Q-learning

From Wikipedia, the free encyclopedia

This article **may be too technical for most readers to understand**. Please help improve this article to make it understandable to non-experts, without removing the technical details. The talk page may contain suggestions. *(September 2010)*

Q-learning is a model-free reinforcement learning technique. Specifically, Q-learning can be used to find an optimal action-selection policy for any given (finite) Markov decision process (MDP). It works by learning an action-value function that ultimately gives the expected utility of taking a given action in a given state and following the optimal policy thereafter. A policy is a rule that the agent follows in selecting actions, given the state it is in. When such an action-value function is learned, the optimal policy can be constructed by simply selecting the action with the highest value in each state. One of the strengths of Q-learning is that it is able to compare the expected utility of the available actions without requiring a model of the environment. Additionally, Q-learning can handle problems with stochastic transitions and rewards, without requiring any adaptations. It has been proven that for any finite MDP, Q-learning eventually finds an optimal policy, in the sense that the expected value of the total reward return over all successive steps, starting from the current state, is the maximum achievable.

Algorithm [edit]

The problem model consists of an agent, states *S* and a set of actions per state *A*. By performing an action $a \in A$, the agent can move from state to state. Executing an action in a specific state provides the agent with a reward (a numerical score). The goal of the agent is to maximize its total reward. It does this by learning which action is optimal for each state. The action that is optimal for each state is the action that has the highest long-term reward. This reward is a weighted sum of the expectation values of the rewards of all future steps starting from the current state, where the weight for a step from a state Δt steps into the future is calculated as $\gamma^{\Delta t}$. Here, γ is a number between 0 and 1 ($0 \leq \gamma \leq 1$) called the discount factor and trades off the importance of sooner versus later rewards.γ is the likelihood to succeed (or survive) at every step Δt.

The algorithm therefore has a function that calculates the Quantity of a state-action combination:

$$Q : S \times A \rightarrow \mathbb{R}$$

Before learning has started, *Q* returns an (arbitrary) fixed value, chosen by the designer. Then, each time the agent selects an action, and observes a reward and a new state that may depend on both the previous state and the selected action, "Q" is updated. The core of the algorithm is a simple value iteration update. It assumes the old value and makes a correction based on the new information.

$$Q_{t+1}(s_t, a_t) = \underbrace{Q_t(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q_t(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q_t(s_t, a_t)}_{\text{old value}} \right)$$

where R_{t+1} is the reward observed after performing a_t in s_t, and where $\alpha_t(s, a)$ ($0 < \alpha \leq 1$) is the learning rate (may be the same for all pairs).

An episode of the algorithm ends when state s_{t+1} is a final state (or, "absorbing state"). However, Q-learning can also learn in non-episodic tasks. If the discount factor is lower than 1, the action values are finite even if the problem can contain infinite loops.

Note that for all final states s_f, $Q(s_f, a)$ is never updated and thus retains its initial value. In most cases, $Q(s_f, a)$ can be taken to be equal to zero.

Influence of variables on the algorithm [edit]

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
فارسی
Italiano
日本語
Norsk bokmål
Română
Русский
中文
Edit links

Learning rate [edit]

The learning rate determines to what extent the newly acquired information will override the old information. A factor of 0 will make the agent not learn anything, while a factor of 1 would make the agent consider only the most recent information. In fully deterministic environments, a learning rate of $\alpha_t(s,a) = 1$ is optimal. When the problem is stochastic, the algorithm still converges under some technical conditions on the learning rate, that require it to decrease to zero. In practice, often a constant learning rate is used, such as $\alpha_t(s,a) = 0.1$ for all t.[1]

Discount factor [edit]

The discount factor γ determines the importance of future rewards. A factor of 0 will make the agent "myopic" (or short-sighted) by only considering current rewards, while a factor approaching 1 will make it strive for a long-term high reward. If the discount factor meets or exceeds 1, the action values may diverge. For $\gamma = 1$, without a terminal state, or if the agent never reaches one, all environment histories will be infinitely long, and utilities with additive, undiscounted rewards will generally be infinite.[2]

Initial conditions (Q_0) [edit]

Since Q-learning is an iterative algorithm, it implicitly assumes an initial condition before the first update occurs. A high initial value, also known as "optimistic initial conditions",[3] can encourage exploration: no matter what action will take place, the update rule will cause it to have lower values than the other alternative, thus increasing their choice probability. Recently, it was suggested that the first reward r could be used to reset the initial conditions[*citation needed*]. According to this idea, the first time an action is taken the reward is used to set the value of Q. This will allow immediate learning in case of fixed deterministic rewards. Surprisingly, this resetting-of-initial-conditions (RIC) approach seems to be consistent with human behaviour in repeated binary choice experiments.[4]

Implementation [edit]

Q-learning at its simplest uses tables to store data. This very quickly loses viability with increasing levels of complexity of the system it is monitoring/controlling. One answer to this problem is to use an (adapted) artificial neural network as a function approximator, as demonstrated by Tesauro in his Backgammon playing temporal difference learning research.[5]

More generally, Q-learning can be combined with function approximation.[6] This makes it possible to apply the algorithm to larger problems, even when the state space is continuous, and therefore infinitely large. Additionally, it may speed up learning in finite problems, due to the fact that the algorithm can generalize earlier experiences to previously unseen states.

Early study [edit]

Q-learning was first introduced by Watkins[7] in 1989. The convergence proof was presented later by Watkins and Dayan[8] in 1992.

Variants [edit]

Delayed Q-learning is an alternative implementation of the online Q-learning algorithm, with Probably approximately correct learning (PAC).[9]

Because the maximum approximated action value is used in the Q-learning update, in noisy environments Q-learning can sometimes overestimate the actions values, slowing the learning. A recent variant called Double Q-learning was proposed to correct this. [10]

Greedy GQ is a variant of Q-learning to use in combination with (linear) function approximation.[11] The advantage of Greedy GQ is that convergence guarantees can be given even when function approximation is used to estimate the action values.

Q-learning may suffer from slow rate of convergence, especially when the discount factor γ is close to one.[12] Speedy Q-learning, a new variant of Q-learning algorithm, deals with this problem and achieves a provably same rate of convergence as model-based methods such as value iteration.[13]

A recent application of Q-learning to deep learning, by Google DeepMind, titled "deep reinforcement learning" or "deep Q-networks", has been successful at playing some Atari 2600 games at expert human levels. Preliminary results were presented in 2014, with a paper published in February 2015 in Nature[14]

See also [edit]

- Reinforcement learning
- Temporal difference learning
- SARSA
- Iterated prisoner's dilemma
- Game theory
- Fitted Q iteration algorithm

External links [edit]

- Watkins, C.J.C.H. (1989). Learning from Delayed Rewards. PhD thesis, Cambridge University, Cambridge, England.
- Strehl, Li, Wiewiora, Langford, Littman (2006). PAC model-free reinforcement learning

- *Reinforcement Learning: An Introduction* by Richard Sutton and Andrew S. Barto, an online textbook. See "6.5 Q-Learning: Off-Policy TD Control".
- Piqle: a Generic Java Platform for Reinforcement Learning
- Reinforcement Learning Maze, a demonstration of guiding an ant through a maze using Q-learning.
- Q-learning work by Gerald Tesauro
- Q-learning work by Tesauro Citeseer Link
- Q-learning algorithm implemented in processing.org language
- Solution for the pole balancing problem with Q(lambda) / SARSA(lambda) and the fourier basis in javascript

References [edit]

1. ^ Reinforcement Learning: An Introduction. Richard Sutton and Andrew Barto. MIT Press, 1998.
2. ^ Stuart J. Russell; Peter Norvig (2010). *Artificial Intelligence: A Modern Approach* (Third ed.). Prentice Hall. p. 649. ISBN 978-0136042594.
3. ^ http://webdocs.cs.ualberta.ca/~sutton/book/ebook/node21.html
4. ^ Shteingart, H; Neiman, T; Loewenstein, Y (May 2013). "The Role of First Impression in Operant Learning". *J Exp Psychol Gen.* **142** (2): 476–88. doi:10.1037/a0029550. PMID 22924882.
5. ^ Tesauro, Gerald (March 1995). "Temporal Difference Learning and TD-Gammon". *Communications of the ACM* **38** (3). Retrieved 2010-02-08.
6. ^ Hado van Hasselt. Reinforcement Learning in Continuous State and Action Spaces. In: Reinforcement Learning: State of the Art, Springer, pages 207-251, 2012
7. ^ Watkins, C.J.C.H., (1989), Learning from Delayed Rewards. Ph.D. thesis, Cambridge University.
8. ^ Watkins and Dayan, C.J.C.H., (1992), 'Q-learning.Machine Learning'
9. ^ Alexander L. Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L. Littman. Pac model-free reinforcement learning. In Proc. 22nd ICML 2006, pages 881–888, 2006.
10. ^ van Hasselt, Hado (2011). "Double Q-learning" (PDF). *Advances in Neural Information Processing Systems* **23**: 2613–2622.
11. ^ Hamid Maei, and Csaba Szepesv{\'a}ri, Shalabh Bhatnagar and Richard Sutton. Toward off-policy learning control with function approximation. In proceedings of the 27th International Conference on Machine Learning, pages 719-726, 2010.
12. ^ Csaba Szepesva´ri. The asymptotic convergence-rate of Q-learning. *Advances in Neural Information Processing Systems* 10, Denver, Colorado, USA, 1997.
13. ^ Gheshlaghi Azar, Mohammad; Munos, Remi; Ghavamzadeh, Mohammad; Kappen, Hilbert J. (2011). "Speedy Q-Learning" (PDF). *Advances in Neural Information Processing Systems* **24**: 2411–2419.
14. ^ Mnih, Volodymyr et al. (2015). "Human-level control through deep reinforcement learning" (PDF) **518**. pp. 529–533.

Categories: Machine learning algorithms