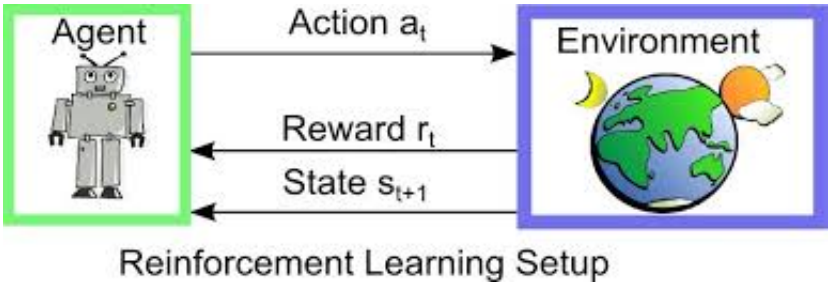
Q - Learning

1. 强化学习



Reinforcement Learning Setup

强化学习通常包括**两个实体agent和environment**。两个实体的交互如下，在environment的state s_t 下，agent采取action a_t 进而得到reward r_t 并进入state s_{t+1} 。

强化学习的问题，通常有如下特点：

- 不同的action产生不同的reward
- reward有延迟性
- 对某个action的reward是基于当前的state的

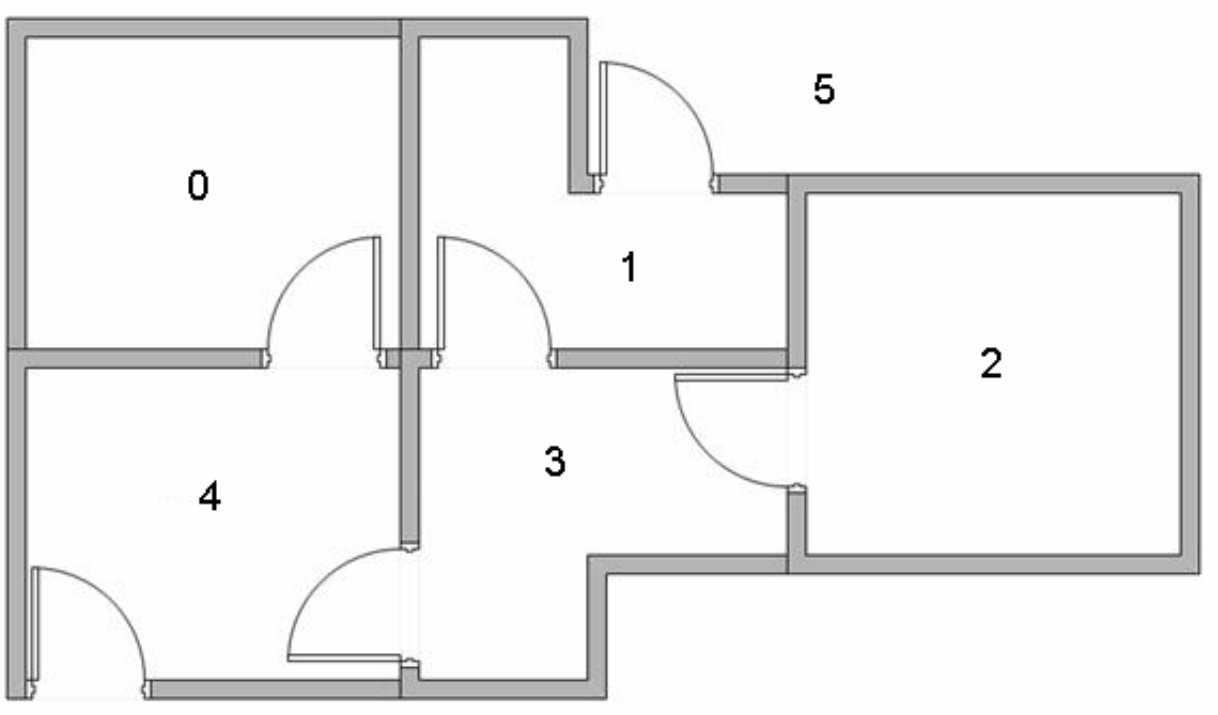
2. Q-Learning

Q为动作效用函数（action-utility function），用于评价在特定状态下采取某个动作的优劣，可以将之理解为智能体（Agent）的大脑。

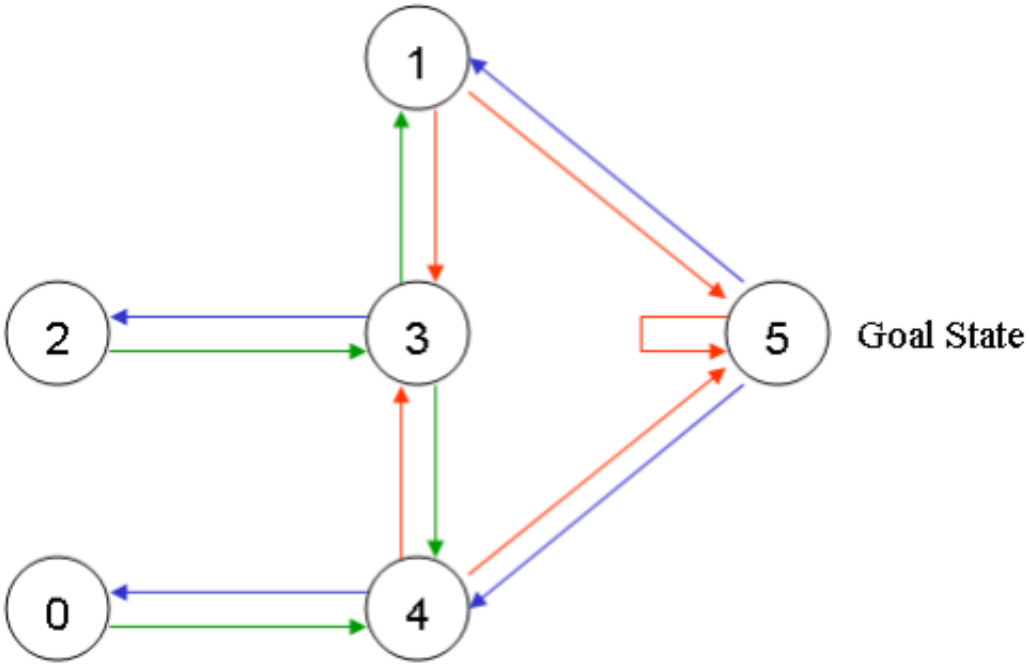
2.2 Example

2.2.1 模型建立

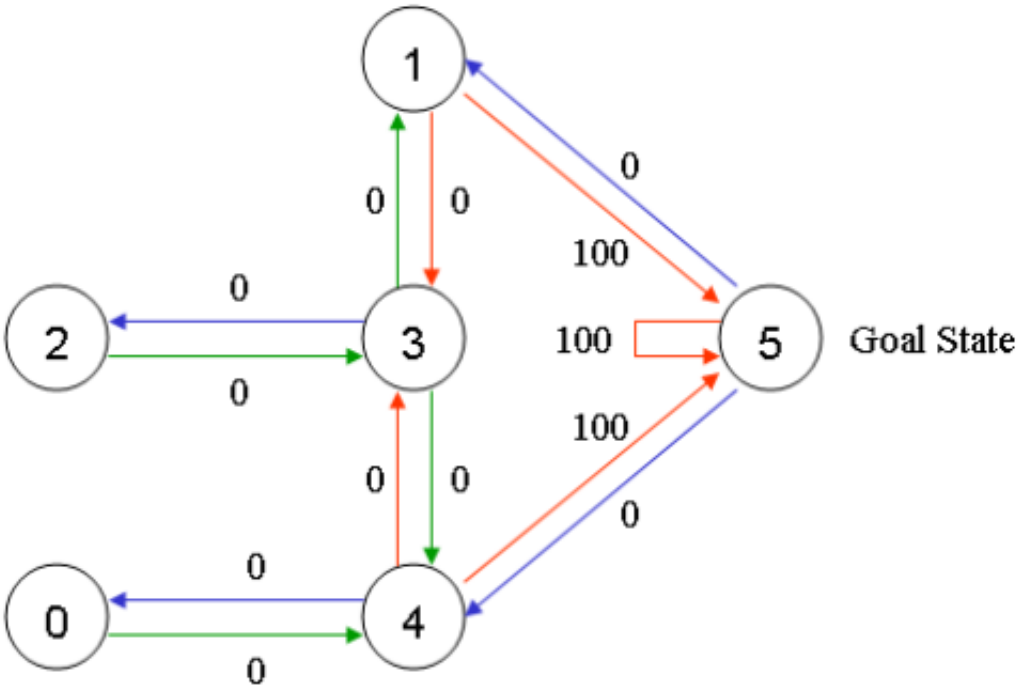
假设有一个楼层，该楼层内有5个房间，房间之间通过一道门相连，如图所示。将房间编号为房间0到房间4，楼层的外部可以被看作是一间大房间，编号为5。注意到房间1和房间4可以直接通到房间5。



我们可以用图来表示上述的房间，将每一个房间看作是一个节点，每一道门看作是一条边（链路）。



在这个例子中，我们在任意一间房间中放置一个智能体（agent），并期望该智能体能够从该房间开始走出这栋楼（可以认为是我们的目标房间）。换句话说，智能体的目的地是房间5。为了设置这间房间作为目标，我们为每一道门（节点之间的边）赋予一个奖励值。能够直接通到目标房间的门赋予一及时奖励值100，而其他的未与目标房间直接相连的门赋予奖励值0。因为每一道门都有两个方向，因此，每一道门在图中将描述为两个箭头。如下所示：



毫无疑问，房间5指向自身，且对应的建立为100，而且其它与5号房间相连的房间同样具有100的收益。在Q-Learning中，我们的目标是达到收益最大的状态，到了这个状态之后，我们的agent将会永远停留在那里。这种目标成为“吸收目标”。（对应于本例中的状态5）

根据状态图和即时奖励值建立矩阵R：（表中的-1代表空值，表示节点之间无边相连）

$$
R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{c}
\begin{array}{cccccc} \multicolumn{6}{c}{\text{Action}} \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{bmatrix}
-1 & -1 & -1 & -1 & 0 & -1 \\
-1 & -1 & -1 & 0 & -1 & 100 \\
-1 & -1 & -1 & 0 & -1 & -1 \\
-1 & 0 & 0 & -1 & 0 & -1 \\
0 & -1 & -1 & 0 & -1 & 100 \\
-1 & 0 & -1 & -1 & 0 & 100
\end{bmatrix}
\end{array}
$$

2.2.2 Q-Table

Q-table：建立一个类似于矩阵R的矩阵Q，代表智能体从经验中所学到的知识。矩阵Q的行代表智能体当前的状态（state），列代表到达下一个状态的可能的动作（action）。

因为智能体在最初对环境一无所知，因此矩阵Q被初始化为0。在这个例子中，为了阐述方便，我们假设状态的数量是已知的（设为6）。如果我们不知道有多少状态时，矩阵Q在最初被设为只有一个元素。如果新的状态一旦被发现，对矩阵Q增加新的行和新的列非常简单。

2.2.2 学习规则与算法

Q-Learning的学习规则如下：

$$Q(S,A) \leftarrow (1-\alpha) * Q(S,A) + \alpha * [R(S,A) + r * max_a Q(S',A)]$$

其中：

- α 为学习速率(learning rate)
- γ 为折扣因子（discount factor）
- $R(S,A)$ 为状态S和动作A在矩阵R中对应的奖励
- $max_a Q(S',A)$ 为下一个状态时Q矩阵中的最大值。

Q-学习算法的计算过程如下：

```
1、设置参数Gamma，以及矩阵R中的环境奖励值；
2、初始化Q矩阵为0；
3、对每一轮学习（one episode）：
      随机选择一个状态；
      Do while 目标状态未到达
         对当前状态的所有可能的动作中，选择一个可能的动作；//策略可以是ε-greedy等
         使用这个可能的动作，到达下一个状态；
         对下一个状态，基于其所有可能的动作，获得最大的Q值；
         计算：$Q(S,A)\leftarrow(1-\alpha)*Q(S,A)+\alpha*[R(S,A) + r*max_aQ(S',A)]$$
         设置下一个状态作为当前状态；
   End For
```

agent利用上述算法不断地从经验中学习，每一轮学习中，它不断地探索环境，然后获得奖赏。训练的目的是强化agent的“大脑”，此处用Q表示。

e.g.

1. 设置学习率Gamma等于0.8，初始的状态是房间1。初始的矩阵Q是一个零矩阵，如下：

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

2. 观察R矩阵的第二行（状态1），对状态1来说，存在两个可能的动作：到达状态3，或者到达状态5。通过随机选择，我们选择到达状态5。

$$
R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{c}
\begin{array}{cccccc} \multicolumn{6}{c}{\text{Action}} \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{bmatrix}
-1 & -1 & -1 & -1 & 0 & -1 \\
-1 & -1 & -1 & 0 & -1 & 100 \\
-1 & -1 & -1 & 0 & -1 & -1 \\
-1 & 0 & 0 & -1 & 0 & -1 \\
0 & -1 & -1 & 0 & -1 & 100 \\
-1 & 0 & -1 & -1 & 0 & 100
\end{bmatrix}
\end{array}
$$

3. 现在，让我们想象如果我们的智能体到达了状态5，将会发生什么？观察R矩阵的第六行，有3个可能的动作，到达状态1,4或者5。 Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)] Q(1, 5) = R(1, 5) + 0.8 * Max[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0.8 * 0 = 100 由于矩阵Q此时依然被初始化为0，Q(5, 1), Q(5, 4), Q(5, 5)全部是0，因此，Q(1, 5)的结果是100，因为即时奖励R(1,5)等于100。 下一个状态5现在变成了当前状态，因为状态5是目标状态，因此我们已经完成了一次尝试。我们的智能体的大脑中现在包含了一个更新后的Q矩阵。

$$
Q = \begin{array}{c}
\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 100 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
$$

4. 对于下一次训练，我们随机选择一个初始状态，这一次，我们选择状态3作为我们的初始状态。观察R矩阵的第4行，有3个可能的动作，到达状态1,2和4。我们随机选择到达状态1作为当前状态的动作。

5. 现在，我们想象我们在状态1，观察矩阵R的第2行，具有2个可能的动作：到达状态3或者状态5。现在我们计算Q值： Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)] Q(3, 1) = R(3, 1) + 0.8 * Max[Q(1, 2), Q(1, 5)] = 0 + 0.8 * Max(0, 100) = 80 我们使用上一次尝试中更新的矩阵Q得到：Q(1, 3) = 0 以及 Q(1, 5) = 100。因此，计算的结果是Q(3,1)=80，因为奖励值为0。现在，矩阵Q变为：

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

6. 现在，想象我们处于状态5，有3个可能的动作：到达状态1,4或5。我们计算Q值如下： Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)] Q(1, 5) = R(1, 5) + 0.8 * Max[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0.8 * 0 = 100 更新后的矩阵Q中，Q(5, 1), Q(5, 4), Q(5, 5)依然是0，故Q(1, 5)的值是100，因为即时奖励R(5,1)是100，这并没有改变矩阵Q。 因为状态5是目标状态，我们完成了这次尝试。我们的智能体大脑中包含的矩阵Q更新为如下所示：

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

7. 如果我们的智能体通过多次的经历学到了更多的知识，Q矩阵中的值会达到一收敛状态，如下：

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array}$$

最终，我们通过学习得到了Q-Table。

2.2.2 Q-Table的使用

当矩阵Q通过不断地训练达到一种收敛的状态的时候，agent已经学得了到达目标状态的最优路线。此时，根据Q-Table可以很容易地得到特定初始状态下的最优路线，算法如下：

```
1、设置当前状态=初始状态；
2、从当前状态开始，寻找具有最高Q值的动作；
3、设置当前状态=下一个状态；
4、重复步骤2和3，直到当前状态=目标状态。
```

e.g.

例如，从初始状态2，智能体在矩阵Q的指导下进行移动： 在状态2时，由矩阵Q中最大的值可知下一个动作应该是到达状态3； 在状态3时，矩阵Q给出的建议是到达状态1或者4，我们任意选择，假设选择了到达状态1； 在状态1时，矩阵Q建议到达状态5； 因此，智能体的移动序列是2-3-1-5。

3.参考链接

1. https://blog.csdn.net/u013405574/article/details/50903987
2. http://mnemstudio.org/path-finding-q-learning-tutorial.htm
3. https://www.zhihu.com/question/26408259