强化学习原理：
——值函数近似表示

杨博渊

yby@nankai.edu.cn

2023.03.31

提纲

- 概述
- 参数化函数逼近
  - 线性回归
  - 人工神经网络
- 参数估计
  - 增量式方法
  - 批方法
- 非参数函数逼近

概述

动态规划值函数迭代算法

输入：状态转移概率 $P_{ss'}^{a}$,回报函数R_s^a ，折扣因子γ

初始化值函数:$v(s)=0$ 初始化策略 π_0

Repeat l=0,1,…

for every s do

$$v_{l+1}(s)=\max_{a} R_s^a+\gamma\sum_{s'\in S}P_{ss'}^{a}v_l(s')$$

Until $v_{l+1}=v_l$

输出： $\pi(s)=\underset{a}{\operatorname{argmax}} R_s^a+\gamma\sum_{s'\in S}P_{ss'}^{a}v_l(s')$

[1] 初始化所有: $s\in S, a\in A(s), Q(s,a)\leftarrow arbitrary$

$Returns(s,a)\leftarrow empty\ list$

$\pi(s)\leftarrow arbitrary\ \varepsilon$-$soft$策略,

Repeat:

[2] 从 S_0, A_0 开始以策略 π 生成一次实验(episode),

[3] 对每对在这个实验中出现的状态和动作, s, a:

$G\leftarrow s,a$第一次出现后的回报

将G附加于回报Returns(s, a)上

$Q(s,a)\leftarrow average(Returns(s,a))$对回报取均值

策略评估

[4] 对该实验中的每一个s:

策略改进

$$\pi(a|s)\leftarrow\begin{cases}1-\varepsilon+\dfrac{\varepsilon}{|A(s)|} & if\ a=\arg\max_a Q(s,a)\\ \dfrac{\varepsilon}{|A(s)|} & if\ a\neq\arg\max_a Q(s,a)\end{cases}$$

概述

1. 初始化 $Q(s,a),\forall s\in S, a\in A(s)$,给定参数$\alpha$, γ

2. Repeat:

行动策略和评估策略都是 ε 贪婪策略

给定起始状态 s, 并根据 贪婪策略在状态 s 选择动作 a

Repeat(对于一幕的每一步)

(a) 根据 贪婪策略在状态 s 选择动作 a , 得到回报 r 和下一个状态s' , 在状态 s'

根据 贪婪策略得到动作a'

(b) $Q(s,a)\leftarrow Q(s,a)+\alpha\left[r+\gamma Q(s',a')-Q(s,a)\right]$

(c) s=s' , a=a'

Until s 是终止状态

Until 所有的 收敛

3. 输出最终策略: $\pi(s)=\arg\max_a Q(s,a)$

概述

1. 初始化 $Q(s,a), \forall s \in S,\ a \in A(s)$,给定参数$\alpha$，$\gamma$

2. Repeat:

   给定起始状态 s， 并根据 ε贪婪策略在状态 s 选择动作 a

   Repeat(对于一幕的每一步)

   行动策略为 ε 贪婪策略

   $\boxed{\text{(a) 根据 } \varepsilon \text{ 贪婪策略在状态 s\_t 选择动作a\_t ， 得到回报r\_t 和下一个状态 } S_{t+1}}$

   (b) $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right]$ 目标策略为贪婪策略

   (c) s=s’ ， a=a’

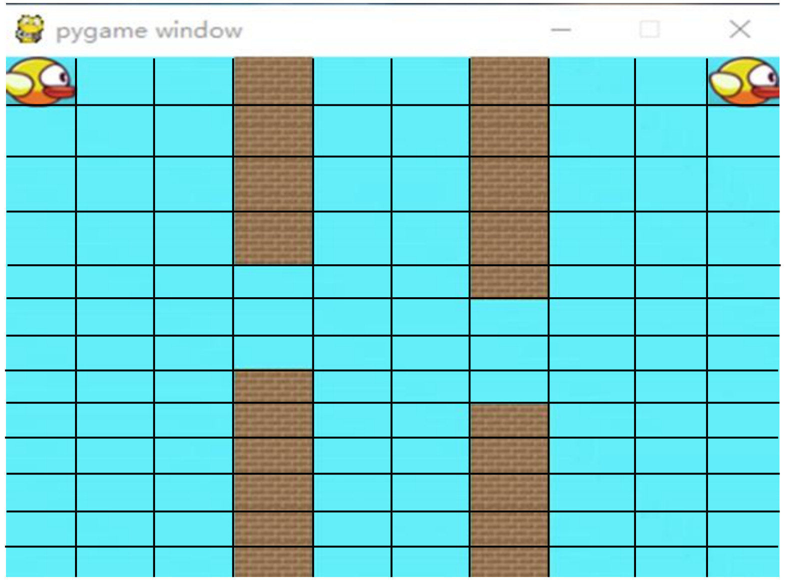
   Until s 是终止状态

   Until 所有的 $Q(s,a)$ 收敛

3. 输出最终策略：$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a)$

概述

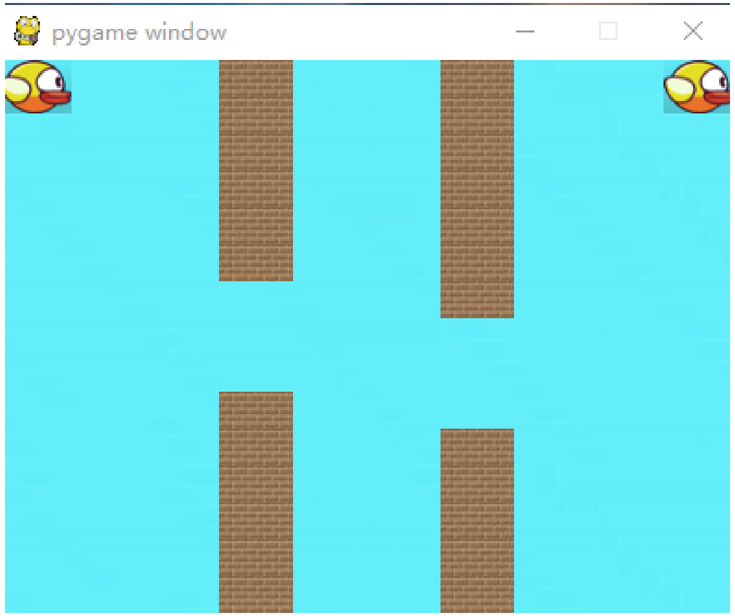
$q(s,a)$



行为值函数的数目： $|S|\cdot|A|$

大的MDP问题：状态空间连续或状态数无穷多

1. 存储空间无穷大
2. 学习速度很慢

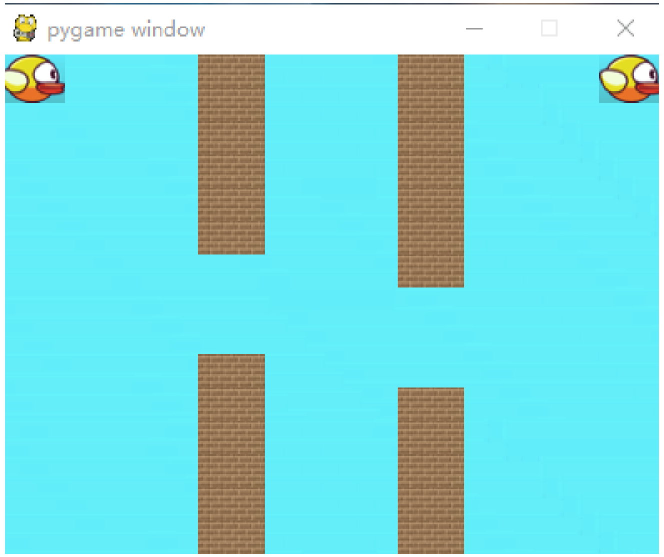


状态空间表示为：图像，如分辨率为（400，500）
图像空间的大小为 256^{20000}

1.存储空间无穷大，计算时间无穷。

2. 几乎每次遇到的状态下次都不会再遇到（泛化能力），需要泛化能力强

概述

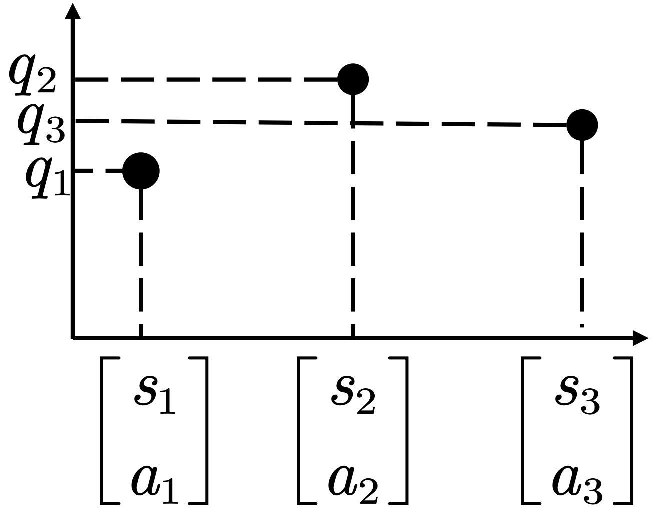


采集到的数据:

$$Q(s,a)=\{q_1,q_2,\cdots,q_n\}$$

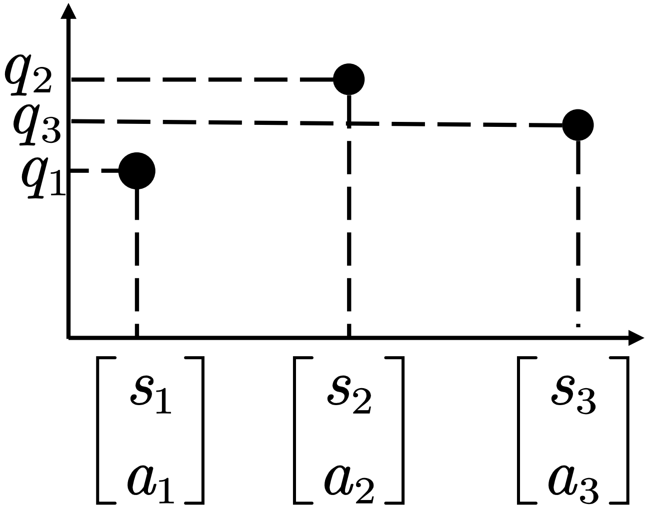
如何得到其他状态行为值函数?

$$Q(s',a')$$



利用泛化方法得到其他状态行为处的值函数

概述



利用泛化方法得到其他状态行为处的值函数

基于函数逼近的强化学习：

强化学习+泛化方法

泛化方法：函数逼近理论，机器学习！

Remark: 不同的地方，非静态性，自举和延迟目标

利用函数逼近方法估计值函数 $\hat{q}(s,a;\theta)$

1. 参数化逼近
2. 非参数化逼近， 如基于核的方法

参数化方法：

线性参数化：各种人为基底

非线性参数化：人工神经网络，决策树，模糊网络

提纲

- 概述
- 参数化函数逼近
  - 线性回归
  - 人工神经网络
- 参数估计
  - 增量式方法
  - 批方法
- 非参数函数逼近

线性回归

线性逼近：

$$\hat{q}(s,a;\theta) = \theta^T \phi(s,a)$$

$\phi(s)$ 称为状态 s 的特征函数。

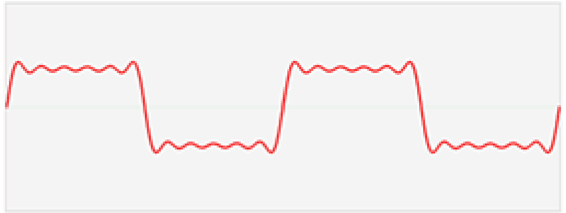
常用的基函数类型：

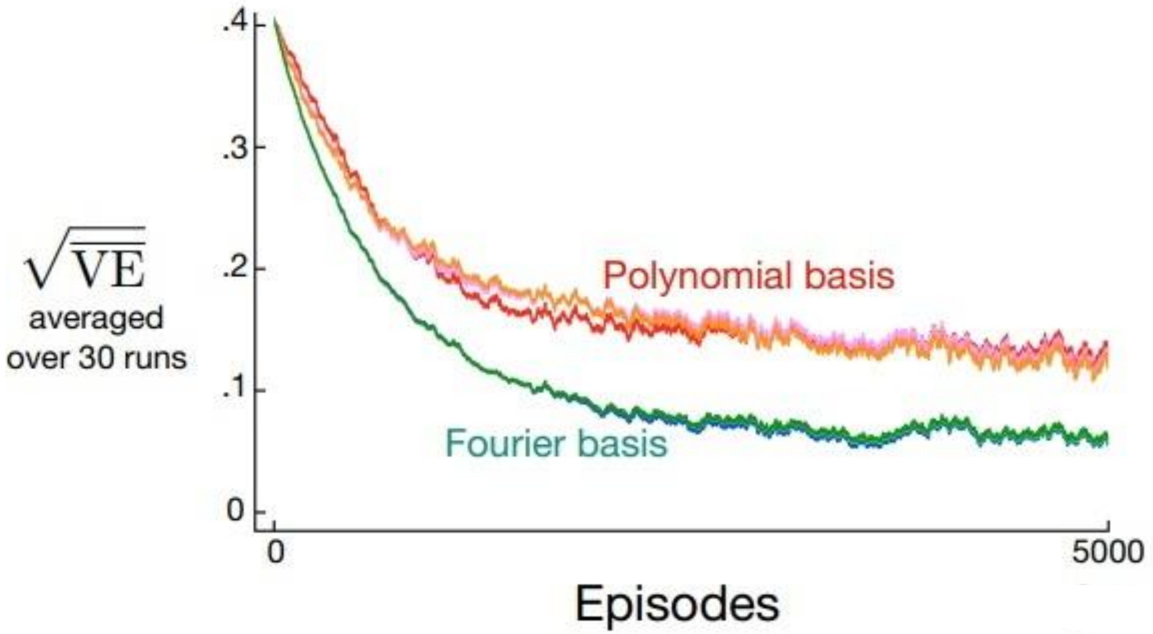
多项式基函数：$(1, s_1, s_2, s_1 s_2, s_1^2, s_2^2, \cdots)$

傅里叶基函数：$\phi_i(s) = \cos(i\pi s),\ s \in [0,1]$

径向基函数：$\phi_i(s) = \exp\left(-\frac{\|s - c_i\|^2}{2\sigma_i^2}\right)$

傅里叶基函数





$\sqrt{\mathrm{VE}}$
averaged
over 30 runs
.4
.3
.2
.1
0
0
5000
Episodes
Polynomial basis
Fourier basis

提纲

- 概述
- 参数化函数逼近
  - 线性回归
  - 人工神经网络
- 参数估计
  - 增量式方法
  - 批方法
- 非参数函数逼近

神经网络

提纲

- 概述
- 参数化函数逼近
  - 线性回归
  - 人工神经网络
- 参数估计
  - 增量式方法
  - 批方法
- 非参数函数逼近

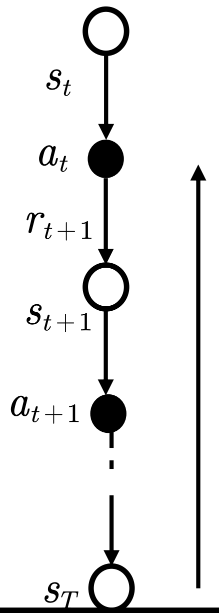
值函数估计过程

表格型值函数估计

Backup 值：

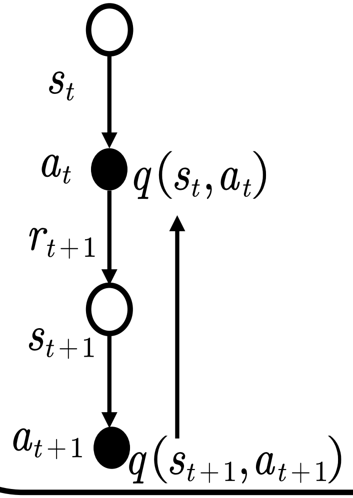
$$\text{DP}\quad Q_\pi(s,a) = R_s^a + \gamma \sum_{s'\in S} P_{ss'}^a \sum_{a'\in A} \pi(a'|s')\, Q_\pi(s',a')$$

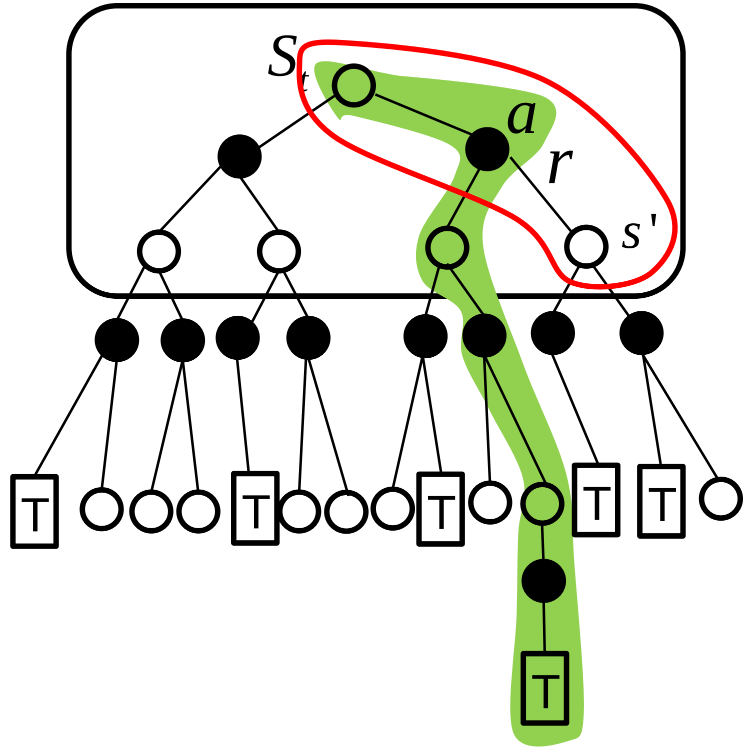
MC的回溯值为： $\sum_{k=1}^{N} \gamma^{k-1} r_{t+k}$



TD回溯值为：

$$r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$$



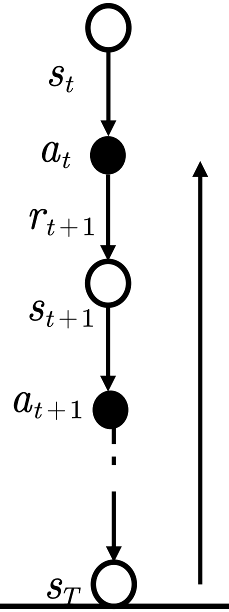


值函数估计过程

Backup 值：

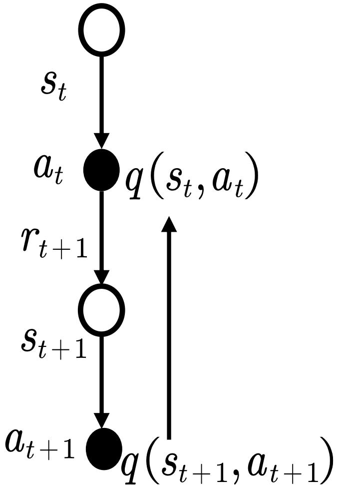
DP $Q_\pi(s,a) = R_s^a + \gamma \sum_{s'\in S} P_{ss'}^a \sum_{a'\in A} \pi(a'|s')\, Q_\pi(s',a')$

MC的回溯值为：$\sum_{k=1}^{N} \gamma^{k-1} r_{t+k}$

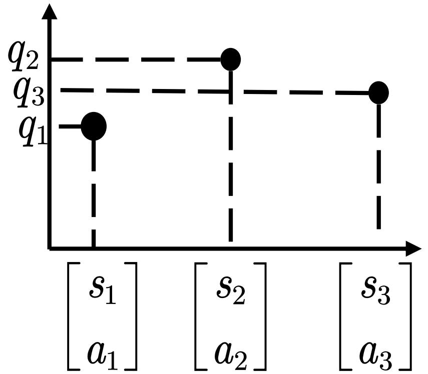


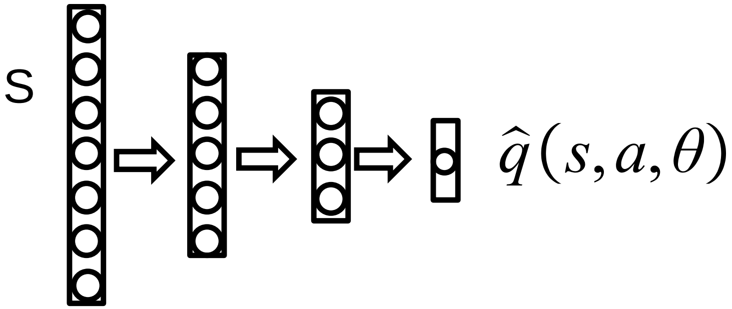
TD回溯值为：

$r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$



函数逼近：$\hat{q}(s,a,\theta)$



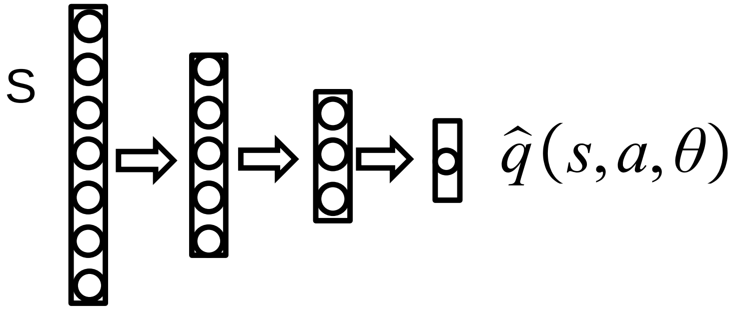


训练目标：$\arg\min_{\theta} \in \left(q(s,a) - \hat{q}(s,a,\theta)\right)^2$

强化学习：在线学习

值函数逼近的损失函数

函数逼近: $\hat{q}(s,a,\theta)$

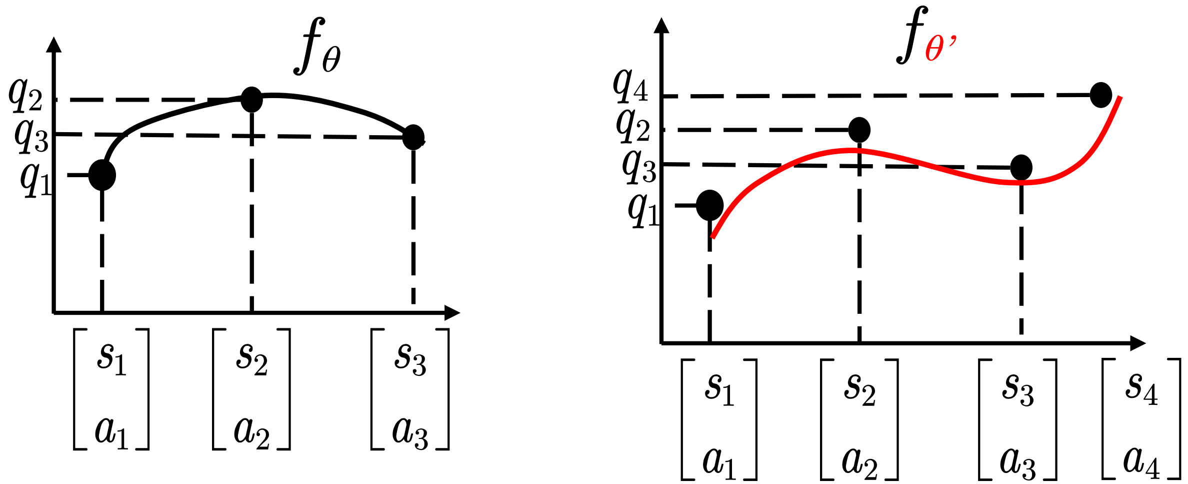


训练目标：$\arg\min_{\theta} \in \left(q(s,a) - \hat{q}(s,a,\theta)\right)^2$

强化学习：在线学习

参数改变后，其他状态值函数跟着也发生变化

非静态目标函数：增量式学习方法和批方法



目标函数的构建:

$$\overline{VE}(w) = \sum_{s\in\mathcal{S}} \mu(s)\left[q_\pi(s,a) - \hat{q}(s,w)\right]^2$$

提纲

- 概述
- 参数化函数逼近
  - 线性回归
  - 人工神经网络
- 参数估计
  - 增量式方法
  - 批方法
- 非参数函数逼近

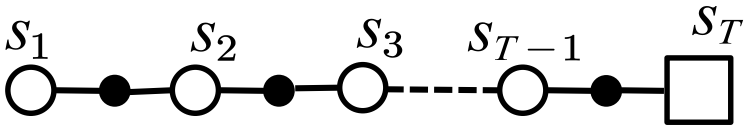
随机梯度下降

随机梯度下降法：

$$\theta_{t+1} = \theta_t + \alpha[U_t(s,a) - \hat{q}(s_t,a;\theta)]\nabla\hat{q}(s_t,a_t;\theta)$$

对于蒙特卡罗方法： $U_t(s) = G_t(s)$

给定策略 π 产生一次试验：



值函数估计过程：

监督学习，训练数据集为：$\langle s_1, G_1\rangle, \langle s_2, G_2\rangle, \cdots$

策略评估过程为：

$$\Delta\theta = \alpha[U_t(s,a) - \hat{q}(s_t,a;\theta)]\nabla\hat{q}(s_t,a_t;\theta)$$

α 比较小，用来平衡不同状态值函数的误差

基于梯度的蒙塔卡罗值函数评估算法

输入：要评估的策略 π，一个可微逼近函数 $\hat{v}: S\times R^n \to R$

恰当地初始化的值函数权重 θ （例如 $\theta = 0$ ）

Repeat:

利用策略 π 产生一幕数据 $S_0, A_0, R_1, S_1, A_1, \cdots, R_T, S_T$

$$for\ t = 0, 1, \cdots, T-1$$

$$\theta \leftarrow \theta + \alpha[G_t - \hat{q}(s_t,a;\theta)]\nabla\hat{q}(s_t,a;\theta)$$

半梯度下降

对于TD(0)，DP等

Bootstrapping

$$U_t = R_{t+1} + \gamma \boxed{\hat{q}(S_{t+1}, a; \theta)}$$

半梯度方法:

$$\theta_{t+1} = \theta_t + \alpha[U_t(s,a) - \hat{q}(s_t,a;\theta)]\nabla\hat{q}(s_t,a_t;\theta)$$

$U_t(s)$ 依赖于当前的参数估计 θ

只考虑参数 θ 对估计值函数 $q(s_t,a_t;\theta_t)$ 的影响而忽略对目标函数u_t的影响，称为半梯度法。

好处：学习速度快，可以应用到连续系统中

基于半梯度的TD(0)值函数评估算法

输入：要评估的策略 π，一个可微逼近函数 $\hat{v}: S \times R^n \to R$

恰当地初始化的值函数权重 θ （例如 $\theta = 0$ ）

Repeat:

初始化状态S,

Repeat(对于一幕中的每一步)

选择动作 $\upsilon(S_t, \theta_t)$

采用动作 A 并观测回报 R, S'

$$\theta \leftarrow \theta + \alpha[G_t - \hat{q}(s_t,a;\theta)]\nabla\hat{q}(s_t,a;\theta)$$

$$S \leftarrow S'$$

直到 S’ 是终止状态

半梯度Sarsa算法

输入：一个要逼近的可微动作值函数：$\hat{q}: S \times A \times R^n \rightarrow R$ 任意地初始化的值函数权重 θ（例如$\theta = 0$）

Repeat（for each episode）：

初始化状态行为对 S, A

Repeat（对于每一幕数据中的每一步）：

采用动作 A, 得到回报 R, 和下一个状态 S’

如果 S’ 是终止状态：

进入下一幕

利用软策略选择一个动作 A’ ，以便估计动作值函数 $\hat{q}(S', A', \theta)$

$$\theta \leftarrow \theta + \alpha\left[R + \gamma\hat{q}(S', A', \theta) - \hat{q}(S, A, \theta)\right]\nabla\hat{q}(S, A, \theta)$$

$$S \leftarrow S'$$

$$A \leftarrow A'$$

值函数的线性逼近

基于梯度或半梯度的的值函数逼近：

$$\theta_{t+1}=\theta_t+\alpha[U_t(s)-\hat{q}(s_t,a;\theta_t)]\nabla\hat{q}(s_t,a;\theta_t)$$

线性逼近：

$$\hat{q}(s,a;\theta)=\theta^T\phi(s,a)$$

线性逼近的好处：在线性情况，仅有一个最优值，因此可收敛到全局最优。

$\phi(s)$ 称为状态 s 的特征函数。

常用的基函数类型：

多项式基函数：$(1,s_1,s_2,s_1s_2,s_1^2,s_2^2,\cdots)$

傅里叶基函数：$\phi_i(s)=\cos(i\pi s)$，$s\in[0,1]$

径向基函数：$\phi_i(s)=\exp\left(-\dfrac{\|s-c_i\|^2}{2\sigma_i^2}\right)$

蒙特卡罗方法值函数更新：

$$\Delta\theta=\alpha[U_t(s)-\hat{q}(s_t,a_t;\theta_t)]\nabla\hat{q}(s_t,a;\theta_t)$$

$$=\alpha[G_t-\theta^T\phi]\phi$$

TD(0)线性逼近值函数更新为：

$$\Delta\theta=\alpha[R+\gamma\theta^T\phi(s')-\theta^T\phi(s)]\phi(s)$$

$$=\alpha\delta\phi(s)$$

正向视角 $TD(\lambda)$

$$\Delta\theta=\alpha(G_t^\lambda-\theta^T\phi)\phi$$

反向视角 $TD(\lambda)$

$$\delta_t=R_{t+1}+\gamma\theta^T\phi(s')-\theta^T\phi(s)$$

$$E_t=\gamma\lambda E_{t-1}+\phi(s)$$

$$\Delta\theta=\alpha\delta_t E_t$$

提纲

- 概述
- 参数化函数逼近
  - 线性回归
  - 人工神经网络
- 参数估计
  - 增量式方法
  - 批方法
- 非参数函数逼近

批方法

增量式方法：计算简单，但往往存在样本效率不高的缺点。

批的方法寻找最好的拟合函数值$\hat{q}(s,a;\theta)$ 。

给定经验数据集：

$$D=\{\langle s_1,v_1^{\pi}\rangle,\langle s_2,v_2^{\pi}\rangle,\cdots,\langle s_T,v_T^{\pi}\rangle\}$$

最小二乘方法：找到最优参数

$$LS(\theta)=\sum_{t=1}^{T}(q_t^{\pi}-\hat{q}(s_t,a;\theta))^2$$

$$=E_D[(q^{\pi}-\hat{q}(s,a;\theta))^2]$$

线性最小二乘逼近

$$\Delta\theta=\alpha\sum_{t=1}^{T}[q_t^{\pi}-\theta^T\phi(s_t,a)]\phi(s_t)=0$$

最小二乘蒙特卡罗方法：

LSMC:

$$\theta=\left(\sum_{t=1}^{T}\phi(s_t)\phi(s_t)^T\right)^{-1}\sum_{t=1}^{T}\phi(s_t)G_t$$

最小二乘时间差分方法：

LSTD:

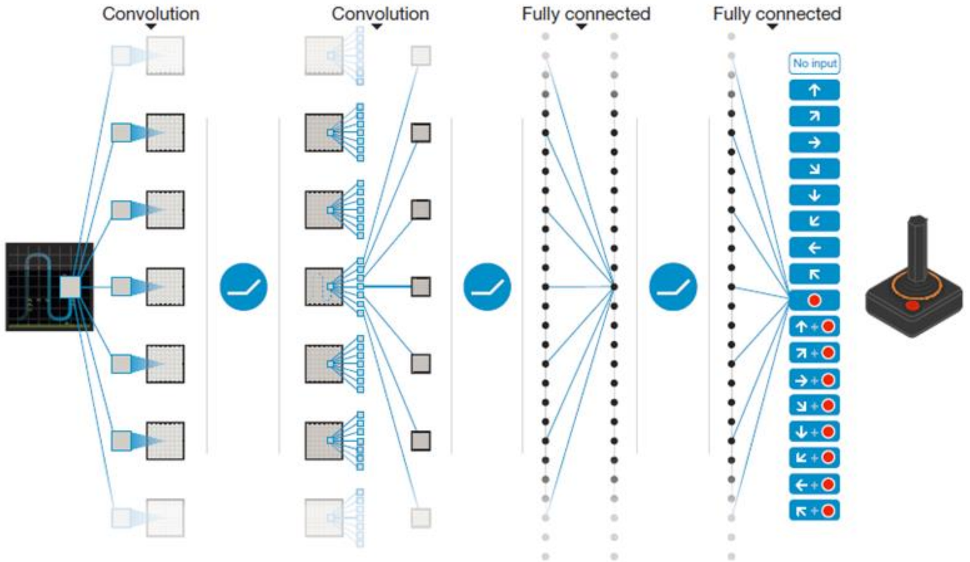
$$\theta=\left(\sum_{t=1}^{T}\phi(s_t)(\phi(s_t)-\gamma\phi(s_{t+1}))^T\right)^{-1}\sum_{t=1}^{T}\phi(s_t)R_{t+1}$$

最小二乘TD(λ)方法：

LSTD（λ）：

$$\theta=\left(\sum_{t=1}^{T}E_t(\phi(s_t)-\gamma\phi(s_{t+1}))^T\right)^{-1}\sum_{t=1}^{T}E_tR_{t+1}$$

DQN



DQN网络

1. 框架为qlearning
2. 值函数表示为卷积神经网络

DQN

Qlearning伪代码

1. 初始化 $Q(s,a), \forall s \in S, a \in A(s)$,给定参数$\alpha$，$\gamma$

2. Repeat:

3. 给定起始状态 s, 并根据 ε贪婪策略在状态 s 选择动作 a

4. Repeat(对于一幕的每一步)

5. $\boxed{\text{(a) 根据 } \varepsilon \text{ 贪婪策略在状态 } s_t \text{ 选择动作 } a_t\text{，得到回报 } r_t \text{ 和下一个状态 } s_{t+1}}$

6. (b) $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right]$

7. (c) s=s’，a=a’

8. Until s 是终止状态

9. Until 所有的 $Q(s,a)$ 收敛

10. 输出最终策略：$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a)$

行动策略为 s_t 贪婪策略

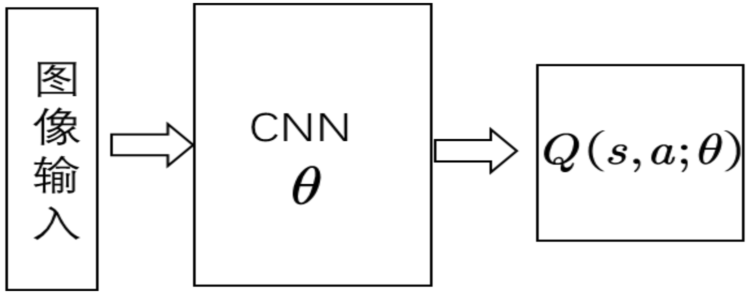
目标策略为贪婪策略

DQN伪代码

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

DQN要点

（1）DQN利用卷积神经网络逼近行为值函数



（2）DQN利用经验回放对强化学习过程进行训练

$\langle s_1,a_1,r_2,s_2\rangle$
$\langle s_2,a_2,r_3,s_3\rangle$
$\langle s_3,a_3,r_4,s_4\rangle$
$\langle s_4,a_4,r_5,s_5\rangle$
$\langle s_5,a_5,r_6,s_6\rangle$
$\vdots$

（3）DQN设置了目标网络来单独处理时间差分算法中的TD偏差。

$$\theta_{t+1}=\theta_t+\alpha\left[r+\gamma\max_{a'}Q(s',a';\theta^-)-Q(s,a;\theta)\right]\nabla Q(s,a;\theta)$$

作业

1.阅读《Reinforcement Learning： An Introduction》第9章