COMP 532
Machine Learning and BioInspired Optimization

Lecture 7 Reinforcement Learning

Dr. Shan Luo

Department of Computer Science

shan.luo@liverpool.ac.uk

Module Syllabus (approximate)

- Parallel Problem Solving from Nature (2)
- Single-Agent RL (6)
- Deep Learning (6)
- Multi-Agent RL (6)
- Swarm Intelligence (4)
- Artificial Immune Systems (3)
- DNA Computing (3)

Task 1 – Reinforcement Learning

- Counts 10% towards final mark
- Work in pairs of two
- Some programming, some pen & paper

- Deadline: 9 March 2018

Module Syllabus (approximate)

W1
* 29 Jan: lec 3-4pm, PPSN 1
* 30 Jan: lec 9-10am, PPSN 2
* 2 Feb: lec 2-3pm, RL 1
- 2 Feb: tutorial 4pm-5pm

no class (make pairs of two)

W2
* 5 Feb: lec 3-4pm, RL 2
* 6 Feb: lec 9-10am, RL 3
* 8 Feb: lec 2-3pm, RL 4
* 9 Feb: tutorial 4pm-5pm

W3
* 12 Feb: lec 3-4pm, RL 5
* 13 Feb: lec 9-10am, RL 6
* 16 Feb: lec 2-3pm, DEEP 1
* 16 Feb: tutorial 4pm-5pm, task 1

W4
* 19 Feb: lec 3-4pm, DEEP 2
* 20 Feb: lec 9-10am, DEEP 3
* 23 Feb: lec 2-3pm, DEEP 4
* 23 Feb: tutorial 4pm-5pm, task 1

Module Syllabus (approximate)

W5
* 26 Feb: lec 3-4pm, DEEP 5
* 27 Feb: lec 9-10am, DEEP 6
* 2 Mar: lec 2-3pm, MARL 1
* 2 Mar: tutorial 4pm-5pm, task 1

W6 (Task 1 Due)
* 5 Mar: lec 3-4pm, MARL 2
* 6 Mar: lec 9-10am, MARL 3
* 9 Mar: lec 2-3pm, MARL 4
* 9 Mar: tutorial 4pm-5pm, task 1

W7
* 12 Mar: lec 3-4pm, MARL 5
* 13 Mar: lec 9-10pm, MARL 6
* 16 Mar: lec 2-3pm, SI I
* 16 Mar: tutorial 4pm-5pm, make groups of 3

W8 (task 2 paper selection)
* 9 Apr: lec 4-5pm, SI 2
* 10 Apr: lec 12-1pm, SI 3
* 13 Apr: lec 2-3pm, SI 4
* 13 Apr: tutorial 4pm-5pm, task 2

Module Syllabus (approximate)

W9

* 16 Apr: lec 3-4pm, AIM 1
* 17 Apr: lec 9-10am, AIM 2
* 20 Apr: lec 2-3pm, DNA 1
* 20 Apr: tutorial 4pm-5pm task 2

W10 (TASK 2 due)

* 23 Apr: lec 3-4pm, DNA 2
* 24 Apr: lec 9-10am, WRAP UP
* 27 Apr: lec 2-3pm, STUDENT LEC 1
* 27 Apr: tutorial 4pm-5pm task 2

W11

* 30 Apr: lec 3-4pm, STUDENT LEC 2
* 1 May: lec 9-10am, STUDENT LEC 3
* 4 May: lec 2-3pm, STUDENT LEC 4
* 4 May: tutorial 4pm-5pm, STUDENT LEC 5

W12

* 7 May: Early May Bank Holiday
* 8 May: lec 9-10am, STUDENT LEC 6
* 11 May: lec 2-3pm, QUESTIONS?
* 11 May: tutorial 4pm-5pm QUESTIONS?

TD Prediction

Policy Evaluation (the prediction problem):

for a given policy π, compute the state-value function V^{π}

Recall: Simple every-visit Monte Carlo method:

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t - V(s_t)]$$

target: the actual return after time t

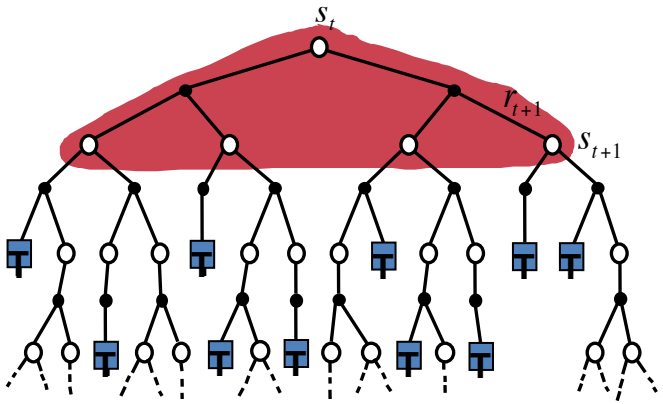
The simplest TD method, TD(0):

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

target: an estimate of the return

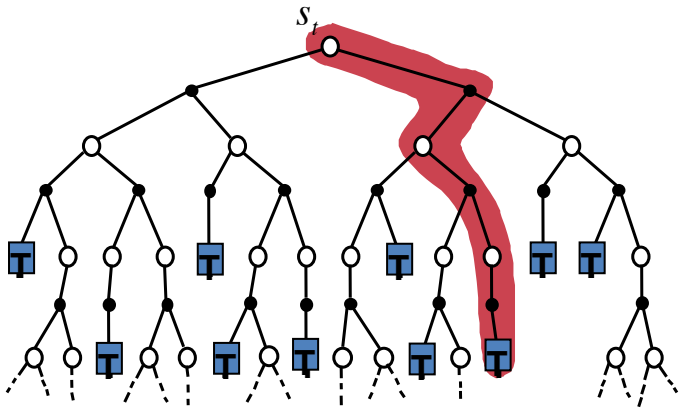
(TD error)

$$V(s_t) \leftarrow E_\pi\{r_{t+1} + \gamma V(s_{t+1})\}$$



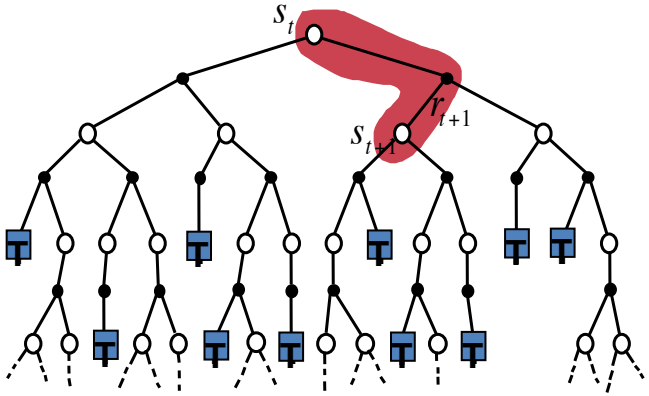
Dynamic Programming

$$V(s_t) \leftarrow V(s_t) + \alpha[\mathrm{G}_t - V(s_t)]$$



Monte Carlo

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$



Temporal Difference

Classification of RL methods

Model of the environment?

Bootstrap?

	YES	NO
YES	Dynamic programming	Temporal Difference (TD)
NO	_____	Monte Carlo Methods

TD methods bootstrap and sample

- Bootstrapping: update involves an *estimate*
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps

- Sampling: update does not involve an *expected value*
  - MC samples
  - DP does not sample
  - TD samples

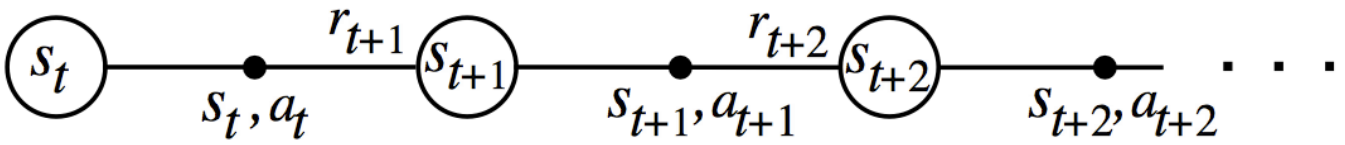
Advantages of TD Learning

- TD methods do not require a model of the environment, only experience

- TD, but not MC, methods can be fully incremental
  - You can learn before knowing the final outcome
    - Less memory
    - Less peak computation
  - You can learn without the final outcome
    - From incomplete sequences

Learning an Action-Value Function

Estimate Q^{π} for the current behaviour policy π .



After every transition from a non-terminal state s_t, do this:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

If s_{t+1} is terminal, then $Q(s_{t+1}, a_{t+1}) = 0$.

Sarsa: On-Policy TD Control

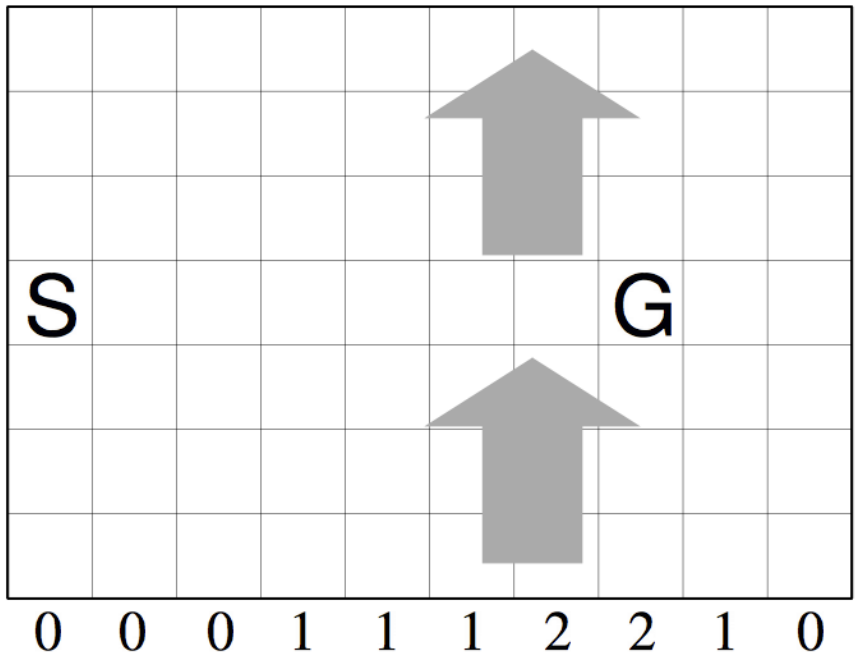
Turn this into a control method by always updating the policy to be greedy with respect to the current estimate:

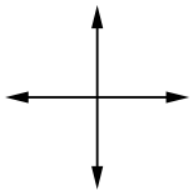
Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Choose a from s using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$; $a \leftarrow a'$;
  until s is terminal

Note: $s_t = s$
$s_{t+1} = s'$

SARSA: State Action Reward State Action

Windy Gridworld

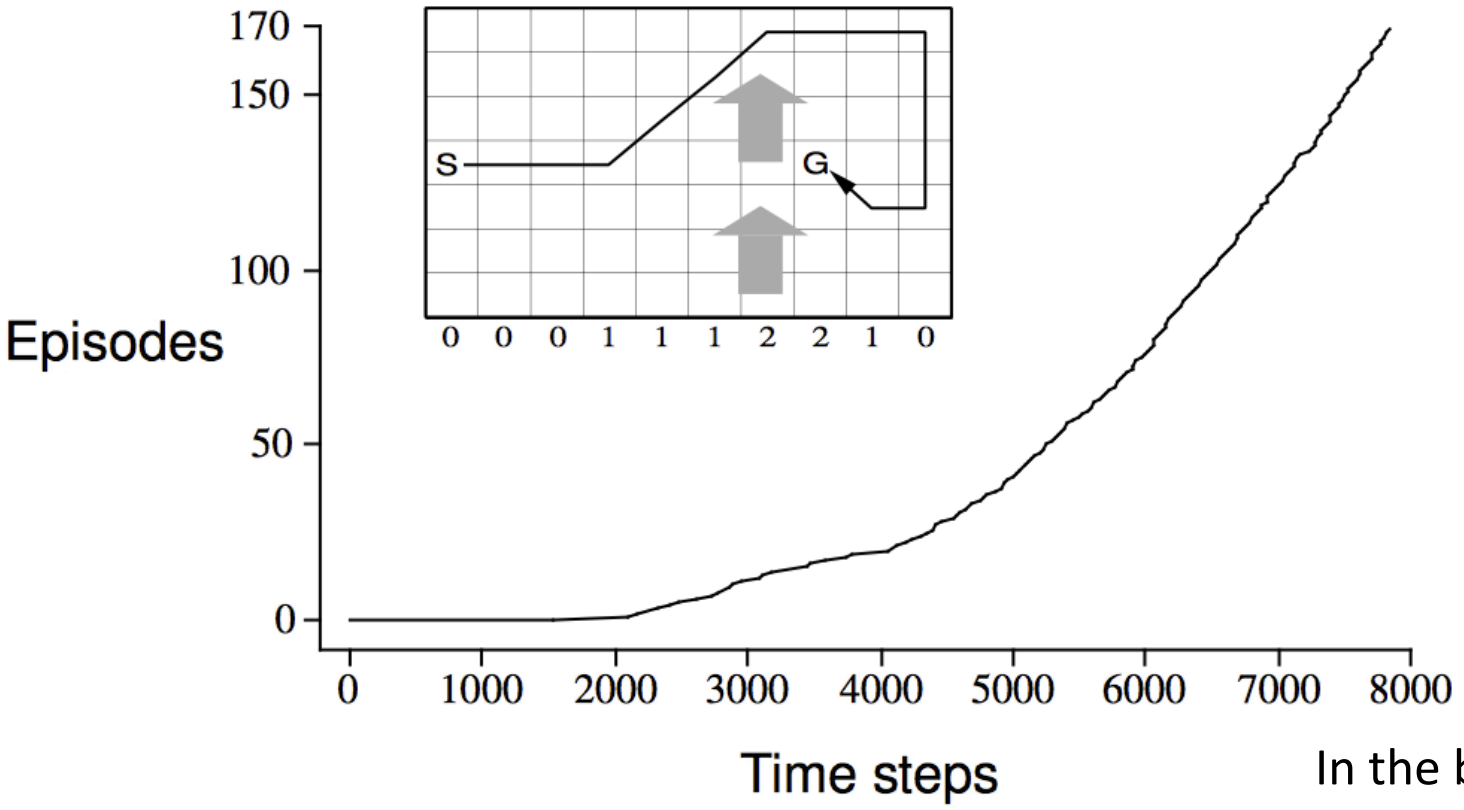




standard moves

undiscounted, episodic, reward = –1 until goal

Sarsa on the Windy Gridworld



In the book p.106

$\alpha = 0.5, \varepsilon = 0.1$

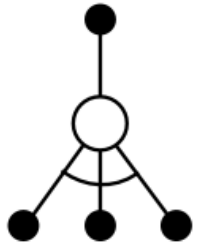
Windy Gridworld

- Would DP find the same policy?
  - Under which assumptions?
  - What is the advantage of Sarsa in this case?
- What about MC?

Q-Learning: Off-Policy TD Control

One-step Q-learning:

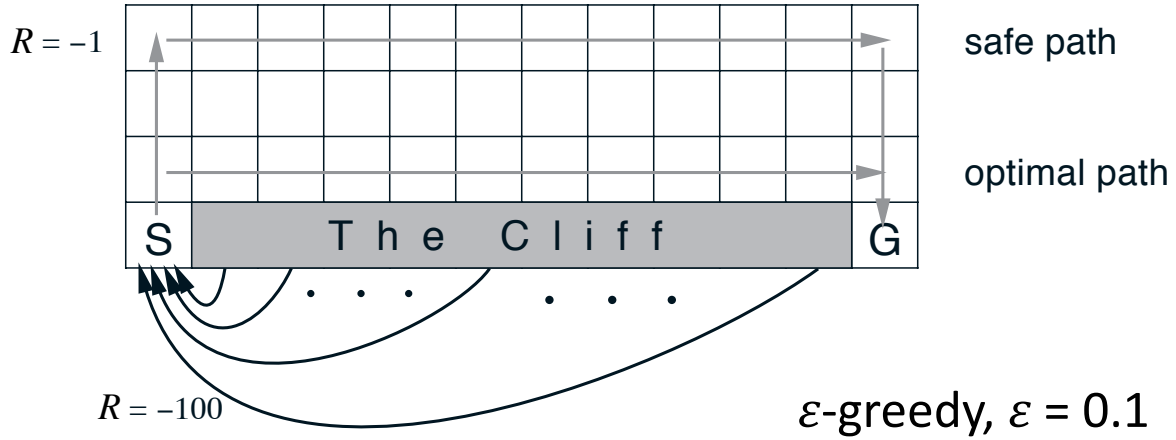
$$\mathrm{Q}(s_t, a_t) \leftarrow \mathrm{Q}(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a \mathrm{Q}(s_{t+1}, a) - \mathrm{Q}(s_t, a_t) \right]$$



Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$;
  until s is terminal

Why off-policy?

Cliffwalking

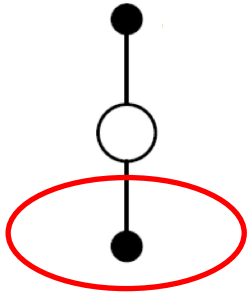


ε-greedy, ε = 0.1

In the book p.108

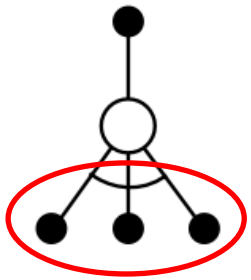
On-Policy vs. Off-Policy

Sarsa:



$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Q-learning:



$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

Convergence to Optimality?

Sarsa: difficult, but..

- basic Sarsa converges given some assumptions
  - infinite visits to all state-action pairs
  - policy approaches greedy policy in the limit

Q-learning: easier, because off-policy

- converges given some assumptions
  - infinite visits..
  - decreasing step-size α (but not too fast)

Even without these assumptions, but algorithms have shown good performance in many tasks!

Reinforcement Comparison

- If you do better than expected, reinforce..
  - How to define “better than expected”?

- Reinforcement Comparison:
  - Keep track of average reward ρ
  - “Better than expected” if $r_t > \rho$

- Using average reward per timestep instead of return: no need for discounting!

Average Reward Per Time Step

Average expected reward per time step under policy π:

$$\rho^{\pi} = \lim_{n \to \infty} \frac{1}{n} \sum_{t=1}^{n} E_{\pi}\{r_t\}$$

the same for each state if ergodic

Instead of the usual (discounted) return, use the differential:

$$G_t = r_{t+1} - \rho^{\pi} + r_{t+2} - \rho^{\pi} + r_{t+3} - \rho^{\pi} + \cdots$$

TD error is defined as:

$$\delta_t = r_{t+1} - \rho^{\pi} + V(s') - V(s)$$

R-learning

Initialize ρ and $Q(s,a)$, for all s, a, arbitrarily
Repeat forever:
$s \leftarrow$ current state
Choose action a in s using behavior policy (e.g., ε-greedy)
Take action a, observe r, s'
$Q(s,a) \leftarrow Q(s,a) + \alpha\,[r - \rho + \max_{a'} Q(s',a') - Q(s,a)]$
If $Q(s,a) = \max_a Q(s,a)$, then:
$\rho \leftarrow \rho + \beta\,[r - \rho + \max_{a'} Q(s',a') - \max_a Q(s,a)]$

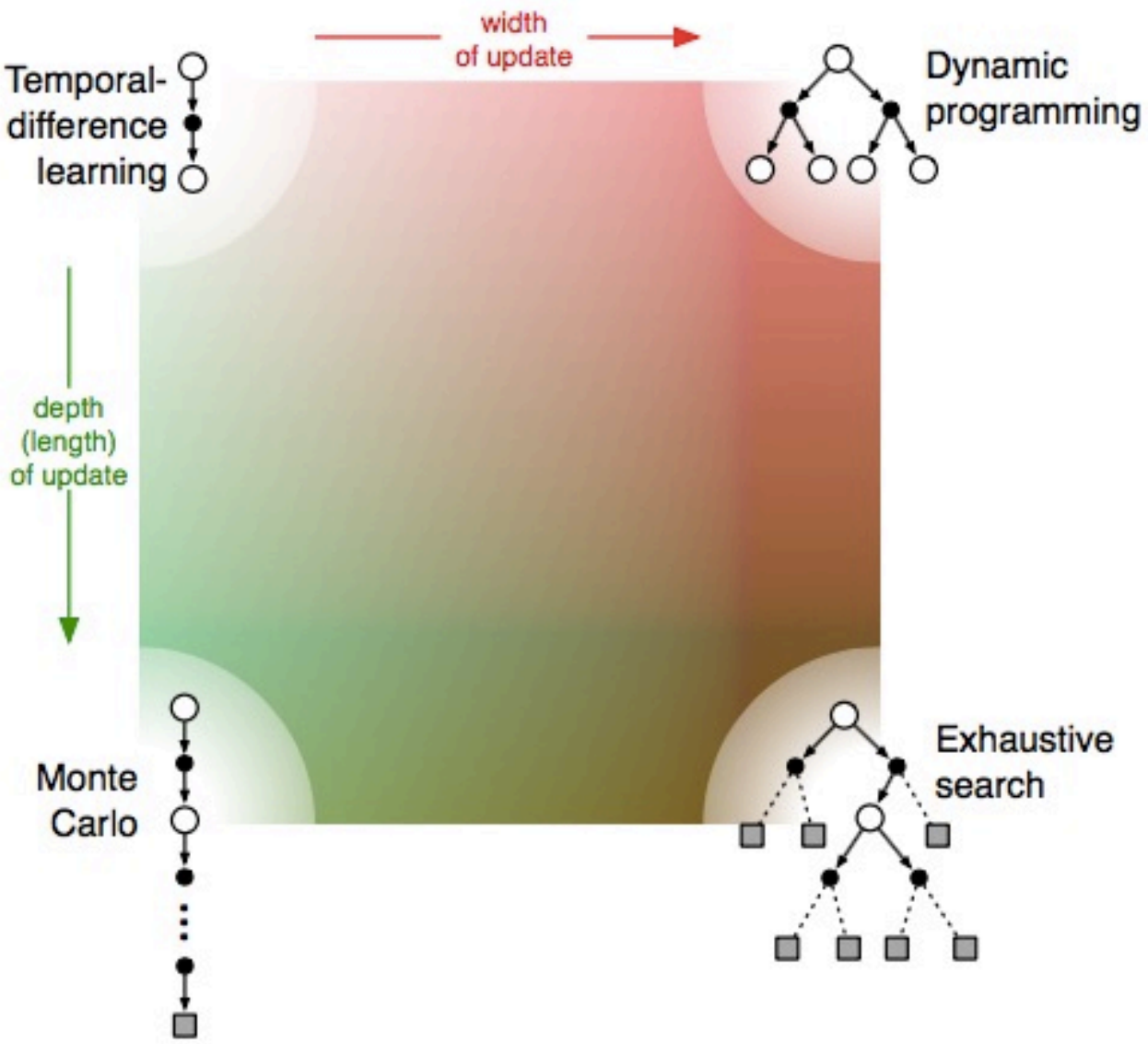
Summary

- TD prediction
- Introduced *one-step tabular model-free TD methods*
- Extend prediction to control by employing some form of generalized policy iteration
  - On-policy control: Sarsa
  - Off-policy control: Q-learning and R-learning
- These methods bootstrap and sample, combining aspects of DP and MC methods

The Book

- Introduction
- Part I: Tabular Solution Methods
  - Multi-armed Bandits
  - Finite Markov Decision Processes
  - Dynamic Programming
  - Monte Carlo Methods
  - Temporal Difference Learning
- Part II: Approximate Solution Methods
- Part III: Looking Deeper

Unified View



Questions

- What can *I* tell you about RL?
- What is common to all three classes of methods? – DP, MC, TD
- What are the principle strengths and weaknesses of each?
- In what sense is our RL view complete?
- In what senses is it incomplete?
  - What are the principal things missing?
- The broad applicability of these ideas…
- What does the term bootstrapping refer to?
- What is the relationship between DP and learning?