Maximizing Induced Cardinality Under a Determinantal Point Process

Google NYC MAD Science Team

Jennifer Gillenwater†, Alex Kulesza†, Zelda Mariet‡, Sergei Vassilvitskii†

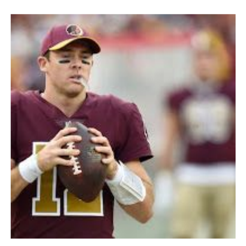
†Google Research New York, ‡Massachusetts Institute of Technology

Motivation

Diversity can be useful for recommender systems, for two main reasons:

- **Uncertainty** --- search engine query "java" has multiple interpretations


- **Exploration** --- news feed contents should span topics of user interest


Sports


Technology


Politics


Business

Determinantal Point Processes (DPPs)

DPPs are a means of trading off item **quality** with **diversity**. A DPP over n items is parameterized by an n-by-n matrix L whose diagonal captures item quality and whose off-diagonal captures item-item similarity.

Example --- Game app recommendation:

$$L = \begin{bmatrix} 2.5 & 2.3 & 0.5 & 0.2 & 0.2 \\ 2.3 & 2.4 & 0.4 & 0.2 & 0.2 \\ 0.5 & 0.4 & 2.0 & 0.1 & 0.1 \\ 0.2 & 0.2 & 0.1 & 1.5 & 1.4 \\ 0.2 & 0.2 & 0.1 & 1.4 & 1.4 \end{bmatrix}$$

Probability of a set:

$$\mathcal{P}_L(S) \propto \det(L_S)$$

Example:

quality - similarity

$$\det(L_{1,2}) = L_{11}L_{22} - L_{12}^2 = 2.5 * 2.4 - 2.3^2$$

Highest-probability set:

$$\max_{S:|S|=3} \det(L_S) =$$

Training a DPP Recommender System

Goal --- Recommend k items from a much larger set of n items.

Training data --- r previously-recommended k-sets: $[S_1, S_2, \dots, S_r]$ and resulting user engagement sets: $[E_1, E_2, \dots, E_r]$ (e.g., which items a user clicked on, or watched, or read, etc.).

Likelihood objective --- Modeling user behavior as a DPP, maximize probability of engaged sets by optimizing parameters θ that define L.

$$\max_\theta \sum_{i=1}^r \log(\mathcal{P}_{L^{(i)}(\theta)}(E_i))$$

$$E_i \subseteq S_i$$

$$|S_i| \times |S_i| \text{ matrix}$$

Generating Recommendations

Standard inference-time objective --- Maximum a posteriori (**MAP**):

MAP $\quad \max_{S:|S|=k} \mathcal{P}_L(S) = \max_{S:|S|=k} \det(L_S)$

Mis-match --- Training modeled *engaged-with* items as draws from a DPP, not the set of all *recommended* items. Hence, this MAP objective really represents the probability that a user will engage with *every* item in S.

More natural goal --- Recommend S that maximizes expected cardinality of the induced engagements E; maximum induced cardinality (**MIC**):

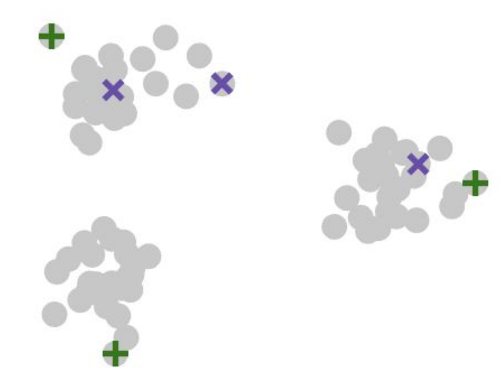
MIC $\quad \max_{S:|S|=k} \sum_{E \subseteq S} |E| \mathcal{P}_{L_S}(E)$

Main contribution of this work --- Proposal and analysis of MIC.

MAP Failure Case

Low rank kernels --- If rank(L) < k, then **MAP** has equal value (zero) for all size-k sets. **MIC** on the other hand differentiates among k-sets.

Example --- Each item is represented by a 2-dimensional feature vector and data forms 3 clusters. **MIC** selects one item in each cluster, while **MAP** selects 3 items at random.



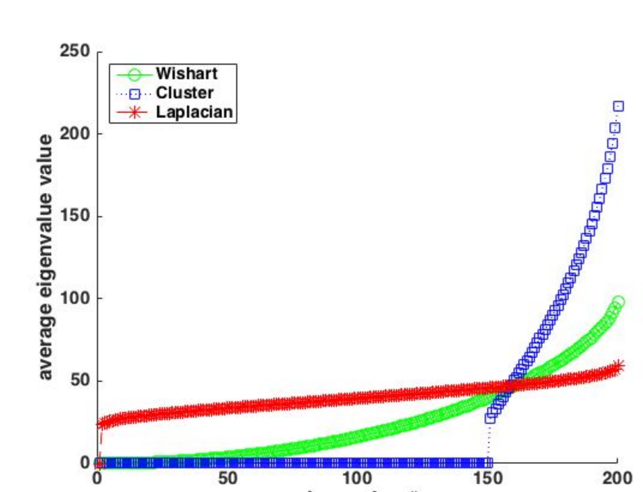
Properties of Induced Cardinality

- Computable in $O(k^3)$ time:

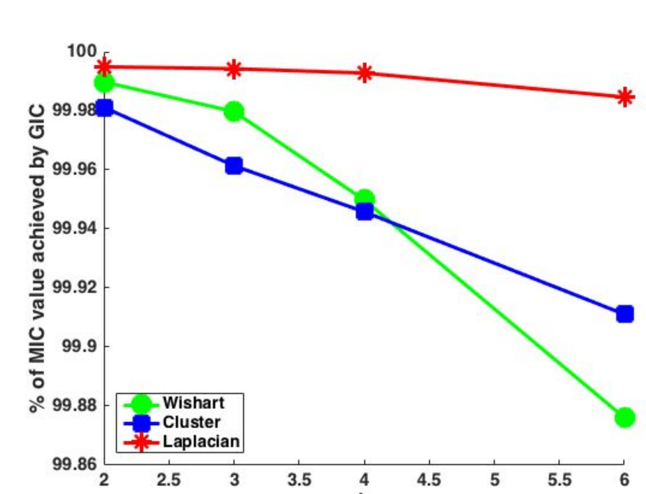
$$f(S) = \sum_{E \subseteq S} |E| \mathcal{P}_{L_S}(E) = \boxed{\mathrm{Tr}(I - (L_S + I)^{-1})}$$

- Monotone increasing and fractionally subadditive
- Submodular if L is an M-matrix (all off-diagonal entries are non-positive)
- NP-hard to maximize

Direct Optimization



Kernel matrix types --- Experimented with three types of L matrices, each with a distinct spectrum: **Wishart**, **cluster** (n items divided into k Gaussian clusters), and **graph Laplacian** (n-node graph, Erdos-Renyi model with edge existence parameter p = 0.2).



Small kernel: n = 12
MIC --- Exact max.
GIC --- Greedy algorithm on f. No approximation guarantees in general, but performs well in practice. Best on Laplacians (which are M-matrices), and achieves more than 99% of maximum possible value for other kernels.

Series Approximation

Geometric series representation ---

- Define: $m = \lambda_n(L) + 1$ and $B = (m-1)I - L$
- Then using the Neumann series representation of the matrix inverse:

$$f(S) = |S| - \sum_{i=0}^{\infty} \frac{\mathrm{Tr}(B_S^i)}{m^{i+1}}$$

- The first few terms are a **monotone submodular approximation**:

$$\boxed{\hat{f}(S) = |S| - \frac{|S|}{m} - \frac{\mathrm{Tr}(B_S)}{m^2} - \frac{\mathrm{Tr}(B_S^2)}{m^3}}$$

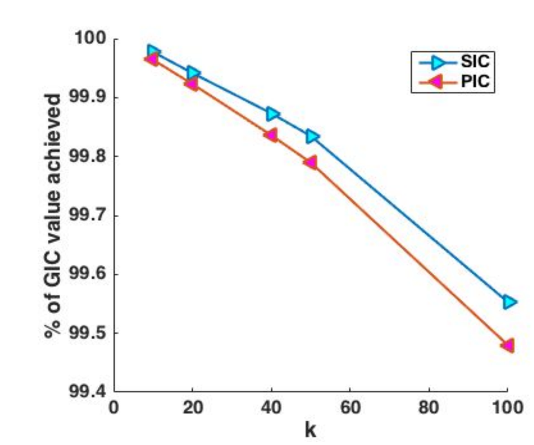
Goodness of approximation --- For all sets S of size k:

$$\frac{f(S)}{\hat{f}(S)} \geq 1 - \frac{mr_3}{(m-1)k - r_1 - r_2}, \quad \text{with } r_i = \sum_{j=n-k+1}^{n} \left(\frac{\lambda_j(B)}{m}\right)^i$$

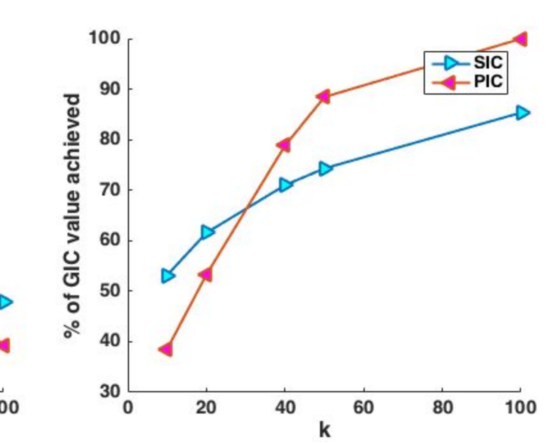
Best when smaller eigenvalues of L are close to $\lambda_n(L)$.

Optimization of Approximations

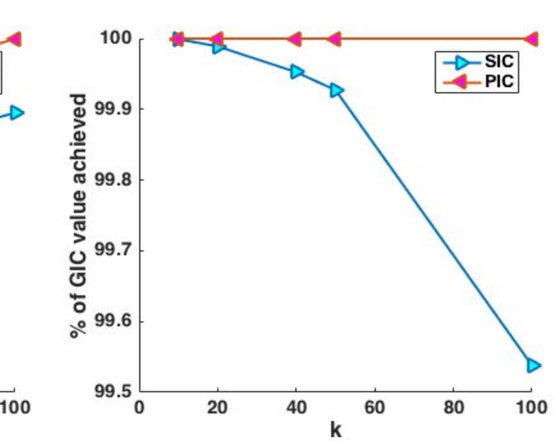
Kernel size: n = 200.
PIC --- Greedy algorithm on f after projecting L to an M-matrix.
SIC --- Greedy algorithm on the (submodular) series approximation.



Wishart kernels

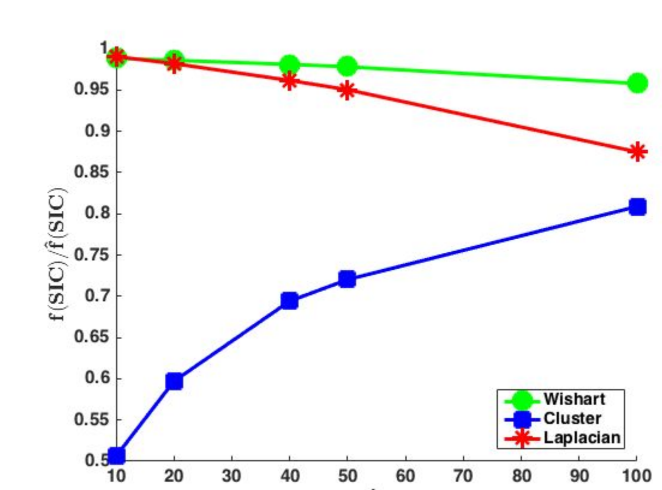


Cluster kernels



Laplacian kernels

PIC performance --- PIC does well when the projection to M-matrix does not alter the objective too much; graph Laplacian kernels are already M-matrices, so PIC is equivalent to GIC in the third graph.



SIC performance --- SIC does well for Wishart and Laplacian kernels, but struggles with the cluster kernels. This is because the $f/\hat{f}$ ratio decays slowly with k for Wishart and Laplacian, but grows dramatically with k for cluster kernels. (See eigenvalue plot.)`

Runtime --- GIC (and PIC, ignoring the initial projection step) are $O(nk^3)$ while SIC is a factor of k faster. For n = 500 and k = 250, SIC runs about 18 times faster than GIC.

Conclusion --- Use SIC when speed is important, or when approximation guarantee is required.