CS395T: Foundations of Machine Learning for Systems Researchers

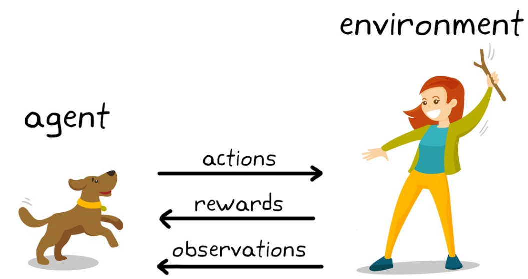
Fall 2025

Lecture 6:

Reinforcement Learning & Markov Decision Processes (MDPs)

Reinforcement Learning

Standard presentations: agent takes actions and is rewarded by environment; like training a dog



Our view:

Optimal control: sequence of actions *under uncertainty* to reach a goal while optimizing some objective function

Example: moon shot vs. gun shot

- Goal: reach the moon
- Burn minimal amount of fuel
- Uncertainty: imperfect modeling of rocket, gravitation, etc.
- Solution: series of mid-course corrections

Reinforcement learning:

- Optimal control **when you have little to no knowledge of the dynamics of system** (not even Newton's Laws or gravitation!)
- However, you are allowed to perform many experiments to build a working model of the dynamics

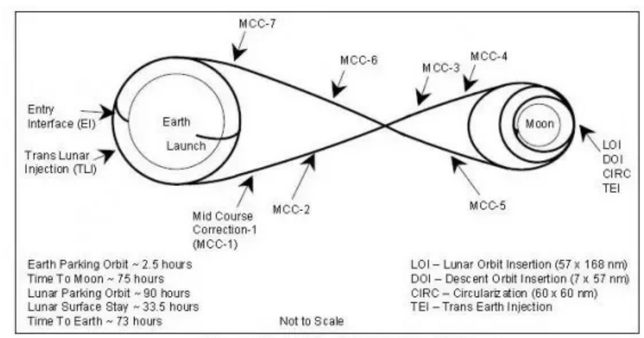
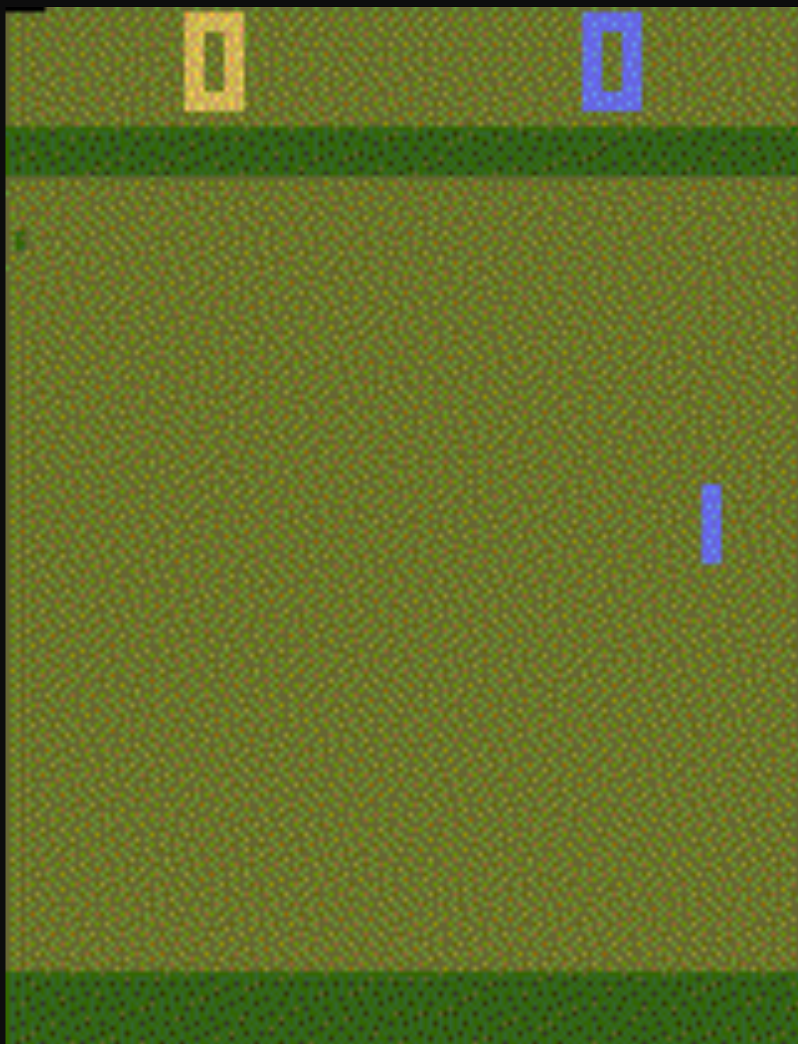


Figure 4. Nominal Apollo 13 mission profile.[12]

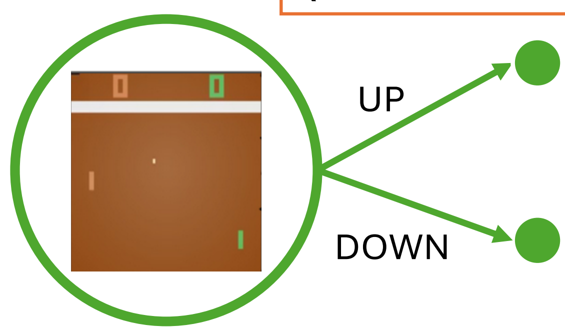


Atari Pong

Traditional program to play Pong

Set of rules (state → action)

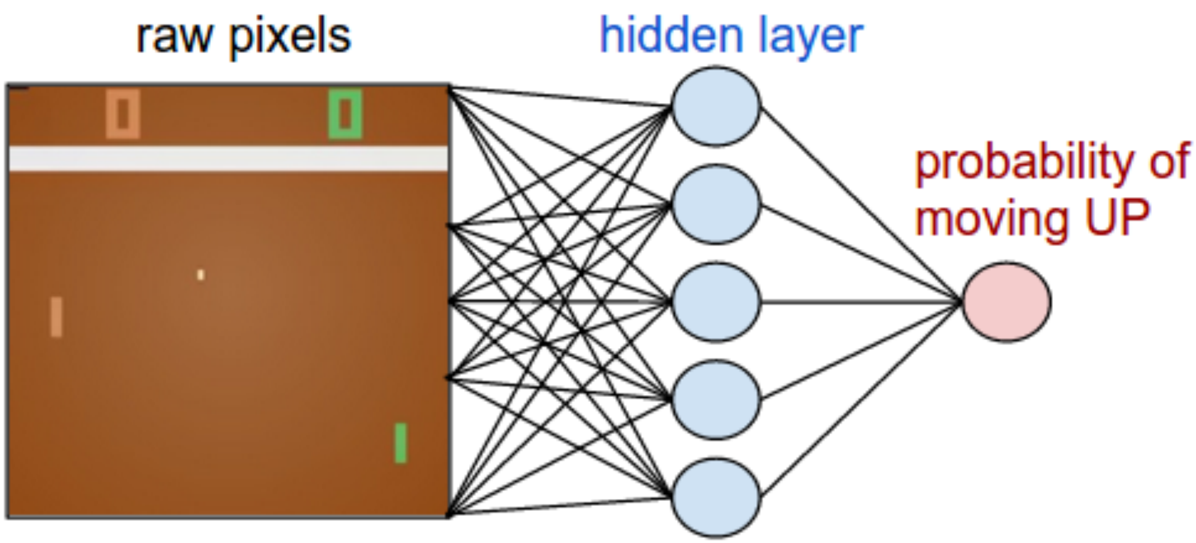
Possible state features:
- Position of paddle A
- Position of paddle B
- Position of ball
- Direction of ball
- ...



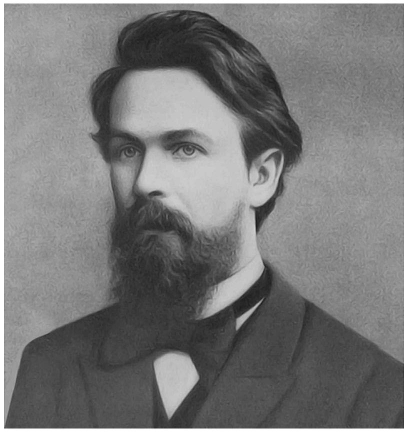
State transition system
Need to design set of state features
Designing action rules requires knowledge of collision dynamics

Solving Pong with NNs

Pong from Pixels (Andrej Karpathy) – Pong in <150 LOC



Organization



Andrei Markov (1856-1922)

Background: Discrete-time Markov process

Markov *Decision* Processes (MDP):

- Adds *actions and rewards* to Markov process model
- Key abstraction for RL

Time horizon

- Episodic task: agent stops after H time steps
- Continuous tasks: no termination

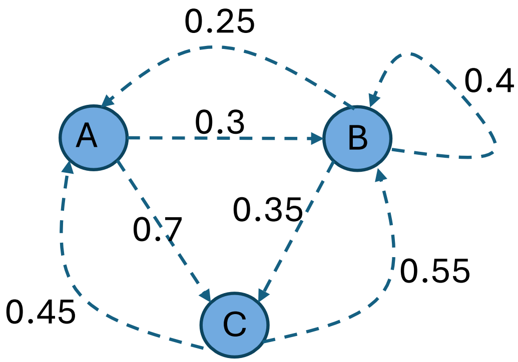
Policy and policy optimization

- Policy tells agent what action to take at each step
- Policy optimization problem: what policy maximizes expected reward?

Solving policy optimization

- Value iteration: dynamic programming, Bellman iteration
- Policy iteration: search over space of policies
- Easy to understand if we use *MDP transition diagrams*

Discrete-time Markov Process

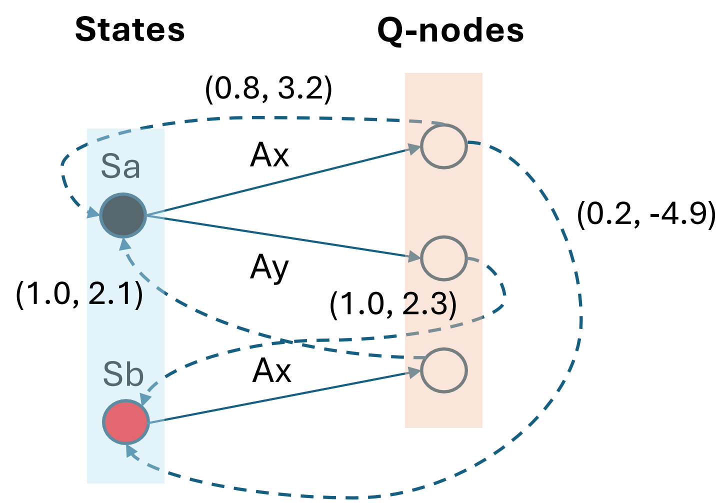


Transition matrix T = $\begin{array}{cc} \begin{array}{ccc} \text{A} & \text{B} & \text{C} \end{array} & \\ \begin{pmatrix} 0.0 & 0.3 & 0.7 \\ 0.25 & 0.4 & 0.35 \\ 0.45 & 0.55 & 0.0 \end{pmatrix} & \begin{array}{c} \text{A} \\ \text{B} \\ \text{C} \end{array} \end{array}$

- Set of agent states: {A,B,C}
- Discrete-time
- Agent starts in some state and makes transitions probabilistically between states at each time step
  - Markovian: transition probabilities depend only on current state
- Transition matrix T(i,j) = probability of transition from state i to state j
- T^k: transition probabilities after k steps

Markov *Decision* Process (MDP)

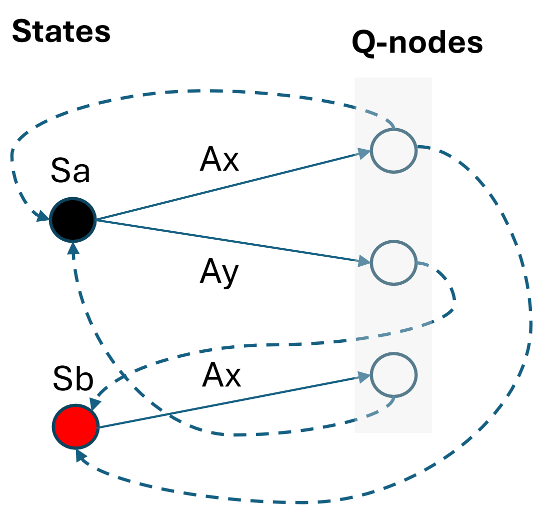
- Adds **actions** and **rewards** to Markov processes
- In each state, agent performs action, and transitions to another state probabilistically
- Probabilistic transitions model effect of environment on outcome of action (e.g. wind)
- Reward for transitions
- **Policy π**: what action to take at each step
- **Optimal policy** maximizes expected reward for task
  - **Policy optimization:** find optimal policy π^*



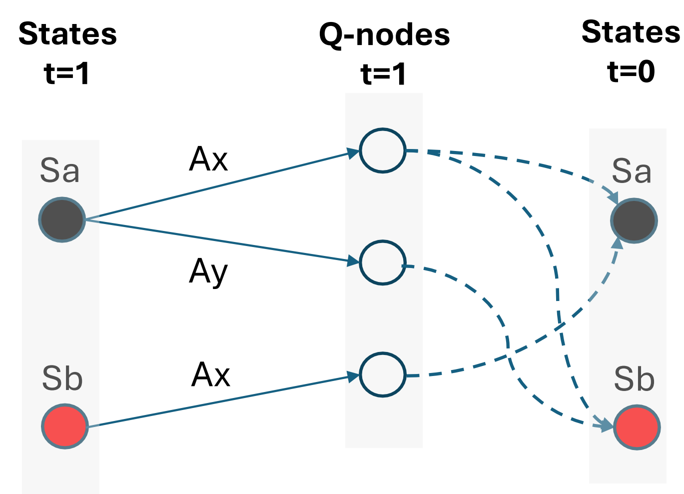
States S = {Sa, Sb}
Actions A = {Ax, Ay}
Q-nodes: S x A (<state, action> pairs, used for book-keeping)
Transition edges: dotted edges (labeled by probability and reward)

MDP *Transition Diagram*

Markov Decision Process



MDP Transition Diagram (one-step)



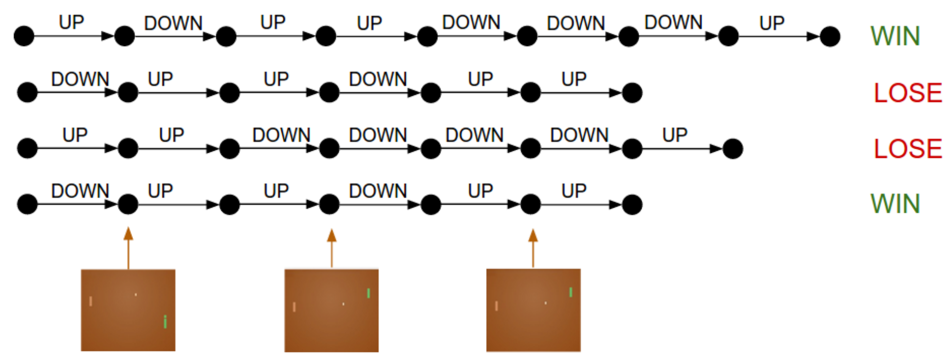
Game plan for policy optimization

Tasks:

Episodic tasks: $H = 1$

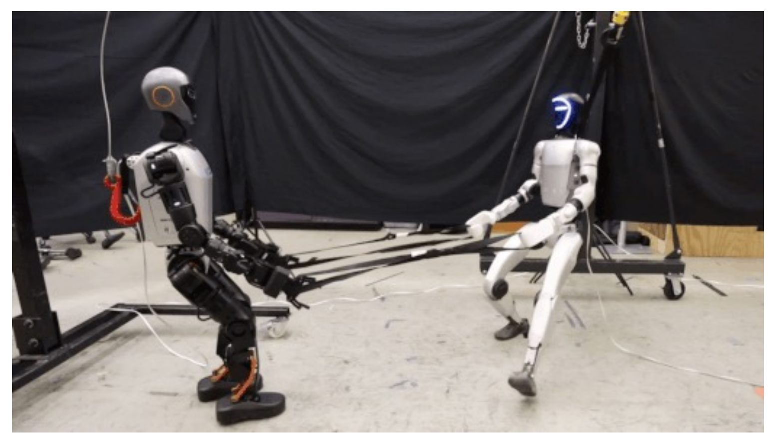
Episodic tasks: $\infty > H > 1$

Continuous tasks: $H = \infty$

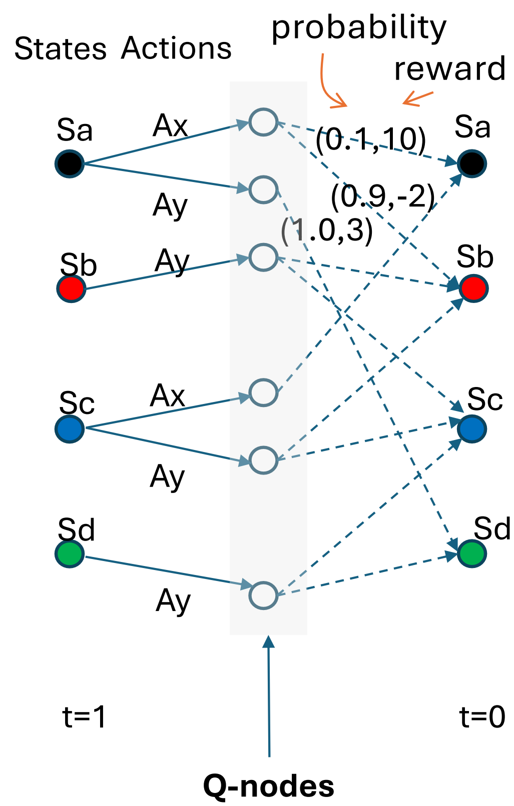


Algorithms:

Value iteration
Policy iteration

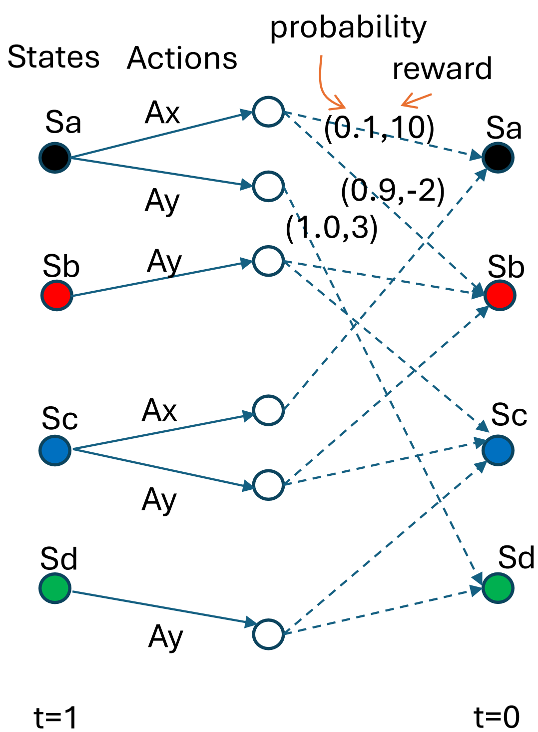


MDP Example: H =1



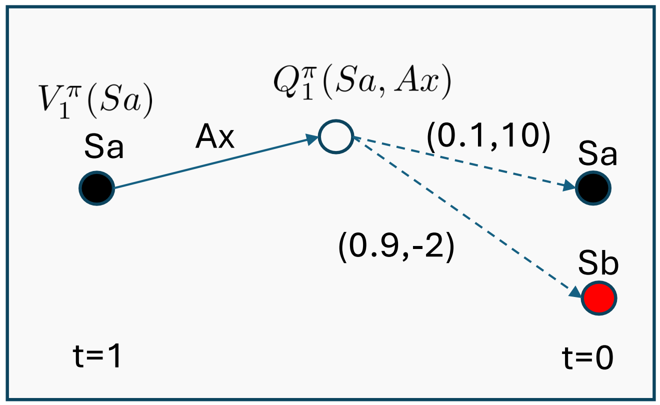
Policy π: S $\rightarrow$A

Policy optimization by searching policy space



- Example: focus on Sa (other states similar)
  - Policy π_1: at Sa, take Action Ax
    - Expected reward = (0.1*10+0.9*(-2)) = -0.8
  - Policy π_2: at Sa, take Action Ay
    - Expected reward = 1.0*3 = 3
  - Policy π_2 is optimal
- In general
  - **Policy iteration**
    - Finite number of policies: $|A|^{|S|}$
    - Search space of policies
  - **Policy evaluation**: how good is a given policy?
    - Compute expected reward

Terminology



Policy

State

Time-step

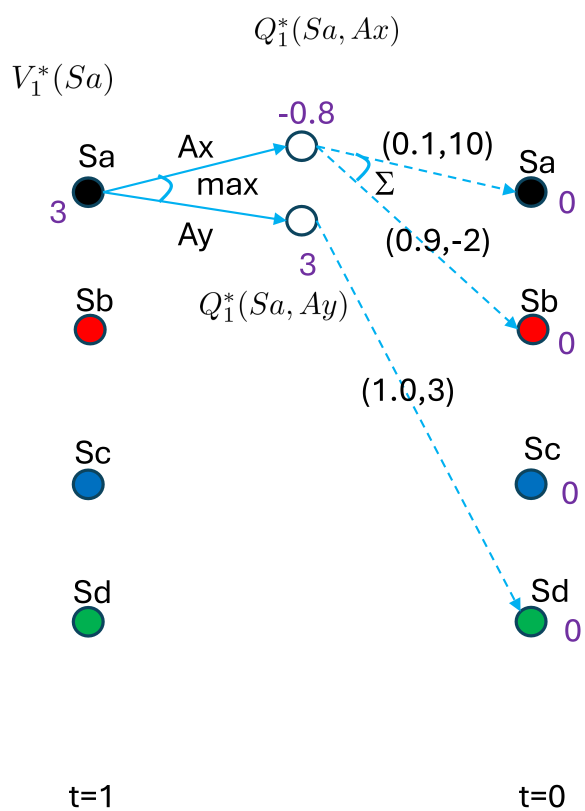
We are computing a *valuation* V_1^{π}(Sa)

- Expected reward at Sa at time 1 under policy π

V_1^{*}(Sa): Valuation under optimal policy (= 3.0 for our problem)

Notation extends to Q-values: Q_1^{π}(Sa,Ax)

Another solution: Value iteration (Expectimax)



Bottom-up propagation of values

- Boundary (t=0) node values = 0
- Transfer functions on (state x action → state) edges
  - Add expectation to boundary value
- Confluence operator at Q-nodes: sum
- Confluence operator at states: max

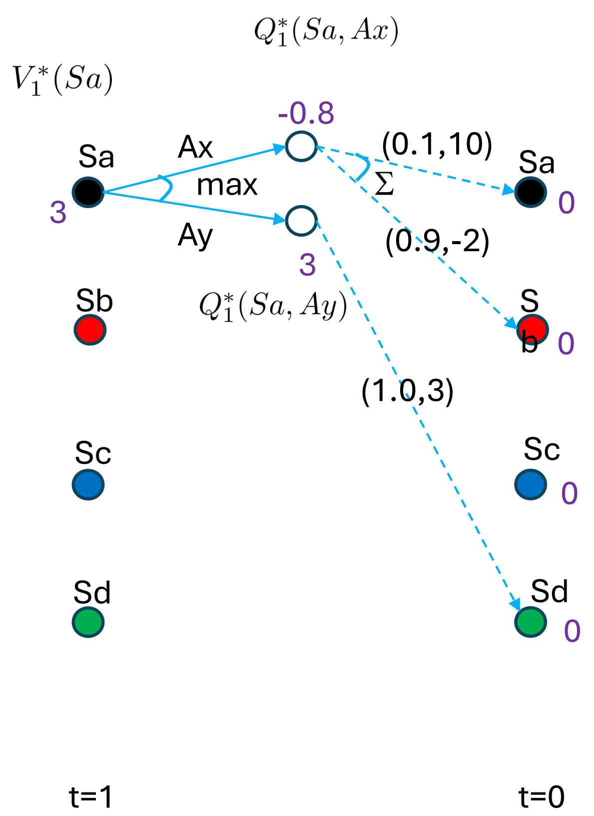
If terminal states have rewards

- Use those values as boundary node values

Optimal policy: read off from Q values

Like minimax in game trees

Putting it all together (H=1)



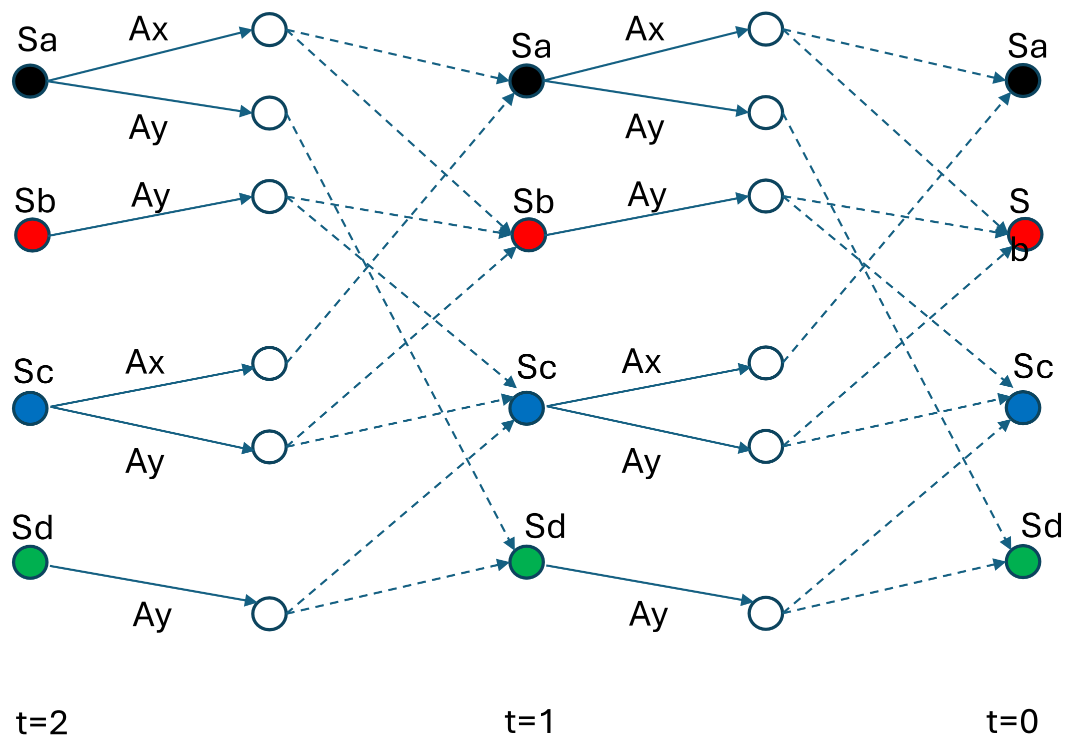
Value iteration:

$$V_0^*(s) = 0 \quad (\forall s \in S)$$
$$Q_1^*(s,a) = \Sigma_{s'} P(s,a,s') * (R(s,a,s') + V_0^*(s'))$$
$$V_1^*(s) = max_a\ Q_1^*(s,a)$$

Policy evaluation: π**: S → A**

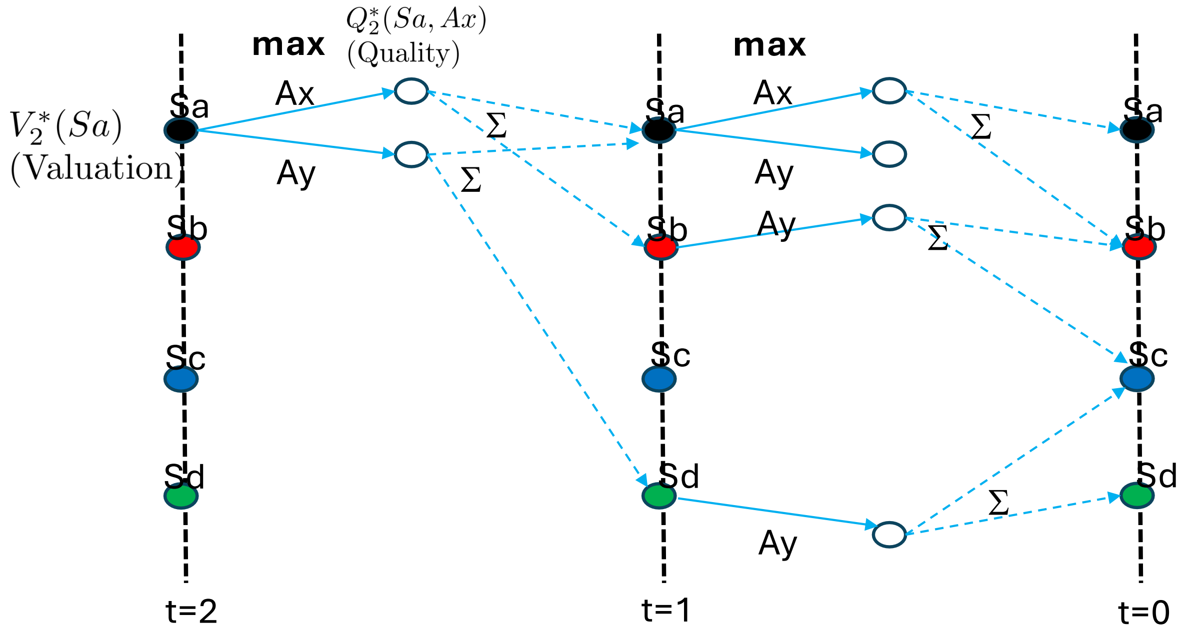
$$V_0^\pi(s) = 0 \quad (\forall s \in S)$$
$$V_1^\pi(s) = \Sigma_{s'} P(s,\pi(s),s') * (R(s,\pi(s),s') + V_0^\pi(s'))$$

Multiple time-steps: ideas carry over in obvious way



Policy: S x T → A (system is still Markovian!)

Value Iteration (H=finite)



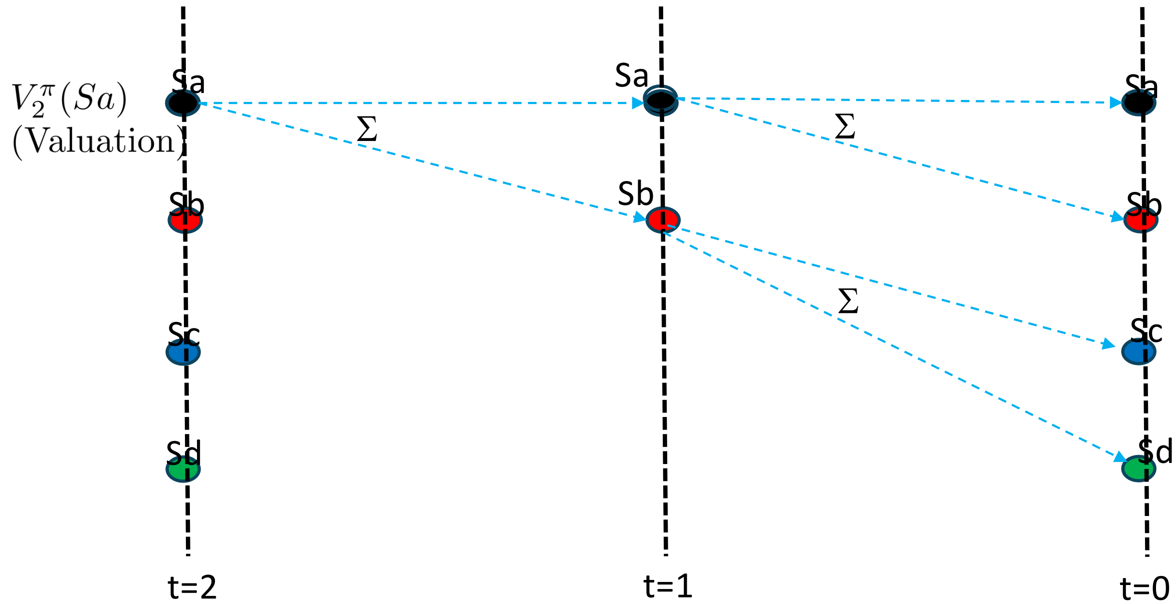
Bottom-up computation in transition diagram

$$V_0^*(s) = 0 \quad (\forall s \in S)$$
$$Q_t^*(s,a) = \Sigma_{s'} P(s,a,s') * (R(s,a,s') + V_{t-1}^*(s'))$$
$$V_t^*(s) = max_a \; Q_t^*(s,a)$$

Dynamic Programming (Bellman)

Read off optimal policy from Q*-values

Policy Iteration (H=finite)



Search over set of policies: $|A|^{|S| x H}$

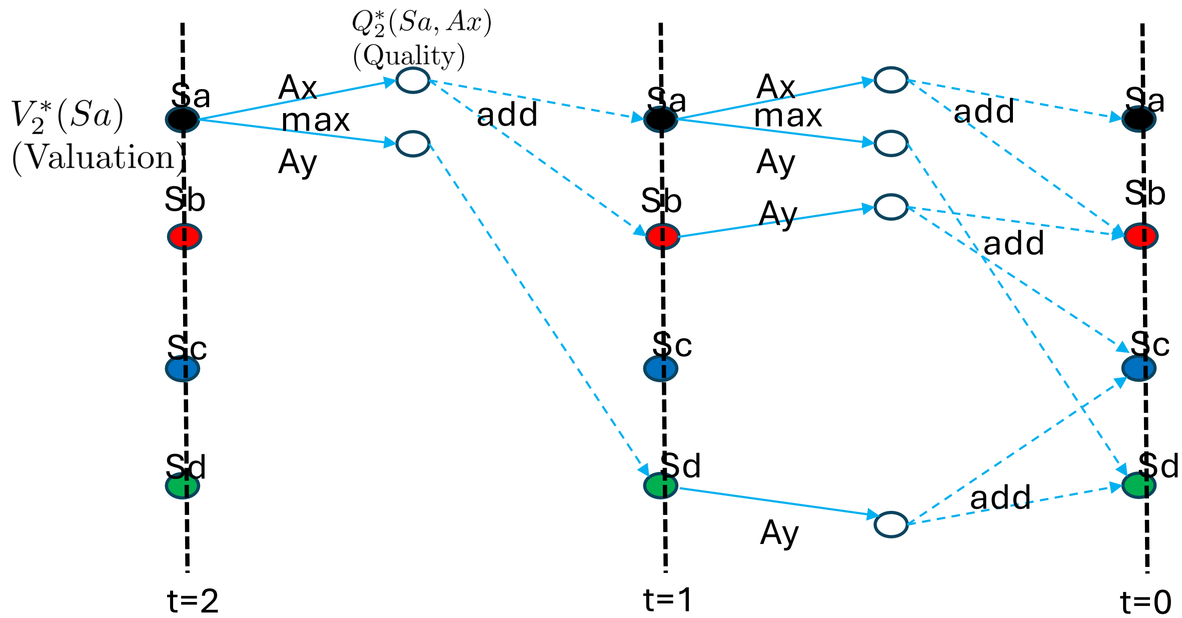
Policy evaluation: π: SxT→A

- Specialization of value iteration for action π(s,t)

$$V_0^{\pi}(s) = 0 \quad (\forall s \in S)$$

$$V_t^{\pi}(s) = \Sigma_{s'} P(s, \pi(s,t), s') * (R(s, \pi(s,t), s') + V_{t-1}^{\pi}(s'))$$

Discount



Future rewards are discounted by a factor of $0 \leq \gamma < 1$ for each time step

Value iteration:

$$V_0^*(s) = 0 \quad (\forall s \in S)$$
$$Q_t^*(s,a) = \Sigma_{s'} P(s,a,s') * (R(s,a,s') + \gamma * V_{t-1}^*(s'))$$
$$V_t^*(s) = max_a(Q_t^*(s,a))$$

Continuous Tasks

Infinite time horizon: agent performs actions indefinitely ($H \to \infty$)

Policy π: S → A

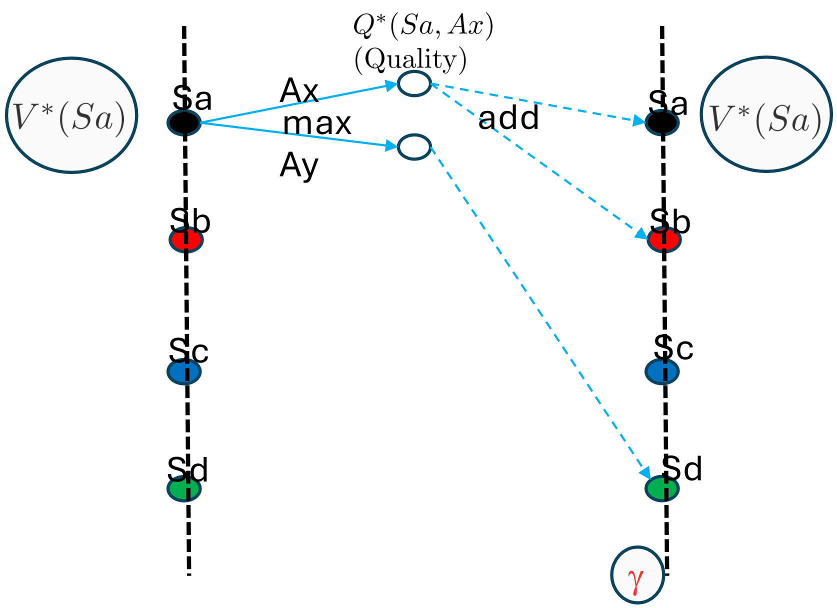
- Action at state is not time-dependent since time is unbounded and system is Markovian
  - Compare with episodic tasks: π: SxT → A
- Finite number of policies: $|A|^{|S|}$

Discount: $0 \leq \gamma < 1$

Intuition for valuations:

- Unbounded number of unbounded paths from each state
- Each (unbounded) path has a finite total discounted reward
  - Assume there is a maximum one-step reward (R_{max})
  - Discount gives you an upper bound on total discounted reward of path ($\frac{R_{max}}{1-\gamma}$)

Value Iteration (H = ∞)



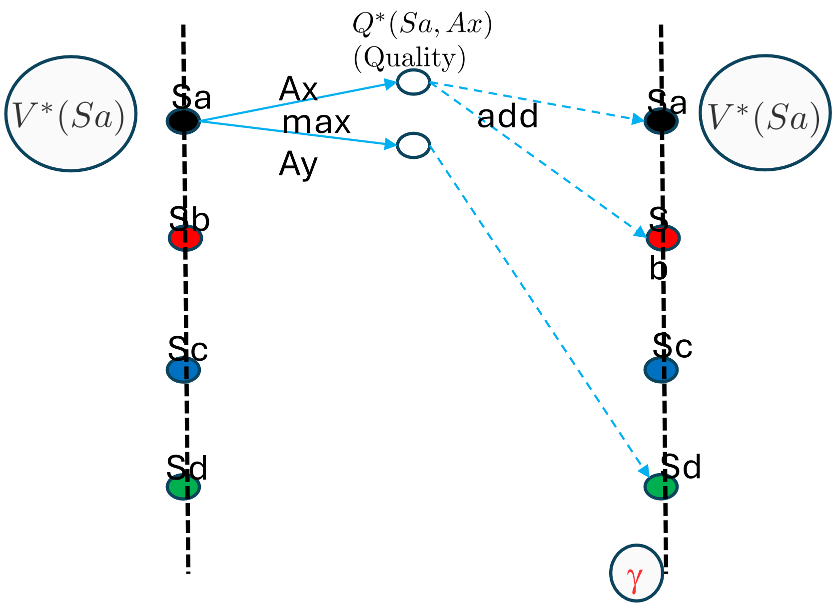
- Fixpoint equation:

$$V^*(s) = \max_a \sum_{s'} P(s,a,s')\Big[R(s,a,s') + \gamma V^*(s')\Big]$$

- Does it have a solution? Unique solution? If so, how do we compute it?
  - Use Banach-Cacciopoli fixpoint theorem

$$\underbrace{V^*(s) = \max_a \sum_{s'} P(s,a,s')\Big[R(s,a,s') + \gamma V^*(s')\Big]}_{\text{Contraction operator } \underline{V} \to \underline{V} \text{ in } (\Re^n, \text{ maxnorm}) \text{ where n = \#states}}$$

Value iteration: solving fixpoint equation

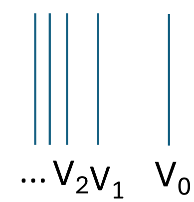


- Fixpoint equation:

$$V^*(s) = \max_a \sum_{s'} P(s,a,s')\left[R(s,a,s') + \gamma V^*(s')\right]$$

- Iterative solution: perform value iterations to convergence

$$V_0(s) = arbitrary\ value \quad (\forall s \in S)$$
$$Q_t(s,a) = \Sigma_{s'} P(s,a,s') * (R(s,a,s') + \gamma * V_{t-1}(s'))$$
$$V_t(s) = max_a(Q_t(s,a))$$



Policy Iteration (H = ∞)

Policy iteration

Policy evaluation π: S → A

- System of linear equations

$$V^{\pi}(s) = \Sigma_{s'} P(s, \pi(s), s') * (R(s, \pi(s), s') + \gamma * V^{\pi}(s'))$$

- Solve using direct methods like LU factorization or iterative methods

$$V_0^{\pi}(s) = arbitrary\ value \quad (\forall s \in S)$$

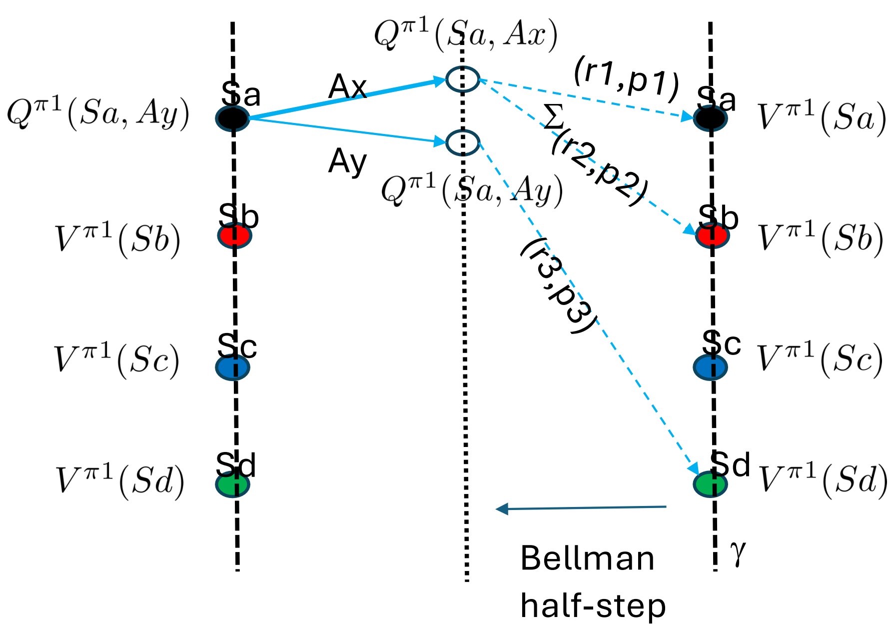
$$V_t^{\pi}(s) = \Sigma_{s'} P(s, \pi(s), s') * (R(s, \pi(s), s') + \gamma * V_{t-1}^{\pi}(s'))$$

Improving efficiency of policy iteration: can we avoid evaluating every policy in space of policies?

Policy **improvement**

- Intuition: greedy improvements to current policy
- Will find optimal policy without searching entire policy space

Policy Improvement



Given policy $\pi 1$, compute $V^{\pi 1}$ by solving linear system

Take "Bellman half-step" and compute associated $Q^{\pi 1}$ values

At each state Sa, examine $Q^{\pi 1}$ *(Sa,*)* values to see if switching to another action improves value of *Sa* "locally"

- If switching does not improve value of any state, $\pi 1$ is optimal policy
- Otherwise switch to better action at one or more states (call resulting policy $\pi 2$), compute $V^{\pi 2}$ and repeat

Guarantee: procedure terminates and finds optimal policy

- Each switch finds *strictly* better policy and set of policies is finite

Summary

H=1	
Value Iteration: $V_0^*(s) = 0 \quad (\forall s \in S)$ $Q_1^*(s,a) = \Sigma_{s'} P(s,a,s') * (R(s,a,s') + V_0^*(s'))$ $V_1^*(s) = max_a \; Q_1^*(s,a)$	**Policy Evaluation:** $V_0^\pi(s) = 0 \quad (\forall s \in S)$ $V_1^\pi(s) = \Sigma_{s'} P(s,\pi(s),s') * (R(s,\pi(s),s') + V_0^\pi(s'))$
H=finite	
Value Iteration: $V_0^*(s) = 0 \quad (\forall s \in S)$ $Q_t^*(s,a) = \Sigma_{s'} P(s,a,s') * (R(s,a,s') + V_{t-1}^*(s'))$ $V_t^*(s) = max_a \; Q_t^*(s,a)$	**Policy Evaluation:** $V_0^\pi(s) = 0 \quad (\forall s \in S)$ $V_t^\pi(s) = \Sigma_{s'} P(s,\pi(s,t),s') * (R(s,\pi(s,t),s') + V_{t-1}^\pi(s'))$
Continuous	
Value Iteration: $V^*(s) = \max_a \sum_{s'} P(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right]$	**Policy Evaluation:** $V^\pi(s) = \Sigma_{s'} P(s,\pi(s),s') * (R(s,\pi(s),s') + \gamma * V^\pi(s'))$

Other Resources

- Book on reinforcement learning by Barto and Sutton
- Github repo with well-written tutorials on RL with code and demos from Tim Miller, University of Queensland
  - https://gibberblot.github.io/rl-notes/single-agent/value-iteration.html
- David Silver's lecture series (DeepMind)
  - https://www.youtube.com/watch?v=lfHX2hHRMVQ
- Pieter Abbiel's course (Berkeley)
  - https://www.youtube.com/watch?v=2GwBez0D20A