

CSCI 5622 Machine Learning

ML Reinforcement Learning

DATE	TOPIC	DUE
		<i>Peer Feedback Overdue</i>
Wed, Dec 9	Reinforcement Learning	Peer Feedback Grades
Fri, Dec 11	-----	<u><i>Final Paper Due</i></u>
W 16 th 4PM	<u><i>1777 Exposition Dr, Room 102</i></u>	Final Presentations
.. (Dec 16)	<u><i>Mandatory attendance & participation</i></u>	<u><i>Ask Questions</i></u>

www.RodneyNielsen.com/teaching/CSCI5622-F09/

Instructor: Rodney Nielsen

Assistant Professor Adjunct, CU Dept. of Computer Science

Research Assistant Professor, DU, Dept. of Electrical & Computer Engr.

Research Scientist, Boulder Language Technologies

ML Reinforcement Learning

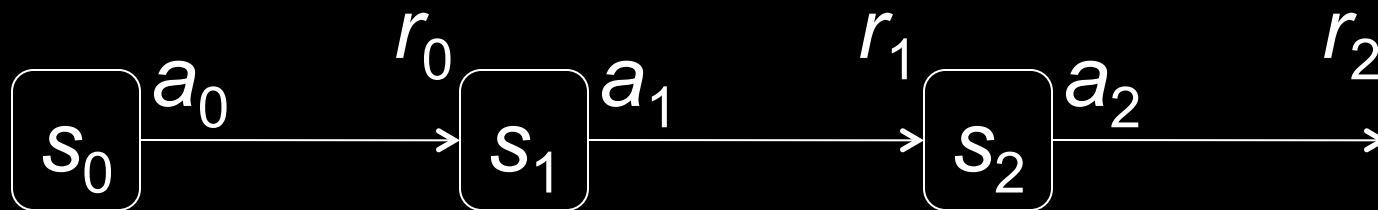
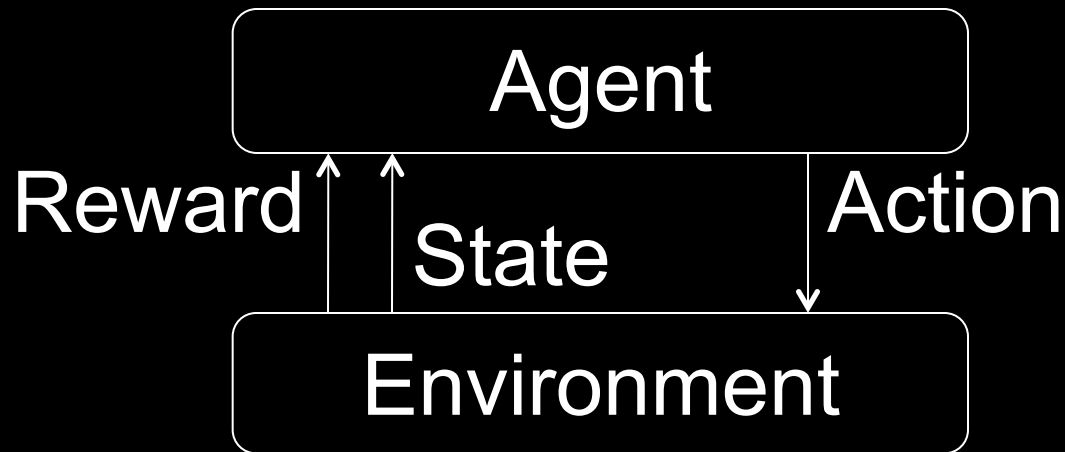
- Control learning
- Control policies that choose *optimal* actions
- Q-Learning
- Convergence

ML Control Learning

- Learning to choose actions:
 - Robot learning to dock with battery charger, ...
 - Learning optimal factory machine settings
 - Learning to play a game like backgammon
- Problem characteristics
 - Delayed reward
 - Opportunity for exploration
 - State may only be partially observable
 - Possibly multi-task learning from same inputs

- Learn to play backgammon
- Immediate reward
 - Win: +100
 - Lose: -100
 - Else: 0
- Trained by playing 1.5M games against itself
- Reached world-class player status

ML Reinforcement Learning



- Goal: Learn an action policy that maximizes reward: $r_0 + \gamma r_1 + \gamma^2 r_2 + \dots$; $0 \leq \gamma < 1$

ML Reinforcement Learning

- Finite set of states, S
- Finite set of actions, A
- At time, t , the agent observes the state, s_t in S
- Chooses an action a_t in A
- Receives the immediate reward r_t
- With a change of state to s_{t+1}

ML Markov Decision Processes

- Markov assumption: $s_{t+1} = \delta(s_t, a_t)$ and $r_t = r(s_t, a_t)$
- s_{t+1} and r_t depend only on the current state and action
- The functions δ and r might be nondeterministic
- δ and r are not necessarily known to the agent

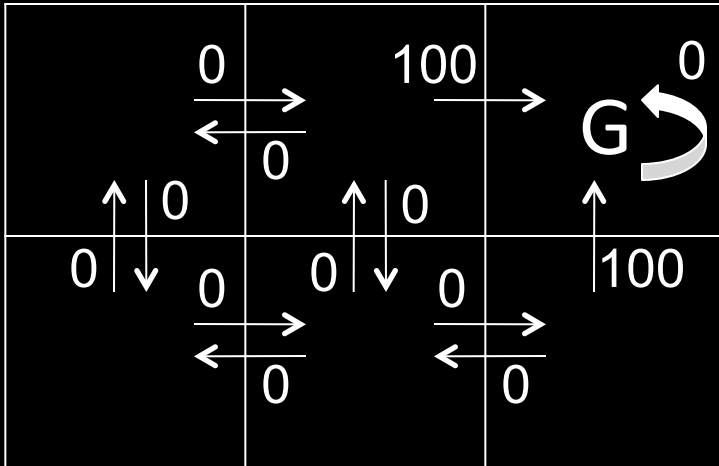
ML Agent's Learning Task

- Execute actions in environment, observe, and
 - Learn action policy $\pi : S \rightarrow A$, maximizing $E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots]$, starting anywhere in S
 - $0 \leq \gamma < 1$ is the discount factor for future rewards
- No training examples of the form $\langle s, a \rangle$
- Instead, training examples are $\langle \langle s, a \rangle, r \rangle$

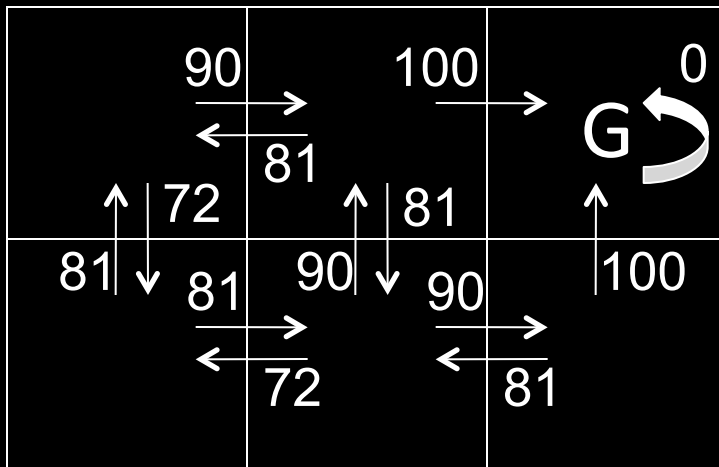
ML Value Function

- Consider deterministic worlds
- For each possible policy function π , define the evaluation function $V^\pi(s) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots$
 $= \sum_{i=0.. \infty} \gamma^i r_{t+i}$
- Where the r_k are generated by policy π starting at state s
- Goal: Learn the optimal policy π^*
$$\pi^* = \operatorname{argmax}_{\pi} V^\pi(s)$$

Rewards



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



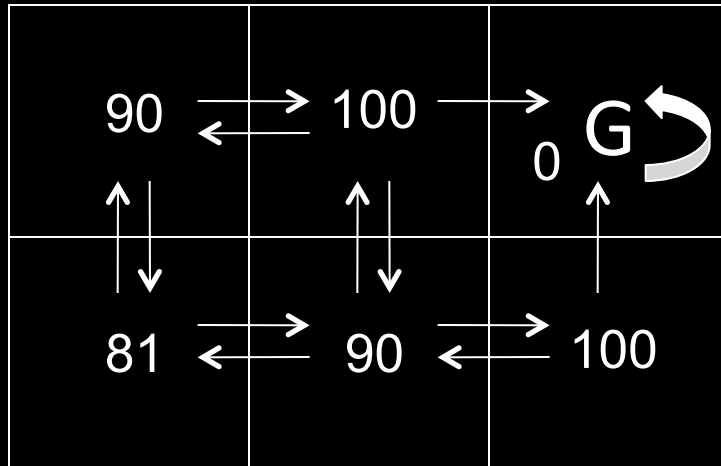
$Q(s, a)$ values

	90 → ← 81	100 →	G 0 ↻
81 ↑ ↓ 72	81 ↑ ↓ 81	90 ↑ ↓ 81	100 ↑
	← 72	← 81	

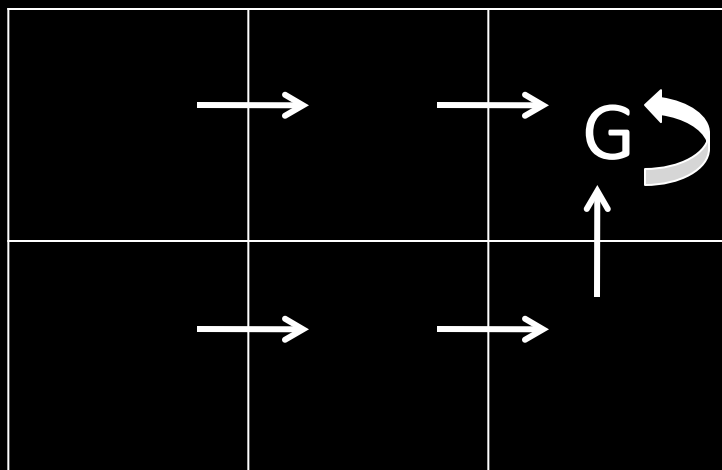
$Q(s, a)$ values

90	↔	100	→	G 0 ↻
↑ ↓		↑ ↓		↑
81	↔	90	↔	100

$V^*(s)$ values



$V^*(s)$ values



An optimal policy

- Option 1: $V^{\pi^*} = V^*$
- It could then do a lookahead search to choose best action from any state s because $\pi^*(s) = \operatorname{argmax}_a [r(s,a) + \gamma V^*(\delta(s,a))]$
- Problem:
 - Works well if agent knows $\delta : S \times A \rightarrow S$ and $r : S \times A \rightarrow \mathcal{R}$
 - But when it doesn't it can't choose actions this way

- Define a new function similar to V^*

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

- If agent learns Q , it can choose optimal action even without knowing δ

$$\pi^*(s) = \operatorname{argmax}_a [r(s,a) + \gamma V^*(\delta(s,a))]$$

$$\pi^*(s) = \operatorname{argmax}_a Q(s,a)$$

- Q is the evaluation function the agent will learn

ML Training Rule to Learn Q

- Note Q and V^* closely related

$$V^*(s) = \max_{a'} Q(s, a')$$

- Which allows us to write Q recursively as

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

- Let Q^\wedge denote learner's current approximation to Q . Consider training rule:

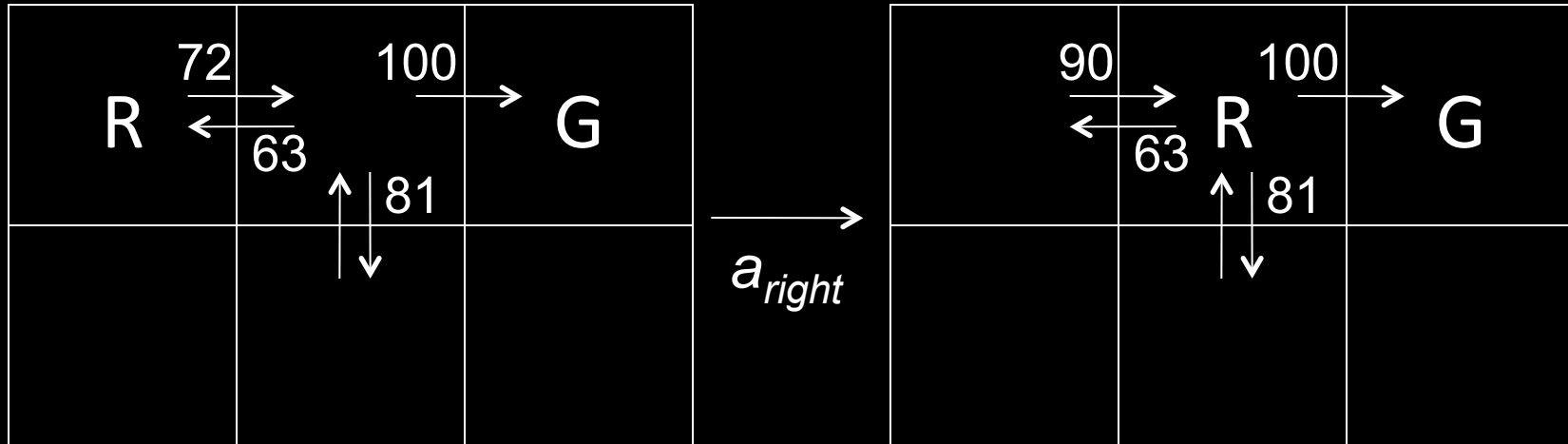
$$Q^\wedge(s, a) = r + \gamma \max_{a'} Q^\wedge(s', a')$$

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

ML Q Learning in Deterministic Cases

- For each s, a initialize table entry $Q^{\wedge}(s,a) \leftarrow 0$
- Observe current 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 $Q^{\wedge}(s,a)$ as follows:

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

Updating Q^\wedge 

$$\begin{aligned}
 Q^\wedge(s_1, a_{right}) &\leftarrow r + \gamma \max_{a'} Q^\wedge(s_2, a') \\
 &= 0 + 0.9 \max \{63, 81, 100\} = 90
 \end{aligned}$$

- If reward is non-negative, then for all $\langle s, a, n \rangle$

$$\begin{aligned}
 Q^\wedge_{n+1}(s, a) &\geq Q^\wedge_n(s, a), \text{ and} \\
 0 &\leq Q^\wedge_n(s, a) \leq Q(s, a)
 \end{aligned}$$

ML Convergence

- Q^{\wedge} converges to Q .
- The book shows a proof for the case of a deterministic world where we see each $\langle s, a \rangle$ visited infinitely often

ML Nondeterministic Case

- What if reward and next state are non-deterministic?
- We redefine V, Q by taking expected values

$$V^\pi(s) \leftarrow E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots]$$
$$= E[\sum_{i=0.. \infty} \gamma^i r_{t+i}]$$

$$Q(s,a) \leftarrow E[r(s,a) + \gamma V^*(\delta(s,a))]$$

ML Nondeterministic Case

- Q learning generalizes to nondeterministic worlds

- Alter training rule to:

$$Q_n^{\wedge}(s,a) \leftarrow (1-\alpha_n)Q_{n-1}^{\wedge}(s,a) + \alpha_n[r + \max_{a'} Q_{n-1}^{\wedge}(s',a')]$$

- where

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

- Can still prove convergence of Q^{\wedge} to Q

ML Temporal Difference (TD) Learning

- Q learning: reduce discrepancy between successive Q estimates
- One step time difference:

$$Q^{(1)}(s_t, a_t) \leftarrow r_t + \gamma \max_a Q^{\wedge}(s_{t+1}, a)$$

- Why not two steps?

$$Q^{(2)}(s_t, a_t) \leftarrow r_t + \gamma r_{t+1} + \gamma^2 \max_a Q^{\wedge}(s_{t+2}, a)$$

- Or n ?

$$Q^{(n)}(s_t, a_t) \leftarrow r_t + \gamma r_{t+1} + \dots + \gamma^{(n-1)} r_{t+n-1} + \gamma^n \max_a Q^{\wedge}(s_{t+n}, a)$$

- Blend all of these

ML TD Learning

- Blend all of time steps:

$$Q^\lambda(s_t, a_t) \leftarrow (1-\lambda)[Q^{(1)}(s_t, a_t) + \lambda Q^{(2)}(s_t, a_t) + \lambda^2 Q^{(3)}(s_t, a_t) + \dots] \\ = r_t + \gamma[(1-\lambda)\max_a Q^\lambda(s_t, a_t) + \lambda Q^\lambda(s_{t+1}, a_{t+1})]$$

- TD(λ) algorithm uses above training rule
- Sometimes converges faster than Q learning
- Converges for learning V^* for any $0 \leq \lambda \leq 1$
- TD-Gammon used this algorithm

ML Questions

Questions???

ML Presentations

– Wed, Dec 9: Tony & Xinyu

ML Projects

- Will stay until all questions are addressed, or
- Please make an appointment if you have any questions at all regarding your project