Reinforcement Learning:

Lessons for AI

Richard S. Sutton

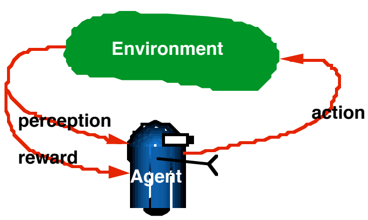
University of Massachusetts

www.cs.umass.edu/~rich

Outline

- Definitions of RL
- History of RL
- State of the Art
- Lessons
- RL at the Knowledge Level

RL is Learning from Interaction



RL is like life!

- complete agent
- temporally situated
- continual learning & planning
- object is to affect environment
- env stochastic & uncertain

More Formally: Markov Decision Problems (MDPs)

An MDP is defined by $\langle S, A, P, R, \gamma \rangle$

S – set of states of the environment

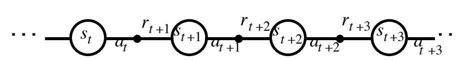
$A(s)$ – set of actions possible in state $s \in S$

$P(s,s',a)$ – probability of transition from s to s' given action a

$R(s,s',a)$ – expected reward on transition s to s' given a

γ – discount rate for delayed reward

discrete time, $t = 0, 1, 2, \ldots$



The Objective

- Find a way of behaving that gets a lot of reward in the long run
- Find a policy $\pi : s \in S \rightarrow a \in A(s)$ (could be stochastic)

that maximizes the value (expected future reward) of each s :

$$V^{\pi}(s) = E\left\{ r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \,\middle|\, s_t = s, \pi \right\}$$

and each s,a pair:

rewards

$$Q^{\pi}(s,a) = E\left\{ r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \,\middle|\, s_t = s, a_t = a, \pi \right\}$$

These are called value functions - cf. evaluation functions

Radical Generality of the RL problem

RL	Classical
General stochastic dynamics	Determinism
General goals	Goals of achievement
Uncertainty	Complete knowledge
Reactive decision-making	Closed world
	Unlimited deliberation

Definitions of Reinforcement Learning

Learning about, from, and while interacting with an external environment to achieve a goal.

Learning what to do by trial and error.

Any way of solving the MDP problem

i.e., of finding a policy that maximizes long-term reward

An approach to AI emphasizing the above.

AI's Founding Fathers would find RL Familiar

Minsky's PhD thesis is earliest work in RL (1954)

SNARC - "Stochastic Neural-Analog Reinforcement" Calculator

Samuel (1959) learned evaluation functions using a temporal-difference method

State/action/reward ideas commonplace in early AI

and in animal learning psychology,

and in optimal control theory

In the 1970s, RL Died Out

- Learning fell out of fashion in AI
- Learning fell out of fashion in Psychology

- RL was confused with supervised learning, pattern recognition
- Little work on genuine trial-and-error learning

Exceptions: Michie, Andreae, Learning Automata, Klopf

Rebirth of RL in the 1980s

- Realization that trial-and-error learning had been lost – Klopf (1972), Barto & Sutton (1981)
- TD(λ) – Sutton (1988)
- Q-learning and connections to dynamic programming – Watkins (1989)
- TD-Gammon – Tesauro (1992)

Modern RL

Very active area, centered in

- Machine learning
- Neural Networks
- Operations Research
- MDP planning in AI

Also spin-offs in Psychology and Neuroscience

Not a separate field

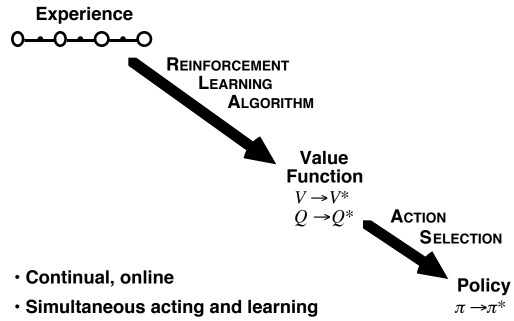
Strands of History of RL

Trial-and-error learning	Temporal-difference learning	Optimal control, value functions
Thordike (Ψ) 1911	Secondary reinforcement (Ψ)	Hamilton (Physics) 1800s
		Shannon
Minsky	Samuel	Bellman/Howard (OR)
Klopf		
	Witten	Werbos
Barto et al		
	Sutton	
		Watkins

Outline

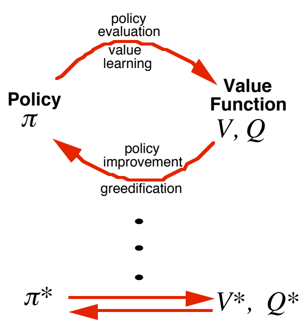
- Definitions of RL
- History of RL
- State of the Art
- Lessons
- RL at the Knowledge Level

What RL Algorithms Do

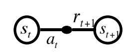


- Continual, online
- Simultaneous acting and learning

All RL Algorithms work by Interaction of Policy and Value



1-Step Tabular Q-Learning

On each state transition: 

Update:

$$\underbrace{Q(s_t,a_t)}_{\text{a table entry}} \leftarrow Q(s_t,a_t) + \alpha \underbrace{\left[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t) \right]}_{\text{TD Error}}$$

$$\lim_{t\to\infty} Q(s,a) \to Q^*(s,a)$$

$$\lim_{t\to\infty} \pi_t \to \pi^*$$

Optimal behavior found without a model of the environment!

(Watkins, 1989)

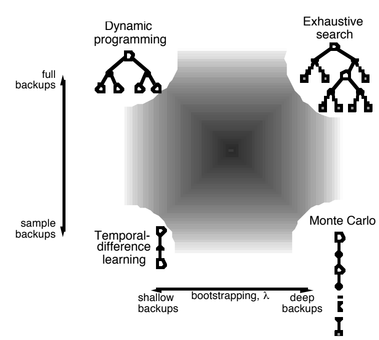
Assumes finite MDP

Dimensions of RL algorithms

- Function Approximation
- On-line Experience or Simulated Experience
- Amount of Search in Action Selection
- Exploration Method
- Kind of Backups

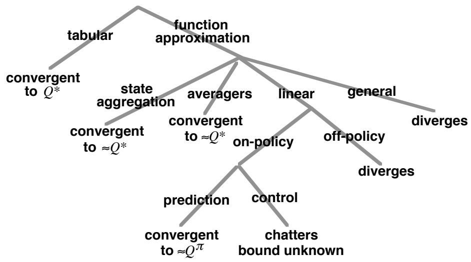
⋮

Kinds of Backups



Summary of Convergence theory

Asymptotic results only – almost no rate results



World-Class Applications of RL

- **TD-Gammon and Jellyfish** – Tesauro, Dahl
  World's best backgammon player
- **Elevator Control** – Crites & Barto
  World's best down-peak elevator controller
- **Inventory Management** – Van Roy, Bertsekas, Lee & Tsitsiklis
  10-15% improvement over industry standard methods
- **Dynamic Channel Assignment** – Singh & Bertsekas, Nie & Haykin
  World's best assigner of radio channels to mobile telephone calls

All these applications are large, stochastic, optimal control problems

- too hard for conventional methods to solve exactly
- require problem to be simplified

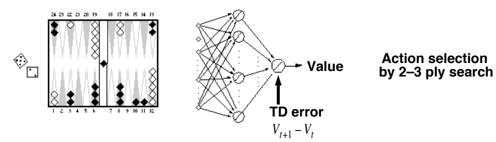
RL just finds an approximate solution

. . . which can be much better!

Lesson #1: Approximate the Solution, Not the Problem

Tesauro, 1992–1995

TD-Gammon



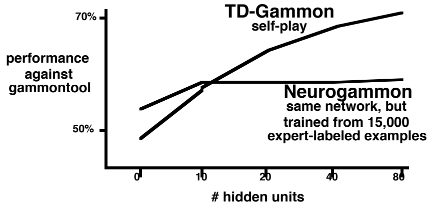
Start with a random network

Play millions of games against self

Learn a value function from this simulated experience

This produces arguably the best player in the world

Lesson #2: The Power of Learning from Experience



Tesauro, 1992

Expert examples are expensive and scarce

Experience is cheap and plentiful!

And teaches the real solution

Lesson #3: The Central Role of Value Functions

...of modifiable moment-by-moment estimates of how well things are going

All RL methods learn value functions

All state-space planners compute value functions

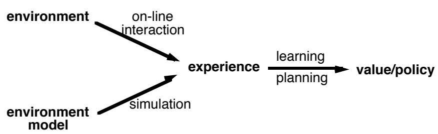
Both are based on "backing up" value

Recognizing and reacting to the ups and downs of life is an important part of intelligence

Lesson #4: Learning and Planning can be radically similar

Historically, planning and trial-and-error learning have been seen as opposites

But RL treats both as processing of experience



1-Step Tabular Q-Planning

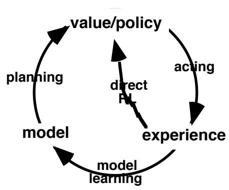
1. Generate a state, s, and action, a
2. Consult model for next state, s', and reward, r
3. Learn from the experience, s,a,r,s' :

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

4. go to 1

With function approximation and cleverness in search control (Step 1), this is a viable, perhaps even superior, approach to state-space planning

Lesson #5: Accretive Computation "Solves" Planning Dilemmas



Processes only loosely coupled

Can proceed in parallel, asynchronously

Quality of solution accumulates in value & model memories

Reactivity/Deliberation dilemma
"solved" simply by not opposing search and memory

Intractability of planning
"solved" by anytime improvement of solution

RL at the Knowledge Level

RL has a form of planning/reasoning
but it seems too low-level, too flat

RL learns values and models
but needs to re-learn values when goal changes

How can we learn re-usable components?
How can we learn to do A, B, and C
and then recombine them in new ways?

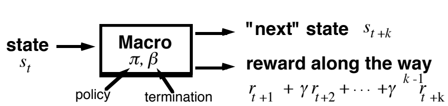
Need Actions at a Higher Level

e.g., `open-the-door` rather than `twitch-muscle-17`
`walk-to-work` rather than `1-step-forward`

Macro Actions

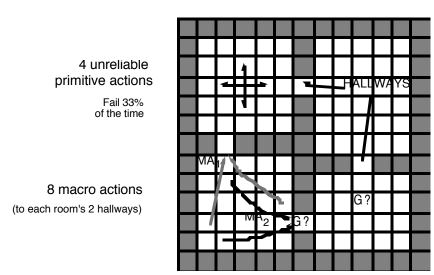
- take variable number of time steps
- specified by a whole (sub)policy
- and by a termination condition

cf. macro-operators



Can be treated much like primitive actions

Rooms Example



4 Learning Problems for Macro Actions

Selection among macro actions

Treat them as regular actions, learn their values

Learn models of macro actions

predict outcome of executing macro action

Subgoal Credit Assignment

Learn the policy inside a macro action

e.g., reward the way you did something, while punishing the decision to do it

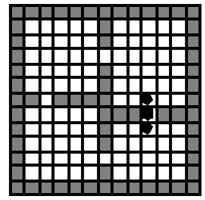
Discovery of suitable macro actions

Which subgoals? Utility issues

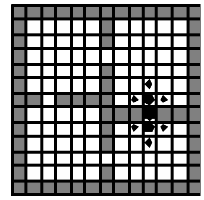
Value Iteration in Rooms Example

$V_0(s) = 0 \;\; \forall s \in S \qquad V_0(\textbf{goal}) = 1$

$$V_{k+1}(s) = \max_a \sum_{s'} P(s,s',a)\left[R(s,s',a) + \gamma V_k(s')\right] \;\; \forall s \in S$$

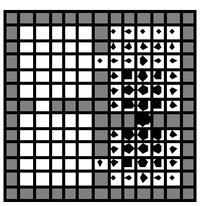


Iteration #1

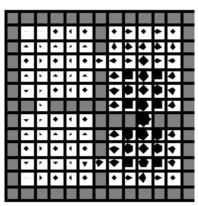


Iteration #2

Value Iteration with Models of Macro Actions



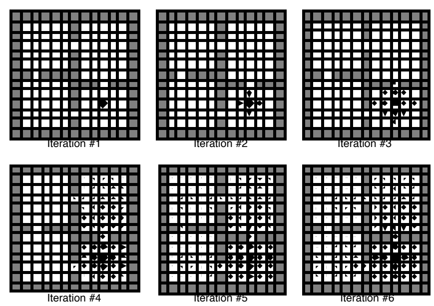
Iteration #1



Iteration #2

Example with Goal≠Subgoal

using models of both primitive and macro actions



Models of Macro Actions

- **Produce guaranteed correct plans**
- **Can be learned by TD methods**
- **Can be based on subgoals**
  - policies to achieve subgoals can be learned
  - user can provide subgoals
  - agent can propose own subgoals
- **Enable learning at the Knowledge level**
  - re-usable knowledge with a clear semantics
  - for a general context: stochastic, closed-loop, reward goals

Lesson #6: Generality is no impediment to working with higher-level knowledge

Summary of Lessons

1. Approximate the Solution, Not the Problem
2. The Power of Learning from Experience
3. The Central Role of Value Functions in finding optimal sequential behavior
4. Learning and Planning can be Radically Similar
5. Accretive Computation "Solves Dilemmas"
6. A General Approach need not be Flat, Low-level