IND E 599 Project Report

Dynamic Pricing BwK Problem and Reinforcement Learning

Lun Jiang

1 Abstract

Dynamic pricing with limited supply is a typical bandits with knapsacks (BwK) problem, which has an increasing popularity in areas like machine learning and operation research since recent years. In this course project, a basic version of dynamic pricing with two products under single global constrain was studied. Traditional multi-armed bandit algorithms, classical BwK algorithm, and reinforcement learning algorithms are compared with each other in exploring optimal policies and solving the problem.

2 Problem statement

Bandits with knapsacks (BwK) is a general framework which models knapsack problems in analogy with multi-armed bandit (MAB) problems.[1]. BwK problems share many common features with MAB, whereas the principal distinctive feature of BwK is having global constrains (aka. Knapsack size) in addition to time step limit. Typical real-world BwK problems include dynamic pricing, pay-per-click ads, repeated auction, repeated bidding, etc, all of which belong to the family of knapsack problems.[2] Different with traditional MAB, the goal of solving a BwK problem is not to find the best arm, which may not even exist, but to obtain the optimal deterministic or stochastic policy of arm choosing and resources allocation.

3 Model description

In this project, a simplified dynamic pricing with limited supply model was constructed as follows:

There were two product `A` and `B` for sale in time rounds `T`, and a group of buyers being offered the two products sequentially. The offering process followed a bandits setting, where a single buyer could only be offered one product at one price per each round, whereas in a semi-bandits setting, multiple products could be offered at the same time.

Each buyer had a value `v` hidden in mind called *valuation*, the distribution of which was unknown to the seller. In each round, if the offered price `p` were lower than the buyer's valuation for this product, then the product would be sold at the offered price, otherwise not. Each offer was described as a (product, price) pair named *atom*[3], and the total number of available atoms was `n`, analogous to all available arms in a MAB problem.

A knapsack condition was applied to the seller's side. Certain resource was finite (e.g. supply or inventory), inducing a global constrain that the total number of products could be sold was limited, depicted as the budget `B`. Every time a product was sold, 1 unit of resource was consumed. The entire process terminated when total time steps `T` or budget limit `B` was reached. Hence, in each round the outcome vector should be a (reward, consumption) pair, determined by whether the buyer accepted the offered atom or not.

$$\text{outcome vector} = \begin{cases} (p, 1) & \text{if price } \texttt{p} \text{ is accepted} \\ (0, 0) & \text{otherwise} \end{cases}$$

4 Solution methods

4.1 MAB algorithm

The problem stated above is naturally episodic and resembles multi-armed bandit in many ways. However, in general traditional MAB algorithms like epsilon-greedy and Thompson sampling can not solve the BwK problem without modifications.

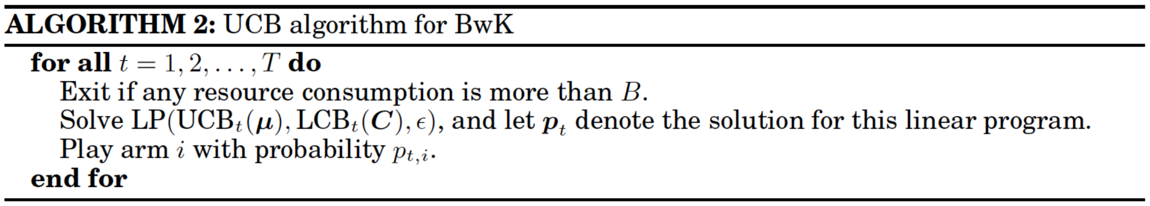
This is because those traditional MAB algorithms choose arm based on the maximum per-round expected reward of each arm. But in a knapsack setting with limited budget `B`, the arm with highest immediate reward may not be the optimal, since it may also consume more resources, thus leads to an early termination with lower total rewards. A special case is when total time step `T` is smaller than budget `B`, in which case the resources could be considered infinite within finite time `T`, and thus the BwK problem reduces to a MAB problem solvable by traditional MAB algorithms.

In this project, epsilon-greedy method was chosen as the representative of MAB algorithms, the performance of which was tested and compared with other methods later in the experiments section.

4.2 BwK algorithm

Since the first study of BwK, a few algorithms have been designed specially to solve this family of problems [1], all of which are focused on optimizing the allocation of resources between arms, thus to maximize total expected rewards rather than per-round expected reward. Here the goal is to obtain the optimal stochastic policy. Since in BwK problems, usually the best fixed-distribution policy performs better than the best fixed-arm policy, and when the probability of a single arm approaches 1, the stochastic policy reduces to a deterministic one.

In this project, a BwK method based on Upper Confidence Bound (UCB) and linear programming was described and tested, the idea of which originates from section 4.2 of [4]:



ALGORITHM 2: UCB algorithm for BwK

```
for all $t = 1, 2, \ldots, T$ do
    Exit if any resource consumption is more than $B$.
    Solve LP($\mathrm{UCB}_t(\boldsymbol{\mu}), \mathrm{LCB}_t(\boldsymbol{C}), \epsilon$), and let $\boldsymbol{p}_t$ denote the solution for this linear program.
    Play arm $i$ with probability $p_{t,i}$.
end for
```

Here the linear programming problem inside the pseudo-code is shown as:

$$\begin{array}{ll} \max_{\boldsymbol{p}\in\Delta_m} & \mathrm{UCB}_t(\boldsymbol{\mu})\cdot\boldsymbol{p} \\ \text{s.t.} & \mathrm{LCB}_t(\boldsymbol{C})\boldsymbol{p} \preceq \frac{B}{T}\mathbf{1} \end{array}$$

The $\boldsymbol{\mu}$ and $\boldsymbol{C}$ vectors denote per-round estimates of reward and consumption. The variable $\boldsymbol{p}$ to be solved is the policy (aka. probabilities to choose each arm) at current time step. Here the upper confidence bound of estimated reward and lower confidence bound of estimated consumption are used instead of the estimated values themselves. The reason is to induce an effort of exploration, which agrees with the philosophical principle of UCB: optimism in the face of uncertainty. There exists theoretical proof that the regret of above algorithm is bounded with probability. Moreover, the LP problem can also be solved with a shrunken constrain set, which is implemented by replacing the B on RHS with B$(1-\epsilon)$, whereas $\epsilon = 0$ was used throughout this project.

4.3 Reinforcement learning

Reinforcement learning methods may also be used here. Since dynamic pricing with limited supply problem is naturally episodic and can easily be modeled as a Markov decision process. In this project, multiple RL algorithms were tested on the dynamic pricing problem and compared with MAB and BwK algorithms. Monte-Carlo method and two Temporal Difference methods: Sarsa and Q-learning, were implemented following [5].

All three RL methods should converge to the optimal state-based epsilon-greedy policies. The major difference between Monte-Carlo and Temporal Difference is that Monte-Carlo updates the Q values of each (state, action) pair only after the completion of each episode, while Temporal Difference methods update the Q values per each step.

5 Experiment results

In this project, the performance of the above methods were tested. Random guessing and fix-arm policy were used as benchmarks. Specially, the UCB algorithm for BwK was tested offline, which means the algorithm first calculated the optimal policy and then evaluated it, rather than updated its current best policy per each step. This is because the Python LP solver *cvxpy* is computationally slow, and the algorithm converges very early (in < 5 episodes). The optimal policy obtained after first few episodes almost kept unchanged later on, so an online policy updating was not necessary if buyer's valuation distribution also kept constant, as in this experiment. However, the performance of MAB and RL methods (epsilon-greedy, Monte-Carlo, Sarsa, Q-learning) were tested in an online setting, in order to show their adaptive learning ability in dynamic pricing problem.

All experiments were carried out under the same configuration: either of the 2 products was offered 3 difference prices [0.25, 0.5, 0.75], thus action space size n = 6. Total time step T = 100 and budget B = 50. Buyer's valuation data for either product was synthesized randomly following an iid normal distribution truncated in the range [0, 1]. The mean value $\mu =$ [0.2, 0.4] respectively, and the standard deviation $\sigma =$ 0.1.

5.1 Offline benchmarks

The results (total rewards) of random guessing, best fixed-arm policy and the optimal stochastic policy obtained by UCB BwK algorithm are presented below. Each rewards data is the average over 1000 assembly averaging trails under the same configuration.

algorithm	**avg. total reward**
random guessing	6.52
best fixed-arm	12.50
optimal stochastic	14.21

Table 1: Offline Benchmarks

By solving UCB algorithm for BwK, the optimal stochastic policy were obtained, shown as the probabilities to pick each arm, as follows:

$$[0.0, \quad 4.215608237433114 \times 10^{-23}, \quad 4.21560823747404 \times 10^{-23}, \\ 0.45402914841131264, \quad 0.5459708515886874, \quad 4.215608237433114 \times 10^{-23}]$$

which can be approximated as a stochastic policy between 2 of the 6 arms, arm 3 and arm 4 (0-based), with chosen probabilities = [0.4540, 0.5460] respectively. This policy performs better than random guessing and the best fixed-arm policy (arm 3, actually).

5.2 Online learning

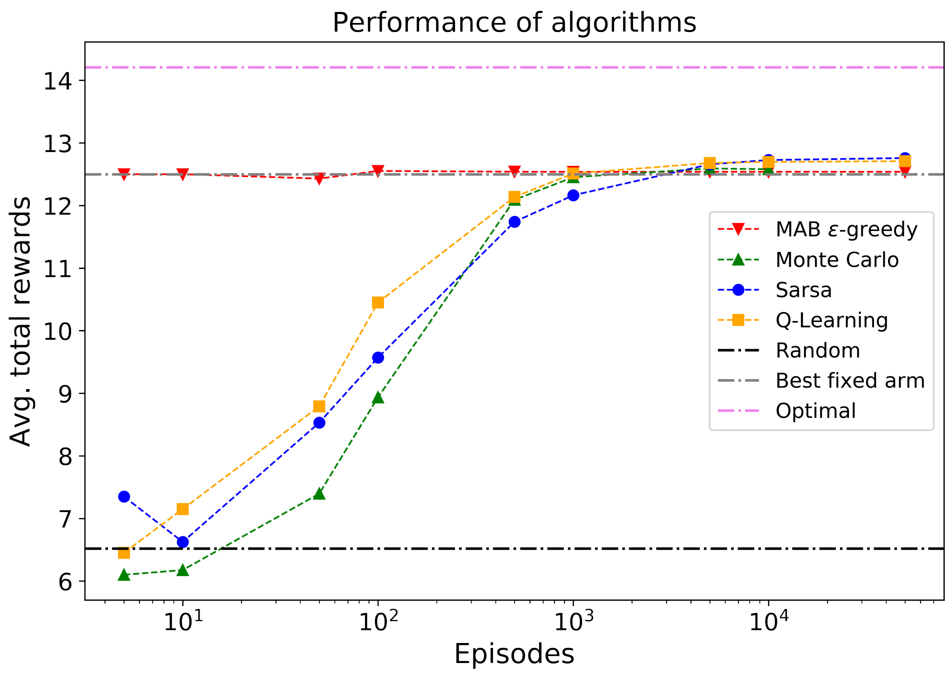


Figure 1: Comparison of online learning performance

The performance of multiple methods: epsilon greedy, Monte Carlo, Sarsa, Q-learning, along with the benchmarks are presented in Figure 1 above, measured by average total rewards over episodes. All algorithms were implemented and tested in an online setting: each algorithm acquired one single data sample at each time step and used the new information to update its policy. Monte Carlo method was an exception, which did not update its policy per step but after the completion of one episode. But Monte Carlo method is still considered online in this project since it processed the information from a data stream step by step although did not utilize it immediately.

As is shown in Figure 1, the epsilon greedy method implemented with MAB settings (red line) gave almost identical rewards as the best fixed arm policy (gray line) right after the first few episodes, and kept such performance later on. Its rewards value was significantly lower than the optimal performance (violet line), but was still the best performance as treating the BwK problem as a traditional MAB problem. The optimal policy was stochastic, but the epsilon greedy method converged to a deterministic (single arm dominating) policy.

For the three RL methods: Monte Carlo (green line), Sarsa (blue line) and Q-learning (orange line), the performance started near random guessing (black line), and then increased almost linearly with respect to the log-scale episode number. After about 1000 episodes, all three methods started to converge. Monte Carlo converged to almost the same place as epsilon greedy and best fixed arm policy, while Sarsa and Q-learning converged to somewhere slightly higher than that. But the difference was not very significant and the performance was still far from the optimal stochastic policy obtained from UCB for BwK algorithm.

The reason why RL methods did not converge to optimal policy may be due to the limitation of epsilon greedy policy itself. To balance the tradeoff between exploration and exploitation, all three RL methods were implemented using epsilon greedy approaches over Q values to update their state-based policies, which is guaranteed to converge to the best epsilon-soft policy [5], but not necessarily the global optimal.

From the perspective of online learning, all RL methods demonstrated the ability to deal with real-time data stream, though optimal performance was not achieved.

6 Conclusion and future work

In this project, a typical bandits with knapsacks problem, dynamic pricing with limit supply was explored. Multiple algorithms were tested under the same problem configuration. The experiment results showed that the policy given by UCB for BwK algorithm gave the highest performance, while traditional multi-armed bandit algorithms and reinforcement learning methods based on epsilon greedy could only converge to a sub-optimal epsilon-soft policy.

Future works can be focused on using a shifting or time-dependent distribution of buyer's valuation instead of the stationary normal distribution in this project, which may be a good test of the adaptive learning ability of different algorithms.

Acknowledgements

I would like to thank my dear sweetheart and little princess Amber for her consistent support, advice and encouragement throughout the project. :-)

References

[1] Badanidiyuru, A., Kleinberg, R., & Slivkins, A. (2013) *Bandits with knapsacks.* In Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on (pp. 207-216). IEEE.

[2] Alex Slivkins. (2016) *Lecture 11: Bandits with Knapsacks.* University of Maryland. `http://www.cs.umd.edu/~slivkins/CMSC858G-fall16/lecture11.pdf`

[3] Sankararaman, K. A., & Slivkins, A. (2017). *Combinatorial Semi-Bandits with Knapsacks.* arXiv preprint arXiv:1705.08110.

[4] Agrawal, S., & Devanur, N. R. (2014, June). *Bandits with concave rewards and convex knapsacks.* In Proceedings of the fifteenth ACM conference on Economics and computation (pp. 989-1006). ACM.

[5] Archis Ghate. (2018) *Slides 9. Reinforcement learning.* University of Washington. `https://faculty.washington.edu/archis/ddo/slides9.pdf`