Reward Bound Estimation in Multi-armed Bandit

YILEI WANG, The University of Hong Kong

Algorithms based on upper-confidence bounds are very popular in the multi-armed bandit problem. However, most of these algorithms assume that the rewards come from a bounded support, say $[0, b]$, in which b is known by the gambler initially. This report gives an algorithm analysis based on UCB1 algorithm and considers the case that the rewards are bounded in $[a, b]$ where a and b is unknown to the gambler, and then use this algorithm to solve an online routing problem.

General Terms: Reward Bound, Bandit, UCB, Multi-armed

1. INTRODUCTION AND NOTATIONS

The multi-armed bandit problem is a problem in which a gambler at a row of K slot machines, whose rewards are specified by K unknown but fixed distributions. At each round, the gambler has to decide which machines to play. When played, each machine (also called arm) provides a random reward according to its distribution. The objective of the gambler is to maximize the sum of rewards earned after T rounds. We assume $T \gg K$ here.

The key to solve this problem is to balance exploration and exploitation. On the one hand, we want to make sure which machine gives the best reward (exploration). On the other hand, we want to have more chance to play the best arm so as to get more rewards (exploitation). There are lots of research on this problem. Among these research, **UCB** algorithms are very popular because they are simple and effective. These algorithms works by computing upper confidence bounds for all the arms and then choosing the arm with the highest such bound. Our result is based on **UCB1**, the simplest UCB algorithm.

We denote the reward received when the k-th arm is pulled the t-th time by $X_{k,t}$. $X_{k,1}, \dots, X_{k,T}$ are independent and identically distributed, simply denoted by X_k, and its expected value denoted by μ_k. Further, let $n_k(t)$ denote the number of times arm k is chosen during the first t plays and let I_t denote the index of the arm played at time t. If we have known the X_{k^*} has the greatest expected value μ^*, we would always play this arm. Therefore, we define **Regret** to judge an algorithm as

$$R_T = \sum_{t=1}^{T} X_{k^*,t} - \sum_{t=1}^{T} X_{I_t, n_{I_t(t)}}.$$

We say an algorithm is good means that it runs with small regret. Our goal is to minimize the **Expected Regret**

$$\mathbb{E}[R_T] = \sum_{k=1}^{K} \mathbb{E}[n_k(T)]\Delta_k$$

where $\Delta_k = \mu^* - \mu_k$ is the expected loss of arm k.

2. MAIN RESULT

Our result shows that for some UCB strategy, the knowledge about the bound of the reward distributions is not necessary.

LEMMA 2.1. *Suppose $X_1, \dots, X_n$ and X are independent and identically distributed in $[a, b]$. Let $A_n = \min\{X_1, \dots, X_n\}$ and $B_n = \max\{X_1, \dots, X_n\}$. If for any $e \in (a, b)$,*

$\mathbb{P}(X > e) < 1$ *and* $\mathbb{P}(X < e) < 1$*, then*

$$\lim_{n\to\infty} \mathbb{E}[A_n] = a$$
$$\lim_{n\to\infty} \mathbb{E}[B_n] = b$$

PROOF. Denote F the cumulative distribution function of X, then the cumulative distribution function of B_n is

$$\begin{aligned} F_{B_n}(x) &= \mathbb{P}(B_n \le x) \\ &= \prod_{i=1}^{n} \mathbb{P}(X_i \le x) \\ &= F^n(x). \end{aligned}$$

Hence the expected value of B_n is

$$\begin{aligned} \mathbb{E}[B_n] &= \int_{-\infty}^{\infty} x \, \mathrm{d}F^n(x) \\ &= b \cdot F^n(b) - a \cdot F^n(a) - \int_a^b F^n(x) \, \mathrm{d}x \end{aligned}$$

The second equation holds because of the representations as **Riemann-Stieltjes** integral and integration by parts (see [Wikipedia, 2016]). Note that $|F(x)| \le 1$, $F(b) = 1$ and $0 \le F(a) < 1$, we have

$$\begin{aligned} \lim_{n\to\infty} \mathbb{E}[B_n] &= b - \lim_{n\to\infty} \int_a^b F^n(x) \, \mathrm{d}x \\ &= b - \int_a^b \lim_{n\to\infty} F^n(x) \, \mathrm{d}x \\ &= b \end{aligned}$$

according to **dominated convergence theorem**. Similarly, we get $\lim_{n\to\infty} \mathbb{E}[A_n] = a$. $\square$

Remark 2.2. The speed that $\mathbb{E}[A_n - B_n]$ converge to $a - b$ depends on the distribution of X. Let's consider random variable X with support in $[0, 1]$ with $F(x) = x^\alpha (\alpha > 0)$. Use the equation above, we get $\mathbb{E}[B_n(x)] = n\alpha/(n\alpha + 1)$. When α goes to 0, the speed that this value goes to 1 can be arbitrarily slow.

THEOREM 2.3. *If Algorithm 1 is run on K machines having i.i.d reward distributions $X_1, \ldots, X_K$ with support in $[a, b]$, then its expected regret after T of plays is at most*

$$[8(b-a)^2 \sum_{i:\mu_i<\mu^*} \frac{\ln T}{\Delta_i}] + (18 + C)(\sum_{j=1}^{K} \Delta_j) \tag{1}$$

where $C = \min_t\{t \in \mathbb{N} : \int_a^b [F^t(x) + (1-F(x))^t] \, \mathrm{d}x \le (b-a)/4\}$ and $F(x)$ is the cumulative distribution function of each arm.

PROOF. The proof of this is similar to the proof of Theorem 1 in [Auer et al., 2002]. Notice that

$$\zeta(\frac{9}{8}) = \sum_{n=1}^{\infty} n^{-9/8} \approx 8.6 < 9$$

ALGORITHM 1: UCB1 with reward bound estimation (**UCBRBE**)

```
Input: $K$ slot machines with fixed reward distribution
for $t = 1, \dots, K$ do
    play the $t$-th arm and get reward $p$
    set $S_t = p$, $n_t = 1$
end
set $a = \min\{S_1, \dots, S_K\}$, $b = \max\{S_1, \dots, S_K\}$
for $t = K+1, \dots, T$ do
    set $m = \arg\max_{1 \le j \le K}\{S_j/n_j + (b-a)\sqrt{2\ln t/n_j}\}$
    play the $m$-th arm and get reward $p$
    set $S_m = S_m + p$, $n_m = n_m + 1$
    if $p < a$ then
        set $a = p$
    end
    if $p > b$ then
        set $b = p$
    end
end
```

and the lemma above, we conclude this result. □

We call Algorithm 1 **UCBRBE** (Reward Bound Estimation). The only difference between **UCB1** and **UCBRBE** is that, the former uses the exact difference between upper bound b and lower bound a as the coefficient of the bias $\sqrt{2\ln t/n_j}$, while the latter uses estimate bound.

3. EXAMPLES

3.1. Distributions and Regret

For common distributions, the constant C in Equation 1 is not large.

— For uniformly distributed random variable, $C = 7$.
— For beta distributed random variable with parameter $\alpha = \beta = 0.5$, $C = 5$.
— For binomial distributed random variable with parameter p and n, we have

$$C < -\frac{\ln 8}{\ln \max\{1-p^n, p^n\}}.$$

These results show that the regret bound is also $O(\ln T)$ and mostly would not be much larger than having known the reward bound initially.

In our experiment, we try 3 different distributions and compare the regret of **UCB1** and **UCBRBE**, as Figure 1 shows, where **Naive** is an algorithm that we always play the arm with the maximum mean reward on the historic data.

3.2. Reward Bound and Regret Bound

To show running **UCB1** with haphazardly selected reward bound is not a good choice, we've done some other experiments.

Denote $R(\alpha)$ the regret if we believe $\alpha(b-a)$ is the difference between upper bound and lower bound, i.e., we use $\alpha(b-a)\sqrt{2\ln t/n_j}$ as bias in **UCB1**. What's more, denote R^* as the regret of using Algorithm 1. The Table I shows that how this different estimate would influence the regret bound.

In Table I, we know that if α deviate 1 too much (see $R(0.1)$ and $R(10)$), the regret can be very large. We also find that our algorithm **UCBRBE** runs as good as **UCB1**.

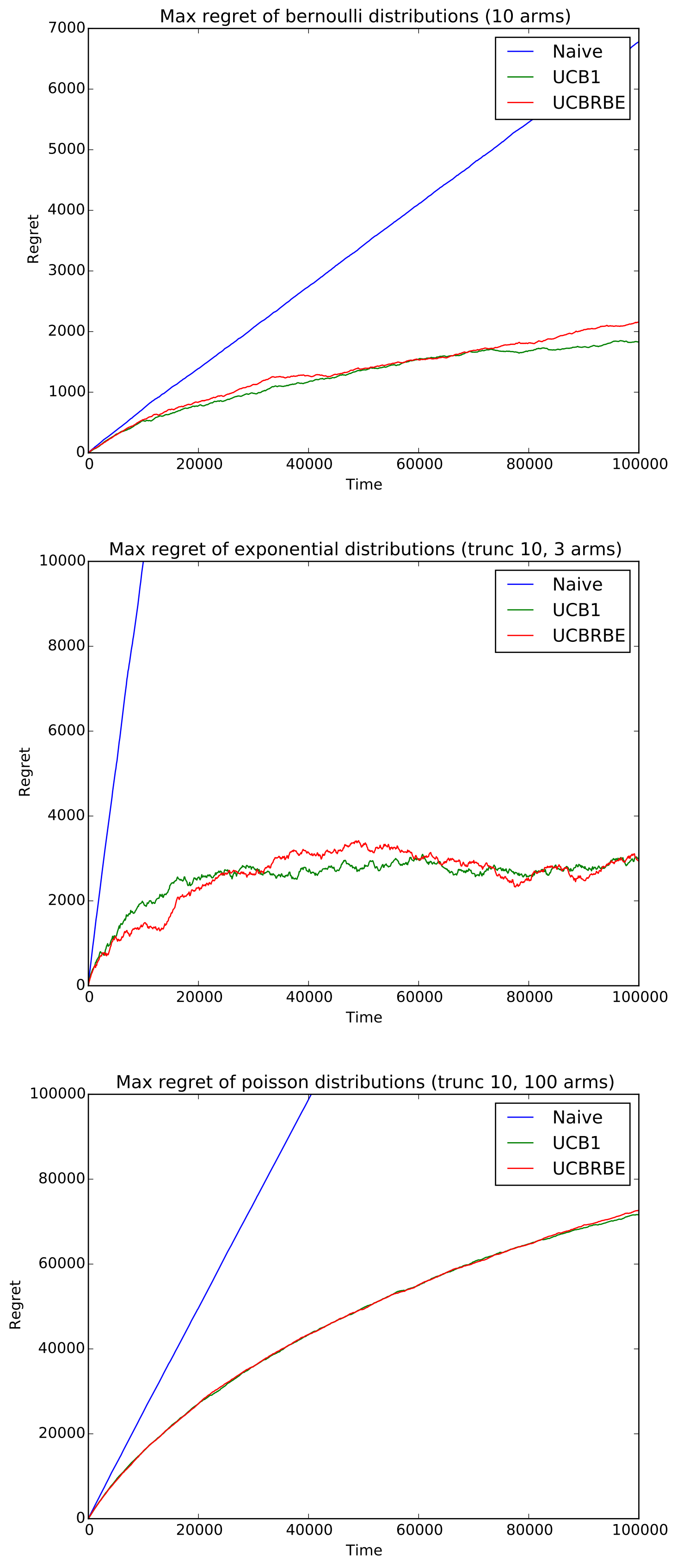

Fig. 1. The relationship between time and regret in different distributions

Table I. Regret Bound and Reward Bound in UCB1

Distribution	***K***	***T***	***R*(0.1)**	***R****	***R*(1)**	***R*(10)**
Beta 1	4	25000	5318	164	171	3952
Beta 2	5	20000	3822	429	429	1981
Beta 3	6	16666	4056	304	304	3433
Bernoulli 1	4	25000	8375	156	156	4192
Bernoulli 2	8	12500	453	340	340	3445
Bernoulli 3	10	10000	1019	459	459	2672

Note: For 100 times we shuffle each arm's rewards and run the algorithm to get regret. The regret in this table is the maximal one.

4. APPLICATION

Now, let's come to the routing problem as follows. We want to send big data, e.g., T MB, from computer S to computer D. Suppose we have K cloud servers, denoted by $C_1, C_2, \dots, C_K$. Our strategy is to choose one as intermediate node, i.e., we send the data to a server, and then the server send these data to D. As Figure 2 shows, we choose C_2 as intermediate node for example.

Since the path structure and network state between S and C_i and between C_i and D are very complex, we could not predict which server is our best choice. Therefore, we describe our problem as online learning problem. We "learn" which server is the best choice by sending data and get feedback. That's what Algorithm 2 means.

ALGORITHM 2: Online Routing Algorithm

```
Input: $K$ cloud servers
while there are data to be sent do
    pick $k \in \{1, 2, \dots, K\}$
    send data to $C_k$ for a while $\Delta t$
    get the transmission speed as feedback
end
```

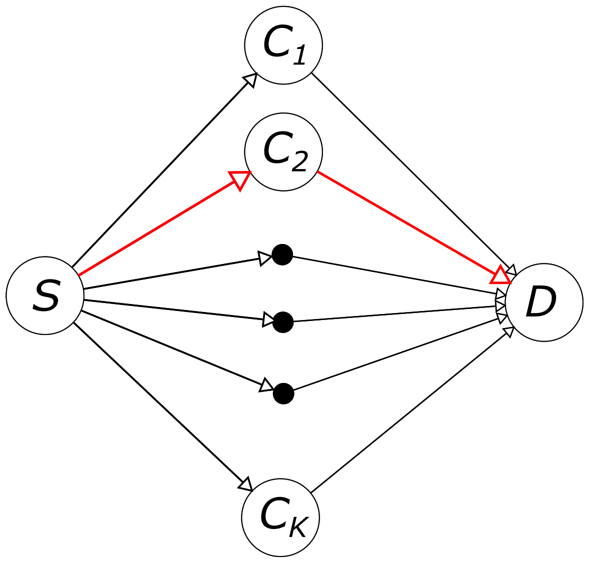

Fig. 2. Online Routing Problem

Notice that the online routing problem is exactly the multi-armed problem with unknown reward bound. In our experiment, we have 3 different servers in Canada and America. Using Algorithm 1, each second we send data to a server and record the transmission speed (reward), as Figure 3 shows, where **e-Greedy** is an algorithm that with probability e we choose a random arm and with probability $1 - e$ we choose the arm with the maximum mean reward on the historic data. (In Figure 3, $e = 0.01$.)

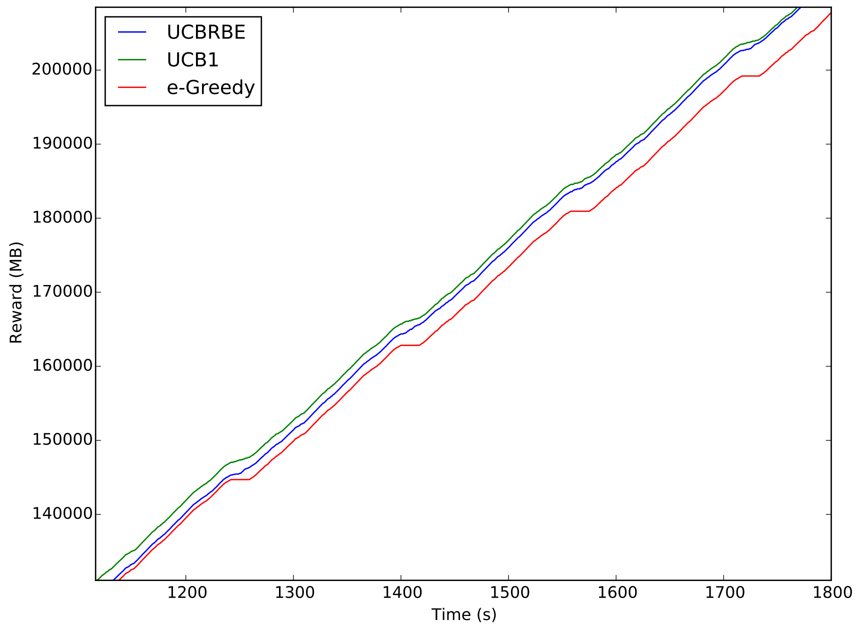

Fig. 3. The relationship between time and rewards in online routing problem

5. EXTENSION

Though without detailed analysis like **UCB1**, we believe that all of UCB algorithms can run with reward bound unknown, using estimate reward bound as the bound in the bias.

In Table II, we test the same data in TableI with **UCB-V** (see [Audibert et al., 2007]). It's regret bound is

$$\mathbb{E}[R_T] \leq 10 \sum_{k:\mu_k<\mu^*} [\frac{\sigma_k^2}{\Delta_k} + 2(b-a)] \ln T.$$

The main advantage of **UCB-V** is that the regret bound's rely on $(b-a)^2$ is decreased to $(b-a)$. Therefore, $R(10)$ not that large compared with **UCB1**. We see **UCBRBE** based on **UCB-V** works as well as **UCB-V**.

ACKNOWLEDGMENTS

The authors would like to thank Dr. C. Wu for providing guidance and advice.

Table II. Regret Bound and Reward Bound in UCB-V

Distribution	K	T	$R(0.1)$	R^*	$R(1)$	$R(10)$
Beta 1	4	25000	1985	61	61	220
Beta 2	5	20000	1785	165	165	398
Beta 3	6	16666	3170	85	85	361
Bernoulli 1	4	25000	9619	66	73	243
Bernoulli 2	8	12500	498	143	143	509
Bernoulli 3	10	10000	710	193	193	648

Note: The rewards in this table are the same with Table I while the algorithm is different.

REFERENCES

Audibert, J.-Y., Munos, R., and Szepesvári, C. (2007). Variance estimates and exploration function in multi-armed bandit. In *CERTIS Research Report 07–31*. Citeseer.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

Wikipedia (2016). Riemannstieltjes integral — wikipedia, the free encyclopedia. [Online; accessed 26-August-2016].