Project Report

Learning Algorithm

The agent is implemented with deep-Q network (DQN) learning. Two copies of DQNs are maintained, one as local network, the other as target network. The agent chooses an action according to the Q values computed by the local network, Q_local, following ϵ-greedy policy. Parameters in the local network θ are optimized in order to obtain a minimum prediction error:

$$\delta = r + \gamma \max_{a_{t+1}} Q_{target}(s_{t+1}, a_{t+1}; \omega) - Q_{local}(s_t, a_t; \theta),$$

where r is the reward at step t, s_t and a_t are respectively the state and the action at step t, θ is the parameters for the local network and ω is the parameters for the target network, updated by:

$$\theta := \theta + \alpha \delta \nabla_\theta Q_{local}(s_t, a_t; \theta),$$

where α is the learning rate. ω is soft-updated from θ using

$$\omega := (1 - \tau)\omega + \tau\theta$$

Here τ is a predefined constant which determines how fast ω is updated towards θ.

During a game, tuples of state, action, reward, next state and terminal status are stored in a memory buffer as "experiences". In the trainig step, experiences are sample from the memory buffer and used to supervise the optimization of network parameters.

Network Structure and Hyper Parameters

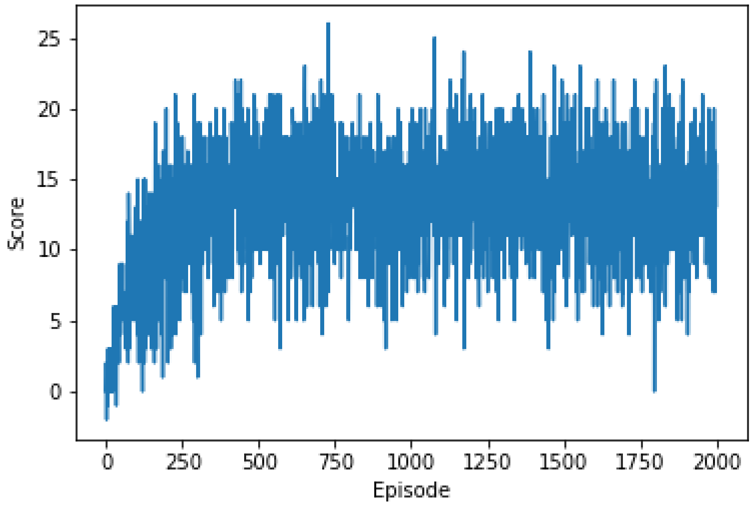
- 2 hidden layers: `256`, `128`

Hyper Parameters	Value
Discount factor γ	`0.99`
Learning rate α	`5e-4`
Soft update rate τ	`1e-3`

A size of `1e5` for the memory buffer, and a batch size of `64` for experience replay sampling are used. The target network is soft-updated every `4` steps.

Score Visualization

Average score vs. Episode



After 400 episodes, the average score stablized over 13.

Future Improvements

An entropy term may be added to regularize the training. Within the scope of DQN, Double-DQN may be used to improve the performance.