Chapter 16

Q-Learning

[1]Q-learning is a cornerstone of the field of reinforcement learning. It simply consists in replacing the Q-factor updates in Q-value iteration, which require computations of expectations, by a simulation based approximation. It can be viewed as a stochastic approximation algorithm, from which the convergence to the optimal Q-factors follows, under certain assumptions regarding sufficient state-action pairs exploration. We obtain a method that is completely model free, that is, it does not require the knowledge of the transition probabilities of the underlying Markov chain, only the capability of simulating these transitions or of executing them on a physical system. The method does not really scale however to large state and control spaces, and must be combined or replaced by methods using approximation architectures in complex problems, which we will cover in more details in chapter 17.

16.1 Review of Q-factor Based DP

Recall the definition of the optimal Q-factors

$$Q^*(x,u) = E[c(x_0,u_0,x_1) + \alpha J^*(x_1) | x_0 = x, u_0 = u]$$

and of the operator F used in Q-value iteration (see section 7.1)

$$(FQ)(x,u) = E[c(x_0,u_0,x_1) + \alpha \min_{u' \in \mathsf{U}(x_1)} Q(x_1,u') | x_0 = x, u_0 = u]$$

For finite state spaces, we have

$$\begin{aligned} Q^*(i,u) &= \sum_{j=1}^n p_{ij}(u)(c(i,u,j) + \alpha J^*(j)), \\ (FQ)(i,u) &= \sum_{j=1}^n p_{ij}(u) \left(c(i,u,j) + \alpha \min_{u' \in \mathsf{U}(j)} Q(j,u') \right). \end{aligned}$$

[1]This version: November 3rd 2009.

The analog of Bellman's equation in terms of the Q-factors is then $Q^* = FQ^*$, i.e., for finite state spaces,

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u) \left(c(i,u,j) + \alpha \min_{u' \in \mathsf{U}(j)} Q^*(j,u') \right),$$

and the Q-value iteration $Q_{k+1} = FQ_k$ converges to Q^*. In fact the theory for Q-factors is a special case of the standard theory if we take as new state the state-action pairs (i,u). It is easy to see that the operator F is again an α-contraction.

The Q-learning update rule is a simulation-based approximation of the expectation in the expression of FQ. We can use it for example in the asynchronous Q-value iteration of section 7.1. Recall that this algorithm works as follows. An infinite sequence of state-action pairs $\{(x_k, u_k)\}$ is generated, such that each pair appears infinitely often. Then if the pair (x_k, u_k) was just generated, asynchronous Q-VI updates the Q-factor $Q(x_k, u_k)$ and leaves the other Q-factors unchanged

$$Q_{k+1}(x,u) = \begin{cases} (FQ_k)(x_k, u_k), & \text{if } (x,u) = (x_k,u_k), \\ Q_k(x,u), & \text{if } (i,u) \neq (x_k,u_k). \end{cases} \tag{16.1}$$

When the Q-factor have converged to the optimal ones, we get an optimal stationary policy as

$$\mu^*(x) \in \arg\min_{u \in \mathsf{U}(x)} Q^*(x,u).$$

The update of the Q-factors however requires the computation of an expected value, which can be too difficult for very large state-spaces or impossible if the transition probabilities are not explicitly given (or previously estimated) but only provided indirectly by a simulator. Note also that the computation of the expectation requires a minimization over $u' \in \mathsf{U}(x_{k+1})$ for each possible value of x_{k+1}. We now replace the update (16.1) by the following Q-learning update rule. If (x_k, u_k) was just generated, we generate x_{k+1} according to the probabilities $p_{x_k y}(u_k)$ (e.g., via a simulator) and the Q-factor $Q(x_k, u_k)$ is updated as

$$Q_{k+1}(x,u) = \begin{cases} (1-\gamma_k) Q_k(x,u) + \gamma_k \Big(c(x,u,x_{k+1}) \\ +\alpha \min_{u' \in \mathsf{U}(x_{k+1})} Q_k(x_{k+1}, u') \Big), & \text{if } (x,u) = (x_k,u_k), \\ Q_k(x,u), \ \ \text{if } (x,u) \neq (x_k,u_k), \end{cases} \tag{16.2}$$

where γ_k is a positive stepsize, which should be diminishing to 0 at an appropriate rate. An equivalent way of writing the update in (16.2) is

$$\begin{aligned} Q_{k+1}(x,u) = & Q_k(x,u) + \gamma_k \, 1\{X_k = x, U_k = u\} \Big(c(x,u,X_{k+1}) \\ & + \alpha \min_{u' \in \mathsf{U}(X_{k+1})} Q_k(X_{k+1}, u') - Q_k(x,u) \Big). \end{aligned} \tag{16.3}$$

If the term $1\{X_k = x, U_k = u\}$ is omitted, we are using the Q-learning update rule in the standard Q-value iteration, which uses the same matrix Q_k for all the computations of Q_{k+1}. In any case, we can see that the Q-learning algorithm is a type of stochastic approximation algorithm which aims at solving the fixed point equation $FQ - Q = 0$, by recursively adjusting the parameters $\{Q(x,u)\}_{x,u}$, see chapter 15.

To review the usual averaging intuition this scheme, first replace the expectation operator

$$E\Big[c(x_k, u_k, x_{k+1}) + \alpha \min_{u' \in \mathsf{U}(x_{k+1})} Q_k(x_{k+1}, u')\Big| x_k = x, u_k = u\Big]$$

in asynchronous Q-VI by the sample mean

$$\frac{1}{n_k} \sum_{t \in T_k} c(x_t, u_t, x_{t+1}) + \alpha \min_{u' \in \mathsf{U}(x_{t+1})} Q_k(x_{t+1}, u'),$$

where n_k is the number of times the current state control pair (x_k, u_k) has appeared up to and including time k, and T_k are the actual iterations at which this pair has been generated

$$T_k = \{t | (x_t, u_t) = (x_k, u_k), 0 \leq t \leq k\}.$$

The issue is that the computation of the sample mean might be too hard. The first term

$$\frac{1}{n_k} \sum_{t \in T_k} c(x_t, u_t, x_{t+1})$$

can be recursively updated, but the second term

$$\frac{1}{n_k} \sum_{t \in T_k} \min_{u' \in \mathsf{U}(t_{t+1})} Q_k(x_{t+1}, u')$$

must be completely recomputed at each iteration using the current vector Q_k. On the other hand, this term can be recursively updated if we replace it by

$$\frac{1}{n_k} \sum_{t \in T_k} \min_{u' \in \mathsf{U}(x_{t+1})} Q_t(x_{t+1}, u').$$

This gives the Q-learning algorithm with $\gamma_k = 1/n_k$ using the standard manipulation to obtain recursive updates.

The Q-learning update is also often used in the optimistic policy iteration scheme using Q-factors as in section 7.2. The drawbacks of Q-learning are that the number of Q-factors (i.e., the number of state-control pairs) may be excessive, and the convergence too slow. The large size of the state-control space can be handled using state aggregation or linear approximation architectures for the Q-factors, see [Ber07b, sections 6.4.2-6.4.4].

16.2 Convergence Analysis of Value Iteration with Q-Learning

We now consider the convergence of Q-Value Iteration, for a finite state and control space, using the Q-learning update rule to replace the expectation in $Q_{k+1} = FQ_k$. The convergence analysis can be modified to work for the asynchronous scheme (16.2) without too much difficulty, see e.g. [Bor08]. At stage k of the algorithm, for the update of the Q-factor of the pair (x, u), we simulate a variable $X_{k+1}(x, u)$ with probability distribution $p_{x\cdot}(u)$. Then we view the iterations

$$Q_{k+1}(x,u) = Q_k(x,u) + \gamma_k\Big(c(x,u,X_{k+1}(x,u)) \\ + \alpha \min_{u'} Q_k(X_{k+1}(x,u),u') - Q_k(x,u)\Big)$$

as a stochastic approximation, where the iterates are the $n \times |\mathsf{U}|$ matrices $Q_k, k \geq 0$. So we have

$$Q_{k+1} = Q_k + \gamma_k(f(Q_k) + D_{k+1}),$$

where $f(Q_k)$ is a matrix with components

$$[f(Q_k)]_{x,u} = (FQ_k)(x,u) - Q_k(x,u)$$

and D_{k+1} is a (martingale difference) matrix with components

$$[D_{k+1}]_{x,u} = \alpha\left(\min_{u'} Q_k(X_{k+1}(x,u),u') - \sum_{j=1}^{n} p_{xj}(u) \min_{u'} Q_k(j,u')\right).$$

Assume that the stepsizes satisfy

$$\sum_{k=0}^{\infty} \gamma_k = \infty, \quad \sum_{k=0}^{\infty} \gamma_k^2 < \infty.$$

Then by inspection the corresponding ODE is

$$\dot{Q}(t) = (FQ)(t) - Q(t),$$

which is of the type studied in section 15.2 for F a contraction, see corollary 15.2.2. Since this ODE has a unique globally stable fixed point, the convergence results of chapter 15 guarantee that the Q-factors converge to the optimal ones (Q^* is the unique fixed point of F), if we can show that the stability assumption 4 of section 15.3 holds.

Proof of Stability of the Stochastic Approximation

The bottleneck for the application of the ODE method, in this case and in general for a priori unbounded state spaces, is to prove the stability property

4. There are a few general techniques available for this purpose, for example the stochastic Lyapunov function approach [KY03]. Here we follow instead an approach due to Borkar and Meyn, inspired by stability analysis techniques using fluid models in queueing networks [BM00, Bor08].

Suppose that the iterates x_n in (15.10) are scaled by their initial value

$$\tilde{x}_n = \frac{x_n}{r}, \text{ with } r := \max(|x_0|, 1).$$

Then the rescaled iterates follow the stochastic approximation

$$\tilde{x}_{n+1} = \tilde{x}_n + \gamma_n \left[\frac{f(r\tilde{x}_n)}{r} + \frac{D_{n+1}}{r} \right], \ n \geq 0.$$

Define $f_r(x) = f(rx)/r$. The interpolated process for the scaled iterates then is expected to follow the solutions of the ODE

$$\dot{x}(t) = f_r(x(t)).$$

For stability, we are interested in the situation where the initial condition is large, i.e. $r \to \infty$. Suppose that the (pointwise) limit $f_\infty = \lim_{r\to\infty} f_r$ exists. Then for large initial conditions the rescaled process should follow approximately the solutions of the ODE

$$\dot{x}(t) = f_\infty(x(t)). \tag{16.4}$$

We have the following theorem

Theorem 16.2.1. *Assume that assumptions 1-3 of section 15.3 are satisfied, and that the limit f_∞ exists. Moreover, assume the origin in $\mathbb{R}^d$ is an asymptotically stable equilibrium for the ODE (16.4). Then for any initial condition $x_0 \in \mathbb{R}^d$ we have*

$$\sup_n \|x_n\| < \infty, \ \textit{a.s.}$$

Remark. The function f_∞ satisfies $f_\infty(ax) = af_\infty(x)$ for $a > 0$ so the origin in $\mathbb{R}^d$ satisfies $f_\infty(0) = 0$, and moreover the origin is the only possible isolated equilibrium of (16.4). We can also show that if the origin is an asymptotically stable equilibrium, it is in fact globally exponentially asymptotically stable.

For the Q-learning algorithm, we have $f_\infty(Q) = F_\infty Q - Q$ with

$$[F_\infty Q]_{x,u} = \alpha \sum_{j=1}^{n} p_{xj}(u) \min_{u' in \mathsf{U}(j)} Q(j, u'),$$

which is also an α-contraction for $\| \cdot \|_\infty$, with the origin as fixed point and hence as globally asymptotically stable equilibrium of the corresponding ODE (16.4). Hence from the theorem we conclude that the Q-learning algorithm is stable and it converges to the optimal Q-factors by the discussion in the previous subsection.