Reinforcement learning:
Multi-armed bandits

Thomas Bonald
Telecom ParisTech
thomas.bonald@telecom-paristech.fr

2017 – 2018

Reinforcement learning is a branch of machine learning inspired by the way animals (including humans) learn to live and improve themselves by interacting with their environment. Typically, young animals are curious, try a lot of different actions before concentrating on the best ones, following a *trial-and-error* strategy. It is tempting (but far from easy) to let machines learn this way to improve their intelligence and accomplish some specific tasks. The applications are numerous: robotics, medecine, content recommendation, advertising, marketing, chatbots, games, to quote a few.

In this note, we focus on so-called *multi-armed bandits*[1], a class of reinforcement learning problems where an agent must take a sequence of actions while observing the corresponding sequence of rewards. A key feature of multi-armed bandits is that the agent does *not* modify its environment through its actions. Its assigned objective is to discover the best actions, that is, those offering the highest rewards, and to exploit them. In particular, the agent faces the so-called *exploration-exploitation* dilemna: she must typically try all actions to learn what is the best (exploration) but needs to quickly converge to the (believed) best one to accumulate rewards (exploitation).

1 Multi-armed bandits

The model consists of some finite set of actions A (the *arms* of the multi-armed bandit). We denote by $K = |A|$ the number of actions. Each time an action is chosen, some reward $r \in \mathbb{R}$ is received. No information is known about the rewards the other actions would have provided. The successive rewards produced by the same action $a \in A$ are i.i.d. with some unknown distribution $p(\cdot|a)$, that is, the probability that action a gives reward r is $p(r|a)$. We denote by $q(a)$ the corresponding expectation,

$$q(a) = \mathrm{E}(r|a).$$

Learning is done *online*, that is in a sequential way. At time $t = 1, 2, \dots, T$, the agent selects action $a_t \in A$ and receives reward r_t, drawn independently from the distribution $p(\cdot|a_t)$. Observe that the decision at time t depend on the rewards received up to time $t - 1$. The objective is to maximize the cumulative reward,

$$\sum_{t=1}^{T} r_t.$$

The parameter T, called the time horizon, is generally unknown. It may range from 10^2 to 10^6, depending on the application.

[1]The name comes from slot machines, known as *bandits* as they typically take your money! A multi-armed bandit has several arms, each having its own reward distribution. The objective is to find the best one.

If some action a is always selected, the expected reward is $q(a)$ per action, corresponding to an expected cumulative reward of $Tq(a)$. By the law of large numbers, this expected cumulative reward is is close to the actual cumulative rewards when T is large. We denote by $a^\star$ the best action in terms of expected rewards,

$$a^\star = \arg\max_{a\in A} q(a),$$

yielding the expected reward per action

$$q^\star = \max_{a\in A} q(a).$$

Observe that there may be several best actions in general. The objective of bandit algorithms is to learn as quickly as possible the best action(s), so that the cumulative reward is close to $q^\star T$.

2 Performance metrics

The standard way to assess the performance of a bandit algorithm is to compare its cumulative reward to the highest expected cumulative reward, $q^\star T$. Specifically, we define the *regret* of any algorithm as the quantity

$$R = q^\star T - \sum_{t=1}^{T} r_t.$$

Since the regret is stochastic, due to the rewards and possibly the algorithm itself, it proves more convenient to consider the expected regret,

$$\mathrm{E}(R) = q^\star T - \sum_{t=1}^{T} \mathrm{E}(q(a_t)).$$

Let $N_t(a)$ be the number of times action a is selected up to time t. We have:

$$\begin{aligned}
\mathrm{E}(R) &= q^\star T - \sum_{t=1}^{T}\sum_{a\in A} \mathrm{E}(q(a)1_{\{a_t=a\}}),\\
&= q^\star T - \sum_{a\in A} q(a)\sum_{t=1}^{T} \mathrm{E}(1_{\{a_t=a\}}),\\
&= q^\star T - \sum_{a\in A} q(a)\mathrm{E}(N_T(a)),\\
&= \sum_{a\in A}(q^\star - q(a))\mathrm{E}(N_T(a)). \qquad (1)
\end{aligned}$$

Another interesting metric is the *precision* of the algorithm, corresponding to the fraction of time the best action(s) is selected,

$$P = \frac{1}{T}\sum_{t=1}^{T} 1_{\{a_t=a^\star\}},$$

where, with some slight abuse of notation, we write $a_t = a^\star$ to mean $a_t \in a^\star$ in case there are several best actions. Again, it is generally more convenient to consider the expected precision,

$$\mathrm{E}(P) = \frac{\mathrm{E}(N_T(a^\star))}{T} = \frac{1}{T}\sum_{t=1}^{T} P(a_t = a^\star),$$

Efficient algorithms have sublinear regret, meaning that $\mathrm{E}(R)/T \to 0$. In view of (1), this implies that $\mathrm{E}(N_T(a))/T \to 0$ for any suboptimal action. In particular, we get $\mathrm{E}(P) \to 1$ and $P \to 1$: the algorithm converges to the best action(s).

3 Greedy algorithms

The most natural policy consists in always selecting the best known action. This is a pure exploitation strategy. The algorithm is the following, where $N(a)$ is the number of tries of action a and $Q(a)$ the estimate of the expected reward $q(a)$ of action a:

```
Greedy algorithm

Initialize, for all actions $a$:
    • $N(a) \leftarrow 0$
    • $Q(a) \leftarrow 0$
Repeat:
    • $a \leftarrow \arg\max_a Q(a)$ (random tie breaking)
    • $r \leftarrow \text{reward}(a)$
    • $N(a) \leftarrow N(a) + 1$
    • $Q(a) \leftarrow Q(a) + \frac{1}{N(a)}(r - Q(a))$
```

The initial values of the estimates $Q(a)$ of the expected rewards play a key role. Assume binary rewards for instance. For initial values set to 0 (as above), the first action a yielding a reward 1 is selected for ever (there is no exploration). For initial values set to 1, the algorithm will explore all actions before concentrating on the best ones, which looks like a better strategy. More generally, assuming some prior knowledge on the reward distributions (like their support), it is preferable to choose *optimistic* initial values so as to favor exploration.

One of the most popular bandit algorithms is a slight modification of the greedy policy, where exploration is forced with some fixed (low) probability ε:

```
$\varepsilon$-greedy algorithm

Parameter: $\varepsilon$
Initialize, for all actions $a$:
    • $N(a) \leftarrow 0$
    • $Q(a) \leftarrow 0$
Repeat:
    • $a \leftarrow \begin{cases} \text{random action} & \text{with probability } \varepsilon \\ \arg\max_a Q(a) & \text{with probability } 1 - \varepsilon \end{cases}$
    • $r \leftarrow \text{reward}(a)$
    • $N(a) \leftarrow N(a) + 1$
    • $Q(a) \leftarrow Q(a) + \frac{1}{N(a)}(r - Q(a))$
```

Clearly, the positive probability of exploration prevents this algorithm from achieving a sublinear regret. Indeed, its expected precision cannot exceed

$$1 - \varepsilon \frac{K - K^\star}{K},$$

where $K^\star$ is the number of best actions. An option is to let the parameter ε decrease with time:

```
Adaptive-greedy algorithm

Parameter: $c$
Initialize, for all actions $a$:
    • $N(a) \leftarrow 0$
    • $Q(a) \leftarrow 0$
Repeat for $t = 1, 2, \dots$
    • $\varepsilon \leftarrow \frac{c}{c+t}$
    • $a \leftarrow \begin{cases} \text{random action} & \text{with probability } \varepsilon \\ \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \end{cases}$
    • $r \leftarrow \text{reward}(a)$
    • $N(a) \leftarrow N(a) + 1$
    • $Q(a) \leftarrow Q(a) + \frac{1}{N(a)}(r - Q(a))$
```

The degree of exploration is controlled by the parameter c (the higher c, the more explorative the algorithm). This algorithm is provably efficient for a sufficiently high value of the parameter c (but with a high exploration cost).

4 Upper confidence bound

The most popular bandit algorithm is based on the principle of optimism in face of uncertainty. Specifically, a bonus that quantifies the uncertainty about the reward distribution is given to each action. The best action, in terms of average reward plus bonus, is selected.

```
UCB algorithm

Parameter: $c$
Initialize, for all actions $a$:
    • $N(a) \leftarrow 0$
    • $Q(a) \leftarrow 0$
Repeat for $t = 1, 2, \dots$
    • $a \leftarrow \arg\max_a (Q(a) + c\sqrt{\frac{\log t}{N(a)}})$
    • $r \leftarrow \text{reward}(a)$
    • $N(a) \leftarrow N(a) + 1$
    • $Q(a) \leftarrow Q(a) + \frac{1}{N(a)}(r - Q(a))$
```

Observe that all actions are selected once at the beginning (since $N(a) = 0$ implies infinite bonus). The particular form of the bonus comes from Hoeffding's inequality, which provides an upper bound on the probability of incorrect estimation of the expectation of a bounded random variable with respect to the number of samples [Hoeffding, 1963]. Again, the parameter c controls the exploration-exploitation trade-off (the higher c, the more explorative the algorithm).

It turns out that the UCB algorithm has a sublinear regret. For binary rewards, we have [Auer et al., 2002a]

$$\mathrm{E}(R) \leq 8 \sum_{a \neq a^\star} \frac{\log T}{q^\star - q(a)} + K\frac{\pi^2}{3}.$$

This upper bound suggests that quasi-optimal actions ($q(a) \approx q^\star$) incur the highest regret.

5 Thompson sampling

The second most popular algorithm is also one of the oldest. This is a Bayesian algorithm proposed by Thompson in 1933 in a medical context (online selection of the best treatment) [Thompson, 1933]. The uncertainty in the expected rewards is now captured by probability distributions, one per action. These probability distributions tend to concentrate when more information is obtained from each action. Again, the initial probability distribution of each action, known as the prior, plays a key role. In brief, exploration is guaranteed by the prior, while exploitation is enforced by the concentration of the posterior.

We denote by $P(a)$ the probability distribution associated with the estimation of $q(a)$, the expected reward of action a. It is initialized with some prior (see examples below) and updated by the successive rewards received from action a.

```
Thompson sampling

Initialize, for all actions $a$:
    • $P(a) \leftarrow$ prior
Repeat:
    • for all actions $a$, $Q(a) \leftarrow$ sample($P(a)$)
    • $a \leftarrow \arg\max_a Q(a)$
    • $r \leftarrow$ reward($a$)
    • $P(a) \leftarrow$ update($r$)
```

The main drawback of Thompson sampling is the computational cost of samples and updates. Typically, the prior is chosen so as to facilitate these computations.

For binary rewards, the usual prior is the uniform distribution over $(0,1)$. Observing that

$$p(r|q) = q^r(1-q)^{1-r}, \quad r = 0, 1,$$

it follows from Bayes' rule that for N i.i.d. reward samples $r_1, \dots, r_N$ of the same action,

$$p(q|r_1, \dots, r_N) = \frac{p(r_1, \dots, r_N|q)p(q)}{p(r_1, \dots, r_N)} \propto q^{r_1+\ldots+r_N}(1-q)^{N-(r_1+\ldots+r_N)}.$$

This is a Beta distribution with parameters $\alpha = r_1 + \ldots + r_N + 1$, $\beta = N - (r_1 + \ldots + r_N) + 1$. We obtain the following simple update rule, given the received reward r:

$$\begin{aligned} \alpha &\leftarrow \alpha + r, \\ \beta &\leftarrow \beta + 1 - r. \end{aligned}$$

For continuous rewards, the usual prior is the standard normal distribution,

$$p(q) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}q^2}.$$

Assuming that the rewards themselves are drawn from a normal distribution with unit variance,

$$p(r|q) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(r-q)^2},$$

we obtain for N samples $r_1, \dots, r_N$ of the same action,

$$p(q|r_1, \dots, r_N) = \frac{p(r_1, \dots, r_N|q)p(q)}{p(r_1, \dots, r_N)} \propto e^{-\frac{N+1}{2}\left(q - \frac{1}{N+1}(r_1+\ldots+r_N)\right)^2}.$$

This is a normal distribution with mean $\mu = \frac{1}{N+1}(r_1 + \ldots + r_N)$ and variance $\sigma^2 = \frac{1}{N+1}$. We obtain the following update rule, given the received reward r at the N-th trial:

$$\begin{aligned} \mu &\leftarrow \mu + \frac{1}{N+1}(r - \mu), \\ \sigma^2 &\leftarrow \frac{1}{N+1}. \end{aligned}$$

Although Thompson sampling yields remarkable performance results in practice and has long been considered as optimal, in a sense that is explained in Section 6 below, this has only been proved recently [Kaufmann et al., 2012]. For binary rewards, we have for any $\varepsilon > 0$,

$$\mathrm{E}(R) \leq (1+\varepsilon) \sum_{a \neq a^\star} \frac{q^\star - q(a)}{D(q(a)||q^\star)} (\log T + \log\log T) + C(\varepsilon),$$

where $D(p||q)$ is the Kullback-Leibler divergence between two Bernoulli distributions with respective parameters p and q:

$$D(p||q) = p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q},$$

and $C(\varepsilon)$ is a constant depending on ε and the parameters $q(a), a \in A$. Since $D(p||q)) = O((p-q)^2)$ when $q \to p$, this suggests again that quasi-optimal actions incur the highest regret.

6 Lower bound

While both UCB and Thompson sampling have sublinear regrets, one may wonder whether these algorithms are optimal in some sense (for instance, there may well exist algorithms with *finite* regret). Lai and Robbins have provided a lower bound that is valid for *any* algorithm with sublinear regret [Lai and Robbins, 1985]. For binary rewards, we have for any suboptimal action $a \neq a^\star$,

$$\liminf_{T \to +\infty} \frac{N_T(a)}{\log T} \geq \frac{1}{D(q(a)||q^\star)}.$$

In particular,

$$\liminf_{T \to +\infty} \frac{\mathrm{E}(R)}{\log T} \geq \sum_{a \neq a^\star} \frac{q^\star - q(a)}{D(q(a)||q^\star)}.$$

We can check that Thompson sampling is optimal in this sense. This is not the case of UCB but of a slight adpatation of it known as KL-UCB [Cappé et al., 2013]. Note that this bound quantifies the *asymptotic* behavior of the algorithm. It says nothing about the performance of the algorithm in the practically interesting case of finite time horizons, although provably optimal algorithms like KL-UCB and Thompson sampling also tend to perform well over finite time horizons in practice.

7 Extensions

A number of extensions of the classical model considered so far have been introduced to cope with the diversity of applications of bandit algorithms.

Combinatorial bandits. In many applications (e.g., content recommendation), an action consists in choosing k elements in a set of size n, corresponding to $\binom{n}{k}$ possible actions. For large values of n (say $n > 100$), it is simply not feasible to try all actions. Moreover, the rewards are clearly not independent across actions (for instance, two actions with $k-1$ common elements are expected to give similar rewards). Finding good practical algorithms in this setting is a topic of intense research. Most existing algorithms are adaptations of UCB and differ with respect to the assumed feedback received by the agent.

Contextual bandits. Another interesting class of problems associates some context to each decision (e.g., information on a user to improve the recommendation). Formally, the action a_t now depends on some known state s_t (the context) and provides a reward r_t that depends on both a_t and s_t. In *linear* bandits, a type of contextual bandits, the state space may be $\mathbb{R}^d$ and the set of actions some finite subset of $\mathbb{R}^d$, so that the reward of action a in state s is $a^T s$, plus some random noise. The objective is then to choose an action a that is the most aligned with the current state s. The most popular algorithm, called LinUCB, combines linear regression (to estimate each parameter a) and the UCB policy [Li et al., 2010]

Adversarial bandits. When the statistics of the rewards are unknown or vary with time, we need more robust algorithms. In the adversarial setting, the rewards of each action consist of arbitrary (still unknown) sequences. Surprisingly, learning is still feasible and may still lead to a sublinear expected regret, for some suitable definition of regret. More specifically, the regret is measured with respect to the best action, which depends on the particular sequence of rewards and the time horizon T. To assess the performance of any given policy, we consider the *worst-case* regret, corresponding to the sequence of rewards yielding the largest regret. This forces the algorithm to be stochastic so as to cope with all possible sequences of rewards. The most popular policy is the Exponential-weight algorithm for Exploration and Exploitation (Exp3). The idea is to associate some positive weight to each action and to randomly select an action in proportion to these weights; the weight of the selected action is then updated depending on the received reward. The Exp3 algorithm has a regret in $O(\sqrt{T})$, which is provably order-optimal [Auer et al., 2002b].

8 Further reading

- **Reinforcement learning.** See [Sutton and Barto, 1998] and the course of Silver (UCL).
- **Multi-armed bandits.** See `https://sites.google.com/site/banditstutorial/`
- **Thompson sampling.** See [Russo et al., 2017].

References

[Auer et al., 2002a] Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002a). Finite-time analysis of the multi-armed bandit problem. *Mach. Learn.*, 47(2-3):235–256.

[Auer et al., 2002b] Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002b). The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77.

[Cappé et al., 2013] Cappé, O., Garivier, A., Maillard, O.-A., Munos, R., and Stoltz, G. (2013). Kullback-Leibler upper confidence bounds for optimal sequential allocation. *To appear in Annals of Statistics.*

[Hoeffding, 1963] Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30.

[Kaufmann et al., 2012] Kaufmann, E., Korda, N., and Munos, R. (2012). Thompson sampling: An asymptotically optimal finite-time analysis. In *Algorithmic Learning Theory*, pages 199–213. Springer.

[Lai and Robbins, 1985] Lai, T. L. and Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22.

[Li et al., 2010] Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM.

[Russo et al., 2017] Russo, D., Van Roy, B., Kazerouni, A., and Osband, I. (2017). A tutorial on Thompson sampling. *arXiv preprint arXiv:1707.02038.*

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction.* MIT press Cambridge. Available online, see http://incompleteideas.net.

[Thompson, 1933] Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25:285–294.