**上限置信区间算法(UCT)**

# 前言：

以DNN为基础的机器学习（），本质上是在寻找插值函数f，或者说DNN的网络结构，参数就构成了这个函数本身（这个函数也就是我们通常所说的**规律**）。我们用从标签数据那里学会的规律，来推断无标签数据。如果我们成功了，那么就说明我们的学习到了这个规律。学习也就成功了（比如最著名的CNN卷积神经网络）。

但是，这样一套理论，在现实中很多地方却感觉它用不上。例如：如何练习好唱歌？如何提高学习? 如何种菜？如何在围棋中打败对手？这些问题我们自己光凭经验就知道，它们是**有规律**的，而这种规律肯定是能被DNN学习的，但它们是**时间连续**的。我们无法以某一时刻的状态，来识别未来的结果。

经常有人会在牌局结束后，大叫：“那个谁谁谁，你那一盘那张牌，打——得——臭”，“你该这样....这样....，就翻盘了。”

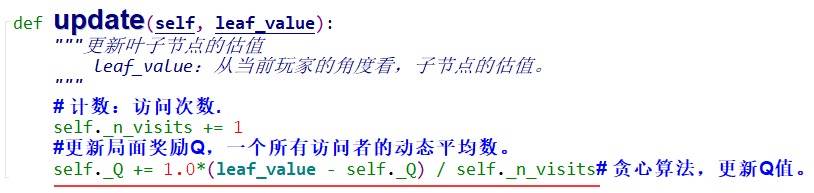
于是，有人想到了。那么，我们把整个过程拆分成n个状态(Sn)；导致Sn-->Sn+1的状态改变的操作(策略/落子)，我们把它叫做操作(An）。这样(Sn，An)的一个集合，最终就能构成一个时间上完整的事情了吧。事情有个了结，那么给它一个评价，就不是那么难了吧？ 我们就把这个评价当成所有(Sn，An)的目标，这也没问题吧？

# 一、贪心算法

令Q(k)记录操作k次的平均奖赏，每次的奖赏为,,...。则平均奖赏为：

上式在计算时，需要记录n个奖赏(,,...)。为了高效，使用增量计算。即每次计算。则更新公式为：

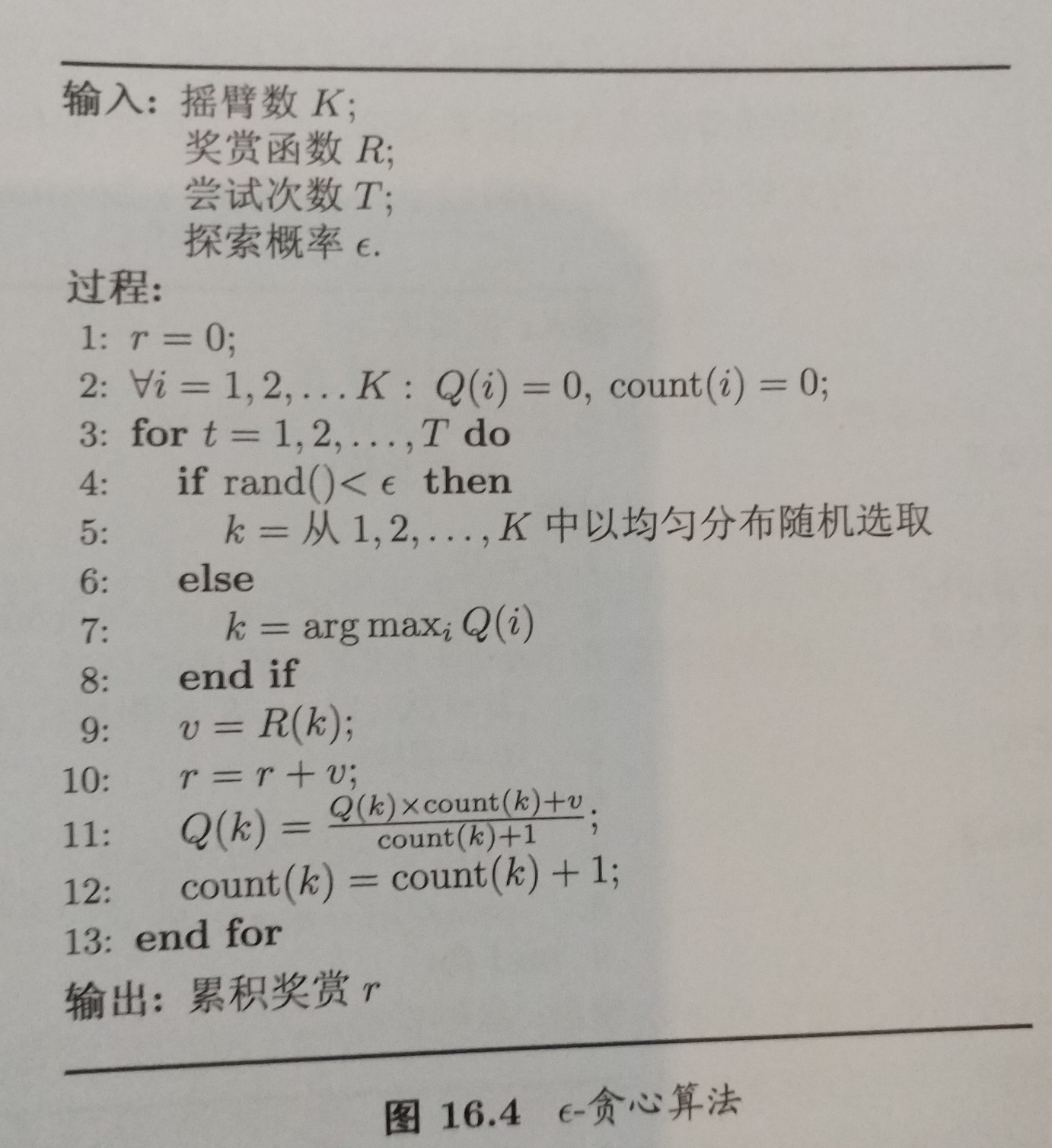
以上公式常用于代码中，更新局面s的奖赏：



**贪心**：

当我即想要探索的**深度**，又想要探索的**广度**时，就用一个概率（(0,1)）来控制。以e的概率进行广度选择，以1-e的概率进行深度选择。

伪代码如下：



通常e比较小，如0.1或0.01。而且当尝试次数足够后，不再需要**广度**，这时，可使用降火算法，使**。**

# 二、Softmax算法

这个在其它地方有详细介绍，就不说明了。