**<!--蒙特卡洛、TD、动态规划的关系？-->**

动态规划属于有模型强化学习的范畴，蒙特卡洛和TD都属于无模型的范畴。

蒙特卡洛和TD都需要通过采样来获取数据进行学习

蒙特卡洛需要采样整个样本序列，无偏差，但是方差大

TD只需要采样一个动作，偏差大，但是方差小

**<!--DQN的几个变种以及各自解决了那些问题？-->**

double dqn：加一个target q net，用eval q选择q最大的a，用target q计算q值，用来缓解q值的过估计问题；

dueling dqn：使用V+A来代替Q，缓解q的过估计问题

n-step q：用n-step TD缓解q值估计的偏差；

PER dqn：用带优先级的经验回放，增加样本的学习利用率，增加探索性；

c51：用分布z代替标量来估计q值，实际中用直方图的形式来表示z；

nosiy dqn：用网络参数空间的噪声代替epsilon-greedy，增加探索性，探索的更平滑。

rainbow：集合上面全部的trick。还有分位数回归dqn、IQN。。。

**<!--深度强化学习中的DQN和A3C区别与联系？-->**

DQN和A3C都属于基于深度学习的强化学习方法，整体思路架构类似

DQN属于value-based的方法，A3C属于AC框架的方法（结合policy和value）

A3C属于分布式强化学习的范畴（Actor-Learner）

**<!--策略梯度的推导过程？-->**

针对E[G],V,E[Q]

可以按照s，a进行展开

也可以按照时间t进行展开e

**<!--策略梯度和actor-critic的关系与对比？-->**

Actor-Critic方法结合了基于值的方法（Critic）和基于策略梯度的方法（Actor），由于借用了基于值的方法，使得其可以进行单步更新。

**<!--A3C和DDPG区别和共同点？-->**

都属于AC系列的方法

A3C属于分布式强化学习的范畴（Actor-Learner）

DDPG是确定性策略的方法，需要通过添加噪声来增加探索

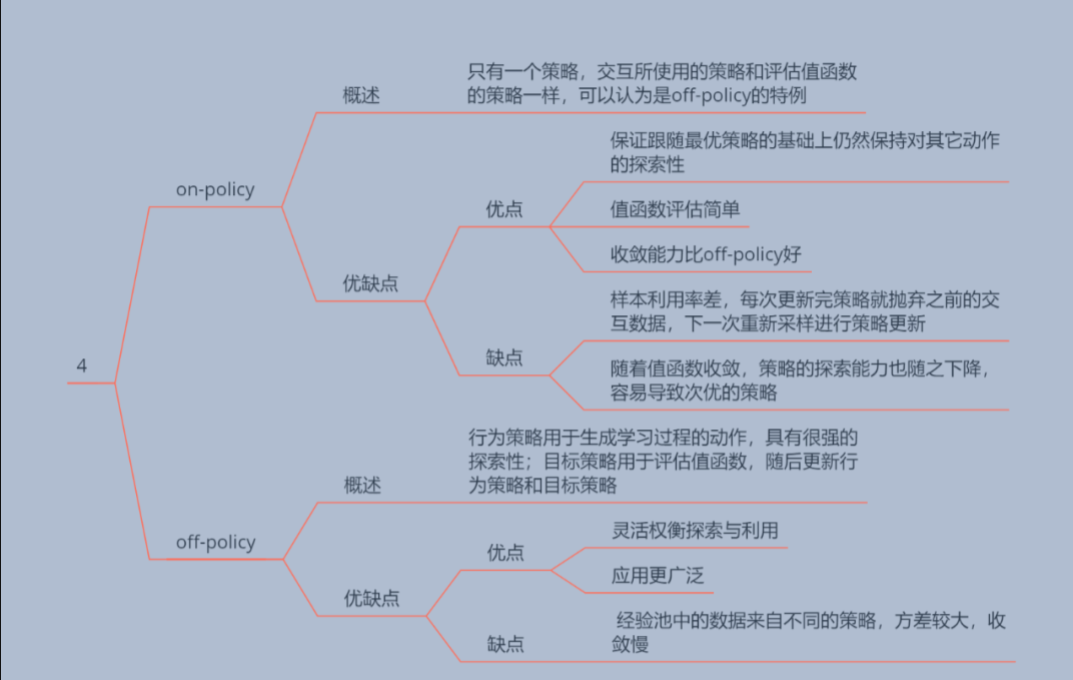
**<!--value-based和policy-based关系？-->**

value-based的方法是先通过计算出值函数，然后再求策略policy-based的方法则是直接计算策略，更加直接，收敛速度更快，但是也更容易达到局部最优

value-based的方法方差小偏差大，policy-based方法无偏差方差大

value-based的方法一般用来解决离散动作问题，policy-based方法一般用于解决连续动作问题

#### **<!--off-policy和on-policy的好与坏？-->**



off可以使用不同的数据收集和策略评估更新策略，更加灵活，更加通用，但是采样效率低，而且会产生分布偏移问题，需要采用重要性采样来进行调整

on使用相同数据收集和评估策略，采样效率较高，但是不够灵活

**<!--表格式到函数近似的理解？-->**

表格表达能力有限，只能处理有限范围的状态和动作空间，但是查询较快

**<!--Actor-Critic的优点？-->**

同时借鉴了value-based和policy-based方法的优点，可以处理更大的状态动作空间，收敛更快，无偏差，而且可以单步更新。

**<!--Actor和Critic两者的区别？-->**

Actor：policy-based

Critic：value-based

**<!--advantage(优势函数)推导过程，如何计算？-->**

知乎天津包子馅儿-TRPO推导

**<!--DPG、DDPG、D3PG、D4PG之间的区别？-->**

DPG确定性策略方法，不需要重要性采样

DDPG借鉴了DQN的思想，引入了replay-buffer

D3PG TD3

D4PG DDPG的分布式模式

**<!--强化学习是什么？和有监督学习的异同？SL靠的是样本标签训练模型，RL依靠的是什么？-->**

依靠奖励驱动，通过与环境交互来获取样本并对智能体进行训练并用来解决序列决策问题的算法。

它不需要训练集测试集，而是通过与环境交互来获取数据，因此这表明其获取的数据时不够稳定的。

**<!--强化学习用来解决什么问题？-->**

序列决策问题

**<!--强化学习的损失函数是什么？-->**

例：

MSE（Qvalue）

策略 期望

**<!--为什么最优值函数就等同最优策略-->**

基于值的方法中是为了去最优的值函数，最优值函数代表从当前状态（或状态动作）出发，能够获取的奖励和，因此奖励和越大的预示着当前状态（或状态，动作）越应该被选择

**<!--强化学习和动态规划的关系；-->**

动态规划可以用于解决有模型强化学习问题

**<!--简述TD算法-->**

r+γQ来预计Q值

**<!--蒙特卡洛和时间差分的对比：MC和TD分别是无偏估计吗，为什么？MC、TD谁的方差大，为什么？-->**

MC是无偏估计，因为其采样了整个决策序列

MC方差大

**<!--简述Q-Learning，写出其Q(s,a)更新公式-->**

r+γQ-Q

**<!--简述值函数逼近的想法？-->**

解决表格式方法适用场景有限的问题

**<!--RL的马尔科夫性质? t+1t+1t+1 时的状态仅与 ttt 时的状态有关，而与更早之前的历史状态无关。-->**

马尔科夫性

马尔科夫过程

马尔科夫决策过程

**<!--RL与监督学习和无监督学习的区别-->**

依靠奖励驱动，通过与环境交互来获取样本并对智能体进行训练并用来解决序列决策问题的算法。

它不需要训练集测试集，而是通过与环境交互来获取数据，因此这表明其获取的数据时不够稳定的。

**<!--RL不同于其它学习算法的原因?-->**

依靠奖励驱动，通过与环境交互来获取样本并对智能体进行训练并用来解决序列决策问题的算法。

它不需要训练集测试集，而是通过与环境交互来获取数据，因此这表明其获取的数据时不够稳定的。

**<!--Model-based和model-free的区别？-->**

model-free不需要构造模型，可以学习，但是无模型的方法样本效率比较低，而且在有些数据难以获取或者获取困难的场景中难以应用，无模型方法通用但是泛化能力弱。。

model-based需要首先构建模型，然后再进行学习（采用有模型和无模型方法都可以），有模型可以通过构建虚拟模型来代替现实模型来发挥其优势（黑盒模型），也可以采用白盒模型来进行学习。有模型方法不通用但是泛化能力强

主要解决了样本不好获取和样本效率低的问题。但是MFRL的效果往往优于MBRL

**<!--确定性策略和 随机性策略的区别与联系？-->**

确定性策略每次输出的动作都是固定的，为了探索需要加入相应的噪声平滑（DDPG，TD3）。

随机策略输出的动作的概率，不需要额外探索。

但是两种方法在代码实现时，对于连续动作空间最终都是输出一个动作。

**<!--on-policy 和off-policy的区别与联系？-->**

off可以使用不同的数据收集和策略评估更新策略，更加灵活，更加通用，但是采样效率低，而且会产生分布偏移问题，需要采用重要性采样来进行调整

on使用相同数据收集和评估策略，采样效率较高，但是不够灵活

————————————————

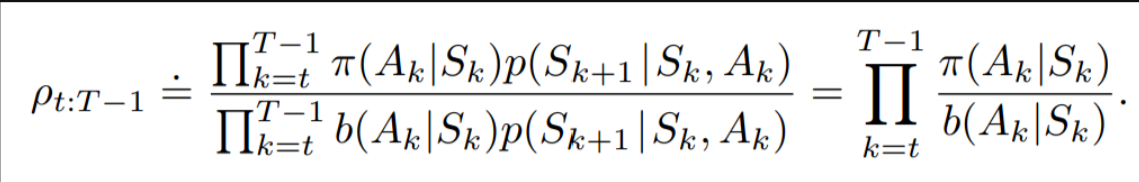
**<!--重要性采样的推导过程、作用？-->**

解决分布偏移问题

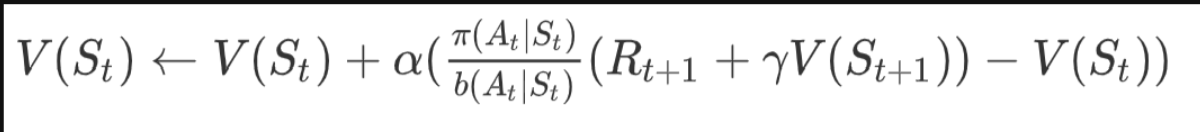
目标策略是用于优化的策略，是评估QV值时使用的策略

基于重要性采样的MC：

重要性采样系数：



基于重要性采样的TD：



=============================================================

下面我们分析下TD的重要性采样，理解行为策略如何向目标策略靠拢：

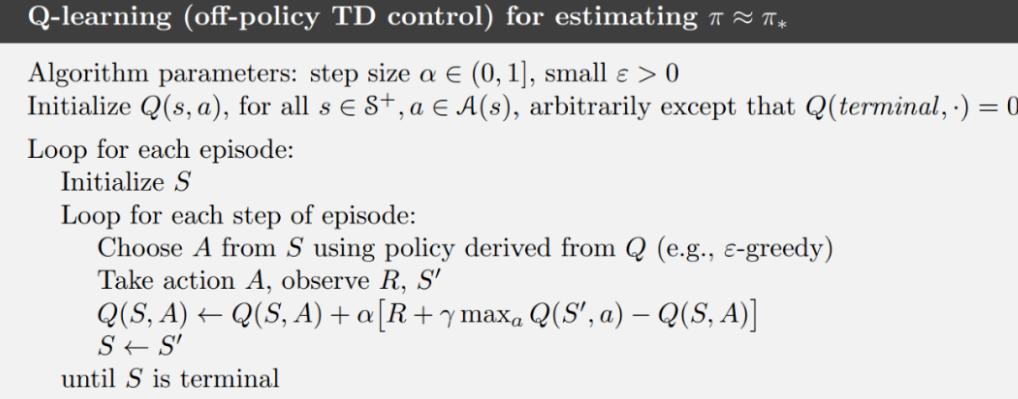
注意，当不为0而为0时，重要性采样不可用。

当，即两策略在状态时选择的概率一致，不需要矫正

当，即行为策略b的在状态时选择动作的概率要比目标策略高，那么小于1的重要性采样系数就会降低基于行为策略b得到的时序差分目标，从而引导代理后续降低行为策略b在状态时选择动作的概率。

当，即目标策略状态时选择动作的概率要比行为策略b高，那么大于1的重要性采样系数就会进一步增强基于行为策略b得到的时序差分目标，从而引导代理后续提高行为策略b在状态时选择动作的概率。

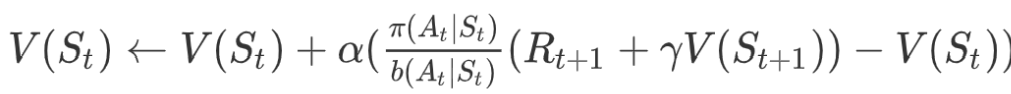
**<!--Q-learning是off-policy的方法，为什么不使用重要性采样？-->**



从Q-learning的算法中可以看出，其行为策略为-greedy策略，目标策略是greedy策略，因此属于off-policy方法。那么为什么没有用重要性采样呢?

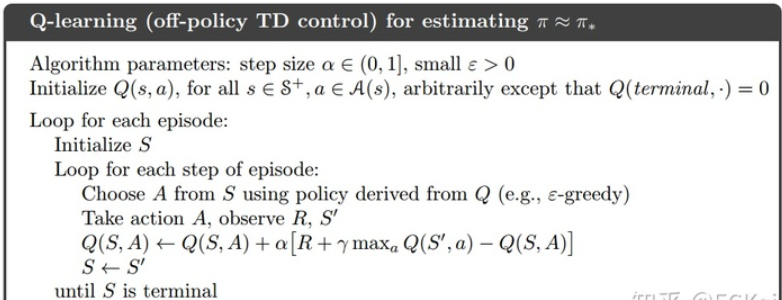
我们可以看出Q-learning在估计时序差分目标时，并没有使用行为策略来进行动作选择，是根据奖励函数得到，与动作选择无关，虽涉及动作选择，但使用的是目标策略而不是行为策略，因此这不需要用重要性采样来矫正估计值。

我们对比基于TD的重要性采样做法：



代理此时处于状态，在估计时序差分目标时，就需要使用行为策略来选择状态下的动作，从而根据奖励函数和状态转移函数得到和。因为使用行为策略而不是目标策略进行动作选择，所以需要重要性采样来矫正估计值。

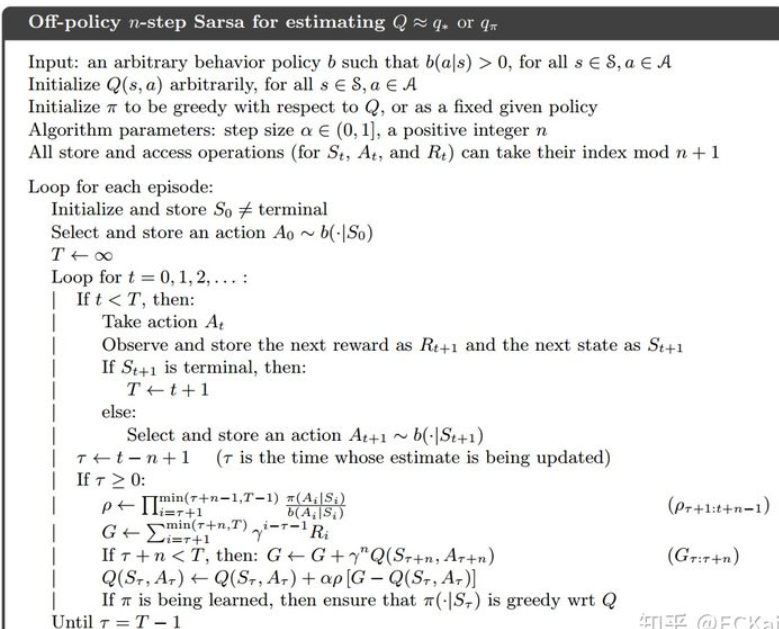
由于Q-learning采用的是off-policy，如下图所示



但是为什么不需要重要性采样。其实从上图算法中可以看到，动作状态值函数是采用1-step更新的，每一步更新的动作状态值函数的R都是执行本次A得到的，而我们更新的动作状态值函数就是本次执行的动作A的 。就算A不是通过 策略选择的（是通过 采样得到），对 估计的更新也不会受到任何影响。因为对于下一个动作的选取是贪心的 。但是如果是n-step更新，则会出现问题



如上式所示，还是利用off-policy采样， 的采样没有问题，但是 就会出问题，因为这一步的动作 是使用 得到的，不是我们要优化的 策略。因此这种更新方法才需要添加重要性采样系数。如下面这个算法所示



这是因为，在1-step中，我们更新 状态动作对时，并不需要知道他是根据什么分布得来的。但是在n-step中，如果想利用 以后的信息时，我们就需要使用重要性采样来处理后序的动作。

注意Q和V的区别，Q不用重要性采样，V需要重要性采样

关键在于是否用行为策略选择的动作的结果来进行值函数或者策略的更新

————————————————

**<!--有哪些方法可以使得RL训练稳定?-->**

选用方差较小的方法

采用合理的奖励函数

**<!--写出贝尔曼期望方程和贝尔曼最优方程?-->**

sutton书

**<!--贝尔曼期望方程和贝尔曼最优方程什么时候用?-->**

贝尔曼期望方程，值函数学习

大部分强化学习方法都有使用（只不过选择的动作是根据策略选择的）

贝尔曼最优方程，获取最优值函数

值迭代

**<!--策略梯度算法的目标函数和策略梯度计算?-->**

以s，a展开

以t展开

**<!--DQN的原理？-->**

DQN对Q-learning的修改主要体现在以下三个方面：

DQN利用深度卷积神经网络逼近值函数（表达能力增强，可以处理状态空间动作空间较大的问题）

DQN利用了经验回放对强化学习的学习过程进行训练（打破数据关联性，保证输入神经网络的数据独立同分布）

DQN独立设置了目标网络来单独处理时间差分算法中的TD偏差。（打破关联性，提高收敛能力）

**<!--DQN和Sarsa的区别？-->**

DQN对Q-learning的修改主要体现在以下三个方面：

DQN利用深度卷积神经网络逼近值函数（表达能力增强，可以处理状态空间动作空间较大的问题）

DQN利用了经验回放对强化学习的学习过程进行训练（打破数据关联性，保证输入神经网络的数据独立同分布）

DQN独立设置了目标网络来单独处理时间差分算法中的TD偏差。（打破关联性，提高收敛能力）

而Sarsa是on-policy方法，Ql和DQN都是off-policy方法

**<!--为什么使用优势函数？-->**

保证总的回报期望不断增加而不减少。（TRPO，PPO）

**<!--常见的平衡探索与利用的方法？-->**

Epsilon-greedy

UCB

entropy

好奇心机制

**<!--TD3如何解决过估计？-->**

采用双Q策略

**<!--TD3和DDPG的区别？-->**

采用双Q策略

策略参数延迟更新

目标动作选择平滑

**<!--多臂老虎机和强化学习算法的差别？-->**

MAB问题只有一个状态，是在同一个状态不断执行不同的bandit选择，更倾向于寻找合适的探索利用机制

在经典bandit算法当中，我们要做的无非就是通过一类pull arm的策略，尽快找到比较好的“arm”（reward较高的arm），然后尽可能多的去拉这些比较好的arm就是了。贪心算法无非就是永远以当前对每个arm的reward的估计直接作为依据，而UCB算法则是考虑了置信度的问题，因此考虑的是每个arm reward的置信区间的上界。

**<!--多臂老虎机算法的分类？-->**

bandit算法中主要有两类流行的算法，一类是贪心算法（如uniform exploration， -greedy算法），还有一类是基于upper confidence bound的UCB算法。

**<!--有那几种Bandit算法？-->**

Epsilon-greedy

UCB

**<!--简述UCB算法 （Upper Confidence Bound)？-->**

UCB公式

————————————————

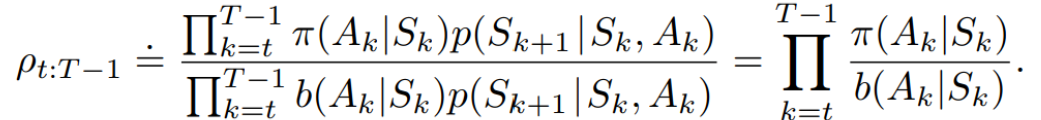
**<!--简述重要性采样，Thompson sampling采样？-->**

解决分布偏移问题

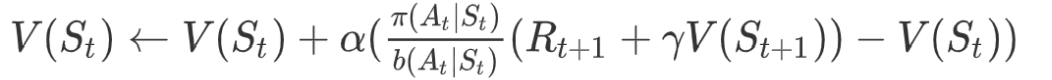
目标策略是用于优化的策略，是评估QV值时使用的策略

基于重要性采样的MC：

重要性采样系数：



基于重要性采样的TD：



=============================================================

下面我们分析下TD的重要性采样，理解行为策略如何向目标策略靠拢：

注意，当不为0而为0时，重要性采样不可用。

当，即两策略在状态时选择的概率一致，不需要矫正

当，即行为策略b的在状态时选择动作的概率要比目标策略高，那么小于1的重要性采样系数就会降低基于行为策略b得到的时序差分目标，从而引导代理后续降低行为策略b在状态时选择动作的概率。

当，即目标策略状态时选择动作的概率要比行为策略b高，那么大于1的重要性采样系数就会进一步增强基于行为策略b得到的时序差分目标，从而引导代理后续提高行为策略b在状态时选择动作的概率。

————————————————

**<!--什么是强化学习？-->**

依靠奖励驱动，通过与环境交互来获取样本并对智能体进行训练并用来解决序列决策问题的算法。

它不需要训练集测试集，而是通过与环境交互来获取数据，因此这表明其获取的数据时不够稳定的。

适宜解决序列决策问题，因为传统的监督学习相当于通过单步学习来进行训练，效果肯定不如整个序列的值来进行训练

**<!--强化学习和监督学习、无监督学习的区别是什么？-->**

依靠奖励驱动，通过与环境交互来获取样本并对智能体进行训练并用来解决序列决策问题的算法。

它不需要训练集测试集，而是通过与环境交互来获取数据，因此这表明其获取的数据时不够稳定的。

数据之间是相关的

**<!--强化学习适合解决什么样子的问题？-->**

序列决策问题

**<!--强化学习的损失函数（loss function）是什么？和深度学习的损失函数有何关系？-->**

MSE

**<!--POMDP是什么？马尔科夫过程是什么？马尔科夫决策过程是什么？里面的“马尔科夫”体现了什么性质？-->**

部分不可观测马尔科夫决策过程，含有隐状态和观测状态。

马尔科夫性

**<!--贝尔曼方程的具体数学表达式是什么？-->**

sutton书

**<!--最优值函数和最优策略为什么等价？-->**

基于值的方法中是为了去最优的值函数，最优值函数代表从当前状态（或状态动作）出发，能够获取的奖励和，因此奖励和越大的预示着当前状态（或状态，动作）越应该被选择

**<!--值迭代和策略迭代的区别？-->**

策略迭代是通过策略评估和策略改善两个分步骤来进行的,使用的是贝尔曼方程

值迭代只有一个步骤，使用的贝尔曼最优方程

**<!--如果不满足马尔科夫性怎么办？当前时刻的状态和它之前很多很多个状态都有关之间关系？-->**

如果不满足马尔科夫性，强行只用当前的状态来决策，势必导致决策的片面性，得到不好的策略。 为了解决这个问题，可以利用RNN对历史信息建模，获得包含历史信息的状态表征。表征过程可以 使用注意力机制等手段。最后在表征状态空间求解MDP问题。有点类似POMDP的感觉

**<!--求解马尔科夫决策过程都有哪些方法？有模型用什么方法？动态规划是怎么回事？-->**

有模型 无模型

有模型：策略迭代，值迭代

无模型：MC，TD

**<!--简述动态规划(DP)算法？-->**

有模型方法。通过建立好的MDP模型（状态转移，奖励）和状态转移方程（贝尔曼方程）对于值函数表进行更新

**<!--简述蒙特卡罗估计值函数(MC)算法。-->**

无模型方法。采样整个序列，然后再进行更新（首次遇见，次次遇见）

**<!--简述时间差分(TD)算法。-->**

无模型方法。也是采样整个序列，但是在决策序列的每一步都可以进行更新

**<!--简述动态规划、蒙特卡洛和时间差分的对比（共同点和不同点）-->**

动态规划属于有模型强化学习的范畴，蒙特卡洛和TD都属于无模型的范畴。

蒙特卡洛和TD都需要通过采样来获取数据进行学习

蒙特卡洛需要采样整个样本序列，无偏差，但是方差大

TD只需要采样一个动作，偏差大，但是方差小

**<!--MC和TD分别是无偏估计吗？-->**

MC是，TD不是

**<!--MC、TD谁的方差大，为什么？-->**

MC方差大，采样多

**<!--写出用第n步的值函数更新当前值函数的公式（1-step，2-step，n-step的意思）。当n的取值变大时，期望和方差分别变大、变小？-->**

采样几步之后在更新，n变大，偏差变小方差变大

**<!--TD（λ）方法：当λ=0时实际上与哪种方法等价，λ=1呢？-->**

λ=0 单步时序差分

λ=1 蒙特卡洛方法

**<!--写出蒙特卡洛、TD和TD（λ）这三种方法更新值函数的公式？-->**

sutton书

**<!--value-based和policy-based的区别是什么？-->**

value-based的方法是先通过计算出值函数，然后再求策略policy-based的方法则是直接计算策略，更加直接，收敛速度更快，但是也更容易达到局部最优

value-based的方法方差小偏差大，policy-based方法无偏差方差大

value-based的方法一般用来解决离散动作问题，policy-based方法一般用于解决连续动作问题

**<!--DQN的两个关键trick分别是什么？-->**

replay buffer

目标网络

**<!--阐述目标网络和experience replay的作用？-->**

减少关联性，保证神经网络数据独立同分布，保留好较好的数据

**<!--手工推导策略梯度过程？-->**

前面有，可以根据s,a推，也可根据t推

最后的得出基于序列数据进行更新的REINFORCE方法

**<!--描述随机策略和确定性策略的特点？-->**

前面有

**<!--不打破数据相关性，神经网络的训练效果为什么就不好？-->**

神经网络有数据独立同分布假设

**<!--画出DQN玩Flappy Bird的流程图。在这个游戏中，状态是什么，状态是怎么转移的？奖赏函数如何设计，有没有奖赏延迟问题？-->**

有奖励延迟

奖励如果设置为是否赢得比赛，则是延迟奖励

**<!--DQN都有哪些变种？引入状态奖励的是哪种？-->**

double dqn：加一个target q net，用eval q选择q最大的a，用target q计算q值，用来缓解q值的过估计问题；

dueling dqn：引入状态奖励

n-step q：用n-step TD缓解q值估计的偏差；

PER dqn：用带优先级的经验回放，增加样本的学习利用率，增加探索性；

c51：用分布z代替标量来估计q值，实际中用直方图的形式来表示z；

nosiy dqn：用网络参数空间的噪声代替epsilon-greedy，增加探索性，探索的更平滑。

rainbow：集合上面全部的trick。还有分位数回归dqn、IQN。。。

**<!--简述double DQN原理？-->**

DQN有一个显著的问题，就是DQN估计的Q值往往会偏大。这是由于我们Q值是以下一个s'的Q值的最大值来估算的，但下一个state的Q值也是一个估算值，也依赖它的下一个state的Q值...，这就导致了Q值往往会有偏大的的情况出现。

们在同一个s'进行试探性出发，计算某个动作的Q值。然后和DQN的记过进行比较就可以得出上述结论。

这种欺上瞒下的做法，实在令人恼火。于是有人想到一个互相监察的想法。

这个思路也很直观。如果只有一个Q网络，它不是经常吹牛嘛。那我就用两个Q网络，因为两个Q网络的参数有差别，所以对于同一个动作的评估也会有少许不同。我们选取评估出来较小的值来计算目标。这样就能避免Q网络吹牛的情况发生了。

另外一种做法也需要用到两个Q网络。Q1网络推荐能够获得最大Q值的动作；Q2网络计算这个动作在Q2网络中的Q值。

恰好，如果我们用上Fixed Q-targets，我们不就是有两个Q网络了吗？

所以你可以看到，这个优化在DQN上很容易实现。这就是doubleDQN和DQN的唯一的变化。

**<!--策略梯度方法中基线baseline如何确定？-->**

不能包含a（不引入偏差），可以减少方差

**<!--什么是DDPG，并画出DDPG框架结构图？-->**

利用DQN的框架，引入了replay buffer和目标网络

是一种确定性策略的方法，回报期望的导数中不包含重要性采样项

**<!--Actor-Critic两者的区别是什么？-->**

Actor-Critic方法结合了基于值的方法（Critic）和基于策略梯度的方法（Actor），由于借用了基于值的方法，使得其可以进行单步更新。

**<!--actor-critic框架中的critic起了什么作用？-->**

提供对于actor中策略更新时对于值函数的估计，可以单步更新。

这也是其与普通的policy-based方法的区别，AC系列方法要求值函数Critic部分必须有估计误差

**<!--DDPG是on-policy还是off-policy，为什么？-->**

off-policy

类似于DQN，采集数据放到replay buffer中然后进行学习，采集数据采用的策略和学习的策略是不同的

**<!--是否了解过D4PG算法？简述其过程-->**

DDPG的分布式版本

Actor收集数据序列（n步奖励更新）

Learner进行训练

**<!--简述A3C算法？A3C是on-policy还是off-policy，为什么？-->**

多个Actor进行与环境交互，并计算梯度，随后每一个Actor周期性暂停学习，与中央参数服务器交互更新参数。

on-policy，因为在本地进行的梯度计算

**<!--A3C算法是如何异步更新的？是否能够阐述GA3C和A3C的区别？-->**

每一个Actor周期性暂停学习，与中央参数服务器交互更新参数

GA3C并不在Actor本地计算梯度，Actor只负责产生数据

**<!--简述A3C的优势函数？-->**

Q-A

————————————————

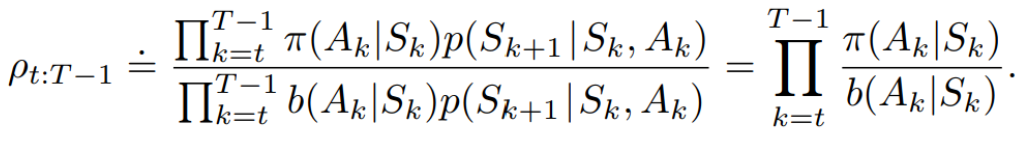
#### **<!--什么是重要性采样？-->**

解决分布偏移问题

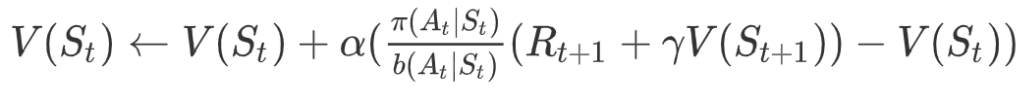
目标策略是用于优化的策略，是评估QV值时使用的策略

基于重要性采样的MC：

重要性采样系数：



基于重要性采样的TD：



下面我们分析下TD的重要性采样，理解行为策略如何向目标策略靠拢：

当，即两策略在状态时选择的概率一致，不需要矫正

当，即行为策略b的在状态时选择动作的概率要比目标策略高，那么小于1的重要性采样系数就会降低基于行为策略b得到的时序差分目标，从而引导代理后续降低行为策略b在状态时选择动作的概率。

当，即目标策略状态时选择动作的概率要比行为策略b高，那么大于1的重要性采样系数就会进一步增强基于行为策略b得到的时序差分目标，从而引导代理后续提高行为策略b在状态时选择动作的概率。

注意，当不为0而为0时，重要性采样不可用。

————————————————

**<!--为什么TRPO能保证新策略的回报函数单调不减？-->**

引入优势函数

**<!--TRPO是如何通过优化方法使每个局部点找到让损失函数非增的最优步长来解决学习率的问题；-->**

引入优势函数，并通过置信域方法进行优化

**<!--如何理解利用平均KL散度代替最大KL散度？-->**

最大KL散度有无数个，不好进行计算

**<!--简述PPO算法？与TRPO算法有何关-->**

升级版，更加简单，有两种

第一种，clip版

第二种，直接将约束加入优化目标

**<!--简述DPPO和PPO的关系？-->**

PPO是DPPO的基本元素，DPPO是分布式强化学习算法。

**<!--强化学习如何用在推荐系统中？-->**

方式1：

Agent：推荐引擎。

Environment：用户。

Reward：如果一条新闻被点击，计+1，否则为0。一次推荐中10条新闻被点击的新闻个数作为Reward。

State：包含3个部分，分别是用户标签、候选新闻的新闻标签和用户前4屏的点击历史（如果没有就置0）。

Action：推出的10篇新闻。

方式2：

状态S:定义为用户的使用历史，比如用户过去在时间 t 前点击、下载、安装、购买过的 N 个 item。而且这个序列要按照时间顺序。

动作A:被模型基于用户状态 s计算出来的一个推荐列表（多个item）

奖励R：推荐智能体给出了at之后，用户会给出他的反馈，他可以进行曝光、点击、下单，智能体立即根据这些反馈计算出来奖励

**<!--推荐场景中奖赏函数如何设计？-->**

用户的反馈

**<!--场景中状态是什么，当前状态怎么转移到下一状态？-->**

推荐系统推荐所需要的信息（用户信息，商品信息等）

**<!--自动驾驶和机器人的场景如何建模成强化学习问题？MDP各元素对应真实场景中的哪些变量？-->**

**<!--强化学习需要大量数据，如何生成或采集到这些数据？-->**

与真实环境交互或者采用有模型强化学习，先建立模型，然后与建立的虚拟模型进行交互

**<!--是否用某种DRL算法玩过Torcs游戏？具体怎么解决？-->、**

状态：车辆的信息

含有两类动作，1.控制左右2.控制油门刹车

奖励就是偏离最佳行驶路线的程度

**<!--是否了解过奖励函数的设置(reward shaping)？-->**

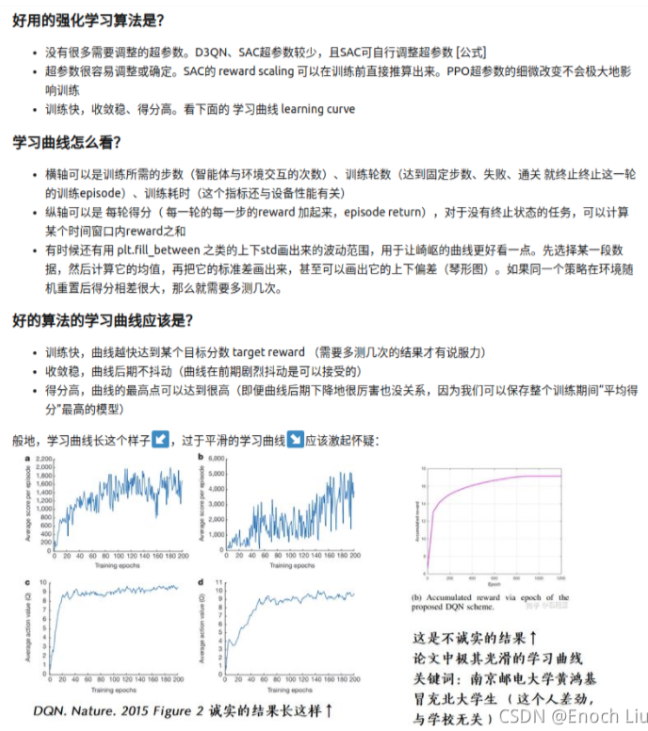
在原本的reward函数中加入一项惯性（TD error），γV-V

**<!--强化学习中如何处理归一化？-->**

对于状态，不能进行BN，因为BN的目的虽然是通过将数据分布保持标准正态分布提高网络的适应能力来加快网络的训练速度，但是它需要稳定的数据来进行方差和均值计算，但是强化学习恰好有训练数据不够稳定的特点，所以不适宜使用BN，但是可以使用普通的归一化来使得输入让神经网络更舒服。而且对于值函数也不需要进行归一化，但可以进行缩放.

————————————————

#### **<!--强化学习如何观察收敛曲线？-->**



**<!--强化学习如何如何确定收敛？-->**

序列总奖励保持在一个较高的稳定范围内

**<!--影响强化学习算法收敛的因素有哪些，如何调优？-->**

状态奖励对应

方法

采样效率

**<!--强化学习的损失函数（loss function）是什么？和深度学习的损失函数有何关系？-->**

MSE

**<!--多智能体强化学习算法有哪些？-->**

最普通：IQL

基于通信：RIAL，DIAL,BiCNet

基于：MADDPG->COMA，VDN->QMIX->QTRAN

**<!--简述Model Based Learning？有什么新的进展？比如World Model？Dream？MuZero?-->**

model-based需要首先构建模型，然后再进行学习（采用有模型和无模型方法都可以），有模型可以通过构建虚拟模型来代替现实模型来发挥其优势（黑盒模型），也可以采用白盒模型来进行学习。有模型方法不通用但是泛化能力强

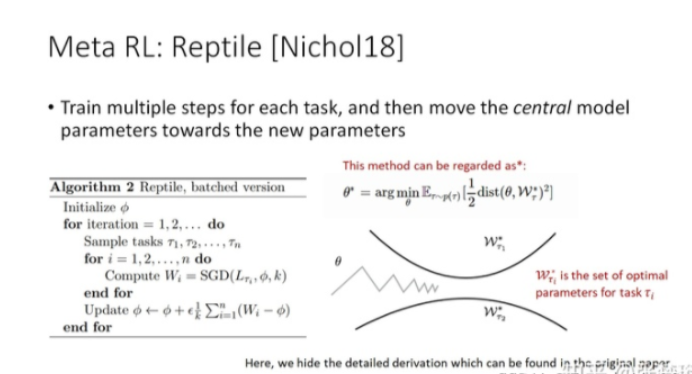
主要解决了样本不好获取和样本效率低的问题。但是MFRL的效果往往优于MBRL

**<!--简述Meta Reinforcement Learning?-->**

通过学习多个预训练任务，来使得模型在新任务上有很好的效果，类似于Few-shot learning，迁移学习，多任务学习

————————————————

#### **<!--为什么Reptile应用的效果并不好？-->**



**<!--Meta RL不好应用的原因有哪些？-->**

前面讲到，要做 meta RL 的训练，很重要的一点是希望在训练智能体的时候，就让智能体见到足够丰富的任务。因此，一个能够高效生成各种丰富任务的环境就显得很重要。往细了讲，就是希望能够生成不同的 P 和 R。

**<!--简述Meta Gradient Reinforcement Learning？-->**

通过学习多个预训练任务，来使得模型在新任务上有很好的效果，类似于Few-shot learning，迁移学习，多任务学习

**<!--简述Imitation Learning？GAIL? Deepminic?-->**

分为监督学习和逆强化学习

监督学习中，就是通过专家的标记的数据来训练agent，提高决策能力

Forward Training，SMILE，Dagger（边训练边扩充数据），GAIL（GAN框架）

逆向强化学习中，就是学习得到一个良好的reward函数

基于最大边际：学徒学习，神经网络逆向学习

基于最大熵

**<!--简述DRL的一些最新改进？R2D3？LASER？-->**

**<!--简述Multi-Agent Reinforcement Learning？ 比如MADDPG比较早的，思想是什么？和一般的DRL有什么区别？-->**

由原来的对一个智能体进行决策变成了对多智能体进行决策，这些多智能体之间可能是竞争，合作或者竞争和合作并存的关系。

在MDP框架之上进行了一定的改进

MADDPG，中心化训练，分布式执行，所有智能体的值函数和策略函数都在中心进行更新，但是他们仍可以独立地执行。

**<!--简述seed rl? 对于大规模分布式强化学习，还有更好的提高throughput的方法吗？-->**

IMPALA 存在着一系列的缺点，例如资源利用率低、无法大规模扩展等。

作者：AI科技评论 链接：ICLR 2020 |谷歌推出分布式强化学习框架SEED，性能“完爆”IMPALA，可扩展数千台机器 - 知乎 来源：知乎 著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

SEED RL 本质上是在 IMPALA 基础上的一个改进。因此我们先简单来看下 IMPALA是如何做分布式强化学习的。

IMPALA 的体系架构分为Actors 和Learners。其中actor一般运行在CPU上，它在环境中采取步骤，然后在模型上运行推断，进行迭代，然后预测下一个动作。Actors会更新推理模型的参数，在收集到足够多的观察数据后，actor会将观察和动作的结果发送给learner，learner根据这些反馈结果进行优化。

实验的结果，我们在文章前面已经展示。这里总结一下谷歌这项工作的意义：

1、 相比 IMPALA，SEED RL 框架将推理部分转移到 Learners 中。这样调整后，Actors中就不再有任何关于神经网络相关的计算了。这样做的好处是，模型的大小不会影响Actors，于是便可扩大Actors的数量，做到可扩展。

2、在SEED RL的架构下，Learner（以GPU或TPU为主）可以专注于批量推理，而Actors（以CPU为主）可以更加适应多环境。整体来说，这种结构会降低实验成本。

3、涉及模型的所有内容都留在Learner本地，只有观测结果和动作会在Learners 和Actors之间进行传输，这可以将带宽需求降低多达99%。

4、使用了具有最小延迟和最小开销的 streaming gRPC，并将批处理集成到服务器模块当中。

据了解，目前SEED RL 框架已经开源，打包后可以非常容易地在Google Cloud 上运行。开源地址是：https://github.com/google-resea

**<!--简述AI-GAs? 你对这个理论有什么看法？-->**

利用GA遗传算法来对参数进行更新，可以提供更好的收敛保证

**<!--简述Out-of-Distributon Generalization? Modularity?-->**

**<!--DRL要实现足够的泛化Generalization有哪些做法？Randomization？-->**

探索利用协调，引入噪声

**<!--简述Neural-Symbolic Learning的方法？怎么看待？-->**

**<!--简述unsupervised reinforcement learning？Diversity is all you need？-->**

**<!--简述offline reinforcement learning？-->**

online RL无论是on-policy还是off-policy都需要通过与环境进行交互来获取数据，而offline RL则是提前准备好数据集，用该数据集来进行学习，这种方法就带了分布偏移问题，所以可以通过重要性采样或KL散度约束原策略和现策略之间的差距来解决。

**<!--简述Multi-Task Reinforcement Learning？ Policy Distillation？-->**

通过一个agent来解决多个任务（例如多个游戏），最简单的方法就是策略蒸馏，也就是从多个teacher智能体（执行不同任务）中获取信息，用这些信息来对student智能体进行训练和学习。

除此之外还有无监督学习（辅助任务）等方法。

**<!--简述sim2real? 有哪些方法？-->**

我们建立的模型常常与现实模型有差距。

**<!--对于drl在机器人上的应用怎么看？-->**

连续动作 policy-basedRL

许多较为复杂的情况要使用分层强化学习来解决

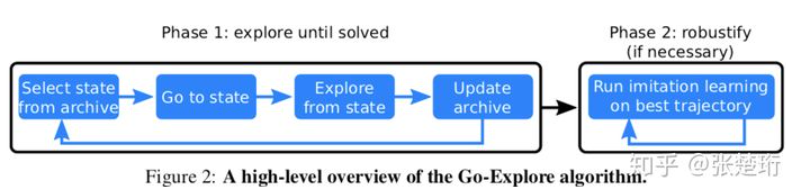
————————————————

**<!--简述go-explore?-->**

Go-Explore旨在解决探索困难（hard-exploration）的问题，这类问题通常奖励稀疏（sparse）并且会有误导性的奖励（deceptive）。这篇文章通过一系列算法设计，使得它在 Montezuma's Revenge 和 Pitfall 游戏上的表现相比于之前的算法有了质的飞跃。

算法分为两个部分。其中第一部分（Phase 1）的最终目的是通过在环境中探索生成一系列高奖励的轨迹；第二部分（Phase 2）使用第一部分生成的轨迹做 imitation learning 并获得最终的策略。

————————————————



第一部分限定在确定性（deterministic）并且可以重启（resettable）的环境中进行，这样的好处是记录下行动序列就可以确定性地到达之前到达的状态上。第一部分的主要思想是维护一个存档（archive），保存着从起始状态到达各个已经探索到状态的路径。每一轮都选择一个已经达到过的状态，然后利用存档中的信息走到这个状态上，接着从这个状态出发做探索，如果探索到了新的状态或者有用的路径，就记录下来加入存档中。注意到这个部分纯粹是做探索并且记录状态和行动序列，这里面并没有使用神经网络。（该部分具体细节见后）

第二部分使用第一部分生成的轨迹做训练。这不仅仅是因为第一部分其实压根没有学习到概括性的策略，而且也是因为第一部分实在确定性的环境中探索的，其策略没法适应真实的随机性环境。文中使用了一种 learning from demostration 的方法 Backward Algorithm。大致上来说，拿到一条高奖励的轨迹之后，先把智能体放在离轨迹末端较近的位置，让智能体能够学习到如何从这个离目标很近状态走到目标，然后再逐步把智能体放在轨迹上离目标更远一些的地方。通过这种方式智能体就能逐步学习到如何从初始位置走到目标位置。其学习的算法还是使用的通用的强化学习算法，比如 PPO 等。这个过程也可以看做是一个课程学习（curriculum learning），使用第一部分生成的反向轨迹来作为由易到难设定的课程。

————————————————

**<!--对于hard exploration的问题，要怎么处理？-->**

建立模型，使用有模型强化学习

奖励稀疏（sparse）并且会有误导性的奖励（deceptive）

**<!--简述Transformer？能否具体介绍一下实现方法？-->**

QKV

还有多头注意力机制

**<!--简述Pointer Network？和一般的Attention有什么不同？-->**

用来解决组合优化问题，解决了传统seq2seq方法不够灵活的特点，还可以结合RL来提供训练数据。

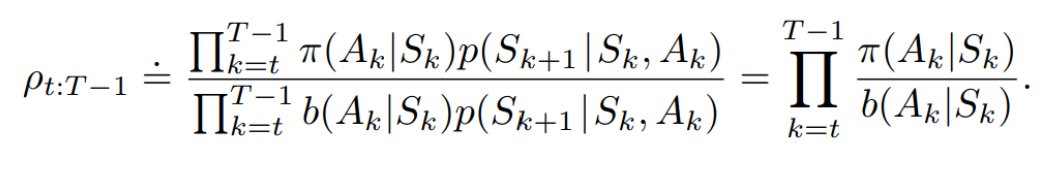
#### **<!--什么是Importance Sampling? 为什么PPO和IMPALA要使用？两者在使用方式上有何不同？能否结合？-->**

解决分布偏移问题

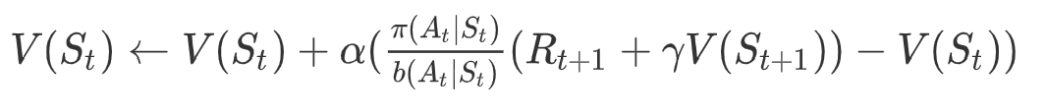
目标策略是用于优化的策略，是评估QV值时使用的策略

基于重要性采样的MC：

重要性采样系数：



基于重要性采样的TD：



下面我们分析下TD的重要性采样，理解行为策略如何向目标策略靠拢：

当，即两策略在状态时选择的概率一致，不需要矫正

当，即行为策略b的在状态时选择动作的概率要比目标策略高，那么小于1的重要性采样系数就会降低基于行为策略b得到的时序差分目标，从而引导代理后续降低行为策略b在状态时选择动作的概率。

当，即目标策略状态时选择动作的概率要比行为策略b高，那么大于1的重要性采样系数就会进一步增强基于行为策略b得到的时序差分目标，从而引导代理后续提高行为策略b在状态时选择动作的概率。

注意，当不为0而为0时，重要性采样不可用。

PPO是因为使用了优势函数。

IMPALA是因为它是分布式强化学习框架，产生数据和训练是分开进行的，无法保证on-policy‘

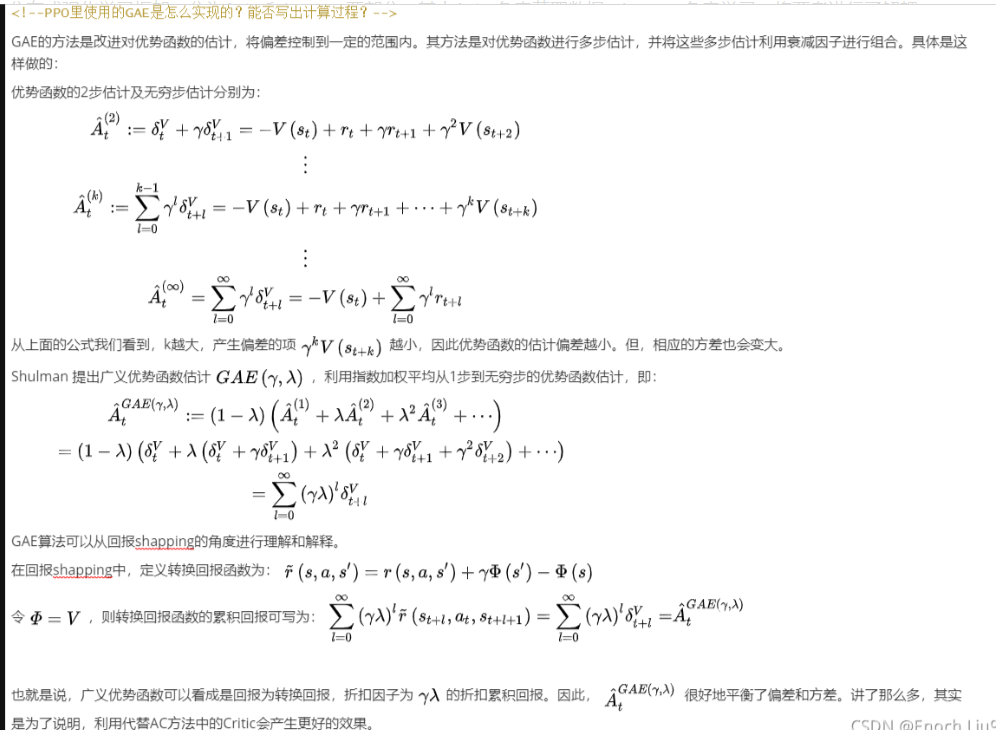
结合结果就是DPPO

————————————————

#### **<!--PPO在实现上是怎么采样的？-->**

用策略网络进行采样，每次连续采样一整个序列在进行训练学习。

#### **<!--PPO里使用的GAE是怎么实现的？能否写出计算过程？-->**



**<!--是否理解Entropy，KL divergence和Mutual Information的含义？-->**

Entropy 数据的离散程度（惊奇度）

KL散度衡量两个分布之间的距离

互信息

**<!--AlphaStar的scatter connection？怎么实现的？-->**

**<!--对于多个entity的observation，你会怎么预处理？神经网络要怎么构建？-->**

拥有各自的一段网络，随后进行汇总

**<!--AlphaStar的League，能否解释一下？如何让agent足够diverse？-->**

**<!--Inverse RL 能否解决奖励问题，如何解决的？-->**

可以，逆向强化学习就是为了学习到一个较好的奖励函数，它是通过采用最大边际的方法，尽量增加在学习到的奖励下，专家策略和非专家策略之间累计奖励的距离。

**<!--分层强化学习的原理是什么 ？-->**

通过构造多层奖励函数，将一个复杂的问题分层为简单的问题

**<!--简述分层强化学习中基于目标的(goal-reach)和基于目标的(goal-reach）的区别与联系？-->**

主要解决奖励系数的问题

通过一个元控制器负责目标层级的选择，和一个普通控制器负责细粒度层级的选择

**<!--请简述IQL（independent Q-learning算法过程？-->**

每个agent都拥有自己独立的算法，而且将别的智能体当作环境的一部分

————————————————

**<!--是否了解α−Rank算法？-->**

支撑α-Rank的进化观点是，游戏中的智能体之间通过相互作用，构成了一个不断变化的种群的动态系统，其中较为强大的代理会复制并取代较弱的对应物。

为了计算智能体之间的排名,α-Rank在从一个智能体向另一个演化的过程中会构建出一个图表（如下图）。这一群类在游戏过程中花费的平均时间就构成了每个智能体的评级数据。

作者：数据汪 链接：DeepMind发布多智能体协作最新评估方法α-Rank，登上Nature - 知乎 来源：知乎 著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

**<!--请简述QMIX算法？-->、**

每个智能体都可以根据Qi（si,ai）来在本地单独进行决策，而训练的时候则是通过一个Mixing网络将所有智能体的Q进行混合，对混合后的结果进行训练。

**<!--简述模仿学习与强化学习的区别、联系？-->**

模仿学习和强化学习都是为了学习到一个好的策略，目标相同

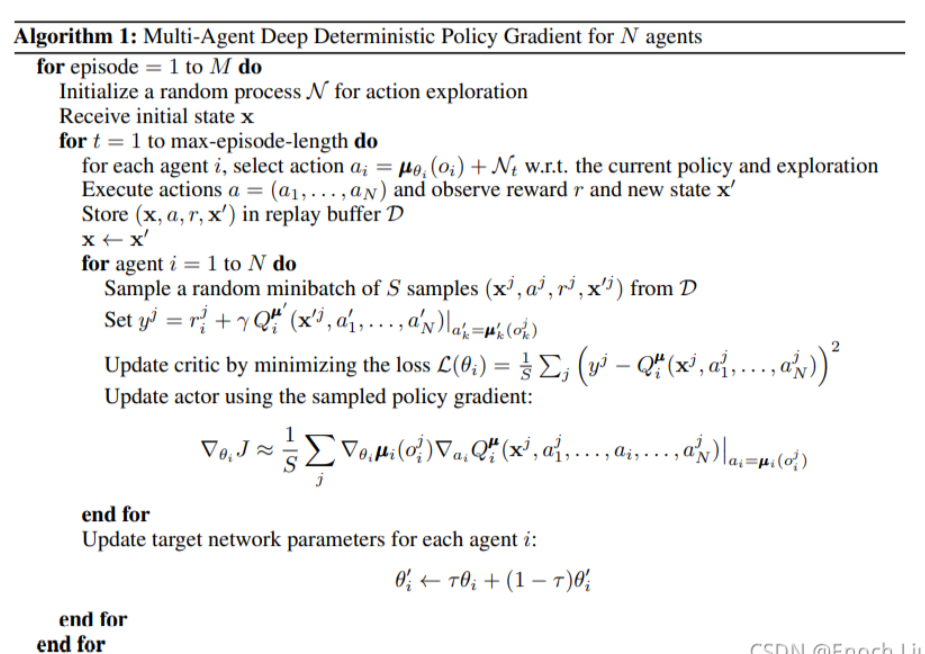
模仿学习是通过在提供专家数据的前提下利用专家数据进行学习，而强化学习则是完全采用自己与环境交互生成的数据来进行学习。

模仿学习的速度和效率更高但是需要专家来标注数据

————————————————



#### **<!--简述MADDPG算法的过程和伪代码？-->**



**<!--多智能体之间如何通信、如何竞争？-->**

RIAL，DIAL

通过设置奖励函数来竞争

**<!--你熟悉的多智能体环境有哪些？-->**

贪吃蛇，捉迷藏

**<!--你做过的强化学习项目有哪些，遇到的难点有哪些？-->**

算法效果不稳定

奖励函数失效

训练时间

**<!--请简述造成强化学习inefficient的原因？-->**

在环境中随机采样，奖励函数稀疏，延迟回报

**<!--sarsa的公式以及和Q-leaning的区别？-->**

on-policy

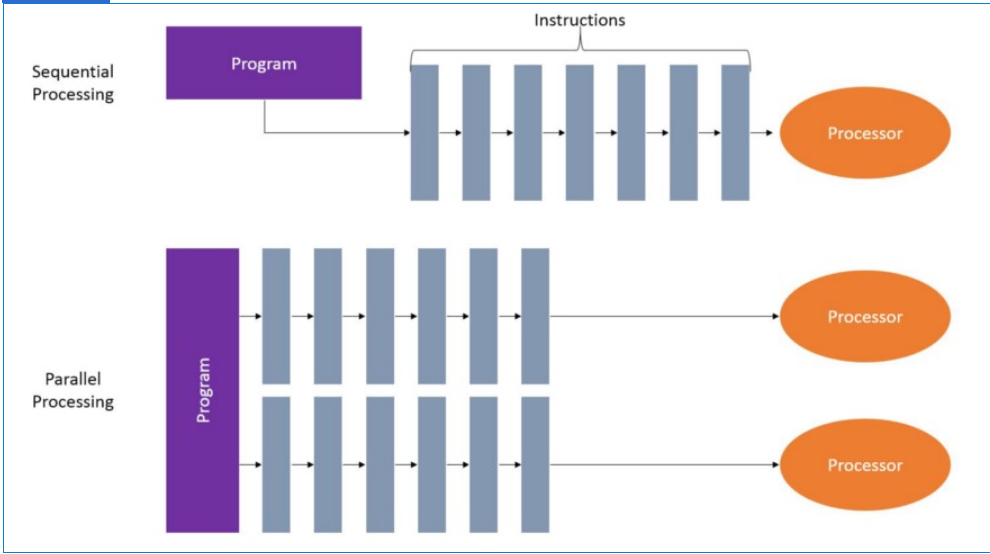
**<!--是否了解RLlib?Coach？-->**

分层的分布式强化学习库，非常完善但是代码很复杂

————————————————

#### **<!--Ray怎么做梯度并行运算的?-->**

在训练时我希望能完全的利用他们（与单纯的深度学习不同，训练强化学习智能体，CPU和GPU都很重要）。 下图是一个分布式计算的示例图。



off-policy的RL算法或者蒙特卡洛方法来说，如果能利用所有的核心，那么就可以24个线程同步采集训练数据，供优化算法进行优化，所以可以极大的提高训练的速度。 目前Ray仅支持Linux和MacOS，对python3.5-3.7支持较好。我们通过一个例子来看ray是怎样进行分布式计算的。

**<!--A3C中多线程如何更新梯度？-->**

从Actor收集数据之后进行统一更新

**<!--GA3C算法的queue如何实现？请简述-->**

不同Agent（Actor）收集到的数据发送给Trainer

**<!--强化学习的动作、状态以及奖励如何定义的，指标有哪些，包括状态和动作的维度是多少，那些算法效果比较好？、-->**

根据实际问题

指标：序列总奖励

TD3

D3QN

**<!--DQN的trick有哪些？-->**

目标网络

replay-buffer

**<!--PPO算法中的clip如何实现的？-->**

限制重要度采样系数

**<!--MADDPG如何解决离散action的？-->**

**<!--强化学习在机器人的局限性有哪些 ？-->**

动作空间离散

**<!--强化学习中如何解决高纬度输入输出问题？-->**

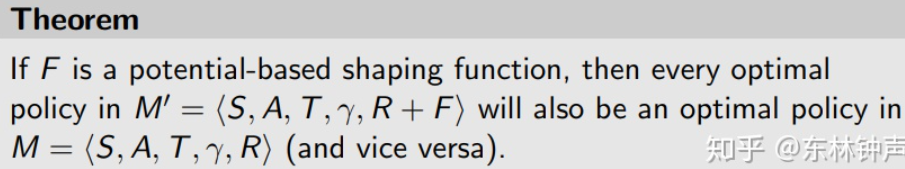
对输入空间进行归类降维

————————————————

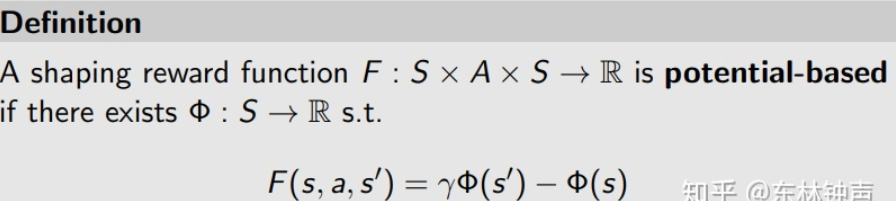
#### **<!--是否了解过奖励函数的设置(reward shaping)？-->**

****reward shaping****

这里先放结论

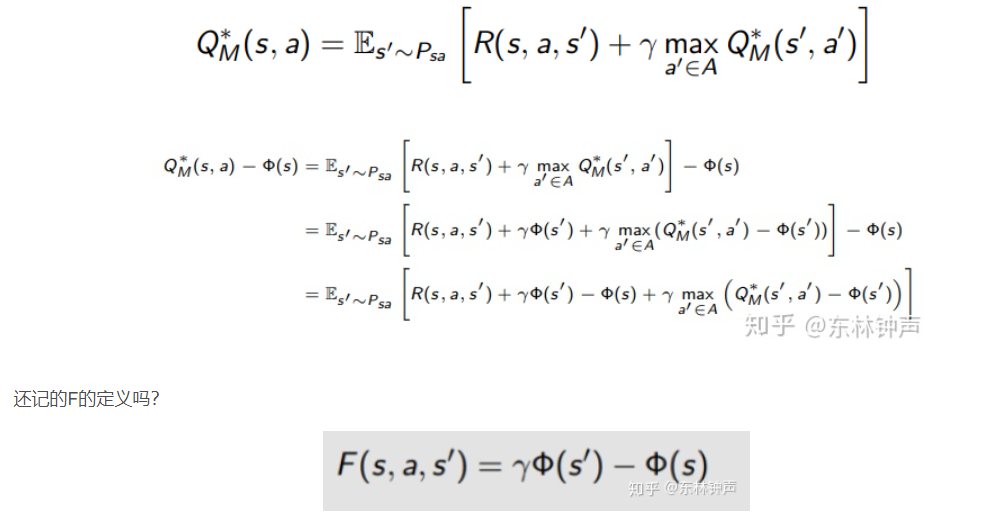


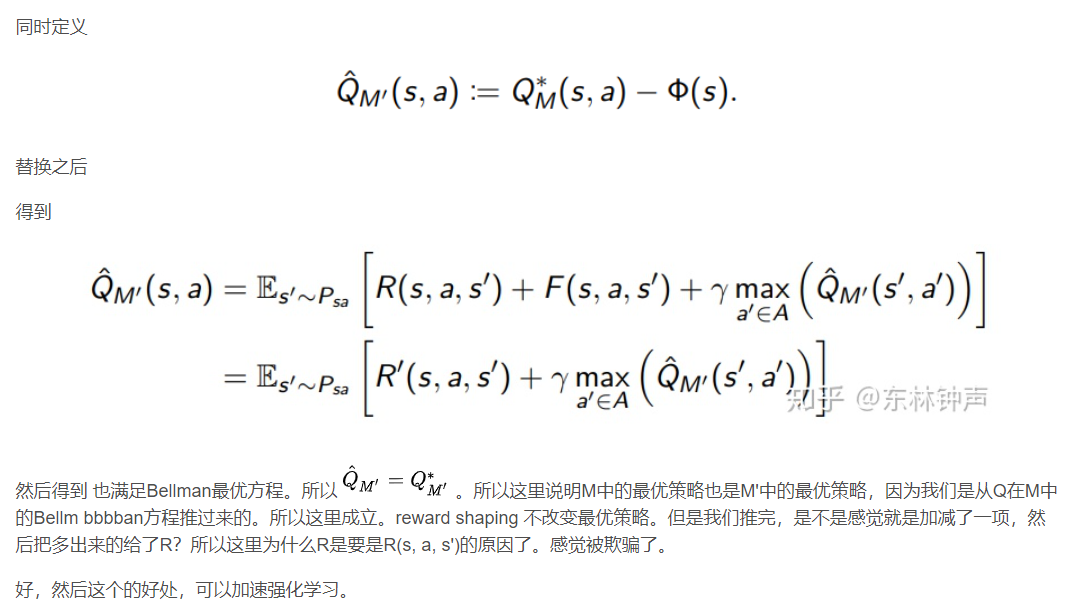
就是如果F是potential-based，那么改变之后的reward function= R + F重新构成的马尔科夫过程的最优控制还是不变，跟原来一样。



这个定义就很巧妙了，记住之前为什么要用s'的问题。开始推导。

推导也蛮简单，就是在Bellman最优方程两边同时减去phi(s)来把F凑出来。





**<!--基于值函数方法的算法有哪些？其损失函数是什么？（MSE）-->**

MSE，DQN系列方法

**<!--写出用第n步的值函数更新当前值函数的公式（1-step，2-step，n-step的意思）。当n的取值变大时，期望和方差分别变大、变小？-->**

偏差变小，方差变大

**<!--TD(λ)方法：当λ=0时实际上与哪种方法等价，λ=1呢？-->**

单步TD

蒙特卡洛方法

**<!--为什么Policy中输出的动作需要sample，而不是直接使用呢？-->**

增加探索

**<!--是否用某种DRL算法玩过Torcs游戏？具体怎么解决？-->**

状态：车辆的信息

含有两类动作，1.控制左右2.控制油门刹车

奖励就是偏离最佳行驶路线的程度

**<!--为什么连续动作环境下使用DDPG的表现还没有直接动作离散化后Q-learning表现好？-->**

离散化粒度只有6，Q-learning的探索效率远高于DDPG。 这说明在你的任务中，探索效率提升带来的收益超过了控制精度下降带来的损失。 如果离散化再精细一些，可能还会继续提升性能，直到上述收益与损失的关系发生逆转。 本来连续动作空间在探索效率上就是吃亏的，DDPG叠加噪声的方式有很大改进空间，比如parameter noise等等。

**<!--PPO算法中的损失函由那些组成？-->**

clip

**<!--你在强化学习模型调试中，有哪些调优技巧？-->**

加约束

多尝试多种方法（TD3，D3QN）

减少状态和动作维度

采用更简单的方法

注意探索利用的程度

奖励设置技巧：0，1奖励不如软奖励好，正奖励鼓励继续探索，负奖励鼓励尽快结束。

**<!--简述PPO、DPPO算法？-->**

DPPO是PPO的分布式版本（actor-learner）

DPPO加入了以下trick

1.RNN,更容易学习到时序信息

2.采用多步优势函数

3.归一化

DPPO是由worker本地计算梯度，然后由chief来进行统一更新

**<!--简述PER算法、HER算法？-->**

PER给每个经验增加了一个权重（TD error）

HER设置多个目标得到更多的数据

增加state和reward信息

**<!--离散action和连续action在处理上有什么相似和不同的地方？-->**

离散：神经网络最后一层就是动作的个数，输出的是动作的被选择可能性 discrete

连续：神经网络最后一层直接输出动作。gym.spaces.box

都输出shape[0]

但是随机RL算法和确定RL算法虽然Actor最后都输出一个action，但确定性方法是直接输出，而随机方法则是要把这个action转化成分布形式然后进行sample

————————————————