Reinforcement Learning for Recommender Systems

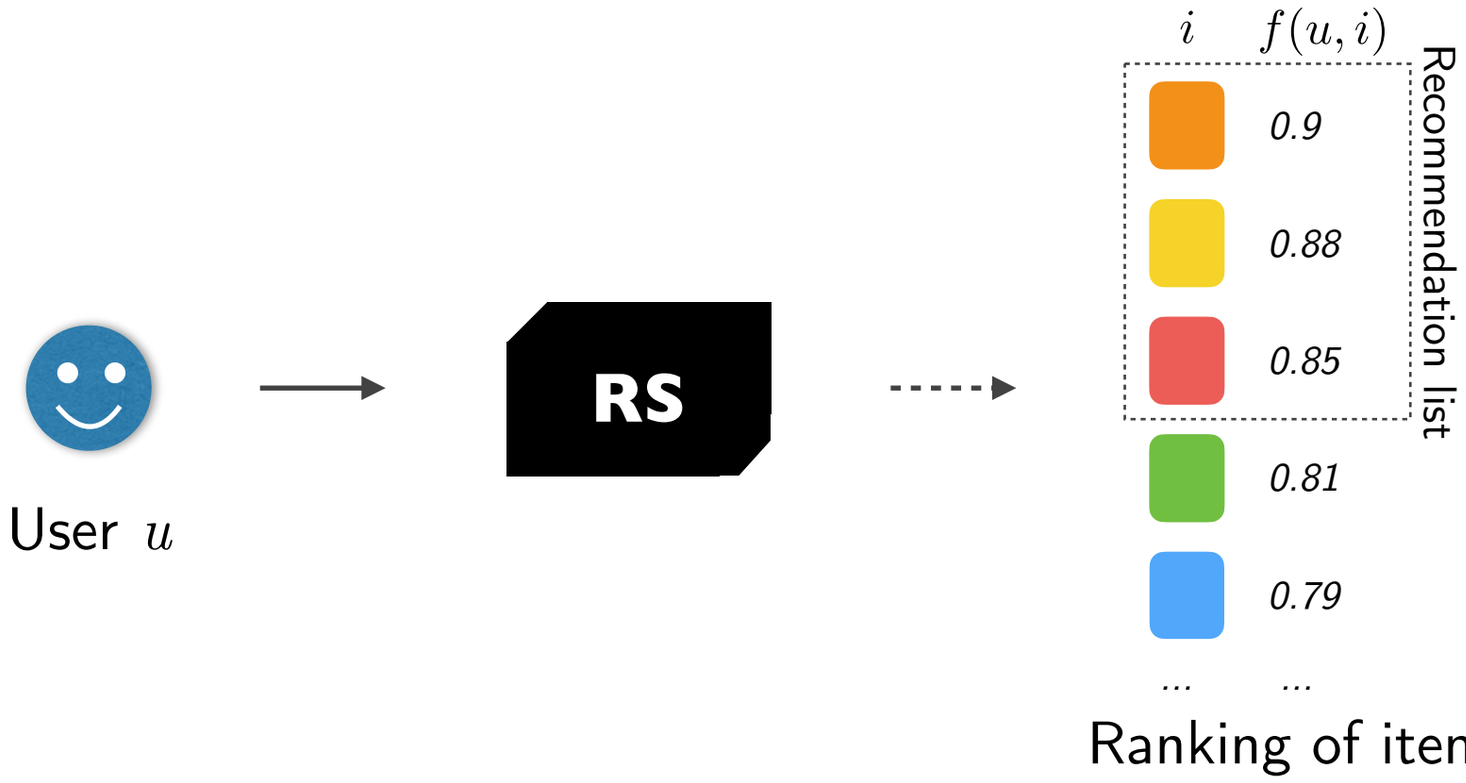
Marie Al Ghossein

June 10, 2024

Crossing Minds

Dauphine | PSL
UNIVERSITÉ PARIS

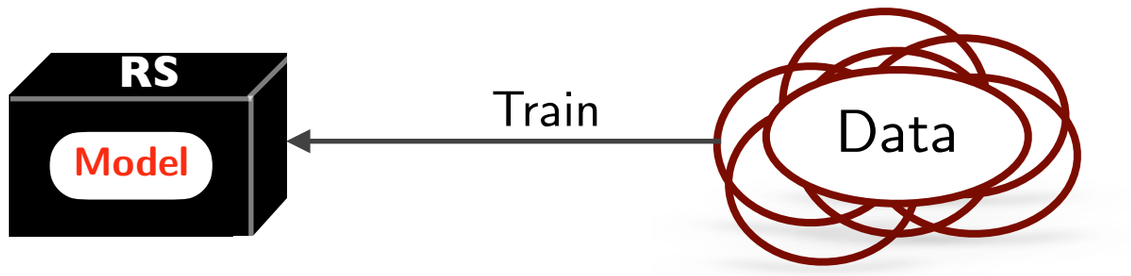
Recommender System (RS)



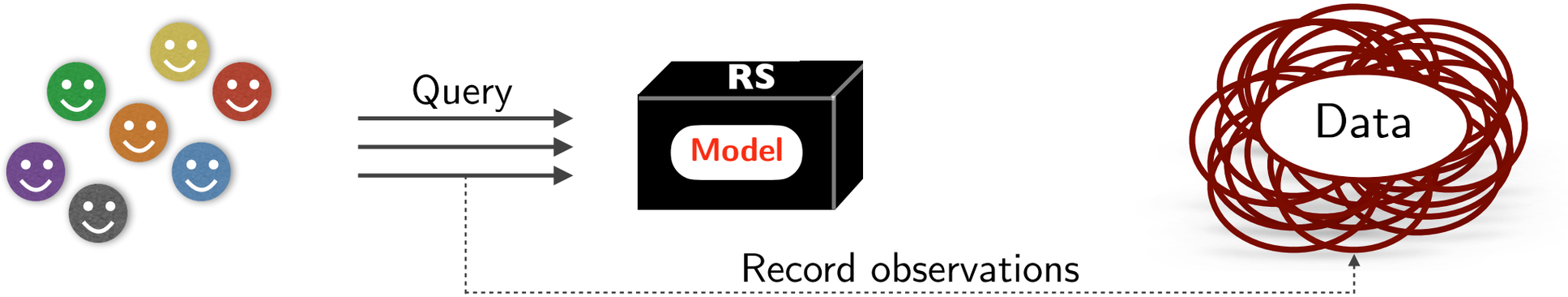
User u

Ranking of items

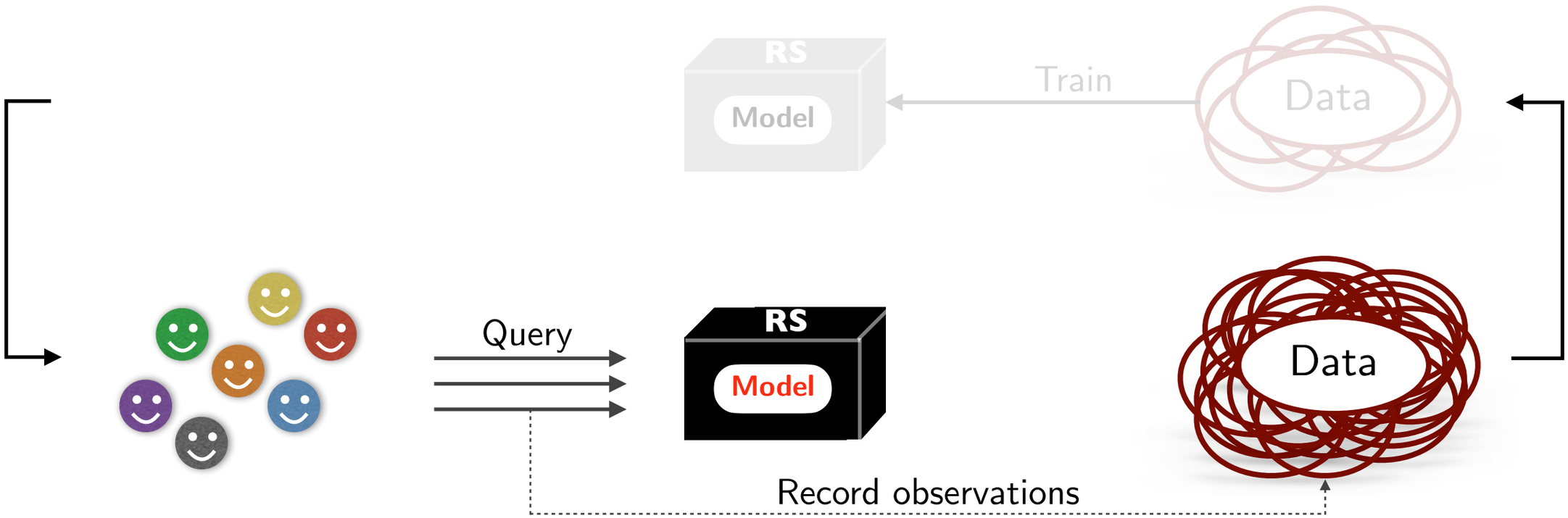
Functioning of traditional RS



Functioning of traditional RS



Functioning of traditional RS



What happens between two training phases?

- New events are observed
  - New items are added to the catalog
  - New users join the service
  - New interactions are collected

What happens between two training phases?

- New events are observed
  - New items are added to the catalog
  - New users join the service
  - New interactions are collected
- Challenges for traditional RS
  - User and item cold-start

What happens between two training phases?

- New events are observed
  - New items are added to the catalog
  - New users join the service
  - New interactions are collected
- Challenges for traditional RS
  - User and item cold-start
  - Non-stationarity
    - User preferences and needs evolve over time
    - Content popularity and item perceptions change over time

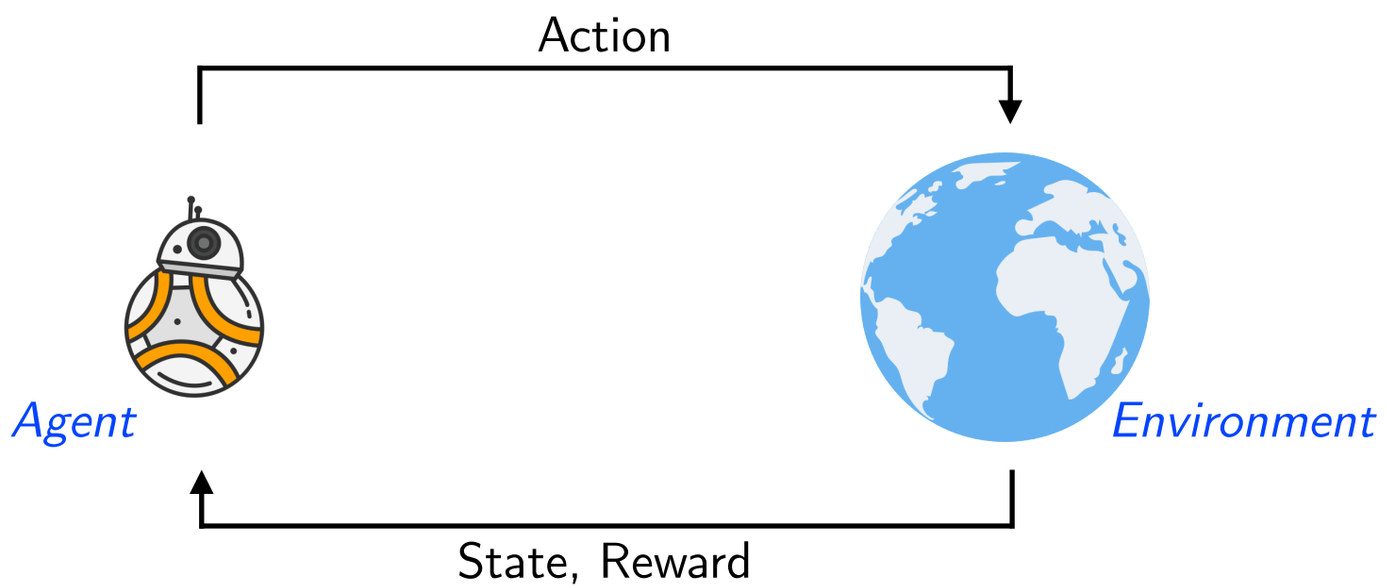
What happens between two training phases?

- New events are observed
  - New items are added to the catalog
  - New users join the service
  - New interactions are collected
- Challenges for traditional RS
  - User and item cold-start
  - Non-stationarity
    - User preferences and needs evolve over time
    - Content popularity and item perceptions change over time

$\longrightarrow$ Explore another set of solutions based on Reinforcement Learning

Reinforcement Learning (RL)

Learning from interacting with the environment what actions to take in a given situation, in order to maximize a reward.



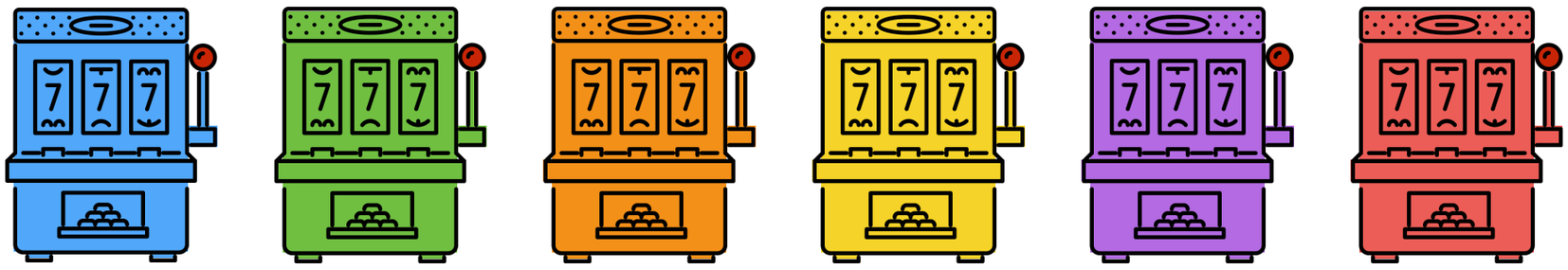
Reinforcement Learning (RL)

Learning from interacting with the environment what actions to take in a given situation, in order to maximize a reward.

Characteristics of RL

- The learning system's actions influence the subsequent data it receives
- There is no supervisor, but rather a reward signal
- Feedback may be delayed
- Sequential decision making process

Multi-armed bandits



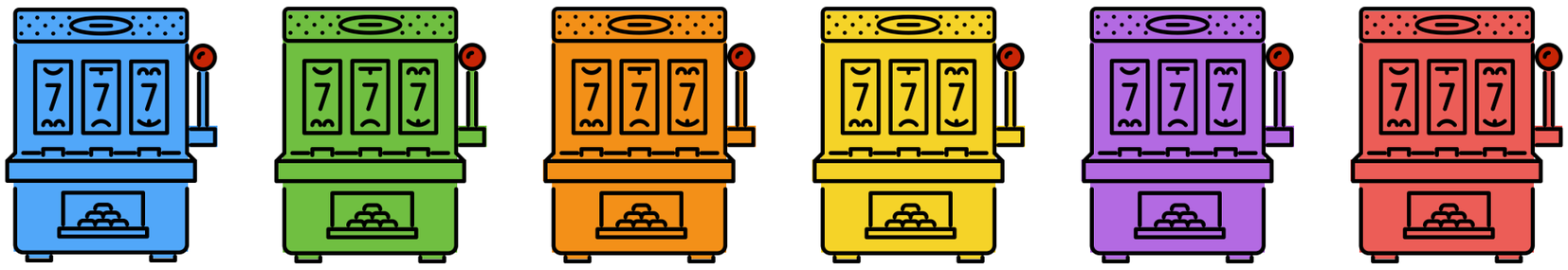
You are at a casino.

There are several slot machines, *i.e.*, one-armed bandits.

Each one of them has an unknown probability of how likely you will win at one play.

You have a limited amount of ressources, *i.e.*, number of times to play.

Multi-armed bandits



You are at a casino.

There are several slot machines, *i.e.*, one-armed bandits.

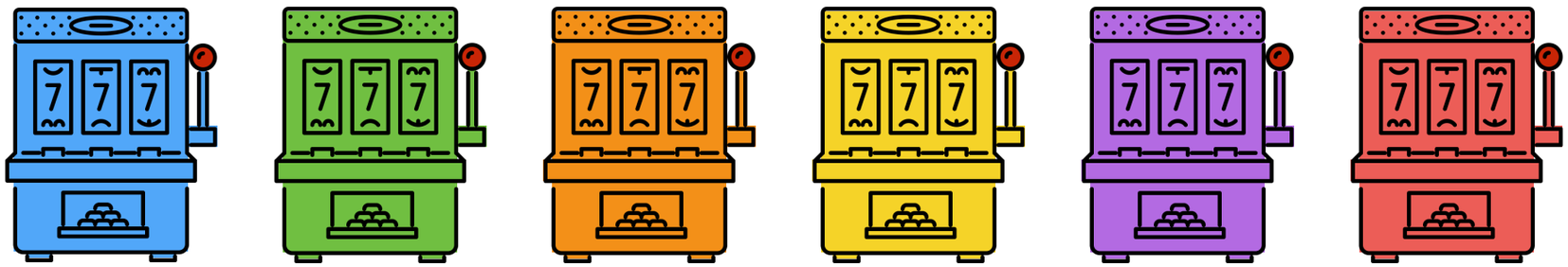
Each one of them has an unknown probability of how likely you will win at one play.

You have a limited amount of ressources, *i.e.*, number of times to play.

Goal: Get rich 💰, the richest possible. **Best strategy?**

How to learn which arm is the best in the shortest amount of time?

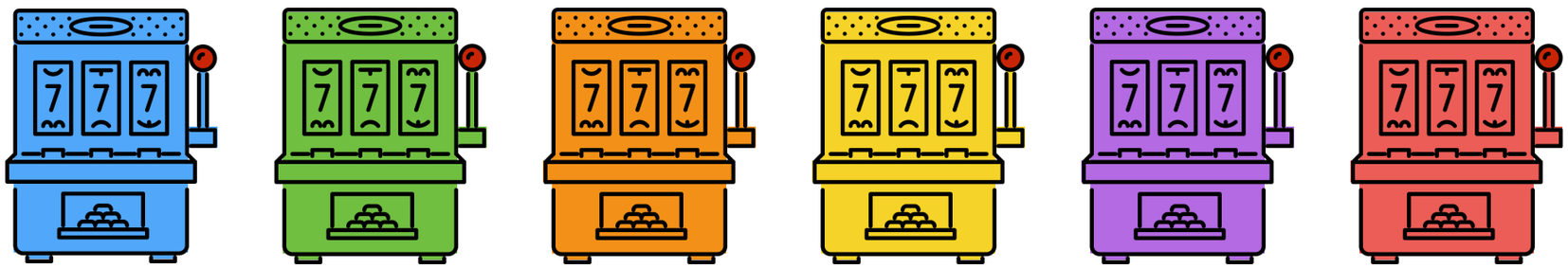
Multi-armed bandits



Challenges:

- You do not know the performance of the arm you do not play
- Whenever you pull a bad arm, you are wasting an opportunity to win

Multi-armed bandits



Challenges:

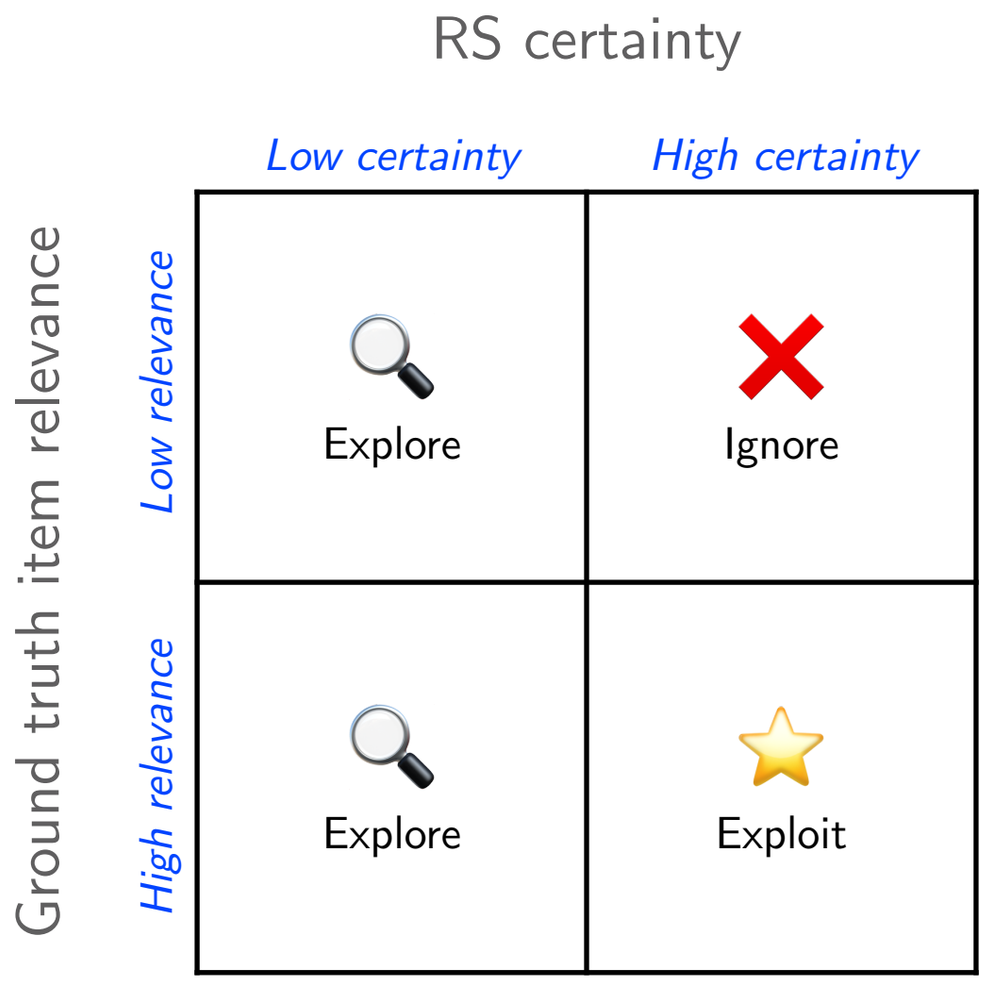
- You do not know the performance of the arm you do not play
  → you should gain knowledge by pulling different arms → **exploration**
- Whenever you pull a bad arm, you are wasting an opportunity to win
  → you should repeatedly pull the best arm → **exploitation**

Exploration-Exploitation dilemma

With no initial knowledge, each step involves a fundamental choice:

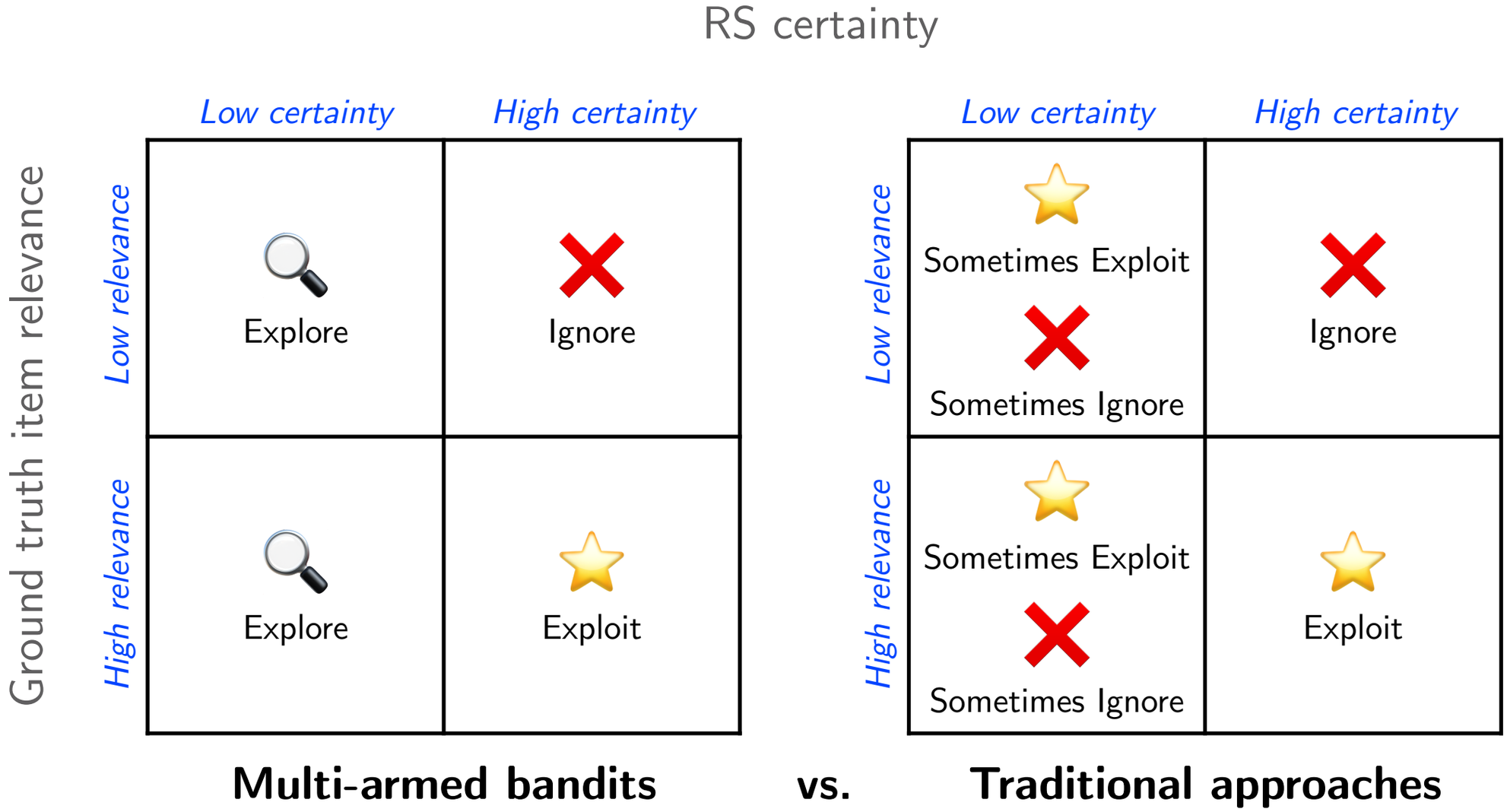
- Exploitation. Take the best decision given current knowledge,
  i.e., play the machine that gives the highest expected payoff.
- Exploration. Gather more knowledge,
  i.e., play other machines, hoping to find one giving a higher payoff.

Exploration-Exploitation dilemma



James McInerney et al. *Explore, Exploit, and Explain: Personalizing Explainable Recommendations with Bandits*. In Proc. RecSys, 2018.

Exploration-Exploitation dilemma



James McInerney et al. *Explore, Exploit, and Explain: Personalizing Explainable Recommendations with Bandits*. In Proc. RecSys, 2018.

Exploration-Exploitation dilemma

Concepts

- Naive exploration

Exploration-Exploitation dilemma

Concepts

- Naive exploration
- Optimistic initialization

  Assume the best until proven otherwise

Exploration-Exploitation dilemma

Concepts

- Naive exploration
- Optimistic initialization

  Assume the best until proven otherwise
- Optimism in the face of uncertainty

  Favor actions with uncertain values

Exploration-Exploitation dilemma

Concepts

- Naive exploration
- Optimistic initialization

  Assume the best until proven otherwise
- Optimism in the face of uncertainty

  Favor actions with uncertain values
- Probability matching

  Select actions according to the probability that they are the most optimal

Multi-armed bandits for recommendation

A multi-armed bandit is a tuple $\langle R, A \rangle$ where

- A is a known set of m actions or arms,
- $R^a(r) = \mathbf{P}[r \mid a]$ is an unknown probability distribution over rewards

Multi-armed bandits for recommendation

A multi-armed bandit is a tuple $\langle R, A \rangle$ where

- A is a known set of m actions or arms,
- $R^a(r) = \mathbf{P}[r \mid a]$ is an unknown probability distribution over rewards

With an agent following the process described below:

- Enter the world with zero knowledge
- At each time step t,
  - Pick an action $a_t \in A$
  - Observe a reward $r_t \sim R^{a_t}$

Multi-armed bandits for recommendation

A multi-armed bandit is a tuple $\langle R, A \rangle$ where

- A is a known set of m actions or arms,
- $R^a(r) = \mathbf{P}[r \mid a]$ is an unknown probability distribution over rewards

With an agent following the process described below:

- Enter the world with zero knowledge
- At each time step t,
  - Pick an action $a_t \in A$
  - Observe a reward $r_t \sim R^{a_t}$

Goal: Maximize cumulative reward $\sum_{t=1}^{T} r_t$

David Silver. *Exploration and Exploitation*, UCL course on RL.

Multi-armed bandits for recommendation

A multi-armed bandit is a tuple $\langle R, A \rangle$ where

- A is a known set of m actions or arms, → items
- $R^a(r) = \mathbf{P}[r \mid a]$ is an unknown probability distribution over rewards

With an agent following the process described below:

- Enter the world with zero knowledge
- At each time step t,
  - Pick an action $a_t \in A$ → Recommend item
  - Observe a reward $r_t \sim R^{a_t}$ → Get user feedback

Goal: Maximize cumulative reward $\sum_{t=1}^{T} r_t$

Multi-armed bandits for recommendation

- The expected reward for action a:

$$Q(a) = \mathbf{E}(r \mid a)$$

- The best action in terms of expected rewards:

$$a^* = \arg\max_{a \in A} Q(a)$$

yielding the expected reward of:

$$V^* = Q(a^*) = \max_{a \in A} Q(a)$$

Multi-armed bandits for recommendation

- The expected reward for action a:

$$Q(a) = \mathbf{E}(r \mid a)$$

- The best action in terms of expected rewards:

$$a^* = \arg\max_{a \in A} Q(a)$$

yielding the expected reward of:

$$V^* = Q(a^*) = \max_{a \in A} Q(a)$$

- The regret is defined as the total opportunity loss:

$$L_t = \mathbf{E}\Big[\sum_{t=1}^{T} \Big(q^* - r_t\Big)\Big]$$

Multi-armed bandits for recommendation

- The expected reward for action a:

$$Q(a) = \mathbf{E}(r \mid a)$$

- The best action in terms of expected rewards:

$$a^* = \arg\max_{a \in A} Q(a)$$

yielding the expected reward of:

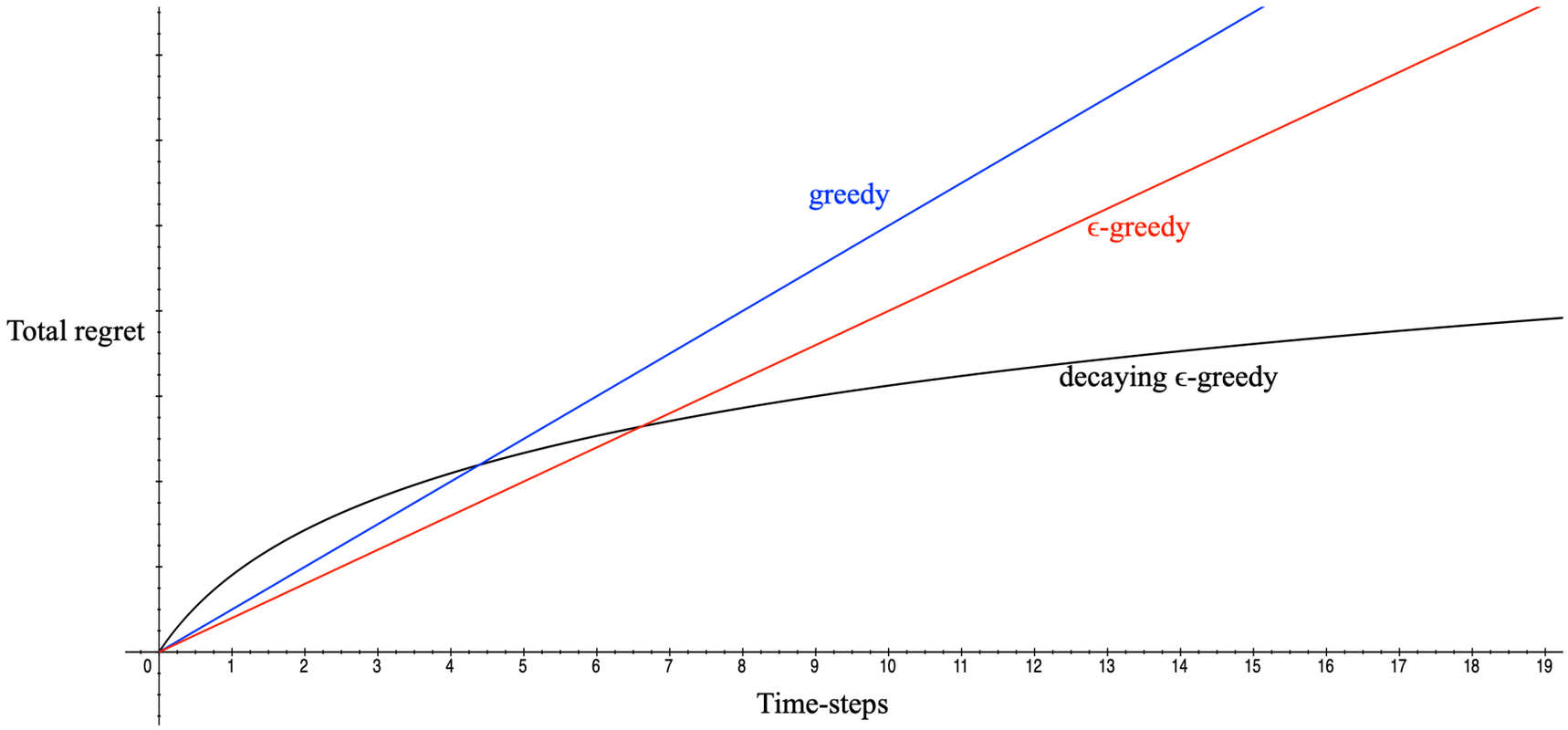
$$V^* = Q(a^*) = \max_{a \in A} Q(a)$$

- The regret is defined as the total opportunity loss:

$$L_t = \mathbf{E}\Big[\sum_{t=1}^{T}\Big(q^* - r_t\Big)\Big]$$

Maximizing the cumulative reward ≡ Minimizing total regret

Sublinear regret



An algorithm that explores forever has a **linear regret**

An algorithm that never explores has a **linear regret**

Can we achieve a **sublinear regret**?, *the algorithm converges to the best action*

[Figure taken from] David Silver. *Exploration and Exploitation*, UCL course on RL.

Some of the algorithms in review

- Greedy algorithms
- Upper Confidence Bound (UCB)
- Thompson Sampling (TS)

Greedy algorithms

Idea.

- Estimate the value of each action, *i.e.*, its expected reward, $\hat{Q}_t(a) \approx Q(a)$

  e.g., $\hat{Q}_t(a)$ is the mean of all rewards obtained when a was played

- Select the action with the highest value,

$$a^* = \arg\max_{a \in A} Q_t(a)$$

Greedy algorithms

Algorithm.

- Initialize for all actions a:
  - $N(a) \leftarrow 0$
  - $Q(a) \leftarrow 0$
- For $t = 1, 2, \ldots$:
  - $a \leftarrow \arg\max_{a \in A} Q(a)$
  - Observe reward r
  - $N(a) \leftarrow N(a) + 1$
  - $Q(a) \leftarrow Q(a) + \frac{1}{N(a)}\Big(r - Q(a)\Big)$

Greedy algorithms

Idea.

- Estimate the value of each action, *i.e.*, its expected reward, $\hat{Q}_t(a) \approx Q(a)$

  e.g., $\hat{Q}_t(a)$ is the mean of all rewards obtained when a was played

- Select the action with the highest value,

$$a^* = \arg\max_{a \in A} Q_t(a)$$

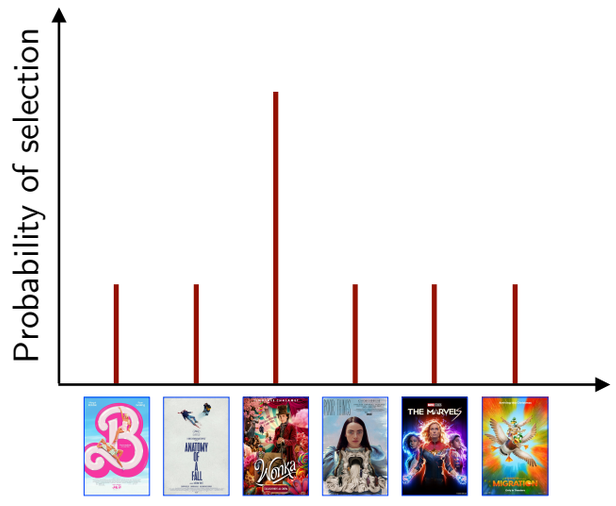
Drawbacks.

- It can lock onto a suboptimal action and exploit it forever
- It has a linear regret

ϵ-greedy algorithm

Idea.

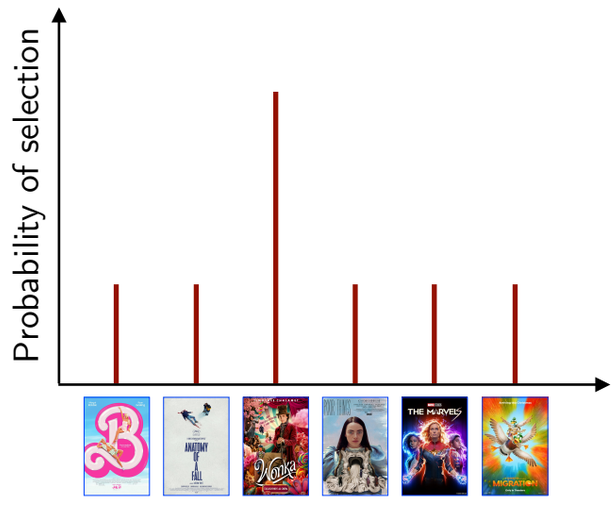
- With probability (1 - ϵ), select the best action $a^* = \arg\max_{a \in A} \hat{Q}_t(a)$
- With probability ϵ, select a random action



ϵ-greedy algorithm

Idea.

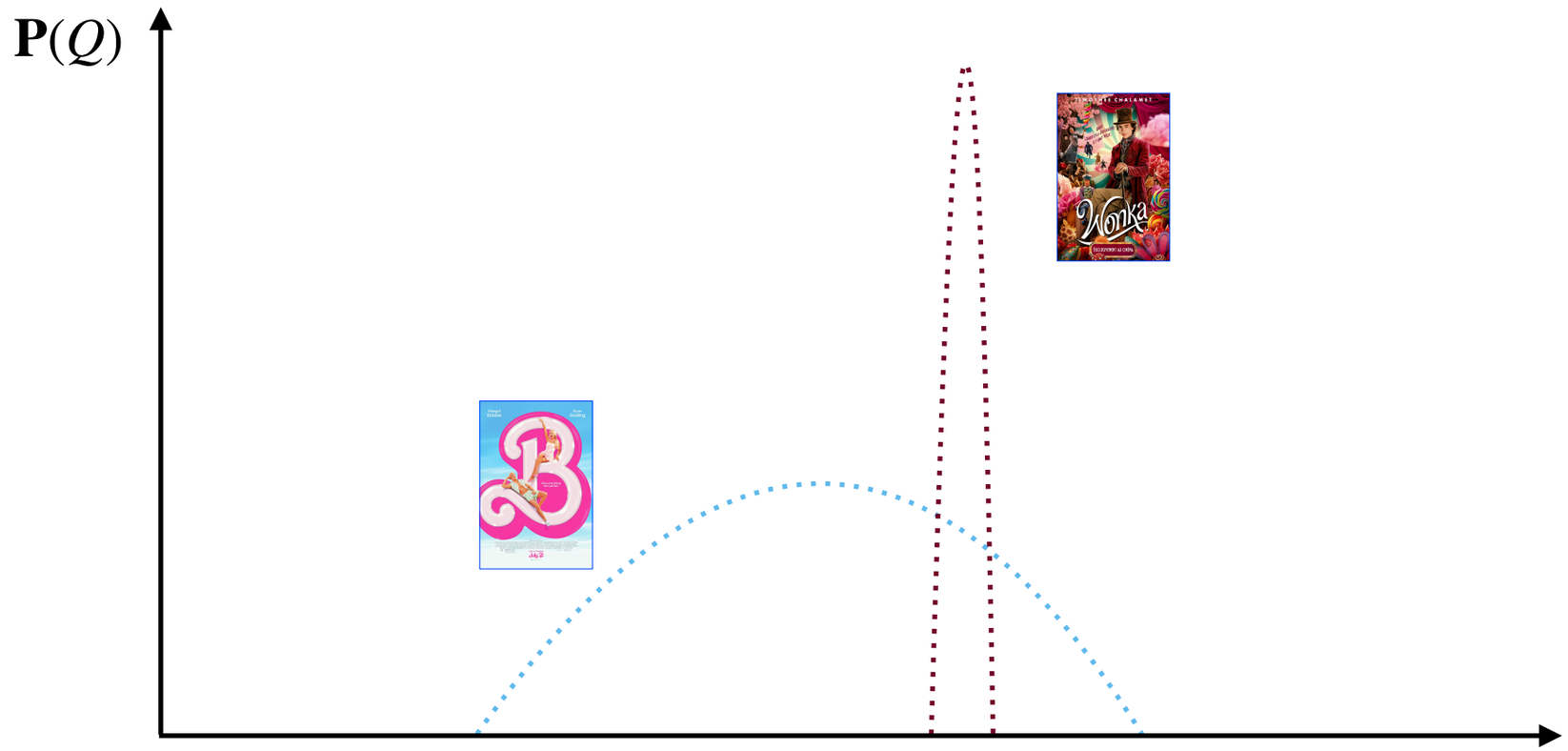
- With probability (1 - ϵ), select the best action $a^* = \arg\max_{a \in A} \hat{Q}_t(a)$
- With probability ϵ, select a random action



(Additional) Drawbacks.

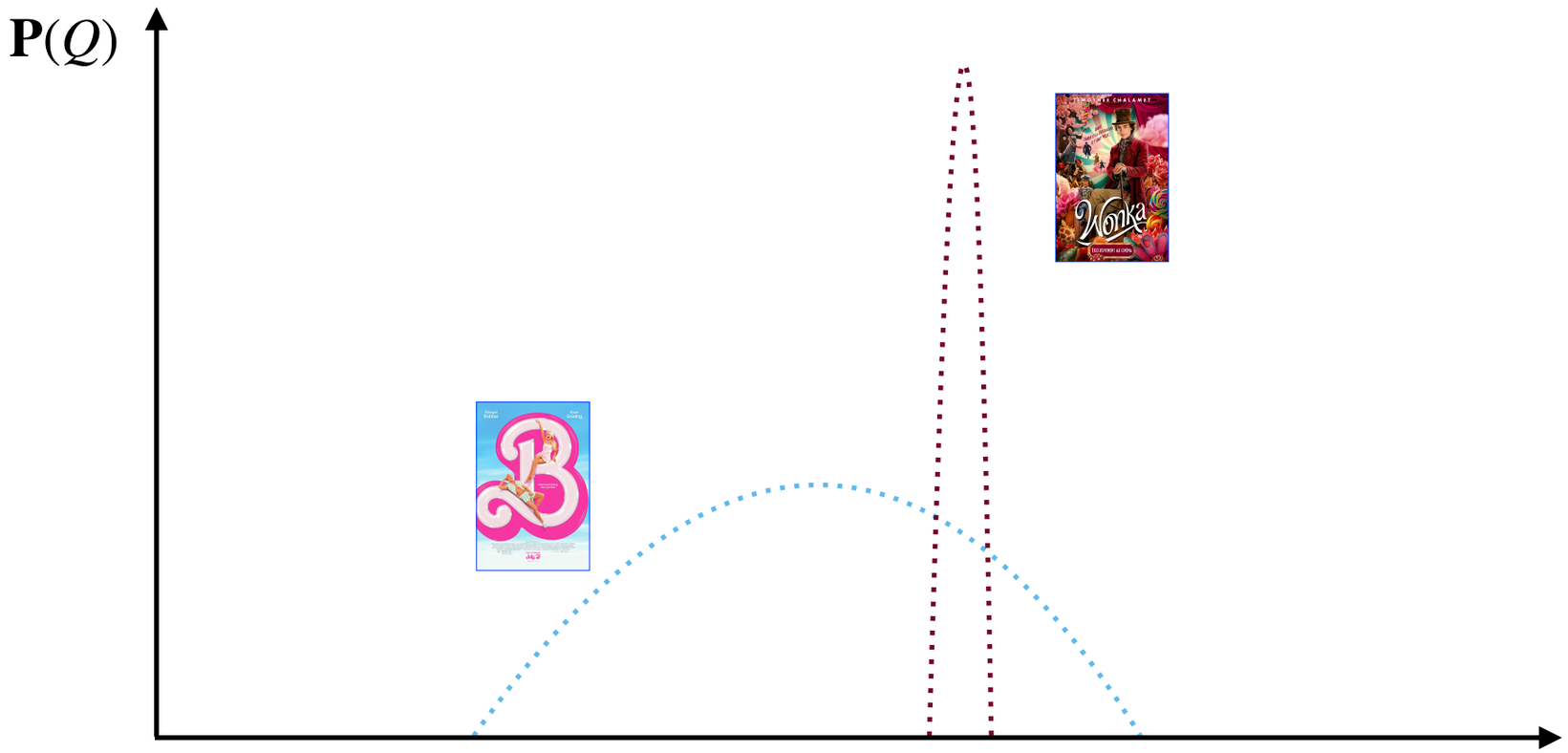
- It explores forever and has a linear regret
- It treats all suboptimal actions equally

Optimism in the face of uncertainty



Which action should we choose?

Optimism in the face of uncertainty



Which action should we choose?

Idea. The more we are uncertain about an action, the more we should explore it, given that it may turn out to be the best action.

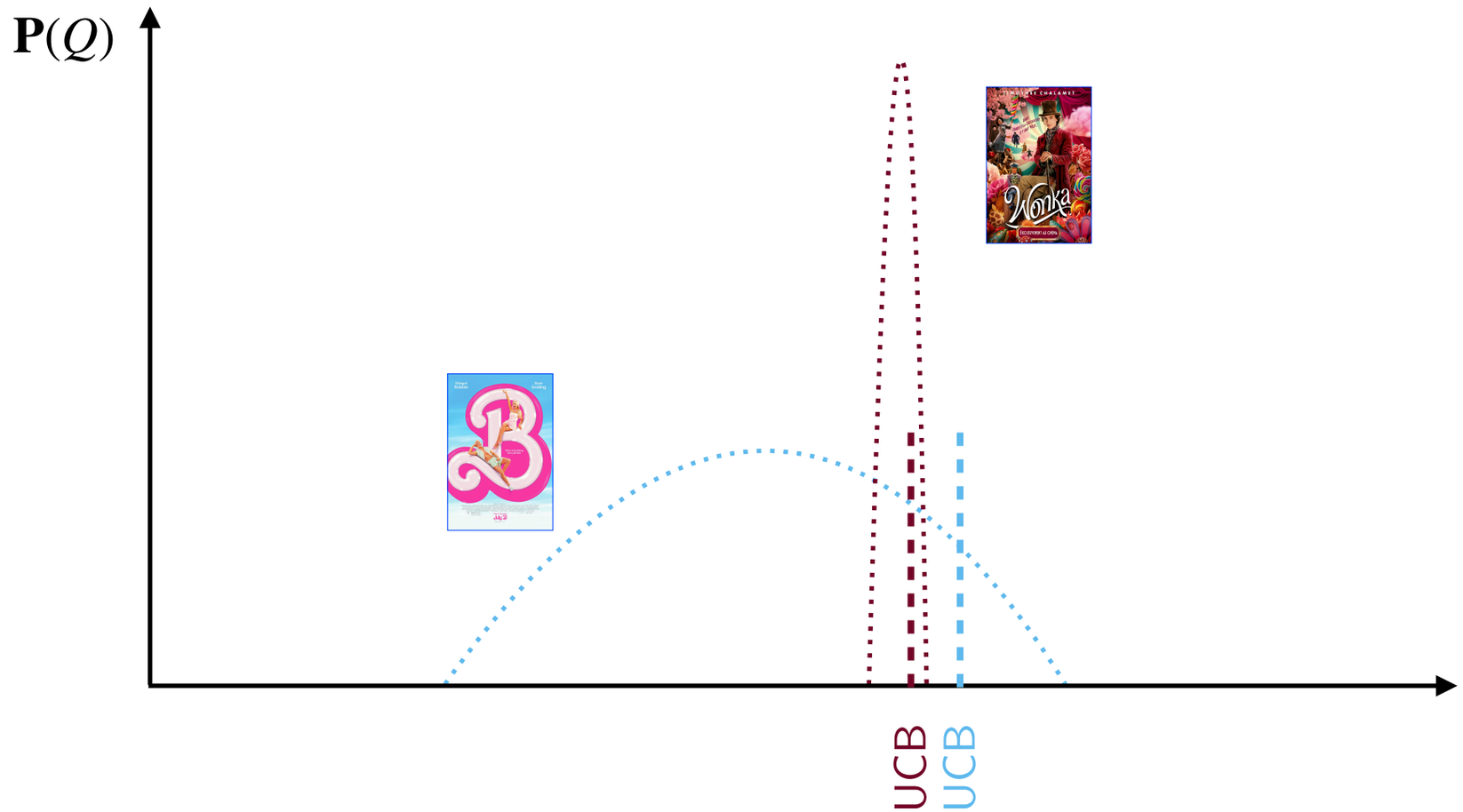
Upper Confidence Bound (UCB)

Idea.

- Estimate an upper confidence $\hat{U}_t(a)$ for each action value, such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with a high probability
- $\hat{U}_t(a)$ depends on the number of times action a has been selected, *i.e.*, $N_t(a)$
  - Small $N_t(a)$ implies large $\hat{U}_t(a)$, since the estimated value is uncertain
  - Large $N_t(a)$ implies small $\hat{U}_t(a)$, since the estimated value is accurate
- Select the action maximizing the Upper Confidence Bound (UCB)

$$a^* = \arg\max_{a \in A} \hat{Q}_t(a) + \hat{U}_t(a)$$

Upper Confidence Bound (UCB)



Hoeffding's inequality

How to compute the UCB?

Hoeffding's inequality

How to compute the UCB?

Theorem. *Let* $X_1, \ldots, X_t$ *be i.i.d. random variables in* $[0,1]$*, and let* $\bar{X}_t = \frac{1}{\tau}\sum_{\tau=1}^{t} X_\tau$ *be the sample mean. Then,*

$$\mathbf{P}\left[\mathbf{E}[X] > \bar{X}_t + u\right] \leq e^{-2tu^2}$$

Hoeffding's inequality

How to compute the UCB?

Theorem. *Let $X_1, \ldots, X_t$ be i.i.d. random variables in $[0,1]$, and let $\bar{X}_t = \frac{1}{\tau}\sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then,*

$$\mathbf{P}\left[\mathbf{E}[X] > \bar{X}_t + u\right] \leq e^{-2tu^2}$$

- We apply Hoeffding's inequality to rewards of the bandits conditioned on selecting action a:

$$\mathbf{P}\left[Q(a) > \hat{Q}_t(a) + U_t(a)\right] \leq e^{-2N_t(a)U_t(a)^2}$$

Computing UCBs

Pick a probability p that the true value exceeds the UCB, and solve for $U_t(a)$

$$e^{-2N_t(a)U_t(a)^2} = p$$

$$U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$$

Computing UCBs

Pick a probability p that the true value exceeds the UCB, and solve for $U_t(a)$

$$e^{-2N_t(a)U_t(a)^2} = p$$

$$U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$$

- Reduce p as we observe more rewards, *e.g.*, $p = t^{-4}$

$$U_t(a) = \sqrt{\frac{2\log p}{N_t(a)}}$$

UCB1

Algorithm.

- Initialize for all actions a:
  - $N(a) \leftarrow 0$
  - $Q(a) \leftarrow 0$
- For $t = 1, 2, \ldots$:
  - $a \leftarrow \arg\max_{a \in A} \left(Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}} \right)$
  - Observe reward r
  - $N(a) \leftarrow N(a) + 1$
  - $Q(a) \leftarrow Q(a) + \frac{1}{N(a)} \Big(r - Q(a) \Big)$

UCB vs. TS

Upper Confidence Bound (UCB).

- Deterministic
- Requires update at every round

Thompson Sampling (TS).

- Probabilistic
- Can accommodate delayed feedback
- Better empirical evidence

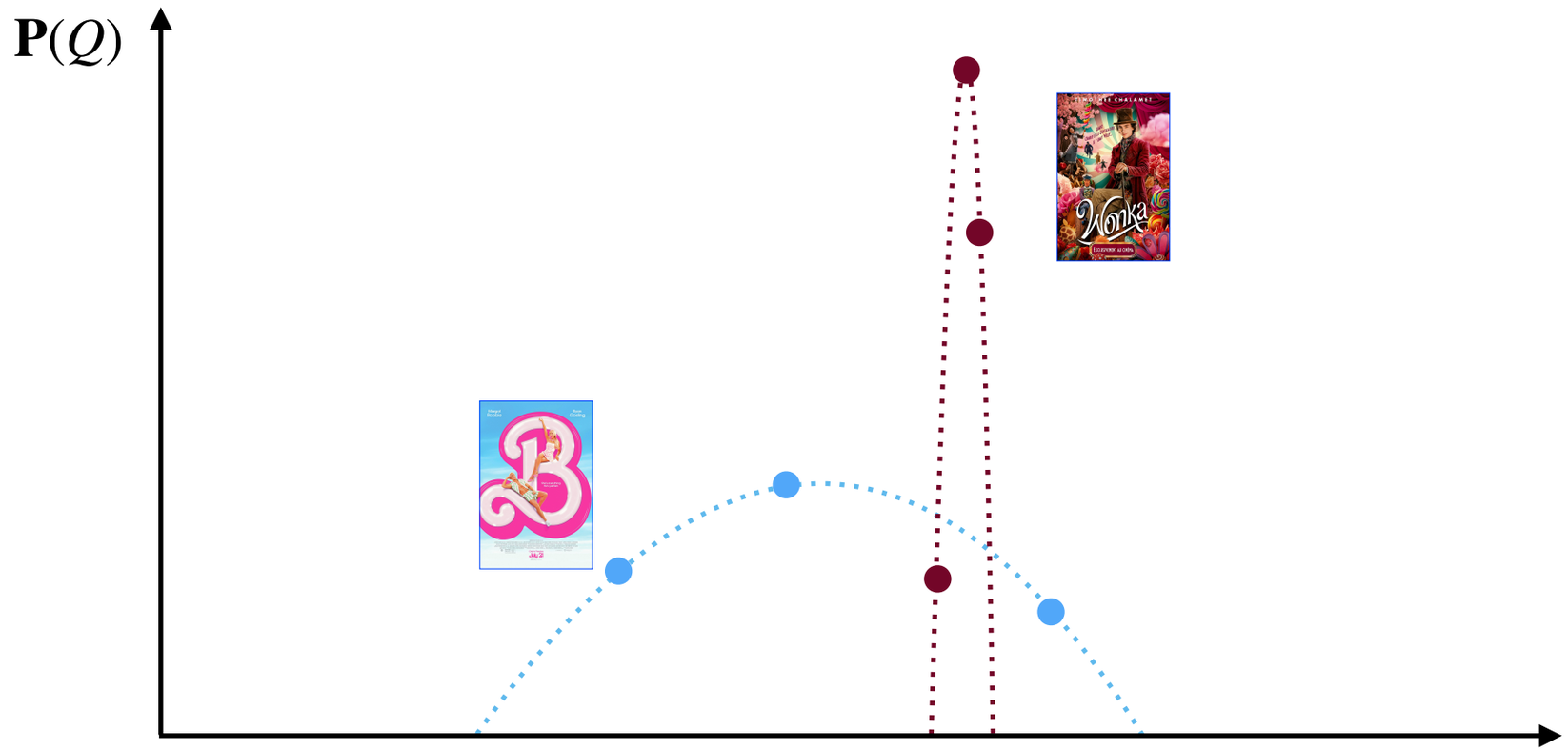
Thompson Sampling (TS)

- Implement probability matching:

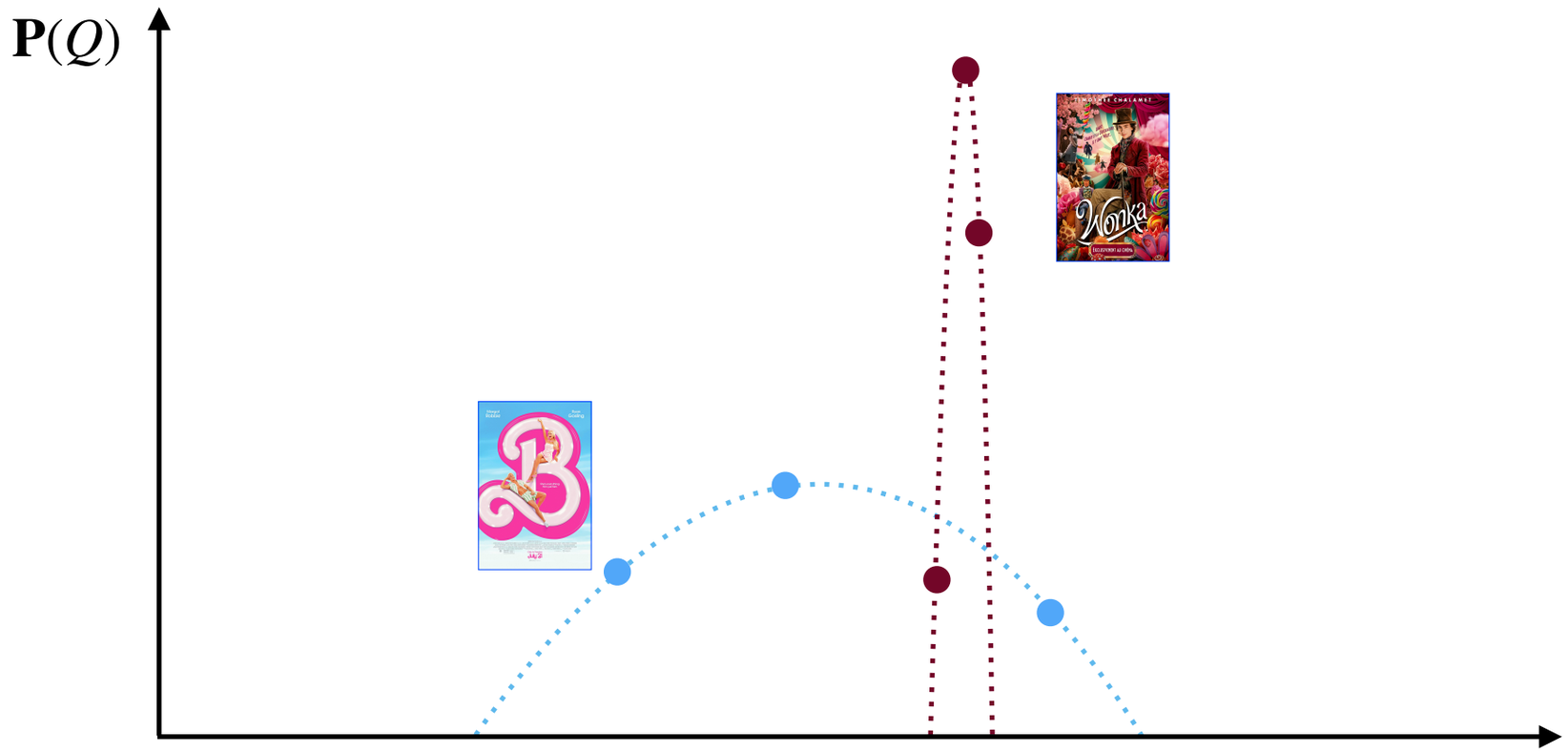
  Select action a according to its probability of being the optimal action

$$\pi(a \mid h_t) = \mathbf{P}\left[Q(a) > Q(a'), \forall a' \neq a \mid h_t\right]$$

Thompson Sampling (TS)



Thompson Sampling (TS)



Exploration is ensured by the prior;

Exploitation is ensured by the concentration of the posterior.

Beta-Bernoulli example

Binary reward; probability of success of action a_k is θ_k

Algorithm.

- Initialize for all actions a_k:
  - Initialize α_k and β_k
- For $t = 1, 2, \ldots$:
  - $\theta_k \sim beta(\alpha_k, \beta_k)$ for all actions a_k where $k = 1, \ldots, K$
  - $a \leftarrow \arg\max_k \theta_k$
  - Observe reward r
  - $(\alpha_k, \beta_k) \leftarrow (\alpha_k + r, \beta_k + 1 - r)$

Multi-armed bandits for recommendation

Agent following the process described below:

- Enter the world with zero knowledge
- At each time step t,
  - Pick an action $a_t \in A$
  - Observe a reward $r_t \sim R^{a_t}$

Goal: Maximize cumulative reward $\sum_{t=1}^{T} r_t$

Contextual bandits for recommendation

Agent following the process described below:

- Enter the world with zero knowledge
- At each time step t,
  - Observe a context x_t
  - Pick an action $a_t \sim \pi(A \mid x_t)$
  - Observe a reward $r_t \sim R_{x_t}^{a_t}$

Goal: Maximize cumulative reward $\sum_{t=1}^{T} r_t$

Practical considerations of RL applied to RecSys

Real-World Applications

Application of ϵ-greedy

- "Explore, Exploit, Explain" at Spotify

  McInerney et al. *Explore, Exploit, and Explain: Personalizing Explainable Recommendations with Bandits*. In Proc. RecSys, 2018.

Real-World Applications

Application of ϵ-greedy

- "Explore, Exploit, Explain" at Spotify

  McInerney et al. *Explore, Exploit, and Explain: Personalizing Explainable Recommendations with Bandits*. In Proc. RecSys, 2018.

Application of UCB

- LinUCB at Yahoo for news recommendation

  Li et al. *A Contextual-Bandit Approach to Personalized News Article Recommendation*. In Proc. WWW, 2010.

- UBM-UCB at Alibaba for item recommendation

  He et al. *Contextual User Browsing Bandits for Large-Scale Online Mobile Recommendation*. In Proc. RecSys, 2020.

Real-World Applications

Application of ϵ-greedy

- "Explore, Exploit, Explain" at Spotify

  McInerney et al. *Explore, Exploit, and Explain: Personalizing Explainable Recommendations with Bandits*. In Proc. RecSys, 2018.

Application of UCB

- LinUCB at Yahoo for news recommendation

  Li et al. *A Contextual-Bandit Approach to Personalized News Article Recommendation*. In Proc. WWW, 2010.

- UBM-UCB at Alibaba for item recommendation

  He et al. *Contextual User Browsing Bandits for Large-Scale Online Mobile Recommendation*. In Proc. RecSys, 2020.

Application of Thompson Sampling

- TS at Amazon for layout optimization

  Hill et al. *An Efficient Bandit Algorithm for Realtime Multivariate Optimization*. In Proc. KDD, 2017.

- TS at DoorDash for cuisine recommendation

  https://doordash.engineering/2020/01/27/personalized-cuisine-filter/

Exploration for new items

Exploration is used to collect information about new items and incorporate it in real-time into models

Examples of implementation:

- Strong exploration on a limited subset of users, e.g.,

  [Yahoo] Users randomly assigned to an exploration bucket that is served news articles at random

  Li et al. *A Contextual-Bandit Approach to Personalized News Article Recommendation*. In Proc. WWW, 2010.

  [Twitter] A random policy serves 1% of the production traffic

  Guo et al. *Deep Bayesian Bandits: Exploring in Online Personalized Recommendations*. In Proc. RecSys, 2020.

Exploration for new items

Exploration is used to collect information about new items and incorporate it in real-time into models

Examples of implementation:

- Strong exploration on a limited subset of users, e.g.,

  [Yahoo] Users randomly assigned to an exploration bucket that is served news articles at random

  Li et al. *A Contextual-Bandit Approach to Personalized News Article Recommendation*. In Proc. WWW, 2010.

  [Twitter] A random policy serves 1% of the production traffic

  Guo et al. *Deep Bayesian Bandits: Exploring in Online Personalized Recommendations*. In Proc. RecSys, 2020.

- Light exploration on all users, e.g.,

  [Netflix] Artwork personalization at Netflix

  https://netflixtechblog.com/artwork-personalization-c589f074ad76