PRIMER FOR *Contextual* MULTI ARM BANDITS

Karthick Raja
Researcher @ Philips
Bangalore
karthick11b36@gmail.com

April 12, 2020

ABSTRACT

This technical note reviews the concept of Multi Arm Bandits(MAB) and its application in the field of personalised web advertisements. There is a need for new algorithm in dynamic recommendation system for web services like advertisements, news recommendation and product recommendation. The existing traditional models like collaborative filtering and content based filtering system faces challenges due to dynamically changing advertisement, articles and the traditional methods lack scale-able models for training and deployment. This technote is serves as a primer for understanding the problem formulation of contextual multi arm bandits and basic solution algorithm implemented on a *yeast* toy dataset.

***K*eywords** MultiArmBandits · Contextual multi arm bandits · Reinforcement Learning, online learning

1 Decision making under uncertainty

Humans involve themselves in several forms of decision making on their day to day activities. It can be as simple as choosing restaurants for dinner to periodically investing in companies for profits. we tend to make decision from our past experience., for example on given a choice we can either choose blue stock companies which we know that we can get decent stable returns or choose to invest in risky new stocks which may give better or bad results.

Clearly, the choice here is a trade off between going with the well known option with low volatility or going with a new option which may give better results but we will know only when we invest. We can choose to exploit the known stable returns stocks or to explore new stocks which may give better results. This is a classical ***Exploitation vs Exploration*** trade-off problem

1.1 Exploration vs Exploitation Trade-off

With the same stock investing example, the dilemma comes here because of the partial information about our options. we need to gather enough information to make best overall decisions while keeping the risk under control.

1. With exploitation, we take advantage of the best option we know.
2. With exploration, we take some risk to collect information about unknown options.

We need an algorithm (strategy) which guides us on exploiting the existing best solution but also explore new options such that the overall returns are higher. Also several exploration trails might fail, which warns us about the chosen option and reduce the chance of exploring the same option in future.

2 Multi Arm Bandits

The term "multi-armed bandit" comes from a hypothetical experiment where a person must choose between multiple actions (i.e. slot machines, the "one-armed bandits"), each with an unknown payout. The goal is to determine the best or most profitable outcome through a series of choices. At the beginning of the experiment, when odds and payouts are unknown, the gambler must determine which machine to pull, in which order and how many times. This is the

"multi-armed bandit problem". In theory, multi-armed bandits should produce faster results since there is no need to wait for a single winning variation.A step by step explanation of a Multi arm bandit.

1. Imagine a gambler standing in front of a row of these machines.
2. Each machine provides different payoff and gambler has to decide which machine's arm to pull.
3. Gambler can continue playing a machine they have already played and know gives good payoff (exploitation).
4. Or they can try to find a different machine which gives even better payoff (exploration).
5. Gambler has to optimize exploration and exploitation to receive maximum payoff.
6. The random reward obtained from playing an machine repeatedly are i.i.d. and independent of the plays of the other machines.
7. The reward is partially observed immediately after playing the machines. we can observe only the reward of the chosen arm.

3 Problem Formulation

The problem of personlised ads can be naturally modeled as a multi-armed bandit problem with or without context information. Here the context means information about the user or the advertisement/article

3.1 Classical Multi arm Bandit Formulation

Formally, a classical-bandit algorithm A proceeds in discrete trails $t = 1, 2, 3..$ In trail t

1. The algorithm observes the current user u_t and a set of A_t of arms or actions for $a \in A_t$.
2. Based on observed payoffs in previous trials, A chooses an arm $a_t \in A_t$, and receives payoff r_t, a_t whose expectation depends on both the user u_t and the arm a_t.
3. The algorithm then improves its arm-selection strategy with the new observation, (a_t, r_{t,a_t}).It is important to emphasize here that no feedback is observed for $unchosen$ arms $a \neq a_t$.

3.2 Contextual Multi arm Bandit Formulation

Formally, a contextual-bandit algorithm A proceeds in discrete trails $t = 1, 2, 3..$ In trail t

1. The algorithm observes the current user u_t and a set of A_t of arms or actions together with their feature vectors $X_{t,a}$ for $a \in A_t$. The vector X_t,a summarizes information of both the user u_t and arm a, and will be referred to as the $context$.
2. Based on observed payoffs in previous trials, A chooses an arm $a_t \in A_t$, and receives payoff r_t, a_t whose expectation depends on both the user u_t and the arm a_t.
3. The algorithm then improves its arm-selection strategy with the new observation, $(x_{t,a_t}, a_t, r_{t,a_t})$.It is important to emphasize here that no feedback is observed for $unchosen$ arms $a \neq a_t$.

The above formulations are simplified versions of Markov Decision Process (MDP) with no states.

In the process above, the total trial payoff(T) of Algorithm A is defined as $\sum_{t=1}^{T} r_{t,a}$. Similarly, we define the optimal expected Trail pay T as $\mathrm{E}[\sum_{t=1}^{T} r_{t,a_t*}]$, where a_t^* is the arm with maximum expected payoff at trial t. Our goal is to design A so that the expected total payoff above is maximized.

In the context of product recommendation, we can view ads in the pool as arms. When a presented article is clicked, a payoff of 1 is incurred; otherwise, the payoff is 0. With this definition of payoff, the expected payoff of an article is precisely its click through rate (CTR), and choosing an article with maximum CTR is equivalent to maximizing the expected number of clicks from users, which in turn is the same as maximizing the total expected payoff in our bandit formulation.

4 Algorithms

As discussed in the literature reviews., the algorithm from the stochastic bandits have been implemented. These algorithms are discussed below with the implementation, benefits and shortcomings.

4.1 ϵ Greedy Algorithm

Algorithm 1 ϵ Greedy algorithm

Require ϵ Set $N = 0_k, Q = 0_k$. ▷ Number of selections, Expected reward for each arm
for $t = 1, 2, \ldots,$ **do**
  R = Generate Random [0,1]
  if R **then** > ϵ
    $a_t := \arg\max_i Q$ and Observe reward r_t
  a_t = Randomly pick any arm with prob $(1/n_arms)$ and Observe reward r_t
  Update$N[a_t] = N[a_t] + 1$.
  Update $Q[a_t] * ((N[a_t] - 1)/N[a_t]) + r_t/n$.

Greedy Algorithm can be defined as the algorithm that picks the best currently available option without taking into consideration the long-term effect of that decision, which may happen to be a suboptimal decision.Given that, we can define epsilon-Greedy Algorithm as a Greedy Algorithm that adds some randomness when deciding between options: Instead of picking always the best available option, randomly explore other options with a probability = ϵ or pick the best option with a probability = $1 - \epsilon$. Therefore, we can add randomness to the algorithm by increasing ϵ, which will make the algorithm explores other options more frequently. Additionally, ϵ is a hyper-parameter that needs to be tuned based on the experiment, i.e. there is no value that works best on all experiments.As a result, the probability of selecting any option randomly if we have N options is ϵ $1/N$; however, the probability of selecting the best option is $1 - \epsilon$.

4.2 Upper Confidence Bound Algorithm

Algorithm 2 Upper Confidence Bound Algorithm

Set $N = 0_k, R = 0_k$. ▷ Number of selections, Sum of rewards for each arm
Play each arm once
for $t = 1, 2, \ldots,$ **do**
  for $i = 1, 2, \ldots, a$ **do**
    $reward_i = R[i]/N[i]; \Delta_i = \sqrt{2 \ln t / N[i]}$
    $Q[i] = reward_i + \Delta_i$
  Play arm $a_t := \arg\max_i Qandobserverewardr_t$.
  Update $N[a_t] = N[a_t] + 1$.
  Update $R[a_t] = R[a_t] + r_t$.

At the start of the campaign, we don't know what is the best arm or ad. So we cannot distinguish or discriminate any arm or ad. So the UCB algorithm assumes they all have the same observed average value. Then the algorithm creates confidence bound for each arm or ad. So it randomly picks any of the arms or ads. Then two things can happen- the user clicks the ad or the arm gives reward or does not. Let's say the ad did not have a click or the arm was a failure. So the observed average of the ad or arm will go down. And the confidence bound will also go down. If it had a click, the observed average would go up and the confidence bound also goes up. By exploiting the best one we are decreasing the confidence bound. As we are adding more and more samples, the probability of other arms or ad doing well is also going up. This is the fundamental concept of UCB.

Based on the algorithm 2 as explained above, Each time an arm a_t is selected, the uncertainty is presumably reduced: $N[a_t]$ increments, and, as it appears in the denominator, the uncertainty term decreases. On the other hand, each time an action other than a_t is selected, t increases, but $N[a_t]$ does not; because t appears in the numerator, the uncertainty estimate increases. The use of the natural logarithm means that the increases get smaller over time; all actions will eventually be selected, but actions with lower value estimates, or that have already been selected frequently, will be selected with decreasing frequency over time. This will ultimately lead to the optimal action being selected repeatedly in the end.

4.3 Thompson sampling

Thompson Sampling is one of the oldest heuristics to solve the Multi-Armed Bandit problem. This is a probabilistic algorithm based on Bayesian ideas. It is called sampling because it picks random samples from a probability distribution for each arm. This could be defined as a Beta Bernoulli sampler. Though Thompson sampling can be generalized to

Algorithm 3 General Framework of Thompson Sampling

Define $\mathcal{D} = \{\}$
for $t = 1, ..., T$ **do**
  Receive context x_t
  Draw θ_t from posterior distribution $P(\theta|\mathcal{D})$
  Select arm $a_t = *argmax_a \mathrm{E}(r|x_t, a, \theta_t)$
  Receive reward r_t
  $\mathcal{D} = \mathcal{D} \cup \{x_t, a_t, r_t\}$

sample from any arbitrary distributions. But the Beta Bernoulli version of Thompson sampling is more intuitive and is actually the best options for many problems in practice.

This algorithm actually makes an auxiliary mechanism to solve the problem, that is, it will not create the machines at every round rather it will create the possible ways these machines could be recreated. With each trial, each takes a sample which has the highest distribution. And with each trial, the distribution will be changed. The distribution will get narrower as we have some information. After a huge number of rounds, we will get the narrowest distribution which we will take as the final outcome. Algorithm 4 explains thompson sampling for bernouli bandits where the prior is modelled using beta distribution, best choice for bernouli reward.

Algorithm 4 Thompson Sampling for Bernouli bandits

Require α, β prior parameters of a beta distribution
Define $S_i = 0, F_i = 0$ ▷ Success and Failure counters
for $t = 1, ..., T$ **do**
  for $i = 1, 2..., K$ **do**
  Select arm $a_t = *argmax_i \theta_i$ and observe reward r
  if r = 1 **then**
    update $S_{a_t} = S_{a_t} + 1$
  else$F_{a_t} = F_{a_t} + 1$

we have two assumptions, first, we say that each ad i gets a reward from a Bernoulli distribution which is the probability of success. You can picture this parameter by showing this ad to a million user and the parameter will be interpreted as the number of times the outcomes were ones divided by the number of ads. Basically, it is the probability of getting ones. The second assumption is that we have a uniform distribution which is a prior distribution, then we have the posterior distribution from the Bayes rule that is the probability of success given the rewards after the round n. By doing this Bayes rule we get the beta distribution here. In Thompson sampling, these random draws mean nothing but the probability of success, because the maximum of these random draws approximating the highest probability of success. And this is the idea behind Thompson sampling. By taking the highest of these draws we are maximizing the probability of success for each of the ads. And this highest probability corresponds to each ad at every round.

4.4 Upper Confidence with Linear Payoffs

In this algorithm, It is assumed that the expected payoff of an arm a is linear in its d-dimensional feature $\vec{x}_{t,a}$ with some unknown coefficient vector θ_a^*; namely, for all t,

$$\mathrm{E}[r_{t,a}|\vec{x}_{t,a}] = \vec{x}_{t,a}^T \theta_a^*. \tag{1}$$

This model is called *disjoint* since the parameters are not shared among different arms. Let $\vec{D}_a$ be a design matrix of dimension $m \times d$ at trial t, whose rows correspond to m training inputs (e.g, m contexts that are observed previously for article a), and $\vec{b}_a \in R^m$ be the corresponding response vector (e.g,) the corresponding m click/no-click user feedback). Applying ridge regression to the training data $(D_a, \vec{c}_a)$ gives an estimate of the coefficients:

$$\hat{\theta}_a = (D_a^T D_a + I_d)^{-1} D_a \vec{c}_a, \tag{2}$$

where I_d is the $d \times d$ identity matrix. When components in $\vec{c}_a$ are independent conditioned on corresponding rows in D_a, it can be shown [**?**] that, with probability at least $1 - \delta$,

$$\vec{x}_{t,a}^T \hat{\theta}_a - \mathrm{E}[r_{t,a}|\vec{x}_{t,a}] \leq \alpha \sqrt{\vec{x}_{t,a}^T (D_a^T D_a + I_d)^{-1} \vec{x}_{t,a}} \tag{3}$$

for any $\delta > 0$ and $\vec{x}_{t,a} \in R^d$, where $\alpha = 1 + \sqrt{\ln(2/\delta)/2}$ is a constant. In other words, the inequality above gives a reasonably tight UCB for the expected payoff of arm a, from which a UCB-type arm-selection strategy can be derived: at each trial t, choose

$$a_t def = \arg\max_{a \in A_t} \left(\vec{x}_{t,a}^T \hat{\theta}_a + \alpha \sqrt{\vec{x}_{t,a}^T A_a^{-1} \vec{x}_{t,a}} \right), \tag{4}$$

where $A_a = D_a^T D_a + I_d$.

The confidence interval in eqn(3.3) may be motivated and derived from other principles. For instance, ridge regression can also be interpreted as a Bayesian point estimate, where the posterior distribution of the coefficient vector, denoted as $p(\theta_a)$, is Gaussian with mean $\hat{\theta}_a$ and covariance A_a^{-1}. Given the current model, the predictive variance of the expected payoff $\vec{x}_{t,a}^T \theta_a^*$ is evaluated as $\vec{x}_{t,a}^T \vec{A}_a^{-1} \vec{x}_{t,a}$, and then $\sqrt{\vec{x}_{t,a}^T \vec{A}_a^{-1} \vec{x}_{t,a}}$ becomes the standard

Algorithm 5 LinUCB

Require: $\alpha > 0, \lambda > 0$
  $A = \lambda \mathrm{I}_d$
  $b = 0_d$
  for t=1, 2, ..., T **do**
    $\theta_t = A^{-1} b$
    Observe features of all K arms $a \in \mathcal{A}_t : x_{t,a} \in \mathrm{R}^d$
    for a=1, 2, ... K **do**
      $s_{t,a} = x_{t,a}^\top \theta_t + \alpha \sqrt{x_{t,a}^\top A_t^{-1} x_{t,a}}$
    Choose arm $a_t = *argmax_a s_{t,a}$, break ties arbitrarily
    Receive reward $r_t \in [0, 1]$
    $A = A + x_{t,a} x_{t,a}^\top$
    $b = b + x_{t,a} r_t$

4.5 Thompson Sampling with Linear Payoffs

We use Gaussian likelihood function and Gaussian prior to design our version of Thompson Sampling algorithm. More precisely, suppose that the likelihood of reward $r_t(t)$ at time t, given context $b_i(t)$ and parameter μ, were given by the pdf of the gaussian distribution $\mathcal{N}(b_i(t)^T \mu, \upsilon^2)$. Here $\upsilon = R\sqrt{24/\epsilon d \ln(1/\delta)}$ with $\epsilon \in [0, 1]$ which parameterizes our algorithm.

$$B(t) = I_d + \Sigma b_{a(t)}(t) b_{a(t)}(t)^T \tag{5}$$

$$\hat{\mu(t)} = B(t)^{-1} + \Sigma b_{a(t)}(t) r_{a(t)}) \tag{6}$$

Then, if the prior for μ at time t is given by $\mathcal{N}(b_i(t)^T \mu, \upsilon^2 B(t)^{-1})$, it is easy to compute the posterior distribution at time $t + 1$,

$$Pr(\hat{\mu} | r_i(t)) \propto Pr(r_i(t) | \hat{\mu}) Pr(\hat{\mu}) \tag{7}$$

as $\mathcal{N}(b_i(t+1)^T \mu, \upsilon^2 B(t+1)^{-1})$ In this Thompson Sampling algorithm, at every time step t, we will simply generate $\hat{\mu(t)}$) from the distribution $\mathcal{N}(b_i(t)^T \mu, \upsilon^2 B(t)^{-1})$ and play the arm i that maximizes $b_i(t)^T \mu(t)$.

We emphasize that the Gaussian priors and the Gaussian likelihood model for rewards are only used above to design the Thompson Sampling algorithm for contextual bandits. The analysis of the algorithm allows these models to be completely unrelated to the actual reward distribution.

Algorithm 6 Thompson Sampling for Contextual bandits

Set $B = I_d, \hat{\mu} = 0_d, f = 0_d$.
for $t = 1, 2, \ldots,$ **do**
  Sample $\tilde{\mu}(t)$ from distribution $\mathcal{N}(\hat{\mu}(t), \sigma^2 B(t)^{-1})$.
  Play arm $a(t) := \arg\max_i b_i(t)^T \tilde{\mu}(t)$, and observe reward $r_{a(t)}(t)$.
  Update $B = B + b_{a(t)}(t) b_{a(t)}(t)^T$, $f = f + b_{a(t)}(t) r_t$, $\hat{\mu} = B^{-1} f$.

References

[1] Chu, Wei, et al A Contextual-Bandit Approach to Personalized News Article Recommendation In *https://arxiv.org/pdf/1003.0146.pdf*

[2] Weng, Lilian The Multi-Armed Bandit Problem and Its Solutions In *lilianweng.github.io/lil-log*