RL 4: $\mathcal{Q}$-Learning II

Michael Herrmann

University of Edinburgh, School of Informatics

23/01/2015

Last time: $\mathcal{Q}$-Learning I (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)

- γ discount factor, η learning rate
  $V_t(s) = \max_a \mathcal{Q}_t(s, a)$, $a_t = \arg\max_a \mathcal{Q}_t(s, a)$ (greedy)
- *Off-policy* algorithm (learning rule works well with exploration)
  We need also an exploration rate: ε for ε-greedy exploration
- The value of a state is the total discounted future reward expected when choosing the action presently considered best and continue with the policy presently considered optimal.
- $V(s_t) = r(s_t, a^*(s_t)) + \gamma V(s_{t+1})$ (ideally, i.e. as learning goal)

Today

- How does $\mathcal{Q}$-learning work?
- How to adapt the algorithm to a practical problem?
- How to set up an RL experiment?

plus: Step-by-step example and a few videos

How does RL ($\mathcal{Q}$-learning) work?

Environment: You are in state 5. You have 4 possible actions.

Agent: I'll take action 2.

Environment: You received a reward of 7 units. You are now in state 3. You have 3 possible actions.

Agent: I'll take action 1.

Environment: You received a reward of -4 units. You are now in state 6. You have 2 possible actions.

Agent: I'll take action 2.

Environment: You received a reward of 8 units. You are now in state 3. You have 3 possible actions.

$\vdots \qquad \vdots$

$$
\begin{array}{lll}
\ldots & s_t = 5 & a_t = 2 \\
r_{t+1} = 7 & s_{t+1} = 3 & a_{t+1} = 1 \\
r_{t+2} = -4 & s_{t+2} = 6 & a_{t+2} = 2 \\
r_{t+3} = 8 & s_{t+3} = 3 & \ldots
\end{array}
$$

Behaviour of the agent after learning

- $\mathcal{Q}$-learning generates trees (or forests, for multiple goals) with goal(s) as root(s)
- After initialisation or reset the agent starts at a random position s (or at one of the starting states) and follows the route to the/a goal that starts with the best action

$$a^* = \arg\max_a \mathcal{Q}(s, a^*)$$

- Under certain conditions this is also the route that provides the maximal (discounted) reward
- If the agent is perturbed (i.e. performs a random action) it continues from the thus reached state towards the goal on a possibly different route. The $\mathcal{Q}$-value of the random action is updated based on the new route.

Assume that for action a_0 state remains unchanged $s_0 = s_1$ and that reward is deterministic $r(s_0, a_0) > 0$, then

$$\mathcal{Q}_t(s_0, a_0) = \mathcal{Q}_{t-1}(s_0, a_0) + \eta\left(r(s_0, a_0) + V(s_0) - \mathcal{Q}_{t-1}(s_0, a_0)\right)$$

Assume also greedy actions

$$a_0 = \arg\max_{a \in \mathcal{A}} \left(\mathcal{Q}(s_0, a)\right)$$

Because

$$V(s_0) = \max_{a \in \mathcal{A}} \left(\mathcal{Q}(s_0, a)\right)$$

then $\mathcal{Q}(s_0, a_0)$ will grow without bounds.

If, in a different setting, $r(a_t, s_t) = 0$ for many or few steps before finally $r > 0$ is achieved, $\mathcal{Q}$ would be the same.

$\Longrightarrow$ Value function should express discounted reward: $\gamma < 1$

Assume $s_t = s_{t+1}$, and deterministic reward $r(s_t, a_t) = r_{\max}$, then

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

Assume greedy action: $a_t = \arg\max_{a \in \mathcal{A}} \left(\mathcal{Q}(s_t, a)\right)$, i.e.

$$V(s_0) = \max_{a \in \mathcal{A}} \left(\mathcal{Q}(s_0, a)\right) = \mathcal{Q}(s_0, a_0)$$

convergence condition: $\eta\left(r_{\max} + \gamma V(s_0) - \mathcal{Q}_t(s_0, a_0)\right) = 0$

$$\frac{r_{\max}}{1-\gamma} = \mathcal{Q}_t(s_0, a_0)$$

finite for $\gamma < 1$ (remember that γ is typically close to 1)

$\Rightarrow r_{\max}/(1-\gamma) \geq \mathcal{Q}(s, a)$ (except for initialisation effects)

useful for optimistic initialisation (if $r_{\max}$ is known)

Preliminary discussion of the relevant time scales

1. Behavioural time horizon $1/(1-\gamma)$, γ discount factor,
2. Sampling in the estimation of the $\mathcal{Q}$-function η (learning rate): η small for stochastic problems, larger ($\eta \leq 1$) for deterministic problems
3. Exploration ε (ε-greedy strategy is usually not a bad choice)

- Order of time scales:

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

- Trials should include behaviourally relevant pieces of trajectories. It should be possible that the agent finds reward within the $1/(1-\gamma)$ time horizon.
- In stochastic problems several trial are needed before the value function is significantly changed.
- On an even slower time scale random actions are performed. The exploration rate should decay such that the definition of value ("continue with best policy") is violated only rarely.

Hints for parameters choices

- Should γ change during learning?
  - Yes, if "stepping stone"-reward is used, i.e. if part behaviours receive small rewards γ can be be smaller (e.g. 0.9 for a 10-step horizon). Later when the full behaviour is learned then γ may reach larger values (e.g. 0.99 for a 100-step horizon)
  - 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)
  - For episodic problems, $\gamma = 1$ can be reasonable
- Should η decrease during learning?
  - In simple deterministic problems constant large values (e.g. $\eta = 0.25$) are fine.
  - In stochastic problems and for convergence of the value function, η should decay slowly (Robbins-Monro conditions $\rightarrow$ later)

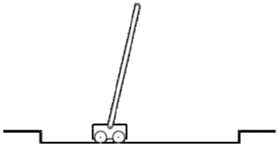
Hints for parameters choices

- Practically, fix maximal number of trials $M < \infty$ and set $\eta \sim (1 - m/M)^{\alpha}$ and $\varepsilon \sim (1 - m/M)^{\beta}$ with $\alpha < \beta$, $m = 1, \dots, M$. Not theoretically justified.
- Although exploratory actions will be rare, all relevant combinations of actions need to occur during learning!
- 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.
- Generally, parameter choices are problem-dependent and require some intuition and exploration

How to set up RL experiments?

1. Define
   - states, actions, rewards, γ, η, ε, initialisation, timeouts, ...
2. Organise experiments
   - episodes (continuous or reset of the agent)
   - repetitions (for statistical evaluation)
3. Analysis
   - convergence of policy and/or value functions
   - performance in terms of reward
   - Did the reward correctly specify the desired behaviour?
   - significance, robustness, generalisability
   - Any improvements possible? [goto 1]

Example: 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:

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

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

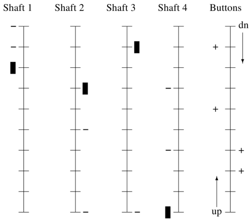
Example: Cart-Pole Problem

- States: 4D state space, few states per dimension
- Actions: $a \in \{\text{Accelerate}, \text{Brake}\}$
- 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
  - Informative initialisation, pretraining in simulation