Sieci samouczące się

Laboratorium 6

Przemysław Rośleń 180150

Problem parkowania samochodu rozwiązałem stosując algorytm aproksymacji liniowej z kodowaniem metodą prototypową i aktualizacji wag zgodnie z metodą TD(λ) z użyciem śladów aktywności.

Poniżej przedstawiłem:

- opis metody TD(λ) z użyciem śladów aktywności
- istotne fragmenty kodu
- najskuteczniejsze wartości hiperparametrów
- wyniki w postaci wizualizacji zachowania samochodu
- wykres zależności końcowych wartości średnich sum nagródi od wartości współczynnika świeżości

Generowanie prototypów:

```
def generate_prototypes(self):
    prototypes = np.random.rand(self.hiper_parameters.num_of_prototypes, 3)
    prototypes[:, 0] *= self.global_variables.street_length
    prototypes[:, 1] *= self.global_variables.street_width
    prototypes[:, 2] = np.random.uniform(-np.pi, np.pi, self.hiper_parameters.num_of_prototypes)
    return prototypes
```

Kodowanie prototypów:

```
def get_state_projections(self, state: State):
    distances = np.linalg.norm(self.prototypes - (state.x, state.y, state.car_angle), axis=1)
    close_indices = np.where(distances <= self.hiper_parameters.r)[0]
    return close_indices

def encode_state(self, state: State, action):
    coded_state = np.zeros(shape=self.get_weights_shape())
    projections = self.get_state_projections(state)
    for projection in projections:
        coded_state[projection, action] = 1.0
    return coded_state.reshape(-1)
```

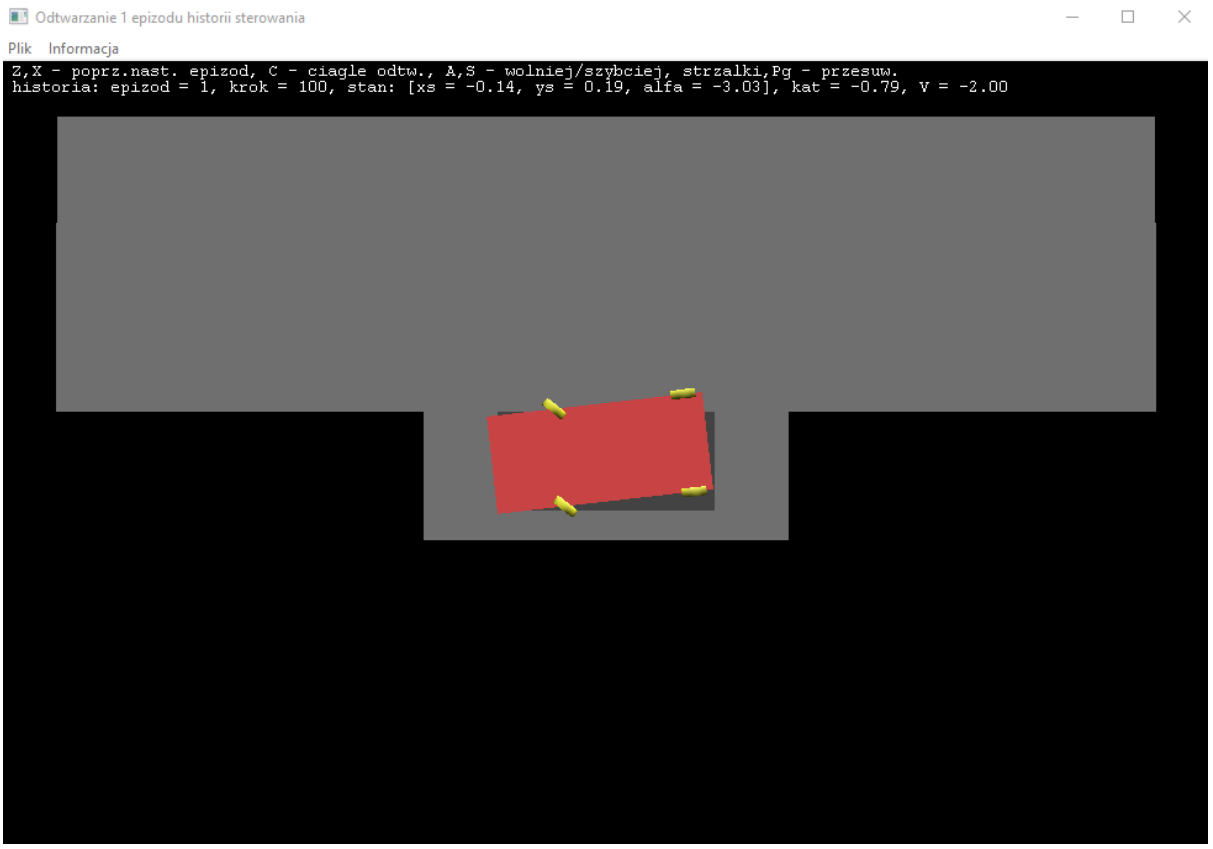
Aktualizacja wag zgodnie z algorytmem TD() + ślady aktywności:

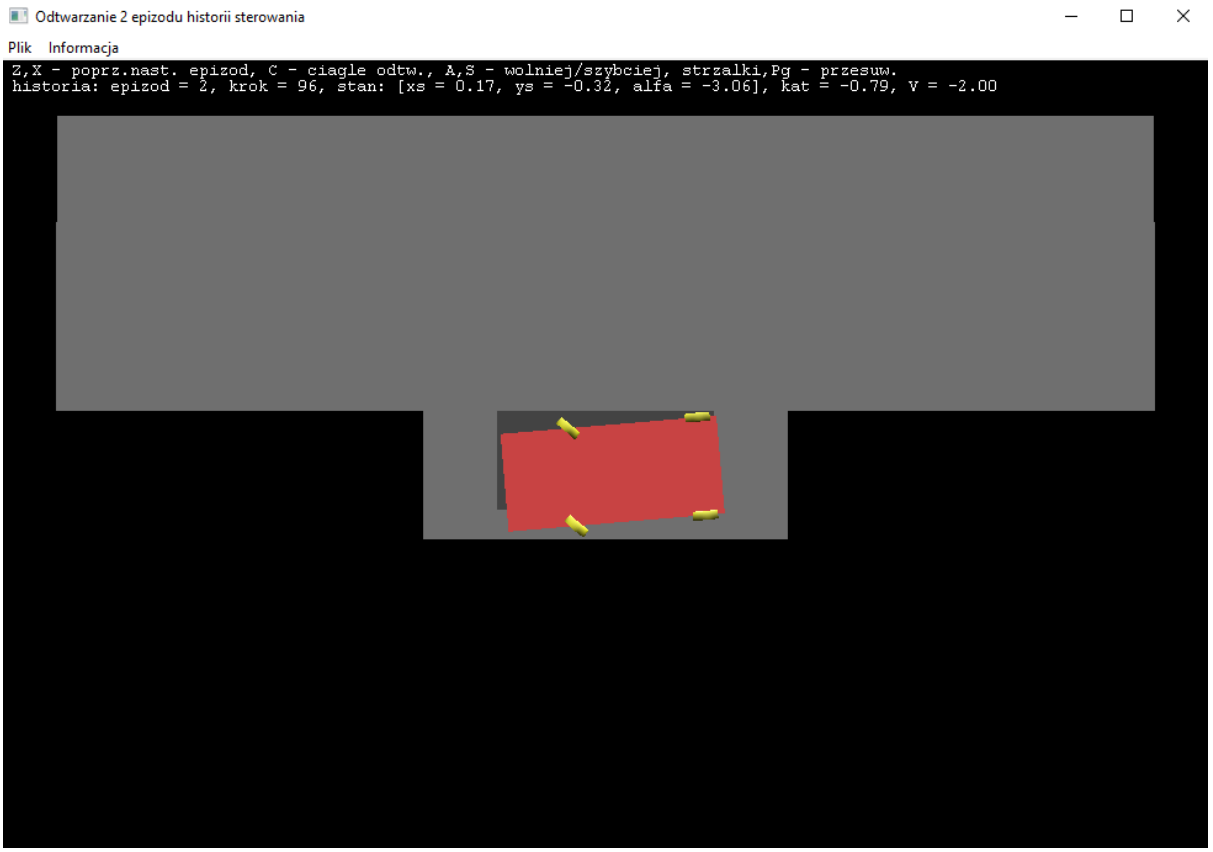
```python
def update_weights_Q_learning(encoder, state, new_state, action, action_prim, weights, alpha, gamma, reward, lambda_coeff, z):
    coded_state = encoder.encode_state(state, action)
    coded_state_prim = encoder.encode_state(new_state, action_prim)
    z = (gamma * lambda_coeff * z) + coded_state
    delta = reward + (gamma * Linear_Approximator.approximate(weights, coded_state_prim)) - Linear_Approximator.approximate(weights, coded_state)
    weights += (alpha * delta * z)
    return z, weights
```

Hiperparametry:

- liczba wartości kąta samochodu: 9
- liczba wartości kąta kół: 9
- wartości prędkości: [-2, 2]
- liczba prototypów: 2000
- promień: 1
- liczba epok: 1000
- alpha: 0.1
- wartość epsylon: 0
- zanik epsylon na epokę: 0
- gamma: 0.95
- współczynnik świeżości (lambda): 0.3

Efekty uczenia:





Wykres zależności końcowych wartości średnich sum nagród od wartości współczynnika świeżości: