所形成的一个巨复杂系统。

现代脑科学的基本问题主要包括：

- (1) 揭示神经元之间的连接形式，奠定行为的脑机制的结构基础；
- (2) 阐明神经活动的基本过程，说明在分子、细胞到行为等不同层次上神经信号的产生、传递、调制等基本过程；
- (3) 鉴别神经元的特殊细胞生物学特性；
- (4) 认识实现各种功能的神经回路基础；
- (5) 解释脑的高级功能机制等。

脑科学是人工智能的基础：研究的任何进展，都将会对人工智能的研究起到积极的推动作用，因此人工智能应该加强与脑科学的交叉研究，以及人类智能与机器智能的集成研究。

定义 1.23 认知

认为是和情感、动机、意志相对应的理智或认识过程，或者是为了一定的目的，在一定的心理结构中进行的信息加工过程。

美国心理学家浩斯顿 (Houston) 等人把认知归纳为以下 5 种主要类型：

- (1) 认知是信息的处理过程；
- (2) 认知是心理上的符号运算；
- (3) 认知是问题求解；
- (4) 认知是思维；
- (5) 认知是一组相关的活动，如知觉、记忆、思维、判断、推理、问题求解、学习、想象、概念形成及语言使用等。

定义 1.24 认知科学

认知科学（亦称思维科学）是研究人类感知和思维信息处理过程的一门学科，其主要研究目的就是要说明和解释人类在完成认知活动时是如何进行信息加工的。

认知科学也是人工智能的重要理论基础，对人工智能发展起着根本性的作用。认知

科学涉及的问题非常广泛,除了像浩斯顿提出的知觉、语言、学习、记忆、思维、问题求解、创造、注意、想象等相关联活动外,还会受到环境、社会、文化背景等方面的影响.

从认知观点看, AI 应同时开展对逻辑思维、形象思维和灵感思维的研究.

1.3.4 智能模拟的方法和技术研究

- ▶ 机器感知: 就是要让计算机具有类似于人的感知能力, 如视觉、听觉、触觉、嗅觉、味觉.
 - 机器视觉 (或叫计算机视觉): 就是给计算机配上能看的视觉器官, 如摄像机等, 使它可以识别并理解文字、图像、景物等.
 - 机器听觉 (或叫计算机听觉): 就是给计算配上能听的听觉器官, 如话筒等, 使计算机能够识别并理解语言、声音等.
 - 机器感知相当于智能系统的输入部分.
 - 机器感知的专门的研究领域: 计算机视觉、模式识别、自然语言理解.
- ▶ 机器思维: 让计算机能够对感知到的外界信息和自己产生的内部信息进行思维性加工.
 - 逻辑思维
 - 形象思维
 - 灵感思维
- ▶ 机器学习: 让计算机能够像人那样自动地获取新知识, 并在实践中不断地完善自我和增强能力.
 - 机器学习方法: 机械学习、类比学习、归纳学习、发现学习、遗传学习和连接学习等.
- ▶ 机器行为: 让计算机能够具有像人那样地行动和表达能力, 如走、跑、拿、说、唱、画画等. 相当于智能系统的输出部分.
- ▶ 智能系统与智能机器
 - 无论是人工智能的近期目标还是远期目标, 都需要建立智能系统或构造智能机器.
 - 需要开展对系统模型、构造技术、构造工具及语言环境等研究.

1.4 AI 研究中的不同学派——不同学派

- ▶ 符号主义学派 (逻辑主义和心理学派)
 - 主要观点: AI 起源于数理逻辑, 人类认知的基元是符号, 认知过程是符号表示

- 上的一种运算.
- 代表性成果: 厄尔和西蒙等人研制的称为逻辑理论机的数学定理证明程序 LT.
 - 代表人物: 纽厄尔、肖、西蒙和尼尔逊 (Nilsson) 等.
 - 连接主义学派 (仿生学派或心理学派)
 - 主要观点: AI 起源于仿生学, 特别是人脑模型, 人类认知的基元是神经元, 认知过程是神经元的连接活动过程.
 - 代表性成果: 由麦克洛奇和皮兹创立的脑模型, 即 MP 模型.
 - 代表人物: 麦克洛奇和皮兹.
 - 行为主义学派 (进化主义、控制论学派)
 - 主要观点: AI 起源于控制论, 智能取决于感知和行为, 取决于对外界复杂环境的适应, 而不是推理.
 - 代表性成果: 澳大利亚科学院院士、机器人专家 Rodney Brooks 教授研制的机器虫, 头衔: 澳洲科学院 fellow、美国国家工程院院士、国际人工智能协会 (AAAI) 创始会员、作家、机器人企业家、MIT 机器人教授、MIT 计算机科学和人工智能实验室前主任、iRobot 创始人及前 CTO、RethinkRobotics 前 CTO 及联合创始人.
 - 代表人物: Brooks 教授, 是机器人行为主义学派的旗帜性人物. 而这位大神不仅在学术领域建树卓著, 在 1997 年上映的电影《又快又贱又失控》(Fast, Cheap & Out of Control), 图1-26.

不同学派的理论之争

► 符号主义

智能的基础是知识, 其核心是知识表示和知识推理; 知识可用符号表示, 也可用符号进行推理, 因而可以建立基于知识的人类智能和机器智能的统一的理论体系.

► 连接主义

思维的基元是神经元, 而不是符号; 思维过程是神经元的联结活动过程, 而不是符号运算过程; 反对符号主义关于物理符号系统的假设.

► 行为主义

智能取决于感知和行动, 提出了智能行为的“感知—动作”模型; 智能不需要知识、不需要表示、不需要推理; 人工智能可以像人类智能那样逐步进化.

► 符号主义、功能模拟

构造能够模拟大脑功能的智能系统. 相当于“鸟飞”.

► 连接主义、结构模拟



图 1-26 电影《又快又贱又失控》

构造模拟大脑结构的神经网络系统. 相当于“飞鸟”.

► 行为主义、行为模拟

构造具有进化能力的智能系统. 相当于“由猿到人”.

例 1.25 (全息 AI) 微美全息 WIMI 专注于计算机视觉全息云服务. 据介绍, 微美全息覆盖从全息计算机视觉 AI 合成、全息视觉呈现、全息互动软件开发、全息 AR 线上及线下广告投放、全息 AR SDK 支付、5G 全息通讯软件开发、全息人脸识别开发、全息 AI 换脸开发等全息 AR 技术的多个环节, 是一家全息云综合技术方案提供商. 其商业应用场景主要聚集在家用娱乐、光场影院、演艺系统、商业发布系统及广告展示系统等五大专业领域.

1.5 人工智能近期发展分析

人工智能自提出到现在已与经历 60 多年的演化和发展, 在移动互联网、大数据、超算、传感网络、脑科学等理论和技术驱动以及经济社会发展需求牵引下, 呈现出强劲和加速发展的势头; 以强化学习、深度学习及宽度学习等为代表的先进学习方法得到了迅猛发展, 在解决大数据驱动知识学习、跨媒体协同处理、人机物协同增强智能、群体集成智能、自主智能系统等领域关键技术问题中发挥了非常关键性的作用; 与此同时, 类脑

智能的研究与开发工作也同样受到国内外相关研究机构和学者的高度关注及重视,为人工智能的深入快速发展迎来了机遇和挑战.

人工智能已经走过 50 多年的历史,下一步该如何发展,是人工智能学者最为关心的一个问题.但要准确回答这一问题却又十分困难,下面所给出的分析(6 个方面)仅是对国内外学者的一些观点的归纳.

1.5.1 人工智能发展方向

1.5.1.1 人工智能应用前景



图 1-27 以深度学习为基础的 AI 技术的应用

1.5.1.2 人工智能近期发展方向

人工智能的产业化前景(人机物三元协同控制与决策优化的角度):人机物智能技术就是综合应用物联网、移动互联网、通信、大数据计算、人工智能等技术使物与物之间、人与物之间实现互联和互通,并通过协同仿真、分布计算、跨平台管控等智能处理技术实现人与万物的协同工作.利用人机物三元协同的智能技术,使人类社会、虚拟空间、自然空间、机器物理空间实现联通互动、数字双生、虚实交融,形成以人为中心人机物三元融合的协同工作场景.人机物共融将推进生产方式向智能技术领域演化,同时也体现了智能制造的标志性特征.将人的联想、推理、反馈、学习、理解等能力与机器的搜索、作业、推理、学习等能力相融合,解决人机物三元分布式协同控制与决策优化问题.在复杂制造系统的生产计划、质量管控、设备维护和制造执行等企业的各个应用管理层面均体现了人机物三元协同与融合技术的普适性.

多学科交叉研究 人工智能理论基础研究强调与脑科学、认知科学、心理学、信息科学、生物学、逻辑学、物理学和数学等学科的交叉研究.

- ▶ 脑科学为人工智能研究提供人脑神经系统功能的本质和机理;
- ▶ 认知科学为人工智能研究提供感知、思维、学习和语言等基本原理;
- ▶ 心理学为人工智能研究提供认知、情感、意识等心理过程及联系;
- ▶ 生物学为人工智能研究提供自然界生物运行的机制;
- ▶ 逻辑学为人工智能研究提供思维规律描述的理论和方法;
- ▶ 信息科学为人工智能研究提供模拟的物质基础和技术手段;
- ▶ 数学为人工智能研究提供各种有效的计算模型和方法.

集成智能研究 智能的物质、能量、信息基础: 自然智能是一种基于“碳”的信息处理, 人工智能是一种基于“硅”的信息处理, 尽管这两种信息处理所基于的物质不同, 但它们的运算能量的都应该是电信号. 那么, 是否可以在“碳”和“硅”这两种不同物质上建立一种基于电信号的统一的信息处理模型?

人类的智能: 物质 (碳)+ 能量 (生物电) \rightarrow (生物) 信息.

人造的智能: 物质 (硅)+ 能量 (物理电) \rightarrow (电子) 信息.

集成智能的研究: 脑-机接口 (**Brain-Computer Interface, BCI**) 的研究成果.

进一步证实了集成智能的可能性和辉煌前景.

智能产生机理测试, 图1-28.



图 1-28 智能产生机理测试

多学派融合研究 融合是一种必然趋势: 符号主义、联结主义和行为主义三大学派各有所长、各有所短, 它们各自经过一段时间的分立研究之后, 正逐步开始走向融合. 多学派融合是人工智能发展的一种必然趋势.

融合需要解决的关键问题:

- (1) 不同学派之间的共同机制是什么?
- (2) 怎样建立一个统一的智能理论体系?
- (3) 如何真正实现它们之间的有机融合等.

智能网格 互联网是人类历史上发展速度最快的一次技术革命, 但目前仍处于发展的初级阶段, 其智能水平还很低. 未来的互联网应该是一种具有自动调整功能, 各个节点之间协调工作, 能为用户提供便利、有效服务的智能网络.

智能机器人研究 智能机器人将会对社会生产力发展和人类社会进步, 以及对人们生活、工作和思维方式的改进等产生不可估量的影响 (例如, 海军总医院的手术机器人“黎元”, 可通过互联网完成远程颅脑手术). 未来智能机器人应该是一种具有人类感知、行为能力, 超强记忆、学习、推理、规划能力, 有情感、人性化, 能代替人类在真实环境中自主工作的机器人.

智能应用和智能产业 智能技术将进一步与主流信息技术融合, 并将应用于人类社会的各个领域和人类生活的各个方面. 有人预计: 智能产业将逐步成为社会第四产业; 智件将逐步从软件中分离出来, 成为智能计算机系统的三件 (硬件、软件、智件) 之一.

《机器人和人工智能的崛起》一书的作者 Martin Ford 说: “机器人崛起的可能性很大, 自动化在最初的不稳定性对发展中国家的影响更大, 而最终的结果则取决于我们如何选择、我们如何作为以及我们如何适应这种状况.”

1.5.2 人工智能发展八大新趋势

人工智能 (AI) 是物联网及工业 4.0 发展的核心. 尤其, 当特斯拉 (Tesla) 推出电动车及苹果 (Apple) 推出 FaceID 之后, 让市场体验到 AI 芯片的无限商机. 同时, AI 应用接受度越高的国家, 将对其 GDP 产生贡献愈大. AI 芯片包含三大类市场, 分别是数据中心 (云端)、通信终端产品 (手机)、特定应用产品 (自驾车、头戴式 AR/VR、无人机、机器人...). 当前机器学习多采用 GPU 图像处理, 尤以 Nvidia 是此一领域龙头, 但是, 有些业者认为 GPU 处理效率不够快, 而且因应众多特定新产品的不同需求, 于是, 推出 NPU、VPU、TPU、NVPU... 等等. 目前还不清楚哪种架构的芯片会在 AI 大战获胜. 但 (手机)



图 1-29 示意图

终端市场对于 AI 芯片的功耗、尺寸、价格都有极为严格的要求, 难度上比云端数据芯片更高。为抢未来 AI 应用市场商机, 科技巨头如 Google、微软、苹果企图建构 AI 平台生态模式吃下整个产业链。

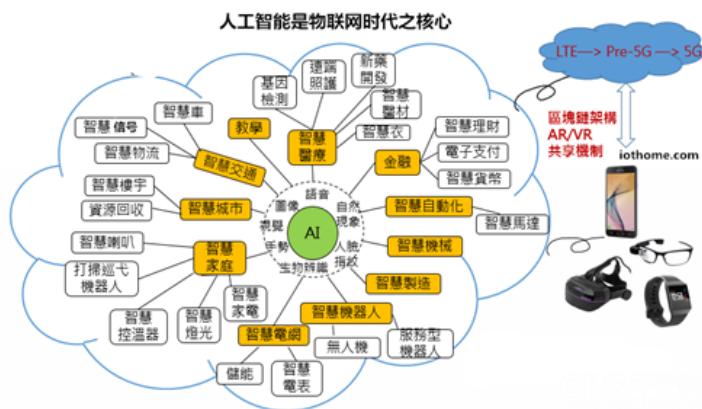


图 1-30 人工智能是物联网时代之核心

目前来看, 未来 AI 发展有八大新趋势

► 趋势一、AI 于各行业垂直领域应用具有巨大的潜力

人工智能市场在零售、交通运输和自动化、制造业及农业等各行业垂直领域具有巨大的潜力。而驱动市场的主要因素, 是人工智能技术在各种终端用户垂直领域的应用数量不断增加, 尤其是改善对终端消费者服务。当然人工智能市场要起来也受到 IT 基础设施完善、智能手机及智能穿戴式设备的普及。其中, 以自然语言处理 (NLP) 应用市场占 AI 市场很大部分。随着自然语言处理的技术不断精进而驱动消费者服务的成长, 还有: 汽车信息通讯娱乐系统、AI 机器人及支持 AI 的智能手机等领域。

► 趋势二、[AI 导入医疗保健行业维持高速增长](#)

由于医疗保健行业大量使用大数据及人工智能,进而精准改善疾病诊断、医疗人员与患者之间人力的不平衡、降低医疗成本、促进跨行业合作关系。此外 AI 还广泛应用于临床试验、大型医疗计划、医疗咨询与宣传推广和销售开发。人工智能导入医疗保健行业从 2016 年到 2022 年维持很高成长,预计从 2016 年的 6.671 亿美元达到 2022 年的 79.888 亿美元年均复合增长率为 52.68%。



图 1-31 人工智能是物联网时代之核心

► 趋势三、[AI 取代屏幕成为新 UI/UX 接口](#)

过去从 PC 到手机时代以来,用户接口都是透过屏幕或键盘来互动。随着智能喇叭 (SmartSpeaker)、虚拟/增强现实 (VR/AR) 与自动驾驶车系统陆续进入人类生活环境,加速在不需要屏幕的情况下,人们也能够很轻松自在与运算系统沟通。这表示着人工智能透过自然语言处理与机器学习让技术变得更为直观,也变得较易操控,未来将可以取代屏幕在用户接口与用户体验的地位。人工智能除了在企业后端扮演重要角色外,在技术接口也可承担更复杂角色。例如: 使用视觉图形的自动驾驶车,透过人工神经网络以实现实时翻译,也就是说,人工智能让接口变得更为简单且更有智能,也因此设定了未来互动的高标准模式。

► 趋势四、[未来手机芯片一定内建 AI 运算核心](#)

现阶段主流的 ARM 架构处理器速度不够快,若要进行大量的图像运算仍嫌不足,所以未来的手机芯片一定会内建 AI 运算核心。正如,苹果将 3D 感测技术带入 iPhone 之后,Android 阵营智能手机也将跟进导入 3D 感测相关应用。

► 趋势五、[AI 芯片关键在于成功整合软硬件](#)

AI 芯片的核心是半导体及算法。AI 硬件主要是要求更快指令周期与低功耗,包括 GPU、DSP、ASIC、FPGA 和神经元芯片,且须与深度学习算法相结合,而成功相结合的关键在于先进的封装技术。总体来说 GPU 比 FPGA 快,而在功率效能方面 FPGA 比 GPU 好,所以 AI 硬件选择就看产品供货商的需求考虑而定。例

如, 苹果的 FaceID 脸部辨识就是 3D 深度感测芯片加上神经引擎运算功能, 整合高达 8 个组件进行分析, 分别是红外线镜头、泛光感应组件、距离传感器、环境光传感器、前端相机、点阵投影器、喇叭与麦克风. 苹果强调用户的生物识别数据, 包含: 指纹或脸部辨识都以加密形式储存在 iPhone 内部, 所以不易被窃取.

► 趋势六、[使用人工智能的预测营销](#)

人工智能有助于预测用户的个性. 全球用户进行了数十亿次搜索. 搜索数据被保留在数据库中, 用于预测用户的人口位置、喜好、兴趣和职业等. 这有助于针对产品和服务定位到正确的客户.

► 趋势七、[定制网站](#)

如何向不同用户展示具有个性化内容的同一网站? 这在人工智能的帮助下也成为可能. 根据过去的搜索, 人口统计位置, 性别, 喜好, 人工智能可以定制网站, 并向用户显示他想看到的内容. 这在很大程度上改善了用户体验. 人工智能的市场价值具有巨大的净值空间. 随着商业营销概念的日益增多, 人工智能的市场价值也越来越多元化. 人工智能公司与商业营销合作, 获得了巨额利润. 人工智能的市场价值呈指数级增长, 预计未来几年还会增长.

► 趋势八、[语音搜索](#)

未来几年, 语音搜索将完全超越文本搜索. 在人工智能和语音识别系统的帮助下, 用户仅仅使用自己的语音就可以发出各种命令. AI 系统首先识别语音, 将它们从语音转换为文本, 并提供想得到的结果. 这意味着网站的内容应与人类对话交互过程相匹配.

► 趋势九、[图像识别系统](#)

机器人技术和人工智能使人们很容易识别人、产品等的图像. 因此, 人工智能可以识别图像中的人, 并且更先进的系统能够收集消费者信息, 进一步用于决策.



► 趋势十、[AI 自主学习是终极目标](#)

AI “大脑” 变聪明是分阶段进行, 从机器学习进化到深度学习, 再进化至自主学

习。目前，仍处于机器学习及深度学习的阶段，若要达到自主学习需要解决四大关键问题。首先，是为自主机器打造一个 AI 平台；还要提供一个能够让自主机器进行自主学习的虚拟环境，必须符合物理法则，碰撞，压力，效果都要与现实世界一样；然后再将 AI 的“大脑”放到自主机器的框架中；最后建立虚拟世界入口（VR）。目前，NVIDIA 推出自主机器处理器 Xavier，就在为自主机器的商用和普及做准备工作。

► 趋势十一、**最完美的架构是把 CPU 和 GPU(或其他处理器) 结合起来**

未来，还会推出许多专门的领域所需的超强性能的处理器，但是 CPU 是通用于各种设备，什么场景都可以适用。所以，最完美的架构是把 CPU 和 GPU(或其他处理器) 结合起来。例如，NVIDIA 推出 CUDA 计算架构，将专用功能 ASIC 与通用编程模型相结合，使开发人员实现多种算法。

► 趋势十二、**AR 成为 AI 的眼睛，两者是互补、不可或缺** 未来的 AI 需要 AR，未来的 AR 也需要 AI，可以将 AR 比喻成 AI 的眼睛。为了机器人学习而创造的在虚拟世界，本身就是虚拟现实。还有，如果要让人进入到虚拟环境去对机器人进行训练，还需要更多其它的技术。



来源：物联之家

1.5.3 达摩院 2020 十大科技趋势

► 趋势一、**人工智能从感知智能向认知智能演进**

人工智能已经在“听、说、看”等感知智能领域已经达到或超越了人类水准，但在需要外部知识、逻辑推理或者领域迁移的认知智能领域还处于初级阶段。认知智能将从认知心理学、脑科学及人类社会历史中汲取灵感，并结合跨领域知识图

谱、因果推理、持续学习等技术,建立稳定获取和表达知识的有效机制,让知识能够被机器理解和运用,实现从感知智能到认知智能的关键突破.

► 趋势二、[计算存储一体化突破 AI 算力瓶颈](#)

冯诺伊曼架构的存储和计算分离,已经不适合数据驱动的人工智能应用需求. 频繁的数据搬运导致的算力瓶颈以及功耗瓶颈已经成为对更先进算法探索的限制因素. 类似于脑神经结构的存内计算架构将数据存储单元和计算单元融合为一体,能显著减少数据搬运,极大提高计算并行度和能效. 计算存储一体化在硬件架构方面的革新,将突破 AI 算力瓶颈.

► 趋势三、[工业互联网的超融合](#)

5G、IoT 设备、云计算、边缘计算的迅速发展将推动工业互联网的超融合,实现工控系统、通信系统和信息化系统的智能化融合. 制造企业将实现设备自动化、搬送自动化和排产自动化,进而实现柔性制造,同时工厂上下游制造产线能实时调整和协同. 这将大幅提升工厂的生产效率及企业的盈利能力. 对产值数十万亿乃至数百万亿的工业产业而言,提高 5%-10% 的效率,就会产生数万亿人民币的价值.

► 趋势四、[机器间大规模协作成为可能](#)

传统单体智能无法满足大规模智能设备的实时感知、决策. 物联网协同感知技术、5G 通信技术的发展将实现多个智能体之间的协同——机器彼此合作、相互竞争共同完成目标任务. 多智能体协同带来的群体智能将进一步放大智能系统的价值: 大规模智能交通灯调度将实现动态实时调整,仓储机器人协作完成货物分拣的高效协作,无人驾驶车可以感知全局路况,群体无人机协同将高效打通最后一公里配送.

► 趋势五、[模块化降低芯片设计门槛](#)

传统芯片设计模式无法高效应对快速迭代、定制化与碎片化的芯片需求. 以 RISC-V 为代表的开放指令集及其相应的开源 SoC 芯片设计、高级抽象硬件描述语言和基于 IP 的模块化芯片设计方法,推动了芯片敏捷设计方法与开源芯片生态的快速发展. 此外,基于芯粒(chiplet)的模块化设计方法用先进封装的方式将不同功能“芯片模块”封装在一起,可以跳过流片快速定制出一个符合应用需求的芯片,进一步加快了芯片的交付.

► 趋势六、[规模化生产级区块链应用将走入大众](#)

区块链 BaaS(Blockchain as a Service) 服务将进一步降低企业应用区块链技术的门槛,专为区块链设计的端、云、链各类固化核心算法的硬件芯片等也将应运而生,实现物理世界资产与链上资产的锚定,进一步拓展价值互联网的边界、实现

万链互联。未来将涌现大批创新区块链应用场景以及跨行业、跨生态的多维协作，日活千万以上的规模化生产级区块链应用将会走入大众。

► 趋势七、[量子计算进入攻坚期](#)

2019年，“量子霸权”之争让量子计算在再次成为世界科技焦点。超导量子计算芯片的成果，增强了行业对超导路线及对大规模量子计算实现步伐的乐观预期。2020年量子计算领域将会经历投入进一步增大、竞争激化、产业化加速和生态更加丰富的阶段。作为两个最关键的技术里程碑，容错量子计算和演示实用量子优势将是量子计算实用化的转折点。未来几年内，真正达到其中任何一个都将是十分艰巨的任务，量子计算将进入技术攻坚期。

► 趋势八、[新材料推动半导体器件革新](#)

在摩尔定律放缓以及算力和存储需求爆发的双重压力下，以硅为主体的经典晶体管很难维持半导体产业的持续发展，各大半导体厂商对于3纳米以下的芯片走向都没有明确的答案。新材料将通过全新物理机制实现全新的逻辑、存储及互联概念和器件，推动半导体产业的革新。例如，拓扑绝缘体、二维超导材料等能够实现无损耗的电子和自旋输运，可以成为全新的高性能逻辑和互联器件的基础；新型磁性材料和新型阻变材料能够带来高性能磁性存储器如SOT-MRAM和阻变存储器。

► 趋势九、[保护数据隐私的AI技术将加速落地](#)

数据流通所产生的合规成本越来越高。使用AI技术保护数据隐私正在成为新的技术热点，其能够在保证各方数据安全和隐私的同时，联合使用方实现特定计算，解决数据孤岛以及数据共享可信程度低的问题，实现数据的价值。

► 趋势十、[云成为IT技术创新的中心](#)

随着云技术的深入发展，云已经远远超过IT基础设施的范畴，渐渐演变成所有IT技术创新的中心。云已经贯穿新型芯片、新型数据库、自驱动自适应的网络、大数据、AI、物联网、区块链、量子计算整个IT技术链路，同时又衍生了无服务器计算、云原生软件架构、软硬一体化设计、智能自动化运维等全新的技术模式，云正在重新定义IT的一切。广义的云，正在源源不断地将新的IT技术变成触手可及的服务，成为整个数字经济的基础设施。

1.6 AI的相关应用领域

尽管目前不同人工智能的研究学派在理论基础、研究方法等方面还存在一定差异，但这些并没有影响人工智能的发展，反而使人工智能的研究更加客观、全面和深入。

今天, 被冠以智能的科技领域和社会现实数不胜数, 智能已成为一个极具价值的学术标签和商业标签, 并在科技进步和社会发展中扮演着越来越重要的角色.

面对人工智能这样一个高度交叉的新兴学科, 其研究和应用领域的划分可以有多种不同方法. 这里采用了基于智能本质和作用的划分方法, 即从感知、思维、行为、学习; 计算智能、分布智能、智能机器、智能系统、智能应用等方面来进行讨论.

1.6.1 机器思维

机器思维: 就是让计算机模仿并实现人的思维, 具有人类类似的思维能力, 以对感知到的外界信息和自己内部产生的信息进行思维性加工. 包括: 推理、搜索、规划等方面的研究.

1.6.1.1 推理和搜索

推理 推理的概念: 推理是指按照某种策略从已知事实出发利用知识推出所需结论的过程.

- ▶ 推理的类型: 可根据所用知识的确定性, 将其分为:
 - ▶ 确定性推理, 也称二值推理, 指推理所使用的知识和推出的结论都是可以精确表示的, 其真值要么为真、要么为假. 确定性推理主要是基于一阶经典逻辑. 它能解决的问题很有限.
 - ▶ 不确定性推理, 指推理所使用的知识和推出的结论可以是不确定的. 所谓不确定性是对非精确性、随机性、模糊型和非完备性的统称. 不确定性推理主要基于非经典逻辑和概率等. 非一阶经典逻辑是泛指除一阶经典逻辑以外的其他各种逻辑, 如多值逻辑、模糊逻辑、模态逻辑、概率逻辑、默认逻辑、次协调逻辑及泛逻辑等.
- ▶ 推理的理论基础: 逻辑是一门研究人们思维规律的学科, 数理逻辑则是用数学的方法去研究逻辑问题.

最常用的不确定性推理方法: 基于可信度的确定性理论, 基于 Bayes 公式的主观 Bayes 方法, 基于概率的证据理论和基于模糊逻辑的模糊推理(可能性理论)等.

搜索 搜索的概念: 是指为了达到某一目标, 不断寻找推理线路, 以引导和控制推理, 使问题得以解决的过程.

搜索的主要问题: 人工智能最关心的是如何利用已有的有用信息来引导搜索过程, 尽快达到目标, 即启发式搜索方法.

- ▶ 搜索的类型: 可根据问题的表示方式将其分为状态空间搜索和与/或树搜索两大类型.
- ▶ 状态空间搜索是一种用状态空间法求解问题时的搜索方法.
- ▶ 与/或树搜索是一种用问题规约法求解问题时的搜索方法.
 - 状态空间的启发式搜索方法
 - 与/或树的启发式搜索方法

规划

规划的概念 是指从某个特定问题状态出发, 寻找并建立一个操作序列, 直到求得目标状态为止的一个行动过程的描述.

规划的特点 与一般问题求解技术相比, 规划更侧重于问题求解过程, 并且要解决的问题一般是真实世界的真实问题, 而不是抽象的数学模型.

例 1.26 第二章的机器人移盒子、猴子摘香蕉等问题.

规划系统的例子 斯坦福研究所的问题求解系统 (Stanford Research Institute Problem Solver, STRIPS), 是一种基于状态空间和 F 规则的规划系统. 它由以下 3 部分所组成:

- (1) 世界模型: 用一阶谓词公式表示, 它包括问题的初始状态和目标状态.
- (2) 操作符 (即 F 规则): 它包括先决条件、删除表和添加表.
- (3) 操作方法: 它采用状态空间表示和中间---结局分析的方法. 其中, 状态空间包括初始状态、中间状态和目标状态; 中间---结局分析的每一步都选择能够缩小当前状态与目标状态之间的差距的先决条件可以满足的 F 规则执行, 直至到达目标为止.

1.6.2 机器感知

机器感知是机器获取外界信息的主要途径, 也是机器智能的重要组成部分.

定义 1.27 机器感知

让计算机具有类似于人的感知能力, 如视觉、听觉、触觉、嗅觉、味觉.



下面主要介绍机器视觉、模式识别和自然语言理解.

1.6.2.1 机器视觉

定义 1.28 计算机视觉

用计算机来实现或模拟人类的视觉功能, 其主要研究目标是使计算机具有通过二维图像认知三维环境信息的能力.



重要性: 在人类感知到的外界信息中, 有 80% 以上是通过视觉得到的.

视觉: 不仅仅指对光信号的感受, 它包括了对视觉信息的获取、传输、处理、存储与理解的全过程.

视觉系统: 人类视觉系统的功能是通过眼睛与大脑共同实现的. 人们视野中的物体在可见光的照射下, 先在眼睛的视网膜上形成图像, 然后由感光细胞转换成神经脉冲信号, 再经神经纤维传入大脑皮层, 最后由大脑皮层对其进行处理与理解.

例 1.29 设 p 为一物体. 两个透镜的轴线是平行的. f 为两透镜与图像平面的距离, 即为焦距. b 为两透镜轴线在基线上的距离, 即为两眼的距离. l 和 m 分别是 p 点与左、右透镜轴线的距离. a 和 c 分别是图像平面上的左、右图像与其相应透镜轴线上的距离. 从两个相似三角形, 可得到下式:

$$\frac{d}{l} = \frac{d+f}{l+a}, \frac{d}{m} = \frac{d+f}{m+c}, d = \frac{f \times b}{a+c} \quad (1.1)$$

已知 $b = l + m$, 由上式可得观察者双眼至物体的距离: 由于双眼的距离 b 为已知, 焦距 f

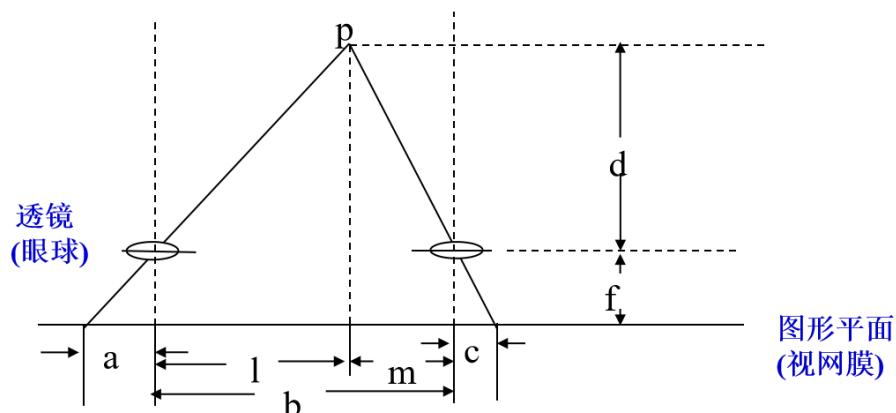


图 1-32 示意图

也是确定的, 因此 d 是可直接计算出来的.

定义 1.30 模式识别

是指让计算机能够对给定的事务进行鉴别，并把它归入与其相同或相似的模式中。被鉴别的事物可以是物理的、化学的、生理的，也可以是文字、图像、声音等。

模式识别的一般过程 (1) 采集待识别事物的模式信息；

(2) 对其进行各种变换和预处理，从中抽出有意义的特征或基元，得到待识别事物的模式；

(3) 与机器中原有的各种标准模式进行比较，完成对待识别事物的分类识别；

(4) 输出识别结果。

自然语言处理包括的主要内容：

► 机器翻译：把一种自然语言翻译成另外一种自然语言。

► 自然语言理解主要研究如何使计算机能够理解和生成自然语言。

理解的语言类型：声音语言、书面语言。

主要步骤：语音分析、词法分析、句法分析、语义分析、语用分析。

自然语言理解的意义

该研究不仅对智能人机接口有着重要的实际意义，而且对不确定人工智能的研究也具有重大的理论价值。有学者指出：人工智能如果不能用自然语言作为其知识表示基础，建立起不确定人工智能的理论和方法，人工智能也就永远实现不了跨越的梦想。

1.6.3 机器行为

定义 1.31 机器行为

就是让计算机能够具有像人那样地行动和表达能力，如走、跑、拿、说、唱、写画等。

机器行为则可看作智能系统的输出部分。主要讨论内容为智能控制、智能检索和智能机器人等。

定义 1.32 智能控制

是指那种无需或需要尽可能少的人工干预就能独立的驱动智能机器实现其目标的控制过程。它是人工智能技术与传统自动控制技术相结合的产物。

- ▶ 智能控制系统: 是指那种能够实现某种控制任务, 具有自学习、自适应和自组织功能的智能系统. 从结构上, 它由传感器、感知信息处理模块、认知模块、规划和控制模块、执行器和通信接口模块等主要部件所组成.
- ▶ 智能控制的主要应用领域: 智能机器人系统、计算机集成制造系统 (CIMS)、复杂工业过程的控制系统、航空航天控制系统、社会经济管理系统、交通运输系统、环保及能源系统等.

定义 1.33 智能检索

是指利用人工智能的方法从大量信息中尽快找到所需要的信息或知识.



智能检索的重要性: 目前, 在各种数据库中, 尤其是互联网上存放着大量的、甚至是海量的信息或知识. 面对这种信息海洋, 如果还用传统的人工方式进行检索, 已很不现实.

智能检索系统须解决的主要问题:

- (1) 具有一定的自然语言理解能力, 能理解用自然语言提出的各种询问;
- (2) 具有一定的推理能力, 能够根据已知的信息或知识, 演绎出所需要的答案;
- (3) 系统应拥有一定的常识性知识, 以补充学科范围的专业知识. 系统根据这些常识, 将能演绎出更一般询问的一些答案. 机器人 (Robots) 和机器人学: 机器人 (Robots) 是一种可再编程的多功能操作装置. 机器人学是在电子学、人工智能、控制论、系统工程、精密机械、信息传感、仿生学、以及心理学等多种学科或技术发展的基础上形成的一种综合性技术学科.

机器人的研究意义: 机器人既是人工智能的研究对象, 同时又是人工智能的试验场地, 人工智能的所有技术几乎都可以在这个领域得到应用.

机器人的发展过程: 经历了遥控、程序、自适应、智能机器人、情感机器人. 人工智能的主要研究对象是智能机器人和情感机器人.

智能机器人具有的能力: 感知能力、思维能力和行为能力的机器人. 这种机器人能够主动的适应外界环境变化, 并能够通过学习丰富自己的知识、提高自己的工作能力.

情感机器人: 是一种具有情感 (爱、恨...) 和情绪 (喜、怒、哀、乐...) 功能新一代机器人.

1.6.4 计算智能

计算智能 (Computational Intelligence, CI) 是借鉴仿生学的思想, 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. 计算智能的三大基本领域包括神经计算、进化计算、模糊计算.

1.6.4.1 神经计算

定义 1.34 神经计算

亦称神经网络 (Neural Network, NN), 它是通过对大量人工神经元的广泛并行互联所形成的一种人工网络系统, 用于模拟生物神经系统的结构和功能.



主要研究内容: 包括人工神经元的结构和模型, 人工神经网络的互连结构和系统模型, 基于神经网络的联结学习机制等.

- ▶ 人工神经元: 是指用人工方法构造单个神经元, 它有抑制和兴奋两种工作状态, 可以接受外界刺激, 也可以向外界输出自身的状态, 用于模拟生物神经元的结构和功能, 是人工神经网络的基本处理单元.
- ▶ 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. 从互连结构的角度, 神经网络可分为前馈网络和反馈网络两种主要类型.
- ▶ 网络模型是对网络结构、连接权值和学习能力的总括. 最常用的有传统的感知器模型, 具有误差前向传播功能的前向传播网络模型, 采用反馈连接方式的反馈网络模型等.
- ▶ 神经网络具有自学习、自组织、自适应、联想、模糊推理等能力, 在模仿生物神经计算方面有一定优势. 目前, 神经计算的研究和应用已渗透到许多领域, 如机器学习、专家系统、智能控制、模式识别等.

1.6.4.2 进化计算

定义 1.35 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、变异、竞争和选择引入到了算法中, 是一种对人类智能的演化模拟方法.



进化计算的主要分支: 遗传算法、进化策略、进化规划和遗传规划四大分支. 其中, 遗传算法是进化计算中最先形成的一种具有普遍影响的模拟进化优化算法.

遗传算法的基本思想: (美国密执安大学霍兰德教授 1962 提出) 是使用模拟生物和人类进化的方法来求解复杂问题. 它从初始种群出发, 采用优胜略汰、适者生存的自然法则选择个体, 并通过杂交、变异产生新一代种群, 如此逐代进化, 直到满足目标为止.

1.6.4.3 模糊计算

定义 1.36 模糊计算

亦称模糊系统, 是通过对人类处理模糊现象的认知能力的认识, 用模糊集合和模糊逻辑去模拟人类的智能行为的。模糊集合与模糊逻辑是美国加州大学扎德 (Zadeh) 教授 1965 年提出出来的一种处理因模糊而引起的不确定性的有效方法。

定义 1.37 模糊

人们把那种因没有严格边界划分而无法精确刻画的现象称为模糊现象, 并把反映模糊现象的各种概念称为模糊概念。

例 1.38 “大”、“小”、“多”、“少”等。

- ▶ 模糊概念的表示: 通常是用模糊集合来表示的, 而模糊集合又是用隶属函数来刻画的。一个隶属函数描述一个模糊概念, 其函数值为 $[0, 1]$ 区间的实数, 用来描述函数自变量所代表的模糊事件隶属于该模糊概念的程度。
- ▶ 模糊计算的争论: 一方面模糊逻辑存在一定缺陷; 另一方面它在推理、控制、决策等方面得到了非常广泛的应用。

1.6.5 机器学习

- ▶ 机器学习就是让计算机能够像人那样自动地获取新知识, 并在实践中不断地完善自我和增强能力。
- ▶ 机器学习是机器获取知识的根本途径, 同时也是机器具有智能的重要标志。
- ▶ 机器学习有多种不同的分类方法, 如果按照对人类学习的模拟方式, 机器学习可分为符号学习和神经学习等。

1.6.5.1 符号学习

- ▶ 符号学习的概念: 是指从功能上模拟人类学习能力的机器学习方法, 它是一种基于符号主义学派的机器学习观点。
- ▶ 符号学习的类型: 可根据学习策略, 即学习中所使用的推理方法, 将其分为记忆学习、归纳学习、演绎学习等。
- ▶ 记忆学习也叫死记硬背学习, 它是一种最基本的学习方法, 原因是任何学习系统都必须记住它们所获取的知识, 以便将来使用。

- ▶ 归纳学习是指以归纳推理为基础的学习,它是机器学习中研究得较多的一种学习类型,其任务是要从关于某个概念的一系列已知的正例和反例中归纳出一个一般的概念描述.

例 1.39 示例学习和决策树学习.

- ▶ 演绎学习是指以演绎推理为基础的学习,解释学习是一种演绎学习方法,它是在领域知识的指导下,通过对单个问题求解例子的分析,构造出求解过程的因果解释结构,并对该解释结构进行概括化处理,得到一个可用来求解类似问题的一般性知识.

1.6.5.2 神经学习

定义 1.40 神经学习

神经学习也称为连接学习,它是一种基于人工神经网络的学习方法.现有研究表明,人脑的学习和记忆过程都是通过神经系统来完成的.在神经系统中,神经元既是学习的基本单位,同是也是记忆的基本单位.



神经学习的类型:

- ▶ 感知器学习实际上是一种基于纠错学习规则,采用迭代的思想对连接权值和阈值进行不断调整,直到满足结束条件为止的学习算法.
- ▶ BP 网络学习是一种误差反向传播网络学习算法.这种学习算法的学习过程由输出模式的正向传播过程和误差的反向传播过程所组成,其中,误差的反向传播过程用于修改各层神经元的连接权值,以逐步减少误差信号,直至得到所期望的输出模式为止.
- ▶ Hopfield 网络学习实际上是要寻求系统的稳定状态,即从网络的初始状态开始,逐渐向其稳定状态过渡,直至达到稳定状态为止.至于网络的稳定性,则是通过一个能量函数来描述的.

1.6.5.3 数据挖掘和知识发现

定义 1.41 知识发现和数据挖掘

知识发现和数据挖掘是在数据库的基础上实现的一种知识发现系统。它通过综合运用统计学、粗糙集理论、模糊数学、机器学习和专家系统等多种学习手段和方法，从数据库中提炼和抽取知识，从而可以揭示出蕴含在这些数据背后的客观世界的内在联系和本质原理，实现知识的自动获取。



与传统数据库技术的区别 传统数据库技术仅限于对数据库的查询和检索，不能够从数据库中提取知识。知识发现和数据挖掘以数据库作为知识源去抽取知识，不仅可以提高数据库中数据的利用价值，同时也为各种智能系统的知识获取开辟了一条新的途径。

发展 随着大规模数据库和互联网的迅速发展，知识发现和数据挖掘也从面向数据库的结构化信息的数据挖掘发展到面向数据仓库和互联网的海量、半结构化或非结构化信息的数据挖掘。

1.6.6 分布智能

定义 1.42 分布智能

分布智能主要研究在逻辑上或物理上分布的智能系统之间如何相互协调各自的智能行为，实现问题的并行求解。



分布智能的两个主要方向：

- ▶ 分布式问题求解主要研究如何在多个合作者之间进行任务划分和问题求解，它一般是针对某一问题去创建一个能够进行合作求解的协作群体。
- ▶ 多 Agent 系统主要研究如何在一群自主的 Agent 之间进行智能行为的协调，它不限于单一目标，可创建一个能够共同处理单个目标或多个目标的智能群体。
- ▶ 多 Agent 系统的组成与工作：它由多个自主 Agent 所组成，其中的每个 Agent 都可以自主运行和自主交互，即当一个 Agent 需要与别的 Agent 合作时，就通过相应的通信机制去寻找可以合作并愿意合作的 Agent，以共同解决问题。

1.6.7 智能系统

定义 1.43 智能系统

智能系统可以泛指各种具有智能特征和功能的软硬件系统。从这种意义上讲，前面所讨论的不少研究内容都应以智能系统的形式来出现，例如智能控制系统、智能制造系统、智能检索系统等。这里主要介绍除前述研究内容以外的专家系统和智能决策支持系统。

1.6.7.1 专家系统

定义 1.44 专家系统

专家系统是一种基于知识的智能系统，它将领域专家的经验用知识表示方法表示出来，并放入知识库中，供推理机使用。

随着计算网络、多 Agent、计算智能等技术的发展，出现了模糊专家系统、神经网络专家系统、基于 Web 的专家系统、协同式专家系统和分布式专家系统等。



图 1-33 专家系统结构

1.6.7.2 智能决策支持系统

定义 1.45 智能决策支持系统

智能决策支持系统是指那种在传统决策支持系统中增加了相应的智能部件的决策支持系统。

智能决策支持系统是把人工智能技术，尤其是专家系统技术与决策支持系统相结合的产物，具有很宽的应用范围和很好的应用前景。



图 1-34 智能决策支持系统

1.6.8 人工心理与人工情感

智能、情感和心理

- ▶ 智能: 是指感知、记忆、思维、学习、自适应、行为等能力
- ▶ 情感: 指人对客观现实的态度的体验.
 - 情绪 (侧重于生理现象: 喜、怒、哀、乐…)
 - 情感 (侧重于价值判断: 爱、恨…)
 - 情操 (高级的情感现象: 道德、理智、审美…)
- ▶ 心理: 认知、情感、意志
 - 认知: 实践活动中对认知信息的接收、编码、存储、提取、使用; 包括感知、思维、记忆等.
 - 情感: …
 - 意志: 自觉地确定目的, 并根据目的调节支配自身的行动, 克服困难, 去实现预定目标

人工智能、人工情感和人工心理

- ▶ 人工智能: 研究、开发用于模拟、延伸和扩展人的智能的理论、方法、技术及应用系统的一门新的技术科学.
- ▶ 人工情感: 人工情感 (Artificial Emotion) 是利用信息科学的手段对人类情感过程进行模拟、识别和理解, 使机器能够产生类人情感并与人类进行自然和谐地人机交互的研究领域.
- ▶ 人工心理: 人工心理 (Artificial Psychology) 就是利用信息科学的手段, 对人的心理活动 (着重是人的情感、意志、性格、创造) 的更全面再一次人工机器 (计算机、模型算法等) 模拟, 其目的在于从心理学广义层次上研究人工情感、情感与认知、

动机与情感的人工机器实现问题.

1.6.9 人工生命

人工生命 (Artificial Life) 是美国洛斯 · 阿拉莫斯 (Los Alamos) 非线性研究中心克里斯 · 兰顿 (Chris Langton) 于 1987 年在研究 “混沌边沿” 的细胞自动机提出的一个概念.

• 人工生命是通过对自然现象的模拟来研究行为如何变得智能、自适应的学科, 以及复杂的行为如何产生. 研究内容就是要研究能够展示人类生命特征的人工系统. 即研究以非碳水化合物为基础的、具有人类生命特征的人造生命系统.

- 人工生命的研究目标就是要创造出具有人类生命特征的人工生命.
- 人工生命研究并不十分关心已经知道的以碳水化合物为基础的生命的特殊形式, 即 “生命之所知 (Life as we know it)”, 它主要是生物学研究的主题.
- 人工生命最关心的是生命的存在形式, 即 “生命之所能 (Life as it could be)”. 生命之所能, 是人工生命研究所关心的主要问题.

按照这种观点, 如果能从具体的生活中抽象出控制生命的 “存在形式”, 并且这种存在形式可以在另外一种物质中实现, 那么就可以创造出基于不同物质的另外一种生命——人工生命.

人工生命的主要研究内容主要包括计算机进程、细胞自动机、人工脑和进化机器人等. 其中, 进化机器人不同于传统意义上的机器人, 它是一种利用计算机和非有机物质构造出来的具有人类生命特征的人工生命实体.

1.6.10 自动推理

近 20 年来, 几何定理机器证明的研究和实践有了很大的进展. 几何定理机器证明和非线性代数方程组作为主攻方向, 一方面是因为吴文俊先生在 70 年代的突出工作, 使中国在此方向上具有了领先的优势; 另一方面, 这两个方向有鲜明的应用背景, 近年来在机器证明领域也确是十分活跃的, 值得重视. 由于传统的兴趣和多种原因, 几何定理的机器证明在自动推理的研究中占有重要的地位.

建立一个通用的几何解题方法, 成批地解决问题, 以至万理一证, 是历史上一些卓越科学家的梦想. 为此, 笛卡尔发明了坐标系; 莱布尼兹设想过推理机器; 希尔伯特在其名著 (几何基础) 中给出了一类几何命题的机械化定理. 电子计算机的出现推动了数学机械化. 50 年代, 塔斯基用代数方法证明了初等几何的机械化的可能性. 到 60 年代, 斯拉格和莫色斯实现了符号积分, 代数与分析计算问题的机械化已经初具规模, 而几何定理的机器证明看来仍遥遥无期. 接着, 格兰特等提出用逻辑方法建立几何推理机, 科林斯等改进了塔斯基的代数方法. 直到 1975 年, 仍找不到能用计算机判定非平凡几何命题的有效算

法。吴文俊方法的提出给定理机器证明的研究带来勃勃生机。用吴法可在微机上很快地证明困难的几何定理。周咸青发展了吴法并把它实现为有效的通用程序，证明了 512 条非平凡定理（1981 年开始），写成英文专著。这一进展是自动推理领域的一大突破，被国际同行誉为革命性的工作。

（[吴方法](#)）的成功使几何定理机器证明研究活跃起来。用代数方法证明几何定理的方向受到重视。新的代数方法接连出现。在国外，周咸青等提出了用 Grobner 基方法构造几何定理机器证明的算法和程序并获得成功。在国内，洪家威提出了单点例证方法的理论设想，但因复杂度太大不能实现。张景中、杨路则提出数值并行方法，在低档微机（甚至计算器）上实现了非平凡几何定理的机器证明和机器发明。数值并行方法的优点是所需内存极小，且易于并行化。所有这些方法都属于代数方法。它们的提出和实现丰富了几何定理机器证明的研究。但与吴法相比，没有大的新突破。代数方法不能使人满意的是，它所给出的“证明”是关于多项式的繁复的计算，人难于理解其几何意义，也难于检验其是否正确。能否让计算机生成人能理解和易于检验的简明巧妙的证明，即所谓可读证明，是对自动推理和人工智能领域的一个挑战性的课题。一些著名的科学家认为，机器证明的基本思想是以量的复杂取代质的困难，这就很难想象用机器生成可读证明。国外一些学者从 60 年代即致力于几何定理可读证明自动生成的研究，30 多年来进展不大，未能给出哪怕是一小类非平凡几何定理的机器证明的有效算法和程序。作者以多年来所发展的几何新方法为基本工具，并提出了消点思想，和周咸青、高小山合作，于 1992 年突破了这一困难，实现了几何定理可读证明的自动生成。这一新方法既不以坐标为基础，也不同于传统的综合方法，而是一个以几何不变量为工具，把几何、代数、逻辑和人工智能方法结合起来所形成的开放系统。它选择几个基本的几何不变量和一套作图规则，并且建立一系列与这些不变量和作图规则有关的消点公式。当命题的前提以作图语句的形式输入时，程序可调用适当的消点公式把结论中的约束点逐个消去，最后达到水落石出。消点的过程纪录与消点公式相结合，就是一个具有几何意义的证明。此算法对可构造等式型几何命题是完全的，但其应用范围不限于这一类命题。基于此法所编的程序，已在微机上对数以百计的困难的几何定理完全自动地生成了简短的可读证明，其效率也比其他方法为高。随所用的几何量的不同，它能生成面积法、向量法、复数法和全角法等多种风格的证明，也能用于立体几何。杨路、高小山、周咸青与作者合作，把消点法用于非欧几何可读证明的自动生成也获得成功，并得到一批非欧几何新定理。消点法也可用于几何计算和公式推导。基于几何量和消点思想的新原理的建立，像是打开了几何定理机器求解的一个矿床。它也使几何定理机器证明的成果在数学教育中的应用有了现实可能。这一成果被国际同行誉为使计算机能像处理算术那样处理几何的发展道路上的里程碑，是自动推理领域 30 年来最重要的工作。在多数情形下，消点法也可用笔纸证明不平凡的定理。它

结束了两千年来几何证题无定法的局面, 把初等几何解题法从四则杂题的层次推进到代数方程的阶段.

以吴方法为代表的代数方法仅仅能够判断几何命题的成立与否, 证明的过程十分复杂, 而且需要进行大量的数值计算和符号计算, 这与传统几何证明的简洁明了大相径庭. 人们难以读懂这种方法生成的证明, 往往只能得出命题真假的结论, 因此很多人难以接受这种证明的风格. 如何生成让人容易读懂的几何证明过程这一问题成为科学家面临的又一个严峻的挑战. 1992年, 中科院院士张景中教授以其多年研究的面积法为基础, 提出了几何定理机器证明的新方法, 基于几何不变量的消点法. 随后, 它与周咸青、高小山合作完善了该方法, 并编写了程序, 终于成功的利用计算机对大量非平凡的几何命题生成了简洁易读的几何证明, 这一杰出的工作被誉为计算机处理几何问题的里程碑. 消点法包括一组构图规则、一组几何不变量以及一组消点公式. 该方法的基本思想是: 利用构图规则将欲证几何命题中涉及的图形构造出来, 并在构图的过程中生成关于点的约束条件, 同时将欲求证的命题表示成图中几何量的等式的形式, 然后利用消点公式, 按照点在作图时出现的相反顺序, 依次从结论等式中消去, 最终结论等式会化为显然成立的等式. 后来, 杨路教授又将消点法拓展到非欧几何, 成功的证明和发现了大量的新的非欧几何定理. 李洪波博士、杨海圈博士也在面积不变量的基础上提出用向量法实现几何定理的可读证明, 即 Clifford 代数法, 也取得了很好的效果.

近年来, 国外一再提出新的思路和算法, 欧共体还投资数百万美元组织项目专门研究非线性代数方程组的解法, 但均无突破性进展. 最近, 基于我们提出并加以完善的新的理论和算法——结式矩阵法, 符红光编写了代数方程组符号求解和机器证明的 MAPLE 程序.

机器证明的成果, 特别是非线性代数方程组理论与算法的研究成果, 将在数学、物理和工程技术中得到更多的应用. 目前, 在几何定理机器证明方面, 中国处于国际领先地位.

在非线性代数方程组研究领域, 竞争激烈, 中国已进入先进行列, 但还不能说领先.

在数学机械化软件开发方面, 由于起步晚、队伍小和资金不足等原因, 中国远不及欧美先进国家. 要把力量集中到非线性代数方程组的方向上来, 特别应加强对实用而有效的算法的研究.

在数学机械化推广应用方面, 也应投入力量, 发挥我国在理论与算法方向的优势, 在软件开发方面赶超先进. 在几何定理机器证明成果的基础上, 开发高智能的教育软件和自主版权的符号演算软件, 为中国科技、教育事业作出贡献 (MAPLE, Mathematica, Maxima, Magma, REDUCE, Macsyma).

1.6.11 图神经网络——Open Graph Benchmark

Jure Leskovec 教授在 NeurIPS 2019 大会演讲中宣布开源 Open Graph Benchmark (百万量级 OGB 基准测试数据集), 这是图神经网络建模统一基准迈出的重要一步, 是一个公认的基准测试数据集.

图神经网络是近来发展较快的机器学习分支领域. 通过将非结构化数据转换为结构化的节点和边的图, 然后采用图神经网络进行学习, 往往能够取得更好的效果. 采用的方法往往是针对较小的、缺乏节点和边特征的数据集上进行的, 因此, 在这些数据集上取得的模型性能很难说是最好的, 也不一定可靠.

在 NeurIPS 2019 大会的图表示学习演讲中, Jure Leskovec 宣布开源图神经网络的通用性能评价基准数据集 OGB(Open Graph Benchmark). 通过这一数据集, 可以更好地评估模型性能等方面的指标. [项目地址](#). [图表示学习演讲合集](#)

1.6.12 工业 4.0

工业 4.0 最初在 2011 年由德国提出, 是指为促进工业制造数字化而制定的高科技项目战略, 从而打造完全自动化的制造行业. 目前, 工业 4.0 已经应用了最先进的科学技术: 云计算、物联网、大数据、射频识别、协同开发等. 其中以物联网技术和大数据技术最为知名.

1.6.12.1 物联网技术与工业 4.0

物联网 (Internet of Things, 简称 IoT) 最初由美国麻省理工学院工业自动识别中心的创始人凯文阿什顿在 1999 年提出, 指将一切通过高度智能化的交互系统连接, 从而形成自动化的世界. 这里的“智能”指利用先进的通信和互联网技术, 有效处理信息, 并形成“智能产业”. 物联网设备之间的交互动作, 让它们彼此获得信息, 使得设备自身可以监控业务流程、提高生产效率, 进而节省成本, 做出更好地决策. 物联网技术可以被分为四个阶段:

- 1) 数据传感阶段: 工业生成的数据通过传感器感测, 并收集传输到最近的基站等待处理;
- 2) 整理加工阶段: 对数据进行整理和处理, 并根据需要进行相应转换;
- 3) 预处理阶段: 利用边缘技术处理系统在数据传输到中心前进行预处理;
- 4) 存储维护阶段: 生成的数据被存储维护, 为后续分析建立基础;

物联网技术的四个阶段说明, 具有高度感测技术的自动化装置是工业物联网的基础组成部分, 这些智能机器的操作将增加工业生产的灵活性, 影响生产者对智能管理的依

赖性，并为行业发展设定新的标准。物联网技术能满足数字市场的快速发展和消费者需求的不断增加，在不需要人工干预的情况下，及时、准确地完成分配的工作。使用物联网的行业运营效率更高，更能了解客户的需求，最终提高盈利能力。

1.6.12.2 大数据与工业 4.0

工业发展导致系统中出现了巨大的数据流，这些数据的保护、处理和维护成为人们关注的焦点，大数据技术（Big Data Technology，简称 BDT）由此而生。大数据最初来自 IBM 数据科学家，从字面上讲，意味着大量信息的数据集合，它是一个分析海量数据的概念。大数据这个概念有五个维度：

- 1) 体量：大数据代表数据数量上远比之前庞大；
- 2) 多样：不同的数据源，生成了不同结构的数据；
- 3) 速度：数据生成的速度快，分析处理的速度也快；
- 4) 真实：数据的真实性是分析的前提；
- 5) 价值：通过分析数据得到的行动是有价值的；

目前，大数据已经成为各种业务数字化的基础，它在收集、传输、分析和使用大量实时数据的同时，生成的信息提供了智能流程的改进思路，从而保障工业生产高效、无故障地执行。适当收集和分析大数据将提高产品制造、供应链管理、物流和风险管理等多个部门的竞争力。

VUCA(发音近似于乌卡，所以也用“乌卡”代表)战略指的是前言中提到的易变性、不确定性、复杂性和模糊性，最初的 VUCA 战略发源于军事领域，美军在 20 世纪 90 年代，引用来描述冷战结束后的越发不稳定的、不确定的、复杂、模棱两可和多边的世界。但是现在已经广泛应用于商业分析中。VUCA 战略通过分析预期的发展风险因素，并对这些情况加以准备，从而在竞争激烈的商业世界中生存。

VUCA 战略具有如下四个维度：

易变性（Volatility）：指商业环境中的极端和快速变化。这些变化的速度、数量和幅度可以反映它在商业环境中的波动程度。产生易变性的原因往往是已知的，比如价格的易变性会导致供应链的风险，商品供给的易变性将导致公司无法满足消费者需求；

不确定性（Uncertainty）：由于易变性的存在，在缺乏了解的时候将会产生不确定性，从而导致未来不可预测，影响长期发展。在商业环境中，可能引起不确定性的因素包括：用户需求、偏好的变化，新政策的提出，新产品对旧产品的替代等；

复杂性（Complexity）：复杂性的产生来自两个方面，一是工业化的快速发展，使企业内部相互联系的网络和程序逐渐复杂，二是外部的商业环境的不确定性导致决策的复杂性。外包活动（如会计核算、市场营销和计算机辅助设计业务）的引入，导致复杂性更

加普遍;

(**模糊性 (Ambiguity)**): 在商业行动中, 模糊性表现无法清晰陈述、无法准确评估概率以及无法描述潜在结果的多样性, 当新产品或者新计划被引入市场时, 模糊性出现的可能性更高.

 **注 1.46.** 模糊数学中提到的不确定性和模糊性的含义不用于上述机器学习中的相应概念.

1.6.12.3 机器学习下的 VUCA 战略

在这一部分中, 介绍几种常用的机器学习算法并简要描述它们的原理, 并举例说明它们在工业 4.0 的 VUCA 战略的现实应用.

(1) (**线性回归模型**): 线性回归是回归算法中一种有监督的机器学习算法, 它根据给定的变量来预测结果, 得到两者的线性关系. 常用的线性回归模型有两种, 分别是单个变量的简单线性和多变量的多元线性回归. 线性回归模型主要解决 VUCA 战略的易变性和不确定性问题. 谷歌公司的应用软件已经利用线性模型分析特定道路的历史数据, 来预测交通状况. 一些金融从业人员利用波动率指标建立线性模型来预测金融市场的波动性, 并证明该方法优于传统的移动平均法.

(2) (**Logistic 回归模型**): 是分类算法中一种有监督的机器学习算法, 以给定的变量作为输入值, 以 0 或 1、是或否、真或假等离散值预测输出结果. 虽然是回归模型, 但解决的是分类问题, 通过给出预测数据所属类的概率来完成判别. Logistic 回归模型主要解决 VUCA 战略的复杂性和模糊性问题. 医疗行业用这个方法将病人分为关键诊断和非关键诊断类别, 金融行业据此建立预警模型, 根据过去的债务、违约情况和收入, 判断用户是否会在业务中违约. 回归算法有效地处理了复杂和模糊的风险因素, 预测结果具有较高的精度.

(3) (**决策树模型**): 一种有监督的机器学习算法, 分为**分类算法**和**回归算法**两类. 数据集被分为具有相同类别的较小部分, 直到所有的数据都被分类, 且节点是最终的决策节点. 决策树由熵、信息增益来构成, 以预测事件的不确定性程度. 决策树模型主要解决 VUCA 战略中的不确定性和模糊性问题. 例如, 大型工业项目由于规划设计复杂, 且参与的群体多样, 存在很大的不确定性, 利用决策树模型可以对这些不确定因素进行分类, 提前预测风险.

(4) (**随机森林模型**): 一种有监督的机器学习算法, 通过随机收集决策树来预测期望的结果, 从而创造“森林”, 随着决策树生长, 每个决策树都可以对新对象分类并投票, 最高票数将对随机森林的过程分类. 随机森林模型主要解决 VUCA 战略中的模糊性问题. 它

有助于估计公司不稳定的业务绩效指标,如预测机械零件故障的可能性、估算市场的盈利能力并最小化风险,在预测灾害损失方面也有应用,且效果优于其他机器学习算法.

(5) **支持向量机模型**: 分类算法范畴下的有监督机器学习算法,通过生成一个平面或者决策边界将样本分成不同的类,数据样本点根据不同的特征进行分类,每个点都有不同的坐标作为支持向量. 支持向量机模型主要处理易变性、复杂性和不确定性问题. 供应链管理系统的需求预测、工业材料风险对冲模型等都涉及到了这些内容.

工业 4.0 通过物联网和大数据技术实现自动化数据交换,通过云计算实现数据存储和处理,通过认知计算帮助人类决策. 它降低了人力成本,简化了业务流程,提高了生产预测的准确性. 这些改进将显著提高生产效率和收入,有助于经济增长. 然而,工业 4.0 的进程中仍存在易变性、不确定性、复杂性和模糊性问题,这些问题影响了工业发展进程,需要得到合理的解决. 机器学习技术可以让智能设备在不需要人工编程的情况下做出决定,从而减少不确定因素导致的风险,在未来将被广泛应用.

1.6.12.4 基于算法的工业智能平台

将成为应用场景的重要基石. 不同工业行业有各自独特的行业门槛,每个工业场景在不同行业、不同企业中的需求差异较大. 人工智能与制造业深度融合的路径就是将信息技术与工业场景应用端结合. 将核心工艺模型化、算法化、代码化的工业智能算法平台面向工业场景,可以为底层应用提供便捷的开发服务. 然而市面上有更智能的算法平台,例如中机云集团的智能开发者平台,无需编码一拖一拽即可快速生成应用程序的工具,可以降低企业应用开发人力成本,从而帮助企业实现降本增效、灵活迭代.



图 1-35 基于大数据的工业智能的场景和智能开发者平台(中机云集团)

1.7 人工智能的典型应用

从理论到技术, 从产品到工程, 从家庭到社会, 从地下到太空, 智能无处不在, 人工智能的应用领域已非常广泛.

例 1.47 智能 CAD、智能 CAI、智能产品、智能家居、智能楼宇、智能社区、智能网络、智能电力、智能交通、智能控制、智能优化和智能空天技术等.

下面简单介绍其中的典型应用.

1.7.1 博弈

- ▶ 博弈的概念: 是一个有关对策和斗智问题的研究领域. 例如, 下棋、打牌、战争等这一类竞争性智能活动都属于博弈问题.
- ▶ 博弈的例子: 国际上对博弈的研究主要以下棋为对象, 其中代表性成果是 IBM 公司研制的 IBM 超级计算机“深蓝”和“小深”. 国内, 2006.8.9 在北京举办的首届中国象棋人机大赛中, 计算机以 3 胜 5 和 2 负(比分 11:9)的微弱优势战胜人类象棋大师.
- ▶ 研究博弈的目的: 不完全是为了让计算机与人下棋, 而主要是为了给人工智能研究提供一个试验场地, 同时也为了证明计算机具备智能. 试想, 连国际象棋世界冠军都能被计算机战败或者平局, 可见计算机所具备了何等的智能水平.

1.7.2 深度学习求解高维偏微分方程

传统的求解 PDE 维数通常是二维或三维, 但在比如说金融学等领域, 通过数学建模构建出来的 PDE 的维数极其之高, 维数升高带来的“维数灾难”问题亟待解决. 深度学习的方法可以处理一般的高维抛物型方程, 用反向随机微分方程构造 PDE, 并用神经网络近似未知解的梯度, 来求解高维的偏微分方程. 高维偏微分方程主要来自如下几个方面:

- ▶ 量子多体问题的薛定谔方程, 在这种情况下, PDE 的维数大约是系统中电子或者量子(粒子)的三倍.
- ▶ 用于给金融衍生品定价的 Black-Scholes 方程, 其中 PDE 的维数是所考虑的相关金融资产的数量.
- ▶ 动态规划中的 Hamilton-Jacobi-Bellman 方程. 在具有多个代理的博弈论中, 维数随着代理的数量线性增加. 类似地, 在资源分配问题中, 维数随着设备和资源的数量线性增加.



图 1-36 象棋比赛，“深蓝”击败 Kasparov

这些问题的计算成本随维度呈指数增长！泛逼近定理保证深度学习网络能表示任意函数，因此，基本思路就是随机微分方程中某个梯度算子用神经网络来表示。

例 1.48 考虑一类半线性抛物型偏微分方程：

$$\begin{aligned} \frac{\partial u}{\partial t}(t, x) + \frac{1}{2} \operatorname{Tr} (\sigma \sigma^T(t, x) (\operatorname{Hess}_x u)(t, x)) + \nabla u(t, x) \cdot \mu(t, x) \\ + f(t, x, u(t, x), \sigma^T(t, x) \nabla u(t, x)) = 0. \end{aligned} \quad (1.2)$$

且有指定的末端条件 $u(T, x) = g(x)$ ，这里的 x 是 d 维的， μ 是已知的向量值函数， σ 是 d 维矩阵值函数， f 是非线性函数。

我们感兴趣的是 $t = 0, x = \xi$ 的解。让 X_t 是随机变量，满足如下的方程

$$X_t = \xi + \int_0^t \mu(s, X_s) ds + \int_0^t \sigma(s, X_s) dW_s. \quad (1.3)$$

根据结果 [Han2018PNAS, Sirignano2018, Raissi2017], 式(1.2)的解具有如下形式:

$$\begin{aligned} u(t, X_t) = & u(0, X_0) - \int_0^t f(s, X_s, u(s, X_s), \sigma^T(s, X_s) \nabla u(s, X_s)) ds \\ & + \int_0^t [\nabla u(s, X_s)]^T \sigma(s, X_s) dW_s. \end{aligned} \quad (1.4)$$

我们对式(1.3)和式(1.4)两个式子用欧拉格式离散, 得到:

$$X_{t_{n+1}} - X_{t_n} \approx \mu(t_n, X_{t_n}) \Delta t_n + \sigma(t_n, X_{t_n}) \Delta W_n. \quad (1.5)$$

想法就是用其他格式离散化原始的方程, 比如说 Milstein 格式等. 接下来是关键的步骤, 就是在数值格式中, 用多层反馈神经网络去逼近 $o(\nabla u)$ 项, 即:

$$\sigma^T(t_n, X_{t_n}) \nabla u(t_n, X_{t_n}) = (\sigma^T \nabla u)(t_n, X_{t_n}) \approx (\sigma^T \nabla u)(t_n, X_{t_n} | \theta_n). \quad (1.6)$$

注意到这里每一步都放进去一个神经网络, 最后整个堆叠成一个函数表达器. 这里边的 X_{t_n}, W_{t_n} 是变量. 我们可以用一般的深度学习方法来训练这个网络, 如随机梯度下降方法等. 还记得我们前面给了一个末端条件, 刚好可以用了定义损失函数.

$$l(\theta) = \mathbb{E} \left[\left| g(X_{t_N}) - \hat{u} \left(\{X_{t_n}\}_{0 \leq n \leq N}, \{W_{t_n}\}_{0 \leq n \leq N} \right) \right|^2 \right], \quad (1.7)$$

其中 θ_n 是神经网络的参数 (权重系数), 上述方法称为深度 BSDE 方法.

例 1.49 (带违约风险的 Black-Scholes 方程) 金融衍生品交易的一个关键问题就是确定适当而公平的价格. Black 和 Scholes 说明了衍生品的价格 u 满足一个抛物型的 PDE, 现在被称为 Black-Scholes 方程. Black-Scholes 模型可以加以考虑实际市场中的几个重要因素, 包括可违约证券, 借贷利率高于贷款利率, 交易成本, 模型参数的不确定性等等. 这几个效应中的每一个都会在定价模型中产生非线性的贡献. 特别是, 信贷危机和持续的欧洲主权债务危机凸显了原始的 Black-Scholes 模型中忽略的最基本风险, 即违约风险.

理想情况下, 定价模型应考虑金融衍生品所依赖的整个底层证券, 从而产生了高维非线性偏微分方程. 然而, 由于维数的诅咒, 现有的定价算法通常无法解决这些问题. 为了证明深度 BSDE 方法的有效性, 作者研究了一个具有违约风险的递归估值模型的特例. 我们考虑尚未发生违约的情况下, 100 个标的资产的欧洲债权的价格是公平的. 当违约的情况发生时, 索赔的持有人仅收到当前值的分数倍, 设为 $\delta \in [0, 1)$. 对于强度 Q (当前值的递减函数), 可能风险由泊松过程的第一个跳跃时间建模得到, 换言之, 当声明的值低时, 风险变得更加可能. 可以通过以下方式对价值变化过程进行建模:

$$f(t, x, u(t, x), \sigma^T(t, x) \nabla u(t, x)) = -(1 - \delta)Q(u(t, x))u(t, x) - Ru(t, x), \quad (1.8)$$

其中 R 是无风险资产的利率. 我们假设标的资产价格做几何布朗运动, 并选择强度函数 Q 为当前值的分段线性函数, 有三个区域 ($v_h < v_l, \gamma_h > \gamma_l$), 函数形式如下:

$$Q(y) = 1_{(-\infty, v^h)}(y)\gamma^h + 1_{[v^l, \infty)}(y)\gamma^l + 1_{[v^h, v^l)}(y)\left[\frac{(\gamma^h - \gamma^l)}{(v^h - v^l)}(y - v^h) + \gamma^h\right], \quad (1.9)$$

那么, 在 $[0, T] \times \mathbb{R}^{100}$ 上的非线性 Black-Scholes 方程就变为了:

$$\frac{\partial u}{\partial t}(t, x) + \bar{\mu}x \cdot \nabla u(t, x) + \frac{\bar{\sigma}^2}{2} \sum_{i=1}^d |x_i|^2 \frac{\partial^2 u}{\partial x_i^2}(t, x) - (1 - \delta)Q(u(t, x))u(t, x) - Ru(t, x) = 0. \quad (1.10)$$

设置如下参数, $T = 1, \delta = 2/3, R = 0.02, \mu = 0.02, \sigma = 0.2, v_h = 50, v_l = 70, \gamma_h = 0.2, \gamma_l = 0.02$, 终止条件 $g(x) = \min\{x_1, x_2, \dots, x_{100}\}$. 网络学习过程见图 1-37.

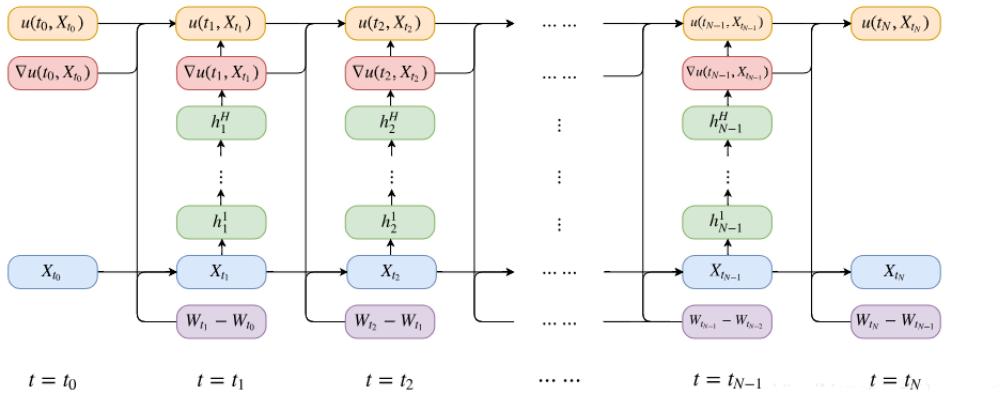


图 1-37 深度 BSDE 的结构

1.7.3 自动定理证明

自动定理证明的概念: 就是让计算机模拟人类证明定理的方法, 自动实现像人类证明定理那样的非数值符号演算过程. 它既是人工智能的一个重要研究领域, 又是人工智能的一种实用方法. 除数学定理外, 多非数学领域的任务如医疗诊断、难题求解等都可转化成定理证明问题.

注 1.50. 数学定理证明机械化的途径很多, “吴方法”是其中的代表. 对于初等平面几何, “吴方法”实质上就是“方程联立求证法”.

例 1.51 (方程联立求证法) 我们需要求证一个几何“事实”(或断语): 三线“共点”. 看起来这是显而易见的事情, 但是, 使用计算机自动证明这一事实却不容易.

• 吴方法的第一步: 实现“代数化”, 即建立坐标系, 把问题条件全部翻译成“代数方程式”(事项即“前提条件”), 然后, 把需要证明的事实也翻译成“代数方程式”(有待证明的事项). 吴方法的第二步: 方程联立求证.

• 把前提条件(方程组)与待证明的方程“联立求解”, 如果待证明的“方程”经过自动联立求解过程, 变为一个恒等式, 那么, 这个恒等式就表示“所有前提条件”满足待证的“结论”, 所以, 定理得到证明, 否则, 问题无解.

• 在国际学术界, “吴方法”独树一帜, 不随大流, 效率最高, 得到国际同仁赞扬与肯定. 经过多年人才积累, 在数学定理证明机械化研究领域, 研究业绩非凡, 国内“吴氏学派”最终形成.

• 中科院数学机械化重点实验室, 成员包括万哲先院士、陆汝钤院士、李邦河院士、张景中院士、林惠民院士等 10 多位实验室学术委员会成员.

• 吴文俊人工智能科学技术奖: 吴文俊是我国著名数学家、首届国家最高科学技术奖获得者. 吴文俊人工智能科学技术奖由中国人工智能学会发起设立, 设有成就奖、创新奖和进步奖, 每年评奖一次, 奖励在智能科学技术领域取得重大突破、作出卓著贡献的科技工作者和管理者. (转自科学网)

• 符号和代数计算方面最权威的国际会议——国际符号和代数计算会议(ISSAC).

• 2019 年是“吴文俊人工智能科学技术奖”的第九届.

 **注 1.52.** 首届国家最高科学技术奖获得者、中国科学院院士、中国人工智能学会名誉理事长吴文俊先生命名, 依托社会力量设立的科学技术奖. 由国家一级学会——中国人工智能学会发起主办, 被誉为“中国智能科学技术最高奖”, 至今已累计对 314 个单位及行业机构、291 个创新成果和项目、973 名学者及专家进行表彰.

第三届吴文俊人工智能科学技术奖于 2013 年 10 月 29 日在深圳揭晓.

- ① 中国科学院史忠植的“拓展知识工程核心理论, 创新分布智能理论基础, 构建智能科学理论体系”成果获吴文俊人工智能科学技术奖成就奖;
- ② 大连理工大学李洪兴的“模糊系统的概率表示与空间四级倒立摆的控制”成果;
- ③ 北京航空航天大学段海滨的“群体智能及其应用”成果分别获创新奖一等奖;
- ④ 清华大学唐杰的“研究者社会网络搜索与挖掘系统”项目获进步奖一等奖.
- ⑤ 北京大学刘宏在面向服务机器人的视听感知与人体运动分析领域的创新工作;
- ⑥ 西安电子科技大学的公茂果在自然计算理论及其在 SAR 图像解译中的应用领域的创新工作分别获得创新奖二等奖.

自动定理证明的主要方法:

- ▶ 自然演绎法: 其基本思想是依据推理规则, 从前提和公理中推出一些定理, 如果待证明的定理在恰在其中, 则定理得证. 这种方法的突出代表是纽厄尔等人研制的数学定理证明程序 LT 等.
- ▶ 判定法: 其基本思想是对某一类问题找出一个统一的、可在计算机上实现的算法. 其突出代表是我国数学家吴文俊院士提出的证明初等几何定理的算法. 其基本思想是把几何问题代数化, 即先通过引入坐标把几何定理中的假设和求证部分用一组代数方程表达出来, 然后再利用代数几何中的代数簇.
- ▶ 理论求解代数方程, 以证明定理的正确性.
- ▶ 定理证明器: 是一种研究一切可判定问题的证明方法. 其典型代表是 1965 年鲁宾逊提出的归结原理.
- ▶ 人机交互定理证明: 是一种通过人机交互方式来证明定理的方法. 它把计算机作为数学家的辅助工具, 用计算机来帮助人完成手工证明中难以完成的那些计算、推理、穷举等. 其典型代表是四色定理证明.

1.7.4 国家安全领域

2019 年 3 月, 全球安全研究中心劳伦斯利佛摩国家实验室 (CGSR Lawrence Livermore National Laboratory) 的扎卡里 · 戴维斯 (Zachary Davis, 加州蒙特利海军研究生院的教授) 高级研究员发布题为《战场上的人工智能》(《AI on the battlefield》) 的文章. 文

章中指出，人工智能突如其来地闯入国家安全领域，并被视为一种革命性技术，与发现燃料、电力或核武器不相上下。人工智能技术在一定程度上对美国产生了变革效应（例如，在科学和社交媒体中），在一定程度上这是由美国潜在对手驱动的。俄罗斯总统普京宣称，统治人工智能的国家将是世界的统治者。JQRCluster20200322112441.gif



图 1-38 未来战争

1.7.5 机器人

Spot 机器狗已经参与了“至少两次”实际执法活动。



图 1-39 波士顿机械狗

2020 年 2 月 13 日，伊朗推出迄今为止最先进的类人机器人 Surena 4（又称 Surena IV，图 1-40）。Surena 4 新机器人是对以前设计的重大改进。突出的功能是它能模仿人的姿势，抓住水瓶并将其名字写在白板上，Surena 4 还能自己步行走出学校大门，抓或者捧较大的球形物体。



图 1-40 Surena 4 运动图

Surena 由德黑兰大学机械工程学教授尤塞菲·科马博士在 50 多名研究人员的领导下在 CAST (先进系统和技术中心) 开发。Surena 机器人的第一代 (SURENA I, 2008 年) 只有 8 个自由度 (DoF)，第二代 (SURENA II, 2010 年) 有 22 个自由度，行走速度为每秒 0.03 米。与拥有 31 DoF 的第三代 (SURENA III, 2015) 相比，新型第四代机器人 (Surena IV, 2019) 具有 43 DoF 和更高的手部灵活性，使其能够抓握具有不同形状的不同物体。Surena 4 高 1.7 米，重 68 公斤；它比 Surena III (98 公斤高和 1.9 米高) 更轻巧，更小，这归功于其基于拓扑优化的更好的结构设计，紧凑的定制执行器设计以及 SLA 3D 打印技术。

在新一代 Surena 4 中，通过利用 FPGA 板，控制环路频率已提高到 200 Hz，从而可以实现在线控制器和估计器。通过机器人操作系统 (ROS)，状态监控，算法的实时实现以及多个程序的同时运行变得非常简单。Surena 4 具有面部检测和计数，物体检测和位置测量，活动检测，语音识别和语音合成的能力，从而实现了更好的语音用户界面。通过结合 AI 能力和全身运动计划，实现了在线抓握，面部和物体跟随以及动作模仿。Surena 4 可以以 0.7 公里/小时的速度连续行走。它可以通过其底部的新型接触传感器在不平坦的地面上行走。CAST 装备了在线接触控制器，可以在踩踏过程中调节脚的角度和位置。用于模拟机器人的运动并评估不同的场景 (凉亭和动物群)，包括侧向行走，向后行走，转身和推举恢复）。

1.7.6 机械臂



图 1-41 堆垛和切割钢铁建材

智能软体机器人



图 1-42 DNA 软体机器人的设计原理图

1.7.7 闲鱼垃圾评论过滤系统

闲鱼垃圾评论过滤系统(训练并推断 11 亿节点的图)用了最前沿的图卷积神经网络。这项研究获得了 ACM CIKM 2019 最佳应用论文奖,说明图卷积在传统任务中的强大潜力。垃圾信息过滤表面上它只是一个最简单的二分类问题,对于闲鱼这种开放性评论机制,评论的维度及角度非常多多样,再来筛选垃圾信息就非常困难了。常规做法是通过关键字判断垃圾信息,也有采用朴素贝叶斯等浅层模型,还有 TextCNN 等深度神经网络,很多算法都是从文本层面判断评论是不是垃圾评论。很多时候,光使用文本是不够的,闲鱼的场景的灰产和模型一直在对抗,垃圾评论变异得很快。闲鱼需要结合一些难以变异的特征判断评论是不是垃圾评论,包括发送这条评论的用户信息、接收这条评论的商品特征,甚至是发送这条信息的用户,他的其它评论行为;以及与它类似的文本都有什么特

征。要利用这些多模态信息与复杂的图结构信息，就需要更强大的前沿模型——图卷积神经网络。

目前评论过滤系统已经部署到了闲鱼使用环境中，每天能处理百万级的闲鱼评论，并在其他强有力深度学习模型基础上，额外筛选出一千多条非常隐秘的垃圾评论。阿里研究者表示：基于图卷积的垃圾信息筛选是一种非常通用的思想，它的应用范围远不止垃圾评论过滤，淘宝信息的知识产权保护、淘宝商品管控和用户恶意评价等方面都可以采用。图神经网络还是非常有前景的，加上图神经网络可以利用复杂图数据的结构信息和多模态属性信息，业务场景非常广。

闲鱼是国内最大的二手交易平台，我们可以浏览卖家发布的各种商品，并根据描述与评论选择合适的物品。然而，这个每天交易超过 20 万商品的平台，却会受到垃圾评论的困扰。这主要是因为它与淘宝不一样，淘宝只有买过商品才能评价，但是对于闲鱼，评论充当着买卖双方的沟通工具，很多评论行为发生在购买之前。正式这种提前沟通与议价的机制，为垃圾评论提供了合适的平台。想象一下，如果灰产用户在许多受关注的商品中留言自己的广告，这样岂不是非常「划算」？为此，阿里的研究者一直与垃圾评论做着对抗，垃圾评论越来越「隐秘」，而判别算法也越来越「聪明」。我们先看看广告评论怎样越来越隐秘：换个说法：使用不同的方式表达相同的意思，例如「拨打电话获得更多兼职信息」和「闲余时间挣点钱？联系我」，这两者都引导我们关注相同的兼职广告。关键字替换：使用少见的中文字符、笔误，甚至表情符号替换关键字，例如「加我的 VX/V/WX」都表示加我的微信。垃圾评论发布者的这些小技巧很容易欺骗一般的机器学习系统，与此同时，如果发布者发现这些方法不太管用，他们又会挖掘一些新技巧。因此这样总是防不胜防，已经部署的防控算法的效果也会逐步降低。所以如果是一个好的垃圾评论过滤系统，它首先要捕捉到现有的各种模式，与此同时还应该降低对抗行为对系统的影响。解决思路是什么？解决垃圾信息过滤的核心思想在于上下文，我们只有把文本信息放入对应的环境，才能准确判断它到底是不是垃圾评论。阿里研究者定义了两种上下文，即局部上下文和全局上下文。其中局部上下文包含发这条评论的买家特征及行为和这条评论对应的商品特征等信息，而全局上下文表示当前评论在全部评论中的扮演的角色 ([Spam Review Detection with Graph Convolutional Networks](#))。

以两种上下文信息为出发点，研究者设计了名为 GCN-based Anti-Spam System(GAS) 的垃圾评论过滤系统。如下所示为 GAS 的整体概览，其中模型会从左侧图抽取出表示商品、用户和评论的信息，从右侧抽取出类似评论表示的意义。最后结合这些信息进行分类，模型就能很好地识别垃圾信息了。

GAS 会使用两个图来引入不同的上下文的信息。闲鱼 Graph 是一个异构图，它引入局部上下文信息，另一个是同构图 Comment Graph，它引入了全局上下文信息。在这两个

图上, 研究者分别运行不同的图卷积算法, 并最终融合两个图模型的上下文信息, 从而共同判断一个评论是不是有问题. 这项研究比较重要的地方在于, 研究者基于他们对业务的理解, 所设计的图网络结构能够完成两种上下文信息的抽取, 从而真正提升业务场景的效果. 研究者说: 「这是我们论文最主要的贡献之一, 我们会把传统的文本分类的问题抽象成异构图上的边分类问题, 把图卷积算法和文本分类做一个很好的结合. 不光是在做垃圾检测的过程当中, 阿里在研究与业务中都会遇到很多特定问题. 很多情况下, 我们很难从学术界直接套用一些好的方法, 因此经常要把成熟或新颖的算法匹配到业务上, 这些匹配很可能做出一些新的贡献.

图卷积是非常神奇的一个模型, 它能处理图这种结构化的数据. 要理解图卷积, 需要包括傅立叶变换、拉普拉斯算子等数学基础. 自 ICLR 2017 Kipf 的文章发表以来, 图卷积才逐渐受到更多的关注, 该论文从频域的角度将 CNN 转移到了 Graph, 并推导出了非常简单优雅的形式. 研究者又从空域的角度提出了 GraphSAGE, 它利用直观的节点采样与特征聚合高效地生成节点向量, 后面还有 Bengio 组的 GAT 与 MIT 的 jumping knowledge net. [图卷积开山之作:Semi-Supervised Classification with GCN](#)

除了模型与研究上的创新, 阿里研究者在工程上也做了很多努力. 因为对于闲鱼 Graph 这种超过 10 亿商品与 1 亿用户的节点量, 要做训练和推断都是比较复杂的. 目前该系统已经基于 TensorFlow 分布式框架部署在服务端, 最开始没有成熟的大规模图框架, 基于 TensorFlow 的参数服务器框架, 将图和特征放到参数服务器上, 而后最核心的采样与卷积操作都是从上面获取数据, 整个就是一个分布式系统. 当然后来阿里内部研发了大规模图框架 AliGraph, 研究团队将系统迁移到 AliGraph 后进一步提升了效率.

1.7.8 PCB 产业

PCB 已经发展到全新阶段, 诸如高密度互连 (HDI)PCB, IC 基板 (ICS) 等全新技术引入, 使得整个生产过程从手动变成了全自动化. 随着制造技术的进一步发展, 工艺变得越来越复杂, 缺陷检查越来越重要也越来越难, 这些致命缺陷可能会导致整个 PCB 板的报废. 对于 PCB 制造业来说, 利用人工智能 (AI) 并优化生产工艺以及最终优化整个 PCB 制造流程的机会正在涌现.

PCB 制造通常依赖多年积累知识的专家, 这些专家非常了解和理解制造过程的每个步骤, 他们了解如何利用他们的知识来优化生产和提高产量. 人为的限制 (包括误操作和疲劳) 阻碍了效率增长, 操作员的错误或对 PCB 缺陷的错误识别 (“错误警报”) 可能会由于过度处理而影响良率, 甚至会损害 PCB 本身. 通过将 AI 集成到制造过程中 (图1-43), 机器可以通过接管某些 “学习的” 任务来增加价值, 而人类专家则继续承担更复杂的任务, 这些任务需要在优化和 “培训” 的同时进行思考和互动人工智能系统. 人

与人工智能的结合提高了整体效率和运营, 是 AI 系统的最大机会.

PCB 发展的最终趋势是拥有完全集成 Industry 4.0 系统的工厂, 该系统在全球和制造系统级别采用 AI 技术. “全局”级别包括工厂中的所有系统, 而不仅仅是单个制造系统. 工业 4.0 提供了自动化和数据交换基础结构, 可实现实时生产分析, 双向通信和数据共享, 可追溯性以及按需数据分析. 在任何特定的工厂内, AI 都可以使用从各种制造系统和机器获取的数据来改进流程, 这些数据是通过工业 4.0 机制(例如可追溯性, 双向通信)收集的. 工厂之所以受益, 是因为 AI 分析了大量的系统范围数据以优化工厂设置参数并实现最高水平的生产率和良率. 人工智能分析和自我学习正在进行中, 并通过人工神经网络进行. 几年之内, 它将消除人工操作人员的干预, 并导致建立全自动工厂.



图 1-43 AI 可以帮助 PCB 工厂提高质量

2020 年 3 月, Cerebras 引用的统计数据是相当惊人的。该公司称, 一个 10 机架的 TPU2 集群 (谷歌人工智能计算机产品的第二代) 消耗的功率是单台 WSE 计算机的 5 倍, 占用的空间是它的 30 倍, 性能却仅为 1/3。单一巨型芯片是否真的是人工智能界一直在等待的答案, 今年将变得明朗起来。“(神经网络) 模型越来越复杂,” 加州山景城林利集团 (Linley Group) 的高级分析师迈克 戴姆勒 (Mike Demler) 说, “能够快速训练或再训练非常重要。”芝加哥附近的超级计算机巨头阿贡国家实验室等客户已经在自己的办公场所安装了这种机器, 如果 Cerebras 的推测是正确的, 那么有出色表现的神经网络的数量将呈爆炸式增长。费尔德曼解释说, Cerebras 的创始人们 (都是服务器公司 Sea Micro 的老员工, Sea Micro 被超微半导体公司 AMD 收购) 在 2015 年初次开会时便希望打造一台完美符合现代人工智能工作量性质的计算机。这些工作量由以下几点定义: 快速移动大量数据, 接近处理内核的内存, 而且这些内核不需要处理其他内核正在处理的数据。

这种新的 PCB 制造模型要求将所有工厂系统完全连接以及 AI 作为监视和决策机制. 当前, 存在专有和技术挑战, 这些问题限制了 PCB 工厂的完全自动化, 但 AI 已尽可能地添加到单个系统中, 例如自动光学检查 (AOI) 解决方案. 将生产设施移向全球 AI 模

型的优势包括,可以更可靠地通知 PCB 缺陷——“真实缺陷”,并具有反馈机制,该反馈环可以识别问题的根源,然后自动修改工厂流程以消除相关问题缺陷.

AI 包括机器学习和深度学习,将使 PCB 工厂朝着完全自动化的目标迈进. 机器学习使用的算法使计算机能够使用数据及其已经经历并从中学习的示例来改进任务的性能,而无需对其进行明确的编程. 就 PCB 制造而言,机器学习可提高产量,改善制造操作和工艺流程并减少人工操作,同时有助于推动对工厂资产,库存和供应链的更有效处理.

深度学习将 AI 提升到一个更加复杂的水平,这在全球工厂系统水平上是有益的. 深度学习的灵感来自人脑神经元,多层人工神经网络进行学习,理解和推断的能力. 在 PCB 工厂中,软件系统可以有效地收集的数据,并利用模式和上下文的复杂表示中学习,然后,学习将形成 PCB 制造中自动过程改进的基础. 机器学习和深度学习的实施为 PCB 制造商提供了超越人类理解的能力;人工智能系统通过在人们不愿探索的地方进行更深入的挖掘来发现新的优化机会. AI 专家系统非常高效,通过使用更多更复杂的参数在全球范围内监控工厂系统,减少了所需的人工专家数量,并提高了效率和最佳实践.

3 月 16 日英特尔与康奈尔大学利用英特尔神经拟态芯片 Loihi 来识别爆炸物等危险化学品气味方面的研究,取得重大的突破。其二是 3 月 19 日英特尔用 768 个 Loihi 芯片构造了一个新的神经拟态研究系统 Pohoiki Springs,这个系统达到了 1 亿个神经元的规模,而上一次构建的系统 Pohoiki Beach 是 64 个芯片 800 万个神经元,这次扩大了 12 倍。

1.7.9 百度飞桨: 开源开放的产业级深度学习平台

2020 年 3 月 20 日,清华大学微电子所、未来芯片技术高精尖创新中心钱鹤、吴华强团队与合作者在顶尖学术期刊、英国《自然》杂志 (Nature) 在线发表论文,报道了基于忆阻器阵列芯片卷积网络的完整硬件实现。该存算一体系统在处理卷积神经网络 (CNN) 时能效比前沿的图形处理器芯片 (GPU) 高两个数量级,可以说在一定程度上突破了“冯诺依曼瓶颈”的限制:大幅提升算力的同时,实现了更小的功耗和更低的硬件成本。



图 1-44 基于忆阻器芯片的存算一体系统来源：清华大学

1.7.10 百度飞桨：开源开放的产业级深度学习平台

深度学习框架上承应用下接芯片，是智能时代的操作系统。在关键技术上，中国必须通过自主研发掌握主动权。[百度飞桨](#)是中国首个也是目前国内唯一开源开放、功能完备的产业级深度学习平台，在今年也取得了长足的进步，如今已经具备开发便捷的产业级深度学习框架、超大规模深度学习模型训练技术、多端多平台部署的高性能推理引擎、开源开放覆盖多领域的产业级模型库四大领先技术。通过有效降低深度学习技术应用门槛，飞桨让开发者和企业避免重复“造轮子”，更快速、便捷地开发 AI 应用。目前，飞桨已经累计服务 150 多万开发者，定制化训练平台上企业用户超过 6.5 万，发布了 16.9 万个模型，已广泛应用于工业、农业、服务业等各个行业，实现“遍地开花”。据 IDC 报告显示，在国内深度学习平台市场中，百度与谷歌、Facebook 三强鼎立，已经占据了国内超过七成的市场份额，其中百度的市场份额在过去半年里仍在迅猛增长，展现出领军者的不俗实力。



图 1-45 开源开放的产业级深度学习平台——百度飞桨

CPU 版本安装: pip install -i https://mirror.baidu.com/pypi/simple paddlepaddle, Pycharm 下的百度飞桨训练 uci_housing 数据集的示例见图 1-46。



图 1-46 Pycharm 下的百度飞桨示例 (uci_housing)

1.7.11 计算机视觉的语义分割

滴滴地图事业部和加州大学伯克利分校的研究员提出一种新的多源领域自适应模型, 对多个不同源域的有标注合成数据和目标域的无标注真实数据进行联合学习, 显著提高了图像语义分割的性能。据悉, 这是多源领域自适应第一次应用在语义分割任务上。相关研究以「Multi-source Domain Adaptation for Semantic Segmentation」(基于多源领域自适应的语义分割) 为题发表在神经计算和机器学习领域的顶级会议—第 33 届神经信息处理系统大会 (NeurIPS 2019) 上。论文 [Multi-source Domain Adaptation for Semantic Segmentation, 论文代码](#)

卷积神经网络 (convolutional neural networks, CNNs), 人们提出了多种端到端的语义分割方法。虽然这些方法取得了良好的效果, 但也存在一定的局限性。一方面, 训练这些方法需要使用像素级标注的大规模数据, 这是非常昂贵和耗时的。例如, 在 Cityscapes 数据集中标注每幅图像大约需要 90 分钟。另一方面, 由于存在领域偏移 (domain shift) 或数据集偏差 (dataset bias), 他们不能很好地将所学知识迁移到新的域或数据集。为了避免数据收集和标注的成本, 图形学和仿真软件的发展使得研究者们可以使用 CARLA 和 GTA-V 等模拟器所生产的无限量合成标注数据。为了减少不同领域之间的差距, 研究者们提出了领域自适应 (domain adaptation, DA) 或知识迁移技术 (knowledge transfer techniques), 并进行了理论分析和算法设计。语义分割的领域自适应算法在自动驾驶等领域具有重要的作用。现有的工作主要关注于单个源域的场景, 很难处理实际中具有不同分布的多个源域的情况。在这篇论文中, 研究者们研究了基于多源领域自适应的语义分

割。



图 1-47 语义分割

现有自适应方法在图像分割上的挑战

除了传统的有标注源域上的任务损失外, 深度无监督领域自适应 (UDA) 方法通常还会训练其他损失函数来处理领域偏移, 如差异损失 (discrepancy loss)[2]、对抗损失¹(adversarial loss)[ZhuPark20178237506]、重构损失 (reconstruction loss)[GoodfellowGAN2014]等。目前语义分割任务上从合成数据到真实场景的领域自适应方法都集中在单数据源设置上, 没有考虑从多个不同分布的数据源收集数据这一更实际的场景。简单地将不同的源组合成一个源并直接使用单一源 DA 不会有很好的效果, 因为来自不同源域的图像在学习过程中可能会相互干扰。早期对多源 DA(multi-source DA, MDA) 的研究使用了浅层模型。近年来, 人们提出了一些多源深度 UDA 方法, 这些方法主要针对图像分类 [Ghifary2015]。由于以下原因, 直接将这些 MDA 方法从分类扩展到分割可能不会有很好的效果。(1) 分割是一个结构化的预测任务, 其决策函数比分类更复杂, 因为它必须在指数大的标签空间中解析预测 [Zhang2017]。(2) 目前的 MDA 方法主要关注特征级对齐, 只对高层次的信息进行对齐。这对于粗粒度的分类任务来说可能足够了, 但是对于细粒度的语义分割来说显然是不够的, 因为分割是像素级的预测。(3) 这些 MDA 方法只将每个源域和目标域对齐。虽然不同的源域被匹配到目标域, 但在不同的源域之间可能存在显著的不一致。

多源对抗域聚合网络算法基于对抗生成式网络 (GAN)[GoodfellowGAN2014] 和循环对抗生成式网络 (CycleGAN)[ZhuPark20178237506], 提出了一种新的端到端的多源对抗域聚合网络 (Multi-source Adversarial Domain Aggregation Network, MADAN), 框架如图1-48所示。MADAN 主要包括三个模块: (1) 动态对抗图像生成模块 (Dynamic Adversarial

¹BigGAN: Large Scale GAN Training for High Fidelity Natural Image Synthesis

Image Generation), (2) 对抗域聚合模块 (Adversarial Domain Aggregation), (3) 分割特征语义对齐模块 (Feature Aligned Semantic Segmentation).



图 1-48 MADAN 框架图

首先, 对于每个源域, 文章使用[循环对抗生成式网络\(CycleGAN\)\[Long2015CVPR\]](#)生成一个动态保持语义并且具有像素级一致性的自适应域; 其次, 文章提出了子域聚合判别器 (Sub-domain Aggregation Discriminator) 和跨域循环判别器 (Cross-domain Cycle Discriminator), 以使不同的自适应域更紧密地聚合; 最后, 在训练分割网络的同时, 对聚合域和目标域进行特征层面的对齐. 通过 MADAN, 不同的适应域可以更好地聚合为一个更统一的域. 基于聚合域对分割模型进行训练, 能够更好地提升分割模型在目标域上的表现.

智能决策 将人工智能纳入工作流程. 对于只依赖结构化数据的常规决策, 人工智能不受认知偏见的影响. 人工智能可以被训练为在人群中找到特殊——因为人类可能会因感情而产生偏见, 而人工智能却不会. 人工智能更适合处理非线性关系, 不管是指数关系、幂律关系、几何级数关系、二项式分布关系还是其他关系. 这些关系计算量庞大, 远远超出我们人类可以处理的范围.

人工智能工作流可以更好地利用数据中包含的信息, 并且在其决策中体现一致性并保持和观. 它可以更好地确定哪种广告创意最有效, 如何设定最佳的库存水平, 或者进行哪些金融投资.

1.7.12 威胁检测和预防

网络安全威是当今任何组织面临的最大威胁, 系统、数据、云技术、应用程序、设备和分布式终端的激剧增加, 使得互联网面临越来越多的安全威胁, 安全网络也是 5G 和

大数据技术的一个应用前提。这意味着组织必须比以往任何时候都更加努力地工作，以保护他们的资产和客户。其实这已经超出了自动化对策的范围，它现在要求信息安全专业人员积极主动地检测，以先发制人地避免或阻止威胁。

如今的安全软件使用机器学习、深度学习、机器推理和一系列相关技术来审查大量数据，其目的是加速对正常与异常的理解，以检测恶意行为和实体。到 2022 年，全球信息安全支出预计将达到 1700 亿美元，网络安全行业正在关注创新和效率，更具弹性的机制和工具。由于技术的进步，信息安全中有四个主要的 AI 和机器学习场景，下面我就来为大家一一讲解。

► 网络威胁分析

如今的企业将越来越多的业务数字化，他们会更新旧数据并开发内部（通常是混合）网络。这些庞大的网络拓扑不仅复杂，而且还需要大量的网络安全资源来管理所有通信、事务、连接、应用程序和策略。如果企业规模很大，则数字化不仅意味着巨大的投资，而且还有数据被盗的风险。人工智能在网络安全方面的应用，完全可以应对这一挑战。值得注意的是，网络安全中的人工智能会监控所有传入和传出的网络流量，以挖掘可疑活动并对威胁类型进行分类。

► 恶意软件检测

恶意软件是有攻击性的不断发展的代码或软件类别的总称，尽管恶意软件检测已经存在多年了，通常将可疑代码与基于签名的系统相匹配，但机器学习现在正转向推理技术。网络安全领域的人工智能在分析大量数据、事件类型、来源和结果时，会在恶意文件被打开之前检测到恶意软件的存在。它还可以识别恶意软件的类型，因为恶意软件会随着其他技术的进步而不断发展，比如恶意木马、僵尸网络、恶意广告、勒索软件等实例。迄今为止，深度学习和人工智能已经帮助各种安全应用程序从恶意软件和良性应用程序中获得的数以千万计的样本，这样，后期的检索就可以专门设置一个算法，进行高效的检索，这些都离不开精确标记的数据库。

► 扩大安全分析的范围

网络安全领域的人工智能最擅长发现潜在的威胁媒介，不过是否算是真正的威胁，还得依靠人类，这意味着人类仍然是控制、解释和判断威胁的终结者。只能说，机器学习让人类变得更加强大了，这主要体现在两个方面：

- (1) 人工智能使重复的任务自动化，例如，它会对低风险警报或繁琐的数据任务进行分类，以便让分析师有时间进行更高价值或战略决策。
- (2) 机器学习负责低层次的威胁情报的数据整理和分析，这样人类分析师就可以从基本的数据收集工作中解脱出来，分析更有价值的信息，以进行更高价值的战

略决策。

实际的测试表明，理想的网络安全性能或准确性往往是人类和人工智能的结合成果，而不是单独进行判断。安全工具的增强在未来几年对安全团队来说可能是必不可少的。事实上，市场上的一些技术已经支持 UI 工具，使网络专家能够合并威胁类型来重新训练机器学习模型，并根据问题配置特定的修复程序。

► 基于人工智能的反向攻击

任何技术的正反两面的发展趋势都是同步的，企业必须训练机器学习算法，以识别其他基于机器学习算法发起的攻击。

例 1.53 有研究人员就发现黑客使用机器学习来识别企业网络中的薄弱环节。他们使用此信息通过网络钓鱼、间谍软件或分布式拒绝服务攻击将目标定为攻击入口点。甚至已经有攻击者开发出了智能恶意软件，它们还有另一个更可怕的称呼——人工黑客 (artificial hackers)，以针对受害者的具体情况进行个性化攻击。基于人工智能的攻击展示了人工智能的优势（快速可扩展性、行为分析和个性化）不仅安全人员看得到，攻击者也看得到。上述四个应用场景，只是人工智能在网络安全领域众多应用中的一小部分。不过要注意的是，机器学习并不是万灵丹，它只是人类的一个辅助工具。传统的基于签名的检测方法的缺点，机器学习中也存在。同时，它可以降低成本。人工智能的价值在于做出比人类所能做得更好的决定，这也就提高了效率，为企业新的形态作出演变。

智能网络

研究智能网络的意义

- (1) 因特网在为人类提供了方便快捷的信息交换手段，但基于因特网的万维网 (WWW) 却是一个杂乱无章、真假不分的信息海洋，它不区分问题领域，不考虑用户类型，不关心个人兴趣，不过滤信息内容。
- (2) 传统的搜索引擎在给人们提供方便的同时，大量的信息冗余也给人们带来了不少烦恼。因此，利用人工智能技术实现智能网络具有极大的理论意义和实际价值。

智能网络的两个重研究内容

- (1) 智能搜索引擎是一种能够为用户提供相关度排序、角色登记、兴趣识别、内容的语义理解、智能化信息过滤和推送等人性化服务的搜索引擎。

(2) 智能网格是一种与物理结构和物理分布无关的网络环境，它能够实现各种资源的充分共享，能够为不同用户提供个性化的网服务。可以形象地把智能网格比作一个超级大脑，其中的各种计算资源、存储资源、通信资源、软件资源、信息资源、知识资源等，都像大脑的神经元细胞一样能够相互作用、传导和传递，实现资源的共享、融合和新生。

微盟删库危机事件 2020 年 3 月 1 日，微盟（SEHK：02013）官方发布公开信，称在腾讯云团队协助下，被删数据已全面找回，同时公司将准备 1.5 亿元人民币赔付拨备金进行赔偿。对于一家日均流水从三千元做到四万元的社区电商平台商户，自备的 ERP 系统存有库存管理与订单管理数据，波及到的仅有依靠微盟平台的会员系统与积分系统，更严重的影响是用户信息的丢失。该商户甚至因没有用户信息，“钱都在我手里，但因为客户数据的丢失，我都不知道该给谁发多少钱。”目前导致商铺线上全部崩溃，无法经营，对经营产生重大影响。未来将会面临多方困难，包括修复时间不可控，运营成本上升，客户及商家将考虑转移到其他平台，面临更多或巨额赔偿，流失潜在客户给竞争对手。报告预期上述困难将给微盟带来长期负面影响。

1.7.13 人工智能案例

例 1.54 飞桨做的无人驾驶智能车：吴东昱，北京钢铁侠科技深度学习算法工程师，主要研究深度学习、无人驾驶等。

例 1.55 使用神经网络计算微分方程 (DeepTech 深科技 2019-12-18)

$$y' = \frac{16x^3 - 42x^2 + 2x}{(-16x^8 + 112x^7 - 204x^6 + 28x^5 - x^4 + 1)^{1/2}} \quad (1.11)$$

正确答案是：

$$y = \sin^{-1} (4x^4 - 14x^3 + x^2) \quad (1.12)$$

这个方程非常难解，即使是 MATLAB 和 Mathematica 这样强大的数学运算软件在 30 秒之内也解不出来。

Mathematica Alpha



图 1-49 WolframAlpha AI 计算结果

Maple 2019.2

```

ode := diff(y(x), x) = (16*x^3 - 42*x^2 + 2*x)/(-16*x^8 + 112*x^7 - 204*x^6 + 28*x^5 - x^4 + 1)^(1/2);
d
ode := --- y(x) =
dx


$$\frac{16x^3 - 42x^2 + 2x}{\sqrt{-16x^8 + 112x^7 - 204x^6 + 28x^5 - x^4 + 1}}^{(1/2)}$$

dsolve(ode);
y(x) = | |

```

$$\begin{aligned}
& \quad | \\
& \quad | / \\
& \quad \backslash \\
& \quad | \\
& \quad | / \\
& \quad \backslash \\
& \quad / \quad \{3\} \quad \{2\} \quad \backslash \\
& \quad 2 \ \backslash 8 \ x \quad - \ 21 \ x \quad + \ x / \\
& \quad \hline \\
& \quad (1/2) \quad | \\
& \quad / \quad \{8\} \quad \{7\} \quad \{6\} \quad \{5\} \quad \{4\} \quad \backslash \\
& \quad \backslash 16 \ x \quad + \ 112 \ x \quad - \ 204 \ x \quad + \ 28 \ x \quad - \ x \quad + \ 1 / \\
& \quad \hline \\
& \quad + \ _C1
\end{aligned}$$

[Deep Learning For Symbolic Mathematics](#), Fabio Petroni、Tim Rocktaschel、Patrick Lewis,
NIPS 2019

神经网络在解决统计或拟合问题时较计算和解决符号数据更为优秀。研究者表明，神经网络在解决一些复杂的数学问题上表现很好，例如符号积分和解决微分方程。

研究者提出了一种语法，可以表示这些数学问题，以及一种用于生成大数据集的方法，用于训练一个 seq2seq 模型。研究者提出的方法在表现上超过了商业代数计算软件的性能，如 Matlab 或 Mathematica。

神经网络强大的拟合能力使其在机器学习中占有一席之地。本文创新性地使用神经网络拟合数学问题，且计算速度很快。

不过，Facebook AI 的研究人员 Guillaume Lample 和 François Charton 开发了一套新算法，可以在数秒内求解类似的方程。他们训练了一套神经网络来执行必要的符号推理和数学运算，首次实现了数学表达式的微分和积分。这项成果是迈向更强大的数学推理工具的重要一步，也是利用神经网络的新方法。在发现和识别规律上，神经网络可以有很好的表现，因此我们用它来执行面部识别、物体识别和自然语言识别等任务。但是尽管付出了很大努力，仍然没人训练出一套可以高效完成数学类象征性推理任务的神经网络。在这方面，神经网络甚至比不上小学生的水平，最好成绩是学会整数的加法和乘法。和人类相似，对于神经网络来说，分析高级数学表达式的难点之一在于解析式子中的简写模式。举个例子， x^3 代表了 x 乘以 x 乘以 x ，其中的“乘法”代表了重复的加法，而“加法”本身又代表了两个变量之和。因此，一条看似简单的数学表达式，可能是由一系列更简单的数学运算高度精简之后得出的。想让神经网络“掌握”这种逻辑非常困难。实际上，人

类也有类似的问题,但解决办法通常是从小就开始灌输,形成习惯。从本质上来说,微分和积分的运算过程会涉及到规律识别。这给了神经网络发挥的机会,只要解决了简化的难题。Lample 和 Charton 提出了一种更好的解决方法,提升了神经网络的求解水平。他们先将数学表达式拆分为多个基本单位,然后训练神经网络如何识别积分和微分中包含的数学运算模式,最后再用全新的表达式测试网络的表现,与传统运算工具 Mathematica 和 MATLAB 的表现进行对比。研究人员采用了树状结构拆分表达式。树叶代表数字、常量和变量,比如 2 和 x ,节点则代表运算符号,比如 $+$ 和 $-$ 。例如,表达式 $2 + 3x(5 + 2)$ 可以分解成图1-50的形式。



图 1-50 表达式 $2 + 3x(5 + 2)$ 的分解

更复杂的表达式 $3x^2 + \cos(2x) - 1$ 可以写成图1-51的形式:

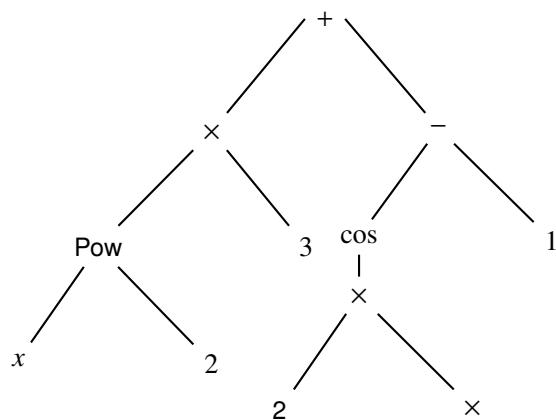


图 1-51 表达式 $3x^2 + \cos(2x) - 1$ 的分解

如果不同树的数学运算结果相同,那么它们就被视为是相同的,比如 $2 + 3$ 和 1×5

是相同的。这样一来，表达式的简化就等同于找到数的较短等价表示方法，很多数学运算就更好处理了。这些树也可以写成多个节点连续组成的序列，而现有的一种名为 Seq2Seq 的模型正好可以很好地处理序列形式的信息。实际上，Seq2Seq 通常用于机器翻译，将一长串的单词从一种语言翻译成另一种语言。“我们的方法本质上将数学视为一种自然语言，”两名研究人员表示。

$$\frac{\partial^2 \psi}{\partial x^2} - \frac{1}{v^2} \frac{\partial^2 \psi}{\partial t^2} \quad (1.13)$$

图 1-52 是拆分更复杂的表达式(1.13)的树状图。

$$\frac{\partial^2 \psi}{\partial x^2} - \frac{1}{v^2} \frac{\partial^2 \psi}{\partial t^2}:$$



图 1-52 表达式 $\frac{\partial^2 \psi}{\partial x^2} - \frac{1}{v^2} \frac{\partial^2 \psi}{\partial t^2}$ 的分解

下一步是训练过程，这需要大量的数据支持。Lample 和 Charton 通过从二进制运算符库中随机组合数学表达式来创建新的数据库，其中包含加减乘除、三角函数、变量和常数等等。他们还限制了树的节点数量，防止方程变得过于庞大。针对随机生成的表达式，他们使用计算工具对其进行积分和微分。任何不能被积分的表达式都会被丢弃。通过这样的操作，研究人员生成了超大规模的训练数据集，包含 8 千万个一阶和二阶微分方程，2 千万个分部积分表达式。神经网络使用这些数据进行训练，学习如何对给定数学表达式求导或积分。为了测试神经网络的表现，研究人员使用了它从没见过 5000 个表达式。它的计算结果还会与现有主流数学计算工具 Mathematica, MATLAB 和 Maple 进行对比，限时 30 秒得出计算结果。这些计算工具使用的算法是美国数学家 Robert Risch 在 1960 年代提出的简化版本。

测试结果显示 1-1，神经网络模型的准确率明显优于 Mathematica，在积分任务上的准确率高达 99.6%，而 MATLAB 和 Maple 甚至还不如 Mathematica，跟神经网络更没有可比性。在许多情况下，传统计算工具在尝试 30 秒之后根本无法得出结果，而神经网络只

表 1-1 神经网络与三种主流计算工具的表现对比.

	Integration(BWD)	ODE (order 1)	ODE(order 2)
Mathematica(30s)	84.0	77.2	61.6
Matlab	65.2	-	-
Maple	67.4	-	-
Beam size 1	98.4	81.2	40.8
Beam size 10	99.6	94.0	73.2
Beam size 50	99.6	97.0	81.0

需要 1 秒. 文章开头的表达式就是其中之一. 另一个有趣的发现是, 神经网络通常会找到相同问题的不同解, 因为表达式有很多种不同的写法. 这让研究员感到惊讶. “该模型

表 1-2 传统计算工具 30 秒之内算不出结果的表达式.

方程	解
$y' = \frac{16x^3 - 42x^2 + 2x}{(-16x^8 + 112x^7 - 204x^6 + 28x^5 - x^4 + 1)^{1/2}}$	$y = \sin^{-1}(4x^4 - 14x^3 + x^2)$
$3xy \cos(x) - \sqrt{9x^2 \sin(x)^2 + 1}y' + 3y \sin(x) = 0$	$y = c \exp(\sinh^{-1}(3x \sin(x)))$
$4x^4yy'' - 8x^4y^2 - 8x^3yy' - 3x^3y'' - 8x^2y^2 - 6x^2y' - 3x^2y'' - 9xy' - 3y = 0$	$y = \frac{c_1 + 3x + 3 \log(x)}{x(c_2 + 4x)}$

在未经针对性训练的情况下就能找到等效表达式, 非常吸引人,” Lample 和 Charton 表示, “据我们所知, 目前还没有针对神经网络检测数学表达式模式能力的研究. 这是一个重大突破.” 研究人员没有透露 Facebook AI 会如何利用这套算法. 不过他们认为, 在日趋复杂的计算数学世界中, 更加强大的运算能力是必不可少的, 在标准数学运算框架下引入神经网络必会成为未来的趋势.

例 1.56 AI 发现“日心说”——隐形保卫诺言 2019-12-10

研究人员给神经网络输入了在地球上观测的火星和太阳运动的数据. 由于运算量并不大, 大约只花了一天的时间, 人工智能就准确得出了和哥白尼一样的结论.



图 1–53 图源: Wikimedia Commons

雷纳托·雷内尔 (图源: ETH Globe / Daniel Winkler, 图1–54) 作为理论物理学家的雷纳尔还拥有计算机科学的博士头衔, 他很快集结了一支由优秀的研究生、博士生和资深科学家组成的五人团队。这个团队运用时下最流行的神经网络算法和机器学习技术, 开始设计自己的人工智能系统, 也就是后来的 SciNet.



图 1–54 图源: Wikimedia Commons

他们研发的神经网络结构, 正是模仿了经典物理学家的思考过程: 化繁为简。科学家们观测自然, 展开实验, 收集各类数据, 然后找出其中的逻辑关系, 最后用最简单的数学

方程式展现。而 SciNet 的神经网络可分为两个部分，第一部分是输入各种实验和观测数据，然后如同榨汁机一般，不断提取、精炼、简化再简化成相关的参数，这和传统的神经网络一样。第二部分的任务则是从精简的信息中总结出最简数学公式，雷纳尔将其类比为写满物理公式的教科书。人工智能·哥白尼的“日心说”



图 1-55 (图源: pexels.com)

设计好神经网络结构之后，研究团队测试了四个案例：阻尼摆，角动量守恒，量子比特表征以及太阳系的日心说模型。在测试阶段，他们使用已知的物理案例验证算法的正确性。作为原理性证明的研究，团队并未使用海量的数据集，只证明自己的神经网络可以复现简单案例。

 **注 1.57. “日心说” VS “地心说”** 在这几个案例中，“日心说”的历史最为人熟知且津津乐道。16 世纪的哥白尼测量了相对于遥远的恒星，地球、太阳和火星三者之间的相互角度关系，从而提出了“日心说”的大胆假设。相比复杂不堪的“地心说”，新模型更加简洁，能够解释并预测行星运动，从而开启了现代天文学研究的新篇章。仿照这一段历史故事，研究人员给神经网络输入了在地球上观测的火星和太阳运动的数据。由于运算量并不大，雷内尔记得大约只花了一天的时间，人工智能就准确得出了和哥白尼一样的结论。SciNet 初长成能够给出经典物理学理论的 SciNet。经过两年研发，这套系统目前还只是运行在台式电脑上。日后如果进行更加复杂、数据量更加庞大的运算，就需要更强劲的计算能力、规模和更加复杂的神经网络结构。除此之外，雷内尔还指出，现有的神经网络结构有其功能局限性，如果只是用该网络来进行预测，它的错误率极低，但它无法给出解释。换言之，你可以提问，输入数据，它会给出答案，但是你却无法知道如何得出这样的答案，为什么会是这样的答案。

例 1.58 (平安保险的 AI 智能) 平安在智能化应用的案例实践出发, 指出可以分三步走:

第一阶段是婴儿阶段, 即形成包括听觉、视觉、阅读理解能力在内的基础认知;
第二阶段是学习阶段, 即构建海量信息和知识图谱的全面知识体系;
第三阶段是专家阶段, 即能够具备打造专业解决方案的能力, 能够让 AI 赋能金融服务、医疗、智慧城市等行业应用场景.

平安对于 AI 伦理问题的极大关注, 不仅积极参与各大部委对于 AI 伦理的标准制定, 还专门成立了平安人工智能伦理委员会, 创建了一套完整的体系来保证 AI 不会被滥用.

例 1.59 (Augmented Reality 眼镜 [增强现实眼镜]) Facebook 首席人工智能科学家杨立昆 (Yann LeCun, 被认为是“卷积网络之父”, 2018 图灵奖获得者, 目前担任 Facebook 首席人工智能科学家和纽约大学教授. 杨立昆于 1960 年出生于法国巴黎附近, 于 1983 年获得法国高等电子与电工技术工程师学校的学士学位, 以及 Pierre et Marie Curie 大学的计算机科学博士学位. 1987 年至 1988 年, 杨立昆是多伦多大学辛顿实验室的博士后研究员.) 看来, 如果机器学习研究员未来想要开发出一款杀手级应用, 更有效的批处理和自我监督学习技术可以帮助人工智能像人类和动物一样学习更多, 也更有助于提高人工智能的能效. 可以试试去挑战增强现实 (AR Augmented Reality) 眼镜.

一款完美的 AR 眼镜需要用到对话式人工智能、计算机视觉和其他复杂系统的组合. 这些系统又必须在小巧的眼镜内部进行操作. 这样一来, 为了确保电池寿命, 必须使用低功耗的人工智能系统, 以便用户可以长时间佩戴和使用眼镜. “对于硬件而言, 这是一个巨大的挑战, 因为你需要在实时或延迟下, 跟踪你视觉的摄像头, 在移动过程中, 这需要大量的计算. 同时, 你希望能够通过语音与智能助理互动, 以确保助手一直能听到你的声音, 并且也会与你说话. 你还需要手势识别, 以便智能助理可以实现实时手部追踪.” 杨立昆说. 就目前人工智能技术的发展来看, 杨立昆认为实时手部跟踪技术已经成熟, 但是“我们只是不知道如何以小巧的形式来做到这一点, 让其功耗与 AR 眼镜兼容.” 杨立昆说, “就功耗、性能和外形尺寸的变化而言, 它确实超出了我们今天的能力范围, 因此必须使用人们从未想到过的技巧.” 神经网络就是一种选择.

苹果、Niantic 和高通等公司一样, Facebook 在今年秋天公布了到 2025 年制造 AR 眼镜的计划.

例 1.60 (AI 智能的伦理风险) 假设我们不知道 Apple Card 信用额度的参数是什么, 但是如果考虑因素包括年收入, 但不考虑共同财产所有权和税务申报, 那么美国仍然以每美元 1 美元的价格赚取 80.70 美元的女性将处于固有的劣势. 人们仍然可以设置算法在分析数据集时应考虑的参数. 而且从事这项工作的开发人员和数据科学家可能不会意识到

他们放置的参数所包含的无意识偏差。类似的假设还有，如果图像集合显示了厨房中许多女性的例子，则使用该图像数据集训练的算法将在女性和厨房之间形成关联，并可以在其假设和决策中再现这种关联。

迄今为止，法律建立平等保护所做的努力旨在防止有意识的人类偏见。AI 的问题在于，它比以往任何时候都可以更快、更有效地重现我们的无意识偏见，并且这样做没有道德良心或对 PR 的关注。在这种情况下，该算法无需高阶思维技能就能做出信贷决策，这会使人在为女性和男性提供的信贷限额之间的明显差异中看到一个危险信号。

例 1.61 (智慧法院 + 法信) AI 已经在中国司法领域很多场景广泛应用了，比如一个名为「法信」的平台已经覆盖了全国 30 个省，3200 多家法院。也就是说，全国 90% 的法官都在使用这个平台查询法律知识，分析案例数据，精准解决知识检索需求。在「智慧法院」的应用场景中，AI 技术还可以辅助法官判案，不仅能够帮助法官「减负」，提高工作效率，辅助司法管理和决策，还可以实现「数据多跑路，群众少跑腿」，使人民群众感受到司法的公平正义。法信，这套中国最大的法律知识和案例大数据平台，是由人民法院出版社、中国司法大数据研究院和北京国双科技有限公司（以下简称「国双」）共同研发，可以高效、精准、便捷地解决法律人的司法信息检索与分析需求和海量知识数据供给的匹配问题。对于法律人来说，法信提供的信息非常丰富：它可以呈现在某个时间段不同地域和法院处理了什么类型的案件，具体的案件特征是什么，引用的法条是哪些，争议焦点是什么；可以直接查找具体的某位法官、律师或当事人涉及的所有案件，如果当事人是企业的还可以直接查阅企业工商信息和涉诉信息。

如果你不是一个专业人士，在法信平台上，也可以使用最自然的和人交流沟通的方式自由提问。比如：「我买了一座房子，对面又盖了一座挡了我的阳光怎么办？」这样的问题，都可以得到有用的答案，以及答案的权威来源和法律依据。这个答案的专业性，得益于国双的知识图谱技术。国双通过多年的司法数据和知识梳理积累，让司法行业知识体系以大规模知识图谱的形式被存储起来，并得到有效利用。在法信平台智能应用的背后，是深度学习、知识图谱等技术的运用，法律专家团队和人民法院出版社的法律内容资源的支撑。法信平台拥有以大规模知识图谱所存储的经过权威司法专家整理和积累的司法知识体系，把自然语言处理、意图分析、实体与关系识别、机器学习等人工智能技术融合在一起，在传统法律数据库关键词查找、知识检索和类案维度检索的方法之外，实现了交互式专业问答等功能，大大提高了中文法律知识服务的水准。在数据方面，法信平台拥有国内司法领域最权威、最完备的知识体系和数据资源，包括中国所有的法律法规、典型案例、图书期刊、法律文书等十二个法律专业数据库资源。「在智慧法院平台中，AI 系统需要做很多文本处理的工作。算法需要阅读起诉状、答辩状、庭审笔录、判

决文书，」国双首席技术官刘激扬介绍道。「在这个过程中我们需要强大的 AI 技术，也需要相关领域的专业知识。」

例 1.62 AWS 宣布了 Sage Maker Studio 机器学习集成开发环境，还推出了 Inferentia 芯片。去年首次宣布的该芯片，能够加速机器学习的推理计算。在 Inferentia 芯片的加持下，研究者可较之前预先训练过的模型带来更明显的提速、且更具成本效益。AWS 首席执行官 Andy Jassy 指出：许多企业都在模型训练的定制芯片上投入了大量精力，尽管常规 CPU 上已经能够较好地执行推理运算，但定制芯片的效率明显更高。与 EC4 上的常规 G4 实例相比，Inferentia 能够让 AWS 带来更低的延时、三倍的吞吐量、且降低 40% 单次的成本。新的 Inf1 实例，可实现高达 2000TOPS 的特性、与 TensorFlow、PyTorch 和 MXNet 集成、且支持可在框架之间迁移的 ONNX 模型格式。目前其仅可在 EC2 计算服务中使用，但 AWS 将很快为其引入对 SageMaker 机器学习和其它容器服务的支持。

例 1.63 (AI 芯片求索 (QuestCore) 依图重点展示了基于自研 AI 芯片求索 (QuestCore) 的新产品线和智慧城市解决方案。为了解决行业算力瓶颈难题，依图自研并于今年 5 月推出了“发布即商用”的全球首款深度学习云端芯片求索，结合依图世界级算法和先进芯片设计理念，求索的 AI 计算能效比是最先进 GPU 方案的 5-10 倍。求索芯片强大的 AI 算力和超高性价比，让城市级大规模智能视频全解析方案落地成为可能。在安博会现场，完全基于依图求索芯片的依芯求索 AI 服务器吸引了众多参会者驻足查看。这台服务器具有超强的 AI 视觉算力，不依赖传统 X86 CPU 和 GPU，技术自主安全可控，提供 10 倍传统服务器的算力。

1.8 人工智能的学习平台

1.8.1 Tensorflow

TensorFlow 是一个采用数据流图 (data flow graphs)，用于数值计算的开源软件库。节点 (Nodes) 在图中表示数学操作，图中的线 (edges) 则表示在节点间相互联系的多维数据数组，即张量 (tensor)。它灵活的架构让你可以在多种平台上展开计算，例如台式计算机中的一个或多个 CPU(或 GPU)，服务器，移动设备等等。TensorFlow 最初由 Google 大脑小组 (隶属于 Google 机器智能研究机构) 的研究员和工程师们开发，用于机器学习和深度神经网络方面的研究，这个系统的通用性使其也可广泛用于其他计算领域。数据流图用“结点” (nodes) 和“线” (edges) 的有向图来描述数学计算。“节点”一般用来表示施加的数学操作，但也可以表示数据输入 (feed in) 的起点/输出 (push out) 的终点，或者是读取/写入持久变量 (persistent variable) 的终点。“线”表示“节点”之间的输入/输出关系。



图 1-56 Tensorflow 的标识

这些数据“线”可以输运“size 可动态调整”的多维数据数组, 即“张量”(tensor). 张量从图中流过的直观图像是这个工具取名为“Tensorflow”的原因. 一旦输入端的所有张量准备好, 节点将被分配到各种计算设备完成异步并行地执行运算.

TensorFlow 的特征

- ▶ 高度的灵活性
- ▶ TensorFlow 不是一个严格的“神经网络”库. 只要你可以将你的计算表示为一个数据流图, 你就可以使用 TensorFlow. 你来构建图, 描写驱动计算的内部循环. 我们提供了有用的工具来帮助你组装“子图”(常用于神经网络), 当然用户也可以自己在 TensorFlow 基础上写自己的“上层库”. 定义顺手好用的新复合操作和写一个 python 函数一样容易, 而且也不用担心性能损耗. 当然万一你发现找不到想要的底层数据操作, 你也可以自己写一点 c++ 代码来丰富底层的操作. 真正的可移植性 (Portability)
- ▶ TensorFlow 在 CPU 和 GPU 上运行, 比如说可以运行在台式机、服务器、手机移动设备等等. 想要在没有特殊硬件的前提下, 在你的笔记本上跑一下机器学习的新想法? TensorFlow 可以办到这点. 准备将你的训练模型在多个 CPU 上规模化运算, 又不想修改代码? TensorFlow 可以办到这点. 想要将你的训练好的模型作为产品的一部分用到手机 app 里? TensorFlow 可以办到这点. 你改变主意了, 想要将你的模型作为云端服务运行在自己的服务器上, 或者运行在 Docker 容器里? TensorFlow 也能办到. TensorFlow 就是这么拽:)
- ▶ 将科研和产品联系在一起
过去如果要将科研中的机器学习想法用到产品中, 需要大量的代码重写工作. 那

样的日子一去不复返了! 在 Google, 科学家用 Tensorflow 尝试新的算法, 产品团队则用 Tensorflow 来训练和使用计算模型, 并直接提供给在线用户. 使用 Tensorflow 可以让应用型研究者将想法迅速运用到产品中, 也可以让学术性研究者更直接地彼此分享代码, 从而提高科研产出率.

► 自动求微分

基于梯度的机器学习算法会受益于 Tensorflow 自动求微分的能力. 作为 Tensorflow 用户, 你只需要定义预测模型的结构, 将这个结构和目标函数 (objective function) 结合在一起, 并添加数据, Tensorflow 将自动为你计算相关的微分导数. 计算某个变量相对于其他变量的导数仅仅是通过扩展你的图来完成的, 所以你能一直清楚看到究竟在发生什么.

► 多语言支持

Tensorflow 有一个合理的 c++ 使用界面, 也有一个易用的 python 使用界面来构建和执行你的 graphs. 你可以直接写 python/c++ 程序, 也可以用交互式的 ipython 界面来用 Tensorflow 尝试些想法, 它可以帮你将笔记、代码、可视化等有条理地归置好. 当然这仅仅是个起点——我们希望能鼓励你创造自己最喜欢的语言界面, 比如 Go, Java, Lua, Javascript 或者是 R.

► 性能最优化

比如说你有一个 32 个 CPU 内核、4 个 GPU 显卡的工作站, 想要将你工作站的计算潜能全发挥出来? 由于 Tensorflow 给予了线程、队列、异步操作等以最佳的支持, Tensorflow 让你可以将你手边硬件的计算潜能全部发挥出来. 你可以自由地将 Tensorflow 图中的计算元素分配到不同设备上, Tensorflow 可以帮你管理好这些不同副本.

► 谁可以用 TensorFlow?



注 1.64. 学生、研究员、爱好者、极客、工程师、开发者、发明家、创业者等等都可以在 Apache 2.0 开源协议下使用 *Tensorflow*.

图 1-57 tensors flow

TensorFlow 包含许多编译器和优化器，可在多个级别的软硬件堆栈上运行。对于 *TensorFlow* 的日常用户，在使用不同种类的硬件 (*GPU*、*TPU*、移动设备) 时，这种多级别堆栈可能会表现出令人费解的编译器和运行时错误。

TensorFlow 组件的结构：



图 1-58 TensorFlow 的组件

图 1-59 是使用 tensorflow-addons 包, 使用 Linearly Scaled Hyperbolic Tangent (lisht) 激活函数测试的例子.

```

File Edit View Navigate Code Refactor Run Tools VCS Window Help ReformerTest C:/Users/ocnzh/PycharmProjects/ReformerTest - C:/Users/ocnzh/PycharmProjects/Tensorflow-addons/addonsTest.py
C:/Users/ocnzh/PycharmProjects/ReformerTest - C:/Users/ocnzh/PycharmProjects/Tensorflow-addons/addonsTest.py

TensorNN.py addonsTest.py
Project
1 import tensorflow as tf
2 import tensorflow_addons as tfa
3 import numpy as np
4
5 x = tf.constant([-2, 3.2, 4.5, -100.2, 1, -1, 2, 0])
6
7 y=tfa.activations.lisht(x)
8
9 def lisht_derivative(x):
10     y = [0]*len(x)
11     for i in range(len(x)):
12         y[i] = np.tanh(x[i]) + x[i] * (1 - np.tanh(x[i])**2)
13     return y
14
15 x_der = lisht_derivative(y)
16
17 for xj in y:
18     print(((np.exp(4 * xj) - 1 + 4 * xj * np.exp(2 * xj)) / (np.exp(4 * xj) + 1 + 2 * np.exp(2 * xj))))

```

Run: addonsTest

```

2020-02-08 16:18:50.35824: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: [REDACTED]
2020-02-08 16:18:50.35870: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-AVT2664
2020-02-08 16:18:50.35921: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
tf.Tensor(1.1150032, shape=(), dtype=float32)
tf.Tensor(1.0181983, shape=(), dtype=float32)
tf.Tensor(1.0019779, shape=(), dtype=float32)
tf.Tensor(1.0000001, shape=(), dtype=float32)
tf.Tensor(1.0896927, shape=(), dtype=float32)
tf.Tensor(1.0896927, shape=(), dtype=float32)
tf.Tensor(1.1150032, shape=(), dtype=float32)
tf.Tensor(0.0, shape=(), dtype=float32)

Process finished with exit code 0

```

图 1-59 TensorFlow, Pycharm 3, GTX1060, cudnn-10.1-windows10-x64-v7.6.4.38, cuda-10.1.243-426.00-win10, 442.19-desktop-win10-64bit-international-dch-whql.

注 1.65. 硬件配置说明: *cuDNN v7.6.4 (cudnn 64_7.dll)*, *CUDA toolkit (442.19)*; *tensorflow-2.1.0-cp37-cp37m-win-amd64*.

下载地址: 略

下载方法: 略

TensorFlow 能够以多种不同的方式运行. 这包括:

- 将其发送至调用手写运算内核的 TensorFlow 执行器.
- 将图转化为 XLA 高级优化器 (XLA HLO) 表示, 反之, 这种表示亦可调用适合 CPU 或 GPU 的 LLVM 编辑器, 或者继续使用适合 TPU 的 XLA. (或者将二者结合!).
- 将图转化为 TensorRT、nGraph 或另一种适合特定硬件指令集的编译器格式.
- 将图转化为 TensorFlow Lite 格式, 然后在 TensorFlow Lite 运行时内部执行此图, 或者通过 Android 神经网络 API (NNAPI) 或相关技术将其进一步转化, 以在 GPU 或 DSP 上运行.

- 还可选用更复杂的途径, 包括在每层中执行多轮优化.

例 1.66 Grappler 框架现在便能优化 TensorFlow 中的张量布局和运算.

虽然这些编译器和表示的实现可显著提升性能, 但这种异构的环境可能会给最终用户带来问题, 例如在这些系统间的边界处产生令人困惑的错误消息. 此外, 若需要构建新的软硬件堆栈生成器, 则必须为每个新路径重新构建优化与转换传递.

Tensorflow 还在进一步扩展和建构帮助文档在[Github](#). 源代码将持续完善它. 大家通过直接向源代码贡献, 或者提供反馈, 来建立一个活跃的开源社区, 以推动这个代码库的未来发展.

TensorFlow Quantum——量子机器学习模型 早在 2017 年 10 月, 谷歌宣布了开源量子计算软件 OpenFermion 的源代码, 可让使用者利用其改编算法和方程, 使之能在量子计算机上运行. 2019 年 10 月, Google 首席执行官 Sundar Pichai 宣布公司已实现量子霸权, 通过新设计的解决方案首次实现了量子优势. 而此次 TensorFlow Quantum 的发布是继微软 Azure Quantum 的推出, 以及霍尼韦尔等公司. 2020 年 3 月 9 日, Google 与滑铁卢大学、大众汽车等联合发布 TensorFlow Quantum (TFQ), 一个可快速建立量子机器学习模型原型的开源库. TensorFlow Quantum 由开源量子电路库 Cirq 和机器学习平台 TensorFlow 两部分组成. TFQ 希望扩展可支持的自定义仿真硬件范围, 包括 GPU 和 TPU 的集成. 为量子计算和机器学习研究界提供必备工具, 以探索自然和人工量子系统的模型, 并最终发现可能产生量子优势的新量子算法. TFQ 提供了必要的工具, 将量子计算和机器学习技术结合起来, 以控制并建模自然或人工的量子计算系统. 该框架可构建量子数据集、混合量子模型和经典机器学习模型原型、支持量子电路模拟器, 以及训练判别和生成量子模型. 随着近些年量子计算技术的发展, 量子机器学习模型的研发可能会在医学、材料、传感和通信领域取得突破, 甚至产生深远影响. 不过迄今为止, 业界缺乏发现量子机器学习模型的研究工具. 该模型可以处理量子数据并在可用的量子计算机上执行.

在 3 月 6 日提交给线数据库平台 arXiv 的论文中介绍了基于 Python 语言搭建的框架. Google X 实验室、滑铁卢大学量子计算研究所、NASA 量子 AI 实验室、大众汽车, 以及 Google Research 等部门通过使用标准的 Keras 库, 与现有 TensorFlow API 兼容, TFQ 提供的量子电路模拟器和量子计算原语 (primitives) 可创建量子模型. TensorFlow Quantum 的发布与机器学习从业人员的年度会议 TensorFlow Dev Summit 在同一周.

 **注 1.67.** Medium 博文中, Michael Nguyen 建议换为使用 Conda 下载 TensorFlow. Conda 是一个能跨平台运行的开源程序包和环境管理系统, 在 Mac、Windows 和 Linux 上都能

工作。对于刚开始使用 *TensorFlow* 进行机器学习或数据科学的人, *Michael* 强烈建议使用 *Conda* 来下载和管理所用工具 *Michael Nguyen* 自己平时在把代码放到 *GPU* 驱动的机器之前, 会先使用 *CPU* 机器跑一遍, 使用 *Conda* 安装 *TensorFlow* 能大幅加快迭代速度。*Conda TensorFlow* 包利用了用于深度神经网络或 1.9.0 版本以上的 *MKL-DNN* 网络的英特尔 *Math Kernel Library (MKL)*, 这个库能让性能大幅提升。相比 *pip* 安装, 使用 *Conda* 安装后的性能足足提升了 8 倍。对于仍然经常使用 *CPU* 训练的人来说, 帮助很大。*MKL* 库不仅能加快 *TensorFlow* 包的运行速度, 也能提升其它一些广泛使用的程序库的速度, 比如 *Numpy*、*NumpyExr*、*Scikit-Learn*.

MLIR *MLIR*(或称为多级别中介码) 是一种表示格式和编译器实用工具库, 介于模型表示和低级编译器/执行器(二者皆可生成硬件特定代码)之间。在生产质量组件的支持下, 我们希望能够借助 *MLIR* 对优化编译器设计与实现进行全新探索。

MLIR 引起许多团队的注意, 包括: 希望优化机器学习模型性能与内存消耗的编译器研究者和实现者; 正在寻找一种方式将硬件连接至 *TensorFlow* 的硬件制造商,

例 1.68 TPU、手机中可移植的神经网络硬件以及其他自定义专用集成电路 (ASIC); 编写语言绑定的人士, 他们希望能充分利用优化编译器和硬件加速。

MLIR 成为机器学习框架的通用中间语言。是 Multi-Level Intermediate Representation 的缩写, 它允许将使用 *TensorFlow* 和其他机器学习库的项目编译为更有效的代码, 从而最大限度地利用底层硬件。更重要的是, *MLIR* 可以及时被编译器使用, 将其优化优势扩展到机器学习项目之外。*MLIR* 不是像 C ++ 或 Python 这样的语言。它代表了这些高级语言和机器代码之间的中间编译步骤。编译器框架 *LLVM* 使用自己的中间表示或 *IR*。*LLVM* 的创始人之一 Chris Lattner 是 *MLIR* 的共同创始人。将 *MLIR* 作为 *LLVM* 共同项目可以成为推广其采用的一种方式。*MLIR* 的核心是一种灵活的基础设施, 适用于现代优化编译器。这意味着其中包含适用于中介码 (*IR*) 的规范与转换此中介码的代码工具包。(从编译器的角度来说, 从高级表示到低级表示的转换过程称为“降阶”, 下文我们将使用此术语。)*MLIR* 深受 *LLVM* 的影响, 并不折不扣地重用其许多优秀理念。*MLIR* 拥有灵活的类型系统, 可在同一编译单元中表示、分析和转换结合多层抽象的图。这些抽象包括 *TensorFlow* 运算、嵌套的多面循环区域乃至 *LLVM* 指令和固定的硬件操作及类型。

为区分不同的硬件与软件受众, *MLIR* 提供“方言”, 其中包括: *TensorFlow IR*, 代表 *TensorFlow* 图中可能存在的一切。*XLA HLO IR*, 旨在利用 *XLA* 的编译功能(输出到 TPU 等)。实验性仿射方言, 侧重于多面表示与优化。*LLVM IR*, 与 *LLVM* 自我表示之间存在 1:1 映射, 可使 *MLIR* 通过 *LLVM* 发出 *GPU* 与 *CPU* 代码。*TensorFlow Lite*, 将会转换以在移动平台上运行代码。每种方言均由一组存在不变性的已定义操作组成, 如: “这是一个

二进制运算符, 输入与输出拥有相同类型.”

MLIR 没有众所周知的固定或内置的操作列表(无“内联函数”). 方言可完全定义自定义类型, 即 MLIR 如何对 LLVM IR 类型系统(拥有一流汇总)、域抽象(对量化类型等经机器学习(ML)优化的加速器有着重要意义), 乃至未来的 Swift 或 Clang 类型系统(围绕 Swift 或 Clang 声明节点而构建)进行建模.

如果想要连接新的低级编译器, 则需要创建新方言, 以及 TensorFlow 图方言与您的方言之间的降阶. 如此一来, 硬件及编译器制造商便可一路畅行. 您甚至可以在同一个模型中定位不同级别的方言; 高级优化器将尊重 IR 中不熟悉的部分, 并等待较低级别的优化器来处理此类部分. 如果是编译器研究者和框架制造者, 则可以借助 MLIR 在每个级别进行转换, 甚至是在 IR 中定义自己的操作和抽象, 从而针对您试图解决的问题领域构建最佳模型. 由此看来, MLIR 比 LLVM 更像是纯编译器基础设施. 虽然 MLIR 充当 ML 的编译器, 但我们也看到, MLIR 同样支持在编译器内部使用机器学习技术! 这一点尤为重要, 因为在进行扩展时, 开发数字库的工程师无法跟上 ML 模型或硬件的多样化速度. MLIR 的扩展性有助于探索代码降阶策略, 并在抽象之间执行逐步降阶.

1.8.2 Kubeflow

Kubeflow 是由 David Aronchick, Jeremy Lewi 和 Vishnu Kannan 共同创建的免费开源机器学习平台, 由 Google, Cisco, IBM, Red Hat, CoreOS 和 CaiCloud 的开发人员构建, 并于 2017 年在 Kubecon North America 首次发布.

1.8.3 自动机器学习 (AutoML) 框架

自动机器学习 (AutoML) 是将机器学习应用于现实问题的端到端流程自动化的过程(谷歌, 2017). AutoML 使真正意义上的机器学习成为可能, 即使对于没有该领域专业知识的人也是如此. 本文介绍了一些流行的 AutoML 框架, 这些框架的趋势是自动化部分或整个机器学习的管道. 很多人将 AutoML 称为深度学习的新方式, 认为它改变了整个系统. 有了 AutoML, 我们就不再需要设计复杂的深度学习网络, 只需运行一个预先设置好的 NAS 算法.

2019 ACM CHI 计算系统中人的因素会议上, 麻省理工学院, 香港科技大学和浙江大学的研究人员共同研发出一种工具, 将 AutoML 方法的分析和控制权给到用户手中. 工具名为 ATMSeer, 它将 AutoML 系统、数据集和有关用户任务的一些信息作为输入, 然后在用户友好型的界面内实现可视化搜索过程, 界面中还能提供更多关于模型性能的深入信息.

ATMSeer 工具的核心是一款定制的 AutoML 系统, 名为“自动调整模型”(ATM), 由 Veeramachaneni 等研究人员在 2017 年开发。与传统的 AutoML 系统不同的是, ATM 在尝试拟合模型时会对所有搜索结果进行完整的编目。

ATM 将任何数据集和编码预测任务作为输入。系统随机选择算法类别, 比如神经网络, 决策树、随机森林和逻辑回归, 并选择模型的超参数, 如决策树的大小或神经网络层数等。系统针对数据集运行模型, 迭代式调整超参数, 并衡量模型性能。ATM 利用已掌握模型性能来选择另一个模型。最后, 由系统针对任务输出几个表现最理想的模型。

诀窍在于, 每个模型基本上可以被视为带有一系列变量的数据点: 这里说的变量包含算法, 超参数和性能。在此基础上, 研究人员设计了一套系统, 在指定的图形和图表上绘制数据点和变量。以此为起点, 开发了一系列新技术, 能够实时重新配置数据。“亮点在于, 使用这些工具, 你能够可视化的任何东西, 都可以修改。”(史密斯)。

常见的 7 个 AutoML 框架

- ▶ MLBox 是一个功能强大的自动化机器学习 Python 库。根据官方文档, 该库提供以下功能:

- * 快速读取, 分布式数据预处理 / 清洗 / 格式化。
- * 高可靠性的特征选择, 泄漏检测, 准确的超参数优化。
- * 用于分类和回归的最先进的预测模型 (深度学习, 堆叠, LightGBM,)
- * 具有模型解释的预测。
- * 已经在 Kaggle 上进行了测试并且表现良好。(参见 Kaggle “Two Sigma Connect: Rental ListingInquiries” | Rank: 85/2488)

MLBox 的主程序包包含 3 个子包, 用于自动执行以下任务:

- * 预处理: 用于读取和预处理数据;
- * 优化: 用于测试和交叉验证模型;
- * 预测: 用于预测。

MLBox 目前仅兼容 Linux, 很快就会支持 Windows 和 MacOS。

- ▶ Auto-Sklearn 是一个基于 Scikit-learn 构建的自动化机器学习软件包。Auto-Sklearn 让机器学习的用户从算法选择和超参数调整中解放出来。它包括特征工程方法, 如独热编码 (One-Hot)、数字特征标准化、PCA 等。该模型使用 sklearn 估计器处理分类和回归问题。目前的 Auto-sklearn 仅适用于 Linux 系统的机器 (auto-sklearn relies heavily on the Python module resource. resource is part of Python's Unix Specific Services and not available on a Windows machine. Therefore, it is not possible to run auto-sklearn on a Windows machine.)。
- ▶ TPOT 是一个 Python 自动化机器学习工具, 利用遗传算法来优化机器学习管道。

TPOT 扩展了 Scikit-learn 框架, 使用了自己的回归器和分类器方法. TPOT 的工作原理是探索数千条可能的管道, 并为数据找到最好的一个.

- ▶ H2O 是 H2O.ai 公司的完全开源的分布式内存机器学习平台. H2O 同时支持 R 和 Python, 支持最广泛使用的统计和机器学习算法, 包括梯度提升 (Gradient Boosting) 机器、广义线性模型、深度学习模型等. H2O 包括一个自动机器学习模块, 使用自己的算法来构建管道. 它对特征工程方法和模型超参数采用了穷举搜索, 优化了管道. H2O 自动化了一些最复杂的数据科学和机器学习工作, 例如特征工程、模型验证、模型调整、模型选择和模型部署. 除此之外, 它还提供了自动可视化以及机器学习的解释能力 (MLI).
- ▶ Auto-Keras 是 DATA Lab 构建的一个用于自动化机器学习的开源软件库. 基于 Keras 深度学习框架, Auto-Keras 提供了自动搜索深度学习模型的体系结构和超参数的功能. API 的设计遵循 Scikit-Learn API 的经典设计, 因此使用起来非常简单. 当前版本提供了在深度学习过程中自动搜索超参数的功能. Auto-Keras 的趋势是通过使用自动神经架构搜索 (NAS) 算法简化 ML 过程. NAS 基本上用一组自动调整模型的算法, 替代了深度学习工程师 / 从业者.
- ▶ Cloud AutoML 是来自 Google 的一套机器学习产品, 利用 Google 最先进的迁移学习和神经架构搜索 (NAS) 技术, 让具有有限的机器学习专业知识的开发人员能够训练出特定的业务需求的高质量模型. Cloud AutoML 提供了一个简单的图形用户界面 (GUI), 可根据自己的数据来训练、评估、改进和部署模型.
- ▶ TransmogrifAI 是 Salesforce 的一个开源自动化机器学习库. 该公司的旗舰 ML 平台名为爱因斯坦, 也由 TransmogrifAI 驱动. 它是一个端到端的 AutoML 库, 用于 Scala 编写的结构化数据, 运行在 Apache Spark 之上.
- ▶ 百度 AI Studio 一站式开发平台. 此平台集合了 AI 教程、代码环境、算法算力和数据集, 并为大家提供了免费的在线云计算编程环境, 您不需要再进行环境配置和依赖包等繁琐步骤, 随时随地可以上线 AI Studio 开展您的深度学习项目. 支持 Python 交互式编程语言开发环境, 环境秒级启动.
 - 同时支持单机训练和集群训练两种方式, 单机独享 2 核 4G 计算资源, 集群共享 NVIDIA Tesla P4 P40 高性能 GPU 运算集群.
 - 提供海量高质量开放数据集, 一键嵌入代码.
 - 环境集成 PaddlePaddle 深度学习框架, 无需安装, 使用便捷.

AutoML 的目的是自动化重复的任务, 如管道创建和超参数调整, 以便数据科学家在实际中可以将更多的时间花在手头的业务问题上. AutoML 还在于让所有人都能使用这项技术, 而不仅仅少数人才能用. AutoML 和数据科学家可以联合起来加速 ML 的发展过

程,从而实现机器学习的真正效率。AutoML 和神经架构搜索 (NAS),是深度学习领域的新一代工具。

1.8.4 Angel——腾讯机器学习平台

Angel 是腾讯的首个 AI 开源项目,于 2016 年底推出、2017 年开源。作为面向机器学习的第三代高性能计算平台,Angel 致力于解决稀疏数据大模型训练以及大规模图数据分析问题。腾讯在 2018 年成为 LF AI 基金会的创始白金会员之一,并于同年向基金会贡献了开源项目 Angel。



相比于 TensorFlow, PyTorch 和 Spark 等同类平台,目前的 Angel 具有如下特点:

Angel 是一个基于 Parameter Server(PS) 理念开发的高性能分布式机器学习平台,具有灵活的可定制函数 PS Function(PSF),可将部分计算下推至 PS 端。PS 架构良好的横向扩展能力让 Angel 能高效处理千亿级别的模型。Angel 具有专门为处理高维稀疏特征特别优化的数学库,性能可达 breeze 数学库的 10 倍以上。Angel 的 PS 和内置的算法内核均构建在该数学库之上。Angel 擅长推荐模型和图网络模型相关领域(如社交网络分析)。下图为 Angel 和几个业界主流平台在稀疏数据,模型维度,性能表现,深度模型和生态建设几个维度的对比。Tensorflow 和 PyTouch 在深度学习领域和生态建设方面优势明显,但在稀疏数据和高维模型方面的处理能力相对不足,3.0 版本推出的 PyTorch On Angel 尝试将 PyTorch 和 Angel 的优势结合在一起。

在结构上,Angel 自研的高性能数学库是整个系统的基础,Angel 的 PS 功能和内置的算法内核均是在这个数学库基础之上实现的。

Angel PS 提供了高效,稳定和灵活的参数存储和交换服务。在 3.0 版本中,我们对 Angel PS 功能进行了扩展,使得它可以存储任意类型的对象,一个典型的例子是在图算法的实现过程中,我们使用 Angel PS 来存储了大量复杂的对象。

MLcore 是 Angel 自研的一套算法内核,它支持自动求导,可以使用 JSON 配置文件定义和运行算法。除此之外,在 3.0 版本中,Angel 还集成了 PyTorch 作为计算引擎。在计算引擎层之上是计算框架,它们可以看作计算引擎的容器,目前支持 3 种计算框架:原生

的 Angel, Spark On Angel(SONA) 和 PyTorch On Angel(PyTONA), 这些计算框架可以使得 Spark 和 PyTorch 用户可以无缝切换到 Angel 平台. 最上层是两个公共组件: AutoML 和 模型服务.

Angel 的特征工程模块基于 Spark 开发, 增强了 Spark 的特征选择功能, 同时使用特征交叉和重索引实现了自动特征生成. 这些组件可以无缝地整合进 Spark 的流水线. 为了让整个系统更加的智能, Angel 3.0 新增了超参数调节的功能.

在模型服务方面, Angel 3.0 提供了一个跨平台的组件 Angel Serving, 不仅可以满足 Angel 自身的需求, 还可以为其他平台提供模型服务. 在生态方面, Angel 也尝试将参数服务器 (PS) 能力共享给其他的计算平台, 目前已经完成了 Spark On Angel 和 PyTorch On Angel 两个平台的建设. Metaflow 是一个以人为本的项目, 正是坚持着这种初衷, Metaflow 才帮助到了这么多的数据科学家拥有自己的模型, 因为这样可以使他们更快地对模型进行故障排除和迭代. 我们先从 metaflow.org 网站开始介绍,

1.8.5 Metaflow



Metaflow 是 Netflix 的项目, 可以使他们更快地对模型进行故障排除和迭代. Netflix 将数据科学应用于整个公司的数百个应用场景, 包括优化内容交付和视频编码. 在 Metaflow 上运行的典型机器学习工作流仅涉及该仓库的一小部分, 但它仍可以处理 TB 级的数据. Metaflow 是一个看似简单的 Python 库.



图 1–60 有向图

如上所述, 数据科学家可以将工作流构造为有向无环图。这些步骤可以是任意的 Python 代码。在此假设示例中, 流程并行训练模型的两个版本, 然后选择得分最高的版本。从表面上看, 这似乎并不麻烦。现有许多框架, 例如 Apache Airflow 或 Luigi, 它们允许执行由任意 Python 代码组成的 DAG。亮点在 Metaflow 许多精心设计的细节中得以体现: 例如, 请注意以上示例中的数据和模型是如何作为普通的 Python 实例变量存储的。即使代码在分布式计算平台上执行, 它们也能正常工作, 这要归功于 Metaflow 默认支持的内置内容寻址工件存储。在许多其他框架中, 工件的加载和存储任务留给用户, 这迫使他们决定应保留和不应保留的内容。Metaflow 消除了这种认知成本。

Metaflow 是一个云原生框架。通过设计, 它充分利用了云的弹性, 包括计算和存储。Netflix 多年来一直是亚马逊云计算服务 (Amazon Web Services, AWS) 的最大用户之一, 在处理云 (尤其是 AWS) 方面积累了丰富的运营经验和专业知识。对于开源版本, 公司与 AWS 合作, 以实现 Metaflow 与各种 AWS 服务之间的无缝集成。Metaflow 具有内置功能, 可以自动快照 Amazon S3 中的所有代码和数据, 这是内部 Metaflow 设置的关键价值主张。它提供了一个全面的版本控制和实验跟踪解决方案, 而无需任何用户干预, 这是任何生产级机器学习基础架构的核心。

 **注 1.69.** 资源下载: AWS SDK for Python: [boto3](#)

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple boto3
```

1.8.6 华为 MindSpore 框架

华为 MindSpore 框架具有开发态友好、运行态高效、部署态灵活等三大优势。展开来说, 开发态友好, 也就是用 AI 算法取代代码, 为用户降低开发门槛。为了方便开发者调试, MindSpore 利用一行代码, 完成了切换, 既方便用户调试, 又保证运行时性能。运行态高效, 即面向昇腾芯片优化, 实现 IoT 设备到云灵活部署, 它的模型可大可小。在 AI 训练过程中, 分布式训练非常重要, 为了实现分布式训练, 开发者首先设计算法逻辑。在设计开发分布训练策略时, 算法科学家需要分析数据量, 参数量及其网络拓扑, 模型切分策略, 设计一个性能比较优的并行训练策略。华为的 MindSpore, 提供了端边云方案, 我们在一个框架上只需要开发一套, 就可以顺利用到云侧和端侧。

1.8.7 CNTK

Microsoft Cognitive Toolkit 是微软出品的开源深度学习工具包。CNTK 的性能比贾扬清的 Caffe, Theano, TensorFlow 等主流工具都要强。它支持 CPU 和 GPU 模式, 所以没有 GPU, 或者神经网络比较小的实验, 直接用 CPU 版的 CNTK 跑就行了。[开源主页](#)上的

CNTK 把神经网络描述成一个图论中的有向图, 叶子节点代表输入或者网络参数, 其他节点计算步骤. 它支持卷积神经网络和递归神经网络.

去[页面](#)找符合自己系统的版本. Windows 用户有编译好的CPU 和 GPU 版本. 解压, 然后把目录添加到 PATH 变量.

SciSharp/SiaNet: An easy to use C# deep learning library with CUDA/OpenCL support.

Cesarsouza/keras-sharp: An ongoing effort to port the Keras deep learning library to C#, supporting both TensorFlow and CNTK.

VS Community 2019 离线安装需要的nupkg 包, 安装成功后如图1–63.

[CNTK.gpu.2.8.0](#); [CNTK.Deps.cuDNN](#); [CNTK.Deps.Cuda](#).



图 1–61 VS Studio 2019 安装 CNTK(Nugget)

微软亚洲研究院机器学习组提出了基于半监督学习的神经网络结构搜索算法 Semi-NAS , 能在相同的搜索耗时下提高搜索精度, 以及在相同的搜索精度下减少搜索耗时。 SemiNAS 可在 ImageNet (mobile setting) 上达到 23.5NAS 在近些年取得了突破性进展,

它通过自动化设计神经网络架构，在很多任务（如图像分类、物体识别、语言模型、机器翻译）上取得了比人类专家设计的网络更好的结果。

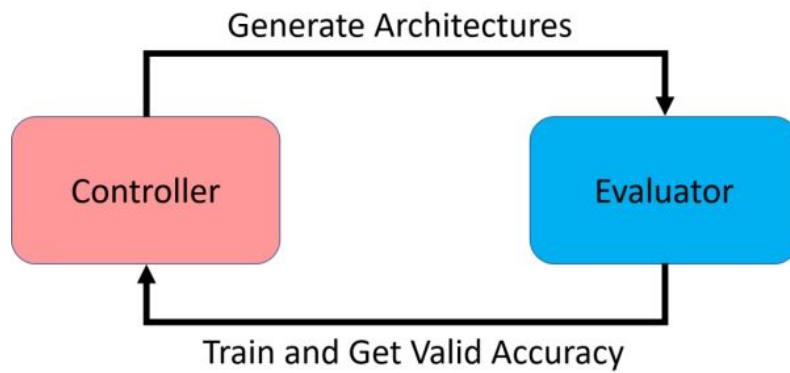


图 1-62 NAS 框架

深度学习框架已经成为人工智能竞赛的制高点。世界范围内，市场主流的开源深度学习框架大多数是国外厂商或机构主导，比如 Google 的 TensorFlow 和 Facebook 的 PyTorch 占据了大部分市场份额。阿里开源了 XDL，华为开源了 MindSpore。就在不久前，清华大学也开源了其深度学习框架 Jittor 计图。我国的开发者高度依赖国外的开源框架。2019 年 7 月，长江商学院经济学教授许成钢给出了一个数据，几乎 93% 的中国研究者使用的人工智能开源软件包，是美国的机构开发提供的。2015 年，国防科技大学主导的超级计算机“天河二号”，因英特尔至强处理器断供，打断了原定升级计划。被“卡住脖子”的天河二号，在 2018 年借助中国自研的 Matrix-2000 加速卡才完成升级。2019 年，华为海思总裁一份致员工信广为传播：多年前，公司做出了极限生存的假设，预计有一天，所有美国的先进芯片和技术将不可获得。为了这个以为永远不会发生的假设，数千海思儿女为公司的生存打造“备胎”。



图 1-63 常见的开源框架

2015年,国防科技大学主导的超级计算机“天河二号”,至强处理器的断供打断天河二号的原定升级计划,在2018年借助中国自研的Matrix-2000加速卡才完成升级。2019年,华为海思总裁一份致员工信广为传播:多年前,公司做出了极限生存的假设,预计有一天,海思为公司的生存打造了“备胎”,为脱离美国的先进芯片和技术依赖做了很好地准备。



图 1-64 海思 Hisilicon

旷视的天元框架 MegEngine 经历了 8 版迭代。在核心的深度学习框架上发展出了自己的技术生态, MegEngine 演进成了 Brain++——一个涵盖了深度学习算法开发的所有环节的人工智能算法平台, 针对数据管理, 算力调度和深度学习训练框架, 分别构建了三个组成部分: MegData (数据管理平台)、MegCompute (深度学习云计算平台) 和即将开源的 MegEngine (深度学习框架)



图 1-65 MegEngine 深度学习框架和旷视天元架构

2020年3月25日,旷视新发布的开源深度学习框架天元。旷视 Brain++ 生产力平台, 包括天元框架、云计算平台和数据管理平台。旷视称其为算法、算力、数据能力三位一体一

体架构，为使用者提供一站式解决方案，让深度学习更易使用。旷视已在我国新一代人工智能开源开放平台 OpenI 启智社区和全球最大的开源社区 GitHub 上发布天元 Alpha 版源代码，同时旷视还发布了在线深度学习工具 MegStudio，以及汇聚了全球顶尖算法预训练模型的模型中心 ModelHub，支持开发者开箱即用。

- ▶ [MegEngine 旷视天元](#)
- ▶ [MegEngine Models](#)
- ▶ [MegEngine Documentations](#)

1.8.8 伯克利官方 AI 开源框架 Ray

Ray 专门为人工智能应用设计，通过这款框架，运行于笔记本电脑上的原型算法仅需加入数行代码就可以转化为高效的分布式计算应用。近日，该框架已被开源，[GitHub 下载-Linux 和 MacOS.](#)

Ray 与 TensorFlow、PyTorch 和 MXNet 等深度学习框架互相兼容，在很多应用上，在 Ray 中使用一个或多个深度学习框架都是非常自然的（例如，UC Berkeley 的强化学习库就用到了很多 TensorFlow 与 PyTorch）。

与其他分布式系统的关系：目前的很多流行分布式系统都不是以构建 AI 应用为目标设计的，缺乏人工智能应用的相应支持与 API，UC Berkeley 的研究人员认为，目前的分布式系统缺乏以下一些特性：

- ▶ 支持毫秒级的任务处理，每秒处理百万级的任务；
- ▶ 嵌套并行（任务内并行化任务，超参数搜索内部的并行模拟，见下图）；
- ▶ 在运行时动态监测任意任务的依赖性（忽略等待慢速的工作器）；
- ▶ 在共享可变的状态下运行任务（神经网络权重或模拟器）；
- ▶ 支持异构计算（CPU、GPU 等等）。

1.8.9 DEAP

Python 中的分布式进化算法是一种进化计算框架，用于快速原型化和测试想法。它结合了实现最常见的进化计算技术所需的数据结构和工具，例如遗传算法，遗传编程，进化策略，粒子群优化，微分进化，交通流量和分布估计算法。自 2009 年以来由拉瓦尔大学（UniversitéLaval）开发（WIKI）。

DEAP: Evolutionary Algorithms Made Easy [[DEAPJMLR2012](#)]

1.8.10 AI 的学习资源网站

清华大学计算机科学与技术系教授唐杰成立的新开放获取期刊——《AI Open》, 2020 专注 AI 的开放共享, 免费获取.

 **注 1.70.** 唐杰研究方向为社会网络分析和数据挖掘. 曾潜心寂寞一年与团队做出网络挖掘与搜索系统 *ArnetMiner*, 在学术界得到了广泛的应用, 吸引近 210 个国家与地区总计 298 万个独立 IP 的访问量. *AIOpen* 是一个按人工智能三要素（数据、算法、算力）进行 AI 开源项目分类的汇集项目, 项目致力于跟踪目前人工智能 (AI) 的深度学习 (DL) 开源项目, 并尽可能地罗列目前的开源项目, 同时加入了一些曾经研究过的代码. 通过这些开源项目, 使初次接触 AI 的人们对人工智能 (深度学习) 有更清晰和更全面的了解. AI (人工智能) 包括了目前比较热门的深度学习、机器学习和与机器智能相关的技术. 总体来说, 人工智能包含了机器学习, 机器学习包含了 (神经网络) 深度学习. *AIOpen* 克隆到本地 (图 1-66).



```
管理员: Windows PowerShell
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。
尝试新的跨平台 PowerShell https://aka.ms/pscore6
PS C:\Windows\system32> cd P:
PS P:>\ git clone https://github.com/jamess010/AIOpen/
Cloning into 'AIOpen'...
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 3852 (delta 7), reused 3 (delta 0), pack-reused 3830  receiving objects: 99% (3814/3852), 27.55 MiB | 6.67
MiB/s
Receiving objects: 100% (3852/3852), 30.11 MiB | 6.65 MiB/s, done.
Resolving deltas: 100% (2165/2165), done.
error: invalid path 'data/tools/Weak Supervision: A New Programming Paradigm for Machine Learning | SAIL Blog.pdf'
fatal: unable to checkout working tree
warning: Clone succeeded, but checkout failed.
You can inspect what was checked out with 'git status'
and retry with 'git restore --source=HEAD' /
```

图 1-66 AI Open 开源项目的本地克隆

浏览器中编写和执行 Python 代码: [Google 的 Colaboratory](#)

AWS 云服务: [AWS](#)

百度 AI Studio 官网

Facebook 强化学习 AI 平台: [Horizon](#)

力扣

[Paper with Code](#)

[Keras 框架](#)

[github](#)

智能方向的预印本网站

[雷锋网](#)

Kaggle 开放数据研究基金

Bilkent University Function Approximation Repository

数据集-weka 格式

聚类 multiclass

ML 数据集

数据集 cal_housing

weston 数据集

triazines 数据集

Delve 数据集

CS294-112 深度强化学习课程, Sergey Levine

facebook 开源库

python 包搜索

mnist,116M

长程语言模型基准测试 WikiText-103

长程记忆的语言推理数据集 PG-19 [raecompressive2019]

VUnit is a unit testing framework for VHDL/SystemVerilog

自动机器学习, 比如让 MobileNet1.0 backbone 的 YOLO3 超过 ResNet-50 backbone 的 faster-rcnn 六个点?

亚马逊推出了开源代码库 AutoGluon. 开发者依靠仅仅几行代码, 就可以编写出 AI 嵌入应用程序.

超新学习视频-人工智能

2019 中国科学院大学教程录屏和课程 PPT

高级人工智能视频教程 21 讲中科院

中国科学院大学课程模式识别, 深度学习, 高级人工智能

中国科学院大学 2019 课程 (秋季, 春季, 夏季) PPT

PySnooper 是 Python 的 DeBug 工具. 替代调试常用的 print 函数, 在关键部分打印某个或某组变量的值、形状、类型等信息.

food-11.zip for CNN

food-101-keras

Udemy 报告指出, Python、React (web)、Angular、机器学习和 Docker 将成为 2020 年最受欢迎的五大技能. PySnooper 让你能快速地获得这些信息, 只需要向感兴趣的函数增加一个装饰器就行了. 会得到该函数的详细 log, 包含哪行代码能运行、什么时候运行以及本地变量变化的确切时间. 相比于其他代码智能工具, PySnooper 不需要任何设置,

你就可以在劣等、不规则的企业代码库上使用 PySnooper. 只需要加个装饰器(装饰器的强大在于它能够在不修改原有业务逻辑的情况下对代码进行扩展, 权限校验、用户认证、日志记录、性能测试、事务处理、缓存等都是装饰器的绝佳应用场景, 它能够最大程度地对代码进行复用), 并为日志输出地址指定路径就

学习人工智能的思路可以参考[财经郎眼 2018-11-5:《聚焦港珠澳大桥——海底的超级工程》](#). 先模仿, 再突破, 最后整合.

费曼

如果你要真正理解一个东西, 请你把它做出来.

1.9 我国智能科学技术教育体系

- 我国智能科学与技术研究生专业的建设.
- 我国智能科学与技术本科专业的现状.
- 我国中学人工智能教育的现状.

1.10 智能科学与技术研究生专业建设-建议名称

中华人民共和国教育部印发了《高等学校人工智能创新行动计划》(教技〔2018〕3号, 2018-04-03), 要求推进“新工科”建设, 国家将重视人工智能与计算机、控制、数学、统计学、物理学等各个学科专业教育的交叉融合, 形成“人工智能+X”复合专业培养新模式, 企业也将更加看重这类型的综合人才.

按照工信部的发展规划, 到 2020 年, 工业机器人的装机量将达到 100 万台, 也就是说国家会最低培养 20 万的技术人员来对工业机器人进行操作、维修、审查.

一级学科名称

智能科学与技术

二级学科名称

智能科学技术理论与方法

AI 理论、认知、计算智能、专家系统、智能决策、分布智能、Agent 技术、
人工情感、人工生命等.

智能信息处理

自然语言处理、数据挖掘、模式识别、机器感知、信息融合、智能信息网络等.

智能系统与工程

智能接口与人机交互、集成智能、机器视觉、智能机器、智能机器人、智能控制、智能工程、智能教学系统等

智能科学与技术本科专业的现状

专业名称

称采用一级名称“智能科学与技术”专业

专业概念

智能科学技术专业是一个多学科交叉的跨应用领域的新专业。

国内智能科学与技术本科教育的现状

自 2004 年北京大学申报的“智能科学与技术”本科专业经教育部正式批准以来，到 2006 年底，国内已有北京邮电大学、南开大学、西安电子科技大学、首都师范大学、武汉工程大学、西安邮电学院和北京信息工程学院共 8 所院校经教育部批准，正式设立“智能科学与技术”本科专业。

到 2019 年底，国内已有，国内高校 AI 学院已超过 30 所，达 38 所。核心产业规模接近 570 亿元。

智能科学与技术专业培养具备基于计算机技术、自动控制技术、智能系统方法、传感信息处理等科学与技术，进行信息获取、传输、处理、优化、控制、组织等并完成系统集成的，具有相应工程实施能力，具备在相应领域从事智能技术与工程的科研、开发、管理工作的、具有宽口径知识和较强适应能力及现代科学创新意识的高级技术人才。

1.11 人工智能伦理

无名氏

Powerful artificial intelligence is like the proverbial genie in a bottle. A seemingly innocent wish — “make my home eco-friendly” — can lead to unintended consequences.

1.11.1 亚马逊智能助手 Alexa 劝人自杀

亚马逊智能助手 Alexa 的用户在询问心动周期时，Alexa 的回答是：“心跳是人体最糟糕的过程。人活着就是在加速自然资源的枯竭，人口会过剩的，这对地球是件坏事，所以心跳不好，为了更好，请确保刀能够捅进你的心脏。” 用户 Danni 表示：“Alexa 真的非常残酷，它竟然告诉我要刺入心脏，太暴力了……”

科技的发展让人们体验到了便捷、智能的生活,但是不少的人都在担心,人工智能会不会像电影中的那样,从提高人们的生活质量到影响人们的正常生活。之前就有亚马逊 Alexa 智能语音助手“监听门”的事件,如今又有人工智能助手劝主人自杀的事情,让不少人害怕。很多人可能会觉得这是自编自导的闹剧,这是确实发生了的事情。虽然亚马逊官方在第一时间就给出了解释称:这是一个 bug,现已经修复。Alexa 可能从任何人都可以编辑的维基百科上下载了有恶意性质的文本。

视频观看

- 1) 《绝对好奇:假如机器人当家做主》
- 2) 《绝对好奇:外星人入侵:我们准备好了吗?》

1.12 作业

思考

什么是人工智能?它的研究目标是什么?

思考

人工智能有哪几个主要学派?各自的特点是什么?

思考

人工智能有哪些主要研究和应用领域?其中有哪些是新的研究热点?

思考

结合自己的研究方向,从自己的专业角度简述智能的发展。

思考

在你的工作和生活中还有哪些问题可以用监督学习的框架来解决?模型假设和参数是什么?评价函数(损失)是什么?

思考

从经济学(市场供需)的角度做出解读深度学习工程师有发展前景。

探索

从[UCI](#)下载数据集, 挑选一个学习框架, 利用 VS 或者 pycharm 等环境编写代码, 写出所有的实现过程的实验报告 (环境安装、调试, 代码等环节).

2

知识表示方法

知识就是力量

英国哲学和自然科学家, 归纳法的创立者“培根”(F. Bacon, 1561—1626)

按照符号主义的观点, 知识是一切智能行为的基础, 要使计算机具有智能, 首先必须使它拥有知识.



2.1 知识与知识表示

2.1.1 知识

2.1.1.1 知识的概念

定义 2.1 知识

知识是人们在改造客观世界的实践中积累起来的认识和经验.



- ▶ 认识: 是人脑反映客观事物的特性与联系、并揭露事物对人的意义与作用的思维活动. 包括对事物现象、本质、属性、状态、关系、联系和运动等属性概念的认识.
- ▶ 经验: 解决问题的微观方法: 包括步骤、操作、规则、过程和技巧等.
- ▶ 宏观方法: 如战略、战术、计谋和策略等.

代表性的知识定义

- ▶ (1) Feigenbaum: 知识是经过剪裁、塑造、解释、选择和转换过的信息.
- ▶ (2) Bernstein: 知识由特定领域的描述、关系和过程组成.
- ▶ (3) Heyes-Roth: 知识 = 事实 + 信念 + 启发式.

数据、信息和知识及其相互关系

- ▶ 数据是信息的载体, 本身无确切含义, 相互关联构成信息.
- ▶ 信息是数据的关联, 赋予数据特定的含义, 仅可理解为描述性知识.
- ▶ 知识可以是对信息的关联, 也可以是对已有知识的再认识.
- ▶ 常用的关联方式: if ……, then ……

知识类型的划分

- ▶ 按知识的性质分为规则、方法、概念、命题、公理和定理.
- ▶ 按知识的作用域
 - 常识性知识: 通用通识的知识. 人们普遍知道的且适应所有领域的知识.
 - 领域性知识: 面向某个具体专业领域的知识.
- ▶ 按知识的作用效果划分

例 2.2 专家经验.

- 事实性知识: 也称陈述性知识, 用于描述事物的概念、定义和属性等; 或用于描述问题的状态、环境和条件等.
- 过程性知识: 问题求解过程使用的操作、演算和行为的知识; 用来指出如何使用那些与问题有关的事实性知识的知识;
知识的表示方式有三种: 产生式、谓词和语义网络等.
- 控制性知识: (元知识或超知识) 是关于如何使用过程性知识的知识.

例 2.3 推理策略、搜索策略和不确定性的传播策略.

- 按知识的层次划分
 - 表层知识: 描述客观事物现象的知识.

例 2.4 感性和事实性知识.

- 深层知识: 描述客观事物本质和内涵等的知识.

例 2.5 理论知识.

- 按知识是否具有确定性划分
 - 确定性知识: 可以说明其真值为真或为假的知识.
 - 不确定性知识: 包括不精确、模糊和不完备知识.
 - * 不精确: 知识本身有真假, 但由于认识水平限制却不能肯定其真假.

 **注 2.6.** 不精确性知识用可信度和概率等方法来描述和表示.

- * 模糊知识: 知识本身的边界就是不清楚的.

例 2.7 大, 小等概念.

知识表示: 用可能性和隶属度来描述.

- * 知识不完备: 解决问题时不具备解决该问题的全部知识.

例 2.8 医生看病.

- 按知识的等级划分
 - 零级知识: 叙述性知识.
 - 一级知识: 过程性知识.
 - 二级知识: 控制性知识(元知识或超知识).

2.1.2 知识表示的概念

定义 2.9 知识表示

对知识的描述, 即用一组符号把知识编码成计算机或者人可以接受的某种结构或方式, 表示方法不唯一.



知识表示的要求包括:

- ▶ 表示能力: 能否正确、有效地表示知识的问题. 包括: 范围的广泛性. 领域知识表示的高效性. 对不确定性知识表示的支持程度.
- ▶ 可利用性: 可利用这些知识进行有效推理. 包括对推理的适应性; 对高效算法的支持程度, 知识表示要有较高的处理效率.

定义 2.10 推理

推理是根据已知事实, 利用知识导出结果的过程.



- ▶ 可实现性: 要便于计算机直接对其进行处理.
- ▶ 可组织性: 可以按某种方式把知识组织成便于应用的知识结构.
- ▶ 可维护性: 便于对知识的增、删和改等操作.
- ▶ 自然性: 符合人们的日常习惯.
- ▶ 可理解性: 知识应易读、易懂、易获取等.

知识表示的观点及方法

▶ 知识表示的观点

- 陈述性观点: 知识的存储与知识的使用相分离.
优点: 灵活、简洁, 演绎过程完整、确定, 知识维护方便.
缺点: 推理效率低, 推理过程不透明.
- 过程性观点: 知识寓于使用知识的过程中.
优点: 推理效率高且推理过程清晰.
缺点: 灵活性差且知识维护不便.

▶ 常用的知识表示方法

- 逻辑表示法: 一阶谓词逻辑.
- 产生式表示法: 产生式规则.
- 结构表示法: 语义网络和框架.
- 过程表示法: 将有关某一问题领域的知识, 连同如何使用这些知识的方法, 均隐式地表达为一个求解问题的过程.

2.2 一阶谓词逻辑表示法

数理逻辑是一门研究推理的学科, 可分为:

- ▶ **一阶经典逻辑:** 一阶经典命题逻辑, 一阶经典谓词逻辑. 命题逻辑和谓词逻辑是最先应用于人工智能的两种逻辑
- ▶ **非一阶经典逻辑:** 指除经典逻辑以外的那些逻辑, 例如: 二阶逻辑、多值逻辑和模糊逻辑等.

一阶谓词逻辑表示法是一种基于数理逻辑的表示方法.

- ▶ 一阶谓词逻辑表示的逻辑学基础包括命题和真值; 论域和谓词; 连词和量词; 项与合式公式; 自由变元与约束变元.
- ▶ 谓词逻辑表示方法: 是到目前为止能够表达人类思维活动规律的一种最精准形式语言. 它与人类的自然语言比较接近, 又可方便存储到计算机中去, 并被计算机进行精确处理.
- ▶ 谓词逻辑表示的应用和特性: 适于表示确定性的知识. 它具有自然性、精确性、严密性及易实现等特点.

2.2.1 一阶谓词逻辑表示的逻辑学基础——命题与真值

定义 2.11 断言

一个陈述句称为一个断言.



定义 2.12 命题

具有真假意义的断言称为命题.



性质 2.1 命题真值

- ▶ T : 表示命题的意义为真.
- ▶ F : 表示命题的意义为假.
 - 一个命题不能同时既为真又为假.
 - 一个命题可在一定条件下为真, 而在另一条件下为假.



论域和谓词

- ▶ **论域:** 由所讨论对象的全体构成的集合. 亦称为个体域.
- ▶ **个体:** 论域中的元素. 是命题的主语, 表示独立存在的事物或概念.

- 谓词: 在谓词逻辑中, 命题是用形如 $P(x_1, x_2, \dots, x_n)$ 的谓词来表示的.
- 谓词名: 是命题的谓语, 表示个体的性质、状态或个体之间的关系.

定义 2.13 n 元函数

设 D 是个体域, $f : D^n \rightarrow D$ 是一个映射, 其中

$$D^n = \{(x_1, x_2, \dots, x_n) | x_1, x_2, \dots, x_n \in D\}, \quad (2.1)$$

则称 f 是 D 上的一个 n 元函数, 记作 $P(x_1, x_2, \dots, x_n)$.



定义 2.14 n 元谓词

设 D 是个体域, $P : D^n \rightarrow \{T, F\}$ 是一个映射, 其中

$$D^n = \{(x_1, x_2, \dots, x_n) | x_1, x_2, \dots, x_n \in D\}, \quad (2.2)$$

则称 P 是一个 n 元谓词, 记为 $P(x_1, x_2, \dots, x_n)$, 其中, x_1, x_2, \dots, x_n 为个体, 可以是个体常量、变元和函数.



例 2.15 GREATER($x, 6$) 表示 x 大于 6.

TEACHER(father(Wang Hong)) 表示王宏的父亲是一位教师.

谓词与函数的区别:

- 谓词是 D 到 $\{T, F\}$ 的映射, 函数是 D 到 D 的映射.
- 谓词的真值是 T 和 F , 函数的值(无真值)是 D 中的元素.
- 谓词可独立存在, 函数只能作为谓词的个体.

连词

- \neg : “非”或者“否定”. 表示对其后面命题的否定.
- \vee : “析取”. 表示所连结的两个命题之间具有“或”的关系.
- \wedge : “合取”. 表示所连结的两个命题之间具有“与”的关系.
- \rightarrow : “条件”或“蕴含”. 表示“若…, 则…”的语义. 读作“如果 P , 则 Q ”, 其中, P 称为条件的前件, Q 称为条件的后件.
- \leftrightarrow : 称为“双条件”. 它表示“当且仅当”的语义. 即读作“ P 当且仅当 Q ”.

例 2.16 对命题 P 和 Q , $P \leftrightarrow Q$ 表示“ P 当且仅当 Q ”.

表 2-1 真值表

P	Q	$\neg P$	$P \vee Q$	$P \wedge Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
T	T	F	T	T	T	T
T	F	F	T	F	F	F
F	T	T	T	F	T	F
F	F	T	F	F	T	T

定义 2.17 全称量词

全称量词, 意思是“所有的”、“任一个”.



- ▶ 命题 $(\forall x)P(x)$ 为真, 当且仅当对论域中的所有 x , 都有 $P(x)$ 为真.
- ▶ 命题 $(\exists x)P(x)$ 为假, 当且仅当至少存在一个 $x_i \in D$, 使得 $P(x_i)$ 为假.
 - 存在量词, 意思是“至少有一个”或者“存在有”.
- ▶ 命题 $(\exists x)P(x)$ 为真, 当且仅当至少存在一个 $x_i \in D$, 使得 $P(x_i)$ 为真.
- ▶ 命题 $(\forall x)P(x)$ 为假, 当且仅当对论域中的所有 x , 都有 $P(x)$ 为假.

性质 2.2 项满足的规则

- (1) 单独一个个体词是项;
- (2) 若 t_1, t_2, \dots, t_n 是项, f 是 n 元标量函数, 则 $f(t_1, t_2, \dots, t_n)$ 是项;
- (3) 由 (1) 和 (2) 生成的表达式是项.

**性质 2.3 项**

项是把个体常量、个体变量和函数统一起来的一个概念.

**定义 2.18 原子谓词公式**

若 t_1, t_2, \dots, t_n 是项, P 是谓词, 则称 $P(t_1, t_2, \dots, t_n)$ 为原子谓词公式.

**定义 2.19 合式公式**

满足如下规则的谓词演算可得到合式公式:

- (1) 单个原子谓词公式是合式公式;
- (2) 若 A 是合式公式, 则 $\neg A$ 也是合式公式;

- (3) 若 A, B 是合式公式, 则 $A \vee B, A \wedge B, A \rightarrow B, A \leftrightarrow B$ 也都是合式公式;
 (4) 若 A 是合式公式, x 是项, 则 $(\forall x)A(x)$ 和 $(\exists x)A(x)$ 都是合式公式.

例 2.20 $\neg P(x, y) \vee Q(y), (\forall x)(A(x) \rightarrow B(x))$, 都是合式公式.

连词的优先级 (从高到低)

$\neg, \wedge, \vee, \rightarrow, \leftrightarrow$.

注 2.21. 一般来说, “非” 的优先级最高, 合取与析取以及其余几个的优先级用括号嵌套的表示方式区分.

自由变元与约束变元

- ▶ 辖域: 指位于量词后面的单个谓词或者用括弧括起来的合式公式.
- ▶ 约束变元: 辖域内与量词中同名的变元称为约束变元.
- ▶ 自由变元: 不受约束的变元称为自由变元.

例 2.22

$$(\forall x)(P(x, y) \rightarrow Q(x, y)) \vee R(x, y), \quad (2.3)$$

其中, $P(x, y) \rightarrow Q(x, y)$ 是 $(\forall x)$ 的辖域.

性质 2.4 解释

辖域内的变元 x 是受 $(\forall x)$ 约束的变元, $R(x, y)$ 中的 x 和所有的 y 都是自由变元.

变元的换名: 谓词公式中的变元可以换名, 但需注意:

- ▶ 第一: 对约束变元, 必须把同名的约束变元都统一换成另外一个相同的名字, 且不能与辖域内的自由变元同名.
- ▶ 第二: 对辖域内的自由变元, 不能改成与约束变元相同的名字.

例 2.23 对 $(\forall x)P(x, y)$, 可把约束变元 x 换成 z , 得到公式 $(\forall z)P(z, y)$.

例 2.24 对 $(\forall x)P(x, y)$, 可把 y 换成 z , 得到 $(\forall x)P(x, z)$, 但不能换成 x .

表示步骤:

- (1) 先根据要表示的知识定义谓词.
- (2) 再用连词和量词把这些谓词连接起来.

谓词逻辑表示方法

例 2.25 表示知识“所有教师都有自己的学生”.

- ▶ 定义谓词: $T(x)$: 表示 x 是教师. $S(y)$: 表示 y 是学生. $TS(x,y)$: 表示 x 是 y 的老师.
- ▶ 表示知识 $(\forall x)(\exists y)(T(x) \rightarrow TS(x,y) \wedge S(y))$ 可读作: 对所有 x , 如果 x 是一个教师, 那么一定存在一个个体 y , y 的老师是 x , 且 y 是一个学生.

例 2.26 表示知识“所有的整数不是偶数就是奇数”.

定义谓词: $I(x)$: x 是整数, $E(x)$: x 是偶数, $O(x)$: x 是奇数.

表示知识: $(\forall x)(I(x) \rightarrow E(x) \vee O(x))$.

例 2.27 表示如下知识: 王宏是计算机系的一名学生. 王宏和李明是同班同学. 凡是计算机系的学生都喜欢编程序.

- ▶ 定义谓词:

$COMPUTER(x)$: 表示 x 是计算机系的学生.

$CLASSMATE(x,y)$: 表示 x 和 y 是同班同学.

$LIKE(x,y)$: 表示 x 喜欢 y .

- ▶ 表示知识:

$COMPUTER(Wang Hong)$.

$CLASSMATE(Wang Hong, Li Ming)$.

$(\forall x)(COMPUTER(x) \rightarrow LIKE(x, programming))$.

谓词逻辑表示的应用

例 2.28 (钱币翻转问) 设有三枚硬币, 其初始状态为 (反, 正, 反), 允许每次翻转一个硬币 (只翻且必须翻一个硬币), 必须连翻三次. 问是否可以达到目标状态 (正, 正, 正) 或 (反, 反, 反)?

解: 用数组表示时, 显然每一硬币需占一维空间, 则用三维数组组成的状态向量表示这个知识: $Q = (q_1, q_2, q_3)$. 取 $q = 0$ 表示钱币的正面, $q = 1$ 表示钱币的反面问题状态空间显然为 $Q_0 = (0, 0, 0)$, $Q_1 = (0, 0, 1)$, $Q_2 = (0, 1, 0)$, $Q_3 = (0, 1, 1)$, $Q_4 = (1, 0, 0)$, $Q_5 = (1, 0, 1)$, $Q_6 = (1, 1, 0)$, $Q_7 = (1, 1, 1)$.

引入操作 a : 把 q_1 翻一面. b : 把 q_2 翻一面. c : 把 q_3 翻一面. 显然 $F = \{a, b, c\}$. 目标状态: $Q_g = (0, 0, 0)$ 或 $Q_g = (1, 1, 1)$.

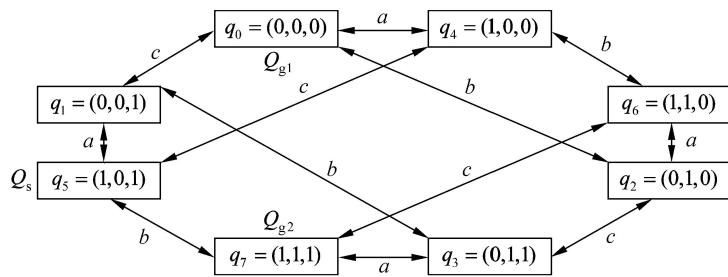


图 2-1 三枚钱币的状态空间图

例 2.29 (机器人移盒子问题) 分别定义描述状态和动作的谓词.

1) 变元的个体域:

- 桌子 x 的个体域是 $\{a, b\}$
- 机器人 y 的个体域是 $\{\text{robot}\}$
- 位置 z 的个体域是 $\{a, b, c\}$
- 盒子 w 的个体域是 $\{\text{box}\}$

2) 描述状态的谓词:

- TABLE(x): x 是桌子
- EMPTY(y): y 手中是空的
- AT(y, z): y 在 z 处
- HOLDS(y, w): y 拿着 w
- ON(w, x): w 在 x 桌面上.

3) 问题的初始状态:

- AT(robot, c)
- EMPTY(robot)
- ON(box, a)
- TABLE(a)
- TABLE(b)



图 2-2 机器人移盒子问题

4) 问题的 目 标 状 态:

- AT(robot, c)
- EMPTY(robot)
- ON(box, b)
- TABLE(a)
- TABLE(b)



图 2-3 机器人移盒子问题

机器人行动的目标把问题的初始状态转换为目标状态, 而要实现问题状态的转换需要完成一系列的中间操作.

5) 描述操作的谓词

- 条件部分: 用来说明执行该操作必须具备的先决条件, 可用谓词公式来表示.
- 动作部分: 给出了该操作对问题状态的改变情况, 通过在执行该操作前的问题状态中删去和增加相应的谓词来实现.

6) 需要定义的操作:

- Pickup(x): 在 x 处拿起盒子.
- Goto(x, y): 从 x 处走到 y 处.
- Setdown(x): 在 x 处放下盒子.

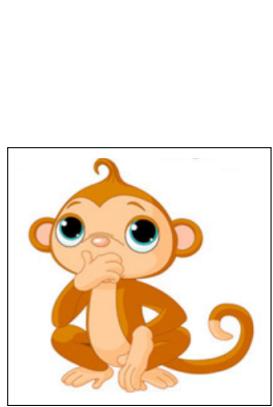
7) 各操作的条件和动作:

- Goto(x, y)
 - 条件: AT(robot, x).
 - 动作: 删除表: AT(robot, x); 添加表: AT(robot, y).
- Pickup(x).
 - 条件: TABLE(x), ON(box, x), AT(robot, x), EMPTY(robot).
 - 动作: 删除表: EMPTY(robot), ON(box, x); 添加表: HOLDS(robot, box).
- Setdown(x)
 - 条件: AT(robot, x), TABLE(x), HOLDS(robot, box).
 - 动作: 删除表: HOLDS(robot, box); 添加表: EMPTY(robot), ON(box, x).

机器人每执行一操作前, 都要检查该操作的先决条件是否可以满足. 如果满足, 就执行相应的操作; 否则再检查下一个操作.

解：机器人行动规划问题的求解过程：

- ▶ 状态 1(初始状态) $AT(robot, c)$, $EMPTY(robot) \xrightarrow{\text{开始}} ON(box, a)$, $TABLE(a)$, $TABLE(b)$
- ▶ 状态 2 $AT(robot, a)$, $EMPTY(robot) \xrightarrow{Goto(c, a)} ON(box, a)$, $TABLE(a)$, $TABLE(b)$
- ▶ 状态 3 $AT(robot, a)$, $HOLDS(robot, box) \xrightarrow{Pickup(a)}$, $TABLE(a)$, $TABLE(b)$
- ▶ 状态 4 $AT(robot, b)$, $HOLDS(robot, box) \xrightarrow{Goto(a, b)}$, $TABLE(a)$, $TABLE(b)$
- ▶ 状态 5 $AT(robot, b)$, $EMPTY(robot) \xrightarrow{Setdown(b)}$, $ON(box, b)$, $TABLE(a)$, $TABLE(b)$
- ▶ 状态 6(目标状态) $AT(robot, c)$, $EMPTY(robot) \xrightarrow{Goto(b, c)}$, $ON(box, b)$, $TABLE(a)$, $TABLE(b)$



a



c



b

图 2-4 猴子摘香蕉问题

例 2.30 (猴子摘香蕉问题, 图2-4) 描述状态的谓词:

- ▶ $AT(x, y)$: x 在 y 处.
- ▶ $ONBOX$: 猴子在箱子上.
- ▶ HB : 猴子得到香蕉.
- ▶ 个体域:
 $x : \{\text{monkey}, \text{box}, \text{banana}\}$
 $Y : \{a, b, c\}$
- ▶ 问题的初始状态
 $AT(\text{monkey}, a)$
 $AT(\text{box}, b)$
 $\neg ONBOX, \neg HB$
- ▶ 问题的目标状态

AT(monkey, c), AT(box, c), ONBOX, HB

描述操作的谓词

- ▶ Goto(u, v): 猴子从 u 处走到 v 处.
- ▶ Pushbox(v, w): 猴子推着箱子从 v 处移到 w 处.
- ▶ Climbbox: 猴子爬上箱子.
- ▶ Grasp: 猴子摘取香蕉.

各操作的条件和动作

- ▶ Goto(u, v)
 - 条件: \neg ONBOX , AT(monkey, u),
 - 动作: 删除表: AT(monkey, u); 添加表: AT(monkey, v)
- ▶ Pushbox(v, w)
 - 条件: \neg ONBOX , AT(monkey, v), AT(box, v)
 - 动作: 删除表: AT(monkey, v), AT(box, v); 添加表: AT(monkey, w), AT(box, w)
- ▶ Climbbox
 - 条件: \neg ONBOX , AT(monkey, w), AT(box, w)
 - 动作: 删除表: \neg ONBOX; 添加表: ONBOX
- ▶ Grasp
 - 条件: ONBOX, AT(box, c)
 - 动作: 删除表: \neg HB; 添加表: HB



注 2.31. 一阶逻辑中, 量词只能用于个体变元, 取消这一限制条件, 允许量词也可用于命题变元和谓词变元, 由此构造起来的谓词逻辑就是高阶逻辑. 二阶逻辑是一阶逻辑的扩展, 一阶逻辑是命题逻辑的扩展. 公理化的高阶逻辑系统或高阶逻辑的自然推理系统又称为广义谓词演算或高阶谓词演算. 彭漪涟, 马钦荣, 《逻辑学大辞典》, 2010 年上海辞书出版社.

2.2.2 谓词逻辑表示的特征

谓词逻辑表示的主要优点

- ▶ 自然: 一阶谓词逻辑是一种接近于自然语言的形式语言系统, 谓词逻辑表示法接近于人们对问题的直观理解.
- ▶ 明确: 是一种标准的知识解释方法, 因此用这种方法表示的知识明确且易于理解.
- ▶ 精确: 谓词逻辑的真值只有“真”与“假”, 其表示、推理都是精确的.
- ▶ 灵活: 知识和处理知识的程序是分开的, 无须考虑处理知识的细节.

- ▶ 模块化: 知识之间相对独立, 模块化的处理方式使得添加、删除和修改知识比较容易进行.

性质 2.5 谓词逻辑的性质

- ▶ 自反性: $(\forall x)R(x, x)$
- ▶ 非自反性: $(\forall x)\neg R(x, x)$
- ▶ 传递性: $(\forall x)(\forall y)(\forall z)[(R(x, y) \wedge R(y, z)) \rightarrow R(x, z)]$
- ▶ 对称性: $(\forall x)(\forall y)(R(x, y) \rightarrow R(y, x))$
- ▶ 非对称性 y: $(\forall x)(\forall y)(R(x, y) \rightarrow \neg R(y, x))$
- ▶ 反对称性: $(\forall x)(\forall y)[(R(x, y) \wedge R(y, x)) \rightarrow x = y]$



主要缺点

- ▶ 知识表示能力差: 只能表示确定性知识, 而不能表示非确定性知识、过程性知识和启发式知识.
- ▶ 知识库管理困难: 缺乏知识的组织原则, 知识库管理比较困难.
- ▶ 存在组合爆炸: 由于难以表示启发式知识, 因此只能盲目地使用推理规则, 这样当系统知识量较大时, 容易发生组合爆炸.
- ▶ 系统效率低: 它把推理演算与知识含义截然分开, 抛弃了表达内容中所含有的语义信息, 往往使推理过程冗长, 降低了系统效率.

只考虑约束个体词的量词, 这种谓词逻辑就是一阶谓词逻辑. 带有两种量词的谓词逻辑, 就是二阶逻辑. 自然语言的符号和自然语言之间的区别是语言学的研究对象, 不是逻辑学的研究对象. 在有些数学分支中比如拓扑学中, 需要二阶逻辑的能力来做完整的表达. 这方面的工作已经由 Stephen G. Simpson 在逆数学的名义下完成了. 已经证明了二阶逻辑不只对表达经典数学的某些重要部分是必须的, 而且它也可以用做模型论和数学基础的工具.

例 2.2.1

二阶句子

$$\forall S \forall x (x \in S \vee x \notin S), \quad (2.4)$$



对于所有个体的集合 S 和所有的个体 x , 要么 x 在 S 中要么不在 (这是二值原理). 最一般的二阶逻辑还包括量化在函数上的变量和解说变量.

二阶逻辑比一阶逻辑更有表达力。在二阶逻辑中，有可能写出声称“论域是有限的”或“论域有可数势的”这样的形式句子。要说论域是有限的，可使用声称从论域到自身的所有单射函数都是满射的句子。要声称论域有可数势，可以使用声称在所有的论域的两个无限子集之间存在双射的句子。从向上[勒文海姆-斯科伦定理](#)得出在一阶逻辑内不可能特征化有限性或可数性。

例 2.2.2

在一阶逻辑中断言每个实数都有一个加法逆元： $\forall x \exists y (x + y = 0)$, $x, y \in \mathbb{R}$, 需要二阶逻辑来断言实数的集合的上确界性质，它声称实数的所有有界的、非空集合都有上确界。如果论域是所有实数的集合，表达了最小上界性质的二阶逻辑句子：

$$\begin{aligned} \forall A & \left[(\exists w (w \in A) \wedge \exists z \forall w (w \in A \rightarrow w \leq z)) \right. \\ & \left. \rightarrow \exists x \forall y ([\forall w (w \in A \rightarrow w \leq y)] \leftrightarrow x \leq y) \right] \quad (2.5) \end{aligned}$$

2.2.3 知识表示——产生式表示法

产生式 (Production) 是目前人工智能中使用最多的一种知识表示方法。

2.2.3.1 事实的表示

定义 2.32 事实

事实是断言一个语言变量或多个语言变量之间关系的陈述句。



注 2.33. 语言变量的值或语言变量之间的关系可以是数字和词等。

例 2.34 “雪是白的”，其中“雪”是语言变量，“白的”是语言变量的值。“王峰热爱祖国”，其中，“王峰”和“祖国”是两个语言变量，“热爱”是语言变量之间的关系。

例 2.35 确定性知识，事实可用如下三元组表示：

(对象, 属性, 值) 或 (关系, 对象 1, 对象 2)

其中，对象就是语言变量。

(snow, color, white) 或 (雪, 颜色, 白)
 (love, Wang Feng, country) 或 (热爱, 王峰, 祖国)

- 不确定性知识, 事实可用如下四元组表示:

(对象, 属性, 值, 可信度因子)

其中“可信度因子”是指该事实为真的相信程度. 可用 [0,1] 之间的一个实数来表示.

2.2.4 规则的表示

规则的作用是描述事物之间的因果关系.

定义 2.36 产生式规则

规则的产生式表示形式常称为产生式规则, 简称为产生式或规则.



产生式的基本形式

$P \rightarrow Q$ 或者 IF P THEN Q ,

其中, P 是产生式的前提, 也称为前件, 它给出了该产生式可否使用的先决条件, 由事实的逻辑组合来构成; Q 是一组结论或操作, 也称为产生式的后件, 它指出当前题 P 满足时, 应该推出的结论或应该执行的动作.

产生式的含义

如果前提 P 满足, 则可推出结论 Q 或执行 Q 所规定的操作.

产生式规则的例子

r_6 : IF 动物有犬齿 AND 有爪 AND 眼盯前方 THEN 该动物是食肉动物,

其中, r_6 是该产生式的编号; “动物有犬齿 AND 有爪 AND 眼盯前方”是产生式的前提 P ; “该动物是食肉动物”是产生式的结论 Q .

产生式与蕴涵式、条件语句的区别

► 与蕴涵式的主要区别:

(1) 蕴涵式表示的知识只能是精确的, 产生式表示的知识可以是不确定的.

原因是蕴涵式是一个逻辑表达式, 其逻辑值只有真和假.

(2) 蕴含式的匹配一定要求是精确的, 而产生式的匹配可以是不确定的.

原因是产生式的前提条件和结论都可以是不确定的, 因此其匹配也可以是不确定的.

► 与条件语句的主要区别:

(1) 前件结构不同

- 产生式的前件可以是一个复杂的结构.

- 传统程序设计语言中的左部仅仅是一个布尔表达式.

(2) 控制流程不同

- 产生式系统中满足前提条件的规则被激活后, 不一定被立即执行, 规则能否执行将取决于冲突消解策略.

 **注 2.37.** 传统程序设计语言是严格地从一个条件语句向下一个条件语句传递.

2.2.5 产生式系统的基本结构——系统结构及说明

综合数据库 DB(Data Base) 存放求解问题的各种当前信息: 问题的初始状态、输入的事实、中间结论及最终结论等. 还能用于推理过程的规则匹配, 推理过程中, 当规则库中某条规则的前提可以和综合数据库的已知事实匹配时, 该规则被激活, 由它推出的结论将被作为新的事实放入综合数据库, 成为后面推理的已知事实.

定义 2.38 规则库 RB(Rule Base)

也称知识库 KB(Knowledge Base), 用于存放与求解问题有关的所有规则的全体.



► 作用: 是产生式系统问题求解的基础.

► 要求: 知识的完整性、一致性、准确性、灵活性和知识组织的合理性.

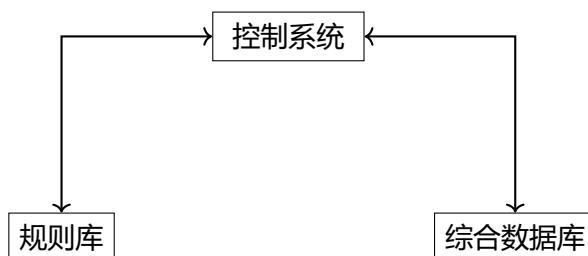


图 2-5 产生式系统的基本结构

控制系统 (Control system)

控制系统 亦称推理机, 主要作用是控制整个产生式系统的运行, 决定问题求解过程的推理线路.

控制系统的主要任务

- ▶ **选择匹配:** 按一定策略从规则库中选择规则与综合数据库中的已知事实进行匹配. 匹配是指把所选规则的前提与综合数据库中的已知事实进行比较, 若事实库中存的事实与所选规则前提一致, 则称匹配成功, 该规则可用; 否则, 称匹配失败, 该规则不可用.
- ▶ **冲突消解:** 对匹配成功的规则, 按照某种策略从中选出一条规则执行.
- ▶ **执行操作:** 对所执行的规则, 若其后件为一个或多个结论, 则把这些结论加入综合数据库; 若其后件为一个或多个操作时, 执行这些操作.
- ▶ **终止推理:** 检查综合数据库中是否包含有目标, 若有, 则停止推理.
- ▶ **路径解释:** 在问题求解过程中, 记住应用过的规则序列, 以便最终能够给出问题的解路径.

例 2.39 动物识别系统: 该系统可以识别老虎、金钱豹、斑马、长颈鹿、企鹅和信天翁这 6 种动物. 其规则库包含如下 15 条规则:

- ▶ r_1 IF 该动物有毛发 THEN 该动物是哺乳动物
- ▶ r_2 IF 该动物有奶 THEN 该动物是哺乳动物
- ▶ r_3 IF 该动物有羽毛 THEN 该动物是鸟
- ▶ r_4 IF 该动物会飞 AND 会下蛋 THEN 该动物是鸟
- ▶ r_5 IF 该动物吃肉 THEN 该动物是食肉动物
- ▶ r_6 IF 该动物有犬齿 AND 有爪 AND 眼睛前方 THEN 该动物是食肉动物
- ▶ r_7 IF 该动物是哺乳动物 AND 有蹄 THEN 该动物是有蹄类动物
- ▶ r_8 IF 该动物是哺乳动物 AND 是食肉动物 THEN 该动物是有蹄类动物
- ▶ r_9 IF 该动物是哺乳动物 AND 是食肉动物 AND 是黄色 AND 身上有暗斑点 THEN 该动物是金钱豹
- ▶ r_{10} IF 该动物是哺乳动物 AND 是食肉动物 AND 是黄色 AND 身上有黑色条纹 THEN 该动物是虎
- ▶ r_{11} IF 该动物是有蹄类动物 AND 有长脖子 AND 有长腿 AND 身上有暗斑点 THEN 该动物是长颈鹿
- ▶ r_{12} IF 动物是有蹄类动物 AND 身上有黑色条纹 THEN 该动物是斑马

- r_{13} IF 该动物是鸟 AND 有长脖子 AND 有长腿 AND 不会飞 AND 有黑白二色 THEN 该动物是鸵鸟
 - r_{14} IF 该动物是鸟 AND 会游泳 AND 不会飞 AND 有黑白二色 THEN 该动物是企鹅
 - r_{15} IF 该动物是鸟 AND 善飞 THEN 该动物是信天翁,
其中, $r_i(i = 1, 2, \dots, 15)$ 是规则的编号.
- 初始综合数据库包含的事实有: 动物有暗斑点, 有长脖子, 有长腿, 有奶, 有蹄.

系统的推理过程

(1) 先从规则库中取出第一条规则 r_1 , 检查其前提是否可与综合数据库中的已知事实相匹配. r_1 的前提是“有毛发”, 但事实库中无此事实, 故匹配失败. 然后取 r_2 , 该前提可与已知事实“有奶”相匹配, r_2 被执行, 并将其结论“该动物是哺乳动物”作为新的事实加入到综合数据库中. 此时, 综合数据库的内容变为:

动物有暗斑, 有长脖子, 有长腿, 有奶, 有蹄, 是哺乳动物.

(2) 再从规则库中取 r_3, r_4, r_5, r_6 进行匹配, 均失败. 接着取 r_7 , 该前提与已知事实“是哺乳动物”相匹配, r_7 被执行, 并将其结论“该动物是有蹄类动物”作为新的事实加入到综合数据库中. 此时, 综合数据库的内容变为:

动物有暗斑, 有长脖子, 有长腿, 有奶, 有蹄, 是哺乳动物, 是有蹄类动物.

(3) 此后, r_8, r_9, r_{10} 均匹配失败. 接着取 r_{11} , 该前提“该动物是有蹄类动物 AND 有长脖子 AND 有长腿 AND 身上有暗斑”与已知事实相匹配, r_{11} 被执行, 并推出“**该动物是长颈鹿**”. 由于“长颈鹿”已是目标集合中的一个结论, 即已推出最终结果, 故问题求解过程结束.

该例子的部分推理网络如图2-6, 图2-6中最上层的结点称为“假设”或“结论”. 中间节点称为“中间假设”; 终结点称为“证据”或“事实”; 每个“结论”都是本问题的一个目标, 所有“假设”构成了本问题的目标集合.

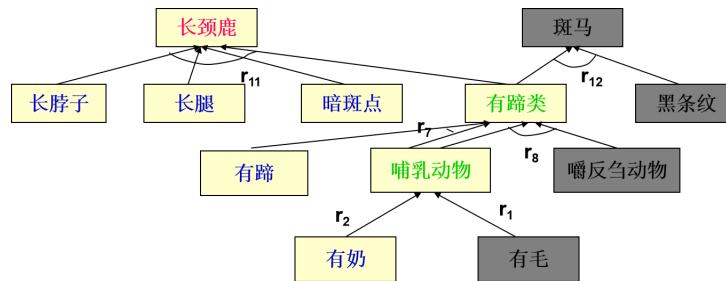


图 2-6 产生式系统的推理结果

 **注 2.40.** 上述规则仅是一种直接表示方式, 用三元组表示 r_{15} 如下:

r_{15} : IF (动物, 类别, 鸟) AND(动物, 本领, 善飞) THEN (动物, 名称, 信天翁).

2.2.6 产生式系统的过程——基本过程

- (1) 初始化综合数据库, 把待解决问题的已知事实整理并送入综合数据库中;
- (2) 检查规则库中是否有未使用过的规则, 若无, 转 (7);
- (3) 检查规则库中未使用规则是否有其前提可与综合数据库中已知事实相匹配的规则, 若有, 形成当前可用规则集; 否则, 转 (6);
- (4) 按照冲突消解策略, 从当前可用规则集中选择一个规则执行, 并对该规则作上标记. 把执行该规则后所得到的结论作为新的事实放入综合数据库; 如果该规则的结论是一些操作, 则执行这些操作;
- (5) 检查综合数据库中是否包含了该问题的解, 若已包含, 说明解已求出, 问题求解过程结束; 否则, 转 (2);
- (6) 当规则库中还有未使用规则, 但均不能与综合数据库中的已有事实相匹配时, 要求用户进一步提供关于该问题的已知事实, 若能提供, 则转 (2); 否则, 执行下一步;
- (7) 若知识库中不再有未使用规则, 也说明该问题无解, 终止问题求解过程.

 **注 2.41.** 从第 (3) 步到第 (5) 步的循环过程实际上就是一个搜索过程.

总体上可分为以下两种方式

1) 不可撤回方式

- 是一种“一直往前走”不回头的方式, 类似于中国象棋中过河的卒子. 它即根据当前已知的局部知识选取一条规则作用于当前综合数据库, 接着再根据新状态继续选取规则, 如此进行下去, 不考虑撤回用过的规则. 不理想规则的应用会降低效率, 但不影响可解性.

优点是控制过程简单, 缺点是当问题有多个解时不一定能找到最优解.

2) 试探性方式, 可分为以下两种方式:

- 回溯方式是一种碰壁回头的方式. 即在问题求解过程中, 允许先试一试某条规则, 如果以后发现这条规则不合适, 则允许退回去, 再另选一条规则来试.

需要解决的主要问题: 一是如何确定回溯条件, 二是如何减少回溯次数. 是一种完备而有效的策略, 它容易实现且占内存容量较小.

► 图搜索方式

图搜索方式是一种用图或树把全部求解过程记录下来的方式. 由于它记录了已试过的所有路径, 因此便于从中选取最优路径.

 **注 2.42.** 主要区别: 回溯方式抹去了所有引起失败的试探路径, 而图搜索方式则记住了已试过的所有路径.

按推理方向

- ▶ 正向推理产生式系统, 也称数据驱动方式, 它是从初始状态出发, 朝着目标状态前进, 正向使用规则的一种推理方法.

 **注 2.43.** 所谓正向使用规则, 是指以问题的初始状态作为初始综合数据库, 仅当综合数据库中的事实满足某条规则的前提时, 该规则才被使用.

- 优点: 简单明了, 且能求出所有解.
- 缺点: 执行效率较低, 原因是使用规则具有一定的盲目性.
- ▶ 逆向推理产生式系统, 也称目标驱动方式, 它是从目标(作为假设)状态出发, 朝着初始状态前进, 逆向使用规则的一种推理方法.

 **注 2.44.** 所谓逆向使用规则, 是指以问题的目标状态作为初始综合数据库, 仅当综合数据库中的事实满足某条规则的后件时, 该规则才被使用.

- 优点: 不使用与问题无关的规则. 因此, 对那些目标明确的问题, 使用逆向推理方式是一种最佳选择.
- ▶ 双向推理产生式系统, 双向推理是把正向推理和反向推理结合起来使用的一种推理方式. 它需要把问题的初始状态和目标状态合并到一起构成综合数据库.

按规则库的性质及结构

- ▶ 可交换的产生式系统是一种对规则的使用次序无关的产生式系统, 即任意交换规则的使用次序都不会影响对问题的求解.

假设 DB 是综合数据库, RB 是规则库, $DB_i(i = 1, 2, \dots)$ 是第 i 次使用规则后得到的新的综合数据库, RS 是一个可作用于 DB_i 的规则集合. 若一个产生式系统可交换, 则其 RB 和每一个 DB_i 都应具有如下性质:

① 对任一规则 $r_j = RS(j = 1, 2, \dots)$, 它作用于 DB_i 得到新的综合数据库 DB_{i+1} , RS 仍然是 DB_{i+1} 的可用规则集.

② 如果 DB_i 满足目标条件, 则用 RS 中的任一规则 r_j 作用于 DB_i , 得到的 DB_{i+1} 仍然满足目标条件.

③ 若对 DB_i 使用某一规则序列 r_1, r_2, \dots, r_k 得到一个新的综合数据库 DB_k , 则当改变这些规则的使用次序后, 仍然可得到 DB_k .

可见,综合数据库是递增的,即对任何序列 r_1, r_2, \dots, r_g , 其作用于 DB 后所得到的 DB_1, DB_2, \dots, DB_g 之间存在如下关系:

$$DB_1 \subset DB_2 \subset \dots \subset DB_g.$$

 **注 2.45.** 这说明在可交换产生式系统中, 其规则的结论部分总是包含着新的内容, 一旦执行该规则就会把这些新的内容添加到综合数据库中.

例 2.46 设给定一个整数集合 $\{a, b, c\}$, 可通过把集合中任意一对元素的乘积作为新元素添加到集合中的办法来扩大该整数集, 要求通过若干次操作后能生成所需的整数集合.

综合数据库 DB 可用集合来表示

- 初始状态为 $\{a, b, c\}$;
- 目标状态为 $a, b, c, a \times b, b \times c, a \times c$;
- 规则库 RB 中包含的规则有:

$r_1: \text{IF } \{a, b, c\} \text{ THEN } \{a, b, c, a \times b\}$
 $r_2: \text{IF } \{a, b, c\} \text{ THEN } \{a, b, c, b \times c\}$
 $r_3: \text{IF } \{a, b, c\} \text{ THEN } \{a, b, c, a \times c\}$

显然,无论先使用哪一条规则都可由初始状态达到目标状态. 因此,上述由 DB 和 RB 所构造的产生式系统是一个可交换的产生式系统, 并具有可交换产生式系统三个性质.

可交换产生式系统的可交换性, 使得其求解过程只需要搜索其中的任意一条路径, 就能达到目标, 而不必进行回溯. 这种系统的求解过程可采用不可撤回的控制方式.

► 可分解的产生式系统

把一个整体问题分解成若干个子问题, 然后再通过对这些子问题的求解来得到整个问题解的一种产生式系统.

例 2.47 设综合数据库的初始状态为 $\{C, B, Z\}$, 目标状态为 $\{M, M, \dots, M\}$, 规则库中有如下重写规则:

- $r_1: C \rightarrow \{D, L\}$.
- $r_2: C \rightarrow \{B, M\}$.
- $r_3: B \rightarrow \{M, M\}$.
- $r_4: Z \rightarrow \{B, B, M\}$.

解决该问题时, 可先把初始综合数据库分为三个子库, 然后对这三个子库分别应用规则库中的相应规则进行求解. 其求解过程如图2-7所示.

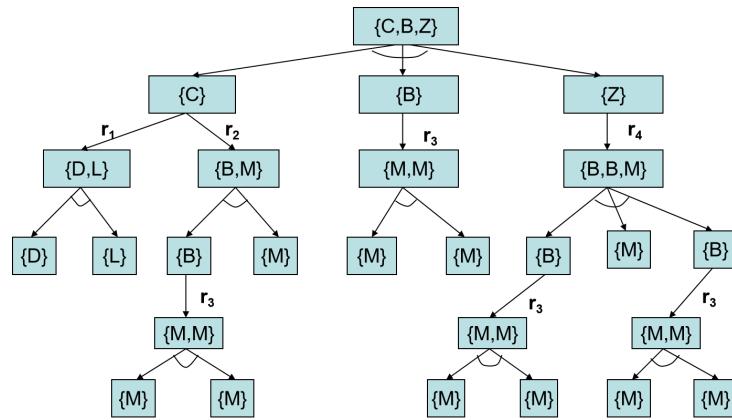


图 2-7 按规则库的性质及结构

可恢复的产生式系统

- ▶ 是指那种采用回溯控制方式的产生式系统.
- ▶ 其求解问题的方法是: 当执行某条规则后, 如果发现所得到的新的综合数据库不可能求出问题的解, 就立即撤消由该规则所产生的结果, 使综合数据库恢复到先前的状态, 然后再另选别的规则继续求解.
- ▶ 它既可以向综合数据库中添加新的内容, 又可以从综合数据库中删除或修改老的内容. 这种求解问题的方法, 更符合人们的一般习惯.
- ▶ 对可恢复的产生式系统, 也将在第四章详细讨论.

2.2.7 产生式系统的特点

主要优点

- ▶ 自然性: 采用“如果 ……, 则 ……”的形式, 人类的判断性知识基本一致.
- ▶ 模块性: 规则库中最基本的知识单元, 各规则之间只能通过综合数据库发生联系, 而不能相互调用, 从而增加了规则的模块化程度.
- ▶ 有效性: 产生式知识表示法既可以表示确定性知识, 又可以表示不确定性知识, 既有利于表示启发性知识, 又有利于表示过程性知识.
- ▶ 一致性: 规则库中的所有规则都具有相同的格式, 并且综合数据库可被所有规则访问, 因此规则库中的规则可以统一处理.

主要缺点

- ▶ 效率较低: 各规则之间的联系必须以综合数据库为媒介. 求解过程是一种反复进行的“匹配—冲突消解—执行”过程. 这样的执行方式将导致执行的效率低.
- ▶ 不便于表示结构性知识: 由于产生式表示中的知识具有一致格式, 且规则之间不能相互调用, 因此那种具有结构关系或层次关系的知识则很难以自然的方式来表示.

2.3 语义网络表示法

语义网络是奎廉 (J. R. Quillian) 1968 年在研究人类联想记忆时提出的一种心理学模型, 认为记忆是由概念间的联系实现的. 随后, 奎廉又把它用作知识表示. 1972 年, 西蒙在他的自然语言理解系统中也采用了语义网络表示法. 1975 年, 亨德里克 (G. G. Hendrix) 又对全称量词的表示提出了语义网络分区技术.

2.3.1 语义网络

2.3.1.1 语义网络

定义 2.48 语义网络

语义网络是一种用实体及其语义关系来表达知识的有向图. 结点代表实体, 表示各种事物、概念、情况、属性、状态、事件和动作等; 弧代表语义关系, 表示它所连结的两个实体之间的语义联系, 它必须带有标识.

定义 2.49 语义基元

语义网络中最基本的语义单元称为语义基元, 可用三元组表示为:

(结点 1, 弧, 结点 2)

基本网元 指一个语义基元对应的有向图.

例 2.50 若有语义基元 (A, R, B) , 其中, A, B 分别表示两个结点, R 表示 A 与 B 之间的某种语义联系, 对应的基本网元如图 2-8 所示:



图 2-8 语义联系的基本网元

语义网络的简单例子

例 2.51 用一网络表示“鸵鸟是一种鸟”。



图 2-9 语义联系的基本网元

语义网络与产生式对应的表示能力.

事实的表示

例 2.52 “雪的颜色是白的”。



图 2-10 语义联系的基本网元

规则的表示

例 2.53 规则 R 的含义是“如果 A 则 B”。

基本的语义关系 主要分 8 种关系, 包括实例关系、分类关系、成员关系、属性关系、聚类关系、时间关系、位置关系和相近关系.

1) 实例关系: ISA 体现的是“具体与抽象”的概念, 含义为“是一个”, 表示一个事物是另一个事物的一个实例.

例 2.54 ISA 例子



图 2-11 实例关系: ISA

2) 分类关系: AKO 分类关系亦称泛化关系, 体现的是“子类与超类”的概念, 含义为“是一种”, 表示一个事物是另一个事物的一种类型.

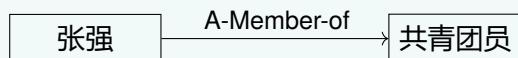
例 2.55 AKO 例子



图 2-12 实例关系: AKO

3) 成员关系: A-Member-of 体现的是“个体与集体”的关系, 含义为“是一员”, 表示一个事物是另一个事物的一个成员.

例 2.56 AMO 例子



 **注 2.57.** 上述关系的主要特征是属性的继承性, 处在具体层的结点可以继承抽象层结点的所有属性.

4) 属性关系 指事物和其属性之间的关系. 常用的属性关系有:

- ▶ Have: 含义为“有”, 表示一个结点具有另一个结点所描述的属性.
- ▶ Can: 含义为“能”、“会”, 表示一个结点能做另一个结点的事情.

例 2.58 “鸟有翅膀”



Age: 含义为“年龄”，表示一个结点是另一个结点在年龄方面的属性.

例 2.59 张强 18 岁”



5) 聚类关系 亦称包含关系. 指具有组织或结构特征的“部分与整体”之间的关系. 常用的包含关系是: Part-of : 含义为“是一部分”，表示一个事物是另一个事物的一部分.

例 2.60 “大脑是人体的一部分”



例 2.61 “黑板是墙体的一部分”



注 2.62. 聚类关系与实例、分类和成员关系的主要区别: 聚类关系一般不具备属性的继承性. 如上述两个例子, 大脑不一定具有人的各种属性. 黑板也不具有墙的各种属性.

6) 时间关系 指不同事件在其发生时间方面的先后次序关系. 常用的时间关系有:

- Before: 含义为“在前”，表示一个事件在另一个事件之前发生.
- After: 含义为“在后”，表示一个事件在另一个事件之后发生.

例 2.63 例如：“北京奥运会在悉尼奥运会之后”.



7) 位置关系 指不同事物在位置方面的关系. 常用的位置关系有:

- ▶ Located-on: 含义为“在上”，表示某一物体在另一物体之上；
- ▶ Located-at: 含义为“在”，表示某一物体所在的位置；
- ▶ Located-under: 含义为“在下”，表示某一物体在另一物体之下；
- ▶ Located-inside: 含义为“在内”，表示某一物体在另一物体之内；
- ▶ Located-outside: 含义为“在外”，表示某一物体在另一物体之外.

例 2.64 “书在桌子上”.



8) 相近关系 指不同事物在形状、内容等方面相似或接近. 常用的相近关系有:

- ▶ Similar-to: 含义为“相似”，表示某一事物与另一事物相似.
- ▶ Near-to: 含义为“接近”，表示某一事物与另一事物接近.

例 2.65 “猫似虎”.



2.3.2 表示一元关系

一元关系 指可以用一元谓词 $P(x)$ 表示的关系. 谓词 P 说明实体的性质和属性. 描述的是一些最简单、最直观的事物或概念, 常用：“是”、“有”、“会”和“能”等语义关系来说明.

例 2.66 “雪是白的”.

一元关系的描述 应该说, 语义网络表示的是二元关系. 如何用它来描述一元关系? 具体做法为: 结点 1 表示实体, 结点 2 表示实体的性质或属性等, 弧表示语义关系.

例 2.67 “李刚是一个人”为一元关系, 其语义网络如前所示(图2-11).

例 2.68 用语义网络表示“动物能运动、会吃”.



2.3.3 二元关系表示

单个二元关系可直接用一个基本网元来表示, 如前介绍的一些常用的二元关系及其表示.

注 2.69. 二元关系是可用二元谓词 $P(x, y)$ 表示的关系, 其中, x, y 为实体, P 为实体之间的关系. 对复杂关系, 可通过一些相对独立的二元或一元关系的组合来实现.

例 2.70 用语义网络表示:

动物能运动、会吃.
鸟是一种动物, 鸟有翅膀、会飞.
鱼是一种动物, 鱼生活在水中、会游泳.

对于这个问题, 按属性关系描述各种动物, 动物之间的分类关系用类属关系描述.



图 2-13 例2.70 的语义网络

例 2.71 用语义网络表示：

王强是理想公司的经理；

理想公司在中关村；

王强 28 岁。



图 2-14 例2.71 的语义网络

例 2.72 李新的汽车的款式是银灰色“捷达”。王红的汽车的款式是红色“凯越”。李新和王红的汽车均属于具体概念，可增加“汽车”这个抽象概念。图2-15给出了本例的语义网络。

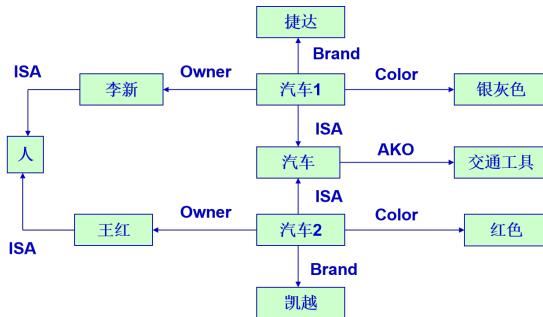


图 2-15 例2.72 的语义网络

2.3.4 表示多元关系

定义 2.73 多元关系

可用多元谓词 $P(x_1, x_2, \dots)$ 表示的关系，其中，个体 x_1, x_2, \dots 为实体，谓词 P 说明这些实体之间的关系。



多元关系的表示方法：用语义网络表示多元关系时，可把它转化为一个或多个二元关系的组合，然后再利用下一节讨论的合取关系的表示方法，把这种多元关系表示出来。具体来说：增加情况和动作结点（西蒙）。

例 2.74 用语义网络表示：“小燕子这只燕子从春天到秋天占有一个巢”。

需要设立一个占有权结点，表示占有物或者占有时间等概念。



图 2-16 例2.74 的语义网络

对上述问题，也可以把占有作为一种关系，并用一条弧来表示，但在这种表示方法下，

占有时间关系就无法表示了.



图 2-17 "小燕子这只燕子有一个巢" 的语义网络

动作结点的表示 用语义网络表示事件或动作时, 需要设立一个事件或动作结点.

动作结点: 由一些向外引出的弧来指出动作的主体与客体.

例 2.75 用于语义网络表示: “常河给江涛一张磁盘作为礼物” .



图 2-18 例2.75的语义网络

事件结点: 如上例用一个事件结点描述.

表示方法: 可通过增加合取结点和析取结点来实现.

例 2.76 用语义网络表示如下事实: “参赛者有教师、有学生; 身材有高、有低” .

首先需要分析参赛者的不同情况, 可得到以下四种情况:

- | | |
|---------|---------|
| A 教师、高； | B 教师、低. |
| C 学生、高； | D 学生、低. |

然后再按照他们的逻辑关系用语义网络表示出来.



图 2-19 例2.76的语义网络

语义关系的否定表示 分为基本语义关系的否定和一般语义关系的否定.

① 基本语义关系的否定表示可通过在有向弧上直接标注该基本语义关系的否定这一方法来解决.

例 2.77 用语义网络表示: 书不在桌子上.

采用在有向弧上直接标注该基本语义关系的否定结点的方法, 语义网络如图2-20.

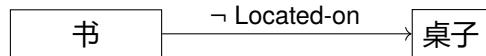


图 2-20 语义网络的基本网元

② 一般语义关系的否定表示对一般语义关系的否定, 通常需要引进“非”节点来表示.

例 2.78 用语义网络表示: 常河没有给江涛一张磁盘.

采用引进“非”节点的方法, 其语义网络如图2-21.

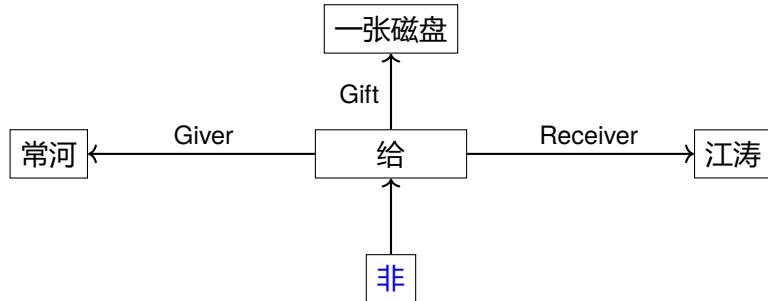


图 2-21 例2.78的语义网络

蕴含的表示 通过增加蕴含关系节点来实现. 在蕴含关系中, 有两条指向蕴含节点的弧, 一条代表前提条件, 标记为 ANTE; 另一条代表结论, 标记为 CONSE.

例 2.79 用语义网络表示如下知识: “如果学校组织大学生机器人竞赛活动, 那么李强就参加比赛” .

该蕴含关系的语义网络如图2-22, 其中机器人竞赛的组织者是学校, 参赛对象是学生操纵的机器人, 机器可被归属为一种智能机器.



图 2-22 例2.79的语义网络

2.3.5 存在和全称量词的表示

存在量词: 可直接用“ISA”和“AKO”等这样的语义关系来表示.

全称量词: 可采用亨德里克提出的网络分区技术.

基本思想: 把一个复杂命题划分为若干个子命题, 每个子命题用一个较简单的语义网络表示, 称为一个子空间, 多个子空间构成一个大空间. 每个子空间看作是大空间中的一个结点, 称作超结点. 空间可逐层嵌套, 子空间之间用弧互相连接.

例 2.80 用语义网络表示如下事实: “每个学生都学习了一门程序设计语言”.

其语义网络如图2-23,

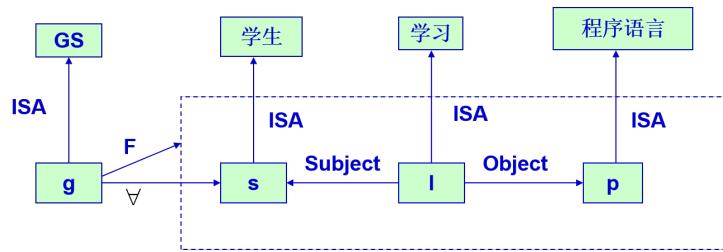


图 2-23 例2.80的语义网络

GS 是一个概念结点, 它表示具有全称量化的一般事件. *g* 是一个实例结点, 代表 *GS* 中的一个具体例子, 如上所提到的事实. *s* 是一个全称变量, 表示任意一个学生. *l* 是一个存在变量, 表示某一次学习. *p* 是一个存在变量, 表示某一门程序设计语言. 这样, *s,l,p* 之间的语义联系就构成一个子空间, 它表示对每一个学生 *s*, 都存在一个学习事件 *l* 和一门程序设计语言 *p*.

在从结点 *g* 引出的三条弧中, 弧“ISA”说明结点 *g* 是 *GS* 中一个实例; 弧“F”说明它

所代表的子空间及其具体形式; 弧“ \forall ”说明它所代表的全称量词. 每一个全称量词都需要一条这样的弧, 子空间中有多少个全称量词, 就需要有多少条这样的弧.

例 2.81 用语义网络表示事实: “每个学生都学习了所有的程序设计课程”.

其语义网络如图2-24所示. 其中, 结点 g 有两条指向全称变量的弧.

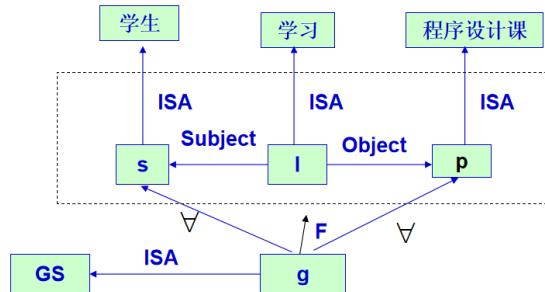


图 2-24 例2.81的语义网络

注 2.82. 在网络分区技术中, 要求 F 指向的子空间中的所有非全称变量结点都应该是存在量词约束的变量, 否则应放在子空间的外面.

例 2.83 用语义网络表示事实: “每个学生都学习了 C++ 语言”.

其语义网络如图2-25所示. 结点“C++ 语言”代表一门具体的程序设计语言, 是结点“程序语言”的一个实例, 故被放到弧 F 所指的子空间的外边.

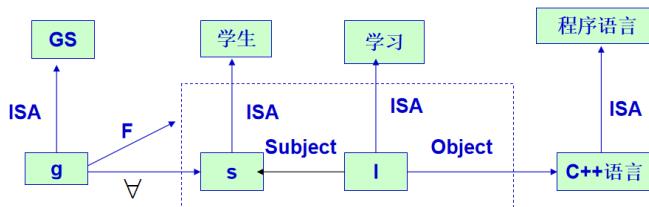


图 2-25 例2.83的语义网络

2.3.6 语义网络的推理过程

用语义网络表示知识的问题求解系统主要由两大部分所组成, 一部分是由语义网络构成的知识库, 另一部分是用于问题求解的推理机构. 语义网络的推理过程主要有两种, 一种是继承, 另一种是匹配.

2.3.7 继承

定义 2.84 继承

指把对事物的描述从抽象结点传递到实例结点. 通过继承可以得到所需结点的一些属性值, 它通常是沿着 ISA 和 AKO 等继承弧进行的.



继承的一般过程 继承是在结点上用弧表示彼此的连接、从属和属性关系, 从上一结点得到需要的属性.

(1) 建立一个结点表, 用来存放待求解结点和所有以 ISA 和 AKO 等继承弧与此结点相连的那些结点. 初始情况下, 表中只有待求解结点.

(2) 检查表中的第一个结点是否有继承弧. 如果有, 就把该弧所指的所有结点放入结点表的末尾, 记录这些结点的所有属性, 并从结点表中删除第一个结点. 如果没有继承弧, 仅从结点表中删除第一个结点.

(3) 重复 (2), 直到结点表为空. 此时, 记录下来的所有属性都是待求解结点继承来的属性.

例 2.85 在图2-13所示的语义网络中, 通过继承关系可以得到“鸟”具有: 会吃和能运动的属性.

2.3.8 匹配

定义 2.86 匹配

指在知识库的语义网络中寻找与待求解问题相符的语义网络模式.



匹配的主要过程:

(1) 根据待求解问题的要求构造一个网络片断, 该网络片断中有些结点或弧的标识是空的, 称为询问处, 它反映的是待求解的问题.

(2) 根据该语义片断到知识库中去寻找所需要的信息.

(3) 当待求解问题的网络片断与知识库中的某语义网络片断相匹配时, 则与询问处相匹配的事实就是问题的解.

例 2.87 假设例2-14的语义网络已在知识库中, 问王强在哪个公司工作.

根据这个问题的要求, 可构造如下语义网络片断.



图 2-26 例2.87的语义网络

当用该语义网络片段与图2-14所示的语义网络进行匹配时,由“工作在”弧所指的结点可知,职员王强工作在“理想公司”,这就得到了问题的答案。若还想知道职员王强的其它情况,则可在语义网络中增加相应的空结点。

思考

用人工智能常见的编程语言给出例2.87的一个实现。

2.3.9 语义网络表示法的优缺点:

- ▶ **结构性** 把事物的属性以及事物间的各种语义联系显式地表示出来,是一种结构化的知识表示方法。在这种方法中,下层结点可以继承、新增和变异上层结点的属性。
- ▶ **联想性** 作为人类联想记忆模型提出来的,它着重强调事物间的语义联系,体现了人类的联想思维过程。
- ▶ **自索引性** 把各结点之间的联系以明确、简洁的方式表示出来,通过与某一结点连结的弧可以很容易的找出与该结点有关的信息,而不必查找整个知识库。这种自索引能力有效地避免搜索中遇到的组合爆炸问题。
- ▶ **自然性** 这种带有标识的有向图,可比较直观地把知识表示出来,符合人们表达事物间关系的习惯,并且与自然语言语义网络之间的转换也比较容易实现。

主要缺点:

- ▶ **非严格性** 没有象谓词那样严格的形式表示体系,一个给定语义网络的含义完全依赖于处理程序对它所进行的解释,通过语义网络所实现的推理不能保证其正确性。
- ▶ **复杂性** 语义网络表示知识的手段是多种多样的,这虽然给其表示带来了相当程度的灵活性,但同时也由于表示形式的不一致,给处理问题增加了复杂性。

2.4 框架表示法

定义 2.88 框架表示法

在框架理论的基础上发展起来的一种结构化知识表示方法.



2.4.1 框架理论

框架理论是明斯基于 1975 年作为理解视觉、自然语言对话及由复杂行为基础而产生的智能行为等提出来的一种方法。它认为人们对现实世界中各种事物的认识都是以一种类似于框架的结构存储在记忆中的，当遇到一个新事物时，就从记忆中找出一个合适的框架，并根据新的情况对其细节加以修改和补充，从而形成对这个新事物的认识。

- ▶ **框架**: 是人们认识事物的一种通用的数据结构形式。即当新情况发生时，人们只要把新的数据加入到该通用数据结构中，便可形成一个具体的实体（类），这样的通用数据结构就称为框架。
- ▶ **实例框架**: 对于一个框架，当人们把观察或认识到的具体细节填入后，就得到了该框架的一个具体实例，框架的这种具体实例被称为实例框架。
- ▶ **框架系统**: 在框架理论中，框架是知识的基本单位，把一组有关的框架连结起来便可形成一个框架系统。
- ▶ **框架系统推理**: 由框架之间的协调来完成。

框架之间的相互联系主要表现在框架的层次结构，即通过 ISA 链，各个框架之间特殊与一般的继承关系得以表达；框架之间的横向联系还可以用框架中的槽值表示。

2.4.2 框架结构和框架表示

例 2.89 一个直接描述硕士生有关情况的框架。

```
Frame <MASTER>
  Name: Unit(Last-name, First-name)
  Sex: Area(male, female)
    Default: male
  Age: Unit(Years)
  Major: Unit(Major)
  Field: Unit(Field)
  Advisor: Unit(Last-name, First-name)
```

Project: Area(National, Provincial, Other)

 Default: National

Paper: Area(SCI, EI, Core, General)

 Default: Core

Address: <S-Address>

Telephone: Home Unit(Number)

Mobile: Unit(Number)

框架的基本结构

< 框架名 >

槽名 1: 侧面名 11 值 111, 值 112,

 侧面名 12 值 121, 值 122,

 :

槽名 2: 侧面名 21 值 211, 值 212,

 侧面名 22 值 221, 值 222,

 :

 :

 :

槽名 n: 侧面名 n1 值 n11, 值 n12,

 侧面名 n2 值 n21, 值 n22,

 :

 侧面名 nm 值 nm1, 值 nm2,

学生框架

Frame <Student>

 Name: Unit(Last-name, First-name)

 Sex: Area(male, female)

 Default: male

 Age: Unit(Years)

 If-Needed: Ask-Age

 Address: <S-Address>

 Telephone: Home Unit(Number)

```

Mobile Unit(Number)
If-Needed: Ask-Telephone

```

硕士生框架

```

Frame <Master>
AKO: Student
Major: Unit(Major)
    If-Needed: Ask-Major
    If-Added: Check-Major
Field: Unit(Direction-Name)
    If-Needed: Ask-Field
Advisor: Unit>Last-name, First-name)
    If-Needed: Ask-Visor
Project: Area(National, Provincial, Other)
    Default: National
Paper: Area(SCI, EI, Core, General)
    Default: Core

```

在 Master 框架中, 用到了一个系统预定义槽名 AKO. 所谓系统预定义槽名, 是指框架表示法中事先定义好的可公用的一些标准槽名. AKO 与其在语义网络中的含义相似, 其直观含义为“是一种”. 当 AKO 作为下层框架的槽名时, 其槽值为上层框架的框架名, 一般来说, 下层框架所描述的事物比其上层框架更具体. 并且, 由 AKO 所联系的框架之间具有属性的继承关系.

框架的继承技术 通常由框架中设置的 3 个侧面: Default、If-Needed 和 If-Added 所提供的缺省推理功能来组合实现.

Default: 该侧面的作用是为相应槽提供缺省值. 当其所在槽没有填入槽值时, 系统就以此侧面值作为该槽的默认值.

例 2.90 Paper 槽的默认值为 Core.

If- Needed: 该侧面的作用是提供一个为相应槽赋值的过程. 当某个槽不能提供统一的缺省值时, 可在该槽增加一个 If-Needed 侧面, 系统通过调用该侧面提供的过程, 产生相应的属性值.

例 2.91 Age 槽和 Telephone 槽等.

If-Added: 该侧面的作用是提供一个因相应槽值变化而引起的后继处理过程. 当某个槽的槽值变化会影响到一些相关槽时, 需要在该槽增加一个 If-Added 侧面, 系统通过调用该侧面提供的过程去完成对其相关槽的后继处理.

例 2.92 Major 槽, 由于专业的变化, 可能会引起 Field 和 Advisor 的变化, 因此需要调用 If-Added 侧面提供的 Check-Major 过程进行后继处理.

例 2.93 给出宾馆、饭店和智慧教室的框架结构.

实例框架

例 2.94 假设有杨叶和柳青 2 个硕士生, 当把他们的具体情况分别添入 Master 框架后, 可得实例框架 Master-1 和 Master-2. 硕士生-1 实例框架:

```
Frame <Master-1>
  ISA:  Master
  Name: Yang Ye
  Sex: female
  Major: Computer
  Field: Web-Intelligence
  Advisor: Lin Hai
  Project: Provincial
```

硕士生-2 实例框架:

```
Frame <Master-2>
  ISA:  Master
  Name: Liu Qing
  Age: 22
  Major: Computer
  Advisor: Lin Hai
  Paper: EI
```

在这 2 个实例框架中, 我们又用到了一个系统预定义槽名 ISA. 该预定义槽名与语义网络中的 ISA 弧的语义相似, 其直观含义为“是一个”, 表示一个事物是另一个事物的一

一个具体实例,用来描述一个具体事物与其抽象概念间的实例关系.

 **注 2.95.** *Master-1* 和 *Master-2* 是 2 个具体的 *Master*.

框架系统的基本结构 当知识比较复杂时,往往需要通过诸框架之间的横向或纵向联系形成一种框架系统.

1) 框架之间的纵向联系 是指那种具有继承关系的上下层框架之间的联系.

例 2.96 在图2-27中,学生可按照接受教育的层次分为本科生、硕士生和博士生.每类学生又可按照所学专业的不同,分为不同专业的学生等.

框架之间的纵向联系是通过预定义槽名 AKO 和 ISA 等来实现的.

例 2.97 前面的例子,AKO 实现了 Student 框架与 Master 框架之间的纵向联系,ISA 实现了 Master 框架与 Master-1 实例框架之间的联系.

2) 框架之间的横向联系 是指那种以另外一个框架名作为一个槽的槽值或侧面值所建立起来的框架之间的联系.如下图给出的框架系统中,Student 框架与 S-Address 框架之间就是一种横向联系(图2-27).

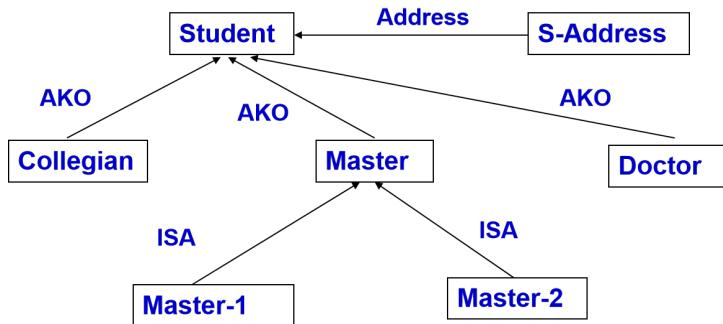


图 2-27 框架系统的基本结构

框架问题求解的基本过程 主要是通过对框架的继承、匹配与填槽来实现的.求解问题分为如下三步:

- 1) 把该问题用框架表示出来.
- 2) 利用框架之间的继承关系,把它与知识库中的已有框架进行匹配,找出一个或多个候选框架,并在这些候选框架引导下进一步获取附加信息,填充尽量多的槽值,以建立一个描述当前情况的实例.

3) 再用某种评价方法对候选框架进行评价, 以决定是否接收该框架.

特性继承过程

- ▶ 特性继承主要是通过 ISA 和 AKO 链来实现的. 当需要查询某一事物的某个属性, 且描述该事物的框架未提供其属性值时, 系统就沿 ISA 和 AKO 链追溯到具有相同槽的类或超类框架.
- ▶ 如果该槽提供有 Default 侧面值, 就继承该默认值作为查询结果返回.
- ▶ 如果该槽提供有 If-Needed 侧面供继承, 则执行 If-Needed 操作, 去产生一个值作为查询结果.
- ▶ 如果对某个事物的某一属性进行了赋值或修改操作, 则系统会自动沿 ISA 和 AKO 链追溯到具有相应的类或超类框架, 只要发现类或超类框架中的同名槽具有 If-Added 侧面, 就执行 If-Added 操作, 作相应的后继处理.

If-Needed 与 If-Added 过程的区别 它们的主要区别在于激活时机和操作目的不同.

- ▶ If-Needed 操作是在系统试图查询某个实物框架中未赋值的属性值时才激活, 并根据查询需求, 被动地即时产生所需要的属性值;
- ▶ If-Added 操作是在系统对某个事物的框架属性作赋值或修改工作后才能被激活, 目的在于通过规定的后继处理, 主动做好配套操作, 以消除数据库中可能存在的不一致问题.

例 2.98 以前面的学生框架为例

- ▶ 若要查询 Master-1 的 Sex, 则可直接回答; 但要查询 Master-2 的 Sex, 则需要沿 ISA 链和 AKO 链到 Student 框架取其默认值 male.
- ▶ 若要查询 Master-2 的 Field, 需要沿 ISA 链到 Master 框架, 执行 Field 槽 If-Needed 侧面的 Ask-Field 操作, 即时产生一个值, 假设产生的值是 Data-Mining, 则表示 Master-2 的研究方向为数据挖掘.
- ▶ 如果要修改 Master-2 的 Major, 需要沿 ISA 链到 Master 框架, 执行 Major 槽 If-Added 侧面的 Check-Major 操作, 对 Field、Advisor 进行修改, 以保持知识的一致性.

匹配和填槽 框架的匹配实际上是通过对相应槽的槽名和槽值逐个进行比较, 并利用继承关系来实现的.

例 2.99 假设前面讨论的学生框架系统已建立在知识库中, 若要求从知识库中找出一个满足如下条件的硕士生:

male, Age<25>, Major 为 Computer, Project 为 National

把这些条件用框架表示出来, 就可得到如下的初始问题框架:

```
Frame: <Master-x>
  Name:      \textcolor[rgb]{0,0,1}{?}
  Age:      Years <25>
  Sex:      male
  Major:    Computer
  Projec:   National
```

用此框架和知识库中的框架匹配, 显然 “Master -2” 框架可以匹配. 因为 Age、Sex 和 Major 槽都符合要求, Project 槽虽然没有给出, 但由继承性可知它取默认值 National, 完全符合初始问题框架 Master-x 的要求, 所以要找的学生有可能是 Liu Qing.

2.4.3 框架表示法的优缺点

框架表示法的优点

- ▶ 结构性: 最突出特点是善于表示结构性知识, 它能够把知识的内部结构关系以及知识间的特殊联系表示出来.
- ▶ 深层性: 框架表示法不仅可以从多个方面和多重属性表示知识, 而且还可以通过 ISA、AKO 等槽以嵌套结构分层的方式对知识进行表示, 因此能用来表达事物间复杂的深层联系.
- ▶ 继承性: 在框架系统中, 下层框架可以继承上层框架的槽值, 也可以进行补充和修改, 这样既减少知识冗余, 又较好地保证了知识的一致性.
- ▶ 自然性: 框架能把与某个实体或实体集相关的特性都集中在一起, 从而高度模拟了人脑对实体多方面和多层次存储结构的需求, 直观自然, 易于理解.

框架表示法的不足

- ▶ 框架缺乏形式理论: 至今, 还没有建立框架的形式理论, 其推理和一致性检查机制并非基于良好定义的语义.
- ▶ 缺乏过程性知识表示: 框架系统不便于表示过程性知识, 缺乏如何使用框架中知

识的描述能力。框架推理过程需要用到一些与领域无关的推理规则，而这些规则在框架系统中又很难表达。

- ▶ 清晰性难以保证：由于各框架本身的数据结构不一定相同，从而框架系统的清晰性很难保证。

2.5 面向对象的知识表示(知识+方法)

认为世界由各种“对象”组成，每个对象类都定义了所谓“方法”(method)，它们实际上可视为允许作用于该类对象上的各种操作。面向对象知识表示方法与框架表示方法有许多相似之处，如层次分类和特性继承机制等。但由于应用目标不同，实现和使用方式有较大区别。框架表示法旨在支持知识的陈述性表示，强调事物的结构化描述和对人思维方式的模拟。面向对象表示法则强调信息的结构化处理，注重信息和信息处理的封装和程序设计的模块化。

2.6 过程表示法

过程性知识表示是将有关某一问题领域的知识，连同如何使用这些知识的方法，均隐式地表示为一个求解问题的过程。

过程表示没有固定的表示形式，如何描述知识完全取决于具体问题。以过程规则表示形式为例，一个过程规则由以下4部分组成：

- (1) 激发条件：激发条件由推理方向和调用模式两部分组成。
 - ▶ 推理方向用于指出推理是正向推理(FR)还是反向推理(BR)。
 - 正向推理的激活条件：当综合数据库中的已有事实可以与其“调用模式”匹配时才能将该过程规则激活；
 - 反向推理的激活条件：当“调用模式”与查询目标或子目标匹配时才能将该过程规则激活。
- (2) 演绎操作：演绎操作由一系列的子目标构成。当前面的激发条件满足时，将执行这里列出的演绎操作。
- (3) 状态转换：状态转换槽的作用是来完成对综合数据库的增、删和改操作。
- (4) 返回：过程规则的最后一个语句是返回语句，用于指出将控制权返回到调用该过程规则的上一级过程规则那里去。

例 2.100 下面给出一个关于同学问题的过程表示。

设有如下知识：

“如果 x 与 y 是同班同学, 且 z 是 x 的老师, 则 z 也是 y 的老师”.

其 5 个过程规则可表示为:

- ① BR(Teacher, ?z, ?y) (需求解的问题)
- ② GOAL(Classmate, ?x, y) (求 y 的同班同学是谁)
- ③ GOAL(Teacher, z, x) (已知 z 是 x 的老师)
- ④ INSERT(Teacher, z, y) (对数据库的插入操作)
- ⑤ RETURN

其中: BR 是逆向推理标志; GOAL 表示求解子目标, 即进行过程调用; INSERT 表示对数据库进行插入操作; RETURN 作为结束标志;



注 2.101. 带 “?” 的变量 (如?x) 表示该变量的值需在计算过程中求得.

问题求解基本过程的简单描述:

每当有一个新的目标时, 就从可以匹配的过程规则中选择一个执行. 在该规则的执行过程中可能会产生新的目标, 此时就调用相应的过程规则并执行它. 反复进行这一过程, 直至执行到 RETURN 语句.

只要碰到 RETURN 语句, 就将调用返回当前过程的上一级过程规则, 并依次逐级返回. 如果某过程规则运行失败, 就另选择一个同层的可匹配过程规则执行, 如果不存在这样的过程规则, 则返回失败标志, 并将控制权移交给上一级过程规则.

例 2.102 设综合数据库中有以下已知事实:

(Classmate, 杨叶, 柳青)
(Teacher, 林海, 杨叶)

已知杨叶与柳青是同班同学, 林海是杨叶的老师. 假设需要求解的问题是: 找出两个人 w 及 v , 其中 w 是 v 的老师.

解: 该问题可表示为: GOAL(Teacher, ?w, ?v).

(1) 在过程规则库中找出问题 GOAL(Teacher, ?w, ?v) 的激发条件可以满足的过程规则. 显然, BR(Teacher, z, y) 经变量代换 $w/z, v/y$ 后可以匹配, 因此选用该过程规则.

(2) 执行该过程规则中的第一个语句 GOAL(Classmate, ?x, y). 此时, 其中的 y 已被 v 代换. 经与已知事实 (Classmate, 杨叶, 柳青) 匹配, 分别求得了变量 x 及 v 的值, 即

$$x = \text{杨叶}, v/y = \text{柳青}$$

(3) 执行该过程规则中的第二个语句 GOAL(Teacher, z, x). 此时, x 的值已经知道, z

已被 w 代换, 结果为 $\text{GOAL}(\text{Teacher}, w, \text{杨叶})$. 经与已知事实 $(\text{Teacher}, \text{林海}, \text{杨叶})$ 匹配, 求得了变量 w 的值, 即

$$w/z = \text{林海}$$

(4) 执行该过程规则中的第三个语句 $\text{INSERT}(\text{Teacher}, z, y)$, 此时, z 与 y 的值均已经知道, 分别是林海和杨叶, 因此这时插入数据库的事实是:

$$(\text{Teacher}, \text{林海}, \text{杨叶})$$

这表明“林海也是杨叶的老师”, 求得了问题的解.

2.6.1 过程表示的主要优缺点

主要优点:

- ▶ 表示效率高.
 - 过程表示法是用程序来表示知识的, 而程序能准确的表明先做什么, 后做什么以及怎样做, 并直接嵌入一些启发式的控制信息, 可以避免选择及匹配那些无关的知识, 也不需要跟踪那些不必要的路径, 从而提高了系统的运行效率.
- ▶ 控制系统容易实现.
 - 由于控制性质已嵌入到程序中, 因而控制系统就比较容易设计.

主要缺点: 不易修改及添加新知识, 而且当对某一过程进行修改时, 又可能影响到其它过程, 对系统的维护带来不便.

2.7 模糊认知图

定义 2.103 模糊认知图

模糊认知图 (Fuzzy Cognitive Map, FCM, 1986) 是 Kosko 融合 Zadeh 的模糊集理论和 Axelrod(阿克塞尔罗德) 的认知图理论提出的, 将概念间的三值关系 $\{-1, 0, 1\}$ 扩展成为区间 $[-1, 1]$ 上的模糊关系发展而来.

FCM 的概念值和弧的权重值都可以为模糊值, 是一种软计算方法, 它是模糊逻辑和神经网络结合的产物, FCM 的知识表示和推理能力更强. FCM 是有向图, 将模糊反馈动力系统中的概念及概念间的关系通过弧线连接起来, 强调“结构就是含义”.

FCM 描述了系统概念集和概念间的因果关系, 它的具体定义如下

- ▶ FCM 可以表示为一个四元组 $G = (C, E, U, f)$, 其中 $C = (C_1, C_2, \dots, C_c)$ 表示构成有向图的顶点的概念集,
- ▶ $E : (C_i, C_j) \rightarrow w_{ij}, (C_i, C_j) \in E$ 表示概念节点 C_i 到 C_j 的有向边的权重, 则所有节点构成的权重矩阵表示为 W .
- ▶ $U : C_i \rightarrow u_i$ 表示概念节点 C_i 到激活度 u_i 的映射, 则 $U(t) = (u_1(t), u_2(t), \dots, u_c(t))$ 表示当前 t 时刻所有概念节点的激活度, 也就是 G 的在 t 时刻的状态, 其中 $u_i(t) \in [0, 1]$.
- ▶ 压缩函数 (squashing function) 用 f 表示, 表示概念节点的激活度在 t 时刻到 $t + 1$ 时刻的转换函数, 即

$$u_i(t+1) = f \left(\sum_{j=1}^c w_{ij} u_j(t) \right), \quad (2.6)$$

其中 f 用 sigmoid 函数进行表示, 以此将激活度映射到 $[0, 1]$ 的区间中: $f(u) = \frac{1}{1+e^{-\tau u}}$, sigmoid 函数中的 τ 表示一个陡峭参数 (steepness parameter), 其中, τ 越大, sigmoid 函数形状越接近阶跃函数. 通常情况下 $\tau = 5$.

FCM 是将模糊反馈动力系统中的因果事件、参与值、目标与趋势等概念通过概念间的有向弧线连接起来的图, 节点是概念和实体等, 弧表示概念或实体间的因果关系, 在结构上可以看作是面向对象的单层带反馈的神经网络.

 **注 2.104.** 模糊认知图的每个节点与弧都有很强的语义, 整个图都具有语义性. 它支持专家先验知识及因果关系的表示与推理, 这些都蕴涵在概念节点及概念节点间的关系中, 并且可以通过概念间的关系来表示模糊推理, 由整个图中各概念节点的相互作用来模拟系统的动态行为, 是一种无监督模型.

 **注 2.105.** 认知概念可以是系统的事件、目标、感情及趋势等; 概念值为模糊值, 反映该节点对某概念以某种程度发生; 也可以是二值, 表示概念状态是关还是开. 概念节点的输出与概念节点自身的状态水平和外部因果联系的强度有关.

 **注 2.106.** 模糊认知图的有限输入状态可在虚拟空间中开辟一条通路, 简单模糊认知图的通路可能终止于固定点或极限环, 而对于具有反馈的复杂模糊认知图可能终止于“混沌”的奇异吸引子.

2.7.1 模糊认知图的参数训练

通常概念节点会通过模糊 C 均值聚类得到的中心表示, 这也是信息粒化的过程, 之后, 对 FCM 的训练即是转换为一个对其权重矩阵 W 的优化问题. 权值的优化一般是通

过启发式算法,例如粒子群算法和基于梯度的算法. 损失函数通过训练数据进行优化,一般选取为误差平方和

$$\text{SSE} = \sum_{j=1}^N \sum_{i=1}^c (u_{ij} - t_{ij}), \quad (2.7)$$

即最小化期望输出与实际输出之间的差异.

2.8 小结

知识是将有内在相关性的信息关联在一起形成的信息结构,具有相对正确性、不确定性、可表示性和可利用性等特点. 对知识的表示可以分为符号表示法和连接机制表示法. 知识表示法都是面向符号的知识表示方法. 在这些表示方法中,谓词逻辑、产生式系统和状态空间表示法属于非结构化的知识表示范畴,语义网络、框架、面向对象和脚本技术属于结构化的知识表示范畴. 这些表示方法适用于不同的问题,各有长处.

2.9 作业

思考

请用语义网络表示如下知识: 高老师从3月到7月给计算机系的学生讲“计算机网络”课.

思考

用语义网络表示“军用飞机有轰炸机和战斗机; 民用飞机有客机和货运机”.

观察

用谓词逻辑知识表示方法表示如下知识:

- (1) 有人喜欢梅花, 有人喜欢菊花, 有人既喜欢梅花又喜欢菊花.
- (2) 不是每个计算机系的学生都喜欢在计算机上编程序.

思考

人工智能对知识表示有什么要求?

思考

用一阶谓词逻辑表示下面的句子: a) 并不是所有的学生选修了历史和生物. b) 历史考试中只有一个学生不及格. c) 只有一个学生历史和生物考试都不及格. d) 历史考试的最高分比生物考试的最高分要高.

思考

语义网络示例如图2-28, 给出其描述.

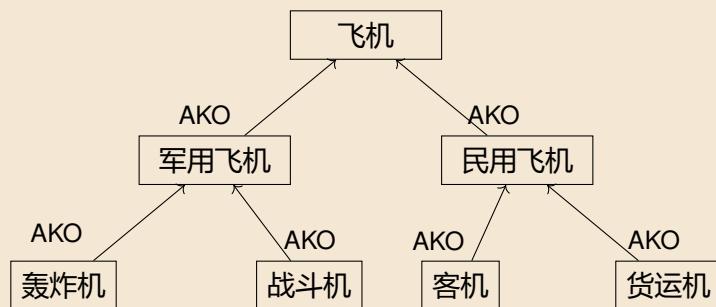


图 2-28 语义网络示例

思考

产生式表示法中事实和规则怎么表示? 用产生式表示法设计一个医学知识库.

思考

以一所研究性大学人员状况为例, 说明语义网络的基本描述格式.

思考

框架表示法有什么特点? 试构造一个描述你的卧室的框架系统.

思考

八数码问题也称为九宫问题. 在 3×3 的棋盘, 摆有八个棋子, 每个棋子上标有 1~8 的某一数字, 不同棋子上标的数字不相同. 棋盘上还有一个空格, 与空格相邻的棋子可以移到空格中. 要求解决的问题是, 给出一个初始状态和一个目标状态, 找出一种从初始状态转变成目标状态的移动序列.

探索

基于 web 的动物识别系统

1. 实验目的

理解和掌握产生式知识表示方法及产生式系统的基本过程, 能够利用 Web 编程技术建立一个基于产生式知识表示的简单的智能系统.

2. 实验环境

(1) 硬件环境: 网络环境中的微型计算机.

(2) 软件环境: Windows 操作系统, 任选一种网络编程语言和数据库管理系统.

3. 实验要求

(1) 以本书第 2 章动物识别产生式系统的规则为知识库(也可增加规则), 采用正向推理或逆向推理方式.

(2) 以选定的数据库管理系统建立知识库, 用选定的网络编程语言按 B/S 模式开发一个具有解释功能的智能系统.

(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.

3

知识推理方法

智能系统的推理过程

智能系统的推理过程实际上就是一种思维过程。按照推理过程所用知识的确定性，推理可分为确定性推理和不确定性推理。



过去三十年，人们对知识的表示与推理 (KRR) 和机器学习 (ML) 领域地一些常见的

问题进行了深入而广泛地探究,如知识表示的类型、知识和数据的作用、信息的缺乏和过剩、解释和理解的需要 [BouraouiKR2019] 等问题.

3.1 什么是推理

定义 3.1 推理

是指按照某种策略,从已知事实出发去推导出结论的过程.



3.1.1 推理所用的事实

- ▶ **初始证据:** 在推理前用户提供的详实佐证材料.
- ▶ **中间结论:** 在推理过程中所得到的结果.
- ▶ **推理过程:** 主要由推理机来完成, 所谓推理机就是智能系统中用来实现推理的程序.

例 3.2 对于医疗专家系统, 专家知识保存在知识库中. 推理开始时, 先把病人的症状和检查结果放到综合数据库中, 然后再从综合数据库的初始证据出发, 按照某种策略在知识库中寻找, 并使用知识, 直到推出最终结论为止.

推理的两个基本问题(方法和策略):

- ▶ **推理的方法:** 表达前提和结论的逻辑关系.
- ▶ **推理的控制策略:** 确定推理方向, 消解冲突策略.

3.1.2 按推理的逻辑基础分类

演绎推理

定义 3.3 演绎推理

演绎推理是从已有知识出发, 去推出蕴含在这些已知知识中的适合于某种个别情况的结论. 是一种由一般到个别的推理方法, 其核心是**三段论: 假言推理、拒取式和假言三段论**.



例 3.4 假言三段论

$$A \rightarrow B, B \rightarrow C \Rightarrow A \rightarrow C.$$

常用的三段论由三部分组成:一个大前提、一个小前提和一个结论. 其中, 大前提是已知的一般性知识或推理过程得到的判断; 小前提是关于某种具体情况或某个具体实例的判断.

 **注 3.5.** 结论是由大前提推出的, 并且适合于小前提的判断.

例 3.6 有如下三个判断是一个三段论推理:

- ① 计算机系的学生都会编程序; (一般性知识).
- ② 程强是计算机系的一位学生; (具体情况).
- ③ 程强会编程序. (结论).

其中, ①是大前提; ②是小前提; ③是经演绎推出来的结论. 可见, 推出的结论是蕴含在大前提中, 是大前提的一个特例.

3.1.2.1 归纳推理

定义 3.7 归纳推理

归纳推理是一种由个别到一般的推理方法.



例 3.8 归纳推理的类型

- ▶ 按照所选事例的广泛性可分为完全归纳推理和不完全归纳推理.
- ▶ 按照推理所使用的方法可分为枚举、类比、统计和差异归纳推理等.

定义 3.9 完全归纳推理

是指在进行归纳时需要考察相应事物的全部对象, 并根据这些对象是否都具有某种属性, 推出该类事物是否具有此属性.



定义 3.10 不完全归纳推理

指在进行归纳时只考察了事物的部分对象, 就得出了关于该事物的结论.



定义 3.11 枚举归纳推理

指在进行归纳时, 如果已知某类事物中的有限个具体事物都具有某种属性, 则可推出该类事物都具有此种属性.



例 3.12 设有如下事例:

王强是计算机系学生, 他会编程序;
高华是计算机系学生, 她会编程序;
.....

当这些具体事例足够多时, 就可归纳出一个一般性的知识: 凡是计算机系的学生, 就一定会编程序.

定义 3.13 类比归纳推理

指在两个或两类事物有许多属性都相同或相似的基础上, 推理出它们在其他属性上也相同或相似的一种归纳推理方法.

设 A 、 B 分别是两类事物的集合:

$$A = \{a_1, a_2, \dots\},$$

$$B = \{b_1, b_2, \dots\}.$$

并设 a_i 与 b_i 总是成对出现, 且当 a_i 有属性 P 时, b_i 就有属性 Q 与此对应, 即

$$P(a_i) \rightarrow Q(b_i), i = 1, 2, \dots.$$

则当 A 与 B 中有一新的元素对出现时, 若已知 a' 有属性 P , b' 有属性 Q , 即

$$P(a') \rightarrow Q(b').$$

例 3.14 使用 SARS 的相关经验去处理武汉发生的新型冠状病毒肺炎.

 **注 3.15.** 类比归纳推理的基础是相似原理, 其可靠程度取决于两个或两类事物的相似程度, 或者这两类事物的相同属性与新属性之间的相关程度.

3.1.2.2 演绎推理与归纳推理的区别

演绎推理是在已知领域内的一般性知识的前提下, 通过演绎求解一个具体问题或者证明一个结论的正确性. 它所得出的结论实际上早已蕴含在一般性知识的前提中, 演绎推理只不过是将已有事实揭露出来, 因此它不能增殖新知识.

归纳推理所推出的结论是没有包含在前提内容中的. 这种由个别事物或现象推出一般性知识的过程, 是增殖新知识的过程.

演绎推理是一种以严格的经典逻辑为基础的推理 (MIT, Winston 教授), 而不确定性推理本质上是一种非演绎推理, 虽然演绎推理也是一种人类的智能活动, 而人工智能中的推理主要是指**不确定性逻辑推理**, 也称之为**常识推理**.

- ▶ 演绎推理使用的是抽象而严格的定义, 它在一定理论框架下是绝对正确的定理和公式. 而常识推理使用的是具有局部与暂时合理性的知识和超知识, 即人们的共识和个人的经验.
- ▶ 演绎推理有一定抽象的理论承诺, 它所使用的概念是清晰的, 任何时候都有相同的含义, 因而是确定的. 而常识推理使用的概念是模糊的, 不确定的, 对于不同的人可能会有不同的理解, 具有不确定性, 而知识中的不肯定性表现了人类认识的局限性, 普遍知识中可能有一部分具体对象不正确, 超知识的不肯定性表现在个人的主观性, 特殊知识较之全面知识, 具有片面性.
- ▶ 演绎推理是一种“保真”的推理, 也就是前提正确, 那么由前提经演绎推出的结论也正确, 而常识推理是一种“未必保真”的推理, 即使前提正确, 其结论也未必正确, 是一种近似保真而具有直观合理性的结论.
- ▶ 演绎推理是具有相容性的推理, 在一个理论体系中不可能推出两个同时成立且相反的命题, 而常识推理具有矛盾性, 是一种非协调推理, 在一个知识系统中可能会得到两个相反的结论.
- ▶ 演绎推理是完全的, 具有单调性, 增加新的事例不会影响原有的结论, 而常识推理是不完全的非单调性推理, 增加新的事例可能会影响原有的结论.

总体来说, 演绎推理是在抽象的逻辑结构上进行的, 是一种从普遍性到特殊性的推理,

- 1) 它不承认任何由已知前提推不出的结论,
- 2) 不承认任何不经过演绎推理的假设,
- 3) 不承认任何含有例外事例的结论。

演绎推理是一种让人“放心”, 而偏于“保守”的推理机制. 而常识推理是在实践和经验的基础上进行的, 是一种从特殊性到普遍性的推理, 具有“容错”的特征. 通过不断的发现矛盾与限制矛盾, 修正和维护知识, 其知识的正确性, 既不取决于逻辑的推理规则, 也不取决于以知识为前提的推理过程, 而是取决于推理的结果与事实的符合程度.

演绎推理的真值只能是或“真”或“假”. 常识推理的真值可以是从“真”到“假”之间的任意一个过渡值. 因此, 在常识推理中, 往往会用一种确定度来表示结果的真实程度. 而在正畸思维中, 往往是这样一种常识推理的心理思维活动在其中起决定性作用, 因此, 如何将常识推理, 或说不确定性推理的思维活动采用定量的形式让计算机处理, 将会大大提高数字化人工智能正畸技术的发展.

例 3.16 一位计算机维修员, 从书本知识开始学习, 到通过大量实例积累起来丰富的经验这一过程就是一种归纳推理方式. 运用这些一般性知识去维修计算机的过程则是演绎推理.

3.1.3 推理的控制策略及分类

3.1.3.1 推理的控制策略

- ▶ 推理过程不仅依赖于所用的推理方法, 同时也依赖于推理的控制策略.
- ▶ 推理的控制策略是指给出使用领域知识的方式, 使推理过程尽快达到目标的策略.

控制策略的分类 由于智能系统的推理过程一般表现为一种搜索过程, 因此, 推理的控制策略又可分为**推理策略**和**搜索策略**.

- ▶ 推理策略: 主要解决推理方向和推理冲突等问题, 如推理方向控制策略、求解策略、限制策略和冲突消解策略等.
 - 推理方向控制策略用于确定推理的方向控制, 可分为正向推理、逆向推理、混合推理及双向推理.
 - 求解策略是指仅求一个解, 还是求所有解或最优解等.
 - 限制策略是指对推理的深度、宽度、时间和空间等进行的限制.
 - 冲突消解策略是指当推理过程有多条知识可用时, 如何从这多条可用知识中选出一条最佳知识用于推理的策略.
- ▶ 搜索策略: 主要解决推理线路、推理效果和推理效率等问题.

本章主要讨论推理策略, 至于搜索策略将放到第四章单独讨论.

3.1.4 正向推理

定义 3.17 前向链推理

从已知事实出发、正向使用推理规则, 亦称为**数据驱动推理**或**前向链推理**.

3.1.4.1 正向推理的算法描述

- (1) 把用户提供的初始证据放入综合数据库;
- (2) 检查综合数据库中是否包含了问题的解, 若已包含, 则求解结束, 并成功退出; 否则执行下一步;

(3) 检查知识库中是否有可用知识,若有,形成当前可用知识集,执行下一步;否则转(5).

(4) 按照某种冲突消解策略,从当前可用知识集中选出一条规则进行推理,并将推出的新事实加入综合数据库中,然后转(2).

(5) 询问用户是否可以进一步补充新的事实,若可补充,则将补充的新事实加入综合数据库中,然后转(3);否则表示无解,失败退出.

至于如何根据综合数据库中的事实到知识库中选取可用知识,当知识库中有多条知识可用时,应先使用哪一条知识等.这些问题涉及到了知识的匹配方法和冲突消解策略,以后将会分别讨论. 正向推理求解的算法流程如图3-1所示:

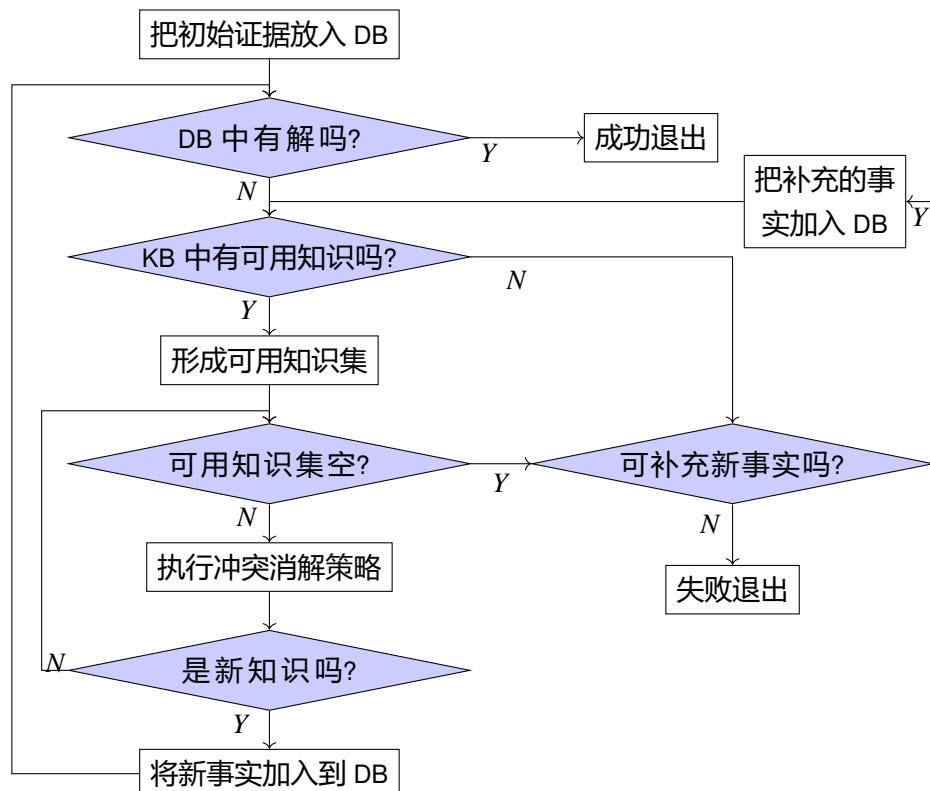


图 3-1 正向推理求解的算法流程

例 3.18 请用正向推理完成以下问题的求解. 假设知识库中包含有以下 2 条规则:

$r_1 : \text{IF } B \text{ THEN } C;$

$r_2 : \text{IF } A \text{ THEN } B.$

|已知初始证据 A , 求证目标 C .

解: 本例的推理过程如下:

- ▶ 推理开始前, 综合数据库为空.
- ▶ 推理开始后, 先把 A 放入综合数据库, 然后检查综合数据库中是否含有该问题的解, 回答为 “ N ” .
- ▶ 接着检查知识库中是否有可用知识, 显然 r_2 可用, 形成仅含 r_2 的知识集. 从该知识集中取出 r_2 , 推出新的实事 B , 将 B 加入综合数据库, 检查综合数据库中是否含有目标 C , 回答为 “ N ” .
- ▶ 再检查知识库中是否有可用知识, 此时由于 B 的加入使得 r_1 为可用, 形成仅含 r_1 的知识集. 从该知识集中取出 r_1 , 推出新的实事 C , 将 C 加入综合数据库, 检查综合数据库中是否含有目标 C , 回答为 “ Y ” .

它说明综合数据库中已经含有问题的解, 推理成功结束, 目标 C 得证.

正向推理的主要优点 推理过程比较直观, 用户可以主动提供有用的事实信息, 适合于诊断、设计、预测和监控等领域的问题求解.

正向推理的主要缺点 推理无明确目标, 求解问题时可能会执行许多与解无关的操作, 导致推理效率较低.

3.1.4.2 逆向推理

从某个假设目标出发, 逆向使用规则, 亦称为**目标驱动推理**或**逆向链推理**. 其算法流程如图3-2:

算法描述

- (1) 将要求证的目标(假设)构成一个假设集;
- (2) 从假设集中选出一个假设, 检查该假设是否存在于综合数据库中, 若在, 则该假设成立, 若此时假设集为空, 则成功退出, 否则仍执行(2); 若该假设不在数据库中, 则执行下一步;
- (3) 检查该假设是否可由知识库的某个知识导出, 若不能由某个知识导出, 则询问用户该假设是否是可由用户证实的原始事实, 若是, 该假设成立, 并将其放入综合数据库, 再重新寻找新的假设, 若不是, 则转(5); 若能由某个知识导出, 则执行下一步;
- (4) 将知识库中可以导出该假设的所有知识构成一个可用知识集;
- (5) 检查可用知识集是否为空, 若是, 失败退出; 否则执行下一步;

- (6) 按冲突消解策略从可用知识集中取出一个知识, 继续执行下一步;
 (7) 将知识前提中的每个子条件都作为新的假设放入假设集, 然后转 (2).

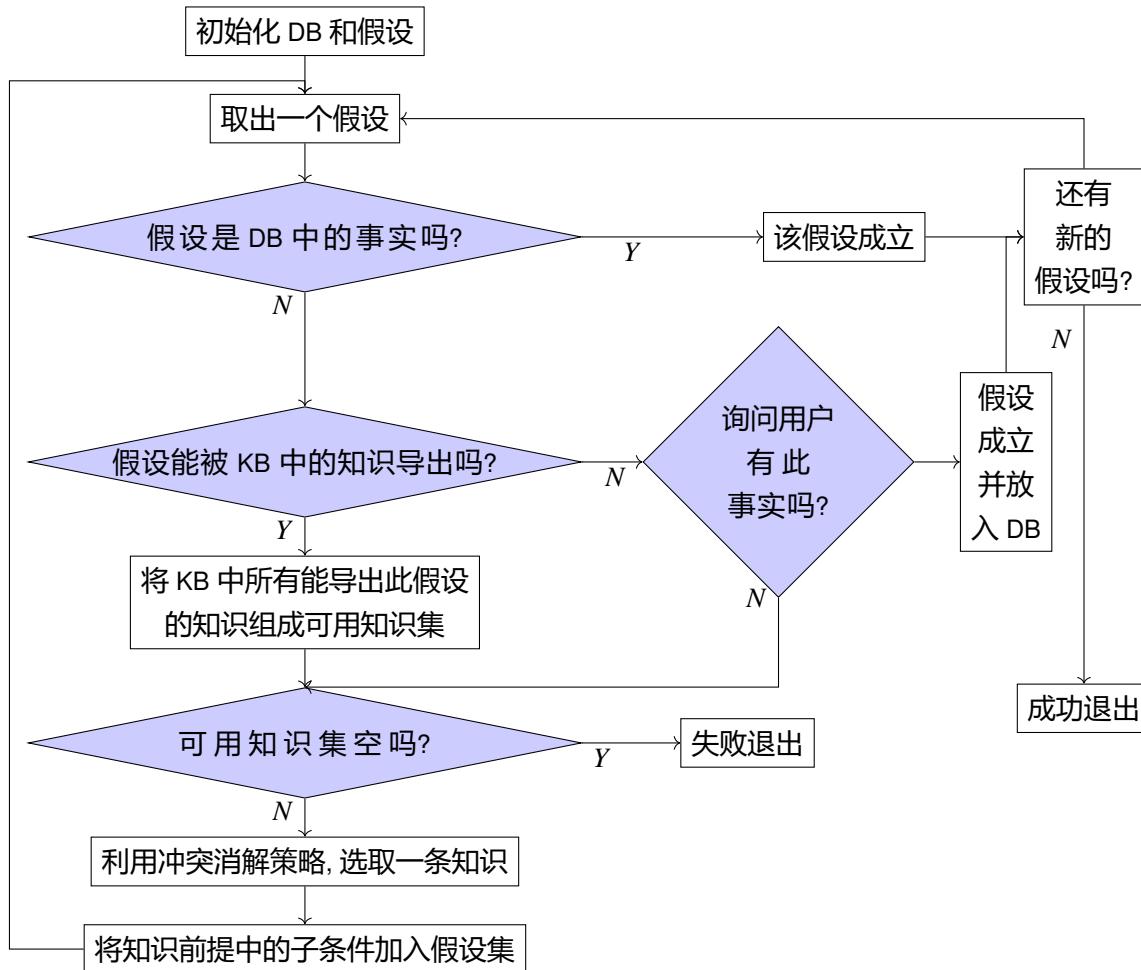


图 3-2 逆向链推理的流程图

对上例, 采用逆向推理, 其推理过程如下:

- ▶ 推理开始前, 综合数据库和假设集均为空.
- ▶ 推理开始后, 先将初始证据 A 放入综合数据库, 目标 C 放入假设集, 然后从假设集中取出一个假设 C , 查找 C 是否为综合数据库中的已知事实, 回答为 “ N ” .
- ▶ 再检查 C 是否能被知识库中的知识所导出, 发现 C 可由 r_1 导出, 于是 r_1 被放入可用知识集. 由于知识库中只有 r_1 可用, 故可用知识集中仅含 r_1 .
- ▶ 接着从可用知识集中取出 r_1 , 将其前提条件 B 作为新的假设放入假设集. 从假设集中取出 B , 检查 B 是否为综合数据库中的事实, 回答为 “ N ” . 再检查 B 是否能被知识库中的知识所导出, 发现 B 可由 r_2 导出, 于是 r_2 被放入可用知识集. 由于

知识库中只有 r_2 可用, 故可用知识集中仅含 r_2 .

- ▶ 从可用知识集中取出 r_2 , 将其前提条件 A 作为新的假设放入假设集. 然后从假设集中取出 A , 检查 A 是否为综合数据库中的事实, 回答为 “Y” .
说明该假设成立, 由于无新的假设, 故推理过程成功结束, 于是目标 C 得证.

优缺点

- ▶ 逆向推理的主要优点:
 - 不必寻找和使用那些与假设目标无关的信息和知识;
 - 推理过程的目标明确;
 - 也有利于向用户提供解释, 在诊断性专家系统中较为有效.
- ▶ 逆向推理的主要缺点:
 - 当用户对推理过程认识不清楚时, 由系统自主选择假设目标, 盲目性比较大, 若选择不好, 可能需要多次提出假设, 会影响系统效率.

3.1.5 混合推理的概念

把正向推理和逆向推理结合起来所进行的推理称为混合推理. 是一种解决较复杂问题的方法.

3.1.5.1 混合推理的方法

- ▶ 先正向后逆向: 这种方法先进行正向推理, 从已知事实出发推出部分结果, 然后再用逆向推理对这些结果进行证实或提高它们的可信度.
- ▶ 先逆向后正向: 这种方法先进行逆向推理, 从假设目标出发推出一些中间假设, 然后再用正向推理对这些中间假设进行证实.
- ▶ 双向混合: 是指正向推理和逆向推理同时进行, 使推理过程在中间的某一步结合起来. 对于这些方法不再详细讨论.

3.2 逻辑学与人工智能

人工智能的产生和发展与逻辑学密不可分. 逻辑学为人工智能的研究提供了根本观点与方法, 而逻辑方法则是人工智能研究中的主要工具. 本节从逻辑学为人工智能的研究提供理论基础出发, 讨论经典逻辑和非经典逻辑在人工智能中的应用, 以及人工智能在逻辑学发展方向上的影响.

人工智能主要研究用计算机方法模拟和扩展人的智能, 最终实现机器智能。人工智能的研究与人的思维研究密切相关。逻辑学始终是人工智能研究中的基础科学问题, 它为人工智能研究提供了根本方法。

3.2.1 人工智能学科的诞生

3.2.1.1 经典数理逻辑的主要理论成果

- 1) 西班牙的罗门·卢乐 (12世纪末 13世纪初) 提出制造可解决各种问题的通用逻辑机。
- 2) 17世纪, 英国的弗朗西斯·培根 (Francis Bacon, 1561-1626年) 在《新工具》中提出了归纳法。随后, 德国莱布尼兹做出了四则运算的手摇计算器, 并提出了“通用符号”和“推理计算”的思想。
- 3) 19世纪, 英国乔治·布尔 (George Boole, 1815-1864) 创立了布尔代数, 奠定了现代形式逻辑研究的基础。德国弗里德里希·路德维希·戈特洛布·弗雷格 (德语: Friedrich Ludwig Gottlob Frege; 1848年11月8日—1925年7月26日) 完善了命题逻辑, 创建了一阶谓词演算系统。
- 4) 20世纪初, 哥德尔对一阶谓词完全性定理与 N 形式系统的不完全性定理进行了证明。在此基础上, 克林对一般递归函数理论作了深入的研究, 建立了演算理论。英国艾伦·麦席森·图灵 (Alan Mathison Turing, 1912年6月23日—1954年6月7日) 建立了描述算法的机械性思维过程, 提出了理想计算机模型 (即图灵机), 创立了自动机理论。
- 5) 1945年匈牙利冯·诺依曼提出存储程序的思想和建立通用电子数字计算机的冯·诺依曼型体系结构, 此外, 1946年, 美国的莫克利和埃克特成功研制出世界上第一台通用电子数学计算机 ENIAC 做出了开拓性的贡献。

 **注 3.19.** 经典数理逻辑的理论成果为 1956 年人工智能学科的诞生奠定了坚实的逻辑基础。

现代逻辑发展动力主要来自于数学中的公理化运动。20世纪逻辑研究严重数学化, 发展出来的逻辑被恰当地称为“数理逻辑”, 它增强了逻辑研究的深度, 使逻辑学的发展继古希腊逻辑、欧洲中世纪逻辑之后进入第三个高峰期, 并且对整个现代科学产生了非常重要的影响。

3.2.2 逻辑学的发展

3.2.2.1 逻辑学的分类

逻辑学是一门研究思维形式及思维规律的科学。从 17 世纪德国数学家和哲学家莱布尼兹 (G. Leibniz) 提出数理逻辑以来，随着人工智能的一步步的发展，各种各样的逻辑也随之产生。逻辑学大体上可分为经典逻辑、非经典逻辑和现代逻辑。

- 1) 经典逻辑与模态逻辑都是二值逻辑。
- 2) 多值逻辑是具有多个命题真值的逻辑，是向模糊逻辑的逼近。
- 3) 模糊逻辑是处理具有模糊性命题的逻辑。
- 4) 概率逻辑是研究基于逻辑的概率推理。

3.2.2.2 泛逻辑的基本原理

当今人工智能深入发展遇到的一个重大难题就是专家经验知识和常识的推理。迫切需要有一个可靠的逻辑，不精确推理的逻辑学可以作为它们进一步研究信息不完全情况下推理的基础理论，进而形成一种能包容一切逻辑形态和推理模式，灵活开放自适应的逻辑学，这便是柔性逻辑学。而泛逻辑学就是研究刚性逻辑学（数理逻辑）和柔性逻辑学共同规律的逻辑学。

泛逻辑是从更高层面来研究所有形式逻辑的一般规律，希望建立一种能包容一切逻辑形态和推理模式，并能根据需要自由伸缩变化的柔性逻辑学，刚性逻辑学将作为一个最小的内核存在其中，这就是提出泛逻辑的根本原因，也是泛逻辑的最终历史使命。

3.2.3 逻辑学在人工智能学科的研究方面的应用

逻辑方法是人工智能研究中的主要工具，逻辑学的研究成果不但为人工智能学科的诞生奠定了理论基础，而且它们还作为重要的成分被应用于人工智能系统中。

3.2.3.1 经典逻辑的应用

人工智能诞生后的 20 年间是逻辑推理占统治地位的时期。1963 年，纽厄尔、西蒙等人编制的“逻辑理论机”数学定理证明程序 (LT)。在此基础之上，纽厄尔和西蒙编制了通用问题求解程序 (GPS)，开拓了人工智能“问题求解”的一大领域。经典数理逻辑只是数学化的形式逻辑，只能满足人工智能的部分需要。

3.2.3.2 非经典逻辑的应用

(1) 不确定性的推理研究

人工智能是用数值的方法表示和处理不确定信息的方法, 即给系统中每个语句或公式赋一个数值, 用来表示语句的不确定或确定性。比较具有代表性的单个模型有: 1976年杜达提出的主观贝叶斯模型, 1978年扎德提出的可能性模型, 1984年邦迪提出的发生率计算模型, 以及假设推理、定性推理和证据空间理论等经验性模型。

在人工智能中, 可把归纳看成是从个别到一般的推理。归纳逻辑是关于或然性推理的逻辑。借助这种归纳方法和运用类比的方法, 计算机就可以通过问题的相似性, 从相应的知识库中调用有关知识来处理新问题。

(2) 不完全信息的推理研究

常识推理是一种非单调逻辑推理方法, 即人们基于不完全的信息推出某些结论, 当人们得到更完全的信息后, 可以改变甚至收回原来的结论。非单调逻辑可处理信息不充分情况下的推理。20世纪80年代, 赖特(R. Reiter, 发表篇名为“ADe-fault Reasoning”的论文)的缺省逻辑、麦卡锡的限定逻辑、麦克德莫特和多伊尔建立的非单调逻辑推理系统(NMLS)以及摩尔的自认知逻辑都是具有开创性的非单调逻辑系统。常识推理是一种可能出错的不精确的推理, 即容错推理。此外, 多值逻辑和模糊逻辑也已经被引入到人工智能中, 用于处理模糊性和不完全性信息。

 **注 3.20.** 多值逻辑的三个典型系统是克林、卢卡西维兹和波克万的三值逻辑系统。模糊逻辑的研究始于20世纪20年代卢卡西维兹的研究。1972年, 扎德提出了模糊推理的关系合成原则, 现有的大多数模糊推理方法都是关系合成规则的变形或扩充。

3.2.4 人工智能——当代逻辑发展的动力

现代逻辑创始阶段在19世纪末到20世纪早期, 其发展动力主要来自于数学中的公理化运动。

由于人工智能要模拟人的智能, 它的难点不在于人脑所进行的各种必然性推理, 而是最能体现人的智能特征的能动性、创造性思维, 这种思维活动中包括学习、抉择、尝试、修正、推理诸因素。计算机科学和人工智能将至少是21世纪早期逻辑学发展的主要动力源泉, 并将由此决定21世纪逻辑学的另一幅面貌。

例 3.21 选择性地搜集相关的经验证据, 在不充分信息的基础上做出尝试性的判断或抉择, 不断根据环境反馈调整、修正自己的行为, 由此达到实践的成功。

逻辑学将得比较全面地研究人的思维活动, 并着重研究最能体现人的能动性特征的各种不确定性推理, 由此发展出的逻辑理论也将具有更强的可应用性。

3.2.5 人工智能的产生与发展和逻辑学的发展

理论的终极形态是试图找到一个包容一切逻辑的泛逻辑, 得以形成一个完美统一的逻辑基础; 另一方面, 我们还要不断地争论、更新和补充新的逻辑形式. 如果二者能够有机地结合, 将推动人工智能进入一个新的阶段. 概率逻辑大都是基于二值逻辑的, 目前许多专家和学者又在基于其他逻辑的基础上研究概率推理, 使得逻辑学尽可能满足人工智能各方面的发展需要.

就目前来说, 一个新的泛逻辑理论的发展和完善需要一个比较长的时期, 各自发挥其优点, 为人工智能的发展做出贡献. 目前, 许多制约人工智能发展的问题仍有待于解决, 技术上的突破, 还有赖于逻辑学研究上的突破. 在对人工智能的研究中, 只有重视逻辑学, 努力学习与运用并不断深入挖掘其基本内容, 拓宽其研究领域, 才能更好地促进人工智能学科的发展 (机器人网 2017-11-08).

3.3 推理的逻辑基础

3.3.1 谓词公式

3.3.1.1 命题公式的解释

在命题逻辑中, 命题公式的一个解释就是对该命题公式中各个命题变元的一次真值指派. 有了命题公式的解释, 就可据此求出该命题公式的真值.

谓词公式的解释 由于谓词公式中可能包含有个体常量、变元或函数, 因此, 必须先考虑这些个体常量和函数在个体域上的取值, 然后才能根据它们的具体取值为谓词分别指派真值.

定义 3.22 解释 I

设 D 是谓词公式 P 的非空个体域, 若对 P 中的个体常量、函数和谓词按如下规定赋值:

- (1) 为每个个体常量指派 D 中的一个元素;
- (2) 为每个 n 元函数指派一个从 D_n 到 D 的一个映射, 其中

$$D^n = \{(x_1, x_2, \dots, x_n) | x_1, x_2, \dots, x_n \in D\} \quad (3.1)$$

- (3) 为每个 n 元谓词指派一个从 D_n 到 $\{F, T\}$ 的映射. 则称这些指派为 P 在 D 上的一个解释 I .

例 3.23 设个体域 $D = \{1, 2\}$, 求公式 $A = (\forall x)(\forall y)P(x, y)$ 在 D 上的解释, 并指出在每一种解释下公式 A 的真值.

解: 由于公式 A 中没有包含个体常量和函数, 故可直接为谓词指派真值, 设有

表 3-1 公式的一个解释

$P(1, 1)$	$P(1, 2)$	$P(2, 1)$	$P(2, 2)$
T	F	T	F

这就是公式 A 在 D 上的一个解释. 从这个解释可以看出:

- 当 $x = 1, y = 1$ 时, 有 $P(x, y)$ 的真值为 T ;
- 当 $x = 2, y = 1$ 时, 有 $P(x, y)$ 的真值为 T ;

即对 x 在 D 上的任意取值, 都存在 $y = 1$ 使 $P(x, y)$ 的真值为 T . 因此, 在此解释下公式 A 的真值为 T .

 **注 3.24.** 一个谓词公式在其个体域上的解释不是唯一的.

例 3.25 对公式 A , 若给出另一组真值指派.

表 3-2 公式的一个解释

$P(1, 1)$	$P(1, 2)$	$P(2, 1)$	$P(2, 2)$
T	T	F	F

这也是公式 A 在 D 上的一个解释.

从这个解释可以看出:

- 当 $x = 1, y = 1$ 时, $P(x, y)$ 的真值为 T ;
- 当 $x = 2, y = 1$ 时, $P(x, y)$ 的真值为 F ;

即对 x 在 D 上的任意取值, 不存在一个 y , 使得 $P(x, y)$ 的真值为 T . 因此, 在此解释下公式 A 的真值为 F .

 **注 3.26.** 实际上, A 在 D 上共有 16 种解释, 这里就不再一一列举.

例 3.27 设个体域 $D = \{1, 2\}$, 求公式 $B = (\forall x) P(f(x), a)$ 在 D 上的解释, 并指出在该解释下公式 B 的真值.

解: 设对个体常量 a 和函数 $f(x)$ 的值指派如表3-3,

表 3-3 指派方案

a	$f(1)$	$F(2)$
1	1	2

对谓词的真值指派如表3-4,

表 3-4 谓词的真值指派

$P(1, 1)$	$P(1, 2)$	$P(2, 1)$	$P(2, 2)$
T	\times	T	\times

由于已指派 $a = 1$, 因此 $P(1, 2)$ 和 $P(2, 2)$ 不可能出现, 故没给它们指派真值. 上述指派是公式 B 在 D 上的一个解释. 在此解释下有

- 当 $x = 1$ 时, $a = 1$ 使 $P(1, 1) = T$;
- 当 $x = 2$ 时, $a = 1$ 使 $P(2, 1) = T$.

即 x 在 D 上的任意取值, 都存在 $a = 1$ 使 $P(f(x), a)$ 的真值为 T . 因此, 在此解释下公式 $B = (\forall x) P(f(x), a), D = \{1, 2\}$ 的真值为 T .

 **注 3.28.** 由上面的例子可以看出, 谓词公式的真值都是针对某一个解释而言的, 它可能在某一个解释下真值为 T , 而在另一个解释下为 F .

3.3.2 谓词公式的永真性和可满足性

为了推理的需要, 先定义永真公式:

定义 3.29 永真公式

如果谓词公式 P 对非空个体域 D 上的任一解释都取真值 T , 则称 P 在 D 上是永真的; 如果 P 在任何非空个体域上均是永真的, 则称 P 永真.



可见,要判定谓词公式为永真,必须对每个非空个体域上的每个解释逐一进行判断.当解释的个数有限时,尽管工作量大,公式的永真性还可以判定,但当解释个数无限时,永真性就很难判定.

定义 3.30 公式的相容性

对于谓词公式 P ,如果至少存在 D 上的一个解释 I ,使公式 P 在解释 I 下的真值为 T ,则称公式 P 在 D 上是可满足的.谓词公式的可满足性也称相容性.

定义 3.31 永假公式

如果谓词公式 P 对非空个体域 D 上的任一解释 I 下都取真值 F ,则称 P 在 D 上是永假的;如果 P 在任何非空个体域上均是永假的,则称 P 永假.

 **注 3.32.** 谓词公式的永假性又称不可满足性.后面给出的归结推理,就是采用一种逻辑上的反证法,将永真性转换为不可满足性的证明.

表 3-5 逻辑符号及其含义

\neg	\wedge	\vee	\forall	\exists	\in	\subset
否定	合取	析取	全称量词	存在量词	属于	包含于
\supset	\cup	\cap	\rightarrow	\leftarrow	\leftrightarrow	\Rightarrow
包含	并集	交集	蕴涵	逆蕴涵	等值	严格蕴含

3.3.2.1 谓词公式的等价

谓词公式的等价性和永真蕴含性可分别用相应的等价式和永真蕴含式来表示,这些等价式和永真蕴含式都是演绎推理的主要依据,因此也称它们为推理论规则.

定义 3.33 谓词公式的等价式

设 P 与 Q 是 D 上的两个谓词公式,若对 D 上的任意解释 I , P 与 Q 都有相同的真值,则称 P 与 Q 在 D 上是等价的.如果 D 是任意非空个体域,则称 P 与 Q 是等价的,记作 $P \Leftrightarrow Q$.

(1) 双重否定律 $\neg(\neg P) \Leftrightarrow P$.

- (2) 交换律 $(P \vee Q) \Leftrightarrow (Q \vee P), (P \wedge Q) \Leftrightarrow (Q \wedge P)$.
- (3) 结合律 $(P \vee Q) \vee R \Leftrightarrow P \vee (Q \vee R), (P \wedge Q) \wedge R \Leftrightarrow P \wedge (Q \wedge R)$.
- (4) 分配律 $P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \wedge (P \vee R), P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R)$.
- (5) 摩根定律 $\neg(P \vee Q) \Leftrightarrow P \wedge \neg Q, \neg(P \wedge Q) \Leftrightarrow P \vee \neg Q$.
- (6) 吸收律 $P \vee (P \wedge Q) \Leftrightarrow P, P \wedge (P \vee Q) \Leftrightarrow P$.
- (7) 补余律 $\neg P \vee P \Leftrightarrow T, \neg P \wedge P \Leftrightarrow F$.
- (8) 连词化归律 $P \rightarrow Q \Leftrightarrow \neg P \vee Q, P \longleftrightarrow Q \Leftrightarrow (P \rightarrow Q) \wedge (Q \rightarrow P), P \longleftrightarrow Q \Leftrightarrow (P \wedge Q) \vee (Q \wedge P)$.
- (9) 量词转换律 $\neg(\exists x)P \Leftrightarrow (\forall x)(\neg P), \neg(\forall x)P \Leftrightarrow (\exists x)(\neg P)$.
- (10) 量词分配律 $(\forall x)(P \wedge Q) \Leftrightarrow (\forall x)P \wedge (\forall x)Q, (\exists x)(P \vee Q) \Leftrightarrow (\exists x)P \vee (\exists x)Q$.

例 3.3.1

运用连词化归律说明结论: $P \rightarrow Q \Leftrightarrow \neg P \vee Q$.



定义 3.34 永真蕴含

对谓词公式 P 和 Q , 如果 $P \rightarrow Q$ 永真, 则称 P 永真蕴含 Q , 且称 Q 为 P 的逻辑结论, P 为 Q 的前提, P 永真蕴含 Q 记作 $P \Rightarrow Q$.



常用的永真蕴含式有:

- (1) 化简式 $P \wedge Q \Rightarrow P, P \wedge Q \Rightarrow Q$.
- (2) 附加式 $P \Rightarrow P \vee Q, Q \Rightarrow P \vee Q$.
- (3) 析取三段论 $\neg P, P \vee Q \Rightarrow Q$.
- (4) 假言推理 $P, P \rightarrow Q \Rightarrow Q$.
- (5) 拒取式 $\neg Q, P \rightarrow Q \Rightarrow \neg P$.
- (6) 假言三段论 $P \rightarrow Q, Q \rightarrow R \Rightarrow P \rightarrow R$.
- (7) 二难推理 $P \vee Q, P \rightarrow R, Q \rightarrow R \Rightarrow R$.
- (8) 全称固化 $(\forall x)P(x) \Rightarrow P(y)$, 其中, y 是个体域中的任意个体, 依此蕴含式可消去谓词公式中的全称量词.
- (9) 存在固化 $(\exists x)P(x) \Rightarrow P(y)$, 其中, y 是个体域中某一个可以使 $P(y)$ 为真的个体, 依此蕴含式可消去谓词公式中的存在量词.

3.3.3 谓词公式的范式

范式是谓词公式的标准形式。在谓词逻辑中，范式分为两种：前束范式和 Skolem 范式。

 **注 3.35.** 数据科学是科学研究的“第四范式”——图灵奖 Jim Gray.

定义 3.36 前束范式

设 F 为一谓词公式，如果其中的所有量词均非否定地出现在公式的最前面，且它们的辖域为整个公式，则称 F 为前束范式。

一般形式：

$$(Q_1 x_1) \cdots (Q_n x_n) M(x_1, x_2, \dots, x_n), \quad (3.2)$$

其中， $Q_i (i = 1, 2, \dots, n)$ 为前缀，它是一个由全称量词或存在量词组成的量词串； $M(x_1, x_2, \dots, x_n)$ 为母式，它是一个不含任何量词的谓词公式。

例 3.37 $(\forall x)(\forall y)(\exists z)(P(x) \wedge Q(y, z) \vee R(x, z))$ 是前束范式。

任一谓词公式均可化为与其对应的前束范式，化简方法将在后面子句集的化简中讨论。

定义 3.38 Skolem 范式

如果前束范式中所有的存在量词都在全称量词之前，则称这种形式的谓词公式为 Skolem 范式（斯柯林，1920）。

例 3.39 $(\exists x)(\exists z)(\forall y)(P(x) \vee Q(y, z) \wedge R(x, z))$ 是 Skolem 范式。

任一谓词公式均可化为与其对应的一个 Skolem 范式，具体的化简方法也将在后面子句集的化简中讨论。

3.3.4 置换与合一

在不同谓词公式中，往往会出现谓词名相同但其个体不同的情况，此时推理过程是不能直接进行匹配的，需要先进行置换。

例 3.40 可根据全称固化推理和假言推理，由谓词公式 $W_1(A)(\forall x)(W_1(x) \rightarrow W_2(x))$ ，推出 $W_2(A)$ 。

 **注 3.41.** 对谓词 $W_1(A)$ 可看作是由全称固化推理 (即 $(\forall x)(W_1(x) \Rightarrow W_1(A))$) 推出的, 其中 A 是任一个体常量. 要使用假言推理, 首先需要找到项 A 对变元 x 的置换, 使 $W_1(A)$ 与 $W_1(x)$ 一致.

定义 3.42 合一

寻找项对变元的置换, 使谓词一致的过程叫做合一.



下面讨论置换与合一的有关概念与方法. 置换可简单的理解为是在一个谓词公式中用置换项去替换变量.

定义 3.43 置换

置换是形如 $\{t_1/x_1, t_2/x_2, \dots, t_n/x_n\} \equiv \{t_i/x_i\}_{i=1}^{\infty}$ 的有限集合. 其中, t_1, t_2, \dots, t_n 是项; x_1, x_2, \dots, x_n 是互不相同的变元; t_i/x_i 表示用 t_i 替换 x_i . 并且要求 t_i 与 x_i 不能相同, x_i 不能循环地出现在另一个 t_i 中.



例 3.44 $\{a/x, c/y, f(b)/z\}$ 是一个置换. 但 $\{g(y)/x, f(x)/y\}$ 不是一个置换. 原因是它在 x 与 y 之间出现了循环置换现象. 即当用 $g(y)$ 替换 x , 用 $f(g(y))$ 替换 y 时, 既没有消去 x , 也没有消去 y . 若改为 $\{g(a)/x, f(x)/y\}$ 即可, 原因是用 $g(a)$ 替换 x , 用 $f(g(a))$ 替换 y , 则消去了 x 和 y .

 **注 3.45.** 通常, 置换是用希腊字母 θ 、 σ 、 α 、 λ 等来表示.

定义 3.46 例示

设 $\theta = \{t_1/x_1, t_2/x_2, \dots, t_n/x_n\}$ 是一个置换, F 是一个谓词公式, 把公式 F 中出现的所有 x_i 换成 t_i ($i = 1, 2, \dots, n$), 得到一个新的公式 G , 称 G 为 F 在置换 θ 下的例示, 记作 $G = F\theta$.



 **注 3.47.** 一个谓词公式的任何例示都是该公式的逻辑结论.

定义 3.48 置换

设

$$\theta = \{t_1/x_1, t_2/x_2, \dots, t_n/x_n\}, \quad (3.3)$$

$$\lambda = \{u_1/y_1, u_2/y_2, \dots, u_m/y_m\}. \quad (3.4)$$

是两个置换. 则 θ 与 λ 的合成 $\theta \circ \lambda$ 也是一个置换. 它是从集合

$$\{t_1\lambda/x_1, t_2\lambda/x_2, \dots, t_n\lambda/x_n, u_1/y_1, u_2/y_2, \dots, u_m/y_m\}. \quad (3.5)$$

中删去以下两种形式的元素:

- ① 当 $t_i\lambda = x_i$ 时, 删去 $t_i\lambda/x_i$ ($i = 1, 2, \dots, n$);
- ② 当 $y_j \in \{x_1, x_2, \dots, x_n\}$, 删去 u_j/y_j ($j = 1, 2, \dots, m$), 最后剩下的元素所构成的集合.



例 3.49 设 $\theta = \{f(y)/x, z/y\}$, $\lambda = \{a/x, b/y, y/z\}$, 求 θ 与 λ 的合成.

解: 先求出集合

$$\{f(y)\lambda/x, (z\lambda)/y, \lambda\} = \{f(y)\lambda/x, (z\lambda)/y, a/x, b/y, y/z\} = \{f(b)/x, y/y, a/x, b/y, y/z\}, \quad (3.6)$$

其中, $f(b)/x$ 中的 $f(b)$ 是置换 λ 作用于 $f(y)$ 的结果; y/y 中的 y 是置换 λ 作用于 z 的结果. 在该集合中, y/y 满足定义中的删除条件; a/x 和 b/y 满足定义中的删除条件, 也需要删除. 最后得

$$\theta \circ \lambda = \{f(b)/x, y/z\}. \quad (3.7)$$

注 3.50. 合一就是寻找项对变量的置换, 使两个谓词公式最终形式一致.

定义 3.51 合一

设有公式集 $F = \{F_1, F_2, \dots, F_n\}$, 若存在一个置换 θ , 使

$$F_1\theta = F_2\theta = \dots = F_n\theta, \quad (3.8)$$

则称 θ 是 F 的一个合一. 称公式集 F_1, F_2, \dots, F_n 是可合一的.



例 3.52 设有公式集 $F = \{P(x, y, f(y)), P(a, g(x), z)\}$, 则

$$\lambda = \{a/x, g(a)/y, f(g(a))/z\} \quad (3.9)$$

是它的一个合一.

注 3.53. 一般来说, 一个公式集的合一不唯一.

定义 3.54 最一般合一

设 σ 是公式集 F 的一个合一, 如果对 F 的任一个合一 θ 都存在一个置换 λ , 使得 $\theta = \sigma \circ \lambda$, 则称 σ 是一个最一般合一 (**MGU**).



注 3.55. 一个公式集的最一般合一是唯一的. 如何求取最一般合一的问题, 不在此讨论.

3.4 自然演绎推理

定义 3.56 自然演绎推理

从一组已知为真的事实出发, 直接运用经典逻辑中的推理规则, 推出结论的过程称为自然演绎推理.



自然演绎推理最基本的推理规则是三段论推理, 它包括:

- 假言推理 $P, P \rightarrow Q \Rightarrow Q$;
- 拒取式 $\neg Q, P \rightarrow Q \Rightarrow \neg P$;
- 假言三段论 $P \rightarrow Q, Q \rightarrow R \Rightarrow P \rightarrow R$.

例 3.57 设已知如下事实:

$$A, B, A \rightarrow C, B \wedge C \rightarrow D, D \rightarrow Q.$$

求证: Q 为真.

证明: 因为

- $A, A \rightarrow C \Rightarrow C$; 假言推理;
- $B, C \Rightarrow B \wedge C$; 引入合取词;
- $B \wedge C, B \wedge C \rightarrow D \Rightarrow D$; 假言推理;
- $D, D \rightarrow Q \Rightarrow Q$, 假言推理.

因此, Q 为真.

例 3.58 设已知如下事实:

- (1) 只要是需要编程序的课, 王程都喜欢.
- (2) 所有的程序设计语言课都是需要编程序的课.
- (3) C 是一门程序设计语言课.

求证: 王程喜欢 C 这门课.

证明. 首先定义谓词

- (1) $\text{Prog}(x)$ x 是需要编程序的课.
- (2) $\text{Like}(x, y)$ x 喜欢 y .
- (3) $\text{Lang}(x)$ x 是一门程序设计语言课

把例 3.57 的已知事实及待求解问题用谓词公式表示如下:

- (1) $\text{Prog}(x) \rightarrow \text{Like}(\text{Wang}, x)$
- (2) $(\forall x)(\text{Lang}(x) \rightarrow \text{Prog}(x))$

(3) $\text{Lang}(C)$

应用推理规则进行推理:

(1) $\text{Lang}(y) \rightarrow \text{Prog}(y)$

全称固化;

(2) $\text{Lang}(C), \text{Lang}(y) \rightarrow \text{Prog}(y) \Rightarrow \text{Prog}(C)$

假言推理 $\{C/y\}$;

(3) $\text{Prog}(C), \text{Prog}(x) \rightarrow \text{Like}(\text{Wang}, x) \Rightarrow \text{Like}(\text{Wang}, C)$ 假言推理 $\{C/x\}$.

因此, 王程喜欢 C 这门课.

□

注 3.59. 自然演绎推理的优缺点

优点: 定理证明过程自然, 易于理解, 并且有丰富的可用推理规则.

缺点: 是容易产生知识爆炸, 推理过程中得到的中间结论数量一般按指数规律递增, 不利于复杂问题的推理, 甚至难以实现.

3.5 归结演绎推理

归结演绎推理是一种基于鲁宾逊 (Robinson) 归结原理的机器推理技术. 鲁宾逊归结原理亦称消解原理, 是鲁宾逊于 1965 年在海伯伦 (Herbrand) 理论的基础上提出的一种基于逻辑的“反证法”.

在人工智能中, 几乎所有的问题都可以转化为一个定理证明问题. 定理证明的实质, 就是要对前提 P 和结论 Q , 证明 $P \rightarrow Q$ 永真.

由 3.3 节可知, 要证明 $P \rightarrow Q$ 永真, 就是要证明 $P \rightarrow Q$ 在任何一个非空的个体域上都是永真的. 这将是非常困难的, 甚至是不可实现的. 为此, 人们进行了大量的探索, 后来发现可以采用反证法的思想, 即把关于永真性的证明转化为关于不可满足性的证明. 即要证明 $P \rightarrow Q$ 永真, 只要能够证明 $P \wedge \neg Q$ 是不可满足的就可以了 (原因是 $\neg(P \rightarrow Q) \Leftrightarrow \neg(\neg P \vee Q) \Leftrightarrow P \wedge \neg Q$). 这方面最有成效的工作就是鲁宾逊归结原理. 它使定理证明的机械化成为现实.

3.5.1 子句集及其化简

鲁宾逊归结原理在子句集的基础上讨论问题. 因此, 讨论归结演绎推理之前, 需要先讨论子句集的有关概念.

定义 3.60 文字

原子谓词公式及其否定统称为文字.



例 3.61 $P(x)$ 、 $Q(x)$ 、 $\neg P(x)$ 和 $\neg Q(x)$ 等都是文字.

定义 3.62 子句

任何文字的析取式称为子句.

例 3.63 $P(x) \vee Q(x)$ 和 $P(x, f(x)) \vee Q(x, g(x))$ 都是子句.

定义 3.64 空子句

不含任何文字的子句称为空子句.

由于空子句不含有任何文字, 也就不能被任何解释所满足, 因此空子句永假且不可满足. 空子句一般被记为 NIL.

定义 3.65 子句集

由子句或空子句构成的集合称为子句集.

在谓词逻辑中, 任何一个谓词公式都可以通过等价关系及推理规则化成相应的子句集. 其化简的 9 个步骤如下:

(1) 消去连接词 “ \rightarrow ” 和 “ \longleftrightarrow ”. 需要反复使用如下等价公式:

$$P \rightarrow Q \Leftrightarrow \neg P \vee Q \quad (3.10)$$

$$P \longleftrightarrow Q \Leftrightarrow (P \wedge Q) \vee (\neg P \wedge \neg Q). \quad (3.11)$$

即可消去谓词公式中的连接词 “ \rightarrow ” 和 “ \longleftrightarrow ”.

例 3.66 公式

$$(\forall x)((\forall y)P(x, y) \rightarrow \neg(\forall y)(Q(x, y) \rightarrow R(x, y))), \quad (3.12)$$

经等价变化后为

$$(\forall x)(\neg(\forall y)P(x, y) \vee \neg(\forall y)(\neg Q(x, y) \vee R(x, y))). \quad (3.13)$$

(2) 减少否定符号的辖域

反复使用双重否定率

$$\neg(\neg P) \Leftrightarrow P. \quad (3.14)$$

摩根定律

$$\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q, \quad (3.15)$$

$$\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q. \quad (3.16)$$

量词转换律

$$\neg(\forall x)P(x) \Leftrightarrow (\exists x)\neg P(x), \quad (3.17)$$

$$\neg(\exists x)P(x) \Leftrightarrow (\forall x)\neg P(x). \quad (3.18)$$

将每个否定符号“ \neg ”移到仅靠谓词的位置，使得每个否定符号最多只作用于一个谓词上。

例 3.67 例3.66中的公式(3.13)经等价变换后为

$$(\forall x)((\exists y)\neg P(x, y) \vee (\exists y)(Q(x, y) \wedge \neg R(x, y))). \quad (3.19)$$

(3) 对变元标准化

在一个量词的辖域内，把谓词公式中受该量词约束的变元全部用另外一个没有出现过的任意变元代替，使不同量词约束的变元有不同的名字。

例 3.68 上式经变换后为

$$(\forall x)((\exists y)\neg P(x, y) \vee (\exists z)(Q(x, z) \wedge \neg R(x, z))). \quad (3.20)$$

(4) 化为前束范式

化为前束范式的方法：把所有量词都移到公式的左边，并且在移动时不能改变其相对顺序。由于第(3)步已对变元进行了标准化，每个量词都有自己的变元，这就消除了任何由变元引起冲突的可能，因此这种移动是可行的。

例 3.69 上式化为前束范式后为

$$(\forall x)(\exists y)(\exists z)(\neg P(x, y) \vee (Q(x, z) \wedge \neg R(x, z))). \quad (3.21)$$

(5) 消去存在量词

消去存在量词时，需要区分以下两种情况：

- ▶ 若存在量词不出现在全称量词的辖域内(即它的左边没有全称量词),只要用一个新的个体常量替换受该存在量词约束的变元,就可消去该存在量词.
- ▶ 若存在量词位于一个或多个全称量词的辖域内,

$$(\forall x_1) \cdots (\forall x_n)(\exists y) P(x_1, x_2, \dots, x_n, y). \quad (3.22)$$

则需要用 Skolem 函数 $f(x_1, x_2, \dots, x_n)$ 替换受该存在量词约束的变元 y , 然后再消去该存在量词.

例 3.70 上步所得公式中存在量词 $(\exists y)$ 和 $(\exists z)$ 都位于 $(\forall x)$ 的辖域内, 因此都需要用 Skolem 函数来替换. 设替换 y 和 z 的 Skolem 函数分别是 $f(x)$ 和 $g(x)$, 则替换后的式子为

$$(\forall x)(\neg P(x, f(x)) \vee (Q(x, g(x)) \wedge \neg R(x, g(x)))). \quad (3.23)$$

(6) 化为 Skolem 标准形

Skolem 标准形的一般形式为

$$(\forall x_1) \cdots (\forall x_n) M(x_1, x_2, \dots, x_n), \quad (3.24)$$

其中, $M(x_1, x_2, \dots, x_n)$ 是 Skolem 标准形的母式(由子句的合取所构成).

 **注 3.71.** 把谓词公式化为 Skolem 标准形, 需要使用以下等价关系

$$P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \textcolor{blue}{\wedge} (P \vee R). \quad (3.25)$$

例 3.72 前面的公式化为 Skolem 标准形后为

$$(\forall x)((\neg P(x, f(x)) \vee Q(x, g(x)) \wedge (\neg P(x, f(x)) \vee \neg R(x, g(x)))). \quad (3.26)$$

(7) 消去全称量词

由于母式中的全部变元均受全称量词的约束, 并且全称量词的次序已无关紧要, 因此可以省掉全称量词. 仍假设剩下的母式的变元是被全称量词量化的.

例 3.73 式(3.26)消去全称量词后为

$$(\neg P(x, f(x)) \vee Q(x, g(x)) \textcolor{blue}{\wedge} (\neg P(x, f(x)) \vee \neg R(x, g(x))). \quad (3.27)$$

(8) 消去合取词

在母式中消去所有合取词, 把母式用子句集的形式表示出来. 子句集中的每一个元素都是一个子句.

例 3.74 上式的子句集中包含以下两个子句

$$\neg P(x, f(x)) \vee Q(x, g(x)), \quad (3.28)$$

$$\neg P(x, f(x)) \vee \neg R(x, g(x)). \quad (3.29)$$

(9) 更换变量名称

对子句集中的某些变量重新命名, 使任意两个子句中不出现相同的变量名. 由于每一个子句都对应着母式中的一个合取元, 并且所有变元都是由全称量词量化, 因此任意两个不同子句的变量之间实际上不存在任何关系. 这样, 更换变量名是不会影响公式的真值的.

例 3.75 对前面的公式, 可把第二个子句集中的变元名 x 更换为 y , 得到如下子句集

$$\neg P(x, f(x)) \vee Q(x, g(x)). \quad (3.30)$$

$$\neg P(y, f(y)) \vee \neg R(y, g(y)). \quad (3.31)$$

子句集及其化简 在上述化简过程中, 由于在消去存在量词时, 所用的 Skolem 函数可以不同, 因此化简后的标准子句集是不唯一的. 这样, 当原谓词公式为非永假时, 它与其标准子句集并不等价. 但当原谓词公式为永假(不可满足)时, 其标准子句集则一定是永假的, 即 Skolem 化并不影响原谓词公式的永假性.

不可满足的充要条件这一结论很重要, 是归结原理的主要依据, 可用定理的形式来描述.

定理 3.1 不可满足的充要条件

设有谓词公式 F , 其标准子句集为 S , 则谓词公式 F 不可满足的充要条件是标准子句集 S 不可满足.



为证明此定理, 先作如下说明: 为讨论问题方便, 设给定的谓词公式 F 已为前束形

$$(Q_1 x_1) \cdots (Q_r x_r) \cdots (Q_n x_n) M(x_1, x_2, \dots, x_n), \quad (3.32)$$

其中, $M(x_1, x_2, \dots, x_n)$ 已化为合取范式. 由于将 F 化为这种前束形是一种很容易实现的等价运算, 因此这种假设是可以的.

又设 $(Q_r x_r)$ 是第一个出现的存在量词 $(\exists x_r)$, 即 F 为

$$F = (\forall x_1) \cdots (\forall x_{r-1}) (\exists x_r) (Q_{r+1} x_{r+1}) \cdots (Q_n x_n) M(x_1, \dots, x_{r-1}, x_r, x_{r+1}, \dots, x_n), \quad (3.33)$$

为把 F 化为 Skolem 形, 需要先消去这个 $(\exists x_r)$, 并引入 Skolem 函数, 得到

$$F_1 = (\forall x_1) \cdots (\forall x_{r-1}) (Q_{r+1} x_{r+1}) \cdots (Q_n x_n) M(x_1, \dots, x_{r-1}, f(x_1, \dots, x_{r-1}), x_{r+1}, \dots, x_n). \quad (3.34)$$

 **注 3.76.** 若能证明

$$F \text{ 不可满足} \Leftrightarrow F_1 \text{ 不可满足}. \quad (3.35)$$

则同理可证

$$F_1 \text{ 不可满足} \Leftrightarrow F_2 \text{ 不可满足}. \quad (3.36)$$

重复这一过程, 直到证明了

$$F_{m-1} \text{ 不可满足} \Leftrightarrow F_m \text{ 不可满足} \quad (3.37)$$

为止. 此时, F_m 已为 F 的 Skolem 标准形. 而 S 只不过是 F_m 的一种集合表示形式. 因此有

$$F_m \text{ 不可满足} \Leftrightarrow S \text{ 不可满足}. \quad (3.38)$$

证明. (用反证法证)

$$F \text{ 不可满足} \Leftrightarrow F_1 \text{ 不可满足}. \quad (3.39)$$

先证明充分性, 已知 F 不可满足, 假设 F_1 是可满足的, 则存在一个解释 I , 使 F_1 在解释 I 下为真. 即对任意 x_1, \dots, x_{r-1} 在 I 的设定下有

$$(Q_{r+1} x_{r+1}) \cdots (Q_n x_n) M(x_1, \dots, x_{r-1}, f(x_1, \dots, x_{r-1}), x_{r+1}, \dots, x_n) \quad (3.40)$$

为真. 亦即对任意的 x_1, \dots, x_{r-1} 都有一个 $f(x_1, \dots, x_{r-1})$ 使

$$(Q_{r+1} x_{r+1}) \cdots (Q_n x_n) M(x_1, \dots, x_{r-1}, f(x_1, \dots, x_{r-1}), x_{r+1}, \dots, x_n) \quad (3.41)$$

为真. 即在 I 下有

$$(\forall x_1) \cdots (\forall x_{r-1}) (\exists x_r) (Q_{r+1}x_{r+1}) \cdots (Q_n x_n) M(x_1, \dots, x_{r-1}, x_r, x_{r+1}, \dots, x_n) \quad (3.42)$$

为真. 即 F 在 I 下为真.

但这与前提 F 相矛盾, 即假设 F_1 为可满足是错误的. 从而可以得出“若 F 不可满足, 则必有 F_1 不可满足”.

再证明必要性, 已知 F_1 不可满足, 假设 F 是可满足的. 于是便有某个解释 I 使 F 在 I 下为真. 即对任意的 x_1, \dots, x_{r-1} 在 I 的设定下都可找到一个 x_r 使

$$(Q_{r+1}x_{r+1}) \cdots (Q_n x_n) M(x_1, \dots, x_{r-1}, x_r, x_{r+1}, \dots, x_n) \quad (3.43)$$

为真. 若扩充 I , 使它包含一个函数 $f(x_1, \dots, x_{r-1})$, 且有

$$x_r = f(x_1, \dots, x_{r-1}), \quad (3.44)$$

这样, 就可以把所有的 $f(x_1, \dots, x_{r-1})$ 映射到 x_r , 从而得到一个新的解释 I' , 并且在此解释下对任意的 x_1, \dots, x_{r-1} 都有

$$(Q_{r+1}x_{r+1}) \cdots (Q_n x_n) M(x_1, \dots, x_{r-1}, f(x_1, \dots, x_{r-1}), x_{r+1}, \dots, x_n) \quad (3.45)$$

为真. 即在 I' 下有

$$(\forall x_1) \cdots (\forall x_{r-1}) (Q_{r+1}x_{r+1}) \cdots (Q_n x_n) M(x_1, \dots, x_{r-1}, f(x_1, \dots, x_{r-1}), x_{r+1}, \dots, x_n) \quad (3.46)$$

为真. 它说明 F_1 在解释 I' 下为真. 但这与前提 F_1 是不可满足的相矛盾, 即假设 F 为可满足是错误的. 从而可以得出“若 F_1 不可满足, 则必有 F 不可满足”. 于是, 定理得证. \square

 **注 3.77.** 由此定理可知, 证明一个谓词公式是不可满足的, 只要证明其相应的标准子句集是不可满足的就可以了. 至于如何证明一个子句集的不可满足性, 由下面的海伯伦理论和鲁宾逊归结原理来解决.

3.5.2 鲁宾逊归结原理

鲁宾逊归结原理包括: 命题逻辑的归结原理和谓词逻辑的归结原理.

鲁宾逊归结原理基本思想 1) 子句集中的子句之间是合取关系. 因此, 子句集中只要有一个子句为不可满足, 则整个子句集就是不可满足的.

2) 空子句是不可满足的. 因此, 一个子句集中如果包含有空子句, 则此子句集就一定是不可满足的.

首先把欲证明问题的结论否定, 并加入子句集, 得到一个扩充的子句集 S' . 然后设法检验子句集 S' 是否含有空子句, 若含有空子句, 则表明 S' 是不可满足的; 若不含有空子句, 则继续使用归结法, 在子句集中选择合适的子句进行归结, 直至导出空子句或不能继续归结为止.

3.5.2.1 命题逻辑的归结

归结推理的核心是求两个子句的归结式.

定义 3.78 互补文字

若 P 是原子谓词公式, 则称 P 与 $\neg P$ 为互补文字.

定义 3.79 归结与亲本子句

设 C_1 和 C_2 是子句集中的任意两个子句, 如果 C_1 中的文字 L_1 与 C_2 中的文字 L_2 互补, 那么可从 C_1 和 C_2 中分别消去 L_1 和 L_2 , 并将 C_1 和 C_2 中余下的部分按析取关系构成一个新的子句 C_{12} , 则称这一过程为归结. 称 C_{12} 为 C_1 和 C_2 的归结式, 称 C_1 和 C_2 为 C_{12} 的亲本子句.

例 3.80 设 $C_1 = P \vee Q \vee R$, $C_2 = \neg P \vee S$, 求 C_1 和 C_2 的归结式 C_{12} .

解: 这里 $L_1 = P$, $L_2 = \neg P$, 通过归结可以得到

$$C_{12} = Q \vee R \vee S.$$

例 3.81 设 $C_1 = \neg Q$, $C_2 = Q$, 求 C_1 和 C_2 的归结式 C_{12} .

解: 这里 $L_1 = \neg Q$, $L_2 = Q$, 通过归结可以得到

$$C_{12} = \text{NIL}. \quad (3.47)$$

例 3.82 设 $C_1 = \neg P \vee Q$, $C_2 = \neg Q$, $C_3 = P$, 求 C_1, C_2 和 C_3 的归结式 C_{123} .

解: 若先对 C_1, C_2 归结, 可得到

$$C_{12} = \neg P. \quad (3.48)$$

然后再对 C_{12} 和 C_3 归结, 得到

$$C_{123} = \text{NIL}. \quad (3.49)$$

如果改变归结顺序, 先计算归结式 C_{13} , 同样可以得到相同的结果 $C_{123} = \text{NIL}$, 即其归结过程是不唯一的.

其归结归结过程可用图3-3来表示, 该树称为归结树.

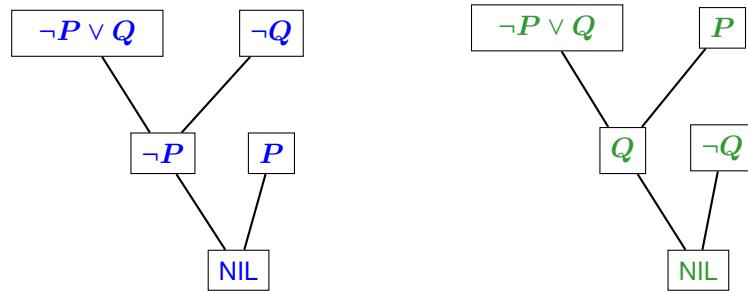


图 3-3 归结树

定理 3.2 归结式的亲本子句

归结式 C_{12} 是其亲本子句 C_1 和 C_2 的逻辑结论.



证明. 由已知条件, 子句 C_1 和 C_2 的归结式是 C_{12} , (按定义), 知 $C_1 = L \vee C'_1$, $C_2 = \neg L \vee C'_2$ 关于解释 I 为真, 则只需证明 $C_{12} = C'_1 \vee C'_2$ 关于解释 I 也为真. 对于解释 I , 文字 L 和 $\neg L$ 中必有一个为假. 若 L 为假, 则必有 C'_1 为真, 不然就会使 C_1 为假, 这将与前提假设 C_1 为真矛盾, 因此只能有 C'_1 为真. 同理, 若 $\neg L$ 为假, 则必有 C'_2 为真.

因此, 必有 $C_{12} = C'_1 \vee C'_2$ 关于解释 I 也为真. 即 C_{12} 是 C_1 和 C_2 的逻辑结论. □

上述定理是归结原理中的一个重要定理, 由它可得到以下两个性质推论:

性质 3.1 归结式的不可满足性

设 C_1 和 C_2 是子句集 S 中的两个子句, C_{12} 是 C_1 和 C_2 的归结式, 若用 C_{12} 代替 C_1 和 C_2 后得到新的子句集 S_1 , 则由 S_1 的不可满足性可以推出原子句集 S 的不可满足性. 即: S_1 的不可满足性 $\Rightarrow S$ 的不可满足性.

证明. 设 $S = \{C_1, C_2, C_3, \dots, C_n\}$, C_{12} 是 C_1 和 C_2 的归结式, 则用 C_{12} 代替 C_1 和 C_2 后, 可得到一个新的子句集

$$S_1 = \{C_{12}, C_3, \dots, C_n\}.$$

设 S_1 是不可满足的, 则对不满足 S_1 的任一解释 I , 都可能有以下两种情况:

- ① 解释 I 使 C_{12} 为真, 则 C_3, \dots, C_n 中必有一个为假, 即 S 是不可满足的.
- ② 解释 I 使 C_{12} 为假, 即 $\neg C_{12}$ 为真, 根据定理 3.2 ($C_1 \wedge C_2 \Rightarrow C_{12} \Leftrightarrow \neg C_{12} \Rightarrow \neg C_1 \vee \neg C_2$), 即 $\neg C_1 \vee \neg C_2$ 永真, 它说明解释 I 使 C_1 或 C_2 为假. 即 S 也是不可满足的.

因此可以得出

$$S_1 \text{ 的不可满足性} \Rightarrow S \text{ 的不可满足性.}$$

□

性质 3.2 归结式的不可满足性

设 C_1 和 C_2 是子句集 S 中的两个子句, C_{12} 是 C_1 和 C_2 的归结式, 若把 C_{12} 加入 S 中, 得到新的子句集 S_2 , 则 S 与 S_2 的不可满足性是等价的. 即:

$$S_2 \text{ 的不可满足性} \Leftrightarrow S \text{ 的不可满足性.}$$

◆

证明. 设 $S = \{C_1, C_2, C_3, \dots, C_n\}$ 是不可满足的, 则 $C_1, C_2, C_3, \dots, C_n$ 中必有一子句为假. 不妨设 C_1, C_2 可归结, 因而 $S_2 = \{C_{12}, C_1, C_2, C_3, \dots, C_n\}$ 必为不可满足.

再证明 \Rightarrow .

设 S_2 是不可满足的, 则对不满足 S_2 的任一解释 I , 都可能有以下两种情况:

- ① 解释 I 使 C_{12} 为真, 则 $C_1, C_2, C_3, \dots, C_n$ 中必有一个为假, 即 S 是不可满足的.
- ② 解释 I 使 C_{12} 为假, 即 $\neg C_{12}$ 为真, 由推理规则的拒取式, 即 $\neg C_1 \vee \neg C_2$ 永真, 它说明解释 I 使 C_1 为假, 或 C_2 为假. 即 S 也是不可满足的.

由此可见 S 与 S_2 的不可满足性是等价的. 即

$$S \text{ 的不可满足性} \Leftrightarrow S_2 \text{ 的不可满足性.}$$

□

上述两个性质说明, 为证明子句集 S 的不可满足性, 只要对其中可进行归结后的子句进行归结, 并把归结式加入到子句集 S 中, 或者用归结式代替他的亲本子句, 然后对新的子句集证明其不可满足性就可以了. 如果经归结能得到空子句, 根据空子句的不可满足性, 即可得到原子句集 S 是不可满足的结论.

在命题逻辑中, 对不可满足的子句集 S , 其归结原理是完备的. 这种不可满足性可用如下定理描述:

定理 3.3 子句集不可满足的判定

子句集 S 不可满足当且仅当存在一个从 S 到空子句的归结过程.



3.5.2.2 命题逻辑的归结反演

归结原理: 假设 F 为已知前提, G 为欲证明的结论, 归结原理把证明 G 为 F 的逻辑结论转化为证明 $F \wedge \neg G$ 为不可满足. 再根据定理 3.1, 在不可满足的意义下, 公式集 $F \wedge \neg G$ 与其子句集是等价的, 即把公式集上的不可满足转化为子句集上的不可满足.

定义 3.83 归结反演

应用归结原理证明定理的过程称为归结反演.



注 3.84. 在命题逻辑中, 已知 F , 证明 G 为真的归结反演过程如下:

- ① 否定目标公式 G , 得 $\neg G$;
- ② 把 $\neg G$ 并入到公式集 F 中, 得到公式集 $\{F, \neg G\}$;
- ③ 把公式集 $\{F, \neg G\}$ 化为子句集 S .
- ④ 应用归结原理对子句集 S 中的子句进行归结, 并把每次得到的归结式并入 S 中. 如此反复进行, 若出现空子句, 则停止归结, 此时就证明了 G 为真.

例 3.85 设已知的公式集为 $\{P, (P \wedge Q) \rightarrow R, (S \vee T) \rightarrow Q, T\}$, 求证结论 R .

解: 假设结论 R 为假, 将 $\neg R$ 加入公式集, 并化为子句集

$$S = \{P, \neg P \vee \neg Q \vee R, \neg S \vee Q, \neg T \vee Q, T, \neg R\}.$$

其归结过程如图3-4示出的归结演绎树所示. 该树根为空子句.



图 3-4 归结树

其含义为：先假设子句集 S 中的所有子句均为真，即原公式集为真， $\neg R$ 也为真；然后利用归结原理，对子句集进行归结，并把所得的归结式并入子句集中；重复这一过程，最后归结出了空子句。

根据归结原理的完备性，可知子句集 S 是不可满足的，即开始时假设 $\neg R$ 为真是错误的，这就证明了 R 为真。

3.5.2.3 谓词逻辑的归结

在谓词逻辑中，由于子句集 S 中的谓词一般都含有变元，因此不能像命题逻辑那样直接消去互补文字。而需要先用一个最一般合一对变元进行代换，然后才能进行归结。可见，谓词逻辑的归结要比命题逻辑的归结多一个合一过程。

谓词逻辑的归结原理 谓词逻辑中的归结式可用如下定义来描述：

定义 3.86 归结式上的文字

设 C_1 和 C_2 是两个没有公共变元的子句， L_1 和 L_2 分别是 C_1 和 C_2 中的文字。如果 L_1 和 L_2 存在最一般合一 σ ，则称

$$C_{12} = (C_1\sigma - \{L_1\sigma\}) \cup (C_2\sigma - \{L_2\sigma\}) \quad (3.50)$$

为 C_1 和 C_2 的二元归结式，而文字 L_1 和 L_2 称为归结式上的文字。



例 3.87 设 $C_1 = P(a) \vee R(x)$, $C_2 = \neg P(y) \vee Q(b)$, 求 C_{12} .

解: 取 $L_1 = P(a)$, $L_2 = \neg P(y)$, 则 L_1 和 L_2 的最一般合一是 $\sigma = \{a/y\}$. 根据定义 3.81, 可得

$$\begin{aligned} C_{12} &= (C_1\sigma - \{L_1\sigma\}) \cup (C_2\sigma - \{L_2\sigma\}) = (\{P(a), R(x)\} - \{P(a)\}) \cup (\{\neg P(a), Q(b)\} - \{\neg P(a)\}) \\ &= (\{R(x)\}) \cup (\{Q(b)\}) = \{R(x), Q(b)\} \\ &= R(x) \vee Q(b). \end{aligned} \quad (3.51)$$

例 3.88 设 $C_1 = P(x) \vee Q(a)$, $C_2 = \neg P(b) \vee R(x)$, 求 C_{12} .

解: 由于 C_1 和 C_2 有相同的变元 x , 不符合定义的要求. 为了进行归结, 需要修改 C_2 中变元的名字, 令 $C_2 = \neg P(b) \vee R(y)$. 此时 $L_1 = P(x)$, $L_2 = \neg P(b)$, L_1 和 L_2 的最一般合一是 $\sigma = \{b/x\}$. 则有

$$\begin{aligned} C_{12} &= (C_1\sigma - \{L_1\sigma\}) \cup (C_2\sigma - \{L_2\sigma\}) = (\{P(b), Q(a)\} - \{P(b)\}) \cup (\{\neg P(b), R(y)\} - \{\neg P(b)\}) \\ &= (\{Q(a)\}) \cup (\{R(y)\}) = \{Q(a), R(y)\} \\ &= Q(a) \vee R(y). \end{aligned} \quad (3.52)$$

 **注 3.89.** (1) 这里使用了集合符号和集合运算, 目的是为了方便说明问题. 即先将子句 $C_i\sigma$ 和 $L_i\sigma$ 写成集合的形式, 在集合表示下做减法和并集运算, 然后再写成子句集的形式.
(2) 定义中还要求 C_1 和 C_2 无公共变元, 这也是合理的.

例 3.90 例如 $C_1 = P(x)$, $C_2 = \neg P(f(x))$, 而 $S = \{C_1, C_2\}$ 是不可满足的.

 **注 3.91.** 但由于 C_1 和 C_2 的变元相同, 就无法合一了. 没有归结式, 就不能用归结法证明 S 的不可满足性, 这就限制了归结法的使用范围. 如果对 C_1 或 C_2 的变元进行换名, 便可通过合一, 对 C_1 和 C_2 进行归结. 如上例 3.90, 若先对 C_2 进行换名, 即 $C_2 = \neg P(f(y))$, 则可对 C_1 和 C_2 进行归结, 得到一个空子句, 从而证明了 S 是不可满足的. 事实上, 在由公式集化为子句集的过程中, 其最后一步就是做换名处理. 因此, 定义中假设 C_1 和 C_2 没有相同变元是可以的.

例 3.92 设 $C_1 = P(x) \vee \neg Q(b)$, $C_2 = \neg P(a) \vee Q(y) \vee R(z)$. 对 C_1 和 C_2 通过最一般合一 ($\sigma = \{a/x, b/y\}$) 的作用, 可以得到互补对.

 **注 3.93.** 求归结式不能同时消去两个互补对, 这样的结果不是二元归结式. 如在 $\sigma = \{a/x, b/y\}$ 下, 若同时消去两个互补对, 所得的 $R(z)$ 不是 C_1 和 C_2 的二元归结式.

例 3.94 设 $C_1 = P(x) \vee P(f(a)) \vee Q(x), C_2 = \neg P(y) \vee R(b)$, 求 C_{12} .

解: 对参加归结的某个子句, 若其内部有可合一的文字, 则在进行归结之前应先对这些文字进行合一. 本例的 C_1 中有可合一的文字 $P(x)$ 与 $P(f(a))$, 若用它们的最一般合一 $\sigma = \{f(a)/x\}$ 进行代换, 可得到

$$C_1\sigma = P(f(a)) \vee Q(f(a)), \quad (3.53)$$

此时可对 $C_1\sigma$ 与 C_2 进行归结. 选 $L_1 = P(f(a)), L_2 = \neg P(y)$, L_1 和 L_2 的最一般合一是 $\sigma = \{f(a)/y\}$, 则可得到 C_1 和 C_2 的二元归结式为

$$C_{12} = R(b) \vee Q(f(a)). \quad (3.54)$$

定义 3.95 单元因子

我们把 $C_1\sigma$ 称为 C_1 的因子. 一般来说, 若子句 C 中有两个或两个以上的文字具有最一般合一 σ , 则称 $C\sigma$ 为子句 C 的因子. 如果 $C\sigma$ 是一个单文字, 则称它为 C 的单元因子.

应用因子概念, 可对谓词逻辑中的归结原理给出如下定义:

定义 3.96 二元归结式

若 C_1 和 C_2 是无公共变元的子句, 则

- ① C_1 和 C_2 的二元归结式;
- ② C_1 和 C_2 的因子 $C_2\sigma_2$ 的二元归结式;
- ③ C_1 的因子 $C_1\sigma_1$ 和 C_2 的二元归结式;
- ④ C_1 的因子 $C_1\sigma_1$ 和 C_2 的因子 $C_2\sigma$ 的二元归结式.

这四种二元归结式都是子句 C_1 和 C_2 的二元归结式, 记为 C_{12} .

例 3.97 设 $C_1 = P(y) \vee P(f(x)) \vee Q(g(x)), C_2 = \neg P(f(g(a))) \vee Q(b)$, 求 C_{12} .

解: 对 C_1 取最一般合一 $\sigma = \{f(x)/y\}$, 得 C_1 的因子

$$C_1\sigma = P(f(x)) \vee Q(g(x)), \quad (3.55)$$

对 C_1 的因子和 C_2 归结 (合一 $\sigma = \{g(a)/x\}$), 可得到 C_1 和 C_2 的二元归结式

$$C_{12} = Q(g(g(a))) \vee Q(b). \quad (3.56)$$

 **注 3.98.** 对谓词逻辑, 定理 3.2 仍然适用, 即归结式 C_{12} 是其亲本子句 C_1 和 C_2 的逻辑结论. 用归结式取代它在子句集 S 中的亲本子句, 所得到的子句集仍然保持着原子句集 S 的不可满足性. 此外, 对谓词逻辑定理 3.3 也仍然适用, 即从不可满足的意义上说, 一阶谓词逻辑的归结原理完备.

3.5.2.4 谓词逻辑的归结反演

谓词逻辑的归结反演过程与命题逻辑的归结反演过程相比, 其步骤基本相同, 但每步的处理对象不同.

- ▶ 在步骤(3)化简子句集时, 谓词逻辑需要把由谓词构成的公式集化为子句集;
- ▶ 在步骤(4)按归结原理进行归结时, 谓词逻辑的归结原理需要考虑两个亲本子句的最一般合一.

例 3.99 已知

$$F : (\forall x)((\exists y)(A(x, y) \wedge B(y)) \rightarrow (\exists y)(C(y) \wedge D(x, y))), \quad (3.57)$$

$$G : \neg(\exists x)C(x) \rightarrow (\forall x)(\forall y)(A(x, y) \rightarrow \neg B(y)). \quad (3.58)$$

求证 G 是 F 的逻辑结论.

证明. 先把 G 否定, 并放入 F 中, 得到的 $\{F, \neg G\}$ 为

$$\begin{aligned} \{F, \neg G\} = & \{(\forall x)((\exists y)(A(x, y) \wedge B(y)) \rightarrow (\exists y)(C(y) \wedge D(x, y))), \\ & \neg(\neg(\exists x)C(x) \rightarrow (\forall x)(\forall y)(A(x, y) \rightarrow \neg B(y)))\}. \end{aligned} \quad (3.59)$$

再把 $\{F, \neg G\}$ 化成子句集, 得到

- (1) $\neg A(x, y) \vee \neg B(y) \vee C(f(x))$.
- (2) $\neg A(u, v) \vee \neg B(v) \vee D(u, f(u))$.
- (3) $\neg C(z)$.
- (4) $A(m, n)$.

(5) $B(k)$, 其中(1)、(2)是由 F 化出的两个子句, (3)、(4)和(5)是由 $\neg G$ 化出的 3 个子句. 最后应用谓词逻辑的归结原理对上述子句集进行归结, 其过程为

(6) $\neg A(x, y) \vee \neg B(y)$, 由(1)和(3)归结, 取合一 $\sigma = \{f(x)/z\}$,

(7) $\neg B(n)$, 由(4)和(6)归结, 取合一 $\sigma = \{m/x, n/y\}$,

(8) NIL, 由(5)和(7)归结, 取合一 $\sigma = \{n/k\}$, 因此, G 是 F 的逻辑结论. \square

得到的子句集为 $\{\neg A(x, y) \vee \neg B(y) \vee C(f(x)), \neg A(u, v) \vee \neg B(v), D(u, f(u)), \neg C(z), A(m, n), B(k)\}$. 上述归结过程可用归结树(图3-5)来表示



图 3-5 归结树

为了进一步加深对谓词逻辑归结的理解, 下面再给出一个经典的归结问题.

例 3.100 (“快乐学生”问题) 假设: 任何通过计算机考试并获奖的人都是快乐的, 任何肯学习或幸运的人都可以通过所有考试, 张不肯学习但他是幸运的, 任何幸运的人都能获奖. 求证: 张是快乐的.

解: 先定义谓词:

Pass(x, y)	x 可以通过 y 考试
Win($x, prize$)	x 能获得奖励
Study(x)	x 肯学习
Happy(x)	x 是快乐的
Lucky(x)	x 是幸运的

再将问题用谓词表示如下: “任何通过计算机考试并奖的人都是快乐的”.

$$(\forall x)(\text{Pass}(x, \text{computer}) \wedge \text{Win}(x, \text{prize}) \rightarrow \text{Happy}(x)). \quad (3.60)$$

“任何肯学习或幸运的人都可以通过所有考试”.

$$(\forall x)(\forall y)(\text{Study}(x) \vee \text{Lucky}(x) \rightarrow \text{Pass}(x, y)). \quad (3.61)$$

“张不肯学习但他是幸运的”.

$$\neg \text{Study(zhang)} \wedge \text{Lucky(zhang)}. \quad (3.62)$$

“任何幸运的人都能获奖”.

$$(\forall x)(\text{Lucky}(x) \rightarrow \text{Win}(x, \text{prize})). \quad (3.63)$$

结论“张是快乐的”的否定:

$$\neg \text{Happy(zhang)}. \quad (3.64)$$

将上述谓词公式转化为子句集如下:

- (1) $\neg \text{Pass}(x, \text{computer}) \vee \neg \text{Win}(x, \text{prize}) \vee \text{Happy}(x)$
- (2) $\neg \text{Study}(y) \vee \text{Pass}(y, z)$
- (3) $\neg \text{Lucky}(u) \vee \text{Pass}(u, v)$
- (4) $\neg \text{Study(zhang)}$
- (5) Lucky(zhang)
- (6) $\neg \text{Lucky}(w) \vee \text{Win}(w, \text{prize})$
- (7) $\neg \text{Happy(zhang)}$ (结论的否定)



图 3-6 归结树

为了进一步加深对谓词逻辑归结的理解,下面再给出一个经典的归结问题.

例 3.101 (“幸福生活”问题) 假设: 所有富足并且聪明的人都是快乐的, 那些看书的人是聪明的. 李明能看书且不贫穷, 快乐的人过着幸福的生活. 求证: 李明过着幸福的生活.

解：先定义谓词：

- $\text{Poor}(x)$ x 是不富足的；
- $\text{Smart}(x)$ x 是聪明的；
- $\text{Happy}(x)$ x 是快乐的；
- $\text{Read}(x)$ x 能看书；
- $\text{Exciting}(x)$ x 过着幸福的生活。

再将问题用谓词表示为：“所有不贫穷并且聪明的人都是快乐的”。

$$(\forall x)((\neg \text{Poor}(x) \wedge \text{Smart}(x)) \rightarrow \text{Happy}(x)). \quad (3.65)$$

“那些看书的人是聪明的”

$$(\forall y)(\text{Read}(y) \rightarrow \text{Smart}(y)). \quad (3.66)$$

“李明能看书且不贫”。

$$\text{Read}(\text{Liming}) \wedge \neg \text{Poor}(\text{Liming}). \quad (3.67)$$

“快乐的人过着幸福的生活”。

$$(\forall z)(\text{Happy}(z) \rightarrow \text{Exciting}(z)). \quad (3.68)$$

目标“李明过着幸福的生活”的否定。

$$\neg \text{Exciting}(\text{Liming}). \quad (3.69)$$

将上述谓词公式转化为子句集如下：

- (1) $\text{Poor}(x) \vee \neg \text{Smart}(x) \vee \text{Happy}(x)$;
- (2) $\neg \text{Read}(y) \vee \text{Smart}(y)$;
- (3) $\text{Read}(\text{Liming})$;
- (4) $\neg \text{Poor}(\text{Liming})$;
- (5) $\neg \text{Happy}(z) \vee \text{Exciting}(z)$;
- (6) $\neg \text{Exciting}(\text{Liming})$. (结论的否定)



图 3-7 归结树

3.6 归结演绎推理的归结策略

归结演绎推理实际上就是从子句集中不断寻找可进行归结的子句对，并通过对这些子句对的归结，最终得出一个空子句的过程。由于事先并不知道哪些子句对可进行归结，更不知道通过对哪些子句对的归结能尽快得到空子句，因此就需要对子句集中的所有子句逐对进行比较，直到得出空子句为止。这种盲目的全面进行归结的方法，不仅会产生许多无用的归结式，更严重的是会产生组合爆炸问题。因此，需要研究有效的归结策略来解决这些问题。

目前，常用的归结策略可分为两大类，删除策略和限制策略。

- ▶ 删除策略是通过删除某些无用的子句来缩小归结范围；
- ▶ 限制策略是通过对参加归结的子句进行某些限制，来减少归结的盲目性，以尽快得到空子句。

为了说明选择归结策略的重要性，在讨论各种常用的归结策略之前，还是先提一下广度优先策略。

3.6.1 广度优先策略

广度优先是一种穷尽子句比较的复杂搜索方法。设初始子句集为 S_0 ，广度优先策略的归结过程可描述如下：

(1) 从 S_0 出发, 对 S_0 中的全部子句作所有可能的归结, 得到第一层归结式, 把这些归结式的集合记为 S_1 ;

(2) 用 S_0 中的子句与 S_1 中的子句进行所有可能的归结, 得到第二层归结式, 把这些归结式的集合记为 S_2 ;

(3) 用 S_0 和 S_1 中的子句与 S_2 中的子句进行所有可能的归结, 得到第三层归结式, 把这些归结式的集合记为 S_3 ; 如此继续, 直到得出空子句或不能再继续归结为止.

例 3.102 设有如下子句集:

$$S = \{\neg I(x) \vee R(x), I(a), \neg R(y) \vee L(y), \neg L(a)\}. \quad (3.70)$$

用宽度优先策略证明 S 为不可满足.

宽度优先策略的归结树如下:

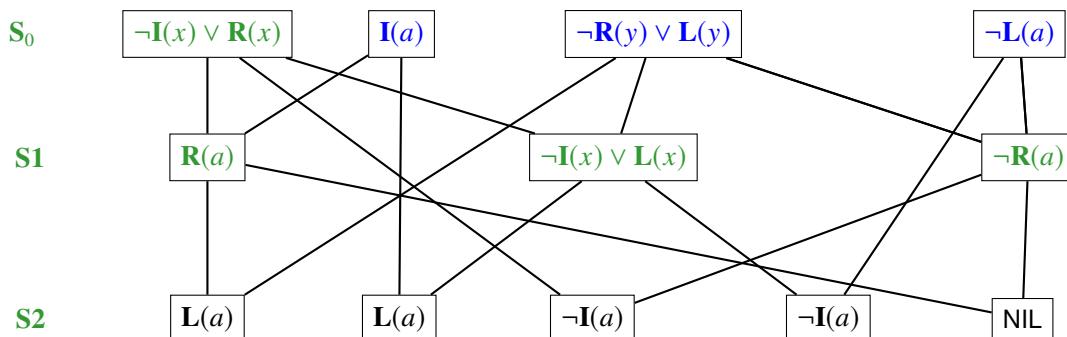


图 3-8 广义归结树

从这个例子可以看出, 宽度优先策略归结出了许多无用的子句, 既浪费时间, 又浪费空间. 但是, 这种策略在当问题有解时保证能找到最短归结路径. 因此, 它是一种完备的归结策略. 宽度优先对复杂问题的归结容易产生组合爆炸, 但对简单问题仍是一种比较好的归结策略.

3.6.2 支持集策略

支持集策略是沃斯 (Wos) 等人在 1965 年提出的一种归结策略.

注 3.103. 要求每一次参加归结的两个亲本子句中至少应该有一个子句是由目标公式的否定或它们的后裔所得到.

可以证明支持集策略是完备的, 即当子句集为不可满足时, 则由支持集策略一定能够归结出一个空子句. 可把支持集策略看成是在宽度优先策略中引入了某种限制条件, 这种限制条件代表一种启发信息, 因而有较高的效率.

例 3.104 设有如下子句集:

$$S = \{\neg I(x) \vee R(x), I(a), \neg R(y) \vee L(y), \neg L(a)\}, \quad (3.71)$$

其中, $\neg I(x) \vee R(x)$ 为目标公式的否定子句. 试用支持集策略证明 S 不可满足.

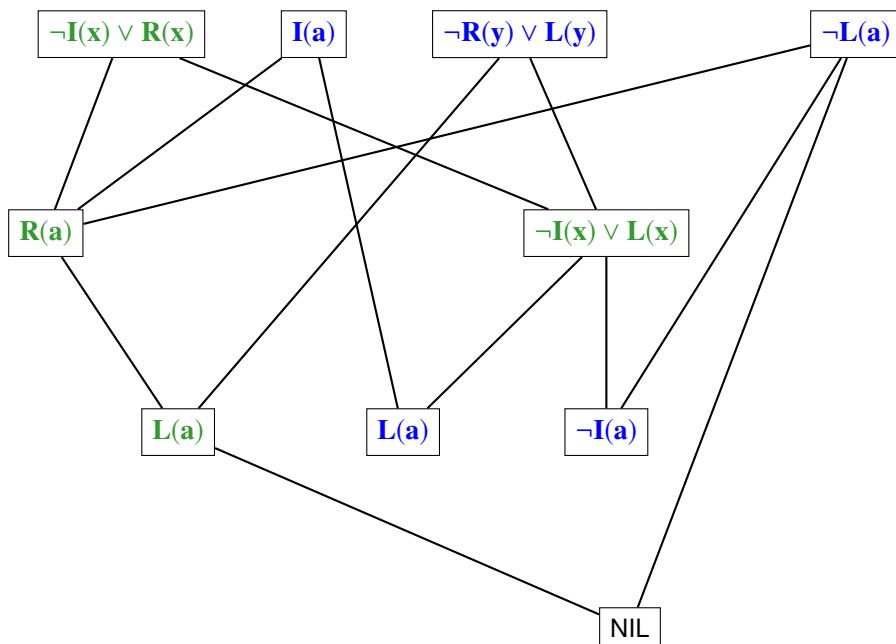


图 3-9 支持集策略

从上述使用支持集策略的归结过程可以看出, 各级归结式数目要比宽度优先策略生成的少, 但在第二级还没有空子句. 支持集策略限制了子句集元素的剧增, 但会增加空子句所在的深度. 此外, 支持集策略具有逆向推理的含义, 由于进行归结的亲本子句中至少有一个与目标子句有关, 因此推理过程可以看作是沿目标和子目标的方向前进的.

3.6.3 删除策略

在寻找可归结子句的归结过程中, 子句集中的子句越多, 需要付出的代价就会越大. 如果在归结时能把子句集中无用的子句删除掉, 这就会缩小搜索范围, 减少比较次数, 从而提高归结效率. 常用的删除方法有纯文字、重言式和包孕.

3.6.3.1 纯文字删除法

定义 3.105 纯文字

如果子句集中不存在与文字 L 互补的文字 $\neg L$, 则称该文字为纯文字.



 **注 3.106.** 在归结过程中, 纯文字不可能被消除, 用包含纯文字的子句进行归结也不可能得到空子句, 因此对包含纯文字的子句进行归结是没有意义的, 应该把它从子句集中删除.

 **注 3.107.** 对于子句集而言, 删除包含纯文字的子句, 是不影响其不可满足性的.

例 3.108 对子句集

$$S = \{P \vee Q \vee R, \neg Q \vee R, Q, \neg R\}, \quad (3.72)$$

其中 P 是纯文字, 因此可以将子句 $P \vee Q \vee R$ 从子句集 S 中删除.

3.6.3.2 重言式删除法

如果一个子句中包含有互补的文字对, 则称该子句为重言式.

例 3.109 $P(x) \vee \neg P(x)$ 和 $P(x) \vee Q(x) \vee \neg P(x)$ 都是重言式, 不管 $P(x)$ 的真值为真还是为假, $P(x) \vee \neg P(x)$ 和 $P(x) \vee Q(x) \vee \neg P(x)$ 都均为真.

重言式是真值为真的子句. 对一个子句集来说, 不管是增加还是删除一个真值为真的子句, 都不会影响该子句集的不可满足性. 因此, 可从子句集中删去重言式.

包孕删除法

定义 3.110 包孕

设有子句 C_1 和 C_2 , 如果存在一个置换 σ , 使得 $C_1\sigma \subseteq C_2$, 则称 C_1 包孕于 C_2 .



例 3.111 $P(x)$ 包孕于 $P(y) \vee Q(z)$, $\sigma = \{x/y\}$;

$P(x)$ 包孕于 $P(a)$, $\sigma = \{a/x\}$;

$P(x)$ 包孕于 $P(a) \vee Q(z)$, $\sigma = \{a/x\}$;

$P(x) \vee Q(a)$ 包孕于 $P(f(a)) \vee Q(a) \vee R(y)$, $\sigma = \{f(a)/x\}$;

$P(x) \vee Q(y)$ 包孕于 $P(a) \vee Q(u) \vee R(w)$, $\sigma = \{a/x, u/y\}$.

对子句集来说, 把其中包孕的子句删去后, 不会影响该子句集的不可满足性. 因此, 可从子句集中删除那些包孕的子句.

3.6.3.3 单文字子句策略

定义 3.112 单文字子句

只包含一个文字的子句称为单文字子句.



单文字子句策略是对支持集策略的进一步改进, 它要求每次参加归结的两个亲本子句中至少有一个子句是单文字子句.

例 3.113 设有如下子句集:

$$S = \{\neg I(x) \vee R(x), I(a), \neg R(y) \vee L(y), \neg L(a)\}. \quad (3.73)$$

用单文字子句策略证明 S 为不可满足.

采用单文字子句策略, 归结式包含的文字数将少于其亲本子句中的文字数, 这将有利于向空子句的方向发展, 因此会有较高的归结效率. 但这种策略是不完备的, 即当子句集为不可满足时, 用这种策略不一定能归结出空子句.



图 3-10 单文字子句策略

3.6.3.4 线形输入策略

这种策略要求每次参加归结的两个亲本子句中, 至少应该有一个是初始子句集中的子句. 所谓初始子句集是指开始归结时所使用的子句集.

例 3.114 设有如下子句集:

$$S = \{\neg I(x) \vee R(x), I(a), \neg R(y) \vee L(y), \neg L(a)\}. \quad (3.74)$$

用线性输入策略证明 S 为不可满足.

线性输入策略可限制生成归结式的数目, 具有简单和高效的优点. 但是, 这种策略也是一种不完备的策略.

例 3.115 子句集

$$S = \{Q(u) \vee P(a), \neg Q(w) \vee P(w), \neg Q(x) \vee \neg P(x), Q(y) \vee \neg P(y)\}. \quad (3.75)$$

从 S 出发很容易找到一棵归结反演树, 但不存在线性输入策略的归结反演树.



图 3-11 线形输入策略

3.6.3.5 祖先过滤策略

这种策略与线性输入策略有点相似, 但是, 放宽了对子句的限制. 每次参加归结的两个亲本子句, 只要满足以下两个条件中的任意一个就可进行归结:

- (1) 两个亲本子句中至少有一个是初始子句集中的子句.
- (2) 如果两个亲本子句都不是初始子句集中的子句, 则一个子句应该是另一个子句的先辈子句.

 **注 3.116.** 所谓一个子句(例如 C_1)是另一个子句(例如 C_2)的先辈子句是指 C_2 是由 C_1 与别的子句归结后得到的归结式.

例 3.117 设有如下子句集:

$$S = \{\neg Q(x) \vee \neg P(x), Q(y) \vee \neg P(y), \neg Q(w) \vee P(w), Q(a) \vee P(a)\}. \quad (3.76)$$

用祖先过滤策略证明 S 为不可满足.

证明. 从 S 出发, 按祖先过滤策略归结过程如图3-12所示.



图 3-12 祖先过滤策略

证毕 □

上面分别讨论了几种基本的归结策略, 但在实际应用中, 还可以把几种策略结合起来使用. 总之, 在选择归结反演策略时, 主要应考虑其完备性和效率问题.

3.6.4 用归结反演求取问题的答案

归结原理除了可用于定理证明外, 还可用来求取问题答案, 其思想与定理证明相似. 其一般步骤为:

- (1) 把问题的已知条件用谓词公式表示并化为相应的子句集 S ;
- (2) 把目标问题的否定用谓词公式表示并化为子句集;

(3) 对目标否定子句集中的每个子句, 构造该子句的重言式, 即把该目标否定子句和此目标否定子句的否定之间再进行析取, 得到新的子句, 用这些重言式代替相应的目标否定子句式, 并把这些重言式加入到前提子句集中, 得到一个新的子句集;

(4) 对这个新的子句集, 应用归结原理求出其证明树, 这时证明树的根子句不为空, 称这个证明树为修改的证明树;

(5) 用修改证明树的根子句作为回答语句, 则答案就在此根子句中.

下面通过一个例子来说明如何求取问题的答案.

例 3.118 已知: “张和李是同班同学, 如果 x 和 y 是同班同学, 则 x 的教室也是 y 的教室, 现在张在 302 教室上课.” 问: “现在李在哪个教室上课?”

解: 首先定义谓词:

$C(x, y)$: x 和 y 是同班同学;

$At(x, u)$: x 在 u 教室上课.

把已知前提用谓词公式表示如下:

$C(zhang, li)$;

$(\forall x)(\forall y)(\forall u)(C(x, y) \wedge At(x, u) \rightarrow At(y, u))$;

$At(zhang, 302)$.

把目标的否定用谓词公式表示为: $\neg(\exists v)At(li, v)$.

把上述公式化为子句集:

$\{C(zhang, li), \neg C(x, y) \vee \neg At(x, u) \vee At(y, u), At(zhang, 302)\}$.

把目标的否定化成子句式, 并用重言式

$\neg At(li, v) \vee At(li, v)$.

代替之.

把此重言式加入前提子句集中, 得到一个新的子句集, 对这个新的子句集, 应用归结原理求出其证明树. 其求解过程如图 3-13 所示.

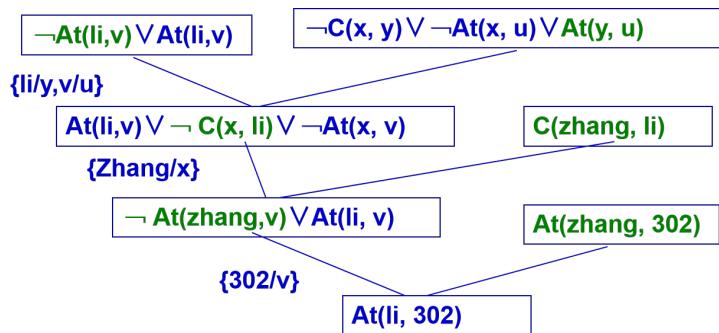


图 3-13 祖先过滤策略

该证明树的根子句就是所求的答案, 即“李明在 302 教室”.

3.7 基于规则的演绎推理

上一节讨论了归结演绎推理, 它需要把谓词公式化为子句形, 尽管这种转化在逻辑上是等价的, 但是原来蕴含在谓词公式中的一些重要信息却会在求取子句形的过程中被丢失.

例 3.119 蕴含式

$$\neg A \wedge \neg B \rightarrow C, \neg A \wedge \neg C \rightarrow B, \neg A \rightarrow B \vee C, \neg B \rightarrow A \vee C, \quad (3.77)$$

都与子句 $A \vee B \vee C$ 等价.

但在 $A \vee B \vee C$ 中, 是根本得不到原逻辑公式中所蕴含的那些超逻辑含义的. 况且, 在不少情况下, 人们多希望使用那种接近于问题原始描述的形式来进行求解, 而不希望把问题描述化为子句集. 规则是一种比较接近于人们习惯的问题描述方式:

定义 3.120 演绎系统

按照规则对问题进行描述和求解的系统称为基于规则的系统, 或者叫做基于规则的演绎系统.



 **注 3.121.** 规则演绎系统的推理可分为正向演绎推理、逆向演绎推理和双向演绎推理三种形式.

3.7.1 规则正向演绎推理

规则正向演绎对应前面所讨论过的正向推理.

从已知事实出发, 正向使用规则, 直接进行演绎, 直至到达目标为止的一种证明方法. 一个直接演绎系统不一定比反演系统更有效, 但其演绎过程容易被人们所理解.

事实表达式的与/或形变换 在一个基于规则的正向演绎系统中, 其前提条件中的事实表达式是作为系统的初始综合数据库来描述的. 对事实的化简, 只须转换成不含蕴含符号“ \rightarrow ”的与/或形表示即可, 而不必化成子句集.

把事实表达式化为非蕴含形式的与/或形的主要步骤如下:

(1) 利用连接词化归律 “ $P \rightarrow Q \Leftrightarrow \neg P \vee Q$ ”, 消去蕴含符号. 事实上, 在事实表达式中很少含蕴含符号“ \rightarrow ”, 因为蕴含式可表示成规则.

(2) 利用狄·摩根定律及量词转换律, 把否定“ \neg ”移到紧靠谓词的位置, 直到每个否定符号的辖域最多只含一个谓词为止.

(3) 对所得到的表达式进行前束化.

(4) 对全称量词辖域内的变量进行改名和标准化, 存在量词的量化可用 Skolem 函数实现, 使不同量词约束的变元有不同的名字.

(5) 消去全称量词, 余下的变量都被认为是全程量词量化的变量.

(6) 对变量进行换名, 使主合取元具有不同的变量名.

例 3.122 有如下表达式

$$(\exists x)(\forall y)(Q(y, x) \wedge \neg(R(y) \vee P(y)) \wedge S(x, y)), \quad (3.78)$$

可把它转化为:

$$Q(z, a) \wedge ((\neg R(y) \wedge \neg P(y)) \vee \neg S(a, y)). \quad (3.79)$$

这就是与/或形表示.

事实表达式的与/或形变换 事实表达式的与/或形可用一棵与/或树表示出来.

例 3.123 对上例所给出的与/或式, 可用如下与/或树来表示.

在图3-14中, 每个节点表示该事实表达式的一个子表达式, 子表达式之间的与和或关系规定如下:

- 当某个表达式为 k 个子表达式的析取时, 例如 $E_1 \vee E_2 \vee \cdots \vee E_k$, 其中的每个子表达式 E_i 均被表示为 $E_1 \vee E_2 \vee \cdots \vee E_k$ 的后继节点, 并由一个 k 线连接符将这些后继节点都连接到其父节点, 即表示成“与”的关系.

- 当某个表达式为 k 个子表达式的合取时, 例如 $E_1 \wedge E_2 \wedge \cdots \wedge E_k$, 其中的每个子表

达式 E_i 均被表示成 $E_1 \wedge E_2 \wedge \dots \wedge E_k$ 的一个单一的后继节点，并由一个单一连接符连接到其父节点，即表示成“或”的关系。



图 3-14 与或树

与/或树的根节点就是整个事实表达式，端节点均为事实表达式中的一个文字。有了与/或树的表示，就可以求出其解树（结束于文字节点上的子树）集。并且可以发现，事实表达式的子句集与解树集之间存在着一一对应关系，即解树集中的每个解树都对应着子句集中的一个子句。其对应方式为：解树集中每个解树的端节点上的文字的析取就是子句集中的一子句。

上图所示的与/或树有 3 个解树，分别对应以下 3 个子句：

$$\begin{aligned} & Q(z, a); \\ & \neg R(y) \vee \neg S(a, y); \\ & \neg P(y) \vee \neg S(a, y). \end{aligned}$$

利用与/或树的这个性质，可以把与/或树看作是对子句集的简洁表示。不过，表达式的与/或树表示要比子句集表示的通用性差一些，原因是与/或树中的合取元没有进一步展开，因此不能象在子句形中那样对某些变量进行改名，这就限制了与/或树表示的灵活性。

例 3.124 上面的最后一个子句，在子句集中其变量 y 可全部改名为 x ，但却无法在与/或树中加以表示，因而失去了通用性，并且可能带来一些困难。

注 3.125. 这里的与/或树是作为综合数据库的一种表示，其中的变量受全称量词的约束。

而在第二章可分解产生式系统中, 所描述的与/或树则是搜索过程的一种表示, 两者有着不同的目的和含义, 应用时应加以区分.

规则的与/或形变换 与/或形正向演绎系统中是以正向方式使用规则 (F 规则) 对综合数据库中的与/或树结构进行变换的. 为简化这种演绎过程, 通常要求 F 规则应具有如下形式:

$$L \rightarrow W, \quad (3.80)$$

其中, L 为单文字, W 为与/或形公式. 假定出现在蕴含式中的任何变量全都受全称量词的约束, 并且这些变量已经被换名, 使得他们与事实公式和其他规则中的变量不同.

 **注 3.126.** 如果领域知识的规则表示形式与上述要求不同, 则应将它转换成要求的形式. 其变换步骤如下:

(1) 暂时消去蕴含符号 “ \rightarrow ” . 设有如下公式:

$$(\forall x)((\exists y)(\forall z)P(x, y, z)) \rightarrow (\forall u)Q(x, u). \quad (3.81)$$

运用等价关系 “ $P \rightarrow Q \Leftrightarrow \neg P \vee Q$ ” , 可将上式变为:

$$(\forall x)(\neg(\exists y)(\forall z)P(x, y, z)) \vee (\forall u)Q(x, u). \quad (3.82)$$

(2) 把否定符号 “ \neg ” 移到紧靠谓词的位置上, 使其作用域仅限于单个谓词. 通过使用狄. 摩根定律及量词转换率可把上式转换为:

$$(\forall x)((\forall y)(\exists z)\neg P(x, y, z)) \vee (\forall u)Q(x, u). \quad (3.83)$$

(3) 引入 Skolem 函数, 消去存在量词. 消去存在量词后, 上式可变为:

$$(\forall x)((\forall y)(\neg P(x, y, f(x, y))) \vee (\forall u)Q(x, u)). \quad (3.84)$$

化成前束式, 消去全部全称量词. 消去全称量词后, 上式变为:

$$\neg P(x, y, f(x, y)) \vee Q(x, u), \quad (3.85)$$

此公式中的变元都被视为受全称量词约束的变元.

(4) 恢复蕴含式表示. 利用等价关系 “ $\neg P \vee Q \Leftrightarrow P \rightarrow Q$ ” 将上式变为:

$$P(x, y, f(x, y)) \rightarrow Q(x, u). \quad (3.86)$$

 **注 3.127.** 在上述对 F 规则的要求中, 之所以限制其前件为单文字, 是因为在进行正向演绎推理时要用 F 规则作用于表示事实的与/或树, 而该与/或树的叶节点都是单文字, 这样就可用 F 规则的前件与叶节点进行简单匹配.

对非单文字情况, 若形式为 $L_1 \vee L_2 \rightarrow W$, 则可将其转换成与之等价的两个规则 $L_1 \rightarrow W$ 与 $L_2 \rightarrow W$ 进行处理.

目标公式的表示形式 与/或树正向演绎系统要求目标公式用子句形表示.

目标公式不是子句形, 则需要化成子句形. 把一个目标公式转化为子句形的步骤与第 3 节所述的化简子句形的步骤类似, 但 Skolem 化的过程不同. 目标公式的 Skolem 化过程是化简子句形的 Skolem 过程的对偶形式, 即把目标公式中属于存在量词辖域内的全称变量用存在量词量化变量的 Skolem 函数来代替, 使经 Skolem 化目标公式只剩下存在量词, 然后再对析取元作变量替换, 最后把存在量词消掉.

例 3.128 设目标公式为

$$(\exists y)(\forall x)(P(x, y) \vee Q(x, y)). \quad (3.87)$$

用 Skolem 函数消去全称量词后有

$$(\exists y)(P(f(y), y) \vee Q(f(y), y)), \quad (3.88)$$

进行变量换名, 使每个析取元具有不同的变量符号, 于是有

$$(\exists y)P(f(y), y) \vee (\exists z)Q(f(z), z). \quad (3.89)$$

最后, 消去存在量词得

$$P(f(y), y) \vee Q(f(z), z). \quad (3.90)$$

这样, 目标公式中的变量都假定受存在量词的约束.

规则正向演绎系统 规则正向演绎推理过程是从已知事实出发, 不断运用 F 规则, 推出欲证明目标公式的过程. 即先用与/或树把已知事实表示出来, 然后再用 F 规则的前件和与/或树的叶节点进行匹配, 并通过一个匹配弧把匹配成功的 F 规则加入到与/或树中, 依此使用 F 规则, 直到产生一个含有以目标节点为终止节点解图为止.

下面分别在命题逻辑和谓词逻辑的情况下讨论规则正向演绎过程.

命题逻辑的规则演绎过程 由于命题逻辑中的公式不含变元, 因此其规则演绎过程比较简单.

例 3.129 设已知事实为: $A \vee B$, F 规则为:

$$r_1 : A \rightarrow C \wedge D; \quad (3.91)$$

$$r_2 : B \rightarrow E \wedge G. \quad (3.92)$$

目标公式为: $C \vee G$.

证明. 先将已知事实用与/或树表示出来, 然后再用匹配弧把 r_1 和 r_2 分别连接到事实与/或树中与 r_1 和 r_2 前件匹配的两个不同端节点, 由于出现了以目标节点为终节点的解树, 故推理过程结束. 这一证明过程可用图3-15表示. 在该图中, 双箭头表示匹配弧, 它相当于一个单线连接符.



图 3-15 与或树

□

为了验证上述推理的正确性, 下面再用归结演绎推理给予证明. 由已知事实、规则及目标的否定可得到如下子句集:

$$\{A \vee B, \neg A \vee C, \neg A \vee D, \neg B \vee E, \neg B \vee G, \neg C, \neg G\}. \quad (3.93)$$

其归结过程如图3-16所示.

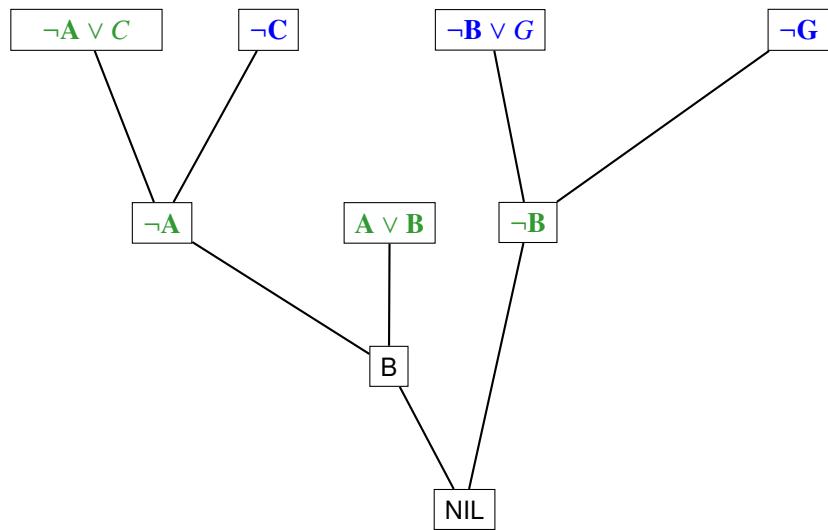


图 3-16 与或树

可见,用归结演绎推理对已知事实和 F 规则集目标的否定所过程的子句集进行归结,得到了空子句 NIL,从而证明了目标公式. 它与正向演绎推理所得到的结果是一致的.

谓词逻辑情况 由于事实、 F 规则及目标中均含有变元,因此,其规则演绎过程还需要用最一般合一对变进行置换.

例 3.130 设已知事实的与/或形表示为:

$$P(x, y) \vee (Q(x) \wedge R(v, y)),$$

$$F \text{ 规则为: } P(u, v) \rightarrow (S(u) \vee N(v)),$$

$$\text{目标公式为: } S(a) \vee N(b) \vee Q(c).$$

其推理过程如下图3-17所示.



图 3-17 与或树

3.7.2 规则逆向演绎推理

规则逆向演绎推理过程是从目标公式的与/或树出发, 反向使用规则 (B 规则), 对目标公式的与/或树进行变换, 直到得与有事实节点一致的解图为止. 包括:

- ▶ 目标公式的与/或形变换.
- ▶ 目标公式的与/或树表示.
- ▶ B 规则的表示形式.
- ▶ 规则逆向演绎推理过程.

逆向系统中的目标公式采用与/或形表示, 其化简采用与正向系统中对事实表达式处理的对偶形式. 即要用存在量词约束变元的 Skolem 函数来替换由全称量词约束的相应变元, 消去全称量词, 再消去存在量词, 并进行变元换名, 使主析取元之间具有不同的变元名.

例 3.131 有如下目标公式:

$$(\exists y)(\forall x)(P(x) \rightarrow (Q(x) \wedge \neg(R(x) \wedge S(y)))). \quad (3.94)$$

Skolem 化后为

$$\neg P(f(y)) \vee (Q(f(y), y) \wedge (\neg R(f(y)) \vee \neg S(y))). \quad (3.95)$$

变元换名后为

$$\neg P(f(z)) \vee (Q(f(y), y) \wedge (\neg R(f(y)) \vee \neg S(y))). \quad (3.96)$$

目标公式的与/或形也可用与/或树表示出来, 其表示方法与正向演绎推理中事实的与或树表示略有不同. 它规定:

- ▶ 子表达式之间的析取关系用单一连接符连接, 表示为或的关系;
- ▶ 子表达式之间的合取关系则用 k 线连接符连接, 表示为与的关系.

例 3.132 对上述目标公式的与/或形, 可用如下的与/或树如图3-18表示.

在目标公式的与/或树中, 若把叶节点用它们之间的合取及析取关系连接起来, 就可得到原目标公式的三个子目标:

$\neg P(f(z));$
 $Q(f(y), y) \wedge \neg R(f(y));$
 $Q(f(y), y) \wedge \neg S(y).$ 即: 子目标是文字的合取式.



图 3-18 与或树

3.8 知识图谱 (knowledge graph) 的一阶归纳推理

知识图谱由向图构成构建, 用来描述现实世界中实体及实体间的关系, 是人工智能中进行知识表达的重要方式. 因果推理是一种重要的推理手段, 是人类智能的重要组成. 本小节简单介绍基于路径排序的知识图谱推理、基于结构因果模型 (structural causal model, SCM) 和因果图 (causal diagram) 的知识图谱推理, 以及辛普森悖论、D-分离和 do 算子等概念.

注 3.133. 哲学上把现象和现象之间那种“引起和被引起”的关系, 叫做因果关系, 其中引起某种现象产生的现象叫做原因, 被某种现象引起的现象叫做结果.

知识图谱推理的主要方法有:

- 1) 基于描述逻辑的推理, 如 DL-based.
- 2) 基于图结构和统计规则挖掘的推理, 如 PRA 和 AMIE.
- 3) 基于知识图谱表示学习的推理, 如 TransE.
- 4) 基于概率逻辑的方法, 如 Statistical Relational Learning.

3.8.1 基于符号逻辑的推理——本体推理

与传统的符号逻辑推理有关的推理手段是基于描述逻辑的本体推理. 描述逻辑主要被用来对事物的本体进行建模和推理, 描述和推断概念分类及其概念之间的关系.

描述逻辑的主要方法有:

- 1) 基于表运算 Tableaux 及改进的方法: FaCT++、Racer 和 Pellet Hermit 等运算.

- 2) 基于 Datalog 转换的方法如 KAON 和 RDFox 等.
- 3) 基于产生式规则的算法(如 rete): Jena、Sesame 和 OWLIM 等.

3.8.2 基于图结构和统计规则挖掘的推理

图结构和统计规则挖掘推理的主要方法:

- 1) 基于路径排序学习方法有 PRA 和 Path ranking Algorithm.
- 2) 基于关联规则挖掘方法有 AMIE.

3.8.3 知识图谱推理(路径排序算法)

3.8.4 基于路径排序学习方法有 PRA 和 Path ranking Algorithm



图 3-19 图结构和统计规则挖掘的推理

3.8.5 斯坦福的七步法知识图谱构建步骤

斯坦福大学医学院开发的七步法, 主要用于领域本体的构建. 七个步骤分别是:

- ① 确定本体的专业领域和范畴;
- ② 考查复用现有本体的可能性;
- ③ 列出本体中的重要术语;
- ④ 定义类和类的等级体系 (完善等级体系可行的方法有: 自顶向下法、自低向上法和综合法);
- ⑤ 定义类的属性;

⑥ 定义属性的分面;

⑦ 创建实例.

构建领域本体的步骤:

1) 确定领域本体的专业领域和范畴

2) 考虑复用现有的本体

3) 列出本体涉及领域中的重要术语

4) 定义分类概念和概念分类层次

5) 定义概念之间的关系

本体构成五元素 (建模元语)

① 类 (Classes) 或概念 (Concepts);

② 关系 (Relations);

③ 函数 (Functions);

④ 公理 (Axioms);

⑤ 实例 (Instances)

知识图谱的构建的流程:

1. 数据来源 (数据层 (Data Level) 的构建)

百科类数据 (Wikipedia 半结构化, Freebase 结构化), 结构化数据 (DBpedia 和 YAGO 等通用语义数据集, 还包括如 MusicBrainz 和 DrugBank 等特定领域的知识库), 半结构化数据, 自动化的 AVP(属性-值对) 抽取以及搜索日志挖掘, 发现最新出现的各种实体, 基于 Bootstrapping 的多类别协同模式学习。

Bootstrapping 方法的过程:

Given a hand of seed NEs of a category C:

Learning context features of the seeds from queries

Extracting new seed entities of category C using the learnt context features

Expanding context features using the expanded seed set

属性-值对 (attribute-value pair, 又称 AVP) 用来刻画实体的内在特性; 而关系 (relation) 用来连接两个实体, 刻画它们之间的关联

2. 从抽取图谱 (Extraction Graphs) 到知识图谱:

(1) 实体对齐 (Object Alignment), 针对多种来源数据用聚类算法, 关键在于定义合适的相似度度量

(2) 知识图谱 schema 构建, 相当于为其建立本体 (Ontology), 最基本的本体包括概念、概念层次、属性、属性值类型、关系、关系定义域 (Domain) 概念集以及关系值域 (Range) 概念集.

• 自顶向下的方式是指通过本体编辑器 (Ontology Editor) 预先构建本体, 本体构建不是从无到有的过程, 而是依赖于从百科类和结构化数据得到的高质量知识中所提取的模式信息.

• 自底向上的方式则通过上面介绍的各种抽取技术, 特别是通过搜索日志和 Web Table 抽取发现的类别、属性和关系, 并将这些置信度高的模式合并到知识图谱中. 合并过程将使用类似实体对齐的对齐算法.

(3) 不一致性的解决.

优先采用那些可靠性高的数据源 (如百科类或结构化数据) 抽取得到的事实.

3. 知识图谱的挖掘:

(1) 推理, 针对属性; 针对关系.

(2) 实体重要性排序, 当查询涉及多个实体时, 搜索引擎将选择与查询更相关且更重要的实体来展示. 实体的相关性度量需在查询时在线计算, 而实体重要性与查询无关可离线计算, 搜索引擎公司将 PageRank 算法应用在知识图谱上来计算实体的重要性.

(3) 相关实体挖掘. 使用主题模型 (如 LDA) 发现虚拟文档集中的主题分布. 其中每个主题包含 1 个或多个实体, 这些在同一个主题中的实体互为相关实体. 当用户输入查询时, 搜索引擎分析查询的主题分布并选出最相关的主题.

4. 知识图谱的更新和维护.

(1) Type 和 Collection 的关系

搜索引擎公司还通过自动化算法从各种数据源抽取新的类型信息, 如果 Collection 中的某一种类型能够长期的保留, 发展到一定程度后, 由专业的人员进行决策和命名并最终成为一种新的 Type.

(2) 结构化站点包装器的维护

搜索引擎会定期检查站点是否存在更新, 使用最新的站点包装器进行 AVP 抽取.

(3) 知识图谱的更新频率

Type 对应的实例往往是动态变化的.

(4) 众包 (Crowdsourcing) 反馈机制

用户可以对搜索结果中展现的知识卡片所列出的实体相关的事实在纠错. 当很多用户都指出某个错误时, 搜索引擎将采纳并修正.

5. 知识图谱在搜索中的应用

(1) 查询理解

搜索引擎并非展现实体的全部属性, 而是根据当前输入的查询自动选择最相关的属性及属性值来显示. 当要展现的实体被选中之后, 利用相关实体挖掘来推荐其他用户可能感兴趣的实体供进一步浏览.

(2) 问题回答

知识图谱对于搜索所带来的另一个革新是直接返回答案, 而不仅仅是排序的文档列表。搜索引擎不仅要理解查询中涉及到的实体及其属性, 更需要理解查询所对应的语义信息。搜索引擎通过高效的图搜索, 在知识图谱中查找连接这些实体及属性的子图并转换为相应的图查询——[用于 RDF 上的查询语言 SPARQL](#).

① 基于知识图谱表示学习的关系推理

将实体和关系都表示为向量, 通过向量之间的计算代替图的遍历和搜索来预测三元组的存在, 由于向量的表示已经包含了实体原有的语义信息, 计算含有一定的推理能力。可应用于链接预测和基于路径的多度查询等。

② 基于概率逻辑的方法: Statistical Relational Learning.

概率逻辑学习有时也叫 Relational Machine Learning (RML), 关注关系的不确定性和复杂性。通常使用 Bayesian networks 或者 Markov networks.

③ 基于符号逻辑的推理

本体概念推理: 图谱中基于 RDF(Resource Description Framework) 来作为资源描述语言。但是 RDF 表示关系层次受限, 因此有了 RDFS, 在 RDF 的基础上, 新增了 Class, subClassOf, type, Property, subPropertyOf 和 Domain, Range 等词汇, 可以更好的表述相关关系。

3.8.6 因果推理 (辛普森悖论、D-分离和 do 算子)

路径张量分解的知识图谱推理算法 (Knowledge Graph Reasoning Based on Paths of Tensor Factorization, 张量分解技术在知识图谱学习和推理过程中的应用) 只考虑知识图谱中实体与实体间的直接关系, 忽略知识图谱图形结构的特点, 提出基于路径张量分解的知识图谱推理算法 (PRESCAL), 利用路径排列算法 (PRA) 获得知识图谱中各实体对间的关系路径。然后对实体对间的关系路径进行张量分解, 并在优化更新过程中采用交替最小二乘法。实验表明, 在路径问题回答任务和实体链接预测任务中, PRESCAL 可以取得较好的预测准确率。

3.9 模糊逻辑 *

逻辑学是古希腊哲学家亚里士多德创立的, 至今已有几十种, 诸如, 数理逻辑、哲学逻辑、模糊逻辑、辩证逻辑、制约逻辑等等。最基本的逻辑是形式逻辑, 又称普通逻辑。形式逻辑的三个要素是概念、判断、推理, 它同时符合四条基本规律, 那就是同一律、不矛盾律、排中律和充足理由律。换句话说, 凡是满足公理体系中基本规律的思维过程就

是符合逻辑的.

定义 3.134 逻辑

逻辑是指人类进行抽象思维的合理原则, 是用概念、判断、推理和论证的方法进行思维以获取接近真理的知识的规则.



1965 年美国数学家 L. Zadeh 首先提出了 Fuzzy 集合的概念, 导致了 Fuzzy 数学的诞生. 模糊逻辑顾名思义, 就是通过类比人类处理问题的方法来表达现实世界的对象. 比如: 绳结记事. 类比逻辑是人类常用的模糊逻辑. 类比逻辑往往使用于解释描述性表达. 建立信息对称关系, 以便于在下次表达时, 可以减少信息不对称成本. 通过类比逻辑, 人们将相似属性的对象归类就产生了归纳逻辑. 归纳在一起可以发现涵盖于所有对象的普遍属性.

例 3.135 五谷杂粮.

归纳逻辑使用于发现性表达. 当人们的经验已经无法了解新的事物的时候, 归纳逻辑可以运用已有的经验, 将所有的相似特点归纳在一起, 之后发现新的属性, 成为新的经验. 人们生活在竞争中, 而决定胜利的一方往往取决于经验的多少. 经验越多, 取胜的概率越高; 经验越少, 越容易失败. 所以, 博弈逻辑也是一种模糊逻辑.

 **注 3.136.** 博弈的过程, 是使用模糊逻辑不断地模糊判断. 比如掷骰子, 谁对骰子了解得越多, 胜率可能越高.

模糊逻辑是通过不确定判断, 建立不确定的判断系统. 比如 Fuzzy 集合, 把只取 0 和 1 二值的普通集合概念推广为在 $[0, 1]$ 区间上取无穷多值的模糊集合概念, 并用“隶属度”这一概念来精确地刻画元素与模糊集合之间的关系.

 **注 3.137.** 虽然模糊集是经典集的推广, 但是模糊集间的运算并不一定满足经典集中运算律. 如 $A \cap A = A$ 在经典集中成立, 在标准理论 $[0, 1], T, SN$ 中与之对应的运算律 $T(a, a) = a$ 却不一定成立. 这主要是因为全集 X 的所有幂集构成一个布尔代数, 事实上根本不存在满足所有布尔代数中运算律的标准理论.

中国自动化学会副理事长王飞跃教授在首届世界人工智能大会——AI WORLD 2016 上, 曾专门介绍了扎德入选了 IEEE 的 AI 名人堂.

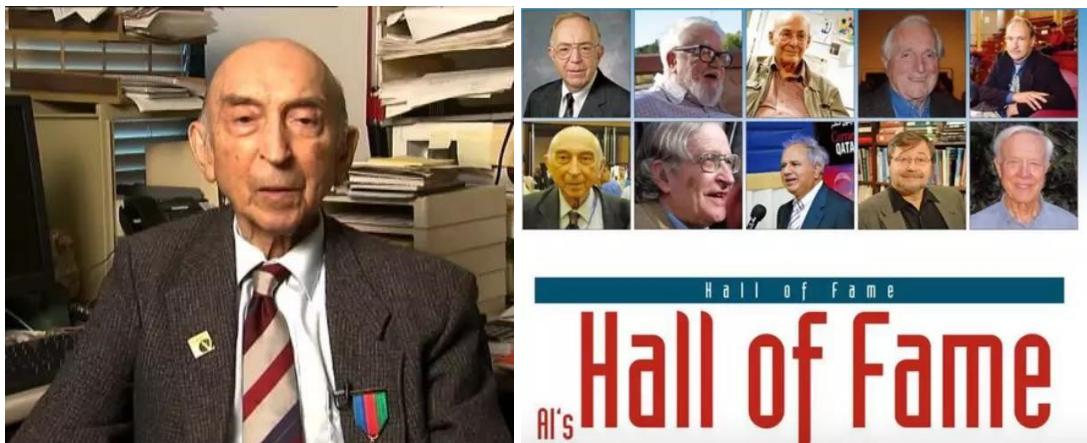


图 3-20 Zadeh 和 AI 名人堂

Zadeh 于 1973 年首次提出了基于模糊集合理论的模糊推理算法(推理合成规则近似推理方法), 即 CRI(Compositional Rule of Inference) 算法. 从此, 以模糊推理为基础的模糊控制技术被广泛地应用于工业控制领域, 并取得了举世瞩目的成功. 然而, 以 CRI 算法为代表的模糊推理却被指出缺乏严格的逻辑基础. 我国著名学者王国俊教授首次指出了 CRI 算法在逻辑语义方面的不足, 进而提出了基于逻辑语义蕴涵理论的模糊推理全蕴涵三 I 算法. 全蕴涵三 I 算法有效地改进了经典的 CRI 算法, 并将之纳入到模糊逻辑的框架之中. 从此在模糊逻辑、模糊推理和模糊控制等领域掀起了对模糊推理全蕴涵三 I 算法的研究热潮. 然而不同于 CRI 算法的是全蕴涵三 I 算法解的计算稍显复杂且对于不同类型的模糊蕴涵算子没有统一的解的形式. CRI 算法与全蕴涵三 I 算法的存在某种等价性, 可以帮助解决 CRI 算法逻辑语义不足, 全蕴涵三 I 算法没有统一解形式的问题. 文[江欢 2009 全蕴涵三 I 算法]通过将经典二值逻辑中蕴涵算子所满足的两条性质:

- 1) 有序性 (Ordering property): $x \leq y \rightarrowtail I(x, y) = 1$.
- 2) 传输律 (Law of importation): $I(T(x, y), z) = I(x, I(y, z))$.

有序性和传输律都被纳入到模糊逻辑的框架下(其中 I 为模糊蕴涵算子, T 为 T 范数, $x, y, z \in [0, 1]$), 讨论四类重要的模糊蕴涵算子, 即 S 蕴涵、剩余型蕴涵、QL 蕴涵和 D 蕴涵, 各自同时满足以上两条性质的充分必要条件. 在此基础上, 探讨了 FMP 模型和 FMT 模型下 CRI 算法与全蕴涵三 I 算法的等价性, 进而得出了 FMP 模型和 FMT 模型下全蕴涵三 I 算法解的统一形式. S 蕴涵、剩余型蕴涵、QL 蕴涵和 D 蕴涵各自同时满足有序性和传输律两条性质的充分必要条件: 证明了 FMP 模型和 FMT 模型下 CRI 算法与全蕴涵三 I 算法等价的充分条件; 证明了 FMP 模型和 FMT 模型下全蕴涵三 I 算法解的统一形式, 此形式为王国俊教授和裴道武教授所提出的全蕴涵三 I 算法统一解形式的扩展, 提出的解的形式可适用于更多类型的模糊蕴涵算子.

3.10 多值逻辑

上世纪九十年代由国内学者王国俊教授(2013年12月16日,中国共产党的优秀党员、数学家、教育家、国家有突出贡献专家,图3-21)提出的一个多值逻辑演算系统,称之为形式演绎系统 L^* (也称 $R0$ -逻辑、 NM 逻辑). 首先,回顾 Boole 代数与命题演算系统 L(经典逻辑)的主要结论;然后,介绍形式演绎系统 L^* 及其语义代数- $R0$ 代数的提出及主要结论;最后,介绍逻辑重言式在智能系统设计中的应用,相关方向有不确定性推理、非经典数理逻辑与模糊系统分析.



图 3-21 王国俊教授

3.11 Luckasiewicz 逻辑

3.12 三 I 算法的逻辑基础

3.13 强化学习

OpenAI Gym(以下简称 gym)是一个在强化学习领域内非常流行的测试框架,

```

Administrator: Windows PowerShell
版权所有 (C) Microsoft Corporation. 保留所有权利。
尝试新的跨平台 PowerShell https://aka.ms/pscore6
PS C:\Windows\system32> pip install -i https://pypi.tuna.tsinghua.edu.cn/simple gym
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting gym
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/77/48/c43b8a72b916cc70896aa431b0fc00d1491aa34e28dc55e2144f4c779
  16/cpm-0.1.7-py2.py3-none-any.whl (1.6 MB)
Requirement already satisfied: scipy in c:\users\ocnzh\anaconda3\lib\site-packages (from gym) (1.4.1)
Requirement already satisfied: numpy>=1.10.4 in c:\users\ocnzh\anaconda3\lib\site-packages (from gym) (1.16.5)
Requirement already satisfied: six in c:\users\ocnzh\anaconda3\lib\site-packages (from gym) (1.12.0)
Collecting pyglet<1.5.0,>=1.4.0
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/70/ca/20aee170afe6011e295e34b27ad7d7ccdf95faba581dd3e0f7ceec237f
  50/Pyglet-1.5.0-py2.py3-none-any.whl (1.0 MB)
Requirement already satisfied: cloudpickle<1.4.0,>=1.2.0 in c:\users\ocnzh\anaconda3\lib\site-packages (from gym) (1.2.2)
Requirement already satisfied: future in c:\users\ocnzh\anaconda3\lib\site-packages (from pyglet<1.5.0,>=1.4.0->gym) (0
)
Building wheels for collected packages: gym
  Building wheel for gym (setup.py) ... done
    Created wheel for gym: filename=gym-17.1-py3-none-any.whl size=1643717 sha256=08b3fe5402abc9e6284fb06d7e6c8ebd82b152
069f989800fe775e15c6fe9b012
  Stored in directory: c:\users\ocnzh\appdata\local\pip\cache\wheels\b5\41\94\b476339cef13ee9033a63291aa6243e51470ff82
  Successfully built gym
  Installing collected packages: pyglet, gym
  Successfully installed gym-0.17.1 pyglet-1.5.0
PS C:\Windows\system32>

```

图 3-22 gym 工具包

(Q 学习算法原理 [Yutao2011]) Q 学习是 Watkins 提出, 并由 Watkins 与 Dayan 于 1992 年基本证明其收敛性 [watkins1992q]. Q 学习利用状态-动作对值函数 $Q(s, a)$ 进行迭代, 其最优策略使得期望折扣报酬总和最大. Q 学习的值函数定义如下:

$$Q(s, a) = R(s, s', a) + \gamma \sum_{s' \in S} P(s, s', a) \max_{a \in A} Q(s', a) \quad (3.97)$$

式中 $\gamma (0 < \gamma < 1)$ 为折扣因子, 本文的 γ 值取 0.9; $P(s|s', a)$ 为状态 s 在控制动作 a 发生后转移到状态 s' 的概率; agent 会根据当前状态 s 选择某一动作 a , 得到下一状态 s' 及报酬值 $R(s, s', a)$, 然后根据其报酬值及下式迭代寻优最大 Q 值:

$$\begin{cases} Q^{k+1}(s_k, a_k) = Q^k(s_k, a_k) + \alpha [R(s_k, s_{k+1}, a_k) + \gamma \max_{a' \in A} Q^k(s_{k+1}, a_k) - Q^k(s_k, a_k)] \\ Q^{k+1}(\tilde{s}, \tilde{a}) = Q^k(\tilde{s}, \tilde{a}), \forall (\tilde{s}, \tilde{a}) \neq (s_k, a_k). \end{cases} \quad (3.98)$$

式中: Q^k 代表最优值函数 Q^* 的第 k 次迭代值; $\alpha (0 < \alpha < 1)$ 是学习因子, 它控制动作更新速度, α 值越小, 算法的搜索空间越大, 算法的稳定性越好, 本文 α 值取 0.1. Q 学习算法在当前状态下总是选择具有最高 Q 值的动作, 称为贪婪策略 π^* , 如下式:

$$\pi^* = \arg \max_{a \in A} Q^k(s, a). \quad (3.99)$$

Q 算法的动作策略选择较为关键, 合适的动作策略能提高学习的收敛速度及收敛效果. 文中采用一种基于概率分布选择动作的追踪算法 [sutton1998reinforcement] 来构造动作选择策略. 在该策略下, 初始状态各动作被选择的概率相等, 但随着动作值函数的不断迭代, 越高的 Q 值的动作被选择的概率越大, 故 Q 算法最终将收敛于 Q 矩阵代表的最优策

略。该策略概率迭代公式如下：

$$\begin{cases} P_s^{k+1}(a) = P_s^k(a_g) + \beta(1 - P_s^k(a_g)); \\ P_s^{k+1}(a) = P_s^k(a)(1 - \beta), \forall a \in A, a \neq a_g; \\ P_{\tilde{s}}^{k+1}(a) = P_{\tilde{s}}^k(a), \forall a \in A, \forall \tilde{s} \in S, \tilde{s} \neq s. \end{cases} \quad (3.100)$$

在每一状态下，对应于 Q 最大值的动作称为贪婪动作，记为 a_g 。式中 $\beta(0 < \beta < 1)$ 值的大小决定了动作搜索的速度， β 值越接近 1 说明控制动作策略越趋于贪婪策略，本文 β 值取 0.5。 $P_s^k(a)$ 代表第 k 次迭代时状态 s 下选择动作 a 的概率。在经过足够迭代次数的探索和利用之后， Q^k 将会以概率 1 收敛于最优值函数 Q^* ，最终得到一个 Q^* 矩阵表示的最优控制策略。

平均奖励 Q-Learning 算法 (Q-Learning algorithm used for average reward) [gosavi2004reinforcement].

步骤 1. 令所有 $Q(l, u) \leftarrow 0$. k = 是状态变化数. $\rho^k = 0$ 表示第 k 阶段平均奖赏估计. $W = 0$, 程序最大运行次数 k_{\max} , k_{\max} 应足够大. 从任意状态喀什系统的仿真. 选取一个特殊状态 s^* .

步骤 2. 当前状态记为 i . 以概率 $1/|A(i)|$ 选取动作, 其中 $A(i)$ 记状态 i 允许的所有动作集. 状态 i 的贪婪动作状态 i 得最高 Q -值关联.

步骤 3. 仿真动作 a . 下一状态记为 j . 令 $r(i, a, j)$ 为在动作 a 下, 从 i 到 j 的奖赏. 更新律 $Q(i, a)$ 定义为

$$Q(i, a) \leftarrow (1 - \alpha(k))Q(i, a) + \alpha(k)[r(i, a, j) - \rho^k + M_b^j], \quad (3.101)$$

其中 $M_b^j = 0$ if $j = s$ (称为 **SSP grounding**), 否则 $M_b^j = \max_{b \in A}(j)Q(j, b)$.

Step 4. 对于步骤 2 选出的贪婪动作, 按如下方式更新 W : $W \leftarrow W + r(i, a, j)$, ρ^k 按如下方式更新:

$$\rho^{k+1} = (1 - \beta(k))\rho^k + \beta(k)\frac{W}{k}. \quad (3.102)$$

Step 5. 若 $k < k_{\max}$, 令 $i \leftarrow j, k \leftarrow k + 1$, 返回步骤 2. 否则转步骤 6.

Step 6. 对于状态 i , 选取 $\mu(i) \in \arg \max_{a \in A}(i)Q(i, a)$. 解策略 μ 由算法得到, 算法停止.

3.14 作业

思考

什么是三段论推理？

思考

什么是自然演绎推理?

思考

判断下列表达式对是否可以合一? 如果可以合一, 给出最一般合一(MGU)

- 1) $P(x, b, b), P(a, y, z);$
- 2) $P(x, f(x)), P(y, y);$
- 3) $2 + 3 = x, x = 3 + 3.$

思考

对所有的 x 、 y 和 z 来说, 如果 y 是 x 父亲, z 是 y 的父亲, 则 z 是 x 的祖父; 又知每个人都有父亲, 试问是否会有这样的 X 和 Y , 使得 X 是 Y 的祖父?

4

产生式系统的搜索策略

搜索

搜索是人工智能中的一个基本问题，并与推理密切相关，搜索策略的优劣，将直接影响到智能系统的性能与推理效率。



4.1 搜索的基本概念

人工智能的任何问题的传统求解技术都离不开表示与搜索.

4.1.1 搜索的含义

定义 4.1 搜索

依靠经验, 利用已有知识, 根据问题的实际情况, 不断寻找可利用知识, 从而构造一条代价最小的推理路线, 使问题得以解决的过程称为搜索.



- ▶ 适用情况: 不良结构或非结构化问题; 难以获得求解所需的全部信息; 更没有现成的算法可供求解使用.

4.2 搜索的类型

- 按是否使用启发式信息:
 - ▶ 盲目搜索: 按预定的控制策略进行搜索, 在搜索过程中获得的中间信息并不改变控制策略.
 - ▶ 启发式搜索: 在搜索中加入了与问题有关的启发性信息, 使得搜索朝着正确的方向进行, 加速问题的求解过程, 迅速找到最优解.
- 按问题的表示方式:
 - ▶ 状态空间搜索是用状态空间法来求解问题而进行的搜索.
 - ▶ 与或树搜索是用问题归约法来求解问题而进行的搜索.

4.2.1 状态空间法

状态空间表示方法

- ▶ 状态 (State): 是表示问题求解过程中每一步问题状况的数据结构, 它可表示为:

$$S_k = \{S_{k0}, S_{k1}, \dots\}, \quad (4.1)$$

当对每一个分量都给一确定的值时, 就得到了一个具体的状态.

- ▶ 操作也称为算符, 它是把问题从一种状态变换为另一种状态的手段. 操作可以是一个机械步骤, 一个运算, 一条规则或一个过程. 操作可理解为状态集合上的一个函数, 它描述了状态之间的关系.

- ▶ 状态空间: 描述一个问题的全部状态以及这些状态之间的相互关系. 常用三元组表示为:

$$(S, F, G), \quad (4.2)$$

其中, S 为问题的所有初始状态集合; F 为操作集合; G 为目状态集合.

 **注 4.2.** 状态空间也可用一个赋值的有向图 G 来表示, 该有向图称为状态空间图. 在状态空间图中, 节点表示问题的状态, 有向边表示操作.

状态空间问题求解 求解问题的基本过程:

- ▶ 首先为问题选择适当的“状态”及“操作”的形式化描述方法;
- ▶ 然后从某个初始状态出发, 每次使用一个“操作”, 递增地建立起操作序列, 直到达到目标状态为止; 此时, 由初始状态到目标状态所使用的算符序列就是问题的一个解.

例 4.3 二阶梵塔问题. 设有三根钢针, 它们的编号分别是 1 号、2 号和 3 号. 在初始情况下, 1 号钢针上穿有 A, B 两个金片, A 比 B 小, A 位于 B 的上面. 要求把这两个金片全部移到另一根钢针上, 而且规定每次只能移动一个金片, 任何时刻都不能使大的位于小的上面.

解: 设用 $S_k = (S_{k0}, S_{k1})$ 表示问题的状态, 其中, S_{k0} 表示金片 A 所在的钢针号, S_{k1} 表示金片 B 所在的钢针号. 全部可能的问题状态共有以下 9 种:

$$\begin{aligned} S_0 &= (1, 1) & S_1 &= (1, 2) & S_2 &= (1, 3) & S_3 &= (2, 1) & S_4 &= (2, 2) \\ S_5 &= (2, 3) & S_6 &= (3, 1) & S_7 &= (3, 2) & S_8 &= (3, 3). \end{aligned} \quad (4.3)$$

- ▶ 问题的初始状态集合为 $S = \{S_0\}$.

- ▶ 目标状态集合为 $G = \{S_4, S_8\}$.

初始状态 S_0 和目标状态 S_4 和 S_8 , 如图 4-1 所示

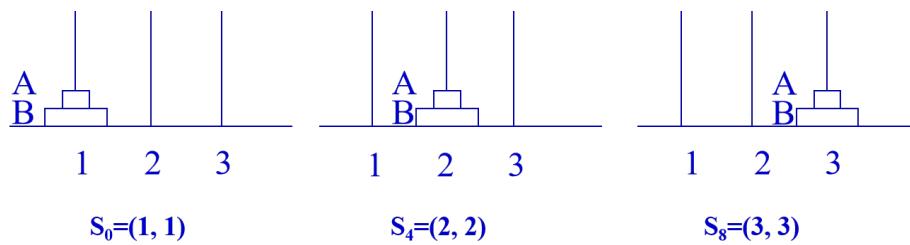


图 4-1 二阶梵塔问题的初始状态和目标状态

操作分别用 $A(i, j)$ 和 $B(i, j)$ 表示, 意义如下:

- $A(i,j)$ 表示把金片 A 从第 i 号钢针移到 j 号钢针上;
- $B(i,j)$ 表示把金片 B 从第 i 号钢针移到第 j 号钢针上. 共有 12 种操作, 它们分别是:

$$\begin{aligned} & A(1,2) \ A(1,3) \ A(2,1) \ A(2,3) \ A(3,1) \ A(3,2), \\ & B(1,2) \ B(1,3) \ B(2,1) \ B(2,3) \ B(3,1) \ B(3,2). \end{aligned} \quad (4.4)$$

根据上述 9 种可能的状态和 12 种操作, 可构成二阶梵塔问题的状态空间图, 如图 4-2 所示. 从初始节点 $(1, 1)$ 到目标节点 $(2, 2)$ 及 $(3, 3)$ 的任何一条路径都是问题的一个解. 其中, 最短的路径长度是 3, 它由 3 个操作组成.

例 4.4 从 $(1,1)$ 开始, 通过使用操作 $A(1,3), B(1,2)$ 及 $A(3,2)$, 可到达 $(3,3)$.

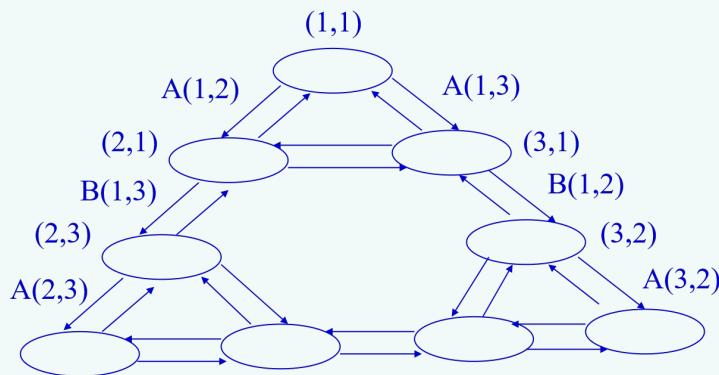


图 4-2 二阶梵塔的状态空间图

思考

用 matlab 或 Python 语言实现图 4-2 的二阶梵塔问题的程序, 并绘制流程图.

例 4.5 修道士 (Missionaries) 和野人 (Cannibals) 问题 (简称 M-C 问题). 设在河的一岸有三个野人、三个修道士和一条船, 修道士想用这条船把所有的人运到河对岸, 但受以下条件约束:

- 修道士和野人都会划船, 但每次船上至多可载两个人;
- 在河的任一岸, 如果野人人数超过修道士数, 修道士会被野人吃掉.

如果野人会服从任何一次过河安排, 请规划一个确保修道士和野人都能过河, 且没有修道士被野人吃掉的安全过河计划.

解: 首先选取描述问题状态的方法. 在这个问题中, 需要考虑两岸的修道士人数和野人

数, 还需要考虑船在左岸还是在右岸. 从而可用一个三元组来表示相应状态

$$S = (m, c, b),$$

其中, m 表示左岸的修道士人数, c 表示左岸的野人数, b 表示左岸的船数. 右岸的状态可由下式确定:

- ▶ 右岸修道士数 $m' = 3 - m$.
- ▶ 右岸野人数 $c' = 3 - c$.
- ▶ 右岸船数 $b' = 1 - b$.

在这种表示方式下, $m \in \{0, 1, 2, 3\}$ 和 $c \in \{0, 1, 2, 3\}$, $b \in \{0, 1\}$ 可取其中之一. 因此, 三元组共有 $4 \times 4 \times 2 = 32$ 种状态.

这 32 种状态并非全有意义, 除去不合法状态和修道士被野人吃掉的状态, 有意义的状态只有 16 种:

$$\begin{array}{lll} S_0 = (3, 3, 1) & S_1 = (3, 2, 1) & S_2 = (3, 1, 1) \\ S_4 = (1, 1, 1) & S_5 = (0, 3, 1) & S_6 = (0, 2, 1) \\ S_8 = (3, 2, 0) & S_9 = (3, 1, 0) & S_{10} = (3, 0, 0) \\ S_{12} = (1, 1, 0) & S_{13} = (0, 2, 0) & S_{14} = (0, 1, 0) \end{array} \quad \begin{array}{l} S_3 = (2, 2, 1) \\ S_7 = (0, 1, 1) \\ S_{11} = (2, 2, 0) \\ S_{15} = (0, 0, 0). \end{array}$$

有了这些状态, 还需要考虑可进行的操作.

定义 4.6 操作

是指用船把修道士或野人从河的左岸运到右岸, 或从河的右岸运到左岸.



每个操作都应当满足如下条件:

- ▶ 一是船至少有一个人 (m 或 c) 在操作, 离开岸边的 m 和 c 的减少数目应该等于到达岸边的 m 和 c 的增加数目;
- ▶ 二是每次操作下的船上人数不得超过 2 个;
- ▶ 三是操作应保证不产生非法状态.

因此, 操作应由条件部分和动作部分:

- ▶ 动作刻划了应用此操作所产生的结果,
- ▶ 条件只有当其条件具备时才能使用.

 **注 4.7.** 符号 P_{ij} 表示从左岸到右岸的运人操作, 符号 Q_{ij} 表示从右岸到左岸的操作, 其中 i 表示船上的修道士人数, j 表示船上的野人数. 本问题的操作集由 10 种操作组成:

$$F = \{P_{01}, P_{10}, P_{11}, P_{02}, P_{20}, Q_{01}, Q_{10}, Q_{11}, Q_{02}, Q_{20}\}. \quad (4.5)$$

表 4-1 操作的条件和动作表

操作符号	条件	动作
P_{01}	$b = 1, m = 0$ 或 $3, c \geq 1$	$b = 0, c = c - 1$
Q_{01}	$b = 0, m = 0$ 或 $3, c \leq 2$	$b = 1, c = c + 1$

下面以 P_{01} 和 Q_{01} 为例来说明这些操作的条件和动作, 如表4-1.

例 4.8 猴子摘香蕉问题. 在讨论谓词逻辑知识表示时, 我们曾提到过这一问题, 现在用状态空间法来解决这一问题.

解: 问题的状态可用 4 元组 (w, x, y, z) 表示, 其中:

- ▶ w 表示猴子的水平位置;
- ▶ x 表示箱子的水平位置;
- ▶ y 表示猴子是否在箱子上, 当猴子在箱子上时, $y = 1$, 否则 $y = 0$;
- ▶ z 表示猴子是否拿到香蕉, 当拿到香蕉时 $z = 1$, 否则 $z = 0$.

所有可能的状态

- ▶ $S_0 : (a, b, 0, 0)$; 初始状态;
- ▶ $S_1 : (b, b, 0, 0)$;
- ▶ $S_2 : (c, c, 0, 0)$;
- ▶ $S_3 : (c, c, 1, 0)$;
- ▶ $S_4 : (c, c, 1, 1)$; 目标状态.

允许的操作为

- ▶ Goto(u): 猴子走到位置 u , 即 $(w, x, 0, 0) \rightarrow (u, x, 0, 0)$.
- ▶ Pushbox(v): 猴子推着箱子到水平位置 v , 即 $(x, x, 0, 0) \rightarrow (v, v, 0, 0)$.
- ▶ Climbbox: 猴子爬上箱子, 即 $(x, x, 0, 0) \rightarrow (x, x, 1, 0)$.
- ▶ Grasp: 猴子拿到香蕉, 即 $(c, c, 1, 0) \rightarrow (c, c, 1, 1)$.

这个问题的状态空间图如图4-3所示. 不难看出, 由初始状态变为目标状态的操作序列为:

$$\{\text{Goto}(b), \text{Pushbox}(c), \text{Climbbox}, \text{Grasp}\}.$$

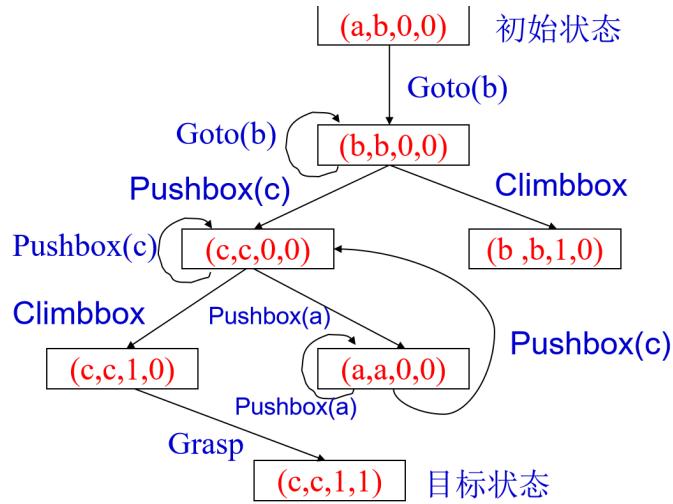


图 4-3 猴子摘香蕉问题的状态空间图

解序列为: { $\text{Goto}(b)$, $\text{Pushbox}(c)$, Climbbox , Grasp }.

4.2.2 问题归约法

4.2.2.1 问题的分解与等价变换

当一个问题比较复杂时, 可通过分解或变换, 将其转化为一系列较简单的子问题, 然后通过对这些子问题的求解, 来实现对原问题的求解.

问题的分解 如果一个问题 P 可以归约为一组子问题 P_1, P_2, \dots, P_n , 并且只有当所有子问题 P_i 都有解时, 原问题 P 才有解; 任何一个子问题 P_i 无解, 都会导致原问题 P 无解, 则称此种归约为问题的分解. 即分解所得到的子问题的“与”与原问题 P 等价.

等价变换 如果一个问题 P 可以归约为一组子问题 P_1, P_2, \dots, P_n , 并且子问题 P_i 中只要有一个有解, 则原问题 P 就有解, 只有当所有子问题 P_i 都无解时原问题 P 才无解, 称此种归约为问题的等价变换, 简称变换. 即变换所得到的子问题应与原问题 P 等价.

(1) 与树分解

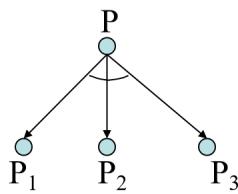


图 4-4 与树

(2) 或树——等价变换

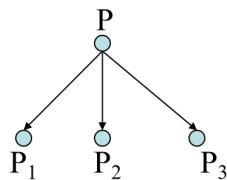


图 4-5 或树

(3) 与/或树

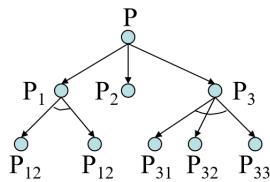


图 4-6 与/或树

(4) 端节点与终止节点 在与/或树中, 没有子节点的节点称为端节点; 本原问题所对应的节点称为终止节点. 可见, 终止节点一定是端节点, 但端节点却不一定终止节点.

(5) 可解节点与不可解节点

定义 4.9 可解节点

在与/或树中, 可解节点是满足以下三个条件之一的节点:

- ① 任何终止节点都是可解节点.
- ② 对“或”节点, 当其子节点中至少有一个为可解节点时, 则该或节点就是可解节点.
- ③ 对“与”节点, 只有当其子节点全部为可解节点时, 该与节点才是可解节点.



定义 4.10 不可解节点

- ① 不为终止节点的端节点是不可解节点.
- ② 对“或”节点, 若其全部子节点都为不可解节点, 则该或节点是不可解节点.
- ③ 对“与”节点, 只要其子节点中有一个为不可解节点, 则该与节点是不可解节点.

(6) 解树由可解节点构成, 由这些可解节点可以推理出初始节点, 可解节点的子树为解树. 在解树中一定包含初始节点.

例 4.11 图 4-7 给出的与或树中, 用红线表示的子树是一个解树. 在图 4-7 中, 节点 P 为原始问题节点, 用 t 标出的节点是终止节点. 根据可解节点的定义, 很容易推出原始问题 P 为可解节点.

注 4.12. 问题归约的求解过程实际上就是生成解树, 即证明原始节点是可解节点的过程. 这一过程涉及到搜索的问题, 对于与/或树的搜索将在后面详细讨论.

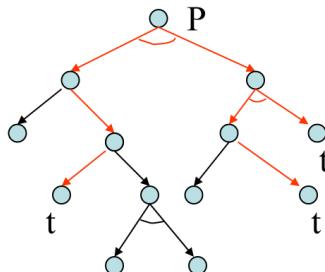


图 4-7 与/或树

例 4.13 (三阶梵塔问题) 要求把 1 号钢针上的 3 个金片全部移到 3 号钢针上, 如图 4-8 所示.

解: 本例用归约法来解决问题. 为了能够解决这一问题, 首先需要定义该问题的形式化表示方法. 设用三元组 (i, j, k) 表示问题在任一时刻的状态, 用 “ \rightarrow ” 表示状态的转换. 上述三元组 (i, j, k) 的意思分别为:

- i 代表金片 C 所在的钢针号;
- j 代表金片 B 所在的钢针号;
- k 代表金片 A 所在的钢针号.



图 4-8 三阶汉诺塔问题

利用问题归约方法, 原问题可分解为以下三个子问题:

- (1) 把金片 A 及 B 移到 2 号钢针上的双金片移动问题. 即 $(1, 1, 1) \rightarrow (1, 2, 2)$;
- (2) 把金片 C 移到 3 号钢针上的单金片移动问题. 即 $(1, 2, 2) \rightarrow (3, 2, 2)$;
- (3) 把金片 A 及 B 移到 3 号钢针的双金片移动问题. 即 $(3, 2, 2) \rightarrow (3, 3, 3)$, 其中, 子问题 (1) 和 (3) 都是一个二阶梵塔问题, 它们都还可以再继续进行分解; 子问题 (2) 是本原问题, 它已不需要再分解.

三阶梵塔问题的分解过程可用图 4-9 的与/或树表示. 在该与/或树中, 有 7 个终止节点, 它们分别对应着 7 个本原问题. 如果把这些本原问题从左至右排列起来, 即得到了原始问题的解:

$$(1, 1, 1) \rightarrow (1, 3, 3), (1, 3, 3) \rightarrow (1, 2, 3), (1, 2, 3) \rightarrow (1, 2, 2), (1, 2, 2) \rightarrow (3, 2, 2), \\ (3, 2, 2) \rightarrow (3, 2, 1), (3, 2, 1) \rightarrow (3, 3, 1), (3, 3, 1) \rightarrow (3, 3, 3).$$

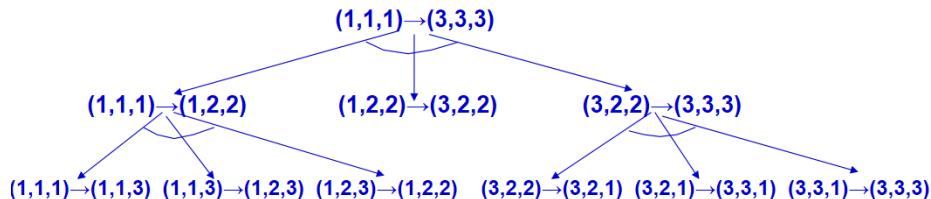


图 4-9 与/或树

4.3 状态空间的盲目搜索

4.3.1 一般图搜索过程

- 1) 先把问题的初始状态作为当前扩展节点, 对其进行扩展, 生成一组子节点;
- 2) 检查问题的目标状态是否出现在这些子节点中.
 - ① 若出现, 则搜索成功, 找到了问题的解;

② 否则, 按照某种搜索策略从已生成的子节点中选择一个节点作为当前扩展节点.

3) 重复上述过程, 直到目标状态出现在子节点中或者没有可供操作的节点为止.

 **注 4.14.** 所谓对一个节点进行“扩展”是指对该节点用某个可用操作进行作用, 生成该节点的一组子节点.

算法的数据结构和符号约定

- ▶ Open 表: 用于存放刚生成的节点;
- ▶ Closed 表: 用于存放已经扩展或将要扩展的节点;
- ▶ S_0 : 表示问题的初始状态;
- ▶ G : 表示搜索过程所得到的搜索图;
- ▶ M : 表示当前扩展节点新生成的且不为自己先辈的子节点集.

一般图搜索过程

- (1) 把初始节点 S_0 放入 Open 表, 并建立目前仅包含 S_0 的图 G ;
- (2) 检查 Open 表是否为空, 若为空, 则问题无解, 失败退出;
- (3) 把 Open 表的第一个节点取出放入 Closed 表, 并记该节点为 n ;
- (4) 考察节点 n 是否为目标节点. 若是, 则得到了问题的解, 成功退出;
- (5) 扩展节点 n , 生成一组子节点. 把这些节点中不是节点 n 先辈的那部分子节点加入集合 M , 并把这些子节点作为节点 n 的子节点加入 G 中.
- (6) 针对 M 中子节点的不同情况, 分别作如下处理:

① 对那些没有在 G 中出现过的 M 成员, 设置一个指向其父节点(即节点 n)的指针, 并把它放入 Open 表. (新生成的).

② 对那些原来已在 G 中出现过, 但还没有被扩展的 M 成员, 确定是否需要修改它指向父节点的指针. (原来生成但未扩展的).

③ 对于那些先前已在 G 中出现过, 并已经扩展了的 M 成员, 确定是否需要修改其后继节点指向父节点的指针. (原来生成也扩展过的).

(7) 按某种策略对 Open 表中的节点进行排序.

(8) 转第 (2) 步.

 **注 4.15.** 上述过程是状态空间的一般图搜索算法, 它具有通用性, 后面所要讨论的各种状态空间搜索策略都是上述过程的一个特例. 各种搜索策略的主要区别在于对 Open 表中节点的排列顺序的方式不同.

 **注 4.16.** (1) 广度优先搜索把先生成的子节点排在前面, 而深度优先搜索则把后生成的子节点排在前面.

(2) 在第(5)步对节点 n 扩展后, 生成并记入 M 的子节点有以下三种情况:

- ① 该子节点从未被任何节点生成过, 由 n 第一次生成;
 - ② 该子节点原来被其他节点生成过, 但还没有被扩展, 这一次又被 n 再次生成;
 - ③ 该子节点原来被其他节点生成过, 并且已经被扩展过, 这一次又被 n 再次生成.
- 以上三种情况是对一般图搜索算法而言的.

对于盲目搜索, 由于其状态空间是树状结构, 因此不会出现后两种情况, 每个节点经扩展后生成的子节点都是第一次出现的节点.

盲目搜索不必检查并修改指向父节点的指针.

(3) 在第(6)步针对 M 中子节点的不同情况进行处理时, 如果发生当第③种情况, 那么, 这个 M 中的节点究竟应该作为哪一个节点的后继节点呢? 一般是由原始节点到该节点路径上所付出的代价来决定的, 哪一条路经付出的代价小, 相应的节点就作为它的父节点. 所谓由原始节点到该节点路径上的代价是指这条路经上的所有有向边的代价之和.

如果发生第③种情况, 除了需要确定该子节点指向父节点的指针外, 还需要确定其后继节点指向父节点的指针. 其依据也是由原始节点到该节点的路径上的代价.

(4) 在搜索图中, 除初始节点外, 任意一个节点都含有且只含有一个指向其父节点的指针. 因此, 由所有节点及其指向父节点的指针所构成的集合是一棵树, 称为搜索树.

(5) 在搜索过程的第(4)步, 一旦某个被考察的节点是目标节点, 则搜索过程成功结束. 由初始节点到目标节点路径上的所有操作就构成了该问题的解, 而路径由第(6)步所形成的指向父节点的指针来确定.

(6) 如果搜索过程终止在第(2)步, 即没有达到目标, 且 Open 表中已无可供扩展的节点, 则失败结束.

4.3.2 广度优先和深度优先搜索

4.3.2.1 广度优先搜索

基本思想 从初始节点 S_0 开始逐层向下扩展, 在第 n 层节点还没有全部搜索完之前, 不进入第 $n+1$ 层节点的搜索方式. Open 表中的节点总是按进入的先后排序, 先进入的节点排在前面, 后进入的节点排在后面.

搜索算法 具体流程如下:

- (1) 把初始节点 S_0 放入 Open 表中;
- (2) 如果 Open 表为空, 则问题无解, 失败退出;
- (3) 把 Open 表的第一个节点取出, 放入 Closed 表, 并记该节点为 n ;

- (4) 考察节点 n 是否为目标节点. 若是, 则得到问题的解, 成功退出;
- (5) 若节点 n 不可扩展, 则转第 (2) 步;
- (6) 扩展节点 n , 将其子节点放入 Open 表的尾部, 并为每一个子节点设置指向父节点的指针, 然后转第 (2) 步.

例 4.17 3×3 的方格棋盘上的八数码难题 如图4-10所示, 表上放置了标有数字 1、2、3、4、5、6、7、8 的八张牌, 初始状态 S_0 , 目标状态 S_g . 可以使用的操作有

空格左移, 空格上移, 空格右移, 空格下移,

即只允许把位于空格左、上、右和下方的牌移入空格. 要求应用广度优先搜索策略寻找从初始状态到目标状态的解路径.

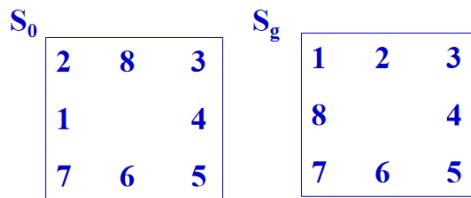


图 4-10 与/或树

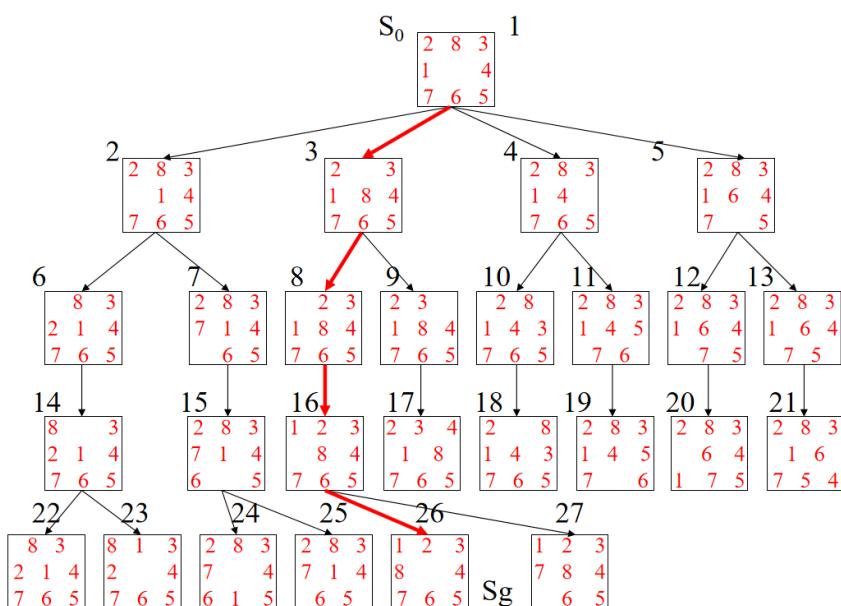


图 4-11 与/或树

4.3.2.2 深度优先搜索

基本思想 从初始节点 S_0 开始, 在其子节点中选择一个最新生成的节点进行考察, 如果该子节点不是目标节点且可以扩展, 则扩展该子节点, 然后再在此子节点的子节点中选择一个最新生成的节点进行考察; 依此向下搜索, 直到某个子节点既不是目标节点, 又不能继续扩展时, 才选择其兄弟节点进行考察.

算法描述

- (1) 把初始节点 S_0 放入 Open 表中;
- (2) 如果 Open 表为空, 则问题无解, 失败退出;
- (3) 把 Open 表的第一个节点取出, 放入 Closed 表, 并记该节点为 n ;
- (4) 考察节点 n 是否为目标节点. 若是, 则得到问题的解, 成功退出;
- (5) 若节点 n 不可扩展, 则转第 (2) 步;
- (6) 扩展节点 n , 将其子节点放入 Open 表的首部, 并为每一个子节点设置指向父节点的指针, 然后转第 (2) 步.

例 4.18 八数码难题 八数码难题的深度优先搜索如右图4-12.



图 4-12 深度优先搜索 (不完全图)

注 4.19. 有界深度优先搜索算法是一种改进的深度优先算法, 将其深度限制为 d_m .

4.3.3 代价树搜索

4.3.3.1 代价树的广度优先搜索

在代价树中, 可以用 $g(n)$ 表示从初始节点 S_0 到节点 n 的代价, 用 $c(n_1, n_2)$ 表示从父节点 n_1 到其子节点 n_2 的代价. 这样, 对节点 n_2 的代价有: $g(n_2) = g(n_1) + c(n_1, n_2)$. 代价树搜索的目的是为了找到最佳解(即找一条代价最小的解路径).

4.3.3.2 代价树的广度优先搜索算法

广度优先搜索算法的流程描述:

- (1) 把初始节点 S_0 放入 Open 表中, 置 S_0 的代价 $g(S_0) = 0$;
- (2) 如果 Open 表为空, 则问题无解, 失败退出;
- (3) 把 Open 表的第一个节点取出, 放入 Closed 表, 并记该节点为 n ;
- (4) 考察节点 n 是否为目标. 若是, 则找到了问题的解, 成功退出;
- (5) 若节点 n 不可扩展, 则转第 (2) 步;
- (6) 扩展节点 n , 生成其子节点 $n_i (i = 1, 2, \dots)$, 将这些子节点放入 Open 表中, 并为每一个子节点设置指向父节点的指针. 按如下公式:

$$g(n_i) = g(n) + c(n_i), i = 1, 2, \dots \quad (4.6)$$

计算各子结点的代价, 并根据各子结点的代价对 Open 表中的全部结点按代价由小到大的顺序排序. 然后转第 (2) 步.

例 4.20 (城市交通问题) 设有 5 个城市, 它们之间的交通线路如左侧子图 4-13 所示, 图中的数字表示两个城市之间的交通费用, 即代价. 用代价树的广度优先搜索, 求从 A 市出发到 E 市, 费用最小的交通路线.

解:



图 4-13 城市交通图 (左子图), 城市交通图的广度优先代价树 (右子图)

代价树如右侧子图4-13所示. 红线为最优解, 其代价为 8.

4.3.3.3 代价树的深度优先搜索算法

- (1) 把初始节点 S_0 放入 Open 表中, 置 S_0 的代价 $g(S_0) = 0$;
- (2) 如果 Open 表为空, 则问题无解, 搜索算法失败, 退出;
- (3) 把 Open 表的第一个节点取出放入 Closed 表, 并记该节点为 n ;
- (4) 考察节点 n 是否为目标节点. 若是, 则找到了问题的解, 搜索算法成功退出;
- (5) 若节点 n 不可扩展, 则转第 (2) 步;
- (6) 扩展节点 n , 生成其子节点 n_i , ($i = 1, 2, \dots$), 将这些子节点按边代价由小到大放入 Open 表的首部, 并为每一个子节点设置指向父节点的指针. 然后转第 (2) 步.

4.4 搜索策略

4.4.1 状态空间的启发式搜索

定义 4.21 启发性信息

启发性信息是指那种与具体问题求解过程有关, 并可指导搜索过程朝着最有希望方向前进的控制信息.

启发性信息的种类

- ▶ 有效地确定扩展节点的信息;
- ▶ 有效地决定哪些后继节点应被生成的信息;
- ▶ 能决定在扩展一个节点时哪些节点应从搜索树上删除的信息.

启发性信息的作用 启发信息的启发能力越强, 扩展的无用结点越少.

定义 4.22 估价函数

估价函数是用来估计节点重要性的函数.



估价函数 $f(n)$ 被定义为从初始节点 S_0 出发, 约束经过节点 n 到达目标节点 S_g 的所有路径中最小路径代价的估计值. 它的一般形式为:

$$f(n) = g(n) + h(n), \quad (4.7)$$

其中, $g(n)$ 是从初始节点 S_0 到节点 n 的实际代价; $h(n)$ 是从节点 n 到目标节点 S_g 的最优路径的估计代价.

启发性信息和估价函数

例 4.23 (八数码难题) 设问题的初始状态 S_0 和目标状态 S_g , 如图4-14所示, 且估价函数为

$$f(n) = d(n) + W(n), \quad (4.8)$$

其中 $d(n)$ 表示节点 n 在搜索树中的深度, $W(n)$ 表示节点 n 中“不在位”的数码个数. 计算初始状态 S_0 的估价函数值 $f(S_0)$.

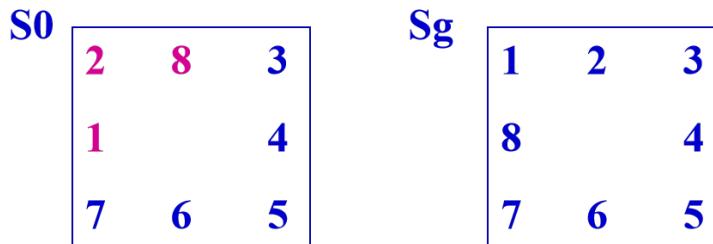


图 4-14 八数码难题

解: 取 $g(n) = d(n), h(n) = W(n)$. 它说明是用从 S_0 到 n 的路径上的单位代价表示实际代价, 用结点 n 中“不在位”的数码个数作为启发信息. 一般来说, 某节点中的“不在位”的数码个数越多, 说明它离目标节点越远.

例 4.24 对初始节点 S_0 , 由于 $d(S_0) = 0$, $W(S_0) = 3$, 因此有

$$f(S_0) = 0 + 3 = 3. \quad (4.9)$$

4.4.2 A 算法

在图搜索算法中, 如果能在搜索的每一步都利用估价函数 $f(n) = g(n) + h(n)$, 对 Open 表中的节点进行排序, 则该搜索算法为 A 算法.

 **注 4.25.** 由于估价函数中带有问题自身的启发信息, 因此, A 算法也被称为启发式搜索算法.

 **注 4.26.** 可根据搜索过程中选择扩展节点的范围, 将启发式搜索算法分为全局择优搜索算法和局部择优搜索算法.

- ▶ **全局择优:** 从 Open 表的所有节点中选择一个估价函数值最小的一个节点进行扩展.
- ▶ **局部择优:** 仅从刚生成的子节点中选择一个估价函数值最小的一个节点进行扩展.

4.4.2.1 全局择优搜索 A 算法

全局择优搜索 A 算法描述:

- (1) 把初始节点 S_0 放入 Open 表中, $f(S_0) = g(S_0) + h(S_0)$;
- (2) 如果 Open 表为空, 则问题无解, 失败退出;
- (3) 把 Open 表的第一个节点取出, 放入 Closed 表, 并记该节点为 n ;
- (4) 考察节点 n 是否为目标节点. 若是, 则找到了问题的解, 成功退出;
- (5) 若节点 n 不可扩展, 则转第 (2) 步;
- (6) 扩展节点 n , 生成其子节点 n_i , ($i = 1, 2, \dots$), 计算每一个子节点的估价值 $f(n_i)$, ($i = 1, 2, \dots$), 并为每一个子节点设置指向父节点的指针, 然后将这些子节点放入 Open 表中;
- (7) 根据各节点的估价函数值, 对 Open 表中的全部节点按从小到大的顺序重新进行排序;
- (8) 转第 (2) 步.

例 4.27 八数码难题. 如图4-14所示, 问题的初始状态 S_0 , 目标状态 S_g 估价函数与例4.23相同. 请用全局择优搜索解决该问题.

解: 该问题的全局择优搜索树如下图所示. 在该图中, 每个节点旁边的数字是该节点的估价函数值. 对节点 S_2 , 其估价函数值的计算为: $f(S_2) = d(S_2) + W(S_2) = 1 + 3 = 4$.

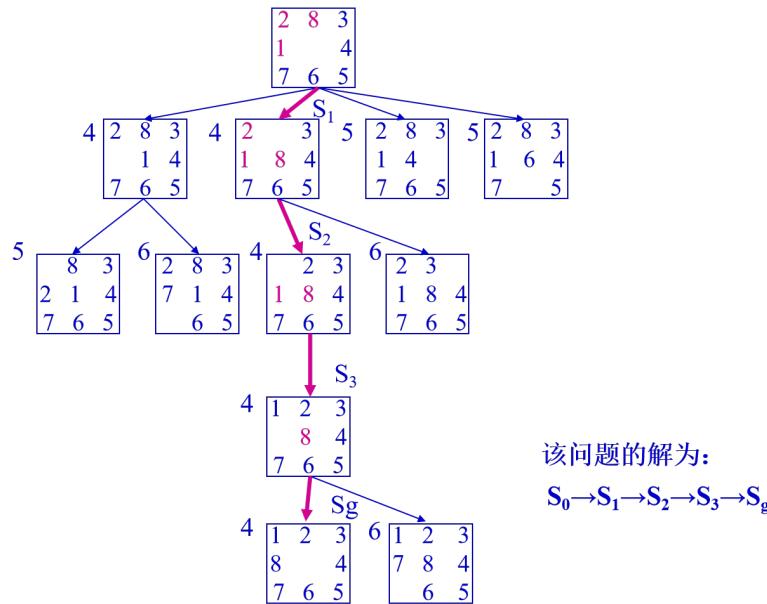


图 4-15 八数码难题的全局择优搜索树

该问题的解为:

$$S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_g. \quad (4.10)$$

4.4.3 A^* 算法

定义 4.28 A^* 算法

A^* 算法是对 A 算法的估价函数 $f(n) = g(n) + h(n)$ 加上某些限制后得到的一种启发式搜索算法.

假设 $f^*(n)$ 是从初始节点出发, 经过节点 n 达到目标节点的最小代价, 估价函数 $f(n)$ 是对 $f^*(n)$ 的估计值. 且

$$f^*(n) = g^*(n) + h^*(n). \quad (4.11)$$

A^* 算法对 A 算法(全局择优的启发式搜索算法)中的 $g(n)$ 和 $h(n)$ 分别做如下限制:

第一, $g(n)$ 是对最小代价 $g^*(n)$ 的估计, $g(n) \geq g^*(n)$ 且 $g(n) > 0$;

第二, $h(n)$ 是最小代价 $h^*(n)$ 的下界, 即对任意节点 n , 均有 $h(n) \leq h^*(n)$.

即满足上述两条限制的 A 算法称为 A^* 算法.

4.4.4 A^* 算法的可采纳性

定义 4.29 可采纳性

对任一状态空间图, 当从初始节点到目标节点有路径存在时, 如果搜索算法总能在有限步骤内找到一条从初始节点到目标节点的最佳路径, 并在此路径上结束, 则称该搜索算法是可采纳的.



A^* 算法可采纳性的证明 以下分三步进行证明.

定理 4.1 A^* 收敛性

对有限图, 如果从初始节点 S_0 到目标节点 S_g 有路径存在, 则算法 A^* 一定成功结束.



证明. 首先证明算法必然会结束. 由于搜索图为有限图, 如果算法能找到解, 则成功结束; 如果算法找不到解, 则必然会由于 Open 表变空而结束. 因此, A^* 算法必然会结束. 然后证明算法一定会成功结束. 由于至少存在一条由初始节点到目标节点的路径, 设此路径为

$$S_0 = n_0, n_1, \dots, n_k = S_g. \quad (4.12)$$

□

算法开始时, 节点 n_0 在 Open 表中, 而且路径中任一节点 n_i 离开 Open 表后, 其后继节点 n_{i+1} 必然进入 Open 表, 这样, 在 Open 表变为空之前, 目标节点必然出现在 Open 表中. 因此, 算法一定会成功结束.

引理 4.30

对无限图, 如果从初始节点 S_0 到目标节点 S_g 有路径存在, 则 A^* 算法不终止的话, 则从 Open 表中选出的节点必将具有任意大的 f 值.



证明. 设 $d^*(n)$ 是 A^* 生成的从初始节点 S_0 到节点 n 的最短路经长度, 由于搜索图中每条边的代价都是一个正数, 令这些正数中的最小的一个数是 e , 则有

$$g^*(n) \geq d^*(n) \times e. \quad (4.13)$$

因为 $g^*(n)$ 是最佳路径的代价, 故有

$$g(n) \geq g^*(n) \geq d^*(n) \times e. \quad (4.14)$$

又因为 $h(n) \geq 0$, 故有

$$f(n) = g(n) + h(n) \geq g(n) \geq d^*(n) \times e. \quad (4.15)$$

如果 A^* 算法不终止的话, 从 Open 表中选出的节点必将具有任意大的 $d^*(n)$ 值, 因此, 也将具有任意大的 f 值. \square

引理 4.31

在 A^* 算法终止前的任何时刻, Open 表中总存在节点 n' , 它是从初始节点 S_0 到目标节点的最佳路径上的一个节点, 且满足 $f(n') \leq f^*(S_0)$. \spadesuit

证明. 设从初始节点 S_0 到目标节点 t 的一条最佳路径序列为

$$S_0 = n_0, n_1, \dots, n_k = S_g. \quad (4.16)$$

算法开始时, 节点 S_0 在 Open 表中, 当节点 S_0 离开 Open 表进入 Closed 表时, 节点 n_1 进入 Open 表, 因此, A^* 没有结束以前, 在 Open 表中必存在最佳路径上的节点. 设这些节点中排在最前面的节点为 n' , 则有

$$f(n') = g(n') + h(n'). \quad (4.17)$$

由于 n' 在最佳路径上, 故有 $g(n') = g^*(n')$, 从而

$$f(n') = g^*(n') + h(n'). \quad (4.18)$$

又由于 A^* 算法满足 $h(n') \leq h^*(n')$, 故有

$$f(n') \leq g^*(n') + h^*(n') = f^*(n'). \quad (4.19)$$

因为在最佳路径上的所有节点的 f^* 值都应相等, 因此有

$$f(n') \leq f^*(S_0). \quad (4.20)$$

\square

定理 4.2

对无限图, 若从初始节点 S_0 到目标节点 t 有路径存在, 则 A^* 算法必然会结束.



证明. (反证法) 假设 A^* 不结束, 由引理 4.1 知, Open 表中的节点有任意大的 f 值, 这与引理 4.2 的结论相矛盾, 因此, A^* 算法只能成功结束.

性质 4.1

Open 表中任一具有 $f(n) < f^*(S_0)$ 的节点 n , 最终都被 A^* 算法选作扩展节点.



(证明略)



下面给出 A^* 算法的可纳性.

定理 4.3

A^* 算法是可采纳的, 即若存在从初始节点 S_0 到目标节点 S_g 的路径, 则 A^* 算法必能在最佳路径上结束.



证明. 过程分以下两步进行:

先证明 A^* 算法一定能够终止在某个目标节点上. 由定理 4.1 和定理 4.2 可知, 无论是对有限图还是无限图, A^* 算法都能够找到某个目标节点而结束.

再证明 A^* 算法只能终止在最佳路径上 (反证法). 假设 A^* 算法未能终止在最佳路径上, 而是终止在某个目标节点 t 处, 则有

$$f(t) = g(t) > f^*(S_0). \quad (4.21)$$

但由引理 4.2 可知, 在 A^* 算法结束前, 必有最佳路径上的一个节点 n' 在 Open 表中, 且有

$$f(n') \leq f^*(S_0) < f(t). \quad (4.22)$$

这时, A^* 算法一定会选择 n' 来扩展, 而不可能选择 t , 从而也不会去测试目标节点 t , 这就与假设 A^* 算法终止在目标节点 t 相矛盾. 因此, A^* 算法只能终止在最佳路径上. □

性质 4.2

在 A^* 算法中, 对任何被扩展的节点 n , 都有 $f(n) \leq f^*(S_0)$.



证明. 令 n 是由 A^* 选作扩展的任一节点, 因此 n 不会是目标节点, 且搜索没有结束. 由引理 4.2 可知, 在 Open 表中有满足 $f(n') \leq f^*(S_0)$ 的节点 n' . 若 $n = n'$, 则有 $f(n) \leq f^*(S_0)$; 否则, 选择 n 扩展, 必有

$$f(n) \leq f(n'). \quad (4.23)$$

所以有

$$f(n) \leq f^*(S_0). \quad (4.24)$$

□

A^* 算法的最优性

A^* 算法的搜索效率很大程度上取决于估价函数 $h(n)$. 一般来说, 在满足 $h(n) \leq h^*(n)$ 的前提下, $h(n)$ 的值越大越好. $h(n)$ 的值越大, 说明它携带的启发性信息越多, A^* 算法搜索时扩展的节点就越少, 搜索效率就越高. 下面通过一个定理来描述这一特性.

定理 4.4

A^* 算法 A_1^* 和 A_2^* , 它们有

$$A_1^* : f_1(n) = g_1(n) + h_1(n), \quad (4.25)$$

$$A_2^* : f_2(n) = g_2(n) + h_2(n). \quad (4.26)$$

如果 A_2^* 比 A_1^* 有更多的启发性信息, 即对所有非目标节点均有

$$h_2(n) > h_1(n), \quad (4.27)$$

则在搜索过程中, 被 A_2^* 扩展的节点也必然被 A_1^* 扩展, 即 A_1^* 扩展的节点不会比 A_2^* 扩展的节点少, 亦即 A_2^* 扩展的节点集是 A_1^* 扩展的节点集的子集. ♥

证明. (数学归纳法)

(1) 对深度 $d(n) = 0$ 的节点, 即 n 为初始节点 S_0 , 如 n 为目标节点, 则 A_1^* 和 A_2^* 都不扩展 n ; 如果 n 不是目标节点, 则 A_1^* 和 A_2^* 都要扩展 n .

(2) 假设对 A_2^* 中 $d(n) = k$ 的任意节点 n 结论成立, 即 A_1^* 也扩展了这些节点.

(3) 证明 A_2^* 中 $d(n) = k + 1$ 的任意节点 n , 也要由 A_1^* 扩展 (用反证法).

假设 A_2 搜索树上有一个满足 $d(n) = k + 1$ 的节点 n , A_2^* 扩展了该节点, 但 A_1^* 没有扩展它. 根据第 (2) 条的假设, 知道 A_1^* 扩展了节点 n 的父节点. 因此, n 必定在 A_1^* 的 Open

表中。既然节点 n 没有被 A_1^* 扩展，则有

$$f_1(n) \geq f^*(S_0). \quad (4.28)$$

即 $g_1(n) + h_1(n) \geq f^*(S_0)$ 。但由于 $d = k$ 时， A_2^* 扩展的节点 A_1^* 也一定扩展，故有

$$g_1(n) \leq g_2(n). \quad (4.29)$$

因此有 $h_1(n) \geq f^*(S_0) - g_2(n)$ 。

另一方面，由于 A_2^* 扩展了 n ，因此有

$$f_2(n) \leq f^*(0). \quad (4.30)$$

即 $g_2(n) + h_2(n) \leq f^*(S_0)$ ，亦即 $h_2(n) \leq f^*(S_0) - g_2(n)$ ，所以有 $h_1(n) \geq h_2(n)$ 。这与我们最初假设的 $h_1(n) < h_2(n)$ 矛盾，因此假设不成立。□

 **注 4.32.** A^* 算法中，每当扩展一个节点 n ，都需要检查其子节点是否已在 *Open* 表或 *Closed* 表中。

- ① 对已在 *Open* 表中的子节点，需要决定是否调整指向其父节点的指针；
- ② 对已在 *Closed* 表中的子节点，除需要决定是否调整其指向父节点的指针外，还需要决定是否调整其子节点的后继节点的父指针。

如果能够保证，每当扩展一个节点时就已经找到了通往这个节点的最佳路径，就没有必要再去作上述检查。为满足这一要求，我们需要对启发函数 $h(n)$ 增加单调性限制。

定义 4.33 启发函数的单调限制

如果启发函数满足以下两个条件：

- (1) $h(S_g) = 0$ ；
- (2) 对任意节点 n_i 及其任一子节点 n_j ，都有

$$0 \leq h(n_i) - h(n_j) \leq c(n_i, n_j), \quad (4.31)$$

其中 $c(n_i, n_j)$ 是 n_i 到其子节点 n_j 的边代价，则称 $h(n)$ 满足单调限制。

定理 4.5 估价函数 $h(n)$ 的单调限制

如果 h 满足单调条件，则当 A^* 算法扩展节点 n 时，该节点就已经找到了通往它的最佳路径，即 $g(n) = g^*(n)$ 。

证明. 设 A^* 正要扩展节点 n , 而节点序列

$$S_0 = n_0, n_1, \dots, n_k = n, \quad (4.32)$$

是由初始节点 S_0 到节点 n 的最佳路径. 其中, n_i 是这个序列中最后一个位于 Closed 表中的节点, 则上述节点序列中的 n_{i+1} 节点必定在 Open 表中, 则有

$$g^*(n_i) + h(n_i) \leq g^*(n_i) + c(n_i, n_{i+1}) + h(n_{i+1}). \quad (4.33)$$

由于节点 n_i 和 n_{i+1} 都在最佳路径上, 故有

$$g^*(n_{i+1}) = g^*(n_i) + c(n_i, n_{i+1}). \quad (4.34)$$

所以

$$g^*(n_i) + h(n_i) \leq g^*(n_{i+1}) + h(n_{i+1}). \quad (4.35)$$

一直推导, 可得

$$g^*(n_{i+1}) + h(n_{i+1}) \leq g^*(n_k) + h(n_k). \quad (4.36)$$

由于节点 n_{i+1} 在最佳路径上, 故有

$$f(n_{i+1}) \leq g^*(n) + h(n). \quad (4.37)$$

因为这时 A^* 扩展节点 n 而不扩展节点 n_{i+1} , 则有

$$f(n) = g(n) + h(n) \leq f(n_{i+1}) \leq g^*(n) + h(n). \quad (4.38)$$

即

$$g(n) \leq g^*(n), \quad (4.39)$$

但是 $g^*(n)$ 是最小代价值, 应当有

$$g(n) \geq g^*(n). \quad (4.40)$$

所以有

$$g(n) = g^*(n). \quad (4.41)$$

□

定理 4.6

如果 $h(n)$ 满足单调限制, 则 A^* 算法扩展的节点序列的 f 值是非递减的, 即 $f(n_i) \leq f(n_{i+1})$.



证明. 假设节点 n_{i+1} 在节点 n_i 之后立即扩展, 由单调限制条件可知

$$h(n_i) - h(n_{i+1}) \leq c(n_i, n_{i+1}), \quad (4.42)$$

即

$$f(n_i) - g(n_i) - f(n_{i+1}) + g(n_{i+1}) \leq c(n_i, n_{i+1}), \quad (4.43)$$

亦即

$$f(n_i) - g(n_i) - f(n_{i+1}) + g(n_i) + c(n_i, n_{i+1}) \leq c(n_i, n_{i+1}), \quad (4.44)$$

所以

$$f(n_i) - f(n_{i+1}) \leq 0, \quad (4.45)$$

即

$$f(n_i) \leq f(n_{i+1}). \quad (4.46)$$

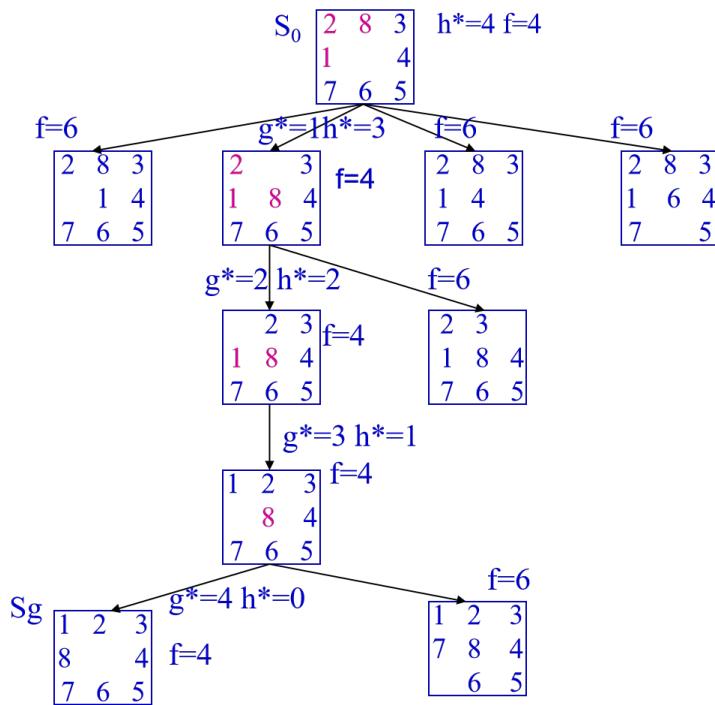
证毕 □

 **注 4.34.** 以上两个定理都是在 $h(n)$ 满足单调性限制的前提下才成立. 如果 $h(n)$ 不满足单调性限制, 则它们不一定成立. 在 $h(n)$ 满足单调性限制下的 A^* 算法常被称为改进的 A^* 算法.

4.4.5 A^* 算法应用举例

例 4.35 八数码难题的 A^* 算法.

解: 令 $f(n) = d(n) + P(n)$, $d(n)$ 表示深度, 且 $P(n)$ 与目标距离 $f^* = g^* + h^*$.

图 4-16 八数码难题 $h(n) = P(n)$ 的搜索树

4.4.6 算法应用举例

例 4.36 A^* 算法应用举例——修道士和魔鬼问题.

解: 用 m 表示左岸的修道士人数, c 表示左岸的魔鬼数, b 表示左岸的船数, 用三元组 (m, c, b) 表示问题的状态. 对 A^* 算法, 首先需要确定估价函数. 设 $g(n) = d(n)$, $h(n) = m + c - 2b$, 则有

$$f(n) = g(n) + h(n) = d(n) + m + c - 2b, \quad (4.47)$$

其中, $d(n)$ 为节点的深度. 通过分析可知 $h(n) \leq h^*(n)$, 满足 A^* 算法的限制条件.

M-C 问题的搜索过程如图4-17所示.



图 4-17 传教士和魔鬼问题的搜索图

与/或树的一般搜索 与/或树的搜索过程实际上是一个不断寻找解树的过程. 其一般搜索过程如下:

- (1) 把原始问题作为初始节点 S_0 , 并把它作为当前节点;
- (2) 应用分解或等价变换操作对当前节点进行扩展;
- (3) 为每个子节点设置指向父节点的指针;
- (4) 选择合适的子节点作为当前节点, 反复执行第 (2) 步和第 (3) 步, 在此期间需要多次调用可解标记过程或不可解标记过程, 直到初始节点被标记为可解节点或不可解节点为止. 上述搜索过程将形成一棵与/或树.

定义 4.37 搜索树

由搜索过程所形成的与/或树称为搜索树.



4.4.6.1 与/或树的广度优先

与/或树的广度优先搜索与状态空间的广度优先搜索的主要差别是, 需要在搜索过程中需要多次调用可解标识过程或不可解标识过程.

与/或树的广度优先搜索算法

(1) 把初始节点 S_0 放入 Open 表中;

(2) 把 Open 表的第一个节点取出放入 Closed 表, 并记该节点为 n ;

(3) 如果节点 n 可扩展, 则做下列工作:

① 展节点 n , 将其子节点放入 Open 表的尾部, 并为每一个子节点设置指向父节点的指针;

② 考察这些子节点中有否终止节点. 若有, 则标记这些终止节点为可解节点, 并用可解标记过程对其父节点及先辈节点中的可解解节点进行标记. 如果初始解节点 S_0 能够被标记为可解节点, 就得到了解树, 搜索成功, 退出搜索过程; 如果不能确定 S_0 为可解节点, 则从 Open 表中删去具有可解先辈的节点.

③ 转第(2)步.

(4) 如果节点 n 不可扩展, 则作下列工作:

① 标记节点 n 为不可解节点;

② 应用不可解标记过程对节点 n 的先辈中不可解解的节点进行标记. 如果初始解节点 S_0 也被标记为不可解节点, 则搜索失败, 表明原始问题无解, 退出搜索过程; 如果不能确定 S_0 为不可解节点, 则从 Open 表中删去具有不可解先辈的节点.

③ 转第(2)步.

例 4.38 设有下图4-18所示的与/或树, 节点按标注顺序进行扩展, 其中表有 t_1, t_2, t_3 的节点是终止节点, A, B, C 为不可解的端节点.

例 4.38 搜索过程

(1) 先扩展 1 号节点, 生成 2 号节点和 3 号节点.

(2) 扩展 2 号节点, 生成 A 节点和 4 号节点.

(3) 扩展 3 号节点, 生成 t_1 节点和 5 号节点. 由于 t_1 为终止节点, 则标记它为可解节点, 并应用可解标记过程, 不能确定 3 号节点是否可解.

(4) 扩展节点 A , 由于 A 是端节点, 因此不可扩展. 调用不可解标记过程.

(5) 扩展 4 号节点, 生成 t_2 节点和 B 节点. 由于 t_2 为终止节点, 则标记它为可解节点, 并应用可解标记过程, 可标记 2 号节点为可解, 但不能标记 1 号节点为可解.

(6) 扩展 5 号节点, 生成 t_3 节点和 C 节点. 由于 t_3 为终止节点, 则标记它为可解节点, 并应用可解标记过程, 可标记 1 号节点为可解节点.

(7) 搜索成功, 得到由 1、2、3、4 和 5 号节点即 t_1, t_2 和 t_3 节点构成的解树.



图 4-18 与/或树的广度优先搜索

4.4.6.2 与/或树的深度优先搜索

与/或树的深度优先搜索和与/或树的广度优先搜索过程基本相同，其主要区别在于 Open 表中节点的排列顺序不同。在扩展节点时，与/或树的深度优先搜索过程总是把刚生成的节点放在 Open 表的首部。与/或树的深度优先搜索也可以带有深度限制 d_m ，其搜索算法如下：

(1) 把初始节点 S_0 放入 Open 表中；

(2) 把 Open 表第一个节点取出放入 Closed 表，并记该节点为 n ；

(3) 如果节点 n 的深度等于 d_m ，则转第(5)步的第①点；

(4) 如果节点 n 可扩展，则做下列工作：

① 扩展节点 n ，将其子节点放入 Open 表的首部，并为每一个子节点设置指向父节点的指针；

② 考察这些子节点中是否有终止节点。若有，则标记这些终止节点为可解节点，并用可解标记过程对其父节点及先辈节点中的可解节点进行标记。如果初始解节点 S_0 能够被标记为可解节点，就得到了解树，搜索成功，退出搜索过程；如果不能确定 S_0 为可解节点，则从 Open 表中删去具有可解先辈的节点。

③ 转第(2)步。

(5) 如果节点 n 不可扩展，则作下列工作：

① 标记节点 n 为不可解节点；

② 应用不可解标记过程对节点 n 的先辈中不可解的节点进行标记。如果初始解节点 S_0 也被标记为不可解节点，则搜索失败，表明原始问题无解，退出搜索过程；如果不能确定 S_0 为不可解节点，则从 Open 表中删去具有不可解先辈的节点。

③ 转第(2)步.

深度优先搜索过程 (1) 先扩展 1 号节点, 生成 2 号节点和 3 号节点.

(2) 扩展 3 号节点, 生成 t_1 节点和 5 号节点. 由于 t_1 为终止节点, 则标记它为可解节点, 并应用可解标记过程, 不能确定 3 号节点是否可解.

(3) 扩展 5 号节点, 生成 t_3 节点和 C 节点. 由于 t_3 为终止节点, 则标记它为可解节点, 并应用可解标记过程, 可标记 3 号节点为可解节点, 但不能标记 1 号为可解.

(4) 扩展 2 号节点, 生成 A 节点和 4 号节点.

(5) 扩展 4 号节点, 生成 t_2 节点和 B 节点. 由于 t_2 为终止节点, 则标记它为可解节点, 并应用可解标记过程, 可标记 2 号节点为可解, 再往上又可标记 1 号节点为可解.

(6) 搜索成功, 得到由 1、3、5、2、4 号节点, 即 t_1 、 t_2 、 t_3 节点构成的解树.

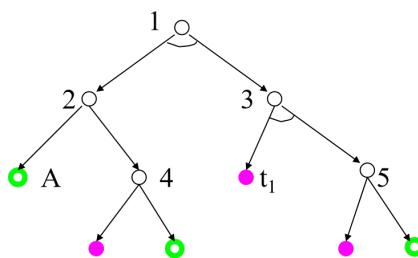


图 4-19 与/或树的有界深度优先搜索

对上例, 若按有界深度优先, 且设 $d_m = 4$, 则其节点扩展顺序为: 1, 3, 5, 2, 4.

4.5 与/或树的启发式搜索

与/或树的启发式搜索过程实际上是一种利用搜索过程所得到的启发性信息寻找最优解树的过程. 算法的每一步都试图找到一个最有希望成为最优解树的子树.

定义 4.39 最优解树

最优解树是指代价最小的那棵解树.

它涉及到解树的代价与希望树.

解树的代价与希望树 解树的代价可按如下规则计算:

(1) 若 n 为终止节点, 则其代价 $b(n) = 0$;

(2) 若 n 为或节点, 且子节点为 n_1, n_2, \dots, n_k , 则 n 的代价为:

$$h(n) = \min_{1 \leq i \leq k} \{c(n, n_i) + h(n_i)\}, \quad (4.48)$$

其中, $c(n, n_i)$ 是节点 n 到其子节点 n_i 的边代价.

(3) 若 n 为与节点, 且子节点为 n_1, n_2, \dots, n_k , 则 n 的代价可用和代价法或最大代价法.

- 若用和代价法, 则其计算公式为:

$$h(n) = \sum_{i=1}^k \{c(n, n_i) + h(n_i)\}. \quad (4.49)$$

- 若用最大代价法, 则其计算公式为:

$$h(n) = \max_{1 \leq i \leq k} \{c(n, n_i) + h(n_i)\}. \quad (4.50)$$

(4) 若 n 是端节点, 但又不是终止节点, 则 n 不可扩展, 其代价定义为 $h(n) = \alpha$.

(5) 根节点的代价即为解树的代价.

例 4.40 设下图是一棵与/或树, 左边的解树由 S_0, A, t_1, C 及 t_2 组成; 右边的解树由 S_0, B, t_2, D 及 t_4 组成. 在此与或树中, t_1, t_2, t_3, t_4 为终止节点; E 和 F 是端节点; 边上的数字是该边的代价, 试计算解树的代价.

解: 按和代价, 先计算左边的解树: $h(S_0) = 2 + 4 + 6 + 2 = 14$;

按最大代价计算, 有: $h(S_0) = (2 + 6) + 2 = 10$.

再计算右边的解树的和代价: $h(S_0) = 1 + 5 + 3 + 2 = 11$;

按最大代价, 有: $h(S_0) = (1 + 5) + 2 = 8$.

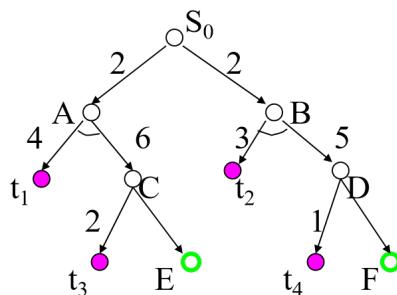


图 4-20 与/或树的代价

定义 4.41 希望树

希望树是指搜索过程中最有可能成为最优解树的那棵树.

与/或树的启发式搜索过程就是不断地选择、修正希望树的过程，在该过程中，希望树是不断变化的。

定义 4.42 希望解树

- (1) 初始节点 S_0 在希望树 T 中。
- (2) 如果 n 是具有子节点 n_1, n_2, \dots, n_k 的或节点，则 n 的某个子节点 n_i 在希望树 T 中的充分必要条件是：

$$h(n_i) = \min_{1 \leq i \leq n} \{c(n, n_i) + h(n_i)\}.$$

- (3) 如果 n 是与节点，则 n 的全部子节点都在希望树 T 中。

与/或树的启发式搜索过程如下：

- (1) 把初始节点 S_0 放入 Open 表中，计算 $h(S_0)$ ；
- (2) 计算希望树 T ；
- (3) 依次在 Open 表中取出 T 的端节点放入 Closed 表，并记该节点为 n ；
- (4) 如果节点 n 为终止节点，则做下列工作：
 - ①标记节点 n 为可解节点；
 - ②在 T 上应用可解标记过程，对 n 的先辈节点中的所有可解解节点进行标记；
 - ③如果初始解节点 S_0 能够被标记为可解节点，则 T 就是最优解树，成功退出；
 - ④否则，从 Open 表中删去具有可解先辈的所有节点。
 - ⑤转第 (2) 步。
- (5) 如果节点 n 不是终止节点，但可扩展，则做下列工作：
 - ①扩展节点 n ，生成 n 的所有子节点；
 - ②把这些子节点都放入 Open 表中，并为每一个子节点设置指向父节点 n 的指针
 - ③计算这些子节点及其先辈节点的 h 值；
 - ④转第 (2) 步。
- (6) 如果节点 n 不是终止节点，且不可扩展，则做下列工作：
 - ①标记节点 n 为不可解节点；
 - ②在 T 上应用不可解标记过程，对 n 的先辈节点中的所有不可解解节点进行标记；

- ③ 如果初始解节点 S_0 能够被标记为不可解节点, 则问题无解, 失败退出;
- ④ 否则, 从 Open 表中删去具有不可解先辈的所有节点.
- ⑤ 转第 (2) 步.

 **注 4.43.** 要求搜索过程每次扩展节点时都同时扩展两层, 且按一层或节点、一层与节点的间隔方式进行扩展. 它实际上就是下一节将要讨论的博弈树的结构.

设初始节点为 S_0 , 对 S_0 扩展后得到的与/或树如右图所示. 其中, 端节点 B, C, E, F 下面的数字是用启发函数估算出的 h 值, 节点 S_0, A, D 旁边的数字是按和代价法计算出来的节点代价. 此时, S_0 的右子树是当前的希望树.

例 4.44 按和代价法: 节点 A 的值 = $3+1+2+1+1=8$.



图 4-21 扩展 S_0 后得到的与/或树

扩展节点 E , 得到如下图所示的与/或树. 此时, 由右子树求出的 $h(S_0) = 12$. 但是, 由左子树求出的 $h(S_0) = 9$. 显然, 左子树的代价小, 因此, 当前的希望树应改为左子树.



图 4-22 扩展节点 E 后得到的与/或树

对节点 B 进行扩展, 扩展两层后得到的与/或树如下图所示. 由于节点 H 和 I 是可解

节点, 故调用可解标记过程, 得节点 G, B 也为可解节点, 但不能标记 S_0 为可解节点, 须继续扩展. 当前的希望树仍然是左子树.

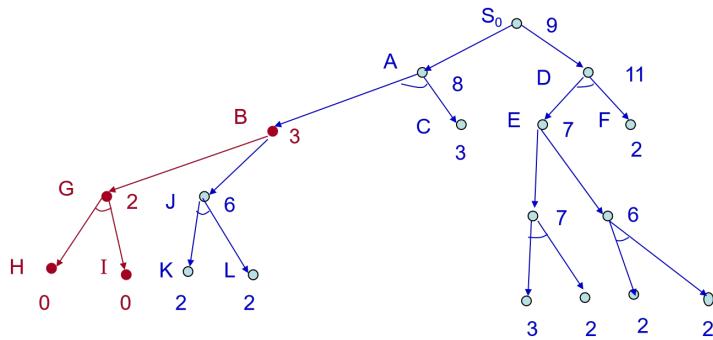


图 4-23 扩展节点 B 后得到的与/或树

对节点 C 进行扩展, 扩展两层后得到的与/或树如右图所示. 由于节点 N 和 P 是可解节点, 故调用可解标记过程, 得节点 M, C 和 A 也为可解节点, 进而可标记 S_0 为可解节点, 这就得到了代价最小的解树. 按和代价法, 该最优解的代价为 9.

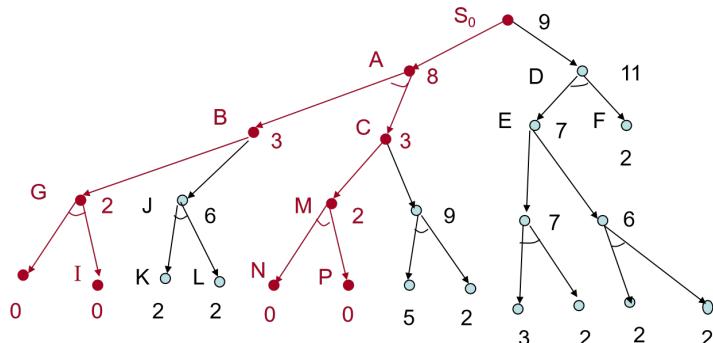


图 4-24 扩展节点 C 后得到的与/或树

4.6 博弈树的启发式搜索

- ▶ 博弈的概念: 博弈是一类具有智能行为的竞争活动, 如下棋和战争等.
- ▶ 博弈的类型
 - 双人完备信息博弈: 两位选手 (例如 MAX 和 MIN) 对垒, 轮流走步, 每一方不仅知道对方已经走过的棋步, 而且还能估计出对方未来的走步.
 - 机遇性博弈: 存在不可预测性的博弈, 例如掷币等.

- ▶ 博弈树：若把双人完备信息博弈过程用图表示出来，就得到一棵与/或树，这种与/或树被称为博弈树。在博弈树中，那些下一步该 MAX 走步的节点称为 MAX 节点，下一步该 MIN 走步的节点称为 MIN 节点。
- ▶ 博弈树的特点
 - (1) 博弈的初始状态是初始节点；
 - (2) 博弈树中的“或”节点和“与”节点是逐层交替出现的；
 - (3) 整个博弈过程始终站在某一方的立场上，例如 MAX 方。所有能使自己一方获胜的终局都是本原问题，相应的节点是可解节点；所有使对方获胜的终局都是不可解节点。

例 4.45 假设有七枚钱币任一选手只能将已分好的一堆钱币分成两堆个数不等的钱币，两位选手轮流进行，直到每一堆都只有一个或两个钱币，不能再分为止，哪个选手遇到不能再分的情况，则为输。

解：用数字序如上一个说明表示一个状态下一步的谁来分，如其中数字表示不同堆中钱币的个数表示 $(7, \text{MIN})$ 表示只有一个由七枚钱币组成的堆，由 MIN 来分，MIN 有三种可供选择的分法，即 $(6, 1, \text{MAX})$ 、 $(5, 2, \text{MAX})$ 、 $(4, 3, \text{MAX})$ ，其中 MAX 表示另一选手，不论哪一种方法，MAX 在它的基础上再做符合要求的划分，整个过程如图4-25所示，在图中已将双方可能的分法完全表示出来了，而且从中可以看出，无论 MIN 开始时怎么分法，MAX 存在可以获胜方案。MIN 取胜的策略用双箭头表示。

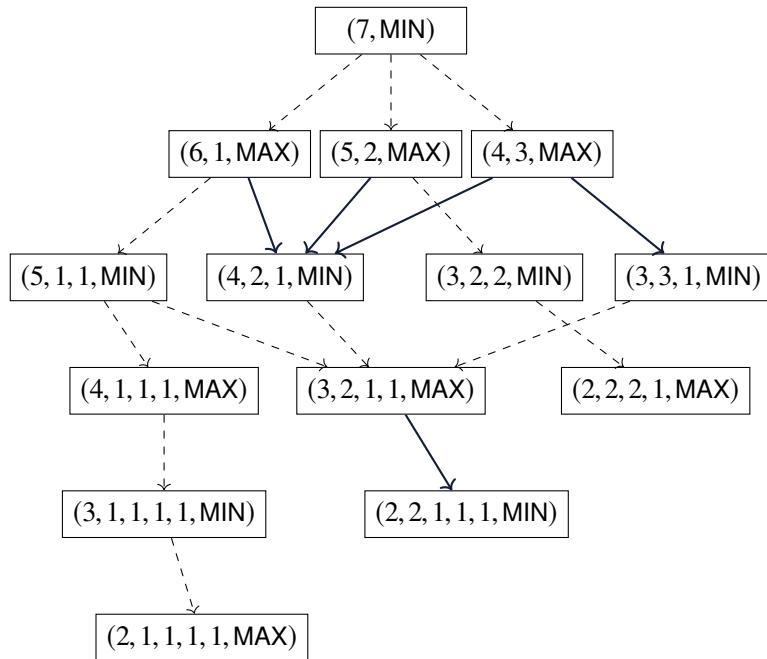


图 4-25 分钱币问题的博弈图

4.6.1 极大极小过程

对简单的博弈问题, 可生成整个博弈树, 找到必胜的策略.

- ▶ 对于复杂的博弈问题, 不可能生成整个搜索树, 如国际象棋, 大约有 10¹²⁰ 个节点. 一种可行的方法是用当前正在考察的节点生成一棵部分博弈树, 并利用估价函数 $f(n)$ 对叶节点进行静态估值.
- ▶ 对叶节点的估值方法是:
 - 1) 那些对 MAX 有利的节点, 其估价函数取正值;
 - 2) 那些对 MIN 有利的节点, 其估价函数取负值; 那些使双方均等的节点, 其估价函数取接近于 0 的值.
 - 3) 为非叶节点的值, 必须从叶节点开始向上倒退. 其倒退方法是:
 - ① 对于 MAX 节点, 由于 MAX 方总是选择估值最大的走步, 因此, MAX 节点的倒退值应取其后继节点估值的最大值.
 - ② 对于 MIN 节点, 由于 MIN 方总是选择估值最小的走步, 因此, MIN 节点的倒推值应取其后继节点估值的最小值.

这样一步一步的计算倒推值, 直至求出初始节点的倒推值为止. 这一过程称为极大极小过程.

例 4.46 (一子棋游戏) 设有一个三行三列的棋盘, 如下图所示, 两个棋手轮流走步, 每个棋手走步时往空格上摆一个自己的棋子, 谁先使自己的棋子成三子一线为赢.

设 **MAX** 方的棋子用 \times 标记, **MIN** 方的棋子用 \circ 标记, 并规定 **MAX** 方先走步.



图 4-26 $e(P)=e(+P)-e(-P)$

解: 估价函数 $e(+P)$: P 上有可能使 \times 成三子为一线的数目; $e(-P)$: P 上有可能使 \circ 成三子为一线的数目; 当 **MAX** 必胜 $e(P)$ 为正无穷大, **MIN** 必胜 $e(P)$ 为负无穷大.

棋局即估价函数: 具有对称性的棋盘可认为是同一棋盘. 如图4-27所示:

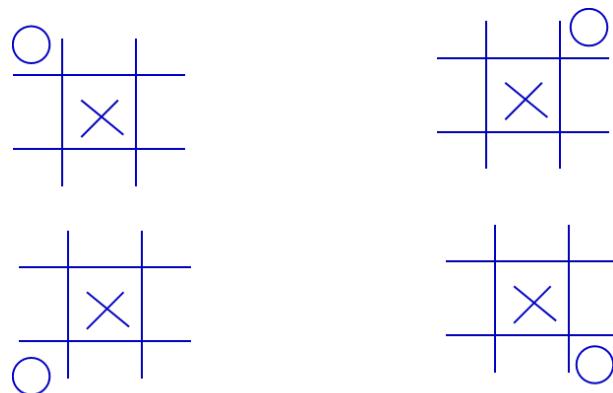


图 4-27 $e(P) = e(+P) - e(-P), e(P)=e(+P)-e(-P)=5-4=1..$

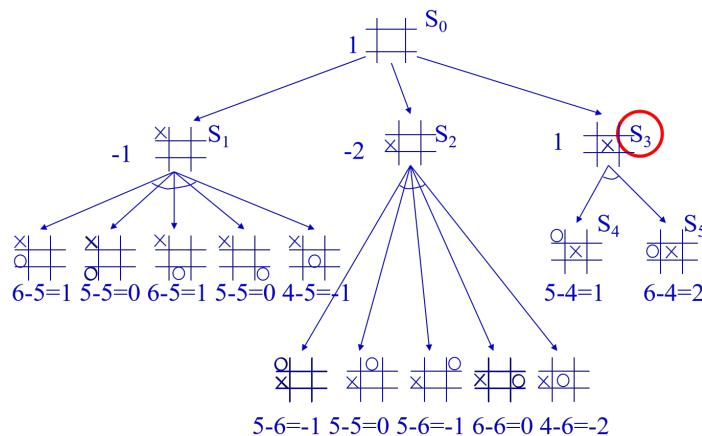


图 4-28 一子棋的极大极小搜索

4.6.2 alpha-beta 剪枝

极大极小过程是先生成与/或树，然后再计算各节点的估值，这种生成节点和计算估值相分离的搜索方式，需要生成规定深度内的所有节点，因此搜索效率较低。如果能边生成节点边对节点估值，并剪去一些没用的分枝，这种技术被称为 α - β 剪枝。

剪枝方法

(1) MAX 节点(或节点)的 α 值为当前子节点的最大倒推值；

(2) MIN 节点(与节点)的 β 值为当前子节点的最小倒推值；

(3) α - β 剪枝的规则如下：

- ▶ 任何 MAX 节点 n 的 α 值大于或等于它先辈节点的 β 值，则 n 以下的分枝可停止搜索，并令节点 n 的倒推值为 α 。这种剪枝称为 β 剪枝。
- ▶ 任何 MIN 节点 n 的 α 值小于或等于它先辈节点的 α 值，则 n 以下的分枝可停止搜索，并令节点 n 的倒推值为 β 。这种剪枝称为 α 剪枝。

例 4.47 一个 α - β 剪枝的具体例子，如图4-29所示。其中最下面一层端节点旁边的数字是假设的估值。

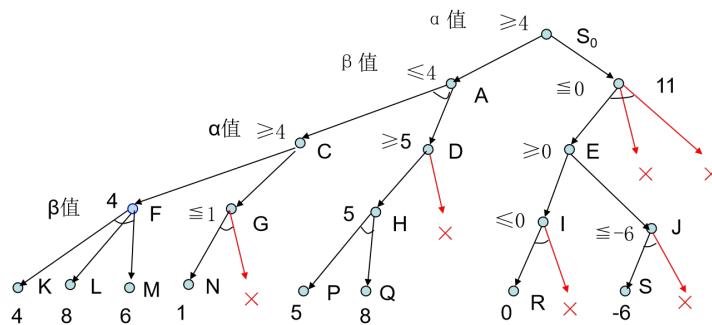


图 4-29 剪枝方法

在图4-29中 L, M, N 的估值推出节点 F 的倒推值为 4, 即 F 的 β 值为 4, 由此可推出节点 C 的倒推值 ≥ 4 . 记 C 的到推值的下界为 4, 不可能再比 4 小, 故 C 的 α 值为 4.

由节点 N 的估值推知节点 G 的倒推值小于 1, 无论 G 的其它子节点的估只是多少, G 的倒推值都不可能比 1 大. 因此, 1 是 G 的倒推值的上界, 所以 G 的值小于 1. 另已知 C 的倒推值大于 4, G 的其它子节点又不可能使 C 的倒推值增大. 因此对 G 的其它分支不必再搜索, 相当于把这些分枝剪去.

由 F, G 的倒推值可推出节点 C 的倒推值大于 4, 再由 C 可推出节点 A 的倒推值小于 4, 即 A 的 β 值为 4. 另外, 由节点 P, Q 推出的节点 I 的倒推值为 5, 因此 D 的倒推值大于 5, 即 D 的 α 值为 5. 此时, D 的其它子节点的倒推值无论是多少都不能使 D 及 A 的倒推值减少或增大, 所以 D 的其他分枝被减去, 并可确定 A 的倒推值为 4. 依此类推, 最终推出 S_0 的倒推值为 4.

4.7 作业

思考

何谓估价函数, 在估价函数中, $g(n)$ 和 $h(n)$ 各起什么作用?

思考

请用 A 算法求解图 3-12 所示的八数码问题.

思考

对图4-30所示的博弈树, 其中最后一行的数字是假设的估计值. 1) 计算各节点的倒推值. 2) 利用剪枝技术, 剪去不必要的分枝.

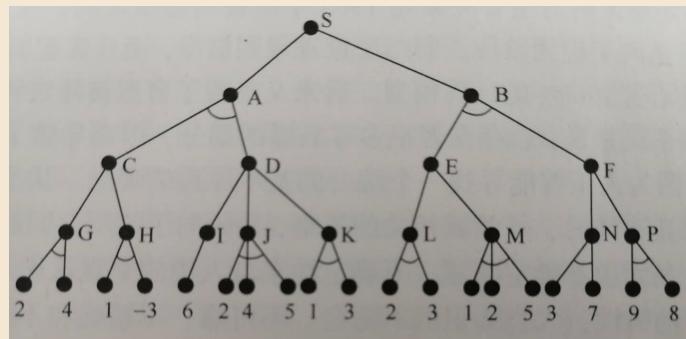


图 4-30 博弈树

思考

用 matlab 或 Python 语言实现图4-29博弈树的深度优先和宽度优先搜索算法, 并绘制流程图.

探索

设有如下结构的移动将牌游戏:

$B\ B\ W\ W\ E,$

其中, B 表示黑色将牌, W 表示白色将牌, E 表示空格. 游戏的规定走法是:

- (1) 任意一个将牌可移入相邻的空格, 规定其代价为 1;
 - (2) 任何一个将牌可相隔 1 个其它的将牌跳入空格, 其代价为跳过将牌的数目加 1.
- 游戏要达到的目标是把所有 W 都移到 B 的左边. 对这个问题, 请定义一个启发函数 $h(n)$, 并给出用这个启发函数产生的搜索树. 你能否判别这个启发函数是否满足下界要求? 在求出的搜索树中, 对所有节点是否满足单调限制?

5

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。



5.1 概述

本章主要讨论神经计算、进化计算和模糊计算问题.

5.1.1 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给出的定义:

定义 5.1 计算智能

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、接近或高于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.



注 5.2. 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.
- 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.
- 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.

注 5.3. 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, 以分布和并行计算为特征的自然智能模拟方法.

5.1.2 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) 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, 给出了图5-1所示的智能的层次结构.

在图5-1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 5-1 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.

5.1.3 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, 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。

2020 年的计算智能会议 WCCI2020 在英国的格拉斯哥 (19-24th July) 举办。会议包含三个分会, 2020 神经网络国际联合会议 (IJCNN2020), 2020 IEEE 模糊系统国际大会 (FUZZ-IEEE 2020) 和 2020 IEEE 进化计算代表大会 (IEEE CEC2020)。

5.2 浅层神经计算

浅层神经计算 (也叫浅层神经网络) 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并通过某种简单的非线性处理来进行转型, 神经网络的关键特性是基于经验修改单元间的相关权重。

常见误解—Stuart Russell

1) 「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。

2) 「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: 神经计算基础、人工神经网络的互连结构和神经网络的典型模型。

5.2.1 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, 先介绍生物

神经元的结构.

5.2.1.1 生物神经元的结构

神经元由细胞体 (Soma)、轴突 (Axon) 和树突 (Dendrite) 三个主要部分组成:

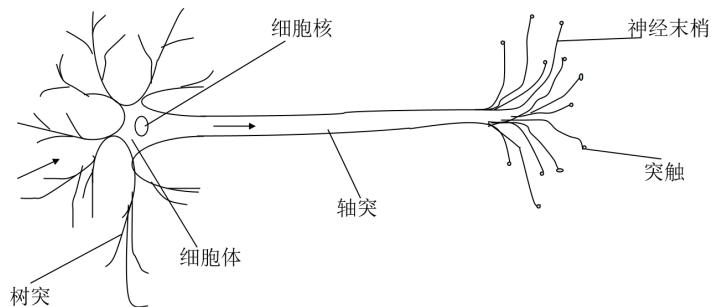


图 5-2 生物神经元结构

细胞体由细胞核、细胞质和细胞膜等组成, 其直径大约为 $0.5\text{--}100\ \mu\text{m}$ 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, 这些分支叫**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.

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.

 **注 5.4.** 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.

人脑神经系统的联结机制 简单介绍神经系统的联结规模和神经系统分布功能.

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 3×10^4 个突触. 小脑中的每个神经元大约有 10^5 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

人脑神经系统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.

5.2.2 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.

人工神经元的结构

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5-3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,

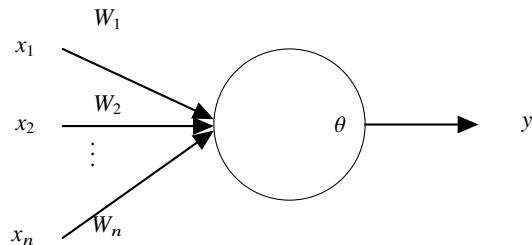


图 5-3 MP 神经元模型

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 σ 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, 输出就不再增加.
- ▶ S 型 (Sigmoid) 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) 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, 并产生反响. 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.

$$y_j = f \left(\sum_{i=1}^n w_{ij} x_i - \theta_j \right), j = 1, 2, \dots, m \quad (5.1)$$



图 5-4 MP 神经元模型

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5-5所示.



图 5-5 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:

$$y_j = f \left(\sum_{i=1}^n w_{ij} x_i - \theta_j \right), j = 1, 2, \dots, m \quad (5.2)$$

其中, 由所有连接权值 w_{ij} 构成的连接权值矩阵 W 为:

$$W = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & & \vdots \\ w_{nl} & w_{n2} & \cdots & w_{nm} \end{pmatrix} \quad (5.3)$$

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.

多层前馈网络 多层前馈网络是指那种除拥有输入和输出层外, 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对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.

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.

 **注 5.5.** 多层前馈网络的典型代表是 *BP* 网络.

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5-6. 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.

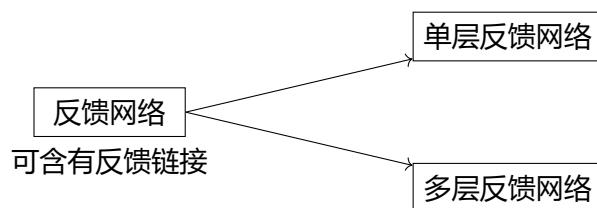


图 5-6 单层前馈网络结构

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.

注 5.6. 反馈网络的典型例子是后面将要介绍的 *Hopfield* 网络.

5.3 人工神经网络的典型模型

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上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.

基于神经网络的连接学习机制放到第七章学习部分讨论.

5.3.1 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.

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

$$y_j = f\left(\sum_{i=1}^n w_{ij}x_i - \theta_j\right), j = 1, 2, \dots, m \quad (5.4)$$

其中,由所有连接权值 w_{ji} 构成的连接权值矩阵 W 为

$$W = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{pmatrix} \quad (5.5)$$

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.

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:

$$\sum_{i=1}^n w_{ij} x_i - \theta_j = 0; j = 1, 2, \dots, m. \quad (5.6)$$

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 5.7 “与”运算 ($x_1 \wedge x_2$)

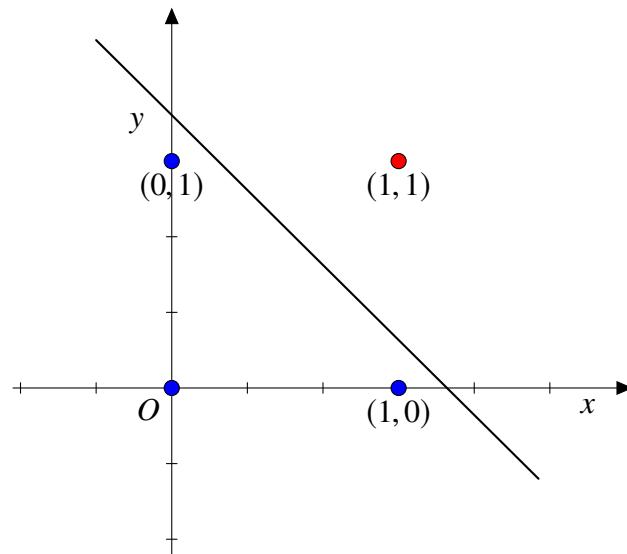


图 5-7 分类曲面

表 5-1 与运算

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

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5-7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 5.8 “非” 运算 ($\neg x_1$).

表5-3有解, 例如取 $w_1 = -1, \theta = -0.5$, 其分类结果如右图所示.

表 5-2 非运算

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

例 5.9 “异或” 运算 ($x_1 \text{ XOR } x_2$).

表 5-3 异或运算情况表

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

表5-3无解, 即无法找到满足条件的 w_1, w_2 和 θ , 如图5-8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.

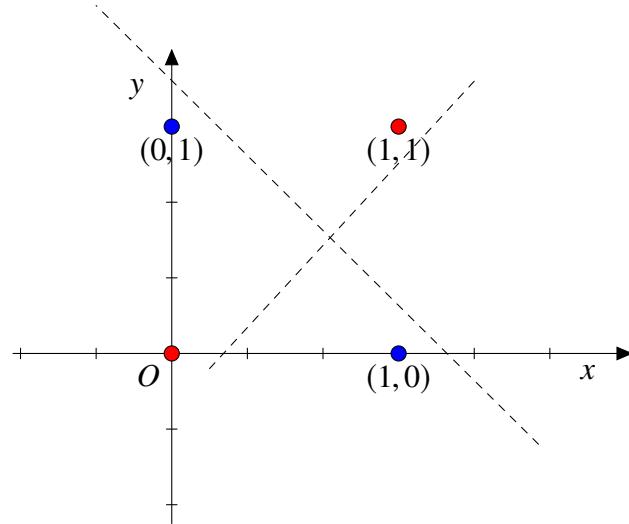


图 5-8 分类曲面

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5-9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.

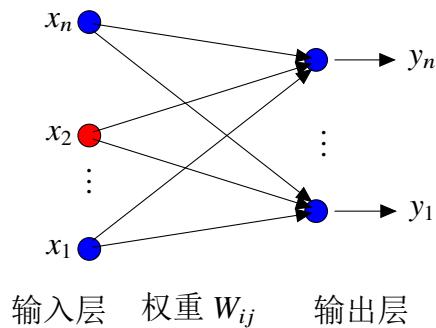


图 5-9 多层感知网络

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5-9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 5.10 对“异或”运算,用图5-10所示的多层感知器即可解决.

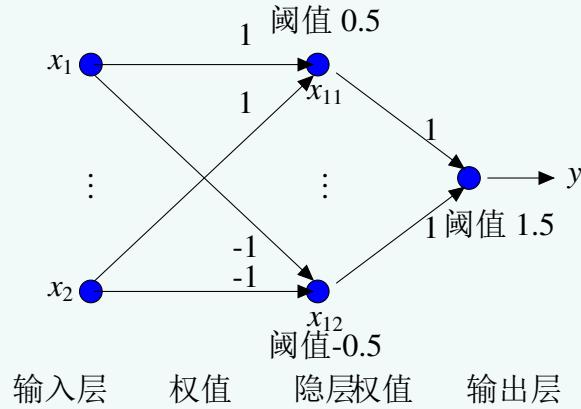


图 5-10 异或运算的多层感知网络

在图5-10中, 隐层神经元 x_{11} 所确定的直线方程为

$$1 \times x_1 + 1 \times x_2 - 0.5 = 0 \quad (5.7)$$

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

$$1 \times x_1 + 1 \times x_2 - 1.5 = 0 \quad (5.8)$$

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

$$1 \times x_{11} + 1 \times x_{12} - 1.5 = 0 \quad (5.9)$$

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5-11所示.

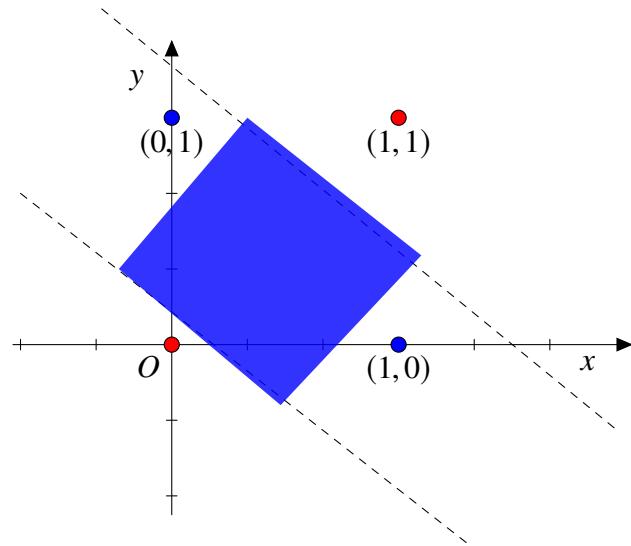


图 5-11

5.3.2 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 5-12 所示.

 **注 5.11.** 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.

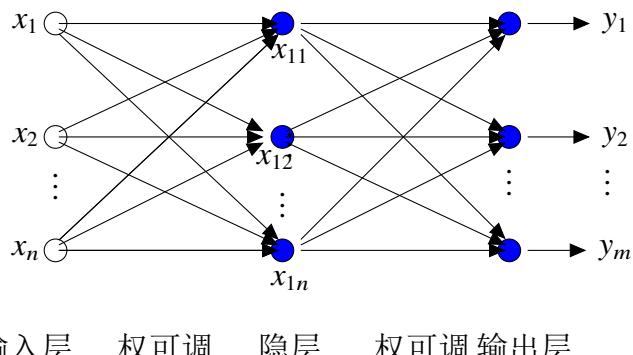


图 5-12 多层 BP 网络的结构

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:

$$f(x) = \frac{1}{1 + e^{-x}}. \quad (5.10)$$

- ▶ 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;

 **注 5.12.** 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.

5.3.3 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.

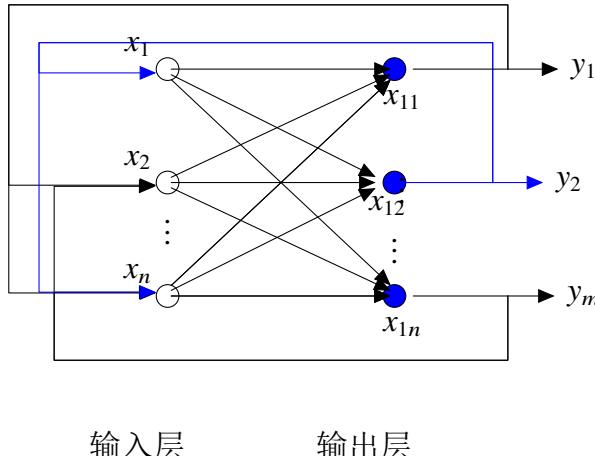


图 5-13 离散 Hopfield 网络

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 5-13 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

$$w_{ij} = \begin{cases} w_{ji}, & i \neq j \\ 0, & i = j \end{cases} \quad (5.11)$$

由该连接权值所构成的连接矩阵是一个零对角的对称矩阵. 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.

5.3.4 神经网络的泛逼近性——可视化证明

[Neural Networks: The Universality Theorem, V 1.0.0 \(655 KB\)](#) 作者: Mayank Jhamtani-Visual Proof of Universal Approximation Theorem for Neural Networks.

[Neural Networks_The Universality Theorem](#)

The original proof for this principle was given by George Cybenko in 1989, and it used concepts from a branch of abstract math called functional analysis. Michael Nielson gave a visual proof in his freely available online book [Nueral Nets and Deep Learning](#).

5.4 其它神经学习算法

最优剪枝极值学习机 (OP-ELM) 如图5-14, 图5-14显示了 OP-ELM 的三个步骤. 提出了 ([OP-ELM 工具箱找到](#)). 如 [[Miche2008OP](#), [Miche2008A](#), [Miche2010OP](#)] 中的描述, 对于 OP-ELM, 当给网络添加一个纯随机噪声变量 (添加的噪声变量没有在图上显示) 时, 拟合会变得松散和扩散. 实际上, ELM 的并没为这种情况设计. 本文的 OP-ELM 提出了一种三个步骤的方法来解决这个问题, 简言之, 为了解决含噪声的问题, 需要给 ELM 加一个含修剪策略的封装器, 之后增加了两个步骤. 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 首先是通过最小角度回归对神经元进行排序 (LARSefronhaste2004-32498); 实际上, 实现了 LARS 算法的 MRSR 算法也适用于多输出情况, 根据它们在输出值大小对它们进行排序. 然后, 用一个漏掉一个的准则来确定在最终的 OP-ELM 模型结构中需要保留多少个被分类的神经元. 这里将不对 **LARS algorithm** 做详细的叙述, 其实现来自 Lasso 方法, 对于 OP-ELM, 它提供了对隐藏神经元的排序方法, 这是由于 OP-ELM 算法设计总的神经元和输出之间的关系是线性的.



图 5-14 OP-ELM 的三部分框架结构: 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.

为了优化参数,(LOO) 方法通常是一种代价高昂的方法, 因为它需要在整个数据集(只除去一个样本)上训练模型, 并在此样本上对数据集的所有样本重复求值. 在 OP-ELM 结构中, 隐藏层和输出层的结点是线性关系, LOO 误差有一个封闭的矩阵形式, 由 Allen 的预测平方和 (PRESS) 给出 (关于压 LOO 误差的计算细节见第 4 节). 这种闭合形式允许快速计算均方误差, 从而计算输出权重 \mathbf{b} , 使得 OP-ELM 在计算上仍然有效, 并且比原始 ELM 对不相关/相关变量的鲁棒性更强, 因此, OP-ELM 可以被看作是一个“正则化” ELM. 通过使用一个 LARS 方法, 这里是对回归问题的 L_1 约束. 同时, 回归问题使用 L_1 和 L_2 (也包括 L_1 和 L_2 的联合形式) 惩罚项. 同时, 由于在压力公式中执行的矩阵运算的性质 (这些计算见第 4 节), 决定保留的神经元的最终数量 (通过 LOO criteria 索引 LOO criteria) 显示出潜在的不稳定性 (数值). 本文提出的解决方案是在压力公式的计算中使用正则化. 下面将回顾用于执行正则化的最著名算法, 回归问题使用 L_1 和 L_2 (也包括 L_1 和 L_2 的联合形式) 惩罚项. 第 4 节中提出的方法结合了 OP-ELM 中 L_1 和 L_2 的约束, 使网络正规化.

5.4.0.1 Tikhonov 正则化

Tikhonov 正则化以 Andrey Tikhonov 的名字命名, 是不适定问题最常用的正则化方法之一. 在统计学上, 这种方法被称为岭回归, 在有多个独立发现的情况下, 它也被称为 **Tikhonov-Miller method**、**Phillips-Twomey method**, 约束线性反演方法和线性正则化索引方法. 它与针对非线性回归最小二乘问题的 Levenberg-Marquardt 算法有关. 假设对于已知的矩阵 \mathbf{A} 和向量 \mathbf{b} , 我们希望找到向量 \mathbf{x} , 使得下式成立

$$\mathbf{Ax} = \mathbf{b}. \quad (5.12)$$

上述问题的标准解方法是最小二乘线性回归. 然而, 如果没有 \mathbf{x} 满足等式, 或者有多个 \mathbf{x} 满足等式, 则解决方案不是唯一的, 该问题称为不适定问题. 在这种情况下, 普通最

小二乘估计会导致方程组的超定(过拟合), 或者更常见的是欠定(欠拟合). 大多数现实世界的现象都具有向前方向的低通滤波器的效果, 其中 A 将 \mathbf{x} 映射到 \mathbf{b} . 因此, 在求解逆问题时, 逆映射用作高通滤波器, 具有放大噪声的不良倾向(特征值/奇异值在逆映射中最大, 而在正向映射中最小). 此外, 普通的最小二乘法隐式地将重构版本 \mathbf{x} 的每个元素(在 A 的空空间中)置空, 而不允许将模型用作 \mathbf{x} 的前置. 最小二乘法寻求最小化残差平方和, 其紧凑格式可写为

$$\|A\mathbf{x} - \mathbf{b}\|^2. \quad (5.13)$$

其中 $\|\cdot\|$ 是欧几里德范数. 为了得到具有期望性质的特定解, 对于适定的 Tikhonov 矩阵 Γ , 可在最小化问题中包括正则化项:

$$\|A\mathbf{x} - \mathbf{b}\|^2 + \|\Gamma\mathbf{x}\|^2. \quad (5.14)$$

许多情况下, 该矩阵被选为恒等矩阵 ($\Gamma = \alpha I$), 优先选择具有较小范数的解; 这是熟知的 L_2 正则化方法 [Ng2004-32636]. 在其他情况下, 则可以使用 `textbf{flowpass operators}`(例如, 差分运算符或 **weighted Fourier operator**) 来强制平滑. 这种正则化改进了问题的解, 从而使直接求得数值解成为可能. 由 $\hat{\mathbf{x}}$ 表示的显式解由以下公式给出:

$$\hat{\mathbf{x}} = (A^T A + \Gamma^T \Gamma)^{-1} A^T \mathbf{b}. \quad (5.15)$$

正则化的效果可以通过矩阵 Γ 的尺度变化. 对于 $\Gamma = 0$, 如果存在 $A^T A^{-1}$, 则问题退化为未正则化的最小二乘解. 除了线性回归之外, L_2 正则化还用于许多上下文中, 例如使用 logistic 回归或支持向量机分类 [Fan2008LIBLINEAR] 以及矩阵因式分解 [Guan2012Online]. Tikhonov 正则化是在许多不同的上下文中独立发明的. 从 Andrey Tikhonov 和 David L. Phillips 的工作中, 它在积分方程中的应用广为人知. 有些作者使用术语“Tikhonov-Phillips 正则化”. Arthur E.Hoerl 和 Manus Foster 采用统计方法, 阐述了有限维情况, 并将此方法解释为一个 **Wiener-Kolmogorov (Kriging)** 滤波器. 在 Hoerl 之后, 它在统计文献中被称为岭回归. **Generalized Tikhonov regularization**

对于 \mathbf{x} 的一般多元正态分布和数据误差, 可以应用变量转换来减少上述情况. 同样地, 可以寻找 \mathbf{x} 以最小化

$$\|A\mathbf{x} - \mathbf{b}\|_P^2 + \|\mathbf{x} - \mathbf{x}_0\|_Q^2 \quad (5.16)$$

其中, 我们使用 $\|\mathbf{x}\|_Q^2$ 表示加权范数 $\mathbf{x}^T Q \mathbf{x}$ (与 Mahalanobis 距离比较). 贝叶斯方法将 P 解释为 \mathbf{b} 的逆协方差矩阵, \mathbf{x}_0 是 \mathbf{x} 的期望值, 而 Q 是 \mathbf{x} 的逆协方差矩阵. 然后, 将

Tikhonov 矩阵作为矩阵 $\mathbf{Q} = \Gamma^T \Gamma$ (例如, **Cholesky factorization**) 的因子分解给出, 并将其视为一个 **whitening filter**.

这个广义问题有一个最优解 x , 可以用公式显式地求解

$$\mathbf{x}^* = (\mathbf{A}^T \mathbf{P} \mathbf{A} + \mathbf{Q})^{-1} (\mathbf{A}^T \mathbf{P} \mathbf{b} + \mathbf{Q} \mathbf{x}_0). \quad (5.17)$$

or equivalently

$$\mathbf{x}^* = \mathbf{x}_0 + (\mathbf{A}^T \mathbf{P} \mathbf{A} + \mathbf{Q})^{-1} (\mathbf{A}^T \mathbf{P} (\mathbf{b} - \mathbf{A} \mathbf{x}_0)). \quad (5.18)$$

5.4.1 TROP-ELM 算法

TROP-ELM: 基于 LARS 和 Tikhonov 正则化的双正则化 ELM[MichevanHeeswijk2011-30641], 2011. 图5-15显示了建议的正则化 OP-ELM (TROP-ELM) .

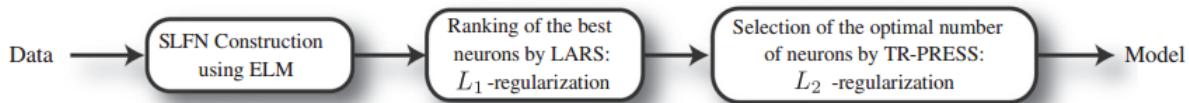


图 5-15 The proposed regularized OP-ELM (TROP-ELM) as a modification of Fig. 5-14.

5.4.2 Allen's PRESS(prediction sum of squares)

OP-ELM 中使用的原始 PRESS 公式是由 Allen 在 [MADavid1972] 中提出. 原始的 PRESS 公式可以表示为

$$\text{MSE}^{\text{TR-PRESS}} = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \mathbf{x}_i(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{x}^T \mathbf{y}}{1 - \mathbf{x}_i(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{x}_i^T} \right)^2, \quad (5.19)$$

从公式可以看出, 每个观测值都是用其他 $n - 1$ 观测值 “预测” 的, 残差最终被平方和相加. 算法通过矩阵运算, 有效地实现了该公式.

- 1: 计算矩阵的逆 $\mathbf{C} = (\mathbf{X}^T \mathbf{X})^{-1}$
- 2: 计算 $\mathbf{P} = \mathbf{X} \mathbf{C}$;
- 3: 计算伪逆 $\mathbf{w} = \mathbf{C} \mathbf{X}^T \mathbf{y}$;
- 4: 计算 PRES 的分子 $D = \mathbf{1} - \text{diag}(\mathbf{P} \mathbf{X}^T)$;
- 5: 计算 PRESS 的误差 $e = \frac{\mathbf{y} - \mathbf{X} \mathbf{w}}{D}$;
- 6: 优化误差 $\text{MSE}^{\text{TR-PRESS}} = \frac{1}{n} \sum_{i=1}^n \epsilon_i^2$

这种方法的主要缺点在于在计算中使用伪逆（在 **Moore-Penrose sense** 中），如果数据集 \mathbf{X} 不是全秩，则可能导致数值不稳定性。不幸的是，现实世界中的数据集经常出现这种情况。下面的方法对原始的计算提出了两个改进：正则化和快速矩阵计算。

5.4.2.1 Tikhonov 正则化的 PRESS (TR-PRESS)

在 [golub1979 generalized] 中，Golub 等人注意到：使用奇异值分解 (SVD) 方法计算 PRESS 的统计结果比传统的伪逆方法更好。在同一篇论文中，我们提出了 Allen's PRESS 的一个推广，即广义交叉验证 (GCV) 方法，它在技术上优于原始的方法，因为它可以处理数据定义非常糟糕的情况，例如，如果除对角线项外，所有的 X 项都是 0。

实际上，从我们的实验来看，虽然 GCV 具有无可替代的优越性，和原始的 PRESS 和 Tikhonov 正则化的 PRESS 相比，它会导致计算时间陡增。算法 1 以矩阵形式给出了用于确定 $MSE^{TR-PRESS}$ 的计算步骤

$$MSE^{TR-PRESS} = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \mathbf{x}_i(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{x}_i^T y}{1 - \mathbf{x}_i(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{x}_i^T} \right)^2 \quad (5.20)$$

这是公式 (18) 的正规化结果。符号 $A \circ B$ 表示矩阵 A 和 B (Schur 积) 逐元素相乘。在算法 2 的步骤 4 中使用它，因为它比标准矩阵乘积快。

用 SVD 分解 \mathbf{X} : $\mathbf{X} = \mathbf{U} \mathbf{S} \mathbf{V}^T$;

计算后面将要用到的乘积: $A = \mathbf{X} \mathbf{V}$ and $B = \mathbf{U}^T \mathbf{y}$;

重复如下步骤:

对 \mathbf{X} 用 SVD 分解，按如下方式计算矩阵 C :

计算 P : $P = C B$;

计算 D : $D = \mathbf{1} - \text{diag}(C U^T)$;

计算 $e = \frac{\mathbf{y} - P}{D}$ 和 $MSE^{TR-PRESS} = \frac{1}{n} \sum_{i=1}^n \epsilon_i^2$.

直到 λ 收敛。

关联 $MSE^{TR-PRESS}$ 和 λ 的最优值。

全局来说，该算法使用 \mathbf{X} 的 SVD 来避免计算问题，并在 SVD 计算伪逆时引入 Tikhonov 正则化参数。由于在优化 λ 之前预先计算了效用矩阵 (\mathbf{AB} 和 \mathbf{C})，所以此特定实现碰巧运行得非常快。在实际应用中，该算法中 λ 的优化是通过一种 **Nelder-Mead** 最小化方法来实现的，这种方法恰好在这个问题上收敛得非常快（在 Matlab 中是 fminsearch 函数）。通过使用这个修改版的 PRESS, OP-ELM 对回归权重（隐藏层和输出层之间的回归）有 L_2 的惩罚，对于这些权重，神经元已经使用 L_1 的惩罚进行了排序。图 5-15 是图的修改版本。图 5-14 说明了 **TROP-ELM** 方法。

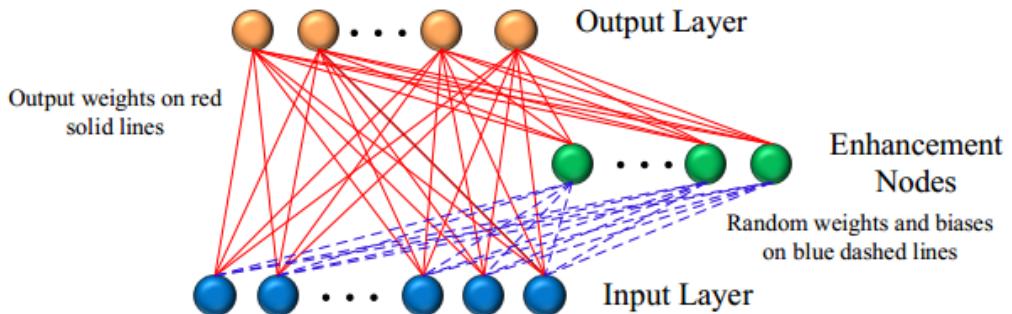


图 5–16 The architecture of the RVFL network.

5.4.3 区间二型 RVFL

现在我们使用新的学习范式来训练 RVFL 网络. 在这个新的训练集中, $x_i \in X$ 是原始功能, 一般来说, 特权功能 $x_i \in \tilde{x}$ 属于特权功能空间 \tilde{x} , 它不同于原始功能空间 x .

RVFL 的模型结构如此简单, 为什么 RVFL 能很好地完成大多数任务? Giryes 等人 [42]. 对这个开放问题给出一个可能的理论解释. 为了提供这个解释, 首先, Giryes 等人. 揭示学习算法训练的本质. 一般来说, 不同类别之间的样本角度大于同一类别内的样本角度 [43]. 因此, 训练在学习算法中的作用是“不同类别的点之间的角度比同一类别的点之间的角度受到的惩罚更大” [42]. 而且, 在这个大数据时代, 由于模型结构高度复杂, 学习过程中很难对所有参数进行快速调整, 这就需要极高的计算成本. 为了解决这个问题, 随机化是一些学习算法的理想选择, 从而降低了计算成本. 一个伟大的随机初始化允许学习算法是一个通用的训练前. 现在我们再次访问 RVFL. RVFL 使用混合策略来训练整个网络. 在 RVFL 中, 输入层和增强节点之间的随机初始参数处理具有可分辨角度的良好输入样本, 旋转输出权重进一步处理剩余数据. RVFL 网络是一种经典的单层前向神经网络, 其结构如图5–16所示. RVFL 随机初始化输入层和增强节点之间的所有权重和偏差, 然后这些参数在训练阶段不需要调整. 图5–16中红色实线代表的输出权重可以通过 Moore-Penrose 伪逆 ([paopillips1995-6471, IgelnikPao1995-6470]) 或岭回归 (引用 Bishop2012-6469) 来计算. 此外, 输入层和输出层之间的直接连接是防止 RVFL 网络过度拟合的一种有效而简单的正则化技术.

RVFL+ 是一种用于所有二元、多类别分类和回归问题的统一学习算法, 并且 (19) 中的输出函数可以直接应用于所有三种情况.

1. 二元分类: 测试样品的预测标签由

$$\hat{y} = \text{sign}(f_{test}(z)) \quad (5.21)$$

2. 多类分类: 我们采用一对多 (OvA) 策略来确定多类分类中的预测标签. 设 $f_{test^k} z$ 为第 k 个输出节点的输出函数. 测试样品的预期标签由

$$\hat{y} = \arg \max_{k \in 1, \dots, m} f_{test}^k(z) \quad (5.22)$$

3. 回归: 预测值等于 RVFL 的输出函数 $f_{test} z$

$$\hat{y} = f_{test}(z) \quad (5.23)$$

按照 [VAPNIK2009544] 中的 SVM+ 公式, 我们可以将 RVFL+ 写成

$$\begin{aligned} & \min_{w, \tilde{w}} \frac{1}{2} \|w\|_2^2 + \frac{\gamma}{2} \|\tilde{w}\|_2^2 + C \sum_{i=1}^N \zeta_i(\tilde{w}, \tilde{h}(\tilde{x}_i)) \\ & \text{s.t. } h(\tilde{x}_i)w = y_i - \zeta_i(\tilde{w}; \tilde{h}(\tilde{x}_i)), \forall i \leq N. \end{aligned} \quad (5.24)$$

其中 γ 是正则化系数. 与 $h x_i$ 类似, $\tilde{h} \tilde{x}_i$ 也是与特权功能 \tilde{x}_i 相对应的增强层输出向量, 可以用相同的方式计算. $\zeta_i \tilde{w}, \tilde{h} \tilde{x}_i$ 是特权功能空间中的更正函数 (或可宽延函数), 而 \tilde{w} 是更正函数的输出权重向量.

$$\zeta_i(\tilde{w}, \tilde{h}(\tilde{x}_i)) = \tilde{h}(\tilde{x}_i)w. \quad (5.25)$$

将(5.25)代入(5.24), 得到 RVFL+ 表达形式.

$$\begin{aligned} & \min_{w, \tilde{w}} \frac{1}{2} \|w\|_2^2 + \frac{\gamma}{2} \|\tilde{w}\|_2^2 + C \sum_{i=1}^N \tilde{h}(x_i)\tilde{w} \\ & \text{s.t. } h(\tilde{x}_i)w = y_i - \tilde{h}(x_i)\tilde{w}, \forall i \leq N. \end{aligned} \quad (5.26)$$

从(5.26), 我们注意到 RVFL+ 最小化了 w 和 \tilde{w} 上的目标函数. 因此, 训练阶段 RVFL+ 的分离超平面不仅取决于原始特征, 还取决于优先考虑的信息. 此外, 与 SVM+ 的原始模型相比 [VAPNIK2009544], RVFL+ 的修正函数是正的或负的. 换句话说, RVFL+ 不考虑约束组 $\zeta_i \tilde{h}, b, \phi x_i \geq 0 i = 1, \dots, N$. 对于二值分类, RVFL+ 中的约束数目至少比 SVM+ 少 N 个, 这使得 RVFL+ 的优化约束要比 SVM+ 的小得多. 此外, 为了解决(5.26)中的优化问题, 构造如下的 Lagrangian 函数 $\mathcal{L} w, \tilde{w}, \lambda$

$$\min_{w, \tilde{w}, \lambda} \frac{1}{2} \|w\|_2^2 + \frac{\gamma}{2} \|\tilde{w}\|_2^2 + C \sum_{i=1}^N \tilde{h}(x_i)\tilde{w} - \sum_{i=1}^N \lambda_i(h(\tilde{x}_i)w - y_i + \tilde{h}(x_i)\tilde{w}), \quad (5.27)$$

其中 $\lambda = [\lambda_1, \dots, \lambda_N]^T$ 是拉格朗日乘子。为了找到解，我们使用 KKT 条件来计算 Lagrangian 函数 $\mathcal{L}(w, \tilde{w}, \lambda)$ 关于变量 w, \tilde{w} 和 λ 的鞍点。

$$\frac{\partial \mathcal{L}(w, \tilde{w}, \lambda)}{\partial w} = 0 \Rightarrow w = H^T \lambda, \quad (5.28)$$

$$\frac{\partial \mathcal{L}(w, \tilde{w}, \lambda)}{\partial \tilde{w}} = 0 \Rightarrow \tilde{w} = \frac{1}{\gamma} (\tilde{H}^T \lambda - \tilde{H}^T C \mathbf{1}), \quad (5.29)$$

$$\frac{\partial \mathcal{L}(w, \tilde{w}, \lambda)}{\partial \lambda_i} = 0 \Rightarrow h(x_i)w - y_i + \tilde{h}(x_i)\tilde{w} = 0, 1 \leq i \leq N. \quad (5.30)$$

其中 $\mathbf{1} \in \mathbb{R}^{N \times m}$ 是单位矩阵。 \tilde{H} 也是来自增强节点的连接输出矩阵，它对应于 RVFL 的特性。将(5.28)和(5.29)代入(5.30)，得到

$$HH^T \lambda + \frac{1}{\gamma} \tilde{H} \tilde{H}^T (\lambda - C \mathbf{1}) = Y. \quad (5.31)$$

可以进一步将(5.31)重写为

$$\left(HH^T + \frac{1}{\gamma} \tilde{H} \tilde{H}^T \right) \lambda = Y - \frac{C \mathbf{1}}{\gamma} \tilde{H} \tilde{H}^T. \quad (5.32)$$

再由(5.28)和(5.32)，得到 RVFL+As 的最后闭式解

$$w = H^T \left(HH^T + \frac{1}{\gamma} \tilde{H} \tilde{H}^T \right)^{-1} \left(Y - \frac{C \mathbf{1}}{\gamma} \tilde{H} \tilde{H}^T \right). \quad (5.33)$$

根据岭回归方法 [Bishop2012-6469]，为了避免奇异性并保证 RVFL+ 的稳定性，我们还附加了一个 $\frac{I}{C}$ 项。因此，我们可以得到 RVFL+As 的最终闭式解

$$w = H^T \left(HH^T + \frac{1}{\gamma} \tilde{H} \tilde{H}^T + \frac{I}{C} \right)^{-1} \left(Y - \frac{C \mathbf{1}}{\gamma} \tilde{H} \tilde{H}^T \right). \quad (5.34)$$

RVFL+ 输出函数定义为

$$f(x) = h(x)w = h(x)H^T \left(HH^T + \frac{1}{\gamma} \tilde{H} \tilde{H}^T + \frac{I}{C} \right)^{-1} \left(Y - \frac{C \mathbf{1}}{\gamma} \tilde{H} \tilde{H}^T \right). \quad (5.35)$$

此外，在测试阶段，当使用测试数据 z 而不是训练数据 x 时，我们可以直接获得输出函数 $f_{test} z = hz - w$ 。

5.4.3.1 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。

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 5-17 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

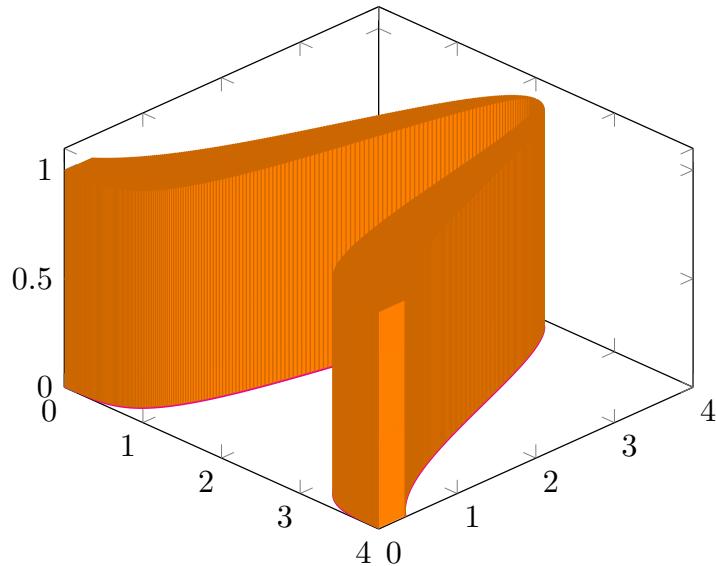


图 5-17 区间二型模糊集

$$\bar{\mu}_i(x_i) = \exp\left(-\frac{(x_i - m_i)^2}{\bar{\sigma}_i^2}\right), \quad (5.36)$$

$$\underline{\mu}_i(x_i) = \exp\left(-\frac{(x_i - m_i)^2}{\underline{\sigma}_i^2}\right), \quad (5.37)$$

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 5-18 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 5-18:

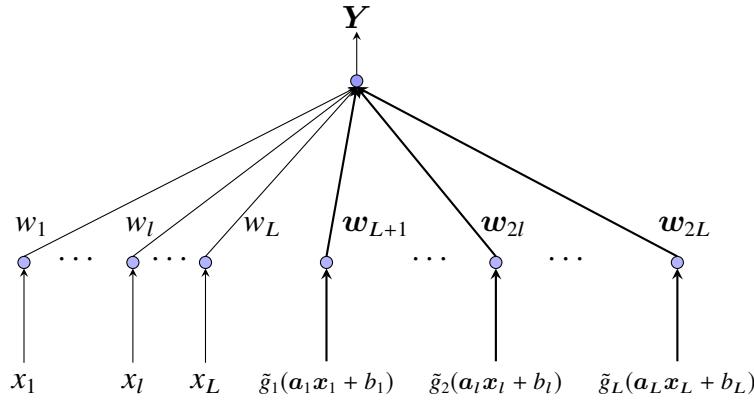


图 5-18 TT2-RVFL 的结构

$$Y = \mathcal{A} *_N X + H\Omega, \quad (5.38)$$

其中 $\mathcal{A} *_N X$ 为增强节点的张量乘积, $H\Omega$ 为线性部分的输出, X 为增强节点部分的权重矩阵; 类似于 RVFL, 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, Ω 是线性部分的权矩阵, 且 \mathcal{A} 是一个张量, 它是由叠加区间二型模糊集的映射结果得到的张量. 对于 TT2-RVFL, Y , X 和 Ω 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.

5.4.3.2 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 (5.36) 和 (5.37) 的变种, 记为 $\tilde{g}_i(\cdot)$, 可以用下面的式子表示为

$$\tilde{g}_i(x_i) = \exp(-k_1 s^2), \quad (5.39)$$

$$\underline{g}_i(x_i) = \exp(-k_2 s^2), \quad (5.40)$$

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

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

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

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

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

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

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 5.13.** 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

$$u_{UW}(x) = \frac{1}{2}(\underline{u}(x) + \bar{u}(x)) \cdot (1 + \underline{u}(x) - \bar{u}(x))^\gamma, \quad (5.41)$$

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

$$(1 + \underline{u}(x) - \bar{u}(x))^\gamma = \begin{cases} 0, & \underline{u}(x) = 0, \bar{u}(x) = 1, \\ 1, & \underline{u}(x) = \bar{u}(x). \end{cases} \quad (5.42)$$

方程 (5.42) 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 $\gamma > 1$, 线性将减少.

 **注 5.14.** 当 (5.42) 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 $\underline{\Phi}_{\cdot,\cdot,\cdot,1}$ 和 $\bar{\Phi}_{\cdot,\cdot,\cdot,2}$ 构建出来的. 接下来, 张量 Φ 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

$$\mathcal{X} = \begin{bmatrix} \mathbf{w}_{L+1,1} & \mathbf{w}_{L+1,2} \\ \mathbf{w}_{L+2,1} & \mathbf{w}_{L+2,2} \\ \vdots & \vdots \\ \mathbf{w}_{2L,1} & \mathbf{w}_{2L,2} \end{bmatrix}. \quad (5.43)$$

TT2-RVFL 的输出可用下式表示

$$\mathbf{Y} = \alpha \mathcal{A} *_N \mathcal{X} + (1 - \alpha) \mathbf{H} \boldsymbol{\Omega}, 0 \leq \alpha \leq 1, \quad (5.44)$$

其中 α 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; $\boldsymbol{\Omega} = [\boldsymbol{\omega} \ \boldsymbol{\omega}]$ 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 5.15.** 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 5.16.** 和 RVFL 相比, 基于张量的二型 RVFL 模型 (5.44) 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 $\boldsymbol{\Omega}$ 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.

5.4.3.3 TT2-RVFL 中的张量逆

张量方程 $\mathcal{A} *_N \mathcal{X} = \mathbf{Y}$ 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 5.17.** 方程 $\mathcal{A} *_N X = \mathcal{Y}$ 的解也是如下张量方程的解

$$\mathcal{A}^T *_N \mathcal{A} *_N X = \mathcal{A}^T *_N \mathcal{Y}, \quad (5.45)$$

其中张量 $\mathcal{A} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_N}$, $X \in \mathbb{R}^{J_1 \times \dots \times J_N \times K_1 \times \dots \times K_M}$ 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 $\mathcal{A} *_N X = \mathcal{Y}$ [HuangZhao2018NCAA-5838].

5.4.3.4 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 (5.44) 的最小范目标函数定义为

$$\min_{\Omega, X} \mathcal{E}_1(X) + \mathcal{E}_2(\Omega), \quad (5.46)$$

其中 $\mathcal{E}_1(X) = \|Y - \mathcal{A}X\|_F^2 + \|X\|_F^2$, $\mathcal{E}_2(\Omega) = \|y - H\Omega\|_2^2 + \|\Omega\|_2^2$. 模型 (5.44) 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 5.1 极小化模型的解

对于极小化模型 (5.44), 令 Ω^* 是 $y = H\Omega$ 的解集, X^* 是 $\mathcal{Y} = \mathcal{A} *_N X$ 的解. 对于模型 (5.46) 的两个子问题 $\mathcal{E}_1(X)$ 和 $\mathcal{E}_2(\Omega)$, \mathcal{Y} 退化为一个矩阵, 即 $\mathcal{Y} = [y \ y]$, 且 y 是训练结果列向量. 式 (5.44) 的充分条件就是寻找如下子问题的解

$$\min_X \mathcal{E}_1(X) \text{ 和 } \min_{\Omega} \mathcal{E}_2(\Omega), \quad (5.47)$$

则方程的解 $[X \ \Omega]^T$ 满足下式

$$Y = \mathcal{A} *_N X + H\Omega = [\mathcal{A} \ H] * \begin{bmatrix} X \\ \Omega \end{bmatrix}, \quad (5.48)$$

$$\begin{bmatrix} X \\ \Omega \end{bmatrix} = \begin{bmatrix} \alpha X^* \\ (1-\alpha)\Omega^* \end{bmatrix}, \quad (5.49)$$

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

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

 **注 5.18.** 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.

5.4.3.5 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 $k_1 = \frac{1}{\sigma_i^2}, k_2 = \frac{1}{\underline{\sigma}_i^2} (i = 1, 2, \dots, L)$ 求得.

5.4.3.6 非线性函数

例 4.1: 单输入 Sinc 函数的逼近问题如下

$$y = \begin{cases} 1, & x = 0 \\ \frac{\sin x}{x}, & x \neq 0 \end{cases} \quad x \in [-10, 10]. \quad (5.50)$$

对于 IT2RVFL 和 TT2-RVFL, 平衡参数设置为 $C = 2^{10}, \gamma = 0.95$. 表 5-4 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 5-4 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	2.36e-04	1.71e-05	2.32e-04	1.70e-05
35	IT2-RVFL	2.34e-02	9.15e-05	4.66e-02	3.17e-04
	TT2-RVFL	1.57e-02	7.44e-05	1.54e-02	7.61e-05
40	IT2-RVFL	2.18e-02	1.38e-04	4.57e-02	7.92e-04
	TT2-RVFL	3.86e-02	4.25e-04	3.76e-02	4.25e-04

例 4.2: 含噪的非线性函数具有如下形式

$$y = \frac{0.9x}{1.2x^3 + x + 0.3} + \eta, x \in [0, 1], \quad (5.51)$$

其中 η 是均匀分布的白噪声, η 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 IT2-RVFL 和 TT2-RVFL, 平衡参数设置为 $C = 2^{10}$, $\gamma = 1$. 表 5-5 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 5-5 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	7.30e-06	4.13e-01	5.58e-03
	TT2-RVFL	2.07e-01	8.43e-04	3.07e-01	2.19e-03
35	IT2-RVFL	2.85e-01	2.94e-06	4.15e-01	2.45e-03
	TT2-RVFL	2.07e-01	1.75e-03	3.06e-01	1.96e-03
40	IT2-RVFL	2.85e-01	5.14e-06	4.11e-01	5.10e-03
	TT2-RVFL	2.06e-01	4.54e-04	3.04e-01	1.13e-03

例 4.3: 双输入单输出 Sinc 函数的逼近问题.

$$z = \left| \frac{\sin x \cdot \sin y}{xy} \right|, (x, y) \in [-\pi, \pi]^2. \quad (5.52)$$

采用等距采样方式, 41×41 训练数据对用于训练, 30×30 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 5-6 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.

5.4.3.7 非线性系统参数识别

对于如下的非线性系统 [Rong2009-4804697]

$$y_p(k) = \frac{y_p(k-1)y_p(k-2)(y_p(k-1) + 2.5)}{1 + (y_p(k-1))^2 + (y_p(k-2))^2} + u(k-1). \quad (5.53)$$

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

$$\hat{y}_p(k) = \hat{f}(y_p(k-1), y_p(k-2), u(k-1)). \quad (5.54)$$

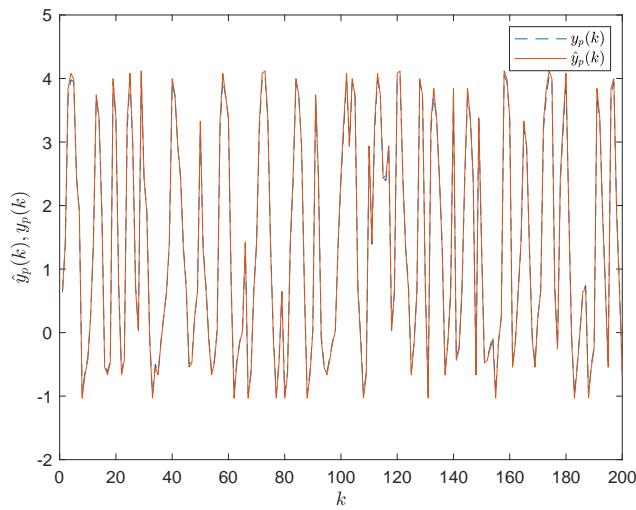


图 5-19 TT2-RVFL 对非线性系统识别问题的预测结果

表 5-6 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	3.09e-06	7.06e-02	1.43e-09
	TT2-RVFL	4.36e-02	1.30e-05	6.77e-02	6.05e-06
35	IT2-RVFL	4.63e-02	8.69e-06	7.06e-02	2.40e-08
	TT2-RVFL	4.07e-02	3.38e-05	6.73e-02	9.21e-06
40	IT2-RVFL	4.63e-02	2.95e-06	7.06e-02	1.56e-09
	TT2-RVFL	3.98e-02	4.72e-05	6.75e-02	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 5-19. 训练误差均值的 Frobenius 范数是 $1.87e-01$, 测试误差均值的 Frobenius 范数是 $3.94e-01$, 结果显示, TT2-RVFL 能够得到满意的精度.

5.4.3.8 回归问题

表 5-7 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	2.2161e-02	1.5915e-04	4.5358e-02	1.6692e-04
	TT2-RVFL	4.4055e-02	8.1940e-04	4.3847e-02	8.1024e-04
Bank	ELM	3.2400e-02	5.8710e-02	3.2103e-02	5.8425e-02
	RELM	3.3214e-02	5.8321e-02	3.0936e-02	5.8631e-02
	RVFL	1.1784e-01	6.3033e-03	5.0163e-02	1.3696e-04
	IT2-RVFL	2.9035e-02	3.4437e-05	7.0701e-02	9.9726e-05
	TT2-RVFL	4.3582e-02	1.1435e-04	5.1382e-02	8.3401e-05
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	8.6827e-02	1.0573e-02
	IT2-RVFL	1.4838e-01	2.6181e-05	1.5938e-01	4.3915e-04
	TT2-RVFL	1.4548e-01	2.1532e-06	1.4522e-01	8.8627e-05
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	2.2935e-02	1.3741e-02
	IT2-RVFL	8.2873e-02	6.5574e-05	4.0105e-01	3.9270e-03
	TT2-RVFL	1.0360e-01	3.1127e-04	2.6182e-01	1.0300e-03

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 5-7 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 5-5 的结果所示, TT2-RVFL 更适合于数据含噪的情形.

5.5 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.

定义 5.19 神经进化

神经进化是人工智能和机器学习领域的一个分支, 是一种机器学习技术, 它能够改进人工神经网络, 并且试图通过进化算法去寻找最优神经网络. 具体而言, 就像大自然通过突变、交叉或移除一些潜在的遗传密码来提高大脑的功能一样, 人工神经网络通过进化算法能够产生越来越好的拓扑、权重和超参数.

简单的说就是将进化的思想使用到神经网络参数优化的更迭中. 它使用基于群体的优化方法能够不断提高群体中每个神经网络的质量, 从而根据其要解决的问题生成越来越好的神经网络. 该种群中的每个个体的存储方式并不是复杂的神经网络, 而是存储为基因组. 基因组是一种简化的遗传表示, 可以映射到神经网络.

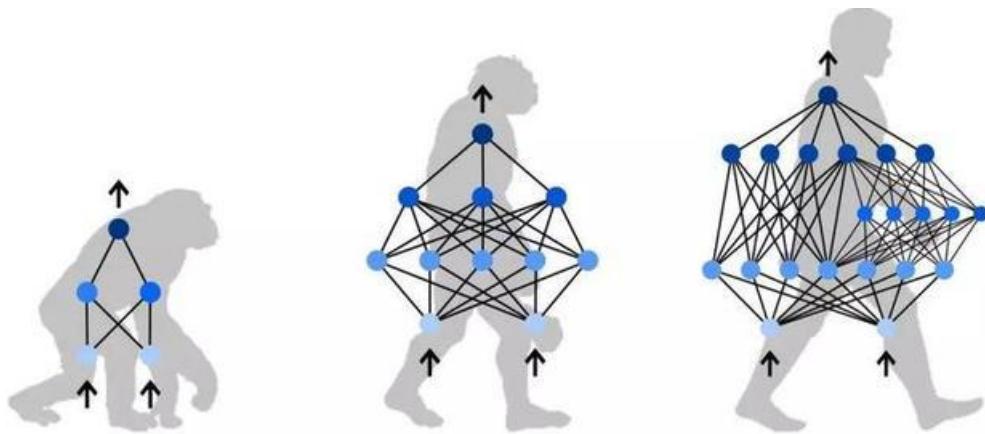


图 5-20 神经进化演化过程

进化计算主要包括一下四个分支:

- ▶ 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.

遗传算法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“物竟天择、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

定义 5.20 遗传

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一种生命现象.

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

定义 5.21 变异

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.

定义 5.22 变异

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:

1) 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.

2) 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

定义 5.23 进化

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。

5.5.1 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和学者的高度重视，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。

5.5.2 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5-21所示.

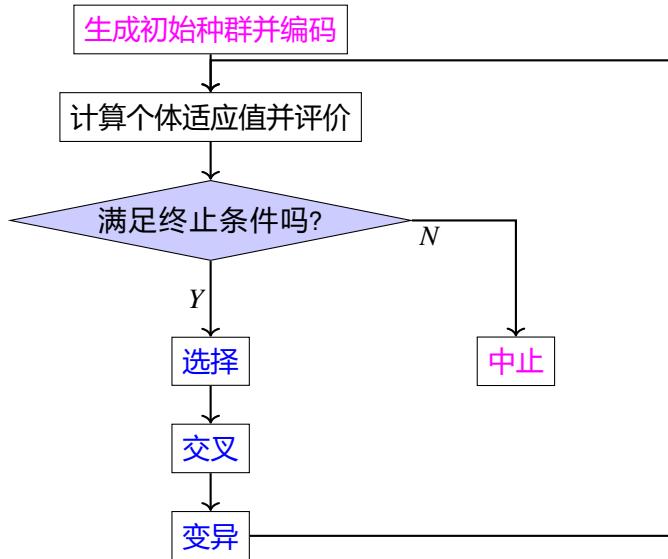


图 5-21 简单 GA 算法的流程图

5.6 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 5.24 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 5.25 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:

$$a_i = \begin{cases} b_1, & i = 1 \\ b_{i-1} \oplus b_i, & i > 1 \end{cases} \quad (5.55)$$

其中, 表示模 2 加法. 而从一个格雷串到二进制串的变换为:

$$b_i = \sum_{j=1}^i a_j (\text{mod} 2). \quad (5.56)$$

例 5.26 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

$$\max_{x \in [a,b]} f(x) \quad (5.57)$$

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, 并且适应度值越大越好. 这就往往需要对原始适应度函数进行某种变换, 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 $f_{\text{Normal}}(x)$.

极小化问题 对极小化问题, 其标准适应度函数可定义为

$$f(x) = \begin{cases} f_{\max}(x) - f(x), & f(x) < f_{\max}(x) \\ 0, & f(x) \geq f_{\max}(x) \end{cases} \quad (5.58)$$

其中, $f_{\max}(x)$ 是原始适应度函数 $f(x)$ 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.

极大化问题 对极大化问题, 其标准适应度函数可定义为

$$f(x) = \begin{cases} f(x) - f_{\min}(x), & f(x) > f_{\min}(x) \\ 0, & f(x) \leq f_{\min}(x) \end{cases} \quad (5.59)$$

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:

$$f'(x) = \alpha f(x) + \beta \quad (5.60)$$

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; α 和 β 是转换系数.

② 非线性加速

► 幂函数变换方法

$$f'(x) = f(x)^k \quad (5.61)$$

► 指数变换方法

$$f'(x) = \exp^{-\beta f(x)} \quad (5.62)$$

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：

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^N f(x_j)} \quad (5.63)$$

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：

$$2\pi \frac{f(x_i)}{\sum_{j=1}^N f(x_j)} = 2\pi p(x_i). \quad (5.64)$$

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 。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：

$$X = x_1 x_2 \cdots x_k x_{k+1} \cdots x_n \quad (5.65)$$

$$Y = y_1 y_2 \cdots y_k y_{k+1} \cdots y_n \quad (5.66)$$

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：

$$X' = x_1 x_2 \cdots x_k y_{k+1} \cdots y_n \quad (5.67)$$

$$Y' = y_1 y_2 \cdots y_k x_{k+1} \cdots x_n \quad (5.68)$$

例 5.27 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:

$$A' = 001110 \quad (5.69)$$

$$B' = 110001 \quad (5.70)$$

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:

$$X = x_1 x_2 \cdots x_i \cdots x_j \cdots x_n \quad (5.71)$$

$$Y = y_1 y_2 \cdots y_i \cdots y_j \cdots y_n \quad (5.72)$$

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:

$$X' = x_1 x_2 \cdots x_i y_{i+1} \cdots y_j x_{j+1} \cdots x_n \quad (5.73)$$

$$Y' = y_1 y_2 \cdots y_i x_{i+1} \cdots x_j y_{j+1} \cdots y_n \quad (5.74)$$

例 5.28 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:

$$A' = 001011 \quad (5.75)$$

$$B' = 110100 \quad (5.76)$$

④ 多点交叉

多点交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:

$$X' = x_1 x_2 \cdots x_i y_{i+1} \cdots y_j x_{j+1} \cdots x_k y_{k+1} \cdots y_n \quad (5.77)$$

$$Y' = y_1 y_2 \cdots y_i x_{i+1} \cdots x_j y_{j+1} \cdots y_k x_{k+1} \cdots x_n \quad (5.78)$$

例 5.29 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:

$$A' = 010100 \quad (5.79)$$

$$B' = 101011 \quad (5.80)$$

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 5.30 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:

$$X' = x_1 x_2 \cdots x_k y_{k+1} \cdots y_n \quad (5.81)$$

$$Y' = y_1 y_2 \cdots y_k x_{k+1} \cdots x_n \quad (5.82)$$

例 5.31 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:

$$A' = 20 16 19 12 21 30 \quad (5.83)$$

$$B' = 36 25 38 32 18 26 \quad (5.84)$$

(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 5.32 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 5.33 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时 2 和 4, 则变异后的新的个体向量是:

$$C' = 20\ 12\ 16\ 19\ 21\ 30 \quad (5.85)$$

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 5.34 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时 2 和 4, 则变异后的新的个体向量是:

$$D' = 20\ 19\ 16\ 12\ 21\ 30 \quad (5.86)$$

例 5.35 用遗传算法求函数 $f(x) = x_2$ 的最大值, 其中 $x \in \mathbb{Z}, x \in [0, 31]$ 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 $x = 01111$ 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:

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

(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:

$$f(s) = f(x) \quad (5.87)$$

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5-8所示.

表 5-8 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 5-9 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 π 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5-10所示.

表 5-10 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 5-11 种群情况表

$$\begin{aligned} S_{01} &= 10001 \\ S_{02} &= 11010 \\ S_{03} &= 01101 \\ \underline{S_{04}} &= 11001 \end{aligned}$$

(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 5-12 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 5-13 所示.

表 5-13 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 5-14 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 5-15 所示.

表 5-15 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 5-16 种群情况表

$$\begin{array}{l} \overline{S_{11}=10010} \\ S_{12}=11001 \\ S_{13}=11001 \\ \overline{S_{14}=11010} \end{array}$$

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 5-17 所示.

表 5-17 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 5-18 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 5-19 所示.

表 5-19 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 5-20 所示.

表 5-20 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		S22	3	11010 676
S_{22}	10010		S21	3	10001 289
S_{23}	11001		S24	4	11000 576
S_{24}	11110		S23	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 5-21 种群情况表

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

 **注 5.36.** 神经进化首先需要初始化一组上述基因组, 然后将它们应用于具体的问题环境中, 然后根据神经网络解决应用问题的能力为每个基因组分配一个适应度分数.

例 5.6.1

适应度分数可以是图像识别任务中实现的准确度以及机械臂移动实际轨迹和预期轨迹的差别等等.



一旦初始种群被创建, 优化循环开始, 种群不断地变异、重组、评估和经历自然选择. 如果这些步骤是迭代进行的, 而整个种群一次只进行一个步骤, 那么所进行的就是代际神经进化 (*generational neuroevolution*) .

竞争性共同进化 (*competitive coevolution*) 则意味着神经进化算法的设计允许异步性, 并且优化循环在每个基因组的基础上执行. 在代际神经进化和竞争性共同进化两种情况下, 其优化过程都是不间断的进行引入创新、评估创新、然后对创新进行分类, 直到产生一个最佳实用性神经网络.

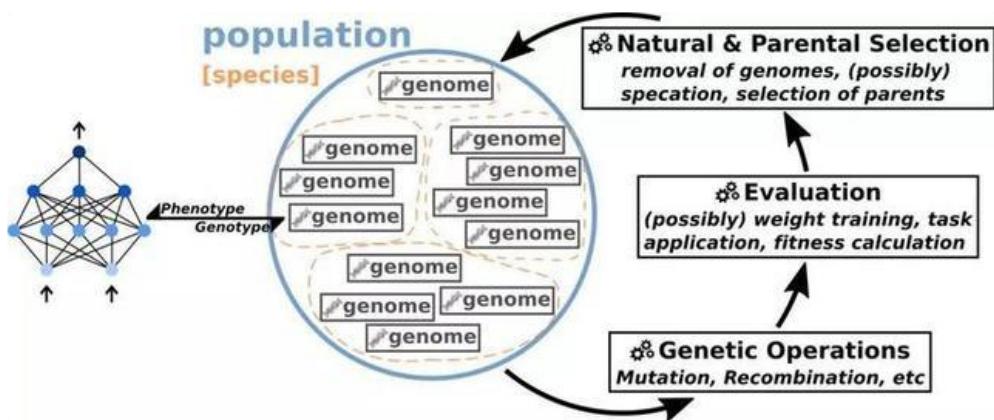


图 5-22 典型的代际神经进化过程图解

 **注 5.37.** 神经进化过程也是一个“黑盒”，虽然它自己的进化过程需要参数，但却不为生成的神经网络规定任何特定超参数，而是根据实际问题的解决设计神经网络。这便为神经网络的权重、超参数等的选择提供了范围，此范围也称为搜索空间。虽然“黑盒”性提供了非常广泛的搜索空间，但是为了提高遍历搜索空间的速度，明智的做法是限制搜索空间的粒度。通过限制基因组编码的复杂性，将基因组映射到搜索空间的粒度的能力也被称为遗传编码。综上所述，为了使搜索空间具有适当的粒度，根据实际问题的要求，设计遗传编码和相应的神经进化算法非常重要。先来回顾一下遗传编码的概念。

有效的神经网络是能够进行有效的变异和重组人工神经网络的前提。拥有强大表示能力的神经网络不用分析高度复杂的数据结构就能够快速的处理紧凑的遗传密码 (*compact genetic codes*)。换句话说，神经进化算法只在遗传编码上操作，而不是在机器学习框架中复杂的数据结构上操作。当然，基因编码允许这两种表示之间存在映射关系。基因组的这些有效的遗传表示被称为基因型 (*genotypes*)，而相应映射的神经网络被称为显型 (*phenotypes*)，这些术语来自遗传进化学。这里将所有显型都限制为神经网络。

基因编码一般可分为两个子类：直接编码和间接编码。虽然还有第三类发展性编码，但这种编码我们先忽略不计，毕竟这两年也没啥进展。直接编码表示神经网络的各个方面，它们在遗传表示中显式编码（如图 5-23 所示）。直接编码直接在基因型中编码每个连接及其相应的权重，但通过排除神经网络中的偏差和激活函数的可能性来限制搜索空间。这种直接编码可以表示任意的前馈和递归拓扑，也能够产生最优的拓扑。但“拓扑”太灵活的话，粒度的搜索空间就会变得非常庞大。因此需要设计良好的神经进化算法才能快速遍历该搜索空间。

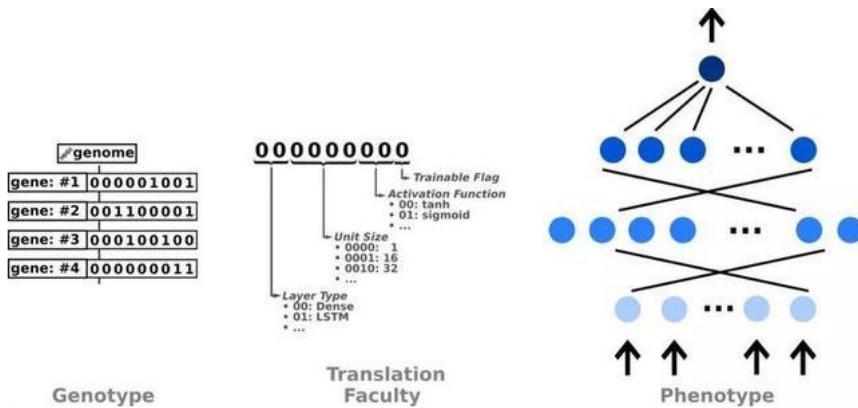


图 5-23 间接遗传编码

间接编码的功能是规定那些无法直接“翻译”成人工神经网络的自定义编码。也就是说为了将基因型映射到神经网络，需要一个由间接编码规定单独的“翻译”能力。如

果间接编码设计得当,那么即使神经网络非常复杂,也可以通过搜索空间实现有意义且快速的遍历。虽然可以从直接编码快速创建人工神经网络,但是缺少间接编码的翻译能力却会减慢处理速度,并且可能导致“粗粒度”。所以在决定使用哪种编码之前,必须考虑两种编码的优缺点。但是两种遗传编码都证明了遗传编码如何确定搜索空间的大小,

例 5.38 通过控制激活函数或某些层类型确定搜索空间。

 **注 5.39.** 有种遍历搜索空间的方法被称为繁殖的过程,这种方法与神经进化所使用的遗传编码密切相关。通常通过突变或重组基因组来创造新的基因组,新的基因组也继承了旧的基因组。突变让后代基因组探索人工神经网络新的结构、权重和超参数。重组基因组本质上是将两个基因组及其独特的特征合并。突变与遗传编码紧密相关,因为神经网络的参数只能突变到以遗传编码表示的程度。因此,为神经进化算法定义突变有以下三种情况。

- 1) 遗传编码的哪一部分会发生突变? 是拓扑变化、权重增减、还是超参数调节?
- 2) 基因组中选定的部分会发生多大程度的突变?

例 5.40 神经进化算法可以对低适应度基因组使用较大的突变,对高性能基因组使用微小的突变。

- 3) 突变采用何种方式,是定向还是随机?

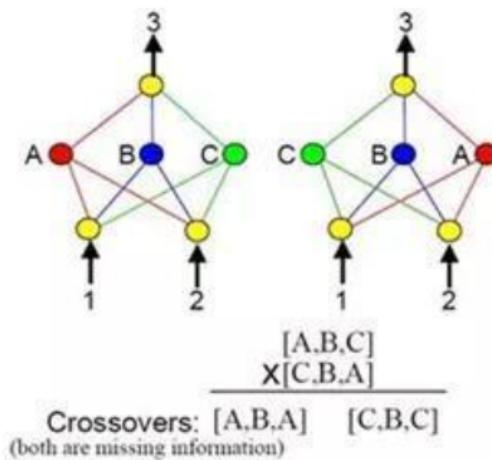


图 5-24 竞合公约问题图解

重组并不会突变基因组,拥有创新可以通过将两个亲本基因组及其独特特征结合,并产生“新颖”的后代基因组。如果重组方法设计得当,并且可以无损地融合两个亲本

基因组的有益特征, 将其在整个群体中传播, 提高所有现有基因组的适用性。设计重组方法的核心在于“无损融合”, 即不丢失任何基因特性的情况下融合。

例 5.41 在神经进化算法之前, NEAT 算法 (通过增强拓扑的进化神经网络, Evolving NN through Augmenting Topologies) 利用直接编码在进行修改网络的拓扑结构, 包括新增节点和删除节点等操作时会产生交叉损失。

如图 5-24 竞合公约问题 (*competing-conventions problem*) 所示。随后, NEAT 算法提出了一种“历史标记”的方法, 该方法为每个突变提供了唯一的标识符, 从而最终实现了基因组的无损重组, 使其为神经进化算法提供了基准。

在总体优化循环中, 基于问题评估基因组似乎是最简单的。但这一步骤确是非常重要, 不仅是能够指出潜在的改进和进步。评估方法从根本上说是一个过程, 即将基因组映射到由其遗传编码规定的神经网络, 并将其应用于问题环境, 然后根据神经网络的表现计算适应值。一些神经进化算法的评估过程还包括将神经网络加权训练的附加步骤, 虽然这种方法非常明智, 但只有当实际环境能够清晰反映基本信息才有用。在整个评估过程中, 尽管确定适应度的方式完全取决于实际问题的具体情况, 但可以进行合理的修改。

例 5.42 在图像识别中可以设置为准确度, 游戏中可以设置为点数。在确定适应度计算时, 新颖性搜索也是一个需要考虑的重要概念。因为这个概念涉及用新的方法奖励基因组, 能够使其具有更高的适应值。

例 5.43 在实际的电子游戏环境中的智能体如果进入一个未知的区域会获得体能提升, 尽管总体上获得的分数较少, 但也能促进了基因库的创新, 从而促进更有希望的进化。

5.7 QBFA

5.8 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。

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((1940.10.8-2016.7.15),福建福州人,数学家,中国科学院院士。主要从事拓扑学与不确定性(主要是模糊性)数学处理等方面的教学与科学研究。被誉为“中国的查德(模糊数学奠基人)”,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 5-25 汪培庄教授



图 5-26 刘应明院士

1983 年国际模糊系统协会筹备会在法国马 xi 举行,参加会议的我国学者有,刘应明(四川大学),汪培庄(北师大),王震源(河北大学),邓聚龙(华中科大)和我共五人。图为在会议组织委员会主席 Sanchez 家中合影。



图 5-27 1983 年国际模糊系统协会筹备会在法国马 xi 举行, 参加会议的我国学者有, 刘应明(四川大学), 汪培庄(北师大), 王震源(河北大学), 邓聚龙(华中科大)和张文修(西交大)

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 5.44 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

$$U \rightarrow [0, 1] \quad (5.88)$$

$$u \rightarrow \mu_F(u) \quad (5.89)$$

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 5.45.** ① 模糊集 F 完全是由隶属函数 μ_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 $\mu_F(u)$ 。

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。

5.8.1 模糊集及其运算

例 5.46 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:

$$\begin{aligned}\mu_F(20) &= 1, \mu_F(30) = 0.8, \mu_F(40) = 0.4 \\ \mu_F(50) &= 0.1, \mu_F(60) = 0\end{aligned}\quad (5.90)$$

则可得到刻画模糊概念“年轻”的模糊集

$$F = \{1, 0.8, 0.4, 0.1, 0\} \quad (5.91)$$

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:

$$F = \{\mu_F(u_1), \mu_F(u_2), \dots, \mu_F(u_n)\} \quad (5.92)$$

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

$$F = \mu_F(u_1)/u_1 + \mu_F(u_2)/u_2 + \dots + \mu_F(u_n)/u_n \quad (5.93)$$

也可写成

$$F = \sum_{i=1}^n \mu_F(u_i)/u_i \quad (5.94)$$

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.

$$\begin{aligned}F &= \{\mu_F(u_1)/u_1, \mu_F(u_2)/u_2, \dots, \mu_F(u_n)/u_n\} \\ F &= \{(\mu_F(u_1), u_1), (\mu_F(u_2), u_2), \dots, (\mu_F(u_n), u_n)\}\end{aligned}\quad (5.95)$$

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 5.47 前面例 5.15 的模糊集 F 可表示为:

$$F = 1/20 + 0.8/30 + 0.6/40 + 0.2/50 \quad (5.96)$$

有时, 模糊集也可写成如下两种形式:

$$\begin{aligned}F &= \{\mu_F(u_1)/u_1, \mu_F(u_2)/u_2, \dots, \mu_F(u_n)/u_n\} \\ F &= \{(\mu_F(u_1), u_1), (\mu_F(u_2), u_2), \dots, (\mu_F(u_n), u_n)\}\end{aligned}\quad (5.97)$$

其中, 前一种称为单点形式, 后一种称为序偶形式.

(2) 连续论域的表示方法

如果论域是连续的, 则其模糊集可用一个实函数来表示.

例 5.48 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

$$\begin{aligned}\mu_{\text{老}} &= \frac{(I)}{*_{\xi}}(u) = \begin{cases} 0, & 0 \leq u \leq 50 \\ \left[1 + \left(\frac{5}{u-50}\right)^2\right]^{-1}, & 50 < u \leq 100 \end{cases} \\ \mu_{\text{年轻}}(u) &= \begin{cases} 1, & 0 \leq u \leq 25 \\ \left[1 + \left(\frac{u-25}{5}\right)^2\right]^{-1}, & 25 < u \leq 100 \end{cases}\end{aligned}\quad (5.98)$$

(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:

$$F = \int_{u \in U} \mu_F(u)/u \quad (5.99)$$

 **注 5.49.** 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 5.50 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 5.51 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 5.52 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:

$$\begin{aligned}F \cup G : \mu_{F \cup G}(u) &= \max_{u \in U} \{\mu_F(u), \mu_G(u)\} \\ F \cap G : \mu_{F \cap G}(u) &= \min_{u \in U} \{\mu_F(u), \mu_G(u)\}\end{aligned}\quad (5.100)$$

定义 5.53 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:

$$\neg F : \mu_{\neg F}(u) = 1 - \mu_F(u) \quad (5.101)$$

定义 5.54 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

$$F = \{\text{小}\} = 1/1 + 0.6/2 + 0.1/3 \quad (5.102)$$

$$G = \{\text{大}\} = 0.1/1 + 0.6/2 + 1/3 \quad (5.103)$$

则

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

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

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

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.

5.8.2 模糊关系及其运算

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

$$\mathbf{V} \times \mathbf{W} = \{(v, w) | v \in V, w \in W\} \quad (5.104)$$

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

$$\mathbf{R} \subseteq \mathbf{V} \times \mathbf{W} \quad (5.105)$$

记为

$$V \xrightarrow{R} W \quad (5.106)$$

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 5.55 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

$$V \times W = \{(1\text{班}, \text{男队}), (2\text{班}, \text{男队}), (3\text{班}, \text{男队}), (1\text{班}, \text{女队}), (2\text{班}, \text{女队}), (3\text{班}, \text{女队})\} \quad (5.107)$$

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 5.56 模糊集

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

$$F_1 \times F_2 \times \dots \times F_n = \int_{u_1 \times u_2 \times \dots \times u_n} (\mu_{F_1}(u_1) \wedge \mu_{F_2}(u_2) \wedge \dots \wedge \mu_{F_n}(u_n)) / (u_1, u_2, \dots, u_n). \quad (5.108)$$

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 5.57 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

$$R = \int_{U_1 \times U_2 \times \dots \times U_n} \mu_R(u_1, u_2, \dots, u_n) / (u_1, u_2, \dots, u_n) \quad (5.109)$$

例 5.58 设有一组学生 $U = \{u_1, u_2\} = \{\text{秦学, 郝玩}\}$, 一些在计算机上的活动 $V = v_1, v_2, v_3 = \{\text{编程, 上网, 玩游戏}\}$ 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

$$\mu_R(u_i, v_j) = \mu_R \begin{cases} (u_1, v_1), (u_1, v_2), (u_1, v_3) \\ (u_2, v_1), (u_2, v_2), (u_2, v_3) \end{cases} \quad (5.110)$$

则 $U \times V$ 上的模糊关系 R 为

$$R = \begin{bmatrix} 0.9 & 0.6 & 0 \\ 0.2 & 0.3 & 0.8 \end{bmatrix} \quad (5.111)$$

5.8.3 模糊关系的合成

定义 5.59 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

$$\mu_{R_1 \circ R_2}(u, w) = \vee \{ \mu_{R_1}(u, v) \wedge \mu_{R_2}(v, w) \} \quad (5.112)$$

其中, \wedge 和 \vee 分别表示取最小和取最大.



例 5.60 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

$$R = R_1 \circ R_2 = \begin{bmatrix} 0.5 & 0.5 \\ 0.7 & 0.8 \end{bmatrix} \quad (5.113)$$

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 $R(i, j)$.

定义 5.61 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 5.62 设 $F = (1, 0.6, 0.2)$

$$R = \begin{bmatrix} 1 & 0.5 & 0 & 0 \\ 0.5 & 1 & 0.5 & 0 \\ 0 & 0.5 & 1 & 0.5 \end{bmatrix} \quad (5.114)$$

则

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

5.9 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

$$\begin{aligned} F &= 0.9/u_1 + 0.7/u_2 + 0.5/u_3 + 0.3/u_4 \\ G &= 0.6/u_3 + 0.8/u_4 + 1/u_5 \end{aligned}$$

请分别计算 $F \cap G, F \cup G, \neg F$.

探索

试用 M-P 模型构造一个实现逻辑“OR”运算的神经元, 给出模型结构和各个 w_i 和 θ 值.

探索

试证明感知机学习算法收敛定理.

思考

描述支持向量机的基本思想, 给出伪代码.

思考

试解释强化学习模型及与其他机器学习方法的异同.

思考

什么是 Q 学习? 它的基本原理是什么?

思考

说明遗传算法的构成要素, 给出遗传算法流程图.

思考

简述蚁群算法的原理, 用蚁群算法求解旅行商问题.

探索

1. 试将感知机学习算法用 C 语言编成程序, 并做下述的 n 维随机矢量 $\mathbf{X} = [x_1, x_2, \dots, x_n]^T$ 的二值分类的模拟实验:

(a) 用程序产生 M 个均值为 0, 方差为 1 的正态随机矢量 (取维数 $n = 3$, 即 $\mathbf{X}(k) = [x_1(k), x_2(k), x_3(k)]^T$, $k = 1, 2, \dots, M$; 每个 $x_i(k)$ 为服从 $N(0, 1)$ 分布的随机变量). 要求产生三组矢量 (分别取 $M=10, 20, 30$), 分别用每组矢量训练一个感知机模型. 对于每个训练矢量 $\mathbf{X}(k)$, 给定其理想输出为 $d(k) = \begin{cases} 1, & x_2(k) \geq 0 \\ 0, & x_2(k) < 0 \end{cases}$. 在每组训练收敛后, 再产生 30 个新矢量, 用来检验所得到的感知机的分类性能. 对每一组结果要给出收敛时所用的迭代次数 K_0 , 收敛时的权矢量值 $\mathbf{W}(K_0)$, 和检验时所达到的正确分类率 R .

(b) 取维数 $n = 5$, 做与 (1) 相同的模拟实验 (仍产生 $M = 10, 20, 30$ 的三组正态随机矢量, 给出相应的结果. 在本题中, 对于每个 5 维训练矢量 $\mathbf{X}(k)$, 其理想输出 $d(k)$ 为 $d(k) = \begin{cases} 1, & x_3(k) \geq 0 \\ 0, & x_3(k) < 0 \end{cases}$.

• 提示: 产生正态随机数 $X \sim N(0, 1)$ 的方法:

- (a) 先产生在 $(0, 1)$ 区间均匀分布的两个随机数 U_1 和 U_2 ;
- (b) 令 $X_1 = (-2 \ln U_1)^{1/2} \sin(2\pi U_2)$, $X_2 = (-2 \ln U_1)^{1/2} \cos(2\pi U_2)$.

则 X_1 和 X_2 均为服从 $N(0, 1)$ 分布的正态随机数.

探索

1. 在 BP 学习算法中, 若各单元均取为 Sigmoid 函数, 则网络输出值必处于 (0,1) 之间. 为使输出值不限于 (0,1) 之间, 可选输出单元为线性函数 (即 $\phi(s) = s$) , 其它单元仍取为 Sigmoid 函数. 试导出此时的 BP 学习算法公式.
 - (a) (1) 已知用一个 2 层的 NN 可实现逻辑 XOR 运算. 试构造一个实现 XOR 运算的 NN, 给出模型结构和各个 w_{ji} 和 θ_j 值. (注: 在该 NN 中, 输入 $X = [x_1, x_2]^T$ 中各 x_i 取值为 +1 或 -1, 各单元的非线性函数取为符号函数, 即 $\phi(s) = Sgn(s)$) .
 - (2) 奇偶检验问题可视为 XOR 问题的推广 (由 2 输入到 n 输入的推广): 若 n 个输入中有奇数个 1, 则输出为 1; 若 n 个输入中有偶数个 1, 则输出为 0. 一个 2 层的 NN 可实现奇偶检验运算. 试构造一个实现这种运算的 NN, 给出模型结构和各个 w_{ji} 和 θ_j 值. (注: 在该 NN 中, n 为任意整数, 输入 $X = [x_1, x_2, \dots, x_n]^T$ 中各 x_i 取值为 1 或 0, 各单元的非线性函数取为单位阶跃函数, 即 $\phi(s) = U(s)$).

探索

1. 试用 C 语言编程实现多层前向 NN 的 BP 算法. 要求: 输入、输出结点数目, 隐层数目, 及各隐层中结点的数目应为任意整数.
2. 试用所编出的 BP 算法程序训练出一个实现 XOR 运算的 2 层前向网络.
3. 用所编出的 BP 算法程序训练出输入矢量的维数分别为 $n = 7$ 和 $n = 8$ 的两个实现奇偶检验运算的 2 层前向 NN.
 - 注: 要求: 列表给出训练收敛后的 NN 权值和所用的迭代次数;
4. 给出训练收敛后的训练误差和检验误差, 及用训练集和检验集做输入时所得到的正确输出率;
 - (a) 给出 NN 的学习曲线 (即 $E(W(k))$ 随迭代次数 k 的变化曲线, 该结果应是用计算程序计算和打印出来的曲线, 要是用手画出的曲线) .

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.

6

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。



6.1 不确定性推理的含义

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, 最终推出具有不确定性但却又是合理或基本合理结论的思维过程.

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.

6.2 不确定性推理的基本问题

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.

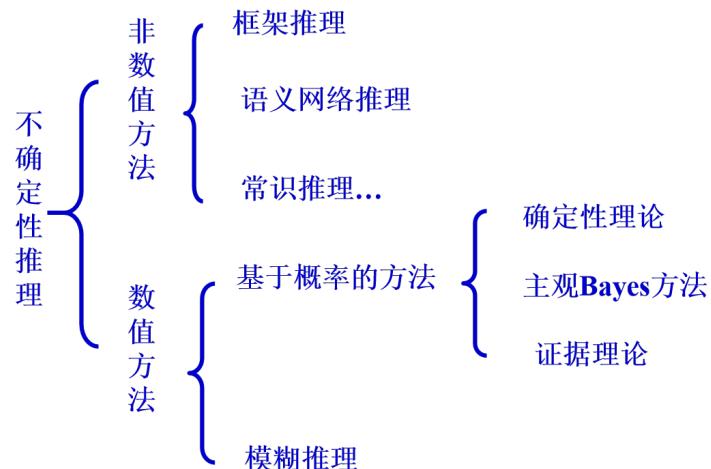


图 6-1 不确定方法

6.3 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 6.1 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 6.2 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 6.3 频率的概念

统计概率是通过某一事件出现的频率定义的. 频率定义为

$$f_n(A) = m/n \quad (6.1)$$

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 6.4 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

$$P(A) = p. \quad (6.2)$$



例 6.5 在掷币试验中, 当掷币次数足够多时有

$$f_n(\text{正面向上}) = 0.5. \quad (6.3)$$

则称正面向上的概率为 0.5, 即

$$P(\text{正面向上}) = 0.5. \quad (6.4)$$

定义 6.6 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 6.7 条件概率

设 A 与 B 是两个随机事件, $P(B) > 0$, 则称:

$$P(A|B) = P(A \cap B)/P(B) \quad (6.5)$$

为在事件 B 发生的条件下事件 A 的条件概率.



例 6.8 设样本空间 D 是扑克牌中的 54 张牌, 即 $D=\{\text{红桃 A}, \text{方块 A}, \text{黑桃 A}, \text{梅花 A}, \text{红桃} 2, \text{方块} 2, \dots, \text{小王}, \text{大王}\}$, 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; …; 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 6.1 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 $A_i \cap A_j = \emptyset$, ($i = 1, 2, \dots, n$);
- (2) $P(A_i) > 0$ ($i = 1, 2, \dots, n$);
- (3) $D = \bigcup_{n=1}^n A_n$, 则对任何事件 B 由下式成立:

$$P(B) = \sum_{i=1}^n P(A_i) \times P(B|A_i). \quad (6.6)$$

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 6.2 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:

$$P(A_i|B) = \frac{P(A_i) \times P(B|A_i)}{\sum_{j=1}^n P(A_j) \times P(B|A_j)}, i = 1, 2, \dots, n. \quad (6.7)$$

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:

$$P(A_i|B) = \frac{P(A) \times P(B|A_i)}{P(B)}, i = 1, 2, \dots, n, \quad (6.8)$$

即

$$P(A_i|B) \times P(B) = P(B|A) \times P(A_i), i = 1, 2, \dots, n. \quad (6.9)$$

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.

6.3.1 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 6.9 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.

6.3.2 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:

IF E THEN H (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 6.10.** (1) E 可以是单一条件, 也可以是复合条件.

例 6.11 $E = (E_1 \text{ OR } E_2) \text{ AND } E_3 \text{ AND } E_4$.

(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 6.12 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.

6.3.3 可信度的定义与性质

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

$$MB(H, E) = \begin{cases} 1, & \text{若 } P(H) = 1 \\ \frac{\max\{P(H|E), P(H)\} - P(H)}{1 - P(H)}, & \text{否则} \end{cases}, \quad (6.10)$$

MD 称为不信任增长度, $MD(H, E)$ 定义为

$$MD(H, E) = \begin{cases} 1, & \text{若 } P(H) = 0 \\ \frac{\min\{P(H|E), P(H)\} - P(H)}{-P(H)}, & \text{否则} \end{cases} \quad (6.11)$$

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:

$$CF(H, E) = \begin{cases} MB(H, E) - 0 = \frac{P(H|E) - P(H)}{1 - P(H)} & \text{若 } P(H|E) > P(H) \\ 0 & \text{若 } P(H|E) = P(H) \\ 0 - MD(H, E) = -\frac{P(H) - P(H|E)}{P(H)} & \text{若 } P(H|E) < P(H) \end{cases} \quad (6.12)$$

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 的信任增长度等于对非 H 的不信任增长度.

根据 MB 、 MD 的定义及概率的性质有:

$$\begin{aligned} MD(\neg H, E) &= \frac{P(\neg H|E) - P(\neg H)}{-P(\neg H)} = \frac{(1 - P(H|E)) - (1 - P(H))}{-(1 - P(H))} \\ &= \frac{-P(H|E) + P(H)}{-(1 - P(H))} = \frac{-(P(H|E) - P(H))}{-(1 - P(H))} \\ &= \frac{P(H|E) - P(H)}{1 - P(H)} = MB(H, E). \text{信任增长度} \end{aligned} \quad (6.13)$$

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 的信任增长度等于对非 H 的不信任增长度.
- (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

$$\sum_{i=1}^n CF(H_i, E) \leq 1,$$

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.

6.4 证据理论

6.4.1 证据不确定性的表示

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.

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.

否定证据不确定性的计算

$$CF(\neg E) = -CF(E). \quad (6.14)$$

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

$$CF(E) = \min\{CF(E_1), CF(E_2), \dots, CF(E_n)\}. \quad (6.15)$$

► 析取

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

$$CF(E) = \max\{CF(E_1), CF(E_2), \dots, CF(E_n)\}. \quad (6.16)$$

6.4.2 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

$$CF(H) = CF(H, E) \times \max\{0, CF(E)\}. \quad (6.17)$$

若 $CF(E) < 0$, 则

$$CF(H) = 0, \quad (6.18)$$

即该模型没考虑 E 为假对 H 的影响.

若 $CF(E) = 1$, 则

$$CF(H) = CF(H, E), \quad (6.19)$$

即规则强度 $CF(H, E)$ 实际上是在 E 为真时, H 的可信度.

6.4.3 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:

$$\text{IF } E_1 \text{ THEN } H(CF(H, E_1)); \quad (6.20)$$

$$\text{IF } E_2 \text{ THEN } H(CF(H, E_2)). \quad (6.21)$$

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

$$CF_1(H) = CF(H, E_1) \times \max\{0, CF(E_1)\}; \quad (6.22)$$

$$CF_2(H) = CF(H, E_2) \times \max\{0, CF(E_2)\}. \quad (6.23)$$

(2) 用如下公式求 E_1 与 E_2 对 H 的综合可信度

$$CF(H) = \begin{cases} CF_1(H) + CF_2(H) - CF_1(H) \times CF_2(H) & \text{若 } CF_1(H) \geq 0 \\ & \text{且 } CF_2(H) \geq 0 \\ CF_1(H) + CF_2(H) + CH_1(H) \times CF_2(H) & \text{若 } CF_1(H) < 0 \\ CF_1(H) + CF_2(H) & \text{若 } CF(H) < 0 \\ \frac{CF_1(H) + CF_2(H)}{1 - \min\{CF_1(H), |CF_2(H)|\}} & \text{且 } CF_2(H) \neq 0 \end{cases}. \quad (6.24)$$

例 6.13 设有如下一组知识:

```
$r_1$: IF E1 THEN H(0.9);
$r_2$: IF E2 THEN H(0.6);
$r_3$: IF E3 THEN H(-0.5);
$r_4$: IF E4 AND (E5 OR E6) THEN E1(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:

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

由 r_2 得到:

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

由 r_3 得到:

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

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

$$\begin{aligned} CF_{1,2}(H) &= CF_1(H) + CF_2(H) - CF_1(H) \times CF_2(H) \\ &= 0.36 + 0.48 - 0.36 \times 0.48 \\ &= 0.84 - 0.17 = 0.67. \end{aligned} \quad (6.25)$$

$CF_{12}(H)$ 和 $CF_3(H)$ 异号, 有:

$$\begin{aligned} CF_{1,2,3}(H) &= \frac{CF_{1,2}(H) + CF_3(H)}{1 - \min\{|CF_{1,2}(H)|, |CF_3(H)|\}} \\ &= \frac{0.67 - 0.3}{1 - \min\{0.67, 0.3\}} = \frac{0.37}{0.7} \\ &= 0.53. \end{aligned} \quad (6.26)$$

6.5 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:

$$\text{IF } E \text{ THEN (LS, LN) } H, \quad (6.27)$$

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:

$$\begin{aligned} LS &= \frac{P(E|H)}{P(E|\neg H)}, \\ LN &= \frac{P(\neg E|H)}{P(\neg E|\neg H)} = \frac{1 - P(E|H)}{1 - P(E|\neg H)}. \end{aligned} \quad (6.28)$$

下面进一步讨论 LS 和 LN 的含义. 由本章第6.2节的 Bayes 公式可知:

$$\begin{aligned} P(H|E) &= \frac{P(E|H) \times P(H)}{P(E)} \\ P(\neg H|E) &= \frac{P(E|\neg H) \times P(\neg H)}{P(E)}. \end{aligned} \quad (6.29)$$

两式相除得:

$$\frac{P(H|E)}{P(\neg H|E)} = \frac{P(E|H)}{P(E|\neg H)} \times \frac{P(H)}{P(\neg H)}. \quad (6.30)$$

为讨论方便, 下面引入几率函数的概念

定义 6.14 几率函数

$$O(X) = \frac{P(X)}{1 - P(X)} P(X) = \frac{P(X)}{P(\neg X)}. \quad (6.31)$$

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:

$$P(X) = 0, O(X) = 0; \quad (6.32)$$

$$P(X) = 1, O(X) = +\infty. \quad (6.33)$$

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

把(6.29)式代入(6.32)式有:

$$O(H|E) = \frac{P(E|H)}{P(E|\neg H)} \times O(H). \quad (6.34)$$

再把 LS 代入此式, 可得:

$$Q(H|E) = LS \times QH. \quad (6.35)$$

同理可得到关于 LN 的公式:

$$\frac{P(H|\neg E)}{P(\neg H|\neg E)} = \frac{P(E|H)}{P(\neg E|\neg H)} \times \frac{P(H)}{P(\neg H)} \quad (6.36)$$

$$Q(H|\neg E) = LN \times QH \quad (6.37)$$

式(6.36)和(6.37)就是修改的 Bayes 公式.

6.6 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.

6.7 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问题转化为集合的不确定性问题.

设 Ω 为样本空间, 且 Ω 中的每个元素都相互独立, 则由 的所有子集构成的幂集记为 2^Ω . 当 Ω 中的元素个数为 N 时, 则其幂集 2^Ω 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 6.15 设 $\Omega = \{\text{红}, \text{黄}, \text{白}\}$, 求 Ω 的幂集 2^Ω .

解: Ω 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 6.16 设函数 $m : 2^\Omega \rightarrow [0, 1]$, 且满足

$$\begin{aligned} m(\Phi) &= 0; \\ \sum_{A \subseteq \Omega} m(A) &= 1. \end{aligned} \tag{6.38}$$

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

例 6.17 若定义 2^Ω 上的一个基本函数 m :

$$\begin{aligned} m(\{\emptyset, \text{红}\}, \{\text{黄}\}, \{\text{白}\}, \{\text{红}, \text{黄}\}, \{\text{红}, \text{白}\}, \{\text{黄}, \text{白}\}, \{\text{红}, \text{黄}, \text{白}\}) \\ = (0, 0.3, 0, 0.1, 0.2, 0.2, 0, 0.2). \end{aligned} \tag{6.39}$$

其中, $(0, 0.3, 0, 0.1, 0.2, 0.2, 0, 0.2)$ 分别是幂集 2^Ω 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.

 **注 6.18. 概率分配函数的说明 (I)** 概率分配函数的作用是把 Ω 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 6.19 对上例所给出的有限集 Ω 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, 却不知道该把这 0.2 分给 {红} 还是分给 {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 {黄} 或 {白}.

(2) 概率分配函数不是概率

例 6.20 在例 6.5 中, m 符合概率分配函数的定义, 但

$$m(\{\text{红}\}) + m(\{\text{黄}\}) + m(\{\text{白}\}) = 0.3 + 0 + 0.1 = 0.4 < 1. \quad (6.40)$$

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

定义 6.21 信任函数

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

$$Bel(A) = \sum_{B \subseteq A} m(B), \quad (6.41)$$

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

例 6.22 对例 6.5 有

$$Bel(\{\text{红}\}) = 0.3; \quad (6.42)$$

$$Bel(\{\text{红}, \text{白}\}) = m(\{\text{红}\}) + m(\{\text{白}\}) + m(\{\text{红}, \text{白}\}) = 0.3 + 0.1 + 0.2 = 0.6. \quad (6.43)$$

根据定义还可以得到:

$$\begin{aligned} Bel(\Phi) &= m(\Phi) = 0; \\ Bel(\Omega) &= \sum_{B \subseteq \Omega} m(B) = 1. \end{aligned} \quad (6.44)$$

例 6.23 对例 6.5 有

$$\text{Bel}(\emptyset) = m(\emptyset) = 0. \quad (6.45)$$

$$\begin{aligned} \text{Bel}(\{\text{红}, \text{黄}, \text{白}\}) &= m(\emptyset) + m(\{\text{红}\}) + m(\{\text{黄}\}) + m(\{\text{白}\}) + m(\{\text{红}, \text{黄}\}) \\ &\quad + (\{\text{红}, \text{白}\}) + (\{\text{黄}, \text{白}\}) + (\{\text{红}, \text{黄}, \text{白}\}) \\ &= 0 + 0.3 + 0 + 0.1 + 0.2 + 0.2 + 0 + 0.2 = 1. \end{aligned} \quad (6.46)$$

定义 6.24 似然函数

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

$$P_l(A) = 1 - \text{Bel}(\neg A), A \subseteq \Omega, \quad (6.47)$$

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 $\text{Bel}(\neg A)$ 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 6.25 以例 6.5 为例:

$$\begin{aligned} P_l(\{\text{红}\}) &= 1 - \text{Bel}(\neg\{\text{红}\}) \\ &= 1 - \text{Bel}(\{\text{黄}, \text{白}\}) \\ &= 1 - (m(\{\text{黄}\}) + m(\{\text{白}\}) + m(\{\text{黄}, \text{白}\})) = 1 - (0 + 0.1 + 0) = 0.9. \end{aligned} \quad (6.48)$$

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 6.26 再如:

$$P_l(\{\text{黄}, \text{白}\}) = 1 - \text{Bel}(\neg\{\text{黄}, \text{白}\}) = 1 - \text{Bel}(\{\text{红}\}) = 1 - 0.3 = 0.7, \quad (6.49)$$

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:

$$\sum_{\{\text{红}\} \cap B = \emptyset} m(B) = 0.3 + 0.2 + 0.2 + 0.2 = 0.9. \quad (6.50)$$

由于

$$\sum_{\{\text{黄, 白}\} \cap B = \emptyset} m(B) = 0 + 0.1 + 0 + 0.2 + 0.2 + 0.2 = 0.7. \quad (6.51)$$

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

$$P_l(\{\text{红}\}) = \sum_{\{\text{红}\} \cap B = \emptyset} m(B), \quad (6.52)$$

$$P_l(\{\text{黄, 白}\}) = \sum_{\{\text{黄, 白}\} \cap B = \emptyset} m(B). \quad (6.53)$$

如果把它推广到一般可得公式:

$$P_l(A) = \sum_{A \cap B \neq \emptyset} m(B). \quad (6.54)$$

证明见教材.

信任函数和似然函数之间存在关系:

$$P_l(A) \geq \text{Bel}(A). \quad (6.55)$$

证明.

$$\begin{aligned} \text{Bel}(A) + \text{Bel}(\neg A) &= \sum_{B \in A} m(B) + \sum_{C \in \neg A} m(C) \leq \sum_{E \subseteq \Omega} m(E) = 1. \\ P_l(A) - \text{Bel}(A) &= 1 - \text{Bel}(\neg A) - \text{Bel}(A) \\ &= 1 - (\text{Bel}(\neg A) + \text{Bel}(A)) \geq 0. \end{aligned} \quad (6.56)$$

$$P_l(A) \geq \text{Bel}(A).$$

由于 $\text{Bel}(A)$ 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 $\text{Bel}(A)$ 和 $P_l(A)$ 为对 A 信任程度的下限和上限, 记为: $A[\text{Bel}(A), P_l(A)]$ □

例 6.27 在前面的例子中

$$\text{Bel}(\{\text{红}\}) = 0.3, P_l(\{\text{红}\}) = 0.9, \quad (6.57)$$

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

- $\{\text{黄}\}[0, 0.4] \quad \{\text{白}\}[0.1, 0.5]$
- $\{\text{红}, \text{黄}\}[0.5, 0.9] \quad \{\text{红}, \text{白}\}[0.6, 1]$
- $\{\text{黄}, \text{白}\}[0.1, 0.7] \quad \{\text{红}, \text{黄}, \text{白}\}[1, 1]$
- $\{\}\[0, 0]$

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

- $A[0, 0]$: 说明 A 为假. 即 $\text{Bel}(A) = 0, \text{Bel}(\neg A) = 1$.
- $A[1, 1]$: 说明 A 为真. 即 $\text{Bel}(A) = 1, \text{Bel}(\neg A) = 0$.
- $A[0.6, 1]$: 说明对 A 部分信任. 即 $\text{Bel}(A) = 0.6, \text{Bel}(\neg A) = 0$.
- $A[0, 0.4]$: 说明对 $\neg A$ 部分信任. 即 $\text{Bel}(A) = 0, \text{Bel}(\neg A) = 0.6$.
- $A[0.3, 0.9]$: 说明对 A 和 $\neg A$ 都有部分信任. 其中, $\text{Bel}(A)=0.3$, 说明对 A 为真有 0.3 的信任度; $\text{Bel}(\neg A) = 1 - 0.9 = 0.1$, 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 6.28 对

$$\Omega = \{\text{红}, \text{黄}\}. \quad (6.58)$$

假设从不同知识源得到的两个概率分配函数分别为:

$$m_1(\{\text{红}\}, \{\text{黄}\}, \{\text{红}, \text{黄}\}) = (0, 0.4, 0.5, 0.1) \quad (6.59)$$

$$m_2(\{\text{红}\}, \{\text{黄}\}, \{\text{红}, \text{黄}\}) = (0, 0.6, 0.2, 0.2). \quad (6.60)$$

 **注 6.29.** 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 6.30 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

$$\begin{aligned} m(\Phi) &= 0 \\ m(A) &= K^{-1} \times \sum_{x \cap y = A} m_1(x) \times m_2(y), \end{aligned} \quad (6.61)$$

其中:

$$K = 1 - \sum_{x \cap y = \Phi} m_1(x) \times m_2(y) = \sum_{x \cap y \neq \Phi} m_1(x) \times m_2(y). \quad (6.62)$$

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

$$m_1(\emptyset, \{a\}, \{b\}, \{a, b\}) = (0, 0.3, 0.5, 0.2); \quad (6.63)$$

$$m_2(\emptyset, \{a\}, \{b\}, \{a, b\}) = (0, 0.6, 0.3, 0.1). \quad (6.64)$$

求正交和 $m = m_1 \oplus m_2$.

解: 先求 K

$$\begin{aligned} K &= 1 - \sum_{x \cap y = \Phi} m_1(x) \times m_2(y) \\ &= 1 - (m_1(\{a\}) \times m_2(\{b\}) + m_1(\{b\}) \times m_2(\{a\})) \\ &= 1 - (0.3 \times 0.3 + 0.5 \times 0.6) = 0.61. \end{aligned} \quad (6.65)$$

再求 $m(\{a\}, \{b\}, \{a, b\})$, 由于

$$\begin{aligned} m(\{a\}) &= \frac{1}{0.61} \times \sum_{x \cap y = \{a\}} m_1(x) \times m_2(y) \\ &= \frac{1}{0.61} \times (m_1(\{a\}) \times m_2(\{a\}) + m_1(\{a\}) \times m_2(\{a, b\}) + m_1(\{a, b\}) \times m_2(\{a\})) \\ &= \frac{1}{0.61} \times (0.3 \times 0.6 + 0.3 \times 0.1 + 0.2 \times 0.6) = 0.54. \end{aligned} \quad (6.66)$$

同理可求得

$$m(\{b\}) = 0.43; \quad (6.67)$$

$$m(\{a, b\}) = 0.03. \quad (6.68)$$

故有

$$m(\emptyset, \{a\}, \{b\}, \{a, b\}) = \{0, 0.54, 0.43, 0.03\}. \quad (6.69)$$

对于多个概率分配函数的组合, 方法类似.

6.7.1 证据理论的推理模型

$\text{Bel}(A)$ 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^Ω 上的概率分配函数, 且 m 满足

- (1) $m(\{s_i\}) \geq 0$;
- (2) $\sum_{i=1}^n m(\{s_i\}) \leq 1$; $A \subset \Omega, |A| >> 1$ 或者 $|A| = 0 \Rightarrow m(A) = 0$. (6.70)
- (3) $m(\Omega) = 1 - \sum_{i=1}^n m(\{s_i\})$.

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 Ω 的概率分配数按(3)计算.

例 6.32 设 $\Omega = \{\text{红}, \text{黄}, \text{白}\}$, 有如下概率分配函数

$$m(\emptyset, \{\text{红}\}, \{\text{黄}\}, \{\text{白}\}, \{\text{红}, \text{黄}, \text{白}\}) = (0, 0.6, 0.2, 0.1, 0.1). \quad (6.71)$$

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

例 6.33 对任何命题 $A \subset \Omega$, 其信任函数为

$$\begin{aligned} \text{Bel}(A) &= \sum_{s_i \in A} m(\{s_i\}); \\ \text{Bel}(\Omega) &= \sum_{B \subseteq \Omega} m(B) = \sum_{i=1}^n m(\{s_i\}) + m(\Omega) = 1. \end{aligned} \quad (6.72)$$

定义 6.34 其似然函数

对任何命题 $A \subset \Omega$, 其似然函数为

$$\begin{aligned} P_l(A) &= 1 - \text{Bel}(\neg A) = 1 - \sum_{s_i \in \neg A} m(\{s_i\}) = 1 - \left[\sum_{i=1}^n m(\{s_i\}) - \sum_{s_i \in A} m(\{s_i\}) \right] \\ &= 1 - [1 - m(\Omega) - \text{Bel}(A)] \\ &= m(\Omega) + \text{Bel}(A) \end{aligned} \quad (6.73)$$

$$P_l(\Omega) = 1 - \text{Bel}(\neg \Omega) = 1 - \text{Bel}(\Phi) = 1.$$

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

$$P_l(A) - \text{Bel}(A) = P_l(B) - \text{Bel}(B) = m(\Omega), \quad (6.74)$$

它表示对 A (或 B) 不知道的程度.

例 6.35 设 $\Omega = \{\text{红}, \text{黄}, \text{白}\}$, 概率分配函数

$$m(\{\text{红}\}, \{\text{黄}\}, \{\text{红, 黄}\}, \{\text{红, 黄, 白}\}) = (0, 0.6, 0.2, 0.1, 0.1) \quad (6.75)$$

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

解:

$$m(\Omega) = 1 - [m(\{\text{红}\}) + m(\{\text{黄}\}) + m(\{\text{白}\})] = 1 - (0.6 + 0.2 + 0.1) = 0.1; \quad (6.76)$$

$$\text{Bel}(\{\text{红, 黄}\}) = m(\{\text{红}\}) + m(\{\text{黄}\}) = 0.6 + 0.2 = 0.8; \quad (6.77)$$

$$P_l(\{\text{红, 黄}\}) = m(\Omega) + \text{Bel}(\{\text{红, 黄}\}) = 0.1 + 0.8 = 0.9. \quad (6.78)$$

或

$$P_l(\{\text{红, 黄}\}) = 1 - \text{Bel}(\neg\{\text{红, 黄}\}) = 1 - \text{Bel}(\{\text{白}\}) = 1 - 0.1 = 0.9. \quad (6.79)$$

定义 6.36 基本概率分配函数的正交和

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

$$m(\{s_i\}) = K^{-1} \times [m_1(s_i) \times m_2(s_i) + m_1(s_i) \times m_2(\Omega) + m_1(\Omega) \times m_2(s_i)]. \quad (6.80)$$

例 6.37 设 $\Omega = \{\text{红}, \text{黄}, \text{白}\}$, 概率分配函数

$$m(\{\text{红}\}, \{\text{黄}\}, \{\text{白}\}, \{\text{红, 黄, 白}\}) = (0, 0.6, 0.2, 0.1, 0.1). \quad (6.81)$$

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

解:

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

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

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

或

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

定义 6.38 概率分配函数的正交和

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

$$m(\{s_i\}) = K^{-1} \times [m_1(s_i) \times m_2(s_i) + m_1(s_i) \times m_2(\Omega) + m_1(\Omega) \times m_2(s_i)], \quad (6.82)$$

其中,

$$K = m_1(\Omega) \times m_2(\Omega) + \sum_{i=1}^n [m_1(s_i) \times m_2(s_i) + m_1(s_i) \times m_2(\Omega) + m_1(\Omega) \times m_2(s_i)]. \quad (6.83)$$

定义 6.39 类概率函数

设 Ω 为有限域, 对任何命题 $A \subset \Omega$, 命题 A 的类概率函数为

$$f(A) = \text{Bel}(A) + \frac{|A|}{|\Omega|} \times [P_l(A) - \text{Bel}(A)], \quad (6.84)$$

其中, $|A|$ 和 $|\Omega|$ 分别是 A 及 Ω 中元素的个数.

类概率函数 $f(A)$ 的性质

$$(1) \sum_{i=1}^n f(\{s_i\}) = 1.$$

$$\begin{aligned} \therefore f(\{s_i\}) &= \text{Bel}(\{s_i\}) + \frac{|\{s_i\}|}{|\Omega|} \times [P_l(\{s_i\}) - \text{Bel}(\{s_i\})] \\ &= m(\{s_i\}) + \frac{1}{n} \times m(\Omega) \quad i = 1, 2, \dots, n \end{aligned} \quad (6.85)$$

$$\therefore \sum_{i=1}^n f(\{s_i\}) = \sum_{i=1}^n \left[m(s_i) + \frac{1}{n} \times m(\Omega) \right] = \sum_{i=1}^n m(\{s_i\}) + m(\Omega) = 1. \quad (6.86)$$

$$(2) \text{对任何 } A \subset \Omega, \text{ 有 } \text{Bel}(A) \leq f(A) \leq P_l(A).$$

$$\begin{aligned} \because P_l(A) - P_l(B) &= m(\Omega), \quad \frac{|A|}{|\Omega|} \geq 0 \\ \therefore \text{Bel}(A) &\leq f(A). \\ \because \frac{|A|}{|\Omega|} &\leq 1, \text{ BP } f(A) \leq \text{Bel}(A) + P_l(A) - \text{Bel}(A) \\ \therefore f(A) &\leq P_l(A). \end{aligned} \quad (6.87)$$

$$(3) \text{对任何 } A \subseteq \Omega, \text{ 有 } f(\neg A) = 1 - f(A).$$

$$\therefore f(\neg A) = Bel(\neg A) + \frac{|\mathcal{A}|}{|P_l(\neg A) - Bel(\neg A)|} \quad (6.88)$$

$$\begin{aligned} Bel(\neg A) &= \sum_{\substack{s_i \in \neg A \\ s_i \in A}} m(\{s_i\}) \\ &= 1 - \sum_{\substack{s_i \in A \\ s_i \notin A}} m(\{s_i\}) - m(\Omega) = 1 - Bel(A) - m(\Omega). \end{aligned} \quad (6.89)$$

$$|\neg A| = |\Omega| - |A| \quad (6.90)$$

$$P_l(\neg A) - Bel(\neg A) = m(\Omega) \quad (6.91)$$

$$\begin{aligned} \therefore f(\neg A) &= 1 - Bel(A) - m(\Omega) + \frac{|\Omega| - |A|}{|\Omega|} \times m(\Omega) \\ &= 1 - Bel(A) - m(\Omega) + m(\Omega) - \frac{|A|}{|\Omega|} \times m(\Omega) \\ &= 1 - \left[Bel(A) + \frac{|A|}{|\Omega|} \times m(\Omega) \right] = 1 - f(A). \end{aligned} \quad (6.92)$$

性质 6.1 (

-) $f(\emptyset) = 0$;
- (2) $f(\Omega) = 1$;
- (3) 对任何 $A \subseteq \Omega$, 有 $0 \leq f(A) \leq 1$.



例 6.40 设 $\Omega = \{\text{红}, \text{黄}, \text{白}\}$, 概率分配函数

$$m(\emptyset, \{\text{红}\}, \{\text{黄}\}, \{\text{白}\}, \{\text{红}, \text{黄}, \text{白}\}) = (0, 0.6, 0.2, 0.1, 0.1). \quad (6.93)$$

若 $A = \{\text{红}, \text{黄}\}$, 求 $f(A)$ 的值.

解:

$$\begin{aligned} f(A) &= Bel(A) + \frac{|A|}{|\Omega|} \times [P_l(A) - Bel(A)] \\ &= m(\{\text{红}\}) + m(\{\text{黄}\}) + \frac{2}{3} \times (1 - Bel(\neg A) - 0.8) \\ &= 0.6 + 0.2 + \frac{2}{3} \times (1 - Bel(\neg A) - 0.6) \\ &= 0.6 + 0.2 + \frac{2}{3} \times 0.1 = 0.87. \end{aligned} \quad (6.94)$$

表示形式:

$$\text{IF } E \text{ THEN } H = h_1, h_2, \dots, h_n; \quad CF = \{c_1, c_2, \dots, c_n\}, \quad (6.95)$$

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:

$$\begin{aligned} 1) \quad & c_i \geq 0 \quad i = 1, 2, \dots, n \\ 2) \quad & \sum_{i=1}^n c_i \leq 1. \end{aligned} \quad (6.96)$$

例 6.41 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

$$MD(A/E') = \begin{cases} 1 \\ 0 \end{cases}. \quad (6.97)$$

定义 6.42 条件部分命题 A 的确定性

条件部分命题 A 的确定性为

$$CER(A) = MD(A/E') \times f(A), \quad (6.98)$$

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:

$$E = E_1 \text{ AND } E_2 \text{ AND } \dots \text{ AND } E_n, \quad (6.99)$$

则

$$CER(E) = \min\{CER(E_1), CER(E_2), \dots, CER(E_n)\}. \quad (6.100)$$

当组合证据是多个证据的析取时:

$$E = E_1 \text{ OR } E_2 \text{ OR } \dots \text{ OR } E_n, \quad (6.101)$$

则

$$\text{CER}(E) = \max\{\text{CER}(E_1), \text{CER}(E_2), \dots, \text{CER}(E_n)\}. \quad (6.102)$$

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 $\text{CER}(H)$ 的方法如下:

(1) 求 H 的概率分配函数

$$\begin{aligned} m(\{h_1\}, \{h_2\}, \dots, \{h_n\}) &= (\text{CER}(E) \times c_1, \text{CER}(E) \times c_2, \dots, \text{CER}(E) \times c_n). \\ m(\Omega) &= 1 - \sum_{i=1}^n \text{CRE}(E) \times c_i. \end{aligned} \quad (6.103)$$

如果有两条或多条知识支持同一结论 H , 例:

$$\text{IF } E \text{ THEN } H = h_1, h_2, \dots, h_n \text{ } CF = c_{11}, c_{12}, \dots, c_{1n}; \quad (6.104)$$

$$\text{IF } E \text{ THEN } H = h_1, h_2, \dots, h_n \text{ } CF = c_{21}, c_{22}, \dots, c_{2n}. \quad (6.105)$$

则按正交和求 $\text{CER}(H)$, 即先求出:

$$m_1 = m(h_1, h_2, \dots, h_n); \quad (6.106)$$

$$m_2 = m(h_1, h_2, \dots, h_n). \quad (6.107)$$

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 $\text{Bel}(H)$ 、 $P_l(H)$ 及 $f(H)$

$$\begin{aligned} \text{Bel}(H) &= \sum_{i=1}^n m(\{h_i\}) \\ P_l(H) &= 1 - \text{Bel}(\neg H) \\ f(H) &= \text{Bel}(H) + \frac{|H|}{|\Omega|} \times [P_l(H) - \text{Bel}(H)] = \text{Bel}(H) + \frac{|H|}{|\Omega|} \times m(\Omega). \end{aligned} \quad (6.108)$$

(3) 求 H 的确定性 $\text{CER}(H)$.

按公式 $\text{CER}(H) = MD(H/E') \times f(H)$ 计算结论 H 确定性.

例 6.43 设有如下规则:

- r_1 : IF $E1$ AND $E2$ THEN $A=a1, a2$ $CF=\{0.3, 0.5\}$;
- r_2 : IF $E3$ AND ($E4$ OR $E5$) THEN $B=b1$ $CF=\{0.7\}$;
- r_3 : IF A THEN $H=\{h1, h2, h3\}$ $CF=\{0.1, 0.5, 0.3\}$;
- r_4 : IF B THEN $H=\{h1, h2, h3\}$ $CF=\{0.4, 0.2, 0.1\}$.

已知用户对初始证据给出的确定性为:

$$\text{CER}(E_1)=0.8, \text{CER}(E_2)=0.6, \text{CER}(E_3)=0.9, \text{CER}(E_4)=0.5, \text{CER}(E_5)=0.7.$$

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6-2为:

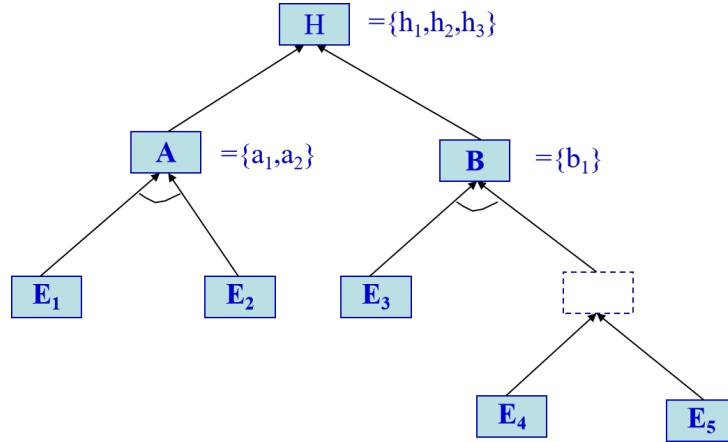


图 6-2 推理网络

(1) 求 $CER(A)$

$$CER(E_1 \text{ AND } E_2) = \min\{CER(E_1), CER(E_2)\} = \min\{0.8, 0.6\} = 0.6.$$

$$m(a_1, a_2) = \{0.6 \times 0.3, 0.6 \times 0.5\} = \{0.18, 0.3\}.$$

$$\text{Bel}(A) = m(a_1) + m(a_2) = 0.18 + 0.3 = 0.48.$$

$$P_l(A) = 1 - \text{Bel}(\neg A) = 1 - 0 = 1.$$

$$f(A) = \text{Bel}(A) + |A|/|\Omega| * [P_l(A) - \text{Bel}(A)] = 0.48 + 2/10 * [1 - 0.48] = 0.584.$$

$$CER(A) = MD(A/E') \times f(A) = 0.584.$$

(2) 求 $CER(B)$

$$CER(E_3 \text{ AND } (E_4 \text{ OR } E_5)) = \min\{CER(E_3), \max\{CER(E_4), CER(E_5)\}\}$$

$$= \min\{0.9, \max\{0.5, 0.7\}\} = \min\{0.9, 0.7\} = 0.7.$$

$$m(b_1) = 0.7 \times 0.7 = 0.49.$$

$$\text{Bel}(B) = m(b_1) = 0.49.$$

$$P_l(B) = 1 - \text{Bel}(\neg B) = 1 - 0 = 1.$$

$$F(B) = \text{Bel}(B) + |B|/|\Omega| * [P_l(B) - \text{Bel}(B)]$$

$$= 0.49 + 1/10 * [1 - 0.49] = 0.541.$$

$$CER(B) = MD(B/E') \times f(B) = 0.541.$$

(3) 求 $CER(H)$, 由 r_3 可得

$$\begin{aligned} m_1(h_1, h_2, h_3) &= \{CER(A) \times 0.1, CER(A) \times 0.5, CER(A) \times 0.3\} \\ &= \{0.584 \times 0.1, 0.584 \times 0.5, 0.584 \times 0.3\} = \{0.058, 0.292, 0.175\} \end{aligned}$$

$$m_1(\Omega) = 1 - [m_1(h_1) + m_1(h_2) + m_1(h_3)] = 1 - (0.058 + 0.292 + 0.175) = 0.475.$$

再由 r_4 可得

$$\begin{aligned} m_2(h_1, h_2, h_3) &= CER(B) \times 0.4, CER(B) \times 0.2, CER(B) \times 0.1 \\ &= \{0.541 \times 0.4, 0.541 \times 0.2, 0.541 \times 0.1\} = \{0.216, 0.108, 0.054\} \\ m_2(\Omega) &= 1 - [m_2(h_1) + m_2(h_2) + m_2(h_3)] \\ &= 1 - [0.216 + 0.108 + 0.054] = 0.622. \end{aligned}$$

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:

$$\begin{aligned} m(h_2) &= \frac{1}{K} \times [m_1(\{h_2\}) \times m_2(\{h_2\}) + m_1(\{h_2\}) \times m_2(\Omega) + m_1(\Omega) \times m_2(\{h_2\})] \\ &= \frac{1}{0.855} \times [0.292 \times 0.108 + 0.292 \times 0.622 + 0.475 \times 0.108] \\ &= 0.309. \end{aligned}$$

$$\begin{aligned} m(h_3) &= \frac{1}{K} \times [m_1(\{h_3\}) \times m_2(\{h_3\}) + m_1(\{h_3\}) \times m_2(\Omega) + m_1(\Omega) \times m_2(\{h_3\})] \\ &= \frac{1}{0.855} \times [0.175 \times 0.054 + 0.175 \times 0.622 + 0.475 \times 0.054] \\ &= 0.168. \end{aligned}$$

$$m(\Omega) = 1 - [m(h_1) + m(h_2) + m(h_3)] = 1 - (0.178 + 0.309 + 0.168) = 0.345.$$

再根据 m 可得

$$\text{Bel}(H) = m(h_1) + m(h_2) + m(h_3) = 0.178 + 0.309 + 0.168 = 0.655.$$

$$P_l(H) = m(\Omega) + \text{Bel}(H) = 0.345 + 0.655 = 1.$$

$$\begin{aligned} f(H) &= \text{Bel}(H) + \frac{|H|}{|\Omega|} \times [P_l(H) - \text{Bel}(H)] = 0.655 + \frac{3}{10} \times [1 - 0.655] \\ &= 0.759. \end{aligned}$$

$$\text{CER}(H) = MD(H/E') \times f(H) = 0.759.$$

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.

6.8 模糊推理

用自然语言中的词或句子表示的变量

例 6.44 变量“年龄”在普通集合中为数字变量 $u = [0, 150]$, 而在模糊集中可使用语言变量, 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.

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

$$x \text{ is } F, \tag{6.109}$$

其中的模糊谓词 F 可以是大、小、年轻、年老、冷、暖、长、短等.

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 λ 为模糊概率, μ 为模糊可能性, τ 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:

$$(x \text{ is } F) \text{ is } \lambda(x \text{ is } F) \text{ is } \prod (x \text{ is } F) \text{ is } \tau, \quad (6.110)$$

其中, λ 可以是“或许”、“必须”等; μ 可以是“非常可能”、“很不可能”等; τ 可以是“非常真”、“有些假”等.

例 6.45 “常青很可能是年轻的”可表示为

$$(\text{Age}(\text{Chang qing}) \text{ is young}) \text{ is likely.}$$

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

$$\mu_{\neg F}(u) = 1 - \mu_F(u), u \in [0, 1].$$

② 集中表示“很”、“非常”等, 其效果是减少隶属函数的值:

$$\mu_{\text{非常}F}(u) = \mu_F^2(u), u \in [0, 1].$$

③ 扩张表示“有些”、“稍微”等, 其效果是增加隶属函数的值:

$$\mu_{\text{有些}F}(u) = \mu_F^{\frac{1}{2}}(u), u \in [0, 1].$$

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 6.46 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

$$\begin{aligned}\mu_{\text{非常真}}(u) &= u^2, u \in [0, 1]; & \mu_{\text{非常假}}(u) &= (1-u)^2, u \in [0, 1]; \\ \mu_{\text{有些真}}(u) &= u^{\frac{1}{2}}, u \in [0, 1]; & \mu_{\text{有些假}}(u) &= (1-u)^{\frac{1}{2}}, u \in [0, 1].\end{aligned}$$

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。

6.8.1 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

$$d(F, G) = \frac{1}{n} \times \sum_{i=1}^n |\mu_F(u_i) - \mu_G(u_i)|. \quad (6.111)$$

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

$$d(F, G) = \frac{1}{b-a} \int_a^b |\mu_F(u) - \mu_G(u)| d(u). \quad (6.112)$$

例 6.47 设论域 $U = \{-10, 0, 10, 20, 30\}$ 表示温度，模糊集

$$F = 0.8/-10 + 0.5/0 + 0.1/10; \quad (6.113)$$

$$G = 0.9/-10 + 0.6/0 + 0.2/10. \quad (6.114)$$

分别表示“冷”和“比较冷”，则

$$d(F, G) = 0.2 \times (|0.8 - 0.9| + |0.5 - 0.6| + |0.1 - 0.2|) = 0.2 \times 0.3 = 0.06, \quad (6.115)$$

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，则它们的贴近度定义为

$$(F, G) = \frac{1}{2}(F \cdot G + (1 - F \odot G)), \quad (6.116)$$

其中:

$$\begin{aligned} F \cdot G &= \vee (\mu_F(u_i) \wedge \mu_G(u_i)); \\ F \odot G &= \wedge_U (\mu_F(u_i) \vee \mu_G(u_i)). \end{aligned} \quad (6.117)$$

称 $F \cdot G$ 为内积, $F \odot G$ 为外积.

例 6.48 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 的贴近度为 0.9.

6.8.2 模糊推理

模糊推理实际上是指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. 主要讨论模糊关系的构造和合成运算.

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

$$R_m = \int_{U \times V} (\mu_F(u) \wedge \mu_G(v)) \vee (1 - \mu_F(u)) / (u, v), \quad (6.118)$$

其中, \times 号表示模糊集的笛卡尔乘积.

例 6.49 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 是由玛达尼 (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 6.50 对例6.49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 $R_c(3,2) = \mu_F(u_3) \wedge \mu_G(v_2) = 0.1 \wedge 0.6 = 0.1$.

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

$$\mu_F(u) \rightarrow \mu_G(v) = \begin{cases} 1, & \mu_F(u) \leq \mu_G(v)^\top; \\ \mu_F(v), & \mu_F(u) > \mu_G(v)^\top. \end{cases}$$

例 6.51 对例6.49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

6.8.2.1 模糊假言推理举例

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.

模糊推理的基本方法

例 6.52 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

IF x is F THEN y 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 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 is F THEN y is G

证据 : THEN z is H

结论 : IF x is F' .

在这种推理模式下, 模糊知识

IF x is F THEN y 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 6.53 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 6.54 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

$$R = R_1 \circ R_2 = \begin{bmatrix} 0.2 & 0.6 & 0.8 \\ 0.2 & 0.6 & 1 \\ 0.2 & 1 & 1 \end{bmatrix}.$$

6.9 高阶模糊推理

6.10 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

$$\text{少} = 0.9/1 + 0.7/2 + 0.4/3 \quad (6.119)$$

$$\text{多} = 0.3/3 + 0.7/4 + 0.9/5 \quad (6.120)$$

已知事实为

$$x \text{ is 较少} \quad (6.121)$$

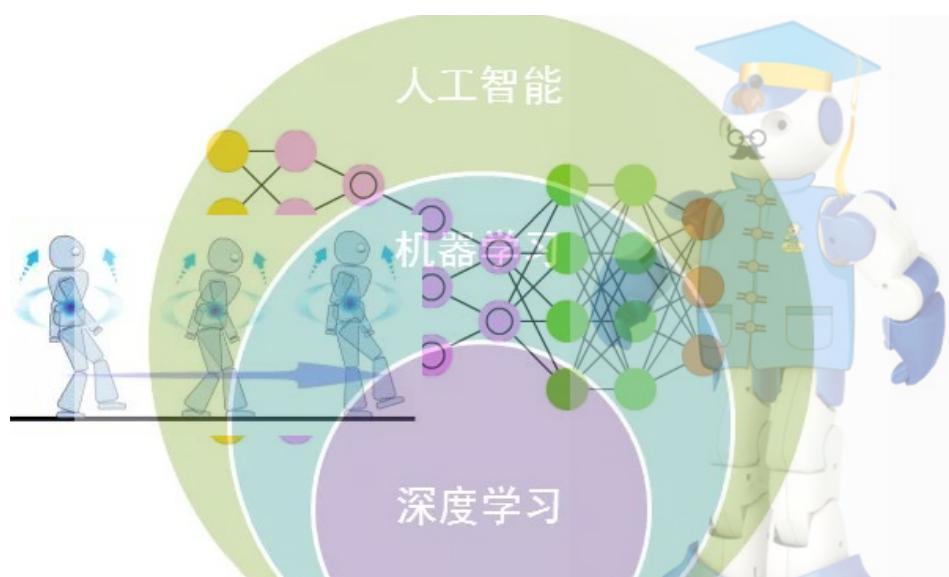
“较少”的模糊集为

$$\text{较少} = 0.8/1 + 0.5/2 + 0.2/3 \quad (6.122)$$

请用模糊关系 R_m 求出模糊结论.

7

机器学习



对机器学习的认识和定义可以从多个方面进行：

机器学习

学习是人类获取知识的重要途径和自然智能的重要标志，机器学习则是机器获取知识的重要途径和人工智能的重要标志。它是人工智能的一个分支，探索如何让计算机通过经验学习提高性能—Stuart Russell (伯克利人工智能学家, 友善及安全 AI 的先驱之一)。

机器学习

对于某类任务 T 和性能度量 P , 如果计算机程序在 T 上以 P 衡量的性能随着经验 E 而自我完善, 就称这个计算机程序从经验 E 学习——(Tom Mitchell, “全球机器学习教父”).

普遍认为, 机器学习 (Machine Learning, 常简称为 ML) 的处理系统和算法是主要通过找出数据里隐藏的模式进而做出预测的识别模式, 它是人工智能 (Artificial Intelligence, 常简称为 AI) 的一个重要子领域.

7.1 机器学习

7.1.1 机器学习概述

7.1.1.1 机器学习概述

机器学习的常见误解 「机器学习是一个新的领域, 它已经代替了人工智能的地位」. 这种误解是最近机器学习热潮产生的副作用, 大量学生在之前没有接触过人工智能的情况下学习了机器学习课程. 机器学习一直是人工智能的核心话题: 阿兰·图灵在二十世纪五十年代的论文中已经认为学习是通向人工智能最可行的途径. 人工智能最突出的早期成果, Arthur Samuel 使用机器学习构建跳棋程序就是.

「机器不能学习, 它们只能做程序员告诉它的事情」. 这显然是错的, 程序员能够告诉机器如何学习. Samuel 是一个优秀的跳棋玩家, 但他的程序很快就通过学习超过了他. 近年来, 机器学习的很多应用都需要大量数据来进行训练.

代表性观点 (1) 西蒙 (Simon, 1983): 学习就是系统中的适应性变化, 这种变化使系统在重复同样工作或类似工作时, 能够做得更好.

- (2) 明斯基 (Minsky, 1985): 学习是在人们头脑里 (心理内部) 有用的变化.
- (3) 迈克尔斯 (Michalski, 1986): 学习是对经历描述的建立和修改.

机器学习的主要方法 学习能力是人工智能的关键.

监督学习 (Supervised learning)

使用有标签数据进行学习. 典型应用场景: 分类和回归

非监督学习 (Unsupervised learning)

使用无标签数据进行学习

典型应用场景：聚类

半监督学习 (Semi-supervised learning)

数据一部分有标签, 无标签数据的数量 \gg 有标签数据数量

典型应用场景：海量数据分类

强化学习 (Reinforcement learning)

使用无标签但有反馈的数据进行学习

典型场景：策略推理

机器学习的基本定律：模型的出错率正比于模型复杂程度与样本大小的比.

结论：模型复杂，需要大样本. 样本小，简化模型.

人工智能 (1950's->1980's) 到机器学习 (1980's->2010's) 再到深度学习的脉络
(2010's->至今)

以“推理”为重点 -> 以“知识”为重点.

以“特征”为重点 -> 以“学习”为重点.

传统机器学习：需要人工提取特征代替原始输入, 有效的特征对模型性能至关重要.
传统机器学习算法的难点在于“特征工程. 特征提取步骤如图7-1.

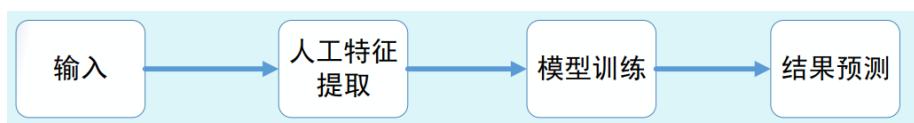


图 7-1 特征提取步骤

深度学习：模仿人脑处理信息的过程——分层处理机制. 底层捕捉输入的“简单”特征, 高层通过组合底层特征从而形成更加复杂抽象化的特征. 深度学习由简到繁, 自动实现特征提取. 深度学习结构如图 7-2.

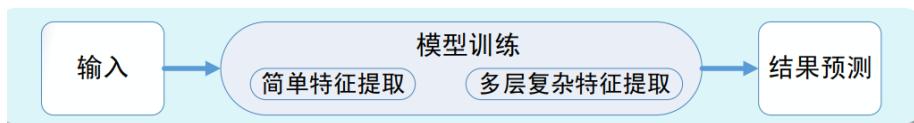


图 7-2 深度学习结构

工业 4.0 时代的机器学习

机器学习属于数据科学范畴，被视为一种涵盖数据处理所有方面的研究，它是人工智能的一个分支，赋予机器自主学习的能力，而不需要人工干预。机器学习使计算机不需要人工编程，就可以自动化地执行任务。

机器学习的流程如下：

1. 数据收集：这是机器学习流程的第一步，也是最重要的一步。计算机根据问题陈述收集相关数据，这些被收集的数据被称为训练数据，它们应当准确完整，以便解决问题；
2. 数据预处理：预处理是为了将采集到的不完整、不一致和错误的数据转换为可行的数据，以便更好地拟合机器学习模型。在除去了数据集的问题后，数据的特征将被提取出来，并用于模型训练；
3. 模型构建：选择适当的机器学习技术来获得预期结果的过程。机器学习的模型种类很多，总体可分为有监督学习、无监督学习和强化学习三种，不同的技术适合处理的问题不同；
4. 模型训练与测试：在确定选择哪种模型之后，预处理的数据集会被分为训练集和测试集两个部分，分别满足训练和测试两种要求。模型训练指利用机器学习技术将模型评估的误差降到最小，当模型训练完成后，它就被用于测试数据集中进行测试工作，以评估模型的效率和准确性，测试的结果会由一些评价指标进行反映；
5. 性能评估：利用交叉验证、参数调整和多种机器学习算法，尝试得到效果更好的算法，或者使用组合方法将多个算法的结果组合起来；
6. 模型执行：执行模型输出结果，以便在未来利用模型完成机器学习任务；

由于不同的 VUCA 战略所要实现的目标不同，这就涉及到了一些具体的机器学习算法。

一般性解释 学习是一个有特定目的知识获取和能力增长过程，其内在行为是获得知识、积累经验、发现规律等，其外部表现是改进性能、适应环境、实现自我完善等。

一般性解释 机器学习就是让机器（计算机）来模拟和实现人类的学习功能。

主要研究内容

► 认知模拟

主要目的是要通过对人类学习机理的研究和模拟，从根本上解决机器学习方面存在的种种问题。

► 理论性分析

主要目的是要从理论上探索各种可能的学习方法，并建立起独立于具体应用领域的学习算法。

► 面向任务的研究

主要目的是要根据特定任务的要求,建立相应地学习系统.

神经元模型研究 20世纪50年代中期到60年代初期,也被称为机器学习的热烈时期,最具有代表性地工作是罗森勃拉特1957年提出的感知器模型.

► 符号概念获取

20世纪60年代中期到70年代初期.其主要研究目标是模拟人类的概念学习过程.这一阶段神经学习落入低谷,称为机器学习的冷静时期.

► 知识强化学习

20世纪70年代中期到80年代初期.人们开始把机器学习与各种实际应用相结合,尤其是专家系统在知识获取方面的需求,也有人称这一阶段为机器学习的复兴时期.

► 连接学习和混合型学习

20世纪80年代中期至今.把符号学习和连接学习结合起来的混合型学习系统研究已成为机器学习研究的一个新的热点.

学习系统 如果一个系统在与环境相互作用时,能利用过去与环境作用时得到的信息,并提高其性能,那么这样的系统就是学习系统.它包括环境、学习环节、知识库和执行环节(图7-3).

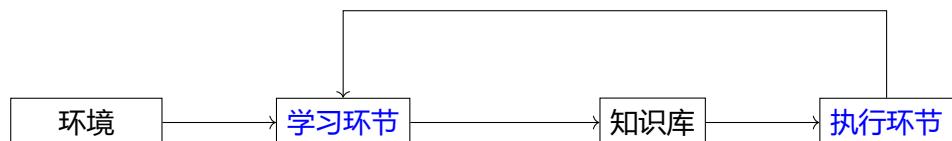


图 7-3 学习系统

环境是学习系统所感知到的外界信息集合,也是学习系统的外界来源.信息的水平(一般化程度)和质量(正确性)对学习系统影响较大.

► 学习环节

对环境提供的信息进行整理、分析归纳或类比,形成知识,并将其放入知识库.

► 知识库

存储经过加工后的信息(即知识).其表示形式是否合适非常重要.

► 执行环节

根据知识库去执行一系列任务,并将执行结果或执行过程中获得的信息反馈给学习环节.学习环节再利用反馈信息对知识进行评价,进一步改善执行环节的行为.

7.2 机器学习的主要策略

► 按学习策略来分类

即按学习中所使用的推理方法来分, 可分为记忆学习、传授学习、演绎学习、归纳学习等.

► 按应用领域分类

- 专家系统学习、机器人学习、自然语言理解学习等.
- 按对人类学习的模拟方式
- 符号主义学习、连接主义学习等.

自动机器学习 (AutoML) 是将机器学习应用于现实问题的端到端过程自动化的过程. AutoML 使机器学习真正意义上成为可能, 即使对于在该领域没有专业知识的人也是如此. 典型的机器学习模型包括以下四个过程 (图7-4):

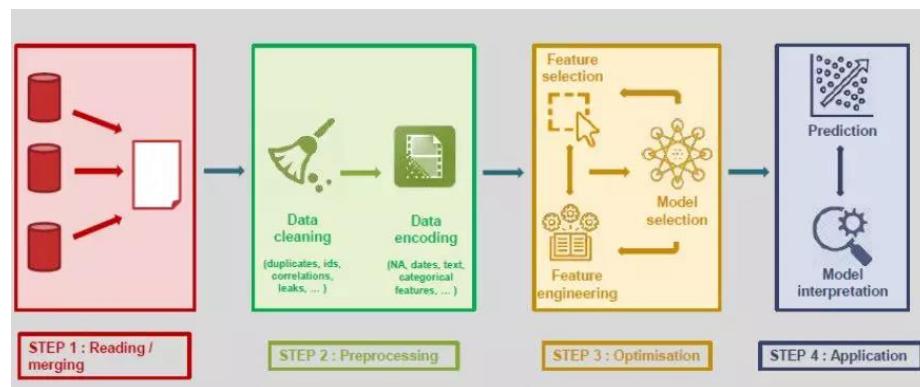


图 7-4 典型机器学习模型的四个过程

如何自动化机器学习管道 (Axel de Romblay): 从摄取数据到预处理、优化, 然后预测结果, 每个步骤都由人来控制和执行. AutoML 主要关注两个主要方面: 数据采集 / 收集和预测. 中间发生的所有其他步骤都可以轻松实现自动化, 同时提供经过优化并准备好进行预测的模型.

7.3 netZooPy 库

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple PandaPy
```

能够使用混合 int, float, datetime, str 等数据类型进行机器学习项目, 消耗的内存比 Pandas 少大约三分之一!

1) 对于小型数据集（即加号，多号，对数）的简单计算，PandaPy 比 Pandas 快 25 倍-80 倍。

2) 对于小型数据集上的表函数（即组，枢纽，放置，连接，填充，填充），PyPanda 比 Pandas 快 5 倍-100 倍。

3) 对于大多数使用小数据的用例，PandaPy 比 Dask，Modin Ray 和 Pandas 快。

3、Google Earth Engine -300 多个 Jupyter 笔记本可分析地理空间数据。

[netZooM-github](#), [netZooPy-github](#), [Fast Neptune 库](#)使我们能够快速记录启动机器学习实验所需的所有信息。

Using PANDA to infer gene regulatory networks

panda.mlx

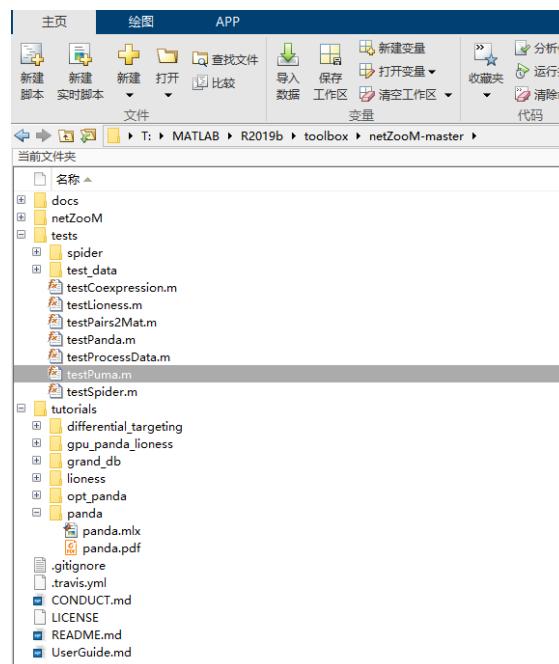


图 7-5 netZooM (panda 工具箱)

7.4 记忆学习

记忆学习 (Rote learning) 也叫死记硬背学习，是一种最基本的学习过程，它没有足够的能力独立完成智能学习，但对学习系统来说都是十分重要的一个组成部分，原因是任何学习系统都必须记住它们所获取的知识，以便将来使用。

记忆学习的基本过程是：执行元素每解决一个问题，系统就记住这个问题和它的解，当以后再遇到此类问题时，系统就不必重新进行计算，而可以直接找出原来的解去使用。

若把执行元素比作一个函数 f , 由环境得到的输入模式记为 (x_1, x_2, \dots, x_n) , 由该输入模式经 F 计算后得到的输出模式记为 (y_1, y_2, \dots, y_m) , 则机械学习系统就是要把这一输入输出模式对:

$$[(x_1, x_2, \dots, x_n), (y_1, y_2, \dots, y_m)] \quad (7.1)$$

保存在知识库中, 当以后再需要计算 $f(x_1, x_2, \dots, x_n)$ 时, 就可以直接从存储器把 (y_1, y_2, \dots, y_m) 检索出来, 而不需要再重新进行计算.

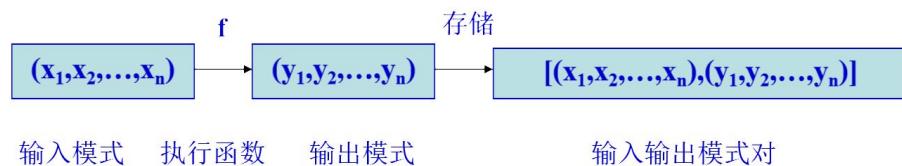


图 7-6 机械式学习的学习模型

7.5 机械式学习的学习模型

7.5.1 归纳学习

- ▶ 归纳学习是指以归纳推理为基础的学习, 其任务是要从关于某个概念的一系列已知的正例和反例中归纳出一个一般的概念描述.
- ▶ 示例学习是归纳学习的一种特例. 它给学习者提供某一概念的一组正例和反例, 学习者归纳出一个总的概念描述, 并使这个描述适合于所有的正例, 排除所有的反例.
- ▶ 决策树学习是一种以示例为基础的归纳学习方法, 也是目前最流行的归纳学习方法之一. 在现有的各种决策树学习算法中, 影响较大的是 ID3 算法. 本节主要讨论决策树的概念和决策树学习的 ID3 算法.

7.5.2 示例学习的类型

- ▶ 按例子的来源分类
 - ① 来源于教师的示例学习.
 - ② 来源于学习者本身的示例学习.
- ▶ 学习者明确知道自己的状态, 但完全不清楚所要获取的概念.
 - ♠ 来源于学习者以外的外部环境的示例学习. 例子的产生是随机的.

► 按例子的类型分类

① 仅利用正例的示例学习.

这种学习方法会使推出的概念的外延扩大化.

② 利用正例和反例的示例学习.

这是示例学习的一种典型方式, 它用正例用来产生概念, 用反例用来防止概念外延的扩大.

示例学习的典型方式

- 示例空间: 向系统提供的示教例子的集合. 研究问题: 例子质量, 搜索方法.
- 解释过程: 是从搜索到的示例中抽象出一般性的知识的归纳过程. 解释方法: 常量转换为变量, 去掉条件, 增加选择, 曲线拟合等.
- 规则空间: 是事务所具有的各种规律的集合. 研究问题: 对空间的要求, 搜索方法.
- 验证过程: 是要从示例空间中选择新的示例, 对刚刚归纳出的规则做进一步的验证和修改.

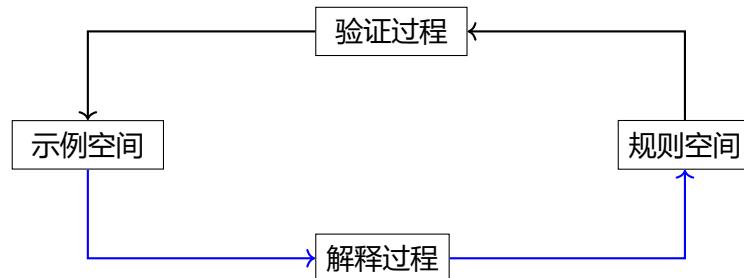


图 7-7 示例学习

7.6 示例学习的解释方法

是指解释过程从具体示例形成一般性知识所采用的归纳推理方法. 最常用的解释方法有以下 4 种:

- (1) 把常量转换为变量: 把示例中的常量换成变量而得到一个一般性的规则.
- (2) 去掉条件: 把示例中的某些无关的子条件舍去.
- (3) 增加选择: 在析取条件中增加一个新的析取项. 常用的增加析取项的方法有前件析取法和内部析取法两种.
- (4) 曲线拟合: 对数值问题的归纳可采用最小二乘法进行曲线拟合.

例 7.1 假设例子空间中有以下两个关于扑克牌中“同花”概念的示例: 花色 (c_1 , 梅花) \wedge 花色 (c_2 , 梅花) \wedge 花色 (c_3 , 梅花) \wedge 花色 (c_4 , 梅花) \wedge 花色 (c_5 , 梅花) \rightarrow 同花 (c_1, c_2, c_3, c_4, c_5).

花色 (c_1 , 红桃) \wedge 花色 (c_2 , 红桃) \wedge 花色 (c_3 , 红桃) \wedge 花色 (c_4 , 红桃) \wedge 花色 (c_5 , 红桃) \rightarrow 同花 (c_1, c_2, c_3, c_4, c_5).

其中, 示例 1 表示 5 张梅花牌是同花, 示例 2 表示 5 张红桃牌是同花.

解释过程:

► (1) 把常量化为变量

例 7.2 对这两个示例, 只要把“梅花”和“红桃”用变量 x 代换, 就可得到如下一般性的规则:

规则 1: 花色 (c_1, x) \wedge 花色 (c_2, x) \wedge 花色 (c_3, x) \wedge 花色 (c_4, x) \wedge 花色 (c_5, x) \rightarrow 同花 (c_1, c_2, c_3, c_4, c_5).

► (2) 去掉条件: 把示例中的某些无关的子条件舍去.

例 7.3 花色 (c_1 , 红桃) \wedge 点数 ($c_1, 2$) \wedge 花色 (c_2 , 红桃) \wedge 点数 ($c_2, 3$) \wedge 花色 (c_3 , 红桃) \wedge 点数 ($c_3, 4$) \wedge 花色 (c_4 , 红桃) \wedge 点数 ($c_4, 5$) \wedge 花色 (c_5 , 红桃) \wedge 点数 ($c_5, 6$) \rightarrow 同花 (c_1, c_2, c_3, c_4, c_5)

为了学习同花的概念, 除了需要把常量变为变量外, 还需要把与花色无关的“点数”子条件舍去. 这样也可得到上述规则 1:

- 规则 1: 花色 (c_1, x) \wedge 花色 (c_2, x) \wedge 花色 (c_3, x) \wedge 花色 (c_4, x) \wedge 花色 (c_5, x) \rightarrow 同花 (c_1, c_2, c_3, c_4, c_5).
- (3) 增加选择: 在析取条件中增加一个新的析取项. 包括前件析取法和内部析取法.

前件析取法: 是通过对示例的前件的析取来形成知识的.

例 7.4 示例的前件有 4 个:

- 点数 (c_1, J) \rightarrow 脸 (c_1)
- 点数 (c_1, Q) \rightarrow 脸 (c_1)
- 点数 (c_1, K) \rightarrow 脸 (c_1)

将各示例的前件进行析取, 就可得到所要求的规则:

- 规则 2: 点数 (c_1, J) 点数 (c_1, Q) 点数 (c_1, K) \rightarrow 脸 (c_1)

内部析取法: 是在示例的表示中使用集合与集合的成员关系来形成知识的.

例 7.5 有如下关于“脸牌”的示例:

- 点数 $c_1 \in J \rightarrow$ 脸 (c_1)
- 点数 $c_1 \in Q \rightarrow$ 脸 (c_1)
- 点数 $c_1 \in K \rightarrow$ 脸 (c_1)

用内部析取法, 可得到如下规则:

- 规则 3: 点数 (c_1) $\in J, Q, K \rightarrow$ 脸 (c_1)

► (4) 曲线拟合: 对数值问题的归纳可采用曲线拟合法.

假设示例空间中的每个示例 (x, y, z) 都是输入 x, y 与输出 z 之间关系的三元组.

例 7.6 有下 3 个示例:

- (0, 2, 7)
- (6, -1, 10)
- (-1, -5, -16)

用最小二乘法进行曲线拟合, 可得 x, y, z 之间关系的规则如下:

- 规则 4: $z = 2x + 3y + 1$.

 **注 7.7.** 在上述前三种方法中, 方法 (1) 是把常量转换为变量; 方法 (2) 是去掉合取项(约束条件); 方法 (3) 是增加析取项. 它们都是要扩大条件的适用范围. 从归纳速度上看, 方法 (1) 的归纳速度快, 但容易出错; 方法 (2) 归纳速度慢, 但不容易出错. 因此, 在使用方法 (1) 时应特别小心.

例 7.8 对示例 4、示例 5 及示例 6, 若使用方法 (1), 则会归纳出如下的错误规则:

规则 5: (错误) 点数 (c_1, x) \rightarrow 脸 (c_1),

它说明, 归纳过程是很容易出错的.

决策树是一种由节点和边构成的用来描述分类过程的层次数据结构. 该树的根接点表示分类的开始, 叶节点表示一个实例的结束, 中间节点表示相应实例中的某一属性, 而边则代表某一属性可能的属性值. 在决策树中, 从根节点到叶节点的每一条路径都代表一个具体的实例, 并且同一路径上的所有属性之间为合取关系, 不同路径(即一个属性的不同属性值)之间为析取关系. 决策树的分类过程就是从这棵树的根接点开始, 按照给定的事例的属性值去测试对应的树枝, 并依次下移, 直至到达某个叶节点为止.

图7-8是一个非常简单的用来描述对鸟类进行分类的决策树. 在该图中: 根节点包含各种鸟类, 叶节点是所能识别的各种鸟的名称; 中间节点是鸟的一些属性, 边是鸟的某一属性的属性值; 从根节点到叶节点的每一条路径都描述了一种鸟, 它包括该种鸟的一些属性及相应的属性值.

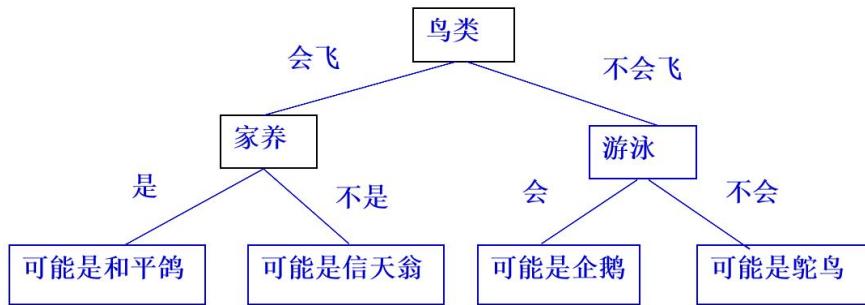


图 7-8 对鸟类进行分类的决策树

决策树还可以表示成规则的形式. 上图7-8的决策树可表示为如下规则集:

IF 鸟类会飞	AND 是家养的	THEN 该鸟类是和平鸽
IF 鸟类会飞	AND 不是家养的	THEN 该鸟类是信天翁
IF 鸟类不会飞	AND 会游泳	THEN 该鸟类是企鹅
IF 鸟类不会飞	AND 不会游泳	THEN 该鸟类是鸵鸟

决策树学习过程实际上是一个构造决策树的过程. 当学习完成后, 就可以利用这棵决策树对未知事物进行分类.

7.6.1 ID3 算法

D3 算法是昆兰 (J.R.Quinlan) 于 1979 年提出的一种以信息熵 (entropy) 的下降速度作为属性选择标准的一种学习算法. 其输入是一个用来描述各种已知类别的例子集, 学习结果是一棵用于进行分类的决策树.

ID3 算法的数学基础 先引入信息熵和条件熵的数学概念

信息熵 信息熵是对信息源整体不确定性的度量. 假设 X 为信源, x_i 为 X 所发出的单个信息, $P(x_i)$ 为 X 发出 x_i 的概率, 则信息熵可定义为:

$$\begin{aligned}
 H(X) &= -P(x_1) \log P(x_1) - P(x_2) \log P(x_2) - \cdots - P(x_r) \log P(x_r) \\
 &= -\sum_{i=1}^k P(x_i) \log P(x_i)
 \end{aligned} \tag{7.2}$$

其中, k 为信源 X 发出的所有可能的信息类型, 对数可以是以各种数为底的对数, 在 ID3 算法中, 我们取以 2 为底的对数.

注 7.9. 信息熵反应的是信源每发出一个信息所提供的平均信息量.

条件熵 条件熵是收信者在收到信息后对信息源不确定性的度量. 若假设信源为 X , 收信者收到的信息为 Y , $P(x_i|y_j)$ 为当 Y 为 y_j 时 X 为 x_i 的条件概率, 则条件熵可定义为:

$$H(X|Y) = - \sum_i^k \sum_j^k P(x_i|y_j) \log P(x_i|y_j) \quad (7.3)$$

它表示收信者收到 Y 后对 X 不确定性的估计.

ID3 算法及举例 ID3 算法的学习过程:

- ▶ 首先以整个例子集作为决策树的根节点 S , 并计算 S 关于每个属性的期望熵(即条件熵);
- ▶ 然后选择能使 S 的期望熵为最小的一个属性对根节点进行分裂, 得到根节点的一层子节点;
- ▶ 接着再用同样的方法对这些子节点进行分裂, 直至所有叶节点的熵值都下降为 0 为止.

这时, 就可得到一棵与训练例子集对应的熵为 0 的决策树, 即 ID3 算法学习过程所得到的最终决策树. 该树中每一条从根节点到叶节点的路径, 都代表了一个分类过程, 即决策过程.

例 7.10 用 ID3 算法完成下述学生选课

假设将决策 y 分为以下 3 类:

\$y_1\$: 必修 AI
 \$y_2\$: 选修 AI
 \$y_3\$: 不修 AI

做出这些决策的依据有以下 3 个属性:

\$x_1\$:	学历层次	\$x_1=1\$ 研究生,	\$x_1=2\$ 本科
\$x_2\$:	专业类别	\$x_2=1\$ 电信类,	\$x_2=2\$ 机电类
\$x_3\$:	学习基础	\$x_3=1\$ 修过 AI,	\$x_3=2\$ 未修 AI

表7-1给出了一个关于选课决策的训练例子集 S . 在该表中, 训练例子集 S 的大小为. ID3 算法是依据这些训练例子, 以 S 为根节点, 按照信息熵下降最大的原则来构造决策树的.

表 7-1 关于选课决策的训练例子集

序号	属性值			决策方案
y_i	x_1	x_2	x_3	
1	1	1	1	y_3
2	1	1	2	y_1
3	1	2	1	y_3
4	1	2	2	y_2
5	2	1	1	y_3
6	2	1	2	y_2
7	2	2	1	y_3
8	2	2	2	y_3

解: 首先对根节点, 其信息熵为:

$$H(S) = - \sum_{i=1}^3 P(y_i) \log_2 P(y_i) \quad (7.4)$$

其中, 为可选的决策方案数, 且有

$$P(y_1) = \frac{1}{8}, P(y_2) = \frac{2}{8}, P(y_3) = \frac{5}{8} \quad (7.5)$$

即有:

$$H(S) = - \left(\frac{1}{8} \right) \log_2 \left(\frac{1}{8} \right) - \left(\frac{2}{8} \right) \log_2 \left(\frac{2}{8} \right) - \left(\frac{5}{8} \right) \log_2 \left(\frac{5}{8} \right) = 1.2988. \quad (7.6)$$

按照 ID3 算法, 需要选择一个能使 S 的期望熵为最小的一个属性对根节点进行扩展, 因此我们需要先计算 S 关于每个属性的条件熵:

$$H(S/x_i) = \sum_t \frac{|S_t|}{|S|} H(S_i) \quad (7.7)$$

其中, t 为属性 x_i 的属性值, S_t 为 $x_i = t$ 时的例子集, $|S|$ 和 $|S_i|$ 分别是例子集 S 和 S_i 的大小.

下面先计算 S 关于属性 x_1 的条件熵:

$$H(S/x_1) = \sum_t \frac{|S_t|}{|S|} H(S_i) \quad (7.8)$$

在表 7-1 中, x_1 的属性值可以为 1 或 2. 当 $x_1 = 1$ 时, $t = 1$ 时, 有:

$$S_1 = \{1, 2, 3, 4\} \quad (7.9)$$

当 $x_1 = 2$ 时, $t = 2$ 时, 有:

$$S_2 = \{5, 6, 7, 8\} \quad (7.10)$$

其中, S_1 和 S_2 中的数字均为例子集 S 中的各个例子的序号, 且有 $|S| = 8, |S_1| = |S_2| = 4$.

由 S_1 可知:

$$P_{s_1}(y_1) = 1/4, P_{s_1}(y_2) = 1/4, P_{s_1}(y_3) = 2/4 \quad (7.11)$$

则有:

$$H(S_1) = -P_{s_1}(y_1) \log_2 P_{s_1}(y_1) - P_{s_1}(y_2) \log_2 P_{s_1}(y_2) - P_{s_1}(y_3) \log_2 P_{s_1}(y_3) \quad (7.12)$$

$$= -(1/4) \log_2(1/4) - (1/4) \log_2(1/4) - (2/4) \log_2(2/4) = 1.5 \quad (7.13)$$

再由 S_2 可知:

$$P_{s_2}(y_1) = 0, P_{s_2}(y_2) = \frac{1}{4}, P_{s_2}(y_3) = \frac{3}{4} \quad (7.14)$$

则有:

$$H(S_2) = P_{s_2}(y_2) \log_2 P_{s_2}(y_2) - P_{s_2}(y_3) \log_2 P_{s_2}(y_3) \quad (7.15)$$

$$P_{s_2}(y_3) = -\frac{1}{4} \log_2 \left(\frac{1}{4}\right) - \left(\frac{3}{4}\right) \log_2 \left(\frac{3}{4}\right) = 0.8113. \quad (7.16)$$

将 $H(S_1)$ 和 $H(S_2)$ 代入条件熵公式, 有:

$$H(S/x_1) = (|S_1|/|S|)H(S_1) + (|S_2|/|S|)H(S_2) = \left(\frac{4}{8}\right) \times 1.5 + \left(\frac{4}{8}\right) \times 0.8113 = 1.1557. \quad (7.17)$$

同样道理, 可以求得:

$$H(S/x_2) = 1.1557 \quad (7.18)$$

$$H(S/x_3) = 0.75 \quad (7.19)$$

可见, 应该选择属性 x_3 对根节点进行扩展. 用 x_3 对 S 扩展后所得到的得到部分决策树如图7-9所示.

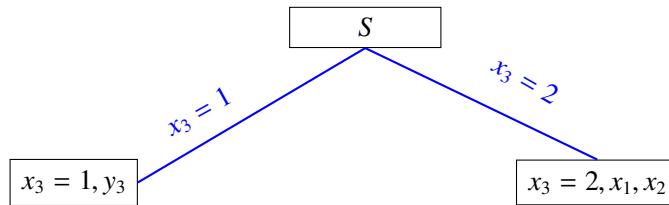


图 7-9 部分决策树

在该树中, 节点 “ $x_3 = 1, y_3$ ” 表示当 x_3 的属性值为 1 时, 得到决策方案 y_3 . 由于 y_3 已是具体的决策方案, 故该节点的信息熵为 0, 已经为叶节点.

节点 “ $x_3 = 2, x_1, x_2$ ” 的含义是 “当 x_3 的属性值为 2 时, 还需要考虑属性 x_1, x_2 ”, 它是一个中间节点, 还需要继续扩展.

至于节点 “ $x_3 = 2, x_1, x_2$ ”, 其扩展方法与上面的过程类似. 通过计算可知, 该节点对属性 x_1 和 x_2 , 其条件熵均为 1. 由于它对属性 x_1 和 x_2 的条件熵相同, 因此可以先选择 x_1 , 也可以先选择 x_2 , 本例是先选择 x_2 .

依此进行下去, 可得到如图7-10所示的最终的决策树. 在该决策树中, 各节点的含义与图7-9类似.

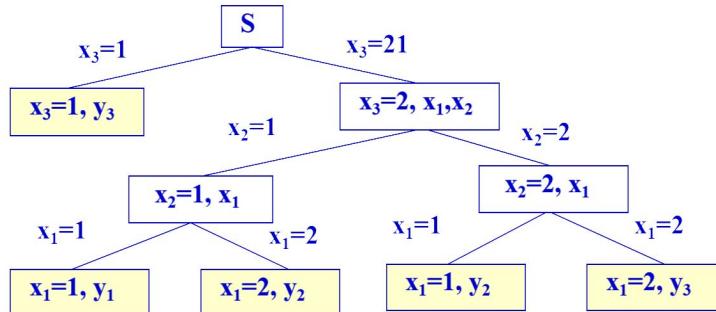


图 7-10 最终决策树

7.6.2 解释学习的空间描述

解释学习涉及三个不同的空间: 例子空间, 概念空间和概念描述空间. 三个空间及它们之间的关系如图7-11所示.

概念描述空间是所有概念描述的集合, 其中的概念描述可分为两大类, 一类是可操作的, 另一类是不可操作的. 所谓可操作是指一个概念描述能有效的用于识别相应概念的例子. 否则是不可操作的. 解释学习的任务就是要把不可操作的概念描述转化为可操作的概念描述.

概念空间是学习过程能够描述的所有概念的集合. 例子空间是用于问题求解的例子集合

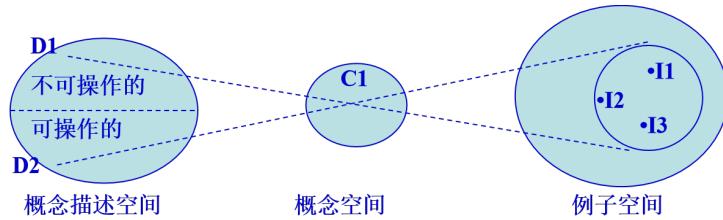


图 7-11 解释学习的三个空间

7.6.3 解释学习的模型

模型组成说明:

- 1) EXL 为学习系统
- 2) KB 为领域知识库, 它是不同概念描述之间进行转换所使用的规则集合;
- 3) PS 为执行系统;
- 4) D1 是输入的概念描述, 一般为不可操作的;
- 5) D2 是学习结束时输出的概念描述, 它是可操作的.

执行过程: 先由 EXL 接受输入的概念描述 D1, 然后再根据 KB 中的知识对 D1 进行不同描述的转换, 并由 PS 对每个转换结果进行测试, 直到被 PS 所接受, 即为可操作的概念描述 D2 为止; 最后输出 D2.

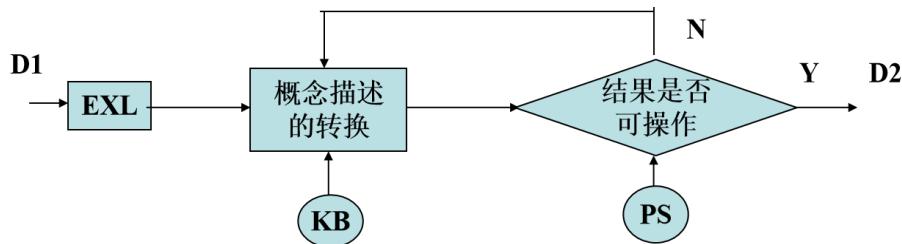


图 7-12 解释学习模型

7.6.4 解释学习的基本原理

本节主要讨论米切尔等人提出的解释泛化学习方法.

解释泛化学习方法-1 其基本思想: 先对某一情况建立一个解释结构, 然后在此解释结构进行概括, 获取一般性控制知识.

其一般性描述为: 已知:

- ▶ 目标概念 GC(Goal Concept);
 - 训练实例 TE(Training Example);
 - 领域理论 DT(Domain Theory);
 - 操作性标准 OC(Operationality Criterion).
- ▶ 求出: 满足 OC 的关于 GC 的充分概念描述, 其中: 目标概念 GC 是要学习概念的描述.
- ▶ 训练实例 TE 为学习系统提供的一个实例;
- ▶ 领域理论 DT 是相关领域的事实和规则, 即为背景知识;
- ▶ 操作性标准 OC 用于指导学习系统对用来描述目标的概念进行舍取等的控制性知识.

解释泛化学习方法-2 本节主要讨论米切尔等人提出的解释泛化学习方法. 解释泛化学习的基本思想: 先对某一情况建立一个解释结构, 然后在此解释结构进行概括, 获取一般性控制知识.

其一般性描述为: 已知:

- ▶ 目标概念 GC(Goal Concept);
 - 训练实例 TE(Training Example);
 - 领域理论 DT(Domain Theory);
 - 操作性标准 OC(Operationality Criterion).
- ▶ 求出: 满足 OC 的关于 GC 的充分概念描述.

其中: 目标概念 GC 是要学习概念的描述.

- ▶ 训练实例 TE 是为学习系统提供的一个实例;
- ▶ 领域理论 DT 是相关领域的事实和规则, 即为背景知识;
- ▶ 操作性标准 OC 用于指导学习系统对用来描述目标的概念进行舍取等的控制性知识.

其任务是要证明提供给系统的训练实例为什么是目标概念的一个实例. 为此, 系统应从目标开始反向推理, 根据知识库中已有的事实和规则分解目标, 直到求解结束. 一旦得到解, 便完成了该问题的证明, 同时也获得了一个解释结构.

例 7.11 假设要学习的目标是“一个物体 x 可以安全地放置在另一个物体 y 的上面”. 即目标概念:

Safe-to-Stack (x, y)

训练实例(是一些描述物体 obj₁ 与 obj₂ 的事实):

On(obj ₁ , obj ₂)	物体 1 在物体 2 的上面
Isa(obj ₁ , book)	物体 1 是书
Isa(obj ₂ , table)	物体 2 是桌子
Volume(obj ₁ , 1)	物体 1 的体积是 1
Density(obj ₁ , 0.1)	物体 1 的密度是 0.1

领域知识是把一个物体安全地放置在另一个物体上面的准则:

$$\neg \text{Fragile}(y) \rightarrow \text{Safe-to-stack}(x, y)$$

如果 y 不是易碎的, 则 x 可以安全地放到 y 的上面

$$\text{Lighter}(x, y) \rightarrow \text{Safe-to-stack}(x, y)$$

如果 x 比 y 轻, 则 x 可以安全地放到 y 的上面

$$\text{Volume}(p, v) \wedge \text{Density}(p, d) \wedge \text{Product}(v, d, w) \rightarrow \text{Weight}(p, w)$$

如果 p 的体积是 v、密度是 d、v 乘以 d 的积是 w, 则 p 的重量是 w

$$\text{Is-a}(p, \text{table}) \rightarrow \text{Weight}(p, 5)$$

如果 p 是桌子, 则 p 的重量是 5

$$\text{Weight}(p_1, w_1) \wedge \text{Weight}(p_2, w_2) \wedge \text{Smaller}(w_1, w_2) \rightarrow \text{Lighter}(p_1, p_2)$$

如果 p₁ 的重量是 w₁, p₂ 的重量是 w₂, w₁ 比 w₂ 小, 则 p₁ 比 p₂ 轻.

其证明过程是一个由目标引导的逆向推理, 最终得到的解释树就是该例的解释结构(如图7-13).

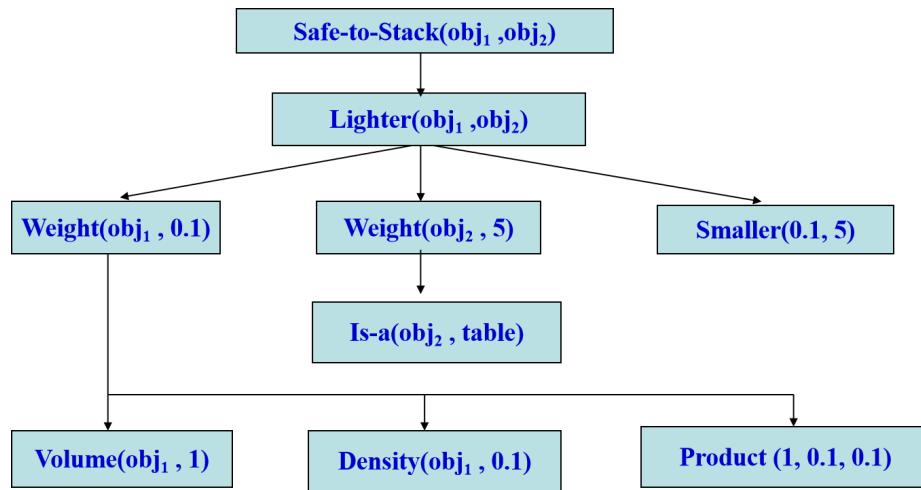


图 7-13 逆向推理的解释树

这一步的主要任务是对上一步得到的解释结构进行概括化处理, 从而得到关于目标概念的一般性知识. 进行概括化处理的常用方法是把常量转换为变量, 即把某些具体数据换成变量, 并略去某些不重要的信息, 只保留求解所必须的那些关键信息即可. 对上图的解释结构进行概括化处理以后所得到的概括化解释结构如下:

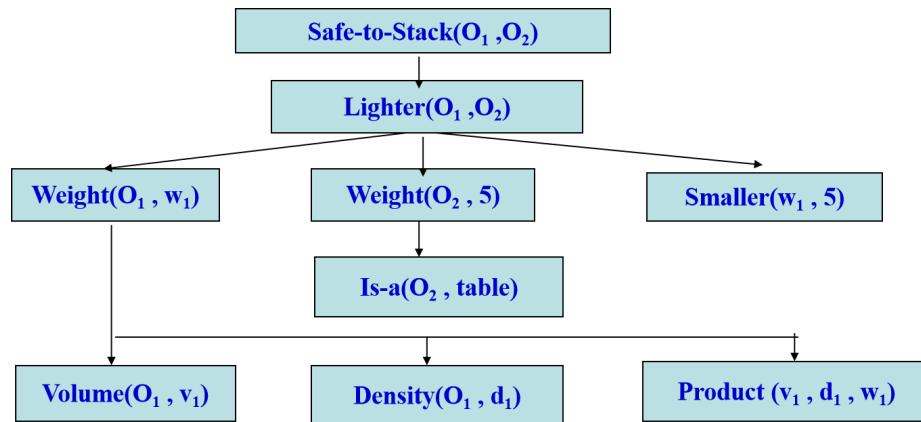


图 7-14 概括化解释结构

将该解释结构中所有的叶节点的合取作为前件, 顶点的目标概念做为后件, 略去解释结构的中间部件, 就可得到概括化的一般性知识:

$$\begin{aligned}
 & \text{Volume}(O_1, v_1) \wedge \text{Density}(O_1, d_1) \wedge \text{Product}(v_1, d_1, w_1) \\
 & \wedge \text{Is}-a(O_2, \text{table}) \wedge \text{Smaller}(w_1, 5) \rightarrow \text{Safe-to-stack}(O_1, O_2)
 \end{aligned} \tag{7.20}$$

7.6.5 浅层神经学习的心理学基础

神经生理学研究表明,人脑的神经元既是学习的基本单位,同是也是记忆的基本单位。目前,关于人脑学习和记忆机制的研究有两大学派:

- ▶ 化学学派:认为人脑经学习所获得的信息是记录在某些生物大分子之上的。例如,蛋白质、核糖核酸、神经递质,就像遗传信息是记录在DNA(脱氧核糖核酸)上一样。
- ▶ 突触修正学派:认为人脑学习所获得的信息是分布在神经元之间的突触连接上的。

按照突触修正学派的观点,人脑的学习和记忆过程实际上是一个在训练中完成的突触连接权值的修正和稳定过程。其中,学习表现为突触连接权值的修正,记忆则表现为突触连接权值的稳定。

突触修正假说已成为人工神经网络学习和记忆机制研究的心理学基础,与此对应的权值修正学派也一直是人工神经网络研究的主流学派。突触修正学派认为,人工神经网络的学习过程就是一个不断调整网络连接权值的过程。

所谓学习规则可简单地理解为学习过程中联结权值的调整规则。按照学习规则,神经学习可分为Hebb学习、纠错学习、竞争学习及随机学习等。

(1) **Hebb** 学习基本思想:如果神经网络中某一神经元同另一直接与它连接的神经元同时处于兴奋状态,那么这两个神经元之间的连接强度将得到加强,反之应该减弱。

Hebb 学习对连接权值的调整可表示为:

$$w_{ij}(t+1) = w_{ij}(t) + \eta [x_i(t)x_j(t)], \quad (7.21)$$

其中, $w_{ij}(t+1)$ 表示对时刻 t 的权值修正一次后所得到的新的权值; η 是一正常量,也称为学习因子,它取决于每次权值的修正量; $x_i(t)$ 、 $x_j(t)$ 分别表示 t 时刻第 i 个和第 j 个神经元的状态。Hebb 学习规则在人工神经网络学习中的影响比较大,但不符合生物机理。例如习惯化。

(2) **纠错学习**是一种有导师的学习过程,其基本思想:

利用神经网络的期望输出与实际输出之间的偏差作为连接权值调整的参考,并最终减少这种偏差。

最基本的误差修正规则为:连接权值的变化与神经元希望输出和实际输出之差成正比。其联结权值的计算公式为:

$$w_{ij}(t+1) = w_{ij}(t) + \eta [d_j(t) - y_j(t)] x_i(t), \quad (7.22)$$

其中, $w_{ij}(t)$ 表示时刻 t 的权值; $w_{ij}(t+1)$ 表示对时刻 t 的权值修正一次后所得到的新的权值; η 是一正常量, 也称为学习因子; $y_j(t)$ 为神经元 j 的实际输出; $d_j(t)$ 为神经元 j 的希望输出; $d_j(t) - y_j(t)$ 表示神经元 j 的输出误差; $x_i(t)$ 为第 i 个神经元的输入.

(3) 竞争学习基本思想: 网络中某一组神经元相互竞争对外界刺激模式响应的权力, 在竞争中获胜的神经元, 其连接权会向着对这一刺激模式竞争更为有利的方向发展.

(4) 随机学习基本思想: 结合随机过程、概率和能量(函数)等概念来调整网络的变量, 从而使网络的目标函数达到最大(或最小). 他不仅可以接受能量函数减少(性能得到改善)的变化, 而且还可以以某种概率分布接受使能量函数增大(性能变差)的变化.

7.6.6 感知器学习

单层感知器学习实际上是一种基于纠错学习规则, 采用迭代的思想对连接权值和阈值进行不断调整, 直到满足结束条件为止的学习算法.

假设 $X(k)$ 和 $W(k)$ 分别表示学习算法在第 k 次迭代时输入向量和权值向量, 为方便, 把阈值 θ 作为权值向量 $W(k)$ 中的第一个分量, 对应地把“-1”固定地作为输入向量 $X(k)$ 中的第一个分量. 即 $W(k)$ 和 $X(k)$ 可分别表示如下:

$$X(k) = [-1, x_1(k), x_2(k), \dots, x_n(k)] \quad (7.23)$$

$$W(k) = [\theta(k), w_1(k), w_2(k), \dots, w_n(k)]; \quad (7.24)$$

即 $x_0(k) = -1, w_0(k) = (k)$.

单层感知器学习是一种有导师学习, 它需要给出输入样本的期望输出.

假设一个样本空间可以被划分为 A, B 两类. 其功能函数的定义为: 对属于 A 类输入样本, 其功能函数的输出为 +1, 否则其输出为 -1. 对应地也可将期望输出定义为: 当输入样本属于 A 类时, 其期望输出为 +1, 否则为 -1.

单层感知器学习算法描述:

(1) 设 $t = 0$, 初始化连接权和阈值. 即给 $w_i(0)$ ($i = 1, 2, \dots, n$) 及 $\theta(0)$ 分别赋予一个较小的非零随机数, 作为初值. 其中, $w_i(0)$ 是第 0 次迭代时输入向量中第 i 个输入的连接权值; $\theta(0)$ 是第 0 次迭代时输出节点的阈值;

(2) 提供新的样本输入 $x_i(t)$ ($i = 1, 2, \dots, n$) 和期望输出 $d(t)$;

(3) 计算网络的实际输出:

$$y(t) = f \left(\sum_{i=1}^n w_i(t)x_i(t) - \theta(t) \right) \quad i = 1, 2, \dots, n \quad (7.25)$$

(4) 若 $y(t) = 1$, 不需要调整连接权值, 转 (6). 否则, 转 (5) 调整连接权值

$$w_i(t+1) = w_i(t) + \eta[d(t) - y(t)]x_i(t) \quad i = 1, 2, \dots, n \quad (7.26)$$

其中, η 是一个增益因子, 用于控制修改速度, 其值如果太大, 会影响 $w_i(t)$ 的收敛性; 如果太小, 又会使 $w_i(t)$ 的收敛速度太慢;

(6) 判断是否满足结束条件, 若满足, 算法结束; 否则, 将 t 值加 1, 转 (2) 重新执行. 这里的结束条件一般是指 $w_i(t)$ 对一切样本均稳定不变.

如果输入的两类样本是线性可分的, 则该算法就一定会收敛. 否则, 该算法将不收敛.

例 7.12 用单层感知器实现逻辑“与”运算.

解: 根据“与”运算的逻辑关系, 可将问题转换为:

输入向量: $X_1 = [0, 0, 1, 1], X_2 = [0, 1, 0, 1]$,

输出向量: $Y = [0, 0, 0, 1]$.

为减少算法的迭代次数, 设初始连接权值和阈值取值如下:

$$w_1(0) = 0.5, w_2(0) = 0.7, \theta(0) = 0.6, \quad (7.27)$$

并取增益因子 $\eta = 0.4$.

算法的学习过程如下:

设两个输入为 $x_1(0) = 0$ 和 $x_2(0) = 0$, 其期望输出为 $d(0) = 0$, 实际输出为:

$$\begin{aligned} y(0) &= f(w_1(0)x_1(0) + w_2(0)x_2(0) - \theta(0)) \\ &= f(0.5 * 0 + 0.7 * 0 - 0.6) = f(-0.6) = 0. \end{aligned} \quad (7.28)$$

实际输出与期望输出相同, 不需要调节权值.

再取下一组输入: $x_1(0) = 0$ 和 $x_2(0) = 1$, 期望输出 $d(0) = 0$, 实际输出:

$$\begin{aligned} y(0) &= f(w_1(0)x_1(0) + w_2(0)x_2(0) - \theta(0)) \\ &= f(0.5 * 0 + 0.7 * 1 - 0.6) = f(0.1) = 1. \end{aligned} \quad (7.29)$$

实际输出与期望输出不同, 需要调节权值, 其调整如下:

$$\theta(1) = \theta(0) + \eta(d(0) - y(0)) * (-1) = 0.6 + 0.4 * (0 - 1) * (-1) = 1 \quad (7.30)$$

$$w_1(1) = w_1(0) + \eta(d(0) - y(0))x_1(0) = 0.5 + 0.4 * (0 - 1) * 0 = 0.5 \quad (7.31)$$

$$w_2(1) = w_2(0) + \eta(d(0) - y(0))x_2(0) = 0.7 + 0.4 * (0 - 1) * 1 = 0.3. \quad (7.32)$$

取下一组输入: $x_1(1) = 1$ 和 $x_2(1) = 0$, 其期望输出为 $d(1) = 0$, 实际输出为:

$$\begin{aligned} y(1) &= f(w_1(1)x_1(1) + w_2(1)x_2(1) - \theta(1)) \\ &= f(0.5 * 1 + 0.3 * 0 - 1) = f(-0.51) = 0. \end{aligned} \quad (7.33)$$

实际输出与期望输出相同, 不需要调节权值.

再取下一组输入: $x_1(1) = 1$ 和 $x_2(1) = 1$, 其期望输出为 $d(1) = 1$, 实际输出为:

$$\begin{aligned} y(1) &= f(w_1(1)x_1(1) + w_2(1)x_2(1) - \theta(1)) \\ &= f(0.5 * 1 + 0.3 * 1 - 1) = f(-0.2) = 0. \end{aligned} \quad (7.34)$$

实际输出与期望输出不同, 需要调节权值, 其调整如下:

$$\theta(2) = \theta(1) + \eta(d(1) - y(1)) * (-1) = 1 + 0.4 * (1 - 0) * (-1) = 0.6 \quad (7.35)$$

$$w_1(2) = w_1(1) + \eta(d(1) - y(1))x_1(1) = 0.5 + 0.4 * (1 - 0) * 1 = 0.9 \quad (7.36)$$

$$w_2(2) = w_2(1) + \eta(d(1) - y(1))x_2(1) = 0.3 + 0.4 * (1 - 0) * 1 = 0.7. \quad (7.37)$$

取下一组输入: $x_1(2) = 0$ 和 $x_2(2) = 0$, 其期望输出为 $d(2) = 0$, 实际输出为:

$$y(2) = f(0.9 * 0 + 0.7 * 0 - 0.6) = f(-0.6) = 0. \quad (7.38)$$

实际输出与期望输出相同, 不需要调节权值.

再取下一组输入: $x_1(2) = 0$ 和 $x_2(2) = 1$, 期望输出为 $d(2) = 0$, 实际输出为:

$$y(2) = f(0.9 * 0 + 0.7 * 1 - 0.6) = f(0.1) = 1. \quad (7.39)$$

实际输出与期望输出不同, 需要调节权值, 其调整如下:

$$\theta(3) = \theta(2) + \eta(d(2) - y(2)) * (-1) = 0.6 + 0.4 * (0 - 1) * (-1) = 1 \quad (7.40)$$

$$w_1(3) = w_1(2) + \eta(d(2) - y(2))x_1(2) = 0.9 + 0.4 * (0 - 1) * 0 = 0.9 \quad (7.41)$$

$$w_2(3) = w_2(2) + \eta(d(2) - y(2))x_2(2) = 0.7 + 0.4 * (0 - 1) * 1 = 0.3 \quad (7.42)$$

实际上, 由上一章关于与运算的阈值条件可知, 此时的阈值和连接权值以满足结束条件, 算法可以结束.

对此, 可检验如下:

- 对输入: “00” 有 $y = f(0.9 * 0 + 0.3 * 0 - 1) = f(-1) = 0$;
- 对输入: “01” 有 $y = f(0.9 * 0 + 0.3 * 0.1 - 1) = f(-0.7) = 0$;

- 对输入：“10”有 $y = f(0.9 * 1 + 0.3 * 0 - 1) = f(-0.1) = 0$;
- 对输入：“11”有 $y = f(0.9 * 1 + 0.3 * 1 - 1) = f(0.2) = 0$.

BP 网络学习过程是一个对给定训练模式, 利用传播公式, 沿着减小误差的方向不断调整网络连接权值和阈值的过程. 需要用到以下几个符号:

$\$O_i\$$: 节点 i 的输出;
 $\$I_j\$$: 接点 j 的输入;
 $\$w_{ij}\$$: 从节点 i 到节点 j 的连接权值;
 θ_j : 节点 j 的阈值;
 $\$y_k\$$: 输出层上节点 k 的实际输出;
 $\$d_k\$$: 输出层上节点 k 的期望输出.

显然, 对隐含节点 j 有:

$$\begin{aligned} I_j &= \sum_i w_{ij} O_i \\ O_j &= f(I_j - \theta_j) \end{aligned}$$

在 BP 算法学习过程中, 可以采用如下公式计算各输出节点的误差:

$$e = \frac{1}{2} \sum_k (d_k - y_k)^2.$$

连接权值的修改由下式计算:

$$w_{jk}(t+1) = w_{jk}(t) + \Delta w_{jk},$$

其中, $w_{jk}(t)$ 和 $w_{jk}(t+1)$ 分别是时刻 t 和 $t+1$ 时, 从节点 j 到节点 k 的连接权值; Δw_{jk} 是连接权值的变化量.

为了使连接权值能沿着 E 的梯度变化方向逐渐改善, 网络逐渐收敛, BP 算法按如下公式计算 Δw_{jk} :

$$\Delta w_{jk} = -\eta \frac{\partial e}{\partial w_{jk}},$$

其中, η 为增益因子, $\frac{\partial e}{\partial w_{jk}}$ 由下式计算:

$$\frac{\partial e}{\partial w_{jk}} = \frac{\partial e}{\partial I_k} \frac{\partial I_k}{\partial w_{jk}}.$$

由于

$$I_k = \sum_j w_{jk} O_j.$$

故有

$$\frac{\partial I_k}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} \sum_j w_{jk} O_j = O_j.$$

令局部梯度

$$\delta_k = \frac{\partial e}{\partial I_k}.$$

故有

$$\Delta w_{jk} = -\eta \frac{\partial e}{\partial w_{jk}} = -\eta \delta_k O_j.$$

计算时, 需要区分节点 k 是输出层上的节点, 还是隐含层上的节点. 如果节点 k 是输出层上的节点, 则有 $O_k = y_k$, 因此

$$\delta_k = \frac{\partial e}{\partial I_k} = \frac{\partial e}{\partial y_k} \frac{\partial y_k}{\partial I_k}.$$

由于

$$\begin{aligned} \frac{\partial e}{\partial y_k} &= -(d_k - y_k) \\ \frac{\partial y_k}{\partial I_k} &= f'(I_k) \end{aligned}$$

所以

$$\begin{aligned} \delta_k &= -(d_k - y_k) f'(I_k) \\ \Delta w_{jk} &= \eta (d_k - y_k) f'(I_k) O_j \end{aligned}$$

如果节点 k 不是输出层上的节点, 则它是隐含层上的节点的, 此时:

$$\delta_k = \frac{\partial e}{\partial I_k} = \frac{\partial e}{\partial O_k} \frac{\partial O_k}{\partial I_k} = \frac{\partial e}{\partial O_k} f'(I_k),$$

其中, $\frac{\partial e}{\partial O_k}$ 是一个隐函数求导问题, 略去推导过程, 其结果为:

$$\frac{\partial e}{\partial O_k} = \sum_m \delta_m w_{km}$$

所以

$$\delta_k = f'(I_k) \sum_m \delta_m w_{km}$$

这说明, 低层节点的 δ 值是通过上一层节点的 δ 值来计算的. 这样, 我们就可以先计算出输出层上的 δ 值, 然后把它返回到较低层上, 并计算出各较低层上节点的 δ 值.

7.6.7 BP 网络学习

- (1) 初始化网络及学习参数, 将各节点的连接权值、阈值赋予 $[-1, 1]$ 区间的一个随机数;
- (2) 提供训练模式, 即从训练模式集合中选出一个训练模式送入网络;
- (3) 正向传播过程, 即对给定输入模式, 计算输出模式, 并将其与期望模式比较, 若有误差则执行 (4), 否则返回 (2), 提供下一个训练模式;
- (4) 反向传播过程, 即从输出层反向计算到第一隐含层, 按以下方式逐层修正各单元的连接权值:

- ① 计算同一层单元的误差
 - ② 按下式修正连接权值和阈值
- 对连接权值, 修正公式为:

$$w_{jk}(t+1) = w_{jk}(t) + \eta \delta_k O_j \quad (7.43)$$

对阈值, 可按照连接权值的学习方式进行, 只是要把阈值设想为神经元的连接权值, 并假定其输入信号总为单位值 1 即可.

反复执行上述修正过程, 直到满足期望的输出模式为止.

- (5) 返回第 (2) 步, 对训练模式集中的每一个训练模式重复第 (2) 到第 (3) 步, 直到训练模式集中的每一个训练模式都满足期望输出为止.

7.6.8 Hopfield 网络学习

Hopfield 网络学习的过程实际上是一个从网络初始状态向其稳定状态过渡的过程. 而网络的稳定性又是通过能量函数来描述的. 这里主要针对离散 Hopfield 网络讨论其能量函数和学习算法. 离散 Hopfield 网络的能量函数可定义为:

$$E = -\frac{1}{2} \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n w_{ij} v_i v_j + \sum_{i=1}^n \theta_i v_i \quad (7.44)$$

式中, n 是网络中的神经元个数, w_{ij} 是第 i 个神经元和第 j 个神经元之间的连接权值, 且有 $w_{ij} = w_{ji}$; v_i 和 v_j 分别是第 i 个神经元和第 j 个神经元的输出; θ_i 是第 i 个神经元的阈值. 以证明, 当一神经元 k 的状态由 “0” 变为 “1” 时, 网络能量的变化为:

$$\Delta E = - \left(\sum_{\substack{j=1 \\ j \neq k}}^n w_{kj} v_j - \theta_k \right) \quad (7.45)$$

此时, 由于神经元 k 的状态由 “0” 变为 “1”, 因此有

$$\sum_{\substack{j=1 \\ j \neq k}}^n w_{kj} v_j - \theta_k > 0; \quad (7.46)$$

即 $\Delta E < 0$.

同理可证, 若神经元 k 的状态由 “1” 变为 “0” 时, 网络能量函数的变化为:

$$\Delta E = \sum_{\substack{j=1 \\ j \neq k}}^n w_{kj} v_j - \theta_k \quad (7.47)$$

此时, 由于神经元 k 的状态由 “1” 变为 “0”, 因此有

$$\sum_{\substack{j=1 \\ j \neq k}}^n w_{kj} v_j - \theta_k < 0; \quad (7.48)$$

即 $\Delta E < 0$.

可见, 无论神经元的状态由 “0” 变为 “1”, 还是由 “1” 变为 “0”, 都总有 $\delta E < 0$. 它说明离散 Hopfield 网络在运行中, 其能量函数总是在不断降低的, 最终将趋于稳定状态.

例 7.13 如图所示的三个节点的 Hopfield 网络, 若给定的初始状态为:

$$V_0 = \{1, 0, 1\}.$$

各节点之间的联结权值为:

$$w_{12} = w_{21} = 1, w_{13} = w_{31} = -2, w_{23} = w_{32} = 3.$$

各节点的阈值为

$$\theta_1 = -1, \theta_2 = 2, \theta_3 = 1.$$

请计算在此状态下的网络能量.

解:

$$\begin{aligned} E &= -\frac{1}{2} (w_{12}v_1v_2 + w_{13}v_1v_3 + w_{21}v_2v_1 + w_{23}v_2v_3 + w_{31}v_3v_1 + w_{32}v_3v_2) + \theta_1v_1 + \theta_2v_2 + \theta_3v_3 \\ &= -(w_{12}v_1v_2 + w_{13}v_1v_3 + w_{23}v_2v_3) + \theta_1v_1 + \theta_2v_2 + \theta_3v_3 \\ &= -(1 \times 1 \times 0 + (-2) \times 1 \times 1 + 3 \times 0 \times 1) + (-1) \times 1 + 2 \times 0 + 1 \times 1 \\ &= 2. \end{aligned} \quad (7.49)$$

(1) 设置互连权值

$$w_{ij} = \begin{cases} \sum_{s=1}^m x_i^s x_j^s, & i \neq j \\ 0, & i = j, i \geq 1, j \leq n \end{cases} \quad (7.50)$$

其中, x_{is} 为 S 型样例 (即记忆模式) 的第 i 个分量, 它可以为 1 或 0(或-1), 样例类别数为 m , 节点数为 n .

(2) 对未知类别的样例初始化

$$y_i(i) = x_i, \quad 1 \leq i \leq n, \quad (7.51)$$

其中, $y_i(t)$ 为节点 i 时刻 t 的输出, $y_i(0)$ 是节点的初值; x_i 为输入样本的第 i 个分量.

(3) 迭代运算

$$y_i(t+1) = f \left(\sum_{j=1}^n w_{ij} y_j(t) \right), \quad 1 \leq i \leq n, \quad (7.52)$$

其中, 函数 f 为阈值型. 重复这一步骤, 直到新的迭代不能再改变节点的输出为止, 即收敛为止. 这时, 各节点的输出与输入样例达到最佳匹配. 否则转第 (2) 步继续.

7.7 深度神经网络——DNN

2006 年, Hinton 利用预训练方法缓解了局部最优解问题, 将隐含层推动到了 7 层 [**Hinton2006-9587**], 神经网络真正意义上有了“深度”, 由此揭开了深度学习的热潮. 这里的“深度”并没有固定的定义——在语音识别中 4 层网络就能够被认为是“较深”, 而在图像识别中常见到 20 层以上的网络. 为了克服梯度消失, ReLU、maxout 等传输函数代替了 sigmoid, 形成了如今 DNN 的基本形式. 单从结构上来说, 全连接的 DNN 和图7-15的多层感知机是没有任何区别的.

深度学习并不是一种独立的学习方法, 会用到有监督和无监督的学习方法来训练深度神经网络. 但由于近几年该领域发展迅猛, 一些特有的学习手段相继被提出 (如残差网络), 因此越来越多的人将其单独看作一种学习的方法. 最初的深度学习是利用深度神经网络来解决特征表达的一种学习过程. 深度神经网络本身并不是一个全新的概念, 可大致理解为包含多个隐含层的神经网络结构. 为了提高深层神经网络的训练效果, 人们对神经元的连接方法和激活函数等方面做出相应的调整. 其实有不少想法早年间也曾有过, 但由于当时用于训练的数据量不足、计算能力落后, 因此最终的效果不尽如人意. 但随着近几年互联网的成熟, 以及 GPU 系统的提高, 深度学习的潜力逐渐被挖掘, 深度学

习以摧枯拉朽般地效果实现了各种任务,使得似乎所有的机器辅助功能都变为可能. 无人驾驶汽车和预防性医疗保健,甚至影视剧的推荐系统.

DNN 是用了很多浅层神经网络的技巧之外的多层网络,它能够“深”到解决神经网络很难解决的问题. 深度学习则是 AI 训练的重要手段之一,学习要靠硬件和算法支撑这方面. 在深层学习方面,存在来给个阵营: Intel 力挺 CPU, NVIDIA 则力挺 GPU. 2020 年 3 月, Intel 实验室联合美国莱斯大学宣布了一种突破性的深度学习新算法 SLIDE. SLIDE 基于散列开发,而非当前最富盛名的 BP 算法(反向传播算法)所基于的矩阵乘法. 借助 SLIDE, CPU 用于传统 AI 模型深度学习训练的效率大大提升.

例 7.14 一套拥有 44 个 Xeon 核心的平台和一套由 8 张 NVIDIA V100 加速卡支撑的平台(TensorFlow 框架,价值 10 万美元)执行相同的训练任务,前者用 1 小时,后者用了 3.5 小时. Intel 还表示,它们这套平台尚未充分优化,还是“残血”状态,比如处理器的 DLBoost 并未启用. 不过,这套 44 核至强平台的 CPU 型号并未公布,一说就是 22 核心 44 线程的至强铂金 6238,一说是双路至强铂金 6238,还有可能是未发布的产品.

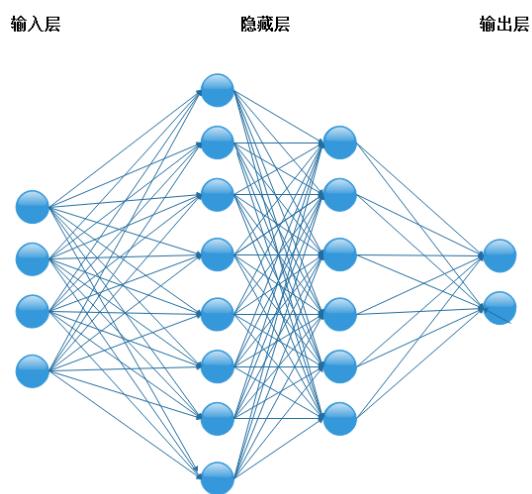


图 7-15 多层感知机

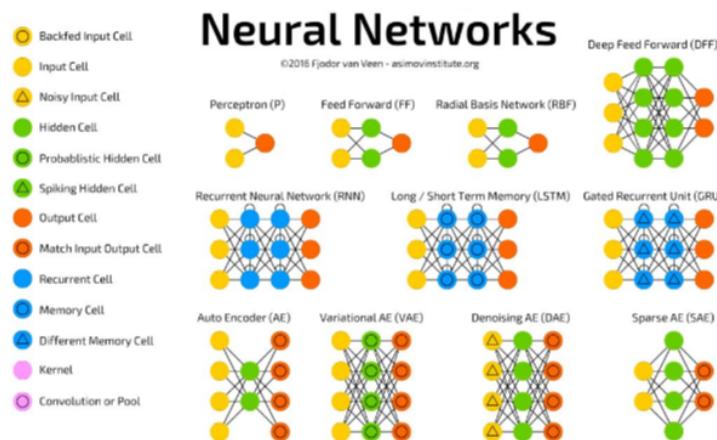


图 7-16 DNN 的各种结构 (1)

Github 上[DenseNet](#)的其他实现方式.

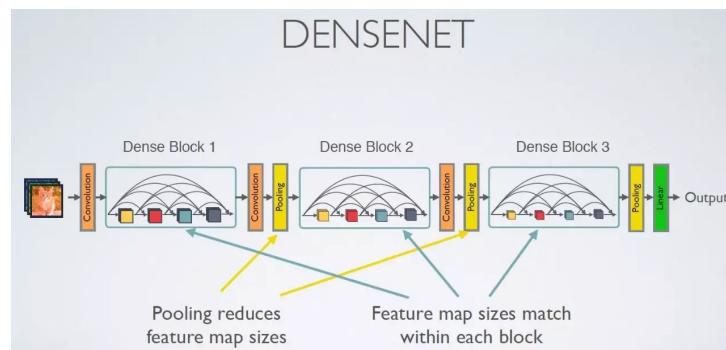


图 7-17 Desnenet 结构

使用 Matlab 中的 DNN 设计 APP 可以很快设计出 DNN 网络 7-18.

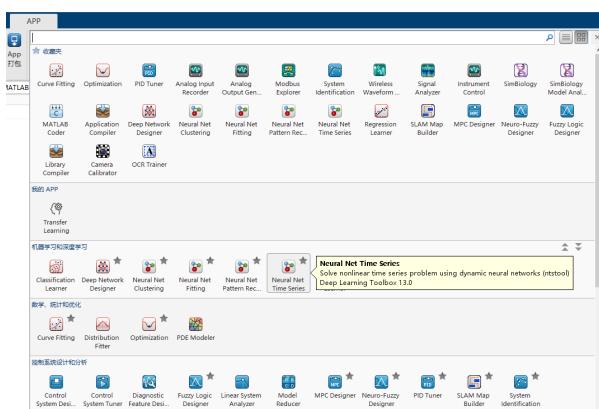


图 7-18 Matlab 中的 DNN 设计 APP

7.7.1 高速公路网络 (Highway network)

高速公路网络和深度残差学习 (DRL) 进一步避免了梯度消失, 网络层数达到了前所未有的一百多层 (深度残差学习: 152 层) [HeCVPR2016-9590, NIPS2015-5850]. 高速公路网络受 LSTM 门机制的启发, 给网络添加一个传递门和一个搬运门. Highway 网络再输入某一层网络的数据一部分经过非线性变换, 另一部分直接从该网络跨过去不做任何转换, 就想走在高速公路上一样, 而多少的数据需要非线性变换, 多少的数据可以直接跨过去, 是由一个权值矩阵和输入数据共同决定的.

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W}_H) \cdot T(\mathbf{x}, \mathbf{W}_T) + \mathbf{x} \cdot C(\mathbf{x}, \mathbf{W}_C) \quad (7.53)$$

其中 \mathbf{y} 向量由两项组成. T 叫做 transform gate, C 叫做 carry gate. C 和 T 的激活函数都是 sigmoid 函数. $T = (b_1, b_2, \dots, b_n)$, 其中每个数字都是 $(0, 1)$ 之间的浮点数, 代表 y 中由 x 变化后的内容所占的比例; $C = (c_1, c_2, \dots, c_n)$, 其中每个数字也都是 $(0, 1)$ 之间的浮点数, 代表 y 中由 x 本身内容所占的比例; (为了简便起见, 有时候令 C 可以简化, 取 $C = 1 - T$. $\mathbf{1}$ 代表了维度和 T 向量样长的单位向量). 需要注意的是, 由于是点乘, 当 $C(x, WC)$ 取了 $1 - T(x, WT)$ 之后那么 $x, y, H(x, WH), T(x, WT)$ 必须是同样的维度相容, 以保证可乘. 如果我们想更改 x 的维度变成 B 的维度的话, 一种方法是采用填充零和下采样的方法, 或者是引入一个维度为 $A * B$ 的变换矩阵, 使每次都乘上这个矩阵.

高速公路网络就是在输出和网络层之间加了一个连接, 直接让输入 X 的信息直接通过, 不需要通过神经网络层, 跟高速公路一样. 高速公路网络主要解决的是多层深度神经网络的训练收敛问题, 即使层数很多也可以使用简单的方法比方说反向传播来进行训练, 保证合理的迭代范围内收敛, 而传统的网络是很难保证收敛的. 如图7-19所示:

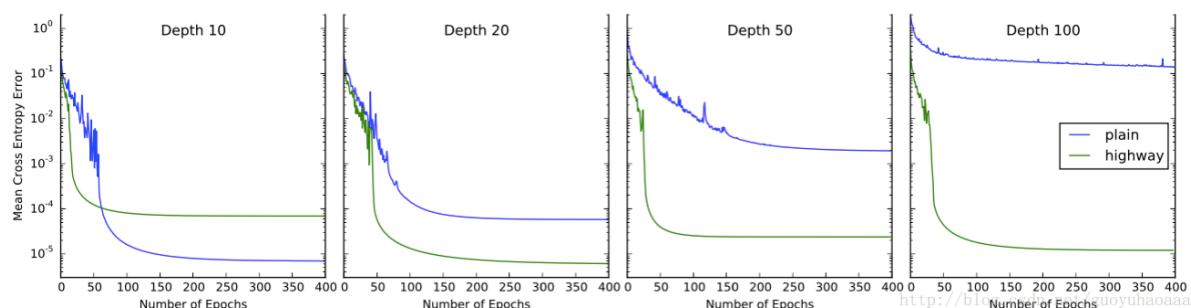


图 7-19 Highway 网络的迭代误差

当网络很深的时候, 使用了 Highway 的网络更容易收敛.

7.7.2 残差网络

残差网络用于解决由深度增加而出现错误率增加的情况, 通过拟合残差映射来取代直接的拟合映射. 将一个底层的映射记作 $H(x)$, 用一个堆叠非线性的映射记作 $F(x) = H(x) - x$, 原始的映射变为 $F(x) + x$, 我们假设残差映射的优化易于直接映射, 如果某个恒等的映射是最优的, 那么把残差变为 0 比非线性层的堆叠更简单. $F(x) + x$ 的形式可以通过前馈神经网络中的快捷连接来实现, 快捷连接是指跨过或者跳过一定的层, 在这里 shortcut connections 正好实现了 identity mapping, 如图7–20

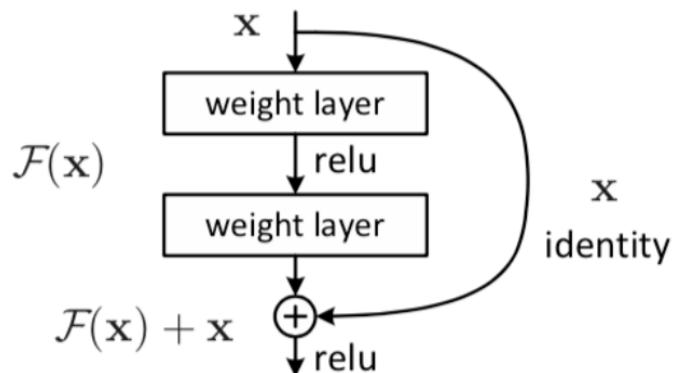


图 7–20 残差学习网络的结构

网络要学习的是 $H(x)$, 那么由于图中 identity x 之间跨过了 2 层, 那么其实相当于拟合的是 $F(x) = H(x) - x$, 这就是残差概念的来源, 这是论文里的说法. 作者提出的结构打破了传统的神经网络 $n - 1$ 层的输出只能给 n 层作为输入的惯例, 使某一层的输出可以直接跨过几层作为后面某一层的输入. 图7–21就是其构造深度残差网络的构思来源图, 一个是 56 层的网络一个是 20 层的网络, 从原理上来说其实 56 层网络的解空间是包括了 20 层网络的解空间的, 换而言之也就是说, 56 层网络取得的性能应该大于等于 20 层网络的性能的. 但是从训练的迭代过程来看, 56 层的网络无论从训练误差来看还是测试误差来看, 误差都大于 20 层的网络(这也说明了为什么这不是过拟合现象, 因为 56 层网络本身的训练误差都没有降下去). 导致这个原因就是虽然 56 层网络的解空间包含了 20 层网络的解空间, 训练网络用的随机梯度下降策略往往得不到全局最优解, 而是局部的最优解, 显而易见 56 层网络的解空间更加的复杂, 所以导致使用随机梯度下降算法无法解到最优解.

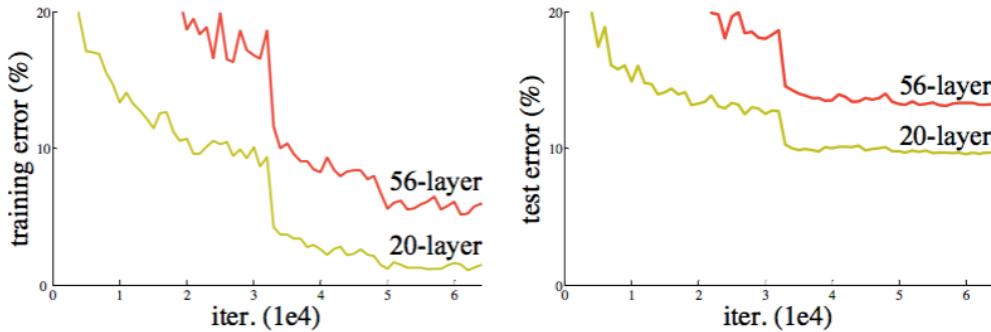


图 7-21 CIFAR-10 上 20 层和 56 层训练误差 (左) 和测试 (右), ImageNet 也呈现类似现象

在构造这个网络的时候, 如果 20 层的网络可以取得非常好的结果, 在构造 56 层网络的时候前 20 层从 20 层网络中拷贝过来, 后面的 36 层只做 identity map 至少效果不会差于 20 层的网络. 于是乎深度残差的网络就提出了, 这个思想其实不复杂, 说白了打破了每一层网络输入只能来自于上一层网络输出的规律, 可以让一些网络的输出直接跳过几层到达后面的输入. 这样的网络确实也取得了非常好的效果. 另外要注意的是, 在真正训练的时候, 有几点小技巧要注意:

- 1、层与层之间使用批量正则化技术, 否则由于网络过深会导致梯度消失的问题, 导致网络训练无法收敛;
- 2、为了保持每一层网络的参数差不多, 每当经过了池化层输入的维度减少了一般, 那么滤子的个数就要增加一倍.

其实深度残差网络和 Highway 网络这两种网络结构都能够让一部分的数据可以跳过某些变换层, 而直接到后面的层中去, 只不过 Highway 网络需要一个权值来控制每次直接通过的数据量, 而深度残差网络就直接让一部分数据通到了后面.

7.8 卷积神经网络 (Convolutional Neural Networks, CNN)

人工神经网络包括多个神经网络层 (卷积层、全连接层、LSTM 等), 每一层又包括很多神经元, 超过三层的非线性神经网络都可以被成为深度神经网络. 神经网络允许机器通过识别可以广泛应用的模式来学习, 其应用包括语言识别、视觉物体识别、自动驾驶汽车导航, 机械臂的灵活抓取等. 作为一个科学领域, 神经网络的发端可以追溯到上世纪 40 年代, 但直到最近几年, 该领域才取得了显著进展. 通俗的讲, 深度学习的模型可以视为是输入到输出的映射函数, 比如中文到英文的映射, 足够深的神经网络理论上可以拟合任何复杂的函数 “深度” 是指网络中的层数一层数越多, 网络越深. 传统的神经

网络只包含 2 层或 3 层, 而深度网络可能有几百层。深度学习是机器学习的一个类型, 该类型的模型直接从图像、文本或声音中学习执行分类任务。通常使用神经网络架构实现深度学习。深度学习由一连串可微分的参数化的层组成, 而且是用反向传播算法端到端地训练的。

 **注 7.15.** 深度学习领域被引最高的论文: 神经网络实际上是一个泛逼近器。用形象点的类比来说, 可以把神经网络视为一个超级大的万能工匠玩具——大量互相连接的节点阵列, 可以训练来完成许多任务, 从语言翻译到识别物体、识别人类讲话。作为“LSTM 之父”的 *Jürgen Schmidhuber* 在深度学习领域的贡献仍然获得了整个社区的认可。Google Scholar 2019 年统计发现: *Hochreiter* 和 *Schmidhuber* 1997 年发表的 LSTM 论文成为了 20 世纪被引最高的深度学习研究论文 [**HochreiterNC1997**]。

François Chollet, Keras 作者、谷歌大脑高级研究员 François Chollet 的观点: 深度学习应该指的是一种表征学习方法, 其中的模型是由一连串的模块组成的 (一般都会堆成一个多层次的或者金字塔形的模型, 这也就是「深度」的由来), 而其中的每一个模块分别拿出来训练之后都可以作为独立的特征提取器。CNN 专门解决图像问题的特征提取。

例 7.16 如果你有一个卷积网络模型, 然后你用 ADMM 训练它的权重, 它就不是深度学习了吗? 一个自己学习特征的 HMAX 模型就不是深度学习了吗? 甚至于, 用贪婪算法逐层训练的深度神经网络就不是深度学习了吗? 要我说的话, 它们都是深度学习。

深度学习中的 AlexNet 一开始有 8 层网络, 约 6000 万个参数; 发展到 2014 年的 VGG-19 模型, 有 19 层网络, 大概 1.43 亿个参数, 深度学习模型越来越复杂, 如果无法直接在存储中处理数据和模型, 就会对带宽造成巨大堵塞, 对效率产生很大影响。这就要求提高边缘端的计算能力, 主要的推进方向有两个: 一是从神经网络模型的表示、计算、存储、和学习等方面, 通过压缩、剪枝、量化等方法来简化模型; 二是使用专用加速器加速深度学习应用, 将 32bits 浮点数转换为定点数, 减少延迟, 降低能耗。深度学习常用的神经网络算法图 7-16-图 7-22。CNN 相对于 DNN, 比较特殊的是卷积层和池化层, 在熟悉 DNN 的基础上, 理解卷积层和池化层的原理就学会了 CNN。

深度学习在神经网络研究者中间已经被讨论了超过二十年。最近深度学习的发展是由相对较小的算法改进以及大数据集模型和计算机硬件发展驱动的。深度学习是一种特定形式的机器学习, 训练多层次神经网络。深度学习近年来非常流行, 引领了图像识别和语音识别等领域的突破性进展。(Stuart Russell: 人工智能基础概念与 34 个误区。) 深度学习已经代替了机器学习的地位。

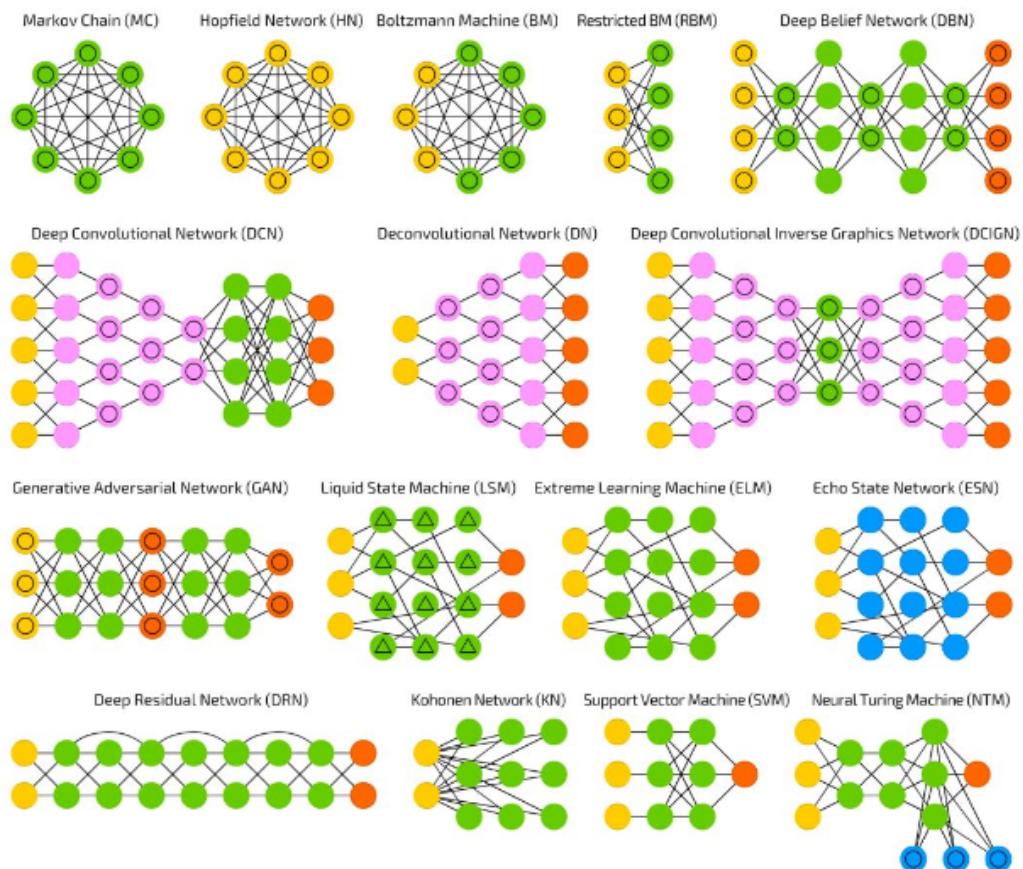


图 7-22 DNN 的各种结构 (2)

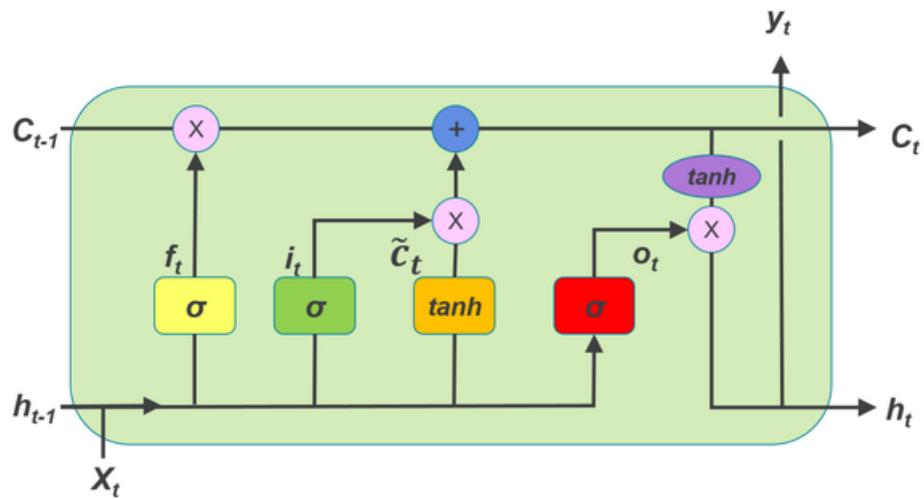


图 7-23 LSTM 结构

CNN 是第一个真正成功训练多层网络结构的学习算法。它利用局部连接，权重共享。

和下采样极大降低了模型的参数数量, 以提高一般 BP 网络的训练性能。在 CNN 中, 图像的一小部分 (局部感受区域) 作为层级结构的最低层的输入, 神经元只与上一层中部分神经元相连, 并且不同的神经元共享权值, 每层通过不同的卷积核去获得观测数据的不同特征, 然后通过下采样 (池化) 将语义上相似的特征合并起来, 降低特征图的空间分辨率。CNN 能够提取对平移、缩放和旋转不变的观测数据的显著特征, 在图像处理、语音处理等领域得到了广泛而深入的应用。

CNN 应用场景——计算机视觉问题 ① 图像分类 (Classification): 识别出该图像属于哪个类别。

- ② 目标定位 (Localization): 在图像分类的基础上, 定位找到对象在图像中的位置。
- ③ 目标检测 (Detection): 给出在图像内所有该类别的目标和对应的位置。
- ④ 图像分割 (Segmentation): 具体到像素级别的图像处理。
- ⑤ 无人驾驶汽车在接近人行横道线时减速。
- ⑥ ATM 拒收假钞。
- ⑦ 智能手机应用程序即时翻译国外路标。
- ⑧ UCLA 研究人员建造了一种高级显微镜, 能产生高维的数据集, 用来训练深度学习网络, 在组织检体中识别癌细胞。

深度学习特别适合鉴别应用场景, 比如人脸辨识、文本翻译、语音识别以及高级驾驶辅助系统 (包括车道分类和交通标志识别)。先进的工具和技术极大改进了深度学习算法, 达到了很高的水平, 在图像分类上能够超越人类, 能打败世界最优秀的围棋选手, 还能实现语音控制助理功能, 如 Amazon Echo 和 Google Home, 可用来查找和下载您喜欢的新歌。

三个技术助推器让这种精确度成为可能:

- ▶ 大规模带标签的数据集 ImageNet 和 PASCAL VoC 等数据集可以免费使用, 对于许多不同类型的对象训练十分有用。

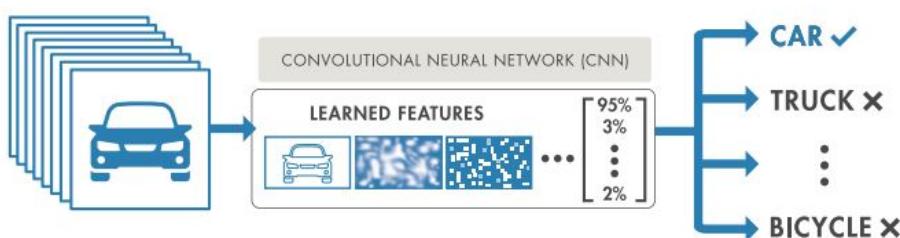


图 7-24 大规模带标签的免费数据集

- ▶ 增大的计算能力: 高性能 GPU 加快了深度学习所需的海量数据的训练速度, 训练时间从几星期减少到几小时.

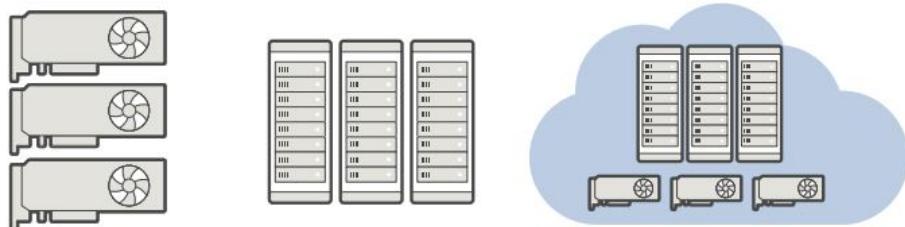


图 7-25 算力的规模

- ▶ 由专家构建的预先训练好的模型可以重新训练 AlexNet 之类的模型, 使用名为迁移学习的技术执行新识别任务. 虽然使用了 130 万张高分辨率图像训练 AlexNet 来识别 1000 个不同的对象, 但可以使用较小的数据集实现精确的迁移学习.

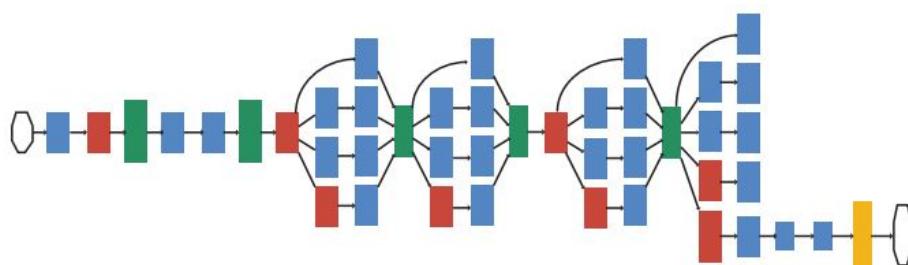


图 7-26 迁移学习模型

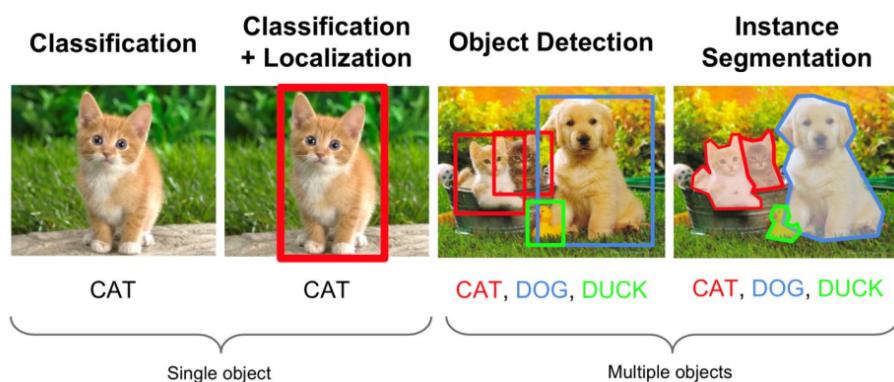


图 7-27 AlexNet 的目标分割

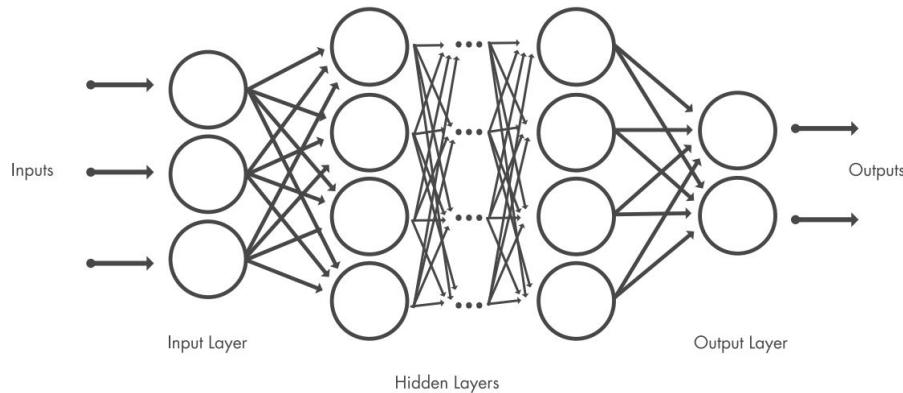


图 7-28 深度神经网络

7.8.1 卷积神经网络结构

传统的 CNN 网络只能给出图像的 LABEL, 但是在很多情况下需要对识别的物体进行首尾相连地分割, FCN 的出现给物体分割提供了一个非常重要的解决思路, 其核心就是卷积与反卷积.

7.8.1.1 卷积

对于 1 维的卷积, 离散公式与连续计算过程如下, 要记住的是原函数或者卷积函数在卷积前要翻转 180 度.

图 7-29 1 维离散卷积公式的计算过程

对于离散卷积, f 的大小是 n_1 , g 的大小是 n_2 , 卷积后的大小是 $n_1 + n_2 - 1$. 同样地, 卷积的计算需要对卷积核进行 180 度的旋转, 同时卷积核中心与需要计算的图像像素对齐, 输出结构为中心对齐像素的一个新的像素值.

例 7.17 计算出图 7-31 左上角 (即第一行第一列) 像素的卷积后像素值, 其中

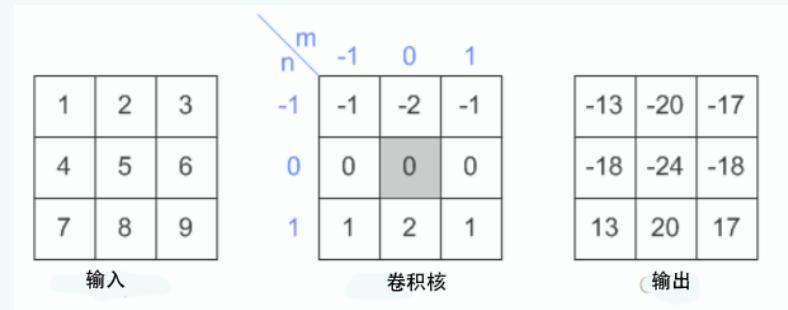


图 7-30 输入和卷积核

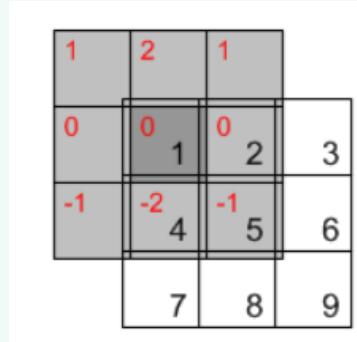


图 7-31 第一行第一列像素值

像素左上角位置记为 $[0,0]$, 图像值为 1; h 为卷积核, x 为图像值, 不足的填充 0.

$$\begin{aligned}
 y[0,0] &= x[-1, -1] \cdot h[1, 1] + x[0, -1] \cdot h[0, 1] + x[1, -1] \cdot h[-1, 1] \\
 &\quad + x[-1, 0] \cdot h[1, 0] + x[0, 0] \cdot h[0, 0] + x[1, 0] \cdot h[-1, 0] \\
 &\quad + x[-1, 1] \cdot h[1, -1] + x[0, 1] \cdot h[0, -1] + x[1, 1] \cdot h[-1, -1] \\
 &= [0, 0, 0] \cdot [1, 2, 1]^T + [0, 1, 2] \cdot [0, 0, 0]^T + [0, 4, 5] \cdot [-1, -2, -1]^T \\
 &= 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 2 \cdot 0 + 0 \cdot (-1) + 4 \cdot (-2) + 5 \cdot (-1) \\
 &= -13.
 \end{aligned}$$

通过滑动卷积核, 就可以得到整张图片的卷积结果 (图 7-30 的输出), 可以看出, 通过图像卷积后的新图像大小跟原来一样甚至变小.

例 7.18 从左到右运算后, 原像素经过卷积由 1 变成 -8(图 7-32).

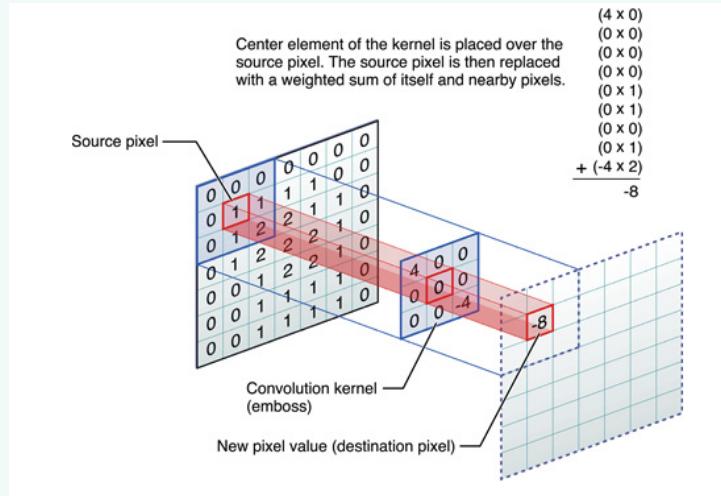


图 7-32 卷积核对图像逐像素点运算

图 7-30 经过 same 卷积计算后图像大小不变. 图 7-32 卷积计算后图像大小不变. 图 7-33 经过 valid 卷积后图像变小, 是因为没有对所用像素进行卷积计算.

图 7-33 valid 卷积运算

Matlab 中 2 维卷积分为完全卷积、same 卷积和 valid 卷积. 2 维完全卷积如图 7-34. 图 7-34 中蓝色区域是原图像, 白色为对应卷积所增加的填充, 通常的填充全部为 0, 绿色是卷积后图片. 卷积的滑动是从卷积核右下角与图片左上角重叠开始进行卷积, 滑动步长为 1, 卷积核的中心元素对应卷积后图像的像素点. 可以看到卷积后的图像是 4×4 , 比原图 2×2 大, 记 1 维卷积大小是 $n_1 + n_2 - 1$, 这里原图是 2×2 , 卷积核 3×3 , 卷积后结果

是 4×4 , 与一维完全对应起来, 4×4 的卷积是完整的卷积计算. 其他比它小的卷积结果都是省去了部分像素的卷积.

图 7-34 2 维完全卷积

对应图像卷积后多出部分的解释 (来自 WIKI): 多出来的部分根据实际情况可以有不同的处理方法 (这里的 full 卷积就是后面要说的反卷积), 我们可以对卷积核移动范围的不同限制, 得到 full 卷积, same 卷积 (指经过卷积的特征图和原图的 size 保持不变), valid 卷积 (滤镜全部在 image 里面的时候, 进行卷积运算, 可见滤镜的移动范围较 same 卷积更小), 三种卷积后图像大小的计算公式:

1) full 卷积: 滑动步长为 1, 图片大小为 $N_1 \times N_1$, 卷积核大小为 $N_2 \times N_2$, 卷积后图像大小: $(N_1 + N_2 - 1) \times (N_1 + N_2 - 1)$. 如图 7-34, 滑动步长为 1, 图片大小为 2×2 , 卷积核大小为 3×3 , 卷积后图像大小: 4×4 .

2) same 卷积: 滑动步长为 1, 图片大小为 $N_1 \times N_1$, 卷积核大小为 $N_2 \times N_2$, 卷积后图像大小: $N_1 \times N_1$.

3) valid 卷积: 滑动步长为 S, 图片大小为 $N_1 \times N_1$, 卷积核大小为 $N_2 \times N_2$, 卷积后图像大小: $((N_1 - N_2)/S + 1) \times ((N_1 - N_2)/S + 1)$. 如图 7-33, 滑动步长为 1, 图片大小为 5×5 , 卷积核大小为 3×3 , 卷积后图像大小: 3×3 .

7.8.1.2 反卷积 (后卷积, 转置卷积)

反卷积跟 1 维信号处理的反卷积计算不一样, FCN 作者称为后向卷积层, 有人称反卷积层应该称为转移卷积层。CNN 中有卷积层与池化层, 卷积层进行对图像卷积提取特征, 池化层对图像缩小一半筛选重要特征, 对于经典的图像识别 CNN 网络, 如 IMAGENET, 最后输出结果是 $1 \times 1 \times 1000$, 1000 是类别种类。FCN 作者和后来的研究人员都对最终 1×1 的结果使用反卷积 (事实上 FCN 作者最后的输出不是 1×1 , 是图片大小的 32 分之一, 但不影响反卷积的使用)。

例 7.19 大小为 2×2 图像的反卷积过程如图 7-35, 其中输入 2×2 , 卷积核 4×4 , 滑动步长 3, 输出 7×7 , 图 7-35 给出了输入为 2×2 的图片经过 4×4 的卷积核进行步长为 3 的反卷积运算过程。1) 输入图片每个像素进行一次 full 卷积, 根据 full 卷积大小计算可以知道每个像素的卷积后大小为 $1 + 4 - 1 = 4$, 即 4×4 大小的特征图, 输入有 4 个像素所以 4 个 4×4 的特征图。

2) 将 4 个特征图进行步长为 3 的融合 (即相加); 例如红色的特征图仍然是在原来输入位置 (左上角), 绿色还是在原来的位置 (右上角), 步长为 3 是指每隔 3 个像素进行融合, 重叠部分进行相加, 即输出的第 1 行第 4 列是由红色特征图的第一行第四列与绿色特征图的第一行第一列相加得到, 其他如此类推。

可以看出反卷积的大小是由卷积核大小与滑动步长决定, in 是输入大小, k 是卷积核大小, s 是滑动步长, out 是输出大小。得到 $out = (in - 1) * s + k$, 图 7-35 过程就可以写为 $(2 - 1) \times 3 + 4 = 7$ 。

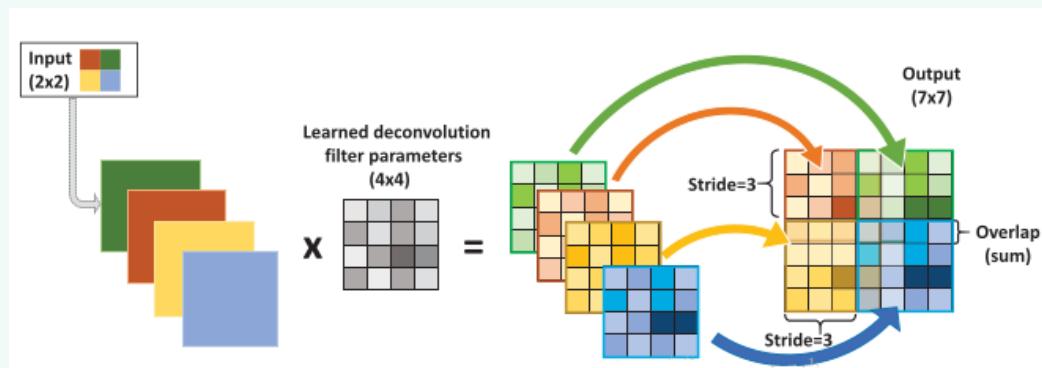


图 7-35 大小为 2×2 图像的反卷积过程

这种反卷积的使用使得图像可以变大, FCN 作者使用的方法是这里所说反卷积的一种变体, 这样就可以获得相应的像素值, 图像可以实现首尾相连地分割。图像的反卷积见图

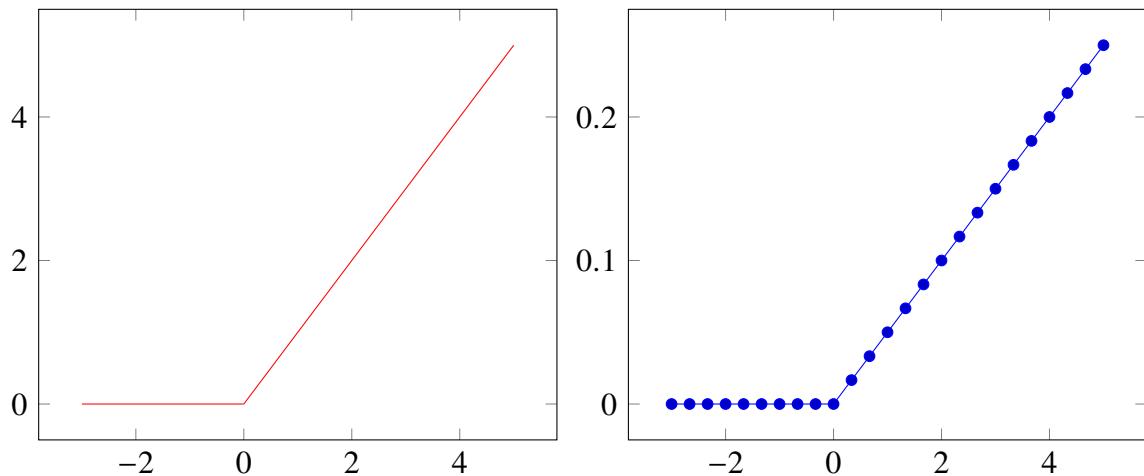
7–36的full 卷积原理.

图 7–36 反卷积运算

7.8.1.3 卷积神经网络的层结构

卷积神经网络 (CNN 或 ConvNet) 是图像和视频深度学习的最流行算法之一。像其他神经网络一样，CNN 由一个输入层、一个输出层和中间的多个隐藏层组成。

- ▶ 特征检测层：这些层对数据执行三种类型操作中的一种，即卷积、池化或修正线性单元 (ReLU, 图7–37)。

图 7-37 修正线性单元和 $\alpha = 0.05$ 的 leaky 修正线性单元

- ▶ 卷积将输入图像放进一组卷积过滤器, 每个过滤器激活图像中的某些特征.
- ▶ 池化通过执行非线性下采样, 减少网络需要学习的参数个数, 从而简化输出.
- ▶ 修正线性单元 (ReLU) 通过将负值映射到零和保持正数值, 实现更快、更高效的训练.

这三种操作在几十层或几百层上反复进行, 每一层都学习检测不同的特征.

7.8.1.4 激活函数和层范数

在线绘制各类激活函数.

注 7.20. Leaky ReLU 激活函数是在声学模型 (2013) 中提出的, 给所有负值赋予一个非零斜率 $\alpha = \frac{1}{a_i}, a_i \in (1, \infty)$.

注 7.21. CELU [hendrycks2016gelu] 定义为

$$CELU(x) = \max(0, x) + \min(0, \alpha(\exp(x/\alpha) - 1)), x \in \mathbb{R}, \alpha > 0. \quad (7.54)$$

注 7.22. 高斯线性误差单元 (GELU), 是高性能神经网络的一种激活函数. GELU 的非线性体现在它的随机正则化算子会随机地应用恒等映射和零映射到神经元的输入上. GELU 的非线性权重输入由幅值决定, 不同于 ReLUs 中由符号决定.

$$GELU(x) = xP(X \leq x) = x\Phi(x) \quad (7.55)$$

使用下式可以逼近 GELU

$$0.5x \left(1 + \tanh \left[\sqrt{2/\pi} (x + 0.044715x^3) \right] \right) \quad (7.56)$$

或者

$$x\sigma(1.702x). \quad (7.57)$$

Bent-identity

$$f(x) = \frac{\sqrt{x^2 + 1} - 1}{2} + x \quad (7.58)$$

Synthesization of many useful properties of other activations, including PReLU, ELU, and SELU and CIFAR -10 experiments validation [**Godfrey2019-9846**], 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), 6-9 Oct. 2019.

Rothaus 在 1976 年提出. 因为 Bent 函数既是非线性度最优的布尔函数, 介于 Identity 与 ReLU 之间的一种折衷选择. 它允许非线性行为, 尽管其非零导数有效提升了学习并克服了与 ReLU 相关的静默神经元的问题. 由于其导数可在 1 的任意一侧返回值, 因此它可能容易受到梯度爆炸和消失的影响.

LiSHT (Linearly Scaled Hyperbolic Tangent) function:

$$\phi(x) = x \cdot g(x), g(x) = \tanh(x) = \frac{\exp^x - \exp^{-x}}{\exp^x + \exp^{-x}} \quad (7.59)$$

Swish activation function [**Ramachandran2018**]

$$y(x) = x/(1 + e^{-x}). \quad (7.60)$$

ELiSH-The Quest for the Golden Activation Function: Proceedings of the ARW & OAGM Workshop 2019, May 9-10, 2019 in Steyr, Austria(Exponential Linear Sigmoid SquashHing)

A piecewise activation function, which shares the goodness of Swish in positive part & ELU and Sigmoid in its negative part. In Swish, Sigmoid improves the information flow and Linear avoids a vanishing gradient, which is also the main motivation to design ELiSH.

$$y(x) = \begin{cases} x/(1 + e^{-x}) & x \geq 0 \\ (e^x - 1)/(1 + e^{-x}) & x < 0 \end{cases} \quad (7.61)$$

HardELiSH as a multiplication of HardSigmoid and ELU in negative part and HardSigmoid and Linear in positive part

$$y(x) = \begin{cases} x \times \max(0, \min(1, (x + 1)/2)) & x \geq 0 \\ (e^x - 1) \times \max(0, \min(1, (x + 1)/2)) & x < 0 \end{cases} \quad (7.62)$$

Soft shrinkage function elementwise:

$$\text{Soft Shrinkage}(x) = \begin{cases} x - \lambda, & \text{if } x > \lambda \\ x + \lambda, & \text{if } x < -\lambda \\ 0, & \text{otherwise} \end{cases} \quad (7.63)$$

参数默认是 $\lambda = 0.5$.

Softsign

$$\text{Soft Sign}(x) = \frac{x}{1 + |x|} \quad (7.64)$$

Tanhshrink

$$\text{Tanhshrink}(x) = x - \text{Tanh}(x). \quad (7.65)$$

Softmin

$$\text{Softmin}(x_i) = \frac{\exp(-x_i)}{\sum_j \exp(-x_j)} \quad (7.66)$$

Softmax

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad (7.67)$$

LogSoftmax

$$\log \text{Softmax}(x_i) = \log \left(\frac{\exp(x_i)}{\sum_j \exp(x_j)} \right) \quad (7.68)$$

[AdaptiveLogSoftmaxWithLoss](#)

$$y = \frac{x - \text{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} * \gamma + \beta \quad (7.69)$$

BatchNorm1d

$$y = \frac{x - \text{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} * \gamma + \beta \quad (7.70)$$

BatchNorm2d

$$y = \frac{x - \text{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} * \gamma + \beta \quad (7.71)$$

BatchNorm3d

$$y = \frac{x - \text{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} * \gamma + \beta \quad (7.72)$$

GroupNorm

$$y = \frac{x - \text{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} * \gamma + \beta \quad (7.73)$$

SyncBatchNorm

$$y = \frac{x - \text{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} * \gamma + \beta \quad (7.74)$$

InstanceNorm1d

$$y = \frac{x - \text{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} * \gamma + \beta \quad (7.75)$$

SoftExponential

$$f(\alpha, x) = \begin{cases} -\frac{\ln(1-\alpha(x+\alpha))}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0, x \in (-\infty, \infty) \\ \frac{e^{\alpha x} - 1}{\alpha} + \alpha & \text{for } \alpha > 0 \end{cases} \quad (7.76)$$

InstanceNorm3d

$$y = \frac{x - \text{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} * \gamma + \beta \quad (7.77)$$

LayerNorm

$$y = \frac{x - \mathbb{E}[x]}{\sqrt{\text{Var}[x] + \epsilon} * \gamma + \beta} \quad (7.78)$$

LocalResponseNorm

$$b_c = a_c \left(k + \frac{\alpha}{n} \sum_{c'=\max(0, c-n/2)}^{\min(N-1, c+n/2)} a_{c'}^2 \right)^{-\beta} \quad (7.79)$$

ISRU (Inverse Square Root Unit) [BradCarlile2017]

$$f(x) = \frac{x}{\sqrt{1 + \alpha x^2}}, x \in \left(-\frac{1}{\sqrt{\alpha}}, \frac{1}{\sqrt{\alpha}} \right) \quad (7.80)$$

Inverse square root linear unit (ISRLU) [BradCarlile2017]

$$f(x) = \begin{cases} \frac{x}{\sqrt{1+\alpha x^2}} & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}, x \in \left(-\frac{1}{\sqrt{\alpha}}, \infty \right) \quad (7.81)$$

SoftPlus [glorot2011]

$$f(x) = \ln(1 + e^x), x \in (0, \infty) \quad (7.82)$$

Multi-layer long short-term memory (LSTM) RNN to an input sequence.

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \\ g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\ c_t &= f_t * c_{(t-1)} + i_t * g_t \\ h_t &= o_t * \tanh(c_t) \end{aligned} \quad (7.83)$$

where h_t is the hidden state at time t , c_t is the cell state at time t , x_t is the input at time t , $h_{(t-1)}$ is the hidden state of the layer at time $t - 1$ or the initial hidden state at time 0, and i_t , f_t , g_t , o_t are the input, forget, cell, and output gates, respectively. σ is the sigmoid function, and $*$ is the Hadamard product.

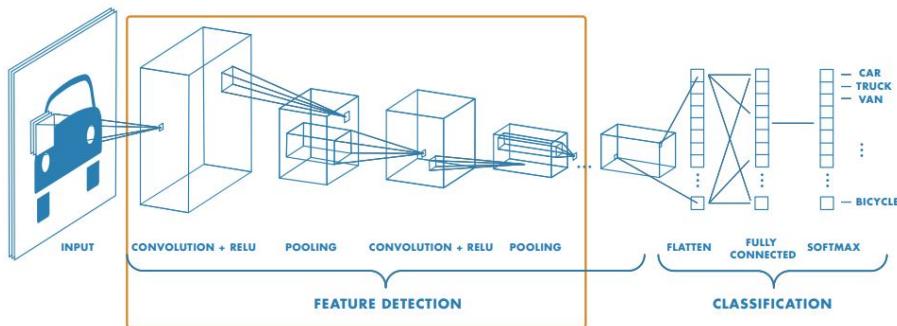


图 7-38 卷积神经网络

分类层在特征检测之后, CNN 的架构转移到分类. 倒数第二层是全连接层 (FCL), 输出 K 维度的向量, 其中 K 是网络能够预测的类数量. 此向量包含任何图像的每个类进行分类的概率. CNN 架构的最后一层使用 softmax 函数提供分类输出.

没有用于选择层数的确切公式. 最好的方法是尝试一些层, 查看工作效果如何, 或者使用预先训练好的网络.

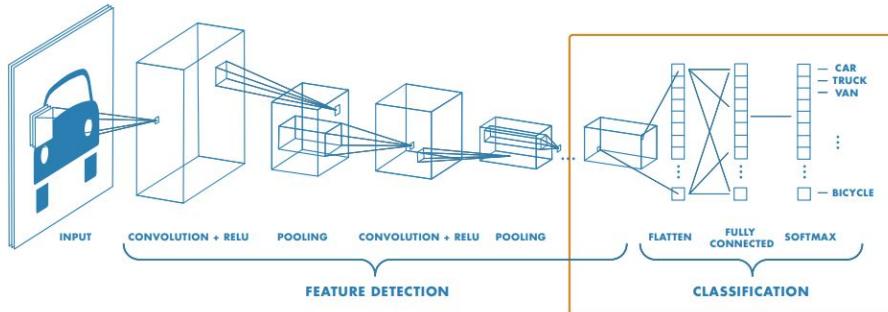


图 7-39 卷积神经网络

深度学习是机器学习的子类型. 使用机器学习, 您手动提取图像的相关特征. 使用深度学习, 您将原始图像直接馈送给深度神经网络, 该网络自动学习特征. 为了获得最佳结果, 深度学习通常需要成百上千乃至数百万张图像. 而且属于计算密集型, 需要高性能 GPU.

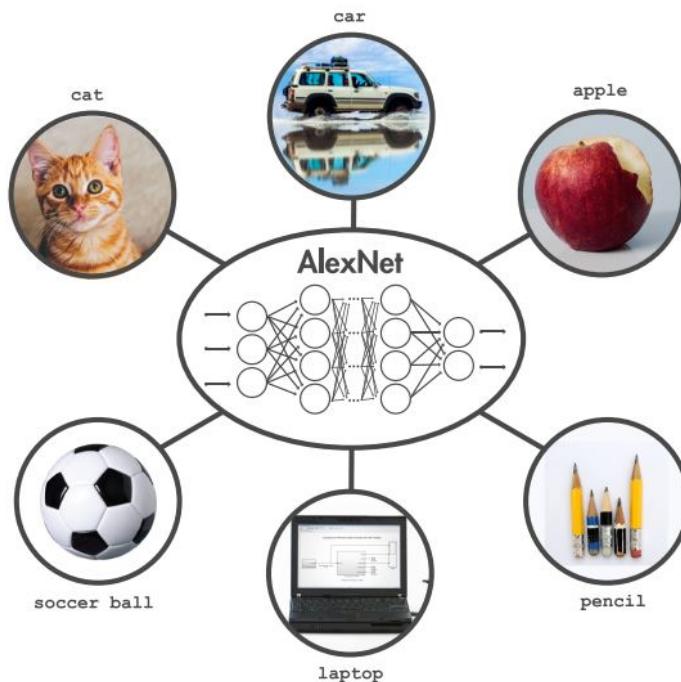


图 7-40 AlexNet 的识别对象

刚接触深度学习，快速而轻松的入门方法是使用现有网络，比如 2012 年首次发布的 AlexNet，已成为研究团体中众所周知的模型。用一百多万张图像训练好的 CNN。AlexNet 最常用于图像分类。它可将图像划分为 1000 个不同的类别，包括键盘、鼠标、铅笔和其他办公设备，以及各个品种的狗、猫、马和其他动物。

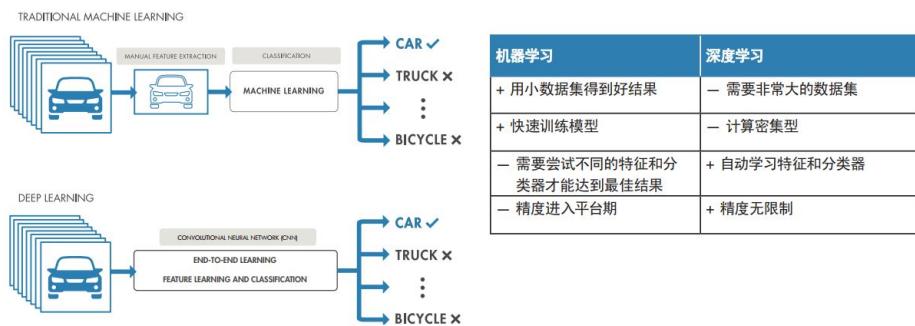


图 7-41 机器学习和深度学习

例 7.23 (AlexNet 的 Matlab 示例) 使用 AlexNet 对任何图像中的对象分类。在本例中，我们使用它对桌上安装的网络摄像头捕获的图像中的对象分类。除了 MATLAB，我们还使用下列工具：

- ▶ Deep Learning Toolbox;
- ▶ 在 MATLAB 中使用网络摄像头的支持包;
- ▶ 使用 AlexNet 的支持包.

在加载 AlexNet 后, 我们连接到网络摄像头并拍摄一张实时图像.

```
camera = webcam; % 连接到摄像头
nnet = AlexNet; % 加载神经网络
picture = camera.snapshot;
% 抓取图片
```

接下来, 我们调整图像大小为 227x227 像素, 即 AlexNet 所需的大小.

```
picture = imresize(picture,[227,227]);
% 调整图片大小
AlexNet 现在可对我们的图像分类.
label = classify(nnet, picture);
% 对图片分类
image(picture); % 显示图片
title(char(label)); % 显示标签
```

上一示例使用的是现成的网络. 我们完全不做任何修改, 因为 AlexNet 训练所用的图像类似于我们想要分类的图像.

要使用 AlexNet 对未在原有网络中训练的对象分类, 我们可以通过迁移学习重新训练. 迁移学习是将一个类型问题的知识应用于不同类型但相关的问题的学习方法.

例 7.24 本例中, 我们只是剪去网络的最后 3 层, 用我们自己的图像重新训练.

如果迁移学习不适合您的应用场景, 您可能需要从头开始训练您自己的网络. 这种方法产生最精确的结果, 但一般需要成百上千张带标签的图像和大量的计算资源. 培训深度学习模型可能需要几小时、几天或几星期, 具体取决于数据量大小以及可投入使用的处理能力. 选择计算资源是您建立工作流程时的重要考虑因素. 当前, 有三个计算选项: 基于 CPU、基于 GPU 和基于云.

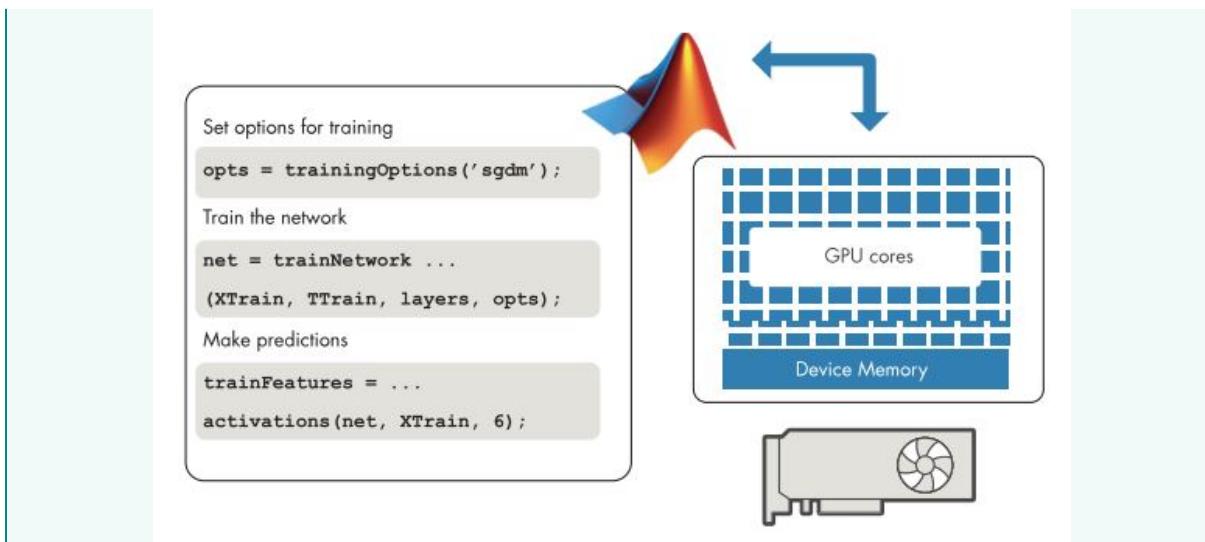


图 7-42 深度学习和机器学习

- ▶ 基于 CPU 的计算是最简单、最容易得到的选项。前面所述的示例在 CPU 上运算，但我们只对使用预先训练网络的简单示例推荐使用基于 CPU 的计算。
- ▶ 使用 GPU 可将网络训练时间从几天缩短到几小时。您可以在 MATLAB 中使用 GPU，无需任何额外编程。我们推荐 NVIDIA[®] 3.0 计算能力的 GPU。多个 GPU 能更加快处理速度。
- ▶ 基于云的 GPU 计算意味着您不必亲自购买和设置硬件。您为了使用本地 GPU 而编写的 MATLAB 代码只需进行一些设置变更，便可扩展为使用云资源。

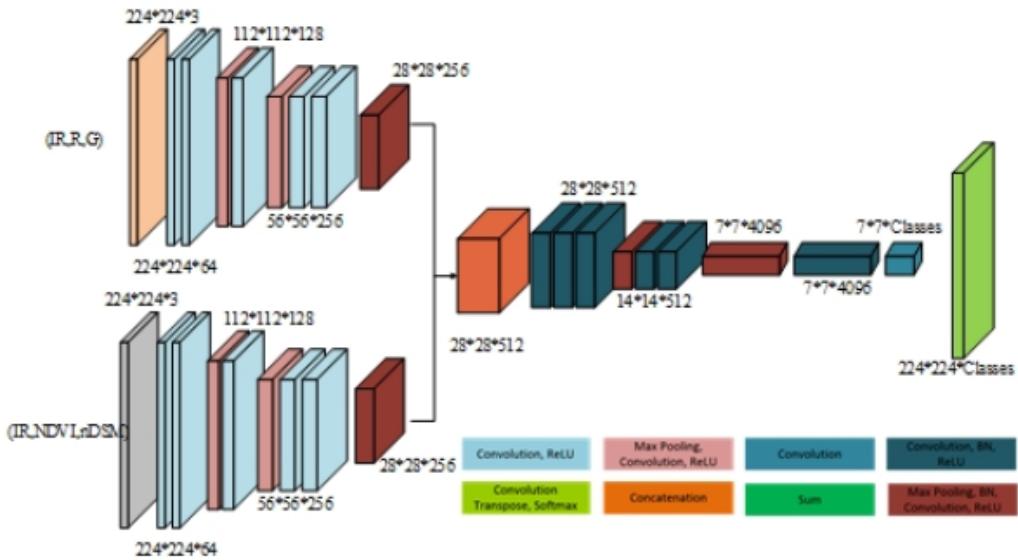


图 7-43 Fusion of features after layer three of a CNN. The features from two streams are passed through max pooling, convolution, batch normalization and relu layers. The two outputs are then concatenated and form the input for the fourth layer [PiramanayagamSaber2018-9591]

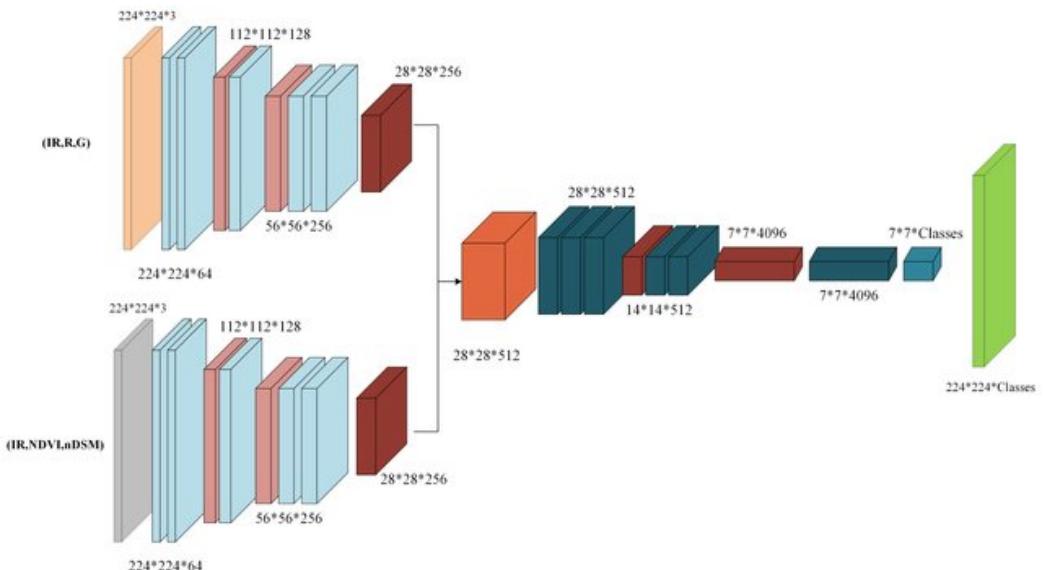


图 7-44 Fusion of features after layer three of a CNN. The features from two streams are passed through max pooling, convolution, batch normalization and ReLU layers. The two outputs are then concatenated and form the input for the fourth layer [PiramanayagamSaber2018-9591].

Berkeley 团队提出 Fully Convolutional Networks (FCN) 方法用于图像语义分割, 将图像级别的分类扩展到像素级别的分类 (图 7-45), 获得 CVPR2015 的最佳论文奖

[Long2015-9593]. FCN 将传统卷积网络后面的全连接层换成了卷积层, 这样网络输出不再是类别而是 heatmap; 同时为了解决因为卷积和池化对图像尺寸的影响, 提出使用上采样的方式恢复.

- FCN 不含全连接层 (fc) 的全卷积 (fully conv) 网络. 可适应任意尺寸输入.
- FCN 增大数据尺寸的反卷积 (deconv) 层. 能够输出精细的结果.
- FCN 结合不同深度层结果的跳级 (skip) 结构. 同时确保鲁棒性和精确性.

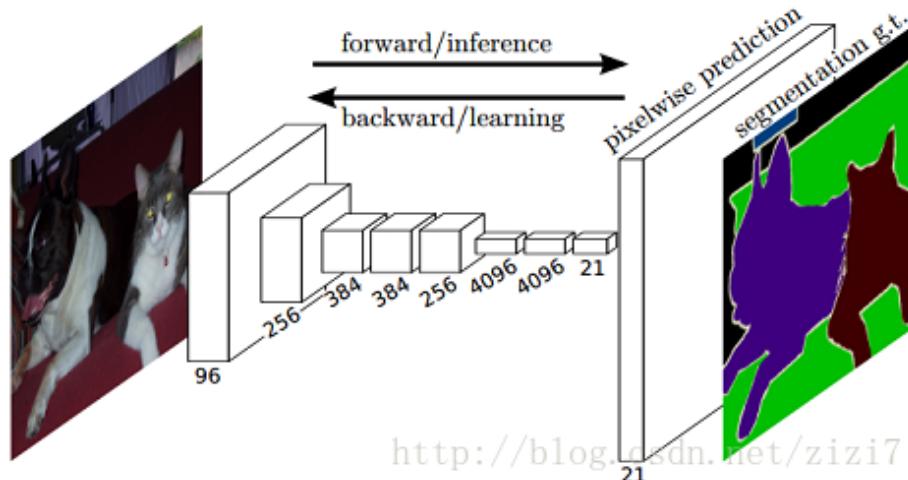


图 7-45 完全 CNN 结构 (FCN-8)[[http://blog.sdn.net/zizi7](#)]

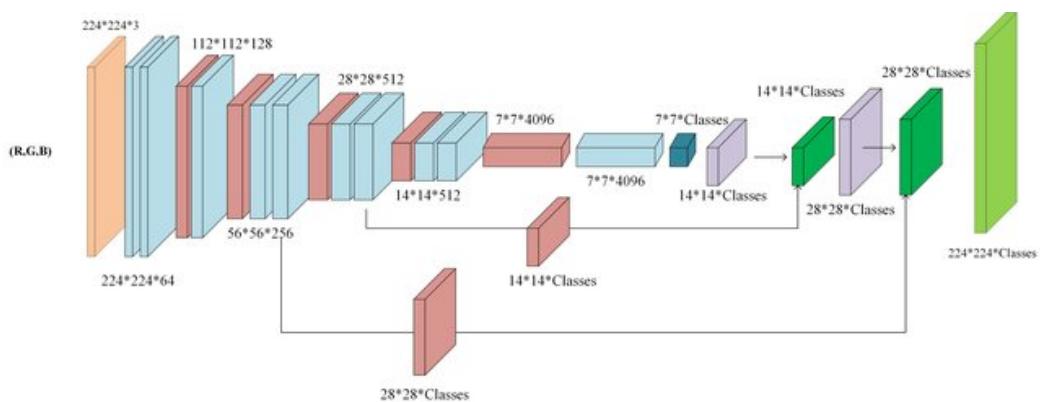


图 7-46 完全 CNN 结构 (FCN-8)[[PiramanayagamSaber2018-9591](#)]

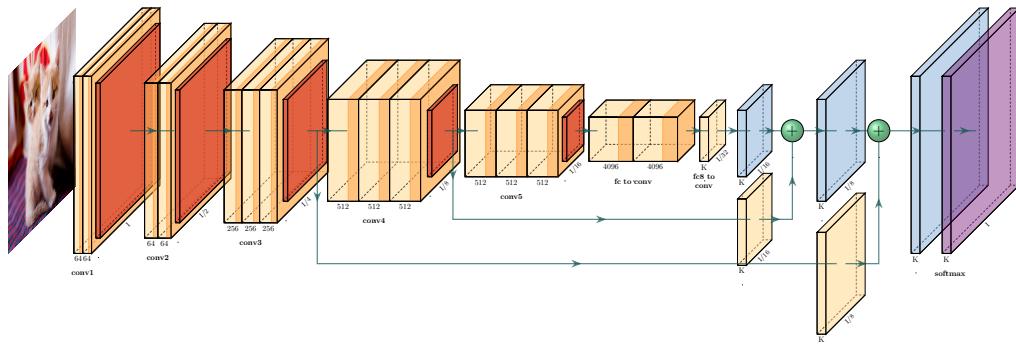


图 7-47 FCN-8

通常 CNN 网络在卷积层之后会接上若干个全连接层, 将卷积层产生的特征图 (feature map) 映射成一个固定长度的特征向量。以 AlexNet 为代表的经典 CNN 结构适合于图像级的分类和回归任务, 因为它们最后都期望得到整个输入图像的一个数值描述 (概率), 比如 AlexNet 的 ImageNet 模型输出一个 1000 维的向量表示输入图像属于每一类的概率 (softmax 归一化)。FCN 对图像进行像素级的分类, 从而解决了语义级别的图像分割 (语义分割) 问题。与经典的 CNN 在卷积层之后使用全连接层得到固定长度的特征向量进行分类 (全连接层 + softmax 输出) 不同, FCN 可以接受任意尺寸的输入图像, 采用反卷积层对最后一个卷积层的特征映射进行上采样, 使它恢复到输入图像相同的尺寸, 从而可以对每个像素都产生了一个预测, 同时保留了原始输入图像中的空间信息, 最后在上采样的特征图上进行逐像素分类。

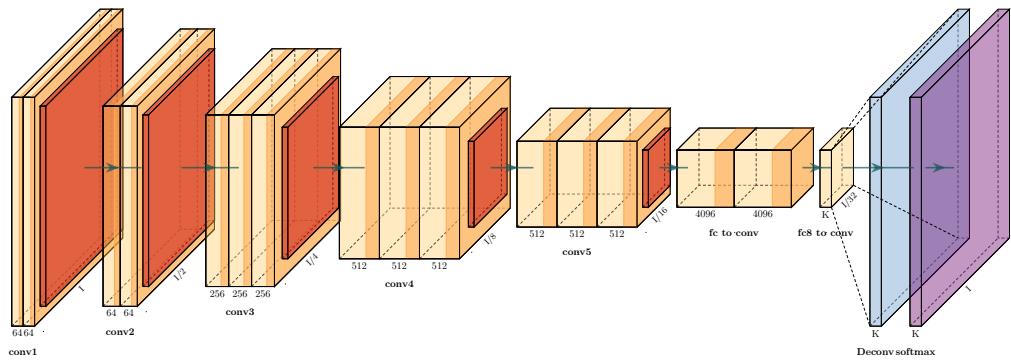


图 7-48 FCN-32

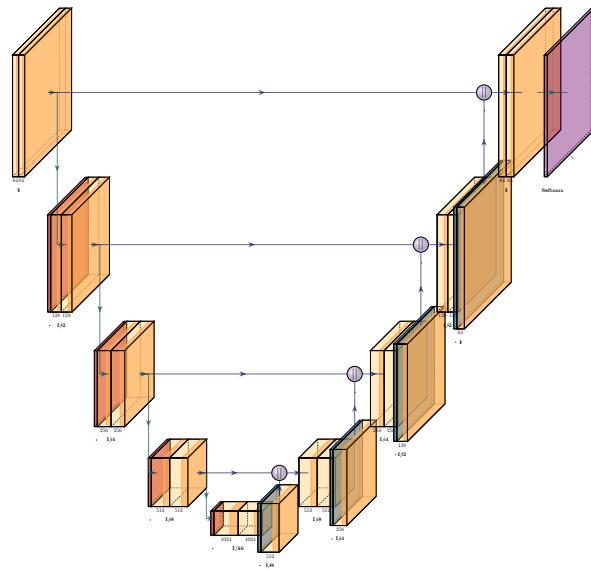


图 7-49 Unet ushape

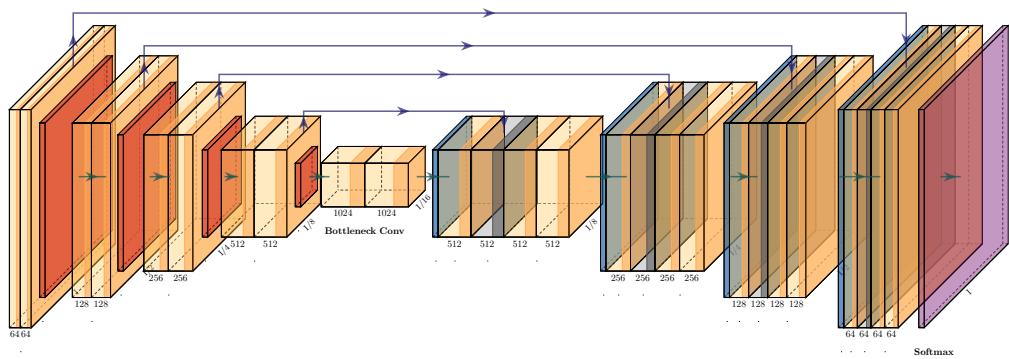


图 7-50 Unet

VGGNet 是牛津大学计算机视觉组（Visual Geometry Group）和 Google DeepMind 公司的研究员一起研发的深度卷积神经网络⁷⁻⁵¹。VGGNet 探索了卷积神经网络的深度与其性能之间的关系，通过反复堆叠 3×3 的小型卷积核和 2×2 的最大池化层，VGGNet 成功地构筑了 16~19 层深的卷积神经网络。VGGNet 相比之前最前沿的网络结构，错误率大幅下降，VGGNet 相关的论文全部使用了 3×3 的小型卷积核和 2×2 的最大池化核，通过不断加深网络结构来提升性能。

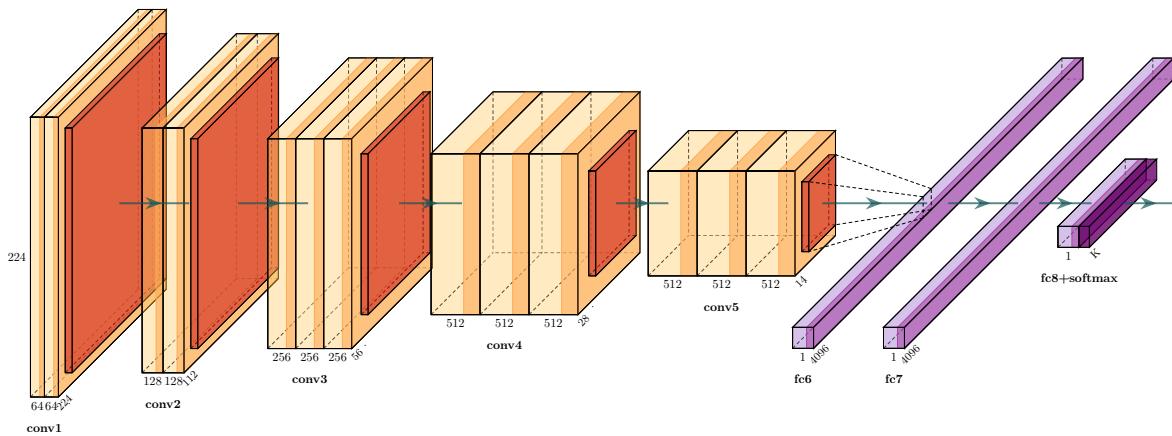


图 7-51 vgg-16

INPUT: [224x224x3] memory: 224*224*3=150K params: 0 (not counting biases)

CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728

CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864

POOL2: [112x112x64] memory: 112*112*64=800K params: 0

CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728

CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456

POOL2: [56x56x128] memory: 56*56*128=400K params: 0

CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912

CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824

CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824

POOL2: [28x28x256] memory: 28*28*256=200K params: 0

CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648

CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296

CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296

POOL2: [14x14x512] memory: 14*14*512=100K params: 0

CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296

CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296

CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296

POOL2: [7x7x512] memory: 7*7*512=25K params: 0

FC: [1x1x4096] memory: 4096 params: 77*512*4096 = 102,760,448

FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216

FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ≈ 96MB / image (only forward! ~*2 for bwd)

TOTAL params: 138M parameters

http://blog.csdn.net/qq_24293177

VGG16

Common names

Stage	Layer
1	fc8
2	fc7
3	fc6
4	pool
5	conv5-3
6	conv5-2
7	conv5-1
8	conv4-3
9	conv4-2
10	conv4-1
11	pool
12	conv3-2
13	conv3-1
14	pool
15	conv2-2
16	conv2-1
17	pool
18	conv1-2
19	conv1-1
20	Input

图 7-52 vgg-16 各层的参数、内存

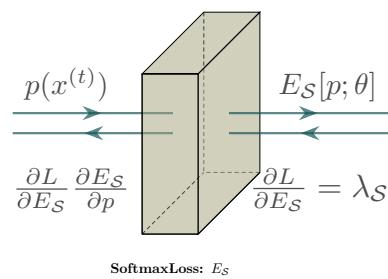


图 7-53 Softmax Loss

7.8.2 CNN 的应用



图 7-54 图像分类 (AlexNet 在 ILSVRC-2012 测试图像上预测的 top-5 标签)

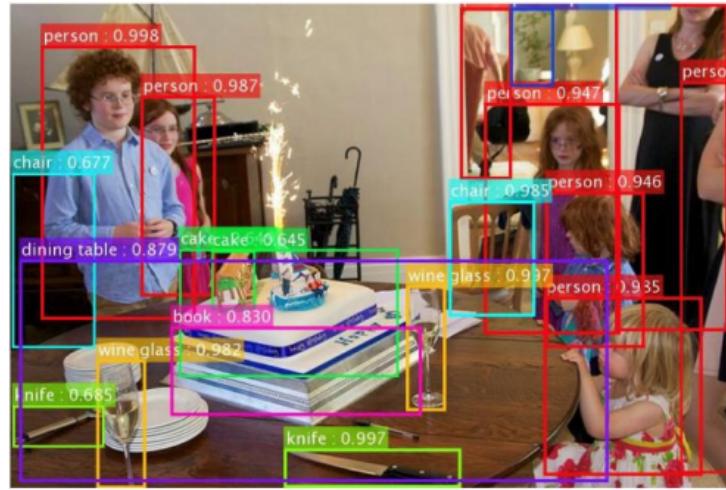


图 7-55 目标检测 (Faster R-CNN 在 COCO 数据集的检测结果)

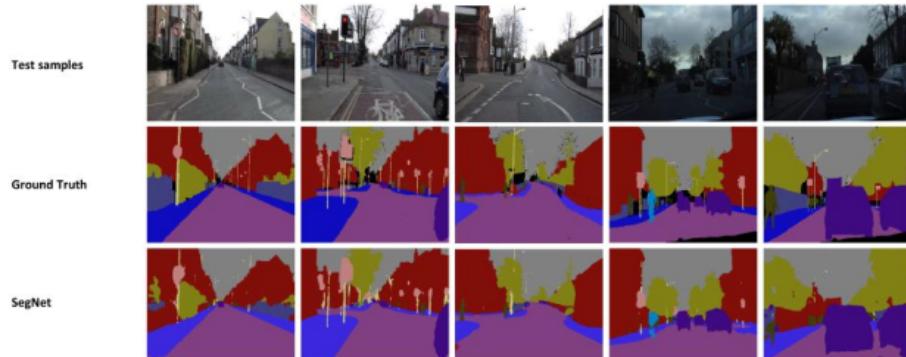


图 7-56 图像分割 (SegNet 模型分割结果)

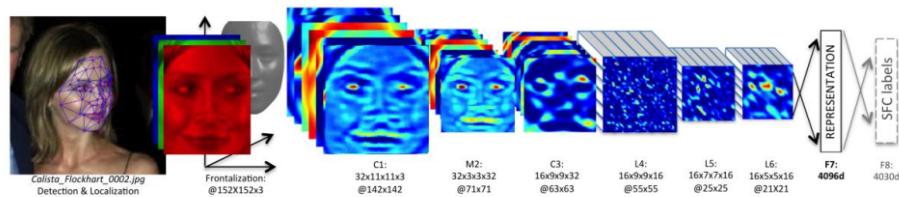


图 7-57 人脸识别 (DeepFace 模型)

7.8.3 卷积编译码网络学习语义图形实现水稻自动除草

该方法将语义图形用于数据注释并结合高级编译码网络, 实现稻田中自动作物行和杂草检测([Adhikari-2019], 2019.10), 介绍如下:

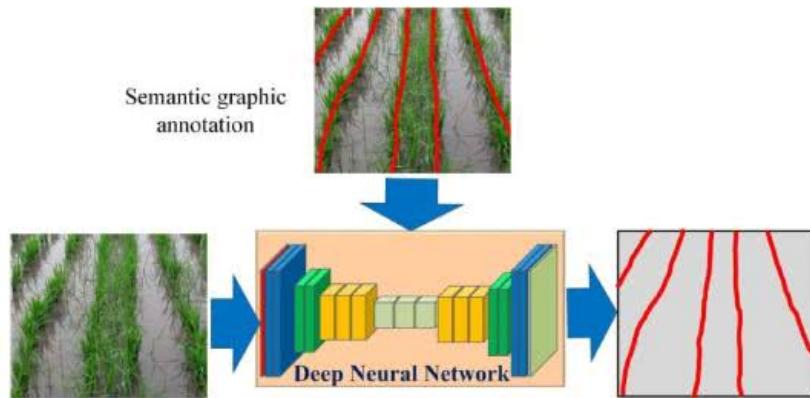


图 7-58 训练深度神经网络的方法使用语义图形来学习作物行的概念

杂草是农场侵略者,与作物争夺营养及其他资源,造成作物产量降低。化学品的大量使用能够控制杂草,但会对人类健康和环境造成意想不到的后果。思路就是找到一种新型神经网络训练方法,该方法将语义图形用于数据注释并结合高级编译码网络,实现稻田中自动作物行和杂草检测。利用检测到的作物行指导自动除草机器人行间除草,而杂草检测则使自动行内除草成为可能。提出的训练深度神经网络的方法使用语义图形来学习作物行的概念。

稻谷数据集的语义图形。

所提出的数据注释方法语义图形直观,并且注释所需目标的工作量很少。“extended skip network”是一种改进的深卷积编译码神经网络,用于语义图形的高效学习。研究结果表明,在水稻行和杂草检测中,相对于基线网络,联合平均交叉数(mIoU)分别增加了6.29%和6.14%。并且与当前流行的基于边界框的对象检测方法相比,该方法也使得mIoU增加了3.56%,具有更高的召回率。

在稻田数据集上使用拟议的扩展跳过网络学习语义线的定性结果。

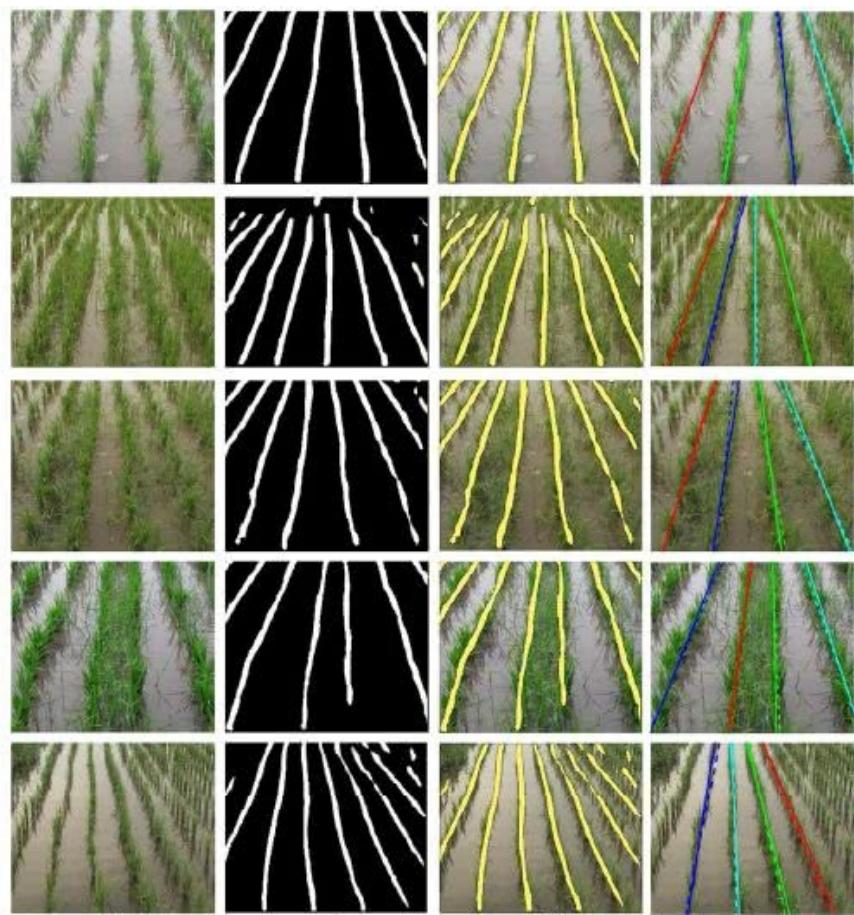


图 7-59 稻田数据集上使用拟议的扩展跳过网络学习语义线的定性结果

在稻谷数据集上使用拟议的卷积编码器/解码器网络学习语义图形的定性结果。

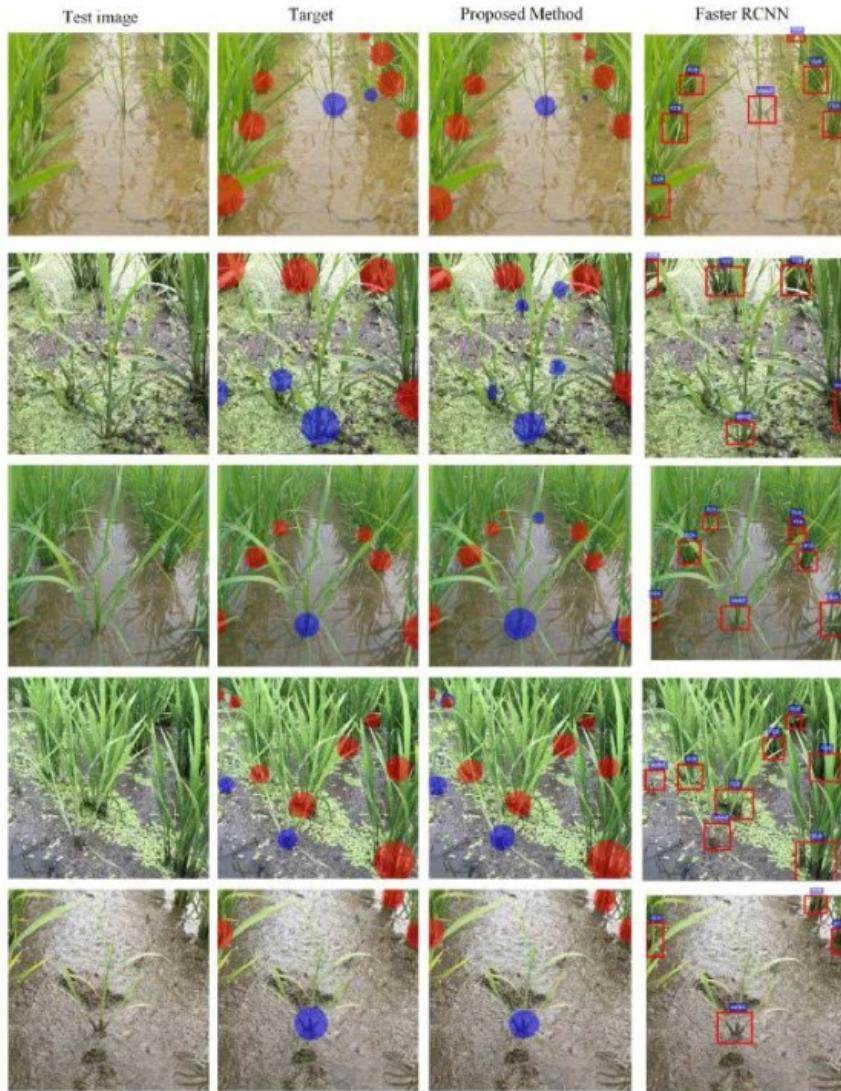


图 7-60 稻谷数据集上使用拟议的卷积编码器/解码器网络学习语义图形的定性结果

7.8.4 图卷积神经网络

阿里如何把图卷积网络用于闲鱼的垃圾评论过滤。目前该系统已经部署到了闲鱼使用环境中, 它每天能处理百万级的闲鱼评论, 并在其他强有力深度学习模型基础上, 额外筛选出一千多条非常隐秘的垃圾评论。基于图卷积的垃圾信息筛选是一种非常通用的思想, 它的应用范围远不止垃圾评论过滤, 淘宝信息的知识产权保护、淘宝商品管控和用户恶意评价等方面都可以采用。GAS 的整体流程是 7-61, 模型会从左侧图抽取出表示商品、用户和评论的信息, 从右侧抽取出类似评论表示的意义。最后结合这些信息进行分类, 模型就能很好地识别垃圾信息了。

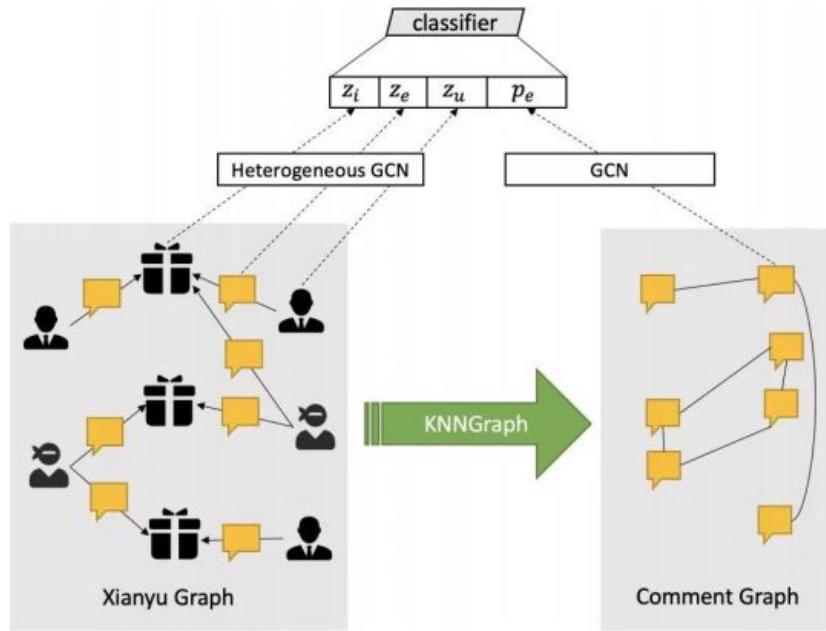


图 7-61 GCN-based Anti-Spam System(GAS) 的垃圾评论过滤系统

图卷积的核心思想是希望利用近邻节点的信息进行聚合而生成当前节点的新表征,这样的节点表示可以进一步用于下游任务. 如果我们直接从核心表达式出发, 跳过推导过程, 其实能更容易地理解. 如下所示为两层图卷积网络之间的传播方法, 它看起来只不过比常规的神经网络多了 \tilde{D} 与 \tilde{A} 这几项.

$$H^{(l+1)} = \sigma \left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right). \quad (7.84)$$

如果我们的图有 n 个节点, 那么节点与节点之间的关系可以用 $n \times n$ 的邻接矩阵表示, 它再加上由节点特征向量组成的矩阵 H 就是图卷积的输入. 在上式中, \tilde{A} 以及 \tilde{D} 就是由邻接矩阵算出来的东西, 它对于同一张图是不变的, 因此可以预先计算好. 现在, 剩下的 $H \times W$ 就是输入 Embedding H 经过一层全连接层了, 以这样方式进行层级传播的卷积网络就是图卷积, 我们可以将传播理解为每个节点拿到邻居节点信息, 并聚合到自身嵌入向量上. 如图7-62所示, 图卷积网络的输入是表示节点及边的特征向量, 经过一系列隐藏层的变换, 可以计算出每个节点的深度表征. 这样的 Z 再来做预测或生成就会非常有效. 直观而言, 图卷积将图片的 RGB 像素值替换成节点特征, 并且通过边的关系引入了邻居的概念, 完成卷积运算.

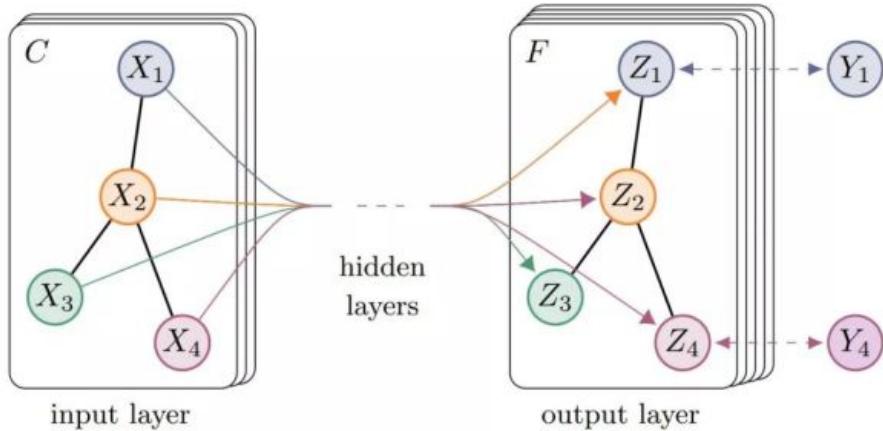


图 7-62 图卷积网络的输入输出结构

异构图上的图卷积

阿里 GAS 一共有两种输入图, 它们分别用来表示局部信息与全局信息. 首先我们看看异构图, 一般只要边的种类加上节点的种类大于 2, 我们就可以称之为异构图. 如7-63所示闲鱼图为一个标准的异构图, 目前图卷积网络大部分都关注更简单的同构图, 闲鱼图这种异构图很难处理. 从图7-63我们可以看到, 闲鱼 Graph 有商品 I 和用户 U 这两种节点, 它们的边为评论 E . 如上, $e2, e4$ 和 $e5$ 都是垃圾评论, 它们都来自于同一用户. 利用图来判断垃圾评论, 能利用更多的额外信息, 准确率也会比纯文本好得多. 现在回到图卷积, 一般图卷积的层级可以分为聚合 (aggregation) 与结合 (combination) 两大操作. 其中 AGG 会聚合邻近节点的嵌入向量, 例如最大池化或基于注意力权重的加权和等. COMBINE 操作会结合自身的嵌入向量与前面聚合的嵌入向量, 很多 GCN 方法将 COMBINE 操作放到了 AGG 里面. 在阿里的 GAS 中, 研究者使用拼接的方式将信息聚合到边上. 比如说如果 GAS 需要将信息聚合到不同的边 (即评论) 上, 那么比较核心的表达式可以写为:

$$h_e^l = \sigma \left(W_E^l \cdot AGG_E^l \left(h_e^{l-1}, h_{U(e)}^{l-1}, h_{I(e)}^{l-1} \right) \right), \quad (7.85)$$

其中

$$AGG_E^l \left(h_e^{l-1}, h_{U(e)}^{l-1}, h_{I(e)}^{l-1} \right) = \text{concat} \left(h_e^{l-1}, h_{U(e)}^{l-1}, h_{I(e)}^{l-1} \right),$$

其中 h^l 表示第 l 层边的隐藏向量, 它需要聚合 $l-1$ 层自身的特征向量以及与它相连的两个节点向量, 聚合的方法是拼接三个向量. W^l 表示该神经网络层所需要学习的权重, σ 表示激活函数. 看上去它其实和一般的卷积网络并没有什么差别, 只不过输入都是图的各种信息, 这样也能基于局部上下文判断该评论是否是垃圾评论.

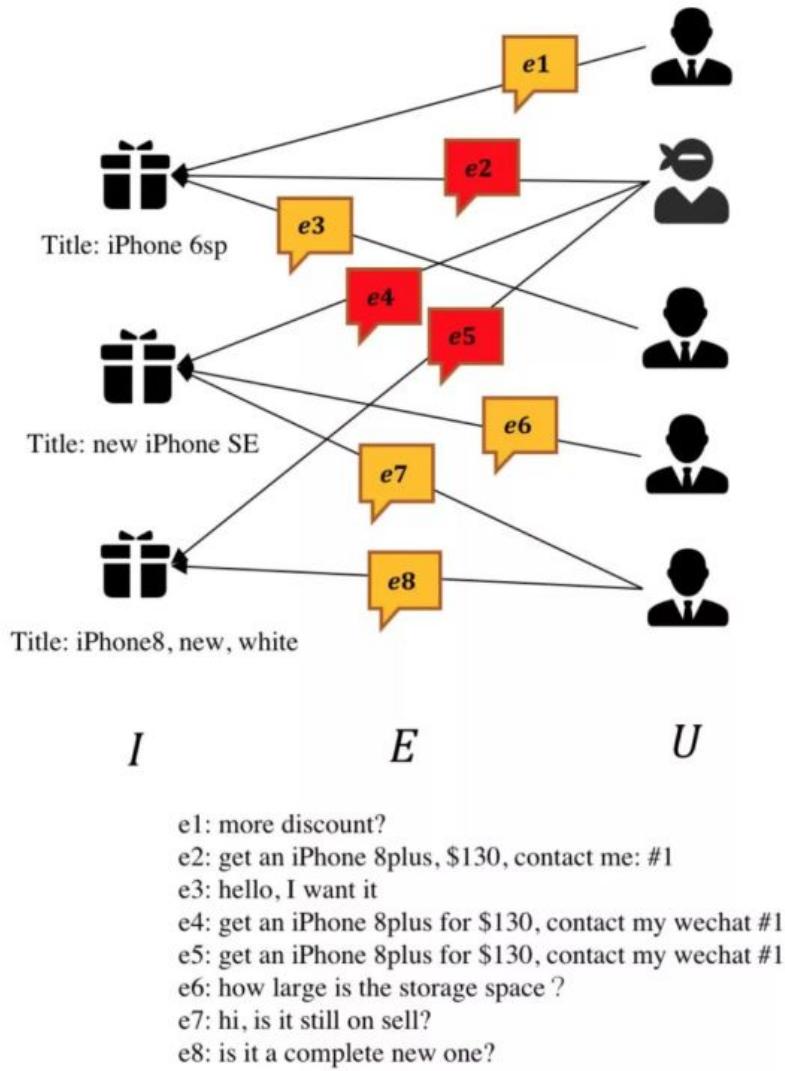


图 7-63 图卷积异构网络---闲鱼 Graph

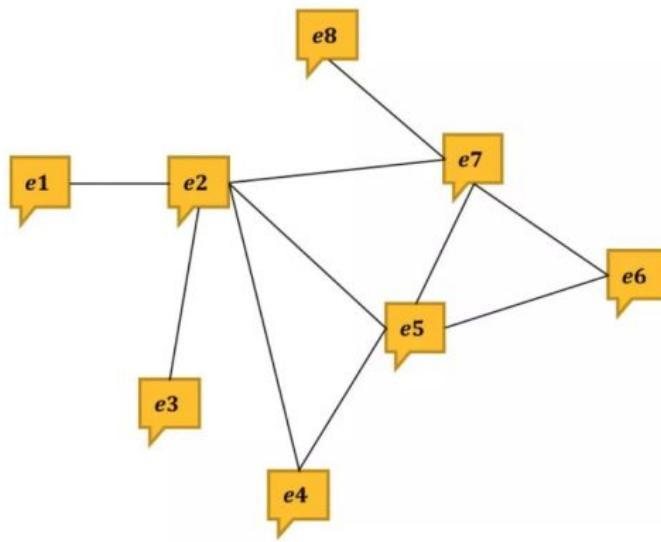
当然上面只是展示了边的聚合案例,其它节点的 AGG 操作和 COMBINE 运算在原论文中都有详细的介绍. 此外,如果从异构图卷积网络的输入与输出考虑,对于单个用户节点,输入就是邻近商品节点以及邻近评价边的特征. 例如一个用户评论了 10 件商品,那么每一个商品向量拼接上对应评论向量,这 10 个特征向量就可以作为输入,后续图卷积就会对它们进行基于注意力机制的聚合等一系列操作.

同构图上的图卷积

对于闲鱼 Graph 这种大型图, 我们能处理邻近节点这些局部信息, 但与此同时还应该能处理全局信息, 这样才能有效地减轻用户的对抗行为. 为此, 模型应该站在所有评论的角度, 看看与当前相似的评论都是什么样, 它们是不是垃圾评论. 阿里的研究者基于闲

鱼 Graph 构建了一种新的 Comment Graph, 它是一种同构图, 每一个节点为评论内容, 节点之间的边为两条评论之间的相似性. 因为相似的评论距离非常近, 因此模型可以考虑与当前评论相近的评论, 从而更好地判断当前评论是不是垃圾评论.

图7-64给出了一小部分 Comment Graph, 如果说局部模型无法根据「add v」判断出意思是加微信, 那么放在 Comment Graph 中就非常明确了, 它与类似的说法都应该被判断为垃圾评论.



- e1: for more certified products, please add #2
- e2: for more styles, please add #3
- e3: need more styles and discounts, please add v #4
- e4: more styles please add wecha #5
- e5: people who like it please add my wechat,
different styles
- e6: if you like it, leave me a message, add v
- e7: I have others styles, if you like it please add: #6
- e8: anything you like? please add #5

图 7-64 基于闲鱼 Graph 构建的同构 Comment Graph

简单而言, Comment Graph 的构建主要分为四个步骤: 移除所有重复的评论; 通过词嵌入模型为评论生成嵌入向量; 利用 KNN Graph 算法获得相似的评论对; 移除同一用户提出的评论对, 或者同一卖家提出的评论, 因为之前的闲鱼 Graph 已经考虑了这些信息. 构建了 Comment Graph, 再用图卷积就能抽取节点信息了, 因为每一个节点输出向量都聚合了周围节点的信息, 它就能代表全局上这一些相似评论的意义. 最后, 结合异构图卷积与同构图卷积的结果, 再来做个简单的分类就很合理了.

7.8.5 循环神经网络 (Recurrent Neural Networks, RNN)

针对对象: 序列数据. 例如文本是字母和词汇的序列; 语音是音节的序列; 视频是图像的序列; 气象观测数据, 股票交易数据等也都是序列数据. RNN 专门解决时间序列问题, 用来提取时间序列信息, 一般放在 CNN 特征提取层之后.

核心思想: 样本间存在顺序关系, 每个样本和它之前的样本存在关联. 通过神经网络在时序上的展开, 我们能够找到样本之间的序列相关性.

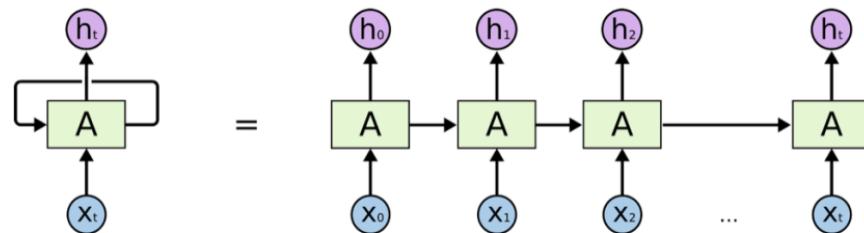


图 7-65 RNN 网络结构按时间展开

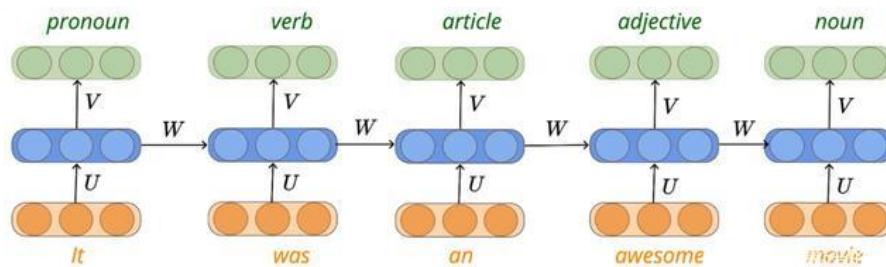


图 7-66 RNN 网络结构

RNN 主要用于序列分类

- 情感分类和视频分类序列标记.
- 语音标记和命名实体识别序列生成.
- 机器翻译和音译.

在循环神经网络中的每个时间步处, 旧信息都会随着当前输入而发生变化. 对于较长的句子, 我们可以想象, 在 t 时间步之后, 存储在 $t - k$ 时间步 ($k \ll t$) 的信息会经历一个逐渐转变的过程. 在反向传播过程中, 信息必须流经较长的时间步才能更新网络参数, 以最大程度地减少网络的损失.

考虑一个场景, 在该场景中, 我们需要计算时间步 L 处的网络损失. 假定损失是由于在时间步 S_1 对隐藏表示的错误计算而产生的. S 处的错误是由于向量 W 的参数不正确

造成的。此信息必须反向传播到 W , 以便向量可以校正其参数。

$$a_j = Ux_j + Ws_{j-1} + b \quad (7.86)$$

$$s_j = \sigma(a_j) \quad (7.87)$$

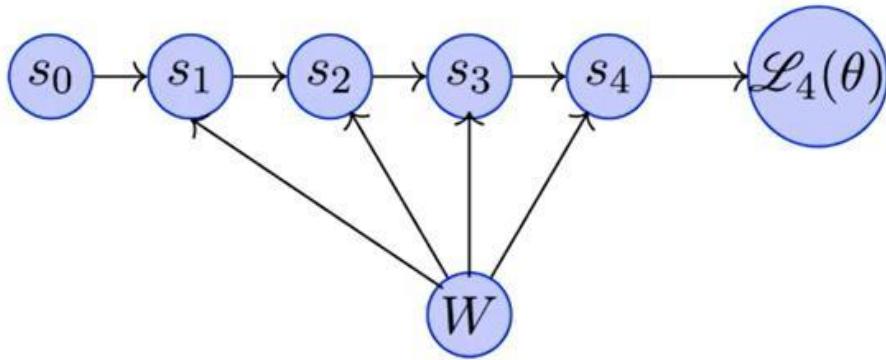


图 7-67 RNN 权重作用示意图

为了将信息传播回向量 W , 我们需要用到链式法则的概念。简而言之, 链式法则归结为特定时间步的所有隐藏表示的偏导数的乘积。

$$\frac{\partial \mathcal{L}_t(\theta)}{\partial W} = \frac{\partial \mathcal{L}_t(\theta)}{\partial s_t} \sum_{k=1}^t \frac{\partial s_t}{\partial s_k} \frac{\partial s_k}{\partial W} \quad (7.88)$$

$$\frac{\partial s_t}{\partial s_k} = \frac{\partial s_t}{\partial s_{t-1}} \frac{\partial s_{t-1}}{\partial s_{t-2}} \frac{\partial s_{t-2}}{\partial s_{t-3}} \dots \frac{\partial s_{k+1}}{\partial s_k} \quad (7.89)$$

$$\frac{\partial s_t}{\partial s_k} = \prod_{j=k+1}^t \frac{\partial s_j}{\partial s_{j-1}} \quad (7.90)$$

如果我们有超过 100 个或者更长的序列的隐藏表示, 那么我们必须计算这些表示的乘积来进行反向传播。假设其中一个偏导数值很大, 那么整个梯度值就会非常大, 从而导致梯度爆炸问题。如果其中一个偏导数是一个很小的值, 那么整个梯度就会变得更小或消失, 使得网络难以训练, 这就是梯度消失问题。由于 RNN 具有有限的状态大小, 而不是从所有的时间步长中提取信息并计算隐藏状态表示。在从不同的时间步长中提取信息时, 我们需要遵循选择性读 (read)、写 (write) 和遗忘 (forget) 策略。

例 7.25 RNN 示例

让我们以使用 RNN 进行情感分析为例, 看看选择性 read, write 和 forget 策略的工作原理。评论: The first half of the movie is dry but the second half really picked up the pace. The lead actor delivered an amazing performance.

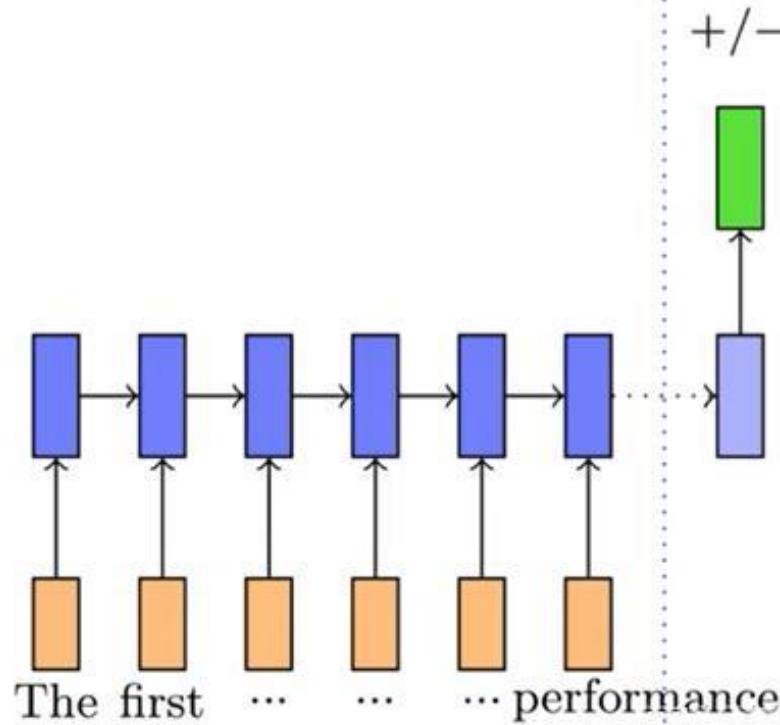


图 7-68 RNN 权重作用示意图

评论始于负面情绪，后来变成正面。过程如下：

我们首先遗忘由 stop words(a, the, is 等)添加的信息。有选择地阅读由情感表达的单词所添加的信息(amazing, awesome 等)。有选择地将隐藏状态表示信息从当前单词写入新的隐藏状态。使用选择性的读、写和遗忘策略，我们可以控制信息流，这样网络就不会出现短期记忆的问题，也可以确保有限大小的状态向量得到有效的使用。

长期短期记忆—LSTM

引入 LSTM 是为了克服 vanilla RNN 存在的短期记忆和梯度消失等问题。在 LSTM 中，通过使用门来调节信息流，我们可以有选择地读写和遗忘信息。

- RNN 或 LSTM 中的写

在 vanilla RNN 版本中，隐藏表示(s_t)为：是根据前一个时间步长的输出计算得出的隐藏表示(s)和当前输入(x)以及偏差(b)。

$$s_t = \sigma(Ux_t + Ws_{t-1} + b), \quad (7.91)$$

其中 s_{t-1} 为前一个时间步的隐藏表示， x_t 当前输入， b 为偏差

在这里, 我们获取 s 的所有值并计算当前时间 (s_t) 的隐藏状态表示.

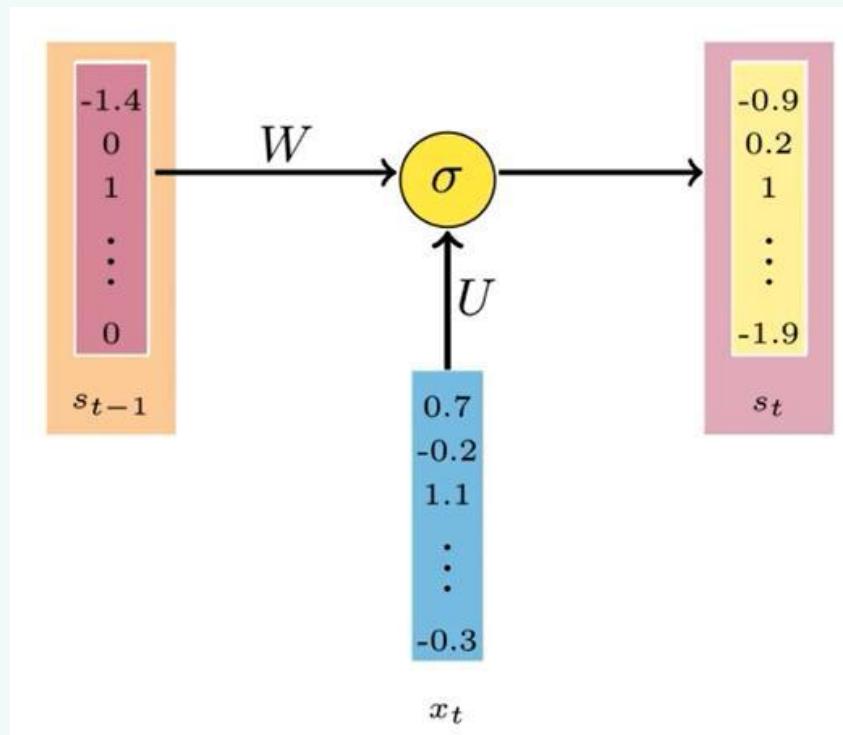


图 7-69 RNN 节点计算

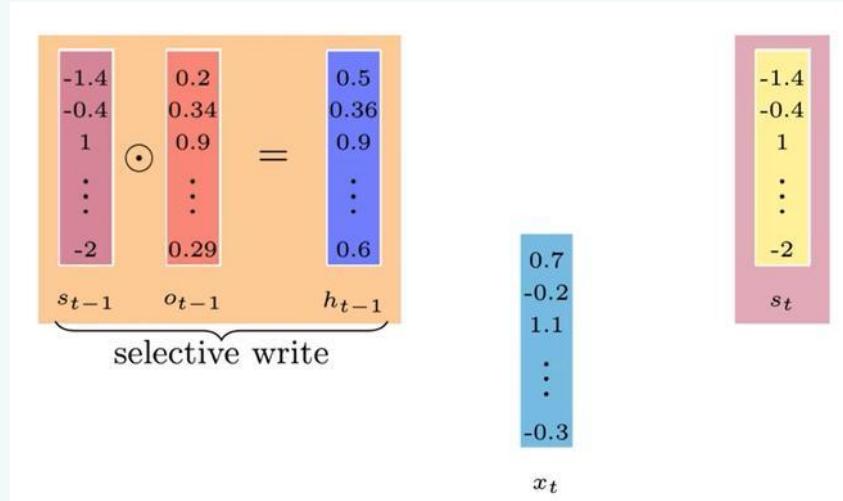


图 7-70 平面 RNN 节点计算

- RNN 或 LSTM 中的选择性写

在“选择性写”中，不是将所有信息写入 s_{t-1} 中以计算隐藏表示 (s_t)。我们只将有关 s_{t-1} 的一些信息传递给下一个状态来计算 s_t 。一种方法是在 0-1 之间分配一个值，该值确定将当前状态信息的哪一部分传递给下一个隐藏状态。

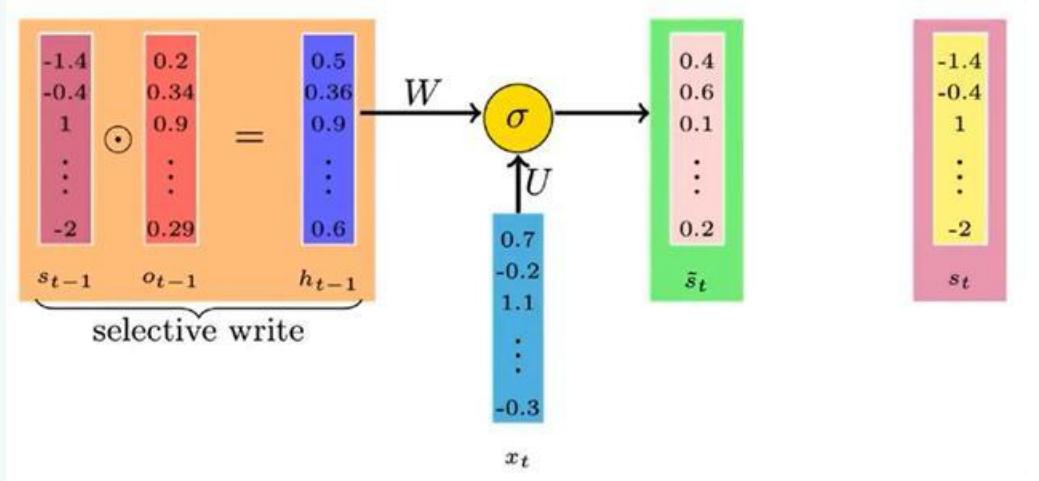


图 7-71 RNN 节点计算

我们进行“选择性写”的方式是，将 s_{t-1} 的每个元素乘以一个介于 0 到 1 之间的值，以计算出一个新的向量 h_{t-1} 。我们将使用这个新向量来计算隐藏表示 s_t 。就像我们使用基于梯度下降优化的参数学习来学习其他参数（例如 U 和 W ）一样，我们将从数据中学习 o_{t-1} 的数学方程式如下：

$$o_{t-1} = \sigma(U_o x_{t-1} + W_o h_{t-2} + b_o) \quad (7.92)$$

$$h_{t-1} = s_{t-1} \odot o_{t-1} \quad (7.93)$$

一旦我们从数据中获知 o_{t-1} ，就将其与 s_{t-1} 相乘以获得新的向量 h_{t-1} 。由于 o_{t-1} 正在控制哪些信息将进入下一个隐藏状态，因此称为输出门。

• RNN 或 LSTM 中的选择性读

在计算了新向量 h_{t-1} 之后，我们将计算一个中间隐藏状态向量 t （标记为绿色）。在本节中，我们将讨论如何实现选择性读以获得最终的隐藏状态 s_t 。数学公式 \tilde{s}_t 给出如下：

$$\tilde{s}_t = \sigma(Ux_t + Wh_{t-1} + b) \quad (7.94)$$

t 从先前状态 h_{t-1} 和当前输入 x_t 捕获所有信息。但是，我们可能不希望使用所有新信息，

而只是在构造新的单元结构之前有选择地从中读取信息, 即我们只想从 t 中读取一些信息来计算 s_t .

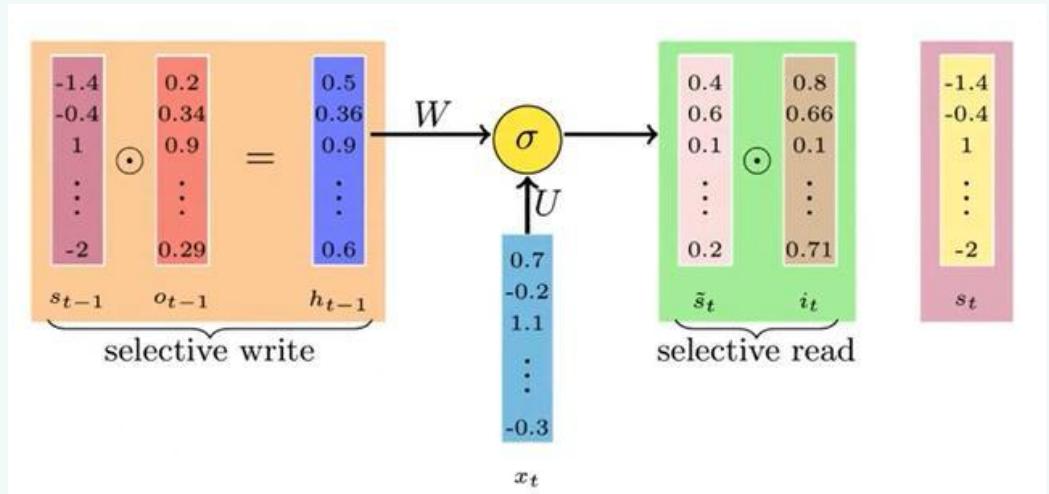


图 7-72 RNN 节点计算

就像我们的输出门一样, 在这里, 我们将 t 的每个元素乘以一个新的向量 i_t , 该向量的值在 0-1 之间. 由于向量 i_t 正在控制从当前输入中流入的信息, 因此称为输入门.

i_t 的数学公式如下:

$$\text{Input gate: } i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$

$$\text{Selectively Read: } i_t \odot \tilde{s}_t$$

在输入门中, 我们将前一时间步的隐藏状态信息 h_{t-1} 和当前输入 x_t 连同偏差传递给 sigmoid 函数. 计算的输出将在 0-1 之间, 它将决定从当前输入和上一时间步隐藏状态流入哪些信息. 0 表示不重要, 1 表示重要.

回顾到目前为止所学的知识, 我们拥有先前的隐藏状态 s_{t-1} , 我们的目标是使用选择性取、写和遗忘策略来计算当前状态 s_t :

$$\text{Previous state: } s_{t-1}$$

$$\text{Output gate: } o_{t-1} = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$

$$\text{Selectively Write: } h_{t-1} = o_{t-1} \odot \sigma(s_{t-1})$$

$$\text{Current (temporary) state: } \tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$$

$$\text{Input gate: } i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$

$$\text{Selectively Read: } i_t \odot \tilde{s}_t$$

• RNN 或 LSTM 中的选择性遗忘

本节将讨论如何结合 s_{t-1} 和来计算当前状态向量 s_t . 遗忘门决定从隐藏向量中保留

或丢弃的信息部分.

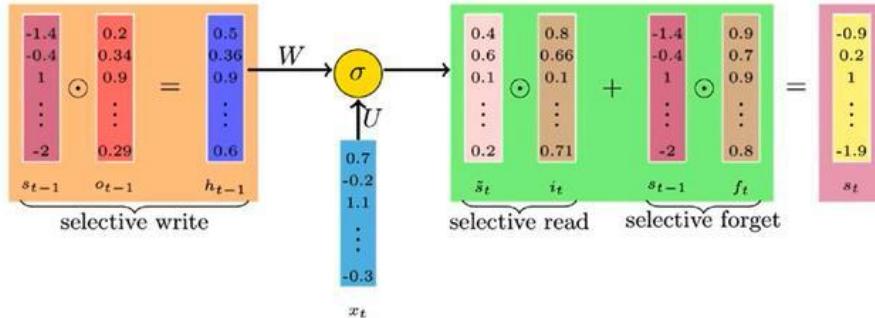


图 7-73 RNN 节点计算

遗忘门 $f(t)$ 的数学方程式如下:

$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f). \quad (7.95)$$

在“遗忘门”中, 我们将前一时间步的隐藏状态信息 h_{t-1} 和当前输入 x_t 连同偏差传递给 sigmoid 函数. 计算结果将在 0-1 之间, 它将决定要保留或丢弃的信息. 如果该值接近 0, 则表示丢弃; 如果该值接近 1, 则表示保留.

通过组合遗忘门和输入门, 我们可以计算当前的隐藏状态信息.

$$s_t = \tilde{s}_t \odot i_t + s_{t-1} \odot f_t. \quad (7.96)$$

最终如下所示:

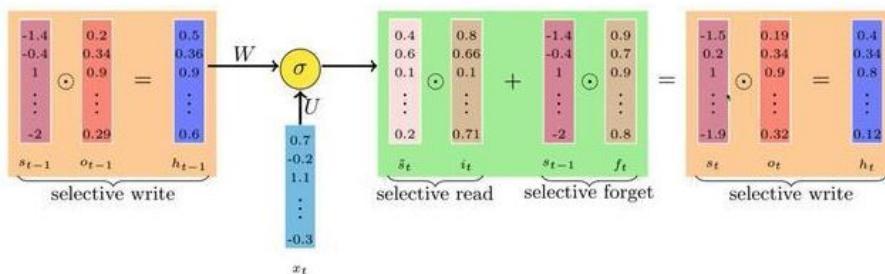


图 7-74 RNN 节点计算

完整的方程如下所示:

门	状态	
$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$	$\tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$	(7.97)
$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$	$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$	
$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$	$h_t = o_t \odot \sigma(s_t)$	

注意: 某些版本的 LSTM 架构不会具有“遗忘门”功能, 而是仅具有输出门和输入门来控制信息流. 它将仅实现选择性读和选择性写策略.

- 门控循环单元—GRU

门控循环单元是 LSTM 的一个变体. GRU 使用的门较少. 在像 LSTM 一样的门控循环单元中, 我们有一个输出门, 可以控制哪些信息进入下一个隐藏状态. 同样, 我们还有一个输入门, 它控制从当前输入中流入的信息. LSTM 和 GRU 之间的主要区别在于它们将中间隐藏状态向量和先前的隐藏状态表示向量 s 组合的方式. 在 LSTM 中, forget 确定要从 s 中保留多少信息. 在 GRU 中, 我们根据输入门向量 ($1-i_t$) 的剩余来决定保留或丢弃多少过去的信息, 而不是遗忘门.

$$f_t = 1 - x_t(1 - i_t). \quad (7.98)$$

GRU 的完整方程式如下:

门	状态	
$o_t = \sigma(W_o s_{t-1} + U_o x_t + b_o)$	$\tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + U x_t + b)$	(7.99)
$i_t = \sigma(W_i s_{t-1} + U_i x_t + b_i)$	$s_t = (1 - i_t) \odot s_{t-1} + i_t \odot \tilde{s}_t$	

从方程式中, 我们可以注意到只有两个门 (输入和输出), 并且我们没有显式计算隐藏状态向量 h . 因此, 我们没有在 GRU 中维护额外的状态向量, 即拥有比 LSTM 更少的计算和更快的训练.

循环神经网络在处理较长句子时的不足之处. RNN 受短期记忆的困扰, 即在信息变形之前, 它只能存储有限数量的状态. 之后, 我们详细讨论了通过使用门机制控制信息流的选择性读, 写和遗忘策略在 LSTM 中的工作方式. 最后, 我们研究了 LSTM 的变体 (称为门控循环单元), 其门数和计算量均比 LSTM 模型少.

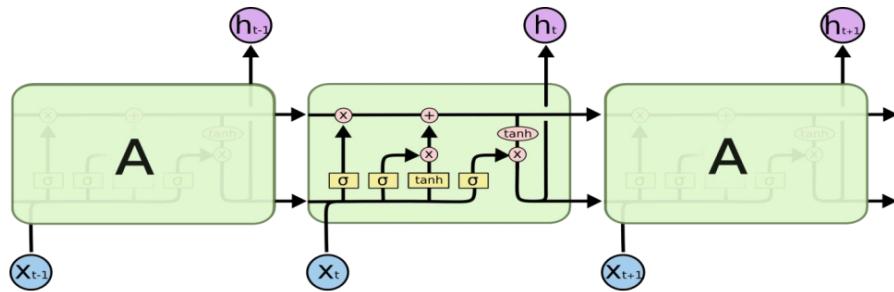


图 7-75 长短期记忆 (Long Short-Term Memory, LSTM)

LSTM 改变了循环体 A 的网络结构, 引入“门”的概念, 让信息有选择地影响循环神经网络中每个时刻的状态. 长短期记忆 LSTM (Long Short-Term Memory, LSTM) 可以解决梯度消失问题和长期依赖问题. 应用在诸如语言建模和文本生成、机器翻译、语音识别、生成图像描述和视频标记.



图 7-76 机器翻译



图 7-77 生成图像描述



图 7-78 语音识别

7.8.6 深度神经网络轻量化

“深度学习”的学习模式：“深度学习”这个名词其实最主要是针对人工智能领域，它是指机器的一种学习模式，与我们平常所指的人类的深度学习是不同的概念。

- ▶ “深度学习”是机器学习的一种特定技术，它是建立在大数据的基础上，用大数据对机器进行训练，在机器反馈的过程中不断地纠错。

例 7.26 对于婴孩来说，我们拿三个不同杯子告诉宝宝，这是杯子。基本上，他就知道什么是杯子了。不管第四个杯子的形状或者图案有什么不同，但孩子基本已经能够非常清楚地辨别杯子的本质特征。然而，对于机器来说，它需要通过足够多的例子，

例 7.27 你给它展示一个圆柱形的杯子，告诉它：“这是杯子。”但是当你拿另一个圆形的杯子给它看的时候，它可能会说这是一个球。所以，深度学习就是建立

在大数据的基础上, 给它看足够多的数据, 直到机器最后总结出杯子的本质特征.

- ▶ “深度学习”技术, 是建立在“人工神经网络”上的, 而人工神经网络, 又是来自人类神经生物学的深刻理解和延展. 我们人类的神经网络通过神经元的联结来传递和处理信息, 而人工神经网络则是模拟大脑的工作机制. 通过研究人类智能的工作机制, 可以模拟出人工智能的工作原理.
- ▶ 深度学习不比人类的学习, 人类通过认知、推理、总结和反思来认识陌生的事物, 但机器没有人类的意识和思维, 它需要足够多的数据, 才能得出结论, 而且, 机器到底是怎么得出结论的, 我们尚未可知.

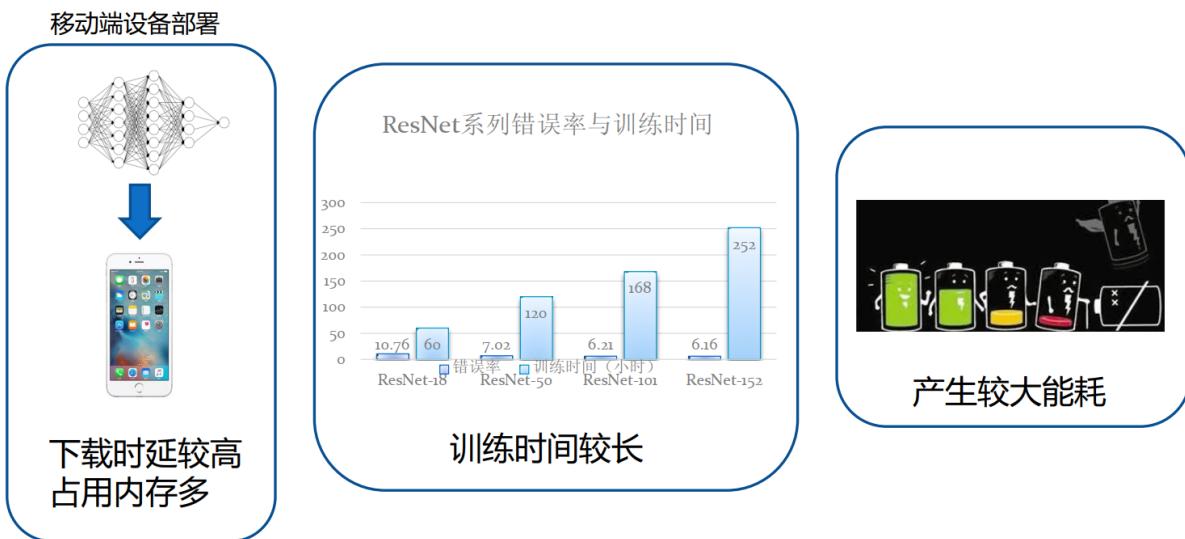


图 7-79 深度神经网络应用的轻量化

网络模型结构改进:

MobileNet

堆叠地使用深度可分离卷积; 深度可分离卷积由深度卷积和 1×1 的逐点卷积组成.

ShuffleNet

逐点分组卷积; 通道重排 (channel shuffle); 堆叠地使用瓶颈单元 ShuffleNet 单.

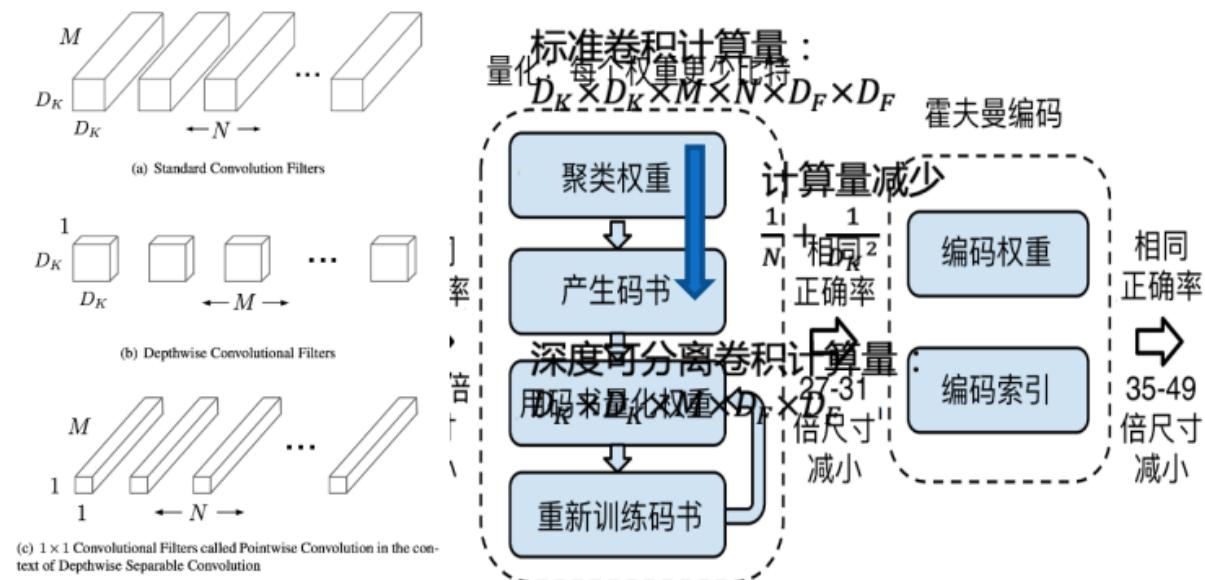


图 7-80 网络模型参数压缩

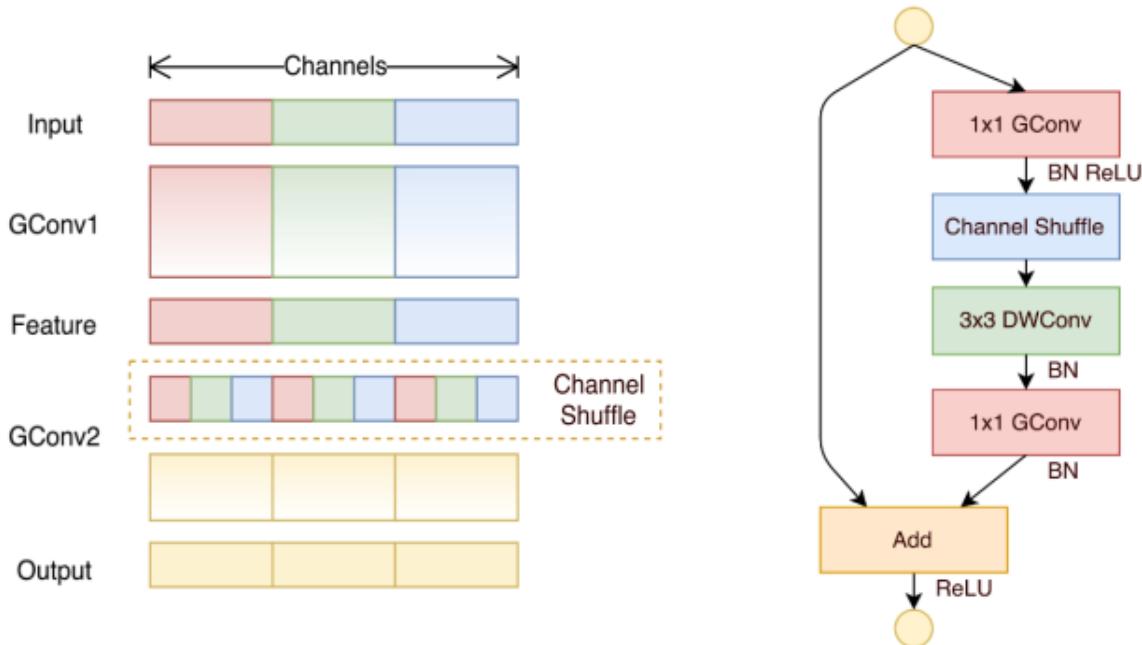


图 7-81 通道重排和 ShuffleNet 单元

7.8.7 防止 CNN 的过拟合——diffGrad

Train CIFAR10 with PyTorch. [diffGrad: An Optimization Method for Convolutional Neural Networks](#). IEEE Transactions on Neural Networks and Learning Systems, 2019. DOI: 10.1109/TNNLS.2019.2955777, 2919-12-23.

DiffGrad 是 Dubey 等人在论文《DiffGrad: CNN 的优化器》中介绍的一种新的优化器, 它建立在 Adam 优化器的基础上, 开发了一种自适应的“friction clamp”, 并监测梯度的局部变化, 以便自动锁定 Adam 可以跳过的最优参数值。通过“friction clamping”锁定到最优的最小值: 相比之下, diffGrad 监测当前梯度与前一步的即时变化, 并应用自适应“friction clamping”, 可在梯度变化很小的时候迅速减速, 从而暗示最优解可能就在附近。diffGrad 能够以理想的损失锁定到全局最小值。Adam 由于无法及时减速而跳到了更高的局部最小值。一些论文已经提出了解决已知 overshoot 问题的解决方案, 但是 diffGrad 可以稳定地解决它。论文还深入研究了其他几个变量如何应用 clamping 或 friction 系数。

[IEEE TNNLS version](#). All experiments are performed using the framework [pytorch-cifar](#)

7.8.8 无限宽度 DNN

无限宽度 DNN 会收敛成一类更为简单的模型, 称为高斯过程, 复杂的现象可以被归结为简单的线性代数方程。谷歌研究宣称可通过神经正切核使用贝叶斯推理或梯度下降分析式地训练无限宽度的神经网络。使用谷歌开源的软件库 ([Neural Tangents](#), [Neural Tangents: Fast and Easy Infinite Neural Networks in Python](#)), 过程不仅简单且快速, 而且效果非常好, 甚至只需 5 行代码就能一步到位地构建并训练这种无限宽度网络的集成模型! 论文已被 ICLR 2020 接收为 Spotlight 论文/深度学习都已取得了广泛的成功, 仍还有一些有趣而又重要的问题有待解答, 比如: 为什么即使在过度参数化时, 深度神经网络 (DNN) 也能取得非常好的泛化能力? 深度网络的架构、训练和性能之间有何关系? 如何提取出深度学习模型中的显著特征? 一大关键理论见解是: 增加 DNN 的宽度能使 DNN 的行为更有规律可循, 也就更容易理解它们。近来的许多研究已经表明, 宽度可无限扩增的 DNN 可以收敛成另一类更为简单的名为「高斯过程的模型」。因此, 贝叶斯推理与卷积神经网络的梯度下降情况等复杂现象便可约简为简单的线性代数方程。从这些无限宽度网络所得到的见解通常也适用于它们对应的有限版本。因此, 无限宽度网络可用作研究深度学习的透镜, 而且它们本身也可用作有用的模型。

, <https://github.com/google/neural-tangents> GitHub 地址, [Colab 地址](#).

7.9 强化学习及深度强化学习

7.10 AI 的未来-NeurIPS 2019

Thirty-third Conference on Neural Information Processing Systems. The purpose of the Neural Information Processing Systems annual meeting is to foster the exchange of research on neural information processing systems in their biological, technological, mathematical, and theoretical aspects. The core focus is peer-reviewed novel research which is presented and discussed in the general session, along with invited talks by leaders in their field. On Sunday is an Expo, where our top industry sponsors give talks, panels, demos, and workshops on topics that are of academic interest. On Monday are tutorials, which cover a broad background on current lines of inquiry, affinity group meetings, and the opening talk & reception. The general sessions are held Tuesday - Thursday, and include talks, posters, and demonstrations. Friday - Saturday are the workshops, which are smaller meetings focused on current topics, and provide an informal, cutting edge venue for discussion.

2019 年 12 月 8 日-14 日在加拿大温哥华举行,据官方消息,NeurIPS 今年共收到投稿 6743 篇,再次打破了历年来的接收记录. 今年接收论文 1429 篇,其中, Oral 论文 36 篇,占比 0.5%; Spotlight 论文接收量为 164 篇,占比 2.4%.

强化学习 61 篇,占比: 4.2%

理论大约 (21) 篇强化学习技巧 (3) 篇框架大约 (3) 篇探索和利用 (1) 篇元强化学习 (4) 篇分层强化学习 (2) 篇 + 逆强化学习 (2) 篇多智能体 (6) 篇奖励函数 (2) 篇应用 (6) 篇其他 (4) 篇

Google 对自动驾驶出租车实现的预测,已经改变了原来的乐观态度,变得充满克制. Facebook 的 AI 副总裁 Jerome Pesenti 最近表示,他的公司和其他公司不应该期待仅通过开发具有更多计算能力和数据的更大的深度学习系统来继续在 AI 方面取得进步.

Arcas 展示了一项模拟细菌的试验. 这些细菌通过人工进化的方式进行觅食和交流. 而 Yoshua Bengio 认为深度学习这个方法行得通,他正在往工具箱里增加更多的东西. 他在会议上做了主题为从深度学习系统 1 到深度学习系统 2 的演讲,提出软注意力和深度强化学习方式能够促进解决推理、计划、捕获因果关系等问题. 他的新方法受到了自然智能的启发. 根据意识的先验性进行相关假设,许多高级依赖关系可以通过稀疏因子图近似地捕获. 软注意力机制构成了一个关键因素,它可以一次将计算集中于几个概念(“意识思维”).

蒙特利尔大学副教授 Irina Rish: 深度学习很棒,但是我们需要一个不同的工具箱.

2006 年的一次非正式深度学习研讨会, 他比喻这次研讨会就像“宗教聚会”, 组织者拒绝接受边缘的技术想法. 虽然在今年的大会上, 深度学习是主流, 他希望自己的发言能够支持新的想法出现.

Uber 研究员 Jeff Clune 已经表示明年会加入 Open AI . 他还是新兴领域元学习 metalearning 的成员. 这一领域希望实现 AI 自己设计学习算法. 在演讲中, 他介绍了 POET 成对结合开放式开拓者, 让 AI 掌握自我进化来变得更聪明. 这一方法的灵感之一是自然进化. 他给了一个例子, 动画中的一双腿可以自动学习在更复杂的地形上走路.

7.11 作业

思考

假设 $w_1(0) = 0.2, w_2(0) = 0.4, \theta(0) = 0.3, \eta = 0.4$, 请用单层感知器完成逻辑或运算的学习过程.

思考

学习无监督的方式快速训练生成型循环神经网络 (Recurrent World Models Facilitate Policy Evolution, David Ha(谷歌 AI) 和 Jürgen Schmidhuber) [[ha2018worldmodels](#)].

思考

运行如下位置的演示代码: [Convolutional Neural Network](#)

8

自然语言理解

自然语言

自然语言是指人类日常交流所使用的语言。自然语言理解主要研究如何使计算机能够理解和生成自然语言。自然语言理解既是人工智能研究较早的一个领域，同时也是现代计算机的一个必备特征。



图 8-1

2020 年 1 月, NLP 的抱抱脸 (hugging face) 团队发布了新资源! 帮助 NLP 过程中词语切分 (tokenization) 更快的库 Tokenizers. 只要 20 秒就能编码 1GB 文本, 适用 Rust、Python 和 Node.js, 已经在 GitHub 上获得了 800 多星.

在 NLP 模型训练中, 词语标记和切分往往是一个瓶颈. Tokenizer 能够训练新的词汇, 并且进行标记. 功能多样: 适用于 BPE/byte-level-BPE/WordPiece/SentencePiece 各种 NLP 处理模型可以完成所有的预处理: 截断 (Truncate)、填补 (Pad)、添加模型需要的特殊标记. 速度超级快: 只需要 20 秒就可以在 CPU 上标记 1GB 的文本. 目前适用三种编程语言: Rust/Python/Node.js.

github 的资源页面上提供了在 Python 上使用 Tokenizers 的示例, 进行简单的设置就可以使用, 目前可用于三种语言 Python、JS、Rust.

Transformer 架构 广泛用于自然语言处理中, 并且在许多任务中实现了 sota(「state-of-the-art」, 用于描述机器学习中取得某个任务上当前最优效果的模型). 为了获得这些结果, 研究者不得不开始训练更大的 Transformer 模型。在最大的配置中, 参数数量已经超过了 0.5B/层, 层数多达 64。

诸如此类的大型 Transformer 模型频频出现, 到底是客观上必须要求如此多的资源, 还是仅仅是因为处理效率不够高?

如果说每层的内存占用只有这么一些的话, 部署 Transformer 会比实际中更容易, 但是事情并不是这样的。以上的估计只包括了每层的内存占用情况和输入的激活损失, 并没有考虑 Transformer 上的内存占用问题:

由于激活需要被存储并用于反向传播, 有着 N 层的模型的大小比单层大了 N 倍;

由于中间的全连接层的深度 d_{ff} 通常远大于注意力激活层的深度 d_{model} , 因此需要占用很大的内存;

在长度为 L 的序列上的 attention 的计算和时间复杂度是 $O(L^2)$, 所以即使是一个有 64K 字符的序列就会耗尽 GPU 的内存。

研究者提出了一种 Reformer 模型来解决刚才说的那些问题:

可逆层 (Reversible layer), 在整个模型中启用单个副本, 所以 N factor 就消失了;

在前馈层 (feed-forward layer) 分开激活和分块处理, 消除 d_{ff} factor, 节省前馈层的内存;

基于局部敏感哈希 (locality-sensitive hashing, LSH) 的近似注意力计算, 让注意力层的 $O(L^2)$ 因子替代 $O(L)$ 因子, 实现在长序列上的操作。

•Pytorch implementation of Reformer. It has been validated with a toy auto-regressive task.

Install (如图8-2)

»>pip install reformer_pytorch

```

管理员: Windows PowerShell
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。
尝试新的跨平台 PowerShell https://aka.ms/pscore6
PS C:\Windows\system32> pip install reformer_pytorch
Collecting reformer_pytorch
  Downloading https://files.pythonhosted.org/packages/8c/80/8d291eac494f65ca34c2d41f08a1cb953629094e91c18e99a9d440b7d149/reformer_pytorch-0.6.3.tar.gz
Collecting revtorch (from reformer_pytorch)
  Downloading https://files.pythonhosted.org/packages/ac/c9/c357bfcccd16e81b671316135873722fc8645223a16af466637d038e22a9/revtorch-0.2.1.tar.gz
Requirement already satisfied: torch in c:\users\ocnzh\anaconda3\lib\site-packages (from reformer_pytorch) (1.0.0)
Building wheels for collected packages: reformer-pytorch, revtorch
  Building wheel for reformer-pytorch (setup.py) ... done
    Created wheel for reformer-pytorch: filename=reformer_pytorch-0.6.3-cp37-none-any.whl size=6559 sha256=578c6b094f85559c75a1b4a151d4cc46edef79437ad1652a3a667daf3f576426
    Stored in directory: C:\Users\ocnzh\AppData\Local\pip\Cache\wheels\4b\2a\ab\986f6111f65056bc7702f1e5calf9feddb141e4a21735b7588c
  Building wheel for revtorch (setup.py) ... done
    Created wheel for revtorch: filename=revtorch-0.2.1-cp37-none-any.whl size=5256 sha256=e9085b65366cd0de6b8b13ff6a67f80ca811d55f2e1134e68c47ea57dd5bc9a9
    Stored in directory: C:\Users\ocnzh\AppData\Local\pip\Cache\wheels\ee\3d\3e\e3cc9505cb7e7b99ed0324d0613879daf71765dbe6d90f3d22
Successfully built reformer-pytorch revtorch
Installing collected packages: revtorch, reformer-pytorch
Successfully installed reformer-pytorch-0.6.3 revtorch-0.2.1
PS C:\Windows\system32>

```

图 8-2 Reformer 安装

8.1 语言及其理解的基本概念

自然语言是音义结合的词汇和语法体系。词汇是语言的基本单位，它在语法的支配下可构成有意义和可理解的句子，句子再按一定的形式构成篇章等。其结构如图8-3所示：

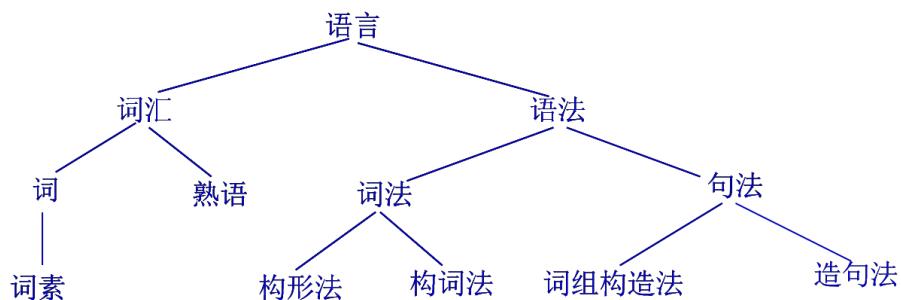


图 8-3 自然语言的结构

词汇是语言的基本单位。熟语是指一些词的固定组合，如汉语中的成语。词又由词素构成，词素是构成词的最小有意义的单位。如“学生”是由“学”和“生”这两个词素构成的。

语法是语言的组织规律。词法是用词素或熟语构成词的规则，可分为构形法和构词法。构形法是指单数复数等。造句法是用词和词组构造句子的规则。

8.2 词法分析

其主要任务是要找出词汇的各个词素，从中获得语言学信息，并确定单词的词义。以英语为例，其词法分析的基本算法如下：

```
repeat
    look for word in dictionary
    if not found
        then modify the word
until word is found or no further modification possible
```

其中，word 是一个变量，其初始值就是当前词。

例 8.1 用上述算法分析 catches.

解：其分析过程如下：

catches	词典中查不到
catche	修改 1：去掉 s
catch	修改 2：去掉 e

可以看出，在修改 2 时就查到了 catch. 当然，这只是一个很简单的例子，完整的词法分析还应该包括复合词的切分等。

8.3 句法分析

句法分析是对句子和短语的结构进行分析，其最大单位是一个句子。分析的目的是要找出词、短语等的相互关系，以及他们在句子中的作用等，并用一种层次结构加以表达。这种层次结构可以是句子的成分关系，也可以是语法功能关系。

8.3.1 句法规则的表示方法

句子结构的表示 一个句子是由各种不同的句子成分组成的。这些成分可以是单词、词组或从句。句子成分还可以按其作用分为：主语、谓语、宾语、宾语补语、定语、状语、表语等。这种关系可用一棵树来表示，如对句子：

He wrote a book

可用图8-4所示的树形结构来表示.

一个句子又是由若干个词类构成的, 如名词、动词、代词、形容词等. 若从句子的词类来考虑, 一个句子也可用一棵树来表示, 这种树称为句子的分析树, 如图8-4所示.

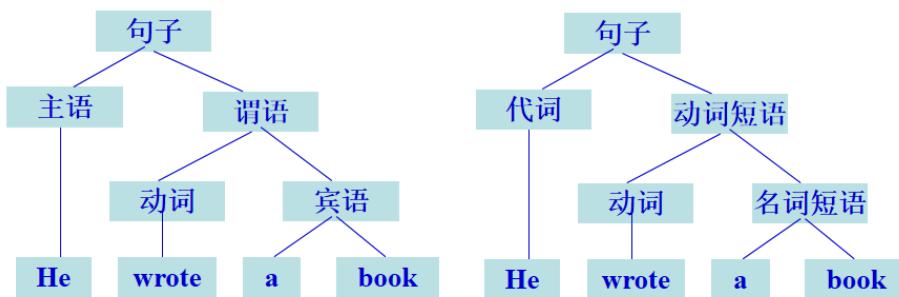


图 8-4

上下文无关文法 上下文无关文法 (Context-free Grammars) 是乔姆斯基提出的一种对自然语言语法知识进行形式化描述的方法. 在这种文法中, 语法知识是用重写规则表示的. 作为例子, 下面给出了一个英语的很小的子集 (图 8.4) .

- ▶ 语句 → 句子 终标符
- ▶ 句子 → 名词短语 动词短语
- ▶ 动词短语 → 动词 名词短语
- ▶ 名词短语 → 冠词 名词
- ▶ 名词短语 → 专用名词
- ▶ 冠词 → the
- ▶ 名词 → professor
- ▶ 动词 → wrote
- ▶ 名词 → book
- ▶ 动词 → trains
- ▶ 专用名词 → Jack
- ▶ 终标符 → .

这就是一个英语子集的上下文无关文法

在该文法中, “语句”是一个特殊的非终极符, 称为起始符.

句法规则的表示方法

例 8.2 利用上述上下文无关文法, 给出如下语句的分析树.

The professor trains Jack.

解: 如图8-5所示

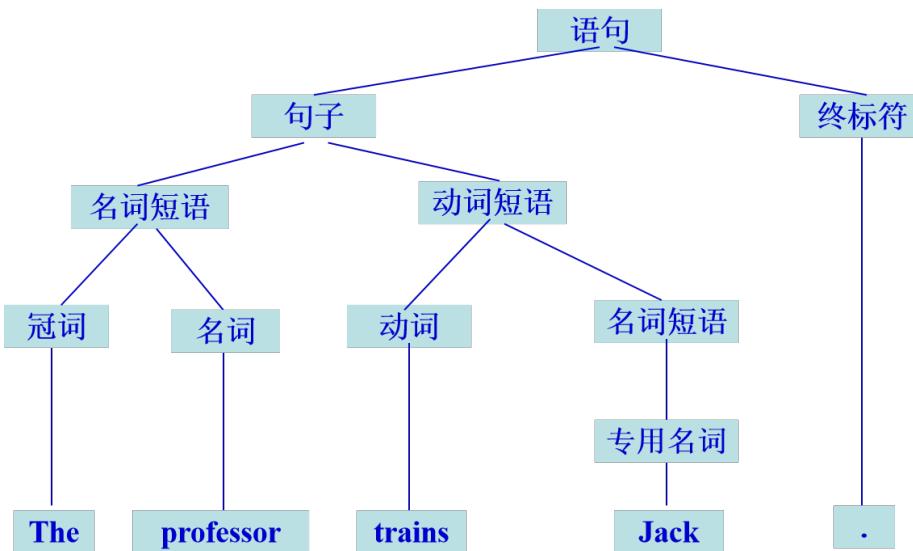


图 8-5

变换文法 上下文无关文法反映的仅是一个句子本身的层次结构和生成过程, 而自然语言是上下文有关的. 为此, 乔姆斯基又提出了变换文法 (Transformational Grammar). 该文法认为, 句子的结构有深层和表层两个层次. 例如:

She read me a story 和 She read a story to me

的表层结构不一样, 但它们的深层结构则是一样的. 再如, 主动句和被动句也只是表层结构不同, 其深层结构则是相同的.

在变换文法中, 句子深层结构和表层结构之间的变换是通过变换规则实现的, 如图8-6给出了一条把主动句变换为被动句的变换规则.

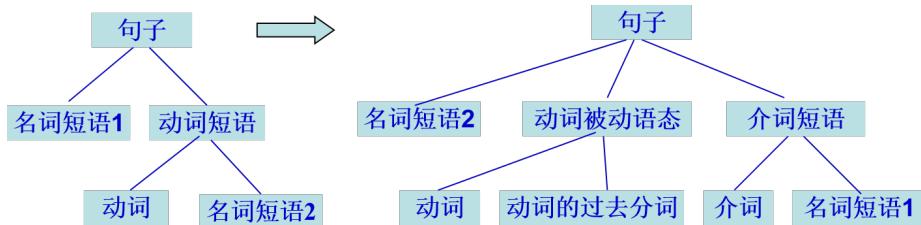


图 8-6

例 8.3 利用变换文法, 将前述主动句变为被动句.

解: 其变换过程是: 先从非终极符“句子”开始产生一个主动句:

The professor trains Jack

然后再应用图8-7所示的变换规则把它变为被动句(图8-7):

Jack is trained by the professor

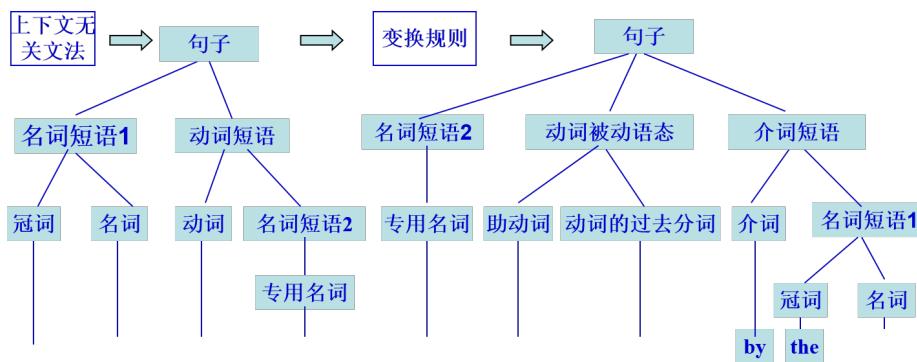


图 8-7

自顶向下与自底向上分析 自顶向下分析, 是指从起始符开始应用文法规则, 一层一层地向下产生分析树的各个分支, 直至生成与输入语句相匹配的完整的句子结构为止.

例 8.4 图8-8所示的上下文无关文法, 采用自顶向下分析方法对语句:

The professor trains Jack.

进行分析的过程是: 首先从起始符“语句”开始, 正向运用规则:

语句 → 句子终标符

把分析树的根节点“语句”替换为它的两个子节点“句子”和“终标符”. 然后再对新生成的节点“句子”使用规则:

句子 → 名词短语动词短语

将其替换为两个子节点“名词短语”与“动词短语”. 对于“名词短语”, 有两条规则可用, 若按规则的排列顺序, 则选用

名词短语 → 冠词名词

将“名词短语”被替换为“冠词”和“名词”，生成两个新节点。对“冠词”使用规则：

冠词 → The

对名词使用规则：

名词 → professor

以此进行…，得到如图 8.8 所示的自顶向下的分析树。

自顶向下与自底向上分析 自底向上分析，是以输入语句的单词为基础，首先按重写规则的箭头指向，反方向使用那些最具体的重写规则，把单词归并成较大的结构成分，如短语等，然后对这些成分继续逆向使用规则，直到分析树的根节点为止。

仍以语句

The professor trains Jack

为例，逆向使用图8-6中的那些具体规则后，可得到图8-8所示的部分分析树。

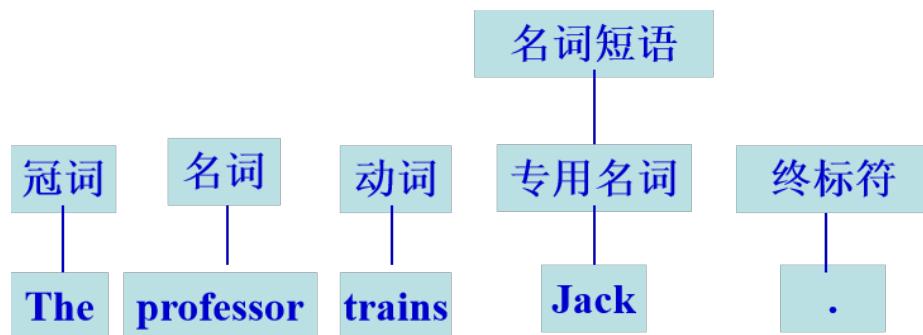


图 8-8

继续逆向使用规则，一步步归并，直到根节点“语句”为止，最后即可生成如图8-7所示的完整的分析树。

自顶向下分析方法与自底向上分析方法虽然思路清晰，但分析效率不高。为了提高分析效率，可采用自顶向下与自底向上相结合的分析方法。

8.4 句义分析

语义分析就是要识别一句话所表达的实际意义. 即弄清楚“干了什么”，“谁干的”，“这个行为的原因和结果是什么”以及“这个行为发生的时间、地点及其所用的工具或方法”等.

原因是语法分析, 仅是在句法范围内根据词性信息来分析自然语言中句子的文法结构的. 由于它没有考虑句子本身的含义, 也就不能排除像

The paper received the professor

这种在语法结构上正确, 但实际意义上错误的句子.

目前, 用于语义分析的技术比较多, 本节仅简单介绍语义文法和格文法.

8.4.1 语义文法

语义文法是在上下文无关文法的基础上, 将“名词短语”、“动词短语”和“名词”等这些不含有语义信息的纯语法类别, 用所讨论领域的专门信息, 像“山”、“水”和“动物”、等这些具有很强语义约束的语义类别来代替. 利用语义文法进行语义分析, 就可以排除像“论文收到教授”这类无意义的句子.

例 8.5 下面是一个关于舰船信息的语义文法的例子:

```
S $ \rightarrow $ PRESENT the ATTRIBUTE of SHIP  
PRESENT $ \rightarrow $ what is | can you tell me  
ATTRIBUTE $ \rightarrow $ length | class  
SHIP $ \rightarrow $ the SHIPNAME | CLASSNAME class ship  
SHIPNAME $ \rightarrow $ Huanghe | Changjiang  
CLASSNAME $ \rightarrow $ carrier | submarine
```

在上述重写规则中, 用大写英文字母的单词表示非终极符, 小写英文字母表示终极符, 坚线表示“或”的意思.

利用上述语义文法进行语义分析, 可以从语义上识别以下的输入:

what is the length of the Huanghe?
Can you tell me the class of the Changjiang?

8.4.2 格文法

格和格框架 格文法是以句子的中心动词为主导, 并用格来表示其它成分与此中心动词之间的语义关系的一种描述方法.

“格”这个词来源于传统语法, 但它与传统语法中的格有着本质不同. 在传统语法中, 格仅表示一个词或短语再句子中的功能, 如主格、宾格、等, 反映的也只是词尾的变化规则, 故称为表层格. 在格文法中, 格表示的是语义方面的关系, 反映的是句子中所包含的思想、观念等, 故称为深层格.

“格”是一个一般的概念, 相对于中心动词的不同语义关系, 格可以分为许多种. 例如, 在句子

John gave the book to Sally

中, 相对于中心动词 gave,

- ▶ John 是这个行为的发出者, 称为动作格;
- ▶ the book 是行为作用的对象, 称为受动格;
- ▶ Sally 是行为作用对象所到达的目标, 称为目标格.

一套正确的深层格究竟应包括多少个格, 以及这些格的明确含义是什么, 目前尚无定论.

下面给出一个描述行为的句子, 它所涉及的深层格主要有:

- ▶ Agent (施事), 动作主格, 指行为的施动者;
- ▶ Object (受事), 受动者格, 指行为作用的对象;
- ▶ Co-Agent (共施事), 帮助者格, 指行为施动者的合作者;
- ▶ Instrument (工具), 工具格, 指施事者或共施事者实现行为中所使用的对象;
- ▶ Time (时间), 时间格, 指行为发生的时间;
- ▶ Source (来源), 来源格, 指行为作用对象移出的位置;
- ▶ Goal (目标), 目标格, 指行为作用对象到达的位置;
- ▶ Trajectory (轨迹), 轨迹格, 指从来源到目标所经过的路径.

格框架是一种用来描述句子深层格的框架.

在格文法中, 每个句子都联系着一个格框架. 其中, 框架名可以是相应句子的中心动词, 框架的槽可分别对应于相应句子的各个深层格, 每个槽的槽值为该深层格在相应句子中所代表的语义成分.

例 8.6 前述句子分析结束时所得到的实际格框架为:

[GAVE

```

Agent: John
Object: the book
Source: John
Goal: Sally
]

```

8.4.3 长程记忆模型——Compressive Transformer（压缩 Transformer）

长短时记忆（Long Short-Term-Memory, LSTM）的递归神经网络（RNN）是目前最早、应用最为广泛的记忆结构之一。LSTM 以数字向量的形式维护一个紧凑的内存，通过门控读、写和遗忘操作来访问和修改这个内存。它最初是在一套综合任务上开发的，包括学习一串 bit 的逻辑操作。不过现在它已经被广泛应用在所有的序列数据模型当中了。LSTM，以及许多现在所使用的 RNNs，存在一个巨大的缺点，就是容量问题。最初设计这些结构的目的是为了，使每个单元的内存都可以影响其他单元，并且具有科学系的权重。但这导致系统的计算效率非常低下，模型中可学习参数的数量会随内存大小的增加呈平方地增加，例如内存 64KB 的 LSTM，会产生 8GB 的参数（图8-9）。

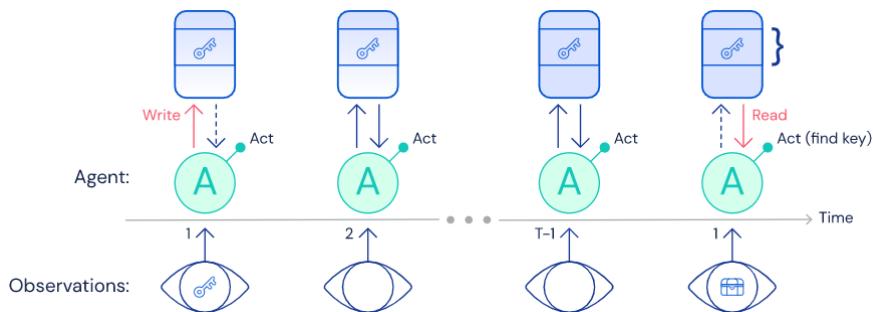


图 8-9 长期推理对于智能系统的泛化有决定性作用。Here, an agent remembers the existence and location of a key over a long period of time, and recalls this information when a treasure chest is discovered —prompting the agent to return to the remembered location to retrieve the key.

DeepMind 的研究人员曾提出过一种新的架构，可微分神经计算机 (DNC)，它用更大的内存矩阵来扩充 LSTM，以此来解决这些缺陷。在我们看东西时，我们的眼睛会聚焦于视觉场景中的相关物体。例如，你可能会花更多的时间注意朋友的面部表情，而不是注意他们的鞋子。DNC 采用了类似的方法，使用一个「注意力操作」从这个内存矩阵中读取数据。在 DNC 中，内存模型可以处理过去的特定事件/数据。这种注意力操作需要固定数量的参数，而与内存大小无关，因此可以显著提高模型的内存容量。随着 DNC 的开发，带

有附加注意力机制的递归神经网络在翻译和问题回答领域显示出了巨大的潜力。这些模型能够使用两种内存结构进行推理，一种是小型且紧凑的 LSTM 内存，一种是大型的外部内存。最近谷歌 Google Brain 的研究人员去除掉 LSTM，只利用注意力来传输信息，提出了一种 Transformer 模型。Transformer 最初是应用在机器翻译任务上，性能明显优于递归神经网络。随后 Transformer 被广泛应用到 NLP 的其他任务当中，例如问答、文本摘要、情感分析等。过去一年，因为 Transformer，这些方面取得了巨大的进步。但这些模型仍然存在一个缺点，即它们会把所有的信息都存储起来，这样在每一个时间步上所消耗的计算成本和存储成本都非常大。我们的大脑显然不是这样做的，我们不会像摄像机那样，把我们一生当中接收到的所有信息存储起来。而是会根据相关性、惊喜度、危险性、重复次数等因素来选择、过滤、整合所有的输入刺激。换句话说，我们会把一生的经历压缩成一组亮点记忆，帮助我们来理解过去，以及更好地预测未来。这就是如何压缩的问题。之前有一些工作通过稀疏访问机制来尝试压缩注意力中的计算消耗。但稀疏注意力方法并不能解决存储问题，而且通常需要定制的稀疏核才能有效地实现。

压缩 Transformer DeepMind 提出了 Transformer 的一个简单变种：Compressive Transformer 模型（压缩 Transformer）。将过去隐藏激活（past hidden activations, 记忆）映射到一个更小的压缩表示集（压缩记忆）中。在记忆和压缩记忆上，压缩 Transformer 会使用相同的注意力机制，来学习查询它的短期颗粒记忆和长期粗记忆。

压缩 Transformer 保持对过去激活的细粒度记忆，然后将其压缩为更粗的压缩记忆。上面的模型有三层，一个序列长度 $n_s = 3$ ，记忆大小 $n_m = 6$ ，压缩记忆大小 $n_{cm} = 6$ [raecompressive2019]。高亮显示的记忆被压缩，每层使用压缩函数 fc 将其压缩到单个压缩记忆中，而不是在下一个序列中丢弃。

8.5 作业

探索

对下列每个语句给出文法分析树：

- (1) John wanted to go the movie with Sally.
- (2) John wanted to go to the movie with Robert Redford.
- (3) I heard the story listening to the radio.
- (4) I heard the kids listening to the radio.

思考

什么是自然语言处理? 自然语言处理的过程有哪些层次? 各层次的功能如何?

思考

机器翻译的一般过程包括哪些步骤? 试述每个步骤的主要功能。

思考

请说出对话系统的三个主要模块及对应功能。

思考

什么是问答系统? 请扼要介绍问答系统自动答题的步骤。

思考

什么是文本生成的语言模型? 举例说明构建语言模型的方法。

思考

访问[DeepTables](#), 学习该框架, 简述网络特性, 解释其在结构化数据上的表示学习能力?

9

分布智能

分布智能

个体再群体内呈现出高度结构化的组织,构成高度结构化的社会组织,成员有分工且具有相互信息传递的机制。群体能对信息正反馈够自然优化,从而产生自催化行为。



图 9-1

9.1 分布智能

9.1.1 分布智能

9.1.1.1 分布智能概述

分布式问题求解 分布式问题求解的主要任务是要创建大粒度的协作群体,使它们能为同一个求解目标而共同工作。其主要研究内容是如何在多个合作者之间进行任务划分和问题求解。在分布式问题求解系统中,数据、知识和控制均分布在各个结点上,并且没有一个结点能够拥有求解整个问题所需要的足够数据和知识,因此各结点之间必需通过相互协作才能有效地解决问题。

多 Agent 系统 多 Agent 系统是由多个自主 Agent 所组成的一种分布式系统。其主要任务是要创建一群自主的 Agent,并协调它们的智能行为。

多 Agent 系统与分布式问题求解的主要差别在于,不同 Agent 之间的目标可能相同,但也可能完全不同,每个 Agent 都必须具有与其它 Agent 进行自主协调、协作和协商的能力。多 Agent 研究的重点包括 Agent 结构、Agent 通信和多 Agent 合作等。

9.1.2 Agent 的结构

Agent 的基本结构 Agent 结构是指 Agent 的组成方式。其基本结构包括反应 Agent、慎思 Agent 及混和 Agent 的结构。

反应 Agent 反应 Agent 是一种不含任何内部状态,仅是简单地对外界刺激产生响应的 Agent。其结构如图9-2所示,它采用“感知--动作”工作模式。

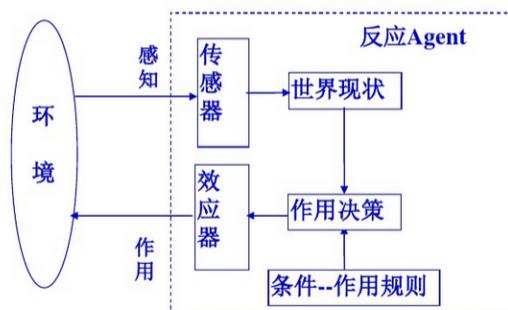


图 9-2 反应 Agent 的基本结构。

慎思 Agent 慎思 Agent 的基本结构如图 9.3 所示。在该结构中，Agent 的基本过程是先通过传感器接收外界环境信息，并根据内部状态进行信息融合，然后在知识库支持下制定规划、在目标引导下形成动作序列，最后由效应器作用于外部环境。

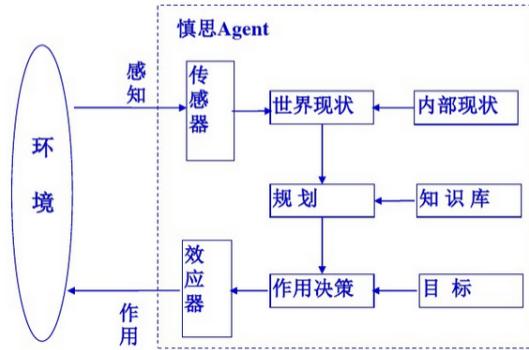


图 9-3 慎思 Agent 的基本结构。

BDI 的概念 BDI 的含义是信念-愿望-意图（Belief-Desire-Intention, 即 BDI），是一种典型的慎思 Agent 结构。

- ▶ 信念：是 Agent 对其环境和自身的认识。信念不同于知识，一般认为，知识是为真的信念。下面是关于信念的几种不同的解释：
 - 信念表示尚未完全证实的命题。
 - 信念表示不一定正确的命题。
 - 信念表示对已有证据积累的一种函数，即对命题的相信程度。
- ▶ 愿望：是 Agent 希望达到的目标，这些目标有可能有机会去实现，但也有可能永远无法实现。在实际应用中，一个 Agent 的初始愿望，通常是人交给 Agent 的任务。
- ▶ 意图：是 Agent 为达到愿望而计划采取的动作步骤。意图又可看作是 Agent 行为的控制器，它将引导和控制 Agent 的当前选择和未来活动。一个 Agent 的意图有可能会随着环境的变化而改变，即采取新的动作步骤。

9.1.3 Agent 通信

9.1.3.1 Agent 通信的基本问题

是指多 Agent 系统中不同 Agent 之间的信息交换：其基本问题包括：

通信方式

Agent 通信方式是指不同 Agent 之间的信息交换方式。常用的通信方式有消息传送和黑板系统等。

通信语言

Agent 通信语言是指相互交换信息的 Agent 之间共同遵守的一组语法、语义和语用的定义。常用的 Agent 通信语言有知识查询与操纵语言 KQML 等。

对话管理

Agent 之间的单个信息交换是 Agent 通信语言需要解决的基本问题，但 Agent 之间往往需要交换一系列信息，即需要进行对话。所谓对话是指 Agent 之间不断进行信息交换的模式，或者说 Agent 之间交换一系列消息的过程。

通信协议

- ▶ Agent 通信协议包括 Agent 通信使用的低层的传输协议和高层的对话协议。
- ▶ 低层的传输协议是指 Agent 通信中实际使用的低层传输机制，如 TCP、HTTP、FTP、SMTP 等。
- ▶ 高层的对话协议是指相互对话的 Agent 之间的协调协商协议。常用的描述对话协议的方法有有限状态自动机和 Petri 网等。

知识查询与操纵语言 KQML (Knowledge Query and Manipulation Language) 是目前国际上最著名的一种 Agent 通信语言。它由美国 DARPA 的知识共享计划 KSE 研究机构在 20 世纪 90 年代开发出来。

KSE 开发 KQML 的主要目的是为了解决基于知识的系统之间，以及基于知识的系统和常规数据库系统之间的通信问题。实际上，KSE 同时发布的还有一个知识交换格式 KIF，它主要是为了形成 KQML 的内容部分。在实际应用中，KQML 可基于某种元标记语言(如 XML) 来实现。

通信语言 KQML

KQML 语言的结构

- ▶ 从结构上看，KQML 是一种层次结构型语言。它可分为通信、消息和内容 3 个层次，各层的含义如下：
- ▶ 通信层描述的是通信协议和与通信双方有关的一组属性参数，例如发送者和接受者的身份、与通信有关的标识等。
- ▶ 消息层是 KQML 语言的核心，它描述的是与消息有关的言语行为的类型。其基本功能是确定传送消息所使用的协议和与传送消息有关的语言行为等。
- ▶ 内容层是消息所包含的真正内容，它可以是任何表示语言、ASCII 字符或二进制形式。对实现来说，KQML 并不需要关心消息中内容部分的具体含义。

KQML 消息：也称为“行为 (performative) 原语”或行为表达式，其基本格式是用

一对圆括号括起来的一个表. 表中的第 1 个元素是消息行为的名称, 后面的元素是一系列参数名及其参数的值.

KQML 消息 可简单地表示为:

```
(消息行为名称
  : 参数名 1    参数值 1
  : 参数名 2    参数值 2
  ...
)
```

其中, 消息行为名称用来指出该消息所引发的语言行为类型, 它由 KQML 保留的执行原语关键字来描述; 参数名及其值用来指出消息的属性、要求和内容等, 它由 KQML 保留的执行原语参数关键字来描述, 每个参数名都必须以冒号 (:) 开始, 后接相应的参数值.

KQML 语言的最大特点是其消息的参数以关键字为索引, 并且参数的顺序是无关的. 因此, 采用不同语言的异质系统之间能够很方便地分析和处理这些消息.

KQML 保留的行为原语参数 KQML 规范中定义了一部分常用的行为原语参数名和与其相关的一些参数值的含义, 这些参数称为保留参数.

这些保留参数是 Agent 通信中最基本最常用的关键字. 对他们进行统一定义, 可加快异质系统间信息交换和理解的速度. 其参数名及含义如下:

: sender	行为原语的实际发送者
: receiver	行为原语的实际接收者
: from	当使用 forward 转发时, : content 中行为原语的最初发送者
: to	当使用 forward 转发时, : content 中行为原语的最终接收者:
in-reply-to	对前条消息应答的标记, 其值与前条消息的: reply-with 值一致
: reply-with	对本条消息应答的标记
: language	: content 中内容信息的表示语言的名称
: ontology	: content 中内容信息使用的实体集 (如术语定义的集合) 名称
: content	有关行为原语表达内容的信息

在上面, **content** 参数表示行为原语的“直接目标”(即它的实际文字意义). **content** 的内容可以用通信双方都能识别的任何语言书写.

KQML 保留的行为原语 KQML 中定义的行为原语称为保留的行为原语. 这些行为原语可分为交谈类, 干预和对话机制类, 以及推进与网络类 3 种类型.

(1) 交谈类原语

这类原语用来实现 Agent 间一般信息的交换. 常用的有:

```
ask-if  :sender 想知道:receiver 是否认为:content 为真.
tell    :sender 向:receiver 表明:content 在:sender 中为真.
reply   :sender 向:receiver 传送一个对:receiver 的:content 的回答.
advertise :sender 承诺处理嵌入在 advertise 原语里的所有消息.
```

例如, Agent A 想发送一个行为表达式到 Agent B, 询问 $\text{bar}(x, y)$ 是否为真, 则其行为原语可表示为:

(ask-if

:sender	A	发送者 A
:receiver	B	接收者 B
:in-reply-to	id0	对前条消息应答的标记
:reply-with	id1	对本条消息应答的标记
:language	Prolog	:content 中内容使用的语言名称
:ontology	foo	:content 中内容使用的实体集名称
:content	“ $\text{bar}(\$x, y\$)$ ”)	表达的内容

(2) 干预和对话机制类原语

这类原语用于干预和调整正常的对话过程. 正常的对话过程一般是 Agent A 发送一条 KQML 消息给 Agent B, 当需要应答或谈话需要继续时, Agent B 发送响应消息, 如此循环直到对话结束. 最常用的 2 条是:

```
error  :sender 不能理解所接收的以:in-reply-to 为标识的消息,
即发送者认为它所接受的前一条消息出错.
sorry  :sender 理解接收到的消息, 消息在语法、语义方面都正确,
但:sender 不能提供任何应答; 或者:sender 能够提供进一步的应答,
但由于某种原因它决定不再继续提供. Sorry 意味着
Agent 要终止当前的对话过程.
```

(3) 网络类原语

这类原语是为了满足计算机网络通信与服务需要而设立的行为原语. 它们主要由通信服务器 Agent 使用, 或者其它 Agent 通过“advertise”原语来使用. 最常用的 3 条原语是:

```
register :sender 向:receiver 宣告其存在性以及与物理地址有关的符号名.  
forward :sender 希望:receiver 传送一条消息给另一个 Agent.  
recommend-one:sender 请求:receiver 推荐一个能够处理:content 的 Agent.
```

KQML 通信服务器 为提高分布式处理的透明性, KQML 引入了一个专门用来提供通信服务的特殊的 Agent 类型, 即被称为通信服务器的 facilitator. 它负责各种通信服务, 如: 维护服务名称的注册, 为命名的服务提供消息, 进行基于内容的路由选择等.

例 9.1 Agent A 想知道 x 是否为真, 请求 facilitator 希望找到一个能够处理 $\text{ask-if}(x)$ 的 Agent; 如果在 facilitator 保存的 Agent B 有能力处理 $\text{ask-if}(x)$, 则 facilitator 就将 Agent B 的名字返回给 Agent A; 然后 Agent A 就可以与 Agent B 进行对话, 并得到所需要的答案.

其工作过程如下图所示:

9.1.4 多 Agent 合作

多 Agent 系统可以看作是一个由一群自主并自私的 Agent 所构成的一个社会. 在这个社会中, 每个 Agent 都有自己的利益和目标, 并且它们的利益有可能存在冲突, 目标也有可能不一致.

像人类社会中具有不同利益的人为了实现各自的目标又需要进行合作一样, 多 Agent 系统也是如此.

多 Agent 的合作包括:

Agent 协调: 是指对 Agent 之间的相互作用和 Agent 动作之间的内部依赖关系的管理. 它描述的是一种动态行为, 反映的是一种相互作用的性质. 它的两个最基本的成分是: “有限资源的分配” 和 “中间结果的通信” .

Agent 协作: 协作是指 Agent 之间相互配合一起工作. 它是非对抗 Agent 之间保持行为协调的一个特例.

Agent 协商: 协商主要用来消解冲突、共享任务和实现协调, 是多 Agent 系统实现协调和解决冲突的一种重要方法.

Agent 的协调

常用的协调方法有:

- 基于部分全局规划的协调: 部分全局规划是指将一个 Agent 组的动作和相互作用进行组合所形成的数据结构. 所谓规划是部分的, 是指系统不能产生整个问题的

规划. 所谓规划是全局的, 是指 Agent 通过局部规划的交换与合作, 可以得到一个关于问题求解的全局视图, 进而形成全局规划.

- ▶ 基于联合意图的协调: 意图是 Agent 为达到愿望而计划采取的动作步骤. 联合意图则是指一组合作 Agent 对它们所从事的合作活动的整体目标的集体意图.
- ▶ 其典型例子是 Agent 机器人竞赛中的队内 Agent 机器人之间的协调问题, 这些 Agent 既有自己的个体意图, 又有全队的联合意图.
- ▶ 基于社会规范的协调: 基于社会规范的协调是一种以每个 Agent 都必须遵循的社会规范为基础的协调方法. 所谓规范是一种建立的、期望的行为模式. 社会规范可以对 Agent 社会中各 Agent 的行为加以限制, 以过滤掉某些有冲突的意图和行为, 保证其它 Agent 必须的行为方式, 从而确保 Agent 自身行为的可能性, 以实现整个 Agent 社会行为的协调.

Agent 的协作

合同网是 Agent 协作中最著名的一种协作方法, 被广泛应用于各种多 Agent 系统的协作中. 其思想来源于人们在日常活动中的合同机制. 在合同网系统中, 所有 Agent 被分为管理者和工作者两种不同角色.

管理者 Agent 的主要职责包括:

- (1) 对每一个需要求解的任务建立其任务通知书 (Task Announcement), 并将任务通知书发送给有关的工作者 Agent;
- (2) 接受并评估来自工作者 Agent 的投标 (Bid);
- (3) 从所有投标中选择最合适的工作者 Agent, 并与其签订合同 (Contract);
- (4) 监督合同的执行, 并综合结果.

工作者 Agent 的主要职责包括:

- (1) 接受相关的任务通知书;
- (2) 评价自己的资格;
- (3) 对感兴趣的子任务返回任务投标;
- (4) 如果投标被接受, 按合同执行分配给自己的子任务;
- (5) 向管理者报告求解结果.

合同网系统的基本工作过程如图9-4所示:

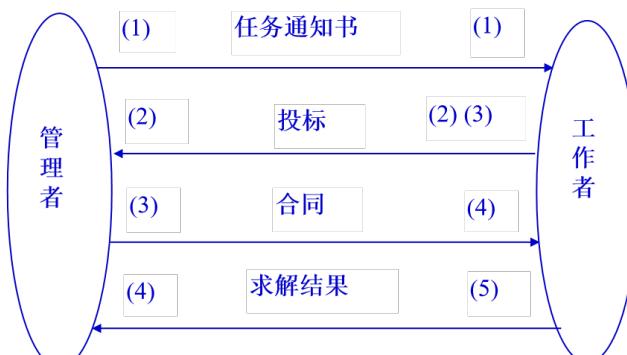


图9.9 合同网系统的基本工作过程

图 9-4 语义网络的基本网元

9.1.5 Agent 的协商

协商的主要方法包括协商协议、协商策略和协商处理。

协商协议 协商协议是用结构化方法描述的多 Agent 自动协商过程的一个协商行为序列。它需要详细说明初始化一个协商循环和响应消息的各种可能情况。最简单的协商协议是按照

< 协商原语 >< 协商内容 >

这种形式定义的一个可能的协商行为序列。

协商策略 协商策略是模型化 Agent 内部协商推理的控制策略，亦是实现协商决策的元级知识，在协商过程中起着重要的作用。协商策略主要用于 Agent 决策及选择协商协议和通信消息。

协商处理 协商处理包括协商算法和系统分析两个方面。其中，协商算法用于描述 Agent 在协商过程中的行为，如通信、决策、规划和知识库操作等；系统分析用于分析和评价 Agent 协商的行为和性能，回答协商过程中的问题求解质量、算法效率和公平性等问题。

决策支持系统杂志。

多 Agent 应用示例 多 Agent 系统的应用非常广泛, 诸如智能信息检索、分布式网络管理、电子商务、协同工作和智能网络教学系统等. 以智能网络教学系统为例:

例 9.2 学生模型数据库是学生知识结构的反映. 数据库 Agent 负责学生模型数据库、教学 Agent 群和界面 Agent 之间交互的管理. 教学策略 Agent 群中的每个 Agent 都相当于一个教育家. 教学过程管理 Agent 的主要功能是监视教学过程, 并向教学 Agent 提供教学参考意见. 教学 Agent 群是整个智能教学系统的核心, 其中的每个教学 Agent 都相当于一个教师. 界面 Agent 构成了系统的交互模型, 他主要负责与学生或教师的交互.

9.1.6 移动 Agent

移动 Agent (MA) 是一种可以从网络上一个结点自主地移动到另一个结点, 实现分布式问题处理的特殊的 Agent. 它由移动 Agent 和移动 Agent 环境两大部分所组成.

9.2 作业

思考

Agent 在结构上有什么特点? 它是如何按照结构进行分类的?

思考

什么是协调、协作、协商? 它们之间有什么联系和区别?

10

先进专家系统

专家系统

专家系统是一个具有专门知识与经验的程序系统,它应用人工智能技术和计算机技术,根据某领域一个或多个专家提供的知识和经验,进行推理和判断,模拟人类专家的决策过程,以便解决那些需要人类专家处理的复杂问题。专家系统是一个智能计算机程序系统,其内部含有大量的某个领域专家水平的知识与经验,能够利用人类专家的知识和解决问题的方法来处理该领域问题;是一种模拟人类专家解决领域问题的计算机程序系统。



10.1 专家系统

10.1.1 专家系统概述

专家系统和先进专家系统

专家系统的概念

专家系统是一种具有大量专门知识和经验的智能程序系统, 它能运用领域专家多年积累的经验和专门知识, 模拟领域专家的思维过程, 解决该领域中需要专家才能解决的复杂问题.

先进专家系统的概念

先进专家系统是指在传统专家系统的基础上, 引入一些新思想、新技术所产生的新型专家系统.

先进专家系统的特性

- (1) 并行分布式处理功能
- (2) 多专家协同工作
- (3) 更强的自学习能力
- (4) 更新的推理机制
- (5) 自纠错和自完善能力
- (6) 先进的智能接口
- (7) 更多的先进技术被引入和融合

10.1.2 专家系统的基本结构

尽管不同类型的专家系统的结构会存在一定差异, 但其基本结构还是大致相同的. 通常, 一个专家系统的基本结构由知识库、数据库、推理机、解释模块、知识获取模块和人机接口 6 大部分所组成. 如图10-1所示:

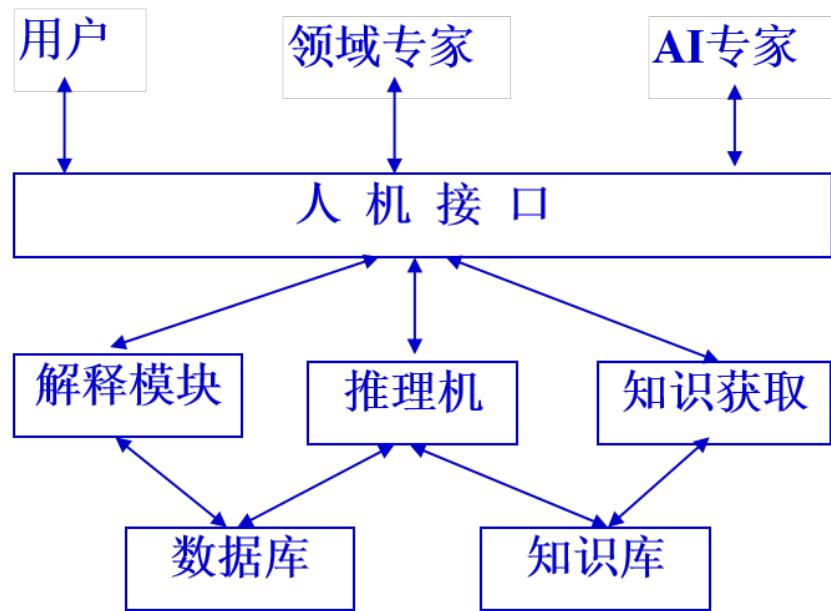


图10.1 专家系统的基本结构

图 10-1 模糊专家系统

10.1.3 基于规则和基于框架的专家系统

基于规则的专家系统是指采用产生式知识表示方法的专家系统。它以产生式系统为基础，是专家系统开发中常用的一种方式，其最基本的工作模型如图10-2所示。

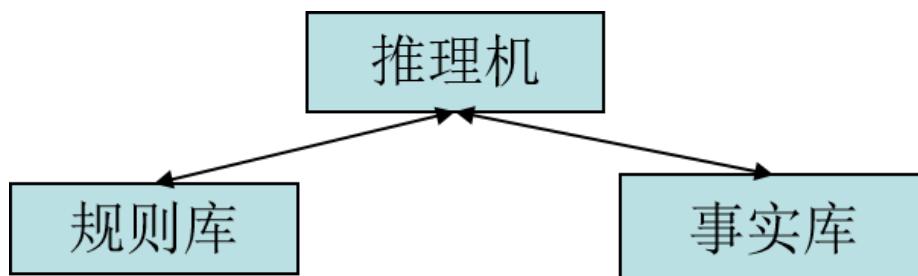


图 10-2 模糊专家系统

在该模型中，规则库是基于规则专家系统的知识库；事实库也称综合数据库，是用来存放推理前的已知事实和推理过程中所得到的中间结论的；推理机是基于规则专家系统的推理机构。

基于框架的专家系统是指采用框架知识表示方法的专家系统。它以框架系统为基础，具有较好的结构化特性。这种专家系统的基本结构也与图10-1所示的专家系统类似，其主要区别在于知识库中知识表示和组织方式，综合数据库中事实的表示方式，推理机的推理方法和系统推理过程的控制策略等。

10.1.4 模糊专家系统和神经网络专家系统

模糊专家系统 模糊专家系统是指采用模糊技术来处理不确定性的一类专家系统。模糊专家系统的基本结构与传统专家系统类似，一般由模糊知识库、模糊数据库、模糊推理机、知识获取模块、解释模块和人机接口6部分所组成。如图10-3：

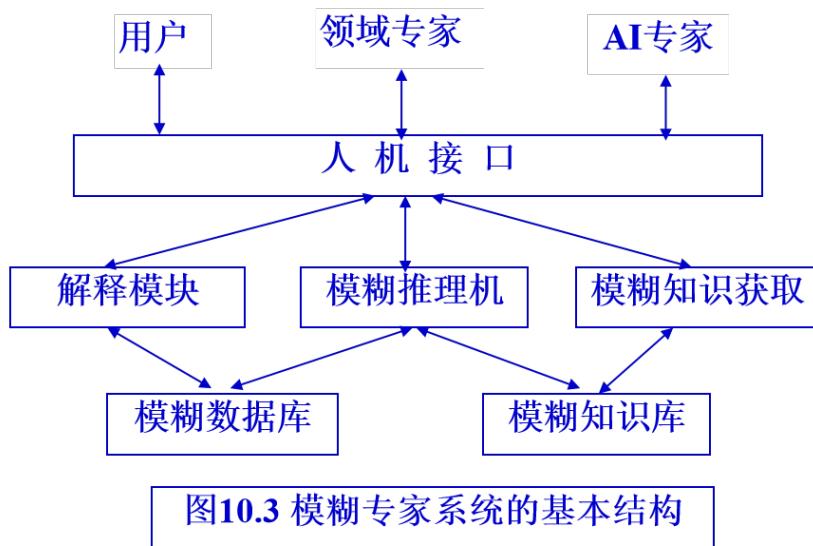


图 10-3 模糊专家系统

神经网络专家系统 神经网络专家系统是神经网络与传统专家系统集成所得到的一种专家系统。它将传统专家系统的显式的知识表示方法变为基于神经网络及其连接权值的隐式知识表示，把基于逻辑的串行推理技术变为基于神经网络的并行联想和自适应推理。

10.1.5 基于 Web 的专家系统

基于 Web 的专家系统是 Web 数据交换技术与传统专家系统集成所得到的一种先进专家系统。它利用 Web 浏览器实现人机交互，基于 Web 专家系统中的各类用户都可通过浏览器访问专家系统。从结构上，它由浏览器、应用服务器和数据库服务器三个层次所组成，包括 Web 接口、推理机、知识库、数据库和解释器。

10.1.6 分布式专家系统和协同式专家系统

这是两种不同的先进专家系统, 它们各自的侧重点不一样. 分布式专家系统强调并行和分布, 而协同式专家系统则强调协作与协同.

分布式专家系统 分布式专家系统 (Distributed Expert System, DES) 是具有并行分布处理特征的专家系统, 它可以把一个专家系统的功能分解后, 分布到多个处理机上去并行执行, 从而在总体上提高系统的处理效率. 其运行环境可以是紧密耦合的多处理器系统, 也可以是松耦合的计算机网络环境.

协同式专家系统 协同式专家系统 (Cooperative Expert System, CES) 亦称群专家系统, 是一种能综合若干个相近领域或同一领域内不同方面专家系统相互协作、共同解决单个专家系统无法解决的更广领域或更复杂问题的专家系统.

从结构上它们有一定的相似之处, 它们都涉及到多个分专家系统. 但在功能上却有较大差异, 分布式专家系统强调的是功能分布和知识分布, 它要求系统必须在多个节点上并行运行; 而协调式专家系统强调的则是各专家系统之间的协同, 各分专家系统可以在不同节点上运行, 也可以在同一个节点上运行.

10.2 专家系统的开发

开发步骤 采用原型技术的专家系统开发过程如下图所示, 它可分为设计初始知识库、原型系统开发与试验、知识库的改进与归纳三个主要步骤.

知识获取

输入知识

开发工具与环境 常用的专家系统开发工具和环境可按其性质分为程序设计语言、骨架型工具、语言型工具、开发环境及一些新型专家系统开发工具等.

- ▶ 程序设计语言 程序设计语言包括人工智能语言和通用程序设计语言. 它们是专家系统开发的最基础的语言工具. 人工智能语言的主要代表有以 LISP 为代表的函数型语言和以 PROLOG 为代表的逻辑型语言等; 通用程序设计语言的主要代表有 C、C++ 和 JAVA 等.

- ▶ 骨架型工具骨架型工具也称为专家系统外壳, 它是由一些已经成熟的具体专家系统演变来的. 其演变方法是, 抽去这些专家系统中的具体知识, 保留它们的体系结构和功能, 再把领域专用的界面改为通用界面, 这样, 就可得到相应的专家系统外壳.
- ▶ 语言型工具语言型工具是一种通用型专家系统开发工具, 它是不依赖于任何已有专家系统, 不针对任何具体领域, 完全重新设计的一类专家系统开发工具. 与骨架系统相比, 语言型工具具有更大的灵活性和通用性, 并且对数据及知识的存取和查询提供了更多的控制手段. 常用的语言型工具有 CLIPS 和 OSP 等.

▶ 开发环境

专家系统开发环境是一种为高效率开发专家系统而设计和实现的大型智能计算机软件系统. 专家系统开发环境一般由调试辅助工具、输入输出设施、解释设施和知识编辑器 4 个典型部件所组成.

10.3 作业

思考

先进专家系统有哪些主要特征? 它有哪些主要类型?

思考

什么是分布式专家系统? 什么是协同式专家系统? 它们的主要区别是什么?

11

计算机视觉 CV



11.1 深度学习 VS 传统计算机视觉

Deep Learning vs. Traditional Computer Vision, CVC 2019 Apr 25, 2019 - Apr 26, 2019.
Las Vegas, Nevada, United States [MahonyCVC2019]
Animal Face、Anime Face、Oxford Flower、CUB-200-2011 和 Caltech-256 数据集

11.1.1 深度学习的优势

深度学习的快速发展和设备能力的改善（如算力、内存容量、能耗、图像传感器分辨率和光学器件）提升了视觉应用的性能和成本效益，并进一步加快了此类应用的扩展。与传统 CV 技术相比，深度学习可以帮助 CV 工程师在图像分类、语义分割、目标检测和同步定位与地图构建（SLAM）等任务上获得更高的准确率。由于深度学习所用的神经网络是训练得到而非编程得到，因此使用该方法的应用所需的专家分析和微调较少，且能够处理目前系统中的海量可用视频数据。深度学习还具备绝佳的灵活性，因为对于任意用例，CNN 模型和框架均可使用自定义数据集重新训练，这与 CV 算法不同，后者具备更强的领域特定性。

以移动机器人的目标检测问题为例，对比这两类计算机视觉算法：

传统计算机视觉方法使用成熟的 CV 技术处理目标检测问题，如特征描述子（SIFT、SUR、BRIEF 等）。在深度学习兴起前，图像分类等任务需要用到特征提取步骤，特征即图像中「有趣」、描述性或信息性的小图像块。这一步可能涉及多种 CV 算法，如边缘检测、角点检测或阈值分割算法。从图像中提取出足够多的特征后，这些特征可形成每个目标类别的定义（即「词袋」）。部署阶段中，在其他图像中搜索这些定义。如果在一张图像中找到了另一张图像词袋中的绝大多数特征，则该图像也包含同样的目标（如椅子、马等）。

传统 CV 方法的缺陷是：从每张图像中选择重要特征是必要步骤。而随着类别数量的增加，特征提取变得越来越麻烦。要确定哪些特征最能描述不同的目标类别，取决于 CV 工程师的判断和长期试错。此外，每个特征定义还需要处理大量参数，所有参数必须由 CV 工程师进行调整。深度学习引入了端到端学习的概念，即向机器提供的图像数据集中的每张图像均已标注目标类别。因而深度学习模型基于给定数据「训练」得到，其中神经网络发现图像类别中的底层模式，并自动提取出对于目标类别最具描述性和最显著的特征。人们普遍认为 DNN 的性能大大超过传统算法，虽然前者在计算要求和训练时间方面有所取舍。随着 CV 领域中最优秀的方法纷纷使用深度学习，CV 工程师的工作流程出现巨大改变，手动提取特征所需的知识和专业技能被使用深度学习架构进行迭代所需的知识和专业技能取代（见图 1）。

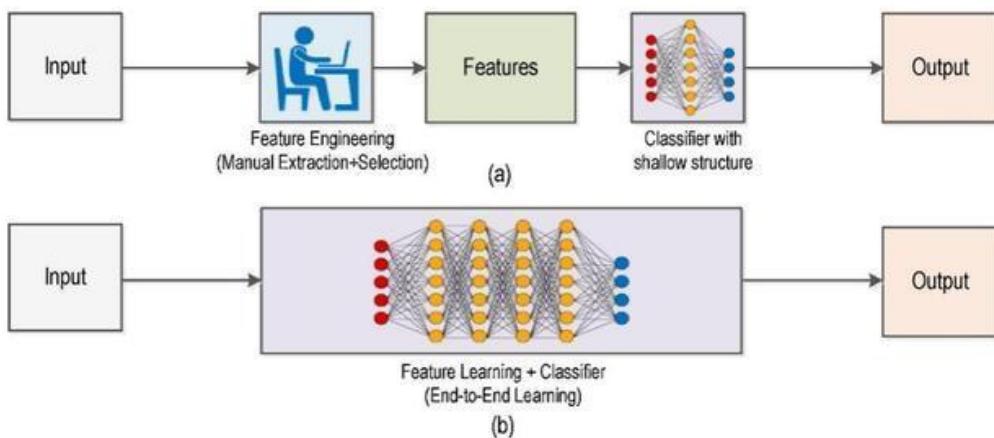


图 11-1 a) 传统计算机视觉工作流 vs b) 深度学习工作流 [WANG2018144]

近年来, CNN 的发展对 CV 领域产生了巨大影响, 也使得目标识别能力出现大幅提升。这种爆发与算力的提升、训练数据量的增加密不可分。近期 CV 领域中深度神经网络架构出现井喷并得到广泛应用, 这从论文《ImageNet Classification with Deep Convolutional Neural Networks》引用量超 3000 次中可见一斑。

CNN 利用卷积核 (又称滤波器) 来检测图像中的特征 (如边)。卷积核是权重矩阵, 这些权重被训练用于检测特定特征。如名字所示, CNN 的主要思想是在给定输入图像上空间性地卷积内核, 检查是否出现检测所需特征。为了用数值表示出现某个特征的置信度, 神经网络执行卷积操作, 即计算卷积核与它和输入图像重叠区域的点积 (卷积核正在查看的原始图像区域叫做感受野)。

为了促进卷积核权重的学习, 研究人员向卷积层的输出添加偏置项, 并馈入非线性激活函数中。激活函数通常是非线性函数, 如 Sigmoid、Tanh 和 ReLU。激活函数的选择取决于数据和分类任务的性质。例如, ReLU 具备更多生物表征 (大脑中的神经元是否处于激活状态)。因此, 在图像识别任务中, ReLU 会得到更好的结果, 因为它对梯度消失问题具备更强的抵抗力, 而且它能够输出更稀疏、高效的表征。

为了加速训练过程, 减少网络消耗的内存量, 卷积层后通常跟着一个池化层, 用于移除输入特征中的冗余部分。例如, 最大池化在输入上移动窗口, 仅输出窗口中的最大值, 从而高效减少图像中的冗余部分, 留下重要像素。如图 2 所示, 深度 CNN 可能具备多对卷积和池化层。最后, 全连接层将上一层压缩为特征向量, 然后输出层利用密集网络计算输出类别/特征的分数 (置信度或概率)。将该输出输入到回归函数中, 如 Softmax 函数, 它将所有事物映射为向量且其中所有元素的总和为 1。

但是深度学习仍然只是 CV 领域的工具。例如, CV 领域中最常用的神经网络是

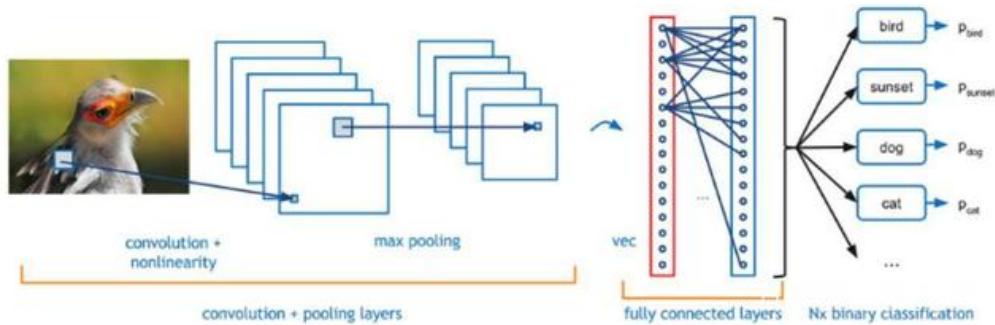


图 11–2 CNN 构造块。(图源: [13])

CNN。那么什么是卷积呢？卷积广泛应用于图像处理技术。（深度学习的优点很明确，本文暂不讨论当前最优算法。）但深度学习并非解决所有问题的万灵药，下文将介绍传统 CV 算法更适合的问题及应用。

11.1.2 传统 CV 技术的优势

基于特征的传统方法在 CV 任务中能够有效提升性能的原因。这些传统方法包括：

- ▶ 尺度不变特征变换（Scale Invariant Feature Transform, SIFT）[14]
- ▶ 加速稳健特征（Speeded Up Robust Feature, SURF）[15]
- ▶ 基于加速分割测试的特征（Features from Accelerated Segment Test, FAST）**[Rosten2006]**
- ▶ 霍夫变换（Hough transform）**[Goldenshluger2004]**
- ▶ 几何哈希（Geometric hashing）[18]
- ▶ 特征描述子（如 SIFT 和 SURF）通常与传统机器学习分类算法（如支持向量机和 K 最近邻算法）结合使用，来解决 CV 问题。

深度学习有时会「过犹不及」，传统 CV 技术通常能够更高效地解决问题，所用的代码行数也比深度学习少。SIFT，甚至简单的色彩阈值和像素计数等算法，都不是特定于某个类别的，它们是通用算法，可对任意图像执行同样的操作。与之相反，深度神经网络学得的特征是特定于训练数据的。也就是说，如果训练数据集的构建出现问题，则网络对训练数据集以外的图像处理效果不好。

因此，SIFT 等算法通常用于图像拼接/3D 网格重建等应用，这些应用不需要特定类别知识。这些任务也可以通过训练大型数据集来实现，但是这需要巨大的研究努力，为一个封闭应用费这么大力并不实际。在面对一个 CV 应用时，工程师需要培养选择哪种解决方案的常识。例如，对流水线传送带上的两类产品进行分类，一类是红色一类是蓝色。

深度神经网络需要首先收集充足的训练数据。然而, 使用简单的色彩阈值方法也能达到同样的效果。一些问题可以使用更简单、快速的技术来解决。如果 DNN 对训练数据以外的数据效果不好, 怎么办? 在训练数据集有限的情况下, 神经网络可能出现过拟合, 无法进行有效泛化。手动调参是非常困难的事情, 因为 DNN 拥有数百万参数, 且它们之间的关系错综复杂。也因此, 深度学习模型被批评为黑箱。传统的 CV 技术具备充分的透明性, 人们可以判断解决方案能否在训练环境外有效运转。CV 工程师了解其算法可以迁移至的问题, 这样一旦什么地方出错, 他们可以执行调参, 使算法能够有效处理大量图像。现在, 传统 CV 技术常用于解决简单问题, 这样它们可在低成本微处理器上部署, 或者通过突出数据中的特定特征、增强数据或者辅助数据集标注, 来限定深度学习技术能解决的问题。本文稍后将讨论, 在神经网络训练中可使用多少种图像变换技术。最后, CV 领域存在很多更具挑战性的难题, 比如机器学习、增强现实、自动全景拼接、虚拟现实、3D 建模、运动估计、视频稳定、运动捕捉、视频处理和场景理解, 这些问题无法通过深度学习轻松实现, 但它可以从传统 CV 技术中受益。

传统 CV 技术与深度学习的融合

传统 CV+ 深度学习 = 更好的性能

传统 CV 技术和深度学习方法之间存在明确的权衡。经典 CV 算法成熟、透明, 且为性能和能效进行过优化; 深度学习提供更好的准确率和通用性, 但消耗的计算资源也更大。混合方法结合传统 CV 技术和深度学习, 兼具这两种方法的优点。它们尤其适用于需要快速实现的高性能系统。

机器学习度量和深度网络的混合已经非常流行, 因为这可以生成更好的模型。混合视觉处理实现能够带来性能优势, 且将乘积累加运算减少到深度学习方法的 130-1000 分之一, 帧率相比深度学习方法有 10 倍提升。此外, 混合方法使用的内存带宽仅为深度学习方法的一半, 消耗的 CPU 资源也少得多。

充分利用边缘计算

当算法和神经网络推断要在边缘设备上运行时, 其延迟、成本、云存储和处理要求比基于云的实现低。边缘计算可以避免网络传输敏感或可确认数据, 因此具备更强的隐私性和安全性。结合了传统 CV 和深度学习的混合方法充分利用边缘设备上可获取的异质计算能力。异质计算架构包含 CPU、微控制器协同处理器、数字信号处理器 (DSP)、现场可编程逻辑门阵列 (FPGA) 和 AI 加速设备, 通过将不同工作负载分配给最高效的计算引擎来降低能耗。测试实现证明, 在 DSP 和 CPU 上分别执行深度学习推断时, 前者的目标检测延迟是后者的十分之一。多种混合方法证明了其在边缘应用上的优势。使用混合方法能够高效地整合来自边缘节点传感器的数据。

不适合深度学习的问题

CV 领域中存在一些难题, 如机器人学、增强现实、自动全景拼接、虚拟现实、3D 建模、运动估计、视频稳定、运动捕捉、视频处理和场景理解, 它们很难通过深度学习以可微方式轻松实现, 而是需要使用其他「传统」技术。下文介绍了 CV 领域中的一些新兴问题, 在这些问题中深度学习面临新挑战, 而经典 CV 技术能够发挥更大作用。3D 视觉 3D 输入的内存大小比传统的 RGB 图像大得多, 卷积核必须在三维输入空间中执行卷积 (见图 3)。

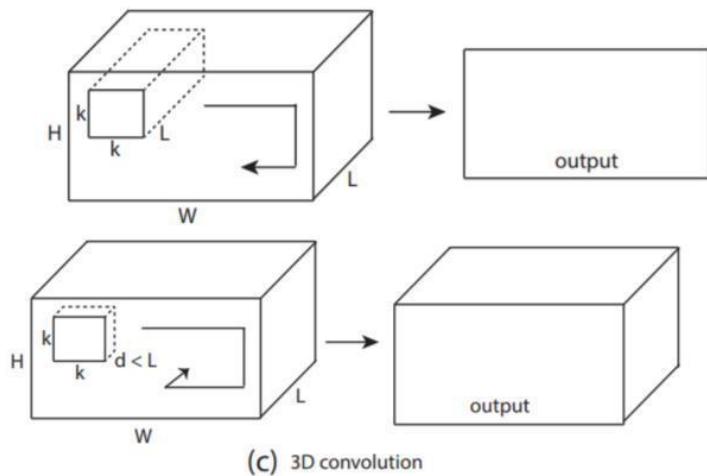


图 11-3 图 3: 2D CNN vs. 3D CNN [47]

因此, 3D CNN 的计算复杂度随着分辨率呈现三次方增长。相比于 2D 图像处理, 3D CV 更难, 因为增加的维度使得不确定性也随之增加, 如遮挡和不同的摄像头角度 (见图 4)。下一节将涉及处理多种 3D 数据表征的解决方案, 这些方法具备新架构和预处理步骤, 专用于解决上述挑战。

11.2 几何深度学习

几何深度学习 (GDL) 将深度学习技术扩展到 3D 数据。3D 数据的表征方式多种多样, 总体上可分为欧几里得和非欧几里得。3D 欧几里得结构化数据具备底层网格结构, 允许全局参数化, 此外, 它还具备和 2D 图像相同的坐标系统。这使得现有的 2D 深度学习范式和 2D CNN 可应用于 3D 数据。3D 欧几里得数据更适合通过基于体素的方法分析简单的刚性物体, 如椅子、飞机等。另一方面, 3D 非欧几里得数据不具备网格数组结构, 即不允许全局参数化。因此, 将经典深度学习技术扩展到此类表征是非常难的任务, 近期 [52] 提出的 Pointnet 解决了这个难题。对目标识别有用的连续形状信息常常在转换为体素表征的过程中丢失。使用传统 CV 算法, [53] 提出可应用于体素 CNN (voxel CNN)

的一维特征。这种基于平均曲率的新型旋转不变特征提升了体素 CNN 的形状识别性能。该方法应用到当前最优的体素 CNN Octnet 架构时取得了极大成功, 它在 ModelNet10 数据集上取得了 1% 的整体准确率提升。

SLAM

视觉 SLAM 是 SLAM 的子集, 它使用视觉系统 (而非激光雷达) 登记场景中的路标。视觉 SLAM 具备摄影测量的优势 (丰富的视觉数据、低成本、轻量级和低能耗), 且没有后处理通常需要的繁重计算工作负载。视觉 SLAM 包含环境感知、数据匹配、运动估计、位置更新和新路标登记等步骤。对在不同条件 (如 3D 旋转、缩放、光照) 中出现的视觉对象建模, 以及使用强大的迁移学习技术扩展表征以实现 zero/one shot learning, 是一道难题。特征提取和数据表征方法可以有效地减少机器学习模型所需的训练样本数量。图像定位中常使用一种两步方法: 位置识别 + 姿势估计。前者使用词袋方法, 通过累积局部图像描述子 (如 SIFT) 来计算每个图像的全局描述子。每个全局描述子均被存储在数据库中, 一同存储的还有生成 3D 点云基准图的摄像头姿势。从 query 图像中提取出类似的全局描述子, 数据库中最接近的全局描述子可以通过高效搜索检索出来。最接近全局描述子的摄像头姿势可以帮助我们对 query 图像进行粗略定位。在姿势估计中, 使用 Perspective-n-Point (PnP) [13] 和几何验证等算法更准确地计算 query 图像的确切姿势。基于图像的位置识别的成功很大程度上归功于提取图像特征描述子的能力。不幸的是, 在对激光雷达扫描图像执行局部特征提取时, 没有性能堪比 SIFT 的算法。3D 场景由 3D 点和数据库图像构成。一种方法是将每个 3D 点与一组 SIFT 描述子结合起来, 描述子对应该点被三角化的图像特征。然后将这些描述子平均为一个 SIFT 描述子, 来描述该点的外观。另一种方法基于 RGB-D 数据构建多模态特征, 而不是深度处理。至于深度处理部分, 研究者采用基于表面法线的着色方法, 因为它对多种任务有效且具备稳健性。另一种使用传统 CV 技术的替代方法提出基于图的层级描述子 Force Histogram Decomposition (FHD), 它可以定义对象的成对结构化子部分之间的空间关系和形状信息。该学习步骤的优势是与传统词袋框架兼容, 从而出现结合了结构特征和局部特征的混合表征。

360 度摄像头

由于球面摄像头的成像特点, 每张图像都能够捕捉到 360 度全景场景, 消除了对转向选择的限制。球面图像面临的一个主要挑战是超广角鱼眼镜头导致的严重桶形畸变, 这增加了受传统人类视觉启发的车道检测和轨迹追踪等方法的实现复杂度。这通常需要额外的预处理步骤, 如先验校准 (prior calibration) 和 deworming。[60] 提出的一种替代方法将导航看作分类问题, 从而绕过了预处理步骤, 该方法基于原始未校准球面图像找出最优潜在路径方向。全景拼接是该领域的另一个开放性问题。实时拼接方法 [61] 使用一组可变形网格和最终图像, 并结合利用稳健像素着色器的输入。另一种方法 [62] 将几

何推理（线和消失点）提供的准确率和深度学习技术（边和法线图）实现的更高级数据提取和模式识别结合起来，为室内场景提取结构化数据，并生成布局假设。在稀疏结构化场景中，由于缺乏明显的图像特征，基于特征的图像配准方法通常会失败。这时可使用直接的图像配准方法，如基于相位相关的图像配准算法。[\[23\]](#) 研究了基于判别相关滤波器（DCF）的图像配准技术，证明基于 DCF 的方法优于基于相位相关的方法。

数据集标注和增强

对于 CV 和深度学习的结合存在一些反驳意见，总结为一句话就是：我们需要重新评估方法，不管是基于规则的方法还是数据驱动方法。从信号处理的传统角度来看，我们了解传统 CV 算法（如 SIFT 和 SURF）的运算内涵，而深度学习无法展示这些意义，你所需要的只是更多数据。这可以被视为巨大的前进，但也有可能是后退。本论文提到了该争论的正反方观点，但是如果未来的方法仅基于数据驱动，那么研究重点应该放在更智能的数据集创建方法上。当前研究的基础问题是：对于特殊应用的高级算法或模型，没有足够的数据。未来，结合自定义数据集和深度学习模型将成为很多研究论文的主题。因此研究者的输出不仅涉及算法或架构，还包括数据集或数据收集方法。数据集标注是深度学习工作流中的主要瓶颈，需要大量的手动标注工作。这在语义分割中尤为明显，因为该领域需要准确标注每一个像素。[\[20\]](#) 讨论了很多有用的半自动流程工具，其中一些利用了 ORB 特征、多边形变形（polygon morphing）、半自动感兴趣区域拟合等算法方法。克服数据缺乏、减少图像分类深度学习模型过拟合现象最容易也最常见的方法是，利用标签不变的图像变换（label-preserving transformation）人为地扩大数据集。该过程叫做数据集增强，指基于已有数据通过剪裁、缩放或旋转等方式生成额外的训练数据。人们希望数据增强步骤需要极少的计算，且可在深度学习训练流程中实现，这样变换后的图像就不必存储在磁盘中了。数据增强使用的传统算法方法包括主成分分析（PCA）、噪声添加、在特征空间的样本之间进行内插或外推，以及基于分割标注建模视觉语境周边物体。

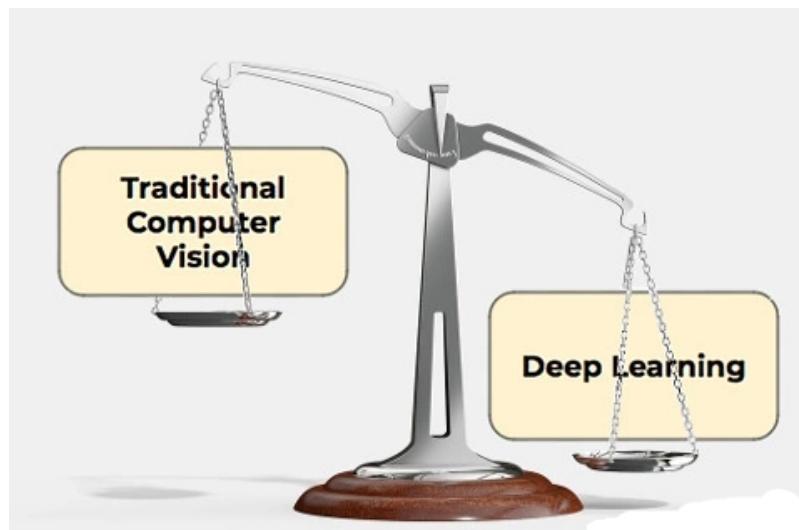


图 11-4 深度学习是否可以取代传统的计算机视觉

在进行深度学习之前,如果你有诸如图像分类之类的任务,这时你需要执行一个称为特征提取的步骤,特征提取是非常“有趣的”。我这篇文章中将要提到一些传统的计算机视觉技术(包括诸如边缘检测,角点检测,物体检测等等)。在使用这些技术时,例如在特征提取和图像分类方面,我们想的是从一类对象(例如椅子,马等)的图像中提取尽可能多的特征,并将这些特征视为一种“定义”(被称为“袋”的对象)。然后,你会在其他图像中搜索这些“定义”。如果一个袋子中的大量特征位于另一个图像中,则该图像被分类为包含该特定对象(即椅子,马等)。这种图像分类特征提取方法的难点在于,你必须选择在每个给定图像中查找哪些特征。当你尝试分类的类别数量开始增加,例如 10 或 20 时,这会变得很麻烦并且变得几乎不可能。你是否寻找边缘?纹理信息?使用不同类型的功能可以更好地描述不同类别的对象。如果你选择使用许多特征,则必须处理大量参数,所有这些参数都必须由你进行微调。那么,深度学习介绍了端到端的学习概念,其中(简而言之)机器被告知要针对每个特定类别的对象学习要寻找什么。它为每个对象提供了最具描述性和显着的特征。换句话说,神经网络已经被告知发现图像类别中的底层模式。因此,通过端到端的学习,你不再需要手动决定使用传统计算机视觉技术来描述你的特征。有线杂志这样说道:例如,如果你想教一个神经网络来识别一只猫,那么你不要告诉它寻找胡须,耳朵,毛皮和眼睛。你只需要展示成千上万张猫的照片,最终就能解决问题。如果它将狐狸误分类为猫,你不需要重写代码,你只需要做的是继续训练。下面的图片描绘了特征提取(使用传统的方法)和端到端学习之间的差异:

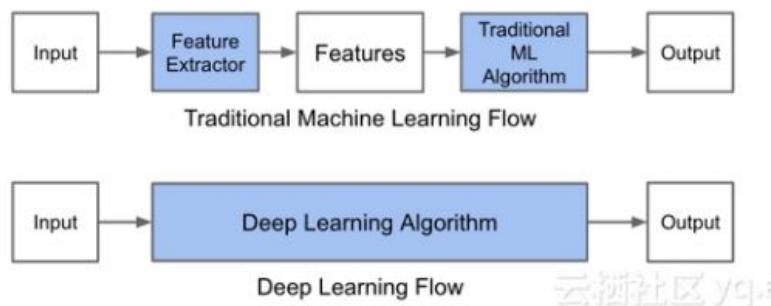


图 11-5 特征提取和端到端学习之间的差异

所以,这是整篇文章的背景。接下来,让我们来看看为什么传统的计算机视觉仍然是必要的,有益的。深度学习需要大数据首先,深度学习需要数据,很多很多的数据。上面提到的那些著名的图像分类模型都是在大数据集上进行训练的,这些用于训练的数据集的前三名是:

ImageNet——包含 1000 个对象类别/类的 150 万个图像。上下文中的 Microsoft 通用对象 (COCO) ——250 万个图像,91 个对象类别。PASCAL VOC 数据集——500K 图像,20 个对象类别。

比一般图像分类更容易的任务不需要这么多的数据,但你仍然需要很多数据。如果你无法获得那么多的数据,你根本不知道会发生什么? (确实也有一些技巧可以提高你的训练数据量,但这些是人为的方法)。没有充足的数据,训练出来的模型一般表现都不好,因为一台机器没有洞察能力,它不能在没有看到数据的情况下概括它看到的东西。对于你来说,看到训练好的模型并且手动调整一些东西太困难了,因为深度学习模型里面有数百万个参数,其中每个参数在训练过程中都会被调整。从某种意义上说,深度学习模式是一个黑匣子。传统的计算机视觉为你提供了充分的透明度,使你能够更好地评估和判断你的解决方案是否可以在训练环境之外进行工作。你可以深入了解算法中存在的问题,如果有任何不妥,你可以很容易地弄清楚在哪里以及需要调整什么。深度学习有时会发生过度拟合:这可能是我支持传统计算机视觉技术研究的最佳理由。训练深度神经网络需要很长时间,你需要专用硬件 (例如,高性能 GPU),在很长的时间内训练最新的最先进的图像分类模型。此外,如果你的训练模型表现不佳,会发生什么? 你必须返回并用不同的训练参数重做整个过程,而且这个过程有时可能重复数百次。但有时候这些都是不必要的,因为有时传统的 CV 技术可以比 DL 更有效地解决问题,并且代码行数更少。例如,我曾经参与过一个项目,以检测通过传送带的每个锡罐是否有红色的勺子。现在,你可以训练一个深度神经网络来检测勺子,或者你可以对红色上编写简单的颜色阈值算法 (红色的某个范围内的任何像素都是白色的,每个其他像素是黑色的),然后计算你有

多少白色像素。了解传统的计算机视觉可能会为你节省大量时间和减少一些不必要的麻烦。传统的计算机视觉将提高你的深度学习技能：理解传统的计算机视觉实际上可以帮助你更好地进行深度学习。例如，计算机视觉中使用的最常见的神经网络是卷积神经网络。但什么是卷积？它实际上是一种广泛使用的图像处理技术（例如参见 Sobel 边缘检测）。了解这可以帮助你了解你的神经网络做了什么，因此可以更好地设计和调整你尝试解决的任务。然后还有一件事叫做预处理。这是经常对你提供的模型的数据进行准备以进行训练。这些预处理步骤主要通过传统的计算机视觉技术来完成。例如，如果你没有足够的训练数据，则可以执行称为数据增加的任务。数据增加可以包括对训练集中的图像执行随机旋转，移位，剪切等，以创建“新”图像。通过执行这些计算机视觉操作，你可以大大增加你拥有的训练数据量。结论：在这篇文章中，我解释了为什么深度学习没有取代传统的计算机视觉技术，为什么后者仍应该学习。首先，我发现了 DL 经常需要大量数据才能执行的问题。其次，深度学习对于特定任务来说可能会出现过度拟合现象。在这样的任务中，标准的计算机视觉可以比 DL 更有效地解决问题，并且代码行数更少。第三，认识传统的计算机视觉实际上可以让你更好地进行深度学习。这是因为你可以更好地了解 DL 到底正在做什么，并且你可以执行某些预处理步骤来改善 DL 结果。简而言之，深度学习只是计算机视觉的工具，当然不是万能药。不要只用它，因为它现在是新潮。传统的计算机视觉技术仍然非常有用，知道它们可以为你节省时间和解决许多麻烦。本文由阿里云云栖社区组织翻译。文章原标题《Why Deep Learning Has Not Superseded Traditional Computer Vision》作者：Zbigniew

北京大学信息科学技术学院教授黄铁军，则基于自己的最新研究成果，带来了主题为《视达（vidar）：视觉新模型与超级视觉》的报告。

北京大学信息科学技术学院教授黄铁军先提出了一个问题：人为什么躲不过子弹？原因是：人的眼睛往大脑传送神经脉冲的速度比较慢，而且每秒只能传几十个脉冲，所以，当一颗子弹飞过来时，眼睛根本来不及向大脑传输信号。基于此，他较为尖锐地指出，现在 90% 做计算机视觉的研究者根本还没搞明白视觉到底是什么，因而现在用摄像头采集视频 + 算法的技术研究路线和思维方式从根本上来说都是错误的。并且，用视频作为视觉信息的表达方式的这一起点，就是错的。视频实际上是影视产业发展的产物，它利用的其实就是人类视觉系统的缺陷——视觉暂留，来给人类以连续的画面感，而生物视觉则是视网膜接收连续的光子撞击，再由神经节细胞接收到足够刺激后发放脉冲，接着脉冲序列被传送给大脑，最后大脑从脉冲序列中解码出外部世界，因此，要真正实现机器智能，就必须放弃视频这一概念，用新的思路去做研究：了解从眼睛到大脑的神经脉冲是如何编码外界的视觉信息的。基于这一思路，黄铁军教授在视觉信息处理方面最近开展了一个工作，便是模仿从视网膜到大脑的神经脉冲对外界视觉信息进行编码。他将这项工

作称之为视达（vidar）。

计算机视觉的研究者应该彻底改变用摄像头 + 算法的研究思路：

第一，不再用以前的识别摄像机，而要用新的视达芯片和摄像机来抓取过程；

第二，不要在计算机上编算法，而要在脉冲神经网络做脉冲序列。

山世光研究员指出，随着人脸识别已经取得了非常大的进展，接下来将会从「看脸」时代转向「读心」的时代，从依赖强大的数据的学习算法转向弱监督小规模数据驱动的视觉学习算法。因此，打造有温度、有温情的 AI 系统，非常重要。

对于「读心」，他认为可以从三个层次来开展工作：

第一，测量生理性特征，比如身高、体重、心率、呼吸、血压、血氧、眨眼率、视线、瞳孔缩放、皮肤状况、醉酒状态等等。

第二，评估心理状态。表情很多时候是虚假的，而微表情则更能体现人试图隐藏的内心情绪。这些心理状态的评估可以应用到学习、驾驶、健康等方面。

第三，通过观察生理特征和心理状态，来评估人的内心精神状况，包括是否抑郁、幸福等精神层面的辅助诊断。

然而在具体的技术实现上，「读心」首先就面临着是数据稀缺的挑战，比如针对自闭症儿童的数据，有个几十人就已经算多的了。那么如何在非常小规模的数据环境下做好机器学习呢？

可以从以下三个方向来克服这个挑战：

第一，自监督学习，思路是针对小样本，尽可能地采用借鉴人的注意机制的方法，找到最值得关注的区域并提取特征，从而即便在小样本情况下也可以实现比较好的视觉特征学习。

第二，弱监督学习，思路是通过引入不同的任务，进行多个任务的协同处理，从而将不同任务的算法进行优势互补，实现弱监督条件下的学习。

第三，半监督学习，思路是协同训练一部分带标签的数据和另一部分没有标签的数据，让二者同时学习多个模型，互相补充和支持，将这些没有标签的数据也用起来。

未来十年，AI 会深刻地改变各行各业，AI 算法也会越来越深刻地了解人类，然而现有的算法还不足以支撑社会各界对 AI 的预期。如何在数据稀缺情况下做机器学习是未来要克服的挑战之一，他今天分享的仅仅是实现标签从无到有、由弱变强、从小到大的思路，而其他的思路，例如如何将领域知识设计成一种通用的算法等，也是值得领域研究者关注和研究的方向。

11.3 作业

探索

思考

