The agent is trained via Deep-Q learning, an algorithm based on neural network as function approximator of the action-value function. Replay-buffer is also adopted (details below).

Training lasts as long as the desired score (the average return over last 100 episodes) is lower than 16; however, it's stopped if desired score is not achieved in 1800 episodes.

Each episode consists of 1000 steps at most.

In each step the agent picks an action according to an epsilon-greedy policy (*epsilon* is equal to 1 at the first episode and decays by 0.993 after each episode until it gets equal to 0.01; the higher is *epsilon*, the higher is the randomness fraction in picking the action).

The action is followed by the environment 'reaction', expressed by the next state and reward (done is always false).

The SARS sequence (current state, action, reward and next state) is hence 'memorized' by the agent, i.e. it's stored in a memory buffer (the buffer can store at most 10.000 sequences: if full, the oldest sequence is replaced by the newest one).

Every 4 sequences, the agent updates its action-value function model with the 'experience' of 64 sequences randomly picked from the replay buffer (as soon as available).

The model is a neural network (Q-network) with 2 hidden layers and 64 rectified linear units in each one. It takes the current state as input (input layer size = state size = 37) and throws out the expected action-value of each action in that state (output layer size = action size = 4).

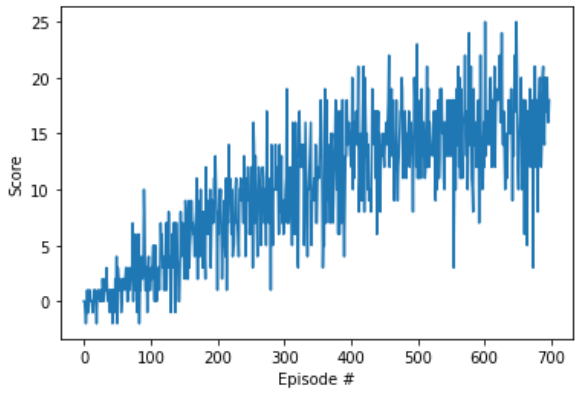
Updating the action-value function model with multiple SARS sequences is nothing but training the neural network, with stochastic gradient descent and back-propagation (learning rate is 5e-4). The loss function is defined as the mean square difference between multiple *'q-target'* and *'q-expected'* (one for each SARS sequence) values.

For a given sequence, *q-expected* is the action-value expected by the network for the action in the sequence in the current-state of the sequence.

Q-target is the action-value of the action in the sequence, in the state of the sequence, with return computed by assuming that the action in next state will be greedy according to the Q-network - a 'delayed' copy, actually - and that its value is given by the same network. The return is computed with *gamma* equals to 0.99.

By 'delayed' copy, we mean a copy of the trained Q-network – both are equal at the beginning - that is updated taking in account 0.1% of the trained-Q network at the end of every training step (delayed-copy weights = 99% delayed copy weights + 0.1% trained Q-network weights). In this sense, the 'delayed' copy slowly 'follows', step by step, the Q-network evolution.

With the described algorithm and hyperparameters, the agent has achieved the desired score (16) in 698 episodes:



Future work may be dedicated to implement dueling DQN and prioritized experience replay.