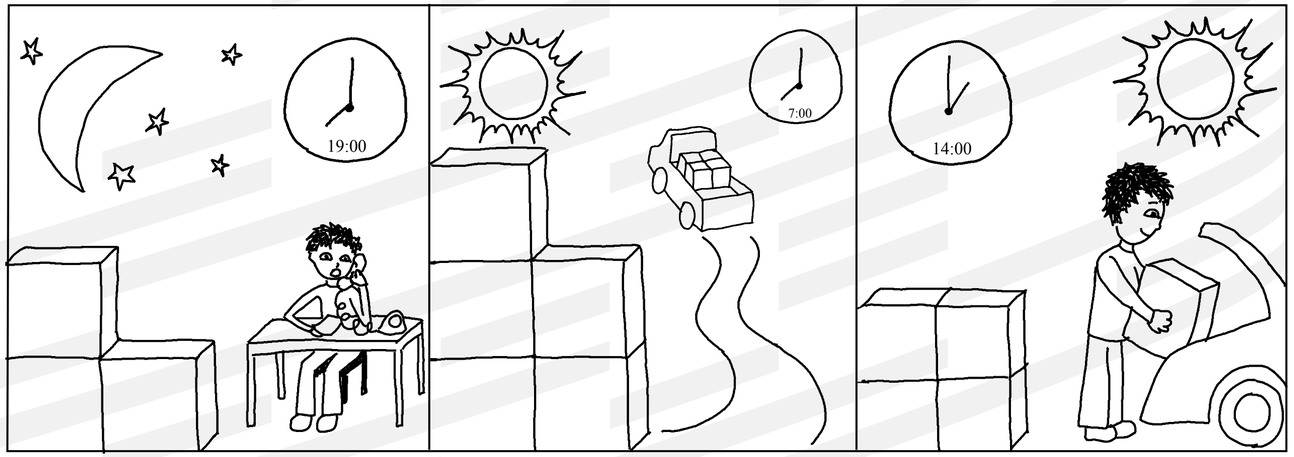
Introduction to Reinforcement Learning

Example 1 : Inventory Management

- Consider a trader with a large inventory, storage has a cost proportional to size of stock
- Each day, the trader is faced with the following choices
  A. Order N items to add the current stock
  B. Do nothing, i.e. keep current stock
- Trader is paid money daily based on sales.
- Demand randomly changes every day
- **Task:** Design a strategy for the trader to follow each day to maximize long term gain.

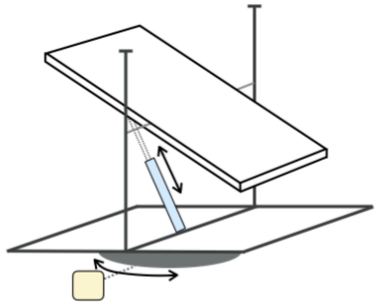


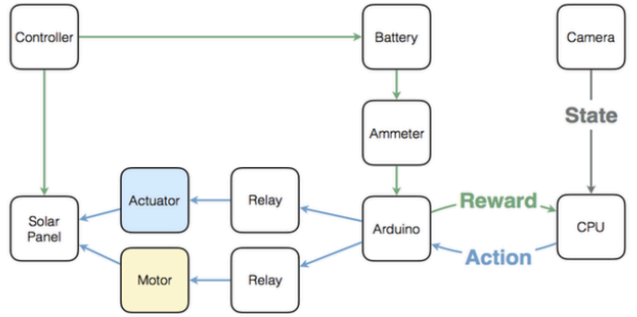
IC: Csaba Szepesv´ari, Algos for RL

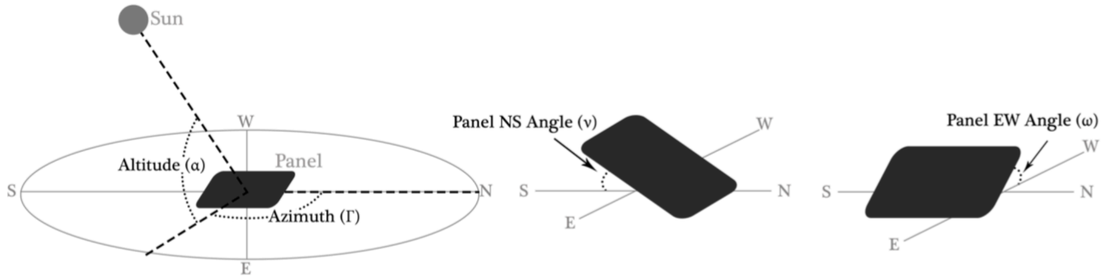
Example 2 : Solar Panel Efficiency

- We have a solar panel, mounted as shown on the right.
- Every X mins, we have the choice to **tilt** or **rotate** the panel or simply do **nothing**.
- Each rotation or tilting uses energy, goal is to maximize output for, e.g home usage.
- The sky may be covered with clouds at some points. We have a camera to 'perceive' cloud cover.
- **Task:** Design a strategy for manipulating the panel to maximize energy output.

IC: https://www.redarc.com.au



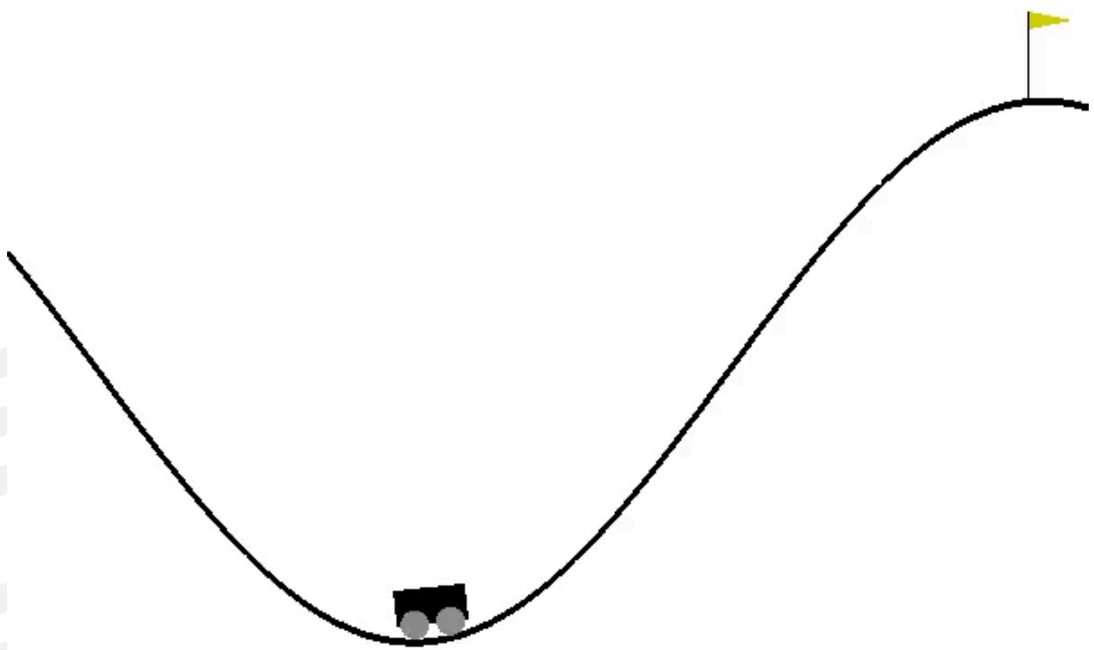




IC: Abel et al. Improving Solar Panel Efficiency using Reinforcement Leaning

Example 3 : Mountain Car (Discrete Actions)

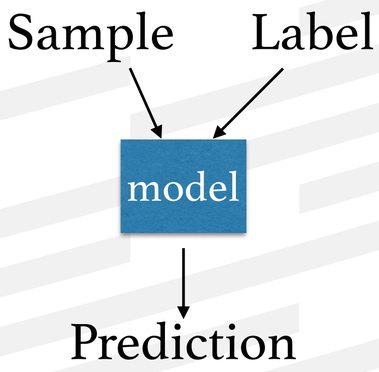
- A car travels in a one dimensional track, using a ‘weak’ engine.
- Each time step, the car can try three actions
  - Move left (i.e. reverse)
  - Move right (i.e. drive forward)
  - Nothing, i.e. sit still
- Each move costs a penalty of -1.
- We know car position and velocity every time. Starts always a the bottom.
- **Task**: Design a strategy (sequence of actions) to get the car to to flag.

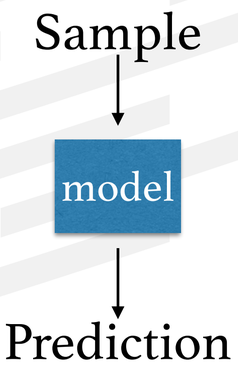


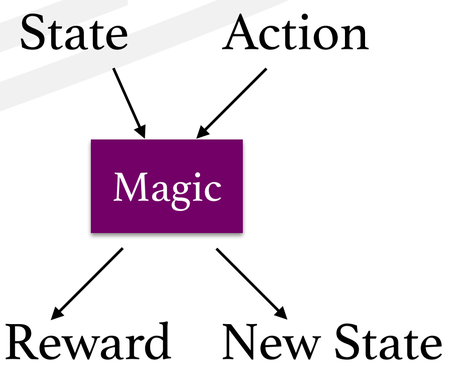
IC: https://gym.openai.com/envs/MountainCar-v0/

Recap

	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Data Property	i.i.d	i.i.d	non-i.i.d, Correlated
Targets/Labels	Available	None	Sparse, delayed
Generalization	New samples	New samples	New experiences / environments





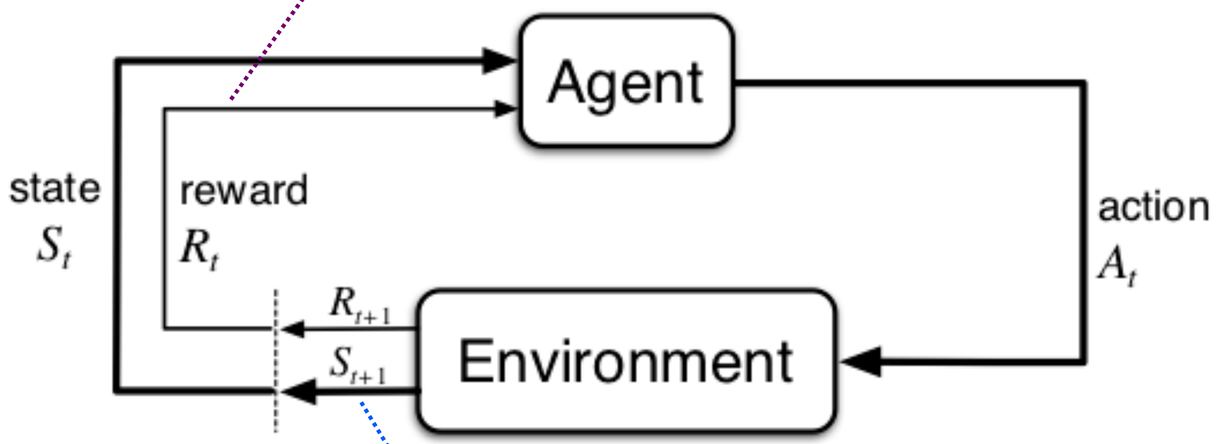


Reinforcement Learning — Formulation (2)

- RL is a paradigm for learning to make a **good sequence** of **decisions.**
  - The algorithm (cf. agent) learns by interacting with its environment
  - Agent learns without human intervention, i.e it simply maximizes rewards
- Reward is received only after taking an action, which causes a state transition.

$$\Pr(R_t = r \mid S_t = s, A_t = a)$$

Reward function



$$\Pr(S_{t+1} = s' \mid S_t = s, A_t = a)$$

Dynamics (transition) function

Reinforcement Learning — Formulation (1)

Interaction Protocol

- Collect a sequence of rewards. Sum = return.

$$G = R_{t+0} + R_{t+1} +, \ldots, + R_{t+H}$$

- Rewards maybe discounted

$$G = \gamma^0 R_{t+0} + \gamma^1 R_{t+1} +, \ldots, + \gamma^H R_{t+H}$$

  - ‘Inflation’ type scaling of future rewards

- Value of state — expected return

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G \mid S_t = s]$$

- Policy — strategy for picking actions. Optimality?

$$\pi : \mathcal{S} \times \mathcal{A} \longrightarrow [0, 1]$$

```
import gym

env = gym.make('MountainCar-v0')
env.reset()
policy = load_policy('PATH_TO_POLICY')

for i in range(num_episodes):
   reward_sum = 0.0  # G_i
   state = env.reset()
   while not finished:
      action = policy(state)
      new_state, reward, _,_ = env.step(action)
      reward_sum += reward  # * gamma

print("Total rewards: {}".format(reward_sum))
```

Reinforcement Learning — Tasks

- Policy Evaluation — find the expected return of a policy
  - Simple idea: Run policy many times and average — Monte Carlo
- Value and/or Policy Search
- Control

```
import gym

env = gym.make('MountainCar-v0')
env.reset()
policy = load_policy(‘PATH_TO_POLICY’)
s0 = Some state.
v_pi_s0_samples = []
for i in range(num_episodes):
   state = env.reset()
   if state == s0:
      reward_sum = 0.0  # G_i
      while not finished:
        action = policy(state)
        new_state, reward, _,_ = env.step(action)
        reward_sum += reward  # * gamma
      v_pi_s0_samples.append(reward_sum)

v_pi_s0 = average(v_pi_s0_samples)
```

Policy evaluation sketch for single state.

Q Learning

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

Recall

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G \mid S_t = s]$$

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G \mid S_t = s, A_t = a]$$

- Used to 'search' for policy
- Temporal difference updates

	a1	a2	a3	'max' a
s1	3	5.5	4	a2
s2	6.2	4	5.4	a1
s3	3.3	3.1	2.9	a1

Greedy policy, {s1:a2, s2:a1, s3:a1}

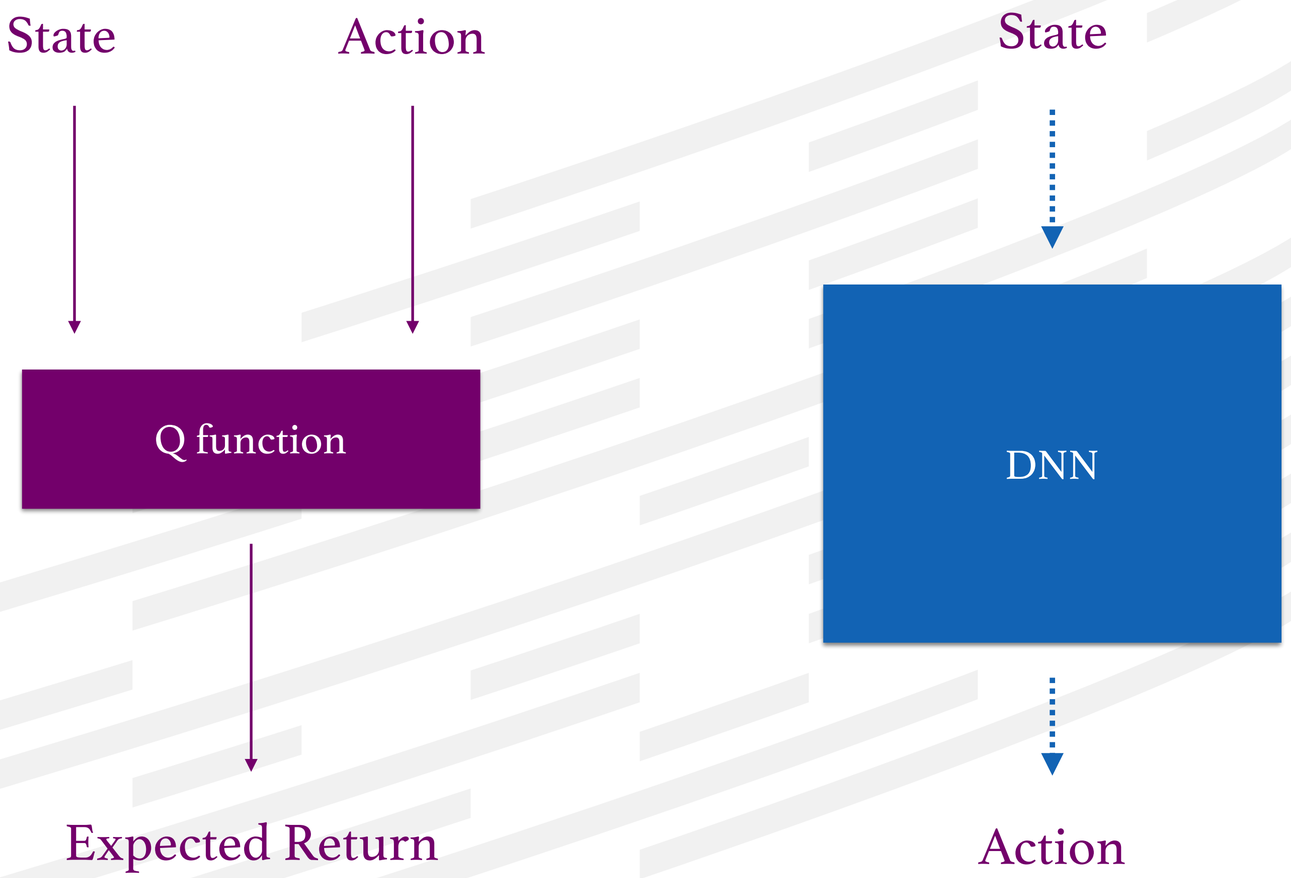
Q Learning on Mountain Car

- Demo

Q Learning — Function Approximation

The table representation is not scalable in

- Large state and action spaces
- Continuous actions, states

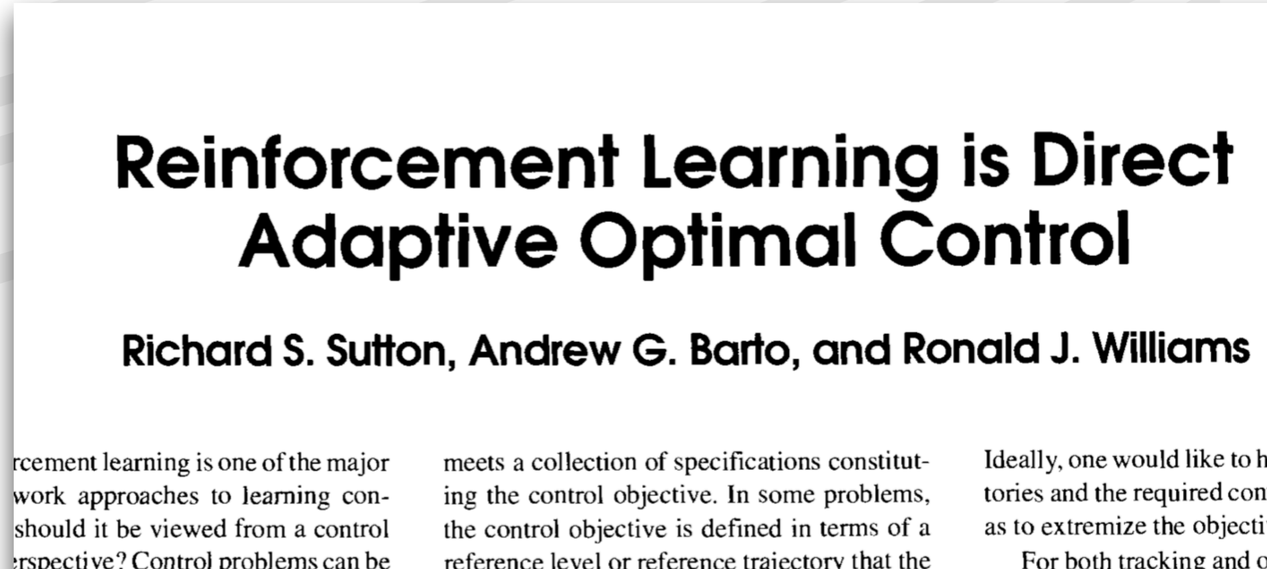


Solution:

- Use a proxy function to represent Q
- Directly learn a mapping from states to actions
- State aggregation (does not necessarily need DNNs)

Reinforcement Learning — Other Aspects

- Where do rewards come from?
  - Inverse RL — learning from experts
- How to learn relevant state aggregation, function approximation
- Relation to control



Reinforcement Learning is Direct Adaptive Optimal Control

Richard S. Sutton, Andrew G. Barto, and Ronald J. Williams

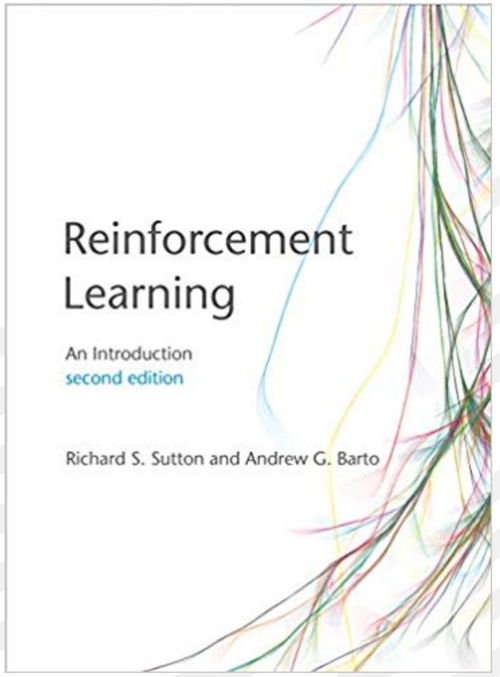
rcement learning is one of the major
work approaches to learning con-
should it be viewed from a control
erspective? Control problems can be

meets a collection of specifications constitut-
ing the control objective. In some problems,
the control objective is defined in terms of a
reference level or reference trajectory that the

Ideally, one would like to h
tories and the required con
as to extremize the objecti
For both tracking and o

References, Materials

- Books, with formal treatment



- Developer getting started tutorials

  https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

- Software
  - Applications
    - Gym Duckitown

      https://github.com/duckietown/gym-duckietown
    - Inventory management

      https://github.com/paulhendricks/gym-inventory
    - Others

      https://github.com/aikorea/awesome-rl