Saurabh Mangal Navigation

Overview:

Aim is to get the maximum combined score, hence play as long as possible.

Approach:

I am using Q learning to solve this problem.

Parameters:

- *BUFFER_SIZE = int(1e5) # replay buffer size*
- *BATCH_SIZE = 64 # minibatch size (number of episodes that it can learn from)*
- *GAMMA = 0.99 # discount factor*
- *TAU = 1e-3 # for soft update of target parameters*
- *LR = 5e-4 # learning rate (slower the better but too small limits the training speed)*
- *UPDATE_EVERY = 4 # how often to update the network*
- *MEMORY_CAPACITY = 2000*

I use three linear layer simply in my model with the first two layers having the same hidden layers size as environment isn't very complex and I don't need to cater to extract so many features from the input data, I didn't use a batch normalization layer as don't expect the overfitting in this case.

My Policy network is designed like this:

- *super(QNetwork, self).__init__()*
- *self.seed = torch.manual_seed(seed)*
- *self.fc1 = nn.Linear(state_size, fc1_units)*
- *self.fc2 = nn.Linear(fc1_units, fc2_units)*
- *self.fc3 = nn.Linear(fc2_units, action_size)*

Hope this explanation helps.

Training Graphs

Model stabilizes after 6000 episodes but overall average of about 13 was obtained at the 18000 episodes.

Saurabh Mangal Navigation