(http://lib.csdn.net/base/deeplearning)

深度学习 (http://lib.csdn.net/base/deeplearning) - 深度学习特征表达 (http://lib.csdn.net/deeplearning/node/752) - 强化学习（Reinforcement Learning） (http://lib.csdn.net/deeplearning/knowledge/1750)

谈Reinforcement Learning与能够学习进化的程序

作者：ppn029012 (http://my.csdn.net/ppn029012)

1.婴儿怎么学习?

举个例子，回想一下当您刚出生时，您是怎么学习抓住一个物体的? 很多人都觉得，我们生来就会啊， 这还用学? 但是科学证明，婴儿学会抓住一个物体的技能需要大概三个月来完成。而且，这个过程是由婴儿独立学习完成的，可以完全不需要人的指导。这个过程是怎么完成的呢?

首先，婴儿必须能看到物体，然后控制手臂上的肌肉，使手不断地靠近目标。这是一个很高难度的过程，首先，假设人的手臂上有20块肌肉, 每块肌肉10种不同程度的伸缩度，而手靠近目标需要花费大概5秒的时间，那么每一秒，我们将面临10^20种选择，那么假设花5秒去靠近物体，那么总共会有10^20^5种肌肉控制方式的可能，假如让你每秒试10000次(夸张...), 那么你还是得花上几十万年才有可能碰到你亲爱的小狗什么之类的...

一个简单的抓取物品的手臂控制就几乎不可能靠简单地尝试完成，那么其他的活动您也别参加了，好好在家躺着就好了。

以上讲的尝试的方法是可怕的。 一个人只靠盲目地尝试，甚至不能在地球毁灭之前学会控制自己的一只手臂。 所以我们肯定遗漏了些什么东西，这些东西能够帮助我们更快更好地去学习， 不管是抓取物品还是呀呀学语。

现在想一想，我们人在这个学习中能获得什么信息? 对了，我们自己能看到自己手的位置, 这个信息正是帮助我们学习的关键。 每次我们尝试不同的肌肉强度组合，我们都能看到, 手位置变化的结果。我们能够通过手位置的变化，获得有用的调整肌肉强度的信息， 从而使得整个使用肌肉的学习具有强烈的导向性。而我们会记住这些肌肉使用的规律，并且在日后的学习中，不断地改进规律，最后使我们使用手臂简单而轻松。

除了婴儿学抓东东， 当然还有一些更为"计算机科学家"喜闻乐见的学习例子，比如学下象棋。而我们在熟悉了象棋规则之后，就能通过自己与相同或者不同的人对弈，就能不断提高自己的棋艺，甚至成为一代高手。但是仔细地看学下棋这个过程， 每一步棋的时候，都会有大概50种选择，那么50步的棋就会有50^50种策略的选择。又是一个天文数字，我们怎么能够就在几十或者几百次尝试之后就能渐渐地找到一些很好的策略，来帮助我们获得胜利呢?

在我们尝试地解决以上问题之前，让我们弄清楚我们最后想要的是什么东西----是一个策略(Policy). 这个策略会告诉我们，在某一步(或某一个状态)该进行什么动作。 策略会将 (状态域) ---影射到 --> (动作域). 比如在下象棋时，别人中炮将军时，你可以选择飞象，或者可以把将移开，... and so on. 又如婴儿学动手， 当手在某一个位置时，策略会告诉你，下一步应该怎么动哪几块肌肉了，于是您在策略的倾情指导下，就能够不断的靠近目标，完成所想之事。

2. 搜索策略?

如何在茫茫策海中，找到我们心宜的那个最优策略? 于是我们整个问题就变成了一个在大空间内搜索的问题了.

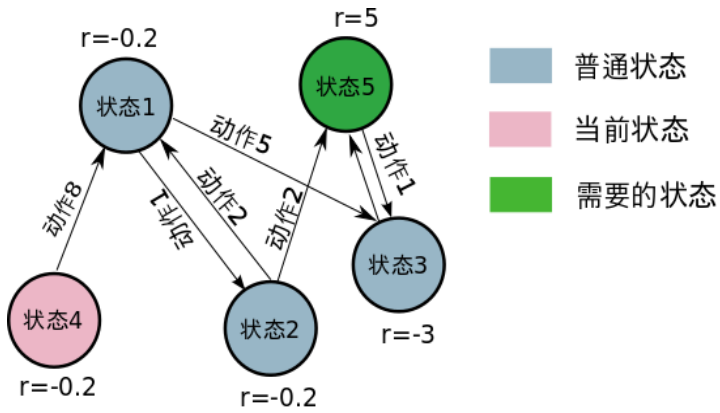
最笨的方法是，把所有的策略都试一遍，然后比较哪一个策略更好。而策略的数量太多了，根本不可能把所有策略都找到，并且比较。

所以有很多先进的找策略的方法, 比如使用遗传算法，随机把(策略1, 2,3,4,...,M)看一个个种群，然后，通过比较这些种群的适应程度(比如说策略1能让你下棋100局赢20局，那么适应度就是0.2)，进行选择， 复制， 变异，让那些表现不错的策略的种群数量更多一些，这样最后就能够找到一些较为满意的策略。

遗传算法的确能够让你找到一些较为满意的策略。但是在状态空间太大的时候，遗传算法需要的时间和空间都比较多（因为你需要一个庞大的种群来实现并行的搜索）。

增强学习(Reinforcement Learning) 将是另一种强大的策略搜索工具，特别适用于有交互的学习过程，也就是说问题可以抽象成状态和动作的变化的模型。比如说打篮球， (你手的位置和速度, 球所在的位置，球的速度) 就是一个状态， 而你的动作，将是（对肌肉的作用1，对肌肉的作用2...n）每块肌肉的强度指令。 而你的动作将会对你的下一个状态产生影响， e.g.改变你手的位置，速度，球的位置和速度。 一旦你的问题能够抽象成这种 状态 与动作的模型，而 状态之间的转移又是靠动作来发生的。 那么就能够通过RL进行学习，从而改进你对 每次动作的选择，最后达到最优。

而策略在RL，中的表示就是，某个状态该选择哪个动作的对应关系。 更美丽地说， 你需要用RL来解决问题，你的问题必须能抽象成像下图这种模型。



RL 需要处理的问题的模型, (r 是reward, 表示每个状态的回报)

有了RL的模型，那么RL解决问题的本质就在于, 通过赋于每个状态一个reward, 和这些状态的转移特性，我们能够找到一个状态转移的策略，使得所有经过的状态的回报总和(价值)达到最大。像上图中， 当前处于状态4，RL解决问题的方法在于，找到一个策略，能从(状态4 --> 状态1 --> 状态5). 这就是这个模型下最优的策略。 这个策略就像一个查询表(look-up table), 将状态对应到动作上.

状态	动作
4	8
1	1
2	2
3	1
5	N/A

一个最优的策略的例子

总的来说， RL的学习过程就是， 先定义状态的回报（比如赢棋为1, 输棋为-1, 其余为-0.02）. 然后通过不断地尝试，从而统计出状态之间的转移关系。 最后通过动态规划求解出一个最优策略。 我们能够通过不断地尝试，获得更多的状态 和 这些状态之间的更精确 转移概率。 于是我们就能通过 动态规划 求解出一个更好的策略来。 所以 (更多的经验和尝试) 将带来 (更好的策略)。

当然，这篇文章里，只力求能给你一个增强学习(RL)的大致概念。更精确的模型会有更加诱人的特性。比如

1. 状态之间的转移并不是确定的，而是概率分布的，比如中国象棋中，你中炮将军的话，下一个状态很可能是，别人"化士", 或者 把“将”移开，或者吃掉你的"炮". 所以这时候，你的状态转移模型就需要用概率转移模型来表示了。 但类似的是，不管是概率分布的概率转移模型还是确定的状态转移模型，我们都可以通过 尝试，并且统计尝试的结果来获得这些转移模型。

2. 价值 不再是 该策略下经过的状态回报总和了, 而是一个期望值。

p.s. 更精确的模式的定义，和更精确的描述可以参考第5部分的一些资源。 需要详细了解的内容可能有， 如何定义价值函数，如何建立概率转移模型，如何使用动态规划。

3. 学习系统的建立

所以RL的学习系统的建立分为几个步骤:

1. 把世界抽象成 具有状态, 动作(选择)会影响状态的变化的模型(马尔可夫决策过程)。

2. 确定这个学习系统的目标，从而为这个世界中的一些状态赋以收益值。目标就是获得最大的收益的总和(价值)。 设计收益=给定目标

(设定完以后，你现在拥有: 一, 收益分布函数(每个状态对应的收益是多少), 二, 状态的集合， 动作的集合)

---------- 设定结束， 以下是学习----------

3. 让学习系统随便跑几次，通过统计获得状态转移概率。

4. 使用 动态规划 求解出在当前 状态转移概率下的 最优的 价值，和最优的 策略。

4. 使用RL解决一个会下棋的程序

如果你对上面的流程还不是太清晰，那么下面的例子会帮助你看到怎么使用RL建立一个 可学习的模型， 并且如何在不断的尝试中成长!!!

一个最简单的下棋游戏, Tic Tic Tac. 如果不清楚这个棋是什么。。。那算了...没有童年,问问你旁边的人吧...

这个游戏一共有9个格， 可能出现的状态有1024个. 现在我们要用RL建立一个模型，能够使其在不断尝试中进化，学习，改进自己。

3.1. 首先我们要建立状态的描述.

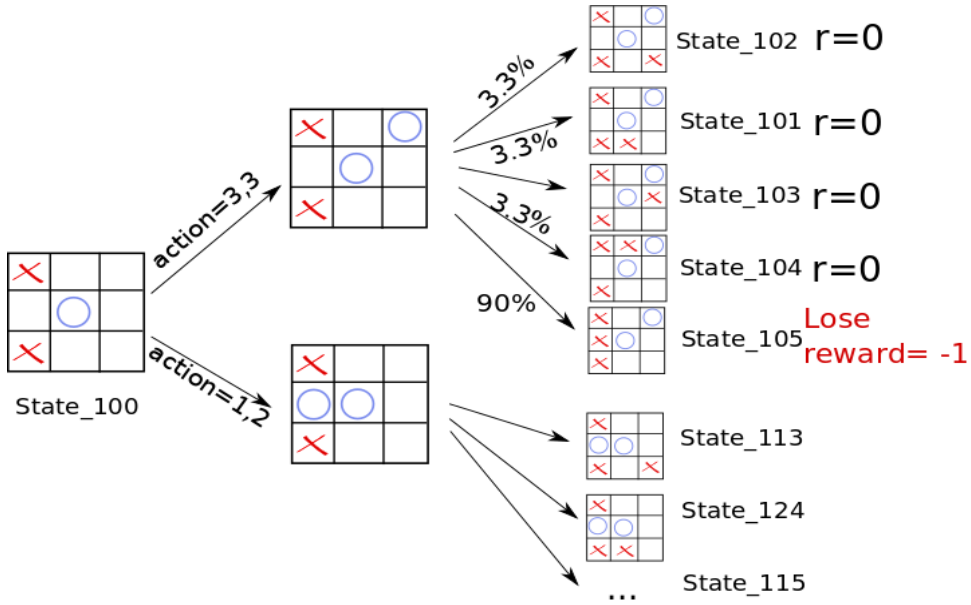
状态 = [x11, x12, x13; x21, x22, x23; x31, x32, x33]; x 的值可能为3个值 (0 无子， 1 下O, 2 下X).

3.2 定义状态的回报(reward)

这时候，我们定义只有获胜的情况reward = 1, 其他状态的reward = 0.

3.3. 通过尝试获得状态转移概率表

通过多次尝试得出一个状态转移概率表， 如下图, 可以计算出状态100到状态101, 102,...113, 124, 115的转移概率。



3.4 通过已知的转移概率表，用 动态规划 计算出 每个状态的价值(Value)， 和一个最优的策略。 这里很显然，action 1,2 的选择要比 action 3,3要好。最优的策略肯定会选择action 1,2而不会选择3,3.

3.5 通过不断地 尝试获得更多的状态， 和更精确的状态转移概率。 从而，再进行3.4步， 得到更优的策略。

5. RL与其他可用的思想算法之间的比较

5.1. 搜索树

搜索树需要对后面的N步进行模拟，并且对结果进行统计分析。需要对问题进行庞大的预测和搜索， 这在更复杂的游戏中，是很难实现的。

可以看到，RL最大的优点在于，RL善于处理那些拥有很大未知性的问题。 并且可以通过，已知的知识建立起一些很好的策略。最大程度地去解决问题。它并不需要巨大的搜索空间，模型的建立是通过程序的不断尝试和交互进行改进的。

5.3 增强学习 是监督学习吗?

监督学习(supervised learning)和RL的区别在于， 监督学习必须提供十分精确的例子。 比如说学习下棋的时候，必须给出每一步的例子，进行训练。 或者在训练一个声带系统发声的时候，需要给出每块声带肌肉震动收缩的例子。 但是实际上，人们有时候很难得到完整精确的例子（比如说打球的时候，身体每块肌肉的运动的例子），却只能给出每次尝试以后的结果，比如说，这次击球的误差，声带系统发声的相似程度，或者告诉你这盘棋的最后结果。

而且RL学习的系统，给出的反馈往往不是实时的，而是有延时的， 也就是你下棋，下了N步之后，在最后的一步才能得到评价输或赢的反馈。 而你必须使用这些反馈去指导你之前做决策的过程。 这种有延时的反馈信息， 很难被监督学习所利用。 监督学习更多的是去学习，同一个时间内，两个事情的对应关系。

6. 学习RL的资源

Andrew Ng 讲的RL，真是不错. http://www.youtube.com/watch?v=RtxI449ZjSc (http://www.youtube.com/watch?v=RtxI449ZjSc)

introduction to RL: http://webdocs.cs.ualberta.ca/~sutton/book/the-book.html (http://webdocs.cs.ualberta.ca/~sutton/book/the-book.html)

希望这篇文章能对你有所帮助, 有关博客的内容，可以发邮件进行交流, pynblogs AT gmail

查看原文>> (http://blog.csdn.net/ppn029012/article/details/8666328)

1

看过本文的人也看了：

- 深度学习知识结构图 (http://lib.csdn.net/base/deeplearning/structure)
- Deep Reinforcement Learning 基础知识（... (http://lib.csdn.net/article/deeplearning/54744)
- 【深度学习】使用tensorflow实现VGG19... (http://lib.csdn.net/article/deeplearning/62921)
- 俄罗斯方块AI_强化学习 (http://lib.csdn.net/article/deeplearning/54749)
- Policy Gradient Methods in Reinforceme... (http://lib.csdn.net/article/deeplearning/68147)
- tensorflow41《TensorFlow实战》笔记-08... (http://lib.csdn.net/article/deeplearning/68146)

发表评论

输入评论内容

发表

0个评论