南開大學
Nankai University

人工智能学院

强化学习实验报告

实验二　动作选择策略

姓名：石若川

学号：2111381

专业：智能科学与技术

2024 年 3 月 12 日

1 实验目的

1. 学习 $\epsilon-greedy$、UCB、玻尔兹曼三种动作选择策略的基本原理
2. 使用三种动作选择策略训练智能体进行四子棋对弈
3. 对比三种动作选择策略的优劣

2 实验原理

$\epsilon-greedy$ 策略的学习公式如下

$$A_t = \begin{cases} \arg\max_a Q_t(a) & \text{with probability } 1-\varepsilon \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

其中 ϵ 为大于 0 的常数，用于平衡利用和试探。

基于置信度上界（Upper-Confidence-Bound, UCB）策略的公式如下

$$A_t = \arg\max_a [Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}]$$

其中，$N_t(a)$ 表示在时刻 t 之前动作 a 被选择的次数。c 是一个大于 0 的常数，控制试探的程度。UCB 策略的思想是，平方根项是对 a 动作值估计的不确定性或方差的度量。因此，最大值的大小是动作 a 的可能真实值的上限，参数 c 决定了置信水平。每次选 a 时，不确定性可能会减小；由于 $N_t(a)$ 出现在不确定项的分母上，因此随着 $N_t(a)$ 的增加，这一项就减小了。另一方面，每次选择 a 之外的动作时，在分子上的 t 增大，而 $N_t(a)$ 却没有变化，所以不确定性增加了。自然对数的使用意味着随着时间的推移，增加会变得越来越小，但它是无限的。所有动作最终都将被选中，但是随着时间的流逝，具有较低价值估计的动作或者已经被选择了更多次的动作被选择的频率较低。

基于玻尔兹曼的策略公式如下

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

其中，$\pi_t(a)$ 用来表示动作 a 在时刻 t 时被选择的概率，$H_t(t) = \frac{Q_t(a)}{\tau}$，$\tau$ 为大于 0 的常数。

3 部分代码

在实验一代码的基础上，主要修改的为 Player 类。在初始化时，增加了 policy、N 和 kwargs 等成员变量。self.policy 用于记录使用的动作选择策略，N 对应于 UCB 策略中的 $N_t(a)$，用于记录棋盘每个位置被选择的次数，kwargs 用于记录 ϵ、c 和 τ 参数。

```
def __init__(self, step_size=0.1, epochs=int(1e5), policy='e-greedy', **kwargs):
        self.estimations = dict()
        self.step_size = step_size
        self.policy = policy
        self.states = []
```

```
        # self.greedy = []
        self.symbol = 0
        self.epochs = epochs
        self.kwargs = kwargs
        self.N = np.zeros((4, 4))
```

在 act 函数中，对三种策略进行了代码实现，并且当 policy 无效时，弹出错误。

```
def act(self):
        state = self.states[-1]
        next_states = []
        next_positions = []
        for i in range(BOARD_ROWS):
            for j in range(BOARD_COLS):
                if state.data[i, j] == 0:
                    next_positions.append([i, j])
                    next_states.append(state.next_state(
                        i, j, self.symbol).hash())

        if self.policy == 'e-greedy':
            epsilon = self.kwargs['epsilon']
            if np.random.rand() < epsilon:
                action = next_positions[np.random.randint(len(next_positions))]
                action.append(self.symbol)
                # self.greedy[-1] = False
                return action

            values = []
            for hash_val, pos in zip(next_states, next_positions):
                values.append((self.estimations[hash_val], pos))
            np.random.shuffle(values)
            values.sort(key=lambda x: x[0], reverse=True)
            action = values[0][1]
            action.append(self.symbol)
            return action

        elif self.policy == 'ucb':
            values = []
            c_ratio = self.kwargs['c_ratio']
            for hash_val, pos in zip(next_states, next_positions):
                visits = self.N[pos[0], pos[1]]
                # print(visits)
                exploration_bonus = c_ratio * np.sqrt(np.log(len(self.states) + 1) /
```

```
                (visits + 1e-5))
            values.append((self.estimations.get(hash_val, 0) + exploration_bonus,
                pos))

        values.sort(key=lambda x: x[0], reverse=True)
        action = values[0][1]
        action.append(self.symbol)
        self.N[action[0], action[1]] = self.N[action[0], action[1]] + 1
        return action

    elif self.policy == 'boltzmann':
        tau = self.kwargs['temperature']
        probabilities = [np.exp(self.estimations[hash_val] / tau) for hash_val in
            next_states]
        probabilities /= sum(probabilities)
        selected_index = np.random.choice(len(next_positions), p=probabilities)
        action = next_positions[selected_index]
        action.append(self.symbol)
        return action

    else:
        raise ValueError("Invalid strategy. Choose from 'epsilon-greedy', 'ucb',
            or 'boltzmann'")
```

4 实验结果

实验中，以三种动作选择策略分别训练了 $10^3, 10^4, 10^5$ 轮，并互相对弈，结果如表1、2和3所示。可以发现，对于各策略：

- 随着迭代次数增加，策略的获胜概率增加。这说明随迭代次数增加，可以训练得到更好的对弈策略。增加迭代次数意味着智能体有更多的机会学习并优化其策略。然而，这并不意味着能够无限制地提高性能，而是只能逼近最优策略。
- 当对弈策略不变，交换先后手顺序时，原先后手的一方，获胜概率增加。这说明先手具有一定先发优势。如果两名玩家都采用最佳的策略，那么如果双方都不犯错误，游戏将以平局结束。但在实际游戏中，如果双方没有采用最佳策略或者一方犯了错误，先发方可能会利用这些机会取得胜利。

表 1: $\epsilon - greedy$ 策略的对弈结果（先手胜/后手胜/平局）

后手 \ 先手	10^3	10^4	10^5
10^3	0.7/0/0.3	0.5/0.3/0.2	1/0/0
10^4	0.7/0.3/0	0.5/0.3/0.2	0.8/0/0.2
10^5	0.3/0.7/0	0.2/0.7/0.1	0.3/0.7/0

表 2: UCB 策略的对弈结果（先手胜/后手胜/平局）

后手 \ 先手	10^3	10^4	10^5
10^3	0.4/0.4/0.2	0.1/0.1/0.8	0.6/0.4/0
10^4	0.5/0.3/0.2	0.6/0.4/0	0.7/0.2/0.1
10^5	0.7/0.3/0	0.1/0.1/0.8	0.7/0.3/0

表 3: 玻尔兹曼策略的对弈结果（先手胜/后手胜/平局）

后手 \ 先手	10^3	10^4	10^5
10^3	0.4/0.3/0.3	0.5/0.1/0.4	0.3/0.4/0.3
10^4	0.2/0.7/0.1	0.2/0.3/0.5	0.4/0.4/0.2
10^5	0.2/0.3/0.5	0.1/0.2/0.7	0.3/0.4/0.3

另外，实验中分别利用训练 10^5 的三种动作选择策略互相对弈，结果如表4。可以发现，$\epsilon - greedy$ 的表现最优，UCB 其次，玻尔兹曼策略最差。实验中展示了 $\epsilon - greedy$ 分别与 UCB 和玻尔兹曼策略的对弈过程，如图4.1和4.2。

表 4: 各策略间的对弈结果（先手胜/后手胜/平局）

后手 \ 先手	$\epsilon - greedy$	UCB	Boltzmann
$\epsilon - greedy$	0.1/0.3/0.6	0.2/0.7/0.1	0/0.8/0.2
UCB	0.7/0.2/0.1	0.7/0.3/0	0.2/0.6/0.2
Boltzmann	0.8/0.1/0.1	0.8/0.1/0.1	0.7/0/0.3

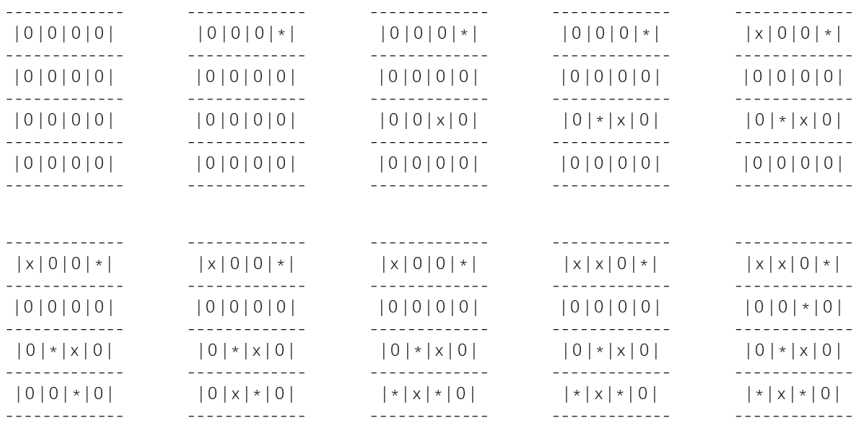

图 4.1: $\epsilon - greedy$ 与 UCB 策略的某一次对弈过程

```
-------------  -------------  -------------  -------------
|0|0|0|0|      |0|0|0|*|      |0|0|0|*|      |0|0|0|*|
-------------  -------------  -------------  -------------
|0|0|0|0|      |0|0|0|0|      |0|0|0|0|      |0|0|0|0|
-------------  -------------  -------------  -------------
|0|0|0|0|      |0|0|0|0|      |0|0|x|0|      |0|*|x|0|
-------------  -------------  -------------  -------------
|0|0|0|0|      |0|0|0|0|      |0|0|0|0|      |0|0|0|0|
-------------  -------------  -------------  -------------

-------------  -------------  -------------  -------------
|0|0|0|*|      |0|0|0|*|      |0|0|0|*|      |0|0|0|*|
-------------  -------------  -------------  -------------
|0|0|0|0|      |0|0|*|0|      |0|0|*|0|      |0|0|*|0|
-------------  -------------  -------------  -------------
|0|*|x|x|      |0|*|x|x|      |0|*|x|x|      |0|*|x|x|
-------------  -------------  -------------  -------------
|0|0|0|0|      |0|0|0|0|      |0|x|0|0|      |*|x|0|0|
-------------  -------------  -------------  -------------
```

图 4.2: $\epsilon - greedy$ 与玻尔兹曼策略的某一次对弈过程

实验中绘制了三种动作选择策略进行训练时，先后手胜率的变化曲线，如图4.3所示。$\epsilon - greedy$ 策略和玻尔兹曼策略随迭代轮数增加，均稳定在某个值附近。而使用 ucb 策略时，胜率接近 0 后可能会出现重新上升的情况。

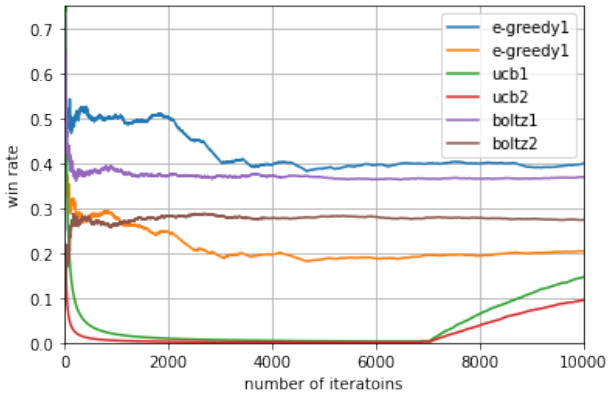

图 4.3: 三种策略训练过程中先后手的胜率曲线

为了探究这一现象的原因，实验中改变 UCB 策略的 c 值，观察先手的胜率曲线。如图4.4所示，当 c 值增大时，曲线出现上升的轮数越大，曲线上升的幅度越小。

以下为这一现象的可能原因。曲线出现上升可能是由于训练过程中出现了未探索的全新状态，类似逃离局部最优解的情况，此时策略会出现较大幅度的变化。而 c 值控制了 UCB 策略探索的程度。c 值越大，在训练初期探索的状态空间越大，未探索的状态较少，所以会在训练过程中更靠后的位置出现未探索状态。

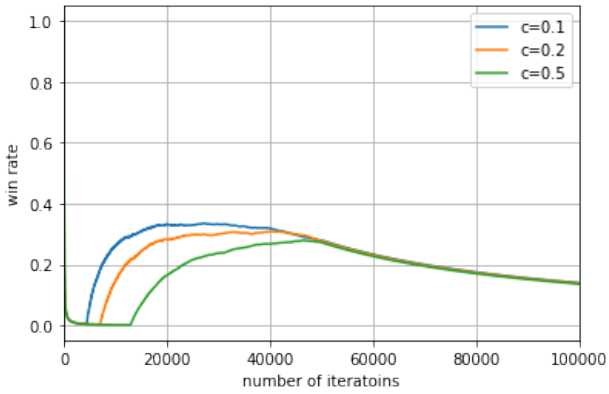

图 4.4: 不同 c 值训练过程中先手的胜率曲线

5　实验总结

通过本次实验，我学习了 $\epsilon - greedy$、USB 和玻尔兹曼动作选择策略的基本原理，并编写了对应策略的代码，利用三种策略训练智能体进行四子棋对弈。

通过对比这三种策略的对弈结果可以发现，$\epsilon - greedy$ 策略的表现最好，UCB 次之，玻尔兹曼表现最差。另外实验中发现，UCB 策略在训练过程中会出现胜率曲线突然上升的情况。通过改变 UCB 策略的 c 值，实验发现这一现象可能与 c 值的选择有关。c 值增大时，曲线出现上升的轮数越大，曲线上升的幅度越小。这可能是由于 c 值控制了 UCB 策略探索的程度。c 值越大，在训练初期探索的状态空间越大，未探索的状态较少，所以会在训练过程中更靠后的位置出现未探索状态。