Discrete Choice & Applications

Flavio Chierichetti
Sapienza University
Rome

Ravi Kumar
Google
Mountain View, CA

Andrew Tomkins
Google
Mountain View, CA

Outline

Background

- Random utility models
- Definitions, equivalence properties
- Multinomial logit (MNL) models
- BTL, Plackett–Luce, Nested logit models
- k-RUMS, RUMS with ties
- Independence of irrelevant attributes

Algorithms

- Representations
- Compression and coresets
- Fitting
- Learning
- Special Cases of RUMs

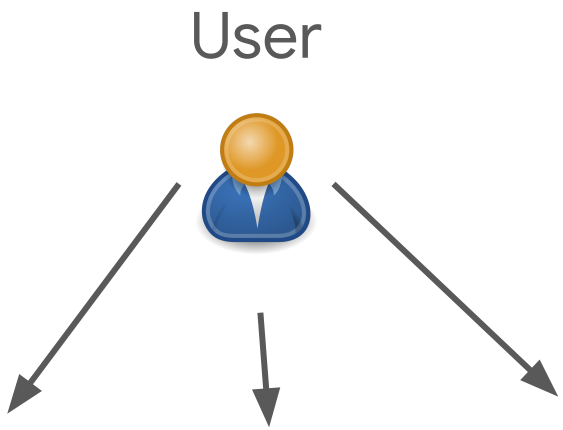
Applications

- Connections to deep networks
- Network formation
- Document ranking
- User segmentation
- Recommendation systems
- Social choice and voting

Background

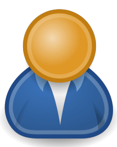
- Discrete choice
- Random utility models (RUMs)
- Definitions, equivalence properties
- Multinomial logit (MNL) models; BTL
- Independence of irrelevant attributes
- PCMC, Nested logit models, k-RUMS

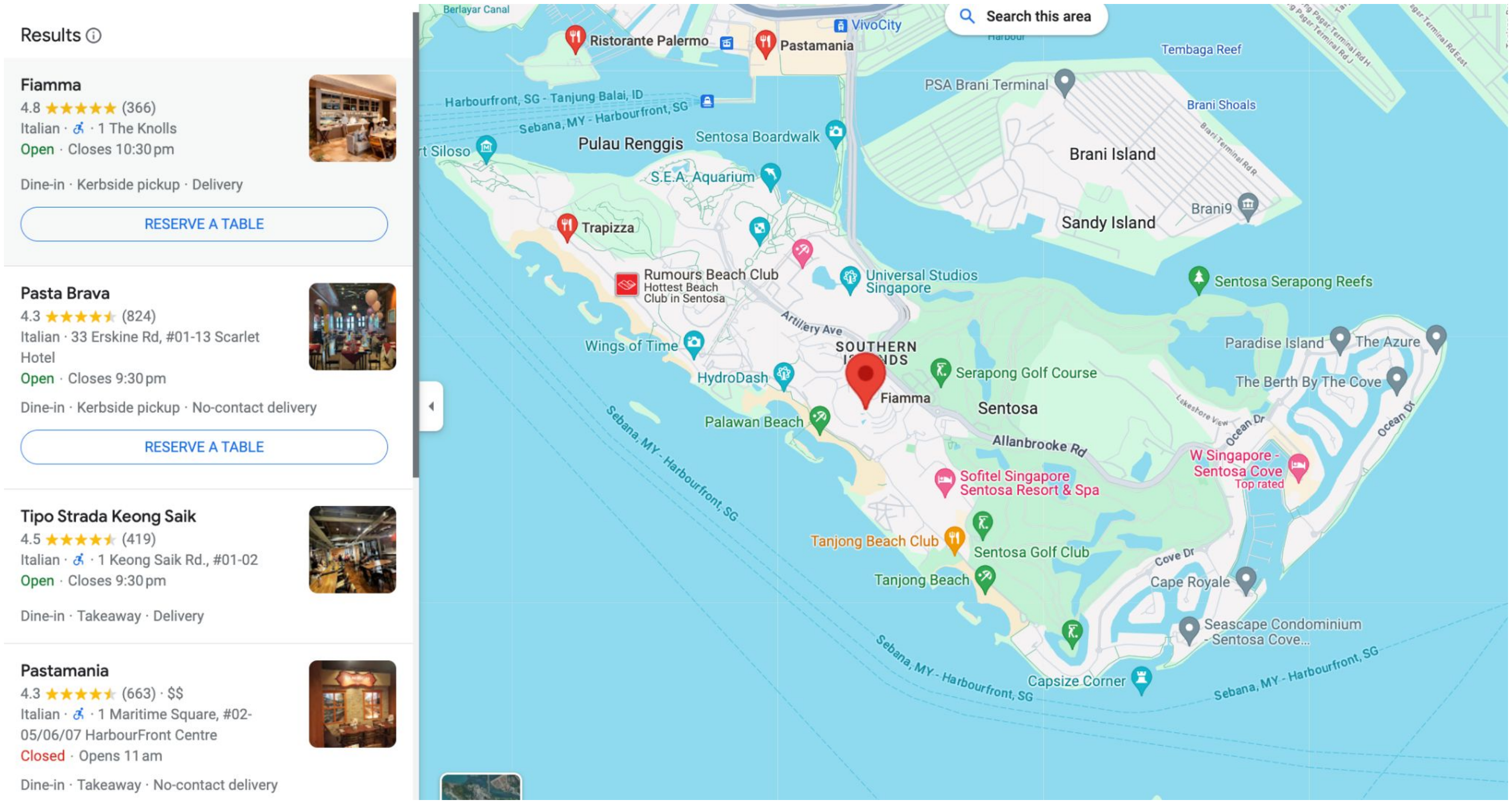
Discrete choice



Where shall we eat tonight?

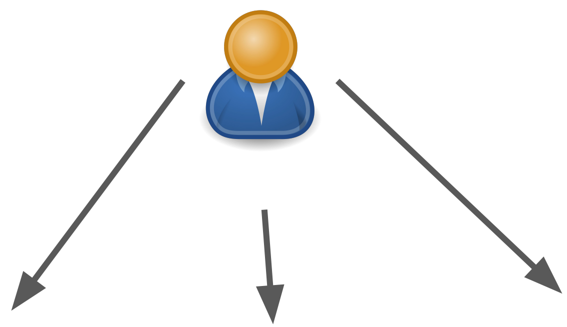
Discrete choice





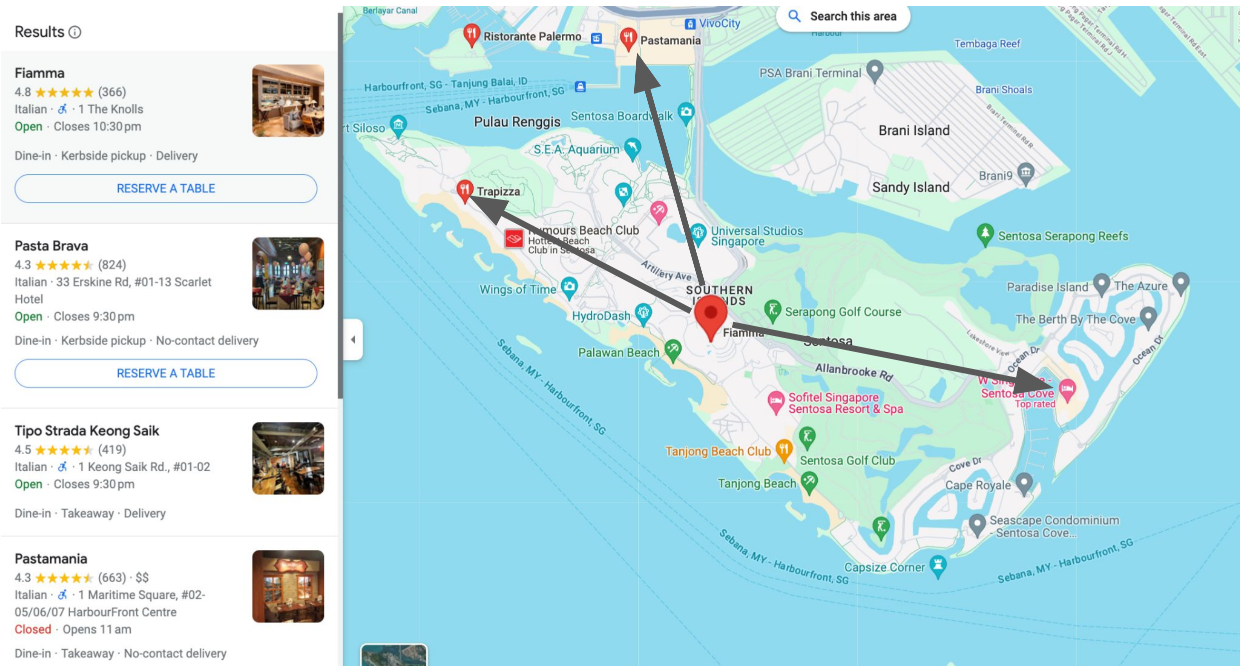
Discrete choice

User



How do we get there?

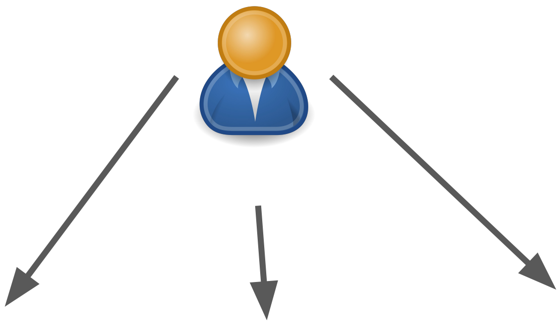
Discrete choice: Factors



- Quality
- Distance
- Price
- Cuisine type
- Time since last visit
- Companion opinion

Discrete choice: Repeat consumption

User

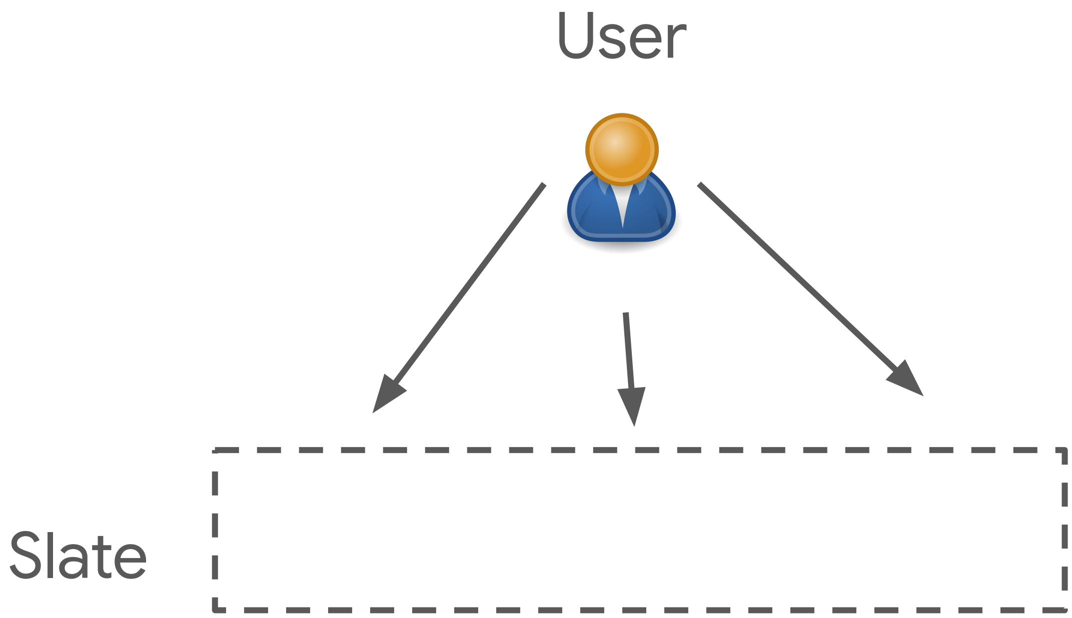


Most items we consume are not for the first time

- Sometimes go for reliability
- Sometimes go for novelty

Each day ...

Goal of discrete choice



Explain rational choice among discrete alternatives

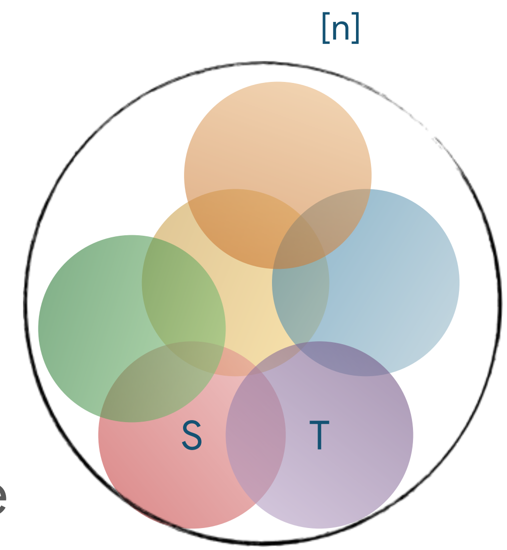
Discrete choice as a field of study

- Important model in behavioral economics, social sciences, machine learning, etc
- Widely used in studying consumer demand in practice
- Especially important in online/interactive settings (search results, product alternatives, recommendations, etc)

- Daniel McFadden, 2000 Economics Nobel Prize

"for his development of theory and methods for analyzing discrete choice"

Modeling discrete choice



Universe = [n] = {1, ..., n}

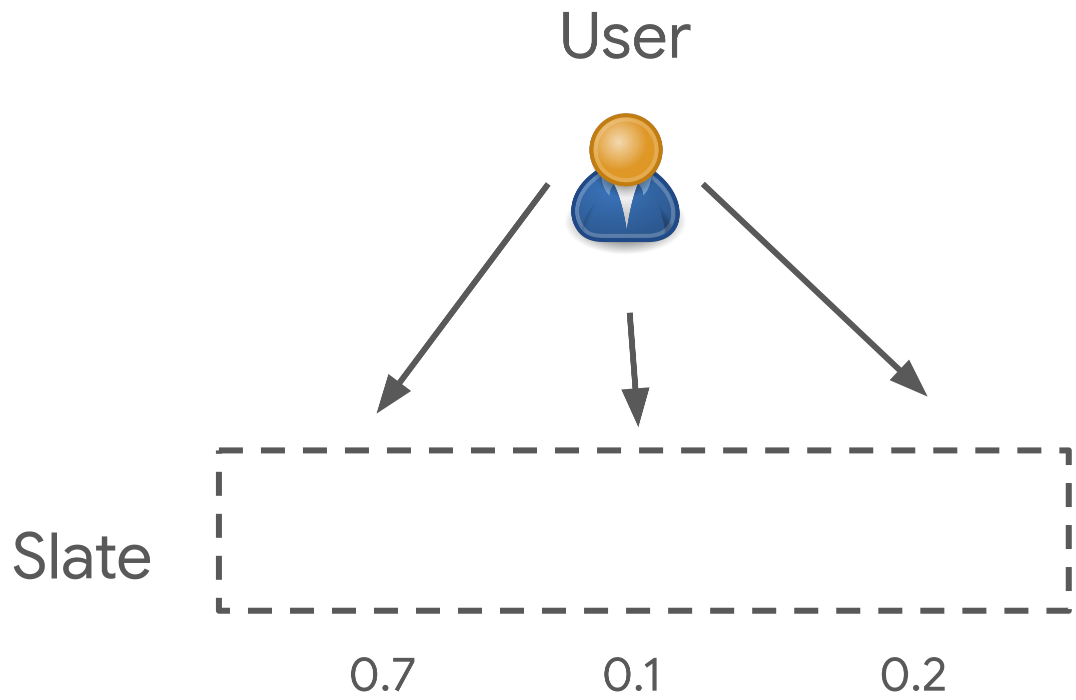
Slates = non-empty subsets of [n]

Model. A function f: slate ⟶ distribution over slate

- Captures uncertainty
- Can codify rational behavior

S and T highly overlap ⟹ f(S) and f(T) may be related

Example

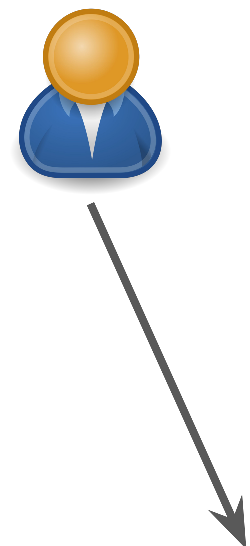


Random utility model (RUM) [Marschak 1960]

- There is a distribution U on utility vectors { $[n] \longrightarrow \mathbb{R}$ }
- Each user is drawn from U and will choose highest utility option in a slate

Utility vectors

Given a universe of n items, the user samples a utility vector $(u_1, \ldots, u_n)$ from a joint continuous distribution U



Given a slate S, the user will select the slate item with largest perceived utility

S = {  }

RUMs

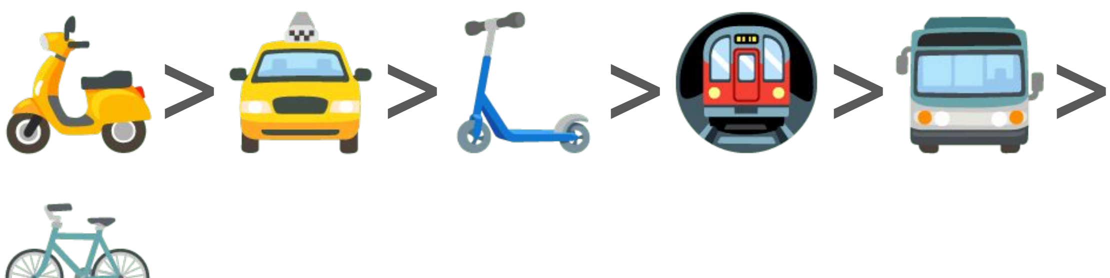
- Continuous distribution U on utility vectors $\{ [n] \longrightarrow \mathbb{R} \}$
  - For simplicity, assume no ties
- Each user is $(u_1, \ldots, u_n) \sim U$ iid and will choose highest utility option in a slate T (ie, $\text{argmax}_{t \in T} u_t$)
- Highly overlapping subsets will be related
  - Eg, $\Pr[j \mid T] \geq \Pr[j \mid T \cup \{i\}]$ for $j \in T$ and $i \notin T$
- Regularity: $\Pr[j \mid T] \geq \Pr[j \mid S]$, when $S \supseteq T$
- Rational behavior $\Longrightarrow$ order of utilities determines choice

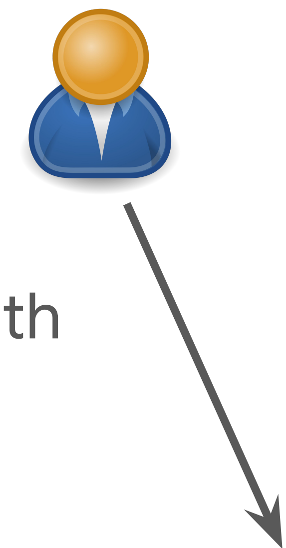
Permutation process

- There is a distribution P on permutations $\{ [n] \leftrightarrow [n] \}$
- Each user is a permutation $\pi \sim P$ and will choose highest ranked option, according to π, in a slate

Permutations

Given a universe of n items, the user samples a permutation $\pi = (u_{i1} > u_{i2} > \cdots > u_{in})$ from a distribution P





Given a slate S, the user will select the item with largest rank in π

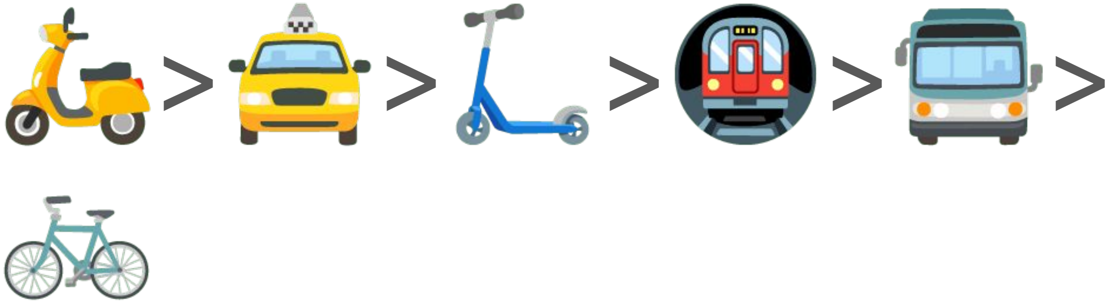
S = {

}

Equivalence

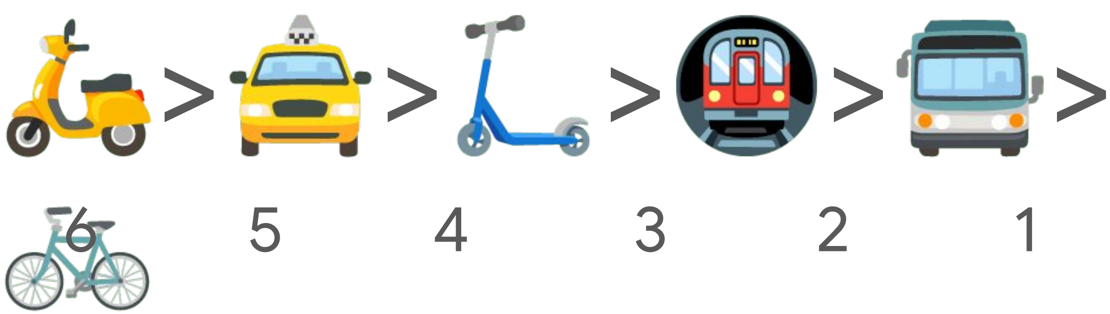
- Given a utility vector u, we can sort the items by utility, to obtain an equivalent permutation π

1.3 2.4 0.2 1.0 3.0 4.3



Equivalence

- Given a utility vector u, we can sort the items by utility, to obtain an equivalent permutation π
- Given a permutation π, we can assign utility (n - i) to the item having rank i in π



Equivalence

- Given a utility vector u, we can sort the items by utility, to obtain an equivalent permutation π
- Given a permutation π, we can assign utility $(n - i)$ to the item having rank i in π

We can transform a RUM defined by a distribution U over utility vector into an equivalent RUM defined by a distribution over permutations P and vice versa

Permutations to winners

30% A > B > C > D

10% B > C > A > D

40% B > A > C > D

20% B > A > D > C

⇒

$D_{AB}(A) = 3/10$

$D_{AC}(A) = 9/10$

...

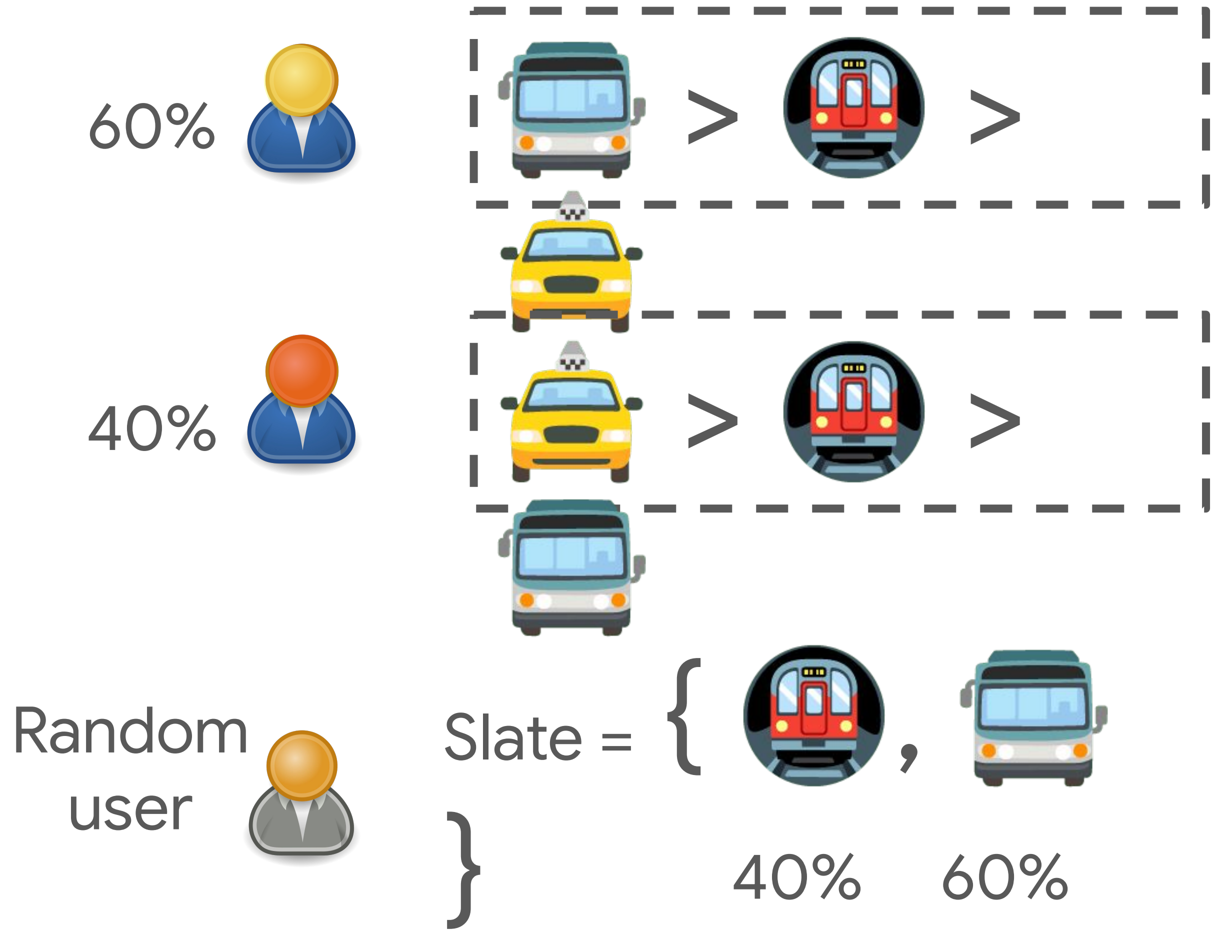
$D_{ABCD}(C) = 0$

Winner distribution

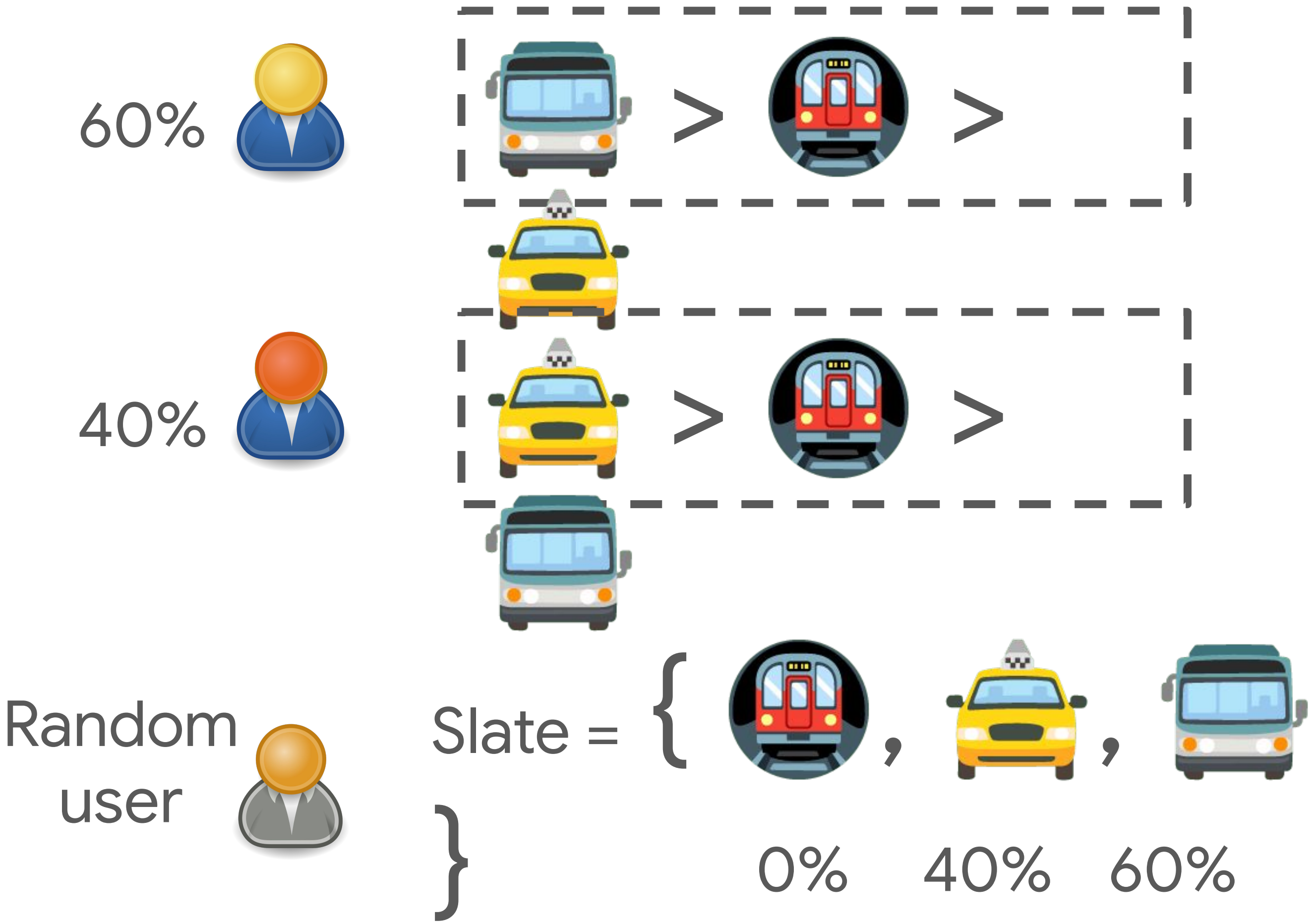
Assume a universe [n] and a distribution on the permutations of [n]

Given a slate $S \subseteq [n]$, let $D_S(i)$ for $i \in S$ be the probability that a random permutation (ie, user) prefers i to every other element of S

Winner distribution (Eg)



Winner distribution (Eg)



Oracles for RUMs

Given a slate S

- max-sample(S): picks an unknown random permutation π, and returns the element of S with maximum rank in π
- max-dist(S): returns $D_S(i)$, for all $i \in S$, ie, the probability that i wins in S given a random permutation

Oracles for RUMs (Eg)

S = { 🚇, 🚖, 🚍 }

- max-sample(S): ⟨ 🚇 > 🚖 > 🚍 ⟩; return 🚇
- max-dist(S): return D_S = ⟨ 0.2, 0.05, 0.75 ⟩

Multinomial logit (MNL)
[Bradley & Terry 1952; Luce 1959]

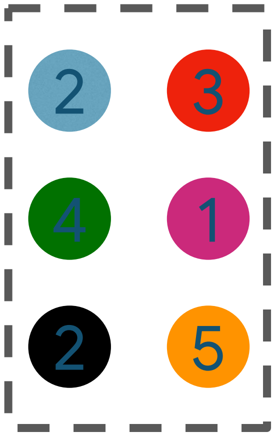
Classical special case of RUMs

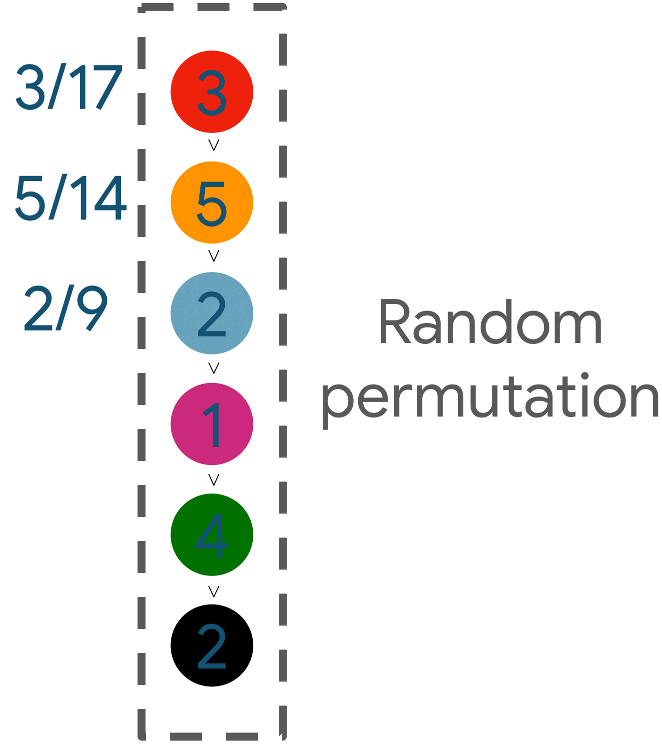
Model. Given a universe [n] of items and a positive weight a_j for each item $j \in [n]$

For a subset (slate) S of [n], the probability of choosing j in slate S is proportional to w_j

$$\Pr[\text{choosing } j \text{ from } S] = w_j \,/\, \Sigma_{k \in S}\ w_k$$

Permutations from an MNL (Eg)





Pick the next item in the permutation at random between the remaining ones, with probability proportional to its weight

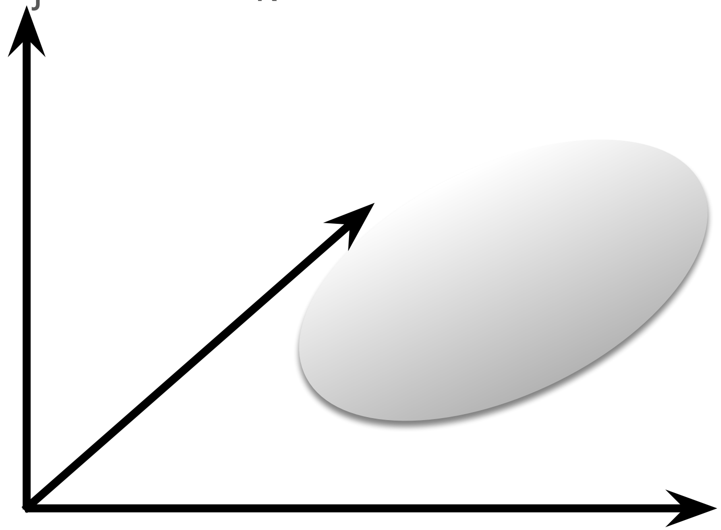
MNLs = RUMs + specific noise

Assume each item j has an absolute “true quality” V_j

Model. Each user deviates from this by random noise ε_j and so the actual utility of user for item j is $u_j = V_j + \varepsilon_j$

Pr[user chooses j] = $\Pr[\forall k \neq j,\ V_j + \varepsilon_j > V_k + \varepsilon_k]$

Suppose ε_j's are iid



A convenient choice of noise

Suppose $\Pr[\varepsilon] = \exp(-(\varepsilon + \exp(-\varepsilon))$

- Gumbel distribution
- Models the distribution of the maximum of samples (from various distributions)

Pr[user chooses j], by simple integration,

$$= \Pr[j = \text{argmax}\, \{V_k + \varepsilon_k\}] \propto \exp(V_j)$$

$\Rightarrow$ Pr[user chooses j from S] $= \exp(V_j) \,/\, \Sigma_{k \in S} \exp(V_k)$

Multinomial regression gives identical choice probabilities to RUM with Gumbel-distributed noise!

Including features in MNL

We can make V_j to depend on item features or user features or both

Suppose $V_j = \langle y_j, x \rangle$, where y_j is item feature for item j and x is the user feature

Suppose $V_j = \langle y, x_j \rangle$, where y is feature of an item and x_j is user feature for item j

Multinomial logit

Pr[user chooses j from S] = $\exp(\langle y_j, x \rangle) / \Sigma_{k \in S} \exp(\langle y_k, x \rangle)$

Choice MNL

Pr[user chooses j from S] = $\exp(\langle y, x_j \rangle) / \Sigma_{k \in S} \exp(\langle y, x_k \rangle)$

MNLs in machine learning

MNLs, or softmax layers, are common in ML

- Multi-class problems
- Dual encoders
- Mixture of MNLs are sometimes used

$$\Pr[\text{output class } j] = \exp(\langle \beta_j, \text{input} \rangle) \,/\, \Sigma_k \exp(\langle \beta_k, \text{input} \rangle)$$

Limitations of MNL

Assume positive weight w_a for each item $a \in [n]$

Options: {a, b}: $\Pr[a \mid a \text{ or } b] = w_a / (w_a + w_b)$

Options: {a, b, c}: $\Pr[a \mid a \text{ or } b] = w_a / (w_a + w_b)$

Relative likelihood of a versus b does not depend on other alternatives: Choices are Independent of Irrelevant Alternatives (IIA), aka Luce's Axiom of Choice

MNL $\equiv$ choice with IIA

MNLs are insufficient to capture common settings

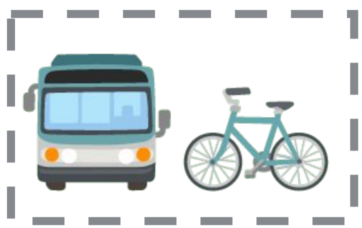
Luce’s axiom of choice

Pr[a | a or b] does not change when c is added to slate

“Menu effect” or “decoy effect” in practice

Stationary rational choice might not follow IIA

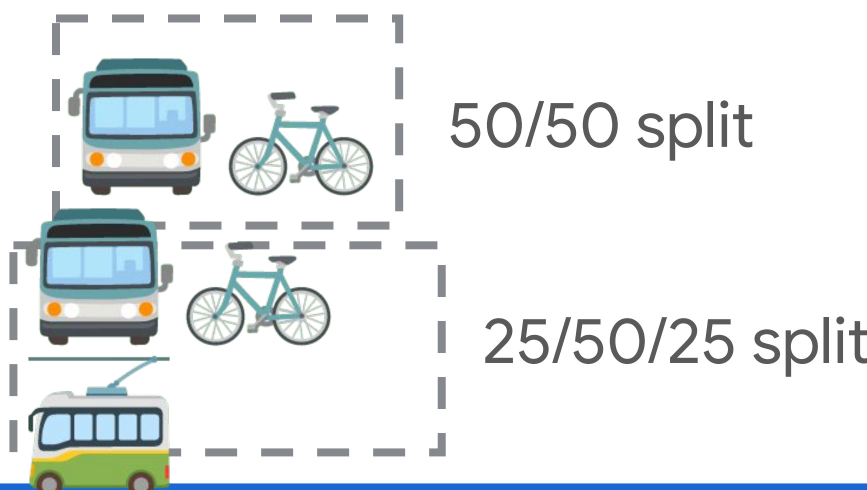
		
User Type 1: 50%	5	100
User Type 2: 25%	100	1
User Type 3: 25%	75	1



50/50 split

Stationary rational choice might not follow IIA

			
User Type 1: 50%	5	100	15
User Type 2: 25%	100	1	75
User Type 3: 25%	75	1	100



Mixture of MNLs

Modeling distinct populations with simple MNL is the problem

Allowing a mixture of population, with a population-specific MNL, can solve the problem

- New items need not cannibalize equally from all other items
- Eg, a new bus route affects only bus riders

2-MNL mixture

Given a universe [n] of items and positive weights u_j and v_j for each item $j \in [n]$

For a slate S, the probability of choosing $j \in S$ equals

$$\gamma \cdot u_j \,/\, \Sigma_{k \in S}\, u_k + (1 - \gamma) \cdot v_j \,/\, \Sigma_{k \in S}\, v_k$$

Uniform mixture when $\gamma = 1/2$

MNL mixtures can approximate arbitrarily well any RUM [McFadden & Train 2000]

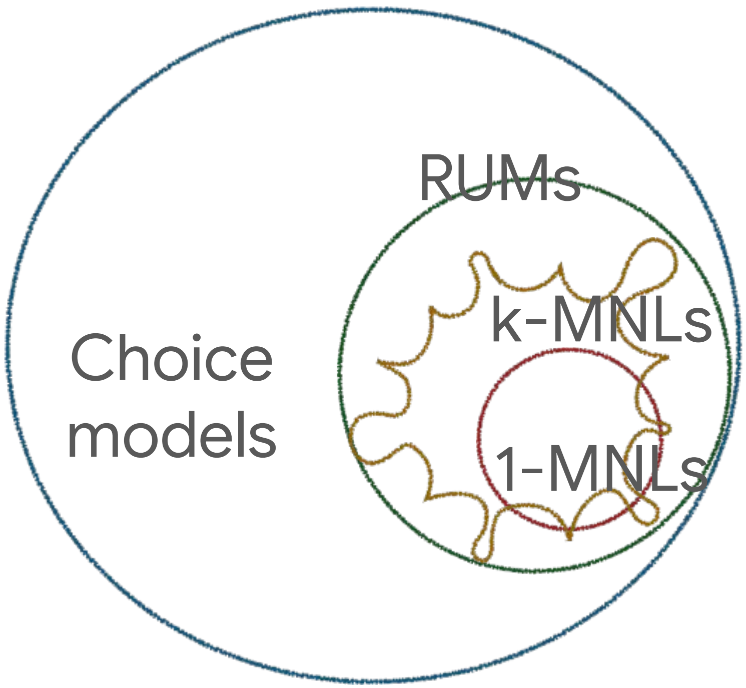
The story so far

RUM

- General approach to characterize choice
- Harder to interpret (and learn)

MNL

- Captures only RUMs with IIA
- Easy and fast to optimize
- Easy to interpret



A brief history of IIA

R Duncan Luce formulated “Axiom of Choice” (1959)

- Arrow (1951) proved the Impossibility Theorem showing that IIA was one of several mutually incompatible properties of a social choice function
- Bradley and Terry (1952) introduced a pairwise comparison choice model
  - Studied by Zermelo (1920s)
  - Often called the BTL model

Later, many authors, notably McFadden, completed the story extending BTL to MNL

What if IIA is violated?

Situation is much more complex....

Most powerful models are:

- Mathematically complex
- Computationally intractable
- Sophisticated external representations of dependence

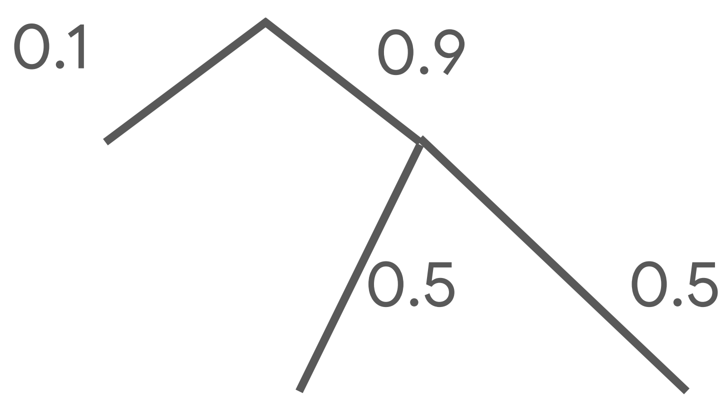
Practitioners with non-IIA data typically use "Nested Logit"

Problems with IIA revisited

0.1 ~~0.9~~ 0.45 0.45

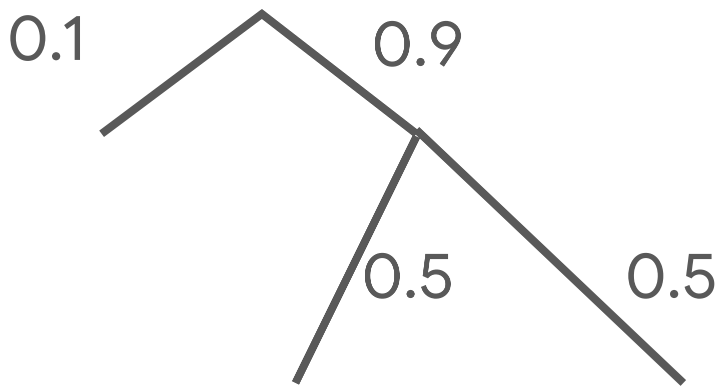
Likelihood(Bus) / Likelihood(Bike) = ~~0.9~~ / 0.1

0.45 / 0.1



Nested logit

Modeling the decision as a tree is a nested, sequential, or hierarchical logit model. It looks like a sequence of multinomial logits. [McFadden 78]



Nested logit: Connections to RUM & MNL

Model. Nested Logit (NL) selects an item by traversing tree from root, applying MNL at each level

Casting NL as RUM:

- Utility of each item is a priori fixed
- Each user's utilities are perturbed
- Perturbation is drawn from specific joint distribution

Power of NL:

- Pros: Captures hierarchical cannibalization cleanly; generalizes MNL
- Cons: Choices must separate cleanly into nests

MNL in graphs

Model. Define a Markov chain given a graph where each node u has score $s_u > 0$

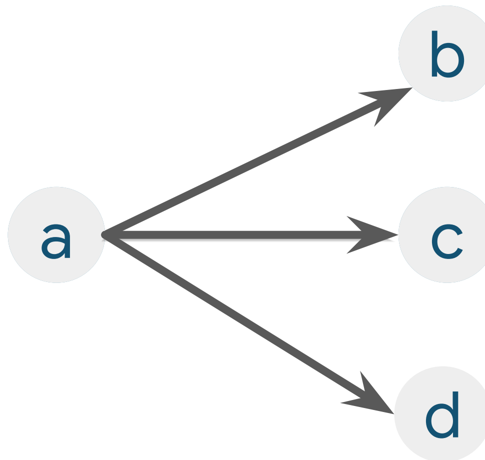
- Transition according to MNL choice

$$M_{uv} = s_v / \Sigma_{w \in \text{neighbors}(u)} s_w$$

MNL in graphs (Eg)

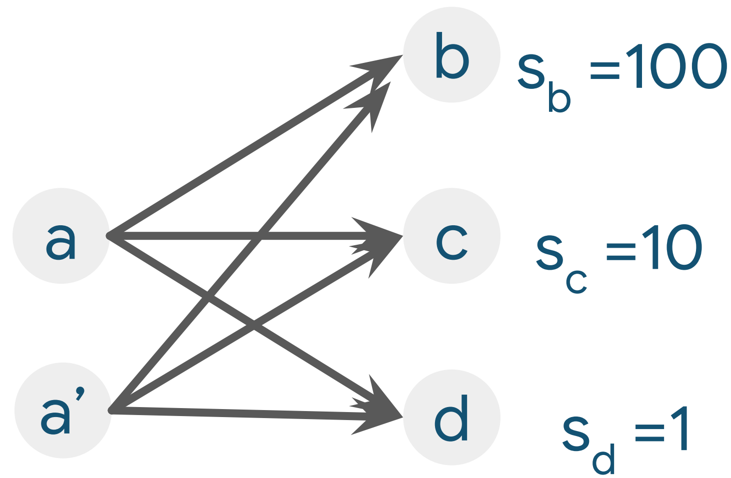
Transition probability proportional to the score of the node

- Eg, $M_{ac} = s_c / (s_b + s_c + s_d)$



Transition probabilities are context dependent

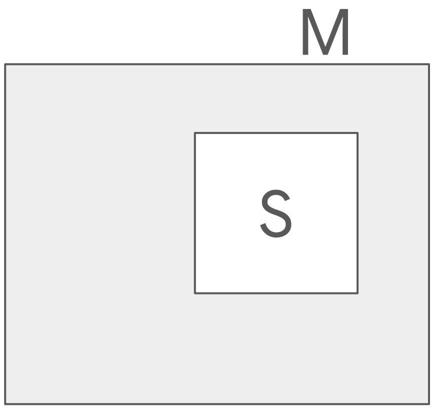
- Eg, $M_{ac} = 0.01$, $M_{a'c} = 0.91$



Pairwise choice Markov chain [Ragain & Ugander 2016]

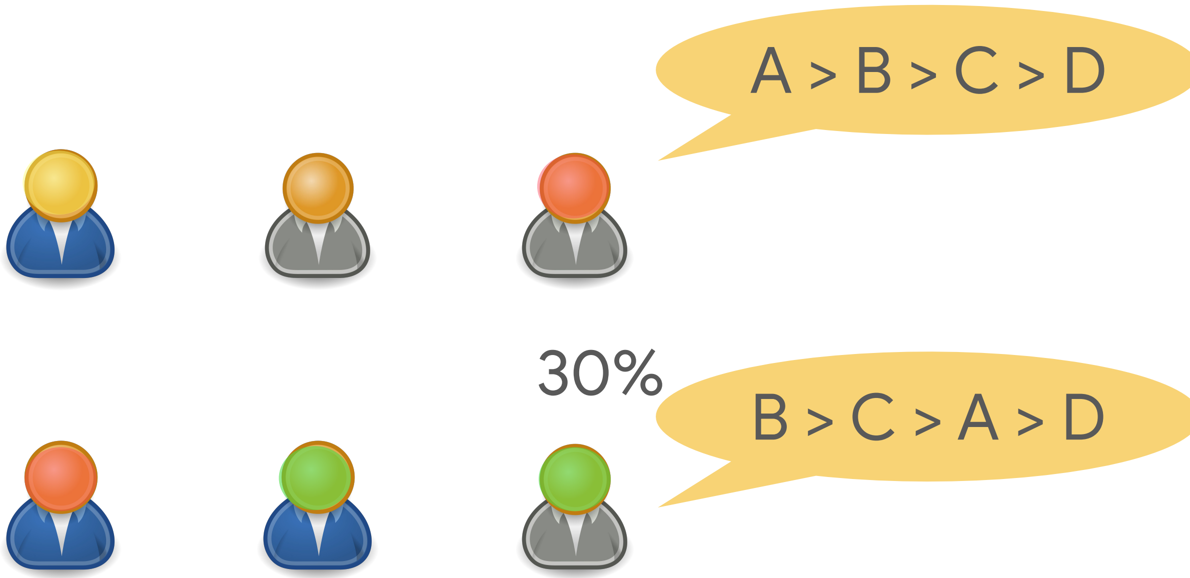
Given pairwise winning matrix M and slate S

- Construct M_S by restricting M to rows and columns of S
- Make M_S stochastic
- Compute stationary π_S of M_S to yield choice probability $\pi_S(i)$ for item i



- Can represent BTL
- Not a RUM in general
  - Can violate regularity
- Has other nice properties

k-RUMS



10%

There are only a few types of users

- Support of the permutation distribution is small
- Pragmatic
- Computationally helpful

Computational problems in RUMs (Eg)

Goal. Learn D_S, for all $S \subseteq [n]$

How to learn the probability distributions governing the choice in a generic slate?

Assume oracle access to RUMs

Assuming large slates is less realistic

Quickly learning the winning distributions of the slates is important for applications

... but there are exponentially many slates!

Computational problems in RUMs (Eg)

Goal. Given pairwise winning matrix, find the closest RUM

Head-to-head contests, online experiences comparing one item and an alternative

Algorithms

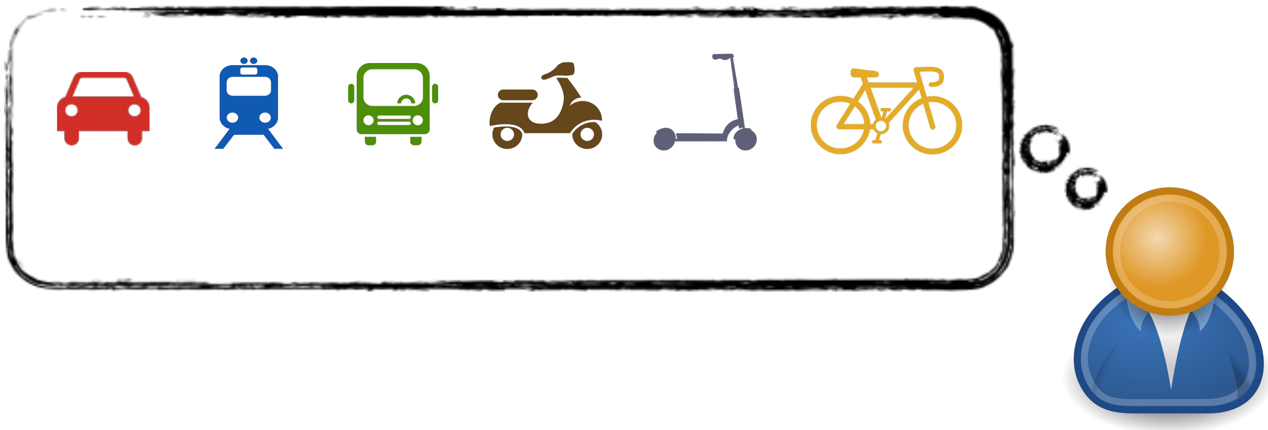
- Representations
- Compression and coresets
- Fitting
- Learning
- Special cases of RUMs

RUM Representations

- How to represent RUMs?
- How do different representations change the computational costs of various RUM tasks (e.g., fitting, learning)?

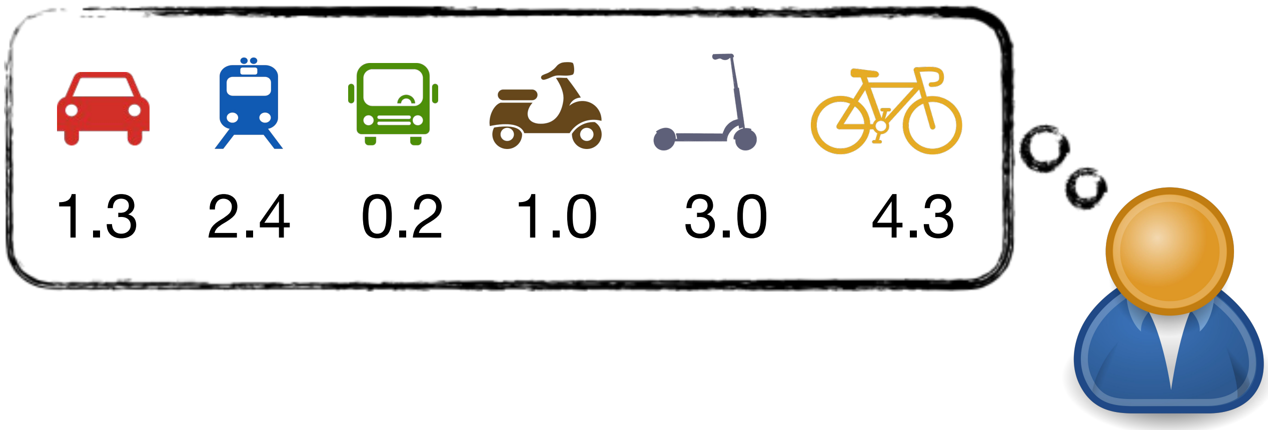
Utility Vectors

- Given the full set of n items, the user samples a utility vector $(u_1, u_2, \ldots, u_n)$ from a joint continuous distribution U



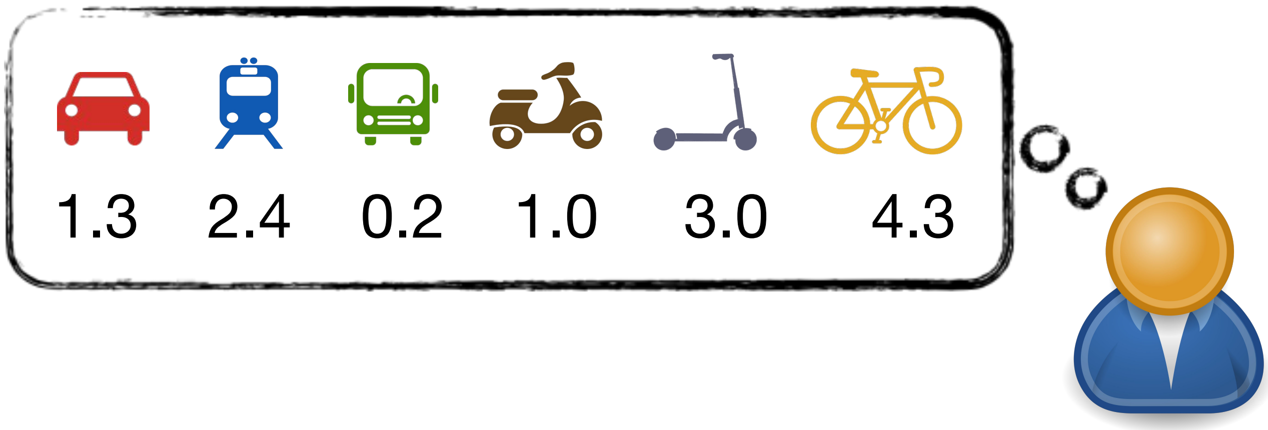
Utility Vectors

- Given the full set of n items, the user samples a utility vector $(u_1, u_2, \ldots, u_n)$ from a joint continuous distribution U



Utility Vectors

- Given the full set of n items, the user samples a utility vector $(u_1, u_2, \ldots, u_n)$ from a joint continuous distribution U



- Given a slate S, the user will select the slate item with largest perceived utility

Utility Vectors

- Given the full set of n items, the user samples a utility vector $(u_1, u_2, \ldots, u_n)$ from a joint continuous distribution U

- Given a slate S, the user will select the slate item with largest perceived utility

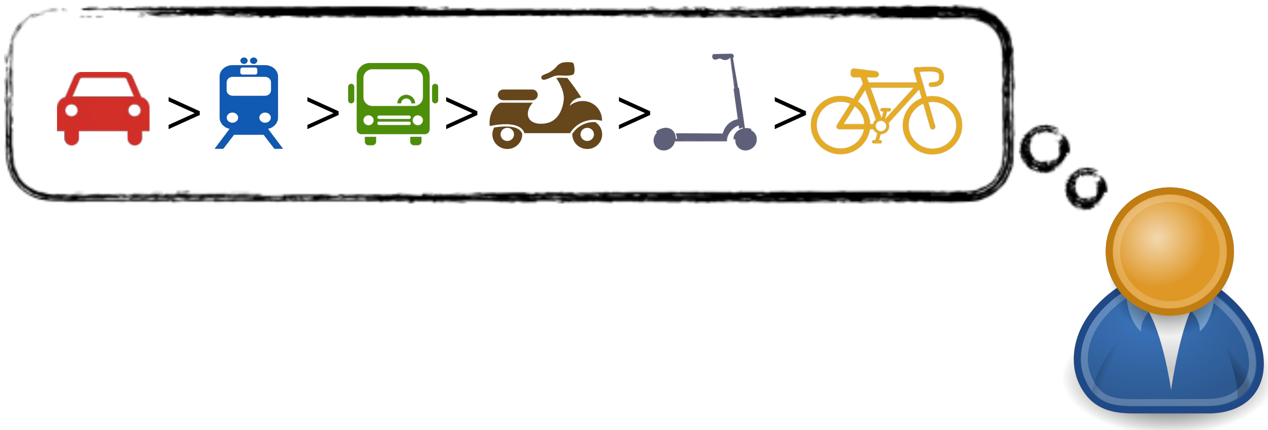
Utility Vectors

- Given the full set of *n* items, the user samples a utility vector $(u_1, u_2, \ldots, u_n)$ from a joint continuous distribution *U*

- Given a slate *S*, the user will select the slate item with largest perceived utility

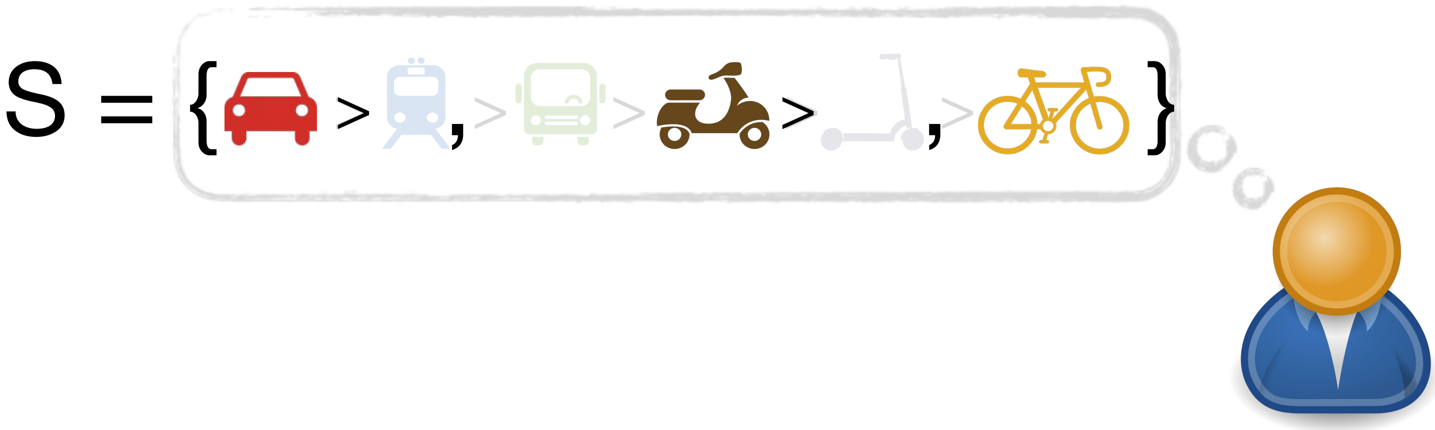
Permutations

- Given the full set of n items, the user samples a permutation $\pi = (u_{i1} > u_{i2} > \ldots > u_{in})$ from a distribution P



Permutations

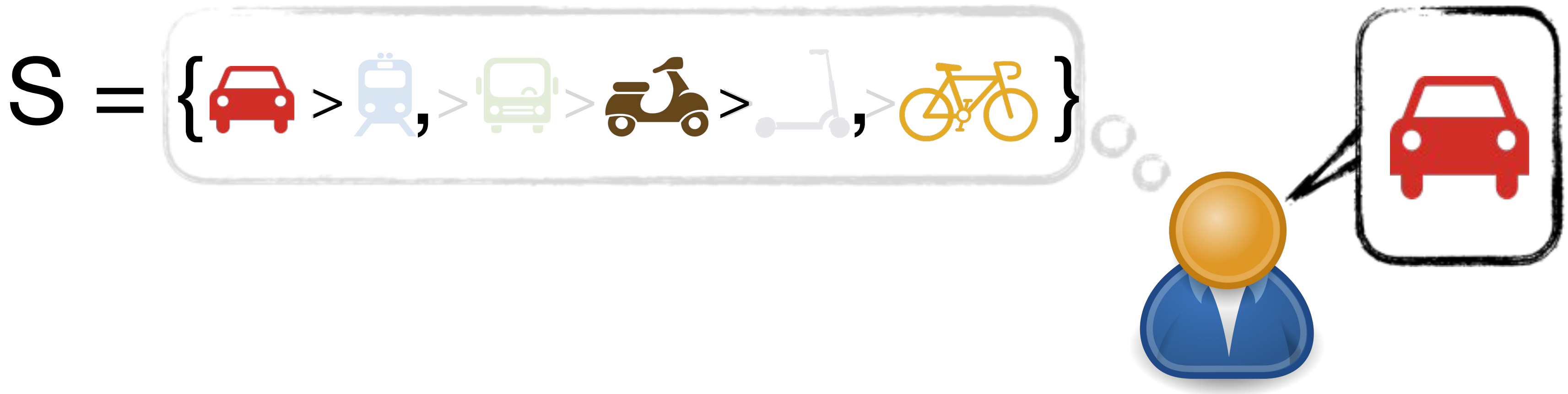
- Given the full set of n items, the user samples a permutation $\pi = (u_{i1} > u_{i2} > \ldots > u_{in})$ from a distribution P



- Given a slate S, the user will select the item with largest rank in π

Permutations

- Given the full set of *n* items, the user samples a permutation $\pi = (u_{i1} > u_{i2} > \ldots > u_{in})$ from a distribution *P*



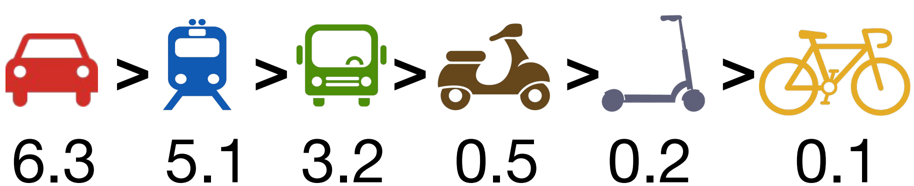
- Given a slate *S*, the user will select the item with largest rank in π

Equivalence

- As we mentioned, the two models are equivalent:
  - given a utility vector *U*, we can sort the items by utility, to obtain an equivalent permutation π

Equivalence

- As we mentioned, the two models are equivalent:
  - given a utility vector *U*, we can sort the items by utility, to obtain an equivalent permutation π

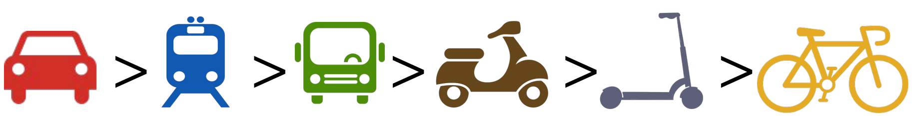


Equivalence

- As we mentioned, the two models are equivalent:
  - given a utility vector U, we can sort the items by utility, to obtain an equivalent permutation π
  - given a permutation π, we can assign utility $n - i$ to the item having rank i in π

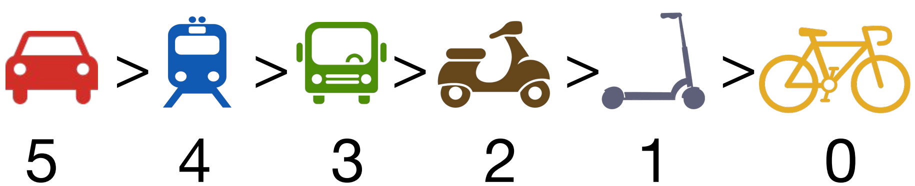
Equivalence

- As we mentioned, the two models are equivalent:
  - given a utility vector *U*, we can sort the items by utility, to obtain an equivalent permutation π
  - given a permutation π, we can assign utility $n - i$ to the item having rank i in π



Equivalence

- As we mentioned, the two models are equivalent:
  - given a utility vector *U*, we can sort the items by utility, to obtain an equivalent permutation π
  - given a permutation π, we can assign utility $n - i$ to the item having rank i in π



Equivalence

- As we mentioned, the two models are equivalent:
  - given a utility vector *U*, we can sort the items by utility, to obtain an equivalent permutation π
  - given a permutation π, we can assign utility $n - i$ to the item having rank i in π

This way, we can transform a RUM defined by a distribution *U* over utility vectors into an equivalent RUM defined by a distribution over permutations *P*, and vice versa.

Representations

- Is any of these two representations preferable for the tasks we are interested in, e.g.,

  1. storing/sketching a RUM,
  2. fitting a RUM, or
  3. learning a RUM?

Storing a RUM

- There are infinitely many utility vectors - a distribution over utility vectors can be impossible to store.

Storing a RUM

- There are infinitely many utility vectors - a distribution over utility vectors can be impossible to store.

- But there are only finitely many permutations... thus, we can perfectly represent a RUM with *n!* many scalars

$$\pi_1 = n > n - 1 > \ldots > 2 > 1$$
$$\pi_2 = n > n - 1 > \ldots > 1 > 2$$

$$\pi_{n!} = 1 > 2 > \ldots > n - 1 > n$$

Storing a RUM

- There are infinitely many utility vectors - a distribution over utility vectors can be impossible to store.
- But there are only finitely many permutations... thus, we can perfectly represent a RUM with *n!* many scalars

$p(\pi_1)$	$\pi_1 = n > n - 1 > \ldots > 2 > 1$
$p(\pi_2)$	$\pi_2 = n > n - 1 > \ldots > 1 > 2$
$p(\pi_{n!})$	$\pi_{n!} = 1 > 2 > \ldots > n - 1 > n$

Storing a RUM

- There are infinitely many utility vectors - a distribution over utility vectors can be impossible to store.

- But there are only finitely many permutations... thus, we can perfectly represent a RUM with *n!* many scalars

Storing a RUM

- There are infinitely many utility vectors - a distribution over utility vectors can be impossible to store.

- But there are only finitely many permutations… thus, we can perfectly represent a RUM with $n!$ many scalars

- Can this representation be shrunk?

Dimensionality

- Since RUMs can be represented as distributions over permutations, they are part of a $n!$-dimensional affine space.

Dimensionality

- Since RUMs can be represented as distributions over permutations, they are part of a *n!*-dimensional affine space.

- On the other hand, there are 2^n-1 winner distributions

Dimensionality

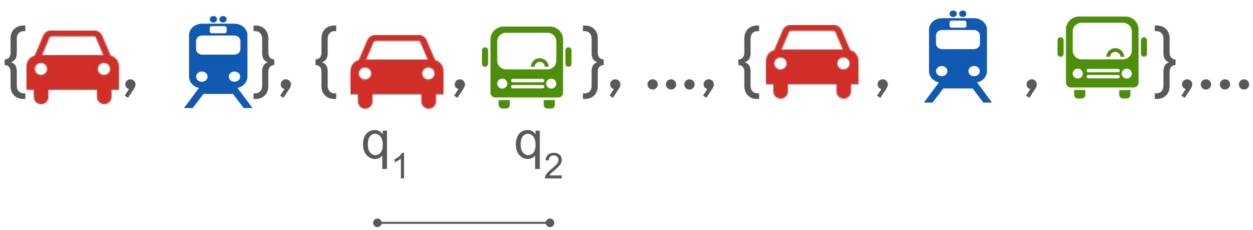
- Since RUMs can be represented as distributions over permutations, they are part of a *n!*-dimensional affine space.

- On the other hand, there are 2^n-1 winner distributions, each of which is part of an affine space with no more than *n* dimensions

Dimensionality

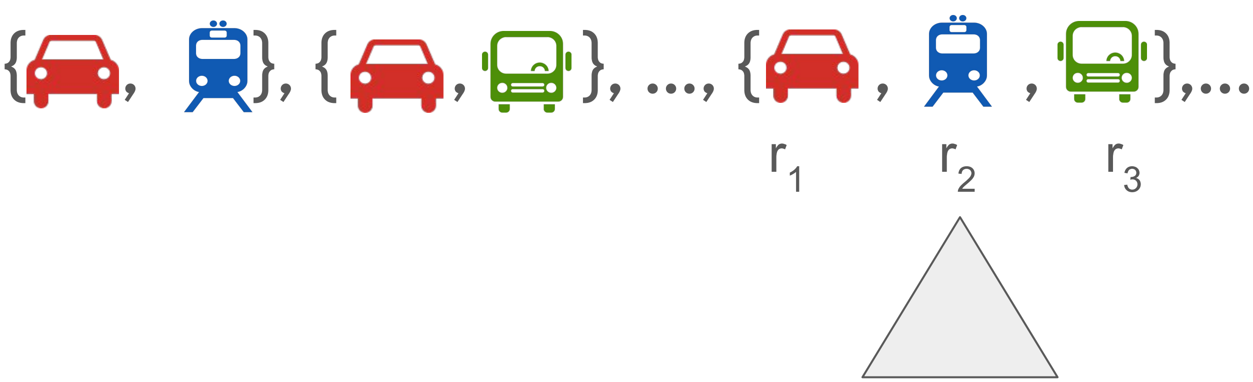
- Since RUMs can be represented as distributions over permutations, they are part of a *n!*-dimensional affine space.

- On the other hand, there are 2^n-1 winner distributions, each of which is part of an affine space with no more than *n* dimensions



Dimensionality

- Since RUMs can be represented as distributions over permutations, they are part of a *n!*-dimensional affine space.
- On the other hand, there are 2^n-1 winner distributions, each of which is part of an affine space with no more than *n* dimensions



Dimensionality

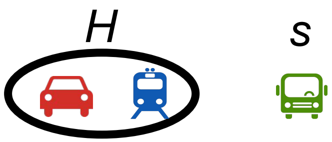
- Since RUMs can be represented as distributions over permutations, they are part of a *n!*-dimensional affine space.

- On the other hand, there are 2^n-1 winner distributions, each of which is part of an affine space with no more than *n* dimensions — that is, the number of dimensions of the *input* affine space (the max-dist class $\{D_S(i)\}_{i \in S \subseteq [n]}$) is bounded by $O(n\, 2^n)$

Dimensionality

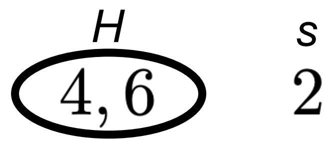
- Since RUMs can be represented as distributions over permutations, they are part of a *n!*-dimensional affine space.
- On the other hand, there are 2^n-1 winner distributions, each of which is part of an affine space with no more than *n* dimensions — that is, the number of dimensions of the *input* affine space (the max-dist class $\{D_S(i)\}_{i \in S \subseteq [n]}$) is bounded by $O(n\,2^n)$
- *Can RUMs be store more efficiently?*

Head Distributions



- Let *H* be a set of RUM items, and *s* an item not in *H*,

Head Distributions



- Let *H* be a set of RUM items, and *s* an item not in *H*,

Head Distributions

H s

$(4, 6)$ 2

- Let *H* be a set of RUM items, and *s* an item not in *H*,

- let $P_{H,s}$ be the probability that a random RUM permutation has the items of *H*, in any order, as its *|H|* top-most items, and it has *s* in position *|H|+1*.

Head Distributions

H *s*

$4, 6$ 2

- Let *H* be a set of RUM items, and *s* an item not in *H*,
- let $P_{H,s}$ be the probability that a random RUM permutation has the items of *H*, in any order, as its *|H|* top-most items, and it has *s* in position *|H|+1*.

$$P_{\{4,6\},2} = \Pr_{\pi}[\pi \text{ begins with } 4 > 6 > 2 > \ldots, \text{ or with } 6 > 4 > 2 > \ldots]$$

Head Distributions

H s

$(4, 6)$ 2

- Let *H* be a set of RUM items, and *s* an item not in *H*,

- let $P_{H,s}$ be the probability that a random RUM permutation has the items of *H*, in any order, as its *|H|* top-most items, and it has *s* in position *|H|+1*.

The Head Distribution of item *s* is, then, $P_{\star,s}$, that is, the probability distribution over the ***subset*** of items that beat *s* in a random permutation (the ***head*** of *s*)

Head Distributions

- The Head Distributions can *answer max-dist* queries:

$$D_S(s) = \Pr_{\pi}\left[s \text{ wins in } S \text{ with } \pi\right]$$

Head Distributions

- The Head Distributions can *answer max-dist* queries:

$$\begin{aligned} D_S(s) &= \Pr_{\pi}\left[s \text{ wins in } S \text{ with } \pi\right] \\ &= \Pr_{\pi}\left[\pi \text{ begins with some set of elements disjoint} \right. \\ &\qquad \left. \text{from } S, \text{ and continues with } s \text{ right after}\right] \end{aligned}$$

Head Distributions

- The Head Distributions can *answer max-dist* queries:

$$D_S(s) = \Pr_{\pi}\left[s \text{ wins in } S \text{ with } \pi\right]$$
$$= \Pr_{\pi}\left[\pi \text{ begins with some set of elements disjoint from } S, \text{ and continues with } s \text{ right after}\right]$$

$$\pi = (x_1 > \cdots > x_i > s > \cdots)$$
$$\text{with } \{x_1, \ldots, x_i\} \cap S = \varnothing$$

Head Distributions

- The Head Distributions can *answer max-dist* queries:

$$\begin{aligned} D_S(s) &= \Pr_{\pi}\left[s \text{ wins in } S \text{ with } \pi\right] \\ &= \Pr_{\pi}\left[\pi \text{ begins with some set of elements disjoint} \right. \\ &\qquad \left. \text{from } S, \text{ and continues with } s \text{ right after}\right] \\ &= \sum_{T \subseteq [n] \setminus S} P_{T,s} \end{aligned}$$

Head Distributions

- The Head Distributions can be learned using the *max-dist* oracle.

Head Distributions

- The Head Distributions can be learned using the *max-dist* oracle.
- Querying $D_{[n]}$ gives us $P_{\varnothing,i}, \forall i \in [n]$.

Head Distributions

- The Head Distributions can be learned using the *max-dist* oracle.
- Querying $D_{[n]}$ gives us $P_{\varnothing,i}, \forall i \in [n]$.

$$D_{[n]}(i) = \Pr_{\pi}\left[\pi \text{ begins with } i >_{\pi} \cdots\right] = P_{\varnothing,i}$$

Head Distributions

- The Head Distributions can be learned using the *max-dist* oracle.

- Querying $D_{[n]}$ gives us $P_{\varnothing,i}, \forall i \in [n]$.

$$D_{[n]}(i) = \Pr_{\pi}\left[\pi \text{ begins with } i >_{\pi} \cdots\right] = P_{\varnothing,i}$$

- As we said, $D_{[n]-H}(i) = \sum_{T \subseteq H} P_{T,i}$

Head Distributions

- The Head Distributions can be learned using the *max-dist* oracle.

- Querying $D_{[n]}$ gives us $P_{\varnothing,i}, \forall i \in [n]$.

$$D_{[n]}(i) = \Pr_{\pi}[\pi \text{ begins with } i >_{\pi} \cdots] = P_{\varnothing,i}$$

- As we said, $D_{[n]-H}(i) = \sum_{T \subseteq H} P_{T,i}$. Thus,

$$P_{H,i} = D_{[n]-H}(i) - \sum_{T \subset H} P_{T,i}$$

Dimensionality

- Head Distributions can then represent any RUM exactly, and form an affine space of $O(n\ 2^n)$ dimensions

Dimensionality

- Head Distributions can then represent any RUM exactly, and form an affine space of $O(n\,2^n)$ dimensions, that is, a space
  - much smaller than that of permutations (which had *n!* dimensions), and
  - having the same dimensionality of the input (the max-dist class $\{D_S(i)\}_{i \in S \subseteq [n]}$).

Dimensionality

- Head Distributions can then represent any RUM exactly, and form an affine space of $O(n\ 2^n)$ dimensions, that is, a space
  - much smaller than that of permutations (which had n! dimensions), and
  - having the same dimensionality of the input (the max-dist class $\{D_S(i)\}_{i \in S \subseteq [n]}$).

- While this is still very large, it cannot be improved if we want to *exactly* represent a RUM.

What is the Smallest Model for approximately representing a RUM?

Can we do lossy compression?

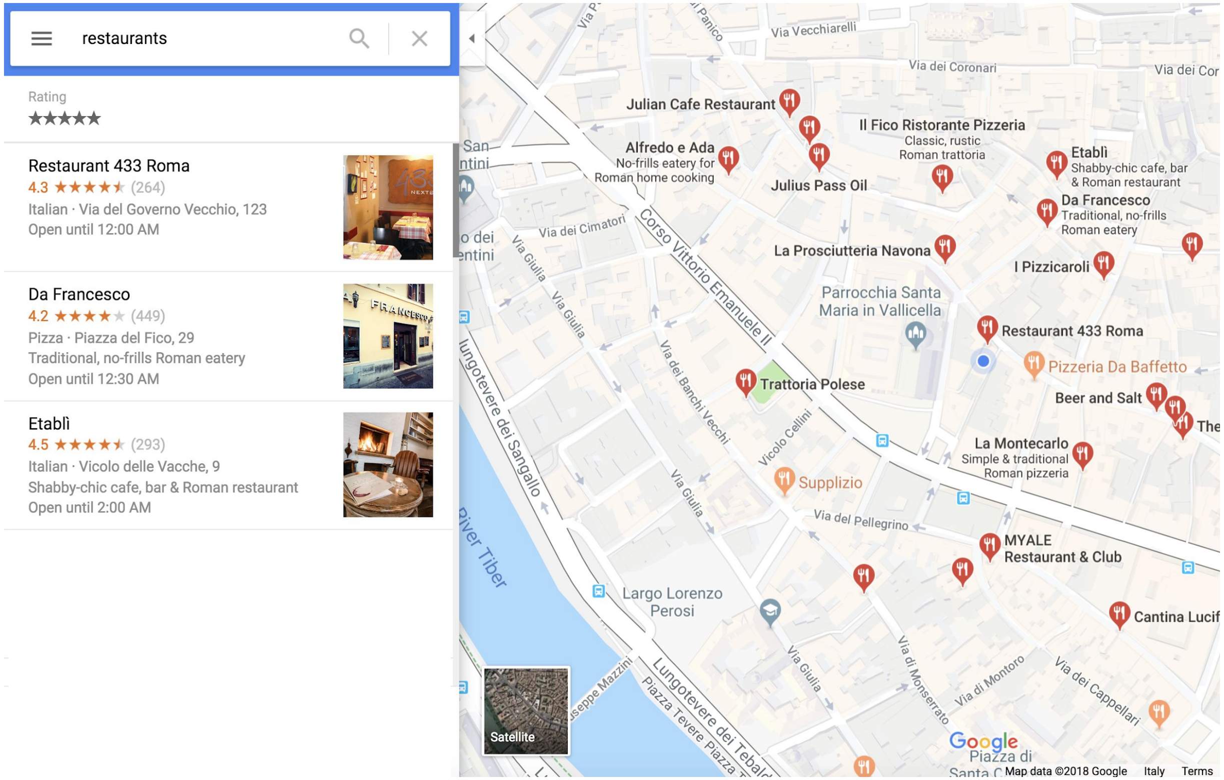
Approximate Representation

Real

0.45

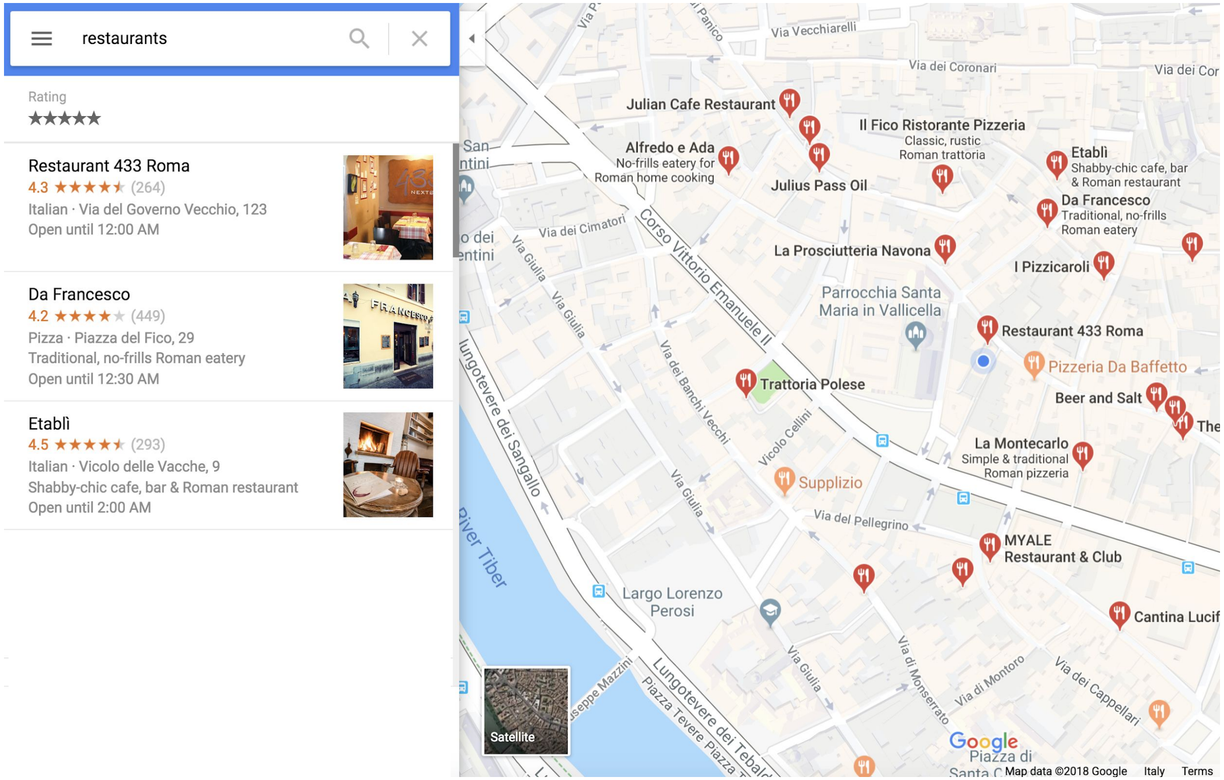
0.25

0.30



Approximate Representation

Apx	Real
0.46	0.45
0.23	0.25
0.31	0.30



Sketching a RUM

Let D be a RUM model on $[n]$
Let D_S be the winner distribution of D on S

Model A ε-approximates D if, for each $S \subseteq [n]$, $|D_S - A_S|_{TV} \leq \varepsilon$

Sketching a RUM

Let D be a RUM model on $[n]$
Let D_S be the winner distribution of D on S

Model A ε-approximates D if, for each $S \subseteq [n]$, $|D_S - A_S|_{TV} \leq \varepsilon$

Total Variation Distance

$|D_S - A_S|_{TV}$ is the maximum gap between event probabilities in D_S and A_S

Sketching a RUM

Let D be a RUM model on $[n]$
Let D_S be the winner distribution of D on S

Model A **ε-approximates** D if, for each $S \subseteq [n]$, $|D_S - A_S|_{TV} \leq \varepsilon$

Total Variation Distance

$|D_S - A_S|_{TV}$ is the maximum gap between event probabilities in D_S and A_S

(0.45,	0.25,	0.30)
1	*2*	*3*

(0.46,	0.23,	0.31)
1	*2*	*3*

Sketching a RUM

Let D be a RUM model on $[n]$
Let D_S be the winner distribution of D on S

Model A **ε-approximates** D if, for each $S \subseteq [n]$, $|D_S - A_S|_{TV} \leq \varepsilon$

Total Variation Distance

$|D_S - A_S|_{TV}$ is the maximum gap between event probabilities in D_S and A_S

(0.45, 0.25, 0.30) (0.46, 0.23, 0.31)
1 2 3 *1 2 3*

Consider the event: "The winner in {1,2,3} is 1 or 3"

Sketching a RUM

Let D be a RUM model on $[n]$
Let D_S be the winner distribution of D on S

Model A ε-approximates D if, for each $S \subseteq [n]$, $|D_S - A_S|_{TV} \leq \varepsilon$

Total Variation Distance

$|D_S - A_S|_{TV}$ is the maximum gap between event probabilities in D_S and A_S

(0.45, 0.25, 0.30) (0.46, 0.23, 0.31)
1 2 3 ***1 2 3***

Consider the event: "The winner in {1,2,3} is 1 or 3"

This event has probability 0.75 in D, and 0.77 in A

Sketching a RUM

Let *D* be a RUM model on *[n]*
Let D_S be the winner distribution of *D* on *S*

Model *A* ε-approximates *D* if, for each $S \subseteq [n]$, $|D_S - A_S|_{TV} \leq \varepsilon$

Total Variation Distance

$|D_S - A_S|_{TV}$ is the maximum gap between event probabilities in D_S and A_S

$|(0.45, 0.25, 0.30) - (0.46, 0.23, 0.31)|_{TV} = (0.01 + 0.02 + 0.01) / 2 = 0.02$

Sketching a RUM

Let *D* be a RUM model on *[n]*
Let D_S be the winner distribution of *D* on *S*

Model *A* ε-approximates *D* if, for each $S \subseteq [n]$, $|D_S - A_S|_{TV} \leq \varepsilon$

If we can find a model *A*,

- representable with few bits, and
- such that *A* ε-approximates *D*,

then we can efficiently sketch the RUM *D* to within Total Variation error ε

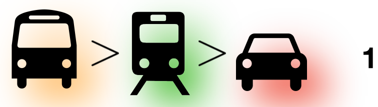
Sketching a RUM

- [CKT21] proves that each RUM *D* on *[n]* can be sketched to within TV error ε, using $O(\varepsilon^{-2} n^2 \log n)$ bits.

Sketching a RUM

- [CKT21] proves that each RUM *D* on *[n]* can be sketched to within TV error ε, using $O(\varepsilon^{-2} n^2 \log n)$ bits.

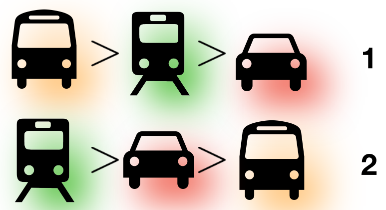
Repeatedly sample permutations from *D*



Sketching a RUM

- [CKT21] proves that each RUM *D* on *[n]* can be sketched to within TV error ε, using $O(\varepsilon^{-2} n^2 \log n)$ bits.

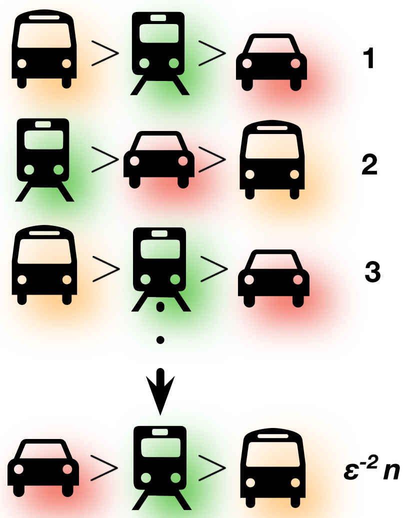
Repeatedly sample permutations from *D*



Sketching a RUM

- [CKT21] proves that each RUM *D* on *[n]* can be sketched to within TV error ε, using $O(\varepsilon^{-2} n^2 \log n)$ bits.

Repeatedly sample permutations from *D*

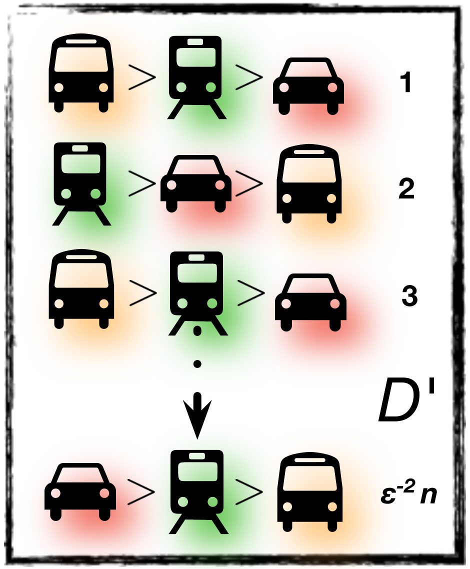


Sketching a RUM

- [CKT21] proves that each RUM *D* on *[n]* can be sketched to within TV error ε, using $O(\varepsilon^{-2} n^2 \log n)$ bits.

Repeatedly sample permutations from *D*

Let *D'* be the RUM obtained by imposing the uniform distribution on the sampled permutations

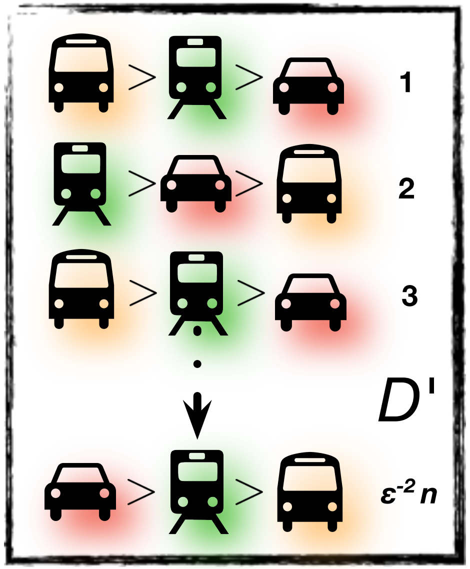


Sketching a RUM

- [CKT21] proves that each RUM *D* on *[n]* can be sketched to within TV error ε, using $O(\varepsilon^{-2} n^2 \log n)$ bits.

Repeatedly sample permutations from *D*

Let *D'* be the RUM obtained by imposing the uniform distribution on the sampled permutations



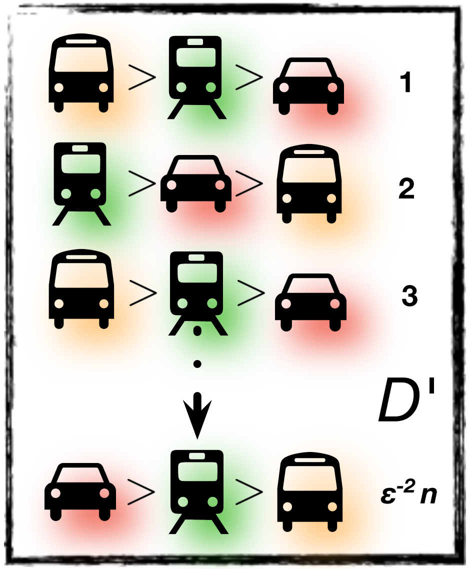
THM: D' sketches *D* to an ε-TV error, w.p. 1-o(1)

Sketching a RUM

- [CKT21] proves that each RUM *D* on *[n]* can be sketched to within TV error ε, using $O(\varepsilon^{-2} n^2 \log n)$ bits.

Repeatedly sample permutations from *D*

Let *D'* be the RUM obtained by imposing the uniform distribution on the sampled permutations



THM: D' sketches *D* to an ε-TV error, w.p. 1-o(1)

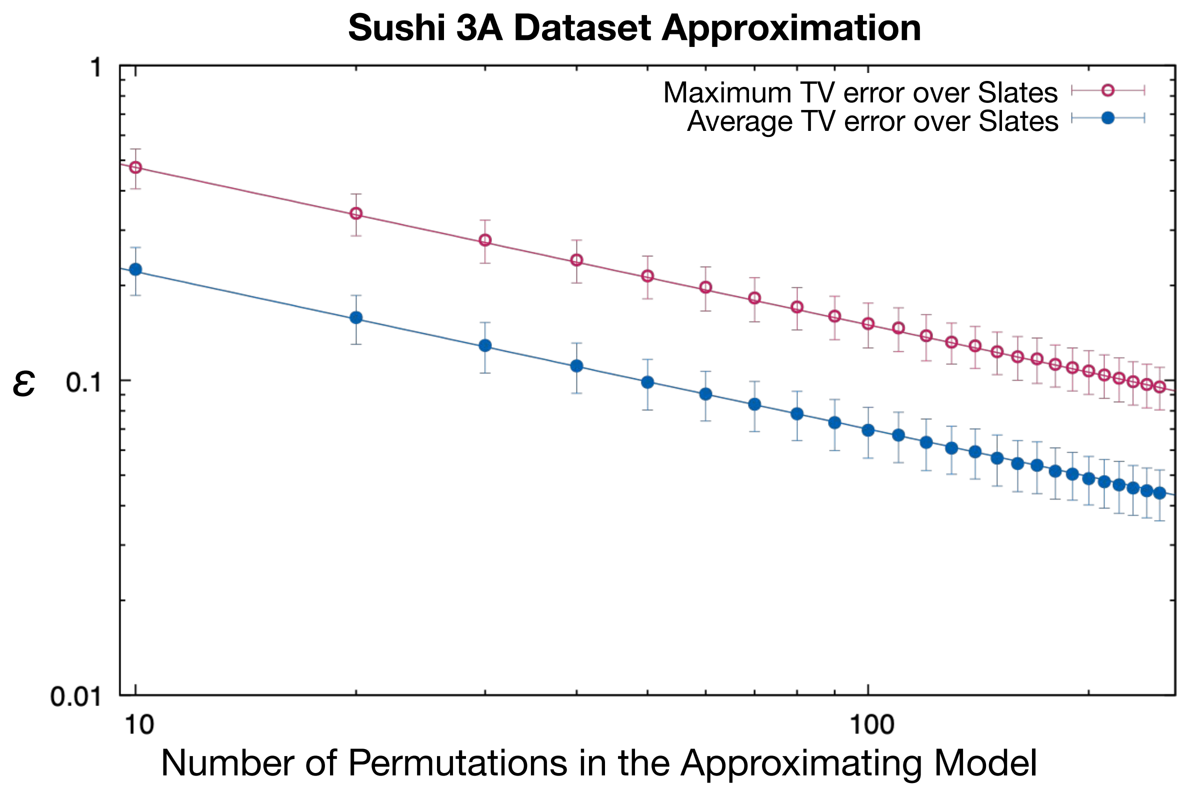
THM: D' can be represented with $O(\varepsilon^{-2} n^2 \log n)$ bits

Sketching a RUM

- [CKT21] proves that each RUM *D* on *[n]* can be sketched to within TV error ε, using $O(\varepsilon^{-2} n^2 \log n)$ bits.

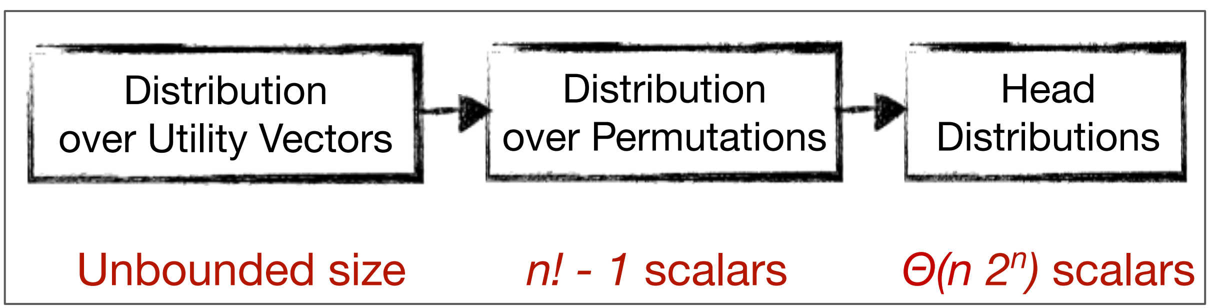
- [CKT21] also proves that one cannot sketch the generic RUM *D* on *[n]* to within TV error 0.01, using $o(n^2)$ bits.

Size of Model vs Approximation Error



Storing a RUM

- RUMs are powerful choice models, whose perfect representations require an exponential number of bits,

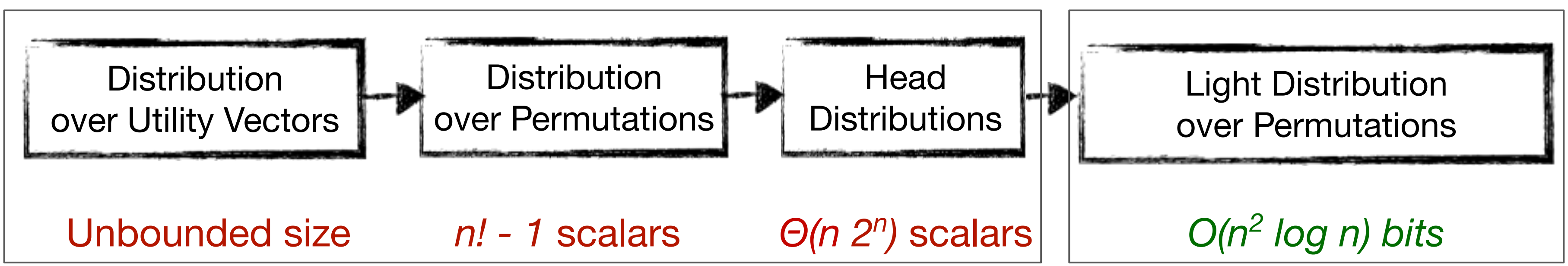


Exact Representations

Storing a RUM

- RUMs are powerful choice models, whose perfect representations require an exponential number of bits,

- but if one allows a tiny error, one can represent them efficiently with a number of bits bounded between $\Omega(n^2)$ and $O(n^2 \log n)$



Exact Representations

ϵ-Error

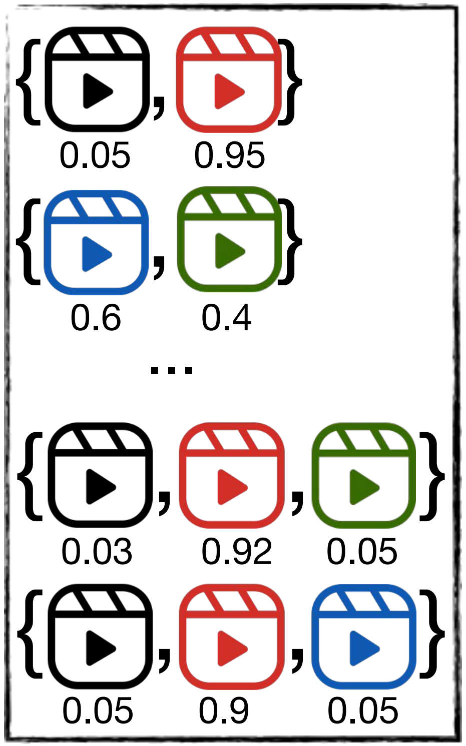
Fitting a RUM

Fitting a RUM

- In most practical applications, we do not observe the permutations, nor the utilities, of a RUM.

Fitting a RUM

- In most practical applications, we do not observe the permutations, nor the utilities, of a RUM. We only observe the probability distributions over the winners of the slates.

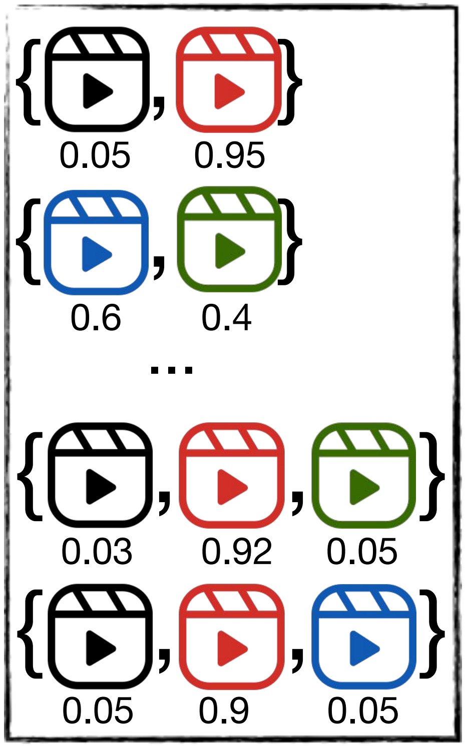


Fitting a RUM

- In most practical applications, we do not observe the permutations, nor the utilities, of a RUM. We only observe the probability distributions over the winners of the slates.

- Recall that $D_S(i)$ is the probability that item i gets selected as the winner of slate S, for $i \in S \subseteq [n]$

- How to fit a RUM to these observed "winner distributions"?



Fitting a RUM

- Let $\mathcal{S}_n$ be the set of permutations over *[n] = {1, 2, ..., n}*.
- Given a permutation $\pi \in \mathcal{S}_n$, and a slate $S \subseteq [n]$, let $\pi(S)$ be the topmost item of *S* in *π*.

Fitting a RUM

- Let $\mathcal{S}_n$ be the set of permutations over $[n] = \{1, 2, ..., n\}$.

- Given a permutation $\pi \in \mathcal{S}_n$, and a slate $S \subseteq [n]$, let $\pi(S)$ be the topmost item of S in π.

If $\pi = 3 > 1 > 2$ and $S = \{1, 2\}$, then $\pi(S) = 1$

Fitting a RUM

- Let $\mathcal{S}_n$ be the set of permutations over *[n] = {1, 2, ..., n}*.
- Given a permutation $\pi \in \mathcal{S}_n$, and a slate $S \subseteq [n]$, let $\pi(S)$ be the topmost item of *S* in *π*.
- If there exists a RUM representing the winner distributions such a RUM can be directly obtained by solving the following LP:

$$\begin{cases} \sum_{\substack{\pi \in \mathcal{S}_n \\ \pi(S)=i}} p_\pi & = D_S(i) & \forall i \in S \subseteq [n] \\ \sum_{\pi \in \mathcal{S}_n} p_\pi & = 1 & \\ p_\pi & \geq 0 & \forall \pi \in \mathcal{S}_n \end{cases}$$

Fitting a RUM

- This LP has *n!* many variables but it allows us to obtain a RUM compatible with the observed winner distributions in $n^{O(n)}$ time.

- The existence of this LP (and of this finite fitting procedure) is another advantage of the combinatorial-based view of RUMs.

$$\begin{cases} \sum_{\substack{\pi \in \mathcal{S}_n \\ \pi(S)=i}} p_\pi & = D_S(i) & \forall i \in S \subseteq [n] \\ \sum_{\pi \in \mathcal{S}_n} p_\pi & = 1 & \\ p_\pi & \geq 0 & \forall \pi \in \mathcal{S}_n \end{cases}$$

Efficiency of Fitting

- The "input" contains $\Omega(n\ 2^n)$ bits, thus a $n^{O(n)}$ algorithm (based on the permutation representation) is not too bad

Efficiency of Fitting

- The "input" contains $\Omega(n\,2^n)$ bits, thus a $n^{O(n)}$ algorithm (based on the permutation representation) is not too bad

In fact, by using a similar LP based on the Head distributions, one can obtain a "polytime" ($2^{O(n)}$) algorithm

Efficiency of Fitting

- The "input" contains $\Omega(n\ 2^n)$ bits, thus a $n^{O(n)}$ algorithm (based on the permutation representation) is not too bad

In fact, by using a similar LP based on the Head distributions, one can obtain a “polytime” ($2^{O(n)}$) algorithm

One can also obtain the RUM "closest" to the input data, if no perfect RUM exists

Efficiency of Fitting

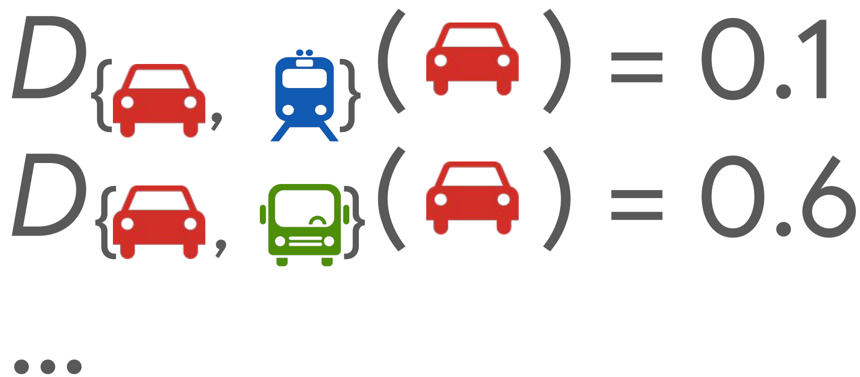
- The "input" contains $\Omega(n\ 2^n)$ bits, thus a $n^{O(n)}$ algorithm (based on the permutation representation) is not too bad

- But, in many real-world situations, one does not have access to the winner distributions of all the slates but only to the winner distributions of slates of small size

- Can one obtain a polynomial-time fitting algorithm in that case?

Pairwise Choices

- For simplicity, let us consider the case of slates of size 2.

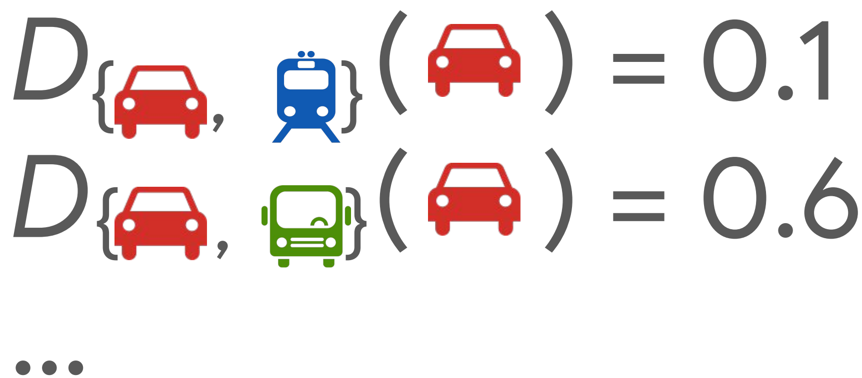


Pairwise Choices

- For simplicity, let us consider the case of slates of size 2.

- The input to the fitting problem is then a matrix

			
		0.1	0.6
	0.9		0.3
	0.4	0.7	



Pairwise Choices

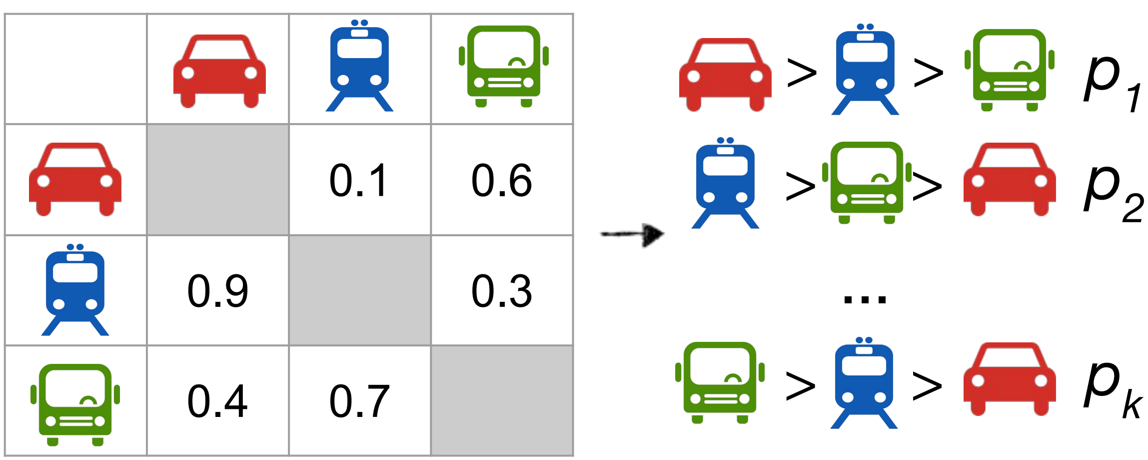
- For simplicity, let us consider the case of slates of size 2.
- The input to the fitting problem is then a matrix

			
		0.1	0.6
	0.9		0.3
	0.4	0.7	

- Many choice models have been proposed for representing tournament matrices:
  - *Blade-Chest* — Chen & Joachims, WSDM '16
  - *Majority Vote* — Makhijani & Ugander, WWW '19
  - *Two-level model* — Veerathu & Rajkumar, NeurIPS '21
  - ...

Pairwise Choices

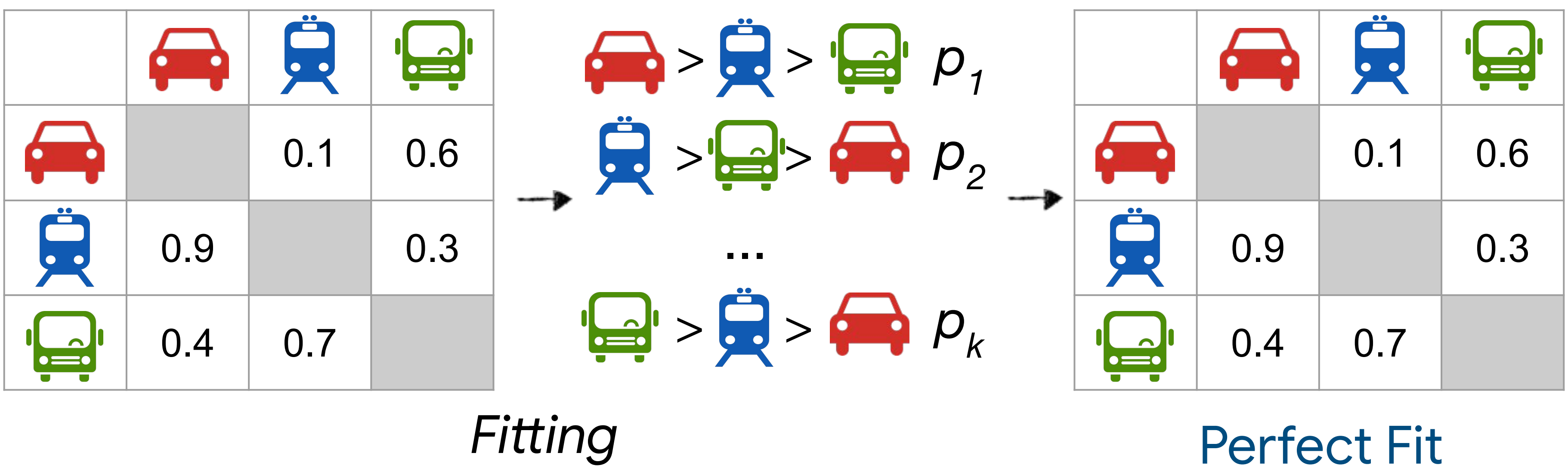
- For simplicity, let us consider the case of slates of size 2.
- The input to the fitting problem is then a matrix, and its output is a RUM



Fitting

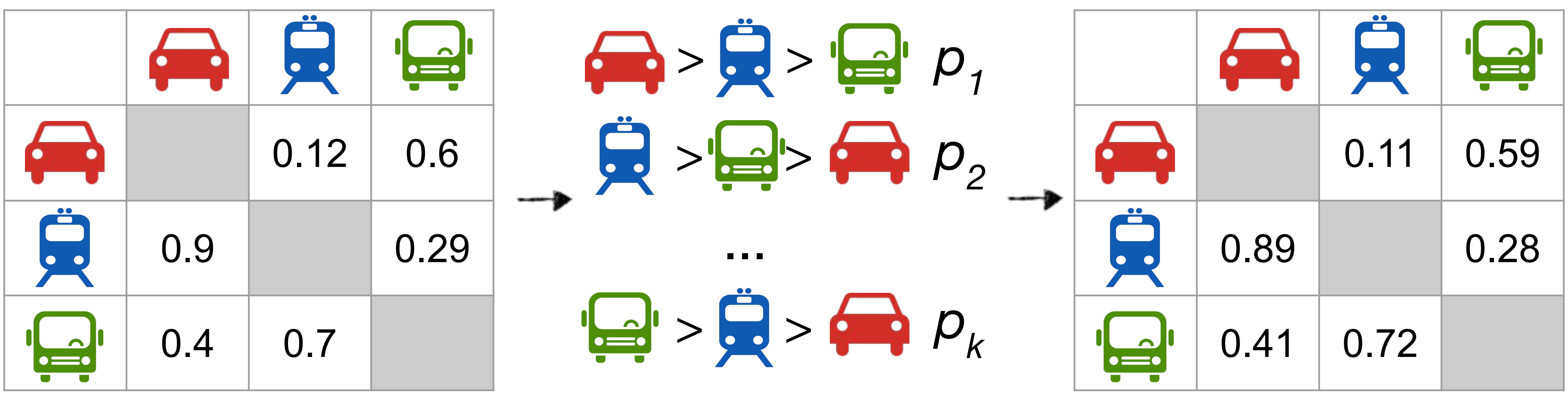
Pairwise Choices

- For simplicity, let us consider the case of slates of size 2.
- The input to the fitting problem is then a matrix, and its output is a RUM



Pairwise Choices

- For simplicity, let us consider the case of slates of size 2.
- The input to the fitting problem is then a matrix, and its output is a RUM



Fitting

Smallest "Error" Fit

Pairwise Choices

- A Linear Program for minimizing the average TV-error:

$$
\begin{cases}
\min \sum_{1 \le i < j \le n} \epsilon_{i,j} & & \\
\sum_{\substack{\pi \in \mathbf{S}_n \\ \pi(\{i,j\}) = i}} p_\pi = D_{\{i,j\}}(i) + e_{i,j} & & \forall 1 \le i < j \le n \\
\epsilon_{i,j} \ge -e_{i,j} & & \forall 1 \le i < j \le n \\
\epsilon_{i,j} \ge e_{i,j} & & \forall 1 \le i < j \le n \\
\sum_{\pi \in \mathbf{S}_n} p_\pi = 1 & & \\
p_\pi \ge 0 & & \forall \pi \in \mathbf{S}_n
\end{cases}
$$

The LP has exponentially many variables!

Pairwise Choices

- The Linear Program for minimizing the average TV-error has exponentially many variables, but only polynomially many constraints.
- Its dual then contains polynomially many variables.

Pairwise Choices

- The Linear Program for minimizing the average TV-error has exponentially many variables, but only polynomially many constraints.

- Its dual then contains polynomially many variables.

Primal LP **min** $c\,x$ **under** $A\,x \geq b$

Dual LP **max** $b\,y$ **under** $y\,A \leq c$

Pairwise Choices

- The Linear Program for minimizing the average TV-error has exponentially many variables, but only polynomially many constraints.
- Its dual then contains polynomially many variables.

Primal LP **min** $c\,x$ **under** $A\,x \geq b$

Dual LP **max** $b\,y$ **under** $y\,A \leq c$

Primal:
- 2 variables per pair of items
- 1 variable per permutation
- 3 constraints per pair of items
- 1 extra constraint

Dual:
- 2 constraints per pair of items
- 1 constraint per permutation
- 3 variables per pair of items
- 1 extra variable

Pairwise Choices

- The Linear Program for minimizing the average TV-error has exponentially many variables, but only polynomially many constraints.

- Its dual then contains polynomially many variables.

O(n!) vars
$O(n^2)$ constrs

Primal LP **min** $c\,x$ **under** $A\,x \geq b$
Dual LP **max** $b\,y$ **under** $y\,A \leq c$

O(n!) constrs
$O(n^2)$ vars

Primal:
- 2 variables per pair of items
- 1 variable per permutation
- 3 constraints per pair of items
- 1 extra constraint

Dual:
- 2 constraints per pair of items
- 1 constraint per permutation
- 3 variables per pair of items
- 1 extra variable

Pairwise Choices

- The Linear Program for minimizing the average TV-error has exponentially many variables, but only polynomially many constraints.

- Its dual then contains polynomially many variables.

Primal LP **min** $c\,x$ **under** $A\,x \geq b$
Dual LP **max** $b\,y$ **under** $y\,A \leq c$

Strong Duality Theorem: $c\,x^* = b\,y^*$

Pairwise Choices

- The Linear Program for minimizing the average TV-error has exponentially many variables, but only polynomially many constraints.

- Its dual then contains polynomially many variables.

- By means of the Ellipsoid method, if one could determine an unsatisfied dual constraint with a given solution, one would be able to optimize the primal and the dual - and, thus, find an optimal RUM.

Pairwise Choices

- The Linear Program for minimizing the average TV-error has exponentially many variables, but only polynomially many constraints.

- Its dual then contains polynomially many variables.

- By means of the Ellipsoid method, if one could
  solve the dual Separation Oracle Problem,
  one would be able to optimize the primal and the dual - and, thus, find an optimal RUM.

Separation Oracle

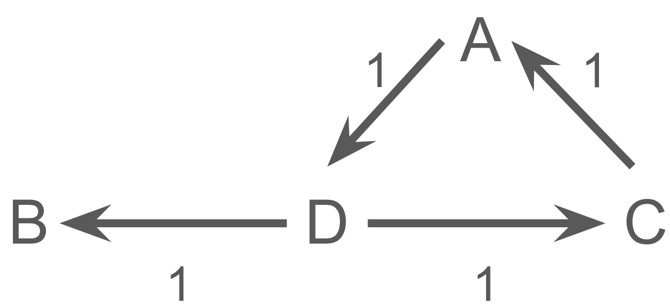
- [ACKPT] observe that the separation oracle problem for the dual of the Pairwise RUM LP is equivalent to the Weighted Minimum Feedback Arc Set (WMinFAS) problem:

    - sort the vertices of a weighted directed graph, with weights bounded in [0,1], so that the total weight of the arcs directed left-to-right is minimized.

Separation Oracle

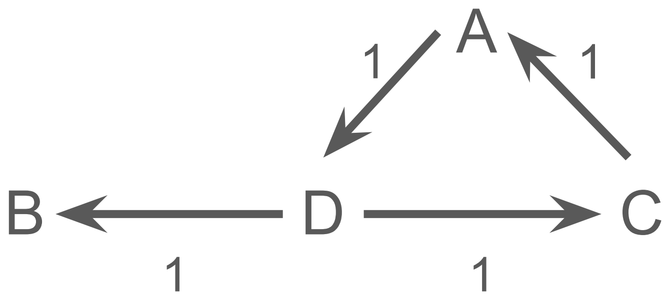
- [ACKPT] observe that the separation oracle problem for the dual of the Pairwise RUM LP is equivalent to the Weighted Minimum Feedback Arc Set (WMinFAS) problem:

    - sort the vertices of a weighted directed graph, with weights bounded in [0,1], so that the total weight of the arcs directed left-to-right is minimized.



Separation Oracle

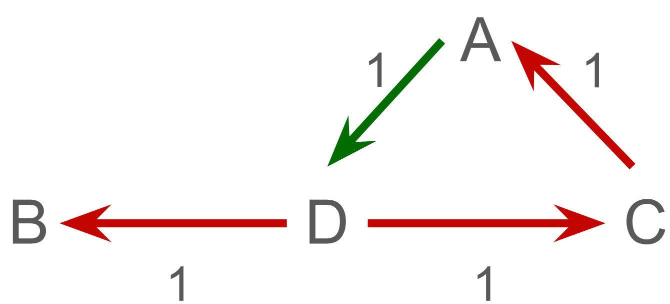
- [ACKPT] observe that the separation oracle problem for the dual of the Pairwise RUM LP is equivalent to the Weighted Minimum Feedback Arc Set (WMinFAS) problem:
  - sort the vertices of a weighted directed graph, with weights bounded in [0,1], so that the total weight of the arcs directed left-to-right is minimized.

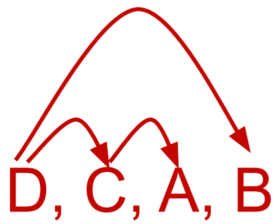


D, C, A, B

Separation Oracle

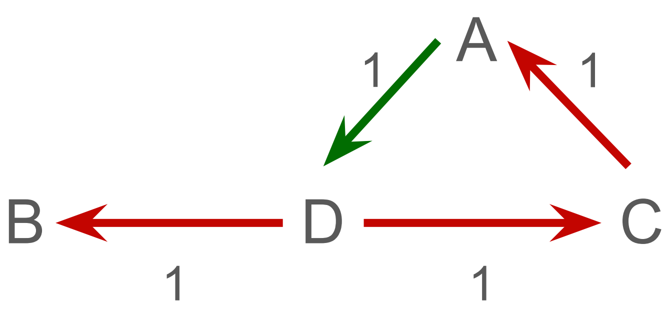
- [ACKPT] observe that the separation oracle problem for the dual of the Pairwise RUM LP is equivalent to the Weighted Minimum Feedback Arc Set (WMinFAS) problem:
    - sort the vertices of a weighted directed graph, with weights bounded in [0,1], so that the total weight of the arcs directed left-to-right is minimized.

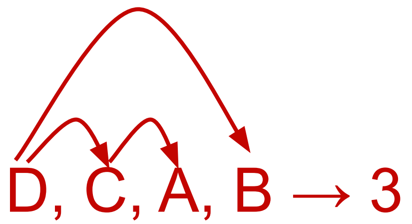




Separation Oracle

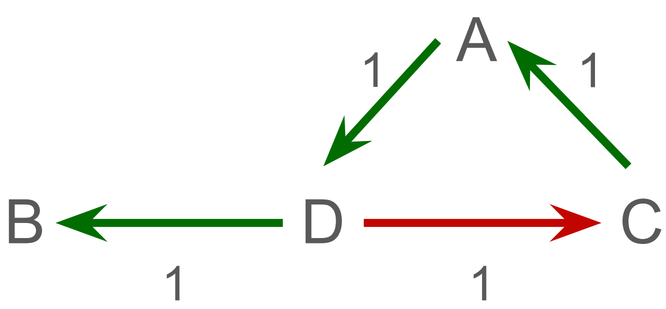
- [ACKPT] observe that the separation oracle problem for the dual of the Pairwise RUM LP is equivalent to the Weighted Minimum Feedback Arc Set (WMinFAS) problem:
    - sort the vertices of a weighted directed graph, with weights bounded in [0,1], so that the total weight of the arcs directed left-to-right is minimized.

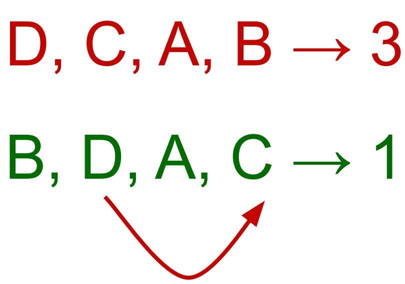




Separation Oracle

- [ACKPT] observe that the separation oracle problem for the dual of the Pairwise RUM LP is equivalent to the Weighted Minimum Feedback Arc Set (WMinFAS) problem:
  - sort the vertices of a weighted directed graph, with weights bounded in [0,1], so that the total weight of the arcs directed left-to-right is minimized.





Separation Oracle

- [ACKPT] observe that the separation oracle problem for the dual of the Pairwise RUM LP is equivalent to the Weighted Minimum Feedback Arc Set (WMinFAS) problem:
    - sort the vertices of a weighted directed graph, with weights bounded in [0,1], so that the total weight of the arcs directed left-to-right is minimized.
- MinFAS can be additively approximated to $O(\varepsilon n^2)$ in polynomial time for any constant $\varepsilon > 0$ [Frieze,Kannan,'99]

Approximate Separation Oracle

- [ACKPT] use this approximation algorithm for MinFAS to provide an Approximate Separation Oracle for the dual of the Pairwise LP.

Primal

$$
\begin{cases}
\min \sum_{1 \le i < j \le n} \epsilon_{i,j} & & \\
\sum_{\substack{\pi \in \mathbf{S}_n \\ \pi(\{i,j\})=i}} p_\pi & = D_{\{i,j\}}(i) + e_{i,j} & \forall 1 \le i < j \le n \\
\epsilon_{i,j} & \ge -e_{i,j} & \forall 1 \le i < j \le n \\
\epsilon_{i,j} & \ge e_{i,j} & \forall 1 \le i < j \le n \\
\sum_{\pi \in \mathbf{S}_n} p_\pi & = 1 & \\
p_\pi & \ge 0 & \forall \pi \in \mathbf{S}_n
\end{cases}
$$

Approximate Separation Oracle

- [ACKPT] use this approximation algorithm for MinFAS to provide an Approximate Separation Oracle for the dual of the Pairwise LP.

Primal
$$\begin{cases} \min \sum_{1\le i<j\le n} \epsilon_{i,j} & & \\ \sum_{\substack{\pi\in\mathbf{S}_n\\ \pi(\{i,j\})=i}} p_\pi & = D_{\{i,j\}}(i) + e_{i,j} & \forall 1\le i<j\le n \\ \epsilon_{i,j} & \ge -e_{i,j} & \forall 1\le i<j\le n \\ \epsilon_{i,j} & \ge e_{i,j} & \forall 1\le i<j\le n \\ \sum_{\pi\in\mathbf{S}_n} p_\pi & = 1 & \\ p_\pi & \ge 0 & \forall \pi\in\mathbf{S}_n \end{cases}$$

- [ACKPT] show that the Ellipsoid method, with this ASO, returns a RUM whose average TV-error is smaller than the min possible average TV-error plus ε, for any constant $\varepsilon > 0$.

Ellipsoid Method

- The Ellipsoid method, while being a polynomial time algorithm, is inefficient in practice.

Ellipsoid Method

- The Ellipsoid method, while being a polynomial time algorithm, is inefficient in practice.

- [ACKPT] also show experimentally that the Approximate Separation Oracle can be used in practice, via a cutting-plane framework, for solving pairwise-RUM fitting on many instances.

Dataset	n	avg. err.	lower bound on avg. err.
A5	16	0	
A9	12		
A17	13		
A48	10		
A81	11		
SF	35	0.001408	0.001408
Jester	100	0.000461	0

Fitting RUMs on Small Slates

- [CGKPT] show that the "pairwise" approach of [ACKPT] can be made to work on slates of size at most $k = O(1)$:
    - to obtain this result, they study a more general LP, and give an algorithm for a generalized version of MinFAS

Fitting RUMs on Small Slates

- [CGKPT] show that the "pairwise" approach of [ACKPT] can be made to work on slates of size at most $k = O(1)$:
  - to obtain this result, they study a more general LP, and give an algorithm for a generalized version of MinFAS
- They show that one can find, in polynomial time, a RUM whose average TV-error is not larger than the minimum possible average TV-error plus ε, for any constant $\varepsilon > 0$.

Fitting RUMs on Small Slates

- [CGKPT] show that the "pairwise" approach of [ACKPT] can be made to work on slates of size at most $k = O(1)$:
  - to obtain this result, they study a more general LP, and give an algorithm for a generalized version of MinFAS
- They show that one can find, in polynomial time, a RUM whose average TV-error is not larger than the minimum possible average TV-error plus ε, for any constant $\varepsilon > 0$.

Fitting so to minimize the **Average Error over the O(1)-slates**, can be ε-approximated in polynomial time

Fitting RUMs on Small Slates

- [CGKPT] show that the "pairwise" approach of [ACKPT] can be made to work on slates of size at most $k = O(1)$:
    - to obtain this result, they study a more general LP, and give an algorithm for a generalized version of MinFAS
- They show that one can find, in polynomial time, a RUM whose average TV-error is not larger than the minimum possible average TV-error plus ε, for any constant $\varepsilon > 0$.

Fitting so to minimize the **Average Error over the O(1)-slates**, can be ε-approximated in polynomial time

[ACKPT] show that the Approximate Separation Oracle for the **Maximum Error over the 2-slates** is NP-hard to approximate to within some additive constant

Learning a RUM

How well does a RUM fitted on slates of size at most k generalize to larger slates?

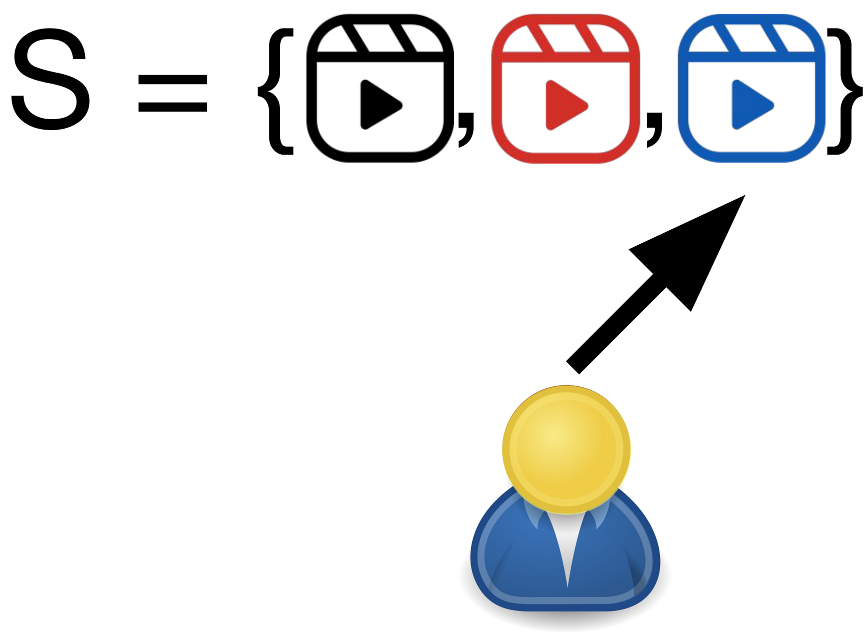
Streaming Services

- Streaming Services can test their users on small slates

$S = \{$  $\}$

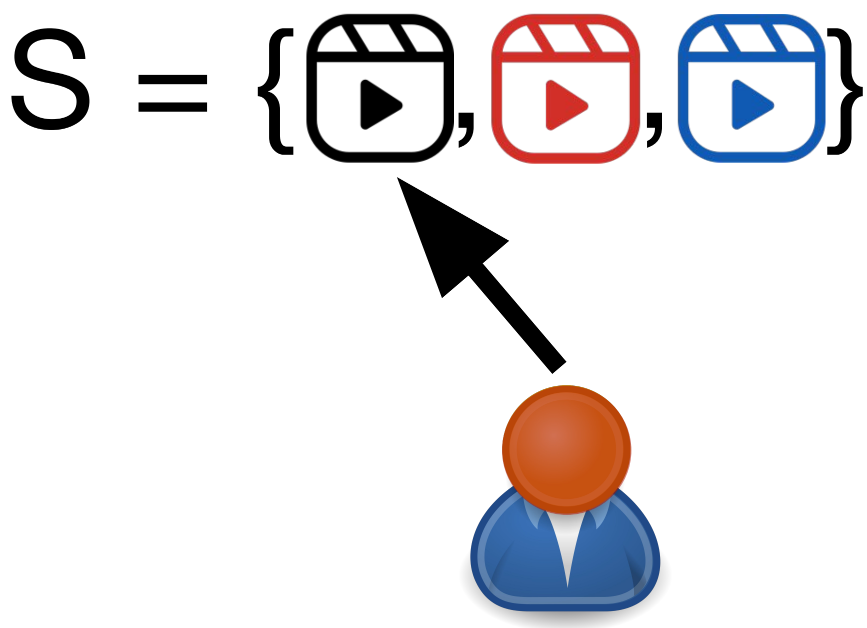
Streaming Services

- Streaming Services can test their users on small slates



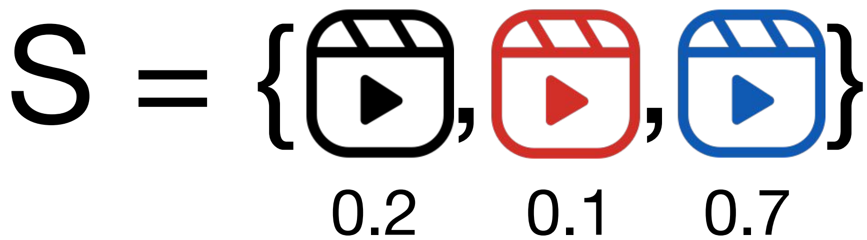
Streaming Services

- Streaming Services can test their users on small slates



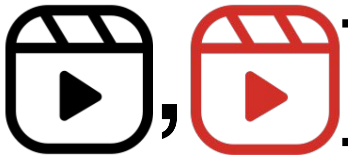
Streaming Services

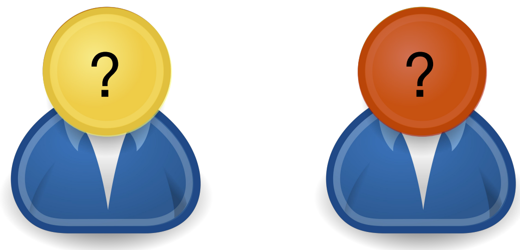
- Streaming Services can test their users on small slates



Streaming Services

- Streaming Services can test their users on small slates

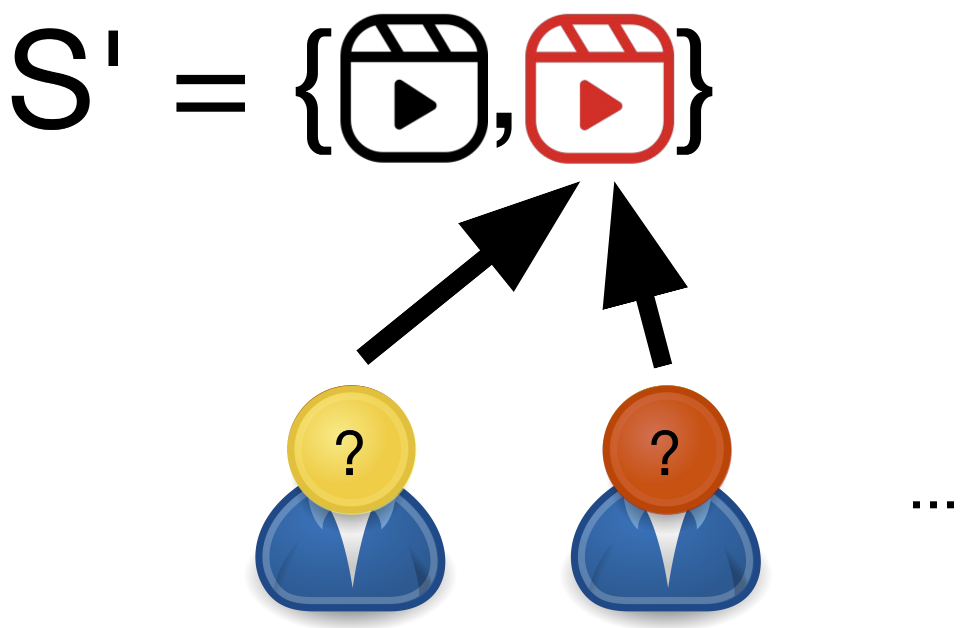
$S' = \{$  $\}$



...

Streaming Services

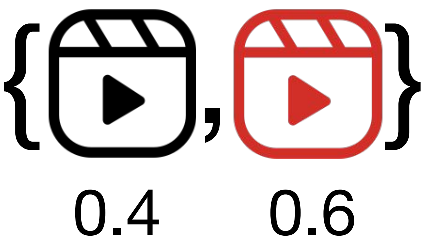
- Streaming Services can test their users on small slates



Streaming Services

- Streaming Services can test their users on small slates

S' =



Streaming Services

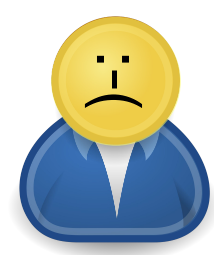
- Streaming Services can test their users on small slates

S'' = {  ,  }

0.8 0.2

Streaming Services

- Streaming Services can test their users on small slates
- It is impossible, though, to test the users on very large slates - very few users would parse through a list of, say, 1000 movies to find their preferred one



Streaming Services

- Streaming Services would love to pinpoint “gems” in their catalogues — items that are “most preferred” by a significant fraction of the user base

Streaming Services

- Streaming Services would love to pinpoint "gems" in their catalogues — items that are "most preferred" by a significant fraction of the user base

- Can they fit a RUM to what they observe on the small slates, and then use the RUM to guess the gems?

Generalization

- In recent work, [CKGPT] show that – by accessing slates of size at most $O\left(\sqrt{n \cdot \ln \frac{1}{\epsilon}}\right)$ – one can approximate, to within an ε TV-error, the winner distribution of all slates of size at most n

Generalization

- In recent work, [CKGPT] show that – by accessing slates of size at most $O\left(\sqrt{n \cdot \ln \frac{1}{\epsilon}}\right)$ – one can approximate, to within an ε TV-error, the winner distribution of all slates of size at most n

Accessing slates of size $O\left(\sqrt{n}\right)$ exposes
the structure of a RUM of n items to within a small error

Generalization

- In recent work, [CKGPT] show that – by accessing slates of size at most $O\left(\sqrt{n \cdot \ln \frac{1}{\epsilon}}\right)$ – one can approximate, to within an ε TV-error, the winner distribution of all slates of size at most n

Accessing slates of size $O\left(\sqrt{n}\right)$ exposes the structure of a RUM of n items to within a small error

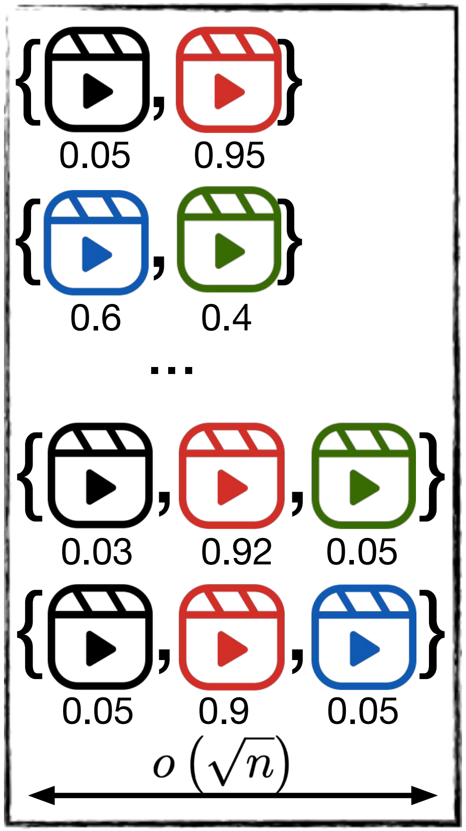
In particular, accessing slates of size $O\left(\sqrt{n}\right)$ allows one to discover **gems**

Generalization

- [CKGPT] also show that — if one can only access slates of size $o\left(\sqrt{n}\right)$ — then one cannot guess if an item has probability at most *ε*, or at least *1-ε*, in the slate *{1,2, ..., n}*.

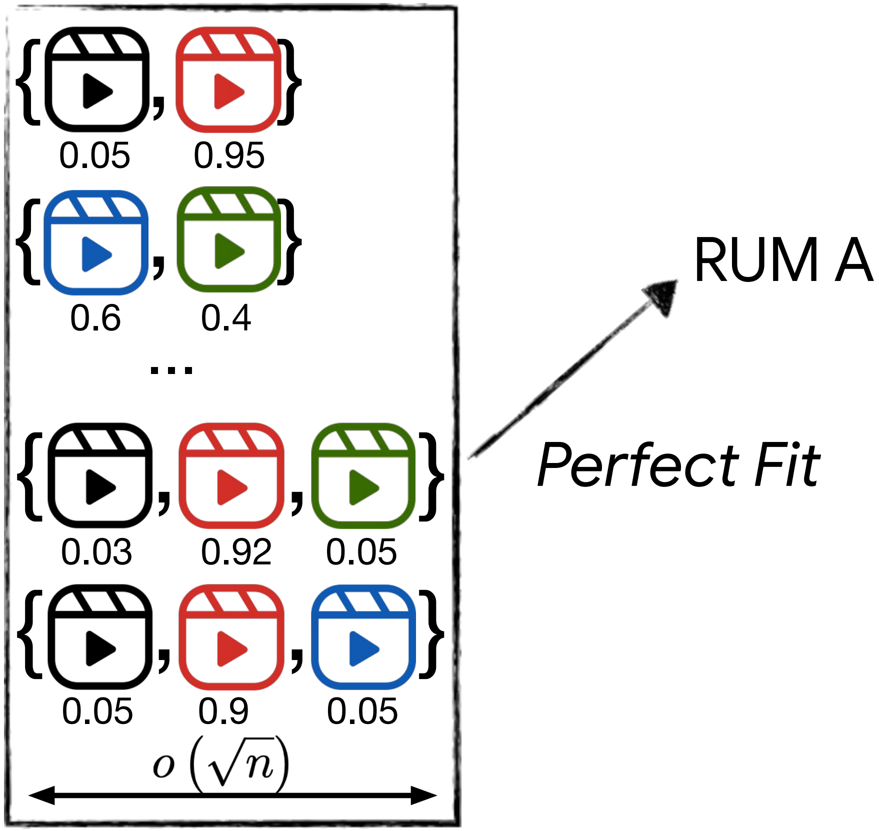
Generalization

- [CKGPT] also show that — if one can only access slates of size $o\left(\sqrt{n}\right)$ — then one cannot guess if an item has probability at most *ε*, or at least *1-ε*, in the slate *{1,2, ..., n}*.



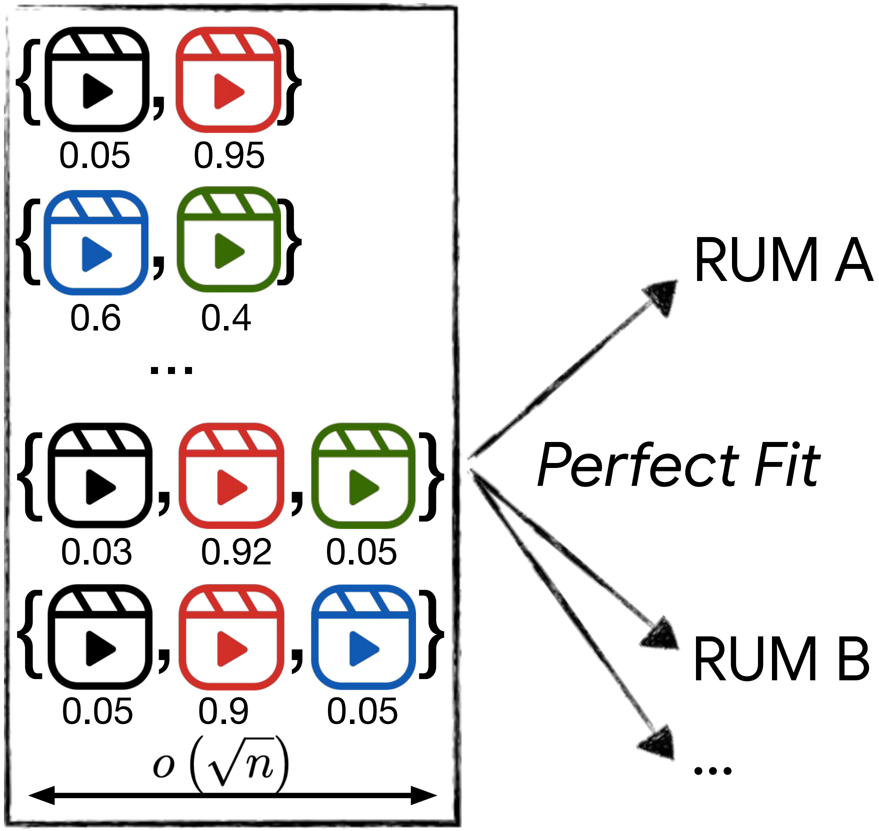
Generalization

- [CKGPT] also show that — if one can only access slates of size $o\left(\sqrt{n}\right)$ — then one cannot guess if an item has probability at most *ε,* or at least *1-ε,* in the slate *{1,2, ..., n}*.



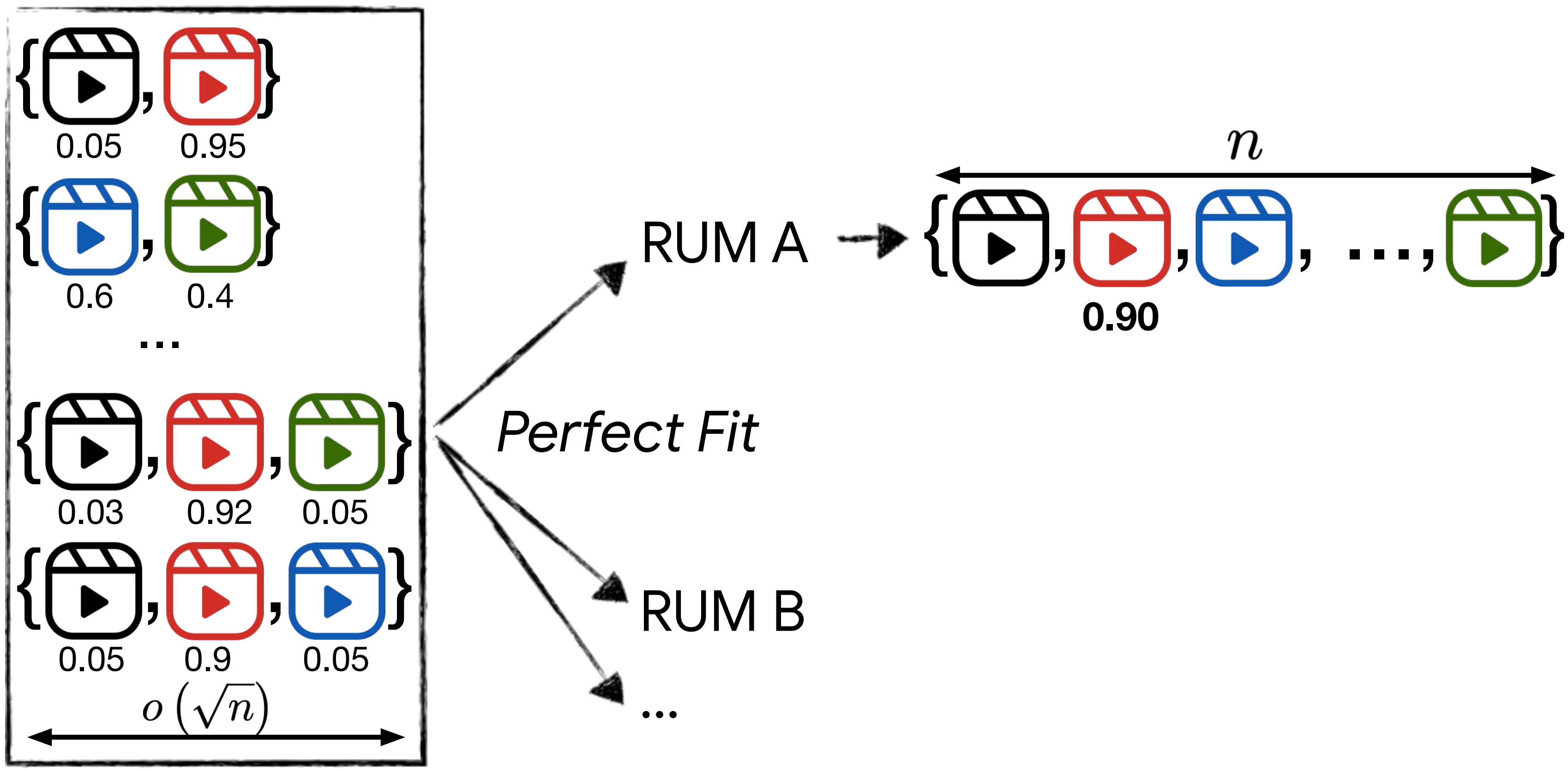
Generalization

- [CKGPT] also show that — if one can only access slates of size $o\left(\sqrt{n}\right)$ — then one cannot guess if an item has probability at most *ε,* or at least *1-ε,* in the slate *{1,2, ..., n}*.



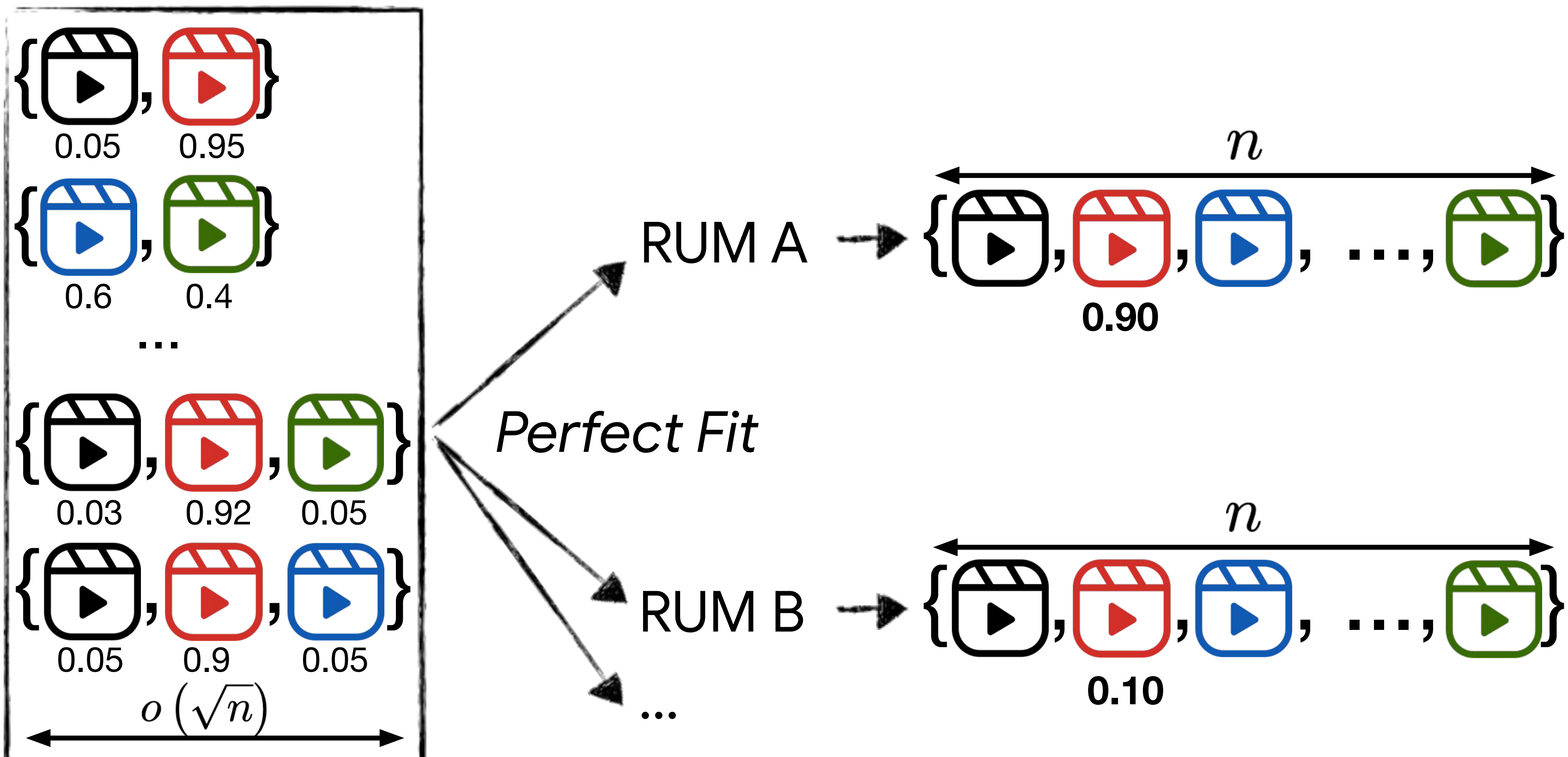
Generalization

- [CKGPT] also show that — if one can only access slates of size $o\left(\sqrt{n}\right)$ — then one cannot guess if an item has probability at most *ε,* or at least *1-ε,* in the slate *{1,2, ..., n}.*



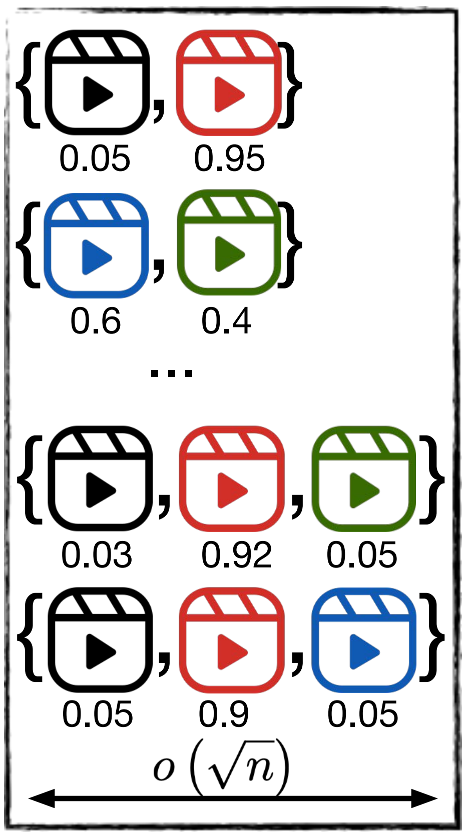
Generalization

- [CKGPT] also show that — if one can only access slates of size $o\left(\sqrt{n}\right)$ — then one cannot guess if an item has probability at most *ε*, or at least *1-ε*, in the slate *{1,2, ..., n}*.



Generalization

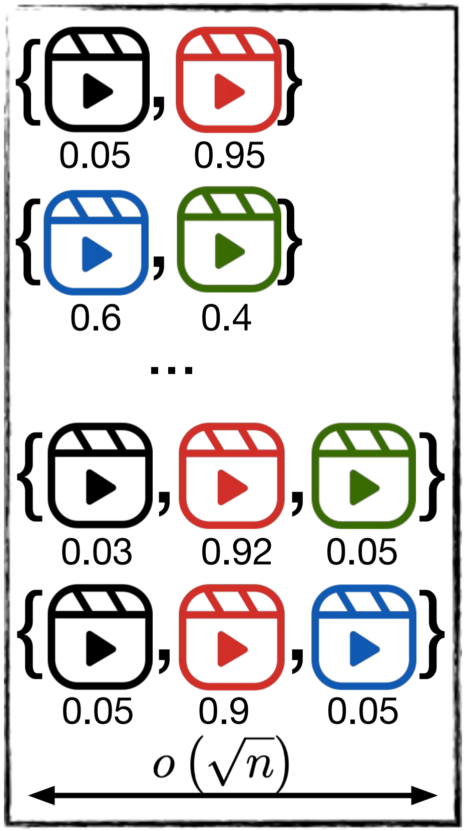
- [CKGPT] also show that — if one can only access slates of size $o\left(\sqrt{n}\right)$ — then one cannot guess if an item has probability at most *ε*, or at least *1-ε*, in the slate *{1,2, ..., n}*.



This data is insufficient to guess whether there exists a **gem** in the catalogue

Generalization

- [CKGPT] also show that — if one can only access slates of size $o(\sqrt{n})$ — then one cannot guess if an item has probability at most *ε*, or at least *1-ε*, in the slate *{1,2, ..., n}*.

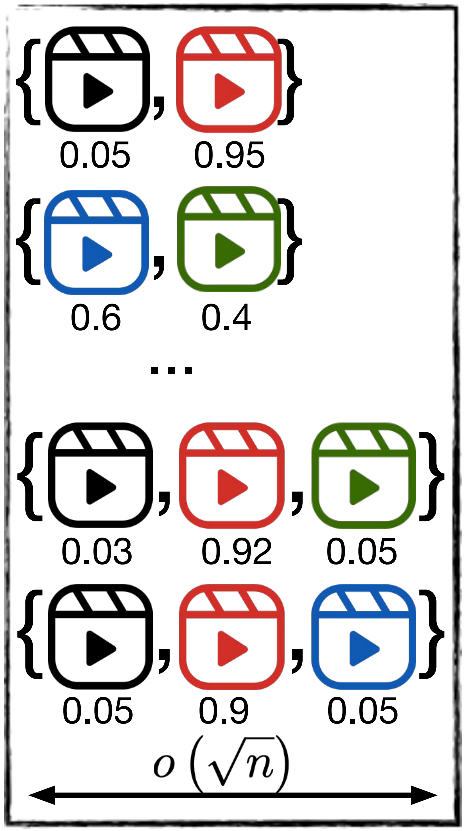


This data is insufficient to guess whether there exists a **gem** in the catalogue

But, as we said, increasing the bound on the slate size to just above $\sqrt{n}$ makes it possible to approximate all the winner distributions

Generalization

- [CKGPT] also show that — if one can only access slates of size $o(\sqrt{n})$ — then one cannot guess if an item has probability at most *ε,* or at least *1-ε,* in the slate *{1,2, ..., n}.*



This data is insufficient to guess whether there exists a **gem** in the catalogue

But, as we said, increasing the bound on the slate size to just above $\sqrt{n}$ makes it possible to approximate all the winner distributions and, thus, to find **gems**

A fourth representation!

- This result shows that one can approximately represent a RUM with its winner distributions of slates of size at most $\approx \sqrt{n}$
- While the size of this representation is very large ($n^{O(\sqrt{n})}$), constructing the RUM this way gets us quite an improvement in the runtime ($2^{O(\sqrt{n}\ln n)}$ vs $2^{O(n)}$) of RUM learning

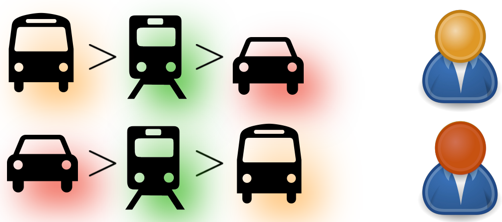
RUM Representations

- RUM Representations
  - Joint Utility Distribution
  - (Light) Distribution over Permutations
  - Head Distributions
  - Winner Distributions over slates of size at most $O(\sqrt{n})$
- They vary in their bit costs, and in the computational costs of various algorithmic tasks.
- **Choose your representation wisely! :-)**

Special Classes of RUMs

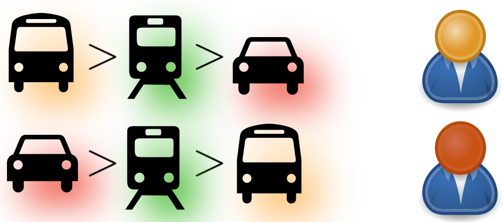
RUMs with Small Support

- Suppose that a RUM contains only *k* permutations in its support.



RUMs with Small Support

- Suppose that a RUM contains only *k* permutations in its support.
- Then, for each cardinality *c*, there can be at most *k* pairs (*H*,*s*), with *|H|* = *c*, such that $P_{H,s}$ is non-zero.



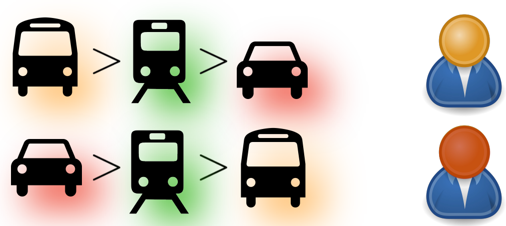
RUMs with Small Support

- Suppose that a RUM contains only *k* permutations in its support.
- Then, for each cardinality *c*, there can be at most *k* pairs (*H,s*), with *|H| = c*, such that $P_{H,s}$ is non-zero.

$P_{H,s}$ is the probability that a random permutation has the elements of *H*, in any order, as its |*H*| top-most elements, and that it has *s* in position |*H*|+*1*

RUMs with Small Support

- Suppose that a RUM contains only *k* permutations in its support.
- Then, for each cardinality *c*, there can be at most *k* pairs (*H*,*s*), with |*H*| = *c*, such that $P_{H,s}$ is non-zero.

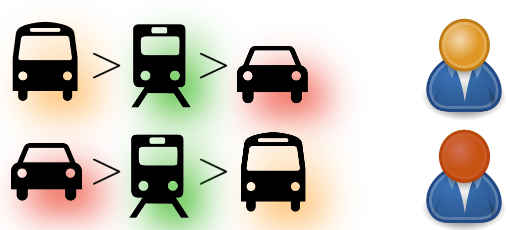


c = 0

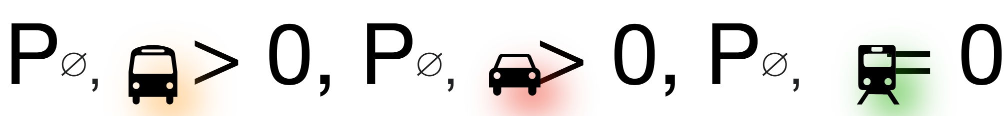
$P_{\varnothing,}$  > 0, $P_{\varnothing,}$  > 0

RUMs with Small Support

- Suppose that a RUM contains only *k* permutations in its support.
- Then, for each cardinality *c*, there can be at most *k* pairs (*H*,*s*), with |*H*| = *c*, such that $P_{H,s}$ is non-zero.

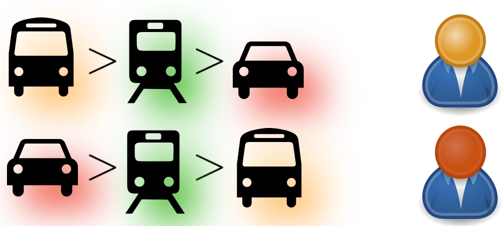


c = 0

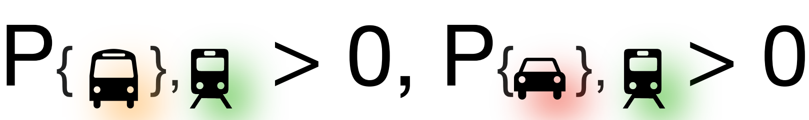


RUMs with Small Support

- Suppose that a RUM contains only *k* permutations in its support.
- Then, for each cardinality *c*, there can be at most *k* pairs (*H*,*s*), with *|H|* = *c*, such that $P_{H,s}$ is non-zero.

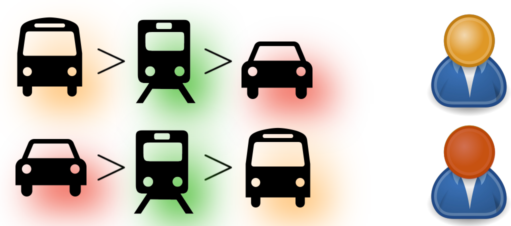


c = 1

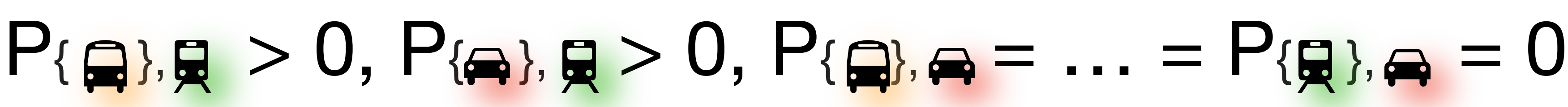


RUMs with Small Support

- Suppose that a RUM contains only *k* permutations in its support.
- Then, for each cardinality *c*, there can be at most *k* pairs (*H*,*s*), with *|H|* = *c*, such that $P_{H,s}$ is non-zero.



c = 1



RUMs with Small Support

- Suppose that a RUM contains only *k* permutations in its support.
- Then, for each cardinality *c*, there can be at most *k* pairs (*H,s*), with *|H|* = *c*, such that $P_{H,s}$ is non-zero.
- Thus, the formula $P_{H,s} = D_{[n]-H}(s) - \sum_{T \subset H} P_{T,s}$

  lets us learn the RUM with *O*(*n k*) max-dist queries.

Multinomial Logit MNL

- Classical special case of Random Utility Model
- Given a universe *U* of items and a positive weight a_i for each item *i* in *U*, the probability that *i* wins in the slate *S* is equal to

$$D_S(i) = \frac{a_i}{\sum_{j \in s} a_j}$$

Learning a MNL

For $i = 1, \ldots, n-1$, query the MNL using the slate $\{i, n\}$

obtaining the choice distribution $$\left(\frac{a_i}{a_i + a_n}, \frac{a_n}{a_i + a_n}\right)$$

Learning a MNL

For *i = 1, ..., n-1*, query the MNL using the slate *{i, n}*

obtaining the choice distribution $\left(\frac{a_i}{a_i + a_n}, \frac{a_n}{a_i + a_n}\right)$

System of equations

$$\frac{a_n}{a_1 + a_n} = D_{1,n}(n)$$
$$\frac{a_n}{a_2 + a_n} = D_{2,n}(n)$$
$$\vdots$$
$$\frac{a_n}{a_{n-1} + a_n} = D_{n-1,n}(n)$$
$$\sum_{i=1}^{n} a_i = 1$$

Learning a MNL

For *$i = 1, \ldots, n-1$*, query the MNL using the slate *$\{i, n\}$*

obtaining the choice distribution $\left(\frac{a_i}{a_i + a_n}, \frac{a_n}{a_i + a_n}\right)$

System of equations

$$\frac{a_n}{a_1 + a_n} = D_{1,n}(n)$$
$$\frac{a_n}{a_2 + a_n} = D_{2,n}(n)$$
$$\vdots$$
$$\frac{a_n}{a_{n-1} + a_n} = D_{n-1,n}(n)$$
$$\sum_{i=1}^{n} a_i = 1$$

→

$$a_n = D_{1,n}(n) \cdot (a_1 + a_n)$$
$$a_n = D_{2,n}(n) \cdot (a_2 + a_n)$$
$$\vdots$$
$$a_n = D_{n-1,n}(n) \cdot (a_{n-1} + a_n)$$
$$\sum_{i=1}^{n} a_i = 1$$

Full Rank LP

Learning a MNL

For $i = 1, \ldots, n-1$, query the MNL using the slate $\{i, n\}$

obtaining the choice distribution $\left(\frac{a_i}{a_i + a_n}, \frac{a_n}{a_i + a_n}\right)$

Querying $O(n)$ slates of size 2, and solving this LP, gets us a valid set of weights

$$
\begin{aligned}
a_n &= D_{1,n}(n) \cdot (a_1 + a_n) \\
a_n &= D_{2,n}(n) \cdot (a_2 + a_n) \\
&\vdots \\
a_n &= D_{n-1,n}(n) \cdot (a_{n-1} + a_n) \\
\sum_{i=1}^{n} a_i &= 1
\end{aligned}
$$

Full Rank LP

Mixture of MNLs

- MNL is insufficient to capture many practical settings
- 2-MNL mixture: Given a universe *U* of items and positive weights a_i and b_i for each item *i* in *U*

For a slate *S*, the probability of choosing *i* in *S* equals

$$\gamma \cdot \frac{a_i}{\sum_{j \in S} a_j} + (1 - \gamma) \cdot \frac{b_i}{\sum_{j \in S} b_j}$$

(Uniform mixture when $\gamma = 1/2$)

2-MNL Learning

- [CKT18] show that
  - Uniform 2-MNLs can be uniquely identified by the choice distributions of slates of sizes 2 and 3
  - There is a linear-time adaptive algorithm to learn the weights of uniform 2-MNLs using the choice distributions of slates of sizes 2 and 3

2-MNL Learning

- [CKT18] show that
  - Uniform 2-MNLs can be uniquely identified by the choice distributions of slates of sizes 2 and 3
  - There is a linear-time adaptive algorithm to learn the weights of uniform 2-MNLs using the choice distributions of slates of sizes 2 and 3

Compare with general RUMs where, as we showed, one needs slates of size $O\left(\sqrt{n}\right)$

2- and 3-Slates are sufficient

- **Theorem**
  For any uniform 2-MNL system, and for any set of 3 items $S = \{i, j, k\}$, the choice distributions of all the subsets of S determine uniquely the weights (up to rescaling) of i, j, k in each of the two MNLs.

Uniqueness

- This polynomial system induced by the choice distributions of the subsets of a generic set *{i,j,k}* has a unique solution

$$
\begin{cases}
\frac{a_i}{a_i+a_j} + \frac{b_i}{b_i+b_j} = 2D_{\{i,j\}}(i) \\
\frac{a_i}{a_i+a_k} + \frac{b_i}{b_i+b_k} = 2D_{\{i,k\}}(i) \\
\frac{a_j}{a_j+a_k} + \frac{b_j}{b_j+b_k} = 2D_{\{j,k\}}(j) \\
\frac{a_i}{a_i+a_j+a_k} + \frac{b_i}{b_i+b_j+b_k} = 2D_{\{i,j,k\}}(i) \\
\frac{a_j}{a_i+a_j+a_k} + \frac{b_j}{b_i+b_j+b_k} = 2D_{\{i,j,k\}}(j) \\
a_i + a_j + a_k = 1 \\
b_i + b_j + b_k = 1 \\
a_i, a_j, a_k, b_i, b_j, b_k > 0
\end{cases}
$$

Algorithmic Implications

- **Theorem**
  There exists an *adaptive* algorithm performing max-dist queries on *O(n) slates* of sizes 2 and 3, that reconstructs the weights of any uniform 2-MNL system on n elements.

Special Classes of RUMs

- RUMs supported on *k* permutations can be learned very efficiently
- Winner Distributions over slates of size at most $O(\sqrt{n})$ let you *approximately* represent any RUM
  - Winner Distributions over slates of size at most 2 let you represent any MNL
  - Winner Distributions over slates of size at most 3 let you represent any uniform 2-MNL
- What about *k*-MNLs? Are slates of size *O(k)* sufficient for representation?

Applications

Applications

- ML Applications
- Geographic Choice
- Choice on Graphs
- Reconsumption

Conjoint Analysis

Initially developed by [Luce and Tukey 1964] – axiomatic formulation

Picked up soon by marketers in late 60's, eg [Green and Rao 1971]

B-T model:

Flavor	Mango	Chocolate
Price	2.95	3.95
Size	120g	200g

Conjoint Analysis

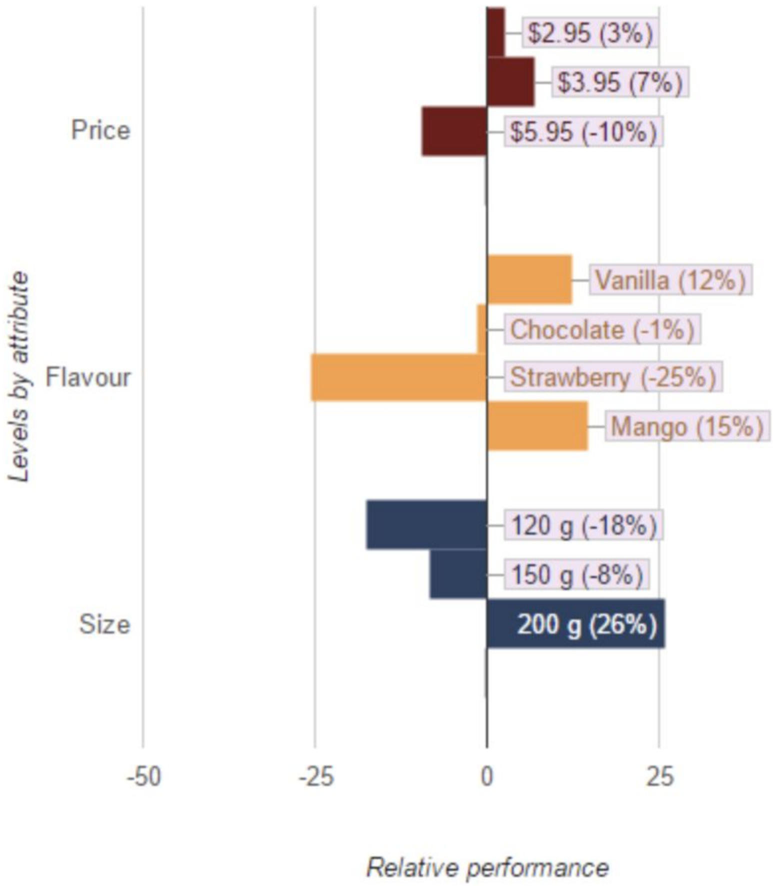
Initially developed by [Luce and Tukey 1964] – axiomatic formulation

Picked up soon by marketers in late 60's, eg [Green and Rao 1971]

B-T model:

Flavor	Mango		Chocolate
Price	2.95	>	3.95
Size	120g		200g

Conjoint Analysis Outcomes



Widely used in marketing

“Like giving dynamite to babies”

Influential case study on Marriott Courtyard hotels

Courtyard by Marriott

ROOM PRICE PER NIGHT IS $ 44.85

BUILDING SIZE, BAR/LOUNGE

Large (600 rooms) 12-story hotel with:

- Quiet bar/lounge
- Enclosed central corridors and elevators
- All rooms have very large windows

LANDSCAPING/COURT

Building forms a spacious outdoor courtyard

- View from rooms of moderately landscaped courtyard with:
  — many trees and shrubs
  — the swimming pool plus a fountain
  — terraced areas for sunning, sitting, eating

FOOD

Small moderately priced lounge and restaurant for hotel guests/friends

- Limited breakfast with juices, fruit, Danish, cereal, bacon and eggs
- Lunch—soup and sandwiches only
- Evening meal—salad, soup, sandwiches, six hot entrees including steak

HOTEL/MOTEL ROOM QUALITY

Quality of room furnishings, carpet, etc. is similar to:

- Hyatt Regencies
- Westin "Plaza" Hotels

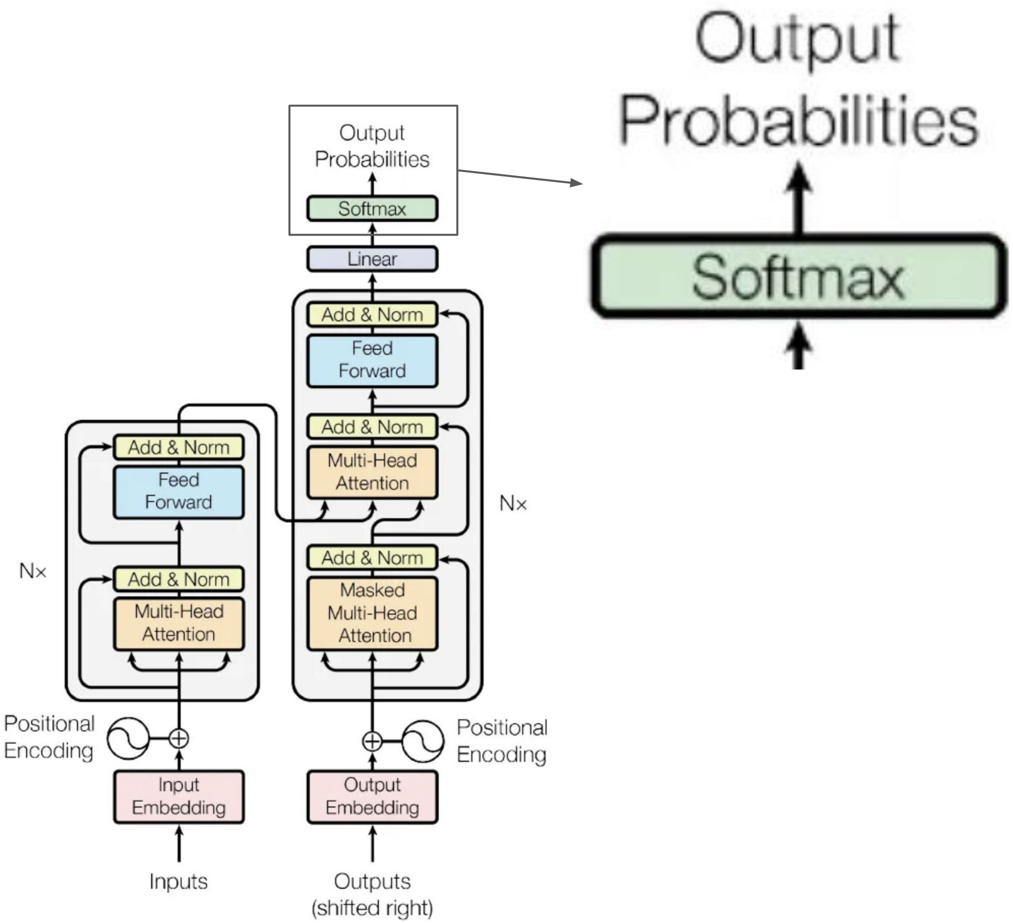
Courtyard by Marriott

Attribute	Levels	Description	Part Worths
Hotel Size	1	Small (125 rooms) 2-story hotel (.00)*	1.06
	2	12-story (600 rooms) with large lobby, meeting rooms, etc. (7.15)	0.00
Corridor/View	1	Outside stairs and walkways to all rooms. Restricted view. People walking outside window. (.00)	0.00
	2	Enclosed central corridors and stairs. Unrestricted view. Rooms have balcony or large window. (.65)	1.85
Pool Location	1	Not in courtyard (.00)	0.00
	2	In courtyard (.00)	1.37
Pool Type	1	No pool (.00)	0.61
	2	Rectangular pool (.45)	1.25
	3	Freeform pool (.50)	0.29
	4	Indoor/outdoor pool (.85)	0.00
Landscaping	1	Minimal landscaping (.00)	0.81
	2	Moderate landscaping (.10)	0.97
	3	Elaborate landscaping (.50)	0.00
Building Shape	1	"L" shape building with modest landscaping (.00)	0.00
	2	Building forms an outdoor landscaped courtyard for sitting, eating, sunning, etc. (.45)	0.37

*Figure in parentheses after each description = price premium.

Softmax and discrete choice

A generic transformer (from [Vaswani et al 2017])



Softmax bottleneck

[Yang et al, 2018]

$$\mathbf{H}_\theta = \begin{bmatrix} \mathbf{h}_{c_1}^\top \\ \mathbf{h}_{c_2}^\top \\ \cdots \\ \mathbf{h}_{c_N}^\top \end{bmatrix};\ \mathbf{W}_\theta = \begin{bmatrix} \mathbf{w}_{x_1}^\top \\ \mathbf{w}_{x_2}^\top \\ \cdots \\ \mathbf{w}_{x_M}^\top \end{bmatrix};\ \mathbf{A} = \begin{bmatrix} \log P^*(x_1|c_1), & \log P^*(x_2|c_1) & \cdots & \log P^*(x_M|c_1) \\ \log P^*(x_1|c_2), & \log P^*(x_2|c_2) & \cdots & \log P^*(x_M|c_2) \\ \vdots & \vdots & \ddots & \vdots \\ \log P^*(x_1|c_N), & \log P^*(x_2|c_N) & \cdots & \log P^*(x_M|c_N) \end{bmatrix}$$

$$F(\mathbf{A}) = \{\mathbf{A} + \mathbf{\Lambda}\mathbf{J}_{N,M} | \mathbf{\Lambda} \text{ is diagonal and } \mathbf{\Lambda} \in \mathbb{R}^{N \times N}\},$$

Property 1. *For any matrix $\mathbf{A}'$, $\mathbf{A}' \in F(\mathbf{A})$ if and only if Softmax$(\mathbf{A}') = P^*$. In other words, $F(\mathbf{A})$ defines the set of* ***all*** *possible logits that correspond to the true data distribution.*

Property 2. *For any $\mathbf{A}_1 \neq \mathbf{A}_2 \in F(\mathbf{A})$, $|rank(\mathbf{A}_1) - rank(\mathbf{A}_2)| \leq 1$. In other words, all matrices in $F(\mathbf{A})$ have similar ranks, with the maximum rank difference being 1.*

Goal of Languge Modeling: $\mathbf{H}_\theta \mathbf{W}_\theta^\top = \mathbf{A}'.$

Softmax bottleneck: rank of A’ is at most the embedding dimension d

Softmax bottleneck – another view

Consider two nearby word representations – very difficult to separate

All “usage patterns” must be embedded into R^d

Mixture of softmax

$$P_\theta(x|c) = \sum_{k=1}^{K} \pi_{c,k} \frac{\exp \mathbf{h}_{c,k}^\top \mathbf{w}_x}{\sum_{x'} \exp \mathbf{h}_{c,k}^\top \mathbf{w}_{x'}}; \text{ s.t. } \sum_{k=1}^{K} \pi_{c,k} = 1$$

MoS shows empirical wins over Softmax

The authors argue this is because it addresses the rank deficiency of the "softmax bottleneck"

Note that MoS is exactly mixed logit, and is there's equivalent to the full RUM family, where a user type is an assignment of "utilities" for each token

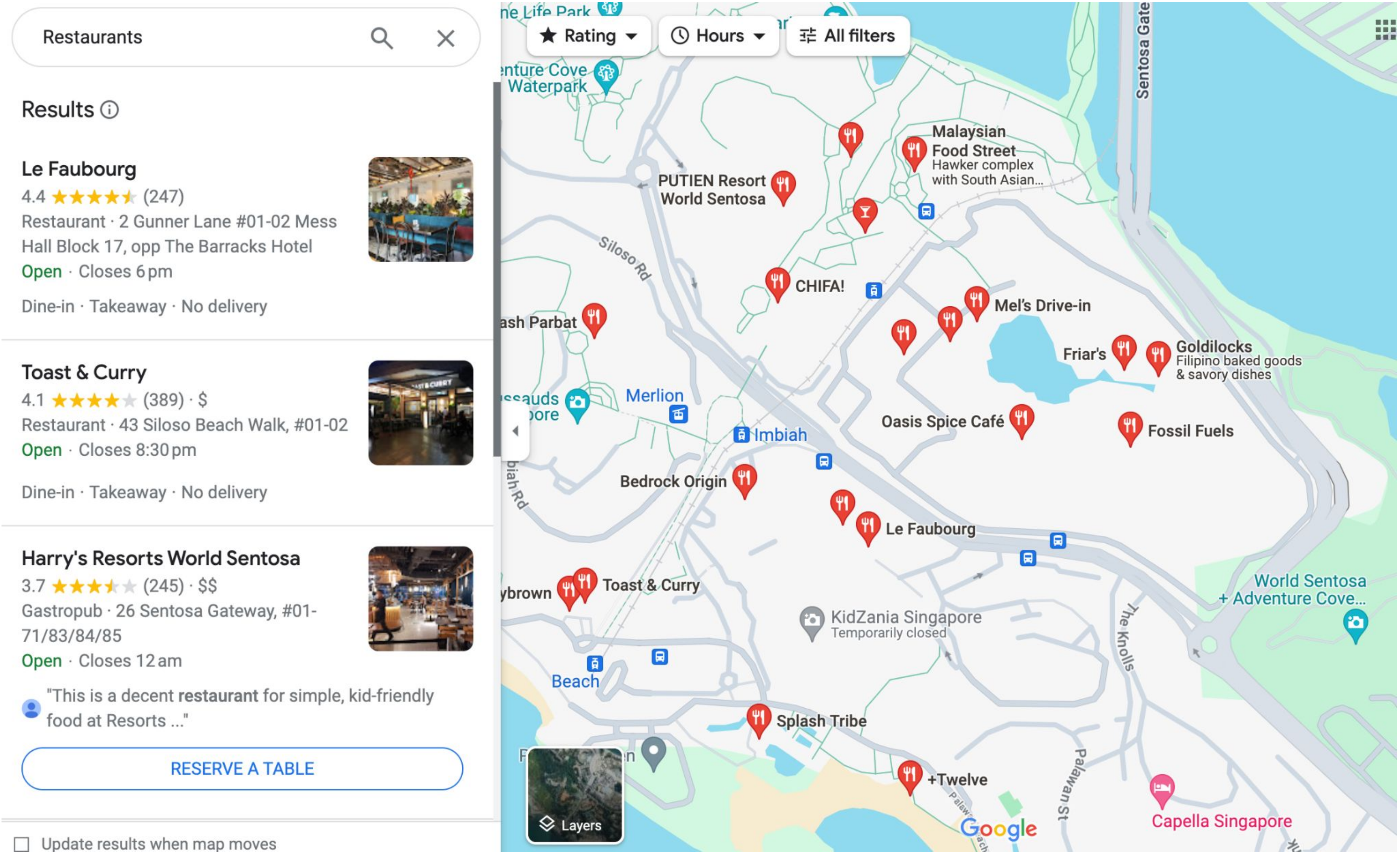
Utilities are a non-linear function of the context so far

Another take on the power of MNLs versus RUMs

Application: Geographic Choice

(or: where should we have dinner tonight?)

Where shall we eat tonight, revisited….



Some Factors in Restaurant Choice

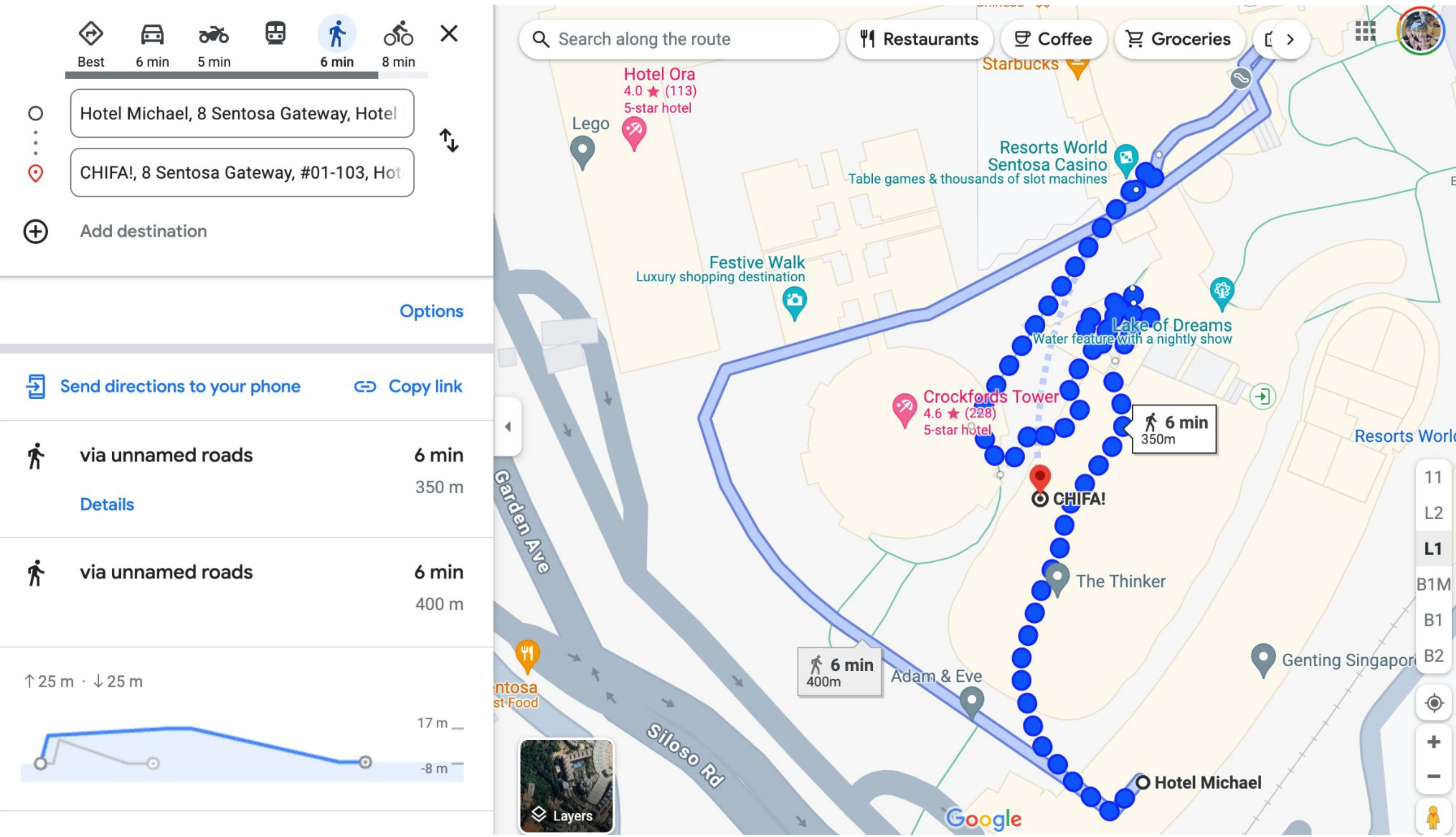
Deciding where to go for dinner:

- Quality of the restaurant
- Distance from Hotel Michael
- Price
- Cuisine type
- Ambience
- Time since last visit
- Opinions of dining companion(s)
- ...

Some data for this problem

Directions queries:

- Number of directions queries to US/Canadian restaurants in Google Maps
- Random sample of 15.5M queries to ~400K restaurants


Best
6 min
5 min
6 min
8 min
Hotel Michael, 8 Sentosa Gateway, Hotel
CHIFA!, 8 Sentosa Gateway, #01-103, Hot
Add destination
Options
Send directions to your phone
Copy link
via unnamed roads
6 min
350 m
Details
via unnamed roads
6 min
400 m
↑25 m · ↓25 m
17 m
-8 m
Search along the route
Restaurants
Coffee
Groceries
Starbucks
Hotel Ora
4.0 ★ (113)
5-star hotel
Lego
Resorts World Sentosa Casino
Table games & thousands of slot machines
Festive Walk
Luxury shopping destination
Lake of Dreams
Water feature with a nightly show
Crockfords Tower
4.6 ★ (228)
5-star hotel
6 min
350m
Resorts World
CHIFA!
The Thinker
6 min
400m
Adam & Eve
Genting Singapor
Hotel Michael
Garden Ave
Siloso Rd
Layers
11
L2
L1
B1M
B1
B2

Dataset

Directions queries:

- Number of directions queries to a US/Canadian restaurants in Google Maps
- Random sample of 15.5M queries to ~400K restaurants

Caveats:

- Not all visits have an associated directions search
  - Familiar locations
  - Spontaneous decisions
- Not all searches result in visits
  - Aspirational searches
  - Traffic & time estimates

Classical Discrete Choice Models

Recall our basic discrete choice model:

- Assign a score to each alternative
- Select with probability proportional to score

$$\Pr[x|A] = \frac{w_x}{\sum_{y \in A} w_y}; w_x = e^{V_x}$$

Goal:

- Better understand the score

Score function

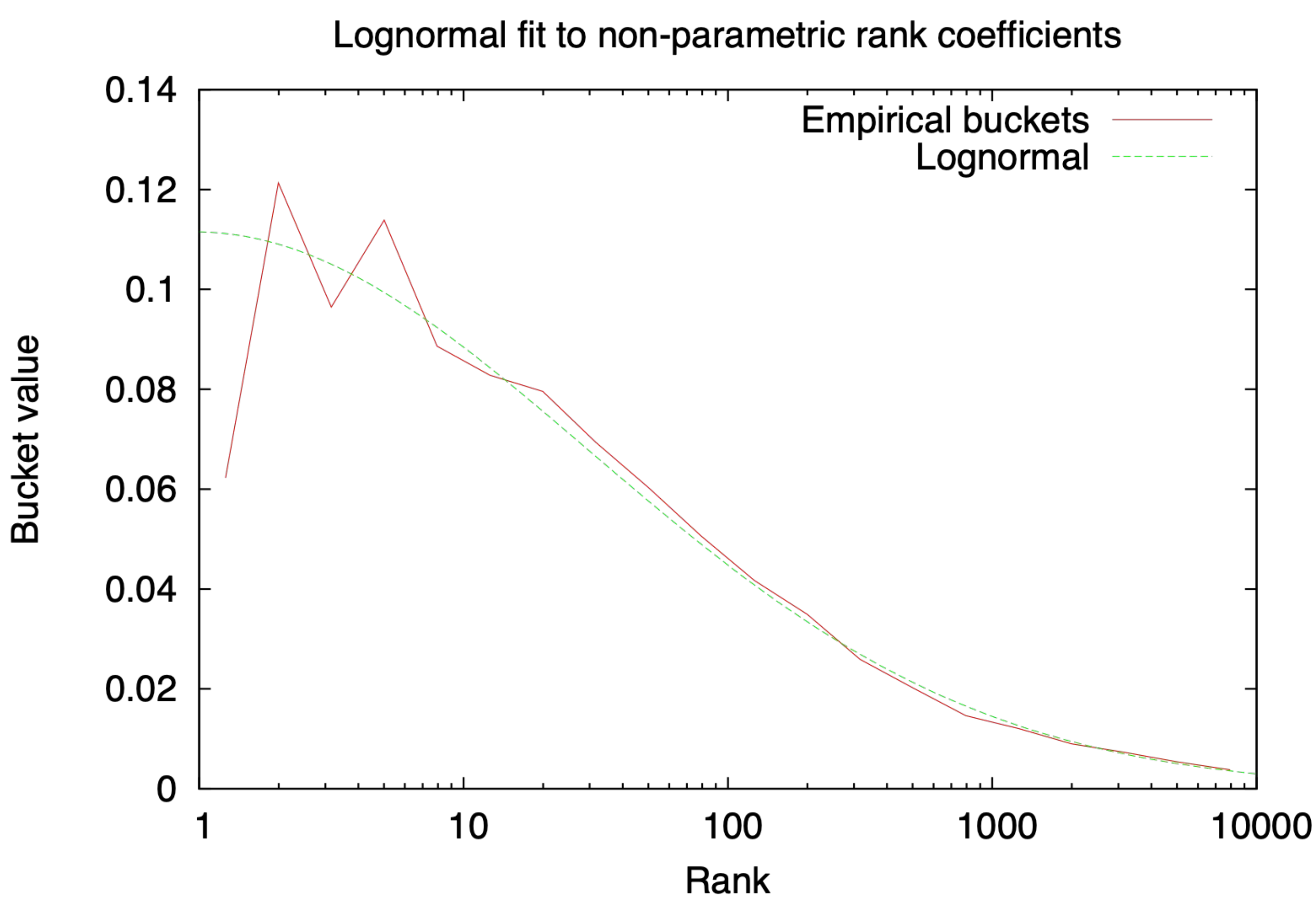
Today:

- Distance to the restaurant d
- Number of closer restaurants, rank: r
  - Captures density of restaurants
  - Acts as a proxy for the amount of competition
- Quality of particular restaurant: q
- Assume utility is linear in these features $V_x = d_x + r_x + q_x$

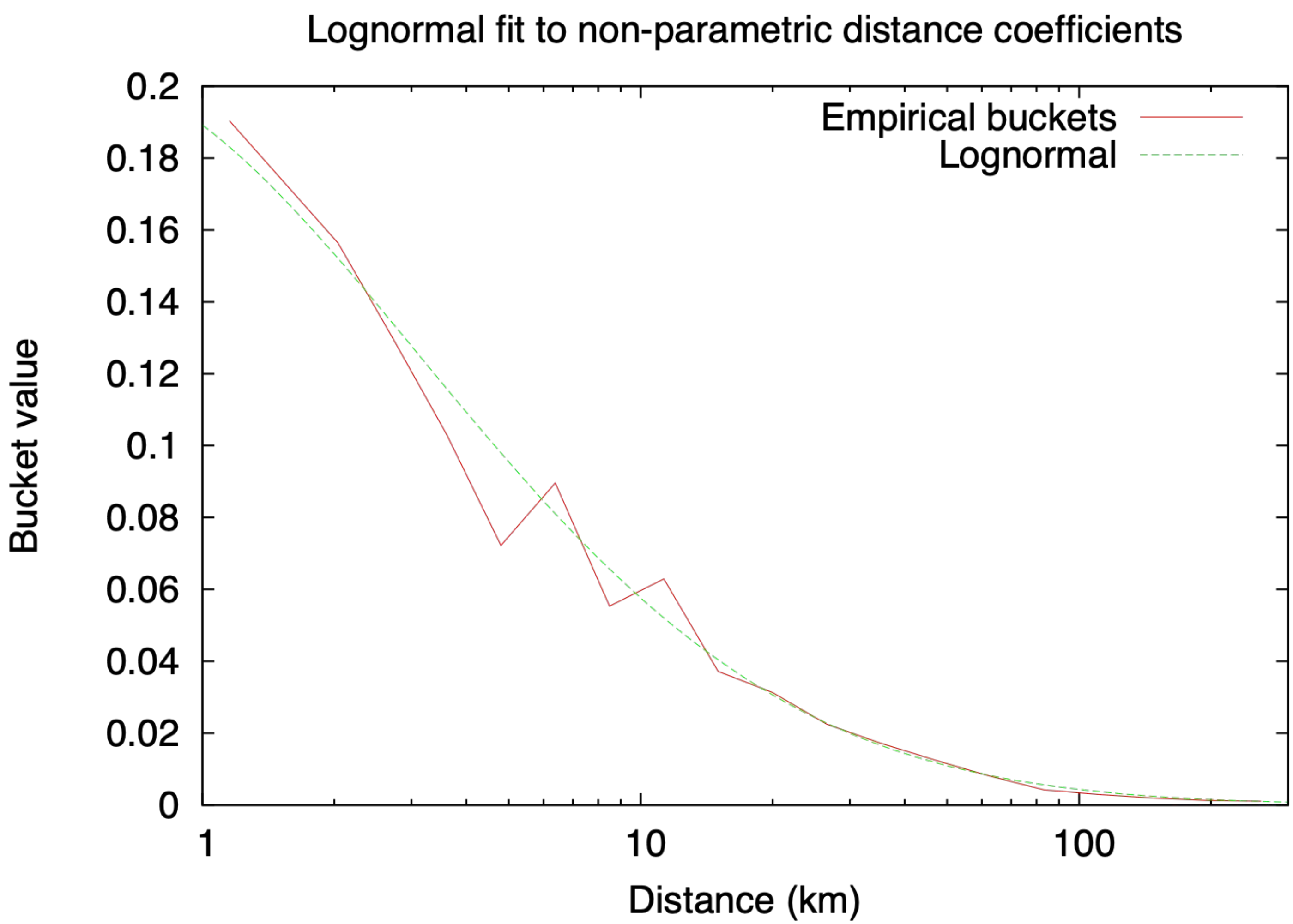
Not Today:

- Personal (user specific) preference
- Time since last visit
- Companions' desires

Imputed Rank Function



Imputed Distance Function



Results

Predict Likelihood on a held out test set:

Method	**Likelihood**
Uniform choice	1.1
Distance only model	3.9
Rank only model	4.6

Model

Fit both rank and distance functions by log-normals

- Four parameter model: $\mu_{\text{rank}}, \sigma^2_{\text{rank}}, \mu_{\text{distance}}, \sigma^2_{\text{distance}}$

$$s_i = \frac{1}{r_i \sigma_{\text{rank}}} \exp\left(-\frac{(\ln r_i - \mu_{\text{rank}})^2}{2\sigma^2_{\text{rank}}}\right) \cdot \frac{1}{d_i \sigma_{\text{distance}}} \exp\left(-\frac{(\ln d_i - \mu_{\text{distance}})^2}{2\sigma^2_{\text{distance}}}\right)$$

Results

Predict Likelihood on a held out test set:

Method	Likelihood
Uniform choice	1.1
Distance only model	3.9
Rank only model	4.6
Lognormal coefficient fit (4 parameters)	5.1

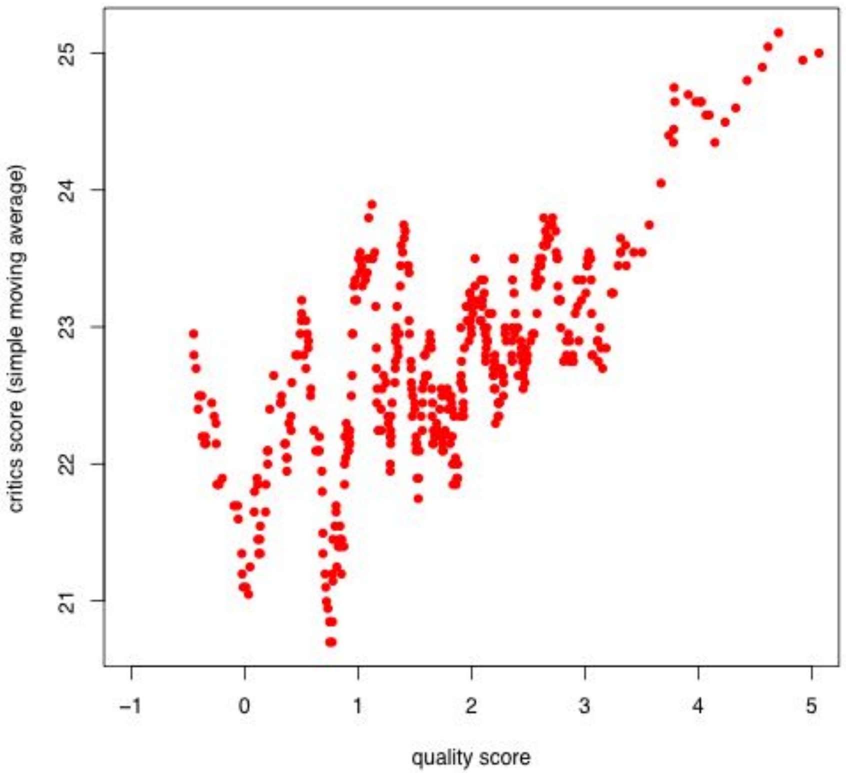
Results

Predict Likelihood on a held out test set:

Method	Likelihood
Uniform choice	1.1
Distance only model	3.9
Rank only model	4.6
Lognormal coefficient fit (4 parameters)	5.1
Non-parametric factored model	5.3

Quality Factor

- Quality is restaurant specific, makes the model much richer
- Learn it as the residual on ranks, distances
- Evaluation: correlation with critics' scores



Geographic Choice: what have we seen?

Multinomial Logistic Regression with buckets is a powerful technique to assess influence of features based on intensity

Captured interactions may give significantly different influence weights than feature correlations

Given the output of such models, it is possible to observe deeper structure

From this structure, we may find models that are far more parsimonious (why lognormal?)

These new models are much easier to fit when data is sparse

Application: Graphs

Ravi Kumar, Andrew Tomkins, Sergei Vassilvitskii and Erik Vee

[Ref: WSDM 2015]

Reverse Engineering a Markov Chain

Random Walks & Markov Chains

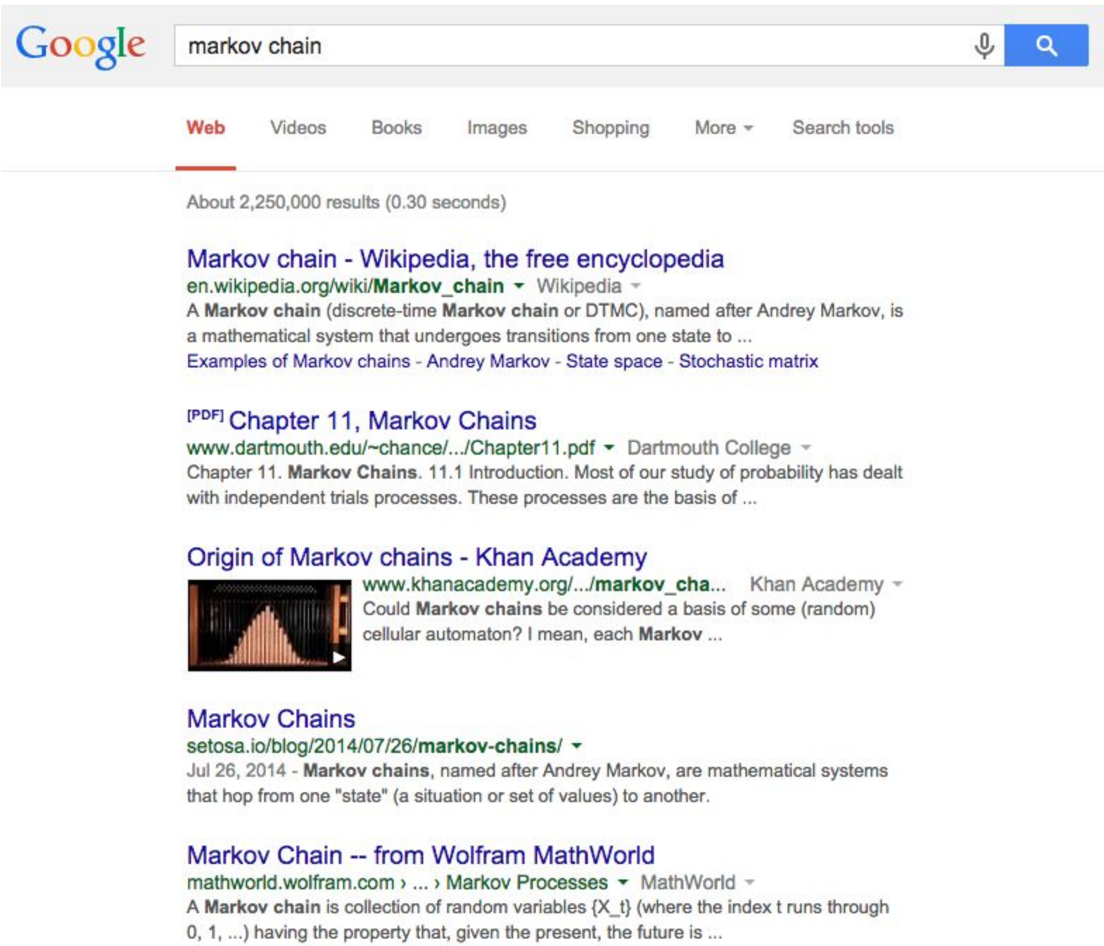
Markov Chains in Data Analysis:

- Simple, yet capture a lot of interactions
- Typically: compute & use the stationary distribution
- Beautiful theory with great applications

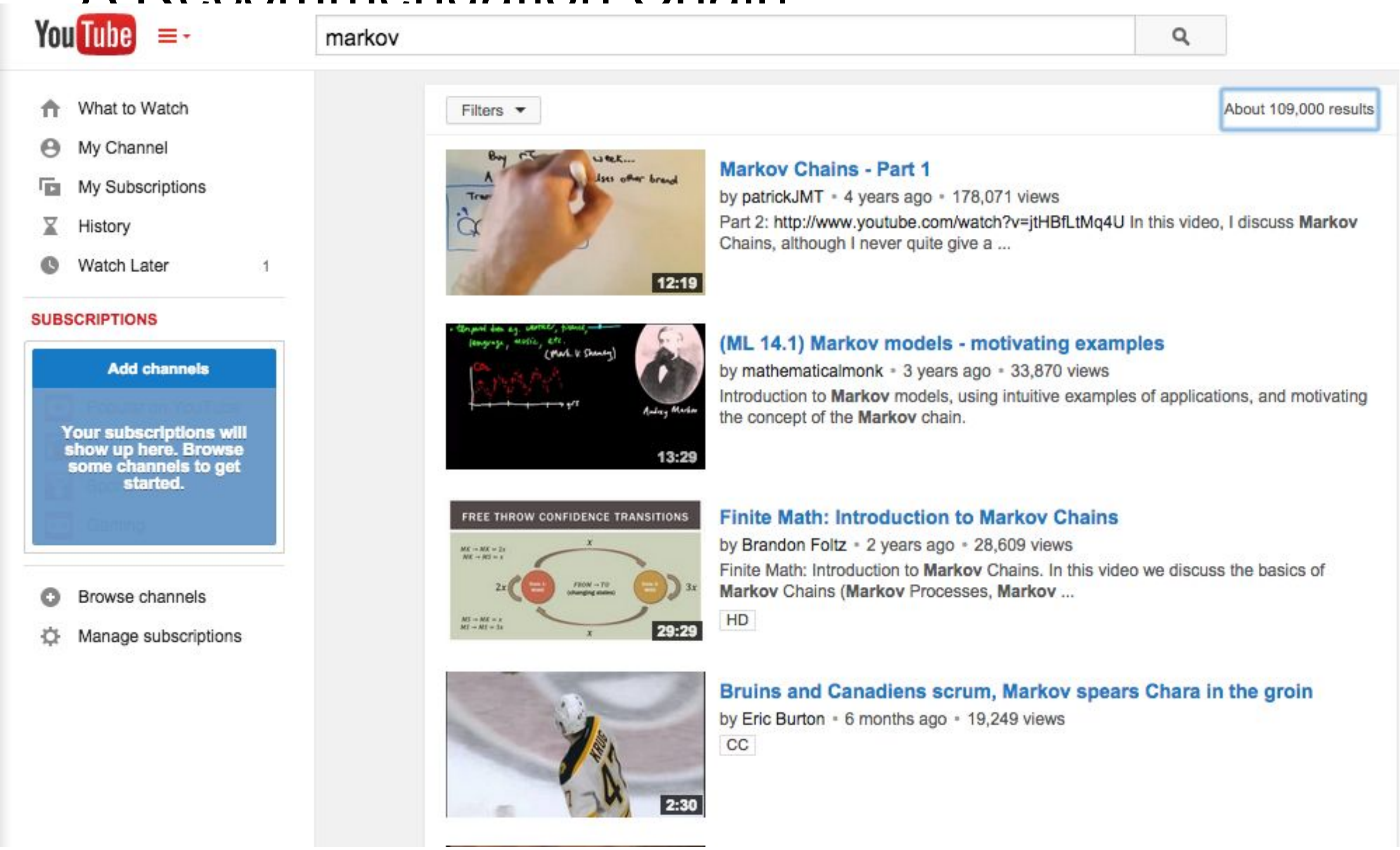
Examples:

- PageRank: Random surfer stationary distribution
- Translation: Use language models to build phrases
- ...

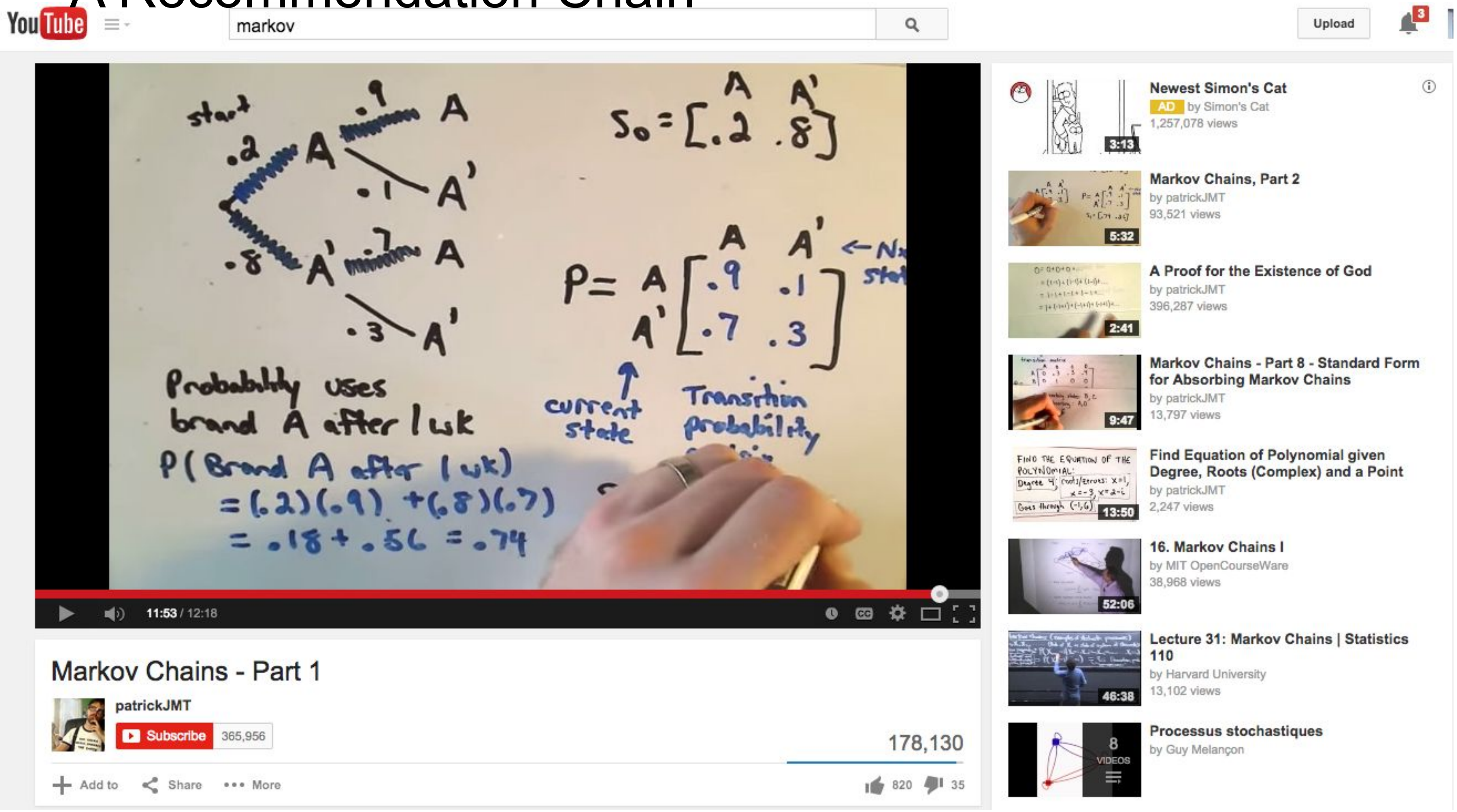
A Recommendation Chain



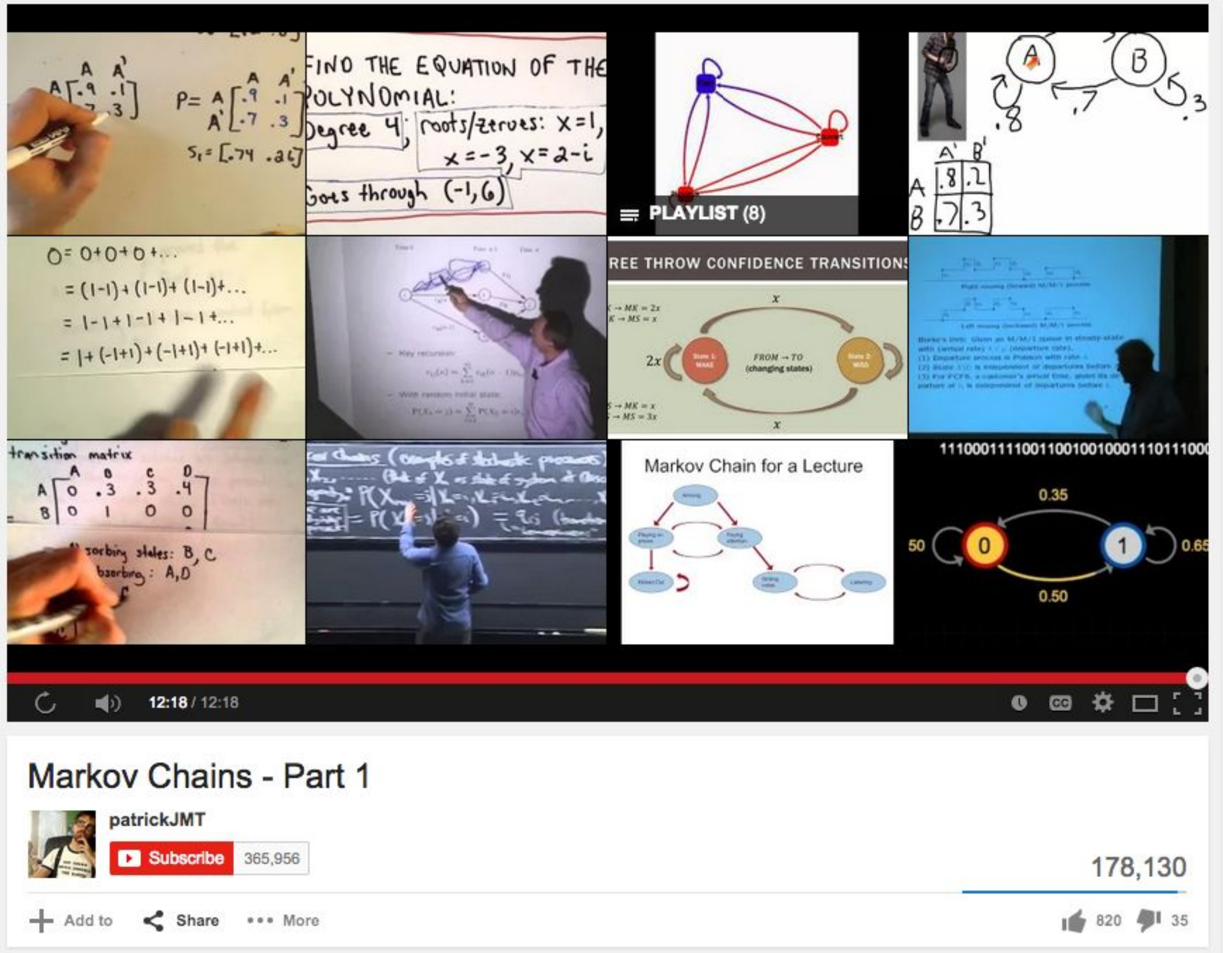
A Recommendation Chain



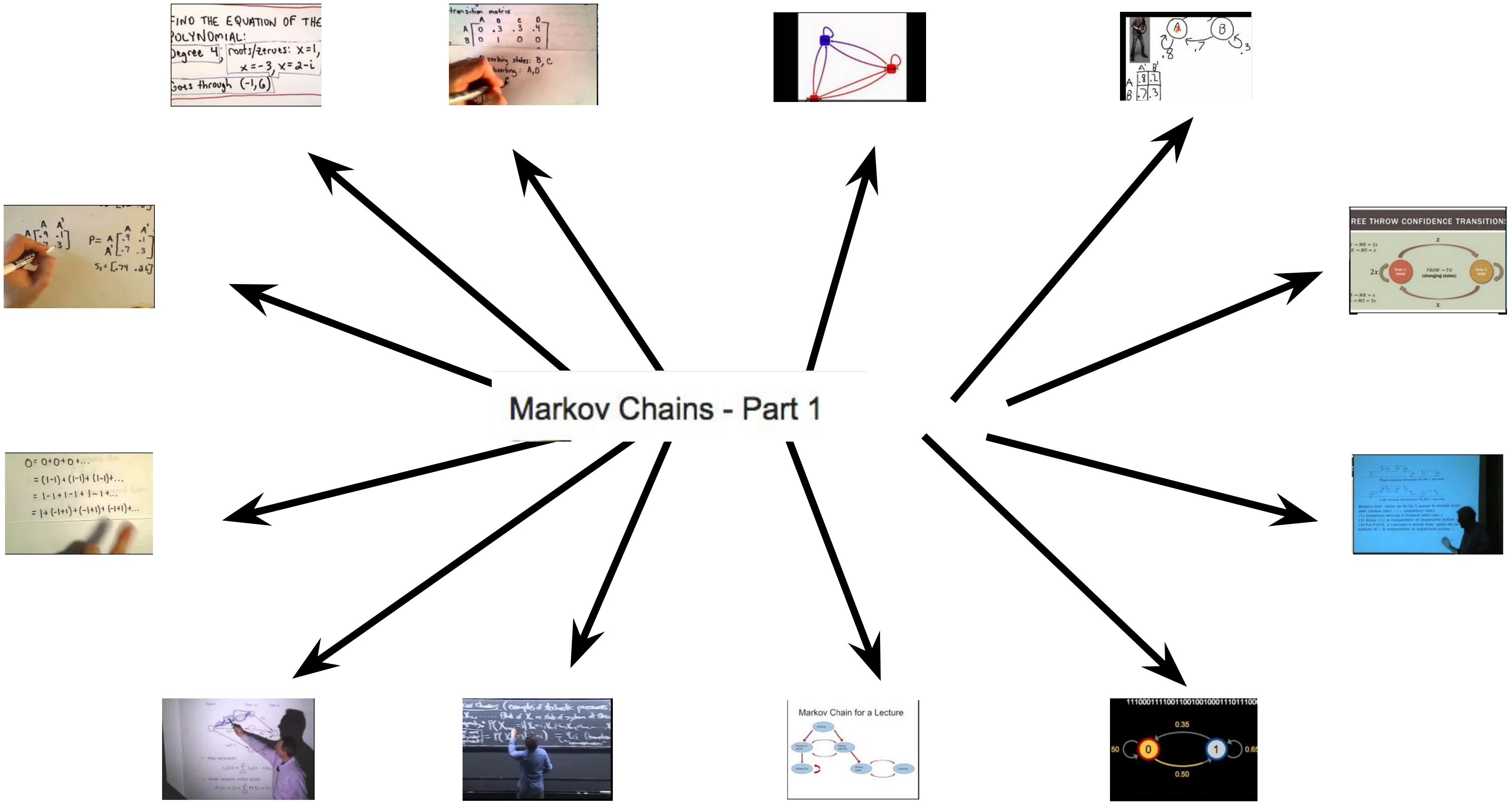
A Recommendation Chain



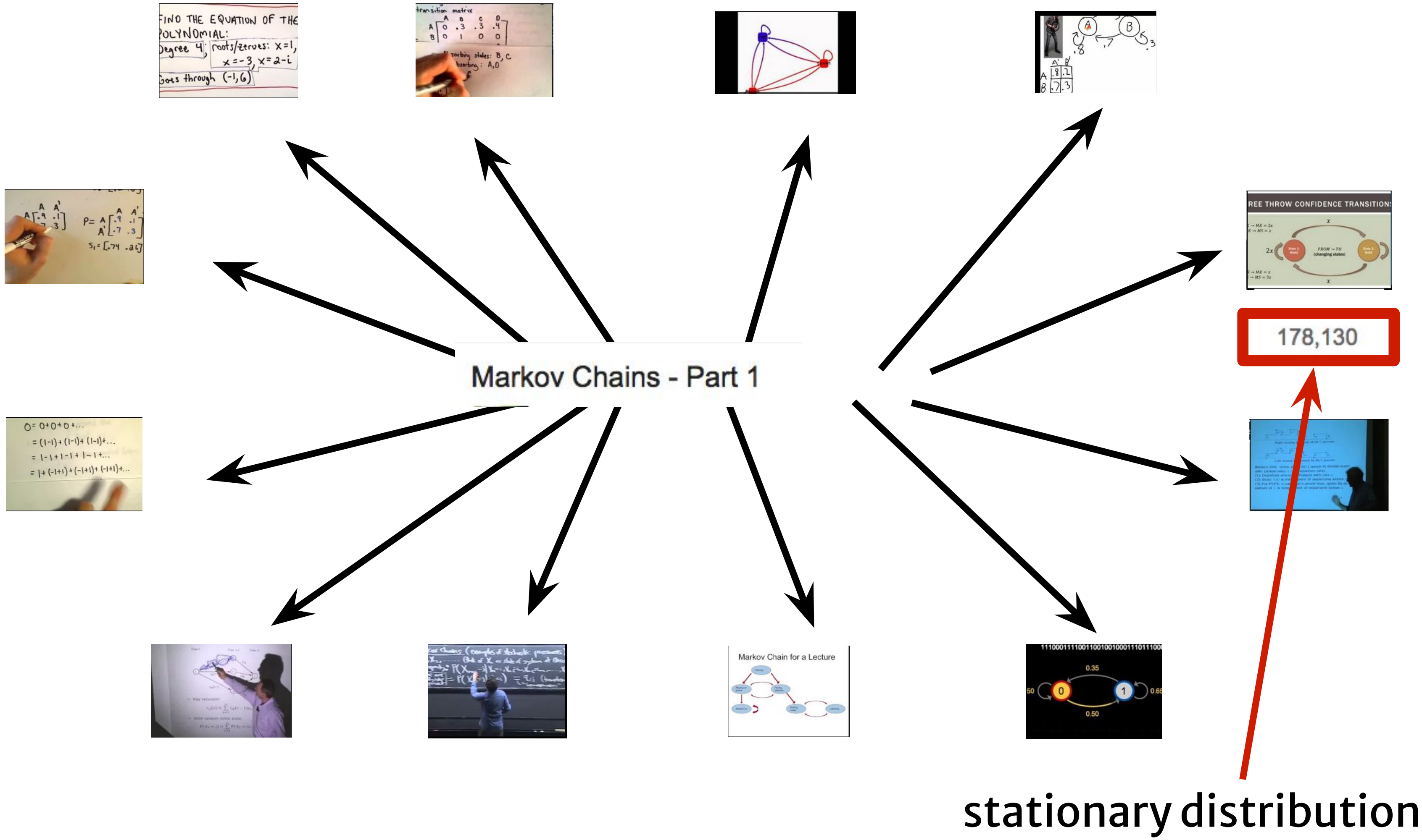
A Recommendation Chain



A Recommendation Chain



A Recommendation Chain



A Recommendation Chain

Example:

- Items: videos
- Stationary Distribution: view counts

Why are some videos more popular:

- Better (higher quality) videos
- More frequently recommended

Today:

- Disentangle these two reasons

Inverting a Markov Chain

Problem:

- Given a stationary distribution, find the Markov Chain that generated it.

Given:

- Graph G
- Distribution π

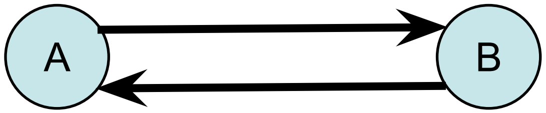
Output:

- Transition Matrix M that generated it

Feasibility

Feasibility:

- Not always feasible



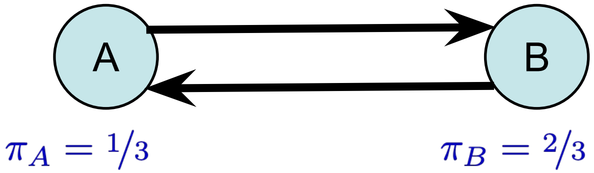
$\pi_A = {}^1\!/_3$ $\pi_B = {}^2\!/_3$

π

Feasibility

Feasibility:

- Not always feasible



Definition:

- A directed graph is consistent if there is a flow that preserves the steady state.
- Any strongly connected graph with self loops is consistent

Theorem:

- For any consistent graph, there exists a Markov chain with π as its stationary distribution.

Constraints

The problem is under-constrained:

- n constraints
- $m - n \gg n$ variables

Constraints

The problem is under-constrained:

- n constraints
- $m - n \gg n$ Variables

Approaches

- [Tomlin `03]: MaxEnt objective on variables (regularization)

Constraints

The problem is under-constrained:

- n constraints
- $m - n \gg n$ Variables

Approaches

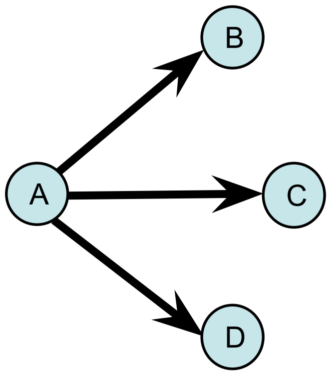
- [Tomlin `03]: MaxEnt objective on variables (regularization)
- [Today] Limit the degrees of freedom

- For each vertex v_i let s_i be its score. The Markov Chain is the function of the scores
- Scores express “quality” or “attractiveness”

From Scores to Transitions

Transition probability $M_{A \to C}$ depends on:

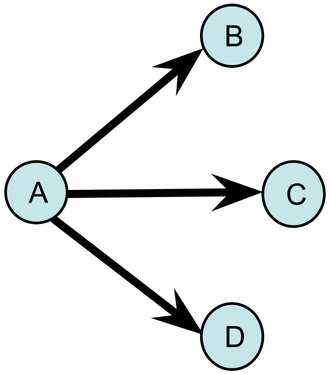
- Score of the destination s_c
- Parameter of the edge w_{AC}



Simplest Example

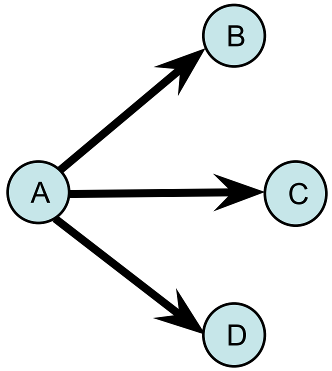
Weighted Random Walk:

- All of the edge weights are set to 1
- Transition probability proportional to the score



$$M_{A \to C} = \frac{s_C}{s_B + s_C + s_D}$$

Simplest Example

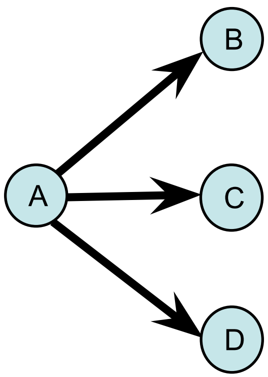


Weighted Random Walk:

- All of the edge weights are set to 1
- Transition probability proportional to the score

$$M_{A \to C} = \frac{s_C}{s_B + s_C + s_D}$$

- Transition probabilities are context dependent:



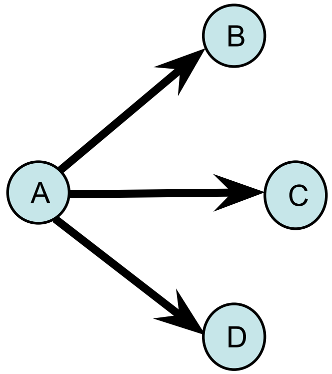
$s_B = 100$

$s_C = 10$

$s_D = 1$

$$M_{A \to C} = 0.09$$

Simplest Example

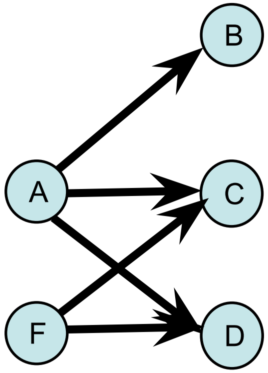


Weighted Random Walk:

- All of the edge weights are set to 1
- Transition probability proportional to the score

$$M_{A \to C} = \frac{s_C}{s_B + s_C + s_D}$$

- Transition probabilities are context dependent:



$s_B = 100$

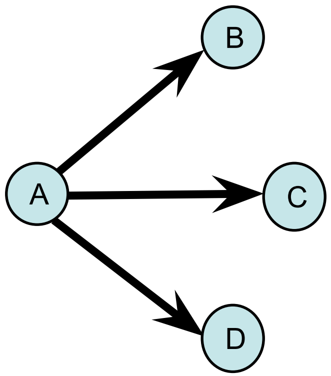
$s_C = 10$

$s_D = 1$

$M_{A \to C} = 0.09$

$M_{F \to C} = 0.91$

From Scores to Transitions



Transition probability $M_{A\to C}$ depends on:

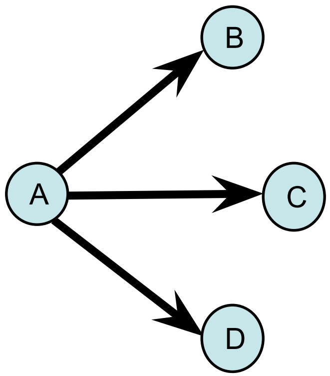
- Score of the destination s_c
- Parameter of the edge w_{AC}
- Call this function f

Formally:

$$M_{A\to C} \propto f(s_C, w_{AC})$$

$$M_{A\to C} = \frac{f(s_C, w_{AC})}{f(s_C, w_{AC}) + f(s_B, w_{AB}) + f(s_D, w_{AD})}$$

From Scores to Transitions



Transition probability $M_{A \to C}$ depends on:

- Score of the destination s_c
- Parameter of the edge w_{AC}
- Call this function f

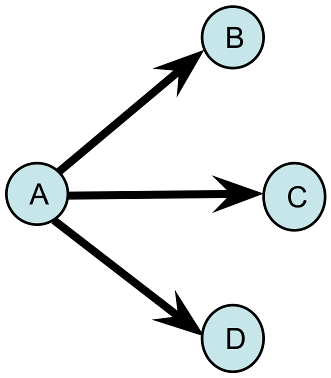
Formally:

$$M_{A \to C} \propto f(s_C, w_{AC})$$

Sanity Check on : f

- Continuous in s
- Monotone in s

From Scores to Transitions



Transition probability $M_{A \to C}$ depends on:

- Score of the destination s_c
- Parameter of the edge w_{AC}
- Call this function f

Formally:

$$M_{A \to C} \propto f(s_C, w_{AC})$$

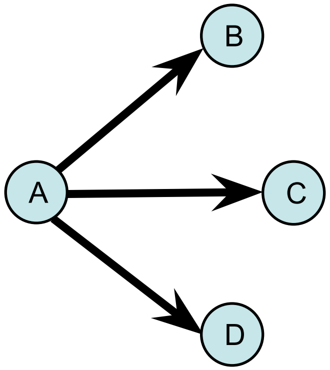
Sanity Check on : f

- Continuous in s
- Monotone in s
- Unbounded in s

$$\lim_{s \to \infty} f(s, w) \to \infty$$

$$\lim_{s_c \to \infty} M_{A \to C} = 1$$

Simplest Example

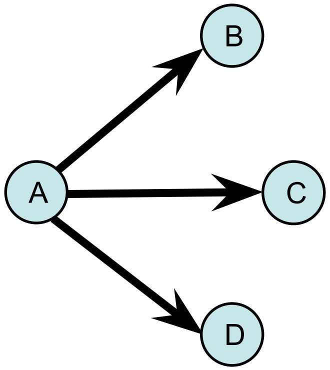


Weighted Random Walk:

- All of the edge weights are set to 1
- Transition probability proportional to the score

$$M_{A \to C} = \frac{s_C}{s_B + s_C + s_D}$$

More Examples



Weighted Random Walk:

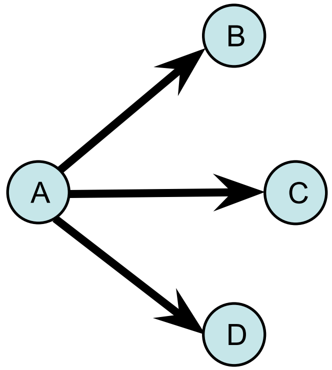
- All of the edge weights are set to 1
- Transition probability proportional to the score

$$M_{A \to C} = \frac{s_C}{s_B + s_C + s_D}$$

Seeking Similar Content:

- Edge weight: similarity between two nodes $M_{A \to C} \propto w_{AC} \cdot s_C$

More Examples



Weighted Random Walk:

- All of the edge weights are set to 1
- Transition probability proportional to the score

$$M_{A \to C} = \frac{s_C}{s_B + s_C + s_D}$$

Seeking Similar Content:

- Edge weight: similarity between two nodes $M_{A \to C} \propto w_{AC} \cdot s_C$

Overall:

- Decide whether items are popular due to high scores (attract all of the incoming traffic) or due to location (attract a little bit from many locations)

Main Theorem

Given:

- A consistent input G, π
- Monotone, continuous and unbounded function f

There exists:

- A unique set of scores $s_1, \ldots, s_n$
- So that π is the stationary distribution induced by f
- Moreover, the scores can be found in polynomial time

Main Theorem

Given:

- A consistent input G, π
- Monotone, continuous and unbounded function f

There exists:

- A unique set of scores $s_1, \ldots, s_n$
- So that π is the stationary distribution induced by f
- Moreover, the scores can be found in polynomial time

up to scaling

Main Theorem

Given:

- A consistent input G, π
- Monotone, continuous and unbounded function f

There exists:

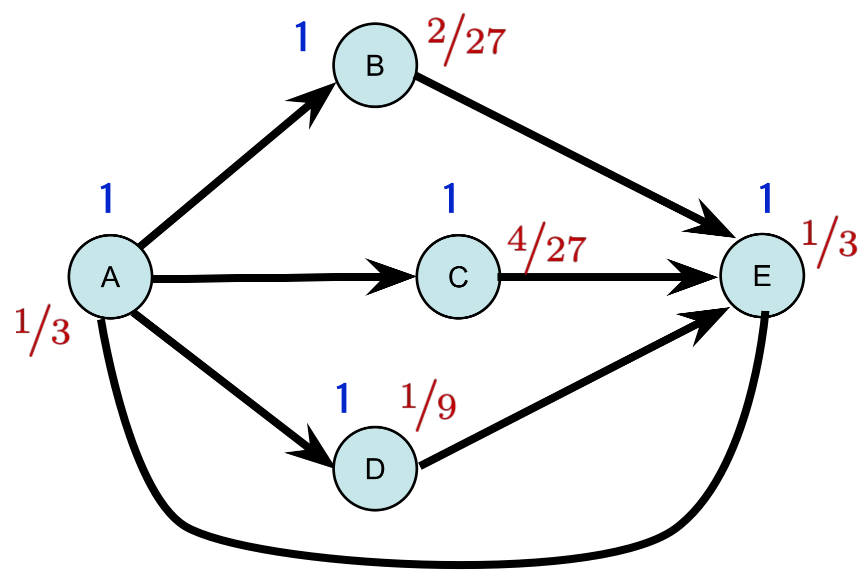
- A unique set of scores $s_1, \ldots, s_n$
- So that π is the stationary distribution induced by f
- Moreover, the scores can be found in polynomial time

up to scaling

up to $(1 \pm \epsilon)$

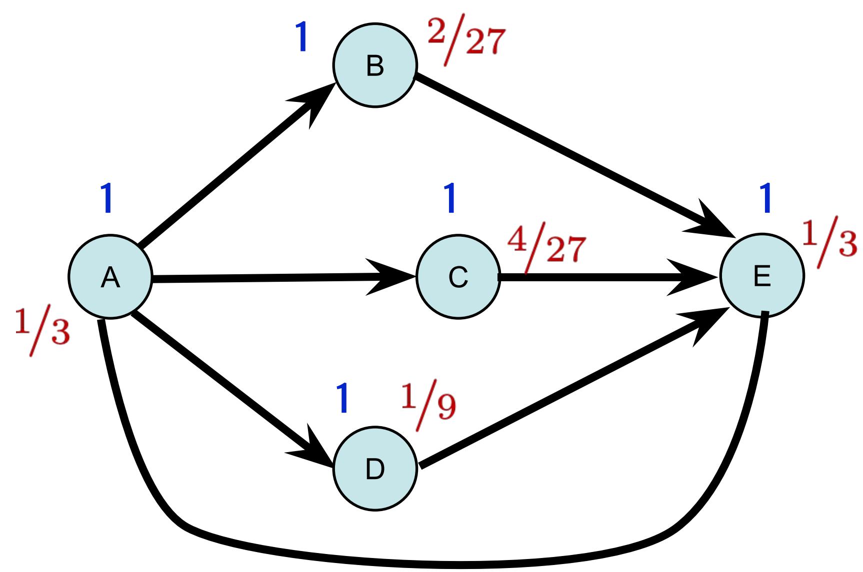
Definitions

- Fix a set of scores s and distribution π



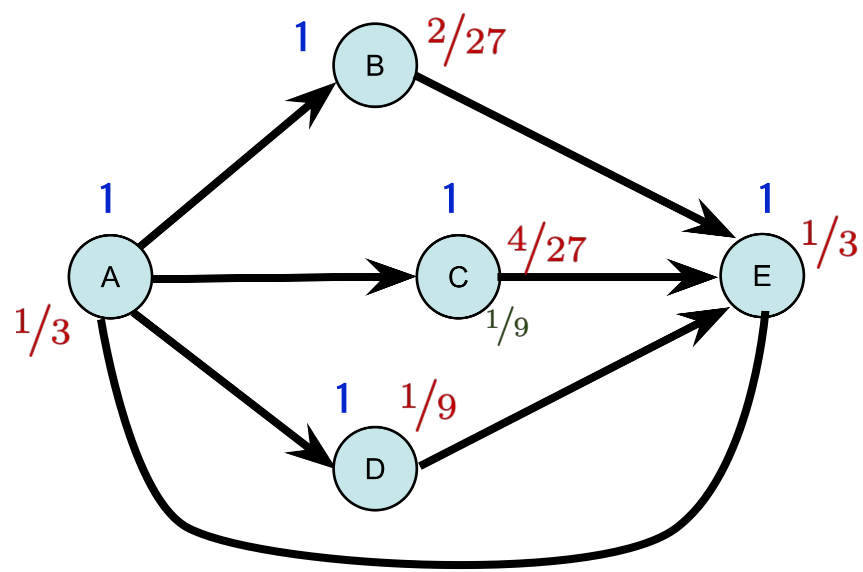
Definitions

- Fix a set of scores s and distribution π
- Let $q_i(s)$ be the expected mass at v_i starting with s using π



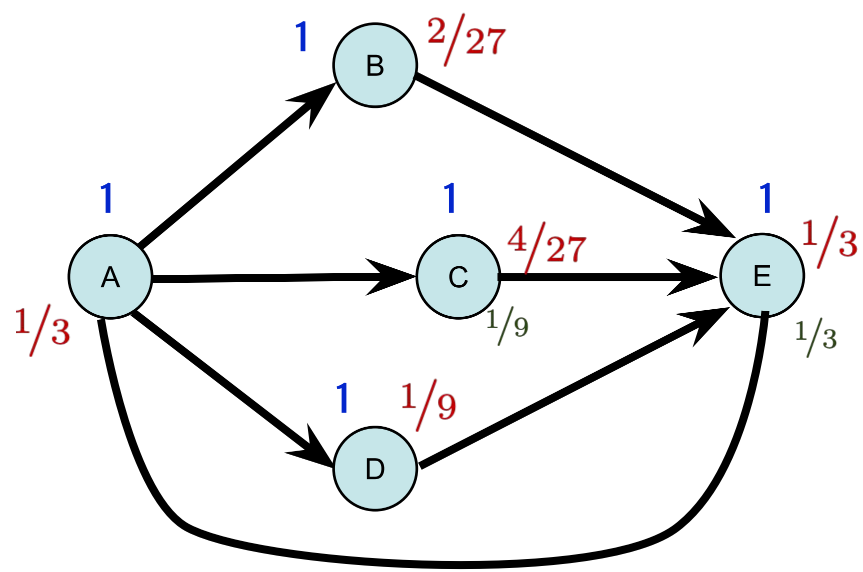
Definitions

- Fix a set of scores s and distribution π
- Let $q_i(s)$ be the expected mass at v_i starting with s using π



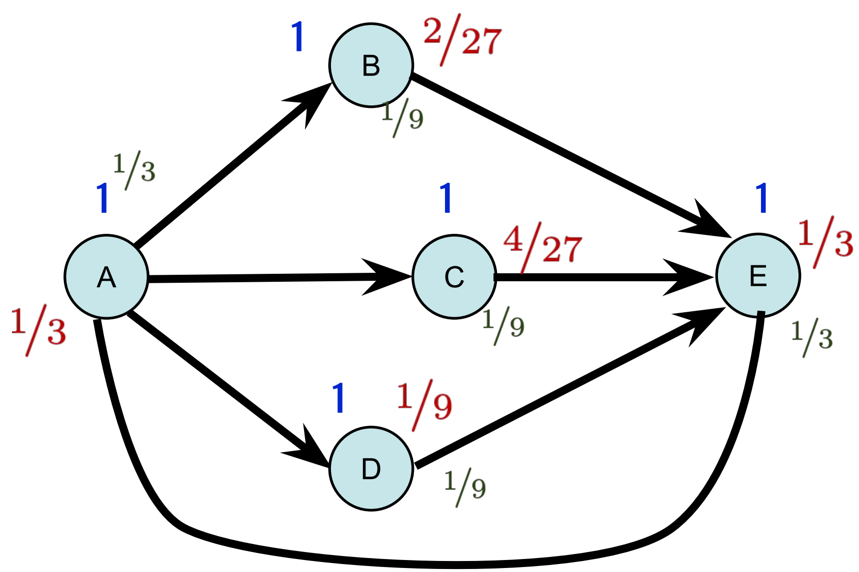
Definitions

- Fix a set of scores s and distribution π
- Let $q_i(s)$ be the expected mass at v_i starting with s using π



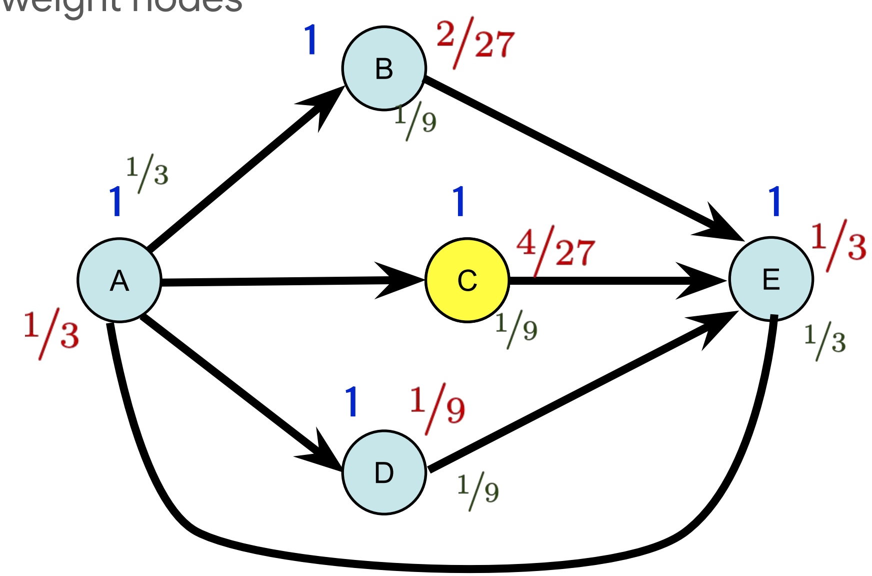
Definitions

- Fix a set of scores s and distribution π
- Let $q_i(s)$ be the expected mass at v_i starting with s using π



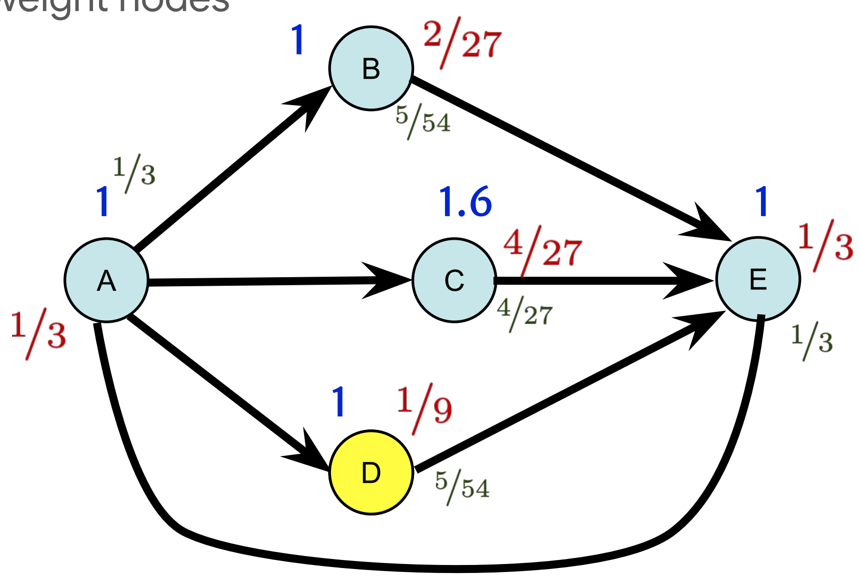
Definitions

- Fix a set of scores s and distribution π
- Let $q_i(s)$ be the expected mass at v_i starting with s using π
- Call a node underweight if $q_i(s) < (1-\epsilon)\pi_i$
- Algorithm:
  - Repeatedly increase scores of underweight nodes



Definitions

- Fix a set of scores s and distribution π
- Let $q_i(s)$ be the expected mass at v_i starting with s using π
- Call a node underweight if $q_i(s) < (1-\epsilon)\pi_i$
- Algorithm:
    - Repeatedly increase scores of underweight nodes



Definitions

- Fix a set of scores s and steady state π
- Let $q_i(s)$ be the expected mass at v_i starting with π using s
- Call a node underweight if $q_i(s) < (1-\epsilon)\pi_i$

Algorithm:

- Start with $s_i^0 = 1/n$
- For $t = 1, \ldots$
  - For each $v_i \in V$:
  - If v_i underweight:

    Set $s_i^t : q_i(s_{-i}^{t-1}, s_i^t) = (1-\epsilon/2)\pi_i$
  - else:

    Set $s_i^t = s_i^{t-1}$

Definitions

- Fix a set of scores s and steady state π
- Let $q_i(s)$ be the expected mass at v_i starting with π using s
- Call a node underweight if $q_i(s) < (1-\epsilon)\pi_i$

Algorithm:

- Start with $s_i^0 = 1/n$
- For $t = 1, \ldots$
  - For each $v_i \in V$:
  - If v_i underweight:
    Set $s_i^t : q_i(s_{-i}^{t-1}, s_i^t) = (1-\epsilon/2)\pi_i$
  - else:
    Set $s_i^t = s_i^{t-1}$

Guaranteed to exist because f is monotone, continuous, unbounded & G is consistent

Definitions

- Fix a set of scores s and steady state π
- Let $q_i(s)$ be the expected mass at v_i starting with π using s
- Call a node underweight if $q_i(s) < (1-\epsilon)\pi_i$

Note: scores never decrease

Algorithm:

- Start with $s_i^0 = {}^1/_n$
- For $t = 1, \ldots$
  - For each $v_i \in V$:
  - If v_i underweight:
    Set $s_i^t : q_i(s_{-i}^{t-1}, s_i^t) = (1-\epsilon/2)\pi_i$
  - else:
    Set $s_i^t = s_i^{t-1}$

Guaranteed to exist because f is monotone, continuous, unbounded & G is consistent

Definitions

- Fix a set of scores s and steady state π
- Let $q_i(s)$ be the expected mass at v_i starting with π using s
- Call a node underweight if $q_i(s) < (1-\epsilon)\pi_i$

Algorithm:

- Start with $s_i^0 = 1/n$
- For $t = 1, \ldots$
  - For each $v_i \in V$:
  - If v_i underweight:
    Set $s_i^t : q_i(s_{-i}^{t-1}, s_i^t) = (1-\epsilon/2)\pi_i$
  - else:
    Set $s_i^t = s_i^{t-1}$

Note: scores never decrease

If q is ever below π, it will always stay below

Guaranteed to exist because f is monotone, continuous, unbounded & G is consistent

Proof of Convergence

Key Lemma:

- There is an explicit bound M such that $s_i^t \leq M$ for all i, t.

Proof of Convergence

Key Lemma:

- There is an explicit bound M such that $s_i^t \leq M$ for all i, t.

Proof Sketch:

- Consider a set of scores that grows without bound
- These scores all must be underweight (these are the only scores that increase)
- Not all scores can be underweight (sum of underweight scores below 1)
- The scores growing without bound are taking all of the probability mass from those bounded
- By consistency, this demand must be met, a contradiction.

Proof of Convergence

Key Lemma:

- There is an explicit bound M such that $s_i^t \leq M$ for all i, t.

Finishing the Proof:

- Scores increase multiplicatively by factor of $(1 + \epsilon/2)$
- M is bounded by $\left(\frac{n^2 W}{\epsilon p_{\min}}\right)^n$
- Overall: $O\left(\frac{n^2}{\epsilon} \log \frac{nW}{\epsilon p_{\min}}\right)$ iterations suffice.

But Does it Work...

Experimental Evaluation:

- Dataset: empirical transitions
- Input: Transition graph and the steady state distribution
- Output: Transition probabilities
- Metrics: LogLikelihood or RMSE

Datasets

Wiki:

- Navigation paths through wikipedia.
- About 200k transition pairs, 51k user traces over 4.6k nodes

Rest:

- Results of broad restaurant queries to Google.
- 100k transitions, 65k nodes

Entree:

- Chicago restaurant recommendation system from 90s
- 50k transitions, 27k nodes

Comedy:

- Given a pair of videos, predict which one is judged funnier
- 225k transitions, 75k nodes

Baselines

Popularity:

- Transition proportionally to the steady state distribution (score = pi)

Uniform:

- Uniform over out-edges

Pagerank:

- Transition proportionally to the node pagerank

Temperature:

- MaxEnt regularization approach

Inversion:

- Our algorithm

Results

RMSE Prediction:

	Popularity	Uniform	PageRank	Temp	Inversion
Wiki	1	0.65	0.83	0.65	**0.57**
Rest	1	1.17	1.39	1.21	**0.59**
Entree	1	0.69	1.01	0.56	**0.42**
Comedy	1	0.65	0.9	0.78	**0.36**

Application: Sequential Choice

Repeat consumption

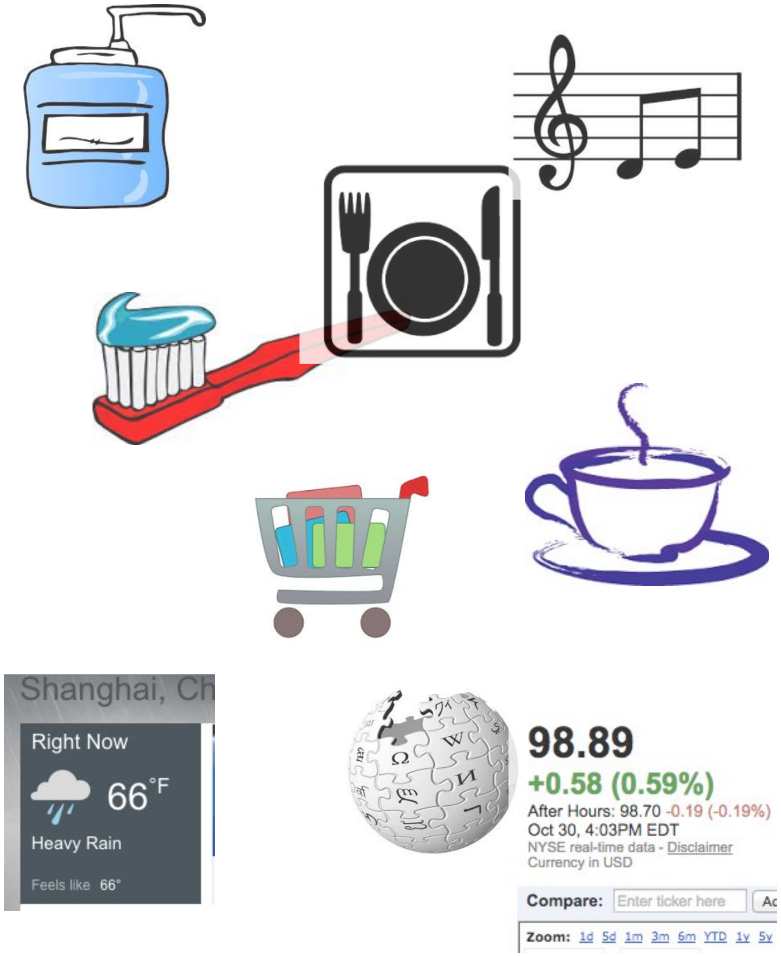
Most of the items we consume are not for the first time

Sometimes go for reliability

Sometimes go for novelty

- Boredom
- New options

We focus on the repeated consumption, not the novel choice.



Repeat consumer choice

Marketing studies

Consumer behavior

Music listening experiment [Kahnx et al 97]

- Melioration/overconsumption: listen to favorite on each trial
- Maximization: preserve the high level of enjoyment

Possible explanations

- Difficulties in prediction of taste
- Users try to create the best memory (five flavors vs one flavor LifeSavers)
- Zen principles (pain vs pleasure)

Re-searching

Repeat queries in search logs [Teevan et al]

40% of queries are re-finding queries

Navigational queries are more likely to be repeated

- Information re-finding

Repeat behavior leads to easier prediction of which results will be clicked

Re-visiting web pages

Web page revisitation using browser logs [Adar et al]

50-80% of the web pages are revisited

Revisitation reasons

- Bookmarks/use as hub
- Track content change
- Backbutton

Types of revisitation

- Fast: shopping pages, references, traffic
- Medium: mail, forums, news, ...
- Slow: weekend activity, software updates, ...

Domains of reconsumption

Location checkins

- BrightKite
- Google+

Clicks

- Businesses on maps
- Restaurants on maps
- Wikipedia

Media

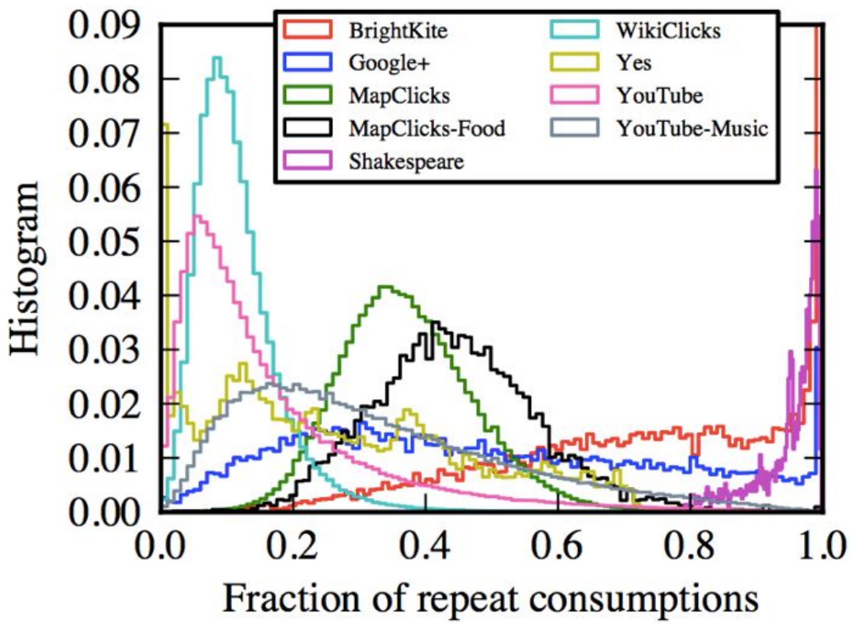
- Youtube
- Music videos
- Playlists from a radio station

Shakespeare!

Characterizing Reconsumption

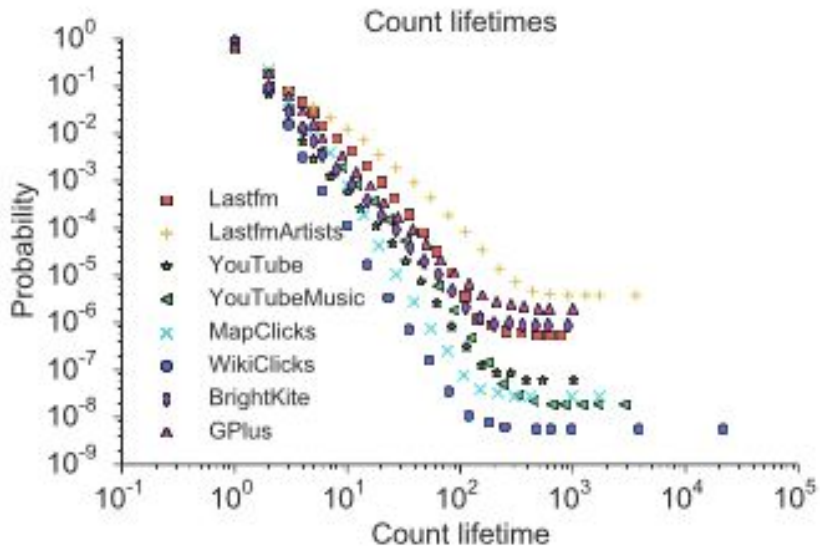
Does it exist?

Distribution of the fraction of repeat consumption



Lifetime distributions

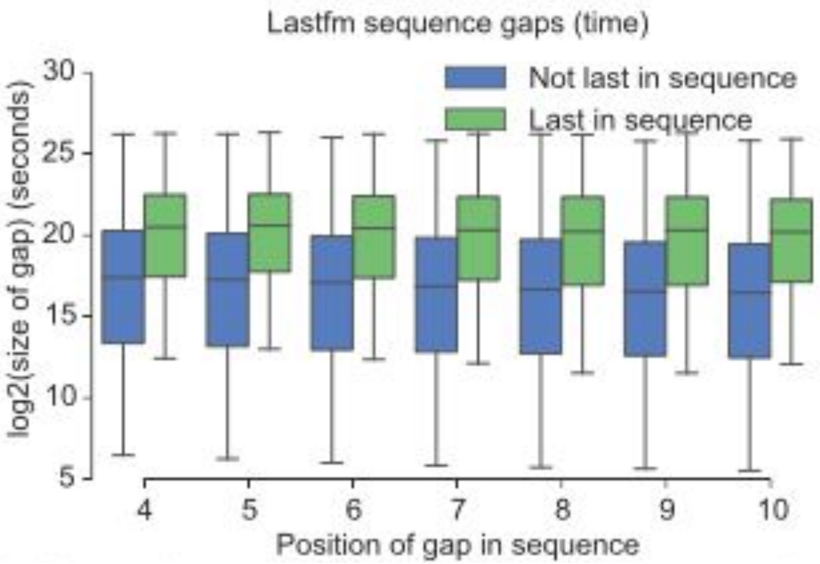
Do items have finite lifetimes?

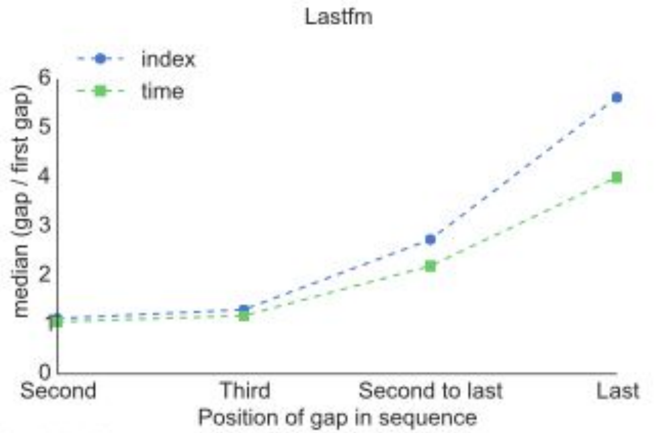


Boredom

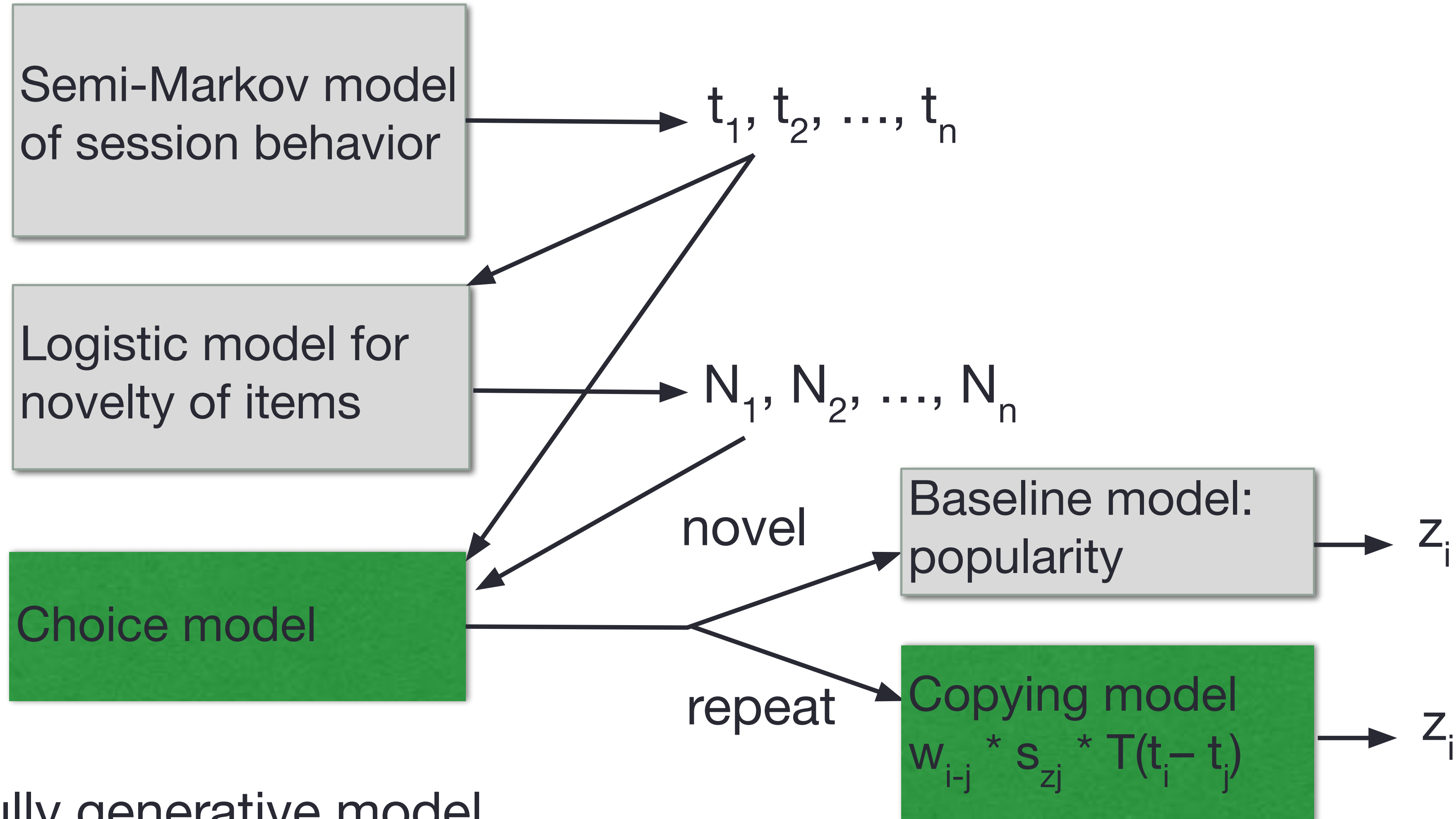
Do users get bored with repeat consumption?

- Marketers, advertisers care about this
- Churn/variety-seeking behavior





Summary of model



Fully generative model.

Also matches macroscopic properties (up next!)

Three key factors

- How popular is the item?
- Time gap since it was last consumed
- How recently was it consumed?

- Can we develop a holistic mathematical framework powerful yet simple enough to explain patterns of reconsumption we observe in real data?

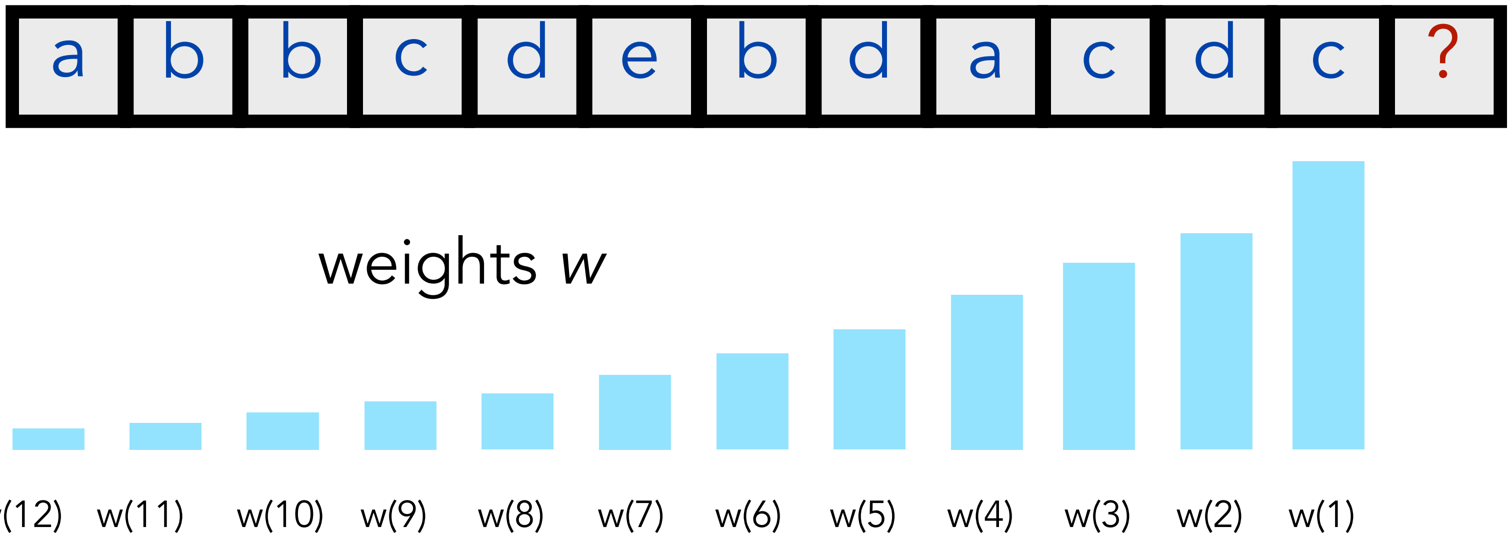
Recency model

Empirically, recency seems to play a strong role in reconsumption

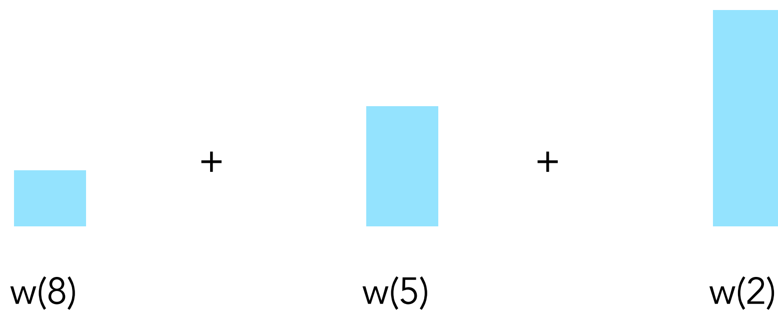
Technical approach: Combine discrete choice model with “copying model” [Simon, 55] based on recency

Example

consumption history



Pr[d is consumed next] ~



Score-based model

Each item *x* has a score s_x

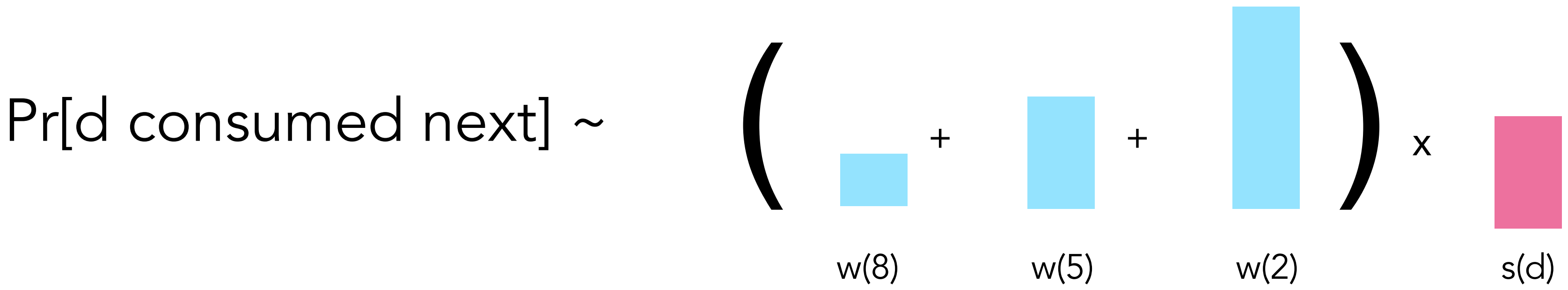
The score reflects the quality of the item

The score dictates the reconsumption pattern

Pick next item *x* using discrete choice, with probability:

$$\Pr[x|X] = \frac{s_x}{\sum_{y \in A} s_y}$$

Combining Recency and Quality



At position i, pick item *x* with probability:

$$\frac{\sum_{j<i} I(x_j = x) w_{i-j} s_{x_j}}{\sum_{j<i} w_{i-j} s_{x_j}}$$

Stochastic gradient ascent

Alternating updates to scores and weights

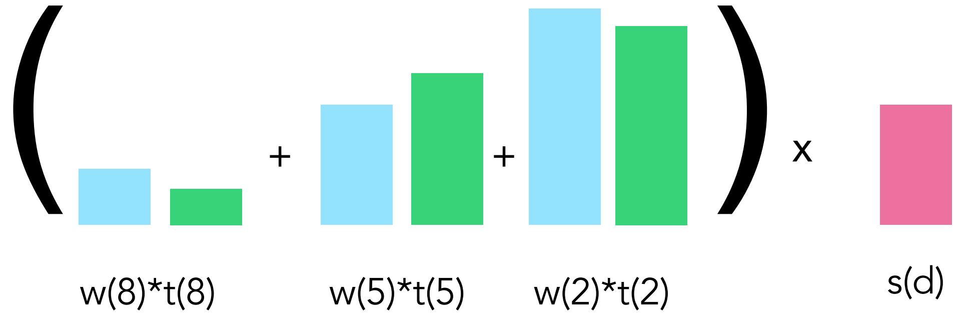
Likelihoods (wrt hybrid model)

$s(\cdot) =$ $w(\cdot) =$	popularity -	popularity learned	learned uniform	uniform learned
BRIGHTKITE	0.375	0.617	0.637	0.936
GPLUS	0.587	0.801	0.794	0.877
MAPCLICKS	0.383	0.931	0.414	0.989
WIKICLICKS	0.503	0.724	0.687	0.945
YOUTUBE	0.636	0.677	0.924	0.962

- Recency comes close to hybrid model
- Recency much better than quality
- Popularity seems to bring the models down even with recency

Combining Recency, Quality, and Time

Pr[d consumed next] ~



At position i, pick item *x* with probability:

$$\frac{\sum_{j<i} I(x_j = x) w_{i-j} s_{x_j} t_{t_i - t_j}}{\sum_{j<i} w_{i-j} s_{x_j} t_{t_i - t_j}}$$

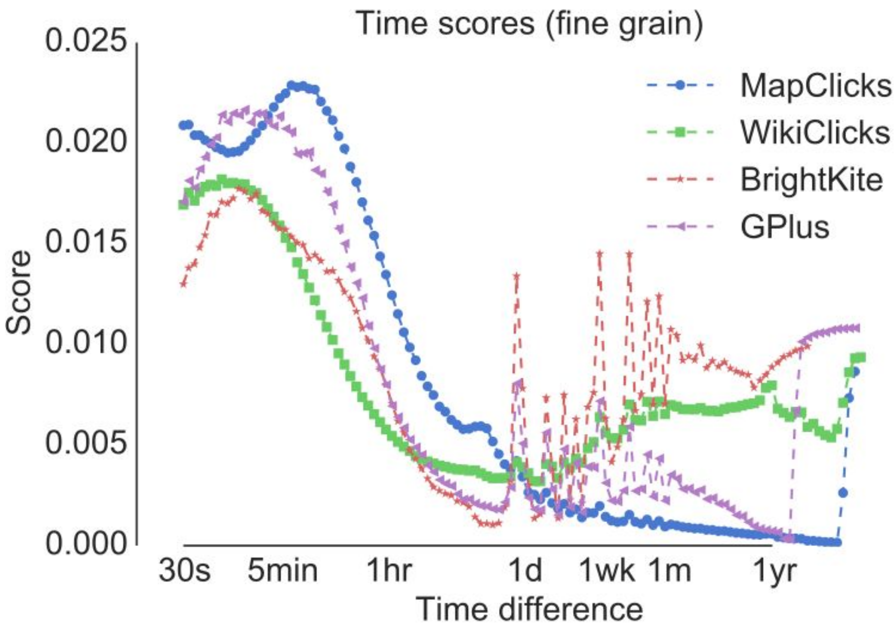
Stochastic gradient ascent

Alternating updates to scores and weights

Learned time scores $T(t_i - t_j)$

- Learned time scores are complex

- Capture, e.g., cyclic behavior in check-in data.



Model Quality

Dataset	Learned scores w	w and s	w and T
BRIGHTKITE	0.91	0.92	0.98
GPLUS	0.87	0.92	0.94
LASTFM	0.99	0.99	1.00
LASTFMARTISTS	0.96	0.96	1.00
YOUTUBE	0.91	0.94	0.96
YOUTUBEMUSIC	0.92	0.93	0.97
MAPCLICKS	0.81	0.82	0.99
WIKICLICKS	0.78	0.81	0.91

- Score-only and Popularity-only not competitive
- Recency is most important feature
- Time is more important than item quality
- All model components bring some gain

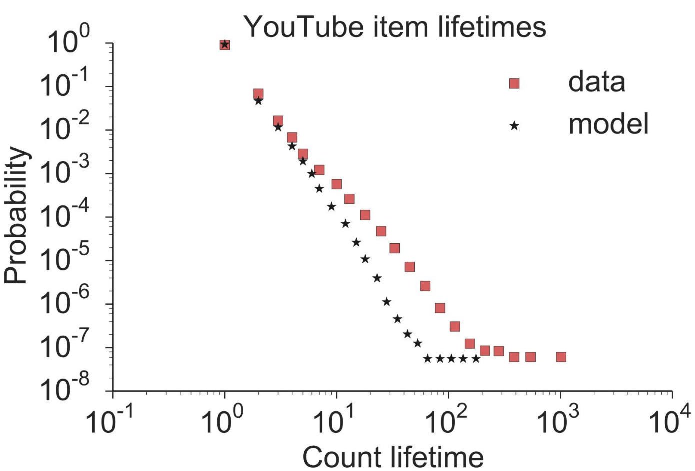
Macroscopic observations

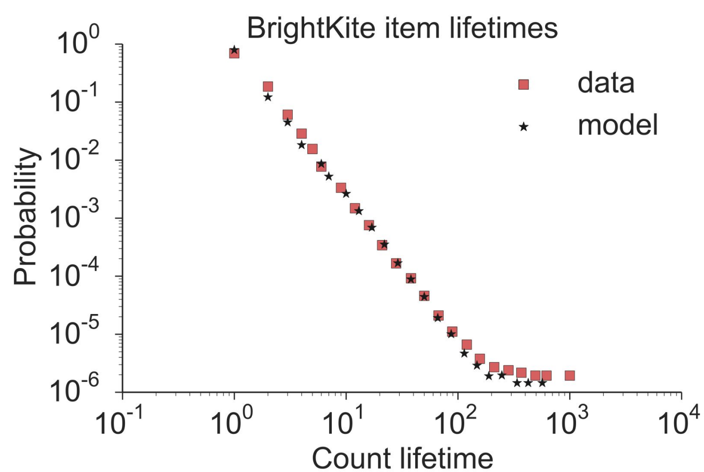
1. **Eventual abandonment**: item lifetime distributions are heavy-tailed and often finite.

2. **Boredom**: at the end of an item's life, gaps between consumptions increase monotonically.

Item lifetimes

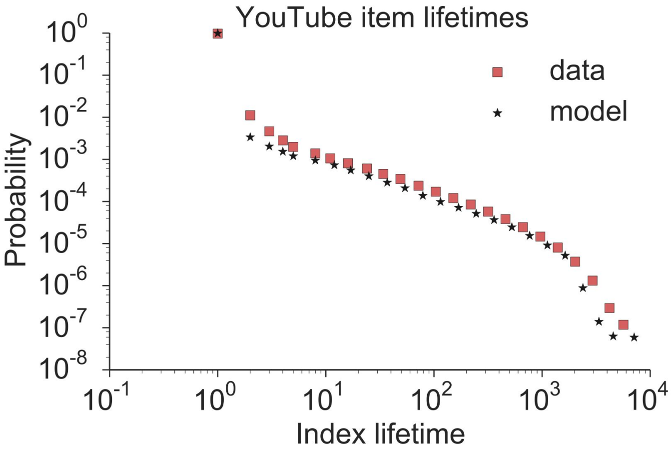
Count lifetime:
number of times an item is consumed.

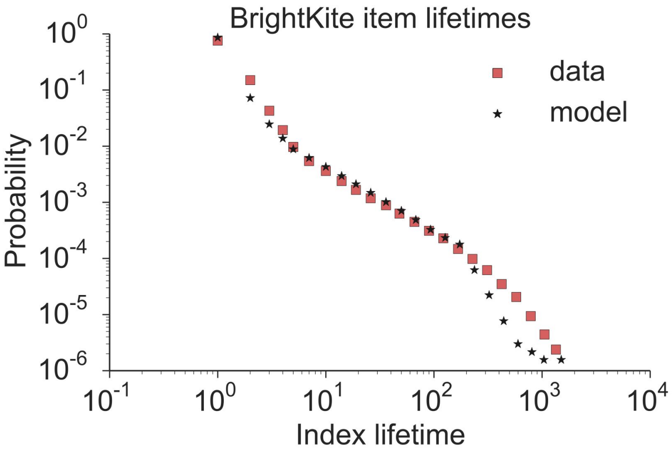




Item lifetimes

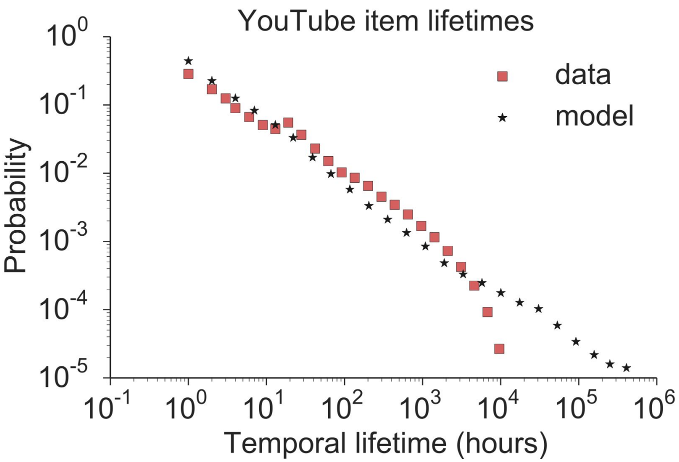
Index lifetime:
total number of items consumed between first and last consumption of a given item.

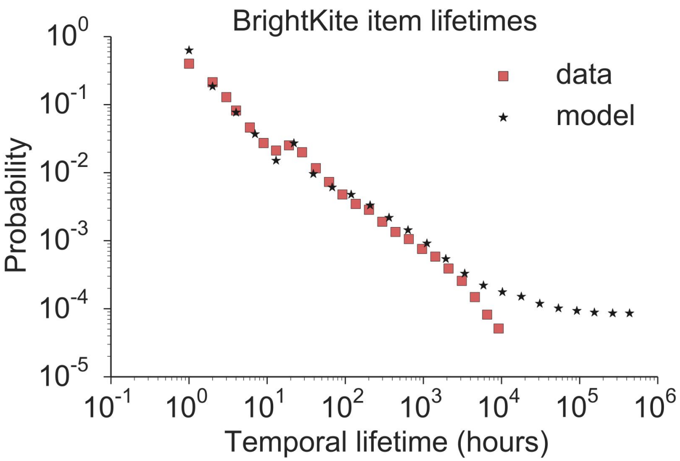




Item lifetimes

Temporal lifetime:
total elapsed time between first and last consumption of an item.





Item Lifetimes Theoretical Analysis

For simple “copying” model with recency only, we can analyze conditions in which an item lives forever:

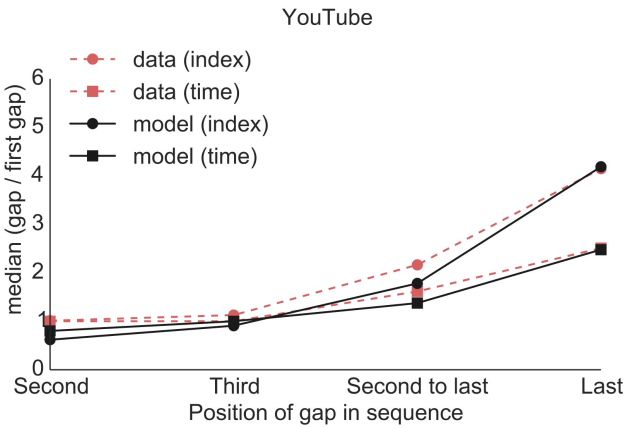
Theorem:

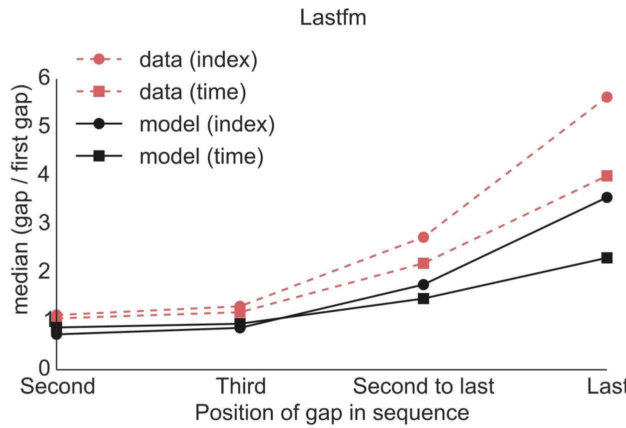
$$\text{Let } \alpha \text{ be probability of novel item}$$

$$\text{If } \sum_{i=1}^{\infty} w_i < 1/\alpha \text{ then } \Pr[\text{lifetime}(x) < \infty] \to 1$$

Boredom

Before items are abandoned, the gap between consumptions of that item grows in both “index” and “real” time.





Boredom

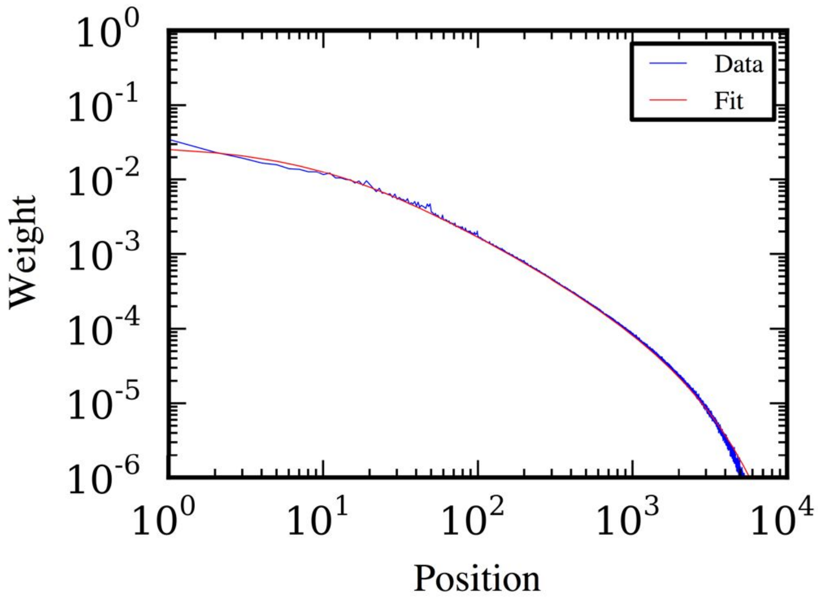
Consider a simplified choice model with uniform time and item quality scores.

Theorem: Suppose that the weights *w* are monotonically decreasing. Then:

1. $E[j^{th}\text{gap}] < E[(j-1)^{st}\text{gap}]$
2. $E[j^{th}\text{gap}|\text{last occurrence}] > E[j^{th}\text{gap}]$
3. $\forall j > J_0 : E[j^{th}\text{gap}|j^{th} \text{ is last}] > E[j-1^{st}\text{gap}|j^{th} \text{ is last}]$

Parsimonious model

- Recency weights can be compressed
- Good fit: power law with exponential cutoff:



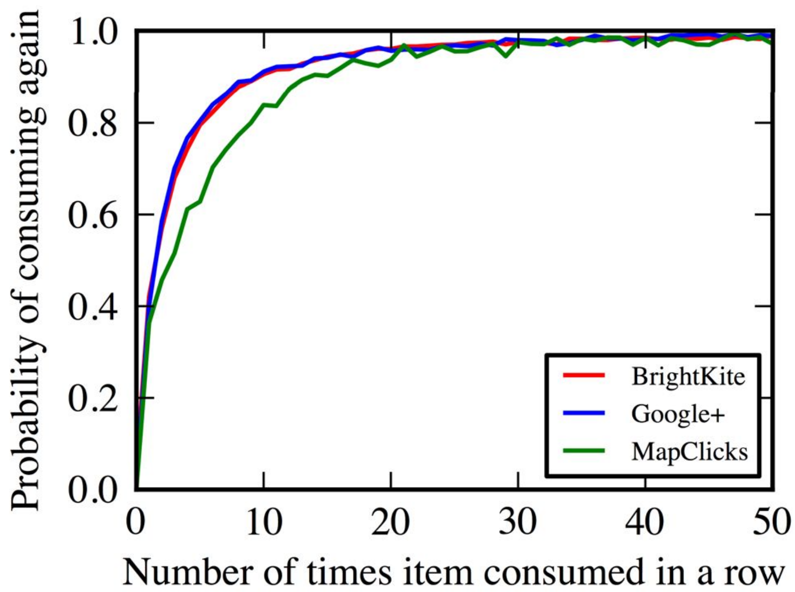
$$\Pr[x] \propto (x+c)^{-a} e^{-bx}$$

Parsimonious model

Dataset	Recency@50	PLECO
BRIGHTKITE	0.654	0.926
GPLUS	0.710	0.987
MAPCLICKS	0.668	0.921
WIKICLICKS	0.971	0.999
YOUTUBE	0.917	0.997

Recency model can be expressed using just three parameters!

Satiation



No evidence of satiation in online user behavior

Chierichetti, Kumar, Tomkins

Additivity assumption

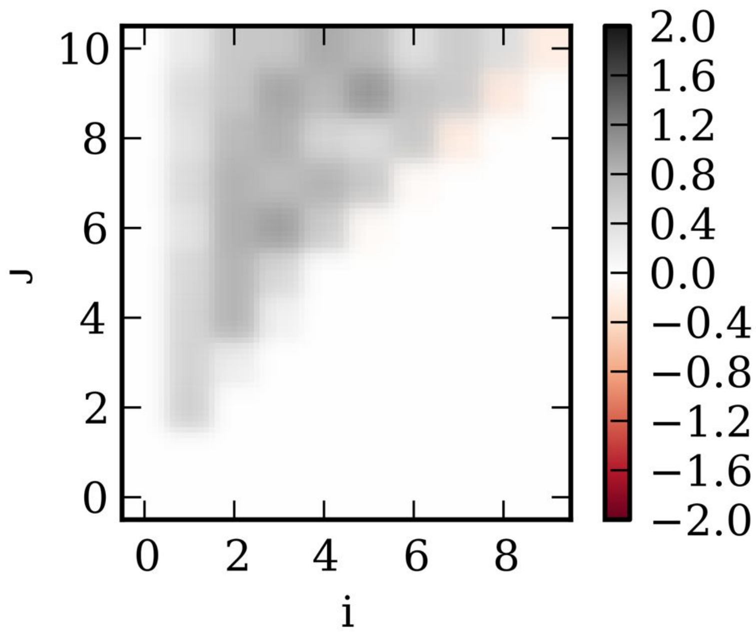
Very small deviations from additive behavior

Mildly superadditive as popular items chosen

$$\frac{w_i + w_j}{w(i,j)}$$

Getting addicted: superadditive?

Getting bored: subadditive?



Tipping behavior

In the recency model, tipping occurs if after a certain time, only one item is repeatedly consumed

Assume weights are decreasing: $w(p) \geqq w(p+1)$

Claim. If sum of weights is finite, then tipping occurs with constant probability

Claim. If the sum of weights is infinite, then tipping does not occur

Conclusions

We studied a number of algorithmic problems related to discrete choice

We believe this class of problems is theoretically important and relevant in practice

Some open questions

Can one reconstruct, with poly(n) `max-sample` queries, the winning probabilities of all slates with o(1) ℓ_1-error?

What is the relative power of the `max-sample` / `max-dist` oracles?

How well can one approximate general mixtures of MNLs with the two oracles?

Identifiability of non-uniform 2-MNLs, k-MNLs