Reinforcement Learning: A Tutorial

Satinder Singh

Computer Science & Engineering
University of Michigan, Ann Arbor

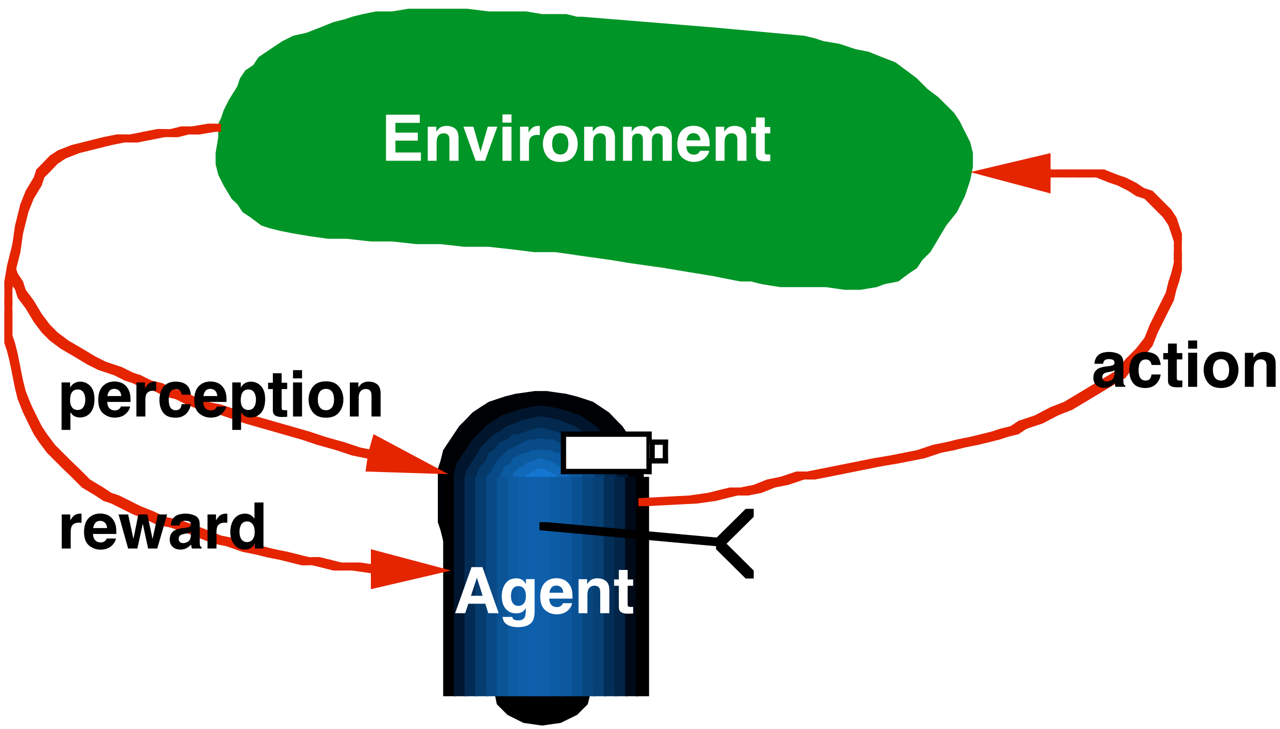
with special thanks to
Rich Sutton, Michael Kearns, Andy Barto, Michael Littman, Doina Precup, Peter Stone, Andrew Ng,...

http://www.eecs.umich.edu/~baveja/NIPS05Tutorial/

Outline

- History and Place of RL
- Markov Decision Processes (MDPs)
    - Planning in MDPs
    - Learning in MDPs
    - Function Approximation and RL
- Partially Observable MDPs (POMDPs)
- Beyond MDP/POMDPs
- Applications

RL is Learning from Interaction



RL is like Life!

- complete agent
- temporally situated
- continual learning and planning
- object is to affect environment
- environment is stochastic and uncertai

RL (another view)

$$o_1\, a_1\, r_2\;\; o_2\, a_2\, r_2\;\; \cdots\;\; \underline{o_i\, a_i\, r_{i+1}\, o_{i+1}}\;\; \cdots$$

Agent's life

Unit of experience

Agent chooses actions so as to maximize expected cumulative reward over a time horizon

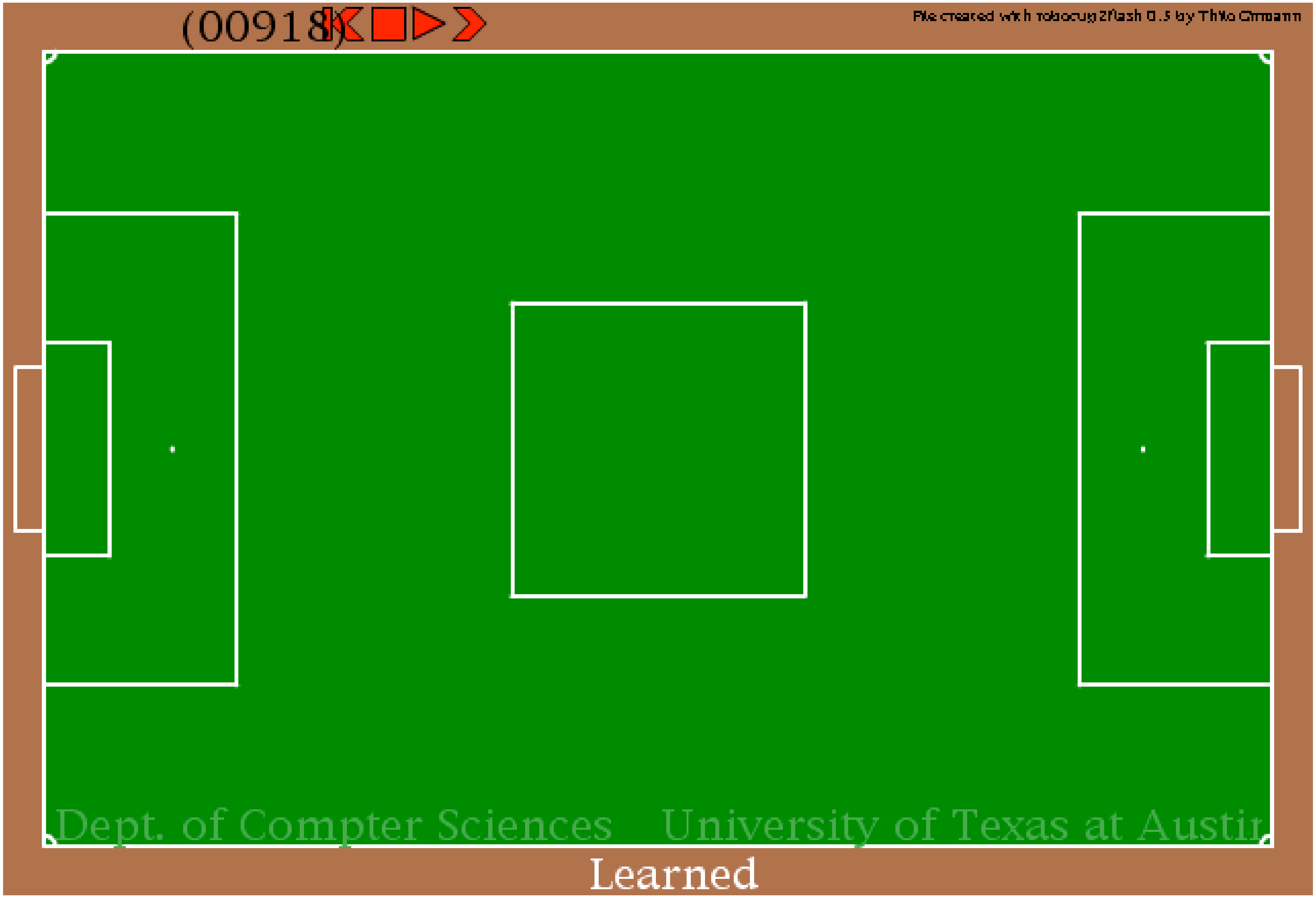
Observations can be vectors or other structures
Actions can be multi-dimensional
Rewards are scalar but can be arbitrarily uninformati

Agent has partial knowledge about its environment

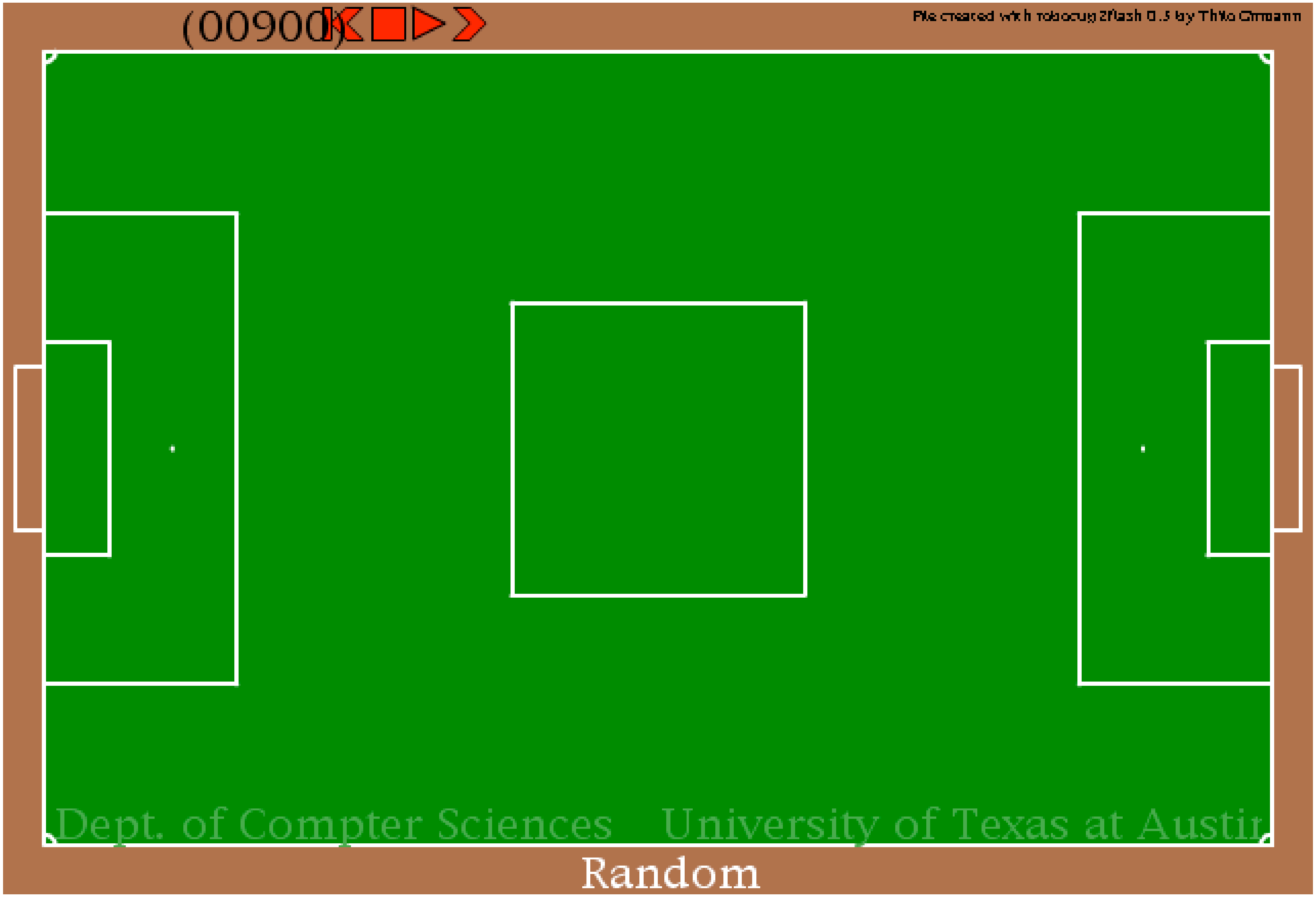
Key Ideas in RL

- Temporal Differences (or updating a guess on the basis of another guess)
- Eligibility traces
- Off-policy learning
- Function approximation for RL
- Hierarchical RL (options)
- Going beyond MDPs/POMDPs towards AI

Demos...



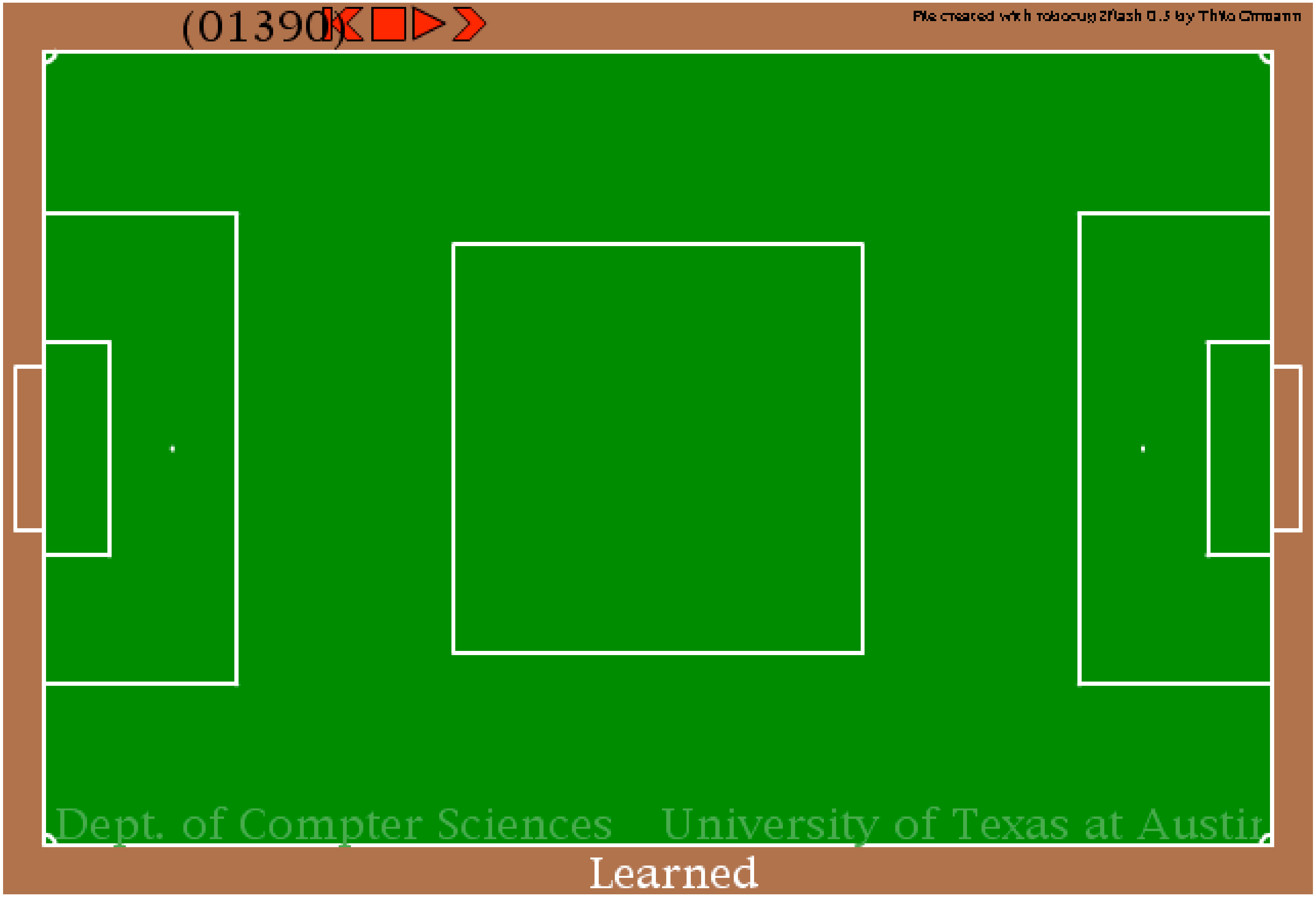
Stone & Sutton



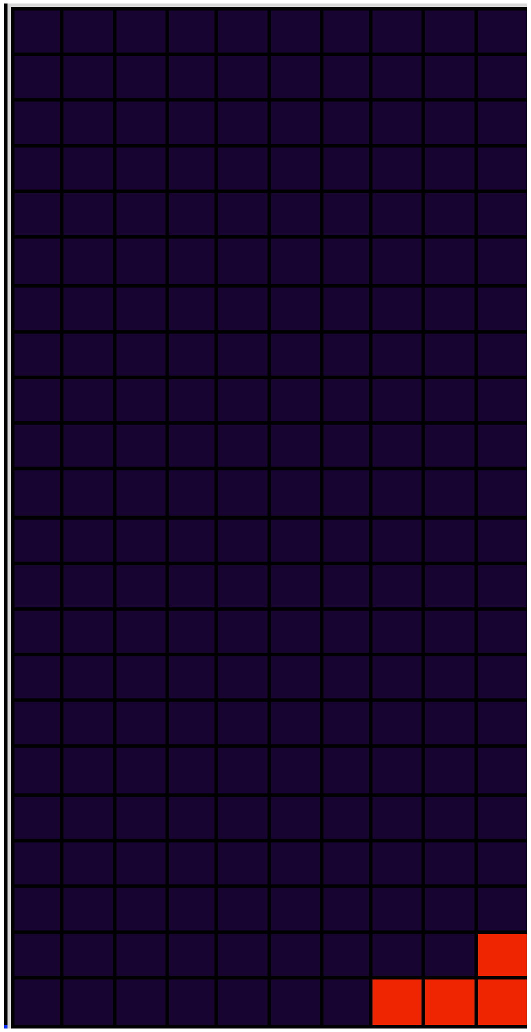
Stone & Sutton

Keepaway Soccer (Stone & Sutton)

- 4 vs 3 keepaway
  - Learned could keep the ball for ***10.2 seconds***
  - Random could keep the ball for ***6.3 seconds***
- 5 vs 4 keepaway
  - Learned could keep the ball for ***12.3 seconds***
  - Random could keep the ball for ***8.3 seconds***



Stone & Sutton



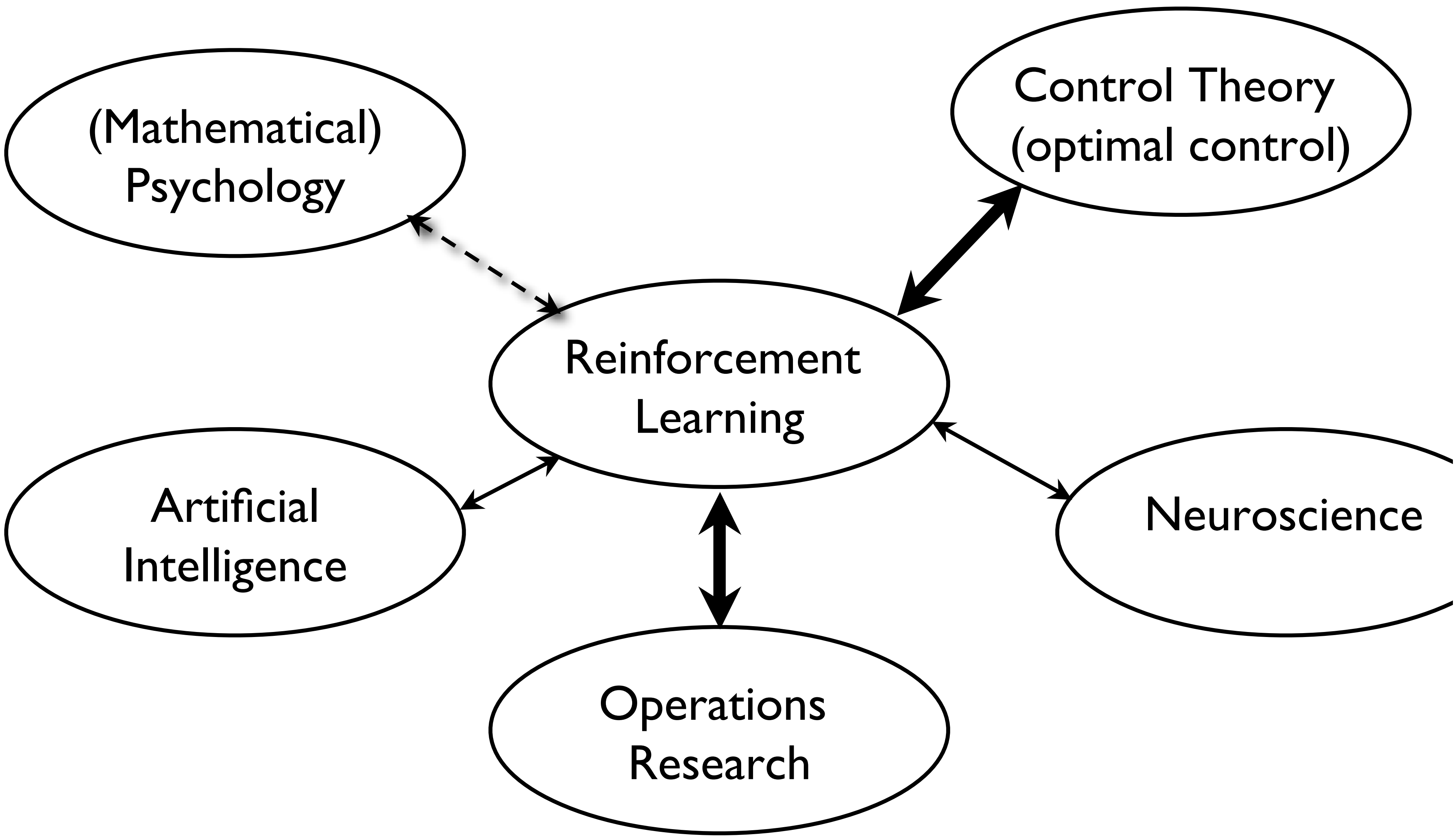
Tetris Demo

Learned

by
J Bagnell &
J Schneider

History & Place
(of RL)

Place


(Mathematical) Psychology
Control Theory (optimal control)
Reinforcement Learning
Artificial Intelligence
Neuroscience
Operations Research

(Partial) History

- *"Of several responses made to the same situation, those whic are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connecte with the situation, so that, when it recurs, they will be more likely to recur; those which are accompanied or closely followe by discomfort to the animal will, other things being equal, hav their connections with that situation weakened, so that, when recurs, they will be less likely to occur. The great the satisfaction or discomfort, the greater the strengthening or weakening of the bond."*
- (Thorndike, 1911, p. 244)
- **Law of Effect**

(Partial) History...

Idea of programming a computer to learn by *trial and error (*Turing, 1954)

SNARCs (Stochastic Neural-Analog Reinforcement Calculators) (Minsky,

Checkers playing program (Samuel, 59)

Lots of RL in the 60s (e.g., Waltz & Fu 65; Mendel 66; Fu 70)

MENACE (Matchbox Educable Naughts and Crosses Engine (Mitchie, 63)

RL based Tic Tac Toe learner (GLEE) (Mitchie 68)

Classifier Systems (Holland, 75)

Adaptive Critics (Barto & Sutton, 81)

Temporal Differences (Sutton, 88)

RL and Machine Learning

1. Supervised Learning (error correction)
   - learning approaches to *regression & classification*
   - learning from examples, learning from a teacher
2. Unsupervised Learning
   - learning approaches to *dimensionality reduction, density estimation, recoding data based on some principle, etc.*
3. Reinforcement Learning
   - learning approaches to *sequential decision making*
   - learning from a critic, learning from delayed reward

(Partial) List of Applications

- Robotics
  - Navigation, Robosoccer, walking, juggling, ...
- Control
  - factory processes, admission control in telecomm, resource control in multimedia networks, helicopters, elevators, ...
- Games
  - Backgammon, Chess, Othello, Tetris, ...
- Operations Research
  - Warehousing, transportation, scheduling, ...
- Others
  - HCI, Adaptive treatment design, biological modeling, ...

List of Conferences and Journals

- Conferences
  - Neural Information Processing Systems (NIPS)
  - International Conference on Machine Learning (ICML)
  - AAAI, IJCAI, Agents,COLT,...
- Journals
  - Journal of Artificial Intelligence Research (JAIR) [*free online*]
  - Journal of Machine Learning Research (JMLR) [*free online*]
  - Neural Computation, Neural Networks
  - Machine Learning, AI journal, ...

Model of Agent-Environment Interaction

$$o_1\, a_1\, r_2 \;\; o_2\, a_2\, r_2 \;\; \cdots \;\; o_i\, a_i\, r_{i+1} \;\; \cdots$$

Transition probabilities: $Pr(o_{t+1}|o_t, a_t, o_{t-1}, a_{t-1}, \ldots, o_1, a_1)$

Reward probabilities: $Pr(r_{t+1}|o_t, a_t, o_{t-1}, a_{t-1}, \ldots, o_1, a_1)$

Markov Decision Processe (MDPs)

Markov Assumption

$$Pr(o_{t+1}|o_t, a_t, o_{t-1}, a_{t-1}, \ldots, o_1, a_1) = Pr(o_{t+1}|o_t, a_t)$$
$$Pr(r_{t+1}|o_t, a_t, o_{t-1}, a_{t-1}, \ldots, o_1, a_1) = Pr(r_{t+1}|o_t, a_t)$$

Transition Probabilities: $P^a_{ss'} = Pr(s_{t+1} = s'|s_t = s, a_t = a)$

Payoff Function: $R^a_{ss'} = E\{r_{t+1}|s_{t+1} = s', s_t = s, a_t = a\}$

MDP Preliminaries

- S: finite state space
  A: finite action space
  P: transition probabilities $P(i|j,a)$ [or $P^a(ij)$]
  R: payoff function $R(i)$ or $R(i,a)$
  π: deterministic non-stationary policy S -> A
  $V^{\pi}(i)$:return for policy when started in state

$$V^{\pi}(i) = E_{\pi}\{r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots | s_0 = i\}$$

Discounted framework $(0 \leq \gamma < 1)$

Also, average framework: $V^{\pi} = \mathrm{Lim}_{T \to \infty} E_{\pi} 1/T \{r_0 + r_1 + \ldots +$

MDP Preliminaries...

π^* : optimal policy; $\pi^* = argmax_\pi V^\pi$

V^* : optimal value function $V^*(i) = \max_\pi V^\pi(i)$

- In MDPs there always exists a deterministic stationary policy (that simultaneously maximizes the value of every state)

$$V^\pi : S \rightarrow \Re \qquad\qquad V^* : S \rightarrow \Re$$

Bellman Optimality Equations

Policy Evaluation (Prediction)

$$V^{\pi}(i) = E_{\pi}\{r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots | s_0 = i\}$$

Markov assumption!

$$\boxed{\forall s \in S,\ V^{\pi}(s) = R(s,\pi(s)) + \sum_{s' \in S} P(s'|s,\pi(s)) V^{\pi}(s')}$$

$$Q^{\pi}(s,a) = E_{\pi}\{r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots | s_0 = s, a_0 = a$$

$$\forall s \in S, a \in A,\ Q^{\pi}(s,a) = R(s,a) + \sum_{s' \in S} P(s'|s,\pi(s)) Q^{\pi}(s',\pi($$

Bellman Optimality Equations

Optimal Control

$$\boxed{\forall s \in S,\ V^*(s) = \max_{a \in A}[R(s,a) + \sum_{s' \in S} P(s'|s,a)V^*(s')]}$$

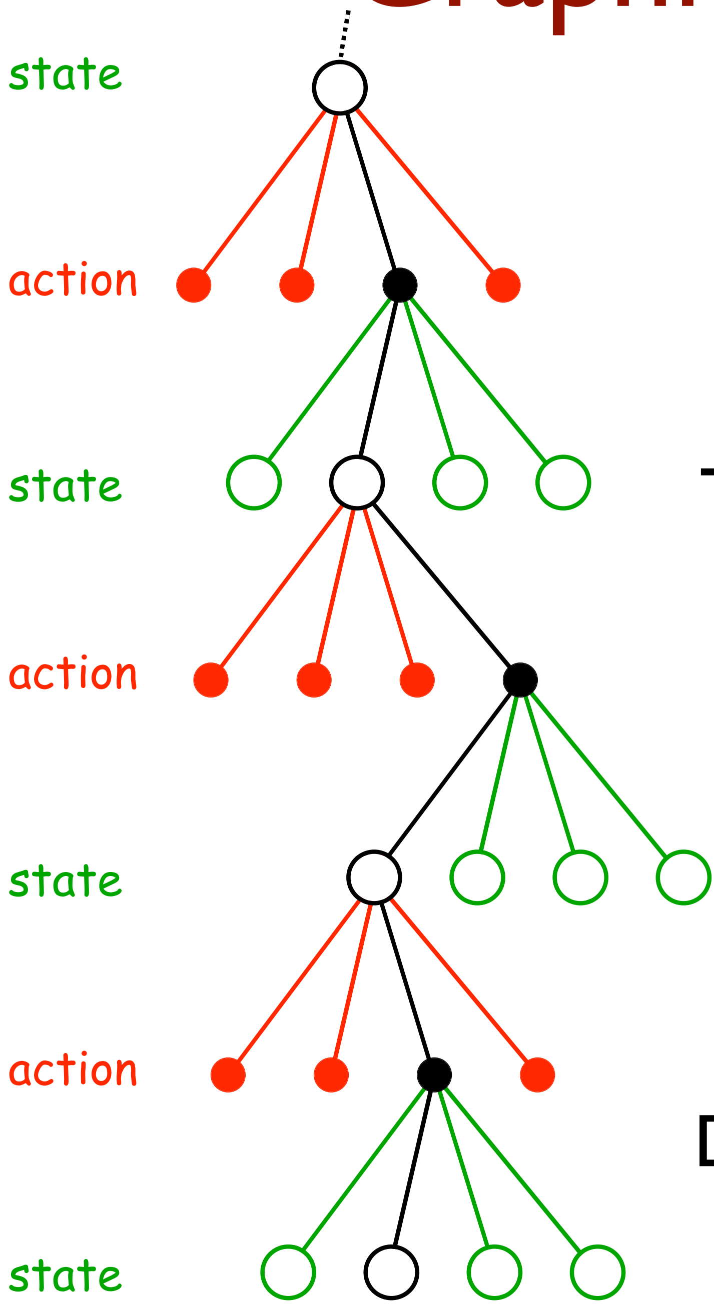
$$\forall s \in S, \pi^*(s) = argmax_{a \in A}[R(s,a) + \sum_{s' \in S} P(s'|s,a)V^*(s')]$$

$$\boxed{\forall s \in S, a \in A,\ Q^*(s,a) = R(s,a) + \sum_{s' \in S} P(s'|s,a) \max_{b \in A} Q^*(s',b)}$$

$$\forall s \in S, \pi^*(s) = argmax_{a \in A} Q^*(s,a)$$

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

Graphical View of MDPs



Temporal Credit Assignment Proble

Learning from *Delayed Reward*

Distinguishes RL from other forms of M

Planning & Learning

in

MDPs

Planning in MDPs

- Given an exact model (i.e., reward function, transit probabilities), and a fixed policy π

Value Iteration (Policy Evaluation)

For k = 0,1,2,...

$$\forall s \in S,\ V_{k+1}(s) = \sum_{a \in A} \pi(a|s)\left[R(s,a) + \sum_{s' \in S} P(s'|s,a)V_k(s')\right.$$

$$\forall s \in S,\ V_{k+1}(s) = R(s,\pi(s)) + \sum_{s' \in S} P(s'|s,\pi(s))V_k(s')$$

Stopping criterion: $\max_{s \in S} |V_{k+1}(s) - V_k(s)| \leq \varepsilon$

Arbitrary initialization: V_0

Planning in MDPs

Given a exact model (i.e., reward function, transition probabilities), and a fixed policy π

Value Iteration (Policy Evaluation)

For k = 0,1,2,...

$$\forall s \in S, a \in A \; Q_{k+1}(s,a) = R(s,a) + \sum_{s' \in S} P(s'|s,a) \Big(\sum_{b \in A} \pi(b|s') Q_k(s',$$

$$\forall s \in S, a \in A \; Q_{k+1}(s,a) = R(s,a) + \sum_{s' \in S} P(s'|s,a) Q_k(s', \pi(s'))$$

Stopping criterion: $\max_{s \in S, a \in A} |Q_{k+1}(s,a) - Q_k(s,a)| \leq \varepsilon$

Arbitrary initialization: Q_0

Planning in MDPs

Given a exact model (i.e., reward function, transition probabilities)

Value Iteration (Optimal Control)

For k = 0,1,2,...

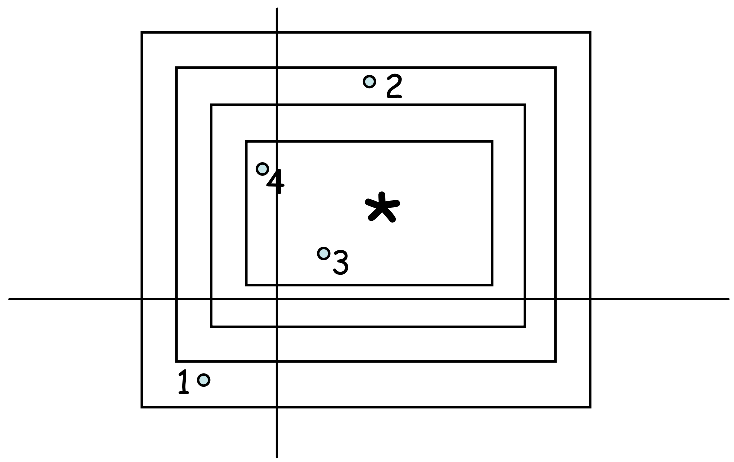
$$\forall s \in S,\ V_{k+1}(s) = \max_{a \in A}[R(s,a) + \sum_{s' \in S} P(s'|s,a) V_k(s')]$$

$$\forall s \in S, a \in A\ Q_{k+1}(s,a) = R(s,a) + \sum_{s' \in S} P(s'|s,a) \max_{b \in A} Q_k(s', b$$

Stopping criterion: $\max_{s \in S} |V_{k+1}(s) - V_k(s)| \leq \varepsilon$

$$\max_{s \in S, a \in A} |Q_{k+1}(s,a) - Q_k(s,a)| \leq \varepsilon$$

Convergence of *Value Iteration*



$$\forall k, ||Q_{k+1} - Q_k||_\infty = \max_{s \in S, a \in A} |Q_{k+1}(s,a) - Q_k(s,a)| \leq \gamma < 1$$

$$\forall k, ||V_{k+1} - V_k||_\infty = \max_{s \in S} |V_{k+1}(s) - V_k(s)| \leq \gamma < 1$$

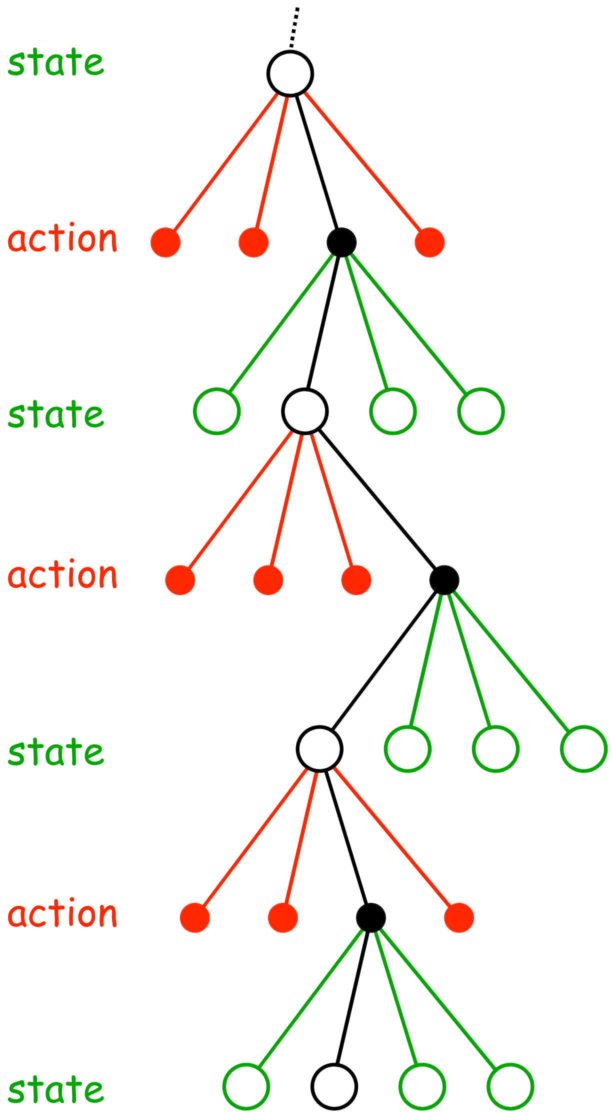
Contractions!

Proof of the DP contraction

Let $\Delta_k = ||Q^* - Q_k||_\infty$

$$
\begin{aligned}
Q_{k+1}(s,a) &= R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) \max_{b \in A} Q_k(s',b) \\
&\leq R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) \max_{b \in A} [Q^*(s',b) + \Delta_k] \\
&= \left[R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) \max_{b \in A} Q^*(s',b) \right] + \gamma \Delta_k \\
&= Q^*(s,a) + \gamma \Delta_k
\end{aligned}
$$

Learning in MDPs



- Have access to the "real system" but no model

Generate experience

$$s_0 a_0 r_0 s_1 a_1 r_1 \cdots s_k a_k r_k \cdots$$

This is what life looks like!

Two classes of approaches:

1. *Indirect methods*
2. *Direct methods*

Indirect Methods for Learning in MDP

- Use experience data to estimate model

$$\hat{P}(j|i,a) = \frac{\# j \leftarrow i,a}{\# j \leftarrow i,\cdot}$$

- Compute optimal policy w.r.to estimated model (Certainly equivalent policy)

- Exploration-Exploitation Dilemma

Model converges asymptotically provided all state-action pairs are visited infinitely often in the limit; hence certainty equivalent policy converges asymptotically to the optimal policy

Parametric models

Direct Method: Q-Learning

$s_0 a_0 r_0 \; s_1 a_1 r_1 \; s_2 a_2 r_2 \; s_3 a_3 r_3 \ldots \; s_k a_k r_k \ldots$

A unit of experience $< s_k \; a_k \; r_k \; s_{k+1} >$

Update:

$$Q_{new}(s_k,a_k) = (1-\alpha)\, Q_{old}(s_k,a_k) +$$
$$\alpha[r_k + \gamma \max_b Q_{old}(s_{k+1},b)]$$

step-size

Only updates state-action p
that are visited...

Big table of Q-values?

Watkins, 1988

Q-Learning Convergence w.p.1

$Q_{new}(s_k,a_k) = (1-\alpha)\, Q_{old}(s_k,a_k) +$

$\alpha[r_k + \gamma \max_b Q_{old}(s_{k+1},b)]$

Critical Observation: $E\{ r_k + \gamma \max_b Q_{old}(s_{k+1},b)\} =$

$R(s_k) + \gamma[\sum_{j \in S} P(j|s_k,a_k) \max_{b \in A} Q_{old}(j,b)]$

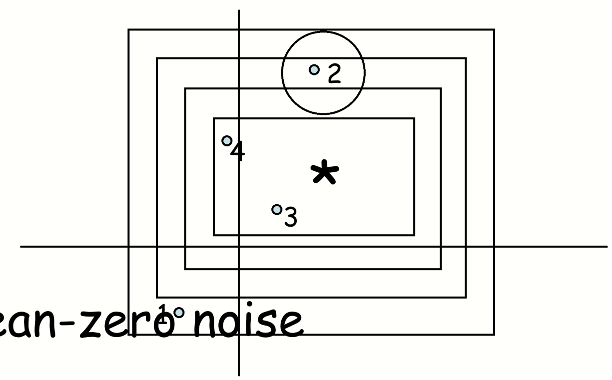
Q-learning is a stochastic approximation version of Q-value iteration! That is,

Q-value iteration is a deterministic algorithm

$Q_{k+1}=T(Q_k)$ and

Q-learning is a stochastic algorithm of the form

$Q_{k+1} = (1 - \alpha)Q_k + \alpha[T(Q_k) + \eta_k]$ where η_k is mean-zero noise



w.p.1	every state-action pair is updated infinitely often; tabular representation; $\sum\alpha=\infty$; $\sum\alpha^2$ is finite *Jaakkola, Jordan, & Singh;* *Tsitsiklis*

So far...

- Q-Learning is the first provably convergent *direct adaptive optimal control algorithm*
- Great impact on the field of modern Reinforcement Learning
  - smaller representation than models
  - automatically focuses attention to where it is needed, i.e., no sweeps through state space
  - though does not solve the exploration versus exploitation dilemma
    - epsilon-greedy, optimistic initialization, etc,...

Monte Carlo?

Suppose you want to find $V^{\pi}(s)$ for some fixed state

Start at state s and execute the policy for a long trajectory and compute the empirical discounted retur

Do this several times and average the returns across trajectories

How many trajectories?

Unbiased estimate whose variance improves with n

Application: Direct Methoc

Dog Training Grour
by
Kohl & Stone

Before Training
by
Kohl & Stone

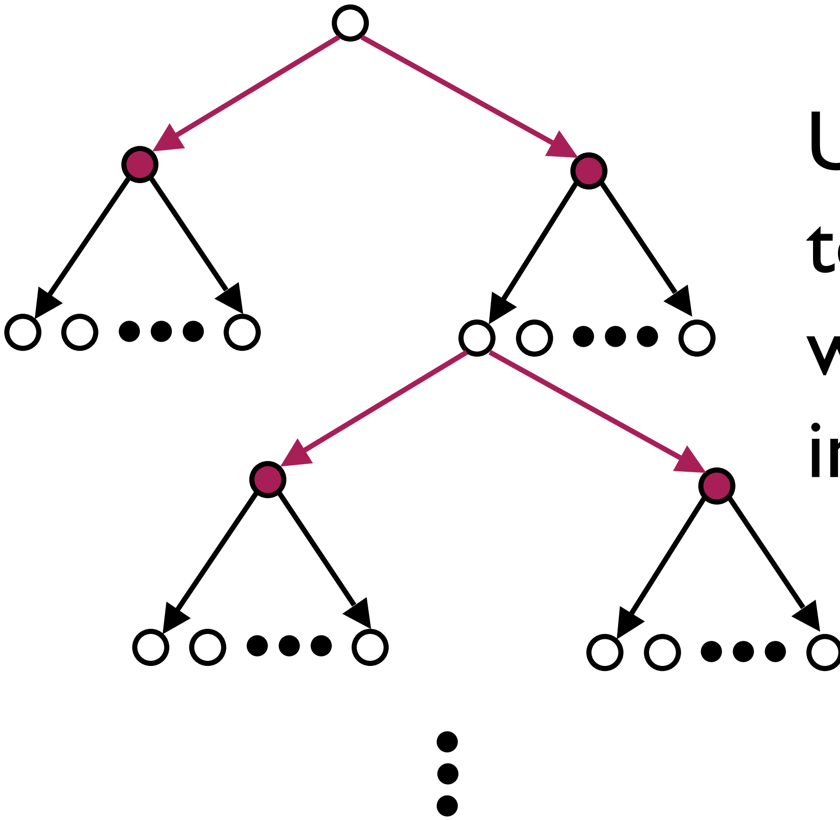
After Training
by
Kohl & Stone

Application: Indirect Metho

by Andrew Ng and colleagues

by Andrew Ng and colleagues

Sparse Sampling



Use generative model
to generate depth 'n' tree
with 'm' samples for each actic
in each state generated

Near-optimal action at root state in
time independent of the size of state sp
(but, exponential in horizon!)

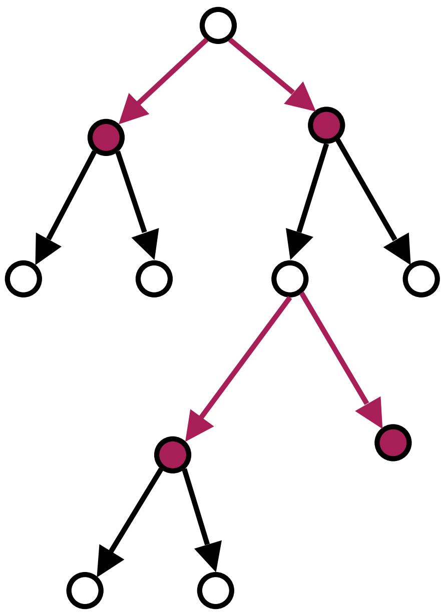
Kearns, Mansour & Ng

Classification for RL

- Use Sparse Sampling to derive a data set of examples of near-optimal actions for a subset of states
- Pass this data set to a classification algorithm
- Leverage algorithm and theoretical results on classification for RL

Langford

Trajectory Trees…



Given a set of policies to evaluate, the number of policy trees needed to find a near-optimal policy from the given set depends on the "VC-dim" of the class of policies

Kearns, Mansour & Ng

Summary

- Space of Algorithms:
  - (does not need a model) linear in horizon + polynomial in states
  - (needs generative model) Independent of states + exponential in horizon
  - (needs generative model) time complexity depends on the complexity of policy class

Eligibility Traces
(another key idea in RL)

Eligibility Traces

- The policy evaluation problem: given a (in general stochastic) policy π, estimate

  $$V^{\pi}(i) = E_{\pi}\{r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \dots \mid s_0 = i\}$$

  from multiple experience trajectories generated by following policy π repeatedly from state i

  A single trajectory:

  $$r_0 \quad r_1 \quad r_2 \quad r_3 \quad \dots\dots \quad r_k \quad r_{k+1} \quad \dots\dots$$

TD(λ)

$r_0 \quad r_1 \quad r_2 \quad r_3 \quad .... \quad r_k \quad r_{k+1} \quad ....$

0-step (e_0): $r_0 + \gamma V(s_1)$

$$V_{new}(s_0) = V_{old}(s_0) + \alpha\,[r_0 + \gamma V_{old}(s_1) - V_{old}(s_0)]$$

temporal difference

$$V_{new}(s_0) = V_{old}(s_0) + \alpha\,[e_0 - V_{old}(s_0)]$$

TD(0)

TD(λ)

r_0 r_1 r_2 r_3 .... r_k r_{k+1} ....

$r_0 + \gamma V(s_1)$

1-step (e_1): $r_0 + \gamma r_1 + \gamma^2 V(s_2)$

$$V_{new}(s_0) = V_{old}(s_0) + \alpha\,[e_1 - V_{old}(s_0)]$$

$$V_{old}(s_0) + \alpha\,[r_0 + \gamma r_1 + \gamma^2 V_{old}(s_2) - V_{old}(s_0)]$$

TD(λ)

$$r_0 \qquad r_1 \qquad r_2 \qquad r_3 \qquad .... \qquad r_k \qquad r_{k+1} \quad ....$$

$$w_0 \qquad e_0: \qquad r_0 + \gamma V(s_1)$$

$$w_1 \qquad e_1: \qquad r_0 + \gamma r_1 + \gamma^2 V(s_2)$$

$$w_2 \qquad e_2: \qquad r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 V(s_3)$$

$$\vdots$$

$$w_{k-1} \qquad e_{k-1}: \qquad r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \ldots \gamma^{k-1} r_{k-1} + \gamma^k V(s_k)$$

$$\vdots$$

$$w_\infty \qquad e_\infty: \qquad r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \ldots \gamma^k r_k + \gamma^{k+1} r_{k+1} + \ldots$$

$$V_{new}(s_0) = V_{old}(s_0) + \alpha \left[\textstyle\sum_k w_k e_k - V_{old}(s_0)\right]$$

TD(λ)

$$
\begin{array}{llllllll}
 & r_0 & r_1 & r_2 & r_3 & \ldots & r_k & r_{k+1} \quad \ldots \\
(1-\lambda) & r_0 + \gamma V(s_1) & & & & & & \\
(1-\lambda)\lambda & r_0 + & \gamma r_1 + \gamma^2 V(s_2) & & & & & \\
(1-\lambda)\lambda^2 & r_0 + & \gamma r_1 + & \gamma^2 r_2 & + \gamma^3 V(s_3) & & & \\
\vdots & & \vdots & & & & & \\
(1-\lambda)\lambda^{k-1} & r_0 + & \gamma r_1 + & \gamma^2 r_2 & + \gamma^3 r_3 + & \ldots \gamma^{k-1} r_{k-1} + \gamma^k V(s_k) & & \\
\vdots & & & \vdots & & & &
\end{array}
$$

$$V_{new}(s_0) = V_{old}(s_0) + \alpha \left[\sum_k (1-\lambda)\lambda^k e_k - V_{old}(s_0)\right]$$

$0 \leq \lambda \leq 1$ interpolates between 1-step TD and Monte-Carlo

TD(λ)

$r_0 \quad r_1 \quad r_2 \quad r_3 \quad \ldots. \quad r_k \quad r_{k+1} \quad \ldots.$

$\Delta_0 \qquad r_0 + \gamma V(s_1) - V(s_0)$

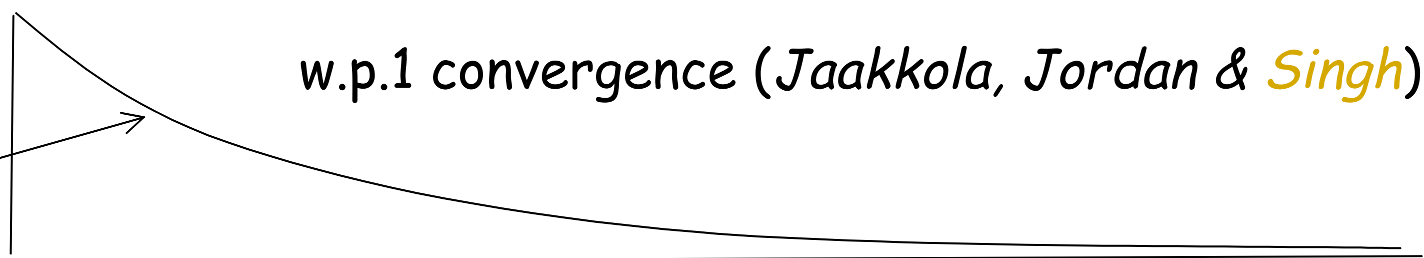
$\Delta_1 \qquad r_1 + \gamma V(s_2) - V(s_1)$

$\Delta_2 \qquad r_2 + \gamma V(s_3) - V(s_2)$

$\Delta_k \qquad r_{k-1} + \gamma V(s_k) - V(s_{k-1})$

$$V_{new}(s_0) = V_{old}(s_0) + \alpha \left[\sum_k (1-\lambda)\lambda^k \Delta_k\right]$$

eligibility trace



w.p.1 convergence (*Jaakkola, Jordan & Singh*)

Bias-Variance Tradeoff

decreasing bias

increasing variance

$$
\begin{array}{lllllllll}
 & r_0 & r_1 & r_2 & r_3 & \ldots & r_k & r_{k+1} & \ldots \\
e_0: & r_0 + \gamma V(s_1) & & & & & & & \\
e_1: & r_0 + & \gamma r_1 + \gamma^2 V(s_2) & & & & & & \\
e_2: & r_0 + & \gamma r_1 + & \gamma^2 r_2 & + \gamma^3 V(s_3) & & & & \\
 & & & \vdots & & & & & \\
e_{k-1}: & r_0 + & \gamma r_1 + & \gamma^2 r_2 & + \gamma^3 r_3 + & \ldots \gamma^{k-1} r_{k-1} + \gamma^k V(s_k) & & & \\
 & & & \vdots & & & & & \\
e_\infty: & r_0 + & \gamma r_1 + & \gamma^2 r_2 & + \gamma^3 r_3 + & \ldots \gamma^k r_k + & \gamma^{k+1} r_{k+1} + \ldots & &
\end{array}
$$

TD(λ)

$$< s, a, r, s' >$$

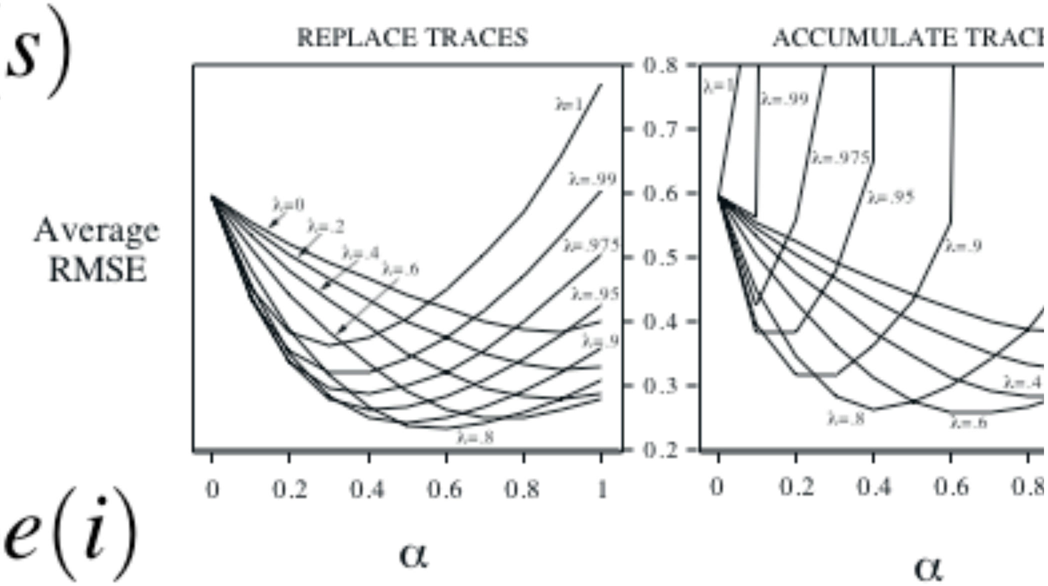
$$\delta \leftarrow r + \gamma V(s') - V(s)$$

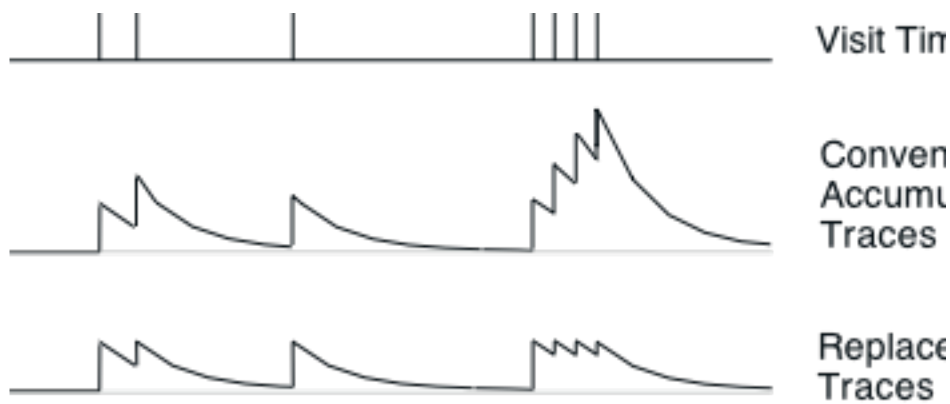
$$e(s) \leftarrow e(s) + 1$$

$$\forall i \in S:$$

$$V(i) \leftarrow V(i) + \alpha \delta e(i)$$

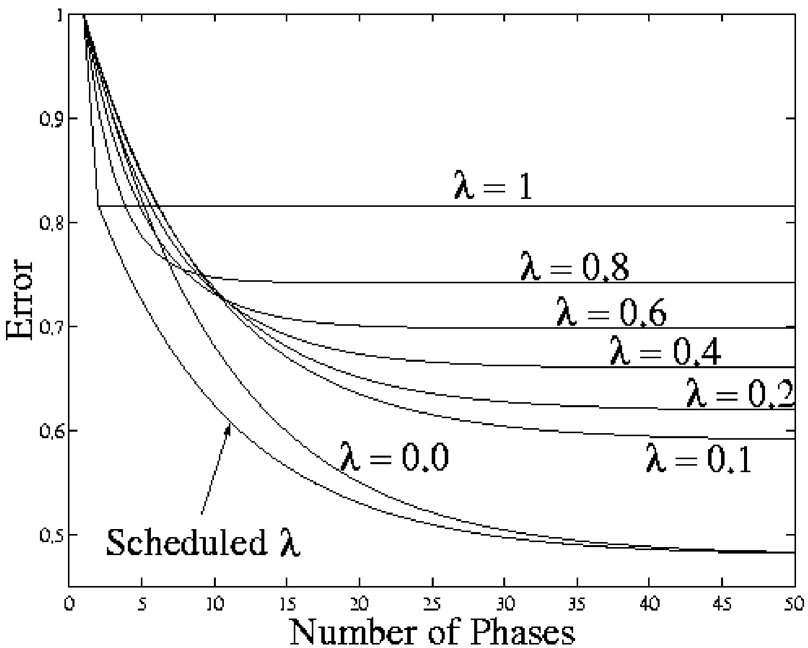
$$e(i) \leftarrow \gamma \lambda e(i)$$





Bias-Variance Tradeoff

Constant step-size



$$error_t \le a_\lambda \frac{1 - b_\lambda^t}{1 - b_\lambda} + b_\lambda^t$$

$t \uparrow \infty$, error asymptotes at $\dfrac{a_\lambda}{1 - b_\lambda}$

(an increasing function of λ)

Rate of convergence is b_λ^t(exponentia

b_λ is a decreasing function of λ

Intuition: start with large λ and then decrease over time

Kearns & Singh, 2000

Near-Optimal Reinforcement Learning ir Polynomial Time

(solving the exploration versus exploitation dilemma)

Setting

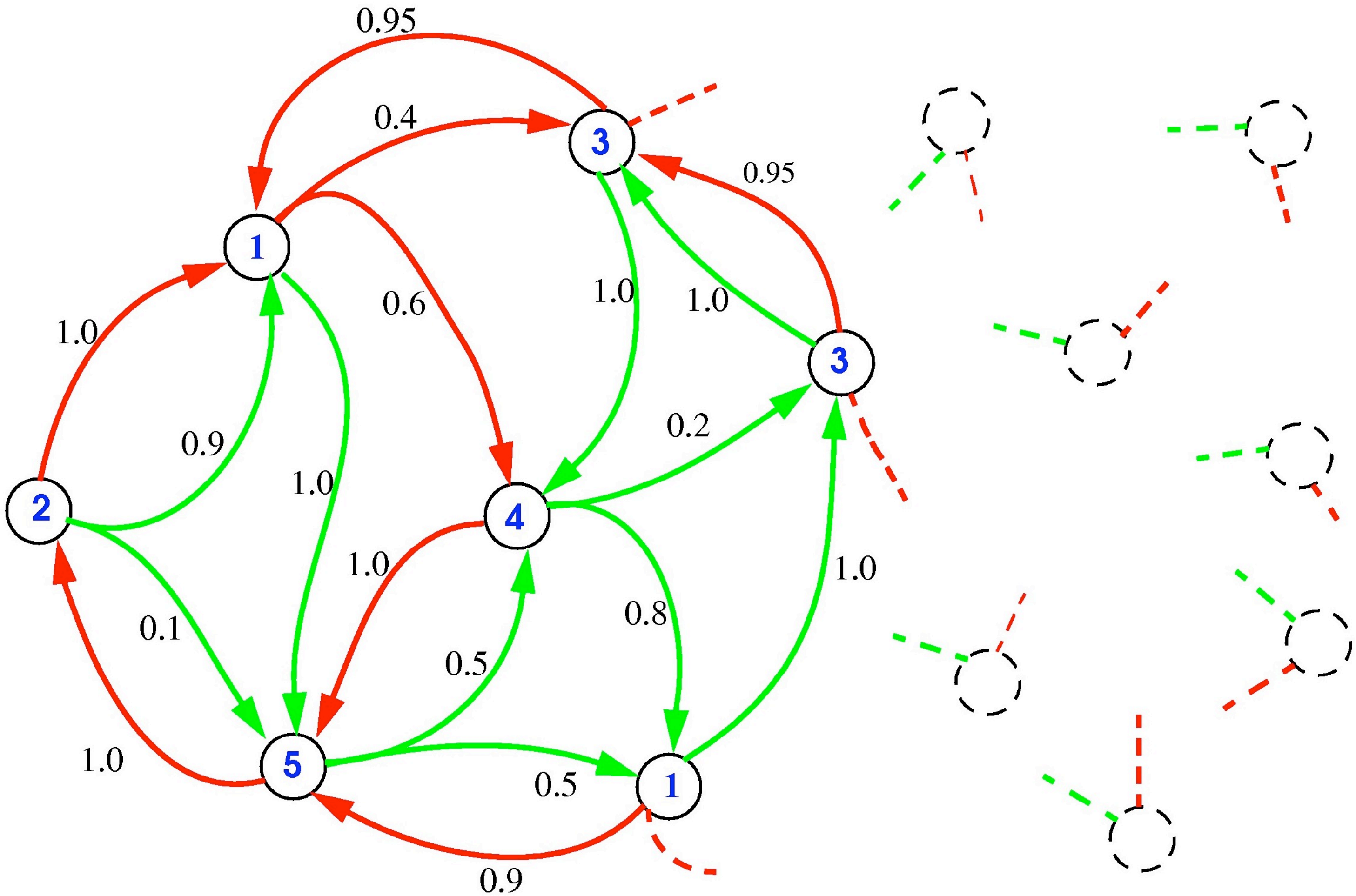
- Unknown MDP   M
- At any step: *explore* or *exploit*
- Finite time analysis
- Goal: Develop an algorithm such that an agent following that algorithm will in time polynomial in the complexity of the MDP, will achieve nearly the same *payoff per time step* as an agent that knew the MDP to begin with.
- Need to solve exploration versus exploitation
- Algorithm called E^3

Preliminaries

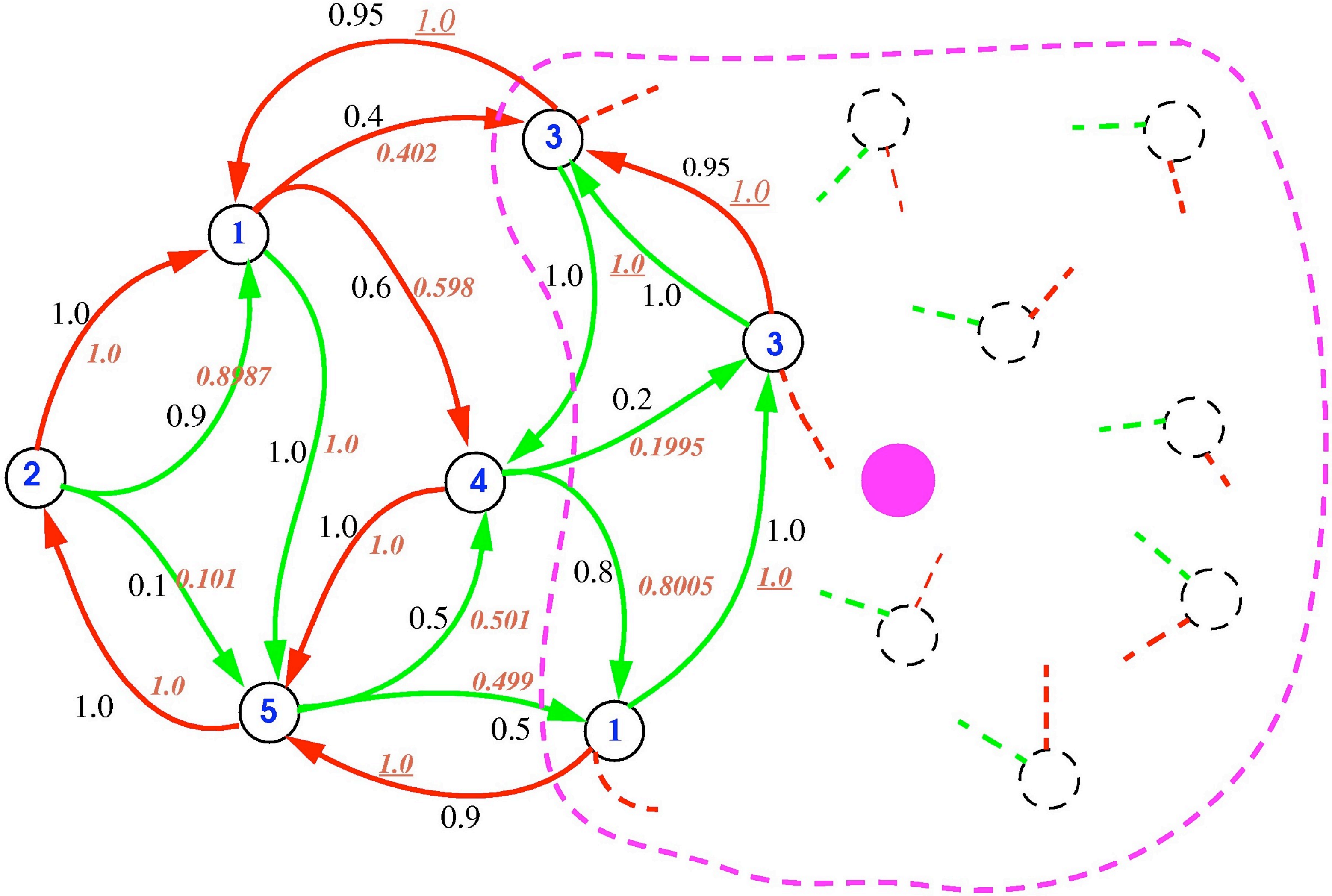
- Actual return: $\frac{1}{T}(R_1 + R_2 + \ldots + R_T)$
- Let T^* denote the (unknown) *mixing time* of the MDP
- One key insight: even the optimal policy will take time $O(T^*)$ to achieve actual return that is near-optimal
- E^3 has the property that it always compares favorably to the best policy amongst the policies that mix in the time that the algorithm run.

The Algorithm (informal)

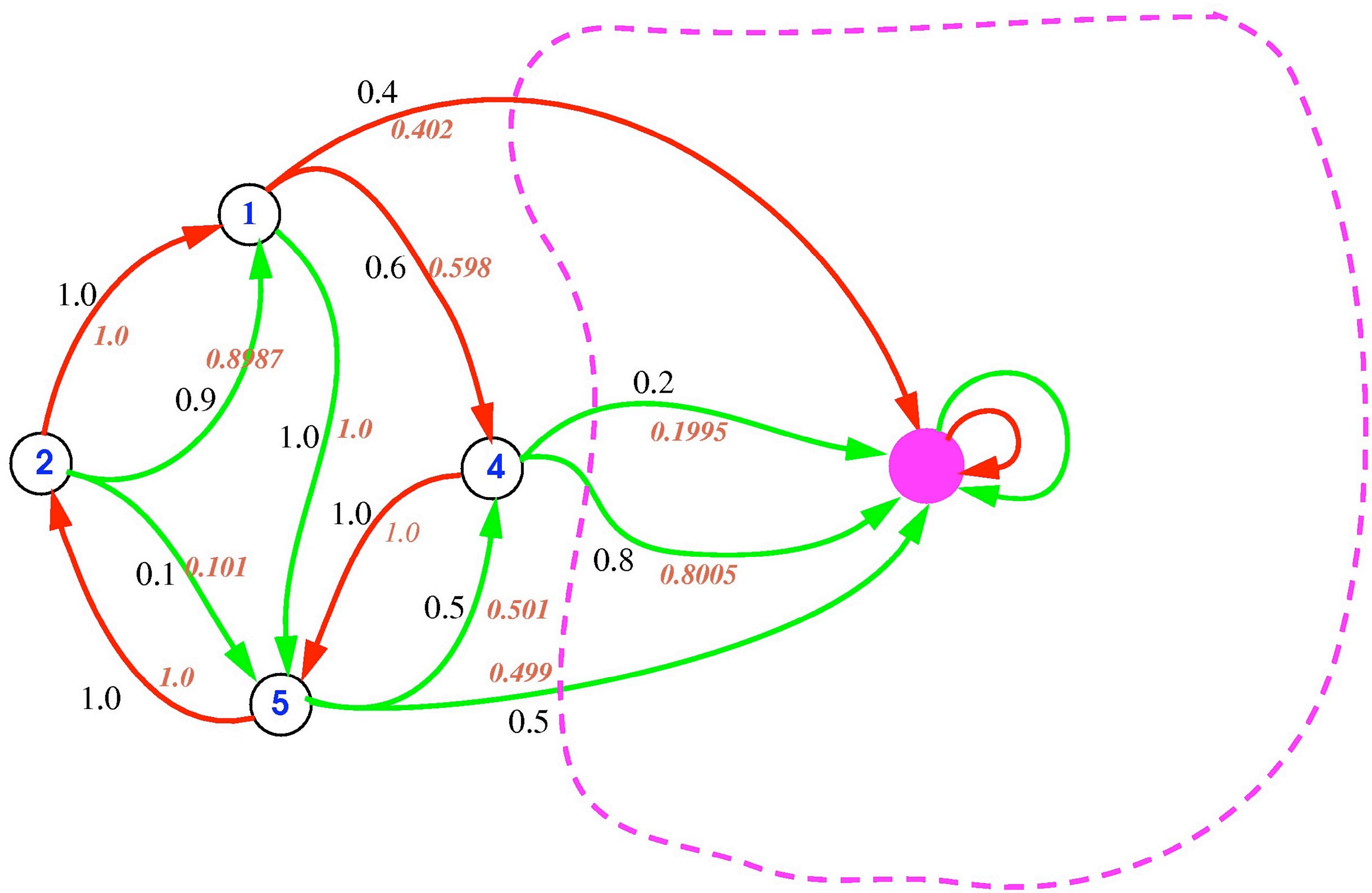
- Do "balanced wandering" until some state is *known*
- Do forever:
  - Construct *known-state* MDP
  - Compute optimal *exploitation* policy in known-state MDP
  - If return of above policy is near optimal, execute it
  - Otherwise compute optimal *exploration* policy in known-state MDP and execute it; do balanced wandering from unknown states.



0.95
0.4
3
0.95
1
1.0
1.0
0.6
1.0
3
0.9
0.2
2
1.0
4
1.0
1.0
0.1
0.8
0.5
1.0
5
0.5
1
0.9



0.95
1.0
0.4
0.402
3
1
0.95
1.0
0.6
0.598
1.0
1.0
1.0
1.0
3
1.0
1.0
0.8987
0.9
0.2
0.1995
1.0
1.0
2
4
1.0
1.0
1.0
0.1
0.101
0.8
0.8005
1.0
0.5
0.501
1.0
0.499
1.0
5
0.5
1
1.0
0.9



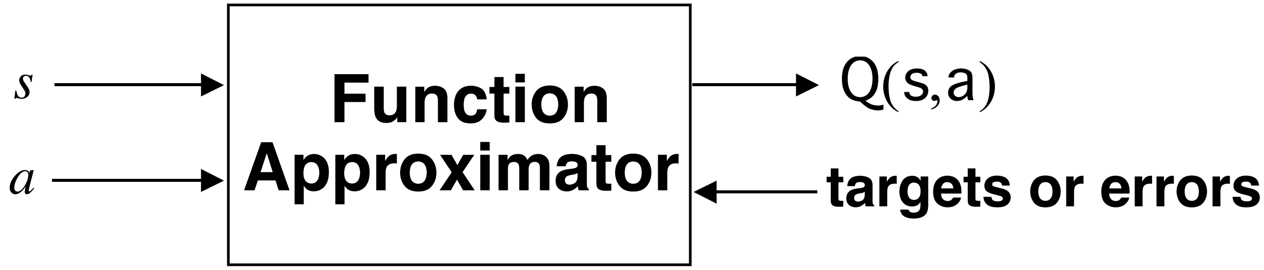
M: *true* known state MDP $\qquad$ $\hat{M}$: *estimated* known state MDP

Main Result

- A new algorithm E^3, taking inputs ε and δ such tha
for **any** V^* and T^* holding in the unknown MDP:
  - Total number of actions and computation time required by E^3 are **poly**($\frac{1}{\varepsilon}$, $\frac{1}{\delta}$, T^*, N)
  - Performance guarantee: with probability at least (1- δ) amortized return of E^3 so far will exceed $(1 - \varepsilon)V^*$

Function Approximation and Reinforcement Learning

General Idea



Could be:

- table
- Backprop Neural Network
- Radial-Basis-Function Network
- Tile Coding (CMAC)
- Nearest Neighbor, Memory Based
- Decision Tree

gradient-descent methods

Neural Networks as FAs

$$Q(s,a) = f(s,a,w)$$

weight vector

standard backprop gradient

e.g., gradient-descent Sarsa:

$$w \leftarrow w + \alpha\left[\underbrace{r_{t+1} + \gamma Q(s_{t+1},a_{t+1})}_{\textbf{target value}} - Q(s_t,a_t)\right] \nabla_w f(s_t,a_t,w)$$

estimated value

Linear in the Parameters FAs

$$\hat{V}(s) = \vec{\theta}^T \vec{\phi}_s \qquad \nabla_{\vec{\theta}} \hat{V}(s) = \vec{\phi}_s$$

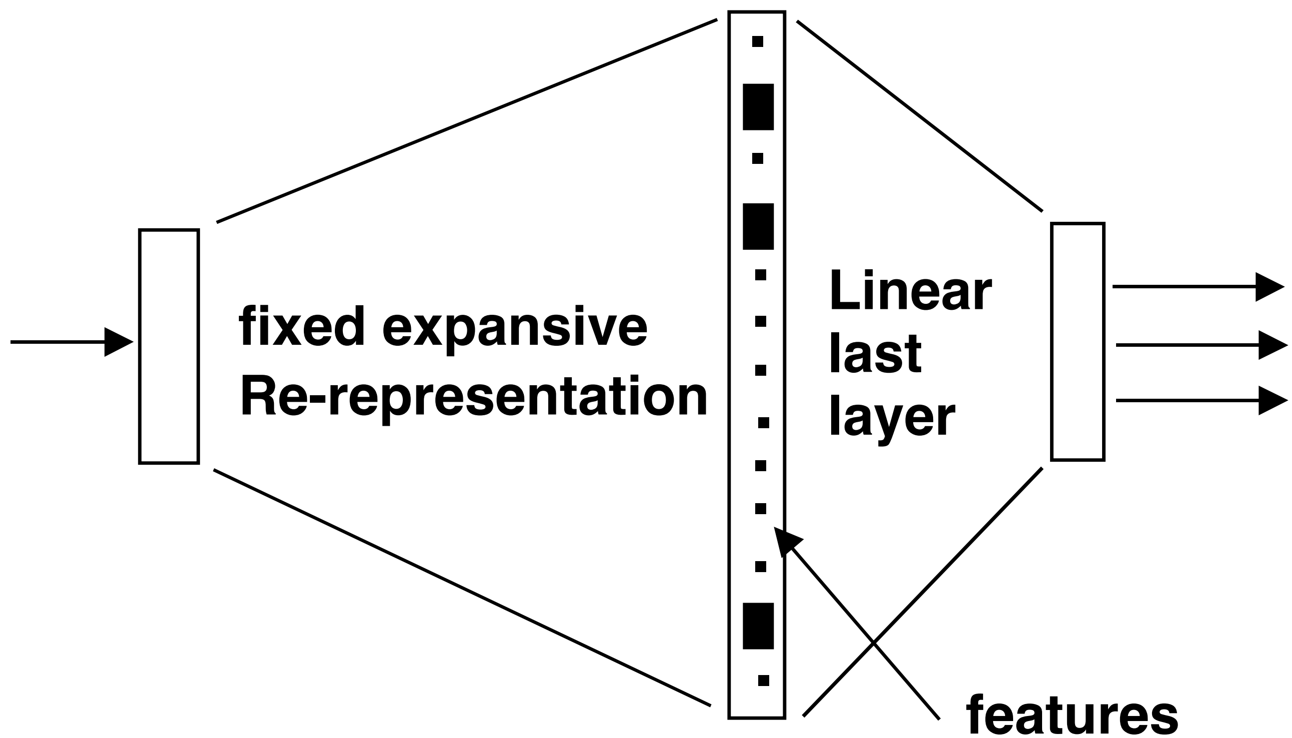
Each state s represented by a feature vector $\vec{\phi}_s$

Or represent a *state-action pair* with $\vec{\phi}_{sa}$
and approximate *action values*:

$$Q^\pi(s,a) = E\left\langle r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots \middle| s_t = s, \underline{a_t = a}, \pi \right\rangle$$

$$\hat{Q}(s,a) = \vec{\theta}^T \vec{\phi}_{s,a}$$

Sparse Coarse Coding



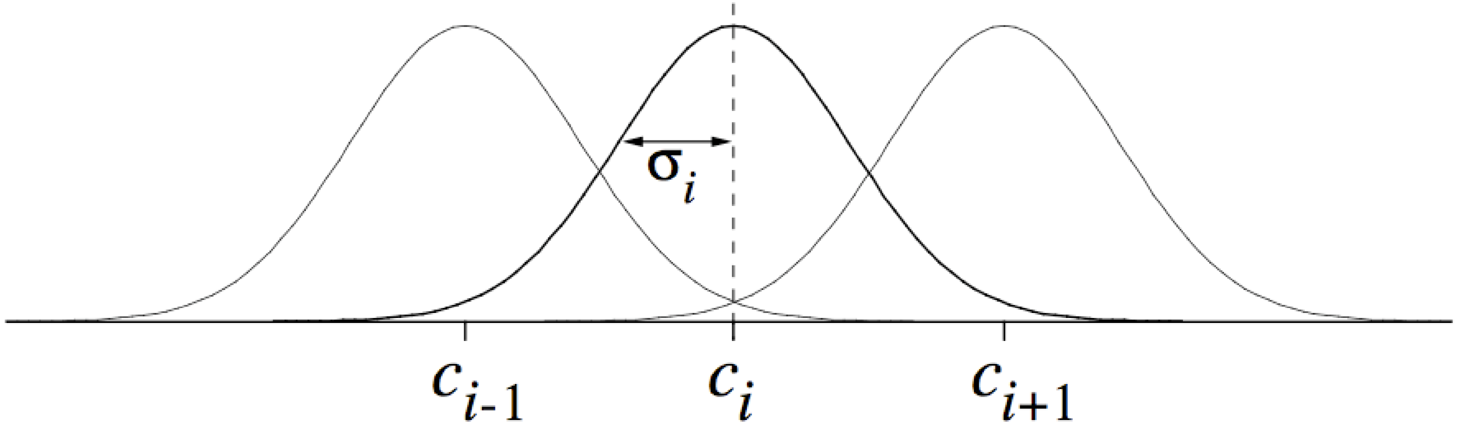
Coarse: Large receptive fields

Sparse: Few features present at one time

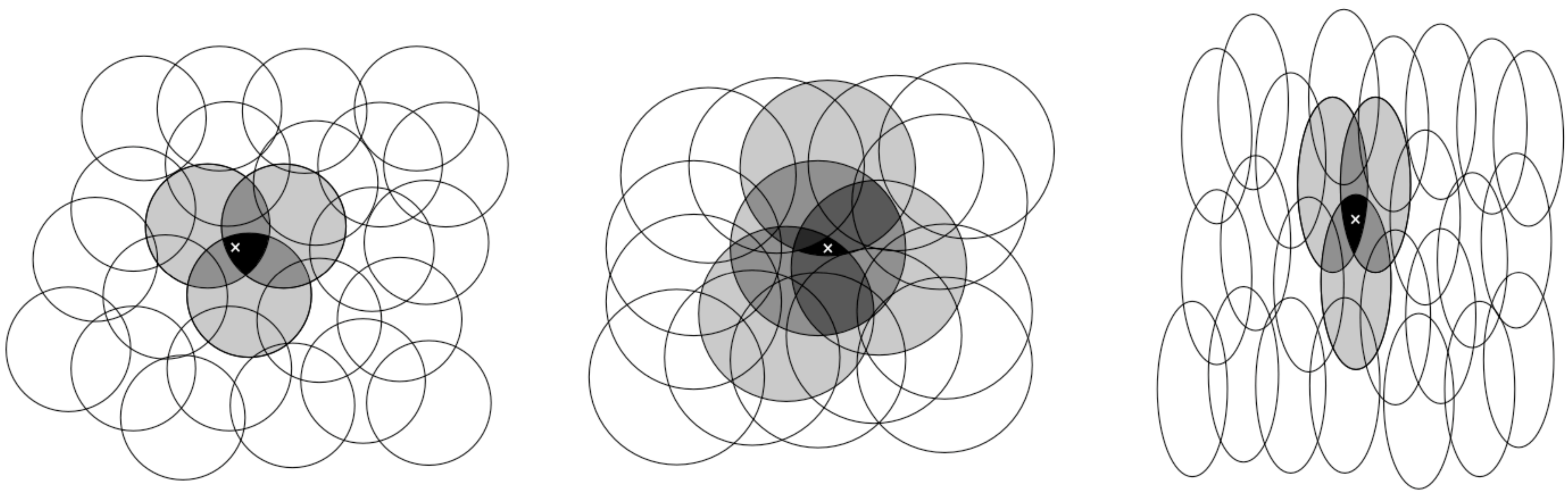
Radial Basis Functions (RBFs)

e.g., Gaussians

$$\phi_s(i) = \exp\left(-\frac{\left\| s - c_i \right\|^2}{2\sigma_i^2} \right)$$

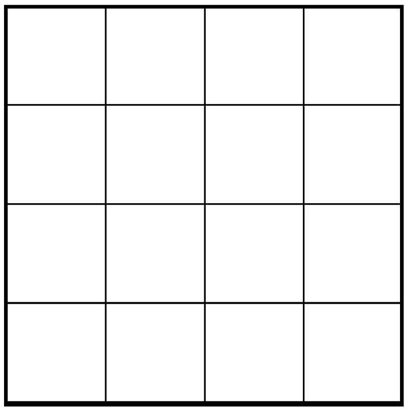


Shaping Generalization in Coarse Coding

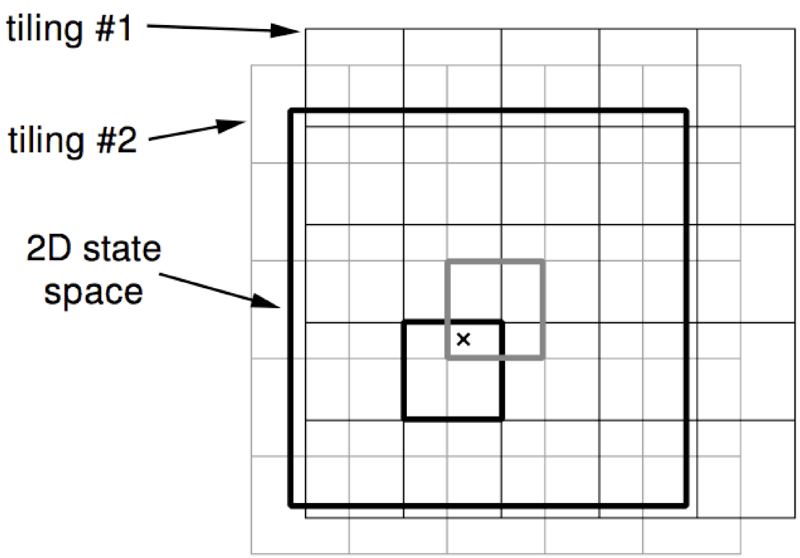


a) Narrow generalization b) Broad generalization c) Asymmetric generalizatio

Tile Coding



- Binary feature for each tile
- Number of features present at any one time is constant
- Binary features means weighted sum easy to

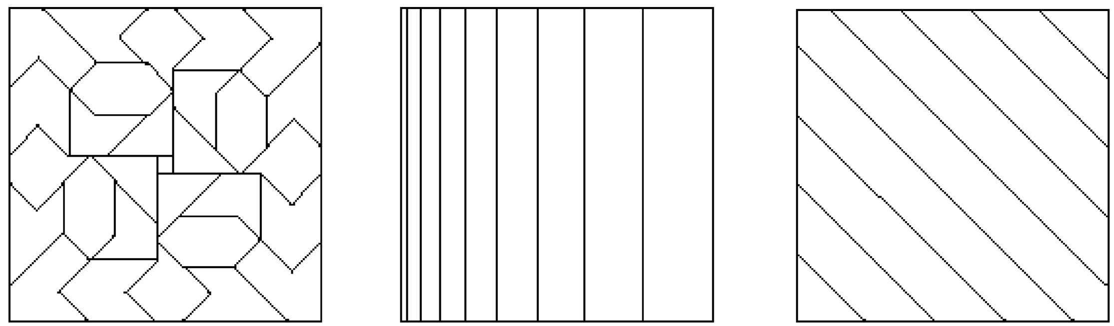


Shape of tiles ⇒ Generalization

#Tilings ⇒ Resolution of final approximation

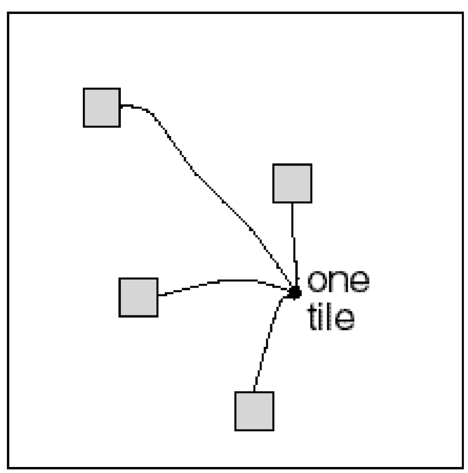
Tile Coding Cont.

Irregular tilings



a) Irregular b) Log stripes c) Diagonal stripes

Hashing



CMAC
"Cerebellar model arithmetic computer"
Albus 1971

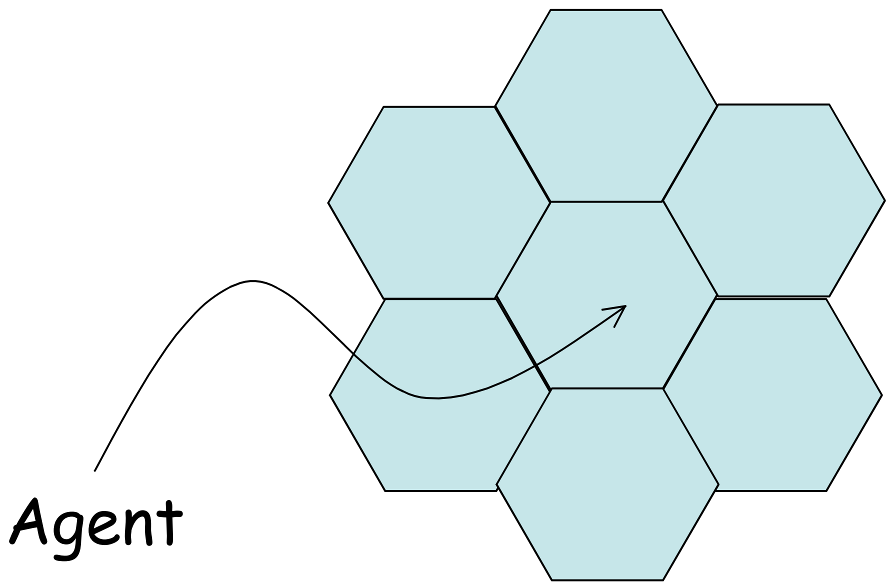
FAs & RL

- Linear FA (divergence can happen)
  Nonlinear Neural Networks (theory is not well developed)
  Non-parametric, e.g., nearest-neighbor (provably not divergent; bounds on error)
  Everyone uses their favorite FA… little theoretical guidance yet!
- Does FA really beat the curse of dimensionality?
  - Probably; with FA, computation seems to scale with the complexity of the solution (crinkliness of the value function) an
    how hard it is to find it
- Empirically it works
  - though many folks have a hard time making it so
  - no off-the-shelf FA+RL yet

by Andrew Ng and colleagues

Dynamic Channel Assignment in Cellular Telephones

Dynamic Channel Assignment



State: current assignments
Actions: feasible assignments
Reward: 1 per call per sec.

Channel assignment in cellular telephone systems

- what (if any) conflict-free channel to assign to caller

Learned better dynamic assignment policies than competition

Singh & Bertsekas (NIPS)

Run Cellphone Demo

(http://www.eecs.umich.edu/~baveja/Demo.html)

After MDPs...

- Great success with MDPs
- What next?
  - Rethinking Actions, States, Rewards
  - Options *instead* of actions
  - POMDPs

Rethinking Action (Hierarchical RL) Options

(Precup, Sutton, Singh)

MAXQ by Dietterich
HAMs by Parr & Russell

Related Work

“Classical” AI

Fikes, Hart & Nilsson(1972)
Newell & Simon (1972)
Sacerdoti (1974, 1977)

Macro-Operators
Korf (1985)
Minton (1988)
Iba (1989)
Kibler & Ruby (1992)

Qualitative Reasoning
Kuipers (1979)
de Kleer & Brown (1984)
Dejong (1994)

Laird et al. (1986)
Drescher (1991)
Levinson & Fuchs (1994)
Say & Selahatin (1996)
Brafman & Moshe (1997)

Robotics and Control Engineering

Brooks (1986)
Maes (1991)
Koza & Rice (1992)
Brockett (1993)
Grossman et. al (1993)
Dorigo & Colombetti (1994)
Asada et. al (1996)
Uchibe et. al (1996)
Huber & Grupen(1997)
Kalmar et. al (1997)
Mataric(1997)
Sastry (1997)
Toth et. al (1997)

Reinforcement Learr and MDP Planning

Mahadevan & Connell (199
Singh (1992)
Lin (1993)
Dayan & Hinton (1993)
Kaelbling(1993)
Chrisman (1994)
Bradtke & Duff (1995)
Ring (1995)
Sutton (1995)
Thrun & Schwartz (1995)
Boutilier et. al (1997)
Dietterich(1997)
Wiering & Schmidhuber (19
Precup, Sutton & Singh (19
McGovern & Sutton (1998)
Parr & Russell (1998)
Drummond (1998)
Hauskrecht et. al (1998)
Meuleau et. al (1998)
Ryan and Pendrith (1998)

Abstraction in Learning and Planning

- A long-standing, key problem in AI !
- How can we give abstract knowledge a clear semantics?

  e.g. “I could go to the library”
- How can different levels of abstraction be related?
  - spatial: states
  - temporal: time scales
- How can we handle stochastic, closed-loop, temporally extended courses of action?
- Use RL/MDPs to provide a theoretical foundation

Options

A generalization of actions to include courses of acti

An option is a triple $o = < \mathsf{I}, \pi, \beta >$

- $\mathsf{I} \subseteq \mathsf{S}$ is the set of states in which o may be started
- $\pi : \mathsf{S} \times \mathsf{A} \rightarrow [0,1]$ is the policy followed during o
- $\beta : \mathsf{S} \rightarrow [0,1]$ is the probability of terminating in each state

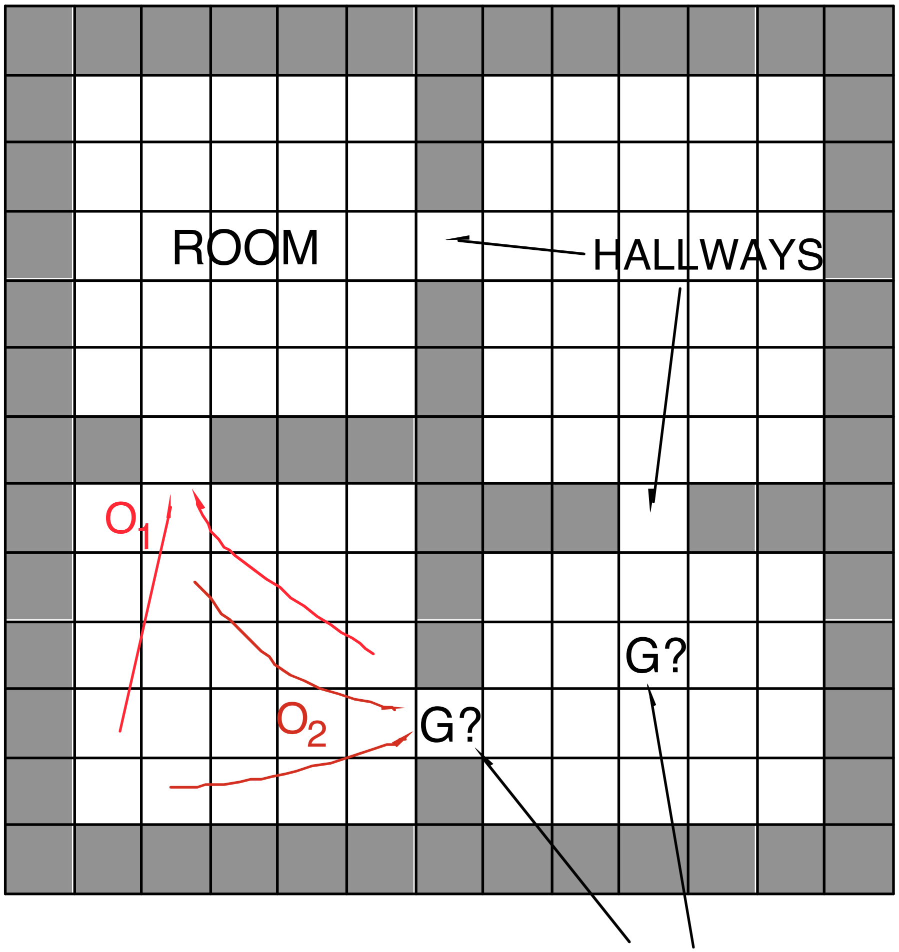
Option execution is assumed to be call-and-return

Example: `docking`

I : all states in which charger is in sight
π : hand-crafted controller
β : terminate when docked or charger not visible

Options can take variable number of steps

Rooms Example

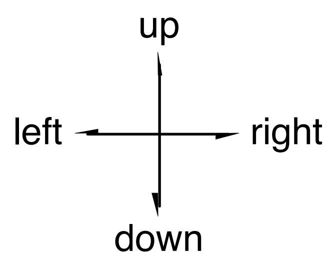


Goal states are given
a terminal value of 1

4 rooms

4 hallways

4 unreliable
primitive actions



Fail 33%
of the time

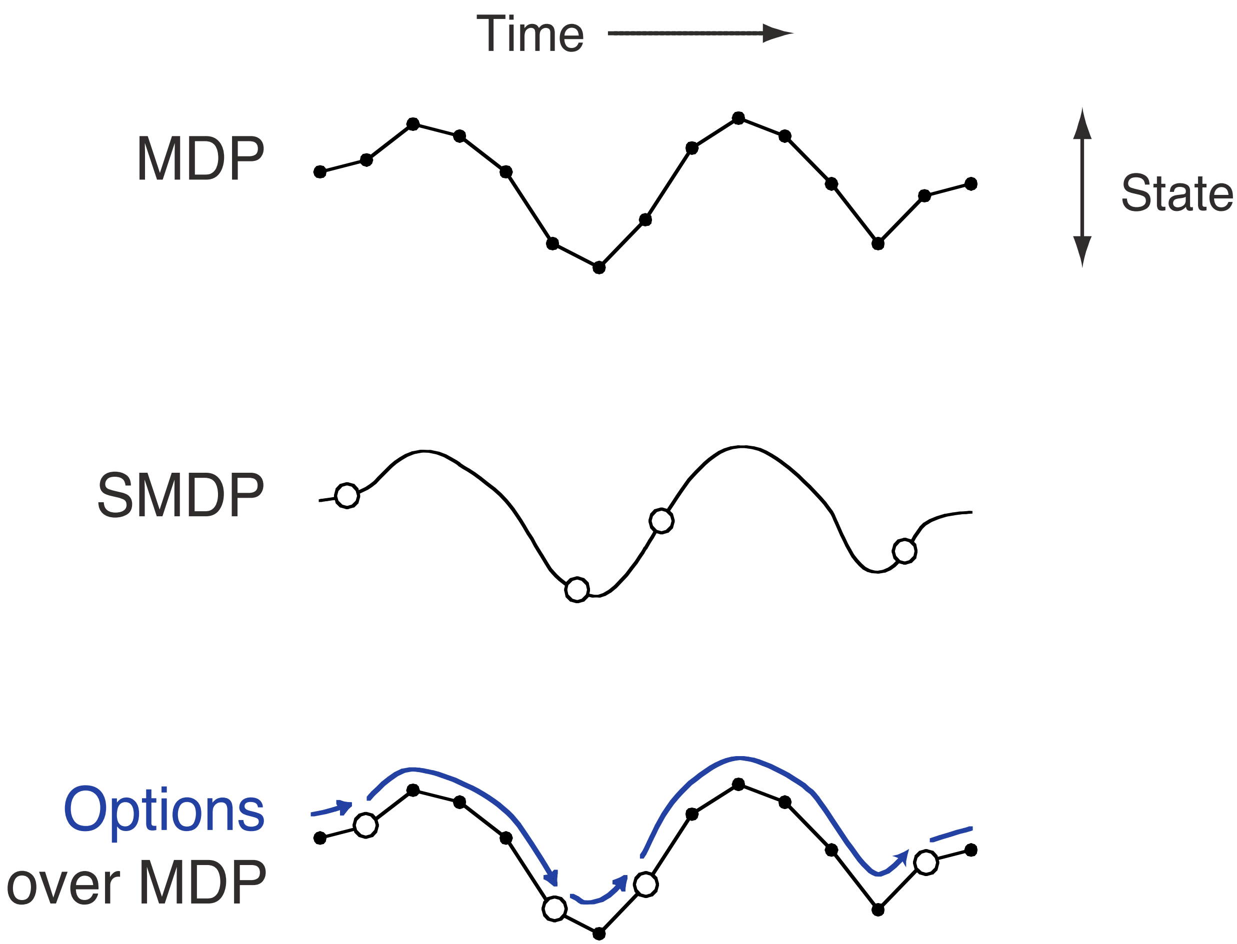
8 multi-step options

(to each room's 2 hallways)

Given goal location,
quickly plan shortest route

All rewards zero
$\gamma = .9$

Options define a Semi-Markov Decis Process (SMDP)



Discrete time
Homogeneous discou

Continuous time
Discrete events
Interval-dependent dis

Discrete time
Overlaid discrete ever
Interval-dependent dis

A discrete-time SMDP <u>overlaid</u> on an MDP
Can be analyzed at either level

MDP + Options = SMDP

Theorem:

For any MDP,
and any set of options,
the decision process that chooses among the options,
executing each to termination,
is an SMDP.

Thus all Bellman equations and DP results extend for value functions over options and models of options (cf. SMDP theory).

What does the SMDP connection give us?

- Policies over options: $\mu : \mathcal{S} \times \mathcal{O} \rightarrow [0,1]$
- Value functions over options: $V^{\mu}(s), Q^{\mu}(s,o), V^{*}_{\mathcal{O}}(s), Q^{*}_{\mathcal{O}}(s,o)$
- Learning methods: Bradtke & Duff (1995), Parr (1998)
- Models of options
- Planning methods: e.g. value iteration, policy iteration, Dyna
- A coherent theory of learning and planning with courses of action at variable time scales, yet at the same level

<u>A theoretical fondation for what we really need!</u>

But the most interesting issues are beyond SMDPs...

Value Functions for Options

Define value functions for options, similar to the MDP ca

$$V^{\mu}(s) = E\{r_{t+1} + \gamma r_{t+2} + \ldots \mid E(\mu, s, t)\}$$
$$Q^{\mu}(s,o) = E\{r_{t+1} + \gamma r_{t+2} + \ldots \mid E(o\mu, s, t)\}$$

Now consider policies $\mu \in \Pi(O)$ restricted to choose only from options in O :

$$V_O^*(s) = \max_{\mu \in \Pi(O)} V^{\mu}(s)$$
$$Q_O^*(s,o) = \max_{\mu \in \Pi(O)} Q^{\mu}(s,o)$$

Models of Options

Knowing how an option is executed is not enough for reasoning ab
it, or planning with it. We need information about its consequences

The model of the consequences of starting option o in state s has:

- a reward part

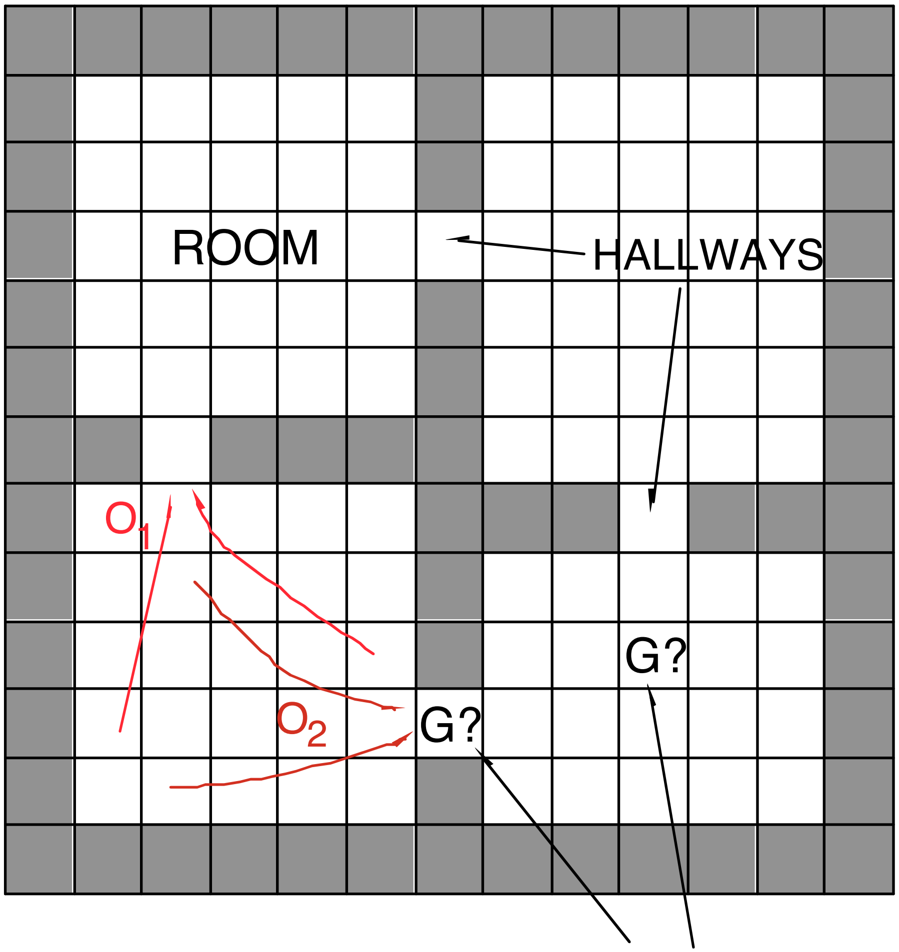
  $$r_s^o = E\{r_1 + \gamma r_2 + \ldots + \gamma^{k-1} r_k \mid s_0 = s,\ o \text{ taken in } s_0, \text{ lasts } k \text{ steps}\}$$

- a next-state part

  $$p_{ss'}^o = E\{\gamma^k \underbrace{\delta_{s_k s'}}_{1 \text{ if } s' = s_k \text{ is the termination state, } 0 \text{ otherwise}} \mid s_0 = s,\ o \text{ taken in } s_0, \text{ lasts } k \text{ steps}\}$$

This form follows from SMDP theory. Such models can be used
interchangeably with models of primitive actions in Bellman equatic

Room Example

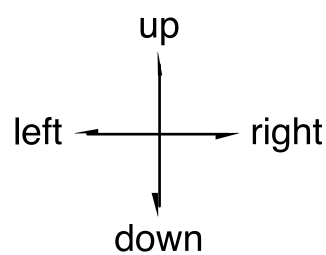


Goal states are given a terminal value of 1

4 rooms

4 hallways

4 unreliable primitive actions



Fail 33% of the time

8 multi-step options

(to each room's 2 hallways)

Given goal location, quickly plan shortest route

All rewards zero

$\gamma = .9$

Example: Synchronous Value Iterati
Generalized to Options

$$\text{Initialize: } V_0(s) \leftarrow 0 \qquad \forall s \in \mathsf{S}$$

$$\text{Iterate: } \quad V_{k+1}(s) \leftarrow \max_{o \in \mathsf{O}} [r_s^o + \sum_{s' \in \mathsf{S}} p_{ss'}^o V_k(s')] \qquad \forall s \in \mathsf{S}$$

The algorithm converges to the optimal value function, given the optio

$$\lim_{k \to \infty} V_k = V_{\mathsf{O}}^*$$

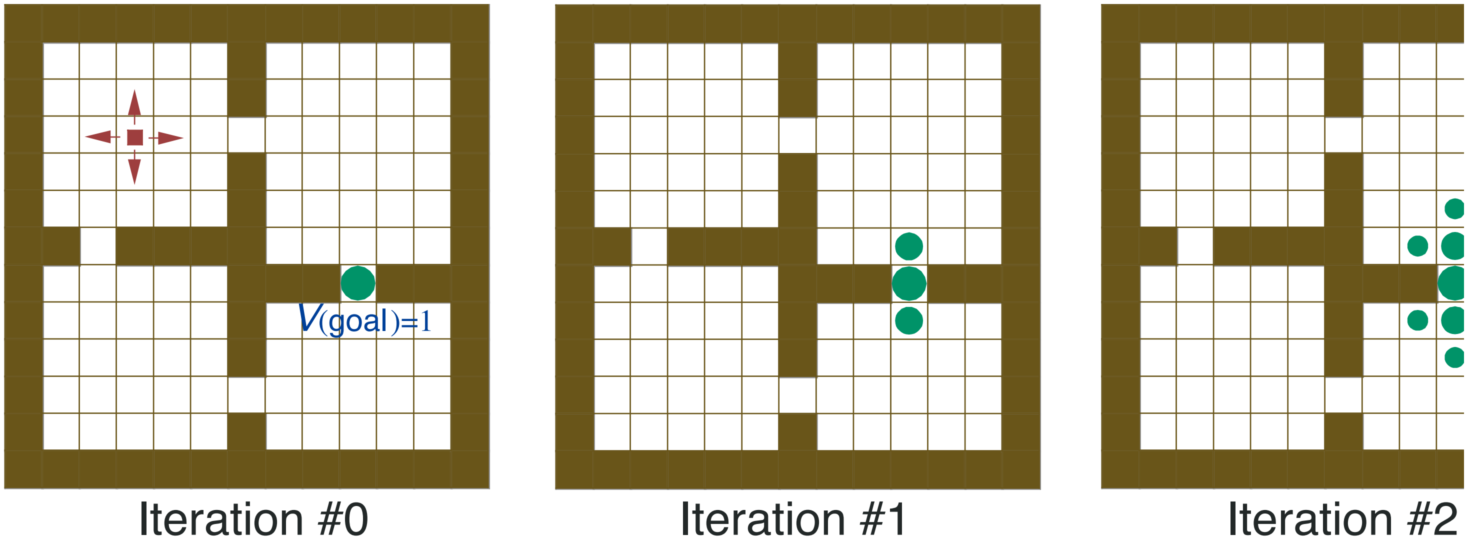
Once V_{O}^* is computed, μ_{O}^* is readily determined.

If $\mathsf{O} = \mathsf{A}$, the algorithm reduces to conventional value iteration

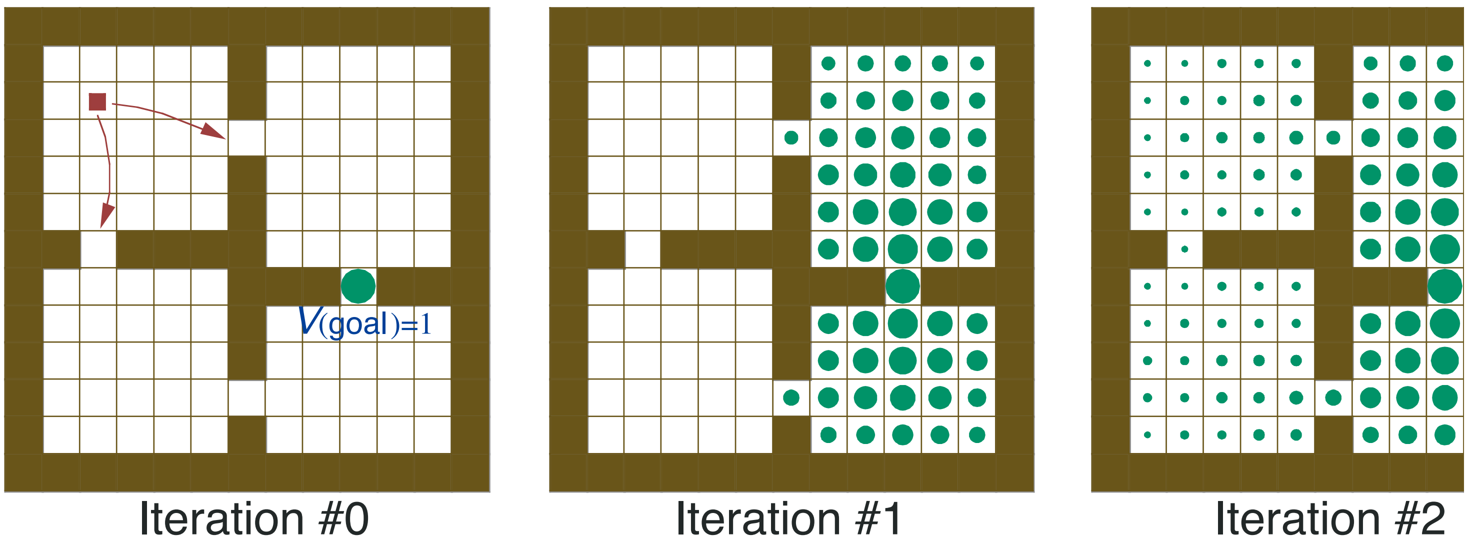
If $\mathsf{A} \subseteq \mathsf{O}$, then $V_{\mathsf{O}}^* = V^*$

Rooms Example

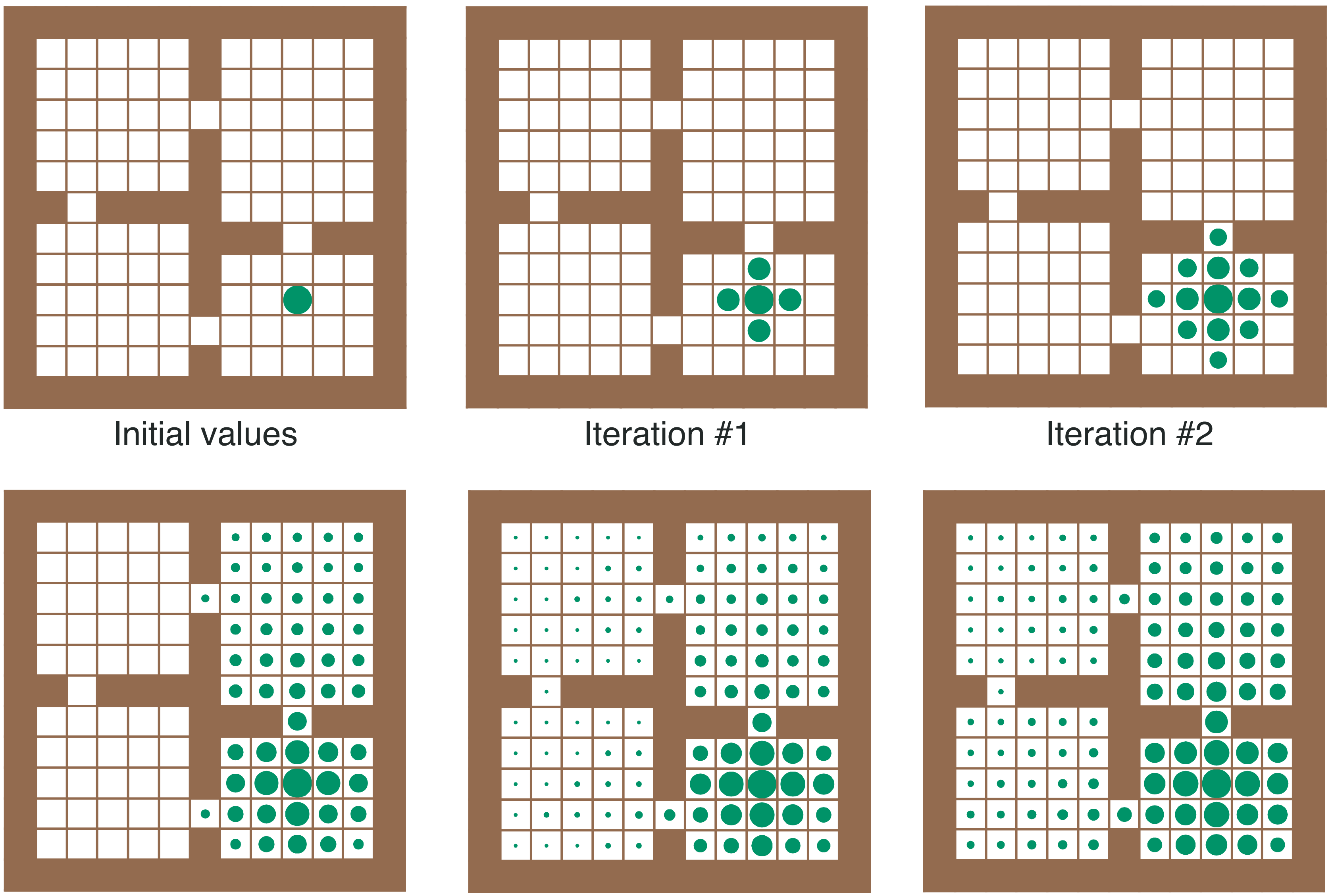
with cell-to-cell primitive actions



with room-to-room options



Example with Goal≠Subgoal
both primitive actions and options



Initial values | Iteration #1 | Iteration #2 | Iteration #3 | Iteration #4 | Iteration #5

What does the SMDP connection give us?

- Policies over options: $\mu : S \times O \mapsto [0,1]$
- Value functions over options: $V^{\mu}(s), Q^{\mu}(s,o), V_O^*(s), Q_O^*(s,o)$
- Learning methods: Bradtke & Duff (1995), Parr (1998)
- Models of options
- Planning methods: e.g. value iteration, policy iteration, Dyı
- A coherent theory of learning and planning with courses of action at variable time scales, yet at the same level

A theoretical foundation for what we really need!

But the most interesting issues are beyond SMDPs...

Advantages of Dual MDP/SMDP Viev

At the SMDP level

Compute *value functions and policies over options* with the benefit of increased speed / flexibility

At the MDP level

Learn *how* to execute an option for achieving a given goal

Between the MDP and SMDP level

Improve over existing options (e.g. by terminating ea

Learn about the effects of several options in parallel *without executing them to termination*

Between MDPs and SMDPs

- **Termination Improvement**

  Improving the value function by changing the termination conditions of options

- **Intra-Option Learning**

  Learning the values of options in parallel, without executing the[m] to termination

  Learning the models of options in parallel, without executing them to termination

- **Tasks and Subgoals**

  Learning the policies inside the options

Termination Improvement

Idea: We can do better by sometimes interrupting ongoing optior
- forcing them to terminate before β says to

Theorem: For any policy over options $\mu : \mathsf{S} \times \mathsf{O} \to [0,1]$,
suppose we interrupt its options one or more times, when

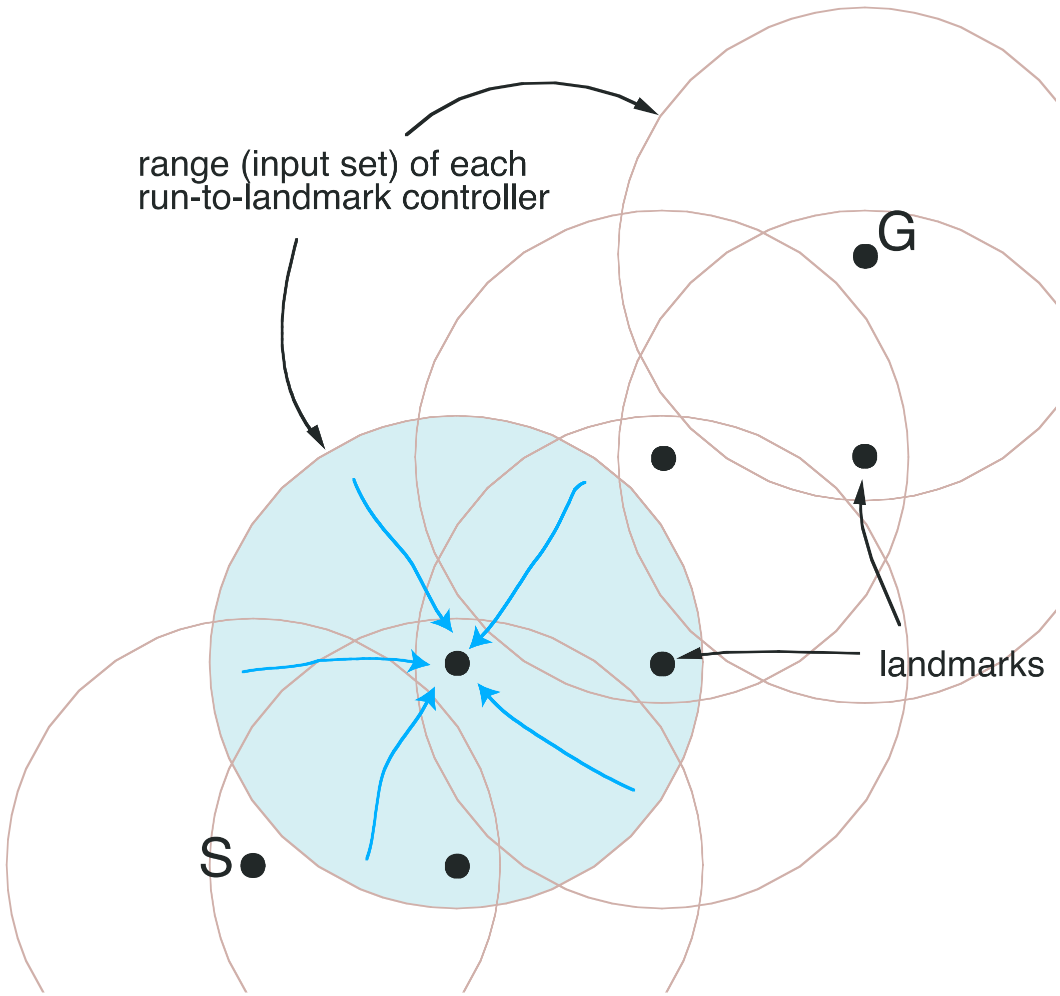
$$Q^{\mu}(s,o) < Q^{\mu}(s,\mu(s)), \quad \text{where } s \text{ is the state at that time, } o \text{ is the ongoing option}$$

to obtain $\mu' : \mathsf{S} \times \mathsf{O}' \to [0,1]$,

Then $\mu' > \mu$ (it attains more or equal reward everywhere)

Application: Suppose we have determined Q^*_{O} and thus $\mu = \mu^*_{\mathsf{O}}$.
Then μ' is guaranteed better than μ^*_{O}
and is available with no additional computation.

Landmarks Task



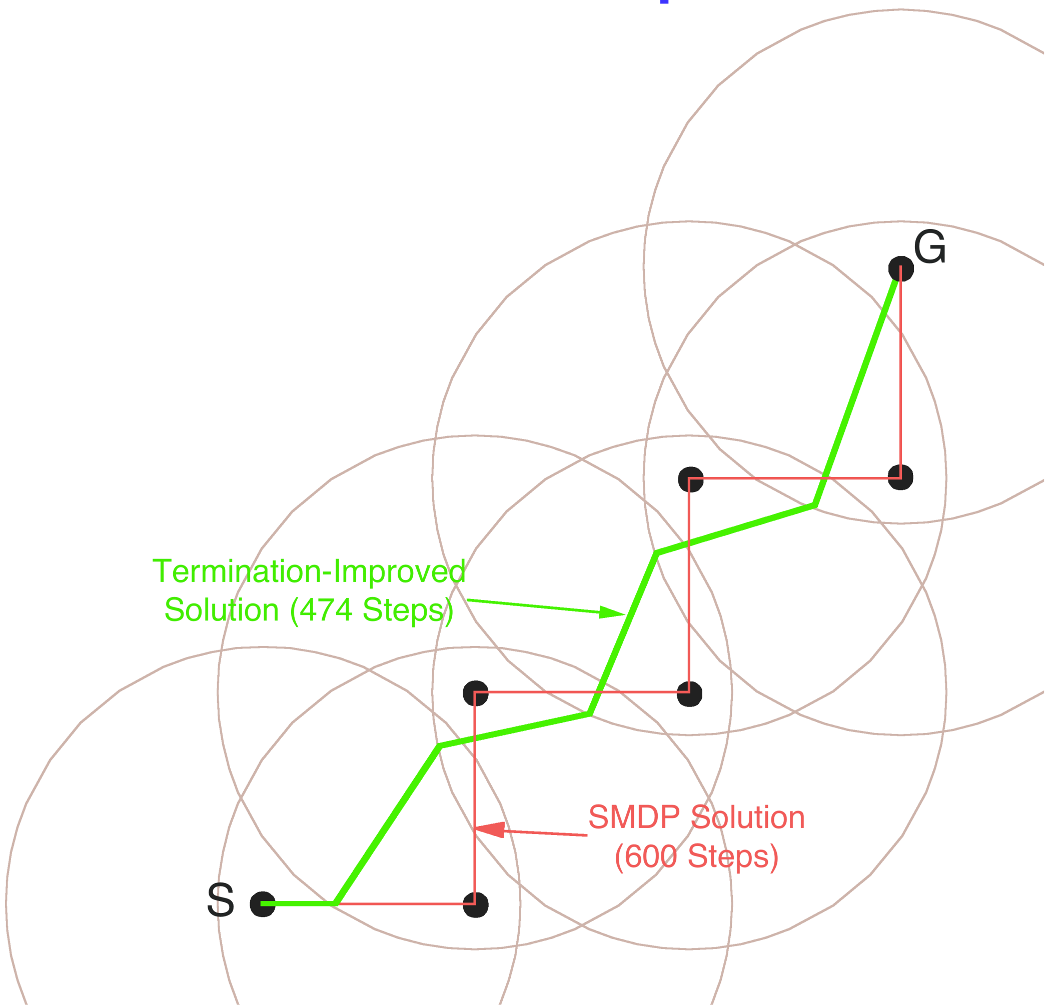
Task: navigate from S to
fast as possible

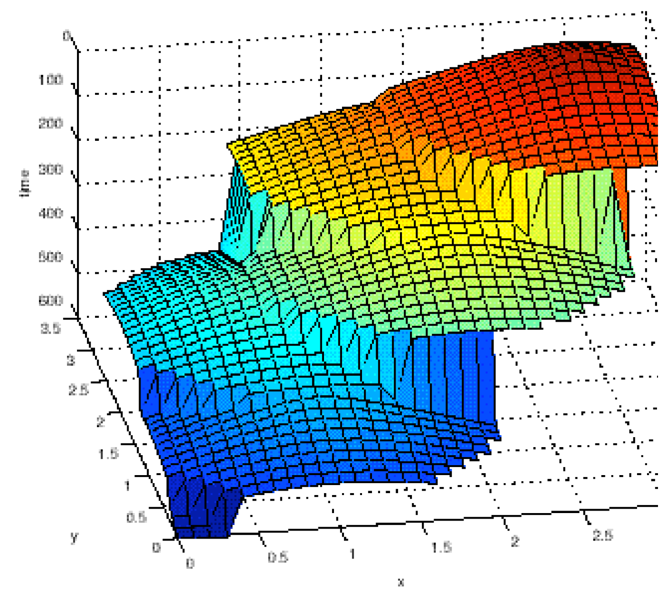
4 primitive actions, for ta
tiny steps up, down, left,

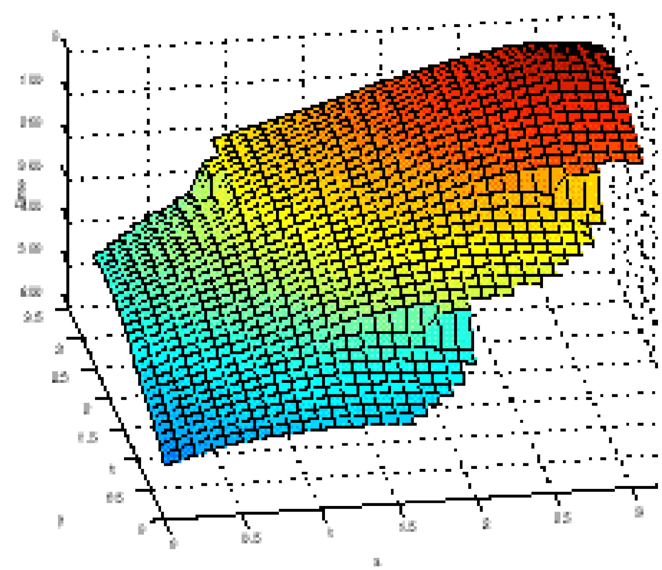
7 controllers for going str
to each one of the landm
from within a circular reg
where the landmark is vi

In this task, planning at the level of primitive actions is computationally intractable, we <u>need</u> the controllers

Termination Improvement for Landmarks Tas

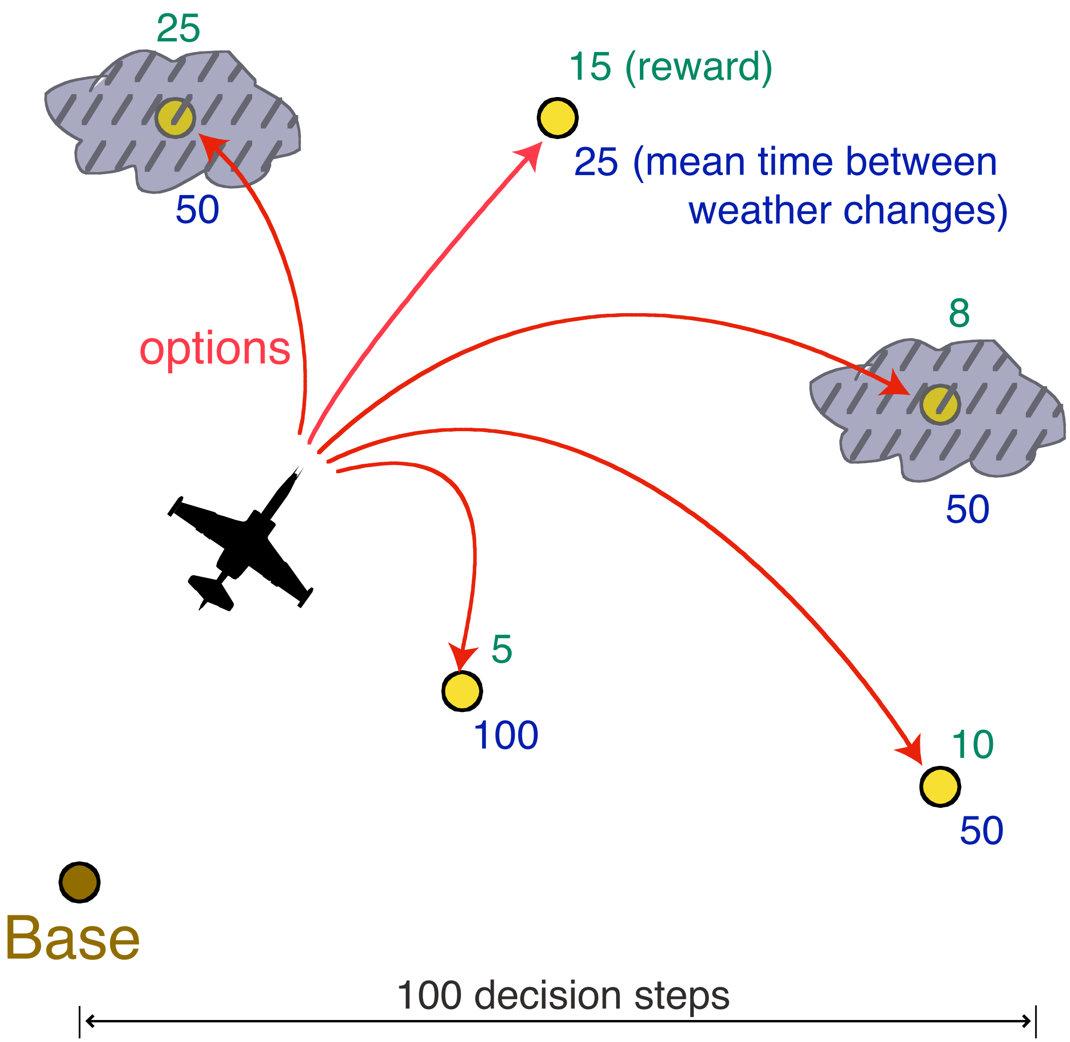






Allowing early termination based on models improves the value function at no additional cost!

Illustration: Reconnaissance Mission Planning (Problem)



- Mission: Fly over (observe) mos valuable sites and return to base
- Stochastic weather affects observability (cloudy or clear) of
- Limited fuel
- Intractable with classical optimal control methods
- Temporal scales:
  - Actions: which direction to fly no
  - Options: which site to head for
- Options compress space and tin
  - Reduce steps from ~600 to ~6
  - Reduce states from ~10^{11} to ~1(

$$Q^*_{O}(s,o) = r^o_s + \sum_{s'} p^o_{ss'} V^*_{O}(s')$$

any state (10^6)

sites only (6

Illustration: Reconnaissance Mission Planning (Results)

Temporal abstraction finds better approximation than static planner, with little more computation than SMDP planner

- SMDP planner:
  - Assumes options followed to completion
  - Plans optimal SMDP solution
- SMDP planner with re-evaluatio
  - Plans as if options must be follo completion
  - But actually takes them for only step
  - Re-picks a new option on every
- Static planner:
  - Assumes weather will not chang
  - Plans optimal tour among clear
  - Re-plans whenever weather cha

Intra-Option Learning Methods for Markov Options

Idea: take advantage of each fragment of experience

SMDP Q-learning:

- execute option to termination, keeping track of reward along the way
- at the end, update only the option taken, based on reward anc value of state in which option terminates

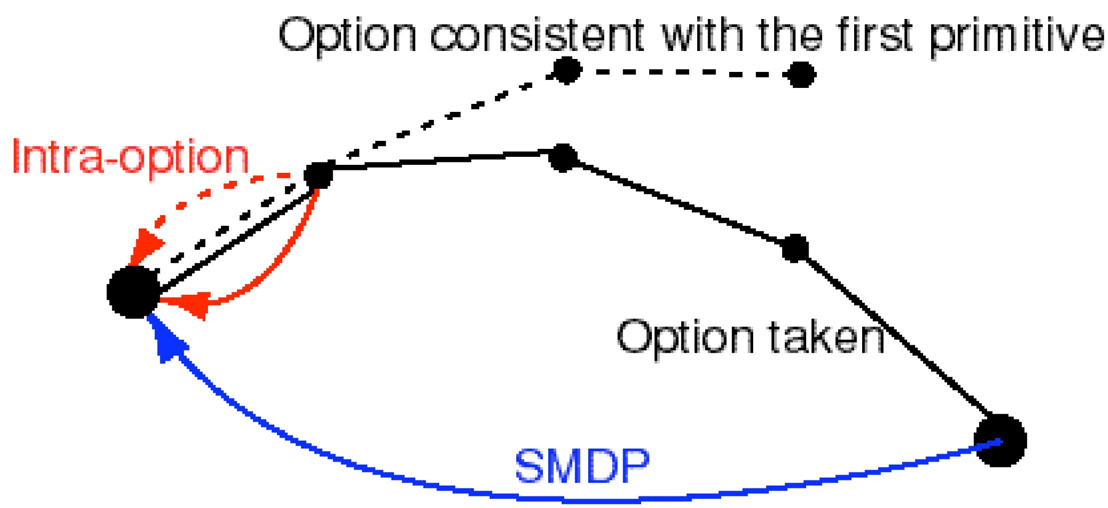
Intra-option Q-learning:

- after each primitive action, update all the options that could ha taken that action, based on the reward and the expected valu from the next state on

Proven to converge to correct values, under same assumptions as 1-step Q-learning

Intra-Option Learning Methods for Markov Options

Idea: take advantage of each fragment of experience

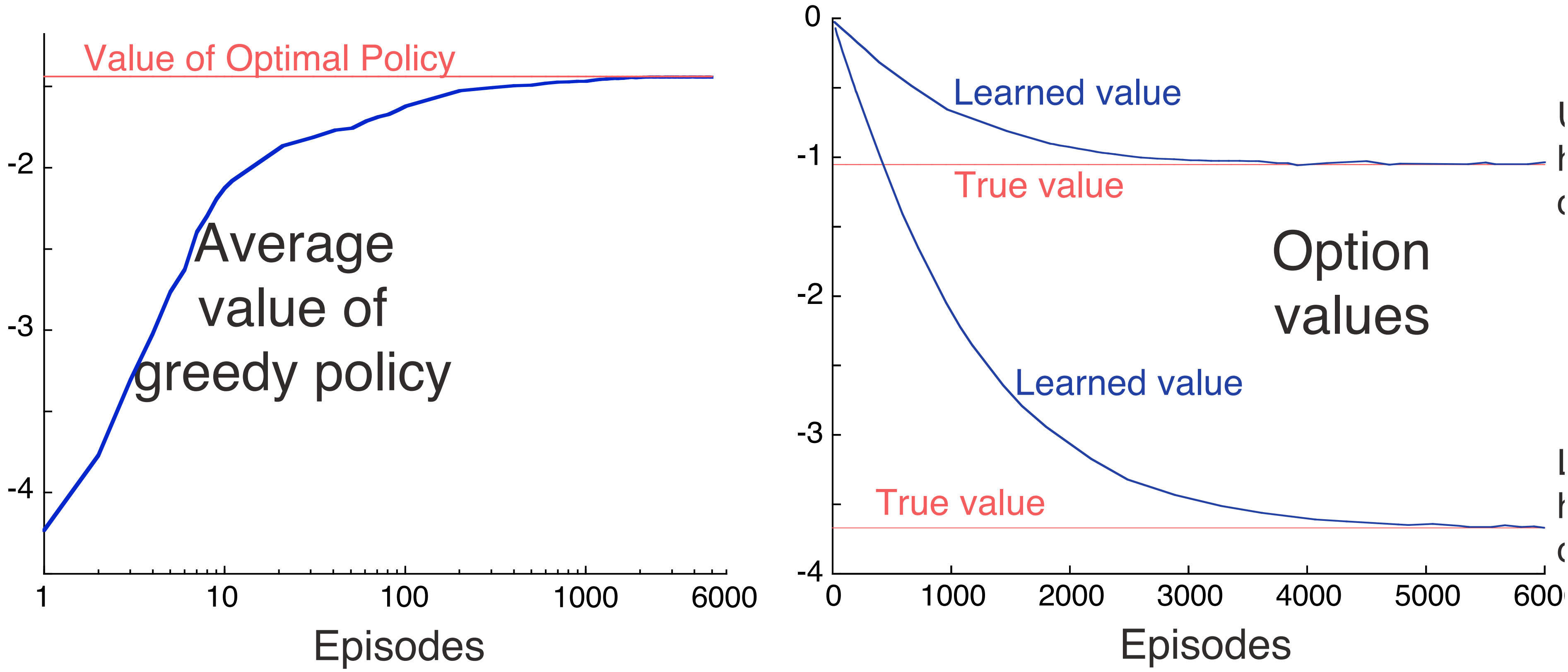


SMDP Learning: <u>execute option to termination</u>,then update only the option taken

Intra-Option Learning: <u>after each primitive action</u>, update all the opti that could have taken that action

Proven to converge to correct values, under same assumptions as 1-step Q-learning

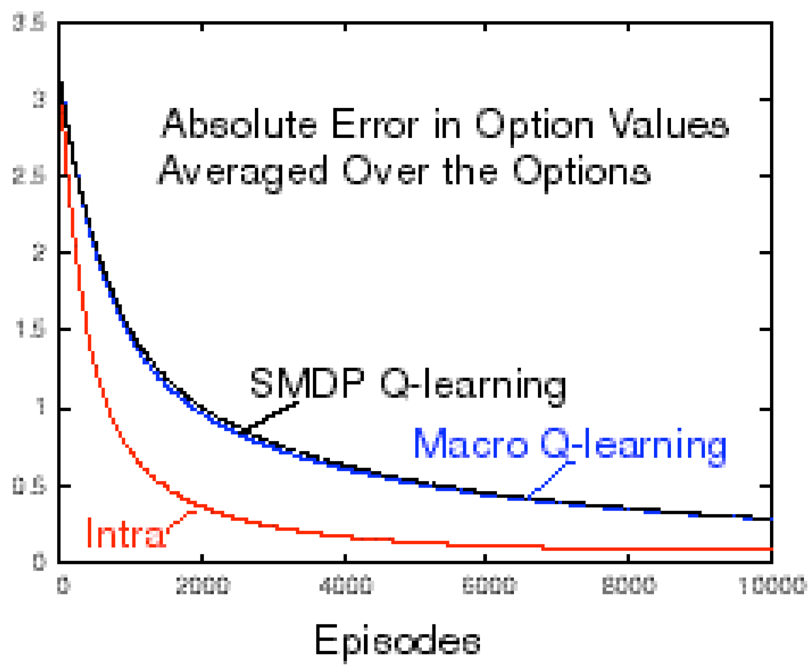
Example of Intra-Option Value Learni

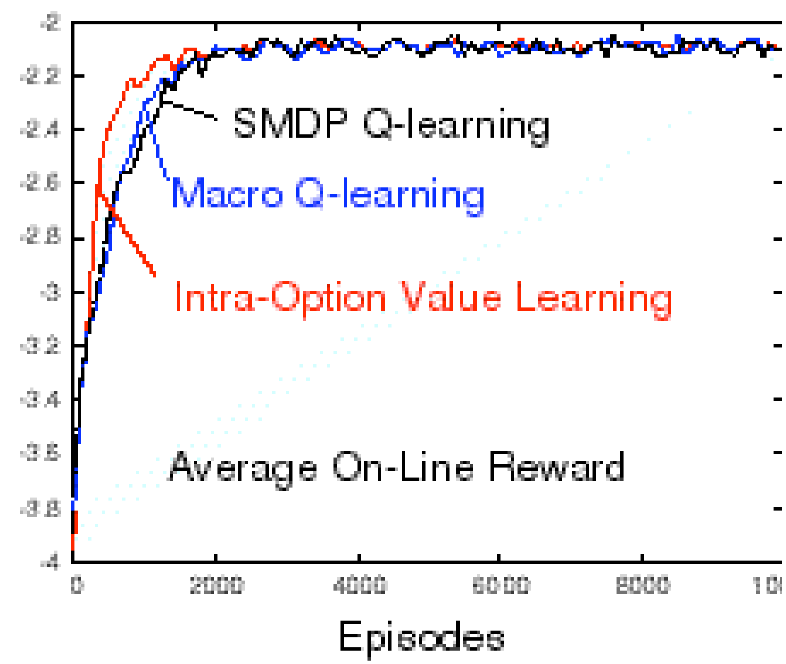


Random start, goal in right hallway, random actions

Intra-option methods learn correct values without ever taking the options! SMDP methods are not applicable he

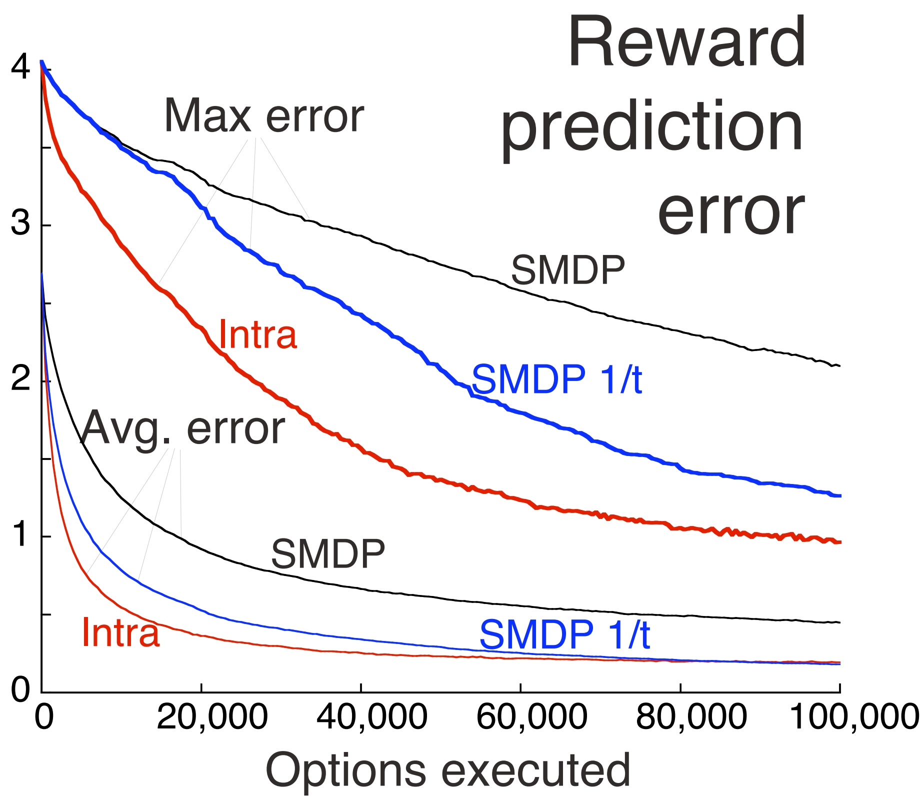
Intra-Option Value Learning Is Faste
Than SMDP Value Learning

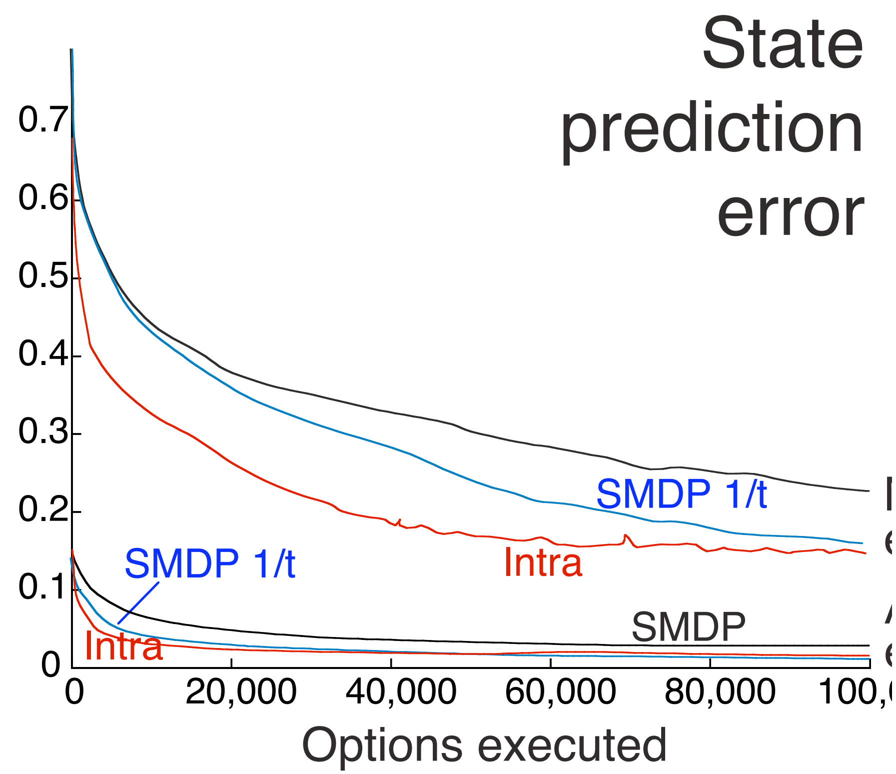




Random start, goal in right hallway, choice from A U H, 90% gree

Intra-Option Model Learning





Random start state, no goal, pick randomly among all options

Intra-option methods work much faster than SMDP metho

Tasks and Subgoals

It is natural to define options as solutions to subtasks

e.g. treat hallways as subgoals, learn shortest paths

We have defined subgoals as pairs: $<\mathcal{G}, g>$

$\mathcal{G} \subseteq \mathcal{S}$ is the set of states treated as subgoals

$g : \mathcal{G} \rightarrow \Re$ are their subgoal values (can be both good and bad)

Each subgoal has its own set of value functions, e.g.:

$$V_g^o(s) = E\{r_1 + \gamma r_2 + \ldots + \gamma^{k-1} r_k + g(s_k) \mid s_0 = s, o, s_k \in \mathcal{G}\}$$

$$V_g^*(s) = \max_o V_g^o(s)$$

Policies inside options can be learned from subgoals, in intra-option, off-policy manner.

Between MDPs and SMDPs

- **Termination Improvement**

  Improving the value function by changing the termination conditions of options

- **Intra-Option Learning**

  Learning the values of options in parallel, without executing the[illegible] to termination

  Learning the models of options in parallel, without executing them to termination

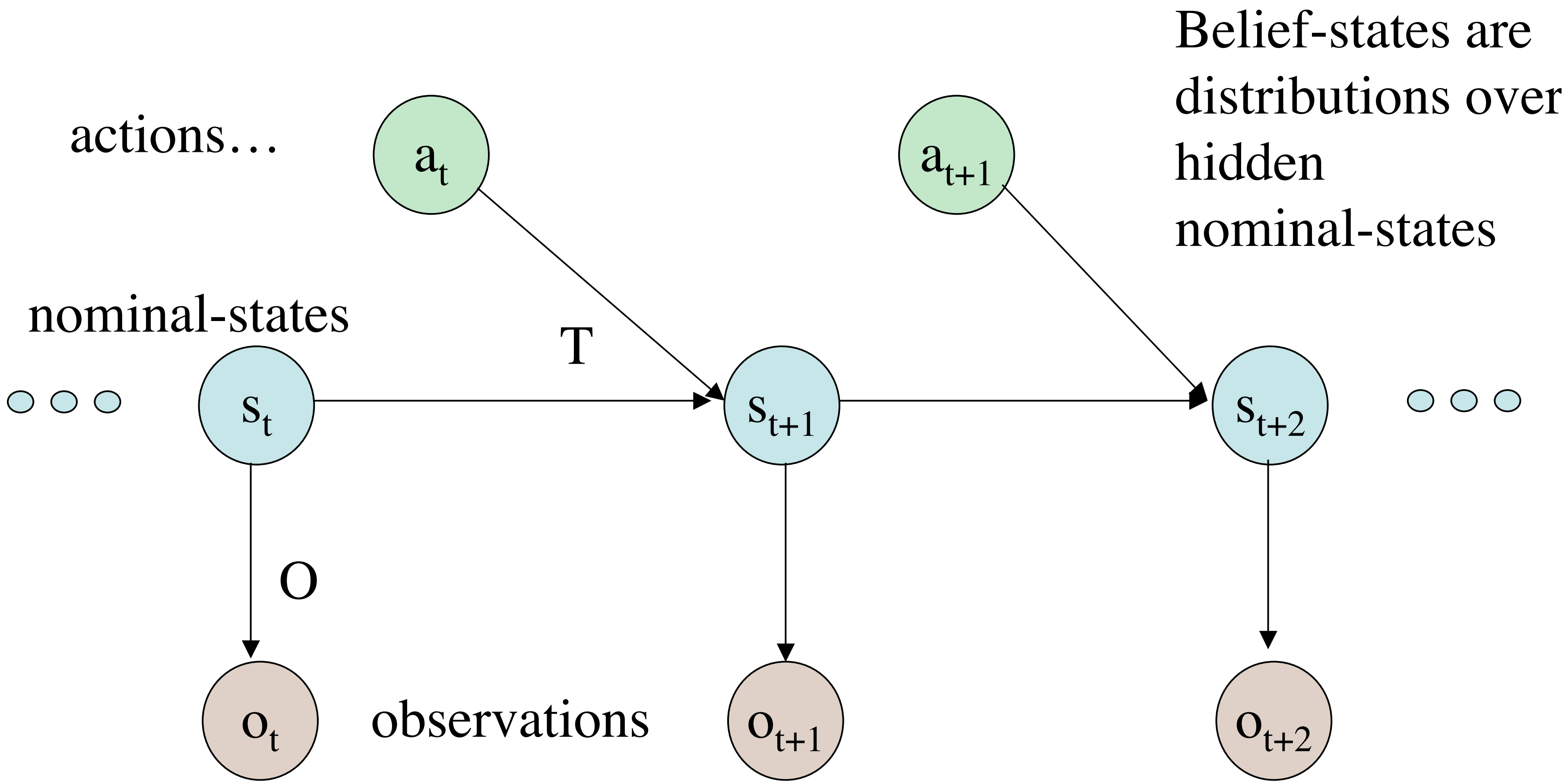
- **Tasks and Subgoals**

  Learning the policies inside the options

Summary: Benefits of Options

- Transfer
  - Solutions to sub-tasks can be saved and reused
  - Domain knowledge can be provided as options and subgoals
- Potentially much faster learning and planning
  - By representing action at an appropriate temporal scale
- Models of options are a form of knowledge representati
  - Expressive
  - Clear
  - Suitable for learning and planning
- Much more to learn than just one policy, one set of valu
  - A framework for “constructivism” – for finding models of the world that are useful for rapid planning and learning

POMDPs



Belief-states are distributions over hidden nominal-states
actions…
a_t
a_{t+1}
nominal-states
T
s_t
s_{t+1}
s_{t+2}
O
o_t
observations
o_{t+1}
o_{t+2}

POMDPs...

- n underlying nominal or *hidden* states
- b(h) is a belief-state at history h
- T^a : transition probabilities among hidden states fc action a
- $O^{ao}(ii)$ is the probability of observation o on actio a in state i
- $b(hao) = b(h)T^aO^{ao}/Z = b(h)\ B^{ao}/Z$

Rethinking State

(Predictive State Representations or PSRs)
(TD-Nets)

Initiated by Littman, Sutton & Singh
…Singh's group at Umich
…Sutton's group at UAlberta

Go to NIPS05PSRTutorial

Rethinking Reward

(Intrinsically Motivated RL)

By Singh, Barto & Chentanez
… Singh's group at Umich
… Barto's group at UMass

Go to NIPS05IMRLTutorial

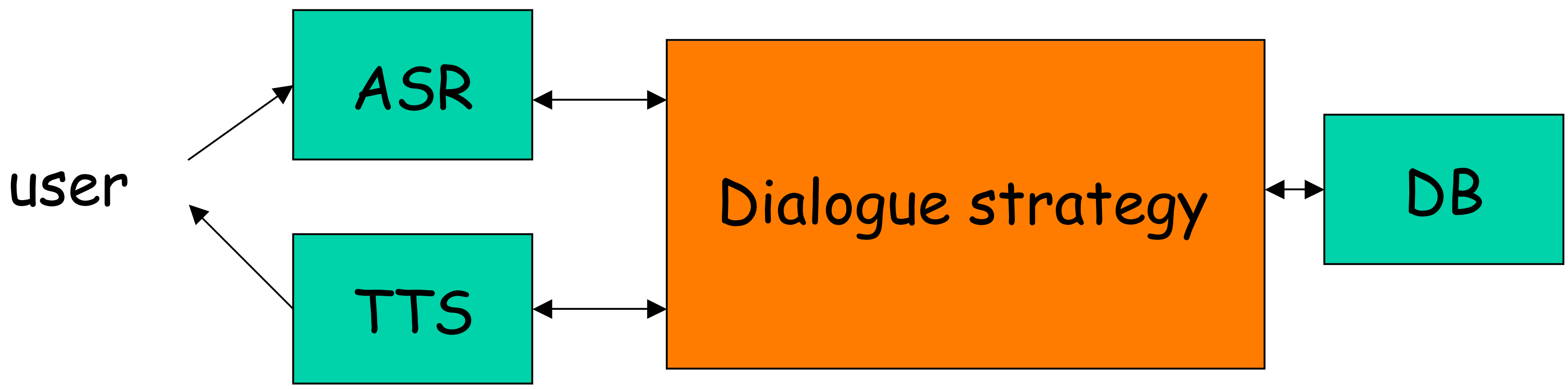
Applications of RL

List of Applications

- Robotics
  - Navigation, Robosoccer, walking, juggling, ...
- Control
  - factory processes, admission control in telecomm, resource control in multimedia networks, ...
- Games
  - Backgammon, Chess, Othello, ...
- Operations Research
  - Warehousing, transportation, scheduling, ...
- Others
  - Adaptive treatment design, biological modeling, ...

RL applied to HCI

Spoken Dialogue Systems



Sample Dialogue

- S1: Welcome to NJFun. How may I help you?

U1: I'd like to find um winetasting in Lambertville in the morning.
(ASR output: I'd like to find out wineries the in the Lambertville in the morning.)

S2: Did you say you are interested in Lambertville?

U2: Yes

S3: Did you say you want to go in the morning?

U3: Yes.

S4. I found a winery near Lambertville that is open in the morning. It is […] Please give me feedback by saying "good", "so-so" or "bad".

U4: Good

NJFun

- Spoken dialogue system providing telephone access to a DB of activities in NJ
- Want to obtain 3 attributes:
  activity type (e.g., wine tasting)
  location (e.g., Lambertville)
  time (e.g., morning)
- Failure to bind an attribute: query DB with don't-care

NJFun

- Spoken dialogue system providing telephone access to a DB of activities in NJ
- Want to obtain 3 attributes:
  activity type (e.g., wine tasting)
  location (e.g., Lambertville)
  time (e.g., morning)
- Failure to bind an attribute: query DB with don't-care

NJFun

- Spoken dialogue system providing telephone access to a DB of activities in NJ
- Want to obtain 3 attributes:
  activity type (e.g., wine tasting)
  location (e.g., Lambertville)
  time (e.g., morning)
- Failure to bind an attribute: query DB with don't-care

Approximate State Space

N.B. Non-state variables record attribute values;
state does not condition on previous attributes!

Action Space

- Initiative (when T = 0):
  open or constrained prompt?
  open or constrained grammar?
  N.B. might depend on H, A,...
- Confirmation (when V = 1)
  confirm or move on or re-ask?
  N.B. might depend on C, H, A,...
- Only allowed "reasonable" actions
- Results in 42 states with (binary) choices
- Small state space, large strategy space

The Experiment

- Designed 6 specific tasks, each with web survey
- Gathered 75 internal subjects
- Split into training and test, controlling for M/F, native/non-native, experienced/inexperienced
- 54 training subjects generated 311 dialogues
- Exploratory training dialogues used to build MDP
- Optimal strategy for objective TASK COMPLETION computed and implemented
- 21 test subjects performed tasks and web surveys for modified system generated 124 dialogues
- Did statistical analyses of performance changes

Estimating the MDP

Initial system utterance

Initial user utterance

$s_1 \rightarrow u_1 \rightarrow s_2 \rightarrow u_2 \rightarrow s_3 \rightarrow u_3 \rightarrow \ldots$

+ system logs

Actions have prob. outcomes

$a_1 \rightarrow e_1 \rightarrow a_2 \rightarrow e_2 \rightarrow a_3 \rightarrow e_3 \rightarrow \ldots$

estimate transition probabilities...

P(next state | current state & action)

...and rewards...

R(current state, action)

Models population of users

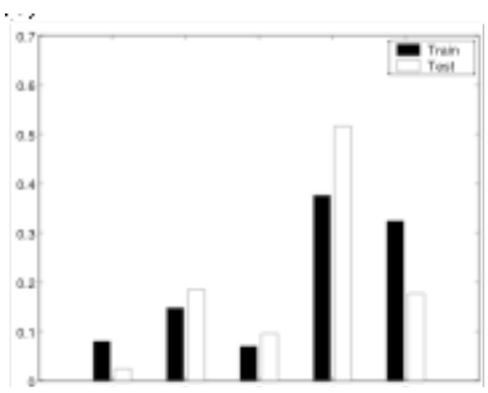
...from set of exploratory dialogues

Reward Function

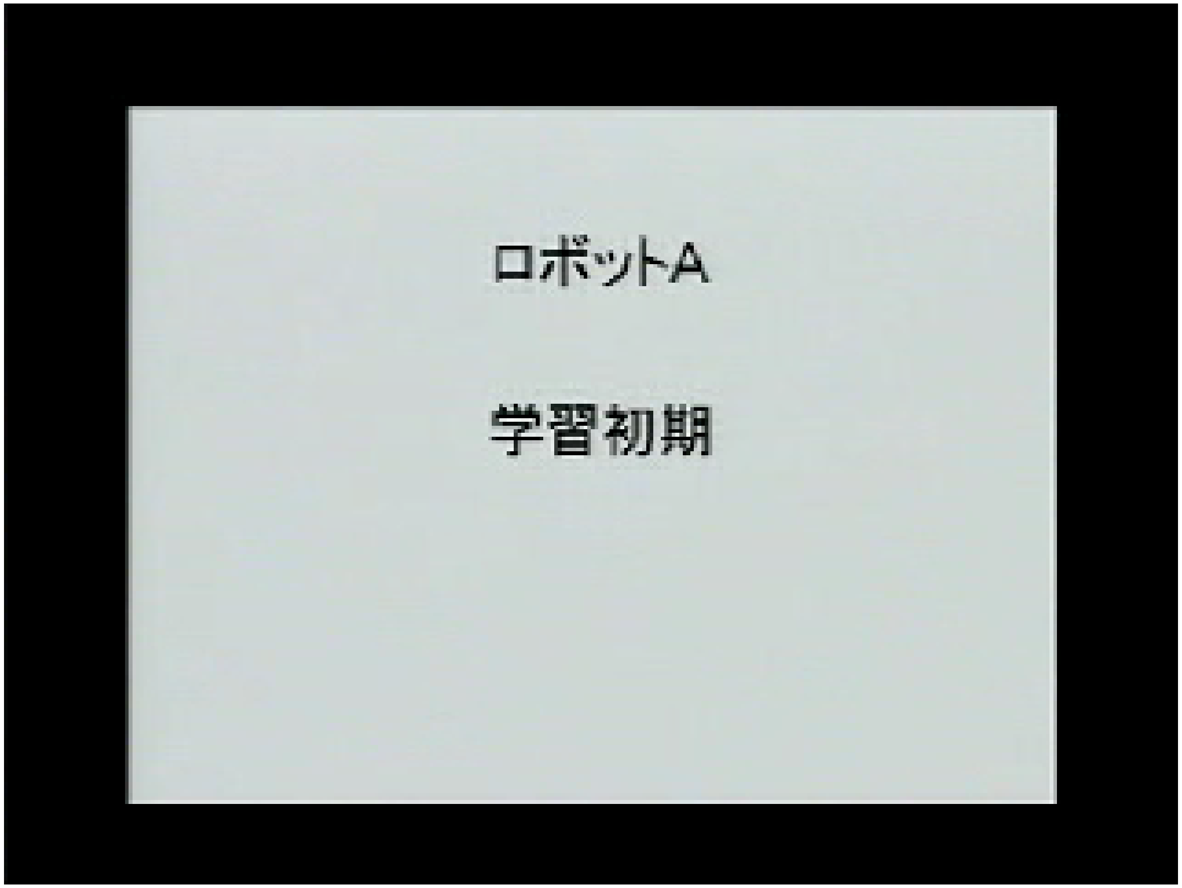
- Objective task completion:
  -1 for an incorrect attribute binding
  0,1,2,3 correct attribute bindings
- Binary version:
  1 for 3 correct bindings, else 0
- Other reward measures: perceived completion, user satisfaction, future use, perceived understanding, user understanding, ease of use
- Optimized for objective task completion, but predicted improvements in some others

Main Results

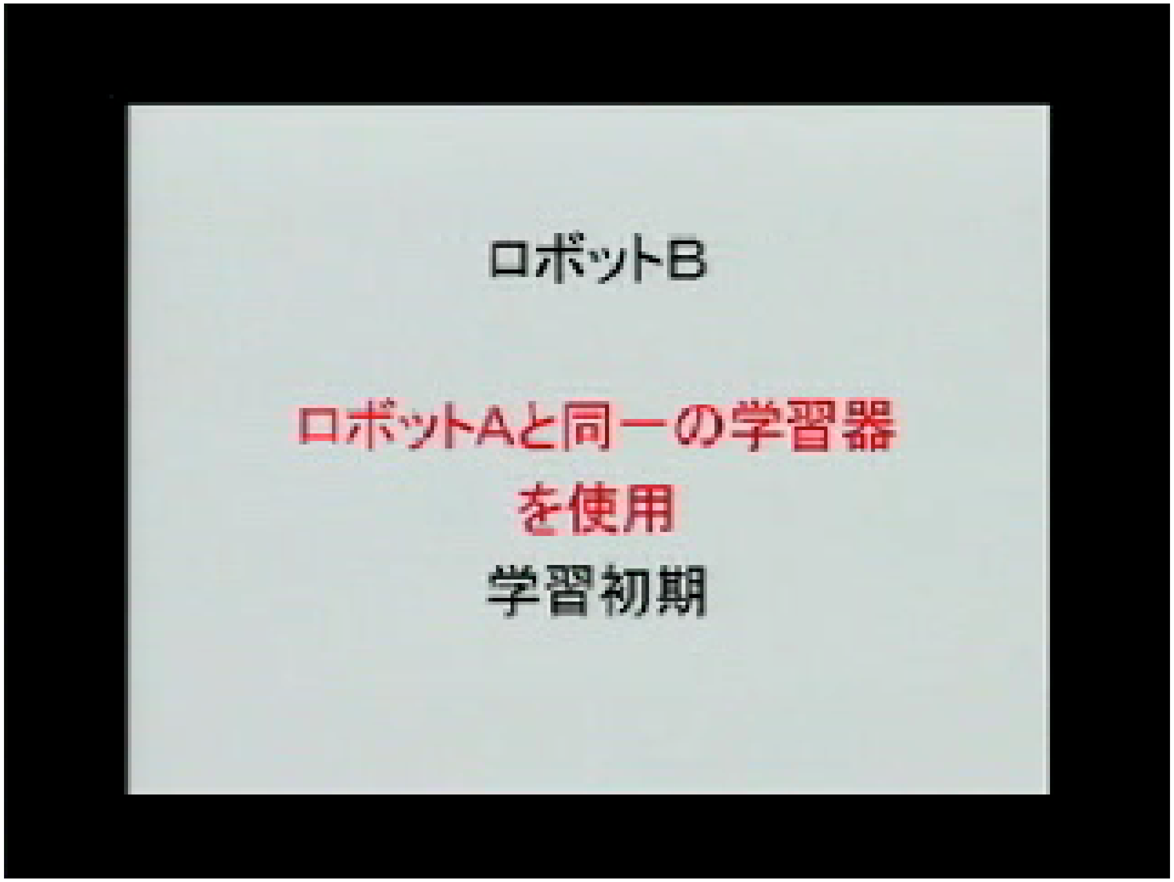
- Objective task completion:
  train mean ~ 1.722, test mean ~ 2.176
  two-sample t-test p-value ~ 0.0289
- Binary task completion:
  train mean ~ 0.515, test mean ~ 0.635
  two-sample t-test p-value ~ 0.05
- Outperformed hand-built policies



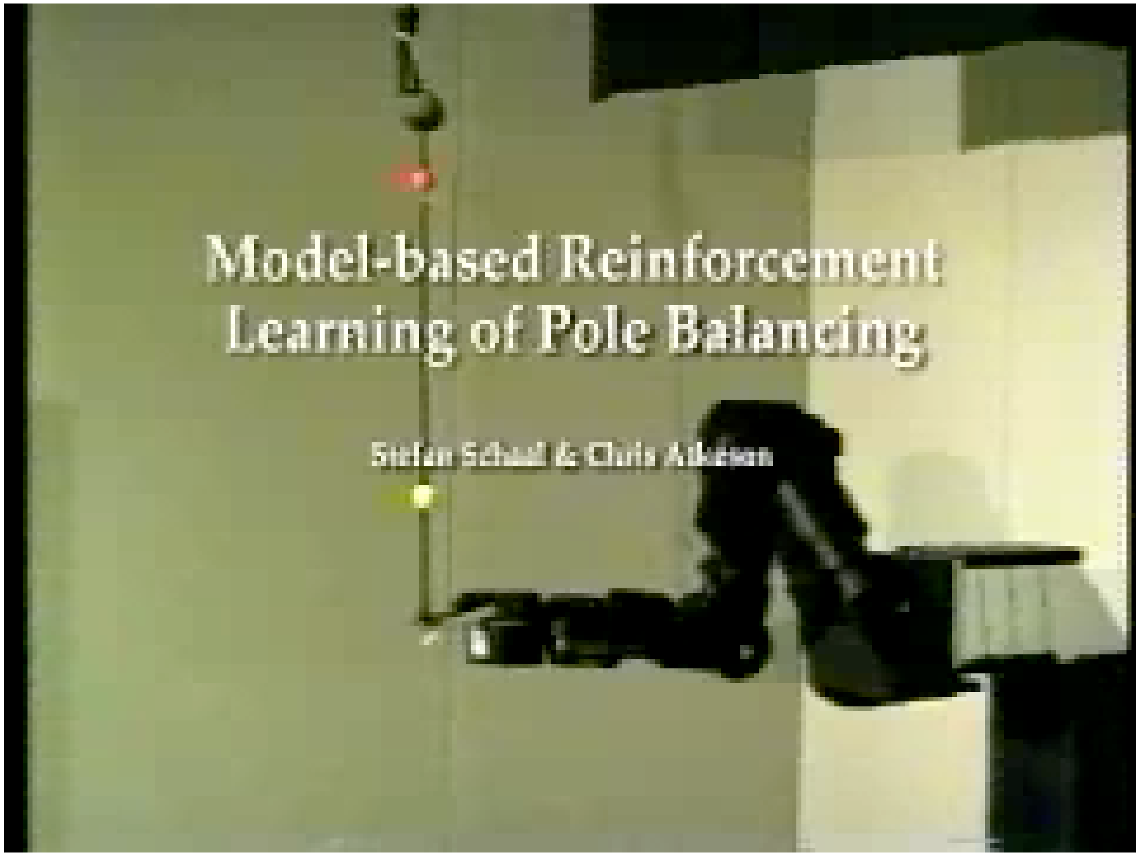
move to the middle



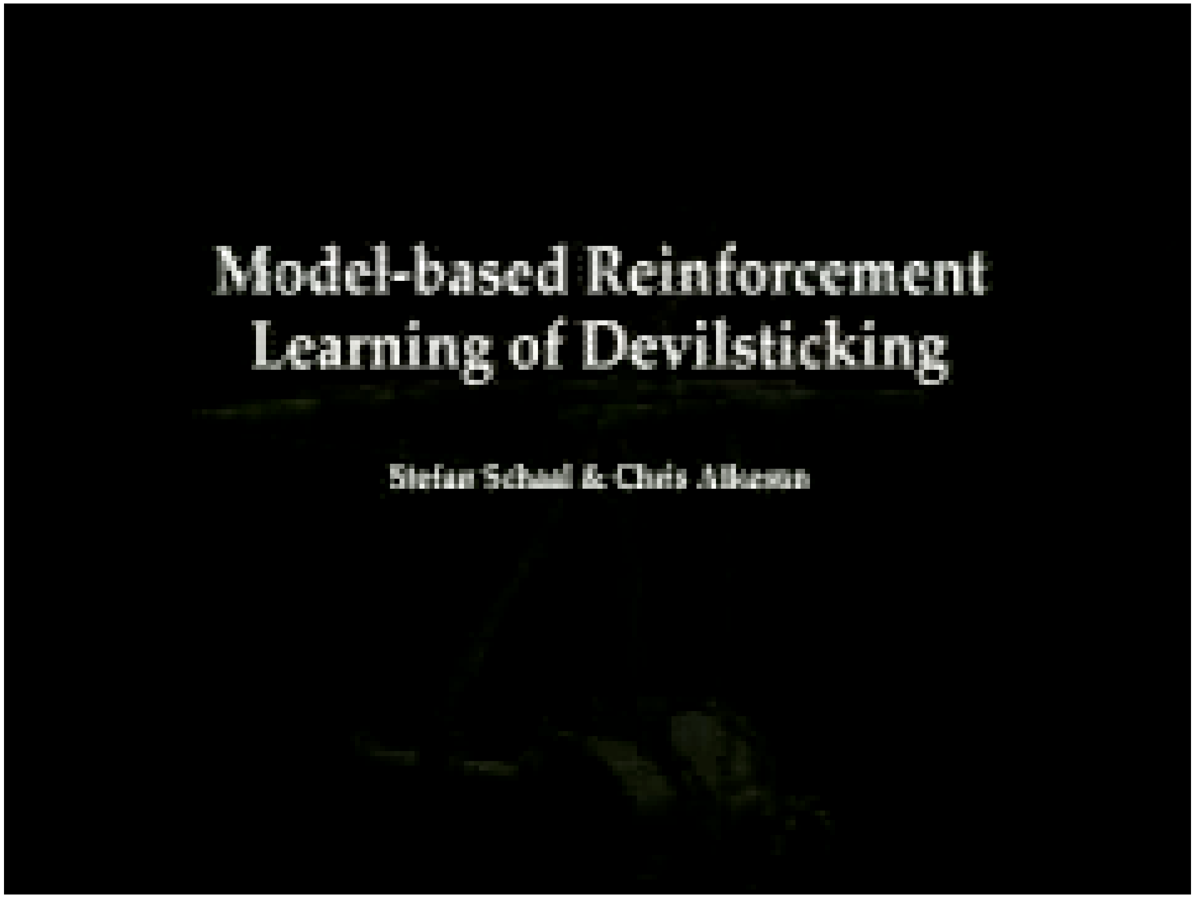
by
Hajime Kimura



by
Hajime Kimura



by
Stefan Schaal
&
Chris Atkeson



by
Stefan Schaal
&
Chris Atkeson

by
Sebastian Thrun
&
Colleagues

Textbook References

- *Reinforcement Learning: An Introduction*

by Richard S. Sutton & Andrew G. Barto

MIT Press, Cambridge MA, 1998.

- *Neuro-Dynamic Programming*

by Dimitri Bertsekas & John Tsitsiklis

Athena Scientific, Belmont MA, 1996.

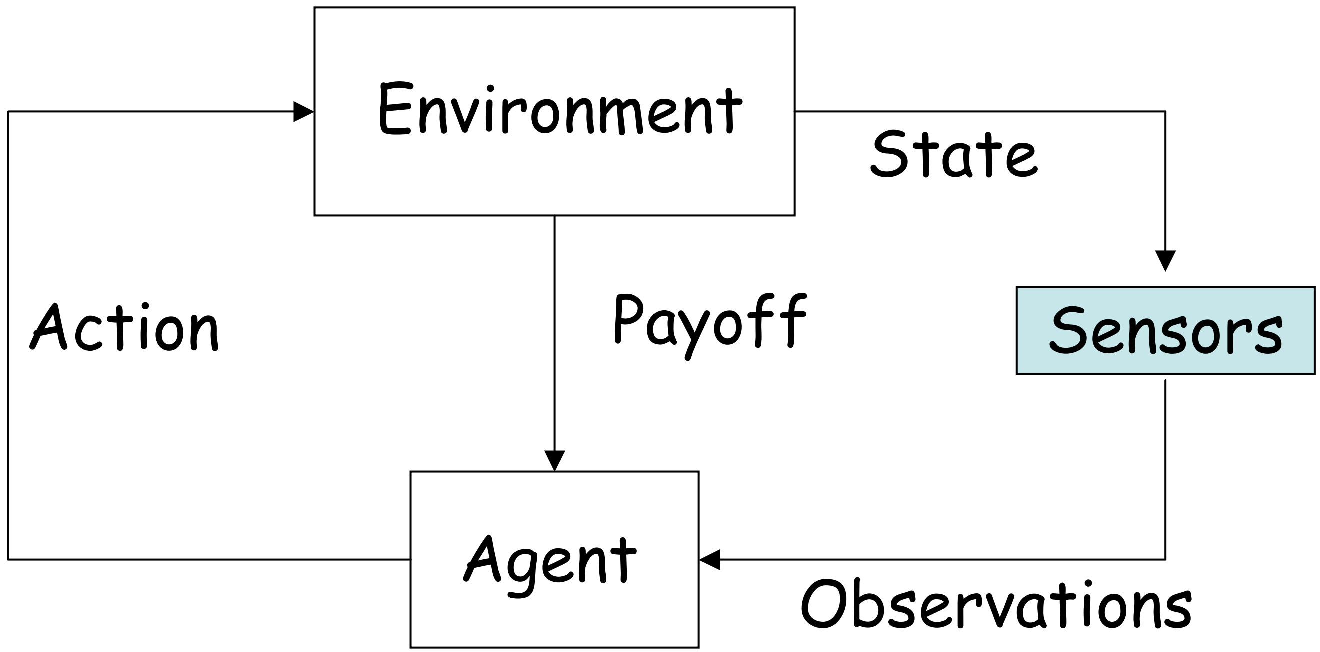
Myths of RL

- RL is TD or perhaps Q-learning
- RL is model-free
- RL is table lookup
- RL is slow
- RL does not work well with function approximation
- POMDPs are hard for RL to deal with
- RL is about learning optimal policies

Twiki pages on RL

- Myths of RL
- http://neuromancer.eecs.umich.edu/cgi-bin/twiki/view/Main/MythsofRL
- Successes of RL
- http://neuromancer.eecs.umich.edu/cgi-bin/twiki/view/Main/SuccessesOfRL
- Theory of RL
- http://neuromancer.eecs.umich.edu/cgi-bin/twiki/view/Main/TheoryOfRL
- Algorithms of RL
- http://neuromancer.eecs.umich.edu/cgi-bin/twiki/view/Main/AlgorithmsOfRL
- Demos of RL
- http://neuromancer.eecs.umich.edu/cgi-bin/twiki/view/Main/DemosOfRL

RL Abstractly...



Goal: maximize expected payoff over some time horiz

Life is an *optimal control* problem!