Bandits: Thompson Sampling and Contextual Bandits

Lucas Janson

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

Today

- 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

Today

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

Bayesian bandit

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: $\pi(\nu^{(1)}, \ldots, \nu^{(K)})$

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: $\pi(\nu^{(1)}, \ldots, \nu^{(K)})$

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 $\mu^{(k)}$

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: $\pi(\nu^{(1)}, \ldots, \nu^{(K)})$

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.,
$(\mu^{(1)}, \ldots, \mu^{(K)}) =: \boldsymbol{\mu} \sim \text{Uniform}([0,1]^K)$

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: $\pi(\nu^{(1)}, \ldots, \nu^{(K)})$

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., $(\mu^{(1)}, \ldots, \mu^{(K)}) =: \boldsymbol{\mu} \sim \text{Uniform}([0,1]^K)$

Note that the Bernoulli bandit reduced everything unknown about the bandit system to a K-dimensional vector $\boldsymbol{\mu}$

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: $\pi(\nu^{(1)}, \ldots, \nu^{(K)})$

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., $(\mu^{(1)}, \ldots, \mu^{(K)}) =: \boldsymbol{\mu} \sim \text{Uniform}([0,1]^K)$

Note that the Bernoulli bandit reduced everything unknown about the bandit system to a K-dimensional vector $\boldsymbol{\mu}$

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

Bayesian bandit summary

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 $\boldsymbol{\mu}$, so prior is: $\pi(\boldsymbol{\mu})$

Bayesian bandit summary

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 $\boldsymbol{\mu}$, so prior is: $\pi(\boldsymbol{\mu})$

Bayes rule at time step t gives us a distribution (called the posterior distribution)

$$\mathbb{P}(\boldsymbol{\mu} \mid r_0, a_0, r_1, a_1, \ldots, r_{t-1}, a_{t-1})$$

that exactly characterizes our uncertainty about $\boldsymbol{\mu}$

Bayesian bandit summary

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 $\boldsymbol{\mu}$, so prior is: $\pi(\boldsymbol{\mu})$

Bayes rule at time step t gives us a distribution (called the posterior distribution)

$$\mathbb{P}(\boldsymbol{\mu} \mid r_0, a_0, r_1, a_1, \ldots, r_{t-1}, a_{t-1})$$

that exactly characterizes our uncertainty about $\boldsymbol{\mu}$

Bernoulli bandit's posterior is Beta-distributed with simple parameter updates

Bayesian bandit summary

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 $\boldsymbol{\mu}$, so prior is: $\pi(\boldsymbol{\mu})$

Bayes rule at time step t gives us a distribution (called the posterior distribution)

$$\mathbb{P}(\boldsymbol{\mu} \mid r_0, a_0, r_1, a_1, \ldots, r_{t-1}, a_{t-1})$$

that exactly characterizes our uncertainty about $\boldsymbol{\mu}$

Bernoulli bandit's posterior is Beta-distributed with simple parameter updates

Note that although we are now treating $\boldsymbol{\mu}$ as random, we still assume its value is only drawn once (from the prior) and then stays the same throughout t

Bayesian bandit summary

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 $\boldsymbol{\mu}$, so prior is: $\pi(\boldsymbol{\mu})$

Bayes rule at time step t gives us a distribution (called the posterior distribution)

$$\mathbb{P}(\boldsymbol{\mu} \mid r_0, a_0, r_1, a_1, \ldots, r_{t-1}, a_{t-1})$$

that exactly characterizes our uncertainty about $\boldsymbol{\mu}$

Bernoulli bandit's posterior is Beta-distributed with simple parameter updates

Note that although we are now treating $\boldsymbol{\mu}$ 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 $\boldsymbol{\mu}$, i.e., the posterior distribution, as we collect more and more data by pulling arms via a bandit algorithm

Today

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

Thompson sampling

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

Thompson sampling

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 $\mathbb{P}(k = k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1})$

Thompson sampling

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 $\mathbb{P}(k = k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1})$

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

Thompson sampling

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 $\mathbb{P}(k = k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1})$

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

For $t = 0,\ldots,T-1$:

$a_t \sim$ distribution of $k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1}$

Thompson sampling

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 $\mathbb{P}(k = k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1})$

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

For $t = 0,\ldots, T-1$:

$a_t \sim$ distribution of $k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1}$

How can we sample from this distribution?

Thompson sampling

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 $\mathbb{P}(k = k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1})$

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

For $t = 0, \ldots, T-1$:

$a_t \sim$ distribution of $k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1}$

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

Thompson sampling

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 $\mathbb{P}(k = k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1})$

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

For $t = 0,\ldots,T-1$:

$a_t \sim$ distribution of $k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1}$

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

That's it! Statistically, this is a super simple and elegant algorithm

Thompson sampling

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 $\mathbb{P}(k = k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1})$

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

For $t = 0, \ldots, T-1$:

$a_t \sim$ distribution of $k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1}$

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

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)

Thompson sampling intuition

Thompson sampling: $a_t \sim$ distribution of $k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1}$

Thompson sampling intuition

Thompson sampling: $a_t \sim$ distribution of $k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1}$

Why is this a good idea?

Thompson sampling intuition

Thompson sampling: $a_t \sim$ distribution of $k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1}$

Why is this a good idea?

A good tradeoff of exploration vs exploitation should:

Thompson sampling intuition

Thompson sampling: $a_t \sim$ distribution of $k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1}$

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 intuition

Thompson sampling: $a_t \sim$ distribution of $k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1}$

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 intuition

Thompson sampling: $a_t \sim$ distribution of $k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1}$

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

Thompson sampling intuition

Thompson sampling: $a_t \sim$ distribution of $k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1}$

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

Intuitively: want to sample arms proportionally to how promising they are

Thompson sampling intuition

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

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

Intuitively: want to sample arms proportionally to how promising they are

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"

Thompson sampling intuition

Thompson sampling: $a_t \sim$ distribution of $k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1}$

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

Intuitively: want to sample arms proportionally to how promising they are

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 δ tuning parameter, but do have to choose prior π

Thompson sampling intuition

Thompson sampling: $a_t \sim$ distribution of $k^\star \mid r_0, a_0, \ldots, r_{t-1}, a_{t-1}$

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

Intuitively: want to sample arms proportionally to how promising they are

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 δ tuning parameter, but do have to choose prior π

π 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 vs other algorithms

Thompson sampling samples arms proportionally to how promising they are

Thompson sampling vs other algorithms

Thompson sampling samples arms proportionally to how promising they are

Note this sampling is much more sophisticated than, say, ε-greedy, which really just samples according to 2 categories: “most promising” and “other”

Thompson sampling vs other algorithms

Thompson sampling samples arms proportionally to how promising they are

Note this sampling is much more sophisticated than, say, ε-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 vs other algorithms

Thompson sampling samples arms proportionally to how promising they are

Note this sampling is much more sophisticated than, say, ε-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 vs other algorithms

Thompson sampling samples arms proportionally to how promising they are

Note this sampling is much more sophisticated than, say, ε-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 vs other algorithms

Thompson sampling samples arms proportionally to how promising they are

Note this sampling is much more sophisticated than, say, ε-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 in practice

Thompson sampling in practice

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

Thompson sampling in practice

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

Thompson sampling in practice

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 any bandit algorithm:

$$\liminf_{T \to \infty} \frac{\mathbb{E}[N_T^{(k)}]}{\ln(T)} \geq \frac{1}{d(\nu^{(k^\star)}, \nu^{(k)})},$$

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

Thompson sampling in practice

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 any bandit algorithm:

$$\liminf_{T\to\infty} \frac{\mathbb{E}[N_T^{(k)}]}{\ln(T)} \geq \frac{1}{d(\nu^{(k^\star)}, \nu^{(k)})},$$

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 in practice

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 any bandit algorithm:

$$\liminf_{T \to \infty} \frac{\mathbb{E}[N_T^{(k)}]}{\ln(T)} \geq \frac{1}{d(\nu^{(k^\star)}, \nu^{(k)})},$$

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 in practice

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 any bandit algorithm:

$$\liminf_{T\to\infty} \frac{\mathbb{E}[N_T^{(k)}]}{\ln(T)} \geq \frac{1}{d(\nu^{(k^\star)}, \nu^{(k)})},$$

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)

Thompson sampling in practice (cont'd)

So Thompson sampling is basically exactly optimal for large T

Thompson sampling in practice (cont'd)

So Thompson sampling is basically exactly optimal for large T

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

Thompson sampling in practice (cont'd)

So Thompson sampling is basically exactly optimal for large T

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

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

Thompson sampling in practice (cont'd)

So Thompson sampling is basically exactly optimal for large T

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

- $t = 0$: $a_0 = 1$, $r_0 = 1$
- $t = 1$: $a_1 = 2$, $r_1 = 0$

Thompson sampling in practice (cont'd)

So Thompson sampling is basically exactly optimal for large T

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

Thompson sampling in practice (cont'd)

So Thompson sampling is basically exactly optimal for large T

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

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

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

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.

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

Thompson sampling in practice (cont'd)

So Thompson sampling is basically exactly optimal for large T

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

- $t = 0$: $a_0 = 1$, $r_0 = 1$
- $t = 1$: $a_1 = 1$, $r_1 = 0$
- $t = 2$ (last time step, with $\hat{\mu}_2^{(1)} = 1$ and $\hat{\mu}_2^{(2)} = 0$): $a_2 = ?$

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.

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

For small T, Thompson sampling is not greedy enough

Thompson sampling in practice (cont'd)

For small T, Thompson sampling is not greedy enough

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

Thompson sampling in practice (cont'd)

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

Thompson sampling in practice (cont'd)

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 $\boldsymbol{\mu}$ 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

Thompson sampling in practice (cont'd)

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 $\boldsymbol{\mu}$ 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

All of these favor arms that the algorithm has more confidence are good (i.e., arms that have worked well so far), as opposed to arms that *may* be good

Thompson sampling in practice (cont'd)

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 $\boldsymbol{\mu}$ 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

All of these favor arms that the algorithm has more confidence are good (i.e., arms that have worked well so far), as opposed to arms that *may* be good

Such tuning can improve Thompson sampling's performance even for reasonably large T (the asymptotic optimality of vanilla TS is *very* asymptotic)

Exactly optimality in Bayesian Bernoulli bandit

Exactly optimality in Bayesian Bernoulli bandit

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

Exactly optimality in Bayesian Bernoulli bandit

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

Gittins index 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

Exactly optimality in Bayesian Bernoulli bandit

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

Gittins index 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

Exactly optimality in Bayesian Bernoulli bandit

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

Gittins index 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

Today

- Feedback from last lecture
- 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 a bandit, we are presented with the same decision at every time

In practice, often decisions are not the same every time

Beyond simple bandits

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:

Beyond simple bandits

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

Beyond simple bandits

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

Beyond simple bandits

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

Contextual bandit environment

Contextual bandit environment

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

Contextual bandit environment

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 ν_x on sample space $\mathcal{X}$

Contextual bandit environment

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 ν_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, $\nu^{(k)}(x_t)$

Contextual bandit environment

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 ν_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, $\pi_t(x_t)$

Contextual bandit environment

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 ν_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

$$\pi^\star(x_t) := \arg\max_{k \in \{1,\ldots,K\}} \mu^{(k)}(x_t), \qquad \text{where } \mu^{(k)}(x) := \mathbb{E}_{r \sim \nu^{(k)}(x)}[r]$$

Contextual bandit environment

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 ν_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, $\pi_t(x_t)$

If we knew everything about the environment, we'd want to use the optimal policy

$$\pi^\star(x_t) := \arg\max_{k \in \{1,\ldots,K\}} \mu^{(k)}(x_t), \qquad \text{where } \mu^{(k)}(x) := \mathbb{E}_{r \sim \nu^{(k)}(x)}[r]$$

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

Contextual bandit environment (cont'd)

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

Contextual bandit environment (cont'd)

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

For $t = 0 \to T - 1$

Contextual bandit environment (cont'd)

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

For $t = 0 \rightarrow T - 1$

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

Contextual bandit environment (cont'd)

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

For $t = 0 \to T - 1$

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

Contextual bandit environment (cont'd)

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

For $t = 0 \to T - 1$

1. Learner sees context $x_t \sim \nu_x$ Independent of any previous data
2. Learner pulls arm $a_t = \pi_t(x_t) \in \{1, \ldots, K\}$

Contextual bandit environment (cont'd)

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

For $t = 0 \to T - 1$

1. Learner sees context $x_t \sim \nu_x$ — Independent of any previous data
2. Learner pulls arm $a_t = \pi_t(x_t) \in \{1, \ldots, K\}$ — π_t policy learned from all data seen so far

Contextual bandit environment (cont'd)

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

For $t = 0 \to T - 1$

1. Learner sees context $x_t \sim \nu_x$ — Independent of any previous data
2. Learner pulls arm $a_t = \pi_t(x_t) \in \{1, \ldots, K\}$ — π_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

Contextual bandit environment (cont'd)

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

For $t = 0 \rightarrow T - 1$

1. Learner sees context $x_t \sim \nu_x$ — Independent of any previous data
2. Learner pulls arm $a_t = \pi_t(x_t) \in \{1,\ldots,K\}$ — π_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

Note that if the context distribution ν_x always returns the same value (e.g., 0), then the contextual bandit reduces to the original multi-armed bandit

Contextual bandit environment (cont'd)

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

For $t = 0 \to T - 1$

1. Learner sees context $x_t \sim \nu_x$ — Independent of any previous data
2. Learner pulls arm $a_t = \pi_t(x_t) \in \{1, \ldots, K\}$ — π_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

Note that if the context distribution ν_x always returns the same value (e.g., 0), then the contextual bandit reduces to the original multi-armed bandit

π_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!

Contextual bandit algorithms

What was π_t for UCB? (π_t has no argument because there was no context)

Contextual bandit algorithms

What was π_t for UCB? (π_t has no argument because there was no context)

$$\pi_t = \arg\max_k \mathsf{UCB}_t^{(k)}$$

Contextual bandit algorithms

What was π_t for UCB? (π_t has no argument because there was no context)

$$\pi_t = \arg\max_k \text{UCB}_t^{(k)}$$

For Thompson sampling?

Contextual bandit algorithms

What was π_t for UCB? (π_t has no argument because there was no context)

$$\pi_t = \arg\max_k \mathsf{UCB}_t^{(k)}$$

For Thompson sampling?

π_t was a *randomized* policy that sampled from the posterior distribution of $k^\star$

Contextual bandit algorithms

What was π_t for UCB? (π_t has no argument because there was no context)

$$\pi_t = \arg\max_k \text{UCB}_t^{(k)}$$

For Thompson sampling?

π_t was a *randomized* policy that sampled from the posterior distribution of $k^\star$

Now what about contextual versions?

Contextual bandit algorithms

What was π_t for UCB? (π_t has no argument because there was no context)

$$\pi_t = \arg\max_k \text{UCB}_t^{(k)}$$

For Thompson sampling?

π_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!

Contextual bandit algorithms

What was π_t for UCB? (π_t has no argument because there was no context)

$$\pi_t = \arg\max_k \text{UCB}_t^{(k)}$$

For Thompson sampling?

π_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 $\{\nu^{(k)}(x)\}_{k\in\{1,\ldots,K\},x\in\mathcal{X}}$,

Contextual bandit algorithms

What was π_t for UCB? (π_t has no argument because there was no context)

$$\pi_t = \arg\max_k \text{UCB}_t^{(k)}$$

For Thompson sampling?

π_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 $\{\nu^{(k)}(x)\}_{k\in\{1,\ldots,K\},x\in\mathcal{X}}$,

but now this is $K|\mathcal{X}|$ (usually $\gg K$) distributions, so need more complicated prior

Contextual bandit algorithms

What was π_t for UCB? (π_t has no argument because there was no context)

$$\pi_t = \arg\max_k \text{UCB}_t^{(k)}$$

For Thompson sampling?

π_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 $\{\nu^{(k)}(x)\}_{k\in\{1,\ldots,K\},x\in\mathcal{X}}$,

but now this is $K|\mathcal{X}|$ (usually $\gg K$) distributions, so need more complicated prior

Still can update distribution on $\{\nu^{(k)}(x)\}_{k\in\{1,\ldots,K\},x\in\mathcal{X}}$ after each reward $r_t \sim \nu^{(a_t)}(x_t)$

Contextual bandit algorithms

What was π_t for UCB? (π_t has no argument because there was no context)

$$\pi_t = \arg\max_k \text{UCB}_t^{(k)}$$

For Thompson sampling?

π_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 $\{\nu^{(k)}(x)\}_{k\in\{1,\ldots,K\},x\in\mathcal{X}}$,

but now this is $K|\mathcal{X}|$ (usually $\gg K$) distributions, so need more complicated prior

Still can update distribution on $\{\nu^{(k)}(x)\}_{k\in\{1,\ldots,K\},x\in\mathcal{X}}$ 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 $\pi_t(x_t)$

UCB for contextual bandits

UCB for contextual bandits

UCB algorithm also conceptually identical as long as $|\mathcal{X}|$ finite:

$$\pi_t(x_t) = \arg\max_k \hat{\mu}_t^{(k)}(x_t) + \sqrt{\ln(2TK|\mathcal{X}|/\delta)/2N_t^{(k)}(x_t)}$$

UCB for contextual bandits

UCB algorithm also conceptually identical as long as $|\mathcal{X}|$ finite:

$$\pi_t(x_t) = \arg\max_k \hat{\mu}_t^{(k)}(x_t) + \sqrt{\ln(2TK|\mathcal{X}|/\delta)/2N_t^{(k)}(x_t)}$$

- 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 for contextual bandits

UCB algorithm also conceptually identical as long as $|\mathcal{X}|$ finite:

$$\pi_t(x_t) = \arg\max_k \hat{\mu}_t^{(k)}(x_t) + \sqrt{\ln(2TK|\mathcal{X}|/\delta)/2N_t^{(k)}(x_t)}$$

- 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
- 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 for contextual bandits

UCB algorithm also conceptually identical as long as $|\mathscr{X}|$ finite:

$$\pi_t(x_t) = \arg\max_k \hat{\mu}_t^{(k)}(x_t) + \sqrt{\ln(2TK|\mathscr{X}|/\delta)/2N_t^{(k)}(x_t)}$$

- 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
- Added $|\mathscr{X}|$ inside the log because our union bound argument is now over all arm mean estimates $\hat{\mu}_t^{(k)}(x)$, of which there are $K|\mathscr{X}|$ instead of just K

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

UCB for contextual bandits

UCB algorithm also conceptually identical as long as $|\mathcal{X}|$ finite:

$$\pi_t(x_t) = \arg\max_k \hat{\mu}_t^{(k)}(x_t) + \sqrt{\ln(2TK|\mathcal{X}|/\delta)/2N_t^{(k)}(x_t)}$$

- 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
- 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

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

UCB for contextual bandits

UCB algorithm also conceptually identical as long as $|\mathcal{X}|$ finite:

$$\pi_t(x_t) = \arg\max_k \hat{\mu}_t^{(k)}(x_t) + \sqrt{\ln(2TK|\mathcal{X}|/\delta)/2N_t^{(k)}(x_t)}$$

- 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
- 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

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

Solution: share information across contexts x_t, i.e., don't 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:

UCB for contextual bandits

UCB algorithm also conceptually identical as long as $|\mathcal{X}|$ finite:

$$\pi_t(x_t) = \arg\max_k \hat{\mu}_t^{(k)}(x_t) + \sqrt{\ln(2TK|\mathcal{X}|/\delta)/2N_t^{(k)}(x_t)}$$

- 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
- 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

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

Solution: share information across contexts x_t, i.e., don't 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: Not *identical* readership, but still both on NYT, so probably still *similar* readership!

Modeling in contextual bandits

Modeling in contextual bandits

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

Modeling in contextual bandits

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) $\Rightarrow x \in \{0,1\}^2$

Modeling in contextual bandits

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) $\Rightarrow x \in \{0,1\}^2$

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

Modeling in contextual bandits

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) $\Rightarrow x \in \{0,1\}^2$

$|\mathcal{X}| = 4$ $\Rightarrow$ 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 $\theta_k \in \mathbb{R}^2$

Modeling in contextual bandits

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) $\Rightarrow x \in \{0,1\}^2$

$|\mathcal{X}| = 4$ $\Rightarrow$ 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

Modeling in contextual bandits

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) $\Rightarrow x \in \{0,1\}^2$

$|\mathcal{X}| = 4$ $\Rightarrow$ 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 supervised learning

Today

- Feedback from last lecture
- 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:
bit.ly/3RHtlxy