软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.006076]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

机器学习赋能的软件系统自适应性*

张明悦[1,2], 金 芝[1,2], 赵海燕[1,2], 罗懿行[1,2]

[1](北京大学 信息科学技术学院计算机科学与技术系,北京 100871)

[2](高可信软件技术教育部重点实验室(北京大学),北京 100871)

通讯作者: 金芝, E-mail: zhijin@pku.edu.cn

摘 要: 软件系统自适应机制提供了应对动态变化的环境和不确定的需求的技术方案.在已有的软件系统自适应性的相关研究中,有一类工作将软件系统自适应性转换为回归、分类、聚类、决策等问题,并利用强化学习、神经网络/深度学习、贝叶斯决策理论和概率图模型、规则学习等进行问题建模,并以此构造软件系统自适应机制.本文通过系统化的文献调研,综述了机器学习赋能的软件系统自适应性的工作.首先介绍基本概念,然后从不同视角对当前工作进行分类;按被控系统、监测和控制过程、学习算法、学习赋能方式等方面进行分析,并讨论不同机器学习方法赋能的软件系统自适应性的切入点及其优势和不足;最后对未来研究进行展望.

关键词: 自适应软件系统;软件系统自适应性;机器学习;不确定需求;环境动态性

中图法分类号: TP311

中文引用格式: 张明悦,金芝,赵海燕,罗懿行.基于机器学习的软件自适应性.软件学报. http://www.jos.org.cn/1000-9825/6076.htm

英文引用格式: ZHANG Ming-Yue, JIN Zhi, ZHAO Hai-Yan, LUO Yi-Xing. Survey of Machine Learning Enabled Software Self-Adaptation. Ruan Jian Xue Bao/Journal of Software, (in Chinese). http://www.jos.org.cn/1000-9825/6076.htm

Survey of Machine Learning Enabled Software Self-Adaptation

ZHANG Ming-Yue[1,2], JIN Zhi[1,2], ZHAO Hai-Yan[1,2], LUO Yi-Xing[1,2]

[1](Department of Computer Science and Technology, School of EECS, Peking University, Beijing 100871, China)

[2](Key Lab of High Confidence Software Technologies Ministry of Education (Peking University), Beijing 100871, China)

Abstract: Software self-adaptation (SSA) provides a way of dealing with dynamic environment and uncertain requirement. There are some literature that transform the dynamic and uncertainty concerned by SSA into regression, classification, cluster, or decision task; and apply machine learning algorithms, including reinforcement learning, neural network/deep learning, Bayesian decision theory and probabilistic graphical model, rule learning, to SSA. We call these kinds of work “Machine learning based software self-adaptation” in this paper. We conduct a survey on the state-of-the-art research about machine learning enabled SSA by firstly explaining the related concepts of SSA and machine learning; and then proposing a taxonomy based on current work from SSA perspective and machine learning perspective respectively; analyzing the controlled system, monitor process, control process, learning algorithm, learning-enabled method under this taxonomy; as well as identifying finally deficiency of current work and highlighting future research trends.

Key words: self-adaptive software; software self-adaptation; machine learning; requirement uncertainty; environment dynamic

软件系统自适应性(software self-adaptation),指系统能够在运行时(run-time)根据其对环境和系统本身的感知,自主地调整自身的行为,以有效地应对动态变化的环境和不确定的需求,从而使系统具有更好的灵活性、可靠性、鲁棒性等[1].其核心是构造自适应策略,策略的构造方式有三: (1)设计者根据具体场景和具体问题,通过定

* 基金项目: 国家自然科学基金(61620106007, 61751210)
Foundation item: National Natural Science Foundation of China (61620106007, 61751210)
收稿时间: 2019-10-15; 修改时间: 2020-04-04; 采用时间: 2020-05-07; jos 在线出版时间: 2020-05-26

义一组规则作为自适应策略;(2)基于控制论的思想将软件系统自适应策略显式地建模为“感知(sense)-规划(plan)-执行(act)”的控制回路;(3)将软件自适应策略的构造转换为机器学习任务,通过学习算法得到自适应策略.目前基于控制论的方法比较主流,已有文献对其进行了系统性的综述[4,63].

通过机器学习构建软件系统的自适应策略是一种有效手段,其原理是将软件系统自适应行为转换为分类、回归、聚类、决策等机器学习擅长的问题,利用监督学习、无监督学习、强化学习、贝叶斯网络等技术,获取自适应策略.其优点是:(1)在预先缺少领域知识(domain knowledge)的情况下,学习出满足性能要求的自适应策略;(2)借助高效的算法来解决软件系统自适应行为中隐含的搜索、最优化和不确定性分析等问题[2].目前已有不少工作研究机器学习赋能的软件系统自适应性,它们各自有着不同的关注点,解决不同的自适应问题.有必要对这些工作进行梳理,以定位仍存在的问题和潜在的挑战,并展望今后的研究方向.本文首先提出用于分析已有工作的三个研究问题,并以此为基础梳理相关工作.文献检索的过程为,首先在常用论文库中,根据一组关键词检索相关研究;然后人工去除和研究问题无关的论文,并根据筛选后论文的参考文献、引证文献、作者论文列表等补充首轮搜集时遗漏的论文;最终,选出与研究问题高度相关的论文.

本文关注的三个问题是:

(1) RQ1.现有机器学习赋能的软件系统自适应性研究有哪些关注点,如何分别从自适应性和机器学习角度进行分类?目的是建立系统化的分类框架,以概括相关工作所关注的话题和涉及的方向.

(2) RQ2.当前软件系统自适应任务的哪些方面采用了哪些机器学习技术?目的是在RQ1所概括的分类框架下,建立机器学习技术和软件系统自适应任务的关联.

(3) RQ3.当前基于机器学习赋能的软件系统自适应性还存在哪些不足?未来发展趋势怎样?目的是在已有研究的基础上,展望其发展趋势,为进一步的研究提供参考.

本文第1节介绍相关的背景知识.第2节给出本文采用的综述方法以及获取、筛选相关工作的标准与过程.第3节提出分类框架,并对现有工作进行分析与研究.第4节讨论机器学习赋能的软件系统自适应性存在的问题,并展望未来的研究趋势.第5节对全文进行总结.

1 背景知识

1.1 系统自适应性和自适应系统

现代软件系统日趋复杂,其交互环境和需求都可能动态变化并具有不确定性,这要求软件系统具有自适应性,以应对运行场景的复杂性、动态变化性和不确定性.从概念上说,自适应性是指系统能够在运行时根据对环境和系统本身的感知,自主地调整自身行为[1].“软件系统自适应性”和“自适应软件系统”这两个词常常混用,但在含义上略有所差别,前者侧重表达软件系统的自适应能力,后者侧重表达具有自适应性的软件系统.

就自适应软件系统设计而言,其核心是软件系统的自适应策略的设计.最朴素的手段是依靠经验定义一组形为“条件-动作”的规则,表达在某种“条件”下采取某种“动作”(即ad hoc方法).相当多的工作引入控制论方法,基于经典控制论[10,11,12]、现代控制论[13,14,15]、智能控制论[16,17,18]等建立基于控制回路的自适应软件系统架构.在控制论的观点下,自适应系统分为控制器和被控对象两部分,控制器实现自适应逻辑,被控对象实现应用逻辑.自适应逻辑通过感知器感知环境和应用逻辑的变化,通过执行器对应用逻辑做出调整,如图1所示.自适应逻辑多由“收集(collect)-分析(analyze)-决策(decide)-动作(act)”的过程来实现,用前馈(feedforward)或反馈(feedback)的方式进行应用逻辑的调控[6].MAPE-K[7]是一个典型的自适应软件系统体系结构.除MAPE-K外,以反馈控制回路为基础设计的自适应软件系统架构还有Rainbow[8],MRAS[9]等.

近年来,很多工作将机器学习引入到自适应软件系统中,比如把机器学习算法应用于软件系统自适应行为的部分过程中[19,20,21],或者采用机器学习算法实现软件系统的自适应输入/输出逻辑[22, 23, 24]等,以提升自适应系统的自治性和自演化能力,这开始成为一个热点研究方向.

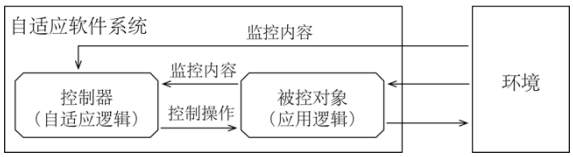


Fig.1 Constituent parts of a self-adaptive system: control theory perspective

图 1 自适应系统:基于控制论的观点

1.2 用于解决系统自适应问题的机器学习算法

机器学习“研究让算法利用经验来改善系统自身性能”,Mitchell 给出机器学习的定义[25]:存在某类任务 T,P 为程序在 T 上的性能,如果一个程序利用经验 E 在 T 上改善了 P,则说,关于 T 和 P,该程序对 E 进行了学习.在软件系统自适应性问题上引入机器学习方法的切入点是,这类问题或者其子问题可以转换为分类、回归、聚类、决策等问题,而这些是机器学习算法擅长解决的问题.目前,强化学习、深度神经网络、贝叶斯决策理论、概论图模型、规则学习、迁移学习等均出现在软件系统自适应性的研究工作中.

(1)强化学习:学习如何将环境状态映射到系统动作,以获得最大奖励[26].基本模型是马尔可夫决策过程(Markov Decision Process,MDP),它定义为四元组<S,A,T,r>,其中 S 是环境状态集合,A 是系统动作集合,T 是状态转移函数,$\mathrm{T}: \mathrm{S} \times \mathrm{A} \times \mathrm{S} \rightarrow [0,1]$, $r: \mathrm{S} \times \mathrm{A} \rightarrow$是奖励函数,$\mathbb{R}$ 为实数集合,$r(s,a)$表示在状态s下采取a动作后能获得的奖励.系统行为策略$\pi: S \times A \rightarrow [0,1]$表示系统对环境的反应,表示在状态 s 下采取 a 动作的概率.系统在与环境交互过程中获得累积折扣奖励为$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$.在策略 π 下,状态价值函数为,$v_\pi(\mathrm{s}) = E_\pi[G_t | S_t = s]$,状态动作价值函数为,$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a]$.强化学习的常用策略是使累积折扣奖励最大化,典型算法有 Q-learning[26]、SARSA[26]等.

软件系统自适应性采用强化学习的原因是:(a)强化学习中系统和环境交互的机制,可对应于自适应软件系统的反馈回路[23];(b)强化学习的“探索-利用”(exploration-exploitation)思路,对自适应软件系统策略演化[35]有指导意义;(c)强化学习可以在没有先验知识的情况下,通过系统和环境的交互学到最佳策略[34].但强化学习在实际应用中却存在诸如学习效率低,随机探索易对系统或环境造成损害等不足[34,36,37].

(2)神经网络和深度学习:神经网络是由具有适应性的简单单元组成的并行互连网络[27].深度学习则利用包含多个隐藏层的人工神经网络提高特征学习能力[27],如多层感知器(Multilayer Perceptron,MLP)、卷积神经网络(Convolutional Neural Network,CNN)、循环神经网络(Recurrent Neural Network,RNN)等.软件系统自适应性研究利用深度学习,是因为其中的性能建模[48,54,55]、环境预测[50,52,56]、自适应调整[49,51,53]等任务可以转化为回归问题,一般情况下,如果能获得软件系统的相关原始数据(即“生数据”),则可以采用深度学习模型对上述问题进行求解.还可以和控制论结合,用神经网络控制(neural network control)实现对系统状态的控制,以便在外界发生变化时,软件系统能自动调整到设定的运行状态[53,59].深度强化学习在强化学习的基础上结合深度学习的模型,既有强化学习处理决策问题的能力,又有深度学习对高维数据的感知能力,通常可直接根据对环境的观测信息(如图像、语音或者其他传感器信号)进行学习,获得最优策略.常用的深度学习算法包括深度 Q 网络(Deep Q-networks, DQN)[28]、DDPG[76]、TRPO 算法[29]等.

(3)贝叶斯决策理论和概率图模型:研究在不完全信息下变量的概率以及多个变量间的概率依赖关系等,关注不确定性的建模.贝叶斯决策理论在概率框架下实施最优决策[30].具有不确定性的多分类任务指:对 N 种可能的类别$\mathrm{Y} = \{c_1, c_2, \ldots, c_N\}$, $\lambda_{i,j}$是将c_j类的样本划分为c_i所带来的损失,$P(c_i|x)$为将样本x划分为c_i类的后验概率,则样本 x 的条件风险(conditional risk)为$\mathrm{R}(c_i|x) = \sum_{j=1}^{N} \lambda_{i,j} P(c_j|x)$.用判定准则$h: \mathrm{X} \rightarrow \mathrm{Y}$来最小化条件风险 $\mathrm{R}(h(x)|x)$,即:$h^*(x) = \underset{c \in Y}{arg\min} R(c|x)$.相关研究包括贝叶斯分类器、贝叶斯网络、EM 算法等.

概率图模型用图表达变量间的关系,把学习任务归结成对变量概率分布的计算[31] ,分为两类:有向图模型和无向图模型.隐马尔可夫模型(Hidden Markov Model, HMM)是典型的有向图模型,其变量的联合概率分布为:$\mathrm{P}(x_1, y_1, \ldots, x_n, y_n) = \mathrm{P}(y_1)\mathrm{P}(x_1|y_1) \prod_{i=2}^{n} \mathrm{P}(y_i|y_{i-1})\mathrm{P}(x_i|y_i)$.其中$y_i$是状态变量,为不可被直接观测量,$x_i$是观

测值.马尔可夫随机场(Markov Random Field, MRF)是典型的无向图模型,其变量的联合概率为:$P\left(\vec{x}\right)=(1/Z)\prod_{Q\in C}\varphi_Q\left(\vec{x}_Q\right)$,其中$\vec{x}=\{x_1,x_2,\ldots,x_n\}$,$Z$ 为规范化因子,φ_Q为势函数.在系统自适应性研究中,这类算法在概率模型的基础上,进行最优化决策.

目前这类工作的应用场景包括: (a) 贝叶斯网络用于表达和分析环境或系统中和自适应相关的不同随机变量的关系;(b) 隐马尔可夫链用来建模存在不确定性时的系统运行状态; (c) 贝叶斯分类器或条件随机场用于刻画自适应系统的需求获取和系统设计时的不确定性; (d) 马尔可夫过程(Markov decision process,MDP)用于支持含不确定性的自适应策略的生成[64,65,66],也有工作改进强化学习,并加入高斯过程,来应对存在不确定性时的自适应决策[64,65].

(4)规则学习:“规则”指语义明确且能描述蕴含于数据分布背后的规律或者领域知识的逻辑规则[32].规则的形式为:$r\leftarrow f_1\wedge f_2\wedge\ldots\wedge f_n$.规则学习根据正例(E+)和反例(E-)构造规则集,使得规则集可以尽可能多地保留 E+排除 E-[39].规则学习一般采用序贯覆盖(sequential covering)算法[32].包括命题规则学习、一阶规则学习、归纳逻辑程序设计等.在赋能系统自适应性方面,规则学习只能用于自适应逻辑非常明确的情况,但采用规则的自适应软件系统有很好的可验证性,可靠性和可理解性.规则学习还可用于发掘逻辑表达式所构成的系统模型的潜在特性,计算出操作的前置(pre-)和触发(trigger-)条件[39]. 常用工具包括 XHAIL[68]、Progol5[69]、INTHELEX[71]等.

(5)迁移学习:迁移学习常用于数据不足的场景,利用在训练数据充分的问题上训练好的模型,通过迁移操作使模型适合于新的问题[67].假设源域(source domain)数据集D_S和学习任务T_S,目标域(target domain)数据集D_T和学习任务T_T,且$D_S\neq D_T$或$T_S\neq T_T$,迁移学习就是用D_S和T_S的知识来提高预测函数$f_T(\cdot)$的在D_T上的准确率.其实现方法有样本迁移(instance-based-)、特征迁移(feature-based-)、模型迁移(model-based-)、关系迁移(relational-)等.构造自适应系统,通常是离线训练,由于硬件损耗、安全问题,训练数据常常不能直接从真实系统中获取,而只能依赖模拟系统生成的“虚拟数据”[43].迁移学习常用于把在模拟环境中训练出的模型迁移到真实环境中[43,60],以获得可用的策略或加快真实环境中的学习效率.

2 综述方法

参考 Kitchenham[3]的系统性文献综述(Systematic Literature Review,SLR)方法, 本文的文献调研分为 3 个步骤:规划(planning)、实施(conducting)和报告(reporting).其中,规划阶段定义研究问题、确定文献筛选标准和方法等,实施阶段进行文献检索、选择、分类、分析等,报告阶段则将调研的工作进行系统性展示.

机器学习赋能的软件自适应性涉及到机器学习与自适应软件系统两个研究方向,本文以((machine learning) or (learning) or (learn)) and ((adaptation) or (adaptive) or (self-adaptive) or (self-adaptation) or (dynamic) or (uncertainty))为关键词,在 IEEE Xplore Digital Library, ACM Digital Library, Springer Digital Library, Elsevier Science Direct, Google Scholar,中国知网等文献数据库中检索.并采取“滚雪球”的方法:(1)在已获取文献的作者的论文列表中发现新文献;(2)在已搜集文献的参考文献和引证文献中发现新文献;(3)将新发现的文献加入已获取的文献集合中,继续(1)和(2),直到不再发现新文献为止,尽可能多地覆盖已有的相关工作.这些文献需要进一步筛选,筛选标准为:

(1) 文献发表时间在 2003~2019 年之间;
(2) 文献采用了机器学习技术(包括线性模型、深度学习、决策树、支持向量机、贝叶斯决策理论、聚类、规则学习、强化学习、迁移学习等);
(3) 文献的贡献领域为自适应软件系统或软件系统自适应性;
(4) 文献的语言为英文或中文;
(5) 2018 年及以前的文献,其被引次数高于 3.

经过以上步骤检索文献,并按照上述标准进行筛选,最后共收集到 78 篇文献.其中有 60 篇会议论文,15 篇期刊论文,3 本专著.会议论文主要来自于 International Conference on Software Engineering(ICSE),International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS),International Conference on Service-Oriented Computing (ICSOC), International Conference on Autonomic Computing (ICAC),International Conference on Self-Adaptive and Self-Organizing Systems (SASO)等.其中 SASO 与 ICAC 于 2020 年合并为 International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS).期刊论文主要来自于 IEEE transactions on software engineering 等.

3 分类框架

本节回答研究问题 RQ1,提出对现有文献的分类标准,分软件系统自适应性和机器学习两个视角进行考察,软件系统自适应性视角重点在自适应软件系统的体系结构、行为、建模等方面,机器学习视角侧重于从任务出发,分析如何将软件系统自适应性的问题规约到机器学习任务.

3.1 系统自适应性视角

软件系统自适应性可以从三个视角进行分类:外部特性视角、系统结构视角和功能部件视角:

外部特性视角. 关注软件系统的输入/输出特性.输入是监控信息,包括物理数据和应用需求.物理数据指软件系统运行过程中可感知的客观数据,来自于外部环境或系统自身.输出是软件系统的自适应操作,按粒度从细到粗可分为参数调整,行为调整,结构调整等,该层的结构如图 2 所示.

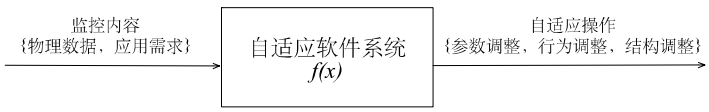


Fig. 2 External characteristic layer of software self-adaptation

图 2 自适应系统外部特性层

系统结构视角.关注软件系统的控制器和被控系统,以及与环境间的交互,用控制回路来表示.如图 3 所示,对控制回路而言,应用需求作为输入,控制器根据应用需求和监控数据进行决策做出自适应操作,改变后的被控系统作用到环境中,并向控制器返回系统自身数据.这里,控制器实现自适应逻辑,决定系统的外部特性;被控对象是自适应逻辑的受体,其行为受控制器的控制.常见的被控对象包括面向服务的系统(如云服务,Web 服务等)[19]、自主无人系统(如机器人,无人驾驶汽车,无人飞行器等)[23]、信息物理融合系统(如智能城市,智能家居等)[39]等.

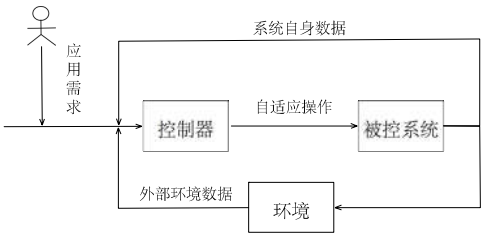


Fig.3 Architecture layer of software self-adaptation

图 3 自适应系统结构层次

大部分研究侧重“自适应逻辑”,即控制器的设计.关注自适应目标、原因、和决策方式.保持系统稳态和最优化需求目标是两种典型的自适应目标.根据需求目标是否可变,后者又可分为最优化可变目标和最优化不可变目标.自适应原因来自于动态性(dynamic)和不确定性(uncertainty),前者包括外部环境变化、系统自身变化、以及其他系统的变化.虽然其他系统可看作外部环境,但外部环境变化和其他系统的变化有较大差异,比如,外

部环境的动态性一般是客观现象,不会因为与系统的交互而改变;而其他系统的动态性,有可能会由于系统间的交互而发生改变[40,41,42].不确定性包括需求时不确定性、设计时不确定性和运行时不确定性[38].决策方式分为反应/前摄式和连续控制式.反应/前摄式指采用离散化的方式,根据某种触发条件做出调整.连续控制式是指采用控制函数,持续地监控数据并进行调整.在已有的基于机器学习方法的自适应性研究中,反应/前摄式通常采用规则/策略来实现.根据运行时策略是否发生变化,可分为静态策略和动态策略,实现静态策略的方法称为"策略生成"(rule generation),实现动态策略的方法称为"策略演化"(rule evolution)[35].

功能部件视角.将系统结构层中的控制器和被控系统展开为具体的功能部件.大多数工作基于 IBM 的 MAPE-K 框架[7],即"监控-分析-规划-执行"过程形成的闭环控制结构,如图 4 所示.监控过程收集相关环境数据,分析过程分析这些数据并做出是否需要适应调整的决定,规划过程根据自适应逻辑,制定调整方案,执行过程实施调整操作.其中,每个过程都在知识库(knowledge base)的指导下完成,其中可以包含规则/策略、案例(case)以及控制函数等.图 4 是 MAPE-K 基本框架,一些文献[8,9,43,44]还针对不同被控系统提出了不同的自适应框架.

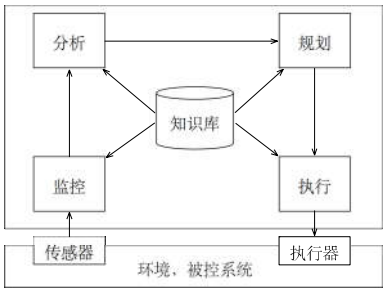


Fig. 4 Functional component layer MAPE-K control loop of software self-adaptation

图 4 自适应功能部件层次 MAPE-K 控制回路

据此,可以根据已有工作构建出机器学习赋能的软件自适应性的特征模型,如图 5 所示.

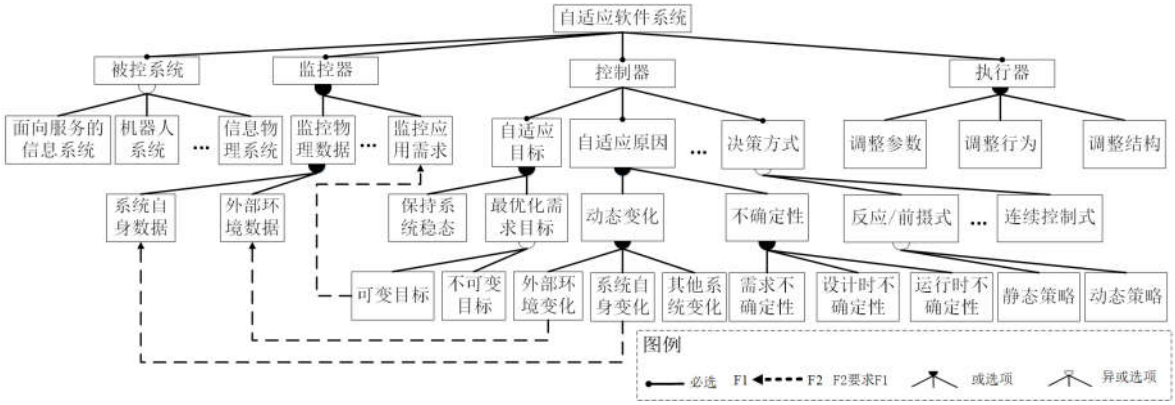


Fig.5 Feature model of self-adaptive software system

图 5 自适应软件系统特征模型

从自适应逻辑实现的角度,可以从如下两个方面来分类: 一是实现自适应逻辑的方法,二是自适应逻辑的构造方式.其中,实现自适应逻辑的方法分为规则/策略式和控制函数式两种.规则/策略式将自适应逻辑表达为事件或状态到动作的映射,描述了软件系统在某个条件下应当采取某种措施.传统的规则/策略式一般用 PDL 策略描述语言[45].策略由 Event-Condition-Action(ECA)规则范式定义:event E action A if condition C,表示当发生事件 T 且满足条件 C,则执行动作 A.有时也简化为 Condition-Action(CA)规则范式[47].控制函数式则采用 PID

(proportion-integral-differential)控制器实现自适应结构中的控制回路[46].

构造自适应逻辑的方法有人工设计和自动设计两种.人工设计主要根据专家经验来定义规则.自动设计采用学习算法,先获得自适应软件系统在需求、设计、运行时的相关数据,提取出任务、经验、性能,再利用学习算法获得规则/策略或控制函数.

除了上述三个层次和两个维度,验证评价也是一个重要问题.验证评价指对方法的效果以及系统的运行性能进行评价分析.主要方法有理论证明、案例研究(case study)、模拟仿真(simulation).目前理论分析的工作几乎没有,大部分验证评价都是采用后两种方法.案例研究指搭建基于特定场景的实验系统,并进行实验评估.模拟仿真则借助于仿真工具对自适应算法进行评估.

3.2 面向自适应目标的机器学习任务视角

软件系统自适应性本质上是应对不确定性,含场景不确定性,环境和系统的动态变化性,效果的不确定性.这三类不确定性分别为机器学习方法限定了不同的问题域(problem domain).下面将根据面向目标的方法对此进行分析,从问题域出发,划分出不同的任务,最终到具体的机器学习技术.

场景的不确定性:系统设计时,对环境或系统的知识掌握不完整或存在不一致,难以确定其环境变量的取值或系统的配置,这是产生自适应需求的根本原因.这类不确定性包括需求缺失、需求二义性、需求改变、环境假设不准确或错误、传感器误差或错误、监控信息不完整等.

系统和环境的动态变化性:系统运行时,其运行环境、交互环境或系统自身的状态都会发生动态改变,这是产生系统自适应需求的运行时原因,具体包括环境特性改变、环境行为改变、系统结构改变、系统行为改变、系统属性改变等.

操作效果的不确定性:系统运行时,由于操作延迟、假设错误、外界干扰等,导致系统行为的实际结果和预期结果不一致.

如果以“系统自适应性（G)”作为根目标,根据上述三个关注点,可分解出三个子目标,即“应对场景的不确定性（G1)”,“应对系统和环境的动态变化性（G2)”,和“应对操作效果的不确定性（G3)”,向下延伸构建系统自适应性的目标模型,进而考察机器学习在其中发挥的作用.如图 6 所示,各个子目标的含义见表 1.

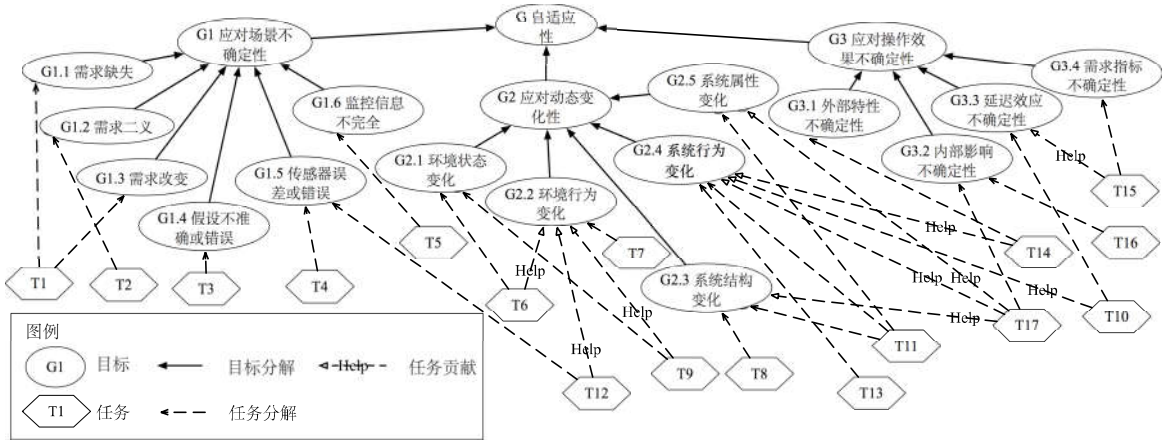


Fig.6 Goal model from machine learning perspective

图 6 机器学习赋能的自适应目标模型

根据系统自适应目标模型,对已有文献的工作进行任务分类,可以提取出 17 类不同任务(T1-T17),这些任务和目标的支撑关系如图 6 所示,任务的出处和采用的方法见表 2.根据所解决问题的特征,17 类任务可分为两大类,分别是端到端的方法和阶段/过程方法,前者从系统外部特性入手.后者从系统内部逻辑入手.具体分析如下:

(1)端到端的方法.考虑系统的输入/输出的外部特性,忽略系统内部各功能部件,通过机器学习算法,得到满

足自适应目标的外部特性的自适应策略.采用端到端的场景通常具有“外部逻辑清晰,内部结构复杂”的特点:(a)系统的输入/输出的关联明晰,且有足够的数据或者训练环境以便算法学习从环境数据到策略的直接映射;(b)系统内部结构相互关联,非常复杂,某些功能部件或者过程甚至无法有效观测.端到端的方法可以进一步细分为预测和控制,预测指输入是环境/系统运行数据,输出是环境/系统未来状态;控制指输入是环境/系统运行数据,输出是当前时刻应采取的自适应操作.

强化学习、神经网络/深度学习是常用的端到端学习算法.强化学习中的智能体、策略、状态、动作、奖励、环境可以和自适应软件系统、自适应逻辑、环境或系统状态、自适应操作、效用、上下文(context)对应,在构建自适应逻辑时,只需考虑系统输入的状态、效用,输出的自适应操作,而不必关心系统内部对状态、效用、自适应操作的分析[23,34,77].但由于自适应系统的复杂性,这种单纯端到端的做法只是最理想的情况,针对不同被控系统,不同环境假设,在学习机制之外,可能需要添加一些额外的过程以保证能够获得有效的自适应策略.

Table 1 Learning enabled software self-adaptation goal decomposition

表 1 机器学习赋能的自适应目标分解

一级子目标	二级子目标	任务功能说明
G1 应对场景不确定性	G1.1 应对需求缺失	需求获取和分析阶段未成功提取出某些需求,但这些需求同软件系统行为密切相关
	G1.2 应对需求二义性	某些需求表达不清晰,存在不同的解释方式
	G1.3 应对需求改变	一些需求没有反应出用户对系统真实需要和对需求约束
	G1.4 应对假设不准确或错误	对无法提前获取的环境或系统特性的不准确或错误的主观估计,不准确是指通过分析方法得到的估计值和真实值有差异,错误是指分析估计的方法不正确
	G1.5 应对感知误差或错误	行时监测器受限于其精度或性能,导致测量值和真实值存在误差,或者监测器损坏、真实值超出测量范围或被攻击导致测量结果错误
G2 应对系统和环境的动态变化性	G2.1 应对环境状态变化	环境特性发生改变,系统可以观察到这些变化,例如,智能家居中,白天、黄昏、夜晚等,自动驾驶中,晴天、雾天、雨天、雪天等
	G2.2 应对环境行为变化	环境行为是指环境对系统的主动交互,例如,Web 服务器系统中,网站访问量、访问者浏览网页的偏好等,机器人系统中,人对机器人进行路径或者动作的干扰、允许机器人进入的区域改变等
	G2.3 应对系统结构变化	系统构件或构件间的联系发生改变,例如云计算系统中,底层网络架构或信息存储架构发生改变等
	G2.4 应对系统行为变化	运行过程中,在内外因素的作用下,系统的行为没有按照事前的规划执行,包括了系统主动调整的行为变化、由于失效产生的错误行为等,例如物联网系统中,监控的时间周期改变、网络传输优先级改变等
	G2.5 应对系统属性变化	系统属性指设计时对系统功能性或非功能性的需求和支撑系统的关键指标,例如,Web 服务器中,通信网络的带宽、服务质量(QoS)、电能消耗等,机器人系统中,CPU 计算性能、传感器精度、内部系统总线的速度等
G3 应对操作效果的不确定性	G3.1 应对系统外部特性的不确定性	指系统的行为操作,其实际的输入/输出和规划的输入/输出发生偏差,例如,机器人系统中,控制器向动力系统发出动作指令并期望得到某个效果,但其物理负载、机械误差、外部环境使动力系统执行的动作指令的效果发生偏差
	G3.2 应对内部影响的不确定性	系统行为导致系统自身产生了未经验证的改变,这些系统和系统变化间的相互作用,难以建立精确的联系,例如,物联网系统中,降低 CPU 频率以降低系统功耗,引发整个系统监控周期、通信延迟、响应时间等性质的改变
	G3.3 应对延迟效应的不确定性	系统行为不能直接作用到系统或外部环境中,会有一段延迟,在行为到产生效果这段过程中,系统处于不稳定的状态,例如,Web 服务器,从开启备用服务器到备用服务器成功运行服务有延迟,且延迟时间取决于系统当前负载、网络带宽、延迟期间内外部服务请求数量、开启备用服务器的顺序等因素攸关
	G3.4 应对需求指标的不确定性	对于系统行为,很多情况下不能直接知道行为会给系统的需求指标(例如 QoS,安全性要求等)带来什么样的影响,例如,Web 服务器,调整服务响应的优先级顺序后,不能直接获得 QoS 指标

(2)阶段/过程的方法.根据 3.1 节的系统结构层和功能部件层,自适应软件系统由多个功能部件组成,其自适应过程分为多个阶段.阶段/过程方法就是指解决其中某个阶段上的分类、回归或决策任务.已有工作展示,机器

学习可以用在不同的阶段,承担不同的角色,大致分五种:策略生成、策略演化、参数估计、效果预测和环境预测.策略生成指在离线数据上进行训练得到自适应策略[35,42].策略演化指在系统运行时收集数据,随着系统运行时的动态变化可以不断调整自适应策略[35,78].参数估计指对自适应的不确定性进行建模,得到可以量化的概率指标[61,79].效果预测指根据自适应操作和需求目标的实际数据,建立二者的联系,实现根据自适应操作预测出需求目标的指标,需求目标可以是系统延迟、吞吐量、响应时间、目标满意度、物理风险等[35,19,48].环境预测指根据系统的历史数据和当前数据,建立对环境的抽象模型,并对环境的动态变化做出预测[52,80].对应到自适应系统功能部件上,策略生成/演化、参数估计、效果预测和环境预测,可看作为机器学习分别应用于自适应系统的规划过程、分析过程、执行过程和监控过程.

Table 2 Self-adaptation task and solution

表 2 自适应任务及解决方法

编号	任务描述	解决方法	解决方法分类	采用的机器学习算法
T1	需求反射机制	ACon[80],需求反馈框架[39],SACRE[118]	策略演化	深度学习,马尔可夫决策过程
T2	需求估计	需求模糊逻辑[56]	参数估计	神经网络
T3	假设参数化	模型修正[49],模型迁移[43]	参数估计	马尔克夫决策过程,迁移学习,
T4	加工生数据	ACon [80], 自适应大数据分析[50]	效果预测	深度学习,聚类算法
T5	推测隐藏信息	RE-STORM[61],动态决策[86]	参数估计	部分可观测马尔可夫过程,动态决策网络
T6	环境状态建模	环境状态自动机[34],环境模型自适应[98],	环境预测	强化学习
T7	环境行为预测	运行时环境模型[52],环境模型自适应[98]	环境预测	深度学习
T8	系统状态建模	系统服务组合规划[64]	效果预测	高斯过程,强化学习
T9	触发事件分析	条件触发机制[23],运行时检测[20]	策略演化,环境预测	强化学习,马尔可夫过程,BAAM-WELCH 算法
T10	操作延迟分析	时间敏感资源调度[53],服务排序[36]	环境预测,效果预测	强化学习
T11	操作到系统变化映射	FUSION[19], 系统模型迁移[60],系统行为模型[100]	效果预测,策略生成	深度学习,迁移学习,马尔克夫过程,
T12	环境历史数据分析	ACon[80],自适应大数据分析[50]	环境预测	深度学习,数据挖掘
T13	其他系统行为预测	面向服务的多智能体体系结构[65]	环境预测	多智能体强化学习
T14	外部特性分析	自适应策略生成演化[78]	策略演化	强化学习
T15	目标满意度分析	FUSION[19],IDES[44],系统动态配置[95]	效果预测	深度学习
T16	操作到系统性能映射	FUSION[19],动态服务配置[77]	策略生成	深度学习,强化学习
T17	系统历史数据分析	自适应大数据分析[50]	效果预测	深度学习,数据挖掘

4 机器学习与系统自适应性

本节针对研究问题 RQ2,以软件系统自适应性为目标,分析其与机器学习方法之间的关联,讨论为实现系统自适应性而采用的各类机器学习赋能技术.

4.1 自适应系统中的机器学习

如图 7 所示,现有工作中常用包括强化学习、贝叶斯决策理论和概率图模型(简称贝叶斯方法)、神经网络/深度学习、规则学习、迁移学习等方法. 比如,最常用的是强化学习(33 篇),其次是神经网络/深度学习(24 篇),有两篇文献采用了迁移学习.有的工作还用到包括 EM 算法[19]、聚类算法[20]、L*算法[72]、随机森林[73]、梯度增强回归模型[73]、极端提升树算法[73]、决策树[74]、线性回归模型[75]、卡尔曼滤波器估计[62]、多智能体协作学习算法[40]等的其他方法,但这些方法在本文的调研范围中仅被 1 篇文献所采用.

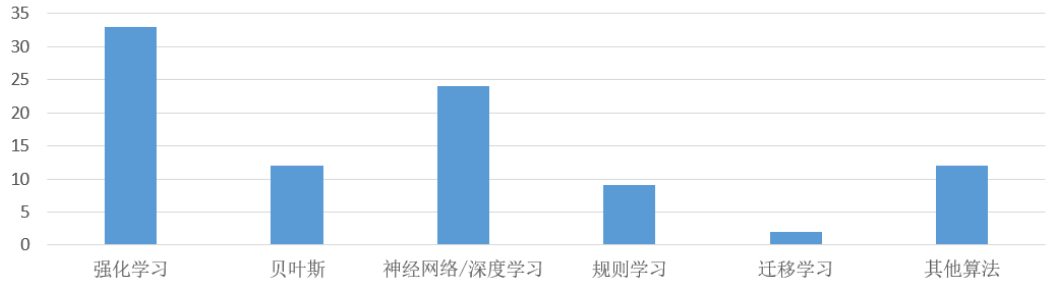


Fig.7 Different machine learning algorithms in literature

图 7 不同机器学习算法的文献数量对比

图 8 统计了从 2004 年到 2018 年,在软件自适应性中采用机器学习算法的论文数量随年份的变化,论文发表数量整体呈上升趋势.下面分析这些常用的机器学习算法.

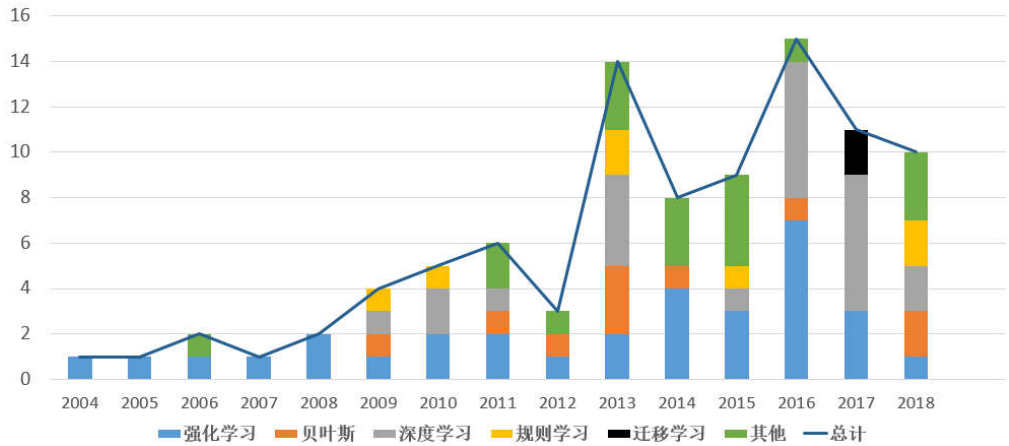


Fig.8 Publications in each year

图 8 每年论文发表数量

- 采用强化学习的工作分为基于模型和免模型两类.基于模型的方法用马尔可夫过程来建模实际环境,比如,传统的马尔可夫模型(MDP)[34,85]、结合高斯过程的马尔可夫模型(Integrating-Gaussian-Process MDP)[64,65]、部分可观测的马尔可夫决策过程(POMDP)[61]等.当模型存在不确定性时,则利用贝叶斯方法进行处理,引入隐马尔可夫链、条件随机场等对环境建模.在环境模型的基础上,用动态规划、蒙特卡洛等进行最优决策规划.免模型的算法不构造环境模型,而是直接生成策略[41,96,110,],基础的算法有 Q-learning[26]、SARSA[26]、Actor-critic[33]等,以及结合了深度学习的 DQN[28].
- 贝叶斯决策理论和概率图模型除了可以为自适应决策提供建模支持外[19,66,86],更重要的是可以描述自适应软件设计、开发的整个过程中的不确定性,包括估计运行时系统中与软件自适应相关的参数[58,79],检查在存在不确定性的情况下,模型是否满足属性[93],对非功能特性进行持续验证[61],以及在需求发生改变时对系统进行调整[118]等.
- 神经网络/深度学习通常被应用于采用端到端方法进行效果建模、环境预测、效果预测等任务,采用黑盒的方法,只考虑所关注部分的输入/输出的外部特性,不考虑该部分的内部结构或逻辑.适用的场景有两个特点: (1)系统或环境内部结构复杂或难以观测; (2)系统或环境的输入/输出的数据非常明

确,且容易获取.也有文献采用部分可观测马尔可夫决策过程(POMDP)[61]、去中心化的马尔可夫决策(DECMDP)[84,111]、多智能体模型[65]等方法来描述复杂系统内部结构的变化规律.神经网络/深度学习的优势是充分发挥其对高维复杂数据的处理能力,在不分析系统或环境结构和逻辑的情况下,建立输入数据和输出数据的关联.如果系统或环境的真实数据难以获取,也可以采用模拟的方式生成数据,并将在模拟数据上训练出的深度学习模型迁移到真实场景中[43,60].

- 规则学习以一阶逻辑或命题逻辑的形式表达自适应策略,这样构造的系统具有较高的可验证性,可靠性,可理解性.MORPH[72]提供了自适应软件分层体系结构,将规则学习纳入到策略颁布层(strategy enactment),用以生成系统配置的调整策略.Sykes D 等人给出了在机器人行为模型发生改变时,用一阶逻辑构造可靠的自适应策略的方法[100].ACon[80]框架给出了运行时处理需求阶段带来的不确定性的方法,用规则学习来构造系统框架的调整策略.应用规则学习的场景通常是机器人系统[72,100,106]、实时系统[21]等,这些场景对系统的可靠性有很高的要求.

机器学习算法是和任务相关的,不同的任务可以采用不同的方法,例如文献[64]同时应用了强化学习和贝叶斯方法,文献[19]同时应用了贝叶斯方法和 EM 算法,文献[36,51,96]同时应用了强化学习和神经网络/深度学习,文献[97]同时应用了规则学习和决策树、以及频繁项挖掘.

4.2 机器学习赋能的交互

从系统外部特性视角出发,自适应涉及系统的输入/输出.其中,输入包含外部环境数据、系统自身数据和应用需求,这些输入由系统的监测过程来收集.监测过程收集的这些未经处理的原始数据(即“生数据”),可能是不准确的、模糊的、有噪音的、甚至是错误的,系统需要对“生数据”作预处理,以解决可能出现的需求缺失、需求二义、需求改变、传感器误差或错误等问题,这里涉及 4 类任务:T1 需求反射机制、T2 需求估计、T3 假设参数化、和 T4“生数据”再加工.文献[86]针对 T1、T2 任务,采用贝叶斯决策理论,选择最优需求设置,采用贝叶斯网络(BNs)、动态决策网络(DDNs)[86]等算法.文献[80]针对 T1、T2 任务,持续监控需求,得到需求数据,采用支持向量机(SVM)对需求不确定性进行分类.针对 T3 任务,为应对假设和真实场景不符的情况,文献[43,60]采用迁移学习,将在假设条件下的自适应策略迁移到真实场景中.针对 T4 任务,一些工作利用轻量级的算法,对“生数据”进行简单预处理再用于后续学习算法的训练[61,62].同时,还有工作根据采集到的当前数据与历史数据,建立数据间的对应关系,支持自适应系统对环境进行预测,以便自适应逻辑能在预测数据上进行自适应决策[66].这称作“主动延迟感应自适应”(proactive latency-aware adaptation),自适应逻辑同时考虑当前情况(包括当前的系统状态、环境状态、效用等)和预期情况(包括未来的系统状态、环境状态、效用等),以及操作的延迟性.马尔可夫决策过程是一种比较常用的主动延迟感应自适应的方法[64,65,37].如果环境模型已建立,则直接根据模型进行预测;如果没有模型,则根据历史数据进行预测.

自适应系统的输出是自适应操作,即对被控系统(自适应逻辑和应用逻辑完全分离时)或者环境(自适应逻辑和应用逻辑没有完全分离)的行为动作,由执行过程实施.自适应操作按其粒度从细到粗可以分为: (1)参数调整,改变应用逻辑中的参数设置;(2)行为调整,改变应用逻辑中的方法、函数调用;(3)结构调整,改变应用逻辑中的构件(component)或构件之间的关系.

自适应系统外部特性视角和端到端的机器学习方法联系紧密,端到端的方法只考虑系统外部特性,主要有强化学习和深度学习等方法,自动建立从输入端的监测内容到输出端的自适应操作的映射.端到端的方法将在 4.5 节详细分析.图 9、图 10 统计了现有工作,可以看出,就监测内容而言,外部环境数据的有 61 篇,系统自身数据的有 60 篇,应用需求的有 17 篇.大多数工作(35 篇)的监测内容既有外部环境数据,又有系统自身数据.目前还没有只监测应用需求的工作.在自适应操作方面,只进行参数调整的最多,有 25 篇,其次是只进行行为调整,有 21 篇,目前没有发现同时进行参数调整、行为调整和结构调整的工作.

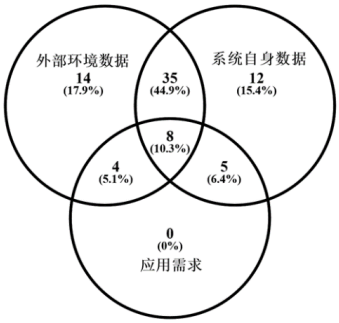


Fig.9 Different monitoring data in literature
图 9 不同监测内容的文献数量

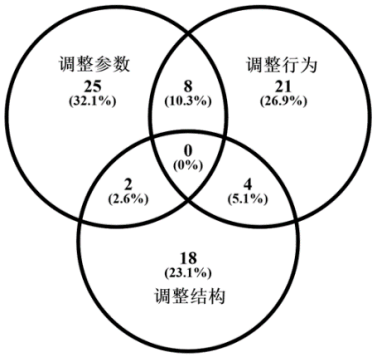


Fig.10 Different self-adaptive execution in literature
图 10 不同自适应操作的文献数量

4.3 机器学习赋能的控制

自适应系统结构视角关注控制器和被控系统.其中在控制器部分主要关注自适应目标、自适应原因和自适应决策方式.自适应目标有两种,即保持系统稳态,和最优化需求目标.

保持系统稳态的问题可描述为:对任意$\delta \in \Delta, s \in S$, 当$\delta\big(f(s,t_0)\big) \notin A(s)$,且 $\delta\big(f(s,t_0)\big) \notin F(s)$时, $\exists \Delta t, \forall t > t_0 + \Delta t$, 都有 $\delta\big(f(s,t)\big) \in A(s)$.其中 δ 是扰动, s 是状态, f 是自适应逻辑, A(s)是在 s 状态下的可接受域,F(s)是在 s 状态下的不可接受域,Δt 是系统恢复时间,上述过程用自然语言可以描述为,系统 f 在受到 δ 的干扰后, 系统偏离可接受的外部特性(不在 A 内), 且没有崩溃(没有进入 F), 系统 f 可以在 Δt 时间内恢复到可接受的外部特性的状态[46].

最优化需求目标是一个优化问题,可以描述为:

$$\min_{\theta} F(\theta) = [f_1(\theta), f_2(\theta), \dots f_n(\theta)]^T, \text{s.t.}\, c_1, \dots, c_m$$

其中 f 为目标函数,n>1 时,是多目标优化,c_i 是限制条件,通常是不等式形式.最优化需求目标可分为可变目标和不变目标两类.前者处理需求获取阶段无法准确表达需求目标和目标间关系的情况;后者处理需求获取阶段可准确表示系统目标,且运行阶段需求目标也不会发生变化的情况.从统计结果可以看出(如图 11),目前大部分工作都研究最优化不可变需求目标(45 篇)问题,以保持系统稳态为目标的工作较少(2 篇).其原因可能是机器学习在处理保持系统稳态的能力上,不如控制论方法直接.

触发系统自适应能力的原因是动态变化和不确定.能够应对动态变化和不确定,且保持系统鲁棒性是自适应系统有别于其它软件系统的重要特征[2].动态变化可分为外部环境变化、系统自身变化和其他系统变化.从机器学习的角度,单系统和环境的交互可视为马尔可夫决策过程,该过程中的状态跳转仅和当前状态、采取的动作有关系;而多系统和环境交互无法再用传统的马尔可夫决策过程准确描述,这类交互一方面可以看作博弈过程,另一方面也保留了马尔可夫决策过程的特性[83,84].这类问题在多智能体领域的研究很多.但多智能体系统、博弈论等领域的成熟技术,在软件自适应领域中的使用还较少.不确定性根据产生的时间,可分为需求不确定性、设计时不确定性、运行时不确定性.模糊控制、贝叶斯决策理论和概率图模型是处理不确定性的常用方法,模糊控制主要侧重在需求获取阶段和自适应规则设计阶段,处理二值逻辑难以描述清楚的情况[55,59,85],贝叶斯决策理论和概率图模型主要侧重于系统和环境的不同随机变量的因果关系、关联关系[61,64,86].

统计现有工作,如图 12,有 76 篇论文关注动态变化,单独考虑外部环境变化的最多(32 篇),其次是考虑外部环境变化和系统自身变化(20 篇).没有同时考虑外部环境变化和其他系统变化的工作.有 54 篇关注不确定(如图 14),单独考虑运行时不确定的最多(22 篇),其次是考虑运行时不确定和设计时不确定的工作(13 篇).尚没有同时考虑需求时不确定和设计时不确定的工作.

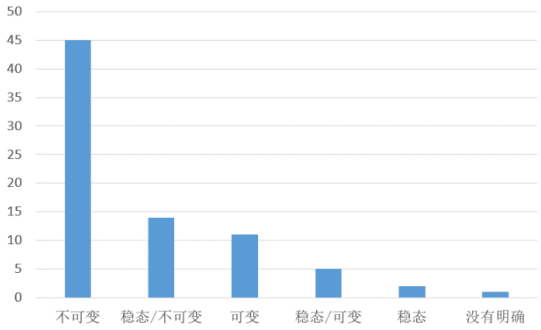


Fig.11 Different self-adaptive goals in literature
图 11 不同自适应目标的文献数量对比

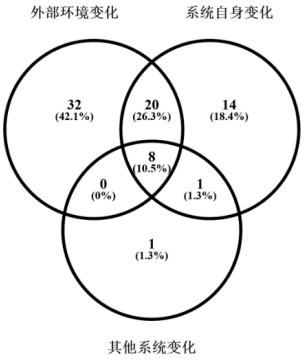


Fig.12 Different dynamics in literature
图 12 不同动态变化的文献数量对比

自适应逻辑的实现又称决策方式,分为反应/前摄式(reactive/proactive)和连续控制式.反应式是在自适应条件发生之后被实施的,前摄式是在自适应条件发生之前根据模型进行预测而实施的.连续控制式不涉及显式的自适应触发条件,而是在整个控制过程中持续地进行调控.反应式和前摄式的概念分别源于控制论中的反馈控制和前馈控制.前馈控制系统是基于补偿原理,根据系统外部环境的扰动量进行工作;反馈控制系统是根据被控量和给定值之间的偏差进行工作[46].当系统存在延迟敏感(即,从自适应操作到产生效果存在时间延迟,且该延迟会影响到对自适应操作好坏的评价)时,需要采用前摄式来提前调整,前摄式的关键问题有前摄模型的构造和对自适应攸关的数据的预测.

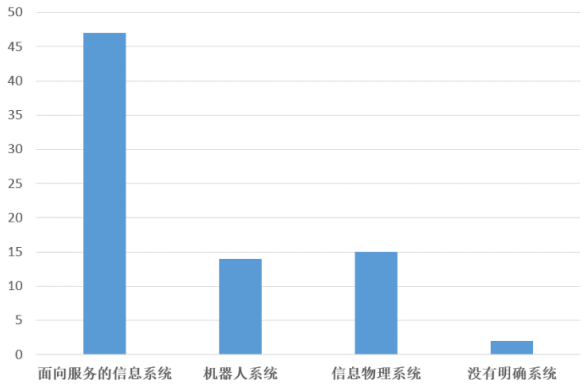


Fig.13 Different controlled systems in literature
图 13 不同被控系统的文献数量对比

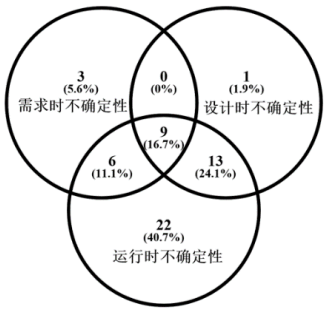


Fig.14 Different uncertainties in literature
图 14 不同不确定性的文献数量对比

反应/前摄式在编码阶段通常采用 ECA 范式[45,47],连续控制式在编码阶段通常采用控制函数(例如 PID、模糊控制、神经网络控制)来实现[46,53,59].反应/前摄式的自适应控制还可以分为静态和动态两种.静态反应/前摄式在运行过程中,按照预先设计的策略/规则做出自适应调整;动态反应/前摄式在运行过程中,一方面按照预先设计的策略/规则做出自适应调整,另一方面会监控调整的结果,对目标满意度进行评估,当预先设计的策略/规则无法满足目标时,则会在线学习新的策略/规则.从学习的角度来看,静态策略/规则设计可以看作“策略生成”,动态策略/规则设计可以看作“策略演化”[35]. 图 15 统计了目前的工作,反应/前摄式的决策的工作有 54 篇,其中静态 32 篇,动态 22 篇,连续控制式 23 篇,1 篇没有明确指出其决策方式.

被控对象可分为面向服务的信息系统(service oriented information system)、机器人系统(robotic system)和信息物理系统(cyber physical system).不同的被控系统对自适应逻辑有不同的需求,例如,机器人系统一般对自适应决策的实时性有较高的要求,机器人系统和信息物理系统一般对自适应决策的可靠性和风险控制能力有

较高的要求.这些非功能性需求需要用不同的机器学习赋能技术来实现.可以利用目标模型(goal model),分析出目标间的关联性,从而探索什么样的学习机制适宜于什么样的系统[39,81,82].从现有工作来看,如图 13 所示,目前大多数工作的被控系统都是面向服务的信息系统.

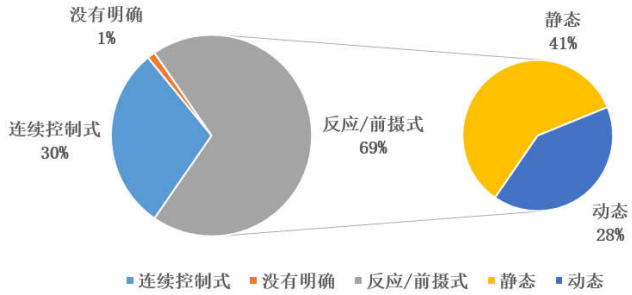


Fig.15 Different decision-making methods in literature

图 15 不同决策方式的文献数量对比

不同被控对象背后所涉及的是自适应软件系统的应用场景.这些场景既有共通之处,又各有其侧重.下面从自主性、安全性、实时性、动态变化、不确定性等方面对三类被控系统进行考察.

面向服务的信息系统的自主性体现在应用可能发生变化,用户数量不断增加且其行为模式难以建模,以及大量并发访问等,需要系具有资源调整、服务优化等的自我管理能力[115,143,185].安全性、实时性方面通常在 QoS 中分析.在动态性、不确定性方面,主要考虑运行时并发访问、资源消耗的情况.利用机器学习算法构造自适应策略,需要处理在资源调度过程中的组合优化问题[119,143,179].当系统运行能确定地建模时,也可以采用一些传统的非学习的分析方法(例如排队模型、任务调度算法)[36,97],但在缺乏先验知识时,这类方法往往无法得到正确的结果,只能借助机器学习从系统运行的历史数据中自动地提取出资源调整的最优策略.

机器人系统对自主性的要求较高,许多机器人系统都以无人自主系统的形式出现,同时机器人时刻与环境进行交互,在实时性方面也有较高要求,需要机器人对环境变化能迅速作出调整[101].同面向服务的信息系统不同,机器人系统所要控制的变量往往都是连续变量,例如速度、高度、角度等.主流的方法是基于控制论的方法[4].现代控制论通常使用状态空间方程对系统进行描述[46],马尔可夫决策模型正是现代控制理论所倚重的一种数学工具,而马尔可夫决策模型又是强化学习的基石.因此现代控制以及智能控制的方法自然延伸到了强化学习的方法.这是控制论和强化学习的内在关联,因而强化学习在自适应机器人系统中得到了广泛的应用.

信息物理系统在自主性、实时性方面的需求同前两种被控系统类似,但由于其系统内部结构复杂,对外交互复杂,兼具信息空间交互、物理空间交互;系统故障容易对外界产生严重的影响,使得它在安全性方面较前两种有着更高的要求.也对机器学习算法在收敛性、稳定性、准确率方面有更高的要求.在信息物理系统中,学习功能通常由系统所控制的智能体承担,可以是机器人[123],智能传感器[89],智能网格等[70]. 和强化学习中的反馈回路类似,智能体在环境中接收刺激-采取行动,同时,智能体所观测的数据可能涉及到众多非线性函数、隐变量,因此常常采用强化学习、深度学习、贝叶斯方法.文献[57]综述了包括信息物理系统在内的多智能体自适应系统中,使用机器学习技术的相关工作,可供进一步参考.

4.4 机器学习赋能的自适应过程

从功能部件视角进行分析,目前的工作可以分为分离式架构和混合式架构.分离式架构指,作为控制者的自适应逻辑独立于作为被控对象的应用逻辑的框架,有 Rainbow[8]、MRAS[9]、IDES[44]、MORPH[72]等;混合式架构指,自适应逻辑和应用逻辑交织在一起,并没有显示分离出来的框架[18].

分离式框架和混合式框架都将自适应过程分为四个阶段:监测、分析、规划和执行.其中监测过程和执行过程在 4.2 节已有详述.分析过程是自适应系统的核心,涉及到了五个子任务,包括: T5 推测系统或环境的隐藏

信息;T6 建立环境状态变化模型;T7 预测环境行为模式;T8 建立系统状态变化模型;T13 预测其他自适应系统行为变化等.现有工作中,实现 T5,采用贝叶斯方法进行参数估计和环境预测,包括隐马尔科夫模型[82,86]、EM 算法[19]等.实现 T6 和 T8,用自动机对环境状态建模,再利用马尔可夫决策过程进行求解[66].预测任务 T7 和 T13 通常采用深度学习和博弈论,深度学习根据历史信息找到数据变化规律,从而预测未来发展趋势[51,52],博弈论是经济学方法,用于预测理性人在市场中的决策结果,在自适应系统中,用于预测多智能体场景下的系统运行结果[65,111].统计现有工作,机器学习可以应用于功能部件层的多个过程,结果如图 16 所示,应用于规划过程的最多(53 篇).

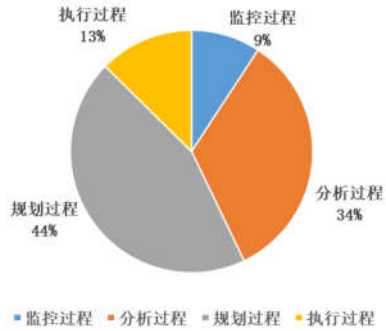


Fig.16 Different processes using machine learning
图 16 应用机器学习的自适应过程文献数量对比

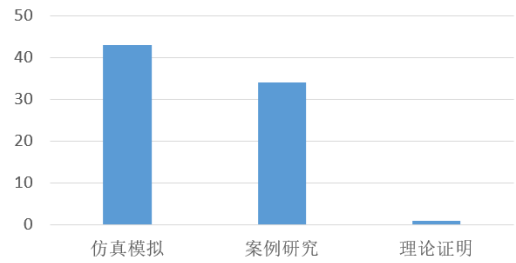


Fig.17 Different Verification & Validation
图 17 不同验证评价的文献数量对比

除了这三个视角上的工作,验证评价,即“Verification & Validation”过程,也是一个重要问题.主要手段有理论证明、案例研究(case study)、模拟仿真(simulation).目前理论分析的工作很少,大部分采用后两种方法.案例研究指搭建特定场景的实验系统,并进行实验评估.模拟仿真则借助于仿真工具对自适应算法进行评估.统计现有工作,结果如图 17 所示,采用仿真模拟有 43 篇,采用案例研究有 34 篇,采用理论证明的有 1 篇.在某种程度上说,对机器学习赋能的软件自适应,很少在理论上研究其有效性,而更多是用仿真的方法来验证机器学习赋能的自适应系统的功能和特性是否满足设计要求.对不同的被控系统,有不同的验证方式,面向服务的信息系统通常采用仿真模拟的方法进行验证,RUBiS(http://rubis.ow2.org/)是一种 Web 服务领域的功能测试平台[87],常被用于测试网站系统访问延迟、系统吞吐量、处理器开销、内存占用等指标,可模拟用户访问的动态变化[66,73,88].大型复杂应用场景下的机器人系统,常用仿真模拟,小型简单应用场景下的机器人系统,常用案例研究.机器人仿真模拟平台有:Robocode (http://robocode.alphaworks.ibm.com/home/home.html)、RoboCup (https://rescuesim.robocup.org/)[90]、ROS(Robot Operating System, http://www.ros.org/)[91]等.信息物理系统的验证常用案例研究,例如构造智能家居系统[92]、医疗传感器系统[93]、小型智能城市物联网系统[94]等,模拟仿真平台有 OpenDaVINCI[21].

4.5 自适应任务及机器学习赋能

综上所述,软件系统自适应性主要有如下三个关注点:

- **场景不确定**.这个不确定性贯穿于需求时、设计时和运行时,任务 T1、T2、T3、T4、T5、T12 分别实现对不确定场景的感知、分析和处理.概率模型、模糊逻辑则将不确定性进行定量化.贝叶斯决策理论为分析、处理多个互相关联的随机变量提供支持.估计环境或系统中变量的任务,可以使用期望或贝叶斯概率.稍微复杂一点的情况,变量间存在某种关联的可以使用贝叶斯网络.当系统或环境存在隐变量时,可以采用 EM 算法进行估计.当需要考虑系统或环境的动态性时,可以用带概率状态跳转的马尔克夫决策过程处理.如果在动态变化过程中,真实状态无法被测量,可以用隐马尔科夫模型来处理,隐马尔科夫模型能够有效分析观测量和实际状态在动态变化时的规律.如果变量分布存在某些先验知识,可以用高斯过程、马尔可夫过程等随机过程来假设变量的分布.在真实环境中,监控的“生数

据”不能直接反应出系统或环境的特点,可以采用一些聚类、拟合、分类算法对数据作预处理.总体来看,应对场景不确定时,以贝叶斯决策理论和概率图模型为主,其他手段为辅.

- **环境动态变化**.任务 T6、T7、T8、T9、T11 和 T13 直接应对环境动态变化,支持系统随环境变化而变化,任务 T12、T14 和 T10 支持动态变化目标的实现.环境和系统的动态变化相互关联,环境动态变化是内生的、决定性的,但系统动态变化可以反作用于环境,影响环境变化.环境的动态变化,可用环境建模的方法来刻画,并用贝叶斯决策理论和概率图模型来分析,但当环境的状态跳转特征不明显时,只能从数据上分析环境的规律,这时可用深度学习,从环境的历史数据中发掘规律,以预测环境的变化.对于系统的动态变化,与下面三个任务相关: (1)决定自适应操作在何时何地如何执行;(2)掌握系统状态变化规律;(3)确定自适应操作和系统结构、行为、属性的关系.当缺乏系统外部特性的先验知识,却能给出系统和环境的精确模型时,可采用强化学习的方法解决任务(1),根据系统和环境的交互经验,学习到自适应策略,并可以随着环境的变化,继续演化自适应策略.如果有充足的数据,但缺乏对系统和环境的精确建模,则采用深度学习,把系统作为黑盒,进行端到端控制,只考虑系统的输入/输出,从运行数据中得到自适应逻辑.任务(2)可以通过分析环境的状态变化来实现,建立隐马尔可夫模型、部分可观测马尔可夫模型等.对任务(3),深度学习给出了端到端预测的方法,模型的输入是自适应操作,输出是要考虑的系统指标,需要大量学习样本的支撑,这些学习样本大多数来自于 off-line 阶段,由于 off-line 数据有可能和 on-line 数据有差别,因此又引入迁移学习,将 off-line 数据上学习的模型迁移到 on-line 阶段.总体来看,应对动态变化,以强化学习、深度学习为主,辅之以其他手段.
- **效果不确定**.理想情况是,所有自适应操作都能达到预期的效果.但实际效果常常不确定并可能和预期效果不一致,这也是系统需要进行动态调整的原因.任务 T10、T14、T15 和 T16 就是为了应对预期效果存在不确定的情况.端到端预测的方法采用深度学习建立自适应操作到环境变化的关系,同样也涉及到 off-line 和 on-line 数据差异、历史数据处理的问题.在前摄式控制中需要处理操作延迟效应,通常在马尔可夫决策过程中引入奖励折扣,来模拟自适应操作效果的时间延迟.总体来看,应对效果不确定时,以深度学习为主.

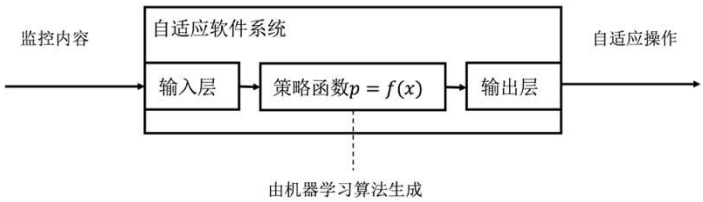


Fig.18 End-to-end learning method for self-adaptive system

图 18 自适应软件系统端到端的学习方法

针对上述三个关注点,现有机器学习赋能技术分为两类:端到端的方法和阶段/过程的方法.端到端方法(如图 18)将系统中所关心的部件看成黑盒,只考虑其输入/输出的外部特性.输入为监测内容(环境/系统运行数据),输出为自适应操作.该问题的数学模型为:监测数据为$\vec{X} = [X_0, \dots, X_i, \dots, X_n]^T$, X_i的值域为$\mathrm{Dom}(X_i)$, 输入数据空间为$\vec{X} = \mathrm{Dom}(X_0) \times \dots \times \mathrm{Dom}(X_n)$, 系统输出为 P,值域为Dom(P).理想的系统的外部特性为$\left(\vec{X}, P\right)$,学习算法构造出的控制函数为$\hat{F}\left(\vec{X}\right) = \mathrm{P}$,优化目标为使控制函数接近理想系统的外部特性,即最小化$loss = \sum_{\vec{X} \in X} \left(\hat{F}\left(\vec{X}\right) - P\right)^2$.当系统内部结构复杂,但输入/输出却非常明确,且真实的输入/输出数据也容易获得

时,通常采用端到端的方法.

端到端方法中采用的算法为神经网络/深度学习,强化学习等.其中,用强化学习来实现端到端的自适应逻辑设计比较直接,通常是将强化学习中的智能体、环境、动作、状态、奖励同自适应中的自适应软件、环境、自适应操作、环境状态、效用一一对应起来,可以用图 19 来表示这种对应关系[34,64,105].

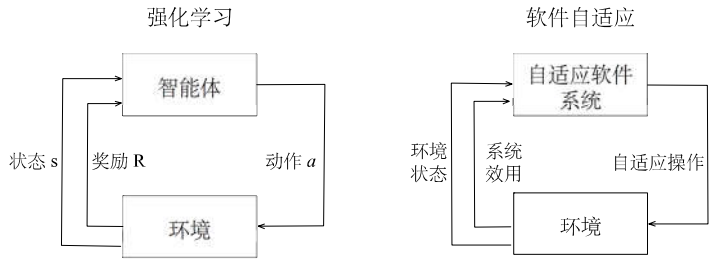


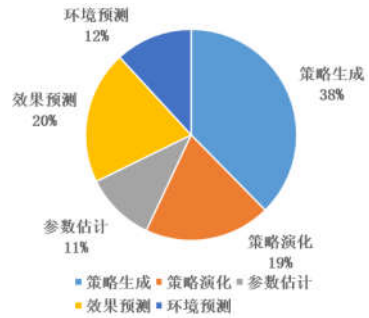
Fig.19 The agent-environment interaction in reinforcement learning and the system-environment interaction in self-adptive software system

图 19　强化学习的智能体-环境交互与自适应软件系统的系统-环境交互

阶段/过程的方法是将机器学习算法应用于软件自适应的不同过程(监控、分析、规划、执行),在不同的阶段设计.按照任务的不同,可以分为策略生成、策略演化、参数估计、效果预测、环境预测.

- 策略生成是静态的,根据离线准备的数据得到运行时策略,通常要求系统在设计时已经很清楚运行时的系统、环境的状态信息[34,77,98].
- 策略演化是在运行时随着系统或环境的变化而发生的策略改变,即机器学习中的“在线学习”(on-line learning),为了提高系统初始化性能,一般不直接让系统在没有任何先验知识的情况下从头学起,通常会配合离线的策略生成或者利用领域知识得到初始自适应策略[21,44,53],然后通过策略演化获得更好的策略.
- 参数估计主要解决场景的不确定性,对系统、环境的不确定用概率或者模糊逻辑的方法进行参数化,从而得到一个可以评估的指标[53,61,99].
- 效果预测建立自适应操作到系统或环境的实际反应的映射.在系统组件的选择[95],和系统的特征选择[19]过程中常常需要进行效果预测,例如一个安全验证模块的开启或关闭,可能会引起系统资源消耗、安全性、响应时间等多方面的影响,需要用效果预测对该模块对系统的作用进行评估.
- 环境预测分为两种,一种是直接处理原始监测内容的“生数据”,根据历史数据和当前数据,预测将来的数据[62];另一种是在环境模型或系统行为模型上,预测环境的变化或系统的变化[100].环境预测常在一些具有延迟性的自适应操作上使用,用于预测系统或环境的发展,做出前摄式的自适应规划.

这五种方法可以在一个系统中同时部署.如图 20 所示,在现有的工作中采用端到端的方法有 17 篇,阶段/过程方法的有 61 篇.其中阶段/过程方法一篇中可能同时应用多种具体的方法,图 21 统计的是所有方法累计被使用的次数,最多的为策略生成(35 篇),其次是效果预测(19 篇),最少的是参数估计(10 篇).



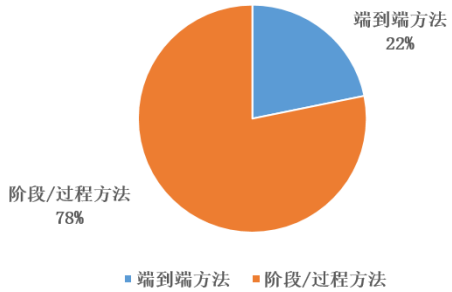


Fig.20 Different methods using learning algorithms

图 20 不同学习算法的应用方式的文献数量对比

Fig.21 Different phase/process methods

图 21 不同阶段/过程方法的文献数量对比

图 22 展示了自适应关注点和机器学习方法的对应关系,图中连线表示两者关联,线条的形式表示文献数量的多少,例如动态变化和策略演化相连,表示关注动态变化并用策略演化的方法来解决;参数估计和贝叶斯方法相连,表示采用参数估计的方法并用贝叶斯方法进行学习.

从统计结果可以发现从自适应需求的三个子目标到六种自适应任务,再到机器学习算法有其天然的对应关系.结合 3.2 节的分类,本文就软件自适应和机器学习赋能技术之间的关系做进一步的分析.就系统自适应性而言,如果设计者对系统的知识是完备(complete)的,那么可以通过排队模型、时间自动机模型、马尔可夫决策模型进行建模分析,给出有效的自适应策略.若在完备的模型上求解自适应策略的计算复杂性高,则会考虑一些启发式算法[4,36,97].

但是,如果设计者对于系统的知识是不完备的(不完备可能来自于很多方面,设计者缺乏专家知识,系统难以精确建模,很多关键变量无法获得等),则需要引入学习机制.一般而言,引入机器学习技术,可以实现在不完备知识的情况下,通过学习机制来赋予软件系统自适应能力,即让系统通过对运行数据的自动分析,提炼出自适应策略.从不同自适应需求目标的特性来看,不同需求目标所对应的问题域不尽相同,在不同的问题域上,采用的机器学习技术也不同.通常,场景不确定性蕴含的主要是存在隐变量的问题,对于隐变量,或是利用 EM 算法,从系统的运行数据中学习得到.或是利用贝叶斯方法,给系统提供一种在含有不确定性的情况下的决策机制[19,61,66].环境动态变化蕴含的主要是策略和真实情况的匹配问题,当现有策略和真实情况不匹配时,需要系统对策略进行调整,利用强化学习,从系统和环境的交互中学习到策略与环境的最佳匹配.此外为了探测这种不匹配,也会涉及到学习技术设计,需要一种拟合从环境的直接观测数据到需求目标的满意度的关系,可采用深度学习等方法[23,34,113].操作效果不确定所蕴含的主要是软件行为到环境作用的对应关系问题.如果有系统运行数据,那么利用深度学习、规则学习,可以给出操作-结果的预测函数;如果缺乏运行数据,则需要通过模拟的方式,利用强化学习,从模拟系统的交互中学习到对应关系[53,78,95].在既缺乏系统数据,又缺乏模拟手段时,不得不在运行时进行学习,这就会引入学习算法的安全性的问题,这一点将在 5.1 继续探讨.

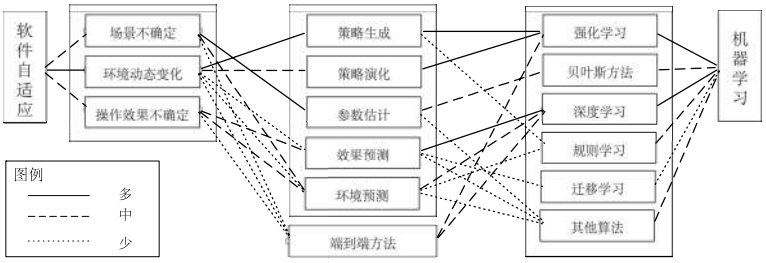


Fig.22 From software self-adaptation to machine learning enabled techniques

图 22 自适应关注点与机器学习赋能技术的关系

5 存在的不足与未来展望

本节在前文分析的基础上,总结机器学习赋能的软件自适应性所存在的问题,并对未来研究进行展望.本节回答研究问题 RQ3.

5.1 当前研究的不足

过去十余年,研究者将机器学习的方法应用于自适应软件系统,提出了及其学习赋能的系统自适应机制以算法和框架.但现有及其学习赋能的系统自适应性研究还存在以下主要问题:

- **缺乏对系统自适应性任务的提炼**.正确应用机器学习的工具,需要充分了解机器学习算法和模型的基本假设及其适用场景,并了解软件自适应性的关注点和目标.例如,强化学习往往适宜于效用明确的场景,而在处理多目标问题上有所限制[101].神经网络/深度学习的方法往往适宜于离线阶段的训练,而不太适合在线学习[99].马尔可夫决策过程通常可以刻画出系统、环境的动态变化特点,而很难完成其他方面的任务[37].目前研究工作的视角多以解决问题为主,而缺乏对“具有哪种特点的自适应任务,适合采用哪种机器学习算法?这类任务采用机器学习算法后,有哪些变化?为什么会产生这样的变化?”等方法学问题的深入探索,因此,尽管软件系统自适应问题丰富、机器学习算法多样,但大多数工作都仅仅集中在基于策略的软件自适应任务(包括策略演化和策略生成等)和强化学习、深度学习一两类算法的应用,而缺少在系统策略建模、系统设计架构、系统运维管理等上的系统化机器学习赋能研究.
- **机器学习算法能力的不足**.体现在两个方面:第一,目前应用于自适应的机器学习算法,大多着重于满足自适应软件的功能性需求,实现具体的功能指标,并没有考虑到自适应任务在系统可靠性、可用性、稳定性、安全性等非功能需求,尤其在安全攸关的场景,缺乏对采用机器学习的自适应系统的验证和测试.第二,采用机器学习算法有时会给自适应软件系统带来了一些负面效果,包括:缺乏有效的训练数据,从而影响机器学习模型的效果.包括,(1)端到端方法、环境预测、效果预测等方法,将软件系统视为黑盒,缺乏对系统内部的分析,来自于软件工程领域不熟悉机器学习算法的普通开发者很难对这类系统进行开发、分析、验证.(2)存在“在线-离线鸿沟”,机器学习的模型选择很大程度依赖于设计者对数据分布的先验知识或基本假设,而由于这些先验知识或基本假设在实际情况中可能并不成立,因此在离线阶段设计、训练出的机器学习模型可能不适合在线的实际情况.(3)存在在线算法的缺陷,机器学习算法本身具有不确定性(例如强化学习的 ε -greedy 机制),容易使系统进入危险状态,破坏系统本身或给系统所处的环境带来伤害.(4)常规的模型检测的方法很难对采用了机器学习算法的软件自适应系统进行验证.(5)当多个采用机器学习算法的自适应系统协作时,会出现学习算法不收敛、合作效率低下等问题.这些都是未来需要解决的问题
- **缺乏在机器学习赋能思想上的自适应系统框架**.目前绝大多数研究工作,其基本思想只是从软件自适应任务中找特定的问题,把机器学习作为解决问题的工具,机器学习思想没有对自适应软件系统的框架体系带来影响,软件系统的自适应需求也没有引起机器学习技术的新的发展方向.缺乏从机器学习的角度分析,是否应当把考虑学习能力作为自适应系统的一个需求目标,怎样赋予自适应系统学习的能力等问题.参考基于控制论的软件自适应性,可以发现机器学习并没有影响或改变自适应软件系统本身的架构,软件自适应性的研究并没有将机器学习中“通过经验来提升性能”的核心思想囊括在内,仅仅是把自适应作为机器学习算法的背景问题,把机器学习算法作为自适应的工具是远远不够的,需要建立成熟的机器学习赋能的自适应软件系统的体系架构,以及促进面向软件系统自适应性的在线学习技术的发展.
- **缺乏机器学习赋能的自适应软件系统开发方法学**.现有的工作大都是“就事论事”的研究方法,缺乏系统化的软件设计方法学,缺少对系统的很多关键非功能需求的可满足性研究,因而目前基本都仅仅在实验环境下的研究,并没有真正应用到实践中.理想的从软件自适应问题到机器学习的过程应在系统

的方法论的指导下,建立起从软件自适应问题到学习任务的映射,再根据提炼出的约束条件,得到对于“在某种约束条件下,进行某种学习任务”的最合适的机器学习赋能机制.

5.2 研究展望

基于上述的分析总结,学习赋能的系统自适应性的研究下一步急需的研究应侧重在如下几个方面:

- **更加适合系统自适应性的机器学习算法设计.**基于现有的自适应任务,改进或提出新的机器学习算法以更好地满足自适应任务的需求.机器学习的应用给软件自适应带来好处的同时也带来了诸多副作用.因此,要结合具体应用场景,对机器学习算法进行改造,使之更加适应于软件自适应的具体应用需求.有价值的研究问题包括:解决强化学习算法在线学习时,探索策略空间的无效率的问题(学习效率、容易进入危险状态、没有利用已有的先验知识等);提高机器学习算法鲁棒性,使得系统中采用了学习算法的部分具有更高的可靠性;提高学习算法的样本利用效率,使得在缺乏数据的软件自适应场景下,学习算法也能被很好地应用.
- **软件系统自适应性对机器学习的能力需求分析**.基于现有的自适应软件系统体系结构,对软件自适应性需求的本质进行更详细准确的分类和特性提炼,进而更好地定位软件自适应亟待解决的具体问题,从而更精准地将自适应问题映射为现有机器学习方法能解决的问题.软件自适应和机器学习领域存在知识和术语的鸿沟,科学的自适应需求分析为两个领域搭建起桥梁,有助于不熟悉机器学习方法的软件开发者使用机器学习方法解决自适应问题,也有助于机器学习研究者改进机器学习算法以更好满足自适应需求.
- **融合机器学习能力的自适应系统体系结构**.随着自适应系统应用面的不断推广,赋予软件系统“学习能力”使之能够充分应对复杂多变的环境,根据运行时的经验,自动地改进系统功能,提升系统性能,这将成为一大趋势.引入机器学习方法后,需要一个新的自适应系统体系结构.需要回答包括如下问题:学习和自适应性的关系是什么?学习能力的具体需求是什么?需要在现有的体系结构下增加学习模块/层次吗?需要以机器学习为中心扩展出新的体系结构吗?学习模块内部应该如何设计?学习模块外部接口是什么,机器学习和其他层次或功能模块如何交互?等等一系列富有挑战性的问题.如同基于控制论的系统自适应提供了各种源自于标准控制系统的体系结构一样,基于机器学习的系统自适应也应当推出带有学习特性的新的系统体系结构.

6 结论

软件系统自适应性是当前软件领域的热点话题.本文综述现有的机器学习赋能的软件系统自适应的相关工作.围绕 3 个研究问题,分别从自适应系统的角度和机器学习的角度对相关工作进行分类,基于分类框架,发掘从系统自适应性到机器学习问题求解的流程,梳理二者的对应关系,最后在此基础上提出不足与展望.

References:

[1] De Lemos R, Giese H, Müller H A, et al. Software engineering for self-adaptive systems: A second research roadmap. Software Engineering for Self-Adaptive Systems II. Berlin, Heidelberg: Springer, 2013. 1-32.

[2] Salehie M, Tahvildari L. Self-adaptive software: Landscape and research challenges. ACM Transactions on Autonomous and Adaptive Systems, 2009, 4(2).

[3] Kitchenham B. Procedures for performing systematic reviews. Keele, UK: Keele University, 2004. 1-26.

[4] Yang Q L, Ma X X, Xing J C, et al. Software Self-Adaptation: Control Theory Based Approach. Chinese Journal of Computers, 2016, 39(11):2189-2215.

[5] Barrett E, Howley E, Duggan J. Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. Concurrency and Computation: Practice and Experience, 2013, 25(12): 1656-1674.

[6] Brun Y, Serugendo G D, Gacek C, et al. Engineering Self-Adaptive Systems through Feedback Loops. Lecture Notes in Computer Science, 2009: 48-70.

[7] Kephart, J.O. Chess, D. M., The vision of autonomic computing. IEEE Computer, 2003,36(1): 41-50.

[8] Garlan, D., Cheng, S.-W., Huang, A.-C., Schmerl, B., Steenkiste, P. Rainbow: architecture-based self-adaptation with reusable infrastructure. IEEE Computer,2004, 37 (10): 46-54.

[9] Müller H, Pezzè M, Shaw M. Visibility of control in adaptive systems. Proceedings of the 2nd international workshop on Ultra-large-scale software-intensive systems. Leipzig, Germany: ACM, 2008. 23-26.

[10] Shen J., Wang Q., Mei H., Self-Adaptive Software: Cybernetic Perspective and an Application Server Supported Framework. Computer Software and Applications Conference. IEEE Computer Society, 2004. 92-95.

[11] Silva Souza V E, Lapouchnian A, Robinson W N, et al. Awareness requirements for adaptive systems. Proceedings of the 6th international symposium on Software engineering for adaptive and self-managing systems. Waikiki, USA: ACM, 2011. 60-69.

[12] Karlsson M., Karamanolis C., Zhu X., Triage: Performance differentiation for storage systems using adaptive control. ACM Transactions on Storage, 2005, 1(4):457-480.

[13] Hu H, Jiang C H, Cai K Y. Adaptive software testing in the context of an improved controlled Markov chain model. 32nd Annual IEEE International Computer Software and Applications Conference. IEEE, 2008. 853-858.

[14] Liu X, Zhu X, Padala P, et al. Optimal multivariate control for differentiated services on a shared hosting platform. 46th IEEE Conference on Decision and Control. IEEE, 2007. 3792-3799.

[15] Litoiu M, Woodside M, Zheng T. Hierarchical model-based autonomic control of software systems. ACM SIGSOFT Software Engineering Notes. Louis, USA: ACM, 2005. 30(4): 1-7.

[16] Weyns D, Malek S, Andersson J. FORMS: a formal reference model for self-adaptation. Proceedings of the 7th international conference on Autonomic computing. Washington, DC, USA: ACM, 2010. 205-214.

[17] Zhang J, Cheng B H C. Model-based development of dynamically adaptive software. Proceedings of the 28th international conference on Software engineering. Shanghai, China: ACM, 2006. 371-380.

[18] Liu C, Jiang C, Hu H, et al. A Control-Based Approach to Balance Services Performance and Security for Adaptive Service Based Systems (ASBS). 33rd Annual IEEE International Computer Software and Applications Conference. IEEE, 2009. 2: 473-478.

[19] Elkhodary A, Esfahani N, Malek S. FUSION: a framework for engineering self-tuning self-adaptive software systems. Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering. Santa Fe, USA: ACM, 2010. 7-16.

[20] Baah G K, Gray A, Harrold M J. On-line anomaly detection of deployed software: a statistical machine learning approach. Proceedings of the 3rd international workshop on Software quality assurance. Portland, USA: ACM, 2006. 70-77.

[21] Rodrigues A, Caldas R D, Rodrigues G N, et al. A learning approach to enhance assurances for real-time self-adaptive systems. IEEE/ACM 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). Gothenburg, Sweden: IEEE, 2018. 206-216.

[22] Tesauro G. Reinforcement Learning in Autonomic Computing: A Manifesto and Case Studies.Internet Computing. IEEE Internet Computing, 2007, 11(1):22-30.

[23] Kim D, Park S. Reinforcement learning-based dynamic adaptation planning method for architecture-based self-managed software. ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems. Vancouver, Canada: IEEE, 2009. 76-85.

[24] Wang H, Zhou X, Zhou X, et al. Adaptive service composition based on reinforcement learning. International Conference on Service-Oriented Computing. Berlin, Heidelberg: Springer, 2010. 92-107.

[25] Mitchell T. Machine Learning. New York: McGraw Hill, 1997.

[26] Sutton R, Barto A . Reinforcement Learning: An Introduction. Massachusetts: MIT Press, 1998.

[27] Lecun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553):436.

[28] Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

[29] Schulman J, Levine S, Abbeel P, et al. Trust region policy optimization. International Conference on Machine Learning Lille, France: JMLR, 2015. 1889-1897.

[30] Heckerman D. A tutorial on learning with Bayesian networks. Dordrecht: Springer, 1998. 301-354.

[31] Koller D, Friedman N. Probabilistic Graphical Models: Principles and Techniques. Massachusetts: MIT Press, 2009.

[32] Fürnkranz J, Gamberger D, Lavrač N. Foundations of rule learning. Springer Science & Business Media, 2012.

[33] Sutton R S. Temporal Credit Assignment in Reinforcement Learning. [Ph.D. thesis]. Amherst: University of Massachusetts,1984.

[34] Ho H N, Lee E. Model-based reinforcement learning approach for planning in self-adaptive software system. Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication. BALI, Indonesia: ACM, 2015.

[35] Zhao T, Zhang W, Zhao H, et al. A reinforcement learning-based framework for the generation and evolution of adaptation rules. IEEE International Conference on Autonomic Computing (ICAC). Columbus, USA: IEEE, 2017. 103-112.

[36] Tesauro G, Jong N K, Das R, et al. A hybrid reinforcement learning approach to autonomic resource allocation. IEEE International Conference on Autonomic Computing. Dublin, Ireland: IEEE, 2006. 65-73.

[37] Pandey A, Moreno G A, Cámara J, et al. Hybrid planning for decision making in self-adaptive systems. IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO). Augsburg, Germany: IEEE, 2016. 130-139.

[38] Ramirez A J, Jensen A C, Cheng B H C. A taxonomy of uncertainty for dynamically adaptive systems. Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. Zurich: IEEE, 2012. 99-108.

[39] Alrajeh D, Kramer J, Russo A, et al. Learning operational requirements from goal models. Proceedings of the 31st international conference on software engineering. IEEE Computer Society, 2009. 265-275.

[40] Guivarch V, Camps V, Péninou A. AMADEUS: an adaptive multi-agent system to learn a user's recurring actions in ambient systems. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal, 2013, 1(3): 1-10.

[41] Liu Q, Sun Y, Zhang S. A scalable web service composition based on a strategy reused reinforcement learning approach. Eighth Web Information Systems and Applications Conference. IEEE, 2011. 58-62.

[42] Nigon J, Gleizes M P, Migeon F. Self-adaptive model generation for ambient systems. Procedia Computer Science, 2016, 83: 675-679.

[43] Jamshidi P, Velez M, Kästner C, et al. Transfer learning for improving model predictions in highly configurable software. 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. Buenos Aires: IEEE, 2017. 31-41.

[44] Gu X. IDES: Self-adaptive Software with Online Policy Evolution Extended from Rainbo. Computer and Information Science 2012. Berlin, Heidelberg: Springer, 2012. 181-195.

[45] Lobo J, Bhatia R, Naqvi S. A policy description language. National conference on artificial intelligence. AAAI, 1999. 291-298.

[46] Hu Shou-Song. Principles of Automatic Control. 6th Edition. Beijing: Science Press. 2013.

[47] Fleurey F, Dehlen V, Bencomo N, et al. Modeling and validating dynamic adaptation. International Conference on Model Driven Engineering Languages and Systems. Berlin, Heidelberg: Springer, 2008. 97-108.

[48] Esfahani N, Elkhodary A, Malek S. A learning-based framework for engineering feature-oriented self-adaptive software systems. IEEE transactions on software engineering, 2013, 39(11): 1467-1493.

[49] Borges R V, d'Avila Garcez A, Lamb L C, et al. Learning to adapt requirements specifications of evolving systems (NIER track). Proceedings of the 33rd International Conference on Software Engineering. Waikiki, USA: ACM, 2011. 856-859.

[50] Schmid S, Gerostathopoulos I, Prehofer C, et al. Self-adaptation based on big data analytics: a model problem and tool. IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). IEEE, 2017: 102-108.

[51] Verstaevel N, Jérémy Boes, Nigon J, et al. Lifelong Machine Learning with Adaptive Multi-Agent Systems. International Conference on Agents & Artificial Intelligence, 2017. 275-286.

[52] Tanabe M, Tei K, Fukazawa Y, et al. Learning environment model at runtime for self-adaptive systems. Proceedings of the Symposium on Applied Computing. Marrakech, Morocco: ACM, 2017. 1198-1204.

[53] Lama P, Zhou X. Autonomic provisioning with self-adaptive neural fuzzy control for percentile-based delay guarantee. ACM Transactions on Autonomous and Adaptive Systems (TAAS), 2013, 8(2): 9.

[54] Porter B, Rodrigues Filho R. Losing Control: The Case for Emergent Software Systems Using Autonomous Assembly, Perception, and Learning. IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO). Augsburg, Geramany: IEEE, 2016. 40-49.

[55] Bodík P, Griffith R, Sutton C A, et al. Statistical Machine Learning Makes Automatic Control Practical for Internet Datacenters. IEEE international conference on cloud computing technology and science. IEEE, 2009.

[56] Han D, Xing J, Yang Q, et al. Handling uncertainty in self-adaptive software using self-learning fuzzy neural network. IEEE 40th Annual Computer Software and Applications Conference (COMPSAC). IEEE, 2016. 2: 540-545.

[57] Mirko D'Angelo, Simos Gerasimou, Sona Ghahremani, Johannes Grohmann, Ingrid Nunes, Evangelos Pournaras, Sven Tomforde. On learning in collective self-adaptive systems: state of practice and a 3D framework. IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). Montreal, Canada: IEEE, 2019. 13-24.

[58] Canavera K R, Esfahani N, Malek S. Mining the execution history of a software system to infer the best time for its adaptation. Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. Cary: ACM, 2012.

[59] Lama P, Zhou X. Autonomic provisioning with self-adaptive neural fuzzy control for end-to-end delay guarantee. IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. IEEE, 2010. 151-160.

[60] Jamshidi P, Siegmund N, Velez M, et al. Transfer learning for performance modeling of configurable systems: An exploratory analysis. Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering. Urbana: IEEE, 2017. 497-508.

[61] Paucar L H G, Bencomo N. RE-STORM: mapping the decision-making problem and non-functional requirements trade-off to partially observable markov decision processes. Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems. Gothenburg, Sweden: ACM, 2018. 19-25.

[62] Filieri A, Grunske L, Leva A. Lightweight adaptive filtering for efficient learning and updating of probabilistic models. Proceedings of the 37th International Conference on Software Engineering. Florence, Italy: IEEE, 2015. 200-211.

[63] Patikirikorala T, Colman A, Han J, et al. A systematic survey on the design of self-adaptive software systems using control engineering approaches. 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. IEEE, 2012: 33-42.

[64] Wang H, Wu Q, Chen X, et al. Integrating gaussian process with reinforcement learning for adaptive service composition. International Conference on Service-Oriented Computing. Berlin, Heidelberg: Springer, 2015. 203-217.

[65] Wang H, Wu Q, Chen X, et al. Adaptive and dynamic service composition via multi-agent reinforcement learning. IEEE International Conference on Web Services. IEEE, 2014. 447-454.

[66] Moreno G A, Cámara J, Garlan D, et al. Efficient decision-making under uncertainty for proactive self-adaptation. IEEE International Conference on Autonomic Computing (ICAC). IEEE, 2016. 147-156.

[67] Pan S J, Yang Q. A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 2009, 22(10): 1345-1359.

[68] Ray O. Using abduction for induction of normal logic programs. Proceedings of 2nd Workshop on AIAI and Scientific Modelling. 2006. 28-31.

[69] Muggleton S H, Bryant C H. Theory completion using inverse entailment. International conference on inductive logic programming. Berlin, Heidelberg: Springer, 2000. 130-146.

[70] P. E. Petruzzi, J. Pitt, and D. Busquets. Inter-institutional social capital for self-organising 'nested enterprises'. 10th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO). Augsburg, Germany: IEEE, 2016. 90–99.

[71] Floriana Esposito, Giovanni Semeraro, Nicola Fanizzi, Stefano Ferilli. Multistrategy Theory Revision: Induction and Abduction in INTHELEX. Machine Learning, 2000, 38 (1-2) : 133-156.

[72] Braberman V, D'Ippolito N, Kramer J, et al. Morph: A reference architecture for configuration and behaviour self-adaptation Proceedings of the 1st International Workshop on Control Theory for Software Engineering. Bergamo, Italy: ACM, 2015. 9-16.

[73] Ghahremani S, Adriano C M, Giese H. Training prediction models for rule-based self-adaptive systems. IEEE International Conference on Autonomic Computing (ICAC). IEEE, 2018: 187-192.

[74] Rook A, Knauss A, Damian D, et al. A case study of applying data mining to sensor data for contextual requirements analysis. IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering (AIRE). Karlskrona, Sweden: IEEE, 2014. 43-50.

[75] Huber N, Walter J, Bähr M, et al. Model-based autonomic and performance-aware system adaptation in heterogeneous resource environments: A case study. International Conference on Cloud and Autonomic Computing. IEEE, 2015. 181-191.

[76] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning. International Conference on Learning Representations. San Diego, CA. 2016.

[77] Mahfoudh H B, Serugendo G D M, Boulmier A, et al. Coordination model with reinforcement learning for ensuring reliable on-demand services in collective adaptive systems. International Symposium on Leveraging Applications of Formal Methods. Cham: Springer, 2018. 257-273.

[78] Zhao T. The generation and evolution of adaptation rules in requirements driven self-adaptive systems. IEEE 24th International Requirements Engineering Conference (RE). Beijing, China: IEEE, 2016. 456-461.

[79] Epifani I, Ghezzi C, Mirandola R, et al. Model evolution by run-time parameter adaptation. Proceedings of the 31st International Conference on Software Engineering. Vancouver, Canada: IEEE Computer Society, 2009. 111-121.

[80] Knauss A, Damian D, Franch X, et al. ACon: A learning-based approach to deal with uncertainty in contextual requirements at runtime. Information and software technology, 2016, 70: 85-99.

[81] Baresi L, Pasquale L. Adaptive goals for self-adaptive service compositions. IEEE International Conference on Web Services. IEEE, 2010. 353-360.

[82] Bencomo N, Belaggoun A. Supporting decision-making for self-adaptive systems: from goal models to dynamic decision networks. International Working Conference on Requirements Engineering: Foundation for Software Quality. Berlin: Springer, 2013. 221-236.

[83] Shoham Y, Leyton-Brown K. Multiagent systems: Algorithmic, game-theoretic, and logical foundations. London: Cambridge University Press, 2008.

[84] Bowling M, Veloso M. Multiagent learning using a variable learning rate. Artificial Intelligence, 2002, 136(2): 215-250.

[85] Jamshidi P, Sharifloo A, Pahl C, et al. Fuzzy self-learning controllers for elasticity management in dynamic cloud architectures. 12th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA). IEEE, 2016: 70-79.

[86] Bencomo N, Belaggoun A, Issarny V. Dynamic decision networks for decision-making in self-adaptive systems: a case study. Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. Francisco, USA: IEEE Press, 2013. 113-122.

[87] Cecchet E, Marguerite J, Zwaenepoel W, et al. Performance and scalability of EJB applications. Conference on Object-oriented Programming Systems, Languages, and Applications. ACM, 2002. 37(11): 246-261.

[88] Chen T, Bahsoon R. Self-adaptive and online qos modeling for cloud-based software services. IEEE Transactions on Software Engineering, 2016, 43(5): 453-475.

[89] S. Rudolph, S. Tomforde, B. Sick, and J. Ha¨hner. A mutual influence detection algorithm for systems with local performance measurement. IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems. IEEE, 2015. 144–149.

[90] Kitano H, Asada M, Kuniyoshi Y, et al. RoboCup: A Challenge Problem for AI and Robotics. Robot Soccer World Cup, 1998. 1-19.

[91] Quigley M, Brain Gerkey, Ken Conley, et al. ROS: an open-source Robot Operating System. International Conference on Robotics and Automation, 2009.

[92] Klös V, Göthel T, Glesner S. Comprehensible and dependable self-learning self-adaptive systems. Journal of Systems Architecture, 2018, 85: 28-42.

[93] Sharifloo A M, Metzger A. Mcaas: Model checking in the cloud for assurances of adaptive systems. Software Engineering for Self-Adaptive Systems III. Assurances. Cham: Springer, 2017. 137-153.

[94] Ismail A, Cardellini V. Decentralized planning for self-adaptation in multi-cloud environment. European Conference on Service-Oriented and Cloud Computing. Cham: Springer, 2014. 76-90.

[95] Kishan Kumar Ganguly, Kazi Sakib. A Reusable Adaptation Component Design for Learning-Based Self-Adaptive Systems. The Twelfth International Conference on Software Engineering Advances (ICSEA), Tahiti: IEEE, 2017. 244-249.

[96] Magableh B. A Deep Recurrent Q Network towards Self-adapting Distributed Microservices architecture. arXiv preprint arXiv:1901.04011, 2019.

[97] Rodrigues A, Rodrigues G N, Knauss A, et al. Enhancing context specifications for dependable adaptive systems: A data mining approach. Information and Software Technology, 2019, 112: 115-131.

[98] Bahati R M, Bauer M A. Towards adaptive policy-based management. IEEE Network Operations and Management Symposium (NOMS). IEEE, 2010. 511-518.

[99] Calinescu R, Rafiq Y, Johnson K, et al. Adaptive model learning for continual verification of non-functional properties. Proceedings of the 5th ACM/SPEC international conference on Performance engineering. New York: ACM, 2014. 87-98.

[100] Sykes D, Corapi D, Magee J, et al. Learning revised models for planning in adaptive systems. 35th International Conference on Software Engineering (ICSE). San Francisco: IEEE, 2013. 63-71.

[101] Moustafa A, Zhang M. Multi-objective service composition using reinforcement learning. International Conference on Service-Oriented Computing. Berlin, Heidelberg: Springer, 2013. 298-312.

[102] Boes J, Migeon F, Glize P, et al. Model-free optimization of an engine control unit thanks to self-adaptive multi-agent systems. Enbedded Real Time Software and Systems, OATAO, 2014.

[103] Boes J, Nigon J, Verstaevel N, et al. The self-adaptive context learning pattern: Overview and proposal. International and Interdisciplinary Conference on Modeling and Using Context. Cham: Springer, 2015. 91-104.

[104] Rao J, Bu X, Xu C Z, et al. A distributed self-learning approach for elastic provisioning of virtualized cloud resources. 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems. IEEE, 2011. 45-54.

[105] Amoui M, Salehie M, Mirarab S, et al. Adaptive action selection in autonomic software using reinforcement learning. Fourth International Conference on Autonomic and Autonomous Systems (ICAS). IEEE, 2008. 175-181.

[106] Braberman V, D'Ippolito N, Kramer J, et al. An Extended Description of MORPH: A Reference Architecture for Configuration and Behaviour Self-Adaptation. Software Engineering for Self-Adaptive Systems III. Assurances. Cham: Springer, 2017. 377-408.

[107] Yan Y, Zhang B, Guo J. An adaptive decision making approach based on reinforcement learning for self-managed cloud applications. IEEE International Conference on Web Services (ICWS). IEEE, 2016. 720-723.

[108] Dowling J, Cahill V. Self-managed decentralised systems using K-components and collaborative reinforcement learning. Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems. Newport Beach, USA: ACM, 2004. 39-43.

[109] Porter B, Grieves M, Rodrigues Filho R, et al. REX: A Development Platform and Online Learning Approach for Runtime Emergent Software Systems. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). USENIX, 2016. 333-348.

[110] Jamshidi P, Sharifloo A M, Pahl C, et al. Self-learning cloud controllers: Fuzzy q-learning for knowledge evolution. International Conference on Cloud and Autonomic Computing. IEEE, 2015. 208-211.

[111] Wang H, Chen X, Wu Q, et al. Integrating on-policy reinforcement learning with multi-agent techniques for adaptive service composition. International Conference on Service-Oriented Computing. Berlin, Heidelberg: Springer, 2014. 154-168.

[112] Bahati R M, Bauer M A. Reinforcement learning in policy-driven autonomic management. IEEE Network Operations and Management Symposium (NOMS). IEEE, 2008. 899-902.

[113] Berral J L, Gavalda R, Torres J. Adaptive scheduling on power-aware managed data-centers using machine learning. Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing. IEEE Computer Society, 2011. 66-73.

[114] Tesauro G. Online resource allocation using decompositional reinforcement learning. AAAI Conference on Artificial Intelligence (AAAI). AAAI, 2005. 5: 886-891.

[115] Zhao T, Zan T, Zhao H, et al. Integrating Goal Model into Rule-Based Adaptation. 23rd Asia-Pacific Software Engineering Conference (APSEC). IEEE, 2016. 289-296.

[116] Dezfuli Z T, Nazemi E, Ziari M. An algorithm for online planning to improve availability and performance of self-adaptive websites. 5th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS). IEEE, 2017: 213-218.

[117] Klös V, Göthel T, Glesner S. Be Prepared: Learning Environment Profiles for Proactive Rule-Based Production Planning. 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE, 2018. 89-96.

[118] Zavala E, Franch X, Marco J, et al. SACRE: Supporting contextual requirements' adaptation in modern self-adaptive systems in the presence of uncertainty at runtime. Expert Systems with Applications, 2018, 98: 166-188.

[119] Ding Z, Zhou Y, Zhou M C. Modeling self-adaptive software systems with learning Petri nets. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2015, 46(4): 483-498.

[120] Gil R. Automated Planning for Self-Adaptive Systems. IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE). ACM, 2015. 839-842.

[121] Krupitzer C, Pfannemüller M, Kaddour J, et al. SATISFy: Towards a self-learning analyzer for time series forecasting in self-improving systems. IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS* W). IEEE, 2018. 182-189.

[122] Leitner P, Hummer W, Dustdar S. Cost-based optimization of service compositions. IEEE Transactions on Services Computing, 2011, 6(2): 239-251.

[123] W. Richert and B. Kleinjohann. Adaptivity at every layer: A modular approach for evolving societies of learning autonomous systems. ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). ACM, 2008. 113–120.

附中文参考文献:

[4] 杨启亮, 马晓星, 邢建春, 胡昊, 王平, 韩德帅. 软件自适应:基于控制理论的方法.计算机学报, 2016,39(11):2189-2215.

[46] 胡寿松. 自动控制原理.第六版.北京:科学出版社, 2013.

附表 1 基于机器学习的软件自适应论文分类总结

分类 / 文献	机器学习视角			软件自适应视角						
	应用方式		学习算法	被控对象	监控内容	自适应目标	引起自适应学习的原因	决策方式	自适应操作	验证评价
	端到端	阶段/过程								
19	0	P4	L2 L6	C1	M2 M3	G2	R1 R4 R6	D1	E3	V2
23	0	P1 P2	L1	C2	M1 M2	G3	R3 R6	D2	E3	V2
43	0	P4	L5	C2	M1 M2	G3	R1 R2 R5 R6	D1	E2 E3	V2
34	E	0	L1	C1	M2	G3	R1 R6	D2	E1 E2	V2
98	E	0	L1	C1	M1 M2	G3	R1 R5 R6	D2	E2	V2
48	0	P4	L3	C1	M2 M3	G2	R1 R4 R6	D1	E3	V2
20	0	P3	L6	C1	M2 M3	G1 G3	R2	D1	E2	V2
60	0	P4	L5	C2	M1 M2	G3	R1 R2 R5 R6	D1	E2 E3	V2
100	0	P3 P5	L4	C2	M1 M2	G3	R5 R6	D2	E3	V2
44	0	P1 P2	L1	C1	M1 M2	G3	R1 R6	D2	E3	V2
80	0	P5	L4	C2	M1	G3	R1 R4 R5 R6	D1	E1	V3
72	0	P1	L4 L6	C2	M1 M2 M3	G1 G2	R1 R2	D3	E3	V2
106	0	P1	L4 L6	C2	M1 M2 M3	G1 G2	R1 R2	D3	E3	V2
39	0	P1	L4	C3	M2 M3	G3	R2 R4	D1	E1	V2
49	0	P3	L3 L4	C2	M1 M3	G1 G3	R1 R4 R6	D1	E1	V2
50	E	0	L3	C3	M2	G3	R2 R6	D2	E1	V3
21	0	P1 P2	L4 L6	C3	M1 M2	G3	R1 R5 R6	D1	E1 E2	V3
92	0	P1 P2	L4	C3	M1	G3	R1 R2	D3	E2 E3	V2
77	E	0	L1	C3	M1	G3	R1 R6	D2	E1 E2	V2
51	E	0	L1 L3	C3	M1	G3	R1	D2	E2	V2
52	0	P2	L3	C2	M1	G3	R1 R5	D1	E1	V2
62	0	P3 P5	L3 L6	unknown	M1	unknown	R6	D1	E1	V1 V3
105	E	0	L1	C1	M1 M2	G3	R1 R6	D1	E3	V2
22	0	P1 P2	L1	C1	M1 M2	G3	R1	D2	E2	V3
85	0	P1 P4	L1	C1	M1 M2	G1 G3	R1 R2 R6	D2	E1 E2	V2

95	0	P4	L3	C1	M2	G2	R2	D1	E1	V2
64	E	0	L1 L2	C1	M1 M2	G3	R1 R6	D1	E2	V3
107	E	0	L1	C1	M2	G3	R2 R6	D2	E2	V2
65	E	0	L1	C1	M2	G3	R2	D1	E2	V3
24	E	0	L1	C1	M2	G3	R2	D2	E2	V3
35	0	P1 P2 P4	L1 L3	C1	M1 M2 M3	G2	R1 R2 R4 R5 R6	D2	E1	V3
61	0	P2 P3	L2	C2	M1 M3	G2	R1 R4 R5 R6	D2	E1	V2
37	0	P2	L1	C1	M1 M2	G3	R1 R5 R6	D1	E1 E3	V3
36	0	P1	L1 L3	C1	M1 M2	G3	R1 R2	D1	E2	V2
101	E	0	L1	C1	M1	G3	R1 R4 R5 R6	D1	E3	V3
53	0	P1 P2 P3	L3	C1	M1 M2	G1 G3	R1 R2 R5 R6	D3	E1	V3
59	0	P1 P2	L3	C1	M1 M2	G3	R1 R2 R6	D3	E1 E2	V3
86	E	0	L2	C1	M1 M2	G2	R1 R4 R5 R6	D2	E2	V2
66	0	P1 P5	L2	C1	M1 M2	G3	R1 R5 R6	D3	E1 E2	V3
108	0	P1	L1	unknown	M1 M2	G3	R1 R2 R6	D2	E3	unknown
79	0	P5	L2	C1	M1 M2	G3	R1 R2 R4 R5 R6	D3	E1	V3
63	0	P5	L2	C1	M1 M2	G3	R1 R2 R6	D1	E1	V3
93	0	P4 P5	L2 L3	C3	M1	G3	R1 R5 R6	D3	E2	V2
99	0	P3 P4	L2	C1	M2	G3	R2 R6	D3	E1	V3
54	0	P1 P2	L1 L3	C1	M2	G1 G3	R2 R6	D3	E3	V3
109	0	P1 P2	L1 L3	C1	M2	G3	R2 R6	D3	E3	V3
110	0	P1 P4	L1	C1	M1 M2	G3	R1 R6	D2	E3	V2
111	E	0	L1	C1	M1 M2	G3	R1 R2 R3	D1	E2	V3
55	0	P4	L3	C1	M2	G3	R2	D3	E1	V3
112	0	P1 P4	L1	C1	M2 M3	G2	R2 R4	D1	E1	V3
113	0	P1 P4	L3	C1	M2	G3	R2 R4	D2	E1	V3
114	0	P1	L1	C1	M1 M2	G3	R1	D1	E1	V3
82	0	P1 P2	L2	C2	M1 M2 M3	G2	R1 R2 R4 R5 R6	D2	E2	V2
102	0	P1	L1 L6	C2	M1 M2 M3	G1 G2	R1 R2 R3	D3	E1	V3
103	0	P1	L6	C2	M1 M2 M3	G1 G2	R1 R2 R3	D3	E2	V3
40	0	P5	L6	C3	M1 M2	G1	R1 R2 R3	D3	E2 E3	V3
42	0	P2	L6	C3	M1 M2	G1	R1 R2 R3	D3	E2	V3
41	0	P1	L1 L6	C1	M1 M3	G2	R1 R2 R3	D1	E1	V3
115	0	P1 P2	L1 L3	C1	M1 M2 M3	G2	R1 R2 R6	D2	E1	V3
116	0	P1	L1	C1	M1	G1 G3	R1 R6	D1	E2	V2
56	0	P3 P4	L3	C3	M1	G1 G3	R1 R4 R5 R6	D3	E1	V2
78	0	P1 P2	L1	C1	M1 M3	G2	R1 R4 R6	D2	E1	V3
73	0	P4	L6	C1	M1 M2	G3	R1 R2	D1	E3	V3
117	0	P1	L6	C3	M1	G1 G3	R1 R6	D1	E2	V3
118	0	P1 P4	L2 L3	C2	M1 M2 M3	G1 G2	R1 R2 R4 R5 R6	D2	E1 E2	V3
97	0	P4 P5	L4 L6	C3	M1	G3	R1 R5 R6	D3	E1 E3	V2
119	0	P3	L3 L6	C1	M1 M2	G1 G3	R2 R3	D3	E2	V3

88	0	P1 P4	L3	C1	M1 M2	G1 G3	R1 R5 R6	D3	E1	V3
104	E	0	L1 L6	C1	M1 M2	G3	R1 R2 R3	D1	E2	V3
120	0	P1	L6	C1	M1 M2	G3	R1 R5 R6	D3	E3	V2
121	0	P1 P2	L3 L6	C3	M1	G1 G3	R1 R6	D3	E1	V3
94	0	P1	L1 L6	C3	M1 M2	G3	R1 R2 R3 R6	D1	E3	V3
96	E	0	L1 L3	C1	M1 M2	G1 G3	R1 R4 R6	D1	E3	V3
122	E	0	L3 L6	C1	M1 M2	G3	R1 R2	D1	E2	V3
58	0	P1 P4 P5	L2 L6	C1	M1 M2	G1 G3	R1 R2	D1	E2	V3
74	0	P3	L6	C3	M1	G1 G3	R1 R4 R6	unknown	E1	V2
75	0	P5	L6	C1	M1 M2	G3	R1 R2	D3	E3	V2
5	E	0	L1	C1	M2	G3	R2 R5 R6	D1	E1 E2	V3

(表中符号的含义为:0 未使用;unknown 论文中不明确;E 端到端方法;P1 策略生成;P2 策略演化;P3 参数估计;P4 效果预测;P5 环境预测;L1 强化学习;L2 贝叶斯决策理论与概率图模型;L3 神经网络/深度学习;L4 规则学习;L5 迁移学习;L6 其他算法;C1 面向服务的信息系统;C2 机器人系统;C3 信息物理系统;M1 外部环境数据;M2 系统自身数据;M3 应用需求;G1 保持系统稳态;G2 最优化可变目标;G3 最优化不可变目标;R1 外部环境变化;R2 系统自身变化;R3 其他系统变化;R4 需求不确定性;R5 设计时不确定性;R6 运行时不确定性;D1 静态策略;D2 动态策略;D3 连续控制;E1 调整参数;E2 调整行为;E3 调整结构;V1 理论证明;V2 案例研究; V3 仿真模拟)