COMP6247: Reinforcement and Online Learning

Mahesan Niranjan

School of Electronics and Computer Science
University of Southampton

Part II: Reinforcement Learning

Class One: Introduction; Exploration & Exploitation

Spring Semester 2020/21

Where are we?

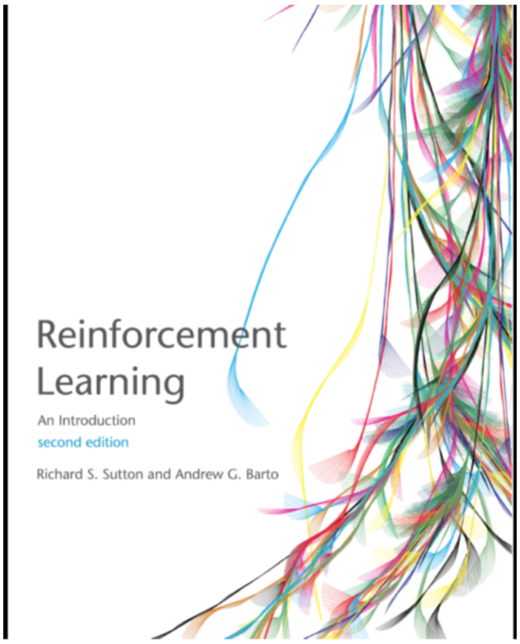
- Part I: Online Learning
  - Recursive Least Squares (RLS) Algorithm
  - State-space models and Kalman filtering
  - Bayesian Inference, esp. Importance Sampling
  - Particle Filtering
- Part II: Reinforcement Learning
  - Sutton and Barto: Reinforcement Learning
    `http://incompleteideas.net/book/RLbook2020.pdf`
  - Some recent papers

Assessment

- 50% Assignments on Online Learning
  - Task I 10%: Graded
  - Task II 10%: Graded
  - Task III 30%: Due 19 April 2021
- 50% Assignment on Reinforcement Learning
  - Issue: Week 9
  - Due: Week 12

Reference Text for Reinforcement Learning

Free PDF: `http://incompleteideas.net/book/RLbook2020.pdf`



Several Recent Papers

- Arulkumaran, K. *et al.* Deep Reinforcement Learning, *IEEE Signal Processing Magazine* (2017)
- Silver, D. *et al.* Mastering the game of Go, *Nature* (2016)
- Mnih, V. *et al.* Human-level control... [Atari games], *Nature* (2015)
- Tesauro, G. Temporal Difference [Backgammon], Communications ACM (1995)
- Yuan, X. *et al.* Reinforcement Learning for Elevator Control, *IFAC Conference* (2008)
- Mnih, V. *et al.* Recurrent Models of Visual Attention, *NeurIPS Conference* (2014)
- Li, K. and Malik, J. Learning to Optimize Neural Networks, *ICLR Conference* (2018)
- Popova, M. *et al.* Deep reinforcement learning for *de novo* drug design, *Science Advances* (2018)
- Mao, H. *et al.* Resource management with deep reinforcement learning, 15^{th} *ACM Workshop on Hot Topics in Networks* (2016).

Reinforcement Learning in Context

What we have seen so far...

- **Static**, supervised, unsupervised problems

$$\{\mathbf{x}_n, y_n\}_{n=1}^N$$
$$\{\mathbf{x}_n\}_{n=1}^N$$

- **Dynamical** Systems
  - Time series, control problems

$$\begin{aligned}\theta(n+1) &= \theta_n + \mathbf{w}_n \\ y_n &= f(\mathbf{x}_n \theta_n) + v_n\end{aligned}$$

  - Linear system, Gaussian noise: optimal estimation of θ from data via Kalman filter.
  - Nonlinear / Non-Gaussian: approximate (probabilistic) estimation via particle filtering

Learning by Interacting

Planning problems

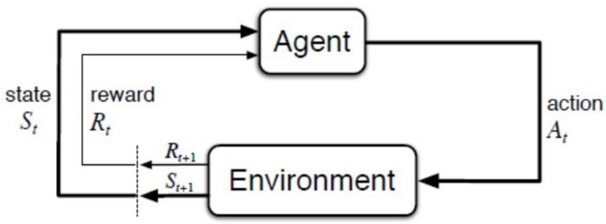


Figure 3.1: The agent–environment interaction in a Markov decision process.

Important Parameters / Concepts:

- State
- Action
- (Immediate / Short-term) Reward
- Policy
- (Long-term) Return

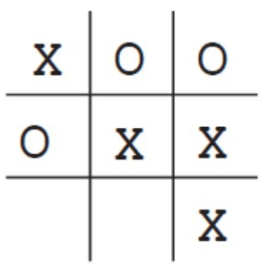
Introduction

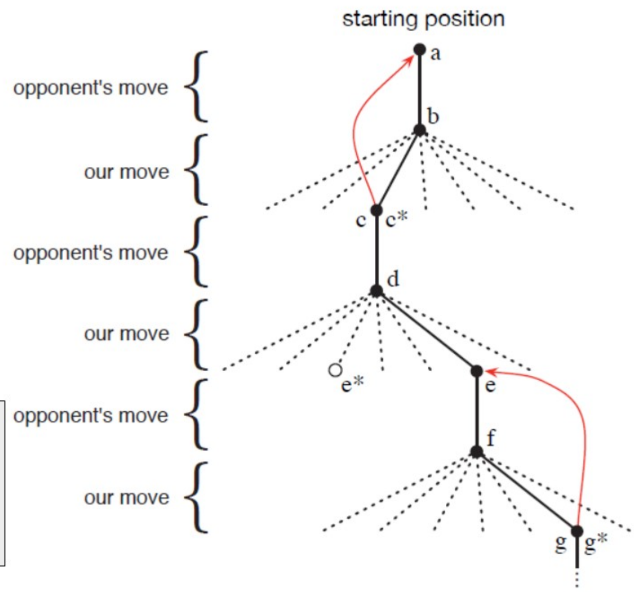
Chapter 1: Sutton and Barto, Reinforcement Learning

- Learning problems much closer to biological learning.
  - Consider ImageNet Challenge
  - Infants (or even adults) do not learn by collecting 1M images in 1000 categories and learning to separate them!
  - But will do not make a theory of biological learning and then mimic it on computer
  - Our approach is still fundamentally *computational*
- Algorithms that map situations (states) to actions
- Criterion of optimality and action choice (policy) should be optimal
- Computational difficulties and ways to circumvent them
- Closely related to optimal control, but with unknown (Markov) processes, learning about underlying systems and how to control them.
- *Exploration* and *Exploitation*

Example: Tic-tac-toe

Section 1.5: Sutton and Barto





Optimization (evolutionary algorithm)

versus

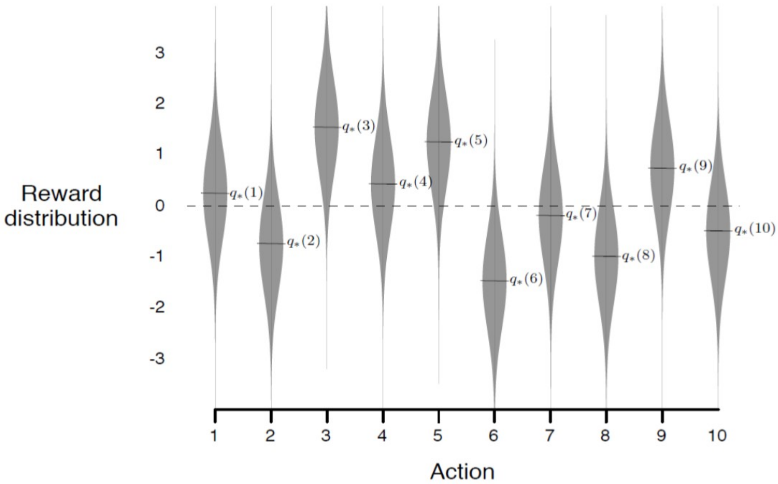
Learning to improve during play

- $V(S_t) \leftarrow V(S_t) + \alpha \left[V(S_{t+1}) - V(S_t) \right]$
- Note structure of algorithm
- Compulsory reading: Section 1.5; Optional reading; Section 1.7.

Bandit Problems

Brief Introduction: Exploration - Exploitation

- Chapter 2: Sutton and Barto



$k-$armed Bandit Problem

- Reinforcement Learning uses training information to *evaluates* actions
  - Supervised learning has targets specified $\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N$
- Faced repeatedly with a choice among k different options, or actions;
- After each choice you receive a numerical reward;
- Reward chosen from a stationary probability distribution for the chosen action;
- Objective: maximize the expected total reward over a certain number of *time steps*
- Slot machine analogy
  - Action selection = play of one of the levers
  - Reward = payoffs at each play

Exploration and Exploitation

- Each of the k actions has an expected or mean reward (call this *value* of the action)
- time t, action selected A_t reward R_t
- Value of an action $q_*(a) \; = \; E\left[R_t \mid A_t = a\right]$
- If we knew the value of actions, problem is solved
- Estimated value of action: $Q_t(a)$
- Working with estimates, at any time step there is at least one action whose value is greatest
- Selecting this action is *greedy* approach, *exploits* the present estimate
- But we may also want to *explore*, because the current estimate $Q_t(a)$ may not be close to the best $q_*(a)$. estimated value is greatest
- Strike a balance between exploring and exploiting

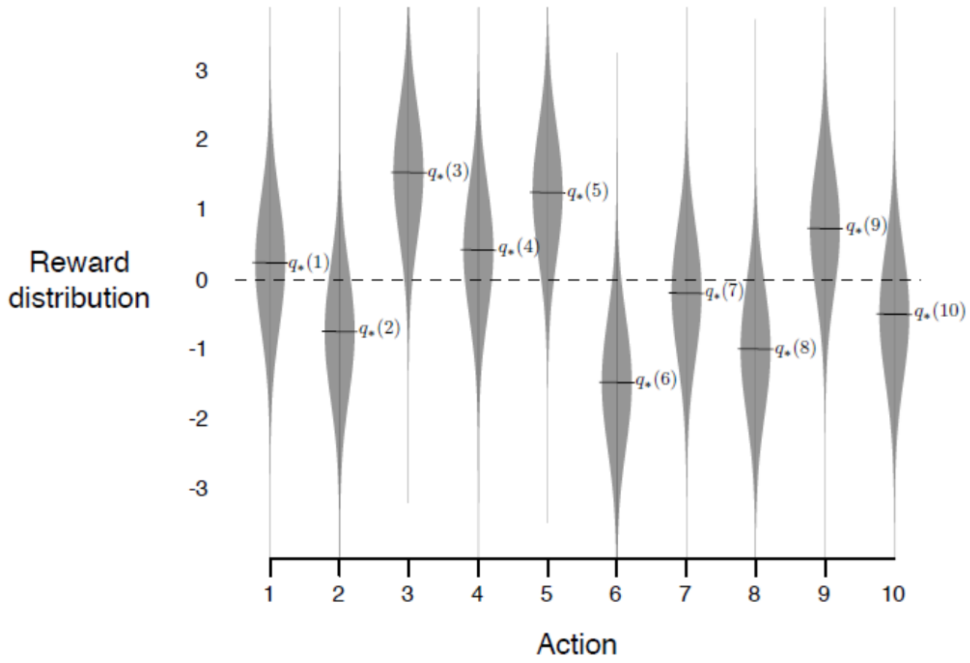
Action-value Methods

- True value of an action is the mean (expected) reward when that action is selected.
- A sample-average method:

$$
\begin{aligned}
Q_t(a) \quad &= \quad \frac{\text{Sum of rewards when a was taken prior to t}}{\text{number of times a taken prior to t}} \\
&= \quad \frac{\sum_{i=1}^{t-1} R_i \, \boldsymbol{I}_{A_i=a}}{\sum_{i=1}^{t-1} \boldsymbol{I}_{A_i=a}}
\end{aligned}
$$

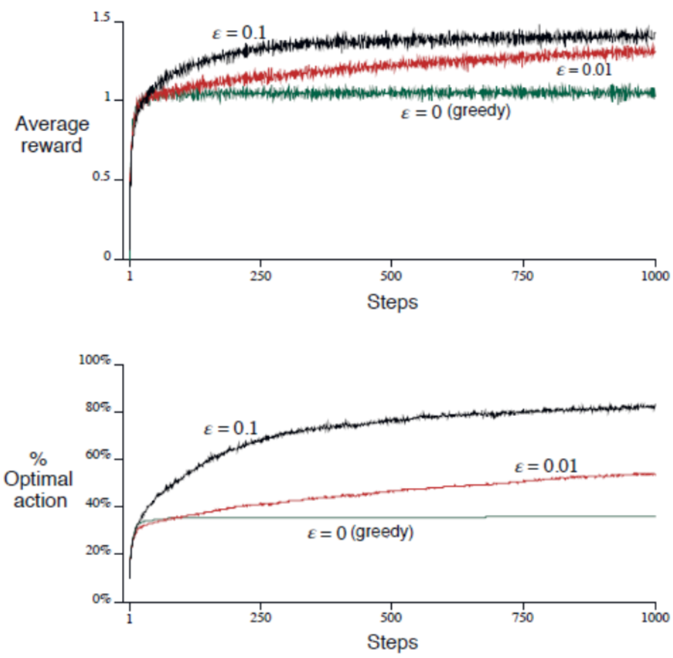
- Greedy selection: $A_t \; = \; \underset{a}{\text{argmin}} \, Q_t(a)$
- To explore, select a random ation with probability ϵ

Example: 10-Armed Bandit



Example: 10-Armed Bandit

Simulation Results



Incremental Implementation

- Online learning is important
- $Q_n = (R_1 + R_2 + ... + R_{n-1}) / (n-1)$
- Can be computed incrementally (recursively)

$$\begin{aligned} Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\ &= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\ &= \frac{1}{n}(R_n + (n-1)Q_n) \\ &= \frac{1}{n}(R_n + nQ_n - Q_n) \\ &= Q_n + \frac{1}{n}(R_n - Q_n) \end{aligned}$$

A Simple Bandit Algorithm

A simple bandit algorithm

Initialize, for $a = 1$ to k:
$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$

Repeat forever:
$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$
$R \leftarrow bandit(A)$
$N(A) \leftarrow N(A) + 1$
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\big[R - Q(A)\big]$

NewEstimate = OldEstimate + StepSize * (Target - OldEstimate)

Tracking a Nonstationary Signal

Section 2.5: Sutton and Barto

$$\begin{aligned} Q_{n+1} &= Q_n + \alpha\,[R_n - Q_n] \\ &= \alpha\,R_n + (1-\alpha)\,Q_n \\ &= \alpha\,R_n + (1-\alpha)\,[\alpha\,R_{n-1} + (1-\alpha)Q_{n-1}] \\ &= \ldots \\ &= (1-\alpha)^n\,Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i}\,R_i \end{aligned}$$

- Note: $(1-\alpha)^n + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} = 1$
- Hence Q_{n+1} is a weighted sum over Q_1 and the rewards received R_i.
- Weight given to R_i depends on how long ago, $n-i$, R_i was observed as reward.
- Hence *exponential recency-weighted average*
- Sometimes step size is varied: $\alpha_n(a)$. $\alpha_n(a) = \frac{1}{n}$ is taking sample average.
- To guarantee convergence: $\sum_{n=1}^{\infty} \alpha_n(a) = \infty$ and $\sum_{n=1}^{\infty} \alpha_n^2(a) < \infty$
- A result from *Stochastic Approximation* literature.
- Conditions met for sample average, but not for constant step size.

Upper Confidence Bound Action Selection

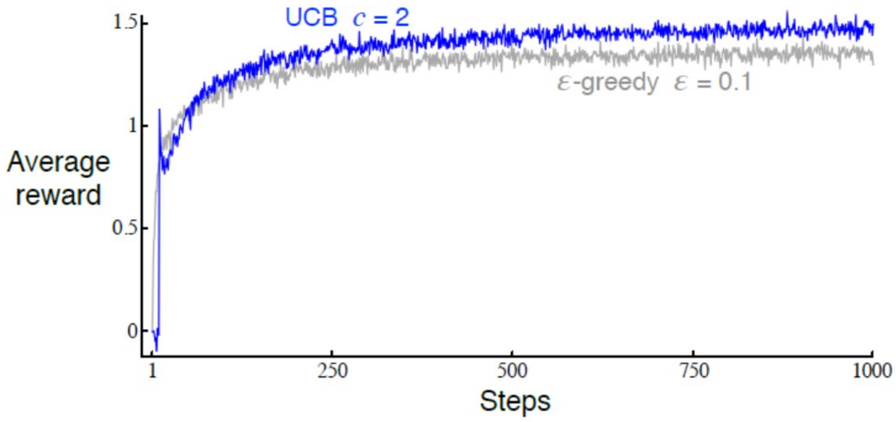
- Upper Confidence Bound selection:

$$A_t = \underset{a}{\operatorname{argmax}} \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

- $\ln t$: natural logarithm
- $N_t(a)$: Number of times action a was selected prior to time t.
- $c > 0$ controls amount of exploration
- The term in square-root is measure of the uncertainty (variance) in the estimate of a's value.
- The quantity being max'ed over is thus a sort of upper bound on the possible true value of action a. c level of confidence.

Upper Confidence Bound Versus $\epsilon-$ Greedy

Figure 2.4: Sutton and Barto



Gradient Bandit Algorithms

- A numerical preference for action a as $H_t(a)$
- Larger the preference, more frequently is that action taken
- Convert to a probability by softmax:

$$\pi_t(a) = \Pr(A_t = a) = \frac{\exp H_t(a)}{\sum_{b=1}^{k} \exp H_t(b)}$$

- Learning algorithm: Select action A_t and receive reward R_t

$$\begin{aligned} H_{t+1}(A_t) &= H_t(A_t) + \alpha (R_t - \overline{R_t}) (1 - \pi_t(A_t)) \\ H_{t+1}(a) &= H_t(a) - \alpha (R_t - \overline{R_t}) \pi_t(a), \quad \forall a \neq A_t \end{aligned}$$

- $\overline{R_t}$ is average of all rewards received so far (can be computed incrementally).
- If current reward is higher than average so far (baseline) increase the preference for selected action;
- Formal analysis, relating to stochastic approximation: ** skip **

Summary

- Bandit algorithms give a formal setting to study exploration and exploitation
- We touched on the basic ideas:
  - Epsilon Greedy
  - Incorporating confidence / uncertainty (UCB)
  - Formulating preferences (plus softmax)

Next:

- Markov Decision Processes
- Dynamic Programming