CAS CS 440
Lec 32

Policy Learning V: Reinforcement Learning III

1. State-Action-Reward-State-Action (SARSA)
    a. Close relative of Q-learning
    b. Q-learning rule:

    $$Q(s,a) \leftarrow Q(s,a) + \alpha\left(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

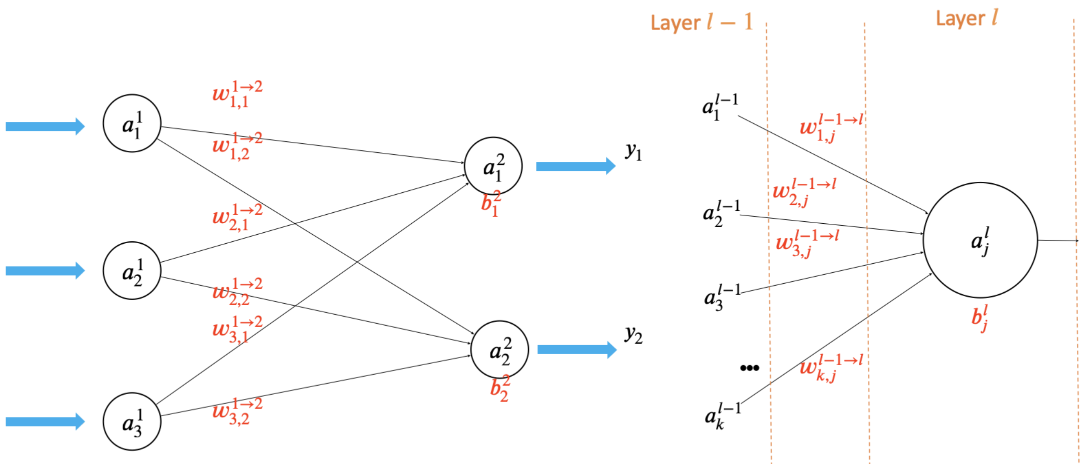
    c. SARSA learning rule:

    $$Q(s,a) \leftarrow Q(s,a) + \alpha\big(R(s) + \gamma\, Q(s',a\ ) - Q(s,a)\big)$$

    d. The difference:
        i. Q-learning uses best action (ignores actual action taken)
            1. More flexible
        ii. SARSA uses actual action taken (ignores best action unless chosen)
            1. Better if policy depends (even in part) on (an)other agent(s)
2. Generalized RL
    a. So far, functions have been dictionaries (big lookup tables):
        i. Q(s, a) is big dictionary
        ii. U(s) is a big dictionary
    b. Doesn't scale
        i. Need an approach which uses less memory
    c. Function approximation
        i. Represent function f(x) with an approximate function $g_\theta(x)$
3. Function Approximation
    a. Compress entire table into 'n' parameters
        i. Might not be able to do exactly, but "good enough"
        ii. Share knowledge between inputs through parameters
            1. Generalization?
    b. 'n' controls size of hypothesis space
        i. Want "true" function to be a candidate
        ii. Tradeoff: size of hypothesis space (i.e. 'n') vs. learning time
4. Neural Networks
    a. Neural Networks are really powerful function approximators
        i. Really useful models in their own right
        ii. We will use them for function approximators in RL

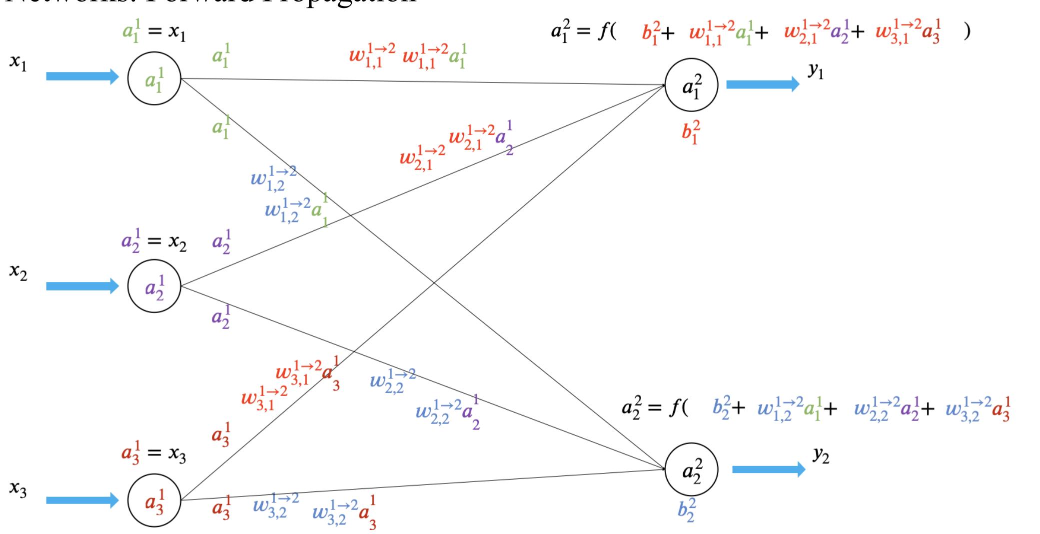
$$a_j^l = \begin{cases} f\left(b_j^l + \sum_{i=1}^{k} w_i^{l-1\to l} a_i^{l-1}\right) & l > 1 \\ x_j & otherwise \end{cases}$$

b.



c.

5. Neural Networks: Forward Propagation



a.

6. Active Q-Agent
   a. Agent updates Q-function after transition s → s' with action a
   b. Agent uses exploratory function 'f' to sometimes ignore policy

function Q-LEARNING-AGENT($percept$) **returns** an action
  inputs: $percept$, a percept indicating the current state s' and reward signal r'
  persistent: Q, a table of action values indexed by state and action, initially zero
    N_{sa}, a table of frequencies for state–action pairs, initially zero
    s, a, r, the previous state, action, and reward, initially null

  if TERMINAL?(s) **then** $Q[s, None] \leftarrow r'$
  if s is not null **then**
    increment $N_{sa}[s, a]$
    $Q[s,a] \leftarrow Q[s,a] + \alpha(N_{sa}[s,a])(r + \gamma \max_{a'} Q[s',a'] - Q[s,a])$
  $s, a, r \leftarrow s', \mathrm{argmax}_{a'} f(Q[s',a'], N_{sa}[s',a']), r'$
  return a

c.