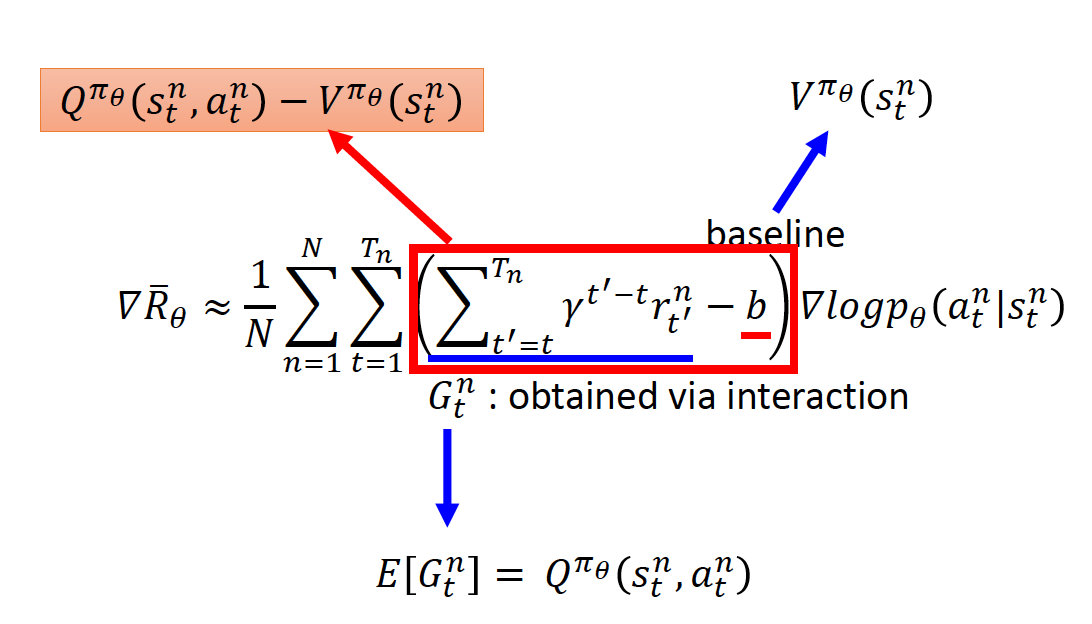
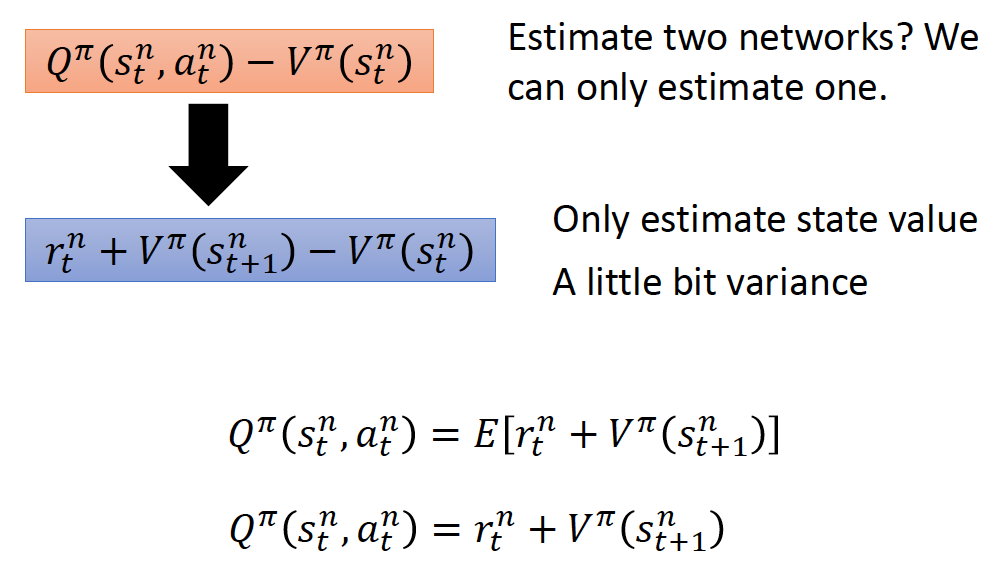
Actor-critic

参数更新公式：

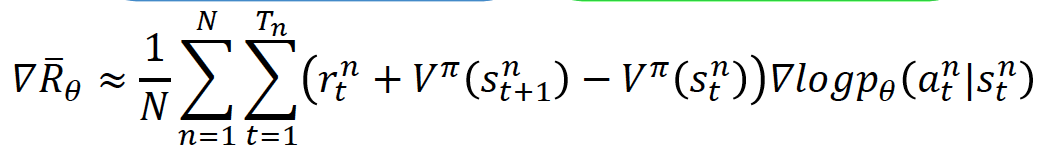


橙色部分表示当在状态s时，采取动作a后增加的value值，若增加了，为正，表明此时选择action的概率应该增加；若减少了，为负，表明此时选择action的概率应该降低。

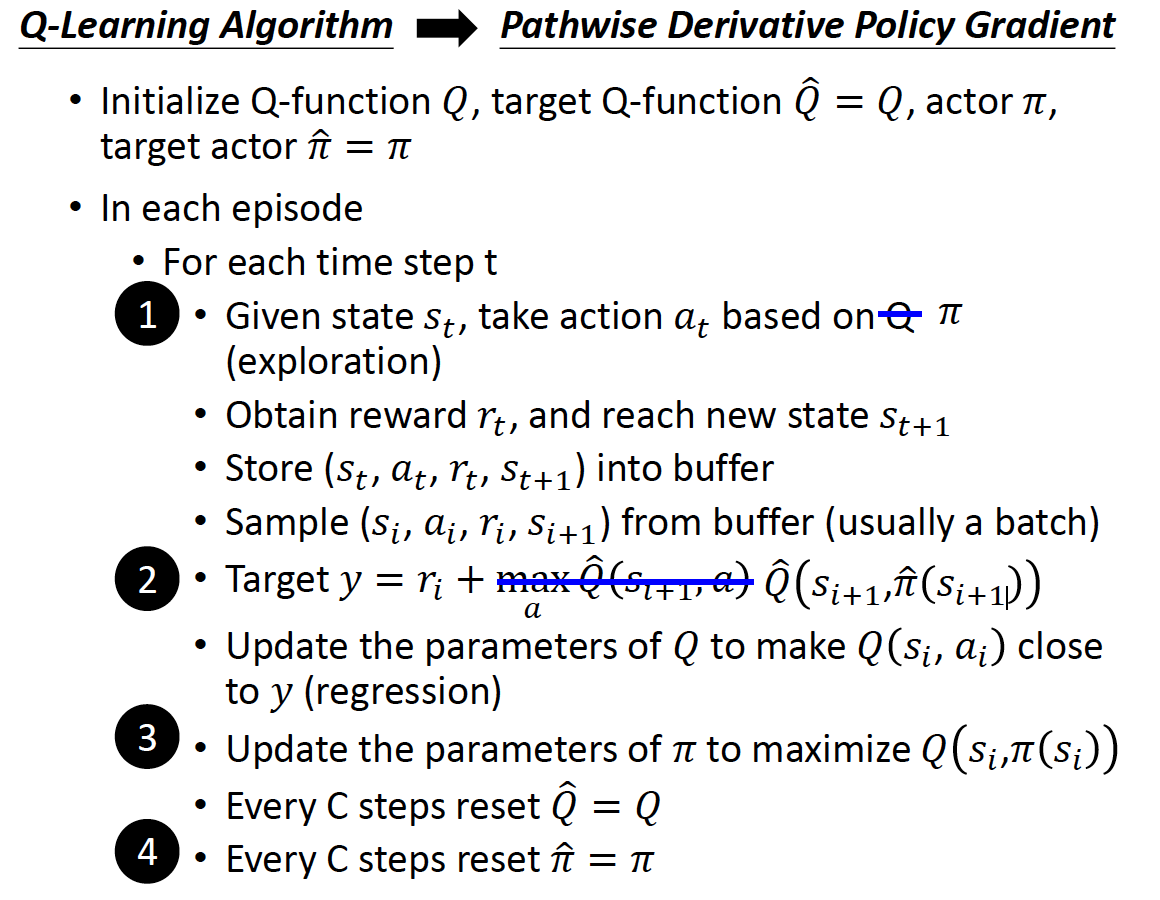
但是这就面临需要估计两个网络？可以只估计一个，只估计状态值。如下：



最终actor-critic对应的公式为：

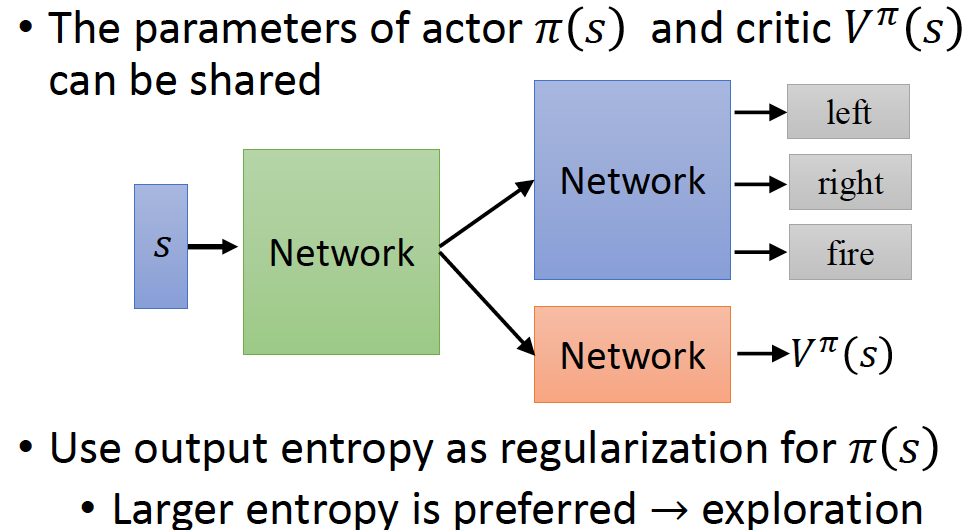


算法训练步骤：



一些训练小tip：

1. actor和critic的部分网络可以共享

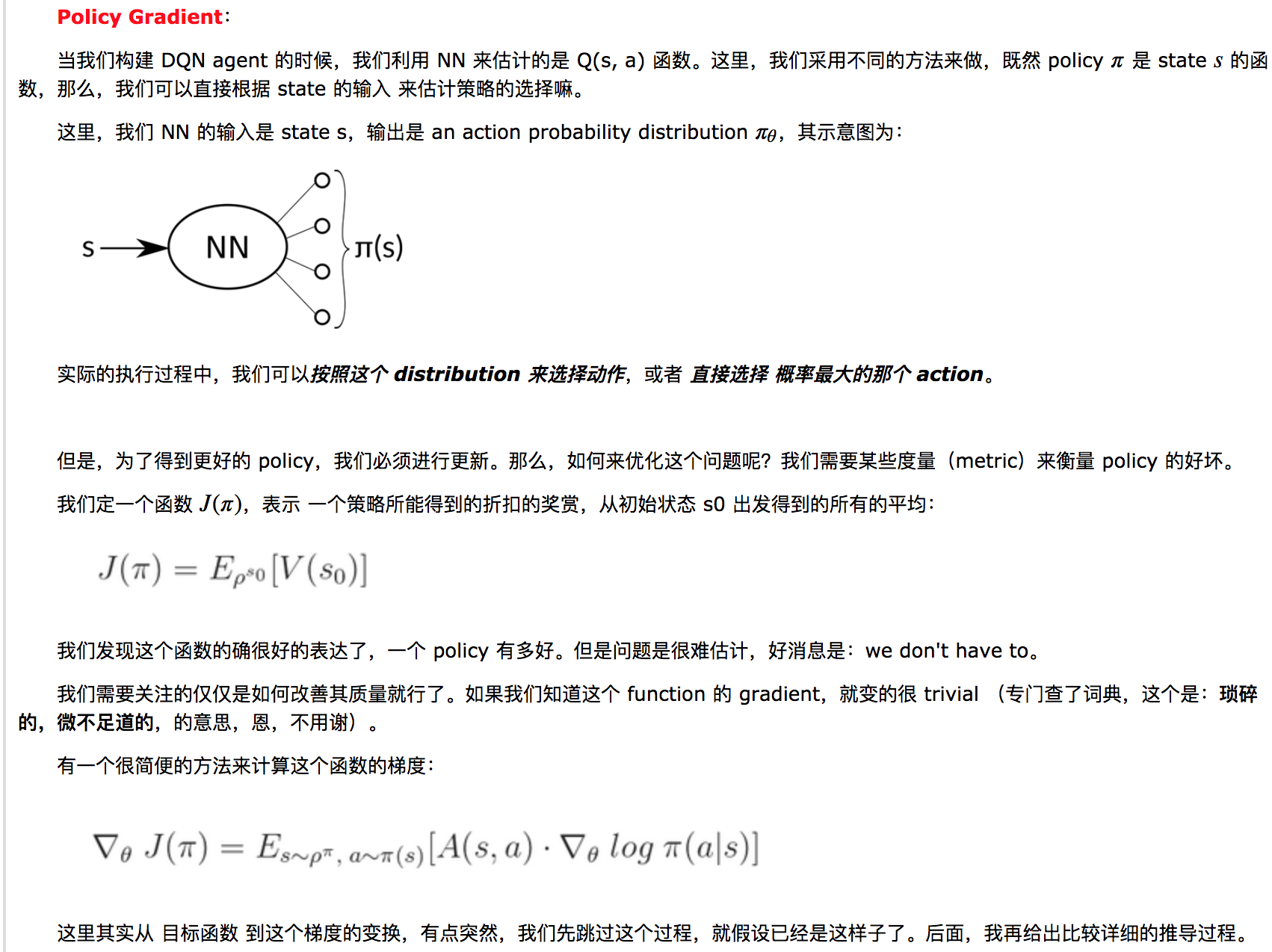


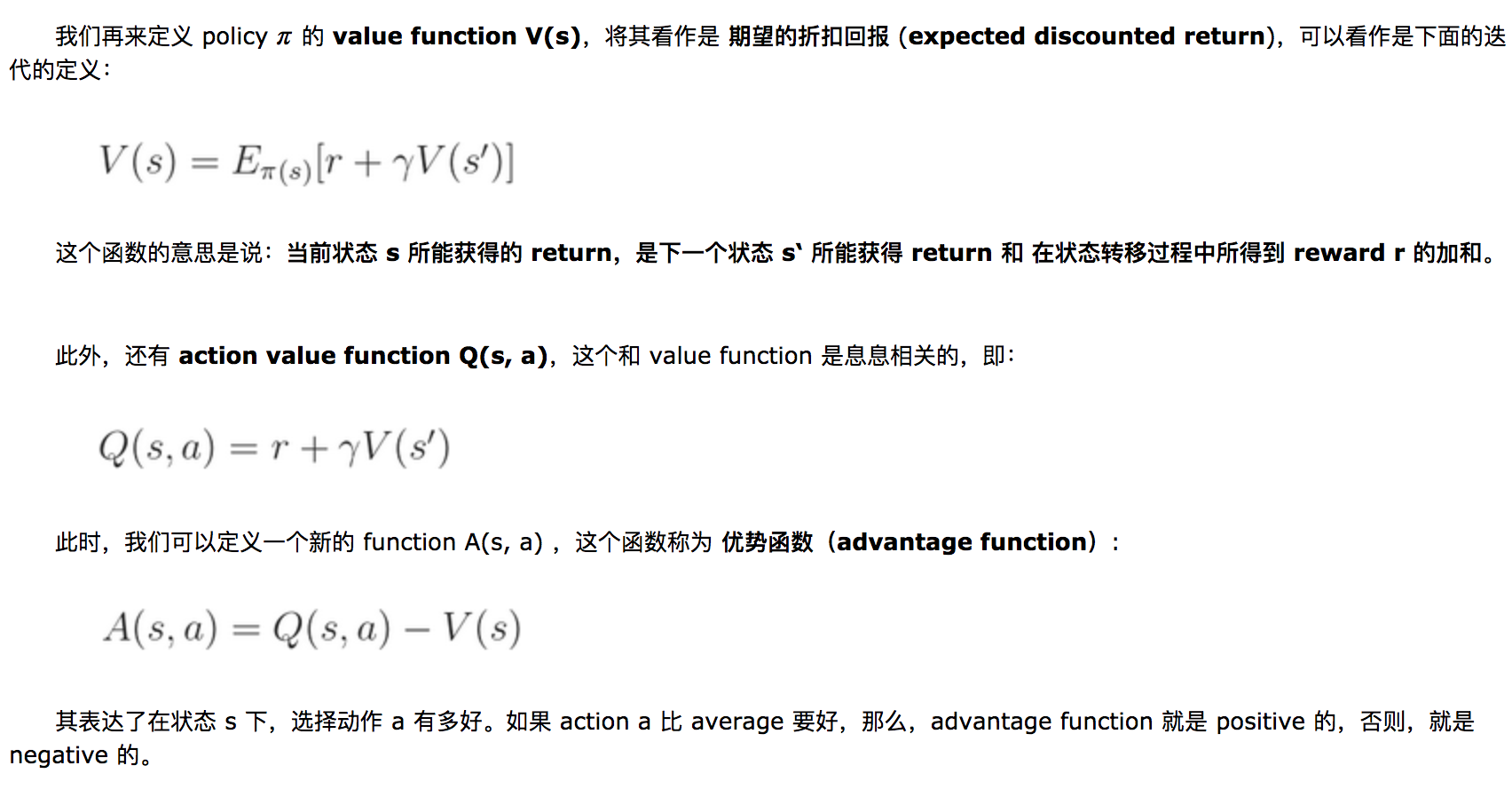


1. 网络结构

两部分网络：actor、critic

actor 执行动作，然后 critic 评价这个动作的选择是好是坏。





整体训练过程：

1. 通过actor网络，在state下，通过此概率分布随机选择一个action；
2. 函数F：输入action，通过环境的交互得到下一个状态s1和reward。
3. 通过critic网络，首先通过s1和critic得到v(s1)，然后通过s，v(s1)和r进入critic网络 得到td\_error==r + gamma\*v(s1)–v(s)。

gradient = grad[r + gamma \* V(s\_) - V(s)]

（4） 通过actor网络，通过s得到acts\_prob, 接着通过输入td\_error，acts\_prob，action来优化actor。

目标，和policy gradient一样：

true\_gradient = grad[logPi(s,a) \* td\_error]

log\_prob = tf.log(self.acts\_prob[0, self.a])  
self.exp\_v = tf.reduce\_mean(log\_prob \* self.td\_error) 最大化