Lecture 13: Beyond Single-Arm Actions

ID 6002W: Online and Reinforcement Learning

Web-enabled M.Tech. program, IIT Madras

Suryanarayana Sankagiri

23rd June, 2026

Recap

Simple bandit model

So far, our bandit model assumes:

- At each round, algorithm picks **one arm**
- Observes reward for that chosen arm only
- Updates belief about that arm
- Goal: maximise cumulative reward

This model includes both MAB and linear bandits

How Do Real Systems Work?

A recommender system typically offers a *slate* of items to the user at one time

Slate Bandits

What is a slate?

- $A_t = (a_{t,1}, \ldots, a_{t,K})$
  - chosen from a larger pool of N items
- the user reacts to this list
  - clicks on one/many/none of the shown items

Slate Bandits

What is a slate?

- $A_t = (a_{t,1}, \ldots, a_{t,K})$
  - chosen from a larger pool of N items
- the user reacts to this list
  - clicks on one/many/none of the shown items
- What the user clicks on, if at all, depends on the entire set of items
- Order also matters!

How To Model Bandits With Slate Actions?

Naive Option 1

Treat Every Slate As One Arm

- Arm = entire ordered list ($A = (a_1, \ldots, a_K)$)
- Observe one reward for the whole slate
  - e.g., did the user click on any shown item or not

Naive Option 1

Treat Every Slate As One Arm

- Arm = entire ordered list ($A = (a_1, \ldots, a_K)$)
- Observe one reward for the whole slate
  - e.g., did the user click on any shown item or not
- Suppose $N = 100,\ K = 5$. Then number of possible slates is $100 \times 99 \times 98 \times 97 \times 96 \ \approx 9$ billion!
  - Impractical to have MAB
    - Showing (A, B, C) tells us almost nothing about (A, B, D)

Naive Option 2

Pick Top K Items By UCB Scores

- For each item i, maintain $UCB_i(t) = \hat{\mu}_i(t) + \text{bonus}_i(t)$
- Normal bandits: pick item with best UCB score
- **Slate bandits**: pick top k items according to UCB scores

Naive Option 2

Pick Top K Items By UCB Scores

- For each item i, maintain $UCB_i(t) = \hat{\mu}_i(t) + \text{bonus}_i(t)$
- Normal bandits: pick item with best UCB score
- **Slate bandits**: pick top k items according to UCB scores
- Update rule:
    - Item i shown and clicked: reward = 1
    - Item j shown and not clicked: reward = 0
- This is simple and scalable

Individual Scoring Is Not Enough

Position Bias

1 Same item at the top

Position 1
CTR: 8%

2 Same item at the bottom

Position 5
CTR: 2%

3 How users browse a ranked list

Same item, different position — different click-through rate

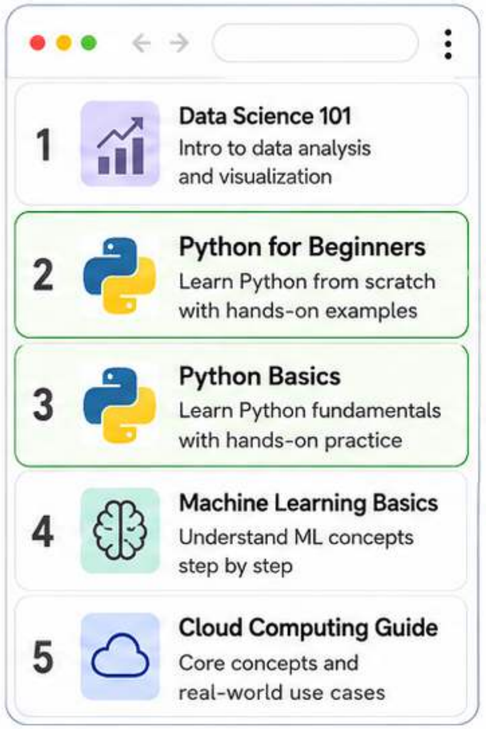
Individual Scoring Is Not Enough

Context Effects

1 Diverse list

CTR: 8%

2 Add a very similar item

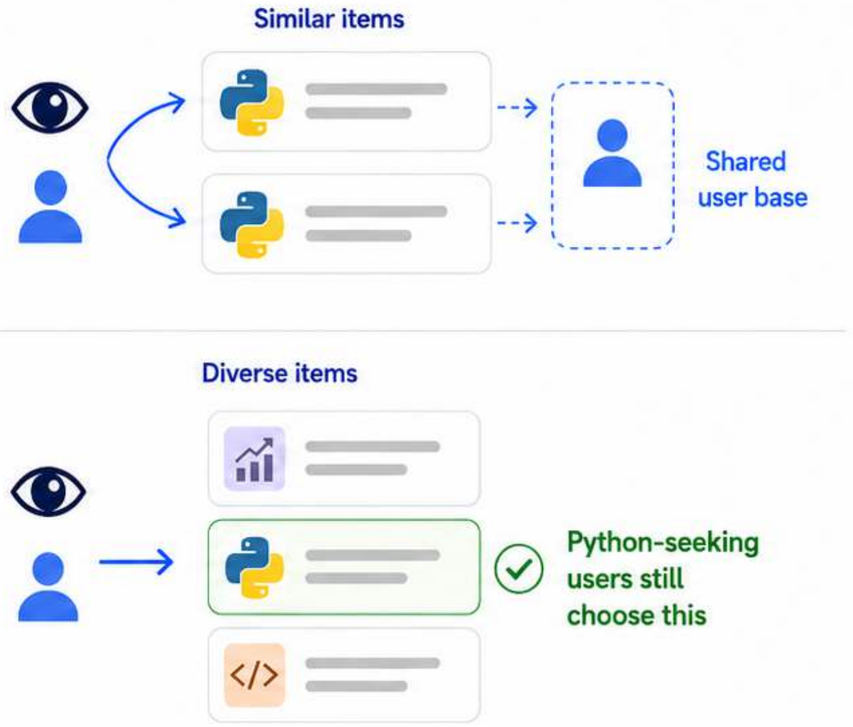


CTR: 5%

Similar items compete (views split)

CTR: 3%

3 Why this happens



An item's click-through rate depends on its context

The Challenge

- In the first solution, actions were entire slates
  - ✓ Position bias, context effects, etc. are automatically accounted
  - - But number of arms is humongous!
  - ‣ Need additional structure to share data across similar arms (slates)

The Challenge

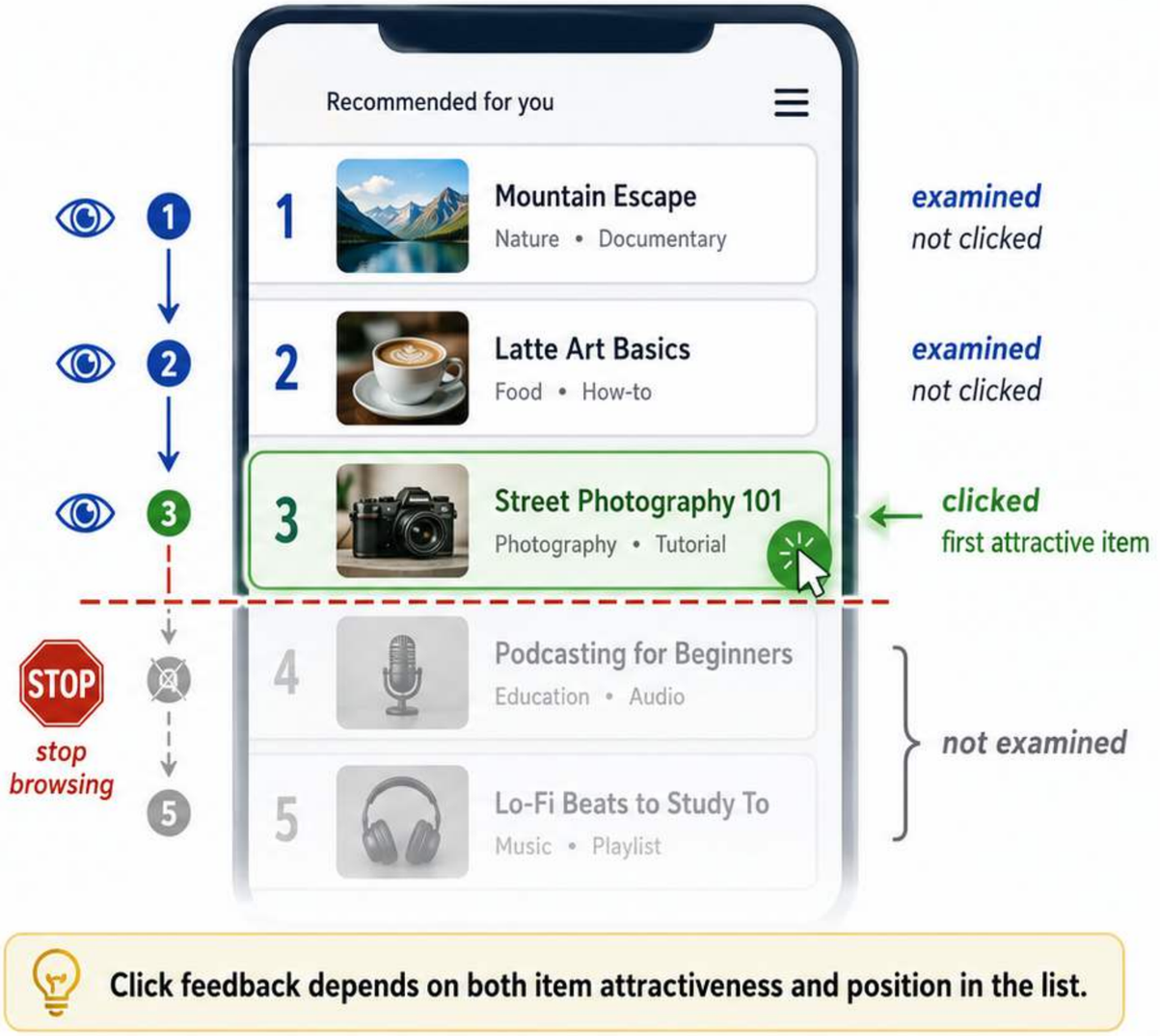
- In the first solution, actions were entire slates
  - ✓ Position bias, context effects, etc. are automatically accounted
  - – But number of arms is humongous!
  - ‣ Need additional structure to share data across similar arms (slates)
- In the second solution, we maintain scores for each item
  - ✓ Tractable, especially using linear bandits framework
  - – Individual scoring misses list-wise effects
  - ‣ Need to model how users scan lists

The Cascade Click Model

A Simple Behavioural Model

For ranked recommendations

- Users scan items from top to bottom
- The user clicks the first attractive item
- After clicking, the user stops browsing

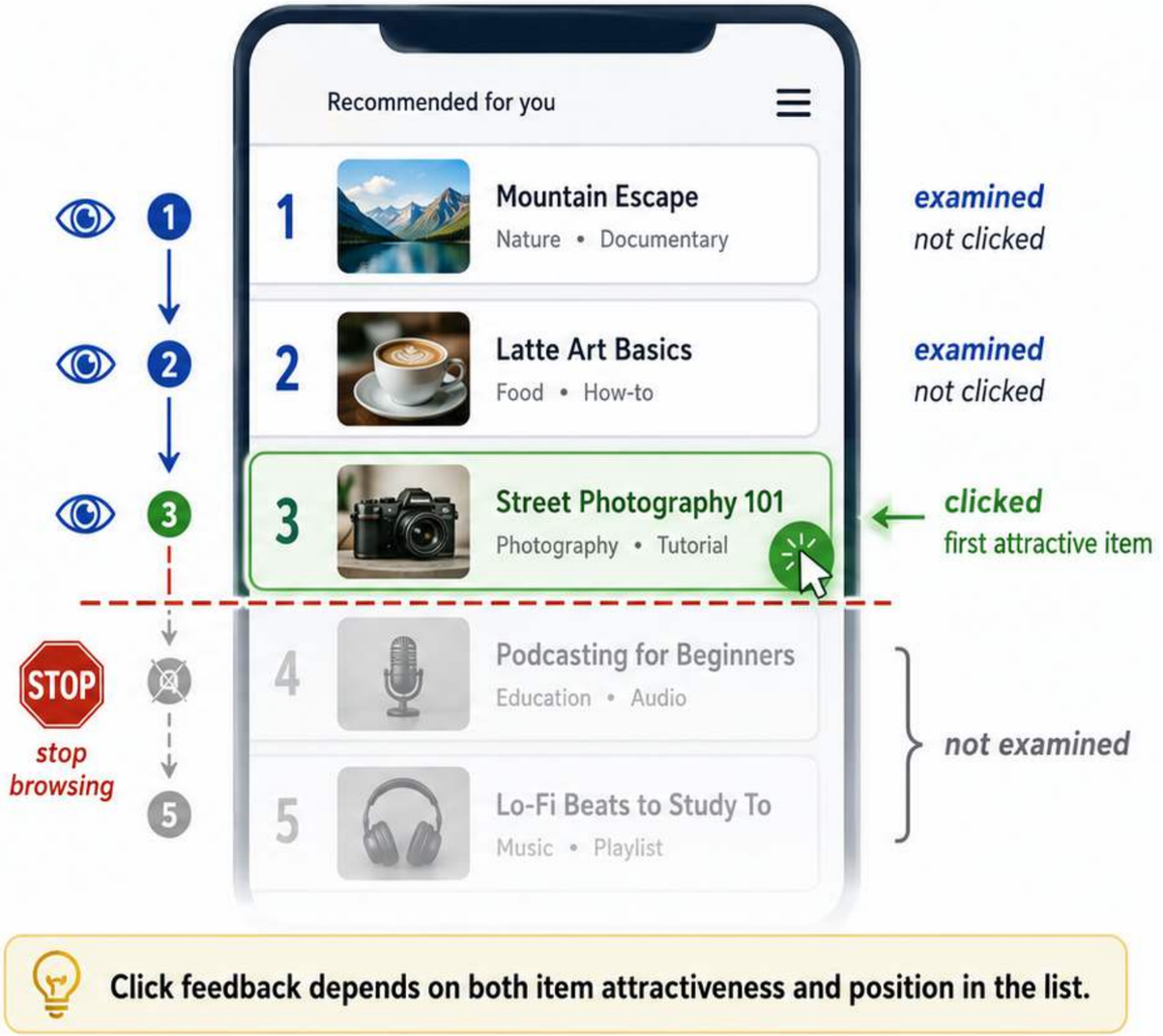


A Simple Behavioural Model

For ranked recommendations

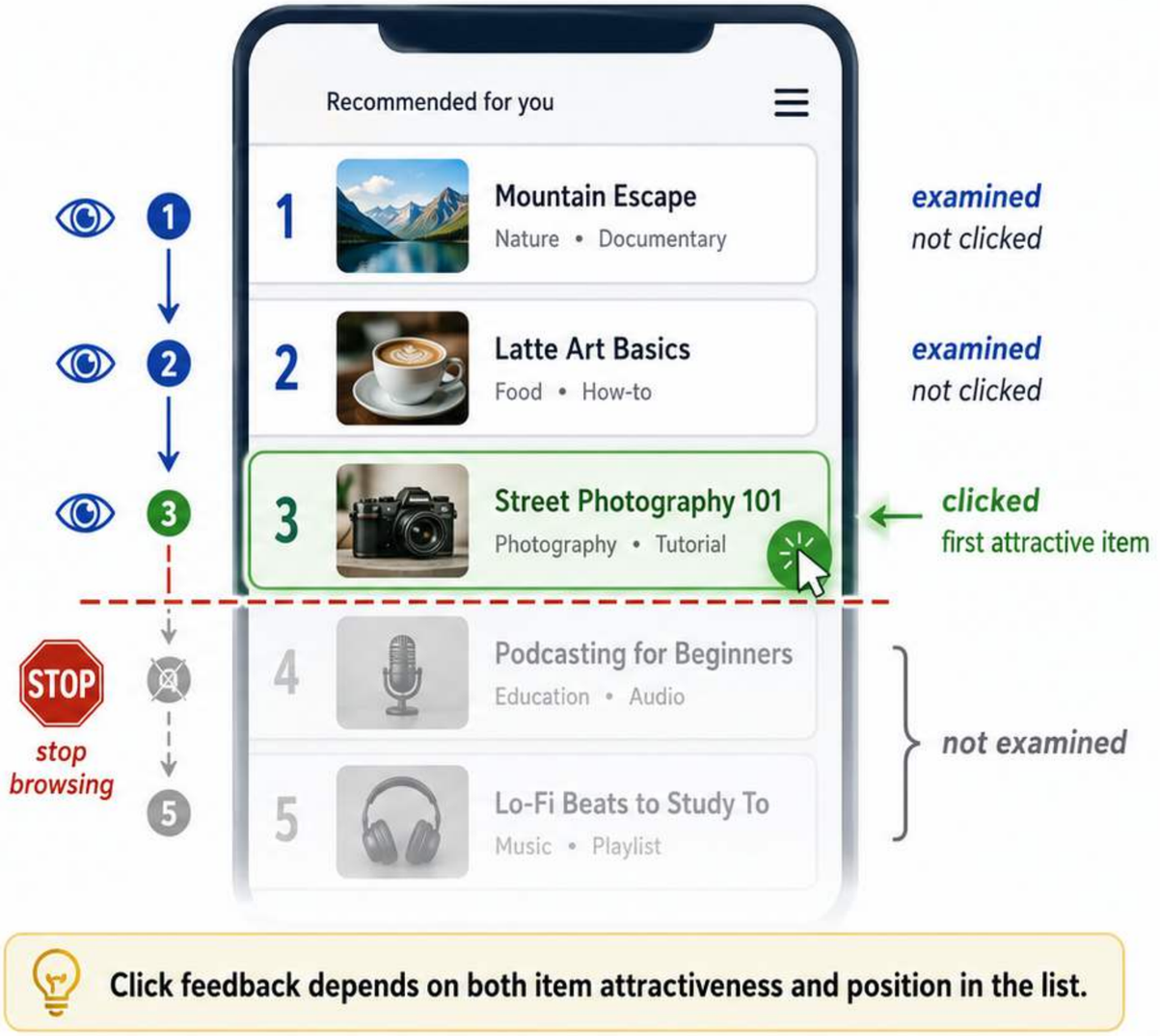
- Users scan items from top to bottom
- The user clicks the first attractive item
- After clicking, the user stops browsing
- A click tells us the item is attractive

➡ *But an item not clicked?*



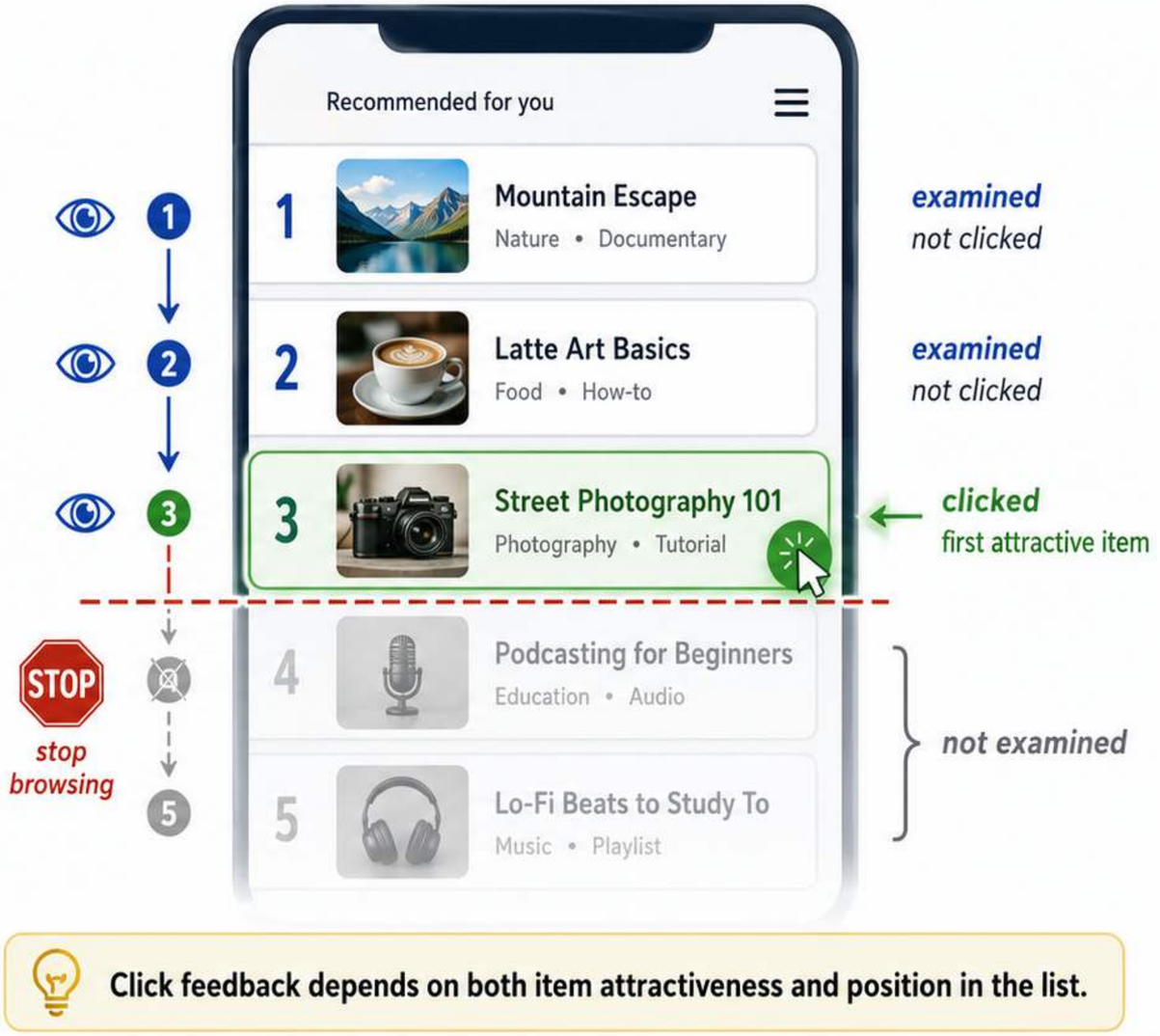
What Does One Click Tell Us?

- Items above click: seen and rejected
  - Can set reward to 0



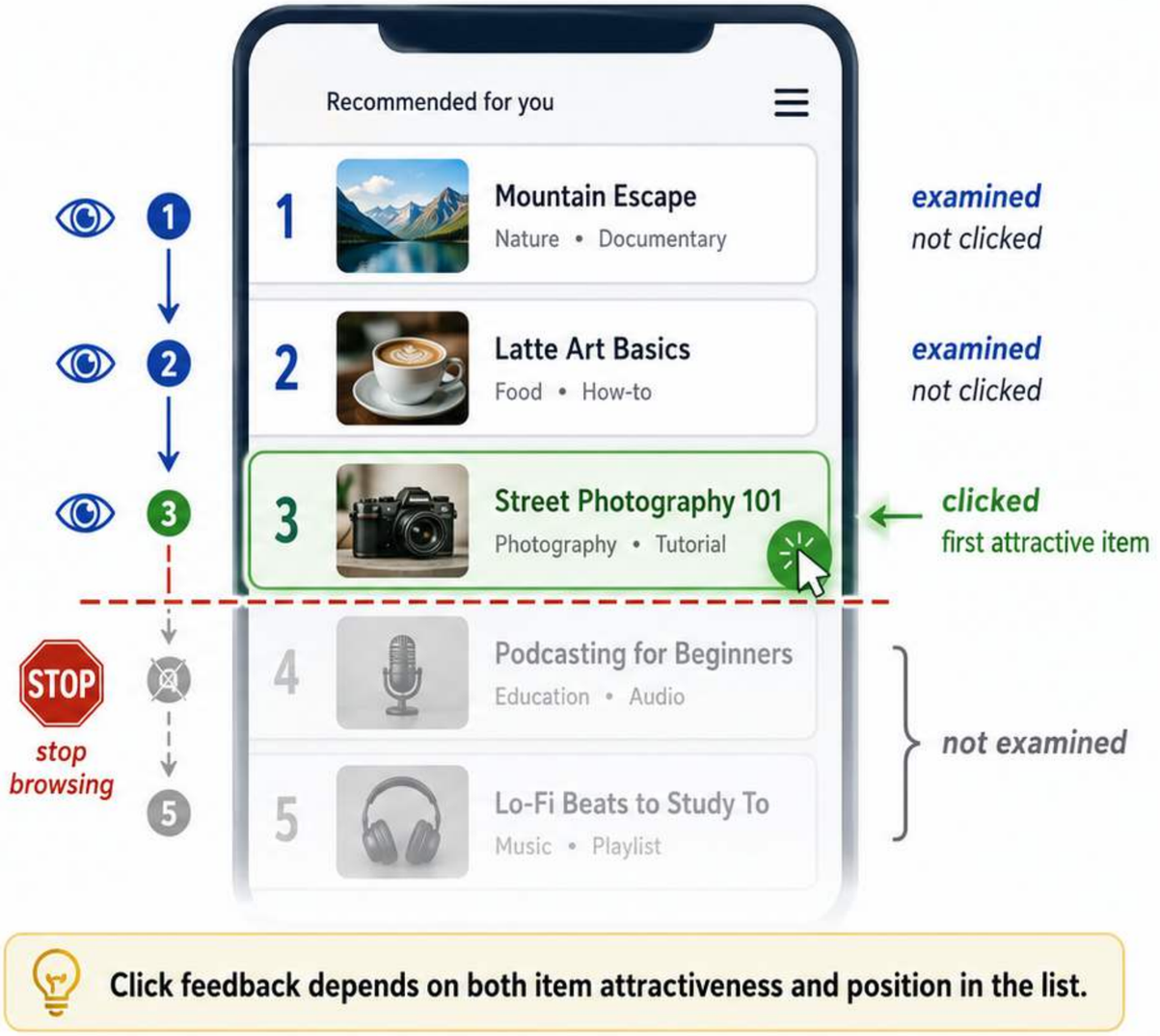
What Does One Click Tell Us?

- Items above click: seen and rejected
  - Can set reward to 0
- Clicked item: seen and accepted
  - Can set reward to 1



What Does One Click Tell Us?

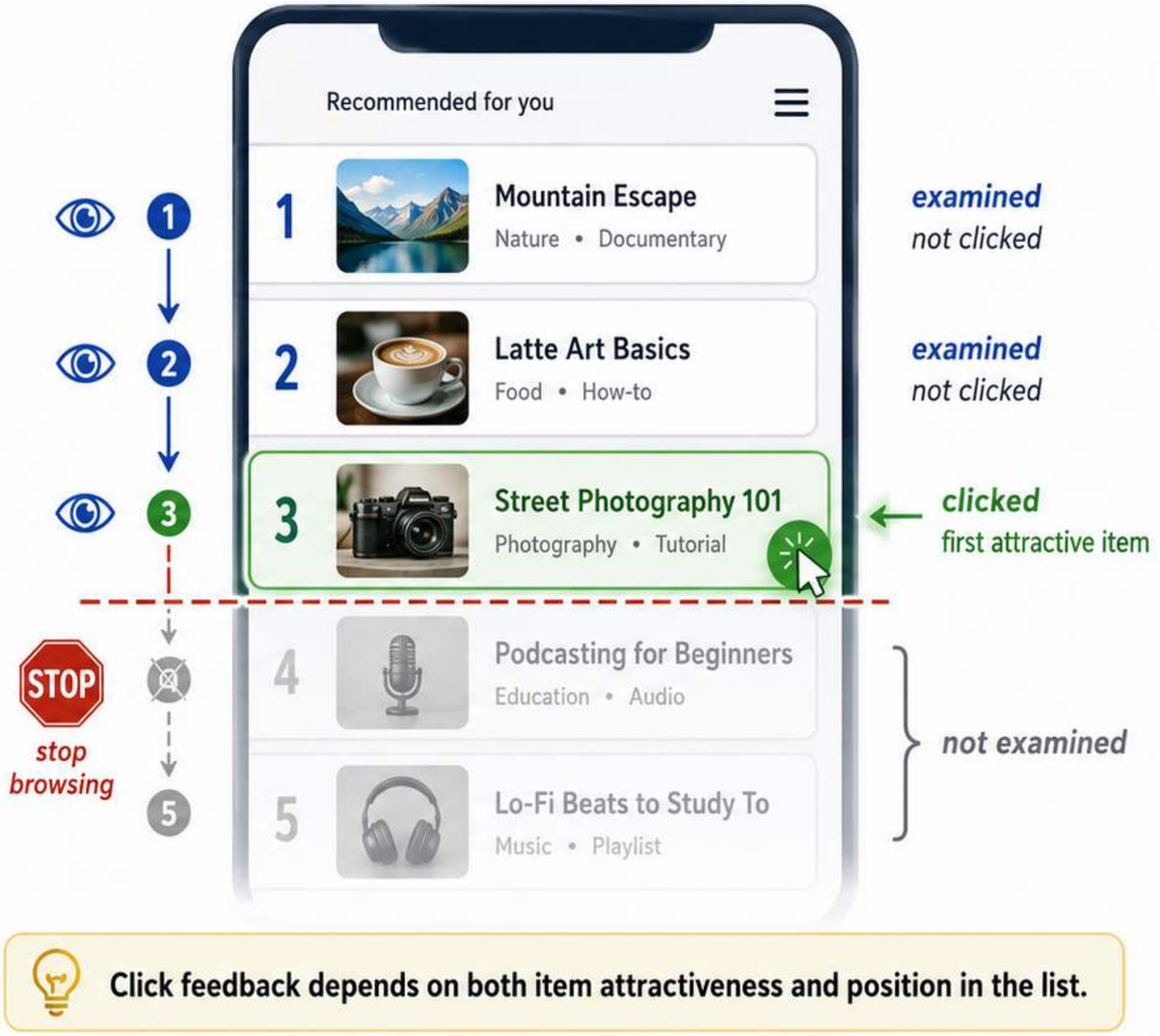
- Items above click: seen and rejected
  - Can set reward to 0
- Clicked item: seen and accepted
  - Can set reward to 1
- Items below click: unseen!
  - Can’t set any reward for them



What Does One Click Tell Us?

- Items above click: seen and rejected
  - Can set reward to 0
- Clicked item: seen and accepted
  - Can set reward to 1
- Items below click: unseen!
  - Can’t set any reward for them

No click is useful feedback!
It gives scores for entire slate of items



Modeling Click Probabilities

- Assume each item i has attractiveness w_i
  - w_i : probability that you click on the item i when shown on top

Modeling Click Probabilities

- Assume each item i has attractiveness w_i
  - w_i : probability that you click on the item i when shown on top
- Probability that you do not click on any item:

$$\mathbb{P}(\text{no click}) = \prod_{j=1}^{K} (1 - w_{a_j})$$

Modeling Click Probabilities

- Assume each item i has attractiveness w_i
  - w_i : probability that you click on the item i when shown on top
- Probability that you do not click on any item:

$$\mathbb{P}(\text{no click}) = \prod_{j=1}^{K} (1 - w_{a_j})$$

- Probability of at least one click: $1 - \prod_{j=1}^{K} (1 - w_{a_j})$

Modeling Click Probabilities

- Assume each item i has attractiveness w_i
  - w_i : probability that you click on the item i when shown on top
- Probability that you do not click on any item:

$$\mathbb{P}(\text{no click}) = \prod_{j=1}^{K} (1 - w_{a_j})$$

- Probability of at least one click: $1 - \prod_{j=1}^{K} (1 - w_{a_j})$
- To maximise clickthrough rate, choose top k items with highest scores

But we do not have the item scores! We need to estimate them from data

The CascadeUCB Algorithm

- For each item i, maintain $\hat{w}_i(t)$, $N_i(t)$

$$UCB_i(t) = \hat{w}_i(t) + \sqrt{\frac{c \log T}{N_i(t)}}$$

The CascadeUCB Algorithm

- For each item i, maintain $\hat{w}_i(t)$, $N_i(t)$

$$UCB_i(t) = \hat{w}_i(t) + \sqrt{\frac{c \log T}{N_i(t)}}$$

- **Decision:** top K items by UCB scores

The CascadeUCB Algorithm

- For each item i, maintain $\hat{w}_i(t)$, $N_i(t)$

$$UCB_i(t) = \hat{w}_i(t) + \sqrt{\frac{c \log T}{N_i(t)}}$$

- **Decision:** top K items by UCB scores
- **Observe:** Click position (or no click)

The CascadeUCB Algorithm

- For each item i, maintain $\hat{w}_i(t)$, $N_i(t)$

$$UCB_i(t) = \hat{w}_i(t) + \sqrt{\frac{c \log T}{N_i(t)}}$$

- **Decision:** top K items by UCB scores
- **Observe:** Click position (or no click)
- **Update:** $A_t = (a, b, c, d, e) \quad C_t = 3 \quad \Rightarrow \; a \leftarrow 0,\; b \leftarrow 0,\; c \leftarrow 1,\; d, e$ unchanged

The CascadeUCB Algorithm

- For each item i, maintain $\hat{w}_i(t)$, $N_i(t)$

$$UCB_i(t) = \hat{w}_i(t) + \sqrt{\frac{c \log T}{N_i(t)}}$$

- **Decision:** top K items by UCB scores
- **Observe:** Click position (or no click)
- **Update:** $A_t = (a, b, c, d, e) \quad C_t = 3 \quad \Rightarrow \; a \leftarrow 0,\; b \leftarrow 0,\; c \leftarrow 1,\; d, e$ unchanged
- **CascadeUCB** = UCB + cascade feedback model

Pros and Cons of Cascade Model

✓ The cascade model is an effective way of accounting for position bias

✓ Allows us to make a simple tweak to the UCB algorithm

✓ Computationally efficient

Pros and Cons of Cascade Model

✓ The cascade model is an effective way of accounting for position bias

✓ Allows us to make a simple tweak to the UCB algorithm

✓ Computationally efficient

- But it does not account for context effects
  - How the presence of similar items can divide traffic
  - How diverse items make individual items appear more distinct

The MNL Bandit Model

Captures Competition Between Items

- At each round, platform offers a slate: $A_t = (a_1, \ldots, a_K)$
- Each item i has unknown attractiveness $v_i > 0$

The MNL Bandit Model

Captures Competition Between Items

- At each round, platform offers a slate: $A_t = (a_1, \ldots, a_K)$
- Each item i has unknown attractiveness $v_i > 0$
- User chooses one item, or nothing:

$$\mathbb{P}(i \mid A_t) = \frac{v_i}{1 + \sum_{j \in A_t} v_j}$$

$$\mathbb{P}(\text{no click} \mid A_t) = \frac{1}{1 + \sum_{j \in A_t} v_j}$$

The MNL Bandit Model

Captures Competition Between Items

- At each round, platform offers a slate: $A_t = (a_1, \ldots, a_K)$
- Each item i has unknown attractiveness $v_i > 0$
- User chooses one item, or nothing:

$$\mathbb{P}(i \mid A_t) = \frac{v_i}{1 + \sum_{j \in A_t} v_j}$$

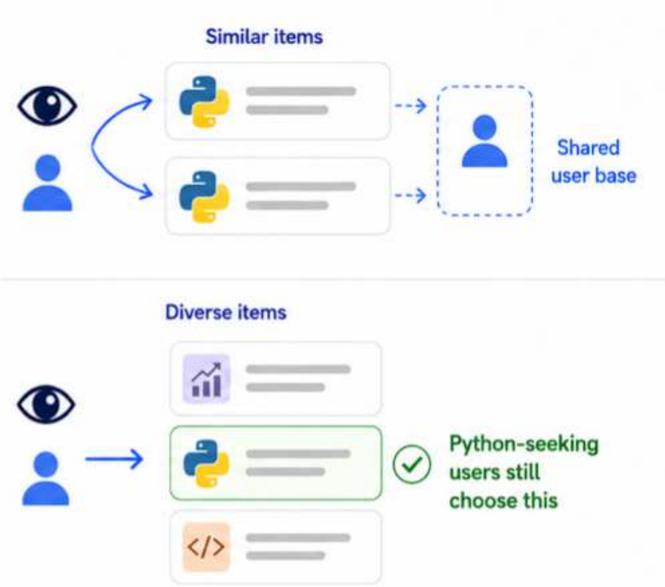
$$\mathbb{P}(\text{no click} \mid A_t) = \frac{1}{1 + \sum_{j \in A_t} v_j}$$

- Including more good items captures ‘market share’
- Used more in marketing literature, but also relevant to slate bandits

Understanding Context Effects

There are well-studied phenomena about context effects:

- **Similarity effect**: near-duplicates split the same user base



Open Problem!

Understanding Context Effects

There are well-studied phenomena about context effects:

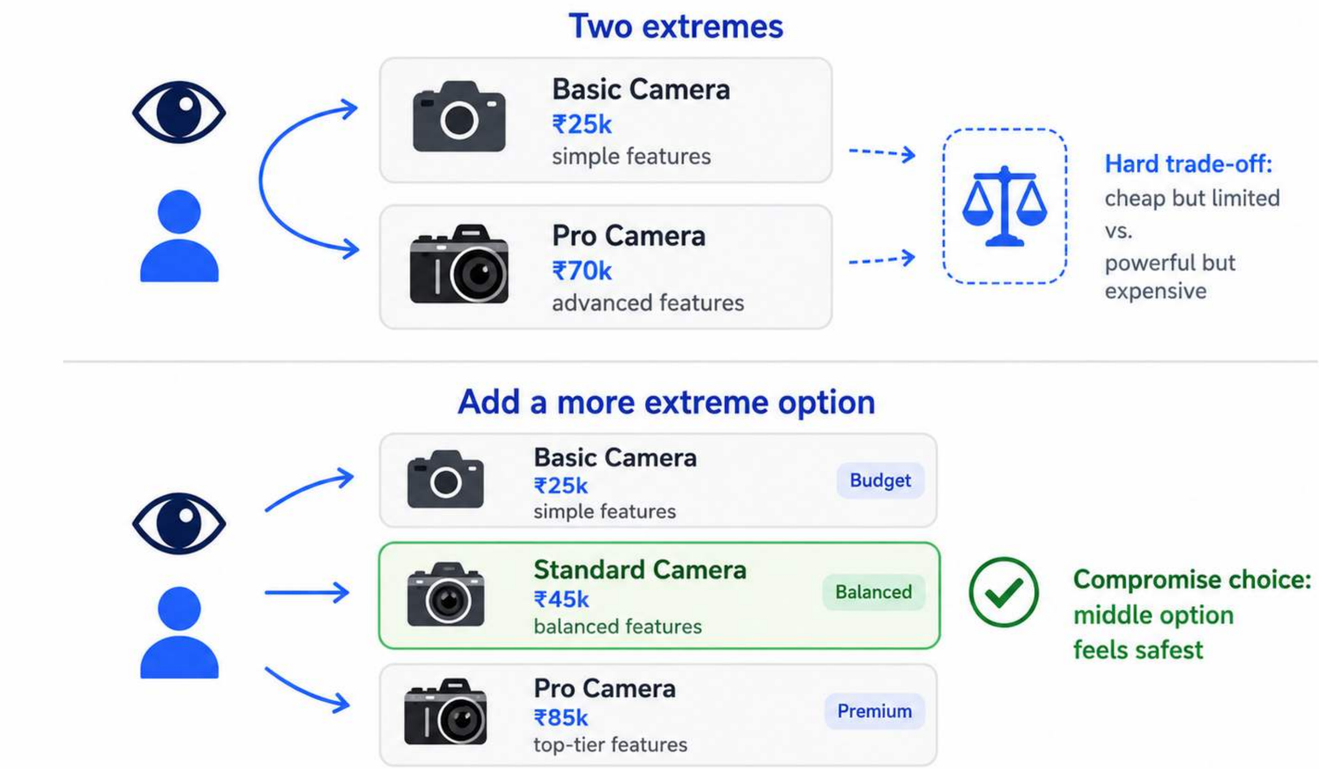
- **Similarity effect**: near-duplicates split the same user base
- **Decoy effect**: an inferior option can make another option look better

Open Problem!

Understanding Context Effects

There are well-studied phenomena about context effects:

- **Similarity effect**: near-duplicates split the same user base
- **Decoy effect**: an inferior option can make another option look better
- **Compromise effect:** users may prefer the “middle” option among extremes

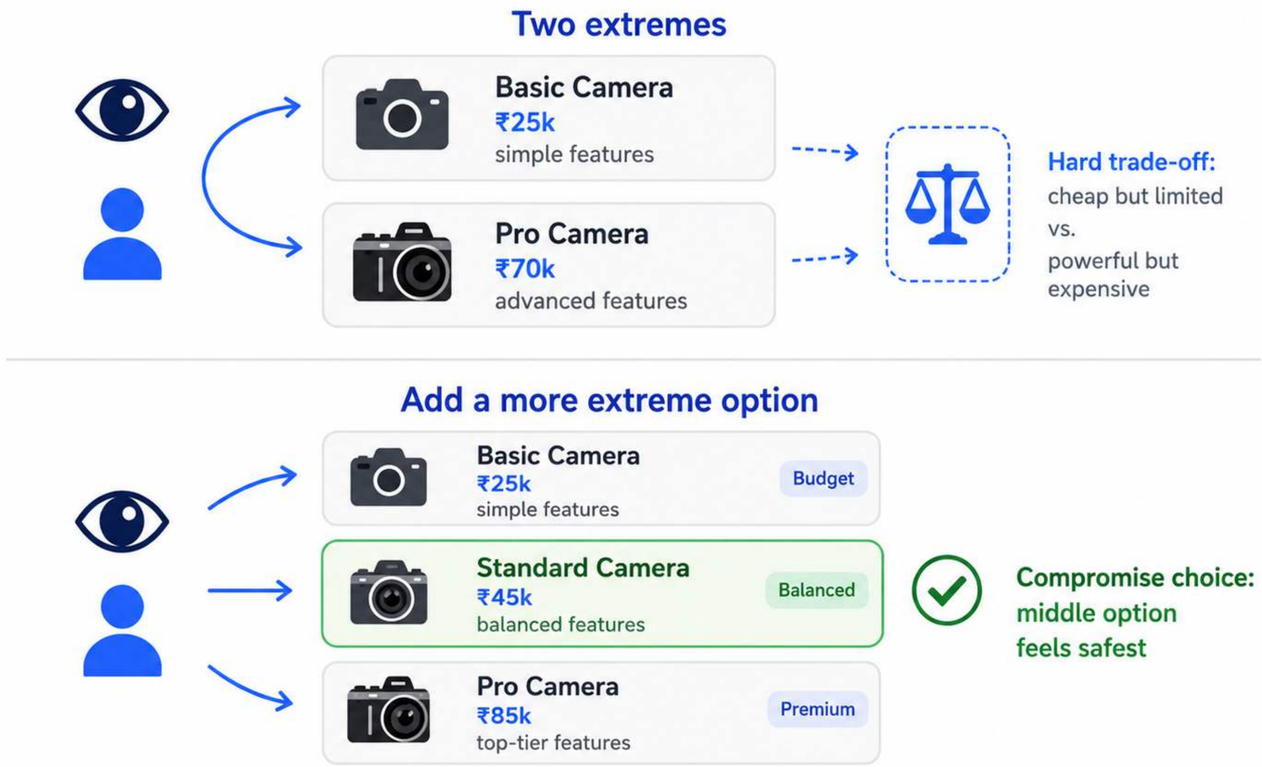


Open Problem!

Understanding Context Effects

There are well-studied phenomena about context effects:

- **Similarity effect**: near-duplicates split the same user base
- **Decoy effect**: an inferior option can make another option look better
- **Compromise effect:** users may prefer the “middle” option among extremes



Modelling context effects requires moving away from item-wise scores to scoring the whole list $V(a_1, \ldots, a_K)$

Open Problem!

From Clicks to Preferences

- Cascade/MNL models still assume some observable click/choice signal
- Clicks are convenient and plentiful, but not always meaningful
  - A user may click something and dislike it
- For subjective items, pairwise comparison is often easier than rating

Instead of “what rating would you like to give item i?”,

Ask, “which of i and j do you prefer?”

Dueling Bandits

Bandits with pairwise preference feedback

- Choose two arms (i, j)
- Observe preference feedback $i \succ j$ or $j \succ i$
- Quantity to estimate: $P(i \succ j)$
- No numerical reward, only noisy pairwise preference
- Learn which arms beat which other arms
- Goal: identify a “winner” arm (Condorcet winner):

$$i^* : P(i^* \succ j) > 1/2 \ \forall \ j \neq i^*$$

Learning From Preferences Use-Case

LLM Alignment

1 Prompt

Examples of user prompts

How can I learn Python quickly?

Write a polite email asking for an extension.

What should I do if I feel overwhelmed?

2 Candidate responses

Model generates two responses for the prompt.

Prompt: How can I learn Python quickly?

A

Response A

Start with the basics: variables, data types, and control flow. Practice with small exercises and build projects. Use resources like the official tutorial and interactive coding platforms. Consistency is key!

helpful | clear | safe

B

Response B

Just copy code from online and you'll be fine. There's no need to understand it. Python is easy anyway.

harmful | rude | misleading

3 Preference data

People compare the responses and choose the better one.

A > B

Preference dataset (many examples)

A	vs B	A ✓
A	vs B	B ✓
A	vs B	A ✓
...		...

Training data = prompt + response A + response B + preference label (A or B)

4 Model updated

The model learns from these preferences.

Learns to give more helpful and less harmful answers.

New prompt (unseen during training)

What are some tips for staying focused while studying?

Model's improved response

Create a quiet study space, set specific goals, and use techniques like the Pomodoro method. Take regular breaks, stay hydrated, and review what you've learned. Consistency and small steps lead to big progress!

helpful | clear | safe

Different Feedback, Same Bandit Philosophy

Setting	Action	Feedback
MAB / Linear bandit	single item	Scalar reward
Duelling bandit	pair of items	Binary reward: one or the other item
MNL bandit	unordered slate (set)	Click/no click
Cascade bandit	ordered slate (list)	Position of click/no click

Estimate unknowns → quantify uncertainty → explore strategically

Conclusion

- Many applications involve offering the user a set (or ordered list) of items
  - Feedback is in the form of a choice made by the user
- Bandit concepts can extend to this setting too!

Conclusion

- Many applications involve offering the user a set (or ordered list) of items
  - Feedback is in the form of a choice made by the user
- Bandit concepts can extend to this setting too!
- Continue with item-level scores as the core basis, which have to be estimated
- Use UCB principles along with a realistic choice model to get good regret performance

Conclusion

- Many applications involve offering the user a set (or ordered list) of items
  - Feedback is in the form of a choice made by the user
- Bandit concepts can extend to this setting too!
- Continue with item-level scores as the core basis, which have to be estimated
- Use UCB principles along with a realistic choice model to get good regret performance
- Real-world systems can deviate from simple models→ need smart engineers!