Tractable Learning for Structured Probability Spaces: A Case Study in Learning Preference Distributions

Arthur Choi and **Guy Van den Broeck** and **Adnan Darwiche**
Computer Science Department
University of California, Los Angeles
{aychoi,guyvdb,darwiche}@cs.ucla.edu

Abstract

Probabilistic sentential decision diagrams (PSDDs) are a tractable representation of structured probability spaces, which are characterized by complex logical constraints on what constitutes a possible world. We develop general-purpose techniques for probabilistic reasoning and learning with PSDDs, allowing one to compute the probabilities of arbitrary logical formulas and to learn PSDDs from incomplete data. We illustrate the effectiveness of these techniques in the context of learning preference distributions, to which considerable work has been devoted in the past. We show, analytically and empirically, that our proposed framework is general enough to support diverse and complex data and query types. In particular, we show that it can learn maximum-likelihood models from partial rankings, pairwise preferences, and arbitrary preference constraints. Moreover, we show that it can efficiently answer many queries exactly, from expected and most likely rankings, to the probability of pairwise preferences, and diversified recommendations. This case study illustrates the effectiveness and flexibility of the developed PSDD framework as a domain-independent tool for learning and reasoning with structured probability spaces.

1 Introduction

One of the long-standing goals of AI is to combine logic and probability in a coherent framework. A recent direction of integration is towards probability distributions over *structured spaces*. In graphical models, the probability space is the Cartesian product of assignments to individual random variables, corresponding to the rows of a joint probability table. A structured probability space instead consists of complex objects, such as total and partial orders, trees, DAGs, molecules, pedigrees, product configurations, maps, plans, etc. Our goal is to develop a general-purpose framework for representing, reasoning with, and learning probability distributions over structured spaces. These tasks so far required special-purpose algorithms. This is in stark contrast with general-purpose techniques for unstructured probability spaces, such as Bayesian networks.

We leverage a recently proposed tractable representation of structured probability distributions, called *probabilistic sentential decision diagrams* (PSDDs) [Kisa *et al.*, 2014]. As in the unstructured case, structured objects are conveniently represented by assignments to a set of variables. However, in a structured space, not every assignment represents a valid object. Hence, probability distributions over such objects are not easily captured by the rows of a joint probability table. Instead, we encode the structure explicitly in propositional logic. Given this formal description, the next challenge is to represent and reason with probability distributions over that space. In PSDDs, such a distribution is captured by a set of local probability distribution over the decisions in the diagram. Finally, we seek to learn these distributions from data.

We develop our ideas in the context of a specific structured space: preference distributions. Preference learning is studied in a broad range of fields, from recommender systems [Karatzoglou *et al.*, 2013], web search [Dwork *et al.*, 2001] and information retrieval [Liu, 2009; Burges, 2010], to supervised learning [Hüllermeier *et al.*, 2008; Vembu and Gärtner, 2011], natural language [Collins and Koo, 2005], social choice [Young, 1995], statistics [Marden, 1996], and psychology [Doignon *et al.*, 2004]. It has given rise to a multitude of different techniques. We follow the *probabilistic* approach, where preferences are generated from a distribution over the set of all rankings, which can express complex correlations and noise. Despite the simplifying assumptions that underly existing representations, *learning* preference distributions remains computationally hard [Meila *et al.*, 2007]. Moreover, ranking data can be very heterogeneous [Busse *et al.*, 2007], consisting of partial and total rankings, ranking with ties, top/bottom items, and pairwise preferences. Different data types and model assumptions often require a new, special-purpose learning and approximation algorithm (e.g, Hunter [2004], Lebanon and Mao [2007], Guiver and Snelson [2009], and Liu [2009]). Finally, once a model is learned, *inference* is typically limited to sampling, which can only be effective for certain types of basic queries.

In this paper, we employ PSDDs for inducing distributions over the space of all total rankings, and the space of all partial rankings (rankings with ties). We extend the PSDD framework to answer arbitrary queries specified using propositional logic. Our proposed approach represents such queries using a structured logical object, called a *sentential decision dia-*

gram (SDD) [Darwiche, 2011]. In particular, given a PSDD model and an SDD query, we propose an algorithm for efficiently and exactly computing the query probability or most-likely explanations (MPE). In the case of preference distributions, this allows one on to pose arbitrary ranking queries, involving pairwise preferences, partial orders, or any other constraint on the ranking. We can handle queries for diversified recommendations, which are not within the scope of most existing models. These queries seek most-likely or expected rankings subject to constraints that enforce diversity, interestingness, or remove redundancy. Our final contribution is in showing that PSDDs can be learned, efficiently, from incomplete data using the EM algorithm [Dempster *et al.*, 1977; Lauritzen, 1995]. Our proposed algorithm applies to a general type of incomplete datasets, allowing one to learn distributions from datasets based on arbitrary constraints.

We start by giving the necessary background on structured spaces and their distributions. We then introduce the basic algorithms for inference and learning. We next show an experimental evaluation on two standard datasets, where we also illustrate diversified recommendations. We finally conclude by discussing some more related work.

2 Representing Structured Spaces

Consider a set of Boolean variables $X_1, \ldots, X_n$. We will use the term *unstructured space* to refer to the 2^n instantiations of these variables. We will also use the term *structured space* to refer to a subset of these instantiations, which is determined by some complex, application-specific criteria.

To provide a concrete example of a structured space, consider the Boolean variables A_{ij} for $i, j \in \{1, \ldots, n\}$. Here, the index i represents an *item* and the index j represents its *position* in a total ranking of n items. The unstructured space consists of the 2^{n^2} instantiations of our n^2 Boolean variables. A structured space of interest consists of the subset of instantiations that correspond to *total rankings* over n items. The size of this structured space is only $n!$ as the remaining instantiations do not correspond to valid, total rankings (e.g., an instantiation that places two items in the same position, or one item in two different positions).

Many applications require probability distributions over structured spaces, and our interest in this paper is in one such application: reasoning about user preferences. The standard approach for dealing with this application alludes to specialized distributions, such as the Mallows [1957] model, which assumes a *central* total ranking σ, with probabilities of other rankings σ' decreasing as their distance from σ increases.

The approach we shall utilize, however, is quite different as it allows one to induce distributions over arbitrary structured spaces (e.g., total rankings, partial rankings, graph structures, etc.). According to this approach, one defines the structured space using a Boolean formula, whose models induce the space. Considering our running example, and assuming that $n = 3$, we can define the structured space using two types of Boolean constraints:

– Each item i is assigned to exactly one position, leading to three constraints for $i \in \{1, 2, 3\}$:

$$(A_{i1} \wedge \neg A_{i2} \wedge \neg A_{i3}) \vee (\neg A_{i1} \wedge A_{i2} \wedge \neg A_{i3}) \vee (\neg A_{i1} \wedge \neg A_{i2} \wedge A_{i3}).$$

– Each position j is assigned exactly one item, leading to three constraints for $j \in \{1, 2, 3\}$:

$$(A_{1j} \wedge \neg A_{2j} \wedge \neg A_{3j}) \vee (\neg A_{1j} \wedge A_{2j} \wedge \neg A_{3j}) \vee (\neg A_{1j} \wedge \neg A_{2j} \wedge A_{3j}).$$

The Boolean formula defining the structured space will then correspond to a conjunction of these six constraints (more generally, $2n$ constraints).

To consider an even more complex example, let us consider the structured space of *partial rankings.* For defining this space, we will use Boolean variables A_{ij} with $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, t\}$. Here, the index i represents an *item,* and the index j represents the *tier* it is assigned to. The semantics is that we would prefer an item that appears in a higher tier (smaller j) over one appearing in a lower tier (larger j), but we will not distinguish between items within a tier. The sizes of tiers can vary as well. For example, the first tier can represent a single best item, the first and second tiers can represent the top-2, the first three tiers can represent the top-4, and so on. This type of partial ranking is analogous to one that would be obtained from a single-elimination tournament, where a 1st and 2nd place team is determined (the finals), but where the 3rd and 4th places teams may not be distinguished (losers of the semi-finals). We can also define this structured space using a Boolean formula, which is a conjunction of two types of constraints. The first type of constraints ensures that *each item i is assigned to exactly one tier.* The second type of constraints ensures that *each tier j has exactly m_j items.*

3 Compiling Structured Spaces

We will describe in the next section the approach we shall use for inducing probability distributions over structured spaces. This approach requires the Boolean formula defining the structured space to be in a tractable form, known as a *sentential decision diagram* (SDD) [Darwiche, 2011].[1] Figure 1 depicts an example SDD for the Boolean formula

$$(A \Leftrightarrow B) \vee ((A \Leftrightarrow \neg B) \wedge C).$$

A circle in an SDD is called a *decision node* and its children (paired boxes) are called *elements.* Literals and constants ($\top$ and $\bot$) are called *terminal nodes.* For element (p, s), p is called a *prime* and s is called a *sub.* A decision node n with elements $(p_1, s_1), \ldots, (p_n, s_n)$ is interpreted as $(p_1 \wedge s_1) \vee \ldots \vee (p_n \wedge s_n)$. SDDs satisfy some strong properties that make them tractable for certain tasks, such as probabilistic reasoning. For example, the prime and sub of an element do not share variables. Moreover, if $(p_1, s_1), \ldots, (p_n, s_n)$ are the elements of a decision node, then primes $p_1, \ldots, p_n$ must

[1] SDDs generalize OBDDs [Bryant, 1986] by branching on arbitrary sentences instead of literals.

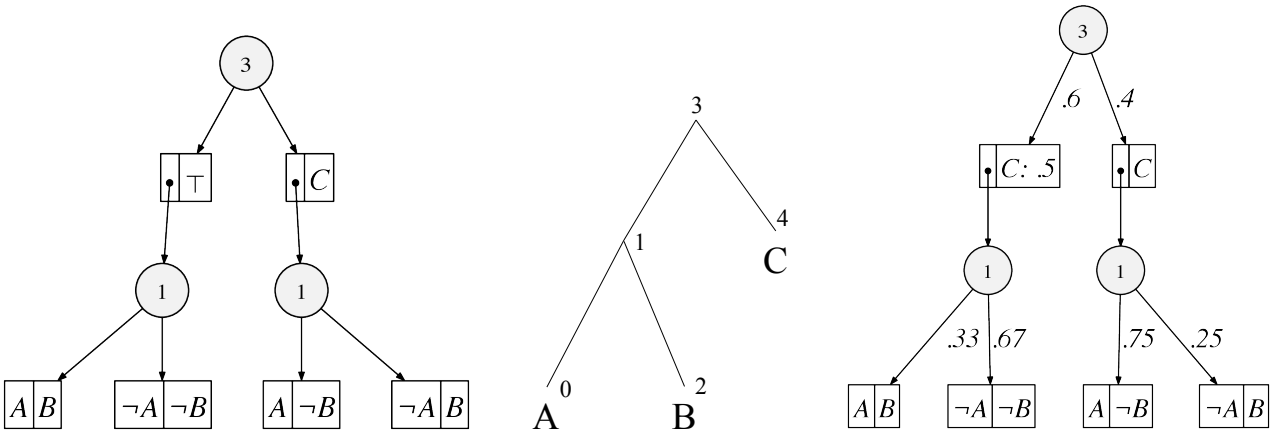

Figure 1: An SDD, a vtree, and a PSDD for the Boolean formula $(A \Leftrightarrow B) \vee ((A \Leftrightarrow \neg B) \wedge C)$.

form a partition ($p_i \neq \mathsf{false}$, $p_i \wedge p_j = \mathsf{false}$ for $i \neq j$, and $p_1 \vee \ldots \vee p_n = \mathsf{true}$).

Each SDD is *normalized* for some *vtree*: a binary tree whose leaves are in one-to-one correspondence with the formula variables. The SDD of a Boolean formula is unique once a vtree is fixed. Figure 1 depicts an SDD and the vtree it is normalized for. More specifically, each SDD node is normalized for some vtree node. The root SDD node is normalized for the root vtree node. If a decision SDD node is normalized for a vtree node v, then its primes are normalized for the left child of v, and its subs are normalized for the right child of v. As a result, terminal SDD nodes are normalized for leaf vtree nodes. Figure 1 labels decision SDD nodes with the vtree nodes they are normalized for.

In our experiments, we used the SDD package[2] to compile Boolean formulas, corresponding to total and partial ranking spaces, into SDDs.[3] For partial ranking spaces, we report results on $n = 64$ items and $t = 4$ tiers, where each tier grows in size by a factor of k, i.e., top-1, top-k, top-k^2 and top-k^3 items. To give a more comprehensive sense of space sizes, the following table shows the sizes of spaces and those of their SDD compilations, for various n and k, fixing $t = 4$.

		Size		
n	k	SDD	Structured Space	Unstructured Space
8	2	443	840	$1.84 \cdot 10^{19}$
27	3	4,114	$1.18 \cdot 10^{9}$	$2.82 \cdot 10^{219}$
64	4	23,497	$3.56 \cdot 10^{18}$	$1.04 \cdot 10^{1233}$
125	5	94,616	$3.45 \cdot 10^{31}$	$3.92 \cdot 10^{4703}$
216	6	297,295	$1.57 \cdot 10^{48}$	$7.16 \cdot 10^{14044}$
343	7	781,918	$4.57 \cdot 10^{68}$	$7.55 \cdot 10^{35415}$

The size of an SDD is obtained by summing the sizes of its decision nodes [Darwiche, 2011]. As we shall see in the next section, this size corresponds roughly to the number of parameters needed to induce a distribution over the SDD models. For example, for $n = 64$ items, and tier growth rate $k = 4$, one needs about $23,497$ parameters to induce a distribution over a structured space of size $3.56 \cdot 10^{18}$.

The SDDs for total rankings did not scale as well as they have a number of nodes that grows exponentially in the number of items n (yet grows more slowly than the factorial function). Hence, we were able to represent spaces for only a moderate number of items (about 20), which is enough for certain datasets, such as the commonly used sushi dataset (consisting of 10 items). We will evaluate encodings for both total rankings and partial rankings (which scale better than total rankings), in our experiments.

4 Distributions over Structured Spaces

To induce a distribution over a structured space, we use *probabilistic sentential decision diagrams* (PSDDs) [Kisa *et al.*, 2014]. According to this approach, the Boolean formula defining the structured space is first compiled into a normalized SDD. The SDD is then parameterized to induce a probability distribution over its models. Figure 1 depicts an SDD and one of its parameterizations (PSDD).

An SDD is parameterized by providing distributions for its decision nodes and its terminal nodes, $\top$. A decision SDD node $n = (p_1, s_1), \ldots, (p_k, s_k)$ is parametrized using a distribution $(\theta_1, \ldots, \theta_k)$, which leads to a decision PSDD node $(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$. The parameters θ_i are notated on the edges outgoing from a decision node; see Figure 1. A terminal SDD node $n = \top$ is parameterized by a distribution $(\theta, 1 - \theta)$, leading to terminal PSDD node $X : \theta$. Here, X is the leaf vtree node that n is normalized for; see Figure 1.

We will identify an SDD/PSDD with its root node r. Moreover, if n is a PSDD node, then $[n]$ will denote the SDD that n parameterizes. That is, while n represents a probability distribution, $[n]$ represents a Boolean formula. According to PSDD semantics, every PSDD node n, not just the root r, induces a distribution Pr_n over the models of SDD $[n]$.

The semantics of PSDDs is based on the notion of a *context*, γ_n, for PSDD node n. Intuitively, this is a Boolean formula that captures all variable instantiations under which the

[2] Available at `http://reasoning.cs.ucla.edu/sdd/`

[3] We did not use dynamic vtree search as provided by the SDD package. Instead, we used a static vtree, based on preliminary experimentation. Basically, for each position j, we created a right-linear vtree over the variables A_{ij}. We then composed these right-linear vtrees together, also using a right-linear structure.

decision diagram will branch to node n. We will not describe the distribution Pr_r induced by a PSDD r, but will stress the following local semantics of PSDD parameters. For a decision PSDD node $n = ((p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$, we have $Pr_r(p_i|\gamma_n) = \theta_i$. Moreover, for a terminal SDD node $n = X{:}\theta$, we have $Pr_r(X|\gamma_n) = \theta$. This local semantics of PSDD parameters is the key reason behind their many well-behaved properties (e.g., the existence of closed-form maximum-likelihood parameter estimates for complete data). We defer the reader to Kisa *et al.* [2014] for a thorough exposition of PSDD syntax, semantics and properties.

5 Querying with Constraints

Suppose now that we have a probability distribution Pr and we want to compute the probability of a Boolean formula α. For example, Pr can be a distribution over preferences in movies, and α could represent "a comedy appears as one of the top-10 highest ranked movies." We may also be interested in $Pr(. \mid \alpha)$, which is the conditional distribution over rankings, assuming that a comedy appears in the top-10.

The ability to reason about arbitrary constraints is a powerful one. For example, it allows one to diversify recommendations in preference-based reasoning, as we discuss later. In most representations, this ability is either not present, or it is intractable. For example, in a Bayesian network, one can use the method of virtual evidence [Pearl, 1988; Mateescu and Dechter, 2008], to represent a logical constraint. However, this will in general lead to a highly-connected network, making inference intractable.

A key technical contribution of this paper is an observation that the probability of a Boolean formula can be computed efficiently, in a distribution induced by a PSDD, given that the formula is represented by an SDD with the same vtree as the PSDD.[4]

Theorem 1. *Suppose we have a PSDD n with distribution Pr_n and size s_n, and a Boolean formula α represented by an SDD m of size s_m. If SDD m has the same vtree as PSDD n, then $Pr_n(\alpha)$ can be computed in time $O(s_n s_m)$.*[5]

Suppose that PSDD n has elements (p_i, s_i, θ_i) and SDD α has elements (q_j, r_j). Our approach for computing the probability of α is based on the following recurrence, which is implemented by Algorithm 1:

$$\begin{aligned} Pr_n(\alpha) &= \sum_j Pr_n(q_j \wedge r_j) \\ &= \sum_i \sum_j Pr_{p_i}(q_j) \cdot Pr_{s_i}(r_j) \cdot \theta_i \end{aligned}$$

[4]This assumption, that the SDD and PSDD share the same vtree, is key to the efficiency of computing the probability of a Boolean constraint. Otherwise, the problem becomes NP-hard, which follows from the hardness of conjoining two OBDDs that respect two different orders [Meinel and Theobald, 1998; Darwiche and Marquis, 2002].

[5]This is a loose upper bound. A more accurate bound is $\sum_v s_{v,n} s_{v,m}$, where v is a non-leaf vtree node, $s_{v,m}$ is the size of decision PSDD nodes normalized for v, and $s_{v,m}$ is the size of decision SDD nodes normalized for v.

Algorithm 1 `pr-constraint`(n, m)

input: A PSDD n inducing distribution Pr_n and an SDD m representing Boolean formula α. The PSDD and SDD respect the same vtree.

output: $Pr_n(\alpha)$.

main:

if $(n, m) \in$ `cache` **then**
  return `cache`$[(n, m)]$
else if n is a decision node **then**
  $\rho \leftarrow 0$
  for each element (p_i, s_i, θ_i) in PSDD n **do**
    for each element (q_j, r_j) in SDD m **do**
      $\rho_{\text{left}} \leftarrow$ `pr-constraint`(p_i, q_j)
      $\rho_{\text{right}} \leftarrow$ `pr-constraint`(s_i, r_j)
      $\rho \leftarrow \rho + \rho_{\text{left}} \cdot \rho_{\text{right}} \cdot \theta_i$
  `cache`$[(n, m)] = \rho$
  return ρ
else {`// n and m are terminals`}
  if $[n] \wedge m$ is false **then**
    return 0
  else if $[n] \wedge m$ is true **then**
    return 1
  else if $[n]$ is a literal **then**
    return 0 if $[n] \wedge m$ is false, otherwise 1
  else if n is a terminal $X{:}\theta_X$, and m is a literal **then**
    return θ_X if m is X, or $1 - \theta_X$ if m is $\neg X$

In a second pass on the PSDD and SDD, we can compute the marginals $Pr(X \mid \alpha)$ for each variable X, as well as the probabilities of non-root SDD nodes. This is analogous to the two-pass algorithm given in Kisa *et al.* [2014]. This ability to compute marginals enables an EM algorithm for PSDDs, which we discuss next.

6 Learning from Constraints

Kisa *et al.* [2014] provided an algorithm for learning the parameters of a PSDD given a *complete* dataset. This algorithm identified the (unique) maximum likelihood parameters, and further, in closed-form. We will now present our second, main technical contribution: An efficient algorithm for learning the parameters of a PSDD given an *incomplete* dataset.

We start first with some basic definitions. An instantiation of *all* variables is a *complete example,* while an instantiation of *some* variables is an *example.* There are 2^n distinct complete examples over n variables, and 3^n distinct examples. A complete dataset is a multi-set of complete examples (i.e., a complete example may appear multiple times in a dataset). Moreover, traditionally, an incomplete dataset is defined as a multi-set of examples. One can view an example as a set of complete examples (those consistent with the example). Hence, we will use a more general definition of an incomplete dataset, defined as a multi-set of Boolean formulas, with each formula corresponding to a set of complete examples (those consistent with the formula). This definition is too general to appear practical. However, as we shall show, if we represent these Boolean formulas (i.e., examples) as SDDs, then The-

orem 1 allows us to efficiently learn parameters from such datasets. We will next provide an EM algorithm for this purpose, which has a polytime complexity per iteration.

Given a PSDD structure, and an incomplete dataset specified as a set of SDDs, our goal is to learn the value of each PSDD parameter. More precisely, we wish to learn *maximum likelihood* parameters: ones that maximize the probability of examples in the dataset. Let Pr_θ denote the distribution induced by the PSDD structure and parameters θ. The *log likelihood* of these parameters given a dataset $\mathcal{D}$ is defined as

$$LL(\theta|\mathcal{D}) = \sum_{i=1}^{N} \log Pr_\theta(\mathcal{D}_i),$$

where $\mathcal{D}_i$ ranges over all N examples of our dataset $\mathcal{D}$. Again, each $\mathcal{D}_i$ is an SDD, whose probability $Pr_\theta(\mathcal{D}_i)$ can be computed using Algorithm 1. Our goal is then to find the maximum likelihood parameters

$$\theta^\star = \operatorname*{argmax}_\theta LL(\theta|\mathcal{D}).$$

For an incomplete dataset, finding the globally optimal parameter estimates may not be tractable. Instead, we can more simply search for stationary points of the log likelihood, i.e., points where the gradient is zero (any global optimum is a stationary point, but not vice-versa). Such points are characterized by the following theorem.

Theorem 2. *Let $\mathcal{D}$ be an incomplete dataset, and let θ be the parameters of a corresponding PSDD with distribution Pr_θ. Parameters θ are a stationary point of the log likelihood $LL(\theta|\mathcal{D})$ (subject to normalization constraints on θ) iff for each PSDD node n with context γ_n:*

- *If n is a decision node with elements (p_i, s_i, θ_i), then:*

$$\theta_i = \frac{\sum_{i=1}^{N} Pr_\theta(p_i, \gamma_n \mid \mathcal{D}_i)}{\sum_{i=1}^{N} Pr_\theta(\gamma_n \mid \mathcal{D}_i)}$$

- *If n is a terminal node $X : \theta_X$, then:*

$$\theta_X = \frac{\sum_{i=1}^{N} Pr_\theta(X, \gamma_n \mid \mathcal{D}_i)}{\sum_{i=1}^{N} Pr_\theta(\gamma_n \mid \mathcal{D}_i)}.$$

This theorem suggests an iterative EM algorithm for finding stationary points of the log likelihood. First, we start with some initial parameter estimates θ^0 at iteration $t = 0$. For iteration $t > 0$, we use the above update to compute parameters θ^t given the parameters θ^{t-1} from the previous iteration. When the parameters of one iteration do not change in the next (in practice, up to some threshold), we say that the iterations have converged to a fixed point.

An iteration of the proposed algorithm is implemented by traversing the dataset, while applying Algorithm 1 to each example (i.e., SDD) and the current PSDD. This is sufficient to obtain the quantities needed by the update equations. For a dataset with m distinct examples, the proposed algorithm will therefore apply Algorithm 1 a total of m times per iteration, leading to a polytime complexity per iteration. We can show that the proposed algorithm is indeed an EM algorithm by showing that it obtains the same updates obtained by the following algorithm. First, we "complete" the dataset by replacing each SDD example by the complete examples consistent with it. Second, we compute the probability of each complete example using the current PSDD. Finally, we use the closed forms in [Kisa *et al.*, 2014] to obtain the next parameter estimates based on the completed data. The same proof technique is used in Darwiche [2009] and Koller and Friedman [2009] for justifying the EM algorithm for Bayesian networks.

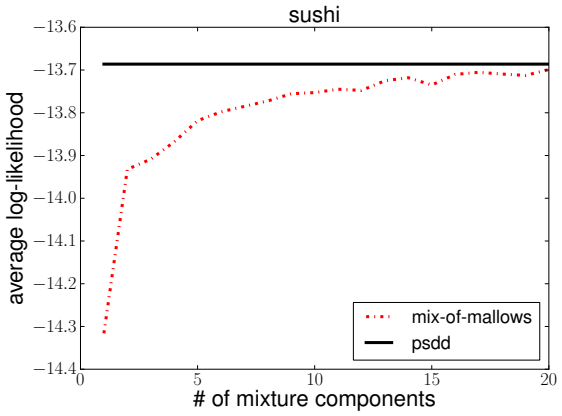

Figure 2: Mallows Mixture Model vs PSDDs

7 Experiments

We now empirically evaluate the PSDD as a model for preference distributions. We first evaluate the encoding for total rankings, where we highlight (a) the quality of learned models and (b) the utility of PSDDs in terms of its query capabilities. We also evaluate the more scalable encoding for partial rankings and further demonstrate (c) the capabilities that PSDDs provide, in particular for diversified recommendations.

7.1 Total Rankings

Consider the `sushi` dataset, which consists of $5,000$ total rankings of 10 different types of sushi [Kamishima, 2003]. We learned a PSDD from the full dataset, using the encoding for total rankings, and analyzed the resulting model. A dataset composed of total rankings is a complete dataset, so we use the closed forms of Kisa *et al.*, 2014 to estimate the maximum likelihood parameters of a PSDD. The corresponding PSDD required $4,097$ (independent) parameters. We further assumed a Dirichlet prior with exponents 2, which corresponds to Laplace smoothing.

Quality of Learned Model. We compared the learned PSDD model for the `sushi` dataset with a learned Mallows mixture model. As performed by Lu and Boutilier [2011], we first split the dataset into a training set of 3,500 instances, and a test set of 1,500 instances. From the training set, we learned a single PSDD model and 10 Mallows mixture models using EM.[6] In Figure 2, we compare the PSDD and Mal-

[6]For the Mallows mixture model, we used the implementation of Lu and Boutilier [2011] with default settings.

lows mixture models in terms of (average) log likelihood of the test set, for an increasing number of mixture components (for the Mallows model). We see that the PSDD dominates the Mallows model for all numbers of mixture components that were evaluated (up to 20, as in Lu and Boutilier [2011]). This suggests that PSDDs are expressive, and better capable of representing distributions such as the one underlying the `sushi` dataset. This is in contrast to the popular Mallows models (and their mixtures), whose underlying assumptions are relatively strong (i.e., that there exists a central ranking).

Query Capabilities. Once a model is learned, PSDDs admit a variety of queries. For example, we can compute the most likely total ranking $\sigma^\star = \text{argmax}_\sigma \, Pr(\sigma)$, which corresponds to the most probable explanation (MPE) in Bayesian networks. We can also compute the expected rank of each item i:

$$E[j] = \sum_j j \cdot Pr(A_{ij}).$$

Moreover, using Theorem 1, it is possible to predict a pairwise preference $i > j$, given a pairwise preference $k > l$. In particular, we are interested in $Pr(i > j \mid k > l)$, which is the probability of a preference $i > j$ given the preference $k > l$. We enumerated all such combinations of i, j, k and l, where i, j, k, l are pairwise distinct, and examined in particular the log odds change, i.e.,

$$\log F(i > j \mid k > l) = \log O(i > j \mid k > l) - \log O(i > j)$$

where we have the odds

$$O(\alpha) = \frac{Pr(\alpha)}{1 - Pr(\alpha)},$$

for an event α; for more on log-odds change, see, e.g., Chan and Darwiche [2005]. A large log-odds change indicates a large increase in a pairwise preference $i > j$ given the preference $k > l$. In our PSDD, the greatest log-odds change was:

$$\log F(\text{egg} > \text{fatty tuna} \mid \text{cucumber roll} > \text{tuna}) = 1.295$$

where the odds of preferring egg to fatty tuna increased from 0.238 to 0.868 when cucumber rolls are preferred to tuna.

7.2 Partial Rankings

Consider now the `movielens` dataset,[7] which consists of over 1 million ratings of approximately 3,900 movies and 6,040 users. Here, each rating is an integer from 1 to 5, in contrast to the total orderings given in the `sushi` dataset. Following Lu and Boutilier [2011], we extracted from these ratings a set of pairwise preferences for each user. In this case, our dataset is incomplete, and we utilized the EM algorithm for PSDDs that we proposed in this paper.

We employ our encoding of partial rankings for this evaluation. In particular, we extract the top 64 most frequently rated movies from the `movielens` dataset, and use the ratings of the resulting 5,891 of 6,040 users who rated at least one of these movies.[8] For each user, we construct an SDD representing their pairwise preferences, based on their ratings. In particular, if a user gives movie i a higher rating than another movie j, we assert a constraint that the movie is at least as highly ranked as the other (or appears in at least as high a tier), i.e., $i \geq j$. We obtain an SDD for each user by conjoining together all such pairwise preferences (the average size of a user SDD was in the tens of thousands). For partial rankings of 64 movies, we assume four tiers representing the top-1, top-5, top-25 and top-64 movies. The corresponding PSDD has $18,711$ (independent) parameters. We finally run EM on the PSDD for 5 iterations. We also use the system of Lu and Boutilier [2011] to learn the Mallows model from pairwise constraints.

Total versus Partial Ranking Models. A comparison with a Mallows models in terms of likelihoods is not possible here as the two models are quite different in scope: one is modeling a distribution over total rankings, while the other is modeling a distribution over partial rankings. We were curious, however, to see the extent to which these models agreed on recommendations. Our general observation has been that they come out close. For example, the central ranking obtained by a Mallows model, and the sorted list of expected rankings given by the PSDD model, agreed on the top 10 movies (but disagreed somewhat on their order). We omit the details of this and other examples here, however, due to space limitations.

Query Capabilities. We will now illustrate some further queries that are permitted by the proposed framework. Most current frameworks for preference distributions cannot handle such queries exactly and efficiently.

First, we may ask for the top-5 movies (by expected tier), given the constraint α: "the highest ranked movie is Star Wars V" (which we encode as an SDD and use as evidence):

1	Star Wars: Episode V - The Empire Strikes Back (1980)
2	Star Wars: Episode IV - A New Hope (1977)
3	Godfather, The (1972)
4	Shawshank Redemption, The (1994)
5	Usual Suspects, The (1995)

We see that another Star Wars movie is also highly ranked. However, if we wanted to use this information to recommend a movie to a user, whose favorite movie was Star Wars V, a recommendation of Star Wars IV would not be particularly useful (as having seen Star Wars V likely implies that one has seen the prequel Star Wars IV as well). We could condition on an additional constraint β, that "no other Star Wars movie appears in the top-5." Going further still, we could assert a third constraint γ, that "at least one comedy appears in the top-5." Conditioning on these three constraints α, β and γ, we obtain the new ranking:

1	Star Wars: Episode V - The Empire Strikes Back (1980)
2	American Beauty (1999)
3	Godfather, The (1972)
4	Usual Suspects, The (1995)
5	Shawshank Redemption, The (1994)

Here, the movie Star Wars IV was replaced by the comedy/drama American Beauty. This provides an illustration of the powerful, and open ended, type of queries permitted by the proposed framework.

[7] Available at `http://grouplens.org/`

[8] Lu and Boutilier [2011] extract the 200 most frequently rated movies. We retain only 64 movies due to time considerations.

8 Conclusion and Related Work

We have introduced a general framework for probabilistic preference learning. Our aim was to provide a different perspective on preference learning, rooted in the recent developments on learning structured probability spaces [Kisa *et al.*, 2014] and the tractable learning paradigm [Domingos and Lowd, 2014]. To support the preference learning application, we extended PSDDs to reason with structured queries and to learn from incomplete structured data. We were particularly motivated by the need to support heterogeneous data, and complex queries, such as diversified recommendations.

Existing work on preference distributions has focused on two representations of historical significance, namely the Plackett-Luce [Plackett, 1975; Luce, 1959] and Mallows [1957] model, and extensions thereof [Fligner and Verducci, 1986; Murphy and Martin, 2003; Meila and Chen, 2010]. Although these models were successfully applied in several applications, they can also be restrictive. As a representation, for example, the Mallows model and its extensions encode the distance from a (small) number of consensus rankings, limiting the number of modes in the distribution. Compact representations of preferences distributions are also pursued by Huang *et al.* [2009] and Huang and Guestrin [2009]. These are sophisticated and dedicated representation of permutations. Moreover, they either do not support tractable exact inference, or all the types of rank data considered here.

The PSDD representation is founded on a long tradition of tractable representations of logical knowledge bases in knowledge compilation [Darwiche and Marquis, 2002]. It is also related to other tractable probabilistic representations, such as sum-product networks, which exploit similar properties for efficient learning [Peharz *et al.*, 2014].

We finally note that the need for diversity (as in diversified recommendations) has been recognized before [McNee *et al.*, 2006; Sanner *et al.*, 2011; He *et al.*, 2012]. Similar observations have also been stated for recommender systems [Rashid *et al.*, 2002] and matching problems [Charlin *et al.*, 2012].

Acknowledgments

We thank Tyler Lu and Craig Boutilier for providing their system for Mallows mixture models, and Scott Sanner for commenting on an earlier draft of this paper. This work was supported by ONR grant #N00014-12-1-0423, NSF grant #IIS-1118122, and the Research Foundation-Flanders (FWO-Vlaanderen). Guy Van den Broeck is also affiliated with KU Leuven, Belgium.

References

[Bryant, 1986] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35:677–691, 1986.

[Burges, 2010] Christopher J.C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. 11:23–581, 2010.

[Busse *et al.*, 2007] Ludwig M Busse, Peter Orbanz, and Joachim M Buhmann. Cluster analysis of heterogeneous rank data. In *Proceedings of ICML*, pages 113–120. ACM, 2007.

[Chan and Darwiche, 2005] Hei Chan and Adnan Darwiche. On the revision of probabilistic beliefs using uncertain evidence. *Artificial Intelligence*, 163:67–90, 2005.

[Charlin *et al.*, 2012] Laurent Charlin, Craig Boutilier, and Richard S Zemel. Active learning for matching problems. In *Proceedings of ICML*, pages 337–344, 2012.

[Collins and Koo, 2005] Michael Collins and Terry Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70, 2005.

[Darwiche and Marquis, 2002] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *JAIR*, 17:229–264, 2002.

[Darwiche, 2009] Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

[Darwiche, 2011] Adnan Darwiche. SDD: A new canonical representation of propositional knowledge bases. In *Proceedings of IJCAI*, pages 819–826, 2011.

[Dempster *et al.*, 1977] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[Doignon *et al.*, 2004] J. Doignon, A. Pekeč, and M. Regenwetter. The repeated insertion model for rankings: Missing link between two subset choice models. *Psychometrika*, 69(1):33–54, 2004.

[Domingos and Lowd, 2014] Pedro Domingos and Daniel Lowd. Learning tractable probabilistic models. UAI Tutorial, 2014.

[Dwork *et al.*, 2001] Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. Rank aggregation methods for the web. In *Proceedings of WWW*, pages 613–622. ACM, 2001.

[Fligner and Verducci, 1986] Michael A Fligner and Joseph S Verducci. Distance based ranking models. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 359–369, 1986.

[Guiver and Snelson, 2009] J. Guiver and E. Snelson. Bayesian inference for plackett-luce ranking models. In *Proc. of ICML*, 2009.

[He *et al.*, 2012] J. He, H. Tong, Q. Mei, and B. Szymanski. GenDeR: A generic diversified ranking algorithm. In *Advances in Neural Information Processing Systems 25*, pages 1142–1150, 2012.

[Huang and Guestrin, 2009] Jonathan Huang and Carlos Guestrin. Riffled independence for ranked data. In *Advances in Neural Information Processing Systems*, pages 799–807, 2009.

[Huang *et al.*, 2009] J. Huang, C. Guestrin, and L. Guibas. Fourier theoretic probabilistic inference over permutations. *The Journal of Machine Learning Research*, 10:997–1070, 2009.

[Hüllermeier *et al.*, 2008] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16):1897–1916, 2008.

[Hunter, 2004] David R Hunter. Mm algorithms for generalized bradley-terry models. *Annals of Statistics*, pages 384–406, 2004.

[Kamishima, 2003] Toshihiro Kamishima. Nantonac collaborative filtering: recommendation based on order responses. In *Proceedings KDD*, pages 583–588, 2003.

[Karatzoglou *et al.*, 2013] Alexandros Karatzoglou, Linas Baltrunas, and Yue Shi. Learning to rank for recommender systems. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 493–494. ACM, 2013.

[Kisa *et al.*, 2014] D. Kisa, G. Van den Broeck, A. Choi, and A. Darwiche. Probabilistic sentential decision diagrams. In *Proceedings of KR*, 2014.

[Koller and Friedman, 2009] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[Lauritzen, 1995] S. L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.

[Lebanon and Mao, 2007] Guy Lebanon and Yi Mao. Non-parametric modeling of partially ranked data. In *Advances in neural information processing systems*, pages 857–864, 2007.

[Liu, 2009] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3), 2009.

[Lu and Boutilier, 2011] Tyler Lu and Craig Boutilier. Learning Mallows models with pairwise preferences. In *Proceedings of ICML*, pages 145–152, 2011.

[Luce, 1959] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Wiley, 1959.

[Mallows, 1957] Colin L. Mallows. Non-null ranking models. *Biometrika*, 1957.

[Marden, 1996] John I. Marden. *Analyzing and modeling rank data*. CRC Press, 1996.

[Mateescu and Dechter, 2008] Robert Mateescu and Rina Dechter. Mixed deterministic and probabilistic networks. *Annals of Mathematics and AI*, 54(1-3):3–51, 2008.

[McNee *et al.*, 2006] Sean M. McNee, John Riedl, and Joseph A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *Proceedings of CHI*, pages 1097–1101, 2006.

[Meila and Chen, 2010] Marina Meila and Harr Chen. Dirichlet process mixtures of generalized mallows models. In *Proceedings of UAI*, 2010.

[Meila *et al.*, 2007] Marina Meila, Kapil Phadnis, Arthur Patterson, and Jeff A. Bilmes. Consensus ranking under the exponential model. In *Proceedings of UAI*, pages 285–294, 2007.

[Meinel and Theobald, 1998] Christoph Meinel and Thorsten Theobald. *Algorithms and Data Structures in VLSI Design: OBDD — Foundations and Applications*. Springer, 1998.

[Murphy and Martin, 2003] Thomas Brendan Murphy and Donal Martin. Mixtures of distance-based models for ranking data. *Computational statistics & data analysis*, 41(3):645–655, 2003.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[Peharz *et al.*, 2014] R. Peharz, R. Gens, and P. Domingos. Learning selective sum-product networks. In *LTPM workshop*, 2014.

[Plackett, 1975] Robin L Plackett. The analysis of permutations. *Applied Statistics*, pages 193–202, 1975.

[Rashid *et al.*, 2002] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of IUI*, pages 127–134, 2002.

[Sanner *et al.*, 2011] S. Sanner, S. Guo, T. Graepel, S. Kharazmi, and S. Karimi. Diverse retrieval via greedy optimization of expected 1-call@k in a latent subtopic relevance model. In *Proceedings of CIKM*, pages 1977–1980, 2011.

[Vembu and Gärtner, 2011] S. Vembu and T. Gärtner. Label ranking algorithms: A survey. In *Preference learning*, pages 45–64. Springer, 2011.

[Young, 1995] Peyton Young. Optimal voting rules. *The Journal of Economic Perspectives*, pages 51–64, 1995.