**Assignment Part II - Dueling Double DeepQ Network**

**Dueling DeepQ Network**

Afbeelding met diagram, schets, ontwerp, lijn

Automatisch gegenereerde beschrijvingIn the next assignment a Dueling Double DeepQ Network is implemented. This method was first proposed by Wang et al. (2016) as an extension to the already existing DeepQ Network. They used already existing algorithms for model-free Reinforcement Learning and only changed the neural network used for training. The idea of the Dueling DeepQ network is that not one network is trained for the Q-values (see figure 1, top), but two separate neural networks are trained inside the general network for both the State-values (V(s)) and the advantage function (A(s,a)) (see figure 1, bottom). This advantage function is designed of the V and Q value based on the following formula: A(s,a) = Q(s,a) – V(s) and can be seen as the relative added value of taking a specific action in the determined state. This formula can be transformed to Q(s,a) = V(s) + A(s,a), so in the end both separate neural networks will be combined to get the Q(s,a) values. This returns a Q-values for every possible action (a) in the inputted state (s).

The authors of the article mention, however, that if the output estimator Q(s,a) is correct, this does no mean that the same holds for the V(s) and A(s,a). There are more (incorrect) combinations of these last two that can lead to a correct Q(s,a) value. We do know, however, that with a deterministic policy and the optimal action: Q(s,a) = V(s) indicating A(s,a) = 0. Therefore we have to control the algorithm to produce an A(s,a) of at maximum 0, the other values should be negative. It does, however, make the optimization more stable to not subtract the maxaA(s,a) but the mean of the A(s,a) resulting in the following finale formula to extract the Q-value: Q(s,a) = V(s) + A(s,a) – mean(Q(s,a))

The added value of training an individual part of the network for the v-value is that the value of a specific state can be seen easily without having to check the effect of the actions. This can make the model learn to not focus on too much on which action to take in case there is almost no added value of a specific action, while it does focus on which action to take in the case it is highly relevant for the outcome.

Figure

**Dueling DeepQ Network in code**

Wang et al. (2016) explains that difference in code with the original DeepQ Network is only in the neural network part. In figure 2 the predefined network layers of the DQN are shown, and in figure 3 the layers of the Dueling DQN. The first convolutional layer stays the same: a layer with

The next layer, however, does not consist of a single layer, but of two separate layers that we called v-values and adv-values. Both get the input from the first convolutional layer and both use the ReLU to make it non-linear. The output of the v-value part of the network is a single V-value and the output of the adv-value part is a different value for each possible action in that state.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Figure 3

Figure 2

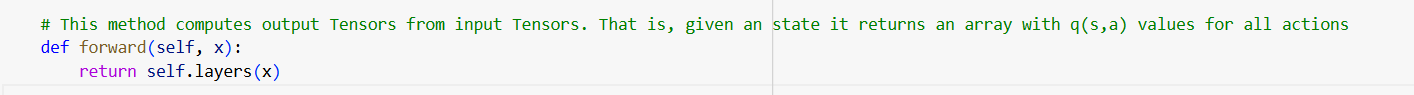
After that the adv-values and v-value should be combined and this is done in the forward function that pushes the given input state via all the layers through the network (figure 5). The output of the adv-values and v-values based on state x are collected and the mean of the adv-value is calculated. Then the formula described above is implemented which returns a q-value for all the different possible actions in the inputted state x. As the output of the neural network is still a Q value so no changes are done to the algorithm part in the rest of the code (figure 4).

Figure 4

Afbeelding met tekst, Lettertype, schermopname

Automatisch gegenereerde beschrijving

Figure 5

**Results of Dueling DeepQ Network**

The results of the dueling DQN can be seen in table 1 below, where they are compared to previously obtained results for the base-, Double-Q- and n-step DQNs. For each of the methods, the agent was trained 10 times. For each of the 10 training rounds, the trained agent was tested by running 10 episodes. The results of the tests were evaluated based on three metrics:

1. *No. Times converged:* Out of the 10 training rounds, this is the amount of rounds where the model converged. The model was considered to have converged once the mean return of the 100 latest rewards was > 195.
2. *Convergence time:* If, during a training round, the model converged, this is the number of episodes that passed before the model converged. This is counted from the start of the 100 episodes whose mean was > 195. For example, if episodes 43-143 are the first 100 consecutive episodes with a mean of > 195, the convergence time is 43 episodes.
3. *Return per episode:* For each training round, the return (cumulative reward) of each episode was averaged. The cumulative reward of an episode is equal to the length (number of steps) of that episode. The episode ends if the pole falls over, the cart moves too far to one side, or if the maximum episode length of 200 steps is reached. This means that the maximum return of an episode is 200, and anything lower means that the agent failed the task of balancing the pole on top of the cart.

**Table 1.** The table shows the results from training the agent 10 times. The table includes the total amount of times the model converged, as well as the means and standard deviations of the convergence time and returns per episode, computed across the 10 training rounds.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | **No. Times converged** | **Convergence time – Mean**  **(No. episodes)** | **Convergence time – Standard deviation**  **(No. Episodes)** | **Return per episode - Mean** | **Return per episode – Standard deviation** |
| Base DQN | 1 | 43.0 | N/A | 193.1 | 13.0 |
| Double-Q DQN | 2 | 37.5 | 10.6 | 190.3 | 20.2 |
| n-step DQN | 2 | 20.0 | 5.7 | 199.6 | 1.4 |
| Dueling DQN | 0 | N/A | N/A | 162.7 | 66.3 |

**Source**

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2016). Dueling network architectures for deep reinforcement learning.