林柏均 107062240

May 2, 2021

Deep Reinforcement Learning

Homework 3

What have I done in this assignment?

Instead of using traditional Deep Q-Learning, this time I used Rainbow to train my agent. The reason why I abandoned traditional way to train my agent because of the performance. After 1000000 frames of training, we can notice that the runner can do something remarkable.

Rainbow

Rainbow integrates all the following seven components into a single integrated agent:

1. DQN
2. Double DQN
3. Prioritized Experience Replay
4. Dueling Network
5. Noisy Network
6. Categorical DQN
7. N-step Learning

Description of my model

Prioritized Experience Replay:

- max_priority (float): max priority
- tree_ptr (int): next index of tree
- alpha (float): alpha parameter for prioritized replay buffer
- sum_tree (SumSegmentTree): sum tree for prior
- min_tree (MinSegmentTree): min tree for min prior to get max weight

The key concept of PER's implementation is *Segment Tree*. It efficiently stores and samples transitions while managing the priorities of them.

Noisy Layer:

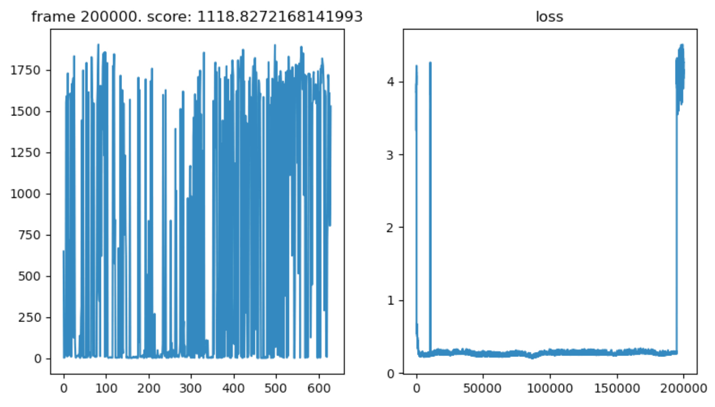
- in_features (int): input size of linear module
- out_features (int): output size of linear module
- std_init (float): initial std value
- weight_mu (nn.Parameter): mean value weight parameter
- weight_sigma (nn.Parameter): std value weight parameter
- bias_mu (nn.Parameter): mean value bias parameter
- bias_sigma (nn.Parameter): std value bias parameter

NoisyNet is an exploration method that learns perturbations of the network weights to drive exploration. The key insight is that a single change to the weight vector can induce a consistent, and potentially very complex, state-dependent change in policy over multiple time steps.

Rainbow Agent:

- env (gym.Env): openAI Gym environment
- memory (PrioritizedReplayBuffer): replay memory to store transitions
- batch_size (int): batch size for sampling
- target_update (int): period for target model's hard update
- gamma (float): discount factor
- dqn (Network): model to train and select actions
- dqn_target (Network): target model to update
- optimizer (torch.optim): optimizer for training dqn
- transition (list): transition information including state, action, reward, next_state, done
- v_min (float): min value of support
- v_max (float): max value of support
- support (torch.Tensor): support for categorical dqn
- use_n_step (bool): whether to use n_step memory
- n_step (int): step number to calculate n-step td error
- memory_n (ReplayBuffer): n-step replay buffer

Conclusion



Above figure record my training score and loss, we can notice that the loss value changed quite rapidly after 10000 frames, and became quite low, it has shown that the Rainbow learning performance is quite great. Rainbow is a complicated model for me to implement so that I studied a lot in order to improve my assignment. We can notice that the runner could not actually run, instead, he learned to climb, so the agent hardly can learn run if we didn't provide it some trajectories.