ECON7960 User Experience and A/B Test

Hong Kong Baptist University

Topic 11: Contextual Bandit and Textual Analytics

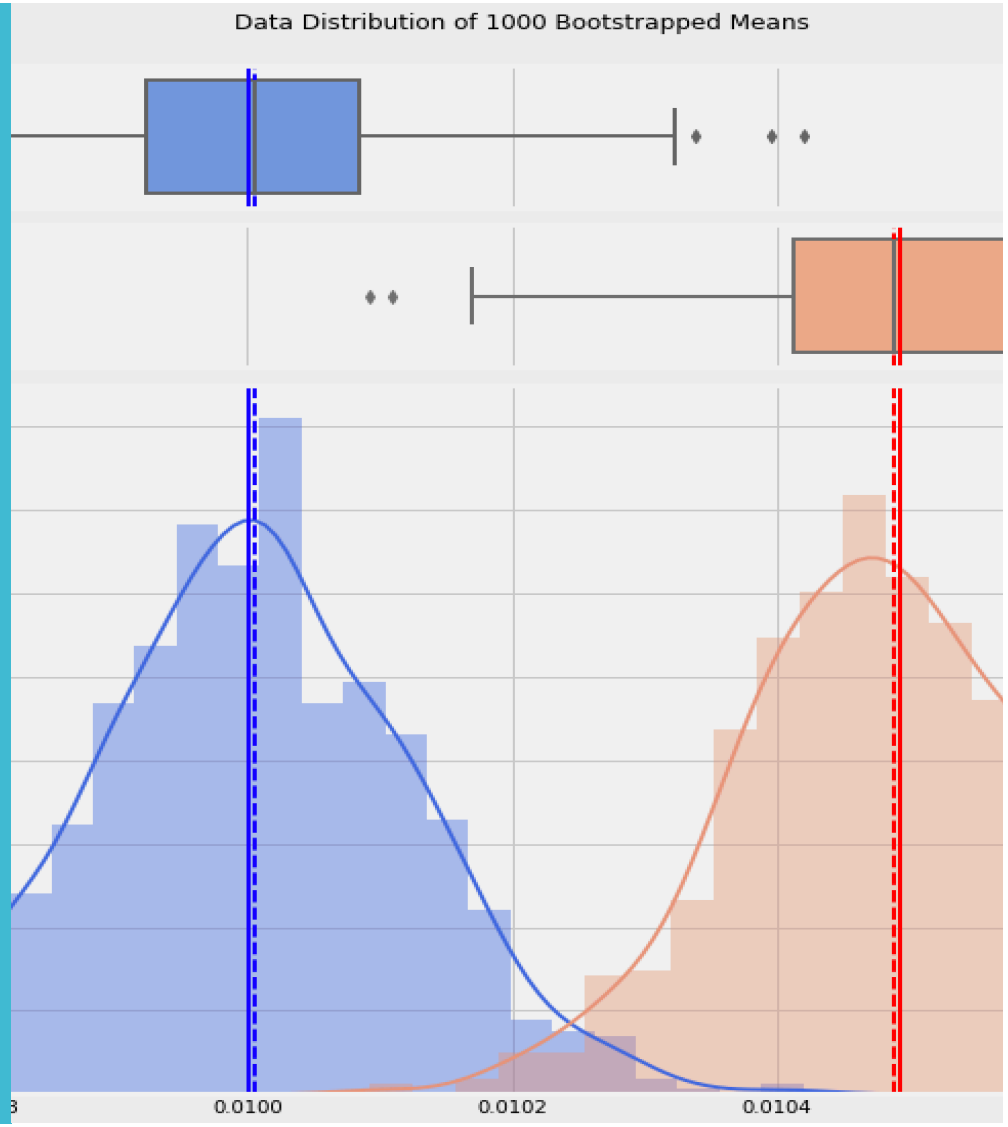
What is a Multi-Bandit Problem

Suppose you are faced with N slot machines (colourfully called multi-armed bandits). Each bandit has an unknown probability of distributing a prize (assume for now the prizes are the same for each bandit, only the probabilities differ). Some bandits are very generous, others not so much. Of course, you don't know what these probabilities are. By only choosing one bandit per round, our task is to devise a strategy to maximize our winnings.

How it relates to this course, about to design a better UX experience digital product services, like an apps or a service through an online platform or an IoT.

In a conventional A/B test, defining what minimum difference between the versions is crucial for the meaning of the test.

In the distributions shown, version A (usually, the current version) has a mean of 0.01. Let's say this is a 1% click-through rate, or CTR. In order to change our website to version B, we want to see a minimum of 5% improvement, or a CTR of at least 1.05%.



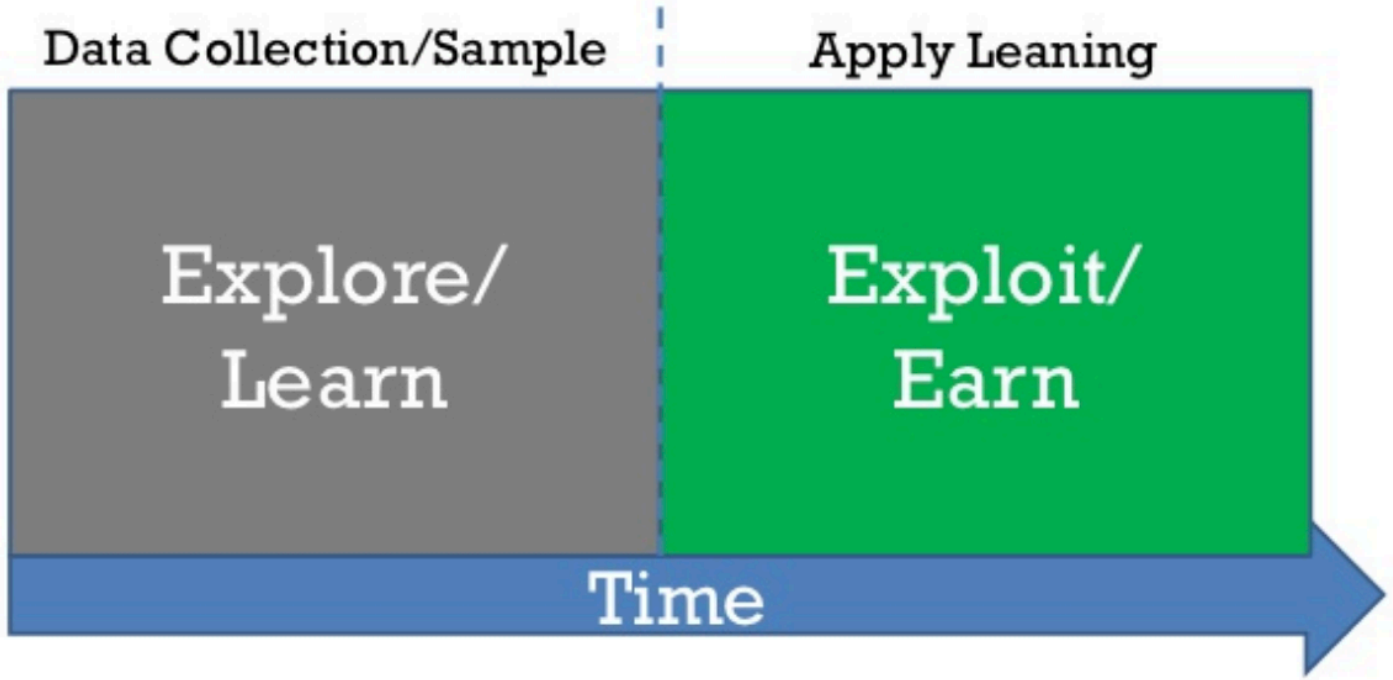
Uncertainty about the Value Differences in Designs

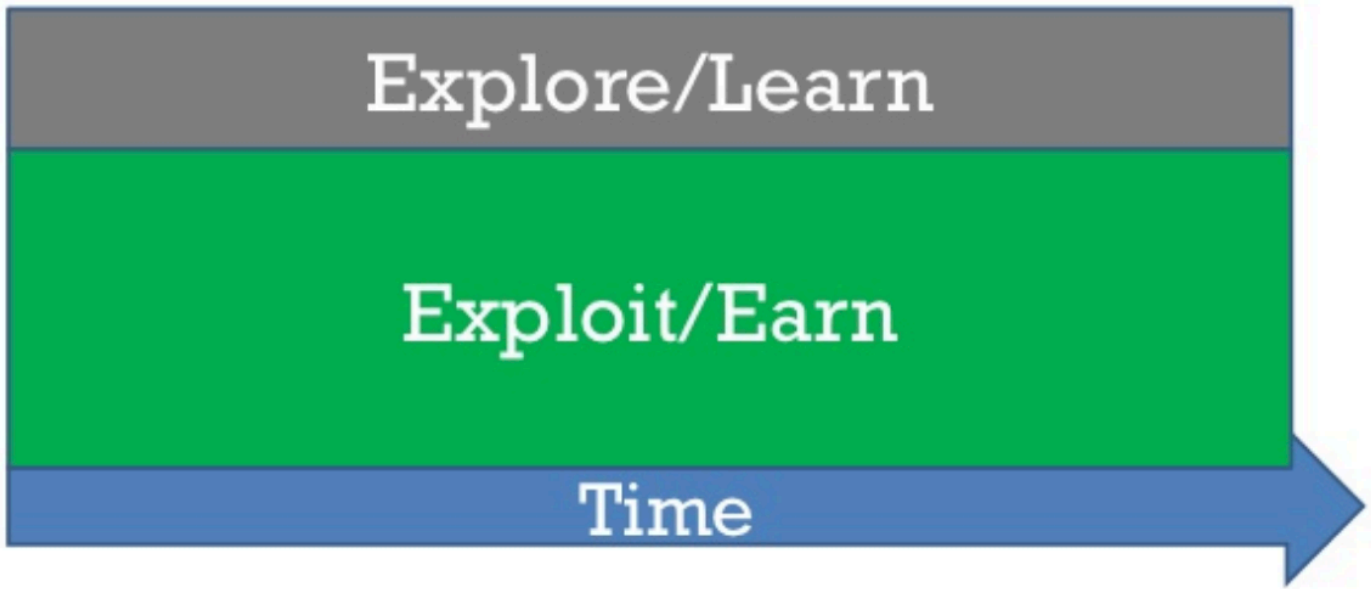
- In any A/B test, set our confidence or significant level, the statistical confidence that our observed results are due to a true difference as opposed to random chance.
- So called **alpha** (significant level) and related to confidence level 95% determine how many observations to collect.
- In power analysis, it determines the required sample size as alpha can be thought of as the acceptable rate of making a Type I error (False Positive) small and **power** can be thought of as the acceptable rate of making a Type II error (False Negative) small.
- As one learn the difference, alpha and beta will determine the size of the sample.
- So you conduct a plan through series of A/B tests to determine the best design along the multi-choices
- (A,B, C, D, E, F. and G, etc.)
- This is a Multi-arm Bandit Problem

Balancing Exploration and Exploitation

- Exploration:
  - Try out each action/option to find the best one, gather more information for long term benefit
- Exploitation:
  - Take the best action/option believed to give the best reward/payoff, get the maximum immediate reward given current information.
- Questions:
  - Select a restaurant for dinner
  - Medical treatment in clinical trials
  - Online advertisement
  - Oil drilling

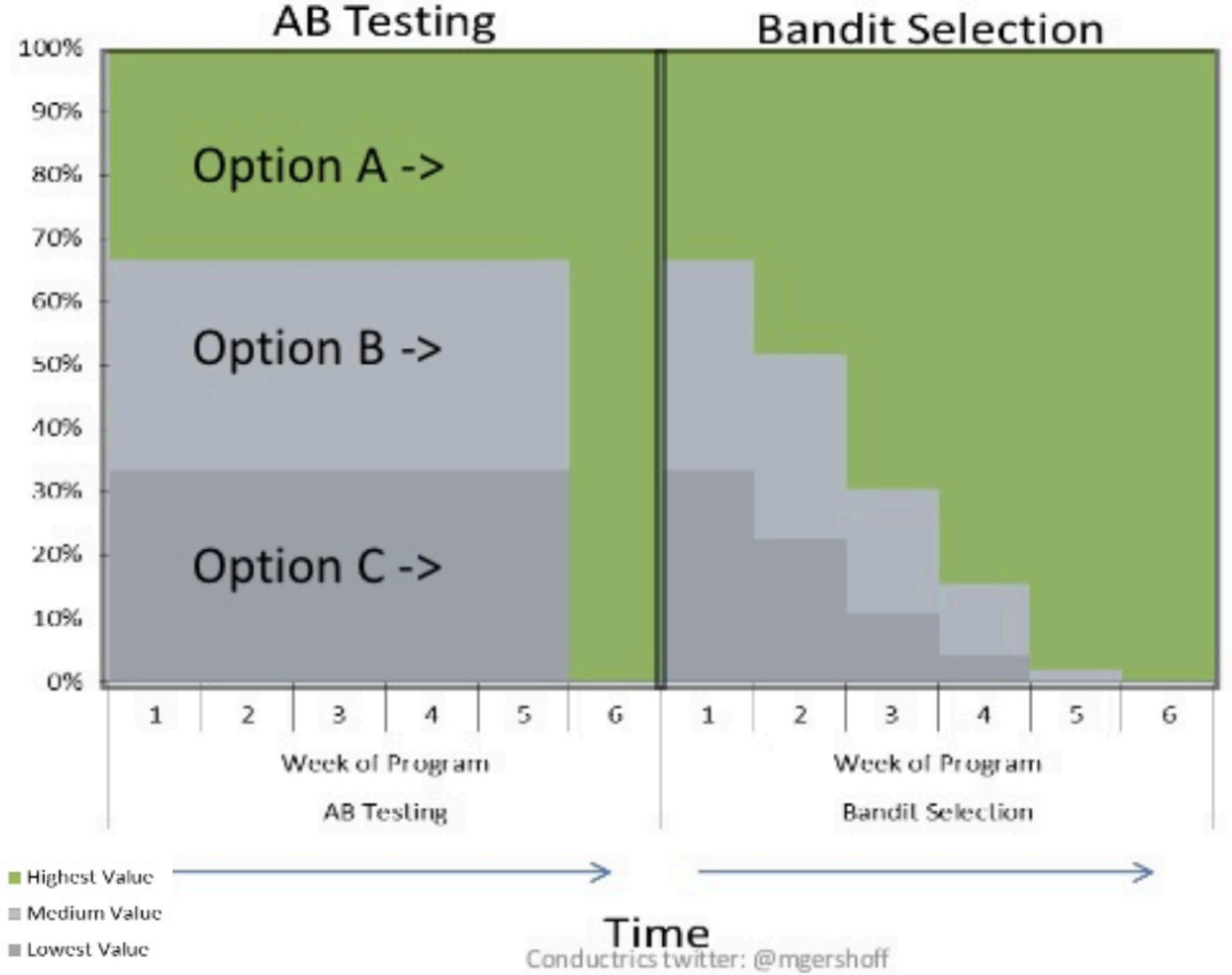
Machine Learning vs Bandit Algorithm





AB Testing V Bandit

AB Test
vs
Bandit
Problem

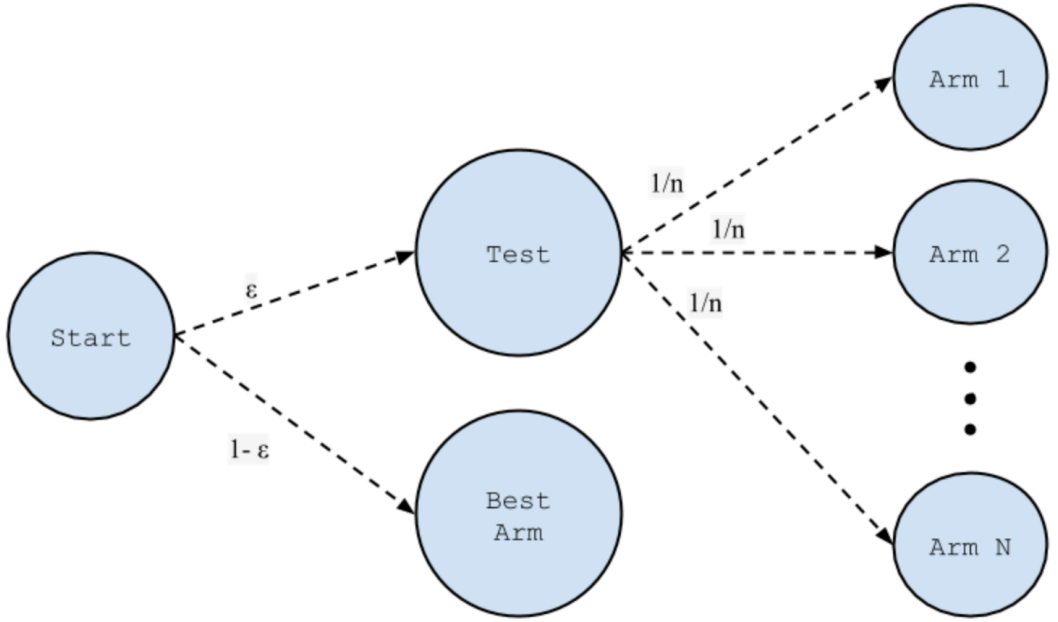


The Naïve Algorithm

- You can think of it as real time A/B/n Testing
- Assign T/K patients to each action a for each arm i, $1 \leq i \leq K$.
- Implement: e.g. round-robin (In sports competitive matches per season, is usually double round-robins. Most association football leagues in the world are organized on a double round-robin basis, in which every team plays all others in its league once at home and once away.) through available actions
- *Implement the round-robin algorithm A/B tests*
- Can we do better to maximize $\Sigma_t r_t$

Weaknesses of ε-Greedy methods:

With probability ε, select an action randomly from all the actions with equal probability.



- Randomly selects an action to explore, does not explore more "promising" actions.
- Does not consider confidence interval. If an action has been taken many times, no need to explore it.

- We consider algorithms that estimate $\hat{Q}_t(a) \approx Q(a)$
- Estimate the value of each action by Monte-Carlo evaluation

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t=1}^{T} r_t \mathbf{1}(a_t = a)$$

- The *greedy* algorithm selects action with highest value

$$a_t^* = \underset{a \in \mathcal{A}}{\text{argmax}}\ \hat{Q}_t(a)$$

- Greedy can lock onto a suboptimal action forever
- $\Rightarrow$ Greedy has linear total regret

- The ϵ-*greedy* algorithm continues to explore forever
  - With probability $1 - \epsilon$ select $a = \underset{a \in \mathcal{A}}{\text{argmax}}\ \hat{Q}(a)$
  - With probability ϵ select a random action
- Constant ϵ ensures minimum regret

$$l_t \geq \frac{\epsilon}{\mathcal{A}} \sum_{a \in \mathcal{A}} \Delta_a$$

- $\Rightarrow$ ϵ-greedy has linear total regret

Algorithm discussed in LAB4

Multi-Bandit Problem

The Bandit problem has many real-world applications, including website optimization, clinical trials, adaptive routing and financial portfolio design - a smarter A/B testing.

The trade-off of exploration/exploitation can be formulated as the choices/balance between trying different bandits to learn how to get more from an expected sum of payoff choosing higher pay-out machine. The best bandit algorithm is to solve the problem of multi-bandits (we learn one in last lecture).

It is argued that based on past performance, Naïve Algorithm, is not the best strategy. One may end up in the suboptimal path and the initial rewards is favourable, and miss the long run benefits.

ε-Greedy Algorithm improves and even decay ε-Greedy Algorithm better

- The *action-value* is the mean reward for action a,

$$Q(a) = \mathbb{E}[r|a]$$

- The *optimal value* V^* is

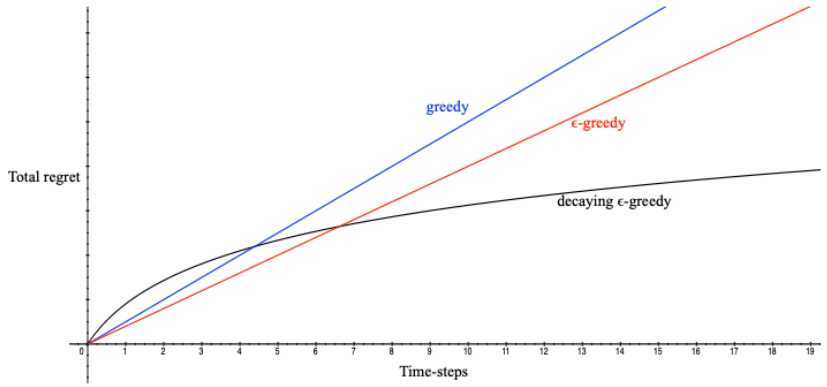
$$V^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a)$$

- The *regret* is the opportunity loss for one step

$$l_t = \mathbb{E}[V^* - Q(a_t)]$$

- The *total regret* is the total opportunity loss

$$L_t = \mathbb{E}\left[\sum_{\tau=1}^{t} V^* - Q(a_\tau)\right]$$



There are more Multi-armed Bandits:

The performance of any algorithm is determined by similarity between optimal arm and other arms

The more uncertain we are about an action-value, the more important it is to explore that action. It could turn out to be the best action

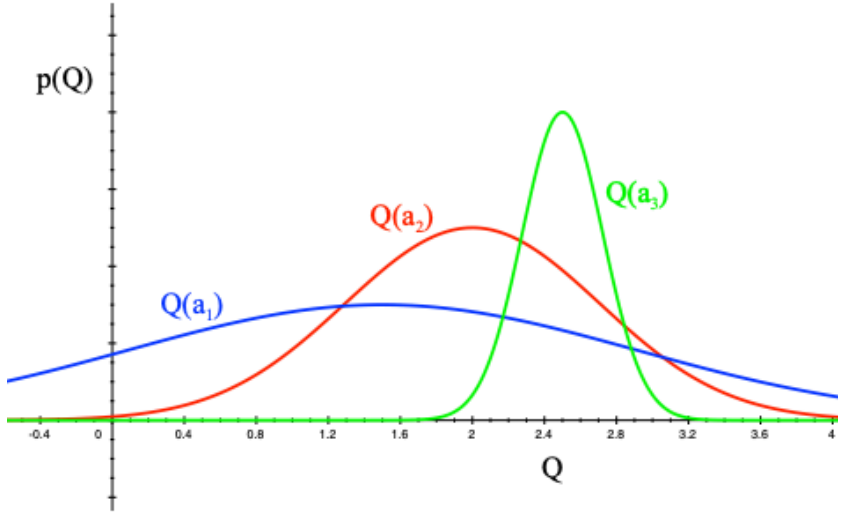
If you take the blue action path, the reward distribution will drive to move to another arm. If you take the green action path, the reward distribution will drive you keep on the path.

Decaying ε_t-Greedy Algorithm

- Pick a decay schedule for $\epsilon_1, \epsilon_2, \ldots$
- Consider the following schedule

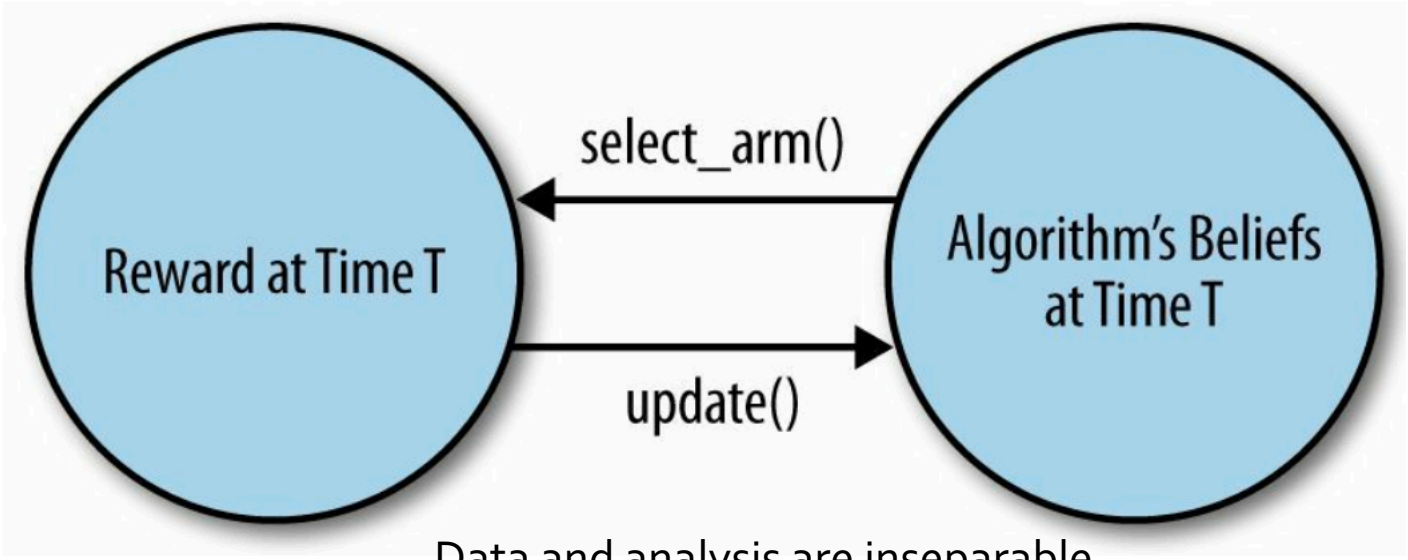
$$c > 0$$
$$d = \min_{a|\Delta_a > 0} \Delta_i$$
$$\epsilon_t = \min\left\{1, \frac{c|\mathcal{A}|}{d^2 t}\right\}$$

- Decaying ϵ_t-greedy has *logarithmic* asymptotic total regret!
- Unfortunately, schedule requires advance knowledge of gaps
- Goal: find an algorithm with sublinear regret for any multi-armed bandit (without knowledge of $\mathcal{R}$)



"Bandit algorithms exemplify two types of learning that are not present in standard ML examples: active learning, which refers to algorithms that actively select which data they should receive; and online learning, which refers to algorithms that analyze data in real-time and provide results on the fly."

White, John Myles. Bandit Algorithms for Website Optimization



Data and analysis are inseparable

- Several way to assess the performance of algorithms
  - Track the Probability of Choosing the Best Arm
  - Track the Average Reward at Each Point in Time
  - Track the Cumulative Reward at Each Point in Time

Other Famous Bandit algorithms: Softmax

i weight the probability with $\exp(rA / tau) / \Sigma(\exp(r_i / tau))$

White, John Myles. Bandit Algorithms

- An obvious flaw in epsilon-Greedy is that it explores completely at random. If we have two arms with very similar rewards, we need to explore a lot to learn which is better and so choose a high epsilon. However, if our two arms have vastly different rewards (and we don't know this when we start the experiment, of course), we would still set a high epsilon and waste a lot of time on the lower paying reward. The Softmax algorithm (and its annealed counterpart) attempt to solve this problem by selecting each arm in the explore phase roughly in proportion to the currently expected reward.

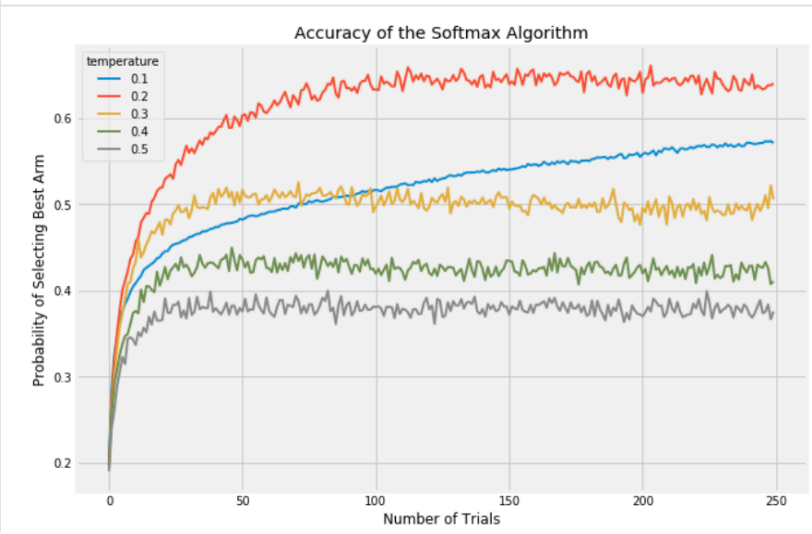
- For example, in one scenario (call it Scenario A), you might have two arms, one of which rewards you 10% of the time and the other rewards you 13% of the time. In Scenario B, the two arms might reward you 10% of the time and 99% of the time. In both of these scenarios, the probability that the epsilon-Greedy algorithm explores the worse arm is exactly the same (it's epsilon / 2), despite the inferior arm in Scenario B being, in relative terms, much worse than the inferior arm in Scenario A.

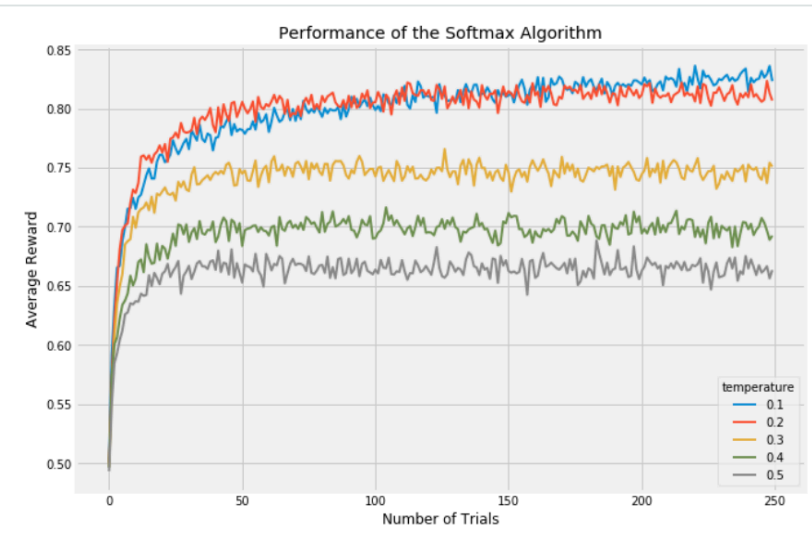
Other Famous Bandit algorithms: Softmax

i weight the probability with exp(rA / tau) / Σ(exp(r_i / tau))

- At one extreme, we set tau = oThis will give us a fully deterministic choice of the arm that has the highest estimated value. At the other extreme, we set tau = ∞, which gives us purely random exploration like we got out of the epsilon-Greedy algorithm.

- White, John Myles. Bandit Algorithms for Website Optimization:





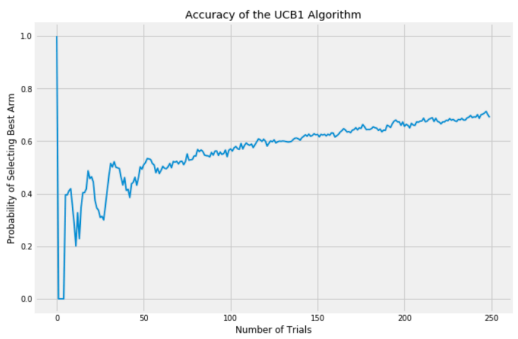
Other Bandit algorithms UCB

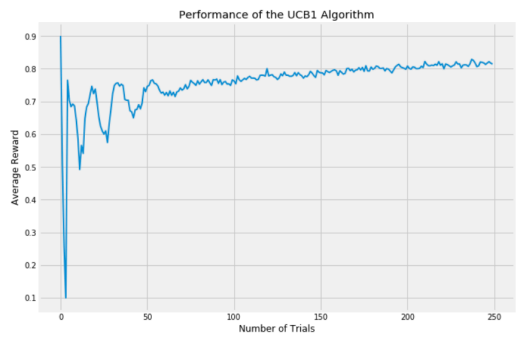
Both the epsilon-Greedy algorithm and the Softmax algorithm share same properties:

The algorithm's default choice is to select the arm that currently has the highest estimated value.

The algorithm sometimes decides to explore and chooses an option that isn't the one that currently seems best.

- Even if have prior belief that some under-explore arms which may have a high pay-out even though they don't have enough data to be confident.
- Taking into account on how much one knows about each arm and encourage an algorithm to slightly favor those arms of which we don't have high confidence in their behavior, so that we can learn more.





- This leads to the UCB1 algorithm

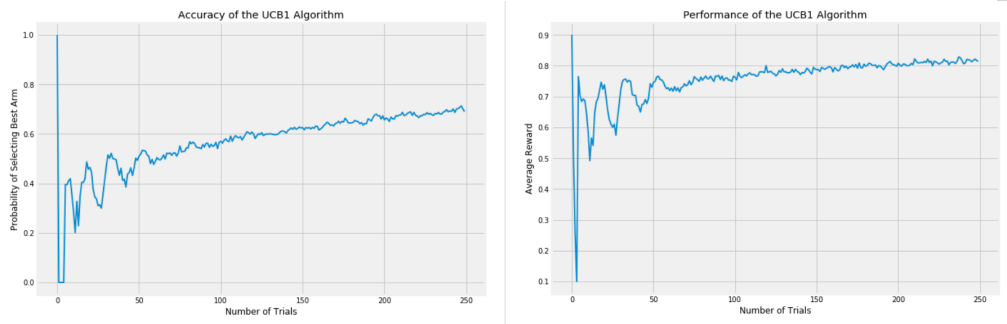
$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

Other Bandit algorithms UCB

Both epsilon-Greedy and Softmax algorithms are gullible, easily misled by a few negative experiences but their use of randomness, make up for this mistakes.

UCB takes a very different approach.

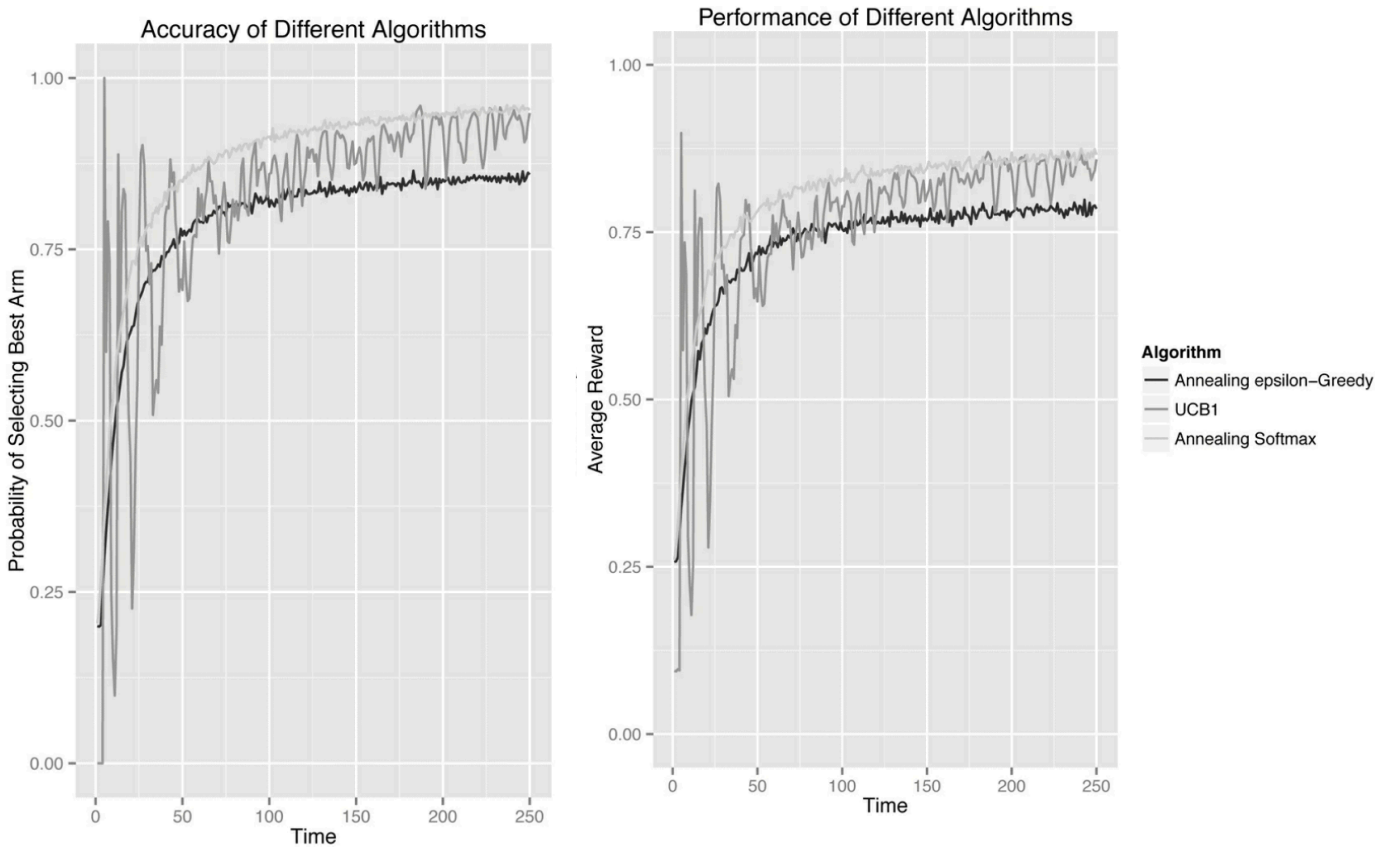
- the UCB algorithm is special for two other reasons:
- UCB doesn't use randomness at all. Unlike epsilon-Greedy or Softmax, it's possible to know exactly how UCB will behave in any given situation. This can make it easier to reason about at times.
- UCB doesn't have any free parameters that you need to configure before you can deploy it meaning that you can start to use UCB without having a clear sense of what you expect the world to behave like.



- This leads to the UCB1 algorithm

$$a_t = \underset{a \in \mathcal{A}}{\text{argmax}}\ Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

The backpedaling matters less and less over time, but it's always present in UCB's behavior, which means that UCB doesn't become a strictly greedy algorithm even if you have a huge amount of data.



More than Simple Multi-Bandit Problem

Developing personalization of user experience for your website or an app, such as

- able to choose which content to display to the user,
- rank advertisements to the appropriate person;
- optimize search results with respect to the browser,
- select the best image to show on the page; and
- recommend movies or songs to fit your taste.

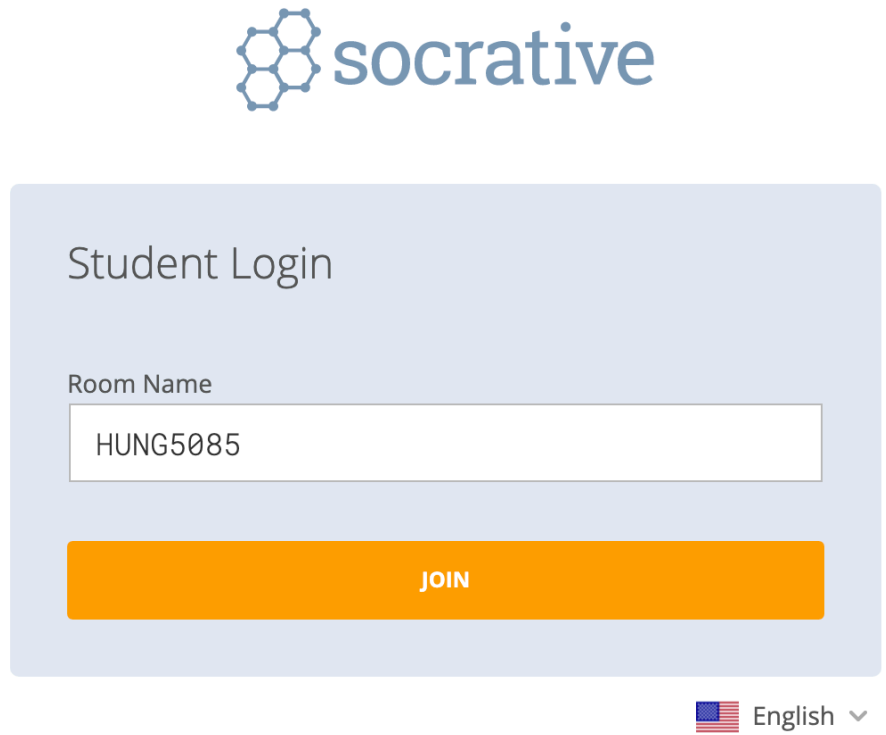
It needs more than simple bandit problem, leading to the contextual bandit algorithms, a kind of simple reinforcement learning.

An introduction on RL

- Please use your phone to download an app or web http://www.socrative.com ("SOCRATIVE" student version) and open it, you should see

Enter the Room Name "HUNG5085"

Please enter the Room and get together to play the LAB5

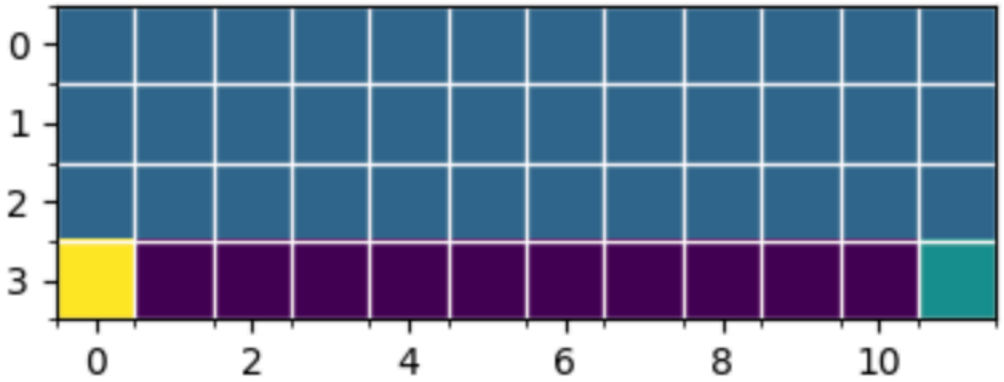


https://github.com/cmivictorius/ECON7960/blob/master/prog/LAB5%20Idea%20of%20RL%20%26%20Contextual%20Bandit.ipynb

LAB5

Simple RoomsEnv

CliffwalkingEnv



In last topic, we introduce Bandits Algorithms

- All these topics are related about how find a right redesign to achieve the "best" user experience, interplay with action and reward.
- In real time, the state of the world changes, 4 scenarios were mentioned in decision when under uncertainty

	Actions don't change the state of the world	Actions change state of the world
Learn Model of Outcomes	Multi-armed bandits	Reinforcement Learning
Given Model of Stochastic Outcomes	Decision theory (most classical economics)	Markov Decision Process (MDP)

Contextual Bandit

A Contextual Bandit Case

- Suppose you face a 2-armed bandit task whose true action values change randomly from time step to time step. Specifically, suppose that, for any time step, the true values of actions 1 and 2 are respectively 0.1 and 0.2 with probability 0.5 (case A), and 0.9 and 0.8 with probability 0.5 (case B). If you are not able to tell which case you face at any step, what is the best expectation of success you can achieve and how should you behave to achieve it? Now suppose that on each step you are told whether you are facing case A or case B (although you still don't know the true action values). This is an associative search task. What is the best expectation of success you can achieve in this task, and how should you behave to achieve it?

- Sutton, Richard S.,Barto, Andrew G.. Reinforcement Learning

Locating the best reward in taking choices in searching process.

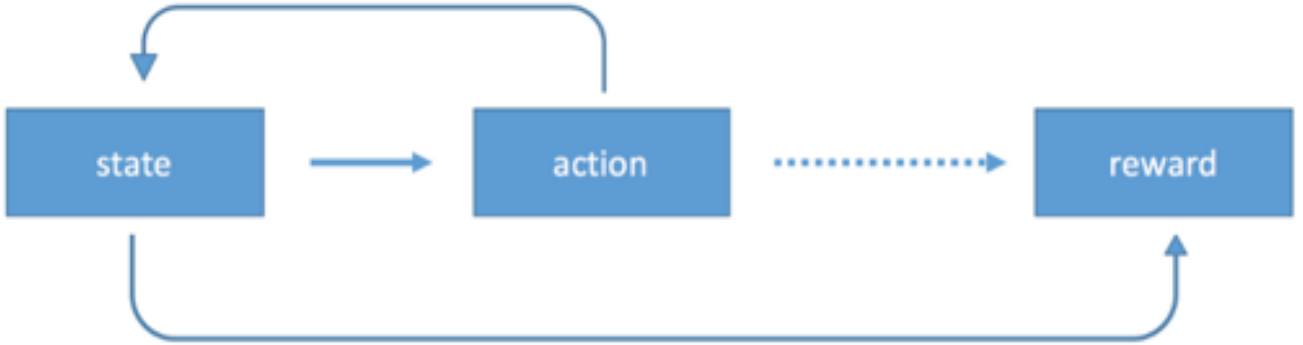
In static world, learning leads to find the optimal one.
In a dynamic world, learning leads to find the contemporary optimal one at one moment.

The multi-armed bandit algorithm outputs an action but doesn't use any information about the state of the environment (context)

Multi-armed Bandit

The contextual bandit extends the model by making the decision conditional on the state of the environment.

Contextual Bandit



Full RL Problem

A reinforcement problem algorithm that an action affects the state and affect the actions, ultimately lead to a reward, like play chess, move and respond (state) interact to the end to determine who win or lose

Contextual Bandits

- For example, a pat online shop will show an image of a cat to a cat person, and an image of a dog to a dog person, one may show different images at different times of the day and days of the week.

- The algorithms observe a context (state), make a decision, choosing one action from a number of alterative actions, and observe an outcome of that decision. An outcome (state, action) define a reward.

- The context is information about the user (personalised state): such as device information, geolocation, etc. Not by collecting all these feature information from the past and to estimate the average response with respect to the persona, then build or design a model to respond with respect to the visitor closed to the features. The learning is online.

Contextual Bandit

A Contextual Bandit Algorithm

LinUCB [Li et al]

- Assume linear relation between rewards and arms
  - Arms have an embedding $x_{t,a} = \phi(s_t, a)$
  - Reward $\mathbb{E}[r_{t,a} \mid x_{t,a}] = \theta_a \cdot x_{t,a}$

- Idea: Use ridge regression for $\hat{\theta}_a$ using $\left(A: \sum x_{t,a} x_{t,a}^T, b: \sum r_{t,a} \cdot x_{t,a}\right)$
- $\hat{\theta}_a = A^{-1} b$
- Add exploration bonus

$$\underset{a}{argmax} \quad \hat{\theta}_a \cdot x_{t,a} + \alpha \sqrt{x_{t,a}^T A^{-1} x_{t,a}}$$

Mean

Conf.Interval

http://proceedings.mlr.press/v15/chu11a/chu11a.pdf