# Algorithm

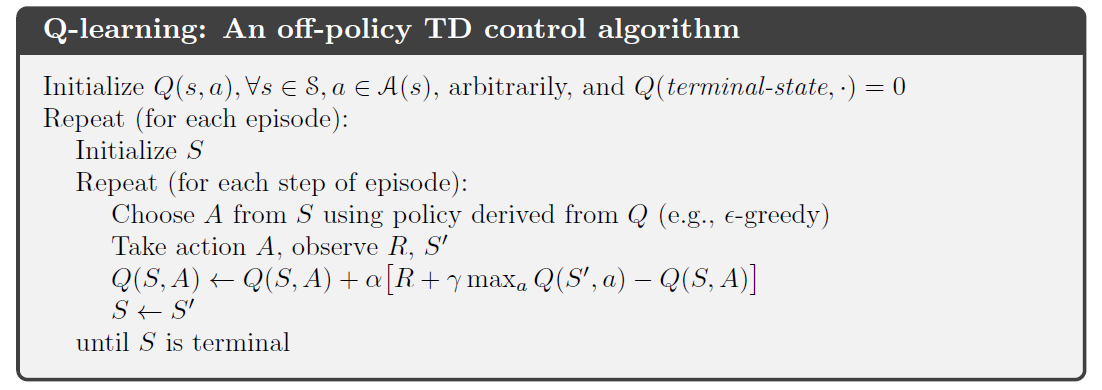
In recent years, reinforcement learning has achieved many impressive results, including playing video games, and training advanced manipulation skills. The goal of this experiment is to realize artificial agents, which can be taught and can find the shortest path in intricate traffic environment. Here we consider using An off-policy TD control algorithm: Q-learning to achieve our goal.

We define a discrete-time finite-horizon discounted Markov decision process(MDP) by a tuple , in which S is a state set, A an action set, r a reward function,  a discount factor.

The agent interacts with an environment over a time of discrete time steps. At each time step, the agent receives a state from S and select an action from A according to its policy , where is a mapping from states S to action A. In return, the agent receives the next state and a scalar reward r. The process continues until the agent reaches a terminal state. The total return is the accumulated reward with discount factor . The action value  is the expected return for selecting action a in state s and following policy The optimal value function gives the maximum, action value for state s and action a achievable by any policy.

In Q-learning, the action value will be updated toward one-step return as below:

## 这公式上次你做了

Each time obtaining a reward r can directly affects the value of the state action pair s,a that led to the reward. The pseudocode of Q-learning algorithm is shown in Algorithm 1. 

**Algorithm 1. Q-learning**

At each step, the agent will choose an action A according to current policy, and get the reward R. But the action value Q will be updated using another policy, here we choose the action which can maximize the next action value. That’s why Q-learning is an off-policy algorithm, the policy we use to choose action and the policy we use to evaluate the value function are different. Here we prefer to the off-policy rather than on-policy algorithm, because off-policy algorithms are usually more efficient than on-policy.

# Simulation

The goal of our simulation is to simulate the real traffic network. The e-puck will take on a role of explorer to find the shortest path between two points of the map.

To make our tasks tractable, we consider a grid-based representation with discrete states and action. The navigational task for our mobile robots could be projected into this presentation by employing a number of actions such as “turn left”, “turn right”, and “drive forward” that only use a low-level controller that takes care of accelerating, moving, and stopping. The different crossroad in the simulated map will be considered as states. We will perform our experiment in the simulator v-rap with our test map, which has 10 states.

In reinforcement learning, the desired behavior is implicitly specified by the reward function. The goal of our reinforcement learning algorithm is to find the optimized trajectory, which can maximize the accumulated long-term reward. In our experiment, we assign a reward ‘+10’ at the destination state and ‘-1’ at the other states. Also at the ‘blind alley’ it will get a very

poor reward such as ‘-10’ in order to let agent avoid those states.

##此处应有模拟图

At the begin of each simulation, two points will set up as the start point and destination, and the e-puck robot will be placed in the start point of the test map, and start to explore the test map. From one state to adjacent state, the robot will drive along the line we drawn on the ground. The under sensor on e-puck will scan the ground all the time in order to keep robot following the line and detect the label we use to distinguish state. Everytime the robot reaches a new state, the Q-value will be updated with the reward we received and a new action will be selected according to the current policy. Once e-puck reaches the terminal state(destination), it will be replaced in the same start point and start a new exploration. Through trial and error, finally the action value will converge at a point, which is called optimal action value, then we can find the optimal policy for this problem.