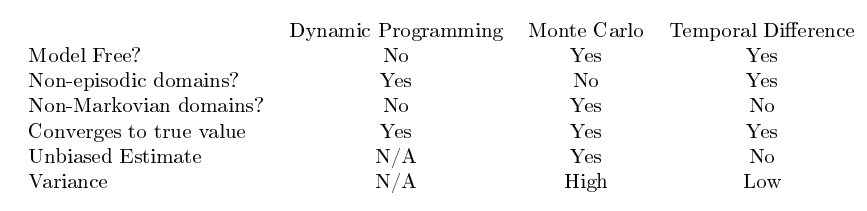
强化学习

问题：

* 策略迭代，on policy，off policy等问题理顺
* RL在机器人的应用？每到一个目的地（目标点奖励设最大，碰障碍物之类惩罚设大）都要训练，似乎有点不合理？
* Notes7、8、9、11、14较难，粗略看
* Notes2中Bellman backup
* 一次打开所有node，查看所有问好并记录，然后按点总结，最后以表格或图的形式总结
* rewards can be stochastic or deterministic。A policy can either be deterministic or stochastic. the policy may be varying with time, which is especially true in the case of finite horizon MDPs是与model-based和model-free有关吗
* We will call policies that do not vary with time stationary policies。对于stationary，不随t改变，某个状态采取动作为π(a|s)，而non-stationary为πt(a|s)
* dynamic programming and Monte Carlo区别：前者只会保存一个备份值函数，每次采样动态更新；而后者常通过多条历史episode来求平均值计算值函数，数据越多估计越准确。
* dynamic programming and TD区别：前者是从最后一个状态开始算？而后者有个bootstrap（自举）过程，就利用已知下一个状态值函数来计算当前状态期望值函数
* Monte Carlo利用整个episode，能一次捕获所有信息，而TD（0）需要一个状态一个状态回传，有些延迟
* Policy evaluation是通过求值函数来评估策略，control是通过探索和利用采取动作。前者主要是状态值函数，后者是动作-状态值函数（Q）

1. 自己总结

* 当机器人在房间行走时，目标点的距离可以大致确定，则需要机器人移动的步数也会有个上限值，然后就可通过训练让机器人在仿真环境反复训练，直到找到最优路径。
* 典型代表：DQN(Deep Q-Networks)、GO、GAN(Generative Adversarial Networks)
* 强化学习是探索和利用的平衡：随机探索，探索新的状态和奖励，按照已有的经验利用获得的知识。处理好短期和长期的利益。
* 设我们的世界可能处于的一组可能状态，设s是一系列观察到的状态，按时间t索引，设a是一组可能的动作。S就是所说的环境。
* 假设Markov Property：

奖励函数（reward function）：rt一般是在st+1观察到的奖励



注意R（s）只是指当前状态下的回报，叫做立即回报

对MDP，π(a|s)有和状态转移矩阵相似

一个强化学习**模型**由状态转移矩阵（概率）和奖励函数组成 。若已有模型，则叫model-based agent，否则叫model-free agent，需要自己去探索状态（环境），奖励（在某个状态发生碰撞之类）

在现实探索中观察（observation）o一般不等于真实所有状态s，这样叫partially observable，o称为belief state。如o=s，则叫fully observable

* 虽然奖励代表的是si+1位置的收益，但是回报为ri
* r0对应刚开始s4处的奖励



* 值函数或action-value函数求得后，那么我就可以每步选择最大的函数值出，把相应转换概率变大，因为值函数是连续的，一般贪婪方法求得最大就是全局最大
* lookup-查找，如lookup table；entry-查询的入口，backup-备份
* 1

1. Monte Carlo（蒙特卡尔）：通过实验的概率来求事件发生概率或求期望值

<https://blog.csdn.net/wuguangbin1230/article/details/80721191>

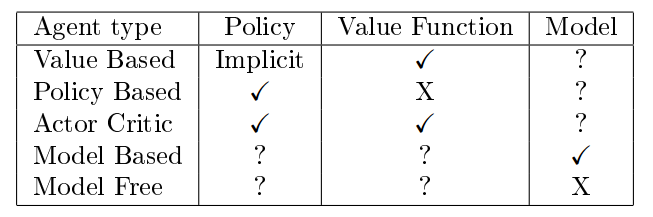
1. Bootstrap（自举法）：就是通过局部分布来估计整体分布，放鱼实验，不像正态分布之类有参数估计，它是非参数估计

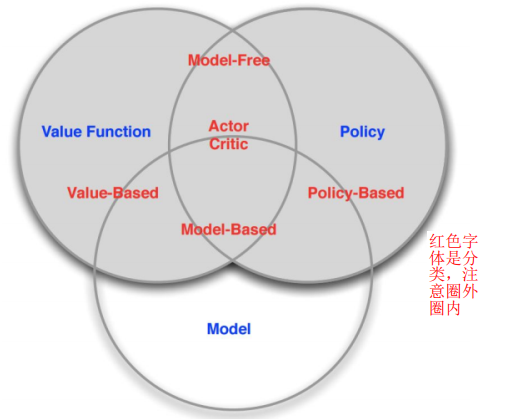
<https://baike.baidu.com/item/Bootstrap/8301573>

1. 模仿学习（未看完）

<https://blog.csdn.net/c2a2o2/article/details/77336551>

1. 强化学习介于监督学习和无监督学习之间，通过与环境的相互作用获得奖励来作决策
2. 强化学习分类

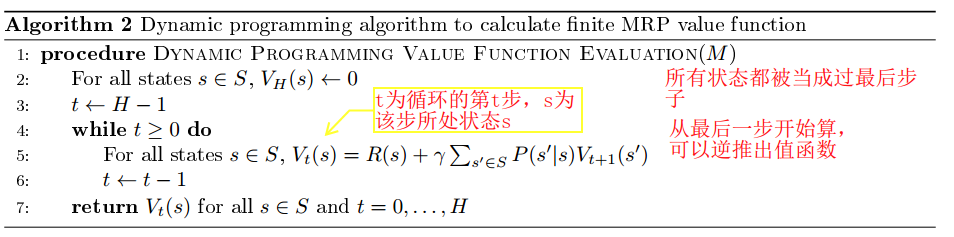


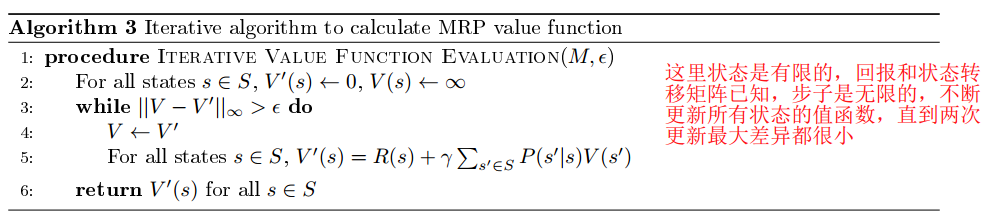


1. model、policy、value function在notes开头有详细描述。策略不是一系列动作，是指整个环境下的状态转移分布？可以是固定的也可是随机的（探索用）？值函数是固定策略下的期望回报Gt

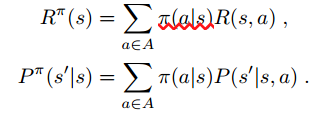
解：在值迭代算法中，见第4点，策略是指一系列动作

1. （notes2）Markov process (MP)、Markov reward process (MRP)、Markov decision process (MDP)简介，三者是建立在Markov Property基础上的，一般假设S、A都是有限的。

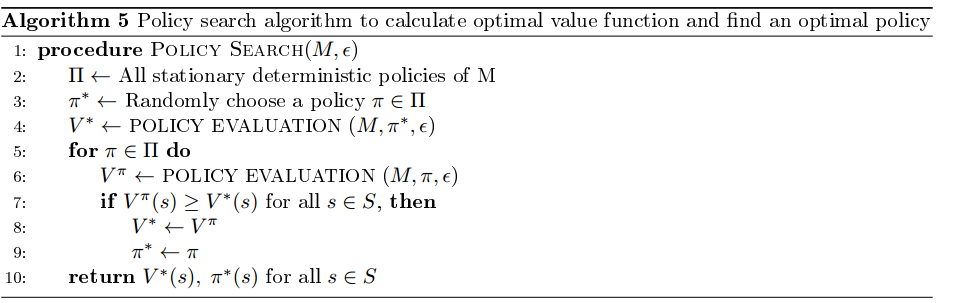
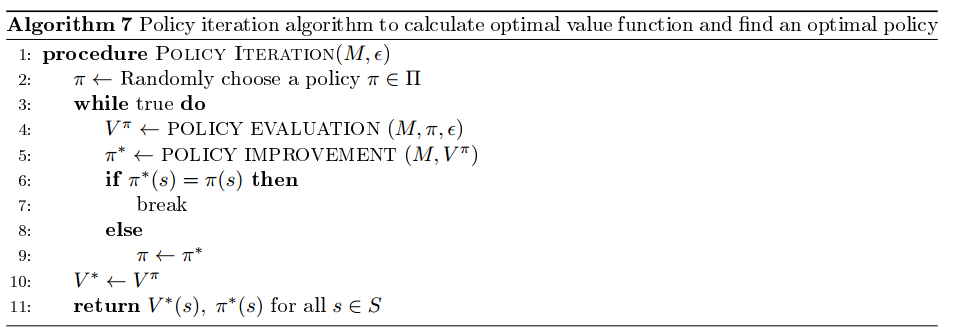
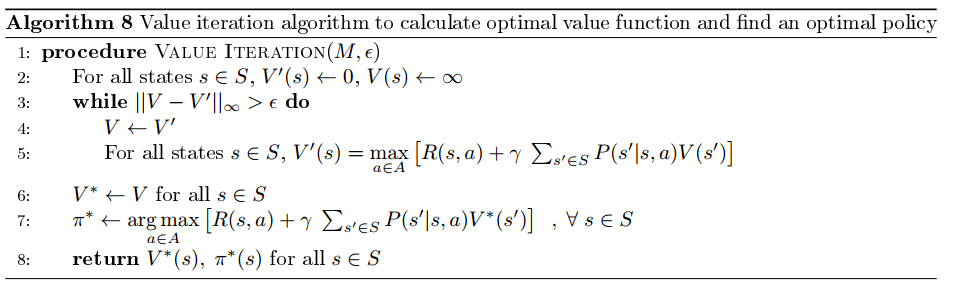
* 三者需新增两个假设：Finite state space、Stationary transition probabilities如下式，i、j是时间先后探索的状态，而转移概率和探索先后无关，只与当前状态有关。转移矩阵如下，为非负，行和为1
* time independent（两个状态之间的变化不随episode的时间改变）：Stationary transition probabilities、Stationary rewards
* MRP包含(S; P; R; γ)，MRP没有动作，任意两个状态之间可以相互转换，可以等同于动作，但是MDP有动作，不同动作对应着不同转换。MRP中计算值函数方法（model已知）：
* Monte Carlo simulation：采样N个episode求平均回报
* Analytic solution：适用于有限时间步episode，不是有限状态，注意P是矩阵，包含所有转换
* Iterative solution：
* 有限episode：dynamic programming，
* 无限episode：停止条件证明见notes2



* 在MDP中有，意思是在s采取动作a后到s’的概率不一定是1？对的，见第10条

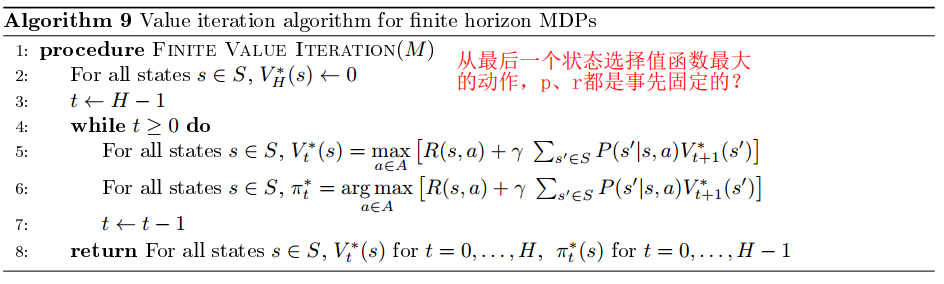


对于无线episode的几种决策方法

* Policy search
* Policy iteration：比policy search有效，有点贪婪算法的意思，有点类似EM算法，先算值函数，再算策略，依次迭代
* value iteration

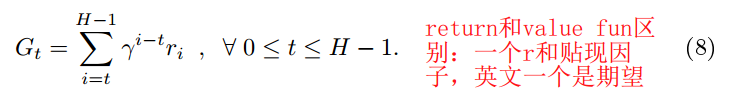
对于有限episode方法，value iteration和Policy iteration区别，前者根据每步根据最优值函数选择动作，有点贪婪的意思，后者先搜索完整策略，再选择最优策略

* value iteration：每步选择最好动作，从后往前选



* 1
* 1

1. Horizon：一个episode的步数。一个H的return如下，t是在一个episode中的时间步，st是表示一个episode所处状态，s为真实环境状态，注意区别回报和值函数两者，值函数相当于回报加了期望：

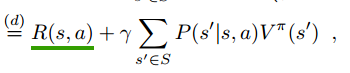
state value函数，需乘上概率分布



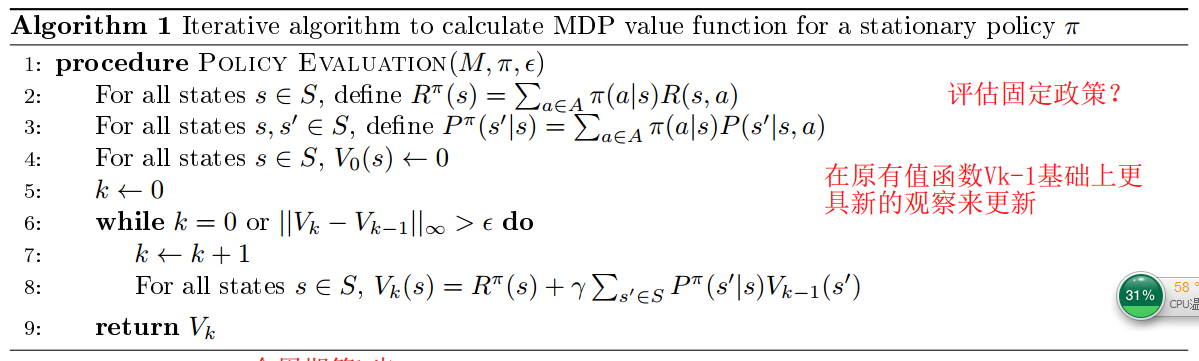
或

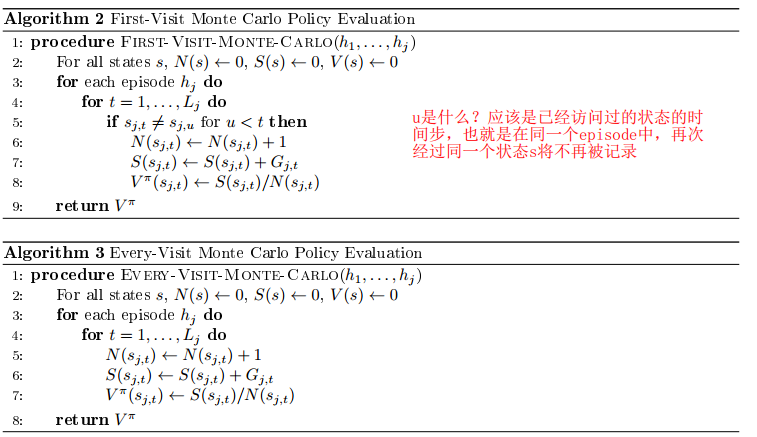


State-action value function：在某状态采取具体动作后值函数，确定采取某个动作后，第一步回报就是确定值了，不用求期望了



1. notes3- Model Free Policy Evaluation：we will assume an innite horizon as well as stationary rewards, transition probabilities and policies

* history：一个episode，每个episode有终止条件
* 提到的几种方法
* Dynamic Programming：
* Monte Carlo On Policy Evaluation：One way to estimate the expected commute time is to simulate our commute many times on the simulator and then take an average（使用开始于s的多个采样episode的平均回报作为期望回报，奖励是自己观察到的） over the simulated commute times. This is called a Monte Carlo estimate of our commute time.

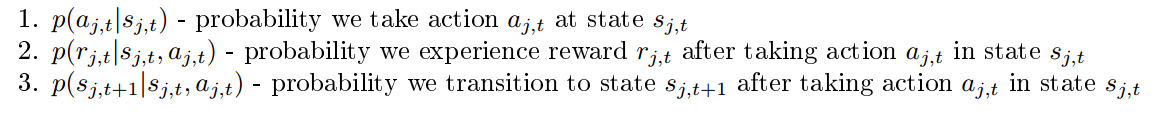
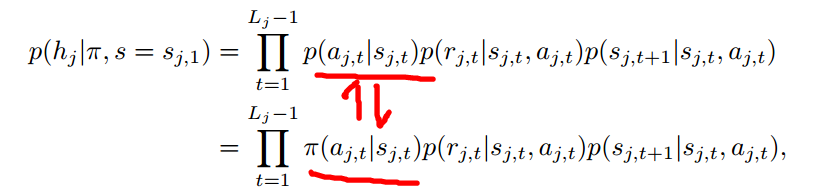
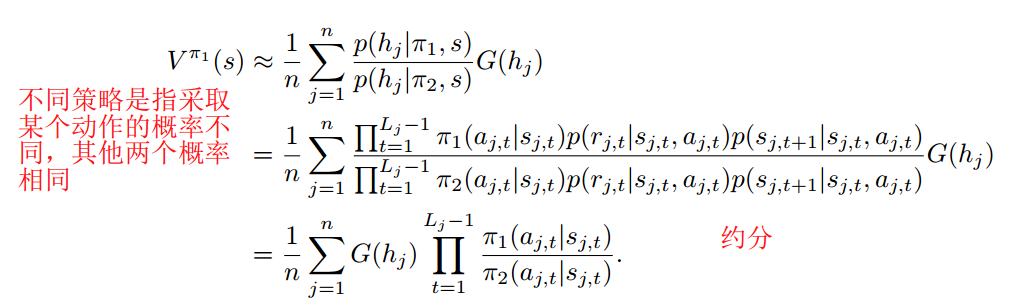
First-Visit Monte Carlo and Every-Visit Monte Carlo如下：

其中可简化如下，还可Replacing 1/N(s1jt) with α



* Monte Carlo Off Policy Evaluation: in many costly or high stakes situations, we aren't

able to obtain rollouts of Gt under the policy that we wish to evaluate. We need to use data taken from one policy to evaluate a different policy.importance sample的公式推导如下，注意各个概率的表述，之所以叫MC因为它最后也是去平均值：



* Temporal Difference (TD) Learning：可用TD(λ)来表示Monte Carlo and dynamic programming的混合。λ=0表示当前算法，λ=1表示MC，中间值表示两者混合

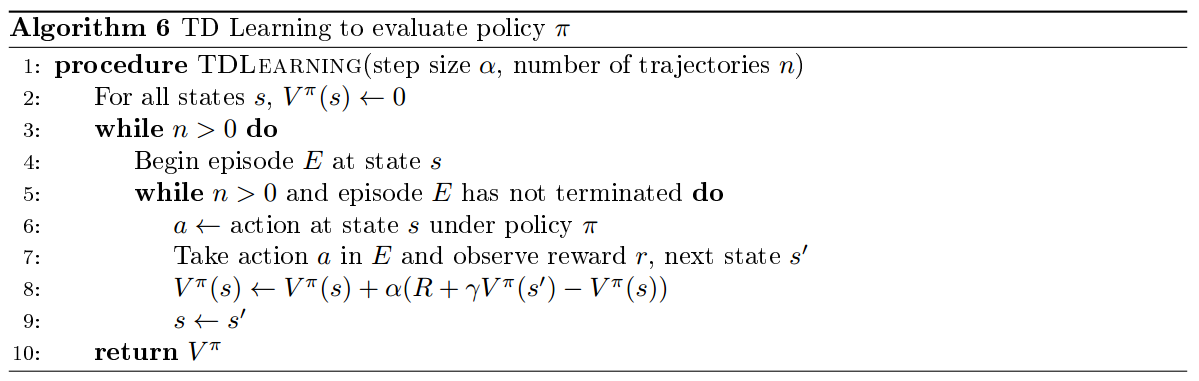
TD error：

TD target：

Algorithm：实践中，通过agent不断在环境中探索和利用通过观察获得R，来更新值函数，值函数更新好了，求状态转移矩阵就简单了。在这过程中agent得能自己观察到啥时停止。

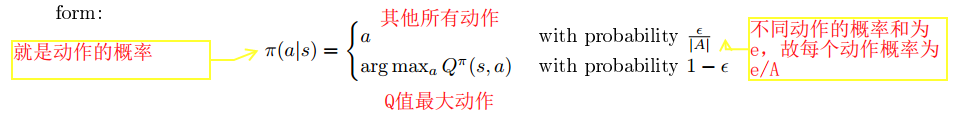
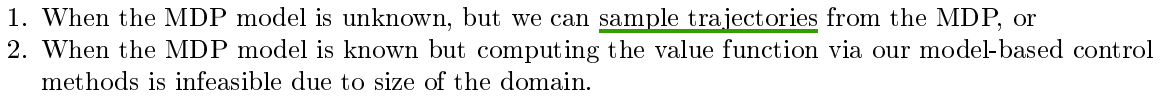
In both dynamic programming and temporal dierence learning, we only backup over one transition (we only look one step ahead in the future), so termination of histories is not a concern, and we can apply these algorithms to non-episodic domains.

意思是转移矩阵是不断更新的，只有一份备份，那么它们可以一直跑更新，不终止也行，最后得到尽可能多的状态的值函数

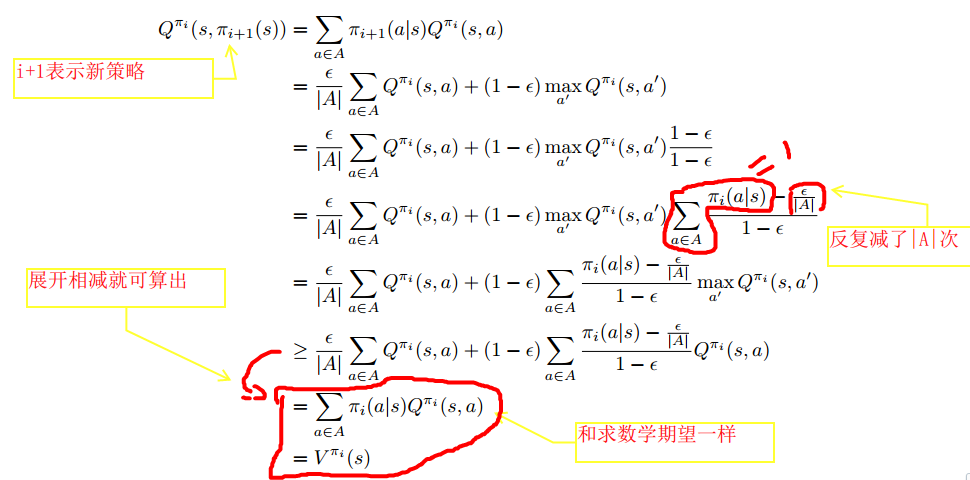


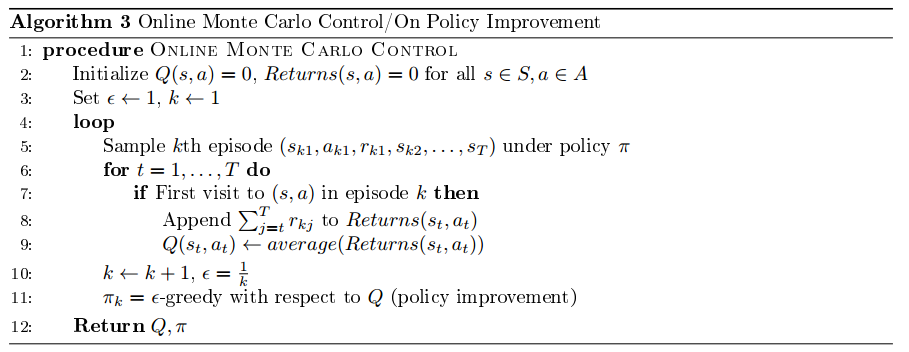
* 1
* 1

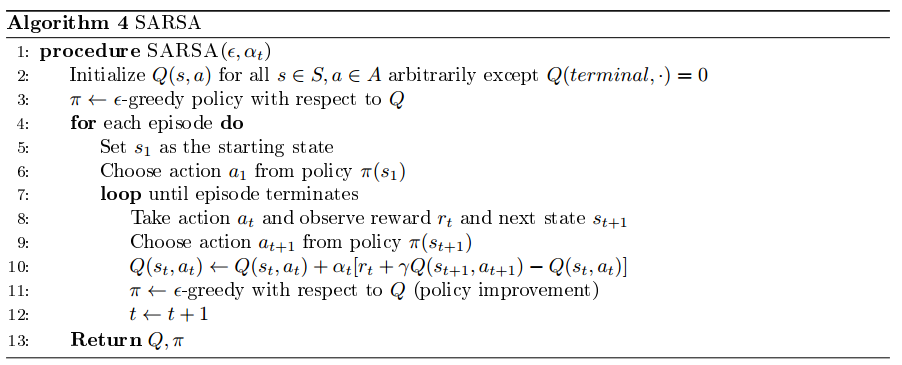
1. notes4- model-free control：只能通过与环境的相互作用获得奖励函数，其他的未知。主要适用以下两种情况：

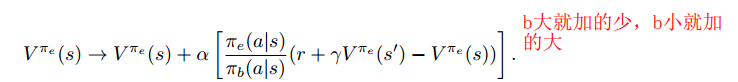
* 探索和利用，-greedy方法。如Greedy in the Limit of Infinite Exploration (GLIE)

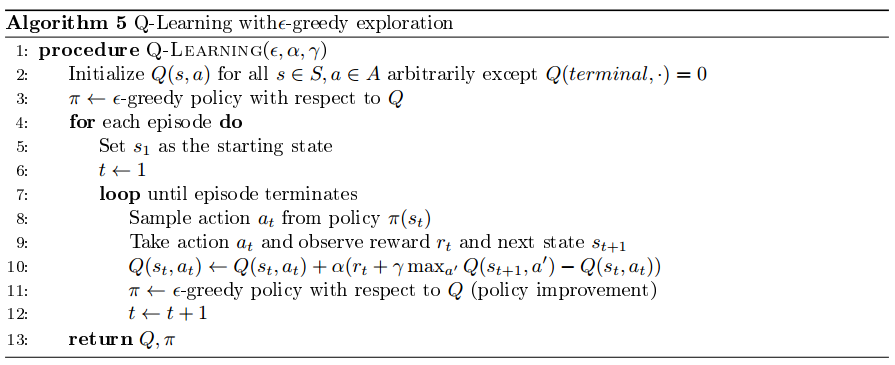
证明该方法两个策略迭代之间有提升

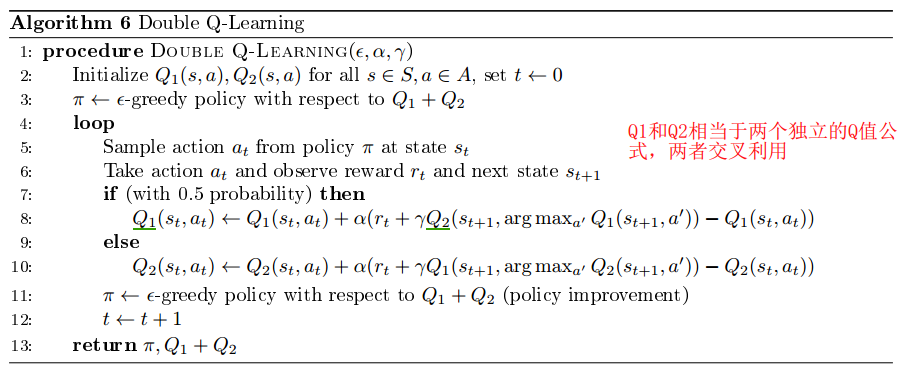


* GLIE用于MC control，注意line 11
* Temporal Difference Methods for Control：加上了control变成SARSA，这是on policy方法



* Off-policy方法公式

* Q-learning：this action is not necessarily the same as the one we would derive from the current policy. Therefore, Q-learning is considered an o-policy algorithm。注意和SARSA对比，line 9区别
* double Q-learning：



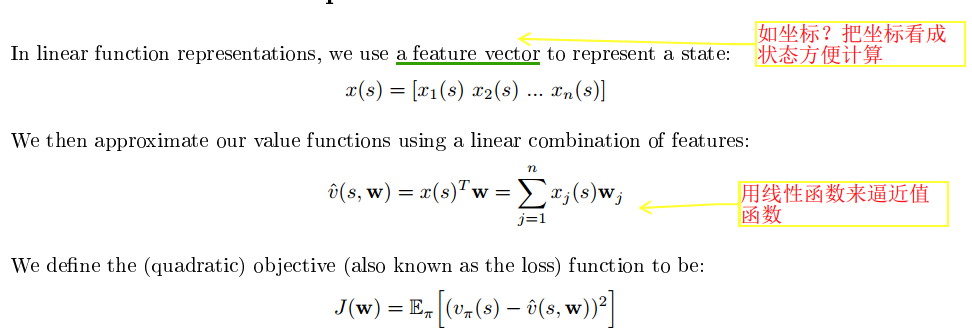
* 1

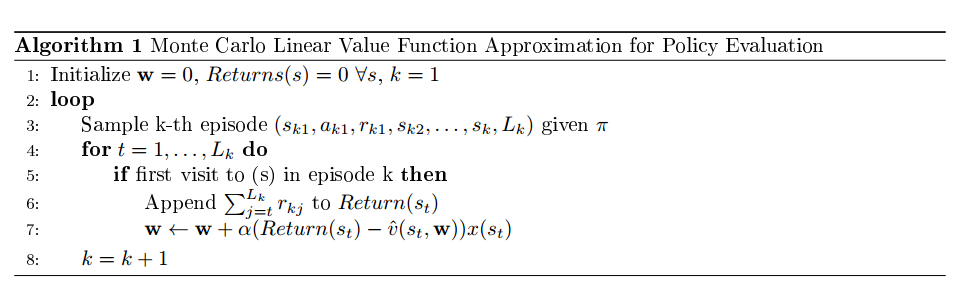
1. notes5- Value Function Approximation（VFA ）

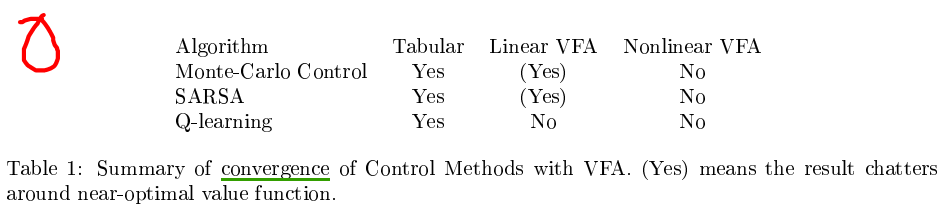
However, this approach might not generalize well to problems with very large state and/or action spaces（占内存，探索时间长，查找慢）, or in other cases we might prefer quickly learning approximations over converged values of each state.

所以可以用下面公式来近似值函数：

常用几种方法：

* Linear combinations of features：可以考虑先用传统方法估算出一些值函数，注意目标函数是期望

如下是具体算法，line 7第二项是方差求导，把回报看成值函数的无偏估计，在其他算法的运用也近似,这是否是会产生不稳定问题？在note中有证明收敛性，如下图

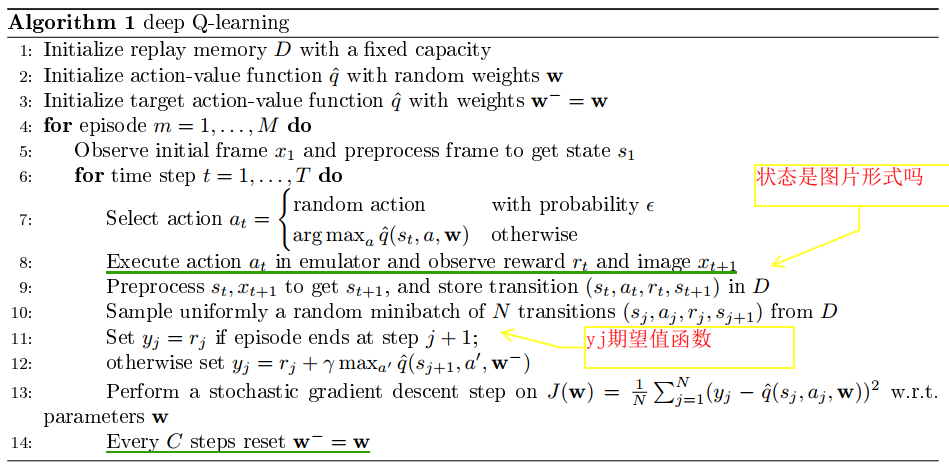


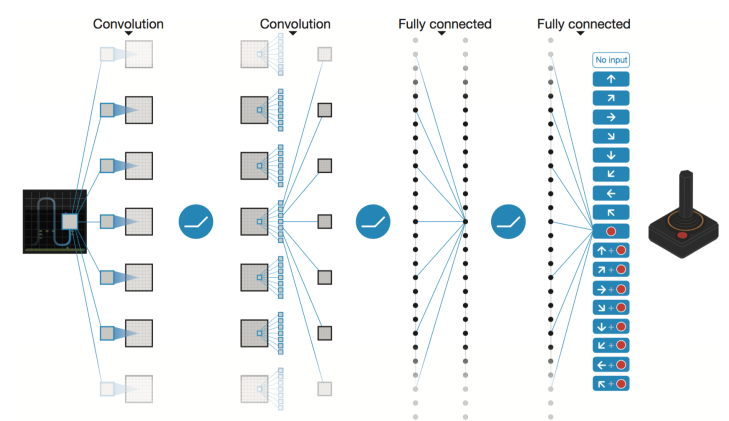
Control using VFA:Q-function公式类似，但是很难保证收敛性

* Neural networks：VFA可以处理很多给定特征集的问题，但是神经网络也是具有自动从图像等原始数据中提取特征的优势。输入原始数据，产生值函数，值函数与概率乘正比，直接用于产生各个动作的概率就好
* Decision trees
* Nearest neighbors
* Fourier / wavelet bases

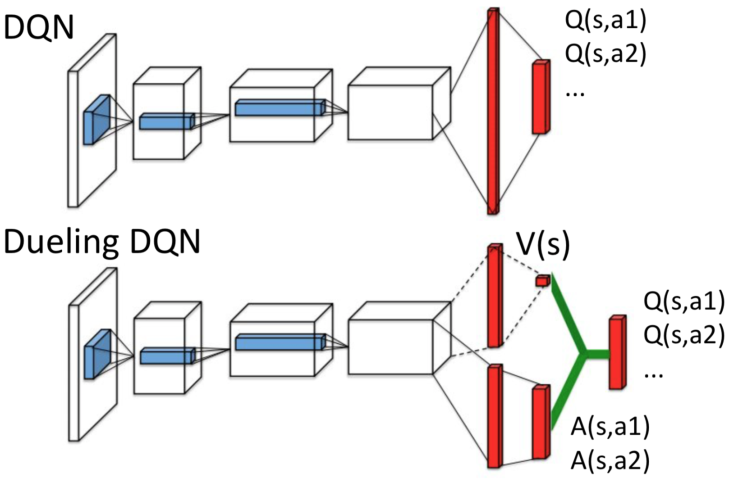
1. notes6- CNNs and Deep Q Learning

learn successful policies directly from high-dimensional inputs。以下是几种常用算法，输入状态s或图像等原始图像，输出q，问题：s存储时怎么比较差异，因为它是连续的：

* Deep Q-Network (DQN) [1] ：为了解决之前学习和训练非常不稳定问题，加入了experience replay, and a separate target network。算法如下：



* Double DQN [2] ：To prevent overestimation and reduce bias。
* Dueling DQN [3].



1. notes7- Imitation Learning

However, if we have known structure in the problem, or we have outside knowledge that we can use, we can explore considerably more e‑ciently. In this lecture,we will talk about how to imitate and learn from human (or expert, generally) behavior on tasks.agent自己探索是会面临回报稀疏的问题，加入模仿学习，能使回报更稠密，加快学习速度。

* Imitation Learning
* Inverse Reinforcement Learning (IRL)是模仿学习的
* Apprenticeship Learning
* Maximum Entropy Inverse RL

1. notes8- Maximum Entropy Inverse RL

our goal is to directly find the policy with the highest value function V π（**Policy-based**）, rather than first finding the value-function of the optimal policy and then extracting the policy from it（value**-based**）.

* Gradient-free optimistic methods
* • Hill climbing
* • Simplex / amoeba / Nelder Mead
* • Genetic algorithms
* • Cross-Entropy method (CEM)
* • Covariance Matrix Adaptation (CMA)
* • Evolution strategies
* Gradient-based optimistic methods

1. 1