AI Assignment 4

1. Giới thiệu Value Iteration

Value Iteration là một phương pháp tính toán tối ưu MDP policy và cho giá trị của nó.Value Iteration hoạt động từ cuối và hoạt động bằng cách tính ngược lại các giá trị ở đầu,tinh chỉnh giá trị của Q* hoặc V*. Vì bài toán trên không có điểm cuối thật sự, nên ta sẽ thành lập một điểm endpoint tùy chọn, V_k là một value function có k trạng thái để đi,Q_K có k trạng thái để đi. Chúng có thể được định nghĩa một cách đệ quy. Phép lặp giá trị bắt đầu với một hàm tùy ý V_0 và sử dụng các phương trình sau để nhận các hàm cho k + 1 trạng thái đi từ các hàm cho k trạng thái đi:

$$Q_{k+1}(s,a) = \sum_{s'} P(s'|s,a)\,(R(s,a,s') + \gamma V_k(s'))\quad \text{for } k \geq 0$$

$$V_k(s) = \max_a Q_k(s,a)\quad \text{for } k>0.$$

Nó có thể lưu mảng V [S] hoặc mảng Q [S, A]. Lưu mảng V dẫn đến việc lưu trữ ít hơn, nhưng khó xác định một optimal action hơn và cần thêm một lần lặp để xác định hành động nào dẫn đến max value.

Evaluate an optimal policy

1. Initialize $V_0^*(s) \leftarrow 0$ for all states s.
2. Repeat until convergence: $t = 1, 2, 3, \ldots$
   // While $\max_s(|V_t^*(s) - V_{t-1}^*(s)|) > \varepsilon \approx 0.00001$
   - For each state s:

$$\underbrace{V_t^*(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_{t-1}^*(s')]}_{Q_{t-1}^*(s,a)}$$

2. Kết quả thực nghiệm của Value Iteration

1. FrozenLake

Thực nghiệm cho thấy việc thực hiện value iteration của **FrozenLake-v0** hội tụ ở iteration thứ 79 (chạy với nhiều map FrozenLake-v0) và của **FrozenLake8x8-v0** là ở iteration thứ 117.Đối với bài toán FrozenLake,ta sẽ thống kê bằng số lượng success trên 1000 ván(đo 50 lần 1000 ván) và xem phân phối của các lần đó.

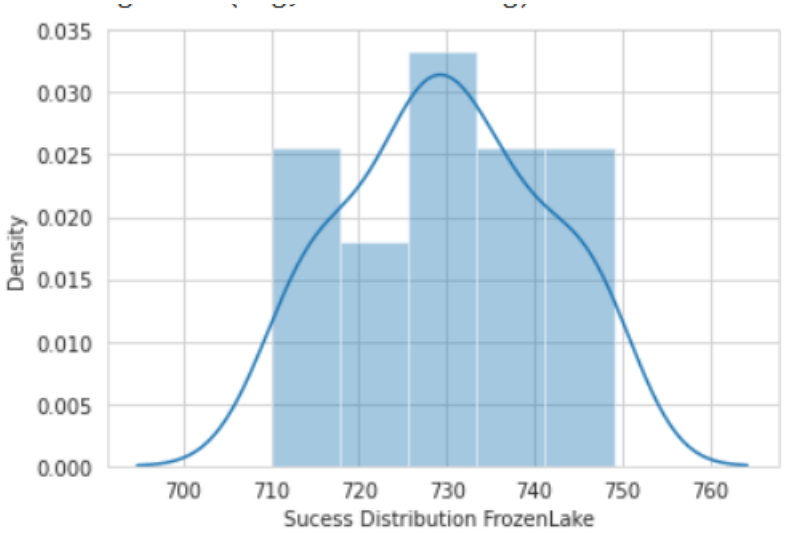
```
arr_Success=[]
for i in range(50):
  env = gym.make('FrozenLake-v0')#load môi trường

  v_values,Converge = value_iteration(env, max_iters=1000, gamma=0.9)
  env.reset()
  policy = policy_extraction(env, v_values, gamma=0.9)

  success,score=play_multiple_times(env, policy, 1000)
  arr_Success.append(success)
```

```
Converged at 79-th iteration.
Number of successes: 727/1000
mean score: 1.0
Converged at 79-th iteration.
Number of successes: 710/1000
mean score: 1.0
Converged at 79-th iteration.
Number of successes: 730/1000
```

Ta thấy rằng kết quả success tập trung chủ yêu ở 730 là nhiu nhất và tập trung ở khoảng (710-750)



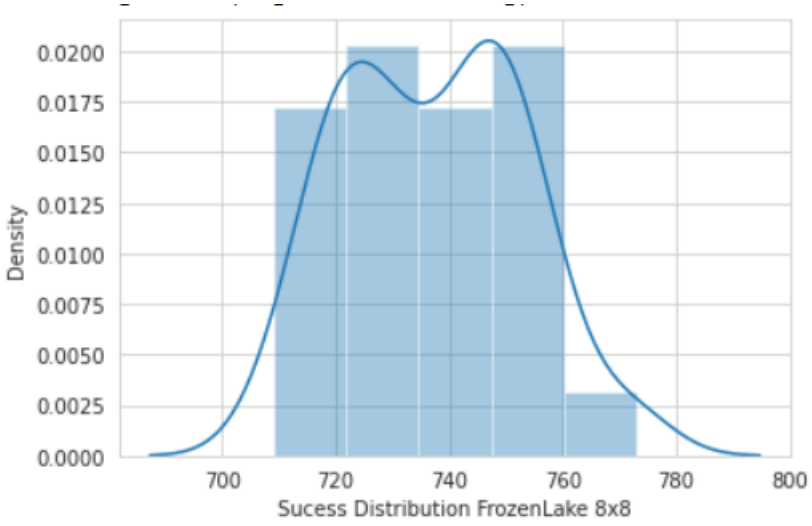
```python
arr_Success=[]
for i in range(50):
  env = gym.make('FrozenLake8x8-v0')#load môi trường

  v_values,Converge = value_iteration(env, max_iters=1000, gamma=0.9)
  env.reset()
  policy = policy_extraction(env, v_values, gamma=0.9)

  success,score=play_multiple_times(env, policy, 1000)
  arr_Success.append(success)
```

```
Converged at 117-th iteration.
Number of successes: 734/1000
mean score: 1.0
Converged at 117-th iteration.
Number of successes: 754/1000
mean score: 1.0
Converged at 117-th iteration.
Number of successes: 720/1000
mean score: 1.0
Converged at 117-th iteration.
Number of successes: 745/1000
mean score: 1.0
Converged at 117-th iteration.
Number of successes: 723/1000
mean score: 1.0
Converged at 117-th iteration.
```

Kết quả tập trung ở khoảng 710-760 là nhìu nhất(nhìu nhất là 720-730 và 750-760)



2. Taxi v3

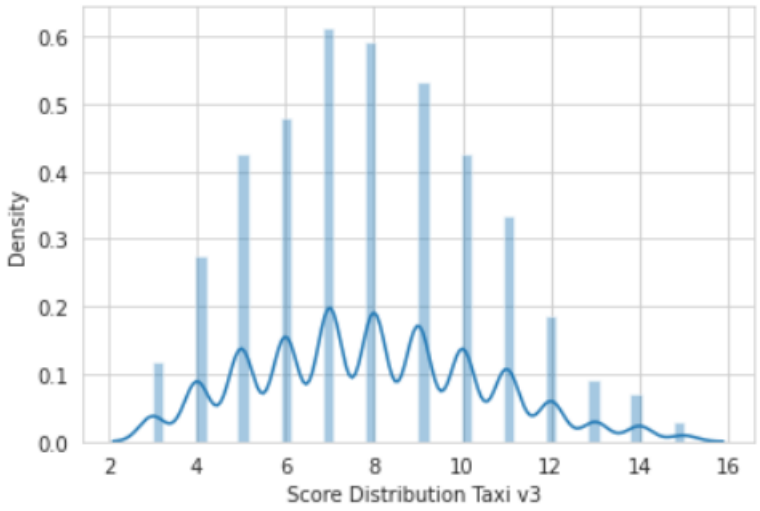
Thực nghiệm cho ta thấy với taxi v3 thì value iteration hội tụ ở iteration thứ 116. Với Taxi v3 ta sẽ không thấy kê bằng số ván thắng mà bằng số điểm thu được qua các lần chơi(vì mỗi lần chơi đều sẽ ghi được điểm >0 => luôn thắng và có sự khác biệt về điểm giữa các lần chơi,kết quả được đo ở 50*1000=500000 lần chạy

```python
arr_Score=[]
for i in range(50):
  env = gym.make('Taxi-v3')#load môi trường

  v_values,Converge = value_iteration(env, max_iters=1000, gamma=0.9)
  env.reset()
  policy = policy_extraction(env, v_values, gamma=0.9)

  success,score=play_multiple_times(env, policy, 1000)
  arr_Score.append(success)
```

```
Converged at 116-th iteration.
Number of successes: 1000/1000
mean score: 7.986
Converged at 116-th iteration.
Number of successes: 1000/1000
mean score: 7.943
Converged at 116-th iteration.
Number of successes: 1000/1000
mean score: 7.891
Converged at 116-th iteration.
Number of successes: 1000/1000
mean score: 7.927
Converged at 116-th iteration.
Number of successes: 1000/1000
mean score: 7.971
Converged at 116-th iteration.
Number of successes: 1000/1000
mean score: 7.808
Converged at 116-th iteration.
Number of successes: 1000/1000
mean score: 7.956
```



3. Giới thiệu Policy Iteration

Policy Iteration là một phương pháp tối ưu policy dựa trên các action và state trước đó. Giả sử chúng ta có một policy (π: S → A) chỉ định một action cho state. Các action sẽ được chọn mỗi khi hệ thống ở state s.

Ý tưởng Policy Iteration

Đánh giá một policy nhất định (ví dụ: tự ý khởi tạo policy cho tất cả các state ∈ S)

bằng cách tính toán value function cho tất cả các state ∈ S theo policy nhất định

$$V_\pi(s) = \mathrm{E}\left[R(s, \pi(s), s') + \gamma V(s')\right]$$

Value function = kì vọng giá trị nhận được ở step đầu+ cộng với discount ở state kế tiếp

Improve policy : tìm một action tốt hơn với state s ∈ S

$$\pi_1(s) = \arg\max_{a \in \mathcal{A}} \mathrm{E}\left[R(s, a, s') + \gamma V(s')\right]$$

Lặp lại step 1,2 cho tới khi value function hội tụ tại optimal value function

4. Kết quả thực nghiệm Policy Iteration

Tương tự như việc thực nghiệm của Value Iteration ta cũng có một số kết quả thực nghiệm như sau:

1. Frozen Laze

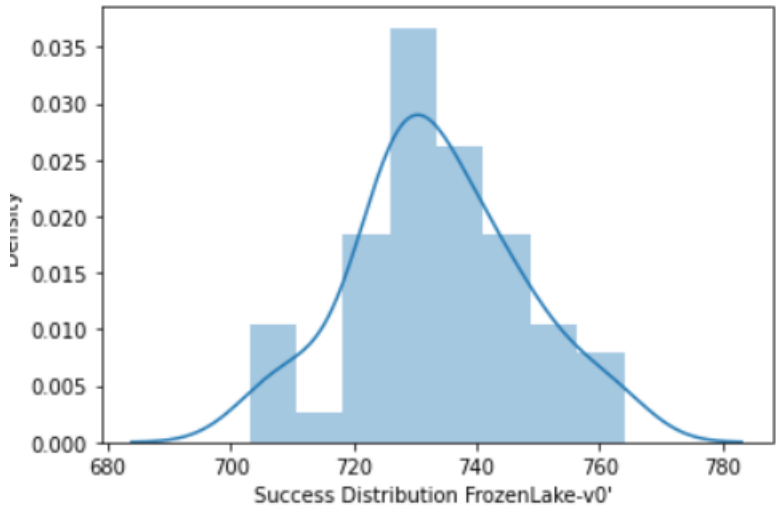
Với map này ta thấy rằng kết quả thực nghiệm Policy Iteration của Frozen hội tụ ở iteration thứ 5 và số lượng success sẽ rơi vào khoảng 730-735 rất nhiều
Tương tự ở map Frozen 8*8 thì kết quả sẽ hội tụ ở Iteration thứ 9.

```
arr_Success=[]

for i in range(50):
  env = gym.make('FrozenLake-v0')#load môi trường

  value_function, policy = policy_iteration(env)
  print(policy)
  success,score=play_multiple_times(env, policy, 1000)
  arr_Success.append(success)
arr_Success=np.array(arr_Success)
x = pd.Series(arr_Success, name="Success Distribution Taxi v3 ")
ax = sns.distplot(x)
```

```
Policy Iteration converged at 5-th.
[0 3 0 3 0 0 0 0 3 1 0 0 0 2 1 0]
Number of successes: 719/1000
mean score: 1.0
Policy Iteration converged at 5-th.
[0 3 0 3 0 0 0 0 3 1 0 0 0 2 1 0]
Number of successes: 730/1000
mean score: 1.0
Policy Iteration converged at 5-th.
[0 3 0 3 0 0 0 0 3 1 0 0 0 2 1 0]
```

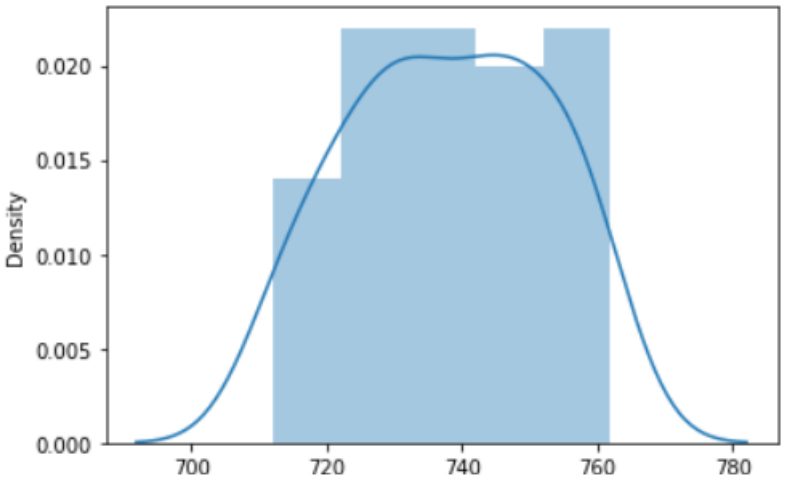


```
arr_Success=[]

for i in range(50):
  env = gym.make('FrozenLake8x8-v0')#load môi trường

  value_function, policy = policy_iteration(env)
  print(policy)
  success,score=play_multiple_times(env, policy, 1000)
  arr_Success.append(success)
arr_Success=np.array(arr_Success)
x = pd.Series(arr_Success, name="Success Distribution FrozenLake-v0' ")
ax = sns.distplot(x)
```

```
Policy Iteration converged at 9-th.
[3 2 2 2 2 2 2 2 3 3 3 3 2 2 2 1 3 3 0 0 2 3 2 1 3 3 3 1 0 0 2 1 3 3 0 0 2
 1 3 2 0 0 0 1 3 0 0 2 0 0 1 0 0 0 0 2 0 1 0 0 1 1 1 0]
Number of successes: 728/1000
mean score: 1.0
Policy Iteration converged at 9-th.
[3 2 2 2 2 2 2 2 3 3 3 3 2 2 2 1 3 3 0 0 2 3 2 1 3 3 3 1 0 0 2 1 3 3 0 0 2
 1 3 2 0 0 0 1 3 0 0 2 0 0 1 0 0 0 0 2 0 1 0 0 1 1 1 0]
Number of successes: 746/1000
mean score: 1.0
```



2. Taxi v3

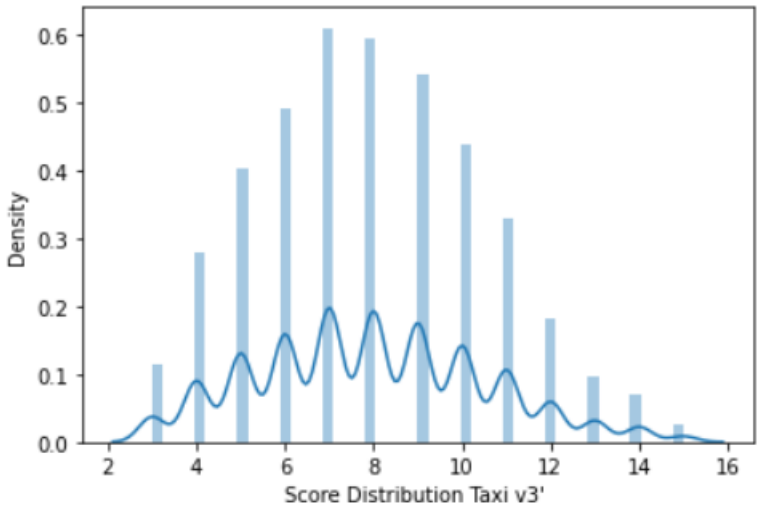
Với map Taxi v3, ta thấy rằng policy iteration cho kết quả thực nghiệm là hội tụ ở iteration thứ 16, và ta cũng sẽ thống kê điểm map taxi v3 với 50000 lần

```
arr_Score=[]

for i in range(50):
  env = gym.make('Taxi-v3')#load môi trường

  value_function, policy = policy_iteration(env)
  print(policy)
  success,score=play_multiple_times(env, policy, 1000)
  arr_Score+=score
```

```
Policy Iteration converged at 16-th.
[4 4 4 4 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 3 3 3 3 0 0 0 0 0 0 0 0 0 0 0 3
 0 0 0 0 0 0 0 2 2 2 2 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 2 2 2 2 0 0 0 0 0 0
 0 0 0 2 0 0 0 0 0 0 4 4 4 4 0 0 0 0 0 0 0 0 0 5 0 0 1 1 1 1 0 0 0 0 0 0 0
 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1
 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1
 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 2 2 2 2 0 0 0 0 2 2 2 2 1 2 0 2 1 1
 1 1 2 2 2 2 3 3 3 3 2 2 2 2 1 2 3 2 3 3 3 3 1 1 1 1 3 3 3 3 2 2 2 2 3 1 3
 2 3 3 3 3 1 1 1 1 3 3 3 3 0 0 0 0 3 1 3 0 3 3 3 3 1 1 1 1 3 3 3 3 0 0 0 0
 3 1 3 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 0 0 0 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 0 1 1 1 1 1 1 1
 1 4 4 4 4 1 1 1 1 1 1 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 4 4 4 1 1 1 5 1
 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 1 1 1 3]
Number of successes: 1000/1000
mean score: 7.898
```



5. Nhận xét

Từ kết quả chạy thục nghiệm ta có thể thấy rằng kết quả về số ván thắng (success) lẫn score của 3 map trên với 2 loại giải thuật Value Iteration và Policy Iteration là không quá chênh lệch. Bù lại thì Policy cho kết quả tốt hơn và có thời gian hội tụ nhanh hơn so với Value Iteration, ngoài ra có một lưu ý rằng kể cả bạn reset 1 map nào đó thì kết quả hội tụ vẫn không đổi, Và phân bố kết quả vẫn chưa phản ánh tốt hoàn toàn vì mỗi lần reset map giữa 2 map là độc lập(ta không đảm bảo rằng các loại map nhận được khi thực thi 2 phương pháp là giống nhau).