Faster Greedy MAP Inference for Determinantal Point Processes

Insu Han[1], Prabhanjan Kambadur[2], KyoungSoo Park[1], Jinwoo Shin [1]

[1]Korea Advanced Institute of Science and Technology (KAIST)
[2]Bloomberg LP

ICML 2017, Sydney
Aug 8th, 2017

Outline: Faster Greedy DPP MAP Inference

Outline: Faster Greedy DPP MAP Inference

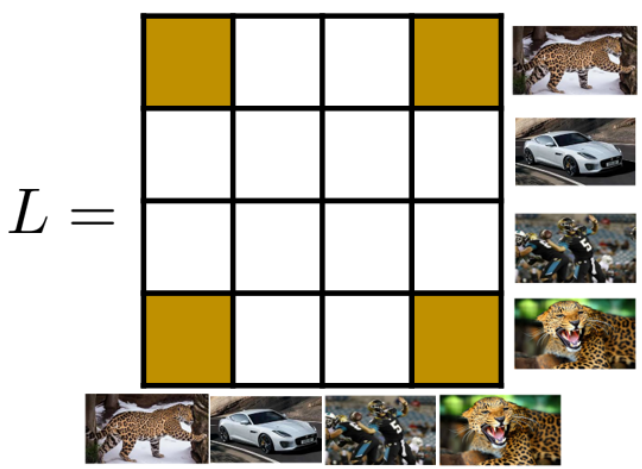
Determinantal Point Processes

Formal definition

- A point process of given finite ground set $\mathcal{Y} = \{1, \ldots, d\}$.
- Given a positive definite kernel matrix $L \in \mathbb{R}^{d \times d}$ (called L-ensemble), probability on 2^d subsets of $\mathcal{Y}$ is defined as

$$\Pr(X) \propto \det(L_X) \quad \text{for } X \subseteq \mathcal{Y},$$

where L_X is a submatrix of L indexed by items of X.



ground set $\mathcal{Y}$

Determinantal Point Processes

DPPs capture both diversity and quality.

Determinantal Point Processes

DPPs capture both diversity and quality.

They play an important role in many machine learning tasks including

- Text/video summarization,
- Informative image search,
- Change-point detection in time-series data.

Determinantal Point Processes

DPPs capture both diversity and quality.

They play an important role in many machine learning tasks including

- Text/video summarization,
- Informative image search,
- Change-point detection in time-series data.

Most probabilistic inference tasks are tractable.

Determinantal Point Processes

DPPs capture both diversity and quality.

They play an important role in many machine learning tasks including

- Text/video summarization,
- Informative image search,
- Change-point detection in time-series data.

Most probabilistic inference tasks are tractable.

- Inferences including normalization, marginalization, conditioning and sampling can be done in $O(d^3)$ time.

Determinantal Point Processes

DPPs capture both diversity and quality.

They play an important role in many machine learning tasks including

- Text/video summarization,
- Informative image search,
- Change-point detection in time-series data.

Most probabilistic inference tasks are tractable.

- Inferences including normalization, marginalization, conditioning and sampling can be done in $O(d^3)$ time.
- Maximum a posteriori (MAP) inference is known as **NP-hard** problem [Ko et al., 1995].

Determinantal Point Processes

DPPs capture both diversity and quality.

They play an important role in many machine learning tasks including

- Text/video summarization,
- Informative image search,
- Change-point detection in time-series data.

Most probabilistic inference tasks are tractable.

- Inferences including normalization, marginalization, conditioning and sampling can be done in $O(d^3)$ time.
- Maximum a posteriori (MAP) inference is known as **NP-hard** problem [Ko et al., 1995].
- Can we find an approximation of DPP MAP inference in $O(d^3)$ time?

DPP MAP Inference

MAP Inference:

$$\underset{X \subseteq \mathcal{Y}}{\operatorname{argmax}} \det L_X$$

DPP MAP Inference

MAP Inference:

$$\underset{X \subseteq \mathcal{Y}}{\operatorname{argmax}} \det L_X$$

- To solve this, we have to investigate all subsets.

DPP MAP Inference

MAP Inference:

$$\underset{X \subseteq \mathcal{Y}}{\operatorname{argmax}} \det L_X$$

- To solve this, we have to investigate all subsets.
- Since $\log\det$ is *submodular*, greedy algorithms for approximating MAP inference of DPP have been of typical choice.

$X \leftarrow \emptyset$
while such i^* exists **do**
  $i^* \leftarrow \underset{i \in \mathcal{Y} \setminus X}{\operatorname{argmax}} \log\det L_{X \cup \{i\}} - \log\det L_X$
  $X \leftarrow X \cup i^*$
end

DPP MAP Inference

MAP Inference:

$$\underset{X \subseteq \mathcal{Y}}{\operatorname{argmax}} \det L_X$$

- To solve this, we have to investigate all subsets.
- Since $\log\det$ is *submodular*, greedy algorithms for approximating MAP inference of DPP have been of typical choice.

$X \leftarrow \emptyset$
while such i^* exists **do**
  $i^* \leftarrow \underset{i \in \mathcal{Y} \setminus X}{\operatorname{argmax}} \log\det L_{X \cup \{i\}} - \log\det L_X$
  $X \leftarrow X \cup i^*$
end

  - The exact calculation of the marginal gain needs $O(d^3)$ operations for each i. In overall, it has $O(d^5)$ time complexity.

DPP MAP Inference

MAP Inference:

$$\underset{X \subseteq \mathcal{Y}}{\operatorname{argmax}} \det L_X$$

- To solve this, we have to investigate all subsets.
- Since $\log\det$ is *submodular*, greedy algorithms for approximating MAP inference of DPP have been of typical choice.

$X \leftarrow \emptyset$
while such i^* exists **do**
  $i^* \leftarrow \underset{i \in \mathcal{Y} \setminus X}{\operatorname{argmax}} \log\det L_{X \cup \{i\}} - \log\det L_X$
  $X \leftarrow X \cup i^*$
end

  - The exact calculation of the marginal gain needs $O(d^3)$ operations for each i. In overall, it has $O(d^5)$ time complexity.

- Our main contribution is to propose faster greedy algorithms requiring $O(d^3)$ operations.

Outline: Faster Greedy DPP MAP Inference

Naïve Implementations

Computational bottleneck of greedy algorithm: for $i \in \mathcal{Y} \setminus X$,

$$\log\det L_{X\cup\{i\}} - \log\det L_X$$

Naïve Implementations

Computational bottleneck of greedy algorithm: for $i \in \mathcal{Y} \setminus X$,

$$\log\det L_{X\cup\{i\}} - \log\det L_X$$

1. One can apply log-determinant approximation [Han et al., 2015].

Naïve Implementations

Computational bottleneck of greedy algorithm: for $i \in \mathcal{Y} \setminus X$,

$$\log\det L_{X\cup\{i\}} - \log\det L_X$$

1. One can apply log-determinant approximation [Han et al., 2015].
   - $\log\det L_{X\cup\{i\}}$ and $\log\det L_X$ can be estimated in $O(d^2)$ time.

Naïve Implementations

Computational bottleneck of greedy algorithm: for $i \in \mathcal{Y} \setminus X$,

$$\log\det L_{X\cup\{i\}} - \log\det L_X = \log\left(L_{i,i} - L_{i,X}L_X^{-1}L_{X,i}\right).$$

1. One can apply log-determinant approximation [Han et al., 2015].
   - $\log\det L_{X\cup\{i\}}$ and $\log\det L_X$ can be estimated in $O(d^2)$ time.
2. Alternatively, it can be computed by solving a linear system.

Naïve Implementations

Computational bottleneck of greedy algorithm: for $i \in \mathcal{Y} \setminus X$,

$$\log\det L_{X\cup\{i\}} - \log\det L_X = \log\left(L_{i,i} - L_{i,X}L_X^{-1}L_{X,i}\right).$$

1. One can apply log-determinant approximation [Han et al., 2015].
   - $\log\det L_{X\cup\{i\}}$ and $\log\det L_X$ can be estimated in $O(d^2)$ time.
2. Alternatively, it can be computed by solving a linear system.
   - One can use conjugate gradient method (CG) of $O(d^2)$ time.

Naïve Implementations

Computational bottleneck of greedy algorithm: for $i \in \mathcal{Y} \setminus X$,

$$\log\det L_{X\cup\{i\}} - \log\det L_X = \log\left(L_{i,i} - L_{i,X}L_X^{-1}L_{X,i}\right).$$

1. One can apply log-determinant approximation [Han et al., 2015].
   - $\log\det L_{X\cup\{i\}}$ and $\log\det L_X$ can be estimated in $O(d^2)$ time.
2. Alternatively, it can be computed by solving a linear system.
   - One can use conjugate gradient method (CG) of $O(d^2)$ time.

- Both implementations have $O(d^4)$ total time complexity since one has to perform both greedy searches and greedy updates $O(d)$ times.

Naïve Implementations

Computational bottleneck of greedy algorithm: for $i \in \mathcal{Y} \setminus X$,

$$\log\det L_{X\cup\{i\}} - \log\det L_X = \log\left(L_{i,i} - L_{i,X}L_X^{-1}L_{X,i}\right).$$

1. One can apply log-determinant approximation [Han et al., 2015].
   - $\log\det L_{X\cup\{i\}}$ and $\log\det L_X$ can be estimated in $O(d^2)$ time.
2. Alternatively, it can be computed by solving a linear system.
   - One can use conjugate gradient method (CG) of $O(d^2)$ time.

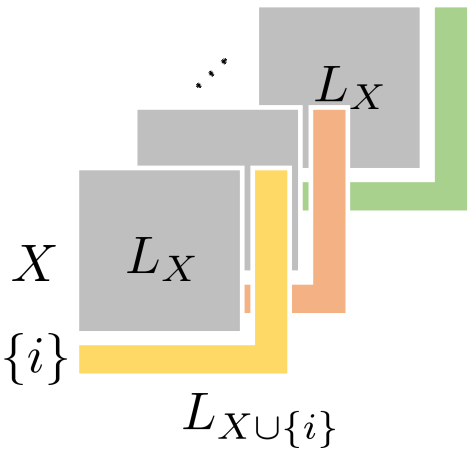
- Both implementations have $O(d^4)$ total time complexity since one has to perform both greedy searches and greedy updates $O(d)$ times.
- How can we estimate the marginal gains much faster?

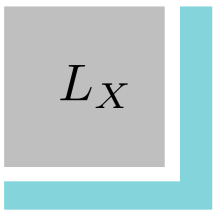
First Ideas: Taylor Expansion

1. First-order Taylor Expansion for Log-determinant

Suppose $\overline{L}_X$ is the average of $L_{X\cup\{i\}}$ for $i \in \mathcal{Y} \setminus X$. Then,

$$\log\det L_{X\cup\{i\}} - \log\det \overline{L}_X \quad \approx \quad \left\langle \overline{L}_X^{-1}, L_{X\cup\{i\}} - \overline{L}_X \right\rangle.$$





$$\overline{L}_X = \frac{1}{|\mathcal{Y} \setminus X|} \sum_{i \in \mathcal{Y}\setminus X} L_{X\cup\{i\}}$$

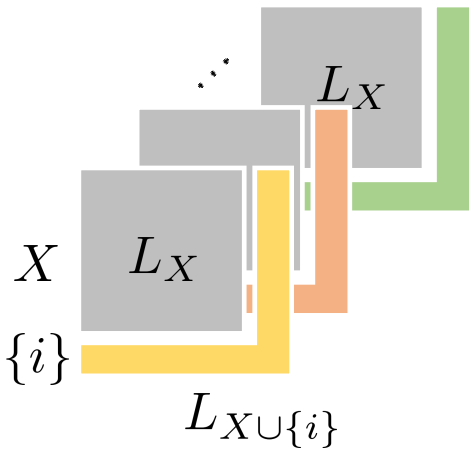
First Ideas: Taylor Expansion

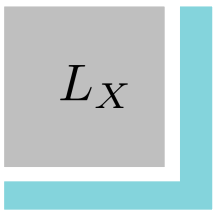
1. First-order Taylor Expansion for Log-determinant

Suppose $\overline{L}_X$ is the average of $L_{X \cup \{i\}}$ for $i \in \mathcal{Y} \setminus X$. Then,

$$\log\det L_{X\cup\{i\}} - \log\det \overline{L}_X \quad \approx \quad \left\langle \overline{L}_X^{-1}, L_{X\cup\{i\}} - \overline{L}_X \right\rangle.$$

- $L_{X\cup\{i\}}$ and $\overline{L}_X$ differ only single column and row.





$$\overline{L}_X = \frac{1}{|\mathcal{Y} \setminus X|} \sum_{i \in \mathcal{Y} \setminus X} L_{X \cup \{i\}}$$

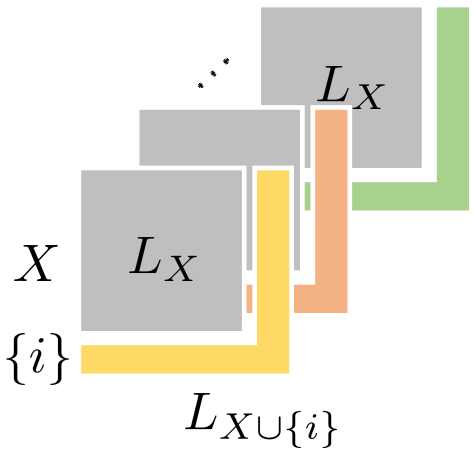
First Ideas: Taylor Expansion

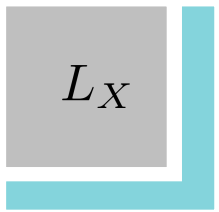
1. First-order Taylor Expansion for Log-determinant

Suppose $\overline{L}_X$ is the average of $L_{X\cup\{i\}}$ for $i \in \mathcal{Y} \setminus X$. Then,

$$\log\det L_{X\cup\{i\}} - \log\det \overline{L}_X \quad \approx \quad \left\langle \overline{L}_X^{-1}, L_{X\cup\{i\}} - \overline{L}_X \right\rangle.$$

- $L_{X\cup\{i\}}$ and $\overline{L}_X$ differ only single column and row.
- To compute the right-hand side (i.e., matrix inner-product), it is enough to compute single column of $\overline{L}_X^{-1}$ and vector inner-product.





$$\overline{L}_X = \frac{1}{|\mathcal{Y} \setminus X|} \sum_{i \in \mathcal{Y}\setminus X} L_{X\cup\{i\}}$$

First Ideas: Taylor Expansion

1. First-order Taylor Expansion for Log-determinant

Suppose $\overline{L}_X$ is the average of $L_{X\cup\{i\}}$ for $i \in \mathcal{Y} \setminus X$. Then,

$$\log\det L_{X\cup\{i\}} - \log\det \overline{L}_X \quad \approx \quad \left\langle \overline{L}_X^{-1}, L_{X\cup\{i\}} - \overline{L}_X \right\rangle.$$

- $L_{X\cup\{i\}}$ and $\overline{L}_X$ differ only single column and row.
- To compute the right-hand side (i.e., matrix inner-product), it is enough to compute single column of $\overline{L}_X^{-1}$ and vector inner-product.
  - One can use CG of $O(d^2)$ complexity.

First Ideas: Taylor Expansion

1. First-order Taylor Expansion for Log-determinant

Suppose $\overline{L}_X$ is the average of $L_{X\cup\{i\}}$ for $i \in \mathcal{Y} \setminus X$. Then,

$$\log\det L_{X\cup\{i\}} - \log\det \overline{L}_X \quad \approx \quad \left\langle \overline{L}_X^{-1}, L_{X\cup\{i\}} - \overline{L}_X \right\rangle.$$

- $L_{X\cup\{i\}}$ and $\overline{L}_X$ differ only single column and row.
- To compute the right-hand side (i.e., matrix inner-product), it is enough to compute single column of $\overline{L}_X^{-1}$ and vector inner-product.
  - One can use CG of $O(d^2)$ complexity.
  - Vector inner-product requires $O(d)$ operations for each i, thus $O(d^2)$ time in total.

First Ideas: Taylor Expansion

1. First-order Taylor Expansion for Log-determinant

Suppose $\overline{L}_X$ is the average of $L_{X\cup\{i\}}$ for $i \in \mathcal{Y} \setminus X$. Then,

$$\log\det L_{X\cup\{i\}} - \log\det \overline{L}_X \quad \approx \quad \left\langle \overline{L}_X^{-1}, L_{X\cup\{i\}} - \overline{L}_X \right\rangle.$$

- $L_{X\cup\{i\}}$ and $\overline{L}_X$ differ only single column and row.
- To compute the right-hand side (i.e., matrix inner-product), it is enough to compute single column of $\overline{L}_X^{-1}$ and vector inner-product.
  - One can use CG of $O(d^2)$ complexity.
  - Vector inner-product requires $O(d)$ operations for each i, thus $O(d^2)$ time in total.
- Thus, each greedy update can be done in $O(d^2)$ time.

First Ideas: Taylor Expansion

1. First-order Taylor Expansion for Log-determinant

Suppose $\overline{L}_X$ is the average of $L_{X\cup\{i\}}$ for $i \in \mathcal{Y} \setminus X$. Then,

$$\log\det L_{X\cup\{i\}} - \log\det \overline{L}_X \quad \approx \quad \left\langle \overline{L}_X^{-1}, L_{X\cup\{i\}} - \overline{L}_X \right\rangle .$$

- $L_{X\cup\{i\}}$ and $\overline{L}_X$ differ only single column and row.
- To compute the right-hand side (i.e., matrix inner-product), it is enough to compute single column of $\overline{L}_X^{-1}$ and vector inner-product.
  - One can use CG of $O(d^2)$ complexity.
  - Vector inner-product requires $O(d)$ operations for each i, thus $O(d^2)$ time in total.
- Thus, each greedy update can be done in $O(d^2)$ time.
- How can we make the approximation much tighter ?

First Ideas: Taylor Expansion

2. Partitioning for Tighter Approximation

We partition $\mathcal{Y} \setminus X$ into $p > 1$ distinct subsets so that

$$\|L_{X\cup\{i\}} - \overline{L}_X\|_F \gg \|L_{X\cup\{i\}} - \overline{L}_X^{(j)}\|_F,$$

where i is in the partition $j \in \{1, \ldots, p\}$ and $\overline{L}_X^{(j)}$ is the average of $L_{X\cup\{i\}}$'s in the partition j.

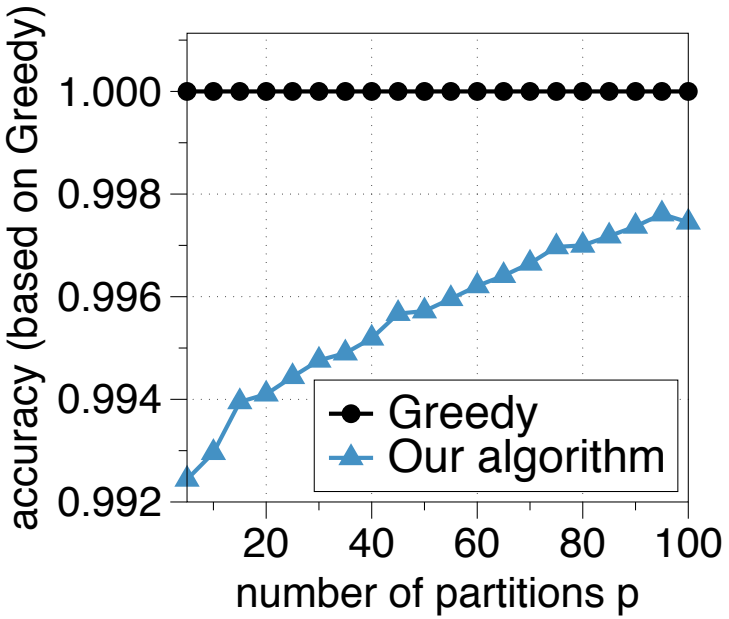
First Ideas: Taylor Expansion

2. Partitioning for Tighter Approximation

We partition $\mathcal{Y} \setminus X$ into $p > 1$ distinct subsets so that

$$\|L_{X\cup\{i\}} - \overline{L}_X\|_F \gg \|L_{X\cup\{i\}} - \overline{L}_X^{(j)}\|_F,$$

where i is in the partition $j \in \{1, \ldots, p\}$ and $\overline{L}_X^{(j)}$ is the average of $L_{X\cup\{i\}}$'s in the partition j.



First Ideas: Taylor Expansion

2. Partitioning for Tighter Approximation

We partition $\mathcal{Y} \setminus X$ into $p > 1$ distinct subsets so that

$$\|L_{X\cup\{i\}} - \overline{L}_X\|_F \gg \|L_{X\cup\{i\}} - \overline{L}_X^{(j)}\|_F,$$

where i is in the partition $j \in \{1, \ldots, p\}$ and $\overline{L}_X^{(j)}$ is the average of $L_{X\cup\{i\}}$'s in the partition j.

- To compute the marginal gains, we calculate the extra term $(*)$:

$$\begin{aligned}&\log\det L_{X\cup\{i\}} - \log\det L_X \\ &\approx \underbrace{\left\langle (\overline{L}_X^{(j)})^{-1}, L_{X\cup\{i\}} - \overline{L}_X^{(j)} \right\rangle}_{\text{can be computed by CG}} + \underbrace{\left(\log\det \overline{L}_X^{(j)} - \log\det L_X\right)}_{(*)}.\end{aligned}$$

First Ideas: Taylor Expansion

2. Partitioning for Tighter Approximation

We partition $\mathcal{Y} \setminus X$ into $p > 1$ distinct subsets so that

$$\|L_{X\cup\{i\}} - \overline{L}_X\|_F \gg \|L_{X\cup\{i\}} - \overline{L}_X^{(j)}\|_F,$$

where i is in the partition $j \in \{1, \ldots, p\}$ and $\overline{L}_X^{(j)}$ is the average of $L_{X\cup\{i\}}$'s in the partition j.

- To compute the marginal gains, we calculate the extra term $(*)$:

$$\begin{aligned}&\log\det L_{X\cup\{i\}} - \log\det L_X \\ &\approx \underbrace{\left\langle (\overline{L}_X^{(j)})^{-1}, L_{X\cup\{i\}} - \overline{L}_X^{(j)} \right\rangle}_{\text{can be computed by CG}} + \underbrace{\left(\log\det \overline{L}_X^{(j)} - \log\det L_X\right)}_{(*)}.\end{aligned}$$

- $(*)$ can also be computed by CG under Schur complement.

First Ideas: Taylor Expansion

2. Partitioning for Tighter Approximation

We partition $\mathcal{Y} \setminus X$ into $p > 1$ distinct subsets so that

$$\|L_{X\cup\{i\}} - \overline{L}_X\|_F \gg \|L_{X\cup\{i\}} - \overline{L}_X^{(j)}\|_F,$$

where i is in the partition $j \in \{1, \ldots, p\}$ and $\overline{L}_X^{(j)}$ is the average of $L_{X\cup\{i\}}$'s in the partition j.

- To compute the marginal gains, we calculate the extra term $(*)$:

$$\log\det L_{X\cup\{i\}} - \log\det L_X$$
$$\approx \underbrace{\left\langle (\overline{L}_X^{(j)})^{-1}, L_{X\cup\{i\}} - \overline{L}_X^{(j)} \right\rangle}_{\text{can be computed by CG}} + \underbrace{\left(\log\det \overline{L}_X^{(j)} - \log\det L_X\right)}_{(*)}.$$

- $(*)$ can also be computed by CG under Schur complement.
- Thus, for each greedy update, we need to run CG of $O(d^2)$ complexity $2p$ times, where we choose $p = O(1)$.

Complexity and Error Bound

Complexity

The overall complexity becomes $O(d^3)$ because

- Each greedy update can be done in $O(d^2)$ time.
- The total number of greedy updates is at most d.

Complexity and Error Bound

Complexity

The overall complexity becomes $O(d^3)$ because

- Each greedy update can be done in $O(d^2)$ time.
- The total number of greedy updates is at most d.

Theorem

Suppose the smallest eigenvalue of L is larger than 1. Then, it holds that

$$\log\det L_X \geq (1-1/e) \max_{Z\subseteq\mathcal{Y},|Z|=|X|} \log\det L_Z - 2|X|\varepsilon,$$

where

$$\varepsilon = \max_{\substack{X\subseteq\mathcal{Y},i\in\mathcal{Y}\setminus X\\ j\in[p]}} \left| \log\frac{\det L_{X\cup\{i\}}}{\det \overline{L}_X^{(j)}} - \left\langle \left(\overline{L}_X^{(j)}\right)^{-1}, L_{X\cup\{i\}} - \overline{L}_X^{(j)} \right\rangle \right|.$$

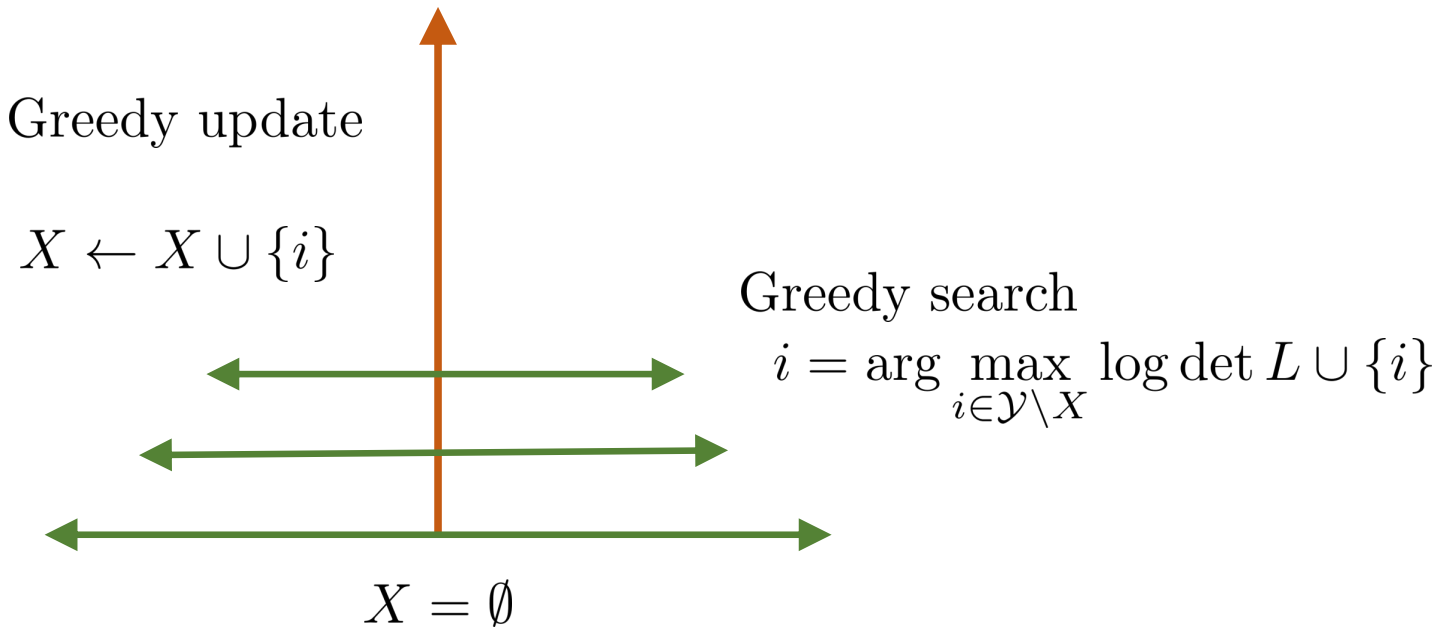
Outline: Faster Greedy DPP MAP Inference

Second Ideas: Further Speedup

To further speedup, we add k-batch subset (instead of single element)

$$X \leftarrow X \cup I \quad \text{for } |I| = k > 1,$$

so that the number of greedy updates can be reduced at most k times.

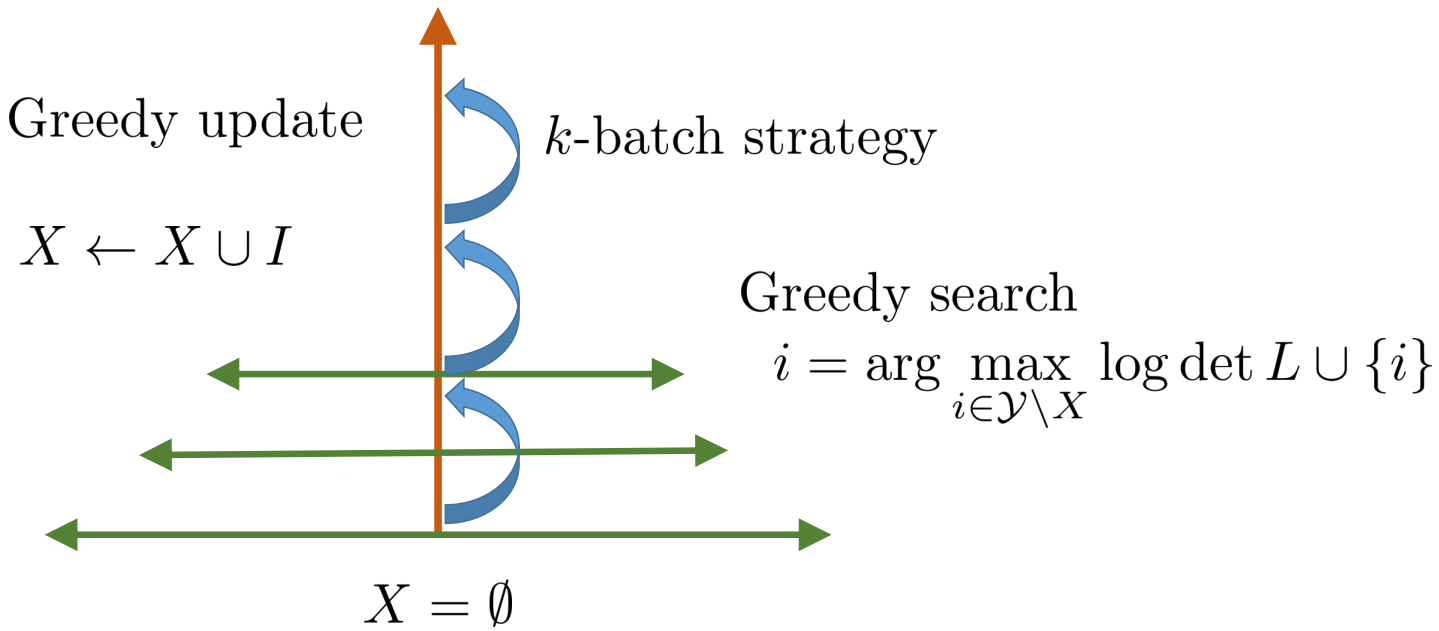


Second Ideas: Further Speedup

To further speedup, we add k-batch subset (instead of single element)

$$X \leftarrow X \cup I \quad \text{for } |I| = k > 1,$$

so that the number of greedy updates can be reduced at most k times.



Second Ideas: Further Speedup

To further speedup, we add k-batch subset (instead of single element)

$$X \leftarrow X \cup I \quad \text{for } |I| = k > 1,$$

so that the number of greedy updates can be reduced at most k times.

1. Stochastic Batch Sampling

- For the optimal k-batch, we have to investigate $\approx \binom{d}{k}$ subsets.
- This is expensive. We randomly sample k-batches and add the best of them to the current set.

Second Ideas: Further Speedup

2. Log-determinant Approximation Scheme (LDAS) for Comparison

- For k-batch strategy, one can compute the extra term $(*)$, i.e.,

$$\log\det \overline{L}_X^{(j)} - \log\det L_X,$$

by running CG k times.

Second Ideas: Further Speedup

2. Log-determinant Approximation Scheme (LDAS) for Comparison

- For k-batch strategy, one can compute the extra term $(*)$, i.e.,

$$\log\det \overline{L}_X^{(j)} - \log\det L_X,$$

  by running CG k times.

- Alternatively, one can estimate all $\log\det \overline{L}_X^{(j)}$'s by running a log-determinant approximation scheme (LDAS) [Han et al., 2015].

method	complexity	number of calls	objective
CG	$O(d^2)$	k	$\log\det \overline{L}_X^{(j)} - \log\det L_X$
LDAS	$O(d^2)$	1	$\log\det \overline{L}_X^{(j)}$

Second Ideas: Further Speedup

2. Log-determinant Approximation Scheme (LDAS) for Comparison

- LDAS approximates $\log\det \overline{L}_X^{(j)}$ using independent random vectors.

Log-determinant Approximation Scheme (LDAS) [Han et al., 2015]

Input : positive definite matrix A

$c_j \leftarrow$ coefficients of polynomial approximation of $\log x$ with degree n

for $i = 1$ **to** m **do**

$\mathbf{v}_i \leftarrow$ random vector uniformly distributed in $\{-1, +1\}^d$

$$\Gamma_i \leftarrow \sum_{j=0}^{m} c_j \mathbf{v}_i^\top A^j \mathbf{v}_i$$

end

Return $\frac{1}{m}\sum_{i=1}^{m} \Gamma_i$

Second Ideas: Further Speedup

2. Log-determinant Approximation Scheme (LDAS) for Comparison

- LDAS approximates $\log\det \overline{L}_X^{(j)}$ using independent random vectors.
- We suggest to run LDAS using the same random vectors for estimating all $\log\det \overline{L}_X^{(j)}$'s.

Log-determinant Approximation Scheme (LDAS) [Han et al., 2015]

Input : positive definite matrix A

$c_j \leftarrow$ coefficients of polynomial approximation of $\log x$ with degree n

for $i = 1$ **to** m **do**

$\mathbf{v}_i \leftarrow$ random vector uniformly distributed in $\{-1, +1\}^d$

$$\Gamma_i \leftarrow \sum_{j=0}^{n} c_j \mathbf{v}_i^\top A^j \mathbf{v}_i$$

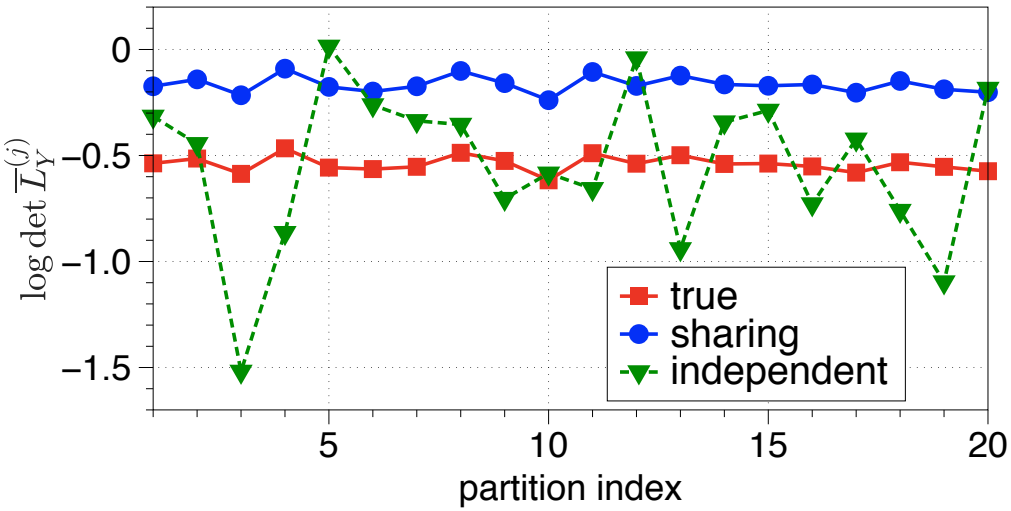
end

Return $\frac{1}{m}\sum_{i=1}^{m} \Gamma_i$

Second Ideas: Further Speedup

2. Log-determinant Approximation Scheme (LDAS) for Comparison

- LDAS approximates $\log\det \overline{L}_X^{(j)}$ using independent random vectors.
- We suggest to run LDAS using the same random vectors for estimating all $\log\det \overline{L}_X^{(j)}$'s.
- Running LDAS's under sharing random vectors is better for comparing $\log\det \overline{L}_X^{(j)}$, i.e., marginal gains.



Second Ideas: Further Speedup

2. Log-determinant Approximation Scheme (LDAS) for Comparison

We provide the variance of LDAS under sharing random vectors as

$$\mathbf{Var}\left[\mathrm{LDAS}\left(\overline{L}_X^{(j)}\right) - \mathrm{LDAS}\left(\overline{L}_X^{(j')}\right)\right] = O\left(\|\overline{L}_X^{(j)} - \overline{L}_X^{(j')}\|_F^2\right).$$

- On the other hand, the variance under independent random vectors is

$$\mathbf{Var}\left[\mathrm{LDAS}\left(\overline{L}_X^{(j)}\right) - \mathrm{LDAS}\left(\overline{L}_X^{(j')}\right)\right] = O\left(\|\overline{L}_X^{(j)}\|_F^2 + \|\overline{L}_X^{(j')}\|_F^2\right).$$

- In our case, $\|\overline{L}_X^{(j)}\|_F^2 + \|\overline{L}_X^{(j')}\|_F^2$ is much larger than $\|\overline{L}_X^{(j)} - \overline{L}_X^{(j')}\|_F^2$.

Outline: Faster Greedy DPP MAP Inference

Experimental Results

Setting

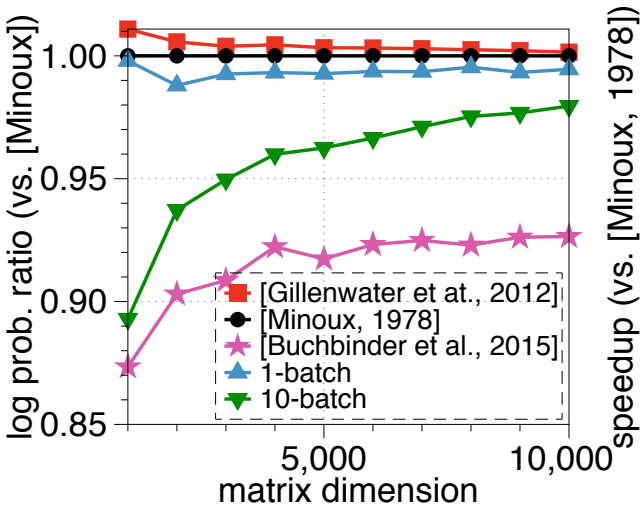
- Accuracy is measured by log-probability ratio of a respective algorithm to the standard (but, accelerated) greedy algorithm [Minoux, 1978].
- Two versions of our algorithms: 1-batch and 10-batch with 50 batch samples.
- We compare our algorithms with other approximations.

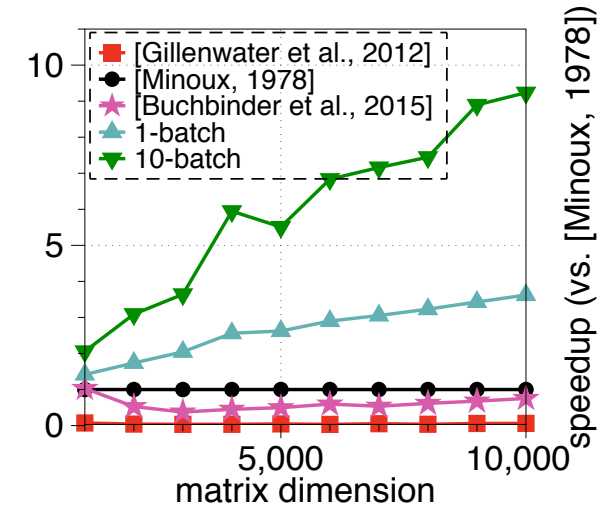
algorithm	complexity	remarks
[Minoux, 1978]	$O(d^5)$	accelerated version of a naïve greedy algorithm
[Buchbinder et al., 2015]	$O(d^4)$	symmetric greedy algorithm
[Gillenwater et al., 2012]	$O(d^4)$	multilinear softmax extension

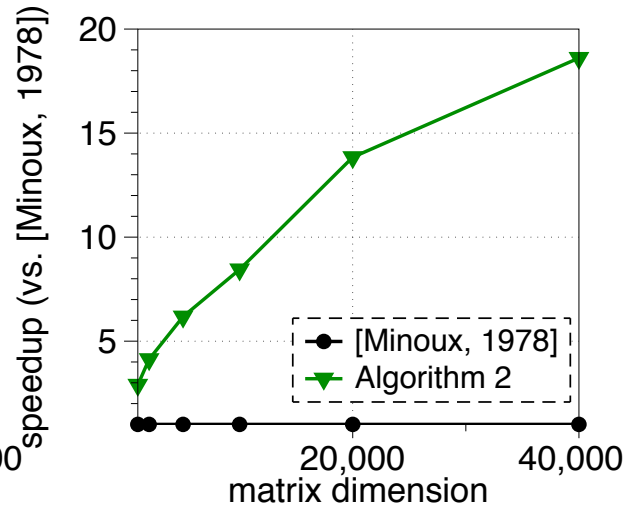
Experimental Results

Synthetic Dataset

- Accuracy is measured by log-probability ratio of a respective algorithm to the standard (but, accelerated) greedy algorithm [Minoux, 1978].
- Two versions of our algorithms: 1-batch and 10-batch with 50 batch samples.
- 1-batch achieves 0.03% and 10-batch achieves 0.3% loss on accuracy.
- 10-batch runs up-to 18 times faster than [Minoux, 1978].



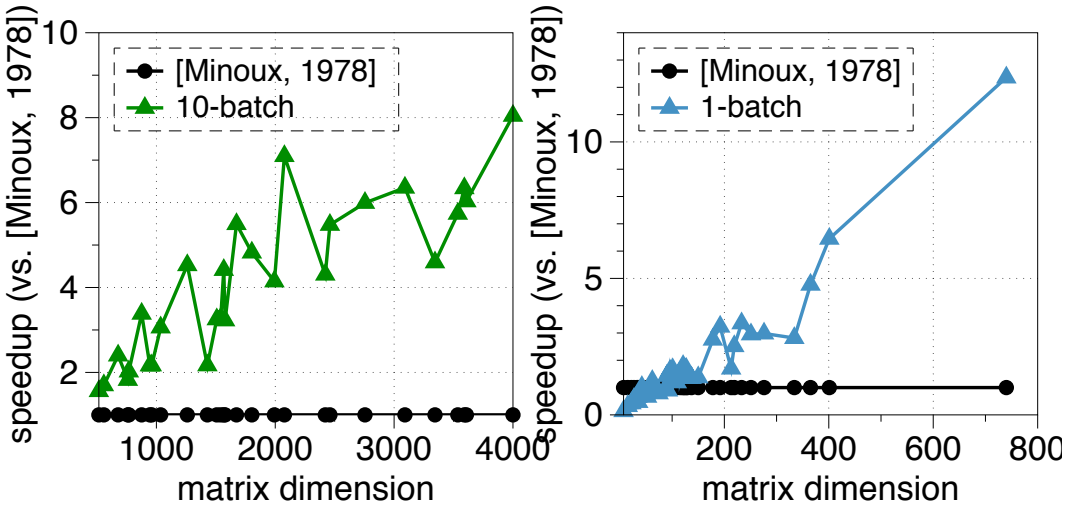




Experimental Results

Real Dataset: Text/Video Summarization

- For both text and video summarizations, the proposed algorithms run $8 \sim 10$ times faster for large instances.
- Our video summaries often have higher F-score than the standard greedy algorithm.



Conclusion

1. We develop fast DPP MAP inference using first-order Taylor expansion of log-determinant and partitioning for tighter approximation.

2. For further speedup, we use a k-batch strategy with stochastic sampling and a log-determinant approximation scheme under sharing random vectors among their runs.

3. Our proposed algorithm runs up-to 18 times faster than the standard (accelerated) greedy algorithm for matrix dimension $d = 40,000$, where the speedup increases as d grows up.

Conclusion

1. We develop fast DPP MAP inference using first-order Taylor expansion of log-determinant and partitioning for tighter approximation.

2. For further speedup, we use a k-batch strategy with stochastic sampling and a log-determinant approximation scheme under sharing random vectors among their runs.

3. Our proposed algorithm runs up-to 18 times faster than the standard (accelerated) greedy algorithm for matrix dimension $d = 40,000$, where the speedup increases as d grows up.

Thank you for your attention !

References I

Buchbinder, N., Feldman, M., Seffi, J., and Schwartz, R. (2015).

A tight linear time (1/2)-approximation for unconstrained submodular maximization.
SIAM Journal on Computing, 44(5):1384–1402.

Gillenwater, J., Kulesza, A., and Taskar, B. (2012).

Near-optimal map inference for determinantal point processes.
In *Advances in Neural Information Processing Systems*, pages 2735–2743.

Han, I., Malioutov, D., and Shin, J. (2015).

Large-scale log-determinant computation through stochastic chebyshev expansions.
In *ICML*, pages 908–917.

Ko, C.-W., Lee, J., and Queyranne, M. (1995).

An exact algorithm for maximum entropy sampling.
Operations Research, 43(4):684–691.

Minoux, M. (1978).

Accelerated greedy algorithms for maximizing submodular set functions.
In *Optimization Techniques*, pages 234–243. Springer.