POMDPs and Policy Gradients
MLSS 2006, Canberra

Douglas Aberdeen

Canberra Node, RSISE Building
Australian National University

15th February 2006
Outline

1. Introduction
   - What is Reinforcement Learning?
   - Types of RL

2. Value-Methods
   - Model Based

3. Partial Observability

4. Policy-Gradient Methods
   - Model Based
   - Experience Based
Reinforcement Learning (RL) in a Nutshell

- RL can learn any function
- RL inherently handles uncertainty
  - Uncertainty in actions (the world)
  - Uncertainty in observations (sensors)
- Directly maximise criteria we care about
- RL copes with delayed feedback
  - Temporal credit assignment problem
Reinforcement Learning (RL) in a Nutshell

- RL can learn any function
- RL inherently handles uncertainty
  - Uncertainty in actions (the world)
  - Uncertainty in observations (sensors)
- Directly maximise criteria we care about
- RL copes with delayed feedback
  - Temporal credit assignment problem
Reinforcement Learning (RL) in a Nutshell

- RL can learn any function
- RL inherently handles uncertainty
  - Uncertainty in actions (the world)
  - Uncertainty in observations (sensors)
- Directly maximise criteria we care about
- RL copes with delayed feedback
  - Temporal credit assignment problem
Reinforcement Learning (RL) in a Nutshell

- RL can learn any function
- RL inherently handles uncertainty
  - Uncertainty in actions (the world)
  - Uncertainty in observations (sensors)
- Directly maximise criteria we care about
- RL copes with delayed feedback
  - Temporal credit assignment problem
### Examples

<table>
<thead>
<tr>
<th>BackGammon: TD-Gammon [12]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beat the world champion in individual games</td>
</tr>
<tr>
<td>Can learn things no human ever thought of!</td>
</tr>
<tr>
<td>TD-Gammon opening moves now used by best humans</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Australian Champion Chess Player</td>
</tr>
<tr>
<td>RL learns the evaluation function at leaves of min-max search</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Elevator Scheduling [6]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crites, Barto 1996</td>
</tr>
<tr>
<td>Optimally dispatch multiple elevators to calls</td>
</tr>
<tr>
<td>Not implemented as far as I know</td>
</tr>
</tbody>
</table>
Partially Observable Markov Decision Processes

MDP

Pr[o|s]

s

POMDP

Pr[o|ls]

Partial Observability

Pr[a|o,w]

Agent

Pr[a|o,w] ~

o

w

Pr[s’|ls,a]

Pr[s'|s,a]

r(s)

RL

The imagination driving Australia’s ICT future.
Types of RL

- MDP
- DP
- POMDP
- RL

- Model Based
- Experience

- Policy
- Value
Optimality Criteria

- The value $V(s)$ is a long-term reward from state $s$
- How do we measure long-term reward??

$$V_\infty(s) = \mathbb{E}_w \left[ \sum_{t=0}^{\infty} r(s_t) | s_0 = s \right]$$

Ill-conditioned from the decision making point of view

- Sum of discounted rewards

$$V(s) = \mathbb{E}_w \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t) | s_0 = s \right]$$

- Finite-horizon

$$V_T(s) = \mathbb{E}_w \left[ \sum_{t=0}^{T-1} r(s_t) | s_0 = s \right]$$
Criteria Continued

- Baseline reward

\[ V_B(s) = \mathbb{E}_w \left[ \sum_{t=0}^{\infty} r(s_t) - \bar{r} \right| s_0 = s \] 

Here, \( \bar{r} \) is an estimate of the Long-term average reward...

- Long-term average is intuitively appealing

\[ \bar{V}(s) = \lim_{T \to \infty} \frac{1}{T} \mathbb{E}_w \left[ \sum_{t=0}^{T-1} r(s_t) \right| s_0 = s \]
Discounted or Average?

**Ergodic MDP**
- Positive recurrent: finite return times
- Irreducible: single recurrent set of states
- Aperiodic: GCD of return times = 1

If the Markov system is *ergodic* then \( \bar{V}(s) = \eta \) for all \( s \), i.e., \( \eta \) is constant over \( s \)

Convert from discounted to long-term average

\[
\eta = (1 - \gamma) \mathbb{E}_s V(s)
\]

We focus on discounted \( V(s) \) for Value methods
Average versus Discounted

\[
\bar{V}(1) = 3.5
\]

\[
V(1) = 14.3
\]

\[
\bar{V}(4) = 3.5
\]

\[
V(4) = 19.2
\]

\[
r(s) = s
\]

\[
delta = 0.8
\]
Dynamic Programming

- How do we compute $V(s)$ for a fixed policy?
- Find fixed point $V^*(s)$ solution to Bellman’s Equation:

$$V^*(s) = r(s) + \gamma \sum_{a \in A} \sum_{s' \in S} \Pr[s'|s, a] \Pr[a|s, w] V^*(s')$$

- In matrix form with vectors $V^*$ and $r$:
  - Define stochastic transition matrix for current policy

$$P = \sum_{a \in A} \Pr[s'|s, a] \Pr[a|s, w]$$

- Now

$$V^* = r + \gamma PV^*$$

- Like shortest path algs, or Viterbi estimation
Analytic Solution

\[ V^* = r + \gamma PV^* \]
\[ V^* - \gamma PV^* = r \]
\[ (I - \gamma P)V^* = r \]
\[ V^* = (I - \gamma P)^{-1}r \]

\[ Ax = b \]

- Computes \( V(s) \) for fixed policy (fixed \( w \))
- No solution unless \( \gamma \in [0, 1) \)
- \( O(|S|^3) \) solution... not feasible
Progress...

- Q-Learning
- POMDP
- Model Based Experience
- Value Policy
- Value & Pol Iteration
- MDP
- TD
- SARSA
- Q-Learning

Experience

Model Based

Value
Partial Observability

- We have assumed so far that $o = s$, i.e., full observability
- What if $s$ is obscured? Markov assumption violated!
  - Ostrich approach (SARSA works well in practice)
  - Exact methods
  - Direct policy search: bypass values, local convergence
- Best policy may need full history

$$\Pr[a_t | o_t, a_{t-1}, o_{t-1}, \ldots, a_1, o_1]$$
Belief States

- **Belief states** sufficiently summarise history
  \[ b(s) = \Pr[s \mid o_t, a_{t-1}, o_{t-1}, \ldots, a_1, o_1] \]

- Probability of each world state computed from history
- Given belief \( b_t \) for time \( t \), can update for next action
  \[ \bar{b}_{t+1}(s') = \sum_{s' \in S} b_t(s) \Pr[s' \mid s, a_t] \]

- Now incorporate observation \( o_{t+1} \) as evidence for state \( s \)
  \[ b_{t+1}(s) = \frac{\bar{b}_{t+1}(s) \Pr[o_{t+1} \mid s]}{\sum_{o' \in \mathcal{O}} \bar{b}_{t+1} \Pr[o' \mid s]} \]

- Like HMM forward estimation
- Just updating the belief state is \( O(|S|^2) \)
Value Iteration For Belief States

- Do normal VI, but replace states with belief state $b$

$$V(b) = r(b) + \gamma \sum_b \sum_a \Pr[b'|b, a] \Pr[a|b, w] V(b')$$

- Expanding out terms involving $b$

$$V(b) = \sum_{s \in S} b(s) r(s) + \gamma \sum_{a \in A} \sum_{o \in O} \sum_{s \in S} \sum_{s' \in S} \Pr[s'|s, a] \Pr[o|s'] \Pr[a|b, w] b(s) V(b_{ao})$$

- What is $V(b)$?

$$V(b) = \max_{l \in \mathcal{L}} l^T b$$
Piecewise Linear Representation

Belief state space

common action $u$

$V(b)$

$1_0$

$1_1$

$1_2$

$1_3$

$1_4$

$b_1 = 1 - b_0$

useless hyperplane
Policy-Graph Representation

The imagination driving Australia’s ICT future.
Complexity

High Level Value Iteration for POMDPs

1. Initialise $b_0$ (uniform/set state)
2. Receive observation $o$
3. Update belief state $b$
4. Find maximising hyperplane $l$ for $b$
5. Choose action $a$
6. Generate new $l$ for each observation and future action
7. While not converged, goto 2

- Specifics generate lots of algorithms
- Number of hyperplanes grows exponentially: P-space hard
- Infinite horizon problems might need infinite hyperplanes
Approximate Value Methods for POMDPs

- Approximations usually learn value of representative belief states and interpolate to new belief states
- Belief space simplex corners are representative states
  - Most Likely State heuristic (MLS)
    \[ Q(b, a) = \arg \max_s Q(b(s), a) \]
- \(Q_{MDP}\) assumes true state is known after one more step
  \[ Q(b, a) = \sum_{s \in S} b(s)Q(s, a) \]
- **Grid Methods** distribute many belief states uniformly [5]
Progress...

The imagination driving Australia's ICT future.

Model Based Experience

Value & Pol Iteration

MDP

POMDP

Value

Policy

Exact VI

SARSA?

TD

SARSA

Q-Learning

Experience
We all know what gradient ascent is?

Value-gradient method: TD with function approximation

Policy-gradient methods learn the policy directly by estimating the gradient of a long-term reward measure with respect to the parameters $\mathbf{w}$ that describe the policy

Are there non-gradient direct policy methods?

- Search in policy space [10]
- Evolutionary algorithms [8]
- For the slides we give up the idea of belief states and work with observations $o$, i.e., $\Pr[a|o, \mathbf{w}]$
Why Policy-Gradient

Pro’s

- No divergence, even under function approximation
- Occams Razor: policies are much simpler to represent
- Consider using a neural network to estimate a value, compared to choosing an action
- Partial observability does not hurt convergence (but of course, the best long-term value might drop)
- Are we trying to learn $Q(0, left) = 0.255$, $Q(0, right) = 0.25$
  Or $Q(0, left) > Q(0, right)$
- Complexity independent of $|S|$
Why Not Policy-Gradient

Con’s

- Lost convergence to the globally optimal policy
- Lost the Bellman constraint → larger variance
- Sometimes the values carry meaning
Recall the long-term average reward

\[
\bar{V}(s) = \lim_{T \to \infty} \frac{1}{T} \mathbb{E}_w \left[ \sum_{t=0}^{T-1} r(s_t) \mid s_0 = s \right]
\]

And if the Markov system is ergodic then \(\bar{V}(s) = \eta\) for all \(s\)

We will now assume a function approximation setting.

We want to maximise \(\eta(w)\) by computing its gradient

\[
\nabla \eta(w) = \left[ \frac{\partial \eta}{w_1}, \ldots, \frac{\partial \eta}{w_P} \right]
\]

and stepping the parameters in that direction.

For example (but there are better ways to do it):

\[
w_{t+1} = w_t + \alpha \nabla \eta(w)
\]
Computing the Gradient

- Recall the reward column vector $\mathbf{r}$
- An ergodic system has a unique stationary distribution of states $\pi(w)$
- So $\eta(w) = \pi(w)^\top \mathbf{r}$
- Recall the state transition matrix under the current policy is

$$P(w) = \sum_{a \in A} \Pr[s'|s, a] \Pr[a|s, w]$$

- So $\pi(w)^\top = \pi(w)^\top P(w)$
Computing the Gradient Cont.

- We drop the explicit dependencies on $w$
- Let $e$ be a column vector of 1’s

The Gradient of the Long-Term Average Reward

$$\nabla \eta = \pi^\top (\nabla P)(I - P + e\pi^\top)^{-1}r$$

Exercise: derive this expression using

1. $\eta = \pi^\top r$ and $\pi^\top = \pi^\top P$
2. Start with $\nabla \eta = (\nabla \pi^\top)r$, and $\nabla \pi^\top = (\nabla \pi^\top)P + \pi^\top (\nabla P)$
3. $(I - P)$ is not invertible, but $(I - P + e\pi^\top)$ is
4. $(\nabla \pi^\top)e = 0$
Solution

\[ \nabla \eta = (\nabla \pi^T)r \]

and

\[ (\nabla \pi^T) = \nabla (\pi^T P) = (\nabla \pi^T)P + \pi^T (\nabla P) \]

\[ (\nabla \pi^T) - (\nabla \pi^T)P = \pi^T (\nabla P) \]

\[ (\nabla \pi^T)(I - P) = \pi^T (\nabla P) \]

Now \((I - P)\) is not invertible, but \((I - P + e\pi^T)\) is.

Also, \(\nabla \pi^T e\pi^T = 0\), so without changing the solution

\[ (\nabla \pi^T)(I - P + e\pi^T) = \pi^T (\nabla P) \]

\[ \nabla \pi^T = \pi^T (\nabla P)(I - P + e\pi^T)^{-1} \]

\[ \nabla \eta = \pi^T (\nabla P)(I - P + e\pi^T)^{-1}r \]
Using $\nabla \eta$

- If we know $P$ and $r$ we can compute $\nabla \eta$ exactly for small $P$
- $\pi$ is the first eigenvector of $P$
- If $P$ is sparse, this works well:
  - Gradient Ascent of Modelled POMDPs (GAMP) [1]
  - Found optimum policy for system with 26,000 states in 30s
- If state space is infinite, or just large, it becomes infeasible
- This expression is the basis for our experience based algorithm
Progress...

- SARSA?
- GAMP
- Exact VI
- POMDP
- MDP
- Value & Pol Iteration
- Model Based
- Value
- Experience
- TD
- SARSA
- Q-Learning

The imagination driving Australia's ICT future.
Experience Based Policy Gradient

**Problem:** No model $P$? Too many states?

**Answer:** Compute a Monte-Carlo estimate of the gradient $\nabla \eta$

$$\nabla \eta = \lim_{\beta \to 1} \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \frac{\nabla \Pr[s_{t+1}|s_t, a_t]}{\Pr[s_{t+1}|s_t, a_t]} \sum_{\tau=t+1}^{T} \beta^{\tau-t-1} r_\tau$$

- Derived by applying the Ergodic Theorem to an approximation of the true gradient [3]

$$\nabla \eta = \lim_{\beta \to 1} \pi^\top (\nabla P) V(s),$$

where

$$V(s) = \mathbb{E}_{w} \left[ \sum_{t=0}^{\infty} \beta^t r(s_t) \middle| s_0 = s \right]$$
GPOMDP($w$) (Gradient POMDP)

1. Initialise $\hat{\nabla}\eta = 0$, $T = 0$
2. Initialise world randomly
3. Get observation $o$ from world
4. Choose an action $a \sim \Pr[\cdot|o, w]$
5. Do action $a$
6. Receive reward $r$
7. $e \leftarrow \beta e + \frac{\nabla \Pr[a|o,w]}{\Pr[a|o,w]}$
8. $\hat{\nabla}\eta \leftarrow \hat{\nabla}\eta + \frac{1}{t+1} (re - \hat{\nabla}\eta)$
9. $t \leftarrow t + 1$
10. While $t < T$, goto 3
The parameter $\beta$ ensures the estimate has finite variance

$$\text{var}(\widehat{\nabla} \eta) \propto \frac{1}{T(1-\beta)}$$

So $\beta \in [0, 1)$

But as $\beta$ decreases, the bias increases

$T$ should be at least the mixing time of the Markov process

Mixing time is $T$ it would take to get a good estimate of $\pi$

This is hard to compute in general

Rule of thumb for $T$: make $T$ as large as possible

Rule of thumb for $\beta$: increase $\beta$ until gradient estimates become inconsistent
Agent must go to right to get a load, then go left to drop it.

Optimal policy: left if loaded, right otherwise.

A reactive (memoryless) policy is not sufficient.

Partial observability because agent cannot detect it is loaded.
## Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>mean η</th>
<th>max. η</th>
<th>var.</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAMP</td>
<td>2.39</td>
<td>2.50</td>
<td>0.116</td>
<td>0.22</td>
</tr>
<tr>
<td>GPOMDP</td>
<td>1.15</td>
<td>2.50</td>
<td>0.786</td>
<td>2.05</td>
</tr>
<tr>
<td>Inc. Prune.</td>
<td>2.50</td>
<td>2.50</td>
<td>0</td>
<td>3.27</td>
</tr>
<tr>
<td>Optimum</td>
<td>2.50</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Average over 100 training and testing runs
- GPOMDP $\beta = 0.8$, $T = 5000$
- Incremental Pruning is an exact POMDP value method
Natural Actor-Critic

- Current method of choice
- Combine scalability of policy-gradient with low variance of value methods

Ideas:

1. **Actor-Critic:**
   - Actor is policy-gradient learner
   - Critic learns projection of the value function
   - Critic value estimate improves actor learning

2. **Natural gradient:**
   - Use Amari’s natural gradient to accelerate convergence
   - Keep an estimate of the Fisher information matrix inverse.
   - NAC shows how to do this efficiently

Progress...

The imagination driving Australia's ICT future.
Reinforcement learning is good when:
- performance feedback is unspecific, delayed, or unpredictable
- trying to optimise a non-linear feedback system

Reinforcement learning is bad because:
- very slow to learn in large environments with weak rewards
- if you don’t have an appropriate reward, what are you learning?

Areas we have not considered:
- How can we factorise state spaces and value functions? [9]
- What happens to exact POMDP methods when $S$ is infinite? [13]
- Taking advantage of history in direct policy methods [2]
- How can we reduce variance in all methods?
- Combining experience based methods with DP methods [7]
The imagination driving Australia’s ICT future.


Matthew R. Glickman and Katia Sycara.
Evolutionary search, stochastic policies with memory, and reinforcement learning with hidden state.

Carlos Guestrin, Daphne Koller, and Ronald Parr.
Solving factored POMDPs with linear value functions.

Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony R. Cassandra.
Solving POMDPs by searching the space of finite policies.

Richard S. Sutton and Andrew G. Barto.
*Reinforcement Learning: An Introduction.*

Gerald Tesauro.
TD-Gammon, a self-teaching backgammon program, achieves master-level play.

Sebastian Thrun.
Monte Carlo POMDPs.