RL 4: $\mathcal{Q}$-Learning: Examples and Theory

(Markovian decision processes I)

Michael Herrmann

University of Edinburgh, School of Informatics

24/01/2014

Last time: $\mathcal{Q}$-Learning (Points to remember)

- Brute force approach: For each policy, $\pi : \mathcal{S} \to \mathcal{A}$, sample returns, and choose the policy with the largest expected return
- Or: Allow samples from one policy to influence the estimates made for another

$$\mathcal{Q}_{t+1}(s_t, a_t) = \mathcal{Q}_t(s_t, a_t) + \eta \left(r(s_t, a_t) + \gamma V_t(s_{t+1}) - \mathcal{Q}_t(s_t, a_t) \right)$$

(C. J. C. H. Watkins, 1989)

- $V_t(s) = \max_a \mathcal{Q}_t(s, a)$, $\underbrace{a_t = \arg\max_a \mathcal{Q}_t(s, a)}_{\text{greedy policy}}$, γ discount factor
- *Off-policy* algorithm (learning rule works well with exploration)
- The value of a state is the total discounted future reward expected when choosing the action presently considered best now and continue with the policy presently considered optimal.
- $V(s_t) = r(s_k, a^*(s_k)) + \gamma V(s_{t+1})$ (ideally)
- $r_{\max} / (1 - \gamma) \geq \mathcal{Q}(s, a)$ (except for initialisation effects)
- $\mathcal{Q}$-learning generates trees (or forests, for multiple goals) with goal(s) as root(s)

Preliminary discussion of the relevant time scales

1. Behavioural time scale $1/(1-\gamma)$ (discount factor)
2. Sampling in the estimation of the $\mathcal{Q}$-function η (learning rate)
3. Exploration ε (e.g. for ε-greedy strategy)

$$1-\gamma \gg \eta \gg \varepsilon$$

Adaptation of time scales: Initially $1-\gamma \approx \eta \approx \varepsilon$ is possible, then decrease both ε and η, but ε faster than η to reach the separation of time scales asymptotically.

Practically, fix maximal number of trials $M < \infty$ and set $\eta \sim (1-m/M)^{\alpha}$ and $\varepsilon \sim (1-m/M)^{\beta}$ with $\alpha < \beta$, $m = 1, \ldots, M$.
Not theoretically justified.

γ may be moved a bit towards 1, i.e. explore first short time scales, later longer ones (assuming there is some reward in both cases)

Examples

1. Define
   - states, actions, rewards, γ, η, ε, initialisation, steps, ...
2. Organise experiments
   - episodes
   - repetitions
3. Analysis
   - significance, robustness, generalisability
   - potential improvements [goto 1]

Example 0: MAB as a Special Case

$\mathcal{Q}$-learning for the N-armed bandit

- States: $s \in \{C\}$ (just one state, namely the Casino)
- Actions: $a \in \{1, \ldots, N\}$
- state transitions $C \to C$, $\forall a$
- Reward: $r = r_a$ with $r_a \sim \mathcal{N}\left(\mu_a, \sigma_a^2\right)$
- Initialisation: $\mathcal{Q}_0\left(a, s\right) = 5 \times \max_a \left\{\mu_a + \sigma_a\right\}$ (optimistic)

$$\begin{aligned}
\mathcal{Q}_{t+1}\left(s_t, a_t\right) &= \mathcal{Q}_t\left(s_t, a_t\right) + \eta\left(r\left(s_t, a_t\right) + \gamma V_t\left(s_{t+1}\right) - \mathcal{Q}_t\left(s_t, a_t\right)\right) \\
&= \mathcal{Q}_t\left(a_t\right) + \eta\left(r\left(a_t\right) + \gamma V_t - \mathcal{Q}_t\left(a_t\right)\right) \\
\tilde{\mathcal{Q}}_{t+1}\left(a_t\right) &= \tilde{\mathcal{Q}}_t\left(a_t\right) + \eta\left(r\left(a_t\right) - \tilde{\mathcal{Q}}_t\left(a_t\right)\right)
\end{aligned}$$

- $a_t = \arg\max_a \mathcal{Q}_{t+1}\left(a\right) = \arg\max_a \tilde{\mathcal{Q}}_{t+1}\left(a\right)$

Example 1: A Simple Maze

- States: $s \in \{4 \times 4 \text{ squares}\}$ without obstacles
- Actions: $a \in \{E, W, N, S\}$
- Reward: $r = 0$ if $s \equiv G$ (two of the corners),
  $r = -1$ for each step taken
- discount factor: $\gamma = 1$ (no discount)

G			
			G

- Initialisation: $\mathcal{Q}(a, s) \sim \mathcal{N}(0, 0.01)$
- Reset to random position after reaching the goal

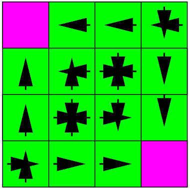
Navigation in a grid world

0	−14	−20	−22
−14	−18	−22	−20
−20	−22	−18	−14
−22	−20	−14	0

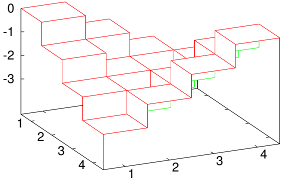
average path length
for random exploration

0	−1	−2	−3
−1	−2	−3	−2
−2	−3	−2	−1
−3	−2	−1	0

minimal path length
using optimal policy



preferred actions
(may stay undecided)



non-discounted
value function
over state
space

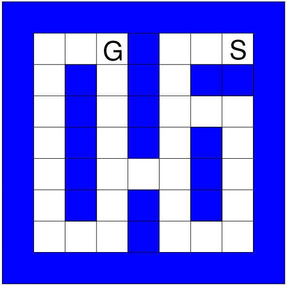
Adapted from
Mance E. Harmon:
Reinforcement Learning:
A Tutorial (1996)

Navigation in a grid world: Discussion

- Random exploration (i.e. ε-greedy with $\varepsilon = 1$) provides (in this example; in general it's better to slowly decrease ε) the information about optimal policy.
- If $\mathcal{Q}(s, a)$ is randomly initialised (and ε is small) then the agent may easily get stuck.
- The values at the goals are due to a special treatment of these states. The reward is immediate reward plus value of next state:
  - if the agent is reset to a random position for the next episode then the next state may have a very low value
  - if the agent stays at G then another step is taken which has a cost of -1, but is this not counted.
  - Practically, we are not using $\mathcal{Q}(G, a)$, only $V(G)$ which is defined to be zero an used to update $\mathcal{Q}(s, a)$ of the previous state that led the agent to the goal.
  - If we did update $\mathcal{Q}(G, a)$ we should (make sure that we) obtain $\mathcal{Q}(G, a) = 0$ for a towards the wall, and $\mathcal{Q}(G, a) = -2$ for the a's back to the maze.

Example 2: Another Maze

- States: $s \in \{7 \times 7 \text{ squares}\}$ with obstacles (see previous lecture)
- Actions: $a \in \{E, W, N, S\}$
- Reward: $r = 1$ if $s \equiv G$, $r = 0$ otherwise
- Initialisation: $\mathcal{Q}_0(a, s) \sim \mathcal{N}(0, 0.01)$

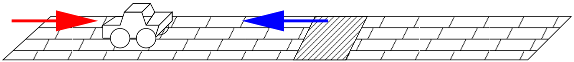


[Homework]

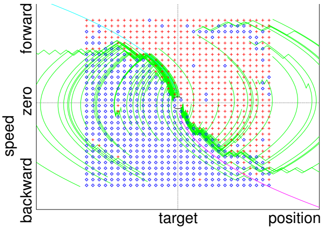
Example 3: Car Positioning
(A simplified version of the Mountain Car problem)

- States: $s \in \{40 \times 40 \text{ squares}\}$
- Actions: $a \in \{\text{accelerate forward, accelerate backward}\}$
- Reward: $r = 1$ if stopping in a small region near the goal, $r = 0$ otherwise
- Initialisation: $\mathcal{Q}_0(a, s) \sim \mathcal{N}(0, 0.01)$
- Discount factor: $\gamma = 0.95$
- Exploration: initially $\varepsilon = 0.25$, decaying over 100,000,000 time steps
- Learning rate $\eta = 0.1$

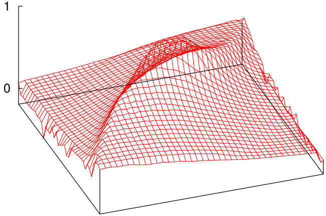
Example 4: Cart positioning by Q-learning



Accelerate cart such that it stops at a given position in min. time
Only one level of acceleration, action is to choose the sign
Size of goal region determines minimal state space resolution
Problems: relevant part of state space, slow convergence



preferred actions

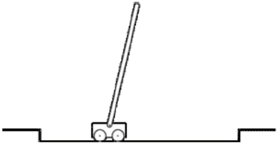


value function

Example 4: Discussion

- Extremely long learning time
- How many time step does is take until the agent reaches a noew state?
- Neighbouring states are doing largely the same
- The boundary between the region is not very well resolved even for a fine democratisation
- Try (using information available during the learning process)
  - success stories: update not only previous time steps, but also everything that lead to the final success
  - adaptive partitioning of the state space: In more homogeneous regions use few states, whereas near critical boundaries more state are needed (e.g. based on a weighted k-means algorithm)
  - use options: only update once new information becomes available

Example 5: Inverted pendulum or "cart-pole"



Avoid **failure:** the pole falling beyond a critical angle or the cart hitting end of track.

As an **episodic task** where episode ends upon failure:

$$\text{reward} = +1 \text{ for each step before failure}$$
$$\Rightarrow \quad \text{return} = \text{number of steps before failure}$$

As a **continuing task** with discounted return:

$$\text{reward} = -1 \text{ upon failure; } 0 \text{ otherwise}$$
$$\Rightarrow \quad \text{return} = -\gamma^k, \text{ for } k \text{ steps before failure}$$

In either case, return is maximized by avoiding failure for as long as possible.

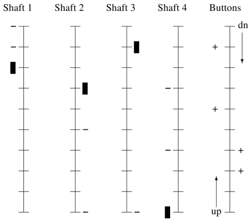
Example 5: Cart-Pole Problem

- States: 4D state space, few states per dimension
- Actions: $a \in \{\text{Accelerate,Brake}\}$
- Reward: $r = 1$ for upright pendulum, $r = -1$ upon failure, $r = 0$ otherwise,
- Initialisation: $\mathcal{Q}\,(a, s) \sim \mathcal{N}\,(0, 0.01)$
- Discount factor: $\gamma = 0.995$
- Exploration: initially $\varepsilon = 0.1$, decaying over 100,000,000 time steps
- Learning rate $\eta = 0.1$

[more later]

Examples from literature

- Elevator control (Barto and Crites, 1996)
- Learning in games:
  - Backgammon (Tesauro, 1994)
  - Go (Silver et al. 2007)



- Learning in robotics
  - Controlling quadrupeds (Kohl and Stone, 2004)
  - Humanoids (Peters et al. 2003)
  - Helicopters (Abbeel et al. 2007)
  - Automotive control
- Finance:
  - Optimal pricing (Tsitsiklis and Van Roy, 1999; Yu and Bertsekas, 2007; Li et al., 2009)

More examples from the literature

- More CS applications
  - Packet routing (Boyan and Littman, 1994)
  - Channel allocation (Singh and Bertsekas, 1997)
  - Dialogue strategy selection (Walker, 2011)
- Operations research
  - targeted marketing (Abe et al. 2004)
  - maintenance problems (Gosavi, 2004)
  - job scheduling (Zhang and Dietterich, 1995)
  - pricing (Rusmevichientong et al. 2006
  - vehicle routing (Proper and Tadepalli, 2006)
  - inventory control (Chang et al., 2007)
  - fleet management (Simão et al., 2009)
- Modelling biological mechanisms

Summary on examples

- Advantages: Sampling, bootstrapping, on-line learning, little domain knowledge required, theory based
- Disadvantages:
  - Solutions usually non-generalisable
  - Finding a good solution is slow, does not scale well
- Problem representation is critical: States, actions, rewards, parameters, ...
- Work on real-world examples has led to better algorithms:
  - Disambiguate stochastic state information
  - Reduce complexity of state/action spaces
  - Increase efficiency

Syllabus

- Reinforcement Learning
- Multi-Armed Bandits
- $\mathcal{Q}$-learning
- Markov chains
- Markov Decision Processes
- Dynamic programming
- Monte-Carlo methods
- Back to RL
- POMDPs
- Continuous problems

Stochastic Processes

- A stochastic process is an indexed collection of random variables $\{X_t\}$
  - e.g. time series of weekly demands for a product
- Discrete case: At a particular time t, labelled by integers, system is found in exactly one of a finite number of mutually exclusive and exhaustive categories or states, labelled by integers, too
- Process could be *embedded*, i.e. time points correspond to occurrence of specific events (or time may be equi-spaced)
- Random variables may depend on others, e.g.,

$$X_{t+1} = \begin{cases} \max\{(3 - D_{t+1}), 0\} & \text{if } X_t < 0 \\ \max\{(X_t - D_{t+1}), 0\} & \text{if } X_t \geq 0 \end{cases}$$

or $\quad X_{t+1} = \sum_{k=0}^{K} \alpha_k X_{t-k} + \xi_t$ with $\xi_t \sim \mathcal{N}(\mu, \sigma^2)$

Markov Chains

The stochastic process is said to have a Markovian property if

$$P(X_{t+1}=j|X_t=i, X_{t-1}=k_{t-1}, \dots, X_1=k_1, X_0=k_0) = P(X_{t+1}=j|X_t=i)$$

for $t = 0, 1, \dots$ and every sequence i, j, $k_0, \dots, k_{t-1}$

Markovian probability means that the conditional probability of a future event given any past events and current state, is independent of past states and depends only on present

The conditional probabilities are transition probabilities,

$$P\left(X_{t+1} = j|X_t = i\right)$$

These are *stationary* if time invariant. The we can write

$$p_{ij} = P\left(X_{t+1} = j|X_t = i\right) = P\left(X_1 = j|X_0 = i\right)$$

Markov Chains

- A stochastic process is a finite-state Markov chain if it has
  - a finite number of states $s \in \mathcal{S}$
  - the Markovian property
  - stationary transition probabilities p_{ij} for all i, j
  - a set of initial probabilities $\pi_i^0 = P\{X_0 = i\}$ for all i
- n-step transition probabilities (looking forward in time)

$$p_{ij}^{(n)} = P(X_{t+n} = j | X_t = i) = P(X_n = j | X_0 = i)$$

- One can write a transition matrix

$$\mathsf{P}^{(n)} = \begin{pmatrix} p_{00}^{(n)} & \cdots & p_{0M}^{(n)} \\ \vdots & \ddots & \vdots \\ p_{M0}^{(n)} & \cdots & p_{MM}^{(n)} \end{pmatrix}$$

Andrey Markov

Markov Chains

- 2-step transition probabilities can be obtained from 1-step *transition probabilities*

$$p_{ij}^{(2)} = \sum_{k=1}^{M} p_{ik} p_{kj}, \ \forall i, j$$

- n-step transition probabilities can be obtained from 1-step *transition probabilities* recursively (Chapman-Kolmogorov)

$$p_{ij}^{(n)} = \sum_{k=1}^{M} p_{ik}^{(v)} p_{kj}^{(n-v)}, \ \forall i, j, n; \ 0 \leq v \leq n$$

- We can get this via the matrix too

$$P^{(n)} = \underbrace{P \cdots P}_{n \text{ times}} = P^n = PP^{n-1} = P^{n-1}P$$

Markov Chains: First Passage Times

- Number of transitions to go from i to j for the first time
  - First Passage Times are random variables $\Rightarrow$ mean FPT etc.
  - If $i = j$, this is the *recurrence time*
- n-step recursive relationship for first passage probability

$$
\begin{aligned}
f_{ij}^{(1)} &= p_{ij}^{(1)} = p_{ij} \\
f_{ij}^{(2)} &= p_{ij}^{(2)} - f_{ij}^{(1)} p_{jj} \\
&\vdots \\
f_{ij}^{(n)} &= p_{ij}^{(n)} - f_{ij}^{(1)} p_{jj}^{(n-1)} - f_{ij}^{(2)} p_{jj}^{(n-2)} - \cdots - f_{ij}^{(n-1)} p_{jj}
\end{aligned}
$$

- For fixed i and j, these $f_{ij}^{(n)}$ are non-negative numbers so that $\sum_{n=1}^{\infty} f_{ij}^{(n)} \leq 1$
- If $\sum_{n=1}^{\infty} f_{ii}^{(n)} = 1$ that state is a *recurrent* state,
- It is *absorbing* if $f_{ii}^{(1)} = 1$

Markov Chains: Classification of States

- State j is *accessible* from i if $p_{ij}^{(n)} > 0$ (for some n)
  - What is the accessibility of states for the inventory example?
  - What does this mean in RL?
- If state j is accessible from i and vice versa, the two states are said to *communicate*
  - What is the status of states in inventory example?
- As a result of communication, one may partition the general Markov chain into states in disjoint classes
  - MC is *irreducible* if there is only one class

Markov Chains: Classification of States

- Many Markov chains in practise consist entirely of states that communicate with each other; hence are irreducible with only positive recurrent states
- Positive recurrence: State is recurrent and has a finite expected return time.
- If the MC can only visit the state at integer multiples of t, we call it *periodic*
- Positive recurrent states that are aperiodic are called *ergodic* states
  - What can you say about how ergodic states will evolve?

Markov Chains: Long-Run Properties

Inventory example: Interestingly, probability of being in state j (after, e.g., 8 weeks) appears independent of *initial* level π^0 of inventory.

For an irreducible ergodic Markov chain, one has limiting probability

$$\lim_{n\to\infty} p_{ij}^{(n)} = \pi_j^*$$

i.e. the limit for each element p_{ij} does not depend on i.

$$\pi_j^* = \sum_{i=1}^{M} \pi_i^* p_{ij} \ \forall j = 1, \ldots, M$$

$\pi^* = (\pi_1^*, \ldots, \pi_M^*)$ is an eigenvector of the matrix $P = (p_{ij})$.

Perron-Frobenius theorem: Matrices with positive entries have a unique largest eigenvalue. For a probability matrix this EV is 1.

Reciprocal of π_j^* gives the recurrence time m_{jj}

Markov Chains: Expected Average Cost

Sometimes aperiodic chain is a strong assumption. If we relax it, the limiting probability needs a slightly different definition:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} p_{ij}^{(n)} = \pi_j^*$$

Suppose you incur a (time-independent) cost $C(X_t)$, using above you can derive the long-run expected average over unit time as

$$\lim_{n\to\infty} \left\{ E\left[\frac{1}{n} \sum_{k=1}^{n} C\left(x_t\right) \right] \right\} = \sum_{j=1}^{M} C_j \pi_j^*$$

Can be more elaborate in general, depending on cost function

Conclusion and Outlook

- Markov Chains will be used as model of the state dynamics in a RL problem
- Not all state dynamices are Markovian
- A fixed policy transforms a Markov chain in a Markov chain with a (generally) different transition matrix
- On-policy reinforcement learning algorithms are often based MDPs in a strict sense ($\mathcal{Q}$-learning is off-policy and therefore not a very good example)

Many slides are adapted from web resources associated with Sutton and Barto's Reinforcement Learning book ... before being used by Dr. Subramanian Ramamoorthy in this course in the last three years.