

Reinforcement Learning

PROF. ERIK SUDDERTH

WINTER 2018

CS 178: Machine Learning

courtesy Sameer Singh, based on slides by David Silva

Machine Learning

Intro to Reinforcement Learning

Markov Processes

Markov Reward Processes

Markov Decision Processes

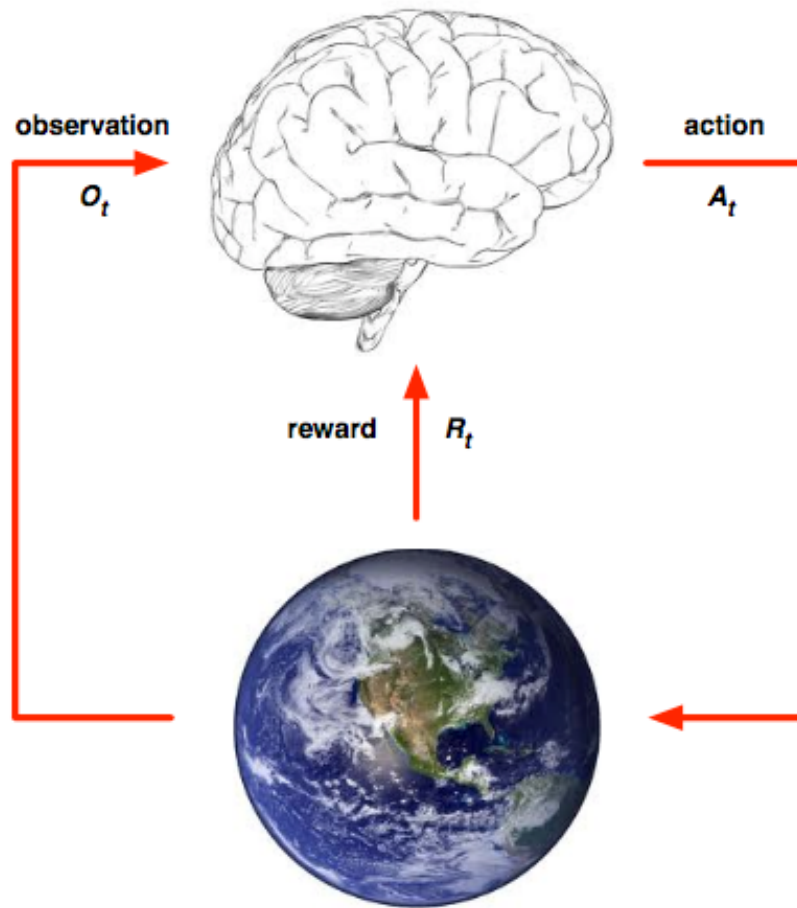
What makes it different?

No direct supervision, only rewards
Feedback is delayed, not instantaneous
Time really matters, i.e. data is sequential
Agent's actions affect what data it will receive

Examples

- Fly stunt maneuvers in a helicopter
- Defeat the world champion at Backgammon or Go
- Manage an investment portfolio
- Control a power station
- Make a humanoid robot walk
- Play many different Atari games better than humans

Agent-Environment Interface



Agent

- decides on an action
- receives next observation
- receives next reward

Environment

- executes the action
- computes next observation
- computes next reward

Reward, R_t

How well the
agent is doing

+, positive (Good)
-, negative (Bad)

Nothing about WHY it is
doing well, could have
little to do with A_{t-1}

Agent is trying to maximize its **cumulative reward**

Example of Rewards

- Fly stunt maneuvers in a helicopter
 - +ve reward for following desired trajectory
 - -ve reward for crashing
- Defeat the world champion at Backgammon
 - +/-ve reward for winning/losing a game
- Manage an investment portfolio
 - +ve reward for each \$ in bank
- Control a power station
 - +ve reward for producing power
 - -ve reward for exceeding safety thresholds
- Make a humanoid robot walk
 - +ve reward for forward motion
 - -ve reward for falling over
- Play many different Atari games better than humans
 - +/-ve reward for increasing/decreasing score

Sequential Decision Making

Actions have long term consequences

Rewards may be delayed

May be better to sacrifice short term reward for long term benefit

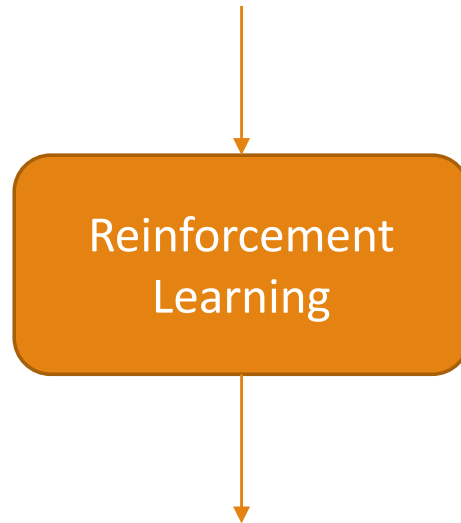
Examples

- A financial investment (may take months to mature)
- Refueling a helicopter (might prevent a crash later)
- Blocking opponent moves (might eventually help win)
- Spend a lot of money and go to college (earn more later)
- Don't commit crimes (rewarded by not going to jail)
- Get started on final project early (avoid stress later)

A key aspect of intelligence: How far ahead are you able to plan?

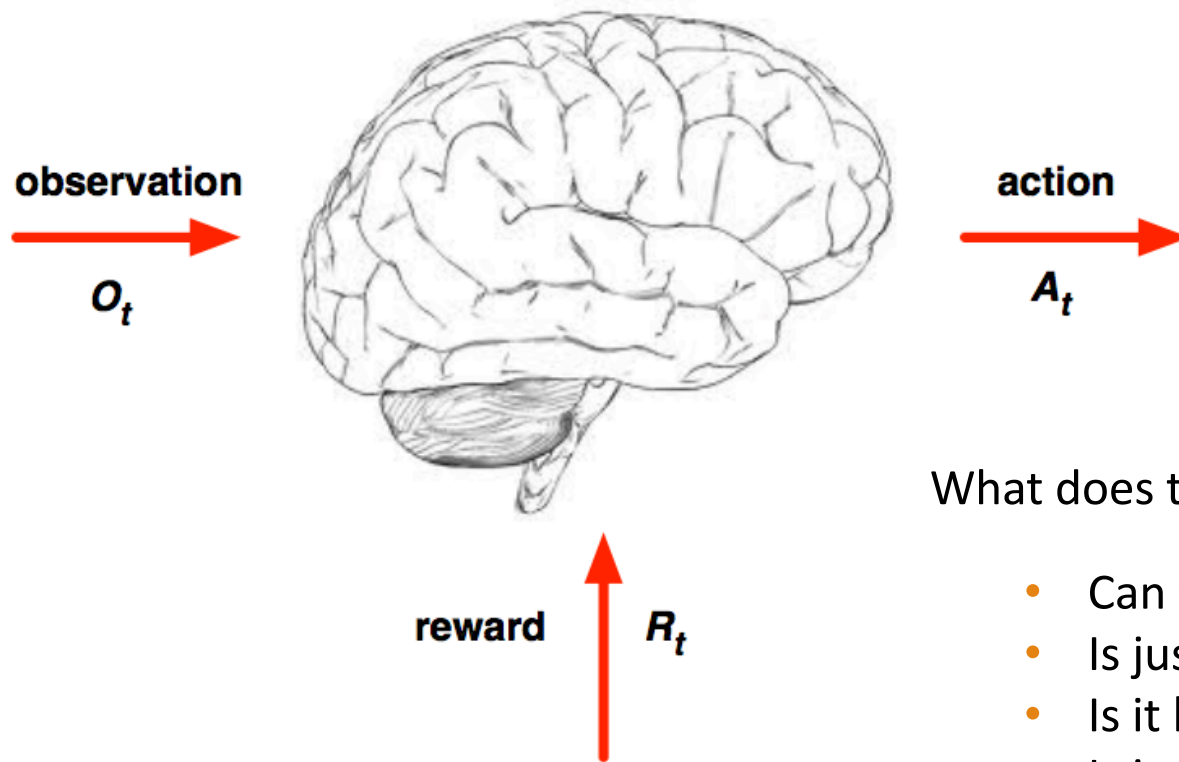
Reinforcement Learning

Given an environment
(produces observations and rewards)



Automated agent that selects actions
to maximize total rewards in the environment

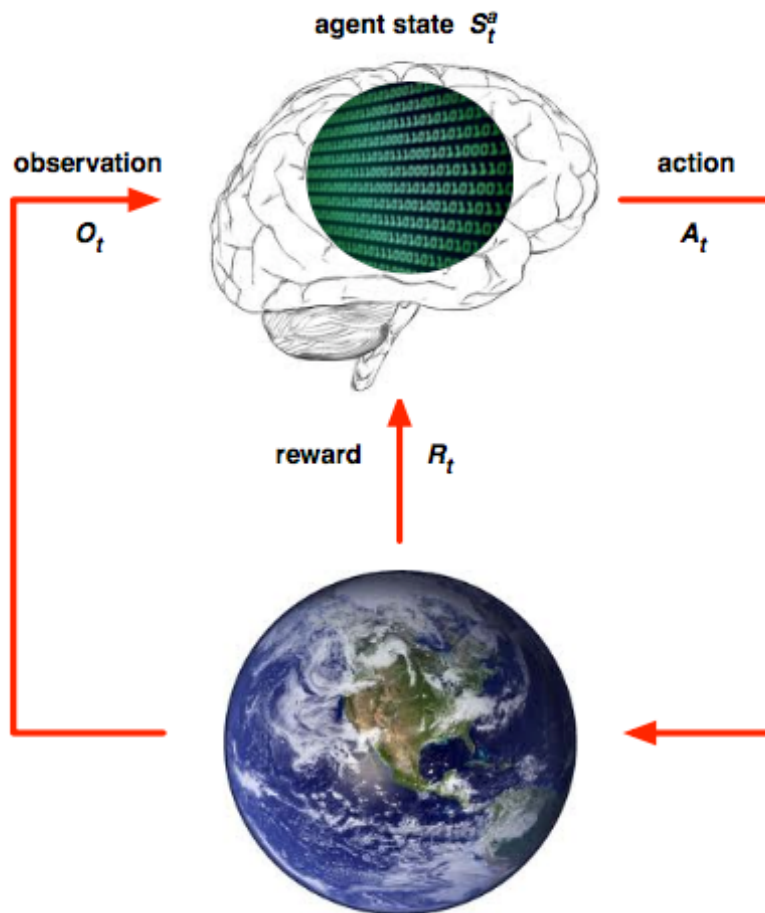
Let's look at the Agent



What does the choice of action depend on?

- Can you ignore O_t completely?
- Is just O_t enough? Or (O_t, A_t) ?
- Is it last few observations?
- Is it all observations so far?

Agent State, S_t



History: everything that happened so far

$$H_t = O_1 R_1 A_1 O_2 R_2 A_2 O_3 R_3, \dots, A_{t-1} O_t R_t$$

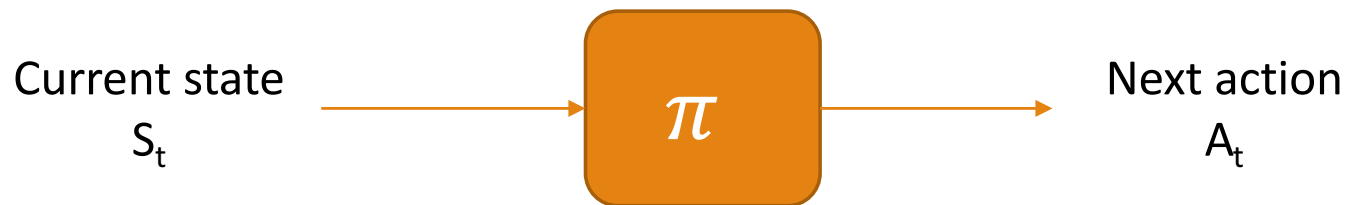
State, S_t can be

O_t
 $O_t R_t$
 $A_{t-1} O_t R_t$
 $O_{t-3} O_{t-2} O_{t-1} O_t$

In general, $S_t = f(H_t)$

You, as AI designer,
specify this function

Agent Policy, π



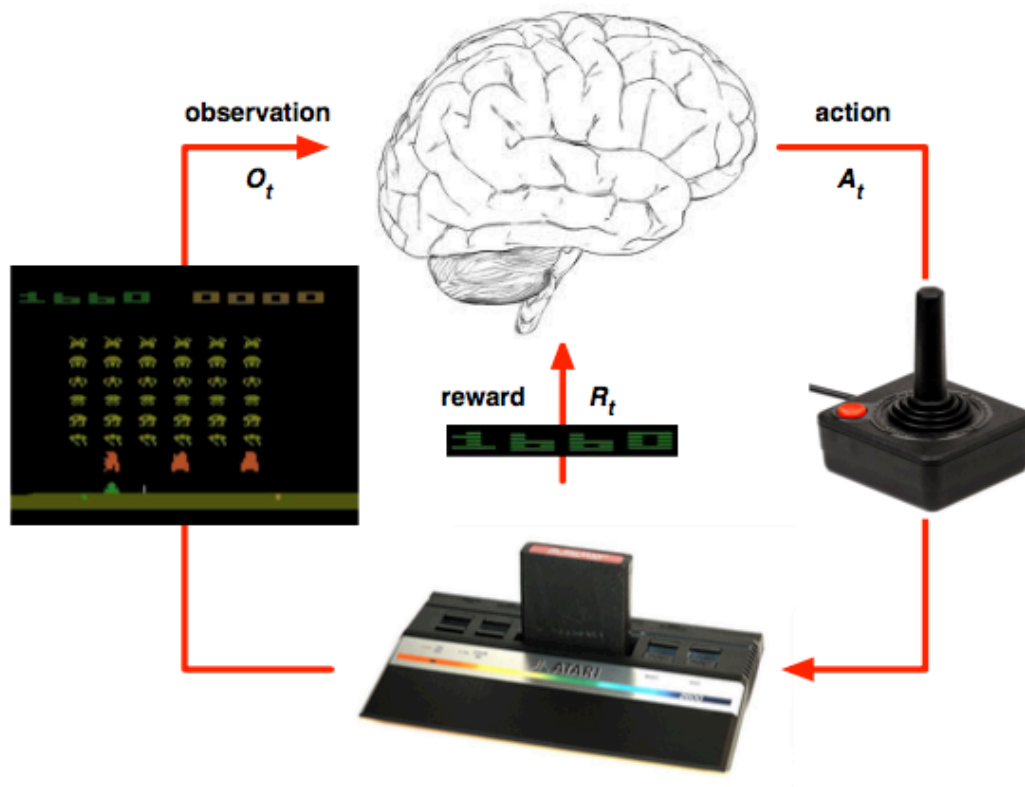
Deterministic Policy: $A_t = \pi(S_t)$

Stochastic Policy: $\pi(a|s) = P(A_t = a|S_t = s)$

Good policy: Leads to larger cumulative reward

Bad policy: Leads to worse cumulative reward
(we will explore this later)

Example: Atari



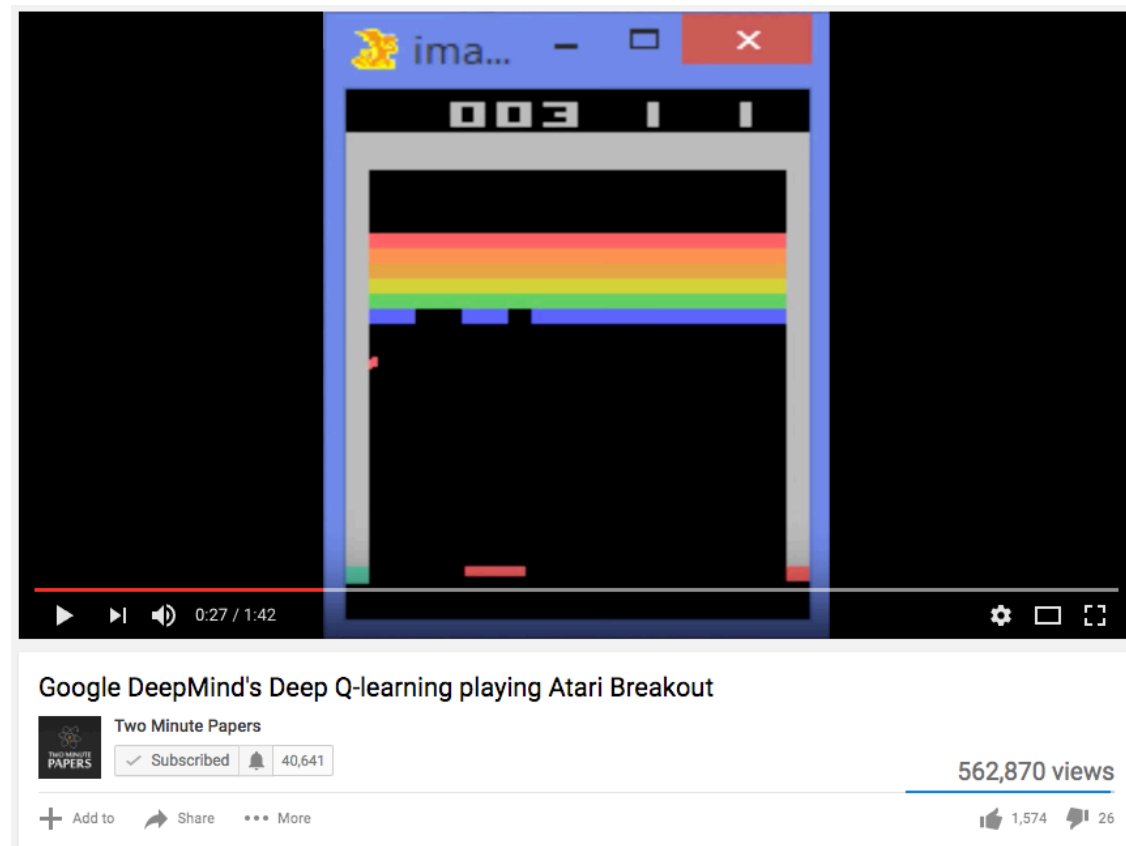
Rules are unknown

- What makes the score increase?

Dynamics are unknown

- How do actions change pixels?

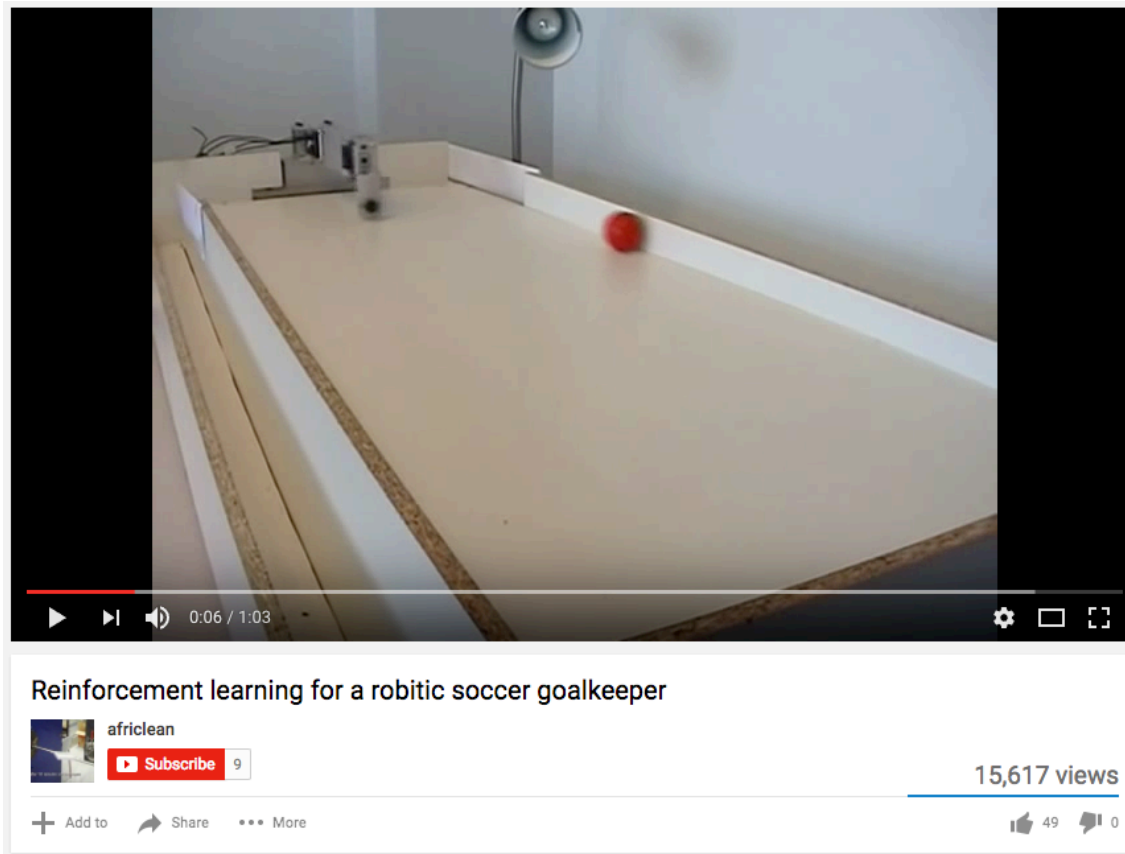
Video Time!



<https://www.youtube.com/watch?v=V1eYniJ0Rnk>

Example: Robotic Soccer

<https://www.youtube.com/watch?v=CIF2SBVY-J0>



AlphaGo

<https://www.youtube.com/watch?v=I2WFvGl4y8c>



Overview of Lecture



Machine Learning

Intro to Reinforcement Learning

Markov Processes

Markov Reward Processes

Markov Decision Processes

Markov Property

“The future is independent of the past given the present”

Markov Property

“The future is independent of the past given the present”

Definition

A state S_t is *Markov* if and only if

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, \dots, S_t]$$

Markov Property

“The future is independent of the past given the present”

Definition

A state S_t is *Markov* if and only if

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, \dots, S_t]$$

- The state captures all relevant information from the history
- Once the state is known, the history may be thrown away
- i.e. The state is a sufficient statistic of the future

State Transition Matrix

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

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

State Transition Matrix

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

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

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

$$\mathcal{P} = \begin{matrix} & \begin{matrix} \text{to} \\ \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{matrix} \\ \begin{matrix} \text{from} \end{matrix} & \begin{bmatrix} \end{bmatrix} \end{matrix}$$

where each row of the matrix sums to 1.

Markov Processes

A Markov process is a memoryless random process, i.e. a sequence of random states S_1, S_2, \dots with the Markov property.

Markov Processes

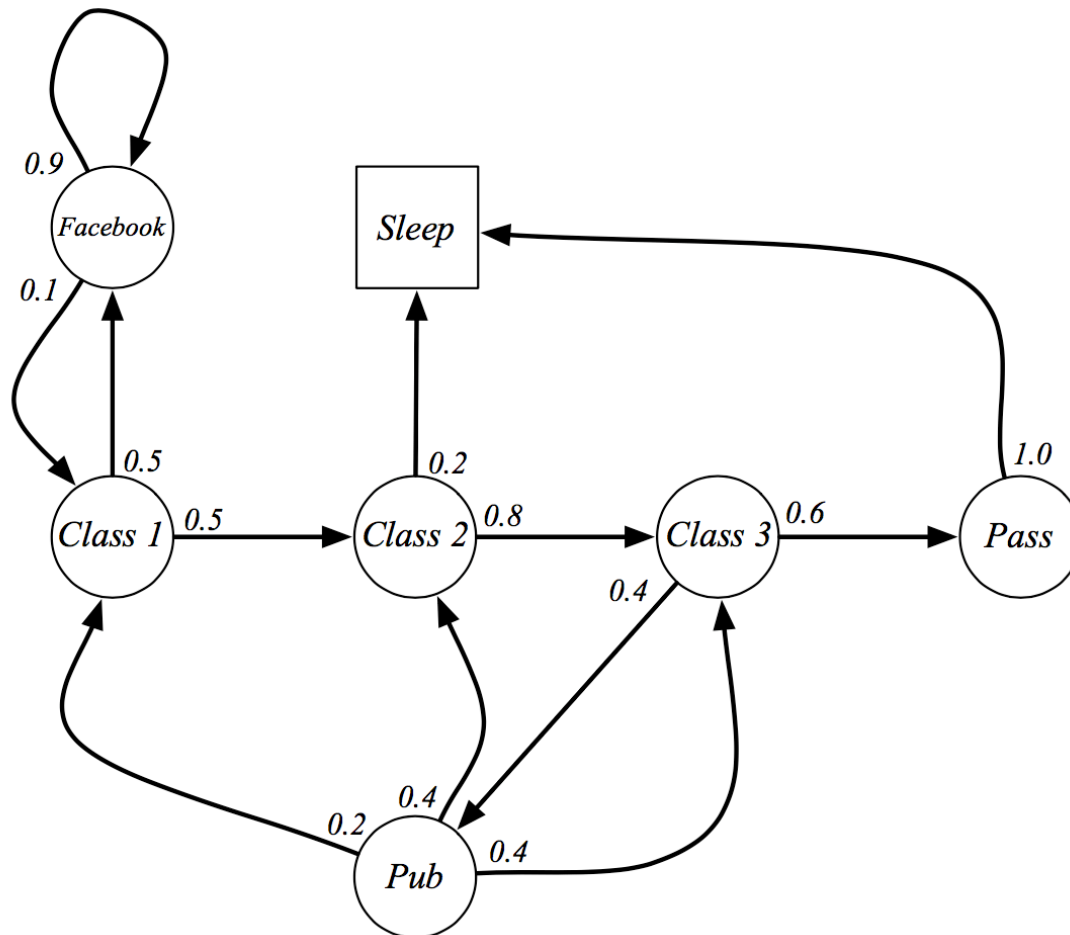
A Markov process is a memoryless random process, i.e. a sequence of random states S_1, S_2, \dots with the Markov property.

Definition

A *Markov Process* (or *Markov Chain*) is a tuple $\langle \mathcal{S}, \mathcal{P} \rangle$

- \mathcal{S} is a (finite) set of states
- \mathcal{P} is a state transition probability matrix,
$$\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$$

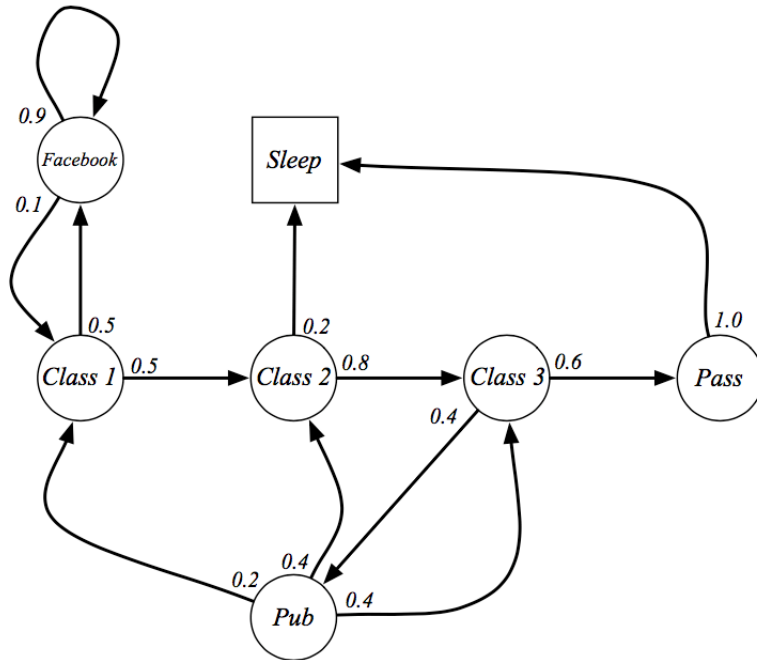
Student Markov Chain



Student MC: Episodes

Sample **episodes** for Student Markov Chain starting from $S_1 = C1$

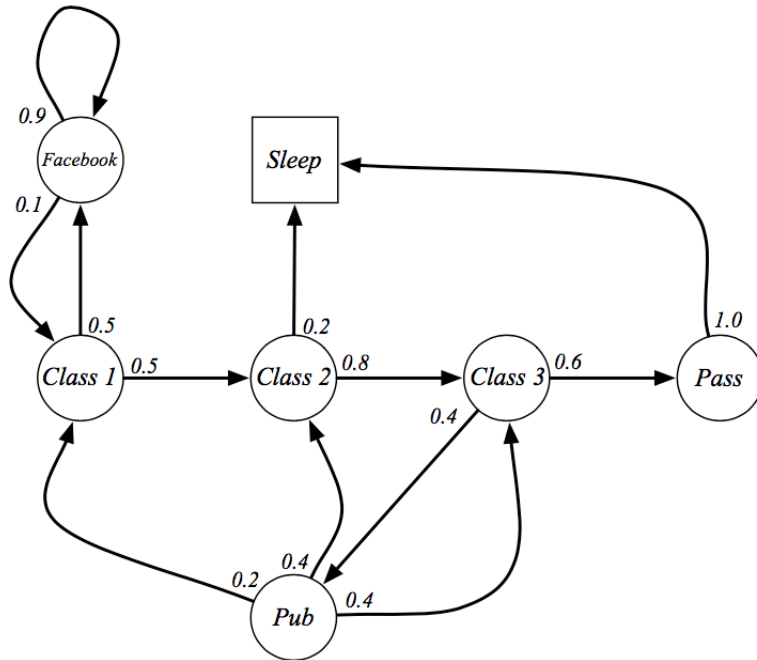
S_1, S_2, \dots, S_T



Student MC: Episodes

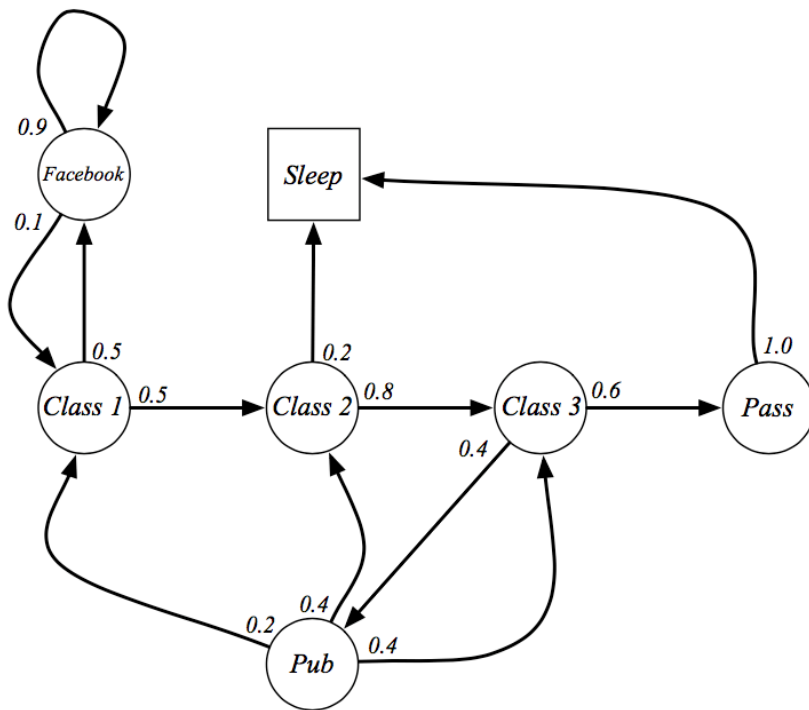
Sample **episodes** for Student Markov Chain starting from $S_1 = C1$

S_1, S_2, \dots, 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

Student MC: Transition Matrix



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

Machine Learning

Intro to Reinforcement Learning

Markov Processes

Markov Reward Processes

Markov Decision Processes

Markov Reward Process

A Markov reward process is a Markov chain with values.

Markov Reward Process

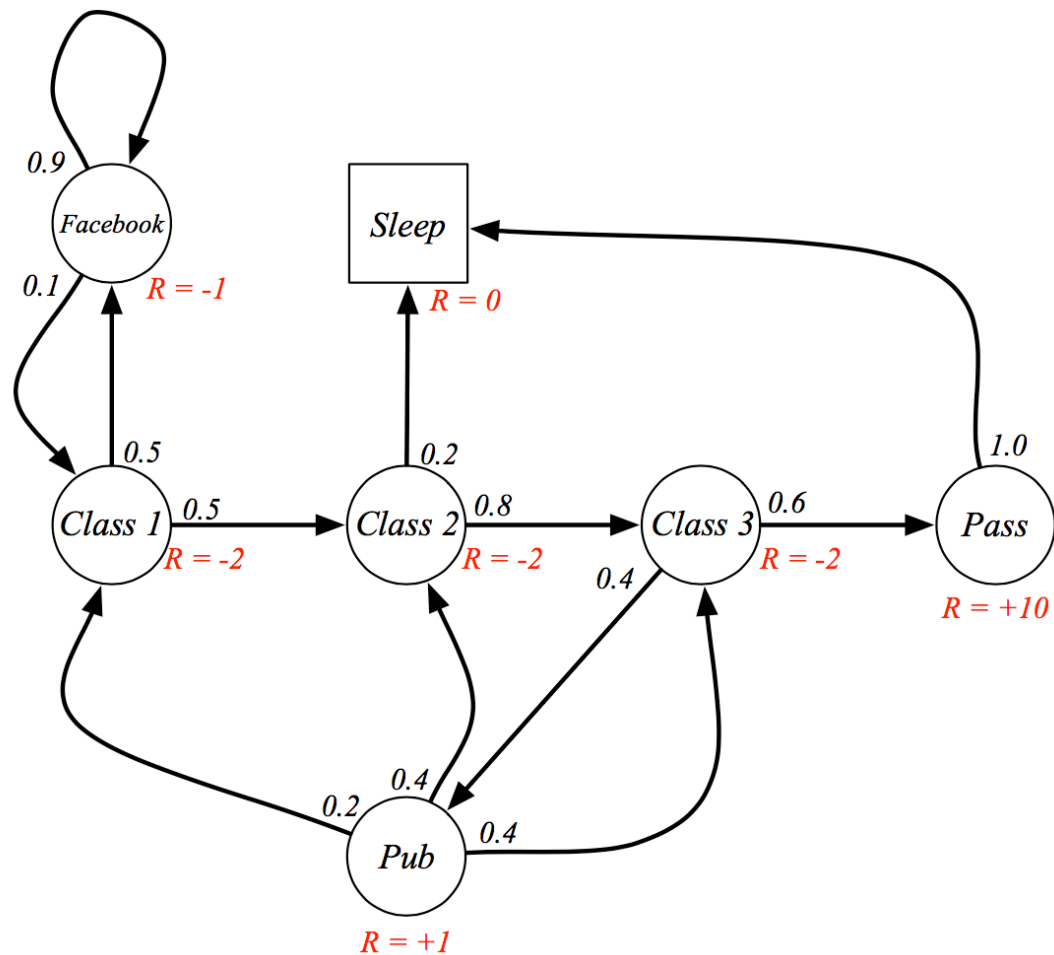
A Markov reward process is a Markov chain with values.

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,
 $\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$
- \mathcal{R} is a reward function, $\mathcal{R}_s = \mathbb{E}[R_{t+1} \mid S_t = s]$
- γ is a discount factor, $\gamma \in [0, 1]$

The Student MRP



Return

Definition

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

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

Return

Definition

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

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

- The *discount* $\gamma \in [0, 1]$ is the present value of future rewards
- The value of receiving reward R after $k + 1$ time-steps is $\gamma^k R$.

Return

Definition

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

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

- The *discount* $\gamma \in [0, 1]$ is the present value of future rewards
- The value of receiving reward R after $k + 1$ time-steps is $\gamma^k R$.
- This values immediate reward above delayed reward.
 - γ close to 0 leads to "myopic" evaluation
 - γ close to 1 leads to "far-sighted" evaluation

Why discount?

Most Markov reward and decision processes are discounted. Why?

Why discount?

Most Markov reward and decision processes are discounted. Why?

- Mathematically convenient to discount rewards
- Avoids infinite returns in cyclic Markov processes

Why discount?

Most Markov reward and decision processes are discounted. Why?

- Mathematically convenient to discount rewards
- Avoids infinite returns in cyclic Markov processes
- Uncertainty about the future may not be fully represented
- If the reward is financial, immediate rewards may earn more interest than delayed rewards

Why discount?

Most Markov reward and decision processes are discounted. Why?

- Mathematically convenient to discount rewards
- Avoids infinite returns in cyclic Markov processes
- Uncertainty about the future may not be fully represented
- If the reward is financial, immediate rewards may earn more interest than delayed rewards
- Animal/human behaviour shows preference for immediate reward

Why discount?

Most Markov reward and decision processes are discounted. Why?

- Mathematically convenient to discount rewards
- Avoids infinite returns in cyclic Markov processes
- Uncertainty about the future may not be fully represented
- If the reward is financial, immediate rewards may earn more interest than delayed rewards
- Animal/human behaviour shows preference for immediate reward
- It is sometimes possible to use *undiscounted* Markov reward processes (i.e. $\gamma = 1$), e.g. if all sequences terminate.

Value Function

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

Value Function

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

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

Student MRP: Returns

Sample **returns** for Student MRP:

Starting from $S_1 = C1$ with $\gamma = \frac{1}{2}$

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

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

Student MRP: Returns

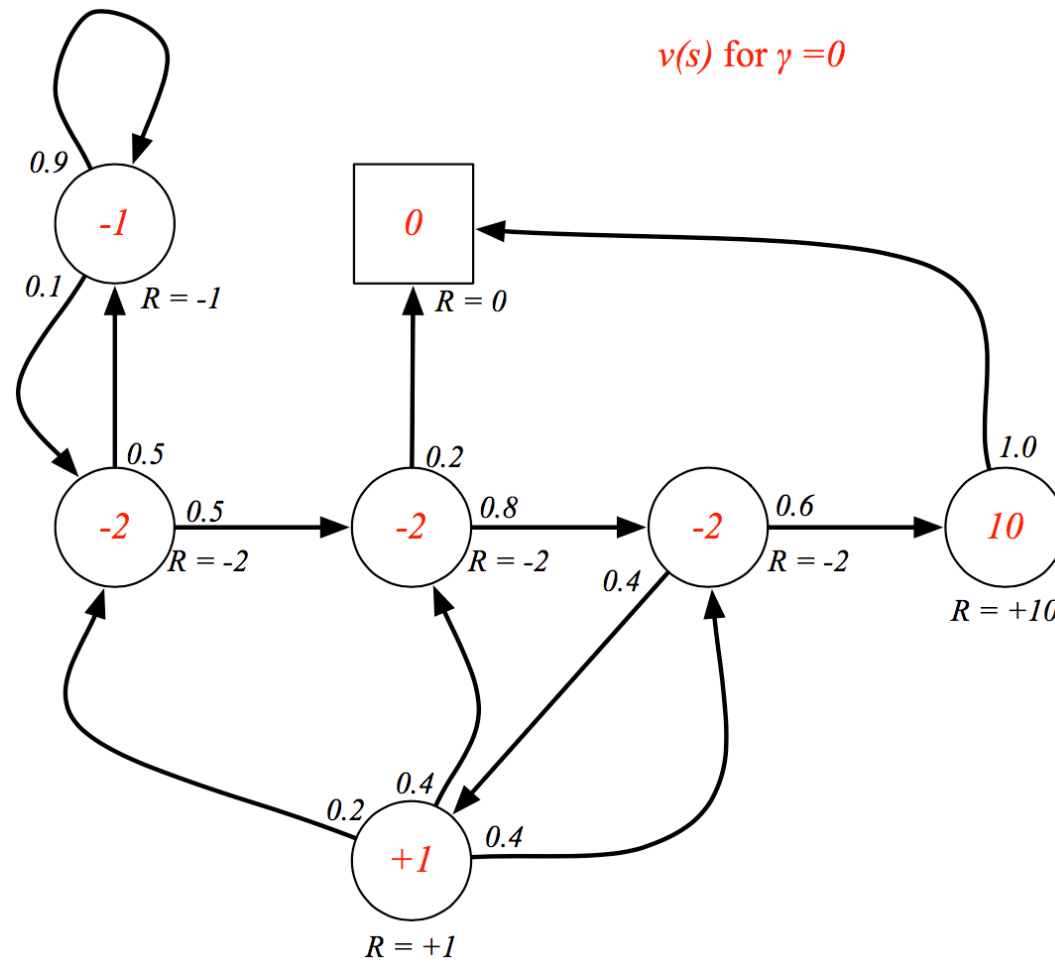
Sample **returns** for Student MRP:

Starting from $S_1 = C1$ with $\gamma = \frac{1}{2}$

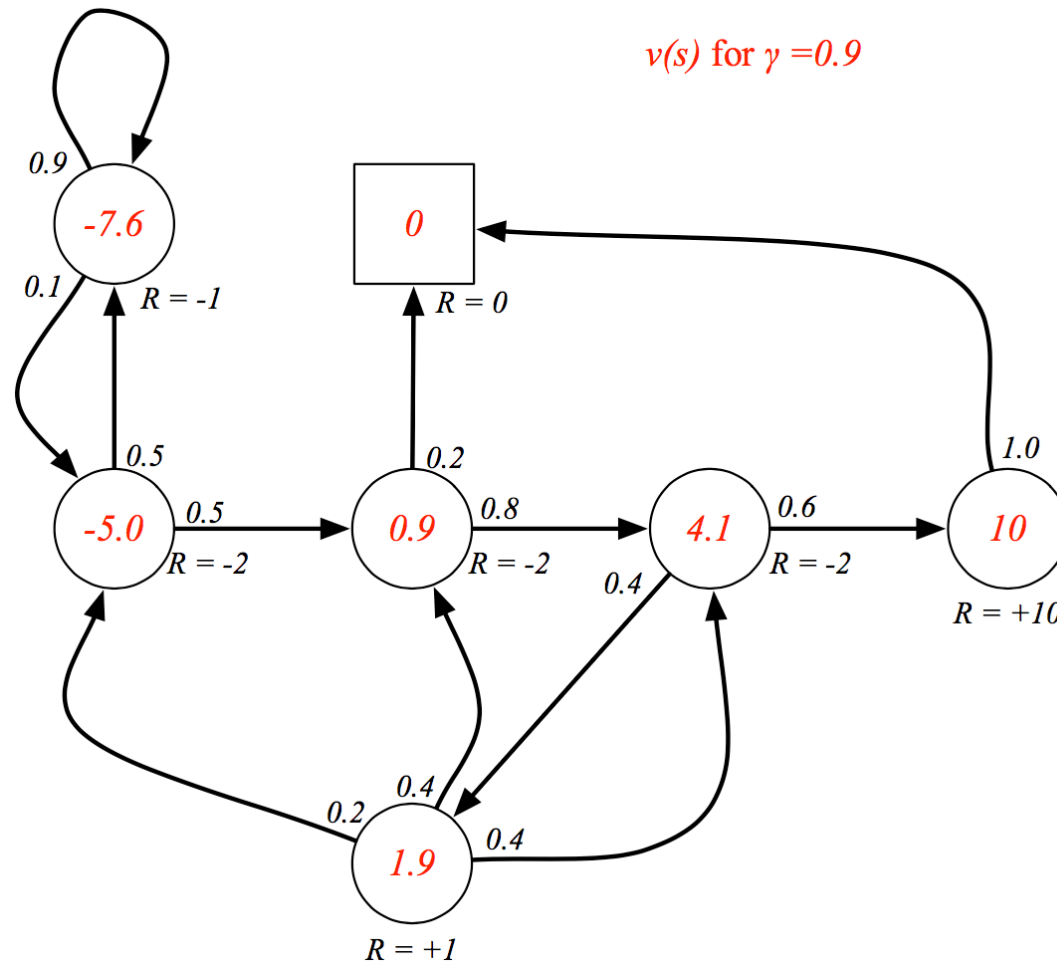
$$G_1 = R_2 + \gamma R_3 + \dots + \gamma^{T-2} R_T$$

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

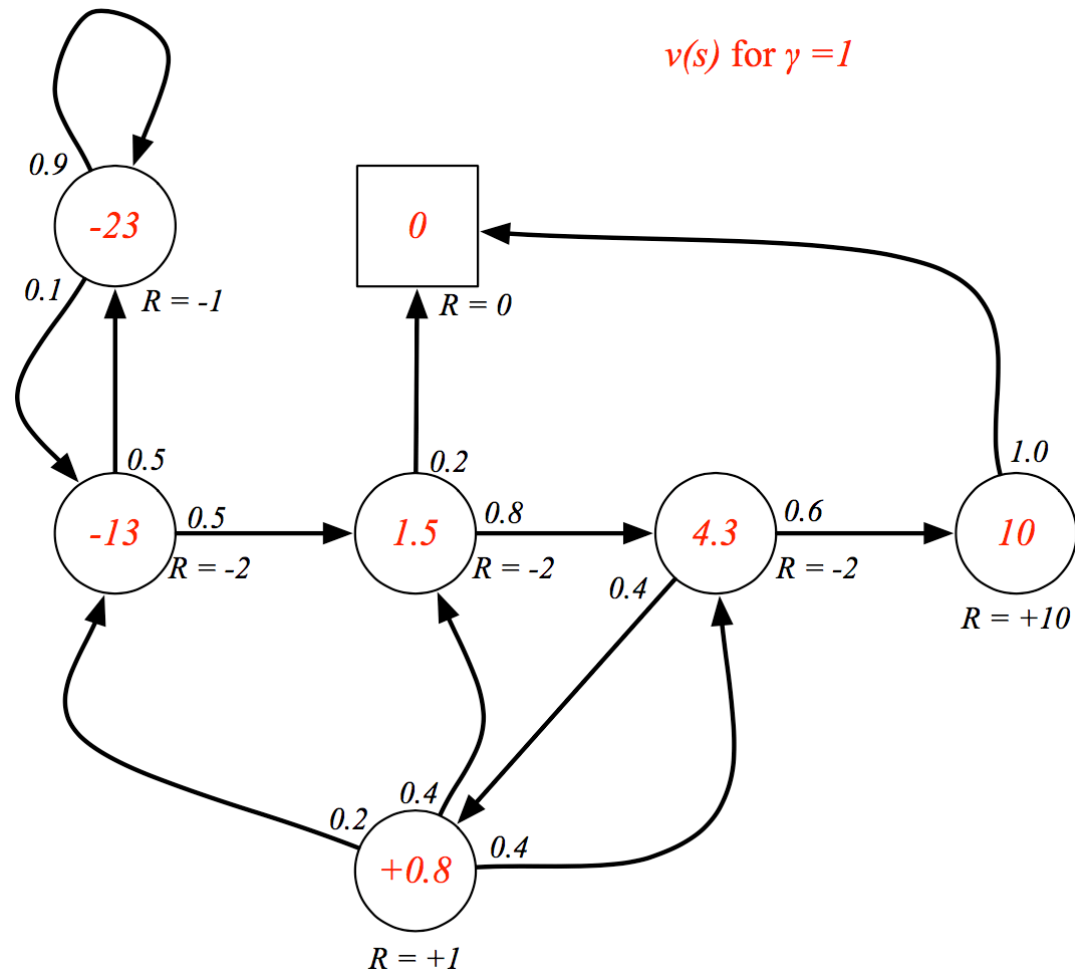
Student MRP: Value Function



Student MRP: Value Function



Student MRP: Value Function



Bellman Equations for MRP

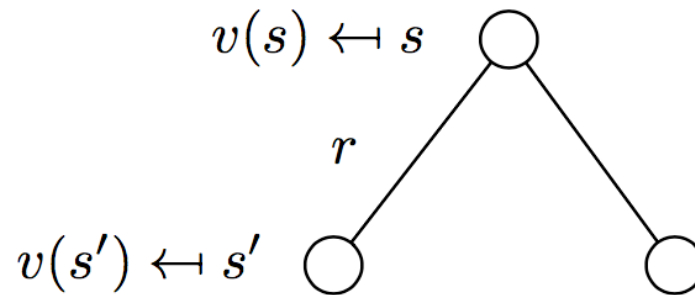
The value function can be decomposed into two parts:

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

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

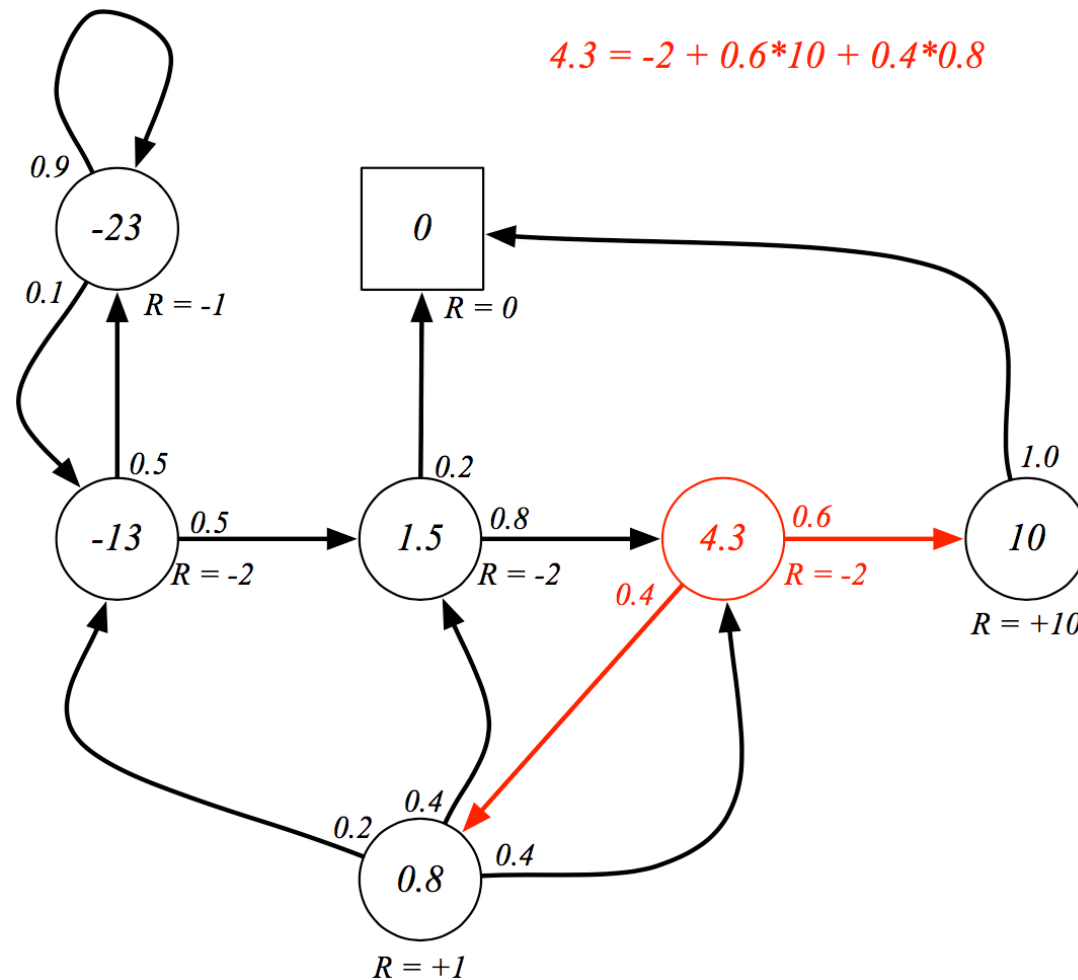
Backup Diagrams for MRP

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



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

Student MRP: Bellman Eq



Matrix Form of Bellman Eq

The Bellman equation can be expressed concisely using matrices,

$$v = \mathcal{R} + \gamma \mathcal{P}v$$

where v is a column vector with one entry per state

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

Solving the Bellman Equation

- The Bellman equation is a linear equation
- It can be solved directly:

$$\begin{aligned}v &= \mathcal{R} + \gamma \mathcal{P}v \\(I - \gamma \mathcal{P})v &= \mathcal{R} \\v &= (I - \gamma \mathcal{P})^{-1} \mathcal{R}\end{aligned}$$

Solving the Bellman Equation

- The Bellman equation is a linear equation
- It can be solved directly:

$$\begin{aligned}v &= \mathcal{R} + \gamma \mathcal{P} v \\(I - \gamma \mathcal{P}) v &= \mathcal{R} \\v &= (I - \gamma \mathcal{P})^{-1} \mathcal{R}\end{aligned}$$

- Computational complexity is $O(n^3)$ for n states
- Direct solution only possible for small MRPs
- There are many iterative methods for large MRPs, e.g.
 - Dynamic programming
 - Monte-Carlo evaluation
 - Temporal-Difference learning

Machine Learning

Intro to Reinforcement Learning

Markov Processes

Markov Reward Processes

Markov Decision Processes

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.

Markov Decision Process

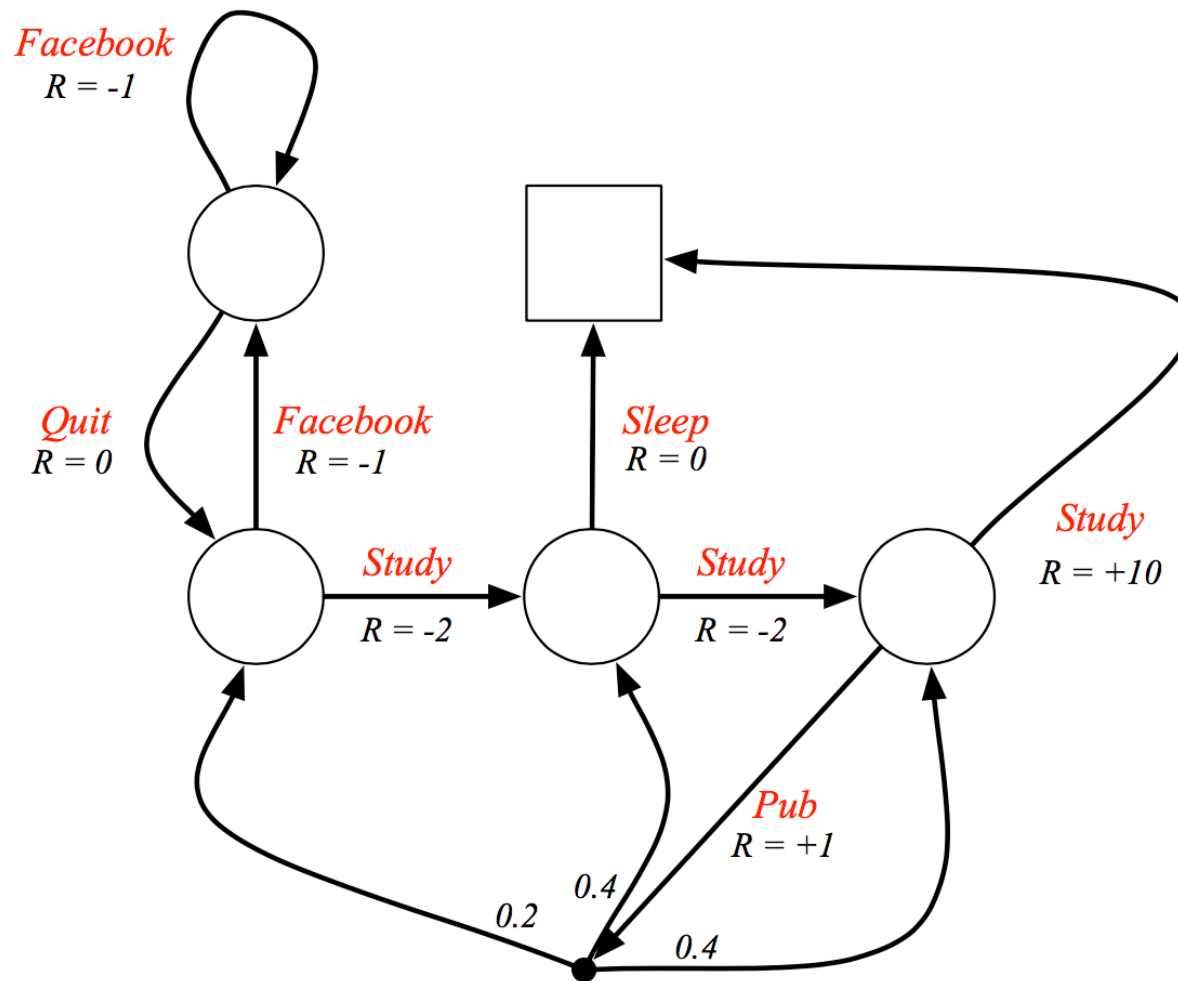
A Markov decision process (MDP) is a Markov reward process with decisions. It is an *environment* in which all states are Markov.

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,
 $\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
- \mathcal{R} is a reward function, $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
- γ is a discount factor $\gamma \in [0, 1]$.

The Student MDP



Policies

Definition

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

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

Policies

Definition

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

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

- A policy fully defines the behaviour of an agent
- MDP policies depend on the current state (not the history)

Policies

Definition

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

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

- A policy fully defines the behaviour of an agent
- MDP policies depend on the current state (not the history)
- i.e. Policies are *stationary* (time-independent),
 $A_t \sim \pi(\cdot|S_t), \forall t > 0$

MPs \rightarrow MRPs \rightarrow MDPs

- Given an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and a policy π

MPs \rightarrow MRPs \rightarrow MDPs

- 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, \dots is a Markov process $\langle \mathcal{S}, \mathcal{P}^\pi \rangle$
- The state and reward sequence S_1, R_2, S_2, \dots is a Markov reward process $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$
- where

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

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

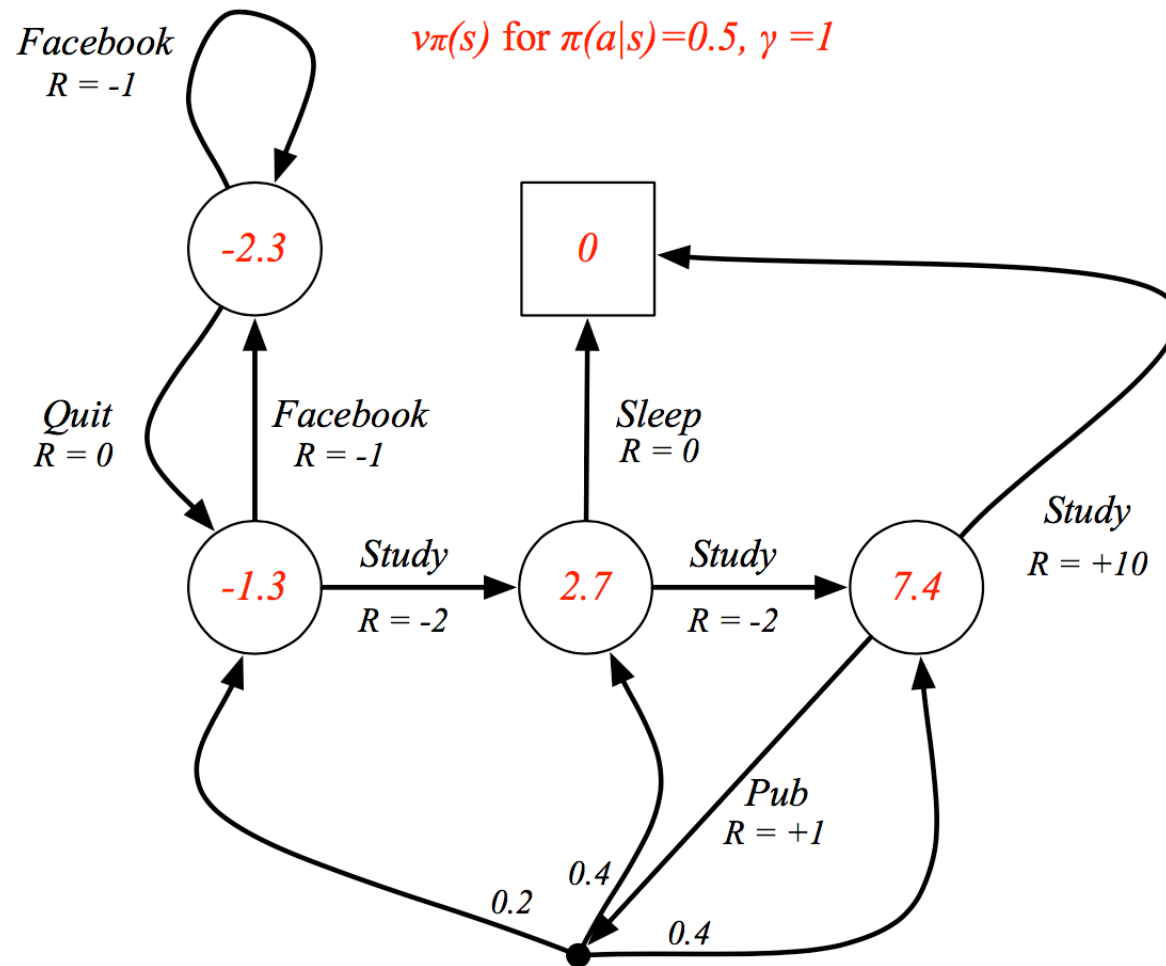
Value Function

Definition

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

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

Student MDP: Value Function

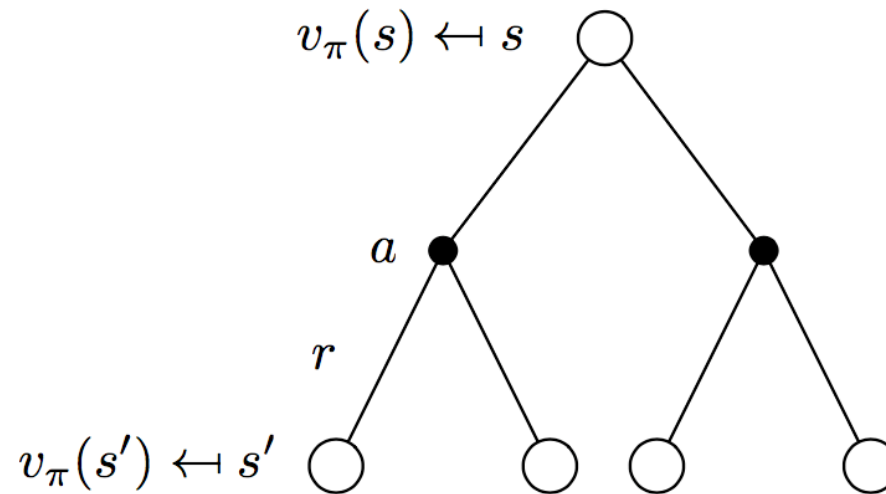


Bellman Expected Equation

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

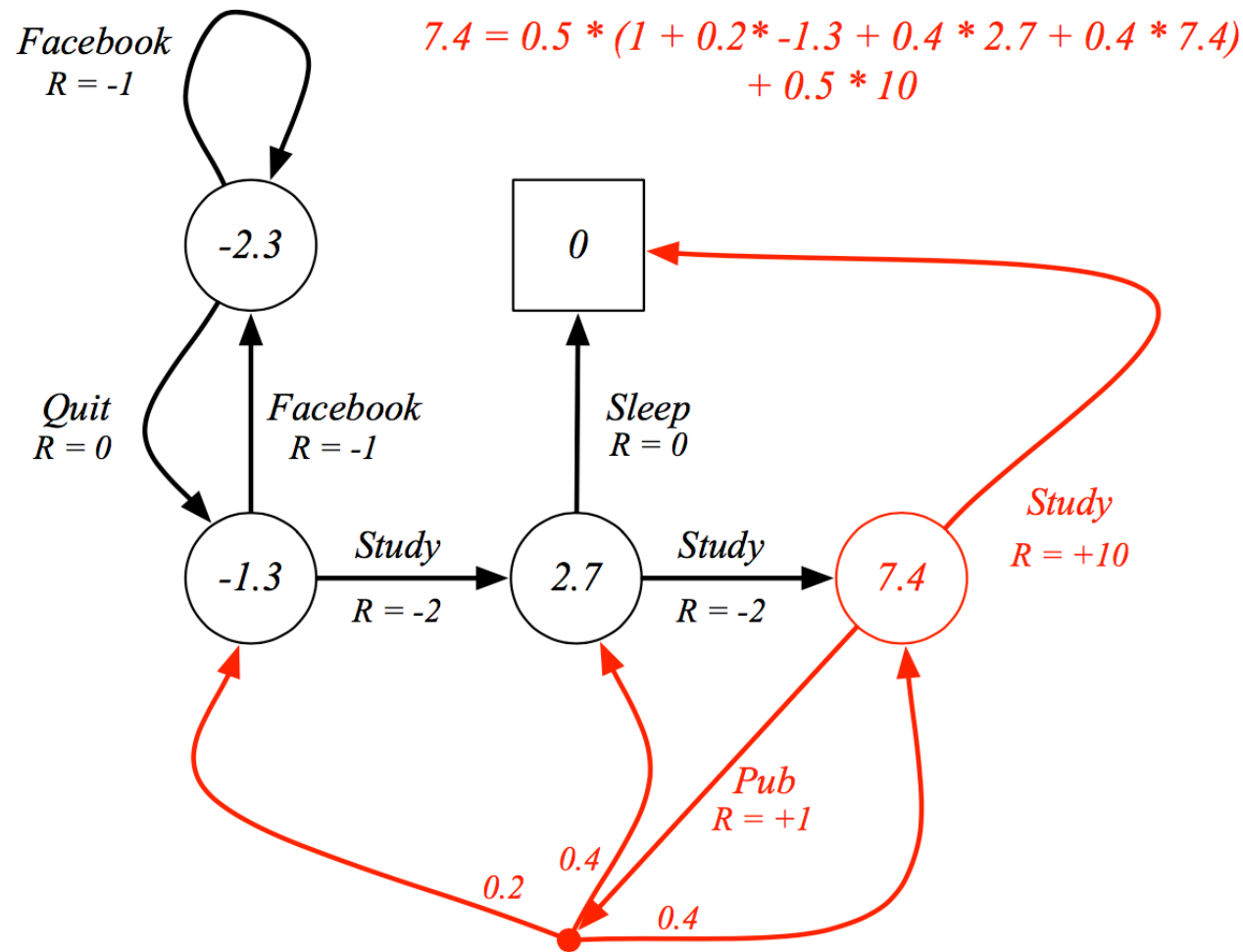
$$v_{\pi}(s) = \mathbb{E}_{\pi} [R_{t+1} + \gamma v_{\pi}(S_{t+1}) \mid S_t = s]$$

Bellman Expected Equation, V



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

Student MDP: Bellman Exp Eq.



Bellman Exp Eq: Matrix Form

The Bellman expectation equation can be expressed concisely using the induced MRP,

$$v_{\pi} = \mathcal{R}^{\pi} + \gamma \mathcal{P}^{\pi} v_{\pi}$$

with direct solution

$$v_{\pi} = (I - \gamma \mathcal{P}^{\pi})^{-1} \mathcal{R}^{\pi}$$

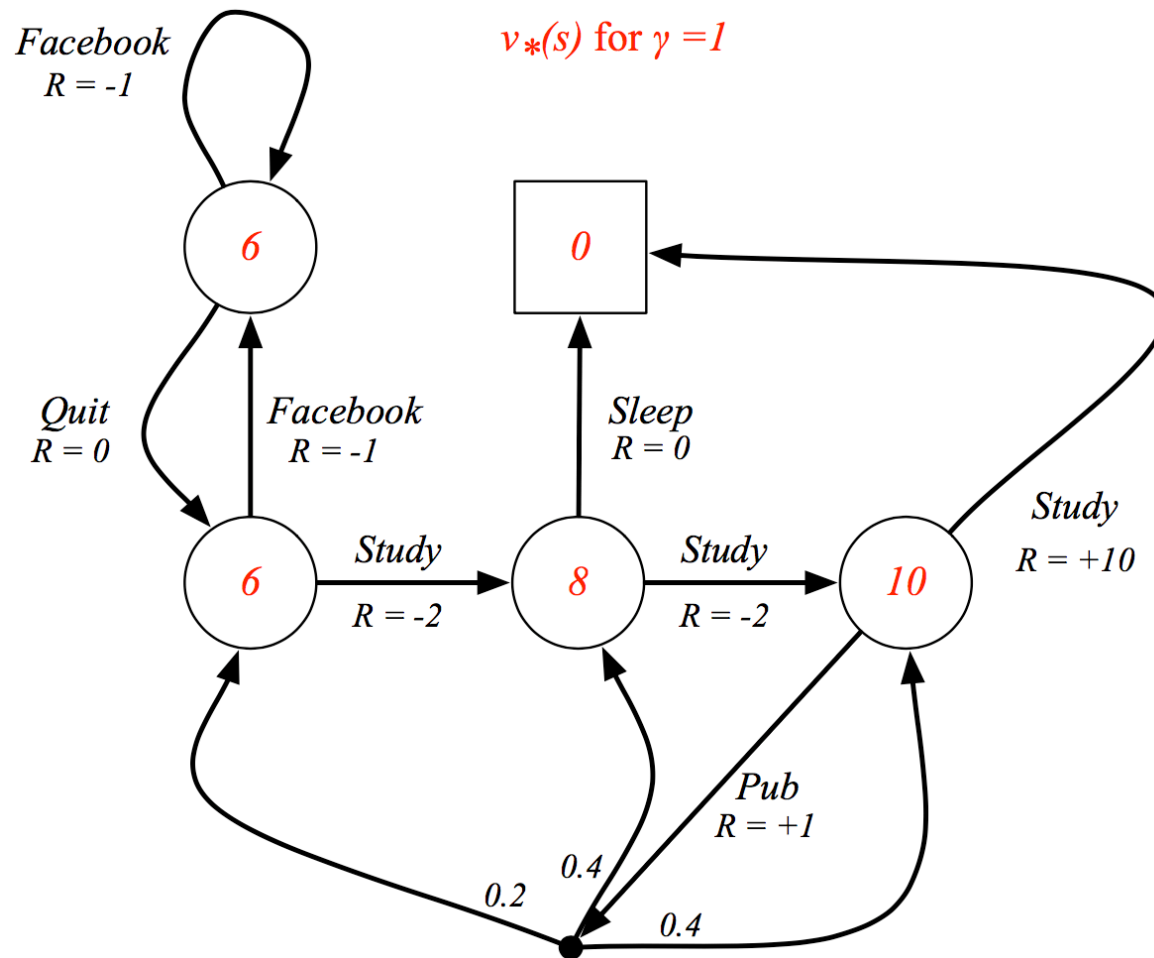
Optimal Value Function

Definition

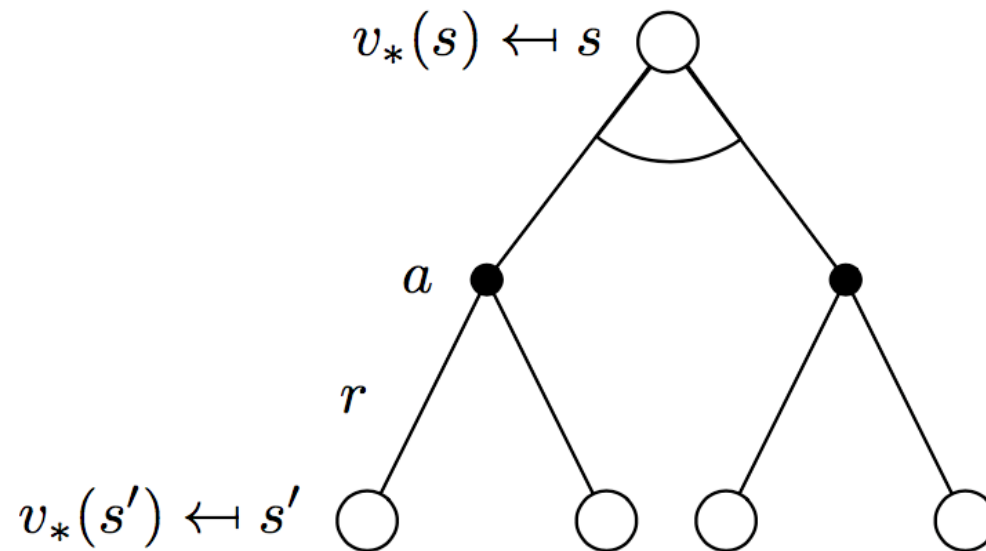
The *optimal state-value function* $v_*(s)$ is the maximum value function over all policies

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

Student MDP: Optimal V

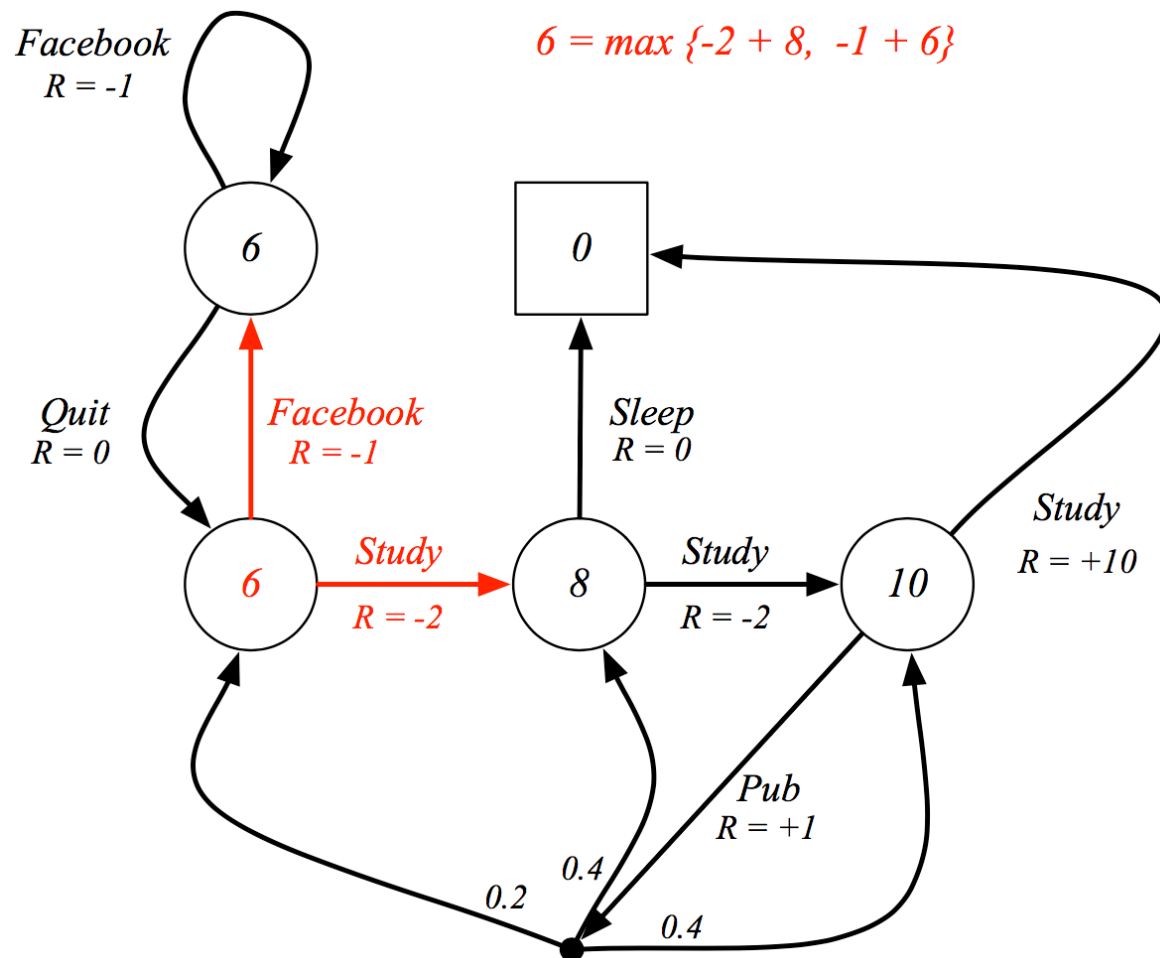


Bellman Optimality Eq, V



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

Student MDP: Bellman Optimality



From MDPs to RL

	Evaluate Policy, π	Find Best Policy, π^*
MDP Known	Policy Evaluation	Policy/Value Iteration
MDP Unknown	MC and TD Learning	Sarsa + Q-Learning

Improving a Policy!

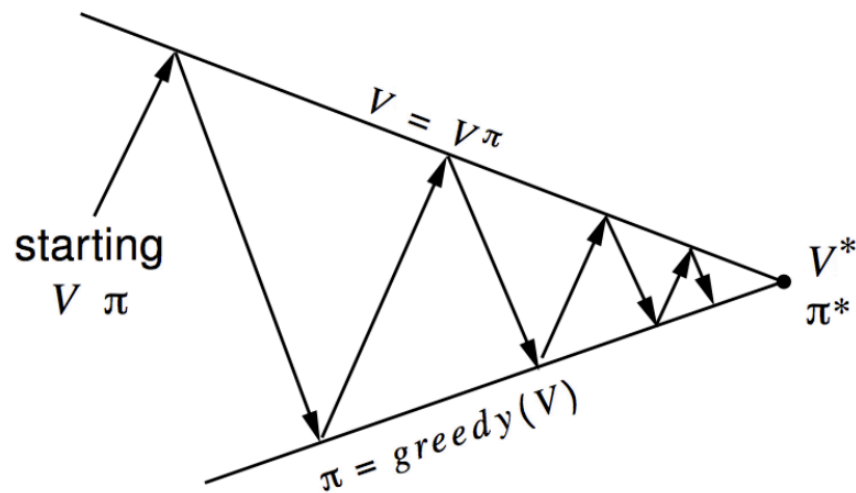
- Given a policy π
 - **Evaluate** the policy π

$$v_{\pi}(s) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \dots | S_t = s]$$

- **Improve** the policy by acting greedily with respect to v_{π}

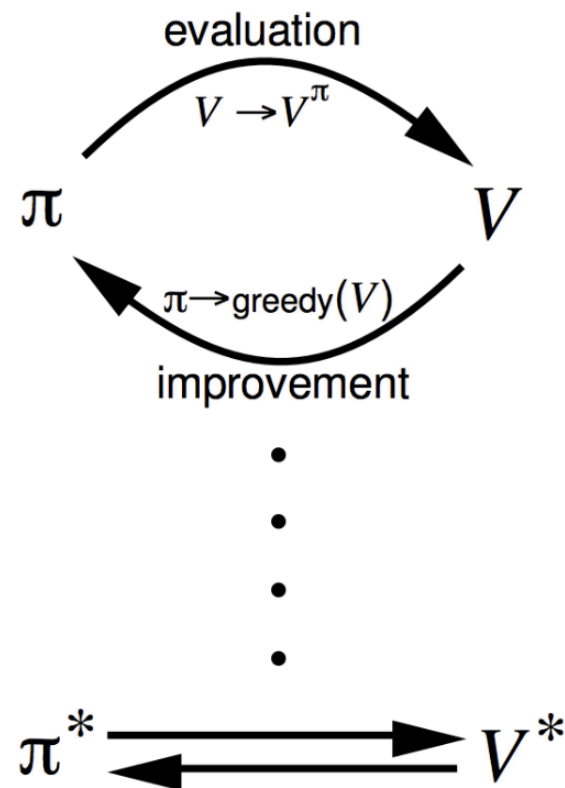
$$\pi' = \text{greedy}(v_{\pi})$$

Policy Iteration



Policy evaluation Estimate v_π
 Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$
 Greedy policy improvement



From MDPs to RL

	Evaluate Policy, π	Find Best Policy, π^*
MDP Known	Policy Evaluation	Policy/Value Iteration
MDP Unknown	MC and TD Learning	Sarsa + Q-Learning

Monte Carlo RL

- MC methods learn directly from episodes of experience
- MC is *model-free*: no knowledge of MDP transitions / rewards

Monte Carlo Policy Evaluation

- Goal: learn v_π from episodes of experience under policy π

$$S_1, A_1, R_2, \dots, S_k \sim \pi$$

- Recall that the *return* is the total discounted reward:

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$$

- Recall that the value function is the expected return:

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

- Monte-Carlo policy evaluation uses *empirical mean* return instead of *expected* return

Every-Visit MC Policy Evaluation

- To evaluate state s
- **Every** time-step t that state s is visited in an episode,
- Increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- Again, $V(s) \rightarrow v_{\pi}(s)$ as $N(s) \rightarrow \infty$

Blackjack Example

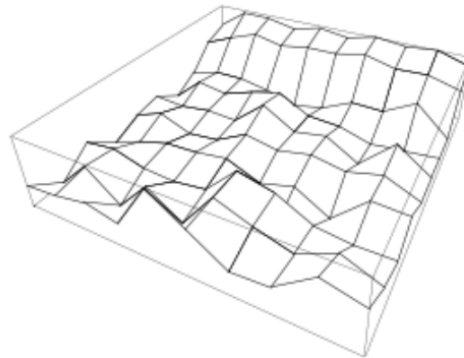
- States (200 of them):
 - Current sum (12-21)
 - Dealer's showing card (ace-10)
 - Do I have a "useable" ace? (yes-no)
- Action **stand** Stop receiving cards (and terminate)
- Action **hit** : Take another card (no replacement)
- Reward for **stand**
 - +1 if sum of cards $>$ sum of dealer cards
 - 0 if sum of cards = sum of dealer cards
 - -1 if sum of cards $<$ sum of dealer cards
- Reward for **hit** :
 - -1 if sum of cards $>$ 21 (and terminate)
 - 0 otherwise
- Transitions: automatically **hit** if sum of cards $<$ 12



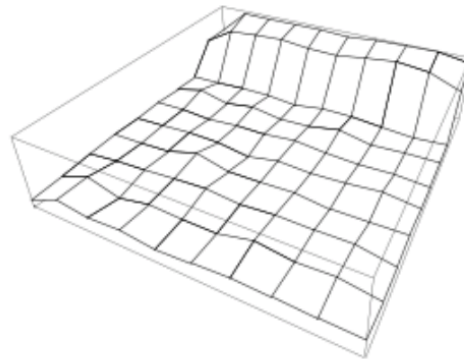
Blackjack Value Function

After 10,000 episodes

Usable
ace

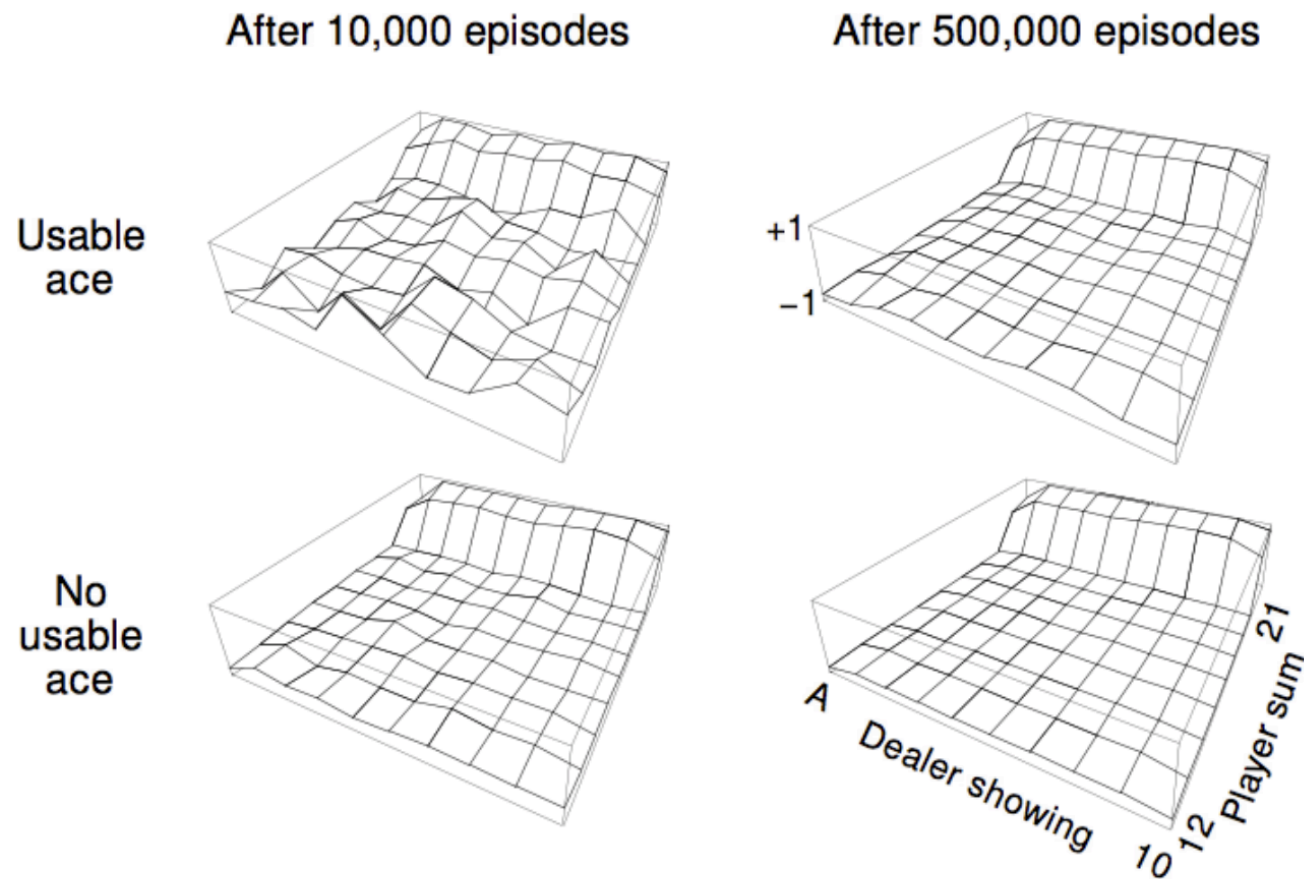


No
usable
ace



Policy: **stand** if sum of cards ≥ 20 , otherwise **hit**

Blackjack Value Function



Policy: **stand** if sum of cards ≥ 20 , otherwise **hit**