Multi-Armed Bandit Problem

Sunith Suresh, Lin Xiao, Ilan Man and Sanjay Hariharan

May 2, 2016

1 Introduction

The Multi-Armed Bandit Problem:

> Suppose you are faced with NN slot machines (colourfully called multi-armed bandits). Each bandit has an unknown probability of distributing a prize (assume for now the prizes are the same for each bandit, only the probabilities differ). Some bandits are very generous, others not so much. Of course, you don't know what these probabilities are. By only choosing one bandit per round, our task is devise a strategy to maximize our winnings.

The task is complicated by the stochastic nature of the bandits. A suboptimal bandit can return many winnings, purely by chance, which would make us believe that it is a very profitable bandit. Similarly, the best bandit can return many duds. Should we keep trying losers then, or give up?
A more troublesome problem is, if we have a found a bandit that returns pretty good results, do we keep drawing from it to maintain our pretty good score, or do we try other bandits in hopes of finding an even-better bandit? This is the exploration vs. exploitation dilemma.

2 Multi-armed bandit review

In Machine Learning literature, the stochastic multi-armed bandit is defined as:

> Given K machines, each with an unknown probability of yielding a reward, which come from a fixed but unknown distribution parameterized by θ_i, for $i \in (1, ..., K)$, and N total plays, decide which machines to play in order to maximize the total reward or minimize the regret.

Define R_N^* and $\hat{R_N}$ as the actual and expected maximum rewards, respectively, after N plays. Therefore the regret of a given multi-armed bandit algorithm is $R_N^* - \hat{R_N}$. Note that since this is a stochastic problem, we aim to minimize regret in expectation or with high probability.

This paper presents various approaches at minimizing the total regret. Specifically, we compare heuristic approaches to theoretically optimal approaches.

The common theme between the algorithms outlined in this paper are that our objective is to select the optimal machine to play at each time step, given that we want to maximize future rewards. The approach to selecting which machine to play is known as the *policy* or acquisition function, denoted as $U(\alpha_i, \beta_i)$. This function balances the trade off between selecting the best machine based on previous plays and the possibility of a better machine that hasn't been selected yet. This trade off is the *exploration/exploitation* dilemma.

2.1 ϵ-greedy

- Sunith

2.2 Upper Confidence Bounds

The upper confidence bound (UCB) family of algorithms simply selects the machine with the largest upper confidence bound at each round. The intuition is this: the more times you play a machine, the tighter the confidence bounds. So, as the number of plays for each machine increases, the uncertainty decreases, and so does the width of the confidence bound.

We want to know with high probability that the true expected payoff of a play μ_i is less than our prescribed upper bound.

The key ingredient here is using Chernoff-Hoeffding Bounds to find the upper bound:

$$\mathrm{P}(Y + a < \mu) \leq e^{-2na^2}$$

UCB1 index-based policy created by the sum of two terms, where the first term is the average reward and the second term is related to the one-sided confidence interval for the average reward according to the Chernoff-Hoeffding bounds, which provides an upper bound on the probability that the sum of random variables deviates from its expected value.

Assuming K machines:

1. Play each arm once
2. Observe rewards r_i, for $i = 1, ..., K$
3. Set $n_i = 1$, for $i = 1, ..., K$
4. set $\bar{\mu_i} = \frac{r_i}{n_i}$
5. For time $t = K + 1, ..., N$:
   (a) Play arm $\hat{i} = \underset{i}{argmax}(\hat{\mu_i} + \sqrt{\frac{2ln(t)}{n_i}})$
   (b) Observe reward r
   (c) $r_{\hat{i}} = r_{\hat{i}} + r$
   (d) $n_{\hat{i}} = n_{\hat{i}} + 1$
   (e) update $\hat{\mu_i} = \frac{r_{\hat{i}}}{n_{\hat{i}}}$

Where $\hat{\mu_i}$ is the average reward obtained from machine i and n_i is the number of times machine i has been played so far.

From our analysis of the Chernoff-Hoeffing bound above we can see that the confidence bound grows with the total number of actions we have taken but shrinks with the number of times we have tried this particular action. This ensures each action is tried infinitely often but still balances exploration and exploitation.

The regret for UCB1 grows at a rate of ln n.

3 Bayesian approaches

3.1 Bernoulli Bandit

We formulate a simple bandit scheme involving K Machines and a Bernoulli Reward (0 or 1). We play one of the N machines at every iteration of the game for T iterations. Each Machine has a latent parameter governing its reward. At every step we draw a reward from the selected machine.

Specifically, for each machine, we generate a probability of getting a reward of 1, θ_i. We assume that θ_i is distributed as a Beta. A natural choice for the likelihood of a reward, r_i is a Bernoulli(θ_i). To be clear:

$$\theta_i \sim \mathrm{Beta}(\alpha_i, \beta_i)$$

$$r_i = \text{Reward from Machine i} \sim \text{Bernoulli}(\theta_i)$$

Exploiting the conjugacy of the Beta-Bernoulli model, the posterior distribution of getting a reward from machine$_i$, after playing for T iterations is:

$$\text{Beta}(\alpha_i + R_T, \beta_i + T - R_T)$$

where $R_T = \sum_{t=1}^{T} r_t$
Before beginning to play, we have no prior knowledge about either Machine's propensity to yield a positive reward. Therefore, we initially set $\alpha_i, \beta_i = 1$, which is a common objective prior for the Beta distribution.

3.2 Thompson Sampling: Hueristic approach

The generic algorithm is as follows:

1. For $t = 1, ..., N$
2. U(α_i, β_i) = i_t // select machine i at time t using the policy/acquisition function
3. $r_t \sim$ Bernoulli(θ_i) i // We play Machine i, and get Reward r_t
4. If $r_t = 1 : \alpha_i = \alpha_i + 1$ // if the reward was successful, increase our positive belief in machine i
5. else if $r_t = 0 : \beta_i = \beta_i + 1$ // if we didn't get a reward, increase our negative belief (i.e. failure) in machine i
6. $R_t = R_t + r_t$ // add reward at time t to running total

This algorithm provides a framework for how to think about this problem in a general way. To maximize expected rewards, we must optimally choose the acquisition function U.
Note that we don't real care how accurate we become about inference of the hidden probabilities – for this problem we are more interested in choosing the best bandit (or more accurately, becoming more confident in choosing the best bandit).

3.3 Gaussian Process and Reinforcement Learning

- Lin

- assume knowledge of GPs, so no need to explain what a GP is
- quick overview of how MAB relates to reinforcement learning
- quick overview of how GPs relate to reinforcement learning
- using the mechanics of GP, what the connection between GP and MAB is
- show a chart or two like in the presentation
- this will be multiple sections

4 Empirical Comparisons

4.1 Data set

4.2 Charts

- using the same data set, compare UCB vs. epsilon vs. Thompson vs. gittins vs. GP

5 Conclusion

6 References