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

BackGammon: TD-Gammon [12]

- Beat the world champion in individual games
- Can learn things no human ever thought of!
- TD-Gammon opening moves now used by best humans

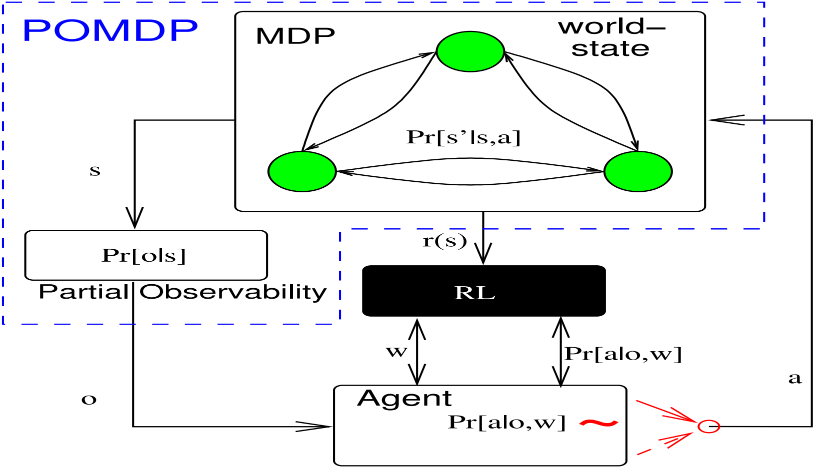
Australian Computer Chess Champion [4]

- Australian Champion Chess Player
- RL learns the evaluation function at leaves of min-max search

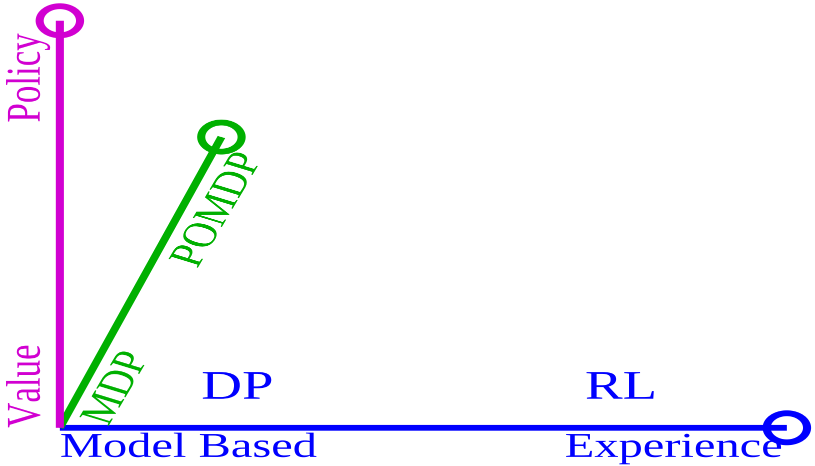
Elevator Scheduling [6]

- Crites, Barto 1996
- Optimally dispatch multiple elevators to calls
- Not implemented as far as I know

Partially Observable Markov Decision Processes



Types of RL



Optimality Criteria

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

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

Ill-conditioned from the decision making point of view

- Sum of discounted rewards

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

- Finite-horizon

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

Criteria Continued

- Baseline reward

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

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

- Long-term average is intuitively appealing

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

Discounted or Average?

Ergodic MDP

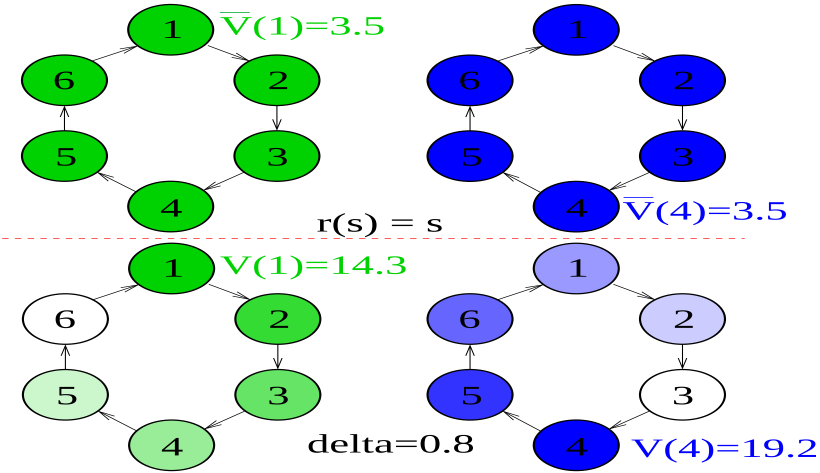
- Positive recurrent: finite return times
- Irreducible: single recurrent set of states
- Aperiodic: GCD of return times = 1

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



Dynamic Programming

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

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

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

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

  - Now

$$\mathbf{V}^* = \mathbf{r} + \gamma P \mathbf{V}^*$$

- Like shortest path algs, or Viterbi estimation

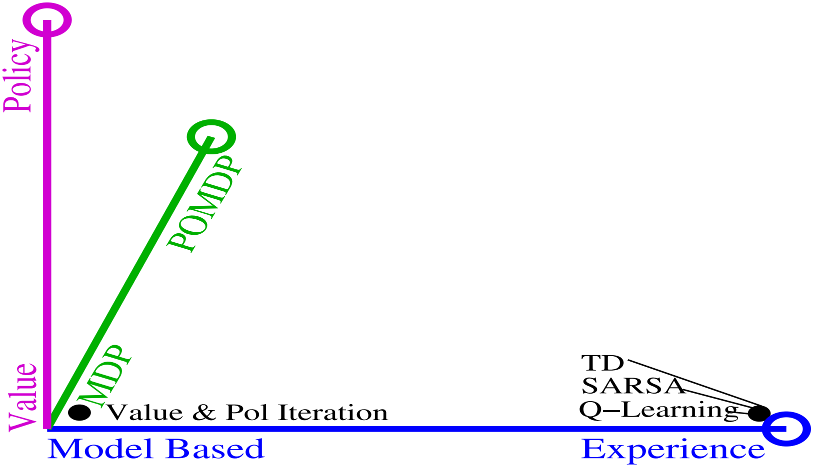
Analytic Solution

$$\begin{aligned} \mathbf{V}^* &= \mathbf{r} + \gamma P \mathbf{V}^* \\ \mathbf{V}^* - \gamma P \mathbf{V}^* &= \mathbf{r} \\ (I - \gamma P)\mathbf{V}^* &= \mathbf{r} \\ \mathbf{V}^* &= (I - \gamma P)^{-1}\mathbf{r} \end{aligned}$$

$$\boxed{A\mathbf{x} = \mathbf{b}}$$

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

Progress...


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

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

$$b(s) = \Pr[s|o_t, a_{t-1}, o_{t-1}, \ldots, a_1, o_1]$$

- Probability of each world state computed from history
- Given belief $\mathbf{b}_t$ for time t, can update for next action

$$\bar{b}_{t+1}(s') = \sum_{s' \in \mathcal{S}} b_t(s) \Pr[s'|s, a_t]$$

- Now incorporate observation o_{t+1} as evidence for state s

$$b_{t+1}(s) = \frac{\bar{b}_{t+1}(s) \Pr[o_{t+1}|s]}{\sum_{o' \in \mathcal{O}} \bar{b}_{t+1} \Pr[o'|s]}$$

- Like HMM forward estimation
- Just updating the belief state is $O(|\mathcal{S}|^2)$

Value Iteration For Belief States

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

$$V(\mathbf{b}) = r(\mathbf{b}) + \gamma \sum_{\mathbf{b}} \sum_{a} \Pr[\mathbf{b}'|\mathbf{b}, a] \Pr[a|\mathbf{b}, \mathbf{w}] V(\mathbf{b}')$$

- Expanding out terms involving b

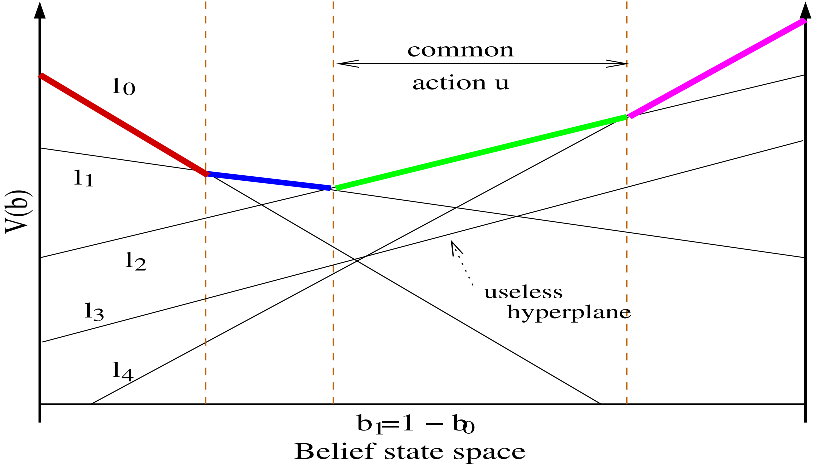
$$V(\mathbf{b}) = \sum_{s \in \mathcal{S}} b(s) r(s) +$$

$$\gamma \sum_{a \in \mathcal{A}} \sum_{o \in \mathcal{O}} \sum_{s \in \mathcal{S}} \sum_{s' \in \mathcal{S}} \Pr[s'|s, a] \Pr[o|s'] \Pr[a|\mathbf{b}, \mathbf{w}] b(s) V(\mathbf{b}_{(ao)})$$

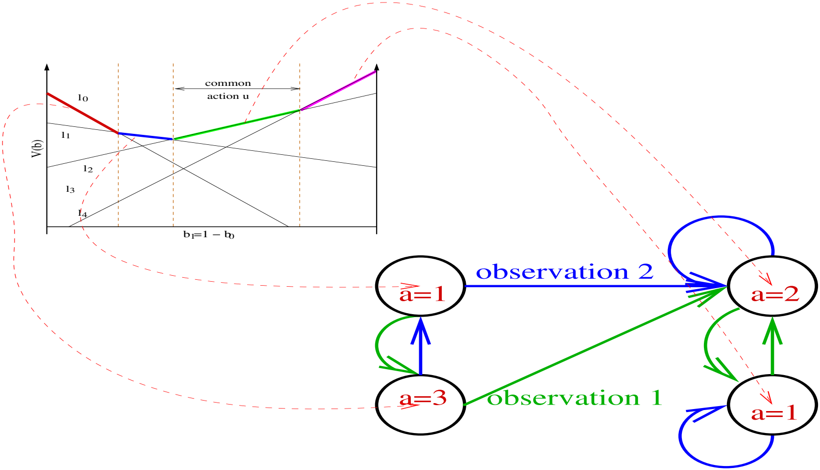
- What is $V(\mathbf{b})$?

$$V(\mathbf{b}) = \max_{\mathbf{l} \in \mathcal{L}} \mathbf{l}^\top \mathbf{b}$$

Piecewise Linear Representation



Policy-Graph Representation



Complexity

High Level Value Iteration for POMDPs

1. Initialise $\mathbf{b}_0$ (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)

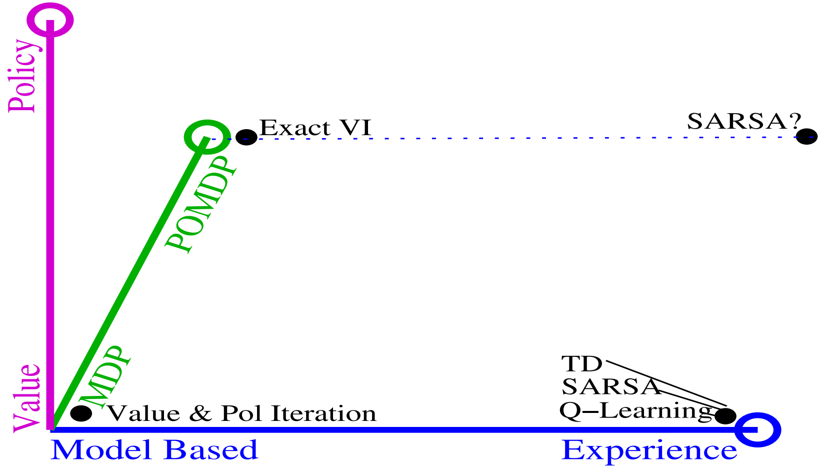
$$Q(\mathbf{b}, a) = \arg\max_{s} Q(b(s), a)$$

  - Q_{MDP} assumes true state is known after one more step

$$Q(\mathbf{b}, a) = \sum_{s \in \mathcal{S}} b(s) Q(s, a)$$

- Grid Methods distribute many belief states uniformly [5]

Progress...



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

Policy-Gradient Methods

- 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 **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 $Q(0, \mathit{left}) = 0.255$, $Q(0, \mathit{right}) = 0.25$ Or $Q(0, \mathit{left}) > Q(0, \mathit{right})$
- Complexity independent of $|\mathcal{S}|$

Why Not Policy-Gradient

Con's

- Lost convergence to the globally optimal policy
- Lost the Bellman constraint $\rightarrow$ larger variance
- Sometimes the values carry meaning

Long-Term Average Reward

- Recall the long-term average reward

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

- 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 $\eta(\mathbf{w})$ by computing its gradient

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

and stepping the parameters in that direction.
- For example (but there are better ways to do it):

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha\nabla\eta(\mathbf{w})$$

Computing the Gradient

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

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

- So $\pi(\mathbf{w})^\top = \pi(\mathbf{w})^\top P(\mathbf{w})$

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

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

- Exercise: derive this expression using
  1. $\eta = \pi^\top \mathbf{r}$ and $\pi^\top = \pi^\top P$
  2. Start with $\nabla \eta = (\nabla \pi^\top)\mathbf{r}$, and $\nabla \pi^\top = (\nabla \pi^\top)P + \pi^\top(\nabla P)$
  3. $(I - P)$ is not invertible, but $(I - P + \mathbf{e}\pi^\top)$ is
  4. $(\nabla \pi^\top)\mathbf{e} = 0$

Solution

$$\nabla\eta = (\nabla\pi^\top)\mathbf{r}$$

and

$$\begin{aligned}
(\nabla\pi^\top) &= \nabla(\pi^\top P) \\
&= (\nabla\pi^\top)P + \pi^\top(\nabla P) \\
(\nabla\pi^\top) - (\nabla\pi^\top)P &= \pi^\top(\nabla P) \\
(\nabla\pi^\top)(I - P) &= \pi^\top(\nabla P)
\end{aligned}$$

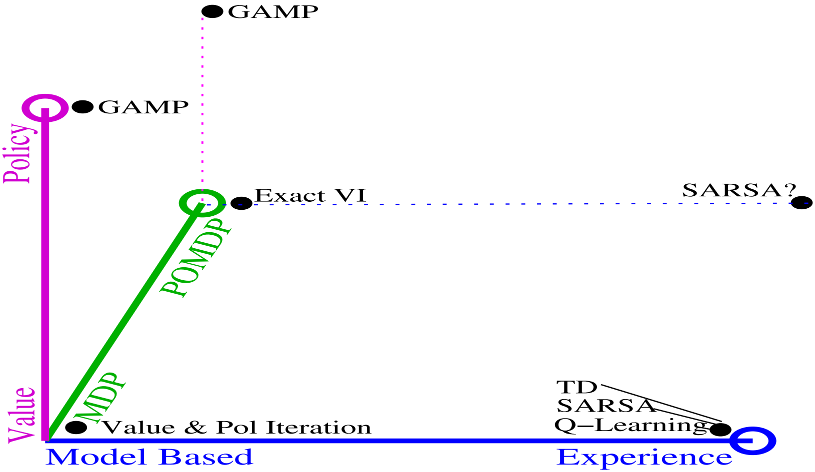
Now $(I - P)$ is not invertible, but $(I - P + \mathbf{e}\pi^\top)$ is.
Also, $\nabla\pi^\top\mathbf{e}\pi^\top = 0$, so without changing the solution

$$\begin{aligned}
(\nabla\pi^\top)(I - P + \mathbf{e}\pi^\top) &= \pi^\top(\nabla P) \\
\nabla\pi^\top &= \pi^\top(\nabla P)(I - P + \mathbf{e}\pi^\top)^{-1} \\
\nabla\eta &= \pi^\top(\nabla P)(I - P + \mathbf{e}\pi^\top)^{-1}\mathbf{r}
\end{aligned}$$

Using $\nabla\eta$

- If we know P and $\mathbf{r}$ we can compute $\nabla\eta$ exactly for small P
- π 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...


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

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

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

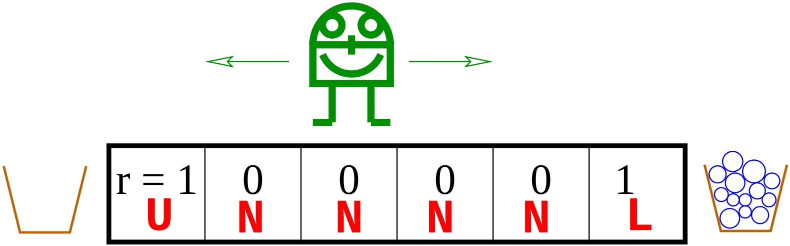
GPOMDP(**w**) (Gradient POMDP)

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

Bias-Variance Trade-Off in Policy Gradient

- The parameter β ensures the estimate has finite variance
- $var(\widehat{\nabla\eta}) \propto \frac{1}{T(1-\beta)}$
- So $\beta \in [0, 1)$
- But as β 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 π
- This is hard to compute in general
- Rule of thumb for T: make T as large as possible
- Rule of thumb for β: increase β until gradient estimates become inconsistent

Load/Unload Demonstration



- 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

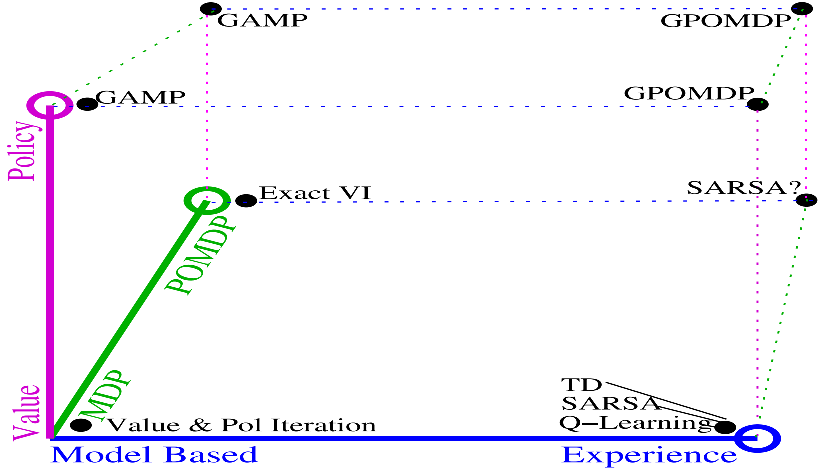
Algorithm	*mean* η	*max.* η	*var.*	*Time (s)*
GAMP	2.39	2.50	0.116	0.22
GPOMDP	1.15	2.50	0.786	2.05
Inc. Prune.	2.50	2.50	0	3.27
Optimum	2.50			

- 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
- Jan Peters, Sethu Vijayakumar, Stefan Schaal (2005), *Natural Actor-Critic*, in the Proceedings of the 16th European Conference on Machine Learning (ECML 2005).

Progress...


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

The End

- 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 $\mathcal{S}$ 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]

[1] Douglas Aberdeen and Jonathan Baxter.
Internal-state policy-gradient algorithms for infinite-horizon POMDPs.
Technical report, RSISE, Australian National University, 2002.
http://discus.anu.edu.au/~daa/papers.html.

[2] Douglas A. Aberdeen.
Policy-Gradient Algorithms for Partially Observable Markov Decision Processes.
PhD thesis, Australian National Unversity, March 2003.

[3] Jonathan Baxter and Peter L. Bartlett.
Infinite-horizon policy-gradient estimation.
Journal of Artificial Intelligence Research, 15:319–350, 2001.

[4] Jonathan Baxter, Andrew Tridgell, and Lex Weaver.
KnightCap: A chess program that learns by combining TD(λ) with game-tree search.
In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 28–36. Morgan Kaufmann, 1998.

[5] Blai Bonet.
An ϵ-optimal grid-based algorithm for partially observable Markov decision processes.
In *19th International Conference on Machine Learning*, Sydney, Australia, June 2002.

[6] Robert H. Crites and Andrew G. Barto.
Elevator group control using multiple reinforcement learning agents.
Machine Learning, 33(2-3):235–262, 1998.

[7] Héctor Geffner and Blai Bonet.
Solving large POMDPs by real time dynamic programming.
Working Notes Fall AAAI Symposium on POMDPs, 1998.
http://www.cs.ucla.edu/~bonet/.

[8] Matthew R. Glickman and Katia Sycara.
Evolutionary search, stochastic policies with memory, and reinforcement learning with hidden state.
In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 194–201. Morgan Kaufmann, June 2001.

[9] Carlos Guestrin, Daphne Koller, and Ronald Parr.
Solving factored POMDPs with linear value functions.
In *IJCAI-01 workshop on Palling under Uncertainty and Incomplete Information*, Seattle, Washington, August 2001.

[10] Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony R. Cassandra.
Solving POMDPs by searching the space of finite policies.
In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 127–136. Computer Science Dept., Brown University, Morgan Kaufmann, July 1999.

[11] Richard S. Sutton and Andrew G. Barto.
Reinforcement Learning: An Introduction.
MIT Press, Cambridge MA, 1998.
ISBN 0-262-19398-1.

[12] Gerald Tesauro.
TD-Gammon, a self-teaching backgammon program, achieves master-level play.
Neural Computation, 6:215–219, 1994.

[13] Sebastian Thrun.
Monte Carlo POMDPs.
In *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.
`http://citeseer.nj.nec.com/thrun99monte.html`.