A Tree-Based Method for Fast Repeated Sampling of Determinantal Point Processes

Jennifer Gillenwater[1], Alex Kulesza[1], Zelda Mariet[1,2], Sergei Vassilvitskii[1]

[1]Google Research NYC, [2]Massachusetts Institute of Technology

Motivation

Diverse results are desirable in many applications of **information retrieval** and **recommender systems**

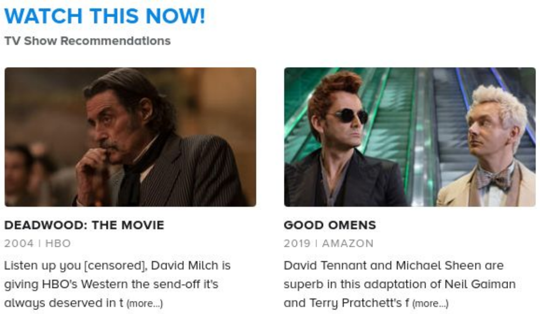
Motivation

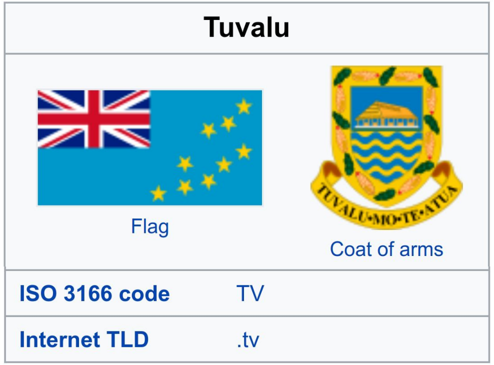
Diverse results combat query ambiguity in search

Example:

TV

TV sets to buy | **TV shows to watch** | **Country with code TV**

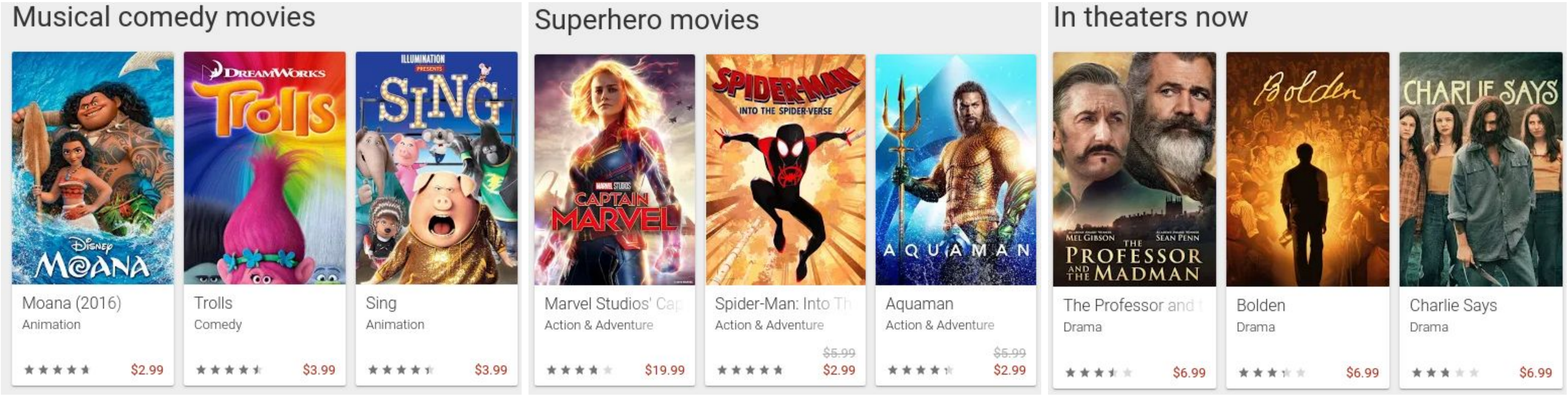




Motivation

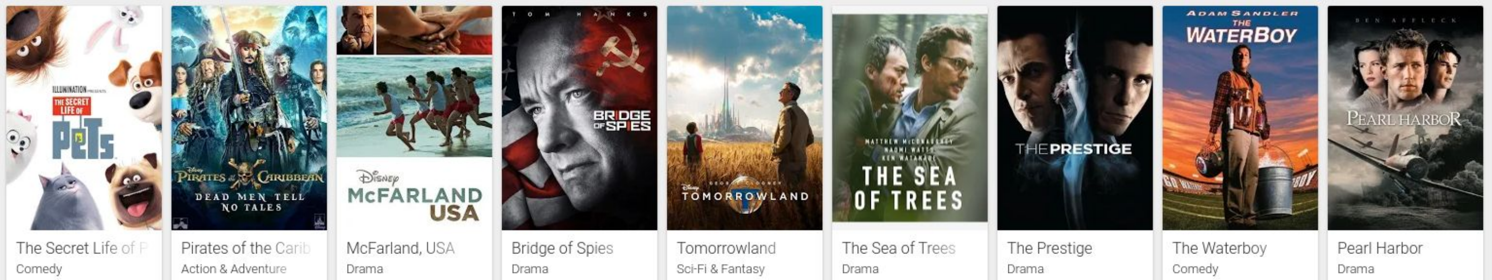
Diverse results increase the chance of engagement with at least one item in recommender systems

Example: movie recommendations



Movie recommendation example

N movies total



...
(millions)

Goal: Select k << N movies to recommend to a user

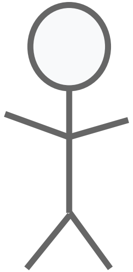
Movie recommendation example

item i

b_i

3.2	*kid vs adult scale*
1.0	*musical?*
...	
1.8	*old vs new scale*

length D



user

w

2.6
0.3
...
1.1

length D

Reweighted movie features

$$\hat{b}_i = w \circ b_i$$

interest match score = $\hat{b}_i^\top \hat{b}_i$

movie similarity score = $\hat{b}_i^\top \hat{b}_j$

Goal

Select k movies that have high interest match but low similarity to each other

Movie recommendation example

Many possible heuristics

E.g., marginal relevance

Probabilistic approach

Determinantal point processes (DPPs)

Diagonal user matrix

$$W \in \mathbb{R}^{D \times D}$$

✖

Movie feature matrix

$$B \in \mathbb{R}^{D \times N}$$

=

Re-weighted movie feature matrix

$$\hat{B} = WB$$

Positive semi-definite kernel

$$\hat{L} = \hat{B}^\top \hat{B}$$

$$\hat{L}_{ij} = \hat{b}_i^\top \hat{b}_j$$

Distribution over all subsets

$$Y \subseteq [N]$$

$$\mathcal{P}_{\hat{L}}(Y) \propto \det(\hat{L}_Y)$$

→ **det**erminantal point process (DPP)

Movie recommendation example

$$\mathcal{P}_{\hat{L}}(Y) \propto \det(\hat{L}_Y)$$

interest in movie j

Intuition for |Y| = 2: $\det(\hat{L}_{\{i,j\}}) = \hat{b}_i^\top \hat{b}_i \; \hat{b}_j^\top \hat{b}_j - (\hat{b}_i^\top \hat{b}_j)^2$

similarity of movies i and j

interest in movie i

Recommendation method: Draw a sample from this distribution

DPP sampling

- **Goal** -- For each user, draw a size-k sample from their DPP
- **Problem** -- Existing algorithms for k-DPP sampling are too expensive
  - Recall: N = # of items (millions), D = # of features
  - D << N by construction or random projection
  - $O(ND^2)$ preprocessing on $L = B^TB$
  - $O(\mathbf{N}k^2 + D^3)$ per personalized (W-weighted) sample afterwards
- **Our contribution** -- A k-DPP sampler where repeated, personalized sampling is more efficient:
  - $O(ND^2)$ preprocessing on $L = B^TB$
  - $O(D^2k^2 \mathbf{\log N} + D^3)$ per personalized (W-weighted) sample afterwards
  - Pay for speed with memory

Standard dual sampling algorithm

- **Pre-processing:** Build dual kernel C = BB$^\top$, O(ND^2)
- **Step 1:** Personalize and eigendecompose, O(D^3)

$$\text{eigendecomposition } \{\hat{\mathbf{v}}_i, \hat{\lambda}_i\}_{i=1}^{D} \text{ of } \hat{C} = WCW$$

- **Step 2:** Select a set E consisting of k of the eigenvectors, O(Dk);
now marginal probabilities of items are defined as follows:

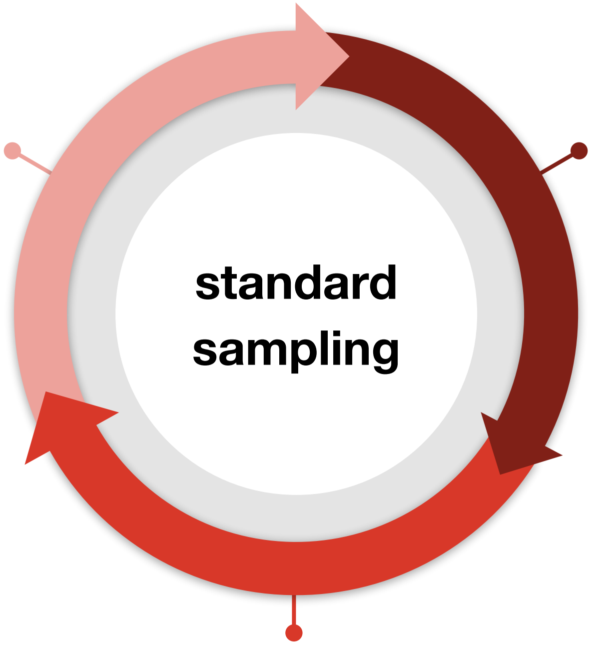
$$\hat{K} = \sum_{i \in E} \frac{1}{\hat{\lambda}_i} (\hat{B}^\top \hat{\mathbf{v}}_i)(\hat{B}^\top \hat{\mathbf{v}}_i)^\top$$

$$P(i \in Y) = \hat{K}_{ii}$$

Standard dual sampling algorithm

condition on selection

$$\hat{K}^Y = \hat{K}_{\bar{Y}} - \hat{K}_{\bar{Y}Y}(\hat{K}_Y)^{-1}\hat{K}_{Y\bar{Y}}$$



O(N) is too expensive

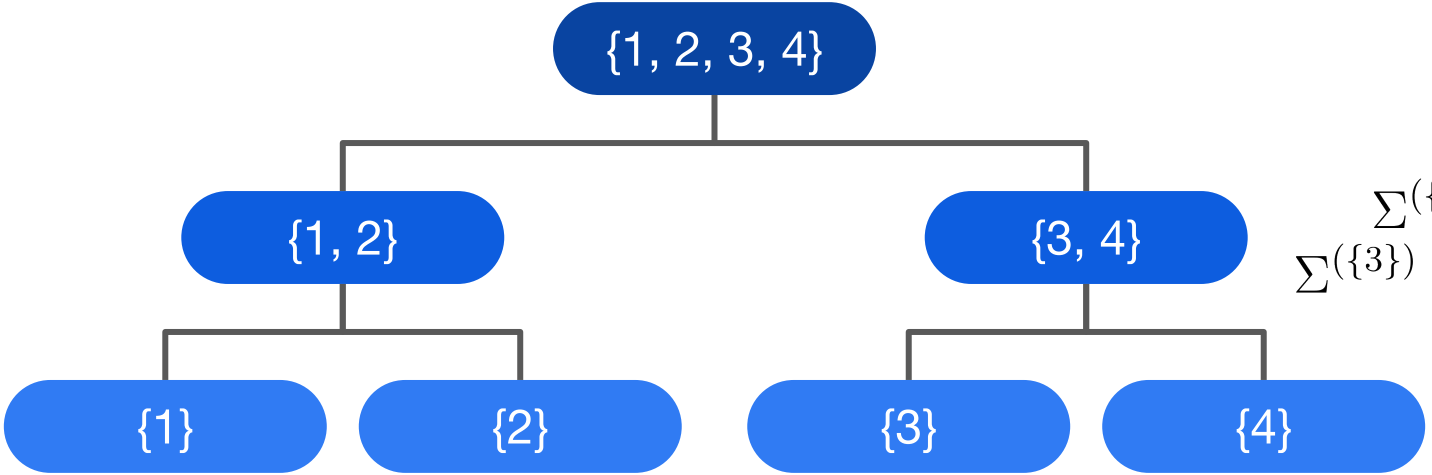
compute N marginals

$$P(i \in Y) = \hat{K}_{ii}$$

select an item $i \sim \frac{\hat{K}_{ii}}{\sum_{j=1}^{N} \hat{K}_{jj}}$

Our tree-based algorithm

Key idea: In pre-processing, create a balanced binary tree of depth log N.



$$\Sigma^{(\{3,4\})} = \Sigma^{(\{3\})} + \Sigma^{(\{4\})}$$

$$\Sigma^{(\{3\})} = \mathbf{b}_3 \mathbf{b}_3^\top$$

D x D

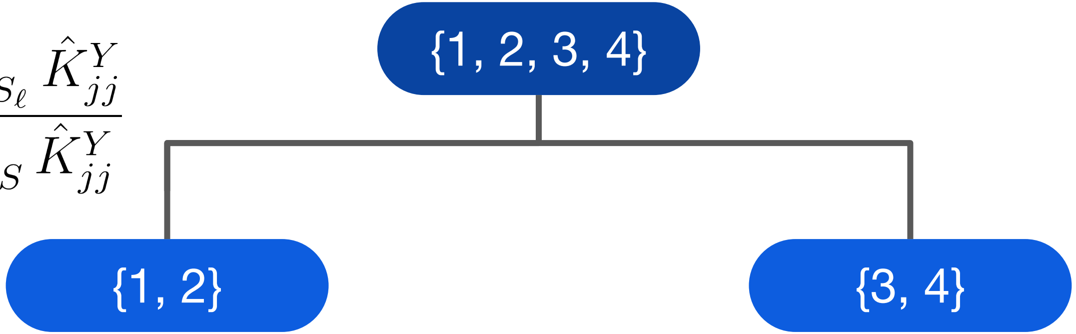
Our tree-based algorithm

Given tree T and C = $BB^\top$, sample from k-DPP with kernel $\hat{L} = (WB)^\top WB$

Add items one at at time, starting from $Y = \{\}$

Traverse tree once for each item addition:

$$\Pr(S_\ell \mid Y) = \frac{\sum_{j \in S_\ell} \hat{K}^Y_{jj}}{\sum_{j \in S} \hat{K}^Y_{jj}}$$



Our tree-based algorithm

With some algebra, we have:

$$\sum_{j \in S} \hat{K}^{Y}_{jj} = \mathbf{1}^\top [R \circ \Sigma^{(S)}]\mathbf{1} - \mathbf{1}^\top [(\hat{K}_Y)^{-1} \circ (F \Sigma^{(S)} F^\top)]\mathbf{1} = f(\Sigma^{(S)}, \hat{\lambda}, \hat{V}, W)$$

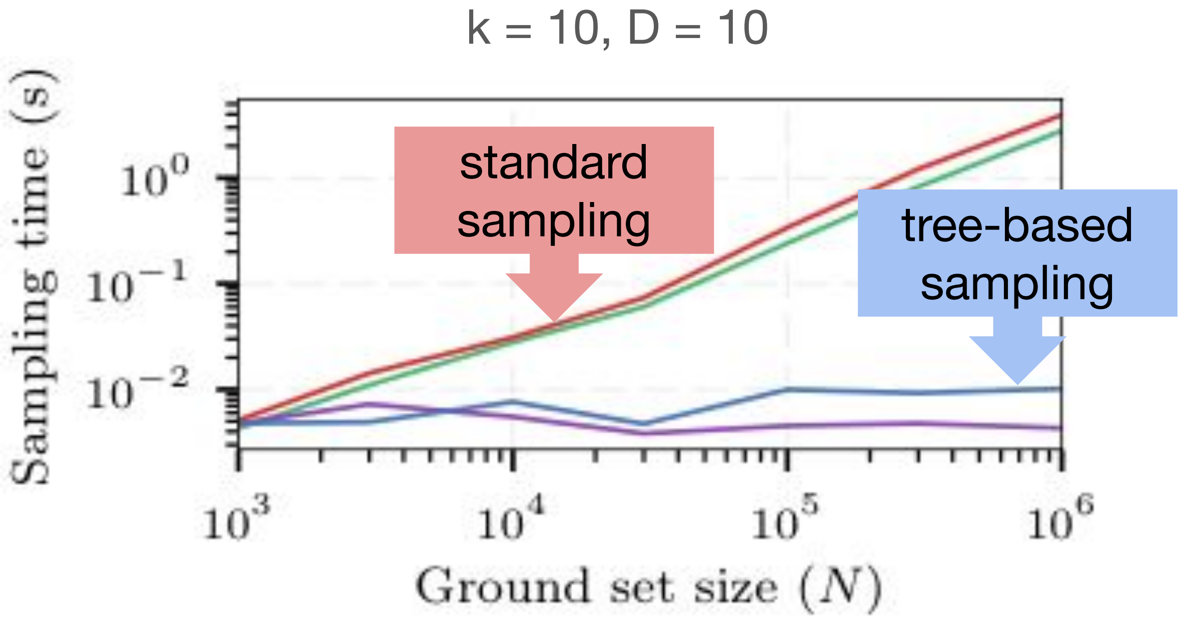
where: $\hat{\Gamma} = (1/\hat{\lambda})\ , \quad M = \hat{V}_{:,E}^\top W\ , \quad \hat{H} = \hat{\Gamma}_E M B_{:,Y}$

$$R = M^\top \hat{\Gamma}_E M\ , \quad F = \hat{H}^\top M$$

Computable in **O(kD²)** time ⇒ **O(kD² log N)** per tree traversal

Overall: O(k²D² log N + D³) time to sample

Experiments

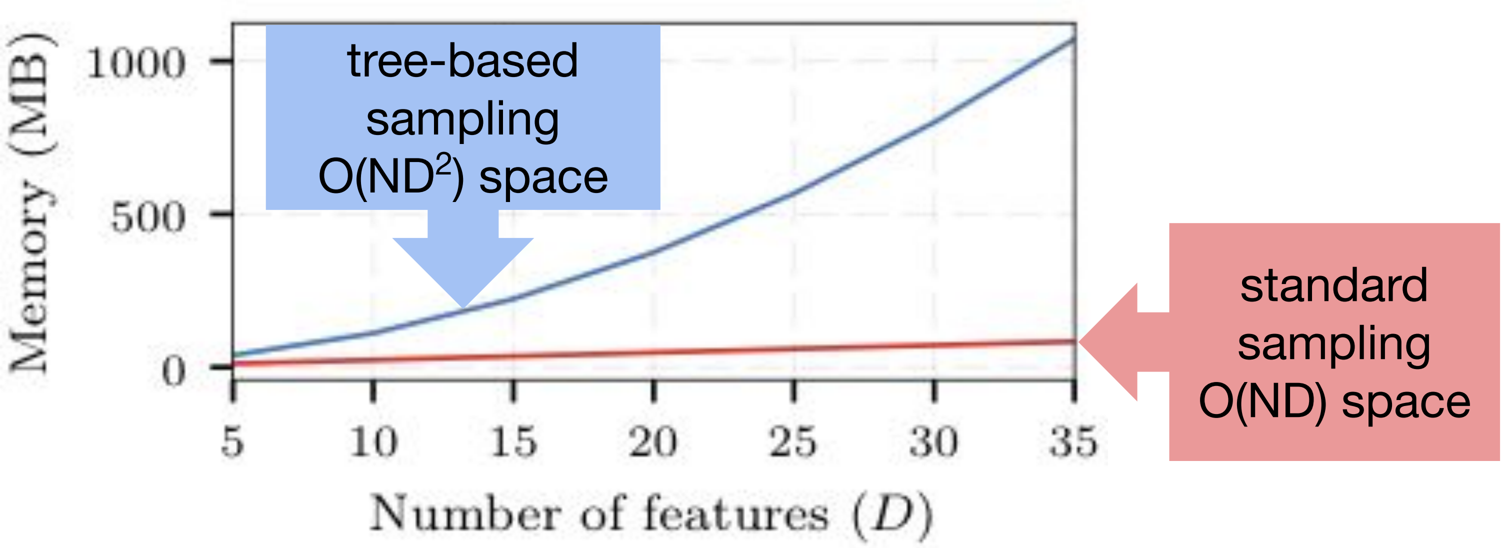


At N = 1 million: Standard sampling takes 4 secs; tree-based takes 0.01 secs

Experiments

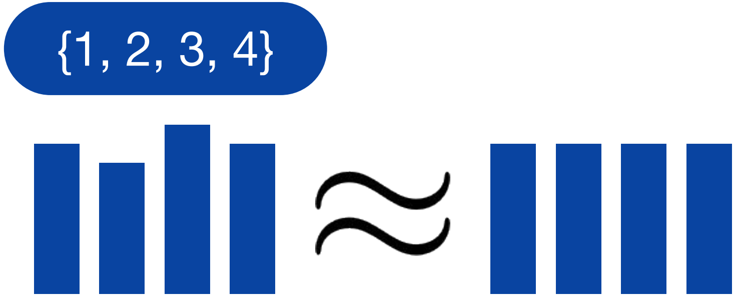
Cost: Memory required to store the tree

k = 10, N = 100,000



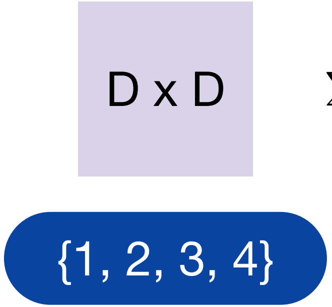
Approximate sampling

Main idea: If the distribution over items at a tree node is close to uniform, then don't bother moving further down the tree.



Approximate sampling

Bounding difference from uniform: Store max difference between node's matrix any child's matrix.



$$\tilde{\Sigma}^{(S)}_{\ell_1 \ell_2} = \max_{j \in S} \left| \Sigma^{(j)}_{\ell_1 \ell_2} - \frac{1}{|S|} \Sigma^{(S)}_{\ell_1 \ell_2} \right|$$

$$\tilde{\Sigma}^{\{1,2,3,4\}}_{\ell_1 \ell_2} = \max \left(\left| \Sigma^{(\{1\})}_{\ell_1 \ell_2} - \frac{1}{4} \Sigma^{(\{1,2,3,4\})}_{\ell_1 \ell_2} \right|, \left| \Sigma^{(\{2\})}_{\ell_1 \ell_2} - \frac{1}{4} \Sigma^{(\{1,2,3,4\})}_{\ell_1 \ell_2} \right|, \ldots \right)$$

Approximate sampling

$$\left|\Pr(j \mid S, Y) - \frac{1}{|S|}\right| \leq \frac{f(\tilde{\Sigma}^{(S)}, \hat{\lambda}, \hat{V}, W)}{f(\Sigma^{(S)}, \hat{\lambda}, \hat{V}, W)}$$

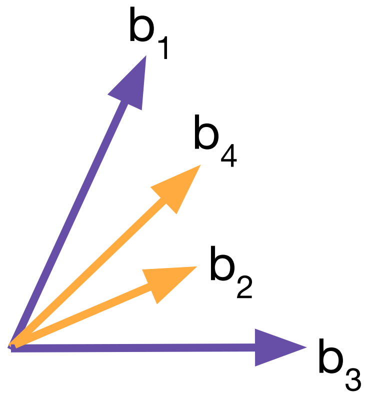
Early stopping algorithm: f-ratio < ϵ / |S|

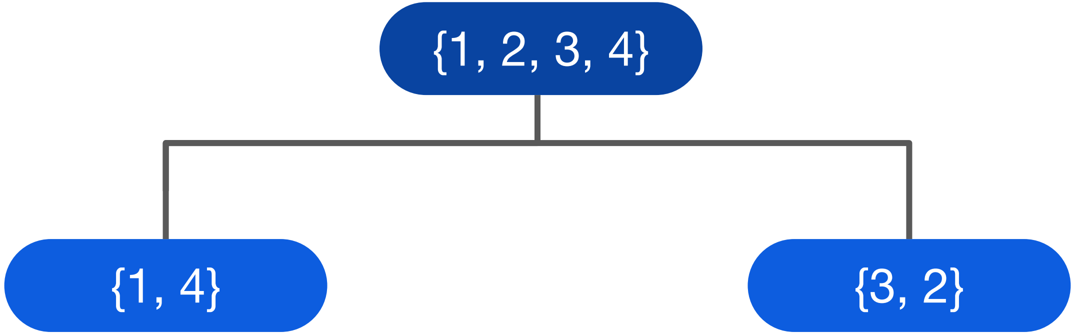
Sampling Y in order y_1, y_2, …, y_k:
|(true probability) - (probability with early stopping)| ≤ **$(1 + \epsilon)^k$ - 1**

Approximate sampling

Idea: Use node-splitting stage of tree construction to increase uniformity

Example: Find distinct items, seed left and right subtrees with these.





Conclusion

Sublinear, personalized k-DPP sampling after one-time preprocessing phase

Tree structure enables approximations

Poster #230