Deep Reinforcement Learning Homework#4

112062574 郭士平

NeurIPS 2019: Learn to Move - Walk Around

Soft Actor-Critic method prevents the learning procedure to get stuck in a local optimal, and it performs well in multi-goal environments. Hence, it is chosen to train a RL agent to perform this task since there might be several different ways to walk or run.

SAC overview

Algorithm 1 Soft Actor-Critic

Initialize parameter vectors ψ, $\bar{\psi}$, θ, ϕ.
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$
  end for
end for

ψ for `value_net`

$\bar{\psi}$ for `value_target`, using soft update

θ_i for $i \in \{1, 2\}$ represent `q_net1` and `q_net2`

ϕ for `policy_net`

```
# Memory and Networks
self.memory = Memory(memory_size = self.memory_size)
self.policy_net = PolicyNetwork(self.n_states, self.n_actions).to(self.device)
self.q_net1 = QNetwork(self.n_states, self.n_actions).to(self.device)
self.q_net2 = QNetwork(self.n_states, self.n_actions).to(self.device)
self.value_net = ValueNetwork(self.n_states).to(self.device)
self.value_target = ValueNetwork(self.n_states).to(self.device)
self.value_target.load_state_dict(self.value_net.state_dict())
self.value_target.eval()

self.value_loss = torch.nn.MSELoss()
self.q_value_loss = torch.nn.MSELoss()

self.value_opt = torch.optim.Adam(self.value_net.parameters(), lr=self.lr)
self.q_net1_opt = torch.optim.Adam(self.q_net1.parameters(), lr=self.lr)
self.q_net2_opt = torch.optim.Adam(self.q_net2.parameters(), lr=self.lr)
self.policy_opt = torch.optim.Adam(self.policy_net.parameters(), lr=self.lr)
```

States Representation

For every state, simply flatten the whole observation dictionary into a numpy array with size `(339, )` as the input state of all networks. The following screenshot is the testing part. `flatten_obs` gets the raw observation and outputs a (339,) numpy array.

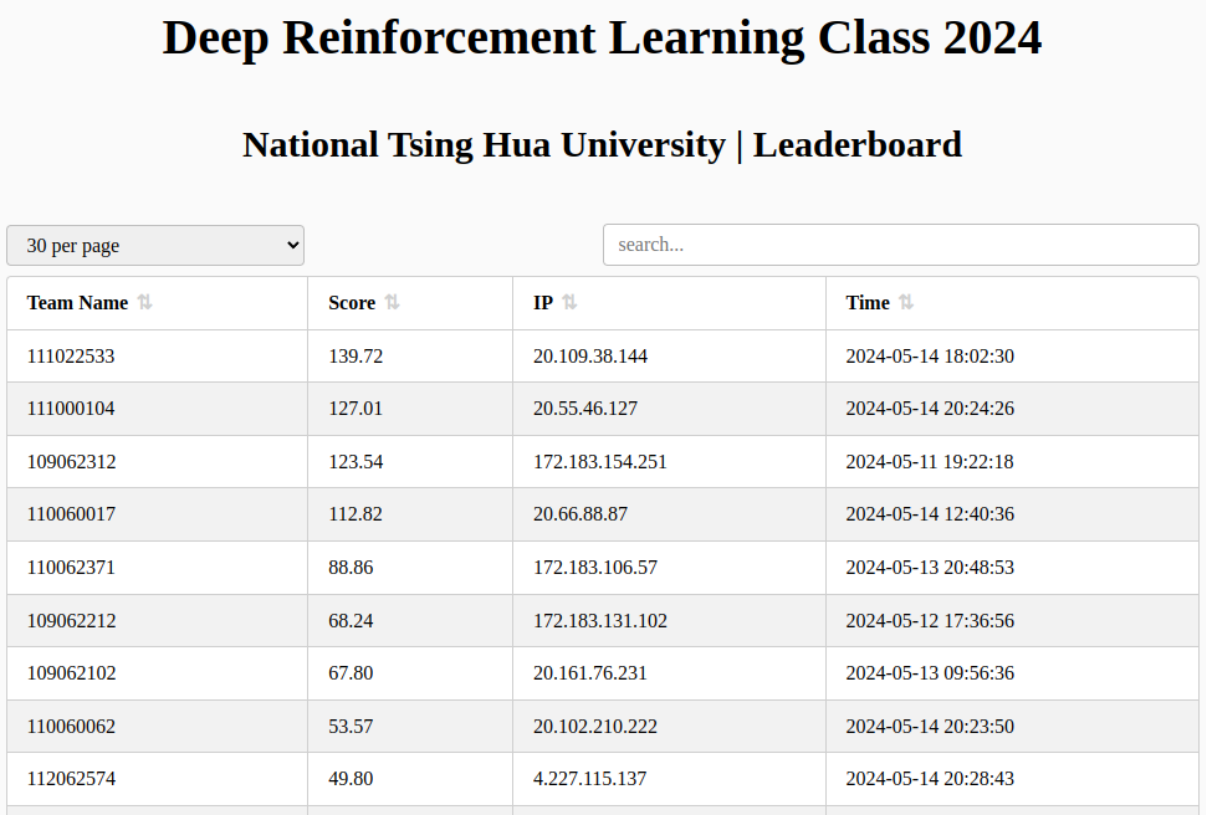
```
class Agent(object):
    def __init__(self):
        self.policy = PolicyNetwork(339, 22)
        self.policy.load_state_dict(torch.load('112062574_hw4_data', map_location=torch.device('cpu')))
        self.policy.eval()
        self.device = 'cpu'

    def act(self, observation):
        state = flatten_obs(observation)
        state = np.expand_dims(state, axis=0)
        state = torch.from_numpy(state).float().to(self.device)
        action, log_probs = self.policy.sample(state)
        return action.detach().cpu().numpy()[0]
```

The rest part of the implementations are common soft actor-critic. Nothing particular. The parameters references the TA's.

```
# Parameters
episodes = 10000
checkpoint = 100
memory_size = 4000000
batch_size = 256
gamma = 0.99
lr = 0.0005
tau = 0.01
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

Results

It hits 49.80 after training about 2800 episodes. The best result on the leaderboard was 52.51 showed in the second leaderboard screenshot.



Deep Reinforcement Learning Class 2024

National Tsing Hua University | Leaderboard

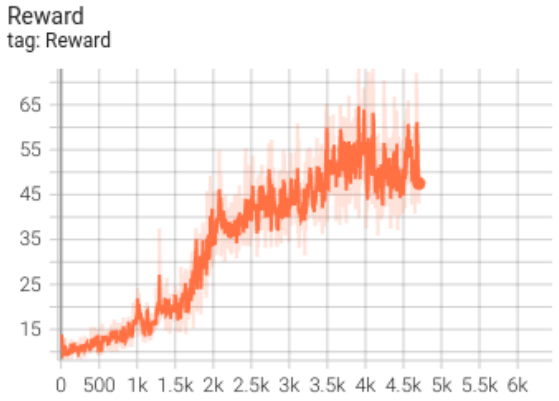
30 per page

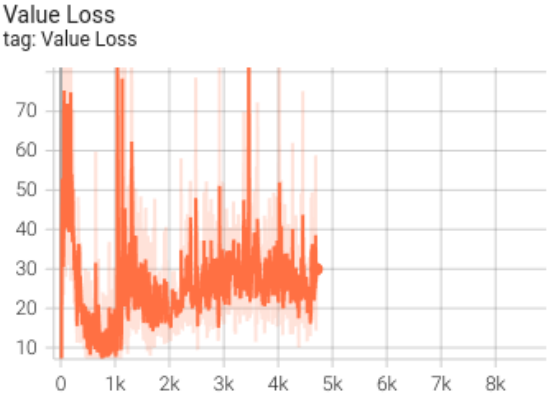
search...

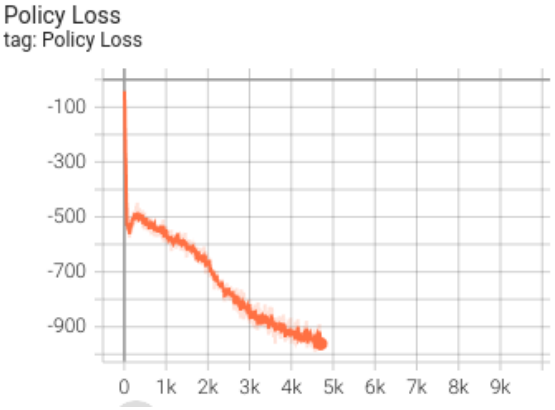
Team Name	Score	IP	Time
111022533	139.72	20.109.38.144	2024-05-14 18:02:30
111000104	127.01	20.55.46.127	2024-05-14 20:24:26
109062312	123.54	172.183.154.251	2024-05-11 19:22:18
110060017	112.82	20.66.88.87	2024-05-14 12:40:36
110062371	88.86	172.183.106.57	2024-05-13 20:48:53
109062212	68.24	172.183.131.102	2024-05-12 17:36:56
109062102	67.80	20.161.76.231	2024-05-13 09:56:36
110060062	53.57	20.102.210.222	2024-05-14 20:23:50
112062574	49.80	4.227.115.137	2024-05-14 20:28:43

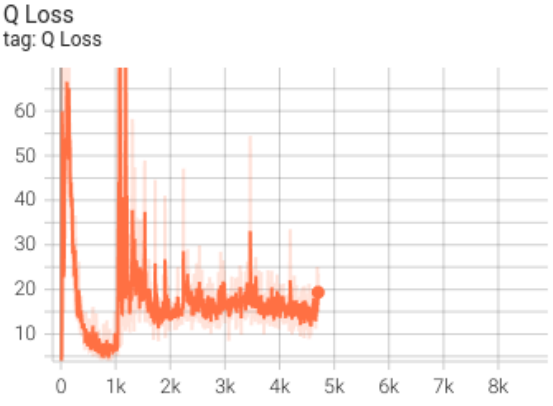
▲project.aseart.com

110060017	112.82	20.66.88.87	2024-05-14 12:40:36
109062212	98.52	20.55.14.224	2024-05-14 23:57:04
110062371	96.89	172.183.53.46	2024-05-14 21:41:19
109062102	94.01	20.172.6.168	2024-05-14 23:58:41
110060062	58.62	172.183.147.114	2024-05-14 21:47:02
110062112	54.48	52.234.26.86	2024-05-14 21:49:20
112062574	52.51	172.183.162.41	2024-05-15 14:22:11
111034521	35.48	172.171.255.193	2024-05-14 23:58:15
112062530	35.10	20.51.198.249	2024-05-11 01:17:33
109062114	29.25	[redacted]	2024-05-13 15:40:24









Discussion

Though the total reward increases though, at test time, the agent didn’t seem to learn how to walk. The rewards come from moving feet slightly without falling down. Some exploration tricks may be needed if we want to hit higher reward.