Bayesian Reinforcement Learning

Rowan McAllister and Karolina Dziugaite

MLG RCC

21 March 2013
Outline

1 Introduction
   - Bayesian Reinforcement Learning
   - Motivating Problem

2 Planning in MDP Environments
   - Markov Decision Process
   - Action-Values

3 Reinforcement Learning
   - Q-learning

4 Bayesian Reinforcement Learning (Model-Based)
   - Bayesian RL as a POMDP
   - Bayesian Inference on Beliefs
   - Value Optimisation
Bayesian Reinforcement Learning - what is it?

Bayesian RL is about capturing and dealing with uncertainty, where ‘classic RL’ does not. Research in Bayesian RL includes modelling the transition-function, or value-function, policy, reward function probabilistically.

Differences over ‘classic RL’:

- Resolves exploitation & exploration dilemma by planning in belief space.
- Computationally intractable in general, but approximations exist.
- Uses and chooses samples to learn from efficiently, suitable when sample cost is high, e.g. robot motion.

many slides use ideas from Goel’s MS&E235 lecture, Poupart’s ICML 2007 tutorial, Littman’s MLSS ‘09 slides
Bayesian Reinforcement Learning - what is it?

Bayesian RL is about capturing and dealing with uncertainty, where ‘classic RL’ does not. Research in Bayesian RL includes modelling the transition-function, or value-function, policy, reward function probabilistically.

Differences over ‘classic RL’:

- Resolves exploitation & exploration dilemma by planning in belief space.
- Computationally intractable in general, but approximations exist.
- Uses and chooses samples to learn from efficiently, suitable when sample cost is high, e.g. robot motion.

*many slides use ideas from Goel’s MS&E235 lecture, Poupart’s ICML 2007 tutorial, Littman’s MLSS ‘09 slides*
Motivating Problem: Two armed bandit (1)

- You have $n$ tokens, which may be used in one of two slot machines.
Motivating Problem: Two armed bandit (1)

- You have $n$ tokens, which may be used in one of two slot machines.
- The $i$'th machine returns $0$ or $1$ based on a fixed yet unknown probability $p_i \in [0, 1]$
Motivating Problem: Two armed bandit (1)

- You have \( n \) tokens, which may be used in one of two slot machines.
- The \( i \)'th machine returns $0 or $1 based on a fixed yet unknown probability \( p_i \in [0, 1] \)
- Objective: maximise your winnings.
Motivating Problem: Two armed bandit (2)

As you play, you record what you see, formatted as (\#wins, \#losses). Your current records shows:
Arm 1: (1,2)
Arm 2: (21,19)
Motivating Problem: Two armed bandit (2)

- As you play, you record what you see, formatted as (#wins, #losses). Your current records shows:
  Arm 1: (1,2)
  Arm 2: (21,19)

- Which machine would you play next if you have 1 token remaining?
Motivating Problem: Two armed bandit (2)

- As you play, you record what you see, formatted as (#wins, #losses).
  Your current records shows:
  Arm 1: (1,2)
  Arm 2: (21,19)

- Which machine would you play next if you have 1 token remaining?

- How about if you have 100 tokens remaining?
Motivating Problem: Two armed bandit (3)

‘Classic’ Reinforcement Learning mentality: Two action classes (not mutually exclusive):

**Exploit**
Select action of greatest expected return given current belief on reward probabilities. i.e. select best action according to best guess of underlying MDP: MLE or MAP → select Arm #2!

**Explore**
Select random action to increase our certainty of underlying MDP. This may lead to higher returns when exploiting in the future.
Motivating Problem: Two armed bandit (3)

‘Classic’ Reinforcement Learning mentality:
Two action classes (not mutually exclusive):

**Exploit**
Select action of greatest expected return given current belief on reward probabilities. i.e. select best action according to best guess of underlying MDP: MLE or MAP → select Arm #2!

**Explore**
Select random action to increase our certainty of underlying MDP. This may lead to higher returns when exploiting in the future.

→ Dilemma (?): how to choose between exploitation and exploration? Seems like comparing apples and oranges...
Many heuristics exist, but is there a principled approach?
Motivating Problem: Two armed bandit (4)

Steps towards resolving ‘exploitation’ vs ‘exploration’:
model future beliefs in Arm 1 (#wins, #losses):

(3,2)
\[ \frac{1}{2} \]

(2,2)
\[ \frac{1}{2} \]
\[ \frac{1}{3} \]

(1,2)
\[ \frac{2}{3} \]

(2,3)

(1,3)
Note: value of exploration depends on how much we can exploit that information gain later, i.e. # tokens remaining. Alternatively, with infinite tokens and discount rate \( \gamma \), effective horizon \( \propto -1 \log(\gamma) \).
Motivating Problem: Two armed bandit (4)

Steps towards resolving ‘exploitation’ vs ‘exploration’:
model future beliefs in Arm 1 (#wins, #losses):

\[(3,2) \leftarrow \text{higher expectation of rewards in this potential future!}\]

\[(1,2) \quad (2,2) \quad (2,3) \]

\[\frac{1}{3} \quad \frac{1}{2} \quad \frac{1}{2}\]

\[\frac{2}{3}\]

\[(1,3)\]
Motivating Problem: Two armed bandit (4)

Steps towards resolving ‘exploitation’ vs ‘exploration’:
model future beliefs in Arm 1 (#wins, #losses):

1. Higher expectation of rewards in this potential future!

2. We can plan in this space, and compute expected additional rewards gained from exploratory actions.
Motivating Problem: Two armed bandit (4)

Steps towards resolving ‘exploitation’ vs ‘exploration’:
model future beliefs in Arm 1 (#wins, #losses):

\[
\begin{align*}
(1,2) & \xrightarrow{\frac{2}{3}} (1,3) \\
(2,2) & \xrightarrow{\frac{1}{2}} (3,2) \\
(2,3) & \xrightarrow{\frac{1}{3}} (2,2)
\end{align*}
\]

← higher expectation of rewards in this potential future!

we can plan in this space, and compute expected additional rewards gained from exploratory actions.

Note: value of exploration depends on how much we can exploit that information gain later, i.e. # tokens remaining. Alternatively, with infinite tokens and discount rate \( \gamma \) on future rewards, effective horizon \( \propto \frac{-1}{\log(\gamma)} \).
Planning in MDP Environments
Planning overview

- Environment: a *familiar* MDP (we can simulate interaction with the world accurately).
- Goal: compute a policy that maximises expected long-term discounted rewards over a horizon (episodic or continual).
Markov Decision Process

\( S \), set of states \( s \)

\( A \), set of action \( a \)

\( \pi : S \rightarrow A \), the policy, a mapping from state \( s \) to action \( a \)

\( T(s, a, s') = P(s'|s, a) \in [0, 1] \), transition probability, that state \( s' \) is reached by executing action \( a \) from state \( s \)

\( R(s, a, s') \in \mathbb{R} \), a reward distribution. An agent receives a reward drawn from this when taking action \( a \) from state \( s \) reaching state \( s' \)
Markov Decision Process

\( \mathcal{S} \), set of states \( s \)

\( \mathcal{A} \), set of action \( a \)

\( \pi : \mathcal{S} \rightarrow \mathcal{A} \), the policy, a mapping from state \( s \) to action \( a \)

**System dynamics**

\( T(s, a, s') = P(s' | s, a) \in [0, 1] \), transition probability, that state \( s' \) is reached by executing action \( a \) from state \( s \)

\( R(s, a, s') \in \mathbb{R} \), a reward distribution. An agent receives a reward drawn from this when taking action \( a \) from state \( s \) reaching state \( s' \)
Robot planning example

**Goal:** traverse to human-specified goal location ‘safely’

**States:** physical space \((x, y, yaw)\)

**Action:** move forward, left, spin anticlockwise, etc.

**Rewards:** ‘dangerousness’ of each \((s, a)\) motion primitives

(a) Path Planning Scenario  
(b) Rewards  
(c) Policy
Rewards

A measure of *desirability* of the agent being in a particular state. Use to encode *what* we want the agent to achieve, not *how*.

example: Agent learns to play chess:
Don’t reward agent for capturing opponent’s queen, only for winning.
(don’t want agent discovering novel ways to capture queens at expense of losing games!)
Rewards

A measure of *desirability* of the agent being in a particular state. Use to encode *what* we want the agent to achieve, not *how*.

example: Agent learns to play chess:
Don’t reward agent for capturing opponent’s queen, only for winning.
(don’t want agent discovering novel ways to capture queens at expense of losing games!)

Caveat: Reward *shaping*, the modification of reward function to give partial credit without affecting the optimal policy (much), can be important in practice.
Optimal Action-Value Function

Optimal action value: expectation of all future discounted rewards from taking action $a$ from state $s$, assuming subsequent actions chosen by the optimal policy $\pi^*$. It can be re-expressed as a recursive relationship.

$$Q^*(s, a) = E_{\pi^*}\left\{ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a \right\}$$

$$= E_{\pi^*}\left\{ R(s_0, a_0) + \gamma \sum_{t=0}^{\infty} \gamma^t R(s_{t+1}, a_{t+1}) | s_0 = s, a_0 = a \right\}$$

$$= \bar{R}(s, a) + \gamma E_{\pi^*}\left\{ \max_{a'} Q^*(s_{t+1}, a') | s_0 = s, a_0 = a \right\}$$

$$= \bar{R}(s, a) + \gamma \sum_{s'} T(s, a, s')[\max_{a'} Q^*(s', a')]$$
Planning in MDP Environments

Action-Value Optimisation

Need to satisfy the Bellman Optimality Equation:

$$Q^*(s, a) = \bar{R}(s, a) + \gamma \sum_{s'} T(s, a, s') \max_{a'} Q^*(s', a')$$

$$\pi^* = \arg\max_a Q^*(s, a)$$

An algorithm to compute $Q^*(s, a)$ is value iteration: for all $s \in S$ repeat until convergence:

$$Q_{t+1}(s, a) \leftarrow \bar{R}(s, a) + \gamma \sum_{s'} T(s, a, s') \max_{a'} Q_t(s', a')$$
Reinforcement Learning
**Reinforcement learning overview**

- **Environment**: an *unfamiliar* MDP \( T(s, a, s') \) and/or \( \tilde{R}(s, a) \) unknown) and possibly dynamic / changing.

- **Consequence**: agent cannot simulate interaction with world in advance, to predict future outcomes. Instead, the optimal policy is learned through sequential interaction and evaluative feedback.

- **Goal**: same as planning (compute a policy that maximises expected long-term discounted rewards over a horizon).
**Q-learning**

With **known** environmental models $\bar{R}(s, a)$ and $T(s, a, s')$, $Q$'s computed iteratively using value iteration (e.g. planning): :

$$Q_{t+1}(s, a) \leftarrow \bar{R}(s, a) + \gamma \sum_{s'} T(s, a, s') \max_{a'} Q_t(s', a')$$

**Q-learning**

With **unknown** environmental models, $Q$’s computed as *point estimates*: on experience $\{s_t, a_t, r_t, s_{t+1}\}$:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t (R_{t+1} + \gamma \max_{a'} (Q(s_{t+1}, a')) - Q(s_t, a_t))$$

if $\{s, a\}$ visited infinitely often, $\sum_t \alpha_t = \infty$, $\sum_t \alpha_t^2 < \infty$, then $Q$ will converge to $Q^*$ (independent of policy being followed!).
When to explore?: Heuristic approach to action selection

A couple heuristic examples agents use for action selection, to mostly exploit and sometimes and explore:

- **$\epsilon$-greedy:**
  \[
  \pi(a|s) = \begin{cases} 
  (1 - \epsilon), & \text{if } a = \text{argmax}_a Q_t(s, a) \\
  \epsilon/|A|, & \text{if } a \neq \text{argmax}_a Q_t(s, a)
  \end{cases}
  \]
  e.g. $\epsilon = 5\%$

- **Softmax:**
  \[
  \pi(a|s) = \frac{e^{Q_t(s,a)/\tau}}{\sum_i e^{Q_t(s,i)/\tau}}
  \]
  i.e. biased towards more fruitful actions. $\tau$ is a crank for more frequent exploration.
When to explore?: Heuristic approach to action selection

A couple heuristic examples agents use for action selection, to mostly exploit and sometimes and explore:

- $\epsilon$-greedy:

  $$\pi(a|s) = \begin{cases} 
  (1 - \epsilon), & \text{if } a = \arg\max_a Q_t(s, a) \\
  \epsilon/|A|, & \text{if } a \neq \arg\max_a Q_t(s, a) 
  \end{cases}$$

  e.g. $\epsilon = 5\%$

- Softmax: $\pi(a|s) = \frac{e^{Q_t(s,a)/\tau}}{\sum_i e^{Q_t(s,i)/\tau}}$

  i.e. biased towards more fruitful actions. $\tau$ is a crank for more frequent exploration.

Note: often, heuristics are too inefficient for online learning! We wish to minimise wasteful exploration.
Bayesian Reinforcement Learning (Model-Based)
Brief Description

- Start with a prior over transition probabilities $T(s, a, s')$, maintain the posterior (update them) as evidence comes in.
- Now we can reason about more/less likely MDPs, instead of just possible MDPs or a single ‘best guess’ MDP.
- Can plan in the space of posteriors to:
  - evaluate the likelihood of any possible outcome of an action.
  - model how that outcome will change the posterior.
Motivation (1)

Resolves ‘classic’ RL dilemma:

- maximise immediate rewards (exploit), or
- maximise info gain (explore)?

Wrong question!

→ Single objective: maximise expected rewards up to the horizon (as a weighted average over the possible futures).
(implicitly trades-off exploration with exploitation optimally)
Motivation (2)

More Pros:

- Prior information is easily used, can start planning straight away by running a full backup.
- Easy to explicit encoding of prior knowledge / domain assumptions.
- Easy to update belief if using conjugate priors, as we collect evidence.

Cons:

- Computationally intractable except in special cases (bandits, short horizons)
Bayesian RL as a POMDP (1)

- Let $\theta_{sas'}$ denotes unknown MDP parameter
  $T(s, a, s') = P(s'|s, a) \in [0, 1]$
  Let $b(\theta)$ be the agent’s prior belief over all unknown parameters $\theta_{sas'}$

- [Duff 2002]: Define hybrid state: $S_p = S$ (certain) $\times \theta_{sas'}$ (uncertain).
  Cast Bayesian RL as a Partially Observable Markov Decision Process (POMDP) $\mathcal{P} = \langle S_p, A_p, O_p, T_p, Z_p, R_p, \gamma, b_p^0 \rangle$

- Use favourite POMDP solution technique. This provides a Bayes Optimal policy in our original state space.
Bayesian RL as a POMDP (2)

\[
S_p = S \times \theta, \text{ hybrid states of known } S \text{ and all unknown } \theta_{sas}', \\
A_p = A, \text{ original action set (unchanged)} \\
O_p = S : \text{ observation space}
\]
Bayesian RL as a POMDP (2)

\[ S_p = S \times \theta, \text{ hybrid states of known } S \text{ and all unknown } \theta_{sas}' \]
\[ A_p = A, \text{ original action set (unchanged) } \]
\[ O_p = S : \text{ observation space } \]

\[
T_p(s, \theta_{sas'}, a, s', \theta'_{sas'}) = P(s', \theta'_{sas'}|s, \theta_{sas'}, a)
\]
\[
= P(\theta'_{sas'}|\theta_{sas'})P(s'|s, \theta_{sas'}, a)
\]
\[
= \delta(\theta'_{sas'} - \theta_{sas'})\theta_{sas'}, \text{ assuming } \theta_{sas'} \text{ is stationary }
\]

\[
R_p(s, \theta_{sas'}, a, s', \theta'_{sas'}) = R(s, a, s')
\]

\[
Z_p(s', \theta'_{sas'}, a, o) = P(o|s', \theta'_{sas'}, a) = \delta(o - s'), \text{ as observation is } s'
\]

\( T(.) \): transition probability (known), \( R(.) \): reward distribution, \( Z(.) \): observation function
Bayesian Inference

let $b(\theta)$ be the agent’s current (prior) belief over all unknown parameters $\theta_{sas'}$. For each $\{s, a, s'\}$ transition observed, the belief is updated accordingly:

$$
\begin{align*}
bsas'(\theta) & \propto b(\theta)P(s'|\theta_{sas'}, s, a) \\
& = b(\theta)\theta_{sas'} \\
\text{(posterior)} & \propto \text{(prior) \times (likelihood)}
\end{align*}
$$
Common Prior: Dirichlet Distribution

$$Dir(\theta_{sa}; n_{sa}) = \frac{1}{B(n_{sa})} \prod_{s'} (\theta_{sas'})^{n_{sas'}} - 1$$

suitable for discrete state spaces

The Dirichlet distribution is a conjugate prior to a multinomial likelihood distribution (counts $\# a$ from $s$ reached $s'$). Thus easy closed for Bayes updates.
Bayesian Inference: Discrete MDPs

For discrete MDPs, we can define $\theta_{sa} = P(.|s, a)$, as a multinomial.

Choosing prior $b(\theta)$ form as a product of Dirichlets

$$\prod_{s,a} \text{Dir}(\theta_{sa}; n_{sa}) \propto \prod_{s,a} \prod_{s'} (\theta_{sas'})^{n_{sas'}-1},$$

the posterior / updated-belief retains the same form:

$$b_{sas'}(\theta) \propto (\prod_{\hat{s},\hat{a}} \text{Dir}(\theta_{\hat{s}\hat{a}}; n_{\hat{s}\hat{a}})) \theta_{sas'}$$

$$\propto \prod_{\hat{s},\hat{a}} \text{Dir}(\theta_{\hat{s}\hat{a}}; n_{\hat{s}\hat{a}} + \delta_{\hat{s},\hat{a},s'}(s, a, s'))$$  \(1\)

(where $n_{sa}$ is a vector of hyperparameters $n_{sas'}$, the \# \{s, a, s’\} transitions observed)
Bayesian Inference: Discrete MDPs

For discrete MDPs, we can define $\theta_{sa} = P(.|s, a)$, as a multinomial.

Choosing prior $b(\theta)$ form as a product of Dirichlets

$$
\prod_{s,a} \text{Dir}(\theta_{sa}; n_{sa}) \propto \prod_{s,a} \prod_{s'} (\theta_{sas'}^{-1}),
$$

the posterior / updated-belief retains the same form:

$$
b_{sas'}(\theta) \propto b(\theta)\theta_{sas'}
$$

$$(\prod_{\hat{s}, \hat{a}} \text{Dir}(\theta_{\hat{s}\hat{a}}; n_{\hat{s}\hat{a}}))\theta_{sas'}
$$

$$
\propto \prod_{\hat{s}, \hat{a}} \text{Dir}(\theta_{\hat{s}\hat{a}}; n_{\hat{s}\hat{a}} + \delta_{\hat{s}, \hat{a}, \hat{s}'}(s, a, s'))
$$

(1)

(1) (where $n_{sa}$ is a vector of hyperparameters $n_{sas'}$, the # $\{s, a, s'\}$ transitions observed)

$\rightarrow$ So belief updated by incrementing corresponding $n_{sas'}$
Factoring structural priors

Can transition dynamics can be jointly expressed as a function of a smaller number of parameters?

Parameter *tying* is a special case of knowing $\theta_{sas'} = \theta_{\hat{s}\hat{a}\hat{s}'}$.

- realistic, real-life action outcomes from one state often generalise
- useful, speeds up convergence / less trials required $\rightarrow$ mitigates expensive hardware collisions etc.
Factoring structural priors: Example (1)

Taxi example: [Dietterich 1998]

- **Goal**: pick up passenger and drop at destination.
- **States**: 25 taxi location $\times$ 4 pickup locations $\times$ 4 dropoff destinations
- **Actions**: N, S, E, W, pickup, dropoff
- **Rewards**: +20 for successful delivery of passenger, -10 for illegal pickup or dropoff, -1 otherwise.

$\# \theta_{sa} = |S| \times |A| = 400 \times 6 = 2400$.

\[\begin{array}{c|c|c|c|c|c|c}
| & | & | & | & | \\
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 0 | 1 | 2 | 3 | 4 | \\
\end{array}\]

**Figure**: possible pickup, dropoff locations: R, Y, G, B
Factoring structural priors: Example (2)

We can factor $\theta_{sa}$: We know \textit{a priori} that navigation to pickup location is independent of dropoff-destination! Furthermore, navigation task is independent of purpose (pickup or dropoff).

$\rightarrow$ $\#$ states required to learn navigation: $25 \times 4 = 100 < 400$.

Using a factored DBN model to generalises transitions for multiple states, we quarter the $\#$ of $\theta_{sa}$ to learn.
Value Optimisation

Classic RL Bellman equation:
\[ Q^*(s, a) = \sum_{s'} P(s'|s, a)[R(s, a, s') + \gamma \max_{a'} Q^*(s', a')] \]

POMDP Bellman equation, in BRL context:
\[ Q^*(s, b, a) = \sum_{s'} P(s'|s, b, a)[R(s, a, s') + \gamma \max_{a'} Q^*(s', b_{sas'}, a')] \]

The Bayes-optimal policy is \[ \pi^*(s, b) = \arg\max_a Q^*(s, b, a) \], which maximises the predicted reward up to the horizon, over a weighted average of all the possible futures.
Big Picture

Task: solve:

\[ Q^*(s, b, a) = \sum_{s'} P(s'|s, b, a)[R(s, a, s') + \gamma \max_{a'} Q^*(s', b_{sas'}, a')] \]

Challenge: Size of \( s \times b_{sas'} \) space grows exponentially with number of \( \theta_{sas'} \) parameters \( \rightarrow \) Bayes-Optimal solution intractable.

Solutions: approximate \( Q^*(s, b, a) \) via:

- discretisation
- exploration bonuses [BEB, Kolter 2009]
- myopic value of info [Bayesian Q-learning, Dearden 1999]
- sample beliefs [Bayesian Forward Search Sparse Sampling, Littman 2012]
- sample MDPs, update occasionally [Thompson Sampling, Strens 2000]
Algorithm: BEETLE (1)

[Poupart et al. 2006]
Exploits piecewise linear and convex property of POMDP value function [Sondik 1971].

- Sample a set of reachable \((s, b)\) pairs by simulating a random policy.
- Uses Point Based Value Iteration (PBVI) [Pineau 2003] to approximate value iteration, by tracking value + derivative of sampled belief points. Proves \(\alpha\)-functions (one per sampled belief) in Bayesian RL are a set of multivariate polynomials of \(\theta_{sas}'\), and \(V_s^*(\theta) = \max_i poly_i(\theta)\).
- Scalable, has a closed form value representation under Bellman backups.

![Figure 1: POMDP value function representation using PBVI (on the left) and a grid (on the right).](image-url)
Algorithm: BEETLE (2)

**Figure 1.** The “Chain” problem

Table 1. Expected total reward for chain and handwashing problems. na-m indicates insufficient memory.

| problem     | |S| |A| free params | optimal (utopic) | discrete POMDP | exploit | Beetle | Beetle time (minutes) | precomputation | optimization |
|-------------|------------------------|-----------------|------------------|------------------|----------------|-------------------|---------|--------|----------------------|----------------|--------------|
| chain_tied  | 5                      | 2               | 1                | 3677             | 3661 ± 27      | 3642 ± 43        | 3650 ± 41 | 0.4    | 1.5                  |                |              |
| chain_semi  | 5                      | 2               | 2                | 3677             | 3651 ± 32      | 3257 ± 124       | 3648 ± 41 | 1.3    | 1.3                  |                |              |
| chain_full  | 5                      | 2               | 40               | 3677             | na-m           | 3078 ± 49        | 1754 ± 42 | 14.8   | 18.0                 |                |              |
| handw_tied  | 9                      | 2               | 4                | 1153             | 1149 ± 12      | 1133 ± 12        | 1146 ± 12 | 2.6    | 11.8                 |                |              |
| handw_semi  | 9                      | 2               | 8                | 1153             | 990 ± 8        | 991 ± 31         | 1082 ± 17 | 3.4    | 52.3                 |                |              |
| handw_full  | 9                      | 6               | 270              | 1083             | na-m           | 297 ± 10         | 385 ± 10  | 125.3  | 8.3                  |                |              |

**Figure: [Strens 2002]**

**Figure: [Poupart 2006]**
Model Based Interval Estimation with Exploration Bonus MBIE-EB (PAC-MDP) and Bayesian Exploration Bonus BEB (Bayesian RL) will be compared. Both:

- count how many times each transition \((s, a, s')\) has happened: 
  \(\alpha(s, a, s')\);
- use counts to produce an estimate of the underlying MDP;
- add exploration bonus to the reward for \((s, a)\) pair if not visited enough;
- act greedily with respect to this modified MDP.
Bellman’s optimality equations with exploration bonus

Let $\alpha_0(s, a) = \sum_{s'} \alpha(s, a, s')$ and $b = \{\alpha(s, a, s')\}$. Then

$$P(s' \mid b, s, a) = \frac{\alpha(s, a, s')}{\alpha_0(s, a)}$$

Attempts to maximize:

- **BEB**

$$\tilde{V}_H^*(b, s) = \max_a \left\{ R(s, a) + \frac{\beta}{1 + \alpha_0(s, a)} \right. \right.$$ 

$$\left. + \sum_{s'} P(s' \mid b, s, a) \tilde{V}_{H-1}^*(s') \right\}$$

- **MBIE-EB**

$$\tilde{V}_H^*(s) = \max_a \left\{ R(s, a) + \frac{\beta}{\sqrt{\alpha_0(s, a)}} \right. \right.$$ 

$$\left. + \sum_{s'} P(s' \mid b, s, a) \tilde{V}_{H-1}^*(b, s') \right\}$$
**Approximate Bayes-Optimal:**

If \( \mathcal{A}_t \) denotes the policy followed by the algorithm at time \( t \), then with probability greater than \( 1 - \delta \)

\[
V_t^A(b_t, s_t) \geq V^*(b_t, s_t) - \epsilon
\]

where \( V^*(b, s) \) is the value function for a Bayes-optimal strategy.

**Near Bayes Optimal:**

With probability \( \geq 1 - \delta \), an agent follows an approximate Bayes-optimal policy for all but a “small” number of steps, which is polynomial in quantities representing the system.
Theorem (Kolter and Ng, 2009)

Let $A_t$ denote the policy followed by the BEB algorithm (with $\beta = 2H^2$) at time $t$, and let $s_t$ and $b_t$ be the corresponding state and belief. Also suppose we stop updating the belief for a state-action pair when $\alpha_0(a,s) > 4H^3/\epsilon$. Then with probability at least $1 - \delta$,

$$V_{H}^{A_t}(b_t, s_t) \geq V_{H}^{\ast}(b_t, s_t) - \epsilon$$

i.e, the algorithm is $\epsilon$-close to the optimal Bayesian policy for all but

$$m = O\left(\frac{|S||A|H^6}{\epsilon^2} \log \frac{|S||A|}{\delta}\right)$$

time steps.
Theorem (Strehl, Li and Littman 2006)

Let $A_t$ denote the policy followed by some algorithm. Also, let the algorithm satisfy the following properties, for some input $\epsilon$:

- acts greedily for every time step $t$;
- is optimistic ($V_t(s) \geq V^*_t(s) - \epsilon$)
- has bounded learning complexity (bounded number of action-value estimate updates and number of escape events)
- is accurate ($V_t(s) - V^\pi_{MKt}(s) \leq \epsilon$)

Then, with probability greater than $1 - \delta$, for all but

$$\tilde{O}\left(\frac{|S|^2|A|H^6}{\epsilon^2}\right)$$

time steps, the algorithm follows an $4\epsilon$ optimal policy.
Theorem (Kolter and Ng, 2009)

Let $A_t$ denote the policy followed an algorithm using any (arbitrary complex) exploration bonus that is upper bounded by

$$\frac{\beta}{\alpha_0(s, a)^p}$$

for some constant $\beta$ and $p > 1/2$. Then $\exists$ some MDP $M$ and $\epsilon_0(\beta, p)$, s.t. with probability greater than $\delta_0 = 0.15$,

$$V^{A_t}_H(s_t) < V^*_H(s_t) - \epsilon_0$$

will hold for an unbounded number of steps.
The proof uses the following inequality.

**Lemma (Slud’s inequality)**

Let $X_1, \ldots, X_n$ be i.i.d. Bernoulli random variables, with mean $\mu > 3/4$. Then

$$P \left( \mu - \frac{1}{n} \sum_{i=1}^{n} X_i > \epsilon \right) \geq 1 - \Phi \left( \frac{\epsilon \sqrt{n}}{\sqrt{\mu(1-\mu)}} \right)$$
Proof

The lower bound on the probability that the algorithm’s estimate of the reward for playing $a_1$ plus the exploration bonus is pessimistic by at least $\beta/n^p$:

$$
P \left(3/4 - \frac{1}{n} \sum_{i=1}^{n} r_i - f(n) \geq \frac{\beta}{n^p}\right) \\
\geq P \left(3/4 - \frac{1}{n} \sum_{i=1}^{n} r_i \geq \frac{2\beta}{n^p}\right) \\
\geq 1 - \Phi \left(\frac{8\beta}{\sqrt{3}n^p-1/2}\right)
$$
Proof

Set

\[ n \geq \left( \frac{8\beta}{\sqrt{3}} \right)^{\frac{2}{2p-1}} \]

and

\[ \epsilon_0(\beta, p) = \beta / \left( \left( \frac{8\beta}{\sqrt{3}} \right)^{\frac{2p}{2p-1}} \right) \]

So at stage \( n \) with probability at least 0.15, action \( a_2 \) will be preferred over \( a_1 \) and the agent will stop exploring \( \Rightarrow \) the algorithm will be more than \( \epsilon \) suboptimal for an infinite number of steps, for any \( \epsilon \geq \epsilon_0 \).
Conclusions

- Both algorithms use the same intuition: in order to perform well, we want to explore enough that we learn an accurate model of the system;
- For PAC-MDP, exploration bonus cannot shrink at a rate faster than $\frac{1}{2}$ or they fail to be near optimal, and slow rate of decay results in more exploration;
- BEB reduces the amount of exploration needed, which allows us to achieve lower sample complexity and use greedier exploration method;
- A near Bayesian optimal policy is not near-optimal: the optimality is considered with respect to the Bayesian policy, rather than the optimal policy for some fixed MDP.