Outline

1. Reinforcement Learning
2. Simulation-Based Search
3. Planning Under Uncertainty
Markov Decision Process

Definition

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

- $S$ is a finite set of states
- $A$ is a finite set of actions
- $P$ is a state transition probability matrix, $P_{ss'} = P[s' \mid s, a]$
- $R$ is a reward function, $R_s^a = E[r \mid s, a]$

Assume for this talk that all sequences terminate, $\gamma = 1$
Planning and Reinforcement Learning

- **Planning:**
  Given MDP $\mathcal{M}$, maximise expected future reward

- **Reinforcement Learning:**
  Given sample sequences from MDP

$$\{s_1, a_1^k, r_1^k, s_2^k, a_2^k, ..., s_T^k\}_{k=1}^K \sim \mathcal{M}$$

Maximise expected future reward
A simulator $\mathcal{M}$ is a generative model of an MDP

- Given a state $s_t$ and action $a_t$
- The simulator can generate a next state $s_{t+1}$ and reward $r_{t+1}$

A simulator can be used to generate sequences of experience

Starting from any “root” state $s_1$

$$\{s_1, a_1, r_1, s_2, a_2, \ldots, s_T\} \sim \mathcal{M}$$

Simulation-based search applies reinforcement learning to simulated experience
Monte-Carlo Simulation

- Given a model $\mathcal{M}$ and a simulation policy
  \[ \pi(s, a) = Pr(a \mid s) \]

- Simulate $K$ episodes from root state $s_1$
  \[ \{s_1, a_1^k, r_1^k, s_2^k, a_2^k, \ldots, s_T^k\}_{k=1}^K \sim \mathcal{M}, \pi \]

- Evaluate state by mean total reward (Monte-Carlo evaluation)
  \[
  V(s_1) = \frac{1}{K} \sum_{k=1}^{K} \sum_{t=1}^{T_k} r_t^k \xrightarrow{P} \mathbb{E} \left[ \sum_{t=1}^{T_k} r_t^k \mid s_1 \right]
  \]
Simple Monte-Carlo Search

- Given a model $\mathcal{M}$ and a simulation policy $\pi$
- For each action $a \in \mathcal{A}$
  - Simulate $K$ episodes from root state $s_t$
    \[
    \{s_1, a, a_1^k, r_2^k, s_2^k, a_2^k, \ldots, s_T^k\}_{k=1}^K \sim \mathcal{M}, \pi
    \]
- Evaluate actions by mean total reward
  \[
  Q(s_1, a) = \frac{1}{K} \sum_{k=1}^K \sum_{t=1}^{T^k} r_t^k \xrightarrow{P} \mathbb{E} \left[ \sum_{t=1}^{T^k} r_t^k \middle| s_1, a \right]
  \]
- Select real action with maximum value
  \[
  a_t = \arg\max_{a \in \mathcal{A}} Q(s_t, a)
  \]
Monte-Carlo Tree Search

- Simulate sequences starting from root state $s_1$
- Build a search tree containing all visited states
- Repeat (each simulation)
  - Evaluate states $V(s)$ by mean total reward of all sequences through node $s$
  - Improve simulation policy by picking child $s'$ with max $V(s')$
- Converges on the optimal search tree, $V(s) \to V^*(s)$
Reinforcement Learning and Simulation-Based Search

Simulation-Based Search

Monte-Carlo Search

max

0/1 → a₁ → 3/4 → a₁ → 0/1

min

0/1 → b₁ → 2/2 → b₁ → 1/1

max

2/2 → a₁ → 1/1

min

9/12 → a₂ → 6/7 → b₁ → 6/7

max

9/12 → a₃ → 2/3 → b₁ → 2/3

min

0/1 → b₂ → 1/1

search tree

roll-outs

reward

root
Advantages of MC Tree Search

- Highly selective best-first search
- Focused on the future
- Uses sampling to break curse of dimensionality
- Works for “black-box” simulators (only requires samples)
- Computationally efficient, anytime, parallelisable
Disadvantages of MC Tree Search

- Monte-Carlo estimates have high variance
- No generalisation between related states
Temporal-Difference Search

- Simulate sequences starting from root state $s_1$
- Build a search tree containing all visited states
- Repeat (each simulation)
  - Evaluate states $V(s)$ by temporal-difference learning
  - Improve simulation policy by picking child $s'$ with $\max V(s')$
- Converges on the optimal search tree, $V(s) \rightarrow V^*(s)$
Linear Temporal-Difference Search

- Simulate sequences starting from root state $s_1$
- Build a linear function approximator $V(s) = \phi(s)^\top \theta$ over all visited states
- Repeat (each simulation)
  - Evaluate states $V(s)$ by linear temporal-difference learning
  - Improve simulation policy by picking child $s'$ with $\max V(s')$
Demo
Consider a history $h_t$ of actions, observations and rewards

$$h = a_1, o_1, r_1, ..., a_t, o_t, r_t$$

What if the state $s$ is unknown?
- i.e. we only have some beliefs $b(s) = P(s \mid h_t)$

What if the MDP dynamics $P$ are unknown?
- i.e. we only have some beliefs $b(P) = p(P \mid h_t)$

What if the MDP reward function $R$ is unknown?
- i.e. we only have some beliefs $b(R) = p(R \mid h_t)$
Belief State MDP

- Plan in augmented state space over beliefs
- Each action now transitions to a new belief state
- This defines an enormous MDP over belief states
Histories and Belief States
We can apply simulation-based search to the belief state MDP since these methods are effective in very large state spaces. Unfortunately, updating belief states is slow. Belief state planners cannot scale up to realistic problems.
Root Sampling

- Each simulation, pick one world from root beliefs: sample state/transitions/reward function
- Run simulation as if that world is real
- Build plan in history space (fast!)
- Evaluate histories $V(h)$ e.g. by Monte-Carlo evaluation
- Improve simulation policy e.g. by greedy action selection
  $a_t = \text{argmax}_a V(h_t a)$
- Never updates beliefs during search
- But still converges on the optimal search tree w.r.t. beliefs, $V(h) \rightarrow V^*(h)$
- Intuitively, it averages over different worlds, tree provides filter
Demo