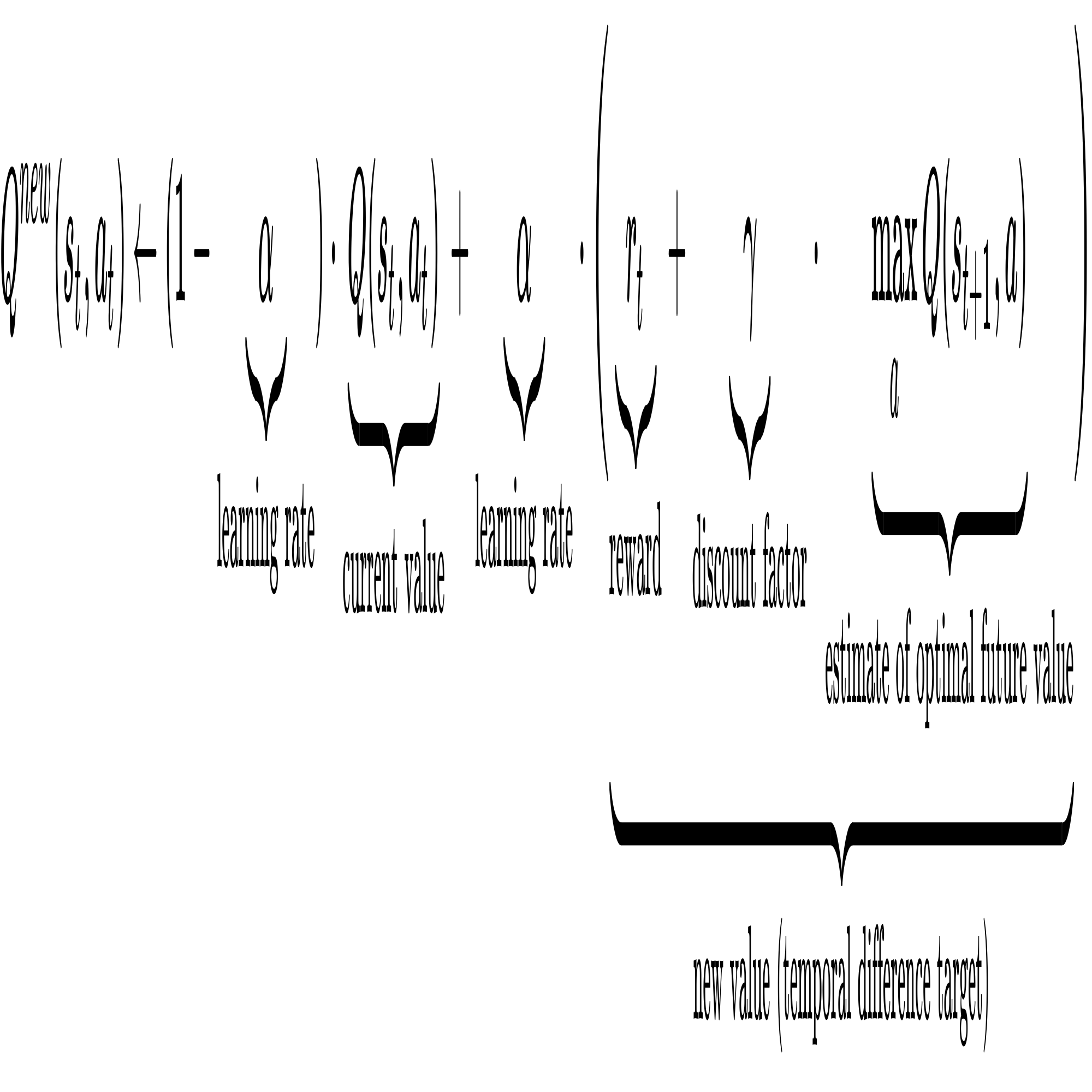
1 理论介绍

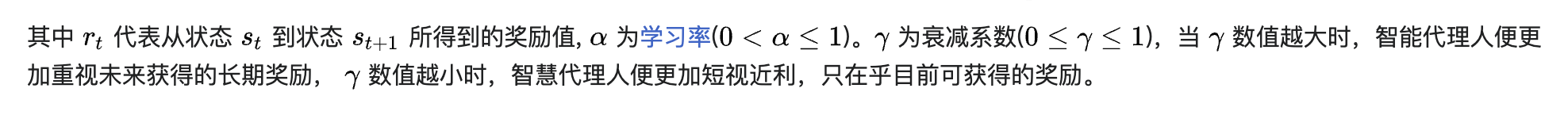
1. *Learning:*

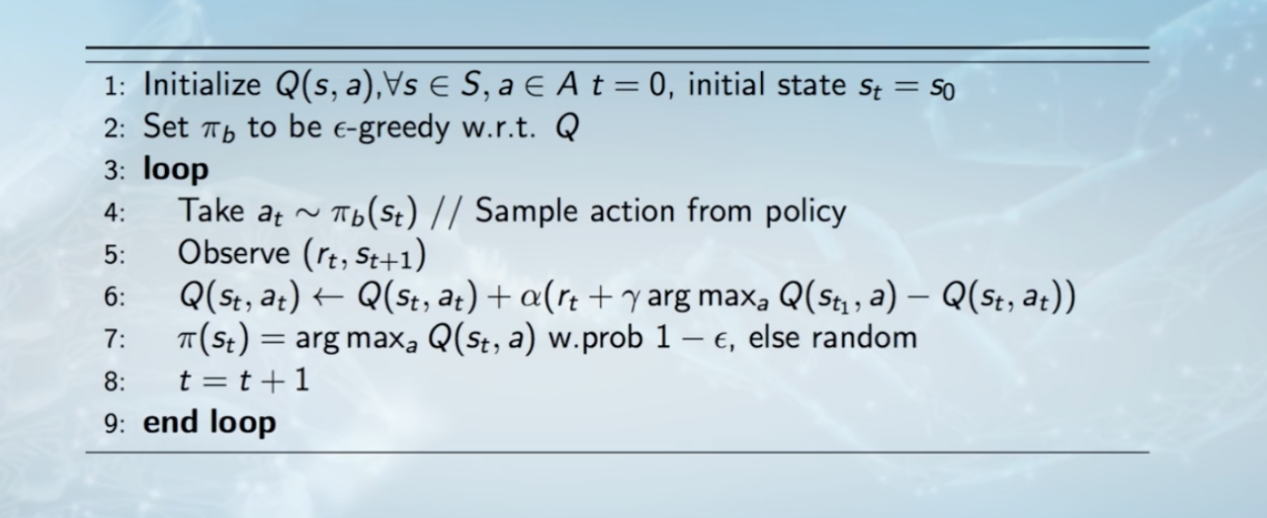
Q-Learning是一种模型无关的强化学习算法，用于记录学习过的策略，并从而告诉智能体什么情况下采取什么行动会有最大的奖励值。Q-Learning不需要对环境进行建模，即使是对带有随机因素的转移函数或者奖励函数也不需要进行特别的改动就可以进行。

对于任何有限的[马可夫决策过程](https://zh.wikipedia.org/wiki/%E9%A6%AC%E5%8F%AF%E5%A4%AB%E6%B1%BA%E7%AD%96%E9%81%8E%E7%A8%8B" \o "马可夫决策过程)（FMDP），Q-Learning可以找到一个可以最大化所有步骤的奖励期望的策略。在给定一个部分随机的策略和无限的探索时间，Q-Learning可以给出一个最佳的动作选择策略。



Q-Learning算法的主要内容为计算状态与行为对应的最大期望奖励函式子Q: S × A → R . 上图为Q-Learning的更新公式。其本质是对Bellman optimality equation的求解。





上图为Q-Learning的算法伪代码。

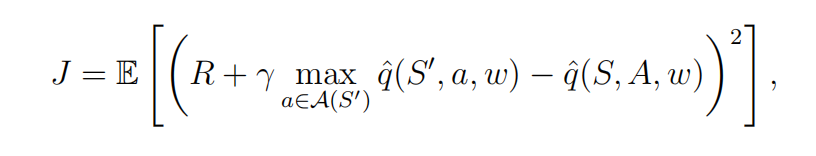
Q-Learning最简单的实现方式就是将奖励值存储在一个表（Q-table）中，但是这种方式经常受限于状态和动作空间的数目。

解决这个问题常用的一个方案是以使用人工神经网络来进行函数逼近。函数逼近的方法在一些问题中会有很好的加速效果，某些时候算法可以通过早期经验的总结可以在一些未出现的状态中依然可以有很好的效果。

*Deep Q-Learning:*

我们可以将深度神经网络与Q-Learning融合起来，获得一种叫做Deep Q-Learning或Deep Q-Network（DQN）的方法。Deep Q-Learning是最早和最成功的深度强化学习算法。值得注意的是神经网络的设计不一定要十分深入。对于诸如我们的游戏中的较为简单的简单任务，具有一个或两个隐藏层的浅层网络可能就已经足够高效。

Deep Q-Learning的目标是求出一下目标函数的最小值。



其中(S, A, R, S’)均为随机变量，分别代表状态，动作，即时奖励，以及下一状态。此目标方程可被视作平方贝尔曼最优性误差。

我们需要着重介绍Deep Q-Learning运用到的一些特殊技术：

·第一种技术是使用主网络和目标网络这两个网络。设w和wT表示主和目标的参数网络。它们最初设置为相同的值。

在每次迭代中，我们从回放缓冲区中提取一小批样本{(s，a，r，s0)}。主网络的输入是s和a. 输出y= q\_hat(s，a，w) 是估计的q值。输出的目标值是截屏2023-11-30 10.22.23，通过在样本{(s, a, yT)}中最小化损失函数来更新主网络。

主网络对w的更新以来对神经网络的训练。故我们需要一部分小样本来训练网络，而不是用单个样本来更新主网络。

主网络每次迭代都会得到更新。

·第二个技术是经验回放。当我们收集一些经验样本后，我们不按照收集的顺序来使用它们，而是将它们储存在回放缓冲区里。一个具体的例子是设(s, a, r, s’)为经验样本，B={(s, a, r, s’)}为回放缓冲区。每次更新主网络时，我们可以从回放缓冲区中提取一个batch的经验样本。对于样本的提取，或者称为经验回放的过程，要求符合均匀分布。

为什么我们需要经验回放呢？而为什么又要求符合均匀分布呢？特别的，要想明确定义目标函数，我们必须弄清楚S, A, R, S0的概率分布。一旦(S, A)给定，R和S0的分布由系统模型决定。而最简单的描述(S, A)的方法就是假设它是均匀分布的。