Introduction to Machine Learning for Economists: Lecture 2

Sonan Memon
Institute of Business Administration,
Karachi

8th December 2020

Outline of Lecture 2

i. Active and Reinforcement Learning
ii. Bandit Problems: Exploration Exploitation Trade off and Adaptive Treatment Assignment
iii. Greedy Algorithms
iv. Thompson Sampling Algorithm
v. Upper confidence bound (UCB) Algorithm
vi. Contextual bandits
vii. Applications
viii. R Pseudo Code

Readings for Lecture 2

i. Bubeck, S. and Cesa-Bianchi, N. (2012). Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. Foundations and Trends in Machine Learning, 5(1):1–122.

ii. Russo, D. J., Roy, B. V., Kazerouni, A., Osband, I., and Wen, Z. (2018). A Tutorial on Thompson Sampling. Foundations and Trends in Machine Learning, 11(1):1–96.

Motivation

i. Traditionally much experimentation is done by assigning a predetermined number of units to each of a number of treatment arms. With 2 arms, this is called A/B testing.

ii. After outcomes are measured the average effect of the treatment would be estimated using the difference in average outcomes by treatment arm.

iii. Inefficient experimentation where we waste units by assigning them to treatment arms that we already know with a high degree of uncertainty to be inferior to some of the other arms.

iv. Modern methods for experimentation focus on balancing exploration of new treatments with exploitation of treatments currently assessed to be of high quality.

Motivation

i. In multi-armed bandits, the assignment for each unit can depend on all the information learned up to that point. Given this information, and given a parametric model for the outcomes for each treatment, and a prior for the parameters of these models, we can estimate the probability of each treatment being the optimal one.

ii. We re-evaluate the assignment probabilities after a batch of new observations has come in, all based on the same assignment probabilities. From this perspective we can view a standard A/B experiment as one where the batch is the full set of observations.

Multi Armed Bandits

Figure: Multi Armed Bandit Problem

Multi Armed Bandits

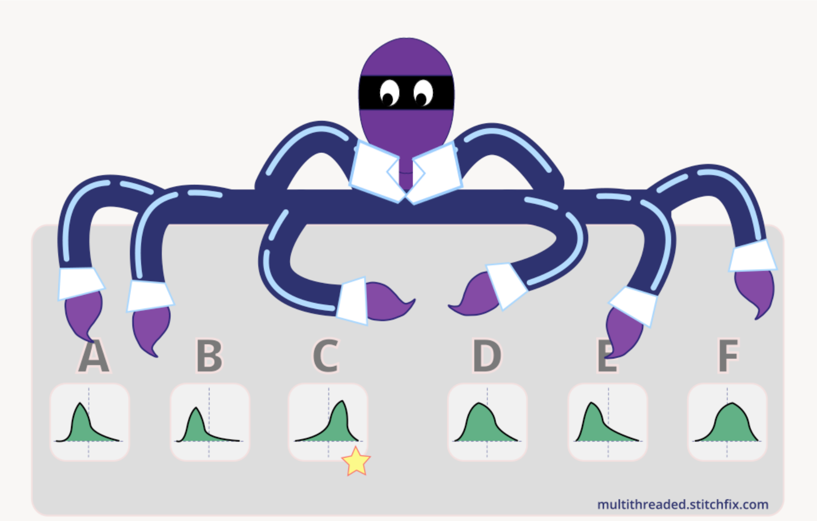


Figure: Multi Armed Bandit Problem

Applications

i. Design of online experiments and online advertising.

ii. Dynamic pricing as in Misra et al (2017), stock investment,

iii. Medical treatment assignment in clinical trials and medication dosing problems.

iv. Economic policy design application from Cario et al (2020).

Defining a Multi Armed Bandit Problem

i. Multi Armed Bandit Problem is a tuple of form $\langle \mathcal{A}, \mathcal{R} \rangle$ where $\mathcal{A}$ is the set of actions.

ii. Each action $a \in \mathcal{A}$ refers to interaction with one arm and $Q(a) = \mathbb{E}[r|a] = \theta$, where $\theta \in \{\theta_1, \theta_2, ..., \theta_K\}$ are reward probabilities for all arms.

iii. $\mathcal{R}$ is reward function so that for action a_t taken at time t $\mathcal{R}^{a_t}(r) = \mathbb{P}\{R = r|A = a_t\}$, i.e $\mathcal{R}_t \sim \mathcal{R}^{a_t}$ is the reward distribution, conditional on action a_t.

iv. Goal in finite horizon case is to maximize $\sum_{t=0}^{T} R_t$, where T is horizon.

Minimizing Total Regret

i. Goal in finite horizon case is to maximize $\sum_{t=0}^{T} R_t$, where T is horizon.

ii. If there is some arm with highest expected value of reward, optimal action should choose this arm all the time and hence maximizing sum of rewards is equivalent to minimizing the regret from choosing sub-optimal action.

iii. Define $v^* = Q(a^*) = \underset{a\in\mathcal{A}}{max}\, Q(a)$

iv. $L_t = \mathbb{E}\left[\sum_{t=1}^{T} v^* - Q(a_t)\right]$ is total regret. We want to find algorithms which minimize total regret or lead to sublinear rate of increase in total regret over time.

Total Regret as Function of Gaps and Counts

i. $L_t = \mathbb{E}\left[\sum_{t=1}^{T} v^* - Q(a_t)\right]$ is total regret.

ii. Define $N_t(a)$ as number of times we have chosen action a up till time t and Δ_a as the gap between expected value of reward under optimal action and under action $a \in \mathcal{A}$.

iii. Then, $L_t = \mathbb{E}\left[\sum_{t=1}^{T} v^* - Q(a_t)\right] = \sum_{a \in \mathcal{A}} \mathbb{E}[N_t(a)].\Delta_a$ can be expressed as function of gaps and counts. Good algorithm ensures small counts for big gaps.

iv. Since $Q(a)$ is not known we estimate it by Monte Carlo evaluation i.e average reward from action a up till time t: $\overline{Q_t(a)} = \frac{1}{N_t(a)} \sum_{t=1}^{T} 1\{a_t = a\}.R_t$.

Bounding Regret

i. If the gaps in rewards across arms is larger, then the regret will be larger as well.

ii. Similarly, conditional on gaps in rewards, if the reward distributions are hard to distinguish, then it will take more experimentation to pin down optimal arm and hence regret will be larger.

iii. We can measure the distance between two distributions by the Kulback-Leibler divergence or relative entropy between the two distributions:
$D_{KL}(f||g) = \mathbb{E}_f \, log\left(\frac{f(w)}{g(w)}\right) = \int_{w\in\mathcal{W}} log\left(\frac{f(w)}{g(w)}\right) f(w)\, dw$

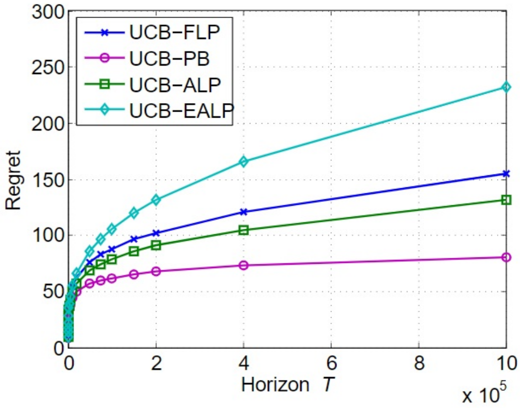
Asymptotic Bound on Regret

Theorem (Lai and Robbins)

The total regret is at least logarithmic in the number of time steps

$$\lim_{t\to\infty} L_t \geq log(t) \sum_{a|\Delta_a>0} \frac{\Delta_a}{D_{KL}(\mathcal{R}^a||\mathcal{R}^{a^*})}$$

Sub-Linear Regret



Greedy Algorithm

i. Chooses a_t so that $a_t = \underset{a \in \mathcal{A}}{argmax}\, \overline{Q_t(a)}$, i.e chooses the arm which has given highest average reward up till time t.

ii. The problem is that the greedy algorithm can lock into a sub-optimal action forever and it has linear total regret.

Optimistic Greedy Algorithm

i. Initializes all arms so that $\overline{Q_1(a)} = R^{max}$, $\forall a \in \mathcal{A}$ i.e assume that all arms give the highest possible average reward to begin with.

ii. Optimism under the face of uncertainty: encourages exploration of unknown values but a few unlucky samples can lock out optimal action forever.

iii. Optimistic greedy also has linear total regret but it works pretty well in many applications and is easy to use.

ϵ Greedy Algorithm

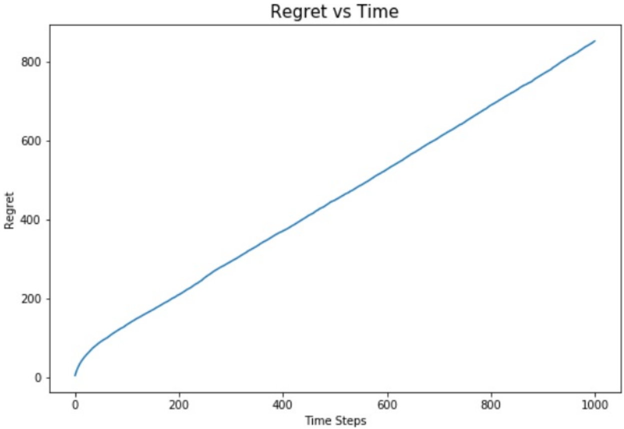
i. With prob $1 - \epsilon$, selects $a_t = \underset{a \in \mathcal{A}}{argmax}\ \overline{Q_t(a)}$ and with probability ϵ, selects a random action $a_t \in \mathcal{A}$.

ii. Also has linear total regret.

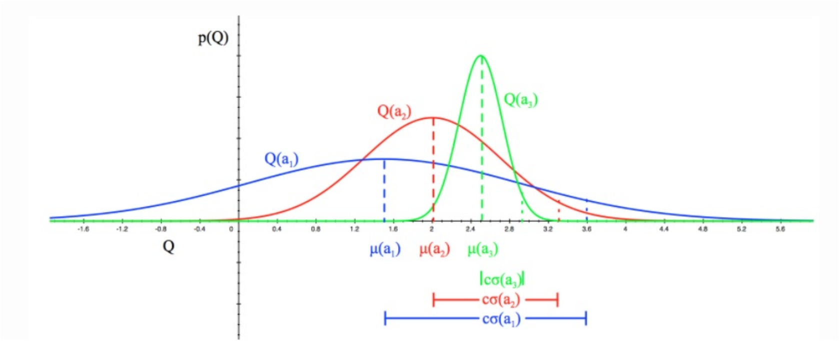
iii. Picking a decay schedule for $\{\epsilon_t\}$ can lead to sub-linear, logarithmic, asymptotic total regret but constructing optimal decay schedule requires knowledge of gaps, which does not exist.

iv. Goal is to find an algorithm which produces sub-linear total regret *without* knowledge of gaps

Regret for Epsilon Greedy Algorithm



Optimism in the Face of Uncertainty



Upper Confidence Bound Algorithm

i. Estimate $U_t(a)$ such that $Q_t(a) \leq \overline{Q_t(a)} + U_t(a)$ with high probability such as 95%.

ii. Lower $N_t(a)$ means higher uncertainty about a so that $U_t(a)$ will be quite large and if $N_t(a)$ becomes large, $U_t(a)$ will become lower since the true distribution of rewards will be learned more precisely.

iii. $a_t = \underset{a \in \mathcal{A}}{argmax}\, \overline{Q_t(a)} + U_t(a)$, i.e pick the action with the highest upper confidence bound i.e optimism in the face of uncertainty.

iv. Larger $U_t(a)$ si good for exploration and smaller $U_t(a)$ is good for exploitation.

Hoeffding's Inequality

Theorem

Let $X_1, ...X_T$ be iid variables in $[0,1]$ and let $\overline{X}_T = \frac{1}{T}\sum_{t=1}^{T} X_t$ be the sample mean. Then

$$\mathbb{P}\{\mathbb{E}[X] > \overline{X}_T + u\} \leq e^{-2Tu^2}$$

Constructing UCB Using Hoeffding's Inequality

i. Estimate $U_t(a)$ using Hoeffding's Inequality.

ii. If we want 95% confidence interval, $p = 0.05$. We use condition $e^{-2N_t(a)U_t(a)^2} = p$ to derive that $U_t(a) = \sqrt{\frac{-log\ p}{2N_t(a)}}$, where $p \in (0, 1)$.

iii. We want p to decay with T so we set $p = T^{-\alpha}$, where α measures degree of decay of p.
$\implies -log(p) = -log(T^{-\alpha}) = \alpha log(T)$

UCB Algorithm

i. Hence $U_t(a) = \sqrt{\frac{\alpha log(T)}{2N_t(a)}}$ and $a_t = \underset{a \in \mathcal{A}}{argmax}\, \overline{Q_t(a)} + \sqrt{\frac{\alpha log(T)}{2N_t(a)}}$

ii. UCB has logarithmic asymptotic total regret.

Probability Matching

i. $\pi(a|\mathcal{H}^T) = \mathbb{P}\{Q(a) > Q(a^{'}), \forall a^{'} \neq a|\mathcal{H}^T\}$

ii. $\pi(a|\mathcal{H}^T) = \mathbb{E}_{\mathcal{R}|h_t}[\mathbb{1}\{a = \underset{a\in\mathcal{A}}{argmax}\, Q(a)\}]$.

iii. $\mathcal{H}^T$ could include history of rewards $R^T = \{R_1, R_2, ...R_T\}$ as well as history of actions $a^T = \{a_1, a_2, ...a_T\}$.

iv. The posterior is updated in Bayesian manner, given the prior distribution.

v. Probability matching reflects optimism in the face of uncertainty and encourages exploration since more uncertain actions or arms have higher probability of being optimal action.

vi. However, analytical computation of posterior may not be feasible so we use Thompson sampling which is sample based.

Thompson Sampling

i. Takes a Bayesian approach by choosing prior distribution over θ, the parameters of interest regarding distribution of outcomes across arms.

ii. Using Bayes' rule, the hyper parameters are updated over time and assignment to arms is done in proportion to the posterior probabilities of rewards for each arm.

Thompson Sampling

i. Choose a prior distribution for $Q(a)$. In the bernoulli context, a natural choice is the beta distribution $\mathcal{B}(\alpha^i_T, \beta^i_T)$ since $Q(a)$ is probability of success in that case and beta distribution has probability mass over $[0, 1]$.

ii. α^i_T corresponds to number of successes in the past and β^i_T measures number of failures for action i up till time T.

iii. The Thompson sampling algorithm samples expected reward $\widehat{Q(a)}$ from the prior distribution $\mathcal{B}(\alpha^i_T, \beta^i_T)$ for each action.

iv. The best action i.e $a_{T+1} = \underset{a \in \mathcal{A}}{argmax}\, \widehat{Q(a)}$.

Thompson Sampling

i. $r_t = 1$ or 0 in bernoulli case.

ii. When true reward is observed, we update hyperparameters in the following manner:
$\alpha^i_{T+1} = \alpha^i_T + r_t \mathbb{1}\{a_{T+1} = a^i\}$ and
$\beta^i_{T+1} = \beta^i_T + (1 - r_t)\mathbb{1}\{a_{T+1} = a^i\}$.

iii. In next period, we sample our $\widehat{Q(a)}$ from the prior distribution $\mathcal{B}(\alpha^i_{T+1}, \beta^i_{T+1})$ for every action.

An array of Beta Distributions

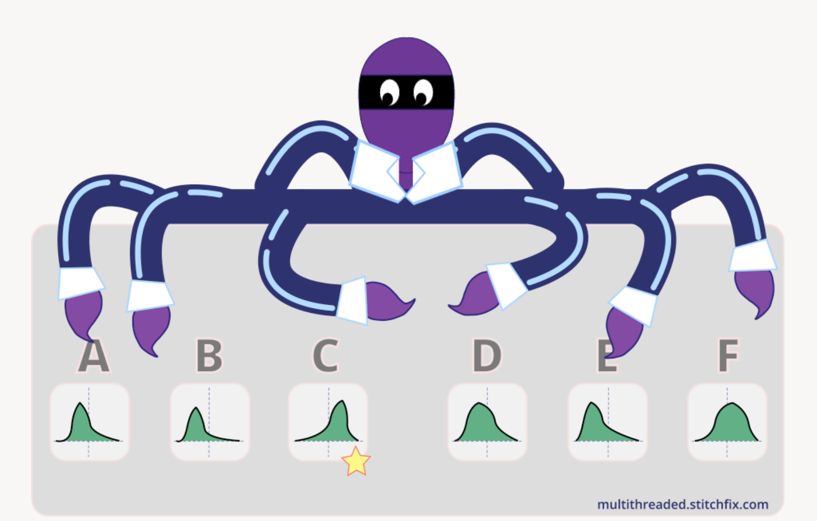
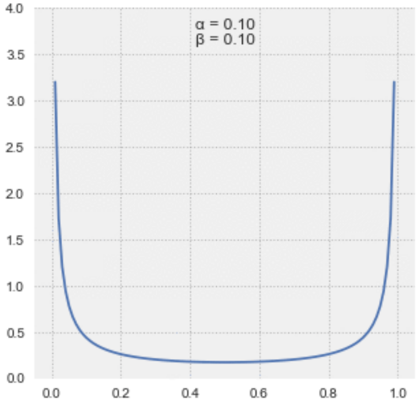


Figure: Multi Armed Bandit Problem

Beta Distribution



Thompson Sampling

i. TS allows for exploration since when uncertainty is high, the Beta distribution will have high variance $\implies$ posterior mean in sample is high. However, when enough learning has occurred, the variance will shrink and if this arm were not optimal, we choose it with lower probability (exploitation).

ii. TS algorithm achieves the theoretical lower bound on regret for Bernoulli Bandits and works really well in practice.

Contextual Bandits

i. Setting where individuals have observed attributes X_i has been termed the contextual bandit problem, since treatment assignments can condition on X_i.

ii. $Y^{dx} \sim F^{d,x}$ and θ^{dx} is estimated as a function of both arms d and covariates x and then exploration exploitation trade off is solved.

iii. In contextual bandits, the choice of the model that maps user characteristics to expected outcomes has to be made.

iv. Dimakopoulou et al. (2017) highlights that unlike non-contextual bandits, contextual bandits have successive units which are unique. The assignment of a particular individual to a treatment thus contributes to learning for the future only indirectly, since the future individuals will have different X_i.

Economic Application Based on Cario et al (2020)

i. Cario et al (2020) use a version of thompson sampling for adaptive, targeted treatment assignment in a field experiment, focusing on treatments for improving job finding rate for Syrian refugees in Jordan.

ii. Since the start of the Syrian conflict, Jordan received close to 700,000 Syrian refugees — one tenth of its original population (UNHCR).

iii. In 2016, Jordan launched the Jordan Compact: in exchange for trade concessions and access to conditional financing, the Government agreed to provide 200,000 work permits for refugees, lifting legal barriers that prevented them from obtaining work.

Economic Application Based on Cario et al (2020)

i. The algorithm balances the goal of maximizing participant welfare and precision of treatment effect estimates.

ii. The immediate employment impacts of cash grant, behavioral nudge and information provision are close to zero relative to control group but targeting through algorithm raise employment by 20%.

iii. After four months, cash has a sizable effect on employment and earnings of Syrians but other variables have only short term effects. This shows the importance of liquidity constraints in hindering employment opportunities for developing countries.

Economic Application Based on Cario et al (2020)

i. The designer starts with a prior over the effectiveness of k different treatments; typically a diffuse and symmetric default prior.

ii. Every period, the designer observes the outcomes of participants and estimates the posterior probability $\hat{p}_t^{dx}$ that treatment/arm d is optimal conditional on strata x and at time t.

iii. Their Tempered Thompson Algorithm assigns treatments in the following way, for individuals from stratum x:
With probability γ: assign treatment d to individual i with probability $\frac{1}{K}$ and with probability $1 - \gamma$ with probability $\hat{p}_t^{dx}$.

iv. $\gamma = 1$ refers to conventional RCT, $\gamma = 0$ is standard thompson sampling.

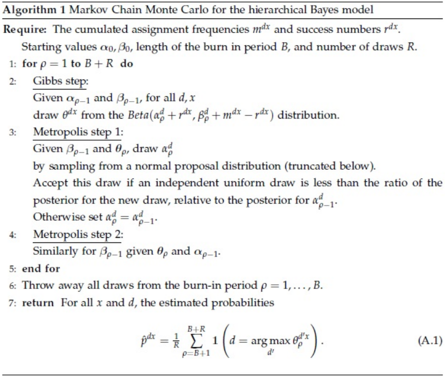
Economic Application Based on Cario et al (2020)

i. They use the following Hiearchical Bayes Model:
$Y_{it}^{d} | \left(X_{it} = x, \theta^{dx}, \alpha^{d}, \beta^{d} \right) \sim Ber(\theta^{dx})$
$\theta^{dx} | \left(\alpha^{d}, \beta^{d} \right) \sim Beta(\alpha^{d}, \beta^{d})$
$(\alpha^{d}, \beta^{d}) \sim \pi.$

ii. θ^{dx} is average potential outcome for treatment d in strata x, (α^{d}, β^{d}) are hyperparameters and π is the hyperprior distribution, preferably uninformative.

iii. Use MCMC algorithm to estimate $\hat{p}_t^{dx}$ and assign treatment of d to agent in strata x with probability $(1 - \gamma)\hat{p}_t^{dx} + \frac{\gamma}{K}$.

MCMC Algorithm for Hierarchical Bayes Model

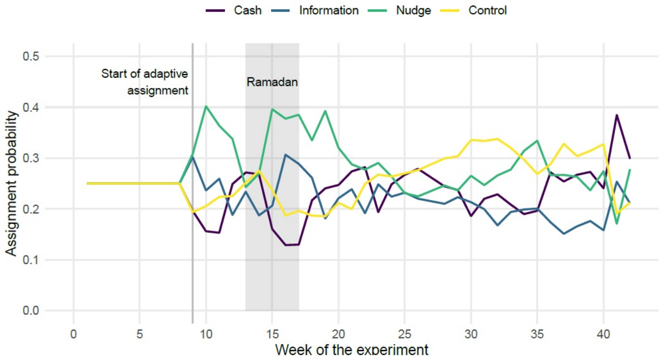


Algorithm 1 Markov Chain Monte Carlo for the hierarchical Bayes model

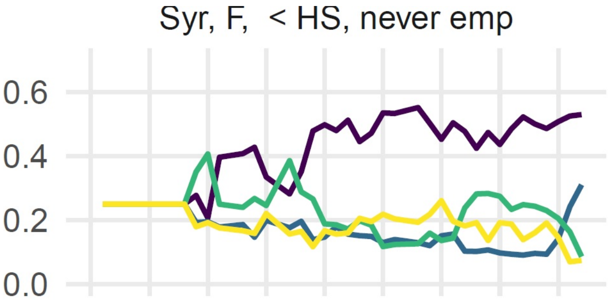
```
Require: The cumulated assignment frequencies $m^{dx}$ and success numbers $r^{dx}$.
    Starting values $\alpha_0, \beta_0$, length of the burn in period $B$, and number of draws $R$.
for $\rho = 1$ to $B + R$ do
    Gibbs step:
    Given $\alpha_{\rho-1}$ and $\beta_{\rho-1}$, for all $d, x$
    draw $\theta^{dx}$ from the $Beta(\alpha^d_\rho + r^{dx}, \beta^d_\rho + m^{dx} - r^{dx})$ distribution.
    Metropolis step 1:
    Given $\beta_{\rho-1}$ and $\theta_\rho$, draw $\alpha^d_\rho$
    by sampling from a normal proposal distribution (truncated below).
    Accept this draw if an independent uniform draw is less than the ratio of the
    posterior for the new draw, relative to the posterior for $\alpha^d_{\rho-1}$.
    Otherwise set $\alpha^d_\rho = \alpha^d_{\rho-1}$.
    Metropolis step 2:
    Similarly for $\beta_{\rho-1}$ given $\theta_\rho$ and $\alpha_{\rho-1}$.
end for
Throw away all draws from the burn-in period $\rho = 1, \ldots, B$.
return For all $x$ and $d$, the estimated probabilities
```

$$\hat{p}^{dx} = \frac{1}{R} \sum_{\rho=B+1}^{B+R} \mathbf{1}\left(d = \arg\max_{d'} \theta_\rho^{d'x}\right). \tag{A.1}$$

Adaptive Treatment Assignment



Adaptive Treatment Assignment



Economic Application 2: Misra et al (2017)

i. Online, real time pricing when there exist millions of transactions.

ii. It is infeasible to have complete knowledge of demand curve for each product. A manager can run price experiments to learn about demand: exploration versus exploitation trade off.

iii. Our proposed pricing algorithm sequentially sets prices to balance currently earning profits and learning about demand for future profits $p \in \{\$0.30, \$0.4, \$0.5, \$0.6, \$0.7\}$.

iv. Combination of UCB based multi armed bandit problem and economic theory to pin down consumer demand curve is used in this paper.

Economic Application 2: Misra et al (2017)

i. In typical UCB algorithm, when a particular price is charged (an arm is played), the firm's observations are limited to profits from that price (arm). They extend this to allow learning across prices based on economics.

ii. For example, if a consumer purchases a good at \$3, the manager can infer she would have purchased at any price below \$3 and if a consumer does not purchases a good at \$3, the manager can infer she would not have purchased at any price above \$3.

iii. Proposed method can calculate about 2 million prices per minute, and can be used for real-time online pricing.

Economic Application 2: Misra et al (2017)

i. Assume indirect utility function $u_i = v_i - p_i$ for consumer i and so consumer purchases product only if $v_i \geq p_i$.

ii. $v_i \in [v_i - \delta, v_i + \delta]$, $\forall i \in s$ (segments).

iii. $p \in \{p_1, p_2, p_3, ..., p_K\}$. $\pi(p) = pD(p)$

iv. Optimal pricing algorithm $p = \phi(\mathcal{H}^{\mathcal{T}})$.

v. Regret $L(\pi, t) = \mathbb{E}\left[\sum_{\tau=1}^{t} \pi^* - \pi_\tau\right]$. π^* is profit corresponding to profit maximizing price p^*.

vi. If a consumer is willing to purchase a product a price p_1, he must be willing to purchase for all prices such that $p_k < p_1$.

Implementation in R and Python

i. Bandits including contexual bandits can be estimated in R by using the `contexual` package: `https://cran.r-project.org/web/packages/contextual/readme/README.html`.

ii. Python application 1 (Basic): `https://github.com/lilianweng/multi-armed-bandit`

iii. Python application 2 (Also includes contexual bandits): `https://://github.com/gdmarmerola/interactive-intro-rl`

Pseudo Code in R

i. library(contextual)

ii. weights = matrix(c(0.7, 0.2, 0.2), 1, 3)

iii. bandit = ContextualBernoulliBandit$new(weights = weights)
agents = list(Agent$new(RandomPolicy$new(), bandit)

iv. Agent$new(ThompsonSamplingPolicy$new(1.0, 1.0), bandit), Agent$new(UCB1Policy$new(), bandit))
history = Simulator$new(agents, horizon = 100, simulations = 1000)$run()

v. plot(history, type = “cumulative”, use_{colors} = FALSE, $legend_{border}$ = FALSE, $limit_{agents}$ = c(“Random”, “UCB1”,“ThompsonSampling”))

Cumulative Regret for UCB, Random Policy and Thompson Sampling

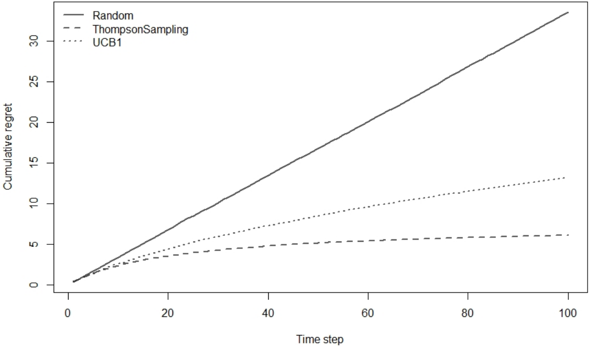


Figure: Regret Paths for UCB, Random and Thompson Sampling

Thank you