July 2024

Price Optimization with Bandits

Pranjal Rawat, DSA Intern

ROBLOX

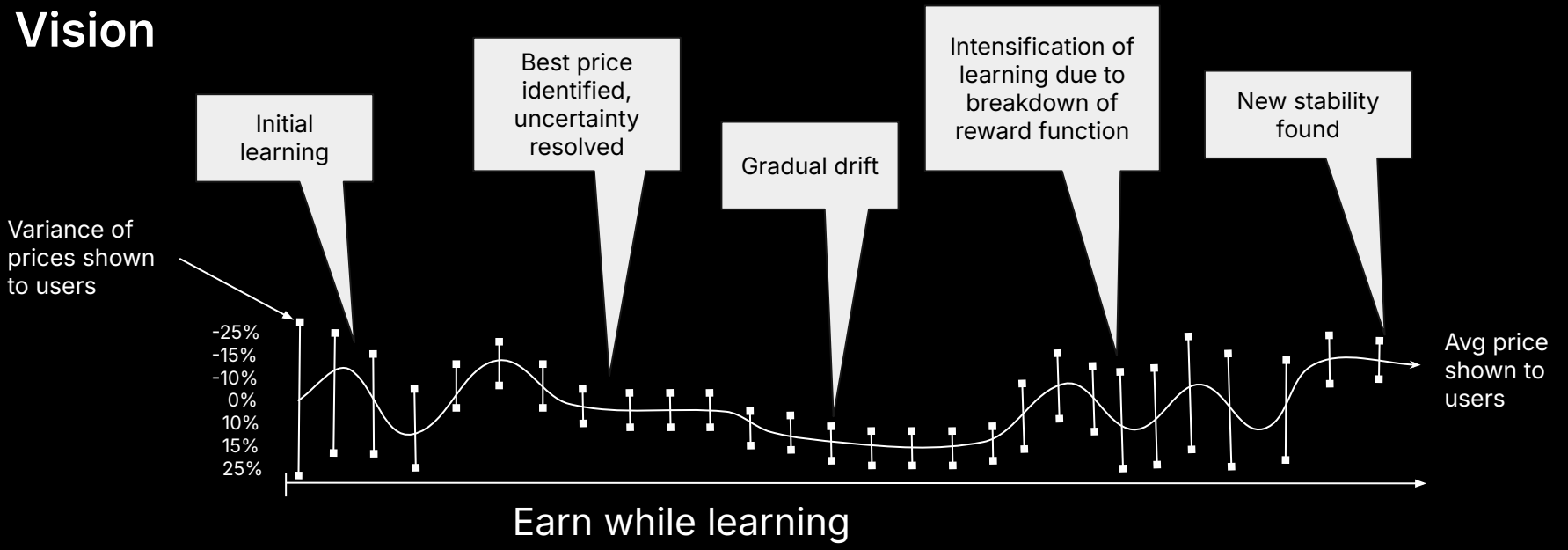
Source doc: here.

Current System

Issues

Issue	Evidence	Solution
Optimal price changes by country	Localized PO For Blox Fruits and Adopt Me	Pricing based on context
Optimal prices are different across item clusters	Pet Sim X Price Optimization Test Results and Recommendations	Cluster level pricing
Optimal price may simply.. “drift”	Expectation vs. Reality of In-Experience Price Optimization Holdouts	Decaying weights / sliding windows to focus on recent experiences

Vision



Offline evaluation tests on **Slap Battles**👏 game makes a case for 7-16% increase in median revenue.

What's next? An A/B Test!

Treatment Arm 1

Aggregate Price Bandit that

- Models revenue as function of price and context.
- Context
  - User Composition
  - Seasonality
- Continuously Explores via Upper Confidence Bounds on Revenue

Treatment Arm 2

Item Cluster level Bandit that

- Factors in substitution patterns between items
- Models the reward *surface* across groups of items.
- Joint optimization all item prices over revenue surface

Hyperparameter tuning and "training" will be done via offline evaluation on past price experiments.

Slap Battles👏: Direct Evaluation

- Had price experimentation run on 5/31 to 6/13.

Periodize	Oracle	Bandits	Monte Carlo
Aggregate raw transaction data in price buckets into 5-minute intervals. User composition in interval becomes “context”.	Train a machine learning model to predict rewards from context for each price point - “Oracle”.	Baseline is to uniformly randomize prices. What’s new? • Efficient Exploration • Online Reward Modelling	Each round, 7 contexts are generated and bandit must price all 7 and receive rewards. Compare performance using Monte Carlos.

Data

Aggregating transaction data to 5 minute intervals:

	round	price	avg_reward	d_gender_male	d_gender_female	d_US_users	d_age_in_days	d_days_since_join	d_days_since_join_game	s_hour_of_day	...
0	0	75.0	20.000000	1.000000	0.000000	1.000000	5168.800000	1408.200000	494.400000	0	...
1	0	80.0	699.000000	1.000000	0.000000	1.000000	6057.000000	520.000000	511.000000	0	...
2	0	85.0	163.857143	0.857143	0.000000	0.714286	5400.142857	1176.857143	634.714286	0	...
3	0	100.0	116.022727	0.795455	0.090909	0.897727	5441.022727	1135.988636	583.443182	0	...
4	0	115.0	186.500000	0.250000	0.250000	0.750000	5811.000000	766.000000	477.250000	0	...
5	0	120.0	799.000000	1.000000	0.000000	1.000000	5847.000000	730.000000	80.000000	0	...
6	0	125.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	...
7	1	75.0	79.181818	1.000000	0.000000	0.636364	5109.454545	1467.545455	321.090909	0	...
8	1	80.0	46.500000	1.000000	0.000000	0.750000	4820.000000	1757.000000	333.500000	0	...
9	1	85.0	799.000000	1.000000	0.000000	1.000000	6494.000000	83.000000	81.000000	0	...

Rows: 28,224,
Columns: 26

Algorithms

Bandit	Logic	Context (x)	Model	Equation
Uniform (Baseline)	Different prices to all 7 for a half the rounds, then best price shown to all.	None	None	$r(p) \leftarrow \frac{1}{T_{1/2}} \sum_{t=1}^{T_{1/2}} r(t) \text{ for } t \leq T_{1/2}$
UCB	Predicts mean rewards for each price, plus an exploration bonus.	None	None	$r(p) \leftarrow \bar{r}(p) + \alpha\sqrt{\frac{2 \log t}{n_p}}$
Linear UCB (Disjoint)	Predicts rewards for each price, given context. Adds an exploration bonus using stderr.	User compositions	OLS	$r(p) \leftarrow \theta_p^T \cdot x + \alpha\sqrt{x^T \cdot A_p^{-1} \cdot x}$
Kalman UCB	Models reward as fn of price, price^sq and price * user composition. Adds small exploration bonus using varcov.	Price, price^sq, p*user compositions	Kalman Filter	$r(p) \leftarrow \theta_p^T \cdot x + \alpha \cdot \sigma_p$
Greedy LGBM	Learns a separate LGBM reward model for each price arm and chooses prices greedily.	User compositions	LGBM	$r(p) \leftarrow \hat{r}(p \mid x)$

All contextual models offer different prices to all 7 users each round.

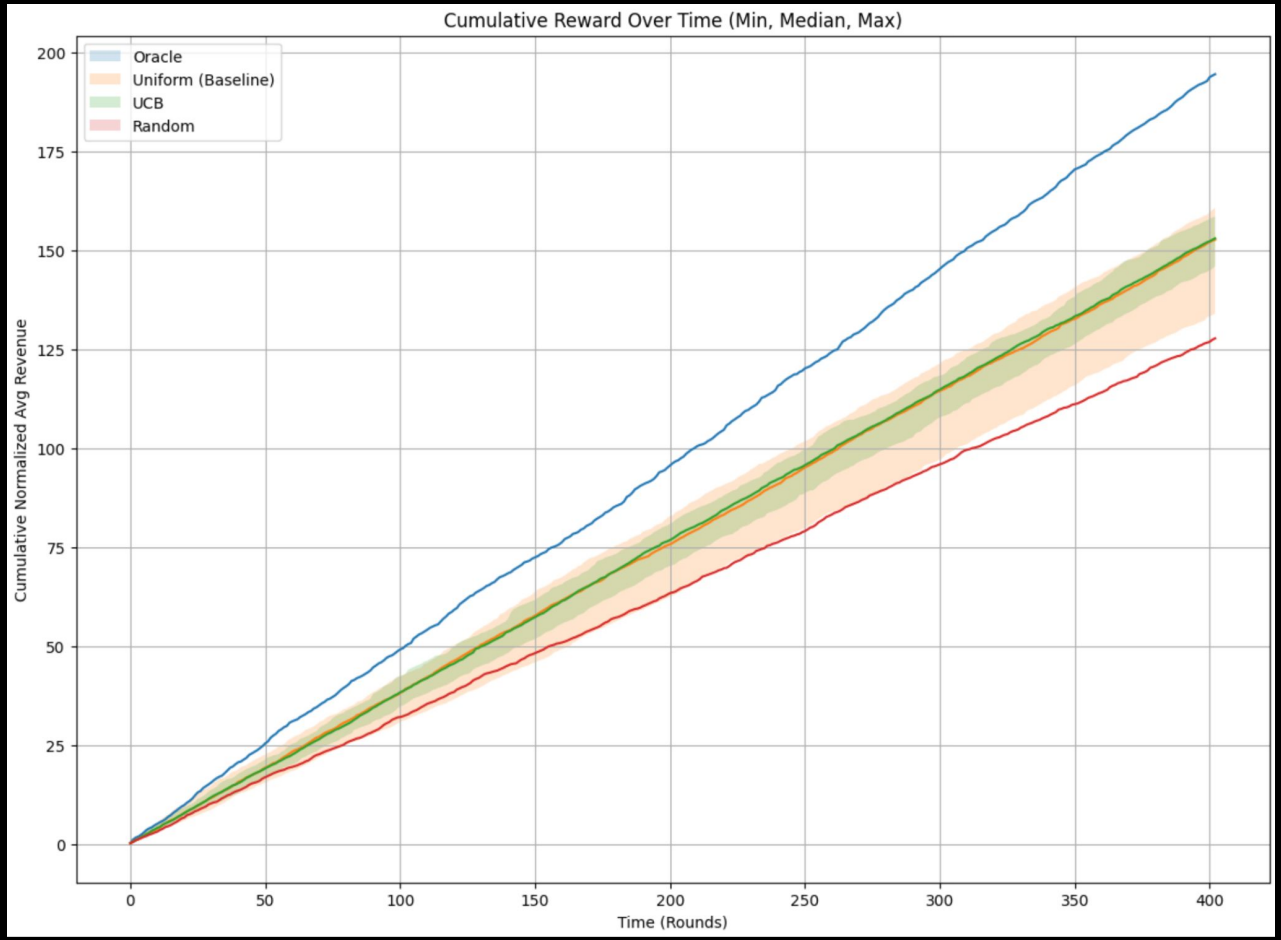
Simulating 1 Day

The UCB (Green) method helps reduce variance of estimates and raise the lower bound.

The Baseline/Uniform (Orange) carries a risk of doing poorly.

Oracle (Blue) is best possible.

Random (Red) is worst possible.

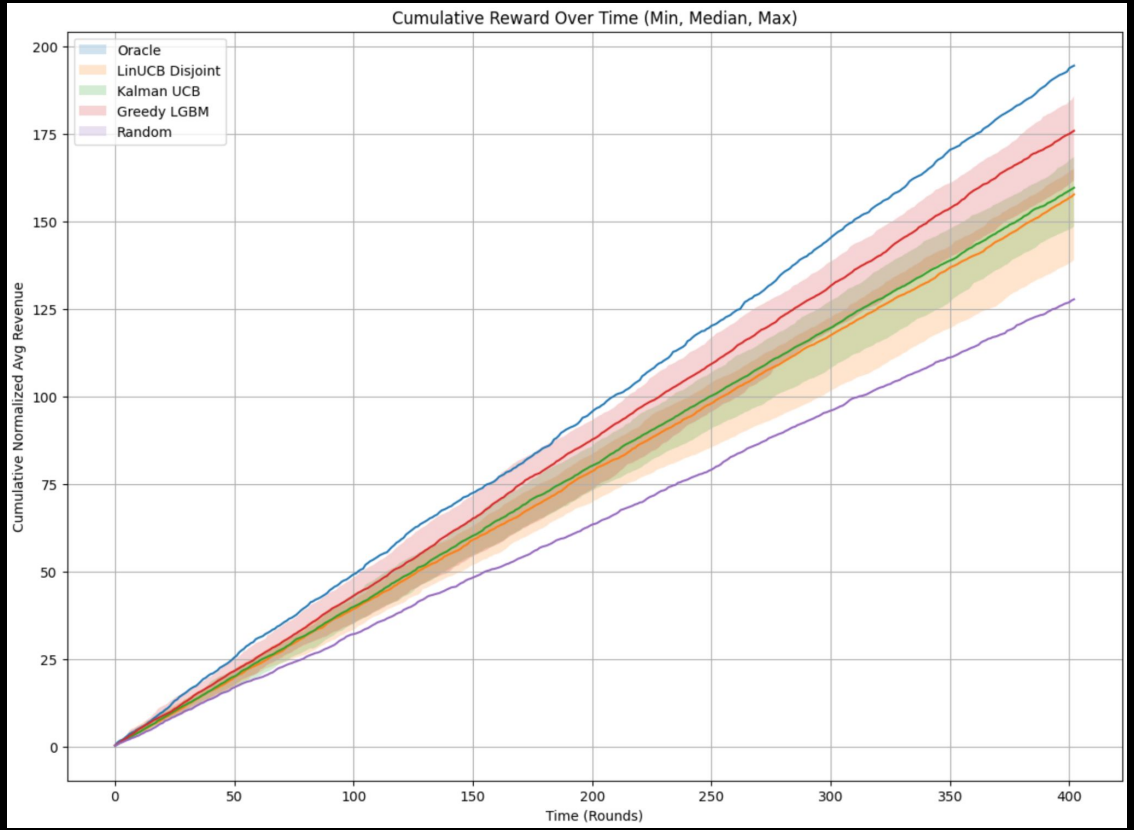


30 simulations per Monte Carlo experiment. This graph compares the UCB to the Uniform (Baseline).

Simulating 1 Day

Greedy LGBM is slow but outperforms all others - indicating that a flexible reward model is important.

However, the Kalman UCB is fast, efficient and second best.



30 simulations per Monte Carlo experiment.

Lift over Baseline

Two reasons to use online learning:

- More Robux: Increase in revenue.
- More Robust: Decrease in risk.

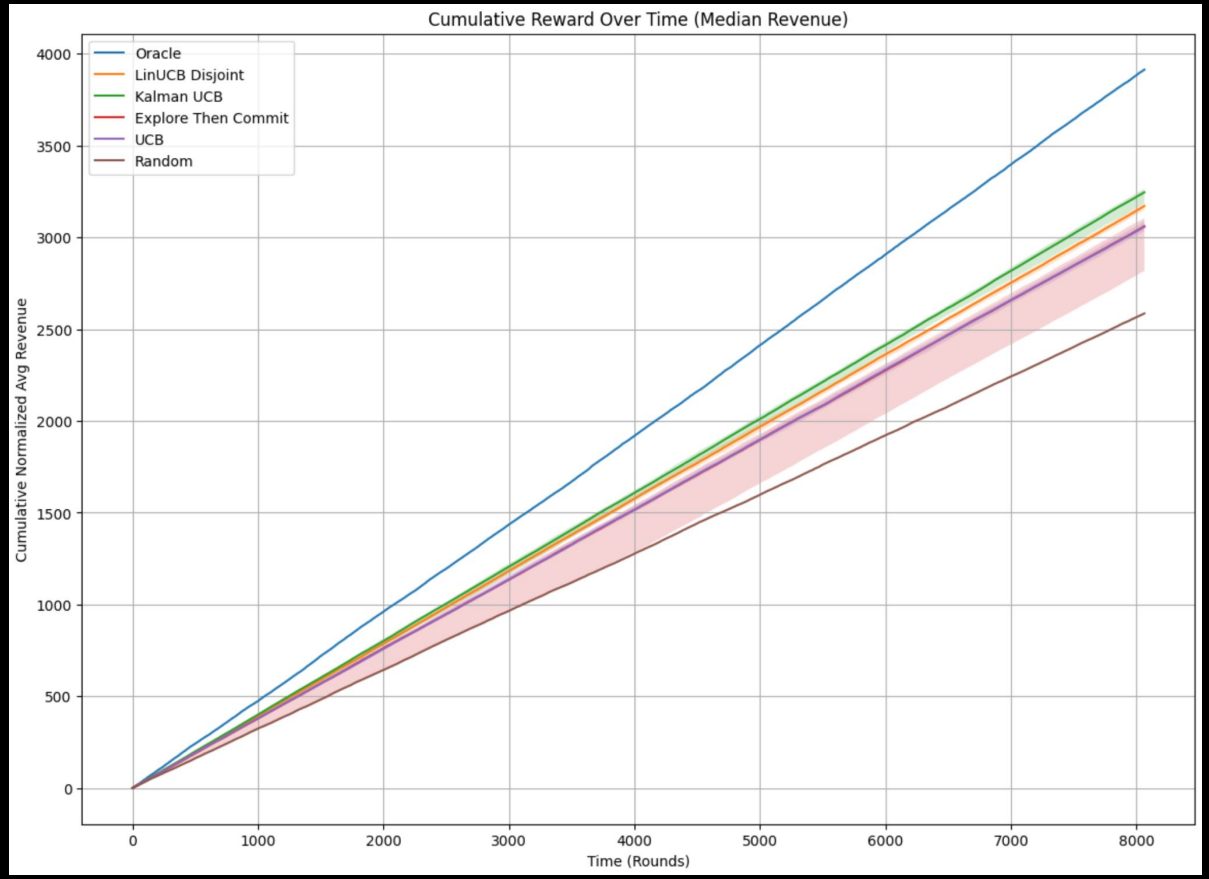
Bandit	Increase in Median Revenue	Decrease in Range of Revenues
UCB	+0.2%	-53%
Linear UCB (Disjoint)	+2.7%	-3.0%
Kalman UCB	+4.6%	-25%
Greedy LGBM	+16%	-10%

Simulating 4 Weeks

UCB and Baseline median revenue is nearly identical, but UCB has much lower variance.

Greedy LGBM was too slow to complete in time.

Kalman UCB was fast, efficient and best.



30 simulations per Monte Carlo experiment. The UCB and ETC median overap, but ETC variance is wide.

Lift over Baseline

The superiority of the Kalman UCB in median revenue might be attributed to it modeling the entire revenue function (i.e. demand learning).

Bandit	Increase in Median Revenue	Decrease in Range of Revenues
UCB	+0.54%	-54.0%
Linear UCB (Disjoint)	+4.18%	-86.2%
Kalman UCB	+6.57%	-72.9%

Flexible models are needed to model the revenue well and yet adding some a-priori information about demand (e.g. including price^2) does help.

Kalman UCB is similar to Linear UCB (Hybrid) which also models arm-specific features.
In this iteration, we were unable to test Greedy LGBM as it gets very slow as its internal database grows.

Thanks

- Steve - for motivation
- Shahzoor - data pipelines, code and context.
- Jose - suggesting the offline evaluation framework, the notion of “user compositions” and Kalman filter based on his experience at Lyft.
- For feedback - Ujwal (all the literature review on topic), Phil, Amar (suggestion of contextual bandit).

Appendix

Prepare Data

Aggregating transaction data to 5 minute intervals:

	round	price	avg_reward	d_gender_male	d_gender_female	d_US_users	d_age_in_days	d_days_since_join	d_days_since_join_game	s_hour_of_day	...
0	0	75.0	20.000000	1.000000	0.000000	1.000000	5168.800000	1408.200000	494.400000	0	...
1	0	80.0	699.000000	1.000000	0.000000	1.000000	6057.000000	520.000000	511.000000	0	...
2	0	85.0	163.857143	0.857143	0.000000	0.714286	5400.142857	1176.857143	634.714286	0	...
3	0	100.0	116.022727	0.795455	0.090909	0.897727	5441.022727	1135.988636	583.443182	0	...
4	0	115.0	186.500000	0.250000	0.250000	0.750000	5811.000000	766.000000	477.250000	0	...
5	0	120.0	799.000000	1.000000	0.000000	1.000000	5847.000000	730.000000	80.000000	0	...
6	0	125.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0	...
7	1	75.0	79.181818	1.000000	0.000000	0.636364	5109.454545	1467.545455	321.090909	0	...
8	1	80.0	46.500000	1.000000	0.000000	0.750000	4820.000000	1757.000000	333.500000	0	...
9	1	85.0	799.000000	1.000000	0.000000	1.000000	6494.000000	83.000000	81.000000	0	...

Rows: 28,224,
Columns: 26

ROBLOX

Oracle Reward Model

For each price, estimate a LGBM model to predict avg. reward from context:

Price	Out-sample R2
75	0.42
80	0.38
85	0.42
100	0.17
115	0.34
120	0.48
125	0.45

N = 4032, K = 26

Bandit Simulation Loop

- In each round t,
  - 7 user contexts are created by Oracle (representing 7 different users)
  - The bandit must price all 7 contexts
  - The oracle will score all 7 contexts according to price played
  - The bandit will collect rewards obtained
  - The bandit will update internal states

Bandit Notation

$r(p)$: Reward function as a function of price p.

x: Context vector (demographic features).

α: Exploration parameter.

$\hat{r}(p \mid x)$: Predicted reward given price p and context x.

$\bar{r}(p)$: Empirical mean reward for price p.

n_p: Number of times price p has been chosen.

σ_p: Standard error for price p.

A_p: Covariance matrix for price p.

T: Number of rounds or time steps.