SMU Lyle School of Engineering

Reinforcement Learning

Partial Observability

Sutton/Barto* Chapter 17.3
AIMA** Chapters 14 and 15

Michael Hahsler, SMU

*Sutton and Barto, Reinforcement Learning: An Introduction, 2nd edition, MIT Press, Cambridge, MA, 2018

**Russell and Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021

Online Material

Topics of this Course

- troduction to reinforcement learning
- **ov decision processes**
- bular Methods
  - rogramming
  - M methods
  - Temp nce learning
  - Multi-st pping
  - Planning a with tabular methods
- Part II: Approxi tion Methods
  - Prediction and Co Approximation
  - Eligibility Traces
  - Policy Gradient Method
- Part III: Modern RL Method
  - Deep Reinforcement Learning
  - Current Applications

Partial Observability

Summary of Notation

General

X	capital letters: random variables
x, p	lower-case letters: realizations of random variables or scalar functions
$\boldsymbol{w}$	Bold lower-case letters: real-valued vectors (even if random variables)
$\boldsymbol{W}$	bold capitals: matrices
α	Greek letters: parameters (vectors if in bolt)
$\Pr\{X = x\}$	probability that a random variable X takes on the value x
$X \sim p$	random variable X selected from distribution $p(x) = \Pr\{X = x\}$
$\mathbb{E}[X]$	expectation of a random variable X, i.e., $\mathbb{E}[X] = \sum_x p(x)x$
$\text{argmax}_a f(a)$	a value of action a at which $f(a)$ takes its maximal value

Value Function

G_t	return (cumulative reward) following time t
$G_{t:h}$	return from t to h (discounted and corrected)
$v_\pi(s)$	value of state s under policy π (expected return)
$v_*(s)$	value of state s under the optimal policy
$q_\pi(s, a)$	value of taking action a in state s under policy π
$q_*(s, a)$	value of taking action a in state s under the optimal policy
V, V_t	array estimates of state-value function v_π or v_*
Q, Q_t	array estimates of action-value function q_π or v_*

MDP

s, s'	states
a	an action
r	a reward
$\mathcal{S}$	set of all (nonterminal) states, $\mathcal{S}^+$ are all states
$\mathcal{A}(s)$	set of all actions available in state s
γ	discount-rate parameter
t	discrete time step
T	final time step of an episode (a.k.a. horizon)
A_t	random variable for the action at time t
S_t	random variable for the state at time t
R_t	random variable for the reward at time t
$p(s', r \mid s, a)$	probability of transition to state s' and receiving reward r, from state s taking action a.
$p(s' \mid s, a)$	probability of transition to state s' fom state s taking action a.
$r(s, a)$	expected immediate reward from state s after action a.
$r(s, a, s')$	expected immediate reward from state s to s' with action a.
$\pi(a \mid s)$	probability of taking action a in state s under stochastic policy π
$\pi(s)$	action taken in state s under deterministic policy π

Foundation

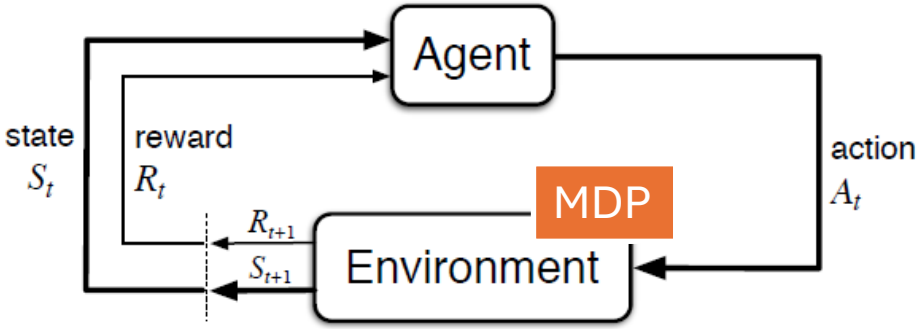
Probabilistic reasoning over time

AIMA Chapter 15

iStock
Credit: bernie_photo

Fully Observable RL

- General MDPs can have continuous state and action spaces.
- We consider finite MDPs. An MDP is **finite** if the
  - sets of states $\mathcal{S}$,
  - set of action $\mathcal{A}$, and
  - set of rewards $\mathcal{R}$ all have a finite number of elements.
  - It always has discrete time steps.



- The **dynamics** of the finite discrete-time MDP are described by the function $p: \mathcal{S} \times \mathcal{A} \times \mathcal{R} \times \mathcal{S} \rightarrow [0,1]$ defined as

$$p(s', r|s, a) \stackrel{\text{def}}{=} \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

- A decision process is an MDP if p completely characterizes the environment's dynamics. This implies:
  - Dynamics only depend on the current observed state s.
  - This means: the state must have sufficient information about the past.
  - If this is true, then it is called an **information state** and the system has the necessary **Markov property**:

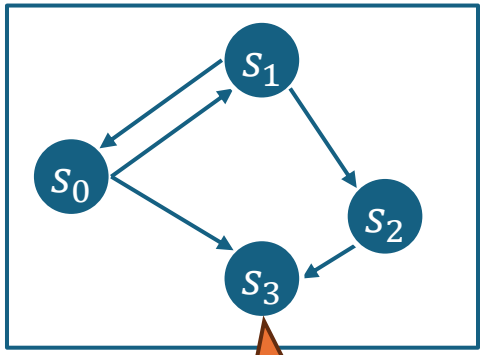
$$\Pr\{s_{t+1}|\, s_t, a_t\} = \Pr\{s_{t+1}|\, s_0, a_0, s_1, a_1, \ldots, s_t, a_t\}$$

Markov Process

In RL, we see a sequence of states. **A Markov chain or Markov process** is a stochastic process describing a sequence of possible events (states) defined by:

- A set of states: $\mathcal{S} = \{s_0, s_1, s_2, \dots, s_n\}$
- A stochastic transition model: $\Pr(s_t|s_{t-1}) \quad \forall s_t, s_{t-1} \in \mathcal{S}$
- The transition model is:
  - **Markovian**: The current state only depends on the previous state:
    $$\Pr(s_t|s_{t-1}, s_{t-2}, \dots, s_0) = \Pr(s_t|s_{t-1})$$
  - **Stationary**: Probabilities do not change over time:
    $$\Pr(s_t|s_{t-1}) = \Pr(s_{t+1}|s_t) \quad \forall t$$

Markov model as a graph



Subscript is the state ID

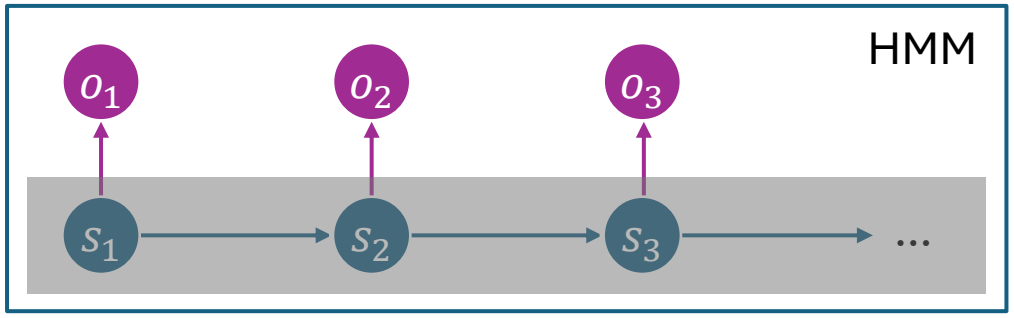
Episode is a trajectory: Markov process "unrolled" over time: $s_0, s_1, s_2, \dots$

Subscript is time t

Note: This is called a discrete-time, stationary, 1st –order Markov process with finite state space.

Hidden Markov Model (HMM)

- A statistical model used to predict a sequence of hidden events (states) by observing a sequence of visible outcomes defined by:
  - A set of hidden states: $\mathcal{S} = \{s_0, s_1, s_2, \dots, s_n\}$
  - A stochastic transition model: $\Pr(s_t|s_{t-1}) \quad \forall s_t, s_{t-1} \in \mathcal{S}$
  - A stochastic observation model: $\Pr(o_t|s_t)$
- Use: State estimation (Filtering): Find most likely state given a sequence of observations.

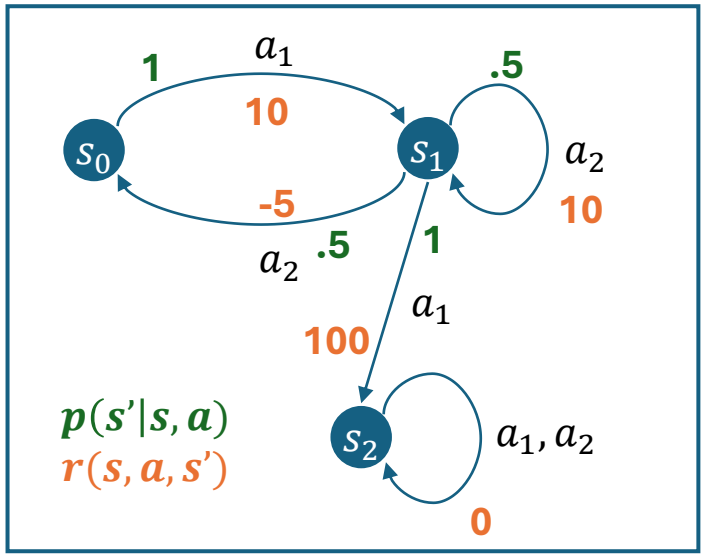


Markov Decision Process (MDP)

Finite MDPs are discrete-time stochastic control processes defined by:

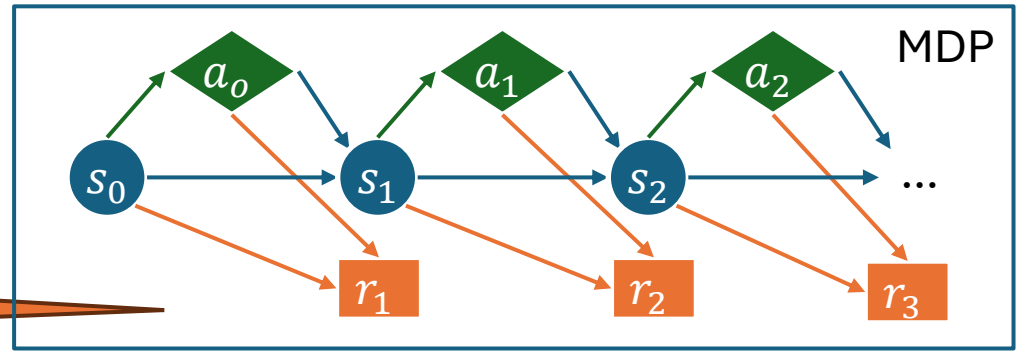
- A finite set of **states** $\mathcal{S} = \{s_0, s_1, s_2, \ldots, s_n\}$
- A set of available **actions** $\mathcal{A}(s) \;\; \forall s \in \mathcal{S}$
- A **transition model** $\Pr(s' \mid s, a)$ where $a \in \mathcal{A}(s)$
- A **reward function** r where the immediate reward depends on the current state and action (often modeled as $r(s, a, s')$)

Example MDP as a Graph



Trajectory as a Markov decision process "unrolled" over time:

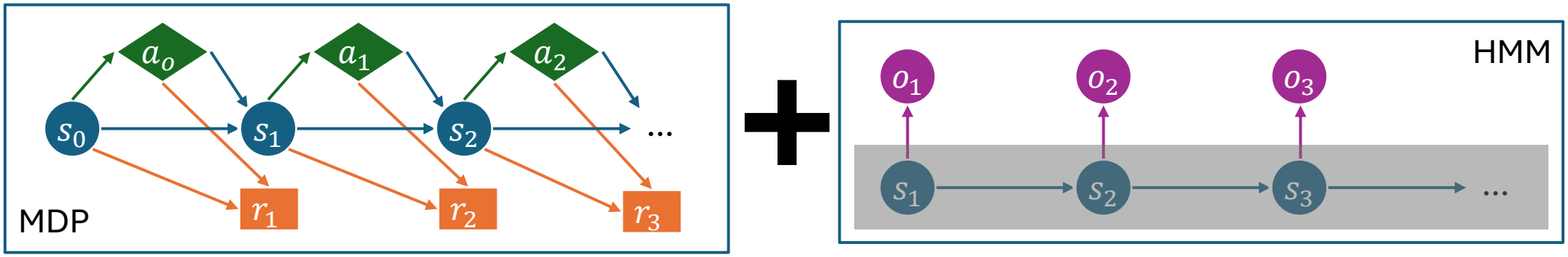
$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, \ldots$$



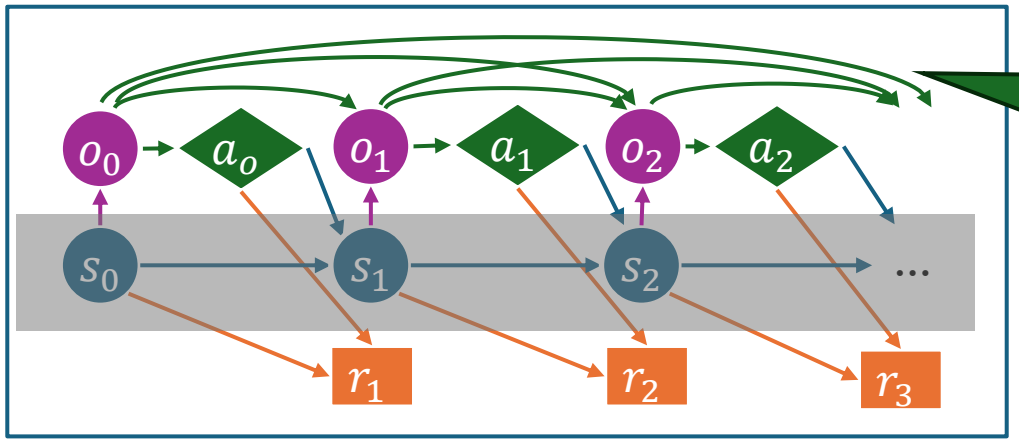
Arrow show dependencies

Partially Observable MDP (POMDP)

- Add hidden states and observations to an MDP



=



Note: Actions depend on all observations seen so far! POMDPs can be solved model-based using belief states.

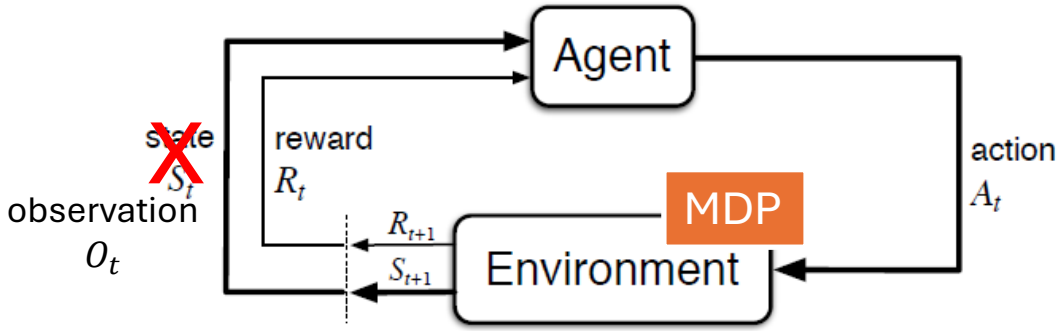
iStock
Credit: RomoloTavani

RL with Partial Observability

- Definition:
  - States: $s \in S$
  - Observations: $o \in \mathcal{O}$
  - Actions: $a \in \mathcal{A}$
  - Rewards: $r \in \mathbb{R}$
  - Transition model: $\Pr(s_t|s_{t-1}, a_{t-1})$
  - Observation model: $\Pr(o_t|a_{t-1}, s_t)$
  - Reward model: $\Pr(r_t|s_t, a_t)$
  - Discount factor: $0 \leq \gamma \leq 1$
  - Horizon: $T \in \mathbb{N}$ or ∞

Unobservable

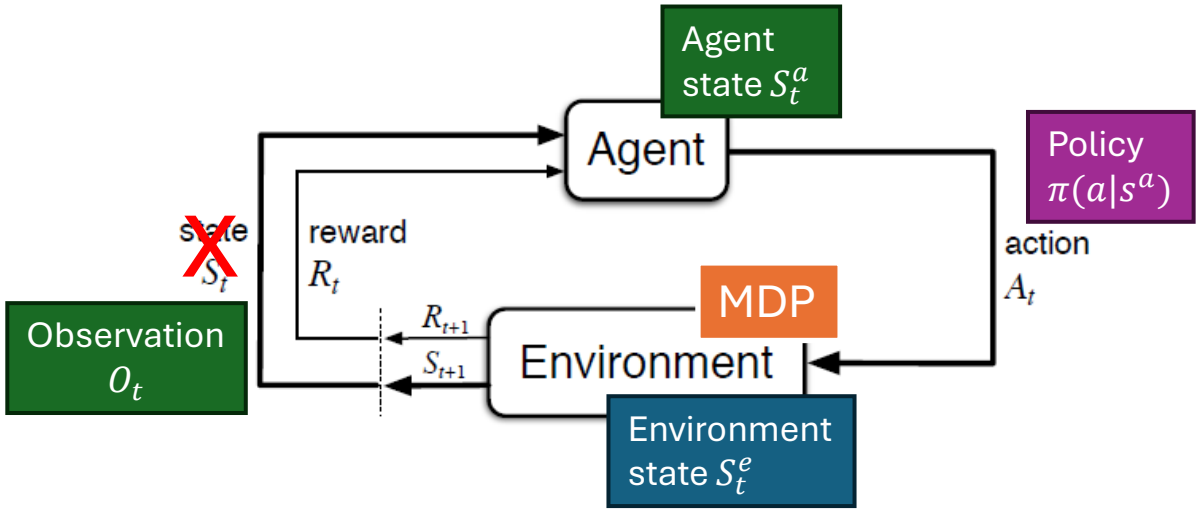
Unknown model



- Goal: find optimal policy

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{t=0}^{T} \gamma^t \mathbb{E}_\pi[r_t]$$

States and Observations

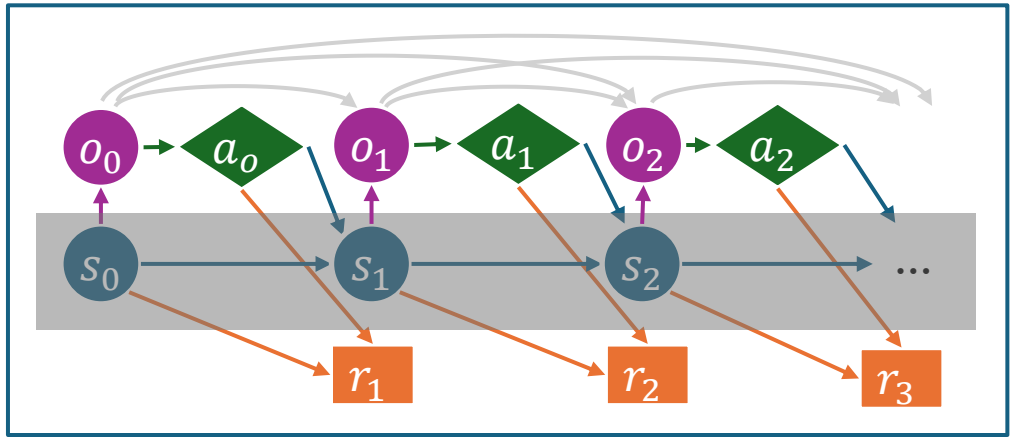


- **Environment state s_t^e**: Contains all information to determine the next state and needed to make the optimal decision for the next action.
- **Observation o_t**: Information the agent receives. Depends on the environment state via $\Pr(o_t|s_t)$.
- **Agent state s_t^a**: Internal representation that the agent uses to choose actions. Typically, $s_t^a = g(o_1, o_2, \ldots, o_{t-1})$

We do not have access to s_t^e in real applications!

How should we build s_t^a so it helps with choosing the next action?

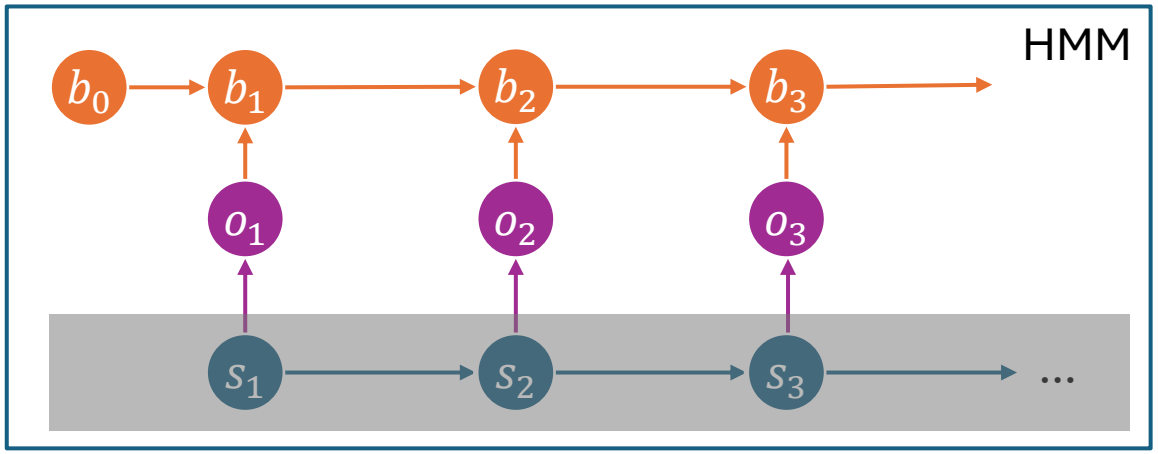
Simple Heuristic: Observation as State



- **Idea**: Use $s_t^e = o_t$ with any RL algorithm.
- **For (DRL) approximation**: Learn a state-action value function $Q(o, a|\boldsymbol{\theta})$

- **Assumption**: o_t contains enough information to decide on action a_t. This is in general not very likely; we may get a very bad approximation $Q(o, a|\boldsymbol{\theta}) \neq Q(s, a|\boldsymbol{\theta})$
- **Improvement**: Using a sliding window $s_t^e = (o_t, o_{t-1}, \ldots, o_{t-k})$ called frame stacking often works better.

Optimal Belief Monitoring



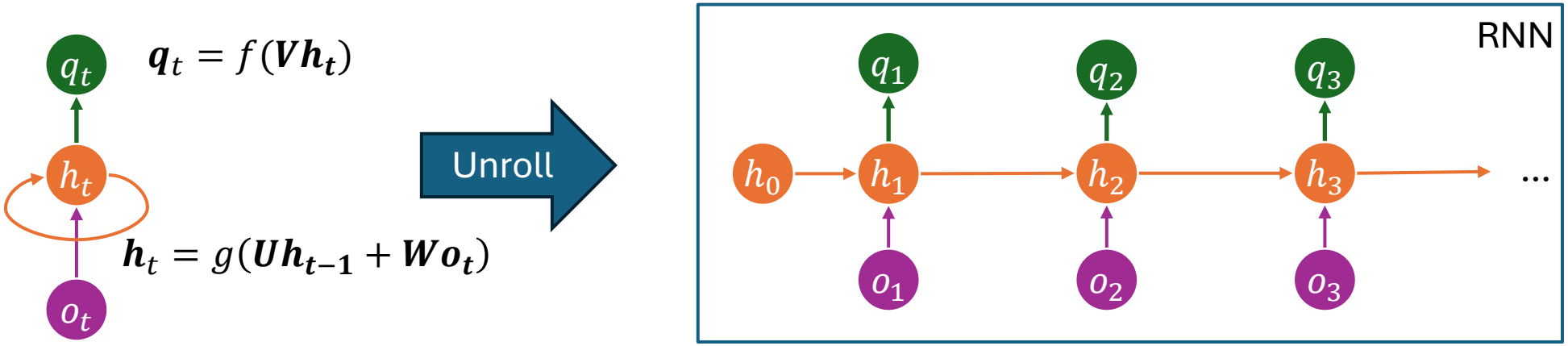
- A sufficient statistic to make a decision is a belief state $\boldsymbol{b_t}$, a probability distribution over all states $s \in \mathcal{S}$ with elements $b_t(s) = \Pr(S_t = s|o_1, o_2, \ldots o_t)$. We need:
  - Initial state distribution: $\boldsymbol{b_0} = \Pr(S_0)$
  - Transition probabilities: $\Pr(s_t|s_{t-1})$
  - Observation probabilities: $\Pr(o_t|s_t)$
- A HMM is a special type of a **Dynamic Bayesian Network (DBN)** and belief states can be updated with observations o_t using Bayesian updates (prediction and filtering):

$$b_t(s_t) \propto \Pr(o_t|s_t) \sum_{s_{t-1} \in \mathcal{S}} \Pr(s_t|s_{t-1})\, b_{t-1}(s_{t-1})$$

- **Issue for RL**: Bayesian belief monitoring requires access to the transition and observation probabilities.

Deep Belief Monitoring with RNNs

Recurrent Neural Networks (RNNs)



The hidden state keeps track of the agent state $s_t^a = h_t$ and is used to predict Q-values used for action selection. The network can be trained using 1-step TD methods like DQN.

Issues:

- Vanishing or exploding gradients during training (backpropagation through time).
- Poor long-range memory: The hidden state forgets older observations.

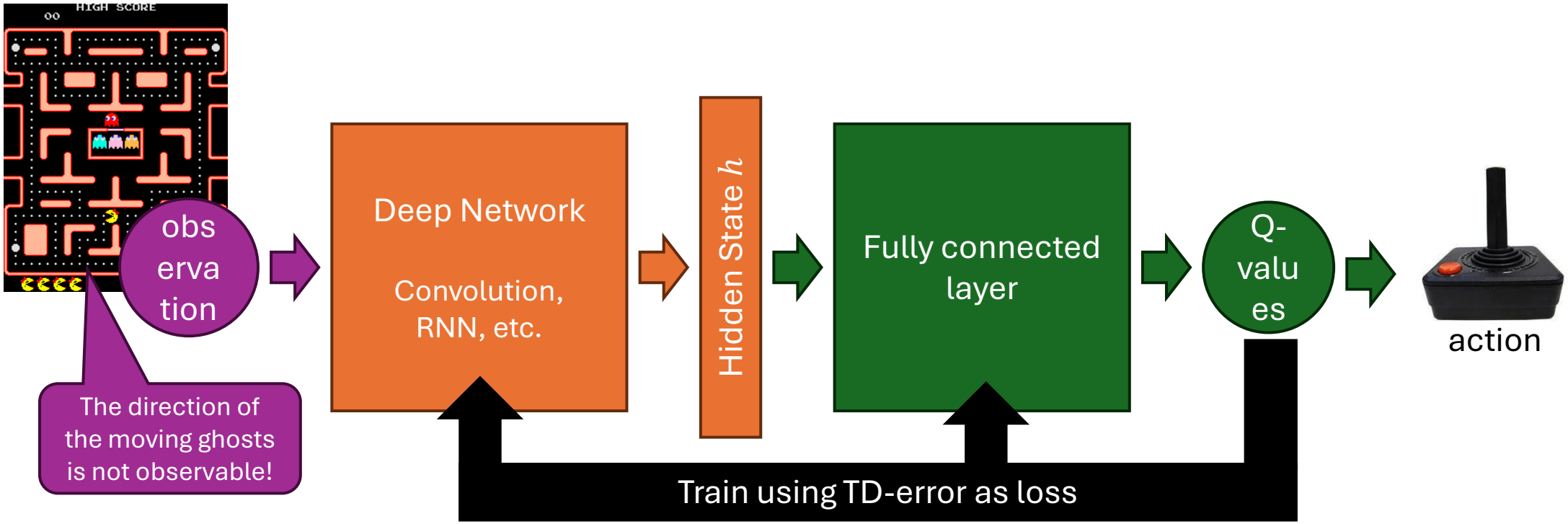
LSTMs can help with these issues.

Example

- **Idea**: Use a deep artificial neural network to create a hidden state that has sufficient information to make decisions. This is also called embed the observation into a latent space.

$$s_t^a = h_t = g(o_t, o_{t-1}, \dots, o_o|\theta_g).$$

- **Training**: Train the network to predict Q-values $Q(g(o_t, \dots, o_0|\theta_g), a|\theta_Q)$ by minimizing the 1-step temporal difference error (Q-learning). This tries to learn a good agent state representation based on observations.

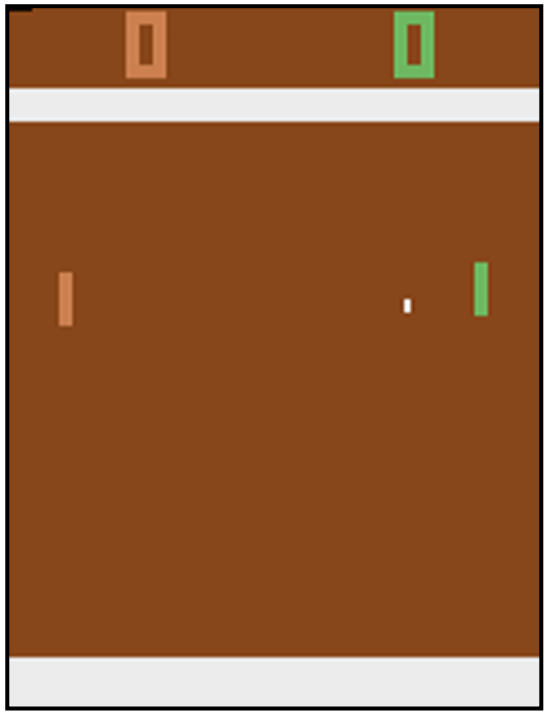


Atari Inc., 1972

Case Study: Atari Pong

Pong simulates table tennis. The players use paddles to hit a ball back and forth. Points are earned when the opponent fails to return the ball. The goal is for each player to reach eleven points before the opponent.

States and Observations

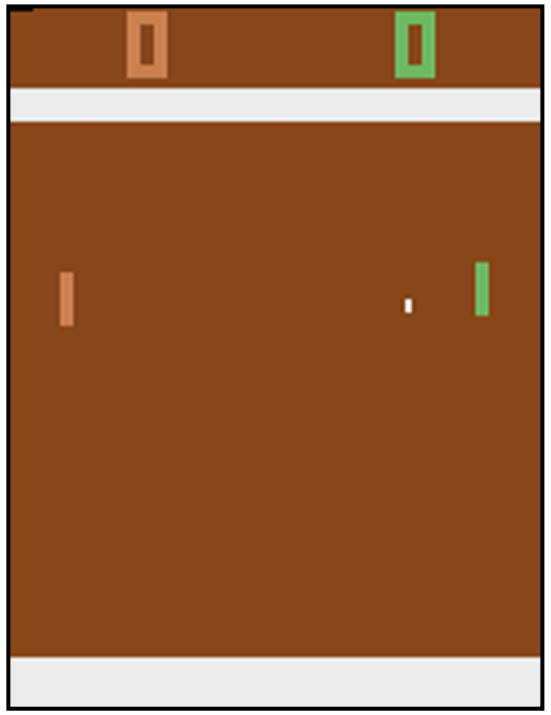


Actions: move paddle up or down.
Reward: +1 when the opponent cannot return the ball.

Define the components of:

- Environment state s_t^e:
- Observation o_t:
- Why is this partially observable?

Single Observation as State

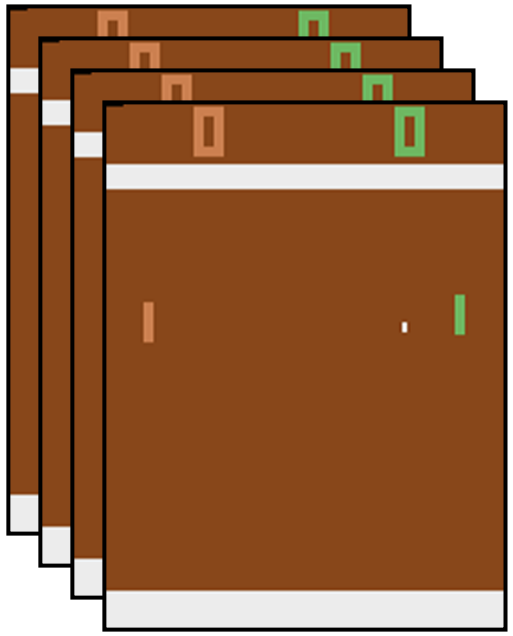


Feature from current frame

Learn an approximate $Q(o_t, a|\theta)$ using a TD method.

How well do you think this will work?

Frame Stacking



Feature from four frames

Learn an approximation

$$h_t = g(o_t, o_{t-1}, o_{t-2}, o_{t-3}|\theta_g)$$

$$Q(h_t, a|\theta_q)$$

using a TD method.

How well does this work? Why?

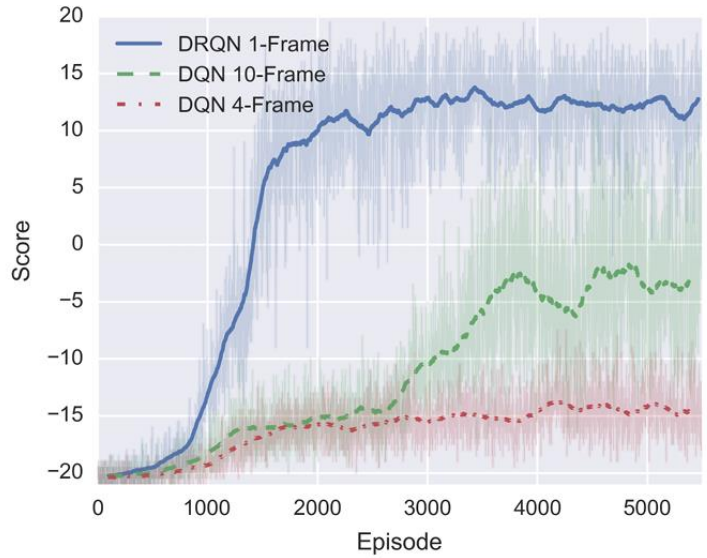
Deep Recurrent Q-Learning

Paper: Matthew Hausknecht and Peter Stone, Deep Recurrent Q-Learning for Partially Observable MDPs, 2015 AAAI Fall Symposium.

Approach: Deep belief monitoring using

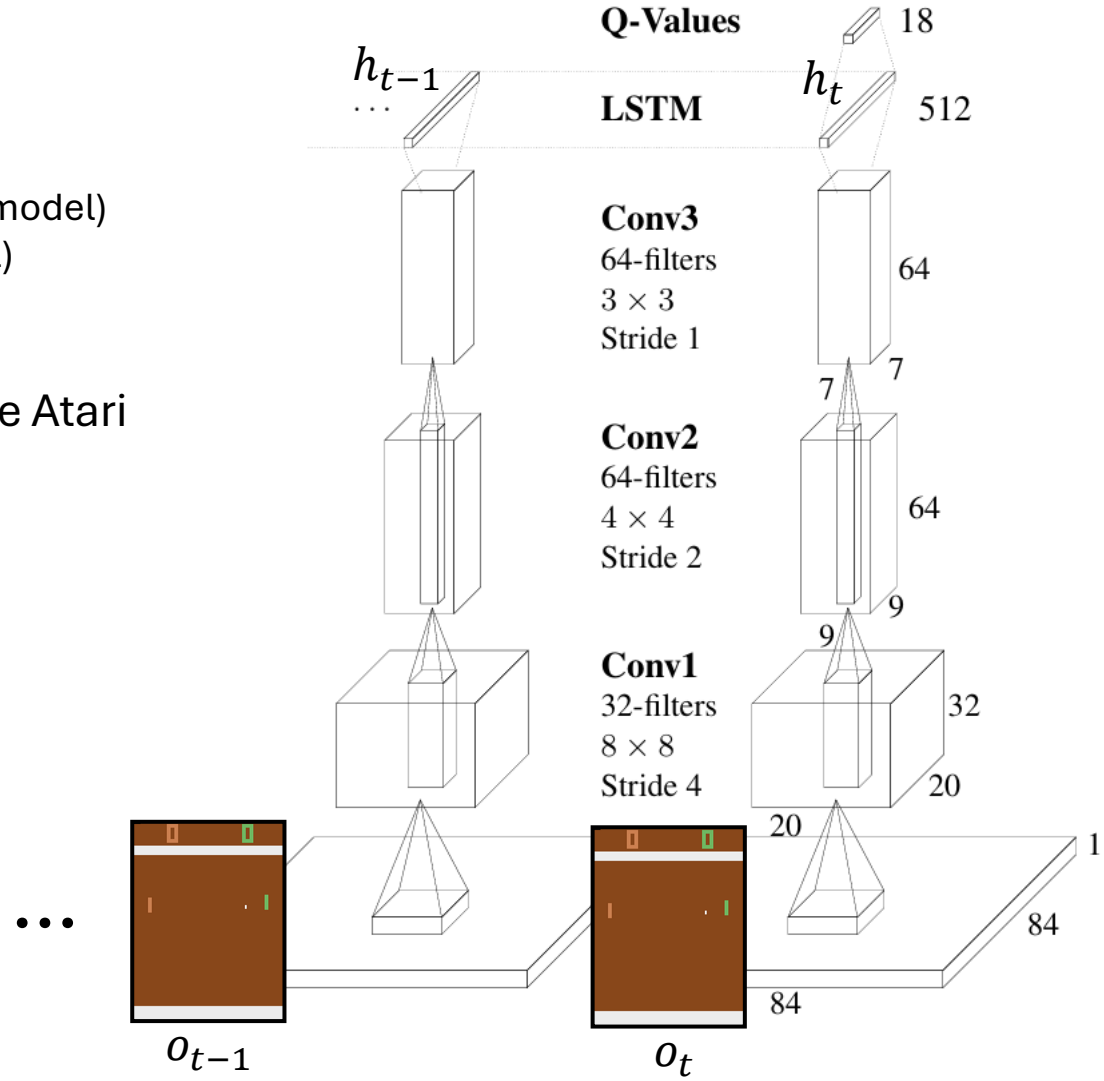
- convolution to process the screen image (observation model)
- RNN (LSTM) to update the agent state (transition model)
- A fully connected layer to predict Q-values

- **Results**: Works better than frame stacking on some Atari games.



Architecture

$$Q(g(o_t, \ldots, o_0|\theta_g), a|\,\theta_Q)$$



What You Should Know

- How observations are related to the environment state and the agent state.
- The relationship between Markov Chains, MDPs, HMMs, and POMDPs.
- Why using (stacked) observations instead of states sometimes works.
- Belief states and belief monitoring. How deep neural networks can be used to learn belief states.