Oscar Petersson, Johan Levinsson

TDDC17 Lab 5 - Reinforcement Learning

Högskoleingenjörsutbildning i datateknik, 180 hp

Laboration report - October 11, 2016
Artificial Intelligence
TDDC17, Linköpings universitet

Assistant:
Josef Fagerström
IDA

Part II

For Part II, there is only one task-specific question: *"What tends to happen if we try turning off exploration before learning, and why?"*

Turning exploration off with angle-based reward only, usually results in the rocket doing nothing and dropping straight down as firing rockets will tilt the rocket resulting in a lower reward.

Part III

States and rewards

States

Defining the state space is non-trivial. A large state space gives the possibility of learning more accurate and specific behaviours but at an increasing time and space cost. However, reasonably good results can be achieved already with a small state space. We went with a larger state space than necessary to achieve a result *we* were satisfied with, and still within the task's soft-cap of 500 000 iterations.

The state space is a combination of three variables (for Part III, part II has states based on angles alone): speeds in two dimensions, and the rocket's angle. The function `discretize` offers a set of "buckets" into which we can drop intervals of values. This is used to define the states.

For the angle, we went with a larger set of buckets as even very small variations in the angle gives large deviations in lateral speed when activating the thrusters. This set of buckets were then distributed over a fairly small angle range. This results in the rocket having issues coping accurately with large deviations, but being "very" stable once hovering.

As it turns out, due to gravity, accelerating downwards happens much more easily than upwards. The speed states were defined to a small interval with smaller sets, with the y-axis velocity interval being biased towards positive velocity. This in order to limit the state space while maintaining reasonable precision.

Rewards

The reward system could be implemented in various ways with similar outcomes. We decided to go with a model with exponentially increased reward the closer to a perfect hover we come.

We wished for the angle to be pointing upwards, as the propulsion and gravitation combination makes a hover upside-down unlikely for longer periods of time. Therefore, `getRewardAngle` will return the reward r as a function of α:

$$r_\alpha = \pi^\pi - \pi^{|\alpha|}$$

For Part III, we need also take the velocities into consideration. We do this by multiplying the inverse sum of the velocities v_x, v_y (plus a small value to avoid division by zero) with the angle reward from `getRewardAngle`:

$$r_v = \frac{1}{|v_x| + |v_y| + 0.0001} * r_\alpha \Leftrightarrow r_v = \frac{r_\alpha}{|v_x| + |v_y| + 0.0001}$$

Q-learning update

```
1  Qtable.put(
2    prev_stateaction,
3    Qtable.get(prev_stateaction)
4      + alpha(Ntable.get(prev_stateaction))
5      * ( previous_reward +
6      GAMMA_DISCOUNT_FACTOR * getMaxActionQValue(new_state)
7      - Qtable.get(prev_stateaction) )
8    );
```

$$Q(s_t, a_t) \rightarrow Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right)$$

In `Qtable` we store a Q-value along with the states and actions resulting in the Q-value. We update these Q values over time by adding (3+) what we've just learned.

The factor `alpha` is based on the learning-rate constant and number of iterations of the state-action combination, resulting in each iteration of the combination being less valuable than the previous iteration. A smaller constant lets us converge to a behavoiur faster, but with a large state space, we risk—just like the comments states—ending up with a poor result if the state space is not explored properly.

The γ-factor times $\max_a Q(s_{t+1}, a)$ defines the importance of future rewards for learning. A low value will cause "short-sightedness", and a higher will aim for high long-term rewards.