Contextual Bandits for Recommendation System

KrishnaKant Singh
Indian Institute of Technology
Hyderabad,India
cs15mtech11007@iith.ac.in

Abhishek Jain
Indian Institute of Technology
hyderabad,India
cs16mtech11001@iith.ac.in

Varun Mishra
Indian Institute of Technology
Hyderabad,India
cs16mtech11018@iith.ac.in

1. INTRODUCTION

The problem of displaying advertisement,news articles,A/B testing of content on websites,recommendation systems are very important and lucrative problems for big and small corporations.All these problems can be cast as an exploration-exploitation tradeoff problem. For example, in medical clinical trials, we want the best treatment for a patient, but we also want to test new drugs that can perform better than the existing drugs. Multi-armed bandits are well suited for these types of problem.

Multi Armed Bandit:The multi-armed bandit is a well studied problem in the domain of Reinforcement Learning.The MAB problem can best be described as a repeated game with a single player choosing in each round one of the n-arms. In each round the player is rewarded with some points based on the arm he pulls. The interesting part of the problem is that player can only observe the reward on the arm he chooses not all the arms. This essentially creates a exploration-exploitation tradeoff. A greedy player would always choose to exploit the best arm seeing optimal reward in the near future. But a optimal strategy would be to mix both exploitation of the best arm and also exploring the others arms.
A major problem with aforementioned method is that these methods do not take into account the player's or arms attributed into account.For example it does not make sense to show a young user advertisements about retirement plans.

Contextual Bandits:Personalised news recommendation and similar problems that involve taking context of user or items into the problem are more suited to Contextual Bandits. In CB in addition to having information about rewards on the arms,we at each round are presented with context which is taken into account when the next decision is made.

Contextual Bandits are very interesting topic[**?**][**?**][**?**] and of late a lot of work has gone which explore different forms of rewards(cumulative rewards),time varying rewards etc.In our work, we try to first explore the existing algorithms for MAB and CB and find the performance on the yahoo today module news. Further, we will like to explore how an ensemble of CAB algorithm improves the performance of the system based on average CTR.

© 2019 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Keywords

Contextual Bandits; Personalization; Recommendation System

2. WORK DONE

- Implementaion of Non-Contextual Algorithms:UCB,Epsilon-Greedy,Thompson implemented[]
- Implemented Simulator for Small Scale Online Evaluation
- Implementation of Contextual Algorithms:Lin-UCB[**?**]
- Implementation of Off-line Policy Evaluator[**?**]
- Implementation of Epoch-Greedy Algorithm[**?**]
- Implementation of Exp-3 and Exp-4 algorithm
- Implementation of Contextual Thomspon Algorithm
- Implementation of Ensembling in CMAB(HyperTS Algorithm)
- Implementation of a frame work for evaluation of the CMAB algorithms
- Contributing to CMAB libraries with code for Ensembling in CMAB
- Using CMAB algorithms for caching in Information Centric Networking

3. METHODOLOGY

Contextual Armed Bandit can formally be defined as
The algo proceeds in discrete trails t = 1,2,3... In trial t :

- The algorithm observes the current r u_t and a set $\mathcal{A}_t$ of arms or actions together with a feature vectors $x_{t,a}$ for $a \in \mathcal{A}_t$ The vector $x_{t,a}$ summarizes information of both the user u_t and arm a, and will be referred to as the context.
- Based on observed payoffs in previous trials, A chooses an arm $a_t \in A_t$, and receives payoff r_t,at whose expectation depends on both the user u_t and the arm a_t.

- 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 that hat no feedback (namely, the payoff $r_{t,a}$) is observed for unchosen arms $a \neq a_t$.

The total T -trial payoff of A is defined as $\sum_{t=1}^{T} r_{t,at}$. Similarly, we define the optimal expected T -trial payoff as $E(\sum_{t=1}^{T} r_{t,a_t^*})$, 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 article recommendation, we may view articles 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 clickthrough 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.

3.1 Existing Algorithms

All the existing algorithms try to maximize the average reward or the CTR value.A fundamental problem with all of the existing algorithm is the tradeoff between exploitation and exploration. We list here a few algorithms that we used to solve the contextual bandit problem .

- Random: it randomly selects an arm to pull.
- ϵ-greedy (ϵ): it randomly selects an arm with probability ϵ and selects the arm of the largest predicted reward with probability 1 âĹŠ ϵ.
- UCB : Rather than just exploring any of the arms we have a confidence bound for each of the arms. The variance is denoted by $\sqrt{\frac{n}{n_j}}$ where n is the total number of actions and n_j is the total number of action for the arm j.
- LinUCB (α): In each trial, it pulls the arm of the largest score, which is a linear combination of the mean and standard deviation of the predicted reward. Given a context x, the score is $\mu^T x + \alpha\sqrt{x^T \sigma 1 x}$, where μ^T and σ^-1 are the estimated mean and covariance of the posterior distribution, and α is a predefined parameter for controlling the balance of exploration and exploitation.
- Epoch-greedy : It defines an epoch with length L, in which a one-step exploration is first performed, and the rest trials in the epoch are used for exploitation
- Exp-3 :Exp3 stands for Exponential-weight algorithm for Exploration and Exploitation. It works by maintaining a list of weights for each of the actions, using these weights to decide randomly which action to take next, and increasing (decreasing) the relevant weights when a payoff is good (bad). We further introduce an egalitarianism factor$\gamma \in [0, 1]$which tunes the desire to pick an action uniformly at random. That is, if $\gamma = 1$, the weights have no effect on the choices at any step.
- Exp-4 : Exp4 is an extension to the Exp3 algorithm in which the experts take into account the contextual information. A expert is nothing but a supervised learning algorithm eg logistic regression where we use the predicted probabilities for the action set as output. In Exp4 algorithm the weighing is also modeled using another probability distribution.
- Ensembling : Ensemble of Bandit algorithm using a MAB algorithm for choosing the best policy at some time t. This policy selects the best arm according to its estimates of rewards of the best arms. This is shown to be equivalent to using a monte carlo estimates for rewards for the policies.

4. DATASET

We end up using 2 sets of data one the Yahoo today module dataset. This dataset consists of 45,811,883 distinct visit events where a user is shown an article in the feature box (drawn from a pool of articles picked by human editors) and the click (or lack of click) is recorded. In relation to the MAB problem, for each event t, we consider each article as an arm (a_t), each pool of potential articles that can be shown as the set of A_t arms for the round, and each user as distinct user u_t. Click events provide binary rewards (0 or 1 for no click or click, respectively) and represented by r_t,a for each article in each round. The context vector x_t,a and its associated features vectors.
All user are distinct which mitigates the possibility of dependent observation We have over 45 million distinct users.Each user is analysed using cojoined analysis and the set of 1000 raw feature vectors are transformed into 6 dimensional vectors. There are 271 articles observed within the course of the 10 days covered by the dataset. A random article is drawn from a pool of around 20 available articles for each event (varying from 19 to 25, with new articles introduced and old articles retired throughout the period observed. We ultimately observe 433 distinct sets of articles, A_t, shown to users throughout the 10 days.The articles are also converted into 6-dimensional vectors using conjoint analysis. Together, the user and article feature vectors give us a R^{36} vector for each user/article pair, x_t,as this is obtained from the outer product of the user and article R^6 feature vectors, which are then flattened into R^{36}
Due to the constraint of resources and time we at present test our work using yahoo test dataset of about 100 users from the expo challenge.

5. EVALUATION

Compared to machine learning algorithm more specifically supervised learning it is much more difficult to evaluate bandit algorithm.In our application of news article recommendation, for example, each user visit results in the following information stored in the log: user information, the displayed news article, and user feedback (click or not). When using data of this form to evaluate a bandit algorithm offline, we will not have user feedback if the algorithm recommends a different news article than the one stored in the log. This is also called the partial label problem. A good way to do is using real live data but these methods are generally very expensive.Rather 2 ways are popular in bandit Literature

- Simulation Based : We construct a monte Carlo simulator based on synthetic data where the rewards are Bernoulli,gaussian distributed etc. We use these to test our implementations of the bandit algorithms.These simulations are suitable only for a small dataset due computation and time constraints.

H

Figure 1: Off-line evaluation

Figure 3: Contextual Random VS LinUCB

- Offline Evaluation : We use the method of [**?**] to construct a offline evaluation model.A key assumption of this model is individual events are i.i.d., and that the logging policy chose each arm at each time step uniformly at random. Formally, algorithm A is a (possibly randomized) mapping for selecting the arm at at time t based on the history h_{t1} of t âĹŠ 1 preceding events together with the current context.The method takes as input a bandit algorithm A.We then step through the stream of logged events one by one. If, given the current history h_{t1}, it happens that the policy A chooses the same arm a as the one that was selected by the logging policy,then the event is retained (that is, added to the history), and the total reward $\hat{G}$ updated. Otherwise,if the policy A selects a different arm from the one that was taken by the logging policy, then the event is entirely ignored, and the algorithm proceeds to the next event without any change in its state.

6. RESULTS AND ANALYSIS

We firstly evaluate our algorithms in a non-contextual way for the aforementioned yahoo test dataset.We plot the best arm selection probabilities for random,epsilon-greedy and softmax algorithm.We observe that clearly epsilon-greedy is the winner out of them which also agrees with the literature on bandit algorithms.

Figure 2: Comparison of MAB algos

We next tested the contextual version of the random algorithm with LinUCB for yahoo test dataset with respect to best arm selection.This was as expected in the literature. The rest of the algorithm results are evaluated on the 5% dataset of the Yahoo Today Module dataset.Figure 4 show that the best epsilon matches exploration and exploitation ie is either too much exploring or too much exploiting gives worse performance.

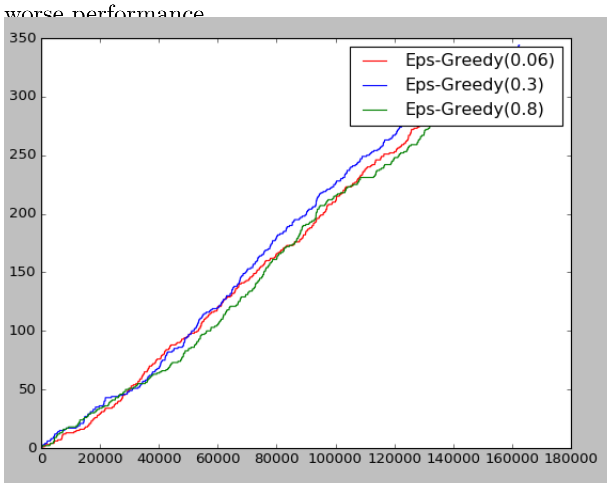

Figure 4: Epsilon Greedy for different values of epsilon.

We next evaluate the performance of all the non contextual algorithms for the dataset using the offline policy evaluation method. Are results clearly show the best performing non contextual information is given by thompson algorithm and exp3 algorithms which is consistent with literature.

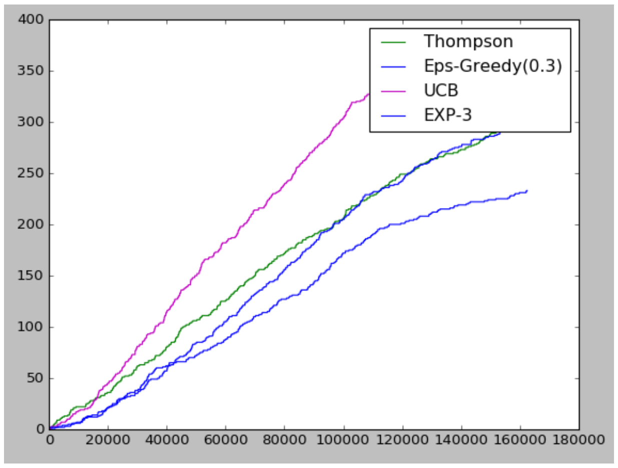

Figure 5: Non-Contextual Performance evaluation

We the evaluate our implementation of the LinUCB algorithm which shows that our implementation does perform better than all other non contextual algorithms.

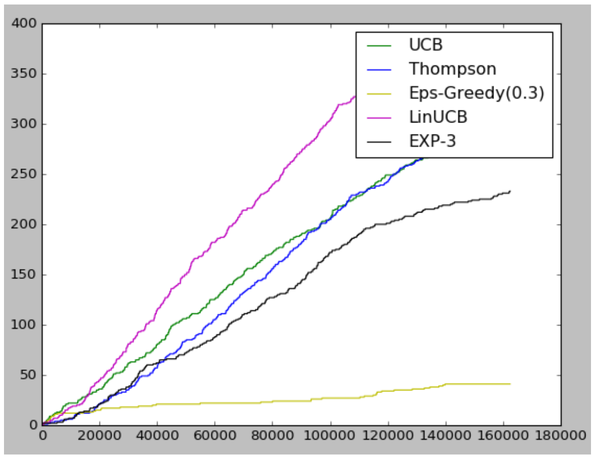

Figure 6: Comparison of LinUCB vs All other non contextual algo

We next evaluate our 2 contextual algorithms as we found out that our epoch greedy and Thompson contextual algorithm were performing poorly on offline policy evaluator.We can clearly see that LinUCB is much better.

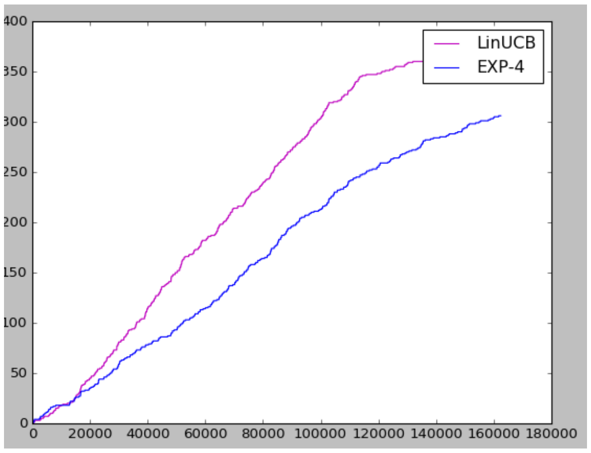

Figure 7: LinUCB vs Exp4

7. APPLICATIONS TO CACHING

We next show a implementation of the MAB with caching in ICN. To the best of our knowledge there is no previous work done with respect to caching in ICN with the use of MAB. We use the ideas of communities with caching in ICN to improve our algorithm performance with respect to other caching strategies like LRU,LFU and other caching strategies in ICN. We use the Yahoo today module for generation of the request patterns for the edge routers. This request model is also unique to our work, no previous work has used a real request pattern. We first describe our algorithm for generation of the request patterns and then we describe are caching algorithm.The intuition for our work comes from the fact that there many users in a community have similar interest and caching those resources that are important for community. We cache based on 3 types of communities local,semi-local and Global. We cluster based on the basis of feature vector of the users.

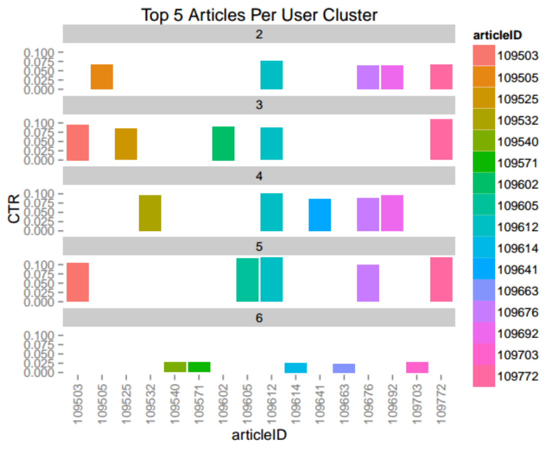

Figure 8: Top5 articles per cluster(Jwang)

The next figure shows that for a cluster only some items are popular. Or in a more technically the popularity distribution is long tailed which agrees with Literature.

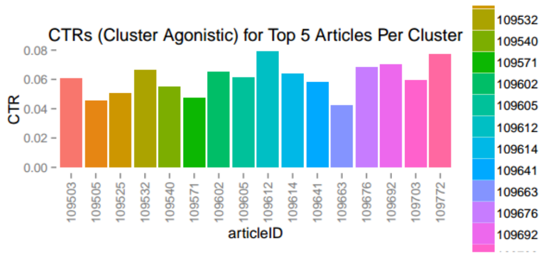

Figure 9: Article CTR for every cluster(Jwang)

Request Pattern

- Input:Set of Edge routers and logged data
- cluster $= L_{\infty}|X|$ where X is the feature vector
- Distribute some portion of the clusters to each of the edge routers(This can be random but can also follow a heavy tailed distribution)
- Using logged data and user id from the cluster form a resource graph. In this resource graph nodes are the resources and there are weighted edges indicating the inverse probability of request for resource A following resource B.
- Using Markov random walk find the steady state probability values for the items in the resource graph.This gives us the most popular items in the edge router locally.

Creating Communities

- Between each edge router create communities which can based on the ratio of particular communities.
- Between each edge routers the communities can also created based upon steady state probability values of resources.Two edge routers are in the same community if they share more than λ popular resources **Caching Strategy**
- We create 3 level cache at each router
  - Most Popular item from the resource graph created at edge router.
  - Most popular item in the community. These are the items that popular in the edge routers $\forall j$ of the community where j $\neq$ i
  - Then we cache the most popular items in the whole universe of the edge routers

  The most popular items are decided by multi armed bandit we are using.

8. RESULTS

We show here the results of simulation for our caching.Based on the caching strategy of UCB and Random caching strategy.We used 3 parameters for distribution of the users to edge routers.The simulation set up is based on the Dutch Telecom graph where there are 23 edge routers. We first consider the results for random MAB.

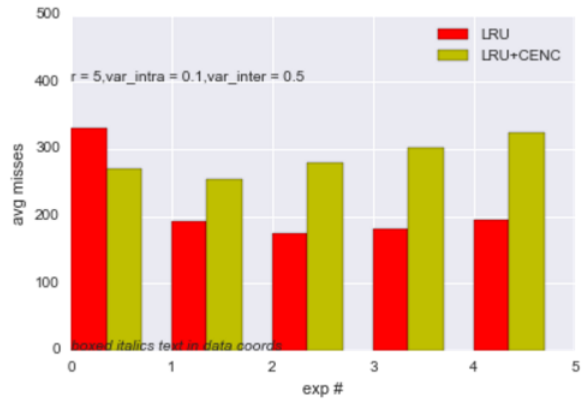

Figure 10: Comparison of cache misses for LRU using Random MAB

We next consider the results for UCB Multi armed bandit. This results shows that our algorithm performs much better than the existing baseline model of LRU caching in ICN.

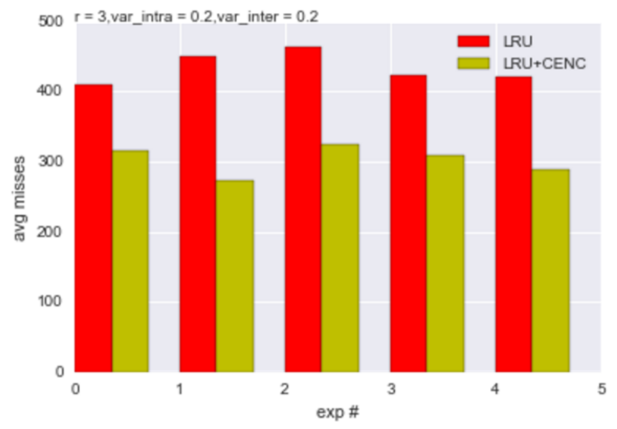

Figure 11: Comparison of cache misses for LRU and Using UCB MAB

9. FUTURE WORK AND DISCUSSION

- Exploring time dependency of rewards in Multi Armed Bandits.
- Using context of content for caching of the context
- Expanding the ideas to IOT.
- Stochastic extension to these ideas.

10. OUR CONTRIBUTION AND KEY TAKEAWAYS

- Providing a framework for offline evaluation of the methods.
- Contributing to open source projects like strata.
- Providing implementation of some new Multi armed bandits algorithm for which implementation our not available
- Showing a real application of MAB in a very important application of caching.