

Deep Reinforcement Learning Nanodegree

Project 1: Navigation - Report

DEEP Q-LEARNING (DQN) ALGORITHM

The DQN algorithm is a way to represent the optimal action-value function, q_* , as a neural network, instead of a typical Q-table. It's architecture is typically comprised of convolutional layers, followed by fully-connecter layers with ReLU activation functions. This expands the range of problems we can solve from environments with a discrete action space, to a continuous action space.

IMPLEMENTATION

- Initialize replay memory D with capacity N (Using a circular-queue that retains the N most recent experience tuples)
- Initialize action-value function \hat{q} with random weights w (randomly from a normal distribution with variance of 2 times the number of inputs to each neuron)
- Initialize target action-value weights $w^- \leftarrow w$
- For the episode $e \leftarrow 1$ to M :
 - Initial input frame x_1
 - Prepare initial state: $S \leftarrow \phi(\langle x_1 \rangle)$ (this pre-processes and stacks a sequence of frames - if we want to stack 4 frames, we need to do something special for the first 3 time steps, like skip storing the experience tuples until we get a complete sequence)
 - For each time step $t \leftarrow 1$ to T

Process 1: Sampling

We'll sample the environment by performing action and store observed experienced tuples in a replay memory/buffer

- Choose action A from state S using policy $\pi \leftarrow \epsilon$ -Greedy($\hat{q}(S, A, w)$)
- Take action A , observe reward R , and next input frame x_{t+1}
- Prepare next state: $S' \leftarrow \phi(\langle x_{t-2}, x_{t-1}, x_t, x_{t+1} \rangle)$
- Store experience tuple (S, A, R, S') in replay memory D
- $S \leftarrow S'$

Process 2: Learning

We'll sample a small batch of tuples from this memory randomly, and learn from it using a gradient descent update step

- Obtain random mini-batch of tuples (s_j, a_j, r_j, s_{j+1}) from D
- Set target $y_j = r_j + \gamma \max_a \hat{q}(s_{j+1}, a, w^-)$
- Update: $\Delta w = \alpha (y_j - \hat{q}(s_j, a_j, w)) \nabla_w \hat{q}(s_j, a_j, w)$
- Every C steps, reset: $w^- \leftarrow w$