Approximately Optimal Arm Identification

Sahasrajit Sarmasarkar Divyam Bapna Adwait Godbole

Group 10

October 2019

The Setting

A Multi Armed Bandit instance with n Bernoulli bandits. The reward probabilities are p_i, where WLOG $p_1 \geq p_2 \geq \cdots \geq p_n$.

The Setting

A Multi Armed Bandit instance with n Bernoulli bandits. The reward probabilities are p_i, where WLOG $p_1 \geq p_2 \geq \cdots \geq p_n$.

(ϵ, m)-optimal arm

An arm j is said to be ϵ, m-optimal if $p_m - p_j \leq \epsilon$.

The Setting

A Multi Armed Bandit instance with n Bernoulli bandits. The reward probabilities are p_i, where WLOG $p_1 \geq p_2 \geq \cdots \geq p_n$.

(ϵ, m)-optimal arm

An arm j is said to be ϵ, m-optimal if $p_m - p_j \leq \epsilon$.

Intuitively, ϵ is the tolerance in returning sub-optimal arms

Problem Statement

Given a bandit instance, the objective is to return a set of m, (ϵ, m)-optimal arms.

Considering the PAC framework, we want an algorithm that, with probability atleast $1 - \delta$, terminates and returns m such arms.

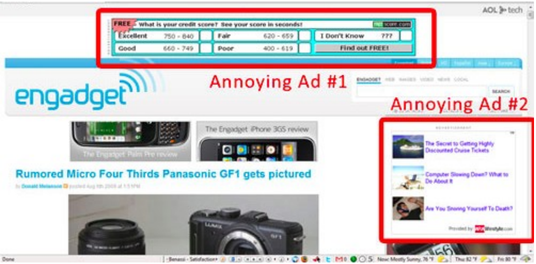
Applications

Webpage Advertisement Problem

A website owner has n options of advertisements but can only put up an m sized subset. His objective is to identify the m most likely to bring in more revenue.

<u>Assumption:</u> Viewership is roughly constant.

$\Rightarrow$ In this scenario, we model the quantized up-time as pulls and clicks as reward 1.



Preliminary Approaches

Kalyanakrishnan, et. al. in [2] suggest some approaches towards algorithms for this problem.

The intuitive idea is that if an arm is sampled sufficiently large number of times, the confidence bounds on that arm is sufficiently tight and hence, with large probability (ϵ, m)-optimal arms will be returned.

We will briefly look at these.

Preliminary Approaches

Direct Algorithm

Each arm is sampled a sufficiently large number of times - $O(\frac{2}{\epsilon^2}\log(\frac{n}{\delta}))$ times. Then the empirically the best m arms are selected.

Preliminary Approaches

Direct Algorithm

Each arm is sampled a sufficiently large number of times - $O(\frac{2}{\epsilon^2}\log(\frac{n}{\delta}))$ times. Then the empirically the best m arms are selected.

Incremental Algorithm

One by one select $(\epsilon, 1)$-optimal arms m-times and accumulate them into a set, which is returned. This has complexity $O(\frac{mn}{\epsilon^2}\log(\frac{m}{\delta}))$

Preliminary Approaches

Direct Algorithm

Each arm is sampled a sufficiently large number of times - $O(\frac{2}{\epsilon^2}\log(\frac{n}{\delta}))$ times. Then the empirically the best m arms are selected.

Incremental Algorithm

One by one select $(\epsilon, 1)$-optimal arms m-times and accumulate them into a set, which is returned. This has complexity $O(\frac{mn}{\epsilon^2}\log(\frac{m}{\delta}))$

Halving Algorithm

The main idea of Halving is to perform binary search on the sequence of arms. This brings down sample to $O(\frac{m}{\epsilon^2}\log(\frac{m}{\delta}))$.

General Algorithmic Framework

Algorithm 1: A general algorithm

Result: m (ϵ, m)-optimal arms

initialize *history*;

while $!\mathtt{shouldStop}(history)$ **do**

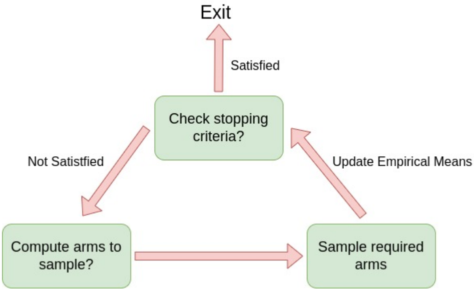
    selectedArms = `selectArms`(history) ;

    result = `sample`(*selectedArms*) ;

    `update`(*history*, *result*) ;

end

return `topK`(*history*)



LUCB1

Sampling Strategy

At every pull, we maintain the sets of 'top' (m) and 'bottom' ($n - m$) arms. The paper discusses a greedy sampling strategy which is quite intuitive. According to this, we sample the arms which are more likely to have been misclassified.

LUCB1

Sampling Strategy

At every pull, we maintain the sets of ‘top’ (m) and ‘bottom’ ($n-m$) arms. The paper discusses a greedy sampling strategy which is quite intuitive. According to this, we sample the arms which are more likely to have been misclassified.

`selectArms`

$$h_*^t = \underset{h \in Top^t}{\text{argmin}} \{\hat{p}_h^t - \beta(u_h, t)\}$$

$$l_*^t = \underset{l \in Bottom^t}{\text{argmax}} \{\hat{p}_l^t + \beta(u_l, t)\}$$

Sampling Strategy

At every pull, we maintain the sets of 'top' (m) and 'bottom' ($n - m$) arms. The paper discusses a greedy sampling strategy which is quite intuitive. According to this, we sample the arms which are more likely to have been misclassified.

`selectArms`

$$h_*^t = \underset{h \in Top^t}{\text{argmin}} \{\hat{p}_h^t - \beta(u_h, t)\}$$
$$l_*^t = \underset{l \in Bottom^t}{\text{argmax}} \{\hat{p}_l^t + \beta(u_l, t)\}$$

Here $\beta(u, t)$ is a function of the number of pulls u and total pulls t.

$$\beta(u, t) = \sqrt{\frac{1}{2u} \log(\frac{k_1 n t^4}{\delta})}, \text{ where } k_1 = \frac{5}{4}$$

Stopping Criteria

After sufficient number of pulls, if the algorithm has gained at least $1 - \delta$ confidence over the correctness of the 'top' set, it stops.

LUCB1

Stopping Criteria

After sufficient number of pulls, if the algorithm has gained at least $1 - \delta$ confidence over the correctness of the 'top' set, it stops.

This is captured by the inequality:

shouldStop

$$\{\hat{p}^t_{l^t_*} + \beta(u^t_{l^t_*}, t)\} - \{\hat{p}^t_{h^t_*} - \beta(u^t_{h^t_*}, t)\} < \epsilon$$

LUCB1

Stopping Criteria

After sufficient number of pulls, if the algorithm has gained at least $1 - \delta$ confidence over the correctness of the 'top' set, it stops.

This is captured by the inequality:

shouldStop

$$\{\hat{p}^t_{l^t_*} + \beta(u^t_{l^t_*}, t)\} \text{ - } \{\hat{p}^t_{h^t_*} \text{ - } \beta(u^t_{h^t_*}, t)\} < \epsilon$$

The sample complexity of this algorithm is $\mathcal{O}(H^{\epsilon/2} log(\frac{H^{\epsilon/2}}{\delta}))$, where $H^{\epsilon/2} = \sum_{a \in arms} \frac{1}{[\Delta_a \vee \frac{\epsilon}{2}]^2}$.

LUCB2 (Modified LUCB1)

What have we done?

We worked on the theorems in (Kalyanakrishnan, et. al. 2012 [3]) to improve the bounds and came up with the following strategy.

LUCB2 (Modified LUCB1)

What have we done?

We worked on the theorems in (Kalyanakrishnan, et. al. 2012 [3]) to improve the bounds and came up with the following strategy.

- We select a t_0 before which the algorithm would never stop even if the previously mentioned inequality is satisfied.

LUCB2 (Modified LUCB1)

What have we done?

We worked on the theorems in (Kalyanakrishnan, et. al. 2012 [3]) to improve the bounds and came up with the following strategy.

- We select a t_0 before which the algorithm would never stop even if the previously mentioned inequality is satisfied.
- We replace δ in the expression of $\beta(u, t)$ by $\tilde{\delta}$.

LUCB2 (Modified LUCB1)

What have we done?

We worked on the theorems in (Kalyanakrishnan, et. al. 2012 [3]) to improve the bounds and came up with the following strategy.

- We select a t_0 before which the algorithm would never stop even if the previously mentioned inequality is satisfied.
- We replace δ in the expression of $\beta(u, t)$ by $\tilde{\delta}$.

$\tilde{\delta}$ should thus be the largest possible value that still would give the PAC guarantees. Intuitively, this will reduce $\beta(u, t)$ and hence number of the runs.

LUCB2 (Modified LUCB1)

The conditions which the variables should satisfy have been described below.

- $4 \cdot \tilde{\delta} \leq \cdot(146 \cdot n \cdot \log(\frac{n}{\delta}))^2$
- $\sum_{i=1}^{\infty} \frac{1}{i^3} \geq (\sum_{i=t_0}^{\infty} \frac{1}{i^3}) \cdot \frac{\tilde{\delta}}{\delta}$

We choose the maximum $\tilde{\delta}$ and minimum t_0 satisfying the above inequalities.
We can argue theoretically[1] that on the modifications mentioned above we can achieve the required confidence as required for the PAC algorithm with much smaller number of runs.

[1] Detailed reasoning and mathematical argument for the same would be provided in the report

Results

We ran LUCB1 (original) and LUCB2 (modified) for several bandit instances. Two of those instances are given below:

```
b1 = [0.9, 0.8, 0.7, 0.6, 0.595, 0.592, 0.3, 0.2, 0.1]
b5 = [0.9, 0.8, 0.795, 0.655, 0.6224, 0.61, 0.6, 0.5, 0.4,
    0.3, 0.2, 0.1]
```

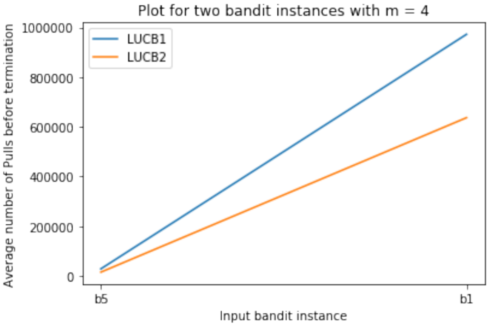


Figura: Sample complexity for $m = 4$, $\delta = 0{,}1$ over b1 and b5

Results

Here are some more instances with varied means and number of arms. In every case $m = 4$, $\delta = 0{,}1$.

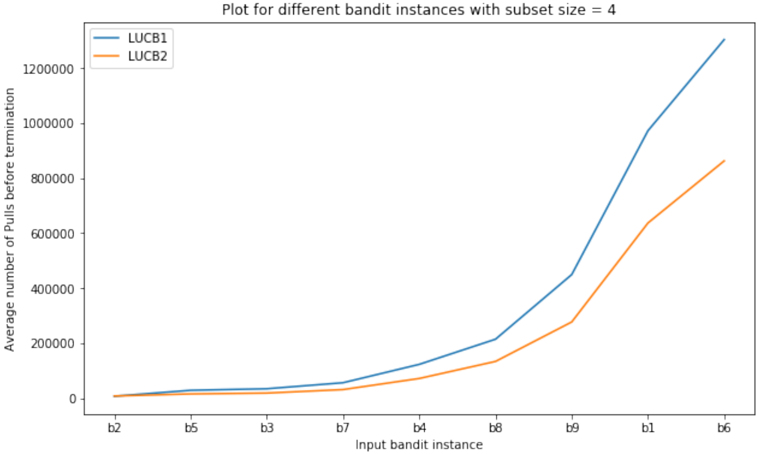


Figura: Sample complexity

An alternative notion of ϵ-closeness

In [1], Cao et. al. consider a similar setting where m arms are to be chosen with a different notion of ϵ-closeness.

An alternative notion of ϵ-closeness

In [1], Cao et. al. consider a similar setting where m arms are to be chosen with a different notion of ϵ-closeness.

Objective

Given parameters ϵ and δ, we want to select an m-sized subset of arms, V, such that with probability at least $1 - \delta$, $p_i^V \geq (1 - \epsilon) p_i^{true}$.

An alternative notion of ϵ-closeness

In [1], Cao et. al. consider a similar setting where m arms are to be chosen with a different notion of ϵ-closeness.

Objective

Given parameters ϵ and δ, we want to select an m-sized subset of arms, V, such that with probability at least $1 - \delta$, $p_i^V \geq (1 - \epsilon)p_i^{true}$.

The main distinction is that here the tolerance is relative (proportional to true reward probability) rather than absolute.

Second Look at Halving [2]

$R_1 \leftarrow T_n$.
$\epsilon_1 \leftarrow \frac{\epsilon}{4}$; $\delta_1 \leftarrow \frac{\delta}{2}$.
for $l = 1$ to $\lceil log(\frac{n}{m}) \rceil$ **do**
  for all $a \in R_l$ **do**
    Sample arm a $\lceil \frac{2}{\epsilon_l^2} ln(\frac{3m}{\delta_l}) \rceil$ times; let $\hat{p}_a$ be its average reward.
  end for
  Find $R_l' \subset R_l$ such that $|R_l'| = \max(\lceil \frac{|R_l|}{2} \rceil, m)$, and $\forall i \in R_l \ \forall j \in (R_l - R_l')$: $(\hat{p}_i \geq \hat{p}_j)$.
  $R_{l+1} \leftarrow R_l'$.
  $\epsilon_{l+1} \leftarrow \frac{3}{4}\epsilon_l$; $\delta_{l+1} \leftarrow \frac{1}{2}\delta_l$.
end for
Return $R_{\lceil log(\frac{n}{m}) \rceil + 1}$.

Figura: Halving Algorithm

Adapting for Multiplicative Tolerance

Algorithm 1: ME-AS

input: B, ϵ, δ, k
for $\mu = 1/2, 1/4, \ldots$ **do**
  $S = \text{ME}(B, \epsilon, \delta, \mu, k)$;
  $\{(a_i, \hat{\theta}^{US}(a_i)) \mid 1 \leq i \leq k\} = \text{US}(S, \epsilon, \delta, (1 - \epsilon/2)\mu, k)$;
  if $\hat{\theta}^{US}(a_k) \geq 2\mu$ **then**
    return $\{a_1, \ldots, a_k\}$;

Algorithm 2: Median Elimination (ME)

input: B, ϵ, δ, μ, k
$S_1 = B$, $\epsilon_1 = \epsilon/16$, $\delta_1 = \delta/8$, $\mu_1 = \mu$, and $\ell = 1$;
while $|S_\ell| > 4k$ **do**
  sample every arm $a \in S_\ell$ for $Q_\ell = (12/\epsilon_\ell^2)(1/\mu_\ell)\log(6k/\delta_\ell)$ times;
  for *each arm* $a \in S_\ell$ **do**
    its empirical value $\hat{\theta}(a)$ = the average of the Q_ℓ samples from a;
  $a_1, \ldots, a_{|S_\ell|}$ = the arms sorted in non-increasing order of their empirical values;
  $S_{\ell+1} = \{a_1, \ldots, a_{|S_\ell|/2}\}$;
  $\epsilon_{\ell+1} = 3\epsilon_\ell/4$, $\delta_{\ell+1} = \delta_\ell/2$, $\mu_{\ell+1} = (1 - \epsilon_\ell)\mu_\ell$, and $\ell = \ell + 1$;
return S_ℓ;

Figura: Guess and Validate Mechanism

More ideas

For this variant, they propose a halving based algorithm with sample complexity $O(\frac{n}{\epsilon^2}\frac{1}{p_k^{true}}\log\frac{m}{\delta})$

Can we have an improvement over this by proposing a greedy strategy similar to LUCB?

References

[1] Cao, W., Li, J., Tao, Y., and Li, Z. On top-k selection in multi-armed bandits and hidden bipartite graphs. In *Advances in Neural Information Processing Systems* (2015), pp. 1036–1044.

[2] Kalyanakrishnan, S., and Stone, P. Efficient selection of multiple bandit arms: Theory and practice. In *Proceedings of the Twenty-seventh International Conference on Machine Learning (ICML 2010)* (2010), J. Fürnkranz and T. Joachims, Eds., Omnipress, pp. 511–518.

[3] Kalyanakrishnan, S., Tewari, A., Auer, P., and Stone, P. PAC subset selection in stochastic multi-armed bandits. In *Proceedings of the Twenty-ninth International Conference on Machine Learning (ICML 2012)* (New York, NY, USA, 2012), J. Langford and J. Pineau, Eds., Omnipress, pp. 655–662.

Thank You


THANK YOU
GRACIAS
ARIGATO
SHUKURIA
DANKSCHEEN
TASHAKKUR ATU
YAQHANYELAY
SUKSAMA
EKHMET
BIYAN
SHUKRIA
TINGKI
GOZAIMASHITA
EFCHARISTO
JUSPAXAR
KOMAPSUMNIDA
MAAKE
GRAZIE
MEHRBANI
PALDIES
BOLZIN
MERCI