–Independent Work Report Fall,2015–

# Multi-dimensional Topic Modeling with Determinantal Point Processes

Jeremy Cohen

April 30, 2015

## 1. Abstract

Probabilistic topics models such as Latent Dirichlet Allocation (LDA) provide a useful and elegant tool for discovering hidden structure within large data sets of discrete data, such as corpuses of text. However, LDA implicitly discovers topics along only a single *dimension*. Recent research on *multi-dimensional* topic modeling aims to devise techniques that can discover multiple *groups* of topics, where each group models some different dimension, or aspect, of the data. For example, applying a multi-dimensional topic modeling algorithm to a text corpus could result in three dimensions that turn out to represent "subject," "sentiment," and "political ideology." In this work, we present a new multi-dimensional topic model that uses a *determinantal point process* prior to encourage different groups of topics to model different dimensions of the data. Determinantal point processes are probabilistic models of repulsive phenomena which originated in statistical physics but have recently seen interest from the machine learning community. Though topic models are usually applied to text, we motivate our method with the problem of discovering "tastes" (topics) in a data set of recipes that have been rated by users of a cooking web site. We present both an unsupervised algorithm, which discovers dimensions and their tastes automatically, and a semi-supervised algorithm, which allows the modeler to steer the tastes (topics) towards ones that will be semantically meaningful, by "seeding" each taste with a small number of representative recipes (words). Our results on the recipe data set are mixed, but we are hopeful that the general technique presented here might very well prove useful to multi-dimensional topic modeling in other domains.

a sentence on findings

## 2. Introduction

We consider the following motivating scenario:

**The Recipe Labeling Problem**

A cooking web site would like to label all of the recipes in its database with exactly one *class* in each of several *dimensions*. The dimensions are “cuisine,” “course,” “cooking method,” and “dietary.” The valid classes for the “cooking method” dimension, for instance, include “baked,” “barbeque,” “fried,” and so on. A recipe is *fully labeled* when it is labeled with one class in each dimension — for example, as an “Indian vegetarian baked appetizer.”

The web site could, in theory, hire an army of chefs to fully label every recipe in its database. However, the company’s executives think they might be able to save time and money on this massive labeling task by leveraging Big Data, which they read about in McKinsey Quarterly. The cooking web site does, indeed, have a large volume of potentially useful data, in the form of a “favorites list” of favorite recipes that each user has constructed. The company would therefore like to hand-label only a small number of *seed recipes*, then use the user/recipe favorites data to automatically label the rest. Intuitively, if “samosas” is seeded with {cuisine = “Indian”}, and users who like “samosas” also like “naan,” then “naan” should also be labeled as {cuisine = “Indian”}.

How can the company leverage Big Data to automatically label all of its recipes with the correct class in each dimension?

## 3. Personal Background

The story behind this independent work project begins in Spring 2014, when I took COS 424: Interacting with Data with David Blei. For the final project, three classmates and I scraped a large amount of data from the popular cooking web site www.AllRecipes.com, and applied a machine learning algorithm called *non-negative matrix factorization* (NNMF) [8] to discover interesting trends within the data. Based only on data about *which* users liked *which* recipes (that is, without peeking at the actual titles or contents of the recipes), NNMF was able to identify recognizable "tastes" such as Indian food, Thanksgiving food, Superbowl food, vegan food, "comfort food," breakfast food, barbeque food, and so on. The project left me excited about machine learning, but wanting to learn how to develop my own algorithms, rather than just apply out-of-the-box techniques written by others. So I took a probability course in the fall in order to gain the mathematical fluency that I felt I lacked, and, when the spring semester rolled around, I started looking for research ideas. My advisor, Professor Engelhardt, suggested that I look into an up-and-coming machine learning technique called *determinantal point processes*.

A determinantal point process (or DPP) is a probabilistic model, with origins in statistical physics, that describes the behavior of a system of particles that tend to "repel" each other. I spent a lot of time over intersession trying to think up a new applications for DPPs, but I kept drawing a blank. So I procrastinated by instead writing a technical blog post about my COS 424 recipes project from the previous year.[1] I posted a link to the article on the programming forum "Hacker News," and I was ecstatic when it floated to the top of the front page. Later that day, I received a surprising email. It was from a woman named Julie Mumford who apparently directs advertising at AllRecipes.com, the cooking web site from which I had scraped the data. She had somehow seen my blog post and wanted to chat on the phone. My initial reaction was worry: did she want to reprimand me for scraping and publishing her company's data? Thankfully, it turned out, she did not. She just wanted to find out more about the machine learning techniques that my classmates and I had used to extract patterns in her company's data. She closed with an offer: if I wanted access to private AllRecipes data for further academic work, she'd see what she could do.

So it was that I ended up incorporating *both* the AllRecipes recipe data set *and* determinantal point processes into this independent work project. Many thanks go to Julie and to AllRecipes.com for

[1]The blog post, titled "Decomposing the Human Palate with Matrix Factorization," is available online at http://www.jeremymcohen.net/posts/taste.

providing me with the company's "secret sauce" — the fruits of labor of a team of employees at AllRecipes who are charged with labeling each recipe by hand. These labels are not exposed on the web site, but are used behind the scenes for recommendations and targeted advertisements. I would not have been able to run nor evaluate my semi-supervised approach without access to this private company label data.

## 4. Contents

In a moment, I will formally define "the taste modeling problem" (along with its multi-dimensional and semi-supervised variants) and I will introduce notation that will be used throughout this paper. Then, in section 6, I will describe Latent Dirichlet Allocation (LDA), a simple probabilistic model which forms the basis for the more sophisticated techniques presented in this work. In section **??**, I will survey existing approaches to multi-dimensional topic modeling. Next, in section **??**, I will introduce determinantal point processes and review their most important properties. In 9, I will describe one particularly relevant recent paper where the authors used a determinantal point process as a prior over latent variables in generative model, much like I am proposing to do here. Finally, in section 10, I will present the core contribution of this work: a multi-dimensional topic model which encourages groups of topics to model different dimensions of the data via a determinantal point process prior. In section **??**, I will derive an inference algorithm to "fit" the model. Then, in **??**, I will extend the model to a semi-supervised setting where we "seed" the dimensions and tastes by labeling a small number of recipes as exemplars of each taste. Finally, I give experimental results in section 13, and then conclude.

## 5. The Taste Modeling Problem

Here, I will formally define "the taste modeling problem" that will be tackled in the remainder of this work. The taste modeling problem comes in three flavors: unidimensional, multidimensional, and multidimensional with semi-supervision. I will cover each in turn.

All three variants share the following facts. A cooking web site has $U$ users, indexed in this paper by the letter $i$, and $R$ recipes, indexed by $j$. Each user $i$ has marked $N_i$ recipes as his favorites, indexed by $n$. User $i$'s $n$th favorite is represented as $\mathbf{x}_{(i)n}$, an indicator vector of length $R$ where $x_{(i)nj} = 1$ if user $i$'s $n$th favorite is recipe $j$ and $x_{(i)nj} = 0$ otherwise. Let $\mathbf{x}_{(i)}$ denote all of user $i$'s favorites, and let $\mathbf{X}$ denote the entire data set of user/recipe favorites.

**5.0.1. Unidimensional** The goal of the unidimensional taste modeling problem is to infer, from the data $\mathbf{X}$, a set of $K$ tastes $\beta = \{\beta_1, \dots, \beta_K\}$. Each taste $\beta_k$ should be a vector of length $R$ containing positive numbers which sum to 1.

$$\sum_{j=1}^{R} \beta_{kj} = 1 \qquad \text{for each } j \tag{1}$$

$$\beta_{kj} > 0 \qquad \text{for each } j,k \tag{2}$$

Each taste can therefore be viewed as a probability distribution over the recipes. If $\beta_{kj}$ is high, then taste $k$ places a high probability mass on recipe $j$. Ideally, one should be able to intuit a meaning for every taste. For example, a taste that placed a high probability mass on the recipes "naan" and "chicken tikka masala" we would interpret as an Indian food taste. Note that the tastes $\{\beta_k\}_{k=1}^{K}$ will also provide a low-dimensional representation of each recipe $j$ in the numbers $\beta_{1j}, \beta_{2j}, \dots, \beta_{Kj}$. As we will see, Latent Dirichlet Allocation should suffice to solve the unidimensional taste modeling problem.

**5.0.2. Multi-dimensional** In the *multi-dimensional* taste modeling problem, we still aim to infer a set of $K$ tastes $\beta = \{\beta_1, \dots, \beta_K\}$ from the data $\mathbf{X}$. As before, each taste $\beta_k$ is still an $R$-vector of positive numbers that sums to 1. However, we now additionally partition the set of tastes $\beta$ into $G$ *taste groups* $\beta = \{\beta^{(1)}, \dots, \beta^{(G)}\}$, where each taste group is intended to represent a different dimension of recipes. For example, one taste group might represent the "cuisine" of a recipe, and the tastes therein might represent Italian, Indian, Chinese, etc. Another taste group might represent the "cooking method" of a recipe, with tastes for baking, broiling, etc.

Let the $g$th taste group contain $M_g$ tastes, indexed by $m$. That is, we have the constraint

$$\sum_{g=1}^{G} M_g = K \tag{3}$$

For notational convenience, we will denote the *m*th taste in taste group $g$ as $\beta_m^{(g)}$. That is, we adopt two views of the same set $\beta$ of tastes: grouped indexing $\beta_m^{(g)}$ and flat indexing $\beta_k$. Assume that there exists some function that converts between grouped indices $(g, m)$ and flat indices $k$. In the remainder of this work, I will sometimes iterate over the tastes $\sum_{g=1}^{G} \sum_{m=1}^{M_g}$ and I will sometimes iterate over them $\sum_{k=1}^{K}$. The two views are equivalent. We define the grouped view so that we can neatly represent a "diversifying" prior over the taste groups $\{\beta^{(1)}, \ldots, \beta^{(G)}\}$, but we keep the flat view so that we can keep the form of the bulk of the LDA model the same.

We will develop a model called DPP-LDA to try to solve the multi-dimensional taste modeling problem.

**5.0.3. Multi-dimensional with semi-supervision** The final variant of the taste modeling problem introduces semi-supervision to the multi-dimensional model. Suppose that, for a small set of *labeled* recipes $\mathbb{L} \subseteq \{1, \ldots, R\}$, we specify a label $\mathbf{y}_j^{(g)}$ for each taste group, where $\mathbf{y}_j^{(g)}$ is an indicator vector such that $y_{jm}^{(g)} = 1$ if recipe $j$ is labeled with taste $m$ in group $g$, and $y_{jm}^{(g)} = 0$ otherwise. For example, if we might label "naan" as an Indian baked food by setting $y_{\text{naan,baked}}^{\text{cooking method}} = 1$ and $y_{\text{naan,Indian}}^{\text{cuisine}} = 1$. The purpose of the semi-supervision is to "guide" the algorithm towards the taste groups and tastes that we want to see. For example, the fictional cooking web site described above in the "recipe labeling" scenario could use a semi-supervised multi-dimensional topic model to label all of its recipes. The company would "seed" the tastes with a small number of hand-labeled recipes, fit a semi-supervised multi-dimensional topic model, and label each recipe $j$ in dimension $g$ with the $m$ for which the learned coefficient $\beta_{mj}^{(g)}$ is highest.

We will add semi-supervision to the DPP-LDA model with a model called S-DPP-LDA,

## 6. Latent Dirichlet Allocation

In this section, I will review Latent Dirichlet Allocation (LDA) [1], and explain how to apply it to the unidimensional taste modeling problem. LDA is widely used in the text modeling community as a "topic model" for extracting latent structure from large corpuses of text. For example, applying LDA to a corpus of newspaper articles might result in topics for sports, politics, crime, and so on. Each topic is a distribution over words. The "sports" topic, for example, would put a high probability mass on the words "game," "inning," and "point." While LDA is very popular as a model for text data, applying it to the taste modeling problem is straightforward: "topics" become tastes, "words" become recipes, and "documents" become users.

Latent Dirichlet Allocation is a probabilistic *generative model* for the user/recipe favorites $\mathbf{X}$. In other words, it defines a random process according to which we assume the data $\mathbf{X}$ was generated. The tastes $\beta$ are a so-called *latent* (or "hidden") variable in the generative process: we do not observe them, but we do observe the favorites data $\mathbf{X}$, which was influenced by the tastes $\beta$. Having defined a probabilistic model that purports to explain how $\mathbf{X}$ was generated conditioned on $\beta$, if we observe $\mathbf{X}$, we can ask: what values of $\beta$ were most likely to have generated $\mathbf{X}$? This is a probabilistic inference problem.

### 6.1. The LDA Model

LDA assumes that each user $i$ has a vector $\theta_{(i)}$ of *taste preferences*. $\theta_{(i)}$ is a vector of length $K$ where the component $\theta_{(i)k}$ quantifies user $i$'s affinity for taste $k$. For example, if taste $k$ is "breakfast," then $\theta_{(i)k}$ will be high for users $i$ who like breakfast foods. All of the components of $\theta_{(i)}$ must be positive, and the whole vector must sum to 1 (for reasons which will be clear momentarily). LDA assumes that $\theta_{(i)}$ was itself drawn from a *Dirichlet distribution*, which is a distribution over vectors of a fixed length (in this case, $K$) that sum to 1.

LDA assumes that user $i$'s favorite recipe choices were generated based on (1) his own taste preferences $\theta_{(i)}$, and (2) the tastes themselves $\beta_1 \dots \beta_K$. Specifically, the model assumes that each of user $i$'s $N_i$ favorite recipes was selected in the following manner:

1. First, someone — say, Tyche, the Greek goddess of fortune — drew a taste $\mathbf{z}_{(i)n}$ at random based on user $i$'s taste preferences $\theta_{(i)}$. (The vector $\mathbf{z}_{(i)n}$ is in bold because it is an indicator vector of length $K$, where $z_{(i)nk} = 1$ if Tyche drew taste $k$.) The probability that Tyche draws taste $k$ is $p(z_{(i)nk} = 1) = \theta_{(i)k}$, each of which defines a distribution over the recipes. (See why $\theta_{(i)k}$ was constrained to sum to 1?) In the language of statistics, we say that $\mathbf{z}_{(i)n}$ was drawn from a *categorical distribution*: $\mathbf{z}_{(i)n} \sim \text{Categorical}(\theta_{(i)k})$. For example, if user $i$'s taste preferences are 60% breakfast food and 40% vegan food, then there is a 60% chance that $z_{(i),n,\text{breakfast}} = 1$, and a 40% chance that $z_{(i),n,\text{vegan}} = 1$.
2. Then, Tyche drew a recipe $\mathbf{x}_{(i)n}$ (an indicator vector of length $R$) at random from the taste indexed by $\mathbf{z}_{(i)n}$. This is possible because the $k$th topic $\beta_k$ is a vector of positive numbers that sums to 1, so it can parameterize a categorical distribution. For example, if $z_{(i),n,\text{breakfast}} = 1$, then Tyche draws the $n$th favorite recipe $\mathbf{x}_{(i)n}$ from the "breakfast" topic, which likely places a high probability mass on the recipes "pancakes" and "waffles." Thus, $p(x_{(i)n,\text{pancakes}} = 1)$ and $p(x_{(i)n,\text{waffles}} = 1)$ would be high.

More compactly, the generative model of LDA is the following:

$$\beta_k \sim \text{Dirichlet}_R(\eta) \tag{4}$$

$$\theta_{(i)} \sim \text{Dirichlet}_K(\alpha) \tag{5}$$

$$\mathbf{z}_{(i)n} \sim \text{Categorical}\left(\theta_{(i)}\right) \tag{6}$$

$$\mathbf{x}_{(i)n} \sim \text{Categorical}\left(\beta_{z_{(i)n}}\right) \tag{7}$$

Only the favorites $\mathbf{X}$ are observed; from these variables, we would like infer the most likely values for the hidden $\mathbf{Z}$, $\Theta$, and especially $\beta$. The inference algorithm for LDA will be described in detail in later sections.

### 6.2. Towards a multi-dimensional taste model

Applying LDA to a the unidimensional taste modeling problem might conceivably result in a set of tastes such as: {vegan, Indian, dessert, Italian, appetizer, fish}. Any recipe could then be represented in a low-dimensional "taste space" by 6 numbers — its coefficients across all 6 topics (note that those numbers do not necessarily sum to 1).

But wait a minute: appetizer and dessert are both courses of a meal, Indian and Italian are both cuisines, and vegan and fish are both dietary categories. What if we could automatically learn a more structured representation of each recipe by learning a set of three *groups* of tastes, each group modeling a different dimension: {{Indian, Italian}, {appetizer, dessert}, {vegan, fish}}. This is the aim of multi-dimensional taste modeling.

## 7. Related Work: Multi-dimensional topic models

Several researchers have recently proposed new topic models that can learn more than one dimension of topics. Many of these multi-dimensional topic models are extensions of LDA. The *Topic-Aspect Model* (TAM) [13] is a two-dimensional topic model where topics where the two dimensions are called "topics" and "aspects." For example, when a TAM with two aspects is applied to a corpus of editorials on the Israeli-Palestinian conflict, one aspect appears to correspond to the Israeli perspective ("war," "society," "terrorist"), while the other aspect corresponds to the Palestinian perspective ("occupation," "resistance," "intifada"). In TAM, each document is associated with both a distribution over topics and a distribution over aspects. Each word is generated by first choosing between whether to sample from a topics or from an aspect.

While TAM is limited to two dimensions, *factorial LDA* [11] allows an arbitrary number of dimensions. Unlike standard LDA, which associates each word appearance with a *single* topic, factorial LDA assumes that each word appearance was generated from a *tuple* of topics—one for each dimension. The distribution over the vocabulary that is associated with each possible tuple of topics is modeled as drawn from a Dirichlet prior that depends on the topics in the tuple, so that similar tuples will be associated with similar distributions over the vocabulary.

Likewise, *finite factored topic models* (FFTMs) [6] also associate each word appearance with one topic from every dimension. FFTMs model the distribution over words, given the latent topic tuple, as simply the sum of the topics from every dimension.

In the following sections, I will present a new algorithm for multi-dimensional taste modeling (topic modeling). Unlike the models above, the solution I propose will keep the bulk of the LDA model intact, but replace the independent prior on the tastes $p(\beta) = \prod_{k=1}^{K} p(\beta_k)$ with a determinantal point process prior that encourages the set of taste groups $\beta^{(1)}, \ldots, \beta^{(g)}$ to model a diverse set of dimensions of recipes.

## 8. Determinantal Point Processes

This section provides an introduction to determinantal point processes. Readers interested in learning more about DPPs should consult Kulesza and Taskar's thorough survey of the subject [7], which the presentation in this section closely tracks. Be advised that the notation in this section will at times clash with notation defined elsewhere in this paper.

A *point process* $\mathscr{P}$ on a ground set $\mathscr{Y}$ is a probability distribution over subsets of $\mathscr{Y}$. For example, If $\mathscr{Y}$ is the real line $\mathbb{R}$, a draw from $\mathscr{P}$ is a set of real numbers. If $\mathscr{Y}$ is the set of all countries in the world, a draw from $\mathscr{P}$ is a set of countries.

A *determinantal point process* (DPP) is a type of point process that favors subsets whose elements are "diverse," according to some pairwise measure of similarity defined for every pair of elements.[2] For example, suppose that the United Nations needs to select a set of countries to serve a rotation on the Security Council. This is a tricky balancing act. If the council includes too many European countries, China and Russia will protest. If too many African and Asian countries are chosen, Europe and America will get upset. In order to keep everyone happy, Ban Ki-Moon must select a "diverse" set of countries.

A good way to do this would be to draw the security council membership from a determinantal point

[2]Determinantal point processes were first studied by physicists who were modeling the "anti-bunching" behavior exhibited by fermions in thermal equilibrium. The Pauli exclusion principle dictates that no more than one fermion may inhabit the same quantum state. As a result, the fermions behave exactly according to a determinental point process.[9]

process. The UN would need only quantify the similarity between every pair of countries, and the determinantal point process would tend to yield sets of countries that are un-similar to each other.

**8.0.1. Definition** A determinantal point process over a ground set $\mathscr{Y}$ of size $N = |\mathscr{Y}|$ is fully parameterized by a *marginal kernel* $\mathbf{K}$, which is an $N \times N$ real, symmetric, positive semi-definite matrix that encodes the similarity between every pair of elements of $\mathscr{Y}$.

For example, if the countries eligible to enter the security council lottery are $\mathscr{Y} = \{\text{Korea}, \text{Japan}, \text{Australia}\}$, then the marginal kernel encoding the pairwise similarity between all countries in $\mathscr{Y}$ would be:

$$\mathbf{K} = \begin{bmatrix} K_{\text{kor,kor}} & K_{\text{kor,jap}} & K_{\text{kor,aus}} \\ K_{\text{jap,kor}} & K_{\text{jap,jap}} & K_{\text{jap,aus}} \\ K_{\text{aus,kor}} & K_{\text{aus,jap}} & K_{\text{aus,aus}} \end{bmatrix} \tag{8}$$

where $K_{\text{kor,jap}} = K_{\text{jap,kor}}$ is the similarity between Korea and Japan.

The "cumulative distribution function" (of sorts) for a DPP $\mathscr{P}$ is defined as follows. Let $\mathbf{Y} \subseteq \mathscr{Y}$ be a random draw from the DPP. Let $\mathbf{A} \subseteq \mathscr{Y}$ be an arbitrary subset of $\mathscr{Y}$. (Note that $\mathbf{Y}$ is a random variable but $\mathbf{A}$ is not.) The probability that every element in $\mathbf{A}$ is included in the draw $\mathbf{Y}$ from $\mathscr{P}$ is:

$$\mathscr{P}(\mathbf{A} \subseteq \mathbf{Y}) = \det(\mathbf{K}_{\mathbf{A}}) \tag{9}$$

where $\det(\cdot)$ is the determinant of a matrix and $\mathbf{K}_{\mathbf{A}}$ is a submatrix of $\mathbf{K}$ consisting of only the rows and columns corresponding to elements in $\mathbf{A}$.

For example, if we let $\mathbf{A} = \{\text{kor}, \text{aus}\}$, then equation 9 tells us that the probability that both Korea and Australia are included in a random draw $\mathbf{Y}$ is:

$$\begin{aligned}\mathscr{P}(\{\text{kor},\text{aus}\} \subseteq \mathbf{Y}) =&\ \det\left(\mathbf{K}_{\{\text{kor},\text{aus}\}}\right) \\ =&\ \det\begin{pmatrix} K_{\text{kor,kor}} & K_{\text{kor,aus}} \\ K_{\text{aus,kor}} & K_{\text{aus,aus}} \end{pmatrix} \\ =&\ K_{\text{kor,kor}} K_{\text{aus,aus}} - K^2_{\text{kor,aus}} \end{aligned} \tag{10}$$

Equation 10 shows that when $K_{\text{kor,aus}}$ — the similarity between Korea and Australia — is large, the probability that both countries will appear together in the set drawn from $\mathscr{P}$ is small. This example illustrates why the set $\mathbf{Y}$ drawn from a DPP will tend to be diverse, with respect to the kernel $\mathbf{K}$ .

Alternatively, if we let $\mathbf{A} = \{\text{jap}\}$, then by equation 9, the probability that Australia will be included in a random draw $\mathbf{Y}$ is:

$$\begin{aligned}\mathscr{P}(\{\text{aus}\} \subseteq \mathbf{Y}) =&\ \det(K_{\text{aus,aus}}) \\ =&\ K_{\text{aus,aus}}\end{aligned} \tag{11}$$

Therefore, $K_{\text{aus,aus}}$ is the "marginal" probability that Australia will be included in $\mathbf{Y}$.

We now have a complete picture of the meaning of the marginal kernel $\mathbf{K}$ that parameterizes a DPP. The diagonal elements $\{K_{ii}\}_{i=1}^N$ are the "qualities" of each element $i$; if $K_{ii}$ is large, then element $i$ is likely to appear in the random draw. On the other hand, the off-diagonal elements $\{K_{ij}\}_{i \neq j}$ are the "similarities" between each pair of elements $i$ and $j$; if $K_{ij}$ is large then elements $i$ and $j$ are unlikely to both appear in the random draw. A DPP is therefore guided by two contradicting imperatives: to select elements that are each, on their own, "high quality," and simultaneously ensure that the whole set of elements selected is "diverse."

**8.0.2. L-Ensembles** Notice that equation 9 gives the probability that any arbitrary set $A \subseteq \mathscr{Y}$ is *included* in the draw $\mathbf{Y}$; it tells us nothing about the probability that any set $Y \subseteq \mathscr{Y}$ actually *is* the set $\mathbf{Y}$ drawn from the DPP. Fortunately, if we want a direct formula for $\mathscr{P}(\mathbf{Y} = Y)$, we can parameterize a DPP using an *L-ensemble* **L** rather than a marginal kernel **K**. Like the marginal kernels discussed above, an L-ensemble is a real, symmetric, $N \times N$ matrix which encodes a similarity measure between each pair of elements in $\mathscr{Y}$.

The unnormalized probability mass function of a DPP $\mathscr{P}_L$ parameterized by an L-ensemble **L** is defined as:

$$\mathscr{P}_L(\mathbf{Y} = Y) \propto \det(\mathbf{L}_Y) \tag{12}$$

One can show that the required normalization constant $\sum_{Y \subseteq \mathscr{Y}} \mathscr{P}_L(\mathbf{Y} = Y)$ is equal to $\det(\mathbf{L} + I)$. Therefore, the exact, normalized probability mass function is given by:

$$\mathscr{P}_L(\mathbf{Y} = Y) = \frac{\det(\mathbf{L}_Y)}{\det(\mathbf{L} + I)} \tag{13}$$

Any $L$-ensemble can be transformed into a DPP parameterized by a marginal kernel. The reverse is not always true: a DPP parameterized by a marginal kernel **K** can only be re-parameterized with an $L$-ensemble if it assigns a non-zero probability to the empty set.

All of the DPPs used in this paper will be $L$-ensembles.

**8.0.3. Quality / diversity decomposition** As mentioned above, a DPP favors selecting sets where each element is individually high-quality, but the whole set is simultaneously diverse. We can make this tradeoff more explicit by decomposing an $L$-ensemble into a *quality model* $\pi$ and a *diversity model* $S$. Each entry of **L** is decomposed as:

$$\mathbf{L}_{ij} = \mathbf{S}_{ij} \sqrt{\pi_i \pi_j} \tag{14}$$

where $\pi_i \in \mathbb{R}^+$ is the individual quality of element $i$ and $S_{ij} \in [-1, 1]$ is the similarity between $i$ and $j$. In the remainder of this paper, I will make use of the quality/diversity decomposition.

**8.0.4. DPPs vs. MRFs** See Appendix **??** for a comparison of DPPs with Markov random fields, another statistical model which could conceivably be used to define a probability distribution over "diverse" subsets of a ground set.

## 9. Related Work on Diversity Priors for Latent Variable Models

One promising application of determinantal point processes to machine learning is as prior distributions for encouraging diversity in generative latent variable models. In this section, I will describe the approach in [14], where the authors Zou and Adams propose DIVERSIFIED-LDA, a variant of LDA where the set of topics is encouraged to be "diverse" via a determinantal point process prior. The motivation for DIVERSIFIED-LDA is that standard LDA often places a high probability mass, in many topics, on the same set of "stopwords." For example, the stopwords "the," "to," and "and" might be assigned high probability under several topics; we would prefer, however, that all stopwords be bunched together into one topic. We desire, then, that the set of topics be "diverse" — that they place high probability mass on different words. This can be achieved by modeling the set of topics as a draw from a DPP.

Recall how standard LDA assumes that the topics $\beta = \{\beta_k\}_{k=1}^{K}$ were drawn independently from identical Dirichlet distributions:

$$\begin{aligned} p(\beta) &= \prod_{k=1}^{K} p(\beta_k) \\ &= \prod_{k=1}^{K} \text{Dirichlet}(\beta_k \,|\, \eta) \end{aligned} \tag{15}$$

DIVERSIFIED-LDA, in contrast, does away with the independence assumption, and instead assumes that the set of topics was drawn all at once from a determinantal point process:

$$p(\beta) = \text{DPP}(\beta \,|L) \tag{16}$$

Previously, we saw that a DPP over a finite set of $K$ elements may be parameterized by a symmetric $K \times K$ kernel $\mathbf{L}$ which encodes the similarity between every pair of elements. Here, however, an "element" is a topic — a $V$-dimensional vector of positive numbers which sums to 1 — and the ground set of possible elements is infinite. Therefore, instead of defining a kernel matrix $\mathbf{L}$ where $L_{k_1k_2}$ gives the similarity between elements $k_1$ and $k_2$, we define a kernel function $L$ where $L(\beta_{k_1}, \beta_{k_2})$ gives the similarity between any two topics $\beta_{k_1}$ and $\beta_{k_2}$.

We would like the prior on the topics $\beta$ to incorporate two separate, possibly conflicting, biases: each topic $\beta_k$ should come from a Dirichlet distribution, but the set of topics should collectively be diverse. The first bias is to discourage, say, a topic that places a probability mass of 1 on one word and 0 on all the rest. The second bias is to discourage the topics from being similar to each other. We may incorporate both of these biases into a determinantal point process by making use of the quality/diversity decomposition described above. That is, we write the kernel function $L$ as the combination of a quality function $\pi$ and a similarity function $S$:

$$L(\beta_{k_1}, \beta_{k_2}) = S(\beta_{k_1}, \beta_{k_2}) \sqrt{\pi(\beta_{k_1})\, \pi(\beta_{k_2})} \tag{17}$$

where $\pi$ is a Dirichlet prior

$$\pi(\beta_k) = \text{Dirichlet}(\beta_k \,|\, \eta) \tag{18}$$

and $S : \mathbb{R}^V \times \mathbb{R}^V \to [0,1]$ is some function that gives the similarity between any pair of topics.

For the similarity function $S$, Adams and Zou choose the probability product kernel [5], which takes a parameter $p$. Under the probability product kernel with parameter $p$, the similarity between any two topics $k_1$ and $k_2$ is:

$$S(\beta_{k_1}, \beta_{k_2}) = \frac{\sum_V (\beta_{k_1,v} \beta_{k_2,v})^p}{\sqrt{\sum_V \beta_{k_1,v}^{2p}} \sqrt{\sum_V \beta_{k_2,v}^{2p}}} \tag{19}$$

Putting it all together, the probability that a topic set $\beta = \{\beta_k\}_{k=1}^K$ is drawn from the DPP is proportional to:

$$
\begin{aligned}
p(\beta) &\propto \text{DPP}(\beta \,|L) \\
&= \det \begin{pmatrix} L(\beta_1, \beta_1) & \dots & L(\beta_1, \beta_K) \\ \vdots & \ddots & \vdots \\ L(\beta_K, \beta_1) & \dots & L(\beta_K, \beta_K) \end{pmatrix}
\end{aligned}
\tag{20}
$$

## 10. Multi-dimensional Taste Modeling

Inspired by the approach taken by DIVERSIFIED-LDA, I will now present the main contribution of this paper: DPP-LDA, a multi-dimensional taste model where the taste groups are encouraged to represent a diverse set of dimensions via a determinantal point process prior.

The generative model for DPP-LDA is identical to that of ordinary LDA except that instead of imagining that the tastes were each drawn independently from a Dirichlet prior:

$$\beta_k \sim \text{Dirichlet}(\eta) \quad \text{for all } k = \{1 \dots K\} \tag{21}$$

we imagine that the set of taste groups $\beta^{(1)}, \beta^{(2)}, \dots \beta^{(G)}$ was drawn all at once from a determinantal point process:

$$\beta \sim \text{DPP}(L) \tag{22}$$

That is, an "element" in the ground set of the DPP is an entire taste group $\beta^{(g)}$, a $M_g \times R$ matrix.[3]

We need to choose an L-ensemble function $L\left(\beta^{(g_1)}, \beta^{(g_2)}\right)$ to parameterize the DPP. Ideally, $L$ should incorporate two biases:

1. Each taste $\beta_m^{(g)}$ should roughly follow a Dirichlet distribution. That is, our DPP should assign a low probability to any set of taste groups where any taste in any group egregiously deviates from

[3]Notice how our approach is completely different from that of DIVERSIFIED-LDA described above, where an "element" in the ground set of the DPP prior was a taste $\beta_k$, a $V$-vector.

a Dirichlet distribution — e.g. puts all of its probability mass on a single word.

2. The set of taste groups $\beta^{(1)} \dots \beta^{(G)}$ should be diverse with respect to some measure of diversity. For example, as an extreme case, our DPP should assign a very low probability to any set of taste groups where two of the taste groups $\beta^{(g_1)}$ and $\beta^{(g_2)}$ are identical.

Following the DIVERSIFIED-LDA approach described above, we will decompose $L$ into a *quality* function $\pi$ and a *similarity* function $S$.

$$L\left(\beta^{(g_1)}, \beta^{(g_2)}\right) = S\left(\beta^{(g_1)}, \beta^{(g_2)}\right) \sqrt{\pi\left(\beta^{(g_1)}\right) \pi\left(\beta^{(g_2)}\right)} \tag{23}$$

For the quality function, we choose a product of independent Dirichlet densities, one for each taste in the taste group:

$$\pi\left(\beta^{(g)}\right) = \prod_{m=1}^{M_g} \text{Dirichlet}(\beta_m^{(g)} \,|\, \eta) \tag{24}$$

Defining the similarity function $S$ between any pair of taste groups $\beta^{(g_1)}$ and $\beta^{(g_2)}$ is trickier. We would like the $G$ taste groups to each represent a different "dimension" of the data. We therefore need a way to quantify the degree to which any two taste groups $g_1$ and $g_2$ represent the same "dimension" of the data. The solution we adopt here is use the Gram matrix of a topic group $\beta^{(g)}$ as a proxy for the "dimension" of the data that $g$ represents. We can then measure the similarity between two taste groups $g_1$ and $g_2$ by comparing their Gram matrices element by element.
The Gram matrix of taste group $g$ is:

$$\Lambda^{(g)} = \beta^{(g)^T} \beta^{(g)} \tag{25}$$

You can think of the Gram matrix $\Lambda^{(g)}$ as a kind of *recipe-similarity* matrix, where $\Lambda^{(g)}_{j_1 j_2}$ is a measure of the similarity between recipes $j_1$ and $j_2$ in topic group $g$. Why is this so? First, note the following crucial point: each taste group $\beta^{(g)}$ defines a *recipe feature space* — that is, a representation of each recipe as a feature vector of length $M_g$. In particular, the feature vector for recipe $j$ is $\beta^{(g)}_{(\cdot)j}$, or the $j$th column of $\beta^{(g)}$. The reason why $\Lambda^{(g)}_{j_1 j_2}$ gives the similarity between recipes $j_1$ and $j_2$ in the recipe feature space defined by topic group $g$ is because $\Lambda^{(g)}_{j_1 j_2}$ is the dot product between $g$'s feature vectors for recipes $j_1$ and $j_2$:

$$\begin{aligned}\Lambda^{(g)}_{j_1 j_2} &= \left(\beta^{(g)^T} \beta^{(g)}\right)_{j_1\, j_2} \\ &= \beta^{(g)^T}_{j_i(\cdot)} \cdot \beta^{(g)}_{(\cdot) j_2} \\ &= \beta^{(g)}_{(\cdot) j_1} \cdot \beta^{(g)}_{(\cdot) j_2} \end{aligned} \tag{26}$$

Since $\Lambda^{(g)}$ is an $R \times R$ matrix describing the similarity between each pair of recipes in the recipe feature space defined by taste group $g$, it makes sense to compare taste groups $g_1$ and $g_2$ by directly comparing their recipe-similarity matrices $\Lambda^{(g_1)}$ and $\Lambda^{(g_2)}$, element by element. Two taste groups represent a similar "dimension" of the data if their recipe-similarity matrices are similar.

One question remains: how should we compare $\Lambda^{(g_1)}$ and $\Lambda^{(g_2)}$? In this paper, we choose the following similarity function $R$, which has a form similar to the probability product kernel used by Zou and Adams:

$$R\left(\Lambda^{(g_1)}, \Lambda^{(g_2)}\right) = \frac{\sum_{j_1=1}^{R} \sum_{j_2=1}^{R} \left(\Lambda^{(g_1)}_{j_1\, j_2} \Lambda^{(g_2)}_{j_1\, j_2}\right)^p}{\sqrt{\sum_{j_1=1}^{R} \sum_{j_2=1}^{R} \left(\Lambda^{(g_1)}_{j_1\, j_2}\right)^{2p}} \sqrt{\sum_{j_1=1}^{R} \sum_{j_2=1}^{R} \left(\Lambda^{(g_2)}_{j_1\, j_2}\right)^{2p}}} \tag{27}$$

where $p$ is a hyperparameter. Note that $R\left(\Lambda^{(g_1)}, \Lambda^{(g_2)}\right) = 1$ if $\beta^{(g_1)}$ and $\beta^{(g_2)}$ are identical.

Thus, we use the following similarity function $S\left(\beta^{(g_1)}, \beta^{(g_2)}\right)$ to compare taste groups $g_1$ and $g_2$:

$$\begin{aligned} S\left(\beta^{(g_1)}, \beta^{(g_2)}\right) &= R\left(\Lambda^{(g_1)}, \Lambda^{(g_2)}\right) \\ &= R\left(\beta^{(g_1)^T} \beta^{(g_1)}, \beta^{(g_2)^T} \beta^{(g_2)}\right) \end{aligned}$$

In summary, under our model, the probability assigned to any set of taste groups $\beta$ is proportional to:

$$
\begin{aligned}
p(\beta) &\propto \mathrm{DPP}(\{\beta^{(g)}\}_{g=1}^{G}|L) \\
&\propto \det \begin{pmatrix} L(\beta^{(1)},\beta^{(1)}) & \dots & L(\beta^{(1)},\beta^{(G)}) \\ \vdots & \ddots & \vdots \\ L(\beta^{(G)},\beta^{(1)}) & \dots & L(\beta^{(G)},\beta^{(G)}) \end{pmatrix}
\end{aligned} \tag{28}
$$

where

$$
L\left(\beta^{(g_1)},\beta^{(g_2)}\right) = S\left(\beta^{(g_1)},\beta^{(g_2)}\right)\sqrt{\pi\left(\beta^{(g_1)}\right)\pi\left(\beta^{(g_2)}\right)} \tag{29}
$$

$$
= R\left(\beta^{(g_1)^T}\beta^{(g_1)},\beta^{(g_2)^T}\beta^{(g_2)}\right)\sqrt{\prod_{m=1}^{M_{g_1}}\mathrm{Dir}\left(\beta_m^{(g_1)}|\eta\right)\prod_{m=1}^{M_{g_2}}\mathrm{Dir}\left(\beta_m^{(g_2)}|\eta\right)} \tag{30}
$$

**10.0.5. Complexity** The probability density in equation 28 can be computed in $\mathscr{O}(V^3G^2)$ time.

**Proof.** Computing the $L$-ensemble function $L$ between any pair of taste groups is an $\mathscr{O}(V^3)$ operation due to the matrix multiplication required to compute each Gram matrix $\Lambda$. So computing the kernel function between *each* pair of the $G$ taste groups costs $\mathscr{O}(V^3G^2)$. (Taking the determinant of a $G \times G$ matrix is negligible.)

## 11. Fitting the Model

In the preceding section, we defined a generative model, DPP-LDA, which describes how we think Tyche generated the user/recipe ratings $\mathbf{X}$. Now, given the observed ratings, we would like to "fit" the model — that is, we would like to find the most likely values for the hidden variables:

1. The tastes $\beta = \{\beta_k\}_{k=1}^{K}$ (using flat indexing).
2. The per-user taste preferences $\Theta = \{\theta_{(i)}\}_{i=1}^{U}$.
3. The per-rating taste "responsibilities" $\mathbf{Z} = \{z_{(1)n}\}_{n=1}^{N_1}\dots\{z_{(U)n}\}_{n=1}^{N_U}$.

Note that the "most likely" values for the hidden variables $\{\beta,\Theta,\mathbf{Z}\}$ are those values that are simultaneously (1) most likely to have been generated according to the priors on the hidden variables, and (2) most likely to have *themselves generated* the observed variables **X**.

We could, in theory, infer the "most likely" values for the hidden variables by making use of Bayes' rule, which tells us that for any events $a$ and $b$,

$$p(b|a) = \frac{p(a|b)p(b)}{p(a)} \tag{Bayes' rule}$$

We could therefore use Bayes' rule to compute the "posterior" distribution of the hidden variables given the observed variables like so:

$$p(\beta, \Theta, \mathbf{Z} \mid \mathbf{X}) = \frac{p(\mathbf{X} \mid \beta, \Theta, \mathbf{Z})\, p(\beta, \Theta, Z)}{p(\mathbf{X})} \tag{31}$$

Our probability model tells us how to compute both terms in the numerator of the RHS of equation 31. $p(\mathbf{X} \mid \beta, \Theta, \mathbf{Z})$ is the probability of generating the user/recipe ratings given the hidden variables, and $p(\beta, \Theta, Z)$ is the probability of generating those hidden variables in the first place.

However, the denominator $p(\mathbf{X})$ — the "marginal likelihood" of the user/recipe ratings — is impossible to compute, as it would require summing over all possible values of the hidden variables:

$$p(\mathbf{X}) = \int_\beta \int_\Theta \int_Z p(\mathbf{X} |\, \beta, \Theta, \mathbf{Z})\, d\beta\, d\Theta\, d\mathbf{Z} \tag{32}$$

Since computing the *exact* posterior distribution over the hidden variables is intractable, we must resort to an approximation. In modern Bayesian statistics, there are two popular families of approximate inference algorithms: Markov Chain Monte Carlo (MCMC) methods, and variational methods. The conceptual difference between the two is that MCMC finds an approximate estimate of the true posterior function, while variational inference finds the exact value of an approximation to the posterior function. In practice, variational inference converges faster than MCMC, but MCMC is usually simpler to derive and implement. In this paper, owing to the large size of the data set, we will use a flavor of variational inference called "variational EM" (variational expectation maximization).

The variational EM algorithm alternates between the following two steps:

1. **(Variational E-Step)** With the tastes $\beta$ fixed, find, for each user $i$: the most likely taste preferences $\theta_i$ and the most likely per-rating taste responsibilities $\mathbf{Z}_i = \{\mathbf{z}_{(i)n}\}_{n=1}^{N_1}$. We can do this by performing variational inference for each user $i$ separately, since our model assumes that each user and their ratings were generated independent of the other users. The e-step for DPP-LDA is identical to that of vanilla LDA.
2. **(M-Step)** Holding $\Theta$ and $\mathbf{Z}$ fixed, find the most likely tastes $\beta$. The m-step for DPP-LDA is different from the m-step for vanilla LDA because it assumes a different, DPP-based prior $p(\beta)$.

I will briefly describe each of the steps.

### 11.1. Variational E-Step

Our goal: assuming that we know the tastes $\beta$, and given the observed user/recipe ratings, we would like to infer the posterior distribution over the hidden variables for each user.

In what follows, we consider a single user and drop the $i$ subscript for convenience. Suppose that the user has rated $N$ recipes. Let $\mathbf{X} = \mathbf{x}_1 \dots \mathbf{x}_N$ be those ratings, i.e. $\mathbf{x}_n$ is an indicator vector for the $n$th rating, such that $x_{nj} = 1$ if the $n$th rating is for recipe $j$ and $x_{nj} = 0$ otherwise. Similarly, let $\mathbf{Z} = \mathbf{z}_1 \dots \mathbf{z}_N$ be indicator vectors for the (latent) taste indices that were responsible for those ratings, such that $z_{nk} = 1$ if taste $k$ generated the $n$th rating. The user's taste preferences are $\theta$, a $K$-vector that sums to 1, where $\theta_k$ represents the user's affinity for taste $k$.

We would like to compute the posterior distribution

$$p(\mathbf{Z}, \theta \,|\, \mathbf{X}) = \frac{p(\mathbf{X}|\mathbf{Z}, \theta) p(\mathbf{Z}, \theta)}{\int_{\mathbf{Z}'} \int_{\theta'} p(\mathbf{X}|\mathbf{Z}', \theta') p(\mathbf{Z}', \theta')} \tag{33}$$

but, as discussed above, we can't, due to the intractable integral in the denominator.

The main idea behind variational inference is to find a function $q(\theta, \mathbf{Z}|\nu)$ which approximates the true, intractable posterior $p(\theta, \mathbf{Z} \mid \mathbf{X})$:

$$p(\theta, \mathbf{Z} \mid \mathbf{X}) \approx q(\theta, \mathbf{Z} \mid \nu) \tag{34}$$

Once we've found a suitable "variational approximation" $q$, we can use it in place of the true posterior $p$. But how do we find a good $q$? Typically, we assume that $q$ belongs to a simplified family of functions indexed by a set of "free variational parameters" $\nu$, and then use an optimization algorithm to find the setting of the free variational parameters $\nu$ that minimizes the "distance" between $q$ and $p$. How to measure the "distance" between two probability distributions? We generally use a measure called the *KL-divergence*. The KL divergence between two probability distributions $q$ and $p$ is defined as:

$$D_{\mathrm{KL}}(q\,||\,p) = \int_{\theta}\int_{Z} q(\theta,\mathbf{Z}|\nu)\log\frac{q(\theta,\mathbf{Z}|\nu)}{p(\theta,\mathbf{Z}\mid\mathbf{X})}\,d\theta\,d\mathbf{Z} \tag{35}$$

which can be rewritten as:

$$D_{\mathrm{KL}}(q\,||\,p) = \int_{\Theta}\int_{Z} q(\theta,\mathbf{Z}|\nu)\log\frac{q(\theta,\mathbf{Z}|\nu)}{p(\theta,\mathbf{Z},\mathbf{X})}\,d\theta\,d\mathbf{Z} + \log p(\mathbf{X}) \tag{36}$$

$$= \underbrace{\mathbb{E}_q\left[q(\theta,\mathbf{Z}|\nu)\right] - \mathbb{E}_q\left[p(\theta,\mathbf{Z},\mathbf{X})\right]}_{-\mathscr{L}(\nu)} + \log p(\mathbf{X}) \tag{37}$$

$$\text{where } \mathscr{L}(\nu) = \mathbb{E}_q\left[p(\theta,\mathbf{Z},\mathbf{X})\right] - \mathbb{E}_q\left[q(\theta,\mathbf{Z}|\nu)\right] \tag{38}$$

where $\mathbb{E}_q\left[\,\cdot\,\right]$ denotes the expectation of an expression taken with respect to the probability density function $q$.

Note that since $p(\mathbf{X})$ is fixed (though unknown), maximizing the "Evidence Lower BOund" (ELBO) $\mathscr{L}(\nu)$ is equivalent to minimizing the KL divergence. Thus, maximizing $\mathscr{L}(\nu)$ w.r.t $\nu$ will yield the closest approximation $q|\nu \approx p$.

Of course, in order to make that optimization problem tractable, we must restrict $q$ to a limited family of distributions. So, we will assume that $q$ obeys the following "mean-field" factorization:

$$q(\theta,\mathbf{Z}|\nu) = q_{\theta}(\theta|\nu_{q_\theta})\prod_{n=1}^{N} q_{z_n}(\mathbf{z}_n|\nu_{q_{z_n}}) \tag{39}$$

That is, we assume that all of the hidden variables are independent of one another — which is blatantly false, since, according to our model, Tyche drew the $\mathbf{z}_n$ variables conditioned on $\theta$. However, this isn't necessarily a problem, since we will optimize $\nu$ *taking into account* the observed ratings.

We can maximize $\mathscr{L}(\nu)$ using a coordinate ascent optimization algorithm, where we update each of the variational free parameters $\nu_{q_\theta}$ and $\nu_{q_{z_1}}, \ldots, \nu_{q_{z_N}}$ in turn. Though I will not reproduce the derivation here, one can show [10] that, in general, the coordinate ascent update rules for mean-field variational inference are:

$$q_\theta(\theta|\nu_{q_\theta}) \propto \exp\left\{\mathbb{E}_{q_{z_1},\ldots,q_{z_N}}\left[\log p(\theta,\mathbf{Z},\mathbf{X})\right]\right\} \tag{40}$$

$$q_{z_n}(\mathbf{z}_n|\nu_{q_{z_n}}) \propto \exp\left\{\mathbb{E}_{q_\theta}\left[\log p(\theta,\mathbf{Z},\mathbf{X})\right]\right\} \tag{41}$$

According to our model, the joint probability (given the tastes and other fixed parameters) of the hidden and observed variables for a single user is:

$$\begin{aligned}
p(\theta,\mathbf{Z},\mathbf{X}|\alpha,\beta) &= p(\theta|\alpha)\prod_{n=1}^{N} p(\mathbf{z}_n|\theta)\,p(\mathbf{x}_n|\mathbf{z}_n,\beta) && (42)\\
&= \mathrm{Dir}(\theta|\alpha)\prod_{n=1}^{N}\mathrm{Cat}(\mathbf{z}_n|\theta)\,\mathrm{Cat}(\mathbf{x}_n|\beta_{\mathbf{z}_n}) && (43)\\
&= \left[\frac{\Gamma(\sum_k \alpha_k)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}\theta_k^{\alpha_k-1}\right]\prod_{n=1}^{N}\left[\prod_{k=1}^{K}\theta_k^{z_{nk}}\right]\left[\prod_{k=1}^{K}\prod_{j=1}^{R}\beta_{kj}^{z_{nk}x_{nj}}\right] && (44)
\end{aligned}$$

Taking the log of equation 44,

$$\begin{aligned}
\log p(\mathbf{w},\mathbf{z},\theta\,|\,\alpha,\beta) = \log\Gamma\left(\sum_k \alpha_k\right) - \sum_{k=1}^{K}(\alpha_k-1)\log\theta_k - \log\Gamma(\alpha_k)\\
+\sum_{n=1}^{N}\left[\sum_{k=1}^{K} z_{nk}\log\theta_k\right] + \left[\sum_{k=1}^{K}\sum_{j=1}^{R} z_{nk}x_{nj}\log\beta_{kj}\right] \qquad (45)
\end{aligned}$$

Applying the coordinate ascent update rules in equations 40 and 41 to the log-likelihood in equation 45, one can show that the variational distribution $q_\theta$ is a Dirichlet, while the variational distributions

$q_{z_n}$ are multinomial

$$q_\theta(\theta) = \text{Dirichlet}(\theta \mid \gamma) \tag{46}$$

$$q_{z_n}(\mathbf{z}_n) = \text{Multinomial}(z_n \mid \phi_n) \tag{47}$$

and that the update rules are:

$$\gamma_k := \alpha_k + \sum_{n=1}^{N} \phi_{nk} \tag{48}$$

$$\phi_{nk} :\propto \beta_{kv_n} \exp\{(\psi(\gamma_k))\} \tag{49}$$

where $:\propto$ means to normalize each $\phi_n$ to sum to 1, and $\psi(\,\cdot\,)$ is the digamma function. This concludes the derivation of the variational e-step for DPP-LDA (which is equivalent to the variational e-step for LDA). For each user $i$, to infer the posterior distribution over the hidden variables $p(\theta_i, \mathbf{Z}_i | \mathbf{X}_i, \beta)$, one need only iterate between the two update rules in equations 48 and 49 for several iterations, then use the variational approximation $q(\theta_i, \mathbf{Z}_i | \nu_i) = q(\theta_i | \gamma_i) \prod_{n=1}^{N} q(\mathbf{z}_{(i)n} | \phi_{(i)n})$ as a surrogate for the posterior $p$.

## 11.2. M-Step

In the m-step, we assume that we have already inferred the posterior distribution over each of the hidden variables $\theta_i$ and $\mathbf{Z}_i$ for each user $i = \{1, 2, \ldots U\}$, and our goal is now to find the "most likely" tastes $\beta$. The "most likely" $\beta$ are those that maximize the expected log-likelihood of the entire data set [2], with the expectation taken with respect to the variational approximation $q$.[4]

$$\hat{\beta} = \text{argmax}_\beta \, \mathbb{E}_q \left[ p(\beta, \Theta, Z, X) \right] \tag{50}$$

$$= \text{argmax}_\beta \, \mathbb{E}_q \left[ \log p(\beta, \Theta, Z, X) \right] \tag{51}$$

[4]In the original LDA paper, Blei uses a different objective function for the m-step of variational EM which does not take into account the Dirichlet prior on $\beta$.

According to our model, the likelihood of the entire data set (tastes + all users) is:

$$p(\beta, \mathbf{x}, \mathbf{z}, \theta \,|\, \alpha) = p(\beta) \prod_{i=1}^{U} \mathrm{Dir}(\theta_i | \alpha) \prod_{n=1}^{N} \mathrm{Cat}(\mathbf{z}_{in} | \theta)\, \mathrm{Cat}(\mathbf{x}_{in} | \beta_{\mathbf{z}_n}) \tag{52}$$

$$= p(\beta) \prod_{i=1}^{U} \left[ \frac{\Gamma(\sum_k \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_{ik}^{\alpha_k - 1} \right] \prod_{n=1}^{N} \left[ \prod_{k=1}^{K} \theta_{ik}^{z_{(i)nk}} \right] \left[ \prod_{k=1}^{K} \prod_{j=1}^{R} \beta_{kj}^{z_{(i)nk} x_{nj}} \right] \tag{53}$$

Taking the log,

$$\log p(\beta, \mathbf{x}, \mathbf{z}, \theta \,|\, \alpha) = \log p(\beta) + \sum_{i=1}^{U} \log \Gamma \left( \sum_k \alpha_k \right) - \sum_{k=1}^{K} (\alpha_k - 1) \log \theta_{(i)k} - \log \Gamma(\alpha_k) +$$

$$\sum_{n=1}^{N} \left[ \sum_{k=1}^{K} z_{(i)nk} \log \theta_{(i)k} + \sum_{j=1}^{R} z_{(i)nk} x_{(i)nj} \log \beta_{kj} \right] \tag{54}$$

We take the expectation with respect to $q$ and ignore all terms that do not depend on $\beta$:

$$\mathbb{E}_q \left[ \log p(\beta, \mathbf{x}, \mathbf{z}, \theta, | \alpha) \right] \propto \log p(\beta) + \sum_{i=1}^{U} \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{j=1}^{R} \mathbb{E}_{q_{z_n}} \left[ z_{(i)nk} \right] x_{(i)nj} \log \beta_{kj} \tag{55}$$

$$\propto \log p(\beta) + \sum_{i=1}^{U} \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{j=1}^{R} \phi_{(i)nk} x_{(i)nj} \log \beta_{kj} \tag{56}$$

In standard LDA, $p(\beta)$ is the product of $K$ independent distributions $\sim$ Dirichlet$(\beta_k | \eta)$, and maximizing equation 55 w.r.t $\beta$ is simple: take the gradient w.r.t $\beta$, set it to zero, and solve. In DPP-LDA, however, the DPP prior on the tastes does not permit such a simple closed-form solution:

$$p(\beta) = \mathrm{DPP}(\beta | L) \tag{57}$$

$$\propto \det(L_\beta) \tag{58}$$

$$\text{where } L_\beta = \begin{bmatrix} L(\beta^{(1)}, \beta^{(1)}) & \dots & L(\beta^{(1)}, \beta^{(G)}) \\ \vdots & \ddots & \vdots \\ L(\beta^{(G)}, \beta^{(1)}) & \dots & L(\beta^{(G)}, \beta^{(G)}) \end{bmatrix} \tag{59}$$

How can we maximize equation 55 with respect to $\beta$ then? Following Zou and Adams, we chose to use *gradient ascent*. Gradient ascent is a simple optimization algorithm used widely in machine learning. The idea is an intuitive one: suppose you are a mountaineer vacationing in the Rockies, and you want to find a peak. One solution is to always walk in the direction where the slope of the mountain is highest. This procedure is guaranteed to eventually take you to a peak (though not necessarily the *highest* peak in the mountain range).

Likewise, if we want to optimize a function of many variables, we can just take steps in the direction of the gradient. (Recall that the *gradient* of a function of many variables is the vector of partial derivatives with respect to each variable.) So long as our step sizes satisfy certain conditions (they are decreasing, etc.), we are guaranteed to eventually reach a local maximum.

Here I will derive a gradient ascent optimization algorithm for the m-step of DPP-LDA inference. Our goal is to solve the following constrained optimization problem:

$$\begin{aligned}\hat{\beta} &= \operatorname{argmax}_{\beta}\, \mathbb{E}_q\left[\log p(\beta, \mathbf{x}, \mathbf{z}, \theta, \mid \alpha)\right] && (60)\\ &= \operatorname{argmax}_{\beta}\left[\log p(\beta) + \sum_{i=1}^{U}\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{j=1}^{R} \phi_{(i)nk}\, x_{(i)nj} \log \beta_{kj}\right] && (61)\end{aligned}$$

subject to the constraint

$$\sum_{j=1}^{R} \beta_{kj} = 1 \text{ for each } k = \{1 \ldots K\} \tag{62}$$

This constraint ensures that each taste is a valid probability distribution over the recipes. In the m-step for standard LDA, one can enforce the constraint in equation 62 by introducing Lagrange multipliers. Here, we will instead reparameterize this constrained optimization problem over $\beta$ into an unconstrained optimization problem over a new set of variables $\mathbf{w}$.

Let there be a one-to-one correspondence between unconstrained variables $w_{mj}^{(g)}$ and constrained variables $\beta_{mj}^{(g)}$. The relationship is as follows:

$$\beta_{mj}^{(g)} = \frac{\exp\left\{w_{mj}^{(g)}\right\}}{\sum_{j'=1}^{R} \exp\left\{w_{mj'}^{(g)}\right\}} \tag{63}$$

That is, $\beta_m^{(g)}$ is the so-called "softmax" function over $\mathbf{w}_m^{(g)}$. Reformulating with softmax is a common strategy for dealing with optimization problems where a set of variables must sum to one [3]. No matter what the variables $w_{m1}^{(g)} \dots w_{mV}^{(g)}$ are, each variable $\beta_{mj}^{(g)}$ will lie between zero and one. You can verify this fact from looking at equation 63 — the numerator and denominator of the RHS will always be positive, and the numerator is by necessity always less than the denominator, so the fraction will always be between 0 and 1. Therefore, we can optimize our objective function with respect to $\mathbf{w}$ without regard for the constraints; then, at the end, we can just compute $\hat{\beta}$ from $\hat{\mathbf{w}}$ to yield a local maximum of the objective function which still respects the constraint in equation 62. The gradient ascent update will be:

$$w_{mj}^{(g)} \leftarrow w_{mj}^{(g)} + \tau \frac{\partial\, \mathbb{E}_q\left[\log p(\beta, \mathbf{x}, \mathbf{z}, \theta \,|\, \alpha)\right]}{\partial\, w_{mj}^{(g)}} \tag{64}$$

where $\tau$ is the "step size."

We can use the chain rule to relate the derivative of the objective function w.r.t $w$ to the derivative of the objective function w.r.t $\beta$:

$$\frac{\partial\, \mathbb{E}_q\left[\log p(\beta, \mathbf{x}, \mathbf{z}, \theta \,|\, \alpha)\right]}{\partial\, w_{mj}^{(g)}} = \frac{\partial\, \mathbb{E}_q\left[\log p(\beta, \mathbf{x}, \mathbf{z}, \theta \,|\, \alpha)\right]}{\partial\, \beta_{mj}^{(g)}} \frac{\partial \beta_{mj}^{(g)}}{\partial w_{mj}^{(g)}} \tag{65}$$

where $\lambda$ is a hyperparameter that we can tune to make the DPP prior arbitrarily strong. (According to the generative story, increasing $\lambda$ is the same as replicating the DPP prior multiple times.)

where, by the definition of $\beta_{mj}^{(g)}$ in equation 63,

$$\frac{\partial \beta_{mj}^{(g)}}{\partial w_{mj}^{(g)}} = \frac{\exp\left\{w_{mj}^{(g)}\right\}\left(\sum_{j'\neq j}\exp\left\{w_{mj'}^{(g)}\right\}\right)}{\left(\sum_{j'=1}^{V}\exp\left\{w_{mj'}^{(g)}\right\}\right)^2} \tag{66}$$

From equation 60, the derivative of the objective function w.r.t $\beta_{mj}^{(g)}$ is given by:

$$\frac{\partial\, \mathbb{E}_q\left[\log p(\beta, \mathbf{x}, \mathbf{z}, \theta \,|\, \alpha)\right]}{\partial \beta_{mj}^{(g)}} = \lambda \frac{\partial \log\det(L_\beta)}{\partial \beta_{mj}^{(g)}} + \frac{1}{\beta_{mj}^{(g)}} \sum_{i=1}^{U}\sum_{n=1}^{N} \phi_{(i)nk}^{(g)} x_{(i)nj} \tag{67}$$

We need to compute $\frac{\partial \log\det(L_\beta)}{\partial \beta_{mj}^{(g)}}$, the derivative of the DPP prior with respect to $\beta_{mj}^{(g)}$.
Recall that the L-ensemble kernel function $L\left(\beta^{(g_1)}, \beta^{(g_2)}\right)$ was defined in equation 23 as a combination of a correlation function $S$ and a quality function $\pi$:

$$L\left(\beta^{(g_1)}, \beta^{(g_2)}\right) = S\left(\beta^{(g_1)}, \beta^{(g_2)}\right)\sqrt{\pi\left(\beta^{(g_1)}\right)\pi\left(\beta^{(g_2)}\right)}$$

In matrix form, we can write:

$$L_\beta = \Pi S_\beta \Pi \tag{68}$$

$$\text{where } \Pi = \operatorname{diag}\left(\sqrt{\pi\left(\beta^{(1)}\right)}, \ldots, \sqrt{\pi\left(\beta^{(G)}\right)}\right) \tag{69}$$

Since, for any matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\det(\mathbf{ABC}) = \det(\mathbf{A})\det(\mathbf{B})\det(\mathbf{C})$, we have:

$$\det(L_\beta) = \det(\Pi)\det(S_\beta)\det(\Pi) \tag{70}$$

$$= \det(S_\beta)\det(\Pi)^2 \tag{71}$$

Since the determinant of a diagonal matrix is the product of the elements on the diagonal, we have:

$$\det(\Pi) = \prod_{g=1}^{G}\sqrt{\pi\left(\beta^{(g)}\right)}$$

and therefore:

$$\det(L_\beta) = \det(S_\beta) \prod_{g=1}^{G} \pi\left(\beta^{(g)}\right) \tag{72}$$

$$\begin{aligned}
\log\det(L_\beta) &= \log\det(S_\beta) + \sum_{g=1}^{G} \log\pi\left(\beta^{(g)}\right) \\
&= \log\det(S_\beta) + \sum_{g=1}^{G} \log\left[\prod_{m=1}^{M_g} \text{Dirichlet}\left(\beta_m^{(g)} | \eta\right)\right] \\
&= \log\det(S_\beta) + \sum_{g=1}^{G} \log\left[\prod_{m=1}^{M_g} \frac{\Gamma(\eta R)}{\Gamma(\eta)^R} \prod_{j=1}^{R} \beta_{mj}^{(g)\,\eta-1}\right] \\
&= \log\det(S_\beta) + \sum_{g=1}^{G} \sum_{m=1}^{M_g} \sum_{j=1}^{R} (\eta-1)\log\beta_{mj}^{(g)} + \text{constant}
\end{aligned} \tag{73}$$

$$\frac{\partial \log\det(L_\beta)}{\partial \beta_{mj}^{(g)}} = \frac{\partial \log\det(S_\beta)}{\partial \beta_{mj}^{(g)}} + \frac{\eta-1}{\beta_{mj}^{(g)}} \tag{74}$$

Combining equations 67 and 74, the derivative of the objective function with respect to the parameter $\beta_{mj}^{(g)}$ is given by:

$$\frac{\partial\, \mathbb{E}_q\left[\log p(\beta, \mathbf{x}, \mathbf{z}, \theta \,|\, \alpha)\right]}{\partial \beta_{mj}^{(g)}} = \lambda \frac{\partial \log\det(S_\beta)}{\partial \beta_j^{(g)}} + \frac{1}{\beta_{mj}^{(g)}} \left(\lambda(\eta-1) + \sum_{i=1}^{U} \sum_{n=1}^{N} \phi_{(i)nm}^{(g)} x_{(i)nj}\right) \tag{75}$$

All that remains is to compute the derivative of $\log\det(S_\beta)$ with respect to $\beta_{mj}^{(g)}$. It is a simple result from matrix calculus that, for any matrix $\mathbf{M}$, $\partial \log\det(\mathbf{M}) = \text{trace}(\mathbf{M}^{-1}\partial\mathbf{M})$.[12] Therefore,

$$\frac{\partial}{\partial \beta_{mj}^{(g)}} \log\det(S_\beta) = \text{trace}\left(S_\beta^{-1} \frac{\partial S_\beta}{\partial \beta_{mj}^{(g)}}\right) \tag{76}$$

So, in order to compute the derivative of the DPP prior w.r.t $\beta_{mj}^{(g)}$, we'll have to: (a) invert the $G \times G$ matrix $S_\beta$ formed by applying the correlation function $S$ to every pair of taste groups; and (b) compute the derivative of every entry in $S_\beta$ w.r.t $\beta_{mj}^{(g)}$. Recall that the $(g_1, g_2)$ entry in $S_\beta$ is just defined as $S\left(\beta^{(g_1)}, \beta^{(g_2)}\right)$. The derivative of $S\left(\beta^{(g_1)}, \beta^{(g_2)}\right)$ with respect to $\beta_{mj}^{(g)}$ is obviously zero if $g \neq g_1 \neq g_2$. Therefore, the $G \times G$ matrix $\frac{\partial S_\beta}{\partial \beta^{(g)}_{mj}}$ is zero everywhere except in row $g$ (where $g_1 = g$) and in column $g$ (where $g_2 = g$). Recall that $S_\beta$ is symmetric, so $\frac{\partial S_\beta}{\partial \beta^{(g)}_{mj}}$ must be symmetric too. Therefore, without loss of generality, we need only compute the derivatives along the $g$th row $\frac{(\partial S_\beta)_{gi}}{\partial \beta^{(g)}_{mj}}$, where $i$ ranges from 1 to $G$. The derivatives along the $g$th column $\frac{(\partial S_\beta)_{ig}}{\partial \beta^{(g)}_{mj}}$ will be identical.

The derivative of the similarity between taste groups $g$ and $i$ with respect to $\beta^{(g)}_{mj}$ is:

$$\frac{(\partial S_\beta)_{gi}}{\partial \beta^{(g)}_{mj}} = \frac{\partial S(\beta^{(g)}, \beta^{(i)})}{\partial \beta^{(g)}_{mj}} \tag{77}$$

$$= \frac{\partial R(\Lambda^{(g)}, \Lambda^{(i)})}{\partial \beta^{(g)}_{mj}} \tag{78}$$

since we defined the similarity $S$ between two taste groups $\beta^{(g)}$ and $\beta^{(i)}$ as the similarity $R$ between their recipe-similarity matrices $\Lambda^{(g)}$ and $\Lambda^{(i)}$.

By the multivariate chain rule,

$$= \sum_{j_1=1}^{R} \sum_{j_2=1}^{R} \frac{\partial R(\Lambda^{(g)}, \Lambda^{(i)})}{\partial \Lambda^{(g)}_{j_1 j_2}} \frac{\partial \Lambda^{(g)}_{j_1 j_2}}{\partial \beta^{(g)}_{mj}} \tag{79}$$

Since the recipe similarity matrix in taste group $g$ is defined as $\Lambda^{(g)} = \beta^{(g)^T} \beta^{(g)}$, we have that the similarity between recipes $j_1$ and $j_2$ in taste group $g$ is defined as the dot product between the $j_1$st column of $\beta^{(g)}$ and the $j_2$nd column of $\beta^{(g)}$:

$$\Lambda^{(g)}_{j_1 j_2} = \sum_{m=1}^{M_g} \beta^{(g)}_{mj_1} \beta^{(g)}_{mj_2} \implies \frac{\Lambda^{(g)}_{j_1 j_2}}{\partial \beta^{(g)}_{mj}} = \begin{cases} 0 & \text{if } j \neq j_1 \neq j_2 \\ 2\beta^{(g)}_{mj} & \text{if } j = j_1 = j_2 \\ \beta^{(g)}_{mj_2} & \text{else if } j = j_1 \\ \beta^{(g)}_{mj_1} & \text{else if } j = j_2 \end{cases} \tag{80}$$

Therefore, equation 79 for the derivative of the similarity between taste groups $g$ and $i$ with respect to $\beta^{(g)}_{mj}$ reduces to:

$$\frac{(\partial S_\beta)_{gi}}{\partial \beta_{mj}^{(g)}} = 2 \sum_{j'=1}^{R} \frac{\partial R(\Lambda^{(g)}, \Lambda^{(i)})}{\partial \Lambda_{jj'}^{(g)}} \beta_{mj'}^{(g)} \tag{81}$$

From the definition of $R$ in equation ??? and a lot of horrible algebra, one can show that the derivative of $R\left(\Lambda^{(g)}, \Lambda^{(i)}\right)$, the similarity function between the recipe-similarity matrices for taste group $g$ and $i$, with respect to $\Lambda_{jj'}^{(g)}$, the recipe-similarity between recipes $j$ and $j'$ in taste group $g$, is:

$$\frac{\partial R(\Lambda^{(g)}, \Lambda^{(i)})}{\partial \Lambda_{jj'}^{(g)}} = \frac{p\Lambda_{j_1 j_2}^{(g)^{p-1}} \left[\Lambda_{j_1 j_2}^{(j)^p} - \Lambda_{j_1 j_2}^{(g)^p} \left(\sum_{j_1=1}^{R} \sum_{j_2=1}^{R} (\Lambda_{j_1 j_2}^{(g)} \Lambda_{j_1 j_2}^{(j)})^p\right) \left(\sum_{j_1=1}^{V} \sum_{j_2=1}^{R} \Lambda_{j_1 j_2}^{(g)^{2p}}\right)^{-1}\right]}{(\sum_{j_1=1}^{R} \sum_{j_2=1}^{R} \Lambda_{j_1 j_2}^{(g)^{2p}})^{\frac{1}{2}} (\sum_{j_1=1}^{R} \sum_{j_2=1}^{V} \Lambda_{j_1 j_2}^{(j)^{2p}})^{\frac{1}{2}}} \tag{82}$$

**Algorithm**

The complete algorithm for the m-step of variational EM for DPP-LDA is as follows:

**Repeat until convergence:**

1. Compute the $G \times G$ matrix $S_\beta$ by applying $S$ to every pair of taste groups.
2. Compute the inverse $S_\beta^{-1}$.
3. For each taste group $g = \{1, 2, \ldots, G\}$:
   (a) For each taste $m = \{1, 2, \ldots, M_g\}$ in group $g$:
      i. For each recipe $j = \{1, 2, \ldots, R\}$, compute the derivative of the objective function $\mathbb{E}_q[\log p(\beta, \mathbf{x}, \mathbf{z}, \theta \,|\, \alpha)]$ w.r.t $\beta_{mj}^{(g)}$ as follows:
         A. Compute the $G \times G$ matrix $\frac{\partial S_\beta}{\partial \beta_{mj}^{(g)}}$, which is all zeros except for the $g$th row and $g$th column. The entries $\frac{\partial (S_\beta)_{gi}}{\partial \beta_{mj}^{(g)}} = \frac{\partial (S_\beta)_{ig}}{\partial \beta_{mj}^{(g)}}$ for every $i = 1 \ldots G$ can be computed using equations 82 and 81.
         B. Compute $\frac{\partial}{\partial \beta_{mj}^{(g)}} \log\det(S_\beta) = \text{trace}\left(S_\beta^{-1} \frac{\partial S_\beta}{\partial \beta_{mj}^{(g)}}\right)$, as we saw in equation 76.
         C. Compute the derivative of the objective function with respect to $\beta_{mj}^{(g)}$ using equation

**??**.

D. Compute the derivative of the objective function w.r.t $w_{mj}^{(g)}$ using equations 65 and 66.

4. For every $g$, $k$, $j$, apply the gradient ascent update to $w_{mj}^{(g)}$:

$$w_{mj}^{(g)} \leftarrow w_{mj}^{(g)} + \tau \frac{\partial\, \mathbb{E}_q \left[\log p(\beta, \mathbf{x}, \mathbf{z}, \theta \,|\, \alpha)\right]}{\partial\, w_{mj}^{(g)}}$$

**Implementation Details**

The algorithm above was implemented in C. The GSL scientific library was used for matrix operations and probability density functions. Most arithmetic was performed in log-space to avoid numerical underflow. Terms which appear in multiple components of the gradient were pre-computed and stored. In each m-step, gradient ascent was carried out for a fixed number of iterations with a constant step size $\tau$.

**Complexity**

**Theorem.** Assume for simplicity that all taste groups have the same number of tastes $\kappa$. Then the big-O cost of *each iteration* of gradient descent is $\mathcal{O}(G\kappa R^2)$.

**Proof.** Computing the "recipe-similarity matrix" (the Gram matrix) for each taste group is $\mathcal{O}(\kappa R^2)$, so computing the recipe-similarity matrices for *every* taste group is $\mathcal{O}(G\kappa R^2)$. Assuming typical values of $G$, $\kappa$, and $R$ (with $G < \kappa < R$), that operation will dominate the complexity.

Other significant operations are:

1. Computing $S_\beta$, the $G \times G$ correlation matrix between the taste groups, is $\mathcal{O}(G^2R^2)$.
2. Computing $\frac{\partial R(\Lambda^{(g)}, \Lambda^{(i)})}{\partial \Lambda_{j_1 j_2}^{(g)}}$ for each $g$, $i$, $j_1$, $j_2$ is also $\mathcal{O}(G^2R^2)$.
3. Forming every derivative of the taste group correlation matrix costs $\mathcal{O}(G^2\kappa R)$
4. Multiplying the inverse of the correlation matrix with each derivative of the correlation matrix is $\mathcal{O}(G^3)$, so multiplying the inverse of the correlation matrix with *every* derivative of the correlation matrix is $\mathcal{O}(G^4\kappa R)$

## 12. Semi-Supervision

As discussed above, we would like to add some form of semi-supervision to the algorithm in order to “guide” the learned taste groups and tastes towards ones that will correspond to known categories and classes of recipes. For each recipe $j$ in some labeled subset of the recipes $\mathbb{L} \subseteq \{1,2,\ldots,R\}$, we would like to specify, for each taste group $g$, a taste $m$ which is strongly exemplified by recipe $j$. We say that $y_{jm}^{(g)} = 1$ if recipe $j$ exemplifies taste $m$ in taste group $g$, and $y_{jm}^{(g)} = 0$ otherwise. For example, in the taste group for “cooking method,” the recipe “baby back ribs” strongly exemplifies the taste “barbeque.” Thus, we would set $y_{\text{ribs,barbeque}}^{\text{method}} = 1$.

How should we incorporate semi-supervision into the DPP-LDA model? Our first idea was to add a new layer of observed variables to the generative model. For each labeled recipe $j \in \mathbb{L}$, we would observe a categorical random variable $\mathbf{y}_j^{(g)}$ (represented as an indicator vector of length $M_g$), where the probability of drawing taste $m$ would be proportional to $\beta_{mj}^{(g)}$, the prominence of recipe $j$ in taste $(g,m)$.

$$\mathbf{y}_j^{(g)} \sim \text{Categorical}(\phi_j^{(g)}) \tag{83}$$

where $\phi_{jm}^{(g)}$ is the probability of drawing $y_{jm}^{(g)} = 1$. Of course, the class-wise probabilities that parameterize a categorical distribution must some to 1, so we must ensure that $\phi_j^{(g)}$ is properly normalized, while still preserving the property that $\phi_{jm}^{(g)}$ is high if $\beta_{mj}^{(g)}$ is high.

There are two obvious options:

$$\phi_{jm}^{(g)} = \frac{\beta_{mj}^{(g)}}{\sum_{j'=1}^{R} \beta_{mj'}^{(g)}} \tag{84}$$

and:

$$\phi_{jm}^{(g)} = \frac{e^{\beta_{mj}^{(g)}}}{\sum_{j'=1}^{R} e^{\beta_{mj'}^{(g)}}} \tag{85}$$

Unfortunately, neither option worked well in practice, as both models made the objective function in the m-step of variational EM too non-convex to solve with gradient descent. Therefore, we sought a model that would allow us to incorporate the semi-supervised information in a way that would still preserve the closed-form solution to the m-step. Following previous work on incorporating assumptions about topics into LDA [4], we decided to "guide" LDA by putting an *asymmetric* Dirichlet prior on the tastes.

Recall that the "quality function" $\pi(\beta_m^{(g)})$ component of our $L$-ensemble kernel $L$, which expresses our marginal prior preference for taste $\beta_m^{(g)}$, took the form of a *symmetric* Dirichlet:

$$\pi\left(\beta_m^{(g)}\right) = \text{Dirichlet}(\beta_m^{(g)}|\eta) \tag{86}$$

$$= \frac{\Gamma\left(\sum_{j=1}^{R}\eta\right)}{\prod_{j=1}^{R}\Gamma(\eta)}\prod_{j=1}^{R}\left(\beta_{mj}^{(g)}\right)^{\eta-1} \tag{87}$$

where $\eta$ is a scalar. An $R$-vector drawn from a symmetric Dirichlet will be equally likely to have probability mass in any position $j \in \{1 \ldots R\}$. On the other hand, in an *asymmetric* Dirichlet, the parameter $\eta$ is a vector rather than a scalar, and if $\eta_j$ is high, a vector drawn from the Dirichlet will be likely to have a high value in position $j$. The density function of an asymmetric Dirichlet distribution with parameter vector $\eta_m^{(g)}$ is:

$$\pi\left(\beta_m^{(g)}\right) = \text{Dirichlet}(\beta_m^{(g)}|\eta_m^{(g)}) \tag{88}$$

$$= \frac{\Gamma\left(\sum_{j=1}^{R}\eta_j\right)}{\prod_{j=1}^{R}\Gamma(\eta_{mj}^{(g)})}\prod_{j=1}^{R}\left(\beta_{mj}^{(g)}\right)^{\eta_{mj}^{(g)}-1} \tag{89}$$

So, if recipe $j$ is labeled (i.e. $j \in \mathbb{L}$) with taste $m$ in group $g$ (i.e. $y_{jm}^{(g)} = 1$), then we want taste $\beta_m^{(g)}$ to place a high probability mass on recipe $j$, so we can just set $\eta_{mj}^{(g)}$ to be a high number.

We will use the following rule to set the Dirichlet parameter vectors $\eta_m^{(g)}$:

$$\eta_{mj}^{(g)} = \begin{cases} \delta\gamma & \text{if } y_{jm}^{(g)} = 1 \\ \delta & \text{if } y_{jm}^{(g)} = 0 \end{cases} \tag{90}$$

Thus, $\delta$ controls the overall concentration of each taste distribution; if $\delta$ is high, the probability mass of each taste is likely to be concentrated in just a few recipes; if $\delta$ is low, the probability mass will be more spread out evenly. And $\gamma$ is a knob that we can turn to adjust the strength of the semi-supervision.

### 12.1. Fitting the model

How does semi-supervision affect the inference algorithm for S-DPP-LDA? Barely at all, as it turns out. Semi-supervision only alters the prior on tastes $p(\beta)$, so the e-step remains unchanged. In the m-step, equation 75 becomes:

$$\frac{\partial\, \mathbb{E}_q\left[\log p(\beta, \mathbf{x}, \mathbf{z}, \theta \,|\, \alpha)\right]}{\partial \beta_{mj}^{(g)}} = \lambda \frac{\partial \log \det(S_\beta)}{\partial \beta_{mj}^{(g)}} + \frac{1}{\beta_{mj}^{(g)}} \left( \eta_{mj}^{(g)} - 1 + \sum_{i=1}^{U} \sum_{n=1}^{N} \phi_{(i)nm}^{(g)} x_{(i)nj} \right) \tag{91}$$

where $\eta_{mj}^{(g)}$ is as defined above in equation 90.

## 13. Experiments

### 13.1. Data Set

The data set was graciously provided by the cooking website AllRecipes.com. It consists of the favorite recipes of $U = 253,283$ users. The total number of recipes is $R = 10,217$. All users collectively selected $\sum_{i=i}^{U} N_i = 27,832,693$ favorite recipes. [5]

[5] The full dataset that AllRecipes furnished me actually consisted of $U = 8,898,669$ users, $R = 74,508$ recipes, and $\sum_i N_i = 291,925,038$ total favorites. In preprocessing, I kept only those recipes that had either at least 50,000 favorites from users, or 3,000 favorites and a label in at least one dimension. I kept only those users who had favorited 50 or more of the remaining recipes.

Additionally, the original data set did not have the exact heirarchy of dimensions and classes that I present here. Instead, it had each recipe labeled with one or more *ad keys*. I grouped together ad keys into semantically meaningful classes and dimensions.

| Dimension | Classes |
|---|---|
| cuisine | asian, european, italian, jewish, mexican, southern, american |
| course | appetizer, beverage, bread, condiment, dessert, main dish, salad, sauce, snack, soup |
| method | baked, barbeque, boiled, fried, simmered, frozen |
| dietary | vegetarian, fish, poultry, meat |

**Figure 1: The dimensions and classes from the AllRecipes dataset**

Each recipe is also labeled with a *class* in up to $G = 4$ dimensions. The dimensions are: cuisine, course, method, and dietary. The available classes in each dimension are listed in figure ??. Classes were assigned to recipes by hand by a team at AllRecipes responsible for maintaining the company's content taxonomy.

We will carry out the following experiments:

1. We qualitatively assess the performance of vanilla, unsupervised LDA on the recipes data set, in order to supply us with a baseline level of performance with which we can compare the performance of other methods. Specifically, we look at the coherence and interpretability of the learned tastes.
2. We qualitatively assess whether unsupervised DPP-LDA succeeds in learning taste groups that model a "diverse" set of dimensions.
3. We quantitatively and qualitatively assess whether semi-supervised S-DPP-LDA is able to learn the dimensions (= taste groups) and classes (= tastes) from the AllRecipes content taxonomy that we "seed" it with.

## 13.2. Unsupervised LDA Baseline

We ran LDA using Blei's original `lda-c` code available on Github. We set $K = 10$ tastes, and we fit the model for 30 iterations of variational EM. In supplementary table 1, we visualize the tastes learned by LDA by printing the top five recipes in each taste. LDA performs moderately successfully on this dataset: is learns tastes for "vegan" (taste 2), "chicken" (3), and "casserole" (8).

## 13.3. Unsupervised DPP-LDA

We ran DPP-LDA using the following settings:

| setting | value |
|---:|:---|
| similarity function parameter $p$ | 0.5 |
| diversity strength $\lambda$ | $1 \times 10^7$ |
| gradient ascent step size | 10 |
| gradient ascent num iterations | 5 |
| # variational EM iters | 6 |
| number of taste groups $G$ | 4 |
| number of tastes in group 1 $M_1$ | 6 |
| $M_2$ | 10 |
| $M_3$ | 4 |
| $M_4$ | 7 |

We used 4 taste groups of sizes $\{4, 6, 10, 10\}$ in order to facilitate comparison with the semi-supervised variant. In supplementary table 2, we visualize the learned tastes by printing out the top five recipes in each taste. We observe that DPP-LDA learns a few interesting tastes, such as fudge (group1, taste 1), pasta (group 2, taste 9), and Mexican (group 3, taste 3). However, the majority of tastes that were learned appear to be mostly noise, and no relationship between the different taste groups is immediately obvious.

## 13.4. Semi-Supervised DPP LDA

To evaluate our algorithm for semi-supervised multi-dimensional taste modeling, we withheld a random half of the labels, and tried to see whether S-DPP-LDA could learn a set of tastes that would allow us to predict the held-out labels. The training data set consisted of $10,842$ labels, with an additional $10,842$ held out. (If every recipe had been fully labeled with one class in every dimension, there would have been $R \times G = 10,217 \times 4 = 40,868$ labels in the full data set, but instead there were 21,684.)

We ran S-DPP-LDA using the following settings:

| setting | value |
|---:|:---|
| similarity function parameter $p$ | 0.5 |
| diversity strength $\lambda$ | $1 \times 10^7$ |
| gradient ascent step size | 100 |
| gradient ascent num iters | 5 |
| semi-supervision strength $\gamma$ | $5 \times 10^3$ |
| Dirichlet concentration $\delta$ | 1 |
| # variational EM iterations | 6 |

We can qualitatively evaluate the learned tastes by printing the top five recipes in each taste (see supplementary table 3). Unfortunately, the semi-supervision does not appear to have had much of an effect: very few of the learned tastes actually correspond to their seeds.

We can quantitatively evaluate the impact of semi-supervision by trying to predict the held-out labels for each recipe $j$ and group $g$ by using the following prediction rule:

$$\hat{y}_j^{(g)} = \text{argmax}_m \, \beta_{mj}^{(g)} \tag{92}$$

This prediction rule achieves an accuracy of 0.2096. In contrast, the baseline prediction rule "guess randomly" :

$$\hat{y}_j^{(g)} = \text{random-uniform}(1, M_g) \tag{93}$$

onlys achieve an accuracy of 0.1495.

Thus, S-DPP-LDA learns a multi-dimensional representation of each recipe that holds some predictive value.

## 14. Conclusion

In this paper, I proposed two multi-dimensional topic models — unsupervised DPP-LDA and semi-supervised S-DPP-LDA which employ a novel determinantal point process prior over the set topic groups in order to encourage the topic groups to represent different dimensions, or aspects, of the data.

Unfortunately, neither unsupervised DPP-LDA nor semi-supervised S-DPP-LDA achieved high levels of qualitative interpretability nor quantitative accuracy in teasing out multiple dimensions of tastes from the recipe dataset. The problem could be any of the following:

1. My model could work, but I chose poor hyperparameters.
2. The Gram matrix of a taste group is a poor proxy for the "dimension" of the data that the taste group models.

3. The DPP-based objective function that I'm trying to maximize in the m-step is too unstable.
4. The recipe data set simply does not contain "multiple dimensions" of data, and no multi-dimensional topic modeling algorithm would ever be able to find any.

In conclusion, we need more experiments on alternative datasets, such as corpuses of text data, in order to ascertain whether a determinantal point process priors could play a role in multi-dimensional topic models.

## 15. Supplementary Tables

### 15.1. Supplementary Table 1: 10-topic LDA

| Taste 1 | Taste 2 | Taste 3 |
|---|---|---|
| downeast-maine-pumpkin-bread | addictive-sweet-potato-burritos | baked-honey-mustard-chicken |
| eggplant-parmesan-ii | grandmas-slow-cooker-vegetarian-chili | a-good-easy-garlic-chicken |
| jamies-cranberry-spinach-salad | tofu-parmigiana | easy-and-delicious-chicken |
| grilled-salmon-i | bean-quesadillas | awesome-slow-cooker-pot-roast |
| guacamole | seven-layer-tortilla-pie | pork-chops-for-the-slow-cooker |

| Taste 4 | Taste 5 | Taste 6 |
|---|---|---|
| shrimp-scampi-bake | mouth-watering-stuffed-mushrooms | best-ever-meat-loaf |
| marinated-grilled-shrimp | awesome-sausage-apple-and-cranberry-stuffing | famous-pork-chops |
| chicken-marsala | sugar-coated-pecans | unbelievable-chicken |
| basil-shrimp | antipasto-pasta-salad | tacos-in-pasta-shells |
| braised-balsamic-chicken | sweet-potato-casserole-ii | easter-breakfast-casserole |

| Taste 7 | Taste 8 | Taste 9 |
|---|---|---|
| awesome-slow-cooker-pot-roast | easy-and-delicious-chicken-and-rice-casserole | bread-machine-garlic-bread |
| worlds-best-lasagna | campbells-green-bean-casserole | chicken-satay |
| slow-cooker-chicken-and-dumplings | broccoli-chicken-casserole-i | gingered-chicken-breast |
| chicken-pot-pie-ix | slow-cooker-dump-and-go-cheesy-chicken | easy-banana-bread |
| chicken-cordon-bleu-ii | porcupines | italian-style-spaghetti-squash |

| Taste 10 |
|---|
| baked-dijon-salmon |
| baked-seafood-au-gratin |
| cajun-seafood-pasta |
| angel-hair-pasta-with-shrimp-and-basil |
| jays-jerk-chicken |

## 15.2. Supplementary Table 2: Unsupervised DPP-LDA

| Group 1: Taste 1 | Group 1: Taste 2 | Group 1: Taste 3 |
|---|---|---|
| irish-cream-truffle-fudge | overnight-raisin-oatmeal-pancakes | creamed-eggs |
| german-chocolate-fudge | porchetta-italiana | reveillon-tourtierre |
| easy-creamy-vanilla-fudge | never-fail-pie-crust-i | reveillon-tourtiere |
| chocolate-orange-fudge | thai-chicken-pizza | pina-colada-rum-cake |
| million-dollar-fudge | whipped-mashed-potatoes | darras-famous-tuna-waldorf-salad-sandwich-filling |

| Group 1: Taste 4 | Group 1: Taste 5 | Group 1: Taste 6 |
|---|---|---|
| breakfast-wellington | cheese-soup-iv | janets-rich-banana-bread |
| grilled-chicken-salad-with-seasonal-fruit | strawberry-yogurt-pie-i | chess-pie-i |
| festive-cherrettes | squash-soup | roasted-red-pepper-oil |
| poppy-seed-bread-iii | cranbury-walnut-relish | vegetarian-nori-rolls |
| herbed-chicken-in-pastry | chocolate-chip-bars | razz-ma-tazz-bars |

| Group 2: Taste 1 | Group 2: Taste 2 | Group 2: Taste 3 |
|---|---|---|
| cottage-dill-bread | spinach-noodle-casserole | fabulous-and-easy-guacamole |
| braided-easter-egg-bread | sauerkraut-cake | chicken-pot-pie-ii |
| kats-alfredo-potatoes | pig-burger | pecan-lace-cookies-i |
| pineapple-pie-iv | black-bottom-cupcakes-i | a-good-barbeque-sauce |
| pan-de-muertos-mexican-bread-of-the-dead | really-raspberry | orange-drop-cookies-iii |

| Group 2: Taste 4 | Group 2: Taste 5 | Group 2: Taste 6 |
|---|---|---|
| not-too–dry-shoofly-pie | healthier-pie-crust | blueberry-bread-i |
| classic-crab-and-shrimp-salad | stuffed-olives | susies-stuffies |
| 25-minute-chicken-and-noodles | coconut-bread-i | preachers-delight |
| asparagus-casserole-i | dr-michaels-yeasted-cornbread | hot-crab-dip |
| winter-fruit-salad-with-lemon-poppyseed-dressing | kims-chicken-alfredo | shaker-lemon-pie |

| Group 2: Taste 7 | Group 2: Taste 8 | Group 2: Taste 9 |
|---|---|---|
| corn-pudding-ii | irish-eggs | southern-style-meat-sauce |
| bolognese-sauce | doggie-treats-i | orange-johnny-cake |
| baked-french-toast | easiest-oven-baked-chicken | tomato-sauce-with-sausage |
| moms-pie-crust | deep-dish-brownies | broccoli-and-carrot-lasagna |

| | | |
|---|---|---|
| red-velvet-cake-iii | walnut-biscotti | bobs-awesome-lasagna |

| **Group 2: Taste 10** | **Group 3: Taste 1** | **Group 3: Taste 2** |
|---|---|---|
| garlic-potatoes-gratin | spicy-pasta-salad | dill-lemon-chicken-pasta-salad |
| matzo-apple-kugel | butterfly-frosting | sex-on-the-beach-ii |
| moms-skillet-chicken-and-rice | light-southwestern-tomato-pasta | italian-wedding-soup-ii |
| salt-rising-bread | spaghetti-torte | acorn-squash |
| fruity-party-punch | festive-holiday-bark | andalusian-gazpacho |

| **Group 3: Taste 3** | **Group 3: Taste 4** | **Group 4: Taste 1** |
|---|---|---|
| three-bean-salad | garlic-pita-bread-bites | ricardos-pizza-crust |
| tomato-salsa-without-onions | glazed-pork-chops | honey-coconut-salmon |
| quick-black-beans-and-rice | grandmas-heavenly-onions | jamaican-jerked-chicken |
| apple-harvest-pound-cake-with-caramel-glaze | rhubarb-bread-i | potato-soup-xi |
| chocolate-revel-bars | chinese-roasted-chicken | creamy-chicken-and-wild-rice-soup |

| **Group 4: Taste 2** | **Group 4: Taste 3** | **Group 4: Taste 4** |
|---|---|---|
| banana-caramel-pie-i | kelaguen-chicken | moms-cucumbers |
| herb-and-chicken-pasta | owens-veggie-stir-fry | the-best-veggie-sandwich |
| cracker-jack-cookies-ii | chocolate-covered-strawberries | valeries-cherry-choco-chip-cake |
| jeannes-slow-cooker-spaghetti-sauce | christmas-fudge | hawaiian-spareribs |
| garlic-mashed-potatoes | sour-cream-chocolate-chip-cake-ii | easy-peach-crisp-i |

| **Group 4: Taste 5** | **Group 4: Taste 6** | **Group 4: Taste 7** |
|---|---|---|
| uncooked-banana-pudding | lemon-meringue-pie-iii | low-fat-apple-bran-muffins |
| blueberry-crisp-ii | taco-soup-iv | cream-of-fresh-asparagus-soup-i |
| balalaika | glazed-pork-chops | tiramisu-i |
| hot-crab-dip | cheesy-pea-pasta | chocolate-chip-muffins |
| fajita-marinade-ii | dorseys-cream-of-crab-soup | brunch-peanut-butter-muffins |

## 15.3. Supplementary Table 3: Semi-supervised DPP-LDA

| Course: appetizer | Course: beverage | Course: bread |
|---|---|---|
| light-wheat-rolls | - | mamas-italian-wedding-soup |
| stained-glass-sugar-hearts | sesame-beef | oriental-hot-n-sour-soup |
| welsh-rarebit-ii | cream-of-chicken-soup | strawberries-and-cream-shake |
| strawberry-rhubarb-coffee-cake | fruit-chiffon-pie | asian-chicken-noodle-salad |
| kalacs-hungarian-cinnamon-swirl-bread | - | just-peachy-peanutty-pork |

| Course: condiment | Course: dessert | Course: main dish |
|---|---|---|
| rhubarb-cake-ii | toffee-bars-by-eagle-brand | bountiful-garden-zucchini-enchiladas |
| chicken-with-asparagus-and-roasted-red-peppers | sidecar | biscochitos-i |
| chicken-encilantrada | fry-bread-i | oven-barbecued-chicken |
| potato-penne-soup | oven-barbecued-pork | chocolate-dream-pie |
| new-england-crab-cakes | chicken-tetrazzini-i | crawfish-fettuccine-i |

| Course: salad | Course: sauce | Course: snack |
|---|---|---|
| concord-grape-pie-ii | outrageous-caesar-salad | - |
| english-muffins | chip-dip | peanut-maple-popcorn |
| sour-cream-lemon-pie | blueberry-cornmeal-muffins | - |
| grape-jelly | pizza-casserole | lauries-cheesy-tomato-pasta |
| - | raspberry-lemonade-pie | heavenly-white-cake |

| Course: soup | Dietary: vegetarian | Dietary: fish |
|---|---|---|
| new-york-cheesecake-iii | moms-best-pork-chops | nancys-chicken-in-puff-pastry |
| sensational-chicken-noodle-soup | blue-cheese-cheesecake | apricot-almond-rugalach |
| yumkins | peachy-chicken | faux-reo-filling |
| spinach-and-pasta-shells | chocolate-macaroons-i | chicken-nuggets |
| italian-nachos-restaurant-style | tangy-egg-salad-spread | pork-for-tamales |

| Dietary: poultry | Dietary: meat | Method: baked |
|---|---|---|
| easy-nacho-dip | tabbouleh-iii | doreens-teriyaki-steak-marinade |
| butterscotch-chocolate-cake | bechamel-sauce | italian-style-soup |
| parmesan-and-basil-chicken-salad | grated-potato-dumplings | cashew-caramel-bars |

|  |  |  |
|---|---|---|
| white-peach-sangria | crunchy-cinnamon-raisin-buns | tea-cookies-ii |
| pumpkin-pudding-ii | one-pot-pasta | tomato-sauerkraut-pork |

| **Method: barbeque** | **Method: boiled** | **Method: fried** |
|---|---|---|
| - | - | cauliflower-cheese-soup-ii |
| molasses-sugar-cookies | rickyrootbeer | spicy-crispy-beef |
| cream-of-broccoli-soup-ii | stuffed-french-toast-ii | artichoke-and-mussel-bisque |
| rhubarb-cake-iii | chinese-chicken-and-potato-soup | crushed-pineapple-fruitcake |
| curry-beef-soup | walnut-pumpkin-pie | grandman-pork-chops-and-rice |

| **Method: simmered** | **Method: frozen** | **Cuisine: asian** |
|---|---|---|
| turtle-bread | pickled-hot-peppers | oven-bbq |
| chicken-salad | famous-pork-chops | easy-sausage-strata |
| very-berry-muffins | sweet-potato-chimichangas | marinated-salad |
| shelias-dip | festive-fruit-squares | blueberry-zucchini-bread |
| low-sugar-date-cake | oatmeal-molasses-bread | easy-egg-tarts |

| **Cuisine: european** | **Cuisine: italian** | **Cuisine: jewish** |
|---|---|---|
| - | impossible-pumpkin-pie-ii | caramelized-brussels-sprouts-with-pistachios |
| honey-wheat-sandwich-rolls | tapenade | maple-pecan-shortbread-squares |
| apricot-chicken-iii | marinated-cauliflower | romano-cheese-crisp |
| double-layer-chocolate-peanut-butter-pie | deep-fried-oysters | monkey-bread-iii |
| gazpacho | grilled-hawaiians | old-fashioned-pineapple-upside-down-cake |

| **Cuisine: mexican** | **Cuisine: southern** | **Cuisine: american** |
|---|---|---|
| - | best-burger-ever | one-bowl-chocolate-fudge |
| carolyns-orange-rolls | bacon-cheese-treats | pizza-on-the-grill-i |
| browned-butter-vegetables-with-almonds | apricot-and-peach-fried-pies | paska-bread |
| boston-brown-bread-iii | sloppy-joe-spaghetti | mozzarella-meatball-sandwiches |
| swiss-vegetable-bake | chicken-chili-i | potato-bread-ii |

## 16. Appendix 1: Notation

This table summarizes the notation used in the paper.

| Notation | Domain | Meaning |
|---:|---|---|
| **Unidimensional Taste Modeling / LDA** | | |
| $U$ | $\mathbb{N}$ | number of users |
| $i$ | $\{1,\dots,U\}$ | index of user |
| $R$ | $\mathbb{N}$ | number of recipes |
| $j$ | $\{1,\dots,R\}$ | index of recipe |
| $K$ | $\mathbb{N}$ | number of tastes |
| $k$ | $\{1,\dots,K\}$ | index of taste |
| $N_i$ | $\mathbb{N}$ | number of favorite recipes of user $i$ |
| $n$ | $\{1,\dots,N_i\}$ | index of favorite of user $i$ |
| $\beta_k$ | $\mathbb{R}^R$ | the $k$th taste |
| $\theta_{(i)}$ | $\mathbb{R}^K$ | user $i$'s taste preferences |
| $\mathbf{z}_{(i)n}$ | $1^K$ | indicator vector for the taste that generated user $i$'s $n$th favorite |
| $\mathbf{x}_{(i)n}$ | $1^R$ | indicator vector for user $i$'s $n$th favorite |
| $\eta$ | $\mathbb{R}_+$ | parameter of symmetric Dirichlet prior on each $\beta_k$ |
| $\alpha$ | $\mathbb{R}_+$ | parameter of symmetric Dirichlet prior on each $\theta_{(i)}$ |
| $\mathbf{X}$ | | set of all users' favorites |
| **Multidimensional Taste Modeling** | | |
| $G$ | $\mathbb{N}$ | number of taste groups |
| $M_g$ | $\mathbb{N}$ | number of tastes in $g$th group |
| $m$ | $\{1,\dots,M_g\}$ | index of taste in $g$th group |
| $\beta_m^{(g)}$ | $\mathbb{R}^R$ | the $m$th taste in group $g$ |
| $\beta^{(g)}$ | $\mathbb{R}^{M_g\times R}$ | matrix of all tastes in group $g$ |
| $\beta$ | | the set of all topic groups $\{\beta^{(k)}\}_{g=1}^G$ |
| $\Lambda^{(g)}$ | $\mathbb{R}^{R\times R}$ | recipe-similarity (Gram) matrix for group $g$ |
| $S\left(\beta^{(g_1)},\beta^{(g_2)}\right)$ | $\mathbb{R}^{M_{g_1}\times R}\times\mathbb{R}^{M_{g_2}\times R}\to[0,1]$ | similarity between topic groups $g_1$ and $g_2$ |
| $R\left(\Lambda^{(g_1)},\Lambda^{(g_2)}\right)$ | $\mathbb{R}^{R\times R}\times\mathbb{R}^{R\times R}\to[0,1]$ | similarity between Gram matrices for $g_1$ and $g_2$ |
| $\pi\left(\beta_m^{(g)}\right)$ | $\mathbb{R}^R\to[0,1]$ | "quality" (marginal probability) of topic $m$ in group $g$ |
| $p$ | $\mathbb{R}_+$ | parameter for similarity function $R$ |
| $\lambda$ | $\mathbb{R}_+$ | strength of diversity penalty |
| **Multidimensional Taste Modeling: Inference e-step** | | |
| $\mathbf{Z}$ | | the set of all hidden variables $z_{(i)n}$ |
| $\Theta$ | | the set of all users' taste preferences $\theta_{(i)}$ |
| $q(\theta,\mathbf{Z})$ | - | the variational approximation to the posterior distribution of the hidden variables given the observed variables |
| $\nu$ | could vary | a vector of variational free parameters |
| $\mathscr{L}$ | domain$(\nu)\to(-\infty,0)$ | ELBO function |
| $q_\theta(\theta)$ | domain$(\theta)\to[0,1]$ | factorized variational approximation for $\theta$ |

| | | |
|---|---|---|
| $\gamma$ | $\mathbb{R}^K$ | free parameter for $q_\theta$ |
| $q_{z_n}(\theta)$ | $\text{domain}(z_n) \rightarrow [0,1]$ | factorized variational approximation for $z_n$ |
| $\phi_n$ | $\mathbb{R}^K$ | free parameter for $q_{\mathbf{z}_n}$ |
| **Multidimensional Taste Modeling: Inference m-step** | | |
| $w_{mj}^{(g)}$ | $\mathbb{R}$ | unconstrained version of $\beta_{mj}^{(g)}$ |
| $\tau$ | $\mathbb{R}_+$ | gradient ascent step size |
| $\Pi$ | $\mathbb{R}^{G\times G}$ | quality function $\pi$ in matrix form |
| $S_\beta$ | $\mathbb{R}^{G\times G}$ | similarity function $S$ in matrix form |
| **Semi-Supervision** | | |
| $\mathbb{L}$ | $\subseteq \{1,\dots,R\}$ | set of labeled recipes |
| $\mathbf{y}_j^{(g)}$ | $1^K$ | label of recipe $j$ along dimension $g$ |
| $\eta_m^{(g)}$ | $\mathbb{R}^R$ | parameter vector for asymetric Dirichlet prior on taste $m$ in group $g$ |
| $\gamma$ | $\mathbb{R}_+$ | concentration of the tastes |
| $\delta$ | $\mathbb{R}_+$ | strength of semi-supervision |

## 17. Appendix 2: Determinantal Point Processes and Markov Random Fields

Here I will briefly compare the determinantal point process with one of its most obvious alternatives: a pairwise, binary Markov random field (MRF). An MRF could also be used to define a probability distribution over "diverse" subsets of a ground set. The tradeoff is a DPP is more computationally tractable than an MRF, but less expressive.

A pairwise, binary Markov random field is a probability distribution over a binary vector $\mathbf{y}$ of length $N$. It is parameterized by a *node potential* $w_i$ for each element and an *edge potential* $w_{ij}$ between each pair of elements. (These parameters are roughly analogous to the on-diagonal and off-diagonal entries in the kernel for a DPP.) The probability of any $\mathbf{y}$ is:

$$p(\mathbf{y}) = \frac{1}{Z}\exp\left\{\sum_i w_i y_i + \sum_{i<j} w_{ij} y_i y_j\right\} \tag{94}$$

where $Z$ is a normalization constant. The higher the node potential $w_i$, the more likely is element $i$ to be included. The higher the edge potential $w_{ij}$, the more likely are elements $i$ and $j$ to be included together.

An MRF can express more complex relationships between the elements than can a DPP. For one, a DPP can only model *anticorrelation* — scenarios where the inclusion of one element makes the inclusion of another element *less* likely — while an MRF can model *positive* correlations between elements $i$ and $j$ by setting $w_{ij} > 0$. In fact, an MRF that models *only* positive correlations (i.e. $w_{ij} > 0$ for all $i, j$), also known as an *associative Markov random field*, possesses the nice property that inference is tractable (CITE?). However, when some of the edge potentials $w_{ij}$ are negative (corresponding to the "repulsive" relationships one can model with a DPP), inference is not tractable for general (i.e. non-tree) graphs.

So, a DPP is tractable, whereas a "diversity"-inducing MRF (i.e. one with negative edge potentials) is not. However, even MRFs with all negative edge potentials can model complex repulsive

relationships that a DPP cannot. Specifically, an MRF with all nonpositive edge potentials can, surprisingly, model certain kinds of positive correlations. Kulesza and Taskar give the example of a 3-node MRF with edge potentials $w_{12}, w_{13} < 0$ and $w_{23} = 0$; elements 2 and 3 are positively correlated because of their mutual anticorrelation from element 1. In contrast, the diversity metric encoded by a DPP must be global and consistent, due to the positive semidefiniteness constraint on the kernel. The diversities between any three elements $i$, $j$, and $k$ in the ground set of a DPP must satisfy a triangle inequality: $i$ and $k$ must be at least as dissimilar as $i$ and $j$ plus $j$ and $k$.
In sum, we have seen that MRFs are more expressive than DPPs, but DPPs are more tractable for inference.

## References

[1] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[2] W. Buntine and A. Jakulin, "Discrete component analysis," 2006. [Online]. Available: http://arXiv:math/0604410

[3] G. Hinton, "Csc 2515 lecture notes on optimization," 2007.

[4] J. Jagarlamudi, H. Daume III, and R. Udupa, "Incorporating lexical priors into topic models," *Conference of the European Chapter of the Association for Computational Linguistics*, vol. 13, pp. 204–213, 2015.

[5] T. Jebara, R. Kondor, and A. Howard, "Probability product kernels," *Journal of Machine Learning Resarch*, vol. 5, pp. 819–844, 2004.

[6] Y. Jiang and A. Saxena, "Discovering different types of topics: Factored topic models," in *In IJCAI*, 2013.

[7] A. Kulesza and B. Taskar, "Determinantal point processes for machine learning," *Foundations and Trends in Machine Learning*, vol. 5, no. 2-3, pp. 123–286, 2012.

[8] D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in Neural Information Processing Systems*, vol. 13, 2001.

[9] O. Macchi, "The coincidence approach to stochastic point processes," *Advances in Applied Probability*, vol. 7, no. 1, p. 83, 1975.

[10] K. P. Murphy, *Machine learning: A Probabilistic Perspective*. MIT Press, 2012.

[11] M. J. Paul and M. Dresde, *Advances in Neural Information Processing Systems*, vol. 24, 2012.

[12] K. Petersen and M. Petersen, *The Matrix Cookbook*, 2006.

[13] M. P. Roxana Girju, "A two-dimensional topic-aspect model for discovering multi-faceted topics," *AAAI Conference on Artificial Intelligence*, vol. 24, pp. 545–550, 2010.

[14] J. Zou and R. Adams, "Priors for diversity in generative latent variable models," *Advances in Neural Information Processing Sysems*, vol. 24, 2012.