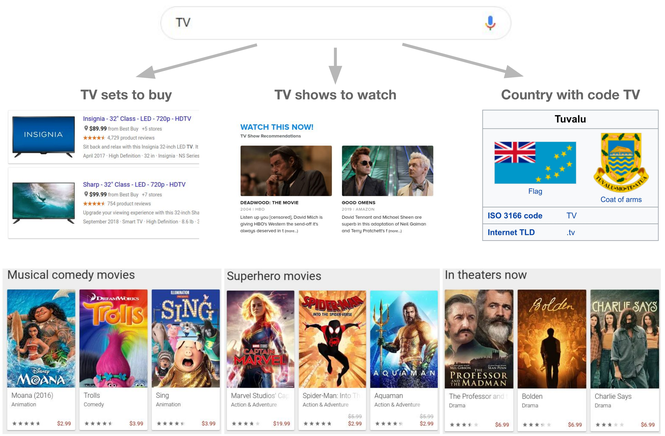
A Tree-Based Method for Fast Repeated Sampling of Determinantal Point Processes

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

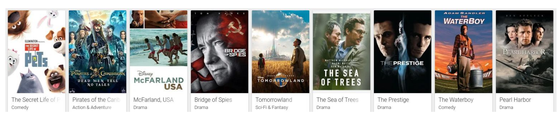
[1]Google Research NYC, [2]Massachusetts Institute of Technology

Motivation

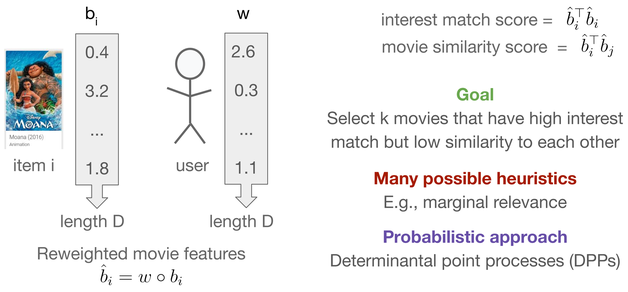
Diverse results are desirable in information retrieval and recommender systems



Determinantal Point Processes



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



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

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)

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

interest in movie j; interest in movie i; similarity of movies i and j

Sampling DPPs

- **Goal:** For each user, draw a size-k sample from their DPP
- **Problem:** Existing algorithms for k-DPP sampling are too expensive
  - D << N by construction or random projection
  - $O(ND^2)$ preprocessing on $L = B^\top B$
  - $O(\mathbf{N}k^2 + D^3)$ per personalized (W-weighted) sample afterwards
- **Our contribution:** Making repeated, personalized k-DPP sampling efficient:
  - $O(ND^2)$ preprocessing on $L = B^\top B$
  - $O(D^2k^2 \log N + D^3)$ per personalized (W-weighted) sample afterwards

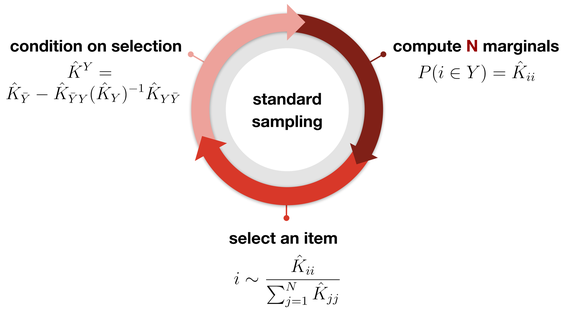
Standard dual algorithm

- **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:

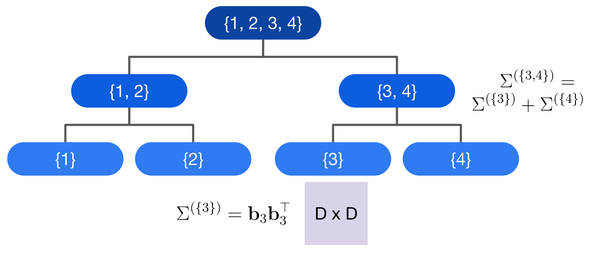
$$\hat{K} = \sum_{i \in E} \frac{1}{\hat{\lambda}_i} (\hat{B}^\top \hat{\mathbf{v}}_i)(\hat{B}^\top \hat{\mathbf{v}}_i)^\top$$

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



Our tree-based algorithm

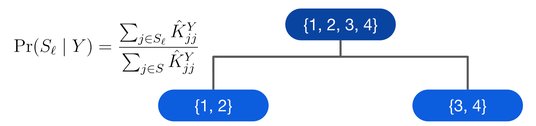
Key idea: In pre-processing, create a balanced binary tree of depth log N.



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:



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

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

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

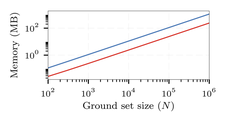
$$R = M^\top \hat{\Gamma}_E M \;, \quad F = \hat{H}^\top M$$

Computable in $O(kD^2)$ time ⇒ $O(kD^2 \log N)$ per tree traversal

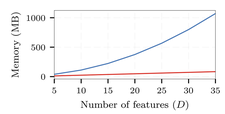
Overall: $O(k^2D^2 \log N + D^3)$ time to sample

Experiments

Preprocessing:

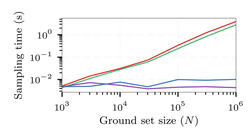


(a) $D = 10$



(b) $N = 10^5$

Sampling:



(a) $D = 10, k = 10$

Personalized Dual — Personalized Tree

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

Cost: Memory required to store the tree

Approximation

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

D x D

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

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

{1, 2, 3, 4}

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

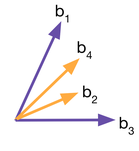
Early stopping algorithm: f-ratio < ϵ / |S|

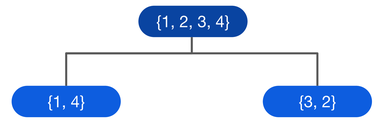
Sampling Y in order $y_1, y_2, \ldots, y_k$:

|(true probability) - (probability with early stopping)| ≤ $(1 + \epsilon)^k - 1$

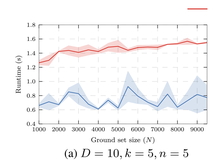
Idea: Use node-splitting stage of tree construction to increase uniformity

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

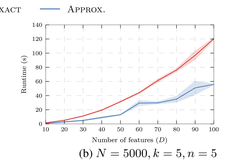




Exact — Approx.



(a) $D = 10, k = 5, n = 5$



(b) $N = 5000, k = 5, n = 5$