DS4001-25SP-HW2：搜索

常征　PB23030850

2025 年 5 月 7 日

1　问题 1：马尔可夫决策过程 [9%=6%+3%]

(a) 由于 ± 2 是吸收态，并且到了这两个状态就停止，因此以下部分我们都定义它们的 $V(s)$ 为 0。**迭代轮次 i=0**

$V^{(0)}(-2) = V^{(0)}(-1) = V^{(0)}(0) = V^{(0)}(1) = V^{(0)}(2) = \mathbf{0}$

迭代轮次 i=1

$V^{(1)}(-2) = V^{(1)}(2) = \mathbf{0}$

$V^{(1)}(-1) = max(\sum_{\hat{s}} T(1, a, \hat{s})[reward(1, a, \hat{s}) + V^{(0)}(\hat{s})]) = max\{0.2 * [(-1) + 0] + 0.1 * [(-1) + 0] + 0.7 * [10 + 0], 0.3 * [(-1) + 0] + 0.2 * [(-1) + 0] + 0.5 * [10 + 0]\} = max\{6.7, 4.5\} = \mathbf{6.7}$

$V^{(1)}(0) = max(\sum_{\hat{s}} T(0, a, \hat{s})[reward(0, a, \hat{s}) + V^{(0)}(\hat{s})]) = max\{0.2 * [(-1) + 0] + 0.1 * [(-1) + 0] + 0.7 * [(-1) + 0], 0.3 * [(-1) + 0] + 0.2 * [(-1) + 0] + 0.5 * [(-1) + 0]\} = max\{-1, -1\} = \mathbf{-1}$

$V^{(1)}(1) = max(\sum_{\hat{s}} T(1, a, \hat{s})[reward(1, a, \hat{s}) + V^{(0)}(\hat{s})]) = max\{0.2 * [(20) + 0] + 0.1 * [(-1) + 0] + 0.7 * [(-1) + 0], 0.3 * [(20) + 0] + 0.2 * [(-1) + 0] + 0.5 * [(-1) + 0]\} = max\{3.2, 5.3\} = \mathbf{5.3}$

迭代轮次 i=2

$V^{(2)}(-2) = V^{(1)}(2) = 0$

$V^{(2)}(-1) = max(\sum_{\hat{s}} T(1, a, \hat{s})[reward(1, a, \hat{s}) + V^{(1)}(\hat{s})]) = max\{0.2 * [(-1) + (-1)] + 0.1 * [(-1) + 6.7] + 0.7 * [10 + 0], 0.3 * [(-1) + (-1)] + 0.2 * [(-1) + 6.7] + 0.5 * [10 + 0]\} = max\{7.17, 5.54\} = \mathbf{7.17}$

$V^{(2)}(0) = max(\sum_{\hat{s}} T(0, a, \hat{s})[reward(0, a, \hat{s}) + V^{(1)}(\hat{s})]) = max\{0.2 * [(-1) + 5.3] + 0.1 * [(-1) + (-1)] + 0.7 * [(-1) + 6.7], 0.3 * [(-1) + 5.3] + 0.2 * [(-1) + (-1)] + 0.5 * [(-1) + 6.7]\} = max\{4.65, 3.74\} = \mathbf{4.65}$

$V^{(2)}(1) = max(\sum_{\hat{s}} T(1, a, \hat{s})[reward(1, a, \hat{s}) + V^{(1)}(\hat{s})]) = max\{0.2 * [(20) + 0] + 0.1 * [(-1) + 5.3] + 0.7 * [(-1) + (-1)], 0.3 * [(20) + 0] + 0.2 * [(-1) + 5.3] + 0.5 * [(-1) + (-1)]\} = max\{3.03, 5.86\} = \mathbf{5.86}$

(b) $(-1, a_1), (0, a_1), (1, a_2)$

2　问题 2：Q-Learning[12%=3%+6%+3%]

(a) $G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + ... = R_t + \gamma(G_{t+1})$

把上式的结果带入 $Q(s, a)$，就有：$Q(s, a) = E[G_t|s_t = s, a_t = a] = E[R_t + \gamma(G_{t+1})|s_t = s, a_t = a]$

我们将上式的期望写成:

$RHS = E[R_t|s_t = s, a_t = a] + \gamma E[G_{t+1}|s_t = s, a_t = a]$

$E[R_t|s_t = s, a_t = a]$ 是条件期望，是一个仅由 s,a 决定的数，即 $R(s, a)$。

$E[G_{t+1}|s_t = s, a_t = a] = \sum_{s' \in S} T(s, a, s')E[G_{t+1}|s_{t+1} = s'] = \sum_{s' \in S} T(s, a, s')V(s')$

那么就有 $Q(s, a) = R(s, a) + \gamma \sum T(s, a(s), s')V(s')$ 其中 $T(s, a(s), s')$ 表示初始状态为 s，采取动作 a 后转移到 s' 的概率。

(b) 我选择取折扣因子 γ 和学习率 η 都为 1。

$G_1 = 4, G_2 = 2, G_3 = -1, G_4 = 0$

首先初始化所有的 q 都为 0。

第一步之后，更新 $q(0, a_1) = G_1 = 4$

第二步之后，更新 $q(1, a_1) = G_2 = 2$

第三步之后，更新 $q(0, a_2) = G_1 = -1$

第四步之后，更新 $q(1, a_2) = 0$

(c) 能够收敛其实是基于我们对于超参数 γ 的设定, 我们要求其在 (0,1) 之间，这样下面的某个数列就会收敛 (这说明如果我们认为存在反膨胀，即未来的 1 块钱比现在的一块钱要价值更高的话就不知道要干什么了（雾)。

此外, 文章实际上只讨论了确定的情况, 即在状态 s 采取动作 a 之后会转移到确定的后继状态 s', 我尝试拓展到不确定的情况 (虽然我觉得很有道理但是不知道实际写的对不对)。文章推荐的 Foundations of Machine Learning 这本书里给出了一个更强的证明, 其中学习率 η 不是一个常数，而是一个由 s, a 决定的函数。

我们先规定一些符号的记法，避免引起歧义 (尽量和文章的符号保持一致)：$Q_t(s, a)$ $V_t'(s')$ 表示迭代 t 次后的估计值，$Q(s, a), V'(s')$ 表示最优策略的实际值，那么我们的目标就是证明前者会收敛至后者。

当 $\eta = 1$ 时，$Q_t(s, a) = r(s, a) + \gamma V_t'(s')$，而在讨论的情境下，r(s,a) 是确定的，因此

$$|Q_{t+1}(s, a) - Q(s, a)| = |\gamma V_t'(s') - \gamma V'(s'')| \tag{1}$$

在这里，我采取了不同的符号 s' 和 s”，以表示二者的取值不一定相同。

$$RHS = \gamma|(V_t'(s') - V'(s''))| = \gamma|max\{Q_t(s', a')\} - max\{Q(s'', a'')\}| \tag{2}$$

接下来我们说明这样一件事：$|max\{Q_t(s', a')\} - max\{Q(s'', a'')\}| \leq max_{s',a'}|Q_t(s', a') - Q(s', a')|$

不失一般性的, 我们假设 s', a', s'', a'' 分别取 s_1, a_1, s_2, a_2 的时候取得最大值, 以及 $max\{Q_t(s', a')\} \leq max\{Q(s'', a'')\}$，那么:

$$Q_t(s_2, a_2) \leq Q_t(s_1, a_1) \leq Q(s_2, a_2) \tag{3}$$

因此

$$|Q_t(s_2, a_2) - Q(s_2, a_2)| \geq |max\{Q_t(s', a')\} - max\{Q(s'', a'')\}| \tag{4}$$

那么就可以得到

$$max_{s',a'}|Q_t(s',a')-Q(s',a')| \geq |Q_t(s_2,a_2)-Q(s_2,a_2)| \geq |max\{Q_t(s',a')\}-max\{Q(s'',a'')\}| \quad (5)$$

上面想要证明的引理就出来了。

回到我们的 (1) 式，我们就有

$$|Q_{t+1}(s,a)-Q(s,a)| \leq \gamma\ max_{s',a'}|Q_t(s',a')-Q(s',a')| \quad (6)$$

由 s,a 的任意性 (以及 MDP) 的有限性，我们就有：

$$max_{s,a}|Q_{t+1}(s,a)-Q(s,a)| \leq \gamma\ max_{s',a'}|Q_t(s',a')-Q(s',a')| \quad (7)$$

由归纳法可知，

$$max_{s,a}|Q_{t+1}(s,a)-Q(s,a)| \leq \gamma^t\ max_{s',a'}|Q_1(s',a')-Q(s',a')| \quad (8)$$

因此当然收敛。

以上，我们证明了学习率 η 等于 1 时的收敛性。

当学习率 η 取 $(0,1]$ 时，我们有 $Q_t(s,a)=(1-\eta)Q_{t-1}(s,a)+\eta(r(s,a)+\gamma V_t'(s'))$

那么 $|Q_t(s,a)-Q(s,a)|=|(1-\eta)(Q_{t-1}(s,a)-Q(s,a))+\eta\ \gamma(V_t'(s')-V'(s'')|$

而 $|(1-\eta)(Q_{t-1}(s,a)-Q(s,a))+\eta\ \gamma(V_t'(s')-V'(s'')| \leq (1-\eta)|Q_{t-1}(s,a)-Q(s,a)|+\eta\gamma|(V_t'(s')-V'(s'')|$
后者我们在上面已经证明收敛，而前者有归纳法可以看出收敛。

3 问题 3：Gobang Programming[53%=33%+10%+10%]

(a) **[代码]**

```python
def get_next_state(self, action: Tuple[int, int, int], noise: Tuple[int, int, int]):
    # BEGIN_YOUR_CODE (our solution is 3 line of code, but don't worry if you deviate from this)
    next_state = copy.deepcopy(self.board)
    next_state[action[1]][action[2]] = action[0]
    # END_YOUR_CODE

    if noise is not None:
        white, x_white, y_white = noise
        next_state[x_white][y_white] = white
    return next_state
```

```python
def sample_noise(self):
    if self.action_space:
        # BEGIN_YOUR_CODE (our solution is 2 line of code, but don't worry if you deviate from this)
        x, y = random.choice(self.action_space)
        self.action_space.remove((x, y))
        # END_YOUR_CODE
        return 2, x, y
    else:
        return None
```

```python
def get_connection_and_reward(self, action: Tuple[int, int, int], noise: Tuple[int, int, int]):
    # BEGIN_YOUR_CODE (our solution is 4 line of code, but don't worry if you deviate from this)
    black_1, white_1 = self.count_max_connections(self.board)
    black_2, white_2 = self.count_max_connections(self.get_next_state(action, noise))
    reward = black_2**2 - white_2**2 - black_1**2 + white_1**2
    # END_YOUR_CODE

    return black_1, white_1, black_2, white_2, reward
```

```python
def sample_action_and_noise(self, eps: float):
    # BEGIN_YOUR_CODE (our solution is 8 line of code, but don't worry if you deviate from this)
    p = random.randint(0, 100)
    act = random.choice(self.action_space)
    if p > eps * 100:
        x, y = self.action_space[0]
        idx = self.array_to_hashable(self.board)
        max_q = -100
        for x, y in self.action_space:
            if idx in self.Q and (1, x, y) in self.Q[idx]:
                if self.Q[idx][(1, x, y)] > max_q:
                    max_q = self.Q[idx][(1, x, y)]
                    act = (x, y)
    self.action_space.remove(act)
    action = (1, act[0], act[1])
    # END_YOUR_CODE
    return action, self.sample_noise()
```

```python
def q_learning_update(self, s0_: np.array, action: Tuple[int, int, int], s1_: np.array, reward: float,
                          alpha_0: float = 1):
    s0, s1 = self.array_to_hashable(s0_), self.array_to_hashable(s1_)
    self.s_a_visited[(s0, action)] = 1 if (s0, action) not in self.s_a_visited else \
        self.s_a_visited[(s0, action)] + 1
    alpha = alpha_0 / self.s_a_visited[(s0, action)]

    # BEGIN_YOUR_CODE (our solution is 18 line of code, but don't worry if you deviate from this)
    gamma = self.gamma
    if s1 not in self.Q:
        self.Q[s1] = {}
    v_s1 = 0
    if self.Q[s1]!= {}:
        v_s1 = max(self.Q[s1].values())
    if s0 not in self.Q:
        self.Q[s0] = {}
    if action not in self.Q[s0]:
        self.Q[s0][action] = alpha * (reward + gamma * v_s1)
    else:
        self.Q[s0][action] = (1 - alpha) * self.Q[s0][action] + alpha * (reward + gamma * v_s1)
    # END_YOUR_CODE
```

(b) 3*3

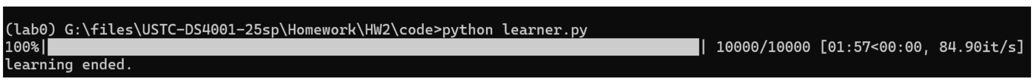
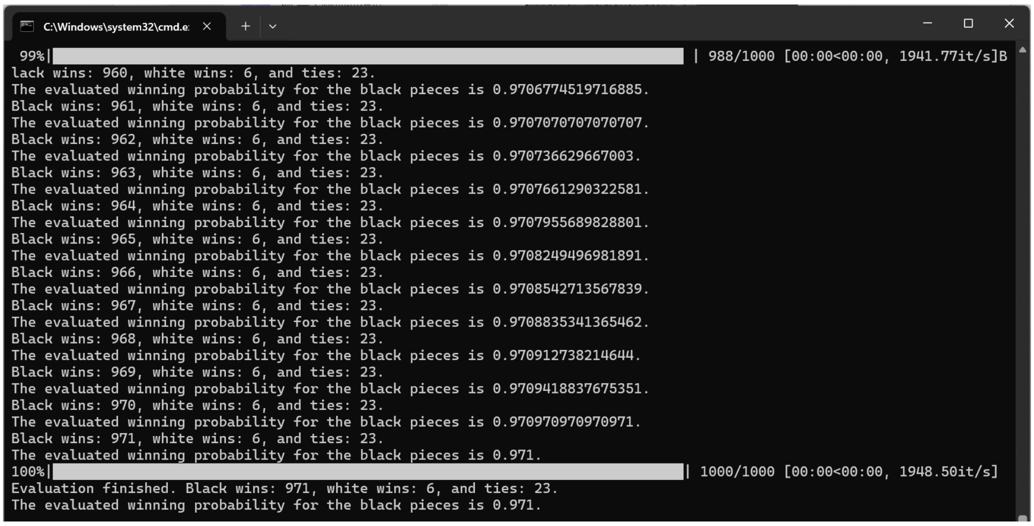


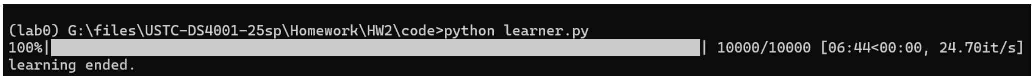
图 1: 3*3 训练结果



```
C:\Windows\system32\cmd.e
 99%|                                                                          | 988/1000 [00:00<00:00, 1941.77it/s]B
lack wins: 960, white wins: 6, and ties: 23.
The evaluated winning probability for the black pieces is 0.9706774519716885.
Black wins: 961, white wins: 6, and ties: 23.
The evaluated winning probability for the black pieces is 0.9707070707070707.
Black wins: 962, white wins: 6, and ties: 23.
The evaluated winning probability for the black pieces is 0.970736629667003.
Black wins: 963, white wins: 6, and ties: 23.
The evaluated winning probability for the black pieces is 0.9707661290322581.
Black wins: 964, white wins: 6, and ties: 23.
The evaluated winning probability for the black pieces is 0.9707955689828801.
Black wins: 965, white wins: 6, and ties: 23.
The evaluated winning probability for the black pieces is 0.9708249496981891.
Black wins: 966, white wins: 6, and ties: 23.
The evaluated winning probability for the black pieces is 0.9708542713567839.
Black wins: 967, white wins: 6, and ties: 23.
The evaluated winning probability for the black pieces is 0.9708835341365462.
Black wins: 968, white wins: 6, and ties: 23.
The evaluated winning probability for the black pieces is 0.970912738214644.
Black wins: 969, white wins: 6, and ties: 23.
The evaluated winning probability for the black pieces is 0.9709418837675351.
Black wins: 970, white wins: 6, and ties: 23.
The evaluated winning probability for the black pieces is 0.970970970970971.
Black wins: 971, white wins: 6, and ties: 23.
The evaluated winning probability for the black pieces is 0.971.
100%|                                                                          | 1000/1000 [00:00<00:00, 1948.50it/s]
Evaluation finished. Black wins: 971, white wins: 6, and ties: 23.
The evaluated winning probability for the black pieces is 0.971.
```

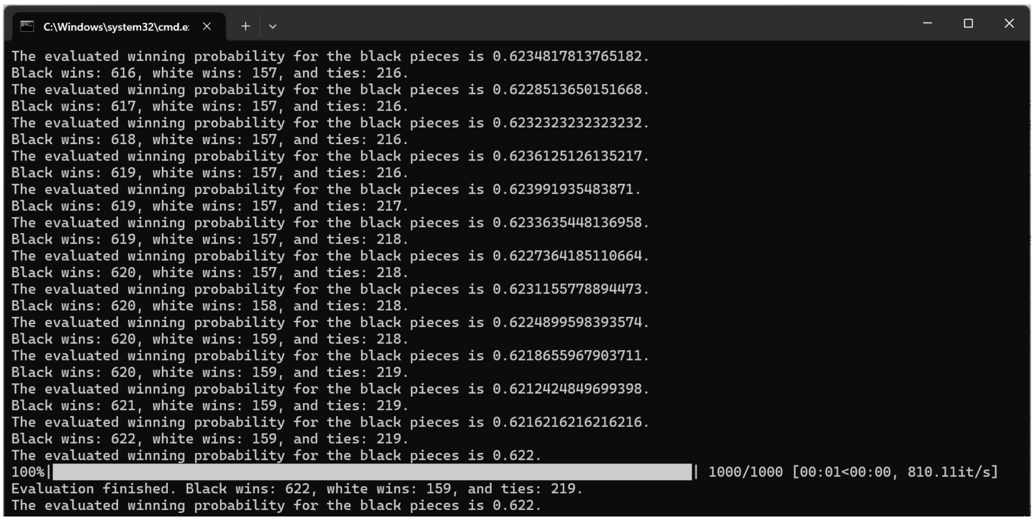
图 2: 3*3 评估结果

(c) 4*4



```
(lab0) G:\files\USTC-DS4001-25sp\Homework\HW2\code>python learner.py
100%|                                                                          | 10000/10000 [06:44<00:00, 24.70it/s]
learning ended.
```

图 3: 4*4 训练结果



```
C:\Windows\system32\cmd.e
The evaluated winning probability for the black pieces is 0.6234817813765182.
Black wins: 616, white wins: 157, and ties: 216.
The evaluated winning probability for the black pieces is 0.6228513650151668.
Black wins: 617, white wins: 157, and ties: 216.
The evaluated winning probability for the black pieces is 0.6232323232323232.
Black wins: 618, white wins: 157, and ties: 216.
The evaluated winning probability for the black pieces is 0.6236125126135217.
Black wins: 619, white wins: 157, and ties: 216.
The evaluated winning probability for the black pieces is 0.623991935483871.
Black wins: 619, white wins: 157, and ties: 217.
The evaluated winning probability for the black pieces is 0.6233635448136958.
Black wins: 619, white wins: 157, and ties: 218.
The evaluated winning probability for the black pieces is 0.6227364185110664.
Black wins: 620, white wins: 157, and ties: 218.
The evaluated winning probability for the black pieces is 0.6231155778894473.
Black wins: 620, white wins: 158, and ties: 218.
The evaluated winning probability for the black pieces is 0.6224899598393574.
Black wins: 620, white wins: 159, and ties: 218.
The evaluated winning probability for the black pieces is 0.6218655967903711.
Black wins: 620, white wins: 159, and ties: 219.
The evaluated winning probability for the black pieces is 0.6212424849699398.
Black wins: 621, white wins: 159, and ties: 219.
The evaluated winning probability for the black pieces is 0.6216216216216216.
Black wins: 622, white wins: 159, and ties: 219.
The evaluated winning probability for the black pieces is 0.622.
100%|                                                                          | 1000/1000 [00:01<00:00, 810.11it/s]
Evaluation finished. Black wins: 622, white wins: 159, and ties: 219.
The evaluated winning probability for the black pieces is 0.622.
```

图 4: 4*4 评估结果

4 问题 4：Deeper Understanding[16%=5%+5%+2%+4%]

4.1 Bellman 算子与压缩映射

(a) TODO 实际上最开始看到最大范数有点懵，不过如果用上我们在数学分析的学到的极限：

$$\lim_{n->+\infty}(a_1^n+a_2^n+...+a_m^n)^{\frac{1}{n}}=max\{a_1,a_2,...,a_m\} \tag{9}$$

那么我们需要证明的式子就变成了

$$max_{s\in S}|Tv_1(s)-Tv_2(s)|\leq\gamma\ max_{s\in S}|v_1-v_2| \tag{10}$$

这里和我们之前证明的东西很类似，实际上说明的都是迭代一次的收敛性问题，那么受到之前的启发，我们只需要证明：

$$\forall s,|Tv_1(s)-Tv_2(s)|\leq\gamma\ max_{s'\in S}|v_a(s')-v_b(s')| \tag{11}$$

而 $Tv_1(s)=max_{a\in Action}\{r_{sa}+\gamma\cdot\sum_{s'\in S}p_{sas'}\cdot v_1(s')\}$ 我们在 2(c) 的式 (5) 已经说明了一个类似的不等式，在此不再累述，总之有：

$$|Tv_1(s)-Tv_2(s)|\leq max_{a\in Action}|(r_{sa}+\gamma\cdot\sum_{s'\in S}p_{sas'}\cdot v_1(s'))-(r_{sa}+\gamma\cdot\sum_{s'\in S}p_{sas'}\cdot v_2(s'))| \tag{12}$$

而 $RHS=\gamma\cdot max_{a\in Action}|\sum_{s'\in S}p_{sas'}\cdot(v_1(s')-v_2(s'))|$

我们记新的 RHS 为 RRHS(雾)，那么直观上理解，RRHS 可以看作各个 state 的加权平均，自然小于其中的最大项。用数学的语言来表述，就是：

$$v_1(s')-v_2(s')\leq max_{s''\in S}|v_1(s')-v_2(s')| \tag{13}$$

那么 RRHS 就有：

$$RRHS\leq\gamma\cdot max_{a\in Action}\sum_{s'\in S}p_{sas'}\cdot max_{s''\in S}|v_1(s')-v_2(s')|=\gamma\cdot max_{s'\in S}|v_1(s')-v_2(s')| \tag{14}$$

(b) 我们不妨假设有两个不动点 v_1,v_2，另取一个不同的函数 v。

为了方便表示，我们记 Δ_{12} 为 $max_{s\in S}|v_1(s)-v_2(s)|$，显然 $\Delta_{12}>0$，否则就是同一个不动点。

记 Δ_1^n 为 $max_{s\in S}|T...T(v_1)(s)-T...T(v)(s)|$，即作用 n 次 Bellman 算子后二者的距离，同样的定义 Δ_2^n。

由 4.1(a) 我们可以知道，$\Delta_1^n\leq\gamma\cdot\Delta_1^{n-1}$，即单调递减收敛到 0，$\Delta_2^n$ 有类似的性质。

那么我们就一定能找到 1 个 n_1，$\Delta_1^{n_1}<\frac{1}{2}\Delta_{12}$，一定能找到 1 个 n_2，$\Delta_2^{n_2}<\frac{1}{2}\Delta_{12}$.

取 $n=\max\{n_1,n_2\}$ 由三角不等式，$\Delta_1^n+\Delta_2^n\geq\Delta_{12}^n$，那么 $\Delta_{12}>\Delta_{12}^n$。

但是由不动点的性质，$T(v_1)(s)=v_1(s)$, 因此 $\Delta_{12}=\Delta_{12}^n$, 得到矛盾。

因此，只存在一个不动点。

4.2 重要性采样

(a) p 分布下 $f(x)$ 的期望：

$$E_{x\sim p}[f(x)] = \int f(x)p(x)dx \tag{15}$$

而对于右式，有：

$$RHS = \int (\frac{p(x)}{q(x)} f(x))q(x)dx = \int f(x)p(x)dx = LHS \tag{16}$$

(b) 前面我们已经证明二者的期望相同，那么自然期望的平方也相同。我们只需要证明：

$$E_{x\sim p}[f^2(x)] - E_{x\sim q}[(\frac{p(x)}{q(x)})^2 f^2(x)] = RHS \tag{17}$$

即证明

$$E_{x\sim q}[(\frac{p(x)}{q(x)})^2 f^2(x)] = E_{x\sim p}[\frac{p(x)}{q(x)} f^2(x)] \tag{18}$$

虽然可以再次推导一次，不过用 a 中的结果就可以得到。

体验反馈 [10%]

(a) [**必做**] 6 小时

(b) [**选做**] 好多好多数学