Advantage Actor-Critic (A2C)

在上一篇文章中我们所提及的Basic Actor Critic算法里，Actor网络就是使用一般的Policy Gradient，而Critic网络使用基于值函数近似的Sarsa算法来估计策略梯度表达式里面的 。在基础的AC算法中，我们通过policy network的参数更新式子也可以发现， 是步长中的一个项，即如果 很大，那么Actor网络就会提高选择该动作的概率。但是，我们在实际情况下往往更加关心 的相对值，如果当前状态s下，动作的action value是1000，这似乎很大，但是如果其他动作的action value是2000，那么实际上也并不是一个很好的动作。反之，如果一个action他的action value是0，但是其他的动作的action value是-1，那么依然可以认为该动作是一个不错的动作。因此，在决定policy network网络参数更新步长的时候，我们将原本的 替换成一个新的函数：，我们称之为“优势函数”。

以上就是A2C中对优势函数的直观理解。下面我们要回答几个问题：

1. 为什么叫做“优势函数”?
2. 直接在策略梯度中引入优势函数是否正确 ?
3. A2C算法如何实现 ?

要回答第一个问题，我们首先需要回顾一下state value的含义： 表达的是在当前状态 下，各种动作的action value的平均值。而 则表达的是在状态 下，某一个动作 的 action value。那么优势函数 就反映了动作 的action value跟平均水平之间的差距。如果 的值比较大，那么说明动作 在状态 下更有优势。

下面，我们要思考：为什么我们可以在策略梯度中引入策略梯度 ? 我们首先来回顾最基本的策略梯度：

因此，我们将优势函数 替换掉上式中的 。值得注意的是：因为是在期望的表达式里面，所以这里状态和动作都用大写字母，表示随机变量。因此，引入优势函数后的策略梯度如下：

我们可以证明的是：引入了优势函数之后，策略梯度的值并不会改变，即引入了 之后不会改变期望的值（即偏差不变）。我们下面证明一下：

我们发现对 求和正好是常数1，对常数求导为0。因此引入baseline并不改变原本的策略梯度。而其实通过证明还可以发现，引入了baseline 之后，可以使得策略梯度中期望里面的部分的方差比basic actor-critic算法的要小。

在了解了优势函数的正确性之后，我们来关心一下A2C算法的工程性问题：即如何实现A2C ? 有了之前几篇文章的基础，我们一看到梯度中带有期望，很显然反应过来需要使用随机梯度上升方法。因此，policy network中网络参数的更新就可以表示为：

然而，我们发现，这里面既有action value，又有state value，这岂不是意味着我们需要两个网络分别估计这两个值 ? 这显然比较麻烦，因此，我们利用Bellman公式和TD算法的思想，将 写成 ，这样我们就可以只用一个critic网络来估计state value。

那么，我们现在就可以来介绍一下A2C的具体流程：首先我们会初始化actor网络 ，以及critic网络 。在一个episode里面，对于每一个step 所对应的状态 下，我们首先根据actor网络 执行动作 ，得到奖励 和下一个状态的观测 。下面我们需要计算优势函数：

然后率先进行critc网络的参数更新：

接下来，对actor网络进行参数更新：