Zadanie uczenie przez wzmocnienie (ε-greedy QLearning) - raport

W celu polepszenia wyników osiąganych przez algorytm QLearning implementację klasy QLearningAgent rozszerzamy o informację nt. położenia ścian i ogona węża względem jego głowy. W tym celu do kod metody game_state_to_observation modyfikujemy następująco:

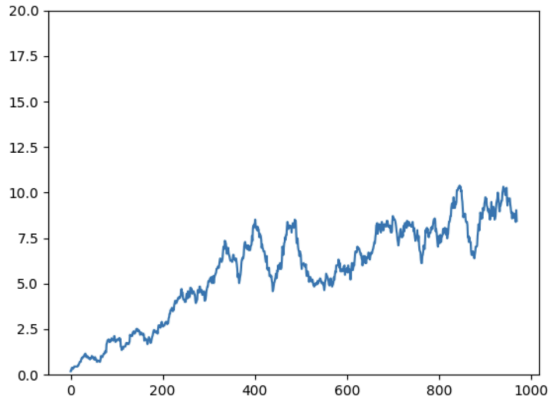
```
# Define relative coordinates of each adjacent to head field
relative_coords = [(-30, -30), (0, -30), (30, -30), (30, 0),
                   (30, 30), (0, 30), (-30, 30), (-30, 0)]
head_x, head_y = gs["snake_body"][-1]

tail = [0] * 8
# Check for each adjacent field if there is a tail
for i, (dx, dy) in enumerate(relative_coords):
    if (head_x + dx, head_y + dy) in gs["snake_body"][:-1]:
        tail[i] = 1

wall = [0] * 8
bounds_x, bounds_y = gs["bounds"]
for i, (dx, dy) in enumerate(relative_coords):
    if (head_x + dx < 0) or (head_x + dx >= bounds_x) or \
            (head_y + dy < 0) or (head_y + dy >= bounds_y):
        wall[i] = 1

return (is_up, is_right, is_down, is_left, *tail, *wall, gs["snake_direction"].value)
```

Wówczas długość zwracanego wektora zwiększa się o 16 i zawiera dodatkowe binarne wartości informujące o obecności ściany lub ogona na każdym z 8 pól sąsiadujących z polem na którym znajduje się głowa węża. Zmianie ulega również rozmiar Q-tabeli:

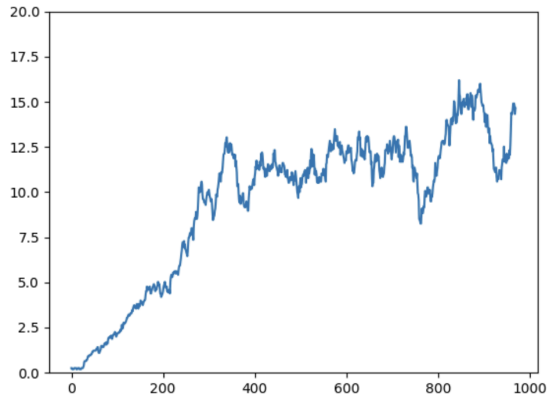
```
qtable_size=(2, 2, 2, 2, *(2 for i in range(16)), 4, 4)
```

Następnie wykonane modyfikacje testujemy dla różnych wartości ε i dyskonty, wykresy średniej wartości nagrody za epizod podczas treningu prezentują się następująco:

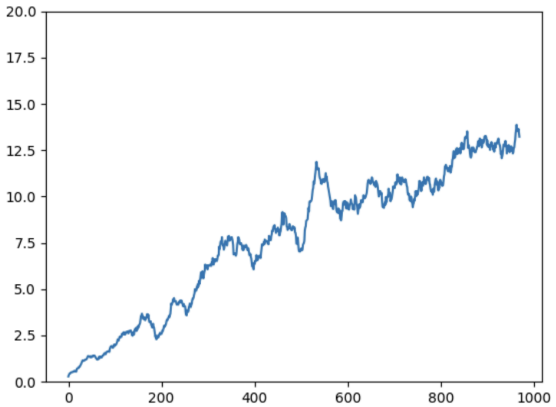
eps: 0.2 discount: 0.99 learning_rate=0.01



eps: 0.1 discount: 0.8 learning_rate=0.018

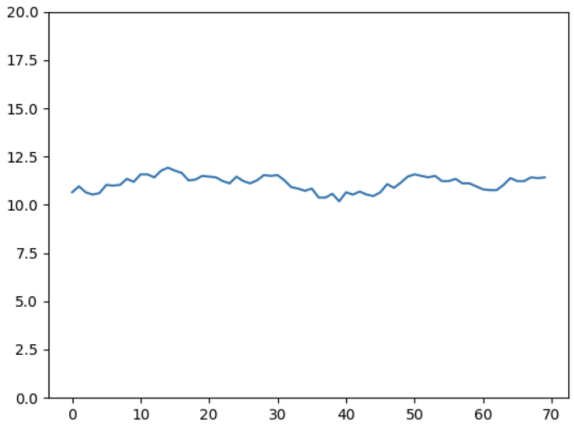


eps: 0.002 discount: 0.95 learning_rate=0.01

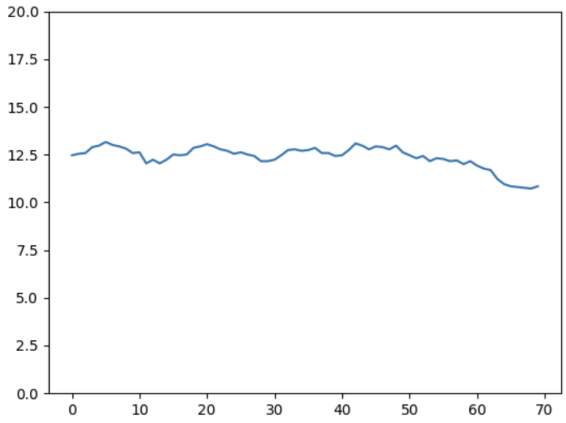


A tak dla średniej wartości podczas testowania wytrenowanych funkcji:

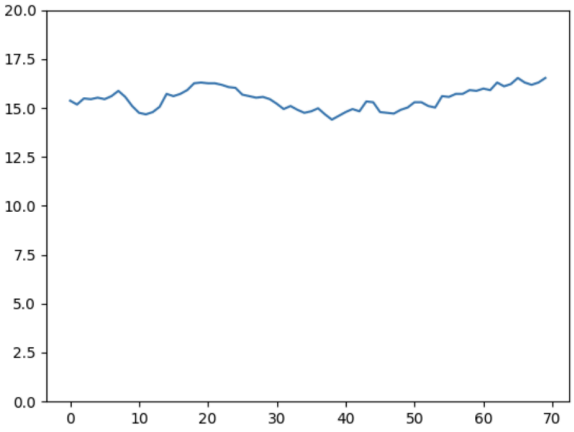
eps: 0.2 discount: 0.99 learning_rate=0.01



eps: 0.1 discount: 0.8 learning_rate=0.018



eps: 0.002 discount: 0.95 learning_rate=0.01



Eksploracja a eksploatacja (exploration-exploatation trade-off):

Odnosi się do sposobu, w jaki agent decyduje, czy eksplorować nowe stany i akcje, czy też eksploatować już znane informacje w celu maksymalizacji swojej funkcji wartości. W algorytmie Q-learning agent uczy się podejmować decyzje w dynamicznym środowisku poprzez eksplorację różnych akcji i obserwowanie ich skutków lub eksploatację najlepszych znanych akcji, bazując na aktualnie oszacowanych wartościach Q (funkcji wartości). Trade-off eksploracji i eksploatacji w Q-learningu jest zazwyczaj reprezentowany przez parametr zwany "epsilon-greedy". Ten parametr kontroluje, w jakim stopniu agent będzie eksplorował nowe akcje (przez losowy wybór) w porównaniu do eksploatowania najlepszej znanej akcji (poprzez wybór akcji o najwyższej wartości Q).