

Reinforcement Learning

Greg Grudic

Reinforcement Learning (RL)

Autonomous agent learns without human intervention

- *Agent learns by stochastically interacting with its environment, getting infrequent rewards*
- *Goal: maximize reward*

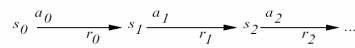
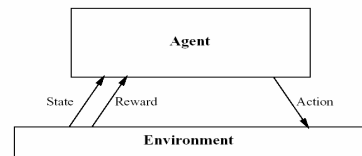
2

Reinforcement Learning

- Addresses the temporal credit assignment problem:
 - Delayed reward (HARD problem!)
- Some successful RL applications:
 - TD gammon (Tesauro)
 - Packing containers (Moore)
 - Elevator dispatch (Crites and Barto)

3

Reinforcement Learning Problem



Goal: Learn to choose actions that maximize

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \dots, \text{ where } 0 \leq \gamma < 1$$

4

Markov Decision Processes

Assume

- finite set of states S
- set of actions A
- at each discrete time agent observes state $s_t \in S$ and chooses action $a_t \in A$
- then receives immediate reward r_t
- and state changes to s_{t+1}
- Markov assumption: $s_{t+1} = \delta(s_t, a_t)$ and $r_t = r(s_t, a_t)$
 - i.e., r_t and s_{t+1} depend only on *current* state and action
 - functions δ and r may be nondeterministic
 - functions δ and r not necessarily known to agent

5

Agent's Learning Task

Execute actions in environment, observe results, and

- learn action policy $\pi : S \rightarrow A$ that maximizes

$$E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots]$$

from any starting state in S

- here $0 \leq \gamma < 1$ is the discount factor for future rewards

Different from supervised learning:

- Target function is $\pi : S \rightarrow A$
- but we have no training examples of form (s, a)
- training examples are of form $\langle (s, a), r \rangle$

6

Value Function

To begin, consider deterministic worlds...

For each possible policy π the agent might adopt, we can define an evaluation function over states

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

where r_t, r_{t+1}, \dots are generated by following policy π starting at state s

Restated, the task is to learn the optimal policy π^*

$$\pi^* \equiv \operatorname{argmax}_{\pi} V^\pi(s), (\forall s)$$

7

What to Learn

We might try to have agent learn the evaluation function V^{π^*} (which we write as V^*)

It could then do a lookahead search to choose best action from any state s because

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

A problem:

- This works well if agent knows $\delta : S \times A \rightarrow S$, and $r : S \times A \rightarrow \mathfrak{R}$
- But when it doesn't, it can't choose actions this way

8

Q Function

Define new function very similar to V^*

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

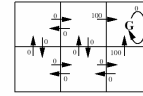
If agent learns Q , it can choose optimal action even without knowing δ !

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

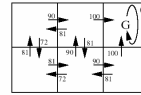
$$\pi^*(s) = \operatorname{argmax}_a Q(s, a)$$

Q is the evaluation function the agent will learn

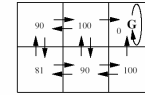
9



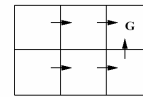
$r(s, a)$ (immediate reward) values



$Q(s, a)$ values ($\gamma = 0.9$)



$V^*(s)$ values ($\gamma = 0.9$)



One optimal policy

10

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

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

Nice! Let \hat{Q} denote learner's current approximation to Q . Consider training rule

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

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

11

Q Learning for Deterministic Worlds

For each s, a initialize table entry $Q(s, a) \leftarrow 0$

Observe current state s

Do forever:

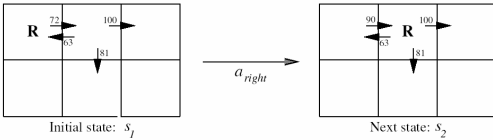
- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s, a)$ as follows:

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

- $s \leftarrow s'$

12

Updating \hat{Q}



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

notice if rewards non-negative, then

$$(\forall s, a, n) \quad \hat{Q}_{n+1}(s, a) \geq \hat{Q}_n(s, a)$$

and

$$(\forall s, a, n) \quad 0 \leq \hat{Q}_n(s, a) \leq Q(s, a)$$

13

\hat{Q} converges to Q . Consider case of deterministic world where see each (s, a) visited infinitely often.

Proof: Define a full interval to be an interval during which each (s, a) is visited. During each full interval the largest error in \hat{Q} table is reduced by factor of γ

Let \hat{Q}_n be table after n updates, and Δ_n be the maximum error in \hat{Q}_n ; that is

$$\Delta_n = \max_{s,a} |\hat{Q}_n(s, a) - Q(s, a)|$$

For any table entry $\hat{Q}_n(s, a)$ updated on iteration $n+1$, the error in the revised estimate $\hat{Q}_{n+1}(s, a)$ is

$$\begin{aligned} |\hat{Q}_{n+1}(s, a) - Q(s, a)| &= |(r + \gamma \max_{a'} \hat{Q}_n(s', a')) - (r + \gamma \max_{a'} Q(s', a'))| \\ &= \gamma |\max_{a'} \hat{Q}_n(s', a') - \max_{a'} Q(s', a')| \\ &\leq \gamma \max_{s', a'} |\hat{Q}_n(s', a') - Q(s', a')| \\ &\leq \gamma \max_{s', a'} |\hat{Q}_n(s', a') - Q(s', a')| \\ |\hat{Q}_{n+1}(s, a) - Q(s, a)| &\leq \gamma \Delta_n \end{aligned}$$

Note we used general fact that

$$|\max_a f_1(a) - \max_a f_2(a)| \leq \max_a |f_1(a) - f_2(a)|$$

14

Nondeterministic Case

What if reward and next state are non-deterministic?

We redefine V, Q by taking expected values

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

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

15

Nondeterministic Case

Q learning generalizes to nondeterministic worlds

Alter training rule to

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

where

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

Can still prove convergence of \hat{Q} to Q [Watkins and Dayan, 1992]

16

Temporal Difference Learning

Q learning: reduce discrepancy between successive Q estimates

One step time difference:

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

Why not two steps?

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

Or n ?

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

Blend all of these:

$$Q^\lambda(s_t, a_t) \equiv (1 - \lambda) \left[Q^{(1)}(s_t, a_t) + \lambda Q^{(2)}(s_t, a_t) + \lambda^2 Q^{(3)}(s_t, a_t) + \dots \right]$$

17

Temporal Difference Learning

$$Q^\lambda(s_t, a_t) \equiv (1 - \lambda) \left[Q^{(1)}(s_t, a_t) + \lambda Q^{(2)}(s_t, a_t) + \lambda^2 Q^{(3)}(s_t, a_t) + \dots \right]$$

Equivalent expression:

$$Q^\lambda(s_t, a_t) = r_t + \gamma \left[(1 - \lambda) \max_a \hat{Q}(s_t, a_t) + \lambda Q^\lambda(s_{t+1}, a_{t+1}) \right]$$

TD(λ) algorithm uses above training rule

- Sometimes converges faster than Q learning
- converges for learning V^* for any $0 \leq \lambda \leq 1$ (Dayan, 1992)
- Tesauro's TD-Gammon uses this algorithm

18

RL Application Domains

Successful domains

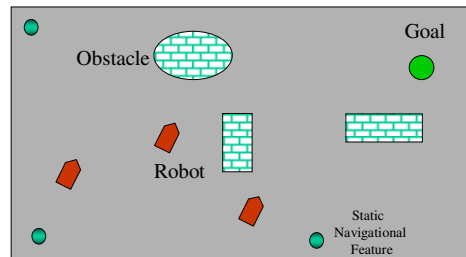
- Low dimensional discrete state space
- 1,000,000's learning runs (*simulation*)

Not so successful domains

- Large continuous state space
- 1,000,000's learning runs *not practical*

19

Continuous Domains? Robotics



- Hit an obstacle: get a **negative** reward
- Reach goal: get a **positive** reward
- Reach goal faster: get a **bigger positive** reward

20

Other Continuous Problems

- Process control
 - Chemical
 - Power
- Financial modeling
- Software agents on web
 - State space defined by hit statistics
- TCOM problems, etc

21

Why Is RL Hard in Large Continuous Domains?

- Stochastic search in large continuous domains is hard
- One possible solution: use prior domain knowledge to direct search

22

Reinforcement Learning (MDP)

- Policy: $\pi(s, a) = \Pr(a_t = a | s_t = s)$
- Reinforcement feedback (environment) r_t
- Goal: modify policy to maximize reward

$$\rho(\pi) = E \left\{ \sum_{t=1}^{\infty} \gamma^t r_t \mid s_0, \pi \right\}$$

- State-action value function

$$Q^\pi(s, a) = E \left\{ \sum_{t=1}^{\infty} \gamma^t r_t \mid s_t = s, a_t = a, \pi \right\}$$

23

Approaches to RL

- Value Function RL
- Policy Gradient RL
- Actor-Critic RL
 - combines value functions and policy gradients

24

Value Function RL

- Learn the value of executing each action in each state (*Value Function* $Q^\pi(s,a)$)
- In each state, execute the most valuable action
- Problem:
 - Value function learning infeasible in high dimensional state spaces

25

Policy Gradient RL

- Parameterize agent's policy (Θ)
- Estimate how the value of a policy (ρ) changes as Θ changes: $\partial\rho/\partial\Theta$
- Update policy (gradient ascent):
$$\Theta_{t+1} = \Theta_t + \alpha \frac{\partial\rho}{\partial\Theta}$$
- Problems:
 - Local minimum and slow convergence

26