12

베이지안 최적화

3. 패키지 활용

패키지 활용

목표함수 정의

```
def black_box_function(layer_num_actor, node_num_actor, epochs_actor,
                       layer_num_critic, node_num_critic, epochs_critic,
                       learning_rate_actor, learning_rate_critic,
                       discount_rate, smooth_rate,
                       penalty, mini_batch_step_size, loss_clipping
                      ):
    config_data = {
        'layer_num_actor':layer_num_actor,
        'node_num_actor':node_num_actor,
        'epochs_actor':epochs_actor,
        'layer_num_critic':layer_num_critic,
        'node_num_critic':node_num_critic,
        'epochs_critic':epochs_critic,

        'learning_rate_actor' :learning_rate_actor,
        'learning_rate_critic':learning_rate_critic,
        'discount_rate'       :discount_rate,
        'smooth_rate'        :smooth_rate,
        'penalty'             :penalty,
        'mini_batch_step_size':mini_batch_step_size,
        'loss_clipping'       :loss_clipping
    }
    agent = Agent(config_data)
    agent.train()
    return np.mean(agent.reward_list)
```

튜닝 파라미터

반환값 정의

- 목표함수는 튜닝할 파라미터를 결정하고, 이 파라미터를 기반으로 프로그램을 실행해서 결과를 반환하는 기능을 수행한다

패키지 활용

파라미터 범위 지정

```
pbounds = {
            'layer_num_actor':(1,2),
            'node_num_actor':(12,128),
            'epochs_actor':(3,6),
            'layer_num_critic':(1,2),
            'node_num_critic':(12,128),
            'epochs_critic':(3,6),

            'learning_rate_actor' :(0.0001,0.001),
            'learning_rate_critic':(0.0001,0.001),
            'discount_rate'       :(0.9,0.99),
            'smooth_rate'        :(0.9,0.99),
            'penalty'             :(-500,-10),
            'mini_batch_step_size':(4,80),
            'loss_clipping'       :(0.1,0.3)
        }
```

- 튜닝할 파라미터를 pbounds 변수에 딕셔너리 형태로 지정한다.

패키지 활용 Agent 클래스 변수 설정

```
def __init__(self, config_data):
    self.env = gym.make('CartPole-v1')
    self.state_size = self.env.observation_space.shape[0]
    self.action_size = self.env.action_space.n
    self.value_size = 1

    self.layer_num_actor = int(round(config_data['layer_num_actor'],0))
    self.node_num_actor = int(round(config_data['node_num_actor'],0))
    self.epochs_actor = int(round(config_data['epochs_actor'],0))
    self.layer_num_critic = int(round(config_data['layer_num_critic'],0))
    self.node_num_critic = int(round(config_data['node_num_critic'],0))
    self.epochs_critic = int(round(config_data['epochs_critic'],0))

    self.learning_rate_actor = config_data['learning_rate_actor']
    self.learning_rate_critic = config_data['learning_rate_critic']
    self.discount_rate = config_data['discount_rate']
    self.smooth_rate = config_data['smooth_rate']
    self.penalty = int(round(config_data['penalty'],0))
    self.mini_batch_step_size = int(round(config_data['mini_batch_step_size'],0))
    self.loss_clipping = config_data['loss_clipping']
```

정수로 변환

패키지 활용 프로그램 실행

베이지안 최적화 프로그램 실행

- 미리 만들어놓은 목표함수와 튜닝을 위한 딕셔너리를 사용해서 BayesianOptimization 클래스를 생성하고 학습을 위해 maximize 함수를 실행한다.

```
optimizer = BayesianOptimization(
    f=black_box_function,
    pbounds=pbounds,
    random_state=1,
)

optimizer.maximize(
    init_points=5,
    n_iter=20
)
```

랜덤 5회, 최적화 20회
총 50회 수행

패키지 활용

전체 코드 리뷰

코드 리뷰

패키지 활용 결과분석

실행로그

```
0.000630 | 0.24   | 11.78  | 60.03  | 92.55  | -297.1 | 0.9045 |
| 6      | 143.0  | 0.9626 | 3.623  | 3.248  | 1.306  | 1.548  | 0.000779 |
0.000576 | 0.1574 | 51.44  | 42.72  | 87.71  | -87.47 | 0.9575 |
| 7      | 33.65  | 0.9424 | 3.826  | 5.102  | 1.779  | 1.849  | 0.000585 |
0.000741 | 0.2949 | 34.85  | 61.93  | 28.71  | -420.1 | 0.9347 |
| 8      | 261.0  | 0.9561 | 5.377  | 4.037  | 1.091  | 1.536  | 0.000949 |
0.000489 | 0.1358 | 17.45  | 90.82  | 122.4  | -133.1 | 0.9446 |
| 9      | 29.04  | 0.9352 | 3.343  | 5.206  | 1.199  | 1.062  | 0.000959 |
0.000135 | 0.2367 | 44.62  | 119.1  | 51.09  | -47.23 | 0.9792 |
| 10     | 23.04  | 0.9682 | 5.624  | 4.292  | 1.77   | 1.288  | 0.000258 |
0.000582 | 0.2948 | 36.58  | 83.53  | 23.39  | -362.3 | 0.9589 |
| 11     | 42.52  | 0.9226 | 3.466  | 5.953  | 1.183  | 1.364  | 0.000418 |
0.000281 | 0.2086 | 10.69  | 119.6  | 18.01  | -368.9 | 0.9539 |
| 12     | 50.81  | 0.9216 | 5.941  | 4.988  | 1.431  | 1.603  | 0.000164 |
0.000936 | 0.2232 | 17.56  | 75.3   | 126.5  | -123.4 | 0.9677 |
| 13     | 28.11  | 0.9274 | 3.438  | 5.431  | 1.271  | 1.792  | 0.000169 |
0.000275 | 0.2057 | 17.54  | 85.69  | 24.01  | -315.8 | 0.9555 |
| 14     | 15.64  | 0.9223 | 4.879  | 4.707  | 1.951  | 1.056  | 0.000329 |
0.000266 | 0.2299 | 61.55  | 55.41  | 118.3  | -390.9 | 0.928  |
| 15     | 17.65  | 0.9873 | 4.764  | 4.072  | 1.858  | 1.077  | 0.000435 |
0.000464 | 0.2783 | 39.85  | 55.34  | 21.24  | -484.9 | 0.9184 |
| 16     | 20.63  | 0.9692 | 3.7    | 5.935  | 1.737  | 1.245  | 0.000185 |
0.000744 | 0.1418 | 72.77  | 38.84  | 66.25  | -443.6 | 0.954  |
| 17     | 28.45  | 0.9384 | 4.063  | 5.433  | 1.973  | 1.702  | 0.000253 |
0.000483 | 0.2822 | 20.95  | 14.45  | 56.22  | -360.9 | 0.9325 |
| 18     | 31.68  | 0.9874 | 4.124  | 5.531  | 1.317  | 1.099  | 0.000648 |
0.000143 | 0.1877 | 64.63  | 66.42  | 46.25  | -324.1 | 0.9804 |
| 19     | 112.5  | 0.9468 | 5.848  | 4.31   | 1.338  | 1.942  | 0.000230 |
0.000143 | 0.1504 | 17.42  | 119.2  | 72.16  | -343.8 | 0.9182 |
| 20     | 207.3  | 0.9015 | 4.888  | 4.1    | 1.926  | 1.801  | 0.000619 |
0.000111 | 0.1151 | 16.53  | 112.0  | 74.65  | -344.8 | 0.9123 |
| 21     | 334.6  | 0.9138 | 4.136  | 3.069  | 1.813  | 1.29   | 0.000704 |
0.000732 | 0.2671 | 18.73  | 119.2  | 74.68  | -347.1 | 0.9321 |
| 22     | 15.76  | 0.9723 | 5.409  | 3.821  | 1.124  | 1.855  | 0.000676 |
0.000359 | 0.189  | 50.06  | 46.28  | 89.35  | -84.37 | 0.9848 |
| 23     | 90.43  | 0.9503 | 4.78   | 4.383  | 1.522  | 1.837  | 0.000163 |
0.000425 | 0.1385 | 25.44  | 123.0  | 72.82  | -348.1 | 0.9822 |
| 24     | 33.81  | 0.9737 | 3.471  | 4.316  | 1.198  | 1.542  | 0.000170 |
0.000305 | 0.2157 | 17.06  | 121.6  | 77.53  | -353.6 | 0.9655 |
| 25     | 166.0  | 0.9869 | 4.916  | 5.744  | 1.673  | 1.487  | 0.000203 |
0.000690 | 0.2577 | 17.62  | 88.07  | 117.1  | -135.2 | 0.9771 |
```

패키지 활용 결과분석

```
target_list = []
i=0
for res in optimizer.res:
    target_list.append([res["target"], i])
    i=i+1
target_list.sort(reverse=True)
target_list
```

실행 결과 순으로 정렬

```
[[334.62, 20],
 [260.99, 7],
 [207.33, 19],
 [165.97, 24],
 [142.99, 5],
 [112.53, 18],
 [90.43, 22],
 [50.81, 11],
 [42.52, 10],
 [37.65, 0],
 [34.92, 4],
 [33.81, 23],
 [33.65, 6],
 [32.31, 2],
 [31.68, 17],
 [29.04, 8],
 [28.45, 16],
 [28.11, 12],
 [25.11, 3],
 [23.1, 1],
 [23.04, 9],
 [20.63, 15],
 [17.65, 14],
 [15.76, 21],
 [15.64, 13]]
```

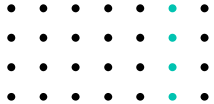
최적의 파라미터 확인

```
print("*result:" , optimizer.res[20]['params'])
```

```
*result: {'discount_rate': 0.9138068228055699, 'epochs_actor': 4.13579
6340297432, 'epochs_critic': 3.068824820615902, 'layer_num_actor': 1.8
127168005702576, 'layer_num_critic': 1.2899196661865222, 'learning_rat
e_actor': 0.0007044665544668867, 'learning_rate_critic': 0.00073253232
36616151, 'loss_clipping': 0.2671282081035625, 'mini_batch_step_size':
18.73240705651665, 'node_num_actor': 119.19096504720964, 'node_num_cri
tic': 74.68079589490598, 'penalty': -347.12015260105, 'smooth_rate':
0.9321116290822046}
```

최적 파라미터 확인

패키지 활용 PPO 알고리즘 개선



파라미터 설정

```
.....
LOSS_CLIPPING = 0.2671282081035625
class Agent(object):
    def __init__(self):
        self.env = gym.make('CartPole-v1')
        self.state_size = self.env.observation_space.shape[0]
        self.action_size = self.env.action_space.n
        self.value_size = 1

        self.layer_num_actor = int(round(1.8127168005702576,0))
        self.node_num_actor = int(round(119.19096504720964,0))
        self.epochs_actor = int(round(4.135796340297432,0))

        self.layer_num_critic = int(round(1.2899196661865222,0))
        self.node_num_critic = int(round(74.68079589490598,0))
        self.epochs_critic = int(round(3.068824820615902,0))

        self.learning_rate_actor = 0.0007044665544668867
        self.learning_rate_critic = 0.0007325323236616151
        self.discount_rate = 0.9138068228055699
        self.smooth_rate = 0.9321116290822046
        self.penalty = int(round(-347.12015260105,0))
        self.mini_batch_step_size = int(round(18.73240705651665,0))

        self.episode_num = 300

        self.moving_avg_size = 20

        self.model_actor = self.build_model_actor()
        self.model_critic = self.build_model_critic()
.....
```

패키지 활용 PPO 알고리즘 개선

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,5))
plt.plot(agent.reward_list, label='rewards')
plt.plot(agent.moving_avg_list, linewidth=4, label='moving average')
plt.legend(loc='upper left')
plt.title('PPO')
plt.show()
```