# 重要概念

## QN基本概念

在强化学习中，四个最基本的元素是

s : 当前状态state

a : 当前状态下能够采取的行动action

r : 在当前状态s下采取行动a之后所获得的即时奖励reward，可以是正激励，也可以是负激励。

: 当前状态s下采取行动a之后达到的下一状态

在DQN中还有另外一个重要概念：经验q

: 在状态s下执行a行动所获得的即时奖励

: 在状态s下执行a行动所获得的经验分数

奖励矩阵是固定的，经验矩阵则是根据的值一步步计算出来的：

* 初始时都是0，通过上式不断迭代最终计算出一个不变的经验矩阵
* 是学习率，是折损率，都是人工设置的
* 代表下一状态采取的所有行动中经验最高的

当学习率时，

在计算出经验矩阵之后，对于任意状态都可以根据选择最优的行动。

## DQN算法设计

准备工作：奖励矩阵，提前设计好每种状态下采取某种行动后得到的奖励分数的计算方式。

模型输入：当前状态s

模型输出：当且状态s下能采取的所有行动，对应的所有经验分数，记为

模型损失函数：

* Q是模型的输出
* target\_Q类似于数据的标签，

target\_Q的计算方式：

1. 一开始是通过以下公式计算：

即

其中可以通过将输入模型得到

1. 后来使用网络模型计算得到，该模型和求Q得模型结构相同，不过参数更新具有延时。

训练数据来源：

构造一个简单的智能体去探索环境。即随机初始化状态，再随机选取行动，不断的循环执行，并记下每一步的，这样就构造了一个记忆库。每次从记忆库中随机抽取一部分

## 算法流程

1. 初始化参数，随机设置一个状态s
2. 循环遍历以下操作：
3. 根据策略选择一个行动a
4. 执行该行动a，获得奖励r，达到下一状态，并判断游戏是否结束
5. 将存入记忆池
6. 记忆池数量达到一定条件：
7. 从记忆池中随机抽取一部分作为训练集
8. 将所有训练集中的s输入模型，得到
9. 根据公式计算出target\_Q
10. 计算loss，更新模型
11. 判断游戏是否结束：
12. 结束：随机设置一个状态s，重新循环
13. 未结束：用更新s

# 代码实现