Reinforcement learning for Adaptive Traffic Light

Problem Setup:

This project aims to utilize reinforcement learning algorithms to regulate \ multiple traffic signal phases, optimizing traffic flow within regions. Our system can react in real-time, alleviating traffic congestion. To enhance the system's intelligence, the system incorporates technologies such as Deep Q-network (DQN) and Prioritised Experienced Replay (PER). These comprehensive approaches, considering various traffic scenarios, makes traffic control system more flexible and efficient.

This project is a Markov Decision Process (MDP) problem. MDP is a discrete time stochastic control process with (S, A, R, S’) pairs; At each time step, the stochastic process is in some state S. The decision-maker can choose an action A available in state S. The random process will randomly enter the new state, S' in the next time step and give the decision maker corresponding feedback R (s, s').

Reinforcement Learning: Reinforcement learning is learning what to do under MDP, to maximize a numerical reward signal. This project follows the idea of reinforcement learning and tries to allow more vehicle pass through junction in given time.

Env: The environment in the project is consist of 1 junction and 12 traffic lanes with different direction, each traffic lane is controlled by a single traffic light. /Add figure.

Notation in this environment:

State: vehicle density of each traffic lane

Action: change the traffic light phase or not (red or green)

Reward: Choose the lane with greatest density, Turn its light green.

(show simulator)

Methodology:

This project mainly applies Deep Q-Network (DQN), which is one approach to approximate the Q-learning algorithm using neural networks to estimate the value function. To see more improvement based on DQN, Prioritized Experience Replay (PER) is applied on trained DQN. PER is a training method that improves the DQN algorithm and improves learning efficiency by giving important experiences a higher sampling probability.

Compare Models: (show results and analysis)

DQN and most Random:

Total return in each episode is lower in Random selecting action, also with high fluctuation. Higher return in DQN with decreasing exploration rate. With more training time, there is convergence and better result.

DQN with PER and no PER:

Convergence comes in fewer episode with PER implementation and more stable convergence when training more time.

DQN with PER, but different exploration rate: (Fix and not Fix)

FIXING epsilon comes with higher return, a convergence can be seen but not that stable compare with changing exploration rate.

DQN with PER but with different TAU:

For decreasing epsilon, the declining return is lower When TAU is larger at 0.1 compared to TAU = 0.05, which means it will converge at a hight return.

Changing Goals:

From multi-agent to Apply DQN, DQN takes long time to see the result and convergence. For multi-agent, take even longer time to train and see the convergence, because each agent needs to be trained and collaborate with each other. Required high level of hardware. Apply improved DQN (with PER) first. See the improvement what how it is improved.

More Implementation