Reinforcement learning for Adaptive Traffic Light

In this project, reinforcement learning algorithms are applied on traffic light system to allow more waiting vehicles pass through the traffic intersection, this traffic light system is called Adaptive Traffic Light. This adaptive traffic light learns from the experience over the time period and make a 'good' decision under a optimal rule to set traffic phases. Though out the project, a proper traffic system reinforcement learning environment is built, testing are made based on this environment. Reinforcement learning algorithms like Deep Q-Network (DQN) and its improvement Double DQN are applied, compared with Actor-Critic and non reinforcement learning algorithm, greedy algorithm. This report provides general definitions of these algorithms and how they work in the reinforcement learning environment of transportation systems. Experiment result through out the project shows that DQN helps improve the efficiency of the urban transportation, the improvement in DQN algorithms can improve the efficiency of the algorithm but it is hard for to increase final result of the algorithm. 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

When talking rewards computing:

From the idea of bellman equation, total reward will be got and maximised as the agent makes good decisions all the time under the optimal policy. In this case, this ATL assume if adaptive traffic light can allow most of vehicles pass through the junction, rewards can be maximised.

Talk about assigning different weights at the beginning. Most of thesis of RL learning by average.

The presentation could have done more to relate the findings back to the motivating scenario: are there simple policies that can be easily implemented? What information do traffic lights need in order to improve throughflow?(in ENVIRONMENT) How much improvement over the current state of the art is possible?(in EXPERIMENTS) The presentation was a bit unpolished to begin with, but improved a lot, particularly towards the end of the Q&A, when more expertise on the topic was demonstrated

Research:

Cartpole: Cartpole is a very classic and standard reinforcement learning environment build by Open AI. in gymnasium library. It is firstly used to verify the correctness of Deep Q network algorithms which will be applied on adaptive traffic light.

Cartpole的环境是什么样的，为什么标准的强化学习环境。

1. Cartpole的目的是为了让小车上的木棍能立住不会倒：立住的时间越长，总体的reward就会越高，因此reinforcement learning在Cartpole环境中的policy就是通过不断改变小车移动的方向
2. Cartpole中的状态是连续的，动作是往左右两个方向移动小车来平衡木棍。Sensitive to强化学习的算法设计

Project Management

Base

Parameter1: DQN(env = env,mode="train",input\_dim=input\_dim,output\_dim=output\_dim,gamma=0.9,replay\_size=20000,batch\_size=256,eps\_start=0.95,eps\_end=0.05,eps\_decay=30000,LR=1e-4,TAU=0.005)

Change gamma

Parameter2: DQN(env = env,mode="train",input\_dim=input\_dim,output\_dim=output\_dim,gamma=0.95,replay\_size=20000,batch\_size=256,eps\_start=0.95,eps\_end=0.05,eps\_decay=30000,LR=1e-4,TAU=0.005)

Change batch size

Parameter3: DQN(env = env,mode="train",input\_dim=input\_dim,output\_dim=output\_dim,gamma=0.9,replay\_size=20000,batch\_size=64,eps\_start=0.95,eps\_end=0.05,eps\_decay=30000,LR=1e-4,TAU=0.005)

Change TAU

Parameter4: DQN(env = env,mode="train",input\_dim=input\_dim,output\_dim=output\_dim,gamma=0.95,replay\_size=20000,batch\_size=256,eps\_start=0.95,eps\_end=0.05,eps\_decay=30000,LR=1e-4,TAU=0.1)

Change LR

Parameter5: DQN(env = env,mode="train",input\_dim=input\_dim,output\_dim=output\_dim,gamma=0.95,replay\_size=20000,batch\_size=256,eps\_start=0.95,eps\_end=0.05,eps\_decay=30000,LR=1e-3,TAU=0.005)

Change epsilon

Parameter6: DQN(env = env,mode="train",input\_dim=input\_dim,output\_dim=output\_dim,gamma=0.95,replay\_size=20000,batch\_size=256,eps\_start=0.95,eps\_end=0.95,eps\_decay=30000,LR=1e-4,TAU=0.005)

Greedy

Parameter2: DQN(env = env,mode="train",input\_dim=input\_dim,output\_dim=output\_dim,gamma=0,replay\_size=20000,batch\_size=256,eps\_start=0.95,eps\_end=0.05,eps\_decay=30000,LR=1e-4,TAU=0.005)

DQN TLENV mean vehicles 958.552