# Bandits: Thompson Sampling and Contextual Bandits

#### Lucas Janson CS/Stat 184(0): Introduction to Reinforcement Learning Fall 2024

- Feedback from last lecture
- Recap
- Thompson sampling
- Contextual bandits



#### Feedback from feedback forms

#### Feedback from feedback forms

1. Thank you to everyone who filled out the forms!

#### Feedback from feedback forms

- 1. Thank you to everyone who filled out the forms!
- 2. Green can be hard to see





- Recap
- Thompson sampling
- Contextual bandits

A Bayesian bandit augments the bandit environment we've been working in so far with a prior distribution on the unknown reward distributions:  $\pi(\nu^{(1)}, \dots, \nu^{(K)})$ 

#### Bayesian bandit



A Bayesian bandit augments the bandit environment we've been working in so far with a prior distribution on the unknown reward distributions:  $\pi(\nu^{(1)}, \dots, \nu^{(K)})$ 

E.g., in a Bernoulli bandit, each  $\nu^{(k)}$  is entirely characterized by its mean  $\mu^{(k)} = \mathbb{P}_{r \sim \nu^{(k)}}$  (r = 1), so a prior on the  $\nu^{(k)}$  is equivalent to a prior on the  $\mu^{(k)}$ 



E.g., in a Bernoulli bandit, each  $\nu^{(k)}$  is entirely characterized by its mean  $\mu^{(k)} = \mathbb{P}_{r \sim \nu^{(k)}}$  (r = 1), so a prior on the  $\nu^{(k)}$  is equivalent to a prior on the  $\mu^{(k)}$ 

One such prior, since all the  $\mu^{(k)}$  are bounded between 0 and 1, is the prior that is *Uniform* on the unit hypercube, i.e.,  $(\mu^{(1)}, \dots, \mu^{(K)}) =: \mu \sim \text{Uniform}([0, 1]^K)$ 

A Bayesian bandit augments the bandit environment we've been working in so far with a prior distribution on the unknown reward distributions:  $\pi(\nu^{(1)}, \dots, \nu^{(K)})$ 



E.g., in a Bernoulli bandit, each  $\nu^{(k)}$  is entirely characterized by its mean  $\mu^{(k)} = \mathbb{P}_{r \sim \nu^{(k)}}$  (r = 1), so a prior on the  $\nu^{(k)}$  is equivalent to a prior on the  $\mu^{(k)}$ 

A Bayesian bandit augments the bandit environment we've been working in so far with a prior distribution on the unknown reward distributions:  $\pi(\nu^{(1)}, \dots, \nu^{(K)})$ 

- One such prior, since all the  $\mu^{(k)}$  are bounded between 0 and 1, is the prior that is *Uniform* on the unit hypercube, i.e.,
  - $(\mu^{(1)}, \dots, \mu^{(K)}) =: \mu \sim \text{Uniform}([0, 1]^K)$
- Note that the Bernoulli bandit reduced everything unknown about the bandit system to a K-dimensional vector  $\mu$



- A Bayesian bandit augments the bandit environment we've been working in so far with a prior distribution on the unknown reward distributions:  $\pi(\nu^{(1)}, \dots, \nu^{(K)})$ 
  - E.g., in a Bernoulli bandit, each  $\nu^{(k)}$  is entirely characterized by its mean  $\mu^{(k)} = \mathbb{P}_{r \sim \nu^{(k)}}$  (r = 1), so a prior on the  $\nu^{(k)}$  is equivalent to a prior on the  $\mu^{(k)}$ 
    - One such prior, since all the  $\mu^{(k)}$  are bounded between 0 and 1, is the prior that is *Uniform* on the unit hypercube, i.e.,
      - $(\mu^{(1)}, \dots, \mu^{(K)}) =: \mu \sim \text{Uniform}([0, 1]^K)$
- Note that the Bernoulli bandit reduced everything unknown about the bandit system to a K-dimensional vector  $\mu$
- Without the Bernoulli assumption, we may need many more dimensions to describe the possible distributions, and hence have to define a much higher-dimensional prior



A Bayesian bandit augments the bandit environment we've been working in so far with a prior distribution on the unknown reward distributions; for Bernoulli bandits,

the reward distributions are entirely characterized by  $\mu$ , so prior is:  $\pi(\mu)$ 



- A Bayesian bandit augments the bandit environment we've been working in so far with a prior distribution on the unknown reward distributions; for Bernoulli bandits, the reward distributions are entirely characterized by  $\mu$ , so prior is:  $\pi(\mu)$
- Bayes rule at time step t gives us a distribution (called the posterior distribution)  $\mathbb{P}(\boldsymbol{\mu} \mid r_0, a_0, r_1)$
- that exactly characterizes our uncertainty about  $\mu$

$$, a_1, \ldots, r_{t-1}, a_{t-1})$$





- A Bayesian bandit augments the bandit environment we've been working in so far with a prior distribution on the unknown reward distributions; for Bernoulli bandits, the reward distributions are entirely characterized by  $\mu$ , so prior is:  $\pi(\mu)$
- Bayes rule at time step t gives us a distribution (called the posterior distribution)  $\mathbb{P}(\boldsymbol{\mu} \mid r_0, a_0, r_1)$
- that exactly characterizes our uncertainty about  $\mu$
- Bernoulli bandit's posterior is Beta-distributed with simple parameter updates

$$, a_1, \ldots, r_{t-1}, a_{t-1})$$





- A Bayesian bandit augments the bandit environment we've been working in so far with a prior distribution on the unknown reward distributions; for Bernoulli bandits, the reward distributions are entirely characterized by  $\mu$ , so prior is:  $\pi(\mu)$
- Bayes rule at time step t gives us a distribution (called the posterior distribution)  $\mathbb{P}(\boldsymbol{\mu} \mid r_0, a_0, r_1)$
- that exactly characterizes our uncertainty about  $\mu$
- Bernoulli bandit's posterior is Beta-distributed with simple parameter updates
- Note that although we are now treating  $\mu$  as random, we still assume its value is only drawn once (from the prior) and then stays the same throughout t

$$, a_1, \ldots, r_{t-1}, a_{t-1})$$







- A Bayesian bandit augments the bandit environment we've been working in so far with a prior distribution on the unknown reward distributions; for Bernoulli bandits, the reward distributions are entirely characterized by  $\mu$ , so prior is:  $\pi(\mu)$
- Bayes rule at time step t gives us a distribution (called the posterior distribution)  $\mathbb{P}(\boldsymbol{\mu} \mid r_0, a_0, r_1, a_1, \dots, r_{t-1}, a_{t-1})$
- that exactly characterizes our uncertainty about  $\mu$
- Bernoulli bandit's posterior is Beta-distributed with simple parameter updates
- Note that although we are now treating  $\mu$  as random, we still assume its value is only drawn once (from the prior) and then stays the same throughout t
- What changes with t is our information about  $\mu$ , i.e., the posterior distribution, as we collect more and more data by pulling arms via a bandit algorithm











- Recap
  - Thompson sampling
  - Contextual bandits



Bayesian bandit environment means that at every time step, we know the distribution of the arm reward distributions conditioned on everything we've seen so far



- Bayesian bandit environment means that at every time step, we know the distribution of the arm reward distributions conditioned on everything we've seen so far
  - In particular, we know the exact probability, given everything we've seen so far, that each arm is the true optimal arm, i.e.,

 $\forall k, \text{ we know } \mathbb{P}(k = k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1})$ 



- Bayesian bandit environment means that at every time step, we know the distribution of the arm reward distributions conditioned on everything we've seen so far
  - In particular, we know the exact probability, given everything we've seen so far, that each arm is the true optimal arm, i.e.,
    - $\forall k, \text{ we know } \mathbb{P}(k = k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1})$

hompson sampling: sample from this distribution to determine next arm to pull



- Bayesian bandit environment means that at every time step, we know the distribution of the arm reward distributions conditioned on everything we've seen so far
  - In particular, we know the exact probability, given everything we've seen so far, that each arm is the true optimal arm, i.e.,

<u>Fhompson sampling</u>: sample from this distribution to determine next arm to pull

For t = 0, ..., T - 1:  $a_{t} \sim \text{distribution of } k^{\star} \mid r_{0}, a_{0}, \dots, r_{t-1}, a_{t-1}$ 

 $\forall k, \text{ we know } \mathbb{P}(k = k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1})$ 



- Bayesian bandit environment means that at every time step, we know the distribution of the arm reward distributions conditioned on everything we've seen so far
  - In particular, we know the exact probability, given everything we've seen so far, that each arm is the true optimal arm, i.e.,

<u>Thompson sampling</u>: sample from this distribution to determine next arm to pull

For t = 0, ..., T - 1:  $a_{t} \sim \text{distribution of } k^{\star} \mid r_{0}, a_{0}, \dots, r_{t-1}, a_{t-1}$ 

How can we sample from this distribution?

 $\forall k, \text{ we know } \mathbb{P}(k = k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1})$ 



- Bayesian bandit environment means that at every time step, we know the distribution of the arm reward distributions conditioned on everything we've seen so far
  - In particular, we know the exact probability, given everything we've seen so far, that each arm is the true optimal arm, i.e.,

 $\forall k$ , we know  $\mathbb{P}(k =$ 

Thompson sampling: sample from this distribution to determine next arm to pull

For t = 0, ..., T - 1:  $a_{t} \sim \text{distribution of } k^{\star} \mid r_{0}, a_{0}, \dots, r_{t-1}, a_{t-1}$ 

How can we sample from this distribution? Draw a sample  $\mu_t \sim distribution of \mu \mid r_0, a_0, \dots, r_{t-1}, a_{t-1}$  and then compute  $a_t = \arg \max \mu_t^{(k)}$ , which is the same thing as  $a_t \sim \text{distribution of } k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1}$ 

$$k^{\star} | r_0, a_0, \dots, r_{t-1}, a_{t-1})$$





- Bayesian bandit environment means that at every time step, we know the distribution of the arm reward distributions conditioned on everything we've seen so far
  - In particular, we know the exact probability, given everything we've seen so far, that each arm is the true optimal arm, i.e.,

 $\forall k$ , we know  $\mathbb{P}(k =$ 

Thompson sampling: sample from this distribution to determine next arm to pull

For t = 0, ..., T - 1:  $a_{t} \sim \text{distribution of } k^{\star} \mid r_{0}, a_{0}, \dots, r_{t-1}, a_{t-1}$ 

How can we sample from this distribution? Draw a sample  $\mu_t \sim distribution of \mu \mid r_0, a_0, \dots, r_{t-1}, a_{t-1}$  and then compute  $a_t = \arg \max \mu_t^{(k)}$ , which is the same thing as  $a_t \sim \text{distribution of } k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1}$ That's it! Statistically, this is a super simple and elegant algorithm

$$k^{\star} | r_0, a_0, \dots, r_{t-1}, a_{t-1})$$





- Bayesian bandit environment means that at every time step, we know the distribution of the arm reward distributions conditioned on everything we've seen so far
  - In particular, we know the exact probability, given everything we've seen so far, that each arm is the true optimal arm, i.e.,

 $\forall k$ , we know  $\mathbb{P}(k =$ 

Thompson sampling: sample from this distribution to determine next arm to pull

For t = 0, ..., T - 1:  $a_{t} \sim \text{distribution of } k^{\star} \mid r_{0}, a_{0}, \dots, r_{t-1}, a_{t-1}$ 

How can we sample from this distribution? Draw a sample  $\mu_t \sim distribution of \mu \mid r_0, a_0, \dots, r_{t-1}, a_{t-1}$  and then compute  $a_t = \arg \max \mu_t^{(k)}$ , which is the same thing as  $a_t \sim \text{distribution of } k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1}$ That's it! Statistically, this is a super simple and elegant algorithm (though computationally, it may not be easy to update the posterior at each time step)

$$k^{\star} | r_0, a_0, \dots, r_{t-1}, a_{t-1})$$





<u>Thompson sampling</u>:  $a_t \sim \text{distribution of } k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1}$ 

<u>Thompson sampling</u>:  $a_t \sim dis$ Why is thi

- <u>Thompson sampling</u>:  $a_t \sim \text{distribution of } k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1}$ 
  - Why is this a good idea?

- <u>Thompson sampling</u>:  $a_t \sim \text{distribution of } k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1}$ 
  - Why is this a good idea?
- A good tradeoff of exploration vs exploitation should:

- <u>Thompson sampling</u>:  $a_t \sim \text{distribution of } k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1}$ 
  - Why is this a good idea?
- A good tradeoff of exploration vs exploitation should: a) Sample the optimal arm as much as possible (duh)

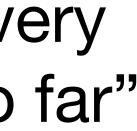
- Thompson sampling:  $a_t \sim \text{distribution of } k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1}$ 
  - Why is this a good idea?
- A good tradeoff of exploration vs exploitation should:a) Sample the optimal arm as much as possible (duh)b) Ensure arms that might still be optimal aren't overlooked

- Thompson sampling:  $a_t \sim \text{distribution of } k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1}$ 
  - Why is this a good idea?
- A good tradeoff of exploration vs exploitation should:
  a) Sample the optimal arm as much as possible (duh)
  b) Ensure arms that might still be optimal aren't overlooked
  c) Not waste undue time on less promising arms

<u>Thompson sampling</u>:  $a_t \sim \text{distribution of } k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1}$ Why is this a good idea?

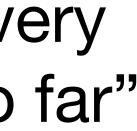
<u>Thompson sampling</u>:  $a_t \sim \text{distribution of } k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1}$ Why is this a good idea?

- This is exactly what Thompson sampling does, where "promising" is encoded very naturally as: "the probability that the arm is the optimal arm, given all the data so far"



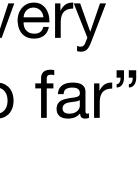
<u>Thompson sampling</u>:  $a_t \sim \text{distribution of } k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1}$ Why is this a good idea?

- This is exactly what Thompson sampling does, where "promising" is encoded very naturally as: "the probability that the arm is the optimal arm, given all the data so far"
  - No arbitrary  $\delta$  tuning parameter, but do have to choose prior  $\pi$



<u>Thompson sampling</u>:  $a_t \sim \text{distribution of } k^* \mid r_0, a_0, \dots, r_{t-1}, a_{t-1}$ Why is this a good idea?

- This is exactly what Thompson sampling does, where "promising" is encoded very naturally as: "the probability that the arm is the optimal arm, given all the data so far"
  - No arbitrary  $\delta$  tuning parameter, but do have to choose prior  $\pi$  $\pi$  can often be chosen "uninformatively" to a default prior such as the uniform, or can encode nuanced prior information/belief about the arms' reward distributions





Thompson sampling samples arms proportionally to how promising they are

9

- Thompson sampling samples arms proportionally to how promising they are
- Note this sampling is much more sophisticated than, say,  $\varepsilon$ -greedy, which really just samples according to 2 categories: "most promising" and "other"



- Thompson sampling samples arms proportionally to how promising they are
- Note this sampling is much more sophisticated than, say,  $\varepsilon$ -greedy, which really just samples according to 2 categories: "most promising" and "other"
- But it's also quite different from UCB, whose OFU approach doesn't really involve
- "sampling" at all, i.e., every  $a_t$  for UCB is a *deterministic* function of the previous data



- Thompson sampling samples arms proportionally to how promising they are
- Note this sampling is much more sophisticated than, say,  $\varepsilon$ -greedy, which really just samples according to 2 categories: "most promising" and "other"
- But it's also quite different from UCB, whose OFU approach doesn't really involve "sampling" at all, i.e., every  $a_t$  for UCB is a *deterministic* function of the previous data
- My interpretation: OFU provides a simple heuristic to accomplish what Thompson sampling does by design, namely, sample arms according to how promising they are



- Thompson sampling samples arms proportionally to how promising they are
- Note this sampling is much more sophisticated than, say,  $\varepsilon$ -greedy, which really just samples according to 2 categories: "most promising" and "other"
- But it's also quite different from UCB, whose OFU approach doesn't really involve "sampling" at all, i.e., every  $a_t$  for UCB is a *deterministic* function of the previous data
- My interpretation: OFU provides a simple heuristic to accomplish what Thompson sampling does by design, namely, sample arms according to how promising they are
- Thompson sampling can do this because of the Bayesian bandit: assuming a prior on the reward distributions makes the arm means random, otherwise it wouldn't even make sense to talk about "the probability that an arm is the best arm"



- Thompson sampling samples arms proportionally to how promising they are
- Note this sampling is much more sophisticated than, say,  $\varepsilon$ -greedy, which really just samples according to 2 categories: "most promising" and "other"
- But it's also quite different from UCB, whose OFU approach doesn't really involve "sampling" at all, i.e., every  $a_t$  for UCB is a *deterministic* function of the previous data
- My interpretation: OFU provides a simple heuristic to accomplish what Thompson sampling does by design, namely, sample arms according to how promising they are
- Thompson sampling can do this because of the Bayesian bandit: assuming a prior on the reward distributions makes the arm means random, otherwise it wouldn't even make sense to talk about "the probability that an arm is the best arm"
  - Although derived from the Bayesian bandit, Thompson sampling has excellent practical performance across bandit problems, whether or not they are Bayesian!



Thompson sampling has excellent performance in practice, but is still just a heuristic

Thompson sampling has excellent performance in practice, but is still just a heuristic



Thompson sampling has excellent performance in practice, but is still just a heuristic

There is an *instance-dependent* lower-bound result that says that for <u>any</u> bandit algorithm:  $\liminf_{T \to \infty} \frac{\mathbb{E}[N_T^{(k)}]}{\ln(T)} \ge \frac{1}{d(\nu^{(k^*)}, \nu^{(k)})},$ 

where d is a distance between distributions called the Kullback – Leibler divergence





Thompson sampling has excellent performance in practice, but is still just a heuristic

There is an *instance-dependent* lower-bound result that says that for <u>any</u> bandit algorithm:  $\liminf_{T \to \infty} \frac{\mathbb{E}[N_T^{(k)}]}{\ln(T)} \ge \frac{1}{d(\nu^{(k^*)}, \nu^{(k)})},$ 

where d is a distance between distributions called the Kullback – Leibler divergence

It turns out that Thompson sampling satisfies this lower-bound with equality!





Thompson sampling has excellent performance in practice, but is still just a heuristic

There is an *instance-dependent* lower-bound result that says that for <u>any</u> bandit algorithm:  $\liminf_{T \to \infty} \frac{\mathbb{E}[N_T^{(k)}]}{\ln(T)} \ge \frac{1}{d(\nu^{(k^*)}, \nu^{(k)})},$ 

where d is a distance between distributions called the Kullback – Leibler divergence

It turns out that Thompson sampling satisfies this lower-bound with equality!

So it is asymptotically optimal, not just in its rate, but its constant too!





- Thompson sampling has excellent performance in practice, but is still just a heuristic
  - However, asymptotically, i.e., as  $T \to \infty$ , it actually is optimal in a certain sense
  - There is an *instance-dependent* lower-bound result that says that for <u>any</u> bandit algorithm:
    - $\liminf_{T \to \infty} \frac{\mathbb{E}[N_T^{(k)}]}{\ln(T)} \ge \frac{1}{d(\nu^{(k^*)}, \nu^{(k)})},$
- where d is a distance between distributions called the Kullback Leibler divergence
  - It turns out that Thompson sampling satisfies this lower-bound with equality!
    - So it is asymptotically optimal, not just in its rate, but its constant too!
      - (UCB is not, but there are more complicated versions of it that are)





What could go wrong for smaller T? Suppose K = 2 and T = 3, and:

What could go wrong for smaller T? Suppose K = 2 and T = 3, and:

• t = 0:  $a_0 = 1$ ,  $r_0 = 1$ 

What could go wrong for smaller T? Suppose K = 2 and T = 3, and:

• 
$$t = 0$$
:  $a_0 = 1$ ,  $r_0 = 1$ 

• 
$$t = 1: a_1 = 2, r_1 = 0$$

What could go wrong for smaller T? Suppose K = 2 and T = 3, and:

• 
$$t = 0$$
:  $a_0 = 1$ ,  $r_0 = 1$ 

• 
$$t = 1$$
:  $a_1 = 2$ ,  $r_1 = 0$ 

• t = 2 (last time step, with  $\hat{\mu}_{\gamma}^{(1)} = 1$  and  $\hat{\mu}_{\gamma}^{(2)} = 0$ ):  $a_2 = ?$ 

What could go wrong for smaller T? Suppose K = 2 and T = 3, and:

- t = 0:  $a_0 = 1$ ,  $r_0 = 1$
- $t = 1: a_1 = 2, r_1 = 0$
- t = 2 (last time step, with  $\hat{\mu}_2^{(1)} = 1$  and

Thompson sampling has a decent probability of choosing  $a_2 = 2$ , since with just one sample from each arm, Thompson sampling isn't sure which arm is best.

Thompson sampling in practice (cont'd) So Thompson sampling is basically exactly optimal for large T

nd 
$$\hat{\mu}_2^{(2)} = 0$$
):  $a_2 = ?$ 

What could go wrong for smaller T? Suppose K = 2 and T = 3, and:

- t = 0:  $a_0 = 1$ ,  $r_0 = 1$
- t = 1:  $a_1 = 2$ ,  $r_1 = 0$
- t = 2 (last time step, with  $\hat{\mu}_2^{(1)} = 1$  and

one sample from each arm, Thompson sampling isn't sure which arm is best.

no reason to explore rather than exploit.

Thompson sampling in practice (cont'd) So Thompson sampling is basically exactly optimal for large T

nd 
$$\hat{\mu}_2^{(2)} = 0$$
):  $a_2 = ?$ 

Thompson sampling has a decent probability of choosing  $a_2 = 2$ , since with just

But  $a_2 = 1$  is clear right choice here: there is no future value to learning more, i.e.,



What could go wrong for smaller T? Suppose K = 2 and T = 3, and:

- t = 0:  $a_0 = 1$ ,  $r_0 = 1$
- $t = 1: a_1 = 2, r_1 = 0$
- t = 2 (last time step, with  $\hat{\mu}_2^{(1)} = 1$  and

one sample from each arm, Thompson sampling isn't sure which arm is best.

no reason to explore rather than exploit.

Thompson sampling doesn't know this, and neither does UCB (although UCB) wouldn't happen to make the same mistake in this case).

Thompson sampling in practice (cont'd) So Thompson sampling is basically exactly optimal for large T

nd 
$$\hat{\mu}_2^{(2)} = 0$$
):  $a_2 = ?$ 

Thompson sampling has a decent probability of choosing  $a_2 = 2$ , since with just

But  $a_2 = 1$  is clear right choice here: there is no future value to learning more, i.e.,



For small T, Thompson sampling is not greedy enough

For small T, Thompson sampling is not greedy enough

Fix: add a tuning parameter to make it more greedy. Some possibilities:

For small T, Thompson sampling is not greedy enough

Fix: add a tuning parameter to make it more greedy. Some possibilities:

- Update the Beta parameters by  $1 + \epsilon$  instead of just 1 each time

Fix: add a tuning parameter to make it more greedy. Some possibilities:

- Update the Beta parameters by  $1 + \epsilon$  instead of just 1 each time
- Instead of just taking one sample of  $\mu$  and computing the greedy action with respect to it, take *n* samples, compute the greedy action with respect to each, and pick the mode of those greedy actions

- For small T, Thompson sampling is not greedy enough



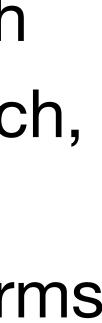
Fix: add a tuning parameter to make it more greedy. Some possibilities:

- Update the Beta parameters by  $1 + \epsilon$  instead of just 1 each time
- Instead of just taking one sample of  $\mu$  and computing the greedy action with respect to it, take *n* samples, compute the greedy action with respect to each, and pick the *mode* of those greedy actions

that have worked well so far), as opposed to arms that may be good

- For small T, Thompson sampling is not greedy enough

All of these favor arms that the algorithm has more confidence are good (i.e., arms



Fix: add a tuning parameter to make it more greedy. Some possibilities:

- Update the Beta parameters by  $1 + \epsilon$  instead of just 1 each time
- Instead of just taking one sample of  $\mu$  and computing the greedy action with respect to it, take *n* samples, compute the greedy action with respect to each, and pick the *mode* of those greedy actions

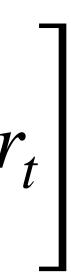
that have worked well so far), as opposed to arms that may be good

- For small T, Thompson sampling is not greedy enough

- All of these favor arms that the algorithm has more confidence are good (i.e., arms
- Such tuning can improve Thompson sampling's performance even for reasonably large T (the asymptotic optimality of vanilla TS is very asymptotic)

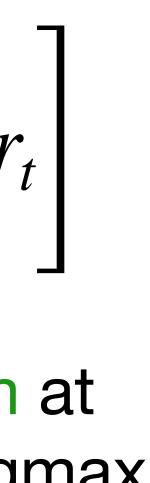


For infinite time horizon with discount factor  $\gamma$ , can *exactly* optimize  $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^{t} r_{t}\right]$ 



For infinite time horizon with discount factor  $\gamma$ , can *exactly* optimize  $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$ 

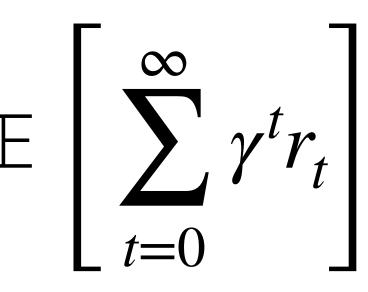
<u>Gittins index</u> is both the name of a quantity that can be computed for each arm at any given time, and often also used to refer to the algorithm that chooses the argmax of the Gittins index at each time point



For infinite time horizon with discount factor  $\gamma$ , can *exactly* optimize  $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^{t} r_{t}\right]$ 

<u>Gittins index</u> is both the name of a quantity that can be computed for each arm at any given time, and often also used to refer to the algorithm that chooses the argmax of the Gittins index at each time point

Algorithm/proof beyond scope of this class, but rely on an efficient approximation to the Gittins index via dynamic programming that updates when an arm is pulled







For infinite time horizon with discount factor  $\gamma$ , can *exactly* optimize  $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$ 

<u>Gittins index</u> is both the name of a quantity that can be computed for each arm at any given time, and often also used to refer to the algorithm that chooses the argmax of the Gittins index at each time point

Algorithm/proof beyond scope of this class, but rely on an efficient approximation to the Gittins index via dynamic programming that updates when an arm is pulled

Performance is great, even if prior is wrong, but algorithmic principle behind it brittle: doesn't easily extend to exact optimality in even slightly more complex settings











- Recap
- Thompson sampling



Contextual bandits



### Beyond simple bandits

### Beyond simple bandits

In a bandit, we are presented with the same decision at every time

### Beyond simple bandits

# In practice, often decisions are not the same every time

In a bandit, we are presented with the same decision at every time

E.g., in online advertising there may not be a single best ad to show all users on all websites:

In a bandit, we are presented with the same decision at every time In practice, often decisions are not the same every time

E.g., in online advertising there may not be a single best ad to show all users on all websites: maybe some types of users prefer one ad while others prefer another, or

In a bandit, we are presented with the same decision at every time In practice, often decisions are not the same every time

on all websites:

- maybe some types of users prefer one ad while others prefer another, or maybe one type of ad works better on certain websites while another
- works better on other websites

In a bandit, we are presented with the same decision at every time In practice, often decisions are not the same every time

E.g., in online advertising there may not be a single best ad to show all users

on all websites:

- maybe some types of users prefer one ad while others prefer another, or maybe one type of ad works better on certain websites while another
- works better on other websites

Which user comes in next is random, but we have some context to tell situations apart and hence learn different optimal actions

In a bandit, we are presented with the same decision at every time In practice, often decisions are not the same every time

E.g., in online advertising there may not be a single best ad to show all users

Context at time t encoded into a variable  $x_t$  that we see before choosing our action



Context at time t encoded into a variable  $x_t$  that we see before choosing our action

 $x_t$  is drawn i.i.d. at each time point from a distribution  $\nu_x$  on sample space  $\mathcal{X}$ 

- Context at time t encoded into a variable  $x_t$  that we see before choosing our action  $x_t$  is drawn i.i.d. at each time point from a distribution  $\nu_x$  on sample space  $\mathcal{X}$
- $x_t$  then affects the reward distributions of each arm, i.e., if we choose arm k, we get a reward that is drawn from a distribution that depends on  $x_t$ , namely,  $\nu^{(k)}(x_t)$



- Context at time t encoded into a variable  $x_t$  that we see before choosing our action  $x_t$  is drawn i.i.d. at each time point from a distribution  $\nu_x$  on sample space  $\mathcal{X}$
- $x_t$  then affects the reward distributions of each arm, i.e., if we choose arm k, we get a reward that is drawn from a distribution that depends on  $x_t$ , namely,  $\nu^{(k)}(x_t)$ 
  - Accordingly, we should also choose our action  $a_t$  in a way that depends on  $x_t$ , i.e., our action should be chosen by a function of  $x_t$  (a policy), namely,  $\pi_t(x_t)$





- Context at time t encoded into a variable  $x_t$  that we see before choosing our action  $x_t$  is drawn i.i.d. at each time point from a distribution  $\nu_x$  on sample space  $\mathscr{X}$
- $x_t$  then affects the reward distributions of each arm, i.e., if we choose arm k, we get a reward that is drawn from a distribution that depends on  $x_t$ , namely,  $\nu^{(k)}(x_t)$ 
  - Accordingly, we should also choose our action  $a_t$  in a way that depends on  $x_t$ , i.e., our action should be chosen by a function of  $x_t$  (a policy), namely,  $\pi_t(x_t)$
  - If we knew everything about the environment, we'd want to use the optimal policy  $\pi^{\star}(x_t) := \arg \max_{k \in \{1, \dots, K\}} \mu^{(k)}(x_t),$ where  $\mu^{(k)}(x) := \mathbb{E}_{r \sim \nu^{(k)}(x)}[r]$







- Context at time t encoded into a variable  $x_t$  that we see before choosing our action  $x_t$  is drawn i.i.d. at each time point from a distribution  $\nu_x$  on sample space  $\mathscr{X}$
- $x_t$  then affects the reward distributions of each arm, i.e., if we choose arm k, we get a reward that is drawn from a distribution that depends on  $x_t$ , namely,  $\nu^{(k)}(x_t)$ 
  - Accordingly, we should also choose our action  $a_t$  in a way that depends on  $x_t$ , i.e., our action should be chosen by a function of  $x_t$  (a policy), namely,  $\pi_t(x_t)$
  - If we knew everything about the environment, we'd want to use the optimal policy  $\pi^{\star}(x_t) := \arg \max_{k \in \{1, \dots, K\}} \mu^{(k)}(x_t),$ where  $\mu^{(k)}(x) := \mathbb{E}_{r \sim \nu^{(k)}(x)}[r]$

 $\pi^{\star}$  is the policy we compare to in computing regret







Formally, a contextual bandit is the following interactive learning process:

Formally, a contextual bandit is the following interactive learning process:

For  $t = 0 \rightarrow T - 1$ 

Formally, a contextual bandit is the following interactive learning process:

For 
$$t = 0 \rightarrow T - 1$$

1. Learner sees context  $x_t \sim \nu_x$ 

Formally, a contextual bandit is the following interactive learning process:

For 
$$t = 0 \rightarrow T - 1$$

1. Learner sees context  $x_t \sim \nu_{\chi}$  Independent of any previous data

For 
$$t = 0 \rightarrow T - 1$$

2. Learner pulls arm  $a_t = \pi_t(x_t) \in \{1, \dots, K\}$ 

Formally, a contextual bandit is the following interactive learning process:

1. Learner sees context  $x_t \sim \nu_x$  Independent of any previous data

For 
$$t = 0 \rightarrow T - 1$$

2. Learner pulls arm  $a_t = \pi_t(x_t) \in \{1, \dots, K\}$ 

Formally, a contextual bandit is the following interactive learning process:

1. Learner sees context  $x_t \sim \nu_x$  Independent of any previous data  $\pi_t$  policy learned from all data seen so far





For 
$$t = 0 \rightarrow T - 1$$

2. Learner pulls arm  $a_t = \pi_t(x_t) \in \{1, \dots, K\}$ 

Formally, a contextual bandit is the following interactive learning process:

1. Learner sees context  $x_t \sim \nu_x$  Independent of any previous data  $\pi_t$  policy learned from all data seen so far 3. Learner observes reward  $r_t \sim \nu^{(a_t)}(x_t)$  from arm  $a_t$  in context  $x_t$ 





For  $t = 0 \rightarrow T - 1$ 

2. Learner pulls arm  $a_t = \pi_t(x_t) \in \{1, \dots, K\}$ 

Note that if the context distribution  $\nu_{\chi}$  always returns the same value (e.g., 0), then the contextual bandit <u>reduces</u> to the original multi-armed bandit

#### Formally, a contextual bandit is the following interactive learning process:

1. Learner sees context  $x_t \sim \nu_x$  Independent of any previous data  $\pi_t$  policy learned from all data seen so far 3. Learner observes reward  $r_t \sim \nu^{(a_t)}(x_t)$  from arm  $a_t$  in context  $x_t$ 



For  $t = 0 \rightarrow T - 1$ 

1. Learner sees context  $x_t \sim \nu_x$  Independent of any previous data  $\pi_t$  policy learned from 2. Learner pulls arm  $a_t = \pi_t(x_t) \in \{1, \dots, K\}$ all data seen so far 3. Learner observes reward  $r_t \sim \nu^{(a_t)}(x_t)$  from arm  $a_t$  in context  $x_t$ 

 $\pi_{t}$  might seem unfamiliar since we haven't talked about a policy in bandits before, but actually we've always had it, it's just that without context, we didn't need a name or notation for it because it was so simple!

#### Formally, a contextual bandit is the following interactive learning process:

Note that if the context distribution  $\nu_{\chi}$  always returns the same value (e.g., 0), then the contextual bandit <u>reduces</u> to the original multi-armed bandit



What was  $\pi_t$  for UCB? ( $\pi_t$  has no argument because there was no context)

What was  $\pi_t$  for UCB? ( $\pi_t$  has no argument because there was no context)  $\pi_t = \arg \max_k \text{UCB}_t^{(k)}$ 

For Thompson sampling?

What was  $\pi_t$  for UCB? ( $\pi_t$  has no argument because there was no context)  $\pi_t = \arg\max_k \mathsf{UCB}_t^{(k)}$ 

 $\pi_t$  was a randomized policy that sampled from the posterior distribution of  $k^{\star}$ 

What was  $\pi_t$  for UCB? ( $\pi_t$  has no argument because there was no context)  $\pi_t = \underset{k}{\operatorname{arg\,max\,UCB}_t^{(k)}}$ 

For Thompson sampling?

- For Thompson sampling?
- $\pi_t$  was a randomized policy that sampled from the posterior distribution of  $k^{\star}$ 
  - Now what about contextual versions?

What was  $\pi_t$  for UCB? ( $\pi_t$  has no argument because there was no context)  $\pi_t = \underset{k}{\operatorname{arg\,max\,UCB}_t^{(k)}}$ 

- For Thompson sampling?
- $\pi_t$  was a randomized policy that sampled from the posterior distribution of  $k^{\star}$

What was  $\pi_t$  for UCB? ( $\pi_t$  has no argument because there was no context)  $\pi_t = \underset{k}{\operatorname{arg\,max\,UCB}_t^{(k)}}$ 

#### Now what about contextual versions?

Thompson sampling with contexts is conceptually identical!

- For Thompson sampling?  $\pi_t$  was a randomized policy that sampled from the posterior distribution of  $k^{\star}$ 
  - Now what about contextual versions?
  - Thompson sampling with contexts is conceptually identical! Still start from a prior on  $\{\nu^{(k)}(x)\}_{k \in \{1,...,K\}, x \in \mathcal{X}}$ ,

What was  $\pi_t$  for UCB? ( $\pi_t$  has no argument because there was no context)  $\pi_t = \underset{k}{\operatorname{arg\,max\,UCB}_t^{(k)}}$ 

- For Thompson sampling?
- $\pi_t$  was a randomized policy that sampled from the posterior distribution of  $k^{\star}$

- Thompson sampling with contexts is conceptually identical!
  - Still start from a prior on  $\{\nu^{(k)}(x)\}_{k \in \{1, \dots, K\}, x \in \mathcal{X}}$ ,
- but now this is  $K|\mathcal{X}|$  (usually  $\gg K$ ) distributions, so need more complicated prior

What was  $\pi_{t}$  for UCB? ( $\pi_{t}$  has no argument because there was no context)  $\pi_t = \underset{k}{\operatorname{arg\,max\,UCB}_t^{(k)}}$ 

Now what about contextual versions?

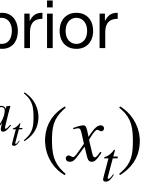


- For Thompson sampling?
- $\pi_t$  was a randomized policy that sampled from the posterior distribution of  $k^{\star}$

#### Now what about contextual versions?

- Thompson sampling with contexts is conceptually identical!
  - Still start from a prior on  $\{\nu^{(k)}(x)\}_{k \in \{1,...,K\}, x \in \mathcal{X}}$ ,
- but now this is  $K[\mathcal{X}]$  (usually  $\gg K$ ) distributions, so need more complicated prior
- Still can update distribution on  $\{\nu^{(k)}(x)\}_{k \in \{1,...,K\}, x \in \mathcal{X}}$  after each reward  $r_t \sim \nu^{(a_t)}(x_t)$

What was  $\pi_t$  for UCB? ( $\pi_t$  has no argument because there was no context)  $\pi_t = \underset{k}{\operatorname{arg\,max\,UCB}_t^{(k)}}$ 

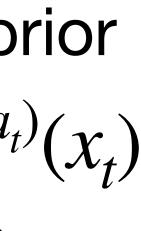


- For Thompson sampling?
- $\pi_t$  was a randomized policy that sampled from the posterior distribution of  $k^{\star}$

#### Now what about contextual versions?

- Thompson sampling with contexts is conceptually identical!
  - Still start from a prior on  $\{\nu^{(k)}(x)\}_{k \in \{1,...,K\}, x \in \mathcal{X}}$ ,
- but now this is  $K[\mathcal{X}]$  (usually  $\gg K$ ) distributions, so need more complicated prior
- Still can update distribution on  $\{\nu^{(k)}(x)\}_{k \in \{1,...,K\}, x \in \mathcal{X}}$  after each reward  $r_t \sim \nu^{(a_t)}(x_t)$ Still know posterior over  $k^{\star}(x_t)$  that can draw from to choose  $a_t$ ; this is  $\pi_t(x_t)$

What was  $\pi_t$  for UCB? ( $\pi_t$  has no argument because there was no context)  $\pi_t = \underset{k}{\operatorname{arg\,max\,UCB}_t^{(k)}}$ 

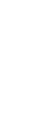




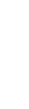












































$$\pi_t(x_t) = \arg\max_k \hat{\mu}_t^{(k)}(x_t)$$

#### UCB algorithm also conceptually identical as long as $|\mathcal{X}|$ finite: $+\sqrt{\ln(2TK|\mathcal{X}|\delta)/2N_t^{(k)}(x_t)}$

$$\pi_t(x_t) = \arg\max_k \hat{\mu}_t^{(k)}(x_t)$$

• Added  $x_t$  argument to  $\hat{\mu}_t^{(k)}$  and  $N_t^{(k)}$  since we now keep track of the sample mean and number of arm pulls separately for each value of the context

- UCB algorithm also conceptually identical as long as  $|\mathcal{X}|$  finite:
  - $+\sqrt{\ln(2TK|\mathcal{X}|/\delta)/2N_t^{(k)}(x_t)}$



$$\pi_t(x_t) = \arg\max_k \hat{\mu}_t^{(k)}(x_t)$$

- mean and number of arm pulls separately for each value of the context
- Added  $x_t$  argument to  $\hat{\mu}_t^{(k)}$  and  $N_t^{(k)}$  since we now keep track of the sample • Added  $|\mathcal{X}|$  inside the log because our union bound argument is now over all arm mean estimates  $\hat{\mu}_{t}^{(k)}(x)$ , of which there are  $K|\mathcal{X}|$  instead of just K
- UCB algorithm also conceptually identical as long as  $|\mathcal{X}|$  finite:  $+\sqrt{\ln(2TK|\mathcal{X}|\delta)/2N_t^{(k)}(x_t)}$



$$\pi_t(x_t) = \arg\max_k \hat{\mu}_t^{(k)}(x_t)$$

- mean and number of arm pulls separately for each value of the context all arm mean estimates  $\hat{\mu}_{t}^{(k)}(x)$ , of which there are  $K|\mathcal{X}|$  instead of just K
- Added  $x_t$  argument to  $\hat{\mu}_t^{(k)}$  and  $N_t^{(k)}$  since we now keep track of the sample • Added  $|\mathcal{X}|$  inside the log because our union bound argument is now over
  - But when  $|\mathcal{X}|$  is really big (or even infinite), this will be really bad!

UCB algorithm also conceptually identical as long as  $|\mathcal{X}|$  finite:  $+\sqrt{\ln(2TK|\mathcal{X}|\delta)/2N_t^{(k)}(x_t)}$ 



$$\pi_t(x_t) = \arg\max_k \hat{\mu}_t^{(k)}(x_t)$$

- mean and number of arm pulls separately for each value of the context all arm mean estimates  $\hat{\mu}_{t}^{(k)}(x)$ , of which there are  $K|\mathcal{X}|$  instead of just K
- Added  $x_t$  argument to  $\hat{\mu}_t^{(k)}$  and  $N_t^{(k)}$  since we now keep track of the sample - Added  $|\mathcal{X}|$  inside the log because our union bound argument is now over

UCB algorithm also conceptually identical as long as  $|\mathcal{X}|$  finite:  $+\sqrt{\ln(2TK|\mathcal{X}|\delta)/2N_t^{(k)}(x_t)}$ 

But when  $|\mathcal{X}|$  is really big (or even infinite), this will be really bad!

<u>Solution</u>: share information across contexts  $x_t$ , i.e., <u>don't</u> treat  $\nu^{(k)}(x)$  and  $\nu^{(k)}(x')$  as completely different distributions which have nothing to do with one another



- mean and number of arm pulls separately for each value of the context all arm mean estimates  $\hat{\mu}_{t}^{(k)}(x)$ , of which there are  $K|\mathcal{X}|$  instead of just K
- Added  $x_t$  argument to  $\hat{\mu}_t^{(k)}$  and  $N_t^{(k)}$  since we now keep track of the sample - Added  $|\mathcal{X}|$  inside the log because our union bound argument is now over

UCB algorithm also conceptually identical as long as  $|\mathcal{X}|$  finite:  $\pi_t(x_t) = \arg\max_k \hat{\mu}_t^{(k)}(x_t) + \sqrt{\ln(2TK|\mathcal{X}|/\delta)/2N_t^{(k)}(x_t)}$ 

But when  $|\mathcal{X}|$  is really big (or even infinite), this will be really bad!

<u>Solution</u>: share information across contexts  $x_t$ , i.e., <u>don't</u> treat  $\nu^{(k)}(x)$  and  $\nu^{(k)}(x')$  as completely different distributions which have nothing to do with one another Example: showing an ad on a NYT article on politics vs a NYT article on sports:

$$\pi_t(x_t) = \arg\max_k \hat{\mu}_t^{(k)}(x_t)$$

- mean and number of arm pulls separately for each value of the context all arm mean estimates  $\hat{\mu}_{t}^{(k)}(x)$ , of which there are  $K|\mathcal{X}|$  instead of just K
- Added  $x_t$  argument to  $\hat{\mu}_t^{(k)}$  and  $N_t^{(k)}$  since we now keep track of the sample - Added  $|\mathcal{X}|$  inside the log because our union bound argument is now over

Not *identical* readership, but still both on NYT, so probably still similar readership!

UCB algorithm also conceptually identical as long as  $|\mathcal{X}|$  finite:  $+\sqrt{\ln(2TK|\mathcal{X}|\delta)/2N_t^{(k)}(x_t)}$ 

- But when  $|\mathcal{X}|$  is really big (or even infinite), this will be really bad!
- <u>Solution</u>: share information across contexts  $x_t$ , i.e., <u>don't</u> treat  $\nu^{(k)}(x)$  and  $\nu^{(k)}(x')$  as completely different distributions which have nothing to do with one another Example: showing an ad on a NYT article on politics vs a NYT article on sports:



Need a model for  $\mu^{(k)}(x)$ , e.g., a linear model:  $\mu^{(k)}(x) = \theta_k^T x$ 

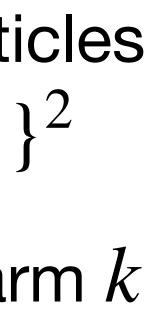
E.g., placing ads on NYT or WSJ (encoded as 0 or 1 in the first entry of x), for articles on politics or sports (encoded as 0 or 1 in the second entry of x)  $\Rightarrow x \in \{0,1\}^2$ 

Need a model for  $\mu^{(k)}(x)$ , e.g., a linear model:  $\mu^{(k)}(x) = \theta_k^T x$ 

Need a model for  $\mu^{(k)}(x)$ , e.g., a linear model:  $\mu^{(k)}(x) = \theta_k^T x$ 

E.g., placing ads on NYT or WSJ (encoded as 0 or 1 in the first entry of x), for articles on politics or sports (encoded as 0 or 1 in the second entry of x)  $\Rightarrow x \in \{0,1\}^2$ 

 $|\mathcal{X}| = 4 \Rightarrow$  w/o linear model, need to learn 4 different  $\mu^{(k)}(x)$  values for each arm k

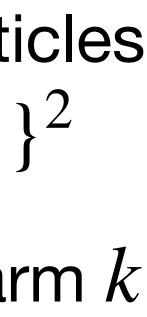


Need a model for  $\mu^{(k)}(x)$ , e.g., a linear model:  $\mu^{(k)}(x) = \theta_k^\top x$ 

E.g., placing ads on NYT or WSJ (encoded as 0 or 1 in the first entry of x), for articles on politics or sports (encoded as 0 or 1 in the second entry of x)  $\Rightarrow x \in \{0,1\}^2$ 

With linear model there are just 2 parameters: the two entries of  $\theta_k \in \mathbb{R}^2$ 

 $|\mathcal{X}| = 4 \Rightarrow$  w/o linear model, need to learn 4 different  $\mu^{(k)}(x)$  values for each arm k



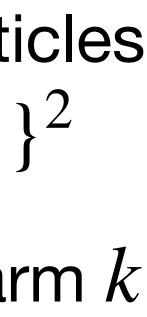
Need a model for  $\mu^{(k)}(x)$ , e.g., a linear model:  $\mu^{(k)}(x) = \theta_k^T x$ 

E.g., placing ads on NYT or WSJ (encoded as 0 or 1 in the first entry of x), for articles on politics or sports (encoded as 0 or 1 in the second entry of x)  $\Rightarrow x \in \{0,1\}^2$ 

Lower dimension makes learning easier, but model could be wrong/biased

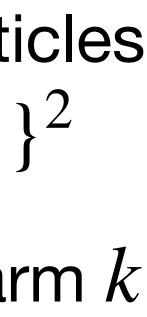
 $|\mathcal{X}| = 4 \Rightarrow$  w/o linear model, need to learn 4 different  $\mu^{(k)}(x)$  values for each arm k

With linear model there are just 2 parameters: the two entries of  $\theta_k \in \mathbb{R}^2$ 



Need a model for  $\mu^{(k)}(x)$ , e.g., a linear model:  $\mu^{(k)}(x) = \theta_k^T x$ 

- E.g., placing ads on NYT or WSJ (encoded as 0 or 1 in the first entry of x), for articles on politics or sports (encoded as 0 or 1 in the second entry of x)  $\Rightarrow x \in \{0,1\}^2$
- $|\mathcal{X}| = 4 \Rightarrow$  w/o linear model, need to learn 4 different  $\mu^{(k)}(x)$  values for each arm k
  - With linear model there are just 2 parameters: the two entries of  $\theta_k \in \mathbb{R}^2$
  - Lower dimension makes learning easier, but model could be wrong/biased
    - Choosing the best model, fitting it, and quantifying uncertainty are really questions of <u>supervised learning</u>





- Recap
- Thompson sampling
- Contextual bandits



#### Summary:

- Thompson sampling samples optimal arm from its (posterior) distribution Thompson sampling achieves excellent performance in practice
- Contextual bandits adds state to bandit problem, but algorithms extend
- For better performance, need modeling via supervised learning

#### Attendance:

bit.ly/3RcTC9T



Feedback:



