强化学习

ML24

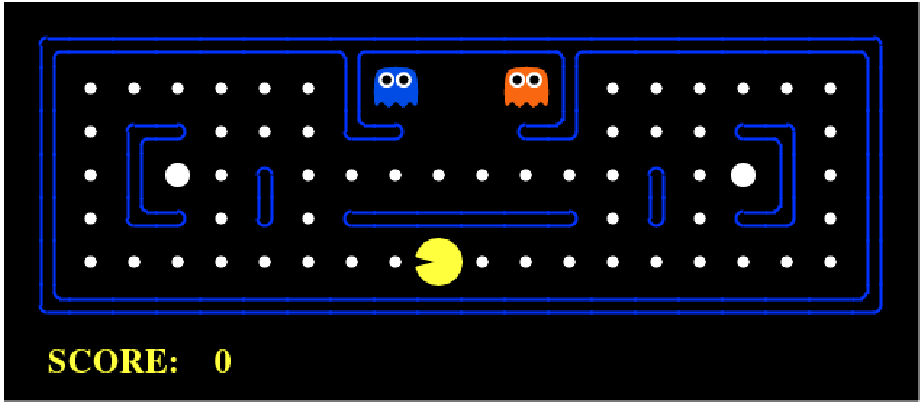
礼欣
www.python123.org

强化学习简介

强化学习

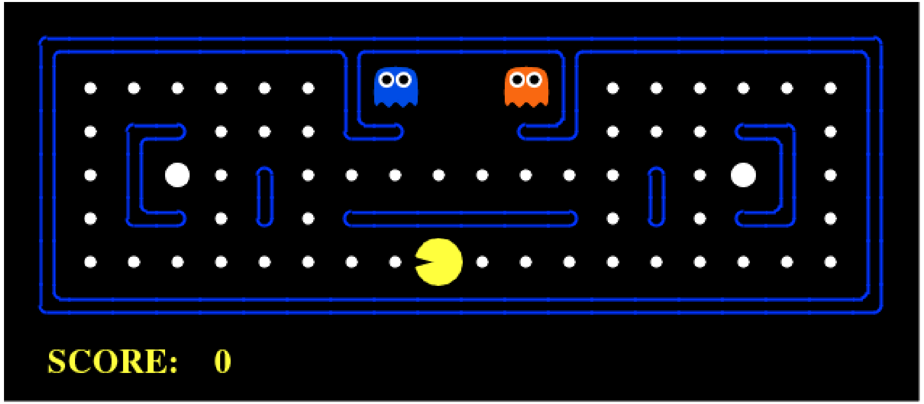
- 强化学习就是程序或智能体（agent）通过与环境不断地进行交互学习一个从环境到动作的映射，学习的目标就是使累计回报最大化。
- 强化学习是一种试错学习，因其在各种状态（环境）下需要尽量尝试所有可以选择的动作，通过环境给出的反馈（即奖励）来判断动作的优劣，最终获得环境和最优动作的映射关系（即策略）。

基本组件



agent： 大嘴小怪物
环境：整个迷宫中的所有信息
奖励：agent每走一步，需要扣除1分，吃掉小球得10分，吃掉敌人得200分，被吃掉游戏结束
动作：在每种状态下，agent能够采用的动作，比如上下左右移动

目标



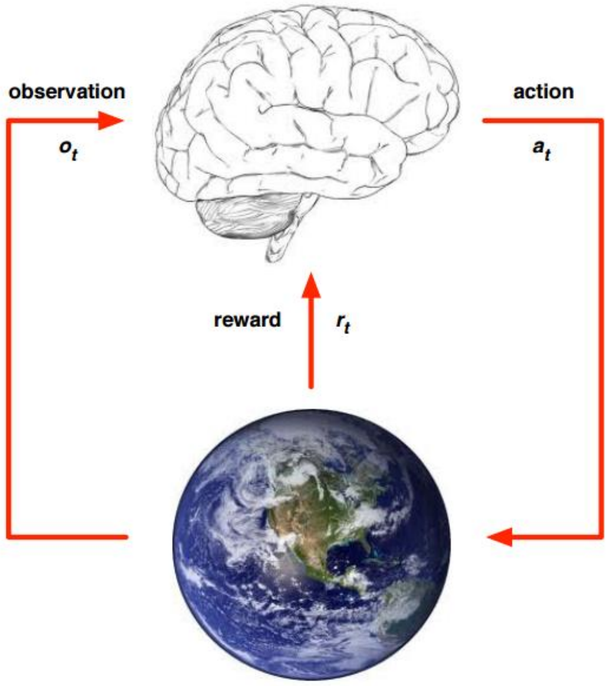
策略：在每种状态下，采取最优的动作

学习目标：获得最优的策略，以使累计奖励最大（即Score）

马尔可夫决策过程

马尔可夫决策过程（MDP）

- 马尔可夫决策过程（Markov Decision Process）通常用来描述一个强化学习问题
- 智能体agent根据当前对环境的观察采取动作获得环境的反馈，并使环境发生改变的循环过程。



MDP 基本元素

$s \in S$:有限状态state集合，s表示某个特定状态；

$a \in A$:有限动作action集合，a表示某个特定动作；

$T(S, a, S') \sim P_r(s'|s,a)$：状态转移模型，根据当前状态s和动作a预测下一个状态s，这里的P_r表示从s采取行动a转移到s'的概率；

$R(s,a)$:表示agent采取某个动作后的即时奖励，它还有 $R(s,a,s')$, $R(s)$ 等表现形式；

Policy $\pi(s) \rightarrow a$：根据当前state来产生action，可表现为$a=\pi(s)$或$\pi(a|s) = P(a|s)$，后者表示某种状态下执行某个动作的概率。

值函数

状态值函数V表示执行策略π能得到的累计折扣奖励：

$$V^{\pi}(s) = E[R(s_0,a_0)+\gamma R(s_1,a_1)+\gamma^2 R(s_2,a_2)+\gamma^3 R(s_3,a_3)+\ldots|s=s_0]$$

整理之后可得：

$$\boldsymbol{V^{\pi}(s) = R(s,a) + \gamma \sum_{s' \in S} p\big(s'|s,\pi(s)\big)V^{\pi}(s')}$$

值函数

状态动作值函数Q(s,a)表示在状态s下执行动作a能得到的累计折扣奖励：

$Q^{\pi}(s,a) =$

$E[R(s_0,a_0)+\gamma R(s_1,a_1)+\gamma^2 R(s_2,a_2)+\gamma^3 R(s_3,a_3)+\ldots|s=s_0,a=a_0]$

整理之后可得：

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in S} p\left(s' | s, \pi(s)\right) Q^{\pi}\left(s', \pi(s')\right)$$

最优值函数

最优值函数：

$$\boldsymbol{V^*(s) = max_{a\in A}[R(s,a) + \gamma\sum_{s'\in S} p(s'|s,a)V^*(s')]}$$

$$\boldsymbol{Q^*(s,a) = R(s,a) + \gamma\sum_{s'\in S} p\left(s'|s,a\right) max_{b\in A}Q^*(s',b)}$$

最优控制

在得到最优值函数之后，可以通过值函数的值得到状态s时应该采取的动作a：

$$\boldsymbol{\pi(s) = argmax_{a\in A}[R(s,a) + \gamma \sum_{s'\in S} p(s'|s,a)V^*(s')]}$$

$$\boldsymbol{\pi(s) = argmax_{a\in A}\ Q^*(s,a)}$$

$$\boldsymbol{V^*(s) = max_{a\in A}\ Q^*(s,a)}$$

Q-learning

蒙特卡洛强化学习

- 在现实的强化学习任务中，环境的转移概率、奖励函数往往很难得知，甚至很难得知环境中有多少状态。若学习算法不再依赖于环境建模，则称为免模型学习，蒙特卡洛强化学习就是其中的一种。
- 蒙特卡洛强化学习使用多次采样，然后求取平均累计奖赏作为期望累计奖赏的近似。

$$<s_0,a_0,r_1,s_1,a_1,r_2,\ldots,s_{T-1},a_{T-1},r_T,s_T>$$

蒙特卡洛强化学习

蒙特卡洛强化学习：直接对状态动作值函数Q(s,a)进行估计，每采样一条轨迹，就根据轨迹中的所有“状态-动作”利用下面的公式对来对值函数进行更新。

$$Q(s,a)=\frac{\mathbf{Q(s,a)*count(s,a)+R}}{\mathbf{count(s,a)+1}}$$

蒙特卡洛强化学习

每次采样更新完所有的“状态-动作”对所对应的Q(s,a)，就需要更新采样策略π。但由于策略可能是确定性的，即一个状态对应一个动作，多次采样可能获得相同的采样轨迹，因此需要借助ε贪心策略：

$$\boldsymbol{\pi(s,a)} = \begin{cases} \boldsymbol{argmax_a Q(s,a)} & \text{以概率}\boldsymbol{1-\varepsilon} \\ \text{随机从}\boldsymbol{A}\text{中选取动作} & \text{以概率}\boldsymbol{\varepsilon} \end{cases}$$

Q-learning算法

- 蒙特卡洛强化学习算法需要采样一个完整的轨迹来更新值函数，效率较低，此外该算法没有充分利用强化学习任务的序贯决策结构。
- Q-learning算法结合了动态规划与蒙特卡洛方法的思想，使得学习更加高效。

Q-learning算法

假设对于状态动作对(s,a)基于t次采样估算出其值函数为：

$$Q_t^{\pi}(s,a)=\frac{1}{t}\Sigma_{i=1}^{t}r_i$$

在进行t+1次采样后，依据增量更新得到：

$$\mathrm{Q}_{\mathrm{t+1}}^{\pi}(\mathrm{s,a})=\mathrm{Q}_{\mathrm{t}}^{\pi}(\mathrm{s,a})+\frac{1}{t+1}\left(r_{t+1}-\mathrm{Q}_{\mathrm{t}}^{\pi}(\mathrm{s,a})\right)$$

然后，将$\frac{1}{t+1}$替换成系数α（步长），得到：

$$\mathrm{Q}_{\mathrm{t+1}}^{\pi}(\mathrm{s,a})=\mathrm{Q}_{\mathrm{t}}^{\pi}(\mathrm{s,a})+\alpha\left(r_{t+1}-\mathrm{Q}_{\mathrm{t}}^{\pi}(\mathrm{s,a})\right)$$

Q-learning算法

以γ折扣累计奖赏为例：

$$\mathbf{r_{t+1}} = \mathbf{R_s^a} + \boldsymbol{\gamma}\mathbf{Q_t^{\pi}}(\mathbf{s'}, \mathbf{a'})$$

则值函数的更新方式如下：

$$\mathbf{Q_{t+1}^{\pi}}(\mathbf{s}, \mathbf{a}) = \mathbf{Q_t^{\pi}}(\mathbf{s}, \mathbf{a}) + \alpha(\mathbf{R_s^a} + \boldsymbol{\gamma}\mathbf{Q_t^{\pi}}(\mathbf{s'}, \mathbf{a'}) - \mathbf{Q_t^{\pi}}(\mathbf{s}, \mathbf{a}))$$

Q-learning算法流程

输入：环境E；动作空间A；起始状态s_0；奖励折扣γ；更新步长α；

过程：

1：$Q(s,a)=0,\ \pi(s,a)=1/|A|$;

2：$s=s_0$;

3：for t=1,2,…do

4： r,s’= 在E中执行动作$\pi^{\epsilon}(s)$产生的奖赏和转移的状态;

5： $a'=\pi(s')$;

6： $Q(s,a)=Q(s,a)+\alpha(r+\gamma Q(s',a')-Q(s,a))$;

7： $\pi(s)=\arg\max_{a''}Q(s,a'')$;

8： $s=s',a=a'$;

9：end for

输出：策略π

请参考周志华老师《机器学习》一书

深度强化学习

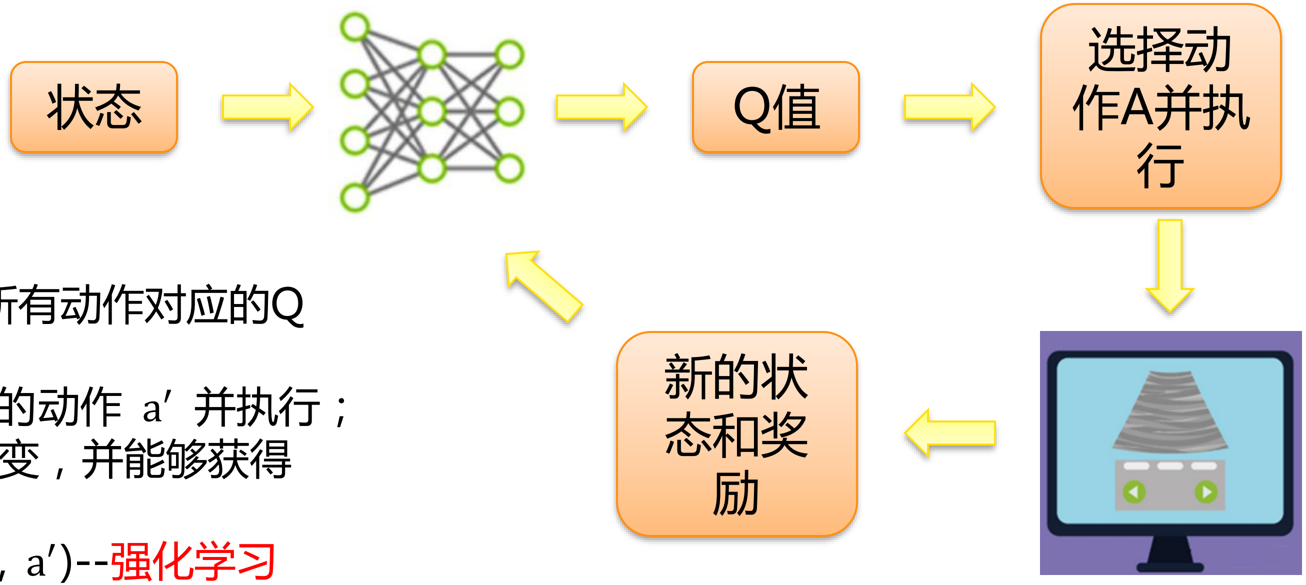
深度强化学习（DRL）

- 传统强化学习：真实环境中的状态数目过多，求解困难。
- 深度强化学习：将深度学习和强化学习结合在一起，通过深度神经网络直接学习环境（或观察）与状态动作值函数Q(s,a)之间的映射关系，简化问题的求解。

Deep Q Network（DQN）

- Deep Q Network（DQN）：是将神经网络(neural network) 和Q-learning结合，利用神经网络近似模拟函数Q(s,a)，输入是问题的状态（e.g.,图形），输出是每个动作a对应的Q值，然后依据Q值大小选择对应状态执行的动作，以完成控制。
- 神经网络的参数：应用监督学习完成

DQN学习过程



学习流程：

1.状态s输入，获得所有动作对应的Q值Q(s,a)；

2.选择对应Q值最大的动作 a′ 并执行；

3.执行后环境发生改变，并能够获得环境的奖励r；

4.利用奖励r更新Q(s, a′)--强化学习

利用新的Q(s, a′)更新网络参数—监督学习

DQN算法流程

初始化D：用于存放采集的（S_t，a_t，r_t，S_{t+1}）状态转移过程，用于网络参数的训练

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

进行多次采样

一次完整的采样

DQN算法流程

Algorithm 1 Deep Q-learning with Experience Replay

随机初始化神经网络的参数

进行多次采样

一次完整的采样

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

DQN算法流程

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

进行多次采样

一次完整的采样

获取环境的初始状态（x是采集的图像，使用图像作为agent的状态；预处理过程是说，使用4张图像代表当前状态，这里可以先忽略掉）

DQN算法流程

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

进行多次采样

一次完整的采样

ε贪心策略：使用ε概率随机选取动作或1- ε的概率根据神经网络的输出选择动作

DQN算法流程

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

进行多次采样

一次完整的采样

在模拟器中执行选定的动作，获得奖励r_t和一个观察x_{t+1}

DQN算法流程

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

进行多次采样

一次完整的采样

设置S_{t+1}，并将状态转移过程（S_t，a_t，r_t，S_{t+1}）存放在D中

DQN算法流程

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

进行多次采样

一次完整的采样

从D中进行随机采样，获得一部分状态转移过程历史信息

DQN算法流程

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 equ
  end for
end for

进行多次采样

一次完整的采样

使用Q-learning方法
更新状态值函数的值
（终止与非终止状态
的更新不同）

DQN算法流程

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 equ
  end for
end for

进行多次采样

一次完整的采样

使用监督学习方法更新网络的参数