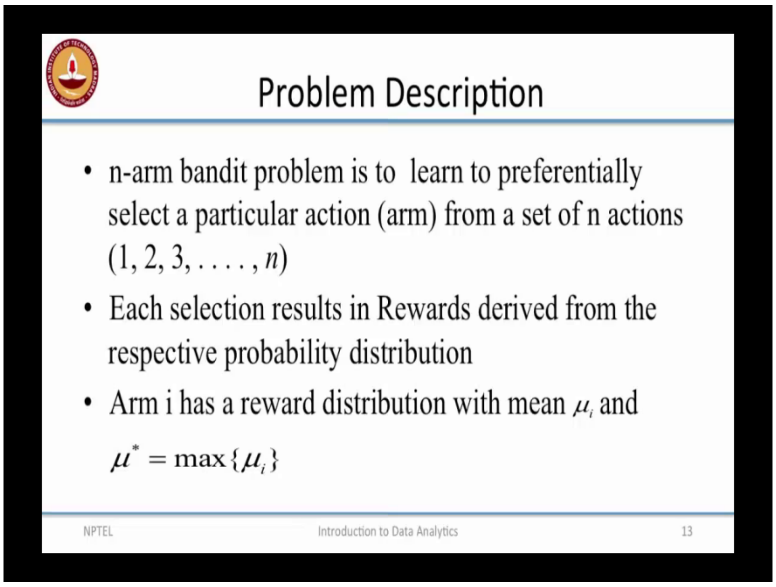
Introduction to Data Analytics

Prof. Nandan Sudarsanam and
Prof. B. Ravindran
Department of Management Studies and
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module – 08
Lecture – 46
An Introduction to Online Learning - Reinforcement Learning

(Refer Slide Time: 00:14)



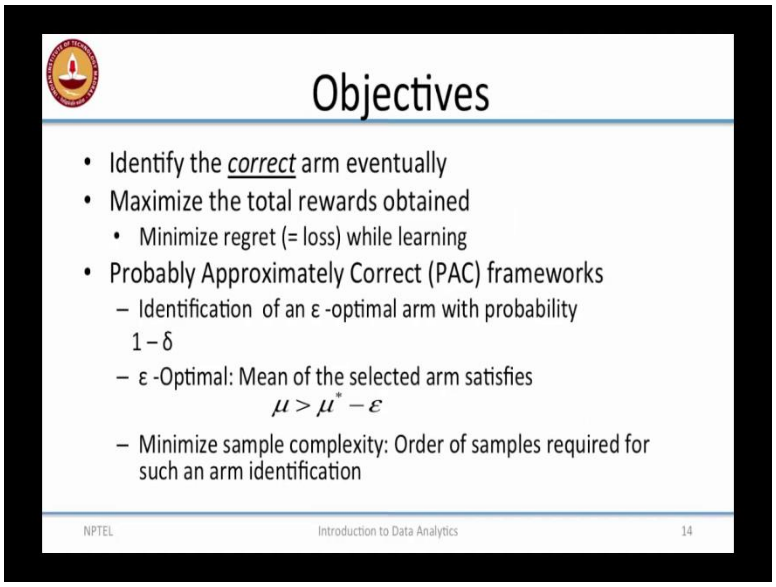
Problem Description

- n-arm bandit problem is to learn to preferentially select a particular action (arm) from a set of n actions $(1, 2, 3, \ldots, n)$
- Each selection results in Rewards derived from the respective probability distribution
- Arm i has a reward distribution with mean μ_i and $\mu^* = \max\{\mu_i\}$

NPTEL Introduction to Data Analytics 13

Hi, so in the previous module we started looking at the multi arm bandit problems and we will continue looking at some of the solution approaches for multi arm bandits. First you recall, in this case we have n possible outcomes and we have to select one of the actions from this set of n actions and each action results in a reward derived from the probability distribution associated with the action, yet our goal is to find the most profitable action.

(Refer Slide Time: 00:46)



Objectives

- Identify the *correct* arm eventually
- Maximize the total rewards obtained
  - Minimize regret (= loss) while learning
- Probably Approximately Correct (PAC) frameworks
  - Identification of an ε -optimal arm with probability $1-\delta$
  - ε -Optimal: Mean of the selected arm satisfies
$$\mu > \mu^* - \varepsilon$$
  - Minimize sample complexity: Order of samples required for such an arm identification

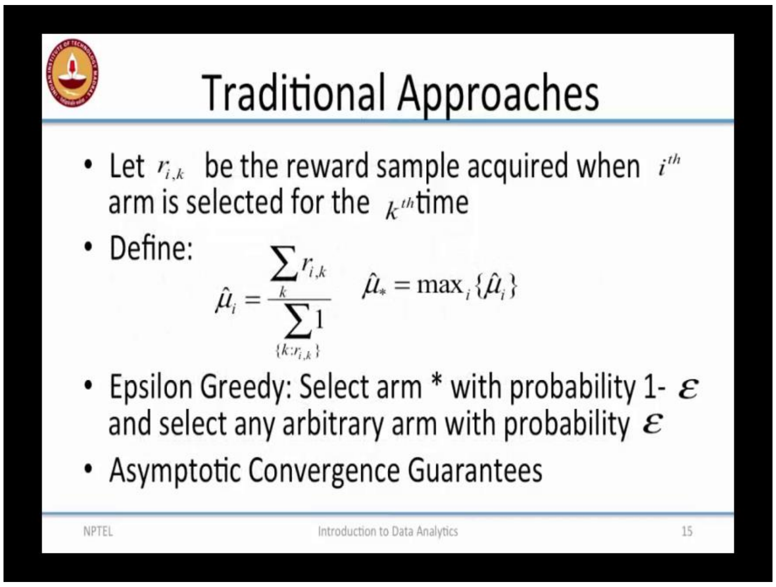
So, the goal could have different shades to it, so we are interested in finding the best arm, but when or how are you interested in finding the best arm. So, the simplest notion of solving the bandit problem is to identify the correct arm eventually, regardless of how many ever attempts we take as long as you can give a guarantee that eventually I will be trying the best arm that is sufficient. But, then if you want to get more practical algorithms we have to look at other notions of solving the problem.

So, look at 2 here, so first one is we would like to maximize the total rewards that you obtain not just the rewards that you obtain eventually, but also the rewards that you obtain during the course of learning. So, this is called sometimes minimizing the loss or regret while learning. So, the regret is how much better you could have done, if you had known the true solution from the beginning.

So, that quantity is known as the regret and you would like to minimize the regret and the second solution concept that people look at is called probably approximately correct solutions. So, I am interested in finding an ε optimal arm, the ε optimal arm, is such that the mean of the selected arm mu is within ε of mu star, which is the expected reward for the best arm. I am going to find an ε optimal arm that gives me the approximately correct part and then, I am going to say that my algorithm will surely return the ε optimal arm in 1 - δ fraction of the problem instances.

So, the probability 1 - δ by algorithm we give you an ε optimal arm. So, for fixed values of ε and δ, we try to figure out how many samples do I need before I can give such a guarantee. So, the probably approximately correct framework that probably comes from the factor that I am interested in returning solutions with probability 1 - δ and approximately correct, because I am interested in returning solutions within ε of the true solution.

(Refer Slide Time: 03:25)



Traditional Approaches

- Let $r_{i,k}$ be the reward sample acquired when i^{th} arm is selected for the k^{th} time
- Define:

$$\hat{\mu}_i = \frac{\sum_k r_{i,k}}{\sum_{\{k: r_{i,k}\}} 1} \qquad \hat{\mu}_* = \max_i \{\hat{\mu}_i\}$$

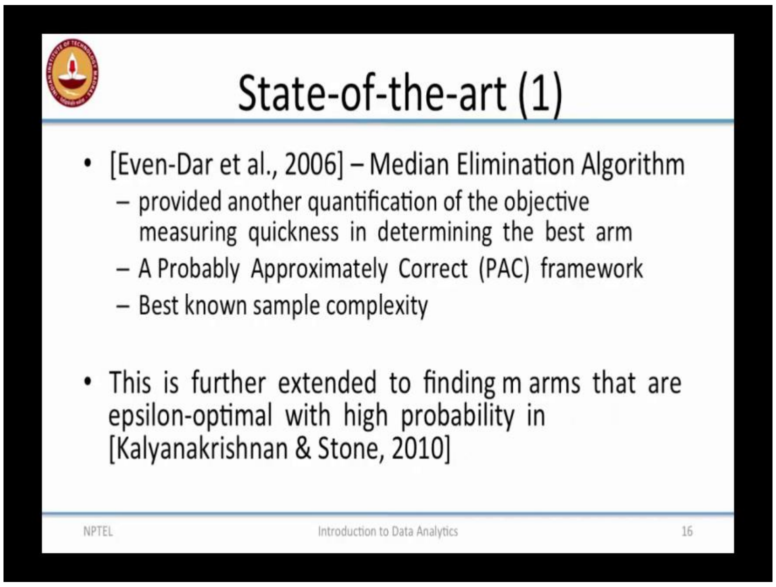
- Epsilon Greedy: Select arm * with probability 1- ε and select any arbitrary arm with probability ε
- Asymptotic Convergence Guarantees

NPTEL Introduction to Data Analytics 15

So, there was that, the most simple traditional approach is to essentially pull the arms multiple times, keep track of the reward that you get over time. So, I would estimate $\hat{\mu}_i$ as the sum of the rewards I get and I pulled the arm i divided by the total number of times I have pulled arm i, that gives me an estimate for the average reward I expect to get when I pull arm i. And we know that, this is an unbiased estimator of the reward and hence with the enough samples as time goes to infinity, these will convergence to the true expectations and I can derive the best arm, I just looking at the arm with the maximum expectation.

To ensure that we select the arms infinitely often, we use an ε greedy strategy. So, you select the arm that you currently think is the best arm, the arm star that gives you the max over all $\hat{\mu}_i$. You select that arm with the probability of 1 - ε and ε is some small constant like 0.1 or something and you can select any arbitrary arm with the probability ε. So, this guarantees that every arm will get selected and therefore, as the number of plays goes to infinity, then every arm gets selected infinitely often. Therefore, asymptotic convergence guaranteed for these kinds of ε greedy action selection.

(Refer Slide Time: 05:12)



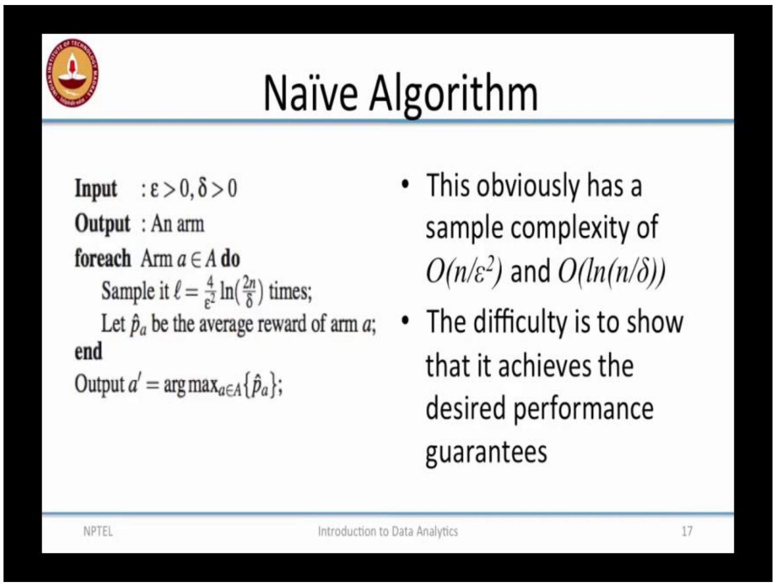
State-of-the-art (1)

- [Even-Dar et al., 2006] – Median Elimination Algorithm
  - provided another quantification of the objective measuring quickness in determining the best arm
  - A Probably Approximately Correct (PAC) framework
  - Best known sample complexity
- This is further extended to finding m arms that are epsilon-optimal with high probability in [Kalyanakrishnan & Stone, 2010]

NPTEL Introduction to Data Analytics 16

The one of the state of the art algorithms that you will look at is called Median Elimination algorithm, so this is a PAC algorithm. So, it gives you probably approximately correct guarantees and it tries to control the sample complexity, it is got one of the best known sample complexities for solving a multi arm bandit problem.

(Refer Slide Time: 05:40)



Naïve Algorithm

```
Input  : $\varepsilon > 0, \delta > 0$
Output : An arm
foreach Arm $a \in A$ do
    Sample it $\ell = \frac{4}{\varepsilon^2}\ln\left(\frac{2n}{\delta}\right)$ times;
    Let $\hat{p}_a$ be the average reward of arm $a$;
end
Output $a' = \arg\max_{a \in A}\{\hat{p}_a\}$;
```

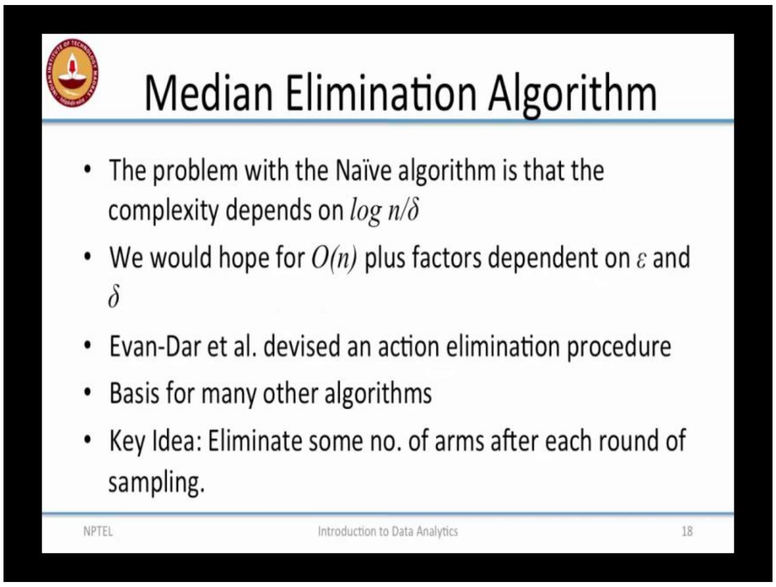
- This obviously has a sample complexity of $O(n/\varepsilon^2)$ and $O(\ln(n/\delta))$
- The difficulty is to show that it achieves the desired performance guarantees

NPTEL Introduction to Data Analytics 17

So, let us to get an intuition into how these things work, let us start by looking at a simple algorithm, which I will call the naive algorithm. So, you start off by sampling each arm l number of times, where l is given by the expression here. So, it is $\frac{4}{\varepsilon^2}\ln\frac{2n}{\delta}$. So, n is the number of arms, the ε and δ or the constants from your PAC results. So, you sample an arm, each arm actually l number of times and let $\hat{p}_a$ be the average reward of arm a.

So, the output, this is essentially the arm that has the maximum of all the arms in the expected reward sense. So, this is; obviously, has a sample complexity of $O\left(\frac{n}{\varepsilon^2}\right)$ and $O\left(\ln\left(\frac{n}{\delta}\right)\right)$. The real difficulty is to show that, it actually achieves the desired performance guarantee that is with the very high probability of 1 - δ you will return the ε optimal arm and define with only probability δ. So, we will not going to, we are not going to get in to the mechanics of the proofs, it is quite beyond the scope of this course, but just I am pointing out that you can come of this simple algorithm and you can often show that they achieve the guarantees that you want.

(Refer Slide Time: 07:06)


Median Elimination Algorithm

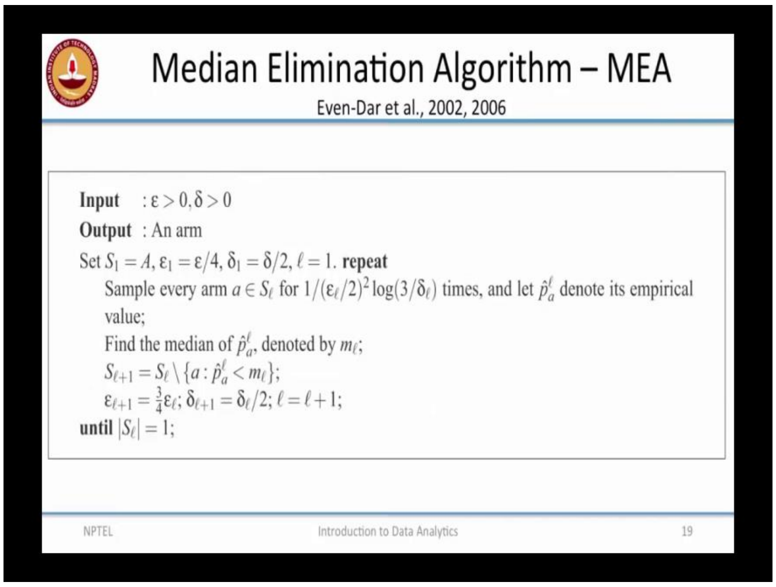
- The problem with the Naïve algorithm is that the complexity depends on $log\ n/\delta$
- We would hope for $O(n)$ plus factors dependent on ε and δ
- Evan-Dar et al. devised an action elimination procedure
- Basis for many other algorithms
- Key Idea: Eliminate some no. of arms after each round of sampling.

NPTEL Introduction to Data Analytics 18

So, one of the drawbacks of the naive algorithm is that the complexity depends on $\log\frac{n}{\delta}$. So, you cannot avoid the order n terms, because you have to pull each arm at least once, but you want the additional factors to be dependent only on ε and δ you do not to have an additional dependence on n in your algorithm. So, Evan-Dar et al. devised an action elimination procedure, which allows you to be pretty efficient in your finding the optimal arm.

It can also serve as the basis for many other algorithms and therefore, I spend a couple of minutes looking at the median elimination algorithm. The key idea here is that, after some number of samples you are going to eliminate some arms and then, you are going to continue sampling with the rest of the arms.

(Refer Slide Time: 07:53)



Median Elimination Algorithm – MEA

Even-Dar et al., 2002, 2006

```
Input  : $\varepsilon > 0, \delta > 0$
Output : An arm
Set $S_1 = A$, $\varepsilon_1 = \varepsilon/4$, $\delta_1 = \delta/2$, $\ell = 1$. repeat
    Sample every arm $a \in S_\ell$ for $1/(\varepsilon_\ell/2)^2 \log(3/\delta_\ell)$ times, and let $\hat{p}_a^\ell$ denote its empirical value;
    Find the median of $\hat{p}_a^\ell$, denoted by $m_\ell$;
    $S_{\ell+1} = S_\ell \setminus \{a : \hat{p}_a^\ell < m_\ell\}$;
    $\varepsilon_{\ell+1} = \frac{3}{4}\varepsilon_\ell$; $\delta_{\ell+1} = \delta_\ell/2$; $\ell = \ell + 1$;
until $|S_\ell| = 1$;
```

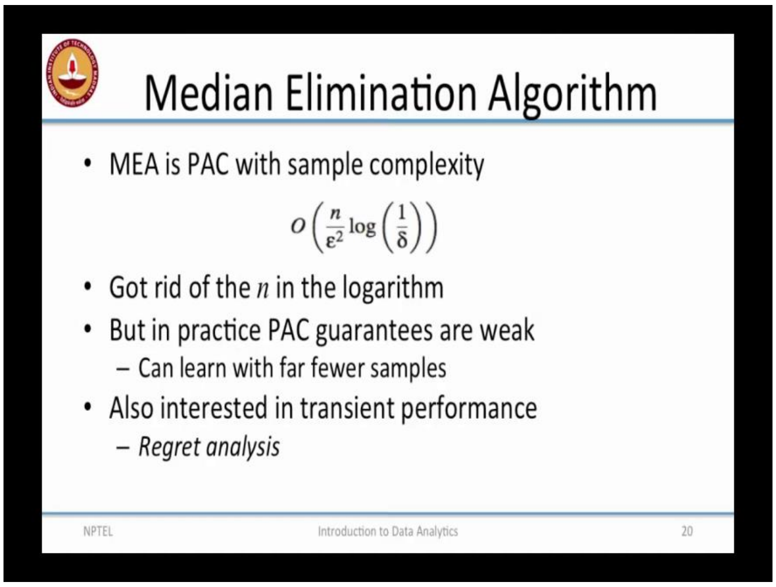
NPTEL Introduction to Data Analytics 19

So, here is the algorithm, it looks a little complex, but it is nothing really it is fairly straight forward. So, you start of by having a set of all the arms available to you start off with an ε and a δ and start off with l = 1 it means that your sampling l equal to 1, which is essentially in first round. So, here now you sample every arm that is left. So, in the first stage it will be all the arm set this many number of times $\frac{1}{\left(\frac{\varepsilon_1}{2}\right)^2}\log\left(\frac{3}{\delta_1}\right)$ and let $\hat{p}_a^1$ denote its empirical value.

So, now, what we have done taken the arms you are pulled each of them some number of times and from that you have formed an estimate of the average in one of the arm. Now, what we do you arrange these averages in some sorted order, let us say you arrange the averages in ascending order and then, you take the first half the half that is below the median. So, the half of the data that is below the median you take those and you eliminate them.

And now, what you are left with you have half the arms, which had the highest possible expected mean from the previous round. So, now, you repeat the whole thing again except that you make your ε and your δ even smaller. So, with the little bit of algebra you can actually show that this gives you nice performance guarantees it gives you a PAC results, so gives you the ε δ guarantees.

(Refer Slide Time: 09:48)



Median Elimination Algorithm

- MEA is PAC with sample complexity

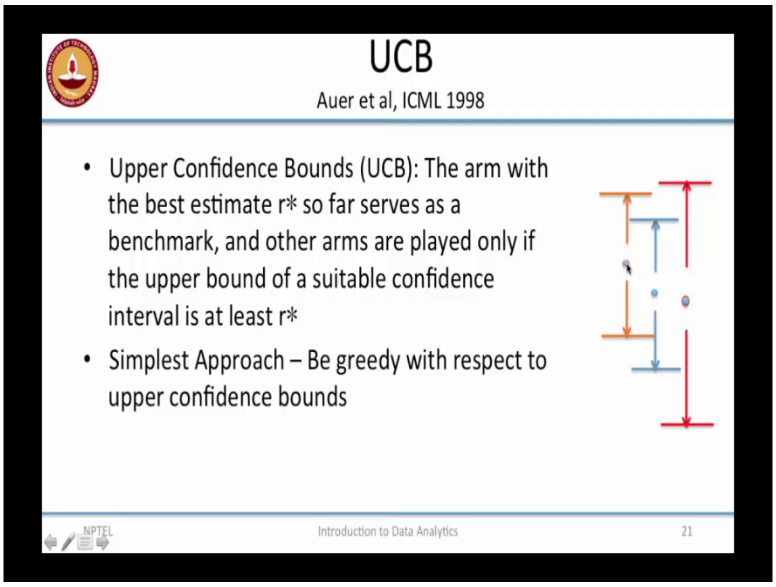
$$O\left(\frac{n}{\varepsilon^2}\log\left(\frac{1}{\delta}\right)\right)$$

- Got rid of the n in the logarithm
- But in practice PAC guarantees are weak
  - Can learn with far fewer samples
- Also interested in transient performance
  - *Regret analysis*

NPTEL Introduction to Data Analytics 20

In addition to that you can also show that it has a sample complexity of $O\left(\frac{n}{\varepsilon^2}log\left(\frac{1}{\delta}\right)\right)$ we go off by $\varepsilon^2 log\left(\frac{1}{\delta}\right)$. So, you got of the n in the logarithm that is a big step and it terms on that it is very hard to improve on median elimination algorithm at least using these kinds of process that we are looking at. But, in practice this kind of PAC guarantees are weak and typically you can get acceptable performance with far fewer samples and the PAC algorithm also do not cared about, how your performance varies while you are learning. And hence increasingly regret analysis of data is becoming more and more problem.

(Refer Slide Time: 10:34)



UCB

Auer et al, ICML 1998

- Upper Confidence Bounds (UCB): The arm with the best estimate r* so far serves as a benchmark, and other arms are played only if the upper bound of a suitable confidence interval is at least r*
- Simplest Approach – Be greedy with respect to upper confidence bounds

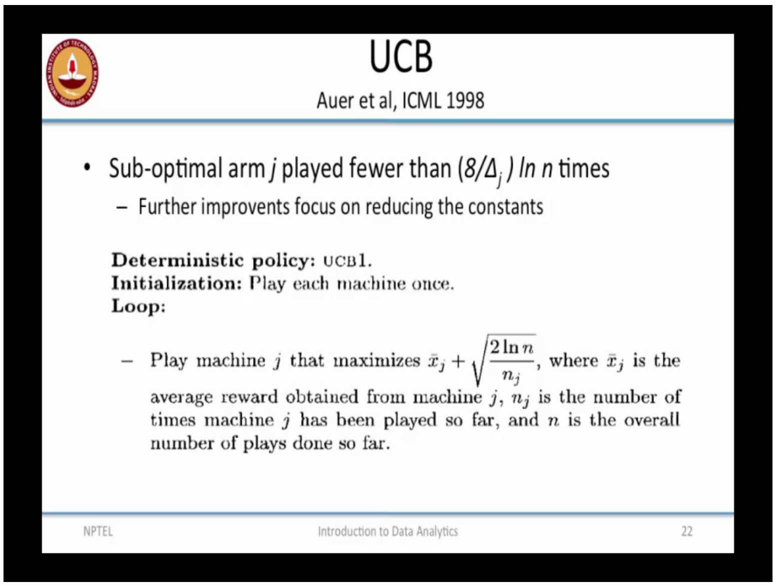
NPTEL Introduction to Data Analytics 21

So, here is a very simple algorithm that can be shown to give very good regret performance. So, you not only maintain the best estimates or the estimate of the rewards that you have, so, far, but you also maintain some kind of an uncertainty interval around the reward. So, look at this orange curve here. So, this is the mean reward I have, but then, I also have would uncertainty or some kind of a confidence interval here that says that a the true reward with the very high confidence would be within this interval.

So, you can see that for some arms even though the estimated mean rewards are lower their uncertainty is much higher. And, so what we really are looking for is, so if you have the best estimate arm and the other arms played only of the upper bound of a suitable confidence interval is at least r one of the simplest approaches could be just be greedy with respect to upper confidence bounds you estimate the upper confidence bounds.

And when you take actions that have the highest upper confidence bounds with; obviously, going to the something higher than essentially this case you would play the arm red as supposed to orange and blue even though orange has the higher expected mean.

(Refer Slide Time: 12:05)



UCB

Auer et al, ICML 1998

- Sub-optimal arm *j* played fewer than $(8/\Delta_j)\ ln\ n$ times
  - Further improvents focus on reducing the constants

Deterministic policy: UCB1.
Initialization: Play each machine once.
Loop:

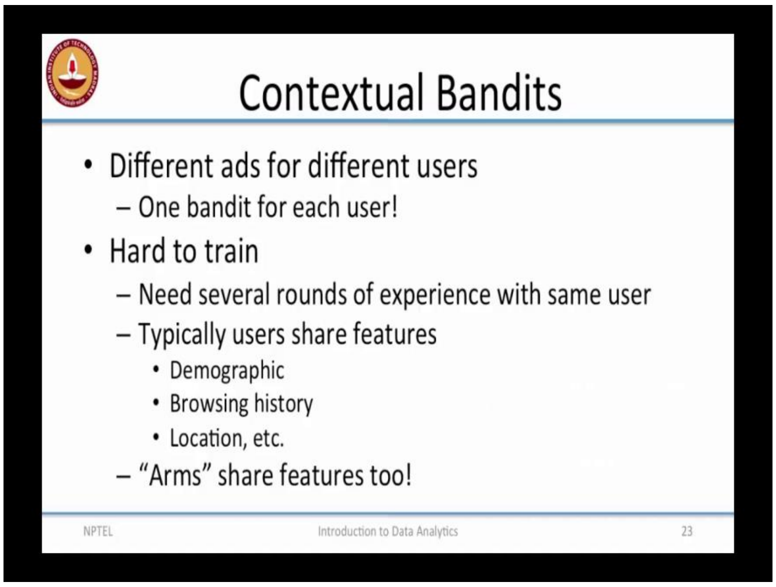
- Play machine j that maximizes $\bar{x}_j + \sqrt{\frac{2\ln n}{n_j}}$, where $\bar{x}_j$ is the average reward obtained from machine j, n_j is the number of times machine j has been played so far, and n is the overall number of plays done so far.

NPTEL Introduction to Data Analytics 22

So, you can show that the sub optimal arms are played fewer than $\frac{8}{\Delta_j}$ times, where Δ_j is the difference between the optimal arm and the sub optimal arm and this allows us to come up with very efficient bounds for this upper confidence bounds algorithm UCB, so the algorithm is very simple. So, at any time you played the arm j that maximizes $\bar{x}_j$, which is your average reward. So, far + this funny expression under the $\sqrt{\frac{2\ln n}{n_j}}$, where n is the number of arms and n_j is the number of times you have played arm j, so far.

So, that expression actually comes from estimating upper confidence bounds and this help us show that the sub optimal on can be played I will be played fewer than $\frac{8}{\Delta_j}$ ln(n) times. So, one of the implication of all of this I am giving you lot of numbers, but in practice you do not really have to worry about this numbers, because all we are interested in this figuring out with there is a fewer number of samples can I estimate what the outcome will be. And UCB essentially gets you there very quickly even though does not guarantee that it gives you the best outcome, but it reaches close to the best outcome pretty quickly.

(Refer Slide Time: 13:34)



Contextual Bandits

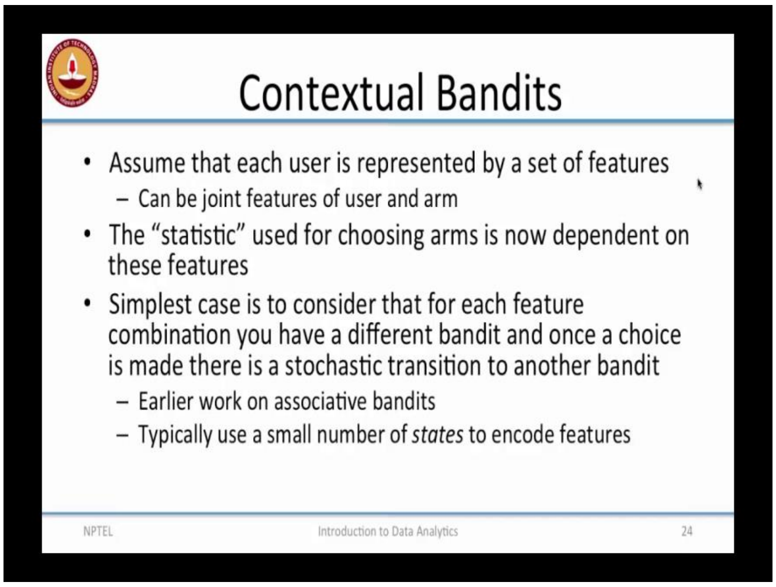
- Different ads for different users
  - One bandit for each user!
- Hard to train
  - Need several rounds of experience with same user
  - Typically users share features
    - Demographic
    - Browsing history
    - Location, etc.
  - "Arms" share features too!

NPTEL Introduction to Data Analytics 23

So, I was talking about showing ads as one of the use cases, but if you think about it for each user you would like to show different ads you do not want to show the same add to different users. But, if you are using a single bandit algorithm that is learning the preferences of all the users that is what you end up doing. So, one solution around this is to build one bandit for each user, but given the large volumes of users there are likely to come to your page and restrictions on computation and search that is not feasible.

So, what we do is you typically share features across different users it could be demographic features, it could be browsing history, it could be location or it if you talking about patient data that it could be history, the patient history that could be variety of different features that you would share and your expected reward for getting the for pulling, then arm. Now be represented as a function suppose to being a number is the function of these features as suppose to being a single estimated number.

(Refer Slide Time: 14:54)



Contextual Bandits

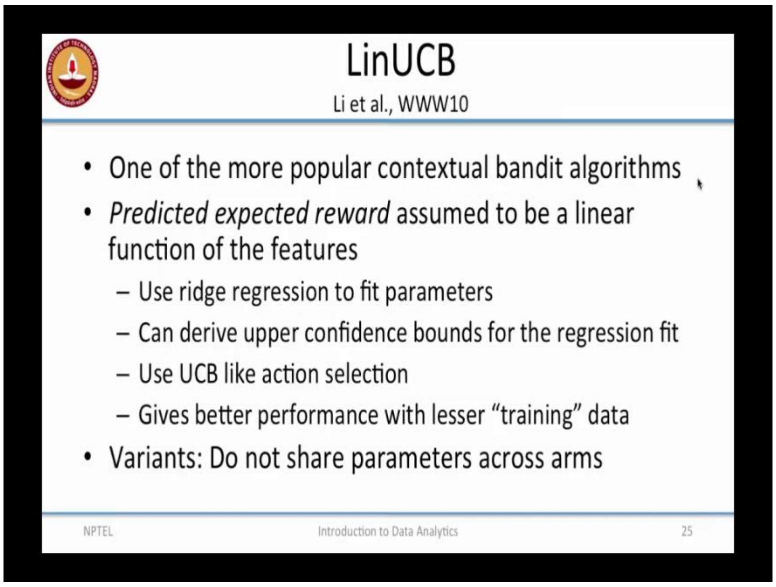
- Assume that each user is represented by a set of features
  - Can be joint features of user and arm
- The "statistic" used for choosing arms is now dependent on these features
- Simplest case is to consider that for each feature combination you have a different bandit and once a choice is made there is a stochastic transition to another bandit
  - Earlier work on associative bandits
  - Typically use a small number of *states* to encode features

NPTEL Introduction to Data Analytics 24

So, you can solve this in many different ways you can assume that each user is represented by a set of features that could be set of joint features of the user and of the arm the statistics used for choosing arms is now, dependent on these features this essentially what I was telling the statistic in some cases can just be the mean reward could be the upper confidence bounds to the verities of things. So, the simplest case is to consider that for each feature combination you have a different bandit once a choice is made there is a stochastic transition to another bandit this just to make it conceptually easier.

But, that is not really the case, because each of this bandits are going to be closely interacting, because the feature values might be similar in which, case the outcomes for one bandits could be same for other bandits. So, earlier work there was on associative bandits where typically use a small number of states to encode this features and you do not really worried about functional representations for the values. So, once I have a function that is going to output a real value as a function of features, what would you think of doing, exactly.

(Refer Slide Time: 16:14)



LinUCB

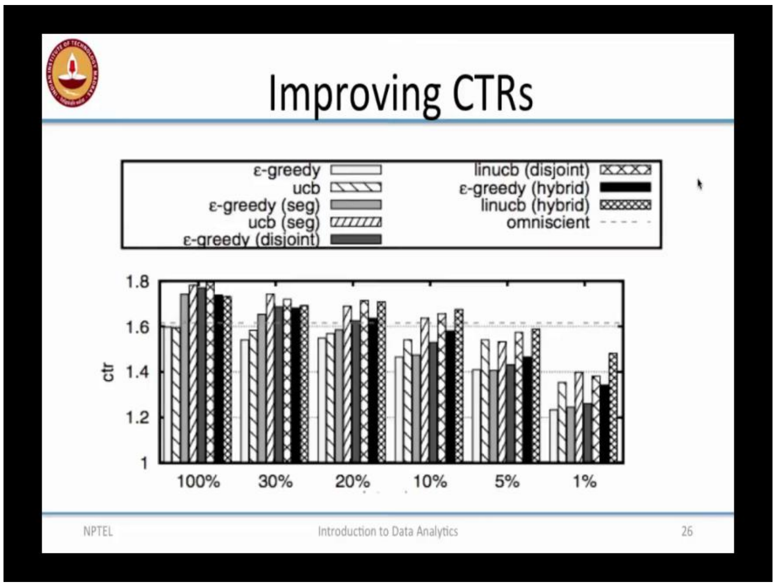
Li et al., WWW10

- One of the more popular contextual bandit algorithms
- *Predicted expected reward* assumed to be a linear function of the features
  - Use ridge regression to fit parameters
  - Can derive upper confidence bounds for the regression fit
  - Use UCB like action selection
  - Gives better performance with lesser "training" data
- Variants: Do not share parameters across arms

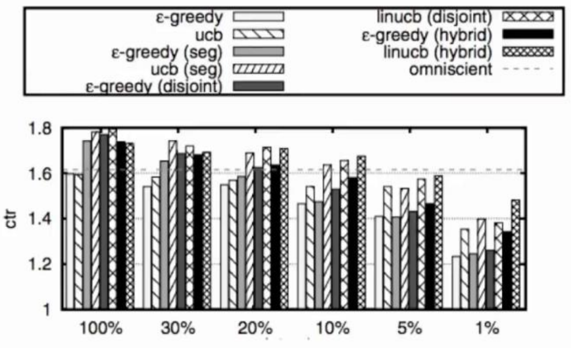
NPTEL Introduction to Data Analytics 25

So, we could do linear regression, so where UCB variation, where you use linear regression to predict the mean reward and you use a upper confidence bounds to from the linear regression algorithm is called Lin UCB and one of the more popular contextual bandit algorithms out there. So, they expected reward is assume to be a linear function of the features people typically end of using ridge regression, which is L2 initialized linear regression to fit the parameters and you can derive upper confidence bounds for the regression fit. So, it gives better performance with lesser training data.

(Refer Slide Time: 17:00)



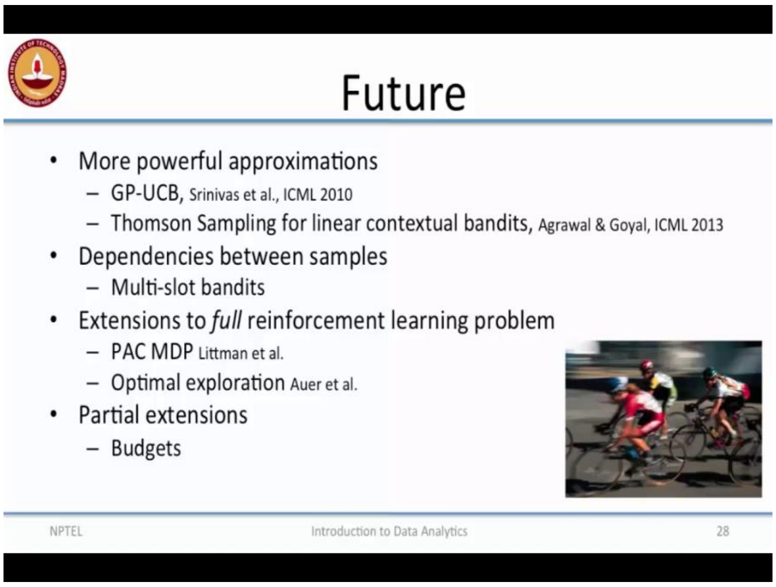
Improving CTRs



NPTEL Introduction to Data Analytics 26

So, is here is a simple set of results for predicting click through rates on ads and you can see that if only one percent of the add data is available for training available for estimating the parameters we could see that Lin UCB methods, which are this and this do much better, than regular UCB method, which don't do this generalization across different users. So, you can see that 5 percent data again, so the Lin UCB variance are doing better, but then if you have a lot of data available for training we could see that the other methods pretty much catch up with Lin UCB. So, the idea here is that if Lin UCB you have better performance with lesser training data.

(Refer Slide Time: 17:57)



Future

- More powerful approximations
  - GP-UCB, Srinivas et al., ICML 2010
  - Thomson Sampling for linear contextual bandits, Agrawal & Goyal, ICML 2013
- Dependencies between samples
  - Multi-slot bandits
- Extensions to *full* reinforcement learning problem
  - PAC MDP Littman et al.
  - Optimal exploration Auer et al.
- Partial extensions
  - Budgets

NPTEL Introduction to Data Analytics 28

And, so people now, are looking at more powerful approximation looking at Gaussian Process looking at other more complex sampling techniques lot of work going on and it's very active variant. So, people are looking at budgeted bandits people are looking at case where there are adversarial bandits and then, people are looking at dependencies between samples and multi slot cases and there are many different problems that people are sharing in the bandits literature.

One thing of interest is to consider, what I call a full reinforcement learning problem in the contextual bandits case. So, you have the context the user comes and picks an action and after that, which is move on to other user. But, in the full reinforcement learning problem, so you have to make a sequence of decisions before you can truly solve the problems. So, the faced with a situation or states or a set of features you take an action then, that results in the set of features being changed other you have to take another action and another action another action until the problem is solved.

So, think of the question of riding a cycle, all of you know how to ride cycle, then think of how you will learned to ride a cycle you dint learned through supervised learning there is no body telling you how to turn the how, what a pressure to put on a peddle and what angle you should tilt to your upper body and things like that if I will telling you the technical aspects of cycling you probably won't learn to cycle. All in get on the cycle and make sure we do not fall of or if you fall of in get hard you do not do the thing in next end and soon you are if find yourself cycling.

So, this is the essentially a form of reinforcement learning when each action you take and if you make a bad move you are just going to keep tilting take until you fall. So, each action you take causes the change in the state and you have to compensate for that and, then continue learning. So, this is called the full reinforcement learning problem and the idea that we describe here like PAC, MDP and optimal exploration, so and so on, so forth, have been extended to the reinforcement learning problem as well.

So, here are couple of that are few tutorials and source material for learning more about bandits if you are interested, so this is the end of the module.