Lecture 12: Off-Policy Evaluation in Bandits

ID 6002W: Online and Reinforcement Learning

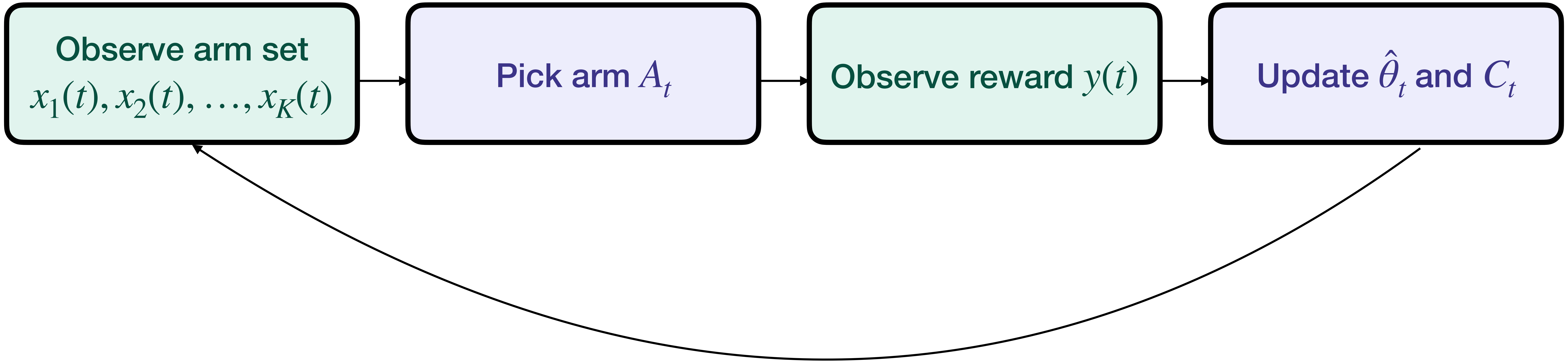
Web-enabled M.Tech. program, IIT Madras

Suryanarayana Sankagiri

17th June, 2026

How Do We Evaluate The Performance Of A Bandit Algorithm In Practice?

Regret Does Not Work In Practice



- In theory, and in simulations, we can calculate regret
- In practice, the best arm is unknown! $\Rightarrow$ regret impossible to calculate
- But we can <u>measure cumulative reward</u> for a bandit algorithm running in a real system

Practical Scenario

As a lead ML engineer

- You are running a bandit algorithm in production
  - e.g., recommending restaurants in Swiggy/Zomato

Practical Scenario

As a lead ML engineer

- You are running a bandit algorithm in production
  - e.g., recommending restaurants in Swiggy/Zomato
- Your team proposes several variants:
  - New reward model (lecture 11)
  - New way of designing context vectors (lecture 10)
  - New exploration-exploitation tradeoff constant (lecture 9)
- Which should you deploy?

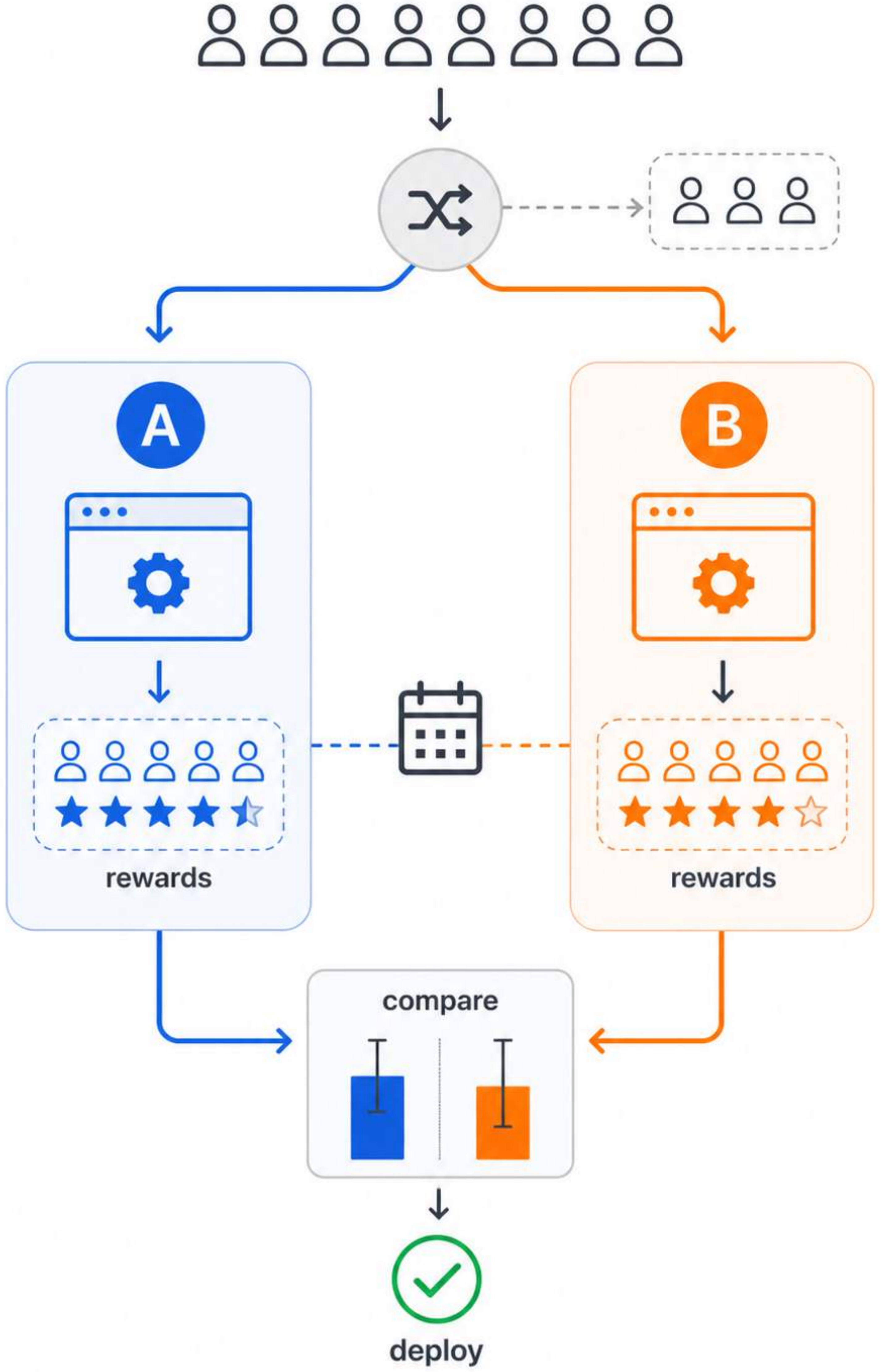
A/B Testing

The Gold Standard

What Is an A/B Test?

- Split incoming traffic between two policies
  - Group A: current production policy
  - Group B: new candidate policy
- Run both for some duration

Recall:
adaptive stopping time ↔
best-arm identification problem!

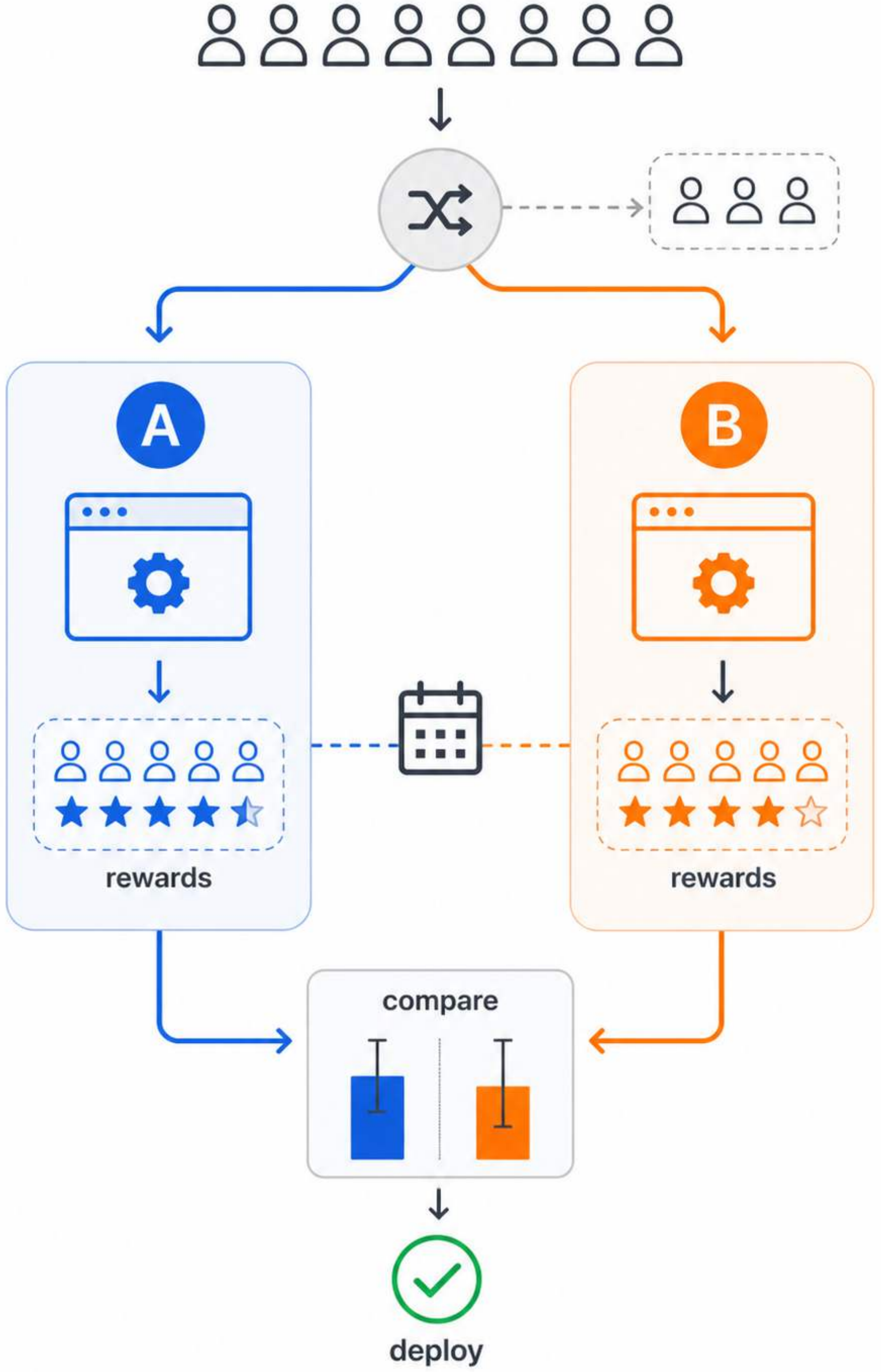


A/B Testing
The Gold Standard

What Is an A/B Test?

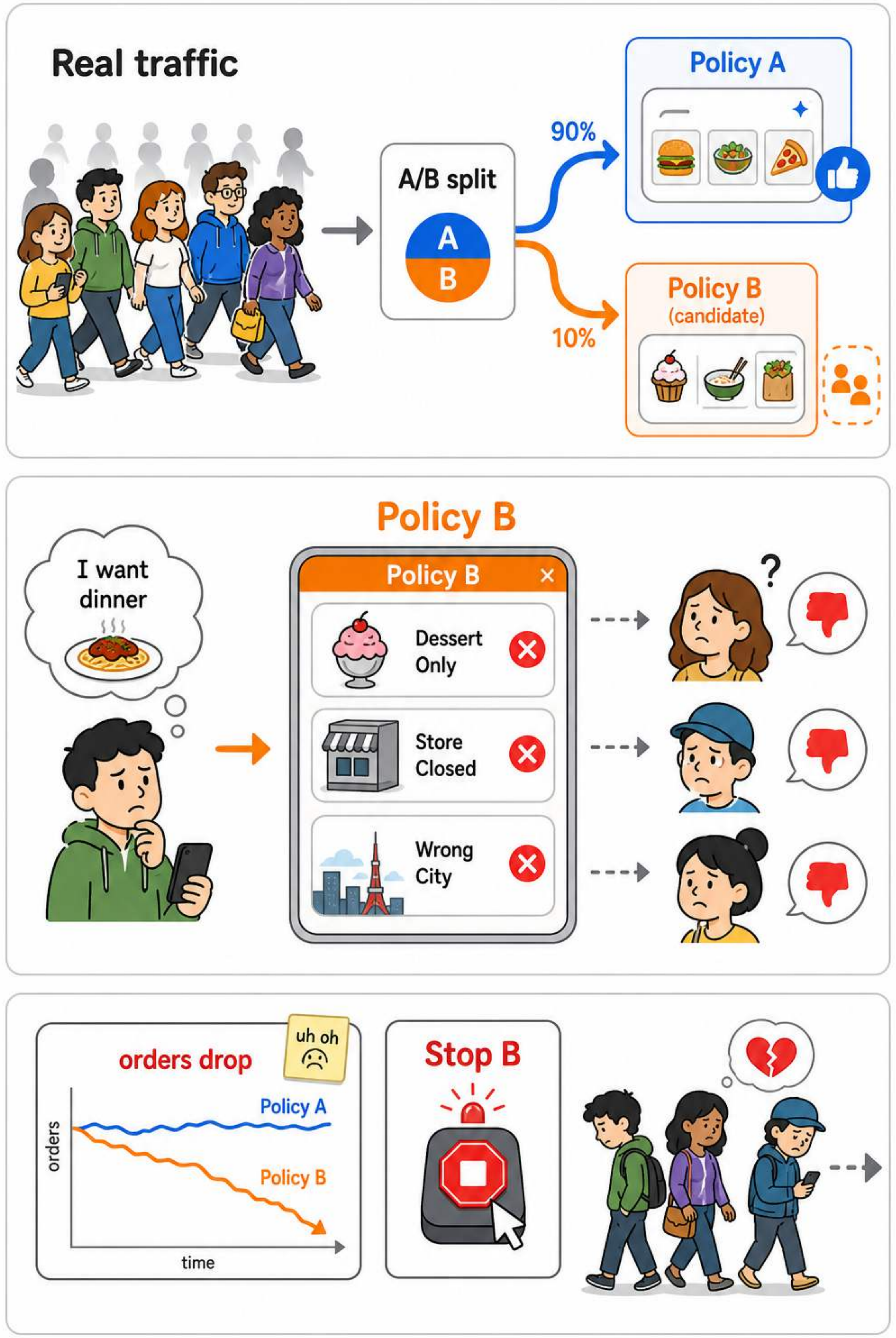
- Split incoming traffic between two policies
    - Group A: current production policy
    - Group B: new candidate policy
- Run both for some duration
- Observe rewards in each group, and compare them

→ decide whether to deploy new policy



A/B Testing Is Expensive

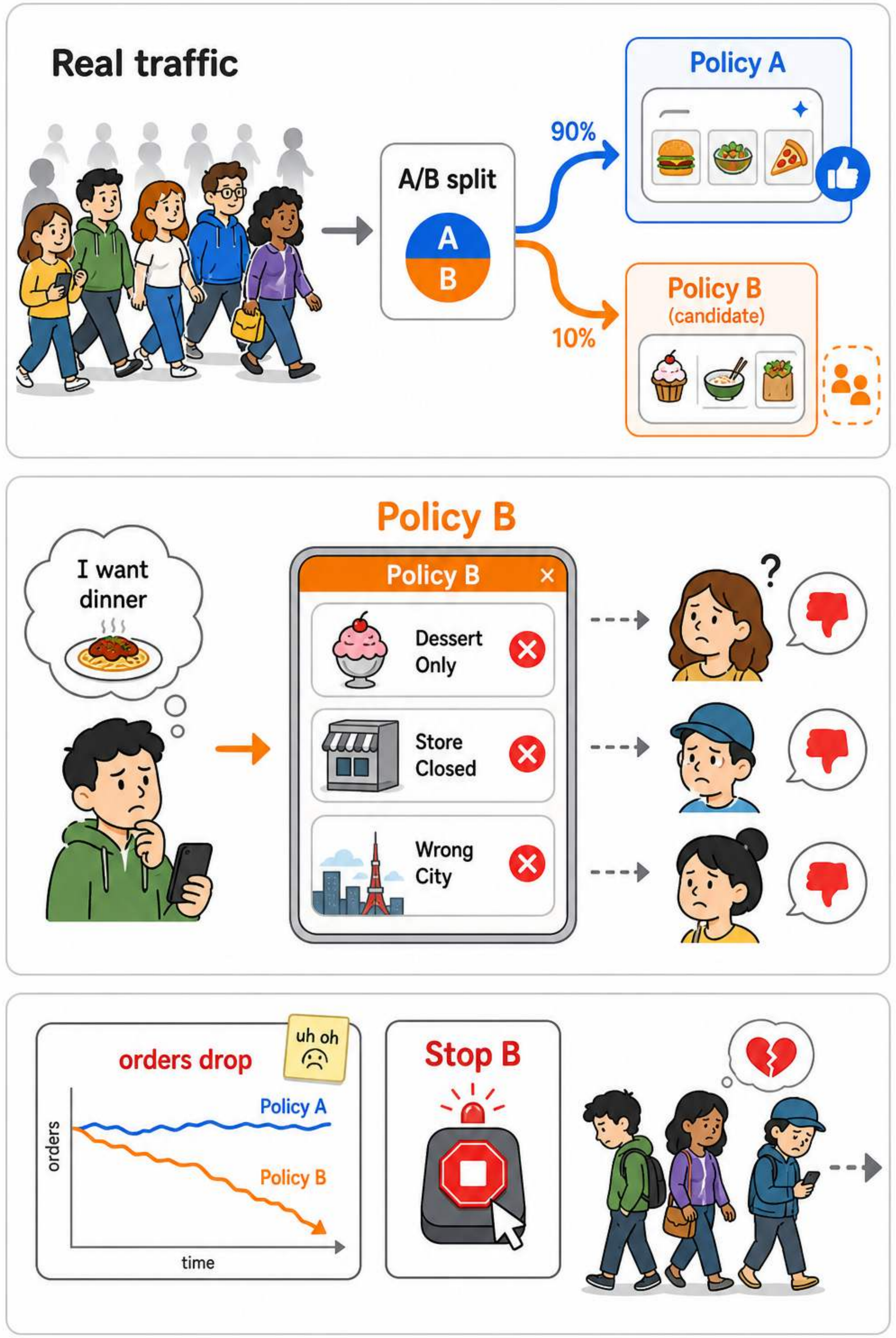
- A/B tests allocate real traffic to a candidate policy
- Highly risky and costly if candidate is worse!
  - May take many weeks
  - If new policy is very bad, users may never return!



A/B Testing Is Expensive

- A/B tests allocate real traffic to a candidate policy
- Highly risky and costly if candidate is worse!
  - May take many weeks
  - If new policy is very bad, users may never return!
- Cannot test every candidate online
  - Need an offline filter to triage candidate policies

How do we do that?



Off-Policy Evaluation Solves This Problem

Data For Off-Policy Evaluation

Production Logs

- A log records what the production policy did at every time step

t	external context $C(t)$	available articles $\mathcal{X}_t$	played $A(t)$	reward $\mathbf{y}(t)$
1	user A asks: sports	{17, 3, 41, ...}	17	1
2	user B asks: tech	{12, 3, 28, ...}	3	0
3	user C asks: politics	{9, 14, 21, ...}	9	1
...	...	...	...	...

Σ Each article $i \in \mathcal{X}_t$ has features $\mathbf{x}_i(t) \in \mathbb{R}^d$

{ } Log of T impressions: $\{\,(C(t),\, \mathcal{X}_t,\, A(t),\, \mathbf{y}(t))\}_{t=1}^{T}$

Off-Policy Evaluation

- Suppose have logs from a policy that ran in production

Can we estimate, from these logs alone,
what a different policy would have done?

Off-Policy Evaluation

- Suppose have logs from a policy that ran in production

Can we estimate, from these logs alone,
what a different policy would have done?

This is called off-policy evaluation (OPE)

- Correct logging is super-critical!

Equally important in reinforcement learning!

But What Is A Policy?

Defining A Policy

- A policy π is a rule for picking an arm from the action set $\mathcal{X} = \{x_1, x_2, \ldots, x_K\}$
- $\pi(i \mid \mathcal{X})$: probability of picking arm i given $\mathcal{X}$

Defining A Policy

- A policy π is a rule for picking an arm from the action set $\mathcal{X} = \{x_1, x_2, \ldots, x_K\}$
- $\pi(i \mid \mathcal{X})$: probability of picking arm i given $\mathcal{X}$
- A policy is fully specified by these probabilities
  - Uniform random: $\pi(i \mid \mathcal{X}) = 1/K$
  - Greedy on $\hat{\theta}$: $\pi(i \mid \mathcal{X}) = 1$ if $i = \arg\max \langle \hat{\theta}, x_i \rangle$
  - ε-greedy: $\pi(i \mid \mathcal{X}) = 1 - \varepsilon$ on $\arg\max \langle \hat{\theta}, x_i \rangle$, ε/K elsewhere

Defining A Policy

- A policy π is a rule for picking an arm from the action set $\mathcal{X} = \{x_1, x_2, \ldots, x_K\}$
- $\pi(i \mid \mathcal{X})$: probability of picking arm i given $\mathcal{X}$
- A policy is fully specified by these probabilities
  - Deterministic policy: $\pi(i \mid \mathcal{X}) \in \{0, 1\}$
  - Randomised policy: $\pi(i \mid \mathcal{X})$ is a non-trivial distribution

Algorithm v/s Policy

- A learning algorithm induces a <u>changing sequence of policies</u>
  - Its internal state $(\hat{\theta}_t, V_t)$ updates as data comes in
    - $\pi(i \mid \mathcal{X})$ changes after each update

Algorithm v/s Policy

- A learning algorithm induces a changing sequence of policies
  - Its internal state $(\hat{\theta}_t, V_t)$ updates as data comes in
    - $\pi(i \mid \mathcal{X})$ changes after each update

- A frozen snapshot of LinUCB — fixed $(\hat{\theta}, V, \beta)$ — is a policy:
  - $\pi(i \mid \mathcal{X}) = 1$ if $i = \arg\max_j \langle \hat{\theta}, x_j \rangle + \beta \|x_j\|_{V^{-1}}$, else 0
    - Deterministic, assuming fixed tie-breaking

Algorithm v/s Policy

- A learning algorithm induces a <u>changing sequence of policies</u>
  - Its internal state $(\hat{\theta}_t, V_t)$ updates as data comes in
    - $\pi(i \mid \mathcal{X})$ changes after each update

- A frozen snapshot of LinTS — fixed posterior over θ — is a policy:
  - Sample $\tilde{\theta}$ ~ posterior, play $\arg\max\langle\tilde{\theta}, x_i\rangle$
    - Randomised, because $\tilde{\theta}$ is sampled fresh each time

Off-Policy Evaluation

Setting and Notation

- In off-policy evaluation, we have two policies
  - Logging policy: $\pi_0(i \mid \mathscr{X})$ — produced the data
  - Target policy: $\pi(i \mid \mathscr{X})$ — a new policy we want to evaluate
  - For simplicity, we will assume they are fixed policies, not learning algorithms

Called logger for short

Off-Policy Evaluation

Setting and Notation

- In off-policy evaluation, we have two policies
  - Logging policy: $\pi_0(i \mid \mathcal{X})$ — produced the data
  - Target policy: $\pi(i \mid \mathcal{X})$ — a new policy we want to evaluate
  - For simplicity, we will assume they are fixed policies, not learning algorithms
- We care about value of a policy: expected reward in one time step if π is used

$$V(\pi) = \mathbb{E}_{\mathcal{X}}[\Sigma \pi(i \mid \mathcal{X}) \langle \theta, x_i \rangle]$$

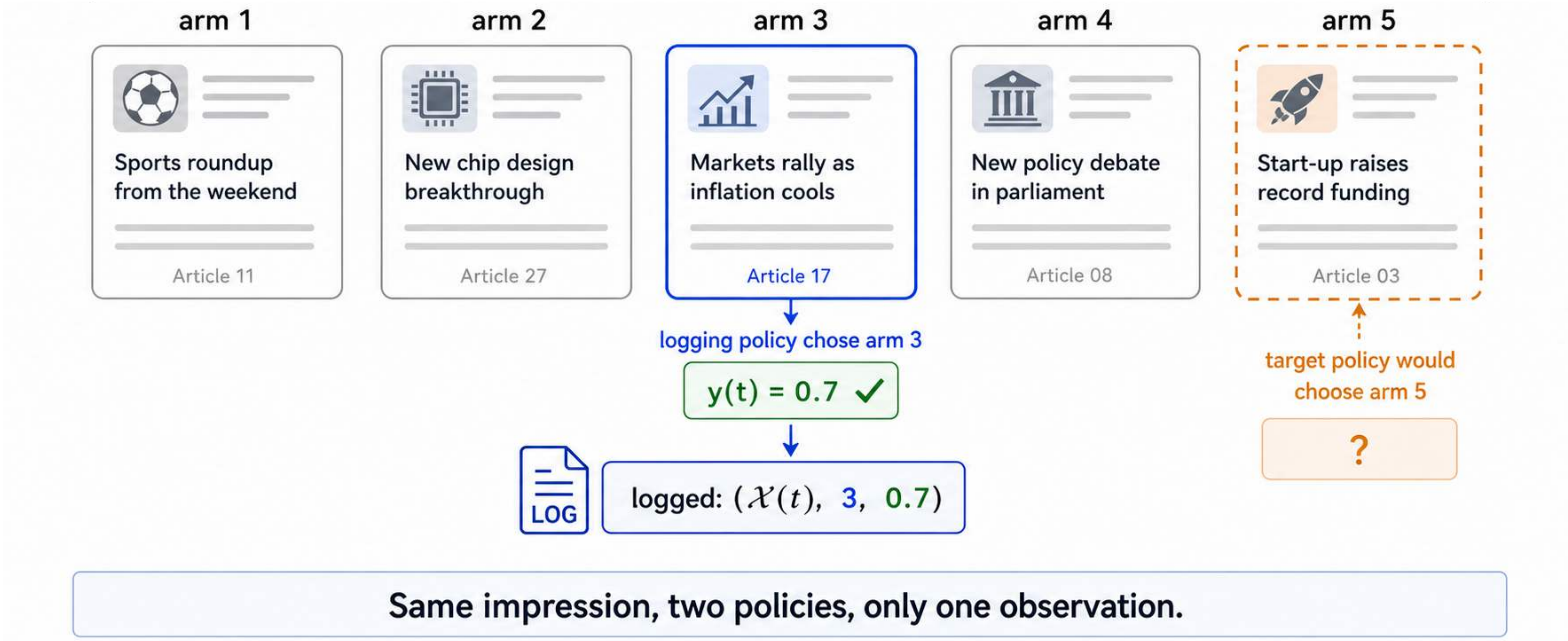
- Goal: estimate $V(\pi)$ using logs collected under π_0

Called logger for short

The Counterfactual Question

What we observe versus what we want

We observe what the logging policy did.
We want to know what a different policy would have done.



The Replay Method

When The Logger Is Uniform

- Suppose $\pi_0(i \,|\, \mathscr{X}) = 1/K$ for every arm i (uniform exploration)
- Suppose target π is deterministic: in each context, π picks one arm

When The Logger Is Uniform

- Suppose $\pi_0(i \mid \mathscr{X}) = 1/K$ for every arm i (uniform exploration)
- Suppose target π is deterministic: in each context, π picks one arm

✳ Idea: go step by step along the log

➡ Keep the rows where the logger happened to pick what π would pick

➡ Discard the rest

When The Logger Is Uniform

- Suppose $\pi_0(i \mid \mathscr{X}) = 1/K$ for every arm i (uniform exploration)
- Suppose target π is deterministic: in each context, π picks one arm

✳ Idea: go step by step along the log

➡ Keep the rows where the logger happened to pick what π would pick

➡ Discard the rest

- Introduced by Li et al. (2010) for evaluating news recommendation on Yahoo!
- a small slice of traffic served by uniform random policy, kept aside specifically for evaluation

Replay Method Mechanics

Simplest Off-Policy Method

row	context / set	logged $A(t)$	target $\pi(C, \mathcal{X})$	reward $y(t)$	use?
1	sports	17	17	1	keep ✓
2	tech	3	12	0	discard ✗
3	politics	9	21	1	discard ✗
4	movies	5	5	0	keep ✓
5	finance	8	2	1	discard ✗

1

0

estimate =
average kept rewards

$(1 + 0) / 2 = 0.5$

Replay keeps only the rows where logger and target agree.

Analysing Replay Method

Why does this method work?

- The logger chooses each arm uniformly at random
  - The event “logger agrees with π” is occurs with probability $1/K$, independent of reward
  - Kept rows are a random sample of the times where π was effectively played
  - Averaging their rewards gives an estimate of $V(\pi)$

Analysing Replay Method

Why does this method work?

- The logger chooses each arm uniformly at random
  - The event “logger agrees with π” is occurs with probability $1/K$, independent of reward
  - Kept rows are a random sample of the times where π was effectively played
  - Averaging their rewards gives an estimate of $V(\pi)$
- Cons:
  - Only $1/K$ of the log is used → wasteful for large K
  - Requires uniform logging → does not apply to production policies

We need a method that works for any known logging policy π_0

The Inverse-Propensity Scoring (IPS) Method

What Must Be Logged

Per impression/interaction/round t, record:

What Must Be Logged

Per impression/interaction/round t, record:

- Action set: $\mathscr{X}(t) = \{x_1(t), \ldots, x_K(t)\}$
- Played arm: $A(t) \in \{1,\ldots,K\}$
- Reward: $y(t)$

What Must Be Logged

Per impression/interaction/round t, record:

- Action set: $\mathscr{X}(t) = \{x_1(t), \ldots, x_K(t)\}$
- Played arm: $A(t) \in \{1, \ldots, K\}$
- Reward: $y(t)$
- Propensity: $p(t) = \pi_0 \left(A(t) \mid \mathscr{X}(t)\right)$

Note that propensity cannot be recovered from other data!

The IPS Estimator

Estimator

$$\widehat{V}_{\mathrm{IPS}}(\pi) = \frac{1}{T}\sum_{t=1}^{T} \frac{\pi(A_t \mid \mathcal{X}_t)}{p_t}\, y_t, \qquad p_t = \pi_0(A_t \mid \mathcal{X}_t).$$

The IPS Estimator

Estimator

$$\widehat{V}_{\mathrm{IPS}}(\pi) = \frac{1}{T}\sum_{t=1}^{T} \frac{\pi(A_t \mid \mathcal{X}_t)}{p_t}\, y_t, \qquad p_t = \pi_0(A_t \mid \mathcal{X}_t).$$

Why the ratio works: condition on one impression $\mathcal{X}$

$$\begin{aligned}
\mathbb{E}_{A\sim\pi_0}\left[\frac{\pi(A \mid \mathcal{X})}{\pi_0(A \mid \mathcal{X})}\mu(\mathcal{X}, A)\right] &= \sum_{i\in\mathcal{X}} \pi_0(i \mid \mathcal{X})\frac{\pi(i \mid \mathcal{X})}{\pi_0(i \mid \mathcal{X})}\mu(\mathcal{X}, i) \\
&= \sum_{i\in\mathcal{X}} \pi(i \mid \mathcal{X})\, \mu(\mathcal{X}, i) \\
&= V(\pi \mid \mathcal{X}).
\end{aligned}$$

The IPS Estimator

Estimator

$$\widehat{V}_{\mathrm{IPS}}(\pi) = \frac{1}{T}\sum_{t=1}^{T} \frac{\pi(A_t \mid \mathcal{X}_t)}{p_t}\, y_t, \qquad p_t = \pi_0(A_t \mid \mathcal{X}_t).$$

Why the ratio works: condition on one impression $\mathcal{X}$

$$\begin{aligned}
\mathbb{E}_{A\sim\pi_0}\left[\frac{\pi(A \mid \mathcal{X})}{\pi_0(A \mid \mathcal{X})}\mu(\mathcal{X}, A)\right] &= \sum_{i\in\mathcal{X}} \pi_0(i \mid \mathcal{X})\frac{\pi(i \mid \mathcal{X})}{\pi_0(i \mid \mathcal{X})}\mu(\mathcal{X}, i) \\
&= \sum_{i\in\mathcal{X}} \pi(i \mid \mathcal{X})\,\mu(\mathcal{X}, i) \\
&= V(\pi \mid \mathcal{X}).
\end{aligned}$$

Takeaway. After averaging over impressions, $\mathbb{E}[\widehat{V}_{\mathrm{IPS}}(\pi)] = V(\pi)$, provided $\pi_0(i \mid \mathcal{X}) > 0$ whenever $\pi(i \mid \mathcal{X}) > 0$.

The Variance Cost

- IPS is unbiased — but variance can be enormous!
- If $p(t)$is very small for some t, the weight $\pi(A(t) \mid x(t))/p(t)$ becomes very large
  - A single such sample can dominate the estimate!
  - Variance grows with square of the weight
- Practical cost of going from replay to IPS
  - We use all the data, but the price is variance
- Many heuristics to reduce variance exist (we won't cover)

Conclusion

- New bandit algorithms need to be safely evaluated before deployment!
- Off-policy evaluation: estimate value of a policy from logs of a different policy
- Requires:
  - a stochastic logger
  - logged propensities
- Estimator: Inverse Propensity Scoring - Need to watch out for high variance!
- OPE is used as a filter before A/B tests, not a replacement for them