Firing Bandits: Optimizing Crowdfunding

Lalit Jain [1] Kevin Jamieson [1]

Abstract

In this paper, we model the problem of optimizing crowdfunding platforms, such as the non-profit Kiva or for-profit KickStarter, as a variant of the multi-armed bandit problem. In our setting, Bernoulli arms emit no rewards until their cumulative number of successes over any number of trials exceeds a fixed threshold and then provides no additional reward for any additional trials - a process reminiscent to that of a neuron firing once it reaches the action potential and then saturates. In the spirit of an infinite armed bandit problem, the player can add new arms whose expected probability of success is drawn iid from an unknown distribution – this endless supply of projects models the harsh reality that the number of projects seeking funding greatly exceeds the total capital available by lenders. Crowdfunding platforms naturally fall under this setting where the arms are potential projects, and their probability of success is the probability that a potential funder decides to fund it after reviewing it. The goal is to play arms (prioritize the display of projects on a webpage) to maximize the number of arms that reach the firing threshold (meet their goal amount) using as few total trials (number of impressions) as possible over all the played arms. We provide an algorithm for this setting and prove sublinear regret bounds.

1. Introduction

In the crowdfunding paradigm, a crowd of many users pledge to contribute a small amount of funding toward a project, but the project is only funded if the total amount of funding pledged exceeds a known reserve price. This is reminiscient of the firing of a neuron once it reaches its action potential - where the number of pledges from the crowd controls the firing. Recent years have seen a huge proliferation of crowdfunding sites, with over 700 platforms in 2012 and an estimated 2000 in 2016. These platforms account for a significant amount of investment; the World Bank estimates that $90 billion will be raised through crowdfunding ventures alone in 2020 (Barnett, 2015). This includes non-profit micro-lending sites such as Kiva, charity organizations DonorsChoose and GoFundMe, and venture capital efforts such as Kickstarter and IndieGoGo.

Crowdfunding platforms seeking to maximize the number of projects funded need to decide when and how to show projects to users visiting their sites. The administrators of a platform can influence the outcomes of which projects are more likely to be funded by showing them more or less often in display results. Naive strategies that do not prioritize the most likely projects to reach the reserve price within a reasonable amount of time are doomed to only partially fund many projects (that fail to fire), versus a preferred outcome of a moderate number fully funded to the reserve price (and fire).

An intuitive strategy to maximize the number of projects that hit their reserve price is to first estimate the popularity of a project by considering the proportion of users who pledged funding relative to the total number of users that reviewed the project. Then projects on the webpage can be prioritized by popularity. This is a challenge because the popularity cannot be accurately estimated without having the prioritization to ensure enough reviews are gathered. Simultaneously, we want to avoid prioritizing unpopular projects. Through personal correspondences with some of these popular platforms, we have learned that these companies rely on poorly motivated heuristics. Indeed, we became aware of this problem from one such platform seeking assistance. Despite the enormous interest, capital investment, and potential for small improvements to make large impact on the number of projects funded, we are unaware of any rigorous mathematical study or approaches to optimize these mechanisms. Our paper seeks to bridge this gap and is organized as follows. In the next section we specify our problem statement precisely. We then present our algorithm in Section 2 and prove a sub-linear regret bound. Finally, we present both synthetic experiments and real-world scenarios derived from data on the Kiva platform.

[1]Computer Science and Engineering, University of Washington, Seattle, USA. Correspondence to: Lalit Jain <lalitj@cs.washington.edu>, Kevin Jamieson <jamieson@cs.washington.edu>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

1.1. The Problem Statement

Two key features define recommendation for the crowdfunding problem. Firstly, since the goal is to fund as many projects as possible, rather than just collect as many pledges as possible, we do not see a reward for an arm until the project has hit its reserve price. In addition, once funded, the project effectively exits the recommendation system. Secondly, in most crowdfunding models, the demand of lenders is vastly exceeded by the supply of funding seekers. Thus, at any given time there is an exceedingly large pool of loans to choose from when making a recommendation to a user, and this pool is constantly growing.

We also make a simplifying assumption that each pledge is the same amount, and that each loan needs the same number of pledges. Thus, instead of worrying how much funding a project raises, we only focus on the number of pledges. This is actually a realistic assumption, since in many platforms, the majority of pledges are a single small amount such as \$25 (see (Barnett, 2015)).

With the above considerations in mind, we model each project as a biased coin with mean μ: showing the project to a lender is a flip of the coin, and μ is the expected proportion of times the project receives a pledge when shown. We assume at any time we can obtain a new project/coin whose mean is drawn i.i.d. from a reservoir distribution F defined on $[0, 1]$. For a project to reach its reserve price, i.e. the coin to fire, we need the number of successes to reach a threshold of $\tau \in \mathbb{N}$ in a sequence of flips from μ. Throughout the rest of the paper, we adopt the vocabulary of "coin" and "flips". Unlike models normally used for search, we assume that a single project is being chosen at each time to be evaluated. We believe that it should be possible to generalize to more general settings in which multiple loans are displayed simultaneously in search results (e.g. cascading bandits (Kveton et al., 2015; Szepesvári et al., 2015) or best-of-K-bandits (Simchowitz et al., 2016)).

The protocol for any algorithm for firing bandits is shown below. The algorithm maintains a set of coins and at every time step we are choose between two actions: either flip an existing coin, or draw a new coin from F and flip it. Our goal is to find a policy that maximizes the number of coins that reach τ successes and fire.

Firing Bandits Protocol

Input: Reserve price τ
$\mathcal{S}$: Set of coins being maintained by algorithm.
for $t = 1, 2, \ldots$
Algorithm does one of the following:

- Draw a new coin from F, flip it and add it to $\mathcal{S}$.
- Choose a coin from $\mathcal{S}$ and flip it. If the coin has had a total of τ successes, it fires to yield a reward of 1 and is then removed from $\mathcal{S}$.

1.2. Fixed Budget Policies

Without loss of generality, we can assume an optimal policy following the protocol has full knowledge of F (indeed, there will be different optimal policies for different reservoir distributions). Since the means of the coins are drawn iid from F, the flips of different coins are independent and, in particular, they provide no information about each other. This immediately implies that there exists an optimal policy that treats each coin individually and identically. The implication of this is crucial: *there exists an optimal policy that looks at coins one at a time, choosing only to keep flipping or discarding a coin based on its history of successeses.*

However, since we do not have access to F, our approach is to optimize over a policy class that we believe contains a near- optimal policy. Such a policy search naturally appears in multi-armed bandit problems. For example in the standard multi-armed bandit each arm gives rise to the policy that only pulls that arm in each round and the optimal policy corresponds to playing the arm with the highest mean. A more interesting example is contextual bandits where policies map context feature vectors to actions. Algorithms for multi-armed bandits optimize over this set of policies to find the optimal arm.

For a given coin with mean $\mu \sim F$ (where we refer to μ simultaneously as the coin and the mean of the coin), let $\{X_i\}_{i=1}^T \subset \{0, 1\}^T$ be an empirical sequence of flips and let $X(t) = \sum_{i=1}^t X_i$. The sequence of flips give rise to a random walk $(t, X(t))$ that we refer to as the coin's *trajectory*. A coin fires at time t only if $X(t) = \tau$. The line with slope 1 corresponds to the trajectory of a coin with probability 1 of success, and similarly the horizontal axis corresponds to a trajectory of a coin with 0 probability of heads– each random walk $X(t)$ lies between these two lines. Furthermore, the average slope of a random walk after t steps, $X(t)/t$ is an empirical estimate of the value of μ. By the theory of random walks we expect $X(t) \in \mu t \pm \sqrt{2\mu(1-\mu)t}$ with constant probability. The minimum t such that $X(t) = \tau$ is a negative binomial random variable with mean τ/μ.

A policy is given by a non-decreasing function $\pi(t)$, where μ is rejected at time t if $X(t) \leq \pi(t)$.

Intuitively, if $X(t) > \pi(t)$ there is some hope that $\mu t > \pi(t)$ and that the positive trend should continue until it reaches τ successes and fires. Because $X(t)$ is non-decreasing, it only makes sense to consider policies $\pi(t)$ that are also non-decreasing. Indeed, if π is ever decreasing, it would behave just as if it remained constant. The difficulty of finding the optimal policy is now apparent–it requires a search over the space of all non-decreasing functions. Even if we knew F, it is not clear how we could carry out such a search.

There exist several natural families of policies arising in the sequential testing literature (see (Siegmund, 1985)). Notable classes include the SPRT which corresponds to $\pi(t)$ being an affine function (Figure 1). In this work we choose to study fixed budget policies, which correspond to vertical thresholds.

It remains to define a rich enough policy class for our algorithm to search over that only considers a single coin at a time. A **Fixed Budget Policy** π_B evaluates a coin by repeatedly flipping it and rejecting it if τ successes have not been reached within B flips(note, to prevent additional notation, we may use B to simultaneously refer to the number of flips and the policy π_B). Note, we expect coins that do *not* fire have means less than τ/B. In practice, π_B corresponds to showing a project on a crowdfunding site at most B times, and then retiring it if it fails to get funded. This type of policy is appealing for a number of reasons: 1) it is intuitively fair to loan seeking applicants and simple to implement, 2) executing π_B also simultaneously evaluates every $\pi_{B'}$ for $B' \leq B$ (we will expand on this later), and 3) leads to interpretable regret bounds.

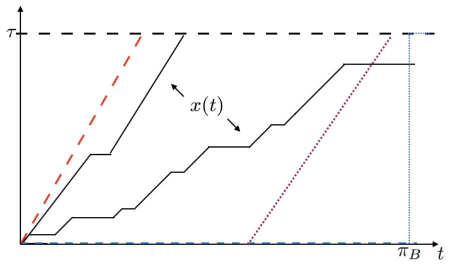

Figure 1. Example trajectory. The red dashed line corresponds to the trajectory of a coin with success probability 1. The purple line is an example of a linear SPRT boundary, and the blue dashed line is a fixed budget policy π_B.

The problem of finding the best B will occupy the rest of this paper. If we had knowledge of F, we could use calculus techniques to find the best value of B. Faced with only access to the empirical flips of coins that expire, we will have to consider all possible values of B.

1.3. Related Work

The firing bandit problem involves several salient features which have not been previously addressed in the literature. Firstly is the aspect of only receiving a reward once enough successes have been observed. Past work on delayed bandits (Vernade et al., 2017; Joulani et al., 2013) has considered the case when rewards may not be received until a later round, however this is very different from only receiving a single reward at some later time. There are also several (very) different bandit models referred to as Threshold bandits (Abernethy et al., 2016; Locatelli et al., 2016). While these papers share some of the same vocabulary, they are variants on the standard multi-armed bandit and there is no notion of a threshold being passed once enough successes are accumulated.

Secondly, once an arm has fired (reached τ heads), it can no longer be played. But we can optionally add as many arms as we want with means drawn i.i.d. from F. Many different bandit models have considered situations where the arms change over time. This includes sleeping bandits (Kleinberg et al., 2010) where the set of arms is variable over time. In mortal bandits (Chakrabarti et al., 2009) each arm has a different budget of the number of pulls it receives before going away and being replaced by a new arm whose underlying mean is drawn from a distribution F. While related, the major difference is that in mortal bandits rewards are accumulated as they are observed instead of only after the arm expires, which is the core difficulty of our problem.

In some sense, our problem is most closely modeled by an infinite armed bandit problem. In this setting we have infinitely many arms to choose from with the goal of maximizing cumulative reward. The mean rewards of the arms are distributed according to an underlying F. In the setting where F is semi-parametric or known, extensive work has been done in both the pure exploration case where the goal is to find an arm close to the optimal (see (Carpentier & Valko, 2015; Jamieson et al., 2016)), and in the cumulative expected regret setting (Berry et al., 1997; Wang et al., 2009; Bonald & Proutiere, 2013). Recent work has provided algorithms for the stochastic and non-stochastic pure exploration setting case when the distribution is unknown (Li et al., 2016).

Most algorithms for infinite armed bandits follow a similar mold. A large number of arms are sampled from the distribution (sometimes requiring partial knowledge of the underlying distribution to ensure a sufficient number, but not too many for their budget) and then an algorithm like UCB (Auer et al., 2002) is executed. Algorithms exploit the fact that the arms are drawn i.i.d. from a distribution by knowing that if they've seen a good arm, there is likely another just as good still out there. Hence, an arm is typically sampled as long as it cannot be ruled out as a potential superior arm to those found so far. In our setting, however, our arms expire and it is not at all clear how such a technique could be adapted. Firstly, since the goal is to have as many coins fire as possible, it's unclear that dropping arms is a good idea - there is the possibility no arms would fire even after a very large number of samples using such a strategy. Secondly, as we will show in the experiments section, naive UCB style algorithms do not work wll in an anytime sense, since they fail to reject coins that take extremely long times to convert. There is the possibility of using a UCB style algorithm that keeps increasing the number of arms it considers as time progresses but we believe such a policy is very difficult to analyze in our setting. See (Wang et al., 2009) for an example of such a strategy.

2. Our Approach

Given a coin drawn i.i.d. from F, let the random variable C_B be the indicator that it fires under policy π_B. Moreover, let N_B be the random variable representing the number of times it is flipped by policy π_B; clearly, $\tau \leq N_B \leq B$. Of course, we will be applying π_B repeatedly to many coins and using the results to estimate $\mathbb{E}[C_B]$ and $\mathbb{E}[N_B]$. If we draw m coins from F and evaluate each one using policy π_B, let $\widehat{C}(B,m)$ the empirical proportion of the m coins that fired, and $\widehat{N}(B,m)$ the empirical mean of the number of times each of the m coins was flipped.

Intuitively, the efficacy of π_B should be given by the expected number of coins firing in a budget of V total flips over all coins. Let $m(V)$ be the random variable representing how many coins end up being evaluated by π_B using V total flips. The last coin we look at may be stopped prematurely, and we let $0 \leq \kappa \leq B$ denote the number of flips it was given. Then the number of coins that fire is given by $m(V)\widehat{C}(B,m)$ and $V = m(V)\widehat{N}(B,m(V)) + \kappa$. The rate at which coins fire is given by,

$$\mathbb{E}\left[\frac{m(V)\widehat{C}(B,m(V))}{V}\right] = \mathbb{E}\left[\frac{\widehat{C}(B,m(V))}{\widehat{N}(B,m(V)) + \kappa/m(V)}\right] \overset{V\to\infty}{=} \frac{\mathbb{P}(C_B = 1)}{\mathbb{E}[N_B]}$$

The last line follows by the law of large numbers since $m(V) \geq (V - B)/B$. This motivates defining for any B,

$$\rho(B) := \frac{\mathbb{P}(C_B = 1)}{\mathbb{E}[N_B]}, \qquad \rho_* := \max_B \rho(B).$$

As the total budget of flips V gets very large, the expected number of coins firing is $V\rho(B)$. Analogous to the empirical estimators given above, we can also define $\widehat{\rho}(B,m) = \widehat{C}(B,m)/\widehat{N}(B,m)$ For any m, $\widehat{\rho}(B,m)$ provides an asymptotically consistent estimator of $\rho(B)$. If our algorithm allocates a budget of V total flips to B as above, versus using an optimal fixed budget policy, asymptotically our expected loss in the number of coins firing is $V\rho_* - V\rho(B)$. We can generalize this to any algorithm that allocates a budget of V among policies. **Definition** Consider an algorithm allocated a budget of V total flips and considers a random number of $m(V)$ coins. In each round i, the algorithm draws a coin from F and flips it according to policy B_i using v_i flips (so $V = \sum_{i=1}^{m(V)} v_i$). Then, the **regret** of the algorithm is $V\rho_* - \mathbb{E}[\sum_{i=1}^{m(V)} v_i \rho(B_i)]$.

The regret expresses the number of coins *fewer* the algorithm fires relative to some policy B that satisfies $\rho(B) = \rho_*$. It's important to mention that our notion of regret is not based on the optimal policy over all policies but rather the weaker notion of the optimal fixed budget policy.

In general, the problem of identifying B with $\rho(B) = \rho_*$ is not well defined - there can be several such optimal policies B. We will let $B^* := \min\{B : \rho(B) = B^*\}$. Finding the optimal value of ρ_* can be difficult due to a lack of good optimzation properties - in general $\rho(B)$ need not be concave, much less unimodal.

Low Probability of Conversion. For most crowdfunding platforms such as Kiva or Kickstarter, the demand of funding seekers is far greater than the total capital available from all lenders. Hence very few projects have a chance of being funded. For example, only 3.6% of technology projects reach their funding goal on IndieGoGo and 80% of all projects do not reach a *quarter* of their reserve price (Jeffries, 2013). This suggests that only the best loans get funded, and that there are very few good loans. Any optimal policy will only target good loans drawn from F, i.e. loans with large values of μ, and even if the policy is funding the maximum rate of loans, the probability of any single loan being funded will be very low.

Hence throughout the rest of the paper, we will assume that the probability of a coin drawn from F obtaining τ successes (i.e., firing) by an optimal fixed budget policy is bounded above. Equivalently, the probability that a coin does not fire, even under the optimal policy, is bounded below.

Specifically, we assume there exists $\alpha > 0$ such that that $P(C_B = 1) < 1 - \alpha$, or equivalently, $P(C_B \neq 1) \geq \alpha$ for any $B \in \{B : \rho(B) = \rho_*\}$. In practice, again we expect the probability of any particular loan firing to be extremely small even for the best policy so we can safely assume $\alpha > 1/2$. Finally, since $P(C_B = 1)$ is increasing with B, we note that $P(C_B = 1) < 1 - \alpha$ for any $B \leq B^*$.

2.1. Algorithm Overview

The problem of finding the π_B with the maximum $\rho(B)$ will occupy the rest of this paper. If we had knowledge of F, we could compute $\rho(B)$ analytically. Faced with only access to the empirical flips of coins that expire leads to a classical exploration/exploitation tradeoff - our algorithm for finding the optimal policy will have to trade off exploring a large range of B with exploiting values that have empirically high values of $\hat{\rho}(B)$.

Our algorithm FiringUCB is presented below. Note that we assume FIRINGUCB and PULL both have access to a global history over all flips. We now describe it at a high level.

Throughout we assume we have a pre-defined sequence b_k, for concreteness we can take $b_k = 2^k$. We begin with a guess on a value of K such that $b_K \geq B^*$, (one can always take $K = 1$). The algorithm partitions the set of policies $(0, b_K]$ (it does not make sense to evaluate policies $B < \tau$ since we need at least τ success to fire, however we ignore this for brevity) into a series of brackets $(0, b_K] = \cup_{k=1}^{K}(b_{k-1}, b_k]$. For each $B \leq B_K$ we maintain empirical estimates $\widehat{\rho}(B, m)$ along with confidence bounds $UCB(B, m), LCB(B, m)$ such that $LCB(B, m) < \rho(B) < UCB(B, m)$ at all times (see Lemma 2.2) where m tracks the total number of coins evaluated by policy π_B. Ignoring INITIALIZE for a moment, at each round of FIRINGUCB , the subroutine PULL is run on the bracket containing the policy with the largest upper confidence bound.

Pulling the k-th bracket corresponds to drawing a coin from F and then executing π_B for the largest B in the bracket that has not yet been eliminated. These flips are then used to evaluate the other policies $\pi_{B'}$ for $B' \leq B$ by looking at the result of the first B' flips. The estimates of ρ, and the corresponding confidence bounds are then updated for each B in the bracket and in addition, sub-optimal policies are culled - any policy whose UCB is less than the maximum LCB in that round is removed. Since $|UCB(B, m_k) - LCB(B, m_k)| \to 0$ as $m_k \to \infty$ and each active policy in a bracket has been evaluated on the same number of coins, the set of active policies shrinks and eventually eliminates all suboptimal policies. A new bracket is added when the highest upper confidence bound drops below a pre-determined value and the UCB game continues with the addition of this new bracket.

As we will see in Theorem 2.3, using multiple brackets opposed to just a single bracket we obtain higher statistical power in our estimates, translating into improved sample complexities. Also, note that our initialization step is necessary to ensure that our confidence bounds hold - see the discussion after Lemma 2.2.

Our main theorem is the following, recall typically $\alpha > 1/2$:

Theorem 2.1. *If* FIRINGUCB *is run with a budget of V samples, and brackets defined by b_k satisfy $b_k/b_{k-1}^2 \leq 1$ for all k, then with probability at least $1-\delta$ the regret incurred is at most*

$$\bar{K}\frac{c}{\alpha^4\rho_*^2}\log(\tfrac{1}{\alpha\rho_*\delta}) + \sqrt{\tfrac{c}{\alpha^4}\bar{K}V\log(V)\log\left(\tfrac{\log(V/\alpha)}{\alpha\rho_*\delta}\right)}$$

where $\bar{K} = \max\{k : b_{k-1} \leq \frac{2}{\alpha\rho_*}\}$. If $b_{k+1} = b_k^2 = 2^{2^k}$ then $\bar{K} \leq \log_2(\log_2(\frac{2}{\alpha\rho_*}))$.

We now discuss each component of the algorithm in greater detail and explain our choice of brackets.

```
FIRINGUCB
Input: $\tau, \alpha, K$
Create brackets $\{(b_{k-1}, b_k]\}_{k=1}^{K}$
Global Variables:
$m_k$ : number of coins allocated to bracket $k$
$\mathcal{A}_k$: the set of active policies in $(b_{k-1}, b_k]$

Subroutine INITIALIZE($k$):
Run PULL ($k$), $U^{-1}(\alpha/4, \delta)$ times (see 1 for definition of $U$).
Remove any $B$ with $\widehat{C}(B, U^{-1}(\alpha/4, \delta)) > 1 - 3\alpha/4$.

UCB Algorithm:
for $k \leq K$: INITIALIZE($k$)
while True
    $\widehat{k} \leftarrow \text{argmax}_{k\in S}\ \max_{B\in\mathcal{A}_k}\ UCB(B, T_k)$
    Call PULL($k$)
    if $\max_{B\leq b_K} UCB(B, m_k) < \frac{2}{\alpha b_K}$
        $K \leftarrow K + 1$
        allocate new bracket $(b_{K-1}, b_K]$
        Call INITIALIZE($k$)
```

```
PULL
Input: bracket $k$
Draw a new coin with mean $\mu \sim F$.

$B := \max\{\mathcal{A}_k\}$
Execute $\pi_B$: Let $\{X_i\}_{i=1}^{m}$ be a sequence of $m \leq B$ flips, where
the flipping is stopped early if there are $\tau$ success and the coin fires.

$m_k \leftarrow m_k + 1$
Update:
for each active policy $B'$ in $\mathcal{A}_k$
    Evaluate policy $B'$ using $\{X_i\}_{i=1}^{B'}$.
    Update $\widehat{C}(B', m_k), \widehat{N}(B', m_k), \widehat{\rho}(B', m_k)$

Eliminate:
Let $LCB = \max_{B\in\mathcal{A}_k} LCB(B, m_k, \delta)$
for $B \in \mathcal{A}_k$
    if $UCB(B, m_k, \delta) < LCB$
        $\mathcal{A}_k \leftarrow \mathcal{A}_k - \{B\}$
```

2.2. Analysis: A Single Bracket

Since every call of PULL gives an additional budget to a fixed bracket, we can analyze the regret a specific bracket contributes to the total regret independently of the other brackets. First we explain the confidence bounds used in PULL .

Much of our analysis relies on an anytime confidence bound. Specifically, we assume the existence of a constant c_U such that if $U(m, \delta) := \sqrt{\frac{c_U \log(\log_2(2m)/\delta)}{m}}$, and X_i are i.i.d. bounded random variables then

$$\mathbb{P}\left(\bigcup_{m=1}^{\infty}\left\{\left|\frac{1}{m}\sum_{i=1}^{m} X_i - \mathbb{E}[X_i]\right| \geq U(m, \delta)\right\}\right) \leq \delta \quad (1)$$

for any $\delta \in (0,1)$. One can show that $c_U = 4$ suffices, but there exist substantially tighter expressions, albeit not as succinctly stated (c.f. Theorem 8 of (Kaufmann et al., 2016)). For the same c_U it is known that $U^{-1}(\epsilon,\delta) = \min\{m : U(m,\delta) \le \epsilon\} \le c_U\epsilon^{-2}\log(2\log(\frac{ec_U\epsilon^{-2}}{\delta})/\delta)$ (c.f. (Jamieson, 2015)). Both $\widehat{C}(B,m)$ and $\widehat{N}(B,m)$ are empirical means of bounded i.i.d. random variables (in $[0,1]$ and $[0,B]$, respectively), thus we can apply Equation 1. Recall that $\widehat{\rho}(B,m) = \frac{\widehat{C}(B,m)}{\widehat{N}(B,m)}$. Define

$$LCB(B,m,\delta) := \frac{\widehat{C}(B,m) - U(m,\delta/4B^2)}{\widehat{N}(B,m) + BU(m,\delta/4B^2)},$$
$$UCB(B,m,\delta) := \frac{\widehat{C}(B,m) + U(m,\delta/4B^2)}{\widehat{N}(B,m) - BU(m,\delta/4B^2)}.$$

and $\phi(B,m,\delta) := \frac{16}{\alpha^2 B}U(m,\frac{\delta}{4B^2})$. Using the bound on $U^{-1}(\epsilon,\delta)$ given above, we see that

$$\begin{aligned}\phi^{-1}(B,\epsilon,\delta) &:= \min\{m : \phi(B,m,\delta) \le \epsilon\}\\ &\le \tfrac{16c_U\epsilon^{-2}}{\alpha^4B^2}\log 8B^2\log\tfrac{16ec_U\epsilon^{-2}}{\alpha^4B^2\delta}/\delta.\end{aligned}$$

Define the events

$$\mathcal{E}_c = \bigcap_{B=1}^{\infty}\bigcap_{m=1}^{\infty}\left\{|\widehat{C}(B,m) - c_B| \le U(m,\tfrac{\delta}{4B^2})\right\}$$
$$\mathcal{E}_n = \bigcap_{B=1}^{\infty}\bigcap_{m=1}^{\infty}\left\{\tfrac{1}{B}|\widehat{N}(B,m) - n_B| \le U(m,\tfrac{\delta}{4B^2})\right\}$$

Lemma 2.2. *For the events defined above, $\mathbb{P}(\mathcal{E}_c^c \cup \mathcal{E}_n^c) < \delta$. On the event, $\mathcal{E}_c \cap \mathcal{E}_n$ the following happen. After* INITIALIZE *is called on bracket k, each $B \in (b_{k-1}, b_K]$ satisfies $P(C_B \neq 1) > \alpha/2$, and $U(m,\delta) < \alpha/4$. In addition, for any $m > 0$ and any $B \in \mathbb{N}$ satisfying these conditions we have that with probability greater than $1-\delta$*

$$\rho(B) \le UCB(B,m,\delta) \le \hat{\rho}(B,m) + \phi(B,m,\delta)$$
$$\rho(B) \ge LCB(B,m,\delta) \ge \widehat{\rho}(B,m) - \phi(B,m,\delta).$$

In general $UCB(B,m,\delta)$ could be negative so the call to INITIALIZE prevents this and ensures the confidence bounds given hold. In practice $\alpha > 1/2$ so the number of calls to Pull needed by INITIALIZE is $O(1)$ for each bracket, so the initialization step will not contribute to regret. Throughout the following, we assume the event $\mathcal{E}_c \cap \mathcal{E}_n$.

Note that the confidence interval given by ϕ implies that it takes fewer pulls to reach the same level of statistical confidence for large values of B compared to smaller values. We are now ready to state our main regret theorem for a single bracket.

Theorem 2.3. *Suppose* PULL *is called m times on bracket k and uses V total flips. Assume on the i-th pull, the maximum active policy is B_i and takes $v_i \le B_i$ flips. Then,*

$$\begin{aligned}V\rho_* - \sum_{i=1}^{m} v_i\rho(B_i) &\le V\Delta_k\\ &+ \sqrt{cV\tfrac{b_k}{\alpha^4 b_{k-1}^2}}\log(V)\log\left(b_{k-1}\tfrac{\log(V/\alpha)}{\delta}\right)\end{aligned}$$

where c is an absolute constant $\rho_{k,*} := \max_{b_{k-1}<B\le b_k}\rho(B)$ and $\Delta_k := \rho_* - \rho_{k,*}$.

Sketch of Proof. For any policy B in bracket k,

$$\rho_* - \rho(B) = \rho_* - \rho_k^* + \rho_k^* - \rho(B) = \Delta_k + (\rho_k^* - \rho(B)).$$

Thus the regret using policy B decomposes into two terms: the regret of the best policy in the bracket relative to playing ρ_*, plus the regret of policy B relative to the best policy in bracket. This gives the first term. For the second term, we bound the number of pulls to the bracket it takes to eliminate policies with regret more than ϵ by $\phi^{-1}(b_{k-1},\epsilon,\delta)$ and then applying a peeling argument. See the supplementary for details.

2.3. Analysis: UCB for Multiple brackets

The appearance of the term b_k/b_{k-1}^2 in the regret expression of Theorem 2.3 shows the advantage of using brackes. If we use a single bracket e.g. $(\tau, B_{\max}]$, we incur an additional factor of $B_{\max}$ in our regret. This also motivates the choice of sequence, namely we should choose b_k to satisfy $b_k/b_{k-1}^2 < 1$. One such sequence is $b_k = 2^{2^k}$, however we find that $b_k = 2^k$ works well in practice. Finally, throughout this section we assume that $B^* < b_K$, i.e. the best policy is contained in the set of current policies and that we are not growing brackets - we will handle the more general case in the next section.

Our argument in this section is similar to that used in the standard UCB analysis. We first quantify the total number of calls of PULL by a sub-optimal bracket. Using this, we bound the regret.

Lemma 2.4. *If* FIRINGUCB *is run with K brackets such that $\max_{B\le b_K}\rho(B) = \rho^*$ then any bracket k with $\Delta_k > 0$ is allocated at most $\phi^{-1}(b_{k-1},\Delta_k/2,\delta)$ coins.*

Proof. For any $B^* \le b_K$ that satisfies $\rho(B^*) = \rho_*$ we have for all $m \in \mathbb{N}$, $\widehat{\rho}(B^*,m) + \phi(B^*,m,\delta) \ge \rho(B^*) = \rho_*$. On the other hand, for any $B \in (b_{k-1}, b_k]$ such that $B^* \notin (b_{k-1}, b_k]$ we have

$$\begin{aligned}\widehat{\rho}(B,t) + \phi(B,t,\delta) &\le \rho(B) + 2\phi(B,t,\delta)\\ &\le \rho_{k,*} + 2\phi(b_{k-1},t,\delta)\end{aligned}$$

Thus if $B_* \le B_K$, for all $t \ge \phi^{-1}(b_{k-1},\Delta_k/2,\delta)$ we have $\widehat{\rho}(B,t) + \phi(B,t,\delta) \le \rho_*$ which implies that the sub-optimal bracket will not have the largest confidence bound. □

Theorem 2.5. *Suppose* FIRINGUCB *is run on K brackets with a total of V flips across all brackets, and such that* $\max_{B \le b_K} \rho(B) = \rho^*$. *The total regret incurred is bounded by* $\sqrt{\frac{c}{\alpha^4} K V \log(V) \log\left(\frac{\log(V/\alpha)}{\delta/b_{K-1}}\right)}$.

Proof. Suppose bracket k has used V_k total flips (e.g., the sum total flips over all coins it had been allocated in calls to PULL), then by Theorem 2.3 the regret incurred by bracket k is bounded by

$$\Delta_k V_k + \sqrt{\frac{c}{\alpha^4} V_k \log(V_k) \log\left(\frac{\log(V_k/\alpha)}{\delta/b_{k-1}}\right)}$$

using the fact that $\frac{b_k}{b_{k-1}^2} \le 1$. The total regret is given by summing over all $k \le K$. Now let's analyze each term separately - first focusing on the summed second term. By Jensen's inequality,

$$\sum_{k \le K} \sqrt{\frac{c}{\alpha^4} V_k \log(V_k) \log\left(\frac{\log(V_k/\alpha)}{\delta/b_{k-1}}\right)} \le \sqrt{\frac{c}{\alpha^4} K \log(V) \log\left(\frac{\log(V/\alpha)}{\delta/b_{K-1}}\right) V}$$

using the fact that $V_k \le V, b_{k-1} \le b_K$ and $\sum_{k \le K} V_k = V$. To bound the summed first term we use a peeling argument (analogous to that in Theorem 2.3). By Lemma 2.4 for any $\Delta_k > 0$ we have $V_k \le b_k \phi^{-1}(b_{k-1}, \Delta_k/2, \delta) \le b_k \frac{64 c_U \Delta_k^{-2}}{\alpha^4 b_{k-1}^2} \log(8 b_{k-1}^2 \log(\frac{64 e c_U \Delta_k^{-2}}{\alpha^4 b_{k-1}^2 \delta})/\delta)$. For any $\eta > 0$

$$\begin{aligned}
\sum_{k \le K} \Delta_k V_k &= \sum_{k:\Delta_k < \eta} \Delta_k V_k + \sum_{k:\Delta_k \ge \eta} \Delta_k V_k \\
&\le \eta V + \sum_{k:\Delta_k \ge \eta} \Delta_k b_k \frac{64 c_U \Delta_k^{-2}}{\alpha^4 b_{k-1}^2} \log\left(\frac{8 \log(\frac{64 e c_U \Delta_k^{-2}}{\alpha^4 b_{k-1}^2 \delta})}{\delta / b_{k-1}^2}\right) \\
&\le \eta V + \sum_{k:\Delta_k \ge \eta} \Delta_k^{-1} \frac{64 c_U}{\alpha^4} \log\left(\frac{8 \log(\frac{64 e c_U \Delta_k^{-2}}{\alpha^4 \delta})}{\delta / b_{k-1}^2}\right) \\
&\le \eta V + \frac{1}{\eta} \frac{64 c_U}{\alpha^4} \sum_{k \le K} \log(8 b_{k-1}^2 \log(\frac{64 e c_U \eta^{-2}}{\alpha^4 \delta})/\delta) \\
&\le \eta V + \frac{K}{\eta} \frac{64 c_U}{\alpha^4} \log(8 b_{K-1}^2 \log(\frac{64 e c_U \eta^{-2}}{\alpha^4 \delta})/\delta)
\end{aligned}$$

Optimizing over η, the righthand side can be bounded by $\sqrt{\frac{c}{\alpha^4} K V \log(\frac{\log(V/\alpha)}{\delta/b_{K-1}})}$. $\square$

2.4. Expanding Brackets

Returning to the definition of $\rho(B)$, recall that by the initialization step for B^* and any B under consideration, $n_B \ge B\mathbb{P}(C_B \ne 1) \ge B\alpha/2$ so that

$$\rho(B) = \frac{c_B}{n_B} \le \frac{c_B}{B\alpha/2} \le \frac{2}{\alpha B}$$

In particular, this implies that $\rho_* \le \frac{2}{\alpha B^*}$. This motivates Algorithm 2.1, we add on brackets opportunistically whenever there is a possibility that B^* is not yet being considered. The proof of Theorem 2.1 is given in the supplementary materials.

3. Experiments

We compare FIRINGUCB with a number of alternatives. The first is a natural modification of FIRINGUCB we call FIRINGBANDITSTHRESHOLD: instead of waiting for a budget B before rejecting a coin that does not convert, we monitor the empirical mean $\widehat{\mu}_k$ after k flips and stop sampling at flip $\min\{k \le B : k\, d(\widehat{\mu}_k, \tau/B) \ge \log(1/\delta)\}$ (marking it as not converting in B flips) where $d(\cdot,\cdot)$ is the KL divergence of two Bernoullis and $\delta = .05$. This variant attempts to predict whether a coin will not convert within B flips before the full B flips are taken and is motivated by the guarantees of a Chernoff bound, but has no rigorous theoretical justification. With the exception of this change, we maintained estimates of $\widehat{\rho}(B, m)$ for each B identically to FIRINGUCB and eliminated brackets as needed.

We also compare to a variant of the standard UCB algorithm. For a fixed value of k, NAIVEUCB-k maintains a pool of k coins. Identically to UCB (we used the anytime formulation given in (Abbasi-Yadkori et al., 2011)), at each round the coin with the highest UCB is flipped and its empirical mean and number of flips are updated. Once a coin reaches τ heads, it is removed from the pool, a reward of 1 is received, and a new coin is drawn to replace it.

Finally, as discussed in Section 1.2, the optimal policy with respect to a given coin reservoir distribution for evaluating coins will consist of an increasing boundary. Though a search over this space is difficult in general, we considered the set of all linear boundaries, parametrized as $at - b, a, b > 0$. Recall the traditional sequential probability ratio test (SPRT) corresponds to one of these boundaries (Siegmund, 1985). For each of the reservoir distributions considered below, we did an exhaustive search to discover the optimal values of a, b (as always in terms of the number of coins converted in a fixed budget). We refer to the algorithm that repeatedly applies the policy $at - b$ as LINEARBOUNDARY-a, b .

We considered two examples of reservoir distributions. Firstly, a synthetic example using $F_1 = \text{Beta}(1/20, 1)$ as the reservoir distribution, and $\tau_1 = 10$.

Secondly, we considered a distribution F_2 derived from actual page click data arising from the search page of the microloan crowdfunding site Kiva (`www.kiva.org`). During a period of a week earlier this year, roughly 10000 microloans were listed on Kiva, and on average each loan was shown about 500 times. The average requested loan

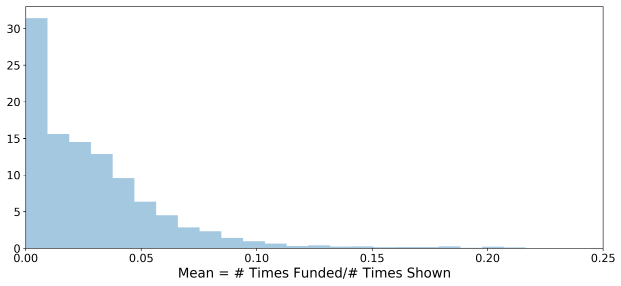

Figure 2. F_2: Empirical distribution of means for Kiva loans shown during in a fixed week.

amount on Kiva is \$500, and the amount pledged per user funding a loan was equal to \$25 more than half the time (the default amount). The underlying distribution of empirical μ's (i.e. # times funded/#times shown to potential funders) is in Figure 3. The mean μ was $\approx .03$. We used this empirical distribution as our reservoir distribution and took $\tau = 25$.

Figure 3 displays the results for F_1 and Figure 3 for the Kiva distribution of shown in Figure F_2. In both cases, we supplied each algorithm with a budget of 5×10^6 total flips over all coins considered, and we considered 50 repetitions of each algorithm with 95% confidence intervals plotted using a normal approximation. The optimal linear boundaries in both cases are indicated in the legend, note $b = 0$ in both cases.

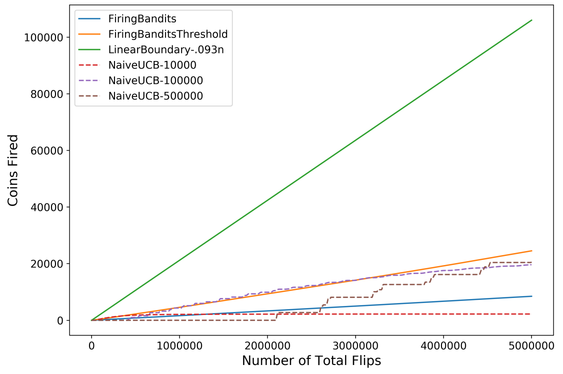

Figure 3. Coins fired when the reservoir distribution is F_1.

The advantage of FIRINGUCB and FIRINGBANDITSTHRESHOLDcompared to NAIVEUCB-k is immediately apparent in this plot - though NAIVEUCB-k can achieve higher reward in a small time horizon for large values of k, it fails to provide an anytime regret guarantee. Indeed,

Algorithm	F_1	F_2
FIRINGUCB	135190 ± 2712	6439 ± 98
FIRINGBANDITSTHRESHOLD	387038 ± 6242	11269 ± 109
LINEARBOUNDARY-a, b	2546582 ± 1044	1242166 ± 1479
NAIVEUCB-1000	-	2105 ± 266
NAIVEUCB-10000	11205 ± 18	17270 ± 332
NAIVEUCB-100000	110627 ± 42	102335 ± 29
NAIVEUCB-500000	508438 ± 37	-

Table 1. Number of Coins considered.

NAIVEUCB-k will quickly convert any good coin it samples, and then be burdened down by many subpar arms that it is forced to convert before receiving a good arm. FIRINGUCB successfully manages to avoid this problem by searching over policy classes that quickly disregard subpar coins.

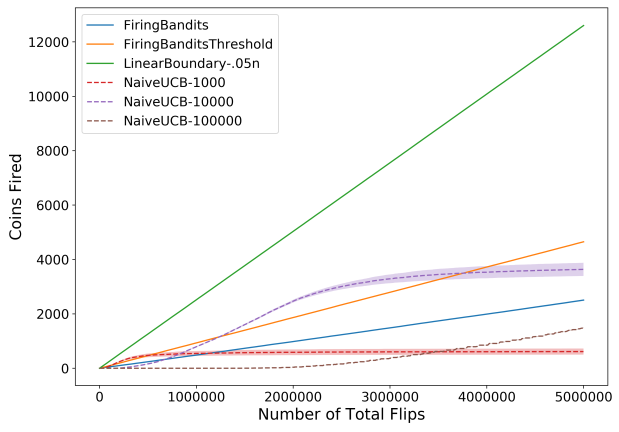

Figure 4. Coins fired when the reservoir distribution is F_2.

In both cases the LINEARBOUNDARY-a, b policy greatly outperforms the FIRINGUCB algorithms (by almost a factor of 4). Recall that in both cases LINEARBOUNDARY-a, b is a line of the form at, thus any coin that does not get a head on the first flip will be immediately rejected. This aggressiveness is most apparent when considering the number of coins that each policy considers, for instance, LINEARBOUNDARY-a, b considers almost 200 times more coins than FIRINGUCB (see Table 1). In a crowdfunding environment, such a policy that rejects loans after displaying it just a single time would certainly be considered unfair and very impractical.

4. Acknowledgments

Finally, we would like to thank Kiva, especially Melissa Fabros and Kevin O'Brien for their invaluable help and providing us with data.

References

Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pp. 2312–2320, 2011.

Abernethy, J. D., Amin, K., and Zhu, R. Threshold bandits, with and without censored feedback. In *Advances In Neural Information Processing Systems*, pp. 4889–4897, 2016.

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

Barnett, C. Trends show crowdfunding to surpass vc in 2016, 2015. URL https://www.forbes.com/sites/chancebarnett/2015/06/09/trends-show-crowdfunding-to-surpass\-vc-in-2016/2/#3d3d1aef666f. [Online; accessed 9-February-2018].

Berry, D. A., Chen, R. W., Zame, A., Heath, D. C., and Shepp, L. A. Bandit problems with infinitely many arms. *The Annals of Statistics*, pp. 2103–2116, 1997.

Bonald, T. and Proutiere, A. Two-target algorithms for infinite-armed bandits with bernoulli rewards. In *Advances in Neural Information Processing Systems*, pp. 2184–2192, 2013.

Carpentier, A. and Valko, M. Simple regret for infinitely many armed bandits. In *ICML*, pp. 1133–1141, 2015.

Chakrabarti, D., Kumar, R., Radlinski, F., and Upfal, E. Mortal multi-armed bandits. In *Advances in neural information processing systems*, pp. 273–280, 2009.

Jamieson, K. *The Analysis of Adaptive Data Collection Methods for Machine Learning*. PhD thesis, University of Wisconsin-Madison, 2015.

Jamieson, K. G., Haas, D., and Recht, B. The power of adaptivity in identifying statistical alternatives. In *Advances in Neural Information Processing Systems*, pp. 775–783, 2016.

Jeffries, A. Indie no-go: only one in ten projects gets fully funded on kickstarter's biggest rival, 2013. URL https://www.theverge.com/2013/8/7/4594824/less-than-10-percent-of-projects-on\-indiegogo-get-fully-funded. [Online; accessed 9-February-2018].

Joulani, P., Gyorgy, A., and Szepesvári, C. Online learning under delayed feedback. In *International Conference on Machine Learning*, pp. 1453–1461, 2013.

Kaufmann, E., Cappé, O., and Garivier, A. On the complexity of best-arm identification in multi-armed bandit models. *The Journal of Machine Learning Research*, 17 (1):1–42, 2016.

Kleinberg, R., Niculescu-Mizil, A., and Sharma, Y. Regret bounds for sleeping experts and bandits. *Machine learning*, 80(2-3):245–272, 2010.

Kveton, B., Szepesvari, C., Wen, Z., and Ashkan, A. Cascading bandits: Learning to rank in the cascade model. In *International Conference on Machine Learning*, pp. 767–776, 2015.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. arxiv 2016. *arXiv preprint arXiv:1603.06560*, 2016.

Locatelli, A., Gutzeit, M., and Carpentier, A. An optimal algorithm for the thresholding bandit problem. In *International Conference on Machine Learning*, pp. 1690–1698, 2016.

Siegmund, D. *Sequential Analysis: Tests and Confidence Intervals*. Springer Science & Business Media, 1985.

Simchowitz, M., Jamieson, K., and Recht, B. Best-of-k-bandits. In *Conference on Learning Theory*, pp. 1440–1489, 2016.

Szepesvári, C., Wen, Z., and Ashkan, A. Cascading bandits: Learning to rank in the cascade model. 2015.

Vernade, C., Cappé, O., and Perchet, V. Stochastic bandit models for delayed conversions. In *Conference on Uncertainty in Artificial Intelligence*, 2017.

Wang, Y., Audibert, J.-Y., and Munos, R. Algorithms for infinitely many-armed bandits. In *Advances in Neural Information Processing Systems*, pp. 1729–1736, 2009.