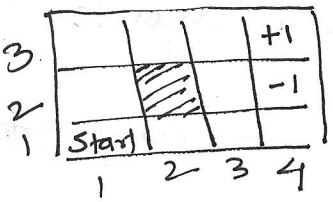
AI - Sequential Decision problems



} Sequential decision problem of a 4×3 environment.

- An agent has to start from Start & reach +1/-1.
- Probability of moving up is 0.8 & that of moving left/right is 0.1
- On reaching goal, the reward is +1/-1 & all other states have -0.04 as reward.
- Available actions are Up, Down, Left & Right.
- The env. is assumed to be fully observable & also not deterministic.
- A specification of the outcome probabilities for each action in each possible state is called a transition model
- $T(s, a, s')$ denote probability of reaching from state s to s' with action 'a'. Can be considered as a 3-D table.
- The probability is assumed to be Markovian - the probability depends only the immediately previous state only & not on the earlier states.
- The utility fn with depend on sequence of states - an environmental history - rather than a single state.
- If the utility f^n is calculated as sum of the rewards recieved, then to reach +1 state, the utility will be 0.6 after 10 steps.
- The specification of a sequential decision problem for a fully observable environment with a Markovian transitional model & additive rewards is called a Markovian decision process (MDP). Its components are

Initial State: S_0

Transition Model: $T(s, a, s')$

Reward Function: $R(s)$ / $R(s, a, s')$

Solution / Policy

- A fixed action sequence can't be soln as an agent might end in a non-goal state.
- Soln must specify what is to be done for every state that agent might reach.
- A soln of this kind is called a policy {π}.

The Policy of state 's' is denoted as $\pi(s)$

- An agent with complete policy always knows what do next.
- The stochastic nature of the env. will lead to diff. environmental history each time policy is run.
- The quality of a policy is ∴ measured by expected utility of the possible env. histories generated by that policy.
- An optimal policy is a policy that generates highest expected utility. denoted as π^*
- Given π^* an agent consults its current percept to obtain the current state, & then executing action $\pi^*(s)$.
- A policy that is guaranteed to reach terminal state is called proper policy; can use $\lambda = 1$ (additive rewards)

Optimality in sequence decision problem

- To find possible choices for performance measures - choices for the utility f^h on env. histories. represented as $U_h([s_0, s_1, \ldots s_n])$.
- Finite horizon - There is a fixed time N after which nothing matters.

  Thus $U_h([s_0, s_1, \ldots s_{N+k}]) = U_h([s_0, s_1, s_N])$ for $\forall k > 0$.
- In finite horizon, the optimal action in a given state could change over time, it is non-stationary. In stationary optimal policy, optimal action depends upon current state.

- Policies for infinite horizon are simpler compare to finite horizon.
- Infinite horizon necessarily means that there no fixed deadlines & not infinite state sequences.
- The utility of state sequence can be calculated using multi-attribute utility theory where each state s_i is viewed as an attribute of $[s_0, s_1, \ldots s_n]$
- 2 ways to assign utilities to sequence
  + Additive Rewards: $U_h([s_0, s_1, \ldots s_n]) = R(s_0) + R(s_1) + \ldots$
  + Discounted Rewards: $U_h([s_0, s_1, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$

  where γ is a discount factor bet^w 0 & 1. It dec^s the preference of the agent for current rewards over future rewards — 0 being used for no preference for rewards in distant future. A discount factor of γ is equivalent to interest rate of $(1/\gamma) - 1$

- A value of policy is the expected sum of discounted rewards obtained where the expectation is taken over all possible state sequences that could occur, given the policy is executed. An optimal policy π^* is

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi\right]$$

Utilities of State

- Utility of states is defined in terms of the utility of State sequences.
- Roughly, the utility of a state is the utility of the state sequence that might follow it.

- If s_t be the state an agent reaches after executing π for 't' steps, then we have

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \,\middle|\, \pi, s_0 = s\right]$$

- The true utility of the state 's' i.e. is ($U^{\pi^*}(s) =$) expected sum of discounted rewards on executing the optimal policy.
- The utility of $U(s)$ allows to select action using Maximum Expected Utility – choose the action that maximizes the expected utility of sub. state

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \left(\sum_{s'} T(s,a,s') U(s')\right).$$

- The utility of state is the immediate reward for that state plus the expected discounted utility of the next state, assuming the agent chooses the optimal action.

$$U(s) = R(s) + \gamma \max_a \sum T(s,a,s') U(s')$$

Bellman's Equation

- Value Iteration algo:
  + Iterate to find optimal utility

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum P(s' \mid s,a) U_i(s')$$

Policy Iteration:
- Alternates betw 2 steps
  + Policy Evaluation: given π_i, calculate $U_i = U^{\pi_i}$
  + Policy Improvement: Calculate MEU policy π_{i+1}, using one step look ahead.

$$U_i(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi_i(s)) U_i(s')$$

  + Modified policy Iteration: Not all // Simplified steps

$$U_{i+1} \leftarrow R(s) + \gamma \sum_{s'} P(s' \mid s, \pi_i(s)) U_i(s')$$

Policy Iteration

- The policy iteration algorithm iterates bet^{w}. 2 steps
  1. Policy Evaluation: Given a policy Π_i, calculate U_i for the utility of each state if Π_i were to be executed
  2. Policy Improvement: Calculate a new maximum expected utility (MEU) policy Π_{i+1}, using one step look ahead based on U_i.

The algo. terminates when policy improvement step yields no improvement in the utilities.

- Now, U_i is a fixed point in Bellman's eqn & Π_i must be the optimal policy.
- Implementing PI routine is much better than solving Bellman's Eqn.
- An i^{th} iteration specifies action $\Pi_i(s)$ for Π_i in 's'.
- Leads to simplified version of Bellman's eqn.

$$U(i)(s) = R(s) + \gamma \sum_{s'} T(s, \Pi_i(s), s') U_i(s')$$

The eqn is also linear. For 'n' states there will be 'n' linear eqns which can be solved in $O(n^3)$ by standard linear algebra methods.

- For small state space, policy evaluation using exact soln methods is often the most efficient approach.
- For large state space, the no. of iterations can be reduced to give reasonably good approximation of the utilities. The simplified Bellman's eqn becomes.

$$U_{i+1}(s) \leftarrow R(s) + \gamma \sum_{s'} T(s, \Pi_i(s), s') U_i(s').$$

This is called modified policy iteration & much more efficient than standard policy iteration or value iteration algos.

- If instead of updating all states, only certain states' utility or policy could be updated. This is asynchronous policy updation. Allows to design better heuristic algorithms.

- AShish R. Gavande