# Introduction to reinforcement learning

Urtzi Ayesta and Matthieu Jonckheere

CNRS-IRIT & CNRS-LAAS

# Markov reward processes and Markov decision processes

CHANGE OF NOTATIONS (we follow Barto and Sutton from now on).

# Information - Set of policies

Markov Decision Process (MDP)

- observable state and reward
- known reward distribution and transition probabilities
- Action depends on current state and action, $p(s'|s, a)$

Partially Observable Markov Decision Process (POMDP)

- Partially observable state: we know $z_t$ with known $P(s_t = s|z)$
- Observed rewards
- Known reward distribution and transition probabilities

Reinforcement learning

- observable state and reward
- Unknown reward distribution, unknown transition probabilities

Adversarial problems

- observable state and reward
- Arbitrary and time-varying reward function and state transitions

# Classification of problems

Known model

**MDP, POMDP**

Unknown model

**Reinforcement learning**

Absence of model

**Adversarial**

# Major components of an RL Agent

- Policy: A map from states to actions,
  $\pi(a|s) = \mathbb{P}(A_t = a|S_t = s)$
- Value function: Prediction of future rewards

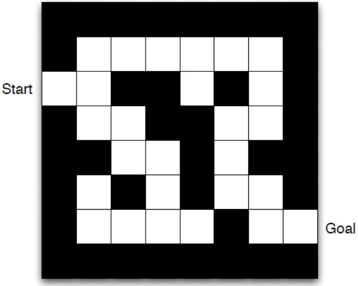
$$v_\pi(s) = \mathbb{E}_\pi(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots |S_t = s)$$

- A model predicts what the environment will do next

$$p(s'|s, a) = \mathbb{P}(S_{t+1} = s'|S_t = s, A_t = a)$$

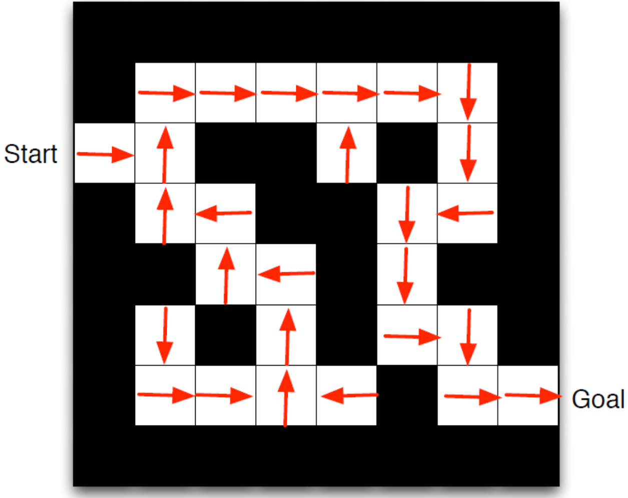
$$r(s, a) = \mathbb{E}(R_{t+1}|S_t = s, A_t = a)$$

# Maze Example



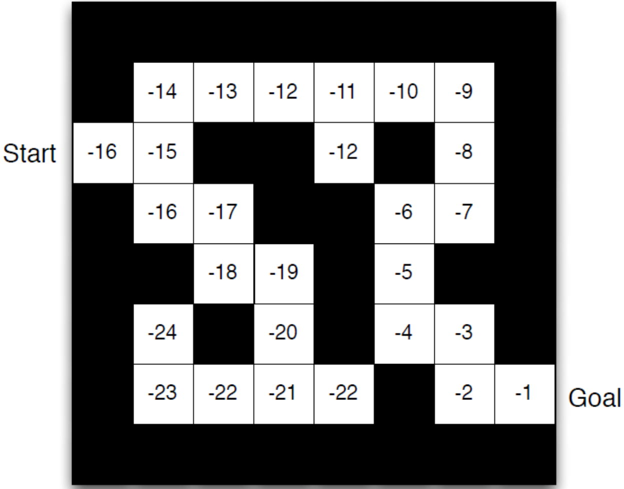

Rewards: -1 per time-step
Actions: N,E,S, W
States: Agent's location

# Maze Example: Policy



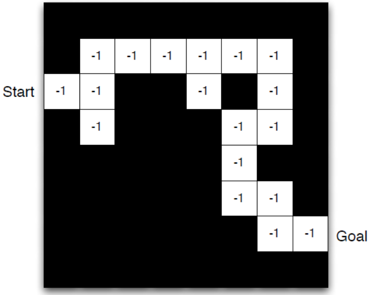

Arrows represent policy $\pi(s)$

# Maze Example: Value function




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

# Maze Example: Model




Dynamics: how actions change state

Rewards: From visited states

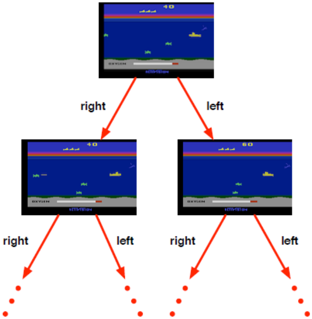
Model may be imperfect

# Learning and planning

- Planning. A perfect model of the environment is known
  $\Longrightarrow$ MDP

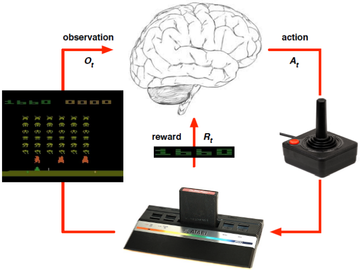
- Reinforcement learning
  - Environment is initially unknown
  - Agent interacts with the environment
  - Agent learns policy

# Atari Example: Planning



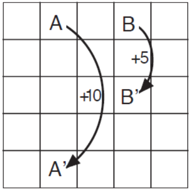

- Rules known
- Can query emulator
- If I take action $a$ in state $s$, what would the score and next state be?
- Tree search to find optimal policy

# Atari Example: Learning




- Rules unknown
- Learn directly from game-play
- Pick actions on joystic, see pixels and scores

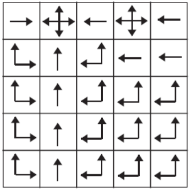
# Prediction and Control




a) gridworld

| | | | | |
|---|---|---|---|---|
| 22.0 | 24.4 | 22.0 | 19.4 | 17.5 |
| 19.8 | 22.0 | 19.8 | 17.8 | 16.0 |
| 17.8 | 19.8 | 17.8 | 16.0 | 14.4 |
| 16.0 | 17.8 | 16.0 | 14.4 | 13.0 |
| 14.4 | 16.0 | 14.4 | 13.0 | 11.7 |

b) $v_*$



c) $\pi_*$

# Markov Process

A Markov Process is a sequence of random states $S_1, S_2, \ldots,$ with the Markov property

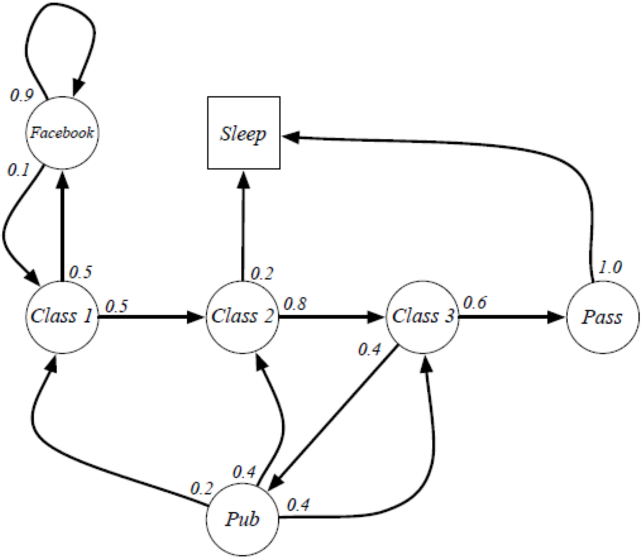
**Definition**

A Markov Chain is a tuple $\langle \mathcal{S}, \mathcal{P} \rangle$

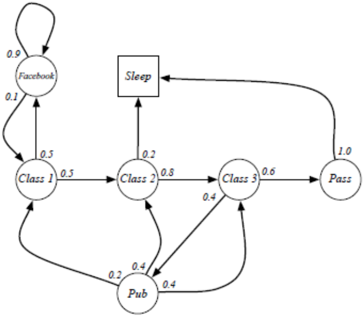
- $\mathcal{S}$ is a (finite) set of states
- is a state transition probability matrix

$$p(s, s') = \mathbb{P}(S_{t+1} = s' | S_t = s)$$

# Example: Student Markov Chain



# Example: Student Markov Chain Transition Matrix




$$\mathcal{P} = \begin{array}{rc} & \begin{array}{ccccccc} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{array} \\ \begin{array}{r} C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{array} & \left[\begin{array}{ccccccc} & 0.5 & & & & 0.5 & \\ & & 0.8 & & & & 0.2 \\ & & & 0.6 & 0.4 & & \\ & & & & & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1 \end{array}\right] \end{array}$$

# Markov Reward Process

A Markov Reward Process is a Markov Chain with values

**Definition**

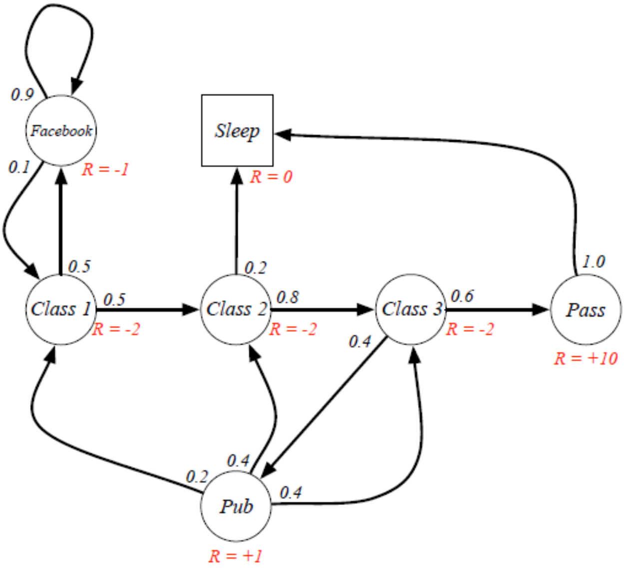
Markov Reward Process is a tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{P}$ is a state transition probability matrix

$$p(s, s') = \mathbb{P}(S_{t+1} = s' | S_t = s)$$

- $\mathcal{R}$ is a reward function $r(s) = \mathbb{E}(R_{t+1} | S_t = s)$
- $\gamma \in [0, 1]$ is a discount factor

# Example: Student MRP



# Markov Reward Process

A Markov Reward Process is a Markov Chain with values

**Definition**

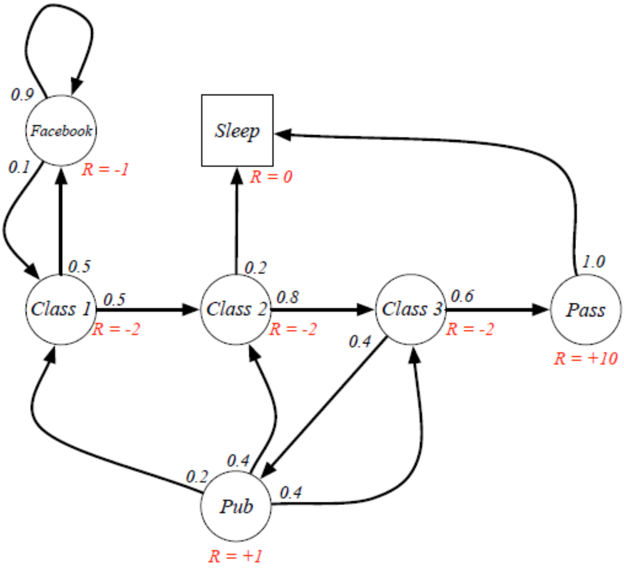
Markov Reward Process is a tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{P}$ is a state transition probability matrix

$$p(s, s') = \mathbb{P}(S_{t+1} = s' | S_t = s)$$

- $\mathcal{R}$ is a reward function $r(s) = \mathbb{E}(R_{t+1} | S_t = s)$
- $\gamma \in [0, 1]$ is a discount factor

# Example: Student MRP



# Return

**Definition:**

The return $G_t$ is the total discounted reward from time-step $t$ on:

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

- The discount $\gamma$ is the present value of future rewards
- The value of receiving reward $R$ after $k+1$ time-steps is $\gamma^k R$
- Immediate rewards more relevant than future ones
  - $\gamma$ close to $0$ referred as "myopic"
  - $\gamma$ close to $1$ referred as "far-sighted"

# Why discount?

Most Markov reward and decision processes are discounted. Why?

- Mathematically convenient to discount rewards
- Avoids infinite returns in non-absorving Markov processes
- If the reward is financial, immediate rewards may earn more interest than delayed rewards
- Animal/human behaviour shows preference for immediate reward

# Value function

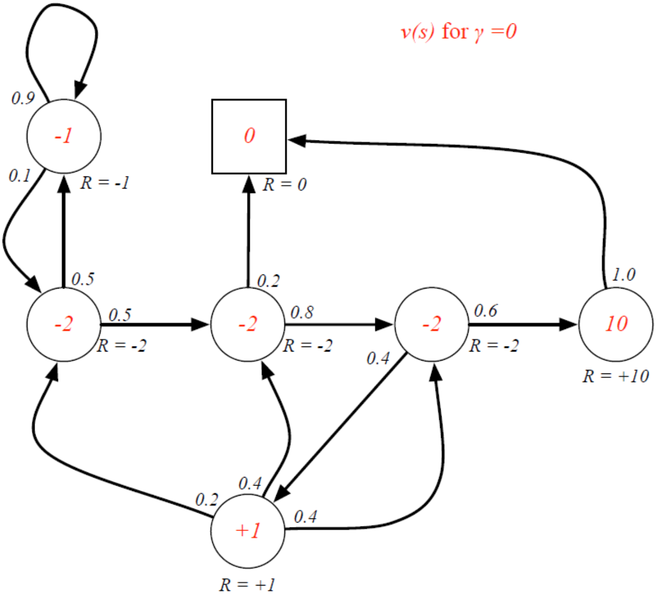
The value function $V(s)$ gives the long-run term value of state $s$

**Definition:**

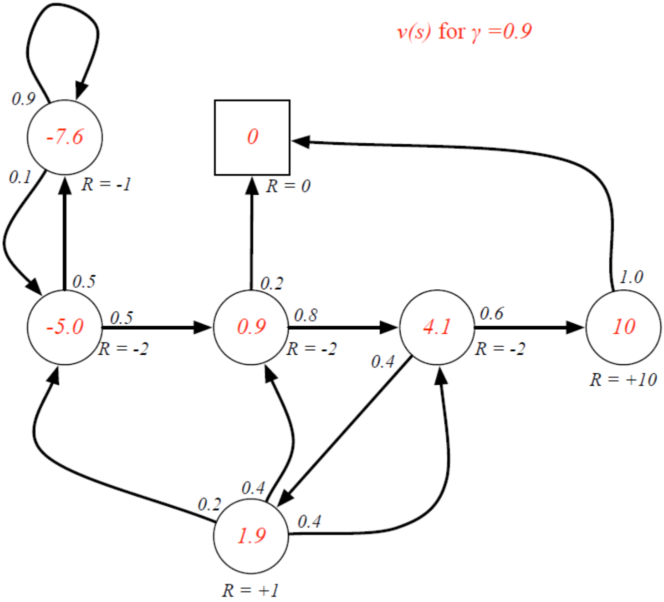
The state value function $V(s)$ of an MRP is the expected return starting from state $s$:

$$V(s) = \mathbb{E}(G_t | S_t = s)$$

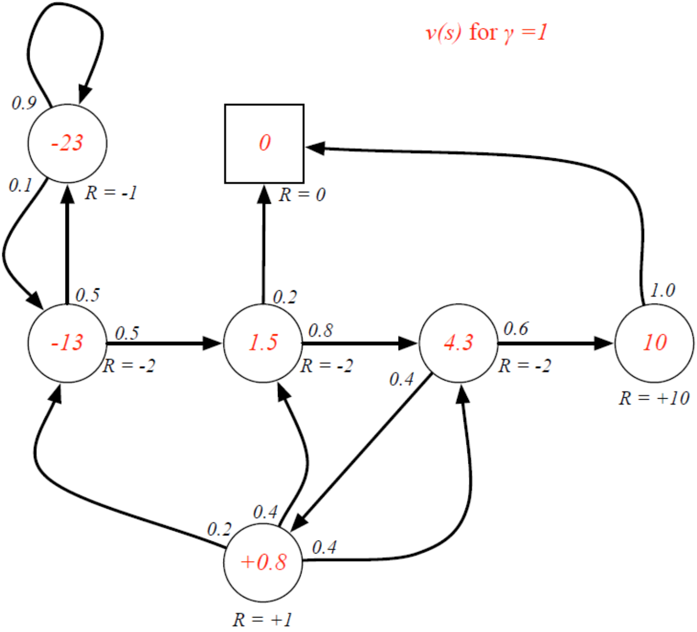
# Example: Student MRP



# Example: Student MRP



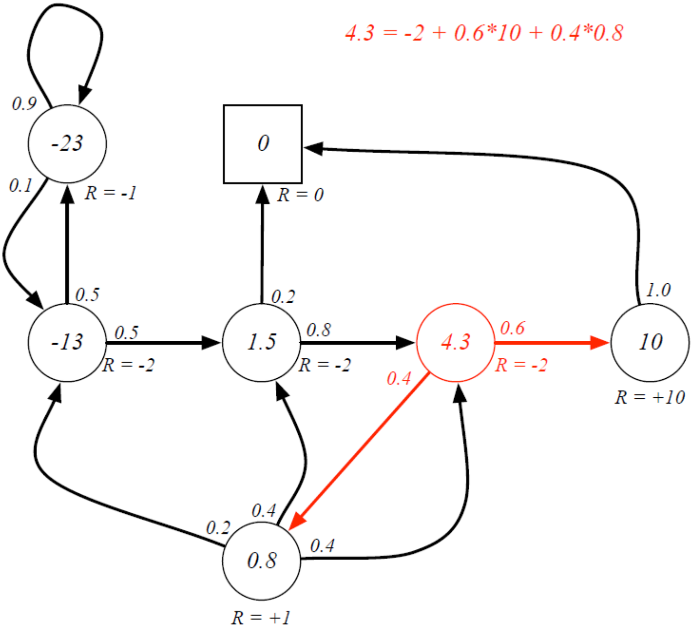
# Example: Student MRP



# Bellman Equation

$$
\begin{aligned}
V(s) &= \mathbb{E}(G_t | S_t = s) \\
&= \mathbb{E}(R_{t+1} + \gamma R_{t+2} + \ldots | S_t = s) \\
&= r(s) + \gamma \sum_{s'} p(s, s') \mathbb{E}(R_{t+2} + \gamma R_{t+3} + \ldots | S_{t+1} = s') \\
&= r(s) + \gamma \sum_{s'} p(s, s') V(s')
\end{aligned}
$$

# Example: Student MRP

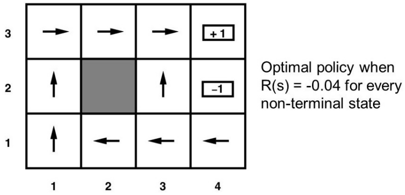


# Bellman Equation in Matrix Form

$$\begin{pmatrix} V(1) \\ \vdots \\ V(n) \end{pmatrix} = \begin{pmatrix} r(1) \\ \vdots \\ r(n) \end{pmatrix} + \gamma \begin{pmatrix} p(1,1) & p(1,2) & \cdots & p(1,n) \\ p(2,1) & p(2,2) & \cdots & p(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ p(n,1) & p(n,2) & \cdots & p(n,n) \end{pmatrix} \begin{pmatrix} V(1) \\ \vdots \\ V(n) \end{pmatrix}$$

It can be solved directly: $V = (I - \gamma\mathcal{P})^{-1}\mathcal{R}$
Computational complexity is $O(n^3)$ for $n$ states
Many iterative methods for large MRP: dynamic programming, Monte-Carlo evaluation, Temporal-Difference learning

# Example: Grid World




Policy Matrix: Each action is represented by a number : Action (Up) is represented by 0, (Rigth) by 1, (Down) by 2 and, finally, (Left) by 3

# Example: Grid World (cont.)

Transition probabilities: Column $0$ represents direction Up, Column $1$ represents direction Right, Column $2$ represents direction Down and Column $3$ represents direction Left.

$$\begin{pmatrix} 0.8 & 0.1 & 0 & 0.1 \\ 0.1 & 0.8 & 0.1 & 0 \\ 0 & 0.1 & 0.8 & 0.1 \\ 0.1 & 0 & 0.1 & 0.8 \end{pmatrix}$$

**Exercise: For $\gamma = 0.999$, calculate the value function for all the states**

# Markov Decision Process (MDP)

An MDP is a Markov Reward process with decisions.

**Definition**

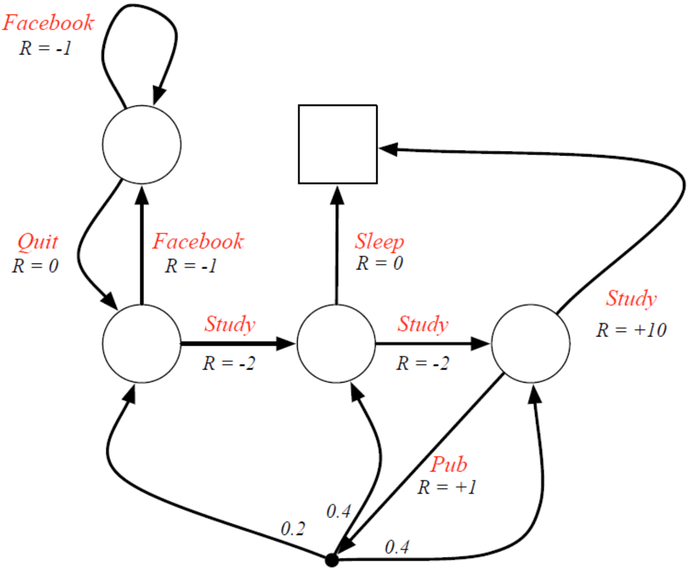
Markov Reward Process is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix

$$p(s'|s,a) = \mathbb{P}(S_{t+1} = s' | S_t = s, \mathcal{A}_t = a)$$

- $\mathcal{R}$ is a reward function $r(s,a) = \mathbb{E}(R_{t+1} | S_t = s, \mathcal{A}_t = a)$
- $\gamma \in [0,1]$ is a discount factor

# Example: Student MDP



# Policies (1)

**Definition:**

A policy $\pi$ is a distribution over actions: :

$$\pi(s) = \mathbb{P}(A_t | S_t = s)$$

- A policy fully defines the behavior of an agent
- Policies are stationary or time-independent

# Policies (2)

- Given an MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and a policy $\pi$
- The state sequence $S_1, S_2, \ldots$ is a Markov Process $\langle \mathcal{S}, \mathcal{P}^{\pi} \rangle$
- The state and reward sequence $S_1, R_2, S_2, R_3 \ldots$ is a Markov reward process $\langle \mathcal{S}, \mathcal{P}^{\pi}, \mathcal{R}^{\pi}, \gamma \rangle$

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

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

# 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 $\pi$

$$V_\pi(s) = \mathbb{E}(G_t | S_t = s, A_{t:\infty} \sim \pi)$$

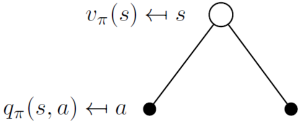
**Definition:**

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

$$q_\pi(s, a) = \mathbb{E}\left(G_t | S_t = s, A_t = a, A_{t+1:\infty} \sim \pi\right)$$

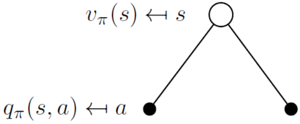
# Bellman Expectation Equation

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

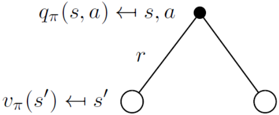


# Bellman Expectation Equation

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

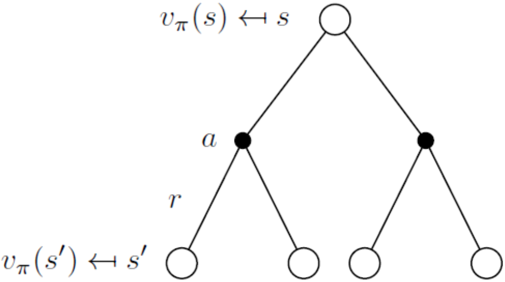



$$q_\pi(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V_\pi(s')$$



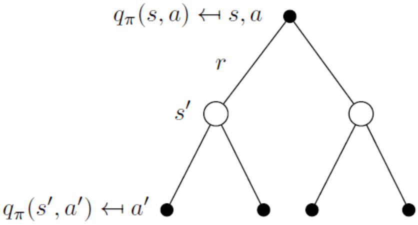
# Bellman Expectation Equation for $V_\pi$

$$V_\pi(s) = \sum_a \pi(a|s)\left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V_\pi(s')\right)$$

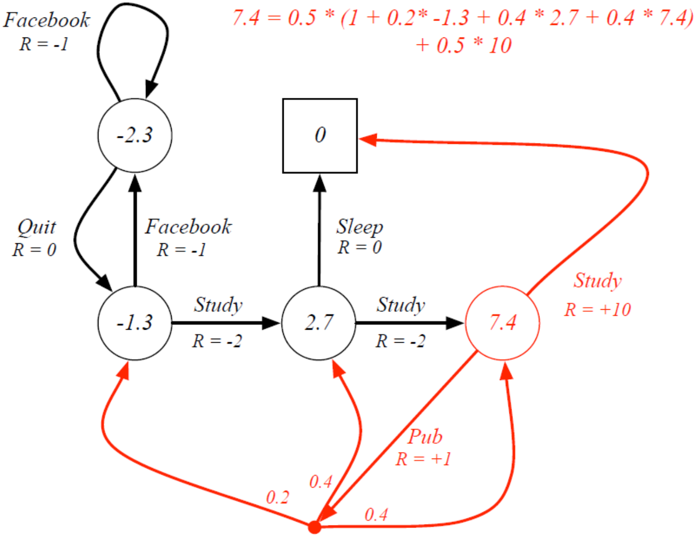


# Bellman Expectation Equation for $q_\pi$

$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')$$



# Bellman Expectation Equation in Student MDP



# Optimal Value Function

**Definition:**

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

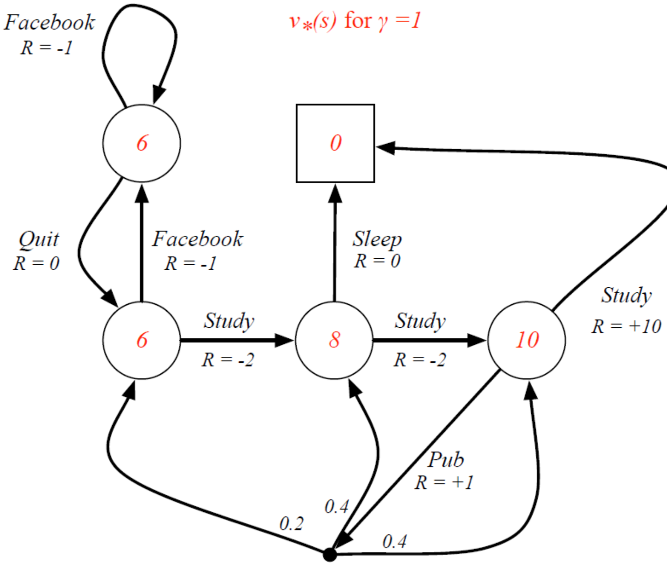
$$V_*(s) = \max_{\pi} V_\pi(s)$$

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

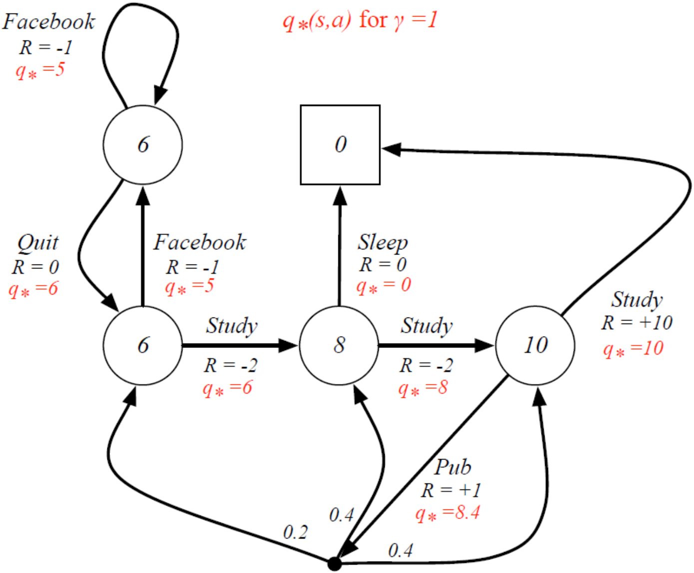
$$q_*(s, a) = \max_{\pi} q_\pi(s, a)$$

An MDP is solved if we find either $V_*(s)$ or $q_*(s, a)$

# Example: Optimal value function for Student MDP



# Example: Optimal action-value function for Student MDP



# Optimal Policy

Define a partial ordering over policies

$$\pi \geq \pi' \quad \textit{if} \quad V_\pi(s) \geq V_{\pi'}(s), \forall s$$

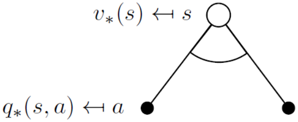
**Theorem:**

For any Markov Decision Processes

- There exists an optimal policy $\pi_*$ that is better or equal to all others, i.e., $\pi_* \geq \pi, \forall \pi$
- All optimal policies achieve the optimal value function $V_{\pi_*}(s) = V_*(s)$
- All optimal policies achieve the optimal action-value function $q_{\pi_*}(s, a) = q_*(s, a)$

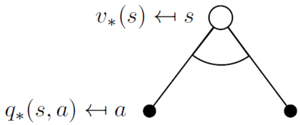
# Bellman Optimality Equation

$$V_*(s) = \max_{a \in \mathcal{A}} q_*(s, a)$$

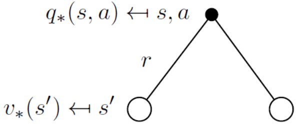


# Bellman Optimality Equation

$$V_*(s) = \max_{a \in \mathcal{A}} q_*(s, a)$$

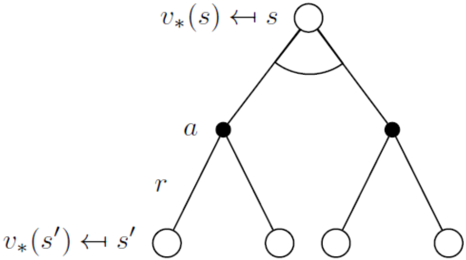



$$q_*(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V_*(s')$$



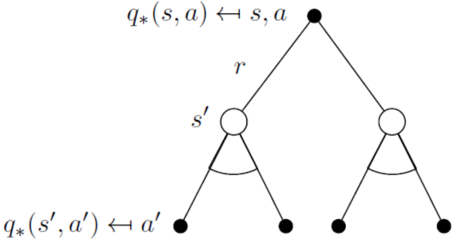
# Bellman Optimality Equation for $V_*$

$$V_*(s) = \max_{a \in \mathcal{A}} \left( r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V_*(s') \right)$$

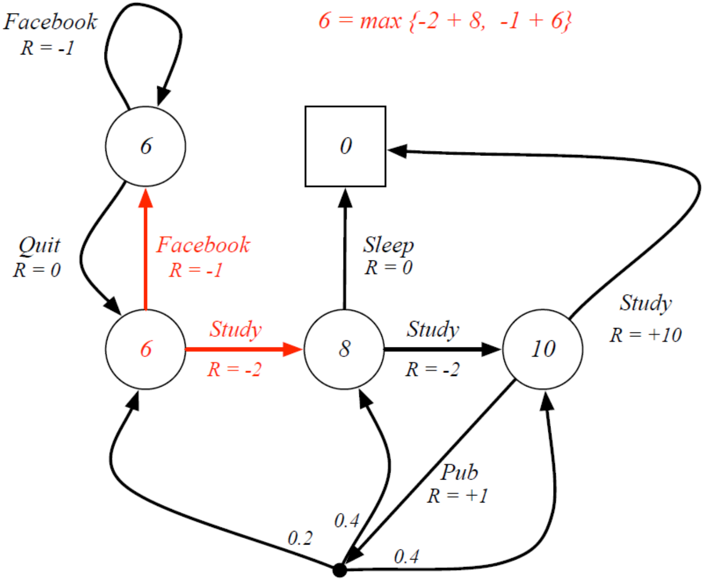


# Bellman Optimality Equation for $q_*$

$$q_*(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \max_{a' \in \mathcal{A}} q_*(s',a')$$



# Bellman Optimality Equation in Student MDP

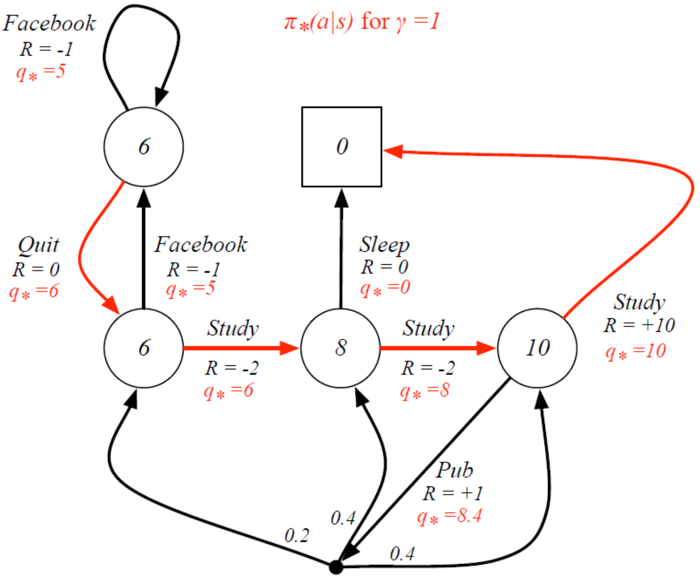


# Finding an Optimal policy

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

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

# Bellman Optimality Equation in Student MDP



# Finite Markov Decision Process (MDP)

$$
\begin{aligned}
V_T^{\pi}(i) &= \mathbb{E}\left(R_1 + R_2 + \cdots + R_T | S_0 = i\right) \\
&= \mathbb{E}_{\pi}(\sum_{t=1}^{T} R_t | S_0 = i)
\end{aligned}
$$

$$
V_t(i) = sup_{\pi} V_t^{\pi}(i)
$$

# Finite Markov Decision Process (MDP)

Imagine action $a$, which yields reward $r(i, a)$. The best we can do from now on is

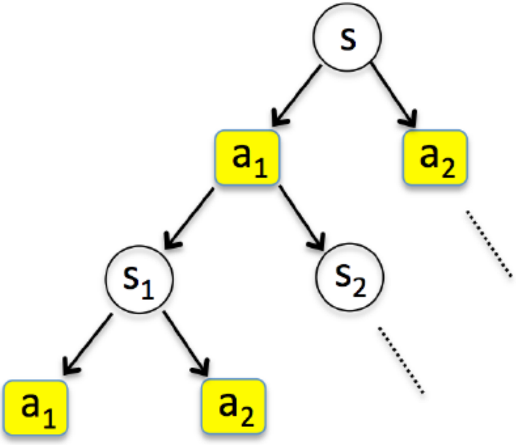
$$r(i, a) + \sum_j p(j|i, a) V^{T-1}(j)$$

Then, the best action is

$$V^T(i) = \max_a \left( r(i, a) + \sum_j p(j|i, a) V^{T-1}(j) \right)$$

Known as Optimality Equation, Dynamic Programming, Howard Equation,...

# Bellman's key idea




Decision tree with depth $T$: $A^T S^{T+1}$ leaves (optimizing over history dependent policies)
Dynamic Programming: $S^2 AT$ operations

# Richard Bellman

1920 - 1984
American applied mathematician

Introduced **Dynamic Programming** (DP) as a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions.

# Extensions to MDPs

- Infinite and continuous MDPs
- Partially observable MDPs
- Undiscounted, average reward MDPs

# Homework Assignment $\#1$

Due date $23/04$. To be done individually or by groups of $2$.

- Using the equation in $31/61$, derive the values for the value function of slides $26/61$, $27/61$ and $28/61$. Explain the values obtained as $\gamma$ changes.
- Verify that the values of the value function in slide $42/61$ satisfy the Bellman Expectation Equation of $40/61$ (as done in the example in red)
- Derive the values of the optimal action-value function $q_*$ of $45/61$ (Hint: To calculate $q(s, a)$ you can use the equation in $39/61$, and then calculate $q_*$ as in the definition)
- Verify and explain how the values in slide $50/61$ satisfy the Bellman optimality equation.

# Exercises

**Exercise**

*A miner is at the bottom of a mine and sees three tunnels: 1, 2 and 3. Tunnel 1 leads to the exit in a hour. Tunnel 2 returns to the same crossroads in 2 hours and tunnel 3 returns to the crossroads in 3 hours. Every time the miner is at the crossroads, he chooses one of the tunnels with probability* $1/3$*, regardless of what he chose before. Define a Markov reward process describing this situation. Let* $T$ *be the time it takes to leave the mine. Compute* $E(T)$*.*

What would be the Bellman Optimality Equation ?

# Exercise: Finite horizon MDP

Revenue management: Littlewood's model

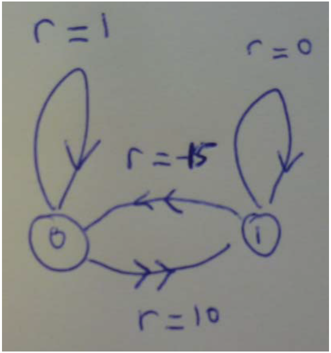
**Problem** An airplane has 20 seats available, and the sell closes in 50 days. At every time epoch, the airplane decides the selling price: Either $p_1 = 5$, and then it will sell a seat with probability $q_1 = 0.1$, or $p_2 = 1$, and then it will sell a seat with probability $q_2 = 0.8$.

- Model the problem as a Finite Horizon MDP with total reward criterion, and write the optimality equation
- What is the optimal selling strategy? which qualitative conclusion you can draw from the solution?

**Help:** Let $s$ denote the remaining seats available, and $V_T(s)$ denote the total reward in state $s$ and with $T$ days left. Use the principle of optimality of slide 54/61.

# Exercise: Infinite Horizon MDP

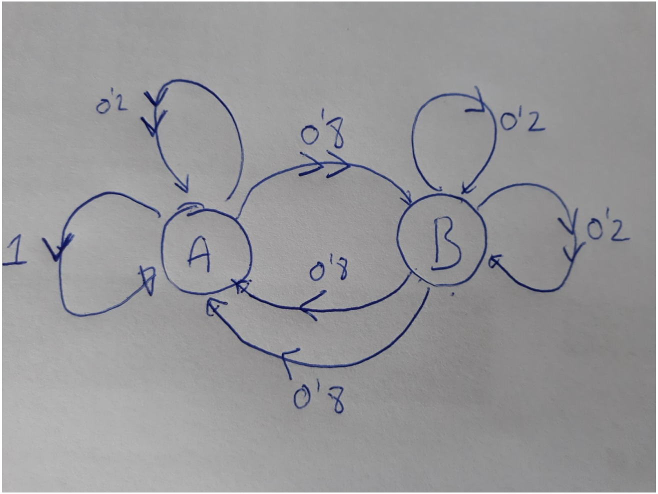
Note: The reward from state $1$ to $0$ under action $<<$ is $-15$




What is the optimal policy (for total discounted reward) for various values of $\gamma$?

- Write the optimality equation
- Solve it analytically as a function of $\gamma$

# Exercise: "You're the reinforcement learner"




$r(A, 1) = 10; r(A, 2) = -10; r(B, 1) = +40, r(B, 2) = +20$

- Write the optimality equation