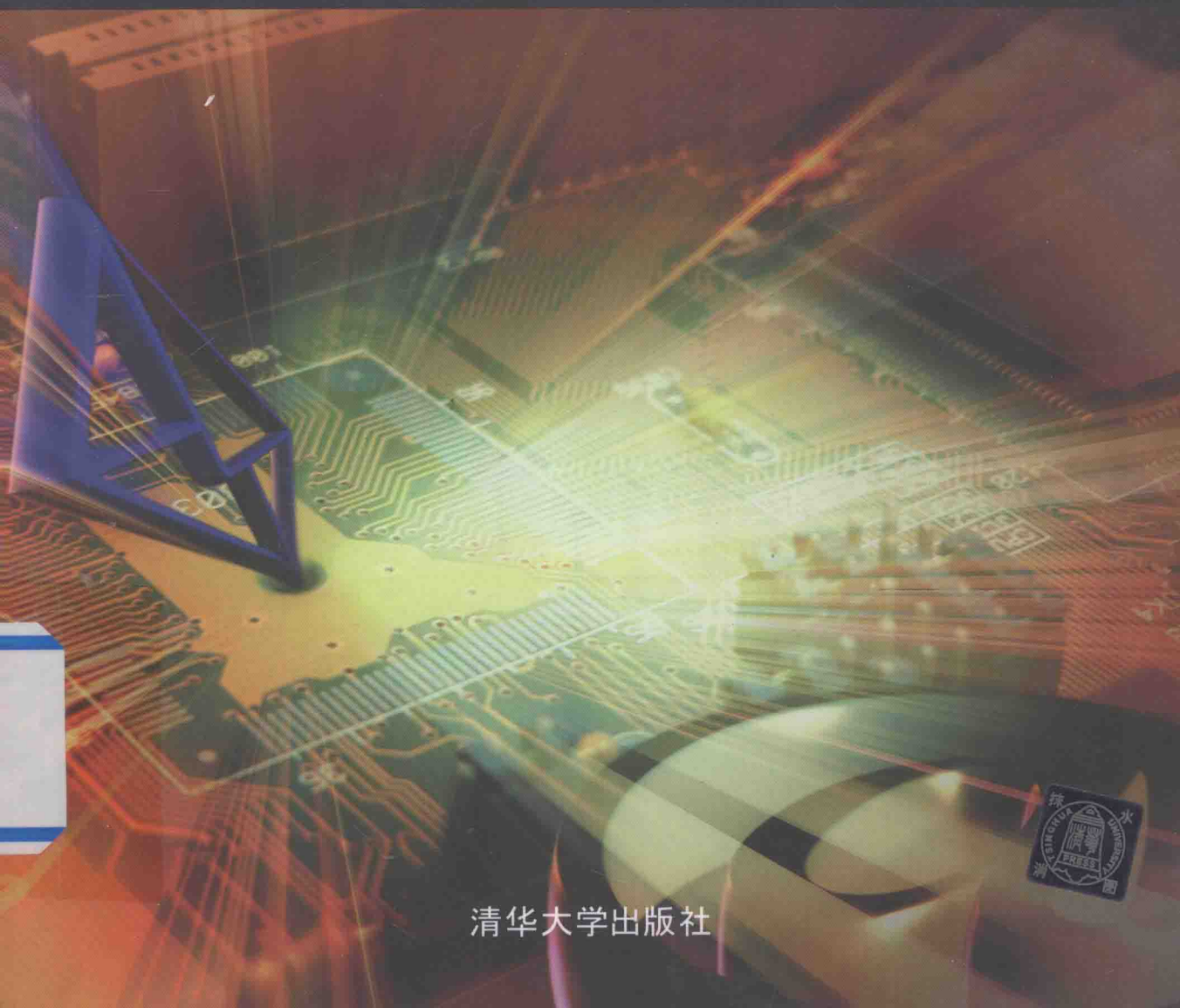
REINFORCEMENT LEARNING AND COORDINATION IN MULTIAGENT SYSTEMS

多主体强化学习协作策略研究

孙若莹（SUN RUOYING）
赵　刚（ZHAO　GANG）/著

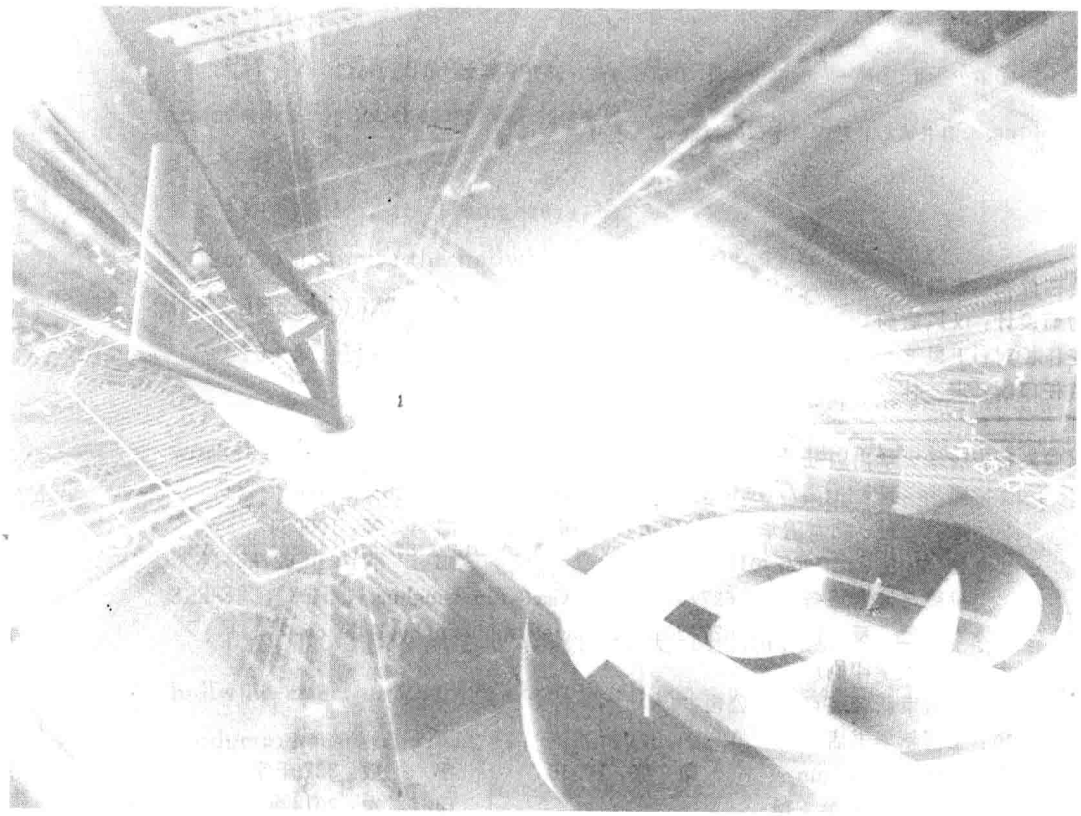


清华大学出版社

REINFORCEMENT LEARNING AND COORDINATION
IN
MULTIAGENT
SYSTEMS

多主体强化学习协作策略研究

孙若莹（SUN RUOYING）
赵　刚（ZHAO　GANG）/著



清华大学出版社
北京

内 容 简 介

多主体的研究与应用是近年来备受关注的热点领域，多主体强化学习理论与方法、多主体协作策略的研究则是该领域重要研究方向，其理论和应用价值极为广泛，备受广大从事计算机应用、人工智能、自动控制以及经济管理等领域研究者的关注。本书清晰地介绍了多主体、强化学习及多主体协作等基本概念和基础内容，明确地阐述了有关多主体强化学习、协作策略研究的发展过程及最新动向，深入地探讨了多主体强化学习与协作策略的理论与方法，具体地分析了多主体强化学习与协作策略在相关研究领域的应用方法。

全书脉络清晰、基本概念清楚、图表分析直观，注重内容的体系化和实用性。通过本书的阅读和学习，读者既可掌握多主体强化学习及协作策略的理论和方法，又可了解在实际工作中应用这些研究成果的手段。本书可作为从事计算机应用、人工智能、自动控制、以及经济管理等领域研究者的学习和阅读参考，同时高等院校相关专业研究生以及人工智能爱好者也可从中获得借鉴。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。
版权所有，侵权必究。侵权举报电话：010-62782989　13701121933

图书在版编目(CIP)数据

多主体强化学习协作策略研究：英文/孙若莹，赵刚著．—北京：清华大学出版社，2014
ISBN 978-7-302-36830-4

Ⅰ．①多…　Ⅱ．①孙…　②赵…　Ⅲ．①学习方法—研究—英文　Ⅳ．①G442

中国版本图书馆 CIP 数据核字(2014)第 134486 号

责任编辑：朱敏悦
封面设计：汉风唐韵
责任校对：王凤芝
责任印制：宋　林

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：清华大学印刷厂
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　张**：10.75　　**字　数**：222 千字
版　　次：2014 年 8 月第 1 版　　**印　次**：2014 年 8 月第1次印刷
印　　数：1～2000
定　　价：48.00 元

产品编号：057582-01

Preface

Researches on the multiagent and its applications have been paid attention in recent years. Reinforcement learning (RL) theory and coordination method in the multiagent system are important research subjects those have been attracting surprisingly attention from researches in computer science and its application, artificial intelligence, automatic control, and economy management, etc.

On the basis of our experience for years of research and the latest results on the relevant projects, such as the National Natural Science Foundation of China, Beijing Municipal Natural Science Foundation and so on, we present this academic book focusing on the aspects of reinforcement learning and coordination in the multiagent system. From an introduction to multiagent system, reinforcement learning, agents coordination, and other perspectives, a holistic view of the reinforcement learning and coordination in the multiagent system is delineated. Further, theories and methods concerned with reinforcement learning and coordination strategy in the multiagent system will be discussed deeply in the forthcoming chapters, and followed by a view of application to different realistic areas, respectively.

Based on the update policy of reinforcement values and the cooperative way of the indirect media communication, this book first presents a multiagent learning system, the Q-ACS learning method. Then, by investigating the active exploration mechanism and the unified action policy with exploration and exploitation in RL, utilizing indirect media communication, this book presents the T-ACS multiagent learning method. Further, this book studies the heterogeneous multiagent learning system, the D-ACS learning method that composites the learning policy of the Q-ACS learning and the T-ACS learning and takes different updating policies of reinforcement values. The agents in our methods are given a simply cooperating way exchanging information in the form of reinforcement values updated in the common model of all agents. Owning the advantages of exploring the unknown environment actively and exploiting learned knowledge effectively, the proposed methods are able to solve both problems with MDPs and combinatorial optimization problems effectively. Moreover, by investigating the action conversion mechanism, for the task under MDP, this book presents a multiagent RL algorithm, the Q-ac multiagent RL method that utilize both the direct communication and the indi-

rect media communication for learning agents to realize the cooperation. Besides per action one-step Q-learning, experience-replay and prioritized sweeping Q-value update are used to update reinforcement values in the Q-ac multiagent RL method. In addition, by investigating the swarm-based routing method and the multiagent RL applications, this book analyses the possibility and merit of adopting RL method in multicast routing protocol for mobile ad hoc networks, and presents a novel multicast routing method, the Q-MAP algorithm. The convergence and rationality of the Q-MAP method are analyzed from the point of view of RL. Based on successful application to dynamical domains, this book presents a multiagent coordination mechanism using reinforcement learning to the supply chain management, which derives better profit comparing with the typical policy in the stochastic supply chain.

We would like to express our gratitude to all the contributors for this book, and thank Tsinghua University press for helping in many ways.

Sun Ruoying

Zhao Gang

November 2013

此为试读,需要完整PDF请访问: www.ertongbook.com

Contents

Chapter 1 Introduction

With the successful applications in real world, machine learning has become more and more acceptable. Neural networks have been trained in recognition of handwriting and are superior to any other handwriting recognition system[1-4]. Decision trees leads to excellent classification of data even in the case that the underlying pattern is obviously meaningless and cost-sensitive decision task[5-7]. A more general method, which can learn structures such as neural networks or decision tree, is reinforcement learning[8-11]. In recent years, the multiagent reinforcement learning systems have received increasing attention in the artificial intelligence community[12-16]. Research in such systems involves the investigation of autonomous, rational and flexible behavior of entities and their interaction and coordination in such diverse areas as robotics, information retrieval, communication network traffic control, and supply chain management. In the first part of this book, we introduce reinforcement learning, multiagent system, and agents' interaction and coordination from a general viewpoint.

1.1 Reinforcement Learning

1.1.1 Generality of Reinforcement Learning

We begin with a general overview of the state-of-the-art in research in reinforcement learning and discuss the integration of learning methods in agent systems.

Reinforcement Learning (RL), which is currently an actively researched topic in Artificial Intelligence (AI), is a computational approach that an agent tries to maximize the total amount of reward it receives when interacting with a complex, dynamic environment by trial-and-error. RL[17-23] has been used to solve many complex tasks normally thought of as quite cognitive, for example, backgammon game[24], robotic soc-

cer[25][26], elevator problem[27], dynamic vehicle routing on goods distribution[28][29], supply chain ordering management[30][31] and so on[32-34].

RL requires learning from interactions in an environment in order to achieve certain goals. The entity interacting with its environment by actions is called agent. At each time step, an agent observes its environment and selects an action based on that observation. In the next time step, the agent obtains the new observation that may reflect the effects of its previous action and a reward indicating the quality of the selected action.

Dynamic Programming (DP)[35][36] solves state sequences optimization problems by solving recurrent relations instead of explicitly searching in the space of state sequences. In its most general form, DP applied to optimization problems in which the costs of objects in the search space have a compositional structure can be exploited to find an object of globally minimum cost without performing exhaustive search. Bellman[37] has introduced the discrete stochastic version of the optimal control problem known as Markov Decision Processes (MDPs), and Howard[38] has devised the policy iteration method for MDPs. All of these are essential elements underlying the theory and algorithms of modern RL. Although DP algorithms avoid exhaustive search in the state sequence space, they are still exhaustive by AI standards since they require repeated generation and expansion of all possible states. The current RL algorithm is to find optimal policies from experience without a priori model of an environment, or even no requiring the model of the environment. This is one of main innovation in RL algorithms for solving problems traditionally solved by DP. The learning agent interacts with its environment directly to obtain information which, by means of an appropriate algorithm, can be processed to produce an optimal policy. Thus, the two most important distinguishing characteristics of RL are: trial-and-error and delayed reward. A major influence on research leading to current RL algorithms is the method used by Samuel[39] to modify a heuristic evaluation for the game of checkers. Due to its compatibility with connective learning algorithms, DP approach has been refined and extended by Sutton[40] and uses heuristically in a number of single agent problem solving task. These algorithms are called Temporal Difference (TD) methods and have obtained some theoretical results about their convergence. Besides, researchers within RL field have also developed a number of different methods under RL, for example, Classifier Systems[41][42], and Q-learning. In the early 1980s, Barto have described a technique for addressing the temporal credit assignment problem, and then this method has culminated in Sutton's TD. In the late 1980s, using the term Incremental Dynamic Programming, Watkins has extended TD algorithms, devel-

此为试读,需要完整PDF请访问: www.ertongbook.com

oped Q-learning by explicitly utilizing the theory of DP for solving MDPs, and noted the approximate relationship between DP, TD, and Q-learning algorithms. The relevance of DP for planning and learning in AI has been articulated in Dyna-Q architecture[43].

In Q-learning, an agent's decision procedure is specified by a policy π that maps states into actions. The environmental feedback is defined by a reward function R that maps states into numerical rewards. The goal of Q-learning is to compute an optimal policy πfi that maximizes the reward an agent receives. This reward can be measured in several ways: discounted cumulative reward, finite horizon reward, and average reward. The discussions in this book are based on discounted cumulative reward Q-learning. Q-learning algorithm has been proven to converge towards the correct values under the condition that the reward values have an upper bound.

Though there are memory space problems to storing all Q-values in a table, the look-up table is often used since its simplicity. And a function approximation of Q-value, e. g. , Neural Network[44], can be used to solve the memory space problems, however, this approach may lead to a more complex update mechanism.

1.1.2 Reinforcement Learning on Markov Decision Processes

The learning agent needs the ability to observe its state in order to select an action using a policy. The states on the environment must contain all necessary information for the learning agent to make sense, which property is called the Markov property. A task domain with this property is called Markov Decision Process (MDP). The basic task for RL is MDP.

Assume that there are a finite number of states, s_1, s_2, …, s_n, a finite number of actions, a_1, a_2, …, a_n, and rewards, r_1, r_2, …, r_n, in the task environment. The Markov property for the RL problem is formally defined at the following. Consider how an environment responds at time $t+1$ to the action taken at time t. Generally, this response may depend on everything that has happened earlier. In this case, the dynamics can be defined by specifying the complete probability distribution:

$$\Pr\{s_{t+1}=s', r_{t+1}=r \mid s_t, a_t, r_t, s_{t1}, a_{t1}, \cdots, r_1, s_0, a_0\}, \tag{1.1}$$

for all s', r, and all possible values of the past events: s_t, a_t, r_t, …, r_1, s_0, a_0. If the environment's response at $t+1$ depends only on the state and action at t, in which case the dynamics of an environment can be defined by specifying

$$\Pr\{s_{t+1}=s', r_{t+1}=r \mid s_t, a_t\}, \tag{1.2}$$

for all s', r, s_t, and a_t, then, it is called the state signal has the Markov property, in other words, if and only if (1.1) is equal to (1.2) for all s', r, and histories, s_t, a_t, r_t, $\cdots$, r_1, s_0, a_0. In this case, the task environment is also said to have the Markov property.

A Markov Decision Process (MDP) is defined in terms of stochastic dynamic environment with finite states and discrete time steps. The basic frame of an MDP is a tuple $M = < S, A, P, R, \beta >$, where

- S is a finite set of states of the environment, $(s_1, s_2, \cdots, s_n)$;
- A is a finite set of actions, $(a_1, a_2, \cdots, a_n)$;
- $P : S \times A \to \prod(S)$ is the state transition function, giving for each state and agent action, a probability distribution over states, i.e., $P(s, a, s')$ is the probability of ending in state s' given that the agent starts in state s and takes action a;
- $R : S \times A \to R$ is the reward function, giving the expected immediate reward gained by the agent for taking each action in each state, that is, $r(s, a)$ is the expected reward for taking an action a in a state s;
- $0 < \beta < 1$ is a discount factor.

Let $V^{\pi}(s)$ be the expected discounted future reward for starting in a state s and executing stationary policy π indefinitely, and then it is recursively defined by

$$V^{\pi} = R(s,\pi(s)) + \beta \sum_{s' \in S} P(s,\pi(s),s') V^{\pi}(s'), \tag{1.3}$$

And, given any value function V, the greedy policy with respect to that value function, π_V, is defined as

$$\pi_V(s) = \operatorname{argmax}_a \left[R(s,a) + \beta \sum_{s' \in S} P(s,a,s') V(s') \right]. \tag{1.4}$$

This policy is obtained by taking the action in each state with the best one-step value according to V.

In an MDP, given an initial state s, an agent is expected to execute the policy π that maximizes $V^{\pi}(s)$. Howard has showed that there exists a stationary policy π^* that is optimal for every starting state. The value function for this policy, written V^*, is defined by the set of equations

$$V^*(s) = \max_a \left[R(s,a) + \beta \sum_{s' \in S} P(s,a,s') V^*(s') \right], \tag{1.5}$$

and any greedy policy with respect to this value function is optimal. If a complete description of states, actions, rewards and transitions on an MDP is given, the optimal policy can be found by DP methods, for example, there are Value Iteration and Policy Iteration methods[45].

The learning agents in RL system have no knowledge about the environment in advance. The Q-learning, which is a representative RL algorithm, works by estimating the values of rules. Q-learning can be viewed as a sampled asynchronous method for estimating the optimal state action values, or Q function, for unknown MDPs. The value $Q(s, a)$ is defined to be the expected discounted sum of future rewards obtained by taking an action a from a state s and following a policy thereafter. Let $Q^*(s, a)$ be the maximum expected discounted reinforcement signal of taking an action a in a state s and continuing by choosing actions optimally. And note that $V^*(s)$ is the value of s assuming the best action is taken, then, $V^*(s) = \max_a Q^*(s, a)$. Hence, $Q^*(s, a)$ can be written recursively as

$$Q^*(s,a) = R(s,a) + \beta \sum_{s' \in S} P(s,a,s') \max_{a'} Q^*(s',a'). \tag{1.6}$$

And since $V^*(s) = \max_a Q^*(s, a)$, it has

$$\pi^*(s) = \operatorname{argmax}_a Q^*(s,a), \tag{1.7}$$

as an optimal policy.

The experience available to an RL agent on MDPs can be defined by tuples $< s, a, r, s' >$. An experience tuple is a snapshot of a single transition: the agent starts in a state s, takes an action a, receives an immediate reward r and ends up in a state s'. Then, the Q-learning rule is

$$Q^*(s,a) = (1-\alpha)Q^*(s,a) + \alpha[r + \beta \max_{a'} Q(s',a')]. \tag{1.8}$$

This creates a new estimate of $Q^*(s, a)$. If each action in each state is executed an infinite number of times on an infinite run and α, the learning factor, is decayed appropriately, the Q-value estimate will converge with probability 1 to Q^*[46]. Once these values have been learned, the optimal action from any state is the one with the largest Q-value.

1.1.3 Integrating Reinforcement Learning into Agent Architecture

The word "agent" means different thing to different group of researchers. Most of the past Machine Learning (ML) research has been focused on "disembodied" learning algorithms, i. e. , without taking into account that the learning algorithm may be embedded in an agent that is situated in an environment. Recently, the context of agent is referred to the system that is embedded in an environment, interacts with the environment, and makes decisions to change the state of the environment. An agent consists of many different interacting modules, vision, planning, etc. , and the learning module is

just one of them. The external system that an agent is "embedded" in, can perceive and act on is called environment. The agent interacts with the environment by selecting actions, and the environment presents the agent new situation responding to those actions. It is a model of an agent interacting synchronously with the agent's environment.

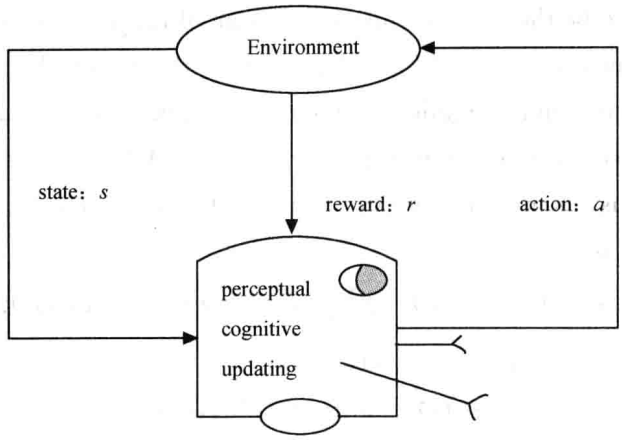


Figure 1.1 Frame of reinforcement learning.

As shown in Fig. 1.1, the agent takes the state of the environment as input and generates actions as output, which affects the state of the environment. Given this model, we can define the learning target of an agent simply as a decision procedure for choosing actions. And there are three types of data sources[47] that can be distinguished:

- External teacher: an external teacher provides examples of actions with the corresponding classification indicating their optimality or appropriateness. This model is equivalent to fully supervised learning;
- Environmental feedback: while the agent acts, it receives a feedback from the environment indicating the benefit of the actions. The feedback is usually defined in terms of the utility of the current state that the agent finds itself in. This training model corresponds to RL. It should be noted that not necessarily all states will result in feedback. This means that once some environmental feedback is received it has to be propagated to all actions that potentially contributed to it. Certainly, actions that contributed strongly should receive more recognition. A common technique to distribute rewards amongst actions is to reward more recent actions higher using a discounted factor;
- Internal agent bias: while the agent is exploring the environment with its actions, it looks out for useful patterns and interesting properties of the environment that enable the agent to generate concepts describing the environment.

Usefulness and interestingness are purely based on the agent's internal bias, and no explicit feedback is given to the agent. It is assumed that the discovered concepts will help the agent to perform future specific goals efficiently and effectively. This learning model is usually denoted by the term unsupervised learning.

As it enters a state in the environment, the agent must identify the status of the state in accordance with the environment, which is usually described as observe.

The Boltzmann distribution and ε-greedy transition policy are usually used as the action selecting policy for solving MDPs. The Boltzmann distribution is expressed as

$$P(a \mid s) = \frac{e^{\gamma Q(s,a)}}{\sum_b e^{\gamma Q(s,b)}} \tag{1.9}$$

where γ tends to infinity as an annealing process so that even a small difference between Q-value will eventually lead to the best action being selected with probability 1.0. And the greedy transition policy implements the action selecting with $\text{argmax}_b Q(s, b)$. And, the ε-greedy transition policy means an agent behaves greedily at most learning time, but with small probability ε, instead, selects a rule at random, independent of the reinforcement values.

An episode is defined as a history of experiences from the beginning of learning to a derivation of a reward or from a derivation of a reward to the following derivation of a reward. During learning episodes, an agent will derive rewards from the embedded environment and update reinforcement values on the states belonging to the episodes.

1.2 Multiagent Reinforcement Learning

1.2.1 Multiagent Systems

MultiAgent Systems (MAS) form a particular type of Distributed Artificial Intelligence (DAI) systems. Environments with multiagent are a large area of interest since communication over the Internet has become such a big part of commerce and daily life. In human society, learning is an essential component of intelligent behavior. However, each individual agent need not learn everything from scratch by its own discovery. In-

deed, they exchange information and knowledge with each other and learn from their peers or teachers. Although there are situations where an agent can operate usefully by itself, increasing interconnection and networking of computers is making such situations rare. When a task is too big for a single agent to handle, they may cooperate in order to accomplish the task. For example, ants are known to communicate about the locations of food, and to move objects collectively. At times, the number of agents may be too numerous to deal with them individually, and it is then more convenient to deal with them collectively, as a society of agents. In fact, many real-world problems such as engineering design, intelligent search, robotics, etc., require multiple agents. It can be imagined that many networking resources such as routers, gateways, or any other kind of server use MAS to improve their efficiency not only internally, but also in the communication with other resources.

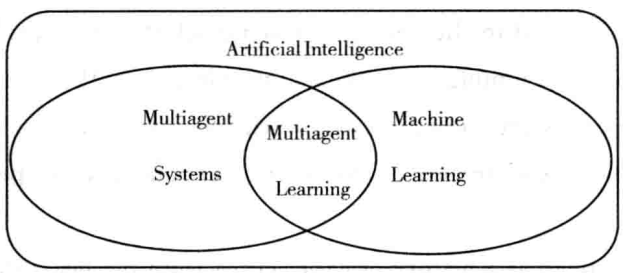


Figure 1.2 Multiagent Learning is at the intersection of MAS and ML.

Multiagent learning is the intersection of MAS and ML, two subfields of AI (see Fig. 1.2). Multiagent learning is done by several agents and becomes possible only because several agents are present. In fact, in certain circumstances, the first clause of this definition is not necessary. It is possible to engage in multiagent learning even if only one agent is actually learning. In particular, if an agent is learning to acquire skills to interact with other agents in its environment, then regardless of whether or not the other agents are learning simultaneously, the agent's learning is multiagent learning. Especially if the learned behavior enables additional multiagent behaviors, perhaps in which more than one agent does learn, the behavior is a multiagent behavior. Notice that this situation certainly satisfies the second clause of the definition: the learning would not be possible if the agents were isolated.

Traditional ML typically involves a single agent that is trying to maximize some utility function without any knowledge, or care, of whether or not there are other agents in the environment. Examples of traditional ML tasks include function approximation, classification, and problem-solving performance improvement given empirical

此为试读,需要完整PDF请访问: www.ertongbook.com