Hybrid Reward Architecture for Reinforcement Learning

Harm van Seijen[1]
harm.vanseijen@microsoft.com

Mehdi Fatemi[1]
mehdi.fatemi@microsoft.com

Joshua Romoff[12]
joshua.romoff@mail.mcgill.ca

Romain Laroche[1]
romain.laroche@microsoft.com

Tavian Barnes[1]
tavian.barnes@microsoft.com

Jeffrey Tsang[1]
tsang.jeffrey@microsoft.com

[1]Microsoft Maluuba, Montreal, Canada

【强化学习算法 31】HRA

张楚珩
清华大学 交叉信息院博士在读

HRA全称是Hybrid Reward Architecture

原文传送门

Van Seijen, Harm, et al. "Hybrid reward architecture for reinforcement learning." Advances in Neural Information Processing Systems. 2017.

特色

提出了把真实的reward进行分解，使用不同的action value function去学习不同部分的奖励，这样避免了总的奖励形态太复杂而学习不到的情况，把复杂问题转化为多个简单问题。尽管解决的方式不普适并且做了很多针对性的工作，但从结果上来看完美解决了吃豆人（Ms. Pac-Man）的游戏。

过程

1. 奖励的分解和Q函数的分解

前面已经说到了其解决问题的主要思路就是把复杂的奖励形态拆分成比较多简单的部分，使得每一个部分学习起来都比较简单。考虑如下奖励的拆分

$$R_{env}(s,a,s') = \sum_{k=1}^{n} R_k(s,a,s'), \qquad \text{for all } s,a,s', \tag{4}$$

考虑每个子奖励都对应自己的 Q_k 函数，考虑离散的动作空间和一个均匀随机的策略 π，有

$$
\begin{aligned}
Q_{env}^{\upsilon}(s,a) &= \mathbb{E}\left[\sum_{i=0}^{\infty}\gamma^i R_{env}(s_{t+i},a_{t+i},s_{t+1+i})|s_t=s,a_t=a,\upsilon\right], \\
&= \mathbb{E}\left[\sum_{i=0}^{\infty}\gamma^i \sum_{k=1}^{n} R_k(s_{t+i},a_{t+i},s_{t+1+i})|s_t=s,a_t=a,\upsilon\right], \\
&= \sum_{k=1}^{n}\mathbb{E}\left[\sum_{i=0}^{\infty}\gamma^i R_k(s_{t+i},a_{t+i},s_{t+1+i})|s_t=s,a_t=a,\upsilon\right], \\
&= \sum_{k=1}^{n} Q_k^{\upsilon}(s,a) := Q_{\text{HRA}}^{\upsilon}(s,a)\,.
\end{aligned}
$$

但同时注意到，如果定义

$$
Q_{\text{HRA}}^{*}(s,a) := \sum_{k=1}^{n} Q_k^{*}(s,a) \qquad \text{for all } s,a\,.
$$

它和真实的最优Q函数 Q_{env}^{*} 是不一样的。即， $Q_{HRA}^{*} \neq Q_{env}^{*}$ ， $Q_{HRA}^{\pi} = Q_{env}^{\pi}$ 。其原因是对于 Q_k^{*} 来说，不同的 k 对应不同的最优策略，直接加和起来是没有意义的；只有它们都共用一套策略，比如一个随机的策略 π 的时候才能够直接这样加起来。

2. 学习算法

在奖励没有拆分的时候，一个Q-learning的算法为

$$
\begin{aligned}
&\mathcal{L}_i(\theta_i) = \mathbb{E}_{s,a,r,s'}[(y_i^{DQN} - Q(s,a;\theta_i))^2]\,, \\
\text{with} \quad &y_i^{DQN} = r + \gamma \max_{a'} Q(s',a';\theta_{i-1}),
\end{aligned}
$$

文章提出奖励拆分之后可以使用这样的算法

$$
\mathcal{L}_i(\theta_i) = \mathbb{E}_{s,a,r,s'}\left[\sum_{k=1}^{n}(y_{k,i} - Q_k(s,a;\theta_i))^2\right],
$$

其中每次更新的目标可以设置为（类似Q-learning）

$$
y_{k,i} = R_k(s,a,s') + \gamma \max_{a'} Q_k(s',a';\theta_{i-1})\,.
$$

或者（类似expected SARSA）

$$y_{k,i} = R_k(s,a,s') + \gamma \sum_{a' \in \mathcal{A}} \frac{1}{|\mathcal{A}|} Q_k(s',a';\theta_{i-1}) .$$

每次策略都更新为综合各个 Q_k 产生的近乎贪心的策略。

实验显示，后一种效果比较好。个人理解，这其实不难想象，虽然把奖励和Q函数拆分了，但是最后一个agent只能采取一种action，不可能每个部分都按照自己的取向去选择最大化自己 Q_k 的行动，因此后一种方法里面使用一个平均的数值更接近行动耦合的实际情况。

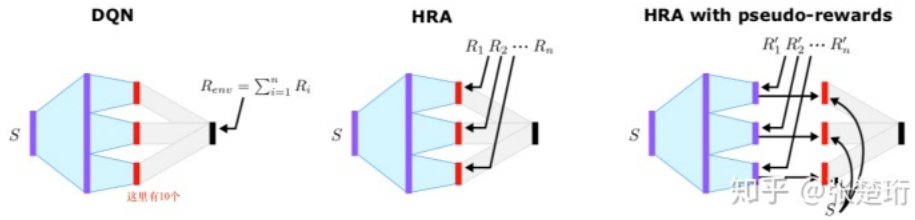
3. 进一步优化

除了把奖励拆分之外，文章还提出了进一步的优化方法

- 对于每一个 Q_k 来说，把与它无关的输入特征去掉；
- 对于每一个 Q_k 来说，如果这一头不能在产生任何奖励了，就直接强制性把它设置为0；
- 对于每一个 Q_k 来说，除了拆分出来的奖励 R_k ，还可以在此基础上加上另外的pseudo-reward；

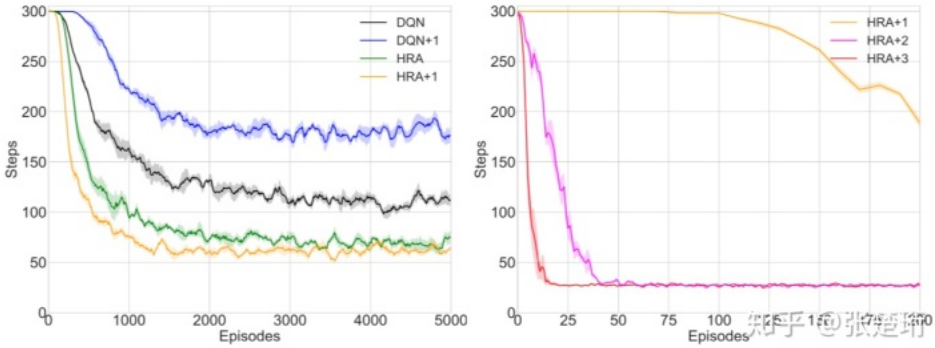
4. 实验一：收集格子地图上的水果

实验是说在 10×10 的格子上面，有固定的10个位置可能水果，每局一共出现5个水果；智能体可以上下左右走，目标是最短的步数收集所有的水果。文章把每个可能出现水果的地方都设置子奖励 R_k （如果没有水果就一直是0）。然后做了和DQN以及加上其他各种进一步优化方法的对比实验。



一个很有欺骗性的图：除了和DQN的对比实验使用了神经网络，后面的实验都是tabular case

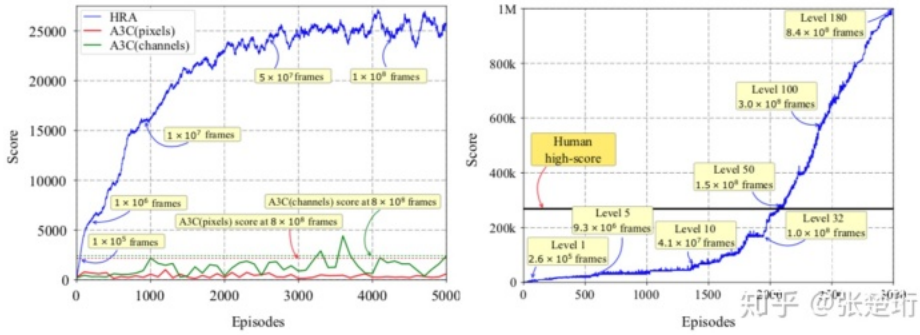
- 和DQN使用相同的结构，但是HRA把分出来的10端（图中红色，每端输出四个数值，代表不同行动的Q函数）分别以 $y_{k,i}$ 为目标进行训练，而DQN直接把合成的输出（图中黑色，输出四个数值，代表不同行动的Q函数）以 y_i 为目标训。结构显示HRA效果更好；
- 使用改进1，把与某一端无关的特征不输入到该端（使用改进1：HRA+1）；
- 使用改进2，如果判断某一端不会再产生任何奖励（即，对应的水果已经获得了）就不再更新该段Q函数（使用改进1&2：HRA+2）；
- 使用改进3，使用general value function（GVF）产生的pseudo-reward，GVF定义为随机游走从任意位置到那10个水果位置的并且获得相应水果奖励的Q函数（使用改进1&2&3：HRA+3)；



实验一的结果：左边使用了神经网络，右边的红线、粉线应该是tabular case

5. 实验二：吃豆人

先说结果是很好了，基本上把游戏打爆了。



不过实在是用了太多的工程技术

- **预处理**：本来是输入像素的任务，他们直接把状态解析出来了，形成了11个 40×40 的二值图加上一个4维的one-hot向量：分别代表4个ghost、4个可打爆的ghost、吃豆人的位置、水果的位置、豆的位置；4维向量代表吃豆人的四个朝向。（这一步下来，局面理解完全解决）；
- **Pseudo Reward**：整个游戏一共只有四个地图，先通过游走，学习到4种地图下，吃豆人的950中状态下采取4种行动，走到400中不同位置的价值函数；然后把相应的奖励作为pseudo reward；

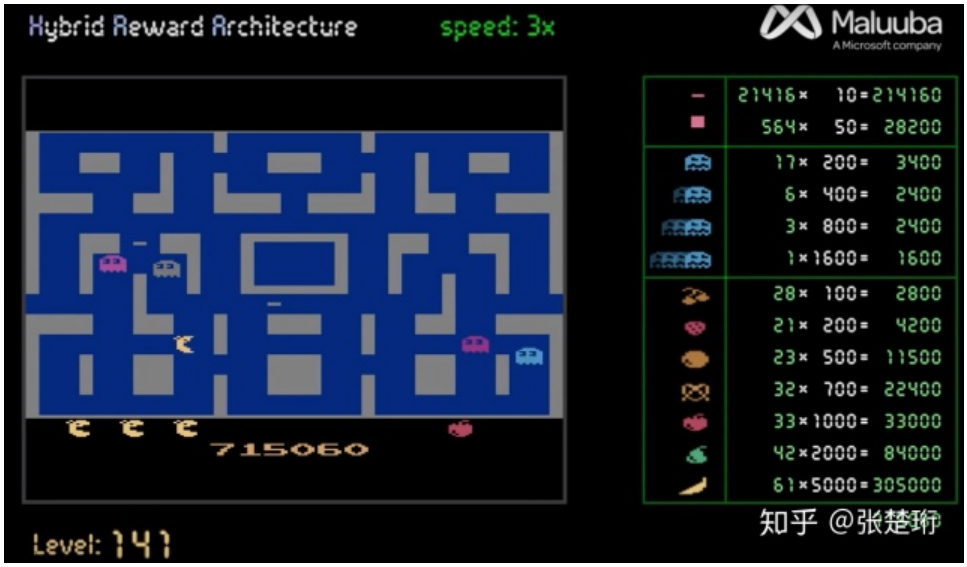
- **增大探索**：早起的时候通过设置随机的Q值来鼓励探索（diversification）；后期基于状态方位计数来使用类似UCB的公式增大探索（count-based）；
- **综合 q_k 的方法**：等权加和的方法最为自然，但是agent就会过于避免碰到ghost；因此他们对于ghost和其他的东西使用了不同的加权；
- **记忆**：这是最bug的一点，为了通关，它们把之前没被杀死成功通关的轨迹记下来，下次尽量多选择之前的成功经验；

这篇工作做得还挺“脏”的，为啥还要分享出来呢？

因为有一点个人觉得很有趣：奖励分开之后，各个不同的Q函数还通过共同的动作相连接，在这一点上没能解耦合，如果能真正做到子任务间优雅地解耦合，就能完全独立地把复杂任务分解出来单个去解决了。可惜的是，这篇文章没有仔细研究这一点，完全是糊弄过去了。

吃豆人游戏长啥样？

视频连接：youtube.com/watch?...



右边得分清单里面分别是小豆豆、大豆豆（可以把小鬼变成蓝色的）、一次性干掉了几只小鬼和各种不同的水果（从被打爆的小鬼身上掉下来的）

编辑于 2018-11-27

强化学习 (Reinforcement Learning)

文章被以下专栏收录

强化学习前沿
读呀读paper

进入专栏