Q Learning

Introduction to Artificial Intelligence with Mathematics
Lecture Notes

Ganguk Hwang

Department of Mathematical Sciences
KAIST

Model-free Learning: Q-learning

In model-based learning, an agent has a great deal of knowledge on the environment

- transition probabilities between states
- rewards on states

In model-free learning, e.g, Q-learning, an agent does not need to have any model of the environment.

- It only needs to know what states exist and what actions are possible for each state.

Reference:

1. D. Bertsekas and J. Tsitsiklis, Neuro-dynamic Programming, Athena Scientific, 1996.
2. R.S. Sutton and A.G. Barto, Reinforcement Learning: An Introduction, 2nd ed., The MIT Press, 2018.

To motivate the discussion, suppose that we are interested in solving a system of equations of the form

$$Hq = q$$

where H is a function from $\mathbb{R}^n$ to itself.

One possible way to solve it is to use

$$q_{k+1} = (1 - \gamma_k)q_k + \gamma_k H q_k.$$

Suppose further that it is difficult to know Hq, but we have access to a random variable of the form $s_k = Hq_k + w_k$ where w_k is a random noise term. In the case, one reasonable choice is to use

$$q_{k+1} = (1 - \gamma_k)q_k + \gamma_k(Hq_k + w_k),$$

which is called *stochastic iterative or stochastic approximation* algorithm.

- In an algorithm of this type, there is an incentive to use a small stepsize γ_k because it reduces the impact of the noise w.
- On the other hand, the smaller values of γ_k generally lead to slower progress.
- This conflict can be resolved by using a variable stepsize that decreases towards zero as the algorithm progresses.

It is usually assumed that

$$\sum_{k=1}^{\infty} \gamma_k = \infty, \qquad \sum_{k=1}^{\infty} \gamma_k^2 < \infty.$$

For the motivation on the assumption, consider the following algorithm studied by Robbins and Monro.

$$q_{k+1} = (1 - \gamma_k) q_k + \gamma_k x_k.$$

where x_k are independent random variables with $E[x_k] = q^*$ and $E[||x_k - q^*||^2] = \sigma^2 < \infty$.

It then follows that

$$\begin{aligned} q_{k+1} - q^* &= (1-\gamma_k)(q_k - q^*) + \gamma_k(x_k - q^*) \\ &= \prod_{n=1}^{k}(1-\gamma_n)(q_1 - q^*) + \sum_{i=1}^{k}\prod_{n=i+1}^{k}(1-\gamma_n)\gamma_i(x_i - q^*). \end{aligned}$$

$$E[(q_{k+1} - q^*)^2] = \prod_{n=1}^{k}(1-\gamma_n)^2 E[(q_1 - q^*)^2] + \sum_{i=1}^{k}\gamma_i^2 \prod_{n=i+1}^{k}(1-\gamma_n)^2\sigma^2$$

where we use the independence of x_k's.

For the first term, as $k \to \infty$ we have

$$\log \prod_{n=1}^{k} (1-\gamma_n)^2 = 2 \sum_{n=1}^{k} \log(1-\gamma_n) \sim -2 \sum_{n=1}^{k} \gamma_n \to -\infty.$$

So the first term goes to 0.

For the second term, suppose that γ_n is decreasing and bounded by 1. Then for any $n \leq l$ we know that

$$\prod_{i=n+1}^{k} (1-\gamma_i) \leq \prod_{j=l+1}^{k} (1-\gamma_j) \leq 1.$$

With $\log(1-x) \leq -x$, for any $1 \leq m \leq k$, we have

$$\sum_{i=1}^{k} \gamma_i^2 \prod_{n=i+1}^{k} (1-\gamma_n)^2$$

$$= \sum_{i=1}^{m} \gamma_i^2 \prod_{n=i+1}^{k} (1-\gamma_n)^2 + \sum_{i=m+1}^{k} \gamma_i^2 \prod_{n=i+1}^{k} (1-\gamma_n)$$

$$\leq \prod_{n=m+1}^{k} (1-\gamma_n)^2 \sum_{i=1}^{m} \gamma_i^2 + \gamma_m \sum_{i=m+1}^{k} \prod_{n=i+1}^{k} (1-\gamma_n)\gamma_i$$

$$\leq e^{\left(-2\sum_{n=m+1}^{k} \gamma_n\right)} \sum_{i=1}^{m} \gamma_i^2 + \gamma_m \sum_{i=m+1}^{k} \left[\prod_{n=i+1}^{k} (1-\gamma_n) - \prod_{n=i}^{k} (1-\gamma_n) \right]$$

$$\leq e^{\left(-2\sum_{n=m+1}^{k} \gamma_n\right)} \sum_{i=1}^{m} \gamma_i^2 + \gamma_m \left[1 - \prod_{n=m+1}^{k} (1-\gamma_n) \right]$$

$$\leq e^{\left(-2\sum_{n=m+1}^{k} \gamma_n\right)} \sum_{i=1}^{m} \gamma_i^2 + \gamma_m \to 0 \text{ as } k \to \infty.$$

To go further for Q-learning, we are interested in a more specific problem as follows. Let v be a random variable whose conditional distribution, given q, is known. Consider the problem

$$E[g(q, v)] = q$$

where g is a known function and the expectation is with respect to the conditional distribution of v, given q.
One possible algorithm is

$$q = (1 - \gamma)q + \gamma E[g(q, v)],$$

but it is usually difficult to compute $E[g(q, v)]$.

We can use the sample mean to estimate $E[g(q, v)]$, but the other extreme is to use a single sample $\tilde{v}$ as follows.

$$q = (1 - \gamma)q + \gamma g(q, \tilde{v}),$$

which can be written as

$$q = (1 - \gamma)q + \gamma(E[g(q, v)] + g(q, \tilde{v}) - E[g(q, v)]).$$

That is,

$$q = (1 - \gamma)q + \gamma(Hq + w)$$

where $Hq = E[g(q, v)]$ and $w = g(q, \tilde{v}) - E[g(q, v)]$.
Note that $E[w] = 0$ which means we have a zero mean noise in the problem. If the variance of the noise is not much large, the convergence of the algorithm is expected.

Q-learning

For a policy π, define the state-action values (or Q-values)

$$Q^{\pi}(s,a) = r(s,a) + \lambda \sum_{j \in S} p(j|s,a) v_{\lambda}^{\pi}(j)$$

which is the expected discounted reward when the agent takes action a at state s and then follows the policy π.

For π^* and $Q^* = Q^{\pi^*}$, from $v_{\lambda}^*(s) = \sup_{a \in A} Q^*(s,a)$

$$Q^*(s,a) = r(s,a) + \lambda \sum_{j \in S} p(j|s,a) \sup_{b \in A} Q^*(j,b)$$

or equivalently, at stage t

$$Q^*(s_t,a_t) = E\left[r(s_t,a_t) + \lambda \sup_{b \in A} Q^*(X_{t+1},b) \right].$$

- Q-learning update with step size α

$$Q_{new}(s_t, a_t) = (1-\alpha_t)Q_{old}(s_t, a_t)+\alpha_t \left(r(s_t, a_t) + \lambda \sup_{b \in A} Q_{old}(X_{t+1}, b) \right)$$

equivalently,

$$\begin{aligned} & Q_{new}(s_t, a_t) \\ & \quad = Q_{old}(s_t, a_t) \\ & \qquad +\alpha_t \left(r(s_t, a_t) + \lambda \sup_{b \in A} Q_{old}(X_{t+1}, b) - Q_{old}(s_t, a_t) \right). \end{aligned}$$

Note that

$$r(s_t, a_t) + \lambda \sup_{b \in A} Q_{old}(X_{t+1}, b) - Q_{old}(s_t, a_t)$$

is the difference between predictions that we make in consecutive stages.

Let

$$HQ(s_t, a_t) = E\left[r(s_t, a_t) + \lambda \sup_{b \in A} Q(X_{t+1}, b)\right].$$

Then,

$$\begin{aligned}
&Q_{new}(s_t, a_t) \\
&\quad = Q_{old}(s_t, a_t) + \alpha_t \left(r(s_t, a_t) + \lambda \sup_{b \in A} Q_{old}(X_{t+1}, b) - Q_{old}(s_t, a_t)\right) \\
&\quad = (1 - \alpha_t) Q_{old}(s_t, a_t) \\
&\qquad + \alpha_t \left(HQ_{old}(s_t, a_t) + r(s_t, a_t) + \lambda \sup_{b \in A} Q_{old}(X_{t+1}, b) - HQ_{old}(s_t, a_t)\right) \\
&\quad = (1 - \alpha_t) Q_{old}(s_t, a_t) + \alpha_t \left(HQ_{old}(s_t, a_t) + w(s_t, a_t)\right)
\end{aligned}$$

where

$$w(s_t, a_t) = r(s_t, a_t) + \lambda \sup_{b \in A} Q_{old}(X_{t+1}, b) - HQ_{old}(s_t, a_t).$$

Convergence in Q-learning

If all state-action pairs (s, a) are visited infinitely often, $\sum_{t=1}^{\infty} \alpha_t = \infty$, and $\sum_{t=1}^{\infty} \alpha_t^2 < \infty$ (with the fact that H is a contraction map and the rewards are bounded), then $Q(s_t, a_t)$ converges to $Q^*(s, a)$ with probability 1.

- We use, e.g., $\alpha_t = \frac{1}{N_t(s_t, a_t)}$
- $N_t(s_t, a_t)$ is the number of visits to (s_t, a_t) until t.
- References
  - C. Watkins, Learning from delayed rewards, Ph.D. Thesis, Univ. of Cambridge, England, 1989.
  - D. Bertsekas and J. Tsitsiklis, Neuro-dynamic Programming, Athena Scientific, 1996.

Proof of Contraction of HQ

Since

$$HQ(s,a) = r(s,a) + \lambda \sum_{j \in S} p(j|s,a) \sup_{b \in A} Q(j,b),$$

it follows that

$$\begin{aligned}
||HQ_1 - HQ_2||_\infty &= \sup_{s,a} \lambda \left| \sum_{j \in S} p(j|s,a) \left[\sup_{b \in A} Q_1(j,b) - \sup_{b \in A} Q_2(j,b) \right] \right| \\
&\leq \sup_{s,a} \lambda \sum_{j \in S} p(j|s,a) \left| \sup_{b \in A} Q_1(j,b) - \sup_{b \in A} Q_2(j,b) \right| \\
&\leq \sup_{s,a} \lambda \sum_{j \in S} p(j|s,a) \sup_{x,b} |Q_1(x,b) - Q_2(x,b)| \\
&= \lambda ||Q_1 - Q_2||_\infty
\end{aligned}$$

Exploitation and Exploration Dilemma

We still have one more problem - exploitation and exploration problem.

- an agent is learning in an online fashion
- how should an agent choose which action to test?
- Intuition
  - When little is known about the environment, it is important to explore and try unknown actions.
  - After some time when a great deal of knowledge is gained, little information will be gained and an agent choose the action that looks the best.

Boltzmann exploration

Boltzmann distribution is used to select action a

$$p(a|s) = \frac{e^{\frac{1}{T}Q(s,a)}}{\sum_{b \in A} e^{\frac{1}{T}Q(s,b)}}$$

- The temperature parameter T controls the probability of selecting non-optimal actions.
- If T is large, all actions will be selected uniformly.
- If T is close to 0, the best action will be chosen.
- We begin with T large and gradually decrease it over time (decreasing exploration).

On-policy and Off-policy

- An off-policy method learns the value of the optimal policy independently of the agent's actions.
- An on-policy learner learns the value of the policy being carried out by the agent including the exploration steps.

We provide an on-policy called SARSA (State-Action-Reward-State-Action) which is given by

$$Q_{new}(s_t, a_t) = (1 - \alpha_t)Q_{old}(s_t, a_t) + \alpha_t \left(r(s_t, a_t) + \lambda Q_{old}(s_{t+1}, a_{t+1})\right).$$

Note that to update the Q-value, we need

$$(s_t, a_t, r(s_t, a_t), s_{t+1}, a_{t+1})$$

and the Q-value is updated based on the policy actually used (i.e., a_{t+1}), which is the reason why SARSA is called an on-policy method.

More topics

- Temporal Difference Learning
- Value function approximations
- Policy Gradient methods
- Deep Reinforcement Learning and Deep Q-Network (DQN)
- etc.