Hierarchical Deep Reinforcement Learning:Integrating Temporal Abstraction andIntrinsic Motivation

论文链接：Hierarchical Deep Reinforcement Learning:Integrating Temporal Abstraction and Intrinsic Motivation

一、 关键词

- sparse feedback， **稀疏反馈问题是指只有达成特定目标或者触碰死亡陷阱时才会给予一定的奖励或惩罚，而进行其他操作时不会有任何反馈。**
- delayed reward， **延迟回报是指action在当前状态之后的回报，与之对应的是即时回报(immediate reward)。一般情况下，只有当action不会影响到接下来的状态转移时，才会有immediate reward，否则都需要考虑delayed reward。**
- intrinsic motivation, Intrinsic motivation refers to behavior that is driven by internal rewards. In other words, the motivation to engage in a behavior arises from within the individual because it is naturally satisfying to you. **内在驱动从字面意义上来讲就是做自己喜欢做的事情，在强化学习领域就是agent按照“自己喜欢”的行为去行动。**
- target driven， **目标驱动，也可以理解为外在驱动，就是为了达成目标而去做的事情，无论自己是否喜欢。在强化学习领域就是agent不断地选择逼近目标的行为去行动。与intrinsic motivvation是对立统一的。**
- options，action abstraction, 都是SMDP中的概念,参考论文Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning。**这个framework包含对动作空间的抽象，在每一个step，agent可以选择一个primitive action或者一个option。每个option定义了actions(可以是primitive actions，也可以是other options)的策略。**
- temporal abstraction， temporal abstraction is a common task, based on temporal reasoning, which provides an intelligent interpretation and summary of large amounts of raw data。参考论文Temporal abstraction is a common task, based on temporal reasoning, which provides an intelligent interpretation and summary of large amounts of raw data。具体一点，可参考另外一篇论文的摘要Temporal abstraction in reinforcement learning:Decision making usually involves choosing among different courses of action over a broad range of time scales. For instance, a person planning a trip to a distant location makes high-level decisions regarding what means of transportation to use, but also chooses low-level actions, such as the movements for getting into a car。**换句话来说，就是agent能进行对时间进行分割及自动推理，对于large-scale的时间内设定子目标，对于small-scale的时间内设定具体的action。就好像要去旅游，首先是设定坐什么交通工具，然后是怎么坐这个交通工具。**
- object-based representations， The state space S is defined by a variable number of independent objects. These objects are organized into classes of objects that behave alike. **将状态空间分割成多个独立的objects去看待，然后把object当作是某种class的实例。** The agent is seen as an object of a specific class, constrained to be instantiated exactly once in each state. Other classes can have none to many instances in any particular state. 参考论文Object Focused Q-learning for Autonomous Agents 。**在每一个状态下，智能体都视为特定class下的一个object，也就是只能取到该class下的对应的状态值。**
- successor representation， The Successor Representation (SR) of state i predicts the expected future occupancy of all other states if you are currently in state i. **在当前状态i下，未来状态j出现的次数占未来访问过的所有状态的比例。**

$$[\bar{\mathbf{x}}_{\mathbf{i}}]_j = [I]_{ij} + \gamma[Q]_{ij} + \gamma^2[Q]_{ij} + \cdots = [(I-\gamma Q)^{-1}]_{ij}$$

二、 简述

1. 需解决的问题

- 强化学习的稀疏回报问题，导致agent不能进行足够的探索，学习到鲁棒性比较好的策略。
- 强化学习的延迟回报问题。

2. 采用的方法及理论

- 提出层级 DQN 框架，组合层级动作值函数，实现在不同的时间尺度上进行目标驱动和内在驱动的强化学习。
  - top-level module: 学习一个子目标
  - bottom-level module: 学习具体的策略去达成上层的子目标

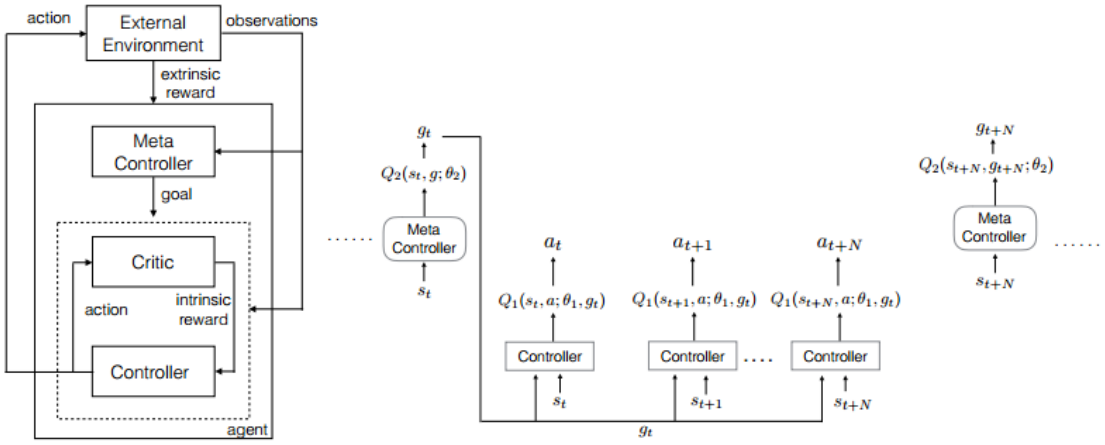
- 提出一种temporal abstraction的方案，同时对options和action policy进行学习，不会给每个option设定一个Q值函数，而是把option当作输入的一部分。这样做有两个优点：
  - there is shared learning between different options，进行所有的options作为输入，进去同一个网络
  - the model is scalable to large number of options，作为输入，无论有多少个options，无非就是增加一个维度的数据而已
- 采用 competence based 内在激励，参考论文What is intrinsic motivation? a typology of computational approaches。
- 采用的Object-based representation不用挖掘object之间的关联。
- 采用DQN的架构
- 借助successor representation (SR)的思想去分解得到子目标。

3. 优化方法

随机梯度下降

三、 建模过程

1. 定义extrinsic function为 $\mathcal{F}:(s)\rightarrow\mathbb{R}$，表示外部回报函数。
2. 为了更好地进行探索，定义了内在目标 $g\in\mathcal{G}$。agent通过策略 π_g来不断设定并获得内在目标序列，使之最大化extrinsic function。agent通过策略 π_a来不断设定并获得内在目标序列，使之最大化intrinsic function。agent利用它的critic来学习策略 π_a，依据的是是否能够到达子目标 g。整个模型的运行流程如下图所示：



   - 整个框架分成两层，meta-controller和controller，对应的是外层控制器和内层控制器，都用独立的DQN来表示。
   - 首先meta-controller接受环境的状态输入，然后通过最大化expected future extrinsic reward，也就是重新估计 $Q_2(s_t,g_t;\theta_2)$ 并输出内在目标策略 π_g ，然后采样获得 g_t 。紧接着controller接受状态输入和当前的内在目标 g_t ，然后通过最大化expected future intrinsic reward，也就是重新估计 $Q_1(s_t,a_t;\theta_1,g_t)$并输出动作策略 π_a 。controller结束条件有两个，分别是episode结束，以及内在目标 g_t 实现。只有当内在目标实现时，critic才会给controller一个正的reward。
   - controller结束以后，meta-controller将会选择一个新的 g ，重复以上操作。
3. controller和meta-controller的更新过程蕴含了temporal abstraction。
   - 当controller的terminal state到来时，内部的critic负责反馈给controller出intrinsic reward，称为 $r_t(g)$。controller的目标函数就是最大化intrinsic reward的累积和$R_t(g)$：

$$R_t(g)=\sum_{t'=t}^{\infty}\gamma^{t'-t}r_{t'}(g)$$

   - 当controller获得action并执行时，环境会返回外部回报 f_t。 类似的，meta-controller的目标函数就是：

$$F_t=\sum_{t'=t}^{\infty}\gamma^{t'-t}f_{t'}$$

   - 需要注意 R_t 和 F_t 的time scales是不一样的，F_t 的time scale是内在目标序列的间隔，明显比 R_t 的time scale大，到来得更慢。
4. 下面列出动作值函数的目标函数：
   对于controller，公式如下。g 表示agent状态为 s_t 时的内在目标。π_{ag} 就是动作策略（上文写为 π_a ）。

$$Q_1^*(s,a;g) = \max_{\pi_{ag}} E\left[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'} \mid s_t = s, a_t = a, g_t = g, \pi_{ag}\right]$$
$$= \max_{\pi_{ag}} E\left[r_t + \gamma max_{a_{t+1}} Q_1^*(s_{t+1}, a_{t+1}; g) \mid s_t = s, a_t = a, g_t = g, \pi_{ag}\right]$$

对于meta-controller，公式如下。 N 表示controller停止时经过的time steps。 g' 是agent状态为 s_{t+N} 时的内在目标，也就是内在目标序列中的下一个。 π_g 是内在目标策略。

$$Q_2^*(s,g) = \max_{\pi_g} E\left[\sum_{t'=t}^{t+N} f_{t'} + \gamma max_{g'} Q_2^*(s_{t+N}, g') \mid s_t = s, g_t = g, \pi_g\right]$$

5. 用神经网络来近似表示 $Q^*(s,g)$, $Q^*(s,g) \approx Q(s,g,\theta)$, $Q \in \{Q_1, Q_2\}$。

6. 把 Q_1, Q_2 的 experiences (s_t, g_t, f_t, s_{t+N})和 $(s_t, a_t, g_t, r_t, s_{t+1})$分别放入 disjoint memory spaces $\mathcal{D}_1$ 和 $\mathcal{D}_2$ 中。

7. Q_1 的loss function可以定义为 $L_1(\theta_1)$ ，公式如下所示，其中 i 表示训练的迭代次数：

$$L_1(\theta_{1,i}) = E_{(s,a,g,r,s')\sim D_1}\left[\left(y_{1,i} - Q_1(s,a;\theta_{1,i},g)\right)^2\right]$$

$$y_{1,i} = r + \gamma max_{a'} Q_1(s',a';\theta_{1,i-1},g)$$

对 $L_1(\theta_{1,i})$ 进行求导，得到

$$-\bigtriangledown_{\theta_{1,i}} L_1(\theta_{1,i}) = E_{(s,a,g,r,s')\sim D_1}\left[\left(r + \gamma max_{a'} Q_1(s',a';\theta_{1,i-1},g) - Q_1(s,a;\theta_{1,i},g)\right) \bigtriangledown_{\theta_{1,i}} Q_1(s,a;\theta_{1,i},g)\right]$$

同理可以得到 $L_2(\theta_{2,i})$ 和 $-\bigtriangledown_{\theta_{2,i}} L_2(\theta_{2,i})$。

8. 算法伪代码如下所示：

Algorithm 1 Learning algorithm for h-DQN

Initialize experience replay memories $\{\mathcal{D}_1, \mathcal{D}_2\}$ and parameters $\{\theta_1, \theta_2\}$ for the controller and meta-controller respectively.
Initialize exploration probability $\epsilon_{1,g} = 1$ for the controller for all goals g and $\epsilon_2 = 1$ for the meta-controller.
for $i = 1, num\_episodes$ **do**
    Initialize game and get start state description s
    $g \leftarrow \text{EPSGREEDY}(s, \mathcal{G}, \epsilon_2, Q_2)$
    while s is **not** terminal **do**
        $F \leftarrow 0$
        $s_0 \leftarrow s$
        while not (s is terminal **or** goal g reached) **do**
            $a \leftarrow \text{EPSGREEDY}(\{s, g\}, \mathcal{A}, \epsilon_{1,g}, Q_1)$
            Execute a and obtain next state s' and extrinsic reward f from environment
            Obtain intrinsic reward $r(s, a, s')$ from internal critic
            Store transition $(\{s, g\}, a, r, \{s', g\})$ in $\mathcal{D}_1$
            $\text{UPDATEPARAMS}(\mathcal{L}_1(\theta_{1,i}), \mathcal{D}_1)$
            $\text{UPDATEPARAMS}(\mathcal{L}_2(\theta_{2,i}), \mathcal{D}_2)$
            $F \leftarrow F + f$
            $s \leftarrow s'$
        end while
        Store transition (s_0, g, F, s') in $\mathcal{D}_2$
        if s is **not** terminal **then**
            $g \leftarrow \text{EPSGREEDY}(s, \mathcal{G}, \epsilon_2, Q_2)$
        end if
    end while
    Anneal ϵ_2 and ϵ_1.
end for

Algorithm 2 : $\text{EPSGREEDY}(x, \mathcal{B}, \epsilon, Q)$

if random() $< \epsilon$ **then**
    return random element from set $\mathcal{B}$
else
    return $\text{argmax}_{m \in \mathcal{B}} Q(x, m)$
end if

Algorithm 3 : $\text{UPDATEPARAMS}(\mathcal{L}, \mathcal{D})$

Randomly sample mini-batches from $\mathcal{D}$
Perform gradient descent on loss $\mathcal{L}(\theta)$ (cf. (3))

上面代码中选取策略采用的是 $\epsilon - greedy$ 算法，对 ϵ的设置为：ϵ 和 ϵ 初始值都设置为1，表示采用完全随机的策略，最终都会退火变为0.1，如果 g 的到达率大于90%，则ϵ_1强制设置为0.1。

四、 总结

1. 工作概括

本文提出一种层次强化学习的方法，通过外层meta-controller对整个目标进行分解，得到子目标序列 $\mathcal{G}$。同时内层的controller不断地指引agent逼近子目标。

这种方法适用于sparse reward的环境，通常这种环境只有一个最终目标，只有当Agent达到最终目标，才会有reward。稀疏反馈会导致agent不能很好地进行探索。为了解决sparse reward的问题，引入了intrinsic reward和extrinsic reward的概念，从而出现了层次强化学习。用人来当例子，extrinsic reward就是好好学习，期末考一百分，instrinsic reward就是好好学习，享受每一次新知识带来的喜悦。

2. 不足之处

个人感觉最关键的是子目标怎么量化，以及内层的critic怎么依据子目标和当前状态得到intrinsic reward。但是本文在理论部分并没有说明获取子目标和内层critic定义的一般性指导方法。另外，本文选取的实验属于本身就具有明显的层级结构，子目标的制定相对而言比较简单。两个实验都可以归结为寻径问题，把子目标量化为靠近最终目标位置的坐标，而critic就是与子目标坐标的距离。