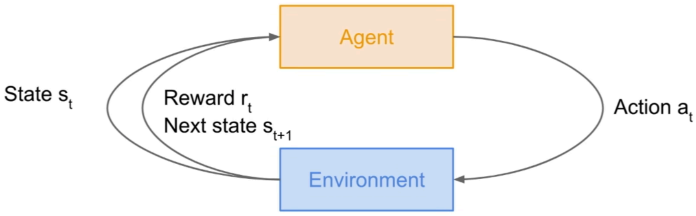
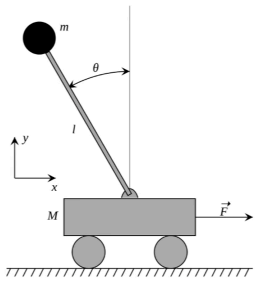
cs231n-7: Deep Reinforcement Learning

1. 以终于到增强学习了！
    1. Introduction
        1. Agent and Environment
            1. 

        2. Cart-Pole problem (or other problem like Robot Locomotion, Artari Games, Go(AlphaGo))
            1. Object: Balance a pole on top of a movable cart
            2. State: angle, angular speed, position, horizontal velocity
            3. Action: horizontal force applied on the cart
            4. Reward: 1 at each time step if the pole is upright, 0 otherwise
            5. 
2. Markov Decision Process
    1. Mathematical Formulation of the RL problem
    2. Markov Property: Current State completely characterizes the state of the world.
        1. (S, A, R, P, y)
            1. S: Set of possible sates
            2. A: Set of possible actions
            3. R: distribution of reward given (state, action) pair
            4. P: transition probability i.e. distribution over next state given (state, action) pair
            5. y: discount factor
        2. **Objective: find a policy pi maximizes cumulative discounted reward**
            1. $$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi\right] \text{ with } s_0 \sim p(s_0), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)$$

- At time step t=0, environment samples initial state $s_0 \sim p(s_0)$
- Then, for t=0 until done:
    - Agent selects action a_t
    - Environment samples reward $r_t \sim R(. | s_t, a_t)$
    - Environment samples next state $s_{t+1} \sim P(. | s_t, a_t)$
    - Agent receives reward r_t and next state s_{t+1}

2.
- A policy **π** is a function from S to A that specifies what action to take in each state
- **Objective**: find policy **π*** that maximizes cumulative discounted reward: $\sum_{t>0} \gamma^t r_t$

3. **How to handle the randomness**
    1. Value Functions and Q-Value Function
        1. How good is a state
            1. The value function at state s, is the expected cumulative reward from following the policy from state s
            2. $$V^{\pi}(s) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, \pi\right]$$
        2. How good is a state-action pair
            1. The Q-value function at state s and action a, is the expected cumulative reward from taking action a in state s and then following the policy
            2. $$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$
        3. 反正…都是给定条件下，Following pi policy条件下累计报酬的期望值
    2. 若Q*(s, a) = max_pi Q(s, a)，则Q* 满足Bellman equation
        1. $$Q^*(s,a) = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s', a') | s, a\right]$$
        2. 如果下一步的最优值已知，那么最优值可以变成当前reward加上下一步最优值乘上衰减因子
        3. 因此可以迭代地求出Q*了，最后Qi会收敛到Q*
            1. $$Q_{i+1}(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q_i(s', a') | s, a\right]$$
        4. 但是上述式子虽然可以求解，但每一步都要计算Qi，该计算量很大，并且不知道何时收敛，所以应该要近似求解
        5. 既然这个函数很难收敛，那么神经网络就该上场了

3. Q-Learning
    1. Use a function approximator to estimate the action-value function

Remember: want to find a Q-function that satisfies the Bellman Equation:

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') | s,a\right]$$

Forward Pass

Loss function: $L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot)}\left[(y_i - Q(s,a;\theta_i))^2\right]$

where $y_i = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) | s,a\right]$

Iteratively try to make the Q-value close to the target value (y_i) it should have, if Q-function corresponds to optimal Q* (and optimal policy π*)

Backward Pass

Gradient update (with respect to Q-function parameters θ):

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot); s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)\right]$$

1.
    1. maxQ
        1. 比如Q是一个table，每一项是对某个参数ø的值
        2. 当前环境下，我们可以找到最佳的s和a，当作max值
2. Atari Games
    1. Current State: 4帧，包含当前值和部分history
    2. 经过一个神经网络
        1. conv，FC
        2. 最后输出FC4，也就是每一个Action对应一个scalar值，一般的话，4-18都可能
    3. 经验重放
        1. 问题
            1. 样本间是相关的：导致不完全学习
            2. 当前Q网络参数的决定了下一个训练样本
                1. 比如最佳操作是向左，那么训练样本中向左的样本会占据上风
                2. 导致bad反馈循环
        2. 方法
            1. 随着游戏事件的进行，持续更新一个重放记忆表
            2. 从重访记忆表中随机选择minibatch来训练网络
3. 算法中的一些要点和核心过程
    1. 选择一个at
        1. 以较小的概率随机选择一个方向
        2. 否则就选择使得当前Q最大的at
    2. 执行at，得到reward rt和新的图片x+1
        1. 得到以后把信息放进replay memory表中
    3. 在重放表中sample一个minibatch，进行训练
4. 但是Q-Learning太复杂了
    1. 可能有high dimensional 的state
    2. 但是policy是很简单的，我们是否能够直接学习policy呢?

4. Reinforce algorithm
    1. Policy Gradient
        1. 定义一个关于θ的policy

Mathematically, we can write:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)} [r(\tau)] = \int_\tau r(\tau) p(\tau;\theta) \mathrm{d}\tau$$

Where r(τ) is the reward of a trajectory $\tau = (s_0, a_0, r_0, s_1, \ldots)$

        1.
        2. 目标依然是最大化reward
    2. 梯度如下：

$$\nabla_\theta J(\theta) = \int_\tau \left(r(\tau) \nabla_\theta \log p(\tau;\theta) \right) p(\tau;\theta) \mathrm{d}\tau = \mathbb{E}_{\tau \sim p(\tau;\theta)} [r(\tau) \nabla_\theta \log p(\tau;\theta)]$$

        1.
    3. 最后我们这样写出p(t|θ)，会发现只跟pi相关了

We have: $p(\tau;\theta) = \prod_{t \geq 0} p(s_{t+1}|s_t, a_t) \pi_\theta(a_t|s_t)$

Thus: $\log p(\tau;\theta) = \sum_{t \geq 0} \log p(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t)$

And when differentiating: $\nabla_\theta \log p(\tau;\theta) = \sum_{t \geq 0} \nabla_\theta \log \pi_\theta(a_t|s_t)$ Doesn't depend on transition probabilities!

Therefore when sampling a trajectory τ, we can estimate J(θ) with

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

        1.
        2. 但是这里我们的方差很大，需要太多次的取样了
    4. Variance Reduction（其中一种综合的方法，包含了cumulative, discount, baseline）

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} \gamma^{t'-t} r_{t'} - b(s_t) \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

        1.
        2. how to choose baseline：可以直接是value function
2. Actor-Critic Learning
    1. actor
        1. the policy
        2. decide which action to take
    2. critic
        1. Q-learning
        2. tell how good is the action
        3. to alleviate the task of the critic, we should use values of (state, action) generated by the policy
    3. other things
        1. can also incorporate Q-learning tricks like experience replay
        2. define A = Q - V
            1. 在网络中同时优化Q和V的参数，使得A最大
3. REINFORCE in action: Recurrent Attention Model (RAM)

        1. 目标：图片分类
        2. 问题：仅仅看到一系列图片的局部（glimpse），来判断是什么数字
            1. action是下一次看那个部分，即坐标
            2. state是当前看到的图
            3. 用reinforcement learning解决
        3. 优点
            1. 可以帮助我们解决机器视觉里面计算量过大的问题（只需要部分图片了）
            2. 去掉不相关的图片区域
        4. 用一个RNN
            1. 看五个地方（可以更多或更少glimpse），怎么看下一个地方循环决定
            2. 输出一个数字
        5. 也可以用来图片注释
5. AlphaGo
    1. 监督学习＋增强学习
    2. AlphaGo首先根据一些专家的棋局学习了一个policy神经网络
    3. 结合value网络＋蒙特卡洛搜索树算法
6. 总结
    1. Policy Gradient
        1. 存在方差较大的问题
        2. 采样不足的问题
        3. 一定可以收敛到J的一个局部最优解，通常挺不错的
    2. Q-Learning
        1. 不总是可以工作，因为只是个approximator，存在Exploration的问题
        2. 没有任何保证…