引言：强化学习(Reinforcement learning)是[机器学习](http://www.dataguru.cn/article-4063-1.html?union_site=innerlink" \t "_blank)中的一个领域，强调如何基于环境而行动，以取得较大化的预期利益

背景

强化学习(Reinforcement learning)灵感来源于心理学中的行为主义理论，即有机体如何在环境给予的奖励或惩罚的刺激下，逐步形成对刺激的预期，产生能获得较大利益的习惯性行为。这个方法具有普适性 ，因此在其他许多领域都有研究，例如博弈论、控制论、运筹学、信息论、模拟优化方法、多主体系统学习、群体智能、统计学以及遗传[算法](http://www.dataguru.cn/article-5747-1.html?union_site=innerlink" \t "_blank)。

强化学习也是多学科多领域交叉的一个产物，它的本质就是解决“决策(decision making)”问题，即学会自动进行决策。其在各个领域体现不同：

在计算机科学(Computer science)领域体现为机器学习算法

在工程(Engineering)领域体现在决定序列行为(the sequence of actions)来得到较好的结果;

在神经科学(Neuroscience)领域体现在理解人类大脑如何做出决策，主要的研究是反馈系统(reward system);

在心理学(Psychology)领域，研究动物如何做出决策、动物的行为是由什么导致的;

在经济学(Economics)领域体现在博弈论的研究。

这所有的问题最终都归结为一个问题，人为什么能够做出最优决策，且人类是如何做到的。

原理

强化学习作为一个序列决策(Sequential Decision Making)问题，它需要连续选择一些行为，从这些行为完成后得到较大的收益作为较好的结果。它在没有任何label告诉算法应该怎么做的情况下，通过先尝试做出一些行为——然后得到一个结果，通过判断这个结果是对还是错来对之前的行为进行反馈。由这个反馈来调整之前的行为，通过不断的调整算法能够学习到在什么样的情况下选择什么样的行为可以得到较好的结果。

强化学习与监督学习有不少区别，从前文中可以看到监督学习是有一个label(标记)的，这个label告诉算法什么样的输入对应着什么样的输出。而强化学习没有label告诉它在某种情况下应该做出什么样的行为，只有一个做出一系列行为后最终反馈回来的reward signal，这个signal能判断当前选择的行为是好是坏。另外强化学习的结果反馈有延时，有时候可能需要走了很多步以后才知道之前某步的选择是好还是坏，而监督学习如果做了比较坏的选择则会立刻反馈给算法。强化学习面对的输入总是在变化，不像监督学习中——输入是独立分布的。每当算法做出一个行为，它就影响了下一次决策的输入。强化学习和标准的监督式学习之间的区别在于，它并不需要出现正确的输入/输出对，也不需要较精确校正次优化的行为。强化学习更加专注于在线规划，需要在Exploration(探索未知的领域)和Exploitation(利用现有知识)之间找到平衡

强化学习决策实现过程需要设定一个agent(图中的大脑部分)，agent能够执行某个action(例如决定围棋棋子下在哪个位置，机器人的下一步该怎么走)。Agent能够接收当前环境的一个observation(观察)，例如当前机器人的摄像头拍摄到场景。Agent还能接收当它执行某个action后的reward，即在第t步agent的工作流程是执行一个动作At，获得该动作之后的环境观测状况Ot，以及获得这个动作的反馈奖赏 Rt。而环境environment则是agent交互的对象，它是一个行为不可控制的对象，agent一开始不知道环境会对不同action做出什么样的反应，而环境会通过observation告诉agent当前的环境状态，同时环境能够根据可能的最终结果反馈给agent一个reward，例如围棋棋面就是一个environment，它可以根据当前的棋面状况估计一下黑白双方输赢的比例。因而在第t步，environment的工作流程是接收一个 At，对这个动作做出反应之后传递环境状况和评估的reward给agent。reward奖赏Rt，是一个反馈标量值，它表明了在第t步agent做出的决策有多好或者有多不好，整个强化学习优化的目标就是较大化累积reward。

强化学习中Agent的组成

一个agent由Policy(策略)、Value function(价值函数)、Model(模型)三部分组成，但这三部分不是必须同时存在的

Policy(策略)：它根据当前看到的observation来决定action，是从state到action的映射。有两种表达形式，一种是Deterministic policy(确定策略)即a=π(s)a=π(s)，在某种状态s下，一定会执行某个动作a。一种是Stochastic policy(随机策略)即π(a|s)=p[At=a|St=s]π(a|s)=p[At=a|St=s]，它是在某种状态下执行某个动作的概率。

Value function(价值函数)：它预测了当前状态下未来可能获得的reward的期望。Vπ(s)=Eπ[Rt+1+rRt+2+…|St=s]Vπ(s)=Eπ[Rt+1+rRt+2+…|St=s]。用于衡量当前状态的好坏。

Model(模型)：预测environment下一步会做出什么样的改变，从而预测agent接收到的状态或者reward是什么。因而有两种类型的model，一种是预测下一个state的transition model即Pass′=p[St+1=s′|St=s,At=a]Pss′a=p[St+1=s′|St=s,At=a]，一种是预测下一次 reward的reward model即Ras=E[Rt+1|St=s,At=a]Rsa=E[Rt+1|St=s,At=a]

探索和利用

强化学习是一种试错(trial-and-error)的学习方式：最开始的时候不清楚environment(环境)的工作方式，不清楚执行什么样的action(行为)是对的，什么样的action(行为)是错的。因而agent需要从不断尝试的经验中发现一个好的policy，从而在这个过程中获取更多的reward。

在学习过程中，会有一个在Exploration(探索)和Exploitation(利用)之间的权衡。

Exploration(探索)会放弃一些已知的reward信息，而去尝试一些新的选择——即在某种状态下，算法也许已经学习到选择什么 action让reward比较大，但是并不能每次都做出同样的选择，也许另外一个没有尝试过的选择会让reward更大，即Exploration希望能够探索更多关于environment的信息。

Exploitation(利用)指根据已知的信息较大化reward。

举个例子，这两者在选择一家餐馆时——Exploration(探索)会选择你最喜欢的餐馆，而Exploitation(利用)则会尝试选择一个新的餐馆。

理论上，对于有明确目标的算法模型都可以用**强化学习**来尝试，如游戏AI。不过实际上，强化学习在游戏AI中的应用更多时候是为了**验证和调试算法模型的准确性**。通过传统的简单游戏已经验证了部分强化学习算法，而对于英雄联盟、王者荣耀这种复杂的即时策略类游戏，其算法设计会相对复杂很多

虽然强化学习在传统的机器学习分类中没有被提及，但事实上它已经能够帮助解决深度学习中很多智能控制和分析预测等领域的实际问题，比如通过用户反馈来优化人机之间的多轮对话策略，以及在自然语言生成中。

就目前来说，在王者荣耀这类“复杂”的游戏中训练出高水平“类人” AI的难度确实大，不过在此讨论的是有什么设计算法的思路和需要考虑的地方。

1. 首先，如何定义“类人”

AI的行为是动态、有记忆的，是能基于新的环境信息做调整的。

类似王者荣耀的游戏 dota, 推测其内置人机对战的AI并非reinforcement learning （“推测”不代表一定），而是可能由大量逻辑堆积的，执行逻辑是固定的而不是动态的。玩家和开发者也可用通过dota bot script (lua) 进行自定义AI

AI实现 ultimate goal (获胜) 的手段是 policy，而非通过利用电脑超越人的“手速”优势来获胜。

计算机具有极快的速度控制能力，但是这并不一定表现出智能，因此agents必须在“Actions Per Minute”方面，在人的灵活性的限制下与游戏进行交互。

在只考虑对一个AI(agent)做强化学习调教的情况下（实际情况是**multi-agent system, 需要协作和群体智慧**），需要定义好：**Environment, agent state**

传统适合RL的简单游戏例如“Cart Pole”，甚至“ Flappy Bird”，他们的时空抽象化都是简单的，且对于goal（目标），其environment 是 fully observable（完全可观察的）。

对于王者荣耀这类游戏，一方只能基于友方单位视野看到地图，所以 **environment 是部分可观察的**，这就需要增加一批探索阴影地图的action。对于 state 的抽象描述，基本就是参考2-D matrix来描述地图了（方便作为DQN CNN的输入）

王者荣耀这类游戏的复杂在于，**不能用单一空间坐标来描述state**，state的所有潜在“feature”至少包括：各单位的位置和攻击，地图可视范围（白天和晚上也不一样），各单位目前携带物品及使用范围，各单位目前所学技能及施法范围，各单位血量蓝量等所有会影响action decision的，需要把这些 “feature”分成不用的layer, 每个layer类似DeepMind对星际争霸采用的 低分辨率的RGB图像数据的地图和小地图，如左图

不同于传统适合RL的简单游戏，如 “Flappy Bird” Flappy Bird的action只有左右或上下的移动。

而类似王者荣耀这类游戏，其复杂之处在于**action 多**，例如基础的action：移动类的有8个移动方向，还有普通攻击类、使用技能类、买卖物品、使用物品等。

**看起来基础的 action 的个数也不是很多，怎么说action多呢？**

简单来说，比如Flappy Bird的基础的action 只有2个：{上，下}移动，但Multi-Step Plan Environment 中考虑的 the action set is the set of all possible n-length action sequence，也就是把基础的action序列作为真正考虑的 action set 中的一员，e.g. 当n=20时，基础的action为{上，下}移动2种，也会有2^20=1e6量级的action数目。

至于为什么要用一串基础action的序列来作为一个实际action，考虑到 n-length 无论n取1还是大于1，相对的完成goal所需的实际steps数和n是倒数关系，所以是否需要考虑 n-length action sequence ，是需要视实验和所处问题的实际情况考虑的

例如王者荣耀游戏的真实玩家操作会是一系列基础action, 比如不会走向友方单位并攻击，不会一买完物品并卖掉，所以**实际action用n-length action sequence的话，在遇到数目大的情况下，有一批针对性的解决办法：**

用Wolpertinger 框架, 其中的 action embedding 可以既能“generalize over the set of actions”又能在性能上做到“sub-linear complexity relative to the size of the action set”，因此对于action 多的问题，还是有希望解决的。

在这篇文章提出用Wolpertinger 框架：先将状态与连续的动作空间对应，得到一个属于连续空间的“原型”动作(proto-action)。再把它映射到原动作空间去。

其实就是寻找k个最接近的原始动作，框架还是actor-critic的框架，多了这个离散到连续，连续到离散的转换，也就能够处理大规模的离散问题了

另有一方面的工作是：**基于用 multi-class classification的思想去解决action 多导致的scale问题、基于error-correcting output codes(ECOC是对class数大时结合binary classifier对multi-class classification的优化)、把learning complexity 从 O(A^2) 降低到了O(Alog(A))和O(log(A))**

Ultimate goal 肯定是我方推倒敌方基地获胜，但如果将goal直接设定为这个的，reward将是延迟很多的。如此一来在sparse feedback情况下，agent不太能学好（尽管可以用epsilon-greedy的方法让agent多尝试不用的action）

于是这里提出了一种 **hierarchical-DQN (h-DQN) framework**，在时间尺度上切分出 intrinsically generated goals，让agent在intrinsically motivated 的情况下探索新的行为去解决这些intrinsically generated goals，**最终 learn an optimal policy to chain them together.**

虽然就目前来说在王者荣耀中实现“类人“AI会比较难，并且更多真实世界里的任务是难于游戏AI的，但人也要像 RL 里的 agent一样不断exploration& exploitation，才能实现最终的 goal.

另外对于王者荣耀、星际争霸这类游戏，除了RL以外，其本质上也是**multi-agent system**问题。