Deep Reinforcement Learning for Multi-Agent Systems:

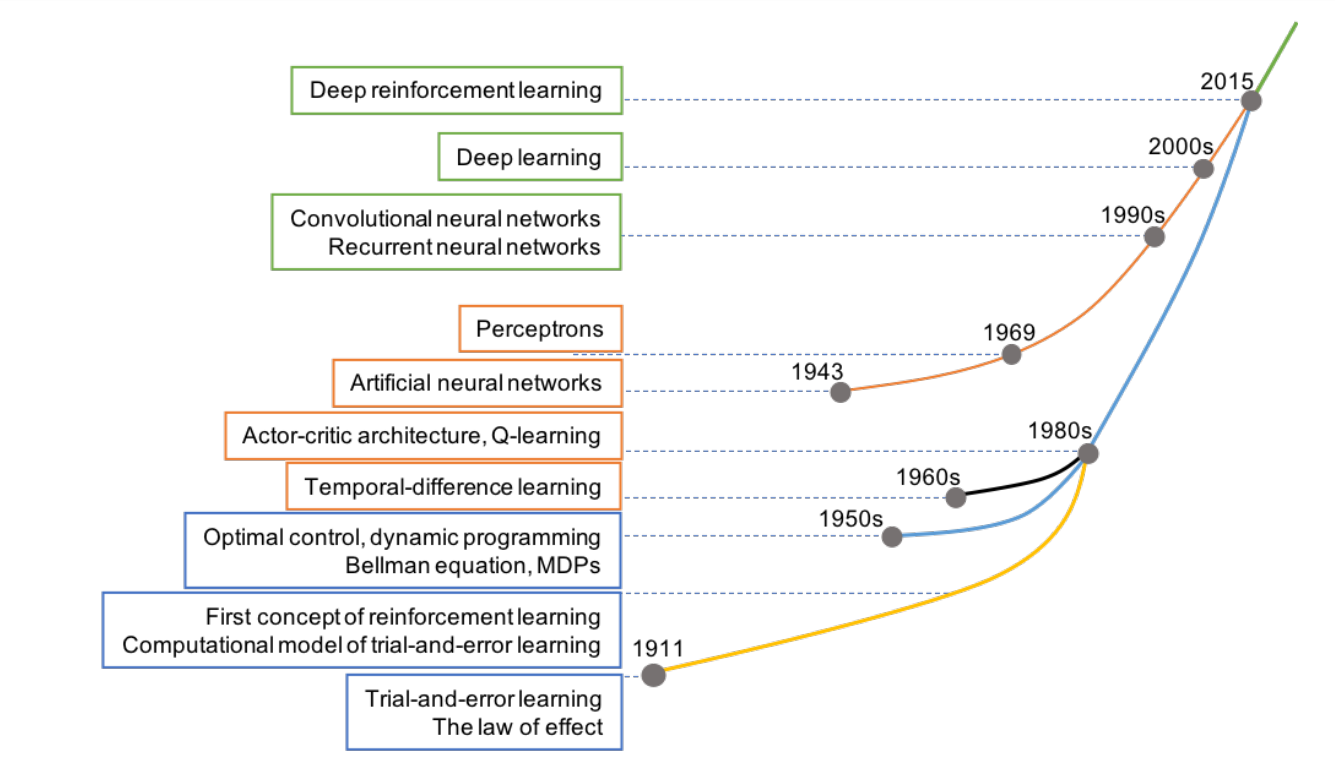
A Review of Challenges, Solutions and Applications

**Introduction：**

Multi-agent problems require multiple agents to communicate and cooperate to solve complex tasks.

Including multi-agent training schemes多智能体训练方式、multi-agent transfer learning多智能体迁移学习、trial and error (TE) procedure & the mechanism of temporal-difference (TD) learning时序差分学习.

The picture below brought the theory of optimal control including Bellman equation and Markov decision process together with temporal-difference learning to form a well-known Q-learning.



SL(supervised learning) is learning from data that define input and corresponding output (often called “labeled” data) by an external supervisor, whereas RL is learning by interacting with the unknown environment. **What are the differences between SL & RL.**

RL is not an unsupervised learning (UL) method. UL is learning to explore the hidden structure of data where output information is unknown (“unlabelled” data). In contrast, RL is a goal-directed learning, i.e., it constructs a learning model that clearly specifies output to maximize the long-term profit. **What are the differences between UL & RL.**

使用深度学习的近似器处理高维数据.

Examine the stability and adaptation aspects of agents.

Review methods for knowledge reuse autonomy in multi-agent RL (MARL).

**RL：**

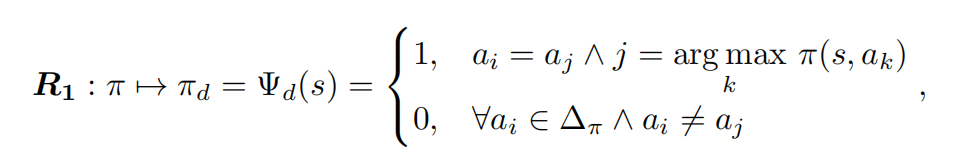
RL is a TE learning 1) by interacting directly with the environment 2) to self-teach over time and 3) eventually achieve designating goal.

The interactions between agent and environment are described via three essential elements: state s, action a, and reward r

In this case, a series of states, actions, and rewards from initial state to terminal state is called an episode.

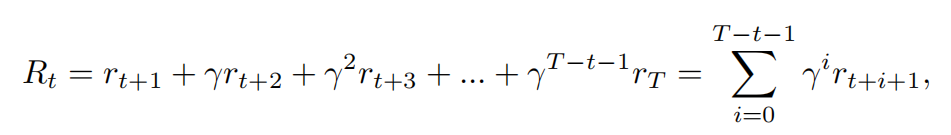
The agent’s decision by defining a concept of policy. A policy is deterministic if the probability of choosing an action a from s: p(a|s) = 1 for all state s. In contrast, the policy is stochastic, if there exists a state s so that p(a|s) < 1.

Any RL problem satisfies this “memoryless” condition is known as Markov decision process (MDP). Therefore, the dynamics (model) of an RL problem is completely specified by giving all transition probabilities p(ai|s)



In this respect, we call that policy πt+1 is better than policy πt and denoted as πt+1 > πt. Therefore, we have a series of policies improved over time as follows:

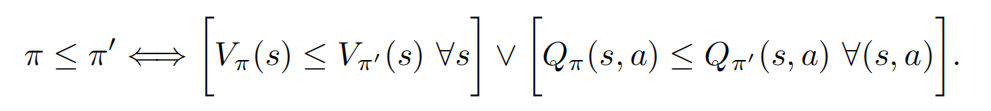
π0 < π1 < ... < πt < πt+1 < ... < π∗.

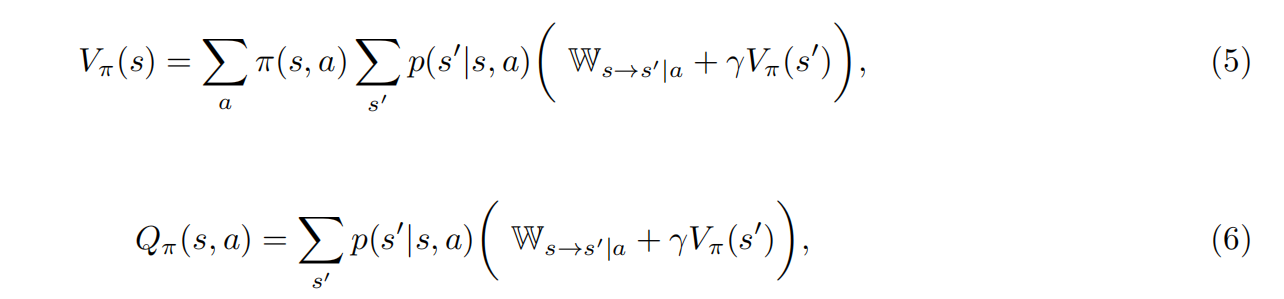


where γ is a discounted factor so that 0 ≤ γ < 1. The agent becomes far-sighted when γ approaches to 1 and vice versa the agent becomes short-sighted when γ is close to 0

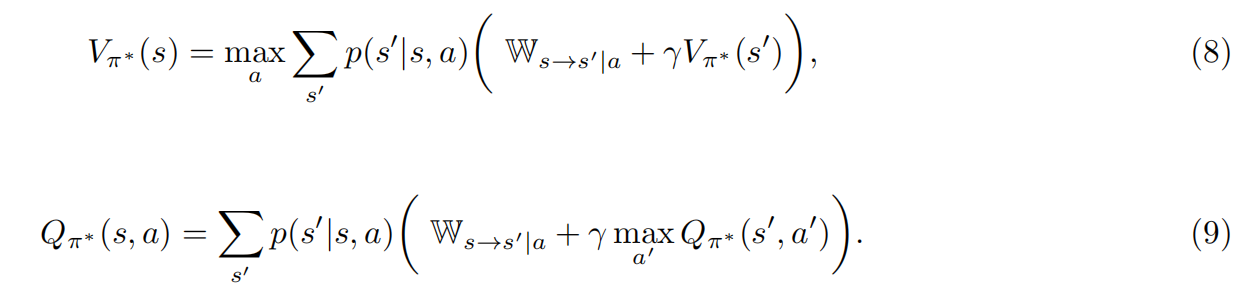
The next step is to define a value function that is used to evaluate how “good” of a certain state s or a certain state-action pair (s, a).

We can use value functions to compare how “good” between two policies π and π0 using the following rule：

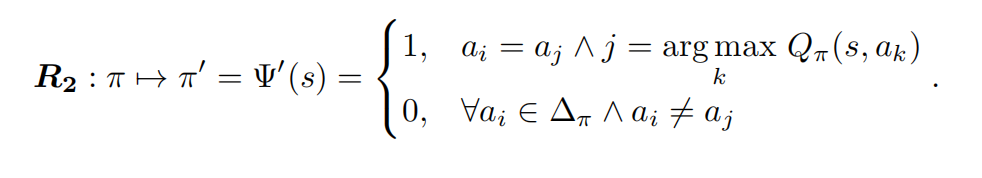


 where Ws→s0|a = E[rt+1|st = s, at = a, st+1 = s0] ≈ rt+1, Equations (5) and (6) are called Bellman equations and widely used in policy improvement.

Instead of repeating policy improvement process, we can estimate directly the value function of optimal policy π∗ using the following optimality Bellman equation：



we can derive an optimal deterministic policy π∗using：



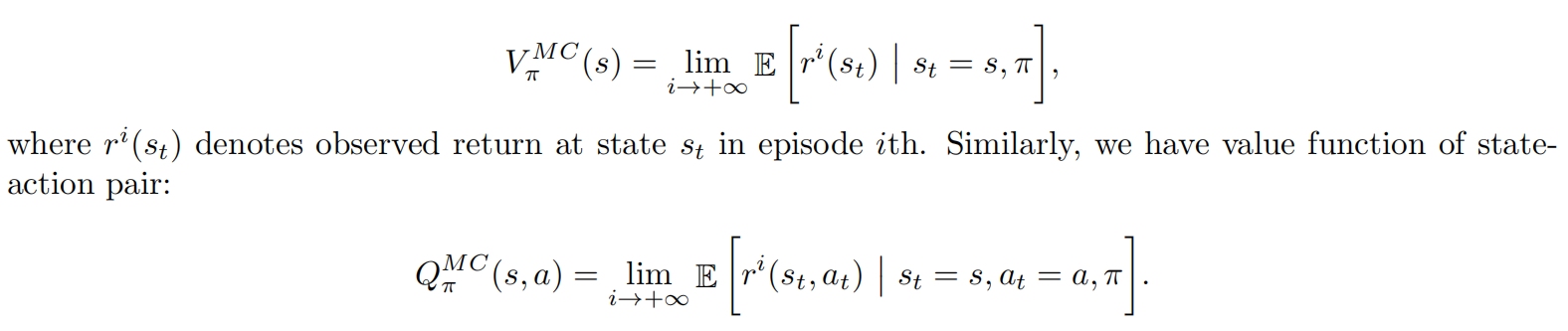
Although we can use dynamic programming to approximate the solutions of Bellman equations, DP要用完整的信息. We will review two model-free RL methods (require no knowledge of transition probabilities p(ai|s)) to approximate the value functions.

**RL Methods：**

In practice, MC and TD learning often use table memory structure (tabular method) to save value function of each state or each state-action pair.

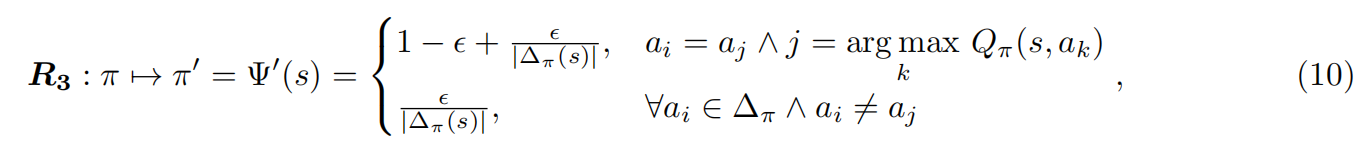
1. Monte-Carlo

Monte-Carlo (MC) method estimates value function by repeatedly generating episodes and recording average return at each state or each state-action pair.



this approach has made two essential assumptions to ensure the convergence happens:

1. the number of episodes is large
2. every state and every action must be visited with a significant number of times.



use ε-greedy algorithm for **Convergence**，

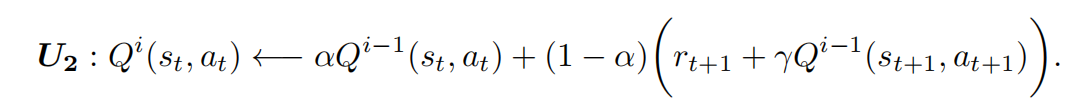
Generally, MC algorithms are divided into two groups: on-policy and off-policy. In on-policy methods, we use policy π for both evaluation and exploration purpose. Therefore, the policy π must be stochastic or soft. In contrast, off-policy uses different policy π0 = π to generate the episodes and hence π can be deterministic. off-policy随机性不太够，不太稳定但是很简单。on-policy method is more stable when working with continuous state-space problems and when using together with a function approximator (such as neural networks)

1. Temporal-Difference learning

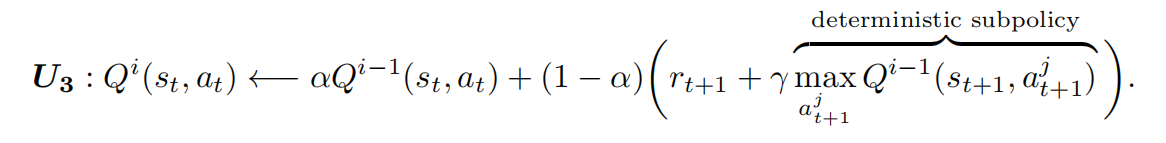
It makes an update on every step within the episode by leveraging 1-step Bellman equation.

TD learning is also divided into two categories: on-policy TD control (Sarsa) and off-policy TD control (Q-learning).

Sarsa:



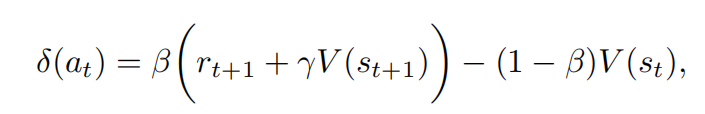
1. leaning:



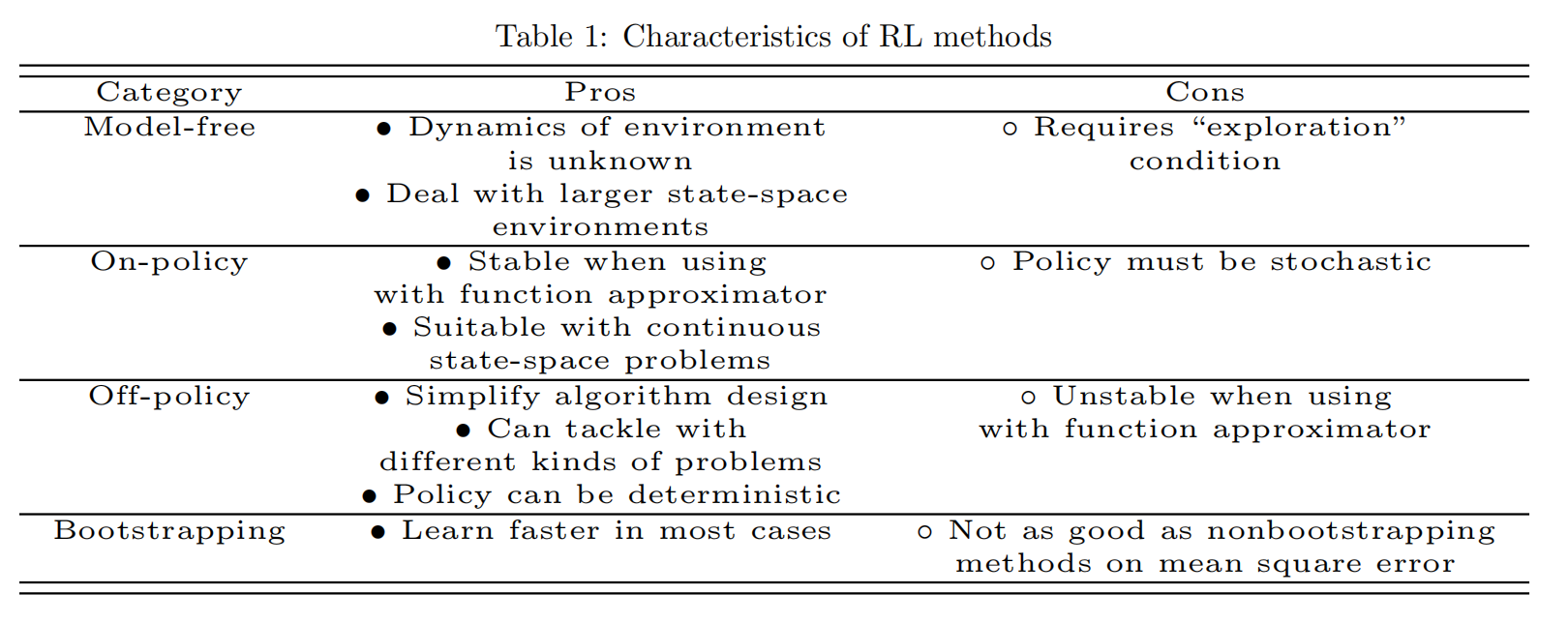
TD learning使用以前的价值估计函数Vi-1以更新Vi, 被称为bootstrapping method.

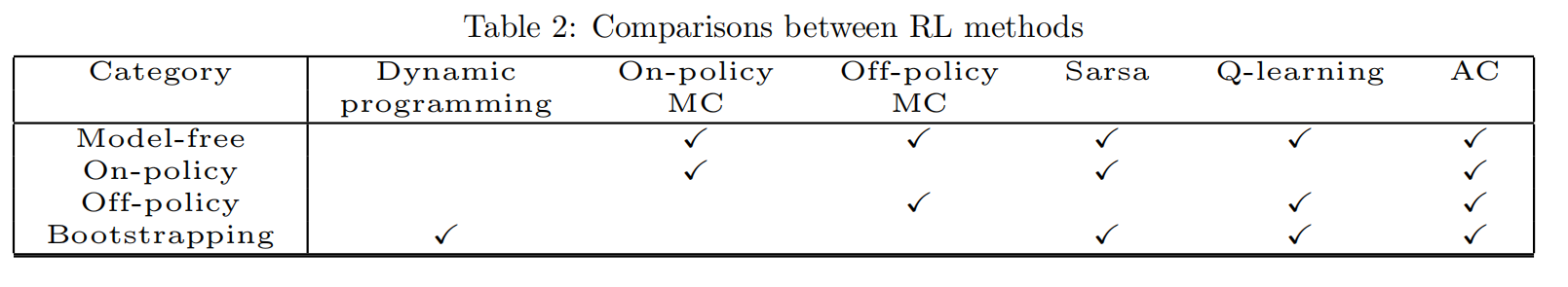
1. AC: Actor-Critic

AC can be on-policy or off-policy depending on the implementation details. Specifically, AC includes two separate memory structures for an agent: actor and critic. Actor structure产生动作扔到critic去估计判断. Critic structure uses the following TD error to decide future tendency of the selected action:





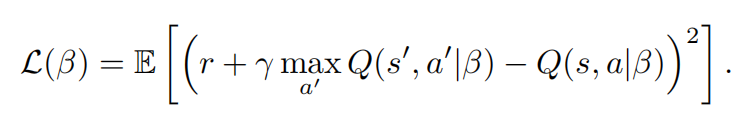




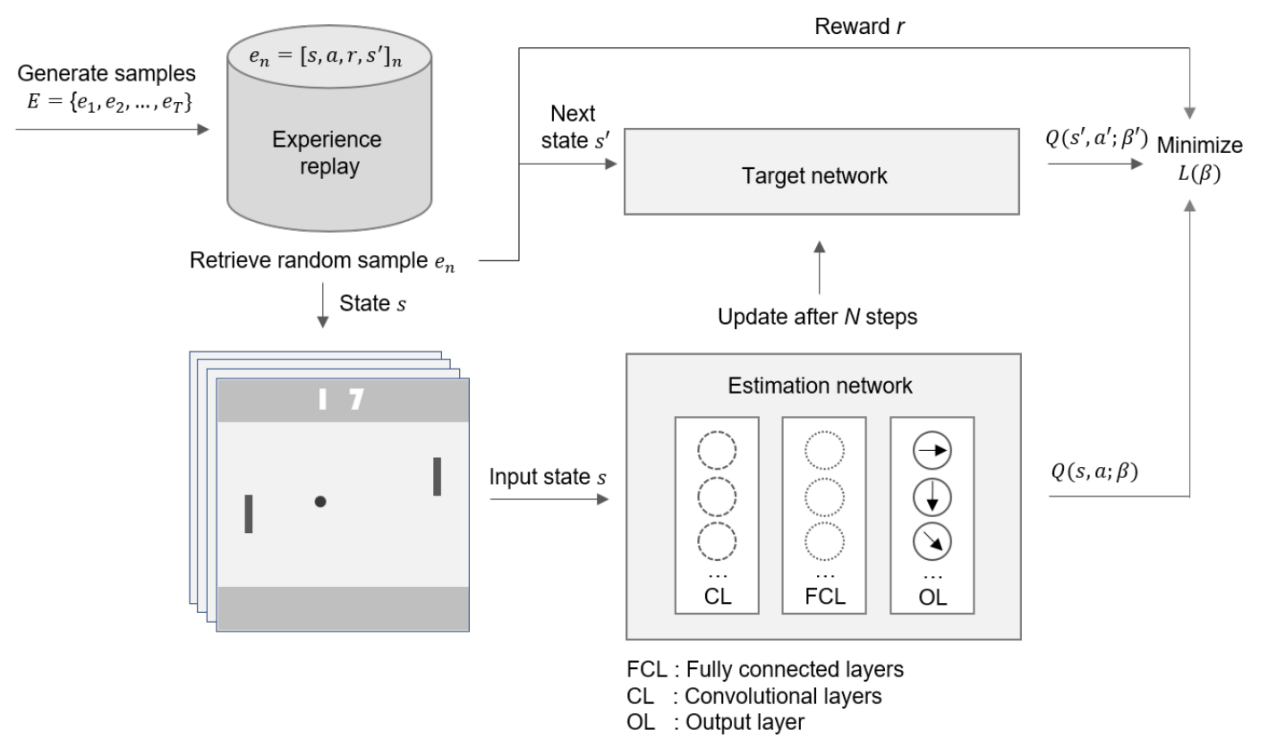
**Deep RL：For single**

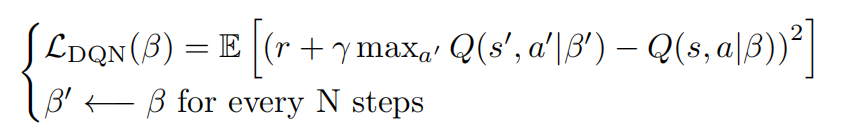
1. DQN

The output of DQN produces Q-values of all possible actions a ∈ ∆τ taken at state s,



为使样本不相关，创建一个N步更新的目标网络，同时样本从experience replay memory中间取来训练。





DQN的缺点:

1. 有些状态在训练中会发现是冗余的（即对于奖赏没什么作用，而有些policy会采用某些冗余状态，但是这些状态是很少被采用的，因此policy evaluation并不能更新到收敛，因此对于评估工作是困难的。
2. 输入不够，因此状态依赖于过去信息和部分观测情况下，DQN无法解决。

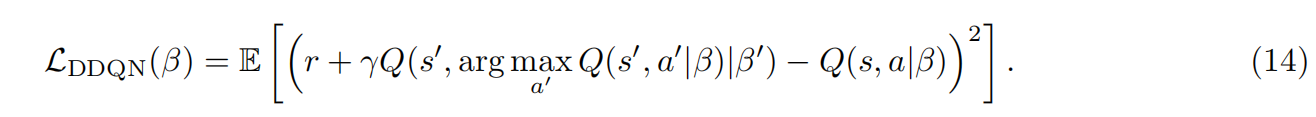
为什么DQN不使用原来的参数进行未来状态动作对估计会发散？

当DQN使用当前参数进行动作选择时，这些参数的意义在于动作选择倾向性，而使用当前的价值函数则会使得DQN打破了随机性，加大了倾向性而容易陷入局部最优，而发散。

1. **DQN的变种**

DDQN:

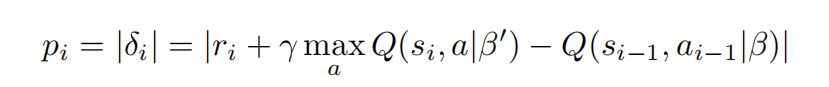
DDQN的想法在于将贪心的选择动作和动作评估分开，以消除对于动作价值函数过度估计的问题。



DDQN with PER(Prioritized experience replay)优先级经验回放: **为解决DQN的缺点1**

特别的，我们更希望经常使用目标相关的样本进行探索策略。

prioritized experience replay that gives priority to a sample i based on its absolute value of TD error:



DRQN: **为解决DQN的缺点2**

直观解决方法是使用LSTM结构代替连接最后一个卷积层的全连接层。这种DQN变种叫做deep recurrent Q-network (DRQN)

DRQN升级版：Deep attention recurrent Q-network (DARQN)

**Deep RL：For Multi-Agent: MADRL**

在多智能体学习领域下，每个智能体的价值函数也受共有动作、共有策略影响。

1. **Challenges & Solutions**

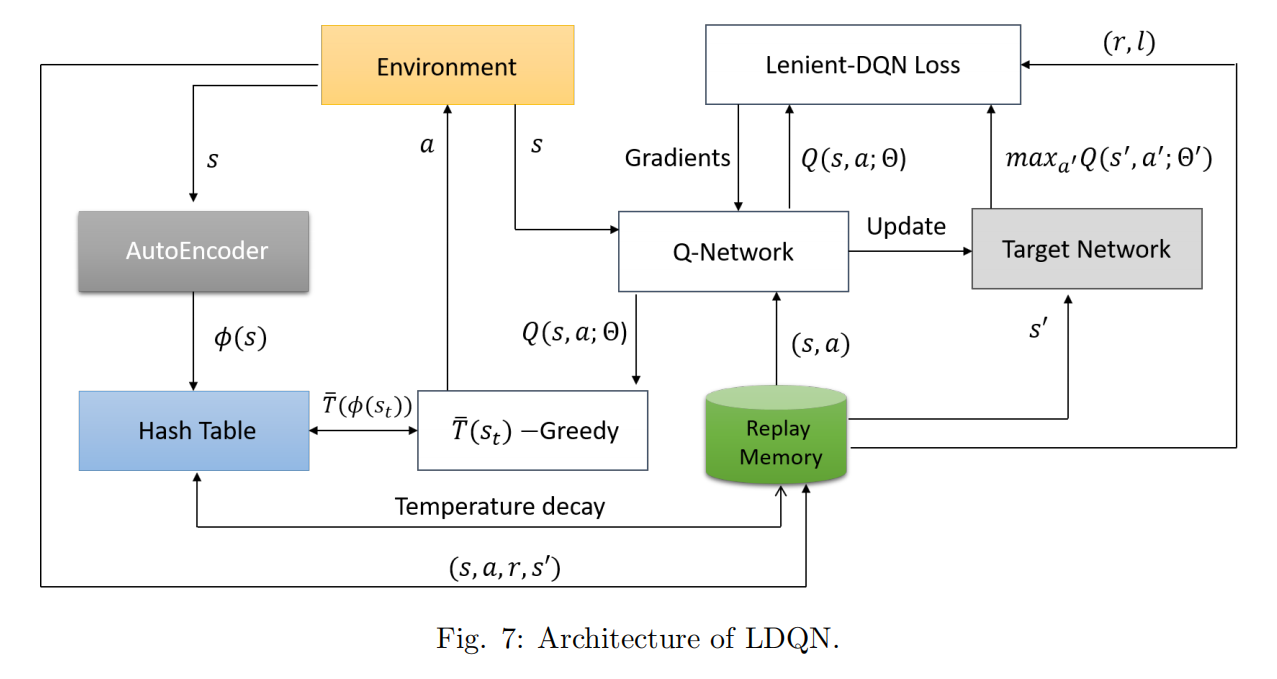
**Non-stability不稳定性**

智能体互动和一起学习以重构环境导致了不稳定性。每一个智能体的单独策略影响整体最优策略，因此，无法单纯学习单个智能体的最优策略。Q-learning在多智能体环境下无法保证收敛性，因为在不稳定环境中，马尔可夫性质无法保证。

在多智能体设定下，有针对探索和利用困境的DQN算法变种。学习何时独立决策何时选择合作，同时可以在不稳定环境中收敛。proposed two variants of DQN, namely deep repeated update Q-network (DRUQN) and deep loosely coupled Q-network (DLCQN), to deal with the non-stability problem in MAS(multi-agent system). **DRUQN??DLCQN??**

处理协作的多智能体物体运输的问题LDQN: is compared with the hysteretic-DQN (HDQN)

WDDQN:Weighted double deep Q-network (WDDQN) in [120] to deal with non-stability in MAS.



**Partially-observed局部可观测性**

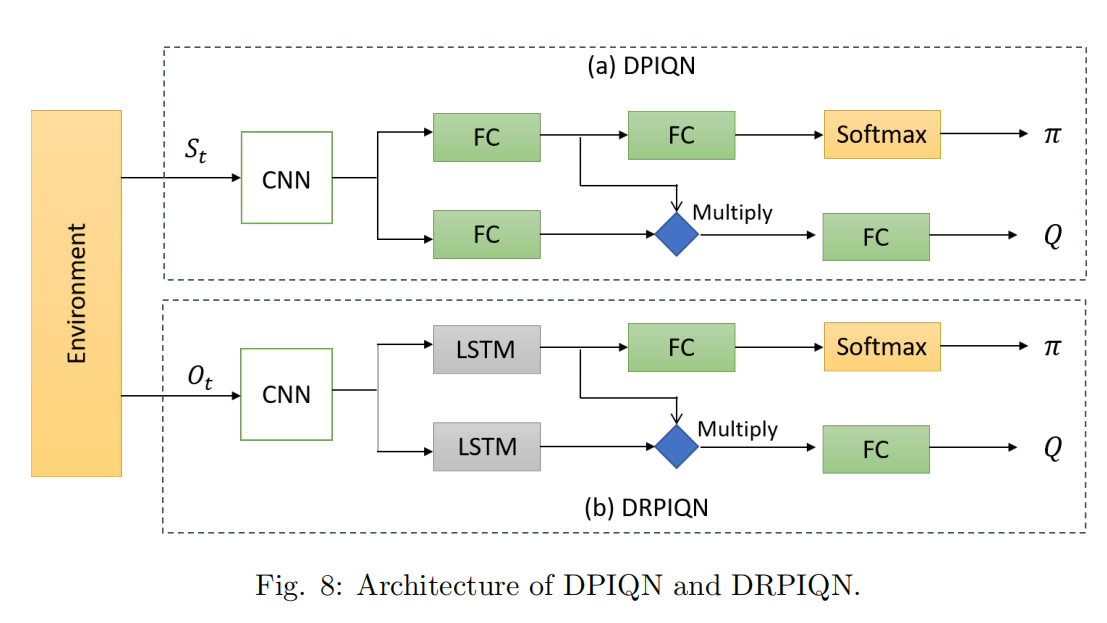
存在有智能体只观测到部分环境信息的情况下，在每一步做最好的决策问题，问题可以被抽象成POMDP问题。the partially observable Markov decision process (POMDP)

推荐的DRQN是基于LSTM的循环网络。在循环网络结构下，智能体可以更健全地在部分观察环境下学习改善后的策略。

由DRQN扩展的DDRQN是为了处理POMDP问题。有三个特点构成，上个动作输入，（LSTM循环网络特点）；内部智能体权重分享，所有智能体共用一个网络结构，残缺经验回放？？简单排除了DQN经验回放的特点。

CL方法集成了3中DRL方法，包括策略梯度，时序差分，和AC，原则是学习简单的任务进行知识经验积累，以处理复杂问题。

深度策略推理QN，是深度循环策略推理QN的升级版以处理部分可观测环境。a deep policy inference Q-network (DPIQN) to model multi-agent systems and its enhanced version deep recurrent policy inference Q-network (DRPIQN) to cope with partial observation.



还需要处理与真实状态无关的噪声观测。

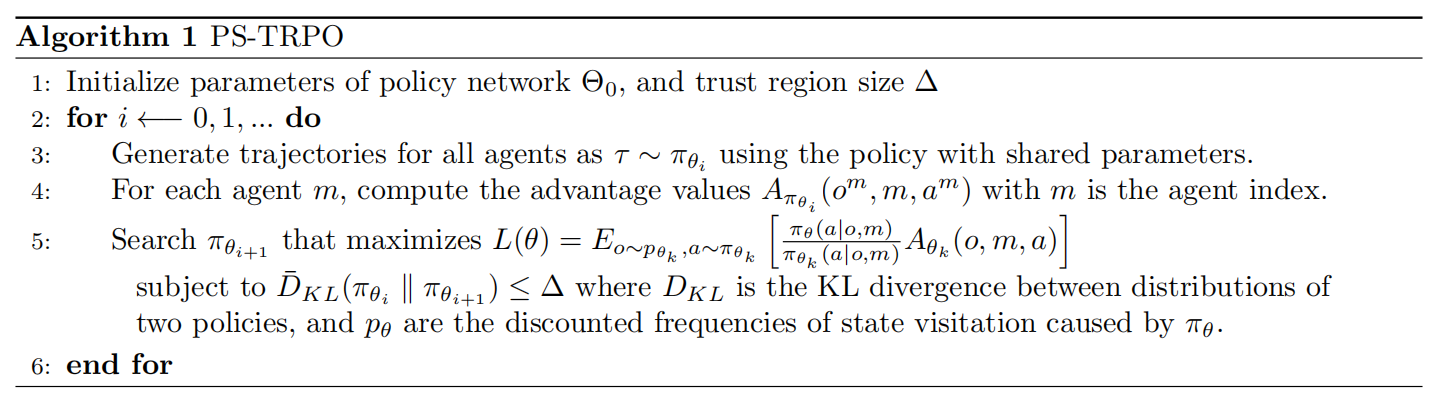
**MAS-training多智能体系统训练问题**

另一种流行的方法是通过信道集中学习和分散执行策略，即独立基于局部观测进行动作选择，适用于部分可观测和通信受限。

集中学习每个智能体的策略已成为多智能体设定的标准范例，因为通信有时可能会受限，同时获得额外的状态信息。

集中策略使用共同的奖赏信号进行学习以便从共有的观测中获得共有的动作。而分散执行策略独立基于私人观测，但同时使用参数分享方式允许智能体之间充分使用其他智能体的经验。

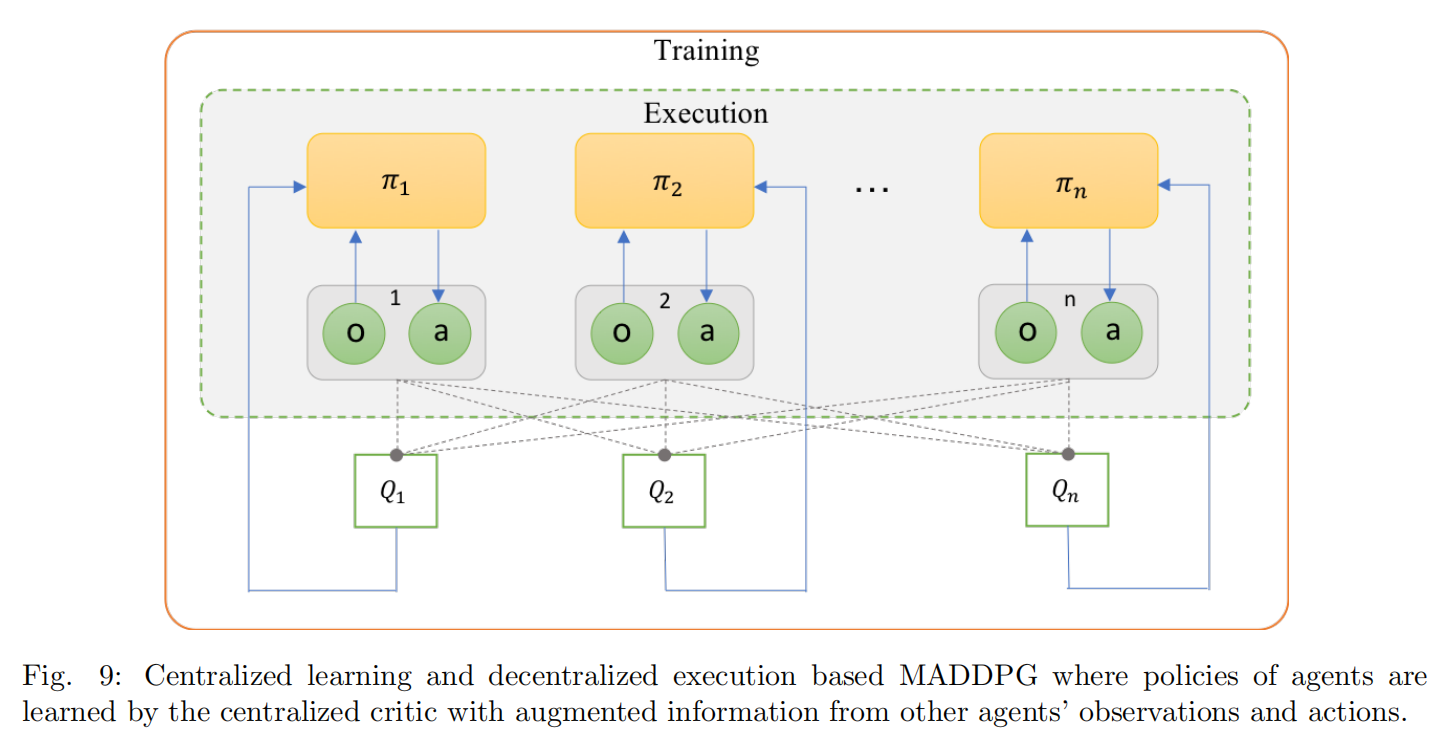
参数分享的TRPO对于高维部分可观测数据和连续动作空间有很好的效果，以下是算法：



DIAL方法通过信道将梯度从一个智能体传输给其他智能体，允许在智能体之间端到端后向传播。Developed communication neural net (CommNet) 允许智能体在完全协作任务中与他们的策略持续互相通信。

将分散执行和集中学习应用于分级主从结构中形成MS-MARL以解决多智能体系统的通信问题。

MADDPG 使用DDPG和集中学习、分散执行的技巧，critic使用额外的信息来简化培训过程同时actor根据局部观测进行动作选择。Fig. 9 illustrates the multi-agent decentralized actor and centralized critic components of MADDPG where only actors are used during the execution phase.



**Continuous Action Space连续动作空间问题**

trust region policy optimization (TRPO) 可以扩展到连续动作空间，以优化robotic locomotion and image-based game playing.

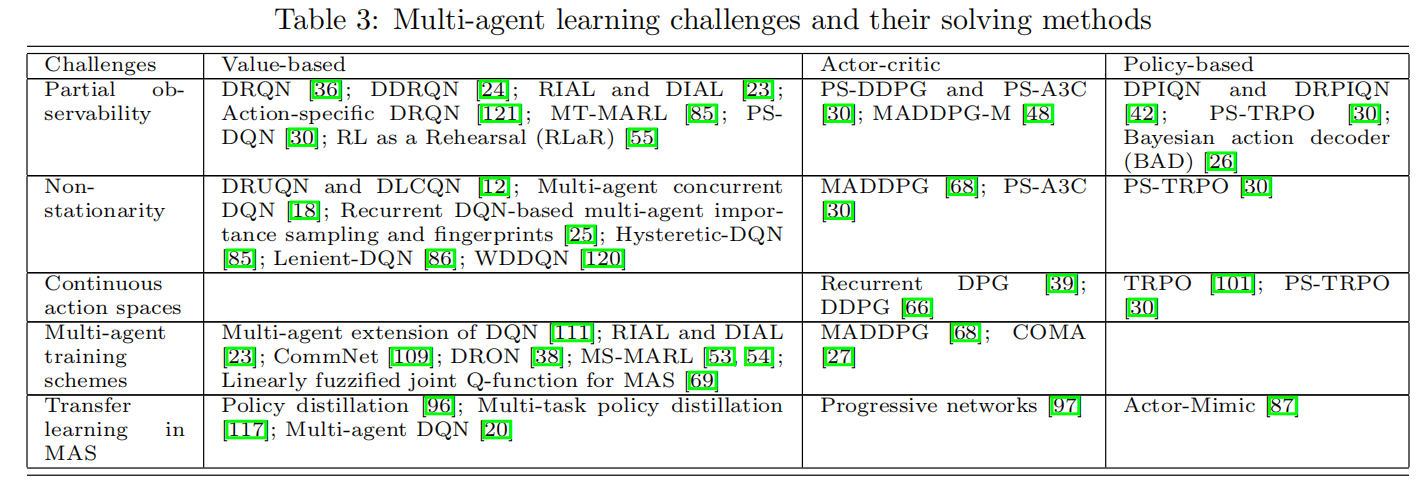
Off-policy: DDPG使用AC结构处理连续动作空间问题。

Extended DDPG to recurrent DPG (RDPG)以解决部分可观测且为连续动作空间的问题。

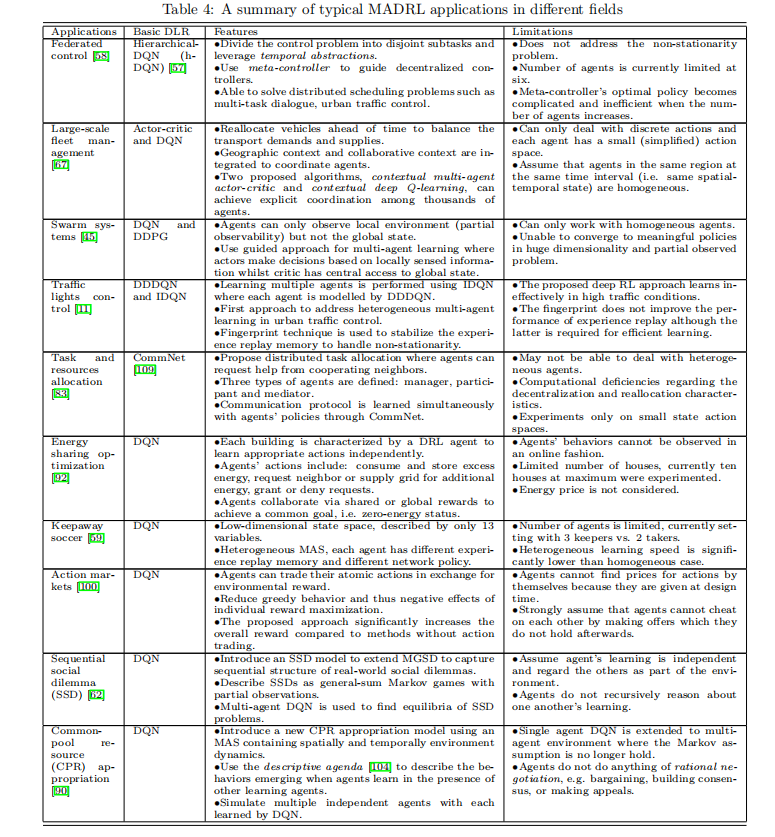
**Transfer-learning for MADRL多智能体深度强化学习的迁移学习问题**

训练成本很高，而深度强化学习的知识迁移又是很重要的。

the actor-mimic method for multi-task and transfer learning 改善了deep-policy-network的训练速度。



1. Applications



**Conclusion：**

文章讲述多智能体问题的挑战和目前在深度强化学习领域的解决办法。

一方面，模仿学习尝试使用监督学习方法将状态映射到动作空间中。

逆强化学习需要从专家经验中推断出环境的一个奖赏函数。

These methods however have not yet been explored fully in multi-agent environments.

Model-free deep RL已经在多智能体和单智能体领域解决了许许多多复杂的问题。

Many applications of MARL can now be solved effectively by MADRL based on its high-dimension handling capability. Therefore, there is a need of further empirical research to apply MADRL methods to effectively solve complex real-world problems such as the aforementioned applications.