RL_exercise

Environment: CartPole-v0 from gym

Agent algorithm: Deep Q-Learning

Algorithm 1: deep Q-learning with experience replay.
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For t = 1,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

Q-learning is a model-free algorithm that learns optimal Q(s,a) action value functions from the agent's history of interaction with the environment.

In Deep Q-learning, neural network is used to calculate action value function. To ensure stable performance, two tricks are used:

1. Experience Replay: At each time step the agent memorizes some experience (state, action, reward, next_action, done), and learns by replaying a batch of experience from its memory. This is done to reduce the effect of correlations between sequence of observations which would have made neural network behave poorly.
2. Delayed Update: A separate "target" network is used to generate target value during experience replay. Every C timesteps the target network is updated to the parameters of training network. Doing so can increase the stability of the model, because it reduces the issue that a very small update to training network may change the outcome of policy, causing possible divergence during training.

Current Results

Currently Deep Q-learning algorithm has been completely implemented including experience replay and delayed update. On top of that a stopping mechanism is implemented to stop learning when the agent can pass a consecutive number of episodes with maximum rewards. Now the agent can consistently pass the given task in around 400 episodes of training, getting maximum reward for all 100 test episodes.

Reference

Deep Q-Learning Nature

Mxnet Tutorial