PPO算法：

这个算法，是openai的默认算法.所以特别叼.参考https://zhuanlan.zhihu.com/p/86306874
自己写过程，尽量吧所有细节都写上.

$\tau = \{s_1, a_1, s_2, a_2, \cdots\}$

θ 取定后 $\rightarrow \pi$ 取定. $\rightarrow$ 给定一个s1，就会得到一个序列 τ

所以说序列跟 θ 有关.给定 θ 后才能生成这个序列，利用bg算法求解更新theta，所以是on-policy的.

obj-func:

$$R(\tau) \triangleq \sum_{t=1}^{T} r_t$$

$$\bar{R}_\theta = \sum R(\tau) P_\theta(\tau)$$

$$P_\theta(\tau) \triangleq P(s_1) P_\theta(a_1|s_1) P_\theta(s_2|a_1,s_1) \cdots$$

注意的一点是，虽然每一个θ取定后pi取定，这个pi对于给定的st，返回的at不是一个确定的值，而是一个action的分布.这样我们学习θ，让他改变这些个分布，从而进行学习.

下面我们求 $\bar{R}_\theta$ 的梯度.

$$\nabla \bar{R}_\theta = \sum R(\tau) \nabla p_\theta(\tau)$$

$$= \sum R(\tau) p_\theta(\tau) \nabla (\log p_\theta(\tau))$$

$$= E_{\tau \sim p_\theta(\tau)} R(\tau) \nabla (\log p_\theta(\tau))$$

$$\text{以采样} = \frac{1}{N} \sum_{n=1}^{N} R(\tau^n) \nabla \log p_\theta(\tau^n)$$

重采样法来 改进为ppo 的off-learn算法

$$\nabla \bar{R}_\theta = E_{\tau \sim p_\theta(\tau)} [R(\tau) \nabla \log p_\theta(\tau)]$$

$$\nabla \bar{R}_\theta = E_{\tau \sim P_{\theta'}(\tau)} \left[\frac{P_\theta(\tau)}{P_{\theta'}(\tau)} R_{\theta'}(\tau) \nabla \log P_\theta(\tau) \right]$$

$$= E_{\tau \sim P_{\theta'}(\tau)} \frac{\nabla P_\theta(\tau)}{P_{\theta'}(\tau)} A_{\theta'}(\tau)$$

上面式子同事对θ等式两边做积分：

$$\bar{R}_\theta = E_{\tau \sim P_{\theta'}(\tau)} \frac{P_\theta(\tau)}{P_{\theta'}(\tau)} A_{\theta'}(\tau)$$

DQN:

为什么使用target net.
李宏毅老师说的是如果你的目标网络也是浮动的话,收敛会波动,收敛速度会变慢.所以使用target net来 每隔多少个周期再更新权重会保证收敛速度.

off-policy: 里面的记忆库用的不是pi来生成的用来,做pi的训练也没关系.因为学的是reward.还是有学习价值的.

李宏毅的double dqn也讲得非常细,细节都到位了.

dqn的缺陷是没法计算continuous action.

$$a = \arg\max_a Q(s, a)$$

这个方程对于连续情况没法解.

解决方法:
1.抽样一堆action带入,选择最优的,也就是离散化
2.当作最优问题做梯度下降,解a.---计算量挺大,也不一定收敛.
3.特别再设计一个网络来解这个方程.

最好的学习方法是,讲给自己听,或者讲给别人听.然后看细节是否都能过去.一个数学家提出的学习方法.