Dependence of the optimal state buffer size on the target length in the Snake problem during Q-learning training

This work studies the effect of the last states buffer size. Specifically, it investigates how it affects an agent trained with Q-learning on the Snake task. The main purpose of this work is to identify the dependence between the desired level that we want to achieve (size of the snake) and the size of the buffer.

This experiment showed that the optimal size of the buffer increases as the desired length goes up. Also, a too large buffer can worsen the results for small goals. We also saw that correlation between local adaptation in the late game exists. Specifically, the table showed that the optimal size of last states buffer shifts upward as the target length increases.

1.1 Snake environment

1. During the game, a player controls a long creature, a snake, which moves on a two-dimensional grid, bounded by walls. The snake collects food and avoids collisions with its own body. Every time the head of the snake is located at the same position as the food, the length of the snake increases by one.
2. Grid size: 10x10
3. The episode ends when the head of the snake goes out of the field boundaries or ends up in the same cell as its tail.
4. The snake's length is the number of cells occupied by the snake (head + body).

1.2 Q-learning algorithm

In this work, the tabular temporal-difference algorithm Q-learning is used as the algorithm that updates the action-value function Q(s, a) using one-step transitions. The actions are selected using the ε-greedy policy. More precisely, with the probability 1 – ε, we choose the action with the maximum Q(s, a). Otherwise, we choose a random action. To calculate the values, we use the maximum over actions in the next state. This is what distinguishes Q-learning from SARSA

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha * (r_t + \gamma * \max_{\{a'\}} Q(s_{\{t+1\}}, a'') - Q(s_t, a_t))$$

During our research, the speed of learning α equals 0.05, the coefficient of the discount factor γ equals 0.95. The parameter ε is initialized to 1, and multiplied by 0.999 after each episode, but it is not allowed to decrease below 0.01 to maintain exploration. The number r_t equals -100 if episode ends during our move (1.1.3). Else if we found an apple (food), it equals 20. Else, it equals -0.01 because we want to prevent endless game.

2. Last states buffer (deque)

During the process of learning, our agent stores a restricted buffer of the last states, which has a fixed length of deque_size. The state is a full description of the current configuration of the game, including positions of the body, head, and food. We will refer to the state as "s".

The buffer is implemented as a sliding window. At each step, our current configuration state is added to the end of the buffer. If the size of the buffer exceeds deque_size, we delete the oldest state using the first-in, first-out (FIFO) rule. This way, we store fresh fragments of experience.

In our implementation, the replay buffer stores the last N state–action pairs (s_t, a_t), where a_t is the action stored in the direct field. Since the environment dynamics (including apple placement) are deterministic given (s_t, a_t), we recover r_t and s_{t+1} during replay and apply the standard Q-learning update.

During our experiment, we will choose last 10 states and 16 random states in the buffer. Also, we will launch the buffer after 10 moves every time.

3. Research hypothesis

Hypothesis
The optimal size of the recent-episode buffer depends on the desired target snake length. For small targets, a smaller buffer speeds up learning by concentrating on typical early-game states, whereas larger targets require a larger buffer to retain rare late-game states.

4. Experimental protocol

4.1 Experimental design

The main purpose of this experiment is to calculate how the size of the sliding buffer influences the speed of achieving the targeted length L when we learn our agent using Q-learning in the Snake environment. The experiment will be conducted as a comparison of configurations based on two parameters: the values of the L and the buffer sizes (deque_size). All other settings will be fixed throughout the experiment.

The following parameter values are considered:

L ∈ {10, 15, 20}, deque_size ∈ {500, 1000, 2000, 5000}.

4.2 Launch protocol

For each (L, deque_size) pair, we run 100 independent training runs, each with a different random seed. In every run, the agent learns from scratch (fresh parameter initialization, cleared Q-table, and an empty buffer), and we record how many episodes it takes to reach the target length L.

4.3 Determining the time to achieve the goal

In each run, we measure T — the number of episodes until the snake first reaches a length of at least L. We count the goal as achieved the first time an episode ends with the maximum length reached during that episode being ≥ L. As soon as that happens, we stop the run and record T.

4.4 Final metric

For each (L, deque_size) configuration, we report a single final metric: the average number of episodes needed to reach the target, computed over 100 runs:

$\mu(L, deque\_size) = (T_1 + T_2 + \ldots + T_{100}) / 100$,

where T_i is the number of episodes it takes to reach the target in the i-th run.
The resulting averages, μ(L,deque_size), are then used to compare configurations and to build the results table.

5. Results

L \ deque_size	500	1000	2000	5000
10	2943.81	3575.42	3620.66	3547.32
15	6860.22	6088.25	5746.88	5950.91
20	9540.17	9222.32	8460.67	8140.54

The table shows the average number of episodes the agent needs to reach the target length L for different deque_size values.

The smallest buffer (deque_size = 500) works best for the short target (L = 10). The agent requires more episodes on average to reach the goal as the size of the buffer grows. So, we can assume that keeping too much past experience can actually slow down learning in tasks with short time frames. The reason can be that this method includes a lot of situations that are not useful. Because of this, we use our updates less efficiently.

The best buffer size shifts towards bigger buffers as the target length increases. For example, when L = 15, the lowest average number of episodes occurs when deque_size = 2000. As we can see, when L = 20, the best result happens when deque_size = 5000, which is the largest buffer in this list.

6. Conclusion

This work investigated the influence of buffer size on the speed of achieving a targeted length in the snake game when training our agent to play using the Q-learning algorithm in the snake environment. The results of this experiment show that for short-term goals, we have to choose a small buffer size because it provides fast adaptation. For long-term goals, we should choose bigger buffer sizes, because they offer the ability to keep past states. The experiment included a 10x10 grid, and the reinforcement learning was implemented in C++ without the use of supplemental AI libraries.