Determinantal Point Processes (DPPs)

Given a ground set \( Y = \{1, \ldots, d\} \) and a positive definite matrix \( L \in \mathbb{R}^{d \times d} \),
\[
Pr(X) \propto \det(L_X)
\]
for \( X \subseteq Y \),
where \( L_X \) is a submatrix of \( L \) indexed by items of \( X \).

- DPPs are probabilistic models capturing both diversity and item quality of subsets.
- Most inference tasks (including normalization, marginalization, conditioning and sampling) can be done in \( O(d^3) \).
- However, MAP inference is known as \( \text{NP-hard} \) problem, that is,
\[
\arg \max_{X \subseteq Y} \det(L_X).
\]
- The MAP inference of DPP has been used for many machine learning applications, e.g., text/video summarization, change-point detection, and informative image search.

Our Contribution: Faster MAP Inference of DPP

Since \( \log \det \) is a submodular function, greedy algorithms for approximating MAP of DPP have been of typical choice.

- A naïve greedy algorithm requires \( O(d^4) \) operations.
  
<table>
<thead>
<tr>
<th>algorithm</th>
<th>complexity</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Minoux, 1978]</td>
<td>( O(d^4) )</td>
<td>accelerated version of a naïve greedy algorithm</td>
</tr>
<tr>
<td>[Buchbinder et al., 2015]</td>
<td>( O(d^4) )</td>
<td>symmetric greedy algorithm</td>
</tr>
<tr>
<td>[Gillenwater et al., 2012]</td>
<td>( O(d^4) )</td>
<td>multilinear softmax extension</td>
</tr>
</tbody>
</table>

We propose faster greedy algorithms requiring \( O(d^3) \) operations.

First Ideas: Taylor Expansion

Greedy algorithms require computing the following marginal gains:
\[
\log \det L_{X \cup \{i\}} - \log \det L_X
\]
For their efficient computations, our key ideas are:

1. First-order Taylor expansion for Log-determinant
\[
\log \det L_{X \cup \{i\}} - \log \det L_X \approx \left( \nabla \log \det L_X \right) (L_{X \cup \{i\}} - L_X).
\]

- \( T_X \) is the average of \( L_{X \cup \{i\}} \) for \( i \in Y \setminus X \).
- \( L_{X \cup \{i\}} \) and \( T_X \) differ only single column and row.
- Single column of \( T_X \) is computed by a linear solver, e.g., conjugate gradient descent.

2. Partitioning
   - For much tighter approximation, we divide \( Y \setminus X \) into \( p \) partitions so that
   \[
   \|L_{X \cup \{i\}} - T_X\|_F \gg \|L_{X \cup \{i\}} - T_X^q\|_F,
   \]
   where \( i \) is in the partition \( j \) \( \in \{1, \ldots, p\} \).
   - To compute the marginal gains, we need to calculate extra term \((*)\):
     \[
     \log \det L_{X \cup \{i\}} - \log \det L_X \approx \left( \nabla \log \det L_X \right) (L_{X \cup \{i\}} - T_X) + \left( \log \det T_X^q - \log \det L_X \right).
     \]
   - \((*)\) is also computable by a linear solver under Schur complement.

The overall complexity becomes \( O(d^3) \) because we choose \( p = O(1) \) and
- In each greedy step, a linear solver can be used to compute both Taylor approximation and \((*)\), thus \( O(d \times d^2) \) operations are required.
- The total number of greedy steps is at most \( d \).

Second Ideas: Batch Strategy

We consider adding \( k \)-batch subset (instead of single element)
\[
X \leftarrow X \cup I \quad \text{for some } |I| = k > 1
\]
so that the number of greedy steps can be reduced at most \( k \) times.

1. Sampling random batches
   - For the optimal \( k \)-batch, one has to investigate \( \binom{|Y|}{k} \) subsets.
   - This is expensive. Instead, we randomly sample batches and add the best of them to the current set.

2. Log-determinant approximation under sharing randomness
   - For \( k \)-batch strategy, one can compute the extra term \((*)\), i.e., \( \log \det T_X^q - \log \det L_X \), by running a linear solver \( k \) times.
   - Alternatively, we suggest estimating all log-determinants \( \log \det T_X^q \) by running a log-determinant approximation scheme (LDAS) [Han et al., 2015], but only once.

\[
\text{method} \quad \text{complexity} \quad \text{number of calls} \quad \text{objective}
\]
| linear solver | \( O(d^3) \) | \( k \) | \( \log \det T_X^q - \log \det L_X \)
| LDAS          | \( O(d^3) \) | \( 1 \) | \( \log \det T_X^q \)

LDAS approximates \( \log \det T_X^q \) using independent random vectors.

We suggest to run LDAS using the same random vectors for estimating all \( \log \det T_X^q \).

Observe that running LDAS’s under sharing random vectors is better for comparing \( \log \det T_X^q \), i.e., marginal gains.

- We provide the following error bound of LDAS under sharing random vectors, where \( A = T_X^q \) and \( B = T_X^q \).

\[
\text{Theorem} \quad (\text{Han, Prabhanjan, Park and Shin, 2017}). \quad \text{Suppose } A, B \text{ are positive definite matrices whose eigenvalues are in } [\delta, 1] \text{ for } \delta > 0. \quad \text{Let } T_A, T_B \text{ be the estimations of } \log \det A, \log \det B \text{ by LDAS using the same } n \text{ random vectors for both. Then, it holds that}
\]
\[
\text{Var}[T_A - T_B] \leq \frac{32M^2\rho^2}{m(\rho - 1)} \left( 1 - 2\rho \right) |A - B|^2_F
\]
where \( M = 5\log(2/\delta) \) and \( \rho = 1 + \frac{2}{\sqrt{MN}} \).

On the other hand, the variance of LDAS under independent random vectors depends on \( |A|^2_F + |B|^2_F \), which is significantly larger than \( |A - B|^2_F \) in our case.


