法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

关注 小象学院

第1讲　概述与环境搭建

强化学习

主讲人：叶梓

上海交通大学博士

主要研究方向：机器学习、深度学习、人工智能

本章内容

- 强化学习的应用
- 强化学习的发展历程
- 强化学习中的基础概念
- 强化学习方法的分类
- 主要研究问题（后续各章节介绍）
- 实验环境介绍
- 著名学者
- gym环境搭建演示

AlphaGo战胜李世石

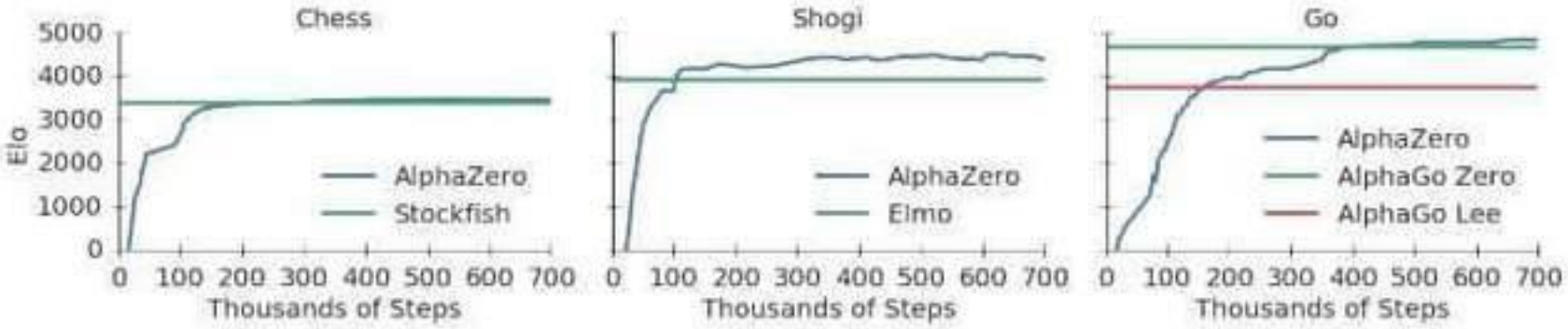
2016年3月5日，AlphaGo战胜李世石，人机大战总比分1:4

面对AlphaGo Master，柯洁哭了

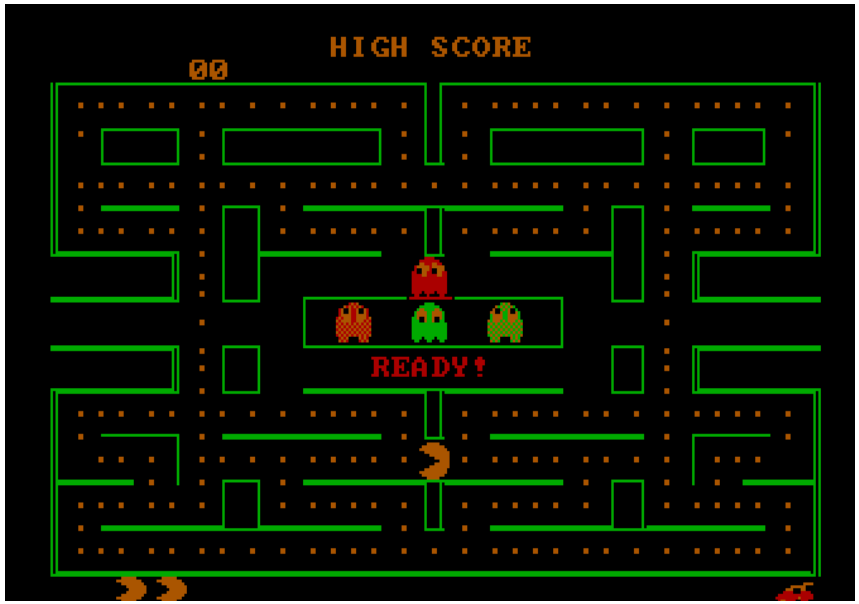
2017年5月27日，升级的AlphaGo大战“围棋第一人”柯洁，3:0 完胜。

AlphaGo Zero和Alpha Zero

- AlphaZero通过使用与AlphaGo Zero一模一样的方法，从零开始训练：
  - 4小时就打败了国际象棋的最强程序Stockfish！
  - 2小时就打败了日本将棋的最强程序Elmo！
  - 8小时就打败了与李世石对战的AlphaGo v18！

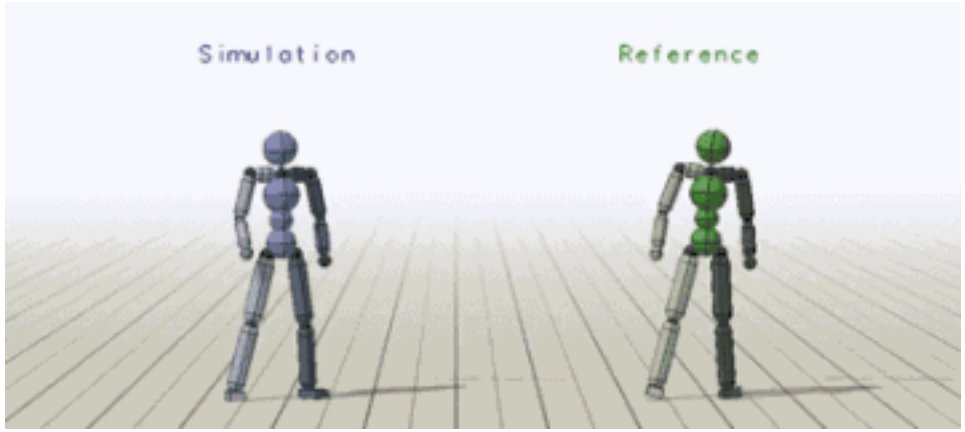


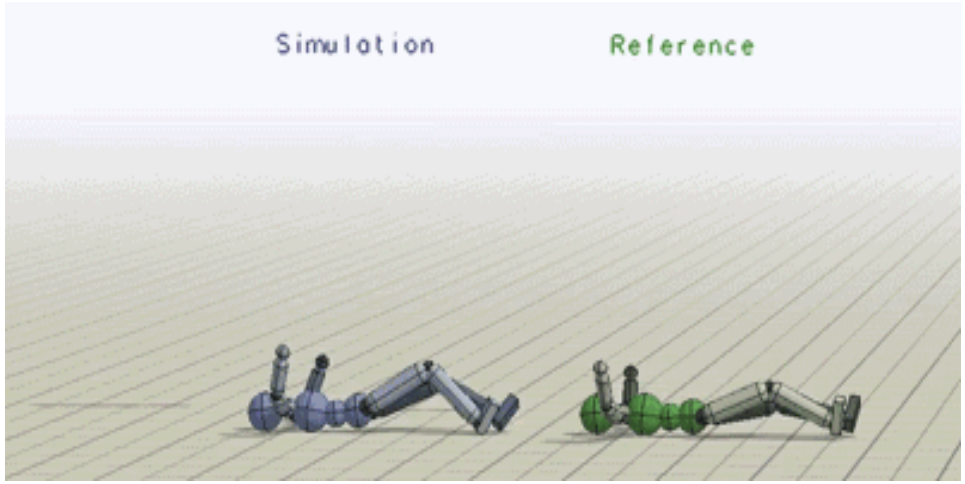
玩ATARI的游戏

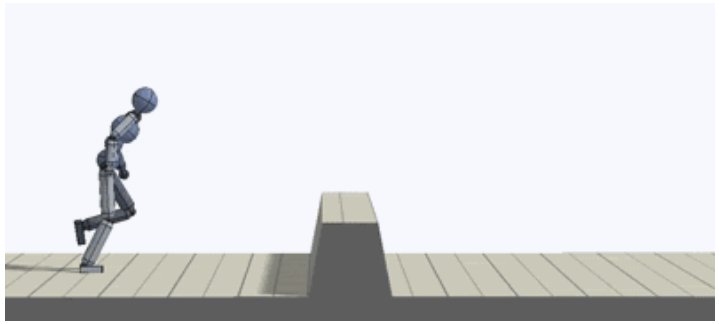


玩《星际争霸2》

Deep mimic实现的案例导向拟人动作







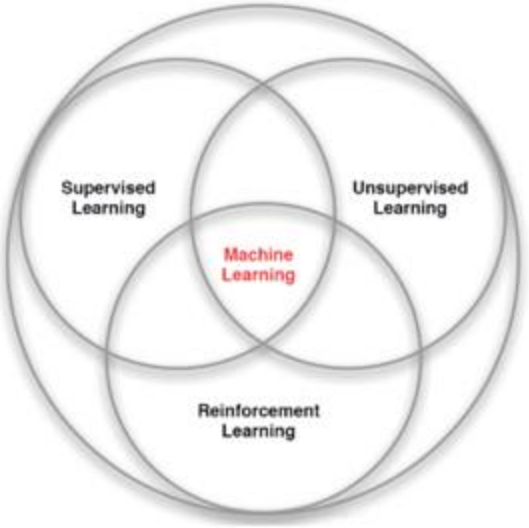
波士顿动力……暂时不是！

强化学习的发展历史

- 1954年，Minsky：提出强化学习的概念和术语。
- 1956年，Bellman：MDP的动态规划方法。
- 1977年，Werbos：自适应动态规划算法。
- 1988年，Sutton：时序差分算法。
- 1992年，Watkins：Q-learning 算法。
- 1994年，rummery：Sarsa算法。
- 2006年，Kocsis：置信上限树算法。
- 2009年，kewis：反馈控制自适应动态规划算法。
- 2014年，Silver：确定性策略梯度（DPG）算法。
- 2015年，Google deep mind：DQN算法。

强化学习与机器学习

- 强化学习是智能体（Agent）以“试错”的方式进行学习，通过与环境进行交互获得的奖励指导行为，目标是使智能体获得最大的奖励。
- 机器学习可以分为三类，分别是：
  - supervised learning
  - unsupervised learning
  - reinforcement learning

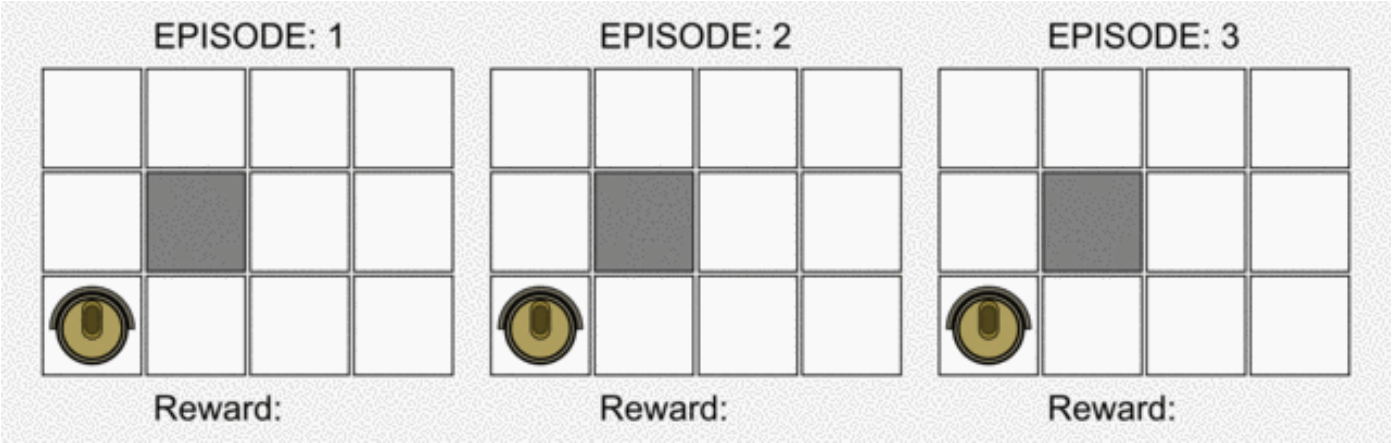


强化学习概念

- 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Reward（奖励值）最大。
- 在强化学习中，算法来把外界环境转化为最大化奖励量的方式的动作，算法并没有直接告诉Agent（行为主体）要做什么或者要采取哪个动作，而是Agent通过看哪个动作得到了最多的奖励来自己发现。
- Agent的动作的影响不只是立即得到的奖励，而且还影响接下来的动作和最终的奖励。

强化学习的特点

- 强化学习与其他机器学习不同之处为：
  - 没有教师信号，也没有label，只有reward。
  - 反馈有延时，不是能立即返回。
  - 数据是序列化的，数据与数据之间是有关的，而不是i.i.d的；
  - agent执行的动作会影响之后的数据。

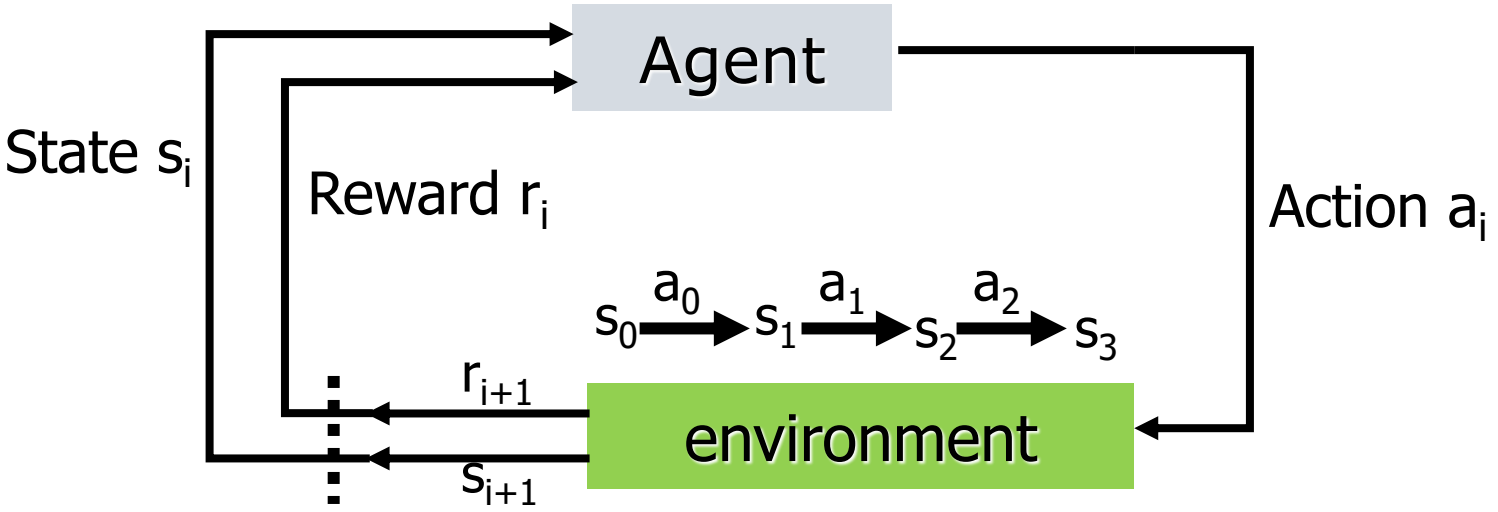


强化学习的关键要素

- 强化学习的关键要素有：environment，reward，action 和 state。有了这些要素我们就能建立一个强化学习模型。
- 强化学习解决的问题是，针对一个具体问题得到一个最优的policy（策略），使得在该策略下获得的return（长期回报）最大。
- 所谓的policy其实就是一系列action，也就是sequential data。

强化学习的模型

- 强化学习的模型如图所示：

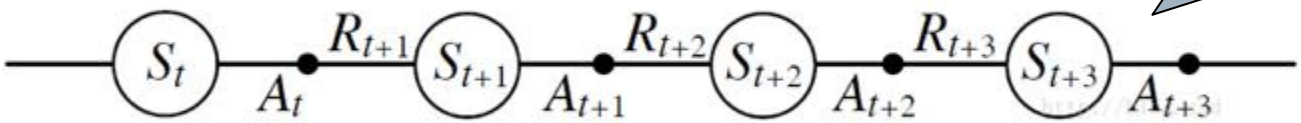


- Agent与环境的交互接口包括行动(Action)、即时奖励(Reward)和状态(State)。

Agent与环境的交互

- 交互过程可更准确的表述如下：

注意！在状态S_i时，执行A_i动作，然后获得r_{i+1}回报，然后到S_{i+1}状态



- 每一步，Agent根据Policy（策略）选择一个行动执行，然后感知下一步状态和即时回报，通过经验再修改自己的策略。
- Agent的目标就是最大化长期回报(return)，仅可考虑即时奖励(reward)是显然不够的。

奖励与动作

reward

- reward通常都被记作R_t，表示第t个time step的返回奖励值。所有强化学习都是基于reward假设的。
- reward是一个标量。
- 注意：回报（return）是奖励（reward）的累积。

action

- action是来自于动作空间，agent对每次所处的state用以及上一状态的reward确定当前要执行什么action。
- 执行action要达到最大化期望reward，直到最终算法收敛，所得的policy就是一系列action的sequential data。

状态与策略

- **state**
  - 就是指当前agent所处的状态。
- **policy**
  - policy就是指agent在特定状态下的行为依据，是从state到action的映射。
  - 分为确定策略和与随机策略。
  - 确定策略：就是某一状态下的确定动作$a=\pi(s)$
  - 随机策略：以概率来描述，即某一状态下执行这一动作的概率：$\pi(a|s)=P[A_t=a \mid S_t=s]$。

策略有两种

- 把用来指导个体产生与环境进行实际交互行为的策略称为：行为策略；
  - 实际采样的策略
- 把用来评价状态或行为价值的策略（或者待优化的策略）称为：目标策略。

强化学习的学习过程

- RL采用的是边获得样例边学习的方式
  - 在获得样例之后更新自己的模型，
  - 利用当前的模型来指导下一步的行动，
  - 下一步的行动获得reward之后再更新模型，
  - 不断迭代重复直到模型收敛。

- 在这个过程中，非常重要的一点在于“在已有当前模型的情况下，如果选择下一步的行动才对完善当前的模型最有利”。

多臂老虎机

- 多臂老虎机是一个有多个拉杆的赌博机，每一个拉杆的中奖几率是不一样的，问题是：如何在有限次数内，选择拉不同的拉杆，获得最多的收益。
- 设老虎机有N个拉杆，最笨的方法就是每个拉杆都试几次，找到中奖概率最大的那个拉杆，然后把之后有限的游戏机会都用在这个拉杆上。

探索与利用

- 在有限次数下，到底是坚持在你认为中奖概率高的拉杆上投入更多的次数(Exploit)呢？还是去试试别的拉杆(Explore)呢？
- 如何分配Explore和Exploit的次数的问题，就是著名的**探索-利用困境(Explore-Exploit dilemma(EE dilemma))**。
  - exploration是指选择之前未执行过的actions，从而探索更多的可能性；
  - exploitation是指选择已执行过的actions，从而对已知的actions的模型进行完善。

强化学习方法的分类

- 理解环境 or 感知环境
- 回合更新 or 单步更新
- 基于价值 or 基于策略
- 同策略 or 异策略

理解环境 or 感知环境

- Model-based：先理解真实世界是怎样的，并建立一个模型来模拟现实世界的反馈，通过想象来预判断下来将要发生的所有情况，然后选择这些想象情况中最好的那种，并依据这种情况来采取下一步的策略。

- Model-free：不依赖环境，不尝试去理解环境，Agent会根据现实环境的反馈采取下一步的动作，一步一步等待真实世界的反馈，再根据反馈采取下一步行动。
  - 例如Q-learning，Sarsa，Policy Gradients。

回合更新 or 单步更新

- Monte-carlo update：游戏开始后，要等待游戏结束，然后再总结这一回合中的所有转折点，再更新行为准则。
  - 例如：policy gradients，MC

- Temporal-difference update：在游戏进行中每一步都在更新，不用等待游戏的结束，这样就能边玩边学习了。
  - 例如：Q-learning，Sarsa，升级版的PG.

基于价值 or 基于策略

- Policy based：目标是找到最优策略，通过感官分析所处的环境，直接输出下一步要采取的各种动作的概率，然后根据概率采取行动。
  - 例如：Policy Gradients
- Value based：目标是找到最优奖励总和，输出的是所有动作的价值，根据最高价值来选动作，这类方法不能选取连续的动作。
  - 例如：Q-learning，Sarsa
- 还有一种名为Actor-Critic的算法：它结合了这两类方法的优势之处。

同策略 or 异策略

- on-policy: 必须Agent“本人”在场，并且一定是Agent边玩边学习，例如Sarsa，Sarsa(λ)，TRPO。
- off-policy: 可以选择自己玩，也可以选择看着别人玩，通过看别人玩来学习别人的行为准则，例如Q-learning，DQN，Deterministic policy gradient。
- on-policy和off-policy本质区别在于：更新Q值的时候是使用既定策略还是使用新的策略。

异策略的特点

- 可以从人类给出的示教样本或其他智能体给出的引导样本中学习；
- 可以重用由旧策略生成的经验；
- 可以在使用一个探索性策略的同时，学习一个确定性策略；
- 可以用一个策略进行采样，然后同时学习多个策略。

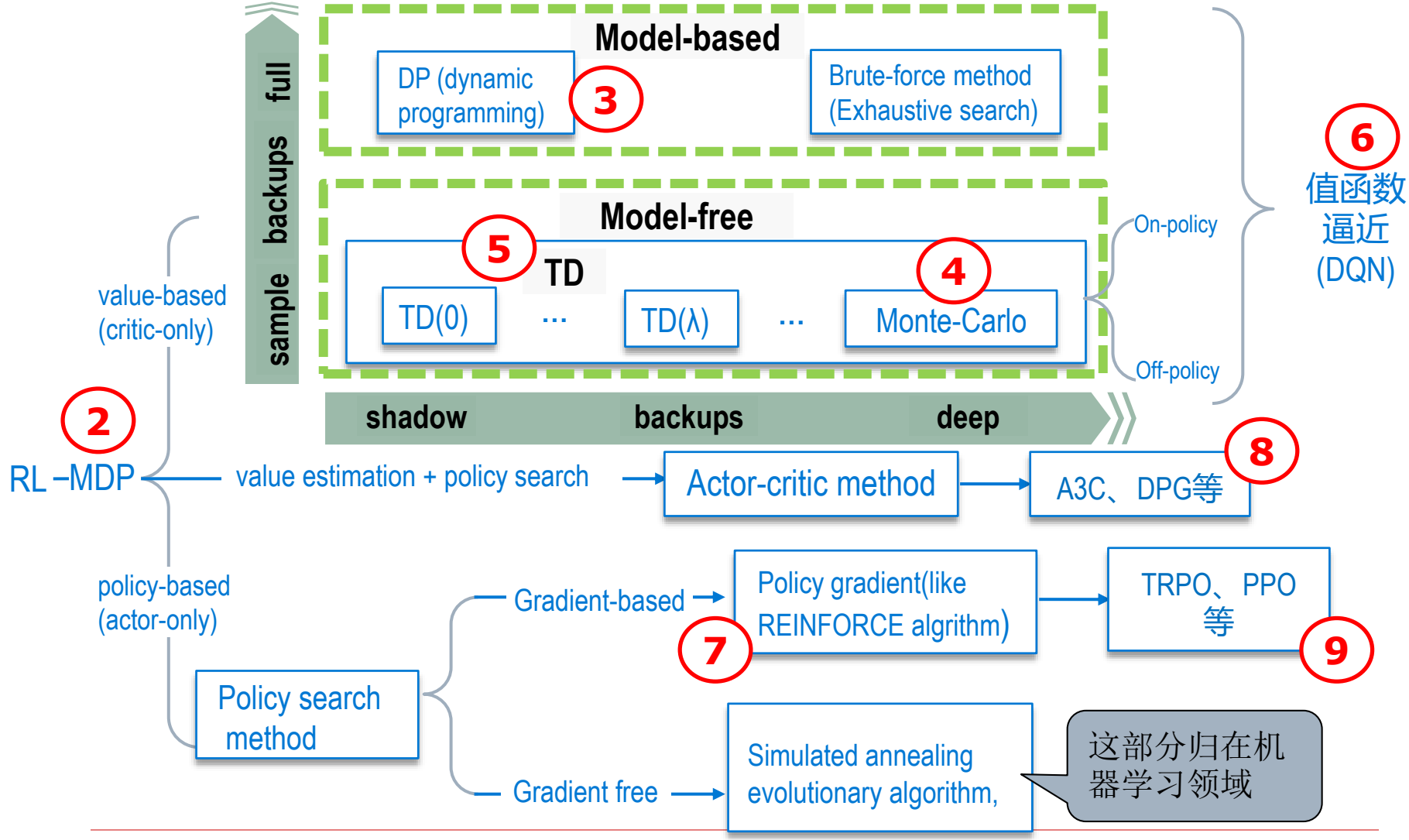
预测与控制

- 预测：给定某个策略，估计该策略将能得到多少奖励；
- 控制：找到一个最优的策略。

- 在RL算法中，通常都是迭代地进行先预测、再控制的过程，直到收敛。

RL学习路线图

10 多Agent强化学习



常用强化学习实验环境

- 和其它的机器学习方向一样，强化学习也有一些经典的实验场景，像Mountain-Car，Cart-Pole等。
- 另外，由于近年来深度强化学习（DRL）的兴起，各种新的更复杂的实验场景也在不断涌现，出现了一系列优秀的平台。
- 常见的一些强化学习实验平台：
  - OpenAI Gym、OpenAI Baselines
  - MuJoCo、rllab、TORCS、PySC2、

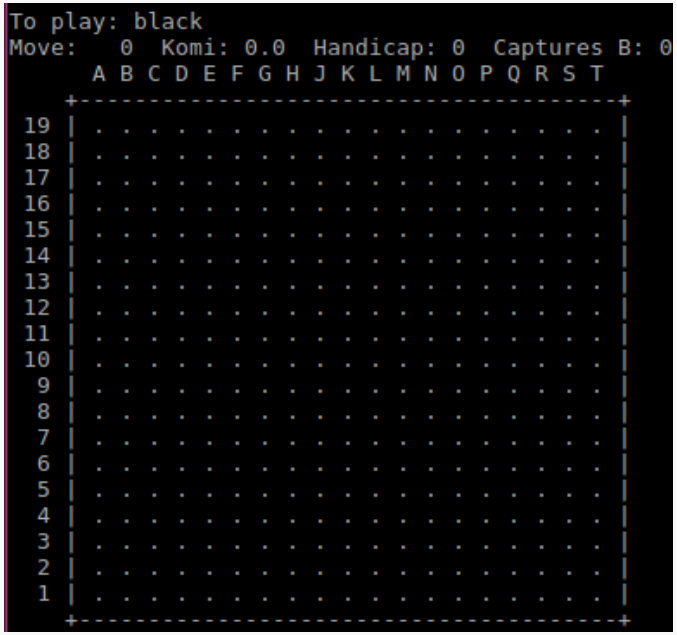
常用实验环境 OpenAI Gym

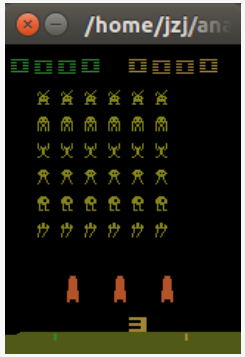
- OpenAI Gym是OpenAI出的研究强化学习算法的toolkit，它覆盖的场景非常多，从经典的Cart-Pole, Mountain-Car到Atari，Go，MuJoCo都有。
- 官方网站为https://gym.openai.com/，源码位于https://github.com/openai/gym，它的readme提供了安装和运行示例，linux下的安装方法：

```
apt-get install -y python-numpy python-dev cmake zlib1g-dev libjpeg-dev xvfb libav-to
pip install -e '.[all]'
```

常用实验环境 OpenAI Gym

- 然后可以跑readme中的例子，如SpaceInvaders, Go, LunarLander, CarPole, MuJoCo等等；
- 通过源码下的examples/scripts/list_envs可以列出所有支持的场景。





常用实验环境 OpenAI Gym

- 先装Anaconda的Python环境，
- 然后pip install gym[all] 即可

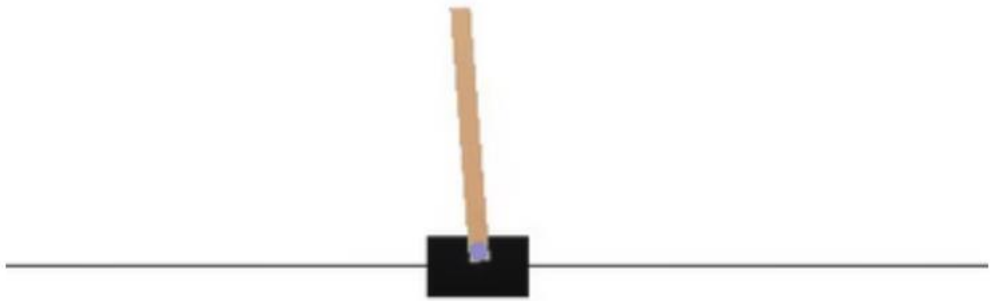
```
import gym
import time

env = gym.make('CartPole-v0')
for i_episode in range(10):
    observation = env.reset()
    time.sleep(0.1)
    for t in range(1000):
        env.render()  # 环境展示
        #time.sleep(0.1)
        print(observation)
        action = env.action_space.sample()  # 随机从动作空间中选取动作
        observation, reward, done, info = env.step(action)  # 根据动作获取下一步的信息
        if done:
            print("Episode finished after {} timesteps".format(t+1))
            break

quit()
```

重要的函数

- env = gym.make('CartPole-v0')
  - 创建CartPole环境
- env.reset()
  - 在强化学习算法中，智能体需要一次次地尝试
  - 一次尝试结束后，智能体需要从头开始，这就需要智能体具有重新初始化的功能。
- env.render()
  - 图像引擎用来显示环境中的物体图像，便于直观显示当前环境物体的状态。
- env.step()
  - 该函数描述了智能体与环境交互的所有信息。
  - 其输入是动作a，输出是：下一步状态，立即回报，是否终止，调试项。

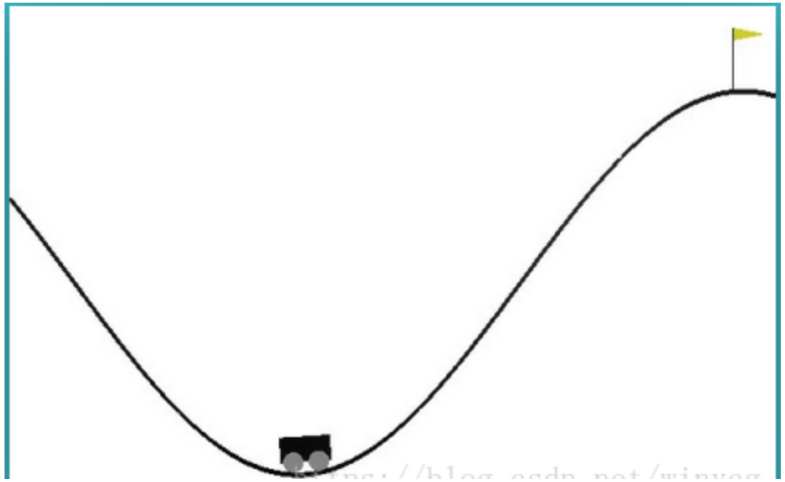
Cart-Pole

- 环境描述：运载体无摩擦地支撑杆子。
- 动作：2个动作：施加-1和+1分别对应向左向右推动运载体
- 状态：4个，x：位置;x_dot：移动速度, theta：角度 theta_dot：移动角速度
- 奖励：每一步杆子保持垂直就可以获得+1奖励
- 终止条件：杆子的摇摆幅度超过了垂直方向15度或运载体偏移初始位置超过2.4个单位



Mountain-Car

- 环境描述：car的轨迹是一维的，定位在两山之间，目标是爬上右边的山顶。可是car的发动机不足以一次性攀登到山顶，唯一的方式是car来回摆动增加动量。
- 动作：有3个动作：向前、不动和向后
- 状态：2个，位置position和速度velocity。position的值在最低点处为-0.5左右，左边的坡顶为-1.2，右边与之相对应的高度位置为0，小黄旗位置为0.6。
- 奖励：每次移动都会得到-1的奖励，直到车开到黄旗位置



常用实验环境 OpenAI Baselines

- 严格来说Baselines本身不是一个开发环境。它是OpenAI出的一些深度强化学习算法（DQN, PPO, TRPO, DDPG）的实现。
- 基于TensorFlow和OpenAI Gym，最新版需要Python 3。
- 源码位于：https://github.com/openai/baselines。按readme中使用下面命令安装：

```
pip install baselines
```

- 除了列出来的依赖，还可能依赖下面的库，使用conda或pip安装：

```
conda install Pillow atari-py
```

常用实验环境 MuJoCo

- MuJoCo（Multi-Joint dynamics with Contact）是一个物理模拟器，可以用于机器人控制优化等研究。

- 官方网站为http://www.mujoco.org/index.html。最新版本为1.50。
- 注意：要用MuJoCo是需要license的，可以在https://www.roboti.us/license.html上申请试用版30天免费license。

常用实验环境 MuJoCo

- OpenAI对MuJoCo引擎做了Python 3的binding-mujoco-py，源码位于https://github.com/openai/mujoco-py。
- 按readme，可以通过下面命令安装：

```
pip3 install -U 'mujoco-py<1.50.2,>=1.50.1'
```

- 如果在后面有其它项目依赖到更高版本（如1.50.1），可以从官方release页面下载源码包（https://github.com/openai/mujoco-py/releases），然后用上面方法安装即可。

常用实验环境 rllab

- 和OpenAI Gym类似，rllab也是一个研究强化学习算法的框架。官方网站为 https://github.com/openai/rllab 。官方支持python 3.5+，基于Theano。
- rllab自己也提供一个基于pygame的可视环境，同时它也可兼容OpenAI Gym。
- 除此之外，它提供了一些强化学习算法的实现，这些参考实现和一些组件可以使得强化学习算法的开发更快上手。
- 安装步骤可按照官方网站：https://rllab.readthedocs.io/en/latest/user/installation.html 。

常用实验环境 TORCS

- TORCS（The Open Racing Car Simulator）是一个跨平台的赛车游戏模拟器，也可作为强化学习的研究平台。官方网站：http://torcs.sourceforge.net/ 。
- gym_torcs是一个TORCS的强化学习环境，提供类似前面OpenAI Gym的接口，网站为https://github.com/ugo-nama-kun/gym_torcs 。
- 然后进入其vtorcs-RL-color子目录，按其中readme编译安装定制版torcs。
- 安装完了运行torcs命令就能看到界面了。

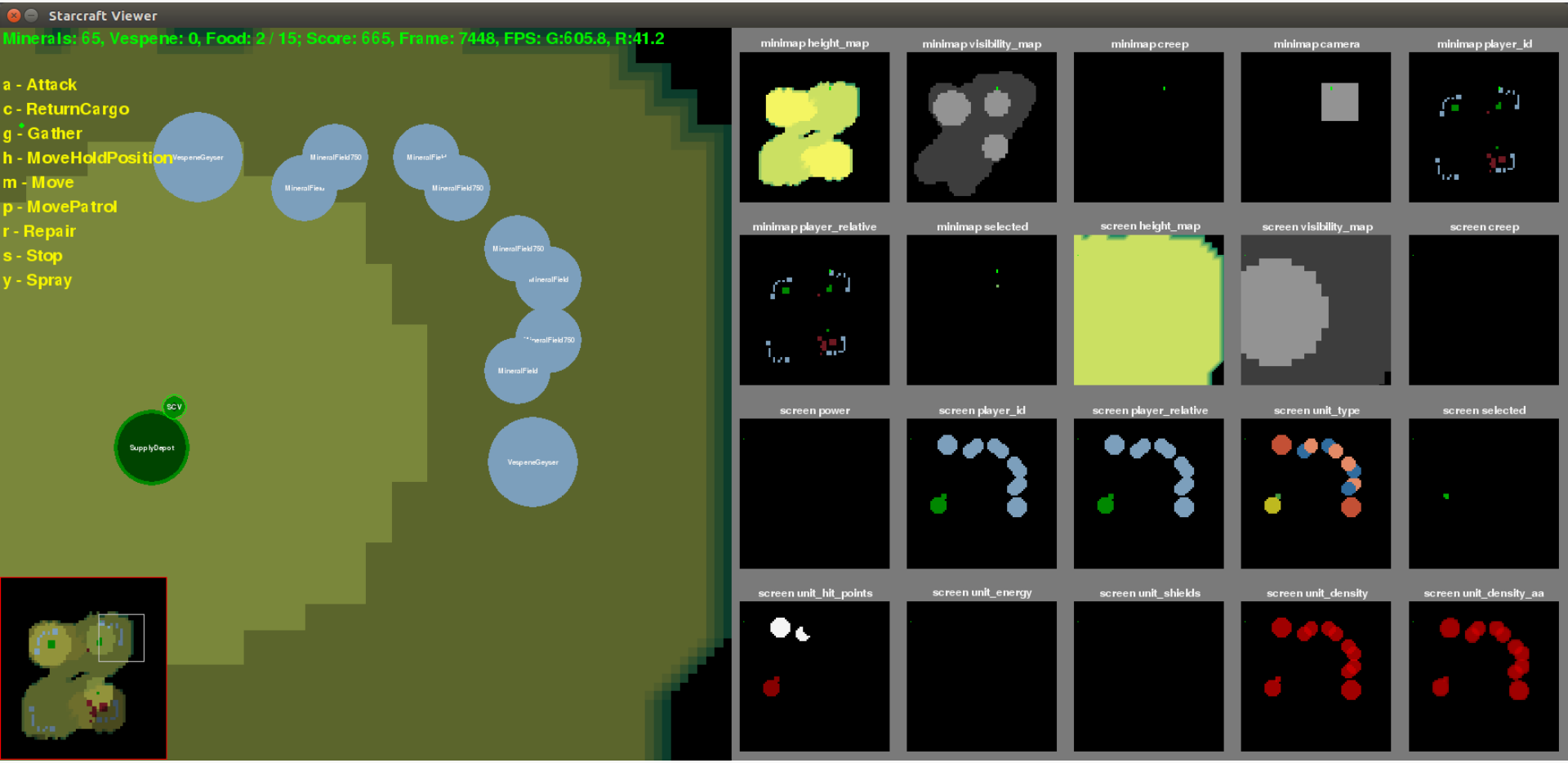
常用实验环境 TORCS

- Readme中的Simple How-To演示了如何在Python中与该环境交互，然后就可以开发测试强化学习算法了。
- 有个实现DDPG算法的例子可以参考：https://yanpanlau.github.io/2016/10/11/Torcs-Keras.html

常用实验环境 PySC2（StarCraft II）

- DeepMind和Blizzard合作出了个StarCraft II的研究平台，称为PySC2。
  - 介绍网站：https://deepmind.com/blog/deepmind-and-blizzard-open-starcraft-ii-ai-research-environment/ 。
  - 论文：https://deepmind.com/documents/110/sc2le.pdf 。
- PySC2是基于Blizzard的StarCraft II Machine Learning API（https://github.com/Blizzard/s2client-proto ）的Python下强化学习环境。
- 源码位于：https://github.com/deepmind/pysc2 。
- 按照readme最简单的安装方法就是：`pip install pysc2`
- 注意：还要安装StarCraft II游戏。

常用实验环境 PySC2（StarCraft II）

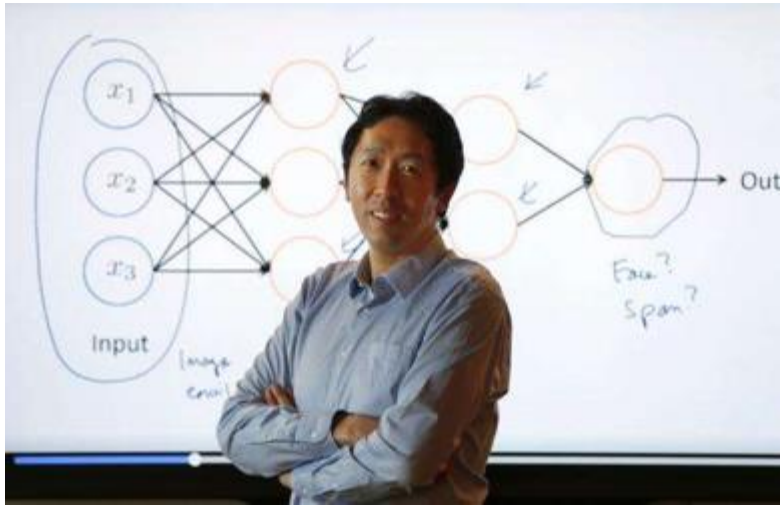


RL教父：Richard Sutton

- Richard S. Sutton 教授被认为是现代强化学习理论的创立者之一。
- 他为该领域做出了许多重大贡献，包括：时间差分学习、策略梯度方法、Dyna 架构。

- 对于强化学习的初学者，《Reinforcement Learning : An Introduction》可能就是最佳选择。

著名学者



- 吴恩达是国际上人工智能和机器学习领域最权威的学者之一。
- 是DL和RL两个领域的“双料大牛”。

- Pieter Abbeel，强化学习领域的“大牛”。
- 2016年4月，Abbeel正式离开伯克利加入OpenAI，从顾问转为全职研究员。

著名学者

- David Silver：DeepMind AlphaGo项目首席研究员
  - DPG的提出者
  - 强化学习公开课
- Demis Hassabis：AlphaGo之父、DeepMind联合创始人兼CEO
  - DQN的提出者

推荐教材

- Richard S. Sutton, Andrew G. Barto:《Reinforcement Learning An Introduction》
  - Sutton在2012年Release更新之后的第二版
  - Silver的教程是基于此书的
- Csaba Szepesvari：《Algorithms for Reinforcement Learning》

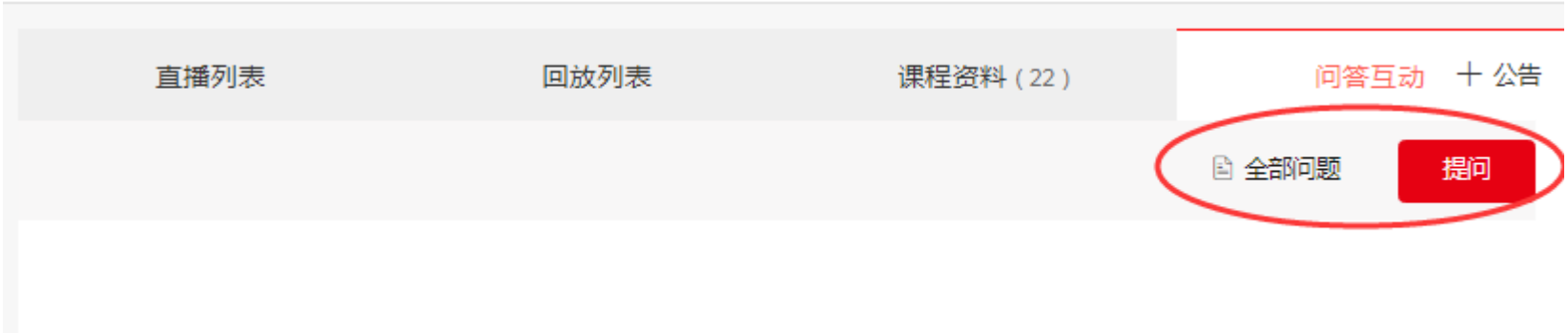
- 郭宪,方勇纯：《深入浅出强化学习:原理入门》
- 冯超：《强化学习精要:核心算法与TensorFlow实现》

参考资料

- https://blog.csdn.net/trillion_power/article/details/70992333
- https://blog.csdn.net/coffee_cream/article/details/57085729
- https://www.analyticsvidhya.com/blog/2017/01/introduction-to-reinforcement-learning-implementation/?winzoom=1

问答互动

在所报课的课程页面，
1、点击“全部问题”显示本课程所有学员提问的问题。
2、点击“提问”即可向该课程的老师和助教提问问题。



联系我们

小象学院：互联网新技术在线教育领航者

– 微信公众号：**小象学院**