Decision Making in Robots and Autonomous Agents

Stochastic System Models: How should a robot reason about uncertainty?

Subramanian Ramamoorthy
School of Informatics

6 February, 2018
Lecture Objectives

This lecture has three objectives:

• Introduce you to the notion of reasoning about the distribution of payoffs – using the simplest example of a decision problem, the Multi-armed bandit (MAB)

• Extend this solution concept to address the computation of an optimal policy for the MDP (generalizing what we saw two lectures ago) – key concept being the Bellman equation

• Introduce the computational procedures of Value and Policy Iteration, along with a simple example

Remark: If you are also registered in RL, this lecture will have overlap of content
Agent (e.g., reinforcement learning algorithm) is:

- Temporally situated
- Continual learning and planning
- Objective is to affect the environment – actions and states
- Environment is uncertain, stochastic
Multi-arm Bandits (MAB)

- $N$ possible actions
- You can play for some period of time and you want to maximize reward (expected utility)

Which is the best arm/machine?

DEMO
Numerous Applications!

- Computer Go
- Brain computer interface
- Medical trials

- Packets routing
- Ads placement
- Dynamic allocation
What is the Choice?

| t=1 | 0.3 | 0.2 | 0.8 | 0.4 | 0.0 |
| t=2 | 0.7 | 0.1 | 0.9 | 0.5 | 0.1 |
| t=3 | 0.5 | 0.3 | 0.7 | 0.3 | 0.3 |

...
The *n*-armed Bandit Problem

- Choose repeatedly from one of *n* actions; each choice is called a *play*.
- After each play *a*\(_t\)*, you get a reward *r*\(_t\)*, where

\[
E \{r_t \mid a_t\} = Q^*(a_t)
\]

These are unknown *action values*. Distribution of *r*\(_t\)* depends only on *a*\(_t\)*.

Objective is to maximize the reward in the long term, e.g., over 1000 plays.

To solve the *n*-armed bandit problem, you must **explore** a variety of actions and **exploit** the best of them.

**Question:** How does this relate to the f (cost to go) in earlier DP lecture?
Exploration/Exploitation Dilemma

- Suppose you form estimates
  \[ Q_t(a) \approx Q^*(a) \]

- The **greedy** action at time \( t \) is \( a_t^* \)
  \[ a_t^* = \arg\max_a Q_t(a) \]
  \[ a_t = a_t^* \Rightarrow \text{exploitation} \]
  \[ a_t \neq a_t^* \Rightarrow \text{exploration} \]

Why?
- You can’t exploit all the time; you can’t explore all the time
- You can never stop exploring; but you could reduce exploring.
Action-Value Methods

- Methods that adapt action-value estimates and nothing else, e.g.: suppose by the \( t \)-th play, action \( a \) had been chosen \( k_a \) times, producing rewards \( r_1, r_2, \ldots, r_{k_a} \), then

\[
Q_t(a) = \frac{r_1 + r_2 + \cdots + r_{k_a}}{k_a}
\]

“sample average”

\[
\lim_{k_a \to \infty} Q_t(a) = Q^*(a)
\]

What is the behaviour with finite samples?
$\varepsilon$-Greedy Action Selection

• Greedy action selection:

$$a_t = a_t^* = \arg \max_a Q_t(a)$$

• $\varepsilon$-Greedy:

$$a_t = \begin{cases} 
  a_t^* \text{ with probability } 1 - \varepsilon \\
  \text{random action with probability } \varepsilon 
\end{cases}$$

... the simplest way to balance exploration and exploitation
A simple bandit algorithm

Initialize, for $a = 1$ to $k$:

$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$

Repeat forever:

$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1 - \varepsilon \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$ (breaking ties randomly)
$R \leftarrow \text{bandit}(A)$
$N(A) \leftarrow N(A) + 1$
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)} [R - Q(A)]$
Worked Example: 10-Armed Testbed

- $n = 10$ possible actions
- Each $Q^*(a)$ is chosen randomly from a normal distrib.: $N(0,1)$
- Each $r_t$ is also normal: $N(Q^*(a_t),1)$
- 1000 plays, repeat the whole thing 2000 times and average the results
10-Armed Testbed Rewards

Reward distribution

Run for 1000 steps
Repeat the whole thing 2000 times with different bandit tasks
ε-Greedy Methods on the 10-Armed Testbed

- Graph showing average reward over plays for different values of ε.
- Graph showing percentage of optimal action over plays for different values of ε.

06/02/18
Reasoning further: Interval Estimation

• Attribute to each arm an “optimistic initial estimate” within a certain confidence interval
• Greedily choose arm with highest optimistic mean (upper bound on confidence interval)

• Infrequently observed arm will have over-valued reward mean, leading to exploration
• Frequent usage pushes optimistic estimate to true values
Interval Estimation (IE) Procedure

• Associate to each arm 100(1-\(\alpha\))% reward mean upper band

• Assume, e.g., rewards are normally distributed

• Arm is observed \(n\) times to yield empirical mean & std. dev.

• \(\alpha\)-upper bound \((u_\alpha)\) can be written in terms of mean \(\mu\) and standard deviation \(\sigma\) as:

\[
u_\alpha = \hat{\mu} + \frac{\hat{\sigma}}{\sqrt{n}} c^{-1}(1 - \alpha)\]

\[c(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} \exp\left( -\frac{x^2}{2} \right) dx\]

\(c\) is the Cum. Distribution Function

Visualize on the board ...
How to Evaluate an Online Alg.: Regret

- After you have played for \( T \) rounds, you experience a regret:
  \[
  \rho = T \mu^* - \sum_{t=1}^{T} \hat{r}_t = T \mu^* - \sum_{t=1}^{T} E[r_i(t)]
  \]
  \[
  \mu^* = \max_k \mu_k
  \]

- If the average regret per round goes to zero with probability 1, asymptotically, we say the strategy has no-regret property.
  ~ guaranteed to converge to an optimal strategy

- \( \varepsilon \)-greedy is sub-optimal (so has some regret). Why?

- If \( \alpha \) is carefully controlled, IE could be zero-regret strategy
Moving Back to the MDP Model

• We have actions \((a_t)\) as well as states \((s_t)\)
• System dynamics are stochastic – represented by a probability distribution for transitions between states
• Problem is defined as maximization of expected rewards

\[
\text{State Transition Dynamics:} \\
P^a_{s's} = \Pr\{s_{t+1} = s'|s_t = s, a_t = a\} \\
\text{Expected Rewards:} \\
R^a_{ss'} = E\{r_{t+1}|s_t = s, a_t = a, s_{t+1} = s'\} \\
\text{Note that:} \\
R^a_s = \sum_{s'} P^a_{ss'} R^a_{ss'}
\]

– Recall that \(E(X) = \sum x_i p(x_i)\) 
  for finite-state systems

06/02/18
What is the criterion for optimization (i.e., learning)?

\[ R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \]

where \( \gamma, 0 \leq \gamma \leq 1 \), is the **discount rate**.

What would be the effect of changing \( \gamma \)?
Expectation of this Criterion: Value Functions

- Value functions are used to determine how good it is for the agent to be in a given state (sometimes also to perform an action at s)
  - Expectation of the criterion in prev. slide (similar to “cost-to-go”)
- This is defined w.r.t. a specific policy, i.e., action distribution \( \pi(s, a) \)

State value function:

\[
V^\pi(s) = \mathbb{E}_\pi \{ R_t \mid s_t = s \} = \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right\}
\]
Value Functions

Note that there are multiple sources of (probabilistic) uncertainty:

- In state $s$, one is allowed to select different actions $a$
- The system may transition to different states $s'$ from $s$
- Depending on the above, return (defined in terms of reward) is a random variable – which we seek to maximize in expectation

$$V^\pi(s) = E_\pi \{ R_t | s_t = s \}$$

$$= E_\pi \{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \}$$

$$= E_\pi \{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s \}$$
Recursive Form of $V$ – Bellman Equation

$$V^\pi(s) = E_\pi \{ r_{t+1} + \gamma \sum_{k=0}^\infty \gamma^k r_{t+k+2} | s_t = s \}$$

We rewrite as follows:

- First term: $\sum_a \pi(s, a) \sum_{s'} P_{ss'}^a R_{ss'}^a$
- Second term: $\sum_a \pi(s, a) \sum_{s'} P_{ss'}^a \gamma E_\pi \{ \sum_{k=0}^\infty \gamma^k r_{t+k+2} | s_{t+1} = s' \}$

$$V^\pi(s) = \sum_a \pi(s, a) \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma E_\pi \{ \sum_{k=0}^\infty \gamma^k r_{t+k+2} | s_{t+1} = s' \}]$$

$$V^\pi(s) = \sum_a \pi(s, a) \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^\pi(s')]$$
Recursive Expression for an *Optimal* Value Function, $V^*$

A key result for Dynamic Programming with Markov Decision Process Models is the following recursive expression that holds true for each state:

$$ V^*(s) = \max_a \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^*(s')] $$

This is a characterization of the value function, i.e., when you find an optimal value function then each state and its neighbours will satisfy this recursive relationship.
Given a Policy, what is Value Function, $V$?

Solve iteratively, with a sequence of value functions, $V_0, V_1, V_2, \ldots : S \rightarrow \mathbb{R}$

$$V_{k+1}(s) = \sum_a \pi(s, a) \sum_{s'} \mathcal{P}_{ss'}^{a} \left[ \mathcal{R}_{ss'}^{a} + \gamma V_k(s') \right] \forall s \in S$$

$V_k = V^\pi$ is a fixed-point for these updates, as $k \rightarrow \infty$

*Iterative* policy evaluation.
Grid-World Example

Four possible actions: $A = \{\text{up, down, right, left}\}$

- the actions change state deterministically (but, not allowed to go off grid)

Encoded in transition probabilities, e.g., $P_{5,6}^{\text{right}} = 1$, $P_{5,10}^{\text{right}} = 0$, $P_{7,7}^{\text{right}} = 1$

Undiscounted, episodic task with reward $-1$ everywhere except goal states.
Iterative Policy Evaluation in Grid World

Note: The value function can be searched greedily to find long-term optimal actions
Now, Given a Value Function, Can We Improve a Policy?

• Yes, compute the following:

\[ \pi'(s) = \arg \max_a E\{r_{t+1} + \gamma V^{\pi}(s_{t+1}|s_t = s, a_t = a) \} \]

\[ \pi'(s) = \arg \max_a \sum_{s'} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V^{\pi}(s')] \]

• ... and this can be iterated upon!

\[ \pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} V^{\pi_2} \ldots \xrightarrow{I} \pi^* \xrightarrow{E} V^* \]
Example: Jack’s Car Rental

• £10 for each car rented (must be available when request received)
• Two locations, maximum of 20 cars at each
• Cars returned and requested randomly
  – Poisson distribution, \( n \) returns/requests with probability \( \frac{\lambda^n}{n!} e^{-\lambda} \)
  – Location 1: Average requests = 3, Average returns = 3
  – Location 2: Average requests = 4, Average Returns = 2
• Can move up to 5 cars between locations overnight (costs £2 each)

Problem setup:
• States, actions, rewards?
• Transition probabilities?
Solution: Jack’s Car Rental

Numbers indicate action: #cars to move

Value
Points to Ponder: Jack’s Car Rental

• Suppose first car moved is free but all others transfers cost £2
  – From Location 1 to Location 2 (not other direction!)
  – Because an employee would anyway go in that direction, by bus

• Suppose only 10 cars can be parked for free at each location
  – More than 10 incur fixed cost £4 for using an extra parking lot

... typical examples of ‘real-world nonlinearities’
Value Iteration

Each step in Policy Iteration needs Policy Evaluation (up to convergence) - can we avoid this computational overhead?

Just update the values for one iteration and then improve the policy.

Update rule:

\[ V = \max_a \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V(s')] \]

... use Bellman equation as update rule

So we sweep through the state space once (and don't wait for \( V \) to stop changing, as in policy evaluation), then improve the policy, then repeat.

This update combines the one-iteration update of \( V \) plus the policy improvement (greedification wrt \( V \)) in one step.
Acknowledgements

The main source for this section is Sutton+Barto, Reinforcement Learning:

• Part 1 [MAB]: Ch 2 (sections 2.1-2.2)
• Part 2 [Bellman/Value]: Ch 3,4 (sections 3.7-3.8, 4.1-4.4)

The interval estimation procedure is from L. Pack Kaelbling, *Learning in Embedded Systems*, MIT Press (Ch 4)

[https://books.google.co.uk/books?id=WiN53ZYd0kgC&dq=leslie+kaelbling+interval+estimation&source=gbs_navlinks_s](https://books.google.co.uk/books?id=WiN53ZYd0kgC&dq=leslie+kaelbling+interval+estimation&source=gbs_navlinks_s)