IJCAI–18

Di Wu, Zhennan Wang, Wenbin Zou, Xia Li, Chen Xu
Shenzhen University[*]
dwu,...@szu.edu.cn

Abstract

The *IJCAI–18 Proceedings* will be printed from electronic manuscripts submitted by the authors. The electronic manuscript will also be included in the online version of the proceedings. This paper provides the style instructions.

1 Introduction

1.1 A* search algorithm

The following is an excerpt from Wikipedia `https://en.wikipedia.org/wiki/A*_search_algorithm`.

A* is an informed search algorithm, or a best-first search, meaning that it solves problems by searching among all possible paths to the solution (goal) for the one that incurs the smallest cost (least distance travelled, shortest time, etc.). It is an extension of Edsger Dijkstra's 1959 algorithm. A* selects the path that minimses:

$$f(n) = g(n) + h(n) \tag{1}$$

where n is the last node on the path, $g(n)$ is the cost of the path from the start node to n, and $h(n)$ is a heuristic that estimates the cost of the cheapest path from n to the goal. The heuristic is problem-specific. For the algorithm to find the actual shortest path, the heuristic function must be admissible, meaning that it never overestimates the actual cost to get to the nearest goal node. Typical implementation of A* use a priority queue to perform the repeated selection of minimum (estimated) cost nodes to expand.

2 dataset

2.1 real weather update

3 Experiments

3.1 Notations

3.2 Baselines

Straight lines

:

[*]Shenzhen Key Lab of Advanced Telecommunication and Information Processing, College of Information Engineering, Shenzhen University.

Notation	meaning
xid, yid	

Table 1: Results of 3D performance evaluation on mean coverage (higher is better) and center error (lower is better).

A* 2D

A* 3D: risky scheme

A* 3D: conservative scheme

4 Deterministic environment

5 Expected Sarsa

6 Dyna

[Sutton, 1990]

7 Prioritized Sweeping

[Peng and Williams, 1993]

8 Eligibility traces

Eligibility traces in conjunction with Temporal Difference (TD) errors provide an efficient, incremental way of shifting and choosing between Monte Carlo (MC) and TD methods. Methods using eligibility traces require more computation than one-step methods, but in return they offer significantly faster learning, particularly when rewards are delayed by many steps. However, in off-line applications in which data can be generated cheaply, perhaps from an inexpensive simulation, then it often does not pay to use eligibility traces. In these cases the objective is not to get more out of a limited amount of data, but simply to process as much data as possible as quickly as possible. In these cases the speedup per datum due to traces is typically not worth their computational cost. Moreover, the tabular case is in some sense the worst cast for the computational complexity of eligibility traces [Sutton and Barto, 2018]. In our problem setting, which is an off-line tabular setting, we leave out the method of eligibility traces.

References

[Peng and Williams, 1993] Jing Peng and Ronald J Williams. Efficient learning and planning within the dyna framework. *Adaptive Behavior*, 1993.

Configuration	train (prev 12-05)		test (prev 12-05)
	mins	crash	min
Straight Line	51318	31	69376
A* 2D	51318	31	65656
A* 3D(risky)	**43010**	**21**	65720
A* 3D(conservative)	48874	29	**62766** [1]

Table 2: Results of 3D performance evaluation on mean coverage (higher is better) and center error (lower is better).

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 2018.

[Sutton, 1990] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the seventh international conference on machine learning*, 1990.

Algorithm 1: 3D A* Double(I haven't implemented double yet) Expected Sarsa (Q-learning(which is better Q or ES?)), Dyna model planning with prioritized sweeping

```
Data:
  $L^s = \{x^s, y^s\}$: 2D location of the starting city
  $L^g = \{x^g, y^g\}$: 2D location of the goal city
  $W_m = \{w_{x,y,t}\}$: Wind predictions of model $m \in \{1, 2, \ldots, M\}$ ($M$ is the number of models) with size as $X \times Y \times T$
        where $X, Y$ are the grid world size and $T$ is the maximum allowed time
1  Generate $M$ trajectories: $J_m$ from 3D A* algorithm (conservative(?)) and their corresponding policies: $\pi_m$
2  Initialise action-value function $Q : X \times Y \times T \leftarrow 0$, $Model(s, a)$ and $PQueue$ to empty
3  Initialise starting states $S_S$ from $L^s$ as $(x^s, y^s, 0)$; goals states $S_G$ from $L_g$ as $(x^s, y^s, t), t \in (1, 2, \ldots T)$ and terminal
   states $S_T$ as $(x, y, T), x \in (1, 2, \ldots X), y \in (1, 2, \ldots Y)$
4  $L$ is the total A* looping number(How to choose the number?), $N$ is the number of planning steps, $\theta$ is the priority
   threshold, $\gamma$ is the discount rate, $\alpha$ is the learning rate
5  for $l \leftarrow 1$ to $L$ do
6      $s_1 \leftarrow S_S$
7      With probability $\frac{1}{m}$ , uniformly select a random wind model number $m_w$ from $\{1, 2, \ldots, M\}$
8      With probability $\frac{1}{m}$ , uniformly select a random policy number $m_{A*}$ from $\{1, 2, \ldots, M\}$ (Should $m_w, m_{A*}$ be the
       same?)
9      for $t \leftarrow 1$ to $T$ do
10         $a_t \leftarrow \pi_{m_{A*}}$ (with probability $\epsilon$ randomly select an action, if $s_t$ is not in $\pi_{m_{A*}}$, then action is selected greedily
           according to $Q$) (How to handle expected sarsa action selectoin?)
11         Execute action $a_t$; observe reward $r_t$ from wind model $W_{m_w}$, and state $s_{t+1}$
12         $Model(s_t, a_t) \leftarrow r_t, s_{t+1}$
13         $P \leftarrow |r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)|$
14         $P \leftarrow |r_t + \gamma \sum_a \pi_\epsilon(a|s_{t+1}) Q(s_{t+1}, a) - Q(s_t, a_t)|$
15         if $P > \theta$, then insert $s_t, a_t$ into $PQueue$ with priority $P$
16         for $n \leftarrow 1$ to $N$ while $PQueue$ is not empty do
17             $s_n, a_n \leftarrow first(PQueue)$
18             $r_n, s_{n+1} \leftarrow Model(s_n, a_n)$
19             $Q(s_n, a_n) \leftarrow Q(s_n, a_n) + \alpha[r_n + \gamma \max_a Q(s_{n+1}, a) - Q(s_n, a_n)]$
20             $Q(s_n, a_n) \leftarrow Q(s_n, a_n) + \alpha[r_n + \gamma \sum_a \pi_\epsilon(a|s_{n+1}) Q(s_{n+1}, a) - Q(s_n, a_n)]$ (here, $\pi_\epsilon$ is $\epsilon$ soft policy)
21             for all $\bar{s}, \bar{a}$ predicted lead to $s_n$ do
22                 $\bar{r} \leftarrow$ predicted reward for $\bar{s}, \bar{a}, s_n$
23                 $P \leftarrow |\bar{r} + \gamma \max_a Q(s_n, a) - Q(\bar{s}, \bar{a})|$
24                 $P \leftarrow |\bar{r} + \gamma \sum_a \pi_\epsilon(a|s_n) Q(s_n, a) - Q(\bar{s}, \bar{a})|$
25                 if $P > \theta$, then insert $\bar{s}, \bar{a}$ into $PQueue$ with priority $P$
26         if $s_{t+1}$ is in terminal states $S_T$ or goal states $S_G$, break
27
Result:
  Trajectory $J$ from the updated $Q$ using greedy action selection
```