

REINFORCEMENT LEARNING

CHAPTER 21, SECTIONS 1–4

Outline

- ◇ Examples
- ◇ Learning a value function for a fixed policy
 - temporal difference learning
- ◇ Q-learning
- ◇ Function approximation
- ◇ Exploration

Reinforcement learning

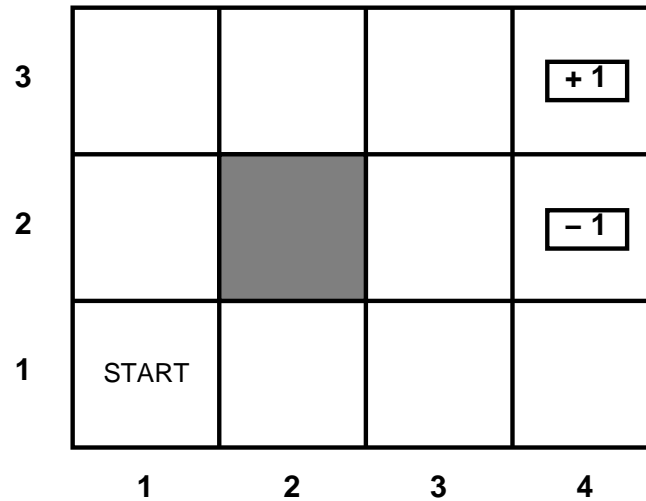
Agent is in an MDP or POMDP environment

Only feedback for learning is percept + reward

Agent must learn a policy in some form:

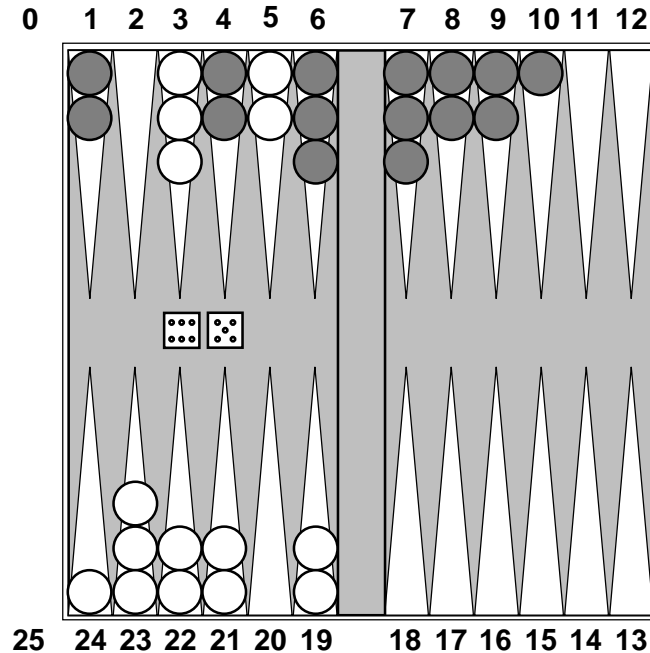
- transition model $T(s, a, s')$ plus value function $U(s)$
- $Q(a, s)$ = expected utility if we do a in s and then act optimally
- policy $\pi(s)$

Example: 4×3 world



$(1, 1) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} \dots (4, 3) +1$
 $(1, 1) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (2, 3) \xrightarrow{-0.04} (3, 3) \xrightarrow{-0.04} \dots (4, 3) +1$
 $(1, 1) \xrightarrow{-0.04} (2, 1) \xrightarrow{-0.04} (3, 1) \xrightarrow{-0.04} (3, 2) \xrightarrow{-0.04} (4, 2) -1 .$

Example: Backgammon



Reward for win/loss only in terminal states, otherwise zero

TDGammon learns $\hat{U}(s)$, represented as 3-layer neural network

Combined with depth 2 or 3 search, one of top three players in world

Example: Animal learning

RL studied experimentally for more than 60 years in psychology

Rewards: food, pain, hunger, recreational pharmaceuticals, etc.

Example: bees learn near-optimal foraging plan in field of artificial flowers with controlled nectar supplies

Bees have a direct neural connection from nectar intake measurement to motor planning area

Example: Autonomous helicopter

Reward = - squared deviation from desired state



Example: Autonomous helicopter



Temporal difference learning

Fix a policy π , execute it, learn $U^\pi(s)$

Bellman equation:

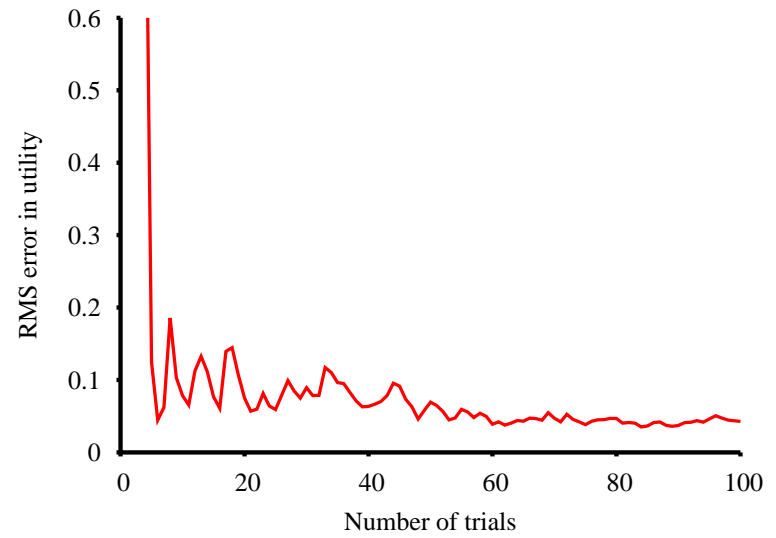
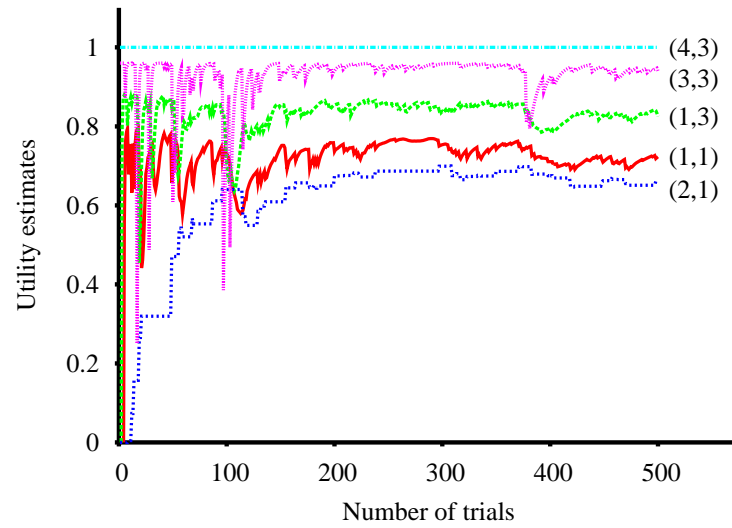
$$U^\pi(s) = R(s) + \gamma \sum_{s'} T(s, \pi(s), s') U^\pi(s')$$

TD update adjusts utility estimate to agree with Bellman equation:

$$U^\pi(s) \leftarrow U^\pi(s) + \alpha (R(s) + \gamma U^\pi(s') - U^\pi(s))$$

Essentially using sampling from the environment instead of exact summation

TD performance



Q-learning

One drawback of learning $U(s)$: still need $T(s, a, s')$ to make decisions

$Q(a, s)$ = expected utility if we do a in s and then act optimally

Bellman equation:

$$Q(a, s) = R(s) + \gamma \sum_{s'} T(s, \pi(s), s') \max_{a'} Q(a', s')$$

Q-learning update:

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

😊 Q-learning is a **model-free** method for learning and decision making

😞 Q-learning is a **model-free** method for learning and decision making
(so cannot use model to constrain Q-values, do mental simulation, etc.)

Function approximation

For real problems, cannot represent U or Q as a table!!

Typically use linear function approximation:

$$\hat{U}_\theta(s) = \theta_1 f_1(s) + \theta_2 f_2(s) + \cdots + \theta_n f_n(s) .$$

Use a gradient step to modify θ parameters:

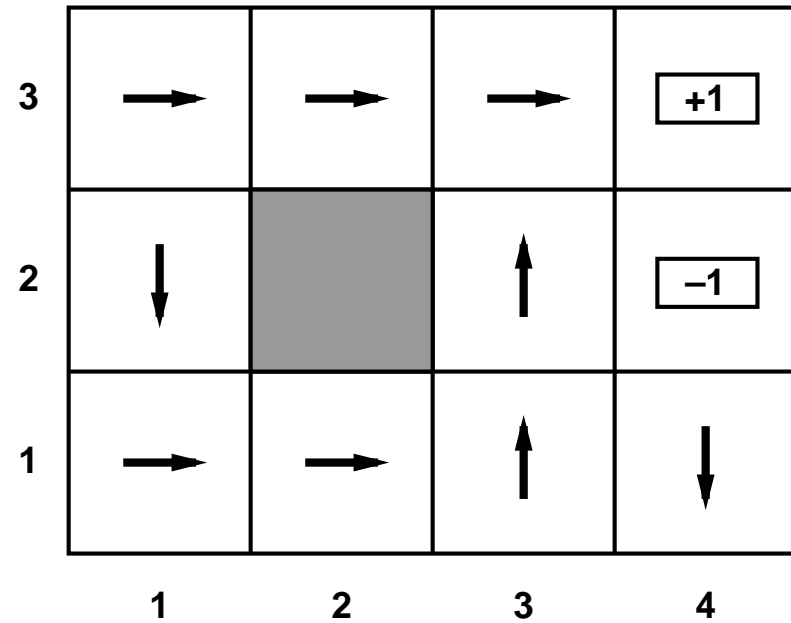
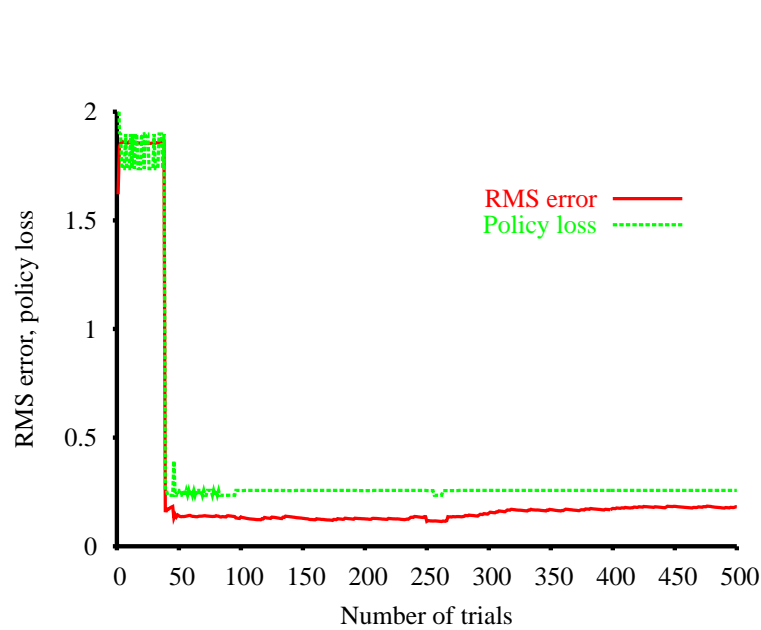
$$\theta_i \leftarrow \theta_i + \alpha [R(s) + \gamma \hat{U}_\theta(s') - \hat{U}_\theta(s)] \frac{\partial \hat{U}_\theta(s)}{\partial \theta_i}$$

$$\theta_i \leftarrow \theta_i + \alpha [R(s) + \gamma \max_{a'} \hat{Q}_\theta(a', s') - \hat{Q}_\theta(a, s)] \frac{\partial \hat{Q}_\theta(a, s)}{\partial \theta_i}$$

Often very effective in practice, but convergence not guaranteed

Exploration

How should the agent behave? Choose action with highest expected utility?



Exploration vs. exploitation: occasionally try “suboptimal” actions!!

Summary

Reinforcement learning methods find approximate solutions to MDPs

Work directly from experience in the environment

Need not be given transition model *a priori*

Q-learning is completely model-free

Function approximation (e.g., linear combination of features) helps RL scale up to very large MDPs

Exploration is required for convergence to optimal solutions