Bandits: Regret Lower Bound and Instance-Dependent Regret

Lucas Janson and Sham Kakade

CS/Stat 184: Introduction to Reinforcement Learning
Fall 2022

Today

- Feedback from last lecture
- Recap
- Regret *lower* bound
- Instance-dependent regret

Feedback from feedback forms

1. Thank you to everyone who filled out the forms!
2. Main feedback: **pace was good!**
3. Pre-lecture posted lecture notes shouldn’t maintain breaks within slides

Today

- ✓ Feedback from last lecture
- Recap
- Regret *lower* bound
- Instance-dependent regret

Recap

Recap

- Pure greedy and pure exploration achieve linear regret $O(T)$

Recap

- Pure greedy and pure exploration achieve linear regret $O(T)$
- ETC and ε-greedy achieve sublinear regret of $\tilde{O}(T^{2/3})$

Recap

- Pure greedy and pure exploration achieve linear regret $O(T)$
- ETC and ε-greedy achieve sublinear regret of $\tilde{O}(T^{2/3})$
- UCB achieves sublinear regret of $\tilde{O}(\sqrt{T})$

Recap

- Pure greedy and pure exploration achieve linear regret $O(T)$
- ETC and ε-greedy achieve sublinear regret of $\tilde{O}(T^{2/3})$
- UCB achieves sublinear regret of $\tilde{O}(\sqrt{T})$
- Can we do even better?

Today

- Feedback from last lecture
- Recap
- Regret *lower* bound
- Instance-dependent regret

Can we do better than $\Omega(\sqrt{T})$ regret?

Can we do better than $\Omega(\sqrt{T})$ regret?

Short answer: no

Can we do better than $\Omega(\sqrt{T})$ regret?

Short answer: no

But how can we know that?

Can we do better than $\Omega(\sqrt{T})$ regret?

Short answer: no

But how can we know that?

Want to construct a *lower bound* on the achievable regret

Can we do better than $\Omega(\sqrt{T})$ regret?

Short answer: no

But how can we know that?

Want to construct a *lower bound* on the achievable regret

So far we our theoretical analysis has always considered a fixed algorithm and analyzed it (by deriving a regret upper bound with high probability)

Can we do better than $\Omega(\sqrt{T})$ regret?

Short answer: no

But how can we know that?

Want to construct a *lower bound* on the achievable regret

So far we our theoretical analysis has always considered a fixed algorithm and analyzed it (by deriving a regret upper bound with high probability)

To get a lower bound, we need to consider what regret could be achieved by *any* algorithm, and show it can't be better than some rate

Can we do better than $\Omega(\sqrt{T})$ regret?

Short answer: no

But how can we know that?

Want to construct a *lower bound* on the achievable regret

So far we our theoretical analysis has always considered a fixed algorithm and analyzed it (by deriving a regret upper bound with high probability)

To get a lower bound, we need to consider what regret could be achieved by *any* algorithm, and show it can't be better than some rate

Useful mathematical device: oracle
An oracle has access to extra information not available to bandit algorithms.

Can we do better than $\Omega(\sqrt{T})$ regret?

Short answer: no

But how can we know that?

Want to construct a *lower bound* on the achievable regret

So far we our theoretical analysis has always considered a fixed algorithm and analyzed it (by deriving a regret upper bound with high probability)

To get a lower bound, we need to consider what regret could be achieved by *any* algorithm, and show it can't be better than some rate

Useful mathematical device: oracle
An oracle has access to extra information not available to bandit algorithms.

If we can show that oracle can't do better than some rate, then no algorithm can

Intuition for lower bound

Intuition for lower bound

1. CLT tells us that with T i.i.d. samples from from a distribution ν, we can only learn ν's mean μ to within $\Omega(1/\sqrt{T})$

Intuition for lower bound

1. CLT tells us that with T i.i.d. samples from from a distribution ν, we can only learn ν's mean μ to within $\Omega(1/\sqrt{T})$
2. Then since in a bandit, we get at most T samples total, certainly we can't learn any of the arm means better than to within $\Omega(1/\sqrt{T})$

Intuition for lower bound

1. CLT tells us that with T i.i.d. samples from from a distribution ν, we can only learn ν's mean μ to within $\Omega(1/\sqrt{T})$
2. Then since in a bandit, we get at most T samples total, certainly we can't learn any of the arm means better than to within $\Omega(1/\sqrt{T})$
3. This means that if an arm $\tilde{k}$ is about $1/\sqrt{T}$ away from the best arm $k^\star$, then at no point during the bandit can we tell them apart with high probability

Intuition for lower bound

1. CLT tells us that with T i.i.d. samples from from a distribution ν, we can only learn ν's mean μ to within $\Omega(1/\sqrt{T})$
2. Then since in a bandit, we get at most T samples total, certainly we can't learn any of the arm means better than to within $\Omega(1/\sqrt{T})$
3. This means that if an arm $\tilde{k}$ is about $1/\sqrt{T}$ away from the best arm $k^\star$, then at no point during the bandit can we tell them apart with high probability
4. Thus, we should expect to sample $\tilde{k}$ roughly as often as $k^\star$, which is at best roughly $T/2$ times (if we ignore any other arms)

Intuition for lower bound

1. CLT tells us that with T i.i.d. samples from from a distribution ν, we can only learn ν's mean μ to within $\Omega(1/\sqrt{T})$
2. Then since in a bandit, we get at most T samples total, certainly we can't learn any of the arm means better than to within $\Omega(1/\sqrt{T})$
3. This means that if an arm $\tilde{k}$ is about $1/\sqrt{T}$ away from the best arm $k^\star$, then at no point during the bandit can we tell them apart with high probability
4. Thus, we should expect to sample $\tilde{k}$ roughly as often as $k^\star$, which is at best roughly $T/2$ times (if we ignore any other arms)
5. Finally, since the regret incurred each time we pull arm $\tilde{k}$ is $1/\sqrt{T}$, and we pull it $T/2$ times, we get a regret lower bound of $1/\sqrt{T} \times T/2 = \Omega(\sqrt{T})$

Coming up with an oracle

Coming up with an oracle

Any oracle will give us a lower bound, but if we make the oracle too strong, that lower bound will be too low/conservative

Coming up with an oracle

Any oracle will give us a lower bound, but if we make the oracle too strong, that lower bound will be too low/conservative

What is an oracle that knows more than any bandit algorithm, but not *too* much?

Coming up with an oracle

Any oracle will give us a lower bound, but if we make the oracle too strong, that lower bound will be too low/conservative

What is an oracle that knows more than any bandit algorithm, but not *too* much?

(also want oracle to be easy to study theoretically)

Coming up with an oracle

Any oracle will give us a lower bound, but if we make the oracle too strong, that lower bound will be too low/conservative

What is an oracle that knows more than any bandit algorithm, but not *too* much?

(also want oracle to be easy to study theoretically)

Proposal: let the oracle see rewards from all arms at every time step

Coming up with an oracle

Any oracle will give us a lower bound, but if we make the oracle too strong, that lower bound will be too low/conservative

What is an oracle that knows more than any bandit algorithm, but not *too* much?

(also want oracle to be easy to study theoretically)

Proposal: let the oracle see rewards from all arms at every time step

- This is definitely more than any bandit algorithm gets

Coming up with an oracle

Any oracle will give us a lower bound, but if we make the oracle too strong, that lower bound will be too low/conservative

What is an oracle that knows more than any bandit algorithm, but not *too* much?

(also want oracle to be easy to study theoretically)

Proposal: let the oracle see rewards from all arms at every time step

- This is definitely more than any bandit algorithm gets
- But oracle still has to learn from data, and only gets ~K times as much data as a bandit algorithm, which we might hope won't change its regret rate in T

Coming up with an oracle

Any oracle will give us a lower bound, but if we make the oracle too strong, that lower bound will be too low/conservative

What is an oracle that knows more than any bandit algorithm, but not *too* much?

(also want oracle to be easy to study theoretically)

Proposal: let the oracle see rewards from all arms at every time step

- This is definitely more than any bandit algorithm gets
- But oracle still has to learn from data, and only gets ~K times as much data as a bandit algorithm, which we might hope won't change its regret rate in T
- Theoretically, the oracle actually does see i.i.d. rewards from each arm

Coming up with an oracle

Any oracle will give us a lower bound, but if we make the oracle too strong, that lower bound will be too low/conservative

What is an oracle that knows more than any bandit algorithm, but not *too* much?

(also want oracle to be easy to study theoretically)

Proposal: let the oracle see rewards from all arms at every time step

- This is definitely more than any bandit algorithm gets
- But oracle still has to learn from data, and only gets ~K times as much data as a bandit algorithm, which we might hope won't change its regret rate in T
- Theoretically, the oracle actually does see i.i.d. rewards from each arm

(oracle still has to pick a single arm to pull a_t for each time)

Coming up with an oracle

Any oracle will give us a lower bound, but if we make the oracle too strong, that lower bound will be too low/conservative

What is an oracle that knows more than any bandit algorithm, but not *too* much?

(also want oracle to be easy to study theoretically)

Proposal: let the oracle see rewards from all arms at every time step

- This is definitely more than any bandit algorithm gets
- But oracle still has to learn from data, and only gets ~K times as much data as a bandit algorithm, which we might hope won't change its regret rate in T
- Theoretically, the oracle actually does see i.i.d. rewards from each arm

(oracle still has to pick a single arm to pull a_t for each time)

Additionally: oracle chooses all a_t after seeing all arm rewards up to time T

(one decision point makes theory easier)

Oracle strategy

Oracle gets to choose all a_t after seeing all T rewards from all arms: $\{r_t^{(k)}\}_{t=0,k=1}^{T-1,K}$

Oracle strategy

Oracle gets to choose all a_t after seeing all T rewards from all arms: $\{r_t^{(k)}\}_{t=0,k=1}^{T-1,K}$

So what's the best thing the oracle can do?

Oracle strategy

Oracle gets to choose all a_t after seeing all T rewards from all arms: $\{r_t^{(k)}\}_{t=0,k=1}^{T-1,K}$

So what's the best thing the oracle can do?

$a_t = \hat{k}_t := \arg \max_{k \in 1,\ldots,K} r_t^{(k)}$ clearly maximizes the total reward

Oracle strategy

Oracle gets to choose all a_t after seeing all T rewards from all arms: $\{r_t^{(k)}\}_{t=0,k=1}^{T-1,K}$

So what's the best thing the oracle can do?

$a_t = \hat{k}_t := \arg\max_{k \in 1,\ldots,K} r_t^{(k)}$ clearly maximizes the total reward

Consider 2-armed Bernoulli bandit with $T = 1000$, with $\hat{\mu}_T^{(1)} = 0.6$ and $\hat{\mu}_T^{(2)} = 0.4$.

Oracle strategy

Oracle gets to choose all a_t after seeing all T rewards from all arms: $\{r_t^{(k)}\}_{t=0,k=1}^{T-1,K}$

So what's the best thing the oracle can do?

$a_t = \hat{k}_t := \arg\max_{k \in 1,\ldots,K} r_t^{(k)}$ clearly maximizes the total reward

Consider 2-armed Bernoulli bandit with $T = 1000$, with $\hat{\mu}_T^{(1)} = 0.6$ and $\hat{\mu}_T^{(2)} = 0.4$.

These estimates are extremely good (CLT standard errors (SE) < 0.02):

Oracle strategy

Oracle gets to choose all a_t after seeing all T rewards from all arms: $\{r_t^{(k)}\}_{t=0,k=1}^{T-1,K}$

So what's the best thing the oracle can do?

$a_t = \hat{k}_t := \arg\max_{k \in 1,\ldots,K} r_t^{(k)}$ clearly maximizes the total reward

Consider 2-armed Bernoulli bandit with $T = 1000$, with $\hat{\mu}_T^{(1)} = 0.6$ and $\hat{\mu}_T^{(2)} = 0.4$.

These estimates are extremely good (CLT standard errors (SE) < 0.02):

- Oracle overwhelmingly confident that $\mu^{(1)} > \mu^{(2)}$ (estimates > 10 SEs apart)

Oracle strategy

Oracle gets to choose all a_t after seeing all T rewards from all arms: $\{r_t^{(k)}\}_{t=0,k=1}^{T-1,K}$

So what's the best thing the oracle can do?

$a_t = \hat{k}_t := \arg\max_{k \in 1,\ldots,K} r_t^{(k)}$ clearly maximizes the total reward

Consider 2-armed Bernoulli bandit with $T = 1000$, with $\hat{\mu}_T^{(1)} = 0.6$ and $\hat{\mu}_T^{(2)} = 0.4$.

These estimates are extremely good (CLT standard errors (SE) < 0.02):

- Oracle overwhelmingly confident that $\mu^{(1)} > \mu^{(2)}$ (estimates > 10 SEs apart)
- Roughly $0.4^2 = 16\,\%$ of the time, $r_t^{(1)} = 0 < 1 = r_t^{(2)}$

Oracle strategy

Oracle gets to choose all a_t after seeing all T rewards from all arms: $\{r_t^{(k)}\}_{t=0,k=1}^{T-1,K}$

So what's the best thing the oracle can do?

$a_t = \hat{k}_t := \arg\max_{k \in 1,\ldots,K} r_t^{(k)}$ clearly maximizes the total reward

Consider 2-armed Bernoulli bandit with $T = 1000$, with $\hat{\mu}_T^{(1)} = 0.6$ and $\hat{\mu}_T^{(2)} = 0.4$.

These estimates are extremely good (CLT standard errors (SE) < 0.02):

- Oracle overwhelmingly confident that $\mu^{(1)} > \mu^{(2)}$ (estimates > 10 SEs apart)
- Roughly $0.4^2 = 16\,\%$ of the time, $r_t^{(1)} = 0 < 1 = r_t^{(2)} \Rightarrow \hat{k}_t = 2$

Oracle strategy

Oracle gets to choose all a_t after seeing all T rewards from all arms: $\{r_t^{(k)}\}_{t=0,k=1}^{T-1,K}$

So what's the best thing the oracle can do?

$a_t = \hat{k}_t := \arg\max_{k \in 1,\ldots,K} r_t^{(k)}$ clearly maximizes the total reward

Consider 2-armed Bernoulli bandit with $T = 1000$, with $\hat{\mu}_T^{(1)} = 0.6$ and $\hat{\mu}_T^{(2)} = 0.4$.

These estimates are extremely good (CLT standard errors (SE) < 0.02):

- Oracle overwhelmingly confident that $\mu^{(1)} > \mu^{(2)}$ (estimates > 10 SEs apart)
- Roughly $0.4^2 = 16\,\%$ of the time, $r_t^{(1)} = 0 < 1 = r_t^{(2)} \Rightarrow \hat{k}_t = 2$

But $\text{Regret}_T = \sum_{t=0}^{T-1} (\mu^\star - \mu^{(a_t)})$ looks at the *true* mean of arm a_t, not actual reward…

Oracle strategy

Oracle gets to choose all a_t after seeing all T rewards from all arms: $\{r_t^{(k)}\}_{t=0,k=1}^{T-1,K}$

So what's the best thing the oracle can do?

$a_t = \hat{k}_t := \arg\max_{k \in 1,\ldots,K} r_t^{(k)}$ clearly maximizes the total reward

Consider 2-armed Bernoulli bandit with $T = 1000$, with $\hat{\mu}_T^{(1)} = 0.6$ and $\hat{\mu}_T^{(2)} = 0.4$.

These estimates are extremely good (CLT standard errors (SE) < 0.02):

- Oracle overwhelmingly confident that $\mu^{(1)} > \mu^{(2)}$ (estimates > 10 SEs apart)
- Roughly $0.4^2 = 16\,\%$ of the time, $r_t^{(1)} = 0 < 1 = r_t^{(2)} \Rightarrow \hat{k}_t = 2$

But $\text{Regret}_T = \sum_{t=0}^{T-1} (\mu^\star - \mu^{(a_t)})$ looks at the *true* mean of arm a_t, not actual reward…

32

$\text{Regret}_T \approx T \cdot 0.16(\mu^{(1)} - \mu^{(2)}) \approx 0.032$ for $a_t = \hat{k}_t$

Oracle strategy

Oracle gets to choose all a_t after seeing all T rewards from all arms: $\{r_t^{(k)}\}_{t=0,k=1}^{T-1,K}$

So what's the best thing the oracle can do?

$a_t = \hat{k}_t := \arg \max_{k \in 1,\ldots,K} r_t^{(k)}$ clearly maximizes the total reward

Consider 2-armed Bernoulli bandit with $T = 1000$, with $\hat{\mu}_T^{(1)} = 0.6$ and $\hat{\mu}_T^{(2)} = 0.4$.

These estimates are extremely good (CLT standard errors (SE) < 0.02):

- Oracle overwhelmingly confident that $\mu^{(1)} > \mu^{(2)}$ (estimates > 10 SEs apart)
- Roughly $0.4^2 = 16\,\%$ of the time, $r_t^{(1)} = 0 < 1 = r_t^{(2)}$ $\Rightarrow \hat{k}_t = 2$

But $\text{Regret}_T = \sum_{t=0}^{T-1} (\mu^\star - \mu^{(a_t)})$ looks at the *true* mean of arm a_t, not actual reward…

$\text{Regret}_T \approx 0.16(\mu^{(1)} - \mu^{(2)}) \approx 0.032$ for $a_t = \hat{k}_t$ but $a_t = 1\ \forall t$ gives $\text{Regret}_T \approx 0$

Oracle strategy (cont'd)

Best strategy in terms of maximizing $\sum_{t=0}^{T-1} \mu^{(a_t)}$ (i.e., minimizing Regret$_T$), is to choose every $a_t = \hat{k}_T = \arg\max_{k \in 1,\ldots,K} \hat{\mu}_T^{(k)}$, since $\hat{k}_T$ is the oracle's best guess of $k^\star$

Oracle strategy (cont'd)

Best strategy in terms of maximizing $\sum_{t=0}^{T-1} \mu^{(a_t)}$ (i.e., minimizing Regret_T), is to choose every $a_t = \hat{k}_T = \arg\max_{k \in 1,\ldots,K} \hat{\mu}_T^{(k)}$, since $\hat{k}_T$ is the oracle's best guess of $k^\star$

This was not mathematically rigorous, but hopefully you can see why this strategy is the best strategy the oracle could employ given the information it has

Oracle regret

Oracle regret

We know by the CLT that:

$$\hat{\mu}_T^{(k)} - \mu^{(k)} \approx \mathcal{N}\left(0, \frac{\mathsf{Var}_{r\sim\nu^{(k)}}(r)}{T}\right)$$

Oracle regret

We know by the CLT that:

$$\hat{\mu}_T^{(k)} - \mu^{(k)} \approx \mathcal{N}\left(0, \frac{\mathrm{Var}_{r\sim\nu^{(k)}}(r)}{T}\right)$$

Which means that

$$\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}$$

Oracle regret

We know by the CLT that:

$$\hat{\mu}_T^{(k)} - \mu^{(k)} \approx \mathcal{N}\left(0, \frac{\mathrm{Var}_{r \sim \nu^{(k)}}(r)}{T}\right)$$

Which means that

$$\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)} = (\hat{\mu}_T^{(k^\star)} - \mu^{(k^\star)}) - (\hat{\mu}_T^{(k)} - \mu^{(k)}) + (\mu^{(k^\star)} - \mu^{(k)})$$

Oracle regret

We know by the CLT that:

$$\hat{\mu}_T^{(k)} - \mu^{(k)} \approx \mathcal{N}\left(0, \frac{\mathrm{Var}_{r\sim\nu^{(k)}}(r)}{T}\right)$$

Which means that

$$\begin{aligned}\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)} &= (\hat{\mu}_T^{(k^\star)} - \mu^{(k^\star)}) - (\hat{\mu}_T^{(k)} - \mu^{(k)}) + (\mu^{(k^\star)} - \mu^{(k)}) \\ &\approx \mathcal{N}\left(\mu^{(k^\star)} - \mu^{(k)}, \frac{\mathrm{Var}_{r\sim\nu^{(k^\star)}}(r) + \mathrm{Var}_{r\sim\nu^{(k)}}(r)}{T}\right)\end{aligned}$$

Oracle regret

We know by the CLT that:

$$\hat{\mu}_T^{(k)} - \mu^{(k)} \approx \mathcal{N}\left(0, \frac{\mathrm{Var}_{r\sim\nu^{(k)}}(r)}{T}\right)$$

Which means that

$$\begin{aligned}\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)} &= (\hat{\mu}_T^{(k^\star)} - \mu^{(k^\star)}) - (\hat{\mu}_T^{(k)} - \mu^{(k)}) + (\mu^{(k^\star)} - \mu^{(k)}) \\ &\approx \mathcal{N}\left(\mu^{(k^\star)} - \mu^{(k)}, \frac{\mathrm{Var}_{r\sim\nu^{(k^\star)}}(r) + \mathrm{Var}_{r\sim\nu^{(k)}}(r)}{T}\right)\end{aligned}$$

Let $C_k := \mathrm{Var}_{r\sim\nu^{(k^\star)}}(r) + \mathrm{Var}_{r\sim\nu^{(k)}}(r)$ and suppose that $\mu^{(k^\star)} - \mu^{(k)} = \sqrt{C_k/T}$, then:

Oracle regret

We know by the CLT that:

$$\hat{\mu}_T^{(k)} - \mu^{(k)} \approx \mathcal{N}\left(0, \frac{\mathrm{Var}_{r\sim\nu^{(k)}}(r)}{T}\right)$$

Which means that

$$\begin{aligned}\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)} &= (\hat{\mu}_T^{(k^\star)} - \mu^{(k^\star)}) - (\hat{\mu}_T^{(k)} - \mu^{(k)}) + (\mu^{(k^\star)} - \mu^{(k)}) \\ &\approx \mathcal{N}\left(\mu^{(k^\star)} - \mu^{(k)}, \frac{\mathrm{Var}_{r\sim\nu^{(k^\star)}}(r) + \mathrm{Var}_{r\sim\nu^{(k)}}(r)}{T}\right)\end{aligned}$$

Let $C_k := \mathrm{Var}_{r\sim\nu^{(k^\star)}}(r) + \mathrm{Var}_{r\sim\nu^{(k)}}(r)$ and suppose that $\mu^{(k^\star)} - \mu^{(k)} = \sqrt{C_k/T}$, then:

$$\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) \approx \mathcal{N}(1,1)$$

Oracle regret (cont'd)

From previous slide: $\sqrt{\dfrac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) \approx \mathcal{N}(1,1)$

Oracle regret (cont'd)

From previous slide: $\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) \approx \mathcal{N}(1,1)$

$$\mathbb{P}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)} < 0) = \mathbb{P}\left(\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) < 0\right)$$

Oracle regret (cont'd)

From previous slide: $\sqrt{\dfrac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) \approx \mathcal{N}(1,1)$

$$\mathbb{P}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)} < 0) = \mathbb{P}\left(\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) < 0\right) \approx \mathbb{P}(\mathcal{N}(1,1) < 0)$$

Oracle regret (cont'd)

From previous slide: $\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) \approx \mathcal{N}(1,1)$

$$\mathbb{P}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)} < 0) = \mathbb{P}\left(\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) < 0\right) \approx \mathbb{P}(\mathcal{N}(1,1) < 0) \approx 16\,\%$$

Oracle regret (cont'd)

From previous slide: $\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) \approx \mathcal{N}(1,1)$

$$\mathbb{P}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)} < 0) = \mathbb{P}\left(\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) < 0\right) \approx \mathbb{P}(\mathcal{N}(1,1) < 0) \approx 16\,\%$$

So if $\mu^{(k^\star)} - \mu^{(k)} = \sqrt{C_k/T}$ for all $k \neq k^\star$, and if all $C_k = C$ for $k \neq k^\star$, then

Oracle regret (cont'd)

From previous slide: $\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) \approx \mathcal{N}(1,1)$

$$\mathbb{P}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)} < 0) = \mathbb{P}\left(\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) < 0\right) \approx \mathbb{P}(\mathcal{N}(1,1) < 0) \approx 16\,\%$$

So if $\mu^{(k^\star)} - \mu^{(k)} = \sqrt{C_k/T}$ for all $k \neq k^\star$, and if all $C_k = C$ for $k \neq k^\star$, then

$$\mathbb{P}(\hat{k}_T \neq k^\star) \gtrsim 16\,\%$$

Oracle regret (cont'd)

From previous slide: $\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) \approx \mathcal{N}(1,1)$

$$\mathbb{P}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)} < 0) = \mathbb{P}\left(\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) < 0\right) \approx \mathbb{P}(\mathcal{N}(1,1) < 0) \approx 16\,\%$$

So if $\mu^{(k^\star)} - \mu^{(k)} = \sqrt{C_k/T}$ for all $k \neq k^\star$, and if all $C_k = C$ for $k \neq k^\star$, then

$$\mathbb{P}(\hat{k}_T \neq k^\star) \gtrsim 16\,\%$$

$$\Rightarrow \mathbb{P}(\mu^{(k^\star)} - \mu^{(\hat{k}_T)} = \sqrt{C/T}) \gtrsim 16\,\%$$

Oracle regret (cont'd)

From previous slide: $\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) \approx \mathcal{N}(1,1)$

$$\mathbb{P}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)} < 0) = \mathbb{P}\left(\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) < 0\right) \approx \mathbb{P}(\mathcal{N}(1,1) < 0) \approx 16\,\%$$

So if $\mu^{(k^\star)} - \mu^{(k)} = \sqrt{C_k/T}$ for all $k \neq k^\star$, and if all $C_k = C$ for $k \neq k^\star$, then

$$\mathbb{P}(\hat{k}_T \neq k^\star) \gtrsim 16\,\%$$

$$\Rightarrow \mathbb{P}(\mu^{(k^\star)} - \mu^{(\hat{k}_T)} = \sqrt{C/T}) \gtrsim 16\,\%$$

$\text{Regret}_T = T(\mu^{(k^\star)} - \mu^{(\hat{k}_T)})$

Oracle regret (cont'd)

From previous slide: $\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) \approx \mathcal{N}(1,1)$

$$\mathbb{P}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)} < 0) = \mathbb{P}\left(\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) < 0\right) \approx \mathbb{P}(\mathcal{N}(1,1) < 0) \approx 16\,\%$$

So if $\mu^{(k^\star)} - \mu^{(k)} = \sqrt{C_k/T}$ for all $k \neq k^\star$, and if all $C_k = C$ for $k \neq k^\star$, then

$$\mathbb{P}(\hat{k}_T \neq k^\star) \gtrsim 16\,\%$$

$$\Rightarrow \mathbb{P}(\mu^{(k^\star)} - \mu^{(\hat{k}_T)} = \sqrt{C/T}) \gtrsim 16\,\%$$

$\text{Regret}_T = T(\mu^{(k^\star)} - \mu^{(\hat{k}_T)})$ $\qquad \Rightarrow \mathbb{P}(\text{Regret}_T = \sqrt{CT}) \gtrsim 16\,\%$

Oracle regret (cont'd)

From previous slide: $\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) \approx \mathcal{N}(1,1)$

$$\mathbb{P}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)} < 0) = \mathbb{P}\left(\sqrt{\frac{T}{C_k}}(\hat{\mu}_T^{(k^\star)} - \hat{\mu}_T^{(k)}) < 0\right) \approx \mathbb{P}(\mathcal{N}(1,1) < 0) \approx 16\,\%$$

So if $\mu^{(k^\star)} - \mu^{(k)} = \sqrt{C_k/T}$ for all $k \neq k^\star$, and if all $C_k = C$ for $k \neq k^\star$, then

$$\mathbb{P}(\hat{k}_T \neq k^\star) \gtrsim 16\,\%$$

$$\Rightarrow \mathbb{P}(\mu^{(k^\star)} - \mu^{(\hat{k}_T)} = \sqrt{C/T}) \gtrsim 16\,\%$$

$\text{Regret}_T = T(\mu^{(k^\star)} - \mu^{(\hat{k}_T)})$ $\quad \Rightarrow \mathbb{P}(\text{Regret}_T = \sqrt{CT}) \gtrsim 16\,\%$

$$\Rightarrow \text{Regret}_T = \Omega(\sqrt{T}) \text{ w/p } \geq 16\,\%$$

Today

- Feedback from last lecture
- Recap
- Regret *lower* bound
- Instance-dependent regret

Instance-dependent regret

So no algorithm can beat $\Omega(\sqrt{T})$

Instance-dependent regret

So no algorithm can beat $\Omega(\sqrt{T})$

But clearly there are situations when that's not true!

E.g., if $\nu^{(1)} = \cdots = \nu^{(K)}$, then $\text{Regret}_T = 0$ for all T for any algorithm

Instance-dependent regret

So no algorithm can beat $\Omega(\sqrt{T})$

But clearly there are situations when that's not true!

E.g., if $\nu^{(1)} = \cdots = \nu^{(K)}$, then $\text{Regret}_T = 0$ for all T for any algorithm

So is our lower-bound wrong?

Instance-dependent regret

So no algorithm can beat $\Omega(\sqrt{T})$

But clearly there are situations when that's not true!

E.g., if $\nu^{(1)} = \cdots = \nu^{(K)}$, then $\text{Regret}_T = 0$ for all T for any algorithm

So is our lower-bound wrong? Let's think about the argument we made…

Instance-dependent regret

So no algorithm can beat $\Omega(\sqrt{T})$

But clearly there are situations when that's not true!

E.g., if $\nu^{(1)} = \cdots = \nu^{(K)}$, then $\text{Regret}_T = 0$ for all T for any algorithm

So is our lower-bound wrong? Let's think about the argument we made…

Recall that we chose $\mu^{(k^\star)} - \mu^{(k)}$ <u>very carefully</u> (and in a T-dependent way)

Instance-dependent regret

So no algorithm can beat $\Omega(\sqrt{T})$

But clearly there are situations when that's not true!

E.g., if $\nu^{(1)} = \cdots = \nu^{(K)}$, then $\text{Regret}_T = 0$ for all T for any algorithm

So is our lower-bound wrong? Let's think about the argument we made…

Recall that we chose $\mu^{(k^\star)} - \mu^{(k)}$ very carefully (and in a T-dependent way)

Correctly inferred w/ choice that the best regret the oracle can guarantee is $\Omega(\sqrt{T})$

Instance-dependent regret

So no algorithm can beat $\Omega(\sqrt{T})$

But clearly there are situations when that's not true!

E.g., if $\nu^{(1)} = \cdots = \nu^{(K)}$, then $\text{Regret}_T = 0$ for all T for any algorithm

So is our lower-bound wrong? Let's think about the argument we made…

Recall that we chose $\mu^{(k^\star)} - \mu^{(k)}$ very carefully (and in a T-dependent way)

Correctly inferred w/ choice that the best regret the oracle can guarantee is $\Omega(\sqrt{T})$

But this is *worst-case*, i.e., it is the best the oracle can guarantee without knowing more about the environment (since our choice of $\mu^{(k^\star)} - \mu^{(k)}$ could be correct)

Instance-dependent regret

So no algorithm can beat $\Omega(\sqrt{T})$

But clearly there are situations when that's not true!

E.g., if $\nu^{(1)} = \cdots = \nu^{(K)}$, then $\text{Regret}_T = 0$ for all T for any algorithm

So is our lower-bound wrong? Let's think about the argument we made…

Recall that we chose $\mu^{(k^\star)} - \mu^{(k)}$ very carefully (and in a T-dependent way)

Correctly inferred w/ choice that the best regret the oracle can guarantee is $\Omega(\sqrt{T})$

But this is *worst-case*, i.e., it is the best the oracle can guarantee without knowing more about the environment (since our choice of $\mu^{(k^\star)} - \mu^{(k)}$ could be correct)

The oracle may do (much) better than this in a given problem instance!

Instance-dependent regret

So no algorithm can beat $\Omega(\sqrt{T})$

But clearly there are situations when that's not true!

E.g., if $\nu^{(1)} = \cdots = \nu^{(K)}$, then $\text{Regret}_T = 0$ for all T for any algorithm

So is our lower-bound wrong? Let's think about the argument we made…

Recall that we chose $\mu^{(k^\star)} - \mu^{(k)}$ very carefully (and in a T-dependent way)

Correctly inferred w/ choice that the best regret the oracle can guarantee is $\Omega(\sqrt{T})$

But this is *worst-case*, i.e., it is the best the oracle can guarantee without knowing more about the environment (since our choice of $\mu^{(k^\star)} - \mu^{(k)}$ could be correct)

The oracle may do (much) better than this in a given problem instance!

E.g., any algorithm's $\text{Regret}_T = 0$ if $\nu^{(1)} = \cdots = \nu^{(K)}$

Instance-dependent regret (cont'd)

When analyzing the properties of an algorithm, we may be interested in how well it performs in different problem instances, not just in the worst-case environment

Instance-dependent regret (cont'd)

When analyzing the properties of an algorithm, we may be interested in how well it performs in different problem instances, not just in the worst-case environment

Instance-dependent regret bounds incorporate information about the particular instance of a bandit environment into their bounds, reflecting the fact that a given algorithm's regret will depend on the instance

Instance-dependent regret (cont'd)

When analyzing the properties of an algorithm, we may be interested in how well it performs in different problem instances, not just in the worst-case environment

Instance-dependent regret bounds incorporate information about the particular instance of a bandit environment into their bounds, reflecting the fact that a given algorithm's regret will depend on the instance

Expect such bounds to be tighter, since they incorporate more information!

Instance-dependent regret (cont'd)

When analyzing the properties of an algorithm, we may be interested in how well it performs in different problem instances, not just in the worst-case environment

Instance-dependent regret bounds incorporate information about the particular instance of a bandit environment into their bounds, reflecting the fact that a given algorithm's regret will depend on the instance

Expect such bounds to be tighter, since they incorporate more information!

Example: pure exploration (if T divides K and deterministically cycle through arms)

Instance-dependent regret (cont'd)

When analyzing the properties of an algorithm, we may be interested in how well it performs in different problem instances, not just in the worst-case environment

Instance-dependent regret bounds incorporate information about the particular instance of a bandit environment into their bounds, reflecting the fact that a given algorithm's regret will depend on the instance

Expect such bounds to be tighter, since they incorporate more information!

Example: pure exploration (if T divides K and deterministically cycle through arms)
Our regret bound started out instance-dependent: $\text{Regret}_T = T(\mu^\star - \bar{\mu})$, since it depends on the $\mu^{(k)}$'s, which depend on the instance.

Instance-dependent regret (cont'd)

When analyzing the properties of an algorithm, we may be interested in how well it performs in different problem instances, not just in the worst-case environment

Instance-dependent regret bounds incorporate information about the particular instance of a bandit environment into their bounds, reflecting the fact that a given algorithm's regret will depend on the instance

Expect such bounds to be tighter, since they incorporate more information!

Example: pure exploration (if T divides K and deterministically cycle through arms)
Our regret bound started out instance-dependent: $\text{Regret}_T = T(\mu^\star - \bar{\mu})$, since it depends on the $\mu^{(k)}$'s, which depend on the instance.

We used it to derive (looser) worst-case bound: $\text{Regret}_T \leq T$

Instance-dependent regret (cont'd)

When analyzing the properties of an algorithm, we may be interested in how well it performs in different problem instances, not just in the worst-case environment

Instance-dependent regret bounds incorporate information about the particular instance of a bandit environment into their bounds, reflecting the fact that a given algorithm's regret will depend on the instance

Expect such bounds to be tighter, since they incorporate more information!

<u>Example</u>: pure exploration (if T divides K and deterministically cycle through arms)

Our regret bound started out <u>instance-dependent</u>: $\text{Regret}_T = T(\mu^\star - \bar{\mu})$, since it depends on the $\mu^{(k)}$'s, which depend on the instance.

No dependence on instance!

We used it to derive (looser) <u>worst-case</u> bound: $\text{Regret}_T \leq T$

Instance-dependent regret for UCB: strategy

Instance-dependent regret for UCB: strategy

1. Now that we can incorporate information about the $\mu^{(k)}$, we'll try to precisely bound how often each suboptimal arm k is sampled, $N_T^{(k)}$

Instance-dependent regret for UCB: strategy

1. Now that we can incorporate information about the $\mu^{(k)}$, we'll try to precisely bound how often each suboptimal arm k is sampled, $N_T^{(k)}$
2. To do that, we'll use the uniform Hoeffding bound to see how often the UCB for $k^\star$ is guaranteed (with high probability) to be higher than the UCB for k

Instance-dependent regret for UCB: strategy

1. Now that we can incorporate information about the $\mu^{(k)}$, we'll try to precisely bound how often each suboptimal arm k is sampled, $N_T^{(k)}$
2. To do that, we'll use the uniform Hoeffding bound to see how often the UCB for $k^\star$ is guaranteed (with high probability) to be higher than the UCB for k
3. Then we'll multiply $N_T^{(k)}$ by the suboptimality of arm k, and sum this over the arms k to get the total regret

Today’s summary:

Today's summary:

Regret lower bound

- No algorithm can do better than $\Omega(\sqrt{T})$
- Algorithms like UCB achieve same worst-case regret as an oracle

Today's summary:

Regret lower bound

- No algorithm can do better than $\Omega(\sqrt{T})$
- Algorithms like UCB achieve same worst-case regret as an oracle

Instance-dependent regret

- Characterizes regret in terms of true arm means
- More descriptive than worst-case analysis

Today's summary:

Regret lower bound

- No algorithm can do better than $\Omega(\sqrt{T})$
- Algorithms like UCB achieve same worst-case regret as an oracle

Instance-dependent regret

- Characterizes regret in terms of true arm means
- More descriptive than worst-case analysis

Next time:

- Bayesian Bandit
- Thompson sampling

Today's summary:

Regret lower bound

- No algorithm can do better than $\Omega(\sqrt{T})$
- Algorithms like UCB achieve same worst-case regret as an oracle

Instance-dependent regret

- Characterizes regret in terms of true arm means
- More descriptive than worst-case analysis

Next time:

- Bayesian Bandit
- Thompson sampling

1-minute feedback form: https://bit.ly/3RHtlxy