TTIC 31250
An Introduction to the Theory of Machine Learning

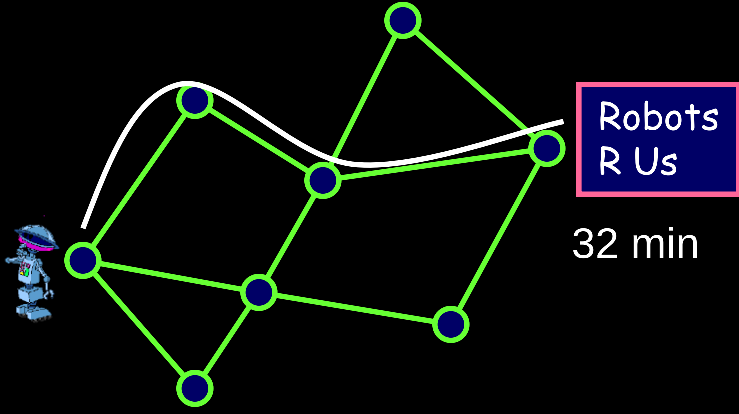
The Adversarial Multi-armed Bandit Problem

Avrim Blum

Start with recap

Consider the following setting…

- Each morning, you need to pick one of N possible routes to drive to work.
- But traffic is different each day.
  - Not clear a priori which will be best.
  - When you get there you find out how long your route took. (And maybe others too or maybe not.)

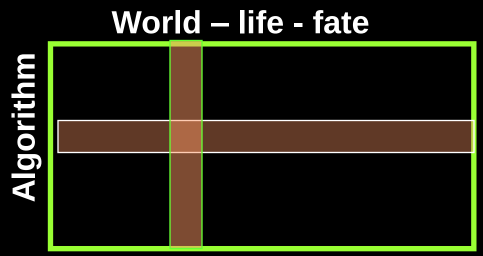


- Want a strategy for picking routes so that in the long run, whatever the sequence of traffic patterns has been, you've done nearly as well as the best fixed route in hindsight. (In expectation, over internal randomness in the algorithm)

"No-regret" algorithms for repeated decisions

General framework:

- Algorithm has N options. World chooses cost vector. Can view as matrix like this (maybe infinite # cols)



- At each time step, algorithm picks row, life picks column.
  - Alg pays cost for action chosen.
  - Alg gets column as feedback (or just its own cost in the "bandit" model).
  - Need to assume some bound on max cost. Let's say all costs between 0 and 1.

"No-regret" algorithms for repeated decisions

Define **average regret** in T time steps as:

(avg per-day cost of alg) – (avg per-day cost of best fixed row in hindsight).

We want this to go to 0 or better as T gets large.

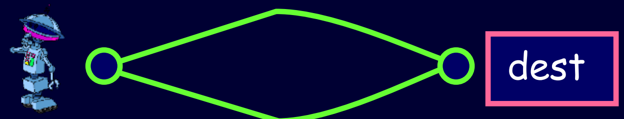
[called a "no-regret" algorithm]

- At each time step, algorithm picks row, life picks column.
  - Alg pays cost for action chosen.
  - Alg gets column as feedback (or just its own cost in the "bandit" model).
  - Need to assume some bound on max cost. Let's say all costs between 0 and 1.

History and development (abridged)

- [Hannan'57, Blackwell'56]: Alg. with regret $O((N/T)^{1/2})$.
  - Re-phrasing, need only $T = O(N/\varepsilon^2)$ steps to get time-average regret down to ε. (will call this quantity T_ε)
  - Optimal dependence on T (or ε). Game-theorists viewed #rows N as constant, not so important as T, so pretty much done.

Why optimal in T?



	World – life - fate	
Algorithm	1	0
	0	1

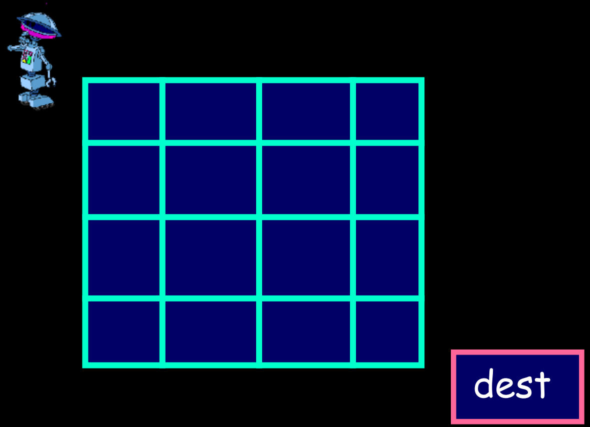
- Say world flips fair coin each day.
- Any alg, in T days, has expected cost T/2.
- But E[min(# heads,#tails)] = $T/2 - O(T^{1/2})$.
- So, per-day gap is $O(1/T^{1/2})$.

History and development (abridged)

- [Hannan'57, Blackwell'56]: Alg. with regret $O((N/T)^{1/2})$.
  - Re-phrasing, need only $T = O(N/\varepsilon^2)$ steps to get time-average regret down to ε. (will call this quantity T_ε)
  - Optimal dependence on T (or ε). Game-theorists viewed #rows N as constant, not so important as T, so pretty much done.
- Learning-theory 80s-90s: "combining expert advice". Imagine large class C of N prediction rules.
  - Perform (nearly) as well as best $f \in C$.
  - [LittlestoneWarmuth'89]: Randomized WM algorithm
    - $E[\text{cost}] \leq OPT(1+\varepsilon) + (\log N)/\varepsilon$.
    - Regret $O((\log N)/T)^{1/2}$. $T_\varepsilon = O((\log N)/\varepsilon^2)$.
  - Optimal as fn of N too, plus lots of work on exact constants, 2nd order terms, etc. [CFHHSW93]…
- Extensions to bandit model (adds extra factor of N).

Efficient implicit implementation for large N…

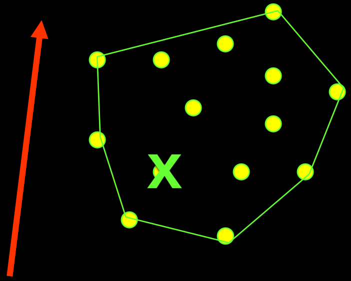
- Bounds have only log dependence on # choices N.
- So, conceivably can do well when N is exponential in natural problem size, if only could implement efficiently.
- E.g., case of paths…



[Kalai-Vempala'03] and [Zinkevich'03] settings

[KV] "online linear optimization" setting:

- Implicit set S of feasible points in R^m. (E.g., m=#edges, S={indicator vectors 011010010 for possible paths})
- Assume have oracle for offline problem: given vector c, find $x \in S$ to minimize $c \cdot x$. (E.g., shortest path algorithm)
- Use to solve online problem: on day t, must pick $x_t \in S$ before c_t is given.
- $(c_1 \cdot x_1 + \ldots + c_T \cdot x_T)/T \rightarrow \min_{x \in S} x \cdot (c_1 + \ldots + c_T)/T$.



[Z] "online convex optimization" setting:

- Assume S is convex.
- Allow c(x) to be a convex function over S.
- Assume given any x' not in S, can algorithmically find nearest $x \in S$.

Plan for today

- What if we only get feedback for the action we choose? (called the "multi-armed bandit" setting)

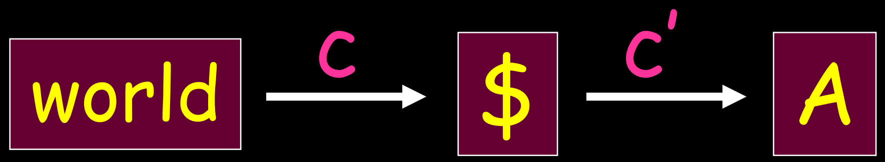
- But first, a quick discussion of [0,1] vs {0,1} costs for RWM algorithm

[0,1] costs vs {0,1} costs.

We analyzed Randomized Wtd Majority for case that all costs in {0,1} (and slightly hand-waved extension to [0,1])

Here is an alternative simple way to extend to [0,1].

- Given cost vector c, view c_i as bias of coin. Flip to create boolean vector c', s.t. $E[c'_i] = c_i$. Feed c' to alg A.



Cost' = cost on c' vectors

- For any sequence of vectors c', we have:
  - $E_A[\text{cost}'(A)] \leq \min_i \text{cost}'(i)$ + [regret term]
- So, $E_\$[E_A[\text{cost}'(A)]] \leq E_\$[\min_i \text{cost}'(i)]$ + [regret term]
- LHS is $E_A[\text{cost}(A)]$. (since A picks weights before seeing costs)
- RHS $\leq \min_i E_\$[\text{cost}'(i)]$ + [r.t.] = $\min_i[\text{cost}(i)]$ + [r.t.]

In other words, costs between 0 and 1 just make the problem easier…

Experts $\rightarrow$ Bandit setting

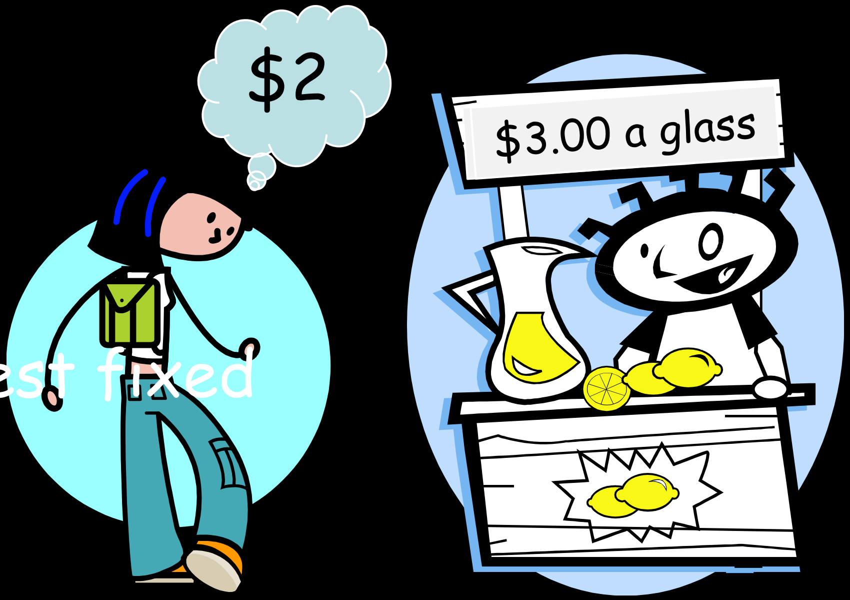
- In the bandit setting, only get feedback for the action we choose. Still want to compete with best action in hindsight.
- [ACFS02] give algorithm with cumulative regret $O((TN \log N)^{1/2})$. [average regret $O(((N \log N)/T)^{1/2})$.]
- Will do a somewhat weaker version of their analysis (same algorithm but not as tight a bound).
- Talk about it in the context of online pricing…

Online pricing

- Say you are selling lemonade (or bottles of water outside a football stadium).
- For t=1,2,…T
  - Seller sets price p^t
  - Buyer arrives with valuation v^t
  - If $v^t \geq p^t$, buyer purchases and pays p^t, else doesn't.
  - Repeat.

View each possible price as a different row/expert

- Assume all valuations $\leq h$.
- Goal: do nearly as well as best fixed price in hindsight.

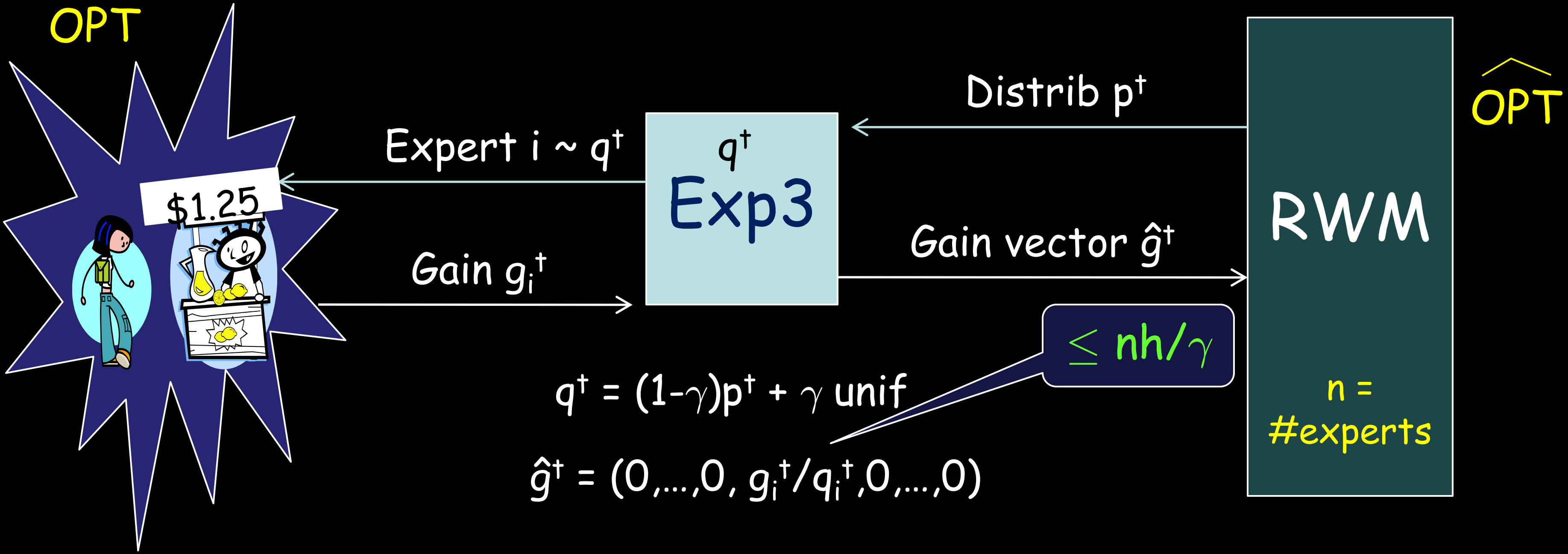


- If v^t revealed, run RWM. E[gain] $\geq$ OPT$(1-\epsilon)$ - $O(\epsilon^{-1}$ h log n).

Multi-armed bandit problem

Exponential Weights for Exploration and Exploitation (exp^3)

[Auer,Cesa-Bianchi,Freund,Schapire]



OPT

$1.25

Expert i ~ q^t

Gain g_i^t

q^t

Exp3

Distrib p^t

Gain vector $\hat{g}^t$

RWM

$\widehat{OPT}$

$\leq nh/\gamma$

n = #experts

$q^t = (1-\gamma)p^t + \gamma$ unif

$\hat{g}^t = (0,\ldots,0, g_i^t/q_i^t,0,\ldots,0)$

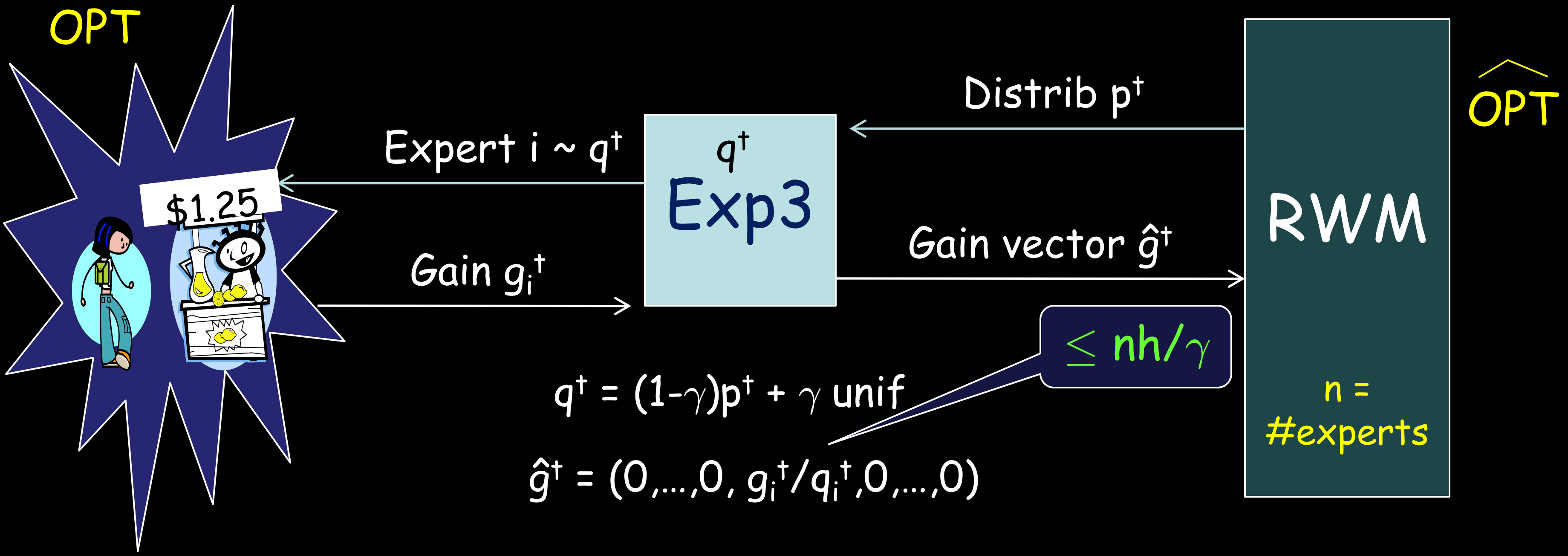
1. RWM believes gain is: $p^t \cdot \hat{g}^t = p_i^t(g_i^t/q_i^t) \equiv g^t_{RWM}$
2. $\sum_t g^t_{RWM} \geq \widehat{OPT}\,(1-\epsilon) - O(\epsilon^{-1}\, nh/\gamma \log n)$
3. Actual gain is: $g_i^t = g^t_{RWM}\,(q_i^t/p_i^t) \geq g^t_{RWM}(1-\gamma)$
4. $E[\widehat{OPT}] \geq OPT$. Because $E[\hat{g}_j^t] = (1- q_j^t)0 + q_j^t(g_j^t/q_j^t) = g_j^t$,

   so $E[\max_j[\sum_t \hat{g}_j^t]] \geq \max_j [\, E[\sum_t \hat{g}_j^t]\,] = OPT$.

Multi-armed bandit problem

Exponential Weights for Exploration and Exploitation (exp^3)

[Auer,Cesa-Bianchi,Freund,Schapire]



OPT

Expert i ~ q^t

Gain g_i^t

q^t

Exp3

Distrib p^t

Gain vector $\hat{g}^t$

RWM

$\widehat{\text{OPT}}$

$\leq nh/\gamma$

n = #experts

$q^t = (1-\gamma)p^t + \gamma$ unif

$\hat{g}^t = (0,\dots,0, g_i^t/q_i^t,0,\dots,0)$

Conclusion ($\gamma = \epsilon$):

$$E[\text{Exp3}] \geq \text{OPT}(1-\epsilon)^2 - O(\epsilon^{-2}\, nh \log(n))$$

Balancing would give $O((\text{OPT}\ nh \log n)^{2/3})$ regret because of ϵ^{-2}. But can reduce to ϵ^{-1} and $O((\text{OPT}\ nh \log n)^{1/2})$ with better analysis.

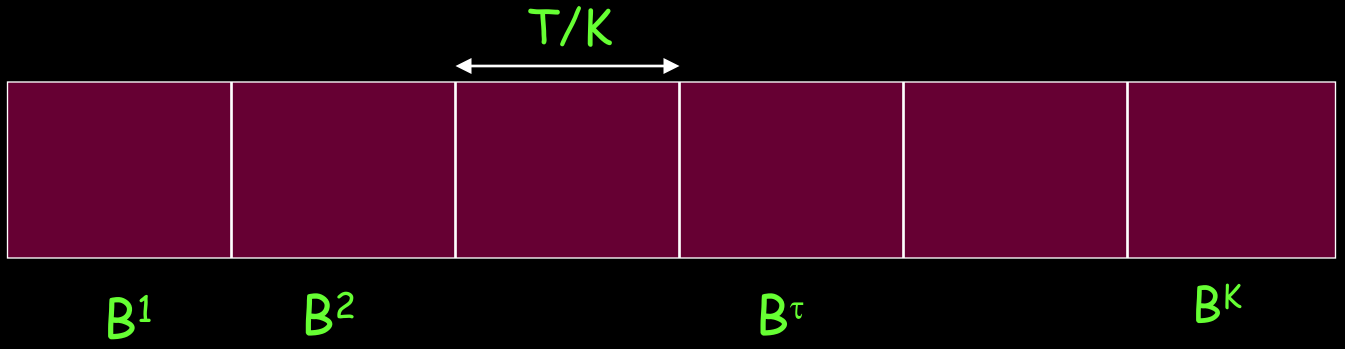
Another reduction (not as good but more generic)

Given: algorithm A for full-info setting with regret $\leq$ R(T).

Goal: use in black-box manner for bandit problem.

Preliminaries:

- First, suppose we break our T time steps into K blocks of size T/K each.



- Use same distrib throughout block and update based on average cost vector c^τ for block τ.
- Then, will get regret $\leq$ R(K) T/K.

Because really paying T/K c^τ per block

- What if we instead update on cost vector $c' \in [0,1]^N$ that's a random variable whose expectation is correct?

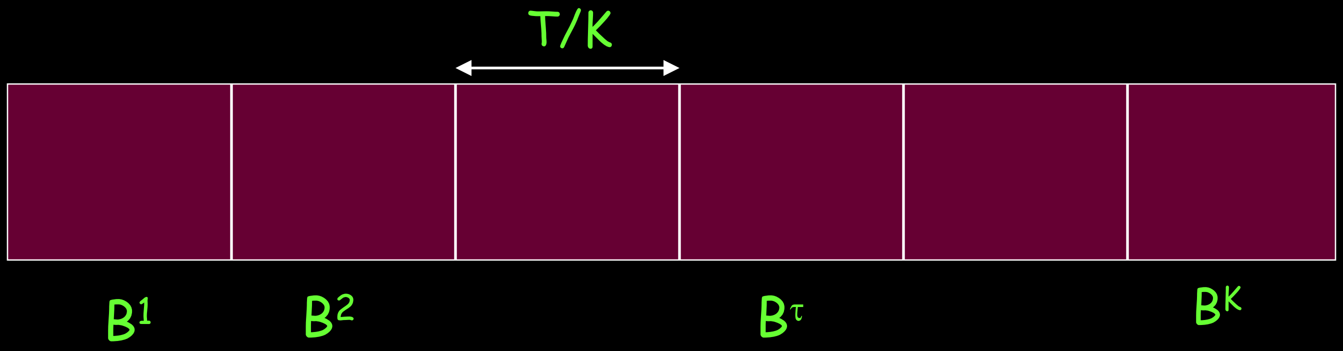
Another reduction (not as good but more generic)

Given: algorithm A for full-info setting with regret $\leq$ R(T).

Goal: use in black-box manner for bandit problem.

Preliminaries:

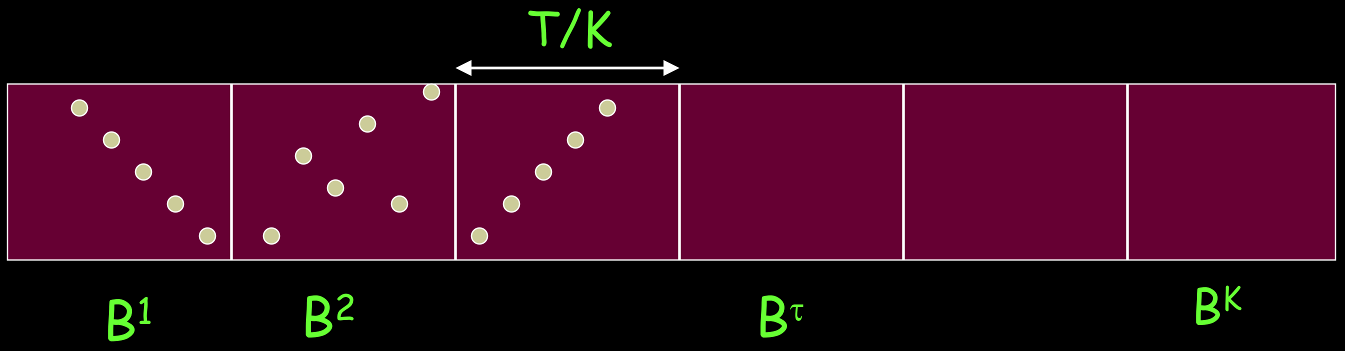
- First, suppose we break our T time steps into K blocks of size T/K each.



- Do at least as well by $\{0,1\}\rightarrow[0,1]$ argument. Still get regret bound R(K) T/K.
- How does this help us for bandit problem?
- What if we instead update on cost vector $c' \in [0,1]^N$ that's a random variable whose expectation is correct?

Experts → Bandit setting

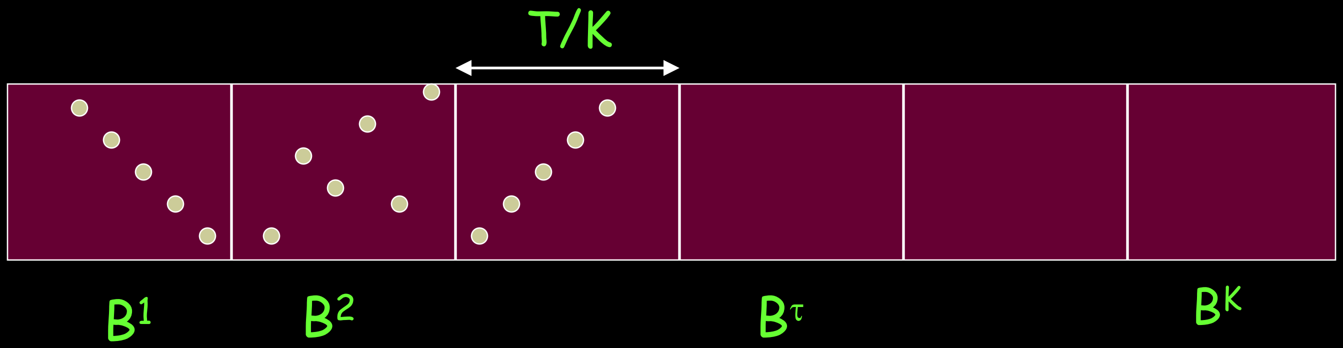
- For bandit problem, for each action, pick random time step in each block to try it as "exploration".
- Define c′ only wrt these exploration steps.
- Just have to pay an extra at most NK for cost of this exploration.



- Do at least as well by {0,1}→[0,1] argument. Still get regret bound R(K) T/K.
- How does this help us for bandit problem?
- What if we instead update on cost vector $c' \in [0,1]^N$ that's a random variable whose expectation is correct?

Experts → Bandit setting

- For bandit problem, for each action, pick random time step in each block to try it as "exploration".
- Define c' only wrt these exploration steps.
- Just have to pay an extra at most NK for cost of this exploration.



- Final bound: R(K) T/K + NK.
- Using $K = (T/N)^{2/3}$ and bound from RWM, get cumulative regret bound of $O(T^{2/3}N^{1/3} \log N)$.

Summary

Algorithms for online decision-making with strong guarantees on performance compared to best fixed choice.

- Application: play repeated game against adversary. Perform nearly as well as fixed strategy in hindsight.

Can apply even with very limited feedback.

- Application: which way to drive to work, with only feedback about your own paths; online pricing, even if only have buy/no buy feedback.