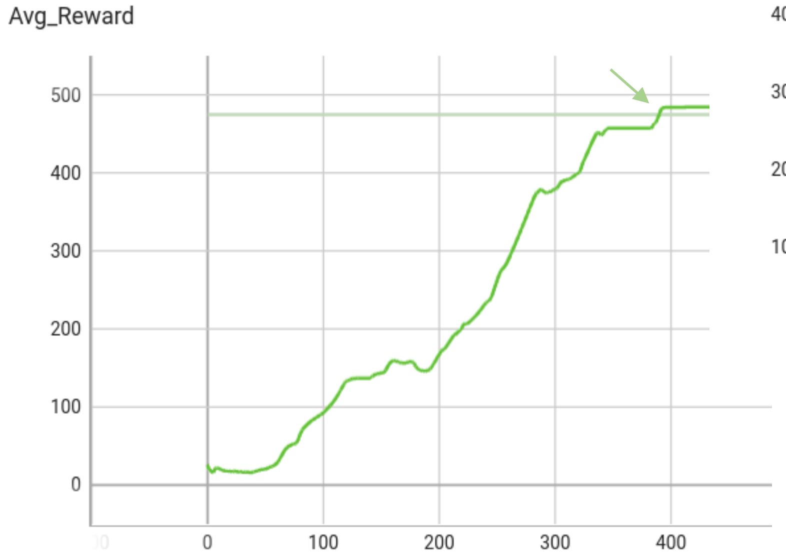
深度强化学习

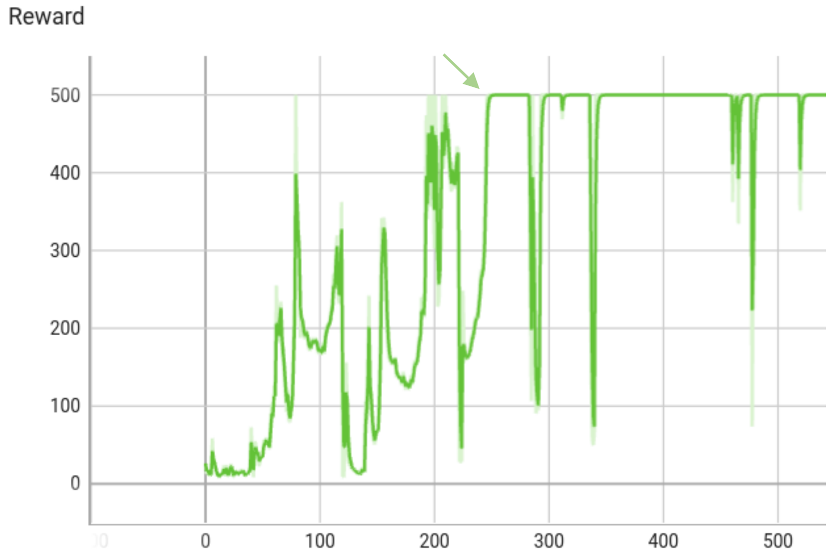
Week 14 Deep Q-learning Network

实验任务

- 用Deep Q-learning Network(DQN)玩CartPole-v1游戏， 框架代码已经给出，至少需要补充‘TODO’标记的代码片段。

- 结果要求与展示
  - 至少完成：
    - 500局(Episodes)游戏内， 达成一次： **连续10局Reward值为500**
    - 展示： “单局Reward值”曲线以及“最近100局的平均Reward值”曲线
  - 进阶：
    - 达成一次： “**最近百局的平均Reward值**” **≥ 475**
    - 更快地达到这个目标
    - 更高的百局平均Reward值
    - …
- Deadline
  - 三周， **6月17日23:59**
- 提交格式
  - E8_学号.zip





续

- 需要补充的代码包括
  - **Qnet**
    - 补充一个线性层
  - **ReplayBuffer**
    - 所有成员函数的实现
  - **DQN**
    - **choose_action**
      - ϵ-greedy策略代码
    - **learn**
      - Q值的计算
      - 目标值计算
      - 损失值计算
      - 梯度下降

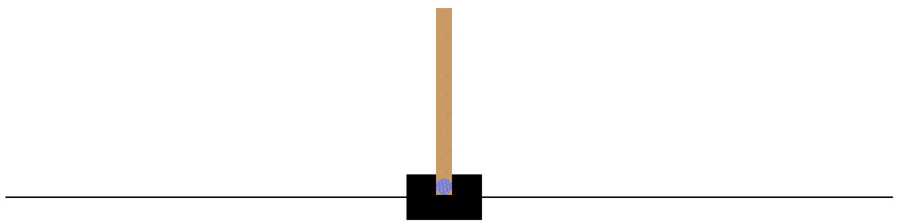
- 可根据需要进一步调整/改进

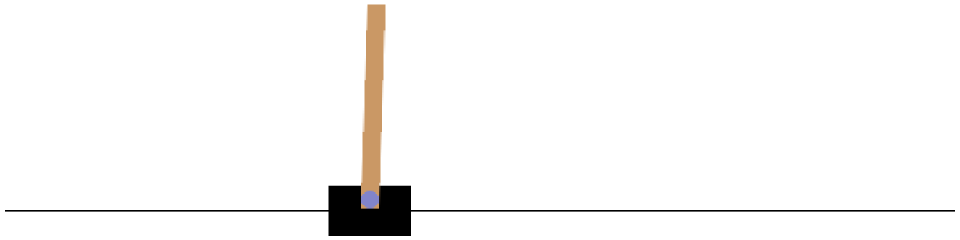
Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory $\boxed{D}$ to capacity N
Initialize action-value function $\boxed{Q}$ with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-}=\theta$
For episode $= 1, M$ **do**
  Initialize sequence $s_1=\{x_1\}$ and preprocessed sequence $\phi_1=\phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t=\text{argmax}_a Q(\phi(s_t),a;\theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1}=s_t,a_t,x_{t+1}$ and preprocess $\phi_{t+1}=\phi(s_{t+1})$
    Store transition $(\phi_t,a_t,r_t,\phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j,a_j,r_j,\phi_{j+1})$ from D
    Set $y_j=\begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j+\gamma\max_{a'}\hat{Q}(\phi_{j+1},a';\theta^{-}) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j-Q(\phi_j,a_j;\theta)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q}=Q$
  End For
End For

CartPole-v1

- 创建环境(gym==0.23)
  - env = gym.make("CartPole-v1")
- 重置环境，并获取初始状态
  - obs = env.reset()
- 执行动作，并获取下一步状态，当前奖励，是否结束等信息
  - next_obs, reward, done, info = env.step(action)
- obs
  - [车位置，车速度，杆角度，杆角速度]
- action
  - 离散值0/1，表示向左/右移动
- reward
  - 不结束时都为1
  - 可根据需要进行修改（奖励工程）





Deep Q-learning Network(DQN)

- 用网络预测Q值，即网络输入状态state，输出该状态下每个动作的Q值
- Q值的更新变为网络参数的更新，因此网络的损失值可定义为均方误差
  - $L = \frac{1}{2}\left(Q(s,a) - \left(r + \gamma \max_{a'} Q(s',a')\right)\right)^2$
- 探索与利用
  - $\epsilon - greedy$ 以概率ϵ随机选择一个动作，以概率 $1-\epsilon$选择最佳动作
- 经验回放
  - 用回放缓存区可以减少与环境做互动的次数，提高训练效率
  - 减少同批次训练数据的依赖关系
  - 做法：将每一步转移的状态、动作、奖励、下一状态等信息存到一个缓冲区，训练网络时从缓冲区随机抽取一批数据作为训练数据

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

DQN2013版（无Target网络）

Deep Q-learning Network(DQN)

- 网络的损失值
  - $L=\frac{1}{2}\left(Q(s,a)-\left(r+\gamma\max_{a'}Q(s',a')\right)\right)^2$
- 目标网络
  - $\left(r+\gamma\max_{a'}Q(s',a')\right)$可看作目标值，目标值跟随Q一直变化会给训练带来困难
  - 将评估网络与目标网络分开，目标网络不训练，评估网络每更新若干轮后，用评估网络参数替换目标网络参数
  - $L=\frac{1}{2}\left(Q_{eval}(s,a)-\left(r+\gamma\max_{a'}Q_{target}(s',a')\right)\right)^2$

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^-=\theta$
For episode = 1, M **do**
  Initialize sequence $s_1=\{x_1\}$ and preprocessed sequence $\phi_1=\phi(s_1)$
  For t = 1,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t=\text{argmax}_a Q(\phi(s_t),a;\theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1}=s_t,a_t,x_{t+1}$ and preprocess $\phi_{t+1}=\phi(s_{t+1})$
    Store transition $(\phi_t,a_t,r_t,\phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j,a_j,r_j,\phi_{j+1})$ from D
    Set $y_j=\begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j+\gamma\max_{a'}\hat{Q}\left(\phi_{j+1},a';\theta^-\right) & \text{otherwise}\end{cases}$
    Perform a gradient descent step on $\left(y_j-Q\left(\phi_j,a_j;\theta\right)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q}=Q$
  End For
End For

DQN2015版（有Target网络）