11

그리드서치 최적화

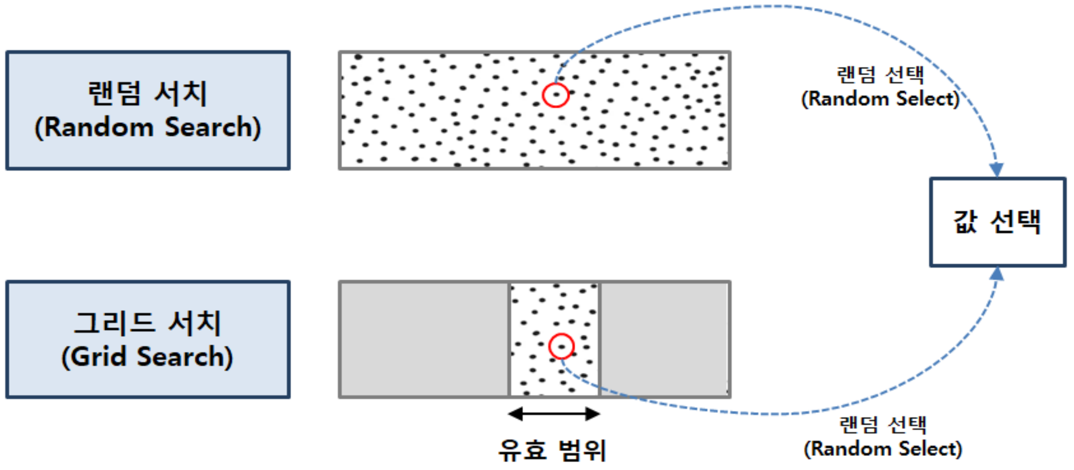
1. 그리드서치

그리드서치

랜덤서치와 그리드서치

랜덤서치와 그리드서치

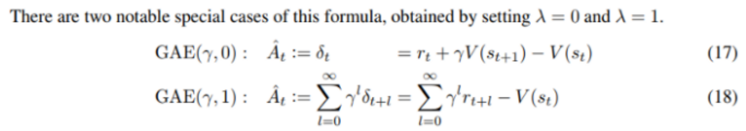
- 랜덤서치 : 무작위로 값을 선택해서 대입해 보는 것
- 그리드 서치 : 기준선 즉 범위를 정해 놓고 그 범위 안에서 값을 선택하는 것



그리드서치

그리드 설정방법

인터넷 검색 ➕ 직접 실행 ➕ 알고리즘 이해

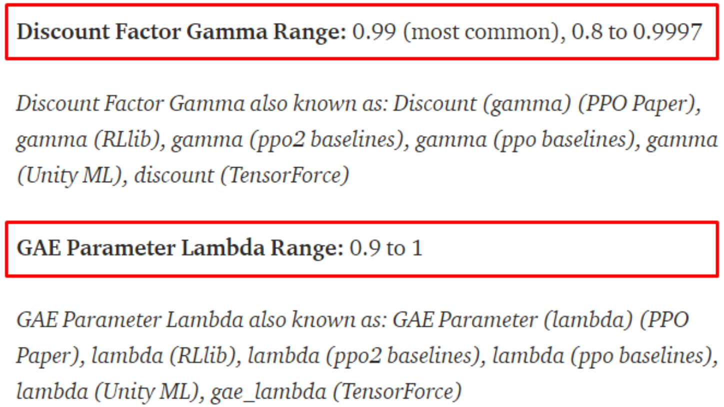


There are two notable special cases of this formula, obtained by setting $\lambda = 0$ and $\lambda = 1$.

$$\text{GAE}(\gamma, 0): \quad \hat{A}_t := \delta_t \quad = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{17}$$

$$\text{GAE}(\gamma, 1): \quad \hat{A}_t := \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} = \sum_{l=0}^{\infty} \gamma^l r_{t+l} - V(s_t) \tag{18}$$

Excerpt from GAE paper



Discount Factor Gamma Range: 0.99 (most common), 0.8 to 0.9997

Discount Factor Gamma also known as: Discount (gamma) (PPO Paper), gamma (RLlib), gamma (ppo2 baselines), gamma (ppo baselines), gamma (Unity ML), discount (TensorForce)

GAE Parameter Lambda Range: 0.9 to 1

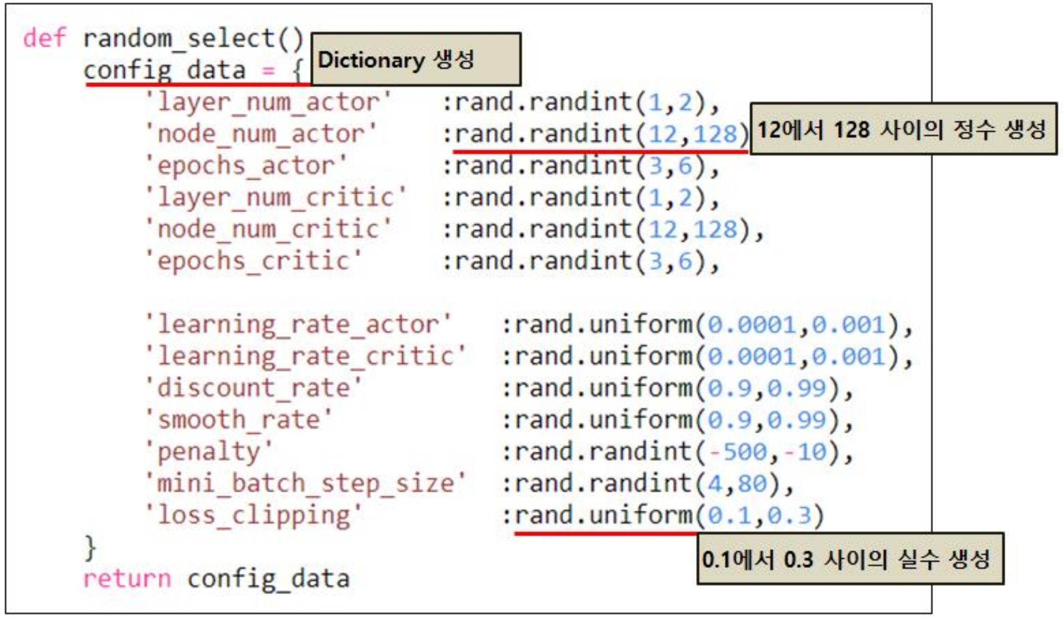
GAE Parameter Lambda also known as: GAE Parameter (lambda) (PPO Paper), lambda (RLlib), lambda (ppo2 baselines), lambda (ppo baselines), lambda (Unity ML), gae_lambda (TensorForce)

출처: https://www.codecademy.com/articles/normalization

그리드서치

코딩

random_select 함수 생성



```
def random_select():
    config_data = {
        'layer_num_actor'    :rand.randint(1,2),
        'node_num_actor'     :rand.randint(12,128),
        'epochs_actor'       :rand.randint(3,6),
        'layer_num_critic'   :rand.randint(1,2),
        'node_num_critic'    :rand.randint(12,128),
        'epochs_critic'      :rand.randint(3,6),

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

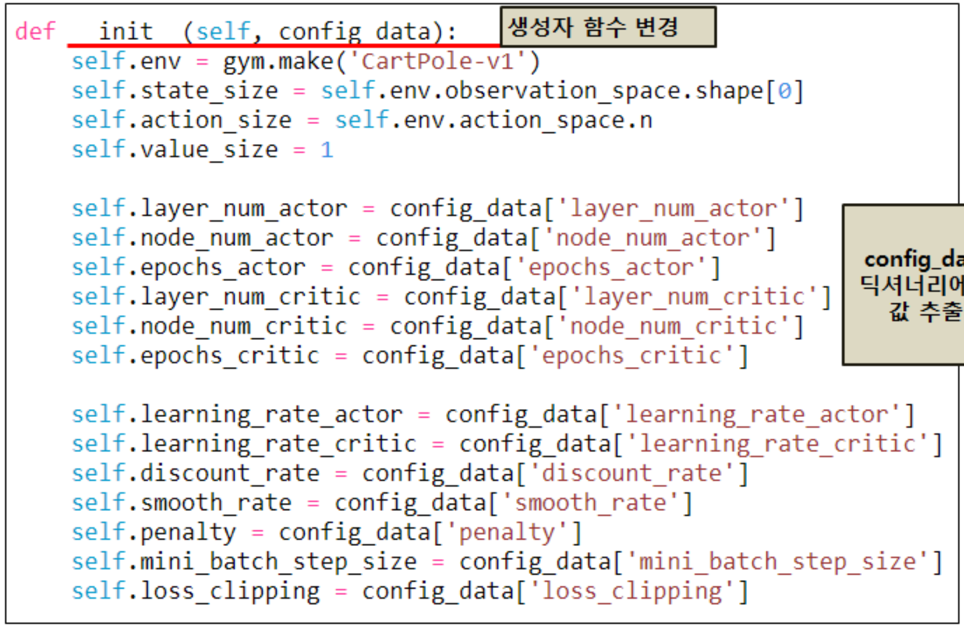
Dictionary 생성

12에서 128 사이의 정수 생성

0.1에서 0.3 사이의 실수 생성

그리드서치 코딩

Agent 클래스 변경

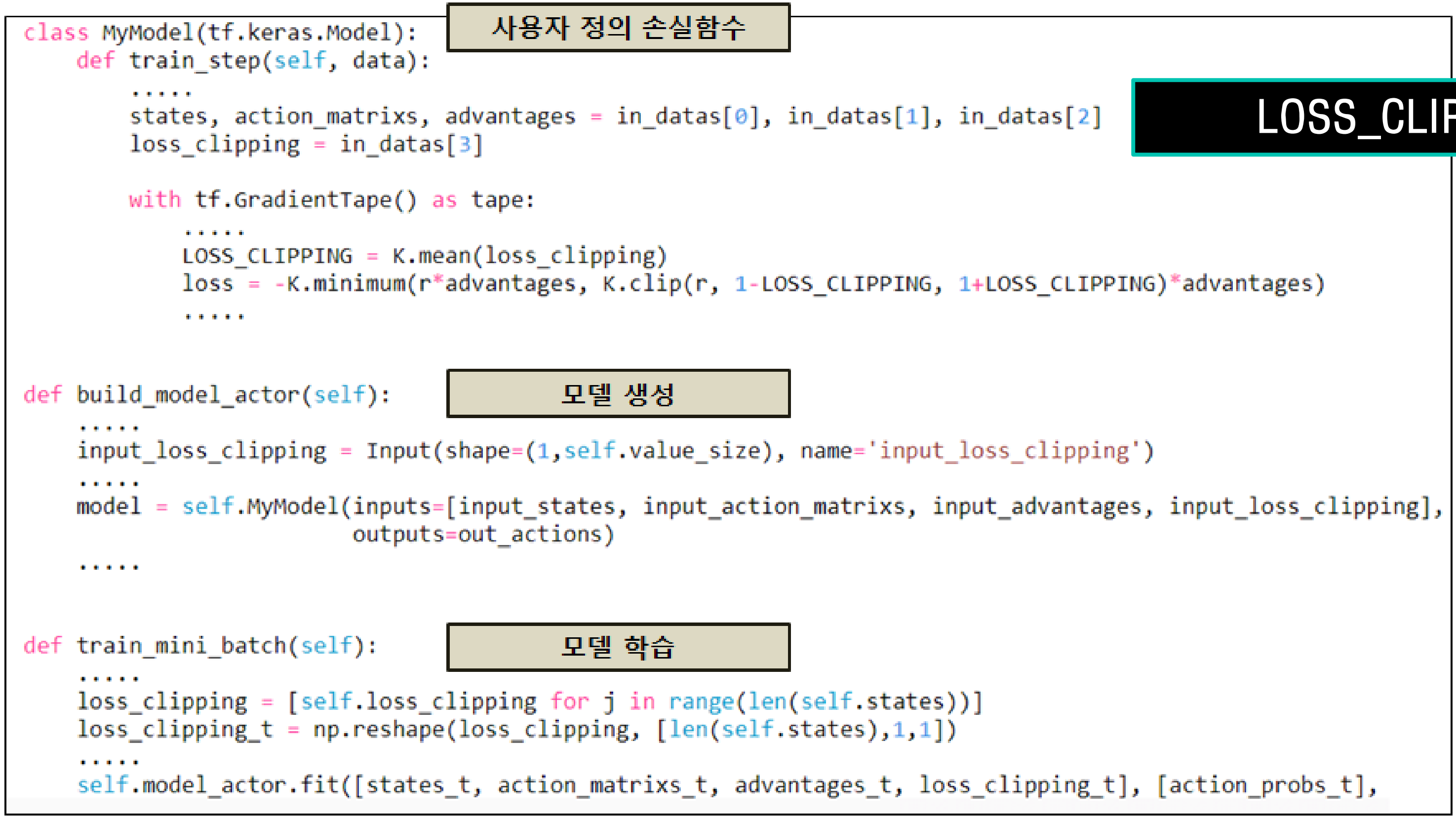


```
def __init__(self, config_data):
    self.env = gym.make('CartPole-v1')
    self.state_size = self.env.observation_space.shape[0]
    self.action_size = self.env.action_space.n
    self.value_size = 1

    self.layer_num_actor = config_data['layer_num_actor']
    self.node_num_actor = config_data['node_num_actor']
    self.epochs_actor = config_data['epochs_actor']
    self.layer_num_critic = config_data['layer_num_critic']
    self.node_num_critic = config_data['node_num_critic']
    self.epochs_critic = config_data['epochs_critic']

    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 = config_data['penalty']
    self.mini_batch_step_size = config_data['mini_batch_step_size']
    self.loss_clipping = config_data['loss_clipping']
```

생성자 함수 변경

config_data
딕셔너리에서
값 추출

그리드서치 코딩



사용자 정의 손실함수

```
class MyModel(tf.keras.Model):
    def train_step(self, data):
        .....
        states, action_matrixs, advantages = in_datas[0], in_datas[1], in_datas[2]
        loss_clipping = in_datas[3]

        with tf.GradientTape() as tape:
            .....
            LOSS_CLIPPING = K.mean(loss_clipping)
            loss = -K.minimum(r*advantages, K.clip(r, 1-LOSS_CLIPPING, 1+LOSS_CLIPPING)*advantages)
            .....
```

LOSS_CLIPPING 변수 전달

모델 생성

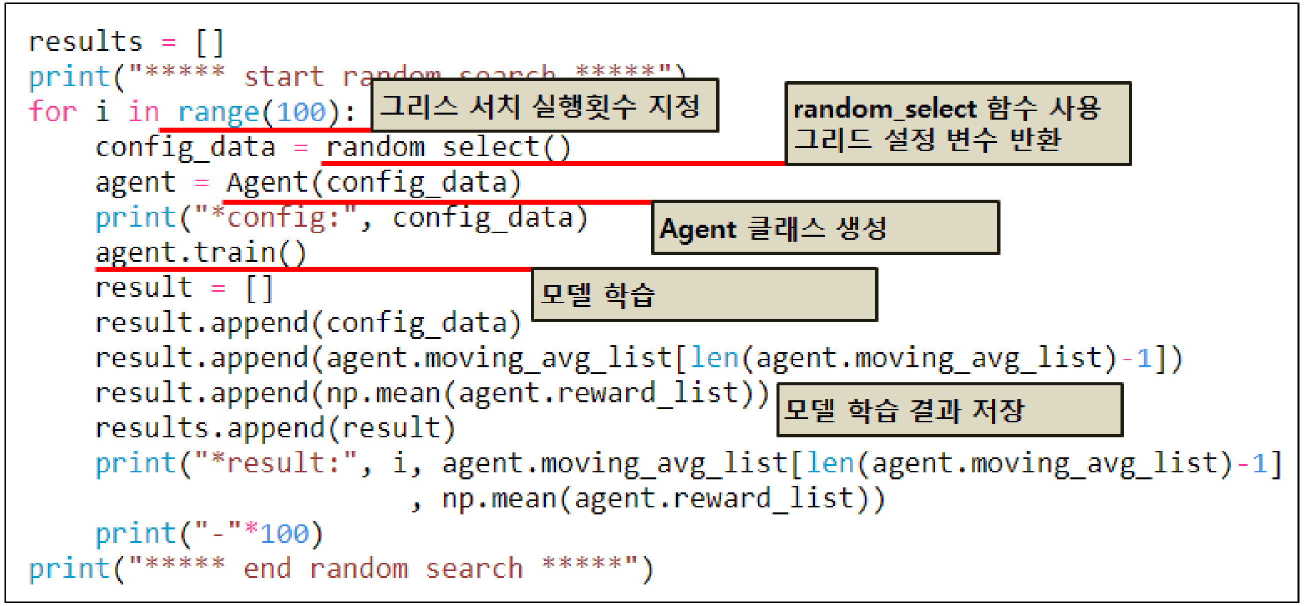
```
def build_model_actor(self):
    .....
    input_loss_clipping = Input(shape=(1,self.value_size), name='input_loss_clipping')
    .....
    model = self.MyModel(inputs=[input_states, input_action_matrixs, input_advantages, input_loss_clipping],
                         outputs=out_actions)
    .....
```

모델 학습

```
def train_mini_batch(self):
    .....
    loss_clipping = [self.loss_clipping for j in range(len(self.states))]
    loss_clipping_t = np.reshape(loss_clipping, [len(self.states),1,1])
    .....
    self.model_actor.fit([states_t, action_matrixs_t, advantages_t, loss_clipping_t], [action_probs_t],
```

그리드서치 코딩

그리드서치 실행



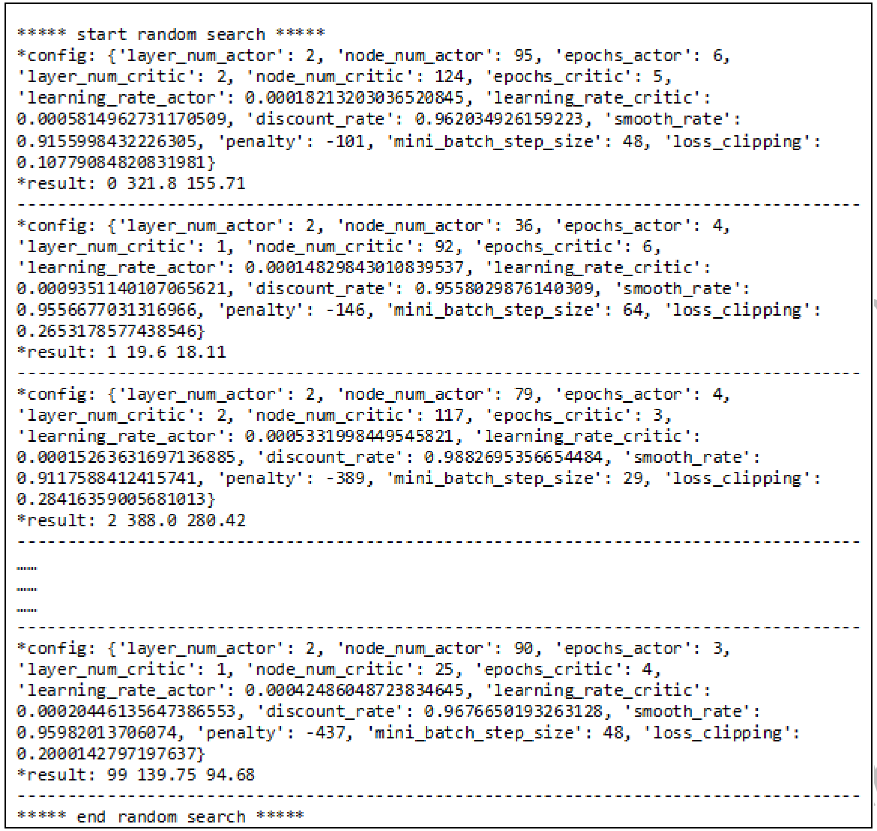
```
results = []
print("***** start random search *****")
for i in range(100):
    config_data = random_select()
    agent = Agent(config_data)
    print("*config:", config_data)
    agent.train()
    result = []
    result.append(config_data)
    result.append(agent.moving_avg_list[len(agent.moving_avg_list)-1])
    result.append(np.mean(agent.reward_list))
    results.append(result)
    print("*result:", i, agent.moving_avg_list[len(agent.moving_avg_list)-1]
                        , np.mean(agent.reward_list))
    print("-"*100)
print("***** end random search *****")
```

그리드서치 코딩

전체 코드 리뷰

코드 리뷰

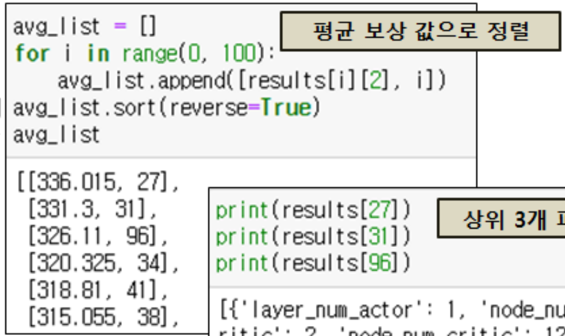
그리드서치 실행결과



```
***** start random search *****
*config: {'layer_num_actor': 2, 'node_num_actor': 95, 'epochs_actor': 6,
'layer_num_critic': 2, 'node_num_critic': 124, 'epochs_critic': 5,
'learning_rate_actor': 0.00018213203036520845, 'learning_rate_critic':
0.0005814962731170509, 'discount_rate': 0.962034926159223, 'smooth_rate':
0.915599843226305, 'penalty': -101, 'mini_batch_step_size': 48, 'loss_clipping':
0.10779084820831981}
*result: 0 321.8 155.71
--------------------------------------------------------------------------------
*config: {'layer_num_actor': 2, 'node_num_actor': 36, 'epochs_actor': 4,
'layer_num_critic': 1, 'node_num_critic': 92, 'epochs_critic': 6,
'learning_rate_actor': 0.00014829843010839537, 'learning_rate_critic':
0.0009351140107065621, 'discount_rate': 0.9558029876140309, 'smooth_rate':
0.9556677031316966, 'penalty': -146, 'mini_batch_step_size': 64, 'loss_clipping':
0.2653178577438546}
*result: 1 19.6 18.11
--------------------------------------------------------------------------------
*config: {'layer_num_actor': 2, 'node_num_actor': 79, 'epochs_actor': 4,
'layer_num_critic': 2, 'node_num_critic': 117, 'epochs_critic': 3,
'learning_rate_actor': 0.0005331998449545821, 'learning_rate_critic':
0.00015263631697136885, 'discount_rate': 0.9882695356654484, 'smooth_rate':
0.9117588412415741, 'penalty': -389, 'mini_batch_step_size': 29, 'loss_clipping':
0.28416359005681013}
*result: 2 388.0 280.42
--------------------------------------------------------------------------------
……
……
……
--------------------------------------------------------------------------------
*config: {'layer_num_actor': 2, 'node_num_actor': 90, 'epochs_actor': 3,
'layer_num_critic': 1, 'node_num_critic': 25, 'epochs_critic': 4,
'learning_rate_actor': 0.00042486048723834645, 'learning_rate_critic':
0.00020446135647386553, 'discount_rate': 0.9676650193263128, 'smooth_rate':
0.95982013706074, 'penalty': -437, 'mini_batch_step_size': 48, 'loss_clipping':
0.2000142797197637}
*result: 99 139.75 94.68
--------------------------------------------------------------------------------
***** end random search *****
```

- Config 부분에는 해당 루틴을 수행할 때 선택된 파라미터 값을 확인할 수 있고
- result 부분에는 수행 횟수, 20회 이동 평균 그리고 전체 보상에 대한 평균 값을 확인할 수 있다

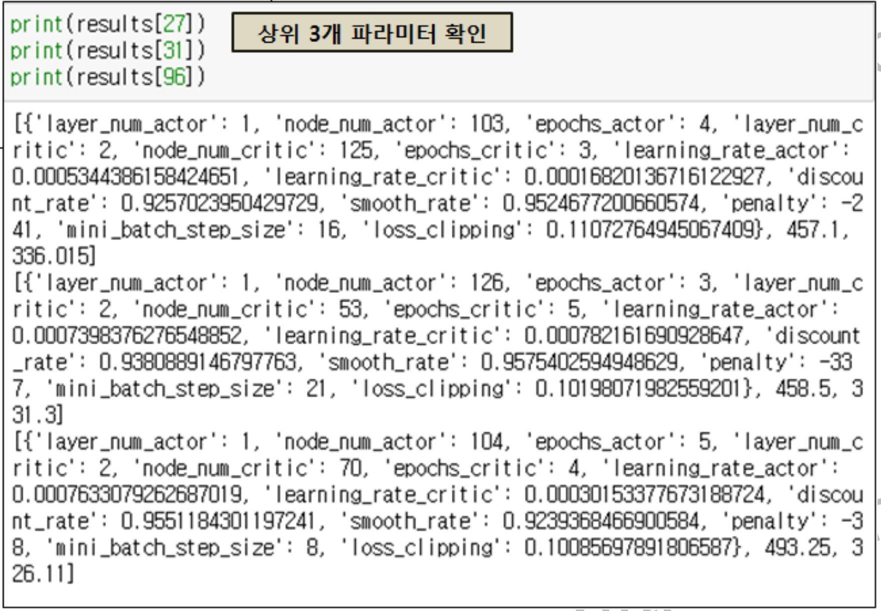
그리드서치 결과분석



```
avg_list = []
for i in range(0, 100):
    avg_list.append([results[i][2], i])
avg_list.sort(reverse=True)
avg_list
```

평균 보상 값으로 정렬

```
[[336.015, 27],
 [331.3, 31],
 [326.11, 96],
 [320.325, 34],
 [318.81, 41],
 [315.055, 38],
```



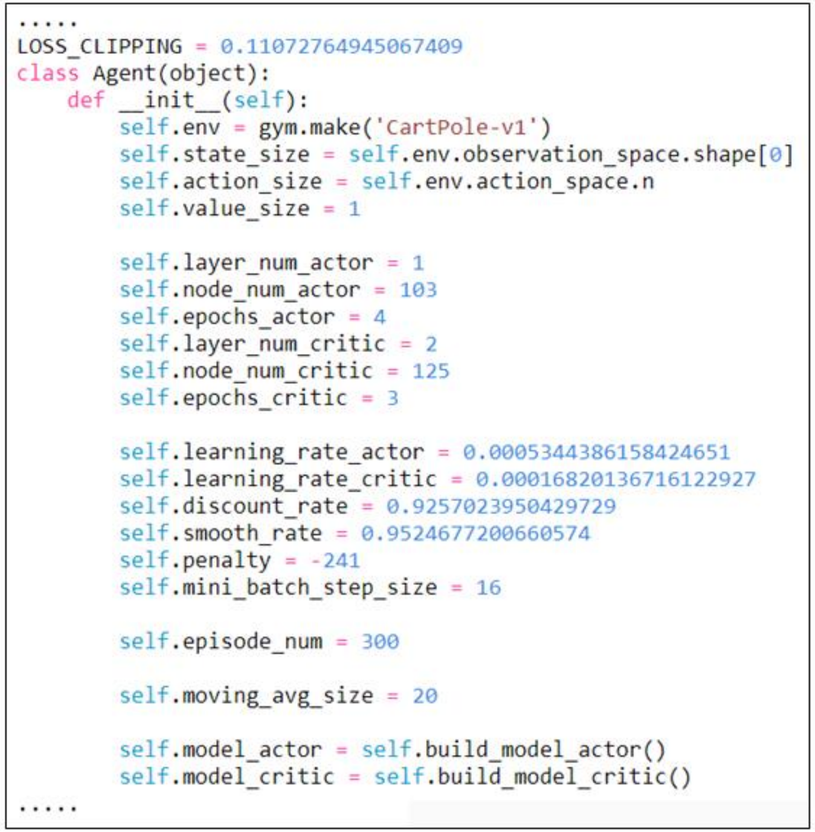
```
print(results[27])
print(results[31])
print(results[96])
```

상위 3개 파라미터 확인

```
[{'layer_num_actor': 1, 'node_num_actor': 103, 'epochs_actor': 4, 'layer_num_c
ritic': 2, 'node_num_critic': 125, 'epochs_critic': 3, 'learning_rate_actor':
0.0005344386158424651, 'learning_rate_critic': 0.00016820136716122927, 'discou
nt_rate': 0.9257023950429729, 'smooth_rate': 0.9524677200660574, 'penalty': -2
41, 'mini_batch_step_size': 16, 'loss_clipping': 0.11072764945067409}, 457.1,
336.015]
[{'layer_num_actor': 1, 'node_num_actor': 126, 'epochs_actor': 3, 'layer_num_c
ritic': 2, 'node_num_critic': 53, 'epochs_critic': 5, 'learning_rate_actor':
0.0007398376276548852, 'learning_rate_critic': 0.0007821616909286447, 'discount
_rate': 0.9380889146797763, 'smooth_rate': 0.9575402594948629, 'penalty': -33
7, 'mini_batch_step_size': 21, 'loss_clipping': 0.10198071982559201}, 458.5, 3
31.3]
[{'layer_num_actor': 1, 'node_num_actor': 104, 'epochs_actor': 5, 'layer_num_c
ritic': 2, 'node_num_critic': 70, 'epochs_critic': 4, 'learning_rate_actor':
0.0007633079262687019, 'learning_rate_critic': 0.00030153377673188724, 'discou
nt_rate': 0.9551184301197241, 'smooth_rate': 0.9239368466900584, 'penalty': -3
8, 'mini_batch_step_size': 8, 'loss_clipping': 0.10085697891806587}, 493.25, 3
26.11]
```

- 평균 보상 값이 알고리즘의 전체 성능을 평가하는 중요한 척도
- 이기 때문에 이 값을 역순으로 정렬
- 상위 3개 값의 인덱스를 확인한 다음 results 변수에서 파라미터 확인

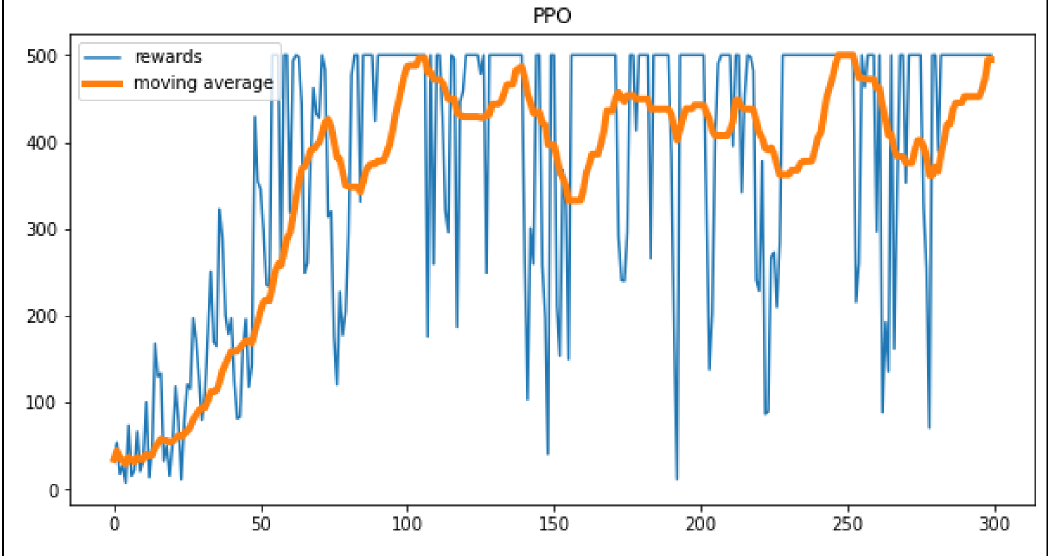
그리드서치 파라미터 적용



```
.....
LOSS_CLIPPING = 0.11072764945067409
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 = 1
        self.node_num_actor = 103
        self.epochs_actor = 4
        self.layer_num_critic = 2
        self.node_num_critic = 125
        self.epochs_critic = 3

        self.learning_rate_actor = 0.0005344386158424651
        self.learning_rate_critic = 0.00016820136716122927
        self.discount_rate = 0.9257023950429729
        self.smooth_rate = 0.9524677200660574
        self.penalty = -241
        self.mini_batch_step_size = 16

        self.episode_num = 300

        self.moving_avg_size = 20

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

그리드서치 실행 결과 분석

```
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()
```


PPO
rewards
moving average