Diverse Personalization with Determinantal Point Process Eigenmixtures

YAHOO! LABS

James Foulds and Dilan Görür

University of California, Irvine and Yahoo Labs

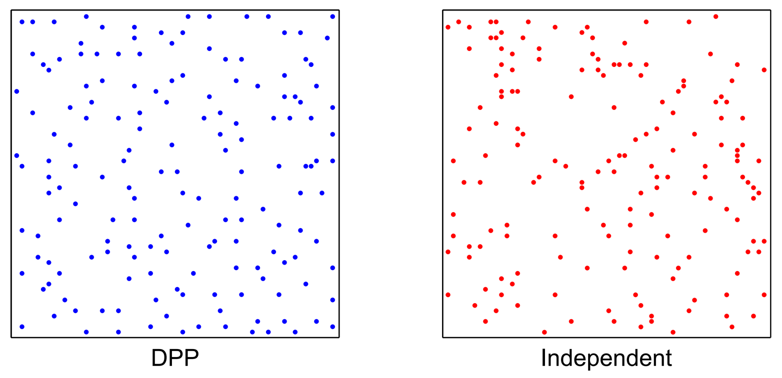
UCIRVINE

Abstract

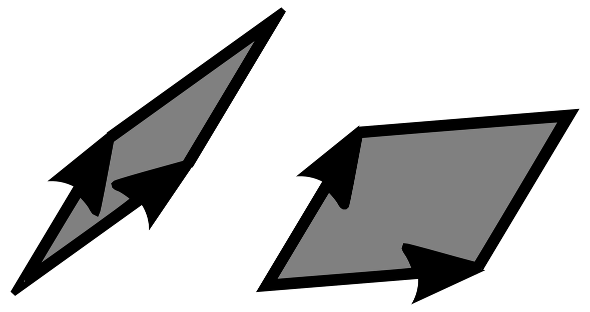
- Recommender systems must balance the estimated level of user **interest** of the recommended items with the **diversity** of the content.
- **Determinantal point processes** (DPPs) are probabilistic models for selecting **diverse**, **high-quality sets**.
- In a personalization context we would typically like to have more control over the recommendations than DPPs afford. To address this, we introduce several approaches for **blending the properties of multiple DPPs**.
- The final proposed approach, the **DPP eigenmixture**, exploits the eigenstructure of the DPP kernel matrices in order to encapsulate the most important properties of several DPPs.
- We demonstrate the utility of the proposed methods on several recommendation tasks.

Determinantal Point Processes

- DPPs are distributions over subsets S of a set $\mathcal{Y}$ of items, which prefer **diverse** sets.



- We focus on the class of DPPs most relevant for machine learning, called **L-ensembles**. These models represent the items with a $k \times N$ feature matrix B.
- Each column B_a of B is a feature vector representing item a
- The DPP selects sets with probability proportional to the **squared volume** of the parallelotope spanned by the feature vectors of the items.



- The DPP takes as input the **Gram matrix** of B, $\mathrm{L} = \mathrm{B}^\intercal\mathrm{B}$. Here, $L_{ab} = \mathrm{B}_a^\intercal\mathrm{B}_b$ corresponds to a similarity score for elements a and b. The probability function for the DPP can be written as

$$\mathcal{P}(S) \propto \det(\mathrm{L}_S)\ .$$

Personalization Using the DPP

- We use item features A to encode diversity information (angles), and collaborative filtering recommendation scores Y_{ia} for user i and item a to encode a personalized notion of quality (length), e.g. $\mathrm{Y} = \mathrm{U}^\intercal\mathrm{A}$, where U contains user features. This gives us an L-ensemble kernel

$$L_{ab}^{(i)} = \mathrm{B}_a^\intercal\mathrm{B}_b = \sqrt{f(Y_{ia})}\mathrm{A}_a^\intercal\mathrm{A}_b\sqrt{f(Y_{ib})}\ .$$

- Sets of items are recommended by drawing from the resulting DPP,

$$\mathcal{P}(S) \propto \det(\mathrm{A}_{:,S}{}^\intercal\mathrm{A}_{:,S}) \prod_{a\in S} f(Y_{ia})\ .$$

Combining Multiple DPPs for Personalization

- For personalization tasks, we would like more control over the behavior of the model to
  - Control the trade-off between **diversity** vs **quality**
  - Cater to both a user's **long term** and **short-term** interests
  - Generate sets that **multiple users** will like
  - Provide both **personalized** and **popular**/**trending** items
  - ...
- We could obtain this control if we could interpolate between the behaviours of multiple DPPs.

Methods for Blending DPPs

- Suppose we have a set of M DPP kernels $\{L^{(1)}, L^{(2)}, \ldots, L^{(M)}\}$ with blending weights $\boldsymbol{\alpha}$, and we would like to blend their properties into a single model. We first consider two simple methods.

Mixture of DPPs

$$\mathcal{P}^{\alpha}_{L_{1:M}} = \sum_{i=1}^{M} \mathcal{P}^{(i)}\alpha_i$$

1: **Given:** a mixture of DPPs $\mathcal{P}^{\alpha}_{L_{1:M}}$
2: Select a DPP with probability $\boldsymbol{\alpha}$
3: Sample from the selected DPP

DPP with convex mixture of kernels

$$\mathrm{L}^{\alpha} = \sum_{i=1}^{M} \mathrm{L}^{(i)}\alpha_i$$

1: **Given:** a collection of kernels and weights $\boldsymbol{\alpha}$
2: Compute the convex kernel combination kernel $\mathbf{L}^{\alpha}$
3: Sample from the DPP with kernel $\mathbf{L}^{\alpha}$

Applications Beyond Simple Blending

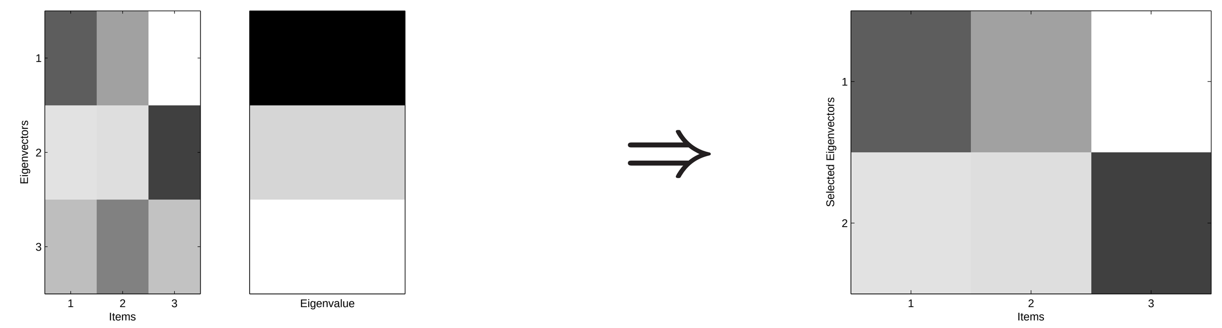
- Suppose each kernel models a family member's preferences in movies, and we would like to recommend a set of movies that will be satisfying to the whole family. Then
  - The mixture model will in general draw sets containing items that are desirable to only one of the users.
  - Suppose Bob likes violent movies. His daughter Alice likes cartoons. The convex mixture will prefer violent cartoons, which are satisfying to neither user.

A Closer Look at DPPs

- We will introduce a more sophisticated method for blending multiple DPPs. First, we must consider more detailed properties of DPPs (see the paper for a more technically precise description).
- DPPs are mixtures of simpler, **elementary** DPPs,

$$\mathcal{P}_L \propto \sum_{J\subseteq\{1,\ldots,N\}} \mathcal{P}^{V_J} \prod_{n\in J}\lambda_n\ .$$

- To draw from a DPP:
  - Compute the eigendecomposition of the kernel matrix $L = \sum_n \lambda_n v_n v_n^\intercal$
  - Draw a subset V of the eigenvectors proportional to the product of their eigenvalues $\boldsymbol{\lambda}$
  - Use these as features $\bar{\mathrm{B}}$ to construct a new elementary DPP with kernel $\bar{\mathrm{L}} = \bar{\mathrm{B}}^\intercal\bar{\mathrm{B}}$
  - Draw $|V|$ items from this DPP, which is easy because the new features are orthogonal



- The chosen eigenvectors are **features in a latent space** which defines the new DPP.

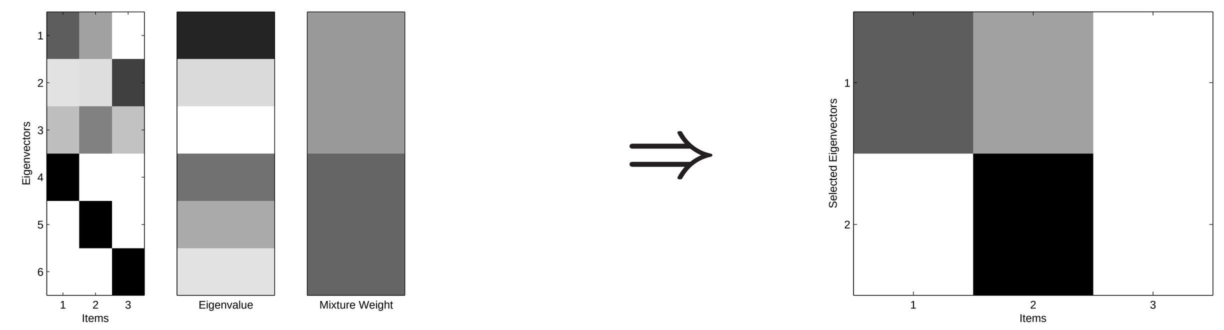
DPP Eigenmixtures

- **Key Idea**: Mix and match the eigenvectors ("eigenfeatures") from the component kernels, to create a **latent space which blends their properties**.
- The model is a **mixture over DPPs** sharing subsets of the **eigenfeature** latent feature representations of the component DPPs:

$$\mathrm{s} \sim \mathrm{Mult}(\alpha, k)$$

$$\mathcal{P}_{\{\mathrm{L}\},s} \propto \sum_{J\in\binom{V^{(m)}}{s_m}} \mathcal{P}^{(V_J,k)} \prod_{n^{(m)}\in J} \lambda_{n^{(m)}}\ .$$

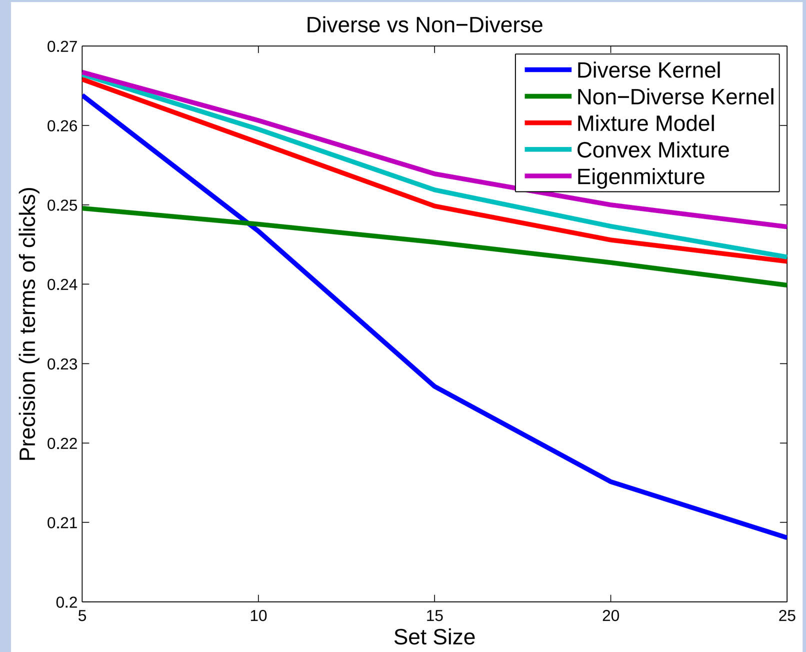
- To draw from a DPP eigenmixture:
  - Compute eigendecompositions of $L^{(1)}, L^{(2)}, \ldots, L^{(M)}$
  - Sample s_m eigenvectors $V^{(m)}$ from each kernel m
  - Use these as features $\bar{\mathrm{B}}$ to construct a new DPP with kernel $\bar{\mathrm{L}} = \bar{\mathrm{B}}^\intercal\bar{\mathrm{B}}$
  - Draw $|V|$ items from this DPP. It is not in general elementary, so sample from it as for any other DPP

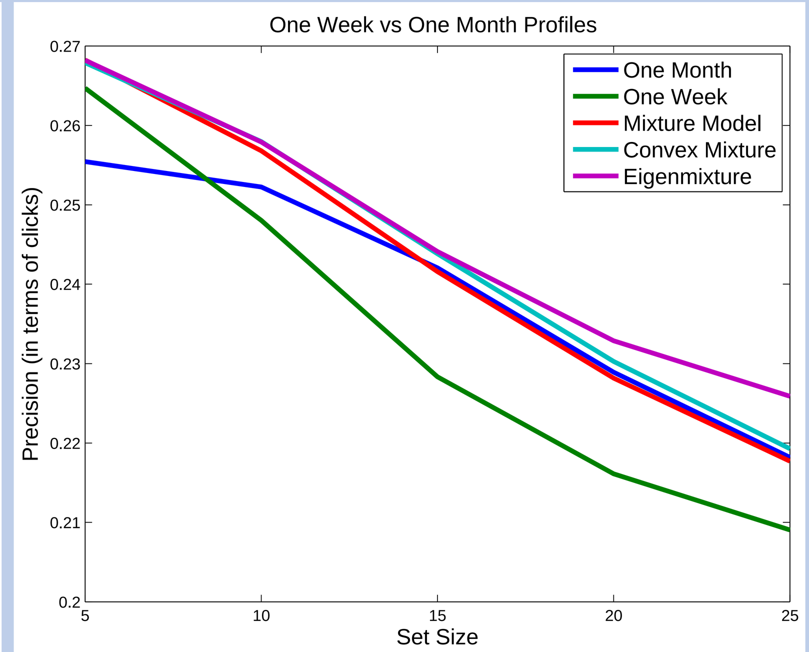


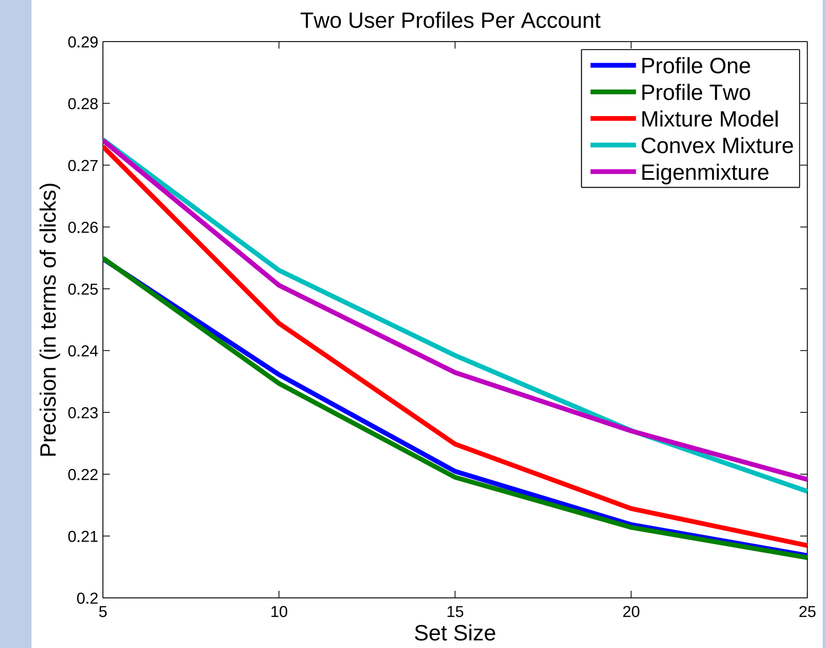
- The squared entries $v(i)^2$ of each eigenfeature v can be viewed as a distribution over the items.
- Alternatively, we can view each eigenfeature as a unit specifying **relative preferences** for each item, and **similarities** between items. Increasing $v(i)^2$ increases the length (implicit quality score) of item i's latent representation, while increasing $v(i)$ and $v(j)$ increases their cosine similarity.
- The convex mixture is equivalent to concatenating the (rescaled) features of the component kernels, while the eigenmixture concatenates the **latent** features.

Personalized News Recommendation

- Data: Yahoo news articles, with click / skip information for 1000 Yahoo users.
- We trained the DPP mixtures via grid search to optimize precision, evaluated with 5-fold cross-validation.
- Task: Draw sets with high precision, in terms of clicks.

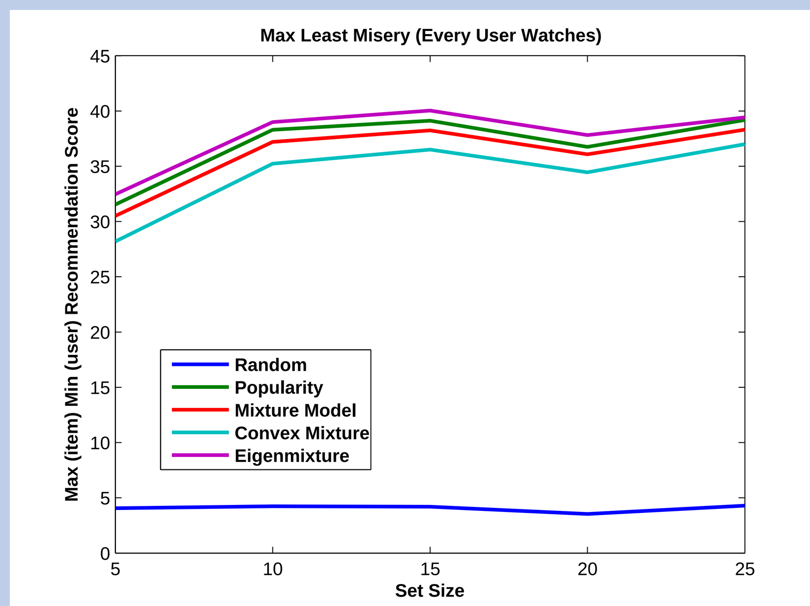


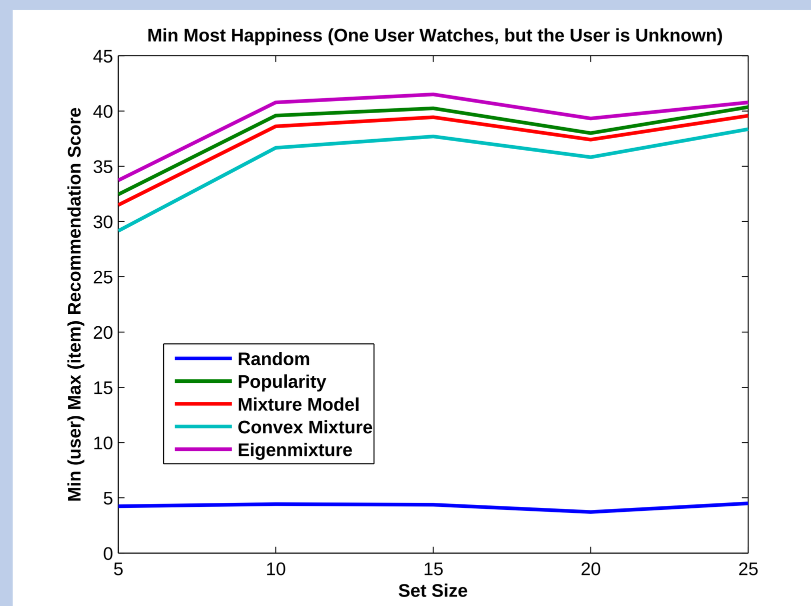


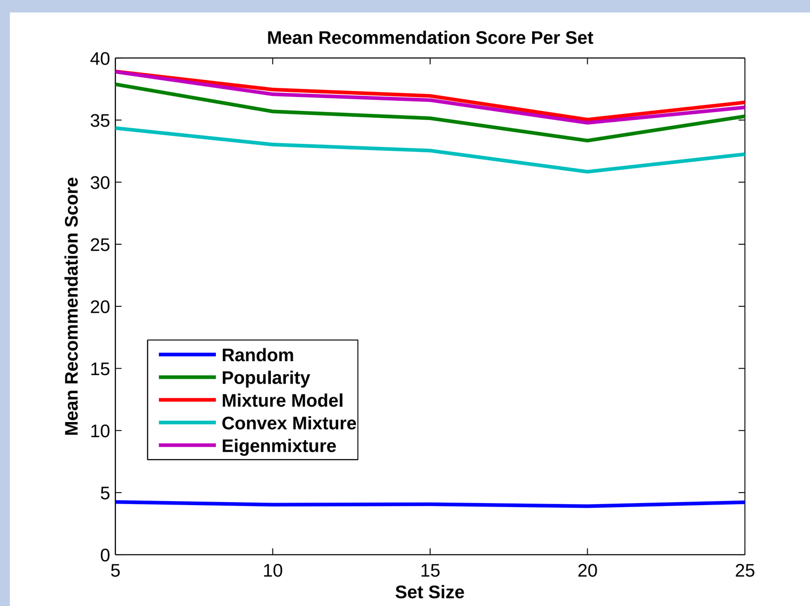


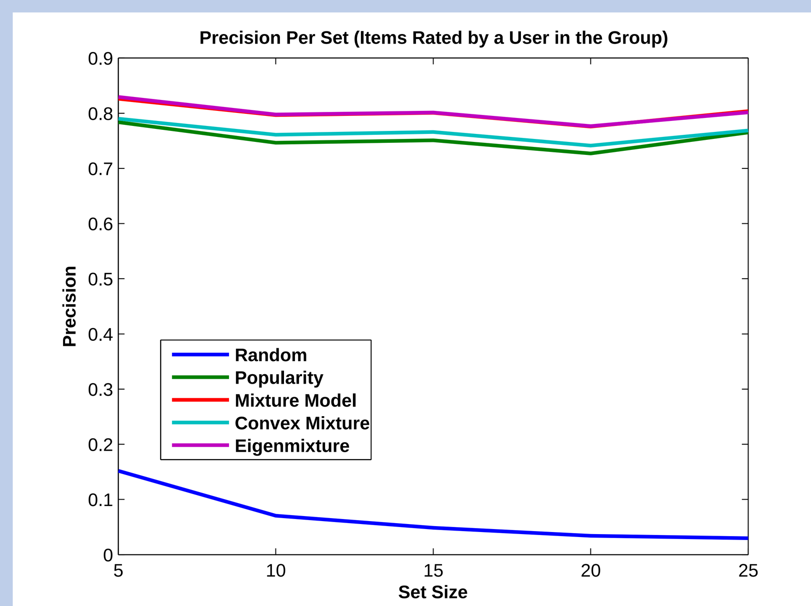
Personalized Group Movie Recommendation

- A group recommendation task on the MovieLens dataset – recommend movies for the entire group.
- 100 groups of 5 users were chosen at random
- Each group was given 1000 sets of movies, chosen from the 10,000 movies.
- Each user had a personalized component DPP. Similarity features: user ratings. Quality scores: neighbourhood-based collaborative filtering. Uniform mixture weights $\boldsymbol{\alpha}$ were used.









{jfoulds@ics.uci.edu, dilangrr@gmail.com}