An Introduction to Determinantal Point Processes

José Miguel Hernández–Lobato[1] and Hong Ge[1]

February 20, 2014

[1]Department of Engineering, Cambridge University, UK.

Useful References

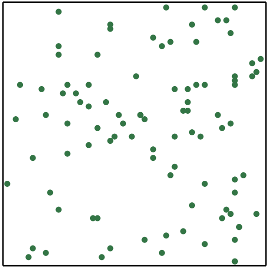
Mainly Ben Taskar and Alex Kulesza's work:

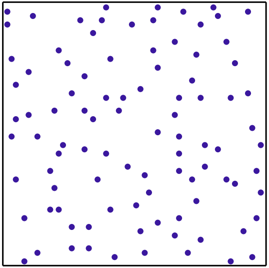
- A. Kulesza and B. Taskar, **Determinantal Point Processes for Machine Learning**, Foundations and Trends in Machine Learning: Vol. 5, No 2-3, 2012. (Available in the arXiv).
- Near-Optimal MAP Inference for Determinantal Point Processes, J. Gillenwater, A. Kulesza, and B. Taskar. Neural Information Processing Systems (NIPS), Lake Tahoe, Nevada, December 2012.
- Learning Determinantal Point Processes, A. Kulesza, and B. Taskar. Conference on Uncertainty in Artificial Intelligence (UAI), Barcelona, Spain, July 2011.
- k-DPPs: Fixed-Size Determinantal Point Processes, A. Kulesza, and B. Taskar. International Conference on Machine Learning (ICML), Bellevue, WA, June 2011.
- ...

Informal Description

A **point process** is a distribution over finite subsets of a fixed ground set $\mathcal{Y}$. We will assume that $\mathcal{Y}$ is finite, that is, $|\mathcal{Y}| = N$.

Determinantal point processes (DPPs) are probabilistic models with global, negative correlations with respect to a similarity measure: DPPs enforce diversity.





DPPs offer computationally efficient algorithms for sampling, marginalization, conditioning and other inference tasks.

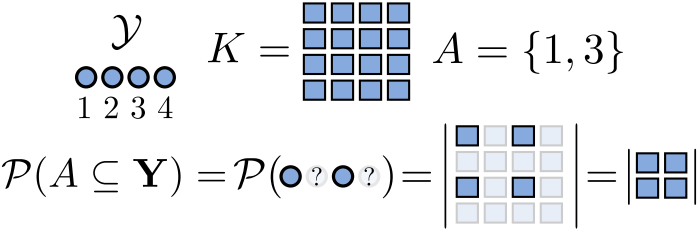
Formal Definition

A point process $\mathcal{P}$ on $\mathcal{Y}$ is a probability distribution on $2^{\mathcal{Y}}$.

$\mathcal{P}$ is a DPP if, when $\mathbf{Y} \sim \mathcal{P}$, then for every $A \subseteq \mathcal{Y}$,

$$\mathcal{P}(A \subseteq \mathbf{Y}) = \det(K_A)\,,$$

where K is a similarity matrix index by the elements of $\mathcal{Y}$ and $K_A \equiv [K_{i,j}]_{i,j \in A}$ restricts K to those entries in A.



We define $\det(K_\emptyset) = 1$. K must satisfy $0 \preceq K \preceq I$.

Negative Correlations in DPPs

If $A = \{i, j\}$ is a two-element set, then

$$\begin{aligned}
\mathcal{P}(A \subseteq \mathcal{Y}) &= \begin{vmatrix} K_{ii} & K_{ij} \\ K_{ji} & K_{jj} \end{vmatrix} \\
&= K_{ii}K_{jj} - K_{ij}K_{ji} \\
&= \mathcal{P}(i \in \mathbf{Y})\mathcal{P}(j \in \mathbf{Y}) - K_{ij}^2 \,.
\end{aligned}$$

- Off-diagonal entries determine the negative correlations.
- If $K_{ij} = \sqrt{K_{ii}K_{jj}}$, i and j never appear together in $\mathbf{Y}$.
- When K is diagonal the elements in $\mathbf{Y}$ are independent.

Correlations are always negative in DPPs!

Many theoretical and physical processes are determinantal.

Conditioning in DPPs

$$
\begin{aligned}
\mathcal{P}(B \subseteq \mathbf{Y} | A \subseteq \mathbf{Y}) &= \frac{\mathcal{P}(A \cup B \subseteq \mathbf{Y})}{\mathcal{P}(A \subseteq \mathbf{Y})} \\
&= \frac{\det(K_{A\cup B})}{\det(K_A)} = \det(K_B - K_{BA}K_A^{-1}K_{AB}) \\
&= \det([K - K_{\star A}K_A^{-1}K_{A\star}]_B)\,.
\end{aligned}
$$

$$
K_{A\cup B} = \begin{bmatrix} K_A & K_{AB} \\ K_{BA} & K_B \end{bmatrix}
$$

Schur Complement of K_A.

$\det(K_{A\cup B}) =$
$\det(K_A)\det(K_B - K_{BA}K_A^{-1}K_{AB})\,.$

DPPs are closed under conditioning!

L-ensembles

For modeling data, it is useful to work with *L-ensembles*.

An L-ensemble defines a DPP through a symmetric matrix L:

$$\mathcal{P}(\mathbf{Y} = Y) \propto \det(L_Y)\,.$$

L-ensembles give the atomic probabilities of inclusion.

The normalization constant is $\sum_{Y \subseteq \mathcal{Y}} \det(L_Y) = \det(L + I)$.

L has to satisfy fewer constraints: $0 \preceq L$.

An L-ensemble is a DPP with marginal kernel K given by

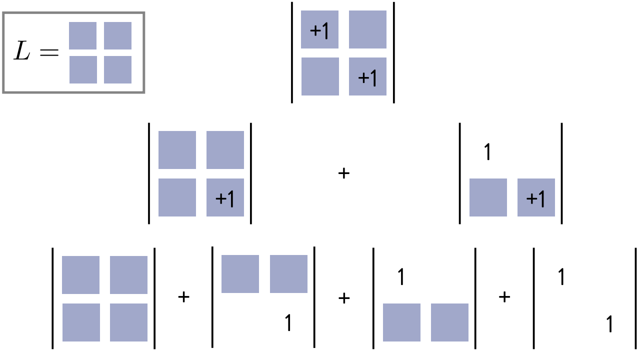
$$K = L(L + I)^{-1} = I - (L + I)^{-1}\,.$$

Not all DPPs are L-ensembles!

Normalization Constant

The normalization constant of an L-ensemble is $\det(L+I)$.

This follows from the multilinearity of determinants.



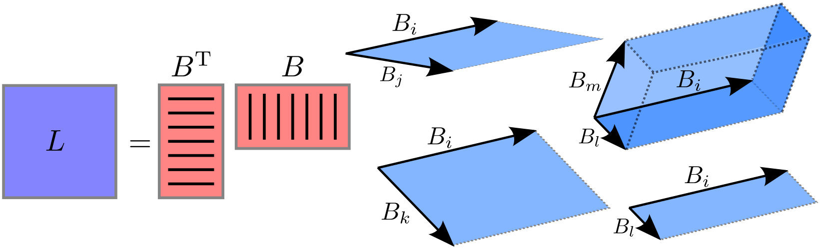
$$\propto \quad \mathcal{P}(\{1,2\}) \qquad \mathcal{P}(\{1\}) \qquad \mathcal{P}(\{2\}) \qquad \mathcal{P}(\emptyset)$$

Geometric Interpretation

When L is a gram matrix, that is, $L = B^{\mathrm{T}}B$, then

$$\det(L_Y) = \mathrm{Vol}^2(\{B_i\}_{i \in Y}),$$

where B_i is the i-th column of B, that is, $L_{ij} = B_i^{\mathrm{T}} B_j$.



Elementary DPPs

A DPPs is elementary if every eigenvalue of K is in $\{0, 1\}$.

$\mathcal{P}^V$ denotes the elementary DPP with kernel $K^V = \sum_{\mathbf{v} \in V} \mathbf{v}\mathbf{v}^{\mathrm{T}}$, where V is a set of orthonormal vectors.

What is $|\mathbf{Y}|$ when we sample from P^V?

$E[|\mathbf{Y}|] = \mathrm{trace}(K^V) = \sum_{\mathbf{v} \in V} ||\mathbf{v}||^2 = |V|$.

Since $\mathrm{rank}(K^V) = |V|$ we have that $p(|\mathbf{Y}| > |V|) = 0$.

Therefore, $\boxed{p(|\mathbf{Y}| = |V|) = 1}$.

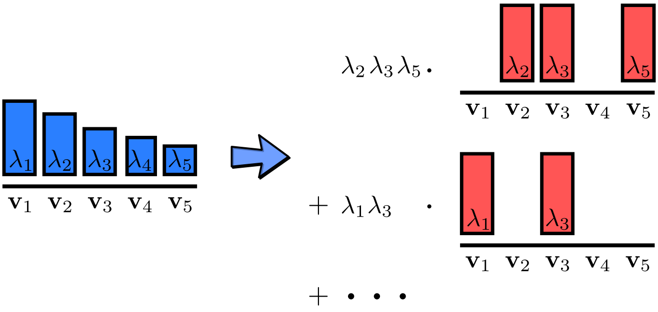
Sampling from $\mathcal{P}^V$ can be done with cost $\mathcal{O}(|V|^3 N)$.

DPPs as Mixtures of Elementary DPPs

Lemma: If $\mathcal{P}_L$ is a DPP with eigendecomposition of L given by $L = \sum_{n=1}^{N} \lambda_n \mathbf{v}_n \mathbf{v}_n^{\mathrm{T}}$. Then $\mathcal{P}_L$ is a mixture of elementary DPPs:

$$\mathcal{P}_L = \frac{1}{\det(L+I)} \sum_{J \subseteq \{1,2,\ldots,N\}} \mathcal{P}^{V_J} \prod_{n \in J} \lambda_n \,,$$

where $V_J = \{\mathbf{v}_n : n \in J\}$.



Sampling Algorithm

$$\boxed{\mathcal{P}_L = \frac{1}{\det(L+I)} \sum_{J \subseteq \{1,2,\ldots,N\}} \mathcal{P}^{V_J} \prod_{n \in J} \lambda_n \,, \quad V_J = \{\mathbf{v}_n : n \in J\}\,.}$$

The mixture representation of DPPs suggests a sampling algorithm based on the following steps:

- **Step 1**: Select an elementary DPP $\mathcal{P}^{V_J}$ with probability proportional to its mixture weight $[\det(L+I)]^{-1} \prod_{n \in J} \lambda_n$.
- **Step 2**: Draw a sample from the selected $\mathcal{P}^{V_J}$.

[Hough et al. 2006]

Step 1 of the Sampling Algorithm

$$\boxed{\mathcal{P}_L = \frac{1}{\det(L+I)} \sum_{J \subseteq \{1,2,\ldots,N\}} \mathcal{P}^{V_J} \prod_{n \in J} \lambda_n \,, \quad V_J = \{\mathbf{v}_n : n \in J\} \,.}$$

Recall that $[\det(L+I)]^{-1} = \prod_{n=1}^{N} (\lambda_n + 1)^{-1}$. The mixture weight for $\mathcal{P}_{V_J}$ is $[\prod_{n \in J} \lambda_n / (\lambda_n + 1)][\prod_{n \notin J} (1 - \lambda_n / (\lambda_n + 1))]$.

Step 1 of the Sampling Algorithm:

Input: eigendecomposition $\{(\mathbf{v}_n, \lambda_n)\}_{n=1}^{N}$ of L.
$J \leftarrow \emptyset$
for $n = 1, 2, \ldots, N$ **do**
    $J \leftarrow J \cup \{n\}$ with probability $\lambda_n / (1 + \lambda_n)$.
end for
$V_J \leftarrow \{\mathbf{v}_n\}_{n \in J}$

Step 2 of the Sampling Algorithm

$$\boxed{\mathcal{P}_L = \frac{1}{\det(L+I)} \sum_{J \subseteq \{1,2,\dots,N\}} \mathcal{P}^{V_J} \prod_{n \in J} \lambda_n \,, \quad V_J = \{\mathbf{v}_n : n \in J\}\,.}$$

We sample from $\mathcal{P}^{V_J}$, whose kernel is $K^{V_J} = \sum_{\mathbf{v} \in V_J} \mathbf{v}\mathbf{v}^{\mathrm{T}}$.

Step 2 of the Sampling Algorithm:

Input: set $V_J = \{\mathbf{v}_n : n \in J\}$ of orthonormal vectors.
$Y \leftarrow \emptyset$
while $|Y| < |V_J|$ **do**
    Choose $j \in \{1, \dots, N\}$ with prob. $\propto K_{jj}^{V_J} = \sum_{\mathbf{v} \in V_J} (\mathbf{v}^{\mathrm{T}} \mathbf{e}_j)^2$.
    $Y \leftarrow Y \cup \{j\}$.
    Update K^{V_J} to condition on $j \in Y$. (cost $\mathcal{O}(N|V_J|^2)$).
end while

The total cost is $\mathcal{O}(N|Y|^3)$.

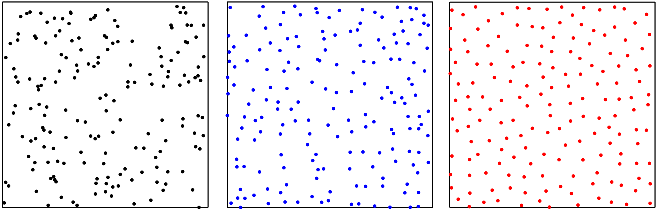
Sampling Example and Some Consequences

Step 1 determines size and likely content of $\mathbf{Y} \sim \mathcal{P}_L$:

- $|\mathbf{Y}|$ is distributed as a sum of independent Bernoulli variables, each one with success prob. $\lambda_n/(\lambda_n + 1)$.
- The likely content of $\mathbf{Y}$ is determined by the chosen elementary DPP.

Size and content are intertwined in DPPs!
For example, no DPP can uniformly sample sets of size k.

Finding the Mode



Finding the set $Y \subseteq \mathcal{Y}$ that maximizes $\mathcal{P}_L(Y)$ is NP-hard.

Submodularity: $\mathcal{P}_L$ is log-submodular, that is,

$$\log \mathcal{P}_L(Y \cup \{i\}) - \log \mathcal{P}_L(Y) \geq \log \mathcal{P}_L(Y' \cup \{i\}) - \log \mathcal{P}_L(Y'),$$

whenever $Y \subseteq Y' \subseteq \mathcal{Y} - \{i\}$.

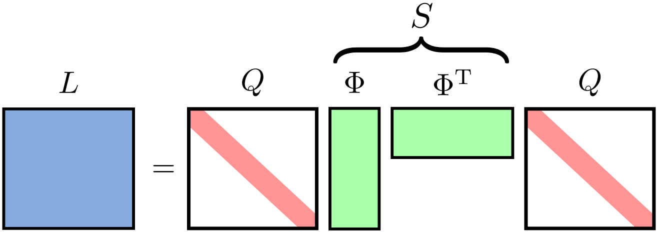
Many results exists for approximately maximizing monotone submodular functions. However, $\mathcal{P}_L$ is highly non-monotone! In practice, this is not a problem [Kulesza et al., 2012].

DPP Decomposition: Quality vs Diversity I

We can take the notation $L = B^{\mathrm{T}}B$ one step further.

Each column B_i satisfies $B_i = q_i \phi_i$, where

- $q_i \in \mathbb{R}^+$ is a quality term.
- $\phi_i \in \mathbb{R}^D$, $||\phi_i|| = 1$ is a vector of diversity features.



We now have $\mathcal{P}_L(Y) \propto \left[\prod_{i \in Y} q_i^2\right] \det(S_Y)$.

The first factor increases with the quality of the items in Y.

The second factor increases with the diversity of the items in Y.

DPP Decomposition: Quality vs Diversity II

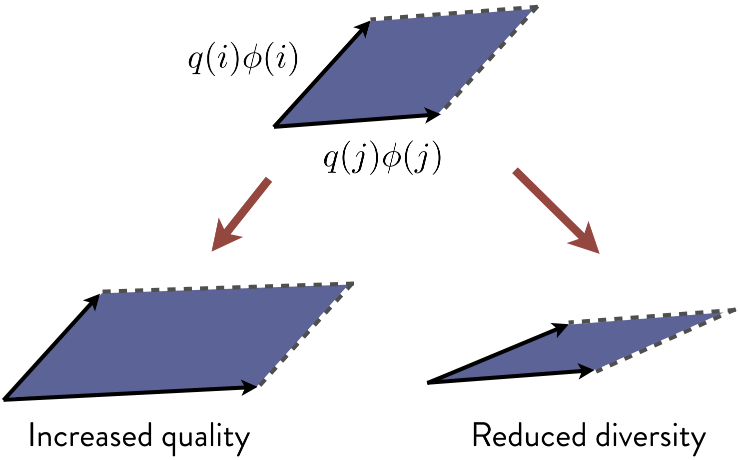
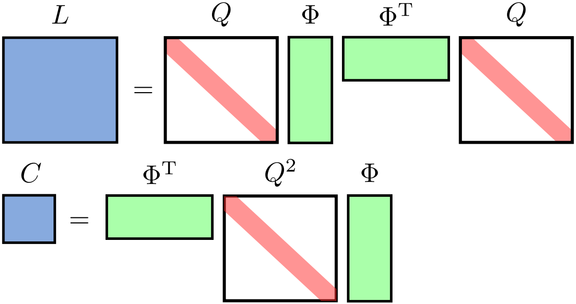


Figure source: [Kulesza and Taskar, 2012].

Dual Representation I

Most algorithms require manipulating L through inversion, eigendecomposition, etc...

When N is very large, directly working with the $N \times N$ matrix L is not efficient.



Let B be the $D \times N$ matrix with $B_i = q_i \phi_i$ so that $L = B^{\mathrm{T}} B$. Instead, we work with the $D \times D$ matrix $C = BB^{\mathrm{T}}$.

Dual Representation II

- C and L have the same (non-zero) eigenvalues.
- Their eigenvectors are linearly related.
- Working with C scales as a function of $D \ll N$.

Proposition:

$$C = BB^{\mathrm{T}} = \sum_{n=1}^{D} \lambda_n \hat{\mathbf{v}}_n \hat{\mathbf{v}}_n^{\mathrm{T}}$$

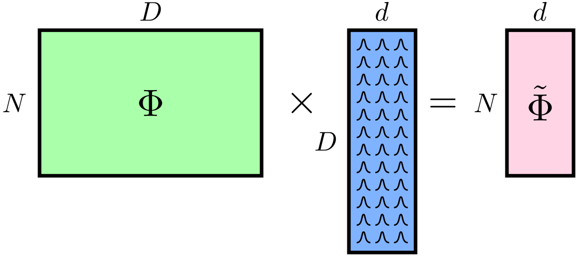
is an eigendecomposition of C if and only if

$$L = B^{\mathrm{T}} B = \sum_{n=1}^{D} \lambda_n \left[\frac{1}{\sqrt{\lambda_n}} B^{\mathrm{T}} \hat{\mathbf{v}}_n \right] \left[\frac{1}{\sqrt{\lambda_n}} B^{\mathrm{T}} \hat{\mathbf{v}}_n \right]^{\mathrm{T}}$$

is an eigendecomposition of L.

Reducing the Dimensionality of the Diversity Features

What if D, the dimension of the features in Φ, is very large?
Solution: project the rows of Φ to a space of low dimension d.



Random projections are known to approximately preserve distances [Johnson and Lindenstrauss, 1984].

Random Projections and Volumes

Random projections also approximately preserve volumes [Magen and Zouzias, 2008].

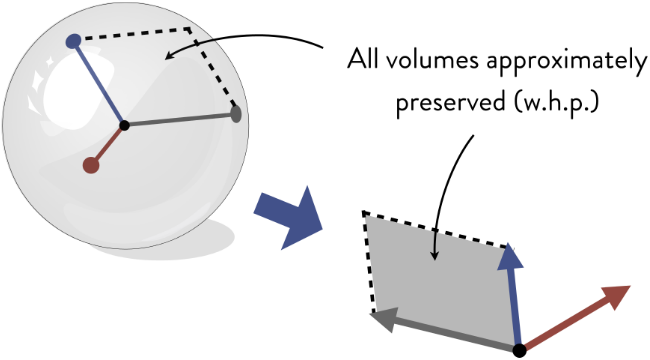


Figure source: [Kulesza and Taskar, 2012].

Theoretical Guarantees of Random Projections

Theorem: when the dimensionality d of the projected vectors satisfies $d = \mathcal{O}(\epsilon^{-2} \log N)$, with high probability we have

$$||\mathcal{P} - \tilde{\mathcal{P}}||_1 \leq \mathcal{O}(\epsilon) \,.$$

- d is logarithmic in N.
- d does not depend on D (the original dimension).
- DPPs can scale to large N and large D by combining random projections with the dual representation of L.

	Small N	Large N
Small D	Standard DPP or dual DPP	Dual DPP
Large D	Standard DPP	Random projection dual DPP

Conditional DPPs

In many problems, using a fixed ground set $\mathcal{Y}$ is inadequate. For example, in document summarization problems.

Solution: use a $\mathcal{Y}(X)$ that depends on an input variable X.

Definition: A conditional DPP $\mathcal{P}(\mathbf{Y} = Y|X)$ is a distribution over each subset $Y \subseteq \mathcal{Y}(X)$ such that

$$\mathcal{P}(\mathbf{Y} = Y|X) \propto \det(L_Y(X)),$$

where $L(X)$ is a positive semidefinite kernel that depends on X.

Using the quality diversity decomposition we write L as:

$$L_{ij}(X) = q_i(X)\phi_i(X)^{\mathrm{T}}\phi_j(X)q_j(X).$$

Supervised learning can then be used to identify the latent functions connecting X with each q_i and ϕ_i.

k-DPP

What if we need exactly k diverse items?

k-DPP

What if we need exactly k diverse items?

Simple idea: condition DPP on target size k.

$$\mathcal{P}^k(Y) = \frac{\det(L_Y)}{\sum_{|Y'|=k} \det(L_{Y'})}$$

k-DPP Inference - Normalisation

Recall that the k-th elementary symmetric polynomial on $\lambda_1, \lambda_2, \ldots, \lambda_N$ is given by

$$e_k(\lambda_1, \lambda_2, \ldots, \lambda) = \sum_{\substack{J \subseteq [N] \\ |J|=k}} \prod_{n \in J} \lambda_n$$

E.g. $e_2(\lambda_1, \lambda_2, \lambda_3) = \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3$, then the *normalization constant* is given by

Proposition 5.1. *The normalization constant for a k-DPP is*

$$Z_k = \sum_{|Y'|=k} \det(L_{Y'}) = e_k(\lambda_1, \lambda_2, \ldots, \lambda_N)\,,$$

where $\lambda_1, \lambda_2, \ldots, \lambda_N$ *are the eigenvalues of* L*.*

k-DPP Inference - Conditioning

Suppose we want to condition a k-DPP on the inclusion of a particular set A. For $|A| + |B| = k$ we have

$$\begin{aligned} P_L^k(\mathbf{Y} = A \cup B | A \subseteq \mathbf{Y}) &\propto P_L^k(Y = A \cup B) \\ &\propto P_L^k(Y = A \cup B) \\ &\propto P_L(Y = A \cup B | A \subseteq Y) \\ &\propto \det(L_B^A) \end{aligned}$$

Thus the conditional k-DPP is a $k - |A|$-DPP whose kernel is the same as that of the associated conditional DPP:

$$L^A = K - K_{*A} K_A^{-1} K_{A*}$$

We can condition on excluding A in the same manner.

k-DPP inference - sampling I

Recall that *elementary* DPPs are DPPs whose eigenvalues are binary, i.e., $\lambda_n \in \{0, 1\}$.

Furthermore, each standard DPP can be viewed as a mixture of *elementary* DPPs

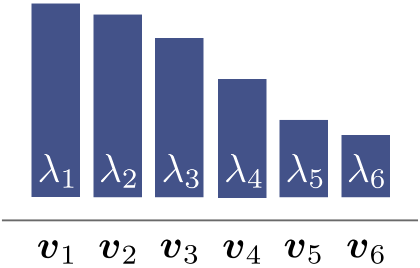
$$\mathcal{P} \propto \sum_{J \subseteq [N]} \mathcal{P}^J \prod_{n \in J} \lambda_n$$

Similarity, a k-DPP can also be represented as

$$\begin{aligned} \mathcal{P} \propto & \sum_{\substack{J \subseteq [N] \\ |J|=k}} \mathcal{P}^J \prod_{n \in J} \lambda_n \\ & \sum_{J \subseteq [N]} \mathcal{P}^J \mathbb{I}(|J| = k) \prod_{n \in J} \lambda_n \end{aligned} \tag{1}$$

k-DPP Inference - Sampling II

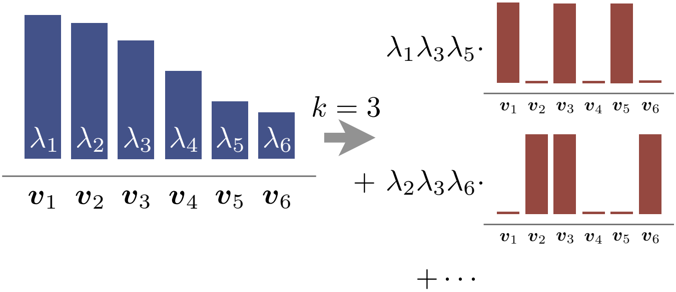
$$\mathcal{P} \propto \sum_{\substack{J\subseteq\{1,\ldots,N\}\\ |J|=k}} \mathcal{P}^J \prod_{n\in J} \lambda_n$$



[Kulesza and Taskar (2012)]

k-DPP Inference - Sampling II

$$\mathcal{P} \propto \sum_{\substack{J \subseteq \{1,\ldots,N\} \\ |J| = k}} \mathcal{P}^J \prod_{n \in J} \lambda_n$$



[Kulesza and Taskar (2012)]

k-DPP Inference - Sampling III

Can use similar sampling procedure for standard DPP.
Need new PHASE ONE to pick $|J| = k$.
SOLUTION: recursion on elementary symetric polynomials:

$$e_k^N = \sum_{\substack{J \subseteq [N] \\ |J|=k}} \prod_{n \in J} \lambda_n$$

PHASE TWO is unchanged.

k-DPP Inference - Sampling IV

Algorithm 1 Sampling from a DPP

Input: eigenvector/value pairs $\{(\boldsymbol{v}_n, \lambda_n)\}$
$J \leftarrow \emptyset$
for $n = 1, \dots, N$ **do**
  $J \leftarrow J \cup \{n\}$ with prob. $\frac{\lambda_n}{\lambda_n + 1}$
end for
$V \leftarrow \{\boldsymbol{v}_n\}_{n \in J}$
$Y \leftarrow \emptyset$
while $|V| > 0$ **do**
  Select y_i from $\mathcal{Y}$ with $\Pr(y_i) = \frac{1}{|V|} \sum_{\boldsymbol{v} \in V} (\boldsymbol{v}^\top \boldsymbol{e}_i)^2$
  $Y \leftarrow Y \cup y_i$
  $V \leftarrow V_\perp$, an orthonormal basis for the subspace of V orthogonal to $\boldsymbol{e}_i$
end while
Output: Y

[Kulesza and Taskar (2012)]

k-DPP Inference - Sampling IV

Algorithm 2 Sampling from a k-DPP

Input: eigenvector/value pairs $\{(\boldsymbol{v}_n, \lambda_n)\}$, size k
$J \leftarrow \emptyset$
for $n = N, \dots, 1$ **do**
  if $u \sim U[0,1] < \lambda_n \frac{e_{k-1}^{n-1}}{e_k^n}$ **then**
    $J \leftarrow J \cup \{n\}$
    $k \leftarrow k - 1$
    if $k = 0$ **then**
      break
    end if
  end if
end for
Proceed with the second loop of Algorithm 1
Output: Y

[Kulesza and Taskar (2012)]

Image Search

$\sim$ 2k images from Google Image search.

3 categories: cars, cities, dog breeds.

Ground truth created via Amazon Mechanical Turk ($0.01 USD for each instance labeled).

Image Search - Data

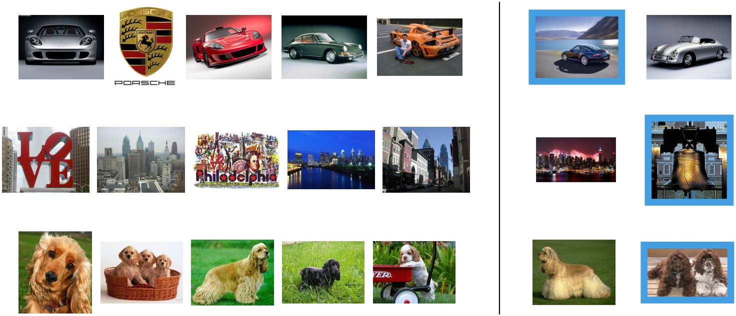


Figure 15: Sample labeling instances from each search category. The five images on the left form the partial result set, and the two candidates are shown on the right. The candidate receiving the majority of annotator votes has a blue border.

[Kulesza and Taskar (2012)]

Image Search - Data

CARS	CITIES	DOGS
chrysler	baltimore	beagle
ford	barcelona	bernese
honda	london	blue heeler
mercedes	los angeles	cocker spaniel
mitsubishi	miami	collie
nissan	new york city	great dane
porsche	paris	labrador
toyota	philadelphia	pomeranian
	san francisco	poodle
	shanghai	pug
	tokyo	schnauzer
	toronto	shih tzu

Table 6: Queries used for data collection.

[Kulesza and Taskar (2012)]

Image Search - Learning

Learn mixture of 55 “expert” k-DPPs.

$$\mathcal{P}_\theta^k = \sum_{l=1}^{55} \theta_l \mathcal{P}_{L_l}^k, \text{ s.t.} \sum_{l=0}^{55} \theta_l = 1$$

Similarity kernel L:

$$L_{ij}^f = \mathbf{f}(i)^T \mathbf{f}(j), \text{ s.t.} ||\mathbf{f}(i)||^2 = 1$$

And feature functions: SIFT, Color histogram, GIST, Center only / all pairs

Image Search - results

Table 2. Percentage of real-world image search examples judged the same way as the majority of human annotators. Bold results are significantly higher than others in the same row with 99% confidence.

Cat.	Best MMR	Best k-DPP	Mixture MMR	Mixture k-DPP
cars	55.95	57.98	59.59	**64.58**
cities	56.48	56.31	60.99	61.29
dogs	56.23	57.70	57.39	**59.84**

[Kulesza and Taskar (2012)]

Summary

DPPs...

- introduce global negative correlations in their samples.
- produce diverse sets according to a specific similarity measure.
- have efficient algorithms for sampling, marginalization and conditioning.
- can be useful in several machine learning applications such as image search.

Appendix: Image Search - Methods

Best k-DPPs

$$\text{k-DPP}_t = \mathbf{argmax}_{i \in C_t} \mathcal{P}_L^6(Y_t \cup \{i\})$$

Mixture of k-DPPs

$$\text{k-DPPmix}_t = \mathbf{argmax}_{i \in C_t} \sum_{l=1}^{55} \theta_l \mathcal{P}_L^6(Y_t \cup \{i\})$$

Best MMR

$$\text{MMR}_t = \mathbf{argmin}_{i \in C_t} [\max_{j \in Y_t} L_{ij}]$$

Mixture MMR

$$\text{MMRmix}_t = \mathbf{argmin}_{i \in C_t} \sum_{l=1}^{55} \theta_l [\max_{j \in Y_t} L_{ij}^l]$$