面向MOBA的场景设计及验证

11612908 赵宇
指导老师：刘佳琳

为什么研究MOBA(Multiplayer Online Battle Arena) AI ？

为什么研究MOBA(Multiplayer Online Battle Arena) AI ?

1. MOBA游戏的高市场占有率

#	图片	游戏	当前在线	今日峰值	最近30日
1		反恐精英：全球攻势	533,471	961,479	
2		刀塔2	363,344	750,308	
3		泰拉瑞亚	150,535	181,767	
4		GTA5	99,660	192,141	
5		绝地求生	74,612	400,836	
6		军团要塞2	73,975	78,532	
7		彩虹六号：围攻	56,376	99,895	
8		文明6	51,916	66,214	
9		足球经理2020	50,657	81,436	
10		腐蚀	50,161	66,064	

图片来源：https://steamstats.cn/

为什么研究MOBA(Multiplayer Online Battle Arena) AI ?

1. MOBA游戏的高市场占有率
2. MOBA游戏丰富的策略性、合作性

#	图片	游戏	当前在线	今日峰值	最近30日
1		反恐精英：全球攻势	533,471	961,479	
2		刀塔2	363,344	750,308	
3		泰拉瑞亚	150,535	181,767	
4		GTA5	99,660	192,141	
5		绝地求生	74,612	400,836	
6		军团要塞2	73,975	78,532	
7		彩虹六号：围攻	56,376	99,895	
8		文明6	51,916	66,214	
9		足球经理2020	50,657	81,436	
10		腐蚀	50,161	66,064	

图片来源：https://steamstats.cn/

主要贡献

- MOBA环境的开发与完善
- 基于完善后的MOBA环境作的场景设计及MOBA智体的训练

MOBA环境的开发与完善

1. 环境介绍
2. 工作介绍
3. 算法设计及测试

MOBA-Env游戏环境

1. 运行平台：OpenAI Gym
2. 游戏逻辑：Go语言开发

MOBA-Env游戏环境

1. 运行平台：OpenAI Gym
2. 游戏逻辑：Go语言开发

优点：
1. 配置场景可以只配置需要的元素(英雄、防御塔)。
2. 运行平台(OpenAI Gym)提供的接口是标准化的，方便使用。

MOBA-Env游戏环境

1. 运行平台：OpenAI Gym
2. 游戏逻辑：Go语言开发

优点：
1. 配置场景可以只配置需要的元素(英雄、防御塔)。
2. 运行平台(OpenAI Gym)提供的接口是标准化的，方便使用。

不足：
缺少很多游戏元素，如地形、战争迷雾等

基于MOBA-Env的游戏开发(1)

主要工作：在MOBA环境中定义了地形，并设计了所有单位移动、攻击的算法

S, T代表起点、终点，V代表点集合，E代表边集合，AR代表单位的攻击距离，算法伪代码如下：

```
Algorithm 2 Original Shortest Path(S, T, G(V, E), AR)
EndVertex := all points v with distance(v, T) = AR
currentShortestLength := INF
currentRoad := NULL
for every endpoint in EndVertex:
    road := AStar(S, endpoint, G)
    if length of road < currShortestLength:
        currShortestLength = length of road
        currRoad = road
return road
```

理论复杂度：
算法1：O(AR|E|)常数2π

基于MOBA-Env的游戏开发(1)

主要工作：在MOBA环境中定义了地形，并设计了所有单位移动、攻击的算法

S, T代表起点、终点，V代表点集合，E代表边集合，AR代表单位的攻击距离，算法伪代码如下：

Algorithm 3 MOBA Shortest Path(S, T, G(V, E), AR)

E_{origin} := G(E)

remove edges from G whose start point is T

add edges between T to all points v whose distance (v, T) = AR, set edge length = 0

road := Astar(T, S, G)

remove T from road

G(E) = E_{origin}

return road

理论复杂度：
算法1：O(AR|E|)常数2π
算法2：O(|E|)

基于MOBA-Env的游戏开发(2)

- 算法测试平台：MOBA-Env
- 测试方法：随机生成10个点，从起始点开始依次到达下一个点，计算走到最后一个点使用的时间，重复五十次累计。除使用的算法外，两组实验其他设置完全相同

理论复杂度：
算法1：O(AR|E|)常数2π
算法2：O(|E|)

攻击距离	10	20	30	40	50
MOBA shortest Path(ms)	89.82	175.60	269.92	353.61	431.50
Original Shortest Path(ms)	5641.59	21950.02	50531.73	88154.96	134412.20
比值 (Original/MOBA)	62.81	125.00	187.21	249.30	311.51
理论比值 (Original/MOBA)	62.83	125.67	188.50	251.33	314.16

多场景MOBA AI的训练

为什么要多场景的训练MOBA AI

为什么要多场景的训练MOBA AI

参与英雄　　防御塔　　视野　　地形　　位置




为什么要多场景的训练MOBA AI

参与英雄　　防御塔　　视野　　地形　　位置

多样的游戏元素 -> 复杂的游戏场景

多样的场景->不同的获胜策略

测试环境

Baseline：OpenAI PPO 算法的baseline[https://github.com/openai/baselines/tree/master/baselines/ppo2]
游戏环境: MOBA-Env
训练平台：Ubuntu16.04(CPU core8)
深度学习框架：tensorflow1.14
Gym版本:v0.1.2

场景设计+训练（1）：寻路场景

实际场景：MOBA游戏中单位A欲快速赶往战场
抽象场景：寻路场景（只和完成任务的时间有关，越快越好）
目的：验证 PPO 算法是否可以训练智体尽快寻路并到达的能力

参数

	血量	攻击力	攻击范围	移动速度
敌方英雄 1	1000	1000	40	0
我方英雄 1	2000	1000	80	40

奖励函数

```
Reward function 1:

function get_reward(curr_state, last_state)
    distance_last := distance between self_hero0 and oppo_hero0 at
last_state
    distance_curr := distance between self_hero0 and oppo_hero0 at
curr_state
    positive_reward := (distance_curr - distance_last) * 0.001
    harm_reward := (time from start_time) * 0.005
return positive_reward - harm_reward
```

训练结果：

timestep	Reward(平均)-Reward(理论)
500	-0.4078
1000	-0.2220
1500	-0.0097
2000	-0.0005

场景设计+训练（2）：放风筝场景

场景：移动速度高的一方(或者攻击距离远的一方)，通过交替进行移动和攻击动作，以消耗最少的血量为目的来击杀移动速度慢的一方
目的：通过简单的场景，验证PPO算法是否会陷入局部最优解（策略更新的步长是否会在合理范围内）

场景设计

	血量	攻击力	攻击范围	移动速度
敌方英雄 1	3000	100	40	40
我方英雄 1	1000	100	80	60

奖励函数

```
Reward function 2:
function get_reward(state)
    positive_reward := (oppo_hero0 max health - oppo_hero0 current
health) *0.01
    harm_reward := (self_hero0 max health - self_hero0 current
health)*0.03
return positive_reward - harm_reward
```

训练结果

timestep	Reward(平均)-Reward(理论)
500	-4.5625
1000	-2.9478
1500	-1.7966
2000	-1.0979
2500	-0.1934
3000	-0.0310

场景设计+训练（3）：轮流抗血（易）

场景：当面对 BOSS（比如英雄联盟的大龙）时，英雄间需要合理的团队策略才能取胜，在简单场景下，我们只要求两个英雄间有一个在承受伤害即可。该场景有两个我方英雄，目的是验证 PPO 算法能否能训练出用于多智能体的策略。

目的：验证PPO算法是否可以用来训练多智能体的MOBA AI

场景设计

	血量	攻击力	攻击距离	移动速度
敌方英雄 1	4500	100	40	50
我方英雄 1	3000	10	40	50
我方英雄 2	300	150	40	50

timestep	Reward(平均)-Reward(理论)
500	-3.2039
1000	-1.8351
2000	-1.0377
3000	-0.5007
4000	-0.3048
5000	-0.0963

奖励函数：

```
Reward function 3:
function get_reward(state)
    harm_reward := 0
    if self_hero0 health is 0:
        harm_reward = harm_reward + 0.5
    end if
    if self_hero1 health is 0:
        harm_reward = harm_reward + 1
    end if
    harm_reward = harm_reward + (self_hero0 max health - self_hero0
    current health)*0.001
    harm_reward = harm_reward + (self_hero1 max health - self_hero1
    current health)*0.005
    positive_reward := oppo_hero0 max health - oppo_hero0 current
    health) *0.001
return positive_reward - harm_reward
```

场景设计+训练（4）：轮流抗血（难）

场景：在实际游戏中，为了避免使对方击杀我方英雄获得金钱和经验，输出可以适当的承受一部分伤害，也就是该场景下，我方两个英雄需要都存活。
目的：学会复杂的轮流抗血策略。

场景设计：

	血量	攻击力	攻击距离	移动速度
敌方英雄 1	4000	200	40	50
我方英雄 1	4000	50	40	50
我方英雄 2	1000	150	40	50

奖励函数

```
Reward function 4:
function get_reward(state)
    harm_reward := 0
    if self_hero0 health is 0:
        harm_reward = harm_reward + 1
    end if
    if self_hero1 health is 0:
        harm_reward = harm_reward + 1
    end if
    harm_reward = harm_reward + (self_hero0 max health - self_hero0
    current health)*0.001

        harm_reward = harm_reward + (self_hero1 max health - self_hero1
        current health)*0.005
        positive_reward := oppo_hero0 max health - oppo_hero0 current
        health) *0.001
    return positive_reward - harm_reward
```

训练结果：

Timestep	Reward(平均)-Reward(理论)
1000	-2.878
2000	-2.203
3000	-1.635
4000	-1.353
5000	-1.128
6000	-0.909
7000	-0.553
8000	-0.552

实验结论

- 设计并实现了基于MOBA的最短路算法;
- 设计了寻路、放风筝、轮流抗血（易）、轮流抗血（难）四个场景;
- 寻路、放风筝、轮流抗血（易）场景达到了场景的设计目的;
- 轮流抗血（难）未达到场景的设计目的（可以获胜但不是最优解），是因为最优策略非常难以随机得到。

Future work:

游戏环境完善：加入战争迷雾等游戏因素
训练场景：引入战争迷雾，设计非完全信息场景并训练
难点：信息是非完全的，需要对训练算法进行一定改动

引用

[1] G.Brockman and V.Cheung,“Openai five defeats dota 2 world champions,”https://openai.com/blog/openai-five-defeats-dota-2-world-champions/, April 15, 2019.
[2] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” 2016.
[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms. arxiv 2017,” arXiv preprint arXiv:1707.06347.
[4] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, “Emergence of locomotion behaviours in rich environments,” arXiv preprint arXiv:1707.02286, 2017.
[5] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in International conference on machine learning, 2015, pp. 1889–1897.
[6] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

Thank you!