CS3243: Introduction to Artificial Intelligence

Instructor: Yair Zick

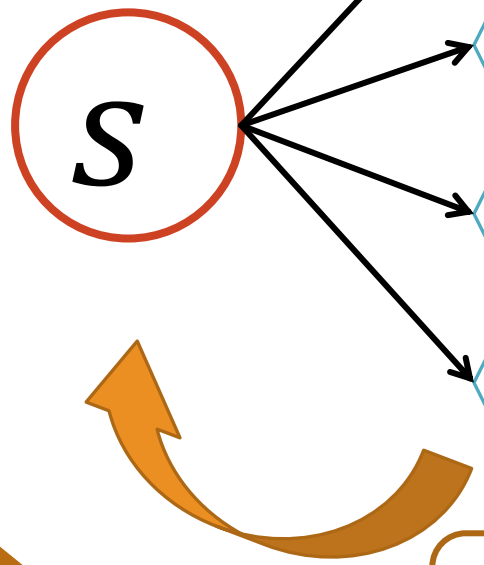
2020

Regret Minimization

"Hoping for the best but expecting the worst"

- Alphaville, *Forever Young*

Choosing the Best Action at a State



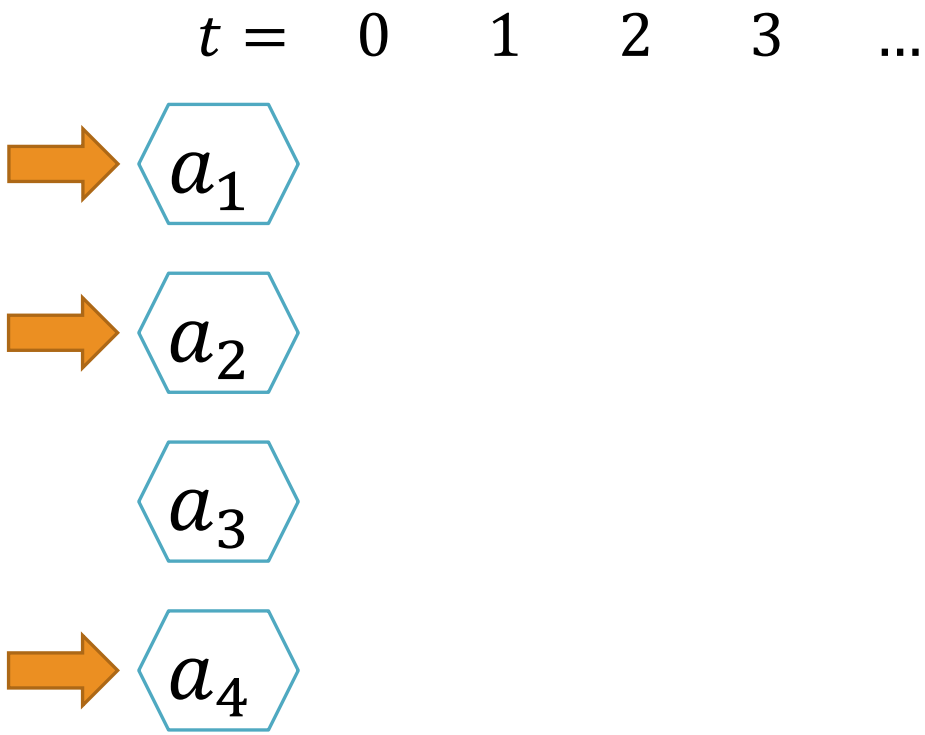
$t =$	0	1	2	3	…
a_1	0	0	1	1	…
a_2	0	1	0	0	…
a_3	0	1	1	1	…
a_4	1	1	0	1	…

We assume $0/1$ rewards

Reward depends on:
- Current state s
- Action taken
- The environment
- A terrible person who **wants** us to lose!

For simplicity, single-state domain. All actions return to s.

Multi-Armed Bandit Problem



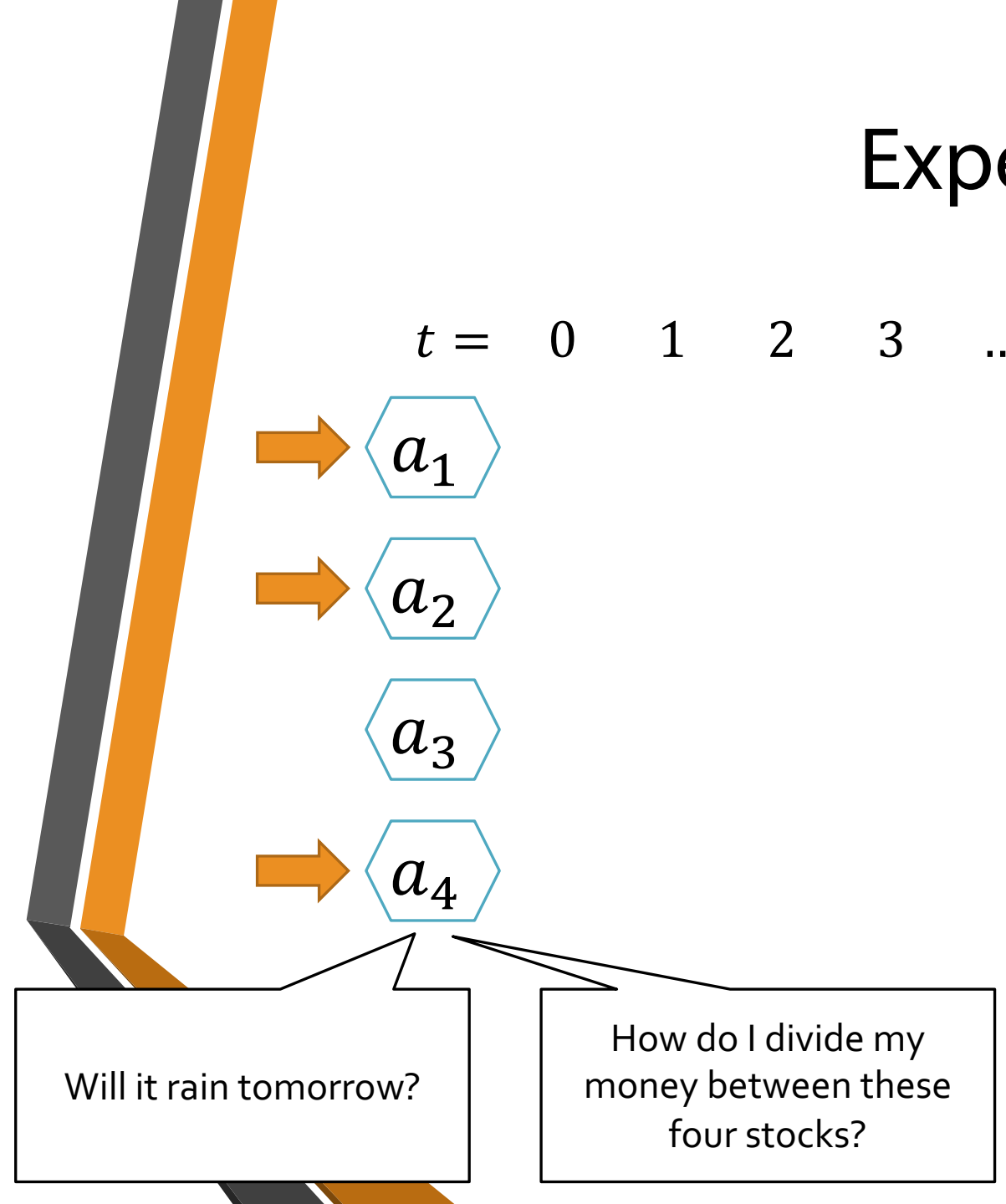
Classic Multi-Armed Bandits (MAB):

- Single state, take actions (that return to the same state)
- Actions offer $0/1$ rewards
- Only observe reward from action chosen, not others.

Goal

Find optimal policy/best action

Expert Systems



Classic Expert Systems:

- Single state, take actions (that return to the same state)
- Actions offer $0/1$ rewards
- Observe rewards from all actions, irrespective of what we picked.

Goal

Find optimal policy

Regret

We want to do well – benchmark against something!

1. Do at least as well as the best algorithm?
2. Do at least as well as the best action?

Definitions

- A policy π assigns a probability p_i^t of playing action i at time t.
- Value of π is $v_\pi^t = \sum_i p_i^t v_i^t$ - the expected reward at time t.
- $V_\pi^t = \sum_{\tau=1}^{t} v_\pi^\tau$: total reward up to time t.

Regret – Compare to Best Action

- Average reward of π vs. average reward of best action: $\frac{1}{\mathrm{T}} \times (V_{best}^t - V_\pi^t)$

Round 1 – Greedy algorithm

- At round t, let S_t be the set of actions that maximize total reward up to time t.
- Pick one of the best actions (choose the one with the **lowest index** if $|S_t| \geq 2$).

… or any other **deterministic** tie-breaking scheme. Tie breaking scheme can even depend on the history.




? ? ? ? $V_1^{t-1} = 0$

? ? ? ? $V_2^{t-1} = 0$

? ? ? ? $V_3^{t-1} = 0$

? ? ? ? $V_4^{t-1} = 0$

$$V_{Greedy}^{0} = 0$$

$$V_{best}^{0} = 0$$

0	?	?	?	$V_1^{t-1} = 0$
1	?	?	?	$V_2^{t-1} = 0$
0	?	?	?	$V_3^{t-1} = 0$
1	?	?	?	$V_4^{t-1} = 0$

$$V_{Greedy}^{1} = 0$$

$$V_{best}^{1} = 1$$

0 ? ? ? $V_1^{t-1} = 0$

1 ? ? ? $V_2^{t-1} = 1$

0 ? ? ? $V_3^{t-1} = 0$

1 ? ? ? $V_4^{t-1} = 1$

$$V_{Greedy}^{1} = 0$$

$$V_{best}^{1} = 1$$

0	1	?	?	$V_1^{t-1} = 0$
1	0	?	?	$V_2^{t-1} = 1$
0	1	?	?	$V_3^{t-1} = 0$
1	1	?	?	$V_4^{t-1} = 1$

$$V_{Greedy}^{2} = 0$$

$$V_{best}^{2} = 2$$

0 1 ? ? $V_1^{t-1} = 1$

1 0 ? ? $V_2^{t-1} = 1$

0 1 ? ? $V_3^{t-1} = 1$

1 1 ? ? $V_4^{t-1} = 2$

$$V_{Greedy}^{2} = 0$$

$$V_{best}^{2} = 2$$

0	1	1	?	$V_1^{t-1} = 1$
1	0	0	?	$V_2^{t-1} = 1$
0	1	1	?	$V_3^{t-1} = 1$
1	1	0	?	$V_4^{t-1} = 2$

$$V_{Greedy}^{3} = 0$$

$$V_{best}^{3} = 2$$

0 1 1 ? $V_1^{t-1} = 2$

1 0 0 ? $V_2^{t-1} = 1$

0 1 1 ? $V_3^{t-1} = 2$

1 1 0 ? $V_4^{t-1} = 2$

$$V_{Greedy}^{3} = 0$$

$$V_{best}^{3} = 2$$

0 1 1 0 $V_1^{t-1} = 2$

1 0 0 1 $V_2^{t-1} = 1$

0 1 1 0 $V_3^{t-1} = 2$

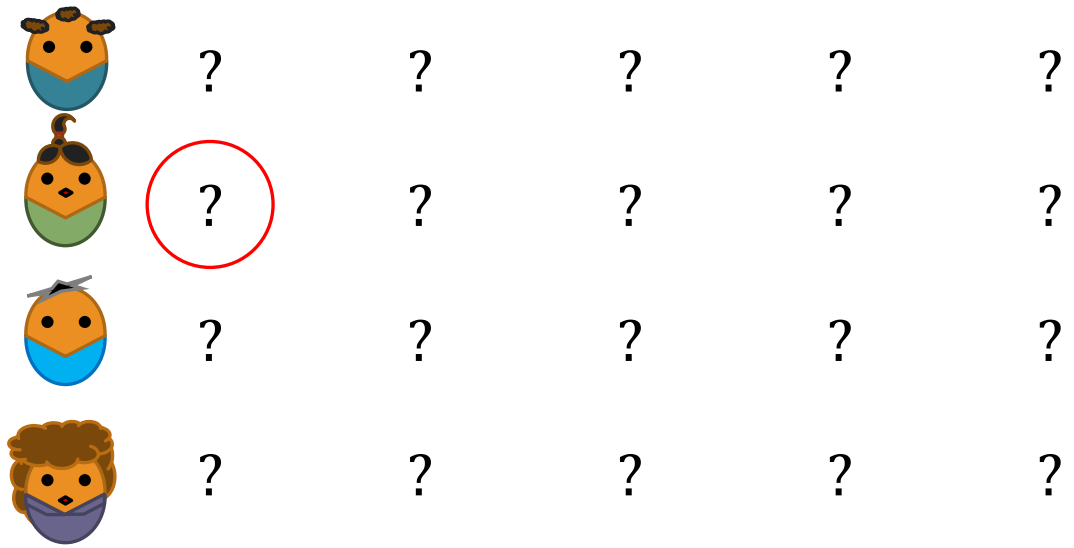
1 1 0 1 $V_4^{t-1} = 3$

$$V_{Greedy}^4 = 0$$

$$V_{best}^4 = 3$$

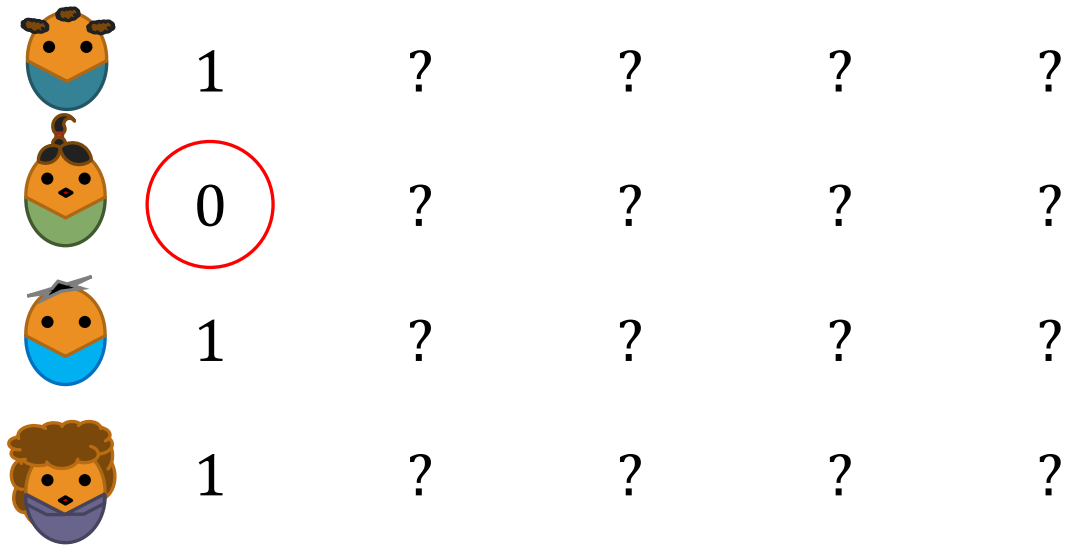
Theorem:
for any **deterministic** algorithm $\mathcal{A}$, there is a sequence of actions for which $V_{best}^{T} \geq \left\lceil \frac{(n-1)\times T}{n} \right\rceil$ but $V_{\mathcal{A}}^{T} = 0$.

Proof:



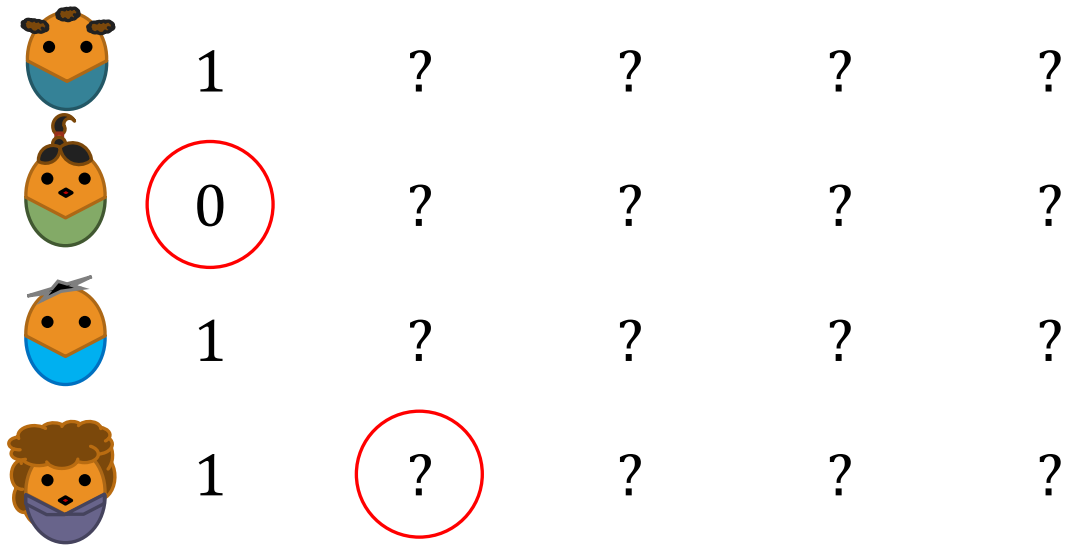
Theorem:
for any **deterministic** algorithm $\mathcal{A}$, there is a sequence of actions for which $V_{best}^{T} \geq \left\lceil \frac{(n-1)\times T}{n} \right\rceil$ but $V_{\mathcal{A}}^{T} = 0$.

Proof:



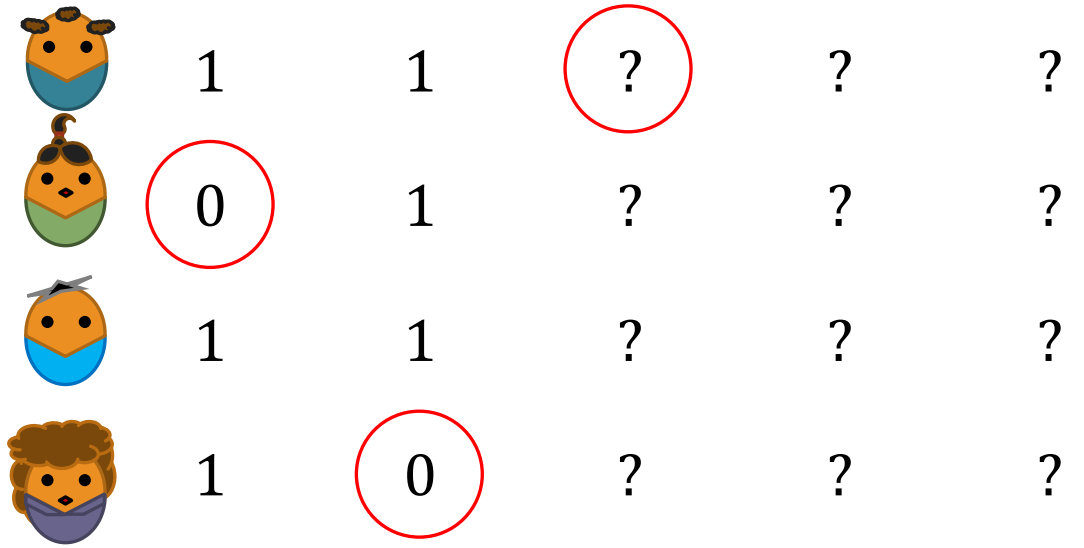
Theorem:
for any **deterministic** algorithm $\mathcal{A}$, there is a sequence of actions for which $V_{best}^{T} \geq \left\lceil \frac{(n-1)\times T}{n} \right\rceil$ but $V_{\mathcal{A}}^{T} = 0$.

Proof:



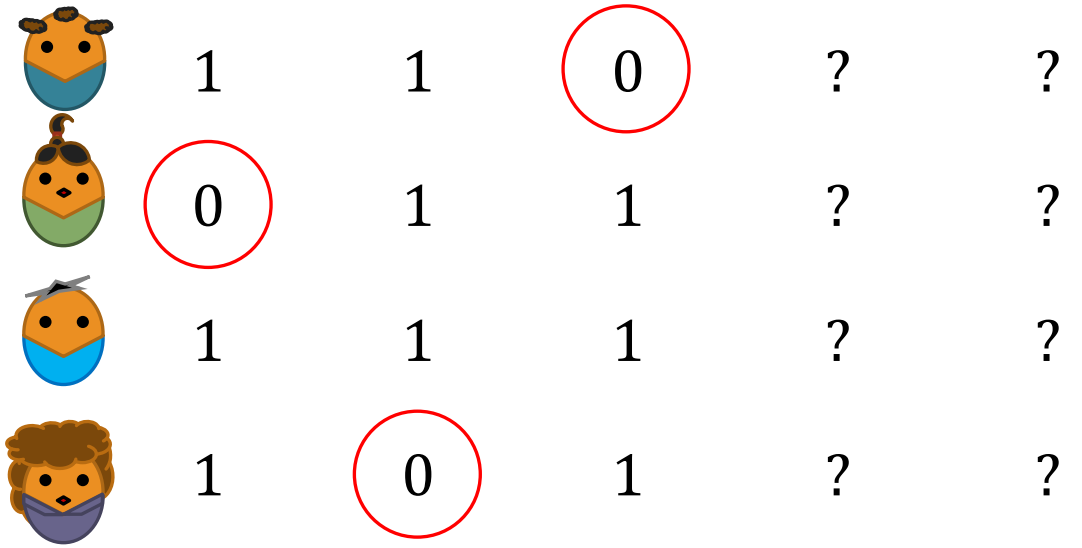
Theorem:
for any **deterministic** algorithm $\mathcal{A}$, there is a sequence of actions for which $V_{best}^{T} \geq \left\lceil \frac{(n-1)\times T}{n} \right\rceil$ but $V_{\mathcal{A}}^{T} = 0$.

Proof:



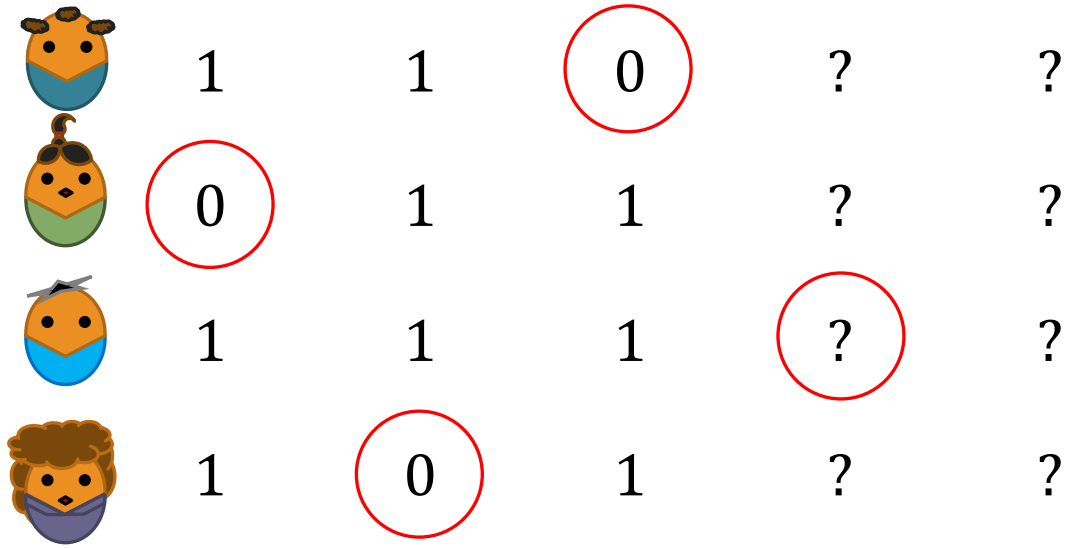
Theorem:
for any **deterministic** algorithm $\mathcal{A}$, there is a sequence of actions for which $V_{best}^{T} \geq \left\lceil \frac{(n-1)\times T}{n} \right\rceil$ but $V_{\mathcal{A}}^{T} = 0$.

Proof:



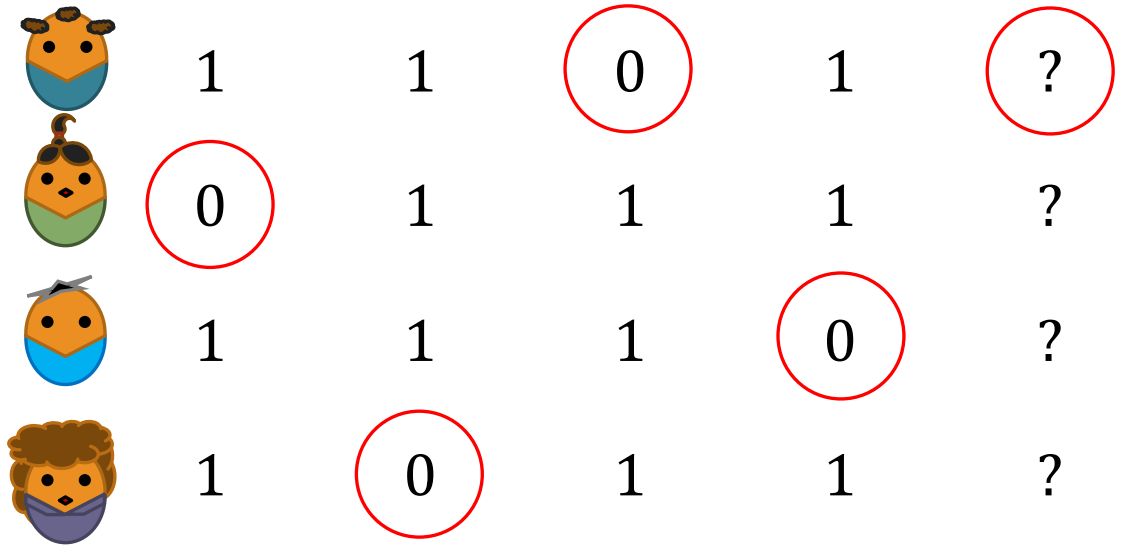
Theorem:
for any **deterministic** algorithm $\mathcal{A}$, there is a sequence of actions for which $V_{best}^{T} \geq \left\lceil \frac{(n-1)\times T}{n} \right\rceil$ but $V_{\mathcal{A}}^{T} = 0$.

Proof:



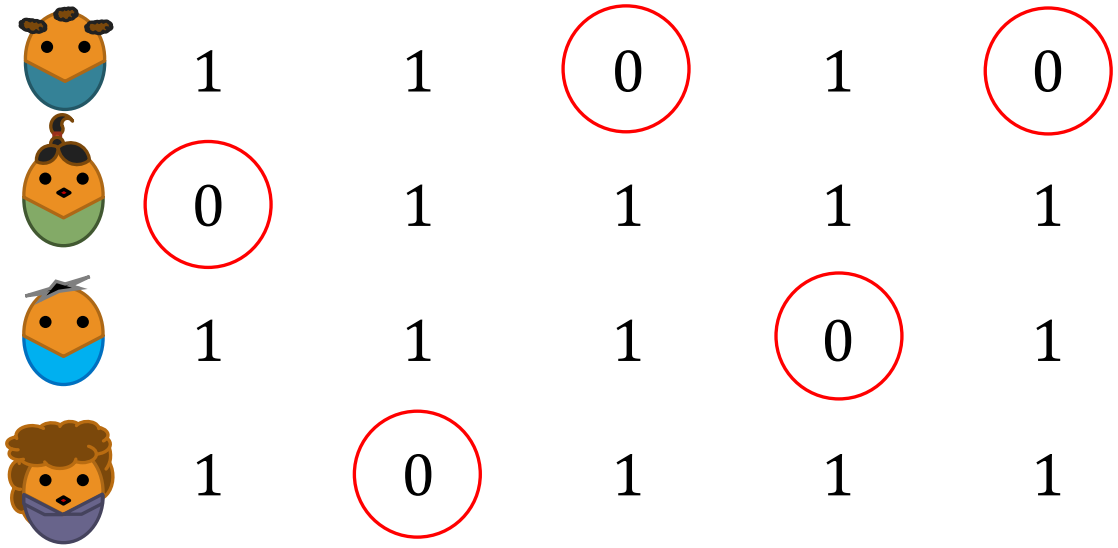
Theorem:
for any **deterministic** algorithm $\mathcal{A}$, there is a sequence of actions for which $V_{best}^{T} \geq \left\lceil \frac{(n-1)\times T}{n} \right\rceil$ but $V_{\mathcal{A}}^{T} = 0$.

Proof:



Theorem:
for any **deterministic** algorithm $\mathcal{A}$, there is a sequence of actions for which $V_{best}^{T} \geq \left\lceil \frac{(n-1)\times T}{n} \right\rceil$ but $V_{\mathcal{A}}^{T} = 0$.

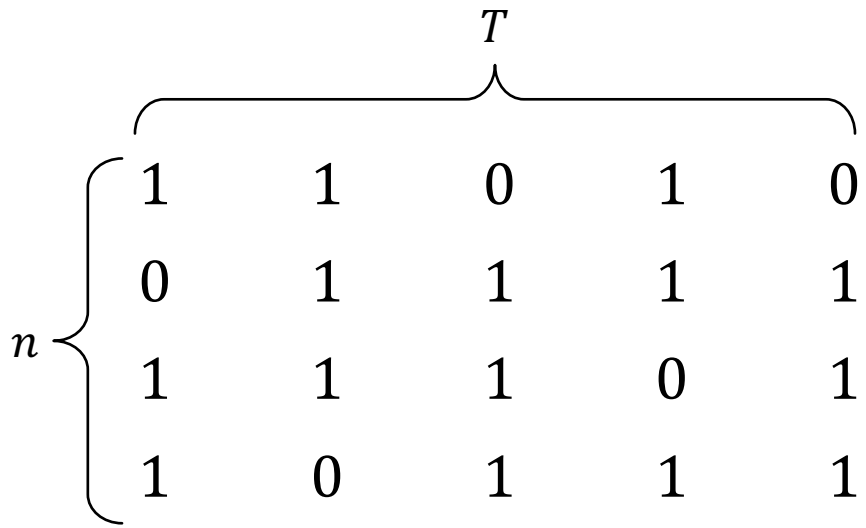
Proof:



Theorem:
for any **deterministic** algorithm $\mathcal{A}$, there is a sequence of actions for which $V_{best}^{T} \geq \left\lceil \frac{(n-1)\times T}{n} \right\rceil$ but $V_{\mathcal{A}}^{T} = 0$.

We constructed an $n \times T$ matrix, with one 0 in each column.

Average number of 1s in each row?



Theorem:
for any **deterministic** algorithm $\mathcal{A}$, there is a sequence of actions for which $V_{best}^{T} \geq \left\lceil \frac{(n-1)\times T}{n} \right\rceil$ but $V_{\mathcal{A}}^{T} = 0$.

We constructed **adversarial input**. Best action is "almost perfect".

Fully predictable algorithms can be exploited by adversaries!

Round 2 – Randomized Greedy

Initialize $V_i^0 = 0$ for all $i \in N$;

At time $t = 1, \ldots, \infty$:

$V_{\max}^t \leftarrow \max_i V_i^{t-1}$;

$S_t \leftarrow \{i \in N : V_i^{t-1} = V_{\min}^t\}$;

For $i \in N$ **do:**

If $i \in S_t$

$p_i^t \leftarrow \frac{1}{|S_t|}$;

Else

$p_i^t \leftarrow 0$;

End If

End For

Compute set of best actions S_t

Pick one of the best actions with prob. $\frac{1}{|S_t|}$

The rest are never picked

Round 2 – Randomized Greedy

The RG algorithm can lose at most $\log n + 1$ times (in expectation) for every loss of the best action.

Key Insight 1: when some actions from the best actions S_t lose, we remove them from the set of best actions.

Key Insight 2: losing over time is worse than losing at once.

"It is the slow knife… that cuts the deepest"

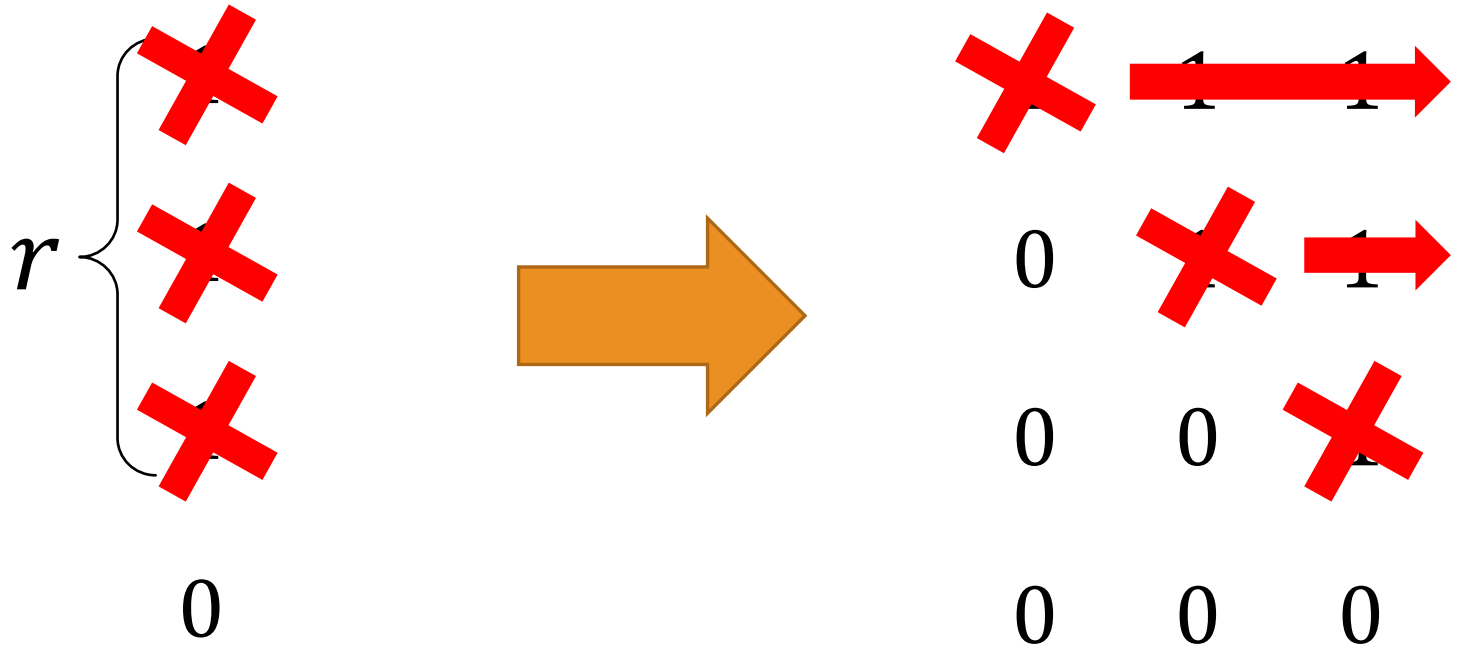
- Talia al-Ghul, *The Dark Knight Rises*

Round 2 – Randomized Greedy

The RG algorithm can lose at most $\log n + 1$ times (in expectation) for every loss of the best action.

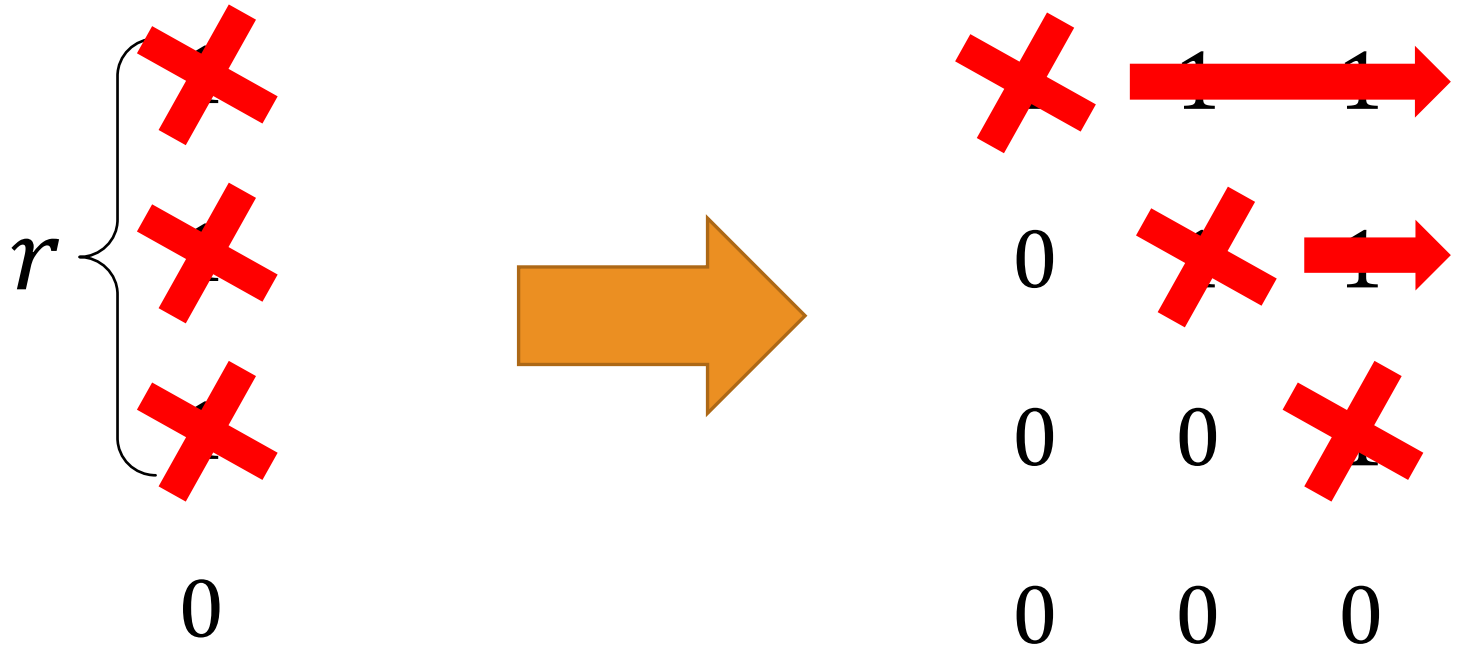
For the next few slides – losses instead of rewards: having a value of 1 is bad!

Round 2 – Randomized Greedy



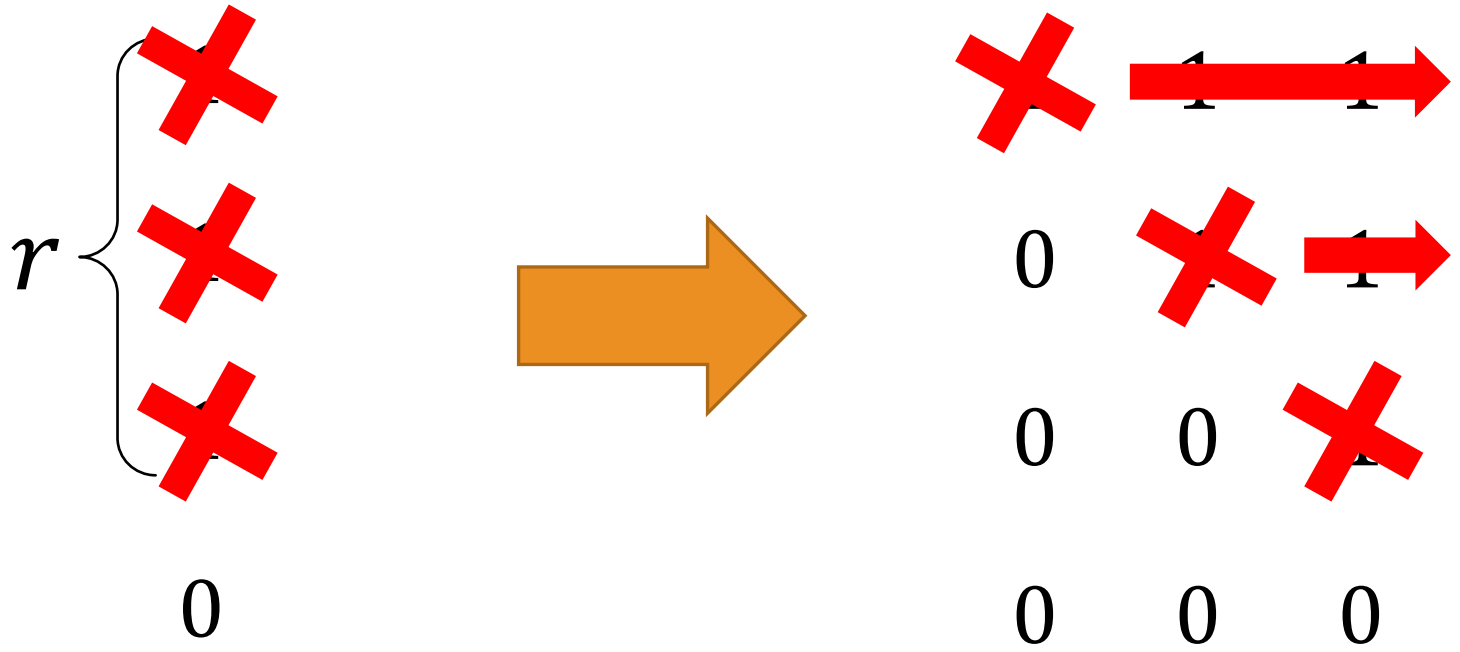
Expected loss: $\frac{r}{|S_t|}$ $\rightarrow$ $\frac{1}{|S_t|} + \frac{1}{|S_t|-1} + \frac{1}{|S_t|-2} \cdots \frac{1}{|S_t|-r+1} \geq \frac{1}{|S_t|} + \frac{1}{|S_t|} + \frac{1}{|S_t|} \cdots \frac{1}{|S_t|} = \frac{r}{|S_t|}$

Round 2 – Randomized Greedy



Expected loss: $\frac{r}{|S_t|}$ $\rightarrow$ $\frac{1}{|S_t|} + \frac{1}{|S_t|-1} + \frac{1}{|S_t|-2} \cdots \frac{1}{|S_t|-r+1} \geq \frac{1}{|S_t|} + \frac{1}{|S_t|} + \frac{1}{|S_t|} \cdots \frac{1}{|S_t|} = \frac{r}{|S_t|}$

Round 2 – Randomized Greedy



Expected loss: $\frac{r}{|S_t|}$ $\rightarrow$ $\frac{1}{|S_t|} + \frac{1}{|S_t|-1} + \frac{1}{|S_t|-2} \cdots \frac{1}{|S_t|-r+1} \geq \frac{1}{|S_t|} + \frac{1}{|S_t|} + \frac{1}{|S_t|} \cdots \frac{1}{|S_t|} = \frac{r}{|S_t|}$

Round 2 – Randomized Greedy

Suppose that the best action loses at time-steps $t_1, \ldots, t_\ell$ (in other words the best action loses ℓ times)

How much can RG lose between t_j and t_{j+1}?

If it lost k_j in total, this loss is maximized under a "slow knife" protocol, for a bound of

$$\frac{1}{|S_{t_j}|} + \frac{1}{|S_{t_j}|-1} + \cdots \frac{1}{|S_{t_j}|-k_j+1} \leq \log|S_t| + 1 \leq \log n + 1$$

Round 2 – Randomized Greedy

The RG algorithm can lose at most $\log n + 1$ times (in expectation) for every loss of the best action.

The worst-case "badness" of the RG algorithm is logarithmically bounded by the "badness" of the best action. Not so bad…

But we can do better!

Round 3 – Multiplicative Weights Updates

"The multiplicative weights algorithm was such a great idea, that it was discovered three times" – C. Papadimitriou [Seminar Talk]

Initialize $w_i^1 \leftarrow 1, p_i^1 \leftarrow \frac{1}{n}, R \leftarrow 0$;

At $t = 1, \ldots, \infty$:

For $i = 1, \ldots n$ **do**:

$w_i^t \leftarrow w_i^{t-1} \mathrm{e}^{-\epsilon \ell_i^{t-1}}$;

End For

$W_t \leftarrow \sum_i w_i^t$;

For $i = 1, \ldots n$ **do**:

$p_i^t \leftarrow \frac{w_i^t}{W_t}$;

End For

Discussion Points

- This entire section can easily be translated to negative rewards
- Rewards don't have to be in $\{0,1\}$; can be any real numbers in a range $[a, b]$: but if $b - a$ is very large, affects learning rate.
- A lot of the things here generalize to multi-armed bandits (need to devote more time to exploration – affects learning rate), but the same idea of the MWU algorithm holds:
  - Place exponential weights on actions
  - Perhaps change ε as a function of time