# Modeling Marketing with Multi-Arm Bandits

**Sophia Barton**
sophiapb@stanford.edu

**Hasna Rtabi**
hasna@stanford.edu

**Danny Takeuchi**
dtakeuch@stanford.edu

## Abstract

Predicting the best product to advertise to an individual consumer, or demographic group of similar potential buyers, is a ubiquitous task that companies in all types of industries face. We model and solve this task as a multi-arm bandit problem - solved both directly with Bayesian Model Estimation and using the Upper Confidence Bound. We evaluate our algorithms against random models which arbitrarily choose a product to advertise, and found that all our solutions greatly outperform random.

## 1 Introduction

Marketing is a challenging field since businesses must do their best to understand their target audiences in order to select particular products that they believe their customers will purchase, as well as create advertising campaigns around these products. At its core, marketing comes down to making decisions under uncertainty: businesses are uncertain about whether a person will have an affinity towards a product, and they are uncertain about whether someone will purchase the product, even given a predisposed affinity to it. This is because humans are complex, often irrational beings and it is difficult to predict their responses to advertisements.

We model our task in the form of a multi-arm bandit problem, which is a class of problems involving maximization of total reward by making a decision out of multiple possibilities, for some number of time-steps. In the most common formulation of a bandit problem, there are *n* possible 'arms' in a slot machine (i.e. choices to be made), in which each 'arm' *i* has a certain payoff $p_i$ with probability $\theta_i$, and a payoff of 0 with probability $1 - p_i$; the player of this slot machine has *h* pulls, and must balance exploration and exploitation in order to generate the highest overall possible payoff. Since the player has no prior knowledge, he must find the balance between exploring new arms - for which an estimate of its individual payoff is still highly unknown - and exploiting arms for which there is a reasonable belief in yielding a high reward, in the process of reinforcement learning. This problem is an equivalent formulation of an *h*-step finite horizon Markov Decision Process (MDP) with just one state, *n* actions, and an unknown reward function.

## 2 Related Work

Although multi-arm bandit problems have been well-defined since World War II, optimal algorithms to solve these types of problems, as well as evaluation metrics are still an open research question. In "Multi-armed Bandit Algorithms and Empirical Evaluation," authors Vermorel and Mohri provide an evaluation of several multi-armed bandit algorithms [2]. The most naive approach to solving this class of problems is called $\epsilon$-greedy. This strategy consists of choosing a random lever with probability $\epsilon$, otherwise choosing the lever with the highest expected reward (based on rewards observed until that point) with probability $1 - \epsilon$, at each time-step. Another approach they discuss is named $\epsilon$-first, which consists of performing all exploration initially. In other words, for a certain number of rounds, all levers are pulled randomly with probability $1 - \epsilon$, and after this exploration phase, the player chooses the lever at each time step that has the highest expected reward. Interestingly, the authors find that the $\epsilon$-greedy approach, although simple, often yields the highest outcome for a variety of problems.

The paper additionally offers a new strategy to solving these types of problems, which they name Price of Knowledge and Estimated Reward (POKER). This approach balances exploration and exploitation by assigning a price for knowledge gained by pulling a lever, which can also be thought of as an 'exploration bonus.' The attempt is to quantify uncertainty with the same units as the reward. It is also based upon the idea that the price of information of unobserved levers can be estimated by levers that have already been observed. Lastly, it takes into account the horizon, since the exploration phase duration should depend on the number of rounds that a game will be played.

Another family of algorithms under the name of Upper Confidence Bound (UCB) was created in 2002 as a simpler strategy to solve multi-arm bandit problems. In his paper, "Algorithms for the multi-armed bandit problem," the author Kuleshov briefly reviews this approach [1]. He writes that the UCB algorithms are based upon mathematics that 'provide strong theoretical guarantees on the expected regret.' Regret, to be formal, is the incurred loss from time spent learning as opposed to choosing the arm yielding the maximum expected reward at every timestep (which is clearly unknown).

The most basic version - UCB1 - achieves the optimal regret up to a certain multiplicative constant. Another version - UCB1-Tuned - is designed to be sensitive to variance of the payoffs of each arm, which makes this algorithm useful for problems in which there are high values of the variance between arms. Kuleshov also finds through his experiments that UCB algorithms often perform well for bandits with a low number of arms and high reward variances, but are not as optimal as other algorithms when the number of arms, *K*, is large.

## 3 Approach

We tackled our multi-arm bandit problem with two distinct strategies: Bayesian Model Estimation and Upper Confidence Bound.

### 3.1 Bayesian Model Estimation

We first modeled the problem as a multi-armed bandit problem where the agent is the business and each arm is an advertisement to launch for a specific product. The goal of the business is to find the right advertisement to maximize its profit. We start with a uniform prior, represented by the Beta Distribution (1,1). Each 'arm' *i* (product advertisement) begins with one 'win,' $w_i$, and one 'loss,' $l_i$. We used a Beta($w_i$ + 1, $l_i$ + 1) posterior instead of Beta($w_i$ + 100, $l_i$ + 100) for example, because we assume the business has no strong prior belief about popularity of its products. This ensures that each additional piece of information would have a significant effect on the Beta distribution. The posterior probability of winning is computed from the following equation:

$$p_i = P(\text{win}_i|w_i, l_i) = \int_0^1 \theta \text{ x Beta}(\theta|w_i + 1, l_i + 1)d\theta = \frac{w_i+1}{w_i+l_i+2}.$$

We use dynamic programming to find the optimal exploration strategy. Our belief state of the expected payoffs can be represented by $w_1, l_1, ..., w_n, l_n$, where $w_i$ represents that ad *i* was successful (i.e. a 'win'), and $l_i$ that ad $i$ failed (i.e. a 'loss'). These $2n$ numbers are stored in a $n$ x 2 matrix where *n* represent the number of arms, or number of unique products we can advertise to a user. We want to create a policy, $\pi^*$, for each iteration of the matrix. We use $Q^*(w_{1:n}, l_{1:n}, i)$ to represent the expected profit after pulling arm $i$ and acting optimally. We define the optimal utility function, $U^*$, and the optimal policy, $\pi^*$, below:

$$U^*(w_1, l_1, ..., w_n, l_n) = \max_i Q^*(w_1, l_1, ..., w_n, l_n, i)$$

$$\pi^*(w_1, l_1, ..., w_n, l_n) = \text{argmax}_i Q^*(w_1, l_1, ..., w_n, l_n, i)$$

$$Q^*(w_1, \ell_1, \ldots, w_n, \ell_n, i) = \frac{w_i + 1}{w_i + \ell_i + 2}\left(p_i - c_i + U^*(\ldots, w_i + 1, \ell_i, \ldots)\right) + \left(1 - \frac{w_i + 1}{w_i + \ell_i + 2}\right)\left(-c_i + U^*(\ldots, w_i, \ell_i + 1, \ldots)\right) \quad (1)$$

In equation (1) above, $p_i$ and $c_i$ represent the price and cost of the product *i* being advertised, respectively. The first $U^*$ term assumes that the ad is successful and the consumer bought the product which yields a reward of $p_i - c_i$, which is why the $w_i$ term is incremented to represent another 'win.' The second $U^*$ term assumes that the ad fails with a cost of $-c_i$, which is why the corresponding number of losses for ad $i$ is incremented.

Our horizon, $h$, represents the number of ads that the business can run. When $h$ = 0, $U^*(w_1, l_1, ..., w_n, l_n)$ = 0.

### 3.2 Upper Confidence Bound

To expand the model to more products and time-steps, we implemented UCB1, since computing it directly via Bayesian Model Estimation is generally intractable as the horizon and action space grow.

#### 3.2.1 Data

We started by generating all of our data. We generated an array of prices in which each element, `prices`$[i]$, represents the price of product $i$; we also generate an array of costs in which each element `costs`$[i]$ is the cost of product $i$. We made the assumption that we have access to information about which ads were clicked on or not by a group of similar users; this is represented by a matrix which we also generated. Each column in this matrix represents a different ad and each row represents a different time step. The cell value is a 1 if the ad was clicked on by the imaginary user, and 0 otherwise. We decided to set the number of possible products that can be advertised to 15, and the horizon to 1000 because these seemed like realistic values in the context of a single company trying to pick a product to advertise at each time step.

#### 3.2.2 Algorithm

The UCB algorithms are based upon the idea of optimism in the face of uncertainty; in other words, choosing actions as if the unknown mean payoffs for each arm, or choice, is as large as possible based on the observed data. The more uncertainty about an arm or action, the more important it is to explore. This is a functional strategy since either the optimism is justified, which means that the player acts optimally, or the optimism is not justified, meaning that the player chooses an action with the belief that it would yield a high payoff, yet doesn't; the player will eventually learn the true payoff and cease to choose that action in the future. Each action essentially has a corresponding distribution of its reward; the distribution with the highest variance has the most uncertainty.

UCB1, which we chose to implement, first chooses each of the *K* actions once, to provide initial estimations for the payoff of each action (which in our case is profit). Then, at each time step after *K*, the action is chosen which maximizes the following equation:

$$Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

where $N_t(a)$ is the number of times action $a$ has been chosen thus far, and $Q(a)$ is the estimation of the expected utility for action $a$. $Q(a)$ is the action-value function, and the second value is the confidence term. Each time that action $a$ is chosen, the uncertainty regarding its payoff decreases, which makes sense since $N_t(a)$ increases, and is in the denominator, so the uncertainty term decreases. Conversely, each time that an action other than $a$ is chosen, the uncertainty term increases, which makes sense given that $t$ increments (in the numerator), while $N_t(a)$ remains constant. The logarithm ensures that the magnitude of the increases slowly gets smaller over time; actions with low value estimates or that have large $N_t(a)$ values are selected with decreasing frequency.

## 4 Analysis

### 4.1 Evaluation Method

We compare the profits generated by each of our two models to those generated when a random policy is used. If the product advertised is purchased at time step $t$, the profit at that time step is the difference between the price of the product and the cost of the advertisement. If the product

advertised at time step $t$ is not purchased, the profit is zero in the UCB calculation, and the profit is the negative of the cost in the Bayesian Estimation formulation. The total profits is the sum of the profits at each time step.

### 4.2 Bayesian Model Estimation Results

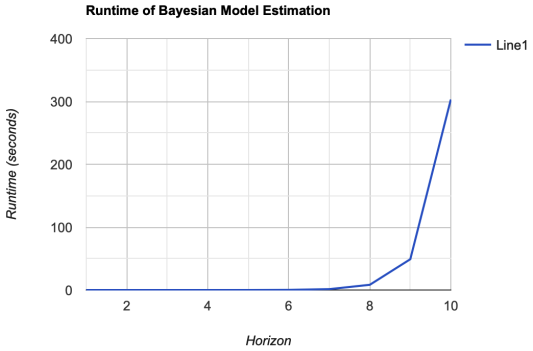
Our toy example consisted of a 3-armed bandit with the following prices and costs:

Prices: [2, 3, 4]
Costs: [1, 1, 1]

We generated the optimal policy, $\pi^*$, and calculated the expected utility when following policy $\pi^*$.

Table 1: Expected Utility of Multi-Armed Bandit Policies

| Horizon | Bayesian Policy | Random Policy |
|---|---|---|
| 1 | 1 | .5 |
| 2 | 2.08 | 1 |
| 3 | 3.25 | 1.5 |
| 4 | 4.42 | 2 |
| 5 | 5.63 | 2.5 |
| 6 | 6.87 | 3 |
| 7 | 8.11 | 3.5 |
| 8 | 9.38 | 4 |
| 9 | 10.66 | 4.5 |
| 10 | 11.95 | 5 |




The following results are as expected. At a horizon of 1, the prior distributions for each arm is Beta(1, 1), and we would pull arm 3. The expected reward is $.5 * (4 - 1) + .5 * (-1) = 1$. Meanwhile, the expected reward of the random policy is $\frac{1}{6} * (2 - 1) + \frac{1}{6} * (3 - 1) + \frac{1}{6} * (4 - 1) + \frac{1}{2} * -1 = .5$. Our policy derived from Bayesian Estimation performs significantly better than a random policy because it emphasizes exploitation. While this policy is optimal, and performs better than an ad-hoc policies, it requires iterating over every possible outcome. The run-time and space-time complexity is exponential over horizon $h$.

### 4.3 UCB1 Results

We created 15 fictitious products, with the following prices and costs generated randomly:

Prices: [94, 9, 2, 50, 10, 3, 71, 13, 34, 27, 97, 99, 92, 24, 13]
Costs: [4, 5, 6, 0, 9, 1, 7, 5, 6, 1, 5, 5, 4, 2, 4]

We also generated the random $1000$x$15$ matrix of 'user clicks.'

The profits generated with our UCB1 algorithm are $\$47616$, which means that UCB1 outperformed the random policy by $159\%$. We also notice that after an initial exploration period, the UCB1 algorithm quickly converges to always targeting the same product, which indicates that this product is the one most likely to be purchased by the user. In our case, the product it converges upon within 18 time-steps is product 11, which it continues to choose for the remaining 982 time-steps.

## 5 Conclusion & Future Work

Both the Bayesian Model and UCB1 significantly outperformed the random policy in choosing which product to target to a user at each time step. The profit generated, which was used to compare the models to the random policy, takes into account both the price of the product and the cost of advertising that product. Depending on the size of the problem, one approach or the other is more appropriate. If the number of products is relatively small, the Bayesian model can be used and if the number of products is large and/or the horizon is lengthier, UCB1 works best.

There are numerous options for future work. The current model can be expanded to more accurately reflect the real world and rely less upon simplifying assumptions. For instance, dependencies between products can be introduced to account for the fact that certain products may be similar to others. More specifically, advertising one product will not only potentially lower the probability that a user will purchase that particular product again in a future time step if they didn't purchase it after receiving an ad for it, but also potentially lower the probabilities of purchase for similar products in the future.

The problem could also be modeled as a POMDP, where the state is a vector of probabilities of purchasing each product. The observation of whether the user purchased the product advertised to them could be made, which is a noisy indication of their probability distribution of buying each product. Similarity between products could be incorporated into the transition function, which would update the belief about the user's probabilities of purchase of not only the product advertised at that time step but also the products similar to it.

## 6 Contributions

We all worked together on coming up with the idea, modeling our task in the given scope, and designing our approaches to solving the task. Danny worked the most on the Bayesian Model while Sophia and Hasna worked the most on UCB. We all wrote the paper together.

## References


[1] Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *CoRR*, abs/1402.6028, 2014.

[2] Joannès Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In João Gama, Rui Camacho, Pavel B. Brazdil, Alípio Mário Jorge, and Luís Torgo, editors, *Machine Learning: ECML 2005*, pages 437–448, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.