第一章 随机梯度下降

1.1 设计思想

1. 采样点存储

   采用二维 vector 保存点的坐标，每一行表示(x, y)。使用 random 函数随机生成点的坐标。易得$\mathbb{E}[X]=0$。

2. 随机梯度下降

   根据采样次数，依次迭代，每次迭代选择一个坐标，根据$w_{k+1}=w_k-\alpha_k(w_k-x_k)$更新均值估计，同时计算$w_k$与$\mathbb{E}[X]$的估计误差。

3. 小批量梯度下降

   根据采样次数，依次迭代，每次迭代根据批量设置选择点的个数，根据$w_{k+1}=w_k-\alpha_k\frac{1}{n}\sum_{i=1}^{n}(w_k-x_i)$更新均值估计，同时计算$w_k$与$\mathbb{E}[X]$的估计误差。

1.2 伪码描述

1. 随机梯度下降

   初始化：初始化w_0

   目标：求解以下最优问题$\min J(w)=E[f(w,X)]$

   对于第 k 次采样，do

      更新w值：$w_{k+1}=w_k-\alpha_k(w_k-x_k)$

2. 小批量梯度下降

   初始化：初始化w_0

   目标：求解以下最优问题$\min J(w)=E[f(w,X)]$

   对于第 k 次采样，do

      采集 m 个样本$\{x_1,x_2,\dots,x_m\}$

      更新w值：$w_{k+1}=w_k-\alpha_k\frac{1}{n}\sum_{i=1}^{m}(w_k-x_i)$

1.3 时间复杂度分析

K:总采样次数，M:小批量梯度下降批量大小

随机梯度下降：$O(K)$

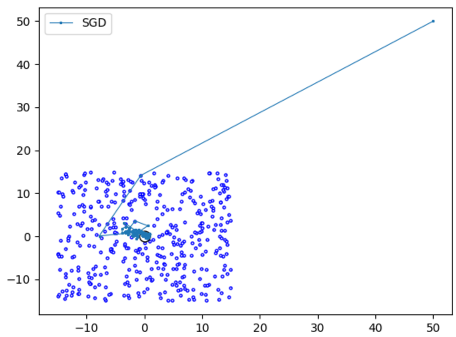
小批量梯度下降：$O(K*M)$

1.4 结果分析

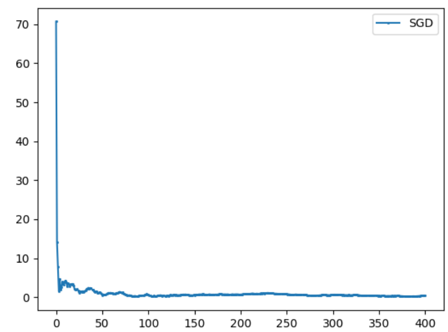
在以原点为中心、边长为 30 的正方形区域内随机采样 400 个样本。均值 $\mathbb{E}(X)=0$。

1. 随机梯度下降

a) $\alpha_k = 1/k$



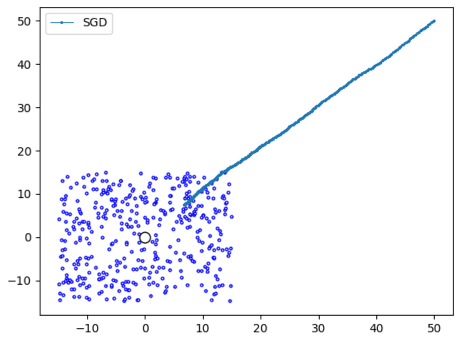
(a) 求解收敛轨迹



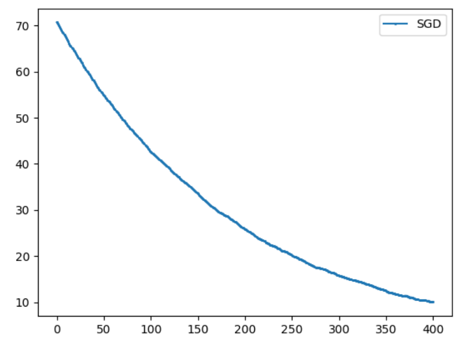
(b) 误差

当$\alpha_k = 1/k$时，从图(a)可以看出w_k前几次估计迅速靠近$\mathbb{E}(X)$，之后在$\mathbb{E}(X)$附近徘徊，图(b)也能得出相同结论，前几次误差迅速下降，之后误差接近 0，能估计出$\mathbb{E}(X)$。

b) $\alpha_k = 0.005$



(c) 求解收敛轨迹

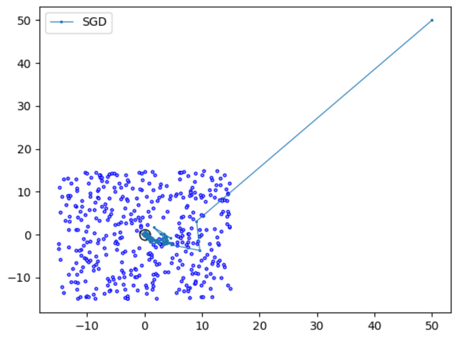


(d) 误差

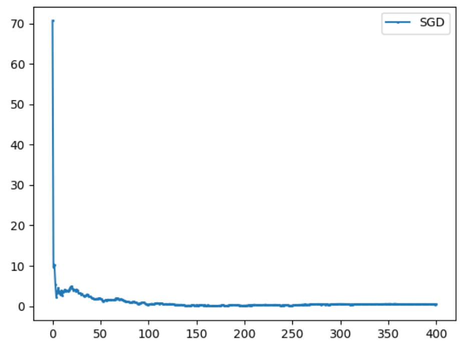
当设置$\alpha_k = 0.005$时，图(c)表明随着采样次数增加，估计值w_k越来越接近$\mathbb{E}(X)$，图(d)中误差随着采样次数增加逐步下降；但是收敛速度较慢，在给定采样次数下(K=400)，未能到达$\mathbb{E}(X)$所在区域，误差仍然较大，根据测试增加采样次数能减小误差、提高估计准确率。

该结果表明，$\alpha_k = 0.005$时，收敛速度远小于$\alpha_k = 1/k$。

c) $\alpha_k = \tanh(k)/k$



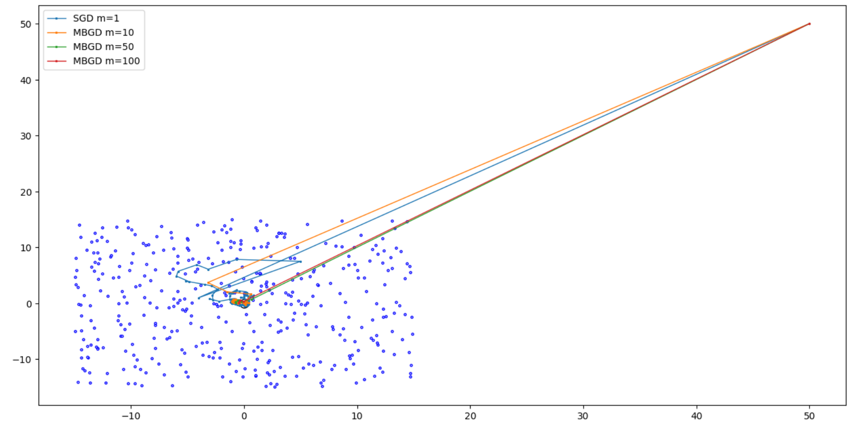
(e) 求解收敛轨迹



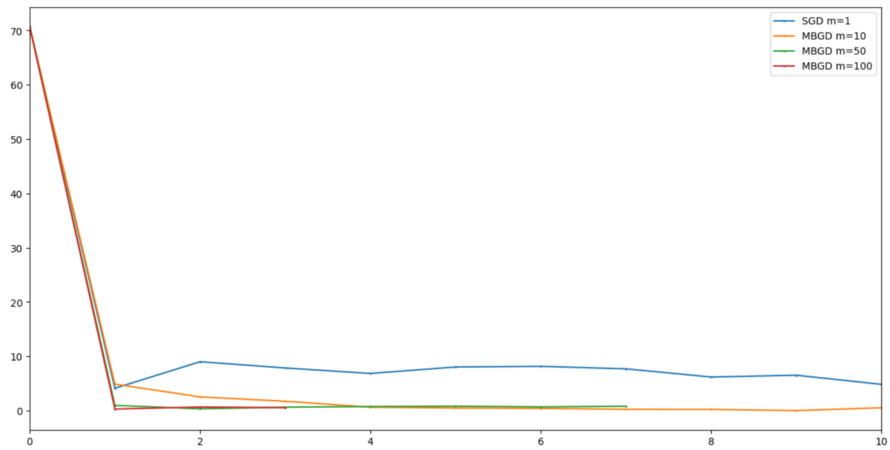
(f) 误差

当$\alpha_k = \tanh(k)/k$时，估计值w_k变化情况与$\alpha_k = 1/k$类似，均能在较短时间内较好的估计$\mathbb{E}(X)$，误差迅速接近 0。同时根据$\tanh(x)$函数定义，$\tanh(x) < 1$，因此$\frac{\tanh(k)}{k} < \frac{1}{k}$，且当 k 较小时，差异更大，因此在采样初期，$\alpha_k = 1/k$跨度更大，但偏差也更大，从图(a)与图(e)对比也可以得出相同结论；当 k 增加，两种α_k均接近 0，差别较小，且此时估计$\mathbb{E}(X)$较好，误差接近 0.

2. 小批量梯度下降，$\alpha_k = 1/k$，m = 1,10,50,100



(g) 不同梯度下降算法均值估计求解收敛轨迹



(h) 不同梯度下降算法均值估计误差

当 m=1 时，MBGD 与 SGD 相同，从图(g)中收敛轨迹可以看出，m 越大，轨迹越容易到达$\mathbb{E}(X)$所在区域，当 m=1 时，轨迹漂移更大；图(h)也展现出来相同特点，在采样次数相同时，m 越大，估计误差越小，与理论相符合，m 越大时，估计的$\frac{1}{n}\sum_{i=1}^{m}(x_i)$越接近$\overline{x}$，MBGD 的估计值比 SGD 更快地接近均值。

第二章 Q-Learning (off-policy) 算法

2.1 设计思想

1. 网格环境定义

网络大小：NUM_STATES = GRID_ROW * GRID_COL

网格结构：使用一维向量 grid 定义网格世界，包含不同的状态和对应的奖励。状态可以是普通状态、禁区、目标或边界。

奖励设置：

OTHERSTEP：普通状态的奖励。

FORBIDDEN：禁区的惩罚。

TARGET：目标状态的奖励。

BORDER：越界惩罚。

设置 gamma。

设置 EPSILON。

2. 动作定义

动作数 NUM_ACTION = 5。

定义五种可能的动作：RIGHT、DOWN、UP、LEFT、STAY。每个动作对应一个整数值。

3. 初始化策略 policy，状态值 V

policy：采用的策略，定义为大小[NUM_STATES, NUM_ACTION]的矩阵，其中 policy[s][a]表示在状态 s 时，选择动作 a 的概率，由 epsilon 决定。

V：状态值向量，长度与状态个数相同，初始化为$v_i = 0$。

4. Q-learning off-policy

初始化动作值 Q(NUM_STATES,ACTION)。初始化目标策略 policy_t

使用行为策略 policy_b 随机生成一条路径 episode，对于 episode 中每一步：

得到当前状态 state，动作 action，获得奖励 reward，下一个状态 next_state，根据$q_{t+1}(s_t,a_t) = q_t(s_t,a_t) - \alpha_t(s_t,a_t)[q_t(s_t,a_t) - (r_{t+1} + \gamma \max_a q_t(s_{t+1},a))]$更新动作值。然后遍历每个状态，更新策略，设置 policy_t(s,best_action)=1， policy_t(s,other_action)=0。

2.2 伪码描述

初始化：对所有(s,a)进行初始猜测$q_0(s,a)$。对所有(s,a)采用行为策略$\pi_b(a|s)$。对于所有(s,a)和所有$t, \alpha_t(s,a) = \alpha > 0$。

目的：由π_b生成的经验样本中学习所有状态的最优目标策略π_T。

对于每个遵循π_b策略，生成的 episode$\{s_0, a_0, r_1, s_1, a_1, r_2 \ldots\}$，do

对于 episode 的每步 t=0,1,2,…, do

更新 q 值：

$$q_{t+1}(s_t, a_t) = q_t(s_t, a_t) - \alpha_t(s_t, a_t)[q_t(s_t, a_t) - (r_{t+1} + \gamma \max_a q_t(s_{t+1}, a))]$$

更新目标策略：

$$\pi_{T,t+1}(a|s_t) = 1 \qquad \text{如果} \quad a = \arg\max_a q_{t+1}(s_t, a)$$
$$\pi_{T,t+1}(a|s_t) = 0 \qquad \text{否则}$$

2.3 时间复杂度分析

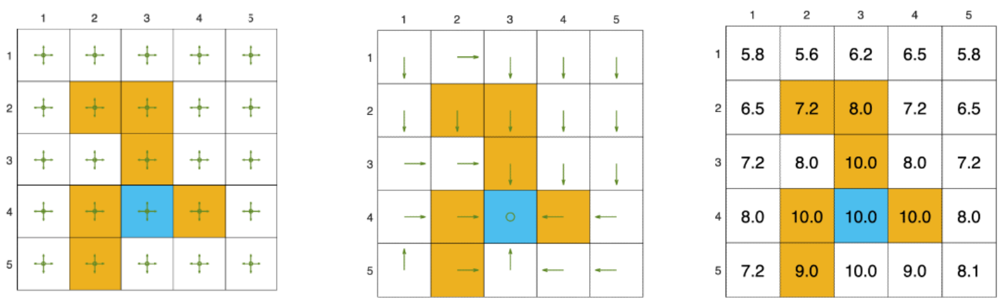
episode 长度为 T，状态个数 N，动作个数 A。

q 值更新$O(1)$，策略更新$O(N * A)$，计算 episode $O(T)$，综合时间复杂度$O(T * N * A)$。

2.4 结果分析

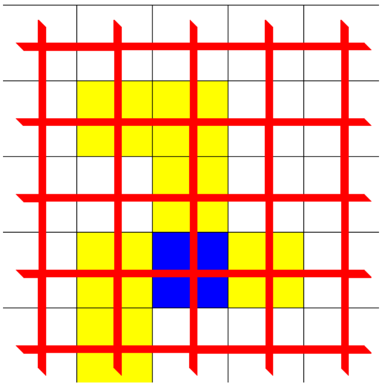
使用 5x5 网格世界，奖励设置为 $r_{boundary} = r_{forbidden} = -1$，$r_{target} = 1$，$r_{otherstep} = 0$。折扣率为$\gamma = 0.9$。 学习率为$\alpha = 0.1$。episode 长度 T=100000。

1. 行为策略$\varepsilon = 1$

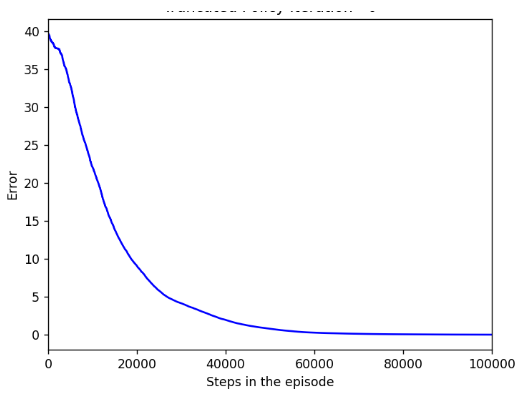


(a) 行为策略 (b) 最优策略 (c) 最优状态值

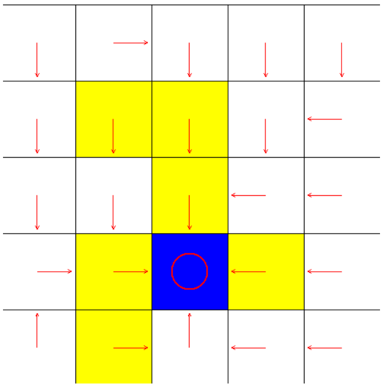
估计结果：



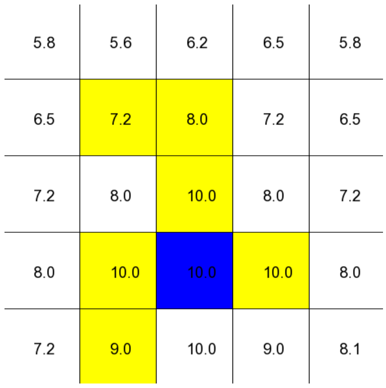
(1) episode 轨迹图



(2) 状态值误差



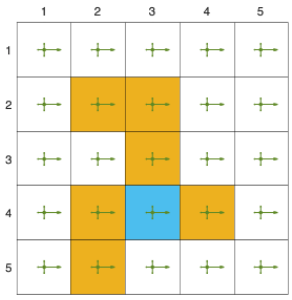
(3) 估计策略



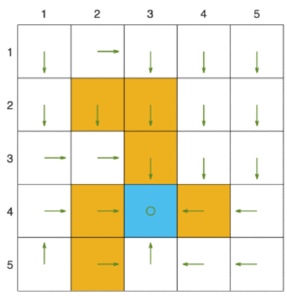
(4) 估计策略状态值

当行为策略为上述给定策略时，长度为 100000 的 episode 能将网格世界中全部动作状态对访问到，如图(1)；图(2)表示随着 episode t 的增加，估计的状态值误差逐渐下降，最终接近 0，能够实现策略估计；图(3)展示了估计的策略，与图(b)中最优策略存在一定差异，说明 Q-learning off-policy 估计会牺牲一部分最优性。

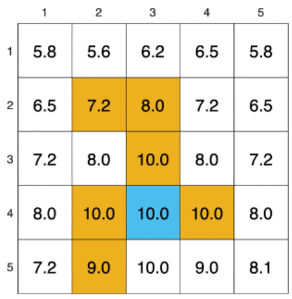
2. 行为策略$\varepsilon = 0.5$



(d) 行为策略

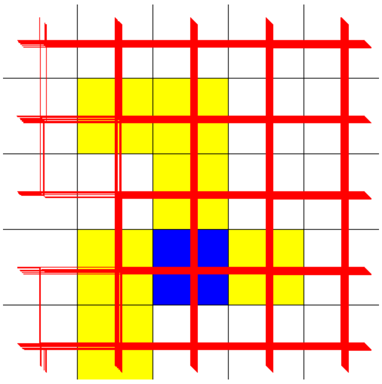


(e) 最优策略

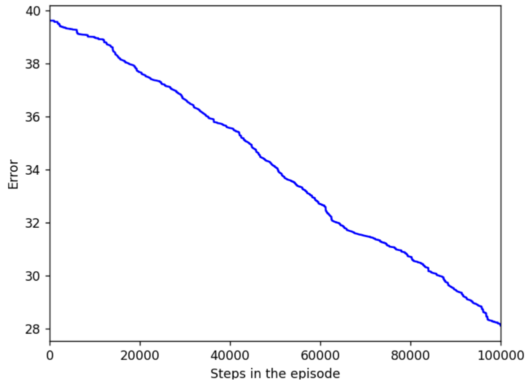


(f) 最优状态值

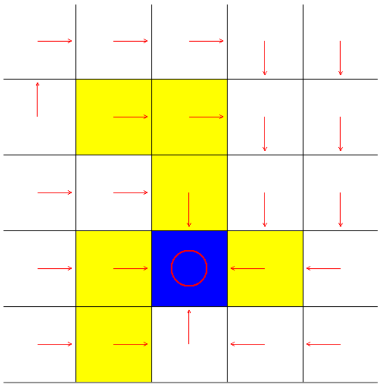
估计结果：



(5) episode 轨迹图



(6) 状态值误差



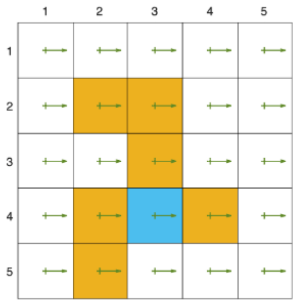
(7) 估计策略

4.7	5.2	5.8	6.5	5.8
4.2	4.8	6.5	7.2	6.5
7.2	8.0	10.0	8.0	7.2
8.0	10.0	10.0	10.0	8.0
7.1	9.0	10.0	9.0	8.1

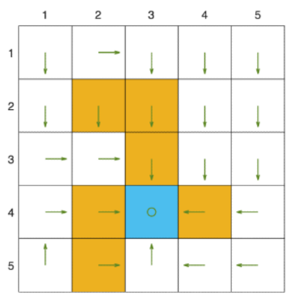
(8) 估计策略状态值

在图(d)的行为策略下，长度为 100000 的 episode 无法覆盖全部(s,a)对，如图(5)，当估计结束，状态误差仍然较大，估计策略、估计状态值与最优策略有一定差异，该行为策略估计表现比图(a)中策略差。

3. 行为策略$\varepsilon = 0.1$



(g) 行为策略

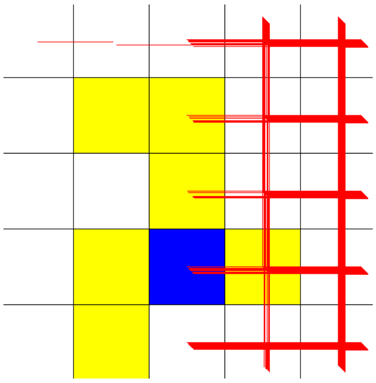


(h) 最优策略

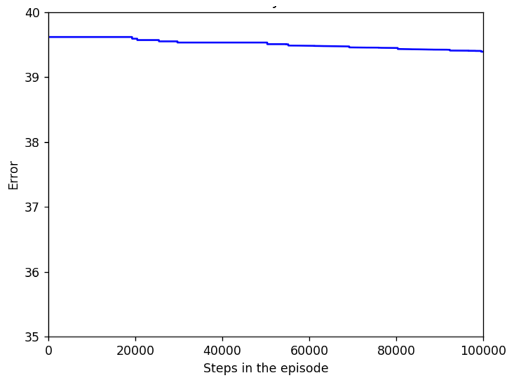
	1	2	3	4	5
1	5.8	5.6	6.2	6.5	5.8
2	6.5	7.2	8.0	7.2	6.5
3	7.2	8.0	10.0	8.0	7.2
4	8.0	10.0	10.0	10.0	8.0
5	7.2	9.0	10.0	9.0	8.1

(i) 最优状态值

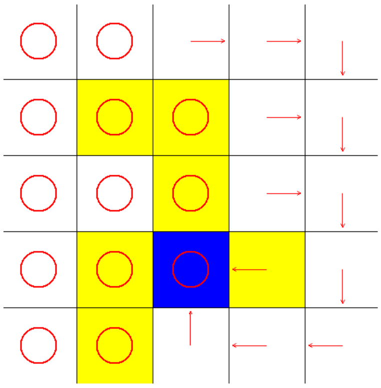
估计结果：



(9) episode 轨迹图



(10)状态值误差



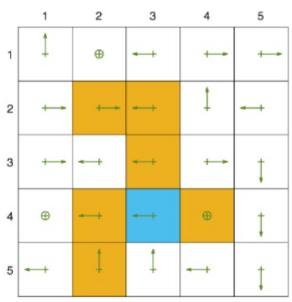
(11)估计策略

0.0	0.0	4.3	4.8	5.3
0.0	-10.0	-10.0	5.3	5.9
0.0	0.0	-10.0	5.9	6.6
0.0	-10.0	10.0	10.0	7.3
0.0	-10.0	10.0	9.0	8.1

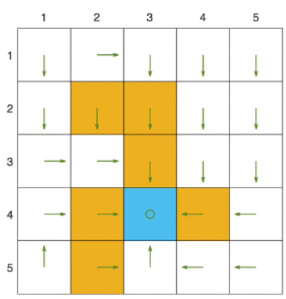
(12)估计策略状态值

使用图(g)中的行为策略，episode 轨迹图如图(9)，从结果看出覆盖的状态-动作对较少，策略探索性较差；图(10)中经过 100000 步后，状态误差值仍然较大，估计策略与估计策略状态值与最优策略相差较大；使用图(9)中行为策略不能实现对最优策略估计。

4. 行为策略$\varepsilon = 0.1$



(j) 行为策略

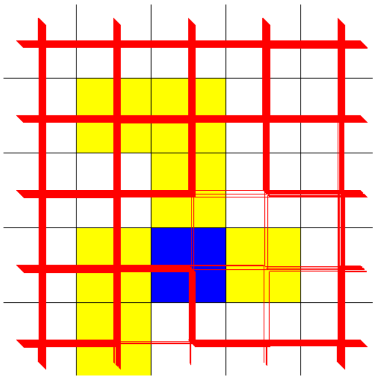


(k) 最优策略

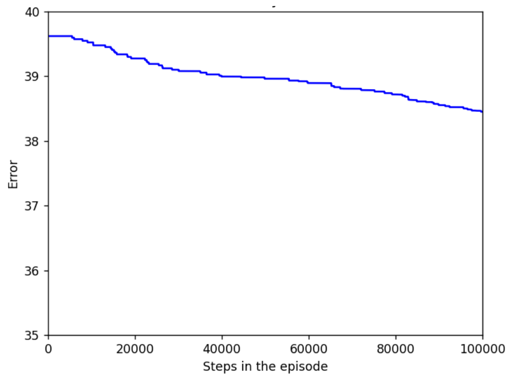
	1	2	3	4	5
1	5.8	5.6	6.2	6.5	5.8
2	6.5	7.2	8.0	7.2	6.5
3	7.2	8.0	10.0	8.0	7.2
4	8.0	10.0	10.0	10.0	8.0
5	7.2	9.0	10.0	9.0	8.1

(l) 最优状态值

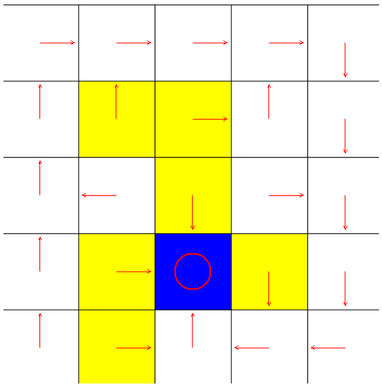
估计结果：



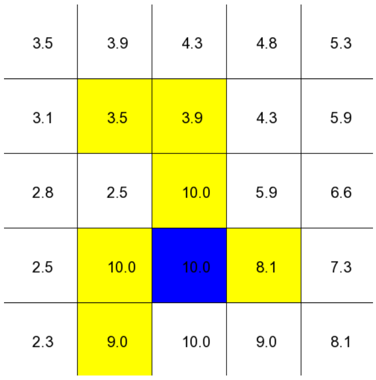
(13)episode 轨迹图



(14)状态值误差



(15)估计策略



(16)估计策略状态值

同样设置$\varepsilon = 0.1$，图(j)中行为策略相比图(g)中行为策略，随机性更强，探索性更强，反映在图(13)中可以看出，episode 轨迹图基本覆盖全部状态-动作对；在更强探索性的行为策略下，图(14)中状态值误差小于图(10)中状态值误差，估计策略与估计策略状态值也更优。

上述结果说明，行为策略π_b的探索能力对最优策略估计有较大影响，行为策略探索性越强，生成的 episode 覆盖的状态-动作对越大且分布更均匀，对最优策略的估计越好。因此在使用 Q-learning off-policy 算法时，应采用探索性更强的行为策略，以更好估计最优策略。