# 如何直观理解PPO算法?[理论篇]

[](https://www.zhihu.com/people/zhang-si-jun-52)

[张斯俊](https://www.zhihu.com/people/zhang-si-jun-52" \t "/home/admin/文档\\x/_blank)

愿成为一把梯子，助你跨过无数的坑。

### 先吹一波PPO

PPO是目前非常流行的增强学习算法，OpenAI把PPO作为目前的baseline算法，也就是说，OpenAI在做尝试的时候，首选PPO。可想而知，PPO可能不是目前最强的，但可能是目前来说适用性最广的一种算法。

PPO是基于AC架构的，也就是说，PPO也有两个网络，分别是Actor和Critic，这是因为AC架构有一个好处。这个好处就是解决了连续动作空间的问题。

由于CartPole是离散动作的游戏，所以AC架构的这个好处并没有体现出来，如果需要处理连续动作空间问题，那么应该怎么做呢？从概率的分布说起。

高能预警：PPO涉及知识点较多，尤其是重要性抽样部分会比较烧脑。本人会尽量说人话，让你直观理解。PPO基于AC框架，如果对AC框架有疑问。可以看看本专栏之前的文章：

[张斯俊：理解Actor-Critic的关键是什么？​](https://zhuanlan.zhihu.com/p/110998399" \t "/home/admin/文档\\x/_blank)

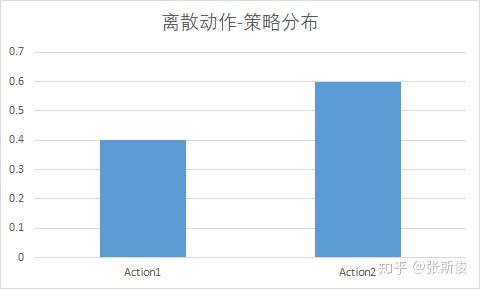
### 正态分布

首先，要想办法处理连续动作的输出问题。

先说离散动作。离散动作就像一个个的按钮，按一个按钮就能智能体就做一个动作。就像在CartPole游戏里的智能体，只有0,1两个动作分别代表向左走，向右走。

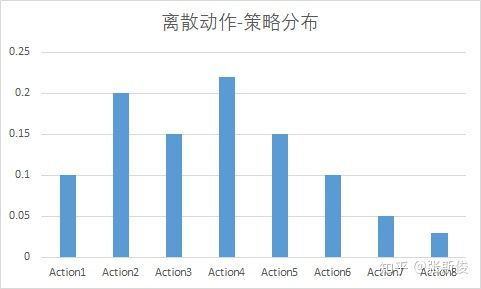
那什么是连续动作呢。这就相当于这些按钮不但有开关的概念，而且还有力度大小的概念。就像开车，不但是前进后退转弯，并且要控制油门踩多深，刹车踩多少的，转弯时候转向转多少的问题。

于是问题来了，在离散动作空间的问题中，最终输出的策略呈现这样的形式。



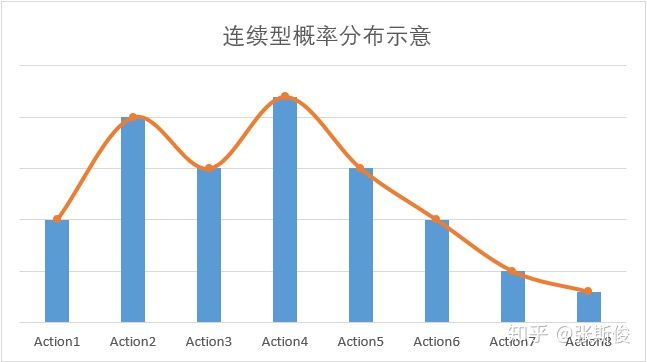
假设动作空间有只有action1 和 action2，有40%的概率选择action1 ，60%概率选择action2。也就是说在这个状态下的策略分布: pi = [0.4, 0.6]。

同理，假如有多个动作，同样可以这样表示。



但连续型动作怎么表示呢，还记得之前学习DQN的时候，DQN的函数就像一张布覆盖到了Qtable上吗？其实连续型动作的理解也是一样的。

可以理解，把连续型概率切成很多很多份。每一份的数值，就代表该动作的概率。

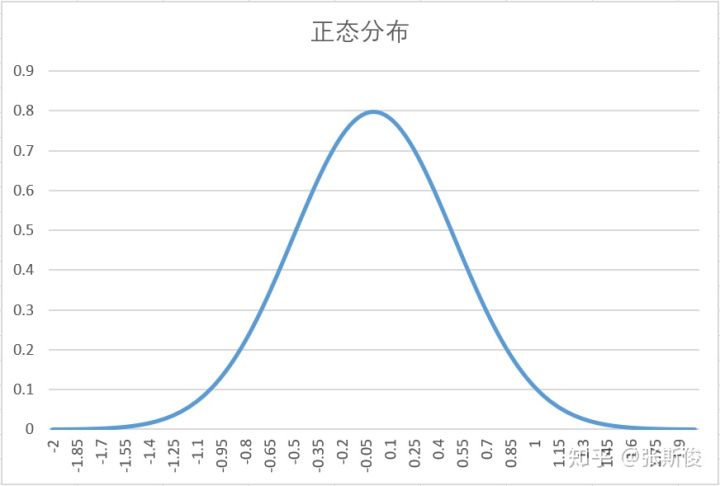


在连续型，不再用数组表示，而是用函数表示。例如，策略分布函数 ： P = Pi（action）代表在策略Pi下，选择某个action的概率P。

但这就有个问题，用神经网络预测输出的策略是一个固定的shape，而不是连续的。那又什么办法可以表示连续型的概率呢？

先假定策略分布函数服从一个特殊的分布，这个特殊的分布可以用一两个参数表示它的图像。

正态分布就是这样一个分布，他只需要两个参数，就可以表示了。



正态分布长得就是这个形状，中间高，两边低。他的形状由两个参数表示sigma，mu。

* mu表示平均数，也就是整个正态分布的中轴线。mu的变化，表示整个图像向左右移动。
* sigma表示方差，当sigma越大，图像越扁平；sigma约小，图像越突出。而最大值所在的位置，就是中轴线。

神经网络可以直接输出mu和sigma，就能获得整个策略的概率密度函数了。

现在已经有概率密度函数，那么当要按概率选出一个动作时，就只需要按照整个密度函数抽样出来就可以了。

### AC的问题

现在，解决了AC的一个问题，证明AC的框架既能处理连续状态空间问题，也能处理连续动作空间问题。

但AC还有一个问题。AC产生的数据，只能进行1次更新，更新完就只能丢掉，等待下一次跑游戏的数据。在数据就是生命的时代，这可是天大的浪费呀，尤其在强化学习中，数据更是弥足珍贵呀。

那为什么只能用一次呢，像DQN也可以用经验回放，把以前的数据存起来，更新之后用, AC为什么就不行呢？

先从策略说起。

为了方便讨论，先理清楚两个概念：

1- **行为策略**——不是当前策略，用于**产出数据**

2- **目标策略**——会更新的策略，是需要**被优化的策略**

如果两个策略是**同一个策略**，那么称为**On Policy**，**在线策略**。如果**不是同一个策略**，那么**Off Policy**，**离线策略**。

这样说有点难以理解，举个例子：

如果在智能体和环境进行互动时产生的数据打上一个标记。标记这是第几版本的策略产生的数据,例如 1， 2... 10

现在的智能体用的策略10，需要更新到11。如果算法只能用10版本的产生的数据来更新，那么这个就是在线策略；如果算法允许用其它版本的数据来更新，那么就是离线策略。

例如PG，就是一个在线策略。因为PG用于产生数据的策略（行为策略），和需要更新的策略（目标策略）是一致。

而DQN则是一个离线策略。会让智能体在环境互动一定次数，获得数据。用这些数据优化策略后，继续跑新的数据。但老版本的数据仍然是可以用的。也就是说，产生数据的策略，和要更新的目标策略不是同一个策略。所以DQN是一个离线策略。

但为什么PG和AC中的Actor更新，就不能像DQN一样，把数据存起来，更新多次呢？

答案是在一定条件下，能，PPO做的工作就是这个。在了解在什么条件下可以的时候，需要先了解一下，为什么不能。

为了给大家直观的理解，来看这么一个简化的例子。

来看这么一个简化的例子：



假设，已知在同一个环境下，有两个动作可以选择。现在两个策略，分别是P和B：

P: [0.5,0.5] B: [0.1,0.9]

现在按照两个策略，进行采样；也就是分别按照这两个策略，以S状态下出发，与环境进行10次互动。获得如图数据。那么，可以用B策略下获得的数据，更新P吗？

答案是不行，可以回顾一下PG算法，PG算法会按照TD-error作为权重，更新策略。权重越大，更新幅度越大；权重越小，更新幅度越小。

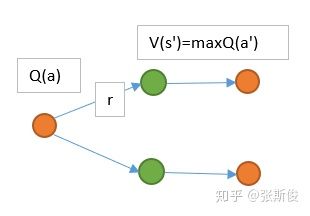
但大家可以从如下示意图看到，如果用行动策略B[0.1,0.9]产出的数据，对目标策略P进行更新，动作1会被更新1次，而动作2会更新9次。虽然动作A的TD-error比较大，但由于动作2更新的次数更多，最终动作2的概率会比动作1的要大。



这自然不是期望看到的更新结果，因为动作1的TD-error比动作2要大，希望选择概率动作1的能更多呀。

从这个例子，大家可以大致明白，为什么在更新策略的时候，不能用其他策略产生的数据了。

但为什么DQN可以多次重复使用数据呢？



可以从两个角度看： 1. 更新Q值，和策略无关。 在同一个动作出发，可能会通往不同的state，但其中的概率是由环境所决定的，而不是的策略所决定的。所以产生的数据和策略并没有关系。

1. 在DQN的更新中，是有"目标"的。 虽然目标比较飘忽，但每次更新，其实都是尽量向目标靠近。无论更新多少次，最终都会在目标附近徘徊。但PG算法，更新是不断远离原来的策略分布的，所以远离多少，远离的次数比例，都必须把握好。

### Important-sampling

那么，PPO是怎样做到离线更新策略的呢？答案是Important-sampling，重要性采样技术。

IS：如果想用策略B抽样出来的数据，来更新策略P也不是不可以。但要把td-error乘以一个重要性权重（IW：importance weight）。

重要性权重：IW = P（a）/ B（a）

应用在PPO，就是目标策略出现动作a的概率除以行为策略出现a的概率。

回到之前的例子，可以计算出，每个动作的重要性权重，

P: [0.5,0.5] B: [0.1,0.9]



以a1为例，计算重要性权重

IW = P / B = ( 1 / 0.5 ) / ( 1 / 0.1 ) = 5

把重要性权重乘以td-error，发现，a1的td-error大幅提升，而a2的td-error减少了。现在即使用P策略: [0.5,0.5]进行更新，a1提升的概率也会比a2的更多。

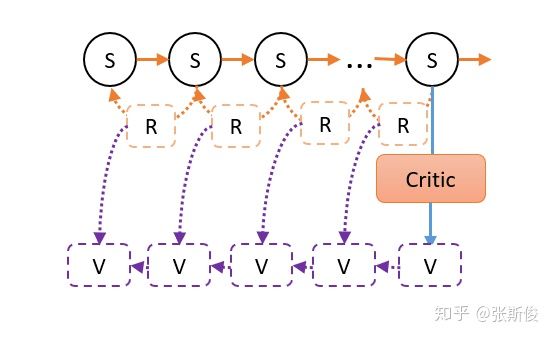
PPO应用了importance sampling，使得用行为策略获取的数据，能够更新目标策略，把AC从在线策略，变成离线策略。

### N步更新

之前已经做出了一些铺垫：包括连续型动作的输出和Importance sampling。

在示例代码中，还使用了N步更新的技术。把之前的TD叫做TD(0)，而N步更新为TD(n)。可以看成TD(0)其实是TD(n)的一种特殊情况。

不过如果大家对之前的算法已经比较熟悉的话，其实很容易明白。



如图，实际上只需要计算最后的V(s')，根据这个估算的V(s'), 反推经过的所有state的V值。这个其实和PG估算G的过程是一样的，只不过并不需要走到最后，而是中途截断，用网络估算。

这种中途截断的，用网络估计的方式相当常见，最著名的围棋AI Alpha Zero，也有用到类似的技术。只不过Alpha Zero截断的不是马可洛夫链，而是蒙地卡罗搜索树而已。

### 小结

现在先来小结一下学到的：

1. 可以用AC来解决连续型控制问题。方法是输入mu和sigma，构造一个正态分布来表示策略；
2. PPO延展了TD(0)，变成TD(N)的N步更新；
3. AC是一个在线算法，但为了增加AC的效率，希望把它变成一个离线策略，这样就可以多次使用数据了。为了解决这个问题，PPO使用了重要性采样。

但实际上，P策略和B策略差异并不能太大，为了能处理这个问题，有两个做法，PPO1 和 PPO2 。

相对来说，PPO2更好理解，做法也更简单，性能并不比PPO1差，目前主流的方式也是用PPO2。所以会以PPO2进行说明。而PPO2将会结合实例代码，在实战篇一同讲解。