实验课 6：机器学习实验-强化学习

一、实验大纲

1. 强化学习理论背景
2. Q 学习算法
3. 强化学习实验

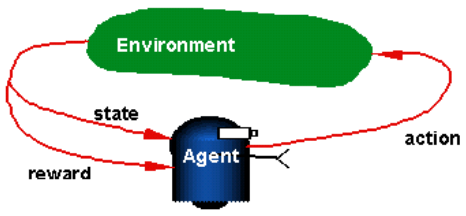
二、强化学习理论背景

智能系统的一个主要特征是能够适应未知环境，其中学习能力是智能系统的关键技术之一。在机器学习范畴，根据反馈的不同，学习技术可以分为监督学习（Supervised learning）、非监督学习（Unsupervised learning）和强化学习（Reinforcement learning）三大类。其中强化学习是一种以环境反馈作为输入的、特殊的、适应环境的机器学习方法。所谓强化学习是指从环境状态到行为映射的学习，以使系统行为从环境中获得的累积奖赏值最大。该方法不同与监督学习技术那样通过正例、反例来告知采取何种行为，而是通过试错（trial-and-error）的方法来发现最优行为策略。

强化学习技术是从控制理论、统计学、心理学等相关学科发展而来，最早可以追溯到巴普洛夫的条件反射实验。但直到上世纪八十年代末、九十年代初强化学习技术才在人工智能、机器学习和自动控制等领域中得到广泛研究和应用，并被认为是设计智能系统的核心技术之一。特别是随着强化学习的数学基础研究取得突破性进展后，对强化学习的研究和应用日益开展起来，成为目前机器学习领域的研究热点之一。

在机器人中的应用，强化学习最适合、也是应用最多的。Wnfriedllg 采用强化学习来使六足昆虫机器人学会六条腿的协调动作；Sebastian Thurn 采用神经网络结合强化学习方式使机器人通过学习能够到达室内环境中的目标；在游戏比赛中的应用，在这方面，最早的应用例子是 Samuel 的下棋程序；在控制系统中的应用，强化学习在控制中的应用的典型实例，就是倒摆控制系统，当倒摆保持平衡时，得到奖励，倒摆失败时，得到惩罚，控制器通过自身的学习，最终得到最优的控制动作。

在强化学习过程中， 学习智能体自身通过训练，误差和反馈，学习在环境中完成目标的最佳策略。我们并没有直接告诉智能体要做什么或采取那个动作，而是智能体通过看那个动作得到了最多的奖励来自己发现。试错搜索(trial-and-error search)和延期强化(delayed reinforcement)这两个特性是强化学习中两个最重要的特性。



标准的强化学习，如图所示，智能体 Agent 作为学习系统，接受环境状态的输入 s，根据内部的推理机制，系统输出相应的行为动作 a。

环境在系统动作作用 a 下，变迁到新的状态 s'。系统接受环境新状态的输入，同时得到环境对于系统的瞬时奖惩反馈 r。如果智能体的某动作 a 导致环境正的奖赏(立即报酬)，那么智能体以后产生这个动作的趋势便会加强；反之，智能体产生这个动作的趋势将减弱。这和生理学中的条件反射原理是接近的。

智能体系统在控制行为与环境反馈的状态及评价的交互作用中，以学习的方式不断修改从状态到动作的映射策略，以达到优化系统性能目的。因此，强化学习的目标是学习一个行为策略 π：$S \rightarrow A$，使系统选择的动作能够获得环境奖赏的累计值最大，也使得外部环境对学习系统在某种意义下的评价(或整个系统的运行性能)最佳。

常用的强化学习算法包括 TD 时序差分(Temporal Difference)算法、Q 学习算法、Sarsa 算法等。其中 Q 学习的算法，就是其中比较好的一种强化学习算法，它可从有延迟的回报中获取最优控制策略。

三、Q 学习算法

强化学习中一类典型的方法就是Q学习（Q-Learning），我们将采用举例说明的方法让大家来理解Q学习的具体内容。

1. 环境建模

如图 1 所示，假设我们有 5 个彼此连通的房间。依次命名为A-E。在这 5 个房间外有一个大房间F，并且通过B和E的门可以到达F。

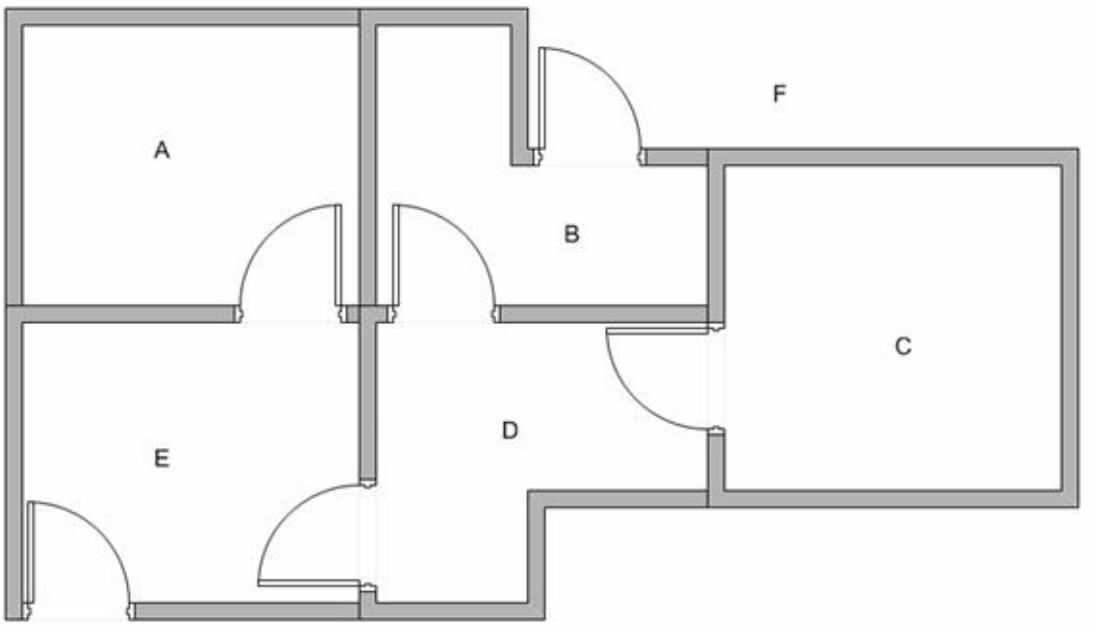


图 1

接下来，我们可以用图来表示上面所提到的这个环境，其中顶点是房间，边是门，于是有图 2。

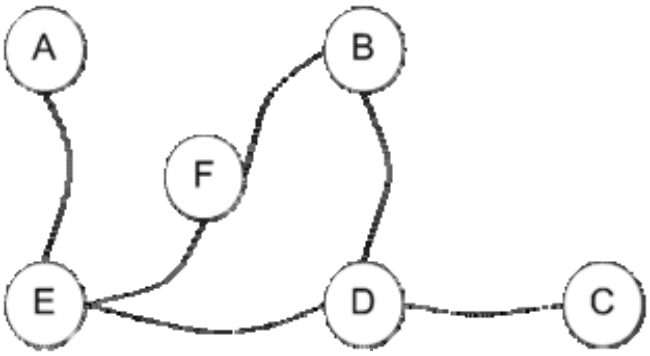


图 2

这个时候，当我们把一个Agent放置到任意一个房间的时候，我们希望Agent能够走出A-E所组成的大房间，也就是说，我们的目标房间是F。设定了这样一个目标，我们给每扇门一个奖励值，直接通往F的门的即时奖励是 100，入图 3 所示的红边，其他没有直接连接到F的边的奖励值为 0，由于门是双向的，因此，该图为一有向图，每一条有向边的权值为即时奖励值。

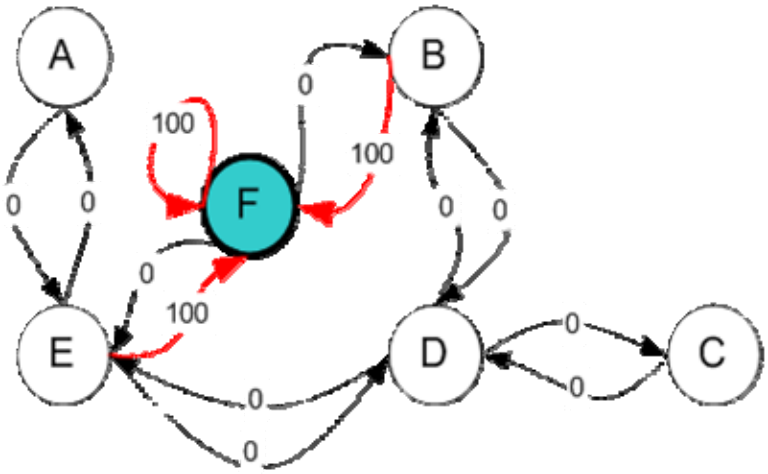


图 3

注意，节点 F 有一自环边，该边的即时奖励值是 100，也就是说，一旦 Agent 到达了目标，它就会一直保持在目标状态处。

2.Agent，状态和行为

假设我们的 Agent 是一个虚拟机器人，它能够通过行为所获得的经验进行学习。这个 Agent 能够从一个房间到达另一个房间，但是它初始时候并没有环境的信息，它并不知道具体应该走哪个门才能走出这间建筑，到达 F。

Agent 穿过房间，走出建筑是我们这个场景中会产生的所有行为，要对其建立某种简单的评估模型，我们可以用如下的方法：现在假设我们有一个 Agent 待在房间 C 并且我们想让这个 Agent 学会如何走出这间建筑到达 F，如图 4。

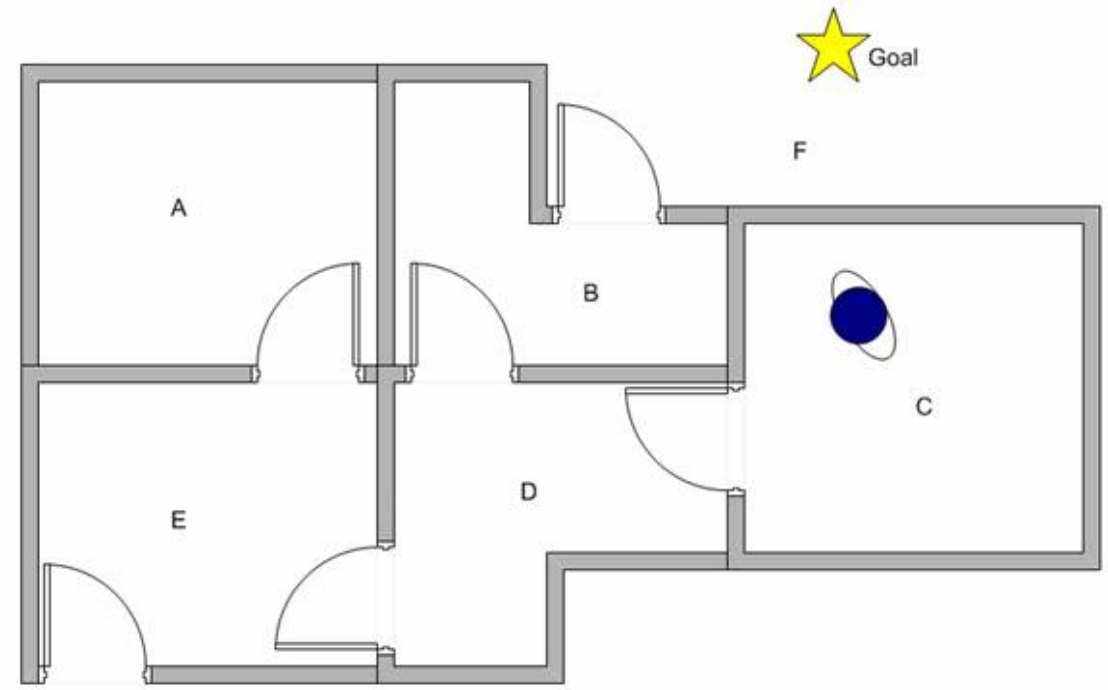


图 4

为了接下来讨论Agent如何通过行为的经验进行学习，我们首先要定义这个场景中所可能产生出的状态以及Agent的行为。即：每一间房间作为一个状态（Agent在某个房间中），Agent的行为是从一间房间走到另一间房间。于是，我们得到了状态转换图，如图 5 所示

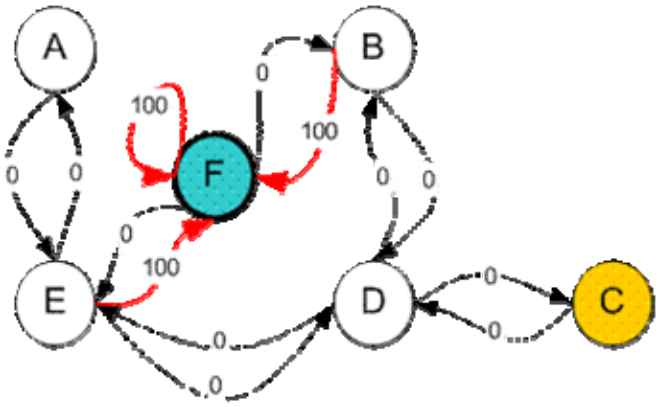


图 5

节点作为每一个可能的状态，带权的边作为行为，边的权值表示的是做出当前这个行为所获得的即时回报值。注意，我们的起始状态是C，根据状态图，我们不能直接从C走到B，因为B和C之间没有直接的边相连，另外，如果Agent处在状态E，那么Agent可能会做出 3 种不同的行为，去A，D或者F。进一步，我们把状态转换图转化为状态矩阵，如表 1 所示。

表 1：动作状态转移矩阵

	Action to go to state					
Agent now in state	**A**	**B**	**C**	**D**	**E**	**F**
A	-	-	-	-	**0**	-

B	**-**	**-**	**-**	**0**	**-**	**100**
C	**-**	**-**	**-**	**0**	**-**	**-**
D	**-**	**0**	**0**	**-**	**0**	**-**
E	**0**	**-**	**-**	**0**	**-**	**100**
F	**-**	**0**	**-**	**-**	**0**	**100**

3.Q学习

前边我们已经对环境以及环境的反馈进行了建模，下面重点讨论 Q 学习算法。具体的方法如下：环境的反馈即表 1 的状态转换矩阵

$$\mathbf{R}= \begin{array}{c} \textit{state}\backslash\textit{action} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} - & - & - & - & 0 & - \\ - & - & - & 0 & - & 100 \\ - & - & - & 0 & - & - \\ - & 0 & 0 & - & 0 & - \\ 0 & - & - & 0 & - & 100 \\ - & 0 & - & - & 0 & 100 \end{bmatrix} \end{array}$$

并且，把该矩阵命名为 R。行表示当前的状态，列表示的是下一个行为所到达的状态。现在我们令设令外一个矩阵 Q，行和列同矩阵 R。该矩阵 Q 表示的是 Agent 的学习过程，矩阵的每个元素(i,j)表示的是 Agent 从房间 i 走到房间 j 的最大收益。初始的时候，这个矩阵是一个全 0 的矩阵。Q 学习的过程可以表示为如下的一个简单的公式。

$$\mathbf{Q}(state, action) = \mathbf{R}(state, action) + \gamma \cdot Max\left[\mathbf{Q}(next\ state, all\ actions)\right]$$

Q 学习的核心就是用上面这个公式来学习出每一个 Q 矩阵中的元素，具体求法是当前状态 state 下做出 action 这个行为的即时回报值 R，外加一个收益最大邻接状态 Q 值乘以一个学习系数 γ。

通过这样一个方法，我们的虚拟 Agent 可以通过经验来学习出在当前环境下达到目标的一个最优方案。在学习的过程中，我们称 Agent 从初始状态到目标状态的一次搜索叫做一个周期（每个状态的跳转的方向可能有多个，因此从初始状态到目标状态存在多条路径，而且在 Q 学习中并不固定初始状态，当 Agent 完成一个周期后，它会随机选择一个状态作为初始状态从而开始下一个周期的探索），Agent 通过若干个周期的探索后就可以从探索的经验中学习到最优的方案。具体的伪代码如下：

Q Learning

输入：状态转移矩阵 R，和目标状态
输出：从任意状态到达目标状态的最短路径
伪代码：

Q Learning Algorithm goes as follow

1. Set parameter γ, and environment reward matrix **R**
2. Initialize matrix **Q** as zero matrix
3. For each episode:
    - Select random initial state
    - Do while not reach goal state
        - Select one among all possible actions for the current state
        - Using this possible action, *consider* to go to the next state
        - Get maximum Q value of this next state based on all possible actions
        - Compute

$$Q(state, action) = R(state, action) + \gamma \cdot Max[Q(next\ state, all\ actions)]$$

        - Set the next state as the current state

End Do

End For

每一个 Agent 进行探索的周期实际是 Q 学习的一次训练，训练直到 Q 矩阵的值不再发生变化为止，这样也即学习出了到达目标的最优方案。

在这一过程中，**学习系数 γ 取值范围是[0,1)，如果 γ 值接近 0，那么表示的是 Agent 更趋向于注重即时的环境反馈值。反之，则 Agent 更加考虑的是未来状态的可能奖励值。**当我们一旦学习出了 Q 矩阵，那么给定一个初始状态，我们可以很容易的找出从该状态到达目标状态的最短路径。其算法如下：

Algorithm to utilize the Q matrix

Input: **Q** matrix, initial state

1. Set current state = initial state
2. From current state, find action that produce maximum Q value
3. Set current state = next state
4. Go to 2 until current state = goal state

---

4. Q学习实例

为了更好的理解 Q 学习算法是如何工作的，我们给出了一个例子的演示，在这里我们选取的学习系数为 $\gamma = 0.8$，第一个周期的初始状态为 B，设置 Q 矩阵为全 0 矩阵，并给出初始状态转移矩阵 R。

$$Q = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \quad R = \begin{array}{c} state \backslash action \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} - & - & - & - & 0 & - \\ - & - & - & 0 & - & 100 \\ - & - & - & 0 & - & - \\ - & 0 & 0 & - & 0 & - \\ 0 & - & - & 0 & - & 100 \\ - & 0 & - & - & 0 & 100 \end{bmatrix} \end{array}$$

观察矩阵 R 可知，状态 B 可能会有 2 种动作，要么去状态 D，要么去 F，B 通过随机选择，最终选了 F，当我们到达了状态 F，根据矩阵 R，F 的会有 3 种动作，B，E，F
则，根据 Q 学习的算法，我们有：

$$Q(state, action) = R(state, action) + \gamma \cdot Max\left[Q(next\ state, all\ actions)\right]$$

$$Q(B,F) = R(B,F) + 0.8 \cdot Max\left\{Q(F,B), Q(F,E), Q(F,F)\right\} = 100 + 0.8\ 0 = 100$$

由于 Q 矩阵初始设为全 0，则 $Q(F,B), Q(F,E), Q(F,F)$ 目前为 0，通过上面的计算结果可知 $Q(B,F)$ 的结果为 100，则下一个状态 F，又由于目标状态为 F，则我们结束了一个周期。则当前 Agent 脑海中的 Q 矩阵被更新为：

$$Q = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

在下一个周期伊始，我们随机选择初始状态，这次我们选择了 D，通过观察 R 矩阵，D 从 3 个随机行为中选择了状态 B，于是，根据 Q 学习算法，我们有：

$$Q(state, action) = R(state, action) + \gamma \cdot Max\left[Q(next\ state, all\ actions)\right]$$

$$Q(D,B) = R(D,B) + 0\ 8 \cdot Max\left\{Q(B,D), Q(B,F)\right\} = 0 + 0.8 \cdot Max\left\{0, 100\right\} = 80$$

通过更新的 Q 矩阵，我们得到了 $Q(D,B) = 80$，则马上更新 Q 矩阵为：

$$Q = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

下一步，状态 B 成为了当前状态，这一次，B 又非常幸运地选择动作去到状态 F。根据 Q 学习算法，

$$Q(state, action) = R(state, action) + \gamma \cdot Max\left[Q(next\ state, all\ actions)\right]$$

$$Q(B,F) = R(B,F) + 0.8 \cdot Max\left\{Q(F,B), Q(F,E), Q(F,F)\right\}$$

$$= 100 + 0.8 \cdot Max\left\{0,0,0\right\} = 100$$

又到达了终结状态，于是我们结束了第 2 个周期，并且由于 $Q(B,F)$ 仍为 100，则此时的 Q 矩阵仍为：

$$Q = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

按照上面的步骤依次进行，最终我们得到了最终的 Q 矩阵为：

$$Q = \begin{array}{c} state \backslash action \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} - & - & - & - & 400 & - \\ - & - & - & 320 & - & 500 \\ - & - & - & 320 & - & - \\ - & 400 & 256 & - & 400 & - \\ 320 & - & - & 320 & - & 500 \\ - & 400 & - & - & 400 & 500 \end{bmatrix} \end{array}$$

通过归一化（除 500 再乘 100），得：

$$\hat{Q}=\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array}\begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} - & - & - & - & 80 & - \\ - & - & - & 64 & - & 100 \\ - & - & - & 64 & - & - \\ - & 80 & 51 & - & 80 & - \\ 64 & - & - & 64 & - & 100 \\ - & 80 & - & - & 80 & 100 \end{bmatrix} \end{array}$$

于是根据上面这个 Q 矩阵，我们可得到 Agent 脑海中的优化的状态转移图，如图 6：

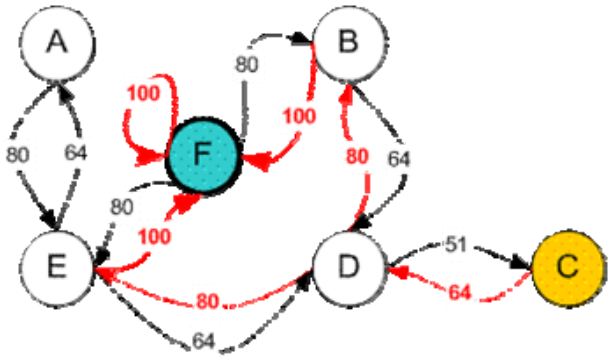


图 6

则给出任意初始状态，我们都可以知道到达目标状态的最优的路径是什么，例如初始状态为 C，则路径为 C->D->B->F。

四、强化学习实验

4.1 实验目的

本次实验为综合性实验，要求基于Q学习算法原理，实验虚拟机器人的强化学习过程，实现具有自主学习能力的智能机器人。

4.2 实验场景

如上所示的机器人走出房间的场景。

4.3 实验步骤

（1）实验场景建模

布置 6 个房间，房间出入关系如 Q 学习算法的示例所示。

（2）初始化 Agent 状态-动作矩阵

初始化状态转移矩阵 R，和目标状态。

（3）Q 学习训练

设置学习系数为 0.8，初始化 Q 矩阵为全 0 矩阵。根据状态转移矩阵 R，随机地选择一个状态，应用 Q 学习算法，开始 Q 学习。

（4）学习规则应用

克隆生成一个新的 Agent，给出任意初始状态，检验其是否具有到达目标状态的最优的路径能力。例如初始状态为 C，则路径为 C->D->B->F。

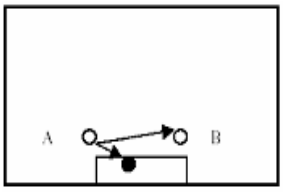
4.4 比较实验

监督学习。虚拟机器人每出现在一个房间里，由 Avatar 主人作为老师给出最优路径值，将所有的状态学习完，即可开始应用规则。请思考比较监督学习和强化学习的本质区别。

选做实验：体验强化学习在机器人领域的应用

仿真机器人足球

应用 Q 学习算法进行仿真机器人足球 2 对 1 训练，训练的目的是试图使智能体学习获得到一种战略上的意识，能够在进攻中进行配合，参见文献宋志伟（2003）。



前锋 A 控球，并且在可射门的区域内，但是 A 已经没有射门角度了；队友 B 也处于射门区域，并且 B 具有良好的射门角度。A 传球给 B，射门由 B 来完成，那么这次进攻配合就会很成功。通过 Q 学习的方法来进行 2 对 1 的射门训练，让 A 掌握在这种状态情况下传球给 B 的动作是最优的策略；智能体通过大量的学习训练（大数量级的状态量和重复相同状态）来获得策略，因此更具有适应性 。

状态描述。将进攻禁区划分为个小区域，每个小区域是边长为 2m 的正方形，一个二维数组()便可描述这个区域。使用三个 Agent 的位置来描述 2 对 1 进攻时的环境状态，利用图 10.11 所示的划分来泛化状态。可认为智能体位于同一战略区域为相似状态，这样对状态的描述虽然不精确，但设计所需的是一种战略层次的描述，可认为 Agent 在战略区域内是积极跑动的，这种方法满足了需求。如此，便描述了一个特定的状态；其中，是进攻队员 A 的区域编号，是进攻队员 B 的区域编号，是守门员的区域编号。区域编号计算公式为：。相应的，所保存的状态值为三个区域编号组成的对。前锋 A 控球，并且在可射门的区域内，但是 A 已经没有射门角度了；队友 B 也处于射门区域，并且 B 具有良好的射门角度。A 传球给 B，射门由 B 来完成，那么这次进攻配合就会很成功。通过 Q 学习的方法来进行 2 对 1 的射门训练，让 A 掌握在这种状态情况下传球给 B 的动作是最优的策略；智能体通过大量的学习训练（大数量级的状态量和重复相同状态）来获得策略，因此更具有适应性 。

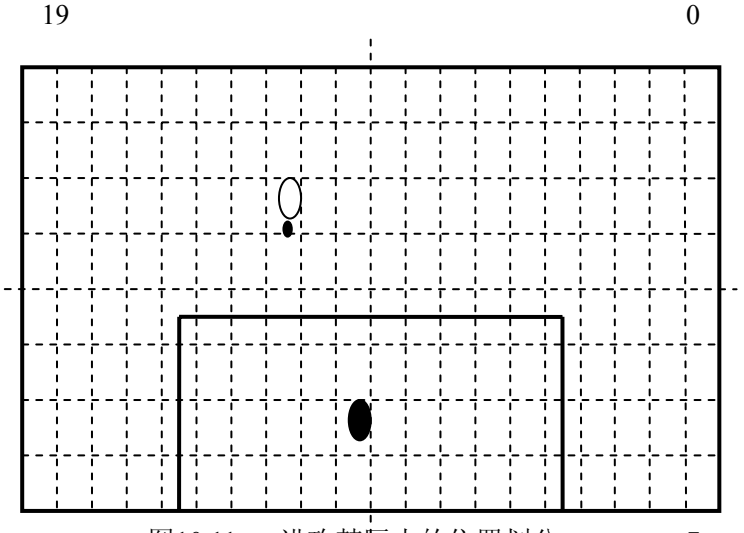


图10.11　进攻禁区内的位置划分　　7

可选动作集：射门，传球，带球

- 射门的策略通过基于概率的射门训练的学习来得到。
- 带球的策略是，始终向受到威胁小，并且射门成功率高的区域带球。为了实现这一策略目标，可划分进攻区域为多个战略区，在每个战略区进行射门评价，记录每个区域的射门成功率。
- 传球 Pass 策略很简单，只需在两个 Agent 间进行传球，即不需要选择球传送的对象，也不需要判断传球路径。如果传球失败的话，则认为在这种状态下执行 Pass 策略是不成功的；经过此训练后，不可能的传球路径也不会被执行了。

奖励状态：四个。假设进攻方在左半场，按照标准的 Soccer server 规范，这四个状态的比赛模式为 play_on、goal_left、goal_kick_right 和 free_kick_right。当达到奖励状态时，给与智能体最终奖励 r。促使到达奖励状态的上一步动作获得的立即回报值为最终奖励值 r，其他未直接促使到达奖励状态的动作均获得过程奖励值作为立即奖励；其中 **goal_left 的 r 最大为 1 表示进球，其他状态下 r 为不同大小的负数**。

学习机制：智能体在经过一定的状态和执行多个动作后获得了终态奖励（到达了奖励状态），这时就会对这个状态-动作序列分配奖励。Q 学习算法的核心就是每一个状态和动作的组合都拥有一个 Q 值，每次获得最终回报后通过更新等式更新这个 Q 值。 确定 Q 更新等式为：

其

$$Q\left(s,a\right)=\left(1-\alpha\right)Q\left(s,a\right)+\alpha\left(r+\gamma\max Q\left(s_{t+1},a_{t+1}\right)\right)$$

中：a =0.1，学习系数=0.95。

学习训练：在实际的训练中，初始 Q 表各项的值为 1。经过大约 2 万次的训练（到达奖励状态为一次），Agent 的 Q 表中的绝大部分项发生了变化，并且已经区分开来。下表是某场景下的训练次数和动作选择的变化示意表。

	初始值	5千次	1万次	2万次
shoot	1	0.7342	0.6248	0.5311
pass	1	0.9743	0.9851	0.993
dribble	1	0.9012	0.8104	0.7242

《宋志伟, 陈小平, 2003. 仿真机器人足球中的强化学习. 《机器人》, 24(7S):761-766. 》