Advanced Prediction Models

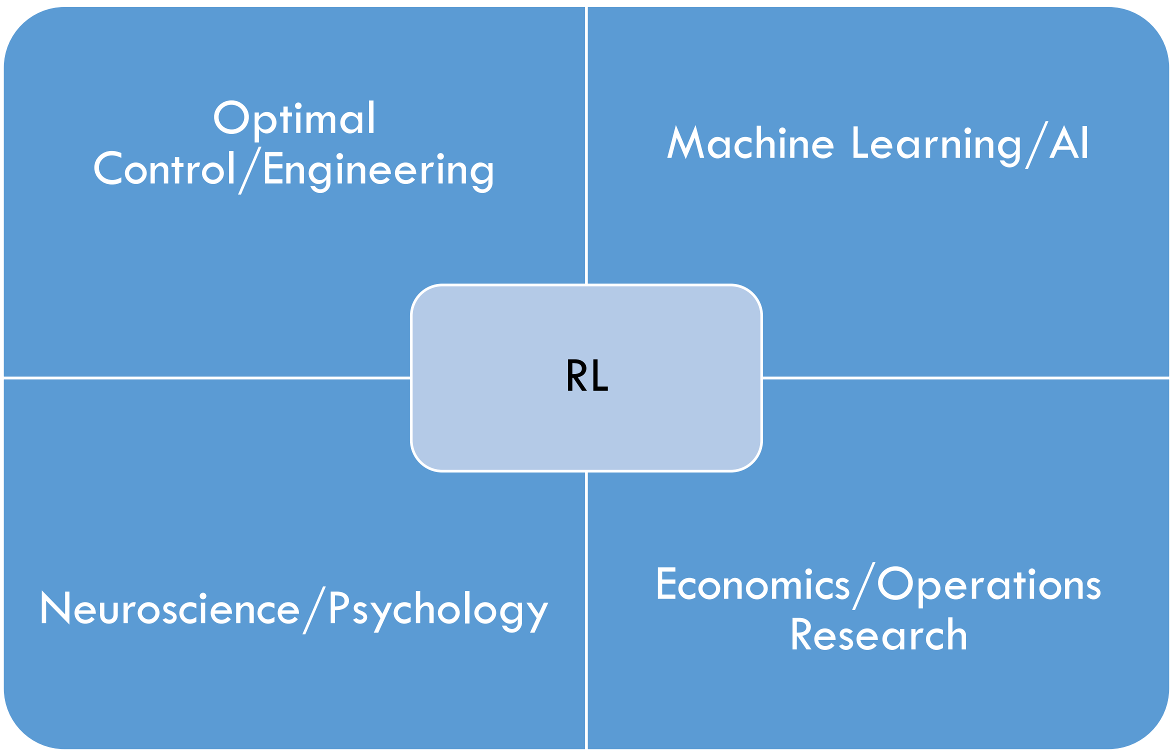
Deep Learning, Graphical Models and Reinforcement Learning

Today's Outline

- Complex Decisions
- Reinforcement Learning Basics
    - Markov Decision Process
    - (State Action) Value Function
- Q Learning Algorithm

Complex Decisions

Complex Decisions Making is Everywhere



Control
- Fly drones
- Autonomous driving

Operations
- Retain customers, UX
- Inventory management

Logistics
- Schedule transportation
- Resource allocation

Games
- Chess, Go, Atari

Complex Decisions Making is Everywhere

Computer Go

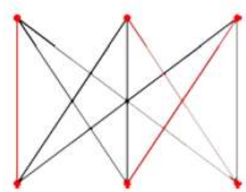
Brain computer interface

Medical trials

Packets routing

Ads placement

Dynamic allocation



Credit: Sebastien Bubeck

Control

- Fly drones
- Autonomous driving

Operations

- Retain customers, UX
- Inventory management

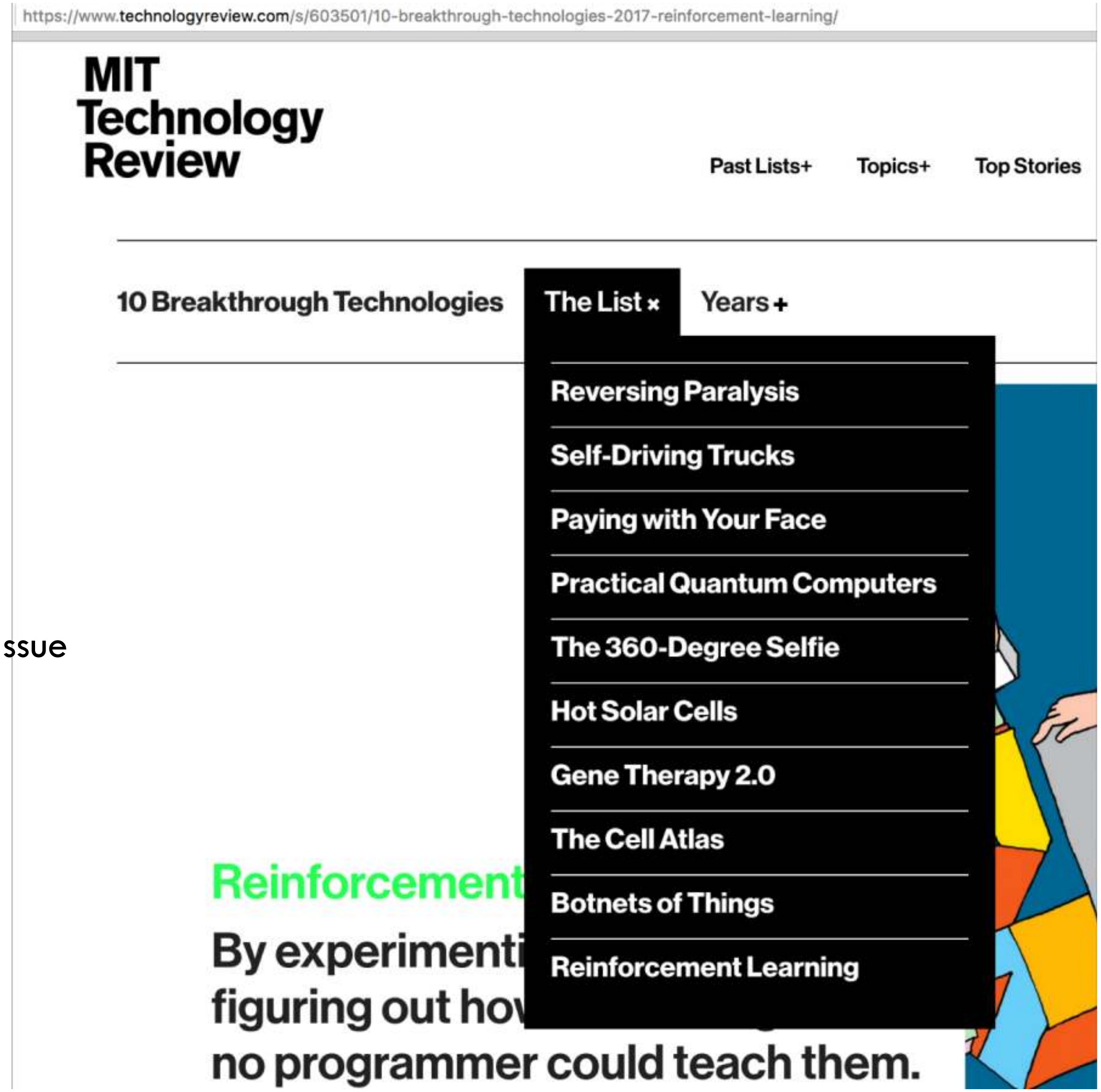
Logistics

- Schedule transportation
- Resource allocation

Games

- Chess, Go, Atari

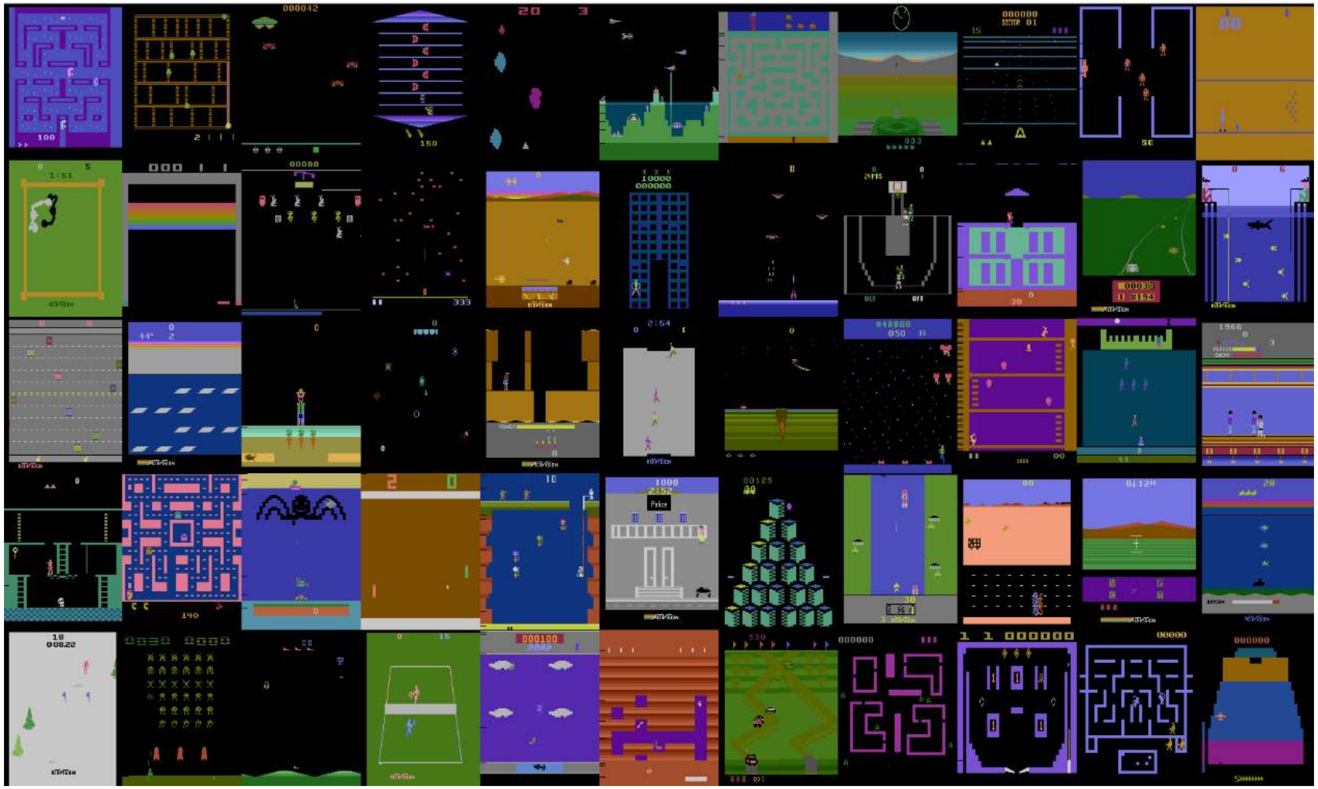
Complex Decision Making can be addressed using RL



March/April 2017 Issue

[1]Reference: technologyreview.com/s/603501/10-breakthrough-technologies-2017-reinforcement-learning/

Playing Atari Using RL (2013)



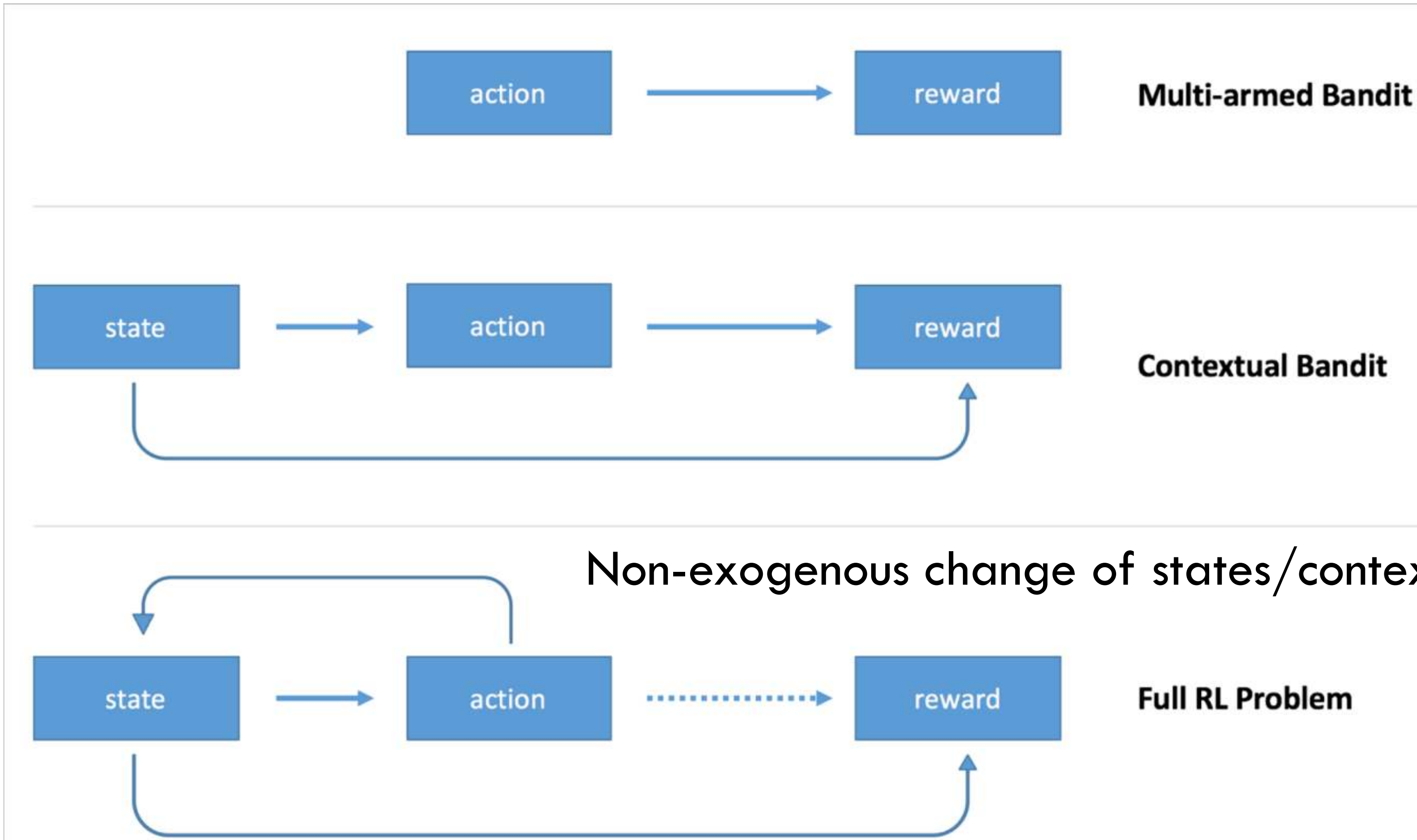
[1]Figure: Defazio Graepel, Atari Learning Environment

AlphaGo Conquers Go (2016)

[1]Reference: DeepMind, March 2016

- Videos

Need for Reinforcement Learning



Non-exogenous change of states/contexts

[1] Reference: https://medium.com/@awjuliani/simple-reinforcement-learning-with-tensorflow-part-1-5-contextual-bandits-bff01d1aad9c

Questions?

Today's Outline

- Complex Decisions
- Reinforcement Learning Basics
    - Markov Decision Process
    - (State Action) Value Function
- Q Learning Algorithm

RL Overview

- Reinforcement Learning (RL) addresses a version of the problem of sequential decision making
- Ingredients:
  - There is an environment
  - Within which, an agent takes actions
  - This action influences the future
  - Agent gets a (potentially delayed) feedback signal
- How to select actions to maximize total reward?
- RL provides several sound answers to this question

The Environment

- Sees Agent's action A_t and generates an observation S_{t+1} and a reward R_{t+1}
- Subscript t indexes time. Current observation S_t is called state

- Assume the future (at times $t+1, t+2, \ldots.$) is independent of the past $(\ldots, t-2, t-1)$ given the present (t): this is called the Markov assumption

$$P(S_{t+1}|S_t) = P(S_{t+1}|S_1, S_2, \ldots S_t)$$

- Assume everything relevant is observed

The Agent

- Agent observes R_{t+1}, S_{t+1} and these are not i.i.d. across time

- Agent's objective is to maximize expected total future reward $\mathrm{E}[R_{t+1} + \gamma R_{t+2} + \cdots]$

- Agent's actions affect what it sees in the future (S_{t+1})

- Maybe better to trade off current reward R_{t+1} to gain more rewards in the future

γ: Discount Factor

The Reward

- A reward R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximise cumulative reward

Reinforcement learning is based on the reward hypothesis

Definition (Reward Hypothesis)

All goals can be described by the maximisation of expected cumulative reward

[1] Reference: David Silver, 2015

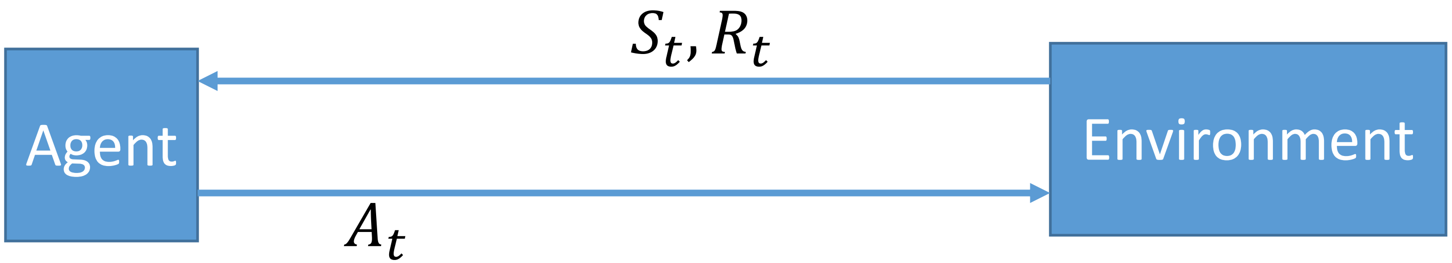
The Goal

- Goal: *select actions to maximise total future reward*
- Actions may have long term consequences
- Reward may be delayed
- It may be better to sacrifice immediate reward to gain more long-term reward
- Examples:
  - A financial investment (may take months to mature)
  - Refuelling a helicopter (might prevent a crash in several hours)
  - Blocking opponent moves (might help winning chances many moves from now)

[1]Reference: David Silver, 2015

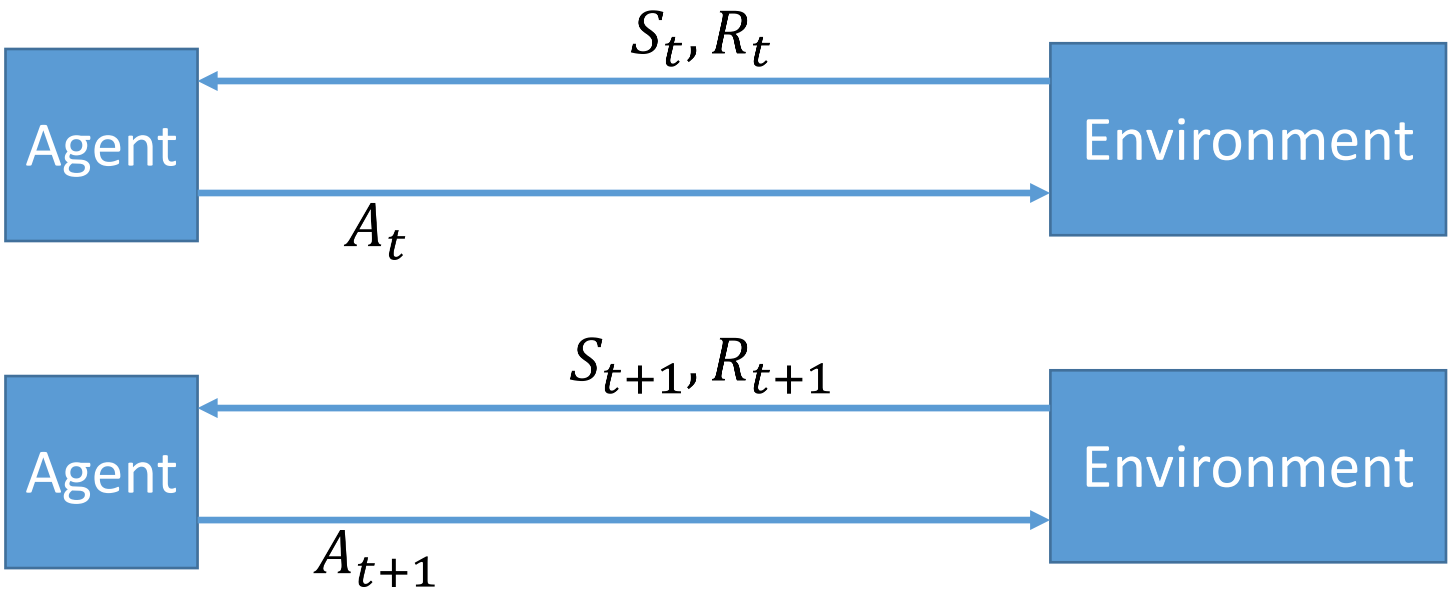
The Interactions

- Pictorially



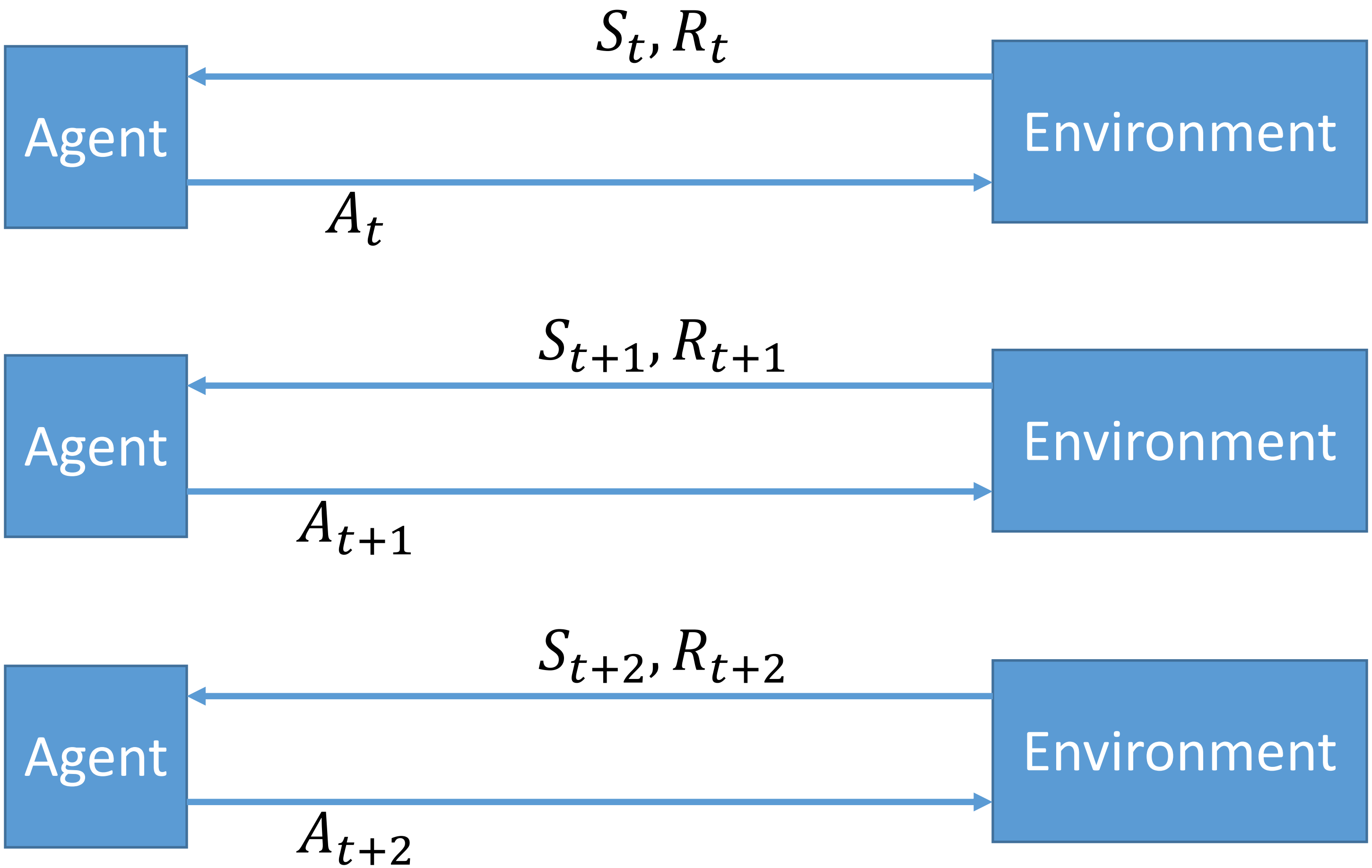
The Interactions

- Pictorially



The Interactions

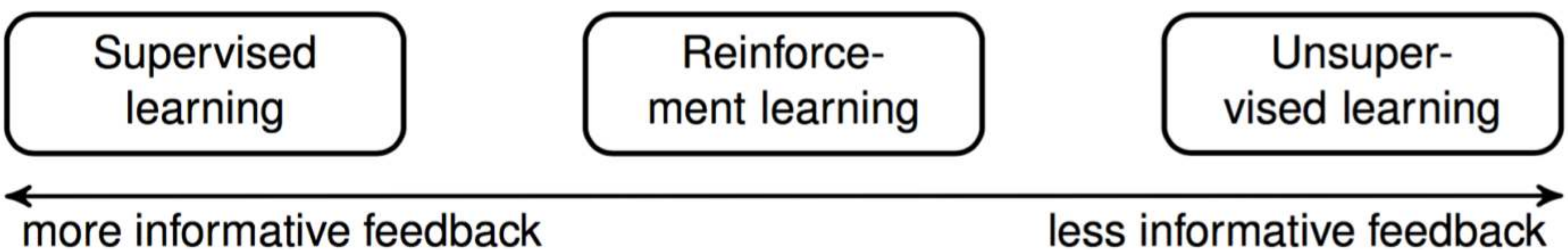
- Pictorially



RL versus other Machine Learning Settings

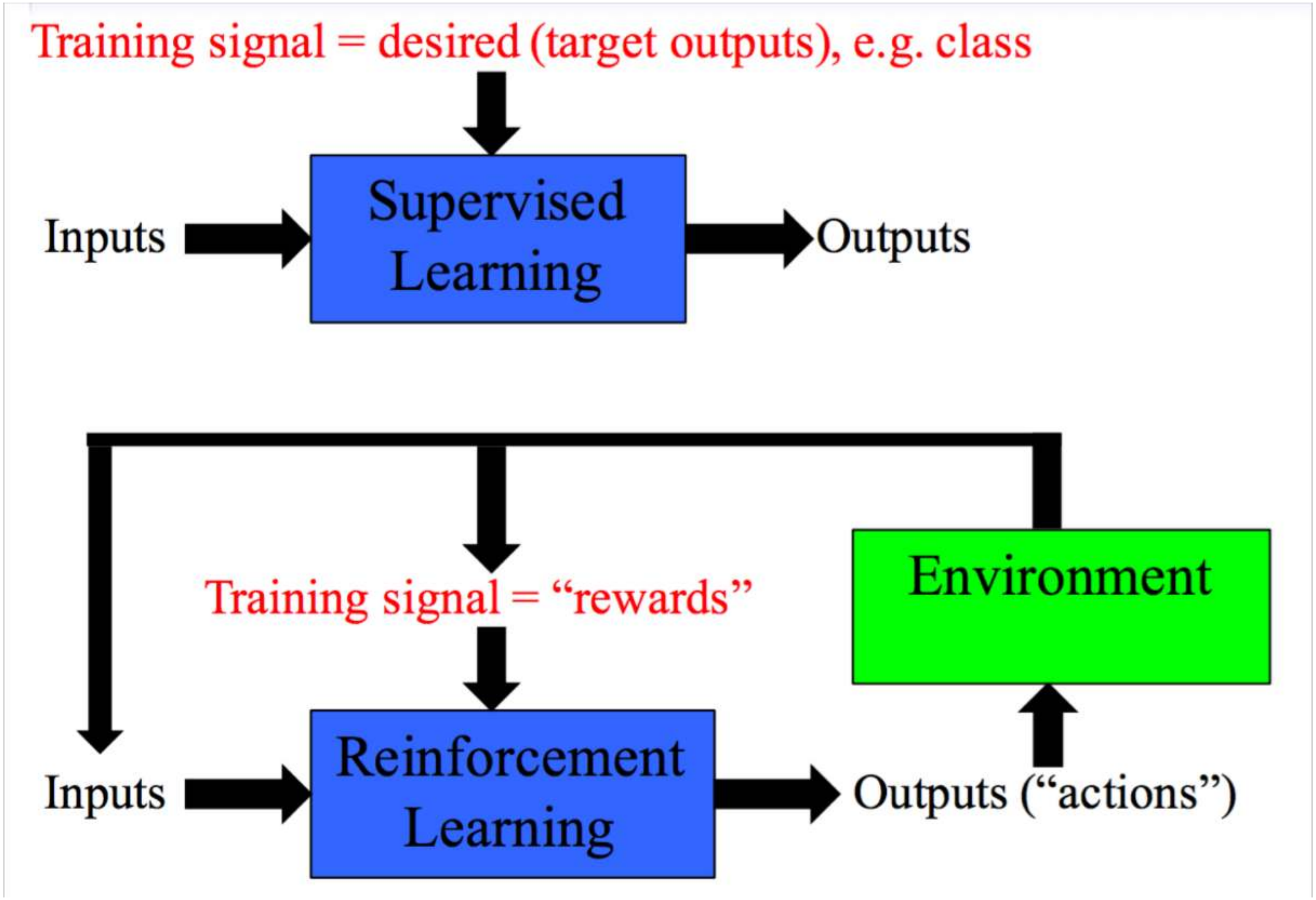
What makes reinforcement learning different from other machine learning paradigms?

- There is no supervisor, only a *reward* signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)
- Agent's actions affect the subsequent data it receives



[1]Reference: David Silver, 2015

RL versus other Machine Learning Settings



[1]Reference: Joelle Pineau, DLSS 2016

Components of an RL Agent

- An RL agent may include one or more of these components:
  - Policy: agent's behaviour function
  - Value function: how good is each state and/or action
  - Model: agent's representation of the environment

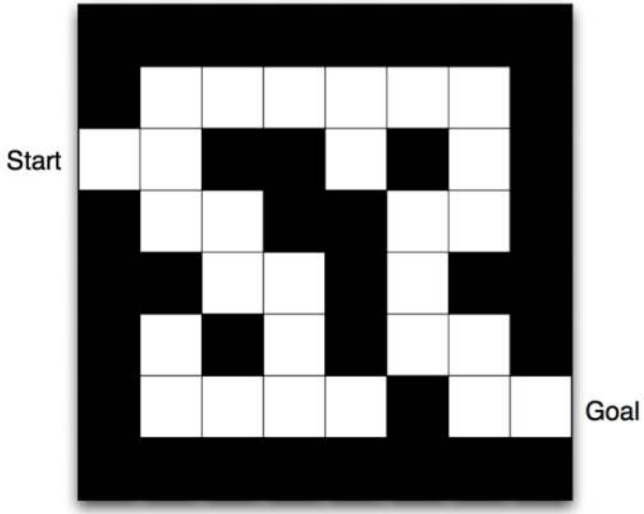
[1]Reference: David Silver, 2015

Components of RL: Policy

- A policy is the agent's behaviour
- It is a map from state to action, e.g.
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$

[1]Reference: David Silver, 2015

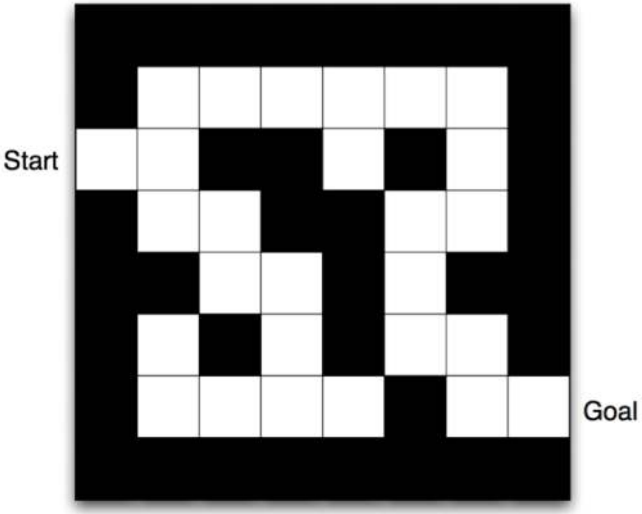
Components of RL: Policy



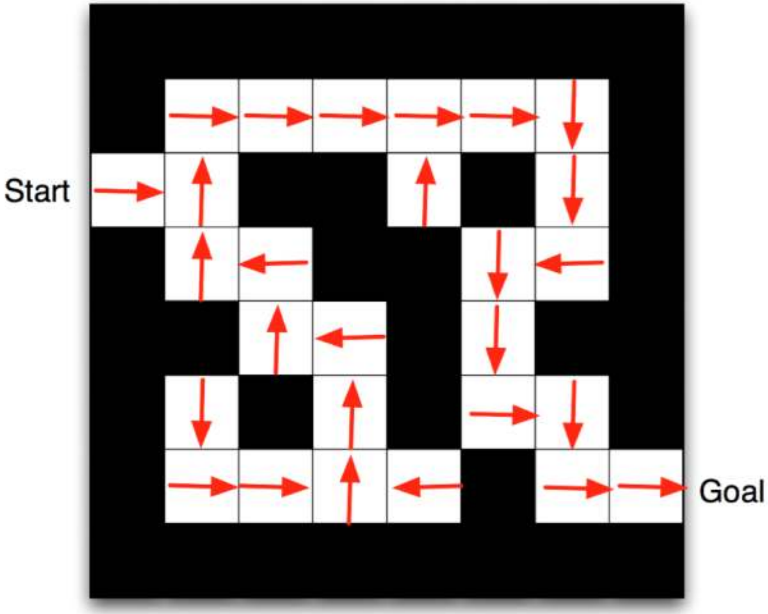
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

[1]Reference: David Silver, 2015

Components of RL: Policy



- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location



- Arrows represent policy $\pi(s)$ for each state s

[1]Reference: David Silver, 2015

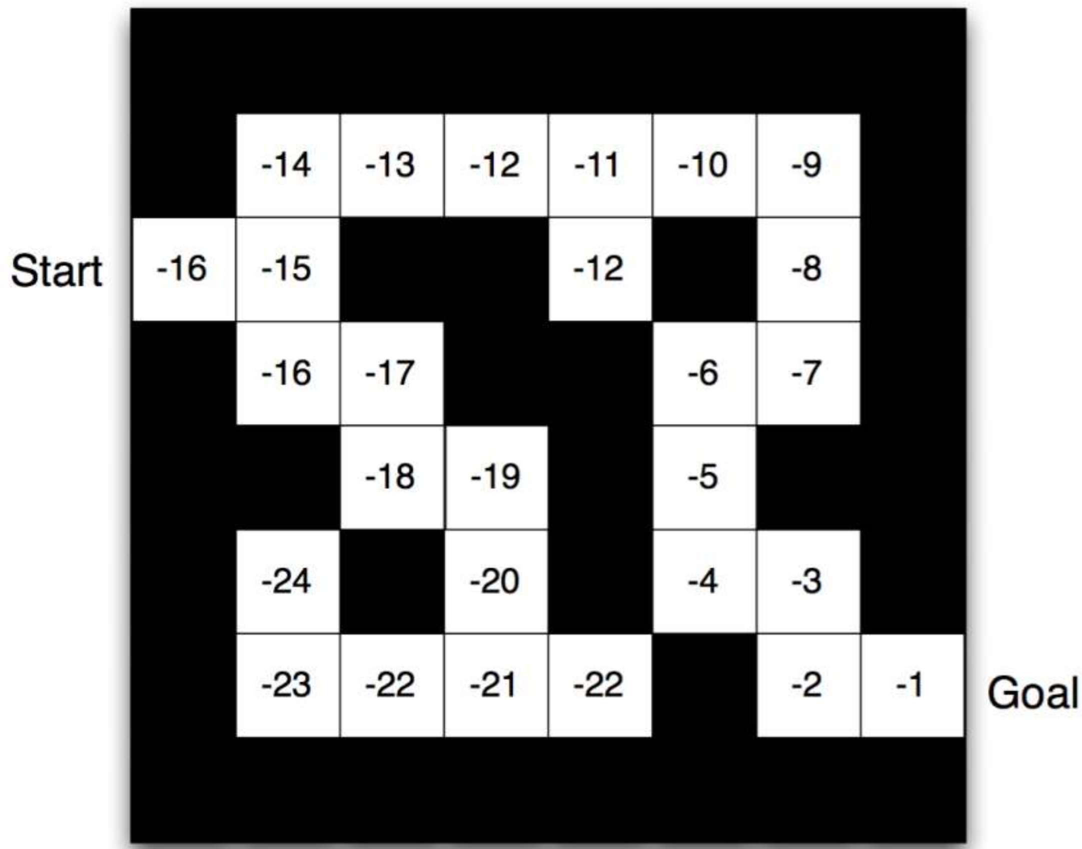
Components of RL: Value Function

- Value function is a prediction of future reward
- Used to evaluate the goodness/badness of states
- And therefore to select between actions, e.g.

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... \mid S_t = s \right]$$

[1]Reference: David Silver, 2015

Components of RL: Value Function



- Numbers represent value $v_\pi(s)$ of each state s

[1]Reference: David Silver, 2015

Components of RL: Model

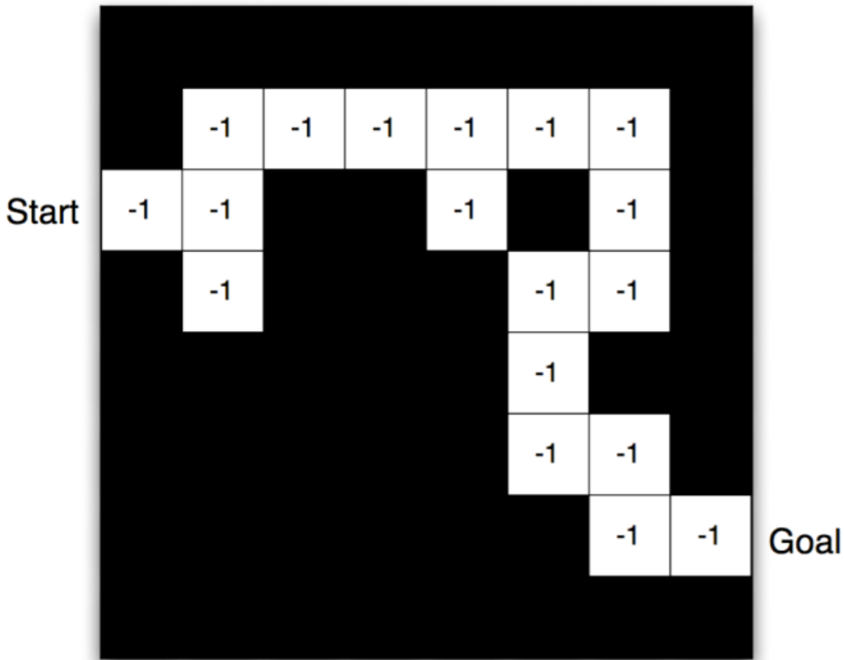
- A model predicts what the environment will do next
- $\mathcal{P}$ predicts the next state
- $\mathcal{R}$ predicts the next (immediate) reward, e.g.

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

$$\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

[1] Reference: David Silver, 2015

Components of RL: Model



- Dynamics: how actions change the state
- Rewards: how much reward from each state

- Grid layout represents transition model $\mathcal{P}^a_{ss'}$
- Numbers represent immediate reward $\mathcal{R}^a_s$ from each state s (same for all a)

[1]Reference: David Silver, 2015

Questions?

Today's Outline

- Complex Decisions
- Reinforcement Learning Basics
    - Markov Decision Process
    - (State Action) Value Function
- Q Learning Algorithm

Components of RL: MDP Framework

- We will now revisit these components formally
  - Policy $\pi(a|s)$
  - Value function $v_\pi(s)$
  - Model $\mathcal{P}_{ss'}^a$ and $\mathcal{R}_s^a$
- In the framework of Markov Decision Processes
- And then we will address the question of optimizing for the best π in realistic environments

Towards a Markov Decision Process

- MDPs are a useful way to describe the RL problem

- MDPs can be understood via the following progression
  - Start with a Markov Chain
    - State transitions happen autonomously
  - Add Rewards
    - Becomes a Markov Reward Process
  - Add Actions that influences state transitions
    - Becomes a Markov Decision Process

[1]Reference: David Silver, 2015

Markov Chain/Process

For a Markov state s and successor state s', the *state transition probability* is defined by

$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$$

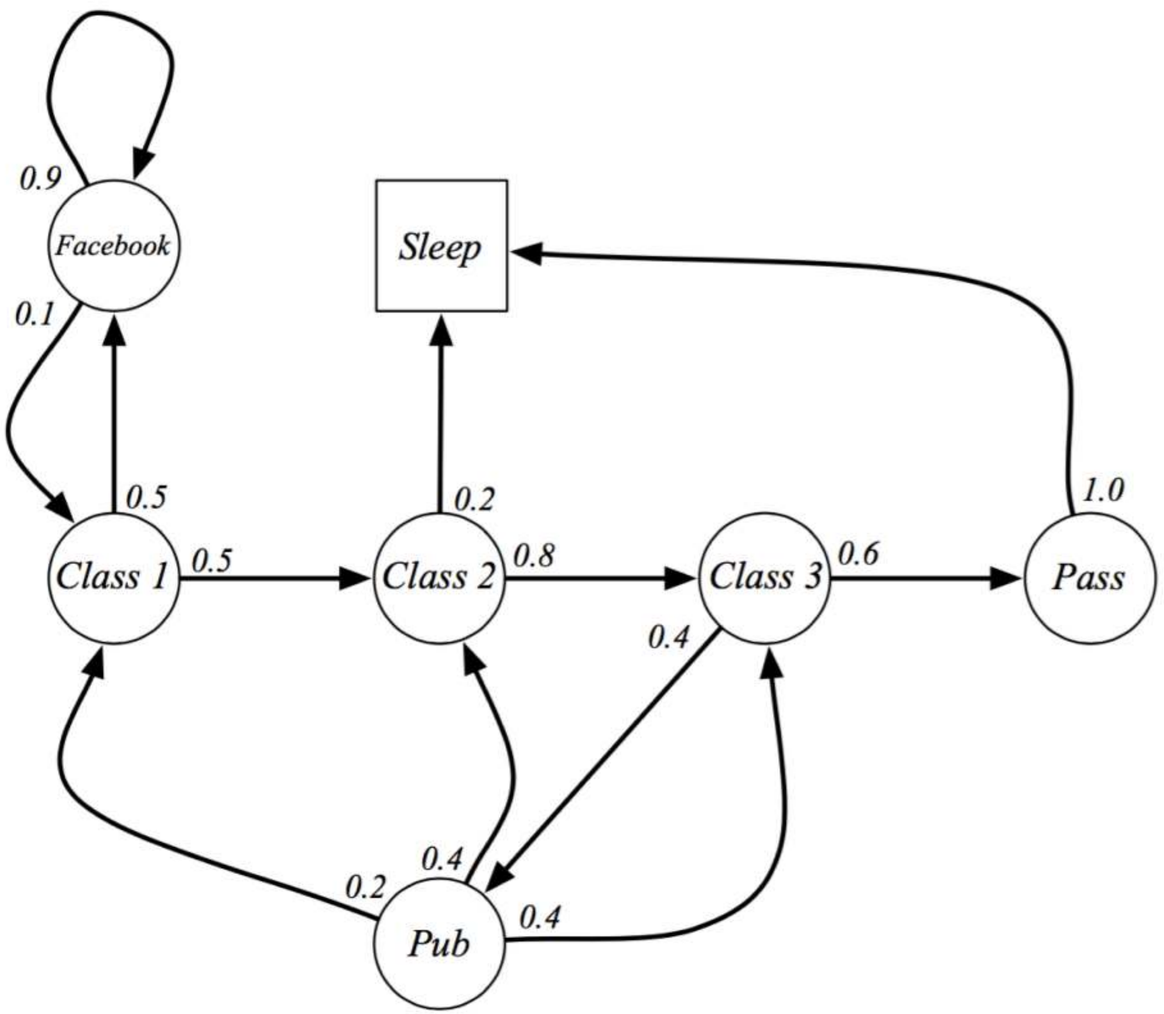
State transition matrix $\mathcal{P}$ defines transition probabilities from all states s to all successor states s',

$$\mathcal{P} = \textit{from} \begin{array}{c} \textit{to} \\ \begin{bmatrix} \mathcal{P}_{11} & \ldots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \ldots & \mathcal{P}_{nn} \end{bmatrix} \end{array}$$

where each row of the matrix sums to 1.

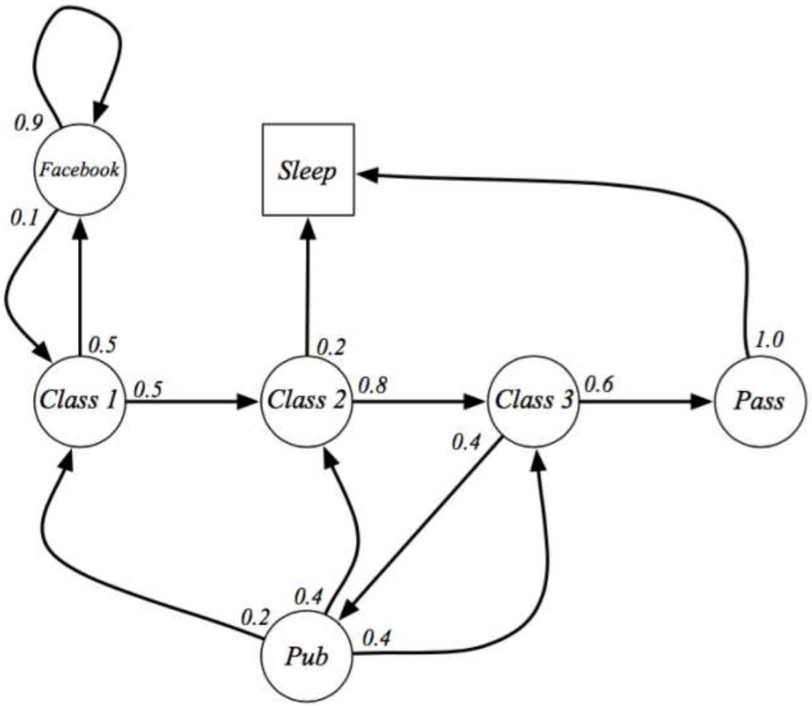
[1]Reference: David Silver, 2015

Example Markov Chain



[1]Reference: David Silver, 2015

Example Markov Chain



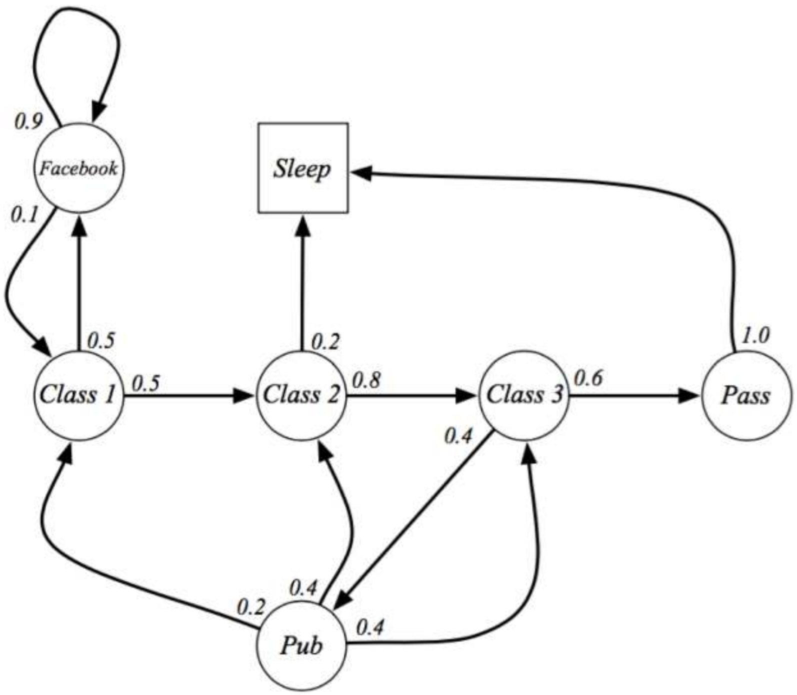
Sample episodes for Student Markov Chain starting from $S_1 = \text{C1}$

$$S_1, S_2, ..., S_T$$

- C1 C2 C3 Pass Sleep
- C1 FB FB C1 C2 Sleep
- C1 C2 C3 Pub C2 C3 Pass Sleep
- C1 FB FB C1 C2 C3 Pub C1 FB FB FB C1 C2 C3 Pub C2 Sleep

[1] Reference: David Silver, 2015

Example Markov Chain



$$\mathcal{P} = \begin{array}{cc} & \begin{array}{ccccccc} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{array} \\ \begin{array}{c} C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{array} & \begin{bmatrix} & 0.5 & & & & 0.5 & \\ & & 0.8 & & & & 0.2 \\ & & & 0.6 & 0.4 & & \\ & & & & & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1 \end{bmatrix} \end{array}$$

[1]Reference: David Silver, 2015

Markov Chain with Rewards

A Markov reward process is a Markov chain with values.

Definition

A *Markov Reward Process* is a tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{P}$ is a state transition probability matrix,
  $\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}_s = \mathbb{E}\left[R_{t+1} \mid S_t = s\right]$
- γ is a discount factor, $\gamma \in [0, 1]$

[1]Reference: David Silver, 2015

Markov Chain with Rewards

Definition

The *return* G_t is the total discounted reward from time-step t.

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

[1]Reference: David Silver, 2015

Markov Chain with Rewards

Definition

The *return* G_t is the total discounted reward from time-step t.

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

The value function $v(s)$ gives the long-term value of state s

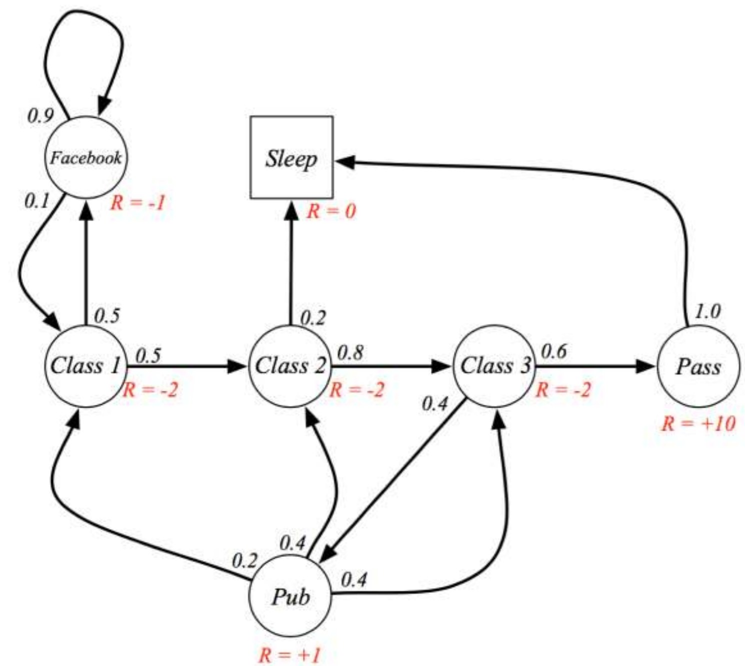
Definition

The *state value function* $v(s)$ of an MRP is the expected return starting from state s

$$v(s) = \mathbb{E}\left[G_t \mid S_t = s\right]$$

[1] Reference: David Silver, 2015

Example Markov Reward Process



Sample returns for Student MRP:
Starting from $S_1 = \text{C1}$ with $\gamma = \frac{1}{2}$

$$G_1 = R_2 + \gamma R_3 + ... + \gamma^{T-2} R_T$$

C1 C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 10 * \frac{1}{8}$	$=$	-2.25
C1 FB FB C1 C2 Sleep	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16}$	$=$	-3.125
C1 C2 C3 Pub C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 1 * \frac{1}{8} - 2 * \frac{1}{16} \ldots$	$=$	-3.41
C1 FB FB C1 C2 C3 Pub C1 ...	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} \ldots$	$=$	-3.20
FB FB FB C1 C2 C3 Pub C2 Sleep			

[1]Reference: David Silver, 2015

Recursions in Markov Reward Process

The value function can be decomposed into two parts:

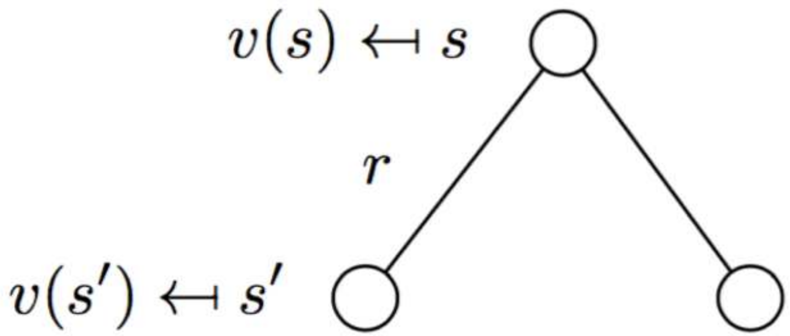
- immediate reward R_{t+1}
- discounted value of successor state $\gamma v(S_{t+1})$

$$
\begin{aligned}
v(s) &= \mathbb{E}\left[G_t \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \ldots\right) \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]
\end{aligned}
$$

[1]Reference: David Silver, 2015

Recursions in Markov Reward Process

$$v(s) = \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]$$



$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')$$

[1]Reference: David Silver, 2015

Markov Decision Process

A Markov decision process (MDP) is a Markov reward process with decisions. It is an *environment* in which all states are Markov.

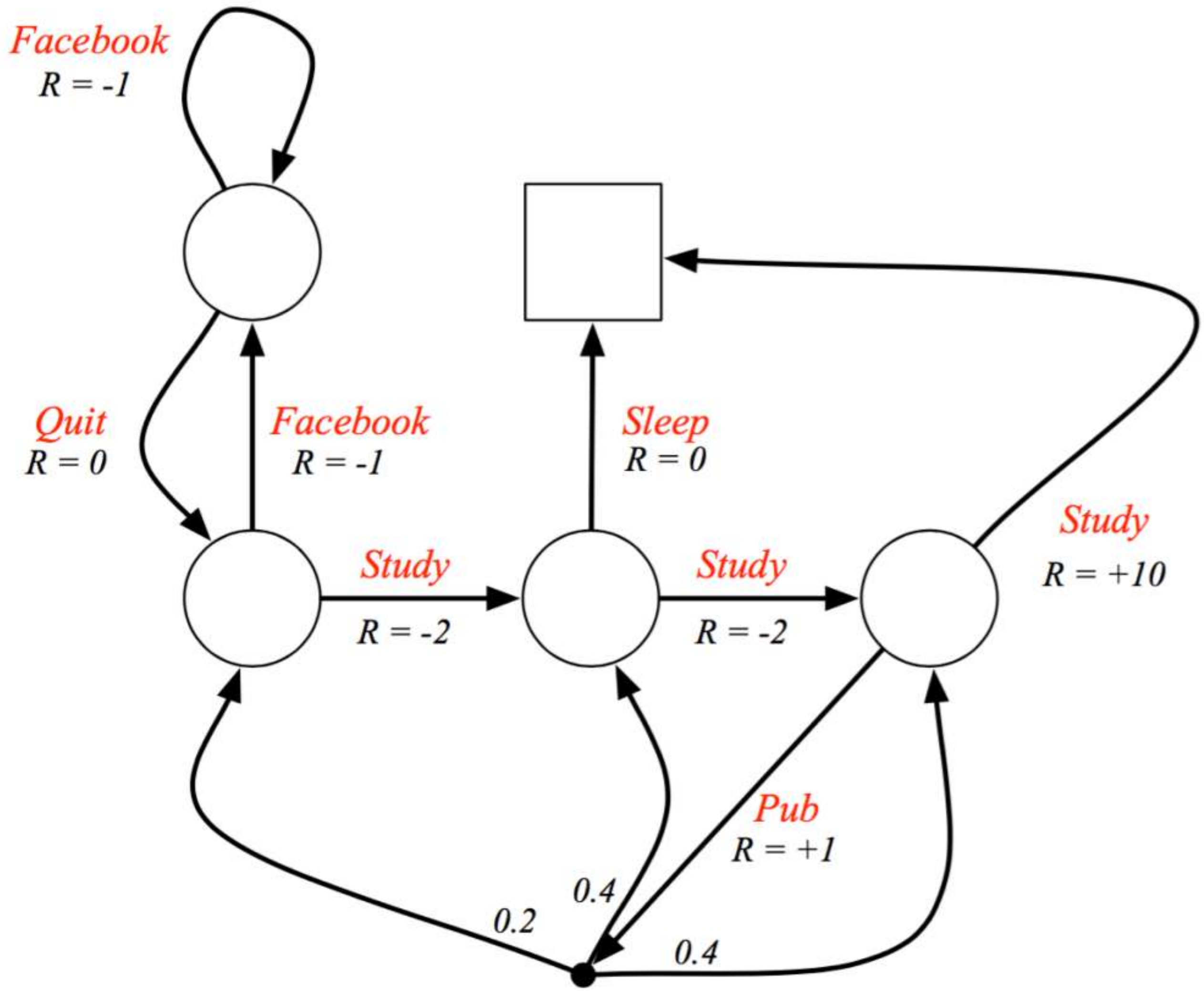
Definition

A *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}^a_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$
- γ is a discount factor $\gamma \in [0, 1]$.

[1]Reference: David Silver, 2015

Example Markov Decision Process



[1]Reference: David Silver, 2015

Markov Decision Process: Policy

- Now that we have introduced actions, we can discuss policies again

- Recall

Definition

A *policy* π is a distribution over actions given states,

$$\pi(a|s) = \mathbb{P}\left[A_t = a \mid S_t = s\right]$$

- A policy fully defines the behaviour of an agent

[1] Reference: David Silver, 2015

MDP is an MRP for a Fixed Policy

- Given an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and a policy π
- The state sequence $S_1, S_2, ...$ is a Markov process $\langle \mathcal{S}, \mathcal{P}^\pi \rangle$
- The state and reward sequence $S_1, R_2, S_2, ...$ is a Markov reward process $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$

[1]Reference: David Silver, 2015

MDP is an MRP for a Fixed Policy

- Given an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and a policy π
- The state sequence $S_1, S_2, \ldots$ is a Markov process $\langle \mathcal{S}, \mathcal{P}^\pi \rangle$
- The state and reward sequence $S_1, R_2, S_2, \ldots$ is a Markov reward process $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$
- where

$$\mathcal{P}^\pi_{s,s'} = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}^a_{ss'}$$

$$\mathcal{R}^\pi_s = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}^a_s$$

[1]Reference: David Silver, 2015

Markov Decision Process: Value Function

- We can also talk about the value function(s)

Definition

The *state-value function* $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right]$$

[1]Reference: David Silver, 2015

Markov Decision Process: Value Function

- We can also talk about the value function(s)

Definition

The *state-value function* $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right]$$

Definition

The *action-value function* $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π

$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right]$$

[1]Reference: David Silver, 2015

Recursions in MDP

*Also called the Bellman Expectation Equations

The state-value function can again be decomposed into immediate reward plus discounted value of successor state,

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right]$$

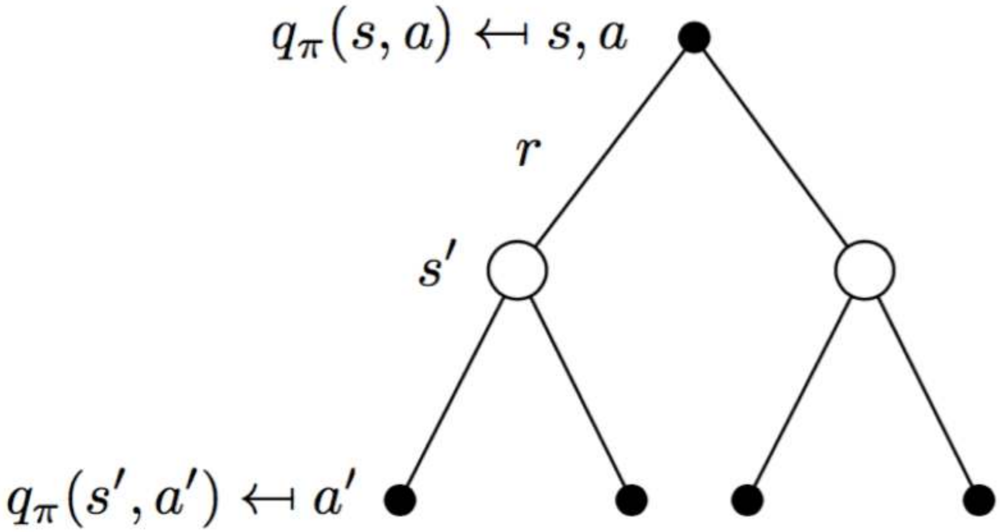
The action-value function can similarly be decomposed,

$$q_\pi(s, a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a\right]$$

[1]Reference: David Silver, 2015

Recursions in MDP

*Also called the Bellman Expectation Equations



$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')$$

[1]Reference: David Silver, 2015

Markov Decision Process: Objective

The *optimal state-value function* $v_*(s)$ is the maximum value function over all policies

$$v_*(s) = \max_{\pi} v_{\pi}(s)$$

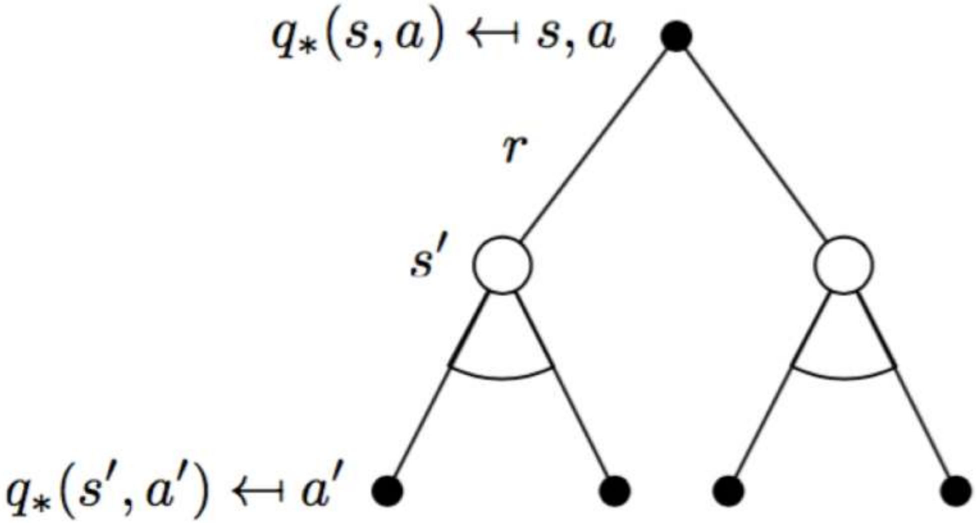
The *optimal action-value function* $q_*(s,a)$ is the maximum action-value function over all policies

$$q_*(s,a) = \max_{\pi} q_{\pi}(s,a)$$

[1]Reference: David Silver, 2015

Markov Decision Process: Objective

*Also called the Bellman Optimality Equation



$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s',a')$$

[1]Reference: David Silver, 2015

Markov Decision Process: Optimal Policy

An optimal policy can be found by maximising over $q_*(s, a)$,

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ q_*(s, a) \\ 0 & \textit{otherwise} \end{cases}$$

[1]Reference: David Silver, 2015

Questions?

Today's Outline

- Complex Decisions
- Reinforcement Learning Basics
  - Markov Decision Process
  - (State Action) Value Function
- Q Learning Algorithm

Finding the Best Policy

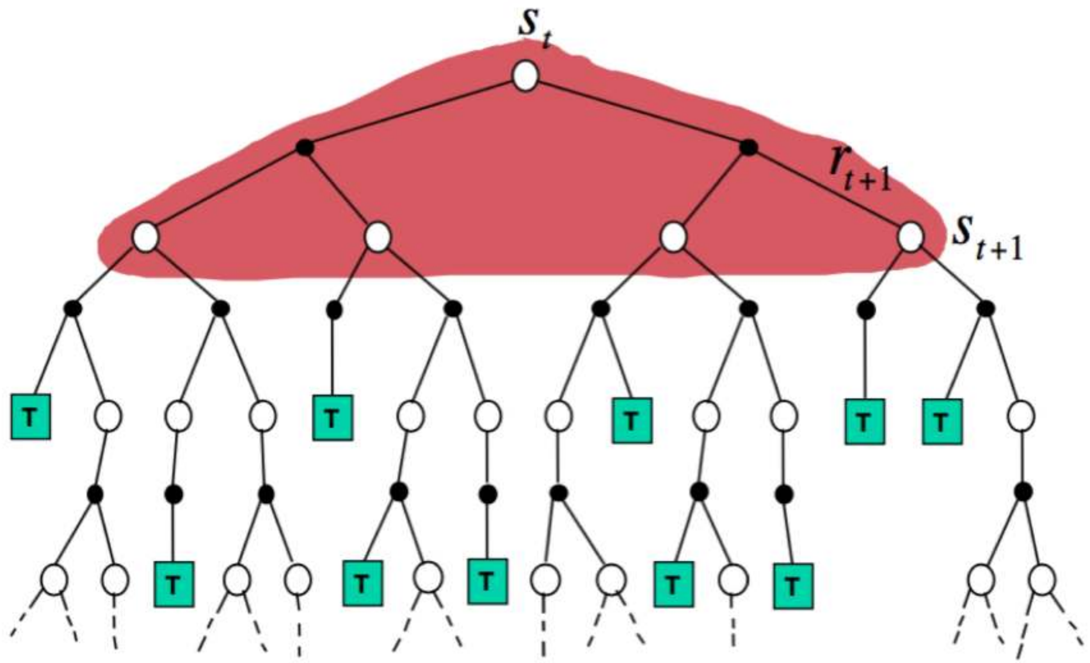
- Need to be able to do two things ideally
  - Prediction:
    - For a given policy, evaluate how good it is
      - Compute $q_\pi(s, a)$
  - Control:
    - And make an improvement from π

- We will focus on the Q Learning algorithm
  - It does prediction and control 'simultaneously'

[1]Reference: David Silver, 2015

Intuition for an Iterative Algorithm

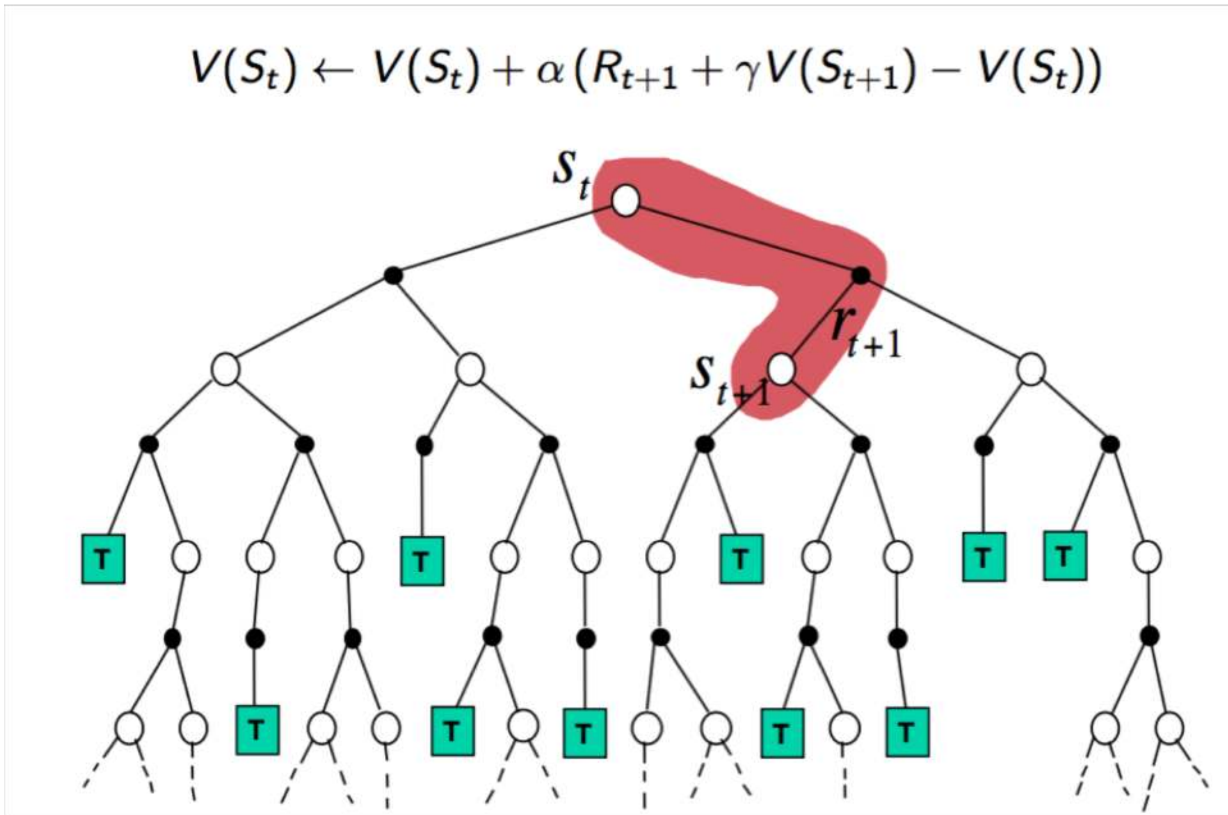
$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1})\right]$$



$$Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q(S', a') \mid s, a\right]$$

[1]Reference: David Silver, 2015

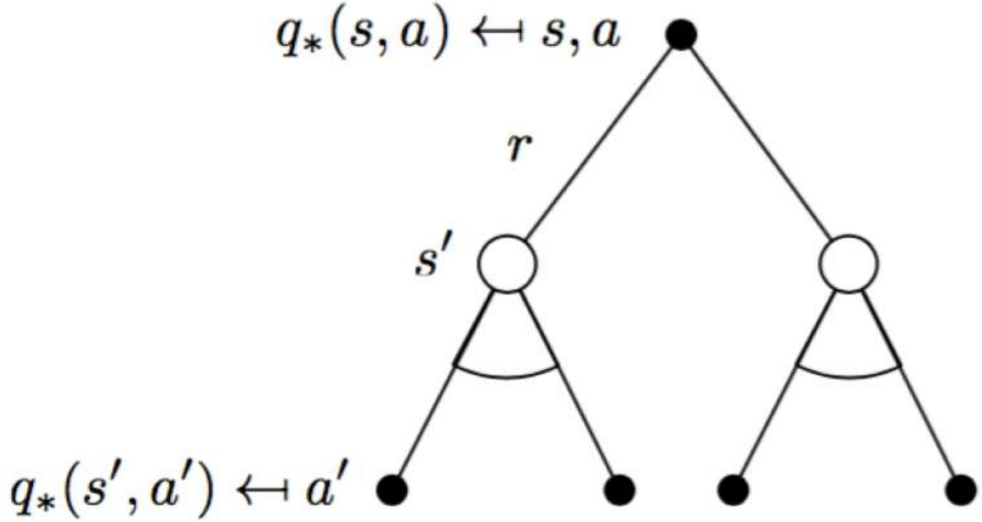
Intuition for an Iterative Algorithm



[1]Reference: David Silver, 2015

The Q Learning Algorithm

- If we know the model
  - Turn the Bellman Optimality Equation into an iterative update
  - This is called Value Iteration

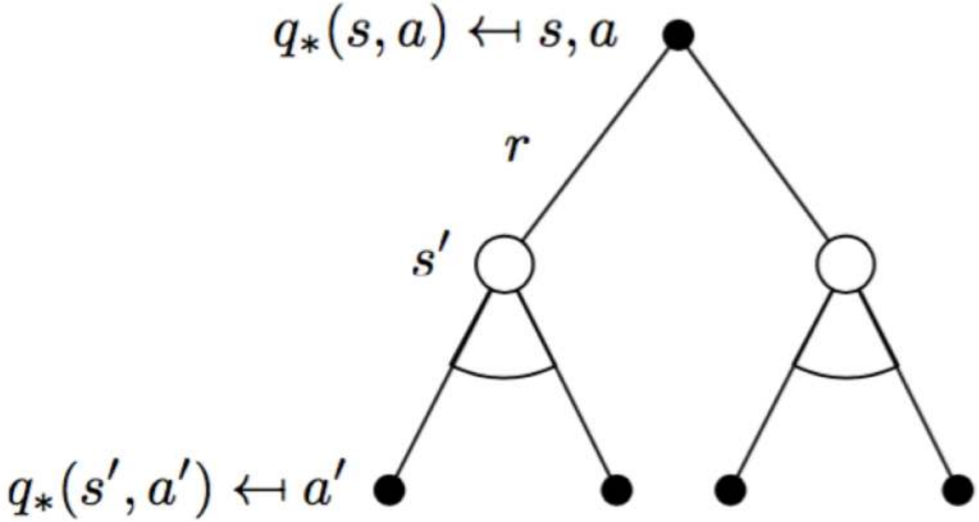


$$q_*(s,a) \boxed{=} \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s',a')$$

[1]Reference: David Silver, 2015

The Q Learning Algorithm

- If we do not know the model
  - Do sampling to get an incremental iterative update
  - Choose next actions to ensure exploration

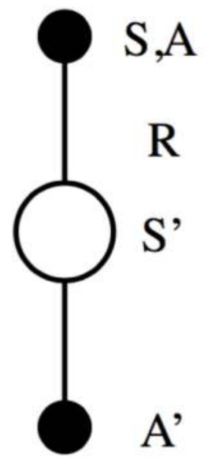


$$q_*(s,a) \boxed{=} \mathcal{R}_s^a + \gamma \boxed{\sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a} \max_{a'} q_*(s',a')$$

[1]Reference: David Silver, 2015

The Q Learning Algorithm

- If we do not know the model
  - Do sampling to get an incremental iterative update
  - Choose next actions to ensure exploration

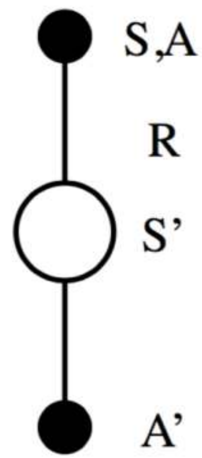


$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma Q(S',A') - Q(S,A)\right)$$

[1]Reference: David Silver, 2015

The Q Learning Algorithm

- If we do not know the model
  - Do sampling to get an incremental iterative update
  - Choose next actions to ensure exploration

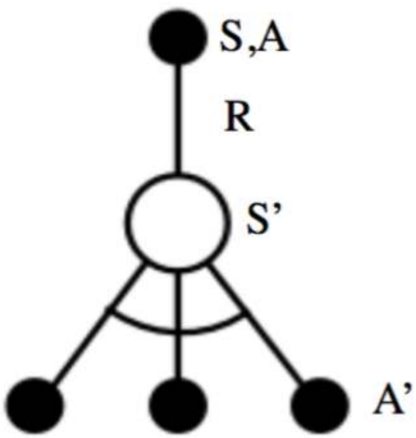


$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma Q(S', \boxed{A'}) - Q(S,A)\right)$$

[1]Reference: David Silver, 2015

The Q Learning Algorithm

- If we do not know the model
  - Do sampling to get an incremental iterative update
  - Choose next actions to ensure exploration



$$Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \gamma \boxed{\max_{a'} Q(S',a')} - Q(S,A)\right)$$

[1]Reference: David Silver, 2015

The Q Learning Algorithm

- Initialize Q, which is a table of size #states$\times$#actions
- Start at state s_1

The Q Learning Algorithm

- Initialize Q, which is a table of size #states×#actions
- Start at state s_1
- For $t = 1,2,3, \ldots.$
  - Take A_t chosen uniformly at random with probability ϵ

Explore

The Q Learning Algorithm

- Initialize Q, which is a table of size #states×#actions
- Start at state s_1
- For $t = 1,2,3, \ldots.$
  - Take A_t chosen uniformly at random with probability ϵ
  - Take $\mathrm{argmax}_{a \in A}\, Q(S_t, a)$ with probability $1 - \epsilon$

Explore

Exploit

The Q Learning Algorithm

- Initialize Q, which is a table of size #states×#actions
- Start at state s_1
- For $t = 1,2,3, \ldots.$
  - Take A_t chosen uniformly at random with probability ϵ
  - Take $\operatorname{argmax}_{a \in A} Q(S_t, a)$ with probability $1 - \epsilon$
  - Update Q:
    - $Q(S_t, A_t) = Q(S_t, A_t) + \alpha_t \underbrace{(R_{t+1} + \gamma \max_{a \in A} Q(S_{t+1}, a) - Q(S_t, A_t))}_{\text{Temporal difference error}}$

Explore

Exploit

The Q Learning Algorithm

- Initialize Q, which is a table of size #states×#actions
- Start at state s_1
- For $t = 1,2,3, \ldots.$
  - Take A_t chosen uniformly at random with probability ϵ — Explore
  - Take $\text{argmax}_{a \in A}\, Q(S_t, a)$ with probability $1 - \epsilon$ — Exploit
  - Update Q:
    - $Q(S_t, A_t) = Q(S_t, A_t) + \alpha_t \underbrace{\left(R_{t+1} + \gamma \max_{a \in A} Q(S_{t+1}, a) \; - Q(S_t, A_t)\right)}_{\text{Temporal difference error}}$

- Parameter ϵ is the exploration parameter
- Parameter α_t is the learning rate

The Q Learning Algorithm

- Initialize Q, which is a table of size #states×#actions
- Start at state s_1
- For $t = 1,2,3, \dots.$
  - Take A_t chosen uniformly at random with probability ϵ — Explore
  - Take $\text{argmax}_{a \in A} Q(S_t, a)$ with probability $1 - \epsilon$ — Exploit
  - Update Q:
    - $Q(S_t, A_t) = Q(S_t, A_t) + \alpha_t \underbrace{\left(R_{t+1} + \gamma \max_{a \in A} Q(S_{t+1}, a) - Q(S_t, A_t)\right)}_{\text{Temporal difference error}}$

- Parameter ϵ is the exploration parameter
- Parameter α_t is the learning rate
- Under appropriate assumptions[1], $\lim_{t \to \infty} Q = Q^*$

[1] Reference: Christopher J. C. H. Watkins and Peter Dayan, 1992

The Q Learning Algorithm: Recap

- Bellman Optimality Equation gives rise to the Q-Value Iteration algorithm
- Making this algorithm incremental, sampled and adding ϵ-greedy exploration gives Q Learning Algorithm

Q-Value Iteration	Q-Learning
$Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q(S',a') \mid s,a\right]$	$Q(S,A) \overset{\alpha}{\leftarrow} R + \gamma \max_{a' \in \mathcal{A}} Q(S',a')$

where $x \overset{\alpha}{\leftarrow} y \equiv x \leftarrow x + \alpha(y - x)$

[1]Reference: David Silver, 2015

Questions?

Summary

- RL is a great framework to make agents intelligent
- Specify goals and provide feedback

- Many challenges still remain: exciting opportunity to contribute towards next generation of artificially intelligent and autonomous agents.

- In the next lecture, we will see that deep learning function approximation based RL agents show promise in large complex tasks: representations matter!
  - Applications such as
    - Self-driving cars
    - Intelligent virtual agents

Appendix

Sample Exam Questions

- What is the difference between a Markov Chain and a Markov Reward Process?
- What is the difference between a Markov Chain and a Markov Decision Process?
- Why is exploration needed in the reinforcement learning setting?
- What does the optimal state-action value function signify?
- What are the two objects (distributions) of an RL model?
- What is the difference between supervised learning and reinforcement learning?

Additional Resources

- An Introduction to Reinforcement Learning by Richard Sutton and Andrew Barto
  - http://incompleteideas.net/sutton/book/the-book.html
- Course on Reinforcement Learning by David Silver at UCL (includes video lectures)
  - http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
- Research Papers
  - Deep RL collection: https://github.com/junhyukoh/deep-reinforcement-learning-papers
  - [MKSRVBGRFOPBSAKKWLH2015] Mnih et al. Human-level control through deep reinforcement learning. Nature, 518:529–533, 2015.
  - [SHMGSDSAPLDGNKSLLKGH2016] Silver et al. Mastering the game of Go with deep neural networks and tree search. Nature, 529: 484–489, 2016.

Cons of RL

- Reinforcement Learning requires experiencing the environment many many times
- This is because it is a trial and error based approach

- Impractical for many complex tasks
- Unless one has access to simulators where an RL agent can practice a billon times

RL versus other Machine Learning Settings

- There is a notion of exploration and exploitation, similar to Multi-armed bandits and Contextual bandits
  - *Exploration* finds more information about the environment
  - *Exploitation* exploits known information to maximise reward
  - It is usually important to explore as well as exploit
- Key difference: actions influence future contexts
  - Reinforcement learning is like trial-and-error learning
  - The agent should discover a good policy
  - From its experiences of the environment
  - Without losing too much reward along the way

[1]Reference: David Silver, 2015

RL versus other Sequential Decision Making Settings

Two fundamental problems in sequential decision making

- Reinforcement Learning:
  - The environment is initially unknown
  - The agent interacts with the environment
  - The agent improves its policy
- Planning:
  - A model of the environment is known
  - The agent performs computations with its model (without any external interaction)
  - The agent improves its policy
  - a.k.a. deliberation, reasoning, introspection, pondering, thought, search

[1]Reference: David Silver, 2015

Types of RL Agents

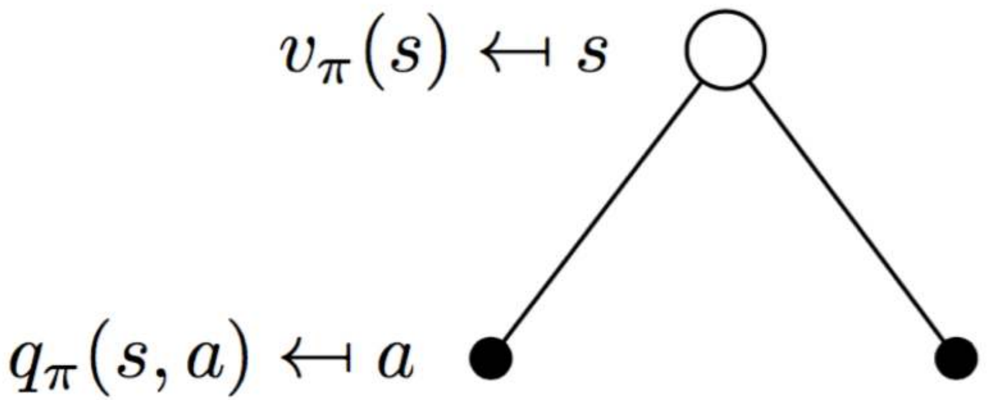
- There are many ways to design them, so we roughly categorize then as below:

- Value Based
  - No Policy (Implicit)
  - Value Function
- Policy Based
  - Policy
  - No Value Function
- Actor Critic
  - Policy
  - Value Function

- Model Free
  - Policy and/or Value Function
  - No Model
- Model Based
  - Policy and/or Value Function
  - Model

[1]Reference: David Silver, 2015

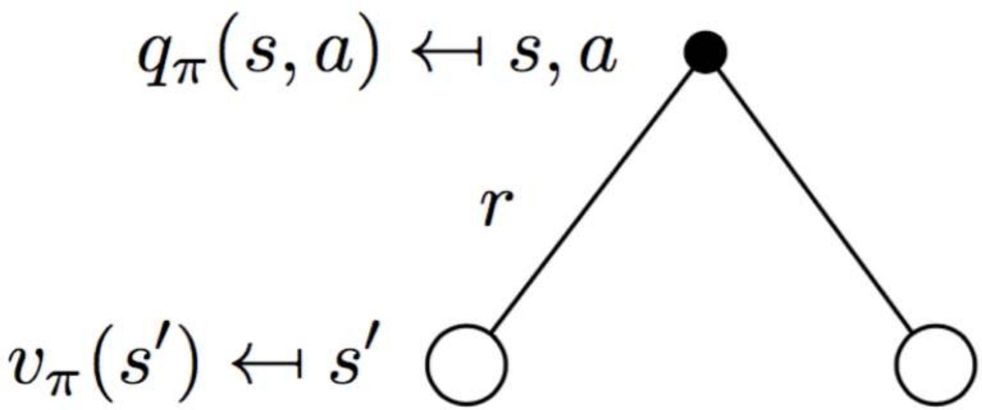
Relating the Two Value Functions I



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

[1]Reference: David Silver, 2015

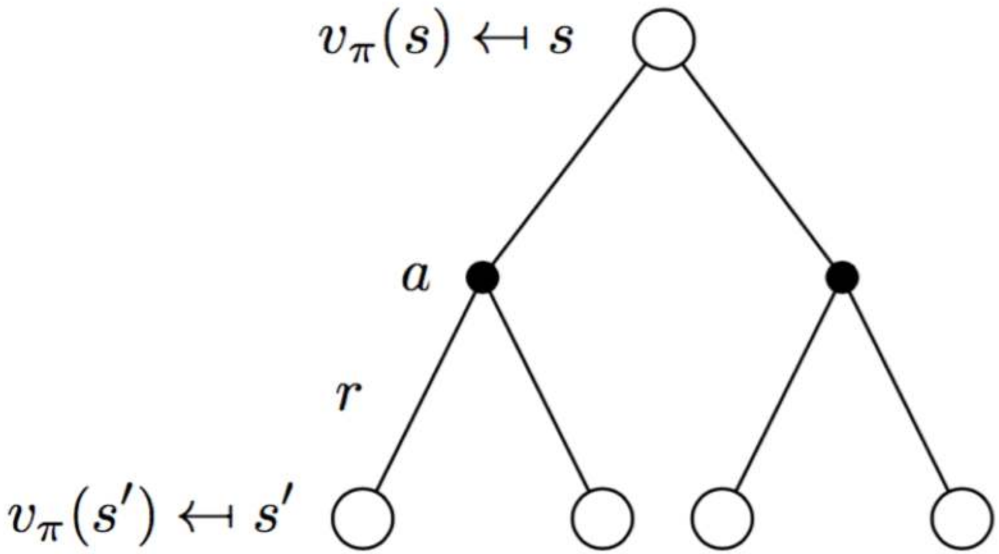
Relating the Two Value Functions II



$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

[1]Reference: David Silver, 2015

Recursion in MDP: Value Function Version



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

[1]Reference: David Silver, 2015

Relating Policy and Value Function

An optimal policy can be found by maximising over $q_*(s, a)$,

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ q_*(s, a) \\ 0 & otherwise \end{cases}$$

[1]Reference: David Silver, 2015