7 Efficient Exploration

7.1 Overview

Efficient exploration of the action and state space is a crucial factor in the convergence rate of a learning scheme. An early survey of early exploration methods can be found in Thrun (1992)[1]. The distinction is made there between directed and undirected methods.

Undirected methods:

- random action selection
- ϵ greedy
- Softmax (Bolzman etc.).

In general these methods take a local (current-state) view of the exploration problem. Convergence rate is often slow.

Directed Methods:
Here a more global view of the process is taken, and the schemes are directly designed to explore "interesting" parts of the state space. Such schemes are often model-based. For example:

- Least visited state: The number of visits for each state is recorded, and actions (or action-sequences) are selected in order to reach such states.
- Optimistic initialization.
- Upper-confidence bounds (to rewards and value functions).
- Recency-base exploration: Visit states that have not been visited recently (suitable for time-varying problems).

We shall consider some direct schemes in more detail below.

[1] S. Thrun, Efficient Exploration in Reinforcement Learning, Technical Report, 1992.

7.2 On-policy learning with diminishing ϵ-greedy exploration

* This section actually belongs to the previous chapter, on basic convergence results.

In our previous convergence results, it was *assumed* that every state and action are sampled infinitely often. This is easy to ensure in *off-policy* methods, by using any persistent exploration method (e.g., ϵ-greedy). In *on-policy* schemes, if we want to converge to the optimal policy, we must ensure that exploration eventually becomes small. As an illustration, we give here a specific result from Singh et al. (2000)[2].

The considered scheme is SARSA(0) (learning the Q-function of the current policy), with ϵ_t-greedy action selection.

Definition: A learning policy is called GLIE (Greedy in the Limit with Infinite Exploration) if it satisfies the following two properties:

1. If a state is visited infinitely often, then each action in that state is chosen infinitely often (with probability 1).
2. In the limit (as $t \to \infty$), the learning policy is greedy with respect to the learned Q-function (with probability 1).

The first condition requires exploration to proceed indefinitely, while the second requires that its magnitude decays in time. The following (undirected, local) schemes can be shown to be GLIE:

1. ϵ-greedy exploration: Pick a random exploration action with probability ϵ_t, where $\epsilon_t = c/n_t(s)$, $n_t(s)$ is the number of previous visits to the current state $s_t = s$, and $0 < c < 1$.

2. Boltzmann exploration:

$$P(a_t = a | s_t = s, Q_t) = \frac{\exp(\beta_t(s) Q_t(s,a))}{\sum_{b \in A} \exp(\beta_t(s) Q_t(s,b))}$$

where $\beta_t(s) = \log n_t(s)/C_t(s)$, and $C_t(s) \geq \max_{a,a'} |Q_t(s,a) - Q_t(s,a')|$.

[2] S. Singh, T. Jaakkola, M. Littman and C. Czepesvari, "Convergence results for single-step on-policy reinforcement-learning algorithms", *Machine Learning* 38(3), 2000, pp. 287-303.

The algorithm:

a. Q-function estimation using SARSA(0):

$$Q_{t+1}(s,a) = (1 - \alpha_t(s_t, a_t))Q_t(s_t, a_t) + \alpha_t(s_t, a_t)[r_t + \gamma Q_t(s_{t+1}, a_{t+1})]$$

b. Action selection: GLIE with respect to Q_t.

Convergence: Suppose that

1. The usual gain conditions hold for α_t.

2. The controlled Markov chain is *communicating*: every state can be reached from any other with positive probability (under some policy).

3. $\text{var}\{R(s,a)\} < \infty$.

Then, w.p. 1, Q_t converges to Q^*, and the learned policy π_t converges to π^*.

7.3 The Problem with Local Methods: Exponential Convergence Times

Consider the following example, after Whitehead (1992):

– States $\{0, 1, \ldots, N\}$.
– Transitions: $a = 1$: $s' = \min\{s + 1, N\}$. $a = 2$: $s' = 0$.
– Reward: 0 for $s < N$, 1000 for $s = N$.
– Initial conditions: $s_0 = 0$, $Q_0 \equiv 0$.

Suppose we start with 2ϵ-exploration, with $\epsilon > 0$. Then:

– The probability to reach N from 0 in one 'trial' is ϵ^N.
– The expected number of such trials till success is therefore $\left(\frac{1}{\epsilon}\right)^N$.
– This equals the expected number of suboptimal actions ($a = 2$) before ever reaching N.
– Consequently, $P(\text{reach state } N \text{ before time } t) < t\left(\frac{1}{\epsilon}\right)^N$.

It can be seen that the time it takes for significant learning to even start (by reaching the coveted state N for the first time) is *exponential* in the number of states N. This is clearly unacceptable even for moderate N.

7.4 The Multi-Armed Bandit Problem

The MAB problem provides a simplified model which focuses on action selection without the complication of state dynamics. This model may be indeed be considered as a single-state MDP where rewards are unknown[3].

The model: Consider a finite action set A. Given $a_t = a \in A$, the reward distribution is $r_t \sim p(r|a)$, with mean $E(r_t|a) = R(a)$.

$R(a)$ is not known a-priori. The goal is essentially to quickly find the optimal action, $a^* = \arg\max_{a \in A}\{R(a)\}$.

Depending on the learning mode, different criteria may be used for learning efficiency:

- Pure learning: Here the goal is to find a^* as quickly as possible, without regard to the cost of exploration.
- On-line learning: Here we assume that the reward accumulated during the learning phase counts. Hence, we want to minimize the reward lost during learning, in some appropriate sense.

This will be further elaborated in the next section.

We focus here on the on-line case. In the non-Bayesian setting[4], the most common measure of on-line learning performance is the *total regret*:

$$L^\pi(n) = nR(a^*) - E^\pi(\sum_{t=1}^{n} r_t)$$

(considered for large n).

[3]We note that the name MAB also refers to a known model, with each arm having Markovian state dynamics that evolves only when that arm is selected. An optimal solution for the discounted-cost problem in this model is obtained by the celebrated Gittin's index policy. The models are related but of course different.

[4]In the Bayesian setting, where we have a probability distribution on $p(r|a)$, a natural criterion is simply the expected total cost, $E(\sum_{t=1}^{N} r_t)$. This entails the usual difficulties in the Bayesian setting: Finding a meaningful prior, and hard-to-compute solutions.

A related quantity is the *total mistake count*. It is easily seen that

$$L^\pi(n) \leq \Delta_{\max} E^\pi(\sum_{t=1}^{n} 1\{a_t \neq a*\}) \,,$$

where $\Delta_{\max} = \max_a\{R(a^*) - R(a)\}$.

The problem with greedy policies: Let

$$\hat{r}_n(a) = \frac{1}{N_n(a)} \sum_{t=1}^{n} R_t 1\{a_t = a\}$$

denote the average reward (sample mean) obtained for action a up to time n. (Note that this coincides with the Maximum Likelihood Estimator under standard assumptions). Consider the greedy policy:

$$a_n = \arg\max_a \hat{r}_{n-1}(a) \,.$$

Suppose $\hat{r}_n(a^*)$ *underestimates* $R(a^*)$ at some point. Then there is positive probability than a suboptimal action a' will be selected thereafter, leading to $\Theta(n)$ regret with positive probability. This is much worse than what can be obtained otherwise. □

Logarithmic regret: A classical result by Lai & Robbins (1985)[5] asserted that the optimal regret rate is $O(\log n)$. In fact, under parametric assumptions on the reward distributions, they showed that any policy that guaranteed $O(n^\alpha)$ regret for all $\alpha > 0$, must sample any inferior action a at least $\frac{1}{c(a,a^*)} \log n$ times, where $c(a, a^*) = D(p(\cdot|a), p(\cdot|a^*))$ is the Kullback-Leibler divergence[6] (relative entropy) between these reward distributions. Their policy, which achieves the optimal regret asymptotically, is based on the concept of Upper Confidence Bounds (UCBs), and required delicate manipulation of these bounds.

Simple schemes for logarithmic regret: Several papers suggested simplified schemes that obtain $O(\log n)$ regret, without the optimal coefficients. In return these algorithms are simpler, required less prior knowledge of the form of the reward distributions, and can provide explicit finite-time bounds on the regret.

[5] T. Lai and H. Robbins, Asymptotically efficient adaptive allocation rules, *Adv. Appl. Math.* 6, 1985, pp. 4–22.

[6] For discrete distributions, $D(p, q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$.

We briefly describe the UCB1 algorithm from Auer et al. (2002)[7]. Suppose $R_t \in [0,1]$. Let $a_{n+1} = \arg\max_a U_n(a)$, where

$$U_n(a) = \hat{r}_n(a) + \sqrt{\frac{2\log n}{N_n(a)}}\,.$$

Then $L(n) \leq A\log n + B$ for all $n \geq 1$. The constants A and B are given by

$$A = \sum_{a:\Delta_a>0} \frac{8}{\Delta_a}\,, \qquad B = (1+\frac{\pi^2}{3})\sum_a \Delta_a$$

where $\Delta_a = R(a^*) - R(a)$.

The rationale of the above UCB can be understood by recalling the Hoeffding bound. Let (X_i) be independent RVs, with $X_i \in [a_i, b_i]$. Then, for $M > 0$,

$$P\left(\sum_{i=1}^{n}(X_i - E(X_i)) > M\right) \leq \exp\left(-\frac{2M^2}{\sum_{i=1}^{n}(b_i-a_i)^2}\right)\,.$$

From this we obtain

$$P\left(U_n(a) < R(a)\right) \leq \text{...fill in \& explain...}$$

[7]P. Auer, N. Cesa-Bianchi and P. Fischer, Finite-time Analysis of the Multiarmed Bandit Problem, *Machine Learning* 27, 2002, pp. 235-256.

7.5 Measures of Learning Efficiency for MDPs

As mentioned, the criteria for efficient learning depend on learning mode. Several dichotomies can be observed. We follow the terminology of Szepesvari (2010).

- Non-interactive vs. Interactive learning: In non-interactive learning, the learning algorithm observes the process, but cannot affect the choice of actions. This is often done in batch mode. In interactive learning, the learning algorithm can affect some (or all) action choices.
- Active vs. On-line learning: These are different variants of interactive learning. In active learning, the goal is to learn an optimal (or good) policy as quickly as possible, with no regard to the reward accumulated in the process. In on-line learning, we assume the system is alreadyin regular operation mode, so that the obtained reward (or reward loss) counts.

We note that both non-interactive and and active learning can be categorized as 'pure learning', as opposed to on-line learning.

PAC-style criteria for active learning:

a. A basic goal for an pure learning algorithm is:

- After T time units, determine a candidate optimal policy.
  (n can be given a-priori, or decided by the algorithm.)
- Provide bounds on sub-optimality of the policy.

b. PAC bounds have the form: The obtained policy is at most ϵ-suboptimal with probability at least $1 - \delta$.

c. A learning method is considered PAC-efficient (of just PAC) if the required data size, T, is *polynomial* in the problem size and accuracy parameters: $T = \text{poly}(N_S, N_A, \frac{1}{\epsilon}, \frac{1}{\delta})$.

d. We note that the definition of PAC may also require polynomial complexity of computation and space. Here we focus on the *sample complexity* of learning: How many samples are required for efficient learning.

Similar criteria may be formulated for non-interactive learning, but require assumptions on the quality of the observed data.

Criteria for on-line learning:

a. The main issue here is how much reward was lost in the learning process (compared to the optimal policy), or how many mistakes (sub-optimal actions) were made during the learning process.

b. A first criterion is the total regret, similar to the MAB case. Here we can define (for given initial state):

$$L^{\pi}(n) = \max_{\pi'} E^{\pi'}(\sum_{t=1}^{n} r_t) - E^{\pi}(\sum_{t=1}^{n} r_t)$$

or (less accurately)

$$L^{\pi}(n) = nh^* - E^{\pi}(\sum_{t=1}^{n} r_t)$$

where h^* is the optimal long-term average cost.
As in the bandit problem, $O(\log n)$ regret may be ensured. It is also required that the relevant coefficients have polynomial dependence on N_S and N_A.

c. PAC-like criteria: Action error count.
Define the total ϵ-error count as

$$N_{\epsilon}^{\text{err}} = \sum_{t=1}^{\infty} 1\left\{Q(s_t, a_t) \leq \max_a Q(s_t, a) - \epsilon\right\} .$$

We can now define PAC-style bounds on $N_{\epsilon}^{\text{err}}$.

c. PAC-like criteria: Policy error count.
Some papers define a more elaborate error measure. For a given learning algorithm $\mathcal{A}$, let $\mathcal{A}_t$ denote its tail from time t onward given the history till t, and let $V^{\mathcal{A}_t)}(s_t)$ denote the expected return from the current state s_t under this algorithm. Now define

$$\tilde{N}_{\epsilon}^{\text{err}} = \sum_{t=1}^{\infty} 1\left\{V^{\mathcal{A}_t)}(s_t) \leq V^*(s_t) - \epsilon\right\} .$$

The above error count is sometimes referred to as the *sample complexity of exploration*. The two errors are easily seen to satisfy $\tilde{N}_{\epsilon}^{\text{err}} \geq N_{\epsilon}^{\text{err}}$.

Polynomial-time Algorithms: We will briefly discuss the following algorithms for on-line learning in MDPs:

– E^3 (Kearns & Singh, 2002).

– R-max (Brafman & Tennenholtz, 2002).

– MBIE (Strehl & Littman,2005).

– Delayed Q-learning (Strehl et al., 2005).

These algorithms, except for the last one, are *model-based*.

7.6 The E^3 algorithm

M. Kearns and S. Singh, "Near-optimal reinforcement-learning in polynomial time", *Machine Learning* 49, 2002, pp. 209-232.

E^3 (Explicit Explore or Exploit) is essentially the first algorithm that established polynomial sample complexity. (Previous work in the same spirit assumed the ability to reset to a base state at will.)

The algorithm uses distinct periods of *pure exploration* and *pure exploitation*.

The basic algorithm requires the optimal value function to be known. Although this is later relieved by a search procedure, it is not practical. Hence its importance is mainly archival.

Algorithm outline:

- Divide the states into *known* and *unknown* states.
- A state s becomes *known* once all of its actions have been sampled a sufficient number of times (given by a preset threshold m_0). Its reward and transitions probabilities are then updated according to their MLE:

$$\hat{R}(s,a) = \frac{1}{n(s,a)} \sum_{i=1}^{n(s,a)} r_i(s,a)\,, \quad \hat{p}(s'|s,a) = \frac{n(s',s,a)}{n(s,a)}\,, \quad \forall a, s'\,.$$

- When in an *unknown* state: Perform 'balanced wandering' (e.g., choose all actions in round robin fashion).
- When reaching a *known* state from an unknown one:
  - Compute a policy that maximizes return over known states (by setting rewards at unknown states to minimum).
  - If the computed return is high enough compared to the optimal one, apply the computed policy for T steps.
  - Otherwise, compute and use a policy that reaches an unknown state as quickly as possible.

  Basic idea: In the exploitation stage, if we keep to the known model we will get the computed (ϵ-optimal) reward. Else, we reach an unknown state and continue learning.

Remarks:

- The computed exploration or exploitation sub-policies are (non-stationary) T-step optimal policies.
- T is the time required to obtain expected return ϵ-close to its long-term (discounted or average) value.
- In the discounted case, $T \simeq \frac{1}{1-\gamma} \log \frac{R_{\max}}{\epsilon(1-\gamma)}$.
- In the average-return case, T is related to the *mixing time* of the Markov chain.

Main results:

- Discounted return:
  Input to the algorithm: $\epsilon, \delta, N_S, N_A$. Result: After a time that is polynomial in $(\epsilon^{-1}, \delta^{-1}, N_S, N_A, T_\gamma = \frac{1}{1-\gamma}, R_{\max})$, with probability $1-\delta$ at least, the algorithm will halt at some state s with a policy that is ϵ-optimal from s.
- Average return:
  Here, with the above probability, after a polynomial number of steps, the algorithm will obtain an average return that is ϵ-optimal (within all policies with T-mixing time).

7.7 The R-max algorithm: Optimistic Initialization

R. Brafman and M. Tennenholtz, "R-max – a general polynomial time algorithm for near-optimal Reinforcement Learning", *Journal of Machine Learning Research (JMLR)* 3:2113-231, 2002.

Basic idea:

- "Unknown" states-action pairs are assigned best possible reward (long term).
- Unknown state-action pairs become "known" once they have been sampled enough. Their parameters are then set according to the obtained samples.
- Use greedy policy with respect to this optimistic model.

In more detail:

– Pre-compute a threshold m_0.
  Let $R_{\max}$ be an upper bound on the reward expected $r(s,a)$.

– While $n(s,a) < m_0$, set

$$\hat{R} = R_{\max}\,, \quad \hat{p}(s'|s,a) = I_{\{}s' = s\}\,.$$

– Once $n(s,a) = m_0$, set $\hat{R}(s,a)$ and $\hat{p}(\cdot|s,a)$ to their MLEs:

$$\hat{R}(s,a) = \frac{1}{n(s,a)} \sum_{i=1}^{n(s,a)} r_i(s,a)\,, \quad \hat{p}(s'|s,a) = \frac{n(s',s,a)}{n(s,a)}\,, \quad \forall s'\,.$$

– Policy: Compute an optimal actions for the above model (discounted or average return).

Original algorithm and result:

- The original paper considered the average-return case, and applied also to stochastic (Markov) games.
- Optimality is among all policies that have a given ϵ-return mixing time T (i.e., the expected return after T steps is ϵ-close to the long term average). Accordingly, the T-step optimal policy is used for action selection.
- The corresponding result: Given $\epsilon > 0$, $\delta > 0$ and T, there exists an m_0 so that with probability $1-\delta$ at least, the R-max algorithm attains an average return within 2ϵ of the optimum (among all policies with ϵ-return mixing time smaller than T), within a number of steps which is polynomial in $(\epsilon^{-1}, \delta^{-1}, N_S, N_A, T)$.

The algorithm was later shown to be PAC-MDP for the discounted case. The following result is from Strehl, Li & Littman (JMLR, 2009):

- Given $\epsilon > 0$, $\delta > 0$, and $\gamma < 1$, there exists m_0 so that with probability $1-\delta$ at least, the total number of ϵ-suboptimal actions taken is upper bounded (to within logarithmic terms) by

$$O\left(\frac{N_A N_S^2 R_{\max}^2}{\epsilon^3 (1-\gamma)^4}\right)\,.$$

  Note that "suboptimal error" here are in the sense discussed in Section 7.5, i.e., N_ϵ^{err} or $\tilde{N}_\epsilon^{err}$.

7.8 Confidence Intervals: MBIE

A. Strehl and M. Littman, "An Analysis of Model-Based Interval Estimation for Markov Decision Processes ", *Journal of Computer and System Sciences - JCSS*, vol. 74, no. 8, pp. 1309-1331, 2008. (Also in ICML 2005.)

This algorithm uses a similar approach to UCB1 bandit algorithm described above, and provides PAC-MDP bounds for the discounted problem. The idea is to add upper (or worst-case) confidence bounds to the estimated reward and transition probabilities, and back-propagate these confidence bounds through the Dynamic Programming equation.

Basic MBIE Algorithm (outline):

- For each (s, a), compute the MLE estimators $\hat{R}(s, a)$ and $\hat{p}(s'|s, a)$.
- Modify $\hat{R}(s, a)$ to $\tilde{R}(s, a) = \hat{R}(s, a) + \sqrt{\frac{\log(2/\delta_R)}{2n(s,a)}}$.
- Modify $\hat{p}(\cdot|s, a)$ to the set

$$\tilde{P}(s, a) = \{q \in \Delta(S) : \|q - \hat{p}(\cdot|s, a)\|_1 \leq \epsilon_{n(s,a)}\}$$

  where

$$\epsilon_{n(s,a)} = \sqrt{\frac{2N_S - 2\log\delta_p}{n(s, a)}}\,.$$

- Solve the DP equation

$$\tilde{Q}(s, a) = \tilde{R}(s, a) + \max_{q \in \tilde{P}(s,a)} \gamma \sum_{s'} q(s'|s, a) \max_{a'} \tilde{Q}(s', a')$$

  and use the greedy action $a_t = \arg\max_a \tilde{Q}(s_t, a)$.

A simpler version of this algorithm is MBIE-EB (exploration bonus), which solves the DP equation

$$\tilde{Q}(s, a) = \hat{R}(s, a) + \gamma \sum_{s'} \hat{p}(s'|s, a) \max_{a'} \tilde{Q}(s', a') + \frac{\beta}{\sqrt{n(s, a)}}$$

for a given constant $\beta > 0$. Here the confidence bounds on the reward and transition probabilities are combined into one term.

The two algorithms are shown to be PAC-MDP for appropriate choice of confidence parameters (δ_R/δ_p or β).

7.9 Delayed Q-Learning

A. Strehl, L. Li and M. Littman, "Reinforcement learning in finite MDPs: PAC analysis", *Journal of Machine Learning Research*, 10:2413-2444, 2009.

This algorithm is claimed to be "model free". Its space complexity is only $O(N_S N_A)$, compared to $O(N_S^2 N_A)$ of model-based algorithms.

Algorithm Outline:

- $Q(s,a)$ is initialized optimally to $U(s,a) \geq V^*(s)$ ("admission heuristic").
- $Q(s,a)$ is updated only after m samples of (s,a), where m is a predefined (large) constant.
- Each potential update has the form

$$\hat{Q}(s,a) := \frac{1}{m} \sum_{i=1}^{m} Q_{s,a}^{[i]} + \epsilon_1$$

  where $Q_{s,a}^{[i]}$ is computed at the i-th visit to (s,a), using

$$Q_{s,a}^{[i]} = r_{s,a}^{[i]} + \gamma \max_{a'} \hat{Q}^{[i]}(s',a') \,.$$

- A potential update is accepted (performed) only if it *decreases* $\hat{Q}(s,a)$ by more than ϵ_1.
- Policy: greedy actions election w.r.t. $\hat{Q}$.

Sample Complexity:

- In the PAC-MDP setup, with proper choice of m and ϵ_1, the (ϵ, δ)-sample complexity of exploration is bounded by

$$O\left(\frac{\sum_{s,a}(U(s,a) - V^*(s))}{\epsilon^4 (1-\gamma)^7} \log(\frac{1}{\delta}) \log(...) \right) \leq O\left(\frac{N_S N_A}{\epsilon^4 (1-\gamma)^7} \log(...) \right) .$$

Note: The paper also provides a *lower bound* on the sample complexity, which is

$$\Omega\left(\frac{N_S N_A}{\epsilon^2 (1-\gamma)^2} \log \frac{N_S}{\delta} \right) .$$