Topic： Deep reinforcement learning

Problem 1: TSP + path planning（obstacles）

问题难度（有待进一步论证）：多任务调度（NP）、局部最优化（避障）、全局最优化……
两个问题结合到一起产生了什么新的挑战。

Related works:

Work[1]: solving the appropriate travelling salesperson problem (Heuristic) & a roadmap of free-space（启发+经典路径规划）
Work[2]: Genetic Algorithm (GA) and A* to solve Traveling Salesman Problem (TSP) （启发+经典路径规划）
Work[3]: soft q-learning multimodal exploration strategies（RL做抓取任务）

Idea:

Multimodal deep reinforcement learning (如何设计网络架构)

Concrete works: theoretical analysis & Experimental verification

[1]Spitz S N, Requicha A A G. Multiple-goals path planning for coordinate measuring machines[C]//Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065). IEEE, 2000, 3: 2322-2327.
[2] Bolourian N, Hammad A. Path Planning of LiDAR-Equipped UAV for Bridge Inspection Considering Potential Locations of Defects[M]//Advances in Informatics and Computing in Civil and Construction Engineering. Springer, Cham, 2019: 545-552.
[3]Haarnoja T, Pong V, Zhou A, et al. Composable deep reinforcement learning for robotic manipulation[C]//2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018: 6244-6251.

Problem 2: Problem 1 + Dynamic

- 在Agent移动过程中，有新的目标节点加入或已规划的节点消失。
- 环境是动态的（障碍物的消失或出现）
- 某条线路处有拥堵（随机延时）

Related works:

Work[1]: the genetic algorithm (GA) and applied to the path planning problem of mobile robots in dynamic environments(启发式)

Work[2]: The proposed path planning method determines not only an optimal path, but also the appropriate acceleration and speed for a vehicle

Work[3]: Socially Adaptive Path Planning in Human Environments Using Inverse Reinforcement Learning强化（逆学习）

[1]Tuncer A, Yildirim M. Dynamic path planning of mobile robots with improved genetic algorithm[J]. Computers & Electrical Engineering, 2012, 38(6): 1564-1572.
[2]Hu X, Chen L, Tang B, et al. Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles[J]. Mechanical Systems and Signal Processing, 2018, 100: 482-500.
[3] Kim B, Pineau J. Socially adaptive path planning in human environments using inverse reinforcement learning[J]. International Journal of Social Robotics, 2016, 8(1): 51-66.

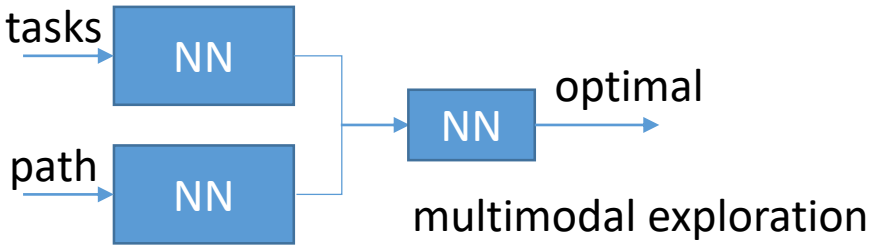
Problem 2: Problem 2 + Multi-objective

- 距离，时间，耗电量（货物）

Related works:

Work[1]:A simulation-based approach by means of particle swarm optimization (PSO)（粒子群优化）
Work[2]:a Multi-Objective Grey Wolf Optimizer (MOGWO) 优化求解器
Work[3]:Deep Optimistic Linear Support Learning (DOL) to solve high-dimensional multi-objective decision problems（DRL）

Idea:



[1] Delgarm N, Sajadi B, Kowsary F, et al. Multi-objective optimization of the building energy performance: A simulation-based approach by means of particle swarm optimization (PSO)[J]. Applied energy, 2016, 170: 293-303.
[2] Mirjalili S, Saremi S, Mirjalili S M, et al. Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization[J]. Expert Systems with Applications, 2016, 47: 106-119.
[3] Mossalam H, Assael Y M, Roijers D M, et al. Multi-objective deep reinforcement learning[J]. arXiv preprint arXiv:1610.02707, 2016.

问题背景：DQN迷宫多目标最短路径，在reward值收敛后，有震荡，寻找震荡原因。

理论：q_learning算法一定收敛于最优解处。

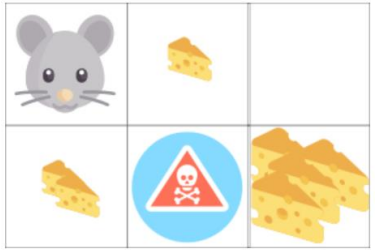
Watkins C J C H, Dayan P. Q-learning[J]. Machine learning, 1992, 8(3-4): 279-292.

实际：q_learning算法epsilon-greed探索，不可靠收敛，epsilon、学习率参数有关

https://stats.stackexchange.com/questions/206944/how-do-i-know-when-a-q-learning-algorithm-converges

Q-learning实例（rat-maze）：

环境



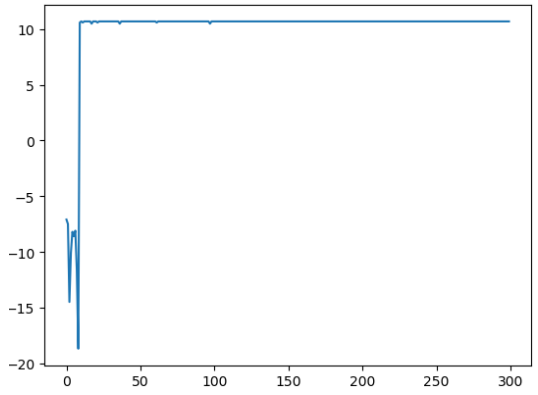
状态、动作

	←	→	↑	↓
Start	0	0	0	0
Small cheese	0	0	0	0
Nothing	0	0	0	0
2 small cheese	0	0	0	0
Death	0	0	0	0
Big cheese	0	0	0	0

Reward

One cheese： +1
Two cheese: +3
Big pile of cheese: +10
(end of the episode)
Trap：-10
(end of the episode)
Moving: -0.1

结果（episode-reward 收敛图）：



Best reward: 10.7,
action: ['right', 'right', 'down']

问题：
reward值并没有收敛于最优解处。

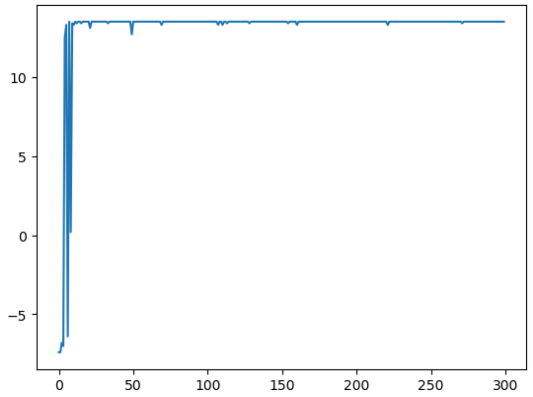
状态信息不全（6*8(cheese or not)）

	←	→	↑	↓
Start	0	0	0	0
Small cheese	0	0	0	0
Nothing	0	0	0	0
2 small cheese	0	0	0	0
Death	0	0	0	0
Big cheese	0	0	0	0

one	two	big
False	False	False
False	False	True
False	True	False
False	True	True
True	False	False
True	False	True
True	True	False
True	True	True

结果

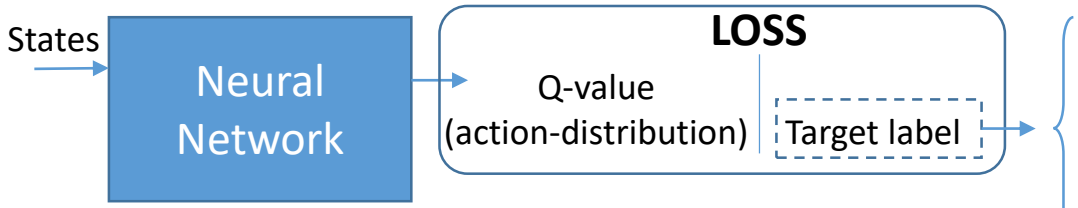
（episode-reward 收敛图）：



Best reward: 13.5,
action: ['down', 'up', 'right', 'right', 'down']

问题：
DQN收敛吗？
收敛后震荡的原因……

DQN收敛吗？
Q-loss, reward



- Supervise learning: a fixed data distribution
- DQN: the data must be sampled on the most recent policy

You Should Know

Even though we describe this as a loss function, it is **not** a loss function in the typical sense from supervised learning. There are two main differences from standard loss functions.

1. The data distribution depends on the parameters. A loss function is usually defined on a fixed data distribution which is independent of the parameters we aim to optimize. Not so here, where the data must be sampled on the most recent policy.

2. It doesn't measure performance. A loss function usually evaluates the performance metric that we care about. Here, we care about expected return, $J(\pi_\theta)$, but our "loss" function does not approximate this at all, even in expectation. This "loss" function is only useful to us because, when evaluated at the current parameters, with data generated by the current parameters, it has the negative gradient of performance.

But after that first step of gradient descent, there is no more connection to performance. This means that minimizing this "loss" function, for a given batch of data, has *no* guarantee whatsoever of improving expected return. You can send this loss to $-\infty$ and policy performance could crater; in fact, it usually will. Sometimes a deep RL researcher might describe this outcome as the policy "overfitting" to a batch of data. This is descriptive, but should not be taken literally because it does not refer to generalization error.

We raise this point because it is common for ML practitioners to interpret a loss function as a useful signal during training—"if the loss goes down, all is well." In policy gradients, this intuition is wrong, and you should only care about average return. The loss function means nothing.

$$NewQ(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma max\, Q'(s',a') - Q(s,a)]$$

- $NewQ(s,a)$: New Q value for that state and that action
- $Q(s,a)$: Current Q value
- α: Learning Rate
- $R(s,a)$: Reward for taking that action at that state
- γ: Discount rate
- $max\, Q'(s',a')$: Maximum expected future reward **given the new s' and all possible actions at that new state**

Experience Replay

https://medium.com/free-code-camp/an-introduction-to-deep-q-learning-lets-play-doom-54d02d8017d8

降低动作的相关性，避免动作灾难性的震荡和不收敛……

策略梯度……（非DQN）

OpenAI: https://spinningup.openai.com/en/latest/spinningup/rl_intro3.html

Topic： 从单体到多智能体

- 获利（分布式架构）：

1. 加快学习速度，经验分享（示教，模拟）；
2. 系统更稳定（鲁棒性），个别agent任务失败（接管）；

- 挑战：

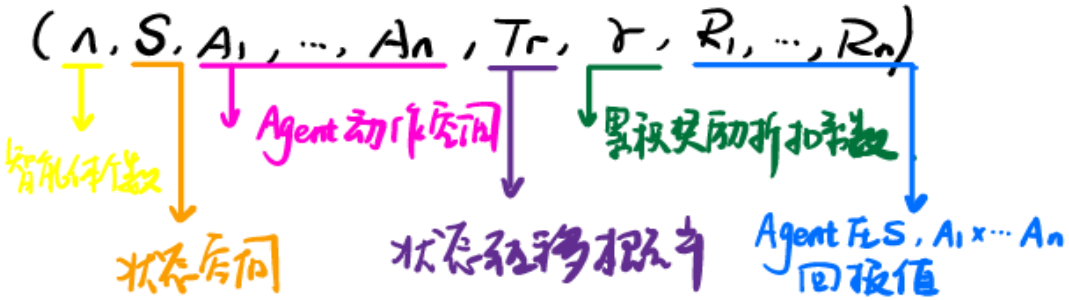
1. 维度爆炸，联合动作空间随agent个数指数增长；
2. Reward难以设计，agent的任务不同，且相互影响；
3. 协同，agent个体动作对环境的改变，依赖于其他agents；
4. 探索/利用策略，不仅针对环境，还针对其他agent；

- 几个博弈的概念和随机博弈（stochastic game）

1. 静态博弈（static game），不存在状态转移，回报值只由agent动作决定；
2. 阶段博弈（stage game），是随机博弈的组成成分，状态s是固定的，相当于一个状态固定的静态博弈，随机博弈中的Q值函数就是该阶段博弈的奖励函数。**若干状态的阶段博弈组成一个随机博弈**。
3. 重复博弈（repeat game），智能体重复访问同一个状态的阶段博弈，并且在访问同一个状态的阶段博弈的过程中收集其他智能体的信息与奖励值，并学习更好的Q值函数与策略。
4. **随机博弈(stochastic game / Markov game)**是马尔可夫决策过程与矩阵博弈（只有一个状态，静态博弈）的结合，具有多个智能体与多个状态，即多智能体强化学习。

- MARL随机博弈过程描述

将随机博弈过程分解为多个阶段博弈。
与环境交互更新每一个状态阶段博弈中的Q值。
马尔可夫决策过程



$$Tr: \quad S \times A_1 \times \cdots \times A_n \times S' \to [0,1]$$

$$R_i: \quad S \times A_1 \times \cdots \times A_n \times S' \to \mathbb{R}$$

- MARL 目标

  每个状态下的**纳什均衡策略**（最优策略）

$$V_i(\pi_1^*, \pi_2^*, \cdots, \pi_i^*, \cdots, \pi_n^*) \geq V_i(\pi_1^*, \pi_2^*, \cdots, \pi_i, \cdots, \pi_n^*)$$

$$\forall \pi_i \in \Pi_i \quad i = 1, \cdots, n$$

所有智能体的联合策略

在纳什均衡处，对于 All Agent 都不能改变自己策略从而获得更大的收益。

- MARL最优策略数学描述

$(\pi_1^*, \pi_2^*, \cdots, \pi_i^*, \cdots, \pi_n^*) \quad i = 1, \cdots, n$

对于 $\forall s \in S$ 满足

① 对于状态价值函数 $V_i(s)$

$$V_i(\pi_1^*, \pi_2^*, \cdots, \pi_i^*, \cdots, \pi_n^*) \geq$$

$$V_i(\pi_1^*, \pi_2^*, \cdots, \pi_i, \cdots, \pi_n^*)$$

$$\forall \pi_i \in \Pi_i \quad i = 1, \cdots, n$$

② 对于状态—动作价值函数 $Q_i(s, a_1, \cdots, a_n)$

$$\sum_{a_1 \cdots a_n \in A_1 \times \cdots \times A_n} Q_i^*(s, a_1, \cdots, a_n) \pi_1^*(s, a_1) \cdots \pi_i^*(s, a_i) \cdots \pi_n^*(s, a_n) \geq$$

$$\sum_{a_1 \cdots a_n \in A_1 \times \cdots \times A_n} Q_i^*(s, a_1, \cdots, a_n) \pi_1^*(s, a_1) \cdots \pi_i(s, a_i) \cdots \pi_n^*(s, a_n)$$

状态价值函数—动作状态价值函数

Bellman 公式

$$Q_i^*(s, a_1, \cdots, a_n) = \sum_{s' \in S} T_r(s, a_1, \cdots, a_n, s') \left[R_i(s, a_1, \cdots, a_n, s') + \gamma V_i^*(s') \right]$$

$$V_i^*(s) = \sum_{a_1 \cdots a_n \in A_1 \times \cdots \times A_n} Q_i^*(s, a_1, \cdots, a_n) \pi_1^*(s, a_1) \cdots \pi_n^*(s, a_n)$$

如何得到最优策略呢？

- MARL 算法

从Task类型和 agent awareness 两个维度分析

① fully cooperative ($R_1 = \cdots = R_n$)

② fully competitive ($n=2$ $R_1 = -R_2$)

③ Mixed

1. Independent（聚焦稳定性的算法）
2. Aware（对其他代理的适应能力）
3. Tracking（只考虑稳定性，不考虑收敛）

对稳定和适应的理解

Stability：趋于平稳的策略。必须性

Adaptation：其他Agent改变策略时保持或改善性能。合理性

自我play，最优策略，Nash equilibrium

所谓收敛到纳什均衡与动态SG的性能之间的联系尚不清晰。

算法分类

BREAKDOWN OF MARL ALGORITHMS BY TASK TYPE AND DEGREE OF AGENT AWARENESS

Task type → ↓ Agent awareness	Cooperative	Competitive	Mixed
Independent	coordination-free	opponent-independent	agent-independent
Tracking	coordination-based	—	agent-tracking
Aware	indirect coordination	opponent-aware	agent-aware

Team Q-learning	Minimax-Q	CE-Q
Coordination graph		Hyper-Q
FMQ		IGA

Multi-agent Reinforcement Learning

从深度强化学习到多智能体

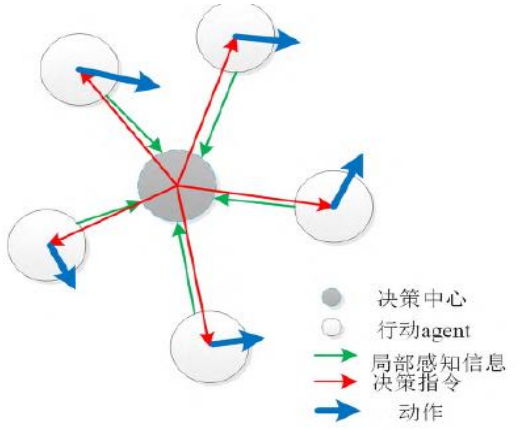
好处：

1. 天生并发，大量样本；
2. 并行计算，提高学习效率；
3. 神经网络，内部通信；

坏处：

1. 动作空间指数上升；
2. 难以指定学习目标；
3. 多agent学习，打破环境平稳性；

MARF算法分类



(a) 全通信集中决策示意图

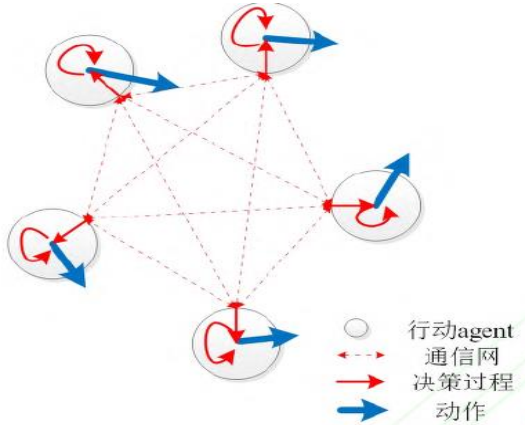
(a) Diagram of Full communication centralized learning

全通信集中决策

基于隐藏层信息池化共享的集中决策架构[1]

基于AC （actor-critic）的多agent 双向协作网络（Bidirectionally-Coordinated Network, BiCNet）[2]

全局奖赏下的值分解网络[3]



(b) 全通信自主决策示意图

(b) Diagram of Full communication decentralised learning

全通信自主决策

自适应端到端的通信协议算法[4]

一般性协作actor-critic网络[5]

平均强化学习分析[6]

(c) 欠通信自主决策示意图

(c) Diagram of Limited communication independent learning

欠通信自主决策

全局奖赏，2个独立DQN网络[7]

采用分散滞后深度RNN 的Q 网络（Dec-HDRQNs）架构[8]

随访问次数的增加而增大接受负TD error 的概率[9]

[1] Sukhbaatar S, Fergus R. Learning multiagent communication with backpropagation[C]//Advances in Neural Information Processing Systems. 2016: 2244-2252.

[2] Peng P, Yuan Q, Wen Y, et al. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games[J]. arXiv preprint arXiv:1703.10069, 2017, 2.

[3] Sunehag P, Lever G, Gruslys A, et al. Value-decomposition networks for cooperative multi-agent learning[J]. arXiv preprint arXiv:1706.05296, 2017.

[4] Foerster J, Assael I A, de Freitas N, et al. Learning to communicate with deep multi-agent reinforcement learning[C]//Advances in Neural Information Processing Systems. 2016: 2137-2145.

[5] Mao H, Gong Z, Ni Y, et al. ACCNet: Actor-Coordinator-Critic Net for" Learning-to-Communicate" with Deep Multi-agent Reinforcement Learning[J]. arXiv preprint arXiv:1706.03235, 2017.

[6] Yang Y, Luo R, Li M, et al. Mean field multi-agent reinforcement learning[J]. arXiv preprint arXiv:1802.05438, 2018.

[7] Tampuu A, Matiisen T, Kodelja D, et al. Multiagent cooperation and competition with deep reinforcement learning[J]. PloS one, 2017, 12(4): e0172395.

[8] Omidshafiei S, Pazis J, Amato C, et al. Deep decentralized multi-task multi-agent reinforcement learning under partial observability[C]//Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017: 2681-2690.

[9] Palmer G, Tuyls K, Bloembergen D, et al. Lenient multi-agent deep reinforcement learning[C]//Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. International Foundation for Autonomous Agents and Multiagent Systems, 2018: 443-451.