RL 15: Model-based RL
Complexity of RL

Michael Herrmann

University of Edinburgh, School of Informatics

11/03/2014

Overview

- Complexity of RL
- Model-based
- The Dyna architecture
- Efficient sampling
- Theory of RL

Complexity of RL

- "The (PO)MDP frameworks are fundamentally broken, not because they are insufficiently powerful representations, but because they are too powerful. We submit that, rather than generalising these models, we should be specialising them if we want to make progress on solving real problems in the real world."

T. Lane, W.D. Smart, Why (PO)MDPs Lose for Spatial Tasks and What to Do About It, ICML Workshop on Rich Representations for RL, 2005.

What is the Issue? (Lane et al.)

- In our efforts to formalise the notion of "learning control", we have striven to construct ever more general and, putatively, powerful models. By the mid-1990s we had (with a little bit of blatant "borrowing" from the Operations Research community) arrived at the (PO)MDP formalism (Puterman, 1994) and grounded our RL methods in it (Sutton & Barto, 1998; Kaelbling et al., 1996; Kaelbling et al., 1998).
- These models are mathematically elegant, have enabled precise descriptions and analysis of a wide array of RL algorithms, and are incredibly general. We argue, however, that their very generality is a hindrance in many practical cases.
- In their generality, these models have discarded the very qualities – *metric, topology, scale*, etc. – that have proved to be so valuable for many, many science and engineering disciplines.

What is Missing in POMDPs?

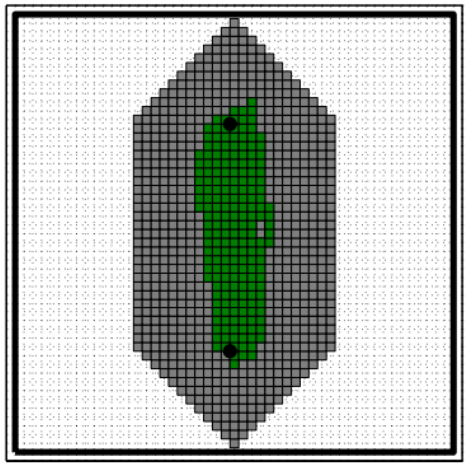
- POMDPs do not describe natural metrics in environment
  - When driving, we know both global and local distances
- POMDPs do not natively recognise differences between scales
  - Uncertainty in control is entirely different from uncertainty in routing
- POMDPs conflate properties of the environment with properties of the agent
  - Roads and buildings behave differently from cars and pedestrians: we need to generalise over them differently
- POMDPs are defined in a global coordinate frame, often discrete
  - We may need many different representations in real problems

Use separable (or hierarchical) representations

- For the natural AC we found that for a special representation of the value function we can improve the policy independently of the quality of the approximation of the value function
- We had to assume that the distribution of states and the rewards ($\mathcal{Q}$) are independent on the (parameters of the) policy
- Experience: How to make optimal use of sample? Represent samples by a model.
- Models in RL are used in order to predict state transitions. Can models be used also in the learning process?

Use metrics of state spaces

Imposes a "speed limit" on the agent – the agent cannot transition to arbitrary points in the environment in a single step.



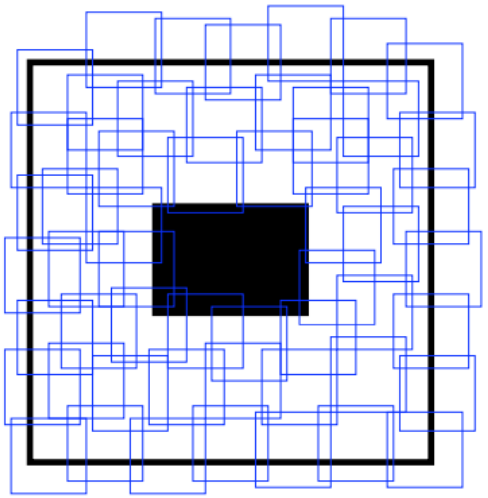
Metric envelope bound for point-to-point navigation in an open-space gridworld environment. The outer region is the elliptical envelope that contains 90% of the trajectory probability mass. The inner, darker region is the set of states occupied by an agent in a total of 10,000 steps of experience (319 trajectories from bottom to top).

Consequences:

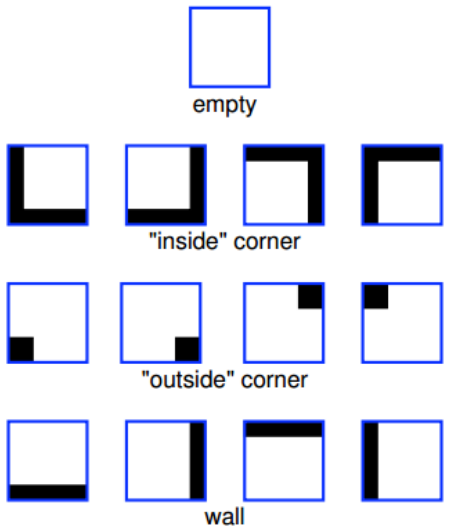
- Agent can neglect large parts of state space when planning.
- More importantly, however, this result implies that control experience can be generalised across regions of state space.
- If the agent learns a good policy for one bounded region of the state space, and it can find a second region that is homeomorphic to the first.

Use local information: Manifold representations

- A manifold representation models the domain of the value function using a set of overlapping local regions, called charts.
- Each chart has a local co-ordinate frame and a (local) Euclidean distance metric. Collection of charts and their overlap regions forms the manifold.
- Embed partial value functions on these charts, and combine them, using the theory of manifolds, to provide a global value function.



A simple navigation domain, covered with charts.



Recall Hierarchical RL

Use point-based algorithms

- Point-based algorithms have been surprisingly successful in computing approximately optimal solutions for POMDPs.
- What are the belief-space properties that allow some POMDP problems to be approximated efficiently, explaining the point-based algorithms' success?

Covering Number

- Covering number of a space is the minimum number of given size balls that needed to cover the space fully
- Hsu et al. show that an approximately optimal POMDP solution can be computed in time polynomial in the covering number of Θ
- Covering number also reveals that the belief space for some interactive problems behaves more like the union of some lower-dimensional spaces (for each agent) rather than an high-dimensional space for all agents
- Softens the difference between partially and fully observable problems

Three sources of error in RL

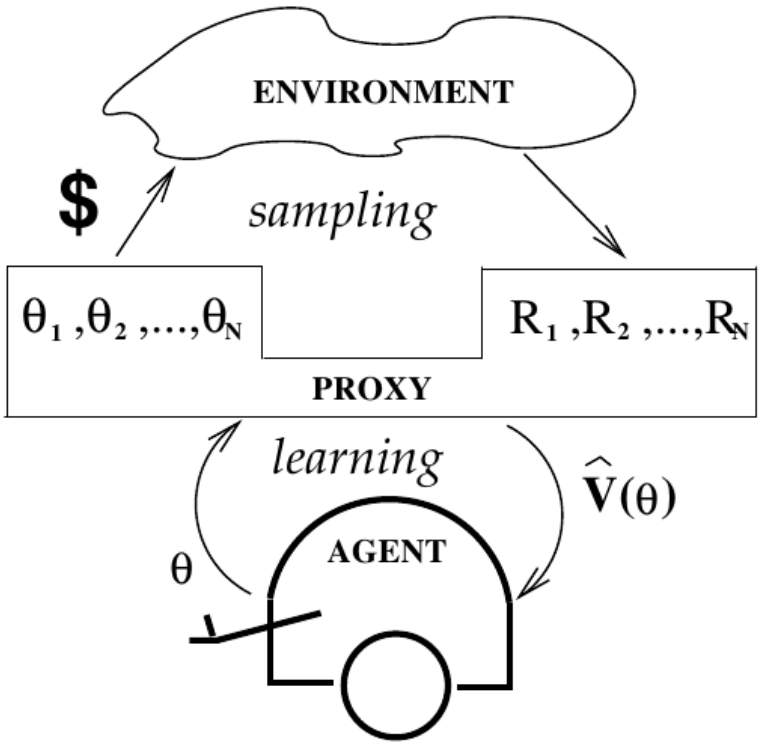
- Misallocation of approximation resources to state space: without knowing the optimal policy one cannot sample from the distribution that it induces on the stochastic system's state space
- Coupling of optimal decisions at each stage: finding the optimal decision rule at a certain stage hinges on knowing the optimal decision rule for future stages
- Inadequate control of generalisation errors: without a model ensemble averages must be approximated from training trajectories

D. Blatt and A. Hero ICAPS Workshop 2006

Types of RL approaches

- Policy search: $\pi : s \rightarrow a$
- Value function based: $(s, a) \rightarrow V$
  implies policy-based methods by search for the action that maximised value
- Model based $(s, a) \rightarrow (s', r)$
  implies value-based methods by solving Bellman equations

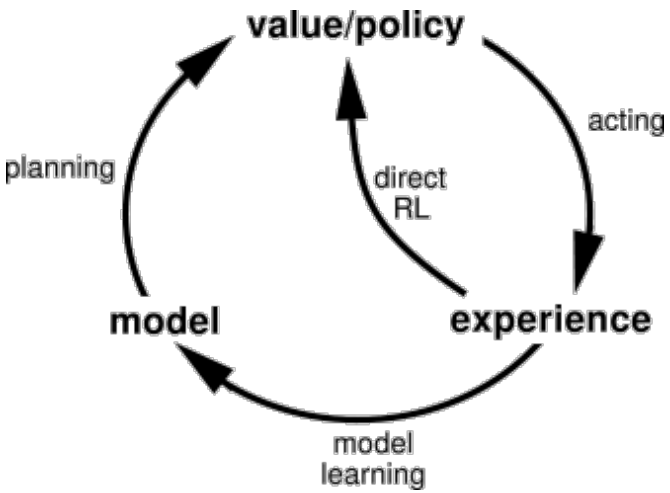
Policy evaluation process: Using a model



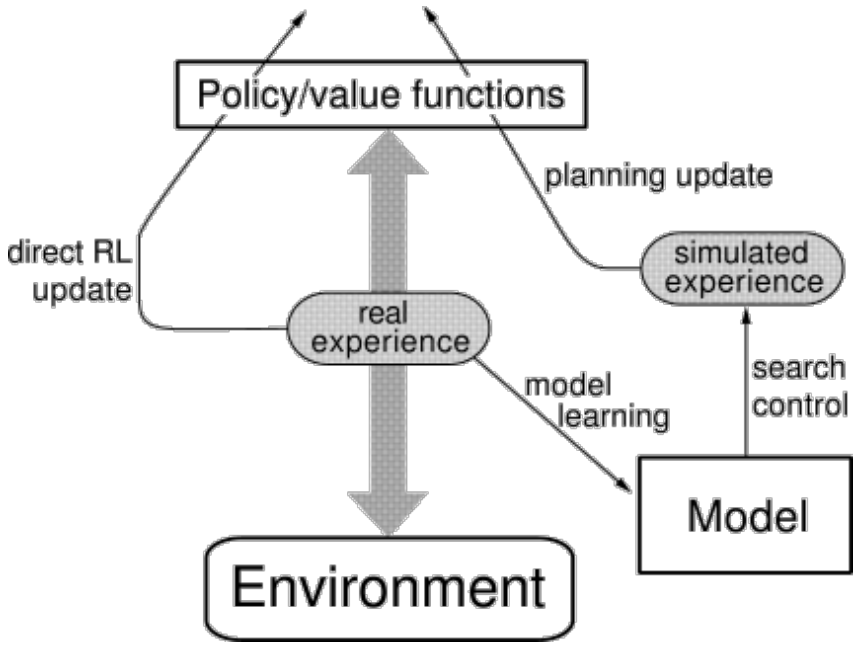
- Sampling process is costly
- Proxy collects the samples from environment and constructs an agent-centric model that predicts the effects of hypothetical agent policies.
- Agent learns by interacting with the proxy.

from Peshkin & Shelton 2001

Model based RL: Dyna



Relationships among learning, planning, and acting.



The general Dyna architecture

Dyna-$\mathcal{Q}$

1. Record state s and select action a
2. Execute action a and record next state s' and reward r,
3. Improve state-action value function using the sample $\langle s, a, r, s' \rangle$
4. Improve world model $M(s, a) \rightarrow (s', r)$
5. Enter planning cycle
   repeat:
   1. Select a random state $\tilde{s}$ and a random action $\tilde{a}$ and
   2. Apply the world model in order to obtain $\tilde{s}'$ and $\tilde{r}$
   3. Improve state-action value function using the sample $\langle \tilde{s}, \tilde{a}, \tilde{r}, \tilde{s}' \rangle$
6. Go to 1

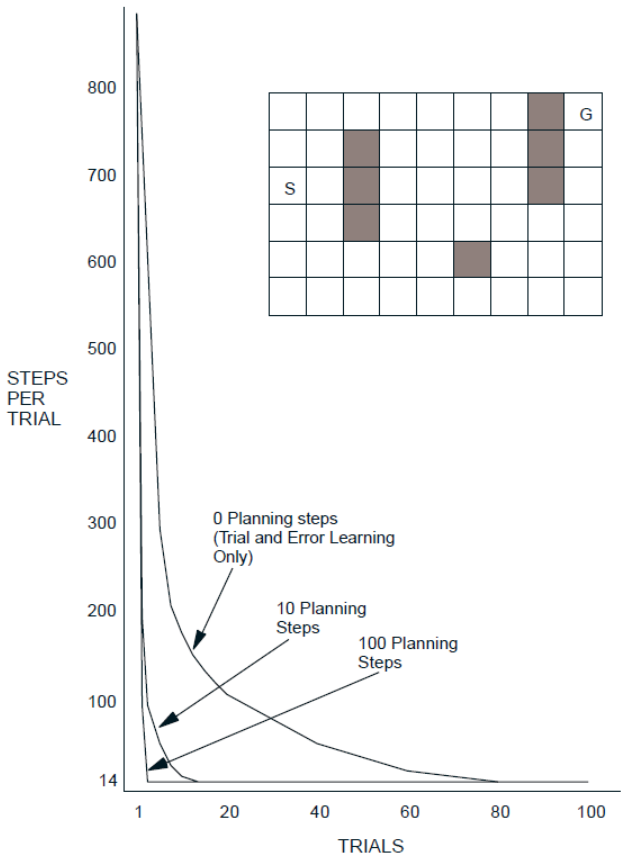
Remarks on Dyna-$\mathcal{Q}$

- Dyna-$\mathcal{Q}$ uses $\mathcal{Q}$ learning as a subroutine
- Dyna-$\mathcal{Q}$ uses "dreaming" to obtain a *consistent* value function
- Dyna-$\mathcal{Q}+$ includes an exploration bonus, e.g. $\kappa\sqrt{t\left(s,a\right)}$, where $t\left(s,a\right)$ is the number of time steps since action a was last executed in state s (in the real world), κ is the exploration strength

$$\mathcal{Q}\left(s,a\right)=\mathcal{Q}\left(s,a\right)+\eta\left(r+\gamma\max_b\mathcal{Q}\left(s',b\right)-\mathcal{Q}\left(s,a\right)+\kappa\sqrt{s\left(s,a\right)}\right)$$

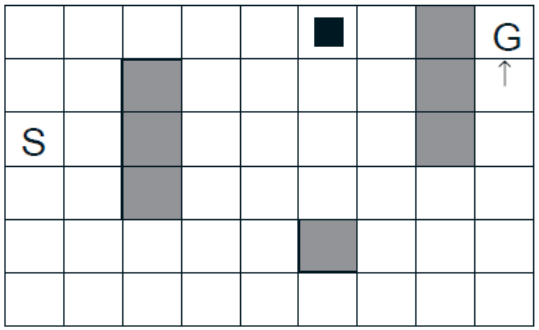
- Dyna can be used with other algorithms e.g. Dyna-AHC (*adaptive heuristic critic* including a prediction of return, i.e. long-term cumulative reward)

Dyna-AHC: Experiment

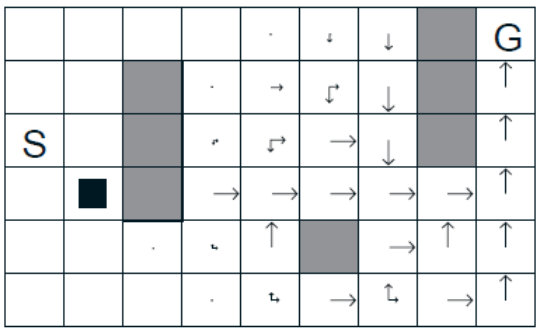


Trial is one trip from start state S to goal state G. Shown are averages over 100 runs.

WITHOUT PLANNING ($k = 0$)



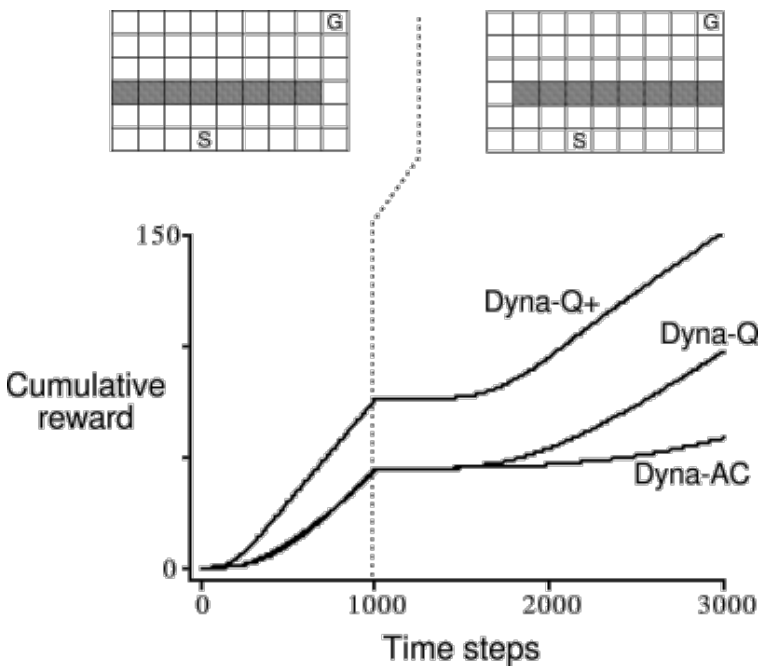
WITH PLANNING ($k = 100$)



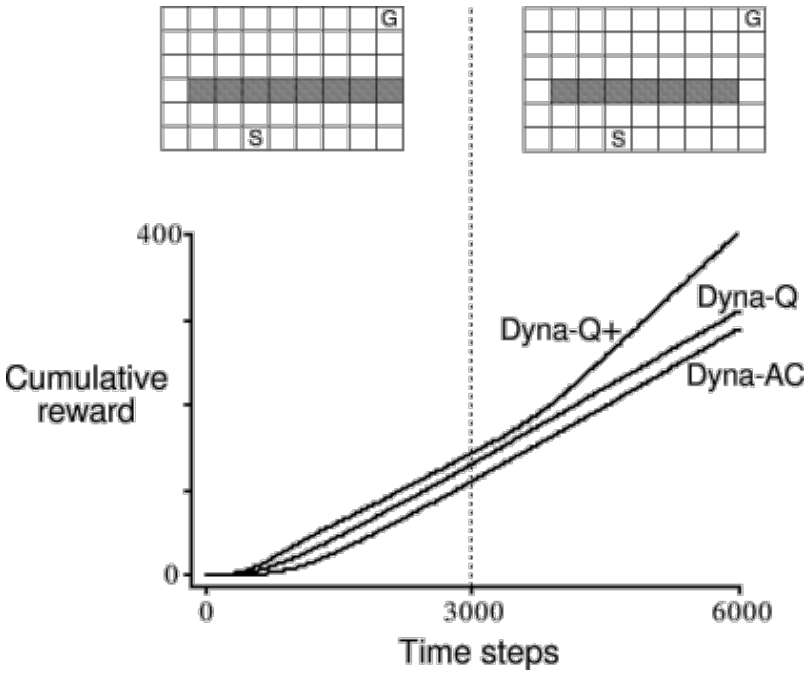
Policies found by the middle of the second trial. Black square is the current location.

R. S. Sutton: Reinforcement Learning Architectures.

When the Model Is Wrong



Blocking task:
Left environment for the first 1000 steps, then right one for the rest.



Shortcut task:
Left environment for the first 3000 steps, then the right one for the rest.

from the Sutton and Barto book

Using a model

- Large state spaces
  - factorisable transition probabilities
- POMDP with a restricted class of strategies Π
  - chose $\pi \in \Pi$ with maximal return
- what is *sample complexity*? From supervised learning
  - How many samples are needed to learn a function $f \in \mathcal{F}$ of a certain complexity?
  - e.g. neural network realises $h(x)$ with $h \in \mathcal{H}$ in order to approximate $f(x)$. Assume $|\mathcal{H}| = n$ then typically only $O(\log(n))$ samples are needed to find a good $h(n)$.
  - Since we are choosing from $\mathcal{H}$ the complexity of f does not play a role (if $|\mathcal{H}|$ is small and $|\mathcal{F}|$ is large)
- Assume a simulator (a generative model) of the POMDP
- Find bounds on the required amount of simulated experience

Sample complexity in a POMDP

- Using the policy $\pi \in \Pi$ and starting state s_0, generate many trials (MC-style) and find $V^{\pi}(s_0)$
- Now for a different $\pi' \in \Pi$ what use can we make of these trials?
- If we cannot re-use these trials we are left with a complexity $O(n)$ if $|\Pi| = n$ (instead of e.g. $O(\log(n))$)
  [Π does not have to be finite here]
- Several methods for generating reusable trajectories:
  - trajectory trees (easier, but specific generative model)
  - random trajectories (harder, but simple generative model)
  - likelihood ratios
- Number of required trajectories indep. of state space size
- Linear in complexity of policy space

Generative model

Given a POMDP M, then a model of M is

- a randomised algorithm that for a given state-action pair (s, a) outputs
  - a state s_0 that is distributed according to the next-state distribution $P(\cdot|s, a)$,
  - an observation o that is distributed according to the distribution $\mathcal{Q}(\cdot|s)$, and
  - the reward $R(s, a)$.

Task: Let M be a POMDP with start state s_0, and let Π be a class of strategies. Find

$$opt(M, \Pi) = \sup_{\pi \in \Pi} V^{\pi}(s_0)$$

where $V^{\pi}(s_0)$ is the expected return of π from s_0.

Summary

- Algorithms should work on the effective complexity as provided by optimal representations
- Models can help reducing the effects of sub-optimal representations
- Models in biology: The ubiquity of model-based reinforcement learning (Doll, Simon and Daw, COiN 2012)
- Models need to be learned and updated (exploration becomes more important)

Next lectures

- 14/3 Complexity of RL
- 18/3 apprenticeship learning, inverse RL
- 21/3 no lecture
- 25/3 revision, unified view, recent trends

Acknowledgements & References

Some material was adapted from web resources associated with Sutton and Barto's Reinforcement Learning book. Today mainly based on

- on slides by Dr. Subramanian Ramamoorthy from last year
- Kearns, M., Mansour, Y., & Ng, A. Y. (1999). Approximate planning in large POMDPs via reusable trajectories. Advances in Neural Information Processing Systems, 12, 1001-1007.
- Peshkin, L., & Shelton, C. R. (2002). Learning from scarce experience. arXiv preprint cs/0204043.

See also:

http://umichrl.pbworks.com/w/page/7597585/Myths of Reinforcement Learning