CAS CS 440
Lec 29

Policy Learning III

1. Policy Iteration
    a. Start with some initial policy (can be chosen at random)
    b. Alternate between two steps:
        i. Policy evaluation:
            1. Given $\pi_{i'}$, calculate $\vec{u}_i = U^{\pi_i}$
        ii. Policy improvement:
            1. Calculate new MEU policy π_{i+1} using one-step lookahead (bellman equation) from $\vec{u}_i$
    c. Terminate when policy improvement → no changes in utility values
        i. Utility values converge between iterations
2. Policy Evaluation Step
    a. Don't need to solve bellman equations (value iteration)
        i. We know the policy! Action is already decided!
    b. Run the policy!
        i. Simplified bellman equation (no max…action is chosen)

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

            1. Called simplified value iter
        ii. Set of linear equations! Can solve (with solver)!
        iii. For big state spaces sometimes faster to solve iteratively (do this k times):

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

            1. Policy iter → modified policy iter

3. The Policy Iteration Algorithm

function POLICY-ITERATION(mdp) **returns** a policy
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$
  local variables: U, a vector of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random
  repeat
    $U \leftarrow$ POLICY-EVALUATION(π, U, mdp) Recalculate utilities
    $unchanged? \leftarrow$ true
    for each state s **in** S **do**
      if $\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s'] > \sum_{s'} P(s' \mid s, \pi[s])\ U[s']$ **then do**
        $\pi[s] \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']$
        $unchanged? \leftarrow$ false
  until $unchanged?$
  return π

Check if policy needs to change

a.

4. Practical Use
   a. In practice, we don't need to fix our algorithm
      i. Operate on subsets of states at a time (rather than all of them at once)
      ii. Within a single iteration:
         1. Pick any subset of states
         2. Apply either kind of updating (policy improvement / simplified value iteration)
      iii. Algorithm is called asynchronous policy iteration
      iv. How to pick subset?
         1. Heuristics!
         2. Focus on subsets which are likely to be reached by a good policy