State-action value function example

February 8, 2023

1 State Action Value Function Example

In this Jupyter notebook, you can modify the mars rover example to see how the values of Q(s,a) will change depending on the rewards and discount factor changing.

[1]:
```
import numpy as np
from utils import *
```

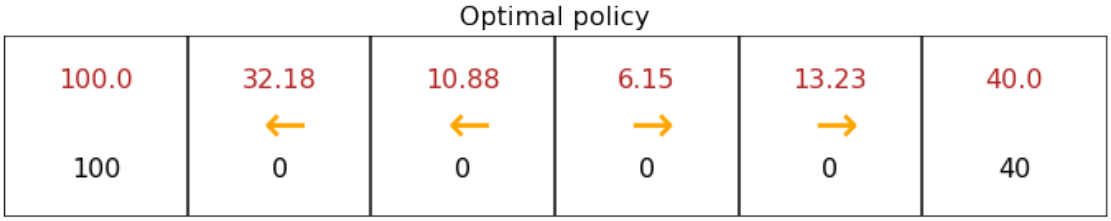
[2]:
```
# Do not modify
num_states = 6
num_actions = 2
```

[3]:
```
terminal_left_reward = 100
terminal_right_reward = 40
each_step_reward = 0

# Discount factor
gamma = 0.5

# Probability of going in the wrong direction
misstep_prob = 0.4
```

[4]:
```
generate_visualization(terminal_left_reward, terminal_right_reward,␣
↪each_step_reward, gamma, misstep_prob)
```



Q(s,a)

100.0 100.0 100	32.18 23.26 0	10.88 8.28 0	5.91 6.15 0	9.84 13.23 0	40.0 40.0 40

[13]:
```
import numpy as np
from utils import *
num_states = 6
num_actions = 2
terminal_left_reward = 100
terminal_right_reward = 40
each_step_reward = 5
# Discount factor
gamma = 0.5
# Probability of going in the wrong direction
misstep_pob = 0.1
generate_visualization(terminal_left_reward, terminal_right_reward,␣
↪each_step_reward, gamma, misstep_prob)
```

Optimal policy

100.0 100	38.93 ← 5	19.67 ← 5	14.93 → 5	19.99 → 5	40.0 40

Q(s,a)

100.0 100.0 100	38.93 30.9 5	19.67 17.27 5	14.9 14.93 5	17.48 19.99 5	40.0 40.0 40

[8]:
```
import utils
help(utils)
```

```
Help on module utils:

NAME
    utils

FUNCTIONS
```

```
    calculate_Q_value(num_states, rewards, transition_prob, gamma, V_states,
state, action)

    calculate_Q_values(num_states, rewards, transition_prob, gamma,
optimal_policy)

    evaluate_policy(num_states, rewards, transition_prob, gamma, policy)

    generate_rewards(num_states, each_step_reward, terminal_left_reward,
terminal_right_reward)

    generate_transition_prob(num_states, num_actions, misstep_prob=0)

    generate_visualization(terminal_left_reward, terminal_right_reward,
each_step_reward, gamma, misstep_prob)

    get_optimal_policy(num_states, num_actions, rewards, transition_prob, gamma)

    improve_policy(num_states, num_actions, rewards, transition_prob, gamma, V,
policy)

    plot_optimal_policy_return(num_states, optimal_policy, rewards, V)

    plot_q_values(num_states, q_left_star, q_right_star, rewards)

DATA
    __warningregistry__ = {'version': 26}

FILE
    /home/jovyan/work/utils.py
```

[9]:
```
generate_rewards(num_states, each_step_reward, terminal_left_reward,␣
↪terminal_right_reward)
```

[9]:
```
[100, 10, 10, 10, 10, 40]
```

[ ]:
```

```