RL 16: Model-based RL and Multi-Objective Reinforcement Learning

Michael Herrmann

University of Edinburgh, School of Informatics

13/03/2015
Overview

- Complexity of RL
- Model-based
- The Dyna architecture
- MORL
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.
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
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-Q

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

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

- 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.

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 $\Pi$
  - 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
Multiobjective Reinforcement Learning

- RL is sequential decision making under uncertainties based on a scalar evaluation signal
- Defining a single reward signal is often the result of a complex design process. Typically several reward signals are available to the agent.
- How can an agent solve several tasks with different rewards simultaneously?
- Does not annihilate information by summing the rewards (which may not be comparable)
- Does the problem become easier or harder for multiple values?
- Robot example: Reach goal(s), avoid wear, keep track of position, avoid getting to close to a human, avoid running out of energy, help other agent that are met on the way ... 
- Two main strategies:
  - Scalar combination of the reward signals
  - Pareto optimisation
Multi-objective Optimisation

**Example:** A machine is characterised by power and torque. A machine is better if – at equal torque – its power is higher.

Combination of utility functions, e.g.

\[ f(x) = |f_1(x)|^\alpha + |f_2(x)|^\alpha \]

\[ f(x) = \alpha f_1(x) + (1 - \alpha) f_2(x) \]

How to set \( \alpha \)?

If \( \alpha \) is not implied by the problem, any value in between the two maxima is equally good.

If a comparison between the two quantities is not possible, a set of solutions should be considered as optimal (Pareto-optimal).

How to optimise one criterion without loosing on other criteria?
Multi-objective Optimisation

$x^*$ is Pareto optimal for a class of fitness functions \( \{f_i\} \) if there exists no \( x \neq x^* \) with \( f_i(x) \geq f_i(x^*) \) for all \( i \) or, equivalently, \( x^* \) is not dominated by any other \( x : \exists x \succ x^* \) (more specifically \( \exists x \succ \{f_i\} x^* \))

Example with three fitness functions

Same example: Pareto area spanned by maxima in a shape-dependent way
Two strategies: ... and questions

- Scalarised approach: Find a single policy that optimises a combinations of the rewards
  - Which reward combination is preferable at which state?
  - Although a weighted sum of rewards might be an option, usually a weighted sum of values is considered to more relevant of the actions choice

- Pareto approach
  - Find multiple policies that cover the Pareto front: Sampling in a high-dimensional case
  - In principle, collective search required for sampling the Pareto set
  - What is a good approximation/representation of the Pareto front?
A parametrised combination of multiple reward signals is used with different parameters in different runs to address different points along the Pareto front. The set of all solutions obtained in this way contains the Pareto front (e.g. in case of a non-connected Pareto front also non-Pareto optimal solutions may be found).

The agent may change the parametrisation according to progress on each of the goals.
<table>
<thead>
<tr>
<th>MORL Approaches</th>
<th>Basic Principle</th>
</tr>
</thead>
<tbody>
<tr>
<td>The weighted sum approach</td>
<td>A linear weighted sum of Q-values is computed as the synthetic objective function.</td>
</tr>
<tr>
<td>The W-learning approach</td>
<td>Each objective has its own recommendation for action selection and the final decision is based on the objective with the largest value.</td>
</tr>
<tr>
<td>The AHP approach</td>
<td>The analytic hierarchy process (AHP) is employed to derive a synthetic objective function.</td>
</tr>
<tr>
<td>The ranking approach</td>
<td>“Partial policies” are used as the synthetic objective function.</td>
</tr>
<tr>
<td>The geometric approach</td>
<td>A target set satisfying certain geometric conditions in multi-dimensional objective space is used as the synthetic objective function.</td>
</tr>
<tr>
<td>The convex hull approach</td>
<td>Learn optimal value functions or policies for all linear preference settings in the objective space.</td>
</tr>
<tr>
<td>The varying parameter approach</td>
<td>Performing any single-policy algorithm for multiple runs with different parameters, objective threshold values and orderings.</td>
</tr>
</tbody>
</table>

(C. Liu et al., 2013)
The Top-Q algorithm chooses simply

\[ a_t = \max_i W_i = \max_i \max_a Q_i(s_t, a) \]

The result depends usually on the scales of the reward signals.

\( W \)-learning: Define a principal value function \( \ell \) and choose

\[ a_\ell(t) = \max_a Q_\ell(s(t), a) \]

Calculate \( W \)-values by

\[ W_i = \max_a Q_i(s(t), a) - Q_i(s(t), a_\ell) \]

or (to avoid oscillations)

\[ W_i(s) = (1 - \alpha) W_i(s) + \alpha P_i(s) \]

\[ P_i(s) = \max_a Q_i(s, a) - \left( r_i + \gamma \max_b Q_i(s', b) \right) \]

set new \( \ell = \arg \max_i W_i \)
- **AHP**: Choose action $a$ if is superior for $L$ out of $N$ objectives with a total improvement over the next best action of at least $\Delta Q$. Combine $L$ and $\Delta Q$ using a fuzzy system.

- **Ranking**: Define an ordering of rewards, and check low-priority rewards only if decision is not possible by high-priority rewards.
Advantages of multi-policy approaches

- Agent remains flexible to decide about goals after learning
- Constraints can be expressed by rewards
- “being dominated by” denotes a partial order which is sufficient for many RL approaches
- Non-domination instead of (greedy) maximisation
- Exploration along and across the non-dominated front
- Use several agents (could be represented by the same robot)
Multiple policies

- Convex hull

- Varying parameter approach:
  Finding a Nash-equilibrium of the returns
Policy gradient techniques to approximate the Pareto frontier

How can gradient information be derived from multi-objective sequential decision problems?

Different MORL approaches based on MO policy gradient

- radial
- Pareto following

see next three slides


Slides and source code at: http://home.dei.polimi.it/pirotta
Multi-Objective Policy Gradient (Parisi et al. 2014)

- Half Spaces
- Ascent Cone
  \[ C(\theta) = \{ l : l \cdot \nabla_\theta J_\xi(\theta) \geq 0 \} \]
- Ascent Simplex
  \[ S(\lambda, \theta) = \sum_{i=1}^{q} \lambda_i \nabla_\theta J_\xi(\theta) \]
- Pareto–Ascent Cone
  \[ S(\lambda, \theta) \cap C(\theta) \]

Null Pareto–Ascent Cone \( \Rightarrow \) (local) optimal solution
Idea: $p$ gradient ascent searches are performed, each one following a different, *uniformly spaced* direction in the *ascent simplex*.

Problem: **weak optimal** solutions.
**Phase 1:** A solution on the Pareto frontier is reached by considering a single objective.

**Phase 2:** Exploration
- Improvement step: move the solution toward one objective at a time.
- Correction step: improvement may lead the point outside the frontier. Correction moves the point again on the frontier.

**Problems:**
- Can reach deterministic policies.
- Need to reintroduce stochasticity (e.g., based on the entropy).
- Tuning of learning rate.
Remain flexible

- Applications, e.g.: Traffic control, Quality of medical service in mobile health care, robot control, network routing, grid computing.

- MARL: In Multi-Agent systems different agents may have different objectives. Different equilibria are possible, differently from the discussed approaches to MORL.


See also:

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