Reinforcement Learning: A Tutorial

Satinder Singh

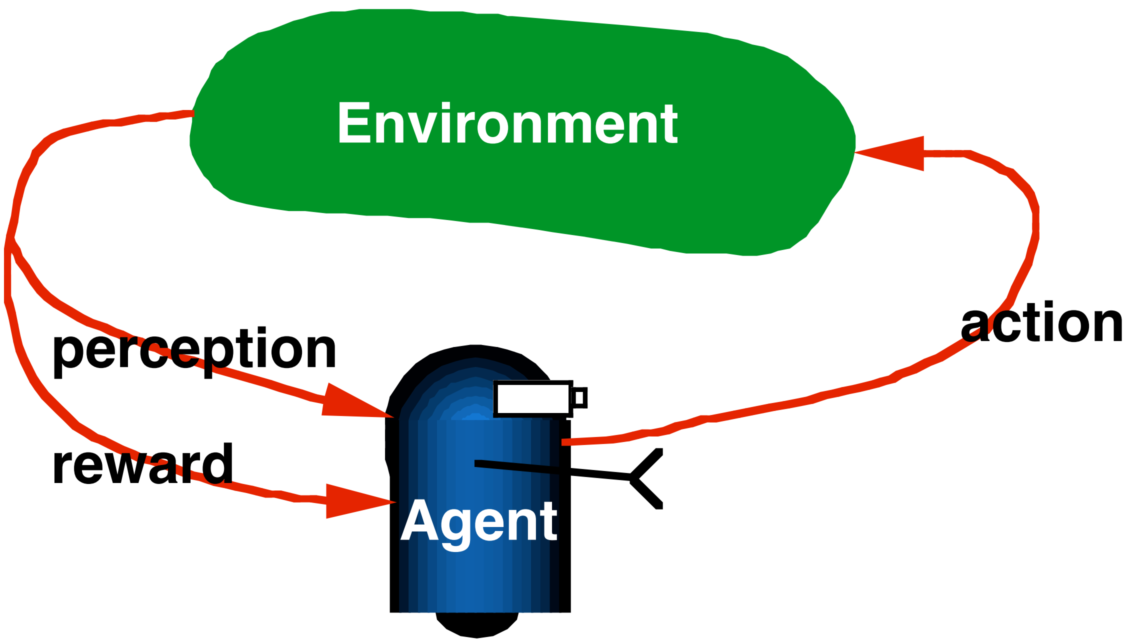
Computer Science & Engineering
University of Michigan, Ann Arbor

http://www.eecs.umich.edu/~baveja/ICML06Tutorial/

Outline

- What is RL?
- Markov Decision Processes (MDPs)
  - Planning in MDPs
  - Learning in MDPs
  - Function Approximation and RL
- Partially Observable MDPs (POMDPs)
- Beyond MDP/POMDPs

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 uncertain

RL (another view)

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

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 & can be arbitrarily uninformative

Agent has partial knowledge about its environment

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

Some 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

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

Discrete time
Discrete observations
Discrete actions

Markov Decision Processes (MDPs)

Markov Assumption

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

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 i

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

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

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

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)

$V^{\pi} : S \longrightarrow \Re$ $\qquad$ $V^* : S \longrightarrow \Re$

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!

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

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

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

Bellman Optimality Equations

Optimal Control

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

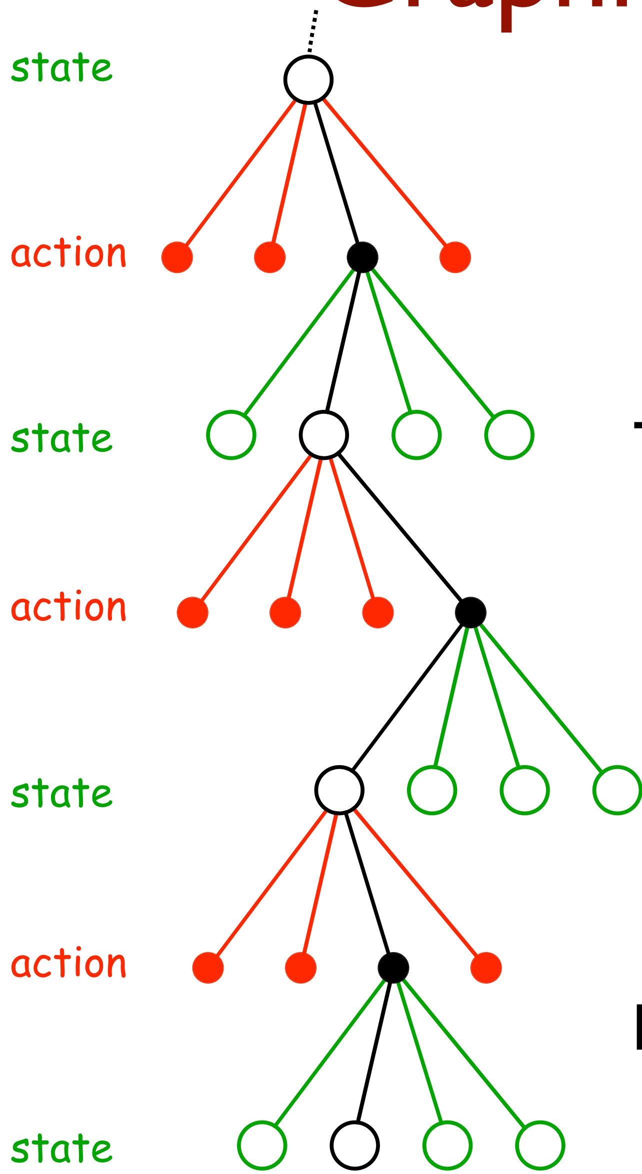
$$\forall s \in S, \pi^*(s) = argmax_{a \in A}[R(s,a) + \sum_{s' \in S} P(s'|s,a)V^*(s')]$$

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

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

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

Graphical View of MDPs



Temporal Credit Assignment Problem!!

Learning from *Delayed Reward*

Distinguishes RL from other forms of ML

Planning & Learning
in
MDPs

Planning in MDPs

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

Value Iteration (Policy Evaluation)

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

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

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

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,...

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

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

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,...

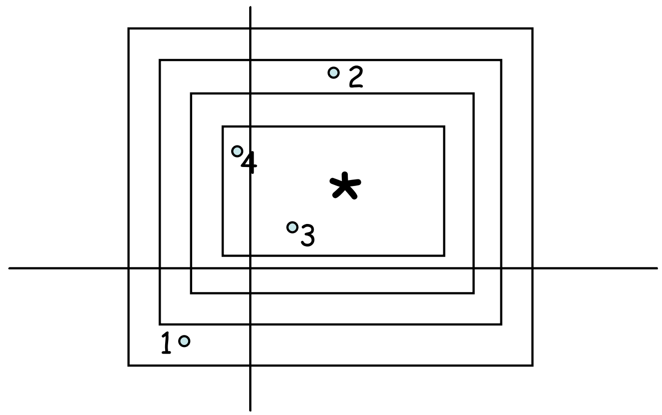
$$\forall s \in S,\ V_{k+1}(s) = \max_{a \in A}[R(s,a) + \sum_{s' \in S} P(s'|s,a)V_k(s')]$$

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

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*



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

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

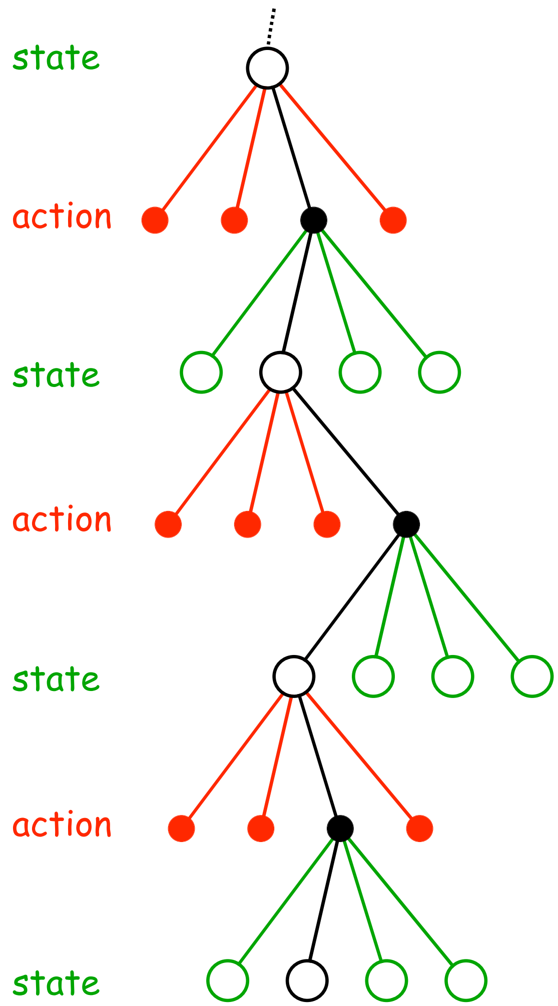
Contractions!

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 MDPs

- 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 $\langle \, s_k \; a_k \; r_k \; s_{k+1} \, \rangle$

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

Big table of Q-values?

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

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: $E\{ r_k + \gamma \max_b Q_{old}(s_{k+1},b)\} = R(s_k) + \gamma[\sum_{j\in S} P(j|s_k,a_k) \max_{b\in A} Q_{old}(j,b)]$

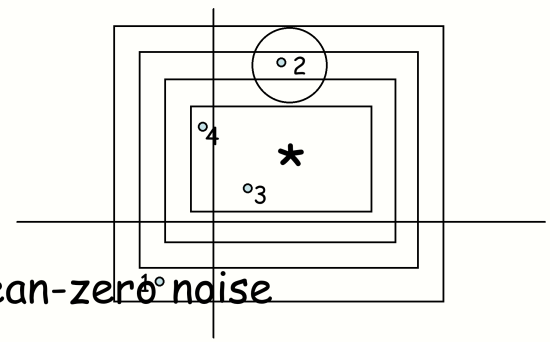
Q-learning is a stochastic approximation version of Q-value iteration! That is,

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 s

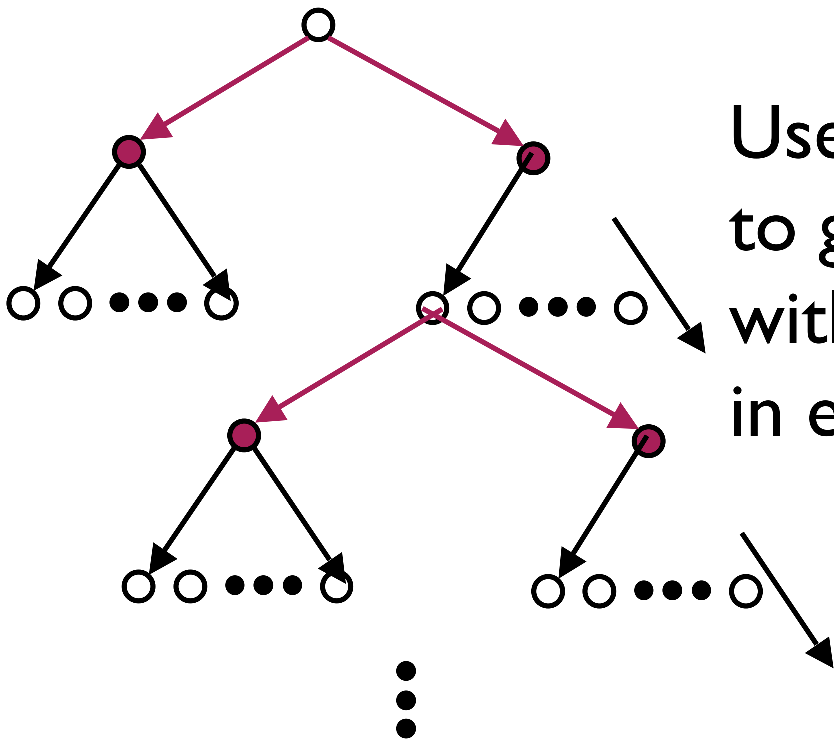
Start at state s and execute the policy for a long trajectory and compute the empirical discounted return

Do this several times and average the returns across trajectories

How many trajectories?

Unbiased estimate whose variance improves with n

Sparse Sampling



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

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

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

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

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

  A single trajectory:

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

TD(λ)

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

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)

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

$r_0 + \gamma V(s_1)$

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)

$$
\begin{array}{lllllllll}
 & & r_0 & r_1 & r_2 & r_3 & \ldots & r_k & r_{k+1} \quad \ldots \\
w_0 & e_0: & r_0 + \gamma V(s_1) & & & & & & \\
w_1 & e_1: & r_0 + & \gamma r_1 + \gamma^2 V(s_2) & & & & & \\
w_2 & e_2: & r_0 + & \gamma r_1 + & \gamma^2 r_2 & + \gamma^3 V(s_3) & & & \\
 & & & & \vdots & & & & \\
w_{k-1} & e_{k-1}: & r_0 + & \gamma r_1 + & \gamma^2 r_2 & + \gamma^3 r_3 + & \ldots \gamma^{k-1} r_{k-1} + \gamma^k V(s_k) & & \\
 & & & & \vdots & & & & \\
w_\infty & e_\infty: & r_0 + & \gamma r_1 + & \gamma^2 r_2 & + \gamma^3 r_3 + & \ldots \gamma^k r_k + \gamma^{k+1} r_{k+1} + \ldots & &
\end{array}
$$

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

TD(λ)

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

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

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

TD(λ)

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

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

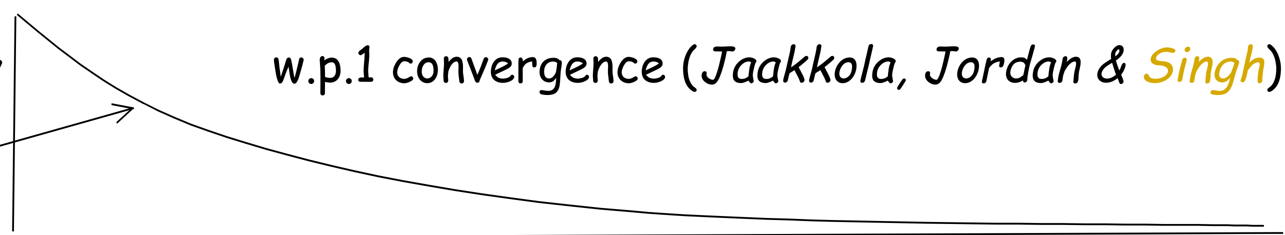
$\Delta_1 \qquad r_1 + \gamma V(s_2) - V(s_{1)}$

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

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

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

eligibility trace



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

Bias-Variance Tradeoff

decreasing bias

increasing variance

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

$$e_0: \quad r_0 + \gamma V(s_1)$$

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

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

$$\vdots$$

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

$$\vdots$$

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

TD(λ)

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

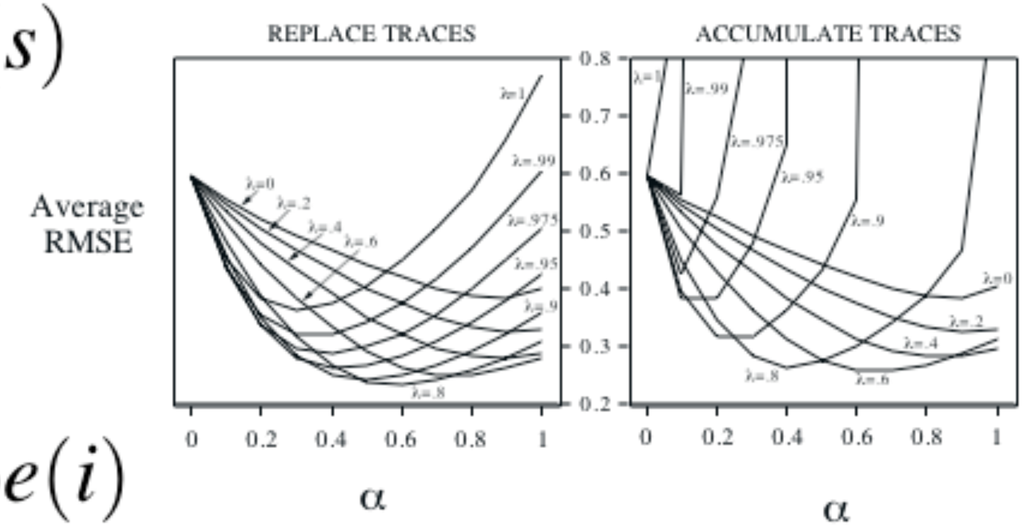
$$\delta \leftarrow r + \gamma V(s') - V(s)$$

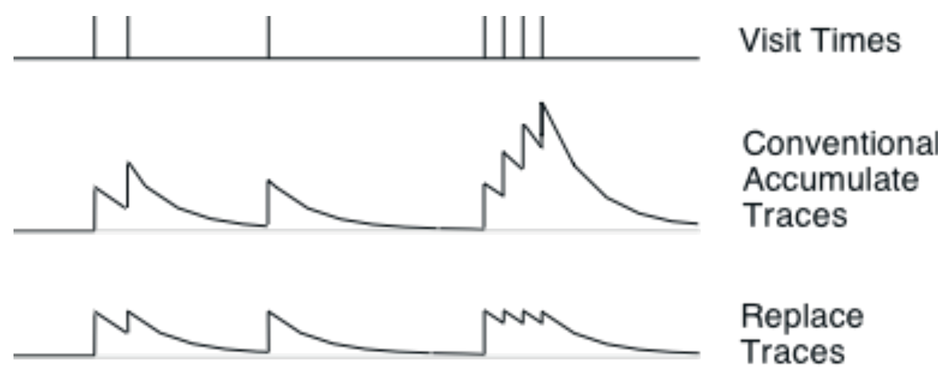
$$e(s) \leftarrow e(s) + 1$$

$$\forall i \in S:$$

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

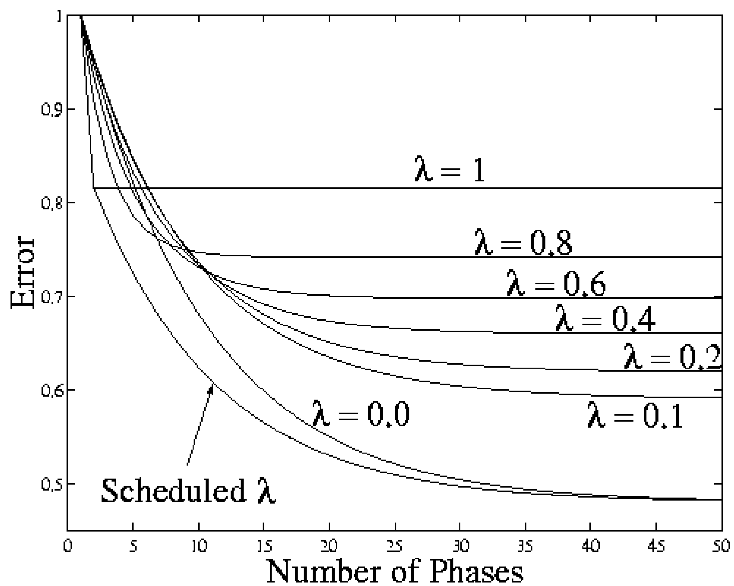
$$e(i) \leftarrow \gamma \lambda e(i)$$





Bias-Variance Tradeoff

Constant step-size



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

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

(an increasing function of λ)

Rate of convergence is b_λ^t (exponential)

b_λ is a decreasing function of λ

Intuition: start with large λ and then decrease over time

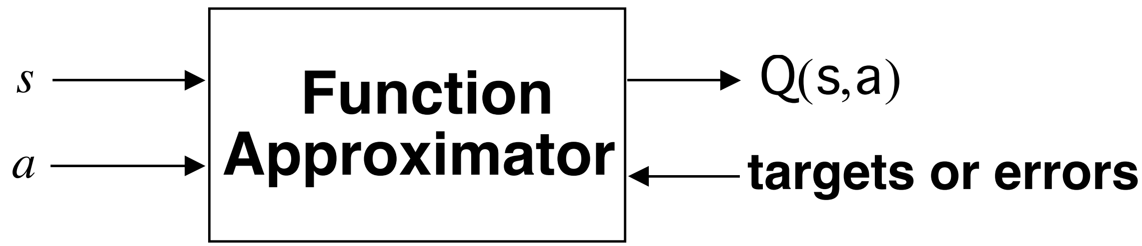
Kearns & Singh, 2000

Near-Optimal Reinforcement Learning in Polynomial Time

(solving the exploration versus exploitation dilemma)

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:

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

estimated value

Linear in the Parameters FAs

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

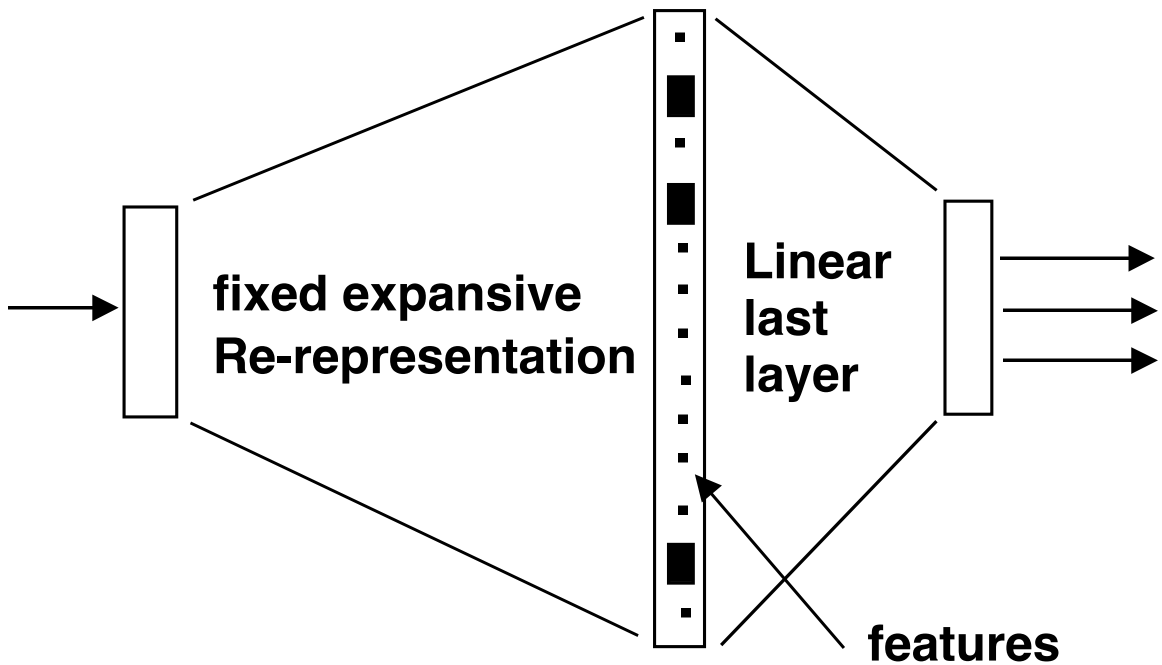
Each state s represented by a feature vector $\vec{\phi}_s$

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

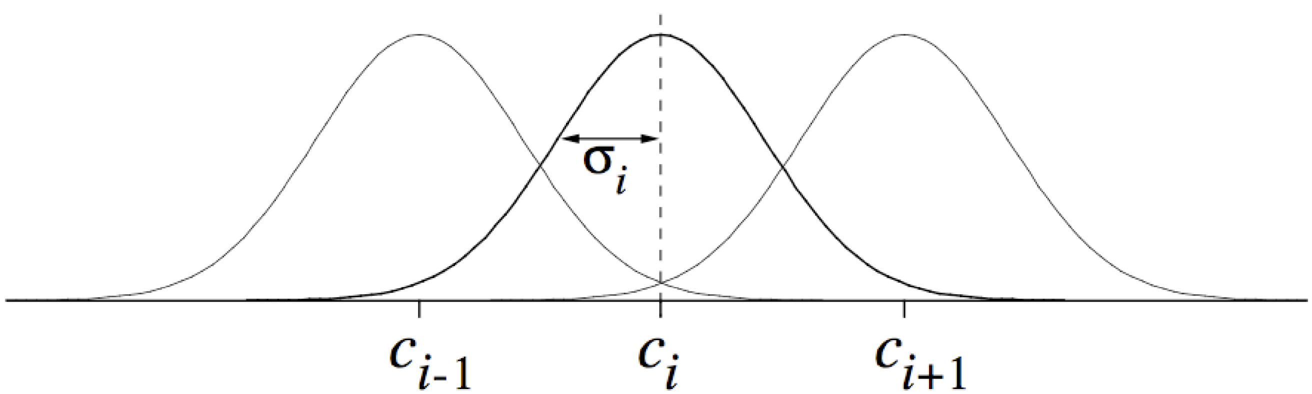


Coarse: Large receptive fields
Sparse: Few features present at one time

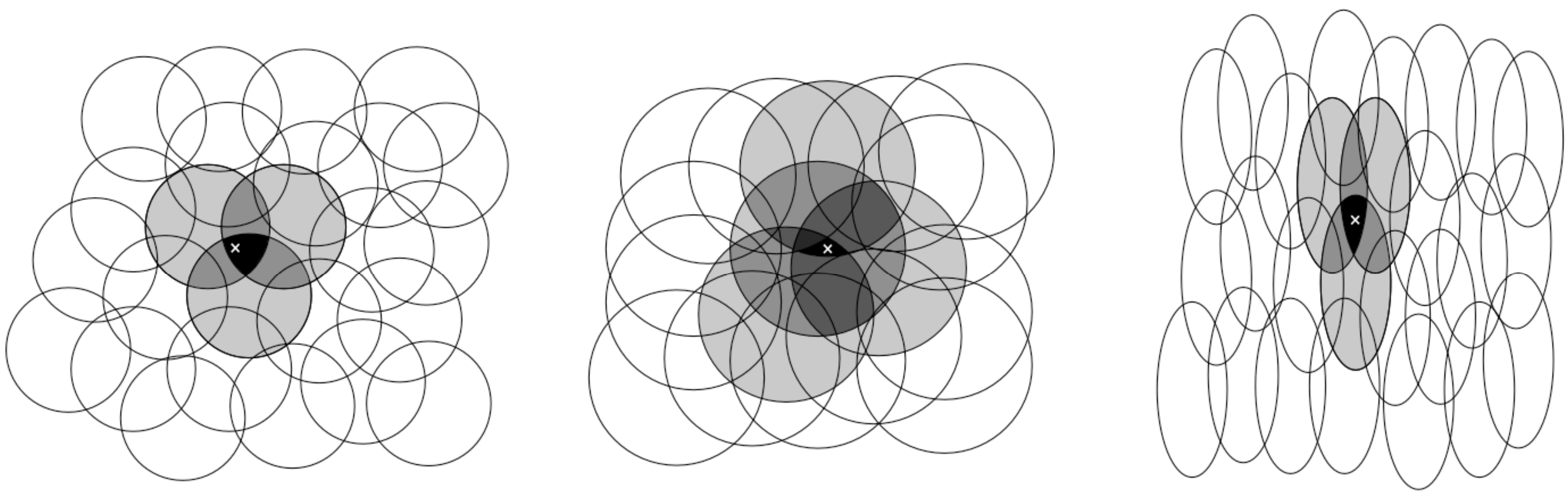
Radial Basis Functions (RBFs)

e.g., Gaussians

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

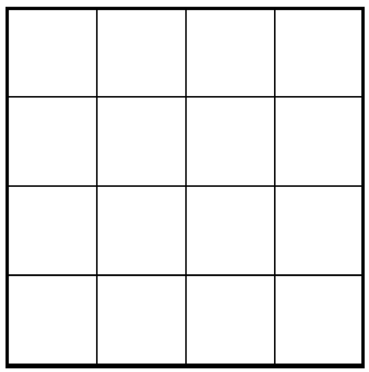


Shaping Generalization in Coarse Coding

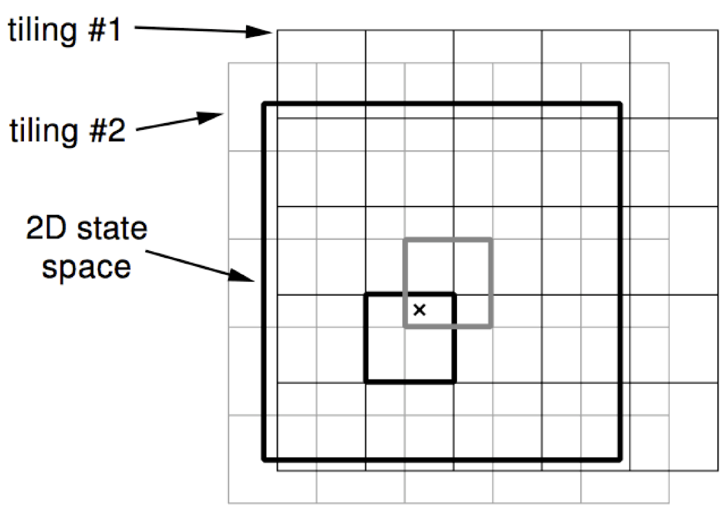


a) Narrow generalization b) Broad generalization c) Asymmetric generalization

Tile Coding



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

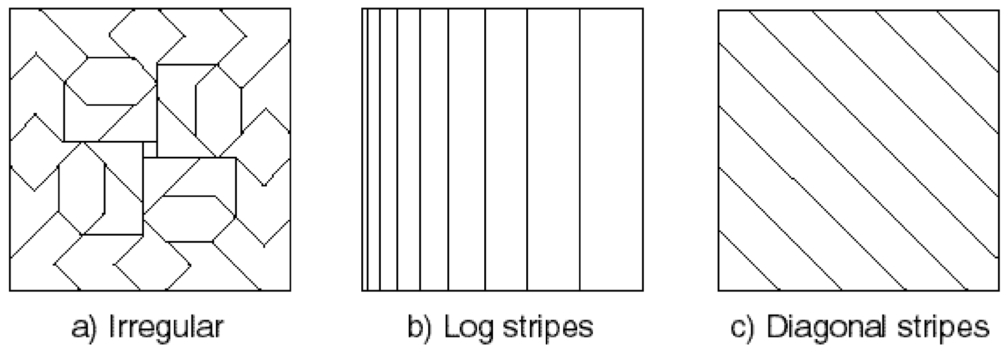


Shape of tiles ⇒ Generalization

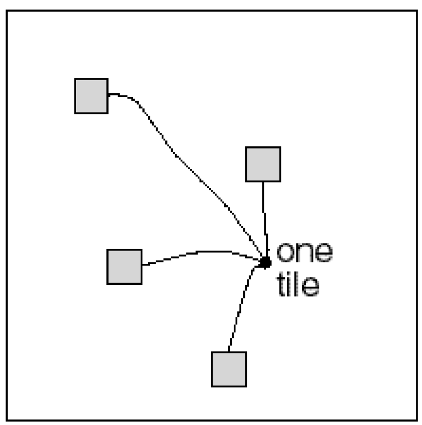
#Tilings ⇒ Resolution of final approximation

Tile Coding Cont.

Irregular tilings



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) and 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

Off-Policy Learning

- Learning about a way of behaving while behaving in some other way

Importance Sampling

- Behave according to policy μ
- Evaluate policy π
- Episode (e): $s_0\ a_0\ r_1\ s_1\ r_2\ \ldots\ s_{T-1}\ a_{T-1}\ r_T\ s_T$
- $Pr(e|\pi) = \Pi^{T-1}_{k=0}\ \pi(a_k \mid s_k)\ Pr(s_{k+1}|s_k,a_k)$
- Importance Sampling Ratio:

$$\frac{Pr(e|\pi)}{Pr(e|\mu)} = \Pi_{k=0}^{T-1} \frac{\pi(a_k|s_k)}{\mu(a_k|s_k)}$$

High variance

Off-Policy with Linear Function Approximation

$$\Delta\theta_t = \alpha[r_{t+1} + \gamma\theta^T\phi_{s_{t+1},a_{t+1}} - \theta^T\phi_{s_t,a_t}]\phi_{s_t,a_t}$$

$$\Delta\theta_t = \alpha[r_{t+1} + \frac{\pi(a_{t+1}|s_{t+1})}{\mu(a_{t+1}|s_{t+1})}\gamma\theta^T\phi_{s_{t+1},a_{t+1}} - \theta^T\phi_{s_t,a_t}]\phi_{s_t,a_t}\left(\Pi_{k=0}^{t}\frac{\pi(a_k|s_k)}{\mu(a_k|s_k)}\right)$$

Precup, Sutton & Dasgupta

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

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 action

An option is a triple $o = \langle \mathrm{I}, \pi, \beta \rangle$

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

Option execution is assumed to be call-and-return

Example: `docking`

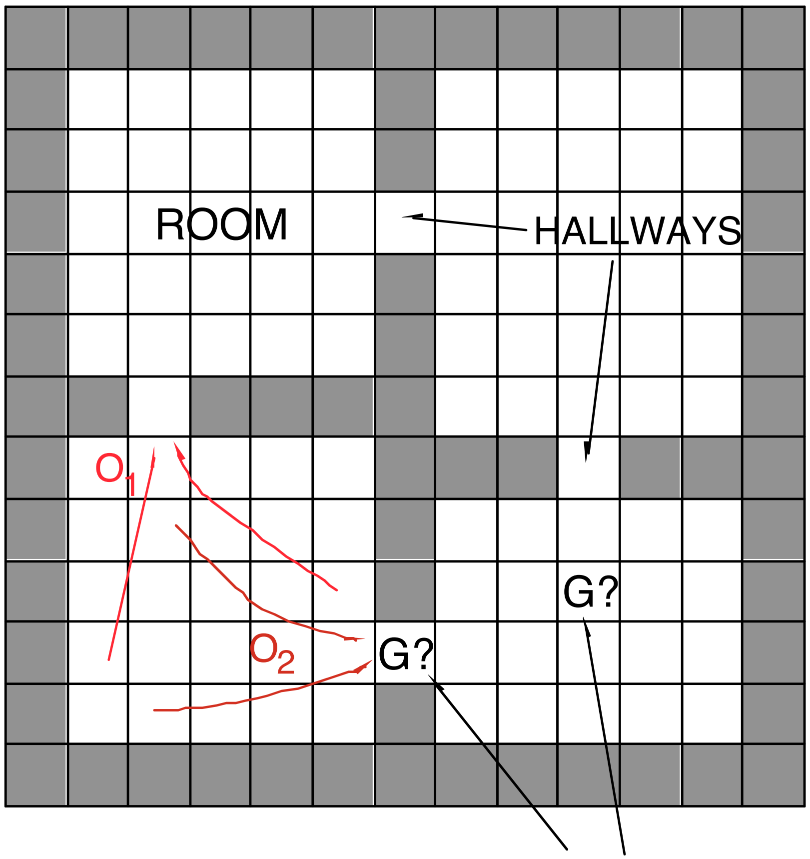
I : all states in which charger is in sight

π : hand-crafted controller

β : terminate when docked or charger not visible

Options can take variable number of steps

Rooms Example

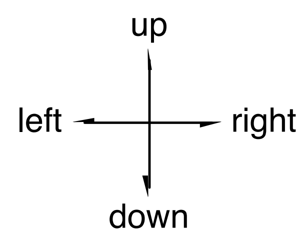


Goal states are given
a terminal value of 1

4 rooms

4 hallways

4 unreliable
primitive actions



Fail 33%
of the time

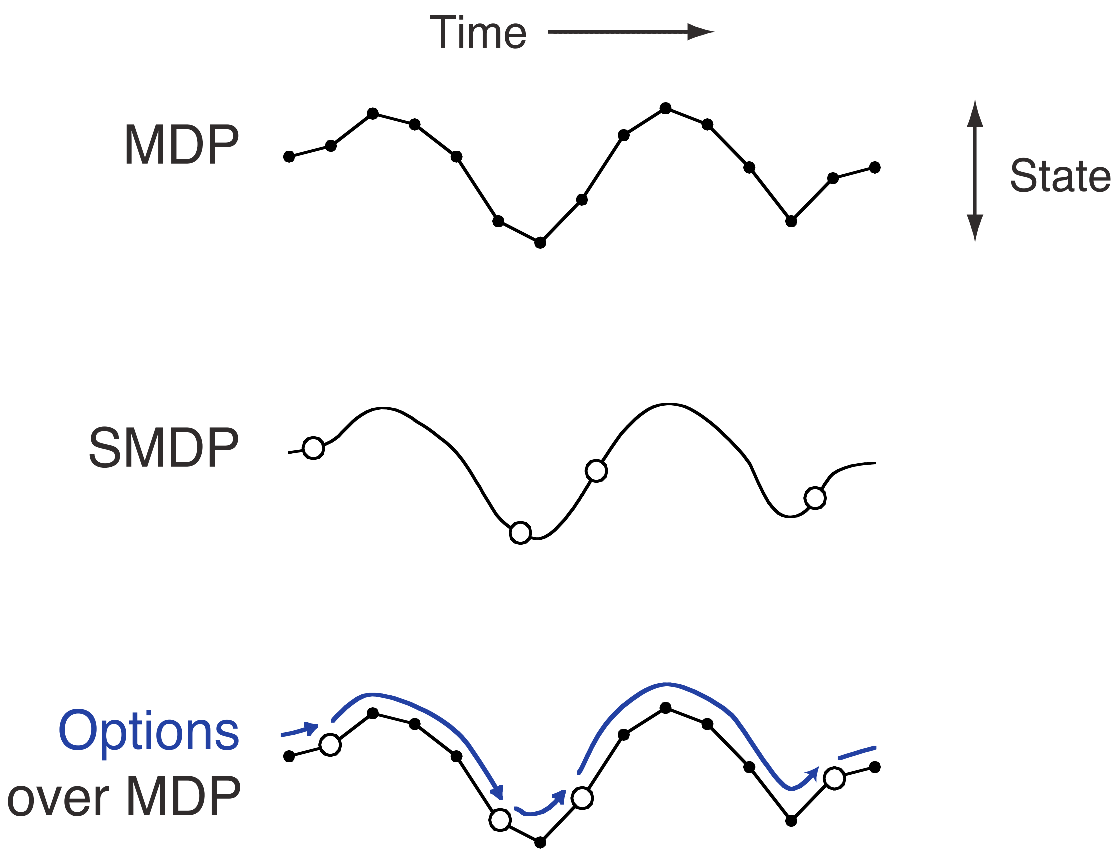
8 multi-step options

(to each room's 2 hallways)

Given goal location,
quickly plan shortest route

All rewards zero
$\gamma = .9$

Options define a Semi-Markov Decison Process (SMDP)



Discrete time
Homogeneous discount

Continuous time
Discrete events
Interval-dependent discount

Discrete time
Overlaid discrete events
Interval-dependent discount

A discrete-time SMDP overlaid 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

A theoretical fondation for what we really need!

But the most interesting issues are beyond SMDPs...

Value Functions for Options

Define value functions for options, similar to the MDP case

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

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 about 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

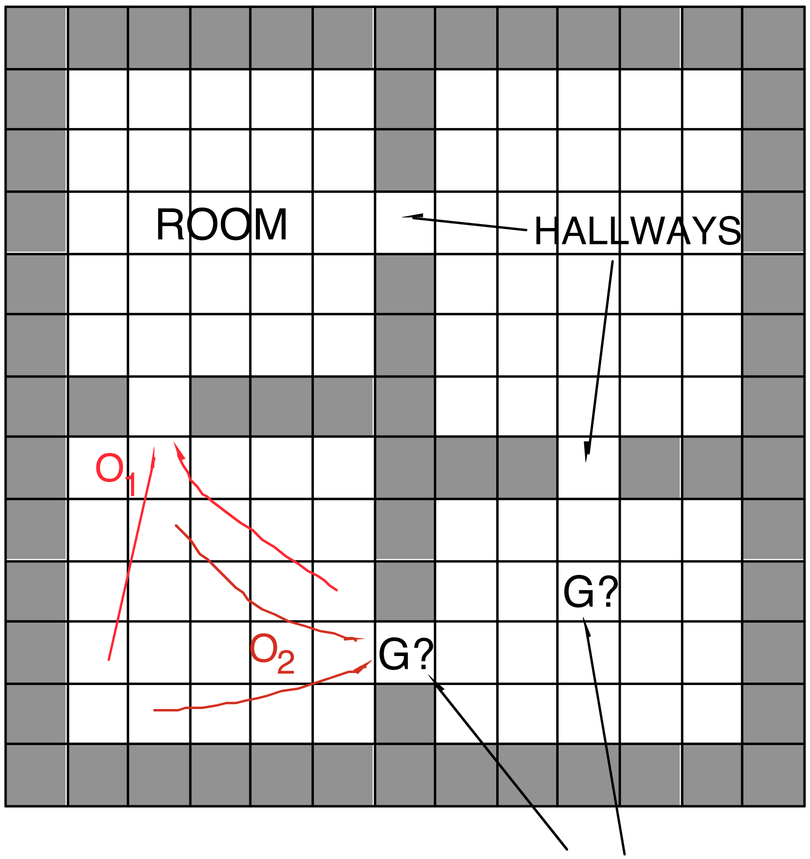
  $$r_s^o = E\{r_1 + \gamma r_2 + \ldots + \gamma^{k-1} r_k \mid s_0 = s,\ o \text{ taken in } s_0, \text{ lasts } k \text{ steps}\}$$

- a next-state part

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

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

Room Example

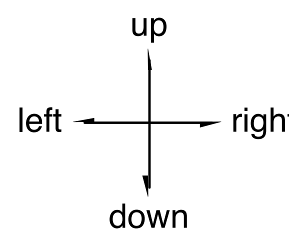


Goal states are given a terminal value of 1

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 Iteration Generalized to Options

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

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

The algorithm converges to the optimal value function, given the options:

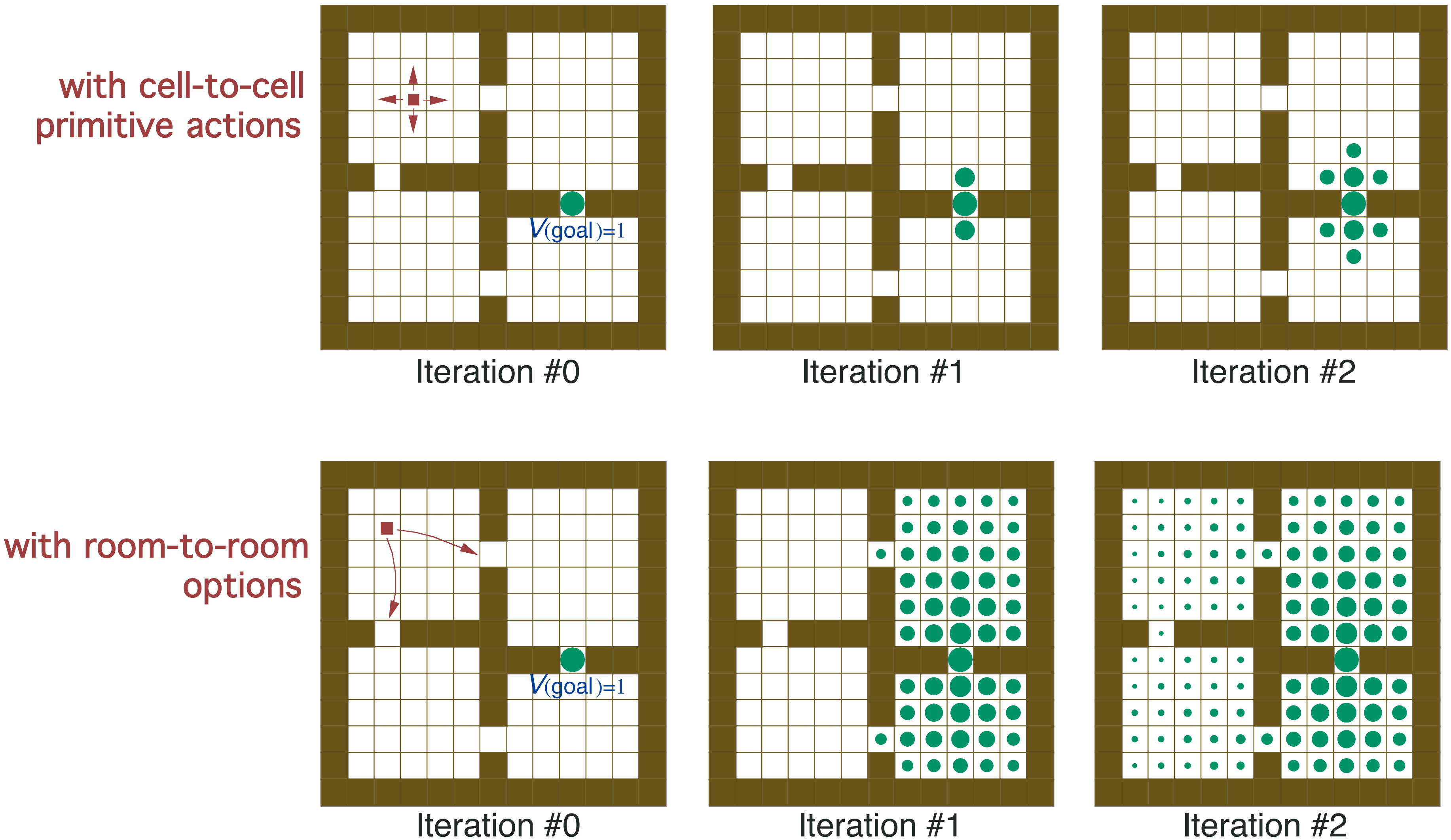
$$\lim_{k \to \infty} V_k = V_{\mathcal{O}}^*$$

Once $V_{\mathcal{O}}^*$ is computed, $\mu_{\mathcal{O}}^*$ is readily determined.

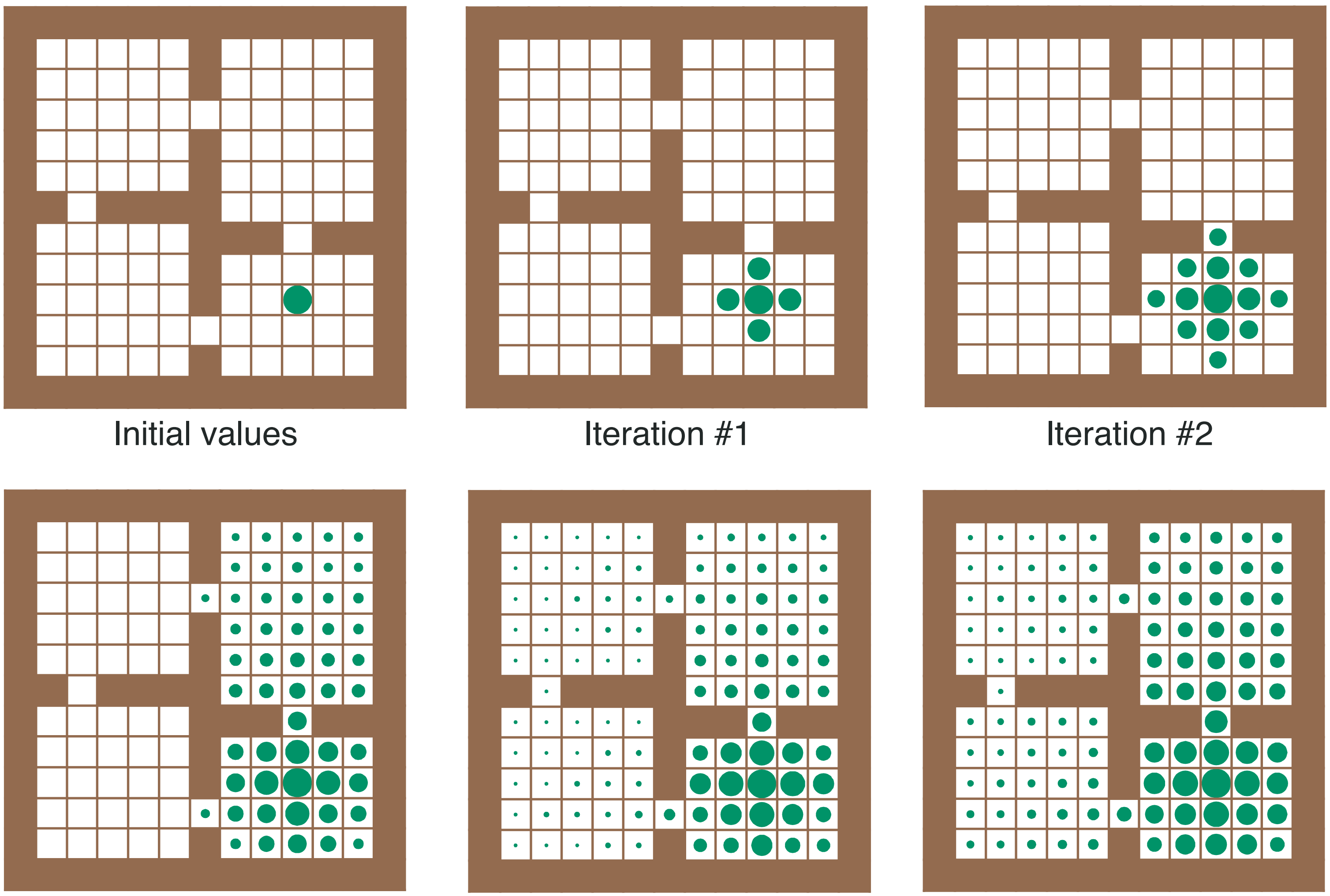
If $\mathcal{O} = \mathcal{A}$, the algorithm reduces to conventional value iteration

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

Rooms Example



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: $\mu : \mathcal{S} \times \mathcal{O} \mapsto [0,1]$
- 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

A theoretical foundation for what we really need!

But the most interesting issues are beyond SMDPs...

Advantages of Dual MDP/SMDP View

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 early)

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 them 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 options
- forcing them to terminate before β says to

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

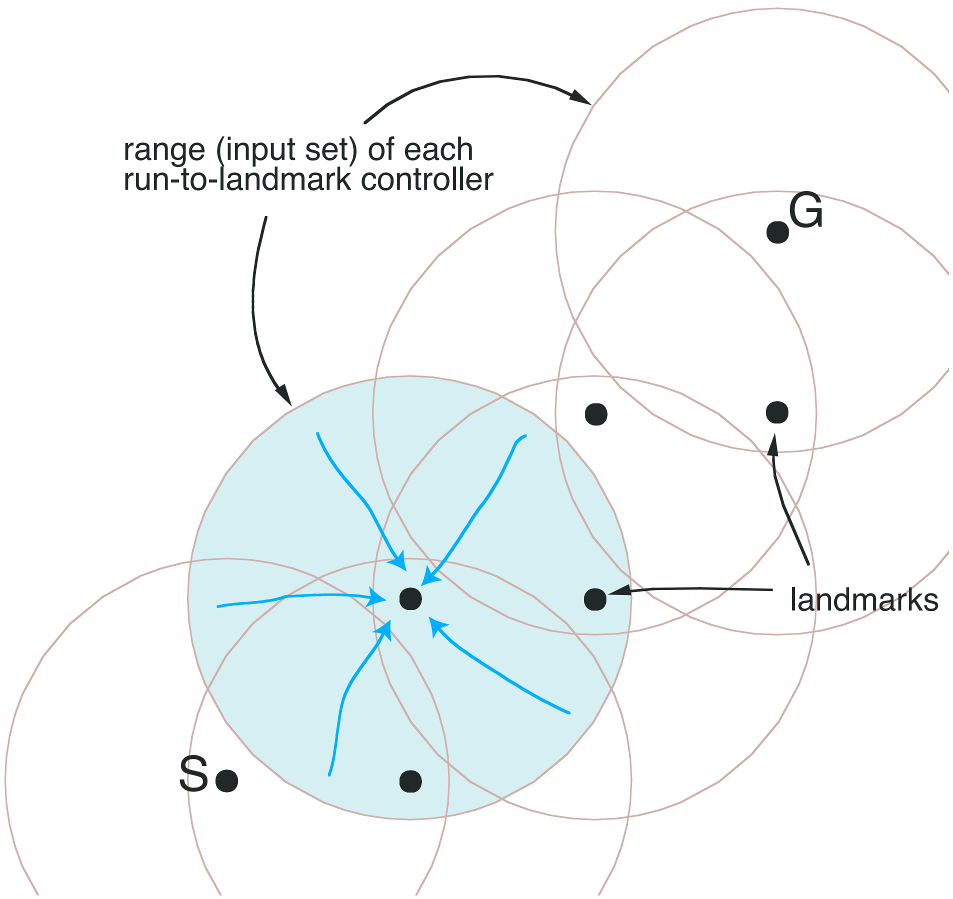
$$Q^{\mu}(s,o) < Q^{\mu}(s,\mu(s)), \quad \text{where } s \text{ is the state at that time, } o \text{ is the ongoing option}$$

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

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

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

Landmarks Task



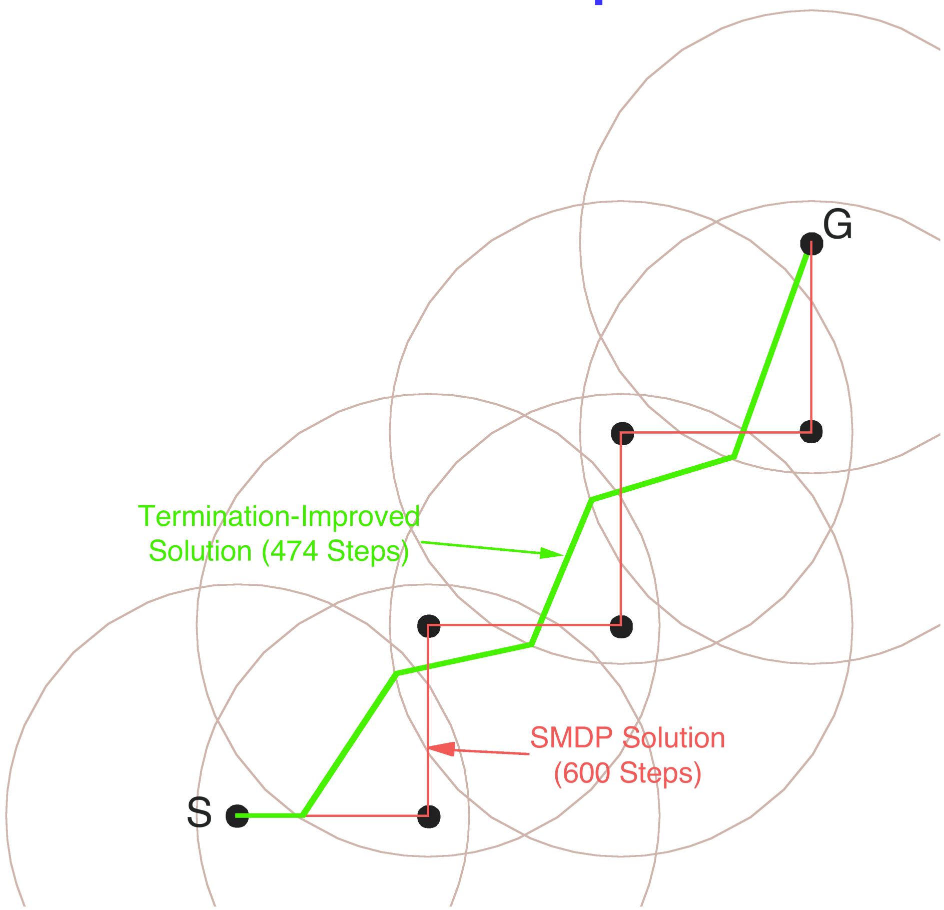
Task: navigate from S to G as fast as possible

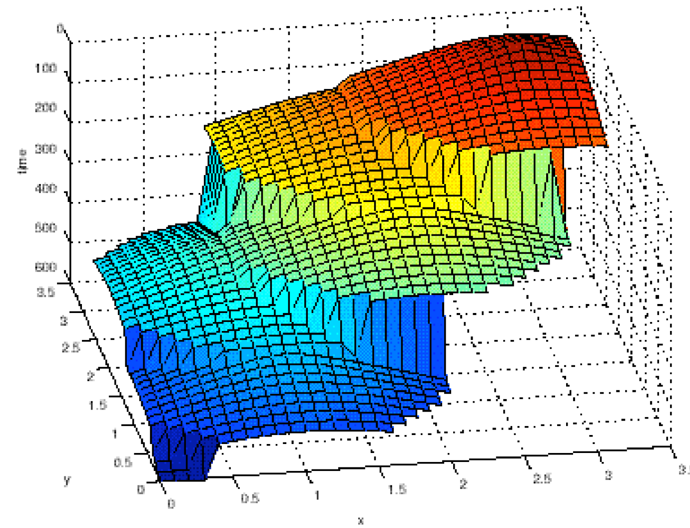
4 primitive actions, for taking tiny steps up, down, left, right

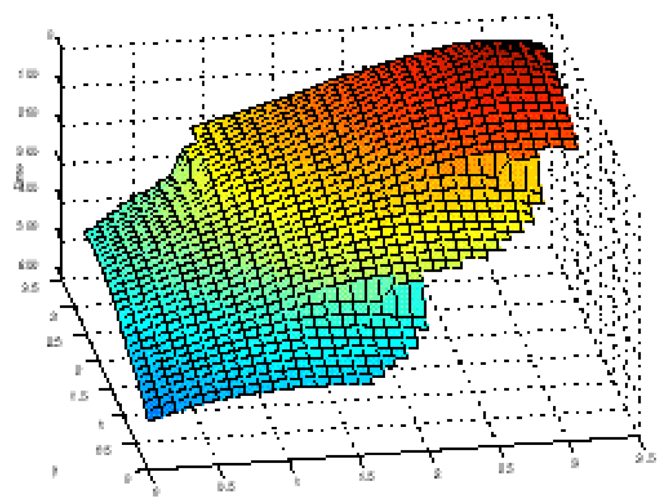
7 controllers for going straight to each one of the landmarks, from within a circular region where the landmark is visible

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

Termination Improvement for Landmarks Task

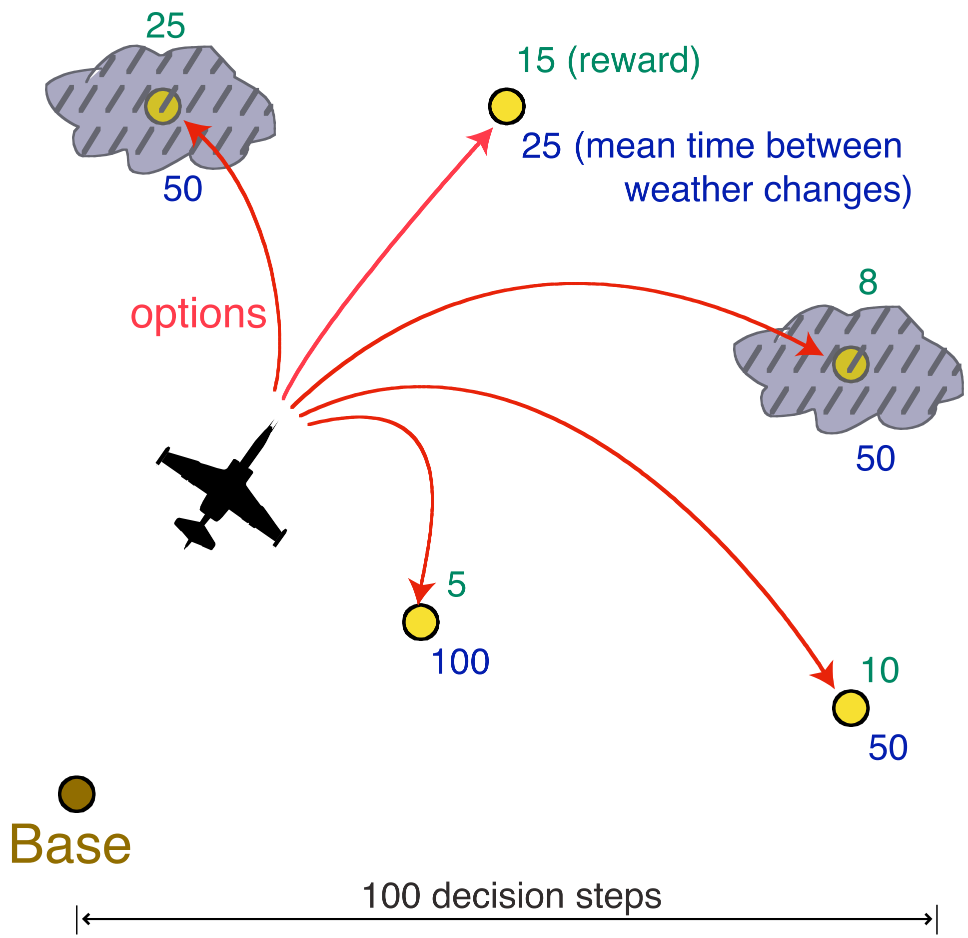






Allowing early termination based on models improves the value function at no additional cost!

Illustration: Reconnaissance Mission Planning (Problem)



- Mission: Fly over (observe) most valuable sites and return to base
- Stochastic weather affects observability (cloudy or clear) of sites
- Limited fuel
- Intractable with classical optimal control methods
- Temporal scales:
  - Actions: which direction to fly now
  - Options: which site to head for
- Options compress space and time
  - Reduce steps from ~600 to ~6
  - Reduce states from ~10^{11} to ~10^6

$$Q_O^*(s, o) = r_s^o + \sum_{s'} p_{ss'}^o V_O^*(s')$$

any state (10^6)

sites only (6)

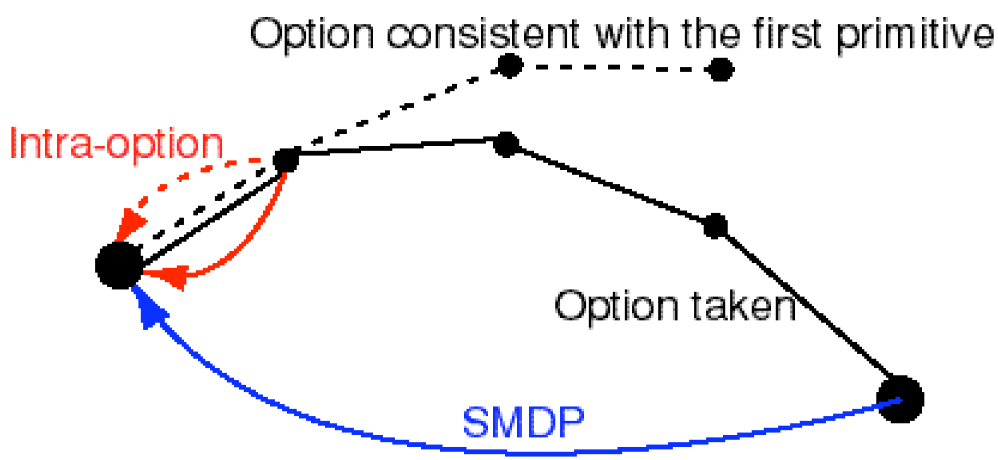
Illustration: Reconnaissance Mission Planning (Results)

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-evaluation
  - Plans as if options must be followed to completion
  - But actually takes them for only one step
  - Re-picks a new option on every step
- Static planner:
  - Assumes weather will not change
  - Plans optimal tour among clear sites
  - Re-plans whenever weather changes

Intra-Option Learning Methods for Markov Options

Idea: take advantage of each fragment of experience

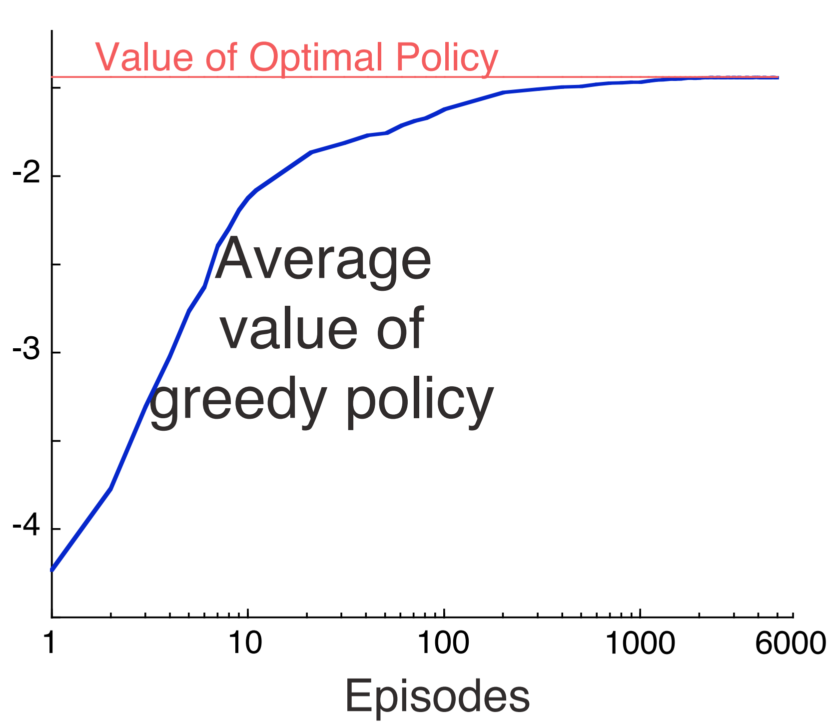


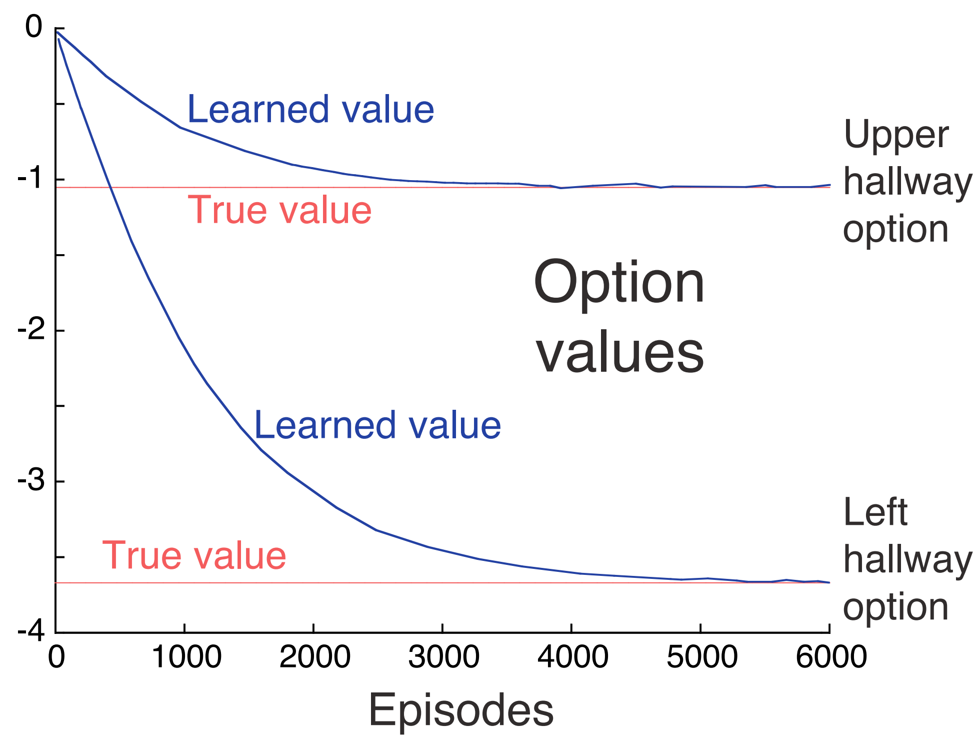
SMDP Learning: execute option to termination,then update only the option taken

Intra-Option Learning: after each primitive action, update all the options 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 Learning

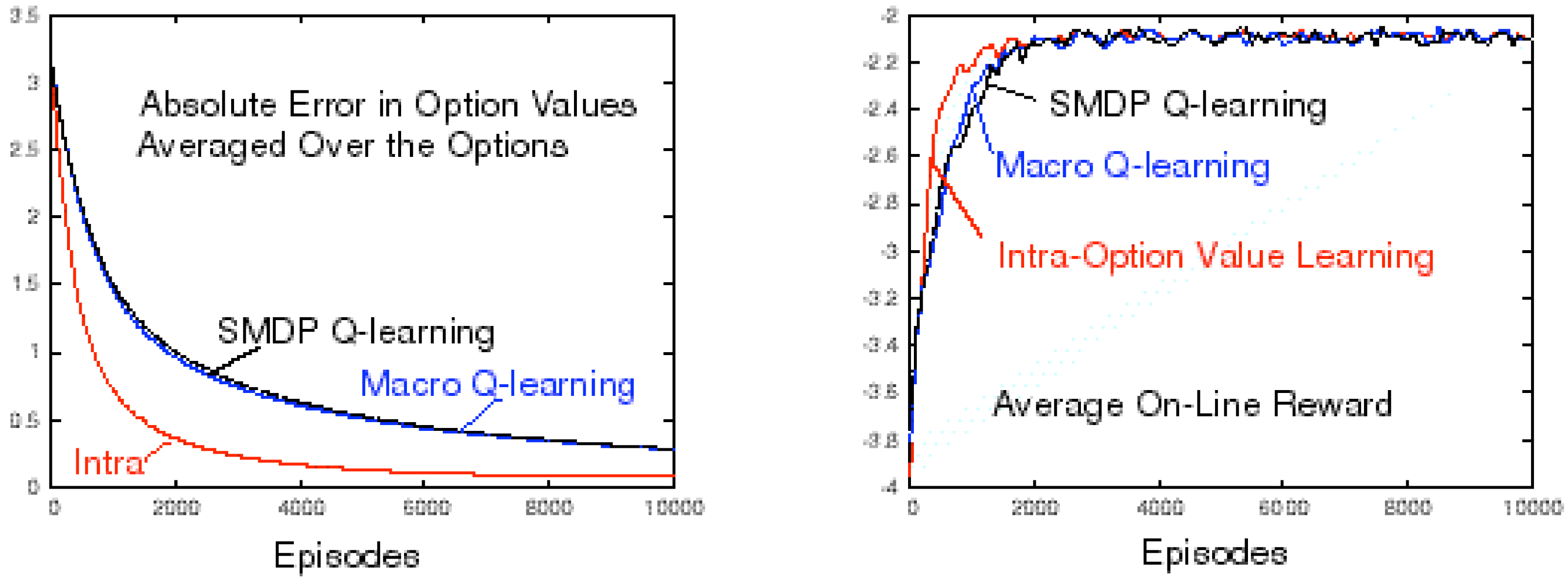




Random start, goal in right hallway, random actions

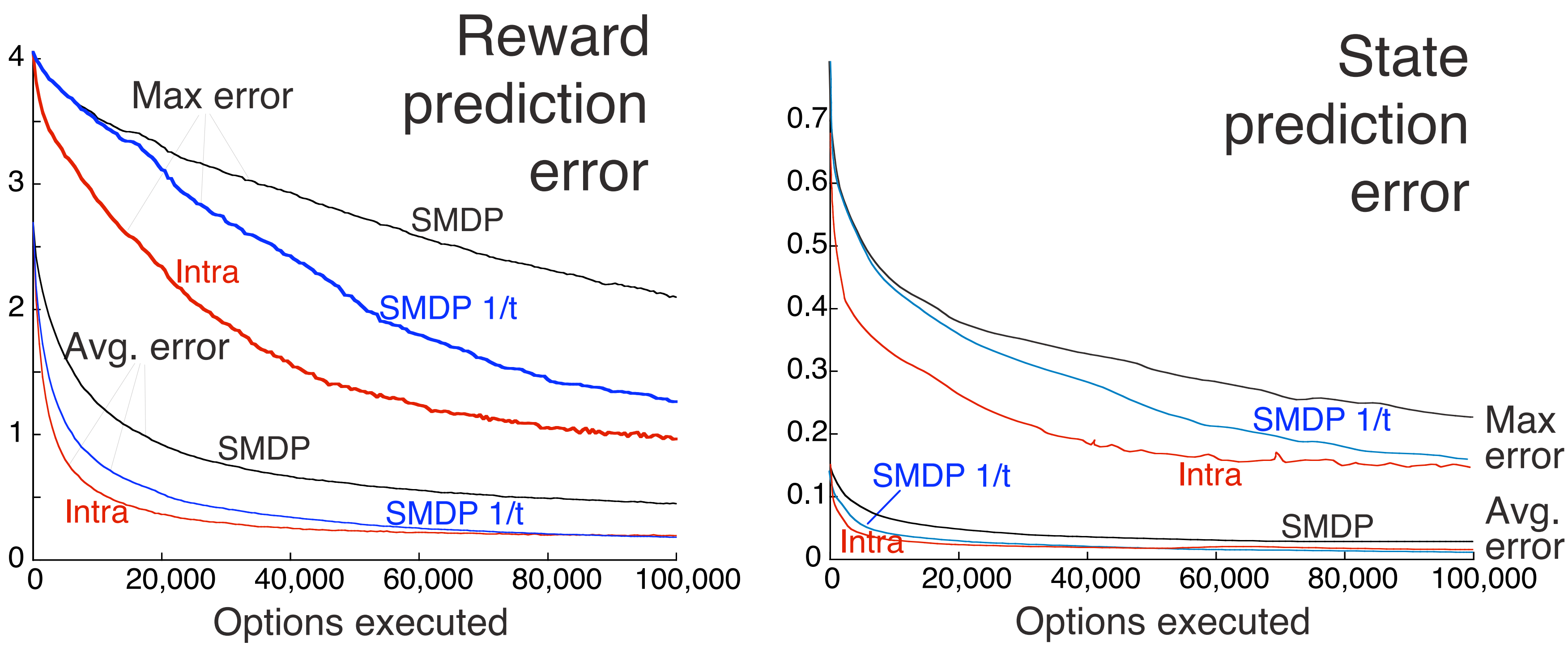
Intra-option methods learn correct values without ever taking the options! SMDP methods are not applicable here

Intra-Option Value Learning Is Faster Than SMDP Value Learning



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

Intra-Option Model Learning



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

Intra-option methods work much faster than SMDP methods

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: $<\mathsf{G}, g>$

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

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

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

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

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

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 them 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 representation
  - Expressive
  - Clear
  - Suitable for learning and planning
- Much more to learn than just one policy, one set of values
  - A framework for "constructivism" – for finding models of the world that are useful for rapid planning and learning