ITCS 6156/8156 Fall 2023
Machine Learning

Reinforcement Learning

Instructor: Hongfei Xue
Email: hongfei.xue@charlotte.edu
Class Meeting: Mon & Wed, 4:00 PM – 5:15 PM, CHHS 376

UNIVERSITY OF NORTH CAROLINA CHARLOTTE

Some content in the slides is based on Dr. Raquel Urtasun's lecture

MDP Formulation

- Goal: find policy π that maximizes expected accumulated future rewards $V^\pi(s_t)$, obtained by following π from state s_t:

$$\begin{aligned} V^\pi(s_t) &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots \\ &= \sum_{i=0}^{\infty} \gamma^i r_{t+i} \end{aligned}$$

- Game show example:
  - assume series of questions, increasingly difficult, but increasing payoff
  - choice: accept accumulated earnings and quit; or continue and risk losing everything
- Notice that:

$$V^\pi(s_t) = r_t + \gamma V^\pi(s_{t+1})$$

What to Learn

- We might try to learn the function V (which we write as V^*)

$$V^*(s) = \max_a [r(s,a) + \gamma V^*(\delta(s,a))]$$

- Here $\delta(s,a)$ gives the next state, if we perform action a in current state s
- We could then do a lookahead search to choose best action from any state s:

$$\pi^*(s) = arg \max_a [r(s,a) + \gamma V^*(\delta(s,a))]$$

- But there's a problem:
  - This works well if we know $\delta()$ and $r()$
  - But when we don't, we cannot choose actions this way

Q Learning

- Define a new function very similar to V^*

$$Q(s,a) = r(s,a) + \gamma V^*(\delta(s,a))$$

- If we learn Q, we can choose the optimal action even without knowing δ!

$$\begin{aligned} \pi^*(s) &= \operatorname{\mathit{arg}} \max_a [r(s,a) + \gamma V^*(\delta(s,a))] \\ &= \operatorname{\mathit{arg}} \max_a Q(s,a) \end{aligned}$$

- Q is then the evaluation function we will learn

Training Rule to Learn Q

- Q and V^* are closely related:

$$V^*(s) = \max_a Q(s,a)$$

- So we can write Q recursively:

$$\begin{aligned} Q(s_t, a_t) &= r(s_t, a_t) + \gamma V^*(\delta(s_t, a_t)) \\ &= r(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') \end{aligned}$$

- Let $\hat{Q}$ denote the learner's current approximation to Q
- Consider training rule

$$\hat{Q}(s,a) \leftarrow r(s,a) + \gamma \max_{a'} \hat{Q}(s', a')$$

where s' is state resulting from applying action a in state s

Q Learning for Deterministic World

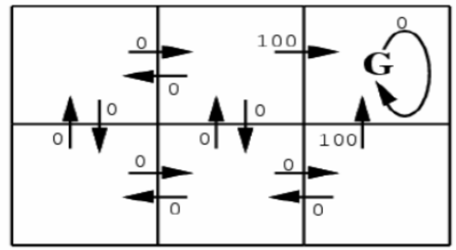
- For each s, a initialize table entry $\hat{Q}(s, a) \leftarrow 0$
- Start in some initial state s
- Do forever:
  - Select an action a and execute it
  - Receive immediate reward r
  - Observe the new state s'
  - Update the table entry for $\hat{Q}(s, a)$ using Q learning rule:

$$\hat{Q}(s, a) \leftarrow r(s, a) + \gamma \max_{a'} \hat{Q}(s', a')$$

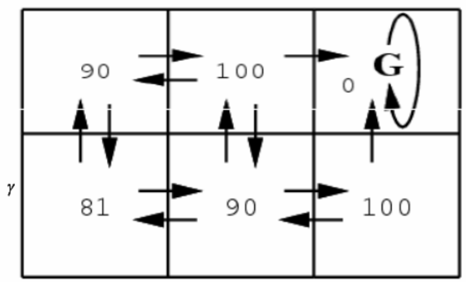
  - $s \leftarrow s'$
- If we get to absorbing state, restart to initial state, and run thru "Do forever" loop until reach absorbing state

Q Learning

$\gamma = 0.9$

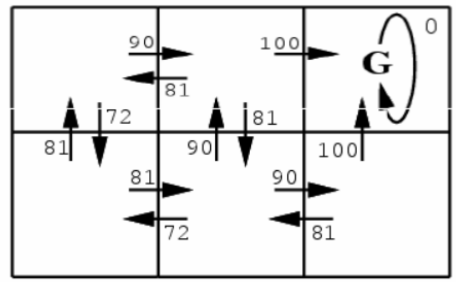


$r(s, a)$ (immediate reward) values

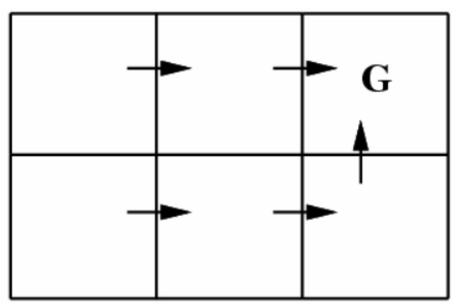


$V^*(s)$ values

$$V^*(s_5) = 0 + \gamma 100 + \gamma^2 0 + \ldots = 90$$



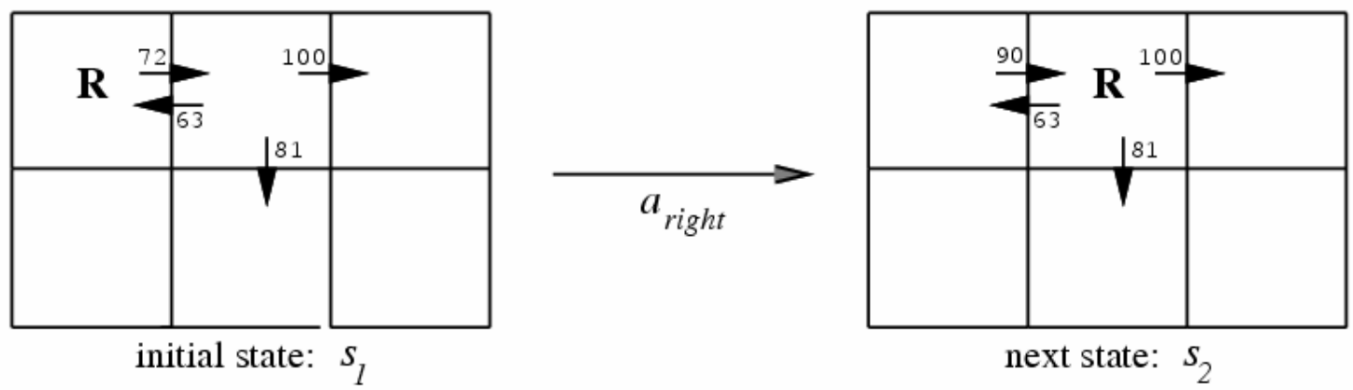
$Q(s, a)$ values



One optimal policy

Updating Estimated Q

- Assume the robot is in state s_1; some of its current estimates of Q are as shown; executes rightward move



$$\begin{aligned}\hat{Q}(s_1, a_{right}) &\leftarrow r + \gamma \max_{a'} \hat{Q}(s_2, a') \\ &\leftarrow r + 0.9 \max_{a}\{63, 81, 100\} \leftarrow 90\end{aligned}$$

- Important observation: at each time step (making an action a in state s **only one** entry of $\hat{Q}$ will change (the entry $\hat{Q}(s, a)$)
- Notice that if rewards are non-negative, then $\hat{Q}$ values only increase from 0, approach true Q

Q Learning: Summary

- Training set consists of series of intervals (episodes): sequence of (state, action, reward) triples, end at absorbing state
- Each executed action a results in transition from state s_i to s_j; algorithm updates $\hat{Q}(s_i, a)$ using the learning rule
- Intuition for simple grid world, reward only upon entering goal state $\rightarrow Q$ estimates improve from goal state back
  1. All $\hat{Q}(s, a)$ start at 0
  2. First episode – only update $\hat{Q}(s, a)$ for transition leading to goal state
  3. Next episode – if go thru this next-to-last transition, will update $\hat{Q}(s, a)$ another step back
  4. Eventually propagate information from transitions with non-zero reward throughout state-action space

Q Learning: Exploration/Exploitation

- Have not specified how actions chosen (during learning)
- Can choose actions to maximize $\hat{Q}(s, a)$
- Good idea?
- Can instead employ stochastic action selection (policy):

$$P(a_i|s) = \frac{\exp(k\hat{Q}(s, a_i))}{\sum_j \exp(k\hat{Q}(s, a_j))}$$

- Can vary k during learning
  - more exploration early on, shift towards exploitation

Non-deterministic Case

- What if reward and next state are non-deterministic?
- We redefine V, Q based on probabilistic estimates, expected values of them:

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots] \\ &= E_{\pi}[\sum_{i=0}^{\infty} \gamma^i r_{t+i}] \end{aligned}$$

and

$$\begin{aligned} Q(s,a) &= E[r(s,a) + \gamma V^*(\delta(s,a))] \\ &= E[r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} Q(s',a')] \end{aligned}$$

Non-deterministic Case: Learning Q

- Training rule does not converge (can keep changing $\hat{Q}$ even if initialized to true Q values)
- So modify training rule to change more slowly

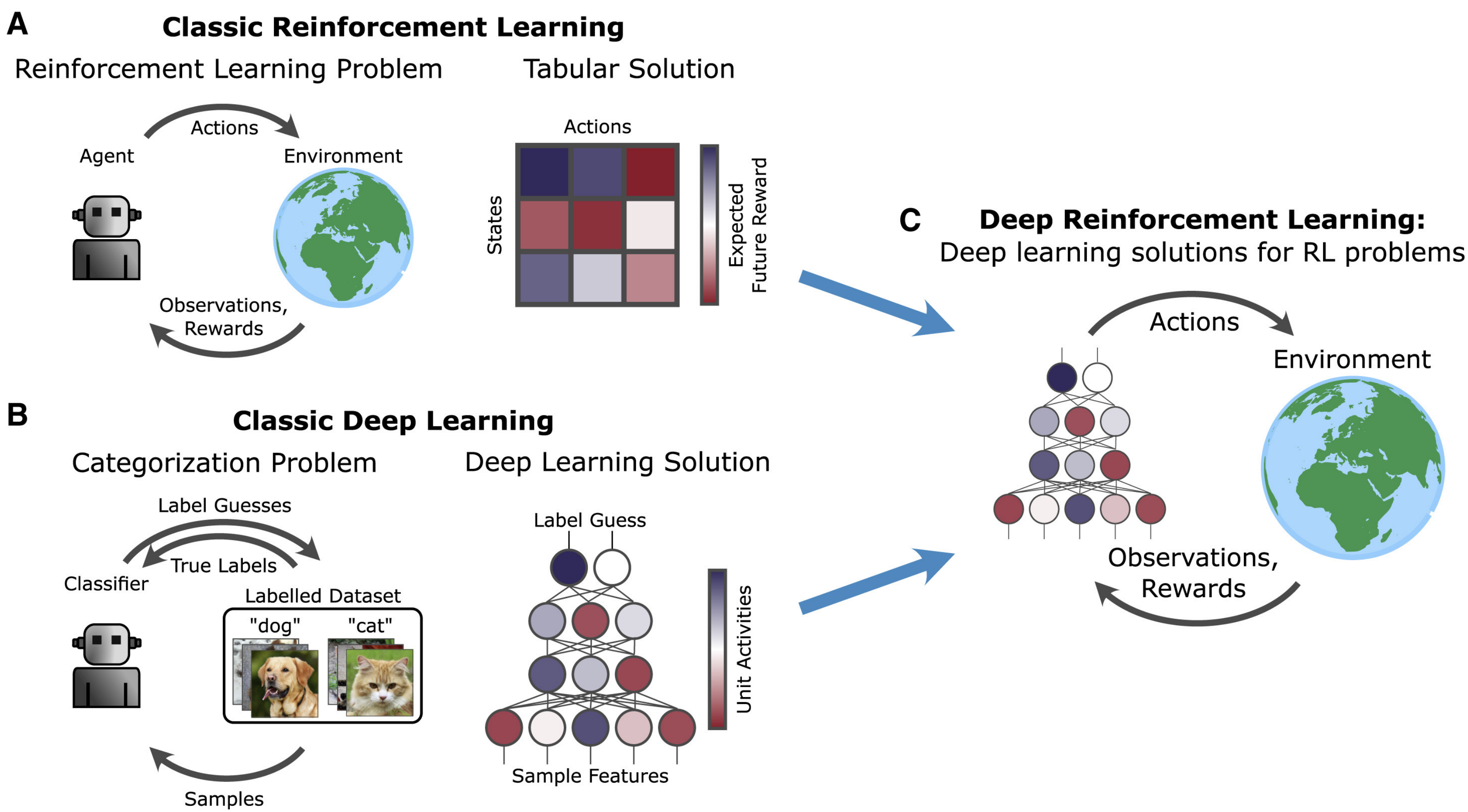
$$\hat{Q}(s,a) \leftarrow (1-\alpha_n)\hat{Q}_{n-1}(s,a) + \alpha_n[r + \gamma \max_{a'} \hat{Q}_{n-1}(s',a')]$$

where s' is the state land in after s, and a' indexes the actions that can be taken in state s'

$$\alpha_n = \frac{1}{1 + \text{visits}_n(s,a)}$$

where visits is the number of times action a is taken in state s

Deep Reinforcement Learning



From https://www.sciencedirect.com/science/article/pii/S0896627320304682

Questions?

UNIVERSITY OF NORTH CAROLINA
CHARLOTTE