BUSS 3620.人工智能导论

强化学习

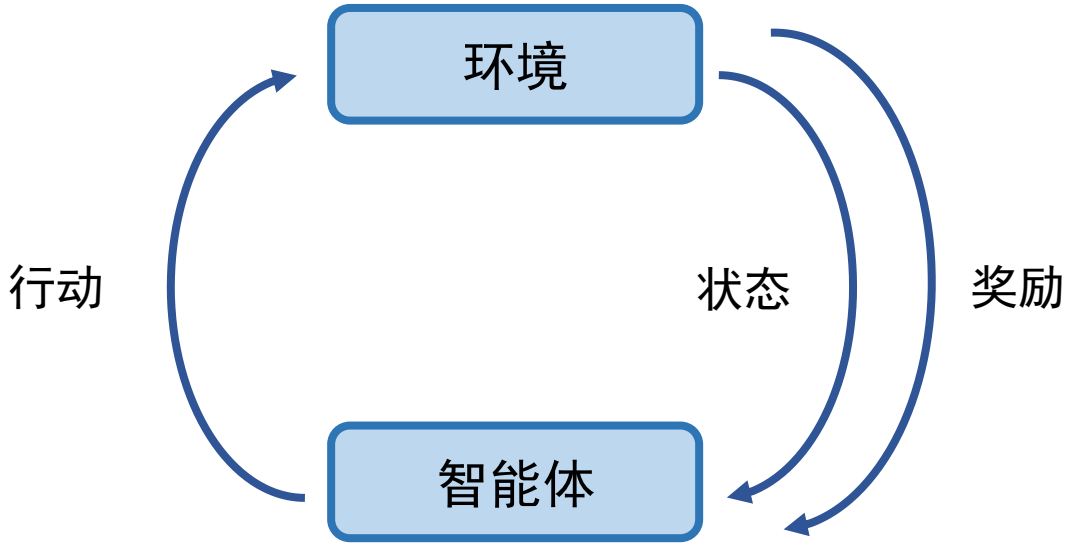
刘佳璐

安泰经济与管理学院

上海交通大学

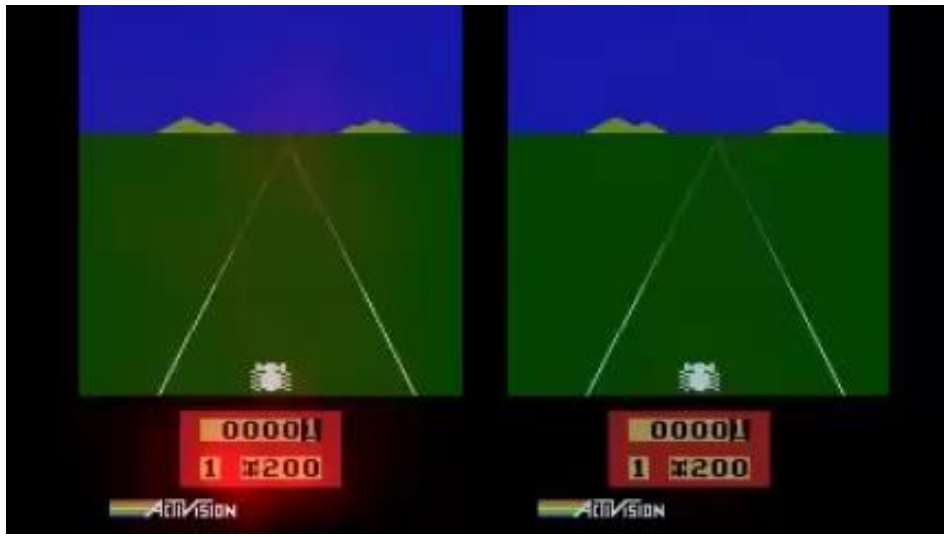
强化学习 Reinforcement Learning

- 通过获得的奖励(rewards)和惩罚(punishment)，来学习未来应该执行什么行动
  - 没有输入输出对 (input-output pair)
  - 从经验中学习
  - 最大化预期奖励



强化学习的一些进展

- 2013: 玩小游戏



Enduro

Sequest

来源: Mnih, V., Kavukcuoglu, K., Silver, D. *et al.* Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015). https://doi.org/10.1038/nature14236

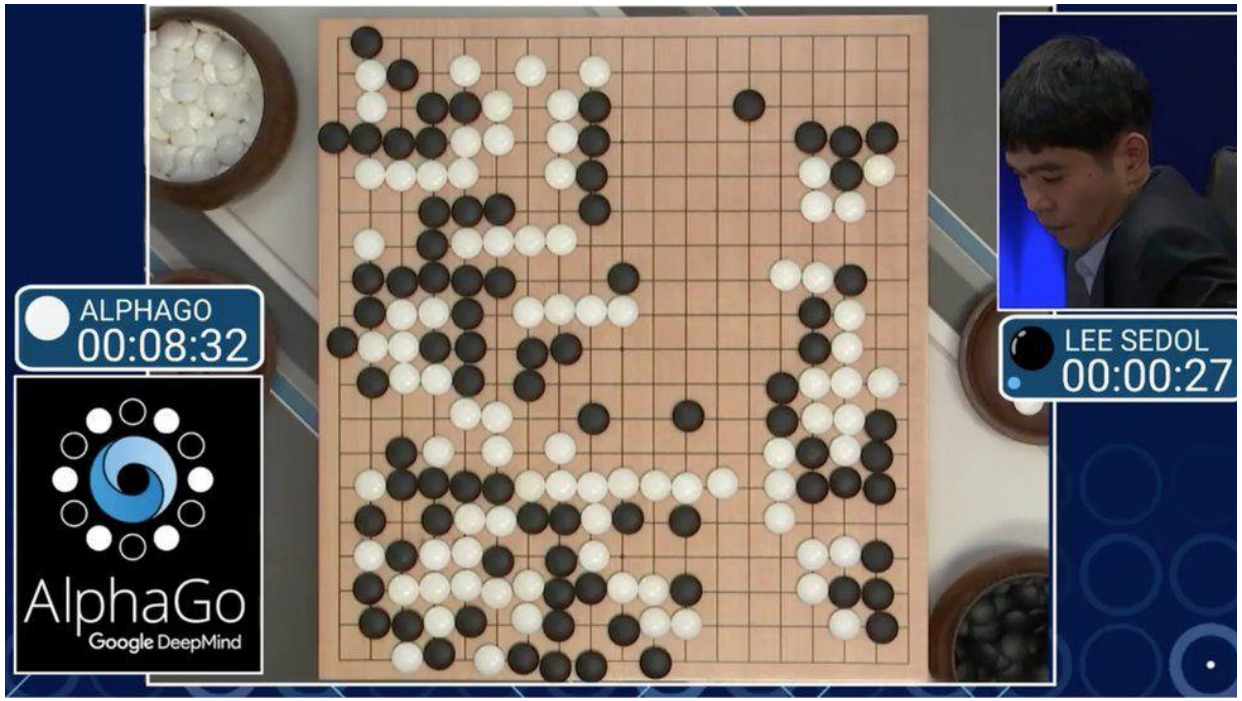
强化学习的一些进展

- 2015: 运动

来源: Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015, June). Trust region policy optimization. In *International conference on machine learning* (pp. 1889-1897). PMLR.

强化学习的一些进展

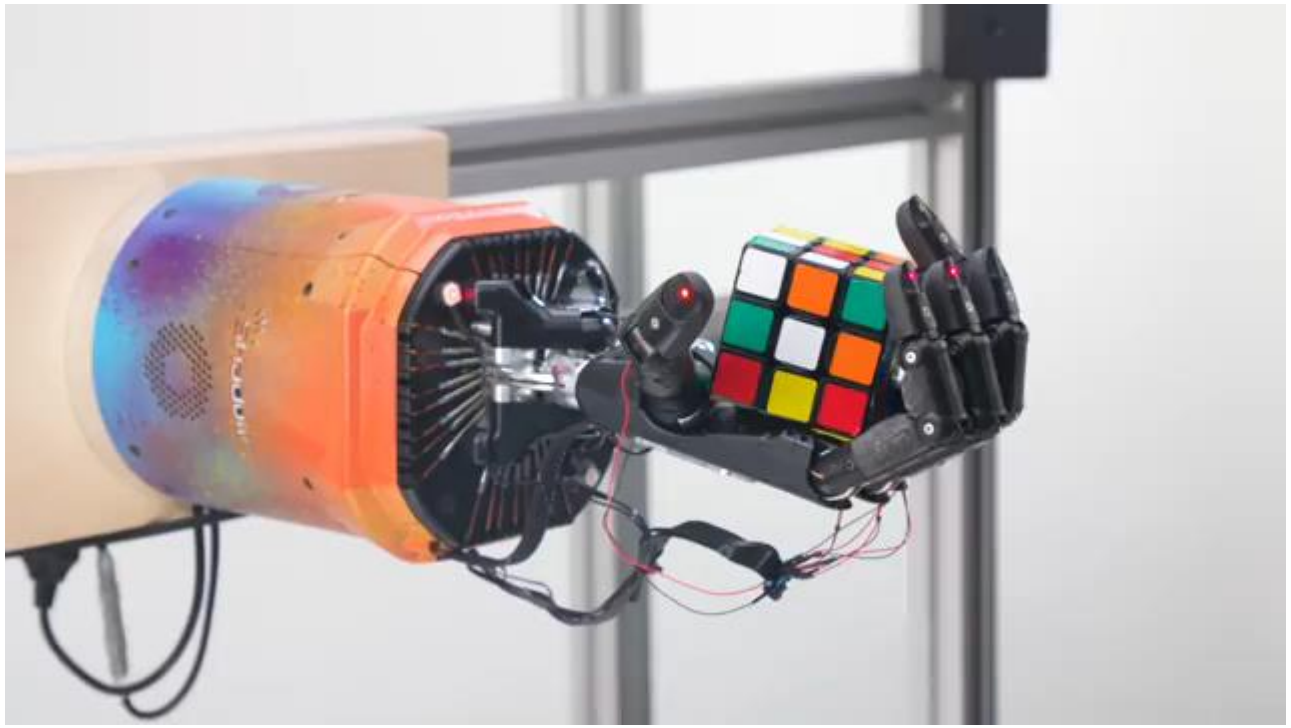
- 2016: 围棋



来源: Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *nature*, *529*(7587), 484-489.

强化学习的一些进展

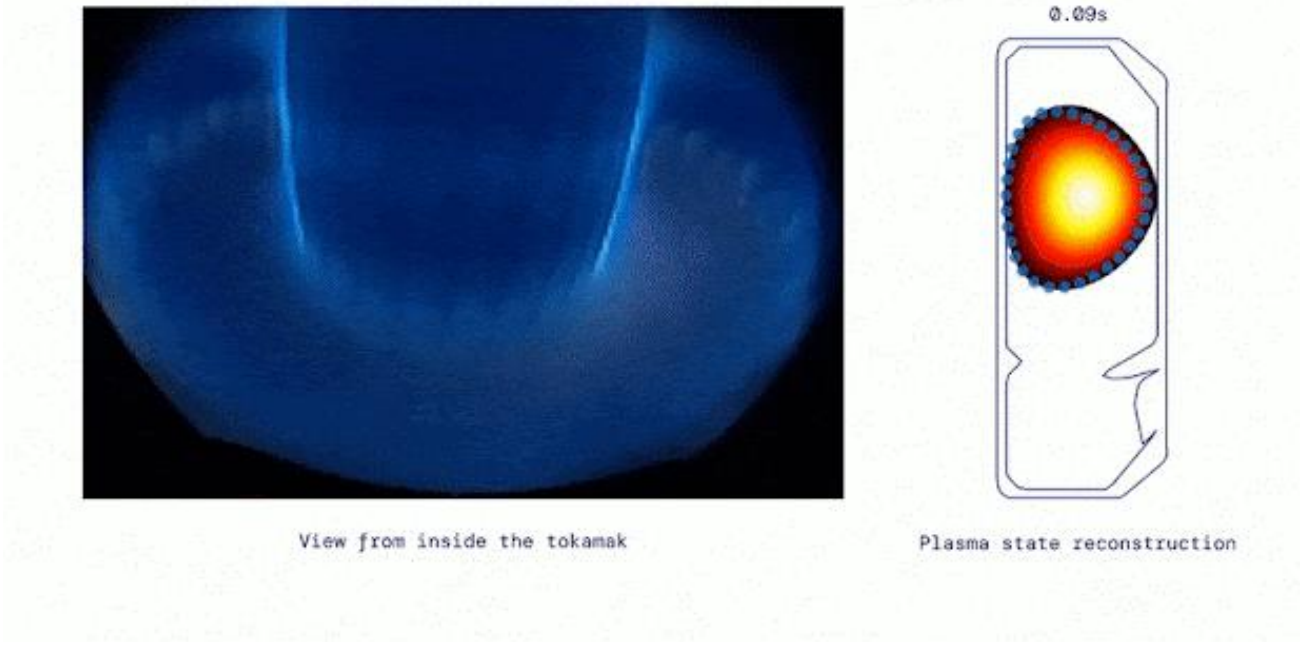
- 2019: 机器人



来源: Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., ... & Zhang, L. (2019). Solving rubik's cube with a robot hand. OPENAI.

强化学习的一些进展

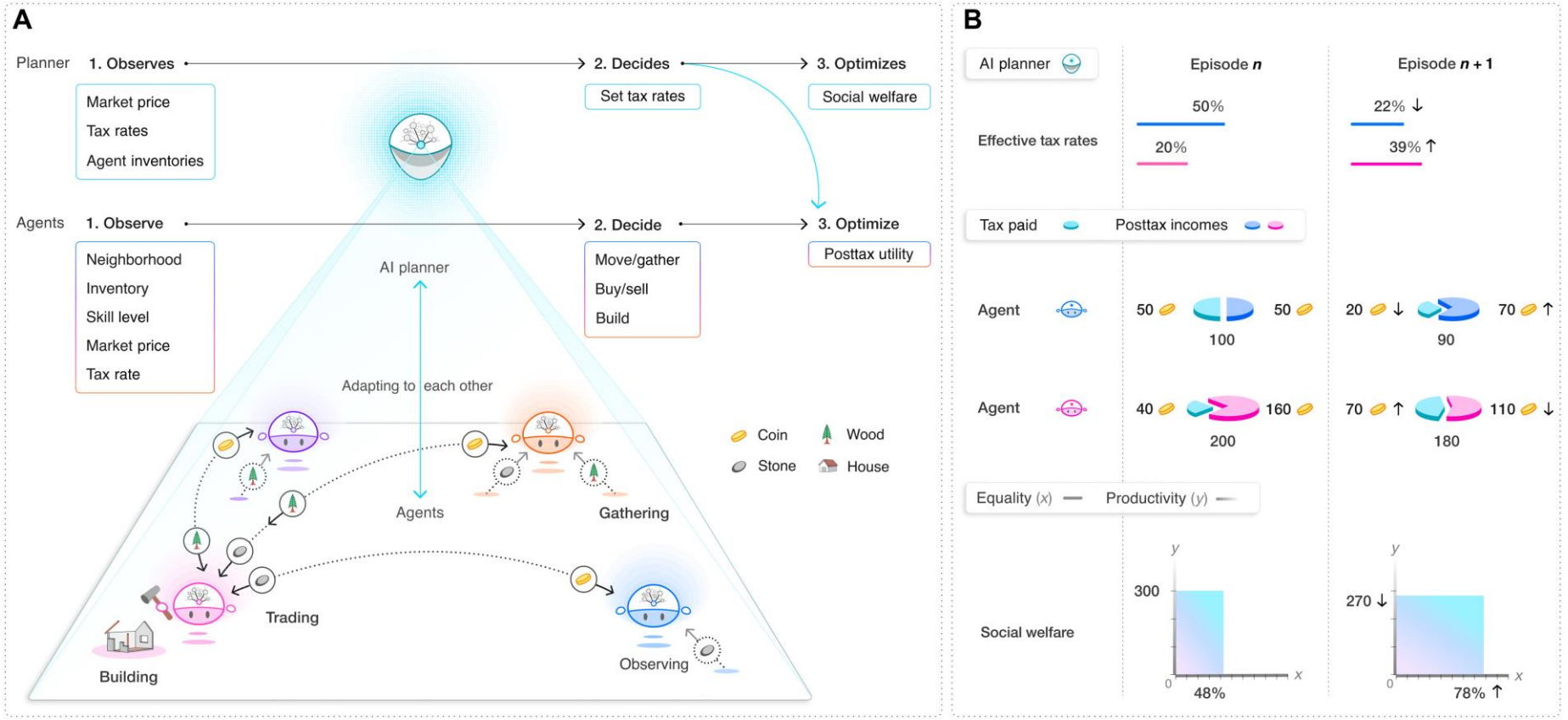
- 2022：核聚变离子气体控制



来源：Degrave, J., Felici, F., Buchli, J. *et al.* Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature* **602**, 414–419 (2022). https://deepmind.google/discover/blog/accelerating-fusion-science-through-learned-plasma-control/

强化学习的一些进展

- 2022：经济政策



来源：Stephan Zheng *et al.*, The AI Economist: Taxation policy design via two-level deep multiagent reinforcement learning. *Science Advance* .**8**, eabk2607(2022).

BUSS 3620.人工智能导论

#1. 马尔可夫决策过程

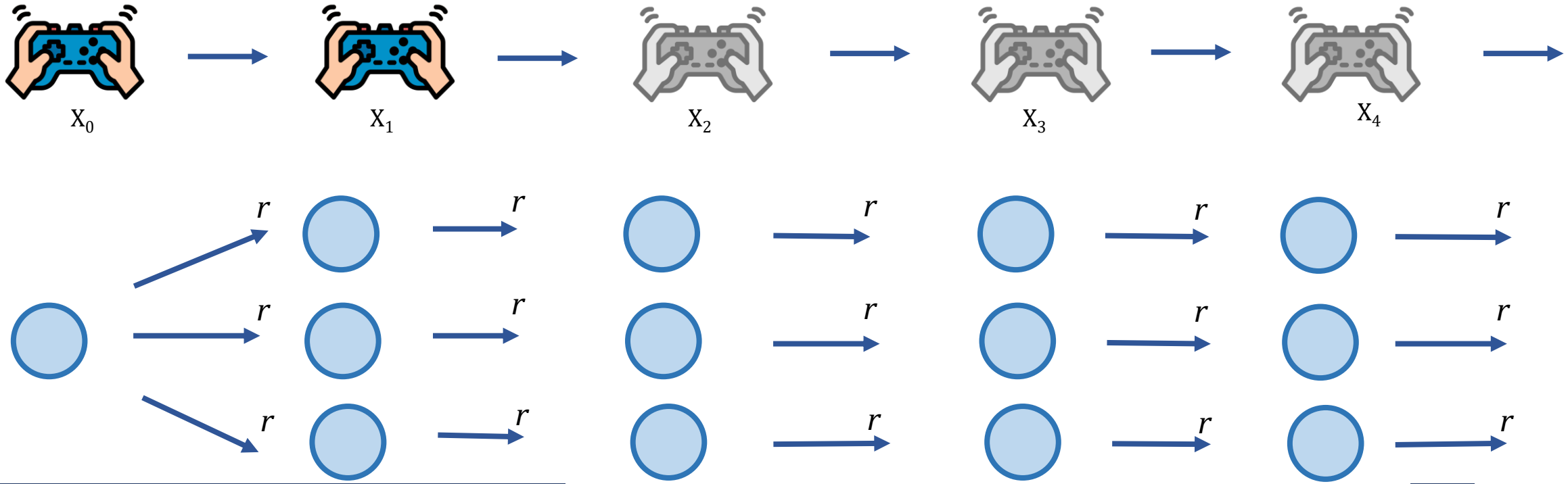
刘佳璐

安泰经济与管理学院

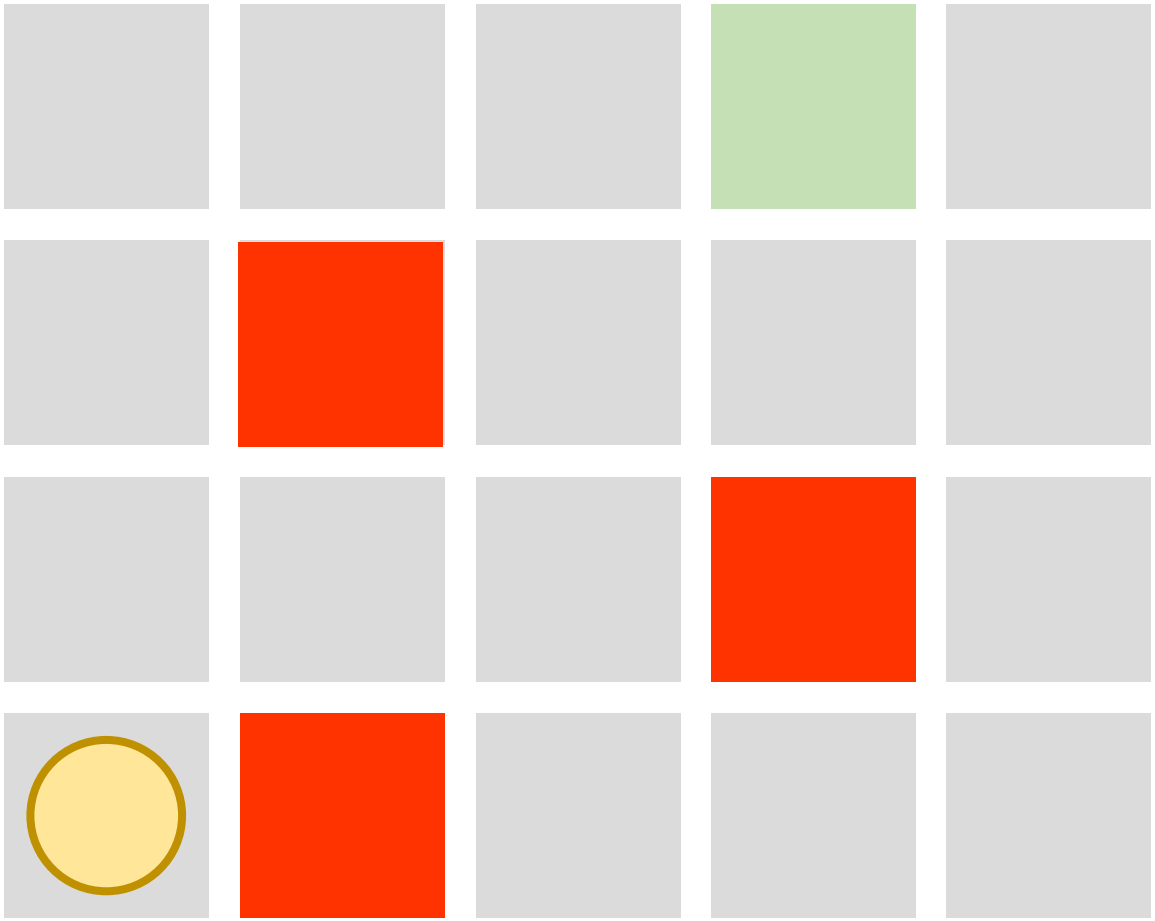
上海交通大学

马尔可夫决策过程 Markov Decision Process

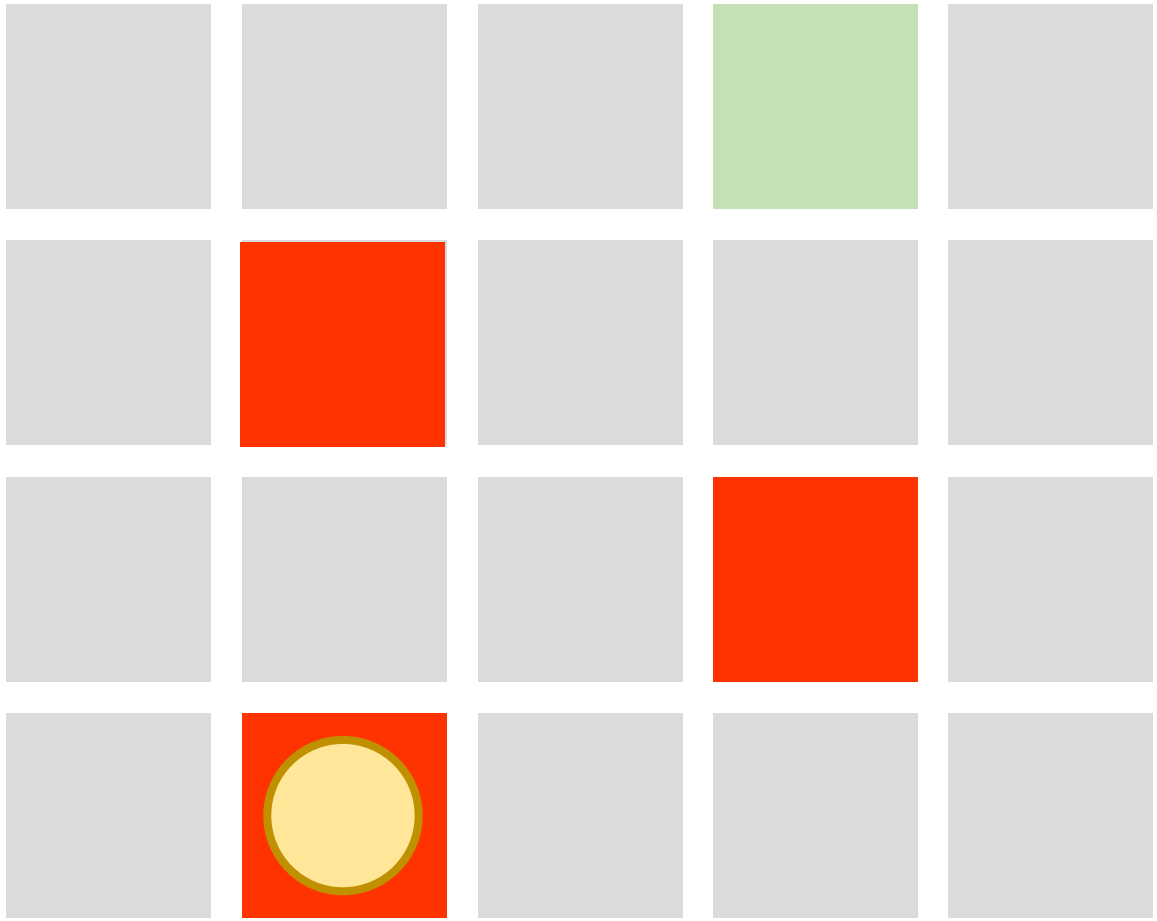
- 模拟智能体的决策，涉及到状态(states)、行动(actions)、和奖励(rewards)



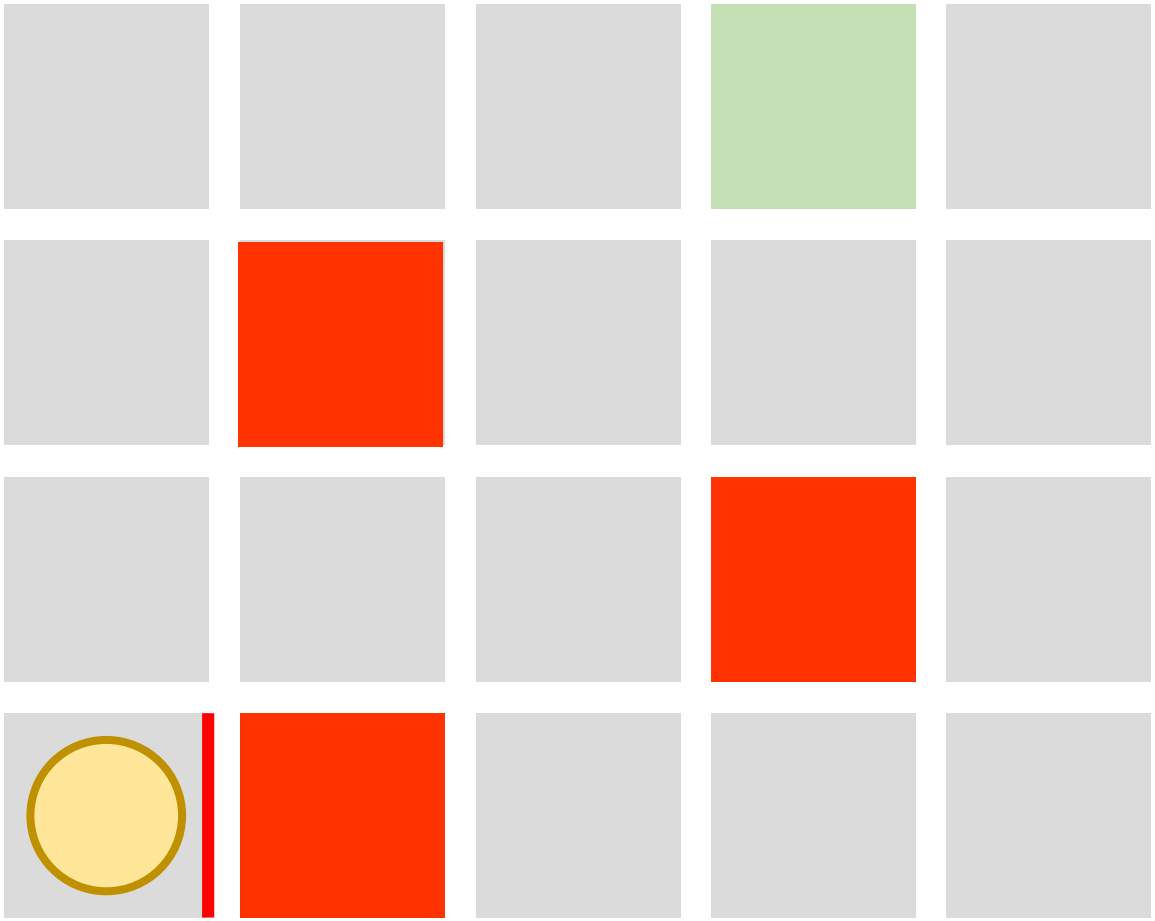
马尔可夫决策过程 Markov Decision Process



马尔可夫决策过程 Markov Decision Process

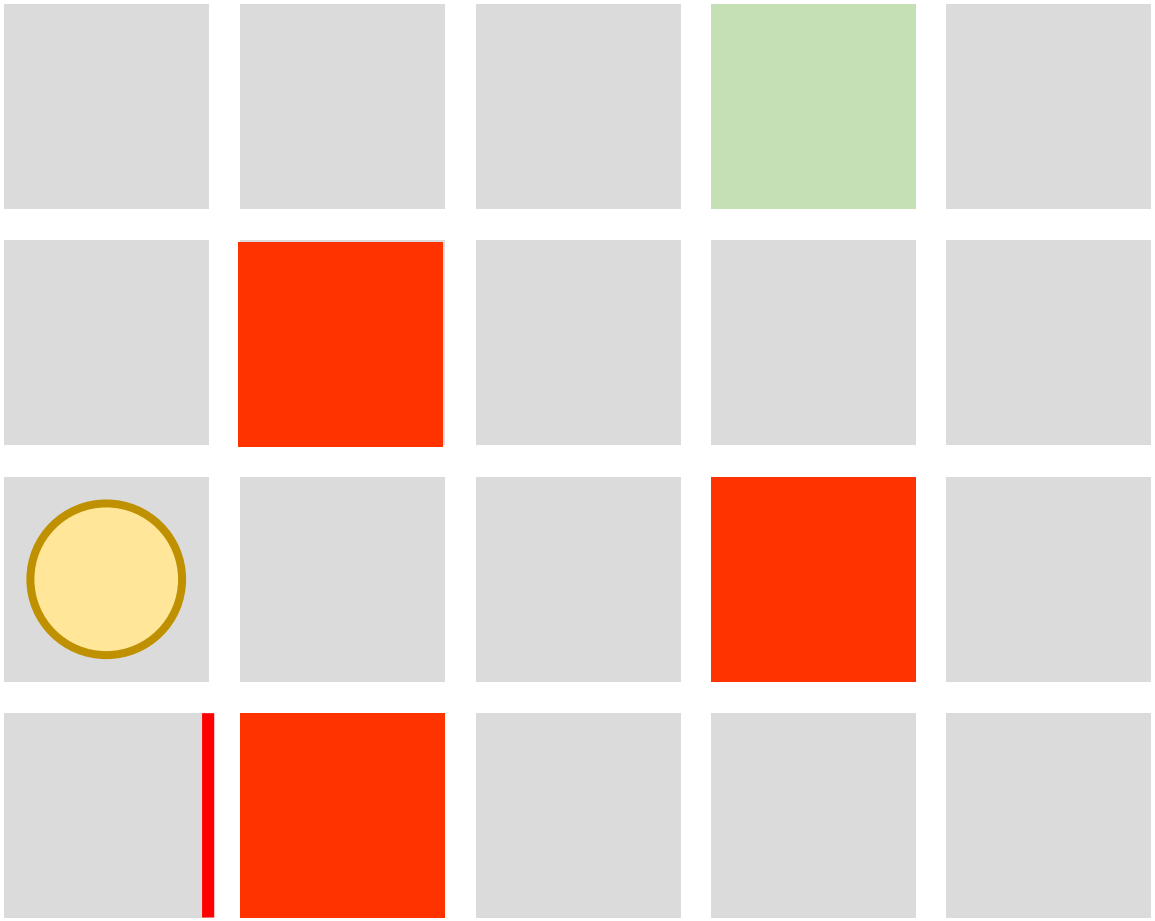


马尔可夫决策过程 Markov Decision Process



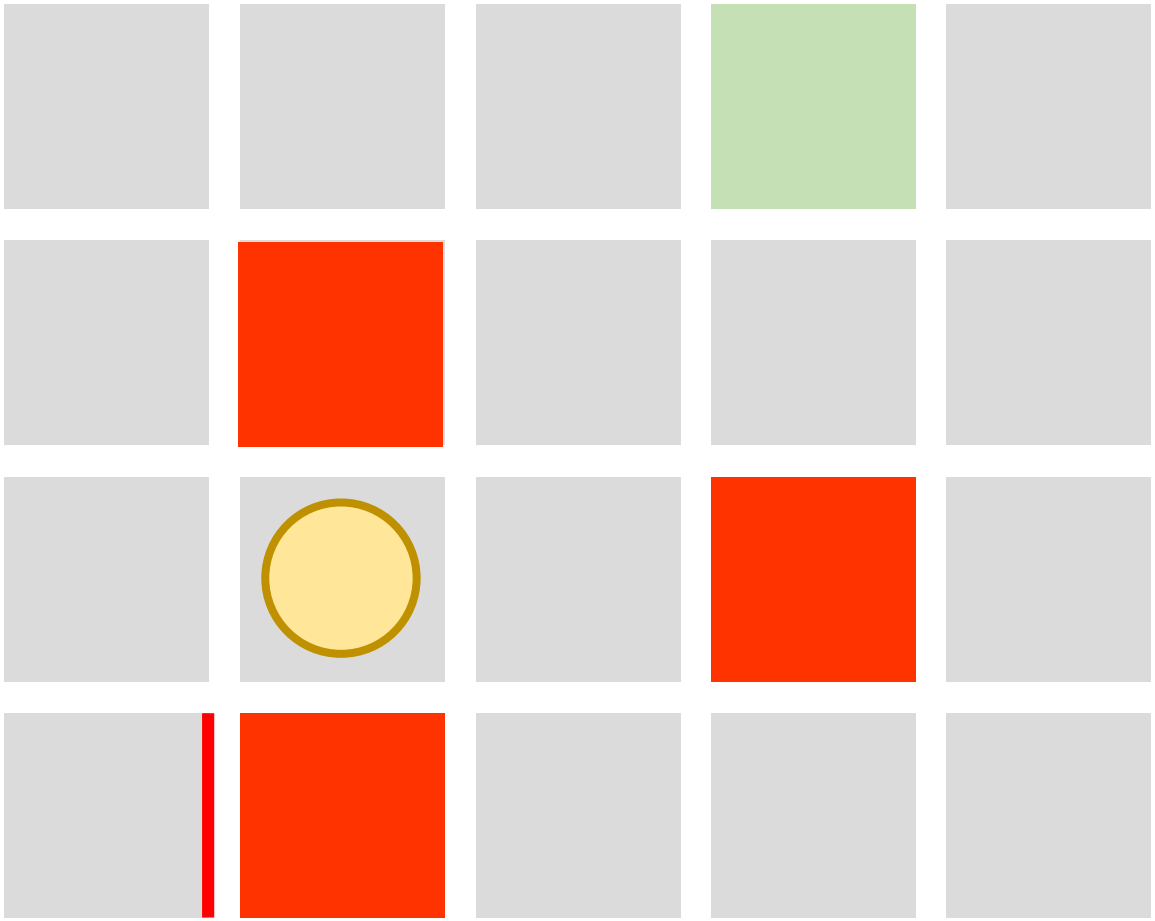
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



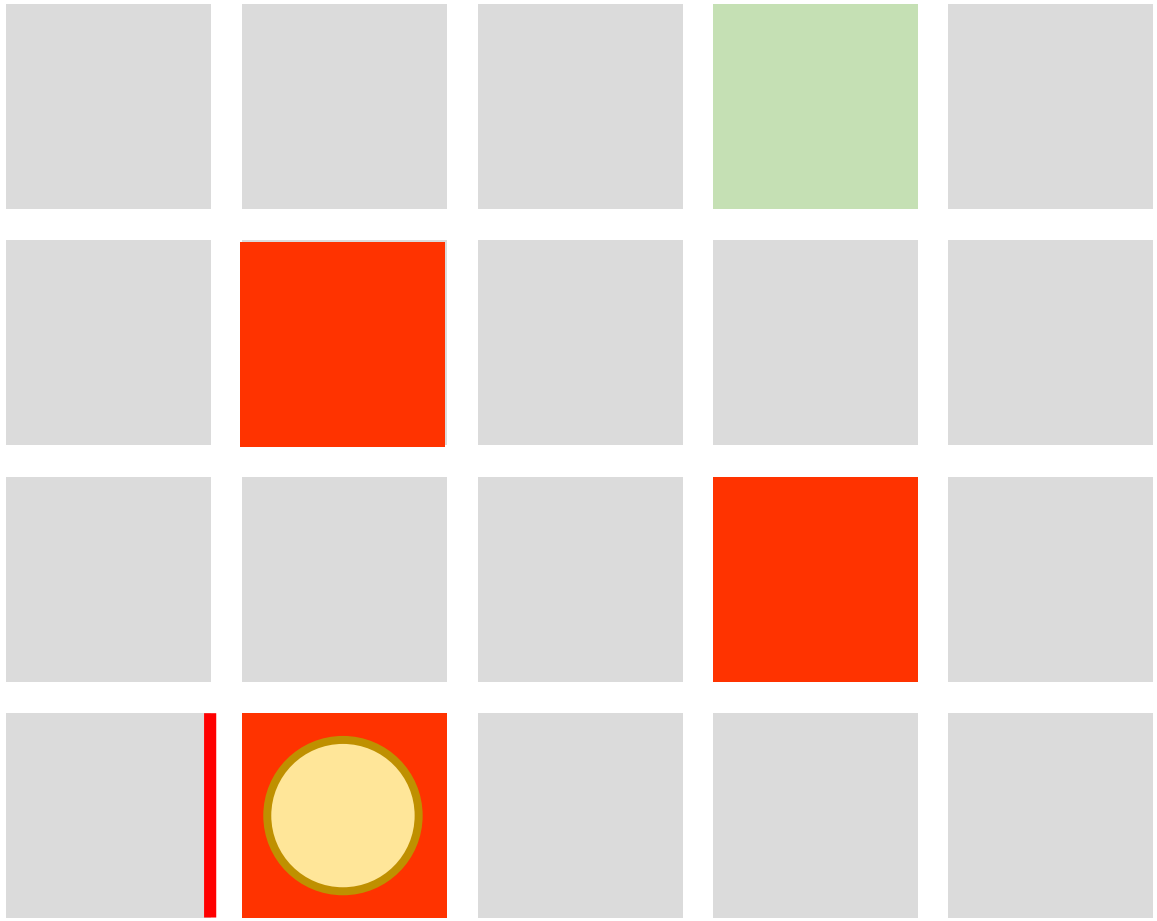
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



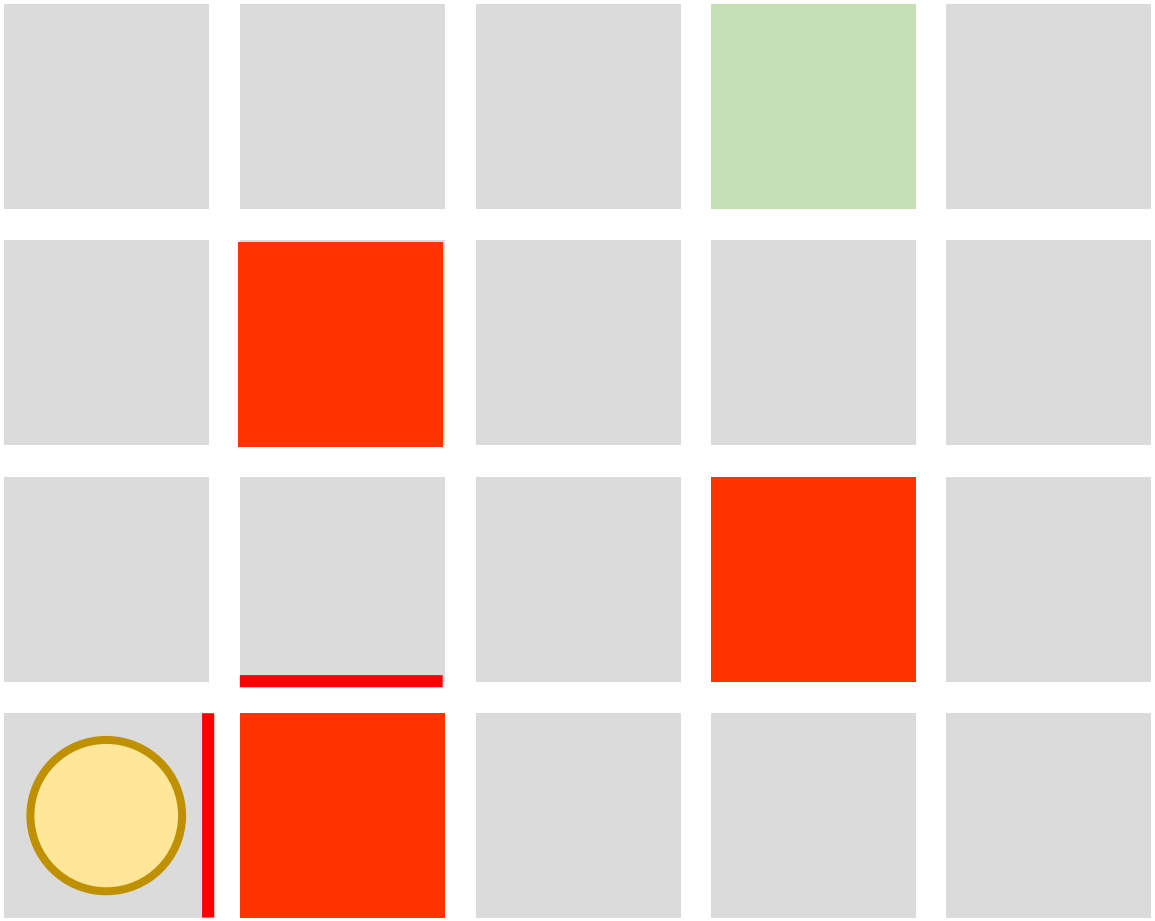
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



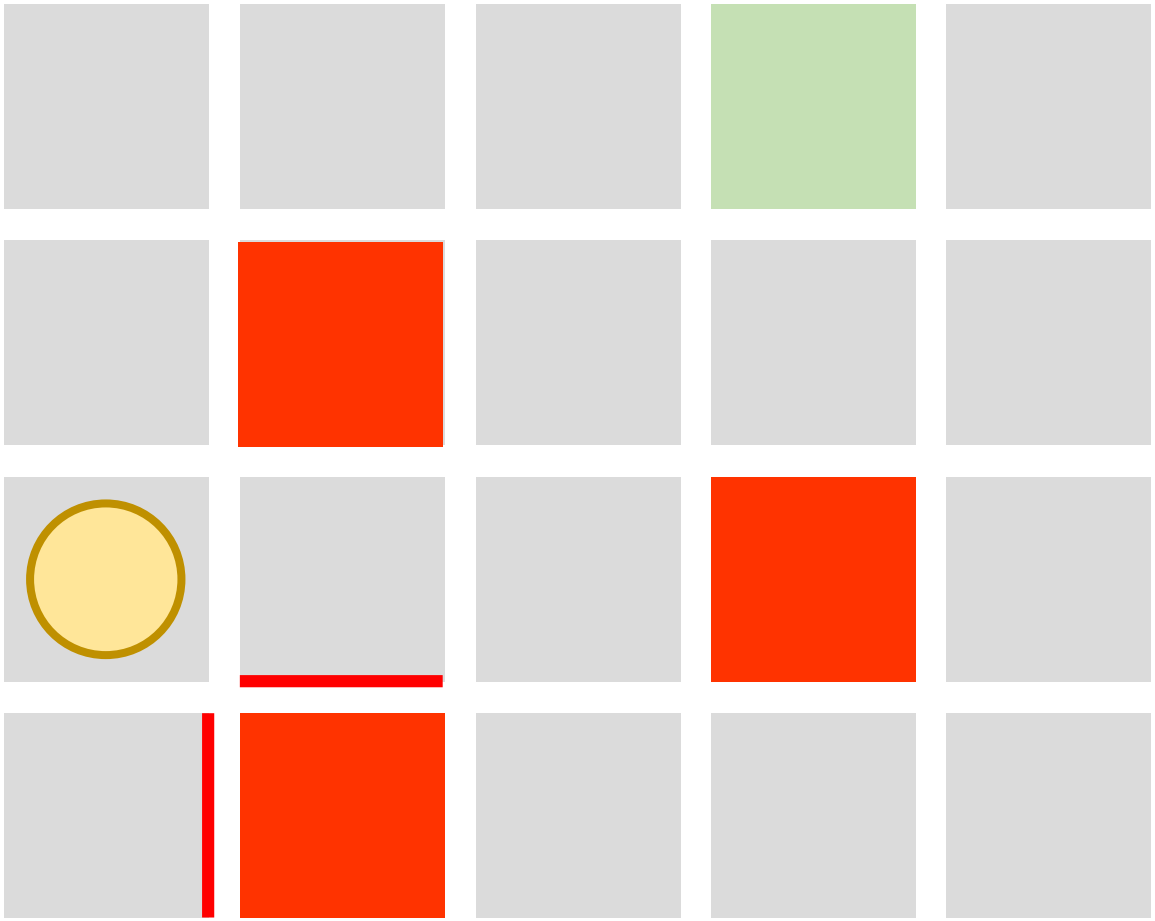
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



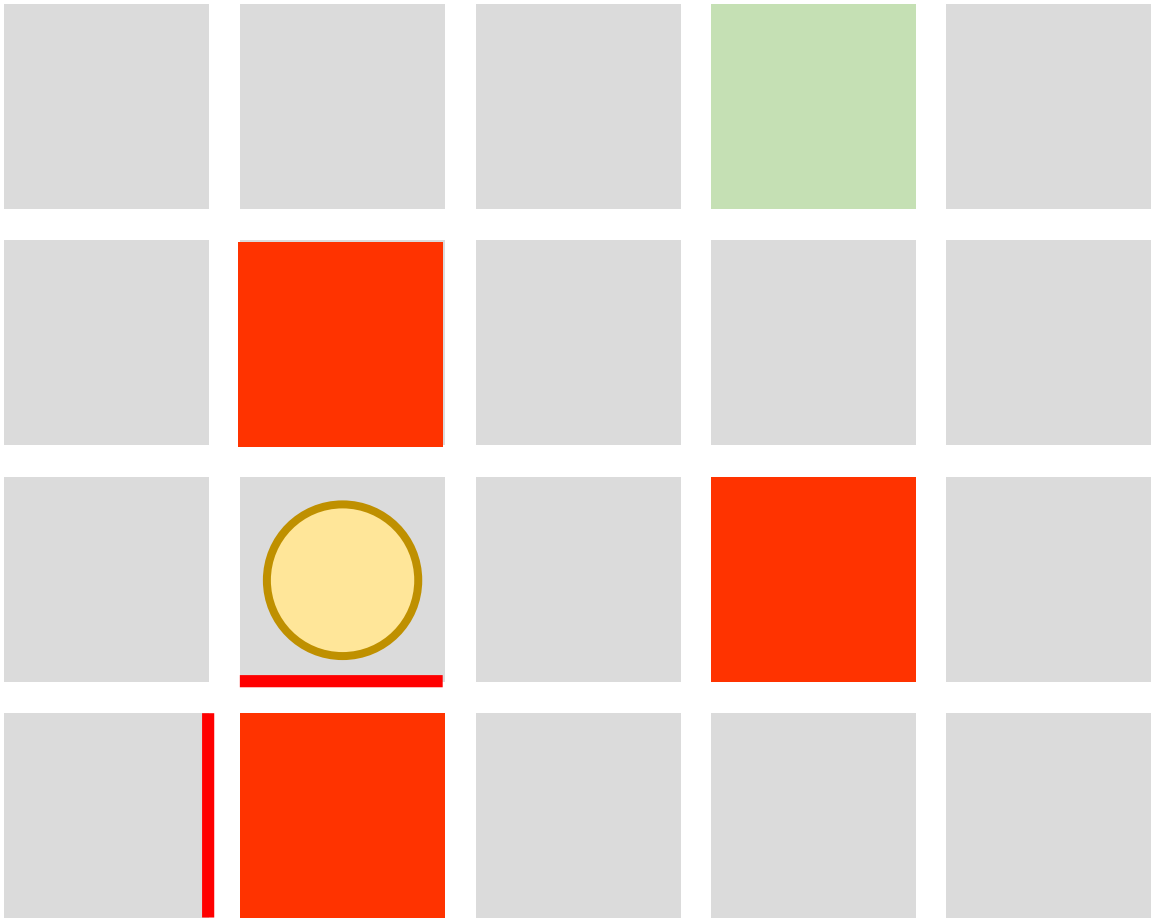
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



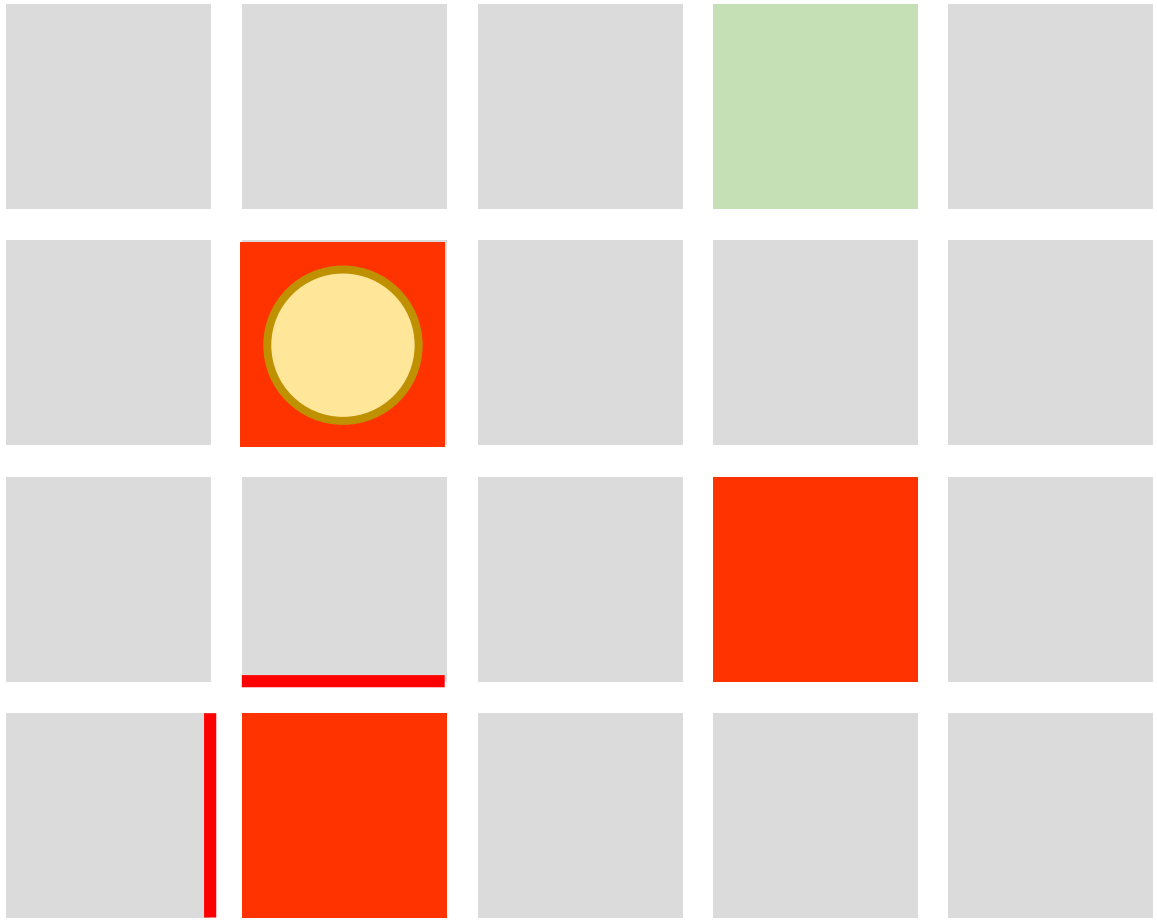
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



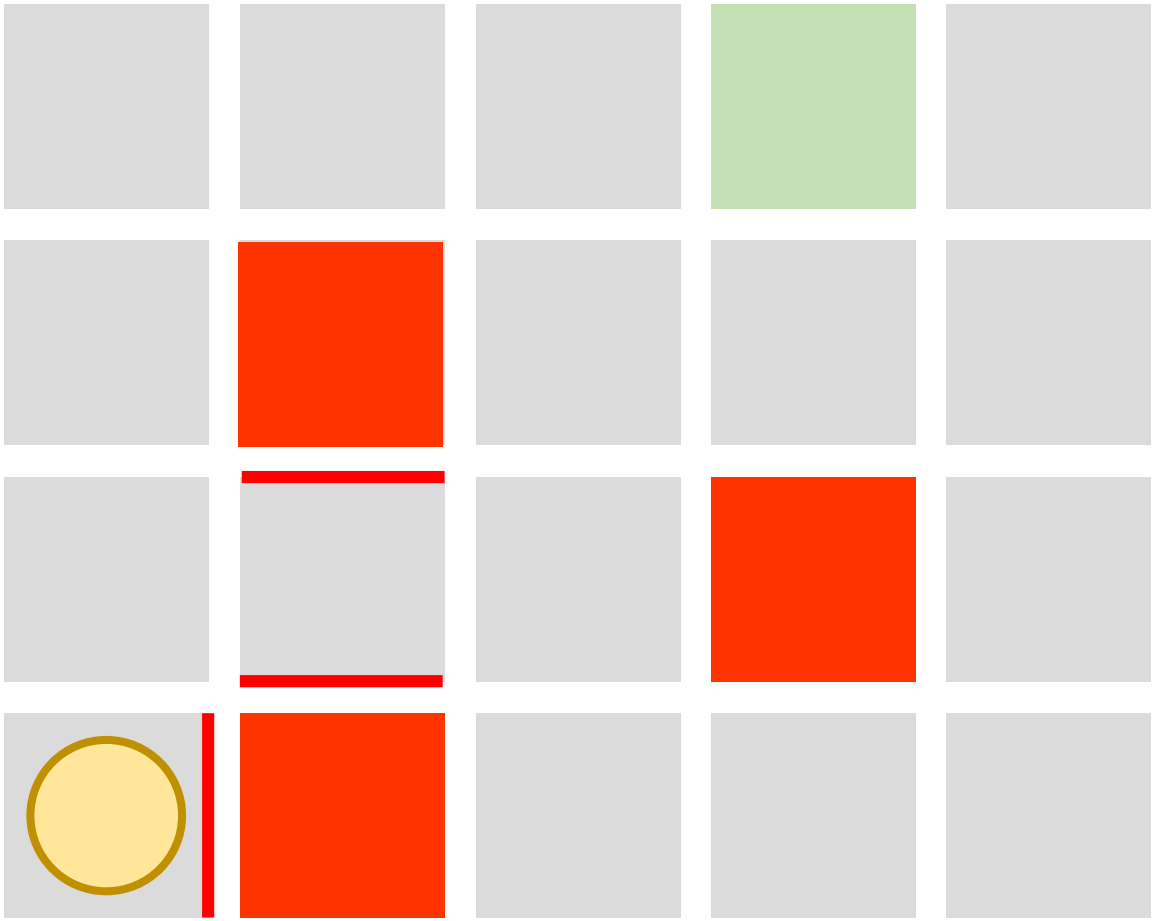
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



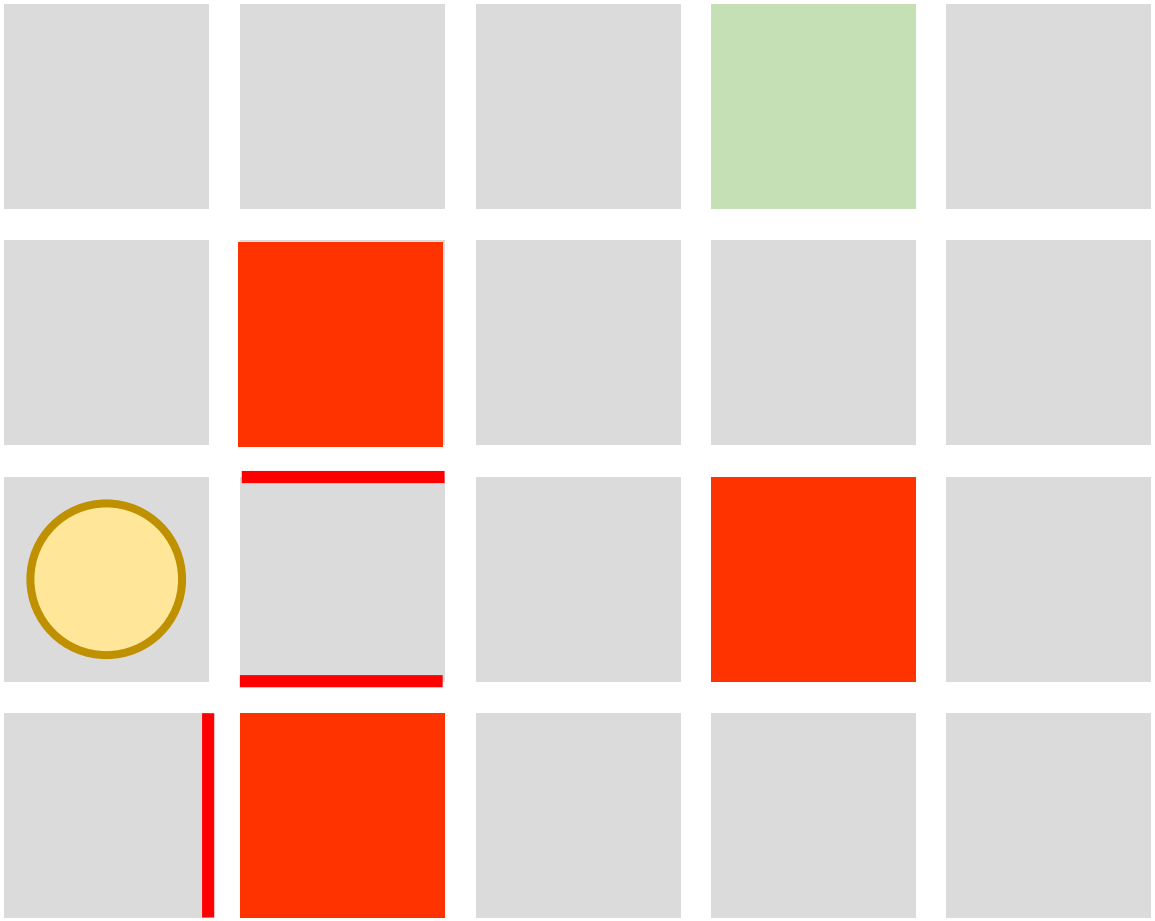
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



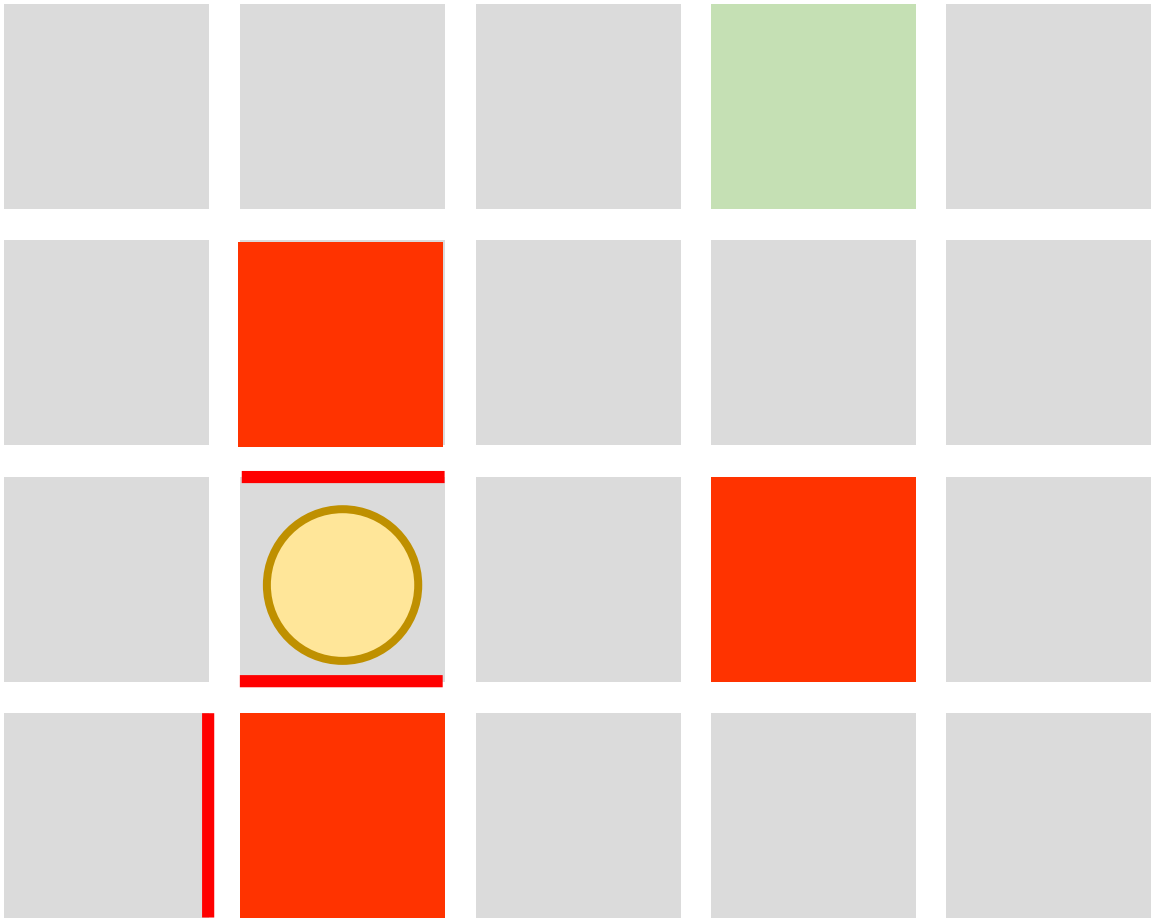
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



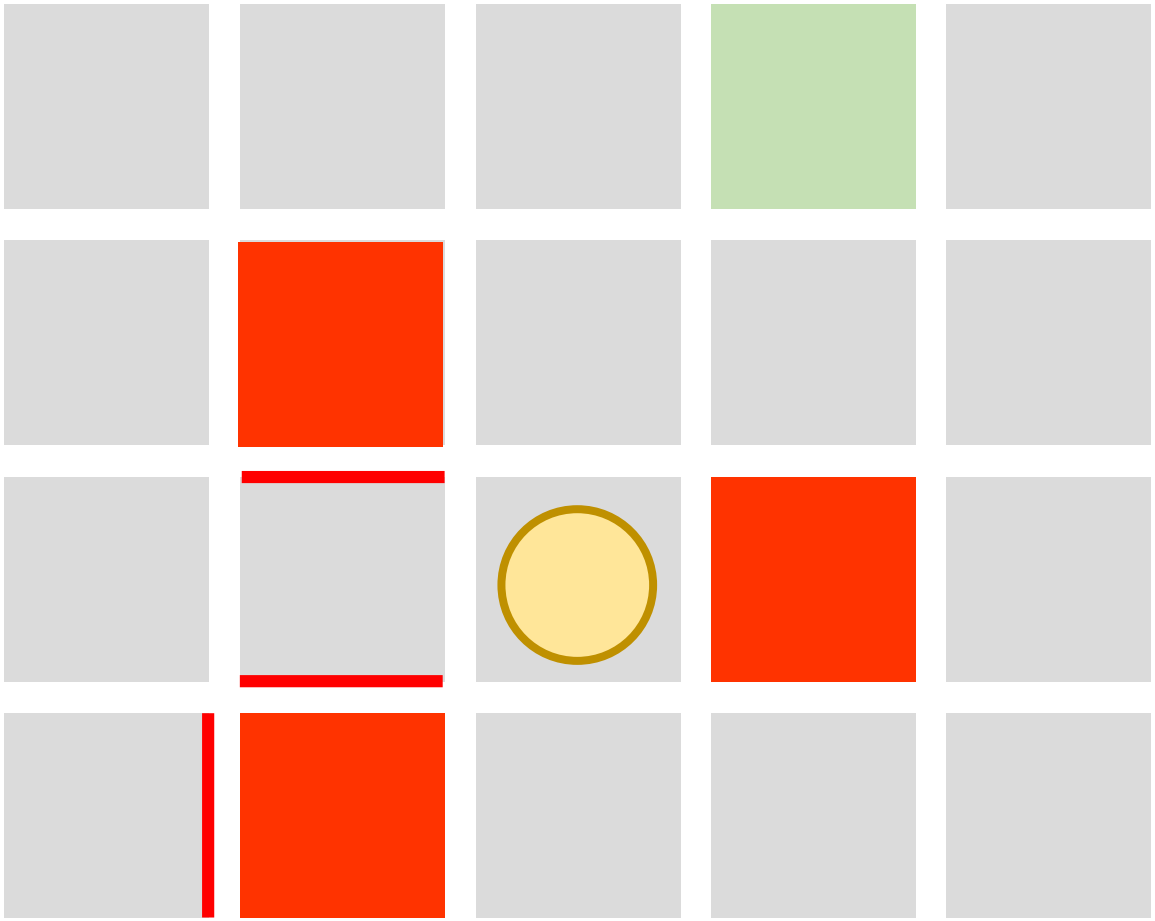
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



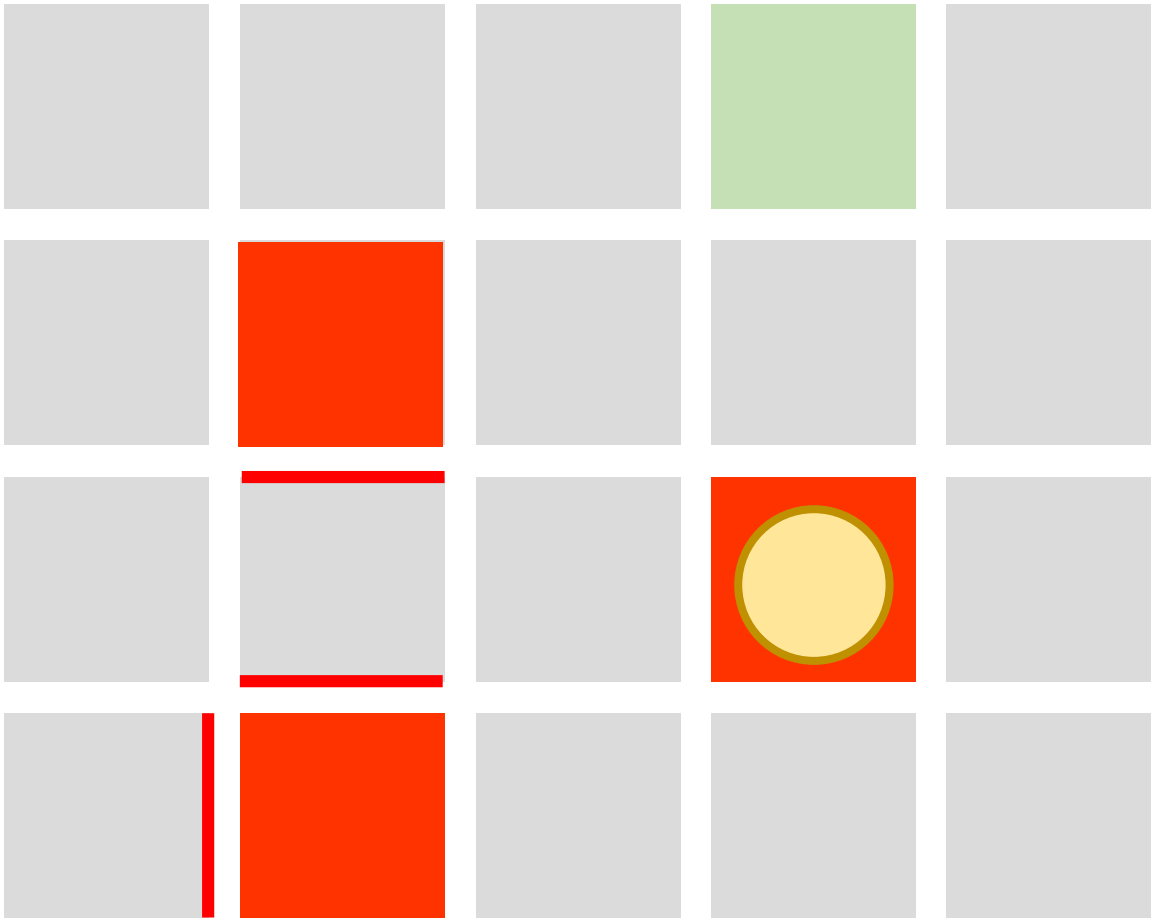
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



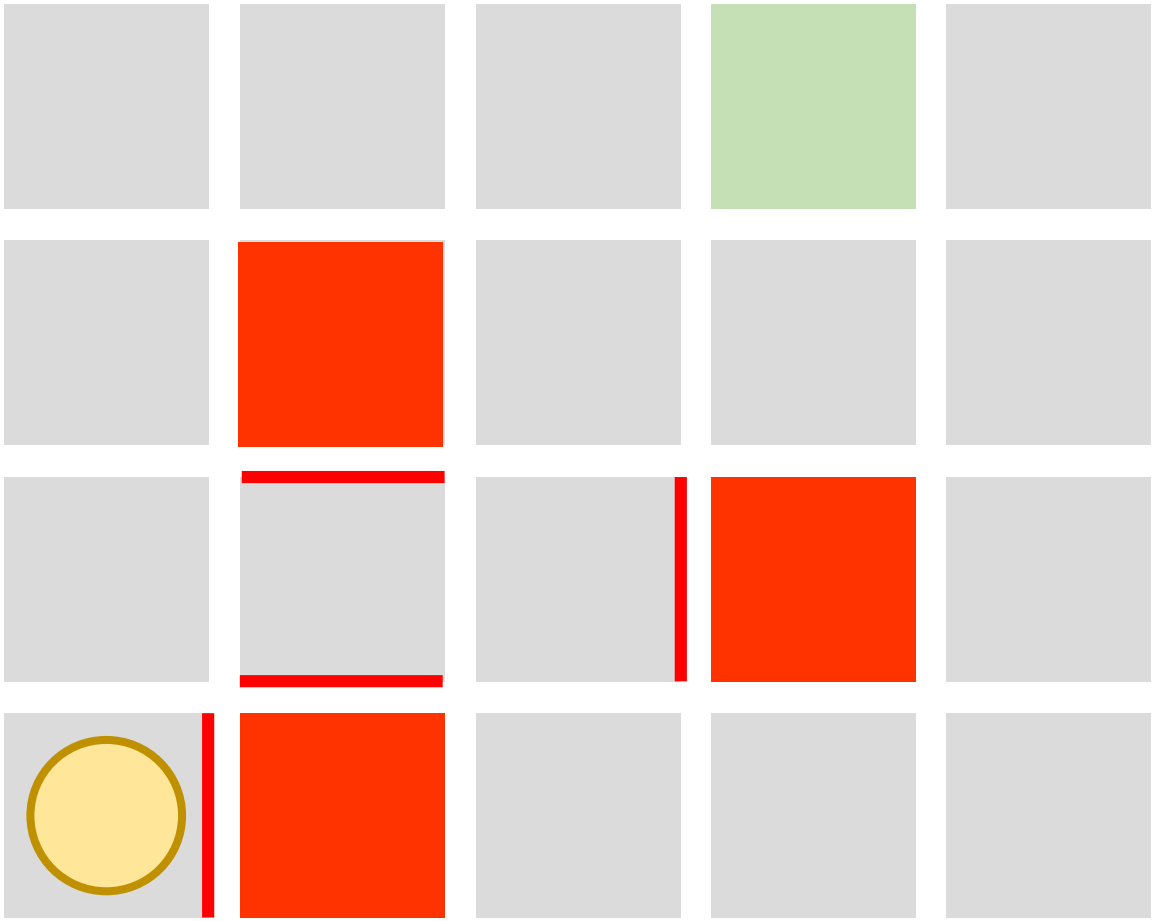
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



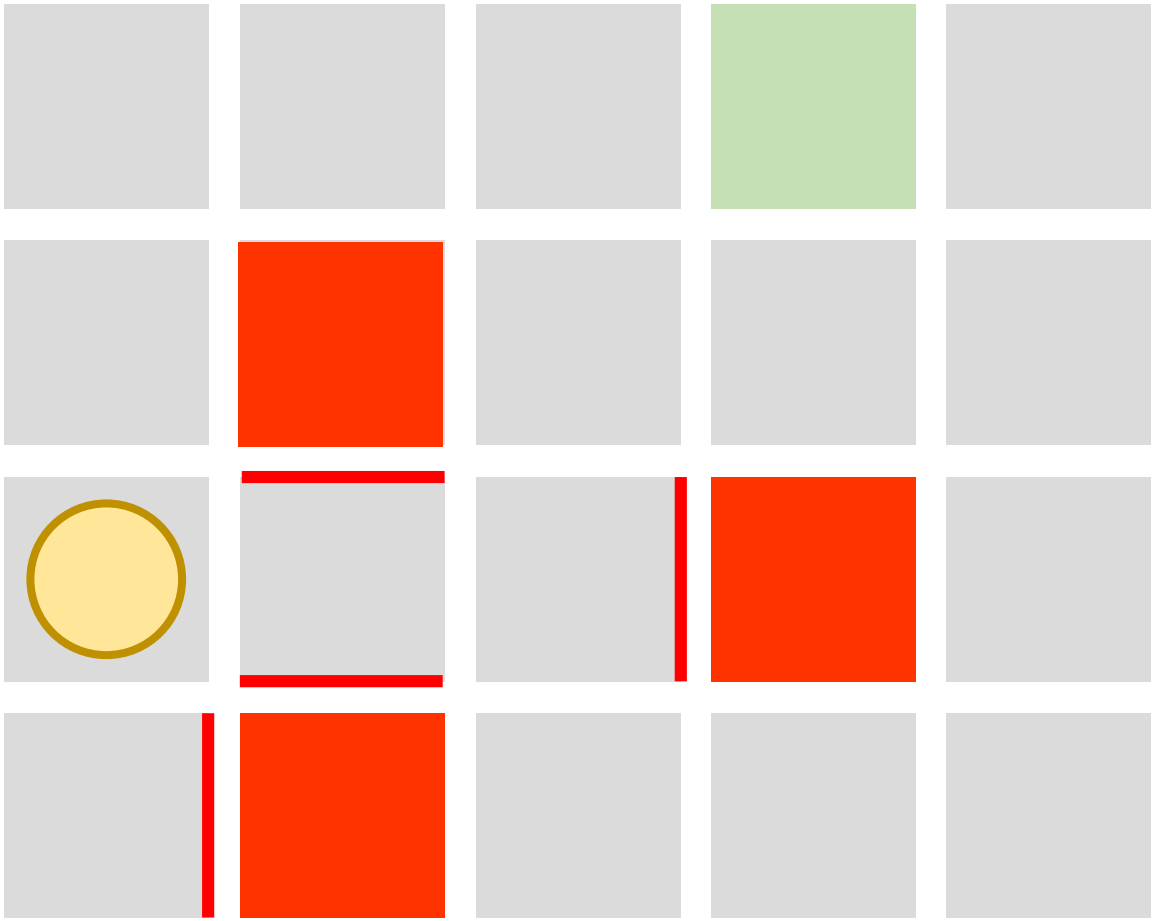
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



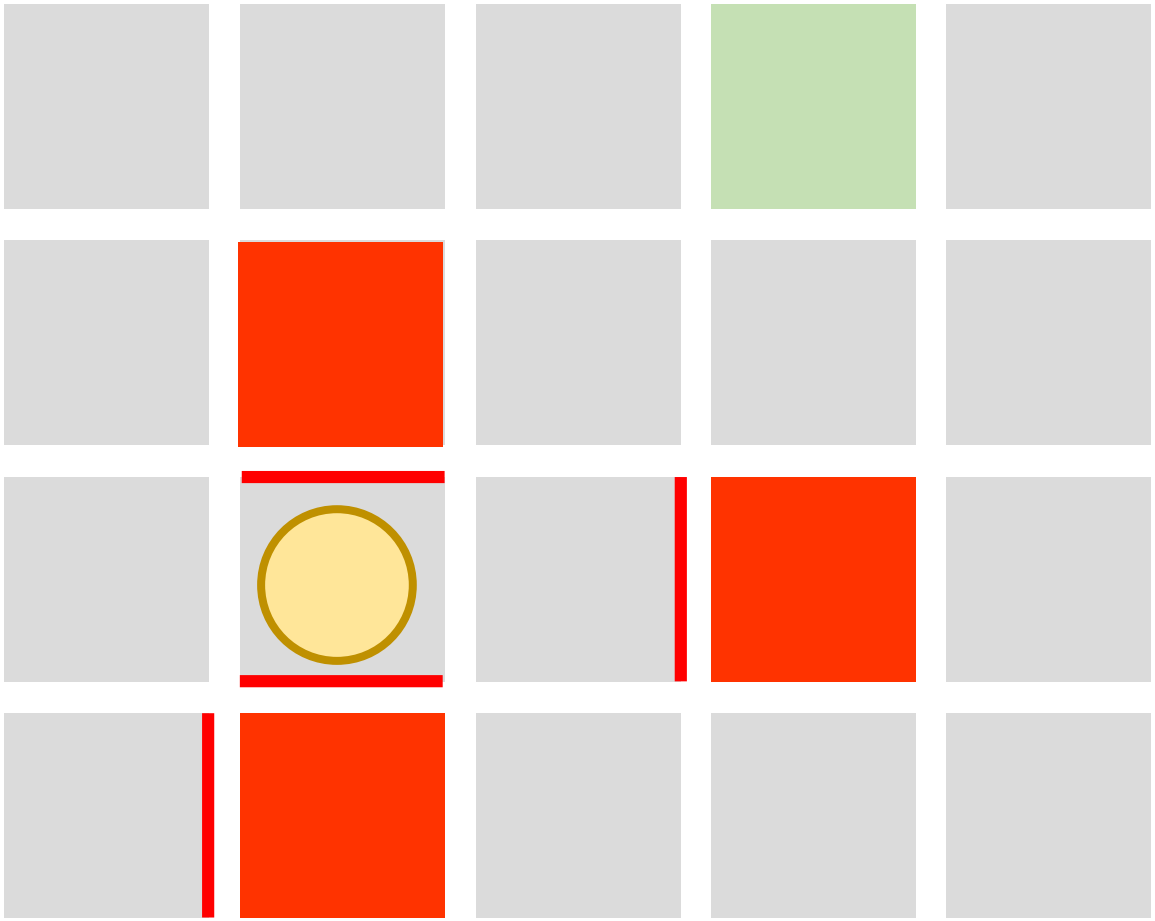
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



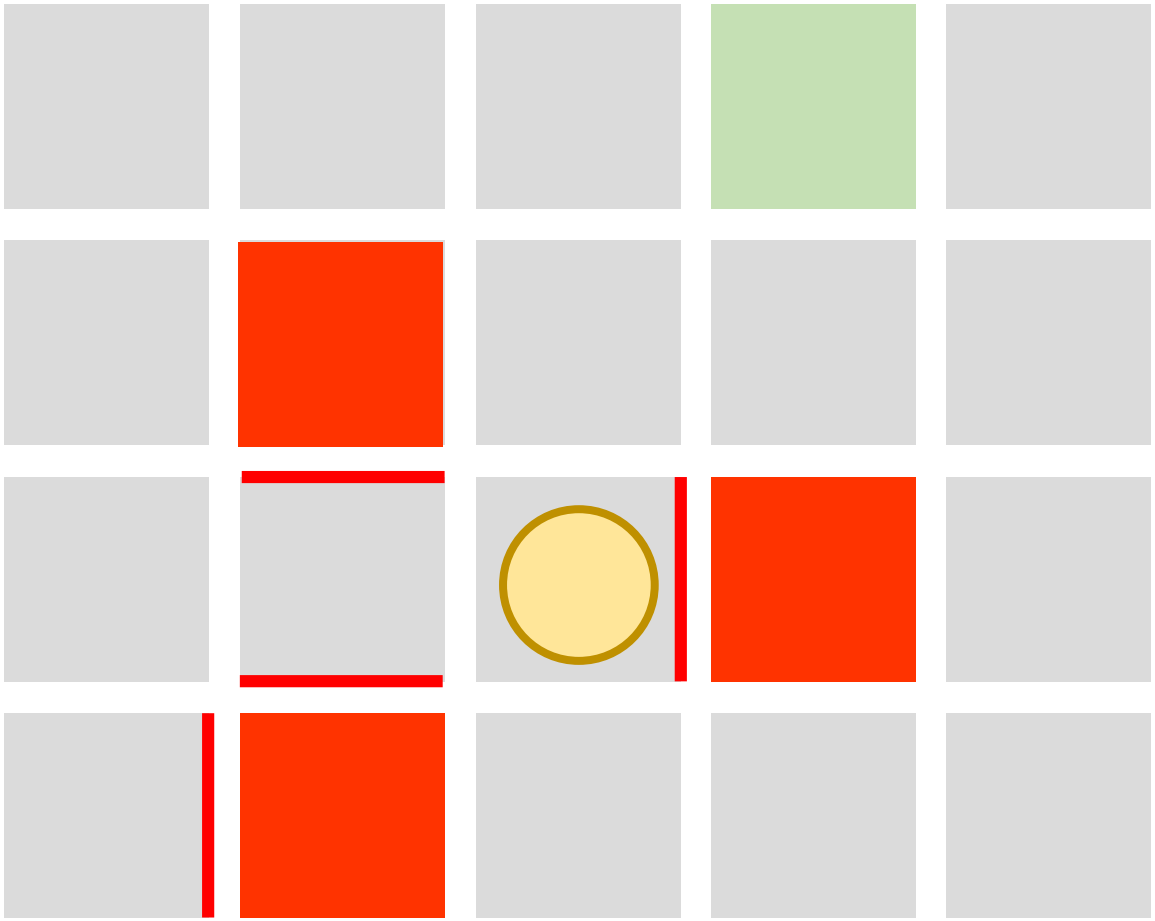
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



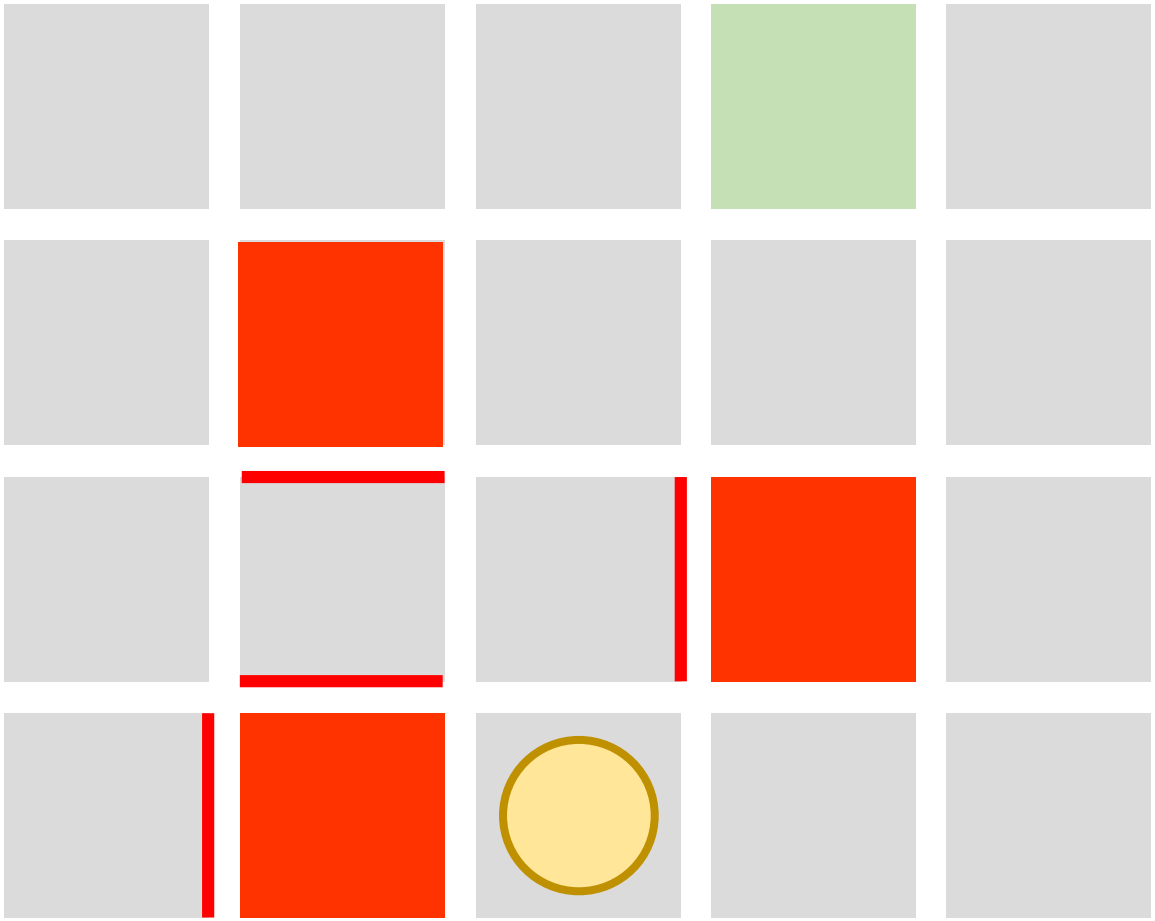
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



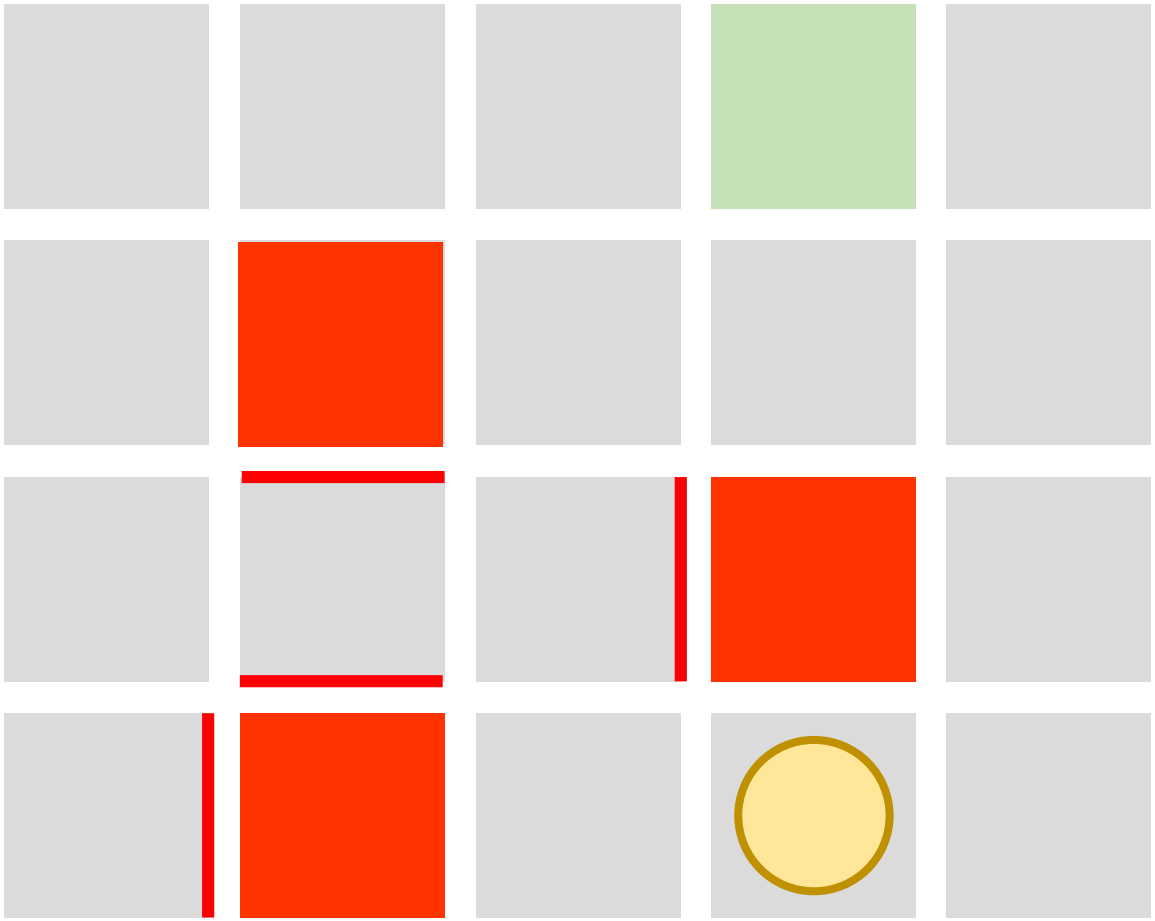
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



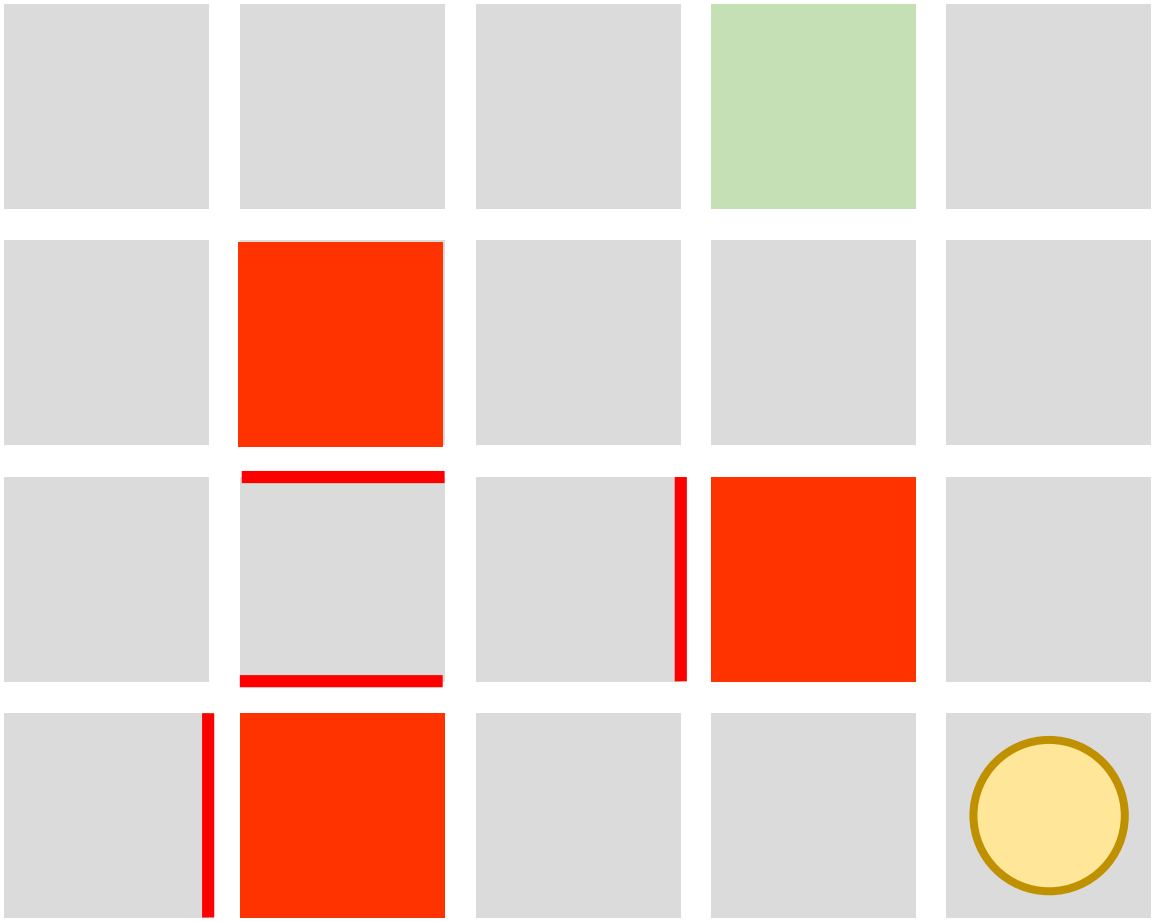
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



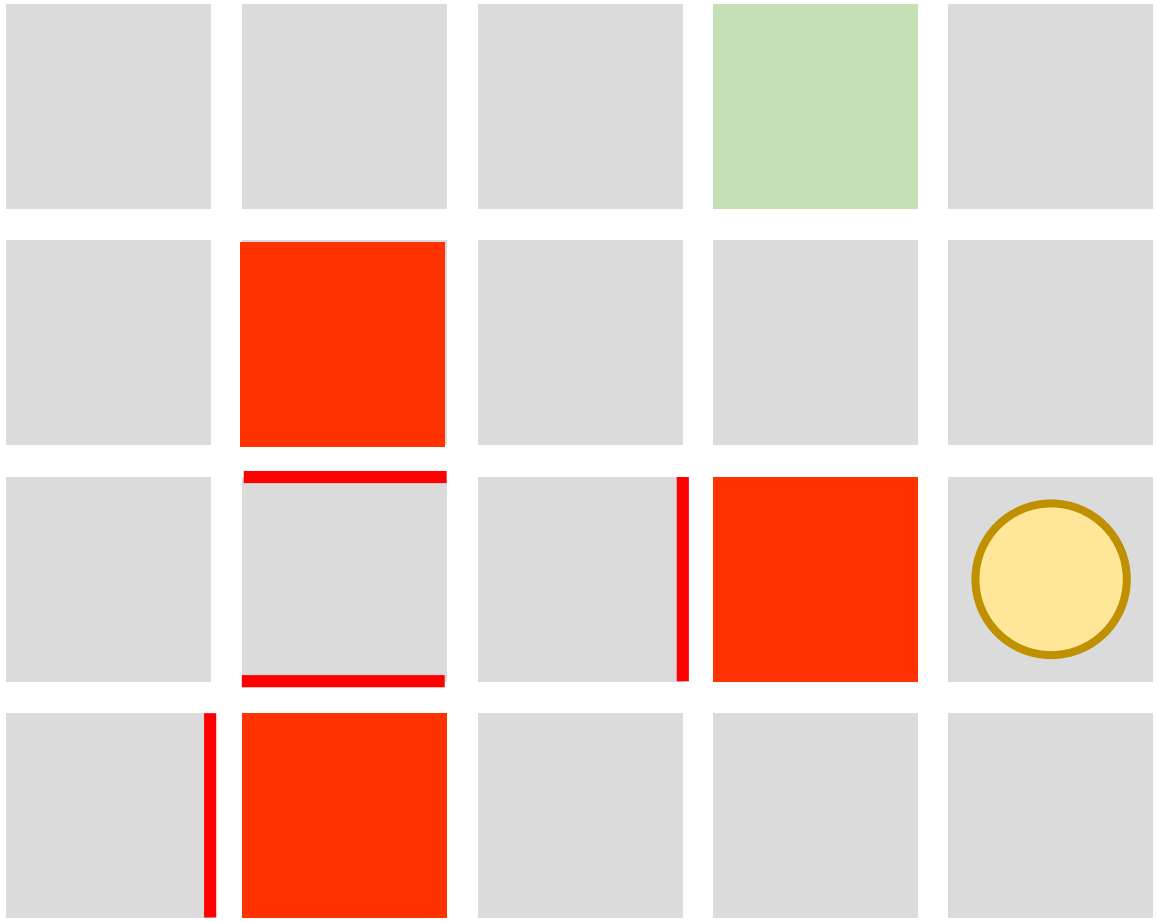
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



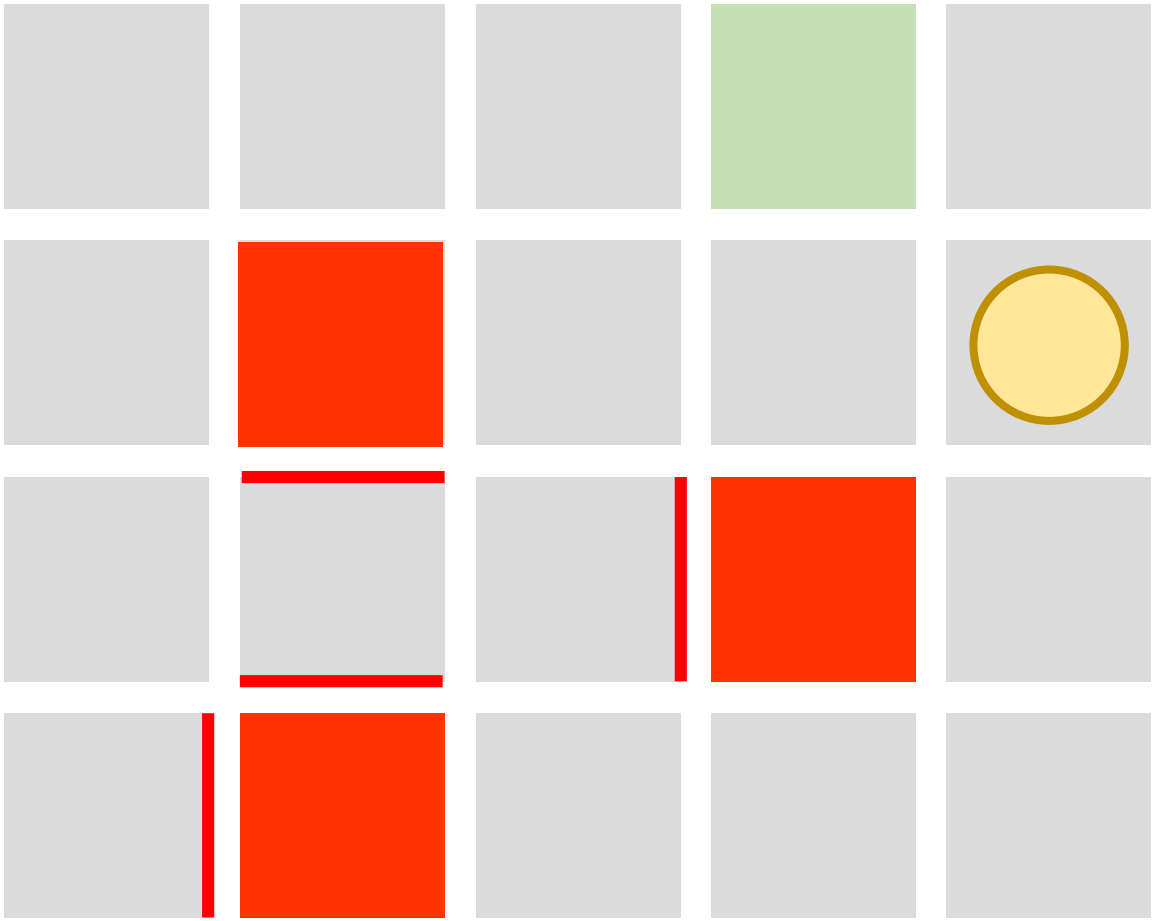
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



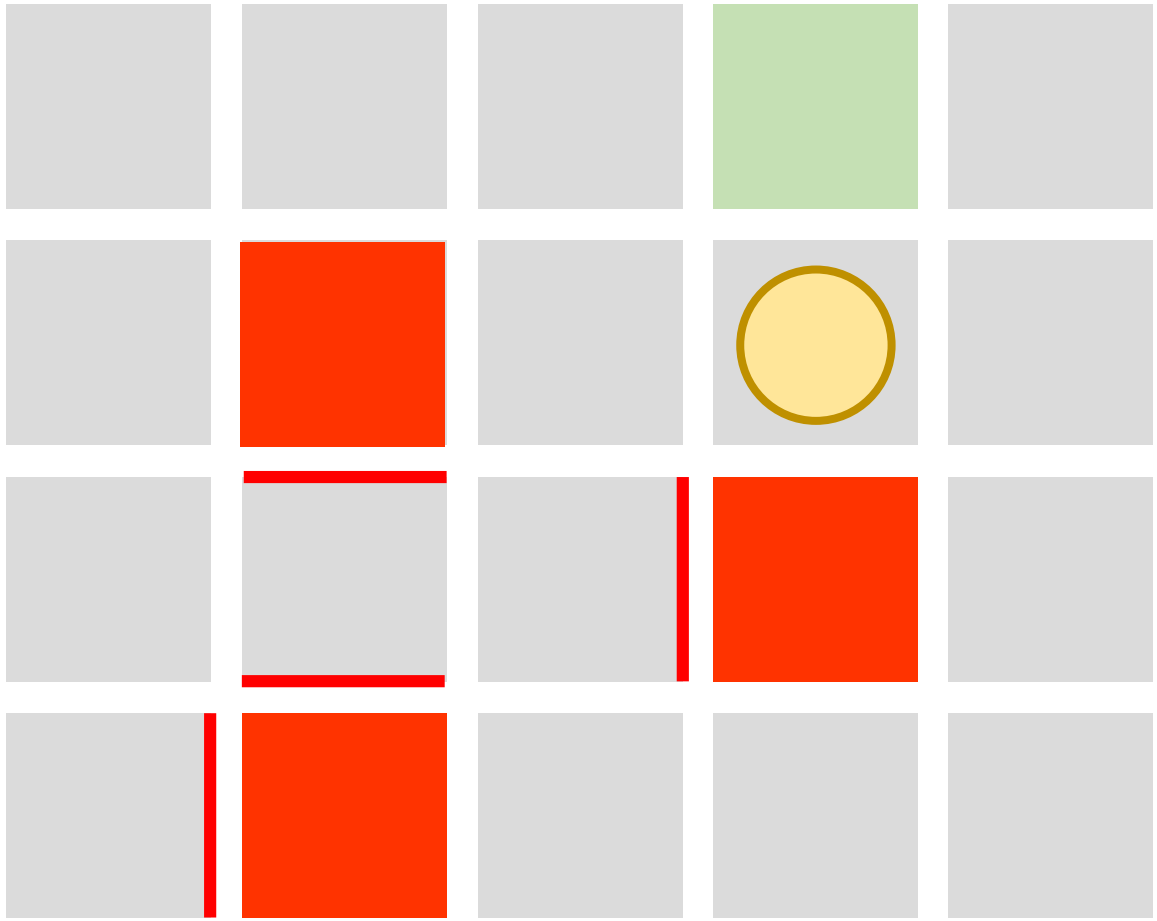
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



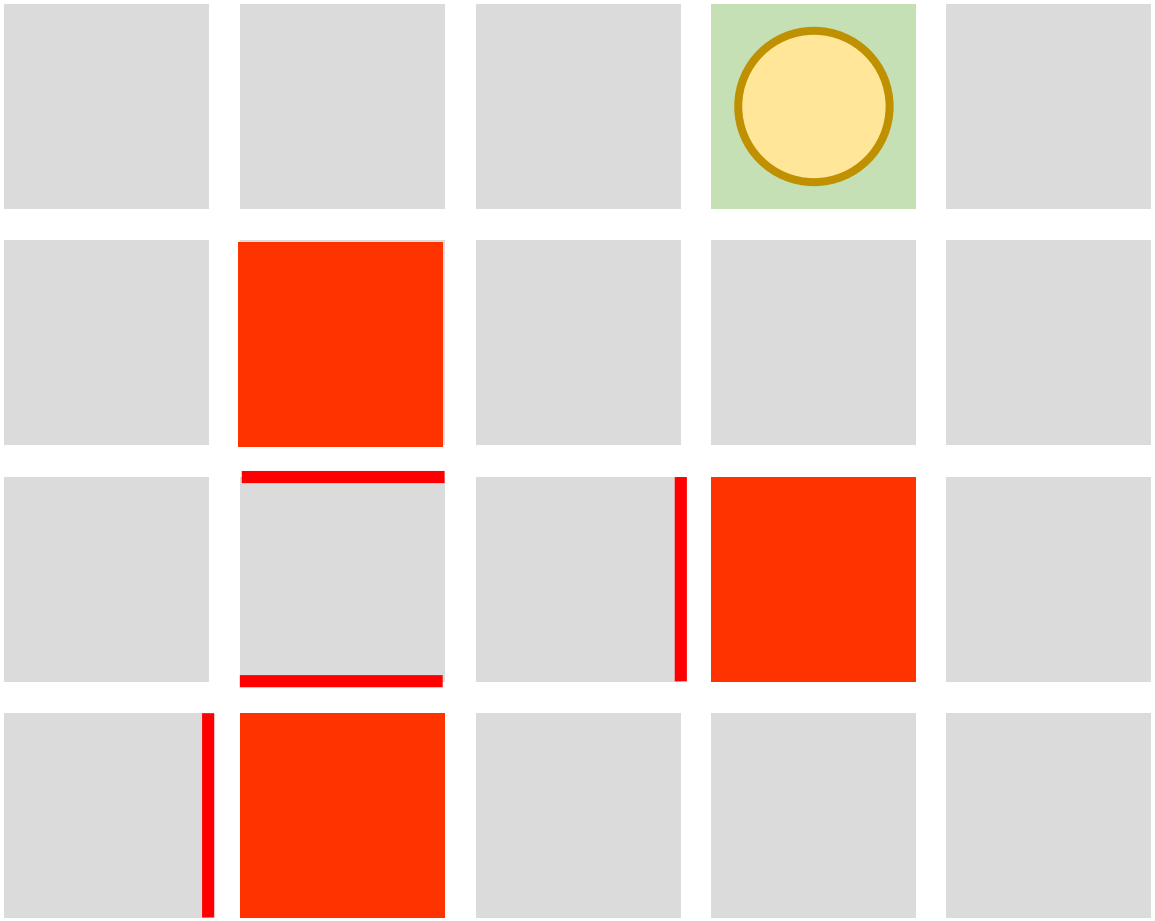
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



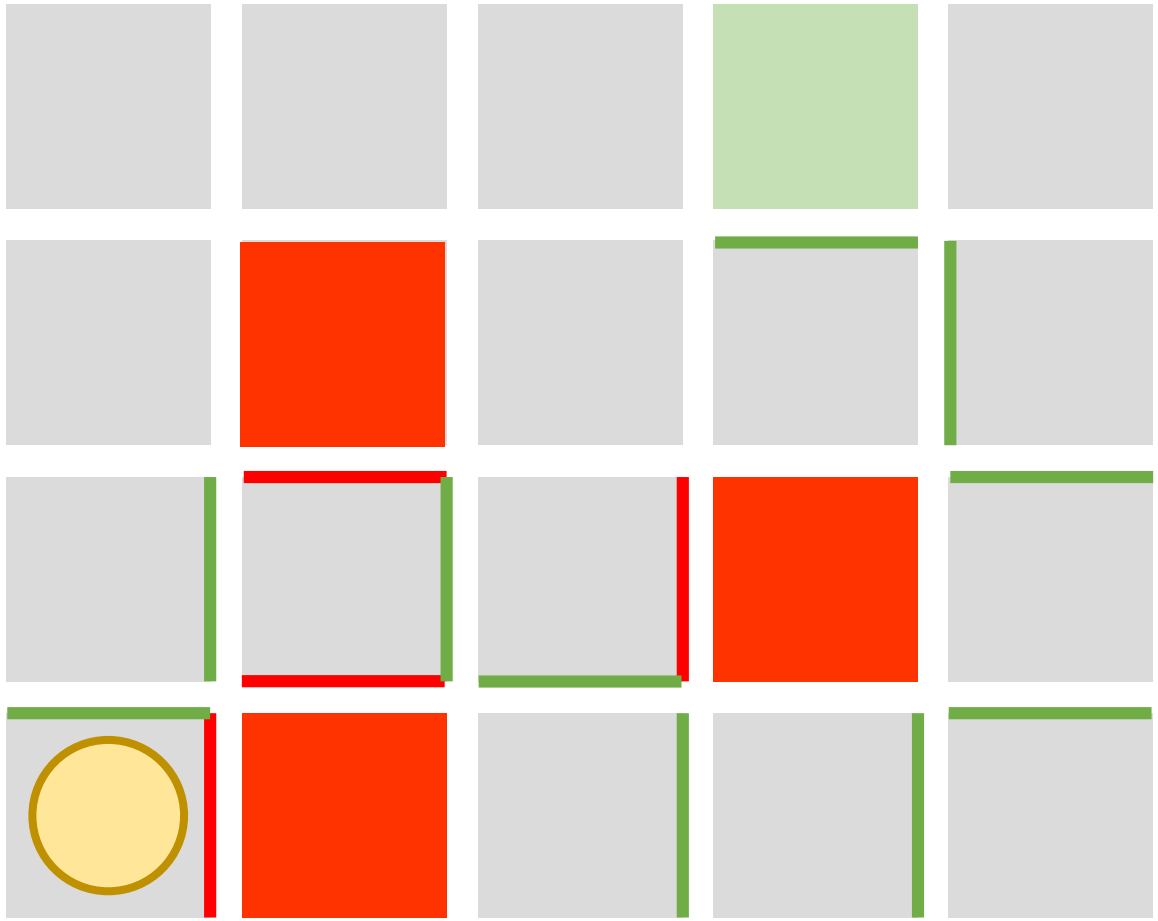
学到一些东西.....

马尔可夫决策过程 Markov Decision Process



学到一些东西.....

马尔可夫决策过程 Markov Decision Process



学习了哪个行动是坏的

学习了哪个行动是好的

马尔可夫决策过程 Markov Decision Process

- 状态集合 state S
- 行动集合 $\mathrm{Action}(s)$
- 转移模型 Transition model $P(s' \mid s, a)$
- 奖励函数 Reward function $R(s, a, s')$

马尔可夫决策过程 Markov Decision Process

- 马尔可夫Markov？
  - 每个行动的结果仅取决于当前状态(Current State)
  - $P(S_{t+1}|S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1}, \dots, S_0 = s_0) = P(S_{t+1}|S_t = s_t, A_t = a_t)$

BUSS 3620.人工智能导论

#2. 策略

刘佳璐

安泰经济与管理学院

上海交通大学

策略 Policy

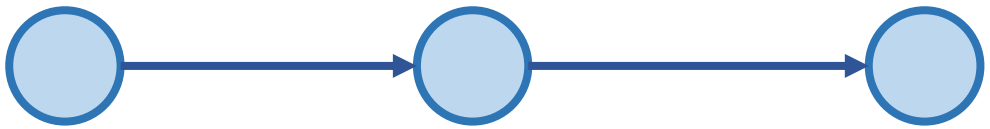
- 策略 π: $State \rightarrow Action$
  - 每个状态的行动
- 最优策略 π^*
  - 执行最优策略可以让我们的收益最大化

行动序列的价值

- 练习 #1：
- 数字代表每个行动获得的奖励，智能体会更偏好哪个行动序列?
  - [1, 2, 3] vs [2, 3, 4]
  - [0, 0, 1] vs [1, 0, 0]

行动序列的价值

- 练习 #1：
- 数字代表每个行动获得的奖励，智能体会更偏好哪个行动序列?
  - [1, 2, 3] vs [2, 3, 4]　　最大化总的奖励
  - [0, 0, 1] vs [1, 0, 0]　　更喜欢更早获得奖励



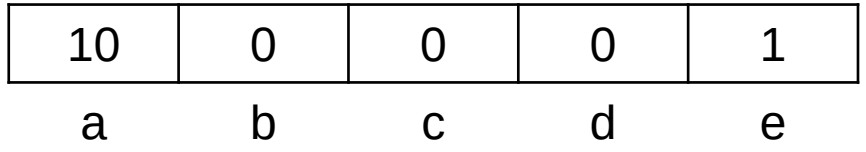
$t-1$　　t　　$t+1$

1　　γ　　γ^2　　总奖励：$1+\gamma+\gamma^2$

$\gamma < 1$

练习 #2:

- 有右图的MDP
  - b,c,d行动：左，右，
  - a,e行动：退出



① 如果$\gamma = 1$, 最优策略是什么？

② 如果$\gamma = 0.1$, 最优策略是什么？

③ 对于状态d来说，γ需要满足什么条件使得向左向右的策略是一样好？

有问题吗?

- 请随时举手提问。

策略 Policy

- 策略$\pi^*(s)$：在状态s下执行的最佳行动
  - 最佳行动：$\underset{a}{\operatorname{argmax}}\, Q(s, a)$
- 价值$Q(s, a)$：在状态s下执行行动a的预期价值
  - 当前收获的奖励 + 未来可能得到的奖励

BUSS 3620.人工智能导论

#3. 价值

刘佳璐

安泰经济与管理学院

上海交通大学

价值 Value

- Q – learning：学习价值函数$Q(s, a)$的方法
  - 一开始, $Q(s, a)$ 是未知的
  - 但智能体可以根据每一步的奖励来学习$Q(s, a)$

Q – learning 概要

- 一开始，对所有s, a，设置 $Q(s, a) = 0$
- 每当智能体执行了一个行动，获得了一些奖励:
  - 根据当前奖励(current reward)和预期未来奖励(expected future reward)来估计一个新的$Q(s, a)$
  - 根据旧的$Q(s, a)$(old estimate)和新的$Q(s, a)$(new estimate)来更新$Q(s, a)$
  - 注意：新的$Q(s, a)$不是更新的$Q(s, a)$

Q – learning

- 一开始，对所有s, a，设置 $Q(s, a) = 0$
- 每当智能体在状态s下执行了行动a，获得了一些奖励r:
- $Q(s, a) \leftarrow (1 - \alpha) \times$ 旧价值估计 $+ \alpha \times$ 新价值估计
  - α :学习速率
    - 与旧信息的重视程度相比，我们对新信息的重视程度
    - $\alpha = 1$:只关心新信息
    - $\alpha = 0$:忽略所有新信息

$Q(s, a) \leftarrow$ 旧价值估计 $+ \alpha \times$ (新价值估计 $-$ 旧价值估计)

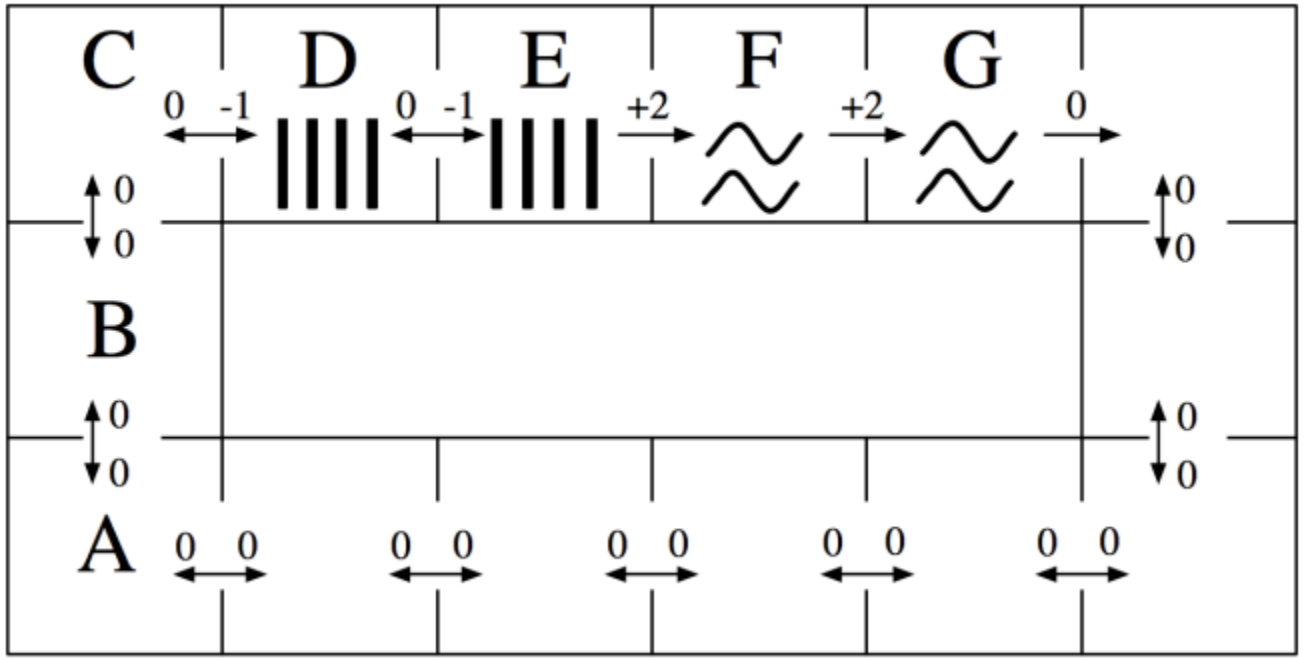
感知机模型的学习法则：$w_i = w_i + \alpha$(真实值 $-$ 估计值) $\times x_i$

Q – learning（贝尔曼方程 Bellman equation)

- 一开始，对所有s, a，设置 $Q(s, a) = 0$
- 每当智能体在状态s下执行了行动a，获得了一些奖励r，到达了新状态s':
- $Q(s,a) \leftarrow (1 - \alpha) \times$ 旧价值估计 $+ \alpha \times$ 新价值估计
- $Q(s,a) \leftarrow (1 - \alpha) \times Q(s,a) + \alpha \times$ 新价值估计
- $Q(s,a) \leftarrow (1 - \alpha) \times Q(s,a) + \alpha \times (r +$ 未来得到的最大的奖励估计)
- $Q(s,a) \leftarrow (1 - \alpha) \times Q(s,a) + \alpha \times (r + \max_{a'} Q(s',a'))$
- $Q(s,a) \leftarrow (1 - \alpha) \times Q(s,a) + \alpha \times (r + \gamma \times \max_{a'} Q(s',a'))$
  - γ：折现系数：与当前获得的奖励相比，我们有多重视未来可能获得的奖励

练习 #3

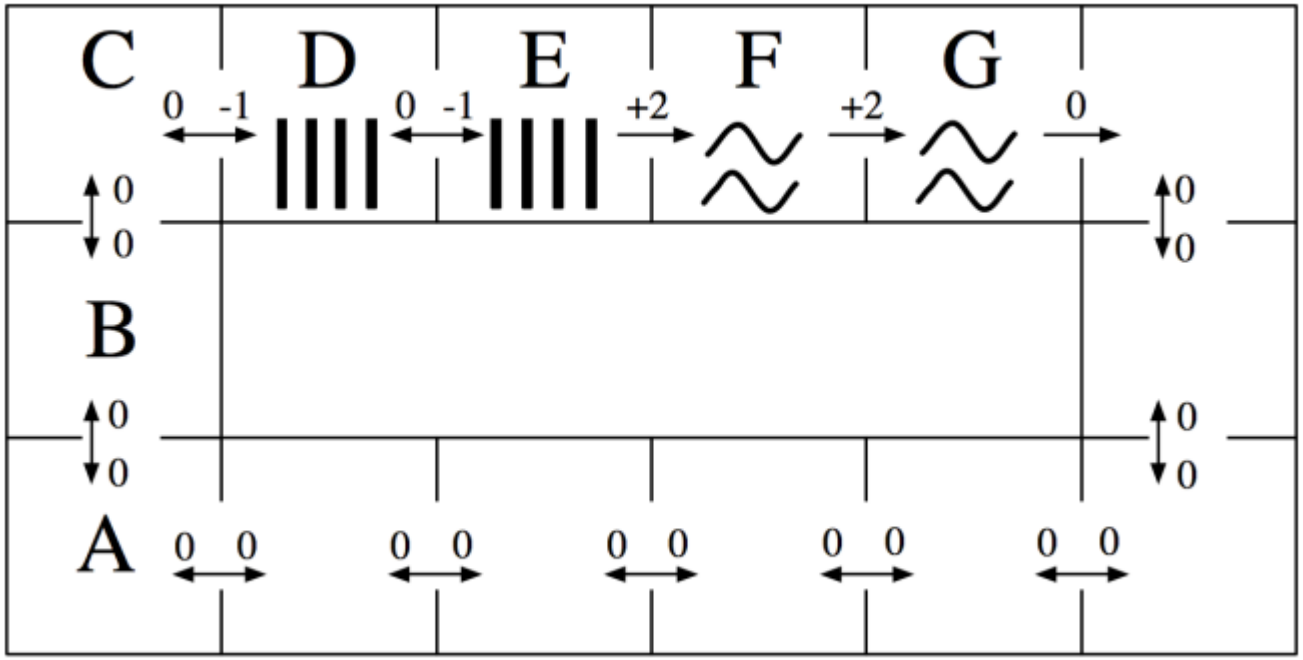
- 有下图所示的地图，每个格子是一个状态。想象整个地图是一个赛车轨道，其中D、E是减速带，会带来-1的奖励，F、G是加速带，会带来+2的奖励。箭头代表每个状态可以执行的行动，箭头上的数字是奖励。智能体必须移动，不可以静止。
- 假设$\alpha = 0.5$， $\gamma = 1$



练习 #3

- 假设$\alpha = 0.5$， $\gamma = 1$，填写下表:

$$Q(s,a) \leftarrow (1-\alpha) \times Q(s,a) + \alpha \times (r + \gamma \times \max_{a'} Q(s',a'))$$



	Q(D, ←)	Q(D, →)	Q(E, ←)	Q(E,→)
初始	0	0	0	0
第一轮：(s=D, a=→)				
第二轮：(s=E, a=→)				
第三轮：(s=E, a=←)				
第四轮：(s=D, a=→)				

有问题吗?

- 请随时举手提问。

BUSS 3620.人工智能导论

#4. 强化学习策略

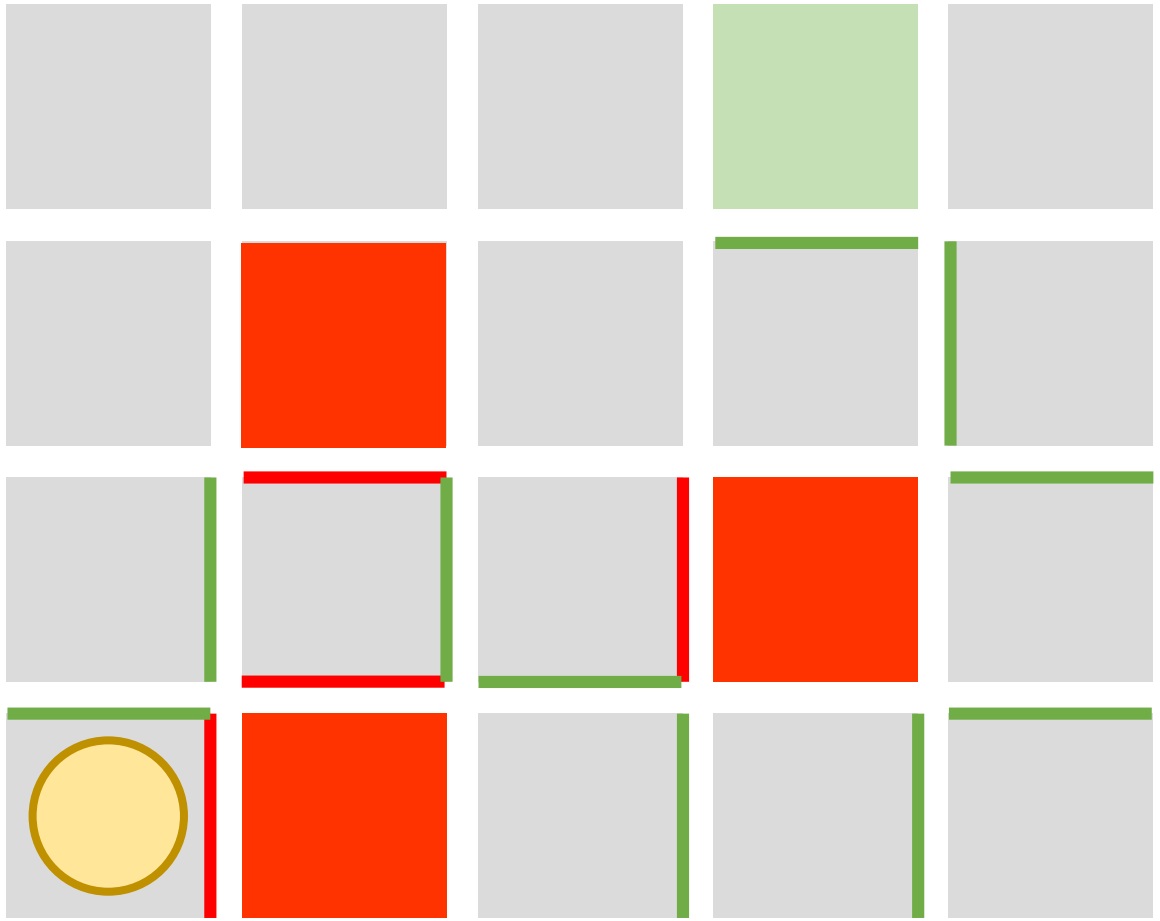
刘佳璐

安泰经济与管理学院

上海交通大学

贪婪决策 Greedy Policy

- 当处于状态s时, 选择$Q(s, a)$**最大**的行动a

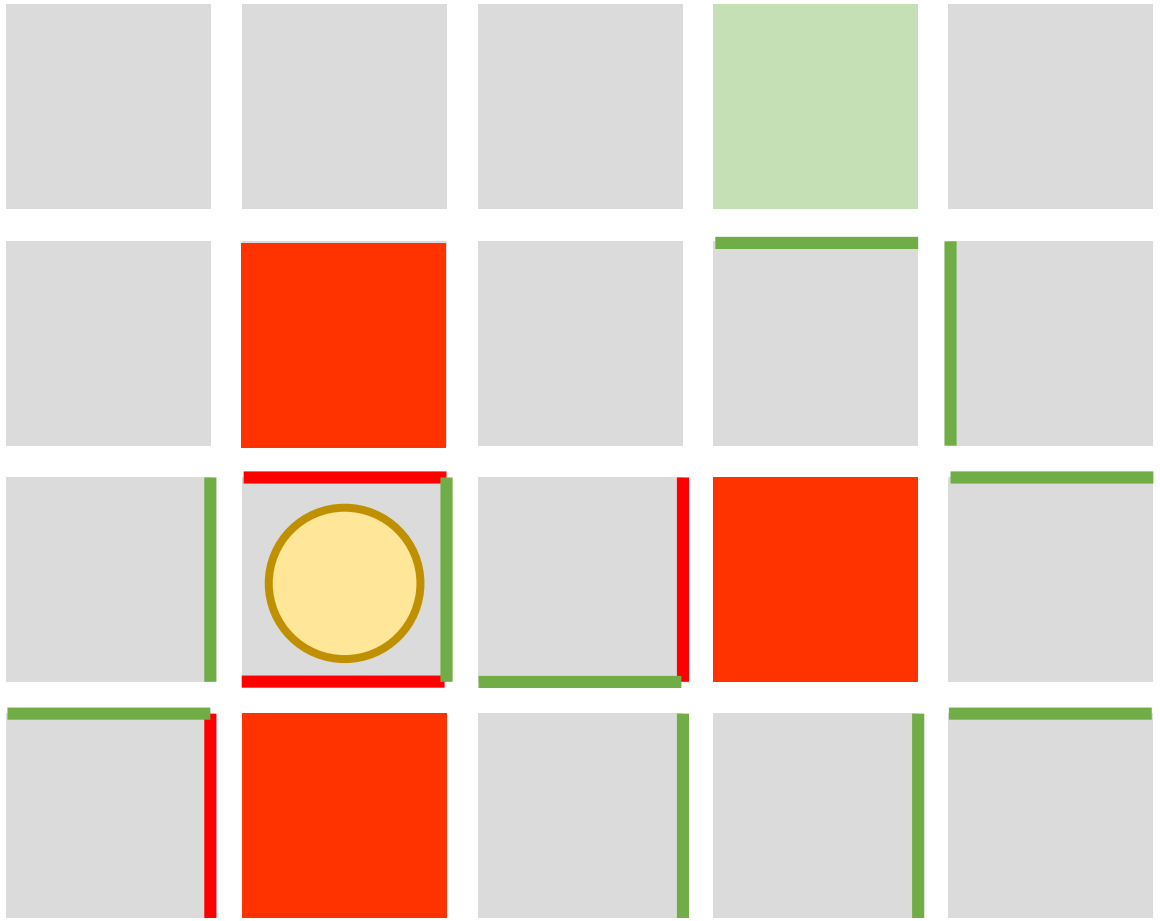


贪婪决策 Greedy Policy

- 当处于状态s时，选择$Q(s, a)$**最大**的行动a

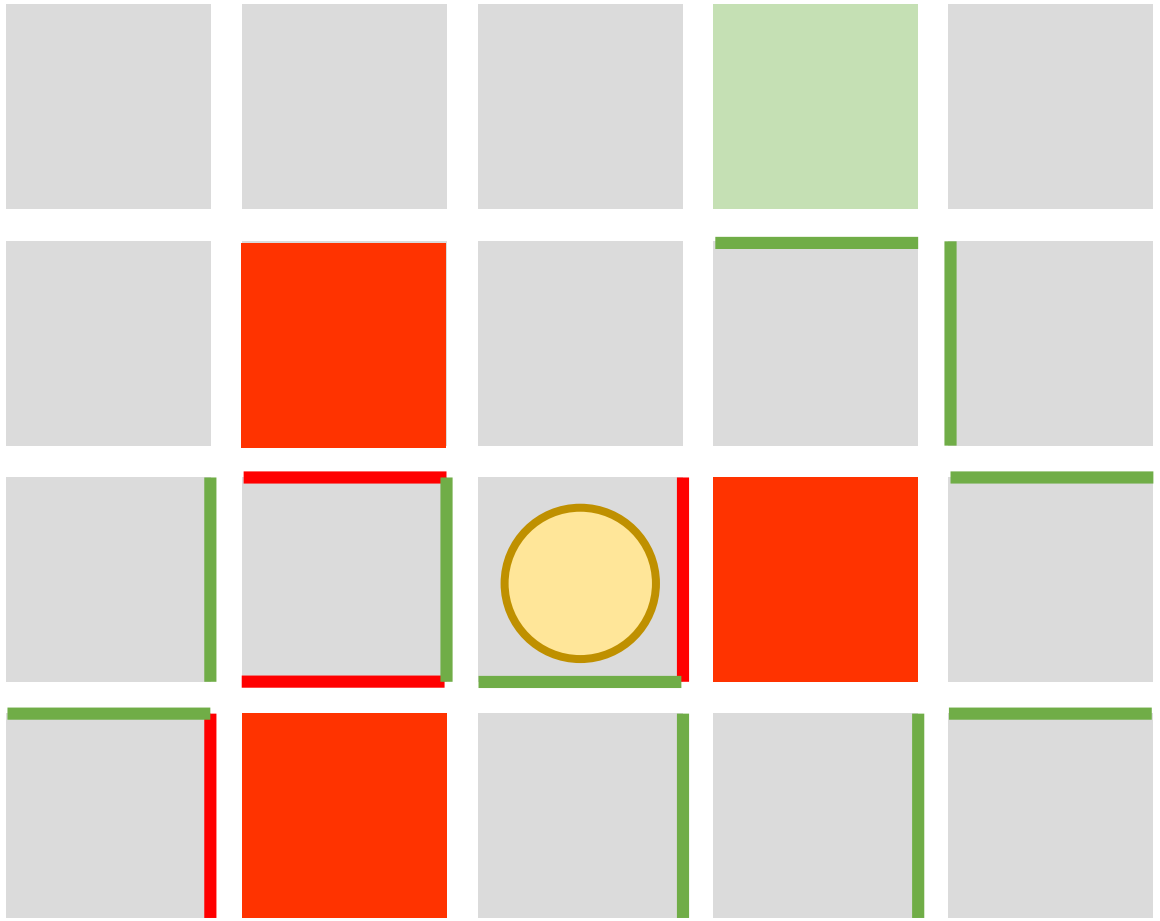
贪婪决策 Greedy Policy

- 当处于状态s时, 选择$Q(s, a)$**最大**的行动a



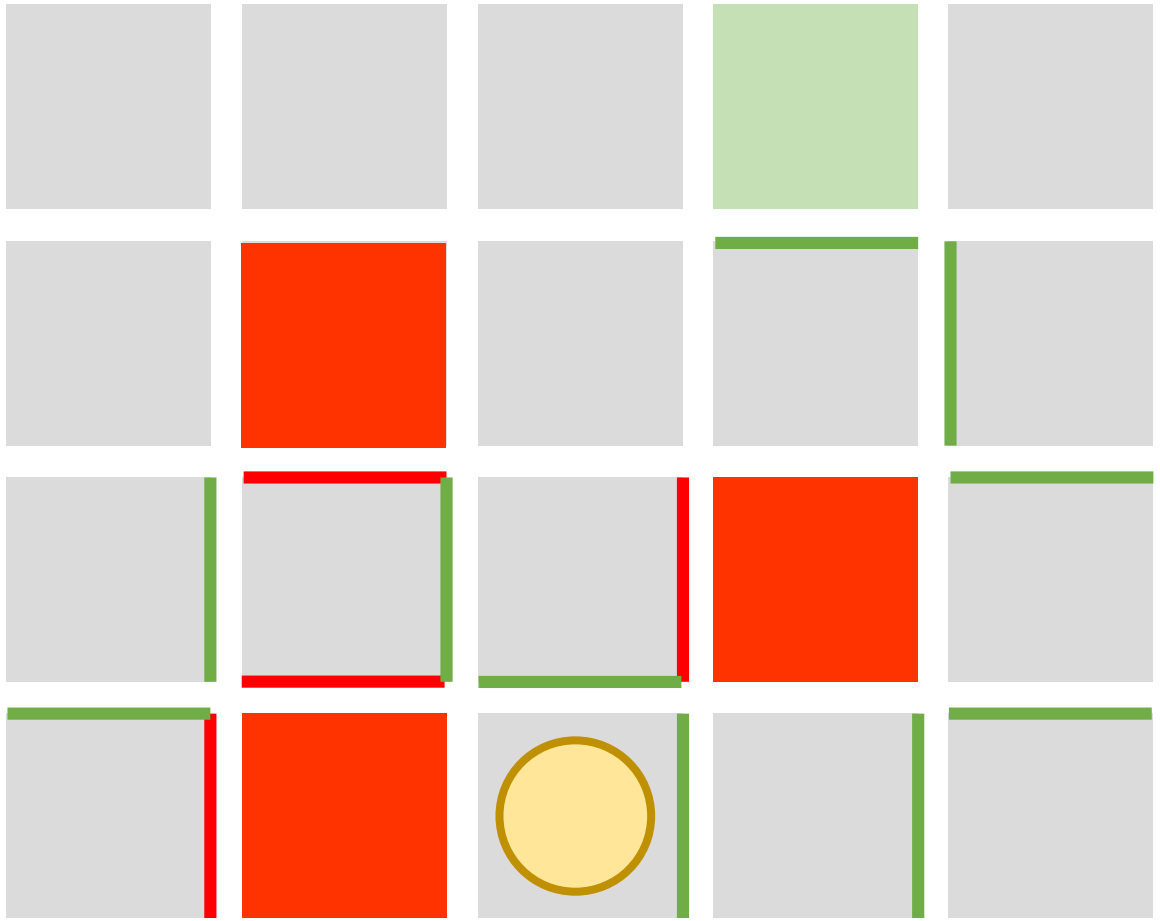
贪婪决策 Greedy Policy

- 当处于状态s时，选择$Q(s, a)$**最大**的行动a



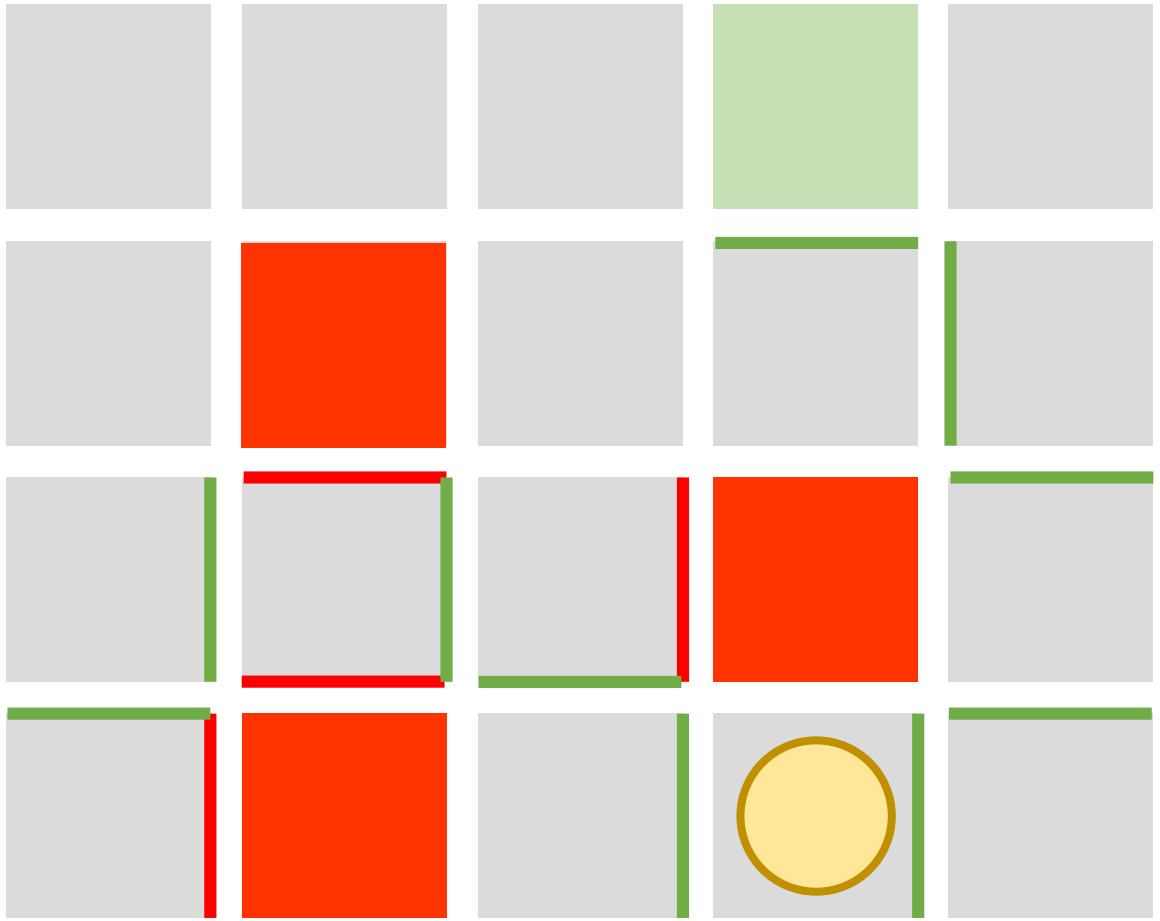
贪婪决策 Greedy Policy

- 当处于状态s时，选择$Q(s, a)$**最大**的行动a



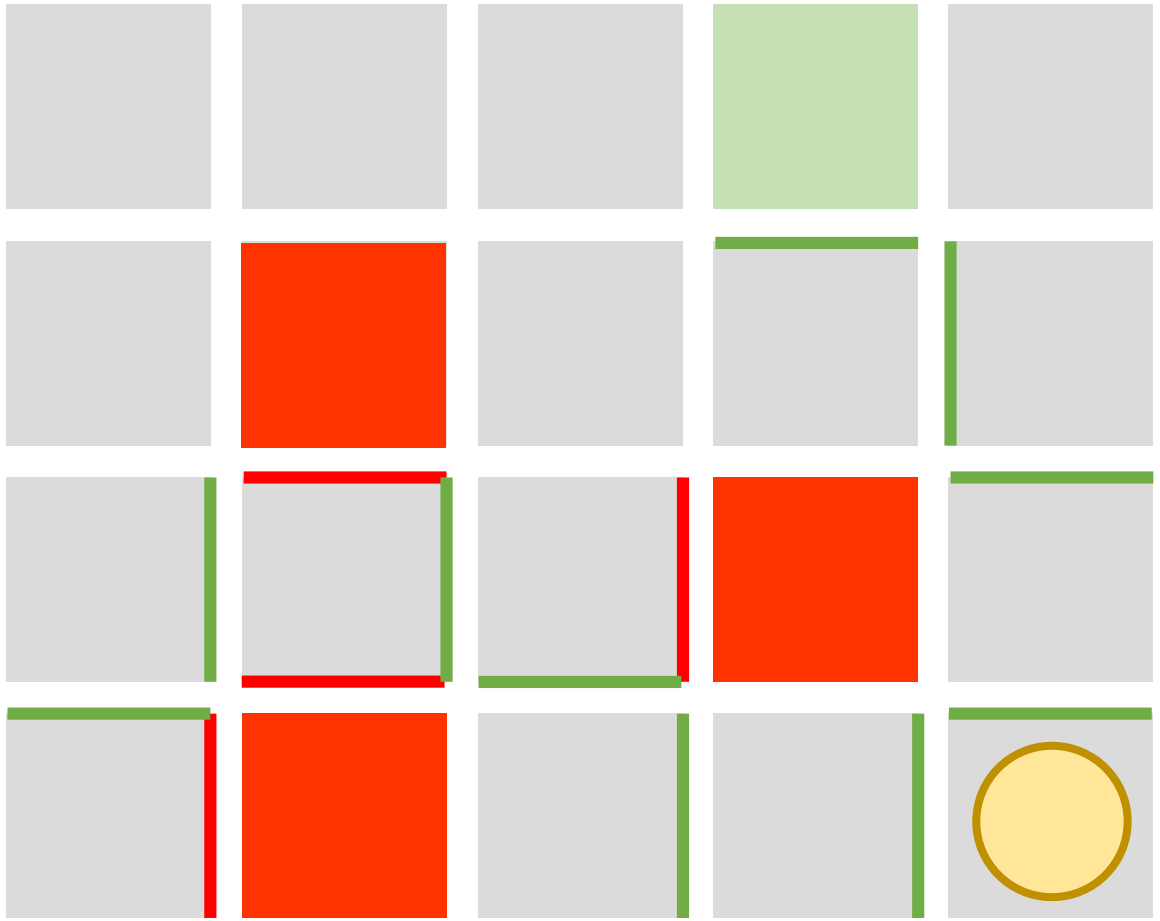
贪婪决策 Greedy Policy

- 当处于状态s时，选择$Q(s, a)$**最大**的行动a



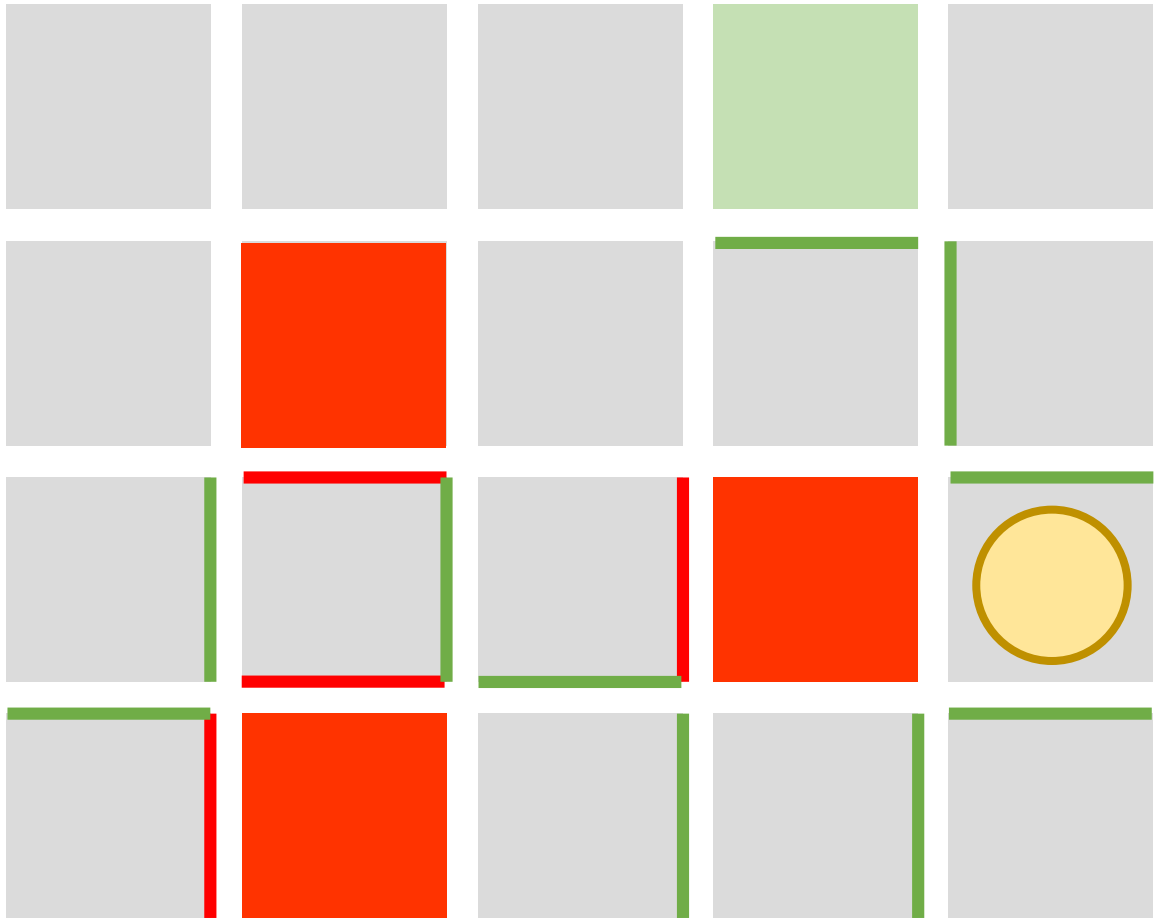
贪婪决策 Greedy Policy

- 当处于状态s时，选择$Q(s, a)$**最大**的行动a



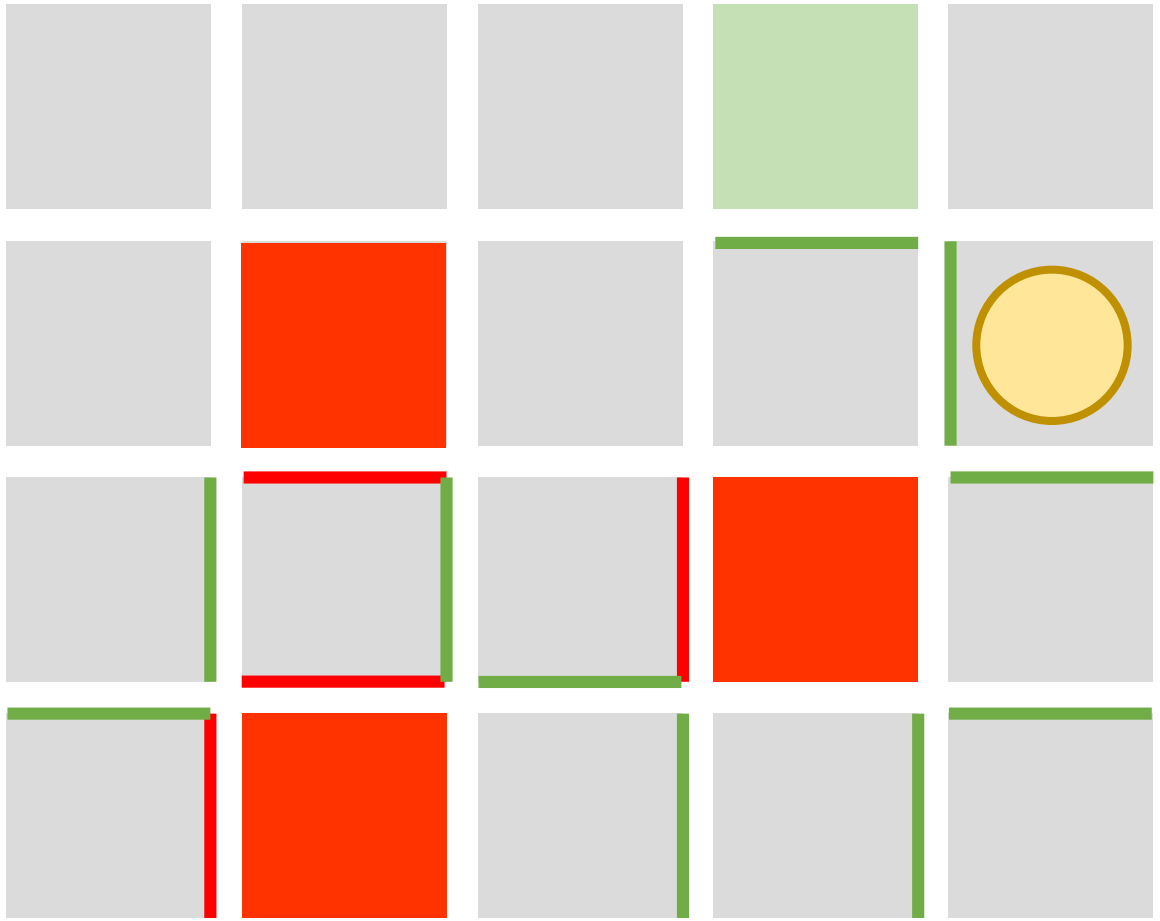
贪婪决策 Greedy Policy

- 当处于状态s时，选择$Q(s, a)$**最大**的行动a



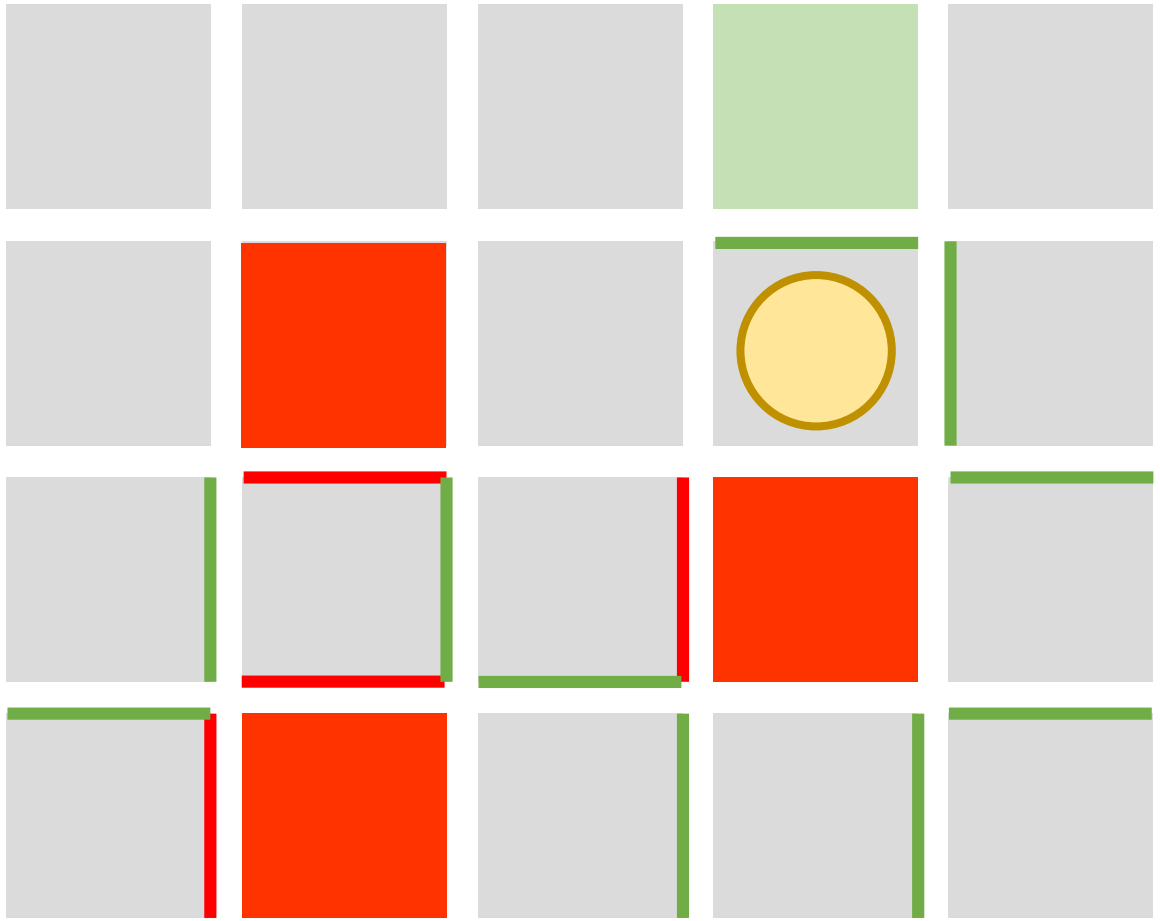
贪婪决策 Greedy Policy

- 当处于状态s时，选择$Q(s, a)$**最大**的行动a



贪婪决策 Greedy Policy

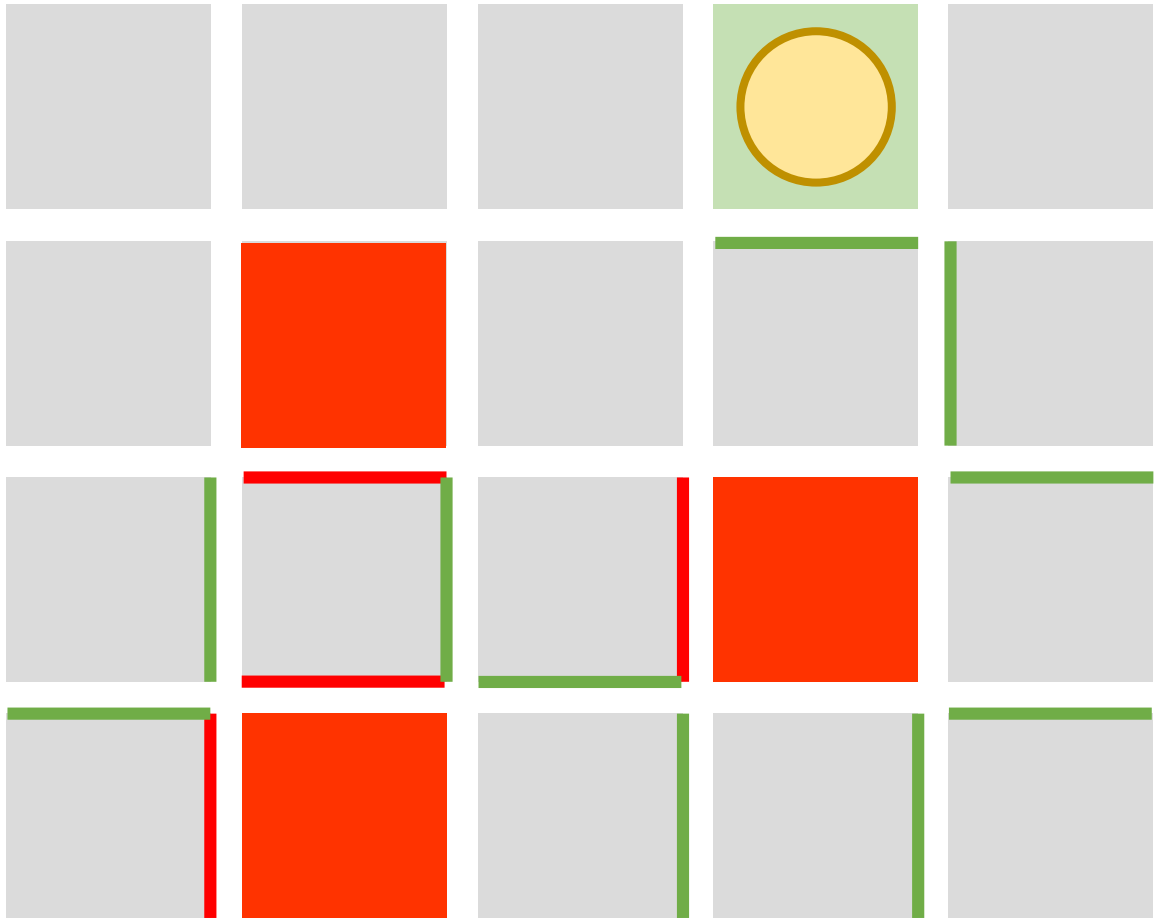
- 当处于状态s时, 选择$Q(s, a)$**最大**的行动a



贪婪决策 Greedy Policy

- 当处于状态s时, 选择$Q(s, a)$**最大**的行动a

这可能不是最佳路线



贪婪决策 Greedy Policy

- 当处于状态s时，选择$Q(s, a)$**最大**的行动a

这可能不是最佳路线

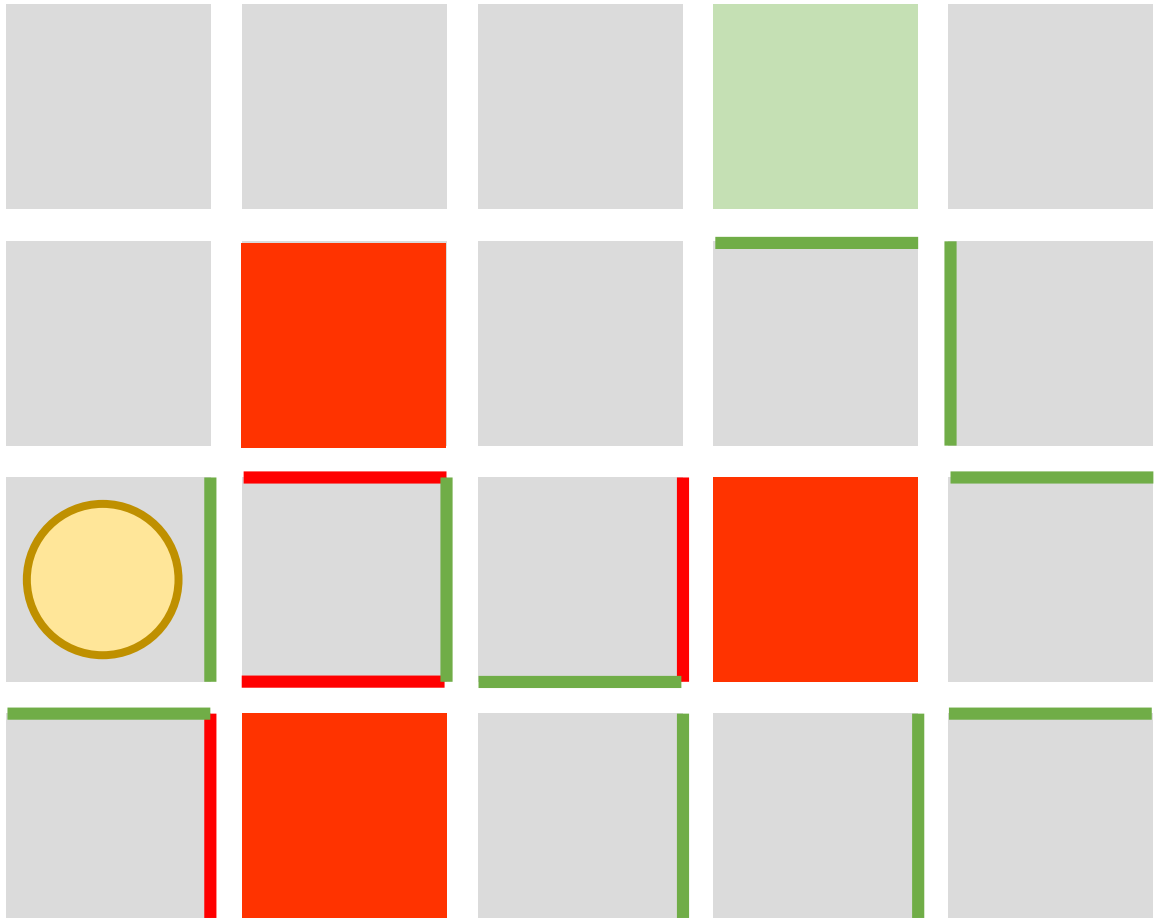
存在其他最佳路线

贪婪决策 Greedy Policy

- 当处于状态s时，选择$Q(s, a)$**最大**的行动a

这可能不是最佳路线

存在其他最佳路线

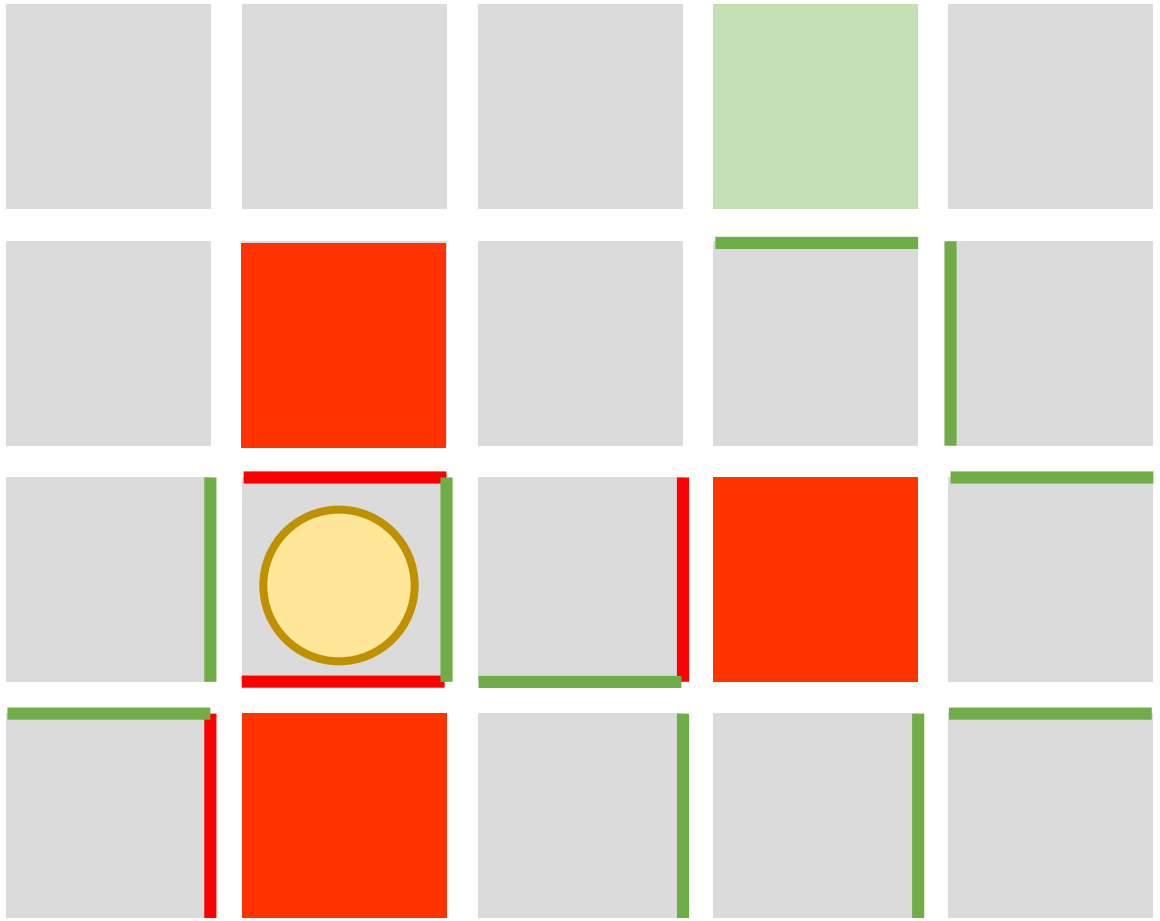


贪婪决策 Greedy Policy

- 当处于状态s时，选择$Q(s, a)$**最大**的行动a

这可能不是最佳路线

存在其他最佳路线

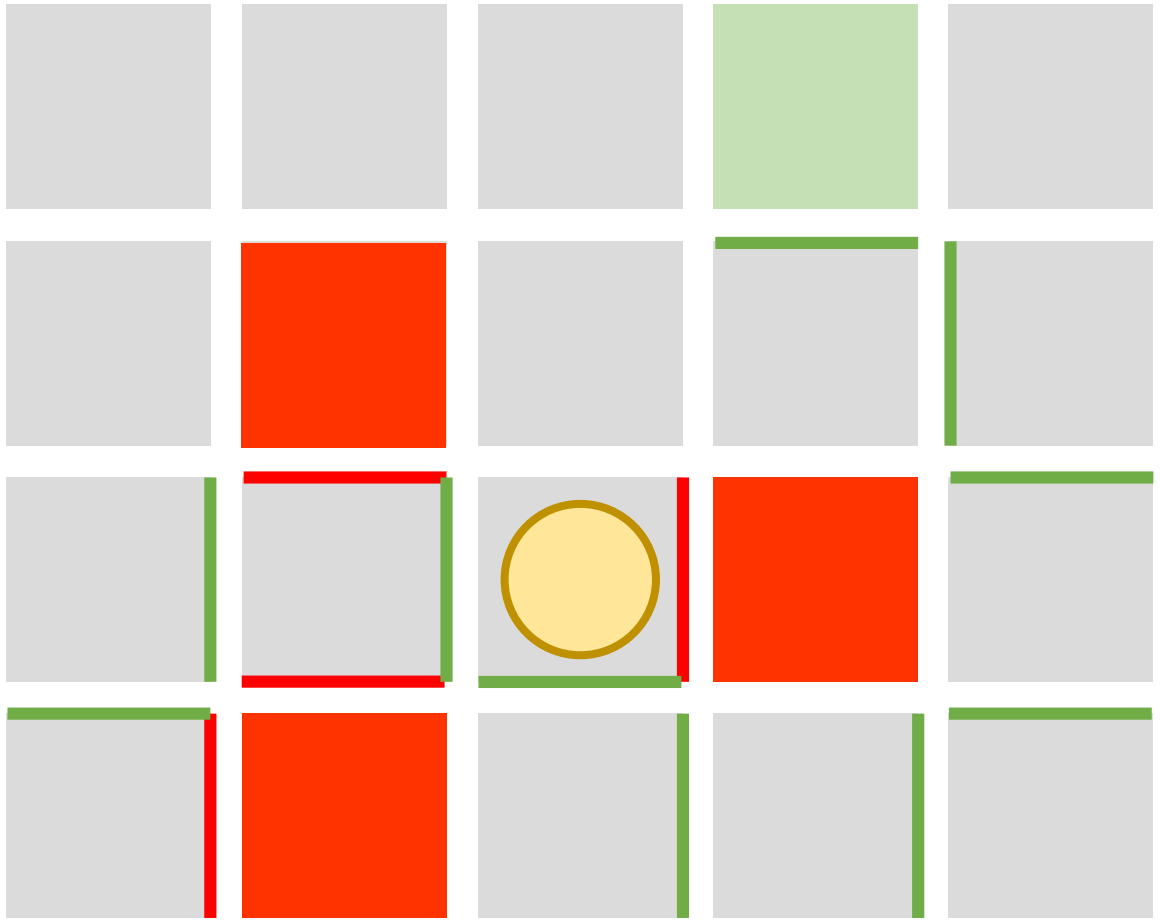


贪婪决策 Greedy Policy

- 当处于状态s时，选择$Q(s, a)$**最大**的行动a

这可能不是最佳路线

存在其他最佳路线

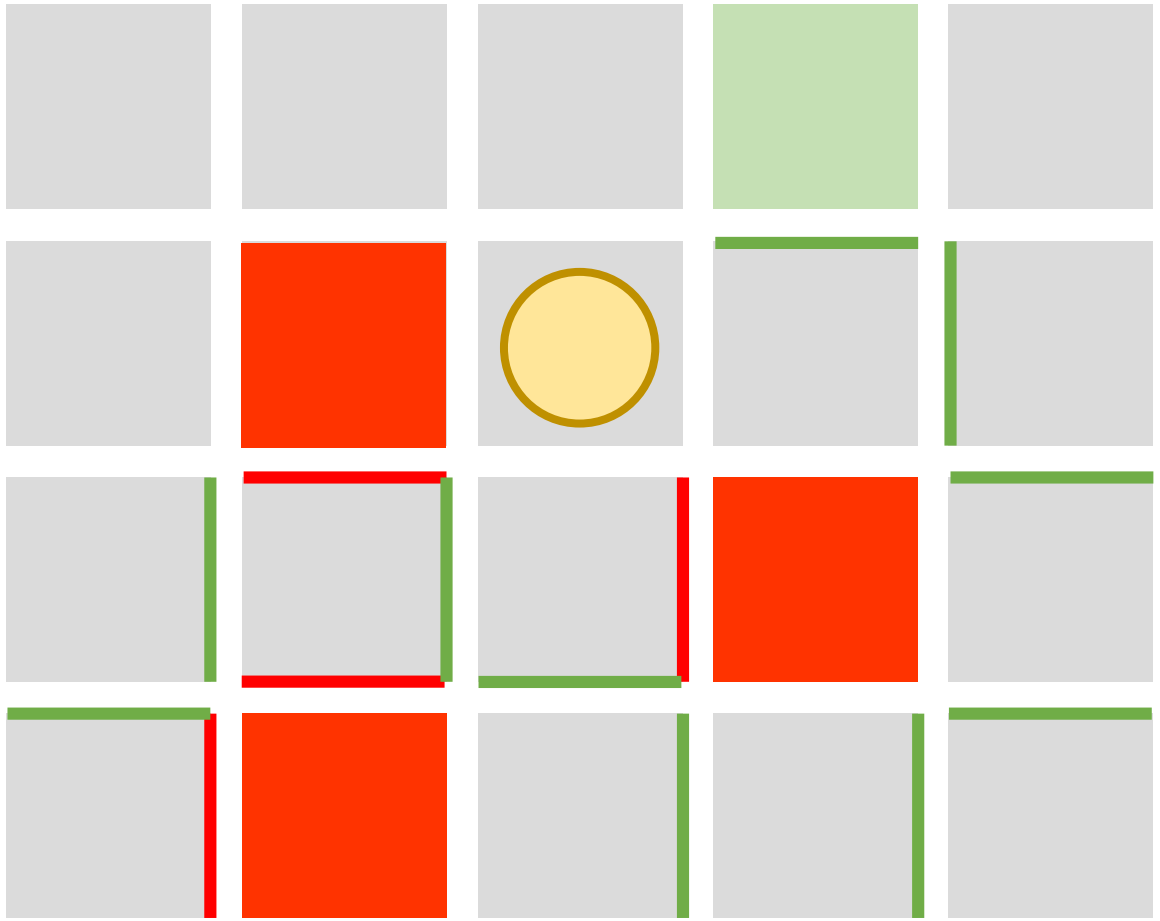


贪婪决策 Greedy Policy

- 当处于状态s时，选择$Q(s, a)$**最大**的行动a

这可能不是最佳路线

存在其他最佳路线

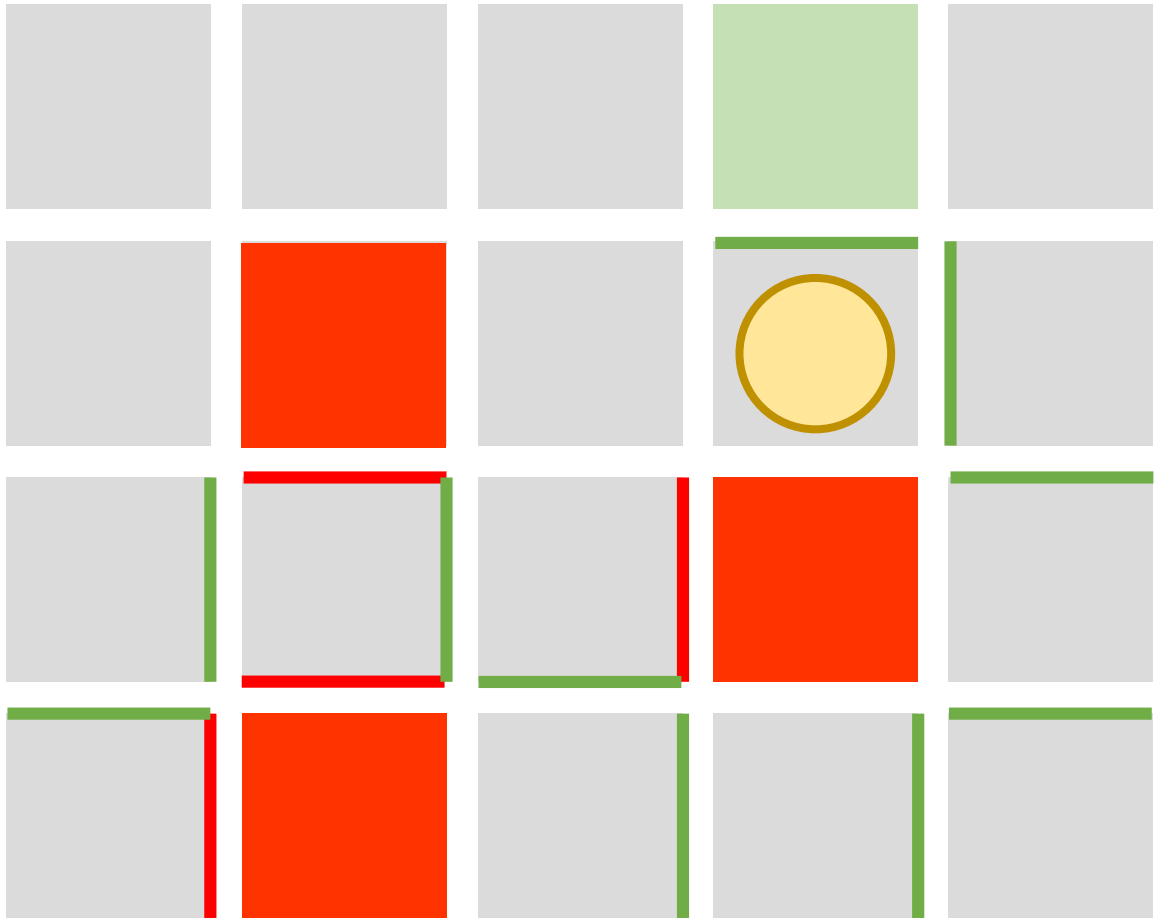


贪婪决策 Greedy Policy

- 当处于状态s时，选择$Q(s, a)$**最大**的行动a

这可能不是最佳路线

存在其他最佳路线

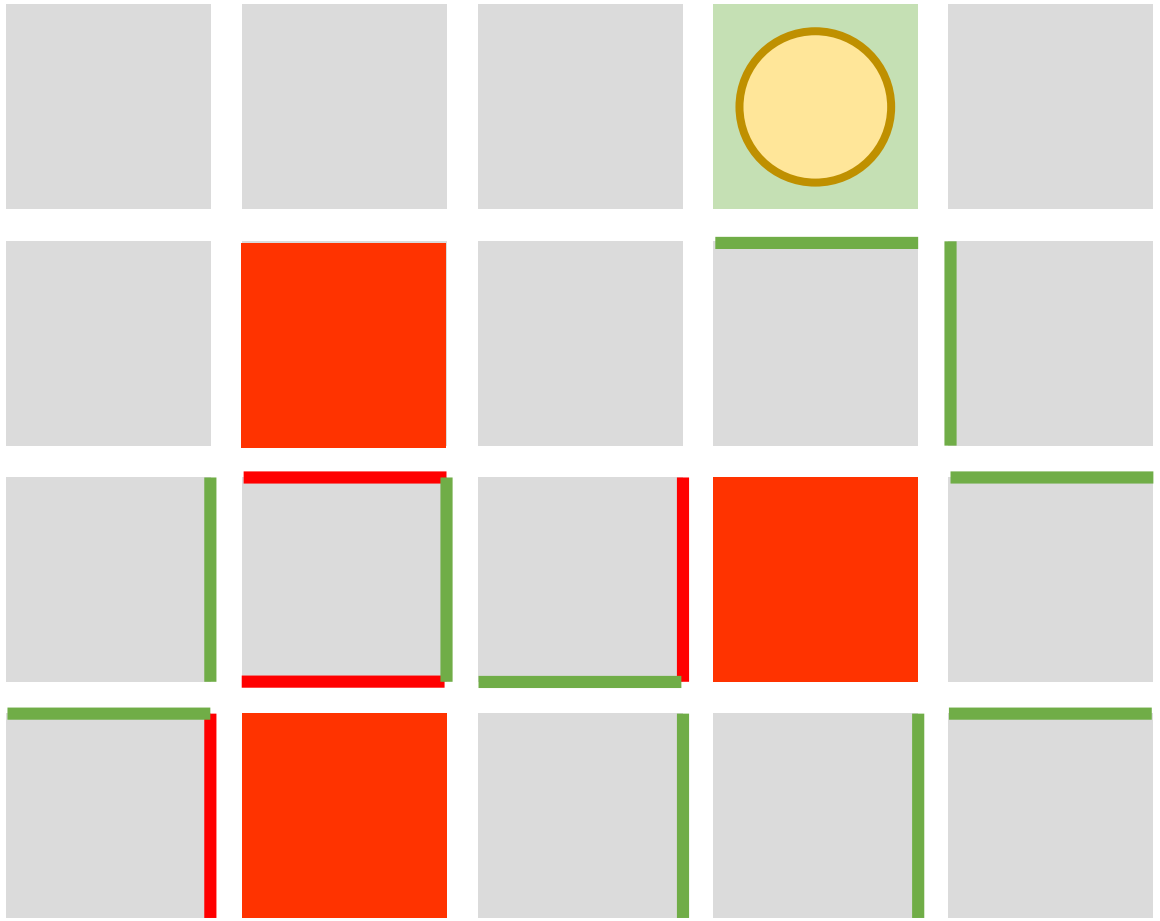


贪婪决策 Greedy Policy

- 当处于状态s时，选择$Q(s, a)$**最大**的行动a

这可能不是最佳路线

存在其他最佳路线

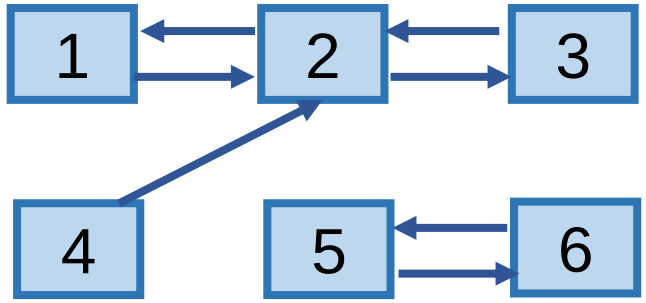


探索 Exploration v.s. 利用 Exploitation

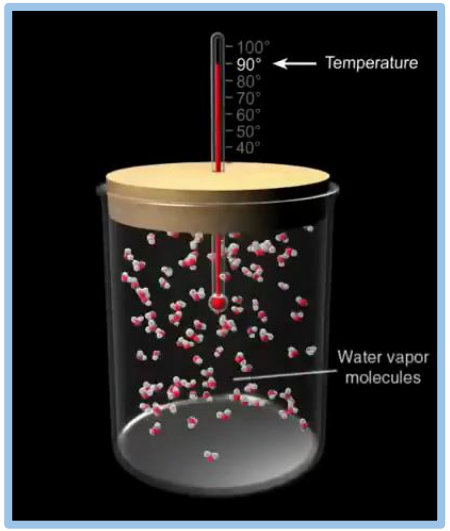
- 探索 Exploration
  - 探索以前没有彻底探索过的其他行动
- 利用 Exploitation
  - 利用 AI 已经拥有的知识
  - 如果只是利用，智能体可能会得到奖励，但奖励不会是最大的

ε贪婪 ε - greedy

- ε为智能体随机移动的概率(探索)
- 概率1 - ε：根据现有知识选取最优行动
- 概率ε：选择一个随机的行动



PageRank



模拟退火 Simulated Annealing

有问题吗?

- 请随时举手提问。

BUSS 3620.人工智能导论

#3. 代码示例：尼姆游戏

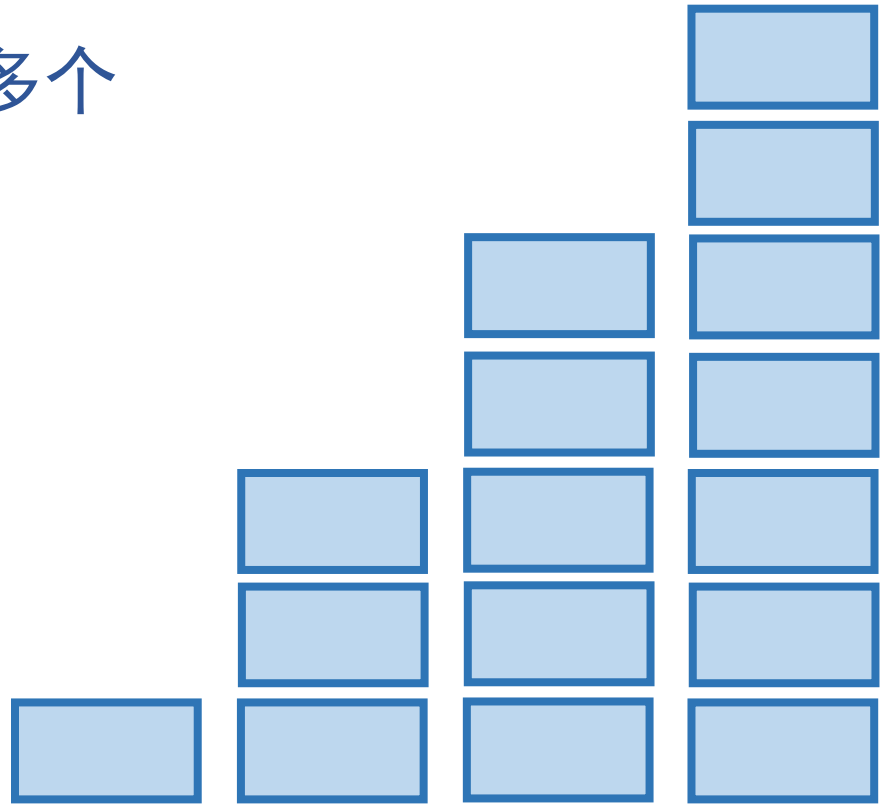
刘佳璐

安泰经济与管理学院

上海交通大学

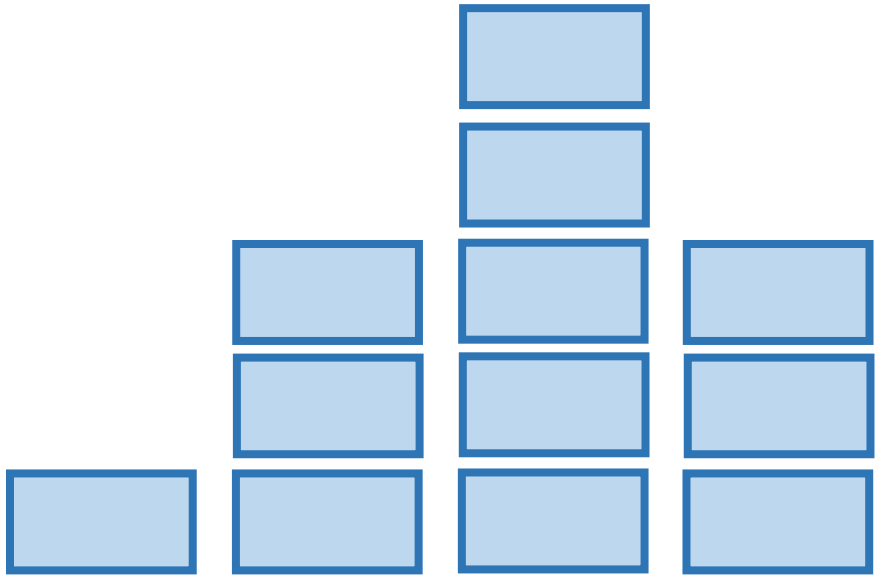
尼姆游戏 Nim

- 不同的物体排列成几堆
- 玩家轮流从一堆木块中取走一个或者多个
- 取走最后一木块的玩家输



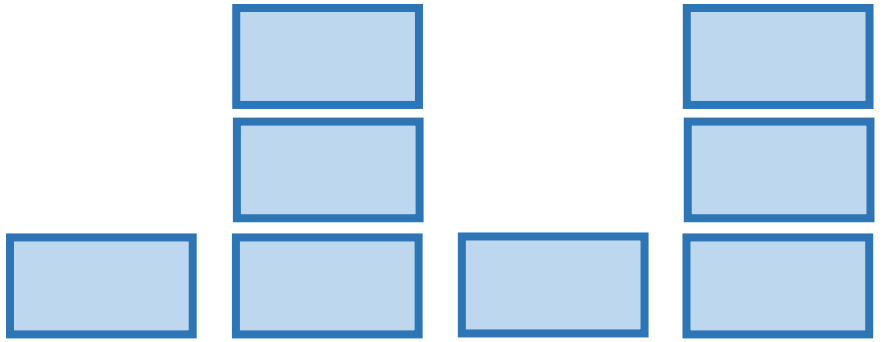
尼姆游戏 Nim

- 不同的物体排列成几堆
- 玩家轮流从一堆木块中取走一个或者多个
- 取走最后一木块的玩家输



尼姆游戏 Nim

- 不同的物体排列成几堆
- 玩家轮流从一堆木块中取走一个或者多个
- 取走最后一木块的玩家输



尼姆游戏 Nim

- 不同的物体排列成几堆
- 玩家轮流从一堆木块中取走一个或者多个
- 取走最后一木块的玩家输

尼姆游戏 Nim

- 不同的物体排列成几堆
- 玩家轮流从一堆木块中取走一个或者多个
- 取走最后一木块的玩家输

尼姆游戏 Nim

- 不同的物体排列成几堆
- 玩家轮流从一堆木块中取走一个或者多个
- 取走最后一木块的玩家输

尼姆游戏 Nim

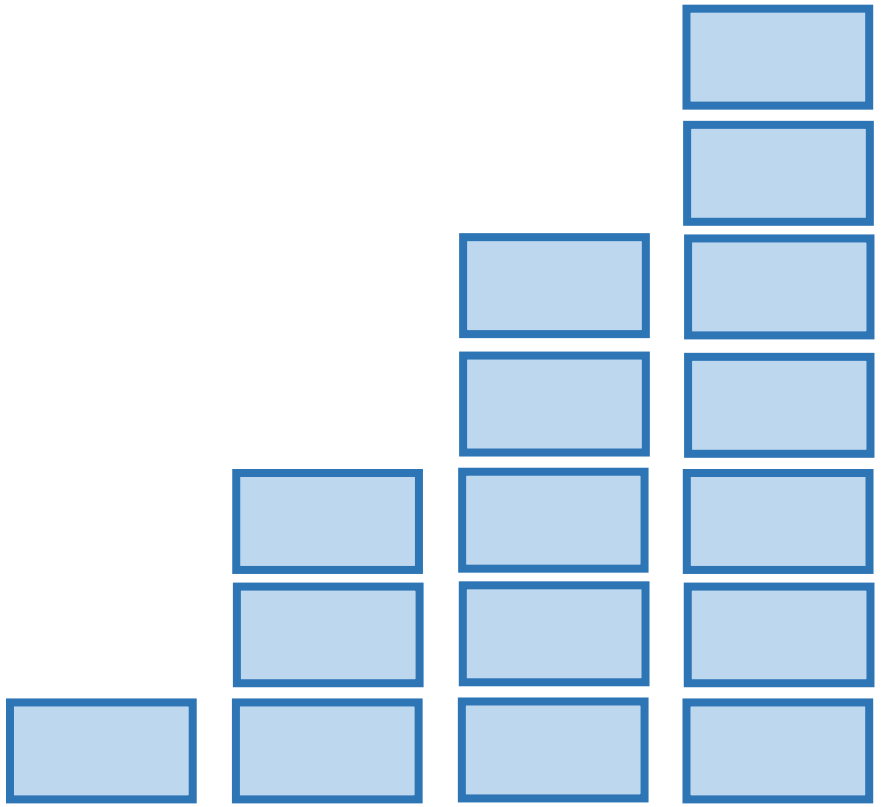
- 不同的物体排列成几堆
- 玩家轮流从一堆木块中取走一个或者多个
- 取走最后一木块的玩家输

强化学习要素

- 环境 Environment
  - 游戏规则（转移模型、中止条件）、选手
- 状态 state
- 行动 action
- 奖励 reward
- 状态价值 Value (Q值)
- 策略 Policy

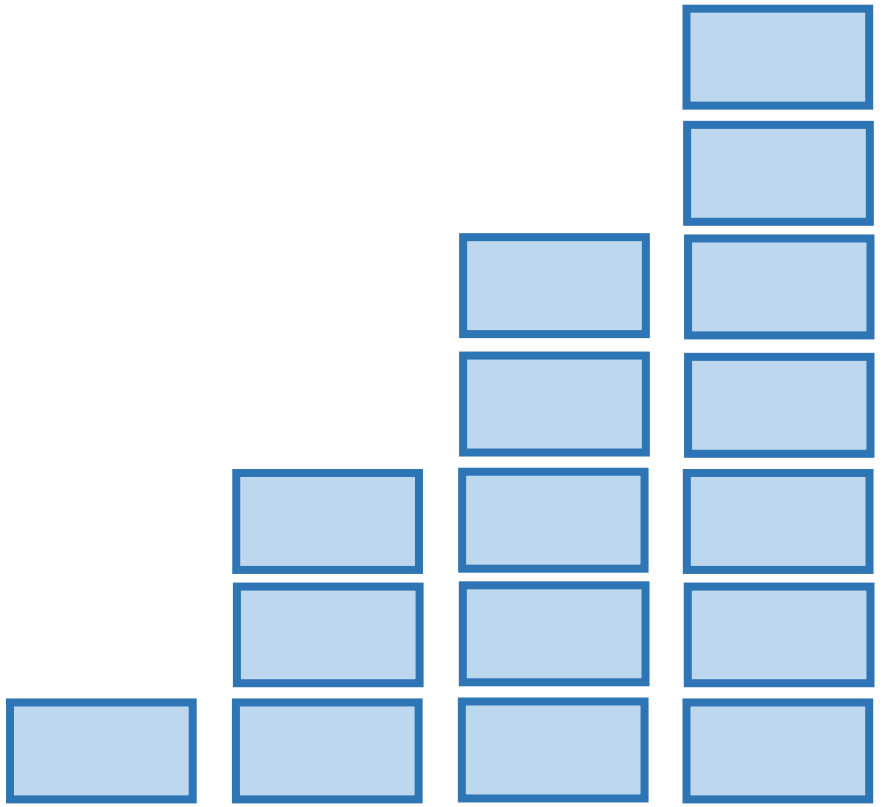
状态 state

- S_0: [1, 3, 5, 7]



行动 action

- 从第 i 堆中拿走 j 个木块
  - (i, j)
  - 从第 3 堆中拿走 3 个木块: (2, 3)
- 转移模型 Result(s, a) = s’
  - s: [1, 3, 5, 7], a: (2,3)
  - s’: [1, 3, 2, 7]



奖励 reward

- 拿走最后一个物体的玩家，输，获得奖励-1
  - If loser, reward = -1
- 拿走最后一个物体的玩家的对手，赢，获得奖励1
  - If winner, reward = 1
- 游戏进行中的动作，获得奖励0
  - If winner is none, reward = 0

状态价值 Value (Q值)

- 储存Q值
  - 字典q(state, action)
    - q[[1, 3, 5, 7], (2,3)]　　错误
    - q[(1, 3, 5, 7), (2,3)]
- Q-learning
  - 一开始，对所有s, a，设置 $Q(s, a) = 0$
  - 每当智能体在状态s下执行了行动a，获得了一些奖励r，到达了新状态s':
  - $Q(s,a) \leftarrow (1 - \alpha) \times Q(s,a) + \alpha \times (r + \gamma \times \max_{a'} Q(s', a'))$
    - $\alpha = 0.5$
    - $\gamma = 1$

策略 Policy

- 贪婪策略 greedy policy
  - 只选取最优行动
- ε-贪婪策略 ε - greedy policy
  - 概率1 - ε：根据现有Q值选取最优行动 (即Q值最大的行动)
  - 概率ε：选择一个随机的行动

代码框架

井字游戏

- 将问题抽象
  - 如何让计算机明白井字游戏？
- 按照解对抗搜索问题的步骤制作游戏AI
- 输出成果

尼姆游戏

- 将问题抽象
  - 如何让计算机明白尼姆游戏？
- 按照强化学习的步骤制作游戏AI
- 输出成果

nim.py 代码框架

- 环境 Environment
  - 类 Nim
    - 游戏相关的代码：游戏规则，每个状态可选的动作，转移模型等等
  - 函数play()
    - 运行游戏的相关代码
- 其他强化学习的要素
  - 类 NimAI
    - 制作一个可以通过奖励来学习如何玩尼姆游戏的AI （状态价值Value、策略Policy）
  - 函数train()
    - 通过运行多次游戏，训练NimAI (反馈合适的奖励reward)

类Nim

- 初始化 __init__()：每堆的物体各数，当前玩家，获胜方
- 可执行的行动 available_actions()
- 当前玩家的对手 other_player()
- 玩家换轮次 switch_player()
- 行动 move()
  - 将j个物体从第i堆中拿走(确认行动合法)
  - 玩家换轮次
  - 检查是否有玩家获胜

函数play()

- 确定玩家顺序
- 新建一个尼姆游戏
- 在终端中打出每堆的物品个数
- 人类的轮次→ 在终端中交互确认人类玩家的动作
- AI的轮次→ AI选择一个动作
- 执行动作
- 确认获胜者

类 NimAI

- 选择下一步行动 (策略 Policy)
  - ε - greedy
- 从经验中学习 (价值 Value)
  - $Q(s,a) \leftarrow (1-\alpha) \times Q(s,a) + \alpha \times (r +$ 未来得到的奖励估计$)$
  - $Q(s,a) \leftarrow (1-\alpha) \times Q(s,a) + \alpha \times (r + \max_{a'} Q(s',a'))$
- 初始化NimAI:
  - ε :我们选择随机行动的概率
  - q: $Q(s,a)$ (例., self.q[(0, 0, 0, 2), (3, 2)])　注意：这里的state，action都是元组tuple
  - α: 有多重视这一次游戏获得的新信息
- 函数available_actions()

练习 #4:

- 构建函数get_q_value():
  - 输入：state 列表，action 元组
  - 功能：
    - 如果字典q有记录，则返回在state下执行action的价值
    - 反之，返回0
  - 输出：数字

$$Q(s,a) \leftarrow (1-\alpha) \times Q(s,a) + \alpha \times (r + \text{未来得到的奖励估计})$$

练习 #5:

- 构建函数best_future_reward():
  - 输入：state 列表
  - 功能：
    - 根据字典q中记录的各个状态执行各种行动的价值，返回出在state下执行行动可能收获的最大值
    - 注：如果字典q没有记录相关信息，返回0
  - 输出：数字

$$Q(s,a) \leftarrow (1-\alpha) \times Q(s,a) + \alpha \times (r + \text{未来得到的奖励估计})$$

练习 #6：

- 构建函数update():
  - 输入：old state，action，在old state执行action到达的new state，获得的奖励reward
  - 功能：根据下面的公式更新Q(old_state, action)
  - 输出：无

$$Q(s,a) \leftarrow (1-\alpha) \times Q(s,a) + \alpha \times (r + \text{未来得到的奖励估计})$$

练习 #7：

- 构建函数choose_action():
  - 输入：state 列表，epsilon
  - 功能：
    - 如果epsilon为True
      - 概率1 - ε：根据现有只是选取最优行动 (即q(state,action)值最大的行动)
      - 概率ε：选择一个随机的行动
    - 如果epsilon为False，只选取最优行动
  - 输出：行动

函数train():

- 训练n次(玩n次)
- 记录每个玩家的最后一步
  - 如果最后一步的状态和最后一步的行动导致输掉游戏
    - 这一步的状态和行动获得奖励 -1
    - 上一步的状态和行动获得奖励 1
  - 如果游戏还没有决出胜负:
    - 这一步的状态和行动获得奖励 0

有问题吗?

- 请随时举手提问。

BUSS 3620.人工智能导论

#4. 函数逼近

刘佳璐

安泰经济与管理学院

上海交通大学

围棋

- 状态及行动都太多了
  - 很难每个都被探索到
  - $Q(s, a)$要储存的东西也太大了
- 可能的解决办法
  - 只尝试少部分状态
  - 将这部分学习到的知识应用到类似的状态中

函数逼近 Function Approximation

- 近似估算$Q(s, a)$,：通常通过学习一个以状态的各种特征为输入的函数，而非记录每个状态s-动作a的值
  - 与监督学习相似
  - 输入 – 输出对
    - 输入:各种特征
    - 输出: $Q(s, a)$
  - 可能的特征

Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari size	8	How many of own stones would be captured
Liberties after move	8	Number of liberties after this move is played
Ladder capture	1	Whether a move at this point is a successful ladder capture
Ladder escape	1	Whether a move at this point is a successful ladder escape
Sensibleness	1	Whether a move is legal and does not fill its own eyes
Zeros	1	A constant plane filled with 0
Player color	1	Whether current player is black

Source: Silver, D., Huang, A., Maddison, C. et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016). https://doi.org/10.1038/nature16961

函数逼近 Function Approximation

$x_1 = f_1(s,a)$ $x_2 = f_2(s,a)$ ... $x_n = f_n(s,a)$

$f(x_1, x_2, \dots, x_n) = Q(s,a)$ 真实的函数关系是未知的、隐藏的

$h(x_1, x_2, \dots, x_n)$ 假设的函数关系 Hypothesis function

如果我们假设假设函数(hypothesis function)是一个线性函数：

$$h(x_1, x_2, \dots, x_n) = w_1 \times x_1 + w_2 \times x_2 + \cdots + w_n \times x_n$$

我们可以使用感知机学习规则(perceptron learning rule)：

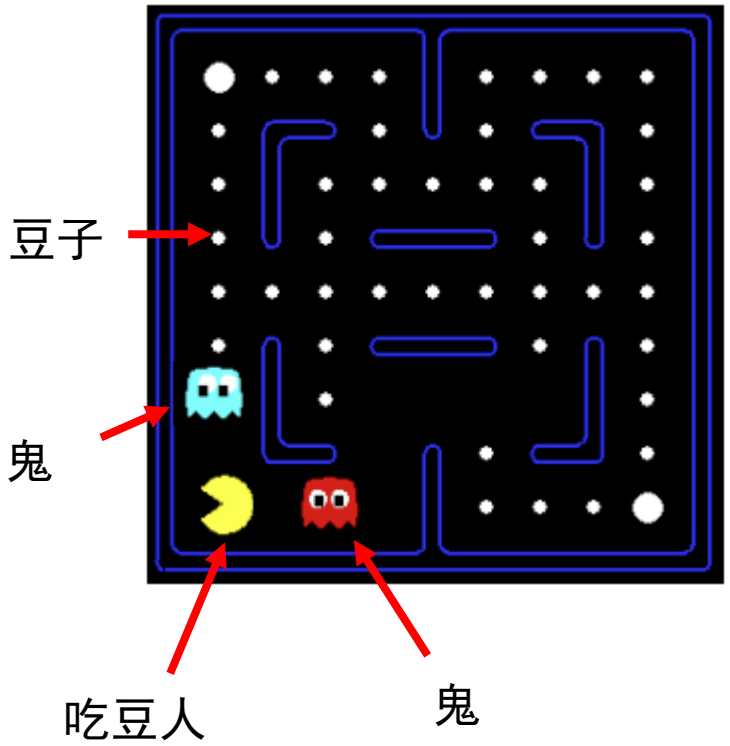
$$w_i = w_i + \alpha(y - h_w(\mathbf{x})) \times x_i$$

$$w_i = w_i + \alpha(\text{真实值} - \text{估计值}) \times x_i$$

$$w_i = w_i + \alpha(Q(s,a) - h_w(\mathbf{x})) \times x_i$$

练习 #8

- 吃豆人游戏
  - 吃到豆子，不能碰到鬼
  - 可以上下左右移动，或者不动，蓝色区域是墙
- 可以有哪些特征来表示状态的价值$Q(s, a)$？



练习 #9

- 我们用函数逼近+Q-learning的方法学习吃豆人游戏
- 每个状态有两个特征
- $F_g(s,a) = A(s) + B(s,a) + C(s,a)$

1步以内有几个鬼

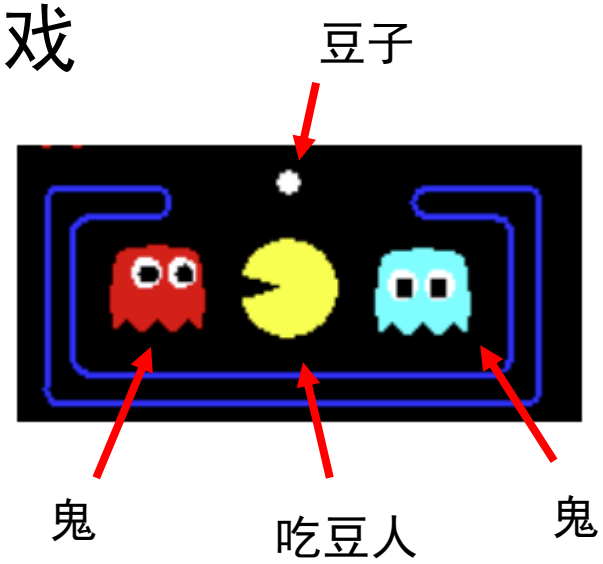
在s执行了a后，新状态下1步以内有几个鬼

在s执行了a后，吃豆人是否碰到鬼

- $F_p(s,a) = D(s) + 2E(s,a)$

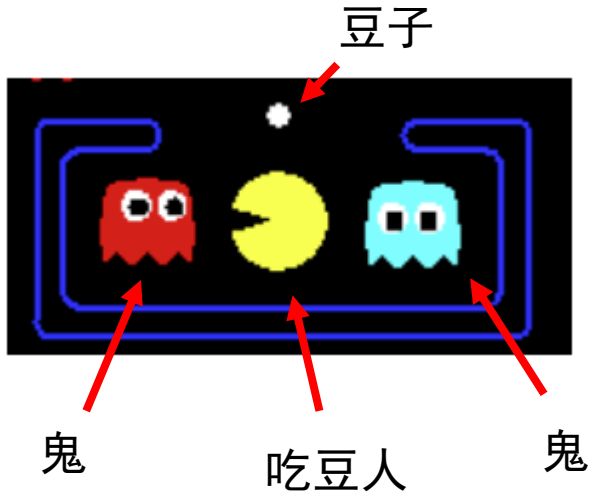
1步以内有几个豆子

在s执行了a后，吃豆人是否吃到豆子



① 当前状态下，这个状态每个行动{up,left,right,stay}的每个特征的值是？

练习 #9



- 我们用函数逼近+Q-learning的方法学习吃豆人游戏

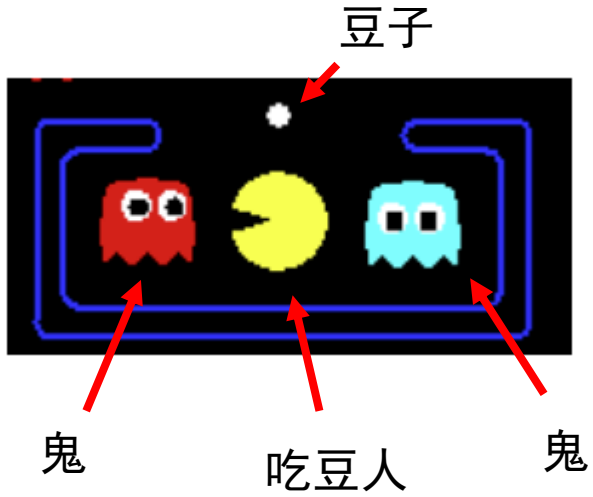
② 假设函数 $h(F_g(s,a),F_p(s,a)) = w_g \times F_g(s,a) + w_p \times F_p(s,a)$ 经过一段时间的学习，目前 $w_g = -10, w_p = 100$

当前状态下，我们估计的

$F_g(s,\text{up})$	2+0+0=2	$F_p(s,\text{up})$	1+2×1=3
$F_g(s,left)$	2+1+1=4	$F_p(s,left)$	1+2×0=1
$F_g(s,right)$	2+1+1=4	$F_p(s,right)$	1+2×0=1
$F_g(s,stay)$	2+0+2=4	$F_p(s,stay)$	1+2×0=1

这个状态每个行动{up,left,right,stay}的$Q(s,a)$是？

练习 #9



- 我们用函数逼近+Q-learning的方法学习吃豆人游戏

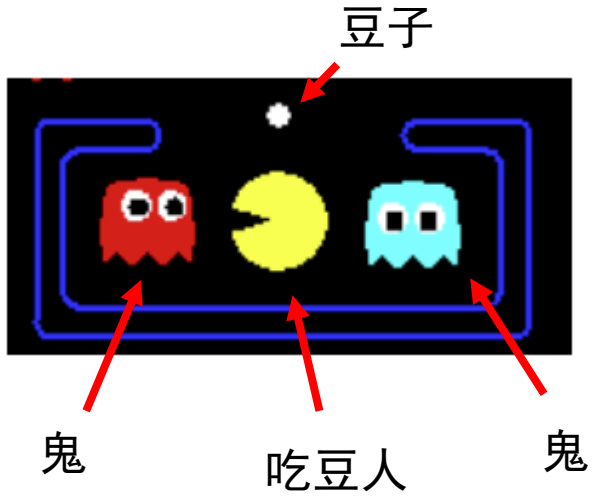
③ 假设吃豆人在这个状态执行了up的行动，获得了奖励250。

新状态可以执行的行动是{down, stay}。计算当前状态的

新的$Q(s, a)$(new estimate)

- 假设$\gamma = 0.5$ $Q_{new}(s,a) = r + \gamma \times \max_{a'} Q(s', a')$

练习 #9



- 我们用函数逼近+Q-learning的方法学习吃豆人游戏

④ 采用感知机学习规则，更新w_g, w_p

- 假设学习速率$\alpha = 0.5$

$F_g(s, \text{up})$	2+0+0=2	$F_p(s, \text{up})$	1+2×1=3
$F_g(s, left)$	2+1+1=4	$F_p(s, left)$	1+2×0=1
$F_g(s, right)$	2+1+1=4	$F_p(s, right)$	1+2×0=1
$F_g(s, stay)$	2+0+2=4	$F_p(s, stay)$	1+2×0=1

$Q(s, \text{up})$	-10×2+100×3=280
$Q(s, left)$	-10×4+100×1=60
$Q(s, right)$	-10×4+100×1=60
$Q(s, stay)$	-10×4+100×1=60

有问题吗?

- 请随时举手提问。