RL 5: On-policy and off-policy algorithms

Michael Herrmann

University of Edinburgh, School of Informatics

27/01/2015

Overview

Off-policy algorithms

- $\mathcal{Q}$-learning (last time)
- R-learning (a variant of $\mathcal{Q}$-learning)

On-policy algorithms

- SARSA
- TD(λ)
- Actor-critic methods

R-learning

Similar to $\mathcal{Q}$-learning, in particular for non-discounted, non-episodic problems

Consider average reward $\rho = \lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} E[r_t]$

Value is define here as "above average":

$$V(s_t) = \sum_{k=1}^{\infty} E[r_{t+k} - \rho | s_t = s]$$

$$\mathcal{Q}(s_t, a_t) = \sum_{k=1}^{\infty} E[r_{t+k} - \rho | s_t = s, a_t = a]$$

Relative value function (relative to the average)

ρ is adapted and measures (average) success

Implies a different concept of optimality in non-episodic tasks

A. Schwartz (1993) A reinforcement learning method
for maximizing undiscounted rewards. In ICML, 298-305

R-learning: Algorithm

1. Initialise ρ and $\mathcal{Q}(s,a)$
2. Observe s_t and choose a_t (e.g. ε-greedy), execute a_t
3. Observe r_{t+1} and s_{t+1}
4. Update

$$\mathcal{Q}_{t+1}(s_t,a_t) = (1-\eta)\,\mathcal{Q}_t(s_t,a_t) + \eta\left(r_{t+1} - \rho_t + \max_a \mathcal{Q}_t(s_{t+1},a)\right)$$

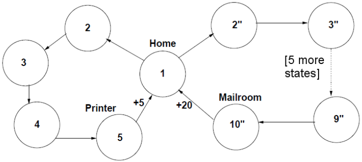
5. If $\mathcal{Q}(s_t,a_t) = \max_a \mathcal{Q}(s_t,a)$ then

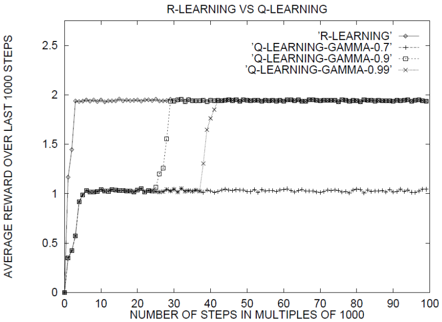
$$\rho_{t+1} = (1-\alpha)\,\rho_t + \alpha\left(r_{t+1} + \max_a \mathcal{Q}_t(s_{t+1},a_{t+1}) - \max_a \mathcal{Q}_{t+1}(s,a)\right)$$

Hint: Choose $\eta \gg \alpha$ (Otherwise, for $r=0$, $\mathcal{Q}$-value may cease to change and the agent may get trapped in a suboptimal limit cycle.)

R vs. $\mathcal{Q}$: A simple example

Only one decision: Robot moves either to nearby printer ("o.k.") or to distant mail room ("good").

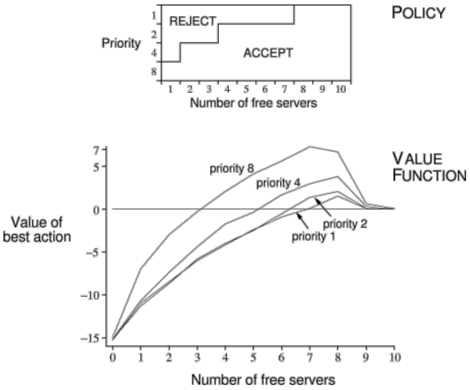




Similar to a 2AB, but waiting times differ for the "arms": $\mathcal{Q}$-learning with low γ favours the nearby goal, while its learning times get longer for larger γ. R-learning identifies the better choice quickly based on trajectory based reward averages.
Note that results may depend on parameters.

R-learning example: Access-control queuing task

- Customers pay 1, 2, 4, or 8 (this is a reward) of four different priorities to be served
- States are the number of for free servers
- Actions: customer at the head of the queue is either served or rejected (and removed from the queue)
- Proportion of high priority customers in the queue is $h = 0.5$



- Busy server becomes free with prob. $p = 0.06$ (p and h are not known to the algorithm) on each time step

http://webdocs.cs.ualberta.ca/~sutton/book/ebook/node67.html

R-learning: Conclusions

- An off-policy learning method for on-going learning
- May be superior to discounted methods
- May get trapped in limit cycles (exploration is important)
- Success depends on the parameters η, α, ε (exploration rate)

For details see:

S. Mahadevan (1996). Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, **22**(1-3), 159-195.

Exploration-exploitation dilemma

- Remember MABs: Exploratory moves were necessary to find promising actions.
- RL: Actions influence the future reward. It is necessary to explore the sequence of actions for all state, i.e. policies.
- Whether or not a policy gives high reward requires the agent to follow this policy.

The agent can either

- choose a policy, evaluate it, and move on to better policies

or

- Collect all available information and use it simultaneously to construct a good policy

What is a policy?

Deterministic: Function that maps states to actions. $\pi : \mathcal{S} \to \mathcal{A}$

$$a = \pi(s)$$

Examples: Standard policy in $\mathcal{Q}$-learning:
$a_t = \pi(s_t) = \arg\max_a \mathcal{Q}(s_t, a)$ (after convergence).

Stochastic: Probability of an action given a state.
$\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ with $\sum_{a \in \mathcal{A}_s} \pi(s, a) = 1$ for all s

$$P(a|s) = \pi(a, s)$$

Examples: random, Boltzmann policy

Partial policy: π is not necessarily defined for all $s \in \mathcal{S}$, e.g. a policy obtained from demonstration by a human teacher. Can be completed by defaults or combined with other partial policies.

On-policy learning vs off-policy learning (preliminary)

On-policy (TD(λ), SARSA)

- Start with a simple soft policy
- Sample state space with this policy
- Improve policy

Off-policy ($\mathcal{Q}$-learning, R-learning)

- Gather information from (partially) random moves
- Evaluate states as if a greedy policy was used
- Slowly reduce randomness

SARSA and $\mathcal{Q}$-learning

- SARSA and $\mathcal{Q}$-learning can be represented as look-up tables

$\mathcal{Q}$-learning (off-policy): $a_t = \arg\max_a \mathcal{Q}(s_t, a)$ (plus exploration)

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

SARSA (on-policy):

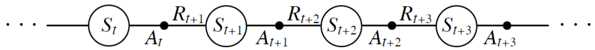
$$\mathcal{Q}_{t+1}(s_t, a_t) = (1-\eta)\,\mathcal{Q}_t(s_t, a_t) + \eta\left(r_{t+1} + \gamma \mathcal{Q}_t(s_{t+1}, a_{t+1})\right)$$

- $\mathcal{Q}$-learning: $V(s_{t+1}) = \max_a \mathcal{Q}(s_{t+1}, a)$, but a_{t+1} can be anything
- SARSA: $a_t \sim \pi(s_t, \cdot)$ and update rule learns the exact value function for $\pi(s, a)$
- How does the policy improve for SARSA ?

SARSA algorithm

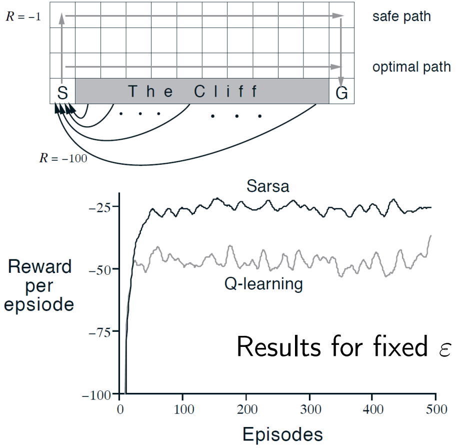
- Initialise $\mathcal{Q}_0(s, a)$
- Repeat (for each episode)
  - Initialise s_0
  - Choose a_0 from s_0 using policy derived from $\mathcal{Q}$ (e.g., ε-greedy)
  - Repeat (for each step of episode):
    - Take action a_t, observe r_{t+1}, s_{t+1}
    - Choose a' form s' using policy derived from $\mathcal{Q}$ (e.g., ε-greedy)
    - $\mathcal{Q}_{t+1}(s_t, a_t) = (1-\eta)\,\mathcal{Q}_t(s_t, a_t) + \eta\left(r_{t+1} + \gamma \mathcal{Q}_t(s_{t+1}, a_{t+1})\right)$
    - set $t = t + 1$
  - until s_{t+1} is terminal

SARSA means: State $\rightarrow$ Action $\rightarrow$ Reward $\rightarrow$ State $\rightarrow$ Action



SARSA vs. Q: The cliff walking task (S&B, example 6.6)

- $r = -1$ for every step, $r = -100$ for falling down the cliff
- ε-greedy with $\varepsilon = 0.1$ (see figure)
- ε-greedy with $\varepsilon \to 0$ both methods converge to the (now safe) optimal path



On-policy learning vs off-policy learning (according to S&B)

- On-policy methods
  - attempt to evaluate or improve the policy that is used to make decisions
  - often use *soft* action choice, i.e. $\pi(s,a) > 0 \ \forall a$
  - commit to always exploring and try to find the best policy that still explores
  - may become trapped in local minima
- Off-policy methods
  - evaluate one policy while following another, e.g. tries to evaluate the greedy policy while following a more exploratory scheme
  - the policy used for behaviour should be soft
  - policies may not be sufficiently similar
  - may be slower (only the part after the last exploration is reliable), but remains more flexible if alternative routes appear
- May lead to the same result (e.g. after greedification)

TD: Temporal difference learning

- TD learns value function $V(s)$ directly.
- TD is on-policy, i.e. the resulting value function depends on policy that is used.
- Information from policy-dependent sampling of the value function is not used immediately to improve the policy.
- TD-learning as such is not an RL algorithm, but can be used for RL if transition probabilities $p(s'|s,a)$ are known.
- One way to use TD for RL is SARSA, where the transition probabilities are implicitly included in the state-action values.

Temporal Difference (TD) Learning for Value Prediction

Ideal value function

$$\begin{aligned} V_t &= \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots \\ &= r_t + \gamma \left(r_{t+1} + \gamma r_{t+2} + \cdots \right) \\ &= r_t + \gamma \sum_{\tau=t+1}^{\infty} \gamma^{\tau-(t+1)} r_\tau \\ &= r_t + \gamma V_{t+1} \end{aligned}$$

During learning, the value function is based on estimates of V_t and V_{t+1} and may not obey this relation. If all states and all actions are sampled sufficiently often, then the requirement of consistency, i.e. minimisation of the absolute value of the δ error (δ for $\delta\iota\alpha\varphi o\rho\acute{\alpha}$)

$$\delta_{t+1} = r_t + \gamma \hat{V}_{t+1} - \hat{V}_t$$

will move the estimates $\hat{V}_t$ and $\hat{V}_{t+1}$ towards the ideal values.

The simplest TD algorithm

Let $\hat{V}_t$ be the t-th iterate of a learning rule for estimating the value function V.

Let s_t the state of the system at time step t.

$$\delta_{t+1} = r_t + \gamma \hat{V}_t(s_{t+1}) - \hat{V}_t(s_t)$$

$$\hat{V}_{t+1}(s) = \begin{cases} \hat{V}_t(s) + \eta\delta_{t+1} & \text{if } s = s_t \\ \hat{V}_t(s) & \text{otherwise} \end{cases}$$

$$\begin{aligned} \hat{V}_{t+1}(s_t) = \hat{V}_t(s_t) + \eta\delta_{t+1} &= \hat{V}_t(s_t) + \eta\left(r_t + \gamma \hat{V}_t(s_{t+1}) - \hat{V}_t(s_t)\right) \\ &= (1-\eta)\hat{V}_t(s_t) + \eta\left(r_t + \gamma \hat{V}_t(s_{t+1})\right) \end{aligned}$$

The update of the estimate $\hat{V}$ is an exponential average over the cumulative expected reward.

TD(0) Algorithm

Initialise η and γ and execute after each state transition

function TD0(s,r,$s1$,V) {

$\delta := r + \gamma * V[s1] - V[s]$;

$V[s] := V[s] + \eta * \delta$;

return V; }

Remarks

- If the algorithm converges it must converge to a value function where the expected temporal differences are zero for all states.
- The continuous version of the algorithm can be shown to be globally asymptotically stable
- TD(0) is a stochastic approximation algorithm. If the system is ergodic and the learning rate is appropriately decreased, it behaves like the continuous version.

Robbins-Monro conditions

How to choose learning rates? If

$$\sum_{t=0}^{\infty} \eta_t = \infty \quad \text{and} \quad \sum_{t=0}^{\infty} \eta_t^2 < \infty,$$

then $V_t(\cdot)$ will behave as the temporally continuous variant

$$\frac{dV(\cdot)}{dt} = r + (\gamma P - I)\, V(\cdot)$$

Choosing e.g. $\eta_t = c\, t^{-\alpha}$, the conditions hold for $\alpha \in \left(\frac{1}{2}, 1\right]$:

- $\alpha > 1$: forced convergence, but possibly without reaching goal
- $\alpha = 1$: smallest step sizes, but still possible
- $\alpha \leq \frac{1}{2}$: large fluctuations can happen even after long time

Iterate-averaging (Polyak & Juditsky, 1992) gives best possible asymptotic rate of convergence

Practically: fixed step sizes or finite-time reduction (see earlier slide)

Actor-Critic Methods

- Policy (actor) is represented independently of the (state) value function (critic)
- A number of variants exist, in particular among the early reinforcement learning algorithms

Advantages[1]

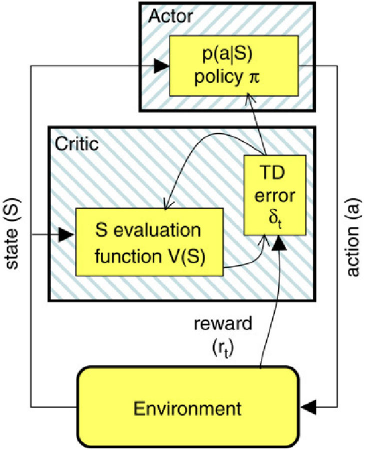
- AC methods require minimal computation in order to select actions which is beneficial in continuous cases, where search becomes a problem.
- They can learn an explicitly stochastic policy, i.e. learn the optimal action probabilities. Useful in competitive and non-Markov cases[2].
- A plausible model of biological reinforcement learning
- Recently also an off-policy variant was proposed

[1] Mark Lee following Sutton&Barto

[2] see, e.g., Singh, Jaakkola, and Jordan, 1994

Actor-Critic Methods

- Actor aims at improving policy (adaptive search element)
- Critic evaluates the current policy (adaptive critic element)
- Learning is based on the TD error δ_t
- Reward only known to the critic
- Critic should improve as well



Example: Policies for the inverted pendulum

- Exploitation (**actor**): Escape from low-reward regions as fast as possible
- aims at max. r
- e.g. Inverted pendulum task: Wants to stay near the upright position
- preferentially greedy and deterministic

- Exploration (**critic**): Find examples where learning is optimal
- aims at max. δ
- e.g. Inverted pendulum task: Wants to move away from the upright position
- preferentially non-deterministic

AC methods vs. other RL algorithms

- SARSA and $\mathcal{Q}$-learning do not have an explicit policy representation, in a sense they are thus "critic-only" algorithms.
- There are also "actor-only" methods which directly try to improve the policy, e.g. REINFORCE (Williams, 1992).
- AC is advantageous for continuous problems (later!), where $\mathcal{Q}$ and SARSA may become unstable due to the concomitant function approximation.

Conclusions

- Both on- and off-policy methods have their advantages
  - If a good starting policy is available: on-policy may be interesting, but may not explore other policies well
  - If more exploration is necessary, then perhaps off-policy is advisable, but maybe slow
- Actor-critic is of historical interest, but we will come back to this.
- Also TD learning including value iteration and policy iteration will be revisited shortly.
- We need a theoretical framework to understand better how the algorithms work. For this purpose, we will study Markov decision problems (MDPs) next.

Literature: R-learning: S&B (2), section 11.2; SARSA: S&B (2), section 6.4