基于值函数近似的Sarsa和Q-Learning

在之前我们所介绍的Sarsa和Q-Learning算法都是基于table的，即Sarsa算法中有一个表格存储某一个策略 下不同state 和 action 所对应的 ，Q-Learning中有一个表格用于存储不同 state 和 action 所对应的最优action value 。然而，在实际应用中，当状态和动作的数目很大时，他们排列组合所得到的state-action pair就会非常多，这极大地限制了基于table的Sarsa和Q-Learning算法的应用。因此，我们在本节中希望通过一个函数来表达 或 。这样一来我们就可以只存储函数的参数，就可以通过这个函数来计算各种state-action pair所对应的action value或optimal action value，他们分别对应的就是值函数近似的Sarsa算法和Q-Learning。因此，希望在本文结束之后我们能够回答下面的几个问题：

1. 如何设置目标函数?
2. 如何更新函数的参数使得函数能够更好地拟合 或 ?

首先，我们先来考虑值函数近似的Sarsa算法。我们将函数的估计值记为：。其中 是函数的参数。那么我们肯定是希望我们的函数不仅仅只在某些state-action pair上的估计误差比较小，而是对于所有的state-action pair的估计值总体上都比较小。因此，可以用期望来定义目标函数：

我们现在解决了第一个问题，下面就是如何更新函数的参数 ? 很显然我们想让目标函数最小化，因此应该使用梯度下降法。如下所示：

因为在梯度下降法中， 只是代表了一个学习的步长，因此我们可以直接用一个字母 代替，没有问题。然而，这个式子中含有期望，因此我们将采用随机梯度下降 (Stochastic gradient descendent, SGD)，即用一个样本值来近似期望，如下：

通过SGD，我们就把期望这个烦人的东西给“去掉”了，但是上面这个函数参数更新的式子在实际中还是无法使用。原因是在更新的时候需要知道在状态 ，策略下，动作 真实的action value。这该怎么办呢? 回顾我们之前所学，有两个办法来解决这一问题：

1. 蒙特卡洛方法：我们首先从状态 出发，采取动作 ，并继续和环境交互得到一个episode：(。然后我们计算这一整条episode的discounted return 作为 的估计值。
2. TD-Learning：我们还可以使用 来近似 。

我们更加喜欢使用TD-Learning，因为不需要等到整个episode结束，边交互边训练。因此我们就可以导出基于值函数近似的Sarsa算法：

类似地，我们也可以导出基于值函数的Q-Learning算法函数参数更新的规则：

实际上，这个就是DQN的雏形了，DQN利用一个神经网络，输入是当前的状态 ，输出各个action对应的action value，Agent选择当前具有最大action value的动作执行，得到 以及状态转移 ，再将 输入至网络，取出网络输出的最大值 。因此DQN的loss函数就是 与 的均方误差。

只不过，在实际的应用中，仅仅使用上面的update rule来实现DQN效果可能并没有那么好，因此我们在真正要实现DQN的时候，会用到很多额外的技术：如经验回放、Target网络，Double DQN, Dueling DQN等等。我们在本文中给出一个具有target网络和eval网络的DQN更新公式：

其中，每次更新的是评估网络 (Eval Network)，target网络只有在固定的更新时间到来时，才去复制评估网络中的参数。